Aix*Marseille université
Socialement engagée

NNT : 2023AIXM0001

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 2 juillet 2024

Malek SENOUSSI

Classification methods for single cell RNA sequencing data.

Learning with partial annotation and without supervision.

Discipline
Informatique

École doctorale
Mathématique et Informatique - N°184

Laboratoire/Partenaires de recherche
Laboratoire d'Informatique et Systèmes (LIS)
Turing Centre for Living Systems (CENTURI)

Composition du jury

Olivier GANDRILLON DR - CNRS, ENS Lyon	Rapporteur
Guillemette MAROT Prof. - Université de Lille	Rapporteure
Alice CLEYNEN CR - CNRS	Examinatrice
Nathalie VIALANEIX DR - INRAE	Présidente du jury
Thierry ARTIERES Prof. - Centrale Méditerranée	Directeur de thèse
Paul VILLOUTREIX CPJ - INSERM	Co-directeur de thèse

CENTURI
TURING CENTRE
FOR LIVING SYSTEMS

LIS
LABORATOIRE
D'INFORMATIQUE
& DES SYSTÈMES
UMR 7020

Affidavit

Je soussigné, Malek Senoussi, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique de Thierry Artieres et Paul Villoutreix, dans le respect des principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect à la fois de la charte nationale de déontologie des métiers de la recherche et de la charte d'Aix-Marseille Université relative à la lutte contre le plagiat.

Ce travail n'a pas été précédemment soumis en France ou à l'étranger dans une version identique ou similaire à un organisme examinateur.

Fait à Marseille le 6 mai 2024



Cette œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0 International.

Liste de publications et participation aux conférences

Liste des publications et/ou brevets réalisées dans le cadre du projet de thèse :

1. *Classification hiérarchique pour des données transcriptomiques faiblement supervisées*
   Malek Senoussi, Thierry Artières, Paul Villoutreix, CAP 2022
   https://hal.science/hal-04327608
2. *Random walk informed heterogeneity detection reveals how the lymph node conduit network influences T cells collective exploration behavior*
   Solène Song, Malek Senoussi, Paul Escande, Paul Villoutreix, PLoS Computational Biology 2023
   https://doi.org/10.1371/journal.pcbi.1011168
3. *Partial label learning for automated classification of single cell transcriptomic profiles*
   Malek Senoussi, Thierry Artières, Paul Villoutreix, PLoS Computational Biology 2024
   https://doi.org/10.1371/journal.pcbi.1012006

Participation aux conférences et écoles d'été au cours de la période de thèse :

1. Neural Information Processing Systems - 2020
2. Neural Information Processing Systems - 2021
3. Conférence d'APprentissage - 2022
4. Measure-Theoretic Approaches and Optimal Transportation in Statistics - Institut Henri Poincaré Paris - 2022

Résumé

La biologie du développement a pour objet d'étude la façon dont l'expression des gènes est regulée dans l'espace et le temps au sein d'un embryon pour produire un organisme fonctionnel. Le séquençage de l'ARN à l'échelle de la cellule unique est une technologie récente d'acquisition de données. Cette technologie permet de mesurer l'expression de chaque gène pour chaque cellule, ce qui donne des vecteurs transcriptomiques dans un espace de plus de 20 000 dimensions. Un défi majeur est de classifier ces vecteurs transcriptomiques en différents groupes de cellules et d'identifier les relations de lignage entre les cellules. Ces relations sont typiquement encodées par un arbre qui résume toutes les divisions cellulaires qui ont lieu au cours du développement.

Ce travail se concentre sur la question de la classification des vecteurs transcriptomiques au sein d'un arbre de lignage. Plusieurs types d'annotation sont disponibles dans les ensembles de données expérimentales : supervision complète, supervision partielle où chaque exemple a un ensemble d'étiquettes candidates, et aucune supervision. Nous proposons de nouvelles méthodes et des extensions pour plusieurs méthodes de classification multi-classes de l'état de l'art afin de traiter ces différentes situations de supervision.

Mots clés : Statistiques, Apprentissage automatique, Apprentissage profond, Classification hiérarchique, Étiquetage partiel, Découverte de nouvelles classes

Abstract

Developmental biology is concerned with the study of how gene expression is regulated in space and time within an embryo to produce a functional organism. Single-cell RNA sequencing is a recent data acquisition technique. This technology makes it possible to measure the expression of each gene for each cell, resulting in transcriptomic vectors in a space of more than 20,000 dimensions. A major challenge is to classify these transcriptomic vectors into different cell groups and to identify the lineage relationships between cells. These relationships are tipically encoded by a tree that summarizes all the cell divisions that occur during embryogenesis.

This work focuses on the classification of transcriptomic vectors in a lineage tree. Several types of annotation are available in the experimental datasets: full supervision, partial supervision where each example has a set of candidate labels, and no supervision. We propose new methods and extensions for several state-of-the-art multi-class classification methods to address these different supervision situations.

Keywords : Statistics, Machine Learning, Deep Learning, hierarchical classification, Partial label, Novel Class Discovery

Remerciements

Lorsque j'ai commencé à m'intéresser à la recherche, j'étais loin d'imaginer que je finirai par écrire une thèse sur des problèmes de classification. La recherche est un ensemble de surprises et surtout de découvertes très enrichissantes. Je souhaite remercier un ensemble de personnes qui ont contribué de près et de loin à l'élaboration de ce travail.

Je remercie sincèrement mes directeurs de thèse, Paul et Thierry. Je leur suis reconnaissant pour leur encadrement et les échanges durant toutes ces années. Je suis content d'avoir pu travailler avec eux, et je les remercie pour leur patience, leur confiance et leur pédagogie.

Je remercie l'ensemble du jury d'avoir accepté de participer à ce travail. J'adresse mes remerciements à Olivier Grandrillon et Guillemette Marot pour leurs rapports et leurs retours constructifs et attentifs. Merci également à Alice Cleynen et Nathalie Vialaneix d'avoir participé à l'évaluation de ce travail de recherche.

Je remercie toutes les personnes qui m'ont permis de discuter et d'apprendre dans de nombreux domaines. Les échanges ont toujours été riches et motivants. J'ai eu la chance, de discuter de mathématiques, d'informatique, de biologie et de physique avec de nombreuses personnes. Je remercie Cécile Capponi, Erwan Hillon, Sébastien Darses, Vincent Bertrand et Pierre Recouvreux pour leur suivi et soutien durant ces dernières années. Je remercie aussi Solène, Clara et Khulganaa pour nos échanges et collaborations. Je remercie aussi l'ensemble de l'équipe Qarma, particulièrement les autres doctorant.e.s, où on a pu se soutenir dans les moments complexes.

Je remercie bien évidemment, ma famille, Assia, Rachid, Rym et Anis. Je suis reconnaissant pour leur confiance, leur partage et leur soutien. Ils et elles ont su me donner goût aux sciences, l'envie d'étudier, et surtout de partager.

Je remercie enfin, tous mes ami.e.s qui ont été présent.e.s et avec qui j'ai eu la chance de partager de belles années. Je remercie chaleureusement Léonore, Jean, Julien, Colin, Fanchon, Ulysse, Yassine, Jérémie, Anatole, Sarah, Louis, Clément, Amandine, Yzabelle, Sylvain, Élie, Sarah, Neva, Sacha, Thomas, Sami, Elina, Eleonora. Pardonnez-moi pour toutes les personnes que je n'ai pas citées, je vous remercie aussi.

Contents

Affidavit 2

Liste de publications et participation aux conférences 3

Résumé 4

Abstract 5

Remerciements 6

Contents 7

List of Figures 10

List of Tables 11

Liste des acronymes 12

Notation 13

1 Introduction 15
- 1.1 Context . . . 16
  - 1.1.1 Study of the embryonic development . . . 16
  - 1.1.2 Cell state trajectory reconstruction . . . 19
- 1.2 Problematic . . . 25
  - 1.2.1 Cell type classification in single-cell RNA sequencing data . . . 25
  - 1.2.2 Statistical learning for solving various classification tasks . . . 28
- 1.3 Content of the thesis . . . 33

2 Methods 34
- 2.1 Multiclass classification . . . 34
  - 2.1.1 Multi-Class Classification . . . 35
  - 2.1.2 Hierarchical multi-classification . . . 43
- 2.2 Unsupervised Learning . . . 46
  - 2.2.1 *k-means* . . . 47
  - 2.2.2 Gaussian Mixture Model . . . 48
- 2.3 Conclusion . . . 52

3 Modalities and Dataset Presentation **53**
3.1 Simulated Datasets . . . 53
3.1.1 Hierarchical label structure . . . 53
3.1.2 Model parameters . . . 54
3.1.3 Simulated datasets . . . 54
3.2 Experimental Datasets . . . 55
3.2.1 *C. elegans* . . . 55
3.2.2 *S. mediterranea* . . . 55
3.2.3 *Myeloid progenitors* . . . 56
3.3 Distance between labels . . . 56

4 Partial label learning for classification of scRNA Seq data **57**
4.1 Introduction . . . 58
4.2 Methods . . . 60
4.2.1 Task formalization . . . 60
4.2.2 Learning strategies . . . 62
4.2.3 Models . . . 66
4.3 Datasets and experimental settings . . . 72
4.3.1 Datasets . . . 72
4.3.2 Experiments . . . 72
4.4 Results . . . 75
4.4.1 Supervised setting . . . 75
4.4.2 Partial label, Overlap = 1 . . . 77
4.4.3 Partial label, Overlap = 0 . . . 79
4.4.4 Real-world datasets . . . 81
4.4.5 Incorporating hierarchy improves performance . . . 86
4.4.6 Supplementary Results . . . 89
4.5 Discussion . . . 93
4.6 Code availability . . . 94

5 Hierarchical Novel Class Discovery for classification of scRNA Seq data **95**
5.1 Introduction . . . 96
5.2 Related works . . . 96
5.3 Methods . . . 99
5.3.1 Problem formalization . . . 99
5.3.2 Two-step approach . . . 101
5.3.3 Single-step approach . . . 103
5.4 Experiments . . . 105
5.4.1 Datasets . . . 105
5.4.2 Experimental settings . . . 105
5.4.3 Metrics . . . 106
5.4.4 Methods and baselines . . . 108
5.5 Results . . . 108

5.6 Discussion 111

6 Conclusion **113**
6.1 Summary of Contributions 113
6.2 Discussion and Perspectives 114
6.2.1 The intersection of developmental biology and bioinformatics brings new challenges 114
6.2.2 Classification of scRNA Seq data to reconstruct cell state trajectory 115
6.2.3 The scale of the label structure 116
6.2.4 Multi-modal data integration 117

Bibliography **119**

List of Figures

1 Early embryogenesis of the nematode *C. elegans* . . . 17
2 Cell state versus cell lineage . . . 20
3 Problem statement . . . 28
4 Machine learning problems associated with the nature of the supervision. . . . 29

5 Tree-based methods . . . 36
6 Support Vector Machines for binary classification . . . 38
7 Prototype-based approach . . . 43
8 Example of hierarchical label structures . . . 44

9 Partial label problem statement and classification methods . . . 61
10 Supervised performance on simulated data . . . 76
11 Partial label results for the setting with label overlap (o=1) . . . 78
12 Partial label results for the setting with no label overlap (o=0) . . . 80
13 Classification performance on the *C. elegans* dataset . . . 83
14 Classification performance on *S. mediterranea* dataset . . . 85
15 Classification performance on *Myeloid progenitors* dataset . . . 86
16 Including hierarchical information improves performance . . . 87
17 Comparison of *IRL* and *IFR* algorithm in the partially labeled scenarios 90
18 Comparison of classification performance of main baseline methods on the three real-world datasets in the fully supervised setting . . . 91
19 Comparison of *SVM* implementations in the partially labeled setting 92

20 Hierarchical Novel Class Discovery Problem statement . . . 97

List of Tables

1.1 **An example of the proportion of data supervision type for a sc-RNA Seq dataset.** 27

3.1 **Characteristics of the simulated datasets** 55
3.2 **Experimental dataset characteristics** 56

4.1 **Details of methods for partial label learning task** 66

5.1 **Dataset's characteristics** 107
5.2 **Comparison of methods on NMI** 109
5.3 **Comparison of methods on ACC** 110
5.4 **Comparison of methods on classification metric** 111

Liste des acronymes

ACC Accuracy Cluster Classification.

EM Expectation Maximization.

GMM Gaussian Mixture Model.

IFR Iterative Full Retraining.

IRL Iterative Refinement Learning.

kNN k Nearest Neighbors.

LR Logistic Regression.

MDS Multidimensional Scaling.

NCD Novel Class Discovery.

NMI Normalized Mutual Information.

PB Prototype Based.

PCA Principal Component Analysis.

PLL Partial Label Learning.

RF Random Forest.

scRNA Seq single cell RNA Sequencing.

SSL Semi Supervised Learning.

SVM Support Vector Machine.

t-SNE Stochastic Neighborhood Embedding.

UMAP Uniform Manifold Approximation and Projection.

XGBM Extreme Gradient Boosting Method.

ZSL Zero Shot Learning.

Notation

Graph.

$\mathscr{A}(.)$ The set of ancestor of a node / class in a Tree.

$\mathscr{D}(.)$ The set of descendant of a node / class in a Tree.

$f[\alpha]$ The node father of the node / class α.

$H \in \mathbb{R}^{C,C}$ The pairwise distances matrix between labels in the lineage tree.

Notations of the problem.

d dimension of the feature space $\mathscr{X}$.

C The number of classes in a dataset.

K The Number of clusters.

L The Number of supervised samples.

M The Number of partially labeled samples.

N The Number of samples.

U The Number of unsupervised samples.

Datasets and spaces.

D A dataset.

$D_s = \{(x_1, y_1), ..., (x_L, y_L) \in \mathscr{X} \times \mathscr{Y}_s\}$ Supervised dataset.

$D_{pl} = \{(x_{L+1}, Y_{L+1}), ..., (x_{L+M}, Y_{L+M}) \in \mathscr{X} \times \mathscr{Y}_{pl}\}$ Partially labeled dataset.

$D_u = \{x_{L+1}, ..., x_{L+U} \in \mathscr{X}\}$ Unsupervised dataset.

$x_i \in \mathbb{R}^d$ A sample.

$y_i \in \mathscr{Y}$ The label or class associated to x_i.

$\gamma \in \mathscr{Y}$ A label for inference.

$z_i \in \{1, ..., K\}$ The latent variable associated to x_i.

$\mathscr{X} = \mathbb{R}^d$ The space of sample.

$X = \{x_1, ..., x_N\}$ A finite subset of samples.

$\mathscr{Y}$ The space of labels / classes.

$\mathscr{Y}_s \subseteq \mathscr{Y}$ The subset of supervised labels.

$\mathscr{Y}_{pl} \subseteq \mathscr{Y}$ The subset of partially labeled labels.

$\mathscr{Y}_u \subseteq \mathscr{Y}$ The subset of unsupervised labels.

Y_i The subset of labels associated to a sample x_i.

$Z = \{z_1, ..., z_N\}$ A subset of latent variables.

Models.

f_W A parametric model f with parameters W.

m mapping function between labels and clusters.

$\mu \in \mathbb{R}^d$ The centroid of a cluster.

ψ Mapping function for the kernel projection.

$\phi \in \mathbb{R}^d$ One prototype associated to a label.

$\Phi \in \mathbb{R}^{C,d}$ The matrix of all the prototypes associated to all the classes.

$\Omega(f)$ The regularization term on a model f.

Θ The set of parameters of a probabilistic model.

1 Introduction

Content

1.1 Context . . . 16
1.1.1 Study of the embryonic development . . . 16
1.1.1.1 Developmental processes . . . 16
1.1.1.2 Data acquisition techniques highlight various biological processes . . . 17
1.1.1.3 Data analysis . . . 18
1.1.2 Cell state trajectory reconstruction . . . 19
1.1.2.1 The cell state trajectories differ from the cell lineage . . . 19
1.1.2.2 Measuring the cell lineage . . . 21
1.1.2.3 Measuring the cell state . . . 21
1.1.2.4 Current approaches for reconstructing cell state trajectories . . . 22
1.1.2.5 Objective: Mapping of cell state to the lineage tree . . . 24
1.2 Problematic . . . 25
1.2.1 Cell type classification in single-cell RNA sequencing data . . . 25
1.2.1.1 Cell type and single-cell classification . . . 26
1.2.1.2 Type of annotation in scRNA Seq datasets . . . 27
1.2.2 Statistical learning for solving various classification tasks . . . 28
1.2.2.1 Supervised classification . . . 29
1.2.2.2 Partial Label Learning . . . 30
1.2.2.3 Novel Class Discovery . . . 31
1.3 Content of the thesis . . . 33

This chapter introduces the context, motivations, and challenges for this work. First, we motivate the main question which consists in studying cell trajectory reconstruction in studies of developmental biology. We then detail the data acquisition techniques usually used as well as their limitations. Following the limitations, we propose to formalize the challenge of reconstructing cell trajectories during development as a machine learning problem. In the second part of this chapter, we then describe the characteristics of the data we are interested in and present the different machine learning tasks that structure this manuscript. Finally, we present the content of the thesis.

1.1 Context

1.1.1 Study of the embryonic development

1.1.1.1 Developmental processes

The motivation for this dissertation comes from studies of developmental biology. This field of biology studies how a single cell, the egg cell is turned into a multicellular organism. After fertilization of the egg cell, the embryo develops through a combination of cell division, cell differentiation, and cell spatial rearrangement until it becomes an autonomous and functional organism [1]. Understanding embryogenesis is essential for elucidating the fundamental principles of life and evolution. The general process of development involves many complex cellular processes, such as cell proliferation, differentiation, spatial organization, cell signaling pathways and tissue remodeling. Understanding these processes enables the development of other areas of research, including genetics, medicine, cancer treatment, and *in vitro* organ development.

Embryonic development is usually broadly divided into several periods, with a first step of cell proliferation, followed by organ formation with organogenesis and morphogenesis [2], [3]. During the various stages of development, many mechanisms interact, leading to the generation of specialized cell types with distinct structures and functions. These specialized cells organize themselves into tissues, organs, and organ systems, forming the structures necessary for the organism's survival and function.

During the early stages of development, the main mechanisms are cell proliferation, which includes both cell divisions and cell growth; the spatial arrangement of cells; and cell differentiation [4]. These diverse processes necessitate individual and coupled analysis for the understanding of development.

Cell proliferation is a dynamic interplay between cell growth and cell division. Cell division is the process by which a cell divides into two daughter cells. The entire cell division history is called the cell lineage and has the structure of a tree. An illustration of the general process is presented in Fig.1, where panel A shows the scheme of division and panel C shows developmental time-lapse microscopy images of the nematode *C. elegans* [6]–[8].

Throughout development, cells transition from stem cells to fully differentiated cells, with each cell state representing a different level of specialization. This transformative process is widely recognized as cell differentiation, whereby cells acquire distinct identities and functionalities. As an example, in the context of myeloid cells, which encompass various types of blood cells, hematopoietic stem cells (HSPs) serve as multipotent cells capable of giving rise to a range of specialized cells. Upon division, the progeny of HSPs differentiate into intermediate cell types such as Common Myeloid Progenitor (CMP) and Common Lymphoid Progenitor (CLP). These intermediate cells,

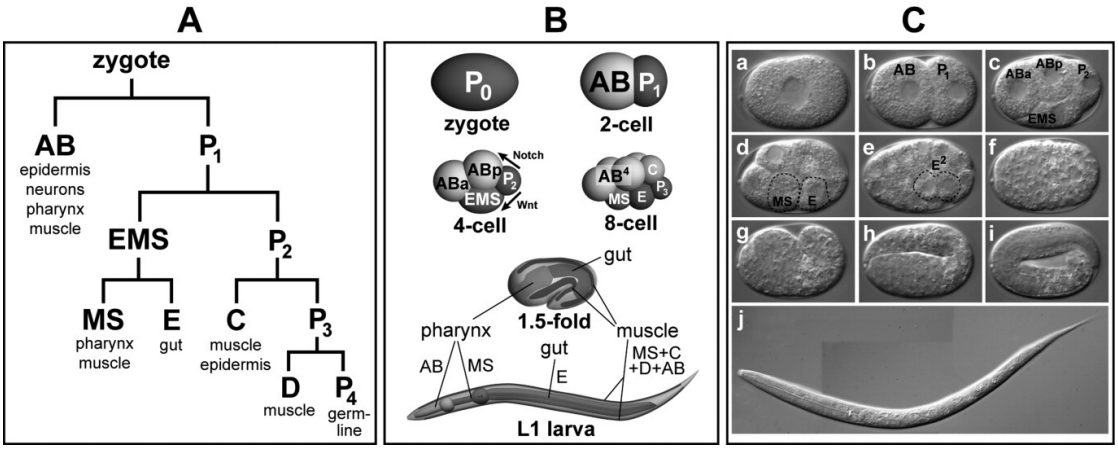

Figure 1 – **Early embryogenesis of the nematode *C. elegans***
The figure is adapted from [5]. **A)** Partial lineage tree of *C. elegans*. This lineage is invariant [6], and as a consequence, each cell in the tree can be named uniquely using an encoding based on its cell type and the orientation of cell division. **B)** Schema of the succession of cell divisions during early embryogenesis, including the corresponding names from the lineage tree of the cells. **C)** Microscopy images of the succession of cell divisions, with annotations of the cells.

in turn, undergo further division and specialization. For instance, CMP cells can differentiate into erythrocytes, while CLP cells can develop into B-cells or T-cells [9], [10].

Understanding cell differentiation necessitates identifying the level of cellular specialization throughout development. This involves pinpointing moments of cell specification and characterizing the significance of these changes for progeny cells.

1.1.1.2 Data acquisition techniques highlight various biological processes

To observe and measure the developmental processes, different acquisition techniques have been developed, with two main categories, microscopy imaging and multi-omic measurements.

Microscopy techniques have developed immensely in the context of developmental biology [11]–[14]. One of the most important technique is fluorescence microscopy. This technique uses a laser to excite fluorophores that are associated to various cellular structures (nucleus, membrane) or specific proteins. After excitation, these fluorophores will release light that can be recorded at various scales [15]. When analyzing these types of datasets, major challenges appear, in particular, one needs to track and segment images and videos to obtain relevant quantitative descriptors of cellular processes.

The other category of acquisition techniques used to assess cell and tissue phenotypes is known as omics. This collection comprises several branches: genomics, which examines the entire genome through DNA sequences [16]; transcriptomics, focusing on gene expression levels [17]; proteomics, which measures protein abundances [18]; metabolomics, dedicated to analyzing metabolite concentrations [19]; and epigenomics, which investigates modifications to DNA and proteins [20]. The integration and the analysis of these multiple types of omics data is commonly called multi-omics [21]–[23]. Embryogenesis is inherently a multi-scale process [24], [25] in which only selected omic,'(e.g. protein expression), can be measured at any given time. Advances in data acquisition technologies, such as genome sequencing, provide an efficient approach to studying the mechanism of embryogenesis [26]–[28].

Although the different acquisition methods generate a large amount of data, they provide complementary measurements, but rarely the full range of information. It is necessary to develop frameworks to analyze and integrate the data from different modalities.

1.1.1.3 Data analysis

These datasets pose various methodological challenges. First, they exhibit high dimensionality, such as in genetic sequencing, encompassing several tens of thousands of features [29]. They also involve dynamic interactions and evolution [30]. Moreover, various sources of noise, including those arising from acquisition techniques and variability between intra- and extracellular populations, further complicate the analysis [31]. These methods yield intricate datasets, which pose challenges for direct analysis and necessitate bioinformatics [32], and statistical frameworks, particularly machine learning, to enable more comprehensive analysis [33].

The increasing amount of data motivates the interest of automated methods. These methods minimize the need for labor-intensive and expert-level manual annotation [34]. Automated methods provide robust and unbiased analysis [35], and facilitate various applications, such as medical assistance, [36], [37]. One of the major challenges currently is to perform automated annotation of scRNA Seq datasets into accurate cell types [34], [38]–[40].

The annotation of omic data is driven by the need to establish cell atlases, which are comprehensive and systematic collections of information about the various cell types present in an organism or a specific tissue. These atlases provide detailed molecular and functional characterizations of individual cells (e.g. gene expression profiles, morphology, spatial organization). Current annotation strategies rely on information provided by existing atlases [41], [42], expert knowledge, and transfer learning from already annotated datasets.

Finally, the use of machine learning methods to analyze large datasets is also driven by deep learning, a subset of statistical learning methods, which has led to breakthroughs in various fields, such as image recognition, signal processing, natural language processing, and many other domains. In the field of bioinformatics, the use of deep learning methods is motivated by their ability to uncover hidden and meaningful patterns within complex datasets [36], [43]–[45]. Characterized by a flexible architecture, these models can be easily customized to address various distinct problems [46], [47] such as dimensionality reduction [48], [49], data completion [50], clearing data [51], discovering new cell types [52] and cell identification [53], [54].

1.1.2 Cell state trajectory reconstruction

Understanding the diversification of cells during embryogenesis poses a fundamental problem in developmental biology. One effective approach to address this problem is to reconstruct individual cell trajectories [55], [56]. This means tracing the specification of cells throughout development. These trajectories are not directly observable as such because many mechanisms are involved at the same time (cell division, tissue organization, protein reactions ...) [57], [58]. We propose to address the problem of reconstructing the individual cell trajectories by considering two main processes of embryogenesis: cell proliferation and cell differentiation.

1.1.2.1 The cell state trajectories differ from the cell lineage

Cell lineage tracing involves tracking the developmental history of individual cells to understand how cells proliferate and contribute to the formation of tissues and organs during embryonic development. From molecular biology to the single-cell era, lineage tracing techniques have grown immensely [27].

In the context of cell state trajectories, a major challenge is to elucidate the landscape of possibilities of cellular differentiation and characterize the different cellular states [59], [60]. These states include fully differentiated or specialized cells (such as neurons or muscles), as well as undifferentiated cells. The definition and characteristics of these states depend on multi-scale processes (either at single cell, tissue, or organ level) and can be caracterized using multi-omic information [61], [62].

To infer cell state trajectories, the objective is to create a combined view of the lineage and the cellular state landscape, however the dynamics of these two aspects may differ. In Fig. 2a, the cell lineage is shown as a hierarchical representation of the history of cell division, and the cell state trajectory is shown as a continuous and evolving process where many directions for the cell state trajectory are possible. The combination of the two processes is not straightforward and a non-exhaustive list of situations is described in the next panels (b-h). The first case is when both cell lineage and cell state are perfectly matched. This means that cell differentiation occurs exactly

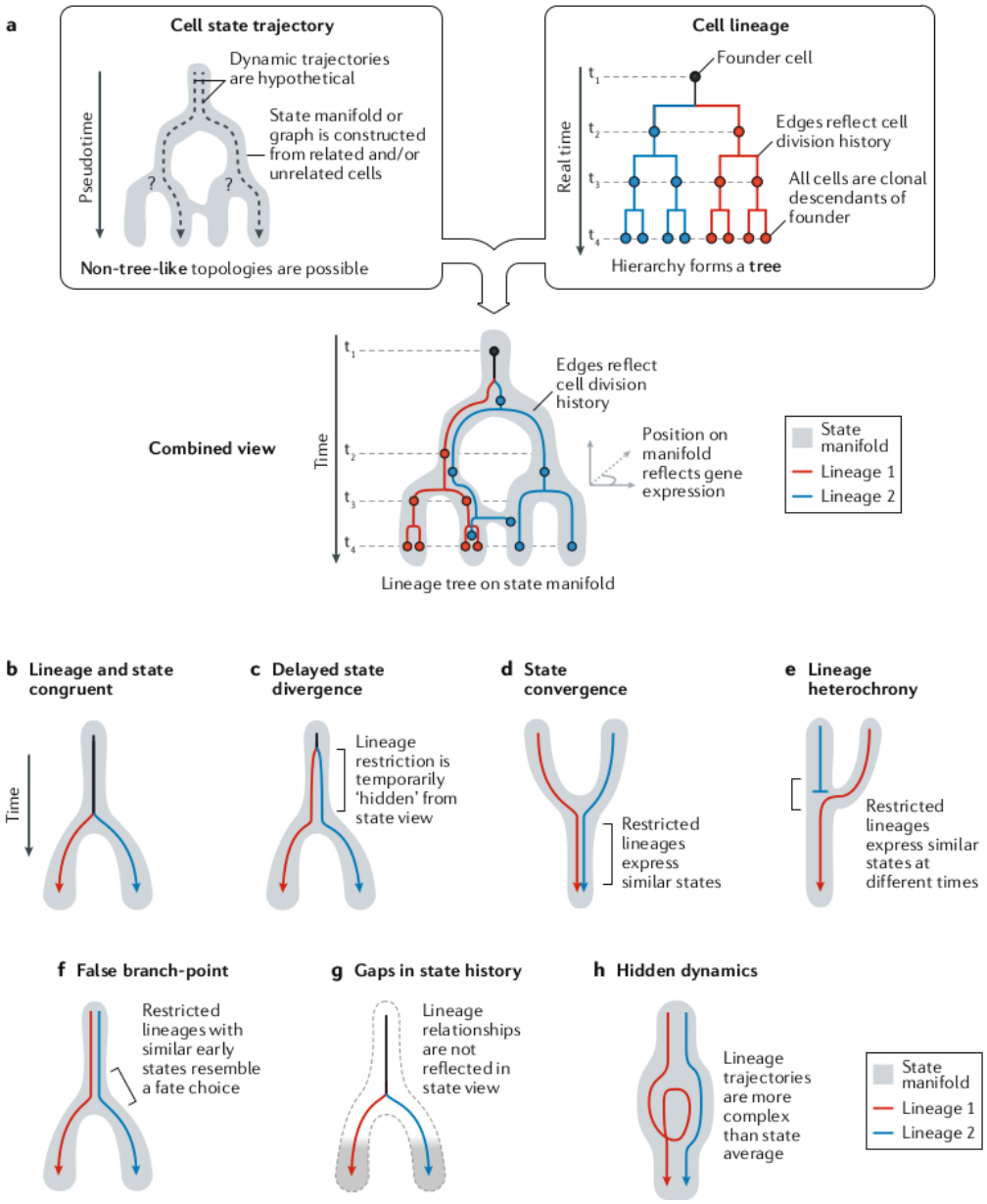

Figure 2 – **Cell state versus cell lineage**
The figure is adapted from [59]. **A**) Difference between cell state trajectory and cell lineage. On one side, there is the cell state manifold, which describes the landscape of the cellular states and defines the set of possible cell state trajectories. On the other side, the lineage tree is the history of cell division. **B-H)** A non-exhaustive list of situations is described where mapping cell state trajectory and cell lineage can either correspond **B-C)** before or after the merging process **D)**. deferred at merging cell states.

at the time of cell division. In the second case, differentiation is delayed compared to cell division. The next cases describe different situations of convergence or divergence

of lineage in the cell state manifold. The last case is more realistic, it describes that in real-world developmental processes, dynamics are complex and are a combination of several simpler cases.

The current section describes the cell lineage, with a focus on the lineage tree. Then we will discuss cell state trajectory based on RNA measurements. Finally, we will justify matching these two modalities to contribute to the general problem of cell state trajectory reconstruction and provide a non-exhaustive list of alternative approaches.

1.1.2.2 Measuring the cell lineage

Lineage tracing aims to understand and characterize the fate of each cell in the cell lineage. This problem has been studied for over a century, and specific techniques and algorithms have been developed [27]. To track cell proliferation, a subset of techniques consists of recording the series of cell divisions and tracking morphological changes over time, notably by using fluorescent 3D microscopy [8], [63], [64].
Another set of techniques is single-cell lineage tracing. It consists of introducing a heritable genetic sequence into a cell, which is then passed down to its descendants as they undergo division. Researchers can analyze the sequences of all descendant cells to determine clonal inheritance and construct the lineage of individual cells based on the similarities of their respective sequences. [27], [65]–[69]

Our focus will be on the lineage tree; which is the history of cell division. This tree represents the set of connections between cells during embryogenesis. A connection exists if there is a direct link between two cells through a mother-daughter relationship. Important discoveries rely on linking other modalities to the lineage tree, such as gene or protein expression [7], [70], [71]. Working with a lineage tree will then make easier identification of key moments of cell differentiation during the cell proliferation step [39], [72].

1.1.2.3 Measuring the cell state

DNA, RNA, and proteins are the basic biomolecules that underpin the central processes of life. DNA is found in the cell nucleus and serves as the repository of genetic information, containing the instructions for the synthesis of RNA. RNA plays a key role in translating the genetic code of DNA into functional proteins. In particular, RNA molecules can act as messengers (mRNA) to facilitate the synthesis of proteins. RNA is specific to each cell in an organism. Its expression is dynamic over time and during development. The Proteins perform diverse functions such as catalyzing chemical reactions (enzymes), providing structure (e.g. collagen), enabling transport (hemoglobin), and acting as antibodies. This intricate interplay between DNA, RNA, and proteins governs the flow of genetic information and the functional diversity essential for life's processes [73].

Many contributions come from the analysis of the protein expression [74], [75], but the analysis of this modality has some limitations for cell state reconstruction, such as the fact that only a few proteins can be measured at the same time, or the cascade of reactions between RNA and protein cannot be directly observed [32], [76]. In parallel, other technologies based on RNA sequencing make it possible to study the role and effects of RNA. It is based on the sequencing of a whole tissue (a significant number of cells), this means measuring the amount of specific mRNA by counting the number of molecules that correspond to a specific gene. These experiments provide an average expression of each gene in the tissue. Such experiments are highly beneficial for identifying genetic anomalies in the tissue [77], [78] but a main limitation is that results are hard to interpret when a tissue is composed of a diversity of cells, each of which has different transcriptomic profiles and specific functions in the tissue (for example in brain tissue, there is a variety of cells as neurons, astrocytes, oligodendrocytes ...).

A more recent and revolutionary technology has been developed over the two last decades, which is the sequencing of RNA from single-cells (scRNA Seq) [79], [80]. As in RNA sequencing technologies, the quantity of expressed mRNA is measured, but at the level of individual cells. This means counting the number of molecules corresponding to each gene for each cell in the tissue. scRNA Seq technologies provide datasets in which each sample represents a cell, and each feature corresponds to the quantity of expressed molecules of a specific gene. These datasets are encoded as a set of vectors known as transcriptomic profiles, of dimension equal to the number of genes (typically ranging from 10,000 to 30,000, depending on the species). This approach allows us to take into account the diversity contained in tissue and is very promising [81], [82].

Even if these datasets are inherently dynamic, as the transcription process is, the observable data are only a “snapshot” of the corresponding transcriptomic profiles [83]. In fact, to perform the measurements, scRNA Seq technologies need to stop embryonic development and then perform a “snapshot” measurement. Then, the complete scRNA Seq datasets are a set of independent “snapshot” measurements of different embryos. This poses new challenges to reconstruct the trajectories of cell state between the time-points [83]–[86]. Another limitation is that scRNA Seq technologies lose the spatial and lineage information of each cell, which means that direct causal analysis is not possible and requires either prior information or manual annotation. New technologies combining both spatial measurement and scRNA sequencing, called spatial transcriptomics, are currently being developed [87]–[90] and can facilitate the reconstruction of cell state trajectories by adding the spatial information in a multi-modal framework [91], [92].

1.1.2.4 Current approaches for reconstructing cell state trajectories

Many studies focus on reconstructing the trajectory of single-cell states [93]. Various strategies have emerged, based on direct trajectory inference in one omic data [69],

[94], [95], or including mapping omic-omic data or omic-microscopy data [93] (linking transcriptomic data to lineage trees [69], or transcriptomic data with spatial information[96]). Recurrent computational tools in this domain include clustering, manifold learning, graph construction, temporal modeling, and integration of multi-modalities. We present briefly what each computational approach brings to the general problem of cell trajectory reconstruction.

Analysis of the cell state manifold

Cell state is usually described in a high-dimensional space, the set of expressed genes. In this space, the set of cell states, for example that individual cells explore during development, constitute what is called the cell state manifold [59]. In many state-of-the-art cell state trajectory inference methods, the cell state manifold is approximated with manifold learning techniques [97]. Manifold learning and especially non-linear dimensionality reduction are essential to handle this data correctly [98]. The idea is to approximate the geometrical structure on which the set of data points lie, and identify, in an unsupervised way, a good parametrization of this structure that will serve as a low dimensional representation of the data. In practice, this is done by approximating the manifold with pairwise similarities between samples through graph construction. This graph enables subsequent trajectory inference [94], [99]–[101]. This pipeline of manifold learning via graph-based approach is often combined with clustering methods to permit trajectory inferences [102]–[106]. One limitation is that scRNA Seq data or other single-cell omic data are high dimensional data and are sensible to the curse of dimensionality [107].

Manifold learning methods include usually a minimal set of assumptions about the structure of the dataset [108]. It is possible however possible to include *a priori* knowledge about the structure of the data when searching for the appropriate geometry. A recent example of such search for appropriate geometry uses the hyperbolic geometry for representing differentiating single-cell transcriptomics profiles [109], [110]. This approach assumes that the transcriptomic profiles diverge from each other over time, while the cells are changing from their undifferentiated state to the more specialized ones.

A recent and fruitful line of research has been the application of the mathematical theory of Optimal Transport to biological problems [111]. Briefly, Optimal Transport is a method to infer a transport map between two probability distributions that minimizes the distortion. It has been used to characterize trajectories in the cell state manifold by infering the coupling between transcriptomic profiles at successive time points [60], [95], [112]. This approach is very efficient and adapted for single cell state trajectory reconstruction, however mainly rely on “local” couplings between transcriptomic profiles, i.e. between consecutive time points.

One way to take into account the global dynamics of the differentiation process is to assume an underlying dynamical system. This dynamical system is modeled as a set of differential equations. The behavior of the dynamical system can be inferred using a neural network architecture called NeuralODE. This approach has been successfully applied on time series of transcriptomic profiles [113]–[115].

Omic-spatial mapping

To understand individual cell trajectories in time, it can be relevant to map single cell transcriptomics profile onto 3D embryo through time and cell division [116]. It has been shown that mapping to other modalities is particularly relevant, especially with deep learning approaches [117], [118].

Optimal Transport has also been used for pairwise modalities mapping [96], [119], and more recently, it has been combined with a dynamical framework [120].
All of these approaches have been shown to be very effective. They can lead to the establishment of dynamic atlases of Spatial Transcriptomic reference [121], [122], in which both spatial and transcriptomics information are available for all the cells; such as [123] or more recently [124].

Multi-modal framework

Multi-modal approaches have been the focus of recent research developments. For instance, techniques like multi-modal alignment, such as those discussed in [125], [126], and the utilization of semi-supervised scenarios [127]–[129] have demonstrated the relevance of aligning different modalities.
In the single-cell omics era, the integration of multi-omics data holds significant promise, with the current challenge lying in the development of integrative models encompassing all modalities [130]–[133].

Limitations

The approaches presented in this section are very efficient and useful in many circumstances, however, the main limitation is the lack of a model unifying the cell state, the cell lineage, and the spatial arrangement of the cells. Even if in this thesis, we do not provide such a model, we aim to contribute to it. In the next paragraph, we will develop the subject of this thesis, the mapping of scRNA Seq data to the lineage tree. By providing automatic methods and algorithms to identify, cell state to cell lineage, it make easier the integration of other modalities.

1.1.2.5 Objective: Mapping of cell state to the lineage tree

Analyzing scRNA Seq data to identify and reconstruct cell state trajectories presents a new challenge. As scRNA Seq data lacks both spatial and lineage information, the direct relationship between transcriptomic profiles is not readily apparent. Therefore, we propose identifying transcriptomic profiles in cell types within the lineage tree in order to reconstruct cell trajectories using both genetic and lineage information. This approach offers several advantages, particularly in identifying key moments of cell differentiation in the lineage tree. Additionally, it can contribute to discoveries in other studies, such as protein expression in the lineage tree. It is motivated by a better understanding of the dynamic regulatory network of gene expression and its effects at both single-cell and tissue scales. Finally, the Identification of transcriptomic profiles

into a relevant structure can contribute to the establishment and reinforcement of single-cell atlases [134].

To identify transcriptomic profiles, the usual workflow is to perform first clustering of all transcriptomic profiles into different sub-groups and then manually identify cell types based on prior information [81], [135], such as unique gene expression or patterns in different groups. This two-step approach is difficult and requires expert annotation. Several strategies have been developed to identify these data, such as correlating reference expression data, or label transfer through supervised classification [34], [136]–[140].

In summary, the current research work aims to reconstruct cell trajectories during embryogenesis. To address this, we focus on studying cell proliferation within the lineage tree and cell specification based on RNA expression, specifically using scRNA Seq data. However, a current limitation is the absence of a direct link between scRNA Seq data and lineage tree information. The objective of this work is to associate each transcriptomic profile with a specific cell within the lineage tree.

1.2 Problematic

We propose in this thesis to contribute to the problem of reconstructing single cell trajectories during development by addressing the question of how to automatically map single-cell RNA Seq profiles onto a lineage tree.

In the previous section, we developed the motivations for the classification of scRNA sequencing data and its mapping on the lineage tree, especially to overcome the two main drawbacks of scRNA Seq data, which are the loss of spatial information and the fact that these datasets comprise the RNA measurements at irregular time points from independent embryos. We now describe the task of classification, then specify different types of supervision on the scRNA Seq datasets and expose the respective problems for each type of supervision.

1.2.1 Cell type classification in single-cell RNA sequencing data

The objective of scRNA Seq data classification is to predict labels from input transcriptomic profiles. In the machine learning framework, models are based on statistical learning, which implies learning on training datasets in order to predict on new datasets. Two fundamental notions need to be further discussed to define this framework: representing labels as cell types and understanding the diverse levels of supervision in the datasets utilized for training.

1.2.1.1 Cell type and single-cell classification

The term cell type classification is relatively uncommon in the literature; instead, it's often referred to as cell type identification, [141], [142], which incorporates both classification and clustering methods. While clustering and classification pose different challenges [143], both contribute to the task of cell trajectory reconstruction. Our interest lies in both clustering and classification, depending on the level of supervision in the scRNA Seq dataset, a concept discussed in the following section. Unlike the clustering problem, classification requiers to define a set of labels. We propose two approaches for choosing the label set: considering the cell types or considering all cells that have appeared in the lineage tree.

The first approach of considering the set of labels as a set of cell types for classification is the most common choice. It supposes that the knowledge of the set of cell types is already known. However, the definition of cell type and the corresponding annotations are ambiguous topics [144], [145]. Depending on the embryonic developmental stage, functionalities, and species, the definition of cell type may change. The selection of the set of cell types for the labels set should align with specific applications and may differ accordingly. For example, the labels chosen for tasks like cell state reconstruction or tumor detection may differ.

The second option is to regard the complete lineage tree as the classification label set. This approach offers a comprehensive depiction of the cell state landscape and seems more suitable for reconstructing cell trajectories. However, this option is often impractical due to the substantial number of labels involved. During development, the number of cells increase exponentially through divisions. In humans, this can reach tens of billions of cells. Moreover, the variability in cell divisions among different embryos complicates establishing a unified lineage tree across datasets. Thus, considering the entire lineage tree may be feasible only during early developmental stages when the number of labels is relatively low, typically around thousands of cells.

Finally, in the scRNA Seq classification problem, the definition of a unique label space is not straightforward. We propose to rely on existing datasets with annotations corresponding to either a set of cell types or to the lineage tree, if available. Note that considering only the cell types is already a challenging classification problem with a high number of heterogeneous labels, going from tens to hundreds [146].

Whether on the scale of the lineage tree or the cell type, the classification problem is inherently hierarchical. For the lineage tree, the hierarchical structure is directly provided by the set of divisions. When considering only the set of cell types, the hierarchical structure might not exhibit a direct correlation. In scenarios where hierarchical dependency aligns with cell differentiation, the hierarchy is direct. For instance, the Myeloid Progenitor cells develop from pluripotent cells to intermediate stages and fully specialized cells [9]. However, in other instances, the hierarchical aspect may be hidden and necessitate inference, particularly with methods like hierarchical clustering [147]–[149].

The hierarchical structure of labels can offer additional insights for various applica-

tions of scRNA Seq classification, particularly in understanding cell differentiation. Incorporating this information into the label structure is crucial and should be integrated into both clustering and classification tasks to enhance analysis.

1.2.1.2 Type of annotation in scRNA Seq datasets

Classification methods aim to predict pertinent labels for input data. The core principle of machine learning models involves training on annotated training data and subsequently making predictions on new datasets. However, obtaining annotations for scRNA Seq data is challenging due to its inherent complexity. Typically, the annotations are obtained by first performing dimensionality reduction, then clustering, and finally, identification of clusters to cell types with prior knowledge (e.g., marker genes). This process heavily relies on existing knowledge and results in non-uniform supervision across transcriptomic profiles in the datasets. There are 3 different types of supervision, full supervision, ambiguous supervision, and no supervision. For example, Tab 1.1, shows the distribution of different types of supervision in the dataset *C. elegans* [150]. We will now discuss different types of annotations.

	D_s	D_{pl}	D_u	Total
Nb sample	5143	42631	41927	89701
%	6	47	47	100
Nb label	95	970	287	1342
%	7	72	21	100

Table 1.1 – **An example of the proportion of data supervision type for a sc-RNA Seq dataset.**
The dataset is taken from [150] and is composed of 89701 transcriptomics profiles of cells that have been present in the development of *C. elegans*. The lineage tree of *C. elegans* is invariant [6], and it is composed of 1342 cells in total. Some of the samples are uniquely annotated and correspond to the dataset D_s, where s stands for supervised, half of the samples are partially annotated and form the dataset D_{pl}, and the last part of the samples are not annotated and form an unsupervised dataset D_u. In this dataset, the cells in the lineage tree have either only supervised samples, only partially annotated samples, or only unannotated samples. The proportions of labels by type of supervision are shown in the second part of the table.

First, there is full supervision, which means that for one transcriptomic profile, there is a unique annotation. This is the ideal situation to perform a statistical framework.

The next type of supervision is partial annotation. This means that for a given transcriptomic profile, we don't have a unique label annotation, but rather a group of candidate labels, of which only one corresponds to the ground truth. This ambiguous annotation contains valuable information and may represent a large proportion of the data (see Tab 1.1). In general, a candidate set is quite small and the ambiguity comes

from the similarity between the cells in the candidate set. In the *C. elegans* dataset [150], the size of the candidate set is between 2 and 8 candidates. The similarities observed between candidate cells can often be attributed to asymmetric division and tissue reorganization, for example for neurons in *C. elegans*, [151].

The last type of supervision is the absence of supervision, and it could represent a large proportion of the data [152]. Finding the good labels of these unsupervised transcriptomic profiles is a very challenging problem.

The prior knowledge of cell types isn't consistently available, making it easier to identify the associated transcriptomic profiles of some cell types compared to others. We leverage this concept to define label overlap, particularly in the scenario of no label overlap, where theoretically, all transcriptomic profiles associated with a cell type are either fully annotated, partially annotated, or unannotated. The concept of label overlap will be integrated into various learning problems.

Considering all these factors, we can delineate the general problem of scRNA Seq classification, as depicted in Fig 3. This problem entails identifying and categorizing transcriptomic profiles within a structured hierarchy corresponding to a lineage tree. This categorization is based on the varying degree of supervision, encompassing both the transcriptomic profiles and the cell types.

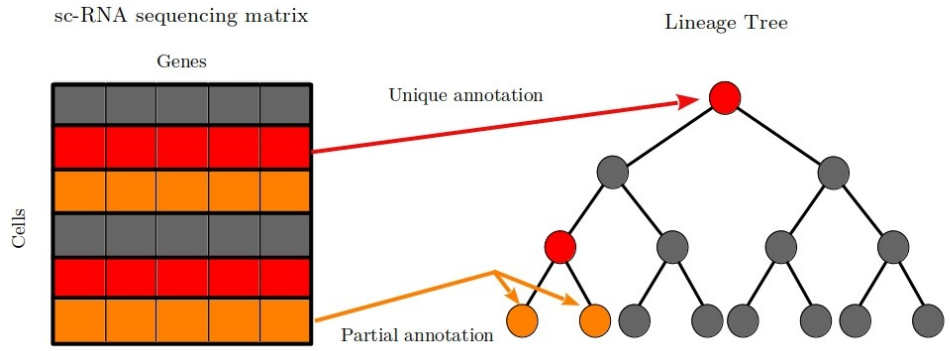

Figure 3 – **Problem statement**
We consider the classification problem of the scRNA Seq matrix on the left. Each line represents a transcriptomic vector and is either uniquely annotated (in red), partially annotated (in orange, i.e. several labels are candidates for a unique annotation), or unannotated (in grey). These labels are structured in the form of a tree (the lineage tree on the right). The aim is to label all the data uniquely.

1.2.2 Statistical learning for solving various classification tasks

The problem statement takes into account the 3 different types of annotations and each type of annotation corresponds to different learning problems: supervised classification, partial label learning for ambiguous annotation, and novel class discovery for

unannotated data. The different scenarios depending on the supervision in a dataset are described in the schema shown in Fig.4. In the following sections, the terms "supervised" and "labeled" will be used interchangeably, despite being considered misnomers in the field of machine learning. "Supervised" refers to learning tasks, while "labeled" denotes the presence of labels in the data.

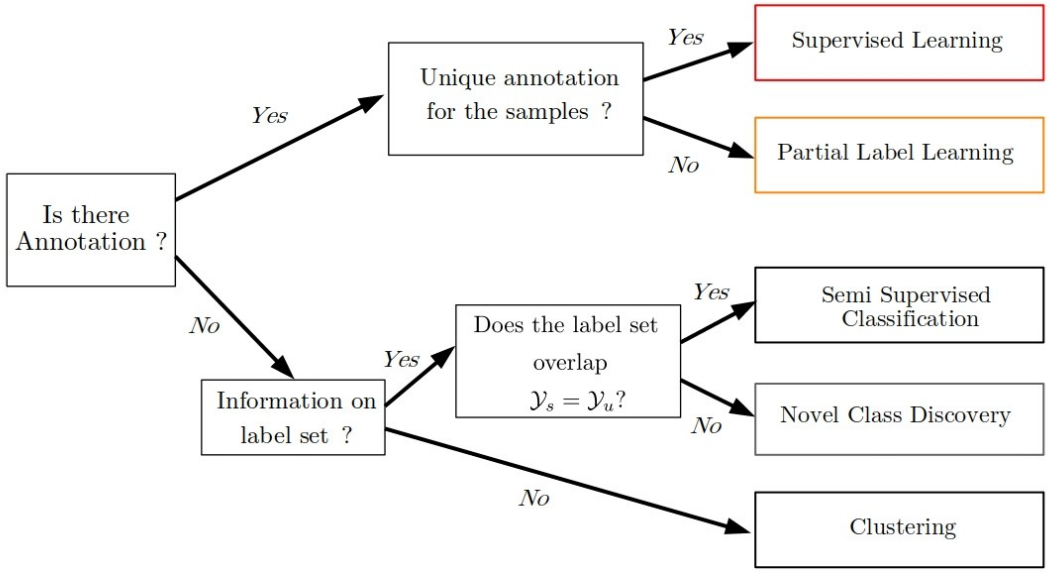

Figure 4 – **Machine learning problems associated with the nature of the supervision.** The nature of the learning problem is determined by the level and type of supervision present in a dataset. In the first scenario, the presence of annotations dictates whether the learning is supervised or weakly supervised. If each sample possesses unique annotations, it's considered supervised learning, while ambiguous annotations lead to partial label learning. In the second scenario, where no annotations are available, the distinction lies in whether there's prior knowledge about the label structure. Without such knowledge, the problem becomes a clustering task. Conversely, if there is knowledge about the label set, it splits into semi-supervised classification when sample labels overlap with those in the supervised subset, and novel class discovery when there's no overlap.

1.2.2.1 Supervised classification

In this study, our primary focus is on training models to classify transcriptomic profiles. However, several challenges emerge from this objective. In the realm of machine learning and deep learning, predictive models wield significant power and have gained attention in bioinformatics research [47]. Nevertheless, developmental biology data, particularly scRNA Seq, present multiple challenges for supervised classification tasks. These challenges come from the limited amount of labeled data, the presence of noise, and the high dimensionality of transcriptomic profiles. Consequently, conventional machine learning models are prone to overfitting, necessitating specialized data

processing for accurate analysis [153]. Additionally, adapting the architecture of the models is essential in addressing these challenges [154]–[156]. Finally, the supervised classification problem will serve as a baseline for the next challenges of partial label learning.

The classification of scRNA Seq data into a lineage tree is inherently hierarchical due to the structured nature of labels. Cells positioned closely in the lineage are expected to share similarities in their transcriptomic profiles, reflecting the dependency between the lineage and cell differentiation. Identifying the characteristics motivates the formulation of a hierarchical classification problem. The objectives are to integrate this knowledge into the supervised classification model and to investigate its impact on performance improvement. Chapter 2 will explore effective methods for incorporating this hierarchical information into classification models, and Chapters 4 and 5 explore several extensions.

1.2.2.2 Partial Label Learning

The main contributions of this thesis are motivated by the partial labeling of scRNA Seq data and the challenges brought by this type of annotations. Partial annotation consists to instead having a unique annotation for each sample, only a subset of candidates are available in the training dataset, where only one is the ground truth label. This scenario is an instance of weakly supervised learning [157], [158]. The core idea is to exploit partial annotations, recognizing that they carry information that may be complementary to that provided by fully supervised data [159]. This leads to the exploration of how existing methods can be adapted and algorithms designed to effectively exploit this partially annotated data.

One challenge is to quantify the performance loss associated with replacing a single annotation with a partial annotation. Indeed, rather than finding the exact correct annotation, a simpler approach might be to identify a set of candidate annotations, thus saving time and resources. In addition, it is important to describe the setting of the partial labeling, with the number of candidates for one transcriptomic profile and characteristics between the candidates. Notably, we explore the significance of label hierarchies by examining how hierarchy influences the selection of candidate labels. For instance, do the candidate are closely related in the hierarchy or not. Consequently, it is crucial to develop hierarchical extensions to the models used for learning from partially labeled data.

Our study will additionally be organized based on the concept of label overlap, which involves comparing labels belonging to fully supervised data with those in partially labeled data. This distinction is prompted by the dependency of transcriptomic profile annotations on the quality and quantity of prior information about their associated labels. Consequently, when examining the distribution of transcriptomic profiles associated with a specific label, the data may demonstrate unique annotations, partial annotations, or a blend of both. Consequently, it becomes necessary

to characterize the supervision of a label based on the supervision of its associated transcriptomic profiles: fully supervised, fully-partially supervised (involving only partially labeled transcriptomic profiles), or a combination of both. This perspective further gives rise to the concept of overlap between fully supervised labels and fully partially labeled ones. This notion of overlap between labels presents varying levels of difficulty, resulting in scenarios of complete overlap, no overlap, and partial overlap. In cases of overlap, a well-trained model is expected to easily disambiguate the data, and the addition of partial information should improve generalization predictions and prevent overfitting. In the opposite scenario, when there is no overlap between labels, the model must rely solely on ambiguous data, so the main challenge is to disambiguate the data effectively.

1.2.2.3 Novel Class Discovery

The last learning problem discussed in this thesis concerns the scenario where transcriptomic profiles are either supervised with a single annotation or entirely unlabeled, thereby presenting a semi-supervised learning problem. An additional characteristic of the label distribution is notable: supervised and unsupervised data possess distinct label sets. To clarify, this means that all unsupervised transcriptomic profiles have their ground truth label included in the set of unseen classes (shown in grey in Fig 3). Conversely, there are no annotated transcriptomic profiles corresponding to the unsupervised labels. This configuration corresponds to the Novel Class Discovery (*NCD*) setting. We will now discuss briefly, the related context of *NCD*, then delineate the objectives of the problem.

Semi-Supervised Learning (*SSL*) stands between supervised and unsupervised learning, utilizing a training dataset containing both labeled and unlabeled data. One of the main interests of SSL is that incorporating unlabeled data can significantly improve prediction quality compared to purely supervised learning approaches [160]. However, conventional *SSL* frameworks often assume that unlabeled data share the same labels as supervised data, which is not the case for our scenario. Adding unlabeled data without label overlap may introduce noise and conflicting attributes to standard classification models [161]. When there is no overlap between labels in supervised and unsupervised data, two approaches within *SSL* can address the classification problem. We briefly detail Zero-Shot Learning (*ZSL*) concepts and explain why it is not relevant in our case. Then we discuss *NCD* formulation and its objective.

The *ZSL* is the main approach for unsupervised data classification. This approach involves learning semantic attributes (e.g., word vectors [162]) from supervised data and then transferring this knowledge to the target dataset containing new classes and semantic attributes. For example, attributes learned from images of cars (e.g., shape, wheels) and airplanes (e.g., wings) can be utilized to predict whether a new image depicts a train based on its closer similarity to car attributes. However, *ZSL* classification is not directly applicable to scRNA Seq data due to the lack of semantic attributes.

The second approach is *NCD*, which leverages both supervised and unsupervised data to perform relevant clustering of the unsupervised data. Unlike *ZSL*, *NCD* doesn't require semantic attributes of the classes.

The objectives of *NCD* for scRNA Seq classification are multifaceted. Firstly, it aims to propose models capable of learning from both supervised and unsupervised data without compromising performance on generalization, ensuring accurate predictions for data with shared labels as those in the supervised set. Secondly, it seeks to conduct relevant clustering of unsupervised data, where each cluster corresponds to a distinct label [163]. By incorporating supervised data, the models should extract pertinent criteria for more meaningful and interpretable clustering compared to methods relying solely on pairwise similarity between unsupervised data [161]. This entails ensuring that each cluster accurately represents a unique cell type. The third objective involves mapping clusters to the cell lineage structure. Traditionally, identifying novel classes involves a two-step process of clustering and mapping clusters to classes based on prior knowledge. However, the mapping task which consists of identifying and assigning a label to each cluster in a lineage tree, becomes challenging. Assuming we have a correct criterion for cluster identification, testing all possible permutations to select the best one becomes computationally infeasible due to its factorial complexity in the number of labels, especially for large cell lineages with hundreds of cells. To address this challenge, one approach is to incorporate the hierarchical aspect directly into *NCD* or clustering models.

Overall, this problem of hierarchical novel class discovery for scRNA Seq data involves clustering, hierarchical classification, and cluster identification. This direct mapping of the scRNA Seq data to the cell lineage provides a solution for reconstructing cell state trajectories within a single framework.

1.3 Content of the thesis

This thesis is divided into three main parts, each of which is centered around the topic of the classification of scRNA Seq data. Chapter 2 provides a basis for the supervised and unsupervised learning. Chapters 4 and 5 are the contributions of this thesis and present the specific statistical learning problem of partially labeled data and novel class discovery, respectively.

Chapter 2 introduces a set of methods that serve as the basis to address the various statistical learning problems presented in this thesis. In the first section, we present a set of methods that belong to the state-of-the-art of the supervised multiclass classification problem, considering both hierarchical and non-hierarchical structures of the labels. The methods developed for supervised learning are studied in depth allowing a clear basis for their extension to the case of partially labeled learning in Chapter 4. In the second part of the chapter, we offer an overview of clustering problems, with a particular focus on two state-of-the-art algorithms. Similarly, these methods will be extended in Chapter 5 to address the novel class discovery problem.

Chapter 3 describes the artificial and experimental datasets used in the experiments of the following Chapters. The process of generating the artificial datasets is described in detail. In addition, the characteristics of the experimental datasets, including a description of their respective label hierarchies, are provided.

Chapter 4 studies the problem of partial label learning, combining the supervised and the partially supervised data. To address this challenge, we propose a set of methods that incorporate partial labeling extensions, complemented by hierarchical extensions. To fully explore these models, we create different learning task scenarios with different degrees of difficulty. We then carefully analyze and compare these methods, providing insights specific to each scenario. The results of this chapter have been recently published [164].

In Chapter 5, we tackle the problem of hierarchical Novel Class Discovery, which combines fully supervised and unsupervised data. The objective is to perform clustering of unsupervised data, and then to identify them as different cell types. To this end, we develop methods that integrate hierarchical classification and clustering aspects. We evaluate these methods using both artificial and experimental datasets, on both clustering and classification criteria. Preliminary results are provided.

Finally, we present a summary of the contribution and a discussion on the problem of classification of scRNA Seq data. Notably, we discuss more generally the applications, limits, and perspectives of this work of research.

2 Methods

Content

2.1 Multiclass classification 34
2.1.1 Multi-Class Classification 35
2.1.1.1 Tree Based Methods 36
2.1.1.2 Support Vector Machines 37
2.1.1.3 Prototype-based classification and regression 41
2.1.2 Hierarchical multi-classification 43
2.1.2.1 Hierarchical Support Vector Machines 45
2.1.2.2 Prototype-based method 46
2.2 Unsupervised Learning 46
2.2.1 *k-means* 47
2.2.2 Gaussian Mixture Model 48
2.3 Conclusion 52

We present in the following a short review of classification methods and of clustering methods. Rather than providing a wide spectrum of machine learning methods we have chosen to focus on a few methods that will serve as tools for analyzing our data and as the basis for developing our original methods in chapters 4 and 5 for the various supervision scenarios that we will deal with. In the first section, we introduce the multi-class classification problem and the hierarchical classification problem. In both cases, we present state-of-the-art methods for solving these problems. We will build on these methods to develop our original contributions for tackling partial label learning and hierarchical classification problems in Chapter 4. In the second section, we discuss state-of-the-art methods for two main unsupervised learning problems: clustering and density estimation. We detail the well-known *k-means* algorithm and the Gaussian Mixture Model (*GMM*). Again we focus on methods that will serve as a basis for the novel class discovery methods we will discuss in Chapter 5.

2.1 Multiclass classification

The task we are interested in is mono-label multiclass classification, meaning there is a single label for every sample (as opposed to multi-label, where samples may have multiple labels).

We adopt the formulation from [165], [166]: Given a labeled dataset $D = \{(x_i, y_i) \in \mathscr{X} \times \mathscr{Y}, 1 \leq i \leq N\}$, the goal is to train a model to predict the correct output for samples

in D while maintaining reliable predictions for unobserved data (generalization data). We will usually consider the case where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y}$ is the set of labels, with $|\mathcal{Y}| = C$ labels.

In general, the optimization problem results in minimizing an empirical loss, which is related to the percentage of classification errors on D. A model is seen as a parametric function $f_W : \mathcal{X} \mapsto \mathcal{Y}$ with parameters W, and learning aims at finding the function f_W, or equivalently the parameters W, which minimizes an objective function.

The objective function is usually defined as the averaged loss evaluated on the finite training set of supervised samples D, namely the empirical risk [167]. Learning aims at minimizing this objective function. It is defined as:

$$\min_{f_W} \sum_{i=1}^{N} loss\big(f_W(x_i), y_i\big) \tag{2.1}$$

The loss function ($loss : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$) describes how much the model fails to predict the right output for a labeled sample. Let's note the prediction (the output of the model) for i^{th} input sample x_i as $\hat{y}_i := f_W(x_i)$, the loss associated with the ith sample $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ is noted $loss(\hat{y}_i, y_i)$.

The goal of machine learning classification is to learn models that generalize well to unseen data, which may require complex classification models. Yet, overfitting likely arises when more powerful models with high learning capacities are used, meaning that the performance on the training set increases while the performance on unseen data (generalization data or test set) degrades. The main solution to prevent overfitting consists of adding what is known as a regularization term $\Omega(f)$ in the objective function, which changes the learning problem into:

$$\min_{f_W} \sum_{i=1}^{N} loss\big(f_W(x_i), y_i\big) + \lambda\Omega(f_W) \tag{2.2}$$

Adding such a regularization term, which usually penalizes the complexity of the model, introduces a bias on the solution towards a simpler model, which reduces the overfitting risk.

2.1.1 Multi-Class Classification

Many machine learning models may be used for multiclass classification, nearest neighbor, logisitic regression, decision trees, Bayes classifier, neural networks... [168]. We present in the following a selected view of a few standard approaches that we will build upon in the next chapters, namely tree-based methods, support vector machines (*SVM*), and prototype-based methods. Although deep learning and neural networks have become very popular and have shown impressive performance in many fields, we did not explore such methods because these usually require large amounts of structured data while we usually have moderate-size datasets of unstructured data.

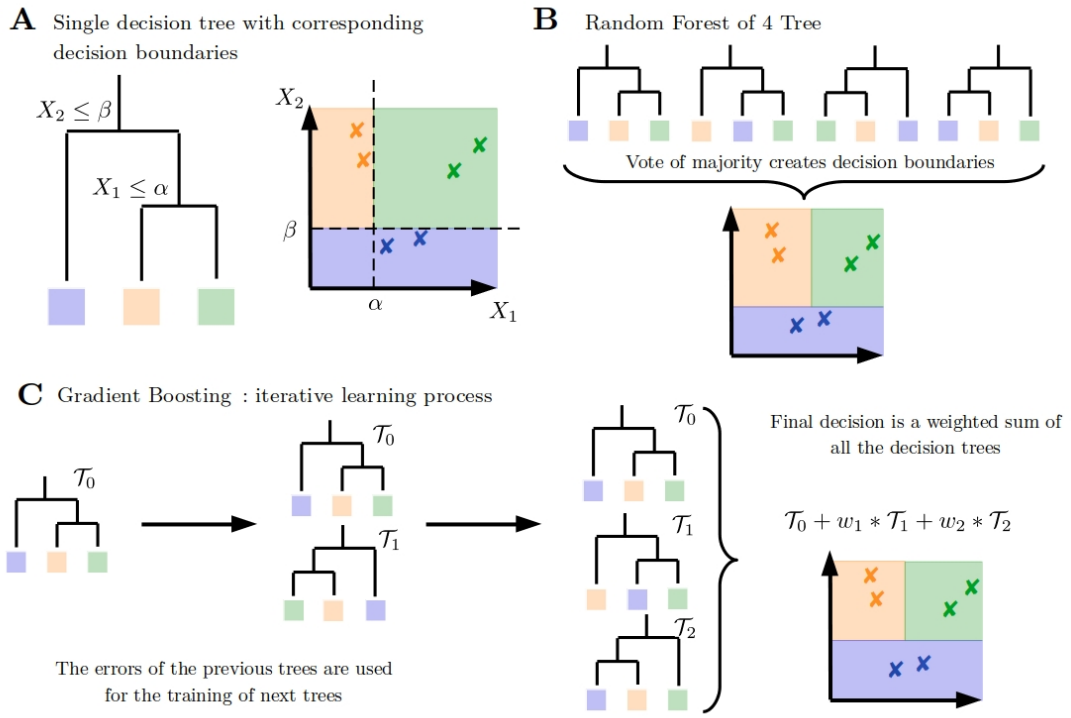

Figure 5 – **Tree-based methods**
A) Representation of a single decision tree method. On the left, the decision tree is a recursive top-down partitioning of the feature space to define a unique region for each class on the right. **B)** Random forest is the method based on a majority vote of multiple decision trees. **C)** Gradient Boosting algorithm is an iterative process where at each step, the new set of decision tree is trained according to the predictions made by the previous ones. The final result is given by the weighted sum of each learner, denoted as $\mathcal{T}_i$

2.1.1.1 Tree Based Methods

Tree-based methods such as Random Forest (*RF*) or Gradient Boosting (*GB*) are ensemble methods and rely on the aggregation of weak learners. In both methods, the decision is made according to a set of decision trees, but the voting process is different. These methods are very efficient for classification. We first describe how a single (weak) learner decision tree works and then explain *RF* and *GB*.

Decision tree, as introduced by [169], is a well-established technique for classification. The core concept is a top-down recursive binary partitioning of the feature space into distinct regions. Starting from the root node, which contains the entire training set, each node selects a feature and a partition value to generate left and right child nodes. After partitioning, the left/right child node contains training examples where the feature value is smaller/greater than the partition value. Typically, the feature and

the partition value are explored exhaustively across all feature values present in the training set to maximize an impurity reduction criterion based on the Gini index. An illustration of this partitioning is shown in Fig 5A.

Random forest is an ensemble method based on the "wisdom of crowd". It consists of a majority vote between several decision trees. Each learner is very sensitive to errors, but the set of errors of the learners compensate each other. *RF* uses the notion of bagging, which combines bootstrap sampling and aggregation. Bootstrapping consists of subsampling data for each learner, and aggregation is the concatenation of different trees made by a majority vote. Actually, *RF* relies on stochasticity at two different levels in the trees (through sampling the training set and the set of features that are used for learning each tree) to maximize their diversity and complementarity. Inference is performed by making all trees in the forest vote for a class (see Fig. 5B).

Unlike *RF* where all trees are independently trained, Gradient Boosting is a sequential approach where the learning of a tree takes into account the predictions and errors of previously trained trees. Moreover all learners contribute to the final decision, but their predictions are weighted according to their behaviour and accuracy, as opposed to Random Forest where all the learner's weights are equal (majority voting). This general training process is usually costly because of the sequential feature of the method. Dedicated methods with significant computational acceleration have been developed, such as Extreme Gradient Boosting Method (*XGBM*) [170] or Light Gradient Boosting Method (*LGBM*). [171].

2.1.1.2 Support Vector Machines

SVM are a prominent and widely used supervised classification approach in the field of machine learning, first introduced in [172]. This method is known for its effectiveness in various applications (including text classification, image recognition, and bioinformatics), its optimization scheme guarantees a global optimum, and strong generalization properties with the margin maximization formulation. While the *SVM* is originally a linear classifier, the well-known kernel trick allows the building of a nonlinear *SVM* classifier with desirable properties of learning a linear model (e.g. guaranteed global optimization).

Multi-class classification: A multi-class linear *SVM* is defined through a weight matrix, with C column vectors that stand for each label's classifier weights. We organize it as a weight matrix $W \in \mathbb{R}^{d,C}$. For each example x and for each class γ, one defines a score function $\text{score}(x,\gamma) = \langle w_\gamma, x\rangle$ and measures how much data x is recognized as being from class γ. Once a model is learned, a new prediction is made by selecting the class whose classifier maximizes the above score function:

$$\hat{y} = \underset{\gamma \in \mathcal{Y}}{\arg\max} \langle w_\gamma, x\rangle$$

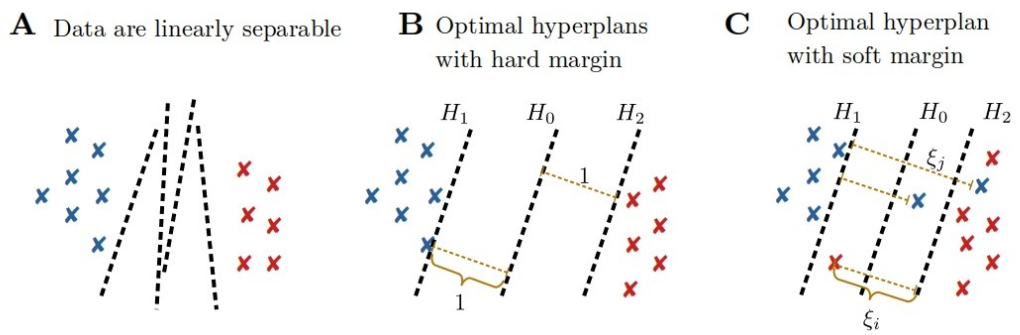

Figure 6 – **Support Vector Machines for binary classification**
A) Data are distributed according to two labels represented by colors. Data are linearly separable by hyper-plans (in dot) but there is an infinity of solutions. **B)** Optimal parallel hyper-plans H_0, H_1, H_2 define a separation hyperplane with a maximum margin for a better generalization. **C)** When the data are not linearly separable, one introduces slack variables ξ that allow releasing the constraints for outliers.

The learning aims at finding weights such that for every training sample, the score of the true label is greater than the score of other labels.

Hard margin: Introduced by [173], the main idea of margin is to maximize the difference between the correct score associated with the ground truth label and all other scores, plus a margin, which is shown to improve generalization. This is expressed as the following optimization problem.

$$\min_W \frac{1}{2}||W||^2$$
$$s.t. \forall i, \langle w_{y_i}, x_i \rangle - \max_{\gamma \neq y_i} \langle w_\gamma, x_i \rangle \geq 1$$

Soft margin: If the data are not linearly separable or if there are outliers in the data (as shown in Fig 6C), the hard margin formulation is no longer appropriate. A solution is obtained by relaxing the constraints by a small amount ξ, called a slack variable. This allows the model to make a few errors, i.e. violating original constraints for a few training samples, while being optimal for all remaining samples. One introduces one slack variable per training example. The optimization problem becomes:

$$\min_{W,\xi} \frac{1}{2}||W||^2 + \frac{\mu}{N}\sum_{i}^{N}\xi_i \qquad (2.3)$$
$$\forall i, \langle w_{y_i}, x_i\rangle - \max_{\gamma \neq y_i}\langle w_\gamma, x_i\rangle \geq 1 - \xi_i$$
$$\forall i, \xi_i \geq 0$$

The optimization problem still consists of maximizing the difference between the score from the correct classifier and all the other classifier scores, while minimizing the sum of all the slack variables. Here, μ is a hyperparameter that needs to be tuned, and it balances the regularization term on W and the sum of slack variables.

It is worth noticing that the sum of $\sum_{i}^{N}\xi_i$ is an upper bound of the number of classification errors so that optimizing the above optimization problem aims at minimizing this upper bound. Indeed, since the criterion must be minimized with respect to ξ_i, ξ_i may be expressed as:

$$\xi_i = max(0, 1 - (\langle w_{y_i}, x_i\rangle - \max_{\gamma \neq y_i}\langle w_\gamma, x_i\rangle))$$

Hence either the ith sample is well classified (i.e. $\forall \gamma \in \mathcal{Y}, \langle w_{y_i}, x_i\rangle > \langle w_\gamma, x_i\rangle$), then $\xi_i = 0$ if it is well classified with a margin or $0 < \xi_i < 1$ otherwise, or it is not well classified, then there exists γ such that $\langle w_\gamma, x_i\rangle > \langle w_{y_i}, x_i\rangle$ and $\xi_i \geq 1$. At the end, the sum of ξ_i is bigger than the number of classification errors.

There are two families of approaches to solving this problem. On the one hand, one may tackle the optimization problem in its primal form as in Eq. (2.3). On the other hand, one may use a dual formulation of the optimization problem and rely on quadratic programming solvers. These two optimization schemes are detailed below.

Dual optimization and the kernel trick The original formulation of *SVM* we described above is intrinsically linear. It is then limited to dealing with linearly separable data. The solution to extend *SVM* to nonlinearly separable data lies in nonlinearly projecting the data into a (usually) higher dimensional space, where we expect the data become linearly separable, and learn a linear *SVM* model in this new feature space. As we will see, this projection may be implicitly performed via a kernel operator, which fits well with the dual form of the optimization problem.

Dual Formulation

The multi-class soft margin *SVM* optimization can be solved in its dual form. We detail the case of binary classification (with labels $y_i = \pm 1$).

The dual form of the optimization problem may be obtained using the well-known Lagrangian mechanics, from the convex optimization field, applied to the constrained

optimization problem in Eq. (2.3). This constrained optimization problem is equivalent to the min-max optimization of the Lagrangian function $\mathscr{L}$ below, where a term has been added to the objective function for each of the constraints in Eq. (2.3):

$$\min_{w,\xi} \max_{\alpha \geq 0, \beta \geq 0} \mathscr{L}(w, \alpha, \beta) = \frac{||w||^2}{2} + \frac{\mu}{N} \sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \alpha_i (1 - z_i w^T x_i - \xi_i) - \sum_{i=1}^{N} \beta_i \xi_i \quad (2.4)$$

where $\alpha = (\alpha_i)_{1 \leq i \leq N}$ and $\beta = (\beta_i)_{1 \leq i \leq N}$ are Lagrange multipliers. This optimization problem may be shown equivalent to the following one, were the min and the max operators have been switched:

$$\max_{\alpha \geq 0, \beta \geq 0} \min_{w,\xi} \mathscr{L}(w, \alpha, \beta) = \frac{||w||^2}{2} + \frac{\mu}{N} \sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \alpha_i (1 - z_i w^T x_i - \xi_i) - \sum_{i=1}^{N} \beta_i \xi_i \quad (2.5)$$

One may then look for optimal parameters' values whatever α's and β's. Setting the partial derivative of $\mathscr{L}$ with respect to w and ξ_i to 0 yields, after a few classical computations, to the following dual optimization problem of *SVM*, which only depends on α's:

$$\max_{\alpha \leq \mu} = \sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i,j}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$
$$\max_{\alpha \leq \mu} = \alpha^T \mathbb{1} - \frac{1}{2} \alpha^T G \alpha$$

where $G \in \mathbb{R}^{N \times N}$ is the (positive semi-definite) Gram matrix wit $G_{i,j} = y_i y_j \langle x_i, x_j \rangle$, and $\mathbb{1}$ is the vector composed of ones. This objective is concave and can be globally optimized, usually with quadratic programming [174], or with gradient optimization [175].

There are two main interests of the dual formulation. The first comes from the fast optimization of the quadratic programming routines (QP). The second is the expression of the objective with pairwise dot product between training samples, which is well suited for kernel methods as we explain now.

The Kernel Trick: Assume one wants to nonlinearly project the data into a new dimensional space using a projection function Ψ and learn a linear *SVM* in this new feature space. One could derive a similar dual optimization problem as above where the Gram matrix terms would now be defined as $G_{i,j} = y_i y_j \langle \psi(x_i), \psi(x_j) \rangle$. One sees that the learning of the model involves dot product between pairs of training samples $\langle \psi(x_i), \psi(x_j) \rangle$ only.

Fortunately, there exists kernel functions $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ such that there exists a mapping Ψ from $\mathscr{X}$ to some feature space such that $K(u, v) = \langle \Psi(u), \Psi(v) \rangle$. Hence using such a K function one can learn nonlinear *SVM* without explicitly projecting

in the Ψ feature space. Usual Kernels include the Gaussian kernel $K(u,v) = e^{-\frac{||u-v||^2}{\sigma^2}}$ (which implicitly project in a infinite dimensional feature space Ψ) and the polynomial kernel $K(u,v) = (\langle u,v\rangle + c)^d$, where σ, c and d are parameters (not learned) of the kernels.

Note that solving the *SVM* learning problem in the dual is costfull as it requires at least to compute and store the Gram matrix G which is $O(N^2)$, preventing from using this approach for large scale problems.

Primal optimization. Alternatively one may learn a *SVM* by optimizing it in the primal. First as explained above the optimal slack variables of the optimization problem in Eq. (2.3) may always be expressed as:

$$\xi_i = max(0, 1 - (\langle w_{y_i}, x_i\rangle - \max_{\gamma \neq y_i} \langle w_\gamma, x_i\rangle)$$

Injecting the above expression in the optimization problem of Eq. (2.3) yields the following optimization problem:

$$\min_W \frac{1}{2}||W||^2 + \frac{\mu}{N}\sum_i^N max(0, 1 - (\langle w_{y_i}, x_i\rangle - \max_{\gamma \neq y_i} \langle w_\gamma, x_i\rangle) \tag{2.6}$$

The above objective function is differentiable almost everywhere, which enables gradient descent optimization. Unlike for the dual formulation, the optimization of the primal form in Eq. (2.6) may scale well with the training set size. Moreover, one may optimize a nonlinear (kernel) extension of *SVM* with this formulation through a kernel approximation method, that we discuss now.

Kernel approximation. Kernel approximation aims at overcoming the problem of computing and storing the full Gram matrix G when the number of training samples increases. A standard method lies in approximating the Gram matrix using the Nyström approximation [176]–[178] or directly approximating the high-dimensional embedding of the data [179], [180]. In the latter case, with the approximation of the embedding, one could solve the learning problem in either the primal or dual form. The choice of using a true kernel or a kernel approximation mainly depends on the size of the training dataset.

2.1.1.3 Prototype-based classification and regression

Prototype-Based (*PB*) classification aims at using a nearest neighbor classification rule amongst representative samples of the classes. Let us assume one uses a single representative sample per class $\phi \in \mathbb{R}^d$ and let us note $\phi_\gamma \in \mathscr{X}$ the representative sample of class γ (for instance it could be the mean of the samples from class γ in the training set). Then one may use a nearest neighbor rule in the set of classes'

representatives, which yields the following classification rule for a sample x:

$$\hat{y} = \underset{\gamma \in \mathcal{Y}}{\arg\min} ||x - \phi_\gamma|| \tag{2.7}$$

Many variants may be used, using multiple prototypes for a class, learning distance from the data rather than the Euclidean distance, etc. In any case, the main learning objective consists of optimizing the ϕ_γ vectors. A series of methods named Learning Vector Quantization (*LVQ*) [181], [182] have been proposed to do so, where the prototypes ϕ_γ are learned to optimize a discriminative criterion to achieve high classification accuracy. Moreover one may enrich the above strategy using the margin idea borrowed from *SVMs*. For instance [183] proposed a margin maximization criterion to optimize the classes' representatives. The idea is to enforce the model to satisfy constraints like the following:

$$\forall i \ \forall \gamma \in \mathcal{Y} \setminus \{y_i\} : ||\phi_{y_i} - x_i||_2^2 \leq ||\phi_\gamma - x_i||_2^2 - 1 + \xi_i \tag{2.8}$$

where ξ_i are slack variables that enable relaxing some of the constraints.

Another line of research focuses on using regression to perform classification. Assume we are given a set of class representative vectors $\Phi = \{\phi_\gamma\}_{\gamma \in \mathcal{Y}}$. The ϕ_γ's may or not belong to $\mathcal{X}$. Let's consider the ϕ_γ's belong to some space $\mathcal{U}$ which is not necessarily $\mathcal{X}$. One may perform classification through regression, where samples are projected into the $\mathcal{U}$, where the nearest neighbor rule on class representatives is applied as above to perform classification. Assuming a linear projection from $\mathcal{X}$ to $\mathcal{U}$, and naming W the projection matrix, the classification rule becomes:

$$\hat{y} = \underset{\gamma \in \mathcal{Y}}{\arg\min} ||Wx - \phi_\gamma|| \tag{2.9}$$

The learning problem may then be formalized as:

$$\underset{W}{\arg\min} \sum_i^N ||Wx_i - \phi_{y_i}||^2 \tag{2.10}$$

The ϕ_γ's may be chosen as one-hot-encoding vectors (in that case $\mathcal{U} = \mathbb{R}^C$) or may be derived from additional information about classes (we provide an example later in section 2.1.2.2 on Hierarchical classification). A natural extension of such a strategy is to use a nonlinear projection function (e.g. a neural network f_W with parameters W) so that the learning resumes now to:

$$\underset{W}{\arg\min} \sum_i^N ||f_W(x_i) - \phi_{y_i}||^2 \tag{2.11}$$

Going further, one may want to jointly optimize the projection and the class repre-

sentatives (for instance in [184]). The optimization problem then becomes:

$$\underset{W,\{\phi_\gamma\}_{\gamma\in\mathcal{Y}}}{\operatorname{argmin}} \sum_{i}^{N} ||f_W(x_i) - \phi_{y_i}||^2 \tag{2.12}$$

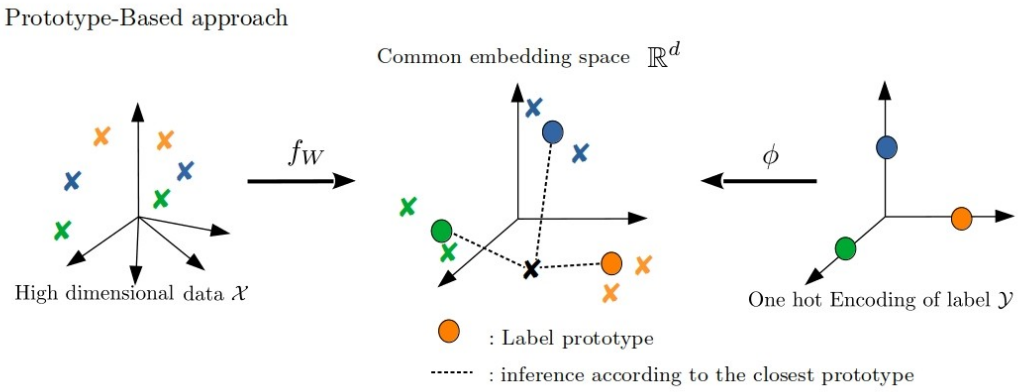

Figure 7 – **Prototype-based approach**
This classification methods approach consists of finding a common embedding of the data X and the label Y. The inference of new data is done according to the closest prototype. Different strategies are possible, either perform regression with a function f, only optimizing the prototype with the embedding ϕ, or combining these two operators.

2.1.2 Hierarchical multi-classification

Hierarchical classification stands for the classification task when the labels are organized in a hierarchical structure. Taking into account the hierarchy may be particularly relevant in scenarios where the classes (and the corresponding samples) exhibit a natural hierarchy, such as taxonomies or ontologies [185]. The hierarchy may encode some information like in the Wikipedia organization of the articles where a father-son link corresponds to a generalization-specialization relationship. Hierarchical classification has received considerable attention in domains such as image classification [186], text classification [187], speech recognition [188], Gene ontology building [189] and many more. Research efforts have been dedicated to specific application domains, addressing large-scale problems related to the very large number of classes (a setting often called extreme classification) [186], [187], [190]–[192].

Hierarchical classification includes many instances of classification problems as several features change the nature of the task. For instance, on distinguishes two main families of hierarchical classification problems, according to whether data may be labeled with all the nodes in the hierarchy (non-mandatory leaf node prediction) or if the labels correspond to leaf nodes only [193], which is called mandatory leaf

node prediction. One common hierarchical approach for the mandatory leaf node prediction, called top-down approach, consists of learning one classifier per node which decides which son a sample should be sent to. Classification of a sample starts at the tree's root node and recursively progresses in the tree based on the local node's classifier decision, to which son the sample goes next, till reaching a leaf node (a label) [185]. We will here focus on the non-mandatory leaf node classification configuration which corresponds to the problem we described in chapter 1 on classifying scRNA Seq data into all the nodes of the lineage tree. Usually methods that are proposed for such a setting are named big-bang (or global) classifiers [194].

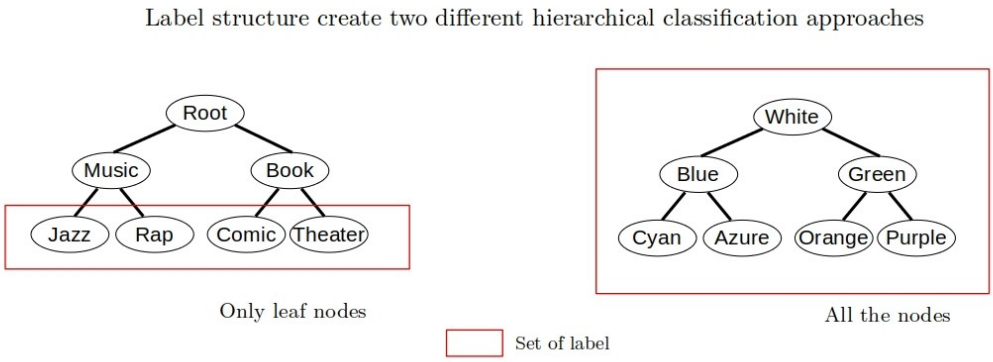

Figure 8 – **Example of hierarchical label structures**
The hierarchical label structure on the left is a mandatory leaf node structure where only the leaf nodes are target labels. This structure is suitable for top-down classification algorithms. The label structure on the right is the non-mandatory leaf node structure, where all nodes are labels.

While accuracy is the usual metric to evaluate the performance of a classifier, such a criterion is less informative about the behaviour and the capacity of the model when the number of labels is large and the accuracy low. A more informative criterion is given by the average distance (in the tree) from the true label to the predicted label for a test sample. In the following, we will consider that we are given a taxonomy of labels and a distance measure between labels which is derived from this taxonomy. We denote $H \in \mathbb{R}^{C,C}$, the (symmetric) matrix of the pairwise distances between labels. The distance between two labels $H_{\gamma,\eta}$ is small if the two labels are close (and null if $\gamma = \eta$) and large if the labels are very far from each other. When given a hierarchy or a tree of labels, standard practice is to define the distance between two labels as the length of the shortest path in the tree (the graph) from one label to the other.

Hence the criterion may be expressed as:

$$\mathscr{E} = \frac{1}{N} \sum_{i=1}^{N} H_{y_i, \hat{y}_i} \tag{2.13}$$

This metric may be used for evaluation, and for learning as well, several methods

have been proposed to minimize this criterion. In other words, the learning objective is to make the classifier predict a label which is, on average, as close as possible to the true label.

2.1.2.1 Hierarchical Support Vector Machines

Several extensions of the regular multi-class *SVM* to hierarchical classification have been proposed [195], [196]. We do not exhaustively review the existing literature on this topic but rather focus on a general framework proposed in [197], which is an instance of the more general structured output prediction framework of [166]. At first, recall that there is one slack variable per training sample in the formalization of multiclass *SVM* in Eq. (2.3). The main idea in [197] is to extend the *SVM* framework by considering that the constraints associated with a training sample (x_i, y_i) should be written by taking into account the distance between the true label and other labels. The more distant a label γ is from the true label y_i, the greater the difference should be between the scores associated to the two labels $(\langle w_{y_i}, x_i\rangle - \langle w_\gamma, x_i\rangle)$. This leads to the following optimization problem:

$$\min_{W,\xi} \frac{1}{2}||W||^2 + \frac{\mu}{N}\sum_i^N \xi_i \tag{2.14}$$

such that

$$\forall i, \forall \gamma \neq y_i, \langle w_{y_i}, x_i\rangle - \langle w_\gamma, x_i\rangle \geq 1 - \frac{\xi_i}{H_{\gamma,y_i}} \tag{2.15}$$

$$\forall i, \xi_i > 0$$

where the slack variable ξ_i is divided by the distance H_{γ,y_i} between the true label y_i and an alternative label γ.

An important feature of this optimization problem lies in that one may show that minimizing the above objective actually minimizes an upper bound of what we defined as a desirable criterion of performance, $\mathscr{E}$. Indeed, one easily shows that for each training sample:

$$\hat{\xi}_i = \max\left(0, \max_{\gamma \neq y_i} H_{\gamma,y_i} \times (1 - (\langle w_{y_i}, x_i\rangle - \langle w_\gamma, x_i\rangle)\right) \tag{2.16}$$

and reasoning as for the multiclass classification case yields the expected result.

As for multiclass *SVMs*, the above optimization problem can be solved in both its primal and dual forms. To get the primal form one may inject the expression of ξ_i in the objective and get an unconstrained objective function that may be minimized through e.g. gradient descent.

2.1.2.2 Prototype-based method

The prototype-based approach in 2.1.1.3 may be extended to hierarchical classification as well. The most intuitive way to do so is to make use of the label hierarchy to compute class representatives that might be used as in Eq. 2.7 on in any method from section 2.1.1.3. For instance, one might use a dimensionality reduction technique such as Multidimensional Scaling (*MDS*) which is designed to infer new coordinates of samples in a low dimensional space, that preserve the pairwise distances between samples in their original space [184]. The *MDS* solutions would be given by:

$$\underset{\Phi \in \mathbb{R}^{C \times d}}{\arg\min} \sum_{\gamma,\eta \in \mathscr{Y}^2} |H_{\gamma,\eta} - ||\phi_\gamma - \phi_\eta||^2| \tag{2.17}$$

where Φ is the set of prototypes in a low dimensional space (likely $d \ll C$).

Like in the flat (non hierarchical) case, one may want to maximize the margin to increase the discriminative capacity of the model. Generalizing the idea in Eq. 2.8 with the margin scaling of *SVMs* in Eq. 2.15, the method from [184] proposed to use the following optimization problem (in its constrained form):

$$\min_{W,\xi} \sum_{i=1}^{N} ||Wx_i - \phi_{y_i}||^2 + \lambda ||W||_F^2 + \sum_{i}^{N} \xi_i \tag{2.18}$$

such that

$$\forall i \ \forall \gamma \in \mathscr{Y} \setminus \{y_i\}$$

$$||\phi_{y_i} - Wx_i||_2^2 \leq ||\phi_\gamma - Wx_i||_2^2 - H_{y_i,\gamma} + \xi_i \tag{2.19}$$

The optimization may be performed in the primal form by building the unconstrained primal objective function, by noticing (like what we did before) that the slack variables ξ_i may be expressed as: $\xi_i = \max(0, \max_\gamma ||\phi_{y_i} - Wx_i)||_2^2 - ||\phi_\gamma - Wx_i||_2^2 + H_{y_i,\gamma})$

2.2 Unsupervised Learning

One of the most common applications of scRNA Seq data analysis is cell type clustering. Since manual annotation is not a reliable strategy because of the difficulty of the task and because of the size of the datasets, current strategies rely on the two main categories of unsupervised learning, which are clustering and manifold learning [166]. Specifically, many state-of-the-art methods consist of a clustering method which is applied on a low dimensional embedding of the high-dimensional scRNA Seq data [149], [198], [199]. We first recall the clustering problem in general, then we detail two main state-of-the-art methods which are the k-means and the Gaussian Mixture Model (*GMM*), these methods are the basis of the approaches we will develop for the novel class discovery problem which we will detail in Chapter 5.

Clustering is the process of categorizing and gathering data by similarity. Roughly speaking if samples share common features, they should belong to the same cluster. In bioinformatics and especially in scRNA Seq data analysis, one of the main goals is to perform clustering on data to categorize cell states, discover new cell types, and refine knowledge about existing ones.

Given an unlabeled dataset $D = \{x_i \in \mathbb{R}^d; i = 1, ..., N\}$, the objective is to find a partitioning $\{P_1, ..., P_K\}$ of the data into K clusters.

2.2.1 *k-means*

We now present the *k-means* algorithm and mainly follow the presentation in [168]. The *k-means* algorithm is likely the most well-known clustering algorithm. It takes as input the number of clusters and a dataset of unlabeled samples and outputs a partitioning of the data in K clusters. Like many unsupervised methods and particularly clustering algorithms, the *k-means* algorithm is built on an specific assumption about the data distribution. The *k-means* assumption is that the data in every cluster follow more or less an isotropic (same dispersion in every direction) Gaussian distribution. The criterion that the *k-means* aims to minimize is the following:

$$\underset{\mu_1,...\mu_k}{\arg\min} \sum_{k=1}^{K} \sum_{i=1}^{N} \delta_{i,k} ||x - \mu_k||^2 \tag{2.20}$$

where $\delta_{i,k}$'s encode to which cluster a sample x_i belongs:

$$\delta_{i,k} = \begin{cases} 1 & \text{if } k = \arg\min_j ||x_i - \mu_j|| \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

and μ_k is the mean of the samples assigned to cluster k:

$$\mu_k = \frac{\sum_{i=1}^{N} \delta_{i,k} x_i}{\sum_{i=1}^{N} \delta_{i,k}} \tag{2.22}$$

The parameters of the *k-means* algorithm, Θ, are the clusters' means $\Theta = \{\mu_k \in \mathbb{R}^d, k = 1, ..., K\}$. The general clustering problem is usually not directly solvable but the Lloyd algorithm [200] allows to solve the *k-means* learning problem efficiently. It is an iterative algorithm, which is an instance of the Expectation Maximization algorithm (EM) which we will detail later. Here is a sketch of the algorithm:

1. Initialize the centroids $\Theta^{(0)}$, $t = 0$
2. Repeat until convergence
   — Partition the data according to the current model's parameters $\Theta^{(t)}$: Assign each sample to the cluster with the closest centroid:

$$\forall (i,k) \in [\![1, N]\!] \times [\![1, K]\!], \delta_{i,k}^{(t+1)} = \begin{cases} 1 & \text{if } k = \arg\min_j ||x_i - \mu_j^{(t)}|| \\ 0 & \text{otherwise} \end{cases}$$

— Compute the centroid of all clusters using the current partitioning of the data:

$$\forall k \in [\![1,K]\!], \mu_k^{(t+1)} = \frac{1}{\sum_i \delta_{i,k}^{(t+1)}} \sum_{i=1}^{N} \delta_{i,k}^{(t+1)} x_i$$

— $t = t+1$

The learning algorithm is highly dependent on the initialization and does not explore the entire landscape of the solutions. Yet the algorithm usually quickly converges and it may be shown to converge to a local minimum. Many variants of the *k-means* algorithm have been proposed, including fuzzy variants, or alternative initialization schemes (e.g. *k-means++* [201]) to avoid critical solutions, etc. The *k-means* algorithm remains today a well-performing and state-of-the-art clustering method, it is included in many pipelines for clustering scRNA Seq data [202]–[205].

2.2.2 Gaussian Mixture Model

The Gaussian Mixture Model (*GMM*) is a probabilistic model that assumes the data follow a mixture of Gaussian distributions. The *GMM* is not a clustering method per nature but may be used for this task. Once it is learned it may be used to recover from which component each sample was drawn. We will see *GMM* may be seen as a probabilistic generalization of the *k-means* algorithm. Noting Θ the set of parameters of the *GMM*, its density function at x is defined as:

$$p(x|\Theta) = \sum_{k=1}^{K} \pi_k p(x|\Theta_k) \tag{2.23}$$

where the $\pi_k \equiv p(k)$ are prior probabilities of the components of the mixture, and the component distributions are Gaussian distributions:

$$p(x|\Theta_k) \sim \mathcal{N}(x; \mu_k, \Sigma_k) \tag{2.24}$$

where μ_k and Σ_k are the mean and the covariance matrix of the kth Gaussian distribution in the mixture. The parameters of a *GMM*, Θ, include the priors and the parameters of the Gaussian components $\Theta = \{(\pi_k, \mu_k, \Sigma_k), k = 1, ..., K\}$.

The learning of a *GMM* is usually performed by looking for the parameters that maximize the likelihood (or more conveniently the log-likelihood) of the data, it is the well-known Maximum Likelihood Estimation (MLE) strategy. Given a training dataset $D = \{x_1, ..., x_N\}$ the log-likelihood reads:

$$\log p(D|\Theta) = \sum_{i=1}^{N} \log p(x|\Theta) \tag{2.25}$$

and the maximum likelihood estimation resumes to finding $\hat{\Theta}$ such that:

$$\hat{\Theta} = \arg\max_{\Theta} \log p(D|\Theta) \tag{2.26}$$

Solving this optimization problem is not straightforward. One usually relies on the EM algorithm [206]. The idea behind the EM algorithm is that, if one knew by which component each training data was sampled, the learning of the parameters would be immediate. The EM algorithm then iterates the estimation of these hidden variables and the reestimation of the parameters, given the guessed values for the hidden variables. It is shown to converge to a local optimum of the likelihood.

EM derivation in the general case. We now describe in detail the EM algorithm in the general case and its instance to the specific case of learning a *GMM*. The objective is to find parameters Θ that maximize the log-likelihood given by $\log p(D|\Theta)$. We assume that the data are partially observed and that there exist hidden (discrete) variables $Z = \{z_1, ..., z_N\}$ such that the joint distribution of D and Z may be expressed as:

$$p(D, Z|\Theta) = p(Z|D, \Theta)\, p(D|\Theta) \tag{2.27}$$

which gives:

$$\log p(D|\Theta) = \log p(D, Z|\Theta) - \log p(Z|D, \Theta) \tag{2.28}$$

where $Z = (z_1, ..., z_N)$ encodes the clustering of the data, with $z_i \in \{1, ..., K\}$ being the number of the cluster assigned to sample x_i, and $F(D, Z) = F(x_1, z_1, x_2, z_2, ...)$ is the criterion which should be minimized.

To make things clearer the Z variables in the case of learning a Gaussian mixture would be the numbers of the components from which samples in D were drawn. For clarity, we denote X the distribution of the data, equivalent to D in the unsupervised learning case.

Let us consider we are given some estimation of the parameters, that we note $\Theta^{(t)}$, indexing with t because $\Theta^{(t)}$ will play the role of the estimate at iteration t later on. Let us now take the expectation, of both sides of the equation, over the possible values of the unknown data Z under the current parameters' estimate $\Theta^{(t)}$:

$$\sum_Z p(Z|X, \Theta^{(t)}) \log p(X|\Theta) = \sum_Z p(Z|X, \Theta^{(t)}) \log p(X, Z|\Theta) - \sum_Z p(Z|X, \Theta^{(t)}) \log p(Z|X, \Theta) \tag{2.29}$$

The first term in the left-hand side simplifies as $\log p(X|\Theta)$ is independent of Z and since $\sum_Z p(Z|X, \Theta^{(t)}) = 1$. The first term of the right-hand side of the equation is usually named auxiliary function in the EM literature, it is noted $Q(\Theta|\ \Theta^{(t)})$ and defined as:

$$Q(\Theta|\ \Theta^{(t)}) = \sum_Z p(Z|X, \Theta^{(t)})\, log\, p(X, Z|\Theta)$$

Finally, the second term of the right-hand side of the equation is the cross entropy $H(\Theta|\ \Theta^{(t)}) = -\sum_Z p(Z|X, \Theta^{(t)}) \log p(Z|X, \Theta)$.

Finally, one gets:

$$\log p(X|\Theta) = Q(\Theta|\ \Theta^{(t)}) + H(\Theta|\ \Theta^{(t)}) \tag{2.30}$$

which holds whatever Θ. In particular, for $\Theta = \Theta^{(t)}$ we get:

$$\log p(X|\Theta^{(t)}) = Q(\Theta^{(t)}|\,\Theta^{(t)}) + H(\Theta^{(t)}|\,\Theta^{(t)}) \tag{2.31}$$

Combining (subtracting) the two last equations yields:

$$\log p(X|\Theta) - \log p(X|\Theta^{(t)}) = Q(\Theta|\,\Theta^{(t)}) - Q(\Theta^{(t)}|\,\Theta^{(t)}) + \underbrace{H(\Theta|\,\Theta^{(t)}) - H(\Theta^{(t)}|\,\Theta^{(t)})}_{\substack{\geq 0 \\ \text{from Gibbs inequality}}} \tag{2.32}$$

Assume now that $\Theta^{(t)}$ is fixed. If one wants to find Θ that maximizes $\log p(X|\Theta)$ (equivalently maximizing $\log p(X|\Theta) - \log p(X|\Theta^{(t)})$), it resumes to maximize $Q(\Theta|\Theta^{(t)})$, (equivalently to maximize $Q(\Theta|\Theta^{(t)}) - Q(\Theta^{(t)}|\Theta^{(t)})$).

This result is the basis of the *M*- step of the EM algorithm. Let's consider that at some iteration t of the algorithm, the model's parameters are stored in $\Theta^{(t)}$. To find a new and better estimate, $\Theta^{(t+1)}$, one may maximize $Q(\Theta|\Theta^{(t)})$, whose maximum is necessarily positive. If the maximum is found (which is the case since Q is concave in many cases, including the *GMM* case), this implies that $\log p(X|\Theta^{t+1}) \geq \log p(X|\Theta^{(t)})$. Hence the EM provides a mechanism that ensures that the log-likelihood of the data increases at every iteration (hence converges).

At the end, the EM algorithm may written as:

1. Initialize $\Theta^{(0)}$, $t = 0$
2. Iterate until convergence
   — *(E-step):* Compute the distribution of the hidden variables given the model's parameters: $p(Z|X,\Theta^{(t)})$
   — *(M-step):* Re-estimmate the model's parameter according to:

$$\Theta^{t+1} = \arg\max_{\Theta} Q(\Theta|\,\Theta^{(t)}) \tag{2.33}$$

   with:

$$Q(\Theta|\,\Theta^{(t)}) = \sum_{Z} p(Z|X,\Theta^{(t)}) \, log\, p(X,Z|\Theta) \tag{2.34}$$

   — t=t+1

Learning a *GMM* with the EM algorithm Let us look at the auxiliary function $Q(\Theta|\,\Theta^{(t)})$ in the case of a *GMM*, and decompose the log-likelihood of complete data:

$$log\, p(X,Z|\Theta) = log\Big(p(Z|\Theta)\, p(X|Z,\Theta)\Big) \tag{2.35}$$

$$= log\Big(\prod_{i=1}^{N} p(z_i|\Theta) \prod_{i=1}^{N} p(x_i|z_i,\Theta)\Big) \tag{2.36}$$

where $p(x_i|z_i,\Theta)$ stands for the density function of the z_i^{th} Gaussian component. One may solve the M-step and derive analytical solutions in the *GMM* case. The EM

algorithm writes in this case:

1. Initialize $\Theta^{(0)}$; $t = 0$
2. Iterate
   — *(E-step):* Compute the posterior distribution $p(Z|X,\Theta^t)$ with the current parameters. We note w_i^k the posterior $w_i^k = P(z_i = k|x_i,\Theta^{(t)})$.

$$w_i^k = \frac{\pi_k \mathcal{N}(x_i|\mu_k^t,\Sigma_k^t)}{\sum_j \pi_j \mathcal{N}(x_i|\mu_j^t,\Sigma_j^t)} \tag{2.37}$$

   — *(M-step):* Maximize the auxiliary function $Q(\Theta,\Theta^t)$ with respect to Θ to get $\Theta^{(t+1)}$. One gets:

$$\pi_k^{(t+1)} = \frac{1}{N}\sum_i^N w_i^k \tag{2.38}$$

$$\mu_k^{(t+1)} = \frac{\sum_i^N w_i^k x_i}{\sum_i^N w_i^k} \tag{2.39}$$

$$\Sigma_k^{(t+1)} = \frac{\sum_i^N w_i^k (x_i-\mu_k^t)(x_i-\mu_k^t)^T}{\sum_i^N w_i^k} \tag{2.40}$$

   — $t = t+1$

Here are some comments on this algorithm. Coming back to the *k-means* algorithm, one may see that the learning algorithm for *k-means* is close to an EM algorithm. Actually, it is a hard assignment variant of the EM algorithm where each data point is assigned to a unique cluster at each iteration rather than taking into account the ambiguity about the cluster one data sample belongs to (through the posterior probabilities). Moreover, if one sets all the covariance matrices in a *GMM* to the identity matrix, then *k-means* and *GMM* are even closer. The only remaining difference is that their learning algorithms are two different instances of the EM algorithm.

While *GMMs* appear as a generalization of *k-means* with increased capacity (modeling dispersion clusters with covariance matrices) it is often harder to learn *GMMs* because of numerical issues arising with small variance estimation. Typically, it is advisable to consider only diagonal covariance matrices to overcome these challenges and, in any case, a good initialization is even more crucial for *GMMs* than for *k-means* so that *GMM* initialization is often performed with the *k-means* algorithm. Like *k-means*, the optimization of *GMM* may converge to a local extremum. There are several strategies to mitigate this, such as running several *GMM* (e.g. bootstrapping [207]).

GMM is a state-of-art model and a powerful tool for clustering, in many domains, in particular for scRNA seq [102], [116], [208]. *k-means* and *GMM* will serve as baseline methods for the Novel Class Discovery study in Chapter 5.

2.3 Conclusion

In this chapter, we introduced essential methods and frameworks that will serve as basis to develop our contributions in the following chapters. On the first part we dealt with supervised learning and focused on two key aspects: the multi-class classification problem and the hierarchical classification problem. We particularly presented the support vector machines (*SVM*) and the prototype-based methods which are methods of choice for which we will propose extensions for our particular classification tasks in chapter 4. We will, in particular, explore extensions of these methods to partial label learning in the next chapter and discuss non nonlinear extensions, as real-world datasets often exhibit complexities beyond those addressed by linear classification approaches.

In the second part of this chapter, we presented state-of-the-art clustering and density estimation methods, namely k-means, and *GMM*. We showed how k-means may be viewed as an instance of *GMM* where both are solved with the EM algorithm. These methods will exploited in Chapter 5 for Novel Class Discovery (*NCD*) when the classes are organized in a hierarchy. We will develop specific extensions of the methods by introducing regularization terms based on the knowledge about the hierarchical organization of the labels.

3 Modalities and Dataset Presentation

Content

3.1 Simulated Datasets 53
3.1.1 Hierarchical label structure 53
3.1.2 Model parameters 54
3.1.3 Simulated datasets 54
3.2 Experimental Datasets 55
3.2.1 *C. elegans* 55
3.2.2 *S. mediterranea* 55
3.2.3 *Myeloid progenitors* 56
3.3 Distance between labels 56

In the next chapters, we use several datasets, that have been obtained experimentally or simulated.

3.1 Simulated Datasets

We used the prosstt library to simulate single-cell RNA Seq data in various conditions [209]. We chose to work with this particular library because of its ability to simulate differentiation processes and implement differentiation trajectories with various topologies.

3.1.1 Hierarchical label structure

First, we design the topology of the label set. To highlight the impact of the topology of the label set on the classification results, we chose to generate three different scenarios:

— The *Linear* case corresponds to a succession of nodes where there is no division or bifurcation. The number of labels is in that case directly equal to the depth of the tree.

— The *Branches* case corresponds to an asymmetric situation where when there is a division or bifurcation, one branch will continue as the linear topology and the other one will continue dividing. The number of labels is in that case given by $\sum_{i=0}^{t+1} i$ where t is the depth of the tree.

— The *Binary* case corresponds to the extreme case where each branch is dividing into two new branches forming a binary tree. The number of labels is given by $\sum_{i=0}^{t+1} 2^i$ where t is the depth of the tree.

3.1.2 Model parameters

The generation of datasets depends on a set of 3 parameters, that we describe here.

Topology

The topology is defines the hierarchical label structure. One of three distinct topologies is selected, namely *Linear*, *Branches*, or *Binary*, and, in parallel, the depth of the tree is set.

Number of genetic programs

In the prosstt library [209] the evolution of gene expression across the differentiation trajectories is modeled using genetic programs that are in practice random walks on a tree. One of the main hyperparameters is the number of intrinsic genetic programs, denoted g. Because having more genetic programs increases the combinatorics of gene expression in each branch, we anticipate that a higher number of programs would lead to more differences between the various parts of the tree and, hence easier classification problems. We set this number to 10 and 50. This number allows us to define the degree of differentiability between the subset of the topology. It can be seen that each branch in the tree is a specific combination of all the genetic programs. For 10 programs, independently of the topology, it should not be enough to differentiate efficiently the label, whereas for 50 programs, it should be enough to fully differentiate the topology branches and partially the topology binary. To sum up, the more is programs, the easier is to identify a cell-type and on the contrary, a few number of programs imply more difficult to differentiate cell-type.

Degree of noise

Finally, the last parameter we use to generate the datasets is the coefficient α which controls the characteristics of the negative binomial probability distribution that is used to generate realistic gene expression [209]. We chose for $\alpha = 0.1$ which corresponds to low levels of noise and $\alpha = 0.5$ which corresponds to high levels of noise.

3.1.3 Simulated datasets

Using these three parameters (topology, g, and α), we generated 12 datasets, that we used in Chapter 4. 4 datasets for each topology, by varying the degree of noise and number of genetic programs. To get comparable results, we fixed the number of labels to 255 for the *Linear* and *Binary* topologies and to 253 for *Branches*. In Chapter 5, we generate only 3 datasets, using the same set of parameters. The Table 3.1, provides a summary of all the simulated datasets used in the next chapters.

		Number of label	Number of sample	Number of feature	Noise α	Genetic program g
Chapter 4	*Linear-254*	254	355600	50	0.1 - 0.5	10 - 50
	Branches-21	253	354200	50	0.1 - 0.5	10 - 50
	Binary-7	255	355600	50	0.1 - 0.5	10 - 50
Chapter 5	*Branches-7*	36	2800	50	0.1	50
	Binary-4	31	1240	50	0.1	50
	Binary-6	131	5000	50	0.1	50

Table 3.1 – **Characteristics of the simulated datasets**

3.2 Experimental Datasets

3.2.1 *C. elegans*

The first experimental dataset comes from an atlas of development for the nematode *C. elegans* at single cell resolution [150]. It initially consists of 89701 vectors of dimension 20222, where each coordinate corresponds to the expression level of a gene. We reduced the dimension to 3842, by selecting a representative subset of genes corresponding to transcription factors obtained from the gene ontology extracted from Wormbase [135]. The labels have a tree structure that is derived from the fact that they encode the successive cell divisions in the embryonic development of the model organism *C. elegans* from the initial cell to the adult. If we consider the complete tree, there are 1342 possible labels [6] which are the cell names in the cell division history. Among the 89701 transcriptomic profiles, 5143 have been annotated with a unique label (6 % of the whole dataset), 42631 have been annotated with at least two labels (47 % of the whole dataset) and 41927 have no labels (47 % of the whole dataset). In our experiments, we used only the 5143 transcriptomic profiles that have a unique label, the associated labels cover 95 labels among the 1342 labels of the whole hierarchy.

3.2.2 *S. mediterranea*

The second experimental dataset comes from an atlas of development of the planaria *S. mediterranea* at single cell resolution. We use as label hierarchy, the lineage tree obtained in [210] using the PAGA method [99]. The dataset consists of 21612 transcriptomic vectors of dimension 28065, distributed among 51 different labels. The labels represent a collection of cell types arranged hierarchically based on cell differentiation. Specifically, the structure is a tree with pluripotent stem cells gathered at the root; and descendant nodes representing differentiated adult cell types. In that case, the lineage tree is not deep, most branches have a length between 2 and 3 labels, and the largest is 6. The original dataset is available at https://shiny.mdc-berlin.de/psca/. In order to have a uniform distribution of transcriptomic profiles per labels we subsampled the set of samples associated to labels have the highest number of examples (such as “neoblast1”, which has 6343 examples initially). This leads us to

7595 transcriptomic profiles.

3.2.3 *Myeloid progenitors*

The third experimental dataset comes from an atlas of hematopoiesis at single cell resolution [211]. This dataset is available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE72857GSE72857. It contains 2730 transcriptomic profiles of dimension 10719. Data are distributed among 26 labels in total. We directly use the set of labels provided by [211]. This set of labels corresponds to the different cell types in blood cells and is organized in the hierarchical structure corresponding to the different states of cell differentiation [9], [10].

Following standard practice, PCA projection was applied to the three datasets using the first 100, 50, and 100 components for *C. elegans*, *S. mediterranea* and *Myeloid progenitors*, respectively. For the two last datasets, PCA components were directly obtained from public available sources [110].

	Number of label	Number of sample	Number of feature (post PCA)	Label structure
C. elegans	1342	5143	50	Lineage Tree
S. Mediterranea	51	7595	50	Hierarchical clusters
Myeloid Progenitor	26	2730	100	Hierarhicical clsuters

Table 3.2 – **Experimental dataset characteristics**

3.3 Distance between labels

For each dataset, we begin with the label structure selected, either the lineage tree for the artificial dataset and *C. elegans*, or the label hierarchy for the two other experimental datasets. We then create a graph according to these structures, where each node is a cell type and the edges encode the genealogical relationships between these for the lineage trees or the hierarchical relationship between the cell types. Finally, we calculate the distance between two labels according to the shortest path in the graph. The pairwise distance between labels is encoded as a matrix $H \in \mathbb{R}^{C,C}$, where C is the number of labels.

4 Partial label learning for classification of scRNA Seq data

Content

4.1 Introduction . . . 58
4.2 Methods . . . 60
4.2.1 Task formalization . . . 60
4.2.2 Learning strategies . . . 62
4.2.2.1 Partially labeled setting . . . 62
4.2.2.2 Hierarchical classification . . . 65
4.2.2.3 Partially labeled hierarchical classification . . . 65
4.2.3 Models . . . 66
4.2.3.1 Multiclass *SVM* . . . 67
4.2.3.2 Prototype-based approach . . . 69
4.2.3.3 *kNN* . . . 71
4.3 Datasets and experimental settings . . . 72
4.3.1 Datasets . . . 72
4.3.2 Experiments . . . 72
4.3.2.1 Partially labeled datasets . . . 72
4.3.2.2 Scenarios . . . 73
4.3.2.3 Tasks . . . 73
4.3.2.4 Metrics . . . 73
4.4 Results . . . 75
4.4.1 Supervised setting . . . 75
4.4.2 Partial label, Overlap = 1 . . . 77
4.4.3 Partial label, Overlap = 0 . . . 79
4.4.4 Real-world datasets . . . 81
4.4.5 Incorporating hierarchy improves performance . . . 86
4.4.6 Supplementary Results . . . 89
4.4.6.1 Comparison of Algorithm *IRL* and *IFR* . . . 89
4.4.6.2 *SVM* implementation and kernel approximation . . . 91
4.5 Discussion . . . 93
4.6 Code availability . . . 94

In this Chapter, we propose to take advantage of partial annotations for the task of single-cell RNA Seq classification. The partial label learning framework assumes that we can obtain a set of candidate labels containing the correct one for each data point, a simpler setting than requiring a fully supervised training dataset. We study and extend when needed state-of-the-art multi-class classification methods, such as *SVM*, *kNN*, and *PB* methods, to the partial label learning framework. Moreover, we study the effect of incorporating the structure of the label set into the methods. We show, on both simulated and real experimental, that these extensions enable to learn from partially labeled data. Finally, we study the level of uncertainty present in the partially annotated data, and derive some prescriptive results on the effect of this uncertainty on the accuracy of the partial label learning methods. The results of this chapter have been published [164].

4.1 Introduction

The field of developmental biology aims to unravel the mechanisms that underlie the transformation of a single-cell into a complex multicellular organism. This process encompasses molecular, cellular, and tissue-level dynamics. To measure this diversity of processes, several complementary acquisition methods are available [33]. In particular scRNA Seq methods make it possible to establish a transcriptomic profile for each cell, i.e. a measure of the quantity of RNA associated with each gene [82]. In practice, this transcriptomic profile is, for each cell, a vector of high dimension (generally of the order of 20,000: the number of genes considered), such that each of the coordinates of this vector represents the number of RNA molecules measured associated with a given gene. Because of their high dimensionality, scRNA seq datasets cannot be explored by hand. Therefore several methodological approaches have been proposed to make sense of these datasets. Particularly, several nonlinear dimension reduction methods have been developed, with the aim of preserving the structure of the data and allowing the identification of subcategories with a biological rationale [212]–[214]. In addition, a number of methods aim at reconstructing how these data points evolve over time, through the task of trajectory inference [93], [95]. Moreover, some works have sought to infer the structure of these datasets by making explicit their link with the spatial organization of cells [96]. Overall, scRNA Seq datasets have proven very useful to answer a variety of questions in particular in developmental biology [150], [215]. The main remaining limitations come from the fact that transcriptomic profiles do not usually contain temporal and precise spatial information [216], and manual annotation of individual cell profiles can be tedious, time-consuming and require expert knowledge [39].

Because of the rapid accumulation of data, one of the major challenges currently is to perform accurate automated annotation of single-cell RNA Seq datasets into accurate cell types [34], [38]–[40]. Annotation strategies rely on the establishment of large atlases and expert knowledge, as well as transfer learning strategies for the use of already annotated datasets to newly generated ones. While often gene markers

can be used to identify unambiguously a given cell type, in some cases, the annotation remains partial with an ambiguity that cannot be resolved by known *a priori* information [150].

We propose in this paper to take advantage of partial label learning strategies for the problem of automated single-cell transcriptomics profile annotation. The idea of partial label learning [217] consists in the task of multiclass classification, where, instead of a single label per sample, a *set* of candidate labels is given per sample, among which only one is correct. One of the application of this approach consists in the case when even with expert knowledge on known marker genes, it is not possible to perform single-cell annotation as it is the case for *C. elegans* [150]. The second class of applications concerns the situation where a reference atlas exists, providing sufficient information for establishing sets of candidate labels for each transcriptomic profile, but not sufficient for a single-cell resolution annotation (because of time, cost, and computational limitations). In this paper, we establish and address the challenge of resolving partially annotated single-cell transcriptomics profile by solving a partial label learning problem. We compare various implementations of the problem that we test both in simulated data and experimental data. We focus in particular on the question of labels with a hierarchy, such as the ones that can be found in developing and differentiating systems [39], [59], [72], [110], [218], [219]. This leads us to a benchmark of partial hierarchical labeling methods adapted from classical approaches to the problem of partial labels.

From the machine learning point of view, the problems we want to solve are hierarchical classification problems with partially labeled training data. Partial label learning has been studied in a variety of real-world learning scenarios, including automatic face naming [220], [221], web mining [222], and multimedia content analysis [221]. Researchers have proposed various approaches to tackle the partial label learning problem using well-known machine learning techniques. These techniques include maximum likelihood estimation [223], convex optimization [217], k-nearest neighbors [220], [223] and more. While maximum margin methods (e.g. *SVM*) are a powerful solution to such a partial labeling setting [224], [225], we will investigate prototype-based methods, nearest neighbor, logistic regression and ensemble methods as well to provide a comparative study of a wide spectrum of methods.

Similarly, hierarchical classification has been addressed by many approaches in the machine learning community and in particular for extreme classification (hierarchical classification with a very large number of labels) [192]. Hierarchical classification has been intensively studied for image data [186] and textual data [187]. Yet the research in this area has mainly focused on specific application contexts for these data, large-scale problems (in the number of classes and in the number of data) [186], [187], [190]–[192], hierarchical multilabel classification, discovery of a hierarchy underlying the data [187]. In comparison, few works have been concerned with learning from small datasets and/or with partial label in a framework similar to ours [226], [227]. For example, [226], [227] proposed an EM-like approach but dedicated to textual data and for a limited number of labels organized in a very flat hierarchy (of height equal to 2 or 3).

We will investigate and compare several methods from various families of approaches to solve our classification problems on biological data: a *SVM*-based approach (e.g. [197], [228]), a prototype-based approach originally inspired from [184], a *kNN*-based approach, logistic regression [229] and ensemble methods such as Random Forest [169] and Extreme Gradient Boosting Methods [230], [231]. We will consider three different settings: standard supervised multiclass classification, multiclass classification with partially labeled data, hierarchical classification with partially labeled data. While the methods are naturally available for the standard supervised multiclass classification setting, we will use variants to the two other settings, when available, or propose an extension otherwise.

4.2 Methods

4.2.1 Task formalization

The problem of automatically classifying partially labeled transcriptomic profiles is summarized in Fig 9. The task we are interested in may be viewed as a multi-class classification problem where samples $x \in \mathcal{X}$ belong to a set of labels $\mathcal{Y}$ that may be organized in a hierarchy. Here we consider $\mathcal{X} = \mathbb{R}^d$ and we note $|\mathcal{Y}| = C$. We explored several settings, that we briefly describe now.

Partially labeled setting

The *partially labeled* setting corresponds to the case where the training set includes samples which are labeled as well as samples whose labeling is incomplete (or partial). We will distinguish between the two kinds of samples as fully labeled samples and partially labeled samples. The supervision for a partially labeled sample, say x_i, is a label set $Y_i \subseteq \mathcal{Y}$ which is known to include the true label of x_i, and whose cardinal is limited (e.g. 2 to 10 in our experiments).

Generally, one wants to learn to classify input samples based on a training set D which consists of the union of two datasets; a fully supervised dataset D_s and a partially labeled dataset D_{pl}:

$$D = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, .., L\} \cup \{(x_i, Y_i) \in \mathcal{X} \times \mathcal{P}(\mathcal{Y}), i = L+1, .., L+M\} \equiv D_s \cup D_{pl} \quad (4.1)$$

where $\mathcal{P}(\mathcal{Y})$ represents the set of subsets of $\mathcal{Y}$.

Fully supervised hierarchical classification

Another important characteristic of the datasets we are interested in lies in the hierarchical organization of the labels (e.g. lineage hierarchy, as in [150]). It has been shown in the machine learning literature that taking into such prior knowledge may be beneficial in various hierarchical multiclass classification approaches [184], [197], [232]. Note that hierarchical classification often comes with a very large number of

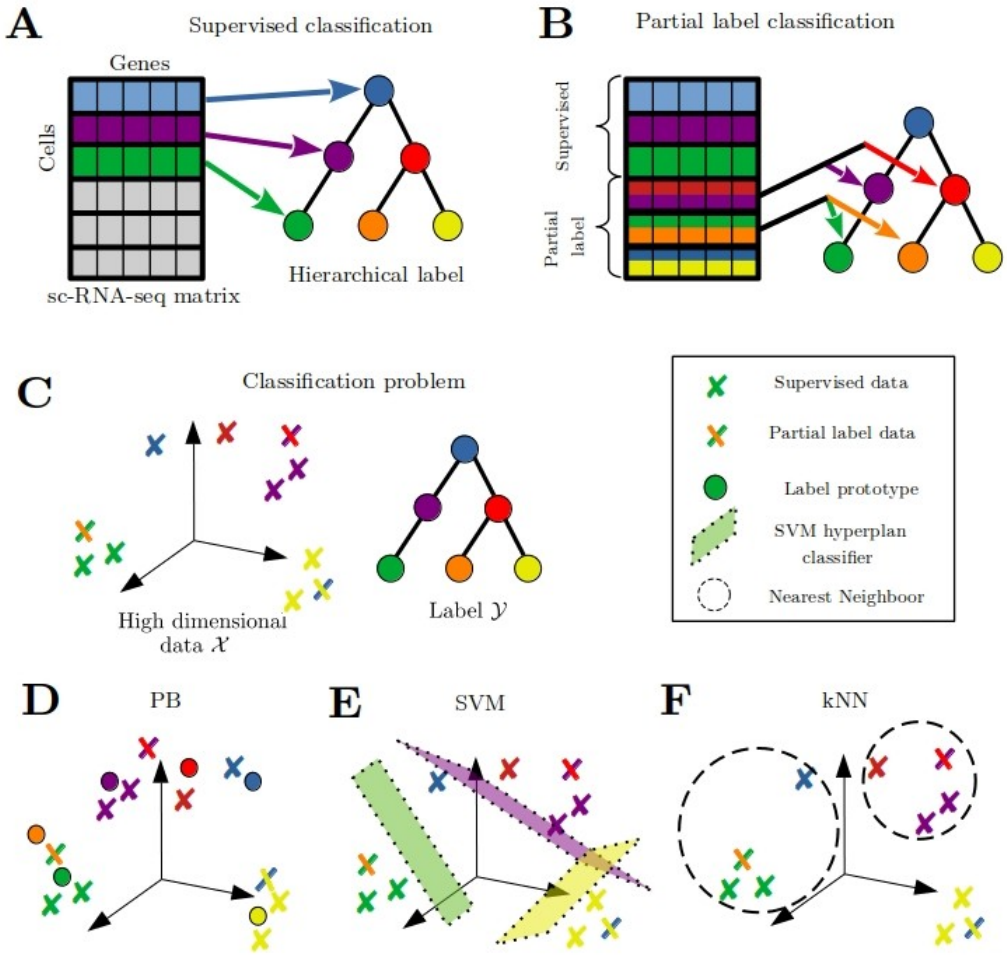

Figure 9 – **Partial label problem statement and classification methods**
A) Classification of transcriptomic profiles with a hierarchical label structure. Each training sample is associated to a unique label. **B)** Partial label learning framework, the transcriptomic profiles can be annotated with one or a *set* of candidate labels. **C)** In practice, the samples lie in a high-dimensional space, a proportion of the training samples is partially labeled, i.e. labeled with a set of candidate labels (illustrated with two colored crosses), while the remaining training samples have a unique label (and a unique color). The hierarchical structure of the labels is represented as a tree. **D-F)** Schematic illustration of some of the various methods explored in this study for the problem of partial label learning. **D)** The *PB* approach method consists in learning a projection of the data X and label Y into a new common embedding space. The embeddings of the labels are their associated prototypes, and the projection of the samples is learned such that the samples are as close as possible to their associated label's prototype. For prediction, the labels of test samples will be the one of the closest prototype in the common embedding space. **E)** The *SVM* approach consists in learning hyperplanes as classifiers in the initial feature space $\mathcal{X}$. The prediction for a new test sample will depend on the position of this sample with respect to the learned hyperplanes. **F)** The *kNN* algorithm predicts a new test sample's label by first identifying the nearest training samples in $\mathcal{X}$ and then using a majority vote to identify the most common label among them.

labels, this type of classification problems is named extreme classification [192]. As usually done in hierarchical classification, we will consider this prior information on labels through a matrix H, such that for two labels $\gamma, \eta \in \mathcal{Y}^2$, $H_{\gamma,\eta}$ stands for a measure of dissimilarity between the two labels (e.g. the shortest path in the hierarchy, which is a tree, between two nodes/labels).

Many approaches have been proposed for multiclass hierarchical classification in the *fully supervised* setting [184], [186], [197], which are extensions of traditional multiclass methods, mainly Support Vector Machines [197] and Prototype-based methods [184], [186]. The formalization of the hierarchical classification problem is often cast as the minimization of the average dissimilarity between the predicted and true labels

Partially labeled hierarchical classification

While it has been rarely explored to our knowledge in the machine learning community, the real datasets we are interested in include both aspects, partial labeling and a hierarchical organization of the labels.

4.2.2 Learning strategies

We investigated several families of methods: *Support Vector Machines based methods (SVM)*; *Logistic Regression (LR)*, *Nearest Neighbor methods (kNN)*, *Prototype-based methods (PB)* and *ensemble-based methods* (Random forest (*RF*) and Extreme Gradient Boosting Methods (*XGBM*)). We first explain in a generic way, independently of the classification model, how we handled the partially supervised setting. We then detail how we handled the hierarchical information to perform hierarchical classification. We note f_W a classification model (e.g. a *SVM*) parameterized by a set of parameters W, $f_W(x)$ the class output by model f_W for a sample x, $score(f_W, x, \gamma)$ the score for model f_W, class γ and sample x (in particular $f_W(x) = \arg\max_\gamma score(f_W, x, \gamma)$). Finally, we note $loss(f_W, x, y)$ the loss computed for a model f_W and a labeled training sample (x, y).

To start with, in the basic supervised multiclass classification problem, when the dataset includes supervised samples only, the solution is usually handled by an optimization problem of the form:

$$\arg\min_W \Omega(f_W) + \frac{\mu}{L} \sum_{i=1}^{L} loss(f_W, x_i, y_i) \tag{4.2}$$

where W are the parameters of model f to be learned, $\Omega(f_W)$ is a regularization term, and μ tunes the trade-off between the two terms of the objective function.

4.2.2.1 Partially labeled setting

The corresponding optimization problem consists of finding the optimal parameters W and labeling $\tilde{y}_i \in Y_i$ of samples $(x_i, Y_i) \in D_{pl}$ with respect to the following objective:

$$\arg\min_{W} \left[\Omega(f_W) + \frac{\mu}{L}\sum_{i=1}^{L} loss(f_W, x_i, y_i) + \frac{\lambda}{M}\sum_{i=1}^{M} \min_{\tilde{y}_{l+i}\in Y_i} loss(f_W, x_{l+i}, \tilde{y}_{l+i})\right] \tag{4.3}$$

where μ and λ tune the trade-off between the various terms of the objective.

Iterative Refinement Learning (*IRL*)

To deal with such an optimization problem, we used an EM-like iterative optimization algorithm (see Algorithm 1) which is inspired by proposals for learning *SVM* with latent variables (e.g. [225], [233]). We instantiate this algorithm here to fit our partially labeled setting. The algorithm consists, after initialization, of an iteration over two steps. The first one infers the labeling of partially labeled samples given the model's parameters, and the second one refines the model parameters using the inferred labels for partially labeled data, as in a fully supervised mode.

More precisely, in the first step, one infers a new guess $\tilde{y}_i^{(t)}$ for every sample (x_i, Y_i) in D_{pl}, using the current model $f_{W^{(t-1)}}$, where exponent (t) is used to name a variable at iteration number t. This is achieved by inferring the most likely label for each $x_i \in D_{pl}$ according to:

$$\tilde{y}_i^{(t)} = \arg\max_{\gamma\in Y_i} score(f_{W^{(t-1)}}, x_i, \gamma) \tag{4.4}$$

.

In the second step one builds a supervised version of D_{pl}, $D_{pl}^{(t)} = \{(x_i, \tilde{y}_i^{(t)}) \in \mathscr{X} \times \mathscr{Y}, i = L+1, .., L+M\}$. Finally, a new model $f_{W^{(t)}}$ is obtained by refining $f_{W^{(t-1)}}$ in fully supervised mode trained on $D_s \cup D_{pl}^{(t)}$.

The initialization may consist of random initialization of $\tilde{Y} = \{\tilde{y}_{l+i}, i = 1, .., M\}$ and a first learning of f in a supervised mode on D_s and on D_{pl} using this labeling or it might be a supervised learning on D_s only.

Iterative Full Retraining (*IFR*)

Note that the *IRL* algorithm (Algorithm 1) is not well suited for all classification models. It can only be used for models whose optimization can be started from the solution learnt at the previous iteration. This is restricted to models whose optimization is performed iteratively with, for example, gradient-based optimization routines. In this case, the refinement step can be implemented with a single step of gradient descent at every iteration. However, this is not adapted for ensemble-based models, nor for nonlinear (kernel) *SVM* when optimization is performed in the dual space. For such cases, we investigated a variant that we detail in Algorithm 2 close to the one in [225]. It relies on a complete learning of the model at each iteration. Because of this, it is computationally expensive.

Moreover, we observed experimentally that the alternative training schema in Algorithm 2 is quite sensitive to the expressivity of the classifier used. If the classifier is too expressive it could be able to learn any labeling of D_{pl} (for instance the first one $D_{pl}^{(1)}$)

Algorithme 1 : *Iterative Refinement Learning* (*IRL*) strategy for partially labeled setting.

Data : $D = D_s \cup D_{pl}$
Initialize : Initialize model $f^{(0)}$; t=1
while *Optimization not complete* **do**
  /* Infer labels for data in D_{pl} using current model (Eq. (4.4)) */
  $\forall i \in [\![L+1, L+M]\!], \tilde{y_i}^{(t)} = \arg\max_{\gamma \in Y_i} score(f_{W^{(t-1)}}, x_i, \gamma)$
  /* Build a labeled view of D_{pl} */
  $D_{pl}^{(t)} = \{(x_i, \tilde{y}_i^{(t)}) \in \mathcal{X} \times \mathcal{Y}, i = L+1, .., L+M\}$
  /* Build a new fully supervised dataset for learning the model */
  $D^{(t)} = D_s \cup D_{pl}^{(t)}$
  /* Refine model f in a fully supervised mode with $D^{(t)}$ */
  $f_{W^{(t)}}$ = One step gradient reestimation of $f_{W^{(t-1)}}$ using supervised dataset $D^{(t)}$
  t=t+1
end

and because this labeling would never change, the algorithm would be inefficient. Specifically, we observed that training very expressive classifiers with this algorithm often yielded poor performances, this suggests restricting the classifier's expressivity to achieve reasonable results. More elaborated optimization routines might be used to reach better results if the model has to be trained until convergence at every iteration.

Algorithme 2 : *Iterative Full Retraining* (*IFR*) learning strategy for the partially labeled setting.

Data : $D = D_s \cup D_{pl}$
Initialize : Initialize model $f^{(0)}$; t=1
while *Optimization not complete* **do**
  /* Infer labels for data in D_{pl} using current model (Eq. (4.4)) */
  $\forall i \in [\![L+1, L+M]\!], \tilde{y_i}^{(t)} = \arg\max_{\gamma \in Y_i} score(f_{W^{(t-1)}}, x_i, \gamma)$
  /* Build a labeled view of D_{pl} */
  $D_{pl}^{(t)} = \{(x_i, \tilde{y}_i^{(t)}) \in \mathcal{X} \times \mathcal{Y}, i = L+1, .., L+M\}$
  /* Build a new fully supervised dataset for learning the model */
  $D^{(t)} = D_s \cup D_{pl}^{(t)}$
  /* Learn model f from scratch in a fully supervised mode with $D^{(t)}$ */
  $f^{(t)}$ = Learn till convergence with the supervised dataset $D^{(t)}$
  t=t+1
end

Inference in the partially labeled setting

At inference time, partial labeling information may be available or not for a test sample,

which leads to two inference strategies, with (Eq. 4.5) or without (Eq. 4.6) such prior information:

$$\hat{y}_i = \underset{\gamma \in Y_i}{\arg\max}\, score(f_W, x_i, \gamma) \tag{4.5}$$

$$\hat{y}_i = \underset{\gamma \in \mathscr{Y}}{\arg\max}\, score(f_W, x_i, \gamma) \tag{4.6}$$

4.2.2.2 Hierarchical classification

There has been several extensions of standard multiclass classification methods to hierarchical classification, the most popular are *SVM*-based and prototype-based methods whose optimization formulation involves inequality constraints such as the one used in *SVM*-based models:

$$\forall \gamma \in \mathscr{Y} \setminus \{y_i\}, score(f_W, x_i, y_i) \geq score(f_W, x_i, \gamma) + 1 \tag{4.7}$$

where the constraint means that for a given training sample, the score of the right class should be bigger than the score of any alternative class, plus a margin (equal to 1).

The hierarchical extension of such a method consists of leveraging the prior information on labels through a matrix H, such that for two labels $\gamma, \eta \in \mathscr{Y}^2$, $H_{\gamma,\eta}$ stands for a measure of dissimilarity between the two labels. As has been done in several studies one may exploit this information to build hierarchical classifiers by using such a dissimilarity measure as a margin in constraints in Eq. (4.7) yielding what is known as a *margin scaling* strategy [228]. The above constraints are changed to:

$$\forall \gamma \in \mathscr{Y} \setminus \{y_i\}, score(f_W, x_i, y_i) \geq score(f_W, x_i, \gamma) + H_{y_i,\gamma} \tag{4.8}$$

Doing so, the score of a very distant class from the true one should be very low (because $H_{y_i,y}$ would be large) while the one of a very close class could be higher (but still below the score of the true label). It may be shown that by formulating the optimization problem with such inequalities in an *SVM* framework, one minimizes an upper bound of the average dissimilarity between the predicted and the true label [228].

Similarly, several works have studied the question of extending ensemble methods to hierarchical problems, however mainly in the context of multilabel classification [234], [235], which differs from our setting because a sample can belong to several classes. Due to the lack of off-the-shelf extension of ensemble methods for hierarchical classification, we will not include them in the experimental study for this particular context.

4.2.2.3 Partially labeled hierarchical classification

This last setting is a mix of the two previous settings and may be dealt with using a combination of the above strategies, for methods for which these strategies apply. The

combination of extensions for *SVM*, *PB* and *kNN* methods is described in detail below.

4.2.3 Models

We experimented with several models and strategies. We distinguish between two families of models. On the one hand, we investigated the use of models that do not require full training at every iteration but which may be iteratively refined along iterations with algorithm 1 - *IRL*), this includes all models that are trained with an iterative algorithm (e.g. gradient descent) such as prototype-based methods (taking inspiration from [184]), linear *SVM*, logistic regression, and *kNN* as a special case since it does not require training at all. Note that we also investigated the use of nonlinear *SVM* with an approximation of the kernel [179], [236] that enables a gradient-based learning routine as well. On the other hand, we considered models whose training at one iteration should be complete (with algorithm 2 - *IFR*) and does not benefit from the training at the previous iteration, such as Random Forests [169], Gradient boosting [231] and nonlinear (kernel) *SVM* [177]. The overall training strategy for these latter models is then much more costly than for the first set of models. All the methods and their implementation are summarized into Tab 4.1.

Methods	Acronym	Algo.	Implementation
Prototype Based with			
- linear projection	*PB -l*	*IRL*	Pytorch [237]
- nonlinear projection (neural network)	*PB-nn*	*IRL*	Pytorch [237]
Support Vector Machine			
- linear	*SVM*	*IRL*	Pytorch [237]
- nonlinear with kernel approximation	$\hat{k}$*-SVM*	*IRL*	Scikit-learn [238] , Pytorch [237]
Logistic Regression	*LR*	*IRL*	Pytorch [237]
k-Nearest Neighbors	*kNN*	N.A	Pytorch[237]
Random Forest	*RF*	*IFR*	Scikit-learn[238]
Extreme Gradient Boosting Method	*XGBM*	*IFR*	*XGBM*[231]
linear *SVM* with algorithm *IFR*	*SVM-IFR*	*IFR*	Pytorch[237]
kernel *SVM*	*k-SVC*	*IFR*	Scikit-learn[238]
LR with algorithm *IFR*	*LR-IFR*	*IFR*	Pytorch[237]

Table 4.1 – **Details of methods for partial label learning task**
Details on the acronyms, learning algorithm, and implementation. *IRL* stands for Iterative Refinement Learning algorithm, *IFR* Iterative Full Retraining algorithm, *N.A* means not applicable.

We focus now on the implementation details of the methods that are optimized

with the Iterative Refinement Learning (*IRL*), and for methods whose extension to hierarchical classification has been investigated. We also provide details on variants of these methods when applicable, in particular, nonlinear extensions of the *SVM*-based and the Prototype-based approaches. All along this section we build on existing methods and mention when we detail a method which was not proposed before. We do not provide any specific details for classification methods (Random Forest, Gradient Boosting and kernel *SVM*), whose extension to the *partially labeled* setting is implemented through the use of the Iterative Full Retraining (*IFR*), as this extension is straightforwardly implemented using their well known supervised learning strategy. Note that whenever it was possible, we used the state-of-the-art implementation for machine learning methods from scikit-learn [238].

4.2.3.1 Multiclass *SVM*

We first present the supervised case where we propose to use standard multiclass *SVM*. We then present variants dedicated to partially labeled data, its extension to hierarchical classification, and a nonlinear version of these. As stated above any mix of these ideas may be easily implemented (e.g. hierarchical classification with both supervised and partially labeled data).

Supervised setting

A multiclass linear *SVM* is defined through a weight matrix $W \in \mathbb{R}^{d\times C}$ which is composed of C column vectors $\{w_\gamma\}_{\gamma\in\mathcal{Y}} \in \mathbb{R}^d$ which stand for each label's classifier weights. We note f_W the *SVM* model with parameters W. One defines a score for a sample and a label γ as $score(f_W, x, \gamma) = \langle w_\gamma, x\rangle$. A sample x is classified according to $\arg\max_\gamma score(f_W, x, \gamma) = \arg\max_\gamma \langle w_\gamma, x\rangle$. Hence a sample x whose class is y is correctly classified if $\arg\max_\gamma \langle w_\gamma, x\rangle = y$.

The supervised learning of a multiclass *SVM* on a dataset D_s may be cast as the following optimization problem:

$$\min_{W,\xi} \frac{1}{2}||W||^2 + \frac{\mu}{L}\sum_i^L \xi_i \tag{4.9}$$

$$\text{such that } \forall i, \langle w_{y_i}, x_i\rangle - \max_{\gamma\neq y_i} \langle w_\gamma, x_i\rangle \geq 1 - \xi_i \tag{4.10}$$

$$\text{and } \forall i, \xi_i > 0 \tag{4.11}$$

where ξ_i are slack variables and μ is a hyperparameter that allows to tune the tradeoff between the two terms in the loss function. Note that there are as many constraints as the number of training samples. Each constraint encodes that the ith sample is correctly satisfied with a margin, with slack variables enabling some errors. This problem can be solved by gradient descent by minimizing its unconstrained form

which we put in the same shape as in Eq. (2) of the main manuscript:

$$\min_{W,\xi} \frac{1}{2}||W||^2 + \frac{\mu}{L}\sum_{i}^{L} loss(f_W, x_i, y_i) \tag{4.12}$$

$$= \min_{W,\xi} \frac{1}{2}||W||^2 + \frac{\mu}{L}\sum_{i}^{L} \max\left(0, 1 - [\langle w_{y_i}, x_i\rangle - \max_{\gamma \neq y_i} \langle w_\gamma, x_i\rangle]\right) \tag{4.13}$$

Partial labeling setting

The extension of *SVM* to the partial labeling setting has already been investigated, we take inspiration from the method in [224]. In the pure partial label learning setting, the training set is $D = D_{pl} = \{(x_i, Y_i), s.t. Y_i \subset \mathscr{Y}\}_{1 \leq i \leq M}$ and the learning is cast as the following minimization problem:

$$\min_{W,\xi} \frac{1}{2}||W||^2 + \frac{\lambda}{M}\sum_{i}^{M} \min_{\tilde{y}_i \in Y_i} loss(f_W, x_i, \tilde{y}_i) \tag{4.14}$$

$$= \min_{W,\xi} \frac{1}{2}||W||^2 + \frac{\lambda}{M}\sum_{i}^{M} \min_{\tilde{y}_i \in Y_i} \max\left(0, 1 - [\langle w_{\tilde{y}_i}, x_i\rangle - \max_{\gamma \notin Y_i} \langle w_\gamma, x_i\rangle]\right) \tag{4.15}$$

This formulation follows the form of Eq. (3) in the main manuscript, where $\tilde{y}_i \in Y_i$ stands for the guessed label for a partially labeled training sample $(x_i, Y_i) \in D_{pl}$ and $\lambda > 0$ tunes the tradeoff between the two terms of the objective function.

Both algorithms *(IRL and IFR)* may be used to optimize this objective where the optimization algorithm iterates inferring new guess for $\tilde{y}_i$ and re-estimate the model's parameters W (through refinement or full retraining).

Hierarchical variant

A classic way of exploiting the hierarchical organization of labels is to consider that it is a more severe error if a predicted label is very far in the label hierarchy from the true label than if it is closer. This may be integrated into an *SVM* formulation with the slack scaling strategy inspired by the structured output prediction field [228]. The related optimization problem becomes the following where the margin +1 is replaced by a quantity that is related to the distance between the labels, we use here $H_{y_i,\gamma}$. The more distant a label γ is from the true label y_i in the hierarchy, the larger the difference between the scores for these two labels should be. Note that in this case the objective function is an upper bound of the average distance from the predicted label to the true one whereas the loss function of standard multiclass *SVM* is an upper bound of the number of classification errors. Of course this hierarchical classification strategy may be combined with the partial labeling approach.

$$\min_{W,\xi} \frac{1}{2}||W||^2 + \frac{\mu}{L}\sum_{i}^{L} \max\left(0, H_{y_i,\gamma} - [\langle w_{y_i}, x_i\rangle - \max_{\gamma \neq y_i} \langle y_\gamma, x_i\rangle]\right) \tag{4.16}$$

Nonlinear extension

The linear *SVM*-like formalization above (that we name *SVM* in our experiments) yields a natural nonlinear extension with the kernel trick. We performed experiments with such a usual nonlinear implementation of *SVM*, that we call *k-SVC* in our experiments. It is trained with the *IFR* algorithm and, hence, does not scale well to large datasets. To enable efficient implementation and scaling to large datasets as well as to ease the extension to the hierarchical classification setting, we also considered using the *IRL* algorithm (Algortihm 1) and relied on a kernel approximation with a Radial Basis Function approach [239], where the score for a class y and for a sample x is implemented by $\langle w_y, \tilde{\psi}(x)\rangle$ with $\tilde{\psi}$ an appropriately chosen mapping function. It is a rather standard implementation, and we used the one provided by the scikit-learn package [238]. We name $\tilde{k}$-*SVM* this implementation of *SVM*.

4.2.3.2 Prototype-based approach

We first introduce a prototype-based approach that is inspired by works in extreme classification (i.e. when the number of classes is large) [184]. We consider first the supervised setting, then the partially labeled setting, and discuss variants, such as a nonlinear version and an extension incorporating the hierarchical structure of the label set.

Supervised setting

We build on the method in [184] as our starting point. It consists in projecting simultaneously data samples and one-hot representation of the labels in a common representation space $\mathcal{Y}$ where a prototype-based approach is used to classify data samples (i.e. a data sample is classified with the label whose projection is closest to the projection of the sample). The projection is learned to maximize a margin criterion. While the method has been designed for extreme classification, i.e. hierarchical classification when the number of labels is large (hundreds of thousands of labels or more), we exploit here a simplified version of the method. A projection operator $P \in \mathbb{R}^{d\times C}$ is learned to project data samples from $\mathcal{X}$ to $\mathbb{R}^C$, with C being the number of labels where a nearest neighbor classification rule is applied. The projections of the labels in $\mathbb{R}^C$ remain fixed and are set to either a one-hot-encoding or a multi-dimensional scaling projection (*MDS*) [240], the best projection is set by cross-validation. The final encoding of label is given by a matrix $\phi = \{\phi_\gamma\}_{\gamma\in\mathcal{Y}}$, composed of C columns vectors of dimension C.

To frame the approach similarly to our generic formalization, we note f_P the prototype-based mode, $f_P(x)$ the prediction for a sample x, and $score(f_P, x, y) = -||Px - \phi_y||_2^2$ the score for model f_P, input sample x and output label y. Note that the score is set minus the norm to be consistent with the maximization strategy of the score. The learning is cast as the following optimization problem:

$$\min_P \frac{1}{2}||P||^2 + \frac{\mu}{L}\sum_i^L loss(f_P, x_i, y_i) \tag{4.17}$$

$$= \min_P \frac{1}{2}||P||^2 + \frac{\mu}{L}\sum_i^L \max_{\gamma \neq y_i}\left(0, 1 - \left[||Px_i - \phi_{y_i}||_2^2 - ||Px_i - \phi_\gamma||_2^2\right]\right) \tag{4.18}$$

The above objective function may be optimized with both algorithms *IRL* and *IFR*, similarly to the *SVM* above. Note that to achieve better results we investigated optimizing a combined objective by adding a regression term to the objective function $L_{reg}(P) = \frac{\nu}{L}\sum_i^L ||Px_i - \phi_{y_i}||^2$ where $\nu > 0$ is a hyper-parameter to set with cross-validation.

Partially labeled setting and hierarchical variant

Like for *SVM*-based methods, one may extend the supervised method to the partially labeled setting by introducing latent variables. Noting the labeling $\tilde{y}_i \in Y_i$ of samples $(x_i, Y_i) \in D_{pl}$, the learning of the prediction model may then be cast as the following optimization problem, where one may add a regression term to the loss, in a similar way to the previous section:

$$\min_P \frac{1}{2}||P||^2 + \frac{\lambda}{M}\sum_i^M \min_{\tilde{y}_i \in Y_i} \max_{\gamma \notin Y_i}\left(0, 1 - \left[||Px_i - \phi_{\tilde{y}_i}||_2^2 - ||Px_i - \phi_\gamma||_2^2\right]\right) \tag{4.19}$$

Again we propose to learn the prototype-based model in the partial labeling setting by alternating the inference of latent variables $\tilde{y}_i$ based on the current model's parameters, and the re-estimation of the model's parameters through supervised learning.

Finally, the hierarchical variant may again be derived by introducing margin scaling replacing the 1 margin in Eq. (4.18) and (4.19) by distances $H_{y_i,\gamma}$ between two labels y_i and γ.

Nonlinear extension

In order to investigate nonlinear projection, one may replace $P \in \mathbb{R}^{d \times C}$, by a Neural Network. We looked at different architectures and found that using 2 hidden layers, whose number of neurons is set to the number of labels and with hyperbolic tangent activation function, was a well-adapted architecture to our problem. More complex models were often subject to overfitting because of the small amount of training data. In our experiments we name *PB-l* the Prototype-based method exploiting a linear projection operator and *PB-nn* the Prototype-based method with a nonlinear neural network projection operator

Euclidean vs Correlation distance

We explored different metrics and replaced the Euclidean distance with the correlation

distance, which has been shown to be relevant for transcriptomic data [241]. In this *Correlation distance* variant, all Euclidean distances in the above equations are replaced by:

$$d_{Correl}(x_1, x_2) = 1 - \frac{\sum_{g=1}^{p} (x_{1g} - \bar{x}_1)(x_{2g} - \bar{x}_2)}{\|x_1 - \bar{x}_1)\| \times \|x_2 - \bar{x}_2\|}$$

where $\bar{x}_1$ stands for the mean value among all the d features of x_1.

4.2.3.3 *kNN*

We build here on standard *kNN* [242] and on a variant of *kNN* for partially labeled data that has been proposed in [220]. We propose an extension to hierarchical classification.

Supervised setting

The classification of a test sample is performed in two steps: first, find the k_{nn} nearest neighbors of the sample to be classified in the training dataset, then select the most frequent label within these neighbors. Noting $top_{k_{nn}}(x)$ the set of the k_{nn} nearest neighbors of an example x in the training set D_s, the *kNN* classifier decision rule is defined as:

$$\hat{y} = \arg\max_{\gamma \in \mathscr{Y}} \sum_{x_j \in top_{k_{nn}}(x)} \mathbb{1}_{y_j=\gamma} \text{ where } \mathbb{1}_{y_j=\gamma} = \begin{cases} 1 \text{ if } y_j = \gamma \\ 0 \text{ else} \end{cases} \tag{4.20}$$

Partial labeling setting

We used a variant proposed in [220] to handle a training dataset $D = D_{pl} = \{x_i, Y_i, s.t. Y_i \in \mathscr{P}(\mathscr{Y})\}$. The decision rule is the following:

$$\hat{y} = \arg\max_{\gamma \in \mathscr{Y}} \sum_{x_j \in top_{k_{nn}}(x)} \sum_{\eta \in Y_j} \mathbb{1}_{\eta=\gamma} \tag{4.21}$$

It is a straightforward extension of *kNN* to this setting where all labels in the labeling sets of the k_{nn} nearest neighbors count equally, with a majority vote at the end to determine the class of an incoming sample.

Hierarchical variant

We designed a hierarchical variant of *kNN* by relying again on the matrix H, which may be applied in the fully supervised setting and in the partial label setting. The idea is that all the labels in all label sets of the nearest neighbors contribute but with a weight that depends on the distance in the hierarchy, we relied on the weights encoded

in the matrix H and used the following decision rule :

$$\hat{y} = \arg\max_{\gamma \in \mathscr{Y}} \sum_{x_j \in top_{k_{nn}}(x)} \sum_{\eta \in Y_j} \frac{1}{2}^{H_{\gamma,\eta}} \tag{4.22}$$

where $H_{\gamma,y}$ is the dissimilarity between the labels γ and y.

4.3 Datasets and experimental settings

4.3.1 Datasets

The different datasets on which we performed experiments to compare the various methods in different settings are presented in detail in Chapter 3. We used simulated datasets using the prosstt library [209] and single-cell RNA Seq reference datasets [150], [210], [211].

4.3.2 Experiments

4.3.2.1 Partially labeled datasets

To investigate the performance of our approaches in various partial label settings we generated a collection of partially labeled datasets on which to test the various methods. From the fully supervised datasets described above, we generated a collection of partially labeled datasets $D = D_s \cup D_{pl}$ corresponding to different settings in terms of the proportion of labels for which we have fully supervised training samples, amount of partial labels, overlap between labels of supervised samples and of partially labeled samples, and balance between supervised samples and partially labeled samples.

To characterize the various experimental settings, we first introduce a set of characteristics for the datasets below. In the following, we note $\mathscr{Y}_s$ the set of labels that occur in D_s and $\mathscr{Y}_{pl}$ the set of labels that occur in D_{pl}.

1. **Percentage of shared labels between D_s and D_{pl}** We generated datasets with different distributions of labels in D_s, in D_{pl} and in their intersection.
   — o is the percentage of unique labels that belong to the intersection $\mathscr{Y}_s \cap \mathscr{Y}_{pl}$ with respect to the number of unique labels in $\mathscr{Y}_s \cup \mathscr{Y}_{pl}$. It is defined as $o = \frac{|\mathscr{Y}_s \cap \mathscr{Y}_{pl}|}{|\mathscr{Y}_s \cup \mathscr{Y}_{pl}|}$.
2. **Number of supervised samples**. We define two parameters to control the amount of supervision when training the models.
   — n is the number of fully supervised samples per label $\gamma \in \mathscr{Y}_s$. This parameter is used only for simulated data.
   — p is the percentage of fully supervised samples used for training data for each label $\gamma \in \mathscr{Y}_s$. This parameter is used for the experimental datasets because of the non uniformity of sample distribution over the set of labels.

3. **Characteristics of the partial labels**. Choosing the way we build partial label Y_i subsets has a clear impact on the model training. We used two parameters:
   — k is the number of possible labels for a given partially labeled sample, i.e. for a training sample $x_i \in D_{pl}$, it is the cardinal of the subset Y_i, $|Y_i|$. We keep it constant for all data in D_{pl}.
   — I is an interval of minimum and maximum distances, to the true label y_i of a sample x_i, within which labels will be randomly chosen to belong to a subset Y_i.

4.3.2.2 Scenarios

We considered a number of experiments in specifically designed experimental settings which are characterized by the set of parameters $s = (o, p, n, k, I)$ discussed above. These numerical experiments are organized around two main scenarios which are illustrated in Figs 11 and 12, panel B.

1. Complete overlap between the labels of supervised and of partially labeled samples (i.e. $o = 1$) (Fig 11B). In that scenario we investigated various experimental settings by varying n or p, k and I.
2. No overlap between the labels of supervised and partially labeled samples (i.e. $o = 0$) (Fig 12B). In that scenario, we investigated various experimental settings by varying n or p, k and I.

4.3.2.3 Tasks

For each of the scenarios and the experimental settings we considered two tasks that allow investigating in deep the behaviors of our methods.

1. *Fully supervised setting*. The model is trained on fully supervised data only.
2. *Partially labeled setting*. The model is trained on both fully supervised and partially labeled data.

4.3.2.4 Metrics

We report experimental results according to two metrics of interest that we detail here. We note n_{tot} the number of samples which are used for computing the criterion. The first metric is computed based on label prediction without prior information using Eq. 4.6 while the second metric is computed based on label prediction with prior information using Eq. 4.5.

— *Precision*: the precision is the percentage of correct classification, it is defined on a set of test data whose true label is known as $\frac{1}{n_{tot}} \sum_i \mathbb{1}_{\hat{y}_i, y_i}$. Note that the precision coincides with the micro-f1 score in the multi-class, mono-label classification setting.
— *Precision with prior*: this criterion is defined for partially labeled samples only (x_i, Y_i). It is defined as the precision when the prediction is performed within the label set Y_i.

Generating partial label data and Train Test split

Data generation follows a multi-step process:

1. Initially, the data is split based on the label and whether there is label overlap. If $o = 0$, the data is divided evenly between two sets. In contrast, if $o = 1$, all labels are included in both sets. A test set is created for each label group, with a proportion of 20% of the samples allocated for testing.
2. The supervised training set is further subdivided based on the proportion (p) value, which can take on values of 0.1 or 1.0, or the number (n) of training examples, set to either 20, 200, or 1000. Stratified train-test splits are used for this process. For the partial label training set, a fixed number of 200 examples is chosen in the case of the simulated datasets (Prosstt *Branches*), and 80% percent of the data points in the training set is selected in the case of the experimental datasets.
3. The partial labeling is generated as follows:
   — For each label, we create a set of pairwise probable labels to increase the co-occurrence, with the number of pairs set arbitrarily to 15 within the specified interval I. In the simulated dataset presented in Figs 11 and 12, we selected a close distance interval (1, 15) and a far interval (15, 253). However, for the experimental datasets, we opted for only close intervals, directly determined by the number of labels. These details and information can be found in Figs 13, 14 and 15.
   — For each partially labeled sample, the partial labeling is sampled uniformly from the associated label co-occurrence set. It contains $k = 2$, 4, or 10. This process is also performed for the partial label example test set, allowing the evaluation of prediction with prior information metric.

Multiple runs, gridsearch and averaged results

We report averaged results in the following. We explain how we get these now.

For every experimental setting, as defined by a set of values $set = (o, p, n, k, I)$, we first generate one dataset with the desired characteristics. Then we perform 5 cuts of the dataset into train and test X_s^{train} and X_{pl}^{train} on one side and X_s^{test} and X_{pl}^{test} on the other side (where s stands for supervised, and pl for partially labeled).

For every train-test cut, we use the train data to learn models and select hyperparameters since the learning of each model involves hyper-parameters. As is usually done in machine learning the optimal hyperparameter values are selected using cross validation on the training set (we used 5 fold cross validation), then the selected hyperparameters are used to learn a model on the full training set, which is then evaluated on the test set.

At the end, for each experimental setting, we get 5 performance values, one for each train-test cut, that are averaged to get one averaged performance value that we report on the figures.

Statistical significance

For each experimental setting, given a row in a figure, we evaluated which one among all the methods had the best performance. We perform t-test between the first and second best methods. When several methods are available in a same family (as *SVM*, *PB* or *kNN*), we select the best version in the family (linear vs nonlinear in Figs 10,11,12,13,14,15, or hierarchical vs flat in Fig 16). For example, if, for a given experimental setting, *PB-nn* was the best between *PB* methods and linear *SVM*was the best between *SVM*methods, we performed a paired t-test between those two. If one of the two methods performed significantly better than the other, with a p-value < 0.1, we reported it on the figure by drawing red square with a dotted line around the best performance value. We chose to set the significance threshold at 0.1 for the p-value because the data are noisy and there are only 5 performance values per experimental setting and per method.

4.4 Results

First, we analyze the fully supervised setting on 12 artificial Prosstt datasets, as shown in Fig 10. We then focus on one specific simulated dataset to explore different settings of the partial label learning problem. The different settings include scenarios with label overlap (Fig 11), and scenarios without label overlap (Fig 12). Furthermore, we analyze the performance of our approaches on three real-world datasets, *C. elegans* [150], *S. mediterranea* [210] and *Myeloid progenitors* [211] in Figs 13, 14 and 15 respectively. Finally, we compare the performance of hierarchical and flat methods in Fig 16 where we focus on partial labeling cases with different settings on two real-world datasets. For the artificial datasets which contain about 300,000 samples each, we only compare scalable methods, the ones exploiting the *IRL* algorithm. When considering the real datasets, which are much smaller, we provide a comprehensive comparison of all methods.

4.4.1 Supervised setting

Fig 10 displays the results for all methods on 12 different datasets.

The dataset topology influences classification accuracy

For each topology, we order datasets by decreasing difficulty: few genetic programs (g=10) with noise (α=0.5), without noise (α=0.1), then more genetic programs (g=50) with noise (α=0.5) and without noise (α=0.1). In Fig 10, for the dataset corresponding to 50 programs, 0.1 noise and 20 examples per label in the training set, the method Prototype-Based neural network (*PB-nn*) achieves 0.59 accuracy for *Linear* topology, 0.76 for *Branches* and 0.71 for *Binary* For the Support Vector Machine (*SVM*), the results are respectively 0.53, 0.61 and 0.56.

***PB-nn* and *SVM* methods are well adapted for classification**

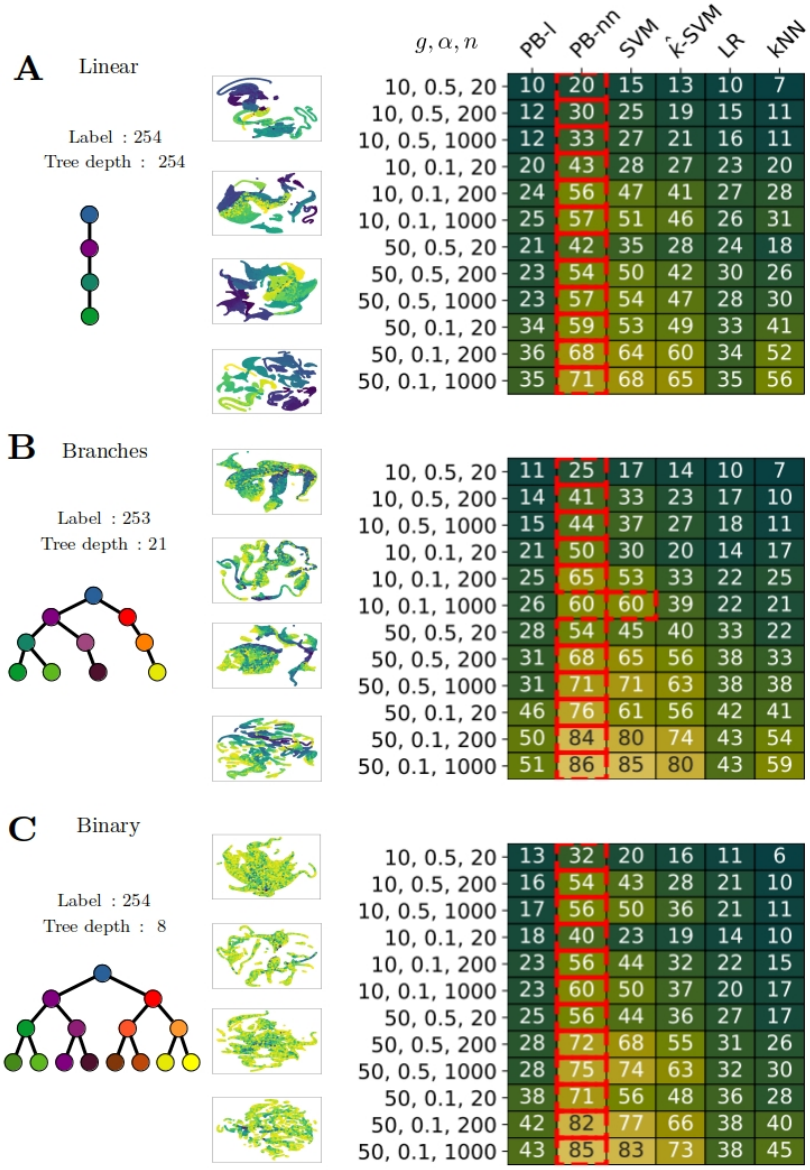

Figure 10 – **Supervised performance on simulated data**
A-B-C) The first column represents the three main topologies studied with simulated scRNA Seq data. The second column shows the *UMAP* projections of the various datasets [213], the color is plotted according to pseudotime, which is computed during the data simulation. Heatmap of performance according to the method and dataset and number of samples. Each row represents a dataset which is named by order, according to the number of programs (g), the coefficient (α) used to control the noise in the simulations, and the number of samples (n) for each label in the training data. Each column shows a different method (as presented in the Material and Methods section). Each square corresponds to the percentage accuracy between the true label and the prediction using the given method. The red squares indicate the best-performing method per row, i.e. per experimental setting, using a t-test and as significance criteria p-value < 0.1.

The general behavior on all datasets is that Prototype-Based with Neural Network (*PB-nn*) performs significantly better (p value < 0.1) than linear *SVM*, which is better than linear Prototype-Based (*PB-l*), Logistic Regression (*LR*) and *kNN*, which are similar. In details, we take *kNN* as a baseline, we first observe that *PB-l* performs slightly better in most of datasets especially with few training examples (20 samples per label), in particular for *Binary*, 10 programs, 0.1 noise, 20 training examples per label; the accuracy is 0.18 (*PB-l*) versus 0.10 (*kNN*). However, with 50 genetic programs, less noise and more examples, *PB-l* tends to be less efficient. The contrast becomes more prominent when comparing *PB-nn* and *SVM* to *kNN*, with accuracy potentially doubling or even tripling depending on the settings. As an example, if we take a difficult dataset like *Binary*, 10 programs, little amount of noise (α=0.1) and a low number of training examples per label (n=20), the accuracies are 0.40 (*PB-nn*), 0.23 (*SVM*) and 0.10 (*kNN*). Finally, when the dataset becomes easier, the accuracy becomes very high: 0.86 (*PB-nn*), 0.85 (*SVM*) and 0.59 (*kNN*), when looking at dataset *Branches*, 50 programs, a small amount of noise (α=0.1) and a lot of training examples per label (n=1000). We explain the contrast between *PB-nn* and *PB-l* by the intrinsic complexity of scRNA seq data, which requires nonlinear projection to achieve good results. We also observed that kernel-*SVM* performs worse than the regular *SVM* and hypothesize that this decreased accuracy comes from the kernel approximation. More discussion can be found in section 4.4.6.

High accuracy with few training samples
We are interested in assessing the influence of the number of training samples per label. Because of the lack and limitation of supervised data in real-world settings, we want good performance even if only a small training dataset is available. The general value varies between 10 and 200 examples per label for several real-world scRNA Seq datasets. We observe that *SVM* works well with 200 examples per label, *PB-nn* works well with 20 examples and the performances with 200 examples are close to *SVM* with 1000 examples. We compare *Branches*, 10 programs, 0.5 noise: with 200 training examples per label, *PB-nn* accuracy is 0.41, whereas the accuracy for *SVM* is 0.33 and with 1000 training examples, *SVM*'s accuracy is 0.37. Similarly for *Linear*, 10 programs, 0.5 noise, 200 training examples per label, *PB-nn* achieves 0.3 and *SVM* 0.25, while the accuracy of *SVM* increases to 0.27 with 1000 training examples.

To summarize, our supervised approaches are relevant for simulated transcriptomic data and we find that both *SVM* and *PB-nn* perform very well on all datasets. Moreover, the *PB-nn* significantly outperforms *SVM, LR* and *kNN* in 35 experimental settings out of 36, making it the method of choice for this task. Finally, we highlight that *PB-nn* has high predictive power even with a small amount of training data.

4.4.2 Partial label, Overlap = 1

In this paragraph, we study the partial label problem with label overlap. It means that for a given label, the training set contains both fully supervised samples and

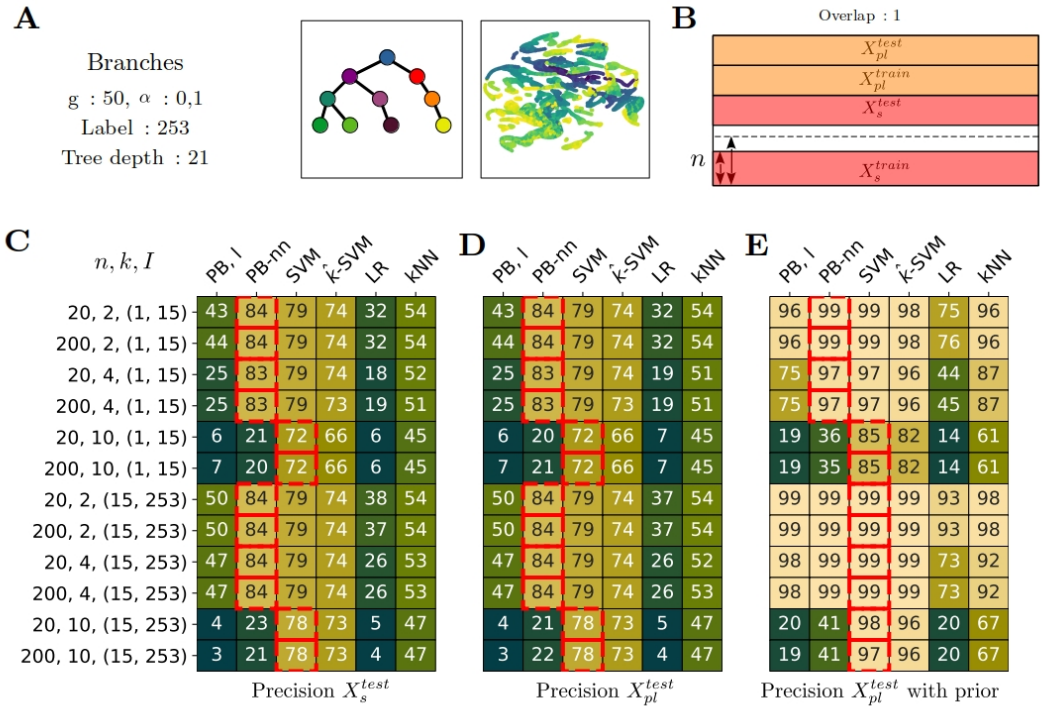

Figure 11 – **Partial label results for the setting with label overlap (o=1)**
A) Dataset *Branches*, 50 genetic programs, low level of noise. *UMAP* of the transcriptomic profiles, colored according to the pseudotime. **B)** Partial label scenario. Full overlap between supervised label and partial ones. For supervised subset, we train on $n = 20$ or $n = 200$ examples per label and 200 examples for each partial label. **C,D,E)** Precision according to the setting. Each row corresponds to a setting depending on, by order, I: the interval for partial label sampling, k: the number of partial label set, n: the number of training examples per label supervised. The ith row of C corresponds to the ith row of D and E. **C)** Precision on X_s^{test}. **D)** Precision on X_{pl}^{test}. **E)** Precision on X_{pl}^{test} when prior information is available for test too. **C-E)** The red squares indicate the best-performing method per row, i.e. per experimental setting, using a t-test and as significance criteria p-value < 0.1.

partially labeled ones. We focus the study on one dataset: *Branches*, 50 programs, a small amount of noise (α=0.1) as shown in Fig 11.

Performances are close to fully supervised setting

On both X_s and X_{pl}, the methods perform very well. Particularly, we found that with 20 supervised plus 200 partially labeled examples, with a set of candidate labels of size k = 2 or 4, the methods *PB-nn* and *SVM* have accuracy of 0.84 and 0.79 versus 0.84 and 0.80 in the fully supervised cases (with 200 fully supervised training examples per label). The performance on the partial label test set is identical to the supervised test set, contrary to the case of no label overlap as shown below. Note that for both

PB-l and *LR*, performances were below other methods on this dataset on the fully supervised case (in Fig 10, accuracies were 50 and 43 respectively with $n = 200$ samples).

With prior information, the results are excellent
When partial label information is also available for the test set, (precision with prior), *PB-nn* and *SVM* have 0.99 accuracy when k=2, 0.97 with k=4. It means that, the method can be used to disambiguate new data with a small amount of information.

***SVM* works very well with difficult partially labeled settings**
When the learning task becomes extreme, for example, 20 supervised examples and 200 partial label ones with a large number of candidate labels (k=10), the *SVM* still performs well: 0.72 and with prior information accuracy goes to 0.85. Whereas the accuracy of *PB-nn* falls with a value of 0.2 and 0.36 with prior information.

The distance between labels influences the partial learning performance
When the candidate labels are far from the real label in the partial labeling information ($I = (15, 253)$), it leads to better results. Especially, with prior information, looking at *PB-nn*, for k=4, and 20 supervised examples, the accuracy goes from 0.97 to 0.99. The parameter I plays a role in partial labeling, as it can impact the results. For example, when selecting two labels situated at the same depth in a lineage tree, they will have identical pseudotime values (distance to the root), but their hierarchical distance might be large.

To summarize, the methods perform very well with partially labeled data. The performances are comparable to the fully supervised setting. We show that just a few amount of supervised data is needed to reach top performance and we provide some criteria to improve the performance according to the characteristics of the candidate label set (cardinal and distances).

4.4.3 Partial label, Overlap = 0

This time, we study the scenario when there is no overlap between supervised label and partial label. This task is much more difficult than the previous one. We still focus on the same dataset: *Branches*, 50 programs, a small amount of noise (α=0.1) as shown in Fig 12.

The precision on supervised data depends on the size of the training dataset
We observe that the accuracy on the supervised test set is sensitive to the quantity of fully supervised examples. Indeed, when there is only n=20 supervised examples per label in the training set, the accuracy is at 0.32 for *PB-nn* (k=2, I=(1,15)), but it increases to 0.84 with n=200 training examples. This is due to the fact that the model will focus more on the partially labeled training data when there is less supervised data. We can also observe that the corresponding accuracy for the partial label test set decreases from 0.9 to 0.84 for the same configuration. Depending on the task, the methods require a compromise between supervised and partial label training data. For example,

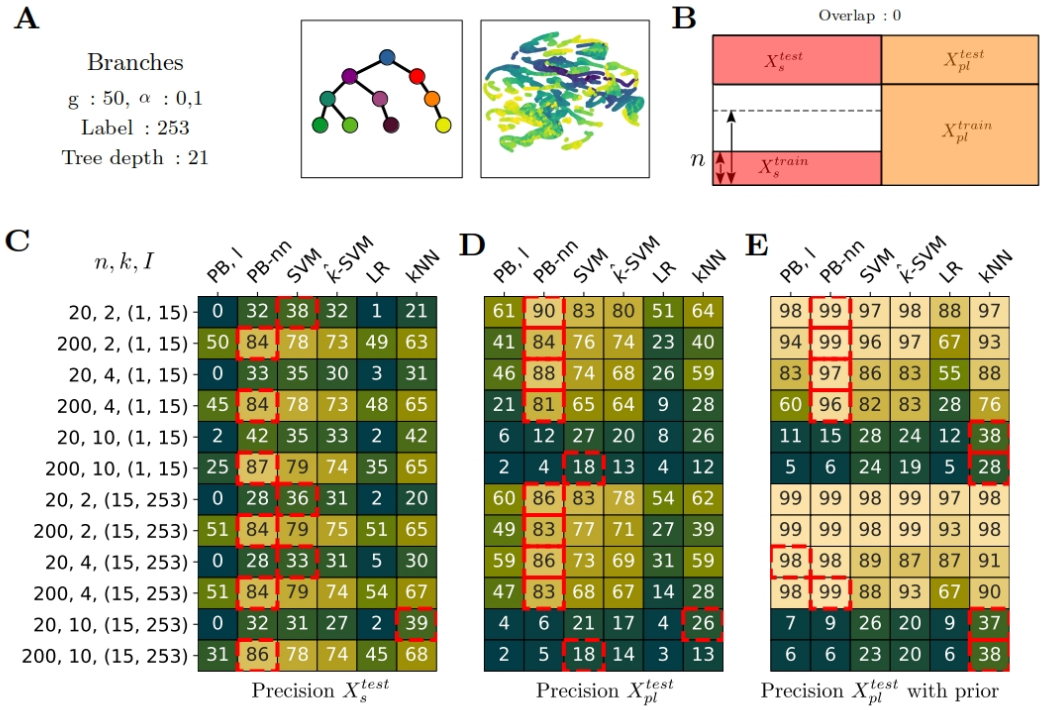

Figure 12 – **Partial label results for the setting with no label overlap (o=0)** **A)** Dataset *Branches*, 50 genetic programs, low level of noise. *UMAP* of the transcriptomic profiles, colored according to the pseudotime. **B)** Partial label scenario. No overlap between supervised label and partial ones. For supervised subset, we train on $n = 20$ or $n = 200$ examples per label and for the partial label subset, we select 200 examples for each partial label. **C,D,E)** Precision according to the setting. Each row corresponds to a setting depending on, I: interval for partial label sampling, k: number of partial label set, n: number of training examples per fully supervised labels. The ith row of C corresponds to the ith row of D and E. **C)** Precision on X_s^{test}. **D)** Precision on X_{pl}^{test}. **E)** Precision on X_{pl}^{test} when prior information is available for test too. **C-E)** The red squares indicate the best-performing method per row, i.e. per experimental setting, using a t-test and as significance criteria p-value < 0.1.

it could be relevant to increase the number of supervised examples for supervised prediction whereas this could decrease the quality of prediction on partially labeled data.

Ability to predict partial labels only

This time, we are characterizing the ability to predict the good label for partially labeled data when there is no supervised data at all. It means that the model can learn to predict good labels without having seen it in the supervised training data. When the partial label set is not too large (up to k=4), the best performances are obtained with *PB-nn*: (for n=20, k=2 and I=(1,15)) the accuracy reaches 0.9 and (for k=4), it goes to

0.88 while the corresponding *SVM* performances are 0.83 and 0.74. When the setting becomes extreme, (k=10), the performances fall but are still better than random. In this situation, *SVM* stays reliable.

Results are excellent with prior information
As before, models can predict the good label even without supervised training example. When the size of partial labels is not too large, *PB-nn* reaches 0.99 (k=2) and 0.97 (k=4), while *SVM* obtains 0.97 (k=2) and 0.82 (k=4). As in the previous paragraph, when the distance between labels increases ($I = (15,253)$ in the partial labeling accuracy goes from 0.96 to 0.99 for *PB-nn* (k=4 and n=200)).

Overall, we demonstrate that the models trained on partially labeled data only can predict the right label with high accuracy even without supervised training data on this label. With prior information, the models have performances close to perfect accuracy. We suggest using *PB-nn* methods while the size of the set of candidate labels is small (k=4) performances are significantly better (p-value < 0.1). For more difficult situations, *SVM* is significantly better (p value < 0.1) and achieves very good results. Overall we find, for simulated data in the partially labeled case, that, out of 72 experimental settings, *PB-nn* performs significantly better than all the other methods in 40 cases (i.e. 55 % of the time), while *SVM* outperforms all the other methods in 26 cases (i.e. 36 %) of the cases.

4.4.4 Real-world datasets

In the following paragraphs, we apply an identical pipeline to each of the three experimental datasets. Because of the relatively smaller size of the experimental datasets compared to the artificial ones, we now include, in addition to the methods studied in the previous section, alternative methods such as ensemble-based approaches that can only be optimized with the costly Algo 2. All related results are presented in Figs 13, 14 and 15. Our analysis begins by examining the classification problem in a supervised setting (panel B in the figures). Afterwards, we explore the two scenarios of partial labeling, one with label overlap and the other without label overlap (panel C in the figures). First, we provide an analysis of the common results observed across the datasets, we then delve into the specific characteristics and nuances of each dataset.

Common results
In the supervised setting, we observed that *PB-nn* performs the best for the *S. mediterranea* and *C. elegans* dataset. For *Myeloid progenitors* dataset, Random Forest (*RF*) achieves the best results. We note that all methods perform well even when they have only half of the training examples ($p = 0.5$). In total, we found that *PB-nn* significantly outperforms all the other methods in 3 cases out of the 6 settings and *RF* 2 cases out of the 6.

For the partial label learning setting, specifically when there is label overlap ($o = 1$),

the precision on the supervised test set is nearly as high as the performance achieved in the fully supervised setting. Moreover, in the partial label test set, we noticed even higher performances, even when using $k = 10$. However, it is important to have a sufficient amount of supervised training data ($p = 1$) to achieve such results.
On the other hand, in scenarios without label overlap ($o = 0$), we observed a behavior similar to the one observed on simulated data, with a compromise between supervised and partial label performances. As the amount of supervised training data increases ($p = 1.0$), the performances on the supervised set increase, while the performances on the partial label set decrease. In certain cases, results are excellent, particularly when $k = 2$ and $k = 4$. Sometimes, the performance in this setting even surpasses that of the fully supervised setting, which can be attributed to the smaller size of the test set.

When examining the performance with prior information available, we observe that the Prototype-Based (*PB*) methods stand out for $k = 2$, across all the datasets. As the value of k increases to 4 or 10, the *SVM* methods remain reliable and maintain competitive performance.

Overall, in the partial label setting, we find that the *PB* family of methods significantly outperforms all the other methods 22 times out of 108 experimental settings (20 % of the time). On the other hand, the *SVM* family of methods significantly outperforms the other methods 22 times among the 108 experimental settings (20 % of the time), *RF* outperforms in 14 cases, (13 % of the time), *kNN* methods are significantly better in 11 cases out the 108 experimental settings (10 % of the time). Finally, *LR* and *XGBM* outperform both in one case.
Both of the ensemble methods (*RF* and *XGBM*) rely on Algo 2. We observe that in general these methods tend to be more efficient in simpler scenarios. Specifically, the performances in fully supervised learning tasks (panel B of the figures) are competitive in all three real-world datasets. In partial label learning tasks, these methods tend to focus more on supervised data (especially with no overlap ($o = 0$)) and achieve good results on the corresponding test set, but performances decrease on the partial label data. The main limitations may come from the algorithm itself; particularly, if the model is too expressive, it might learn the first random inference of labels, and if the model is too simple, performance will be limited. More discussion on the Algo 2 can be found in supplementary results 4.4.6.

Generally, in both partial label learning scenarios (with and without overlap), we consistently observed excellent results, particularly when the proportion of supervised data is low (p=0.1). These findings once again underscore the value and justification for utilizing partially labeled data for classification. The remarkable performance achieved with a limited amount of supervised information further validates the effectiveness and practicality of employing partial label learning methods in this context.

C. elegans
This dataset follows a *Binary* topology with a depth of approximately 10/11, com-

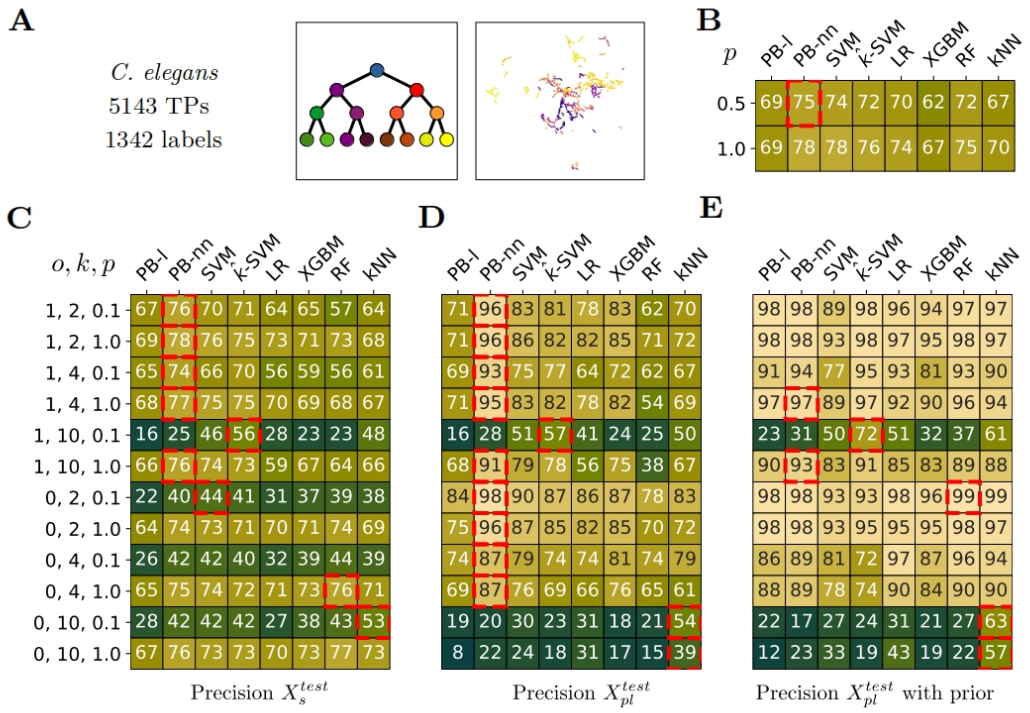

Figure 13 – **Classification performance on the *C. elegans* dataset**
A) *C. elegans* dataset main features. The topology of the label corresponds to the *Binary* one. We use the entire lineage tree [6] for label classification. The *UMAP* projection of the annotated transcriptomic profiles, colored according to their respective label. **B)** Performance of our methods in a fully supervised setting according to the proportion p of supervised training data. 0.5 means that models are trained with half of the training data. **C,D,E)** Precision according to the setting in partial label scenarios. Each row corresponds to a setting depending on, by order, o: Overlap between supervised and partial label, k: number of partial label set, p: proportion of fully supervised training data. The ith row of C) corresponds to the ith row of D) and E). The median value of pairwise label distance is 16 and we chose $I = (1,8)$ for partial label sampling. **C)** Precision on X_s^{test}. **D)** Precision on X_{pl}^{test}. **E)** Precision on X_{pl}^{test} when prior information is available for test too. **B-E)** The red squares indicate the best-performing method per row, i.e. per experimental setting, using a t-test and as significance criteria p-value < 0.1.

prising about 1342 labels. The dataset we are working with covers 95 labels scattered throughout the lineage tree.

In the fully supervised setting, both the *PB-nn* and *SVM* methods exhibit the best performance, achieving an accuracy of 0.78 when all the training data are available ($p = 1.0$).

When dealing with partial label data and label overlap ($o = 1$), we observe excellent results for both the supervised and partial label test sets. Even with a higher level of uncertainty ($k = 10$), all methods prove effective as long as there is sufficient supervised data available ($p = 1.0$).

In the more challenging scenario without label overlap ($o = 0$), the performances remain excellent until $k = 4$ and remain high for $k = 10$, even for the *PB* method, which has shown to be slightly less efficient in this situation.

Additionally, we find that kernel *SVM* is particularly relevant for inference with prior information, especially in cases of label overlap ($o = 1$), where it outperforms the regular *SVM* method. The kernel approximation seems to enhance the predictive capabilities of *SVM*, especially in situations with more complex hierarchical relationships.

S. mediterranea

This dataset exhibits a relatively flat hierarchy, and its topology resembles that of the *Branches* dataset, comprising several concise linear differentiation processes. The intrinsic complexity of the dataset appears to be effectively captured by both the *PB-nn* and *SVM* methods.

In the supervised setting, we find that the performances of all the methods are quite close (*PB-nn* stays significantly better). Both *PB-nn* and *SVM* obtain great results across most settings. Notably, in the label overlap scenario ($o = 1$) with $k = 10$ and ample supervised training data ($p = 1.0$), *PB-nn* achieves an accuracy of 0.84.
Additionally, we also observe that kernel-*SVM* is efficient in handling two of the most challenging settings, ($k = 10$, $o = 1.0$), with either a limited or a complete amount of supervised data ($p = 0.1$ and $p = 1.0$). It shows superior performance on both the supervised and partial label test sets.

In the case of no overlap ($o = 0$), the common behavior, observed across all the methods is to focus on supervised data. It is even more remarkable for *RF*. Notably, when all the supervised training data are available ($p = 1.0$), performances of *RF* outperform all the other methods, for $k = 2, 4, 10$ with an accuracy of 92, 94, 94 respectively. The performances on the partial label data are however quite below.

Myeloid progenitors

The hierarchy is much more present in this dataset. It consists of three main linear differentiation processes. There is a small number of labels (26) and a low amount of training data. We notice that *PB-nn* performs less than *PB-l* due to overfitting. Indeed, in the supervised setting, *PB-nn* demonstrates better performance when all the training data is available ($p = 1$) compared to using only half of it ($p = 0.5$).
In partial label learning task, *PB* methods and *SVM* outperform *kNN* significantly except when there is not enough data (p=0.1) or if it's too difficult (k=10). For the

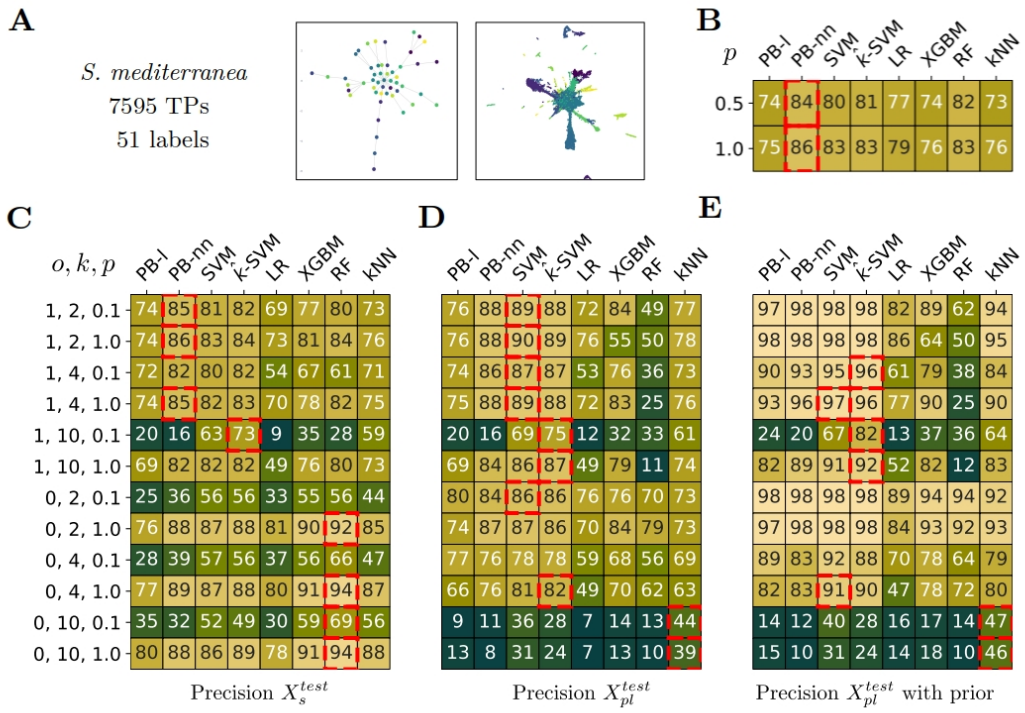

Figure 14 – **Classification performance on *S. mediterranea* dataset**
A) *S. mediterranea* dataset main features. The topology of the label is close to *Branches* case. The *UMAP* of the annotated transcriptomic profiles, colored according to their respective labels. **B)** Performance of the methods in a fully supervised setting according to the proportion (p) of supervised training data. 0.5 means that the models are trained with half of the training data. **C,D,E)** Precision according to the setting in partial label scenarios. Each row corresponds to a setting depending on, by order, o: overlap between supervised and partial label, k: number of partial label set, p: proportion of fully supervised training data. The ith row of C) corresponds to the ith row of D) and E). The median value of pairwise label distance is 3 and we chose $I = (1,4)$ for partial label sampling. **C)** Precision on X_s^{test}. **D)** Precision on X_{pl}^{test}. **E)** Precision on X_{pl}^{test} when prior information is available for test too. **B-E)** The red squares indicate the best-performing method per row, i.e. per experimental setting, using a t-test and as significance criteria p-value < 0.1.

performance on the partial label, *PB-nn* stands out until $k = 4$, especially with label overlap ($o = 1$). However, the lack of data combined with the complexity ($k = 10$) seems to lead to overfitting which is the limit of the method. A solution could then be to reduce the number of hidden layers in the neural network.
In the case of no label overlap ($o = 0$), we find the same results as in *S. Mediterranea* in Fig 14. When all the supervised training data are available ($p = 1.0$), *RF* outperforms in the supervised test data for both $k = 2,4,10$ with an accuracy of 75, 79, 80 respectively.

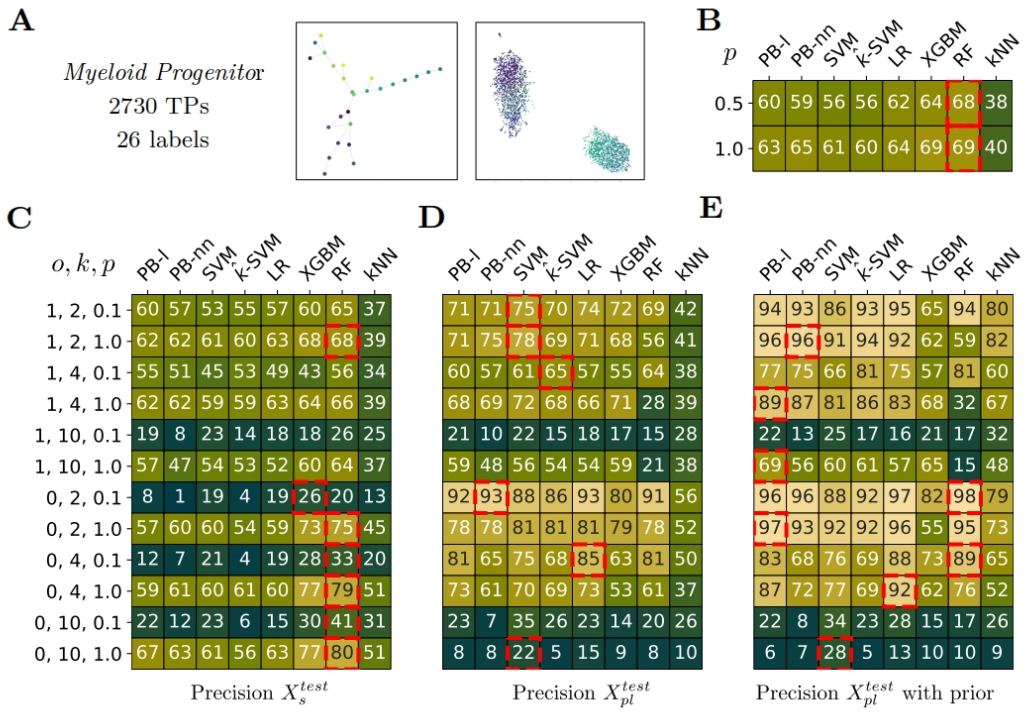

Figure 15 – **Classification performance on *Myeloid progenitors* dataset**
A) *Myeloid progenitors* dataset main features. The topology of the label is close to the *Branches* case. The *UMAP* projection of the annotated transcriptomic profiles, colored according to their respective labels. **B)** Performance of the methods in a fully supervised setting according to the proportion (p) of supervised training data. 0.5 means that models are trained with half of the training data. **C,D,E)** Precision according to the setting in partial label scenarios. Each row corresponds to a setting depending on, by order, o: overlap between supervised and partial label, k: number of partial label set, p: proportion of fully supervised training data. The ith row of C corresponds to the ith row of D and E. The median value of pairwise label distance is 4 and we chose $I = (1,4)$ for partial label sampling. **C)** Precision on X_s^{test}. **D)** Precision on X_{pl}^{test}. **E)** Precision on X_{pl}^{test} when prior information is available for test too. **B-E)** The red squares indicate the best-performing method per row, i.e. per experimental setting, using a t-test and as significance criteria p-value < 0.1.

It indicates one more time, that when there is relatively less information in the partially labeled data, the model focuses on the supervised data.

4.4.5 Incorporating hierarchy improves performance

Including the hierarchical structure of the label set significantly enhances the performance of models. To present the results more concisely, we first described in the

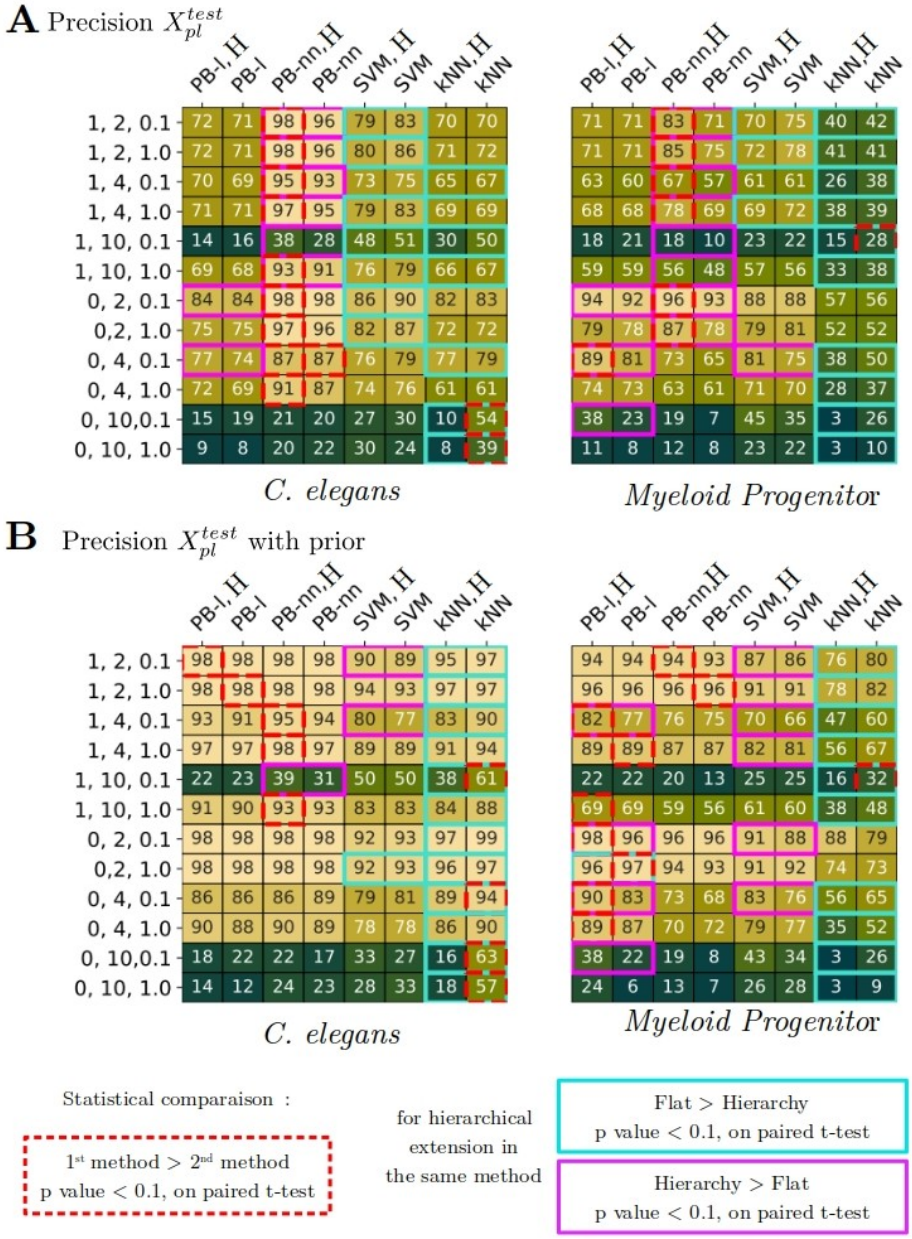

Figure 16 – **Including hierarchical information improves performance**
A-B) Comparison of each method with its hierarchical version, where we take into account H the pairwise distance matrix between the labels in the margin computation. We compute paired t-test and highlight results when hierarchical model are performing significantly better than their non-hierarchical counterparts in magenta, when it's the opposite, in blue. We also report in red, which one of the methods performs significantly better for each experimental setting. **A)** Precision on X_{pl} on *C. elegans* (left table) and *Myeloid progenitor* datasets (right table). **B)** Precision on X_{pl} with prior information on *C. elegans* (left table) and *Myeloid progenitor* datasets (right table).

previous section the results for flat methods which do not take into account hierarchical structures between labels. Now, we aim to compare these flat methods to a hierarchical approach. For the artificial datasets, we found that the hierarchical prototype-based performs equally well as flat-*PB*, and for both *SVM* and *kNN*, flat models were slightly better than hierarchical ones. We will now focus only on the experimental datasets. In the supervised setting, we find that hierarchical models perform with similar performance as Prototype-Based non-hierarchical models, slightly underperform for *SVM*-type models, and similarly for *kNN* models. However, in the partial label setting, hierarchical methods stand out and greatly improve performance. Fig 16 illustrates the methods alongside their hierarchical extensions, and we conducted paired t-tests to highlight instances where hierarchical models perform significantly better than their non-hierarchical counterparts (p-value < 0.1). Moreover, we found that the *PB-nn* method with hierarchy significantly outperformed all other methods (*SVM* and *kNN* in this experiment) in 25 cases among all the 48 experimental settings (52 % of the time) and more generally the *PB* family of methods (flat, hierarchical, linear and nonlinear) significantly outperforms all other methods in 30 cases which represents 62 % of the cases.

Prediction for partial label test set

The first panel of Fig 16, demonstrates that hierarchical *PB* methods consistently outperform their non-hierarchical counterparts. For instance, in the case of *C. elegans* (with $o = 1$, $k = 2$, $p = 0.1$), the performance increases from 0.96 to 0.98. This improvement is even more significant in challenging scenarios (such as when $o = 1$, $k = 10$, $p = 0.1$), where the boost for *C. elegans* is from 0.28 to 0.38, and for *Myeloid progenitors* it is from 0.10 to 0.18.

For the *SVM*, the increase is not systematic, however, it can be detected in some of the cases. In particular, for *Myeloid progenitors*, in very difficult settings ($o = 0$, $k = 10$, $p = 0.1$), the accuracy increases from 0.35 to 0.45.

Boost for prediction with prior information

The hierarchical *PB-nn* method performs better than *PB-nn* (which was initially the best method for *C. elegans*). Similarly, hierarchical *PB-l* outperforms *PB-l* in *Myeloid progenitors*. The hierarchical method *PB-l* highlights the following improvements for no overlap setting ($o = 0, p = 0.1$): ($k = 2$) from 0.96 to 0.98, ($k = 4$) from 0.83 to 0.90, and ($k = 10$) from 0.22 to 0.38.

Additionally, hierarchical *SVM* performs better when a smaller proportion of supervised data is available ($p = 0.1$). In the case of *C. elegans*, the hierarchical method highlights the following improvements for label overlap situation ($o = 1$): ($k = 2$) from 0.89 to 0.90, ($k = 4$) from 0.77 to 0.80, and ($k = 10$) remains at 0.5. For *Myeloid progenitors*, the hierarchical boost for no overlap situation ($o = 0$) is as follows: ($k = 2$) from 0.88 to 0.91, ($k = 4$) from 0.76 to 0.83, and ($k = 10$) from 0.34 to 0.43. In the case of label overlap situation ($o = 1$): ($k = 2$) from 0.86 to 0.87, ($k = 4$) from 0.66 to 0.7, and ($k = 10$) remains at 0.25.

Finally, we find that for both *SVM* and *PB*, the hierarchical extension can significantly improve performance compared to the flat version, especially when a small amount of supervised training data is available.

4.4.6 Supplementary Results

We provide additional experimental results that investigate, first the respective merits of the two algorithms that we introduced in the paper, namely the *Iterative Refinement Learning (IRL)* (Algorithm 1) and the *Iterative Full Retraining (IFR)* (Algorithm 2); secondly the impact of the *SVM* implementation in the nonlinear case.

4.4.6.1 Comparison of Algorithm *IRL* and *IFR*

We now compare the two learning algorithms that we described in detail in the main paper, namely the Iterative Refinement algorithm (*IRL*, Algorithm 1) and the Iterative Full Retraining algorithm (*IFR*, Algorithm 2), which we described in the section 4.2.2. Recall that the difference lies in the optimisation scheme. Given a best candidate in the partial labeling at a time step t, *IRL* performs a single-step gradient re-estimation and then re-estimates the best candidates in the partial labeling; whereas the *IFR* algorithm performs a full training from scratch with the current affected candidates in the partial labeling and the next affectation of best candidates is performed after convergence.

Partially labeled setting

To make a fair comparison between the two algorithms, we ran the same method (Logistic Regression and linear *SVM*, both trained with gradient descent optimization) but optimized with the two different algorithms. The results reported in Fig. 17 show that the *IRL* algorithm significantly outperforms the *IFR* algorithm whatever the dataset and the experimental setting for *SVM* implementations, while it is most often the opposite for *LR* models. We do not have any clear explanation for this behavior given that linear *SVM* and logistic regression are usually considered very similar classifiers. We hypothesize that tuning the hyperparameters of the logistic regression model may be more sensitive and would have required a larger grid search, but we limited the gridsearch effort to a similar budget whatever the method to gain a more fair comparison.

Finally, we observe that for ensemble models, the more expressive the models are, the lower their accuracy on partially labeled data. In particular, for both *RF* and *XGBM*, we noticed an improved accuracy on partially labeled data when reducing the capacity of the model (e.g. lowering the maximal depth of the trees), which seems consistent with the idea that too much expressivity of a classifier, when learned with the *IFR* algorithm, would make it able to learn the first (possibly random) labeling of partially labeled data and hence suffer from strong overtraining.

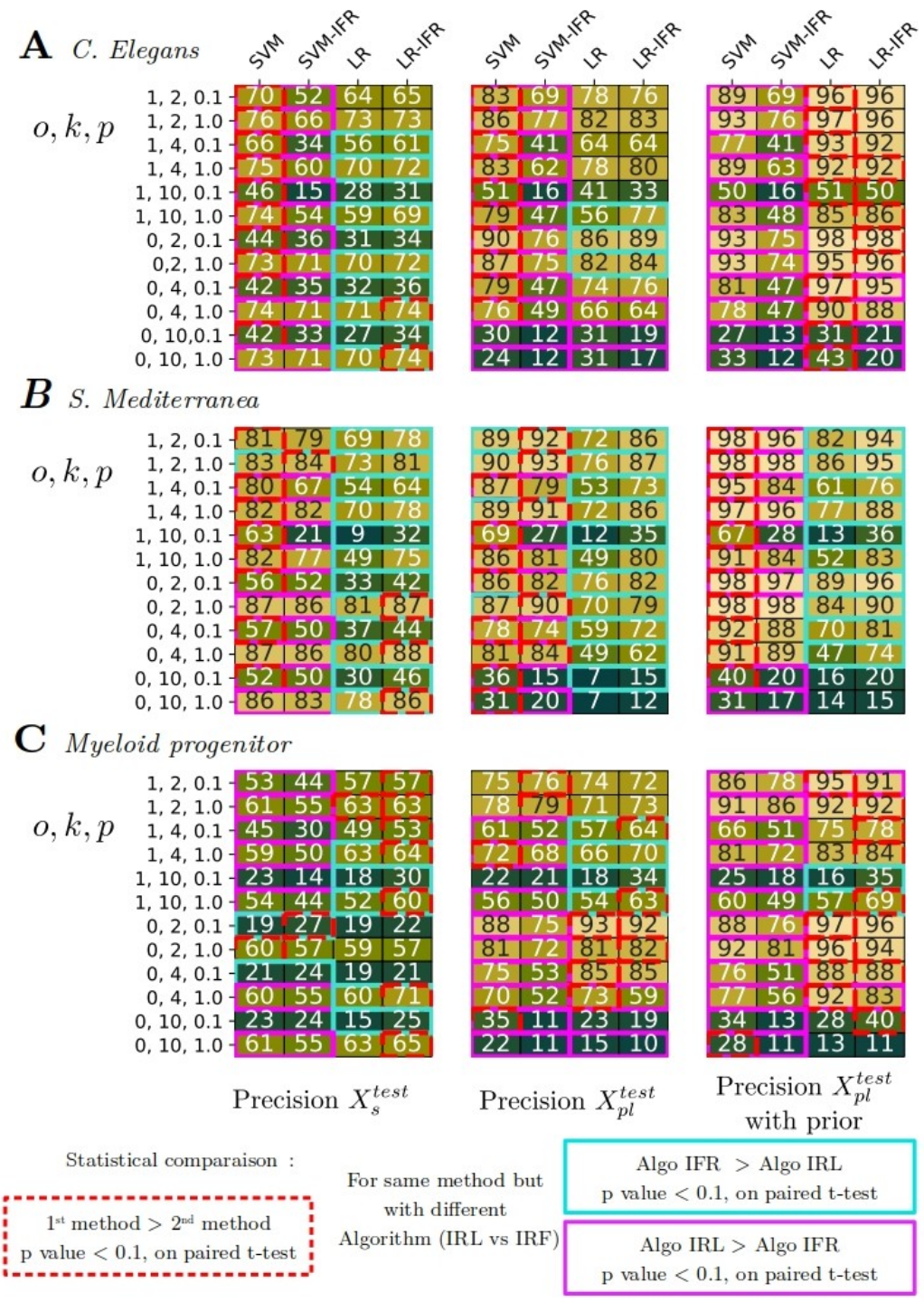

Figure 17 – **Comparison of *IRL* and *IFR* algorithm in the partially labeled scenarios**
A-B-C) Accuracies on the 3 real world datasets, where each heatmap corresponds to performance on X_s^{test}, Precision on X_{pl}^{test}, Precision on X_{pl}^{test} when prior information is available. Each row corresponds to the setting in partial label scenarios, depending on, by order, *o*: Overlap between supervised and partial label, *k*: number of partial label set, *p*: proportion of fully supervised training data. Each column corresponds to the performance of a method, where *SVM* stands for *SVM* with Algo *IRL*, and *SVM -IFR* for *SVM* with Algo *IFR*. The red squares indicate the best performing method per row, i.e. per experimental setting, using a t-test and as significance criteria p-value < 0.1. We also compute paired t-test and highlight the results when model with *IRL* algorithm is performing significantly better than the *IFR* algorithm, we highlighted in magenta and in blue for the opposite situation.

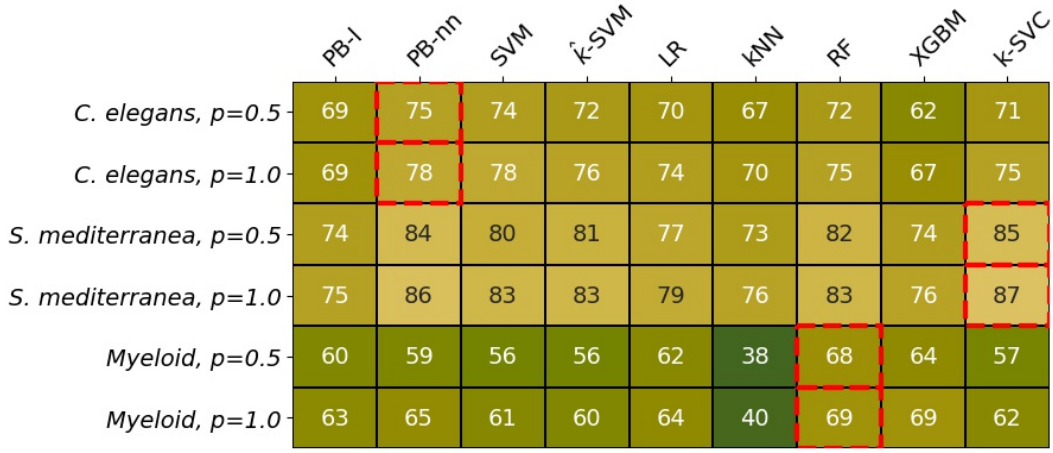

Figure 18 – **Comparison of classification performance of main baseline methods on the three real-world datasets in the fully supervised setting**
Performances (accuracy in %) are reported for two sizes of training datasets (full training dataset $p = 1.0$ or half of it, $p = 0.5$).

4.4.6.2 *SVM* implementation and kernel approximation

As mentioned in section 4.2.2, we implemented nonlinear *SVM* in several ways to deal with the particular settings encountered in our experimental study. In particular, some datasets that we consider are quite large (e.g. all our artificial datasets include hundreds of thousands of training samples) preventing the use of standard implementation of kernel *SVM*, named here *k-SVC* (whose optimization does not scale with the number of training samples). Instead, we relied on the approximation of the Radial Basis Function (RBF) kernel through the Fourrier transformation [179] . This choice enables gradient descent optimization and scales very well with large datasets, we call this method $\hat{k}$-*SVM*.

Supervised results

We first provide in Fig 18 a comparison of all the methods under investigation in a supervised setting. Performances (accuracy) are reported for the three real datasets and two sizes of the training set. One may see here that linear *SVM* and nonlinear *SVM* are similar in all these supervised classification tasks, and that the two nonlinear variants ($\hat{k}$-*SVM* and *k-SVC*) exhibit the same level of performance as the linear variant (*SVM*). More generally one sees that these models are more or less similar with well-performing baselines.

Partially labeled results

Moreover, Fig 19 provides additional results of *SVM* implementations using both learning algorithms. In this figure, we compare the linear *SVM* and the nonlinear $\hat{k}$-*SVM*, both trained with the *IRL* algorithm, to standard kernel *SVM* trained with *IFR* algorithm, *k-SVC*. Compared to the results shown in Fig 17 we report results gained in

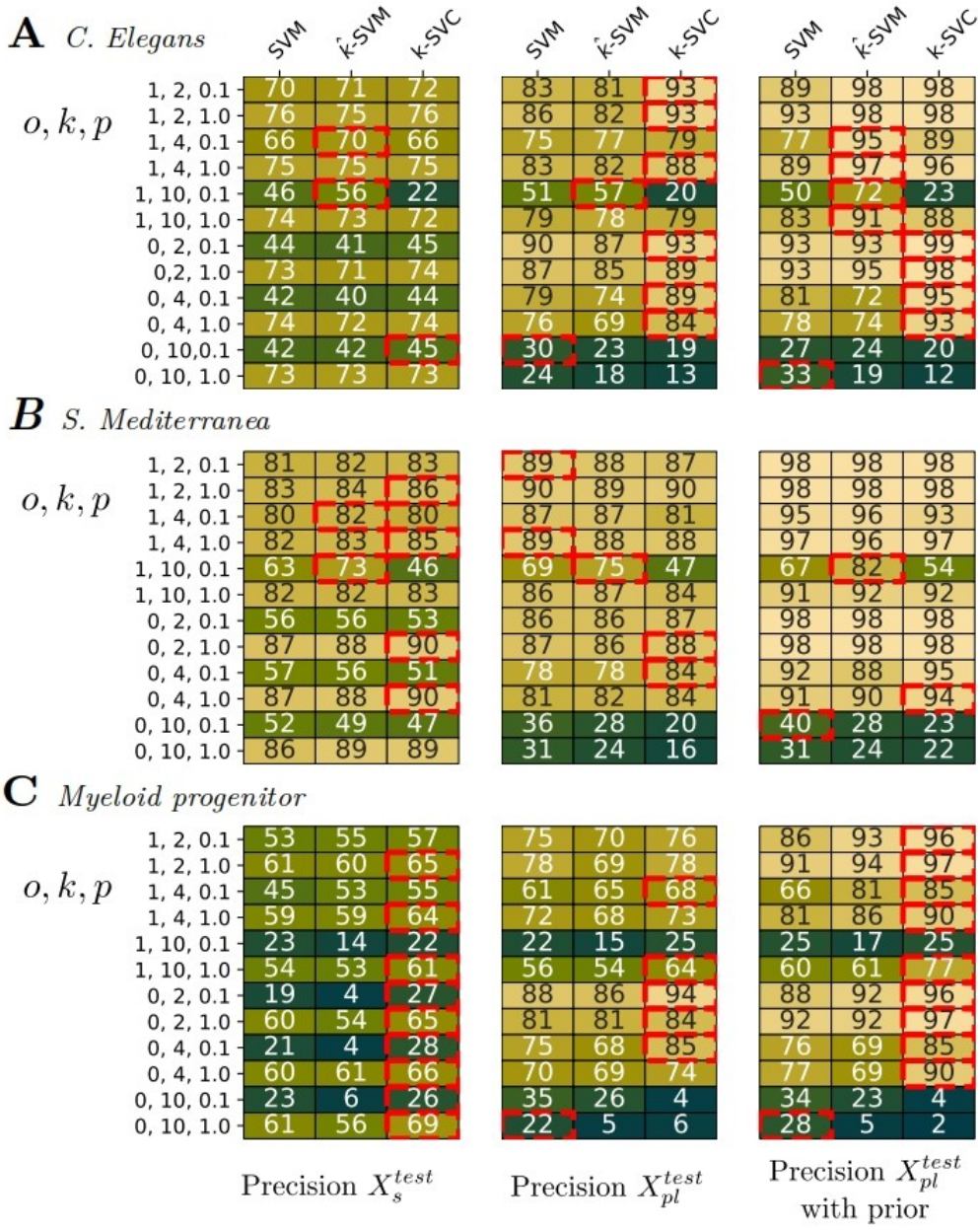

Figure 19 – **Comparison of *SVM* implementations in the partially labeled setting**
A-B-C) Accuracies (in percentage) on the 3 real-world datasets, where each heatmap corresponds to the Precision on X_s^{test}, Precision on X_{pl}^{test}, Precision on X_{pl}^{test} when prior information is available for test too. Each column corresponds to the performance of a method, where *SVM* stands for linear *SVM*, $\hat{k}$-*SVM* for *SVM* with kernel approximation. Both of these implementations rely on Algo *IRL* (as in the main manuscript). Finally, kSVC stands for real kernel and its computation relies on *IFR* algorithm. The red squares indicate the best performing method per row, i.e. per experimental setting, using a t-test and as significance criteria p-value < 0.1.

the partial labeling setting so that one observes both the effect of the implementation and the training algorithm in these results. One sees here again that nonlinear *SVM* relying on a kernel approximation and true kernel *SVM* (*k-SVC*) are most often as good as each other, whatever the dataset and the setting, but with some exceptions where true kernel *SVM* (*k-SVC*) may significantly outperform $\hat{k}$-*SVM*. Yet we chose to systematically focus on the performances of $\hat{k}$-*SVM*, since only this implementation scales to all our datasets. Finally, note that linear *SVM* are most often on par with their nonlinear counterparts.

4.5 Discussion

We proposed several approaches for the classification of partially labeled scRNA Seq data. We provided detailed description of the optimization algorithms used to learn the models. To thoroughly evaluate the performance of all the methods, we devised a comprehensive testing protocol that covered various partial labeling scenarios. This protocol was applied to both simulated and experimental datasets. The experiments demonstrated that our approach, specifically the Prototype-Based Neural Network (*PB-nn*), achieved remarkably high accuracy in a fully supervised setting. Equally impressive was the models performances when only partial label training samples were available, showcasing the models' ability to leverage partial information effectively. Furthermore, we showed that incorporating partially labeled training samples corresponding to certain labels improved the performance on labels for which supervised samples were also available at training time. Our findings also provided insights into the effect of the level of uncertainty in partial labeling, allowing us to achieve the best performance. Ultimately, we found that incorporating hierarchical information could significantly increase performance in partial label settings, particularly in extreme situations with low number of training samples. Our results, showing that we can perform multiclass classification with high accuracy on partially labeled data can be associated to the fact that transcriptomic datasets are highly structured and lie on a low dimensional manifold [212]. Indeed when training a classifier on several transcriptomic profiles, even if they're partially annotated, will necessary take advantage of their geometrical properties. Because we showed that the highest performing partially labeled classifier is a nonlinear Prototype-Based method, we understand that the relationship between cell type identity and position in the feature space is not a linear function. Altogether we found that including the right amount of complexity in the model will lead to a sufficient amount of expressivity to learn the nonlinear distributions of transcriptomic profiles. Moreover, we showed that including the hierarchical structure of the label set as *a priori* information could help improve the models' accuracy. This is in line with the idea of learning the underlying geometry of the set of transcriptomic profiles and suggests moreover that the distribution of transcriptomic profiles in space is consistent with the label set hierarchical structure. Therefore when the number of samples is low, it will be more difficult to learn the underlying geometry of the data as there is less data points to sample the underlying

manifold. Using the hierarchical structure of the label set will compensate by helping reduce the size of the parameter space for the models.

In this work, we focused on one aspect of the structure of the data, namely the hierarchical structure of the label set. We chose to focus on this aspect because of the hierarchical nature of differentiation during development [39], [219]. We envision that our approach could benefit from lineage information obtained directly at the level of data acquisition as it is the case with lineage tracing methods [59]. Moreover, other underlying aspects of the dataset could be influencing the structure of the data. In particular, recent developments in spatial transcriptomics indicate that the spatial organization of cells within a tissue plays an important role in transcriptomic state [96], [243]. Extensions of our approaches could take advantage of this spatial information when it exists. Similarly, recent experimental assays have led to combining multiple -omics [244], automated classification methods could also take advantage of these heterogeneous sources of data to infer cellular identity.

Altogether, we propose an approach that helps reduce the time spent on precise annotation of training datasets by considering sets of candidate labels per sample. In an era where transcriptomic data is being generated increasingly [245], being able to automatize more and with less expert knowledge the task of automatic cell type classification is highly relevant, as the quality of the annotation is currently the bottleneck [40]. Moreover, in addition to the automatic cell type classification task, one of the current challenges is to be able to combine multiple studies and atlases. Our results suggest that we could for example be using the classification results obtained from other already existing annotated datasets as a way to construct partial annotation for a new datasets. Similarly, we could be using tissue level information which is at scale high than cellular by nature and use all of the cell types expected in this tissue as a candidate label set. In conclusion, the flexibility and adaptability of the partial label learning framework is a key step in making transcriptomic single-cell data annotation easier and the corresponding prediction even more routinely used to interrogate the nature of the mechanisms underlying cell differentiation.

4.6 Code availability

The code and data are available on the following GitHub repository:
https://github.com/paulvill/plRNAseq.

5 Hierarchical Novel Class Discovery for classification of scRNA Seq data

Content

5.1 Introduction . . . 96
5.2 Related works . . . 96
5.3 Methods . . . 99
5.3.1 Problem formalization . . . 99
5.3.1.1 Guiding the mapping with prior information . . . 100
5.3.2 Two-step approach . . . 101
5.3.3 Single-step approach . . . 103
5.3.3.1 Hierarchical *k-means* . . . 103
5.3.3.2 Hierarchical Gaussian Mixture Model . . . 104
5.4 Experiments . . . 105
5.4.1 Datasets . . . 105
5.4.2 Experimental settings . . . 105
5.4.3 Metrics . . . 106
5.4.4 Methods and baselines . . . 108
5.5 Results . . . 108
5.6 Discussion . . . 111

We consider in this chapter a frequent setting of experimental transcriptomics datasets, where both labeled and unlabeled data are available at training time, but the sets of the labels of labeled data on one side and of the unlabeled data on the other side, are disjoint. It is an instance of the Novel Class Discovery problem which inherits features from semi-supervised learning, zero-shot learning, and hierarchical classification. The goal is to achieve two objectives, clustering the data and mapping the clusters with labels in the hierarchy. We formalize the problem under investigation and propose two families of approaches, two-step approaches that post-process the result of a clustering or of a novel class discovery method to perform hierarchical classification, and one-step approaches which aim at jointly solving the two components of the problem.

5.1 Introduction

As previously outlined, scRNA Seq datasets usually include various forms of supervision, labeled data, unlabeled data and partially labeled data, hierarchical organization of classes... We focus in this chapter on a setting which occurs in most of the experimental datasets and which differs from the setting we addressed in Chapter 4. We consider the case where both labeled data and unlabeled data are available and one wants to infer the labels of the unlabeled data, where the labels are organized in a hierarchy. Importantly there is no overlap between the set of labels of the labeled data and the set of labels (to infer) of unlabeled data.

The task is an instance of novel class discovery and a case of Semi-Supervised Learning (*SSL*). While a variety of approaches have been proposed for *SSL*, many of these do not match the particular setting of no overlap between the labels of labeled and unlabeled data. For instance, the label propagation strategy is inapplicable, as any supervised sample can propagate the unknown label. To cope with non-overlapping labels of supervised and unsupervised data, two instances of *SSL* are particularly relevant: Zero Shot Learning (*ZSL*) [246] and Novel Class Discovery (*NCD*) [161].

We now introduce or recall useful notations. We consider that we get a labeled dataset $D_s = \{(x_i, y_i) \in \mathbb{R}^d \times \mathcal{Y}_s; i = 1, ..., L\}$, whose samples belong to a set of, what we call, *supervised* labels $\mathcal{Y}_s \subset \mathcal{Y}$, and an unsupervised dataset $D_u = \{x_i \in \mathbb{R}^d; i = L+1, ..., L+U\}$ whose samples belong to a set of, what we call, *unsupervised* labels $\mathcal{Y}_u \subset \mathcal{Y}$, i.e. $\forall x_i \in D_u, y_i \in \mathcal{Y}_u$. As stated above we focus on the particular case where the sets of labels of the samples in D_s and D_u do not overlap, $\mathcal{Y}_s \cap \mathcal{Y}_u = \varnothing$. At last, we are given prior knowledge about the labels that consist of a hierarchical organization of these, i.e. the lineage tree. We note K the number of unsupervised labels, $K = |\mathcal{Y}_u|$ and, following notations introduced in Chapter 2, we note $z_i \in \{1, ..., K\}$ the hidden variable associated with sample $x_i \in D_u$, i.e. z_i the cluster x_i belongs to.

5.2 Related works

We now review state-of-the-art methods that are related to the problem at hand, semi-supervised learning (for classification), zero-shot learning, and novel class discovery.

Semi-Supervised Learning (*SSL*) aims to learn from both labeled and unlabeled data. Many approaches have been shown to be effective in incorporating unlabeled data to improve the prediction quality of classifiers [160]. Two main motivations of semi-supervised learning are, first, to reduce the amount of supervised data and, second, to increase the generalization capacity of a model. Most of the frameworks rely on the hypothesis that supervised and unsupervised data share the same characteristics. Contrary to our case unlabeled and labeled data belong to the same labels. Semi-supervised learning usually consists of minimizing an objective function like the

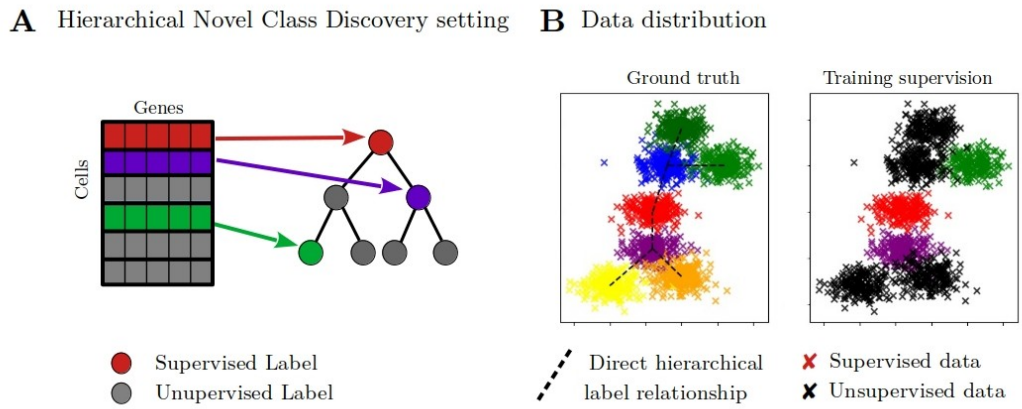

Figure 20 – **Hierarchical Novel Class Discovery Problem statement**
A) The data under consideration are represented by a single-cell RNA sequencing (sc-RNA-Seq) matrix on the left, where each row represents a transcriptomic vector that belongs to a class, where classes are organized in a hierarchy (lineage tree). Part of the data are labeled (colored according to their class) or unlabeled (belong to classes that are illustrated in gray). Unlabeled data belong to classes for which there are no labeled data. **B)** The panel on the left shows the ground truth distribution of the data. The second panel shows the available supervision for training the model, where labeled data are colored and unlabeled data are in plotted in black.

following, with respect to a parametric model f:

$$\sum_{i=1}^{L} loss_s\big(f(x_i), y_i\big) + \lambda_u \sum_{i=L+1}^{L+U} loss_u\big(f(x_i)\big) + \lambda\Omega(f) \tag{5.1}$$

where $loss_s, loss_u$ are loss functions, dedicated to labeled and unlabeled data, f is a parametric model, and Ω is a regularization term. Many methods have been proposed in this line [247], [248], we briefly review approaches that are relevant in our context, pseudo labeling methods, generative models and graph-based methods.

Pseudo-labeling is a commonly used technique in semi-supervised learning, and is an instance of self-training [249], [250]. Methods usually start with training on labeled data only. Then the learning iterates the following steps: a prediction step on unlabeled data, the addition of most confident predictions to the labeled training dataset, and a learning step with the completed training set [251]–[255]. In every iteration, the objective function may be written as:

$$\sum_{i=1}^{L} loss_s\big(f(x_i), y_i\big) + \lambda_u \sum_{i=L+1}^{L+U} loss_s\big(f(x_u), \tilde{y}_u\big) + \lambda\Omega(f) \tag{5.2}$$

where $\tilde{y}_u \in D_u$ is the pseudo-label of unsupervised training sample $x_i \in D_u$. Note that

the unsupervised loss becomes the supervised loss with the current predictions.

Beyond mixture modeling, (e.g. *GMMs*), which are natural probabilistic methods for semi-supervised learning [256], [257], many approaches relying on *generative models* have been proposed [258], [259]–[261], building on the success of techniques such as Generative Adversarial Networks (*GAN*) [262] and Variational AutoEncoders (*VAE*) [263]. The generative feature is indeed appropriate for combining of supervised and unsupervised data [261]. Different strategies have been developed, with disentanglement, whereby the knowledge of supervised data is incorporated into the latent representation [264], or with auxiliary variables [265], [266].

Finally *graph-based methods* rely on the identification of unsupervised data by inference of missing labels with the information provided by the supervised data. The general principle is to encode data as a graph, where nodes correspond to training samples and the weight of edges represents some similarity measurement between nodes. Label inference is performed with different tools from graph theory. The hypothesis behind the inference corresponds to the manifold assumption, which posits that samples that are close in a manifold, have a higher probability of sharing the same labels [248]. Initially, this set of methods were more dedicated to transductive learning [248], but inductive learning was also studied with success [267], [268]. In Graph based Semi-Supervised Learning, the optimization problem usually combines the unsupervised loss and the regularization loss into a graph regularization term Ω_u. The problem consists then of minimizing the loss according to a prediction function f:

$$\sum_{(x_i,y_i)\in D_s} loss_s(f(x_i),y_i) + \lambda\Omega_u(f) \tag{5.3}$$

Usually the weight of the edges of the graph are obtained with a kNN formulation or with a Gaussian kernel [269].

Zero Shot Learning (*ZSL*) is also quite related to our problem. *ZSL* consists of learning to classify data in classes which were not seen at training time, similarly as what occurs in the task we consider. It requires class-level semantic attributes (e.g., word vectors [162]), and it relies on transferring knowledge from the training classes to the target classes. Some works have also studied hierarchical ZSL [270], [271], but these works considered local hierarchical classification only.

Finally, the problem we consider is an instance of what is known under the name Novel class discovery (*NCD*) [272]. The objective is to cluster an unlabeled dataset using guidance from a labeled dataset, with the strong constraint that there is no overlap between classes of labeled data (called hereafter supervised labels) and of unlabeled data (called hereafter unsupervised labels). The first formalization of the task seems to be the one by [273] who used deep transfer clustering. Recently a state-of-the-art approach named *Autonovel* was proposed [274], [275]. The method relies on a classification model which is classically trained in a supervised way with labeled data to discriminate between supervised labels. The model is decomposed into a feature extractor and a classifier. A second classifier is plugged on top of the feature

extractor and aims at discriminating between the unsupervised labels. It is learned to output the same prediction (class) for unsupervised data that are considered similar, where similarity is inferred from the similarity of the encodings of unlabeled samples by the feature extractor.

At the end, the task we aim to deal with exhibits few specificities that prevent from applying standard methods. It is an instance of the *NCD* problem, but where some additional knowledge (the hierarchical organization of classes) may be used in a zero-shot-learning like manner to enable inference in the hierarchical class taxonomy

5.3 Methods

The problem we consider includes clustering the unlabeled data and the mapping of the clusters to unsupervised labels. We first formalize the problem and design the shape of the objective function a method should optimize to cope with the problem. It includes two components, a classical clustering objective, and what we call a label-cluster mapping criterion. Then we detail how we may use prior knowledge, including the knowledge of a hierarchical organization of the classes and its particular meaning for transcriptomic data to define a label-cluster mapping criterion that will enable classification of data in classes for which there are no labeled training samples.

We describe two families of approaches. The first one is a two-step approach which consists of first performing unsupervised clustering, and second determining a mapping of the clusters to unsupervised labels through the optimization of the label-cluster mapping criterion. The approaches of the second family are one-step approaches where the clustering and the label-cluster mapping are optimized simultaneously.

5.3.1 Problem formalization

The learning problem we tackle may be seen as a multi-objective problem where one wants to optimize both a clustering criterion, partitioning of the data into K clusters $(P_1, P_2, ..., P_K)$, and a cluster-label mapping criterion, i.e. finding an optimal (bijective) mapping between the K clusters and the K labels in $\mathcal{Y}_u$. Note that we consider here that there is a one-to-one mapping between clusters and *unsupervised* labels but extensions to more general mapping could be considered. To ease the writing, we rather consider an extended mapping from $\mathcal{Y}$ to the union of $\mathcal{Y}_s$ and of the clusters $\{1, ..., K\}$ ($\mathcal{Y}_s \cup \{1, ..., K\}$) which coincides with the identity mapping on $\mathcal{Y}_s$ (a supervised label are naturally mapped to itself). The mapping we look for is defined as:

$$m : \mathcal{Y} \rightarrow \mathcal{Y}_s \cup \{1, ..., K\} \tag{5.4}$$

$$m(\gamma) = \gamma \text{ if } \gamma \in \mathcal{Y}_s \tag{5.5}$$

$$m(\gamma) \in \{1, ..., K\} \text{ if } \gamma \in \mathcal{Y}_u \tag{5.6}$$

Generally speaking, the learning problem may then be expressed as:

$$\underset{\{P_1,...,P_K\},m}{\operatorname{argmin}} \; L_{clust}(D_s, D_u, \{P_1,...,P_K\}) + L_{map}(m, \{P_1,...,P_K\}, \mathscr{Y}_u) \tag{5.7}$$

where L_{clust} is a clustering criterion (e.g. the one of *k-means* in Eq. 2.20), and L_{map} is a loss term quantifying the goodness of the mapping m.

We will propose two approaches in the following that differ by the way the objective in Eq. (5.7) is minimized: a two-step approach where one first performs clustering by optimizing the clustering loss only, then determines a mapping m by optimizing the mapping loss term while clusters remain fixed; and a single-step approach which aims at jointly learning the clustering and the mapping by minimizing the combined loss.

We will use standard clustering losses, using *k-means* and *GMMs* clustering techniques, and we now discuss the L_{map} loss term. First of all, let's consider a classifier which is built from a clustering and a mapping m which operates as follows. It classifies all samples x_i which belong to a cluster k in the class γ which is mapped to this cluster (i.e. γ is such that $m(\gamma) = k$). Then one may write an evaluation criterion as follows:

$$L_{map}(m, Z, \mathscr{Y}_u) = \frac{1}{L+U} \sum_{\gamma} \sum_{\substack{i=L+1 \\ \text{s.t.} z_i = m(\gamma)}}^{L+U} \mathbb{1}_{[\gamma = y_i]} \tag{5.8}$$

where $\mathbb{1}_{[\gamma=y_i]} = 1$ or 0 depending of γ being equal to y_i or not. The above criterion is the accuracy of the classifier we just defined. Yet, this evaluation criterion cannot be used for learning a mapping as it requires the ground-truth label information. We will propose in the next section a proxy for this criterion to enable learning the mapping.

5.3.1.1 Guiding the mapping with prior information

There is a need for additional information to build an objective function for learning the mapping function. We propose to make use of the known hierarchy of labels, the lineage, and to rely on a rather intuitive assumption, for transcriptomic data, which builds on this hierarchical organization. Indeed, one can assume that the dynamics of the expression of cells in the lineage tree (i.e. during the differentiation process) is continuous and slow, meaning a cell's gene expression should be close to the one of its mother/father cell. Note that the generation process of artificial data with the package [209] does respect this hypothesis. We now turn these assumptions into an optimization term that we may use to learn the mapping.

Let note $\mu_\gamma \in \mathbb{R}^d$ the representative vector, or mean, associated to label/node γ in the hierarchy[1]. Let note $f[\gamma]$ be the father class of a class γ and $\epsilon_\gamma = \mu_\gamma - \mu_{f[\gamma]}$, the assumption translates into $\|\epsilon_\gamma\| = \|\mu_\gamma - \mu_{f[\gamma]}\|$ is likely to be small. Applying the above decomposition ($\mu_\gamma = \mu_{f[\gamma]} + \epsilon_\gamma$) to all labels/nodes in the lineage tree yields, denoting

1. we will call the representative vector of a node a *mean* vector to match the clustering literature although it might not be actually computed as the mean of the node's samples.

$\mathscr{A}(\gamma)$ the set of ancestors in the lineage tree of a node γ.

$$\forall \gamma, \mu_\gamma = \epsilon_\gamma + \sum_{\gamma' \in \mathscr{A}(\gamma)} \epsilon_{\gamma'} \tag{5.9}$$

The above assumption states that ϵ_γ should be low norm vectors. It may be used to define a continuity loss term, L_{cont}, that one may want to minimize:

$$L_{cont} := \sum_{\gamma \in \mathscr{Y}} \|\mu_\gamma - \mu_{f[\gamma]}\|^2 \tag{5.10}$$

$$= \sum_{\gamma \in \mathscr{Y}} \|\epsilon_\gamma\|^2 \tag{5.11}$$

The above criterion is differentiable with respect to the μ vectors (and w.r.t. the ϵ vectors) and may be used as an additional loss term to learn the parameters of a *k-means* or a *GMM* in a one-step approach, as we will detail in section 5.3.3. Alternatively, considering two-step approaches and assuming a clustering of the data has already been performed, yielding $\{\mu_k, k = 1, ..., K\}$ vectors as representative of the clusters, one may search for an optimal mapping m that minimizes the following continuity loss:

$$L_{map}(m, \mu) := \sum_{\gamma \in \mathscr{Y}} \|\mu_{m(\gamma)} - \mu_{m(f[\gamma])}\|^2 \tag{5.12}$$

It is a combinatorial problem for which we propose a greedy solution in the next section.

5.3.2 Two-step approach

The two-step approach we propose is sequential. In the first step, one learns a clustering of the data by minimizing a clustering loss only. In the second step, one optimizes a mapping m while the clusters remain fixed. Any clustering algorithm may be used, we focus in this presentation on *k-means* like algorithms (*k-means* and *GMMs*) and detail our proposal with this choice in mind to ease the reading. In particular we consider a representative "mean" vector μ_k is associated to each cluster k.

Let assume now that clusters have been learned yielding a partitioning $\{P_1, ..., P_K\}$ of the data in D_u, and mean vectors $\{\mu_1, ..., \mu_K\}$, where "mean" vectors are actually representative vectors that do not necessarily coincide as the mean of all samples in a partition. We then want to find a mapping m that minimizes the loss we defined in Eq. (5.12). This label-cluster assignment problem is a combinatorial problem with factorial complexity. Although the problem shape reminds classical graph problems such as the minimum spanning tree problem [276] our problem is slightly different and cannot easily be viewed as an extension of such classical graph problems. We instead investigate solving the problem with a suboptimal greedy algorithm (see Algorithm 3) which iteratively selects a (cluster, label) pair to map, one at a time. It maintains a set

of nodes to be mapped ($S_{\gamma 2map}$ (initialized with $\mathscr{Y}_u$) and a set of clusters that remain to be mapped (initialized with $\{1,...,K\}$), S_{k2map}. It iterates the selection of a best pair to map and the update of the sets $S_{\gamma 2map}$, S_{k2map}.

The selection of the best pair may be performed amongst all possible combinations of remaining unmapped nodes and clusters or it may be restricted to a subset G of these pairs for improved efficiency. restricting the set of potential matches is also a way to implement an ordering strategy for mapping nodes to improve accuracy. The two main components of this algorithm are the definition of G, i.e. which matching priority strategy to apply, and the criterion used to select the best (label, cluster) pair in G. Of course, the sequential nature of the algorithm means that an error cannot be recovered, causing other errors etc. Though one may imagine many extensions to cope with such a cascade of errors problem (using backtracking,beam-search etc) this naive algorithm still provides a useful baseline to assess the performance of the one-step approaches we propose in this chapter.

Let consider now the definition of the score function and of the G subset. First of all, the expression of the continuity loss term in Eq. 5.11 indicates an intuitively good score function for selecting the best (label,cluster) pair to map would be given by $score(\gamma, k) = \|\mu_k - \mu_{f[\gamma]}\|^2$. With this in mind let's consider what happens at some iteration of the algorithm if one considers that G includes all the potential matching pairs (label,cluster), i.e. $G = \{(\gamma, k), \gamma \in S_{\gamma 2map}, k \in S_{k2map}$. As written in Eq. (5.12) the score may be computed for, and only for, nodes γ whose father node's mean, μ, is known (either it is a supervised node or it is a node that has already been mapped with a cluster). Using this simple score function with G including all potential pairs is then not possible. Either another score function or another definition of G must be used. We now detail two strategies that rely on particular choices for G and on the score function.

A simple mapping priority strategy that copes with the problem above involves exploring the label tree in a breadth-first strategy, starting from the root and descending to the leaves, the only requirement is that the root node be supervised (which we will consider true in the following and in our experiments to simplify the problem). Moreover, as it seems a reasonable assumption (e.g. based on the generating process of data in [209] package) that the average distance between father-son classes' samples depends on the depth in the tree, one may enforce comparing scores for labels at the same depth only. Altogether this constitutes a priority mapping strategy, which we call *Breadth First Strategy (BFS)*, where G is defined as follows: Let d be the minimum depth of a (still) unmapped label, then G is the set of all (label, cluster) pairs whose label is (still) unmapped and at depth d and cluster is (still) unmapped.

Another option for designing a priority mapping strategy is to define an alternative score function that does not require the node's father to be supervised or mapped so that a breadth-first exploration is not needed anymore. We investigated the use of the following score function, which is based on distances with representatives of all,

supervised or mapped, ancestors and descendants:

$$score(\gamma, k) = \sum_{\gamma' \in \mathscr{A}(\gamma) \cap S_{\gamma mapped}} \frac{\|\mu_{\gamma'} - \mu_k\|}{H_{\gamma',\gamma}} + \sum_{\gamma' \in \mathscr{D}(\gamma) \cap S_{\gamma mapped}} \frac{\|\mu_{\gamma'} - \mu_k\|}{H_{\gamma',\gamma}} \tag{5.13}$$

where H refers to the distance between labels in the hierarchy that we introduced in section 2.1.2, $\mathscr{A}(\gamma)$ stands for the ancestors of γ in the hierarchy and $\mathscr{D}(\gamma)$ stands for its descendants.

This score function sums the distance from the cluster's mean to the means of all ancestors and descendants, weighted by the inverse of their distance in the tree. We used this score function in priority mapping strategies where G may include (node,cluster) with any node in $S_{\gamma 2map}$. To limit the algorithm complexity and to start with easier cases we consider at one iteration those nodes which have a maximum of their descendants and descendants in $S_{\gamma mapped}$ as the G subset. We refer to this strategy as Maximum Supervision Strategy *(MSS)*.

5.3.3 Single-step approach

We turn to single-step methods where the clustering objective and the cluster-label matching are jointly optimized. We first introduce a hierarchical *k-means* based method (*h-k-means*) which we extend to Gaussian Mixture Models for hierarchies (*h-GMM*).

5.3.3.1 Hierarchical *k-means*

To extend the *k-means* clustering algorithm to the hierarchal case we are interested in, we chose to include the continuity loss in Eq. (5.11) to the global objective function to optimize. The hierarchical *k-means* (*h-k-means*) model we want to learn is then a classical *k-means* on D_u samples, with an additional continuity loss term that will help guide the mapping of clusters to the correct labels in the hierarchy. The objective function is written as:

$$\underset{\{\epsilon_\gamma, \gamma \in \mathscr{Y}_u\}, \{\tilde{y}_i, i=L+1..L+U\}}{\arg\min} \sum_{x_i \in D_s} ||x_i - \mu_{y_i}||^2 + \lambda_u \sum_{x_i \in D_u} ||x_i - \mu_{\tilde{y}_i}||^2 + \lambda_\epsilon \sum_{k=1}^{C} ||\epsilon_k||^2 \tag{5.14}$$

where all μ's are defined as sums of ϵ vectors according to Eq. (5.9), and where $\tilde{y}_i$ are latent variables that need to be optimized.

To solve this problem, we propose to use, as for standard *k-means*, an iterative hard assignment EM algorithm which first updates the pseudo-labeling $\tilde{y}_i$ of the unsupervised data as in the standard *k-means* algorithm based on current parameters,

Algorithme 3 : Label-cluster mapping algorithm

Input :
K: the number of clusters, which is set to $K = |\mathscr{Y}_u|$
$\{\mu_1, ..., \mu_K\}$: the means of the K clusters
$\mathscr{Y}_s$: Set of supervised labels, trivially mapped to themselves
$\{\mu_\gamma, \gamma \in \mathscr{Y}_s\}$: the means of all supervised labels
$\mathscr{Y}_u$: Set of unsupervised labels, i.e. to be mapped
Output :
m: the mapping
Data :
$S_{\gamma 2map}$: Set of labels γ that remain to be mapped
S_{k2map}: Set of clusters k that remain to be mapped
$S_{\gamma mapped}$: Set of clusters that are either supervised or that have been mapped to a cluster
Initialize :
$S_{\gamma 2map} = \mathscr{Y}_u$
$S_{k2map} = \{1, ..., K\}$
$S_{\gamma mapped} = \mathscr{Y}_s$
for $\gamma \in \mathscr{Y}_s$ **do**
  $m(\gamma) = \gamma$
end
while $S_{\gamma 2map} \neq \emptyset$ **do**
  Define $G \subset S_{\gamma 2map} \times S_{k2map}$ `according to Mapping Priority Strategy`
  $(\hat{\gamma}, \hat{k}) = \arg\min_{(\gamma,k)\in G}$ `score`(γ, k)
  $S_{\gamma 2map} = S_{\gamma 2map} \setminus \{\hat{\gamma}\}$
  $S_{\gamma mapped} = S_{\gamma mapped} \cup \{\hat{\gamma}\}$
  $L_{k2map} = L_{k2map} \setminus \{\hat{k}\}$
  $m(\hat{\gamma}) := \hat{k}$
end

according to $\tilde{y}_i = \arg\min_{\gamma\in\mathscr{Y}_u} ||x_i - \mu_\gamma||$ and then reestimate the representative of all unsupervised labels $(\mu_\gamma)_{\gamma\in\mathscr{Y}_u}$ by minimizing the objective function in Eq. (5.14) using gradient descent, while $\tilde{y}_i$ remain fixed. The hyperparameters $\lambda_u, \lambda_\varepsilon$ are set up by gridsearch.

5.3.3.2 Hierarchical Gaussian Mixture Model

We propose Hierarchical Gaussian Mixture Models (*hGMMs*) which extend hierarchical *k-means* like *GMMs* extend *k-means*. We assume that data from D_u follow a *GMM* distribution, with one component per cluster/label.

Like with *h-k-means*, we encode the hierarchical information through the expression

of the mean of each Gaussian component on the mixture:

$$\forall \gamma, \mu_\gamma = \epsilon_\gamma + \sum_{\gamma' \in \mathscr{A}(\gamma)} \epsilon_{\gamma'} \tag{5.15}$$

We note the parameters as $\Theta = \{(\pi_\gamma, \mu_\gamma, \Sigma_\gamma), \gamma \in \mathscr{Y}_u\}$, or equivalently $\Theta = \{(\pi_\gamma, \epsilon_\gamma, \Sigma_\gamma), \gamma \in \mathscr{Y}_u\}$. The optimization is performed through the maximization of the log-likelihood of the data. As we remind it in section 2.2.2, one can optimize the penalized conditional expected likelihood of the complete data $Q(\Theta, \Theta^{(t)})$, given the old values for parameters $\Theta^{(t)}$). The addition of the hierarchical regularization can be done directly in the auxiliary function Q, [277], leading to:

$$\tilde{Q}(\Theta, \Theta^t) = \sum_{i=1}^{N} \sum_{\gamma \in \mathscr{Y}} p(z = \gamma | x_i, \Theta^t) log\, p(x_i, \gamma | \Theta) + log \pi_\gamma - \lambda_\epsilon \sum_{\gamma \in \mathscr{Y}} ||\epsilon_\gamma||^2 \tag{5.16}$$

To solve this problem, we use a regularized Expectation Maximization algorithm, presented in Algo 4. This algorithm is close to the regular EM algorithm, the difference lies in the *M-step* and concerns the optimization of the means. While the $\{(\pi_\gamma, \Sigma_\gamma), \gamma \in \mathscr{Y}_u\}$ parameters are optimized similarly as for standard *GMMs*, the means $\{\mu_\gamma, \gamma \in \mathscr{Y}_u\}$, are not reestimated through an analytical solution due to the regularization term. Yet, as we know it is sufficient to maximize $Q(\Theta, \Theta^t)$ at each iteration to ensure convergence, we optimize $Q(\Theta, \Theta^{(t)})$ with respect to $\{\mu_\gamma, \gamma \in \mathscr{Y}_u\}$ or equivalently to $\{\epsilon_\gamma, \gamma \in \mathscr{Y}_u\}$ through a few steps of gradient ascent [278].

5.4 Experiments

5.4.1 Datasets

We evaluate the methods on artificial and experimental datasets. We used three artificial datasets sampled with the library Prosstt [209]: *Branches-7*, *Binary-4*, and *Binary-6*, where the number corresponds to the height of the lineage tree for the associated topology. The two experimental datasets are scRNA Seq dataset from *Myeloid Progenitor* [211] and *S. Mediteranea* [210]. Both experimental datasets are public and have been projected respectively to the 100 and 50 first components of the Principal Analysis Component method. The main characteristics are summarized in Tab 5.1, and all the remaining details are presented in Chapter 3.

5.4.2 Experimental settings

We investigate the behavior of the methods in various settings, with only a few unsupervised labels up to a large percentage of such labels in $\mathscr{Y}$. The proportion of unsupervised data, $|\mathscr{Y}_u|/|\mathscr{Y}|$, takes values of 0.1, 0.25, and 0.5. For a given percentage value, we create the appropriate datasets by first random sampling of labels in $\mathscr{Y}_s$, $\mathscr{Y}_u$ according to the desired proportion of unsupervised data, with no overlap between

Algorithme 4 : Expectation Maximization for hierarchical Gaussian Mixture Model (*h-GMM*)

Input :
D_u: Unsupervised data set
$\mathscr{Y}_u$: the set of unsupervised labels
K: the number of unsupervised labels $K = |\mathscr{Y}_u|$
$\{\mu_\gamma, \gamma \in \mathscr{Y}_s\}$: the means of the supervised labels
Output :
$\Theta = \{(\pi_\gamma, \mu_\gamma, \Sigma_\gamma), \gamma \in \mathscr{Y}_u\}$
Initialize :
$t = 0$
Initialize parameters $\Theta^{(0)}$
while *model not converged* **do**
- *(E-step)*

$\forall i \in \{L+1, ..., L+U\}, \forall \gamma \in \mathscr{Y}_u, w_i^\gamma := \frac{\pi_\gamma \mathcal{N}(x_i|\mu_\gamma^t, \Sigma_\gamma^t)}{\sum_{\gamma' \in \mathscr{Y}_u} \pi_{\gamma'} \mathcal{N}(x_i|\mu_{\gamma'}^t, \Sigma_{\gamma'}^t)}$

- *(M-step)*

$\forall \gamma \in \mathscr{Y}_u, \pi_\gamma^{(t+1)} = \frac{1}{U} \sum_{i=L+1}^{L+U} w_i^\gamma$

$\forall \gamma \in \mathscr{Y}_u, \Sigma_\gamma^{(t+1)} = \frac{\sum_{i=L+1}^{L+U} w_i^\gamma (x_i - \mu_\gamma^t)(x_i - \mu_\gamma^t)^T}{\sum_{i=L+1}^{L+U} w_i^\gamma}$

$\epsilon^{(t+1)} \equiv \{\epsilon_\gamma^{(t+1)}, \gamma \in \mathscr{Y}_u\} = \arg\max_\epsilon \tilde{Q}(\Theta, \Theta^t)$ where the Q function is defined in Eq. (5.16)

$t = (t+1)$

end

the two sets, then we create X_s and X_u. For each setting (dataset, percentage of unsupervised labels, split between $\mathscr{Y}_s$ and $\mathscr{Y}_u$), we split the data in train-test sets for both supervised $X_s = X_s^{train} \cup X_s^{test}$ and unsupervised datasets $X_u = X_u^{train} \cup X_u^{test}$ with a proportion of 0.2 for the test size. The performances are measured on the X_u^{test} set. For every setting, experiments are repeated 5 times, and reported performances are averaged results. Note that this setup is sensitive to high variation, especially with a small number of labels. Note that, while artificial datasets are balanced (uniform distribution of the labels, experimental datasets are not.

5.4.3 Metrics

We use both clustering metrics (NMI and ACC) and classification metrics (micro-f1) to evaluate the methods. Clustering metrics describe how good the models are at discriminating between unsupervised data from different labels, without considering the mapping problem between clusters and real classes. Classification metrics describe how good the model is at jointly discovering clusters and predicting to which classes

	Branches-7	*Binary-4*	*Binary-6*	*Myeloid Progenitor*	*S. mediterranea*
#features	50	50	50	100	50
#labels	36	31	127	26	51
#samples	2800	1240	5000	2730	7595

Table 5.1 – **Dataset's characteristics**

in the hierarchy they correspond.

The Normalized Mutual Information (NMI) is used to compare two labeling (e.g. output by 2 clustering algorithms). The Mutual Information (MI) is a function that quantifies the degree of agreement between two assignments, it is insensitive to permutations. The NMI metric is a normalization of the Mutual Information, which ranges between 0 (no mutual information) and 1 (perfect correlation). The NMI between two labelings Y and Y' is expressed as:

$$NMI = \frac{2 \times I(Y, Y')}{e(Y) + e(Y')} \tag{5.17}$$

where $e(Y)$ stands for the entropy of the distribution of labels in Y, and$I(Y, Y')$ is the mutual information between the two label distributions Y and Y', computed as $I(Y, Y') = e(Y) - e(Y|Y')$. In the following, we will report NMI as computed between the true distribution of labels of data samples and the distribution of labels output by a clustering algorithm.

The Accuracy Cluster Classification (ACC) is standard practice for evaluating clustering results [279]. Given a ground truth labeling of samples into classes and a labeling output by a clustering algorithm, the ACC metric relies on a mapping of clusters onto classes, where every sample from a given cluster is classified as the class the cluster is mapped to. The ACC is defined as the maximum accuracy one can reach with any possible mapping. Finding the optimal mapping, the one maximizing the accuracy, is not straightforward because it requires solving a combinatorial problem but a sub-optimal mapping may efficiently be obtained using the Hungarian algorithm.

Finally, we use a standard classification metric, the accuracy, which coincides with the micro-f1 score in the multi-class, mono-label classification setting.

We checked the significance of the results with statistical tests. For each experimental setting (one dataset with one proportion of unsupervised labels) we perform paired t-test to compare multiples models. We highlight the significantly better result when the p-value is below 0.1. We performed comparisons of three kinds:

— The best method vs the second best method. In a same row of a table, we highlight significantly better **result** in bold. It concerns the three tables 5.2, 5.3, 5.4.

— The dashed results show the significativity of the comparison of a hierarchical model to its non hierarchical counterpart (*h-k-means* vs *k-means* and *hGMM* vs *GMM*). It concerns the two first Tabs 5.2, 5.3.
— The results with dotted lines show the significativity of the comparison of a single-step approach vs the two-step approach with the mapping algorithm. It concerns only Tab 5.4

5.4.4 Methods and baselines

We investigated several methods:
— Non-hierarchical methods: *k-means*, *GMMs* and *Autonovel* [274]; Note that *k-means* and *GMMs* are implemented with scikitlearn [238] and we used the authors' implementation for *Autonovel*.
— Two steps Hierarchical methods: *Autonovel* followed by the cluster/label matching algorithm (Algorithm 3).
— Single-step Hierarchical methods: *h-k-means* and *h-GMMs*

5.5 Results

We start with the analysis of clustering metrics, see Table 5.2 for the NMI metric and Table 5.3 for the ACC metric.

The results reported in these tables call for two preliminary comments. The first comment concerns the variance of the results which are quite large, especially when the number of unsupervised labels is small. This comes from the fact that if only two or three classes are unsupervised (the classes in $\mathcal{Y}_u$), the data belonging to these classes may be (randomly) drawn from very far to very close labels in the hierarchy (hence likely in the feature space). These large variances explain why, the superiority of methods over others appears significant in some cases, it is not always the case, despite large differences of averaged performances. For instance in the first row of Table 5.2, the hierarchical models look, but are not, significantly better than non-hierarchical methods. The second comment concerns the similarity of the two clustering metrics NMI and ACC, which more or less provide the same information, which is expected here where the number of clusters equals the number of labels. Roughly speaking, whatever the setting (a row) the ranking of the methods is rather similar in both tables.

Next, one may see that all methods perform quite well on the clustering tasks whatever the setting, for artificial as well as for experimental datasets. The results obtained with half unsupervised labels, meaning up to 64 unsupervised labels on artificial datasets (around 0.36 ACC for *GMMs*) and up to 25 unsupervised labels for experimental datasets (around 0.50 ACC for *GMMs*) show that the data are likely not too complex to discriminate. Moreover, standard clustering methods are quite competitive with the state-of-the-art *Autonovel* baseline for the Novel Class Discovery problem.

Finally, one sees that hierarchical methods perform significantly better than their non-hierarchical counterparts in many settings. This was expected but not straightforward as the addition of a hierarchical regularization in the optimization of the models could have hurt the pure clustering performance. For example, for *Binary-6* with 12 unsupervised labels, the NMI of *h-k-means* is at 0.71, while the corresponding *k -means* is 0.65, and that of *h-GMM* is 0.71, while *GMM* is 0.68.

NMI		Methods				
Dataset		Hierarchical model		Baseline		
$\lvert\mathcal{Y}\rvert$	$\lvert\mathcal{Y}_u\rvert$	*h-k-means*	*h-GMM*	*k -means*	*GMM*	*Autonovel*
Branches-7	3	0.75 ± 0.13	0.75 ± 0.13	0.57 ± 0.09	0.57 ± 0.09	0.58 ± 0.15
36	9	0.67 ± 0.03	0.67 ± 0.01	0.60 ± 0.03	0.67 ± 0.04	0.67 ± 0.02
	18	0.58 ± 0.06	0.58 ± 0.05	0.60 ± 0.04	0.63 ± 0.03	**0.67 ± 0.06**
Binary-4	3	0.79 ± 0.14	0.82 ± 0.1	0.70 ± 0.03	0.68 ± 0.1	0.67 ± 0.19
31	7	0.64 ± 0.07	0.63 ± 0.07	0.61 ± 0.07	0.72 ± 0.06	0.68 ± 0.05
	15	0.63 ± 0.03	0.64 ± 0.03	0.59 ± 0.03	0.65 ± 0.02	0.67 ± 0.04
Binary-6	12	0.71 ± 0.05	0.71 ± 0.05	0.65 ± 0.07	0.68 ± 0.07	0.72 ± 0.07
127	32	0.66 ± 0.02	0.66 ± 0.02	0.61 ± 0.02	0.68 ± 0.03	0.61 ± 0.03
	64	0.62 ± 0.01	0.62 ± 0.01	0.61 ± 0.02	**0.66 ± 0.01**	0.48 ± 0.02
Myeloid	2	0.61 ± 0.46	0.61 ± 0.47	0.81 ± 0.36	0.81 ± 0.36	0.61 ± 0.35
Progenitor	6	0.65 ± 0.11	0.61 ± 0.15	0.72 ± 0.15	0.69 ± 0.12	0.59 ± 0.13
26	13	0.66 ± 0.02	0.66 ± 0.03	0.69 ± 0.03	0.69 ± 0.05	0.61 ± 0.05
S. mediterranea	5	0.57 ± 0.09	0.60 ± 0.11	0.50 ± 0.06	0.58 ± 0.08	0.54 ± 0.03
51	12	0.63 ± 0.05	0.63 ± 0.06	0.60 ± 0.07	0.63 ± 0.06	0.64 ± 0.05
	25	0.53 ± 0.03	0.53 ± 0.01	0.59 ± 0.02	**0.63 ± 0.02**	0.49 ± 0.04

Table 5.2 – **Comparison of methods on NMI**
The **results in bold** mean that the best method outperform the second one method, according to a paired t-test with p-value < 0.1. The results in dash compare the hierarchical model to the non hierarchical version (*h-k-means* vs *k-means* and *hGMM* vs *GMM*), with a paired t-test with p-value < 0.1.

We now turn to the classification results reported in Table 5.4. Results (accuracy or f1-score) are reported for four methods, the two one-step hierarchical methods (*h-k-means* and *h-GMM*) and the *Autonovel* method with two variants of the cluster/label mapping algorithm. We report the performance of *Autonovel* only as a two-step method since it was shown to outperform both k-Means and *GMMs* on pure clustering performance. The *BFS* and *MSS* abbreviations stand for *Breadth First Strategy* and *Maximum Supervision Strategy* and distinguish between the two variants of the mapping algorithm. One key element to assess the relevance of the methods lies in comparing the *f1-score* performance of hierarchical methods 5.4 and the ACC performance in Table 5.3. The ACC performance somehow provides an upper bound of the achievable performance so that reaching a f1-score which is close to the ACC

ACC		Methods				
Dataset		Hierarchical model		Baseline		
$\lvert\mathcal{Y}\rvert$	$\lvert\mathcal{Y}_u\rvert$	*h-k-means*	*h-GMM*	*k -means*	*GMM*	*Autonovel*
Branches-7	3	0.90 ± 0.05	0.90 ± 0.05	0.64 ± 0.1	0.64 ± 0.10	0.8 ± 0.10
36	9	0.62 ± 0.03	0.64 ± 0.05	0.52 ± 0.02	0.60 ± 0.03	0.63 ± 0.04
	18	0.44 ± 0.11	0.44 ± 0.09	0.43 ± 0.04	0.47 ± 0.04	**0.54 ± 0.03**
Binary-4	3	0.84 ± 0.13	0.84 ± 0.13	0.76 ± 0.12	0.73 ± 0.15	0.84 ± 0.03
31	7	0.64 ± 0.05	0.69 ± 0.02	0.57 ± 0.07	0.67 ± 0.07	0.65 ± 0.04
	15	0.47 ± 0.07	0.44 ± 0.08	0.43 ± 0.02	0.49 ± 0.04	0.50 ± 0.05
Binary-6	12	0.65 ± 0.08	0.64 ± 0.09	0.50 ± 0.07	0.55 ± 0.10	0.65 ± 0.08
127	32	0.44 ± 0.01	0.44 ± 0.01	0.37 ± 0.02	0.45 ± 0.04	0.30 ± 0.03
	64	0.31 ± 0.01	0.31 ± 0.01	0.29 ± 0.01	**0.36 ± 0.01**	0.14 ± 0.01
Myeloid	2	0.93 ± 0.13	0.93 ± 0.13	0.92 ± 0.14	0.92 ± 0.14	0.88 ± 0.18
Progenitor	6	0.71 ± 0.18	0.78 ± 0.19	0.75 ± 0.14	0.72 ± 0.12	0.60 ± 0.06
26	13	0.67 ± 0.06	0.66 ± 0.05	0.64 ± 0.03	0.66 ± 0.06	0.59 ± 0.06
S. mediterranea	5	0.64 ± 0.10	0.63 ± 0.11	0.47 ± 0.10	0.62 ± 0.09	0.62 ± 0.05
	12	0.56 ± 0.13	0.56 ± 0.12	0.49 ± 0.09	0.53 ± 0.10	0.55 ± 0.04
51	25	0.36 ± 0.11	0.37 ± 0.09	0.42 ± 0.02	**0.49 ± 0.02**	0.30 ± 0.03

Table 5.3 – **Comparison of methods on ACC**
The **results in bold** mean that the best method outperform the second one method, according to a paired t-test with p-value < 0.1. The results in dash compare the hierarchical model to the non hierarchical version (*h-k-means* vs *k-means* and *hGMM* vs *GMM*), with a paired t-test with p-value < 0.1.

metric is very promising.

Let's look at the two-step methods first. The combination of the *Autonovel* method with the mapping algorithm allows reaching moderate performance only. One sees that the gap between the achieved performance in Table 5.4 and the optimal achievable performance in Table 5.3 is quite large, whatever the variant of the mapping algorithm, *BFS* or *MSS*. It appears that the mapping algorithm is far from optimal and exhibits severe weaknesses, in particular the propagation of errors that should be dealt with.

On the other hand, one-step hierarchical methods do perform well on artificial datasets. Very often the f1-score is not far from the ACC performance performance. For instance, with *Binary-4* dataset, *h-k-means* reaches an accuracy of 0.62 (for 7 unsupervised labels) while its ACC was 0.64 (see Tab.5.3). With *Branches-7* *h-GMM* reaches 0.61 f1-score while its ACC was 0.64.

Finally, one sees that the performance are disappointing on experimental datasets except with very few unsupervised labels. It looks like artificial datasets offer favorable settings for hierarchical methods. This is in line with Chapter 4's results where

exploiting the hierarchy was not always bringing any improvement. The hierarchical feature of the experimental datasets is maybe noisy and/or not very strong, while the one of artificial datasets is clearer, given the way the data were built.

f1		Methods			
Dataset		1-step approach		*Autonovel* + Algo mapping	
$\lvert\mathcal{Y}\rvert$	$\lvert\mathcal{Y}_u\rvert$	*h-k-means*	*h-GMM*	*BFS*	*MSS*
Branches-7	3	0.71 ± 0.25	0.83 ± 0.19	0.47 ± 0.31	0.33 ± 0.25
36	9	0.60 ± 0.04	0.61 ± 0.05	0.17 ± 0.03	0.15 ± 0.06
	18	0.29 ± 0.05	0.29 ± 0.06	0.07 ± 0.07	0.05 ± 0.03
Binary-4	3	0.84 ± 0.16	0.84 ± 0.16	0.64 ± 0.24	0.65 ± 0.29
31	7	0.62 ± 0.09	0.58 ± 0.09	0.16 ± 0.14	0.22 ± 0.18
	15	0.36 ± 0.05	0.36 ± 0.07	0.14 ± 0.04	0.13 ± 0.02
Binary-6	12	0.58 ± 0.05	0.57 ± 0.05	0.13 ± 0.02	0.17 ± 0.04
127	32	0.35 ± 0.02	0.35 ± 0.02	0.05 ± 0.02	0.03 ± 0.02
	64	0.19 ± 0.01	0.19 ± 0.01	0.02 ± 0.00	0.02 ± 0.00
Myeloid	2	0.64 ± 0.36	0.59 ± 0.38	0.69 ± 0.33	0.90 ± 0.11
Progenitor	6	0.09 ± 0.16	0.18 ± 0.16	0.12 ± 0.07	0.24 ± 0.09
26	13	0.03 ± 0.03	0.14 ± 0.09	0.10 ± 0.02	0.15 ± 0.09
S. mediterranea	5	0.26 ± 0.11	0.16 ± 0.13	0.23 ± 0.07	0.29 ± 0.17
51	12	0.06 ± 0.03	0.08 ± 0.04	0.09 ± 0.09	0.04 ± 0.03
	25	0.03 ± 0.02	0.03 ± 0.01	0.02 ± 0.01	0.06 ± 0.03

Table 5.4 – **Comparison of methods on classification metric**
The statistical test compares a 1-step approach (either *h-k-means* or *h-GMM*) with the best 2-step approach (*Autonovel* + Algo Mapping). The results with dots lines, means that one of the 1-step methods or one of the 2-step methods is statistically significantly better than the other ones. Note that no methods are significantly better than the second top method (in bold in the last table).

5.6 Discussion

We described in this chapter preliminary works we have done for dealing with a recurrent feature of transcriptomics datasets, when part of the training data are labeled but with a subset only of the set of labels in the lineage hierarchy, and part of the data are unlabeled but are known to belong to another set of labels. The goal is then both to cluster the unlabeled data and to identify which labels the clusters match in the hierarchy. The problem is related to semi-supervised learning, zero shot learning and hierarchical classification.

We detailed a formalization of the problem and proposed two families of methods to solve it. On the one hand, we proposed two-steps methods that rely on a postprocessing of the result of a clustering algorithm to find an optimal match of clusters

to unsupervised labels in the hierarchy. On the other hand we proposed single-step methods that are extensions of clustering methods that jointly optimize the clustering of the data and the matching of clusters to labels in the hierarchy.

Our results show that hierarchical methods do outperform clustering methods with respect to pure clustering metrics, indicating that taking into account the hierarchy may help guiding the clustering of the data. Moreover, we show that the single-step hierarchical methods do reach performances which are often close to an empirically estimated upper bound of their performance. Finally, the two-steps methods although reaching moderate performance are not without merits and could be improved to overcome the cascade of errors problem of the suboptimal and greedy mapping algorithm.

It looks like the methods presented here may reach high performances in favorable and simpler settings, when the hierarchical feature is strongly present in the data and when the percentage of unsupervised labels is moderate, but fail when the percent stage is large and on experimental datasets whose hierarchical feature is less obvious.

6 Conclusion

Content

6.1 Summary of Contributions 113
6.2 Discussion and Perspectives 114
6.2.1 The intersection of developmental biology and bioinformatics brings new challenges 114
6.2.2 Classification of scRNA Seq data to reconstruct cell state trajectory 115
6.2.3 The scale of the label structure 116
6.2.4 Multi-modal data integration 117

6.1 Summary of Contributions

In the first chapter, we introduced the context and challenges that this work contributes to. Our motivation is to reconstruct cell state trajectories by mapping cell states to the cell lineage, employing machine learning methods. We focus on scRNA Seq data in the context of developmental biology. Subsequently, we identified challenges brought by real-world datasets and formalized various statistical learning problems corresponding to different types of supervision. In Chapter 2, we presented state-of-the-art methods for multi-class classification, with and without hierarchy, and the clustering problem. The methods presented in this chapter serve as the basis for the extension on the subsequent two Chapters.

In Chapter 4, we presented the task of partial label learning for scRNA Seq classification. We introduced the formalism of the classification task and proposed a general pipeline to solve it. We studied a set of different methods with partial label extensions, with a focus on methods based on gradient optimization routines. Additionally, we proposed a general algorithm for any supervised classification method. Concurrently, we detailed hierarchical extension for some models, combined with partial label extension. Subsequently, an experimental protocol was developed to test and study a set of different configurations through parameters and different scenarios of learning, either with label overlap or without.

In Chapter 5, we formalized the task of identifying different cell types in a semi-supervised learning problem that involves learning with both labeled and unlabeled data. We provide details on the learning task of novel class discovery and how to

incorporate the hierarchy of labels given by the lineage tree. The objective of this chapter is to perform relevant clusterings of unsupervised data and then to identify these clusters to the cell types within a structured set of labels. We have developed an extension of two state-of-the-art methods in clustering. Additionally, we have provided an alternative solution that combines any clustering algorithm with a mapping algorithm.

6.2 Discussion and Perspectives

In this manuscript, we studied several classification problems for scRNA Seq data. The set of methods and experiments have been developed to address the various issues encountered in these problems, notably the lack of supervision for the application of traditional classification methods. We now discuss the choices made within this framework and the problems encountered.

6.2.1 The intersection of developmental biology and bioinformatics brings new challenges

This dissertation is a multidisciplinary research work involving developmental biology, computer science, and applied mathematics. The intersection of these disciplines is becoming increasingly prevalent in current research topics, presenting new challenges that may not be encountered in non-multidisciplinary work.

Impact within the biological community

The thesis introduces a collection of automated techniques for classifying and clustering of transcriptomic profiles. These methods bring numerous benefits, notably by streamlining the labor-intensive manual annotation process. However, there are some limitations of these methods that make their use by biologists not straightforward.

The automated methods developed by the computer science and machine learning communities are typically tested only on benchmark datasets, and the application of these methods to new datasets is not straightforward. In biology, however, we encounter various systems such as different species or scale (molecules, cells, tissues, ...). Each of these various systems have their own characteristics, and in addition the data can vary greatly for a given system given the state at which it is measured (e.g. at which time point in development). Therefore the general pipeline of pre-processing data and the choice of the classification method require an additional step for each specific dataset.

In general, researchers from other communities tend to prefer dedicated software that includes various tools such as data pre-processing, visualization, and comparison between different datasets. A broad applicability of the methods developed in this manuscript could require an additional software development step to be integrated in typical bioinformatics environment [280].

The application of computational methods enables the identification of underlying principles. The hypotheses formulated here in the context of developmental biology are not immediately evident from measurement techniques and are based on knowledge obtained previously in other studies. For instance, the fact that scRNA Seq data exhibit a hierarchical structure, and cell state trajectories are nonlinear, continuous processes. Statistical analysis can help confirm these hypotheses.

The applications of machine learning stand to gain considerably from collaboration with biologists. The integration of prior knowledge can significantly enhance the performance of different models and reduce the complexity of the problem. The adaptation of current models and the development of new ones benefit from the integration of the knowledge provided by the research in developmental biology.

Impact within the machine learning community

The analysis and statistical treatment of the data generated by developmental biology research present new challenges to the machine-learning community.

The analysis and comparison of methods for classification tasks is typically conducted on existing benchmark datasets and experimental protocols. However, these benchmarks are not consistently available and are currently being developed [281].

The cell types as labels are not yet clearly defined for scRNA Seq data [144], and the identification of cell types may vary according to different research works. Currently, it is not yet straightforward to compare and extend methods across articles and research topics.

The data presents a variety of challenges and requires specific design and architecture. Either multi-omic or microscopy data presents multiple sources of noise and high dimensionality. For the various machine learning tasks, especially classification, the methods are improved when the amount of data increases.
In recent years, the machine learning community has shifted its focus towards other types of data, such as images and speech processing, where the amount of data is nearly unlimited. However, in developmental biology, the measurement techniques are complicated and expensive, resulting in a limited amount of data.
Overfitting is a common problem in machine learning and deep learning. The selection of model architecture is of paramount importance, contingent upon the nature and attributes of the data. Traditional deep learning models that require millions of samples to be trained, may be computationally expensive. In practice, acquiring this amount of data may be infeasible.

6.2.2 Classification of scRNA Seq data to reconstruct cell state trajectory

In the contribution chapters, we formalized the classification problem associated to scRNA Seq data.

Classification problem

The issue of automated classification presents potential solutions to a number of challenges encountered in the field of developmental biology and other domains. The first of these solutions enables the labeling of data that would otherwise be challenging to identify. The complexity of the underlying mechanisms, the number of labels, and the high dimensionality of the data all contribute to the difficulty of analyzing the data. Moreover, the labeling of data is of benefit to numerous applications and research areas, allowing for more rigorous supervision and benchmarks for hypothesis verification and scientific discoveries.

In the absence of supervision, one strategy proposed in the manuscript is to disambiguate the data and refine the assignment iteratively. We based our approach on the Expectation maximization algorithm which is a fundamental algorithm for this problem. The different approaches utilized are based on inductive learning, with the objective of generalization for new data. However, an alternative could be the implementation of transductive learning, where the generalization is not taken into account. The distinct strategies employed are disparate, and it appears promising for application in order to disambiguate scRNA Seq data.

scRNA Seq and other modalities

Other modalities and measurement techniques permit the use of complementary data to describe the cell state. For instance, different omic data such as protein expression, ATAC-Seq, which describes the amount of accessibility of chromatin, can be employed. These data sets share common difficulties, including noise, high dimensionality, and a limited amount of training data. However, the different methods developed in this manuscript can be adapted to address these challenges in different modalities. A promising further research is the investigation of the integration of the spatial organization of cells during development. Currently, there are several challenges related to the classification and mapping of omic-spatial data. All of these features are pertinent to the global issue of cell trajectory reconstruction. The diverse array of features provides a comprehensive and nuanced understanding of the problem, which motivates the multi-modal framework.

6.2.3 The scale of the label structure

Models are easily adaptable to different hierarchical structures

In the various approaches, we assume that the hierarchy is known. In detail, we use either the lineage tree at the scale of single cell binary division, where each sample corresponds to a unique node in a lineage tree; or a hierarchical organization of clusters of cell types. Both structures can be implemented and integrated into the various pipelines, requiring only pairwise relationships between labels.

Different types of hierarchical models

Integration of the hierarchy in models does not provide systemic gain [185], it is an open research domain to prove the efficiency of adding hierarchical knowledge for the different tasks and settings in the analysis of the scRNA Seq data. The majority of the research work focuses on global hierarchical models (where only the leaf nodes are labels). This approach is not directly well suited for the hierarchical structure of a tree, particularly when the number goes high because it can multiply the error and involves a very high number of classifiers. To illustrate this, when models adopt a top-down approach by training a classifier at each division in the tree, an error at the top of the tree is fateful and provides incorrect training data for all the next classifiers.

Different types of hierarchy

We discuss only a few of the different ways to choose a hierarchical structure. The first is to infer the entire structure, but this remains a challenging problem and a whole area of research [27]. An alternative is to infer only the local structure given a hierarchy, for instance evaluating different rearrangements of the hierarchy, such as adding or subtracting nodes, between two levels of the hierarchy. The objective would be to measure each type of rearrangement of the hierarchy and evaluate whether it brings relevant changes for cell type identification. The measure can be expressed in terms of performance on different machine learning tasks such as classification, clustering, or feature extraction. One can imagine a workflow, as in the wrapper methods for feature extraction, that consists of a two-step process where the training of the classifier can evaluate the most informative feature by testing different selections of features. Those can be extended by varying the different hierarchical rearrangements and evaluating them with machine learning models.

6.2.4 Multi-modal data integration

A complementary approach for cell state reconstruction is muti-modal data integration, where the different modalities are provided by multi-omic data and microscopy data. These different modalities provide complementary measures of the cell state [130], [133], [282], and offer a variety of applications, such as individual modality analysis, pairwise modality mapping [129], dimensionality reduction [283] and generative model [126].

The hierarchical classification framework is promising for the integration of multi-modalities

One of major challenges of multi-modal integration is the alignment of the different modalities [284]. Different alignment techniques are currently being developed, such as dictionary learning [132], clustering [125] , and common latent embedding space [126]. The different approaches and methods of classification can provide annotations and supervision, which establish the pairwise identification of data distribution across different modalities. For example, prior knowledge can be provided by a reference

cell atlas, and classification methods learned from these models predict the different cell types for each modality. Furthermore, the incorporation of a known hierarchy can disambiguate some data and, in parallel, allow for a more appropriate latent representation and alignment between modalities.

Bibliography

[1] Lewis Wolpert, Cheryll Tickle, and Alfonso Martinez Arias. *Principles of development*. Oxford University Press, USA, 2015 (cit. on p. 16).

[2] Brigid LM Hogan. "Morphogenesis". In: *Cell* 96.2 (1999), pp. 225–233 (cit. on p. 16).

[3] Daniel D Shaye and Martha C Soto. "Epithelial morphogenesis, tubulogenesis and forces in organogenesis". In: *Current topics in developmental biology* 144 (2021), pp. 161–214 (cit. on p. 16).

[4] H Holtzer, H Weintraub, R Mayne, et al. "The cell cycle, cell lineages, and cell differentiation". In: *Current topics in developmental biology* 7 (1972), pp. 229–256 (cit. on p. 16).

[5] Morris F Maduro. "Cell fate specification in the C. elegans embryo". In: *Developmental dynamics: an official publication of the American Association of Anatomists* 239.5 (2010), pp. 1315–1329 (cit. on p. 17).

[6] Sulston et al. "The embryonic cell lineage of the nematode Caenorhabditis elegans". In: *Developmental biology* (1983) (cit. on pp. 16, 17, 27, 55, 83).

[7] Anthony A Hyman and John G White. "Determination of cell division axes in the early embryogenesis of Caenorhabditis elegans." In: *The Journal of cell biology* 105.5 (1987), pp. 2123–2135 (cit. on pp. 16, 21).

[8] Nicolas Olivier, Miguel A Luengo-Oroz, Louise Duloquin, et al. "Cell lineage reconstruction of early zebrafish embryos using label-free nonlinear microscopy". In: *Science* 329.5994 (2010), pp. 967–971 (cit. on pp. 16, 21).

[9] Kipp Weiskopf, Peter J Schnorr, Wendy W Pang, et al. "Myeloid cell origins, differentiation, and clinical implications". In: *Myeloid Cells in Health and Disease: A Synthesis* (2017), pp. 857–875 (cit. on pp. 17, 26, 56).

[10] Gerald J Spangrude, Shelly Heimfeld, and Irving L Weissman. "Purification and characterization of mouse hematopoietic stem cells". In: *Science* 241.4861 (1988), pp. 58–62 (cit. on pp. 17, 56).

[11] Chelsea L Gregg and Jonathan T Butcher. "Quantitative in vivo imaging of embryonic development: opportunities and challenges". In: *Differentiation* 84.1 (2012), pp. 149–162 (cit. on p. 17).

[12] Pascal De Boer, Jacob P Hoogenboom, and Ben NG Giepmans. "Correlated light and electron microscopy: ultrastructure lights up!" In: *Nature methods* 12.6 (2015), pp. 503–513 (cit. on p. 17).

[13] Nils Gustafsson, Siân Culley, George Ashdown, et al. “Fast live-cell conventional fluorophore nanoscopy with ImageJ through super-resolution radial fluctuations”. In: *Nature communications* 7.1 (2016), p. 12471 (cit. on p. 17).

[14] Sharada Iyer, Sulagna Mukherjee, and Megha Kumar. “Watching the embryo: Evolution of the microscope for the study of embryogenesis”. In: *BioEssays* 43.6 (2021), p. 2000238 (cit. on p. 17).

[15] Jan Huisken and Didier YR Stainier. “Selective plane illumination microscopy techniques in developmental biology”. In: (2009) (cit. on p. 17).

[16] Arthur M Lesk. *Introduction to genomics*. Oxford University Press, 2017 (cit. on p. 18).

[17] Rohan Lowe, Neil Shirley, Mark Bleackley, et al. “Transcriptomics technologies”. In: *PLoS computational biology* 13.5 (2017), e1005457 (cit. on p. 18).

[18] Scott D Patterson and Ruedi H Aebersold. “Proteomics: the first decade and beyond”. In: *Nature genetics* 33.3 (2003), pp. 311–323 (cit. on p. 18).

[19] Aihua Zhang, Hui Sun, Ping Wang, et al. “Modern analytical techniques in metabolomics analysis”. In: *Analyst* 137.2 (2012), pp. 293–300 (cit. on p. 18).

[20] Pauline A Callinan and Andrew P Feinberg. “The emerging science of epigenomics”. In: *Human molecular genetics* 15.suppl_1 (2006), R95–R101 (cit. on p. 18).

[21] Bonnie Berger, Jian Peng, and Mona Singh. “Computational solutions for omics data”. In: *Nature reviews genetics* 14.5 (2013), pp. 333–346 (cit. on p. 18).

[22] Matteo Bersanelli, Ettore Mosca, Daniel Remondini, et al. “Methods for the integration of multi-omics data: mathematical aspects”. In: *BMC bioinformatics* 17 (2016), pp. 167–177 (cit. on p. 18).

[23] Chongyang Chen, Jing Wang, Donghui Pan, et al. “Applications of multi-omics analysis in human diseases”. In: *MedComm* 4.4 (2023), e315 (cit. on p. 18).

[24] Bertrand Bénazéraf, Mathias Beaupeux, Martin Tchernookov, et al. “Multi-scale quantification of tissue behavior during amniote embryo axis elongation”. In: *Development* 144.23 (2017), pp. 4462–4472 (cit. on p. 18).

[25] Lara Busby and Benjamin Steventon. “Tissue tectonics and the multi-scale regulation of developmental timing”. In: *Interface Focus* 11.3 (2021), p. 20200057 (cit. on p. 18).

[26] Sayantan Dutta, Aleena L Patel, Shannon E Keenan, et al. “From complex datasets to predictive models of embryonic development”. In: *Nature Computational Science* 1.8 (2021), pp. 516–520 (cit. on p. 18).

[27] Sadie VanHorn and Samantha A Morris. “Next-generation lineage tracing and fate mapping to interrogate development”. In: *Developmental cell* 56.1 (2021), pp. 7–21 (cit. on pp. 18, 19, 21, 117).

[28] Lu Wen and Fuchou Tang. "Recent advances in single-cell sequencing technologies". In: *Precision clinical medicine* 5.1 (2022), pbac002 (cit. on p. 18).

[29] Yan Wu and Kun Zhang. "Tools for the analysis of high-dimensional single-cell RNA sequencing data". In: *Nature Reviews Nephrology* 16.7 (2020), pp. 408–421 (cit. on p. 18).

[30] David G Spiller, Christopher D Wood, David A Rand, et al. "Measurement of single-cell dynamics". In: *Nature* 465.7299 (2010), pp. 736–745 (cit. on p. 18).

[31] Nils Eling, Michael D Morgan, and John C Marioni. "Challenges in measuring and understanding biological noise". In: *Nature Reviews Genetics* 20.9 (2019), pp. 536–548 (cit. on p. 18).

[32] Jeff Gauthier, Antony T Vincent, Steve J Charette, et al. "A brief history of bioinformatics". In: *Briefings in bioinformatics* 20.6 (2019), pp. 1981–1996 (cit. on pp. 18, 22).

[33] Paul Villoutreix. "What machine learning can do for developmental biology". In: *Development* (2021) (cit. on pp. 18, 58).

[34] Giovanni Pasquini, Jesus Eduardo Rojo Arias, Patrick Schäfer, et al. "Automated methods for cell type annotation on scRNA-seq data". In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 961–969 (cit. on pp. 18, 25, 58).

[35] Fuyong Xing and Lin Yang. "Robust nucleus/cell detection and segmentation in digital pathology and microscopy images: a comprehensive review". In: *IEEE reviews in biomedical engineering* 9 (2016), pp. 234–263 (cit. on p. 18).

[36] Omneya Attallah, Maha A Sharkas, and Heba Gadelkarim. "Deep learning techniques for automatic detection of embryonic neurodevelopmental disorders". In: *Diagnostics* 10.1 (2020), p. 27 (cit. on pp. 18, 19).

[37] Mikkel Fly Kragh and Henrik Karstoft. "Embryo selection with artificial intelligence: how to evaluate and compare methods?" In: *Journal of Assisted Reproduction and Genetics* 38.7 (2021), pp. 1675–1689 (cit. on p. 18).

[38] Cole Trapnell. "Defining cell types and states with single-cell genomics". In: *Genome research* 25.10 (2015), pp. 1491–1498 (cit. on pp. 18, 58).

[39] Silvia Domcke and Jay Shendure. "A reference cell tree will serve science better than a reference cell atlas". In: *Cell* 186.6 (2023), pp. 1103–1114 (cit. on pp. 18, 21, 58, 59, 94).

[40] Bingbing Xie, Qin Jiang, Antonio Mora, et al. "Automatic cell type identification methods for single-cell RNA sequencing". In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 5874–5887. ISSN: 2001-0370. DOI: https://doi.org/10.1016/j.csbj.2021.10.027. URL: https://www.sciencedirect.com/science/article/pii/S2001037021004499 (cit. on pp. 18, 58, 94).

[41] Pablo Moreno, Silvie Fexova, Nancy George, et al. “Expression Atlas update: gene and protein expression in multiple species”. In: *Nucleic acids research* 50.D1 (2022), pp. D129–D140 (cit. on p. 18).

[42] Lisa Sikkema, Ciro Ramírez-Suástegui, Daniel C Strobl, et al. “An integrated cell atlas of the lung in health and disease”. In: *Nature Medicine* 29.6 (2023), pp. 1563–1577 (cit. on p. 18).

[43] Philipp Angerer, Lukas Simon, Sophie Tritschler, et al. “Single cells make big data: new challenges and opportunities in transcriptomics”. In: *Current opinion in systems biology* 4 (2017), pp. 85–91 (cit. on p. 19).

[44] Matthew Brendel, Chang Su, Zilong Bai, et al. “Application of deep learning on single-cell RNA sequencing data analysis: a review”. In: *Genomics, Proteomics & Bioinformatics* 20.5 (2022), pp. 814–835 (cit. on p. 19).

[45] Adam Joseph Ronald Pond, Seongwon Hwang, Berta Verd, et al. “A deep learning approach for staging embryonic tissue isolates with small data”. In: *Plos one* 16.1 (2021), e0244151 (cit. on p. 19).

[46] Qin Ma and Dong Xu. “Deep learning shapes single-cell data analysis”. In: *Nature Reviews Molecular Cell Biology* 23.5 (2022), pp. 303–304 (cit. on p. 19).

[47] Mario Flores, Zhentao Liu, Tinghe Zhang, et al. “Deep learning tackles single-cell analysis—a survey of deep learning for scRNA-seq analysis”. In: *Briefings in bioinformatics* 23.1 (2022), bbab531 (cit. on pp. 19, 29).

[48] Zixiang Luo, Chenyu Xu, Zhen Zhang, et al. “A topology-preserving dimensionality reduction method for single-cell RNA-seq data using graph autoencoder”. In: *Scientific reports* 11.1 (2021), p. 20028 (cit. on p. 19).

[49] Jiarui Ding, Anne Condon, and Sohrab P Shah. “Interpretable dimensionality reduction of single cell transcriptome data with deep generative models”. In: *Nature communications* 9.1 (2018), p. 2002 (cit. on p. 19).

[50] Cédric Arisdakessian, Olivier Poirion, Breck Yunits, et al. “DeepImpute: an accurate, fast, and scalable deep neural network method to impute single-cell RNA-seq data”. In: *Genome biology* 20.1 (2019), pp. 1–14 (cit. on p. 19).

[51] Hui Li, Cory R Brouwer, and Weijun Luo. “A universal deep neural network for in-depth cleaning of single-cell RNA-Seq data”. In: *Nature Communications* 13.1 (2022), p. 1901 (cit. on p. 19).

[52] Maria Brbić, Marinka Zitnik, Sheng Wang, et al. “MARS: discovering novel cell types across heterogeneous single-cell experiments”. In: *Nature methods* 17.12 (2020), pp. 1200–1206 (cit. on p. 19).

[53] Zhengyang Dong and Gil Alterovitz. “netAE: semi-supervised dimensionality reduction of single-cell RNA sequencing to facilitate cell labeling”. In: *Bioinformatics* 37.1 (2021), pp. 43–49 (cit. on p. 19).

[54] A Ali Heydari, Oscar A Davalos, Lihong Zhao, et al. "ACTIVA: realistic single-cell RNA-seq generation with automatic cell-type identification using introspective variational autoencoders". In: *Bioinformatics* 38.8 (2022), pp. 2194–2201 (cit. on p. 19).

[55] Jeffrey A Farrell, Yiqun Wang, Samantha J Riesenfeld, et al. "Single-cell reconstruction of developmental trajectories during zebrafish embryogenesis". In: *Science* 360.6392 (2018), eaar3131 (cit. on p. 19).

[56] Carla Belmonte-Mateos and Cristina Pujades. "From cell States to cell fates: how cell proliferation and neuronal differentiation are coordinated during embryonic development". In: *Frontiers in Neuroscience* 15 (2022), p. 781160 (cit. on p. 19).

[57] Gary Thomas. "Furin at the cutting edge: from protein traffic to embryogenesis and disease". In: *Nature reviews Molecular cell biology* 3.10 (2002), pp. 753–766 (cit. on p. 19).

[58] Julia L Moore, Zhuo Du, and Zhirong Bao. "Systematic quantification of developmental phenotypes at single-cell resolution during embryogenesis". In: *Development* 140.15 (2013), pp. 3266–3274 (cit. on p. 19).

[59] Daniel E Wagner and Allon M Klein. "Lineage tracing meets single-cell omics: opportunities and challenges". In: *Nature Reviews Genetics* 21.7 (2020), pp. 410–427 (cit. on pp. 19, 20, 23, 59, 94).

[60] Jin Wang, Kun Zhang, Li Xu, et al. "Quantifying the Waddington landscape and biological paths for development and differentiation". In: *Proceedings of the National Academy of Sciences* 108.20 (2011), pp. 8257–8262 (cit. on pp. 19, 23).

[61] Christoph Bock, Matthias Farlik, and Nathan C Sheffield. "Multi-omics of single cells: strategies and applications". In: *Trends in biotechnology* 34.8 (2016), pp. 605–608 (cit. on p. 19).

[62] Louise Deconinck, Robrecht Cannoodt, Wouter Saelens, et al. "Recent advances in trajectory inference from single-cell omics data". In: *Current Opinion in Systems Biology* 27 (2021), p. 100344 (cit. on p. 19).

[63] Fernando Amat, William Lemon, Daniel P Mossing, et al. "Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data". In: *Nature methods* 11.9 (2014), pp. 951–958 (cit. on p. 21).

[64] Jean-Baptiste Lugagne, Haonan Lin, and Mary J Dunlop. "DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning". In: *PLoS computational biology* 16.4 (2020), e1007673 (cit. on p. 21).

[65] Lennart Kester and Alexander Van Oudenaarden. "Single-cell transcriptomics meets lineage tracing". In: *Cell stem cell* 23.2 (2018), pp. 166–179 (cit. on p. 21).

[66] Zizhang Li, Wenjing Yang, Peng Wu, et al. "Reconstructing cell lineage trees with genomic barcoding: Approaches and applications". In: *Journal of Genetics and Genomics* (2023) (cit. on p. 21).

[67] Caleb Weinreb, Alejo Rodriguez-Fraticelli, Fernando D Camargo, et al. “Lineage tracing on transcriptional landscapes links state to fate during differentiation”. In: *Science* 367.6479 (2020), eaaw3381 (cit. on p. 21).

[68] Sarah Bowling, Duluxan Sritharan, Fernando G Osorio, et al. “An engineered CRISPR-Cas9 mouse line for simultaneous readout of lineage histories and gene expression profiles in single cells”. In: *Cell* 181.6 (2020), pp. 1410–1422 (cit. on p. 21).

[69] Daniel E Wagner, Caleb Weinreb, Zach M Collins, et al. “Single-cell mapping of gene expression landscapes and lineage in the zebrafish embryo”. In: *Science* 360.6392 (2018), pp. 981–987 (cit. on pp. 21–23).

[70] Mariana Aline Silva Artur, Tao Zhao, Wilco Ligterink, et al. “Dissecting the genomic diversification of late embryogenesis abundant (LEA) protein gene families in plants”. In: *Genome biology and evolution* 11.2 (2019), pp. 459–471 (cit. on p. 21).

[71] Jie Hu, Zhongying Zhao, Hari Krishna Yalamanchili, et al. “Bayesian detection of embryonic gene expression onset in C. elegans”. In: (2015) (cit. on p. 21).

[72] Ivan Croydon Veleslavov and Michael PH Stumpf. “Decision tree models and cell fate choice”. In: *bioRxiv* (2020), pp. 2020–12 (cit. on pp. 21, 59).

[73] Francis Crick. “Central dogma of molecular biology”. In: *Nature* 227.5258 (1970), pp. 561–563 (cit. on p. 21).

[74] Björn Schwanhäusser, Dorothea Busse, Na Li, et al. “Global quantification of mammalian gene expression control”. In: *Nature* 473.7347 (2011), pp. 337–342 (cit. on p. 22).

[75] Mandy G Katz-Jaffe, William B Schoolcraft, and David K Gardner. “Analysis of protein expression (secretome) by human and mouse preimplantation embryos”. In: *Fertility and sterility* 86.3 (2006), pp. 678–685 (cit. on p. 22).

[76] Johan Reimegård, Marcel Tarbier, Marcus Danielsson, et al. “A combined approach for single-cell mRNA and intracellular protein expression analysis”. In: *Communications biology* 4.1 (2021), p. 624 (cit. on p. 22).

[77] Chandra Shekhar Pareek, Rafal Smoczynski, and Andrzej Tretyn. “Sequencing technologies and genome sequencing”. In: *Journal of applied genetics* 52 (2011), pp. 413–435 (cit. on p. 22).

[78] Cong Ma and Carl Kingsford. “Detecting, categorizing, and correcting coverage anomalies of RNA-seq quantification”. In: *Cell systems* 9.6 (2019), pp. 589–599 (cit. on p. 22).

[79] Toshiyuki Shiraki, Shinji Kondo, Shintaro Katayama, et al. “Cap analysis gene expression for high-throughput analysis of transcriptional starting point and identification of promoter usage”. In: *Proceedings of the National Academy of Sciences* 100.26 (2003), pp. 15776–15781 (cit. on p. 22).

[80] Sydney Brenner, Maria Johnson, John Bridgham, et al. “Gene expression analysis by massively parallel signature sequencing (MPSS) on microbead arrays”. In: *Nature biotechnology* 18.6 (2000), pp. 630–634 (cit. on p. 22).

[81] Dominic Grün and Alexander van Oudenaarden. “Design and analysis of single-cell sequencing experiments”. In: *Cell* 163.4 (2015), pp. 799–810 (cit. on pp. 22, 25).

[82] Christoph Ziegenhain, Beate Vieth, Swati Parekh, et al. “Comparative analysis of single-cell RNA sequencing methods”. In: *Molecular cell* 65.4 (2017), pp. 631–643 (cit. on pp. 22, 58).

[83] Byungjin Hwang, Ji Hyun Lee, and Duhee Bang. “Single-cell RNA sequencing technologies and bioinformatics pipelines”. In: *Experimental & molecular medicine* 50.8 (2018), pp. 1–14 (cit. on p. 22).

[84] Daniela F Rodrigues, Vera M Costa, Ricardo Silvestre, et al. “Methods for the analysis of transcriptome dynamics”. In: *Toxicology Research* 8.5 (2019), pp. 597–612 (cit. on p. 22).

[85] Thinh N Tran and Gary D Bader. “Tempora: Cell trajectory inference using time-series single-cell RNA sequencing data”. In: *PLoS computational biology* 16.9 (2020), e1008205 (cit. on p. 22).

[86] Jun Ding, Nadav Sharon, and Ziv Bar-Joseph. “Temporal modelling using single-cell transcriptomics”. In: *Nature Reviews Genetics* 23.6 (2022), pp. 355–368 (cit. on p. 22).

[87] Vivien Marx. “Method of the Year: spatially resolved transcriptomics”. In: *Nature methods* 18.1 (2021), pp. 9–14 (cit. on p. 22).

[88] Jiwoon Park, Junbum Kim, Tyler Lewy, et al. “Spatial omics technologies at multimodal and single cell/subcellular level”. In: *Genome Biology* 23.1 (2022), p. 256 (cit. on p. 22).

[89] Jun Du, Yu-Chen Yang, Zhi-Jie An, et al. “Advances in spatial transcriptomics and related data analysis strategies”. In: *Journal of Translational Medicine* 21.1 (2023), p. 330 (cit. on p. 22).

[90] Zhuxia Li and Guangdun Peng. “Spatial transcriptomics: new dimension of understanding biological complexity”. In: *Biophysics Reports* 8.3 (2022), p. 119 (cit. on p. 22).

[91] Feng Bao, Yue Deng, Sen Wan, et al. “Integrative spatial analysis of cell morphologies and transcriptional states with MUSE”. In: *Nature biotechnology* 40.8 (2022), pp. 1200–1209 (cit. on p. 22).

[92] Yongshuo Zong, Tingyang Yu, Xuesong Wang, et al. “conST: an interpretable multi-modal contrastive learning framework for spatial transcriptomics”. In: *BioRxiv* (2022), pp. 2022–01 (cit. on p. 22).

[93] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, et al. “A comparison of single-cell trajectory inference methods”. In: *Nature biotechnology* (2019) (cit. on pp. 22, 23, 58).

[94] Cole Trapnell, Davide Cacchiarelli, and Xiaojie Qiu. “Monocle: Cell counting, differential expression, and trajectory analysis for single-cell RNA-Seq experiments”. In: (2018) (cit. on pp. 22, 23).

[95] Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, et al. “Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming”. In: *Cell* 176.4 (2019), pp. 928–943 (cit. on pp. 23, 58).

[96] Mor Nitzan, Nikos Karaiskos, Nir Friedman, et al. “Gene expression cartography”. In: *Nature* (2019) (cit. on pp. 23, 24, 58, 94).

[97] Kevin R Moon, Jay S Stanley III, Daniel Burkhardt, et al. “Manifold learning-based methods for analyzing single-cell RNA-sequencing data”. In: *Current Opinion in Systems Biology* 7 (2018), pp. 36–46 (cit. on p. 23).

[98] Ruizhi Xiang, Wencan Wang, Lei Yang, et al. “A comparison for dimensionality reduction methods of single-cell RNA-seq data”. In: *Frontiers in genetics* 12 (2021), p. 646936 (cit. on p. 23).

[99] F Alexander Wolf, Fiona K Hamey, Mireya Plass, et al. “PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells”. In: *Genome biology* 20 (2019), pp. 1–9 (cit. on pp. 23, 55).

[100] Xiaojie Qiu, Qi Mao, Ying Tang, et al. “Reversed graph embedding resolves complex single-cell trajectories”. In: *Nature methods* 14.10 (2017), pp. 979–982 (cit. on p. 23).

[101] Junyue Cao, Malte Spielmann, Xiaojie Qiu, et al. “The single-cell transcriptional landscape of mammalian organogenesis”. In: *Nature* 566.7745 (2019), pp. 496–502 (cit. on p. 23).

[102] Zhicheng Ji and Hongkai Ji. “TSCAN: Pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis”. In: *Nucleic acids research* 44.13 (2016), e117–e117 (cit. on pp. 23, 52).

[103] Chengchen Zhao, Wenchao Xiu, Yuwei Hua, et al. “CStreet: a computed C ell S tate tr ajectory inf ere nce method for t ime-series single-cell RNA sequencing data”. In: *Bioinformatics* 37.21 (2021), pp. 3774–3780 (cit. on p. 23).

[104] Huidong Chen, Luca Albergante, Jonathan Y Hsu, et al. “Single-cell trajectories reconstruction, exploration and mapping of omics data with STREAM”. In: *Nature communications* 10.1 (2019), p. 1903 (cit. on p. 23).

[105] Edroaldo Lummertz da Rocha, R Grant Rowe, Vanessa Lundin, et al. “Reconstruction of complex single-cell trajectories using CellRouter”. In: *Nature communications* 9.1 (2018), p. 892 (cit. on p. 23).

[106] Yang Chen, Yuping Zhang, and Zhengqing Ouyang. "LISA: accurate reconstruction of cell trajectory and pseudo-time for massive single cell RNA-seq data". In: *BIOCOMPUTING 2019: Proceedings of the Pacific Symposium*. World Scientific. 2018, pp. 338–349 (cit. on p. 23).

[107] Yusuke Imoto, Tomonori Nakamura, Emerson G Escolar, et al. "Resolution of the curse of dimensionality in single-cell RNA sequencing data analysis". In: *Life Science Alliance* 5.12 (2022) (cit. on p. 23).

[108] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018) (cit. on p. 23).

[109] Maximillian Nickel and Douwe Kiela. "Poincaré embeddings for learning hierarchical representations". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 23).

[110] Anna Klimovskaia, David Lopez-Paz, Léon Bottou, et al. "Poincaré maps for analyzing complex hierarchies in single-cell data". In: *Nature communications* 11.1 (2020), p. 2966 (cit. on pp. 23, 56, 59).

[111] Cédric Villani et al. *Optimal transport: old and new*. Vol. 338. Springer, 2009 (cit. on p. 23).

[112] Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, et al. "Reconstruction of developmental landscapes by optimal-transport analysis of single-cell gene expression sheds light on cellular reprogramming". In: *BioRxiv* (2017), p. 191056 (cit. on p. 23).

[113] Qian Li. "scTour: a deep learning architecture for robust inference and accurate prediction of cellular dynamics". In: *Genome Biology* 24.1 (2023), pp. 1–33 (cit. on p. 23).

[114] Song Wen, Hao Wang, and Dimitris Metaxas. "Social ode: Multi-agent trajectory forecasting with neural ordinary differential equations". In: *European Conference on Computer Vision*. Springer. 2022, pp. 217–233 (cit. on p. 23).

[115] Kun Wang, Liangzhen Hou, Zhaolian Lu, et al. "Cell division history encodes directional information of fate transitions". In: *bioRxiv* (2022), pp. 2022–10 (cit. on p. 23).

[116] Rahul Satija, Jeffrey A Farrell, David Gennert, et al. "Spatial reconstruction of single-cell gene expression data". In: *Nature biotechnology* 33.5 (2015), pp. 495–502 (cit. on pp. 24, 52).

[117] Tommaso Biancalani, Gabriele Scalia, Lorenzo Buffoni, et al. "Deep learning and alignment of spatially resolved single-cell transcriptomes with Tangram". In: *Nature methods* 18.11 (2021), pp. 1352–1362 (cit. on p. 24).

[118] Mohammad Lotfollahi, Sergei Rybakov, Karin Hrovatin, et al. "Biologically informed deep learning to query gene programs in single-cell atlases". In: *Nature Cell Biology* 25.2 (2023), pp. 337–350 (cit. on p. 24).

[119] Geert-Jan Huizing, Gabriel Peyré, and Laura Cantini. "Optimal transport improves cell–cell similarity inference in single-cell omics data". In: *Bioinformatics* 38.8 (2022), pp. 2169–2177 (cit. on p. 24).

[120] Yutong Sha, Yuchi Qiu, Peijie Zhou, et al. "Reconstructing growth and dynamic trajectories from single-cell transcriptomics data". In: *Nature Machine Intelligence* (Nov. 2023). ISSN: 2522-5839. DOI: 10.1038/s42256-023-00763-w. URL: https://doi.org/10.1038/s42256-023-00763-w (cit. on p. 24).

[121] Cameron G Williams, Hyun Jae Lee, Takahiro Asatsuma, et al. "An introduction to spatial transcriptomics for biomedical research". In: *Genome Medicine* 14.1 (2022), pp. 1–18 (cit. on p. 24).

[122] PL Stahl, F Salmen, S Vickovic, et al. "Visualization and analysis of gene expression in tissue sections by spatial transcriptomics., 2016, 353". In: *DOI: https://doi. org/10.1126/science. aaf2403* (), pp. 78–82 (cit. on p. 24).

[123] Xuehua Ma, Zhiguang Zhao, Long Xiao, et al. "A 4D single-cell protein atlas of transcription factors delineates spatiotemporal patterning during embryogenesis". In: *Nature Methods* 18.8 (2021), pp. 893–902 (cit. on p. 24).

[124] Zhicheng Xu, Weiwen Wang, Tao Yang, et al. "STOmicsDB: a comprehensive database for spatial transcriptomics data sharing, analysis and visualization". In: *Nucleic Acids Research* 52.D1 (2024), pp. D1053–D1061 (cit. on p. 24).

[125] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, et al. "Deep clustering by gaussian mixture variational autoencoders with graph embedding". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6440–6449 (cit. on pp. 24, 117).

[126] Karren Dai Yang, Anastasiya Belyaeva, Saradha Venkatachalapathy, et al. "Multi-domain translation between single-cell imaging and sequencing data using autoencoders". In: *Nature communications* 12.1 (2021), p. 31 (cit. on pp. 24, 117).

[127] Ran Zhang, Laetitia Meng-Papaxanthos, Jean-philippe Vert, et al. "Multimodal Single-Cell Translation and Alignment with Semi-Supervised Learning". In: *Journal of Computational Biology* 29.11 (2022), pp. 1198–1212 (cit. on p. 24).

[128] Rohan Gala, Nathan Gouwens, Zizhen Yao, et al. "A coupled autoencoder approach for multi-modal analysis of cell types". In: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on p. 24).

[129] Noah Cohen Kalafut, Xiang Huang, and Daifeng Wang. "JAMIE: Joint Variational Autoencoders for Multi-Modal Imputation and Embedding". In: (2022) (cit. on pp. 24, 117).

[130] Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, et al. "Integrated analysis of multimodal single-cell data". In: *Cell* 184.13 (2021), pp. 3573–3587 (cit. on pp. 24, 117).

[131] Zhi-Hua Du, Wei-Lin Hu, Jian-Qiang Li, et al. "scPML: pathway-based multi-view learning for cell type annotation from single-cell RNA-seq data". In: *Communications Biology* 6.1 (2023), p. 1268 (cit. on p. 24).

[132] Yuhan Hao, Tim Stuart, Madeline H Kowalski, et al. "Dictionary learning for integrative, multimodal and scalable single-cell analysis". In: *Nature biotechnology* (2023), pp. 1–12 (cit. on pp. 24, 117).

[133] Tim Stuart, Andrew Butler, Paul Hoffman, et al. "Comprehensive integration of single-cell data". In: *Cell* 177.7 (2019), pp. 1888–1902 (cit. on pp. 24, 117).

[134] Dragomirka Jovic, Xue Liang, Hua Zeng, et al. "Single-cell RNA sequencing technologies and applications: A brief overview". In: *Clinical and Translational Medicine* 12.3 (2022), e694 (cit. on p. 25).

[135] Todd W Harris, Igor Antoshechkin, Tamberlyn Bieri, et al. "WormBase: a comprehensive resource for nematode research". In: *Nucleic acids research* 38.suppl_-1 (2010), pp. D463–D467 (cit. on pp. 25, 55).

[136] Simon Mages, Noa Moriel, Inbal Avraham-Davidi, et al. "TACCO unifies annotation transfer and decomposition of cell identities for single-cell and spatial omics". In: *Nature biotechnology* 41.10 (2023), pp. 1465–1473 (cit. on p. 25).

[137] Musu Yuan, Liang Chen, and Minghua Deng. "scMRA: a robust deep learning method to annotate scRNA-seq data with multiple reference datasets". In: *Bioinformatics* 38.3 (2022), pp. 738–745 (cit. on p. 25).

[138] Liang Chen, Qiuyan He, Yuyao Zhai, et al. "Single-cell RNA-seq data semi-supervised clustering and annotation via structural regularized domain adaptation". In: *Bioinformatics* 37.6 (2021), pp. 775–784 (cit. on p. 25).

[139] Yuqi Tan and Patrick Cahan. "SingleCellNet: a computational tool to classify single cell RNA-Seq data across platforms and across species". In: *Cell systems* 9.2 (2019), pp. 207–213 (cit. on p. 25).

[140] Yan Zhou, Minjiao Peng, Bin Yang, et al. "scDLC: a deep learning framework to classify large sample single-cell RNA-seq data". In: *BMC Genomics* 23.1 (July 2022), p. 504. ISSN: 1471-2164. DOI: 10.1186/s12864-022-08715-1. URL: https://doi.org/10.1186/s12864-022-08715-1 (cit. on p. 25).

[141] Tamim Abdelaal, Lieke Michielsen, Davy Cats, et al. "A comparison of automatic cell identification methods for single-cell RNA sequencing data". In: *Genome biology* 20 (2019), pp. 1–19 (cit. on p. 26).

[142] Huy Le, Beverly Peng, Janelle Uy, et al. "Machine learning for cell type classification from single nucleus RNA sequencing data". In: *Plos one* 17.9 (2022), e0275070 (cit. on p. 26).

[143] Ren Qi, Anjun Ma, Qin Ma, et al. "Clustering and classification methods for single-cell RNA-sequencing data". In: *Briefings in bioinformatics* 21.4 (2020), pp. 1196–1208 (cit. on p. 26).

[144] Hongkui Zeng. "What is a cell type and how to define it?" In: *Cell* 185.15 (2022), pp. 2739–2755 (cit. on pp. 26, 115).

[145] Jialin Liu, Chao Gao, Joshua Sodicoff, et al. "Jointly defining cell types from multiple single-cell datasets using LIGER". In: *Nature protocols* 15.11 (2020), pp. 3632–3662 (cit. on p. 26).

[146] Jingjing Wang, Huiyu Sun, Mengmeng Jiang, et al. "Tracing cell-type evolution by cross-species comparison of cell atlases". In: *Cell Reports* 34.9 (2021) (cit. on p. 26).

[147] Zhijin Wu and Hao Wu. "Accounting for cell type hierarchy in evaluating single cell RNA-seq clustering". In: *Genome biology* 21.1 (2020), p. 123 (cit. on p. 26).

[148] Yasin Kaymaz, Florian Ganglberger, Ming Tang, et al. "HieRFIT: a hierarchical cell type classification tool for projections from complex single-cell atlas datasets". In: *Bioinformatics* 37.23 (2021), pp. 4431–4436 (cit. on p. 26).

[149] Raphael Petegrosso, Zhuliu Li, and Rui Kuang. "Machine learning and statistical methods for clustering single-cell RNA-sequencing data". In: *Briefings in bioinformatics* 21.4 (2020), pp. 1209–1223 (cit. on pp. 26, 46).

[150] Packer et al. "A lineage-resolved molecular atlas of C. elegans embryogenesis at single-cell resolution". In: *Science* (2019) (cit. on pp. 27, 28, 55, 58–60, 72, 75).

[151] Vincent Bertrand and Oliver Hobert. "Linking asymmetric cell division to the terminal differentiation program of postmitotic neurons in C. elegans". In: *Developmental cell* 16.4 (2009), pp. 563–575 (cit. on p. 28).

[152] Vladimir Yu Kiselev, Tallulah S Andrews, and Martin Hemberg. "Challenges in unsupervised clustering of single-cell RNA-seq data". In: *Nature Reviews Genetics* 20.5 (2019), pp. 273–282 (cit. on p. 28).

[153] Christoph Hafemeister and Rahul Satija. "Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression". In: *Genome biology* 20.1 (2019), p. 296 (cit. on p. 30).

[154] Tulika Kakati, Dhruba K Bhattacharyya, Jugal K Kalita, et al. "DEGnext: classification of differentially expressed genes from RNA-seq data using a convolutional neural network with transfer learning". In: *BMC bioinformatics* 23.1 (2022), p. 17 (cit. on p. 30).

[155] Tongxin Wang, Wei Shao, Zhi Huang, et al. "MOGONET integrates multi-omics data using graph convolutional networks allowing patient classification and biomarker identification". In: *Nature Communications* 12.1 (2021), p. 3445 (cit. on p. 30).

[156] Zhen Li, Xiaoyang Chen, Xuegong Zhang, et al. "Latent feature extraction with a prior-based self-attention framework for spatial transcriptomics". In: *Genome Research* 33.10 (2023), pp. 1757–1773 (cit. on p. 30).

[157] Lei Feng, Jiaqi Lv, Bo Han, et al. "Provably consistent partial-label learning". In: *Advances in neural information processing systems* 33 (2020), pp. 10948–10960 (cit. on p. 30).

[158] Etienne Côme, Latifa Oukhellou, Thierry Denoeux, et al. "Learning from partially supervised data using mixture models and belief functions". In: *Pattern recognition* 42.3 (2009), pp. 334–348 (cit. on p. 30).

[159] Yingjie Tian, Xiaotong Yu, and Saiji Fu. "Partial label learning: Taxonomy, analysis and outlook". In: *Neural Networks* 161 (2023), pp. 708–734. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2023.02.019. URL: https://www.sciencedirect.com/science/article/pii/S0893608023000825 (cit. on p. 30).

[160] Xiaojin Jerry Zhu. "Semi-supervised learning literature survey". In: (2005) (cit. on pp. 31, 96).

[161] Colin Troisemaine, Vincent Lemaire, Stéphane Gosselin, et al. "Novel Class Discovery: an Introduction and Key Concepts". In: *arXiv preprint arXiv:2302.12028* (2023) (cit. on pp. 31, 32, 96).

[162] Richard Socher, Milind Ganjoo, Christopher D Manning, et al. "Zero-shot learning through cross-modal transfer". In: *Advances in neural information processing systems* 26 (2013) (cit. on pp. 31, 98).

[163] Dongshunyi Li, Jun Ding, and Ziv Bar-Joseph. "Unsupervised cell functional annotation for single-cell RNA-seq". In: *Genome Research* 32.9 (2022), pp. 1765–1775 (cit. on p. 32).

[164] Malek Senoussi, Thierry Artieres, and Paul Villoutreix. "Partial label learning for automated classification of single-cell transcriptomic profiles". In: *PLOS Computational Biology* 20.4 (2024), e1012006 (cit. on pp. 33, 58).

[165] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009 (cit. on p. 34).

[166] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018 (cit. on pp. 34, 45, 46).

[167] Chloé-Agathe Azencott. *Introduction au Machine Learning-2e éd.* Dunod, 2022 (cit. on p. 35).

[168] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006 (cit. on pp. 35, 47).

[169] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32 (cit. on pp. 36, 60, 66).

[170] Tianqi Chen, Tong He, Michael Benesty, et al. "Xgboost: extreme gradient boosting". In: *R package version 0.4-2* 1.4 (2015), pp. 1–4 (cit. on p. 37).

[171] Guolin Ke, Qi Meng, Thomas Finley, et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 37).

[172] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152 (cit. on p. 37).

[173] Koby Crammer and Yoram Singer. "On the algorithmic implementation of multiclass kernel-based vector machines". In: *Journal of machine learning research* 2.Dec (2001), pp. 265–292 (cit. on p. 38).

[174] Léon Bottou and Chih-Jen Lin. "Support vector machine solvers". In: (2007) (cit. on p. 40).

[175] Qiang Wu and Ding-Xuan Zhou. "SVM soft margin classifiers: linear programming versus quadratic programming". In: *Neural computation* 17.5 (2005), pp. 1160–1187 (cit. on p. 40).

[176] Christopher Williams and Matthias Seeger. "Using the Nyström Method to Speed Up Kernel Machines". In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. URL: https://proceedings.neurips.cc/paper_files/paper/2000/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf (cit. on p. 41).

[177] Vladimir N Vapnik. "An overview of statistical learning theory". In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999 (cit. on pp. 41, 66).

[178] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. "A primer on kernel methods". In: *Kernel methods in computational biology* 47 (2004), pp. 35–70 (cit. on p. 41).

[179] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems* 20 (2007) (cit. on pp. 41, 66, 91).

[180] Andreas Müller. "Kernel Approximations for Efficient SVMs (and other feature extraction methods)". In: *Data Science & Machine Learning Resources* (2012) (cit. on p. 41).

[181] Barbara Hammer and Thomas Villmann. "Generalized relevance learning vector quantization". In: *Neural Networks* 15.8-9 (2002), pp. 1059–1068 (cit. on p. 42).

[182] Michael A Arbib. *The handbook of brain theory and neural networks*. MIT press, 2003 (cit. on p. 42).

[183] Paassen B. "Large margin learning vector quantization". In: *CITEC, Bielefeld University, MiWoCI*. 2019 (cit. on p. 42).

[184] Chapelle Weinberger. "Large Margin Taxonomy Embedding with an Application to Document Categorization". In: *NeurIPS* (2008) (cit. on pp. 43, 46, 60, 62, 66, 69).

[185] Carlos N Silla and Alex A Freitas. "A survey of hierarchical classification across different application domains". In: *Data mining and knowledge discovery* 22 (2011), pp. 31–72 (cit. on pp. 43, 44, 117).

[186] Samy Bengio, Jason Weston, and David Grangier. "Label Embedding Trees for Large Multi-Class Tasks". In: NIPS, 2010. URL: https://proceedings.neurips.cc/paper/2010/hash/06138bc5af6023646ede0e1f7c1eac75-Abstract.html (cit. on pp. 43, 59, 62).

[187] Ioannis Partalas, Massih-Reza Amini, Ion Androutsopoulos, et al. "Web-scale classification: web classification in the big data era". In: ed. by Ben Carterette, Fernando Diaz, Carlos Castillo, et al. ACM, 2014. DOI: 10.1145/2556195.2556208. URL: https://doi.org/10.1145/2556195.2556208 (cit. on pp. 43, 59).

[188] Zhongzhe Xiao, Emmanuel Dellandrea, Weibei Dou, et al. "Hierarchical classification of emotional speech". In: *IEEE Transactions on Multimedia* 37 (2007) (cit. on p. 43).

[189] Michael Ashburner, Catherine A Ball, Judith A Blake, et al. "Gene ontology: tool for the unification of biology". In: *Nature genetics* 25.1 (2000), pp. 25–29 (cit. on p. 43).

[190] Paul N. Bennett and Nam Nguyen. "Refined experts: improving classification in large taxonomies". In: ACM, 2009. DOI: 10.1145/1571941.1571946. URL: https://doi.org/10.1145/1571941.1571946 (cit. on pp. 43, 59).

[191] Siddharth Gopal and Yiming Yang. "Hierarchical Bayesian Inference and Recursive Regularization for Large-Scale Classification". In: *ACM* (2015). DOI: 10.1145/2629585. URL: https://doi.org/10.1145/2629585 (cit. on pp. 43, 59).

[192] Bhatia et al. *The extreme classification repository: Multi-label datasets and code*. 2016. URL: http://manikvarma.org/downloads/XC/XMLRepository.html (cit. on pp. 43, 59, 62).

[193] Alex Freitas and André Carvalho. "A tutorial on hierarchical classification with applications in bioinformatics". In: *Research and trends in data mining technologies and applications* (2007), pp. 175–208 (cit. on p. 43).

[194] Sofi Defiyanti, Edi Winarko, and Sigit Priyanta. "A survey of hierarchical classification algorithms with big-bang approach". In: *2019 5th International Conference on Science and Technology (ICST)*. Vol. 1. IEEE. 2019, pp. 1–6 (cit. on p. 44).

[195] Xiao-Lin Wang and Bao-Liang Lu. "Improved hierarchical svms for large-scale hierarchical text classification challenge". In: *Large scale hierarchical text classification challenge short paper* (2009) (cit. on p. 45).

[196] Denny Zhou, Lin Xiao, and Mingrui Wu. "Hierarchical classification via orthogonal transfer". In: (2011) (cit. on p. 45).

[197] Lijuan Cai and Thomas Hofmann. "Hierarchical Document Categorization with Support Vector Machines". In: ACM, 2004. URL: http://doi.acm.org/10.1145/1031171.1031186 (cit. on pp. 45, 60, 62).

[198] Angelo Duò, Mark D Robinson, and Charlotte Soneson. "A systematic performance evaluation of clustering methods for single-cell RNA-seq data". In: *F1000Research* 7 (2018) (cit. on p. 46).

[199] Lihong Peng, Xiongfei Tian, Geng Tian, et al. "Single-cell RNA-seq clustering: datasets, models, and algorithms". In: *RNA biology* 17.6 (2020), pp. 765–783 (cit. on p. 46).

[200] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137 (cit. on p. 47).

[201] David Arthur and Sergei Vassilvitskii. "K-means++ the advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 2007, pp. 1027–1035 (cit. on p. 48).

[202] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, et al. "SC3: consensus clustering of single-cell RNA-seq data". In: *Nature methods* 14.5 (2017), pp. 483–486 (cit. on p. 48).

[203] Eugenio Marco, Robert L Karp, Guoji Guo, et al. "Bifurcation analysis of single-cell gene expression data reveals epigenetic landscape". In: *Proceedings of the National Academy of Sciences* 111.52 (2014), E5643–E5650 (cit. on p. 48).

[204] Lu Yang, Jiancheng Liu, Qiang Lu, et al. "SAIC: an iterative clustering approach for analysis of single cell RNA-seq data". In: *BMC genomics* 18 (2017), pp. 9–17 (cit. on p. 48).

[205] Justina Žurauskienė and Christopher Yau. "pcaReduce: hierarchical clustering of single cell transcriptional profiles". In: *BMC bioinformatics* 17 (2016), pp. 1–11 (cit. on p. 48).

[206] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society: series B (methodological)* 39.1 (1977), pp. 1–22 (cit. on p. 49).

[207] Bryan W Brown and Whitney K Newey. "Generalized method of moments, efficient bootstrapping, and improved inference". In: *Journal of Business & Economic Statistics* 20.4 (2002), pp. 507–517 (cit. on p. 51).

[208] Sandhya Prabhakaran, Elham Azizi, Ambrose Carr, et al. "Dirichlet process mixture model for correcting technical variation in single-cell gene expression data". In: *International conference on machine learning*. PMLR. 2016, pp. 1070–1079 (cit. on p. 52).

[209] Nikolaos Papadopoulos, Parra R Gonzalo, and Johannes Söding. "PROSSTT: probabilistic simulation of single-cell RNA-seq data for complex differentiation processes". In: *Bioinformatics* 35.18 (2019), pp. 3517–3519 (cit. on pp. 53, 54, 72, 100, 102, 105).

[210] Mireya Plass, Jordi Solana, F Alexander Wolf, et al. "Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics". In: *Science* 360.6391 (2018), eaaq1723 (cit. on pp. 55, 72, 75, 105).

[211] Franziska Paul, Ya'ara Arkin, Amir Giladi, et al. "Transcriptional heterogeneity and lineage commitment in myeloid progenitors". In: *Cell* 163.7 (2015), pp. 1663–1677 (cit. on pp. 56, 72, 75, 105).

[212] El-ad David Amir, Kara L Davis, Michelle D Tadmor, et al. "viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia". In: *Nature biotechnology* 31.6 (2013), pp. 545–552 (cit. on pp. 58, 93).

[213] Becht et al. "Dimensionality reduction for visualizing single-cell data using UMAP". In: *Nature biotechnology* (2019) (cit. on pp. 58, 76).

[214] Moon et al. "Visualizing structure and transitions in high-dimensional biological data". In: *Nature biotechnology* (2019) (cit. on p. 58).

[215] Robert G Kelly. "The heart field transcriptional landscape at single-cell resolution". In: *Developmental Cell* (2023) (cit. on p. 58).

[216] Dominik Klein, Giovanni Palla, Marius Lange, et al. "Mapping cells through time and space with moscot". In: *bioRxiv* (2023), pp. 2023–05 (cit. on p. 58).

[217] Timothee Cour, Ben Sapp, and Ben Taskar. "Learning from partial labels". In: *The Journal of Machine Learning Research* 12 (2011), pp. 1501–1536 (cit. on p. 59).

[218] Lieke Michielsen, Mohammad Lotfollahi, Daniel Strobl, et al. "Single-cell reference mapping to construct and extend cell-type hierarchies". In: *bioRxiv* (2022), pp. 2022–07 (cit. on p. 59).

[219] Tanja Stadler, Oliver G Pybus, and Michael PH Stumpf. "Phylodynamics for cell biologists". In: *Science* 371.6526 (2021), eaah6266 (cit. on pp. 59, 94).

[220] Eyke Hüllermeier and Jürgen Beringer. "Learning from ambiguously labeled examples". In: *Intelligent Data Analysis* 10.5 (2006), pp. 419–439 (cit. on pp. 59, 71).

[221] Zinan Zeng, Shijie Xiao, Kui Jia, et al. "Learning by associating ambiguously labeled images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 708–715 (cit. on p. 59).

[222] Jie Luo and Francesco Orabona. "Learning from candidate labeling sets". In: *Advances in neural information processing systems* 23 (2010) (cit. on p. 59).

[223] Yi-Chen Chen, Vishal M Patel, Rama Chellappa, et al. "Ambiguously labeled learning using dictionaries". In: *IEEE Transactions on Information Forensics and Security* 9.12 (2014), pp. 2076–2088 (cit. on p. 59).

[224] Nam Nguyen and Rich Caruana. "Classification with partial labels". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 551–559 (cit. on pp. 59, 68).

[225] Fei Yu and Min-Ling Zhang. "Maximum Margin Partial Label Learning". In: *Asian Conference on Machine Learning*. Ed. by Geoffrey Holmes and Tie-Yan Liu. Vol. 45. Proceedings of Machine Learning Research. Hong Kong: PMLR, 20–22 Nov 2016, pp. 96–111. URL: https://proceedings.mlr.press/v45/Yu15.html (cit. on pp. 59, 63).

[226] Yu Meng, Jiaming Shen, Chao Zhang, et al. "Weakly-Supervised Hierarchical Text Classification". In: 2019. DOI: 10.1609/aaai.v33i01.33016826. URL: https://doi.org/10.1609/aaai.v33i01.33016826 (cit. on p. 59).

[227] Yu Zhang, Xiusi Chen, Yu Meng, et al. "Hierarchical Metadata-Aware Document Categorization under Weak Supervision". In: ACM, 2021. DOI: 10.1145/3437963.3441730. URL: https://doi.org/10.1145/3437963.3441730 (cit. on p. 59).

[228] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, et al. "Large Margin Methods for Structured and Interdependent Output Variables". In: *J. Mach. Learn. Res.* 6 (2005), pp. 1453–1484. URL: http://jmlr.org/papers/v6/tsochantaridis05a.html (cit. on pp. 60, 65, 68).

[229] Scott Menard. *Applied logistic regression analysis*. 106. Sage, 2002 (cit. on p. 60).

[230] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232 (cit. on p. 60).

[231] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794 (cit. on pp. 60, 66).

[232] Moustapha Cissé, Thierry Artières, and Patrick Gallinari. "Learning Compact Class Codes for Fast Inference in Large Multi Class Classification". In: ECML, 2012. DOI: 10.1007/978-3-642-33460-3_38. URL: https://doi.org/10.1007/978-3-642-33460-3%5C_38 (cit. on p. 60).

[233] Chun-Nam John Yu and Thorsten Joachims. "Learning structural SVMs with latent variables". In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*. Ed. by Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman. Vol. 382. ACM International Conference Proceeding Series. ACM, 2009, pp. 1169–1176. DOI: 10.1145/1553374.1553523. URL: https://doi.org/10.1145/1553374.1553523 (cit. on p. 63).

[234] Celine Vens, Jan Struyf, Leander Schietgat, et al. "Decision trees for hierarchical multi-label classification". In: *Machine learning* 73 (2008), pp. 185–214 (cit. on p. 65).

[235] Charalampos Chelmis and Wenting Qi. "Hierarchical MultiClass AdaBoost". In: *2021 IEEE international conference on big data (big data)*. IEEE. 2021, pp. 5063–5070 (cit. on p. 65).

[236] Christopher Williams and Matthias Seeger. "Using the Nyström Method to Speed Up Kernel Machines". In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. URL: https://proceedings.neurips.cc/paper_files/paper/2000/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf (cit. on p. 66).

[237] Adam Paszke, Sam Gross, Francisco Massa, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 66).

[238] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on pp. 66, 67, 69, 108).

[239] Ali Rahimi and Benjamin Recht. "Random Features for Large-Scale Kernel Machines". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt, D. Koller, Y. Singer, et al. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf (cit. on p. 69).

[240] Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005 (cit. on p. 69).

[241] Ebony Rose Watson, Ariane Mora, Atefeh Taherian Fard, et al. "How does the structure of data impact cell–cell similarity? Evaluating how structural properties influence the performance of proximity metrics in single cell RNA-seq data". In: *Briefings in Bioinformatics* 23.6 (Sept. 2022), bbac387. ISSN: 1477-4054. DOI: 10.1093/bib/bbac387. eprint: https://academic.oup.com/bib/article-pdf/23/6/bbac387/47143790/bbac387.pdf. URL: https://doi.org/10.1093/bib/bbac387 (cit. on p. 71).

[242] Belur V. Dasarathy. "Nearest neighbor (NN) norms: NN pattern classification techniques". In: 1991 (cit. on p. 71).

[243] Darren J Burgess. "Spatial transcriptomics coming of age". In: *Nature Reviews Genetics* 20.6 (2019), pp. 317–317 (cit. on p. 94).

[244] Katy Vandereyken, Alejandro Sifrim, Bernard Thienpont, et al. "Methods and applications for single-cell and spatial multi-omics". In: *Nature Reviews Genetics* (2023), pp. 1–22 (cit. on p. 94).

[245] Aviv Regev, Sarah A Teichmann, Eric S Lander, et al. "The human cell atlas". In: *elife* 6 (2017), e27041 (cit. on p. 94).

[246] Chufeng Tan, Xing Xu, and Fumin Shen. "A survey of zero shot detection: methods and applications". In: *Cognitive Robotics* 1 (2021), pp. 159–167 (cit. on p. 96).

[247] Olivier Chapelle and Alexander Zien. "Semi-supervised classification by low density separation". In: *International workshop on artificial intelligence and statistics*. PMLR. 2005, pp. 57–64 (cit. on p. 97).

[248] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 542–542 (cit. on pp. 97, 98).

[249] Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, et al. "Self-training: A survey". In: *arXiv preprint arXiv:2202.12040* (2022) (cit. on p. 97).

[250] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, et al. "S4L: Self-Supervised Semi-Supervised Learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Nov. 2019 (cit. on p. 97).

[251] S Fralick. "Learning to recognize patterns without a teacher". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 57–64 (cit. on p. 97).

[252] Isaac Triguero, Salvador Garcia, and Francisco Herrera. "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study". In: *Knowledge and Information systems* 42 (2015), pp. 245–284 (cit. on p. 97).

[253] Dong-Hyun Lee et al. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". In: *Workshop on challenges in representation learning, ICML*. Vol. 3. 2. Atlanta. 2013, p. 896 (cit. on p. 97).

[254] Yves Grandvalet and Yoshua Bengio. "Semi-supervised learning by entropy minimization". In: *Advances in neural information processing systems* 17 (2004) (cit. on p. 97).

[255] Avital Oliver, Augustus Odena, Colin A Raffel, et al. "Realistic evaluation of deep semi-supervised learning algorithms". In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 97).

[256] David J Miller and Hasan Uyar. "A mixture of experts classifier with learning based on both labelled and unlabelled data". In: *Advances in neural information processing systems* 9 (1996) (cit. on p. 98).

[257] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, et al. "Text classification from labeled and unlabeled documents using EM". In: *Machine learning* 39 (2000), pp. 103–134 (cit. on p. 98).

[258] Xiangli Yang, Zixing Song, Irwin King, et al. "A survey on deep semi-supervised learning". In: *IEEE Transactions on Knowledge and Data Engineering* 35.9 (2022), pp. 8934–8954 (cit. on p. 98).

[259] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015) (cit. on p. 98).

[260] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016) (cit. on p. 98).

[261] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, et al. "Semi-supervised learning with deep generative models". In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 98).

[262] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 98).

[263] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 98).

[264] Yang Li, Quan Pan, Suhang Wang, et al. "Disentangled variational auto-encoder for semi-supervised learning". In: *Information Sciences* 482 (2019), pp. 73–85 (cit. on p. 98).

[265] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, et al. "Auxiliary deep generative models". In: *International conference on machine learning*. PMLR. 2016, pp. 1445–1453 (cit. on p. 98).

[266] Tom Joy, Sebastian M Schmon, Philip HS Torr, et al. "Rethinking semi-supervised learning in VAEs". In: *arXiv preprint arXiv:2006.10102* (2020) (cit. on p. 98).

[267] Ruifeng Zhu, Fadi Dornaika, and Yassine Ruichek. "Inductive semi-supervised learning with graph convolution based regression". In: *Neurocomputing* 434 (2021), pp. 315–322 (cit. on p. 98).

[268] Hanxuan Yang, Zhaoxin Yu, Qingchao Kong, et al. "Variational Graph Auto-Encoder Based Inductive Learning Method for Semi-Supervised Classification". In: *arXiv preprint arXiv:2403.17500* (2024) (cit. on p. 98).

[269] Zixing Song, Xiangli Yang, Zenglin Xu, et al. "Graph-based semi-supervised learning: A comprehensive review". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022) (cit. on p. 98).

[270] Zachary Novack, Julian McAuley, Zachary Chase Lipton, et al. "Chils: Zero-shot image classification with hierarchical label sets". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 26342–26362 (cit. on p. 98).

[271] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, et al. "Generalized zero-and few-shot learning via aligned variational autoencoders". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8247–8255 (cit. on p. 98).

[272] Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. "Learning to cluster in order to transfer across domains and tasks". In: *arXiv preprint arXiv:1711.10125* (2017) (cit. on p. 98).

[273] Kai Han, Andrea Vedaldi, and Andrew Zisserman. "Learning to discover novel visual categories via deep transfer clustering". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8401–8409 (cit. on p. 98).

[274] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, et al. "AutoNovel: Automatically Discovering and Learning Novel Visual Categories". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2021) (cit. on pp. 98, 108).

[275] Bingchen Zhao and Kai Han. "Novel visual category discovery with dual ranking statistics and mutual knowledge distillation". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22982–22994 (cit. on p. 98).

[276] Ronald L Graham and Pavol Hell. "On the history of the minimum spanning tree problem". In: *Annals of the History of Computing* 7.1 (1985), pp. 43–57 (cit. on p. 101).

[277] Xiaofei He, Deng Cai, Yuanlong Shao, et al. "Laplacian regularized Gaussian mixture model for data clustering". In: *IEEE transactions on knowledge and data engineering* 23.9 (2010), pp. 1406–1418 (cit. on p. 105).

[278] Alexander Gepperth and Benedikt Pfülb. "Gradient-based training of gaussian mixture models for high-dimensional streaming data". In: *Neural Processing Letters* 53.6 (2021), pp. 4331–4348 (cit. on p. 105).

[279] Sagar Vaze, Kai Han, Andrea Vedaldi, et al. "Generalized category discovery". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7492–7501 (cit. on p. 107).

[280] Isaac Virshup, Danila Bredikhin, Lukas Heumos, et al. "The scverse project provides a computational ecosystem for single-cell omics data analysis". In: *Nature biotechnology* (2023), pp. 1–3 (cit. on p. 114).

[281] Lijia Yu, Yue Cao, Jean YH Yang, et al. "Benchmarking clustering algorithms on estimating the number of cell types from single-cell RNA-sequencing data". In: *Genome biology* 23.1 (2022), p. 49 (cit. on p. 115).

[282] Christopher A Jackson and Christine Vogel. "New horizons in the stormy sea of multimodal single-cell data integration". In: *Molecular cell* 82.2 (2022), pp. 248–259 (cit. on p. 117).

[283] Mika Sarkin Jain, Krzysztof Polanski, Cecilia Dominguez Conde, et al. "MultiMAP: dimensionality reduction and integration of multimodal data". In: *Genome biology* 22 (2021), pp. 1–26 (cit. on p. 117).

[284] Ricard Argelaguet, Anna SE Cuomo, Oliver Stegle, et al. "Computational principles and challenges in single-cell data integration". In: *Nature biotechnology* 39.10 (2021), pp. 1202–1215 (cit. on p. 117).