Classification hiérarchique pour des données transcriptomiques faiblement supervisées

Malek Senoussi, Thierry Artieres, Paul Villoutreix

Classification hiérarchique pour des données transcriptomiques faiblement supervisées

Malek Senoussi[1], Thierry Artieres[1,2], et Paul Villoutreix[1]

[1]Aix Marseille Univ, Université de Toulon, CNRS, LIS, Turing Centre for Living Systems, Marseille, France
[2]Ecole Centrale de Marseille, Marseille, France

31 mai 2022

Résumé

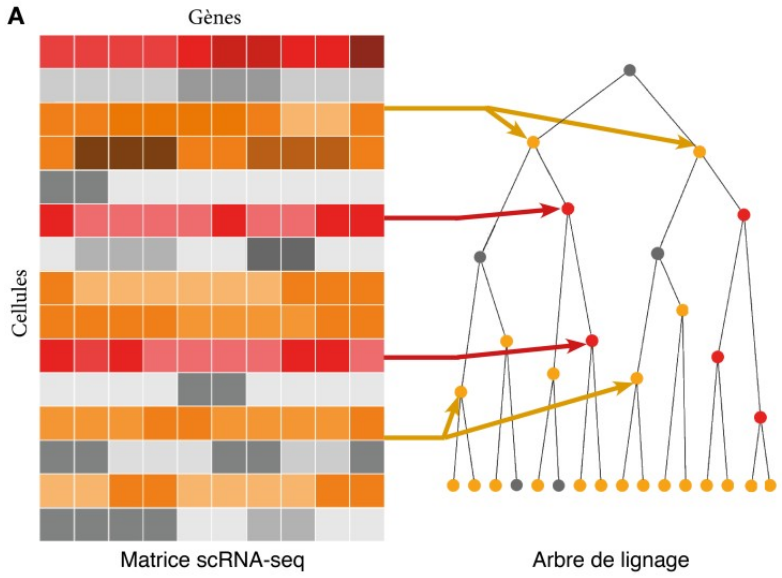
Nous présentons une approche de classification hiérarchique développée pour des données de séquençage ARN faiblement supervisées dans le cadre de la biologie du développement. L'objectif est d'apprendre à classer des données vectorielles, représentant des cellules, structurées dans un arbre traduisant des liens de parenté directe entre deux cellules. Dans ce contexte, du fait de la difficulté d'obtenir des informations de supervision, seule une petite partie des données disponibles est étiquetée et une autre partie des données, éventuellement grande, est supervisée mais avec ambiguïté, c'est à dire qu'une donnée est étiquetée par un ensemble de labels candidats. Nous proposons une méthode de classification hiérarchique adaptée à ce problème en s'inspirant d'une méthode de l'état de l'art en classification hiérarchique supervisée et en proposant une version inspirée de la prédiction structurée avec variables latentes. Il s'agit d'une tâche nouvelle pour laquelle nous ne connaissons pas d'autres approches aujourd'hui en biologie, nous explorons donc le comportement de notre approche et comparons les performances de plusieurs variantes que nous déclinons sur des jeux de données construits à partir d'un jeu de données de référence en biologie pour refléter diverses situations en termes de distribution de données étiquetées ou partiellement étiquetées.

Mots-clef : Classification hiérarchique, Supervision faible, sc-RNAseq.

1 Introduction

L'objectif de la biologie du développement est de comprendre comment une cellule unique se transforme en un organisme multicellulaire. Ce processus complexe implique des dynamiques à l'échelle moléculaire, notamment à travers l'expression génétique, à l'échelle des cellules et à l'échelle des tissus. Pour mesurer cette diversité de processus, il existe plusieurs méthodes d'acquisition complémentaires [Vil21]. D'un côté, les techniques de microscopie vont donner accès aux aspects spatio-temporels des cellules. De l'autre les techniques de séquençage, en particulier de l'ARN, vont donner accès aux aspects moléculaires. Etant donné la taille de ces jeux de données, il est nécessaire de développer des approches automatisées pour leur intégration et leur traitement. Ici, nous nous intéressons plus spécifiquement aux données issues du séquençage de l'ARN à l'échelle de la cellule unique (scRNASeq). Et en particulier nous nous demandons s'il est possible de retrouver les relations de généalogie entre cellules au sein de ce jeu de données à partir du contenu de chaque vecteur transcriptomique (représentant une quantité d'ARN associée à chacun des gènes pour une cellule donnée) et un certain nombre d'annotations manuelles.

La tâche qui nous intéresse peut être formalisée comme un problème de classification hiérarchique dans laquelle on cherche à classer les cellules, des vecteurs transcriptomiques, dans les noeuds d'un arbre de labels dit arbre de lignage qui représente le processus de génèse d'un embryon, partant d'une cellule puis qui, par divisions successives, aboutit à un embryon (voir Fig. (1)). La difficulté vient d'une part du nombre relativement limité de données étiquetées mais également



B

	B_L	B_{WL}	B_U	Total
Nombre	**5143**	**42631**	**41927**	**89701**
%	6	47	47	100
Cardinal de l'ensemble des labels	**95**	**970**	**287**	**1342**
%	7	72	21	100

FIGURE 1 – A) Les données que nous considérons (Matrice scRNASeq à gauche, où chaque ligne représente un vecteur transcriptomique) sont annotées de manière unique (en rouge) ou ambiguë (en orange, c'est à dire que plusieurs étiquettes sont possibles) ou bien sans annotations (en gris). Ces étiquettes sont structurées sous la forme d'un arbre (Arbre de lignage à droite). L'objectif est d'étiqueter toutes les données de manière unique. B) Le jeu de données de référence [P+19] que nous utilisons contient des données étiquetées de manière unique (colonne B_L en rouge), ambiguë (colonne B_{WL} en orange) et des données sans étiquette (colonne B_U en gris).

de la présence de données étiquetées avec ambiguité, c'est à dire que l'étiquetage consiste en un ensemble de noeuds dans lequel se trouve le vrai label de la cellule. Ce problème n'a été que peu étudié dans la littérature sur la classification hiérarchique et nous proposons une approche qui est une extension d'une méthode de l'état de l'art de classification hiérarchique supervisée comme un problème de prédiction structurée à variables latentes.

Nous présentons tout d'abord plus en détail le contexte de cette recherche en biologie et nous décrivons l'état de l'art des méthodes utilisées pour ce type de tâches dans la section 2. Nous décrivons notre approche dans la section 3 en présentant l'approche de [Wei08] puis l'extension que nous en proposons pour gérer le cas de données faiblement supervisées. Dans la section 4, nous décrivons des résultats expérimentaux de notre approche en étudiant le rôle des différentes composantes. Enfin dans la dernière section, nous verrons les perspectives et les applications possibles de notre méthode.

2 Contexte et état de l'art

2.1 Contexte

L'ARN est le porteur de l'information génétique qui permet de produire les protéines associées à chacun des gènes (définis comme des séquences d'ADN). Mesurer les quantités d'ARN permet de quantifier l'état moléculaire d'une cellule. L'ensemble des molécules d'ARN est appelé le transcriptome, en référence au processus de transcription de l'ADN en ARN. La méthode de scRNASeq permet d'établir pour chaque cellule, un profil transcriptomique, c'est à dire, une mesure de la quantité d'ARN associée à chacun des gènes. En pratique, ce profil transcriptomique est, pour chaque cellule, un vecteur de dimension élevée (en général de l'ordre de 20,000 : le nombre de gènes considérés), tel que chacune des coordonnées de ce vecteur représente le nombre de molécules d'ARN mesurées associé à un gène donné. Le profil transcriptomique permet de caractériser quantitativement l'activité moléculaire d'une cellule. Dans le cas de l'étude du développement embryonnaire, il est nécessaire de comprendre comment les populations de cellules qui forment les tissus se

coordonnent et changent d'état moléculaire au cours du processus de différentiation cellulaire. Pour comprendre les dynamiques de population de cellules, il est intéressant d'étudier la dynamique de leurs profils transcriptomiques dans l'espace (physique) et dans le temps. Cependant, la majeure limitation des méthodes scRNASeq est qu'en faisant la mesure des profils transcriptomiques on perd l'information spatiale (physique) et temporelle précise associée à chacune des cellules. Nous nous proposons ici de retrouver l'information temporelle et spatiale associée à chaque profil transcriptomique en utilisant des annotations partielles et la structure globale des données. Nous illustrons notre approche sur le jeu de données Packer et al. [P$^+$19] qui contient une série temporelle de profils transcriptomiques pour l'organisme modèle *C. elegans*.

Au cours du développement embryonnaire, une cellule unique se multiplie par divisions successives. Prises ensemble, ces divisions successives forment un arbre de généalogie, plus communément appelé arbre de lignage cellulaire, qui permet de relier chacune des cellules à toutes les autres [V$^+$16, SPS21]. Ici, nous nous intéressons à l'organisme modèle *C. elegans*, qui est composé d'environ 1000 cellules à l'âge adulte et qui a la particularité d'avoir un arbre de lignage parfaitement reproductible d'un embryon à l'autre [S$^+$83]. Dans la suite, nous utilisons cette propriété pour organiser les données transcriptomiques. Au cours des dernières années, les méthodes permettant de retrouver l'arbre de lignage cellulaire à partir des données d'expression incluent un système de code barre qui, grâce à des mutations successives, permet de retrouver les relations de hiérarchie [RGS18, WK20]. Cependant cette approche est assez lourde et ne permet pas de résoudre l'identification à l'échelle de la cellule individuelle.

Nous proposons ici de déduire la position des vecteurs transcriptomiques dans l'arbre de lignage à partir des vecteurs d'expression eux-mêmes [P$^+$19], voir Fig. 1. Ces données ont été en partie annotées manuellement en utilisant des jeux de données complémentaires [Ma12, A$^+$14]. Parmi les 89701 vecteurs transcriptomiques, 6% ont été identifiés de manière unique, 47% ont été identifiés de manière ambiguë et enfin les derniers 47% n'ont pas de label (Nous ne considérerons pas ces dernières données ici).

2.2 État de l'art

L'analyse des données scRNASeq a été abordée de différentes manières. D'une part, étant donné la grande dimension de ces jeux de données, plusieurs méthodes de réduction de la dimension non-linéaire ont été développées, notamment pour préserver la structure des données et permettre l'identification de sous-catégories ayant une justification biologique [B$^+$19, M$^+$19]. Par ailleurs, un certain nombre de méthodes ont pour objectif de reconstruire la façon dont ces données évoluent au cours du temps, à travers la question de l'inférence de trajectoires [SCTS19, S$^+$19]. D'autre part, certains travaux ont cherché à inférer la structure de ces jeux de données en explicitant leur lien avec l'organisation spatiale des cellules [NKFR19, PAV20]. Enfin, plusieurs travaux ont cherché à inférer la structure d'arbre de lignage à partir de données transcriptomiques, mais générées de telle sorte que ces données accumulent des mutations permettant de récapituler partiellement les divisions successives [ZLBJ20, Wa21]. Cependant, pour trouver des travaux concernant la reconstruction d'arbre de lignage à partir de données faiblement supervisées, il faut se tourner vers des données de microscopie, qui sont obtenues pour des systèmes similaires, mais sont de beaucoup plus faibles dimensions [MM$^+$21]. Dans ce cas, les contraintes sur les données sont très différentes de celles contenues dans les données scRNASeq et ne sont en l'état pas transférables directement à notre problème.

Du côté de la littérature en apprentissage automatique, la classification dans une hiérarchie de labels a été abordée par diverses approches dans la communauté apprentissage automatique : l'extension de SVM [CH04], l'apprentissage d'une projection dans un espace commun des labels et des données [Wei08], la formalisation sous forme d'un problème de régression vers un espace de représentation des labels respectant l'information hiérarchique [CAG12, GY13, M$^+$21]. La classification hiérarchique a été intensivement étudiée pour des données images [BWG10] et textuelles [PAA$^+$14]. Les recherches se sont focalisées sur des contextes d'applications spécifiques à ces données, des problèmes de grande échelle (en nombre de classes et en nombre de données) [BN09, BWG10, PAA$^+$14, GY15, B$^+$16], la classification hiérarchique multilabels, ou encore la découverte d'une hiérarchie sous-jacente aux données [PAA$^+$14]. En comparaison peu de travaux ont concerné l'apprentissage avec peu de données et/ou avec une faible supervision dans un cadre similaire au nôtre [MSZH19, ZCMH21]. Par exemple [MSZH19, ZCMH21] proposent une approche type EM mais dédiée aux données textuelles et pour un nombre limité de labels organisés dans une hiérarchie très plate (de hauteur égale à 2 ou 3).

3 Modélisation

3.1 Définition du problème

On cherche à résoudre un problème de classification hiérarchique de données $x \in \mathbb{R}^p$ dans un arbre $\mathcal{Y}$ de labels. Dans le cas totalement supervisé l'objectif de la classification est d'apprendre à déterminer le label y d'un vecteur x à l'aide d'une collection de données d'apprentissage $\left\{x_i, y_i\right\}_{1 \leq i \leq L} \in \mathcal{X} \times \mathcal{Y}$. De multiples approches ont été proposées pour cela [CH04, BWG10, Wei08].

Nous nous intéressons à la situation dans laquelle seule une partie des données est étiquetée, mais une autre partie des données est étiquetée avec incertitude, c'est à dire que la supervision associée à un exemple x_i de ce type est un sous-ensemble Y_i des étiquettes de $\mathcal{Y}$, de cardinal variable (le plus souvent inférieur à 10 dans notre cas pratique). L'ensemble de données que l'on souhaite classer est donc la réunion de deux sous-ensembles de données :

$$\begin{aligned} B = &\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1..L\} \\ &\cup \{(x_i, Y_i) \in \mathcal{X} \times \mathcal{P}(\mathcal{Y}), i = L+1..L+N\} \\ &\equiv B_L \cup B_{WL} \end{aligned} \quad (1)$$

où $\mathcal{P}(\mathcal{Y})$ représente l'ensemble des parties de $\mathcal{Y}$, B_L est l'ensemble des L données étiquetées et B_{WL} est l'ensemble des N données étiquetées avec ambiguïté. La difficulté vient du fait que l'ensemble des étiquettes apparaissant dans les labels des données étiquetées (parfaitement ou avec ambiguïté) ne couvrent qu'une partie de l'ensemble des labels de $\mathcal{Y}$. On peut alors envisager de s'appuyer sur des hypothèses telles que la proximité entre données x_1 et x_2 implique une proximité entre leur labels respectifs y_1 et y_2.

Dans ce travail préliminaire nous cherchons à apprendre à classer les données de $\mathcal{X}$ dans $\mathcal{Y}$ à partir de données des ensembles d'apprentissage B_L et B_{WL}. Nous présentons en Section 3.2 les principes généraux d'une approche de classification hiérarchique développée dans [Wei08] pour le cas purement supervisé et sur laquelle nous nous appuierons dans ce travail. Nous en décrivons en section 3.3 une extension pour gérer le cas de données étiquetées de façon ambiguë.

3.2 Classification hiérarchique supervisée

L'approche de [Wei08] considère ici un problème d'apprentissage supervisé où l'on veut apprendre à classer des vecteurs dans les labels organisés en arbre à partir d'une base étiquetée $B_L = \left\{(x_i, y_i) \in \mathbb{R}^p \times \mathcal{Y}, 1 \leq i \leq L\right\}$.

L'approche consiste en l'apprentissage de deux projections, l'une opérant sur des représentations non informatives des labels (one-hot-encoding) et l'autre sur les exemples, permettant de plonger labels et exemples dans un espace de représentation commun. La classification est réalisée dans cet espace par la méthode de plus petite distance aux vecteurs prototypiques des labels. Partant d'une solution analytique du problème, les performances de l'approche sont optimisées par l'affinage des projections à l'aide d'un critère de maximum de marge.

Classification par plus proche prototype (*Taxem*) La représentation initiale, non informative, des labels $y_i \in \mathcal{Y}$ est donnée par les vecteurs unitaires de la base canonique de $\mathbb{R}^c$ (où c est le nombre de labels dans $\mathcal{Y}$). On note $e_{y_i} = (0, ..., 0, 1, 0, .., 0) \in \mathbb{R}^c$, qui est un vecteur de 0 sauf à la $y_i^{ème}$ coordonnée qui vaut 1. On note Y l'espace de ces vecteurs de labels.

Plutôt que de réaliser la classification dans l'espace Y on considère une projection P qui projette ces représentations des labels de Y dans un nouvel espace de dimension éventuellement plus faible d (par exemple avec MDS pour que cette projection respecte les distances dans l'arbre entre labels). Au plus simple on peut utiliser une projection linéaire, voire P=Id. Dans la suite les labels ont des représentations $p_{e_{y_i}} = Pe_{y_i}$. On note alors $P(Y)$ l'espace engendré par ces représentations.

Le modèle de classification est appris en projetant les données x_i dans l'espace $P(Y)$, de manière à ce que chaque exemple x_i soit projeté au plus près de la représentation du label qui lui correspond, $p_{e_{y_i}}$. On détermine alors cette projection $W \in \mathbb{R}^{d \times p}$ via le problème d'optimisation suivant :

$$\underset{W \in \mathbb{R}^{d \times p}}{\arg\min} \sum_{i=1}^{n} ||p_{e_{y_i}} - Wx_i||^2 + \lambda_r ||W||_F^2 \quad (2)$$

dont la solution analytique est donnée par : $W = PA$ avec : $A = JX^t(XX^t + \lambda_r I_p)^{-1}$ où $J \in \mathbb{R}^{c \times n}$ est la matrice encodant les classes des exemples, i.e. $\forall c, \forall i, J_{c,i} = \mathbb{1}_{[y_i = c]}$. Une fois le modèle de régression appris un exemple x peut être classé suivant :

$$\hat{y} = \underset{\alpha \in \mathcal{Y}}{\arg\min} ||p_{e_\alpha} - Wx||_2^2 \quad (3)$$

Maximisation de la marge (*LM-Taxem*) Pour augmenter la capacité discriminante du modèle on utilise un critère de marge, afin que les exemples x_i soient projetés au plus proche de leurs labels associés y_i et le plus éloignés des autres labels. On remarque tout d'abord que $||Pe_{y_i} - Wx_i||^2 = ||P(e_{y_i} - Ax_i)||^2 \equiv ||P(e_{y_i} - x'_i)||^2$ en notant $x'_i = Ax_i$. Idéalement on souhaite que la distance d'un point x_i à son label e_{y_i} soit plus petite que la distance à tout autre label α et ce d'autant plus que α et y_i sont distants dans l'arbre. On cherche P tel que :

$$\forall i \ \forall \alpha \in \mathcal{Y} \setminus \{y_i\} : \\ ||P(e_{y_i} - x'_i)||_2^2 \leq ||P(e_\alpha - x'_i)||_2^2 - C_{y_i,\alpha}$$

où $C_{y,y'}$ est une mesure de dissimilarité entre deux labels $y, y' \in \mathcal{Y}^2$, par exemple la distance dans l'arbre. Il s'agit d'une formalisation classique d'un problème de prédiction structurée. Afin de relâcher les contraintes, on introduit une variable ressort et on cherche P tel que :

$$\forall i \ \forall \alpha \in \mathcal{Y} \setminus \{y_i\} : \\ ||P(e_{y_i} - x'_i)||_2^2 \leq ||P(e_\alpha - x'_i)||_2^2 - C_{y_i,\alpha} + \xi_i$$

On formalise l'apprentissage comme le problème de minimisation de la quantité :

$$L = \sum_i^n \xi_i \tag{4}$$

Il s'agit d'un cadre classique de prédiction structurée et on peut montrer que les variables ressort satisfont :

$$\forall i, \xi_i = \max(0, \\ \max_\alpha ||P(e_{y_i} - x'_i)||_2^2 - ||P(e_\alpha - x'_i)||_2^2 + C_{y_i,\alpha}) \tag{5}$$

Le critère optimisé est une borne supérieure de la dissimilarité moyenne entre le label prédit et le vrai label. En notant $\hat{y}_i$ la prédiction du modèle pour x_i, on a :

$$\mathcal{E} = \sum_{i=1}^n C_{y_i,\hat{y}_i} \leq \sum_{i=1}^n \xi_i \tag{6}$$

où $\mathcal{E}$ représente la somme des erreurs d'assignation au sein de l'arbre de lignage. Nous utiliserons la distance dans l'arbre dans nos expérimentations.

3.3 Extension à la prise en compte d'incertitude dans l'étiquetage

Dans cette partie, nous nous intéressons au cas où il existe un sous-ensemble $B_{WL} \subset \mathcal{B}$ des données pour lequel on ne dispose que d'un ensemble de labels possibles. Ici $B = B_L \cup B_{WL} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1..L\} \cup \{(x_i, \mathcal{Y}_i) \in \mathcal{X} \times \mathcal{P}(\mathcal{Y}), i = L+1..L+N\}$.

3.3.1 Variante à variables latentes

Nous proposons d'utiliser un algorithme inspiré de la prédiction structurée avec variables latentes [TJHA05] pour prendre en compte les données de B_{WL} dont la supervision est incertaine. Où l'on considère une variable latente pour chaque exemple $(x_i, \mathcal{Y}_i) \in B_{WL}$ qui correspond à l'étiquetage hypothétique $\tilde{y}_i \in \mathcal{Y}_i$ de cet exemple, qui est progressivement raffiné pendant l'optimisation. L'algorithme consiste à itérer deux étapes, tout d'abord l'inférence d'une étiquette $\tilde{y}_i \in \mathcal{Y}_i$ pour chacun des exemples $x_i \in B_{WL}$, puis la ré-estimation de la projection P à l'aide de la méthode de classification supervisée sur les données de B_L et sur les données pseudo-étiquetées i.e. $B_L \cup (x_i, \tilde{y}_i), i = L+1, .., L+N$. Cet algorithme est décrit dans l'algorithme P.

Algorithme 1 : ALGO P

Entrées : $B_L = \left\{(x_i, y_i)\right\}_{1 \leq i \leq L}$

$B_{WL} = \left\{(x_i, \mathcal{Y}_i)\right\}_{L+1 \leq i \leq L+N}$

Initialiser P, J (Eq. 7)

pour *e dans* $[[0, n_{epoch}]]$ **faire**

  pour chaque $x_i \in B_{WL}$ **faire**

    Attribuer une étiquette $\tilde{y}_i$ (Eq. 8)

  fin

  Calculer $\{\xi_i\}_i$ et $\{\tilde{\xi}_i\}_i$ (Eq. 5 et 9)

  Optimiser P pour minimiser L (Eq. 10)

fin

Nous avons choisi de fixer J comme précédemment pour les données étiquetées ($J_{\alpha,i} = \mathbb{1}_{[y_i=c]}$) et d'injecter l'incertitude de l'étiquetage d'un exemple (x_i, Y_i) de B_{WL} dans la colonne correspondante de J en définissant

$$J_{\alpha,i} = \begin{cases} \frac{1}{k} \text{ si } \alpha \in \mathcal{Y}_i \text{ et } k = |\mathcal{Y}_i| \\ 0 \text{ sinon} \end{cases} \tag{7}$$

Puis on infère pour chaque $x_j \in B_{WL}$ le label le plus vraisemblable donné par :

$$\tilde{y}_i = \arg\min_{\alpha \in \mathcal{Y}_i} ||P(e_\alpha - x'_i)||_2^2 \tag{8}$$

On définit ensuite la variable ressort associée à $\tilde{y}_i$:

$$\tilde{\xi}_i = \max(0, \\ \max_\alpha ||P(e_{\tilde{y}_i} - x'_i)||_2^2 - ||P(e_\alpha - x'_i)||_2^2 + C_{\tilde{y}_i,\alpha}) \tag{9}$$

L'optimisation est réalisée en pondérant les deux termes d'erreur, le premier sur B_L le second sur B_{WL}.

$$\hat{P} = \arg\min_P (L \equiv \mu L_L + (1-\mu) L_{WL}) \qquad (10)$$

avec μ un hyperparamètre et, en utilisant les définitions précédentes de ξ_i et de $\tilde{\xi}_i$:

$$L_L = \sum_{(x_i,y_i)\in B_L} \xi_i \text{ et } L_{WL} = \sum_{(x_i,\mathcal{Y}_i)\in B_{WL}} \tilde{\xi}_i \qquad (11)$$

3.3.2 Gestion de l'incertitude dans la projection

La stratégie précédente vise à projeter les exemples et les labels dans un espace commun puis à apprendre l'opérateur de projection P. Mais l'espace de projection commun est calculé en prenant en compte une incertitude maximale sur les exemples de B_{WL}, puisque les colonnes de J correspondant à ces exemples contiennent une information d'équiprobabilité d'appartenance aux labels de $\mathcal{Y}_i$. Or, au fur et à mesure de l'apprentissage on peut espérer que certaines données de $\mathcal{Y}_i$ soient de mieux en mieux classées. Les colonnes correspondantes de J devraient refléter cette information pour que l'espace de projection commun soit optimisé en fonction de cette information. Nous proposons ainsi d'apprendre de façon combinée P et A en ajoutant une étape de ré-estimation de la matrice J. Plutôt que de figer J à l'initialisation comme précédemment nous proposons de mettre à jour les colonnes correspondant aux données de B_{WL} selon :

$$\tilde{J}_{\alpha,i} = \frac{e^{-\beta||P(e_\alpha - x_i)||}}{\sum_{\alpha\in\mathcal{Y}_i} e^{-\beta||P(e_\alpha - x_i)||}} \qquad (12)$$

où β est un hyper-paramètre à déterminer. Les valeurs $\tilde{J}_{\alpha,i}$ peuvent être interprétées comme des probabilités d'affectation des labels y_α à l'exemple x_i. Cette mise à jour de J est réalisée à chaque itération, avant de calculer à nouveau la régression sur les représentations des labels. Cet algorithme est nommé *ALGO PJ* et décrit dans l'algorithme PJ.

Afin de mieux explorer les effets des deux composantes nous comparerons les deux algorithmes précédents à un troisième algorithme qui ne modifie itérativement que la matrice J. L'algorithme correspondant nommé *ALGO J* est décrit dans l'algorithme J. Bien entendu cet algorithme n'incluant aucun apprentissage de la projection P ne permet pas en général d'obtenir de bonnes performances sur les données de B_L mais les performances obtenues sur B_{WL} sont indicatives de la possibilité d'inférer les bons labels des données de B_{WL} pour une projection P donnée, et du coup de la faisabilité de la variante à variables latentes.

Algorithme 2 : ALGO PJ

Entrées : $B_L = \left\{(x_i, y_i)\right\}_{1\le i\le L}$,
$B_{WL} = \left\{(x_i, \mathcal{Y}_i)\right\}_{L+1\le i\le L+N}$
Initialiser P, J (Eq. 7)
pour *e dans* $[[0, n_{epoch}]]$ **faire**
  Calculer $\tilde{J}$ avec (Eq. 12)
  Effectuer la régression
  $X' = \left(\tilde{J}X^t(XX^t + \lambda_r I)^{-1}\right)X$
  pour chaque $x_i \in B_{WL}$ **faire**
    Attribuer une étiquette $\tilde{y}_i$ (Eq. 8)
  fin
  Calculer $\{\xi_i\}_i$ et $\{\tilde{\xi}_i\}_i$ (Eq. 5 et 9)
  Optimiser P pour minimiser L (Eq. 10)
fin

Algorithme 3 : ALGO J

Entrées : $B_{WL} = \left\{(x_i, \mathcal{Y}_i)\right\}_{L+1\le i\le L+N}$
Initialiser P, J (Eq. 7)
pour *e dans* $[[0, n_{epoch}]]$ *:* **faire**
  Calculer $\tilde{J}$ avec (Eq. 12)
  Effectuer la régression
  $X' = \left(\tilde{J}X^t(XX^t + \lambda_r I)^{-1}\right)X$
fin

4 Expérimentations

4.1 Données et protocole expérimental

Données Les données sur lesquelles nous validons ce travail sont issues de [P+19] et consistent en un ensemble de vecteurs transcriptomiques partiellement annotés, voir Fig. 1 et section 2.1. Ce jeu de données se compose de 89701 vecteurs de dimension 20000, où chaque coordonnée correspond au taux d'expression d'un gène. Nous avons réduit la dimension à 3842, en sélectionnant un sous-ensemble représentatif de gènes à l'aide de connaissances à priori. Les labels ont une structure d'arbre qui est dérivée du fait qu'ils encodent les relations de divisions successives dans le développement embryonnaire de l'organisme modèle *C. elegans*. Si l'on considère l'arbre complet, il y a 1342 labels possibles [S+83].

On définit à partir de cet arbre une matrice de distance entre labels C. Ici, nous avons choisi de prendre, comme distance entre deux labels, le plus court chemin dans l'arbre entre ces deux noeuds. La valeur moyenne de la matrice C est de 14 et la valeur maximale est de 20.

Dans les résultats que nous reportons, nous avons utilisé uniquement la partie du jeu de données qui est annotée de manière non ambiguë. Cela correspond à 5167 vecteurs, pour lesquels nous avons un label unique. Lorsque l'on regarde l'ensemble des labels possibles associés à cette sous partie du jeu de données, cela correspond à 95 labels.

Protocole expérimental. Pour les expériences en mode supervisé, nous avons créé 20 jeux de données où nous divisons aléatoirement le jeu de données en une partie (80%) d'entraînement et une partie (20%) de test. Les résultats sont moyennés sur ces 20 jeux de données.

Pour étudier le cas de données étiquetées avec ambiguïté nous avons créé des jeux de données de la façon suivante : d'abord nous effectuons une partition des labels possibles, conduisant à deux ensembles de labels $\mathcal{Y}_L$ et $\mathcal{Y}_{WL}$ d'intersection vide, ce choix reflète l'organisation réelle des données biologiques. On construit B_L comme l'ensemble des exemples dont le label est dans $\mathcal{Y}_L$. On construit B_{WL} en tirant au hasard pour chaque exemple (x_i, y_i) un sous-ensemble $\mathcal{Y}_i$ de $\mathcal{Y}$ de taille $k \in \{2, 4, 6, 8\}$ incluant y_i. Enfin, pour chacun des deux ensembles B_L et B_{WL}, nous prenons 80% des données en entraînement et 20% en test de manière aléatoire. Les résultats présentés sont moyennés sur ces 10 jeux de données.

Métriques : Nous utilisons plusieurs métriques pour comparer les approches. La moyenne des erreurs d'assignation au sein de l'arbre entre la prédiction et le vrai label est calculée suivant $\frac{1}{n}\sum_{i=1}^{n} C_{y_i,\hat{y}_i}$, nous la notons ε. Nous avons également calculé les scores micro-F1 et Macro-F1 notés $\mu F1$ et $MF1$ respectivement.

4.2 Mode supervisé

L'approche introduite en section 3.2, développée dans [Wei08] comporte deux composantes. La première consiste en une classification par plus proche prototype après un plongement des labels et des exemples dans un espace commun et peut être utilisée seule, nous nommons cette méthode *Taxem*. La méthode de Chapelle est notée *LM-Taxem* en référence à [Wei08]. Le tableau 1 résume les résultats obtenus par les classifieurs OnevsRest, Randomforest, Taxem, LM-taxem. Les meilleurs résultats sont notés en gras. On observe que la méthode *LM-Taxem* est effectivement très performante sur nos données mais également que la première composante de la méthode obtient déjà des résultats tout à fait corrects par rapport à une baseline souvent difficile à battre, OnevsRest.

4.3 Données étiquetées avec ambiguïté

Résultats préliminaires. L'algorithme PJ pour les données ambiguës repose sur le mode supervisé en exploitant une estimation de l'étiquetage des données ambiguës. Afin d'explorer à priori la pertinence de cette approche nous avons réalisé des expériences en mode supervisé en randomisant une partie des labels des données d'apprentissage. Nous avons observé que jusqu'à au moins 20% de données d'apprentissage étiquetées aléatoirement, la performance en généralisation de la méthode n'était pas affectée, ce qui est encourageant.

Détails de l'optimisation. Pour résoudre le problème de minimisation de P, dans les algorithmes P et PJ , nous avons utilisé le module pytorch en Python, avec la librairie d'optimisation associée "autograd" [PGM+19]. Pour les problèmes de minimisation formulés dans les équations 4 et 10, nous appliquons l'algorithme Adam avec un learning rate de 0.05.

L'initialisation de P dans les algorithmes P, PJ et J est réalisée par la projection des labels dans l'espace $\mathbb{R}^d$ avec Multi-Dimensional Scaling (en utilisant la librairie scikit-learn) [PVG+11].

Résultats Nous proposons un comparatif des algorithmes P, J, PJ dans les tables 2, 3, 4 pour un ensemble d'hyperparamètres choisis pour un bon comportement en général des méthodes : $\mu = 0.9, d = 40$. Les calculs ont été faits sur 10 jeux de données construits selon le protocole précédent et nous exposons la moyenne sur les datasets. Les résultats sont présentés en fonction du % de labels contenus dans B_{WL} et de k le nombre d'étiquettes possibles pour $\{\mathcal{Y}_i\}_{L+1 \leq i \leq L_W}$. Nous mettons en gras les meilleurs résultats pour ε sur l'ensemble B_{WL}.

Nous remarquons tout d'abord que les performances sur l'ensemble B_L sont similaires par les algorithmes P et PJ, quelque que soit la quantité de données dans B_{WL}.

	Méthodes	OnevsRest	Random Forest	Taxem	LM-taxem
B_L	ε	3.52 ± 0.20	3.23 ± 0.17	2.92 ± 0.22	**2.11** ± 0.12
	μF1	0.70 ± 0.01	0.72 ± 0.01	**0.75** ± 0.01	0.74 ± 0.01
	MF1	0.61 ± 0.01	0.65 ± 0.01	0.62 ± 0.02	**0.66** ± 0.01

TABLE 1 – Comparaison des méthodes sur l'ensemble supervisé B_L. Les résultats sont moyennés sur 20 expériences avec indication de l'écart type. Les méthodes *Taxem* et *LM-taxem* sont décrites dans la section 3.2

Ensuite, nous observons que les performances sur B_{WL}, sont en générales meilleures pour l'algo PJ que pour l'algo P, particulièrement lorsque la proportion de données ambiguës et le niveau d'ambiguïté augmente : pour 20% de données dans B_{WL},soit la table 2, les algorithmes P et PJ ont des performances semblables sur B_{WL}. Pour les tables 3, 4, soit 50% et 80% de données dans B_{WL}, l'algo PJ obtient des meilleurs résultats en général. Enfin, le gain de l'algo PJ par rapport à l'algo P augmente, lorsque la valeur de k augmente.

5 Conclusion

Nous avons exploré l'extension de méthodes de classification hiérarchique à des données supervisées avec ambiguïté telles qu'il en existe couramment dans des problèmes en biologie du développement. La variante à variables latentes que nous proposons et dont nous avons proposé plusieurs variantes atteint des résultats prometteurs sur un jeu de données classique en biologie sur lequel cette tâche n'est pas abordée pour le moment à notre connaissance.

6 Remerciements

Malek Senoussi et Paul Villoutreix ont été financés par le programme "Investissements d'Avenir" du gouvernement français géré par l'Agence Nationale de la Recherche (ANR-16-CONV-0001) et par l'Initiative d'Excellence d'Aix-Marseille Université - A*MIDEX.

Références

[A+14] Araya et al. Regulatory analysis of the c. elegans genome with spatiotemporal resolution. *Nature*, 2014.

[B+16] Bhatia et al. The extreme classification repository : Multi-label datasets and code, 2016.

[B+19] Becht et al. Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology*, 2019.

[BN09] Paul N. Bennett and Nam Nguyen. Refined experts : improving classification in large taxonomies. ACM, 2009.

[BWG10] Samy Bengio, Jason Weston, and David Grangier. Label embedding trees for large multi-class tasks. NIPS, 2010.

[CAG12] Moustapha Cissé, Thierry Artières, and Patrick Gallinari. Learning compact class codes for fast inference in large multi class classification. ECML, 2012.

[CH04] Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. ACM, 2004.

[GY13] Siddharth Gopal and Yiming Yang. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. ACM, 2013.

[GY15] Siddharth Gopal and Yiming Yang. Hierarchical bayesian inference and recursive regularization for large-scale classification. *ACM*, 2015.

[M+19] Moon et al. Visualizing structure and transitions in high-dimensional biological data. *Nature biotechnology*, 2019.

[M+21] Mittal et al. DECAF : deep extreme classification with label features. ACM, 2021.

[Ma12] Murray and al. Multidimensional regulation of gene expression in the c. elegans embryo. *Genome research*, 2012.

[MM+21] Malin-Mayor et al. Automated reconstruction of whole-embryo cell lineages by learning from sparse annotations. *bioRxiv*, 2021.

[MSZH19] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. Weakly-supervised hierarchical text classification. 2019.

[NKFR19] Mor Nitzan, Nikos Karaiskos, Nir Friedman, and Nikolaus Rajewsky. Gene expression cartography. *Nature*, 2019.

[P+19] Packer et al. A lineage-resolved molecular atlas of c. elegans embryogenesis at single-cell resolution. *Science*, 2019.

[PAA+14] Ioannis Partalas, Massih-Reza Amini, Ion Androutsopoulos, Thierry Artières, Patrick Gallinari, Éric Gaussier, and Georgios Paliouras. Web-scale classification : web classification in the big data era. ACM, 2014.

[PAV20] Julie Pinol, Thierry Artières, and Paul Villoutreix. Towards a general framework for spatio-temporal transcriptomics. In *LMRL Workshop-NeurIPS*, 2020.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor

20 % de données dans B_{WL}						
		k	2	4	6	8
AlgoP	B_L	ε	2.21 ± 0.09	2.30 ± 0.09	2.35 ± 0.13	2.35 ± 0.11
		μF1	0.73 ± 0.01	0.72 ± 0.01	0.72 ± 0.01	0.72 ± 0.01
		MF1	0.65 ± 0.01	0.65 ± 0.01	0.64 ± 0.02	0.64 ± 0.01
	B_{WL}	ε	**2.19** ± 0.30	2.39 ± 0.23	**2.43** ± 0.17	2.41 ± 0.19
		μF1	0.74 ± 0.02	0.73 ± 0.02	0.73 ± 0.01	0.73 ± 0.01
		MF1	0.62 ± 0.03	0.62 ± 0.02	0.61 ± 0.02	0.61 ± 0.02
AlgoJ	B_L	ε	2.67 ± 0.11	2.92 ± 0.16	3.03 ± 0.14	3.08 ± 0.19
		μF1	0.54 ± 0.01	0.49 ± 0.02	0.48 ± 0.02	0.47 ± 0.02
		MF1	0.51 ± 0.01	0.47 ± 0.01	0.45 ± 0.02	0.45 ± 0.01
	B_{WL}	ε	2.68 ± 0.28	2.95 ± 0.29	3.14 ± 0.33	3.17 ± 0.27
		μF1	0.55 ± 0.04	0.51 ± 0.04	0.50 ± 0.03	0.49 ± 0.03
		MF1	0.47 ± 0.02	0.44 ± 0.02	0.42 ± 0.02	0.41 ± 0.02
AlgoPJ	B_L	ε	2.21 ± 0.10	2.28 ± 0.11	2.36 ± 0.09	2.35 ± 0.09
		μF1	0.73 ± 0.01	0.73 ± 0.01	0.72 ± 0.01	0.72 ± 0.01
		MF1	0.65 ± 0.01	0.65 ± 0.02	0.64 ± 0.01	0.64 ± 0.01
	B_{WL}	ε	2.20 ± 0.26	**2.26** ± 0.22	2.49 ± 0.20	**2.37** ± 0.23
		μF1	0.75 ± 0.02	0.74 ± 0.01	0.73 ± 0.01	0.73 ± 0.01
		MF1	0.63 ± 0.03	0.62 ± 0.02	0.61 ± 0.03	0.61 ± 0.02

TABLE 2 – Comparaison des algorithmes P, J et PJ pour 20 % de données ambiguës. Les résultats sont moyennés sur 10 expériences avec indication de l'écart type.

50 % de données dans B_{WL}						
		k	2	4	6	8
AlgoP	B_L	ε	2.42 ± 0.13	2.88 ± 0.16	3.11 ± 0.20	3.20 ± 0.14
		μF1	0.72 ± 0.01	0.68 ± 0.01	0.66 ± 0.02	0.65 ± 0.01
		MF1	0.63 ± 0.02	0.58 ± 0.01	0.57 ± 0.02	0.55 ± 0.02
	B_{WL}	ε	2.33 ± 0.13	2.77 ± 0.17	**2.95** ± 0.25	3.12 ± 0.18
		μF1	0.71 ± 0.01	0.69 ± 0.00	0.67 ± 0.01	0.66 ± 0.01
		MF1	0.62 ± 0.02	0.60 ± 0.01	0.57 ± 0.02	0.56 ± 0.02
AlgoJ	B_L	ε	3.47 ± 0.11	4.71 ± 0.22	5.23 ± 0.17	5.53 ± 0.28
		μF1	0.41 ± 0.01	0.29 ± 0.02	0.25 ± 0.01	0.23 ± 0.02
		MF1	0.38 ± 0.01	0.27 ± 0.03	0.23 ± 0.02	0.22 ± 0.02
	B_{WL}	ε	3.35 ± 0.21	4.57 ± 0.29	4.97 ± 0.30	5.33 ± 0.32
		μF1	0.41 ± 0.02	0.29 ± 0.02	0.26 ± 0.01	0.24 ± 0.02
		MF1	0.38 ± 0.02	0.27 ± 0.02	0.24 ± 0.02	0.23 ± 0.01
AlgoPJ	B_L	ε	2.47 ± 0.19	2.89 ± 0.15	3.10 ± 0.16	3.13 ± 0.10
		μF1	0.72 ± 0.02	0.68 ± 0.01	0.66 ± 0.01	0.65 ± 0.01
		MF1	0.62 ± 0.02	0.59 ± 0.01	0.56 ± 0.02	0.56 ± 0.02
	B_{WL}	ε	**2.32** ± 0.15	**2.72** ± 0.15	2.97 ± 0.16	**3.04** ± 0.20
		μF1	0.72 ± 0.01	0.69 ± 0.01	0.67 ± 0.01	0.66 ± 0.01
		MF1	0.63 ± 0.02	0.60 ± 0.02	0.57 ± 0.01	0.57 ± 0.02

TABLE 3 – Comparaison des algorithmes P, J et PJ pour 50 % de données ambiguës. Les résultats sont moyennés sur 10 expériences avec indication de l'écart type.

80 % de données dans B_{WL}						
		k	2	4	6	8
AlgoP	B_L	ε	2.79 ± 0.18	3.83 ± 0.26	4.30 ± 0.32	4.89 ± 0.27
		μF1	0.68 ± 0.01	0.61 ± 0.02	0.57 ± 0.02	0.52 ± 0.02
		MF1	0.57 ± 0.02	0.50 ± 0.03	0.45 ± 0.03	0.41 ± 0.03
	B_{WL}	ε	**2.84** ± 0.20	3.83 ± 0.19	**4.36** ± 0.20	4.80 ± 0.24
		μF1	0.68 ± 0.01	0.60 ± 0.01	0.56 ± 0.01	0.53 ± 0.01
		MF1	0.59 ± 0.02	0.52 ± 0.01	0.48 ± 0.01	0.45 ± 0.02
AlgoJ	B_L	ε	4.86 ± 0.27	7.85 ± 0.34	8.87 ± 0.24	9.07 ± 0.38
		μF1	0.25 ± 0.02	0.10 ± 0.02	0.06 ± 0.02	0.05 ± 0.01
		MF1	0.21 ± 0.02	0.08 ± 0.02	0.05 ± 0.02	0.05 ± 0.01
	B_{WL}	ε	4.87 ± 0.26	7.66 ± 0.23	8.67 ± 0.29	9.06 ± 0.19
		μF1	0.23 ± 0.01	0.10 ± 0.00	0.07 ± 0.00	0.05 ± 0.00
		MF1	0.21 ± 0.02	0.09 ± 0.00	0.07 ± 0.00	0.05 ± 0.00
AlgoPJ	B_L	ε	2.84 ± 0.28	3.89 ± 0.30	4.47 ± 0.19	4.89 ± 0.36
		μF1	0.69 ± 0.03	0.60 ± 0.02	0.56 ± 0.02	0.53 ± 0.03
		MF1	0.58 ± 0.04	0.48 ± 0.02	0.44 ± 0.02	0.42 ± 0.04
	B_{WL}	ε	2.91 ± 0.18	**3.78** ± 0.13	**4.36** ± 0.22	**4.73** ± 0.18
		μF1	0.67 ± 0.01	0.61 ± 0.01	0.56 ± 0.01	0.53 ± 0.01
		MF1	0.59 ± 0.02	0.52 ± 0.02	0.47 ± 0.02	0.44 ± 0.01

TABLE 4 – Comparaison des algorithmes P, J et PJ pour 80 % de données ambiguës. Les résultats sont moyennés sur 10 expériences avec indication de l'écart type.

Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch : An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. 12 :2825–2830, 2011.

[RGS18] Bushra Raj, James A Gagnon, and Alexander F Schier. Large-scale reconstruction of cell lineages using single-cell readout of transcriptomes and crispr–cas9 barcodes by scgestalt. *Nature protocols*, 2018.

[S+83] Sulston et al. The embryonic cell lineage of the nematode caenorhabditis elegans. *Developmental biology*, 1983.

[S+19] Schiebinger et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 2019.

[SCTS19] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. A comparison of single-cell trajectory inference methods. *Nature biotechnology*, 2019.

[SPS21] Tanja Stadler, Oliver G Pybus, and Michael PH Stumpf. Phylodynamics for cell biologists. *Science*, 371, 2021.

[TJHA05] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6 :1453–1484, 2005.

[V+16] Villoutreix et al. An integrated modelling framework from cells to organism based on a cohort of digital embryos. *Scientific reports*, 2016.

[Vil21] Paul Villoutreix. What machine learning can do for developmental biology. *Development*, 2021.

[Wa21] Wuming and all. Benchmarked approaches for reconstruction of in vitro cell lineages and in silico models of c. elegans and m. musculus developmental trees. *Cell Systems*, 2021.

[Wei08] Chapelle Weinberger. Large margin taxonomy embedding with an application to document categorization. *NeurIPS*, 2008.

[WK20] Daniel E Wagner and Allon M Klein. Lineage tracing meets single-cell omics : opportunities and challenges. *Nature Reviews Genetics*, pages 410–427, 2020.

[ZCMH21] Yu Zhang, Xiusi Chen, Yu Meng, and Jiawei Han. Hierarchical metadata-aware document categorization under weak supervision. ACM, 2021.

[ZLBJ20] Hamim Zafar, Chieh Lin, and Ziv Bar-Joseph. Single-cell lineage tracing by integrating crispr-cas9 mutations with transcriptomic data. *Nature communications*, 2020.