Apprentissage stochastique de représentation binaire pour la classification multi-classe dans un grand nombre de catégories

Thomas Gerald, Nicolas Baskiotis, Ludovic Denoyer

Apprentissage stochastique de représentation binaire pour la classification multi-classe dans un grand nombre de catégories

Thomas Gerald[1], Nicolas Baskiotis[1], et Ludovic Denoyer[1]

[1]Sorbonne Université, CNRS, Laboratoire d'informatique de paris 6, LIP6 , F-75005 Paris, France

Résumé

Dans le contexte de la classification multi-classe à grande échelle, les approches ensemblistes naïves telles que les classifieurs un contre tous sont considérées comme les plus performantes en matière de performance prédictive. Cependant, elles passent difficilement à l'échelle lorsque le nombre de catégories croît, la complexité en temps d'inférence et en taille des modèles évoluant a minima linéairement en fonction du nombre de catégories. En général deux grands types d'approches sont établis pour résoudre ce problème : la classification hiérarchique, qui nécessite souvent l'apprentissage d'une ontologie du problème comme préalable à la classification ; les méthodes ensemblistes telles que des représentations binaires pour compresser l'information (ECOC). Ces méthodes font souvent intervenir des a priori et des heuristiques plus ou moins complexes. Le travail présenté dans cet article introduit un nouveau modèle neuronal exploitant un apprentissage de représentation binaire - des codes sémantiques - pour projeter dans un même espace les entrées et les catégories associées. L'apprentissage des deux projections est effectué de manière simultanée en intégrant une couche discrète probabiliste non différentiable afin d'explorer l'espace de représentation binaire discret des codes associant exemples et classes. Cette architecture nommée *Deep Stochastic Neural Code* (DSNC) permet l'inférence d'une classe en un temps sous linéaire - et constant dans le cas extrême - en fonction du nombre de classes du problème sans nécessiter d'a priori sur les données. Les expériences sur plusieurs jeux de données large échelle permettent d'illustrer l'efficacité de notre approche.

Mots-clef : Classification Mono-Label, Réseaux de Neurones.

1 Introduction

La problématique de la classification de données en présence d'un grand nombre de classes a récemment attiré l'attention de la communauté d'apprentissage statistique, notamment avec des compétitions de classifications automatiques considérant plusieurs milliers de classes comme *LSHTC* [PKB+15] ou *ImageNet* [RDS+15] . La principale difficulté de la tâche est d'atteindre des bonnes performances prédictives tout en diminuant drastiquement la complexité en temps de l'inférence et en mémoire pour le stockage des modèles de façon à garantir l'efficacité des approches lorsque le nombre de classes croît. Si les algorithmes les plus couramment mis en œuvre dans le cas de la classification multi-classe permettent d'obtenir un très bon taux de bonne classification [APHS14], bien souvent le temps (respectivement le stockage) nécessaire à l'obtention de la catégorie est prohibitif : une large majorité des approches tendent à augmenter le temps de classification linéairement par rapport au nombre de classes, c'est le cas des approches un-contre-tous et des perceptrons multi-couches.

Avec l'augmentation des capacités de calcul (parallélisation du calcul matriciel via GPU) le temps d'inférence et d'optimisation a grandement diminué, pourtant le temps de classification reste toujours important pour traité de large volume de données. Dans cet article, nous proposons une nouvelle méthode de classification multi-classe mono-label basée sur l'apprentissage de réseaux multi-couches permettant de prédire la classe d'un exemple en temps sous linéaire relativement au nombre total de classes considérées. Cette approche utilise une projection des données dans un espace de Hamming. Pour cela, chaque exemple est projeté vers un vecteur binaire de faible dimension (un code), chacun de ces codes est alors associé à une catégorie par plusieurs procédés que nous

développerons dans la suite de l'article. L'algorithme d'apprentissage du modèle proposé se fait de bout en bout en apprenant simultanément l'association des codes aux catégories et la projection des données dans l'espace des codes.

Le travail que nous décrivons présente des similitudes avec le domaine de la *recherche d'informations*, et plus particulièrement aux méthodes de *Semantic Hashing*. À des fins d'indexation de documents (images, textes, page web....) et de requêtage, ces méthodes utilisent une projection de l'espace de description des exemples vers un espace discret. Néanmoins, les objectifs que nous poursuivons dans ce travail sont différents : le principal objectif du *semantic hashing* est d'obtenir que les exemples proches dans l'espace projeté soient similaires sémantiquement. - optimiser une mesure de précision ; l'objectif de notre approche est la classification - optimisation du rappel et de la précision - et donc orientée vers l'exploration de code afin de permettre une organisation des catégories.

Le modèle présenté est un réseau de neurones comprenant une couche discrète non différentiable pour permettre l'exploration de l'espace des codes. Dans la plupart des cas, l'apprentissage d'un tel modèle utilisant des projections non continues est résolu par une relaxation continue et un seuillage post-apprentissage. Afin de résoudre le problème de non-différentiabilité, nous proposons d'utiliser une couche stochastique en apprentissage et d'optimiser les paramètres du modèle via une approximation du gradient basé sur l'estimateur *Straight-Through* introduit dans [BLC13a], utilisé entre autres dans [HCS$^+$16].

Les contributions de ce travail sont les suivantes : 1) une nouvelle famille de réseaux de neurones permettant la classification lorsque le nombre de catégories est grand, en apprenant une projection des entrées vers des codes binaires et des codes vers des classes ; 2) un algorithme permettant d'apprendre de bout en bout la projection et ne nécessitant pas d'a priori pour associer codes et exemples, le modèle étant capable d'associer durant l'apprentissage classes, exemples et représentations. Nous illustrons l'efficacité de l'approche en termes de précision et de temps d'inférence à travers une série d'expériences comparant notre modèle à l'état de l'art.

L'article est organisé de la manière suivante : dans un premier temps, nous présenterons en section 2 les méthodes de l'état de l'art dans le domaine de la classification multi-classe ainsi que les approches basées sur le *Semantic Hashing* ; en section 3, nous introduirons les spécificités techniques de notre modèle ainsi que la procédure d'optimisation du modèle. Enfin, la section 4 introduit le protocole expérimental et présente les résultats des expériences réalisées sur des corpus/bases de données large échelle.

2 État de l'art

Un des problèmes prédominant en classification multi-classe coonsidérant un grand nombre de catégories est le compromis entre le taux d'erreur et la complexité en temps de classification d'un exemple - le temps d'inférence par rapport au nombre de catégories K. L'algorithme *un-contre-un* utilise $O(K^2)$ classifieurs, chaque classifieur comparant un couple de classes ; l'algorithme *un-contre-tous* utilise K classifieurs, chacun permettant de discriminer une classe contre toutes les autres. Ces deux approches sont efficaces pour discriminer un grand nombre de catégories, mais au prix d'un temps d'inférence au mieux linéaire en fonction du nombre de classes ; ces méthodes sont donc prohibitives lorsque le nombre de catégories à différencier devient trop important.

Plusieurs familles d'approches ont été proposées afin de réduire le coût en temps de l'inférence. La connaissance a priori d'une hiérarchie entre les classes permet d'utiliser des modèles hiérarchiques [BWG10, WMY13, PB15] qui permettent dans le meilleur des cas un temps d'inférence logarithmique en fonction du nombre de classes.

Cependant, le plus souvent la structure topologique des catégories n'est pas connue (pas d'ontologie des classes). Dans ce cas, l'utilisation des codes correcteurs d'erreurs pour la classification (*ECOC*, [DB95]) est une bonne alternative. Le principe de cette méthode est d'associer à chaque classe une représentation binaire (voire ternaire) et d'apprendre une fonction afin de faire correspondre chaque exemple à son code de classe. S'appuyant sur le fait qu'il est seulement nécessaire d'avoir un code de taille $log(K)$ pour pouvoir différencier K classes, le temps d'inférence résultant est alors en $O(log(K))$ dans le meilleur des cas [1]. Cette approche souffre principalement de deux inconvénients : d'une part le choix des codes associés à chaque classe est décidé soit de manière aléatoire soit en suivant des heuristiques plus ou moins complexes ([ZC13, CAG12]), menant ainsi soit à un calcul lourd soit à une représentation largement sous-optimale. D'autre part, les performances des algorithmes utilisant cette approche sont à l'heure actuelle en deçà des algorithmes classiques tels que l'apprentissage de classifieurs un contre tous. De plus les per-

1. En pratique, une taille du code en $\alpha(log(k))$ est nécessaire pour assurer une bonne précision avec $\alpha \in [10, 20]$

formances des approches ECOC et hiérarchiques sont hautement dépendantes de la séparabilité des sous-ensembles de classes, qui devient de plus en plus complexe avec l'augmentation du nombre de classes.

Apprendre des projections des données vers un espace de représentation discret (connu sous le nom de *Hashing*) est un sujet d'intérêt croissant dans le domaine de la recherche d'information afin d'indexer de grands corpus. Comme les bases de données large échelle contiennent souvent un grand nombre de caractéristiques (l'espace de description), la recherche d'exemples similaires est très coûteuse en temps et en puissance de calcul. Trouver une fonction de *Hashing* de l'espace de description des documents vers un espace discret - souvent binaire - de petite dimension permet de procéder à la recherche de plus proches voisins en un temps sous-linéaire par rapport au nombre d'exemples [NFS12]. Parmi ces approches on différencie deux grandes catégories : celles non supervisées telles que le *Local Sensitive Hashing* [GIM99] qui utilise une projection linéaire aléatoire des données ; et les approches supervisées, optimisant une fonction d'erreur permettant de conserver la similarité entre les exemples [SH09, LPLY15, DDC16, WTF09]. Ces algorithmes ont permis d'obtenir de très bonnes performances sur les mesures évaluées en recherche d'information [WZS+17] (taux de précision moyen par exemple). Malgré l'efficacité des ces approches pour retrouver les documents proches d'un document requête, les mesures optimisées pour la recherche d'information ne correspondent pas à celles du problème de classification : les fonctions de *hashing* permettent de conserver une similarité entre les données, mais pas de discriminer efficacement l'appartenance d'un exemple à une classe ni d'obtenir un bon taux de rappel. Néanmoins, les recherches en sémantique *Hashing* révèlent l'importance de la topologie des représentations : par exemple, les travaux de [ELLW+15] montrent l'influence de la compacité des représentations sur les performances.

3 Réseaux stochastiques profonds pour l'apprentissage de code binaire

3.1 Description du modèle

Considérons $X = \mathbb{R}^n$ l'espace de description des données et $\mathbf{K} = \{1, 2, ..., K\}$ l'espace des classes. Les données que l'on considère pour l'apprentissage correspondent donc à un sous ensemble $D \in X \times \mathbf{K}$. Soit c la dimension de l'espace des codes et $\mathbb{B} \in \{0, 1\}^c$ l'espace des codes (qui correspond à un espace de Hamming), et b_i la i^{eme} composante du vecteur $\boldsymbol{b} \in \mathbb{B}$. Étant donné une entrée $x \in X$, le label prédit est obtenu par le processus de prédiction stochastique suivant (voir figure 3.1) :

- une distribution $P(\boldsymbol{b}|x)$ sur l'espace des codes est inférée à partir de l'entrée x ; cette probabilité est modélisée par une fonction $\phi(x)$, un réseau de neurones multi-couche usuel ;
- un code $\boldsymbol{b}^x$ est tiré aléatoirement selon la distribution $P(\boldsymbol{b}|x)$;
- le code $\boldsymbol{b}^x$ permet de déduire la classe $y \in \mathbf{K}$ par une fonction de décodage.

La distribution $P(\boldsymbol{b}|x)$ n'est pas utilisée en tant que telle lors de l'inférence d'une classe, elle permet essentiellement l'exploration des codes lors de la phase d'apprentissage. L'aspect stochastique du codage permet d'explorer l'espace de représentation lors de l'apprentissage afin de garantir les performances du modèle. Une fois le modèle appris on préférera au tirage l'affectation de la valeur la plus probable du code $\boldsymbol{b}^x$. Pour la modélisation de la distribution, nous supposons que tous les bits du code sont indépendants : chaque code peut alors être modélisé via une distribution de Bernoulli de paramètre $\phi_i(x) = P(b_i = 1|x)$. On se propose d'apprendre les deux fonctions de projection de l'entrée vers un code (codage) et de projection de code vers une classe (décodage) simultanément. On notera d_θ la fonction de projection d'un code vers une classe.

3.2 Décodage vers l'espace des classes

Afin d'obtenir la catégorie d'un exemple, nous proposons trois différents procédés : le premier - que nous appellerons *décodage linéaire* - considère d_θ comme une fonction linéaire, les paramètres de cette fonction seront appris durant l'étape d'apprentissage décrit dans la section suivante. Cette première méthode ne permet pas d'obtenir un gain en temps d'inférence, la complexité étant similaire à celle d'un réseau multi-couche continu.

Le second procédé proposé est d'utiliser une approche de voisinage en comparant la représentation binaire obtenue à celles des exemples d'apprentissage. Pour cela, à la fin de l'entraînement du modèle, nous stockons l'ensemble des codes de l'ensemble d'apprentissage ; pour inférer la classe d'un exemple de l'ensemble de test, nous associons à cet exemple la classe du code d'apprentissage le plus proche. On dénomme ce procédé *décodage par plus proche voisin.*

Dans un espace de Hamming, retrouver le plus proche voisin peut être décidé beaucoup plus rapide-

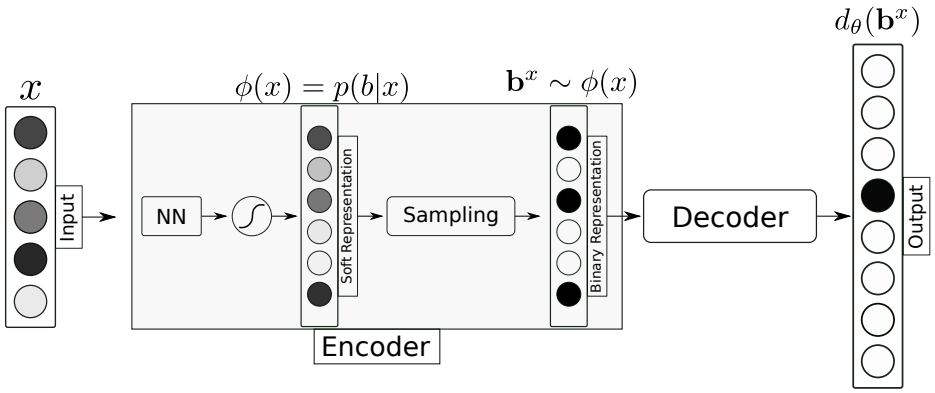

FIGURE 1 – Architecture du modèle, avec $\phi(x)$ la politique sur laquelle est effectué le tirage du code

ment que dans un espace continu. Nous discuterons de la complexité du problème dans la suite de l'article.

Enfin le dernier procédé est un cas extrême du précédent qui consiste à stocker tous les couples (code, classe) pour tous les codes possibles. Le décodage vers la classe à inférer est alors instantané.

3.3 Apprentissage du modèle

Notre réseau intègre une couche non-différentiable du fait du tirage aléatoire des codes à partir de la distribution inférée par l'entrée. Il n'est donc pas possible d'utiliser tel quel l'algorithme usuel de rétro-propagation du gradient pour l'apprentissage du modèle. Étant donné un couple $(x, y) \in D$, la fonction d'erreur exprimée sur la distribution des codes $\phi(x)$ est la suivante (en notant $\boldsymbol{b}^x$ le code tiré aléatoirement selon $\phi(x)$) : $\mathbb{E}_{\boldsymbol{b}^x \sim \phi(x)}(\Delta(d_\theta(\boldsymbol{b}^x), y))$. Pour la fonction de coût Δ, nous considérons la *negative log-vraisemblance* couramment utilisé en classification mono-label :

$$\Delta(y, y') = \sum_{i=0}^{K} -log(y_i) y'_i$$

.

L'optimisation associée au problème peut s'écrire :

$$\arg\min_{(\phi,\theta)} \mathbb{E}_{(x,y)\in D}[\mathbb{E}_{\boldsymbol{b}^x \sim \phi(x)}(\Delta(d_\theta(\boldsymbol{b}^x), y))] \quad (1)$$

L'optimisation de fonctions non différentiables est un sujet fréquemment abordé dans le domaine de l'apprentissage par renforcement. Plus particulièrement, l'algorithme *REINFORCE* [Wil92] permettant d'estimer le gradient de l'espérance par tirage de *Monte-Carlo* est une solution directement applicable à notre problème d'optimisation. Cependant de récents travaux ont montré l'intérêt d'autres variantes pour l'approximation de représentations discrètes. L'estimateur nommé *straight-through* proposé dans [BLC13b] s'est montré particulièrement efficace et beaucoup plus rapide que l'estimateur *REINFORCE*. Son efficacité à estimer un gradient a été par la suite appuyée par plusieurs travaux dont [CB16]. En utilisant cette approximation, les mises-à-jour de ϕ et d_θ sont les suivantes :

$$\theta_{t+1} = \theta_t - \sum_{(x,y)\in D} \nabla_{\theta_t} \Delta(d_\theta(\boldsymbol{b}^x), y) \quad (2)$$

$$\phi_{t+1} = \phi_t - \sum_{(x,y)\in D} \nabla_{\phi_t}(\phi(x)) \times \nabla_{\boldsymbol{b}^x} \Delta(d_\theta(\boldsymbol{b}^x), y) \quad (3)$$

3.4 Régularisation de l'espace de représentation

Au regard du décodage proposé par plus proche voisin, l'objectif étant de garantir un espace de représentation structuré où les exemples appartenant à la même classe soient projetés dans un espace compact. Nous traduisons cet objectif en introduisant un terme de régularisation, permettant de minimiser la distance intra-classe et de maximiser la distance inter-classe. Soit les deux sous-ensembles suivants :

$$\mathcal{D}_{intra} = [((x, y), (x', y')) \in \mathcal{D}^2 | y = y']$$
$$\mathcal{D}_{inter} = [((x, y), (x', y')) \in \mathcal{D}^2 | y \neq y']$$

la fonction d'erreur à minimiser est donc la suivante :

$$J(\phi, \theta) = E_{(x,y)}\left[E_{\boldsymbol{b}^x \sim \phi(x)}\left[\Delta(d_\theta(\boldsymbol{b}^x), y)\right]\right] + \beta \sum_{((x,y),(x',y'))\in \mathcal{D}_{intra}} d(\phi(x), \phi(x')) \quad (4)$$

$$- \gamma \sum_{((x,y),(x',y'))\in \mathcal{D}_{inter}} d(\phi(x), \phi(x')) \quad (5)$$

La minimisation de la distance intra-classe est garantie par le terme 4 minimisant une distance [2] entre deux distributions générées par des exemples de la même classe. La maximisation de la distance inter-classe est elle assurée par le terme 5 de l'équation maximisant une distance entre deux exemples de différentes classes. Les variables α et β correspondent aux coefficients de la régularisation, des hyper paramètres calibrés par grid-search. Dans la suite cette régularisation sera appelée *régularisation par triplet*.

3.5 Analyse de la complexité en inférence

Les gains en termes de complexité d'inférence sont dus essentiellement à la fonction de décodage. Pour le premier procédé, le *décodage linéaire*, la complexité est similaire à celle d'un réseau multi-couche continu : le procédé d'inférence nécessite $O(cK)$ opérations pour calculer les probabilités de chaque classe pour un exemple.

Pour le décodage considérant la recherche par plus proche voisin, trouver de manière naïve le plus proche voisin a une complexité en temps en $O(kc)$ avec k le nombre de codes auxquels on compare l'exemple à retrouver, ici le nombre d'exemples en apprentissage. En pratique, la recherche de plus proches voisins dans des espaces de Hamming peut se faire de manière sous-linéaire : par exemple la méthode proposé par [NPF14] permet d'obtenir les plus proches voisins en $O(\frac{c\sqrt[2]{k}}{log_2 k})$.

Enfin, à condition que la taille de la représentation soit petite, il est possible d'énumérer la totalité des codes et ainsi de stocker l'intégralité des couples (code, classe). Dans ce cas, une fois le modèle appris, tous les codes possibles peuvent être stockés afin d'obtenir la classe correspondante. Avec une structure adaptée (table de hash), nous pouvons déduire la catégorie d'un exemple en temps constant ; en revanche le coût de stockage est exponentiel par rapport à la taille des codes, en $O(2^c)$ espace de stockage nécessaire. Dans la suite nous utilisons cette méthode afin de classifier les exemples lorsque la dimension de la représentation n'excède pas un certain seuil.

4 Expériences et évaluation

4.1 Protocole expérimental

Jeux de données Afin de valider notre approche, on se propose d'évaluer nos résultats sur trois bases de données larges échelles contenant a minima 1000 classes (voir table 1) :

- ALOI : The *Amsterdam Library of Object Images* est une collection de données contenant de petites images d'objets. Dans notre cas nous utiliserons la représentation *SIFT* de ces images.
- DMOZ : *Directory Mozilla* est une collection contenant des descriptions de pages web, avec pour chacune la classe correspondante. Nous utiliserons la représentation bag-of-word (BOW) des documents. Cette collection contient 12275 classes ; à des fins d'évaluation, nous considérerons aussi des sous-ensembles de ce jeu de données contenant chacun 1000 classes (DMOZ-1K) tirés aléatoirement.
- ImageNet : Cette collection d'images contient 1000 classes correspondant au label de l'image. Dans ce cas particulier, l'approche par réseaux convolutionnels reste l'une des plus adaptées. Pour permettre la comparaison des performances, nous utilisons les représentations correspondant à l'avant-dernière couche du modèle *resnet-152*[HZRS16], un réseau convolutionnel de 152 couches.

TABLE 1 – Composition des collections

Base de donnés	nombre de classes	nombre d'exemples
DMOZ-1K	1000	41846 ± 5255
DMOZ-12K	12275	155775
ALOI	1000	108000
IMAGENET	1000	14197122

Apprentissage et sélection de modèles Le protocole expérimental est identique pour tous les jeux de données : 80% des exemples sont sélectionnés aléatoirement afin de former l'ensemble d'apprentissage, 10% des données sont utilisées pour la validation, et les 10% restants sont conservés pour l'évaluation, où nous évaluerons le taux de bonne classification. Les expériences sont réalisées à l'aide de l'estimateur de gradient *Straight-Through* en mettant à jour les poids avec la méthode de descente de gradient adaptatif *Adam* [KB14] et en utilisant des tailles de batch de 100 à 1000 exemples. Dans chacune des expériences menées, nous considérons la fonction d'encodage ϕ comme une fonction linéaire combinée à une fonction d'activation sigmoïde. Pour l'apprentissage, la fonction de décodage est une fonction linéaire combinée à une activation softmax.

Lors de l'évaluation, nous considérerons pour le décodage les deux variantes proposées dans la section 3 : le décodage linéaire utilisant les poids appris durant l'étape d'apprentissage (noté *linear*) et le décodage via la recherche du plus proche voisin (noté

2. On considère dans le cas présent : $||\phi(x) - \phi(x')||_2^2$

NN). Nous analyserons l'influence de la régularisation par triplet en réalisant des expériences avec et sans cette régularisation.

Afin de sélectionner les meilleurs modèles et hyper paramètres, nous avons suivi le schéma classique : choisir la configuration permettant d'obtenir un taux d'erreurs minimal sur l'ensemble de validation.

Algorithmes comparés Nous comparons les résultats de notre modèle à un perceptron multi-couche similaire à notre modèle en taille de représentation, en nombre de couches et en type d'activation (activation continue). Ce réseau équivaut à un classifieur un contre tous. Nous nous comparons également à un modèle *ECOC* où les codes des classes sont choisis aléatoirement et où chaque classifieur de l'ECOC est un classifieur linéaire.

4.2 Résultats et analyse

Performances en classification La table 2 référence les résultats obtenus pour les jeux de données considérés sur les ensembles de test en pourcentage de bonne classification. En prêtant attention aux résultats obtenus, nous pouvons remarquer que les performances de notre modèle sont très proches en ce qui concerne le décodage par plus proche voisin (NN) et le décodage via la fonction linéaire (linear), ce qui montre la capacité de généralisation de l'espace de représentation engendré par notre méthode. Notre approche se révèle bien plus efficace que l'algorithme *ECOC* et a des performances comparables au perceptron multi-couche, tout particulièrement dans le cas où les représentations ont une dimension importante. L'utilisation de la régularisation permet d'augmenter la performance en classification que ce soit dans le cas du décodage linéaire et dans le cas du décodage par plus proches voisins. L'amélioration est d'autant plus importante lorsque la taille de la représentation augmente. A contrario, l'utilisation de la régularisation ne semble pas influer sur les performances du perceptron multi-couche.

Lorsque la complexité en temps est minimale (en particulier lorsque tous les codes peuvent être stockés en mémoire et que le temps de décodage est constant pour notre modèle), les résultats sont en général très dégradés ; néanmoins, notre modèle surpasse la méthode *ECOC*. Lorsque la taille des représentations grandit et que le décodage en temps constant n'est plus envisageable, notre approche surpasse dans la majorité des cas les approches comparées tout en conservant une meilleure complexité en temps.

Complexité en inférence Dans la figure 4.2, nous comparons les résultats en classification en fonction de la complexité en temps d'inférence de la fonction de décodage [3] pour notre modèle et les deux approches ECOC et perceptron multi-couche. La complexité est calculée grâce aux formules données dans la section 3 .

Le rapport complexité/precision reporté dans la partie gauche de la figure représente le taux de bonne classification de notre approche et de l'approche *ECOC* lorsque la taille des représentations est suffisamment petite pour pouvoir être stockée en mémoire. Le modèle avec la plus haute complexité (décodage linéaire) est représenté à droite. Les courbes au centre de la figure font référence à notre approche et l'approche *ECOC* lorsque la taille de la représentation ne peut plus être stockée en mémoire, c'est-à-dire quand la dimension de représentation des codes à une valeur supérieure à 30. Pour la collection DMOZ, les résultats montrent que notre approche réussit à obtenir un bon compromis complexité/précision, ayant une meilleure performance que l'ECOC et meilleure complexité que l'approche perceptron multi-couche tout en étant compétitif en termes de taux de bonne classification.

Régularisation par triplet La table 3 récapitule les effets observables sur les distances intra/inter classes lors de l'utilisation de la régularisation par triplet pour les codes obtenus sur l'ensemble d'apprentissage et sur l'ensemble de test pour la collection *DMOZ-1K*.

Les différences obtenues montrent un impact bien réel de la régularisation sur la topologie de l'espace de représentation : 1) On obtient l'effet escompté de minimisation de la distance intra-classe et de maximisation de l'inter-classe, c'est là une caractéristique importante de l'espace de représentation permettant d'obtenir une meilleure séparation des classes pour appliquer l'algorithme des plus proches voisins ; 2) on observe dans la table le nombre de codes différents associés à l'ensemble d'apprentissage. Cette mesure indique qu'il y a moins de codes différents lors de l'utilisation de la régularisation, ceci permet d'effectuer moins de comparaisons lors du calcul des plus proches voisins, permettant donc d'accélérer le processus d'inférence.

De manière plus général, l'utilisation de la contrainte supplémentaire sur la distance des codes permet dans tous les cas d'augmenter les performances, aussi bien en complexité de temps d'inférence qu'en termes de taux de bonne classification.

3. Tous les modèles présentés ont la même complexité pour la phase de codage, i.e. la projection de l'entrée sur l'espace de représentation.

TABLE 2 – Évaluation du taux de bonne classification du modèle proposé (DSNC) avec et sans régularisation, avec les deux variantes de décodage, et des deux baselines (ECOC, MLP). Les cases grisées indiquent qu'il est possible de stocker l'intégralité des codes en mémoire.

	modèle	DSNC				MLP		ECOC
dataset	code size	linear	NN	linear+Reg.	NN+Reg.		Reg.	
DMOZ-1K	12	31.772	22.425	33.156	23.492	39.204	39.716	10.738
	24	39.736	36.779	41.326	37.028	48.496	48.748	22.144
	36	42.928	41.208	45.414	42.776	51.48	51.716	27.589
	60	46.84	44.734	48.742	47.41	53.532	54.084	33.850
	100	49.164	46.807	50.984	49.888	54.954	55.658	38.212
	200	51.058	49.065	53.052	52.355	56.262	56.908	41.33
DMOZ-12k	12	15.09	15.2	15.34	15.24	20.05	20.18	3.8
	24	21.15	18.79	21.64	19.27	28.74	29.3	17.591
	36	24.83	22.21	25.95	24.17	32.14	32.05	21.71
	60	28.36	25.97	29.71	29.96	35.41	35.36	25.079
	100	30.42	27.6	31.98	33.94	37.45	37.23	27
	200	32.22	29.32	33.95	35.95	39.12	38.25	27.91
	400	31.96	33.02	33.6	38.65	36.71	39.75	28.08
ALOI	12	34.918	34.84	33.328	33.366	82.04	81.992	1.53
	24	67.92	63.66	66.064	64.19	88.174	87.27	4.21
	36	76.73	74.19	75.84	79.94	89.91	88.18	5.81
	60	83.478	83.19	82.66	81.74	92.22	89.78	9.03
	100	88.014	88.58	87.606	88.72	99.96	90.79	13.8
	200	91.288	91.88	90.542	91.26	95.15	92.41	22.4
IMAGENET	12	1.5	0.82	1.49	0.805	14.6	13.11	12.32
	24	15.6	7.705	9.01	4.595	53.19	53.46	32.42
	36	53.82	36.46	45.74	25.14	59.11	58.85	45.03
	60	60.07	48.61	63	53.655	60.83	63.66	56.5
	100	68.66	63.405	68.81	63.515	65.9	66.81	64.5

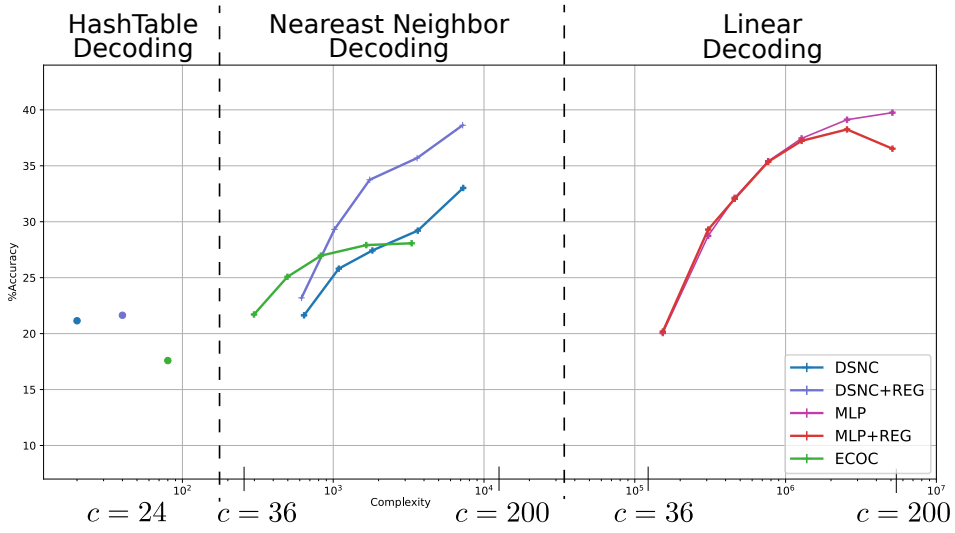

FIGURE 2 – Performances en classification en fonction de la taille des représentations/temps d'inférence pour notre modèle et les baselines

TABLE 3 – Distance intra-classe et inter-classe de l'espace des codes pour l'ensemble d'apprentissage et de test

Corpus	Distance	24		60		100		200	
		Reg	No Reg	Reg	No Reg	Reg	No Reg	Reg	No Reg
train	intra-class	1.1 ± 1	2.3 ± 1	2.8 ± 3	10.9 ± 3	8.82 ± 6	23 ± 5	18.4 ± 11	59.3 ± 10
	inter-class	11.78 ± 0	11.6 ± 0	29.4 ± 0	26.6 ± 0	47 ± 1	40.8 ± 1	94.4 ± 2	72.3 ± 2
	# codes	$8k$	$15k$	$15k$	$28k$	$24k$	$29k$	$26k$	$30k$
test	intra-class	4.8 ± 3	5.9 ± 3	12.1 ± 6	16.6 ± 6	20.1 ± 10	29.6 ± 9	37 ± 19	63.9 ± 17
	inter-class	$10.8 \pm 1.$	$10.5 \pm 1.$	27.0 ± 2	25.6 ± 2	45.0 ± 3	41.6 ± 4	90.8 ± 5	80.6 ± 9

Extension Multi-label : En plus des résultats obtenus sur les jeux de données mono-label, nous proposons une approche similaire pour le problème multi-label, où chaque exemple peut avoir plusieurs classes associées et non plus une seule. L'obtention des classes ne peut se faire simplement en regardant le plus proche voisin ; ainsi à l'instar des approches de classification multi-label basés sur la représentation des exemples, nous retenons les *k plus proches voisins* en agrégeant les vecteurs de labels retenus. Pour évaluer les performances de cette extension, nous utilisons la précision jusqu'à un rang n, *precision@n*, qui est le plus souvent utilisé dans ce contexte. On remarquera aussi que l'utilisation de la régularisation par triplet n'est pas possible telle quelle ; dans ces expériences préliminaires, nous ne considérerons pas l'utilisation d'une régularisation permettant de rapprocher les exemples similaires.

Dans le tableau 4.2, nous reportons les résultats obtenus sur la collection *mediamill* en *precision@n* (cette collection contient 101 labels). Ces premiers résultats sur des exemples possédant plusieurs labels sont prometteurs, même si les performances prédictives sont en deçà de ce que peuvent obtenir des méthodes spécialisées comme *SLEEC* [BJK+15]. De plus la recherche de plus proches voisins ne peut se limiter (voir section 4.2) uniquement au voisin le plus proche, engrangeant une complexité supérieure en pratique à l'approche mono-label.

Precision@k	SLEEC	MLP	DSNC-STE
Linear decoding			
P@1	87.4	85.2	83.5
P@3	73.45	69.8	67.6
P@5	59.17	53.5	58.4
kNN decoding			
P@1	87.4	84.9	85.2
P@3	73.45	70.3	69.8
P@5	59.17	61.3	60.6

TABLE 4 – Résultats sur Mediamill pour la classification multi-label

5 Conclusions et perspectives

Nous proposons dans cet article une nouvelle approche pour la classification multi-classe mono-label en introduisant un modèle basé sur une architecture de type réseaux de neurones stochastiques. Durant la phase d'apprentissage nous apprenons deux fonctions, la première projetant les données sur une espace de représentation binaire, la seconde projetant la représentation binaire sur l'espace des classes. Afin de permettre une exploration de l'espace des représentations, nous avons proposé l'utilisation d'un mécanisme stochastique, ce mécanisme introduisant ainsi une couche non-différentiable.

Pour l'apprentissage du réseau, nous avons proposé d'utiliser une méthode d'approximation du gradient existant dans la littérature, l'approximation de gradient *Straight-Through* qui permet une optimisation rapide est efficace de la couche non-différentiable. Avec ce mécanisme notre modèle apprend de bout en bout les deux projections simultanément. En plus de ce mécanisme, nous avons proposé une régularisation qui permet de structurer les représentations des exemples dans un espace de Hamming, celle-ci permettant de rehausser les performances en prédiction, mais aussi d'abaisser le temps d'inférence. Grâce aux propriétés des espaces discrets, notre approche est capable de prédire en temps sous-linéaire la classe associée à un exemple, et dans certains cas en temps constant relativement au nombre de classes. Les expériences menées ont permis de démontrer les bénéfices de notre approche en termes de temps d'inférence et de prédiction.

Le travail présenté n'est qu'une première étape autour de la problématique de l'apprentissage de représentation discrète dans le contexte de la classification multi-classe. Ainsi nous nous proposons d'approfondir l'investigation de cette approche pour la classification multi-label - chaque exemple peut être étiqueté avec une ou plusieurs catégories. Enfin nous planifions d'explorer/analyser la topologie des espaces de représentation afin de pouvoir explorer d'autres applications.

6 Remerciement

Cette publication est basée sur des travaux supportés par l'université KAUST *King Abdullah University of Science and Technology* Office of Sponsored Research (OSR) under Award No. OSR-2015-CRG4-2639.

Références

[APHS14] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Good Practice in Large-Scale Learning for Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3) :507–520, 2014.

[BJK+15] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 730–738. Curran Associates, Inc., 2015.

[BLC13a] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv :1308.3432*, 2013.

[BLC13b] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[BWG10] Samy Bengio, Jason Weston, and David Grangier. Label embedding trees for large multi-class tasks. In *Advances in Neural Information Processing Systems 23*, pages 163–171. 2010.

[CAG12] M. Cissé, T. Artières, and Patrick Gallinari. Learning compact class codes for fast inference in large multi class classification. In *Machine Learning and Knowledge Discovery in Databases : ECML PKDD*, pages 506–520. Springer Berlin Heidelberg, 2012.

[CB16] Matthieu Courbariaux and Yoshua Bengio. Binarynet : Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[DB95] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. of Artificial Intelligence Research*, 2 :263–286, 1995.

[DDC16] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung. Learning to hash with

binary deep neural network. In *Computer Vision – ECCV 2016 : 14th European Conference, Proceedings, Part V*, pages 219–234. Springer International Publishing, Cham, 2016.

[ELLW+15] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[GIM99] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529. Morgan Kaufmann Publishers Inc., 1999.

[HCS+16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv :1412.6980*, 2014.

[LPLY15] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. *arXiv :1504.03410*, 2015.

[NFS12] Mohammad Norouzi, David J Fleet, and Ruslan R Salakhutdinov. Hamming distance metric learning. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1061–1069. Curran Associates, Inc., 2012.

[NPF14] M. Norouzi, A. Punjani, and D. J. Fleet. Fast exact search in hamming space with multi-index hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6) :1107–1119, 2014.

[PB15] Raphael Puget and Nicolas Baskiotis. Hierarchical label partitioning for large scale classification. In *IEEE International Conference on Data Science and Advanced Analytics, DSAA*, pages 1–10, 2015.

[PKB+15] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artières, George Paliouras, Éric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Gallinari. LSHTC : A benchmark for large-scale text classification. *CoRR*, abs/1503.08581, 2015.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3) :211–252, 2015.

[SH09] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7) :969–978, July 2009.

[Wil92] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3) :229–256, 1992.

[WMY13] Jason Weston, Ameesh Makadia, and Hector Yee. Label partitioning for sublinear ranking. In *Proc. of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 181–189, 2013.

[WTF09] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems 21*, pages 1753–1760. 2009.

[WZS+17] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, - to appear, 2017.

[ZC13] Guoqiang Zhong and Mohamed Cheriet. Adaptive error-correcting output codes. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1932–1938. AAAI Press, 2013.