Rapport - Recherche d'informations visuelles

René Traoré, Rémy Portelas - RI

December 21, 2017

Introduction

Ce document décrit les travaux réalisés dans le cadre de la série de TMEs du cours de Recherche d'Information portant sur la recherche d'image.

Nous avons commencé par nous familiariser avec le squelette de code Java fournie que nous avons augmenté pour permettre la génération d'un dataset de descripteurs d'images de type Bag of Words (BoW). Afin de pourvoir utiliser ce dataset pour différentes tâches d'apprentissage, nous avons implémenté une méthode d'apprentissage supervisée générique, basée sur la descente de gradient.

Nous avons par la suite utilisé ce dataset et ce framework générique d'apprentissage pour implémenter et comparer deux tâches de classification multi-classes. La première méthode utilise un loss 0-1 classique tandis que la seconde se sert de l'ontologie d'ImageNet pour définir un loss hiérarchique.

Enfin, nous avons une fois de plus utilisé notre framework pour apprendre à retourner un ordonnancement de notre dataset d'images pour une requête donnée. Nous clôturerons ce rapport par la présentation des performances de ce modèle d'apprentissage de ranking.

Contents

1 Indexation visuelle - Apprentissage structuré (TME 7)

Dans ce TME l'objectif était de prendre en main l'architecture de code JAVA fournie et de l'augmenter pour charger un sous-dataset d'images comprenant 9 classes, faisant partie du célèbre dataset de benchmarking *ImageNet*.

1.1 Génération des indexes visuels

Pour pouvoir exploiter notre base d'images nous avons implémenté une fonction permettant de charger en mémoire les descripteurs d'images fournies, nommé Bag Of Words (BoW). L'utilisation de BoWs permet une représentation vectorielle concise de nos images. On peut voir sur la figure (1) le BoW d'une de nos image, sous forme d'histogramme. On peut voir que les bins sont correctement créés , la normalisation l_2 appliqué sur le BoW le rend plus expressif. Les valeurs de cet histogramme peuvent être exprimé comme des pourcentages dont la somme serait 1.

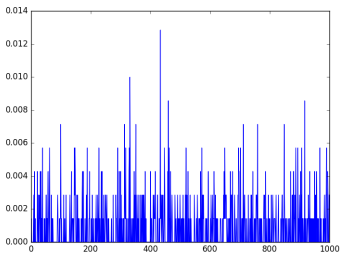


Figure 1: Visualisation de la création d'un BoW normalisé, avant PCA.

Pour réduire le nombre de dimensions de nos BoWs et ainsi réduire la complexité de calculs de nos futures algorithmes d'apprentissage, un PCA (Principal Component Analysis) a été effectué sur le dataset, nous permettant de passer de 1000 à 250 dimensions.

1.2 Algorithme d'apprentissage structuré générique

Une fois le dataset prêt nous nous sommes attaqués à l'implémentation d'un algorithme d'apprentissage structuré générique.

Plus précisément, nous avons implémenté un algorithme d'apprentissage par descente de gradient stochastique, c'est à dire dont la mise à jour des poids du réseau a lieu après chaque exemple d'entraînement. La vitesse de convergence, ou learning rate, est gérée par le paramètre η tandis que le terme λ contrôle le niveau de régularisation du réseau.

La méthode de prédiction du réseau, nommé *loss-augmented inference*, constitue une borne supérieure à notre problème d'apprentissage complexe (non-convexe), son utilisation permet de ramener le problème à une optimisation de fonction convexe, bien plus abordable en terme de temps de calcul.

1.3 Conclusion

Dans ce TME nous avons mis en place les outils nécessaires à l'élaboration de méthodes d'apprentissage sur images. La validité de notre implémentation pourra être constatée dans la suite de se rapport, puisque tous nos modèles qui seront exposés se serviront de ce framework.

2 Classification multi-classes et hiérarchique (TME 8)

2.1 Classification multi-classes et évaluation

À l'aide de notre framework général d'apprentissage nous avons mis en place un modèle de classification multi-classes. Plus précisément, nous avons implémenté sa joint-feature map ϕ et son loss Δ.

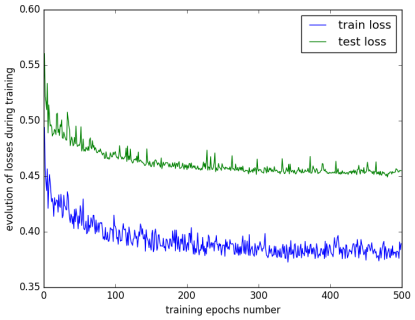
ϕ place simplement le vecteur BoW après PCA à l'emplacement correspondant à y_i dans un grand vecteur de taille $c * d$ (c le nombre de classes et d la dimension du BoW après PCA). L'inférence pourra alors simplement se faire par un produit cartésien entre ϕ et le vecteur de poids w.

Δ est un 0-1 loss classique retournant 0 pour une prédiction correcte, 1 sinon.

Notre classe *MultiClassClassif* permet de tester notre implémentation. Les résultats présentés dans cette section on été réalisés avec $\lambda = 10^{-6}, \gamma = 10^{-2}$ sur 500 époques d'entraînement.

On peut voir sur la figure (2a) que notre modèle apprend, le loss 0-1 diminue bien au cours de notre entraînement, passant d'environ 0.48 à 0.37 sur les données d'entraînement. On constate aussi que notre réseau parvient à généraliser correctement puisque le loss sur le jeu de test diminue également.

Concernant les performances de classification, on peut voir que la matrice de confusion produite sur le jeu de test (fig (2b)) montre de bonnes performances de prédictions à l'exception de la classe **tree_frog** qui a principalement été confondu avec la classe **european_salamander**. Ce genre d'erreurs de prédiction n'est pas alarmant car ces deux classes sont sémantiquement très proche, il est bien plus normal de confondre une grenouille avec une salamandre qu'avec un taxi. On remarquera d'ailleurs qu'une grande partie des erreurs de classification pour chaque classe sont concentrées dans dans les autres classes de sa "super-classe" (d'où l'apparition de "carrés" dans la matrice).



(a) Évolution de notre loss sur 500 époques d'entraînement stochastique avec *loss-augmented inference*. On peut voir que 200 itérations suffisent pour faire converge notre modèle.

(b) Matrice de confusion obtenue en test après entraînement. Les labels utilisés, de gauche à droite et de haut en bas sont les suivants: **taxi,ambulance,minivan, acoustic_guitar,electric_guitar,harp, wood-frog,tree-frog,eu._fire_salamander**

Figure 2: Analyse de convergence et de performance concernant notre modèle d'apprentissage structuré avec un loss 0-1.

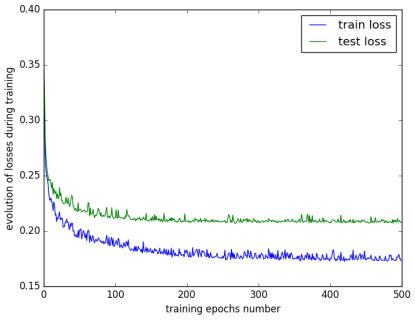
2.2 Classification hiérarchique et évaluation

Nous avons augmenté notre modèle en implémentant une nouvelle fonction Δ de loss hierarchique. Cette fonction utilise un modèle du langage permettant d'attribuer une valeur de similarité entre deux classes, comprise entre 0 et 1 une fois normalisé. En utilisant cette valeur on apporte plus de granularitée par rapport un loss 0-1.

On peut voir que notre modèle converge bien avec ce type de loss (fig (3b)), passant de 0.30 à 0.17 sur le jeu de données d'entraînement. Le loss hiérarchique n'apparaît pas détrimentale pour les capacitées de généralisation du réseau puisque le loss de test converge lui aussi.

Côté performance, ce modèle donne une matrice de confusion (figure (3b)) à la qualitée globale comparable avec le modèle précédent. On peut voir qu'on

a cette fois-ci de meilleurs résultats pour la classe **wood_frog** tandis que les performances pour la classe **taxi** ce sont légerement dégradés. Si on regarde attentivement la matrice, on peut remarquer que les "carrés" de super-classes sont légèrement plus rouges (donc valeur plus élevées) avec un loss hiérarchique. Ceci est dû à la définition de ce loss qui va moins pénaliser une erreur de classification si la mauvaise classe est sémantiquement proche de la classe cible (c-à-d qu'elle fait partie de la même super-classe).



(a) Évolution de notre loss sur 500 époques d'entraînement stochastique avec *loss-augmented inference* et un loss hiérarchique. On peut voir que 200 itérations suffisent pour faire converger notre modèle.

(b) Matrice de confusion obtenue en test après entraînement avec loss hiérarchique. Les labels utilisés sont les mêmes que la matrice précédente (fig (2b)).

Figure 3: Analyse de convergence et de performance concernant notre modèle d'apprentissage structuré avec un loss hiérarchique.

2.3 Conclusion

Dans ce TME nous avons éprouvé notre framework générale d'apprentissage en implémentant, testant et comparant deux variantes d'apprentissage structuré appliquées à une tâche de classification d'image. La prochaine et dernière étape de cette série de TMEs aura pour objectif de se rapprocher d'un problème de recherche d'information en implémentant un modèle d'apprentissage d'ordonnancement.

3 Apprentissage d'ordonnancement (TME 9)

3.1 Ranking Structuré

Pour mettre en place notre méthode de ranking structuré nous avons implémenté une classe *RankingInstantiation* définissant les fonctions $\Psi(x, y)$ et $\Delta(y_i, y)$ qui seront utilisés par le framework d'apprentissage structuré.

$\Delta(y_i, y)$ est une fonction de coût qui n'utilise pas y_i mais renvoi simplement $1 - AP(y)$, avec AP la précision moyenne définie comme l'aire sous la courbe de rappel précision.

le calcul de $\Psi(x, y)$ est ici délicat puisque l'on ne peut pas explorer exhaustivement l'espace Y des ordonnancement possibles. Nous avons utilisé une méthode optimisée résumée par l'algorithme (1), permettant de créer le vecteur de sortie Ψ en ne traitant qu'une seule fois tous les couples (exemple positif, exemple négatif) de la base d'images considérées.

Algorithm 1 $\Psi(x, y)$ ranking

```
1: procedure Ψ(x, y)                          ▷ x liste de BoWs et y sortie de ranking
2:     psi_output ← init(x.size_elt)          ▷ vecteur de sortie de la taille d'un BoW
3:     rank_list ← y.ranking
4:     label_list ← y.labels
5:     for img with pos labels in rank_list do          ▷ positive = classe query
6:         for img with neg labels in rank_list do      ▷ negative = autres classes
7:             if rank_pos < rank_neg then              ▷ image + bien classé avant -
8:                 psi_output ← psi_output + 1 ∗ (bow_pos − bow_neg)
9:             else                                     ▷ image + mal classé après -
10:                psi_output ← psi_output − 1 ∗ (bow_pos − bow_neg)
```

La classe *RankingStructModel* fut elle aussi implémentée, elle contient une surcharge de la méthode *predict* et *loss_augmented_inference*. Nous avons aussi accéléré l'algorithme d'apprentissage en pré-calculant $\Psi(x, y_i)$ dans la méthode *train* de notre *SGDTrainer*, comme suggéré dans la question bonus de ce TME.

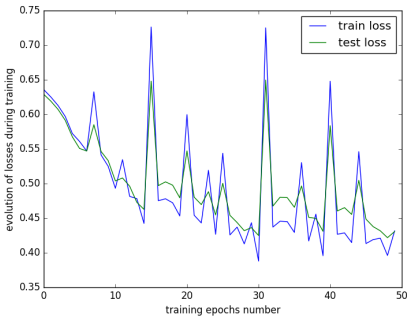
3.2 Évaluation de l'ordonnancement

Nous avons créé une classe *Ranking* nous permettant de tester notre modèle d'apprentissage structuré. Nous allons pouvoir tester les performances de notre modèles sur 9 queries correspondant aux 9 classes de notre base de d'images tirée d'*ImageNet*.

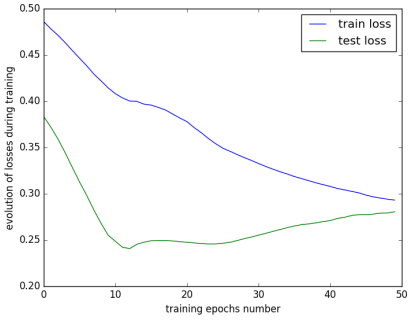
On peut voir que notre modèle apprend correctement en étudiant quelques courbes de loss suivant les queries (fig 4).

Pour chaque query, les courbes de précision rappel en train et en test sont visible dans l'annexe A (fig 5 et 6). Le modèle utilisé pour chaque query est celui qui obtient le loss le plus faible en test (parmi les 50 versions issues des 50 époques d'entraînements). On peut voir qu'on atteint des performances similaires (légèrement plus faibles) aux exemples de résultats visible sur l'énoncé du TME, avec par exemple une AP de 57% en test sur la query **ambulance** contre 65.3% sur le TME. Nous obtenons une AP moyenne de 59.85% en train et 52.32% en test.

Quand on compare notre AP pour de l'apprentissage d'ordonnancement à nos performances de classification (cf matrice de confusion TME 8 fig 2b), on se rends compte d'une certaine similaritée. On peut voir que pour ses deux méthodes le traitement d'images de taxis est plus compliqué que les images d'ambulances et de minivans. Il est également plus facile de traiter les images de harpes plutôt que les images de guitares acoustiques et électriques.



(a) Query **minivan**: un loss certes bruité mais convergeant au cours de l'entraînement.



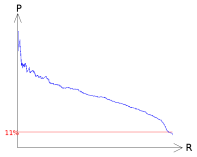
(b) Query **eu. fire salamander**: un bel exemple de convergence avec over-fitting après 10 époques.

Figure 4: 2 exemples de courbes de loss pour notre modèle d'ordonnancement structuré. Nous avons utilisé les paramètres recommandés dans le TME ($lr = 10$,$reg = 10^{-6}$,50 époques).

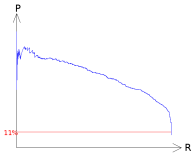
3.3 Conclusion

Dans ce TME nous avons une fois de plus étendu notre framework d'apprentissage par descente de gradient, cette fois pour une tâche d'ordonnancement structuré. Nous avons démontré la validitée de notre implémentation par la présentation de performances d'ordonnancements et de classifications similaires à celles présentés dans les énoncés de TMEs.

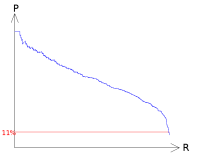
A Ranking structuré: AP en train et test



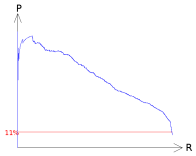
(a) Taxi AP: 0.45



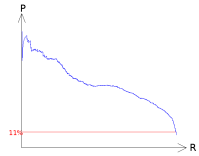
(b) Ambulance AP: 0.63



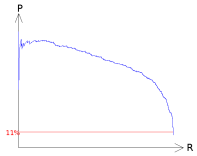
(c) Minivan AP: 0.60



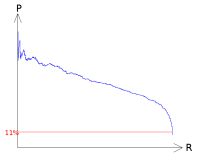
(d) Acoustic Guitar AP: 0.62



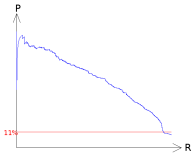
(e) Electric Guitar AP: 0.57



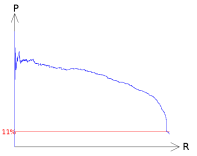
(f) Harp AP: 0.75



(g) Wood Frog AP: 0.57

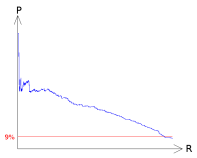


(h) Tree Frog AP: 0.60

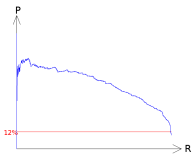


(i) European Fire Salamander AP: 0.60

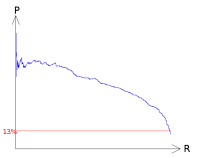
Figure 5: AP (Average Precision) de nos ordonnancements par ranking structuré en Train. Nous obtenons une AP moyenne de 0.5985 .



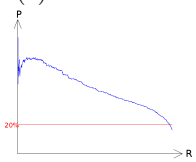
(a) Taxi AP: 0.35



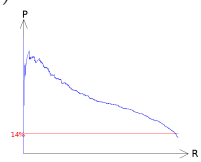
(b) Ambulance AP: 0.57



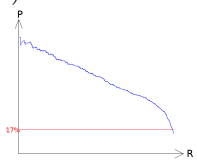
(c) Minivan AP: 0.58



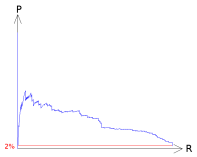
(d) Acoustic Guitar AP: 0.57



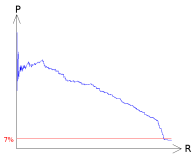
(e) Electric Guitar AP: 0.48



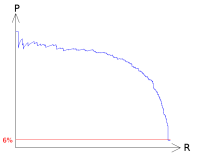
(f) Harp AP: 0.66



(g) Wood Frog AP: 0.25



(h) Tree Frog AP: 0.50



(i) European Fire Salamander AP: 0.76

Figure 6: AP (Average Precision) de nos ordonnancements par ranking structuré en Test. Nous obtenons une AP moyenne de 0.5232 .