RDFIA TP3 : Classification par *feature extraction* avec CNN pré-entrainé

Taylor Mordan, Thomas Robert, Matthieu Cord

23 novembre 2016

Pour les TP sur les réseaux de neurones, vous devrez rendre :

- Un compte rendu à la fin de chaque séance (manuscrit ou par mail à `taylor.mordan@lip6.fr` ou `thomas.robert@lip6.fr` selon votre encadrant de TP, les compte-rendus en retard ne seront pas acceptés). Ce compte-rendu devra décrire le travail effectué pendant la séance et les résultats importants obtenus.
- Un compte rendu global une semaine après la dernière séance (par mail), dont le but est de reprendre au propre le travail effectué pendant l'intégralité des séances de TP, en prenant du recul sur ce qui a été fait. Ce compte rendu représentera la majorité de la note de TP.

Les données et une version numérique du sujet sont accessibles
à l'adresse `http://webia.lip6.fr/~robert`

1 Objectifs

L'objectif de ces TP est de reproduire la méthode décrite dans Chatfield et al. (2014) [1] en l'utilisant sur la base *15 Scene*. On s'intéressera particulièrement à la stratégie visant à extraire des descripteurs *deep* grâce à un réseau pré-entrainé (en utilisant MatConvNet vu au TP précédent) et à les utiliser dans un schema standard de classification avec des SVM lineaires.

L'objectif du TP est donc d'apprendre un modèle de classification d'images performant sur la base de données *15 Scene*.

2 Principe de la démarche

Principe général Le principe de la démarche est assez simple et se compose de deux parties : on utilisera d'abord un réseau de neurones pré-appris pour faire de la *feature extraction* à la suite de laquelle on entraine un modèle de classification.

La première partie peut se voir comme une alternative à l'approche *Bag of Words* (BoW). L'objectif est d'obtenir pour chaque image une représentation vectorielle de l'image décrivant son contenu, que l'on pourra par la suite utiliser pour accomplir diverses tâches, en particulier la classification d'image.

Le principe de l'approche de *feature extraction* est de produire cette représentation des images

à partir d'un réseau de neurones convolutionnel appris pour résoudre un problème autre que celui auquel on s'intéresse (par exemple dans notre cas, un réseau appris pour une tâche de classification des images de la base ImageNet contenant 1,2 million d'images réparties en 1000 classes). Pour produire la représentation d'une image, on conservera la sortie d'une des couches intermédiaires du réseau de neurones que l'on choisira.

Principe pour notre TP Dans le cas de notre TP, nous utiliserons le réseau *CNN-F* étudié au TP précédent, et nous représenterons chaque image par la sortie vectorielle de la couche `relu7`. On apprendra ensuite un classifieur SVM pour chaque classe du jeu de données *15 Scene* pour répondre à notre tâche de classification.

3 Travail à réaliser

3.1 Préparation

1. Télécharger et extraire les données. Le fichier `15SceneSplit.mat` contient :
   — la liste des noms des images en *train* et *test* dans `X_[train|test]_path`
   — les labels associés dans les vecteurs `y_[train|test]` encodés comme un numéro de classe entre 1 et 15
   — les noms des classes correspondantes dans `classes`
2. Télécharger les fonctions fournies
3. Lancer Matlab avec la commande :
   `LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libstdc++.so.6 matlab`
4. Ajouter le dossier `libsvm/matlab` à votre path
5. Initialiser MatConvNet comme vu au TP précédent (à faire à chaque fois)

3.2 Extraction des *features*

Pour extraire une *feature* pour une image, il faut :

— Charger l'image à partir de son chemin dans `X_[train|test]_path`
— Pré-traiter l'image :
  — Dupliquer trois fois le canal gris pour simuler une image à trois canaux RGB
  `img = repmat(img, 1, 1, 3);`
  — Appliquer les pré-traitements classiques du réseau vus au TP précédent
— Faire passer l'image dans le réseau de neurones
— Récupérer la sortie de la couche à laquelle on veut extraire les *features*
— Appliquer une normalisation L2 au vecteur

Pour cela, s'inspirer du code du TP précédent.

À partir de cette procédure, constituez les matrices $\mathbf{X}_{train}$ et $\mathbf{X}_{test}$ où on stockera les *features* par ligne.

3.3 Apprentissage d'un classifieur

À partir des *features* $(\mathbf{X}_{train}, \mathbf{X}_{test})$ et des *targets* (y_{train}, y_{test}) associées, nous allons apprendre un classifieur multi-classe (constitué de plusieurs classifieurs binaires SVM linéaires) avec ces éléments. Ce travail ayant déjà été réalisé dans un TP précédent, vous utiliserez les fonctions suivantes :

- `function model = train_classifier(X_train, y_train, C)`
  Permet d'entrainer un classifieur multi-classe (ensemble de SVM linéaires en *one-vs-all*) sur les données $(\mathbf{X}_{train}, y_{train})$ avec l'hyper-paramètre C donné.
- `function [y_hat, Y_hat_scores] = predict_classifier(model, X_test)`
  Permet d'obtenir les prédictions pour un ensemble de données $\mathbf{X}_{test}$ à l'aide du modèle précédemment appris. Les prédictions sont retournées sous deux formats : le numéro de la classe prédite pour chaque donnée dans `y_hat` (un vecteur comparable à y_{train} ou y_{test} donc) et les scores d'appartenance de chaque donnée à chaque classe dans `Y_hat_scores`.

Entrainez un classifieur multi-classe sur la base d'apprentissage et evaluez son score de classification (*accuracy*) sur la base de test en utilisant les *features deep* extraites précédemment. Comparez votre résultat aux résultats obtenus avec les *BoW*.

3.4 Aller plus loin

Une fois ce schéma de classification opérationnel, vous pourrez étudier l'effet de plusieurs choix sur les performances ou le temps d'apprentissage. Quelques exemples possibles sont listés ci-dessous mais vous êtes libres d'en trouver d'autres !

- Changer la couche à laquelle sont extraites les *features*. Quelle est l'importance de la profondeur de la couche ? Quelle est la taille de la représentation et que cela change-t-il ?
- Essayer d'autre réseaux pré-appris, en particulier les modèles *CNN-M* ou *CNN-S* (voir l'article [1]) disponibles. Quelles sont les différences entre ces réseaux ?
- Régler la valeur du paramètre C pour améliorer les performances.
- Étudier des méthodes de réduction de dimension avant la classification et leurs impacts sur les performances et le temps d'exécution.

Références

[1] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details : Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014.