Note de Synthèse

Vision et reconnaissance d'objet : Comparaison d'algorithmes sur MNIST

Demonet, Grasset, Puppo & Rouhard

5 Mai 2015

Introduction

Pour améliorer ses services, une banque veut automatiser la lecture des montants des chèques pour permettre un transfert plus rapide des fonds sur les comptes de ses clients. Elle décide donc de construire un programme qui lit les chiffres manuscrits. Pour ce faire, elle va implémenter des algorithmes de *machine learning* et choisir le plus performant selon un certain nombre de critères comme le taux de réussite ou le temps de lecture.

Pour écrire un programme qui permet de "lire" ces chiffres, nous pouvons nous servir de MNIST. MNIST est une base de données de chiffres manuscrits, elle est composée d'un corpus de 60.000 chiffres dont voici un extrait :

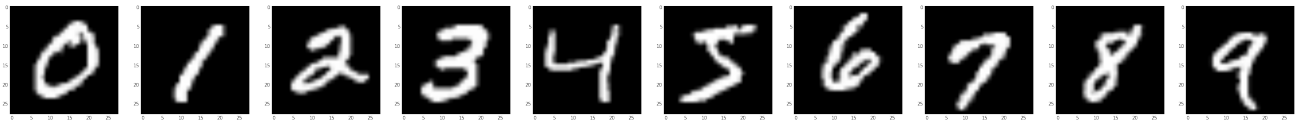


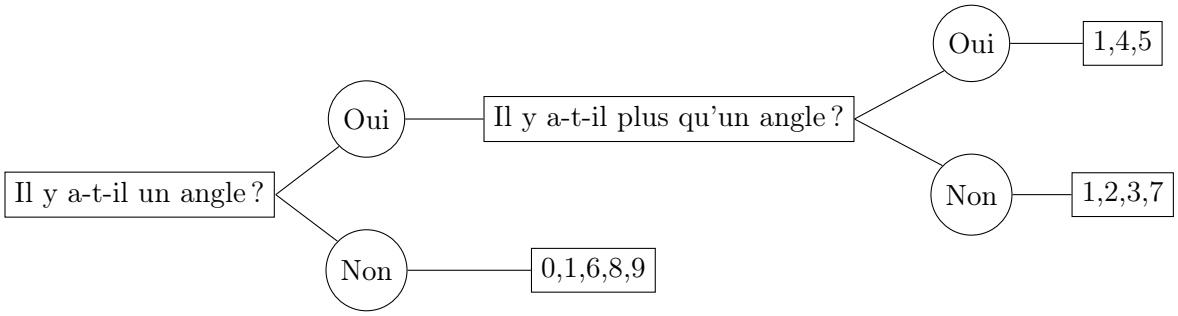
FIGURE 1 – Quelques chiffres de MNIST

Elle est également composée d'un second ensemble de 10.000 chiffres qui permettent de tester les performances des différents algorithmes. Il nous faut donc essayer de bien lire le maximum de ces 10.000 chiffres pour que la banque puisse lire les montants de ses chèques.

1 Comment lire de manière automatique ?

Pour pouvoir lire de manière automatique, il faut utiliser des algorithmes d'apprentissage machine supervisé. Ce type d'algorithme fonctionne selon le principe suivant : à partir d'une base d'entraînement dont nous connaissons les labels (le chiffre associé à l'image), nous entraînons nos algorithmes à identifier les informations que nous ne connaissons pas. C'est-à-dire qu'à partir des 60.000 chiffres déjà lus, dont le label est connu, nous allons créer une fonction qui à une image associe un label.

Par exemple, nous avons utilisé sous plusieurs déclinaisons des arbres de décision qui illustrent bien le principe général de l'apprentissage supervisé. Nous trouvons des critères qui distinguent le mieux différents groupes d'information. Par exemple, nous pouvons diviser nos chiffres en deux catégories : ceux qui ont un angle et ceux qui n'en ont pas. Le 1 qui peut avoir un angle ou pas et sera donc dans les deux classes, ça donne donc l'arbre de décision suivant.



Une fois que nous avons notre arbre, nous lui donnons une nouvelle image et il va vérifier les critères qui lui servent à discriminer pour dire quel est le chiffre sur l'image et ainsi nous avons une prédiction.

La plupart des algorithmes d'apprentissage supervisé dépendent de paramètres qui influencent grandement leurs performances. Une attention toute particulière doit donc être attachés au calibrage de ces paramètres. Un mauvais choix de ces paramètres comporte deux risques :

— Un sous-apprentissage : de trop faibles performances sur l'ensemble de test en raison d'un algorithme trop peu complexe.
— Un sur-apprentissage : de trop faibles performances sur l'ensemble de test en raison d'un algorithme qui colle trop à l'ensemble d'entraînement et qui manque donc de capacités de généralisation.

2 Quelles méthodes utiliser pour commencer ?

Pour commencer, cette banque peut utiliser un certain nombre d'algorithmes "simples". Ici, nous entendons par simple le fait que ces algorithmes sont rapides à la fois à entraîner et pour lire de nouvelles images. Dans cette famille, nous pouvons mettre trois types d'algorithmes :

— Un perceptron : un classificateur linéaire qui cherche des hyperplans séparateur
— Un arbre de décision : consistant à construire un ensemble de choix binaires successifs de critères aboutissant à une partition des données
— Une régression logistique multinomiale : une généralisation de la régression logistique classique à plusieurs classes,

Si ces trois algorithmes demandent des puissances comparables, ils fournissent en revanche des résultats très différents. Le perceptron et l'arbre de décision lisent assez mal, en effet, ils ne sont capable de lire que 86% et 87% des chiffres manuscrits et ils sont très inégaux sur la lecture des différents labels. Par exemple, le perceptron n'arrive à bien lire que 65% des chiffrés classés 5. Cette situation est très problématique pour la banque, heureusement, elle peut faire mieux avec la régression logistique multinomiale qui lit 92% des chiffres et au moins 86% pour chaque label.

3 Des méthodes plus complexes

Ensuite, si la banque a plus de temps ou plus de puissance elle peut utiliser d'autres méthodes qui sont plus coûteuses en temps de calcul. Dans cette catégorie, nous avons utilisé en particulier quatre algorithmes :

— Un algorithme de séparation en grande dimension (les machines à vecteur de support), qui généralise les classificateurs linéaires à l'aide d'une transformation implicite des données, le tout via des fonctions appelées noyaux qui simulent un produit scalaire dans le grand espace.
— Un algorithme basé sur la mémoire (les K plus proches voisins) : on projette au milieu des 60.000 images d'entraînement nos nouveaux points et on regarde le label des plus proches voisin au sens de la norme euclidienne. Ici, on prédit le label en fonction des trois plus proches voisins.
— Un algorithme d'agrégation (une forêt aléatoire), méthode de décision collective pris en regard des résultats fournis par un certain nombre d'arbres de décision utilisant chacun un échantillon de données tirées aléatoirement. L'idée est d'éliminer par les tirages aléatoires et par le vote l'influence des données aberrantes qui peuvent fausser les résultats de l'arbre de décision.
— Un algorithme de *boosting* adaptatif (AdaBoost) qui sélectionne de manière successive un grand nombre de classificateurs faibles en fonction de leur capacité à corriger les erreurs de prévision commises par leurs prédécesseurs et agrège leurs prévisions de manière pondérée.

Ces quatre méthodes ont de très bons scores. Les forêts aléatoires, les SVM (Machine à vecteur de support) sont des méthode modérément longue à entraîner mais qui présente l'avantage de prédire très vite le label d'une nouvelle image. Leurs scores respectifs sont de 96% et 98%. Ces deux méthodes semblent bien répondre aux problématiques de la banque, elles permettent en effet une fois leur entraînement réalisé de lire en temps réel les chiffres.

L'algorithme des K plus proches voisin est lui très long pour prédire une nouvelle valeur même s'il les prédit bien, en effet il arrive a lire 97% des manuscrits. Néanmoins, dans le cadre de la problématique qu'a cette banque, cet algorithme ne serait pas très utile.

Enfin, l'Adaboost sur des arbres de décision de profondeur 14 permet d'atteindre le score de 99% d'exactitude. Cette algorithme est très long à entraîner (plusieurs heures) mais a l'avantage de prédire très vite le label de nouvelles images.

4 La transformation de données

La banque voudrait encore améliorer ses performances sur la lecture automatique des chiffres manuscrits. Pour ce faire, il lui faut augmenter la qualité de ses données, il faut qu'elle les transforme pour trouver de nouvelles distinctions ou pour pouvoir mieux comparer ses images. Par exemple, une méthode élémentaire de transformation de features est de centrer toutes les images. Avec cette méthode, on peut comparer non seulement des formes mais aussi des intensités de couleur dans certaines zones de l'image.

La principale transformation de donnée que nous avons réalisée est un histogramme de gradient orienté. Cette méthode permet de détecter des formes, de tracer des contours. En combinant plusieurs histogrammes de gradient orienté différents, on arrive à améliorer l'efficacité de nos algorithmes de manière significative.

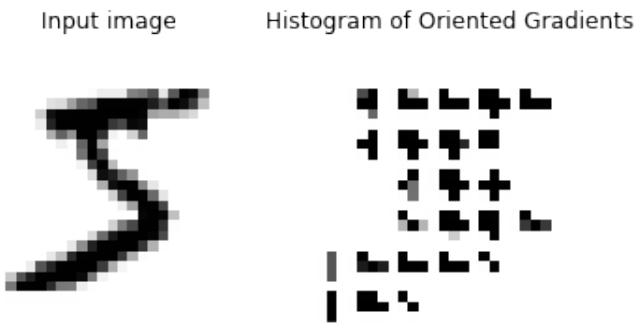


FIGURE 2 – Une image de MNIST et son histogramme de gradient orienté

Ce type d'opération consiste à augmenter la dimension de chaque image. La banque peut utiliser cette transformation pour entraîner une machine à vecteur de support à noyau linéaire pour passer de 91% à 98% de chiffres prédits. On se retrouve ainsi avec un algorithme qui s'entraîne très vite et qui prédit très vite avec un bon score ; néanmoins, le temps de transformation des données peut être assez long et doit s'appliquer à chaque chiffre à prédire.

5 Conclusion

Malgré un certain nombre de problème de calculs sur les machines distantes et de complexité en temps, nous avons pu comparer les performances d'un grand nombre d'algorithmes sur la base de données MNIST.

Nos performances restent en dessous de l'état de l'art [1], mais en se rappelant que le score moyen d'un humain sur MNIST est de 99.7%, nos performances restent honorables puisque que nous arrivons à nous rapprocher de ce seuil critique.

1. voir `http://yann.lecun.com/exdb/mnist/`