Classification d'images

Abouzaid Mehdi; Bonhomme Emma; Questel Antoine

Objectif

Dans la classification d'images, une image est classée en fonction de son contenu visuel. Par exemple, on regarde si elle contient un avion ou pas. Une application importante est la recherche d'images avec un contenu visuel similaire dans un jeu de données d'images.

Le but du projet est d'acquérir une expérience pratique de base en classification d'images. Il comprend: 1. la formation d'un classifieur visuel pour quatre classes d'images différentes (avions, motos, personnes et le reste) 2. évaluer les performances du classifieur en calculant une courbe de rappel de précision 3. modification de la représentation visuelle utilisée pour le vecteur de caractéristiques et de la carte de caractéristiques utilisée pour le classifieur;

Partie 1: Former et tester un classifieur d'images

Étape A: Préparation des données

Les données fournies sont constituées d'images et de vecteurs de caractéristiques pré-calculés pour chaque image. Les images JPEG se trouvent dans données/images. Les données consistent en quatre classes d'images au total : des avions, des motos, des personnes et des images "background", c'est-à-dire des images ne contenant pas aucun de ces objets. Au stade de la préparation des données, ces données sont divisées en:

	Avion	**Moto**	**Personne**	**Background**
Entraînement	112	120	1025	1019
Test	126	125	983	1077
Total	238	245	2008	2096

Le vecteur de caractéristiques consiste en des caractéristiques SIFT calculées sur une grille régulière sur l'image ("SIFT dense") et un vecteur quantifié en mots visuels. La fréquence de chaque mot visuel est ensuite enregistrée dans un histogramme pour chaque mosaïque d'une séparation spatiale. Le dernier vecteur de caractéristiques de l'image est une concaténation de ces histogrammes.

Nous allons commencer par former un classifieur pour les images contenant des avions qui sont répertoriées dans les fichiers data/aeroplane_train.txt et data/aeroplane_val.txt

Étape B: Former un classifieur pour les images contenant des avions

Les images d'entraînement en avion seront utilisées comme éléments positifs et les images d'arrière-plan, en tant que négatifs. Le classifieur est une machine à vecteur de support linéaire (SVM). Entraînez le classifieur en suivant les étapes décrites dans le code.

Nous allons d’abord évaluer qualitativement le fonctionnement du classifieur en l’utilisant pour classer toutes les images d’entraînement. On affiche la liste classée à l'aide de la fonction displayRankedImageList fournie.

Nous utilisons la fonction displayRelevantVisualWords pour afficher les patchs d'image qui correspondent aux mots visuels que le classifieur pense être les plus liés à la classe.

Étape C: Classer les images de test et évaluer les performances

On applique maintenant le classifieur appris aux images de test. Là encore, nous pouvez examiner les performances qualitatives en utilisant le score du classifieur pour classer toutes les images testées. On note que le terme de biais n'est pas nécessaire pour ce classement mais uniquement le vecteur de classification w.

On mesure ensuite la performance de récupération de manière quantitative en calculant une courbe Precision-Recall, la "Precision" étant la proportion d'images obtenues qui sont positives et le "Recall" la proportion d'images positives qui sont obtenues

La courbe Précision-Rappel est calculée en faisant varier le seuil sur le classifieur (de haut en bas) et en reportant les valeurs de précision par rapport au rappel pour chaque valeur de seuil. Afin d'évaluer les performances d'extraction par un nombre unique (plutôt que par une courbe), la précision moyenne (AP, aire sous la courbe) est souvent calculée.

Étape D: Faire apprendre un classifieur pour les autres classes et évaluer ses performances

On répète maintenant les étapes B et C pour chacune des deux autres classes: motos et personnes. Pour ce faire, on réexécute le script exercise1.m après avoir modifié le jeu de données chargé au début de l’étape A. Dans chaque cas, on enregistre les mesures de performance.

Étape E: Varier la représentation de l'image

Jusqu'à présent, le vecteur de caractéristique d'image a utilisé la mosaïque spatiale. Maintenant, nous allons «désactiver cette option» et voir comment les performances changent. Dans cette partie, l'image sera simplement représentée par un seul histogramme enregistrant la fréquence des mots visuels (sans tenir compte de leur position). Ceci est une représentation de sacs de mots visuels.

Un histogramme spatial peut être reconverti en un histogramme simple en fusionnant les mosaïques.

On modifie le code pour activer la partie du code qui le fait. Ensuite, on évalue les performances du classifieur sur les images test.

Étape F: Varier le classifieur

Jusqu'à présent, nous avons utilisé un SVM linéaire, traitant les histogrammes représentant chaque image sous forme de vecteurs normalisés selon une norme euclidienne unitaire. Nous allons maintenant utiliser un classifieur de noyau Hellinger, mais au lieu de calculer les valeurs du noyau,

nous allons calculer explicitement la mappe de caractéristiques, de sorte que le classifieur reste linéaire (dans le nouvel espace de fonctions). La définition du noyau de Hellinger (également appelé coefficient de Bhattacharyya) est la suivante:

k(h,h')= somme(racine(h(i)h'(i))) où h et h 'sont des histogrammes normalisés.

Donc, en fait, tout ce qui est impliqué dans le calcul de la carte des caractéristiques est de prendre la racine carrée des valeurs de l'histogramme et de normaliser le vecteur résultant en une norme euclidienne unitaire.

On modifie le code de sorte que la racine carrée des histogrammes soit utilisée pour les vecteurs de caractéristiques.

Il est avantageux de conserver le classifieur linéaire plutôt que d'utiliser un noyau non linéaire. Quand on supprime l'étape de normalisation L2, cela affecte la performance car les histogrammes sont normalisés L1 par construction.

On supprime maintenant l'étape de normalisation L2 du noyau linéaire.

Remarque: lors de l'apprentissage du SVM, pour gagner du temps d'entraînement, nous ne modifions pas le paramètre C. Ce paramètre influence l'erreur de généralisation et doit être appris sur un jeu de validation lorsque le noyau est modifié.

Étape G: Varier le nombre d'images d'entraînement

Jusqu'à présent, nous avons utilisé toutes les images disponibles. Maintenant, nous éditons le code pour utiliser 10% et 50% des données d'apprentissage. Cela correspond à la variable fraction que l'on modifie (0.1 puis 0.5).

La performance est «saturée» si toutes les images d'entraînement sont utilisées, le fait d'ajouter plus d'images d'entraînement ne donne pas d'amélioration. Il faut plutôt faire varier le paramètre C du SVM.