Kernel Methods for Machine Learning: Image classification data challenge Solo Team: Mehdi Zemni

Mehdi Zemni
CentraleSupelec
`mehdi.zemni@student-cs.fr`
Code: https://github.com/zmehdiz97/Kernel-methods

Abstract

In order to implement and gain practical experience with kernel methods and with general machine learning techniques, the Kernel Methods for Machine Learning Class is ended by a data challenge.

This challenge consists in classifying images into 10 different categories using kernel methods implemented from scratch. Multi-class classification is a classical problem in machine learning and many deep learning based approaches achieve excellent results but our goal is not to get the best recognition rate on this data set at all costs, but instead to learn how to implement things in practice.

For this reason, the use of external machine learning libraries is forbidden. For instance, this includes, but is not limited to, libsvm, liblinear, scikit-learn, ...

1. Introduction

In this data challenge on image classification, we are given 5000 classified images as training data and 2000 images as test data. Each image is represented by a 32 (height) x 32 (width) x 3 (color) vector of values between -1 and 1. There are 10 classes. The performance is evaluated by the classification accuracy on the test data. A public leader board is available and the score is calculated upon approximately 50% of the test data. The final results will be based on the other 50%. Images of the dataset don't look like natural images because they have undergone some transformations, yet we can distinguish the categories of objects in the images.

2. Feature Extraction

Pixel values are not good descriptors for image classification. Training a classifier directly on flattened pixel values will give low accuracy (eg. accuracy of SVM with rbf kernel on pixel values $= 0.26$) because of the poor expressive power of such features.

2.1. SIFT features

As described in [3],this method is composed of the following steps. First we build a pyramid of images convolved with Gaussian filters at different scales, and then we compute the differences of successive Gaussian-blurred images (DoG). Potential keypoints are taken as the local extrema of the difference-of Gaussians pyramid across space (8 neighboring pixels) and scale. Second, we adjust the keypoints to more accurate positions by interpolation using the quadratic Taylor expansion of the Difference-of-Gaussians function taking the candidate keypoints of the previous step as origins. At the same time, we discard keypoints that are either unstable, low-contrast or on edges. Third, we assign an orientation to each keypoint by computing a histogram of oriented gradients in the neighboring pixels. The orientation of the highest peak is assigned. Last, we build a local feature descriptor for each keypoint by creating 4 histograms, taking the orientation of the previous step into account; each histogram is built upon the Gaussian weighted oriented gradients of neighboring pixels. By concatenating 4 histograms, we obtain a local feature descriptor.

To make the implementation simpler, I didn't localize keypoints like in the SIFT standard method but I created a grid on the image, and each element of the grid serves as a reference point for a SIFT descriptor. Around this point, one begins by modifying the local coordinate system to guarantee invariance to the rotation, using a rotation of angle equal to the orientation of the key point, but of opposite direction. At each point, the orientation and amplitude of the gradient are calculated. Then, the retrieved histograms are concatenated and standardized. In order to reduce the sensitivity of the descriptor to changes in brightness, the values are capped and the histogram is again normalized, finally providing the SIFT descriptor of the key point.

2.2. HOG features

We follow the method introduced in [1], which focuses on the structure of the object. It extracts the information of the edges magnitude as well as the orientation of the edges.

- It consists of dividing the image into cells, for each cell accumulating a local 1-D histogram of local gradient directions over the pixels of the region. A histogram corresponds to a local descriptor. We use a a number of bins $n_{bins} = 9$.So we take the 64 gradient vectors of each cell (8x8 pixel per cell) and put them into a 9-bin histogram.
- Then, we group cells into blocks with 50% overlap, so there are going to be $3 \times 3 = 9$ blocks in total for an image of size 32×32, and each block consists of 2×2 cells with 8×8 pixels. We use these blocks to normalize histograms.
- We perform these operations on all channels of the image and then we concatenate features.

We obtain a feature vector of shape: $(\text{nchannels} \times \text{nblocks}_x \times \text{nblocks}_y \times \text{nbins} \times \text{cells per block}_x \times \text{cells per block}_y) = 3 \times 3 \times 3 \times 9 \times 2 \times 2 = 972$

3. Dimensionality reduction

To reduce the dimension of SIFT and HOG features we perform a standard PCA. We always select a number of components that explains around 90% of the variance. This allows us to train classifiers much faster and slightly improves accuracy results.

4. Classification

Once the Features are calculated using one of the previous methods we can train a classifier on these features. I opted for a kernel SVM classifier like the one implemented in the 3rd homework but adapted for multi-class classification. In order to solve the convex optimization, we use QP solver (from the cvxopt package).

I implement two variants of the Multi-class SVM: a "one vs all" SVM model and a "one vs one" model.

In the one-vs-one strategy, given k different classes to classify, the algorithm provides $K \times (K-1)/2$ SVM binary classifiers. Each classifier is trained to correctly classify 2 of the K given classes using in the training process only the entries in the dataset to which it corresponds a label of the 2 classes. To predict the label of a given sample, we use all $K \times (K-1)/2$ classifiers and the final class is computed deploying a voting schema among the classifiers. The voting process is based on the standard predict function for binary SVM classifiers, so the input entry is assigned to the class which wins the highest number of binary comparisons.

In the one-vs-all strategy, given K different classes to classify, the algorithm provides K SVM binary classifiers. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident. I obtained better results using the one-vs-one SVM because we encounter problems of class unbalance in the one-vs-all SVM.

Even though, I train 45 kernel binary SVMs, the training is very fast because kernel functions are implemented in a very efficient way (except for chi2 kernel). My best models is trained in $13s$ only.

5. Kernels

I implement different kernels: linear, rbf, laplacian, chi2, polynomial. My best results were obtained using rbf and laplacian kernels.

6. Results

For parameter tuning, I perform a 5-fold cross validation to choose classifiers, kernels, and the parameters of the kernels. My best submission (0.597 accuracy in public LB.) is obtained by voting among 3 predictions:

- HOG features + PCA(90% variance + scaling) + SVM (Laplacian kernel)
- SIFT features + PCA(90% variance + scaling) + SVM (RBF kernel)
- SIFT+HOG features + PCA(90% variance + scaling) + SVM (RBF kernel)

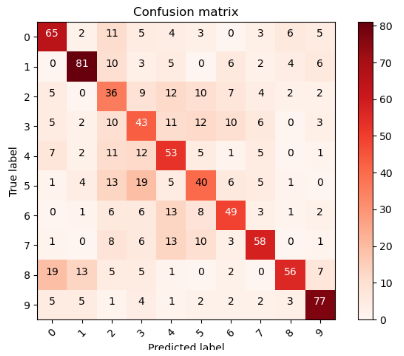

Figure 1. Confusion matrix.

7. Conclusion

In this project, I had the chance to understand some classical yet very important computer vision methods like SIFT and HOG and to implement them. While CNNs are state of the art for many computer vision tasks, this project shows that kernel methods, accompanied with a proper preprocessing, can lead to very decent prediction scores.

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[2] https://phortail.org/club-informatique/definition-informatique 136.html. Date de dernière consultation : 26/08/2019.

[3] D. G. Lowe. Distinctive image features from scale-invariant keypoints, 2003.