MVA Kernel Methods in Machine Learning - Data Challenge

https://github.com/tomsander1998/KernelMethodsMVA

Tom Sander
Ecole Polytechnique
tom.sander@polytechnique.edu

Jean El-Khoury
Ecole Polytechnique
jean.el-khoury@polytechnique.edu

April 7, 2022

Abstract

As part of the course "Kernel Methods in Machine Learning", this assignment aims to implement kernel methods for image classification from scratch. Our implementation can be found here.

Presentation of the project

We approach the problem of image classification in a particular context: no Deep Learning, use of kernel methods, and implementation of algorithms from scratch. In Section 1, we explain our approaches, which are inspired by [2] and [1]. In Section 2, we give an overview of what other methods we tried. In Section 3, we show our results and discuss the chosen parameters.

1. Methods used

One vs One SVM: All of our models use a one versus one SVM on top of the features (classical SVM for each pair of classes, and counting the vote for each class against every other). We chose that instead of one versus the rest because it gives access to the votes.

1.1. HOG and Kernel SVM

HOG features: Histogram of Oriented Gradients (HOG) [2] features counts the occurrence of gradient orientations in parts of an image, and outputs concatenations of histograms.

Kernels: We implemented several kernels: Linear, Polynomial, Gaussian and the Laplacian Kernel which has the following formula:

$$K(x, y) = \exp(-\frac{\|x - y\|_1}{\sigma^2})$$

SVM with Laplacian Kernel on HOG features: Using a one versus one multiclass SVM classifier on HOG features was a great improvement compared to raw data.

1.2. Unsupervised feature learning

This method uses a particular procedure to extract relevant features from images, and uses a SVM classifier to produce predictions. The paper [1] states that it achieves good performance on the CIFAR-10 dataset, so we decided to implement it.

General pipeline: The features are extracted as follows: First, we consider the entire dataset (training and test images), from which we extract a large number of small-scale patches. Then, we perform a clustering of these patches with a selected number of cluster centres. These cluster centres represent typical small patterns that we can find in our images. After extracting these centres, we use them to compute feature vectors: For each image, we extract all possible patches and compute the L2 distance between a patch and all centres. In this way, we obtain a feature vector for each patch of the image. To obtain the features of the image, we simply add the features of all the patches. These features are then fed into a support vector classifier with a kernel.

Image whitening: An important step in this pipeline is image whitening, described in [3]. It consists of transforming a sample to make its covariance matrix identity. This is achieved by computing the eigenvalue decomposition of the covariance matrix C: $C = EDE^T$. We then transform each of our patches x into $\tilde{x} = ED^{-1/2}E^Tx$. This is applied in our pipeline just before the clustering step. According to the paper, this process is crucial to achieve good clustering and, in the end, good classification results.

Details and parameters: Specifically, we extract 200 000 patches of size 6 x 6 from the entire dataset (training and test). We normalize these patches by subtracting their mean and dividing by their standard deviation so that all patches have the same contrast, and finally apply image whitening on them.

We perform clustering with KMeans on these whitened patches to find 400 cluster centroids. Then, for each image, we extract all possible 6 x 6 patches and calculate the L2 distance between the patches and the centroids. The feature vector of a patch contains 400 coordinates. To sparse this feature vector, we set the coordinates that are below the mean to 0 (this gives us about 50 % sparsity). To get the feature vector of an image, we could simply add all the feature vectors of its patches, but a better method is to compute a feature vector for each quarter of the image (top right, top left, bottom right, and bottom left) and concatenate these vectors. In the end, the representation of the image has 1600 coordinates.

1.3. Combining both methods

In the one-versus-one SVM, we train a classifier for each pair of classes and then look at how often a particular label is voted for when it competes against all others. We looked at the distribution of votes in our first method and found that in most cases (about 98%) one of the labels is chosen against all the others, giving a score of 9. We noticed that apart from this label, it is frequent that some non-chosen labels have a good score in terms of how they compare to others (e.g., a label that prevails against 8 others). When this is the case, we check the second model and use its output as a double check. If a label is preferred by model 1 over more than α others and happens to be chosen by model 2 as well, then it will be the output of our aggregated model. Otherwise, we choose the label with the most votes from our first model. This solution gives our final and best prediction.

2. Other methods we considered

Apart from the aforementioned models, we have tried to improve our models in several ways:

- Different Kernels
- Different HOG parameters
- KernelPCA on top of the features
- Adding histogram of colors to the HOG features
- Data augmentation (rotation, flip, crop, color changes)

We tried working with Fisher Kernels for global descriptors, but it seemed too much work to implement them from scratch. It was frustrating not to be able to improve much our first model: we spent too much time tuning its parameters instead of trying completely different things. The second model had good accuracies on the entire CIFAR-10 dataset in [1], so we thought it would work better with our data, but at the end this model performs as well as the first one.

3. Results

3.0.1 Parameters

- In the first model, the HOG features have a 312 length (9 orientations, $3*3$ cells per block and $8*8$ pixels per cell). We used a Laplacian kernel with $\sigma = 3, C = 1$ for the SVM. The corresponding accuracy is 0.55.
- In the second model, apart from the parameters previously explained, the classification is performed with an SVM classifier, using a Gaussian RBF kernel with variance $\sigma = 60$, and a regularization parameter $C = 10$. Those parameters gave us the best score for this method after a grid search. The corresponding accuracy is 0.56.
- We chose as a threshold $\alpha = 6$, an perform 0.574 on Kaggle.

References

[1] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. 15:215–223, 11–13 Apr 2011.

[2] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. 1:886–893, June 2005.

[3] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, 2000.