Kernel Methods in Machine Learning
Course Project
March 12, 2017

Image Classification with Kernel Methods

Olivier Chancé : chanceolivier@gmail.com
Sophia Lazarq : sophia.lazraq@gmail.com
Peter Martigny: peter.martigny@gmail.com

Supervisors :
Jean-Philippe Vert
Julien Mairal

Abstract

The goal of this challenge is to implement kernel methods to classify images from the CIFAR Dataset. Machine Learning libraries are not allowed.

Key words : image, classification, kernel, svm, sift

1 Presentation of the challenge

The challenge consists in classifying images (probably from CIFAR dataset) from 10 different classes: airplane, car, bird, cat, deer, dog, frog, horse, ship and truck. The images are of size 32 x 32 x 3 pixels, and are given with an unknown pre-processing. The sample consists of 5000 balanced training images, and the test sample contains 2000 images to be classified. The challenge is to implement our own algorithms without access to machine learning libraries.

The rules enable us to submit our predictions at most twice a day and get a public score which is associated to half of the final unknown dataset.

The implementation is made on Python, and the package cvxopt is used to implement convex optimization for the selected classification algorithm (SVM). Among the different kernels tested , the Chi2 kernel gives our best score. Our final scores are 0.634 on the public leaderboard and 0.637 on the private leaderboard, which refers to the fact that our model generalizes well.

2 Methodology

The aim of the challenge is to classify images which are not properly represented in pixels basis. Therefore, it is crucial to first transform the image, namely using a suitable extractor.

In order to extract dense features from the images, we implement a SIFT algorithm as preprocessing. This enables to have less and better features to be fed in classification algorithms. To distinguish the different classes, this project implements Support Vector Machines (SVM) with chi-square kernel. We use a one-versus-all scheme to produce multi-class classification.

Basically, concerning the classification algorithm, the approach taken was to test multiple classifiers (SVM , Kperceptron) associated with diverse kernels (linear, gaussian, polynomial, sigmoid, chi-square) and to tune its corresponding parameters. Kperceptron (with coordinate descent for the convex optimization) performs well too (0.57 locally). Since SVM with chi-square kernel performs the best, we choose to detail its implementation.

2.1 Sanity Check and Visualization

First of all, we check that the training data are uniformely distributed on classes. Indeed,in the training set, each class is associated with 500 samples. It allows us to use the classical classification methods.

Then, we visualize some images to get an idea about their appearences and potentially the kind of transformations operated on them. Most images appear blurred and at that stage, we can't affirm that all images have been transformed the same way.

2.2 Image Processing - SIFT

SIFT is an algorithm presented in [Low03] used in computer vision to compare two images. One of the key steps of the method is the construction of the SIFT descriptors. If two image images are similar, they will have similar SIFT descriptors, otherwise the descriptors will be different. The implementation that was carried out for this project focused on the construction of these descriptors.

In the original method, the first step is to find the key points of the image that mostly correspond to the corners and find the best scale of representation. This step has not been implemented. We opted for a simpler implementation that consists in creating a grid on the image. 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.

Our python code is adapted from the matlab implementation of Svetlana Lazebnik (University of Illinois)

2.3 Classification algorithms with kernels

As mentionned before, we use SVM for classification, which was first introduced in [VNV92]. The goal is to find a set of weight and bias such that the margin is maximized. We use kernels instead of feature vectors because the computation is much easier with kernels. In the case of SVM, we replace the dot product term with kernel, hence we don't have to use feature vector at all. We test SVM with a set of classical kernels, namely linear, gaussian, polynomial, sigmoid and chi-square kernels and we realize that chi-square performs the best.

2.4 Multi-class classification

We implement a "one vs all" SVM approach instead of the usual "one vs one" approach that would have taken much more time without necessarily performing well. Indeed, for a "one vs all" approach, we train our model using 10 classifiers whereas the "one vs one" approach would require K(K-1)/2 classifiers, namely 4.5 times more than the "one vs all" approach since we have $K = 10$. In order to solve the convex optimization, we use QP solver (from the cvxopt package) which stores the entire kernel matrix. QP solver works well in this case since the problem is small, still, it is very slow.

2.5 Data Augmentation

It is possible to extend artificially the size of the training set. A classic method consists in carrying random rotations of the images, in order to help the algorithm better generalize. However, this aspect is already taken into account with the SIFT transformation, which is invariant to rotations. Augmenting the training set with rotation create duplicates after SIFT transformation and thus lowers the prediction score. However, it is also possible to flip the images. An horizontal flip create another image which is still very plausible. However, a vertical flip creates an "inverted" image, and we observe that inverted images are very rare in both training and test set.

Thus, we use an horizontal flip to double the size of the training set. This action enabled to gain 1.8% on the public leaderboard.

2.6 Parameters tuning

In order to tune the parameters, we used 3-fold cross validation. Since we have simplified the SIFT method, we decided to cross validate parameters that are conventionally fixed. (Size of patches, number of angles per patch ...) All the parameters we have tuned appear in the signature of the SIFT and SVC classes.

3 Results

Our final model uses data augmentation with a horizontal flip of images. Hence the training set size is doubled. We then extract SIFT features, before using a multi-class SVM (one-versus-all) with a Chi-2 kernel to classify images. Our final score on the private leaderboard is 63.7%. This submission does not include data augmentation, due to larger computation time. On our local test (random 70/30 split), the model with data augmentation score 66.1%, and the kaggle platform indicates 65% (post deadline entry). We note that we had a better score on the public leaderboard. This could be due to an unbalanced 50-50 split of the test set to generate a public score.

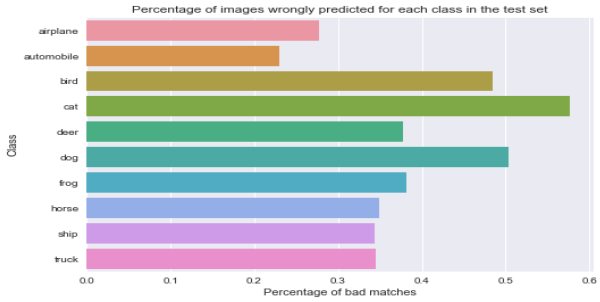
We show in Appendix the errors made by the classification. For a random 70/30 split we represent, for each class, the percentage of images wrongly classified. We observe that the classification are considerably bad for cats, dogs and birds. However, the predictions are way better for automobiles and airplanes.

4 Conclusion

While CNNs are state of the art for many computer vision task, this project shows that kernel methods, accompanied with a proper preprocessing, can lead to very decent prediction scores. However, the training of kernel methods is very slow, due in part to the computation of a kernel matrix. To leverage the power of deep networks and kernels, it would be very interesting to use a deep kernel network to predict the classes of images.

Appendix

4.1 Analysis of errors



References

[Low03] David G. Lowe. Distinctive image features from scale-invariant keypoints, 2003.

[VNV92] Bernhard E Boser Vladimir N Vapnik, Isabelle M Guyon. A training algorithm for optimal margin classiers, 1992.