MVA 21 Kernel Methods data challenge

Team name : RBAR

GitHub : https://github.com/redabelhaj/km_challenge_mva_21

Reda Belhaj-Soullami
reda.belhaj-soullami@polytechnique.edu

Mohamed Amine Rabhi
mohamed.rabhi@polytechnique.edu

1. General idea of the method

The project consists in the classification of CIFAR-10 images using Kernel methods, without using any external implementations. The project featured two steps :

- Design and implementation of a kernel for images, well-suited to the task. We build upon the paper [1], on a method using a dictionary of patches for image classification on CIFAR-10.
- Implementation of a multi-class SVM using the above kernel to perform the task. We used the one-vs-one scheme.

2. Designing the kernel

For this project we chose to reproduce the results of the paper [1]. Note that the authors have provided an implementation of their own but we did not use it in our work as the goal was to implement everything from scratch.

2.1. Overview of the method in [1]

In this paper, the main idea is to use random patches from the dataset to encode an image. After extracting a dictionary $\mathcal{D}$ of patches from the dataset, a patch of the image is encoded as the set its K- nearest (in terms of Euclidean distance) patches in the dictionary. More formally, for an image x, a patch i and a patch of the dictionary d, we can define the encoding $\phi(x, i, d) = 1$ if d is among the K-nearest neighbors of i, 0 otherwise. Then, the kernel is simply given by

$$K(x, x') = \sum_{i,d} \phi(x, i, d)\phi(x', i, d)$$

2.2. Additional features of the method

A number of refinements and variants can be made from the method. We have experimented with the following ones, and discuss our findings in the next section.

Whitening The authors in [1] use a whitening transform on the patches before computing the nearest neighbours and stress that this is an important step that results in significant increase in performance. This consists in estimating the empirical mean μ and covariance Σ of the patches and defining $W = (\lambda I + \Sigma)^{-1/2}$. Then the patches are processed first by being centered by μ and multiplied by W. We rely on a SVD decomposition of Σ to compute W.

Making the dictionary contrast-invariant This consists simply of augmenting the dictionary of patches with the opposites of the patches.

Sum-pooling and batch normalization To reduce the computational burden of the method a mean pooling is applied after computing ϕ. This consists of summing the values of $\phi_{i,d}$ for i in a $m \times m$ square. We opted for average pooling but it is equivalent. Also, batch normalization is applied after computing the features ϕ.

Data augmentation Following [1], we used random cropping as well as random flips to augment our dataset.

Soft assignment This consists in replacing the hard nearest neighbours assignment by a soft one. Let x be an image from our dataset and $\tau_{i,x}$ be the distance to the K-th closest patch in $\mathcal{D}$. Let $p_{i,x}$ be the i-th patch of the image x. The hard assignment defines ϕ as $\phi(x, i, d) = 1(\|p_{i,x} - d\| \leq \tau_{i,x})$ and the soft assignment defines ϕ as defines ϕ as $\phi(x, i, d) = \sigma(\|p_{i,x} - d\| - \tau_{i,x})$, where σ is the sigmoid function.

2.3. Our findings

After experimenting with the above, we found that

- Whitening does not yield a significant increase in accuracy. We believe the reason is that the input images are whitened already.
- Making the dictionary contrast invariant seems to help, which is in line with the observations in [1].
- Soft-assignment yields worse accuracy than the hard assignment so we chose not to use it at all.
- Batch normalization seems to increase accuracy.
- Data augmentation does not help systematically.

In light of these remarks we decided to have 4 different models where we use or not whitening and data augmentation and ensemble the predictions using a simple majority vote.

3. Implementation of the multiclass SVM

To implement the multiclass SVM, we built upon the homework where we implemented the binary SVM. To handle the multiclass case, we used the One-vs-one scheme where we create one binary classifier for each pair of classes and train on the examples where the label is one of the classes. At inference time, we compute the votes of each classifier and assign the class with the highest number of votes. The computational cost of this method is not very large in our case as we have only 10 classes, and we only need to compute the kernel matrices once.

4. Implementation details

For our final experiments, we use a dictionary of size 1000 (2000 if we use data augmentation) and a threshold of 200 neighbours (400 in the case of data augmentation). We use patches of size 4×4 and a mean pooling on a 5×5 window. When using the whitening transform we set $\lambda = 10^{-4}$. For the SVM, we use a regularization parameter of $C = 100$.

We rely on some external libraries for the following

- PyTorch : tensor processing (especially on GPU), SVD, data augmentation (although we restricted ourselves to use very simple ones), convolutions, batch-normalization.
- Numpy for general-purpose tensor operations.

We advise to use a GPU to run the method on all the data as we make use of PyTorch operations for computing the kernel which speeds up the computation time by a large factor.

5. Results and conclusion

We achieve close to 60% accuracy using this method, with only 5000 images from CIFAR-10. In the paper, a benchmark experiment on CIFAR-10 (60000 images) yields an accuracy of 87%. In conclusion, this simple and systematic method is quite effective at finding a good representation of the data. For larger datasets (such as the full CIFAR-10) using an SVM would be computationally hard, and we would need to use stochastic methods such as SGD. This is the approach taken in [1].

References

[1] L. Thiry, M. Arbel, E. Belilovsky, and E. Oyallon. The unreasonable effectiveness of patches in deep convolutional kernels methods. 2021. 1, 2