Rapport

March 21, 2016

1 Repport Kernel methods : Digit Recognition Challenge

Basile CALDERAN & Fabrice ZAPFACK

1.1 Data Reading

Data where given as 3 CSV files:
Xtr.csv : Training images
Ytr.csv : Labels of Training images
Xte.csv : Test images

1.2 Part1 : Analysis and choices making

During that first part we decided to use the different librairies available to quickly test different approaches to resolve this problem. The objective was to be able to quickly find the best models and their associated parameters.

1.2.1 Problem formulation

The task given is a multiclass classification (10 classes). The dimension is high (784) so we have to keep in mind the **curse of dimensionnality**.

The histogram on the training labels let us conclude that the problem was more or less balanced (no need to give different weigths to classes)

1.2.2 Data preprocessing

We have tried different preprocessing techniques in this part. We evaluated the impact of these operations on the classification task by comparing with crossvalidation score obtained using the raw data. The model used was the best one found in scikit learn classification models (see next section):
Transpose : Had no impact on classification (good for visaulition)
Binarization : It was used as asimple segmentation step. It happens to improve a little bit accuracy
Rescaling : Used to project the data in $[0, 1]$ but decreased the accuray : $X_i = (X_i - min(X_i))/(max(X_i) - min(X_i))$
noise filtering : Done with a mean filter (slidding window), lead to a decrease of accuracy
Edge detector :Done with a sliding window. A couple of filters were tested (derivative, laplacian, edge, ...). It Increased accuracy with a cross template.
Gauss Filter : Sliding window where coefficients have a real and an imaginary part (orthogonals). Decreased accuracy.
PCA : Good for dimension reduction. The best score is obtained with 40 components
KPCA : Associated with linear models. Outperformed by svm
Linear Discriminant Analysis : Outperformed by PCA
Wavelets Transformation : Outperformed by PCA + Edge detector
Augmented train set : We tried to add some transformation invariance to our model by adding to the training dataset images obtained by images obtained by adding some transformations (translations

and rotations) to the original images. However that appraoch was dropped because the improvement of performance we got was not worth it compare to the run time induced by doubling or tripling the learning dataset

1.2.3 Models selection

In that part we focused on optimising the 2 models (with kernels) implemented in scikit-learn that we thought were the most appropriated for a muti-class classification problem : **logistic regression** and **SVC**. A kernelized SVC appeared to be the best model with a 'rbf' kernel (sigma=0.02).

1.2.4 Partial conclusion

At the end of that part we came up with a good framework to answer the given problem: Binerization + Edge detector + PCA + SVC

1.3 Part2 : Implementation

The implementation of the edge detector was very straight forward (sliding windows)

1.3.1 PCA

To implement the PCA, we used numpy to perform the eigen values decomposition of the covariance matrix of X. We simply troncate the first k eigenvalues after.

1.3.2 SVM

Dual VS Primal We decided to propose 2 implementations of SVM :

One resolving the **primal** problem using *stochastic gradient descent* (theorical guarantees can be found in http://www.cs.huji.ac.il/~shais/papers/ShalevSiSrCo10.pdf)

One resolving the **dual** problem using a *quadratic solver* (CVXOPT)

We were happy to find that our SGD SVM was able to find good approximate solutions, for this problem, for reasonnable number of iterations.

Multi-class In order to adapt to the multi-class case we implemented a **1 VS 1** approach. We lately compared it with a **1 VS all** approach and the later was better both in terms of time and accuracy.

1.3.3 Conclusion

Compare to the other group, our score was not really good (94% accuracy) even though we used a very rigourous approach and spent a lot of time in pre processing. We tried to give explanations in the gap of perfomance between our models performances and those of our classmates.

Having computers that are not very poweful it took quite a lot of time to train our implemented algorithms (no parallelisation added). Therefore we didn't perform a model selection step and used the optimal parameters values obtained via the scikit-learn implementation (grid search). There is propable that those parameters are not the best ones for our own made implementation

We had trouble when trying to use a gaussian kernel. We lately foud that the training features had to be normalized but we were incomfortable with the idea of normalizing features obtained after PCA.

We also think that other pre processing could help achieve better performance, typically a better segmentation.

We also thougth about replacing the hinge of the SVM primal formulation with others loss (sigmoid, least square) and use our SGD solver. Later use the 3 models to obtain better prediction by weigthed majority votes. However we did't have enough time to implement that idea.

Even if the digit recognition is a well know subject, implement an SVM without using library give us a better knowledge in how it work and how it can be addapted for different task. We are very curious to know what our classmates have done to obtain such great scores.