FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Support Vector Machine

MA5204: Machine Learning 2019

Submitted To:
Felipe Tobar

Submitted By :
Alexandre Poupeau

June 15, 2019

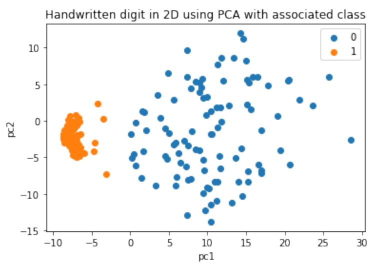
1 Introduction

This report is about SVM - Support Vector Machine - as a classification tools to classify handwriting digits using the MNIST dataset (number of elements : 50000 train, 10000 validation and 10000 test). This report especially focuses on the impact of a hyperparameter and the use of different kernels in the results. Precision : we only used 20000/50000 elements of the train dataset to make the training much faster.

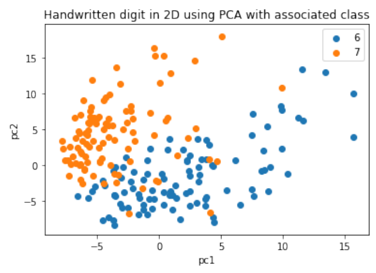
2 Data Visualization

First, we made sure that the images we had were well-associated to their labels. In order to do that, we simply create a function that takes a random element from the train dataset, display the digit image and its label (int between 0 and 9). We did not verify all element in the dataset but it was just to make sure what we work with makes sense.

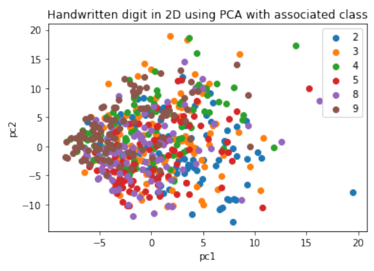
Then, we used principal components analysis to get a better idea of how hard is going the task to classify those handwriting digits in the end. Here are three graphs of the projection of 100 elements from each class (0 to 9).



(a) 0s and 1s projections



(b) 6s and 7s projections



(c) Rest of the projections

One can notice that the digit 1 is the easiest element to classify. This is due to its unique vertical stick shape that makes it so easily comparable. We found out that the 6 and 7 digits are quite well distinguishable too. However, the rest of the digit seem particularly difficult to classify. The use of more dimensions will be crucial to classify them well : the two "best" dimensions are not sufficient.

3 Study of the C hyperparameter

After creating three easy kernel functions to calculate the linear, polynomial and gaussian (also called rbf kernel) kernels, we studied the impact of the hyperparameter C on the accuracy of the model using only the rbf kernel for the moment. Here is an interesting link that contains most kernels : `http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/`

The C hyperparameter is the C in the formula : $min(\frac{1}{2}||\omega||_2^2 + C\sum_{i=1}^{N}\zeta_i)$. Since the ζ_i represent the error that we allows for the element i to not be perfectly well classified, C finally represent how permissive is our model. The smaller the C, the more permissive is our model. Here is a graph that resume most of the study on C :

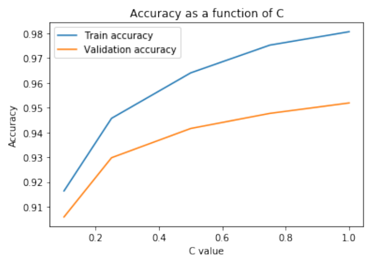
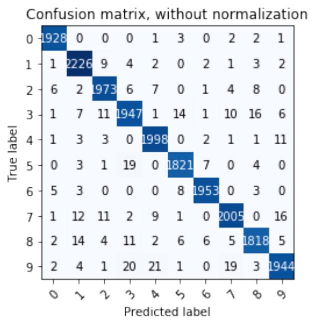
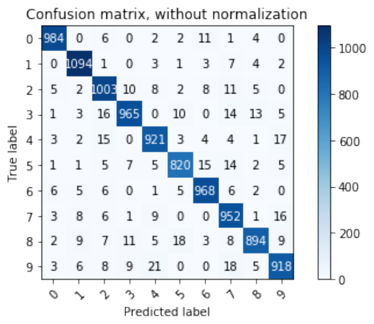


Figure 2: Accuracy as a function of the C hyperparameter

We can clearly see that the model is doing much better with a higher-value of C, which means a rather non-permissive or non-smooth model (or hiperplane). Here are the two confusion matrices we obtain with the rbf kernel with the train and test dataset.



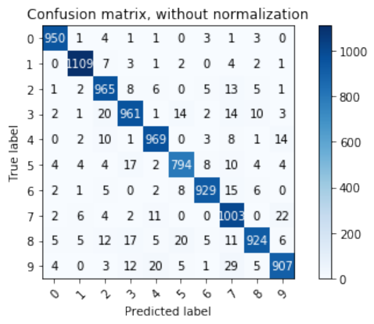
(a) RBF kernel



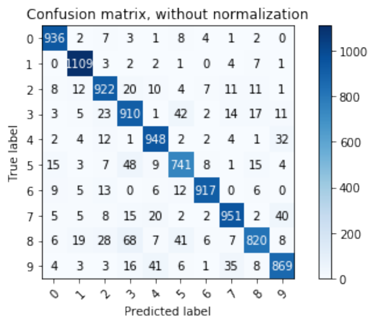
(b) Linear kernel

4 Study of different kernels

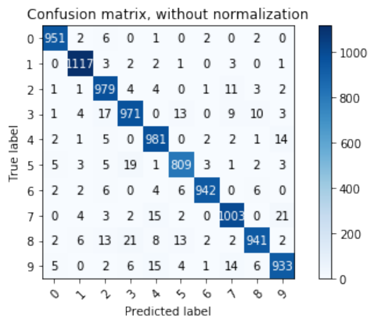
In this part, we used a fixed value of $C = 1.0$. Here are all the confusion matrix we can obtain on the test dataset.



(a) RBF kernel



(b) Linear kernel



(c) Polynomial kernel

Here is a tabular that resume the results obtained using the three different kernels.

Model	Validation accuracy	Test accuracy
RBF	95.19	95.11
Linear	91.23	91.23
Polynomial	96.28	96.27

It is quite normal that we obtain a very similar accuracy for the validation and test set.

What is more interesting to notice is that the linear kernel has the lowest accuracy score. This can be explained because this is the simplest model in that case and it may struggle to capture well the data distribution. This can be quickly deduce by looking at its confusion matrix. We can notice that is really struggle especially to classify the digits 2, 3, 5 and 8. There are some big mistake and ”confusion” happening with this kernel for example : 68 eight digits predicted as three (biggest mistake concentration) or 48 five digits predicted as three also. There is a huge gap with the polynomial and rbf model. The biggest mistake concentration with the rbf model is 22 and with the polynomial only 21.

The polynomial and rbf kernels have a really close accuracy nonetheless the polynomial has a slightly better accuracy score. We can conclude that the polynomial kernel is more efficient in this classification problem than the rbf.