ALM Data Challenge Report

Nils Defauw, Pierre Blanc-Fatin

February 17, 2020

1 Implementing a Support Vector Machine

In this data challenge, the task at hand was to create a binding map for three different Transcription Factors over three different datasets. Each dataset was composed of 2000 training samples and 1000 test samples for which we hadn't the correct mapping. Each training example was a tuple composed of a DNA sequence (a string over the alphabet {A, C, G, T}) and the corresponding TF binding (either 0 or 1). For the test samples, we had only the DNA sequence and had to predict the correct binding.

We decided early to implement a Support Vector Machine (SVM) with a simple kernel that we would improve later. We used the SVM formulation found in "Kernel Methods for Pattern Analysis" by Shawe-Taylor and Cristianini and we implemented it using cvxopt, a well-known Convex Optimization library for Python.

For testing purposes, we implemented the same performance metric as the one used by Kaggle which is the categorization accuracy and which counts the ratio of correct predictions over the total number of predictions.

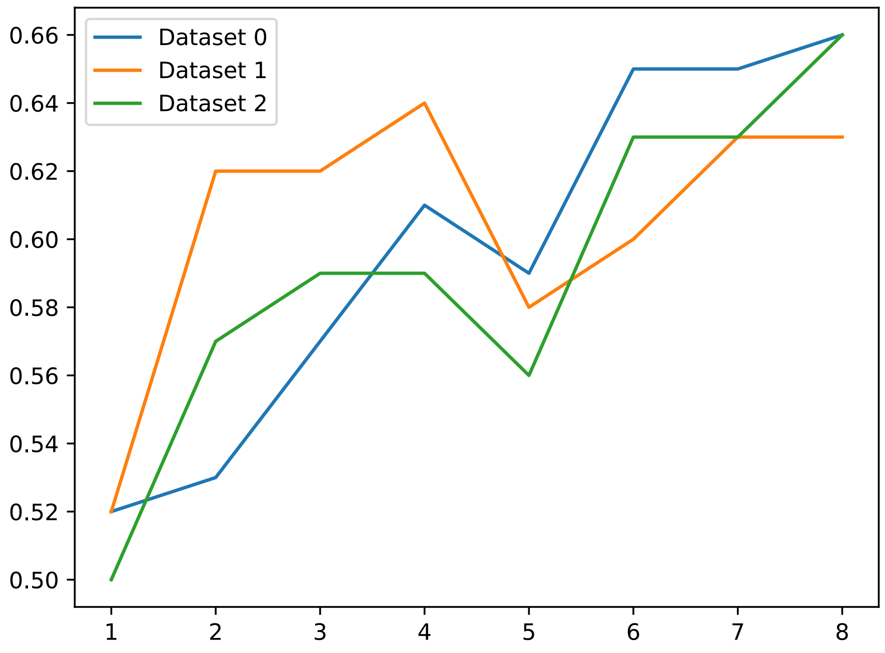
We implemented two different SVM fitting strategies. The first one was designed for experimentation over the kernels. Its working was to train the SVM over two thirds of the training data, using the remaining third for accuracy measurements. The second one was designed to be used once we were confident that our kernel was good. Its working was to train the SVM over the whole training data and to output the predictions over the test set.

2 Digging through the p-spectrum kernels

During our bibliographic search, we found the p-spectrum kernels for comparing strings over a finite alphabet as it is the case with our DNA sequences. The p-spectrum kernel is a kernel that computes the dot product between two vectors of features of two sequences A and B. There is 4^p features per sequence (4 being the size of our alphabet), each feature is counting the number of occurences of a specific contiguous substring of length p in the sequence.

Even if one of the advantages of a kernel is that it is not needed to consider the high-dimensional feature space, we decided during our work to compute the p-spectrum kernels by directly computing, for each sequence, its feature vector. This allowed us to considerably speed up the computation of the kernel matrix as for one given string A, computing $K(A, \cdot)$ involved the computation of the features of A only once, whereas just giving to the SVM the kernel function and building the kernel matrix using it was considerably slower. The main disadvantage with this technique was the memory consumption, as for the p-spectrum kernel, each feature vector was of size 4^p (exponential growth). It still allowed us to compute the p-spectrum kernels up to $p = 8$.

Our final submission was done with $p = 7$ but during our investigations of the p-spectrum kernels, we found some interesting results.



Above is plotted the categorization accuracy over the 3 different datasets for values of p in $\{1, \cdots, 8\}$. Instead of having smoothly growing curves for the 3 datasets, we observe different behaviors, some datasets peform better with small values of p, like the dataset 1, others for high values of p, like the dataset 0. Finally, the three different datasets perform bad with $p = 5$. We don't have a good understanding of this phenomenon at this time.