Report Kernel Method Data Challenge:

Introduction:
(i) Our team name is “667 EKIP”.
(ii) Our team is composed of Doron Israel, Elliot Muller and Nicolas Wagner.
(iii) Our private score is 0.66733 and our public score is 0.67533.

The link to our Git for this Challenge : https://github.com/nicostanw/Kernel_Challenge
The Concise_Code_Kernel_Data_Challenge.ipynb file contains a few examples for each kernel and solvers : this is the file we recommend that graders run.
The Complete_Code_Kernel_Data_Challenge.ipynb file contains all our models. To completely see the amount of work we have done for this project, we recommend graders to read this file.

## I. Process followed

For this challenge, we first applied the "usual" kernels (gaussian and polynomial) on the given matrices, and then the kernels for protein sequences (spectrum and mismatch) directly on the sequences. We then tested each of these kernels with the classifiers Kernel ridge regression (KRR), Kernel logistic regression (KLR) and SVM.

For each classifier, we built a function and a corresponding class. Each class is based on the scikit API to easily optimize hyperparameters with GridSearch. Thus, all the classes contain a fit feature, as well as a predict feature. The fit feature calls the corresponding function which returns from X_train, y_train and from the hyperparameters the best alpha vector (cf. Representer Theorem).

For each classifier and each dataset (X_0train, X_1train and X_2train), we manage to go through a GridSearchCV to find the best values for the parameters. From this point, we then used a Randomized GridSearch with a finer set of possible values (we wanted to use 300 iterations every time, but some classifiers were quite slow to train, so we eventually went down to 100 iterations in certain cases). We have thus refined our search for optimal hyperparameters.

## II. Explanation of kernels construction for protein sequences

For both spectrum and mismatch kernels we put the complete vocabulary in a dictionary where each word in the vocabulary is associated with an index. To compute the representation of sequence x associated to the spectrum kernel with parameter k we first created an empty vector with the size of the vocabulary. Then we run through x and for each subsequence of x of length k we look in the dictionary for the associated index and we add plus one to the empty vector at this index.
We follow the same idea for the mismatch kernel but for each subsequence of length k in x we have to determine all the subsequences equal to this subsequence up to m mismatch. And for a given m there are $\sum_{k=0}^{m} C_k^{m}$ where $C_k^{m}$ is k among m. For m small enough the computation of the mismatch kernel is fast but for m>3 it starts to require more than 5 minutes but this method is still much faster than running through the vocabulary.

## III. Best results

The best scores on the test sets and on Kaggle were obtained for the string kernel spectrum and mismatch, i.e. on the kernels directly applied to the strings, and not on the provided matrices. Thus, we do not detail the results of the other methods.

Here are the results of the best classifiers for these 2 kernels with the optimal parameters (after the Randomized Grid Search which gives even more precise parameters for the hyperparameters).

| Number of submission | Kernel | Method | Average public/private scores |
|---|---|---|---|
| 1 | Gaussian | KLR | 0.62 |
| 2 | Spectrum | KLR | 0.645995 |
| 3 | Spectrum | SVM | 0.649665 |
| 4 | Mismatch | KLR | 0.67133 |
| 5 | Mismatch | SVM | 0.661995 |

Spectrum kernel:
The best results for the spectrum kernel were obtained with Logistic Regression.
KLR: after optimization over the lambda and k, we find :
X_0train : lbd*=0.03, k*=7, mean_test_score (mts)= 0.6555
X_1train : lbd*=0.12, k*=6, mts= 0.6585
X_2train : lbd*=0.004, k*=7, mts= 0.7455

Mismatch kernel:
The best results for the mismatch kernel were obtained with Logistic Regression.
KLR: after optimization over the lambda, k and m, we find :
X_0train : lbd*=0.19 , k*= 9, m*= 1, mts= 0.6635
X_1train : lbd*= 0.36, k*= 9, m*= 1, mts= 0.6620
X_2train : lbd*=6.4 , k*=9 , m*=2 , mts= 0.7575

## IV. Conclusion

Our best score of 0.67533 on the Kaggle public leaderboard was thus obtained from the kernel mismatch with the hyperparameters values mentioned above. As we can see, the spectrum Kernel also gave satisfying results. Working directly on the raw sequences was therefore more efficient with the use of the adapted kernels. We can see that the results are more convincing when the sub-sequences considered are longer, with few mismatches. It could have been interesting to observe the results with a larger value of k (and which would have perhaps given a higher value for m*), but this would have required too much computation time to consider this method really applicable in practice (due in particular to the treatment of all the possible subsequences of length k equal to this subsequence up to m mismatch ).