# BindMeBaby
# Data Challenge Report

Dadi Leello Tadesse  Hamzaoui Dimitri  Bujalance Martin Jesus

**Abstract**

In this report we describe the method we, the BindMeBabies, used to classify DNA strings according to their TF binding affinities. We achieved an average accuracy of 71.633% on the data challenge, ranking 7th and 6th on the public and private leaderboards.

## 1 Selected Approach

Three datasets, each corresponding to different TFs were provided to us. We trained a different classifier for each dataset. In what follows, we give a detailed description of each one of them.

### Dataset 0

Three mismatch kernels [1] $K_1 : (k = 9, m = 1), K_2 : (k = 10, m = 1), K_3 : (k = 11, m = 1)$ are computed from the provided DNA strings. The sum of the three kernels $K = K_1 + K_2 + K_3$ is then fed to an SVM classifier with regularization parameter $\lambda = 0.7$.

### Dataset 1

This dataset is handled just like the previous one. Three mismatch kernels $K_1 : (k = 9, m = 1), K_2 : (k = 10, m = 1), K_3 : (k = 11, m = 1)$ are computed from the data. Their sum is fed to an SVM classifier but this time with regularization parameter $\lambda = 0.8$

### Dataset 2

The 12-spectrum [2] feature map is used to embed the data in a feature space. A Gaussian kernel with $\sigma = 10$ is then used to compute the gram matix K. An SVM classifier with regularization parameter $\lambda = 10^{-7}$ is trained with K as input.

**Model selection :** Model selection was done through 4-fold cross validation for each dataset. We chose the model with the highest average accuracy.

## 2 Computational details

Hyper-parameter search and model selection are computationally heavy procedures. For them not to take hours, a careful choice of data structures was necessary. The k-spectrum features of

the DNA strings were sparse so representing them as hash-maps provided a considerable gain in performance. And for computing the mismatch neighborhoods of k-mers, deques proved to be quite useful as they allowed for fast insertions and deletions. As for our optimization needs, we used the quadratic program solvers included in the `cvxopt` package.

Our code is organized as follows : the `DataHandler` class reads and embeds the DNA string in a chosen feature space, the `Kernel` class handles the computation of gram matrices and function evaluations and the `LargeMargin` class handles the optimization aspects.

# 3 Things that didn't work

The selected approach corresponds to about a months and a half of model tweaking and hyper-parameter search. In this time we managed to establish a long list of methods that didn't work. It is important to note however that simple models did obtain scores that were very close to the best we achieved. For instance, a single (11, 1)-mismatch kernel for dataset 0, (9, 1)-mismatch for dataset 1 and the described gaussian kernel for dataset 2 got an average of 71.33%. But the rankings being as tight as they were, every decimal point counted, so we had to discard them in favor of complex but marginally better models.

**Substring kernel :** We implemented the Substring Kernel [3] described in class. The main issue with this kernel is that it is computationally heavy and it has two parameters to optimize. After a (long) 2-fold cross-validation without better results than spectrum or mismatch in any of the 3 datasets, we abandoned this kernel.

**Multiple Kernel Learning :** The failure of MKL to outperform our chosen model is a rather interesting fact. Our use of a sum of kernels inspired us to try using a weighted sum of even more kernels and optimizing the weights using SimpleMKL [5]. This approach either matched or did worse on the cross-validation than our selected approach. Our implementation of SimpleMKL was tested on toy 2D point problems and seemed to work. It's hard to explain why it didn't systematically either match or outperform the simple sum of kernels (assuming our implementations are bug-free).

Other honorable mentions include k-spectrum feature embeddings coupled with polynomial kernels and Convolutional Kitchen Sinks [4] (although they were conceived for much larger datasets).

# 4 Conclusion

We have described our chosen approach for tackling the data challenge. Our attempts at improving our results over the last days of the contest led us to believe that maybe we had taken mismatch kernel based methods as far as they can go. A sizable improvement might require a radically different approach and not just hyper-parameter tuning. We're eager to read reports of some other teams to confirm or infirm this belief.

## References


[1] E. Eskin, J. Weston, W. S. Noble, and C. S. Leslie. Mismatch string kernels for svm protein classification. In *Advances in neural information processing systems*, pages 1441–1448, 2003. 1

[2] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001. 1

[3] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444, 2002. 2

[4] A. Morrow, V. Shankar, D. Petersohn, A. Joseph, B. Recht, and N. Yosef. Convolutional kitchen sinks for transcription factor binding site prediction. *arXiv preprint arXiv:1706.00125*, 2017. 2

[5] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9(Nov):2491–2521, 2008. 2