Data Challenge for Kernel Methods Memo

Team Name: Pierres qui roulent n'amassent pas mousse

Pierre BOYEAU
Master MVA
École Normale Supérieure Paris-Saclay
CACHAN, 94230
pierre.boyeau@eleves.enpc.fr

Pierre OREISTEIN
Master MVA
École Normale Supérieure Paris-Saclay
CACHAN, 94230
pierre.oreistein@eleves.enpc.fr

Abstract

This report briefly describes the team's demarch to achieve its results in the Data Challenge Competition. Our main concern was to make our algorithms' implementation usable with any kernel or embedding. That way, experimenting with kernels was much easier. Code can be found here: https://github.com/PierreBoyeau/dna-data-challenge and predictions can be obtained with the function predicitons.py.

Data Augmentation

Over the lecture of different articles, it found that the data can be augmented for potentially improving the results. In particular, it implemented two options:

- **Complementary Sequences**:
  DNA has two strands that code the same genes and are complementary in their amino acids. Therefore, it can augment the datasets provided by adding the complementary sequences and classify them in the same way than the original ones.

  The code can be found in: /Modules/DataAugmentation/ComplementarySequences.py

- **Pertubated Sequences** [CJM+18]:
  DNA is sensitive to mutations. Therefore, some small differences can appear on a sequence without modifying the expressions of the genes coded. Therefore, it can increase the datasets by adding some "pertubed sequences". That is, it perturbs some sequences of the datasets by switching some of its amino acids and it adds these created sequences with the same class into the original datasets.

  The code can be found in: /Modules/DataAugmentation/PertubedSerquences.py

Embeddings

Before to use any model of classification, the sequence can be embedded in a dimension space. Therefore, it implemented different possible embeddings:

- **Spectrum Embedding** [LEN01]:
  In practice, it observed that taking into high order k-mers yielded better results in order to predict binding sites. Yet the theorical number of k-mers for a given k explodes exponentially, while in practice many k-mers are not observed and thus do not need to be represented. It worked intensively on our implementation for solving this issue, allowing computational time (otherwise potentially very long) to be reduced, and high values of k to be taken.

  Inspired by classical techniques used in NLP, it also added an option to use a *TF-IDF* representation of k-mers. The idea behind this is to suppose that only some motifs, present only in binding sites, are responsible for the TF classification of the sequence. *TF-IDF* allows to give an higher weight to such k-mers, while reducing the weight associated k-mers present in all sequences (as their pertinence is probably lower)

  The code of this embedding is available in: Modules/Embeddings/SpectrumEmbedding.py and Modules/Embeddings/MotifEmbedding.py

- **Dimismatch Embedding**[LEC$^+$04]:
  DNA is sensitive to mutations. Therefore, some small difference can appear between two sequences. To take into account these small changes, [LEC$^+$04] improved their previous spectrum embedding for taking into account k-mers that are m-mismatch close to the original one.

  The code is available in: Modules/Embeddings/DimismatchEmbedding.py

Some other embedding are also available like the *Triad* one [TWW$^+$09] but they did not produce any good results.

Kernels

It also wanted, if needed, to enhance the expressiveness of above embeddings. In that sense, it implemented different kernels on these embeddings like **LinearKernel**, **PolyKernel** or **GaussianKernel**. The code can be found in: /Modules/Kernels/

Algorithms

It implemented three kernel methods for the task of classification:

- **SVM**:
  The code can be found in: Modules/Models/KernelSVM.py
- **Kernel Logistic Regression and Kernel**:
  The code can be found in: .../KernelLogisticRegression.py
- **Convolutional Kernel Network** [CJM$^+$18]:
  It implemented the unsupervised version of [CJM$^+$18]. However, it did not achieve any good results. In fact, the computation is quite expensive and this kernel is designed for bigger datasets.

It tooks special care in using pertinent optimization schemes, noticing for instance that Kernel SVM program was quadratic with linear constrains. However, in practice it found that the algorithm choice had little impact on overall performance of the model (between the two first methods).

Model Selection

For the fine-tuning of the hyperparameters of the different functions involved (data augmentation, embeddings, kernel and methods of classification), it implemented a function for **Cross-Validation** and for **GridSearch** in a parallelized version. The code can be found in: /Modules/ModelSelection/

Results

After a lot of experiments, it achieved the best result on the public testing set with the following model:

- **Data Augmentation**: No Data Augmentation.
- **Embedding**: Spectrum Embedding with k-mers of size 5, 7 and 12.
- **Kernel**: Polynomial Kernel with degree 2.
- **Method**: KernelLogisticRegression with $\lambda = 1$.

A minimum of 25 Go of memory is required for launching the code.

References

[CJM$^+$18] Dexiong Chen, Laurent Jacob, Julien Mairal, Dexiong Chen, Laurent Jacob, Julien Mairal, Biological Sequence, Convolutional Kernel, and Laurent Jacob. Biological Sequence Modeling with Convolutional Kernel Networks To cite this version : HAL Id : hal-01632912 Biological Sequence Modeling with Convolutional Kernel Networks. 2018.

[LEC$^+$04] Christina S Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[LEN01] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific, 2001.

[TWW$^+$09] Yingjie Tian, Lingyun Wu, Yong Wang, Ling Jing, Naiyang Deng, and Xiaojian Shao. Predicting DNA- and RNA-binding proteins from sequences with kernel methods. *Journal of Theoretical Biology*, 258(2):289–293, 2009.