Kernel Methods AMMI 2020

Israel Azime, Sokhar Samb
Team_IS
African Master of Machine Intelligence(AMMI)

1 Introduction

In this Kaggle Competition we were working on binary classification task of predicting whether a DNA sequence region is binding site to a specific transcription factor.The sequence-specific binding of transcription factors to transcription factor binding sites (TFBSs) is key to the mediation of transcriptional regulation(1)

In this report we tried to include Experiments worth mentioning and we think we learned about.

2 Experiments

2.1 Data Related Experiments

There are many ways of encoding DNA sequences into ordinal vectors to be used in our machine learning models. We were provided with 2000 training samples and 1000 test samples.

DNA sequence data frequently are contained in a file format called ”fasta” format. Fasta format is simply a single line prefixed by the greater than symbol that contains annotations and another line that contains the sequence. The First thing we tried is to find the best training set representation that will give us good performance on our simple models.

- **One Hot Encoding**:
  This approach encodes each letter as one hot version of it. Which means A=(1, 0, 0, 0), C=(0, 1, 0, 0), G=(0, 0, 1, 0), T=(0, 0, 0, 1). This Sequence was given with the data-set we have as an optional data.

  For Example sequence ”ACG” will be represented as [1,0,0,0,0,1,0,0,0,0,1,0]

- **Ordinal encoding DNA**:
  This Approach encodes each nucleotide as an ordinal value. Which means A=0.25 , C=0.50, G=0.75, T=1.0. We found detailed example in kaggle notebook and used this method to test on our simple data.

  For Example sequence ”ACG” will be represented as [0.25,0.50,0.75]

- **K-mer counting/Spectral Embedding**:
  This method treats long biological sequences as a text. First we break them down into n-gram kind of sequences where each ngram is treated as a word. Then we can apply any language prepossessing tool we want.

  For example sequence ”ACGGTTAA” will be represented as [’ACG’,’CGG’,’GGT’, ’GTT’ ,’TTA’,’TAA’]. We used one hot encoding, count vectorizer , Tf-idf counting on this sequence for our experiments.

- **Count Vectorizer / TfidfTransformer**:
  We experimented with sklearns builtin count vectorizer and tf-idf transformers to see if we can get good embeddings. For both cases we used character level preprocessers with ngram value ranging between 2-10 both counting how many times it exist or exist or not approaches.

Additionally we tried to add additional features like count of each nucleotide but the results doesn’t show any improvement.

Dataset Preprocessing	**Optimized Acc**
One Hot	0.487
Ordinal	0.5175
K-mer-6	0.483
Count-Vectorizer-6	0.6224
Tf-idf	0.5085

Table 1: This table Shows logistic regression model performance on different data pre proceeding methods.

2.2 Algorithm Related Experiments

- **Logistic Regression**: Testing one hot data gave 48% accuracy on our logistic regression model.and Count Vectorizer was able to give 62%. The count Vectorizer is word level with ngram range 6-8 and we choose 6 based on our experiment.

- **Ridge Regression**: In this part we focus more on kernel ridge regression than the standard ridge regression. The motivation behind was to better understand the effect of kernels. We didn't work on Ridge Logistic Regression further because its using the Ridge Regression and we assumed they are comparable in performance.

- **Support Vector Machine(SVM)** :We experimented with SVM algorithm discusses in class on our data. On one hot encoded data we were given by default the performance was 52% percent, which is closer to random accuracy we had. but changing the data to spectral embedding with k-mer size 8 gives 65% .

  We then move on with SVM with kernel that we also tried with all the kernels seen in class that are also listed below.

2.3 Kernel Related Experiments

In our experimentation, we have tried different kernels. Most of the kernels we used are the kernel we have seen in class. In the flowing lines we try to define all the kernels we have tried.

- **Linear kernel**:The linear kernel is one of the simplest kernel. It is given by the inner product between two variables.

- **Polynomial kernel**: The polynomial kernel can be seen as the more general form of linear kernel. It represent a vector data in a space features over polynomial of the original data.

- **Radial basis function kernel(rbf kernel** The rbf kernel is a standard kernel that. It take just one parameter gamma which can be seen as the 'spread' of the kernel.

We explored on a couple of kernels like Gaussian,String kernel and mismatch kernel but we were not able to have result on them.specially string mismatch kernel is very computational intensive.

Kernels -Ridge Regression	Optimized Acc /cv
linear	0.6520
rbf	0.6470
polynomial	0.6585

Table 2: This table Shows Ridge regression accuracy with different kernels on cross validation(k=4) score

Kernels - SVM	Optimized Acc /cv
linear	0.6525
rbf	0.6530
polynomial	0.6315

Table 3: This table Shows Kernel SVM accuracy with different kernels on cross validation(k=4) score on spectral data

2.4 Training and Optimization Related Experiments

Our best accuracy score on private score was achieved using Kernel-Regression With spectral data of kmer size 8 using rbf kernel .

To find the best parameters that can improve our result, we did hyper-parameter search for all the algorithm we tried. This help to improve a bit our performance. We used Optuna, an open source hyper-parameter optimization framework for this task. We also did cross validation for hyper-parameter turning and make sure our model is not over fitting to much.

3 Result Discussion

Model	Private	Public
rbf + svm	0.64200	0.65400
linear + ridge regression	0.63600	0.66000
rbf + svm	0.66000	0.65600

Table 4: The fist two shows the submissions we choose for final evaluation. We failed to choose the third one (which gives 0.654/0.65 on private/public) and later improved when we test it with the same hyper-parameter after the competition ended.

References

[1] Daniel Usvyat Narayan Jayaram and Andrew C. R. Martin*. 2016. Evaluating tools for transcription factor binding site prediction. (2016).

[2] Working with DNA sequence data for machine learning - an introduction