Training the Logistic Regression model to classify pairs of genes from Bacillus subtilis that belong to the same operon

Hatem Nassrat

B00393388

nassrat@torch.cs.dal.ca

nassrat@cs.dal.ca

Table of Contents

Abstract.....3

Introduction.....3

Logistic Regression Model.....4

Training the model.....4

Accuracy of the model.....5

Experiment & Results.....5

Conclusion.....6

Appendix A.....7

Appendix B.....10

References.....13

Training the Logistic Regression model to classify pairs of genes from Bacillus subtilis that belong to the same operon

Abstract

Logistic Regression is a supervised learning approach that tries to classify data into one of N classes using a weighted sum of predictor variables. Currently the implemented model deals with classification for N=2, with as many predictor variables as needed. This model can be used to analyze certain features of the gene, for example to find if the gene-pair is part of the same operon (a set of adjacent genes on the same strand of DNA transcribed into the same mRNA molecule (2.4 genes in bacteria on average)) or not. The aim of this project is to build a supervised learning classifier using appropriate predictor variables, training set, validation set and test it using K-fold cross validation.

Introduction

The logistic regression model acts as a powerful tool in classification. Today, due to the popularity of the model, exists a large number of software libraries written to aid researchers in using this model in analyzing their data. Such software is available for clusters (used in analyzing data of high dimensions) and also micro-computers. In this paper, the linear regression libraries of the Biopython module were used. In this module, the logistic regression model is implemented for two classes. This is useful in our analysis, as we are aiming to create a model to predict weather a pair of genes belong to the same operon or not. Therefore we are to classify the pairs into two classes OP (belonging to the same operon) and NOP (not belonging).

An operon is a genetic structure that contains one or more structural genes which are transcribed into one polycistronic mRNA (a molecule that codes for more than one protein). This structure is important when understanding how proteins are generated within a living organism. As can be seen in the figure bellow, associated with an operon, is a promoter sequence, operator sequence, the structural genes, and a terminator sequence. This is important when considering how the proteins coded within the structural genes are formed. In the organism carrying this operon exists an RNA polymerase. This polymerase is responsible for copying the data represented in the structural

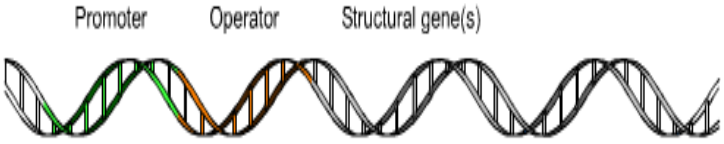

Figure 1: Operon [1]

genes of an operon into the respective protein's genetic code, thus transcribing the correct function onto the protein. The RNA polymerase identifies the promoter sequence and binds to the operon at that segment, the polymerase then traverses the helix reading the sequence that follows. The operator sequence is responsible for telling the polymerase which genes, from the ones that follow, are to be activated and which are to be suppressed. Some of the genes that follow are never expressed in the organism that carries it, due to them not being activated in their respective operator sequences. The genome of living organisms consists of multiple of such operon sequences. The SVM to be implemented, is aimed to be able to classify if a pair of genes belong to one of these groupings, or are actually situated in different groups. The operons to be used will be gathered from known Bacillus subtilis (a spore-forming bacterium found in soil) operon.

Logistic Regression Model

To use this linear classification approach, we need to decide on the input variables to be used to classify the data. These input variables should be measurable and also related to the operon structure. The first variable would be the distance (in base pairs) between the genes [3]. Genes belonging to the same operon would not be far away from each other. However, genes of different operon tend to have a large distance between them to allow for the promoter, operator and terminator sequences. Another variable to be used is the gene expression profile. Genes belonging to the same operon, by definition, have equal gene expression profiles [3]. Never the less, genes of different operon have different gene expression profiles. Therefore we now have two variables that aid us in the prediction that a pair of genes belong to class OP or NOP.

The logistic regression model allows us to do such a classification by computing a score that combines all the variables that aid in the classification. The score is computed as a sum of the input variables, multiplied with appropriate parameters (see figure 2). These parameters (the beta coefficients) are approximated through Maximum likelihood analysis of the training data.

$$S = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Figure 2: logit score [3]

Training the model

Training of the model is intrinsically computing the beta coefficients that will best fit the data distribution without over fitting the data. The training data is available, which consists of both the inputs and the outputs. Therefore, the three variables can be estimated and used to create the classifier. The training data, has been gathered from [4] focusing on Bacillus subtilis operon. Therefore the classifier generated would work best when classifying genes from this organism's genome. The effort in [4] gathers many known operon information, for multiple organisms. However, not all such information

has been found out. For example, in the case of the Bacillus subtilis microorganism, only around 10% of the gene operon structure has been identified. The classifier created via this project may be used to classify the remaining part of the organism's genome. The gathered data can be seen in Appendix A.

Training the model using the entire dataset, gave the following beta parameters: $\beta_0 = 0.62123273438857318, \quad \beta_1 = -0.00074250525769372971, \quad \beta_2 = 4.2325169271300087$, thus giving us our Logistic Regression function. The parameter β_1 is negative, as gene pairs of shorter distances are to be classified as OP (or binary 1). However, β_2 is positive due to the fact that genes of the same operon tend to have a high similarity score (gene expression score).

Accuracy of the model

To calculate the accuracy of the model, Cross-validation will be used. However, two forms of this procedure will be looked at. The first being the standard K-fold cross validation, with K=10. This method divides the data into 10 equal subsets, training the model on 9 of the sets and using the last as a validation set. This process is repeated 10 times, such that each one of the 10 subsets of the data would be used once as the validation set. The best and worst run will be discarded, leaving 8 runs to be analyzed. Specificity (probability of correctly classifying NOP), sensitivity (probability of correctly classifying OP), false positive rate (rate at which to expect a false positive), and false negative rates of the model will be analyzed for each run in the K-fold. The average of these values will be computed to summarize the quality of the classifier.

The second measure of accuracy would be a leave-one-out cross validation approach. This approach is identical to the K-fold cross validation where K in this situation equals N, and N is the size of the input set. For this approach, the above measurements would be calculated, along with a basic accuracy measurement that combines both the sensitivity and specificity variables. This would be calculated by dividing the number of false predictions by the number of total predictions. The section bellow includes the data gathered through the classification process.

Experiment & Results

The experiment was setup as described above, training the model using the data seen in appendix A. Using the techniques described in the previous section. The output of the application can be seen and explained in Appendix B. There all the results of the cross validation can be seen. Bellow is the analysis of the system based on the results.

After taking the average of the 10-fold cross validation removing the two runs that performed best and worst, the average False positive rate was around 19%. Which is a significantly high error rate. Also the average specificity, the probability that the model correctly classifies a pair of type NOP, is 0.808. These two results may be due to the lower amount of training points supplied for class NOP as opposed to the points supplied for class OP. The average false negative rate, across the cross validation was around 4% with an associated Sensitivity, the probability that the model correctly classified a pair of type OP, of 0.96.

```
Leave One Out Cross Validation:
        Accuracy =  0.896296296296
        False +ve rate =  0.176470588235
        False -ve rate =  0.0595238095238
        Sensitivity =  0.940476190476
        Specificity =  0.823529411765
```

Figure 3: program output

The values for the leave one out cross validation can be seen in the figure bellow. As can be seen, the accuracy of the model calculated through this type of validation is around 90%, with a sensitivity of 94% and a specificity of 82%. This classifier seems to better classify genes of the same OP rather than ones of different OP. Reasons (other than the one listed above) may be that the NOP gene pairs chosen for the training were genes that are close to the operon and not very distant. Thus reducing the diversity in the input variables of pairs of class OP and pairs of class NOP.

Conclusion

Our logistic regression model performed well according to the results and thus may be used to classify with good accuracy, unknown genes of Bacillus subtilis. However, two input variables may not be sufficient for such analysis, more variables may result in a better classifier. The Operon Database [4], where the data has been collected for the generation of this model, has been built with such classifiers included. They use 5 variables to do this classifications and the classifiers are available on their website for use by biochemist in need of such a service. They have a large number of classifier for different organisms, and therefore may perform better than a universal classifier, also the existence of a universal classifier may not be plausible since different organisms may have different operon structure and not be classifiable by such a classifier. Therefore such classifiers are important in the study of genomics and have been implemented along side a large database of operons, which allows for a larger set of training data and increased accuracy.

Appendix A

Note: The input data consists of the gene identifiers (can be used to locate the gene sequence and other information from a gene bank such as NCBI) of the pair of genes, followed by a tuple representing the input variables x_1 (pairwise distance) followed by x_2 (gene expression score). Following the tuple is the corresponding class the two genes fall into (i.e. the output variable to be modeled).

Format:

$GeneID_1$,$GeneID_2$: [x_1,x_2],y

BG13387,BG13379: [8607,0.25],0
BG10137,BG10142: [2672,0.69],1
BG10142,BG10143: [187,0.65],1
BG10140,BG12701: [809,0.72],1
BG13387,BG13382: [6478,0.35],0
BG10156,BG10157: [421,0.73],0
BG10137,BG10139: [600,0.78],1
BG13380,BG13382: [771,0.86],1
BG10774,BG10775: [4,0.58],1
BG14123,BG14124: [25,0.47],1
BG10138,BG10140: [865,0.54],1
BG10137,BG12701: [3127,0.54],1
BG10139,BG12701: [1648,0.51],1
BG11248,BG11820: [3296,0.01],0
BG14126,BG12611: [26,0.59],1
BG14126,BG14123: [4647,-0.31],0
BG13388,BG13377: [10694,0.5],0
BG13388,BG13382: [6762,0.46],0
BG13377,BG13378: [51,0.94],1
BG13377,BG13381: [2680,0.82],1
BG14127,BG14123: [5582,-0.30],0
BG11930,BG13378: [1127,0.92],1
BG13388,BG13380: [8019,0.54],0
BG13379,BG13382: [1643,0.87],1
BG10137,BG10138: [16,0.8],1
BG10156,BG10158: [1177,0.51],0
BG13386,BG13382: [5860,0.5],0
BG13386,BG13379: [7989,0.6],0
BG10140,BG10144: [2135,0.45],1
BG13387,BG13377: [10410,0.14],0
BG14125,BG14123: [2868,-0.23],0
BG11930,BG13382: [4522,0.89],1
BG12043,BG12045: [49,-0.44],0
BG13377,BG13379: [1212,0.93],1
BG11930,BG13380: [2980,0.88],1
BG10140,BG10141: [-5,0.61],1
BG13388,BG13378: [9497,0.55],0
BG10139,BG10142: [1193,0.72],1
BG10772,BG10774: [928,0.86],1
BG14128,BG14126: [653,0.09],1
BG12701,BG10143: [26,0.72],1
BG10139,BG10143: [1881,0.59],1
BG13378,BG13380: [707,0.92],1
BG13378,BG13381: [1483,0.83],1
BG11821,BG13466: [66,0.44],1
BG13388,BG13379: [8891,0.51],0
BG11248,BG11821: [2624,-0.17],0
BG11930,BG13379: [2288,0.92],1
BG10138,BG12701: [2529,0.45],1
BG12043,BG12046: [2,-0.46],0
BG10772,BG10775: [2006,0.58],1
BG10154,BG10158: [3529,0.37],0

BG13377,BG13382: [3446,0.80],1
BG10771,BG10775: [3049,0.56],1
BG11820,BG11821: [21,0.69],1
BG12611,BG14123: [3814,-0.44],0
BG14128,BG14127: [6,0.31],1
BG14128,BG14124: [5482,-0.01],0
BG14126,BG14125: [849,0.29],1
BG13378,BG13382: [2249,0.83],1
BG14127,BG12611: [961,0.14],1
BG10138,BG10142: [2074,0.65],1
BG10771,BG10774: [1971,0.83],1
BG10155,BG12702: [1403,0.56],0
BG11930,BG13377: [14,0.93],1
BG10137,BG10141: [2313,0.65],1
BG10632,BG10630: [1120,0.37],0
BG10138,BG10139: [2,0.7],1
BG10139,BG10141: [834,0.59],1
BG10137,BG10144: [4453,0.42],1
BG10137,BG10140: [1463,0.67],1
BG10156,BG12702: [6,0.51],0
BG10138,BG10143: [2762,0.54],1
BG14127,BG14124: [4828,0.37],0
BG10139,BG10144: [2974,0.48],1
BG10771,BG10772: [110,0.74],1
BG10141,BG10143: [687,0.61],1
BG10138,BG10144: [3855,0.46],1
BG11820,BG13466: [738,0.62],1
BG14125,BG14124: [2868,-0.09],0
BG11248,BG13466: [2327,-0.30],0
BG13386,BG13380: [7117,0.69],0
BG10139,BG10140: [-16,0.72],1
BG13387,BG13380: [7735,0.2],0
BG14126,BG14124: [3893,-0.30],0
BG14127,BG14125: [1784,0.12],1
BG11249,BG11821: [3156,-0.10],0
BG12043,BG12044: [1,0.45],1
BG10154,BG10157: [2773,0.59],0
BG10771,BG10773: [1049,0.76],1
BG10773,BG10774: [7,0.84],1
BG13386,BG11930: [10868,0.5],0
BG13381,BG13382: [-17,0.91],1
BG13386,BG13377: [9792,0.53],0
BG10773,BG10775: [1085,0.68],1
BG10140,BG10142: [354,0.7],1
BG12044,BG12046: [9,-0.40],0
BG10138,BG10141: [1715,0.68],1
BG13388,BG11930: [11770,0.47],0
BG10142,BG10144: [1280,0.59],1
BG10141,BG12701: [454,0.65],1
BG13387,BG11930: [11486,0.22],0
BG13386,BG13381: [6329,0.51],0
BG12044,BG12045: [51,-0.49],0
BG13386,BG13378: [8595,0.63],0
BG10142,BG12701: [-46,0.77],1
BG10155,BG10157: [1818,0.61],0
BG13377,BG13380: [1904,0.86],1
BG12045,BG12046: [1,0.83],1
BG14128,BG12611: [1615,0.17],1
BG10155,BG10158: [2574,0.25],0
BG12611,BG14125: [16,0.19],1
BG12611,BG14124: [3060,-0.40],0
BG10140,BG10143: [1042,0.6],1
BG13380,BG13381: [5,0.83],1
BG12701,BG10144: [1119,0.56],1
BG13378,BG13379: [15,0.93],1
BG10137,BG10143: [3360,0.49],1
BG13379,BG13381: [877,0.90],1
BG11249,BG11820: [3828,0.02],0
BG11249,BG13466: [2857,0.39],0
BG14127,BG14126: [-1,0.16],1
BG13388,BG13381: [7231,0.42],0
BG14128,BG14125: [2438,-0.06],1
BG11930,BG13381: [3756,0.82],1
BG13387,BG13378: [9213,0.23],0

BG10154,BG12702: [2358,0.63],0
BG10141,BG10144: [1780,0.64],1
BG13379,BG13380: [101,0.93],1
BG14128,BG14123: [6236,0.16],0
BG11248,BG11249: [88,-0.24],1
BG13387,BG13381: [6947,0.43],0
BG10772,BG10773: [6,0.89],1
BG10141,BG10142: [-1,0.68],1
BG10143,BG10144: [94,0.64],1

Appendix B

Note: the data displayed bellow corresponds to the output generated from the learner used to classify the genes into classes OP and NOP. Each of the first 10 segments labeled 0-9 are the respective runs of the 10-fold cross validation. The data represented in the tuple bellow the number of False positive/negatives represents the probability, given by the model that forced its false decision. For example in the first validation run, there was one pair that generated a false positive, and the model gave a probability of 0.97184624361252814 for that pair being of class OP (run 2 & 5 have been removed from the analysis).

```
Using set  0  for validation yeilds:
        1  false +ves
        [0.97184624361252814]
        0  false -ves
        []
        False +ve rate =  0.333333333333
        False -ve rate =  0.0
        Sensitivity =  1.0
        Specificity =  0.666666666667
Using set  1  for validation yeilds:
        1  false +ves
        [0.87938392011210764]
        0  false -ves
        []
        False +ve rate =  0.142857142857
        False -ve rate =  0.0
        Sensitivity =  1.0
        Specificity =  0.857142857143
Using set  2  for validation yeilds:
        0  false +ves
        []
        0  false -ves
        []
        False +ve rate =  0.0
        False -ve rate =  0.0
        Sensitivity =  1.0
        Specificity =  1.0
Using set  3  for validation yeilds:
        0  false +ves
        []
        0  false -ves
        []
        False +ve rate =  0.0
```

```
        False -ve rate =  0.0
        Sensitivity =  1.0
        Specificity =  1.0
Using set  4  for validation yeilds:
        1  false +ves
        [0.86984711709261309]
        0  false -ves
        []
        False +ve rate =  0.333333333333
        False -ve rate =  0.0
        Sensitivity =  1.0
        Specificity =  0.666666666667
Using set  5  for validation yeilds:
        2  false +ves
        [0.81573098675181988, 0.95753680818561115]
        2  false -ves
        [0.79243663491436211, 0.64998036283601768]
        False +ve rate =  0.666666666667
        False -ve rate =  0.2
        Sensitivity =  0.8
        Specificity =  0.333333333333
Using set  6  for validation yeilds:
        1  false +ves
        [0.75577179051837973]
        1  false -ves
        [0.54412317583981473]
        False +ve rate =  0.142857142857
        False -ve rate =  0.166666666667
        Sensitivity =  0.833333333333
        Specificity =  0.857142857143
Using set  7  for validation yeilds:
        0  false +ves
        []
        0  false -ves
        []
        False +ve rate =  0.0
        False -ve rate =  0.0
        Sensitivity =  1.0
        Specificity =  1.0
Using set  8  for validation yeilds:
        1  false +ves
        [0.87172158780252151]
        0  false -ves
        []
        False +ve rate =  0.25
```

```
        False -ve rate =  0.0
        Sensitivity =  1.0
        Specificity =  0.75
Using set  9  for validation yeilds:
        2  false +ves
        [0.53180125133631029, 0.83074765659859973]
        1  false -ves
        [0.83361024622800506]
        False +ve rate =  0.333333333333
        False -ve rate =  0.142857142857
        Sensitivity =  0.857142857143
        Specificity =  0.666666666667
Leave One Out Cross Validation:
        Accuracy =  0.896296296296
        False +ve rate =  0.176470588235
        False -ve rate =  0.0595238095238
        Sensitivity =  0.940476190476
        Specificity =  0.823529411765
```

References

1. Operon. from http://en.wikipedia.org/wiki/Operon

2. Garson G. Logistic regression. 1998. from http://www2.chass.ncsu.edu/garson/pa765/logistic.htm

3. Hoon M. The logistic regression model. from
http://www.biopython.org/DIST/docs/cookbook/LogisticRegression.html

4. Okuda S. Operon DataBase.

5. Schneider J. Cross validation. 1997. from http://www.cs.cmu.edu/~schneide/tut5/node42.html