Label Propagation for Eukaryotic Splice Junction Identification

Xiaojin Zhu

December 5, 2002

Abstract

We study the use of unlabeled data to help classification in a bioinformatic problem of splice junction identification. We derive two distance measures for DNA segments. With these distance measures, we compute the accuracy of label propagation algorithm which incorporates unlabeled data. We compare the accuracy to several baselines. We found that for this dataset, using unlabeled data does not significantly improve accuracy. We give a possible reason for the observation.

1 Introduction

Many computational biology tasks can be cast as classification problems. Traditional classifiers, e.g. neural networks, decision trees etc., need relatively large amount of LABELED training examples. However labels are often hard / expensive / slow to obtain. In contrast, unlabeled examples are often available in large quantity. It is therefore desirable to devise new machine learning algorithms that use relatively little labeled training examples and take advantage of large amount of unlabeled data. We have proposed such an algorithm, label propagation [1], and showed its effectiveness.

In this project, we apply label propagation to the computational biology problem of eukaryotic splice junction identification. We compare the classification accuracy of using unlabeled data versus not using them, under the same amount of labeled data.

2 The dataset

The dataset is the Primate splice-junction gene sequences dataset, available at the UCI machine learning data repository [2]. There are 3190 examples in this dataset. Each example is a sequence of 60 nucleotides, and has one of three labels:

- EI (the middle of the sequence is an exon → intron boundary)
- IE (the middle is an intro → exon boundary)
- NONE (not a boundary)

There are 25% EI, 25% IE, and 50% NONE examples in the dataset. Here are four examples:

```
EI,    TARHBD-DONOR-468,
GGAAGATGTTGGTGGTGAGGCCCTGGGCAGGTCAGTATCATGGCTATGAGGCAGGCTTAA

EI,    TARHBD-DONOR-817,
AAGCTGCATGTGGATCCTGAGAACTTCAGGGTAAGTCTAGGAGATGTTCTTCTTTGTCCT

IE,    ATRINS-ACCEPTOR-701,
TTCAGCGGCCTCAGCCTGCCTGTCTCCCAGGTCTCTGTCCTTCCACCATGGCCCTGTGGA

IE,    ATRINS-ACCEPTOR-1678,
GGACCTGCTCTGCGTGGCTCGCCCTGGCAGTGGGGCAGGTGGAGCTGGGTGGGGGCTCTA
```

3 Computational approach

The gist of the label propagation algorithm is to exploit the distribution of large amount of unlabeled data points. The assumption is that classes tend to form dense 'clumps'. The few labeled data act as 'seeds' and propagate labels along such clumps. Please refer to [1] for technical details of label propagation.

We randomly split the dataset into training and test parts. We train a classifier by label propagation to predict the class (EI, IE, or NONE) of a sequence. In particular, we perform a set of experiments, with different training data size $|L|$ ranging from extremely small ($|L|$=3, i.e. one labeled training example per class), to reasonably large (e.g. $|L|$=100). For each $|L|$, we perform 10 trails. In each trail, we randomly select $|L|$ examples from the 3190 examples, and use the rest $3190 - |L|$ examples as unlabeled data (and test data). We then train our model on the training set and test it on the unlabeled data. The resulting accuracy is averaged over the 10 trails.

The classification accuracy on test set is the measure we will use. It is the percentage of examples in the unlabeled data correctly classified by our model. If the label propagation algorithm is successful, the outcome would be that with large $|L|$, our model and more traditional algorithms will have comparable accuracies; but with small $|L|$, our model will outperform traditional algorithms since we also use unlabeled data.

We will compare the accuracy by label propagation to the following traditional learning algorithms, under the same setting (same training/test split): k-Nearest-Neighbor (kNN) and Radial basis function kernel classifier (RBF).

We pick these algorithms because they use the same problem specific knowledge as label propagation. An explanation of the two baseline algorithms is given below:

1. k-Nearest-Neighbor (kNN)

   Let $d(x, x')$ be the distance between two samples x, x'. To classify x_{new}, find k closest labeled samples $l_1, ..., l_k$ to x_{new} in terms of $d()$. Classify x_{new} by the majority class in $l_1, ..., l_k$. If there are ties, break them randomly. In our experiment, we compute 1NN, 3NN, 5NN, 7NN and 9NN.

2. Radial basis function kernel classifier (RBF)

   Let $K(x, x') = \exp(-\frac{d(x,x')^2}{\sigma^2})$ be the Gaussian radial basis function kernel, where σ is the band-width parameter. The probability that x_{new} is class c is then

   $$P(c|x_{new}) \propto \sum_{l \in c} K(l, x_{new})$$

   where the sum is over all labeled samples of class c. Intuitively, a Gaussian 'bubble' is put on top of every labeled points. At a new point x_{new}, the influence of different classes is the sum of the bubbles at that point.

4 Distance measures

A crucial step is to design a distance measure between samples. Our samples are sequences of nucleotides of length 60. We experimented with two types of matrices.

4.1 Gaussian weighted distance

Given two sequences $x[i], y[i], i = 1 \dots 60$, the Gaussian weighted distance is

$$d(x, y) = \sum_{i=1}^{60} w_i(1 - \delta(x[i], y[i]))$$

where $\delta(a, b) = 1$ if $a = b$, and 0 otherwise. The weights w_i are Gaussian centered at the middle of the sequence, with variance 20:

$$w_i \propto \exp(-\frac{(i-30)^2}{20})$$

This distance measure is extremely simplistic. It merely treats nucleotides around the boundary more seriously. It doesn't use any deeper biological knowledge.

4.2 Knowledge based distance

In [3] the authors mentioned the following well known splice junction model:

```
 Exon    -->              Intron                       -->   Exon
(A/C) A G | G T (A/G) A G T ... (C/T)6 X (C/T) A G | G (G/T)
```

We will craft a distance measure with the above biological knowledge. It tells us which positions to check (not all 60 positions are important), and certain nucleotides are exchangeable at certain positions.

In particular, if we are looking for an Exon $\to$ Intro boundary,

```
  28   29 30   31 32   33    34 35 36
(A/C) A   G | G   T  (A/G) A   G   T
```

we only need to check position 28 to 36. In addition, at position 28 A and C are equivalent; at position 33 A and G are equivalent. Therefore we define a distance

$$d_{EI} = \sum_{i=28}^{36} d_{EI}^{i}(x[i], y[i])$$

where $d_{EI}^{28}(a,b) = 0$ if $a = b$ or $a, b \in [A, C]$, =0 otherwise; similarly for $d_{EI}^{33}()$ with A, G equivalent. For other positions, $d_{EI}^{i}(a,b) = 1 - \delta(a,b)$.

On the other hand, if we are looking for an Intron $\to$ Exon boundary,

```
21-26   27   28   29 30   31 32
(C/T)6 X (C/T) A   G | G (G/T)
```

we need to check position 21 to 32, but skip 27. Position 21-26, 28 and 32 has equivalent nucleotides similar to those above. We define

$$d_{IE} = \sum_{i=21,-27}^{32} d_{IE}^{i}(x[i], y[i])$$

with $d_{IE}^{21}(a,b) = 0$ if $a = b$ or $a, b \in [C, T]$, =0 otherwise; and so on.

Finally, since we are looking for both IE and EI boundary at the same time, we combine the two by defining the distance to be

$$d(x,y) = \min(d_{EI}(x,y), d_{IE}(x,y))$$

4.3 The distance matrices on dataset

We compute all pairwise distance on the dataset with the two distance measures respectively, and plot the two distance matrices in Figure 1 and Figure 2. The samples in the dataset are sorted so all EI (25%) come first, then IE (25%) and finally NONE (50%). Both axes are samples, and distance is color coded where blue means small distance and red large distance. We have the following observations:

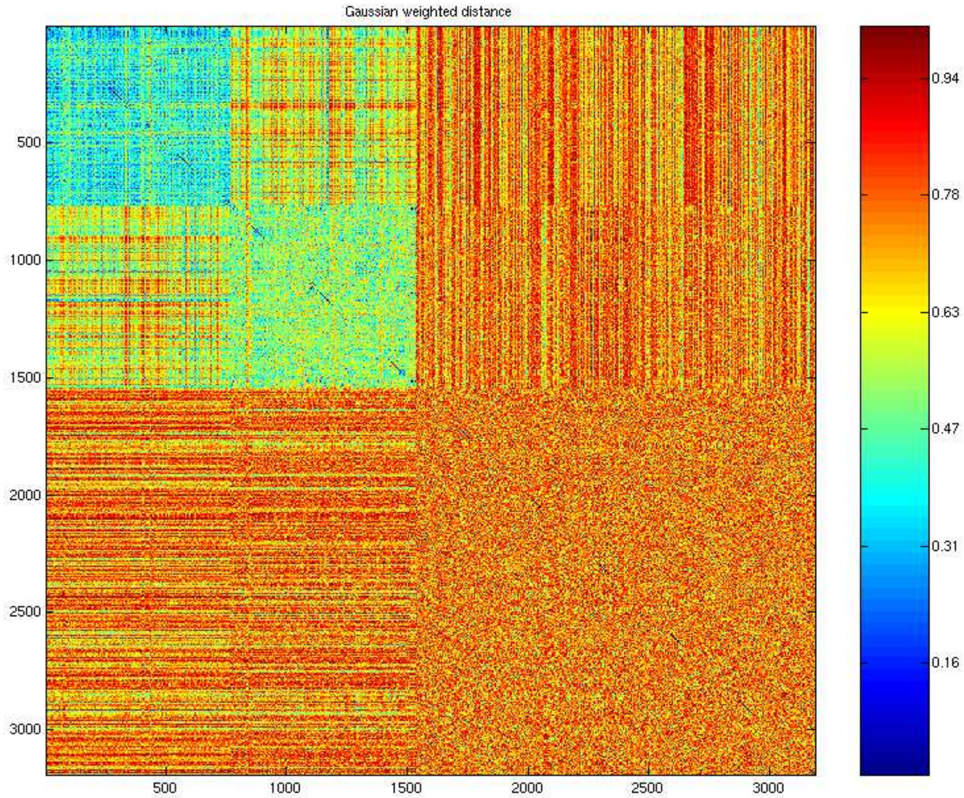

Figure 1: Gaussian weighted distance matrix

1. The matrices are roughly block diagonal for classes EI and IE. This is good since examples in these classes are tightly clustered.

2. The class NONE is not clustered. In fact, it has very large between-sample distances. This is not surprising consider that samples in this class are arbitrary DNA sequences, as long as they don't contain an EI or IE boundary. Thus class NONE is a 'background' class with samples filling up the whole data space. This observation has serious consequences for label propagation, which assumes classes form tight clumps.

3. The knowledge based distance (Figure 2) looks better because EI and IE classes are tighter (bluer).

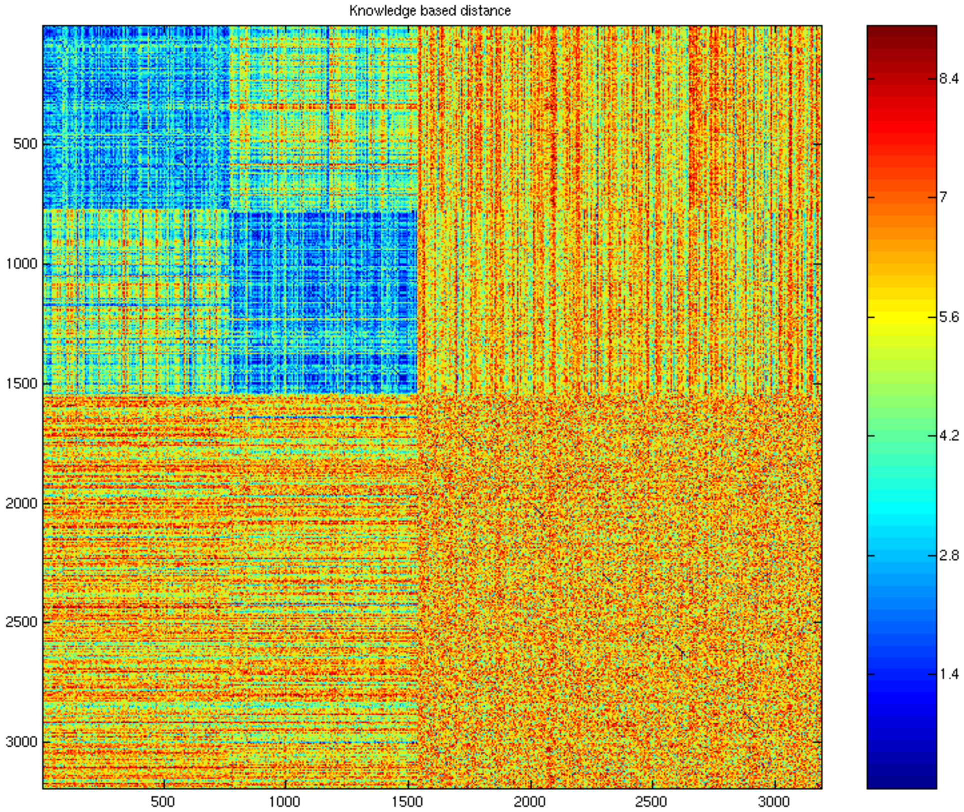

Figure 2: Knowledge based distance matrix

5 Experiment Results

We plot the experiment results with Gaussian weighted distance and Knowledge based distance in Figure 3 and Figure 4 respectively. The band-width parameter is 0.16 and 1.5 respectively. Each curve is the accuracy of an algorithm. The error bars are ± 1 standard error in 10 trials.

In each plot, we can separate algorithms into 4 groups:

1. CMNo, CMNe

   These two are label propagation with post processing to normalize class proportions, one with estimated class prior, the other with oracle prior [1]. They are supposed to be above all other algorithms. However the result is mixed: with the simplistic distance measure, they eventually outperform baselines when $|L| > 60$ (Figure 3); but with the knowledge based distance they are inferior to RBF baselines (Figure 4).

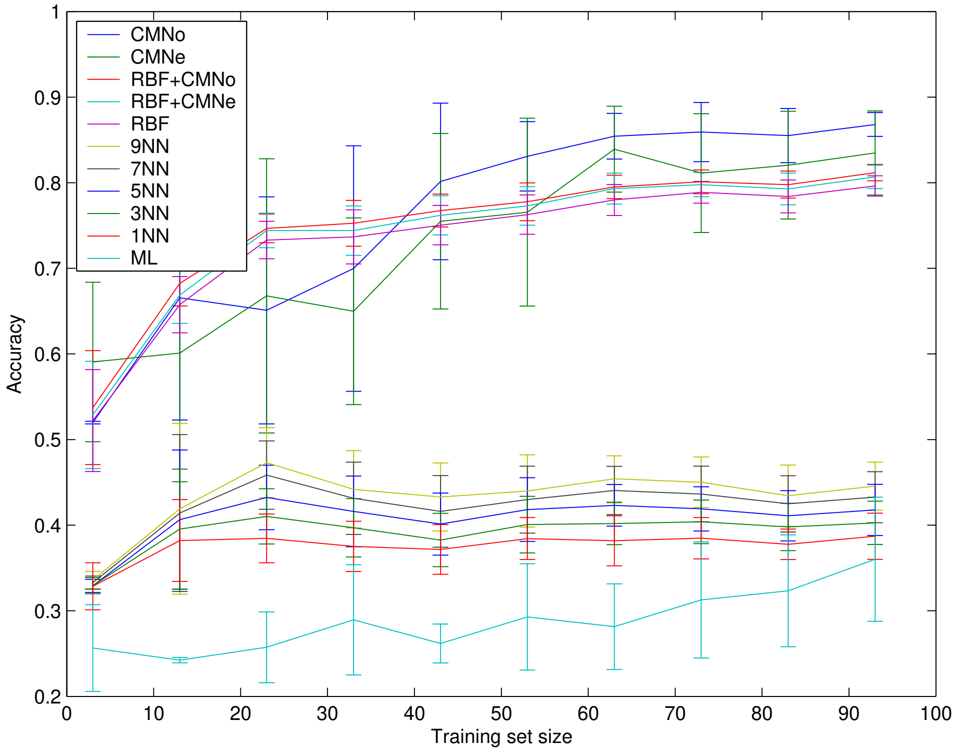

Figure 3: Results with Gaussian weighted distance matrix

2. RBF+CMNo, RBF+CMNe, RBF

   These are RBF baselines, with and without class proportion normalization on the RBF class probabilities. Their accuracies are very similar. They are better than kNN.

3. 1NN, 3NN, 5NN, 7NN, 9NN

   kNN forms the next group. They are much worse than RBF, and worse than label propagation with post processing. Among kNN, larger k gives slightly better accuracy.

4. ML

   This is label propagation without post processing, i.e. the solution to the random walk as is. It has serious unbalanced class proportions, and is thus the worst in terms of accuracy. This is consistent with our observation on some other datasets.

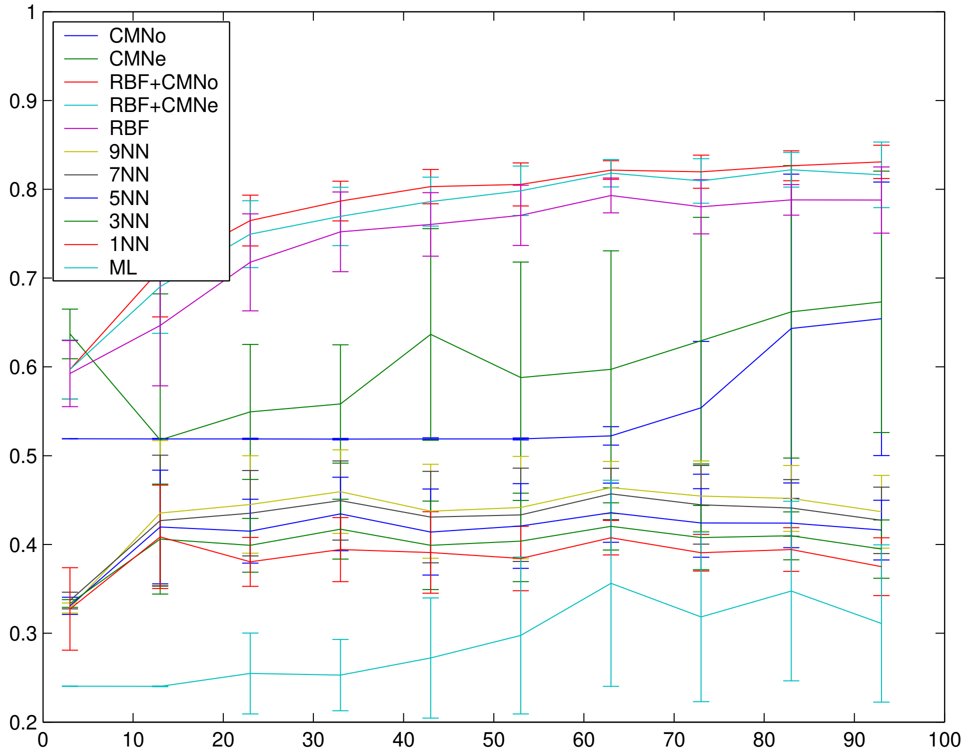

Figure 4: Results with Knowledge based distance matrix

Label propagation (CMNo, CMNe) did not outperform RBF baseline in all cases. We postulate that the main reason is the sparse distribution of the NONE class. Label for this class cannot propagation easily. The fact that the crude Gaussian weighted distance measure seems to be no worse than the sophisticated distance motivated by biological knowledge is also surprising.

6 Related work

Previous studies on this problem used weight-matrix method [4], statistical methods [5], neural-networks and decision tree [6], and knowledge-based artificial neural networks [3]. They are supervised learning methods and cannot utilize unlabeled data. They also have (the equivalent of) different built-in distance metrics. In contrast, our label propagation algorithm used unlabeled data, but needs a distance metrics as input. The distance measures we used for this project are probably far from optimal. It is hard to compare performances, because most previous studies do not operate with so few training examples.

In [3], the authors report approximate accuracies of 90% with KBANN (neural nets with weights initialized by expert knowledge), and 72% with random initial weights neural nets, when $|L| = 50$. Our CMNe (estimated prior, the more realistic case) with Gaussian weighted distance achieves 76%. With $|L| = 100$, KBANN's accuracy is 92%, random initial weights neural nets 82%, and CMNe 83%. We remark that the comparison between CMNe and RBF is of greater interest in our study.

7 Summary

We studied the performance of label propagation on splice junction prediction problem. The results did not show much advantage by incorporating unlabeled data. We suspect the reason is that the 'background' class NONE does not obey the assumption of our algorithm. How to handle such distribution needs further research. We also found the simple Gaussian weighted distance measure is comparable to a more biology knowledge based distance measure. A better distance measure is also a future research direction.

References

.

[1] Learning from Labeled and Unlabeled Data with Label Propagation. Xiaojin Zhu, Zoubin Ghahramani. CMU CALD tech report CMU-CALD-02-107, 2002. Available online at http://www.cs.cmu.edu/ zhuxj/pub/propagate.ps.gz

[2] Blake, C.L., Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[3] M. O. Noordewier and G. G. Towell and J. W. Shavlik, 1991; "Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences". Advances in Neural Information Processing Systems, volume 3, Morgan Kaufmann.

[4] R. Staden. Computer methods to locate signals in DNA sequences. nucleic Acids Research, 12:505-519, 1984

[5] K. Nakata, M. Kanehisa, C. DeLisi. Prediction of splice junctions in mRAN sequences. Nucleic Acids Research, 13:5327-5340, 1985

[6] A. Lapedes, D. Barnes, C. Burks, R. Farber, K. Sirotkin. Application of neural networks and other machine learning algorithms to DNA sequence analysis. Computers and DNA, page 157-182. Addison-Wesley, 1989