Use of supervised learning methods for transcription factor binding site detection

Nathan Snyder

Uconn BioGrid, REU Summer 2009
Department of Computer Science and Engineering
University of Connecticut, Storrs, CT 06269

1Computer Science Department, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213

Abstract. Locating transcription factor binding sites is crucial to understanding the regulatory network of a cell. Several techniques have been proposed for finding additional binding sites based on the known binding sites for a particular transcription factor and shown to produce useful results. However, these techniques require that all training sequences be of the same length, which is not always true in real data sets. This paper considers modifications to these algorithms which will allow them to operate using training sequences of varying length while still maintaining fast execution times and reasonable accuracy. Specifically, we develop the strategy of first aligning the mixed length training sequences and then using them to generate a scoring matrix in which each position is weighted by the number of training sequences aligned there. A specific implementation of this strategy, MLCentroid, is created and tested using a leave one out testing procedure. MLCentroid is shown to have efficient execution time and provide reasonably accurate results.

1 Introduction

In cells, the production of particular proteins is regulated by controlling how often the genes that codes for those proteins are transcribed. This control is accomplished by having proteins called transcription factors bind the DNA near that gene and either enhance or reduce the that gene's ability to be transcribed. One transcription factor often binds to multiple sites in the cell's genome, and we call those genes coregulated. Discovering which genes are coregulated is an important step in understanding the overall behavior of a cell, and for this reason a variety of motif search algorithms have been designed to locate potential transcription factor binding sites.

All of these algorithms rely on the fact that if a particular DNA motif is involved in regulating the cell then it will appear more often than a motif which does not have any biological significance [8]. Some of these algorithms fall into the category of supervised learning algorithms, which build a model from training data. More specifically, these algorithms take as input a set of DNA sequences which are known to be binding sites for a particular transcription factor. Then, they search through genetic data to find sites similar to the training sequences, since these are likely to also be binding sites for that transcription factor. The mathematical definition of similarity varies from algorithm to algorithm, so the different binding site search tools can produce very different results.
Osada et al. tested several of these algorithms by giving them the same training data and sequences to score and compared how well each algorithm did at scoring known transcription factor binding sites above other sequences [5]. All of the algorithms tested relied on the assumption that each training sequence would have the same length, which was true for the test data used. However, this is not always the case. An example where the same length assumption does not hold is the data sets of known transcription factor binding sites used by Tompa et al. in their evaluation of various de novo motif discovery tools, which came from the TRANSFAC database [8]. The goal of my project was to modify the algorithms evaluated by Osada et al. so that they would be able to handle training sequences of mixed length. This will allow these techniques to be applied to a much wider range of data sets and thus increase their overall usefulness.

2 Mixed Length Algorithm

For the purposes of developing and testing a mixed length transcription factor binding site algorithm, I chose to focus on only one of the algorithms evaluated by Osada et al. However, since these methods differ only in their scoring schemes, a facet that I left mostly unchanged, the additions that I made to one technique can be transferred to any of them with little difficulty. The specific technique that I chose to work with was the Centroid algorithm. Centroid works by creating a scoring matrix with length $L$ equal to that of the training sequences it is given. To search through a region of DNA for the sequence most likely to be a fellow binding site of its training sequences, Centroid moves through the search data in a sliding window, scoring each sequence of length $L$ and tracking the highest scoring sequences. Centroid creates its scoring matrix by calculating the frequencies of each nucleotide in each of the $L$ positions of the scoring matrix. To score a test sequence, for each position it adds to the score the frequency of the nucleotide found at that position of the test sequence among the training sequences at that position. This is a very direct measure of similarity, simply giving a test sequence points for each base it shares with a training sequence. In Osada et al.’s analysis, Centroid's gave performance was not significantly lower than any other technique [5], and it was more efficient and relatively easy to implement. The mixed length training sequence version of this algorithm is called MLCentroid.

MLCentroid first uses multiple sequence alignment to match the training sequences up as much as possible, though of course it cannot do this perfectly due to the differing lengths. The idea behind this is that despite their differing lengths, there should be certain patterns and regions which are shared by all of the training sequences. By matching these regions up with each other, we can create a model which captures the essential features of that transcription factor's binding sites. Once the training sequences have been aligned, MLCentroid creates a scoring matrix similar to the one that Centroid creates. The important difference between the two is that each position in MLCentroid's scoring matrix is weighted by how many of the training sequences are actually present at that position in the alignment that was chosen. This gives more weight to the core region of the binding sites which is shared among the training sequences and less to the more diverse extremes of the longer sequences. If the regions which are conserved among the binding sites have been aligned with each other as they should be, there will be more overlap among the training sequences at those regions. This will cause these characteristic regions to be given the most weight when
scoring a possible binding site. Once that is complete, MLCentroid goes on to search through the genetic data it has been given with a sliding window, applying the scoring matrix to each possible site. The length of this sliding window is the difference between the right and left extremes of alignment of the training sequences, so that every base of the training sequences is considered even though the weighting of the positions will likely keep those positions from having much effect on the overall results. The \( n \) highest scoring sites are recorded and output, where \( n \) is the number of results desired by the user.

The alignment strategy used by MLCentroid underwent major changes during development. The first version of MLCentroid simply tried every possible alignment of the training sequences and kept the highest scoring one, where the score of an alignment was the sum of all the pairwise alignment scores. A pairwise alignment score was calculated by simply counting the number of positions at which the two sequences had the same nucleotide. This is a much simpler alignment scoring system than is often used, because it does not make any distinction between different types of mismatches. The use of a more complicated scoring system could improve the algorithm and is discussed in the future work section. This the running time alignment strategy was exponential on the number of training sequences and thus took a prohibitively long time for data sets with many training sequences. The memoization of pairwise alignment scores greatly improved the speed, but was not able to bring it within an acceptable bound. In order to improve the speed of the alignment, MLCentroid no longer tests every possible alignment of the training sequences. Instead, it first sorts the training sequences by length. Then, it aligns them one at a time, with the alignment of each sequence determined by its pairwise alignment scores with all of the previously aligned sequences. The sequences are aligned from shortest to longest, under the rationale that the best alignments for shorter sequences will often be easier to find than for longer sequences. Locking in the position of sequences as they are considered drastically reduces the running time of the program, making it quadratic on the number of training sequences instead of exponential. This version of MLCentroid ran in only a couple of seconds for all of the data sets it was given. Fortunately, a comparison of the alignments generated this way with the ones generated through a consideration of every possible alignment show little difference in the cases where the exponential version ran quickly enough to provide results for comparison.

There remain some cases in which the alignment used in MLCentroid is far from optimal. The best example of this is a data set where half of the sequences have a block of four Ts in the middle and half have a block of four As, such as one used in MLCentroid's testing. The sequences are all very close in length, so it seems intuitively clear that the four Ts are meant to align with the four As. However, using the scoring described above, this alignment would never be chosen because the algorithm sees no connection between the stretch of Ts and the stretch of As, and in fact scores it very poorly due to the complete lack of matches there. In order to align such sequences as a human observer might expect, the alignment strategy would need to somehow consider the structure of the sequences as well as the actual nucleotides. This is also addressed in the future work section.
3 Method of Testing

In their evaluation of transcription factor binding site search algorithms, Osada et al. used leave one out testing [5]. For each set of training sequences, one would be left out. The algorithms would then be given the remaining training sequences to build their scoring matrices, and used to score the binding site which had been left out. Then, the same scoring matrix would be used to score all of the sequences from the other sets of training data. Any of these sequences, which are not binding sites for the transcription factor in question, would be a false positive if encountered in an actual application of the binding site detection algorithm and given a higher score than the actual binding site that was left out. The number of false positives that found is used to assess the performance of the algorithm.

MLCentroid was assessed using the same strategy, but with the Drosophila Melanogaster and yeast data sets used by Tompa. These data sets all contain known binding sites of varying length, so they provide a good test of MLCentroid's intended functionality. Additionally, working with real transcription factor binding site data instead of generated test sets makes us more confident that these results can be applied to actual research situations.

The varying lengths of the sequences posed some difficulties in applying this testing strategy. In some cases, the scoring matrix generated by the aligned training sequences would be shorter than the sequence it was scoring, and in some cases longer. In handling these issues, an effort was made to reflect the way MLCentroid would behave in an actual application. In the case where the sequence to be scored was longer than the scoring matrix, every segment of the test sequence was scored and the best one taken as that sequence's score. In practice, the sliding window search through the genetic data would consider every one of these sequences and take the best one, so this is an accurate reflection of a real situation. The opposite case, in which the scoring matrix is longer than the test sequence, was more difficult to handle. In practice, a short sequence would be surrounded by other nucleotide which would add somewhat to the score of that sequence even if there wasn't a close relationship to it. Since a short test sequence could only benefit from a reduced portion of the scoring matrix, such sequences were at an unfair disadvantage. Even considering every alignment of the scoring matrix onto the test sequence and taking the best would not correct this imbalance. To account for this, in addition to using the best possible of its alignments, a short sequence received the expected value of what the unused portions of the scoring matrix would add to its score if the nucleotides surrounding the test sequence were random. More precisely, for each unused position in the scoring matrix, one fourth of the weight of that position was added to the score. This is not a perfect solution because it relies on assumption that the surrounding bases would be random, but it is sufficient to put short sequences on an equal footing with longer ones.

The tables on the next page contain the experimental results with the Drosophila Melanogaster and yeast data sets. The row indicates which data set was used as the training set, with all other sets being used as potential false positives. The column indicates which sequence from the training set was being left out for that particular run. The numbers in the table are the number of false positives found in the run.
# Drosophila Melanogaster

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
<td>9</td>
<td>0</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>31</td>
<td>11</td>
<td>6</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>41</td>
<td>18</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>41</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>27</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>35</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>44</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# Yeast

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>20</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>46</td>
<td>16</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4 Analysis of Results

The most notable thing about these results is the extreme nature of most of the scores. In almost every case, the true binding site was either one of the very best sequences or hopelessly far down the list. The cause of this is not clear. It cannot be a matter of difficult data sets versus easy ones, as the widely varying values often occur in the same data set. The best example of this is the results for data set 6 of Drosophila Melanogaster, which had six 0s, the best score possible, and 44, the worst score in any data set. It seems that there are particular sequences throughout the data sets that MLCentroid has great difficulty relating to the other sequences from that set. Since these are known transcription factor binding sites, however, it seems unlikely that there are sequences that actually don't fit with their fellow binding sites. If that were the case, it would violate the fundamental idea that all of a transcription factor's binding sites are similar. Rather, it seems more likely that this is the result of some quirk in MLCentroid's alignment process. This deviation may be useful in practice since a researcher would know that if the first few sequences are invalid then it probably isn't worth investigating lower ranking sequences.

Since MLCentroid is dependent upon its training data to distinguish binding sites from other sequences, we would expect it to fare poorly for small data sets. There is evidence of this in the results, notably data set 5 from yeast and data set 2 from Drosophila Melanogaster. However, the excellent accuracy with data set 2 of yeast shows us that the disadvantage of a small training set can be overcome if the sequences are sufficiently well aligned and similar to one another.

In about 62% of the cases, the desired sequence is one of the top three results, and it is fairly high in the list for several more. Overall, this performance is similar to that achieved by the same length version of Centroid in Osada's assessment.

5 Conclusion

Although handling training sequences of varying length presents several significant problems to a supervised learning transcription factor binding site detection algorithm, it is still possible to achieve reasonable results by making appropriate modifications to the algorithms. Furthermore, it is important to do so given the number of data sets which contain binding sites of differing length.

In general, MLCentroid succeeds in building a sensible scoring matrix despite the irregularity of its training data. There exist cases in which it does very poorly, but it usually provides very good results. Additionally, the algorithm is very time efficient, running in only a few seconds even on unusually large data sets. By extending the use of supervised learning transcription factor binding site algorithms to a wider range of data sets, the strategies developed here and implemented in MLCentroid significantly increase the value of these algorithms.
6 Future Work

There are a variety of ways in which this research could be further developed. Most prominently, the scoring schemes currently being used for multiple sequence alignment are extremely basic. By simply giving one point for nucleotide matches and zero points for mismatches, the current method completely ignores difference in nucleotide mismatches. It would be more sensible to vary the scores of mismatches so that ones which represent more significant chemical changes are given greater penalties than relatively minor differences. A more nuanced scoring of mismatches than the one currently in use would likely provide more sensible alignments of the training sequences and thus better overall results.

MLCentroid also does not make use of the information content of positions or pairwise dependencies in its scoring matrix. Information content refers to how useful each position in the scoring matrix is in differentiating between binding sites and other sequences. By giving greater weight to highly indicative positions focuses the algorithm on the important features of the training sequences. Information content is higher in positions which are almost always a particular nucleotide and lower in positions with a more even distribution. Pairwise dependencies deals with some pairs of positions being more important than other pairs, as opposed to the normal case of some individual positions being more important than other individual positions. Both of these considerations were shown to yield substantial improvement in the same length versions of these algorithms by Osada et al., so incorporating them into MLCentroid may provide a boost in prediction accuracy.

Because the strategies used by MLCentroid can be applied to any of the methods Osada et al. evaluated, mixed length versions of all of them could be created. In addition to the benefit of having mixed length versions of each algorithms available to choose from, it would be interesting to see if some adapt to the mixed length extensions better than others.

Lastly, incorporating the scoring techniques examined here into de novo motif detection algorithms may be able to improve their performance. These de novo algorithms are designed to find regulatory motifs which are completely unknown. Because there are no known binding sites in these cases, the supervised learning techniques discussed in this paper cannot be applied directly. However, the de novo algorithms generate multiple candidate motifs and need some way of ranking them so that a scientist using the program would check the motifs most likely to have true biological significance first. The scoring techniques considered here may be useful in doing this ranking.

Acknowledgments. This research was funded in part by the National Science Foundation as part of the 2009 Bio-Grid REU program at the University of Connecticut. I would like to thank Chun-Hsi Huang and Sanguthevar Rajasekaran for their guidance in performing the research.
7 References


