Applying deep learning to derive insights about non-coding regions of the genome

Avanti Shrikumar, Anna Saplitski, Sofia Luna Frank-Fischer (avanti, annasaps, luna16)
CS 229 Final Project, December 2014

Motivation

Most cells in the human body have essentially the same DNA sequence, but the cells in heart tissue behave very differently from the cells in lung tissue. The difference is epigenetics: modifications of the DNA (or proteins associated with DNA) that control which genes are turned on and which are turned off. One important example of such modifications is the activation of regions of the DNA called enhancers. Enhancers active in a given cell type are known to turn on target genes, but the "genomic code" underlying their behavior is largely unknown. Current methods to identify enhancers rely primarily on chemical assays that detect the modifications associated with enhancer activity. However, the link between the underlying DNA sequences at an enhancer and its cell-type specific activity is poorly understood. Previous work has focused on training SVMs using string kernels to distinguish between enhancers and random genomic segments, but SVMs do not scale as well to terabytes of data and require that the features and kernels be known beforehand. Deep learning, on the other hand, allows us to perform unsupervised learning through pre-training to automatically discover relevant features, and the weights learned for these features might give interesting insights into the genomic code at enhancers.

Data

The positive dataset consists of (non-public) ENCODE enhancers identified by experimental assays. We used the central 150bp of all enhancer sequences that were < 1000bp long, which retained 404119 sequences. The rationale for restricting to those regions < 1000bp was that larger regions might actually consist of two smaller enhancer regions in close proximity, and would thus look qualitatively different. There were two negative sets considered at different points in our study: the first was random permutations of the DNA sequence of the positive set, and the second was random segments of the genome. In all cases, the hg19 genome (human) was used. While more data was available, preliminary experiments (not reported here due to space) showed that for our neural net, doubling the input data size did not lower the error rate appreciably.

Features and Preprocessing

To represent raw sequences, we used one-hot encoding, which represents each base in the sequence with four binary bits. Exactly one bit is set, depending on whether the base is an A, C, G, or T. In addition to using one-hot encoding, we used a spectrum kernel to transform the sequences. The spectrum kernel is built on a mapping which, given a subsequence length k, returns the occurrence counts of each length k subsequence (or "k-mer") in the string. We used the Kebabs kernel package implementation of the spectrum kernel.

Models

We explored three main categories of models. The first was Multinomial Naive Bayes trained on subsequences of length 1-9. The second was a Support Vector Machine that used a spectrum kernel with k-mer lengths 1-5 and cost = 15. In order to implement the Support Vector Machine, we relied on the Kernlabs library, which allowed us to use a custom kernel. The third was a neural network with a 2-class softmax output, trained with Stochastic Gradient Descent (batch size 10) and early stopping (the dataset was split as 0.75 training, 0.125 validation, 0.125 testing). Various architectures were explored for the Neural Network, which we will explain on a case-by-case basis in the subsequent sections.

Results and Discussion

Shuffled Sequences as Background

It was known in the literature that enhancers have a higher incidence of “G” and “C” bases (called “GC content”) in their DNA sequence compared to random regions of the genome. Indeed, running Multinomial Naive Bayes trained merely on the incidence of A/C/G/T bases achieves 67% accuracy. We therefore had the hypothesis that by using shuffled versions of the positive set sequences as the background set, we would keep the proportion of G/C bases constant and would thereby force the neural net to pick out features that went beyond G/C content.

Using the one-hot encoding representation of the raw sequence, we trained a neural net with one hidden layer and sigmoidal activations containing 100, 300, 500 and 1000 nodes, attaining test misclassification rates of 20%, 16%, 15% and 14% respectively with early stopping. While we were encouraged by this result, when we scanned the sequence of chromosome 21 to generate a distribution of a probability of a given region being an enhancer; the distribution we produced for non-enhancer regions looked identical to the distribution generated for known enhancer regions (**figure 1**). Contrary to our expectations, the neural net had identified features common to all genomic regions, rather than features unique to enhancers. Our initial hypothesis, from looking at the features detected by the most informative nodes in the neural net, was that the neural net was identifying wide, global patterns, potentially related to a biological phenomenon known as nucleosome positioning. However, as a sanity check, we trained a Multinomial Naive Bayes model on all subsequences of length 2 in the positive set (using shuffled sequences as the negative set), and this model was able to attain a 3.5% misclassification rate. In particular, the 2-mer CG (which is has biological significance as CpG islands) was 4.8x less likely to occur in enhancer sequences relative to shuffled versions of the sequence. It therefore seems likely that most of the discriminative power of the informative nodes in our neural net was due to detecting this bias in the occurrence of CG dimer, but more investigation would be necessary to determine if this is true. A visualization of the most informative node in the 100-node model is included (high activation of this node corresponded to classifying the region as not an enhancer) (**figure 2**).

Neural Net with Random Genomic Sequences

Once we realized that training against shuffled enhancer regions did not pick out the features we were interested in, we switched back to the negative set of random genomic sequences. We hoped that the nets would be able to pick out the features that distinguish enhancer and non-enhancer regions. We tried the following architectures: 1 hidden layer with sigmoidal activation functions and 100/300 nodes, 2 hidden layers with sigmoidal activations and 100 nodes each, and a convolutional layer with rectified linear activation functions and 64 channels with kernels spanning consecutive 7b segments followed by a fully connected layer with 100 hidden nodes. Unfortunately, all of these architectures produced errors in the 31-32% range which barely outperformed Naive Bayes trained on GC content. We examined which features the individual nodes were picking out, and we discovered that the highly-weighted nodes were again mostly picking out GC content.

Next, we tried pre-training with denoising autoencoders to see if that would get the net to pick out more informative features. We pretrained using and encoding layer with 500 hidden nodes, used sigmoidal activations for the encoder and decoder, used the mean binary cross-entropy for the reconstruction error, and set the noise-level to 2. Although pretraining did appear to identify interesting patterns in AT richness (**figure 3**), adding a pretrained layer followed by a hidden layer with 100 nodes and sigmoidal activations didn’t improve the performance beyond the 31%-32% range. We were still intrigued by the nodes identified in figure 3 and hypothesized that they were associated with a biological phenomenon called nucleosome positioning; if so, they may have been useful in detecting an ‘anchor’ point within enhancer sequences that

could have been used to re-center the sequences, which would potentially reduce the need for developing models than can tolerate translational invariance. To test out this hypothesis, we isolated the portion of the weight vectors of informative nodes that appeared to be involved in nucleosome positioning to make an ad-hoc “nucleosome positioning detector” (example in **figure 4**) and used it to scan the enhancer regions; if our detector was indeed identifying nucleosome positioning, we would expect only one strong peak per enhancer regions, as nucleosome positioning does not occur more than once per 150bp. Unfortunately, each enhancer region had, on average, multiple peaks for our “nucleosome detector”, countering our hypothesis so we did not pursue this idea further (data not shown due to the page limit).

Benchmarking with Support Vector Machine and Multinomial Naive Bayes

In order to benchmark our neural net, we ran both Multinomial Naive Bayes (using k-mers as features) and a Support Vector Machine (with a spectrum kernel). For Naive Bayes, the training set involved 974k sequences (balanced between enhancers and random genomic regions) and looked k-mer lengths 1 through 9. The best performance was in k-mer lengths 6 through 9, which had misclassification rates of about 27%. With k-mer length 5, Naive Bayes had an error rate of 29%, and with k-mer length 4 it had an error rate of 31%.

We were not able to test the SVM on the same data set size as the Naive Bayes classifier because the runtime of SVM’s scales quadratically in the training set size. Our particular implementation, which used R’s kebabs package, scaled particularly badly, so our training set size was limited. In addition, using the spectrum kernel with high k-mer lengths was expensive, so we only used up to a k-mer length of 5. In order to generate a learning curve, we ran the SVM with training sizes of 3k, 5k, 10k, 22k, and 43k sequences. The testing error was about 28% with 4-mers and 27% for 5-mers for both of the biggest data sets. The learning curves for k-mer lengths 4 and 5 are included (**figures 5 and 6**). We note that we may have been underfitting our data, given that the testing and training errors converged completely for k-mer length 4. Although the training set was much smaller, the SVM still outperformed Naive Bayes (both trained on k-mers and trained on GC content) and the Neural Net (with one-hot encoded raw sequences as input).

Neural Net with Spectrum Kernel Transformation

Given our success with the spectrum kernel SVM, we decided to create a neural net whose input was the result of mapping the raw sequence to higher dimensional space used by the spectrum kernel. That is, we gave the neural net the counts of k-mers of the raw sequence instead of the raw sequence. The neural net architecture again involved one hidden layer with 100 nodes and sigmoidal activations. The neural net performed better than the SVM, getting a testing error of 25%. It should be noted that the neural net was trained on a larger dataset; we could not do the equivalent comparison of how the SVM would perform on this dataset because the SVM would not scale to that level. However, the learning curve in figure 6 of the SVM suggests that it would not perform substantially better on the larger dataset. It should also be noted that the neural net was trained on the same small dataset of the SVM with 4-mers, the testing error was 0.4% worse that that attained by the SVM, but it is also true that neural nets are generally most effective when trained on larger datasets.

We didn’t investigate which k-mers were most helpful in distinguishing enhancer and non-enhancer regions because this information is not biologically interesting. Instead, this was mostly an exercise to demonstrate that if we could induce our neural net to detect local features such as k-mers, we would likely attain better performance. The reason we wished to derive the k-mer representation directly from raw sequence, rather than use k-mer counts, is that the k-mer count representation discards a lot of positional information and inherently prevents the detection of a lot of higher order positional patterns that we suspect exist at enhancers. Convolutional layers offer one way of detecting local features, but our previous attempts at training convolutional layers did not produce fruitful results out-of-the-box. We thus decided to initialize the kernels of the convolutional layer in a way that would pick out the most informative 4-mers identified

according to Naive Bayes, in the hope that this may cause the stochastic gradient descent to find better local minima (we chose 4-mers rather than 5-mers to make the number of channels in the convolutional layer more tractable). So far we have not been able to obtain the performance with the k-mer-count representation, but we are actively trying to debug why.

Conclusion

Although our first attempts with the Neural Net were not particularly successful, we were able to improve its performance using understanding gleaned from running a Naive Bayes classifier and an SVM. Once we began using the k-mer transformation as preprocessing, the Neural Net began to perform better than either the Naive Bayes or the SVM, especially given that it can handle much larger training sets than an SVM. However, there is still a significant drawback in using the k-mer transformation with a Neural Net as opposed to an SVM, because it requires doing the transformation explicitly instead using a kernel transformation.

Future Directions

The immediate order of business would be to find out how to derive the equivalent information in the k-mer-count representation by using the one-hot representation with convolutional layers. The three hypotheses for why this is not yet working are: (1) that the number of channels we are initializing the convolutional layer with (we have tried various numbers) is not enough to capture all the relevant information, (2) that the initialization we are using for the weights is not actually effective at picking out k-mers in a way that is useful to subsequent layers and (3) parameter explosion from using a large number of channels is creating lots of local minima and our stochastic gradient descent algorithm is getting trapped. Our preliminary exploration of (1) suggests that it is not the case, because when we limit the k-mer-count representation to the number of "top" k-mers identified by Naive Bayes, our network has better performance. Our preliminary exploration of (2) is promising, but the intricacies of our exploration do not fit in this report. We have not done any exploration of (3) yet, but we would plan to investigate it by adding pooling layers after the convolutional layer, to hopefully reduce the number of parameters.

Another idea would be to "fix" the issue with our initial negative set (which was shuffled versions of the positive sequence) by keeping both the proportion of G's and C's as well as the incidence of the CG dimer constant between the positive and the negative set. This may result in a negative set that has the same GC content but is also not so easily distinguished by Naive Bayes.

References

A. Kartzoglou, A. Smola, K. Hornik, A. Zeileis (2004). kernlab – An S4 Package for Kernel Methods. R. Journal of Statistical Software 11(9), 1-20. <http://www.jstatsoft.org/v11/i09/>

I. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, Y. Bengio. Pylearn2: a machine learning research library. arXiv preprint arXiv:1308.4214. <http://deeplearning.net/software/pylearn2/>

J. Palme, U. Bodenhofer (2014). An R Package for Kernel-Based Analysis of Biological Sequences. R package version 1.0.2. <http://www.bioinf.jku.at/software/kebabs/>

M. Handi, D. Lee, M. Mohammed-Noori, M. Beer (2014). Enhanced Regulatory Sequence Prediction Using Gapped k-mer Features. PLoS Computational Biology 10(7): e1003711. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4102394/>

Explanation of Weight Profiles at Informative Nodes

We identify informative nodes by looking at which nodes had the largest outgoing weights to the 2-class softmax layer. For these nodes, we normalize the incoming weight vector to have magnitude 1. Since we have a one-hot encoding, every 4 elements in the weight vector correspond to the four possible bases (A/C/G/T) at a given point in the raw sequence. We thus plot the weights corresponding to different bases in different colors, and stack them so that the smallest weight is in the front.

Figures

Figure 1

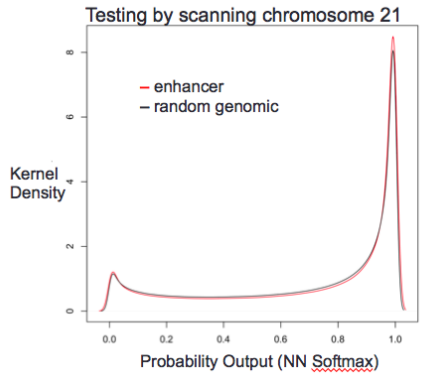


Figure 2

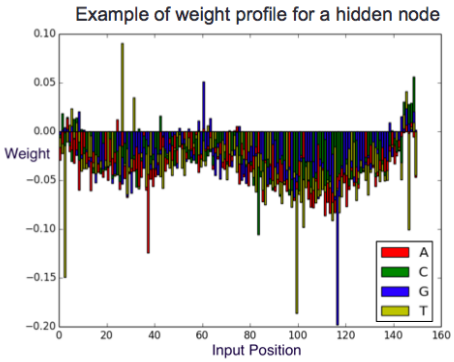
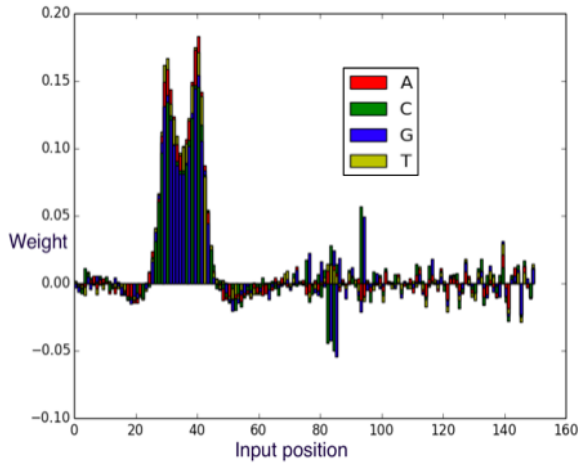
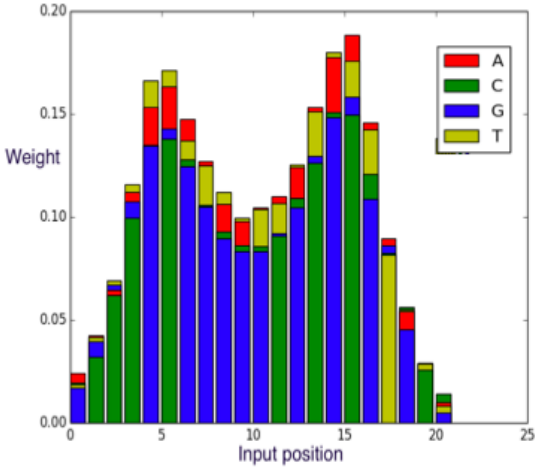


Figure 3



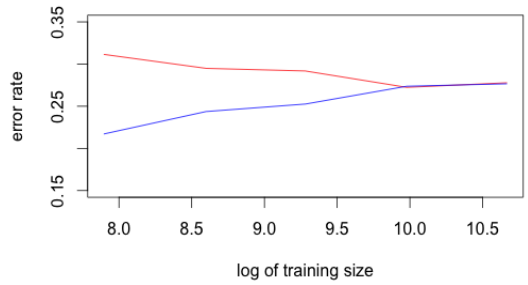
A visualization of the weights placed on various base pairs along a sequence. An explanation of this visualization is at the end of the text. Note the the high weighs given to regions with AT-rich flanks.

Figure 4



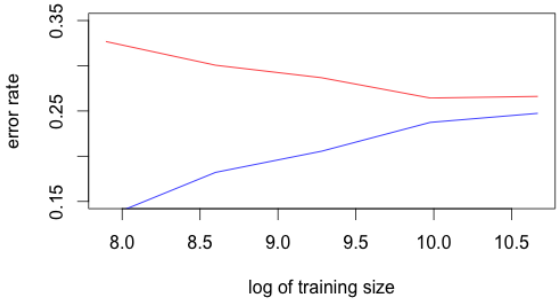
Weight profile for node used as nucleosome positioning detector. An explanation of this visualization is at the end of the text.

Figure 5



Training (blue) vs. testing (red) error rates for the SVM with spectrum kernel, k-mer length 4.

Figure 6



Training (blue) vs. testing (red) error rates for the SVM with spectrum kernel, k-mer length 5.