CS230

Predicting Gene Expression Using Epigenetic Markers on the Genome

Sunil Bodapati
Department of Bioengineering
Stanford University
bodapati@stanford.edu

Timothy Daley
Department of Statistics
Stanford University
tdaley@stanford.edu

Sonja Johnson-Yu
Department of Computer Science
Stanford University
sonjyu@stanford.edu

Abstract

Gene regulation is a complex process by which transcription factors and epigenetic factors modulate gene expression to control protein concentrations. Previous works have examined this regulation either indirectly or by using simple linear models. We propose a deep neural networks approach to predicting gene expression to account for combinatorial regulation and epigenetic interactions. We used a four layer fully connected neural network to predict normalized gene expression from normalized chromatin openness. We found that the though the training error can be made to be small, the generalization error is high, even with various regularization techniques. This indicates that we are missing pertinent information in our regression problem.

1 Introduction

The human genome contains sequences that directly code for proteins (coding regions), as well as sequences that regulate protein expression levels (non-coding regions). Various molecules can bind to these non-coding regions, blocking or facilitating transcriptional access in a process known as epigenetic regulation. Epigenetic regulation plays a critical role in healthy cells (X inactivation [7]), as well as unhealthy cells (cancer [10], a host of developmental disorders [6]), making it an important process to understand. Our project aims to illuminate the role of non-coding epigenetic patterns on gene expression. Specifically, we are interested in how epigenetic modifications on genetic sequences near or in a gene affect the corresponding expression of the gene. Our goal is to obtain more accurate model for predicting gene expression from epigenetic markers.

2 Related work

A directly related work [3] showed that most gene expression can be predicted using a linear L1 penalized regression model. Though, the key to the model is the construction of the features based on prior knowledge of gene regulatory networks, cell type specific transcription factors, and the presence of corresponding transcription factor binding motifs in regulatory regions. They were able to acheive an $R^2 \approx 0.8$ on on a chosen subset of $\sim$5,000 genes in a held out cell type. Specifically, the authors created 5000 separate linear regression models for their target genes of interest. This model ignores combinatorial effects of gene regulation, and therefore we hypothesize that a deep neural network may better predict gene expression. We also hypothesized that epigenetic machinery in a cell is agnostic to genes, allowing us to create a single model that will be able to generalize across all genes.

Recently, there has been an explosion of applications of deep neural networks in genomics, though few directly related to our goal. Most commonly is the use of convolution networks and recurrent networks to predict epigenetic information [5, 8, 12] based only on DNA sequence. [2] used deep neural networks in a multi-task regression problem, but did not use epigenetic features in their regression, using only the gene expression across a wide range of different cell type to predict a target gene expression. Our hope is that the epigenetic information is critical to prediction, something previous work has for the most part ignored. [11] used CNNs to predict whether target gene expression is above or below the median based on the measurement of five epigenetic marks within 5,000 base pairs up- and down-stream of the TSS. There are a few problems with their approach. Their choice to convert a regression problem into a classification problem is meaningless, as there is no inherent meaning to whether gene expression is above or below the median, but large deviations from the median are biologically meaningful (e.g. differential expression [9]); the five assays they use are complicated and expensive to perform on new cell types; and they ignore possible distal regulation.

CS230: Deep Learning, Winter 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

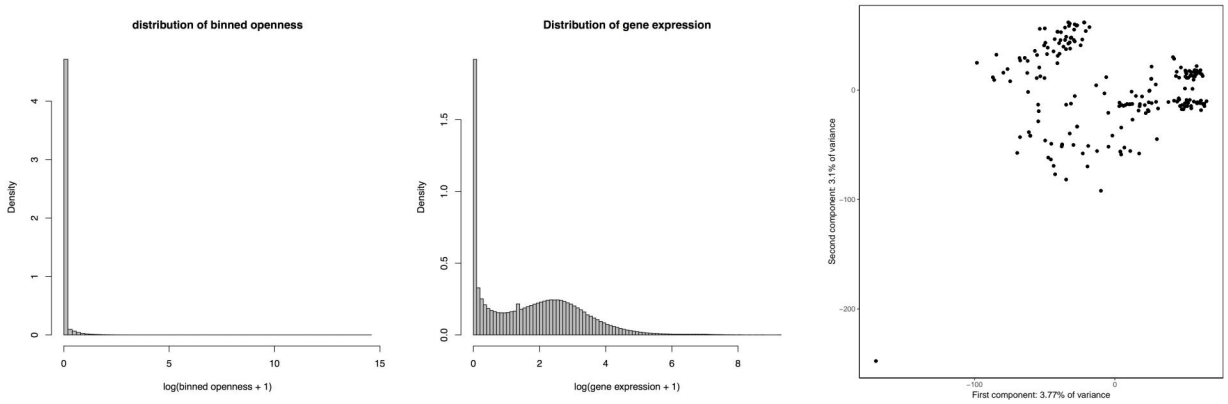

Figure 1: Distribution of openness (left) and gene expression (middle). The far right shows the PCA projection of biological samples based on gene expression data.

3 Dataset and Features

Our primary dataset contains information obtained by using ATAC-seq on 201 unique cell types. ATAC-seq works by using a mutated hyperactive transposase that will randomly insert a specific sequence into the genome at extremely high frequency. Any region of the DNA that is bound by chromatin will block the transposase activity in that region, while regions that are open or have transcription factors binding will be open to the transposase. Thus, when the transposase markers are pulled down and the cell DNA sequenced, regions of the genome that have extremely high reads counts of the transposase sequence are comparatively open versus regions of the genome that have very few transposase reads. Open regions can be thought of as a proxy for a host of possible epigenetic activity [1]. Our dataset contains genome-wide openness data for 201 experiments of various cell types, summarized into bins of high openness and paired with (meaning from the same cell type and produced by the same lab) the gene expression data of 17,794 genes. Thus our openness dataset is of shape of approximately 200k genetic regions x 201 cell types, and our target gene expression dataset is of 17,794 genes x 201 cell types.

4 Data Processing

The epigenetic data is in the form of openness signal in peaks, where peaks are identified by overabundance from a background negative binomial model [4]. To compare these signals across all genes fairly we binned the peaks as a function of distance from each gene's transcription start site (TSS). Specifically, we created one thousand bins for the 1 million base pairs upstream the gene's TSS (in the same direction the gene is transcribed) and one thousand for the 1 million base pairs downstream (in the opposite direction that the gene is transcribed) and computed the signal of all peaks in these bins, divided by the length of the peak. Our output signal is gene expression for 17,794 genes in 201 cell types. Therefore the full X tensor is 17,794 (genes) x 201 (cell types) x 2000 (bins) and our full Y tensor is 17,794 (genes) x 201 (cell types).

The distribution of the gene expression is approximately normal, with an overabundance of zeros (see figure below). In the openness data, the vast majority of entries are zero (see figure below). This is expected as most of the genome is not involved in gene regulation. Due to the extremely large range of signals, we compressed this down on the log scale as well. As pre-preprocessing for both, we normalized each gene and openness by subtracting off the mean and dividing by the standard deviation, computed across all 201 cell lines for each gene.

We used Principal Components Analysis to plot the gene expression data of the 201 cell lines along the top 2 axes to see if there were any obvious batching effects or outliers. One sample seems to stand out as an outlier, sample 91. This is leg muscle from fetal tissue, though there are 3 other samples of fetal leg muscle. This means that the reasons that it is an outlier are unlikely to be biological. We will need to keep an eye out on this sample in the future.

5 Methods

Train/Dev/Test Set Generation: Our full X tensor was 17,794 (genes) x 201 (cell lines) x 2000 (bins), a data structure that took $\sim$170GB of memory in numpy binary format. To allow for easier handling for training, we reformatted this structure into a 3576594 (genes) x 2000 array, randomized this array by its indices, and created our train/dev/test sets via 90/5/5 splits. Our train set was further split into 100 different files to allow for easier traversal of the X tensor during training. We repeated this process for our Y tensor, ensuring that the X to Y relationship was maintained during randomization.

Training Parameters: All neural networks were trained using the Adam optimization algorithm for backpropagation. Our initial tests with a set of 8,000 data points indicated that a learning rate of 0.001, B1 = 0.9, and B2 = 0.999 performed well in optimizing the neural network. We chose a batch size for training of 10,000 samples.

Creating a Baseline: We first implemented an L1 Lasso Regression comparable to the regression used by Duren et al. to serve as a baseline. We created a 1 layer fully connected network with an L1 regularized MSE cost function with regularization parameters of several orders of magnitude, starting with 10^2 and ending with 10^{-6}, and trained until convergence with an epsilon of 10^{-5} or 150 epochs, whichever came first.

Fully Connected Model: We implemented a 4 layer fully connected network, with each layer containing 1000 nodes and the last layer being the regression output $[1000, 1000, 1000, 1]$. We trained until convergence with an epsilon of 10^{-5} or 150 epochs, whichever came first.

Regularization Methods: Our regularization efforts followed four parallel paths: L1 regularized loss, L2 regularized loss, dropout layers, and a different architecture. We stored the train and dev MSE at each epoch for each model that we trained.

L1/L2 Regularized Loss: We trained our fully connected 4 layer network using an L1/L2 penalized MSE cost function until convergence with an epsilon of 10^{-5} or 150 epochs, whichever came first. We tested with the same range of lambdas as the Lasso model.

Dropout Layers: We trained our fully connected 4 layer network using an MSE cost function and a dropout layer at each layer. We trained until convergence with an epsilon of 10^{-5} or 150 epochs, whichever came first. We used the same dropout probability across all layers per model, and varied the probability from 20% to 80% in increments of 5%.

Different Architecture: We trained on a network with a vastly more limited architecture, with 1000 nodes in the first layer, 500 nodes in the second layer, 250 nodes in the third layer, and the one node for the regression output layer [1000,500,250,1]. This reduced model contained ∼2.6M parameters vs the ∼4M parameters of the previous fully connected models described above. We used an unregularized MSE cost function and trained until convergence with an epsilon of 10^{-5} or 150 epochs, whichever came first.

6 Experiments and Results

Given that we normalized the mean of our Y tensor to be 0 with a variance of 1. Zeroing out the weights would yield an MSE of approximately 1.

6.1 Baseline Results

Our LASSO model is either extremely overfit with small lambdas (<10E-6) or is extremely underfit with larger lambdas (>10E-5). Across all lambdas, the dev error remained high, indicating that the linear model does not generalize well. It is interesting to note that our lowest lambda was able to achieve quite a low training MSE, indicating that a simple linear combination of the gene openness bins is able to encode, albeit in an overfit way, the gene expression.

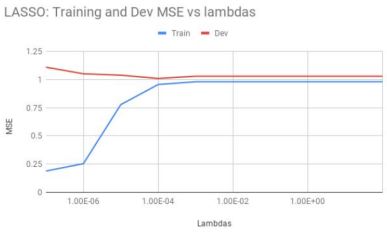

Figure 2: Final training and dev MSE as a function of penalization parameter lambda

6.2 Fully Connected Network Results $[1000, 1000, 1000, 1]$

Our results of the training and dev MSE from a fully connected network indicate a highly overfit model. We can also take away that we need ∼150 epochs of training before MSE appears to stabilize.

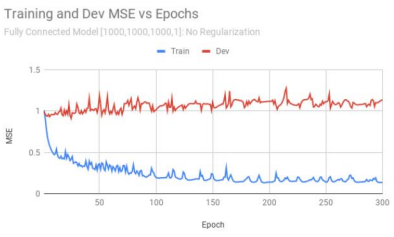

Figure 3: Training and dev MSE for each epoch of training

6.3 L1 Results:

For lambdas greater than 10^{-5}, the weights of the network are suppressed greatly, pushing the model to zero the weights, leading to high MSE. For lambdas less than 10^{-5}, the train MSE looks similar to the unregularized MSE, indicating that there was minimal regularization occuring. Interestingly enough, Dev MSE remained extremely high for all lambdas, indicating that L1 regularization did not help the model generalize to another dataset, even in the case of some partial regularization ($\lambda = 10^{-5}$).

6.4 Dropout:

For all dropout rates, our model overfit the training set. Similar to L1 regularized loss, we were unable to get any kind of regularization or generalization to the dev set.

6.5 Different Architecture [1000, 500, 250, 1]:

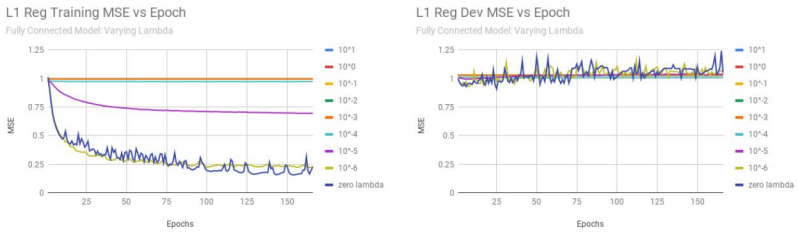

Figure 4: Training (left) and dev (right) MSE as a function of epoch for L1 penalized neural network predictions.

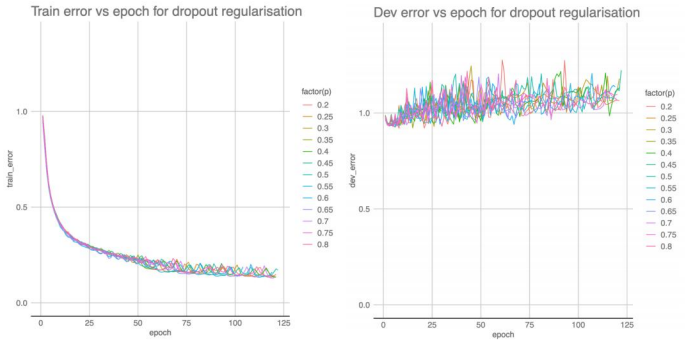

Figure 5: Training (left) and dev (right) MSE as a function of epoch for dropout regularised neural network predictions.

The results from exponentially decreasing the number of neurons per layer show that decreasing the number of parameters had little effect on the generalization problem that we needed to combat.

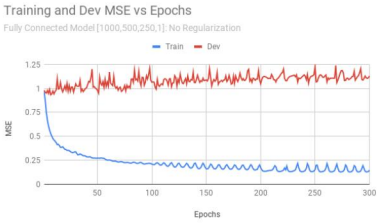

Figure 6: Training and dev error as a function of training epoch for reduced parameter architecture.

7 Discussion and Error Analysis

The LASSO baseline indicated that there was no linear combination of bins that could predict gene expression better than simply zeroing out the weights. Unfortunately, even with this low bar, our regularization efforts were unable to generalize our model to perform better on a dev set. We were able to achieve some semblance of regularization, as witnessed by moderate train MSE, but nothing would improve our dev MSE. Given our inability to generalize our model successfully, our test performance was identical to our poor dev set performance (MSE: ~ 1). We then spent a considerable amount of time analyzing our errors in an effort to determine next steps.

7.1 Error Analysis:

Our first step was to see visually examine how our gene expression predictions fared against the actual gene expression. The best fit line is not a 45 degree line as we would expect, but is skewed closer to predicting zero. This is evident as there are several outliers where our model predicted those highly expressing genes as close to zero. Another informative picture of our error is a scatter plot of the actual gene expression vs MSE. Here, it is evident that our model performed much better for genes that expressed a moderate amount, but was penalized tremendously for highly expressing genes.

To understand the full effect of these outliers, we plotted a histogram of MSE from each example of the test set. We also plotted what the MSE would be as we steadily removed the top ranking samples from the test set. These results confirm that our total MSE is pulled up disproportionately by the presence of outliers, where 30% of samples carry 80% of MSE.

We isolated a large source of the error by visually inspecting the highest MSE points. We saw that the top 10 points in terms of MSE came from a single cell type (the fetal leg tissue we highlighted previously). We plotted true expression vs predicted and highlighted points coming from that sample. We can see that clearly that this sample had a much higher variance than the others.

Finally, given that our model was extremely overfit on the training data, we wanted to see if there were certain bins that were more important to the loss vs other bins. The role of promoters in gene expression makes us believe that the bins near zero

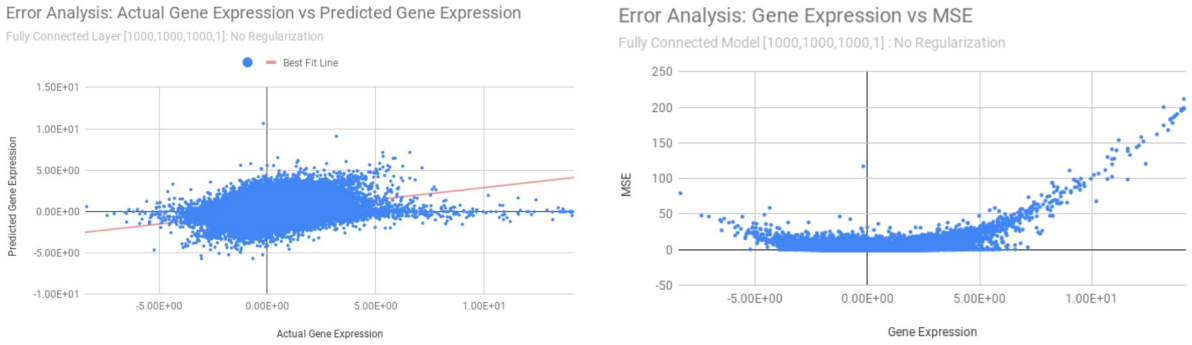

Figure 7: True test set gene expression versus predicted test set gene expression (left) and gene expression versus MSE (right).

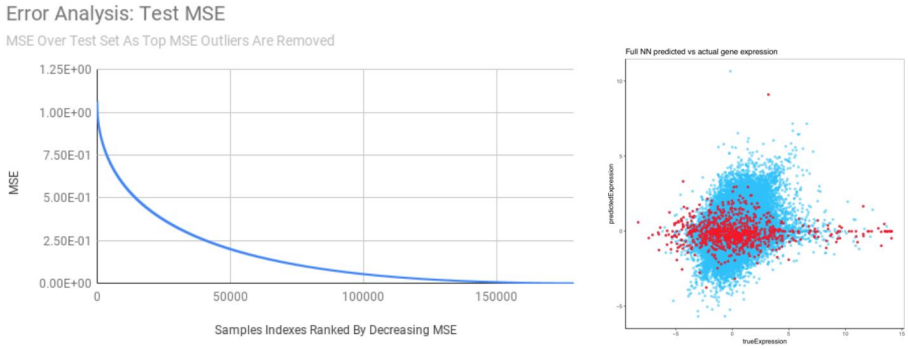

Figure 8: The mean square error decreases exponentially as we remove outliers (left). True test set gene expression versus predicted test set gene expression with observations from sample 91 highlighted in red (right).

should be the most important. To measure bin importance we backpropogated the gradient to the inputs in the full unregularised model over one epoch. Unfortunately, it looked very much like pure noise.

8 Conclusion/Future Work

Our Lasso Baseline resulted in a dev MSE ≈ 1, indicating that it was no better than random guessing. Our unregularized fully connected four layer network was extremely overfit, with a train MSE $\approx .15$ and a dev MSE ≈ 1. Attempts to regularize, using L1 and L2 penalized loss, dropout, and a simpler architecture, were all unsuccessful in generalizing the model to the dev set, and thus also performed similarly poorly on the test set. Subsequent error analysis showed that a few samples carried most of the MSE, with a given cell type performing extremely poorly.

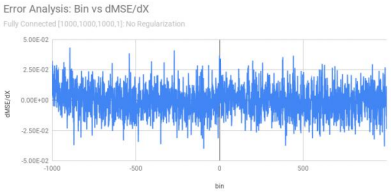

Figure 9: Gradient weights backpropagated to input layer.

Given we saw how cell types play such a large role in MSE distribution, our next step would be to bucket cells that are phenotypically similar and train a model per phenotypic bucket. Hopefully this would reduce cell specific effects that impact generalization.

Additionally, our initial dataset looks at genome openness, but abstracts the actual sequence information. We hypothesize that this assumption is not accurate. The specific sequence that is open or closed is likely important to gene expression due transcription factor binding. Training a model with this sequence information (either an RNN or CNN to take advantage of the sequential and spatial nature of gene sequences) would hopefully provide a more accurate prediction. Ideally, we could then use that model to understand what may be happening on a biological basis.

This project was built in python and pytorch. Scripts available at `https://github.com/sbodapati/Epigenetic_Deep_Learning`.

9 Contributions

SB, TD, and SJY conceived of the project with advice from Zhana Duren and Wing Hung Wong (Department of Statistics, Stanford University). SB, TD, and SJY performed the coding, analysis, and writing.

References

[1] Jason D Buenrostro, Paul G Giresi, Lisa C Zaba, Howard Y Chang, and William J Greenleaf. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nature methods*, 10(12):1213, 2013.

[2] Yifei Chen, Yi Li, Rajiv Narayan, Aravind Subramanian, and Xiaohui Xie. Gene expression inference with deep learning. *Bioinformatics*, 32(12):1832–1839, 2016.

[3] Zhana Duren, Xi Chen, Rui Jiang, Yong Wang, and Wing Hung Wong. Modeling gene regulation from paired expression and chromatin accessibility data. *Proceedings of the National Academy of Sciences*, 114(25):E4914–E4923, 2017.

[4] Jianxing Feng, Tao Liu, Bo Qin, Yong Zhang, and Xiaole Shirley Liu. Identifying chip-seq enrichment using macs. *Nature protocols*, 7(9):1728, 2012.

[5] David R Kelley, Yakir A Reshef, Maxwell Bileschi, David Belanger, Cory Y McLean, and Jasper Snoek. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome research*, 28(5):739–750, 2018.

[6] Eric J Nestler, Catherine J Peña, Marija Kundakovic, Amanda Mitchell, and Schahram Akbarian. Epigenetic basis of mental illness. *The Neuroscientist*, 22(5):447–463, 2016.

[7] Bernhard Payer, Jeannie T Lee, and Satoshi H Namekawa. X-inactivation and x-reactivation: epigenetic hallmarks of mammalian reproduction and pluripotent stem cells. *Human genetics*, 130(2):265–280, 2011.

[8] Daniel Quang and Xiaohui Xie. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic acids research*, 44(11):e107–e107, 2016.

[9] Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2010.

[10] Shikhar Sharma, Theresa K Kelly, and Peter A Jones. Epigenetics in cancer. *Carcinogenesis*, 31(1):27–36, 2010.

[11] Ritambhara Singh, Jack Lanchantin, Gabriel Robins, and Yanjun Qi. DeepChrome: deep-learning for predicting gene expression from histone modifications. *Bioinformatics*, 32(17):i639–i648, 2016.

[12] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931, 2015.