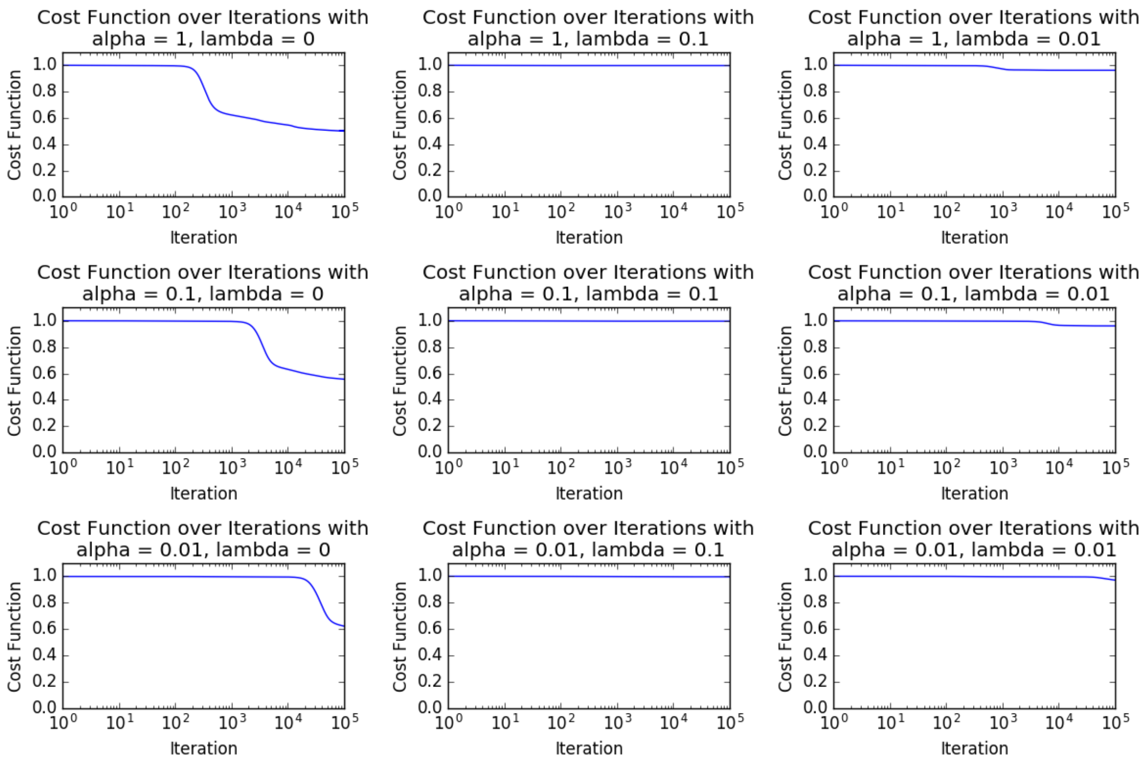
Jason Town
3/22/2017

BMI203 Final Project

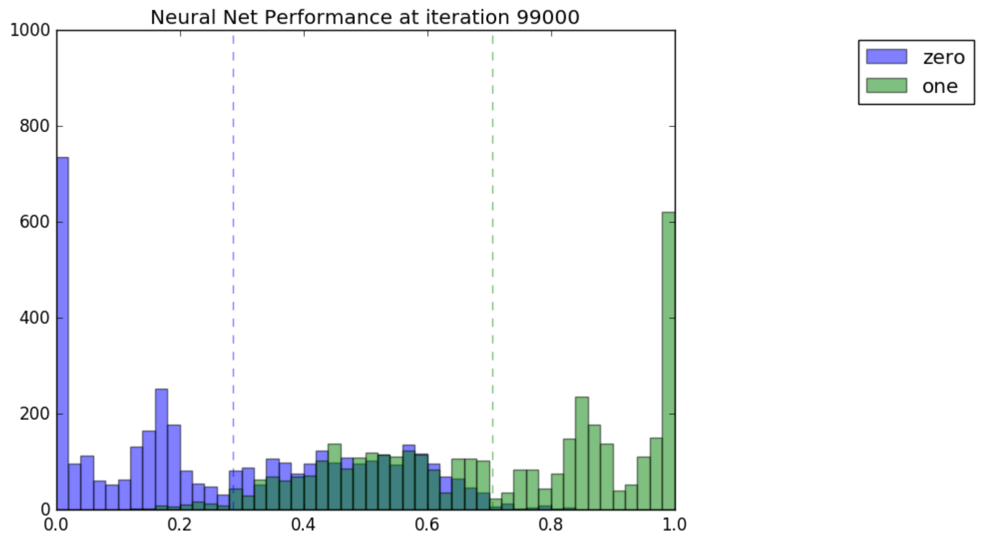
Transcription factor binding is a complex biochemical process that is central to regulation and homeostasis in cells. Sequence logos show general trends in the DNA binding sites, but transcription factors are dynamic chemical structures with many degrees of freedom and these approaches fail to capture the many potential nonlinear relationships in the sequences of these binding sites. Machine learning approaches, including neural networks, can allow us to explore these complex relationships and identify likely binding sites that linear approaches might overlook.

Building a standard neural network

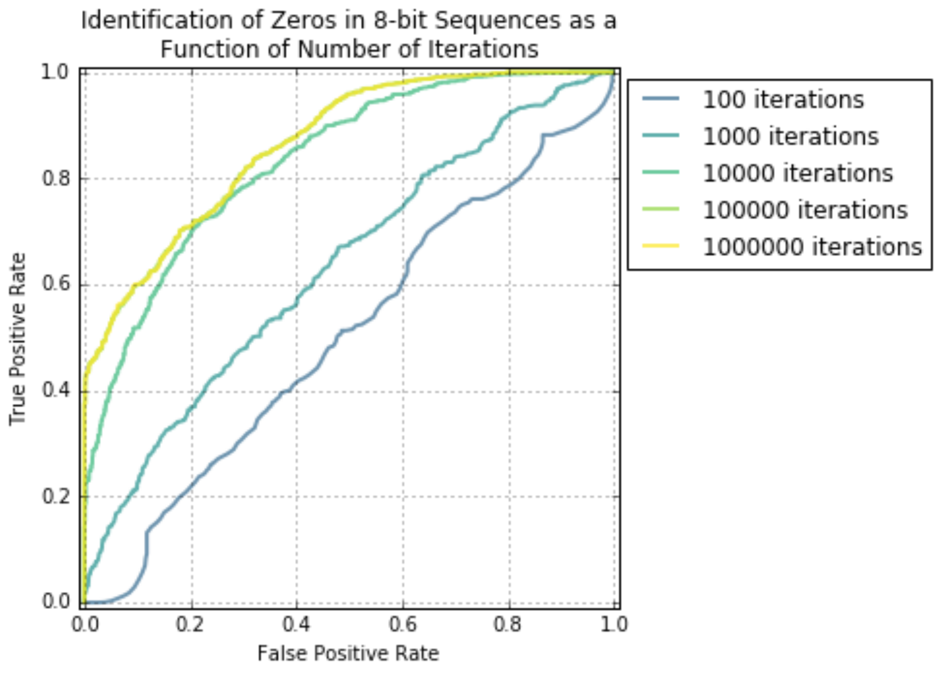
In nn_Q1.py, I wrote a class in python that represents a neural network. I then randomly generated 1000 8-bit numbers and attempted to train an 8-3-8 neural network to autoencode these 8-bit sequences. To try to determine optimal learning rate, α, and weight decay, λ, I tracked the cost function over 100000 iterations for a range of α and λ values. Weight decay appears to dramatically decrease the 'trainability' of my neural network (at least for this particular task), so I set it to zero for this application. I expect overfitting to be a non-issue because there are only $2^8 = 256$ possible values the input can take, so they are likely each represented several times in the training data set.



Unfortunately, even under these optimal parameters, the system did not converge to a perfect prediction rate. Interestingly, after nearly 100000 iterations (at $\alpha = 0.1$ and $\lambda = 0$), some of the ones and zeros were predicted with high confidence, but a large number were also unpredictable. The average predicted value for zero digits (dashed blue line) was much less than then average predicted value for ones (dashed green line), but the predictions are clearly far from perfect (see animated version of this plot here: https://youtu.be/9sKVUcjmEAA).



The output values follow the correct trend, but aren't quite accurate. Thresholding the values allows for generation of ROC curves for various numbers of iterations. The ROC curve does not improve after 100000 iterations, so it is not a matter of simply giving more training time.

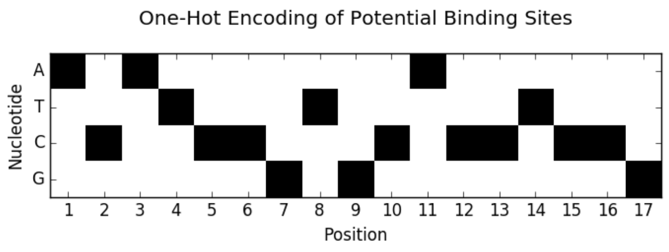


For an autoencoder, we might expect the weights to be symmetric after training. That does indeed appear to be the case when using a larger weight decay term (see animation here: https://youtu.be/TzatBqzbYXQ). However, with a weight decay of zero, the weights of the hidden-to-output matrix appear to be consistently larger than the input-to-hidden weights (see animation here: https://youtu.be/9Nn6dxDb2ZA).

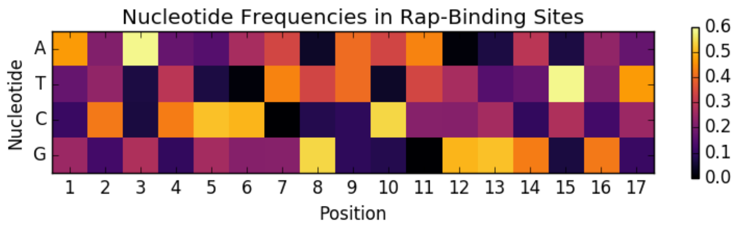
The system is learning *something*, but it doesn't look like it will be very useful for the next steps if it can't solve the autoencoding problem (at least as I understood the problem), so I decided to switch to Keras, a machine learning package for python, for the next steps.

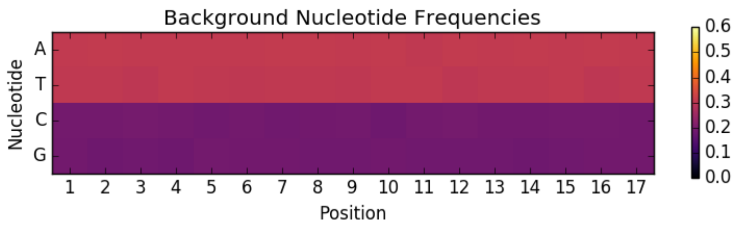
Encoding sequence data for machine learning

Since neural networks can't natively handle categorical data like nucleotide identities, we have to numerically encode the information somehow. One simple approach is substituting each nucleotide with a binary vector representing the identity of that nucleotide. For example, we might encode A as [1,0,0,0], T as [0,1,0,0], C as [0,0,1,0], and G as [0,0,0,1]. Now, we can represent our sequences as matrices.



We should be able to pick out the binding sites if they have a consistent pattern. Count frequencies of each nucleotide at each position does reveal structure as shown below. We could apply a linear approach using these frequencies to try to identify binding sites, but there may be nonlinear relationships that we could miss.

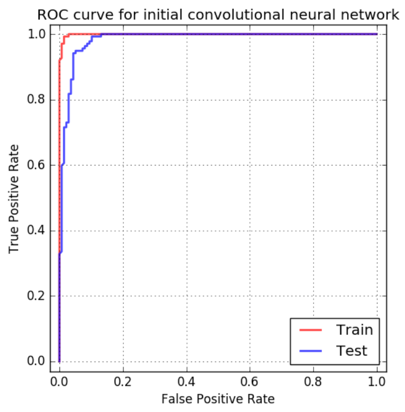




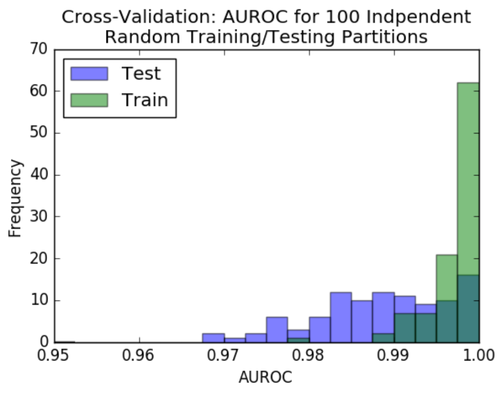
Machine learning approach to binding site identification

Since binding sites have intrinsic 1D structure (and they may actually have 2d structure in vivo as well, but that may be encoded in the 1D sequence and is outside the scope of this project), I decided to attempt using a convolutional neural network to identify possible binding sites. The network I initially set up uses 5 convolutional filters (each having length of 5 nucleotides) connected to a hidden layer of 10 fully connected neurons and finally leading to a single outcome – the predicted probability of the sequence being a Rap1 binding site.

For training and testing, I first collected 17-nucleotide sequences from the negative training set and removed any that were exact matches to sequences in the positive training set. To increase the size of the positive training set, I also used their reverse complement sequences (which is justifiable – both sequences represent the same site in the genome). I then took 500 positive and 500 negative sequences as the entire data set. I trained the network on the first 250 positive sequences and the first 250 negative sequences and then tested the performance on the reserved half. At 250 epochs, the neural network appears to have predicted positive and negative examples in the data set with high accuracy as evidenced by a ROC curve.

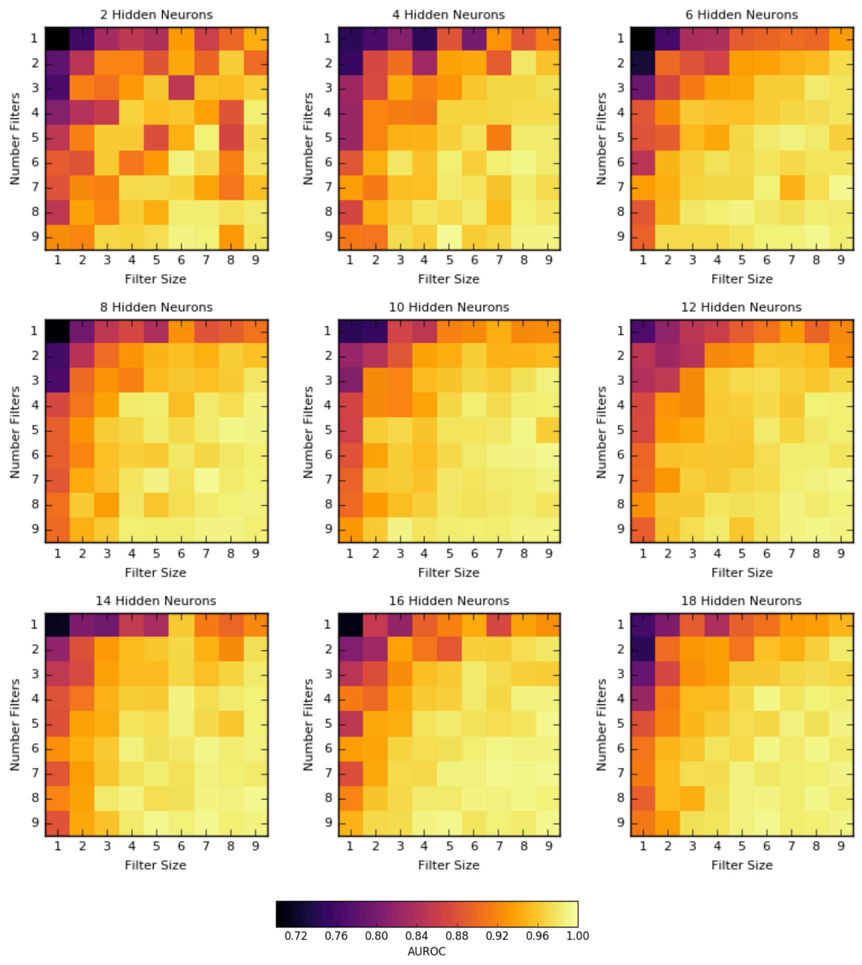


Next, to cross-validate the model, I repeated this experiment 100 times, each time selecting new partitions of the dataset for training and testing. All 100 trials had an area under the ROC curve > 0.96 for the testing set.



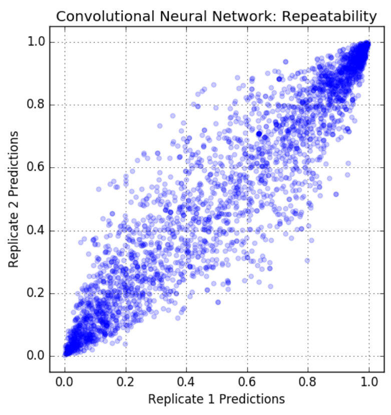
Parameter Optimization

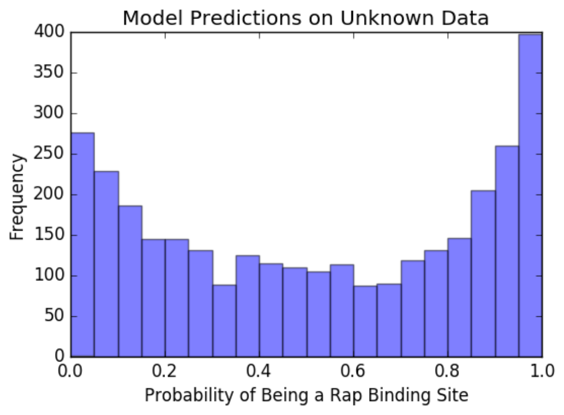
In order to optimize the overall structure of the model, I ran simulations varying the number of filters, the filter length, and the number of hidden neurons. I then calculated the AUROC after 100 epochs of training over 3 iterations and collected the average.



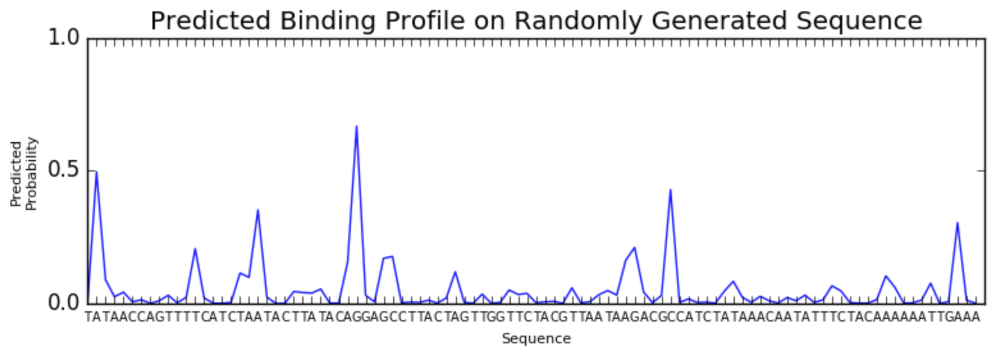
Prediction of unknown binding sites

Applying the trained model to a set of unknown 17-nucleotides sequences generates a set of probabilities that each sequence is a rap1 binding site. Assuming that the training sequences are representative of this dataset, we should be able to identify positive and negative binding sites in this dataset of unknowns. We would the model to generate a bimodal distribution with peaks at one and zero. Furthermore, we would expect high correlation between repetitions. Some sequences are predicted quite reliably, while others are more ambiguous. Nevertheless, the run-to-run Pearson correlation hovers around 0.9 for repeated experiments (0.944, 0.935, 0.881).

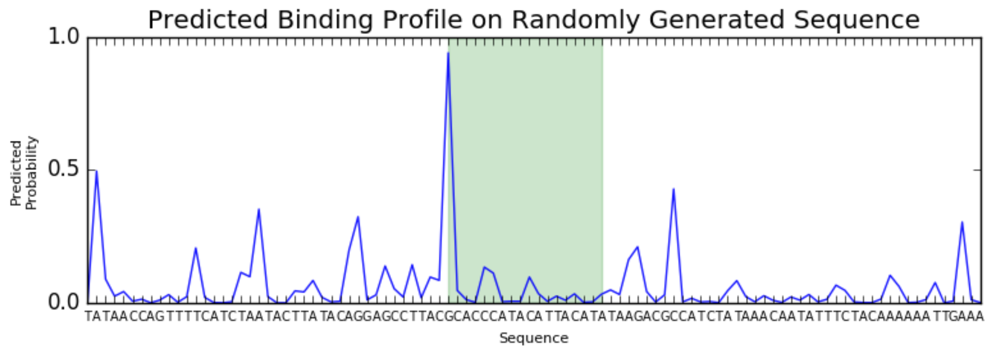




In addition to applying the optimized model to the test set, I wanted to see how the model might perform when applied as a convolution over a sequence. I first simulated a random sequence with nucleotide frequencies that roughly match those found in the yeast genome. In general, there is relatively low probability of binding across this sequence (each probability, p(x), is for the 17 nucleotide sequence starting at x).



When a Rap1 binding site (highlighted in green) is substituted into this random sequence, we see a peak at the left end of the region (it also impacts the shape of the curve preceding the insertion).



Taken across the genome, we might try to apply this convolved function and smooth the results to simulate chip-seq data. An alternative approach to identifying binding site motifs might be training a network to regress Chip-seq signal from the nucleotide sequence (though it would probably require some careful selection of positive and negative training regions if the signal is sparse).