Hacking the genome

Namrata Anand

May 8, 2014

1 Introduction

If you want to hack the genome, you have to find a way to turn genes on and off at will. You have to be careful, though, since all cells have identical genomes...

There are on the order of 10^{13} cells in the human body. This population is made up of hundreds of unique cell types, each with specific molecular and cellular functions. Current gene therapeutic approaches involve treating disorders by delivering plasmid DNA or viral vectors that express therapeutic genes. These genes are expressed continuously in all cell types. Yet many diseases only affect specific cell populations. Expression of exogenous genes in unaffected cell populations is unnecessary and potentially deleterious. So how might we go about turning genes on and off in a cell-type specific manner? One way is to try and find regulatory regions of the genome that do this!

In this paper, we apply machine learning methods to try and determine the cell-type specificity and activity of non-coding genomic sequences.

1.1 Background

In the past few years, thousands of noncoding genomic elements with unique tissue specific expression patterns have been identified. As manipulable regulatory elements, these elements are superior to larger promoter regions because of:

- Size: small, on the order of 0.1kb – 2kb
- Complexity: regions dictate simple interactions with minimal promoters
- Context: gene expression controlled by these elements is independent of orientation and genomic location
- Function: regions can work to either activate or repress gene expression

With these advantages in mind, these smaller noncoding genomic regions are interesting to study in terms of their cell-type specificity. These regions are located by open chromatin assays– assays that look for areas of the genome that are not bound up in chromatin, the protein structures around which DNA is wound. These small areas are likely enhancer or repressor regions; transcription factors will bind to these regions and affect expression of genes–both proximal and distal.

Our approach involves analyzing open chromatin data with the goal of discovering which noncoding genomic elements are active in a given cell type. Identifying such elements could not only give insight into how these noncoding elements work to affect transcription, but could also help direct approaches in gene therapeutics. Many researchers have used machine learning to try and predict different attributes about genomic sequences [1,2,3]. The first part of this paper is a repurposing and reevaluation of methods used in [1, 3]. The goal is to discern advantages and disadvantages of these methods. The second part of this paper is an exploration of deep learning applications to predict expression and specificity of these short non-coding elements.

We divide our approach into two branches: Expression (turning things on and off) and Cell-type specificity (flipping the right switch).

2 Supervised learning

2.1 Data

We use an expression dataset of 2098 predicted enhancer elements from ENCODE, including enhancer and repressor regions from K562 and H1-hESC cell lines. These regions were functionally tested in [3]. For each sequence (121-130bp long), there is a readout of luciferase expression in K562 cells. The dataset includes scrambled sequences as negative controls. We aligned this dataset with 105 histone markers and transcription factor binding site information from UW Encode for the K562 cell line.

To test for cell-type specificity, we look at two example cell lines: Human astrocytes (HAC) and cardiac myoctyes (HCM). For each cell line, we use Dnase-seq data pulled from the UW ENCODE repository [4]. Each cell line has two replicates for the assay. We also use histone mark H3K04me3 data from the UW ENCODE repository. There are 196507 and 211215 unique sequences attributed to HAC and HCM, respectively. The two classes are pretty well balanced. Assay peak locations for each replicate were concatenated and then sorted by chromosome and location. Overlapping regions across replicates were merged and non-overlapping (unique) regions across cell-types were found. Histone marks for each cell type were aligned with the sequences. To clarify, if a histone mark appeared in a location for an experiment in HAC, that mark was aligned with open chromatin locations in HAC, but not in HCM (and vice versa). Finally, sequence data corresponding to chromosomal locations were extracted.

2.2 Feature extraction

For each dataset, we built three feature sets.

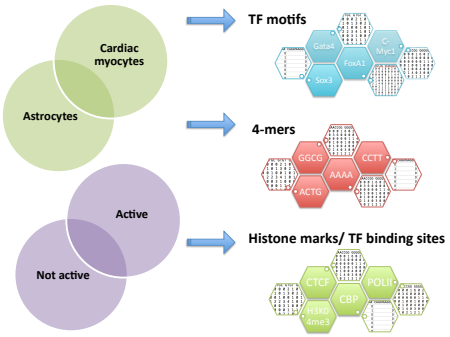

Figure 1: Feature Extraction

- Tf motifs: We used TESS [5] to predict transcription factor binding of 146 known factors (Position weight matrices from HOMER [6]) within each sequence and created a feature set of motif counts.
- Histone marks: We aligned each sequence with publicly available data on histone marks and transcription factor binding sites specific to the cell type in consideration. For each sequence location, we report the number of marks found within that location. For the expression dataset, we have 105 factors and for the cell-type specificity dataset we have a single factor.
- 4-mers: We extracted the occurrence counts of all possible 4-mers for each sequence (256 combinations).

Our feature sets are small in dimension relative to the number of training examples, which led us to expect high bias from linear models. As a result, we chose not to do feature selection immediately and went straight to testing the models.

2.3 Experiments

2.3.1 Expression

Model Accuracy	**Naïve Bayes**		**SVM (linear)**		**SVM (RBF)**			
Feature set	Train	Test	Train	Test	C	gamma	Train	Test
Tf motifs	0.727	0.668	0.909	0.738	10	0.001	0.862	**0.795**
Histone marks/TF binding	0.282	0.256	0.829	**0.777**	1E-04	1E-04	0.788	0.792
4-mers	0.701	0.647	0.96	0.686	10	0.001	0.913	0.775
Joint	0.463	0.411	1	**0.71**	10	0.001	0.802	**0.794**

Table 1: Results for prediction of sequence expression in K562 cells. Models were run with 5-fold validation. Results for training on 44% of the data (922/2098 sequences), tested on 56% (1176/2098 sequences). Grid search for parameters C, gamma on SVM with RBF kernel done over base 10 logspace(-3,2). Data scaled to [0,1] and shuffled before training.

We thresholded the expression data for sequences using the 90th percentile expression level for the scrambled sequence which serve as the negative control, as done in [3]. This gives binary output, which works well with many linear models.[1] We implemented a few of these models in Python, first keeping the feature sets separate. The models display high bias–additional training examples are unlikely to improve model performance. For Linear SVM, training on the feature set of known histone marker and TF motif binding events led to better performance than training on the other feature sets; however, when we used an SVM with an RBF kernel, thereby accounting for nonlinear combinations of features, the performance of the other two models greatly improved.

We then combined our feature sets in an attempt to understand if integration of diverse datasets might boost model performance as seen in [1] (See Figure 5 in the Appendix). We find that the joint model does not outperform the best feature sets in terms of test accuracy, but the model displays less bias. With more training examples, we expect the prediction capability of the joint model to improve more so than the individual feature sets.

Overall, we see that known histone mark overlap and transcription factor binding are the best predictors of activity for open chromatin sequences, although we have poor prediction capability. Our results call into question taken by researchers looking to predict enhancers in [3], as we are not able to build an unbiased, scalable model, even with a similarly constructed feature set.

2.3.2 Specificity

We tested a number of classification models on the cell-type specificity dataset and report our primary results in Table 2. While the models display expected bias, the test accuracy is quite high given the difficulty of the classification task, the limited feature set, and the small number of training examples. As with the expression dataset, the TF motif feature set outperforms the 4-mers for SVM with linear and RBF kernels. Feature elimination improves model accuracy but exacerbates bias (See Section 6.3.1 in the Appendix). Combining the three feature sets leads to improved performance for both models, with a test accuracy of .794 for SVM (RBF). This joint feature set includes a single column for known histone mark H3K04me3. To assess the relative importance of this feature, we remove it from the joint dataset and reevaluate our models. As expected, training accuracy decreases slightly with a simpler model. Counterintuitively, we found that the test accuracy increased when we removed histone mark data, indicating that perhaps histone mark data was making it more difficult to differentiate the two cell types. Since we are only looking at a binary classification problem and a single mark, it is impossible to make any conclusive generalization from this result– we can say no more except that it is interesting!

SVM with the RBF kernel with our joint feature set outperforms the other models and feature sets (See Figure 4). It seems like combining an unbiased approach for feature selection with one informed by current scientific knowledge helps improve the model performance. The kernel trick seems to help mitigate bias by increasing the complexity of the model; sill, this model displays high bias. We could try ensemble methods or try fleshing out the feature space, or we could try something unsupervised. Let's try something unsupervised!

[1] Regression results are listed in Table 4 in the Appendix

Accuracy	**Naïve Bayes**		**SVM (linear)**		**SVM (RBF)**	
Feature set	Train	Test	Train	Test	Train	Test
Tf motifs (HOMER)	0.665	0.654	0.712	0.677	0.722	0.682
4-mers	0.602	0.581	0.712	0.66	0.735	0.666
4-mers (PCA)			0.705	0.666		
Joint (w/ histone)	0.631	0.615	0.77	**0.684**	0.814	**0.705**
Joint (w/out histone)	0.631	0.615	0.768	**0.695**	0.812	**0.706**

Table 2: Results for prediction of sequence specificity in human astrocytes (HAC) or cardiac myocytes (HCM). Models were run with 5-fold validation. Results for training on 80% of a subset of the data (4000/5000 sequences), testing on 20% (1000/2000 sequences). For SVM with RBF kernel identical parameters were used across models: C=1 and gamma=.001 for Grid search for parameters C, gamma on SVM with RBF kernel done over base 10 logspace(-3,2). Data scaled to [0,1] and shuffled before training.

3 Unsupervised feature detection

Unfortunately there is not nearly enough reliable expression data for putative enhancer elements. But there are thousands of open chromatin sequences that are uniquely associated with specific cell types and organ systems and using these, we might begin to try and extract features in an unsupervised manner.

3.0.3 Data

We built an unbiased training set made up of flattened sequence data. We selected a random 150bp patch from each sequence from our cell-type specificity dataset, encoded it in a 4-D binary tensor and then flattened the data into a column vector. The feature set is therefore ('A in position 1, C in position 1, T in position 1, G in position 1, A in position 2 ... T in position 150, G in position 150'). Unlike the 4-mer feature set, here we encode position and ordering information. This is actually restrictive: we are selecting random patches from the sequence, but the feature set is a fixed representation of base pairs in precise locations and is not translationally invariant. Ideally we would build a network that first looks only at local information (a "receptive field") and then later combines local information to get at global context (see Section 6.2.2).

3.0.4 Experiments

We first looked at how linear models perform on this sparse representation as a baseline. Models displayed high training and test errors, with maximum accuracy across models around 51%.

In Table 3 we report some results of experiments with autoencoders. We train the autoencoders on 100K sequences and then use the architecture to train a feed-forward neural network and make predictions on test sequences. For the 6 layer stack, we experimented with having the first layer learn an overcomplete representation of the input data by having more hidden neurons than input dimensions–in essence "extracting" information–and then having subsequent layers map to lower dimensional space. We achieve highest performance with this architecture with a test error of 0.46. Overall, we have pretty terrible classification results.

Our sense is that we might need orders of magnitude more training data to learn features. The input data classes are perfectly balanced, so the network might be learning something, since errors are consistently less than 50%; however, it is not clear that this approach will work very well, since encoded features are not translationally invariant.

Num. Layers	**Architecture**						Error			
							Train (40K)	**Test (10K)**	**Train (4K)**	**Test (1K)**
1	400						0.437	0.491	0.411	0.484
3	400		100		50		0.425	0.474	0.377	0.489
6 (no extract)	400	100	50	25	10	5	0.45	0.477	0.299	0.526
6 (extract)	800	400	100	25	10	5	0.449	**0.475**	0.465	**0.46**

Table 3: Unsupervised feature detection with (stacked) sparse autoencoders and testing on cell-type specificity data. Autoencoder layers with sigmoid activation trained on 100K sequences encoded in 600x1 column vectors. Feed-forward neural network (logistic regression) trained on 4K sequences, tested on 2K. Training on autoencoder layers trained for 1 epoch. Feed-forward neural network trained for 3 epochs.

Tiled Convolutional Neural Networks are convolutional neural nets that allow layers to have a tiled pattern of weights– adjacent hidden units do not need to share weights. This is useful for this project because TCNNs can encode translational invariance of features but also preserve local context of information. We have implemented TCNNs pretrained with Topographic ICA (TICA). This is not at all a thorough analysis, but a quick and dirty attempt to get nicer results and to try out a more refined model. We cut off 76 bp from each training example, reformatted our sequences into 12-by-12 matrices, and implemented a two layer TCNN (L1 window size= 8, L2 window size =3, maps = 6, tile size=2). Our first (and only!) attempt using a TCNN achieves 0.422 training error and 0.482 test error pretraining on 100K sequences and training/testing on 4K/1K sequences. This reflects an improvement over our experiments with stacked sparse autoencoders of similar depth.

4 Discussion and Conclusion

In this paper we attempt to discover whether we can predict non-coding genomic sequence expression and cell-type specificity using linear and nonlinear models. We find that creating a feature set incorporating data from diverse sources by hand allows us to classify sequences with respectable accuracy, but doing so limits the scope of generalization of the model.

While it is important to classify sequences accurately, a more interesting question is: can we use the machine to point us in the direction of novel hypotheses? That is, can we use the machine's predictions to discover biological mechanisms previously obscured? The extent to which we can do this well depends on the data sets and feature set(s) we select. Limiting or biasing the feature set will in turn limit the scope of possible interpretation of the results. For example, if the feature set only includes known TF binding motifs, the hypotheses generated about what precisely differentiates regulatory sequences across cells will only factor in known TF binding motif data– there is no room here to generalize. Suppose the feature set only included histone mark or TF binding data. Then the accuracy of our prediction and any

subsequent hypotheses inferred will be biased by the nature of the experimental assay used to generate those marks. The extent of this bias is reflected in our first set of results and in results for papers with similar methods [1], [3].

Why not add more features? First of all, experiments are expensive! In the expression dataset, because of the cell type used (K562) in the original experiments, we were able to procure lots of cell-type specific data on histone marks and transcription factor binding. But in the case of our cell-type specific analysis, the amount of metadata available was quite limited. For a given biological assay, there is a vast array of possible cell types, organ systems, and experimental conditions to consider. It is not at all feasible to perform experiments covering all these combinations, simply in order to improve our classifiers! Therefore, it is important we discover a way to classify open chromatin sequences according to cell type with minimal to no additional data. Furthermore, as we consider looking at more cell types–a multiclass classification problem– we see that the number of training examples will scale up exponentially with number of cell types or organ systems included, whereas the number of features will either remain constant or scale linearly.

So what can be done? Perhaps we can try and be utterly unbiased– look at just the sequence data and nothing else. In our supervised analysis we used a feature set of overlapping 4-mer counts across sequences. We see that with linear models, this feature set performs similarly and slightly worse that the TF motif feature set data. Bias persists! There are a few reasons why this might be the case. First, the feature set is still quite small (256 features) and therefore the complexity of the model we might build is bounded. Second, the count matrix for 4-mers is less sparse than the TF motif matrix whose motifs are around 8-15bp long. This means there is more noise in the dataset– lots of low values populating the data with few outlier values. Another issue with this feature set is that different features are likely to correlate precisely because we are looking at all combinations of 4 base pairs in a sequence. We saw with our PCA analysis that reducing the dimensionality of the feature space does not help us with our bias problem, but does help us make a slight improvement in classifying the data accurately.

Although our unsupervised feature detection results are quite poor, we will continue experimenting with different models and architectures to try and improve our results. We will continue refining unsupervised feature learning methods, focusing on using tiled convolutional neural networks to boost prediction accuracy. We might begin to visualize features detected and process them in an attempt to understand the kinds of motifs/motif combinations that lead to specificity of gene regulation.

Overall we see decent performance of our linear models, which improve when we have lots of external metadata. As expected, histone marks and TF binding events best predict the extent to which a sequence affects gene expression, but it is less clear what determines which regions are "open" in which cell. Models improve in accuracy when nonlinear combinations of features are taken into account, which supports the hypothesis that combinations of DNA motifs lead to different regulatory outcomes. We definitely need more expression data to scale that analysis, but for the question of specificity, it's clear that the amount of genomics data is scaling much faster than features– given this, how do we eliminate bias? Perhaps through unsupervised feature detection and deep learning, though we were pretty unsuccessful in this first and fast attempt. With our results, it seems like hacking the genome is possible, but we have to watch out for bias in our models.

5 References

[1] Erwin, Genevieve D., Rebecca M. Truty, Dennis Kostka, Katherine S. Pollard, and John A. Capra. 2013. Integrating diverse datasets improves developmental enhancer prediction. ArXiv Preprint arXiv:1309.7382.
[2] Smith, Robin P., Leila Taher, Rupali P. Patwardhan, Mee J. Kim, Fumitaka Inoue, Jay Shendure, Ivan Ovcharenko, and Nadav Ahituv. 2013. Massively parallel decoding of mammalian regulatory sequences supports a flexible organizational model. Nature Genetics 45 (9): 1021-8.
[3] Kwasnieski, J. C., C. Fiore, H. G. Chaudhari, and B. A. Cohen. 2014. High-throughput functional testing of ENCODE segmentation predictions. Genome Research 24 (10) (Oct): 1595-602.
[4] ENCODE Project Consortium. 2011. A user's guide to the encyclopedia of DNA elements (ENCODE). PLoS Biology 9 (4): e1001046.
[5] http://www.cbil.upenn.edu/tess
[6] Heinz, Sven, Christopher Benner, Nathanael Spann, Eric Bertolino, Yin C. Lin, Peter Laslo, Jason X. Cheng, Cornelis Murre, Harinder Singh, and Christopher K. Glass. 2010. Simple combinations of lineage-determining transcription factors prime cis regulatory elements required for macrophage and B cell identities. Molecular Cell 38 (4): 576-89.
[7] Ngiam, Jiquan, Zhenghao Chen, Daniel Chia, Pang W. Koh, Quoc V. Le, and Andrew Y. Ng. 2010. Tiled convolutional neural networks. Paper presented at Advances in Neural Information Processing Systems.

6 Appendix

6.1 Implementation

Data parsing of genomics sequences was made simple by the BEDTools suite. Sequence alignment was done on AWS (Amazon Web Services) EC2 instances. Machine learning models were built in Python, primarily using the Scikit-learn toolbox. Sparse autoencoders and neural networks were implemented in MATLAB using a combination of self-written code, functions from DeepLearnToolbox, and code adapted from [7].

Email namrata.anand2@gmailcom for iPython notebook with all results and code.

6.2 Models

For supervised learning we use Naïve Bayes, linear SVM, and SVM/SVR with Gaussian (RBF) kernel. Regularization parameters were found by grid search.

6.2.1 Sparse Autoencoders

For unsupervised feature detection we use (stacked) sparse autoencoders and tiled convolutional neural networks. Sparse autoencoders are trained in a greedy layerwise manner for pretraining and then trained as a feed-forward neural network with an additional softmax classification layer in order to classify test data. Let $W^{(k,1)}, W^{(k,2)}, b^{(k,1)}, b^{(k,2)}$ represent

weight matrix and bias parameters $W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}$ for the kth autoencoder in a stack of n autoencoders. The encoding step of the SA is given by

$$a^{(l)} = f(z^{(l)})$$
$$z^{(l+1)} = W^{(l,1)}a^{(1)} + b^{(l,1)}$$

And the decoding step is given by

$$a^{(n+l)} = f(z^{(n+l)})$$
$$z^{(n+l+1)} = W^{(n-l,2)}a^{(n+1)} + b^{(n-l,2)}$$

The activations in $a(n)$ give us a representation of the input data in terms of higher order features. We can subsequently use these features in a linear model to classify test data.

6.2.2 Tiled Convolutional Neural Networks

Tiled Convolutional Neural Networks are convolutional neural nets that allow layers to have a tiled pattern of weights– adjacent hidden units do not need to share weights. This is useful in this project because TCNNs can encode translational invariance of features but also preserve local context of information.

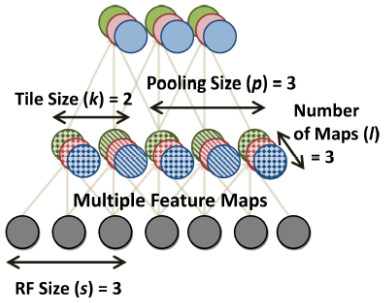

Figure 2: TCNN architecture (from [7]). TCNNs have partially untied local receptive fields in first layer with pooling across maps in second layer.

Learning algorithm is a generalization of Topographic ICA. The TICA network is a two layer network where the first layer learns weights W and the second layer weights V are fixed and represent the topological structure of the neurons in the first layer. Given an input pattern $x^(t)$, TICA learns parameters W by solving

$$\min_W \sum_{t=1}^{T} \sum_{i=1}^{m} p_i(x^{(t)}; W, V) \text{s.t.} WW^T = \mathbf{I}$$

where p_i are the units in the second layer which pool over local hidden units in the first layer. m is the number of hidden units in the first layer, T is the number of inputs, and the layer activations are square and square-root, for the simple units and pooling units respectively. For a more detailed treatment of the model see [7].

6.3 Supplementary Results

6.3.1 Feature elimination on cell-type specificity data

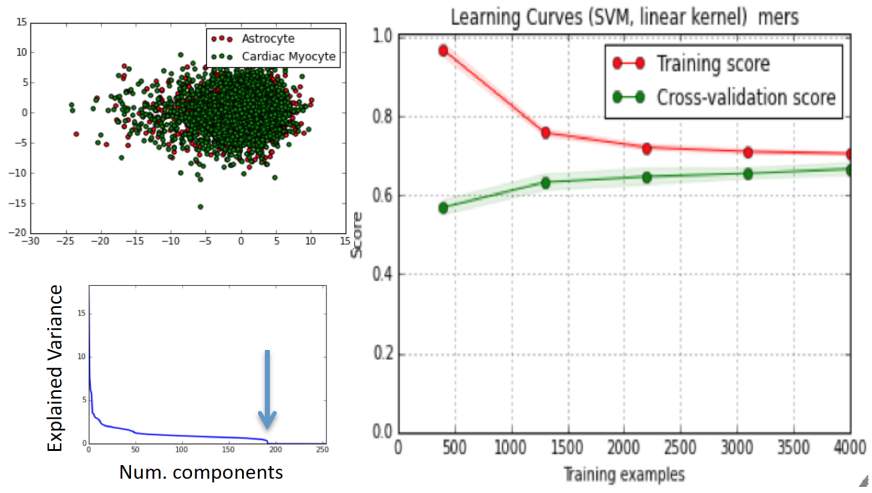

Figure 3: PCA analysis of 4-mer feature set for cell-type specificity analysis. Top left: Projection of data in the space of the first two principal components of 4-mer dataset (256 features). Training done on 4000 sequences, testing on 1000. Bottom left: Expected variance by number of components. Right: Linear SVM on first 200 components

We did not expect feature elimination to boost model performance since the models already display high bias. But in the case of the 4-mer feature set, we expect feature columns to be highly correlated. Consider the occurrence of 'AAAA' in a sequence, which is likely correlated with occurrence of 'AAAC' or 'GAAA,' less so with 'TCGC.' There are also less obvious correlations related to the evolutionary conservation of consensus sequences: for example, the TATA box (TATAAA) is a DNA sequence found in the promoter region of approximately 20% of human genes, and as a result, we might expect the appearance of 'TATA' to correlate with 'ATAA,' 'TAAA,' etc.

We performed PCA on the 4-mer feature set, and plotted the explained variance versus the number of principal components in Figure 3. We see drop-offs in variance around 10, 50, and 175 components. We selected the first 200 components (grid search done across 100, 150, and 200 components) and classified the data with a linear SVM model. We found that bias reduced to some extent (training error is low with few samples) but the model is still not complex enough.

6.3.2 Results for selected models–Expression

See Table 4 for SVR results. See Figure 4 for visualization of SVM (RBF) model performance.

6.3.3 Results for selected models–Specificity

See Figure 5 for visualization of SVM (RBF) model performance.

SVR (RBF)	**Parameters**		**Results**	
Feature set	C	γ	Training R^2	Test R^2
Tf motifs (HOMER)	10	0.001	0.335	0.156
Histone marks/TF binding (ENCODE)	1	0.1	0.293	0.057
4-mers	1	0.1	0.593	0.092
Joint	1	0.0001	0.762	0.196

Table 4: Results for prediction of sequence expression in K562 cells with continuous output. Models were run with 5-fold validation. Results for training on 1176/2098 sequences (80% of the data). Grid search for parameters C, gamma on SVR with RBF kernel done over base 10 logspace(-3,2). Data scaled to [0,1] and shuffled before training.

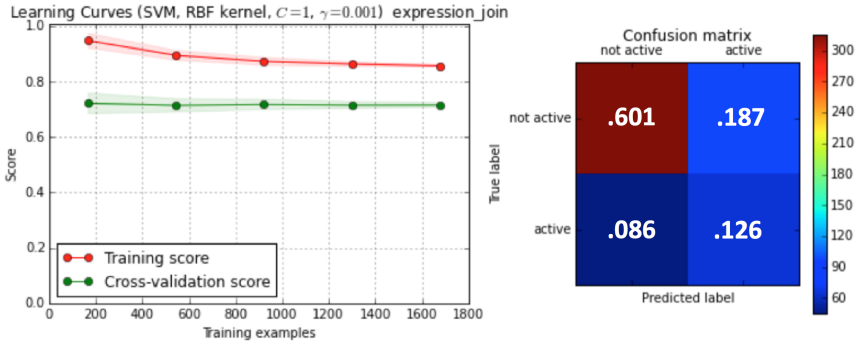

Figure 4: Results for prediction of sequence expression with SVM with RBF kernel for joint feature set (C=10, γ=.001). Training done on 1573 sequences, testing on 525. Right: Learning curve. Left: Confusion matrix.

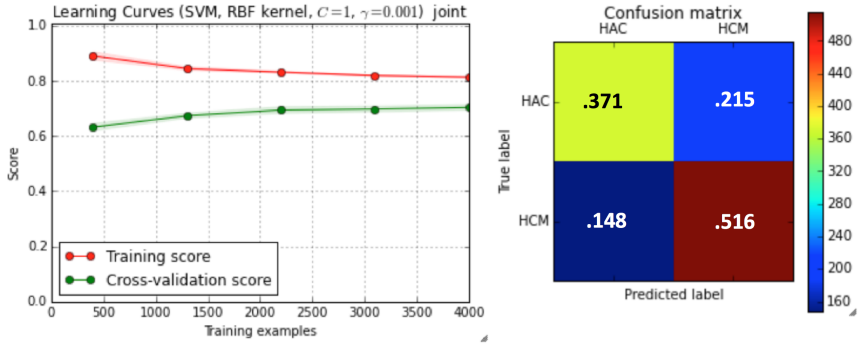

Figure 5: Results for prediction of sequence cell-type specificity with SVM with RBF kernel for joint feature set (C=1, γ=.001). Training done on 4000 sequences, testing on 1000. Right: Learning curve. Left: Confusion matrix.