Learning the topology of the genome from protein-DNA interactions

Suhas S.P. Rao, SUnet ID: suhasrao
Stanford University

I. Introduction

A central problem in genetics is how the genome (which measures 2 meters from end-to-end when stretched out) fits inside the nucleus of a cell that has a diameter on the order of microns wide). The cell has to be able to access precise portions of the genome with high temporal and spatial precision in order to be able to turn on and off genes in a cell-type specific manner, respond to external stimuli and regulate the functional state of the cell. This requirement for tight control of information retrieval from the genome implies a significant need for a deeply structured organization of genome topology, but yet our knowledge of the structural principles of the genome have been limited in the size scales between single nucleosomes (200 base pairs of DNA) and whole chromosomes (20-200 million base pairs of DNA).

We recently generated three dimensional maps of the human genome at unprecedently high (kilobase) resolution [1]. These maps allowed us to uncover many structural principles of the genome, including the positions of all chromatin loops across the genome. Chromatin loops (roughly 10,000 across the genome) bring together small pieces of DNA (hundreds to thousands of base pairs long) that are separated by long one-dimensional distances along the genomic polymer. The anchors of the loops are enriched for non-coding regulatory elements, and as such these loops play an important role in controlling the transcriptional output of the cell, acting as switches to turn genes on and off and setting up contact domains (intervals of chromatin forming preferential contacts within the interval) that act as insulated regulatory neighborhoods in the nucleus [1].

We have shown that these loops are set up by at least two proteins: CTCF and cohesin [1]. Nearly all loop anchors (>87% by a conservative estimate) are bound by CTCF and the cohesin subunits (RAD21, SMC3, etc.) as measured by ChIP-Seq experiments (that measure protein-DNA interactions across the genome). Since CTCF recognizes a specific 20 basepair motif, we were able to integrate protein binding information and the motif information to localize the majority of loop anchors down to single base pair resolution. Additionally, the CTCF motif is asymmetric, it has a direction along the 1D genome polymer. For any pair of loop anchors that form a loop, the random expectation would be that any orientation of two motifs relative to each other would be equally probable (the motifs pointing towards each other, pointing away from each other, both pointing to the left, both pointing to the right). However, we found that nearly all loops (>90%) occurred between motifs pointing towards each other (the convergent orientation) [1]. The precision and accuracy of our loop annotations has been verified via extensive genome editing experiments; by adding as little as one base pair to a CTCF motif annotated as a loop anchor, we were able to re-fold the genome on the scale of millions of bases [2].

II. Related Work

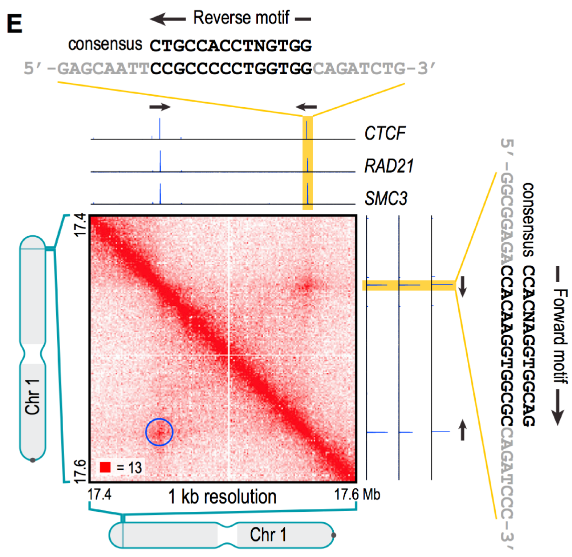
While we have systematically mapped loops genome-wide in roughly a dozen different cell types, there is great need for a effective method to predict the positions of loop anchors and loops. Knowledge of looping patterns in a cell type is crucial to decoding the regulatory network of that cell type, but making high-resolution maps of the 3D genome remains a fairly expensive endeavor (at least relative to the cost of mapping protein binding sites across the genome). We have recently shown that maps of the 3D genome can be fairly accurately recapitulated simply with a knowledge of where loop anchors are [2]. Protein-DNA interactions (both transcription factors and histone modifications) have been mapped in detail in a number of cell types by large numbers of groups around the worlds, including major consortiums like ENCODE and the NIH Epigenomics Roadmap Initiative [3].

Prior studies have attempted to use protein-DNA interaction data to predict genome structure at low resolutions [4], but no one has attempted to learn the positions of chromatin loops using protein-DNA interaction data. Since we have identified proteins (CTCF and cohesin) that tend to always be present at loop anchors, using protein-DNA interactions to infer loop anchor positions (and potentially pairings) seems like a promising approach to be able to infer genome topology across a number of cell types. However, knowledge of just CTCF and cohesin binding sites is not enough, as only a small fraction of CTCF and cohesin binding sites actually participate in looping interactions (between 15-35% of binding sites) [1]. Here, we propose to use protein-DNA interaction data sets to both learn distinguishing features of loop anchors and predict their positions genome-wide.

III. Data

Gold standard chromatin loop annotations for the GM12878 cell line (a human B-lymphoblastoid cell line) were obtained from [1]. By combining these annotations with CTCF, RAD21 and SMC3 ChIP-Seq data from ENCODE [3], we identified 6987 high-confidence loop anchors (down to motif resolution) as well as 43,637 CTCF sites that do not participate in loop interactions (for simplicity, CTCF sites that were not unique inside loop anchors were left out of the analysis, even when the correct looping CTCF could be identified via the convergent rule [1]).

Figure 1: Example of a chromatin loop



Legend: An example of what a chromatin loop looks like in a DNA-DNA interaction dataset (heatmap); the circled focal peak represents a chromatin loop. Protein-DNA interaction tracks shown for CTCF and cohesin above and to the right of the heatmap, illustrate the relationship between protein binding and 3D genome structure, and thus the potential to predict DNA-DNA interactions using only protein binding and sequence information. (reproduced from [1])

For features, we downloaded ChIP-Seq data for 87 transcription factors and histone modifications as well as DNAse accessibility data from ENCODE [3]. For CTCF and cohesin (RAD21 and SMC3) ChIP-Seq data, in order to retain information about strength of protein binding, we discretized the data by calculating average protein binding signal over each of the sites in our gold standard annotation, and then converting the signal to a [0-9] value by dividing the distribution of signals into deciles. For all other transcription factors/histone modifications, we discretized the data simply by peak calling (i.e. binary for peak or no peak). For the sites in our annotations, feature $x_i = 1$ if there was a peak overlapping the 1000bp window surrounding the site and 0 otherwise. To deal with multiple replicates of a particular protein, signals from replicate ChIP-Seq experiments were averaged and peaks called in any ChIP-Seq replicate were included (even if they were not called in all of the replicates). In addition, we used data from an experiment that measured nucleosome positioning genome-wide. For each CTCF binding site, we included the nucleosome signal 150bp upstream of the site, 150bp downstream of the site, and the difference in the two signal (again decile-binned as with the CTCF/cohesin ChIP-Seq signal). Finally, in addition to protein-DNA interaction features, we included a few sequence features: log-likelihood motif match scores against two different position-weight matrices for the CTCF motif and PhyloP 100-vertebrate conservation scores (downloaded from UCSC) averaged over the bases in the CTCF motif corresponding to a CTCF binding site. These signals were also decile-binned as above.

IV. Methods

IV.1. Models

As a first attempt, we trained a Gaussian Naive Bayes classifier (`sklearn.naive_bayes.GaussianNB`), a Logistic Regression classifier (`sklearn.linear_model.LogisticRegression`) and three different SVM classifiers (`sklearn.svm. SVC`) using a linear, polynomial and RBF kernel on our data [5]. We randomly selected 30% of our annotation to hold out as a test set, using the remaining 70% as our training set (keeping the proportion of positive and negative examples balanced between the training and test set).

Gaussian Naive Bayes: Assuming that the features were conditionally independent given the looping status of a CTCF binding site, we maximized:

$$\ell(\theta_y, \theta_{j|y=0}, \theta_{j|y=1}) = \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)})$$

for all parameters $\theta_y = p(y = 1), \theta_{i|y=0} = p(x_i = 1, y_i = 0), \theta_{i|y=1} = p(x_i = 1, y_i = 1)$ where i is the number of training examples (in this case roughly 32,000) and j is the number of features (97). Note that in this case, for some of the θ_js (corresponding to the decile-binned signals), there were actually 9 parameters, rather than just 1, since the feature could take on any value from 0-9. Also, note that the assumption of conditional independence given y employed in the Naives Bayes algorithm is actually highly problematic in this case, as many of the features are highly correlated with one another (for instance CTCF and cohesin binding).

Logistic Regression: We employed logistic regression with L2 regularization, i.e. we maximized:

$$\ell(\theta) = \log \left(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta) \right) - \lambda ||\theta||_2^2$$

where $p(y^{(i)}|x^{(i)} = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$ and $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$. This fits a sigmoid function to our data to separate in into two categories (looping or non-looping). L2-regularization (the $-\lambda||\theta||_2^2$ term above) was introduced to penalize overfitting.

Support Vector Machines: We trained L1 regularized SVMs by solving the following optimization problem (the dual formulation of the problem is given):

$$\max_\alpha \; W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$
$$\text{s.t. } 0 \leq \alpha_i \leq C, i = 1, \ldots, m$$
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

where $\langle x^{(i)}, x^{(j)} \rangle$ varied between the following:

$\langle x^{(i)}, x^{(j)} \rangle = (x^{(i)})^T x^{(j)}$ linear kernel

$\langle x^{(i)}, x^{(j)} \rangle = (x^{(i)})^T x^{(j)} + x)^3$ polynomial kernel

$\langle x^{(i)}, x^{(j)} \rangle = \exp\left(-\frac{\gamma ||x^{(i)} - x^{(j)}||^2}{2\sigma^2} \right)$ Gaussian kernel

Different kernels and different parameters for C (the penalty parameter) and γ (the kernel coefficient) were utilized to maximize the separability of the data, both because it was not known whether the data was linearly separable, and because it was possible that there were errors in the training set that could render the data not separable. For an initial pass, we used $C = 1$ and $\gamma = \frac{1}{\text{\# of features}}$.

IV.2. Grid search

To optimize the hyperparameters for our RBF-kernelized SVM classifier, we performed a grid search over various values for both C and γ. Specifically, we ran our classifier on the training data for values of $C = 10^i$, $\forall -5 \leq i \leq 5$ and $\gamma = 10^i$, $\forall -5 \leq i \leq 5$. We chose the parameters that minimized the generalization error on our test set as the optimal parameters for our classifier.

IV.3. Feature selection

To identify features that were predictive of loop anchor potential, we performed both backwards search and feature filter selection on our 97 features. More specifically, we performed recursive feature elimination (RFE), by training a linear SVM classifier on our data, choosing the feature with the lowest weight, removing it from out feature set, retraining the classifier and repeating until we pruned our list down to 1 feature [6]. We also experimented with first throwing out low variance features (for example, features that had the same value on >95% of examples) before performing RFE. We also calculated the mutual information between each feature and our loop anchor annotation. Since we were ranking features with different domains (i.e. binary features and decile clustered features) and the mutual information is generally higher for features with higher numbers of clusters, we also calculated an adjusted mutual information accounting for chance using sci-kit learn's `adjusted_mutual_info_score` function [5].

V. Results

V.1. Classifier Results

We first implemented five different learning algorithms on our training set (using all of the features), to get an initial sense of performance of the various algorithms.

Table 1: Initial Classification Results

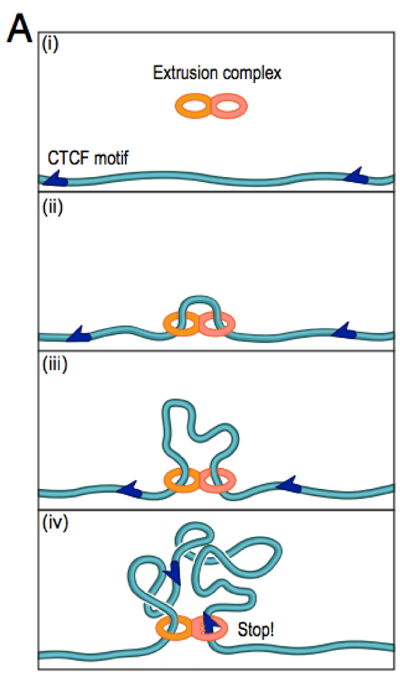
Model	Train Err	Test Err	Sens	Prec
NB	0.25	0.25	0.96	0.35
LR	0.10	0.10	0.63	0.67
SVM (L)	0.10	0.10	0.64	0.69
SVM (P)	0.09	0.09	0.63	0.67
SVM (RBF)	0.09	0.09	0.60	0.68

Legend: NB: Naïve Bayes; LR: Logistic Regression (L2 regularization); SVM (L): SVM with linear kernel and C=1; SVM (P): SVM with polynomial kernel (degree 3) and C=1; SVM (RBF): SVM with RBF kernel and C=1.

Logisitic Regression and the various SVM classifiers performed similarly, showing a sensitivity of about 63% and a precision of about 68% (see Figure 1 and Figure 2). (We chose to measure precision rather than specificity as precision provides a better idea of the utility of the list for predictions of genome topology.) The Naive Bayes classifier had a much higher sensitivity, correctly classifying nearly all of the looping CTCF sites, but a much lower precision of about 35%. This is likely due, as noted above, to the incorrect assumption of conditional independence of the features given information about propensity to form a chromatin loop.

We noted after this initial classification attempt that we had made an incorrect assumption about our dataset. Namely, we had assumed that any CTCF that is not seen participating in a chromatin loop is thus a negative example. However, we have recently provided evidence that chromatin loops form through a process of extrusion, that is a complex binds two places close together and then slides in opposite directions until it hits two correctly oriented CTCF sites capable of acting as brake points and participating in the formation of a loop [2]. Loop formation through extrusion suggest that only CTCF sites contained within an extant loop, i.e. sites that would have been passed over by the extrusion complex, can be correctly considered as negative examples. CTCF sites outside of loops would not have been sampled by the extrusion complex and hence their propensity to form a loop cannot be ascertained. When we restricted our data set to only include negative examples that were CTCF sites not participating in a loop but contained within a loop, our performance jumped (for all classifiers).

Figure 2: Chromatin loop formation through extrusion



Legend: An illustration of the extrusion model for chromatin loop formation. A complex binds to two places on DNA that are very close together and then slides in opposite directions until it hits two correctly oriented CTCF sites that act as brake points, forming a loop. Reproduced from [2]).

On average, our logistic regression and SVM classifiers demonstrated a sensitivity of about 75% (a 12% increase in performance) and a precision of about 73% (a 5% increase in performance). In general, our classifiers did not seem to be overfitting the data, similar training errors and test errors were seen for all classifiers. For further analyses, we utilized the SVM (RBF) classifier.

Table 2: Classification Results After Dataset Modification

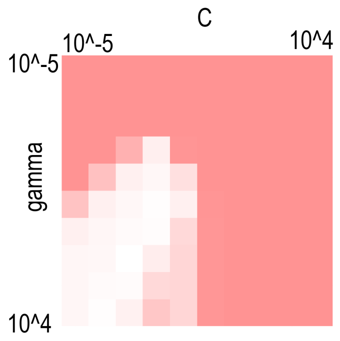
Model	Train Err	Test Err	Sens	Prec
NB	0.23	0.23	0.94	0.43
LR	0.10	0.10	0.74	0.72
SVM (L)	0.10	0.10	0.82	0.68
SVM (P)	0.08	0.09	0.75	0.73
SVM (RBF)	0.09	0.09	0.76	0.73

Legend: NB: Naïve Bayes; LR: Logistic Regression (L2 regularization); SVM (L): SVM with linear kernel and C=1; SVM (P): SVM with polynomial kernel (degree 3) and C=1; SVM (RBF): SVM with RBF kernel and C=1.

V.2. Hyperparameter Optimization

To optimize the hyperparameters for our support vector classifier, we performed a parameter sweep through a range of values for the penalty parameter, C, and the kernel coefficient, γ. Optimal performance (minimized test error) was seen for parameter values $C = 100$ and $\gamma = 0.001$. When C was very small, we saw that all examples were classified as negative examples, likely because the data was not linearly separable. For our best classifier, we saw a sensitivity of 77% and a precision of 73%.

Figure 3: Grid Search Heatmap



Intensity (redness) indicates greater generalization error. Each pixel indicates the generalization error using a particular value of C and γ.

Table 3: Confusion Matrix for SVM RBF classifier

	Positive (Truth)	Negative (Truth)
Positive (predicted)	1643	594
Negative (predicted)	495	9287

V.3. Feature selection

We ranked features based on their mutual information (or adjusted mutual information) with the loop anchor annotation (see Table 4). As expected, features related to CTCF and cohesin ranked highly among the features. We also see that the proteins ZNF143 and YY1 are highly ranked features. Previous studies have suggested a relationship between ZNF143 and YY1 and CTCF, but the ability of YY1 and ZNF143 to discriminate between looping and non-looping CTCF binding sites is promising. Surprisingly, binding strength of CTCF and cohesin appeared to be highly discriminating features between looping and non-looping CTCF binding sites.

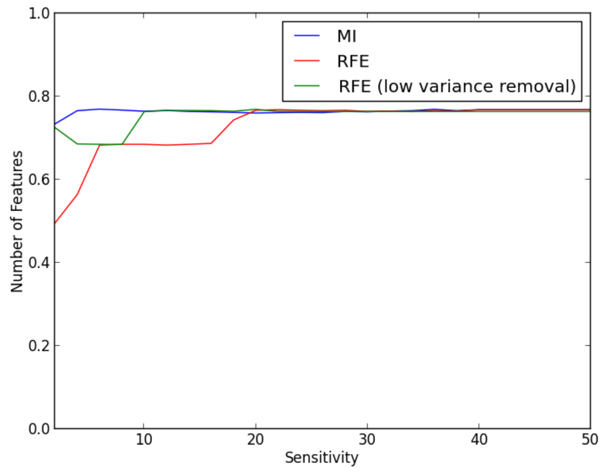
We also attempted to rank features using recursive feature elimination. This method did not seem to work. On initial glance, the results seemed promising, as some of the highly ranked features were the same (SMC3 peak presence, SMC3 binding strength, RAD21 binding strength), and others were intriguing (for example POLR3G binding, as POLR3G transcribes tRNAs and house keeping genes which have been suggested to have an insulating role). However, on further examination, it became clear that many of the features highly ranked by recursive feature elimination had extremely low variance across the training data. For example, the feature 'CTCF peak' had the value '1' for all examples by definition (we are predicting the looping propensity of CTCF binding sites). Even after removing low variance features, the results for recursive feature elimination still showed worse performance that ranking using mutual information.

Table 4: Feature Rankings

Ranking	Feature (MI score)	Feature (RFE))
1	SMC3 BS	SMC3 peak
2	RAD21 BS	CTCF peak
3	CTCF BS	SMC3 BS
4	SMC3 peak	PBX peak
5	RAD21 peak	BRCA1 peak
6	ZNF143 peak	EBF1 peak
7	conservation	POLR3G peak
8	motif 1 strength)	NFYA peak
9	DNase peak	H4K20me1 peak
10	motif 2 strength	RAD21 BS

Results of filter feature selection. Feature (MI score) are the top 10 features ranked by mutual information between the feature and loop anchor annotation. Feature (RFE) are the top 10 features ranked using recursive feature elimination. BS=Binding Strength.

Figure 4: Sensitivity vs. Number of Features

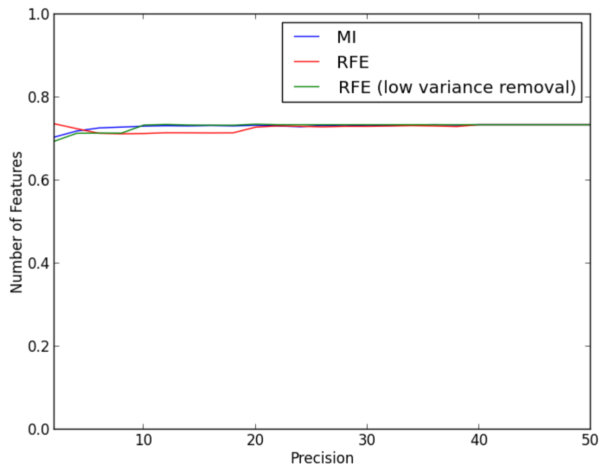


Performance for our SVM RBF classifier using the MI feature ranking plateaued at about 5 features, indicating that most protein binding data other than CTCF and cohesin is really unnecessary for predicting the probability of a CTCF binding site to loop or not (see figure 4 and 5).

V.4. False Positives

We wondered why there was such a high false positive rate for our classifier. When we examined the distance from the separating hyperplane for false positive examples versus examples that were true positives, we found that the false positives were on average further from the separating hyperplane. This was striking; there was no reason to believe that with features like CTCF and cohesin binding strength, at some point increasing strength of binding would decrease looping probability. Thus, we hypothesized that this may be a misclassification error in the data set. To test this, we chose 5% of false positive examples randomly and examined them in the DNA-DNA interaction data from [1]. We found that 18 out of 30 false positive examples were actually misclassification errors, i.e. a loop hadn't been called in the DNA-DNA interaction data where there should have been a loop, or the association between a called loop and a particular CTCF binding site wasn't properly made due to details in how the CTCF to loop anchor mapping was performed. Thus, it is likely that the precision of our classifier is actually closer to 90%. (It is possible that many of the false negatives are a result of misclassification errors as well).

Figure 5: Precision vs. Number of Features



VI. Conclusion

Here we present a classifier that can accurately and sensitively predict the positions of chromatin loop anchors given limited protein-DNA interaction data sets. It is easy to imagine extending this work to predict pairings of loop anchors as well. For instance, one can use probabilities of looping assigned by our classifier as inputs into another classifier, where features would be the probability of site 1 to form a loop, the probability of site 2 to form a loop, and the probability that an extrusion complex will get stuck somewhere in between site 1 and site ($\prod_{i=1}^{m}(1 - p(site_i))$ where i represents all the CTCF binding sites in between site 1 and site 2 and $p(site_i)$ represents the probability that $site_i$ forms a loop. Extending this work in that direction would allow for extremely powerful and interesting comparative analyses to be performed between cell types in an extremely cost-effective manner to better understand the role of chromatin organization in development and gene regulation.

References

[1] Rao, Suhas S.P., Huntley, Miriam H., Durand, Neva C., et al. (2014). "A 3D Map of the Human Genome at Kilobase Resolution Reveals Principles of Chromatin Looping." Cell 159, 1665-1680.

[2] Sanborn, Adrian L., Rao, Suhas S.P., Huang Su-Chen, et al. (2015). "Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes." PNAS 112, E6456-6465.

[3] ENCODE Project Consortium. (2012). "An integrated encyclopedia of DNA elements in the human genome." Nature 489, 57-74.

[4] Zhu, Jiang, et al. (2013). "Genome-wide Chromatin State Transitions Associated with Developmental and Environmental Cues." Cell 152, 642-654.

[5] Pedregosa, F. et al. (2011). "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12, 2825-2830.

[6] Guyon, I., et al. (2002). "Gene selection for cancer classification using support vector machines", Mach. Learn., 46(1-3), 389-422.