Abstract

Gene expression is modulated by transcription factors (TFs), proteins that generally bind adjacent to DNA coding regions. The relation between these regulatory proteins and their targets is generally many to many; each target can be regulated by more than one TF, and each TF can contribute to the regulation of more than one target. Among the consequences of combinatorial regulation is the emergence of control networks, which can regulate nuanced changes in phenotype in response to subtle environmental changes. A first step in developing a complete molecular understanding of transcriptional regulation is to associate each TF with the set(s) of genes that it regulates. In this paper we report on the ability of support vector machines to associate 104 TFs with their binding sites.

Several types of data are used to train classifiers for TF binding prediction in the \textit{Saccharomyces cerevisiae} genome: position-specific scoring matrix (PSSM) matches, conservation of PSSM matches in other genomes, gene expression, phylogenetic profiles, Gene Ontology functional annotation, TF-target expression correlation, and promoter sequence composition using various subsequence features. The classifiers are combined using a weighting scheme so that each type of genomic data can contribute to the classification of a binding site based on its performance. In each case the data are compared to known true positives taken from ChIP-chip data\textsuperscript{1,2}, Transfac\textsuperscript{3}, and the Yeast Proteome Database\textsuperscript{4}.

The SVM works best when all genomic data are combined, and it also rank orders data types by performance. On average, we can classify binding sites with a sensitivity of 73% and a positive predictive value of almost 0.9. New ideas and preliminary work for improving SVM classification on biological data are also discussed.

\textbf{Keywords}: Transcription Factor, Support Vector Machine, TFBS, binding site, motif.

Introduction

A first step in understanding transcriptional regulation is mapping transcription factors to the genes they regulate, and to the particular sequences that they bind. Typically TFs bind sites that are 10-15nt long. Even a cursory examination of the sequences that bind a particular TF indicates that they are not identical, but instead define a motif, or shared pattern of bases. The set of sites that bind a particular TF provides the raw material for computational methods that can be used to find additional sites. These
methods fall into two broad categories: supervised and unsupervised. The former starts with two sets of target sequences - those known to bind a particular TF and those known not to bind, which are used to derive a classification rule by SVM or some other learning algorithm. Unsupervised methods begin with sets believed, based on independent evidence, to be enriched in a characteristic but unknown pattern. A search algorithm such as Gibbs sampling can be used to identify the pattern. Many unsupervised techniques for predicting binding sites have been explored in the literature\textsuperscript{5-12}, and an excellent review of current motif-discovery and pattern analysis methods is available\textsuperscript{13}.

Our approach is meant to easily combine a large number of data types in a supervised learning scheme to more accurately predict the association of a transcription factor and its targets. There are a number of ways to proceed, including support vector machines and Bayesian variants, and an optimal approach might be to use a weighted sum of each. Assigning weights requires that we know the performance of each method. Here we explore support vector machines, which easily accommodate the high dimensionality of current genomic data sets and offer a simple framework for combining heterogeneous data.

A number of supervised approaches have been used to associate transcription factors with their targets. Simonis \textit{et al.} use linear discriminant analysis (LDA) to select from a set of potentially co-regulated genes those that are most likely to be co-regulated and therefore to share transcription factors. On a strict set 1012 regulatory interactions involving 66 TFs (taken from Transfac, the aMAZE database, and a list compiled by Young \textit{et al.} from the Yeast Proteome Database) they report an average positive predictive value of 0.91, and a sensitivity of 73%. Their classification performance based on ChIP-chip data is worse, with only 52% of genes identified by ChIP-chip being discovered. The authors argue that ChIP-chip results likely contain many false positives; however, their results also show that target groups identified by ChIP experiments contain large numbers of significantly overrepresented motifs as compared to random gene sets. This indicates that there is a true pattern in the data generated by high-throughput experiments like chromatin immunoprecipitation.

In an approach more closely related to ours, Qian \textit{et al.} apply support vector machines to microarray data in order to predict TF-target relations\textsuperscript{14}. Positive examples are known TF-target pairs; negatives are randomly chosen relations (note: contrary to our method where a classifier is made for each TF, Qian \textit{et al.} group all TFs into one classifier). The data for each known TF-target association is given as the concatenation of the TF’s and target’s expression vectors over 79 experimental conditions (giving a 158 element vector to describe a positive example). Negatives are constructed similarly for randomly chosen genes, and genes found to lack a TF binding site. Their best reported accuracy is 0.93; however this result is somewhat misleading since their analysis contains a total of only 175 positives. Their classification of a large negative set (1750 negatives) can skew the accuracy since large numbers of negatives are classified correctly. To put their result in perspective, their sensitivity is 55% and their positive predictive value is 63%. While their method shows promise, it still relies completely on the correlation of the transcription factor’s expression to its target. Thus, they are likely to miss interactions depending on cooperating TFs, or factors whose activation is dependent on
post-translational modification, nuclear exclusion, etc.

The approach by Beer et al. uses Bayesian networks to learn the combinatorial relationships of TFs and targets that underlie gene expression data\(^{15}\). Their method involves clustering gene expression data and using hierarchical Bayesian networks to predict the cluster assignment of a test gene based on the sequences in its promoter. They impose constraints which can be learned by the algorithm, allowing them to derive complex logical relationships from the data (e.g., motif A and motif B must both be present and within 20 base-pairs). Although this approach is innovative and can accurately describe the sequence/expression relationships of many genes, it may not be appropriate for our goals since it can be highly dependent on the clustering of the expression data, and the method by which motif discovery is performed on the genes being tested.

Our approach uses SVMs to associate TFs with targets by combining high dimensional heterogeneous data sets. SVMs have been applied successfully to many problems in computational biology. They have been used for the prediction of protein remote homology\(^ {16}\), secondary structure\(^ {17}\), protein sub-cellular localization\(^ {18}\), detection of normal and cancerous tissue samples\(^ {19}\), gene function\(^ {20}\), translation start sites\(^ {21}\), mRNA splice sites, and prediction of signal peptide cleavage sites\(^ {22}\). One notable attempt combines protein sequence similarity, protein-protein interactions, protein hydrophobicity, and gene expression to predict the functional group of a set of proteins\(^ {23}\).

**Background and Brief Review**

We now introduce our methodology and, briefly review for the convenience of the non-specialist some of the basic elements of support vector machines.

We train our SVM on 104 transcription factors (i.e. regulators) individually, using positive and negative training sets as explained below. Each gene in the positive set shares certain attributes, or features, that other genes do not, and it is on the basis of these differences that a classifier for a particular TF is obtained. We use 18 different genomic datasets to generate attributes, as indicated below, such as the number of occurrences of a particular sequence \(k\) long. For example, the number of occurrences of each of the 256 possible nucleotide sequences 4 long would be represented by a 256 component vector, each component of which would be the number of times the corresponding 4-mer occurs upstream from one of the genes in the set.

Thus the data for any particular TF consists of many vectors in a space with thousands of dimensions, each vector representing a gene in the training set, the point it identifies labeling the set of attributes for that gene.

An SVM separates the positive and negative sets by finding a hyperplane whose distance from the two parallel hyperplanes (on either side) passing through the closest data points is maximal. Better generalization (performance in prediction) is often obtained by forgoing perfect separation of training data and allowing some misclassification. This *soft margin* is found subject to the constraint that the distance to the closest cleanly separated data be maximal with some penalty for misclassifications, as explained below.

We denote the attribute vectors for the \(i^{th}\) gene in the training set by \((\mathbf{x}_i, y_i)\) where
\( y_i \) is +1 when \( x_i \) is an attribute vector from the positive set, and -1 otherwise. The vector \( x_i \) has the form \( x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots x_{id}) \) where \( d \) is the number of attributes, i.e. the dimensionality of the space.

The equation for a hyperplane has the form
\[
\mathbf{w} \cdot \mathbf{x} + b = 0
\]  
(1)
(see Figure 1) where the components of \( \mathbf{w} \equiv (w_1, w_2, \ldots, w_d) \) are the weights of the corresponding attributes and \( b \) is the distance from the origin to the closest point on the plane.

We first consider the case where the data are completely separable by a hyperplane. The problem is then to find the values of \( \mathbf{w} \) and \( b \) that give the maximum margin separating hyperplane, i.e., that which separates the two classes most widely. This requires use of only the closest correctly separated points, each representing the attributes of a gene. In a toy two dimensional example (Fig 1 below) the points are \( x_1, x_2 \) and \( x_3 \) and the separator bisects the distance between parallel planes through those points.

The margin is the distance between two planes parallel to the separator, one passing through the closest correctly classified positive data point, and the other passing through the closest correctly classified negative data point. The vector \( \mathbf{w} \) is scaled so that the hyperplanes through the closest data are given by \( 24, 25: \mathbf{w} \cdot \mathbf{x} + b = +1 \) and \( \mathbf{w} \cdot \mathbf{x} + b = -1 \). These two or more data vectors (on either side) are the support vectors. Equivalently the data satisfy the single constraint
\[
y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1
\]  
(2)
since \( y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \) is the distance from the separator to the \( i \)th data point.

The magnitude of the margin, the perpendicular distance between the two hyperplanes that include the support vectors, is given by
\[
m = \frac{2}{\| \mathbf{w} \|}
\]  
(3)
where \( \| \mathbf{w} \| = \sum w_i^2 \) is the magnitude of the weight vector. Eq (3) is readily obtained by noting that if \( \mathbf{x}^+ \) and \( \mathbf{x}^- \) denote the position vectors of two points at the intersection of an orthogonal to the separator with the margin hyperplanes (which can both be chosen to be parallel to \( \mathbf{w} \); see Fig 1) and taking without loss \( \| \mathbf{x}^+ \| > \| \mathbf{x}^- \| \)
\[
m = \| \mathbf{x}^+ - \mathbf{x}^- \| = \| \mathbf{x}^+ \| - \| \mathbf{x}^- \| \]
On the other hand from (2), since \( \mathbf{w} \) and \( \mathbf{x}^\pm \) have been chosen to be parallel, we have \( y_i(\| \mathbf{w} \| \cdot \| \mathbf{x}^\pm \| + b) = 1 \) (using \( \mathbf{x}^\pm \) with \( y_i = \pm 1 \), respectively), from which (3) follows from above using
\[
\| \mathbf{x}^+ \| = \frac{-b + 1}{\| \mathbf{w} \|} \quad \text{and} \quad \| \mathbf{x}^- \| = \frac{-b - 1}{\| \mathbf{w} \|}.
\]  
Thus the problem is to maximize \( m \) given by (3), subject to the set of constraints given by (2); note in the latter inequality, equality holds exactly for the support vectors, a sub-collection which we label \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s \), so that we can reduce the problem to involve this set of vectors only. The constrained maximization
can be solved using Lagrange multipliers\textsuperscript{24, 25}. In particular the problem is to minimize
\[ L = \frac{\|w\|^2}{2} - \sum_i \alpha_i [y_i (w \cdot x_i + b) - 1] \]  \hfill (4)

The weight vector is obtained by setting \( \frac{\partial L}{\partial w} = 0 \) (i.e., the system of equations \( \frac{\partial L}{\partial w_i} = 0 \)),

for the extremal \( x_i \), i.e., those for which equality holds in (2). These are exactly the support vectors, giving
\[ w = \sum_{i=1}^s \alpha_i y_i x_i \]  \hfill (5)

where in this example, the weight vector is \( w \equiv (w_1, w_2) \), the \( i \)-th attribute vector is \( x_i \equiv (x_{i1}, x_{i2}) \), \( d = 1, 2 \), and the number of support vectors, \( s \), is four (three lying on the margin planes and the one misclassified point).

The parameter \( b \) is determined as a weighted average of the distances to the two hyperplanes containing the support vectors,
\[ b = \frac{1}{s} \sum_{i=1}^s y_i - w \cdot x_i \]  \hfill (6)

(in fact in this fully separated case, for each support vector \( x_i, i = 1, \ldots, s \), we have by definition \( b = y_i - w \cdot x_i \)). The procedure for finding the multipliers is somewhat simplified by forming and then maximizing the so-called dual Lagrangian\textsuperscript{24, 25}:
\[ L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j. \]  \hfill (7)

This is obtained by substituting (5) into (4), and noting that \( \frac{\partial L}{\partial b} = 0 \) implies \( \sum_i \alpha_i y_i = 0 \).

We now consider the case of soft margins, in which perfect separation is not possible and a penalty \( \xi_i \) is paid in the Lagrangian for each misclassification of size \( \xi_i \).

The target function and constraints are modified so that the problem is to find
\[ \min_{w, \xi} \|w\|^2 + C \sum_{i=1}^r \xi_i \]
subject to \( \xi_i \geq 0 \) and \( y_i (w \cdot x_i + b) \geq 1 - \xi_i \) for all \( i = 1, \ldots, r \).

Here \( \xi_i \) is the distance of the \( i \)-th misclassified point \( x_i \) from its margin, which is defined as before to be the hyperplane \( (H_1 \text{ or } H_2) \) at a distance \( \frac{1}{\|w\|} \) in the direction of the correct classification of \( x_i \) from the separating hyperplane \( w \cdot x + b = 0 \).

\( C \) is the parameter mediating the tradeoff between maximal margin and misclassification and \( r \) is the number of misclassified points allowed. Essentially, the algorithm proceeds to find the maximum margin by minimizing \( \|w\| \), while balancing this against the amount
\[ \sum_{i=1}^{r} \xi_i \text{ of misclassification with this choice of margin.} \]  
We again use Lagrange multipliers \( \alpha_i \) as in the previous case to form a full Lagrangian, and then minimize it.  
To illustrate in our two dimensional example, we assume the data in Table 2 below.  
In this case we then have after minimizing the Lagrangian:

\[
w_1 = \sum_i \alpha_i y_i x_{i1} = (1.7731)(1)(7.7) + (3.3938)(-1)(6.5) + (3.3793)(-1)(6) = 1.3174
\]

\[
w_2 = \sum_i \alpha_i y_i x_{i2} = (1.7731)(1)(5) + (3.3938)(-1)(1.5) + (3.3793)(-1)(7) = 0.1198
\]

Thus \( w = [1.3174, 0.1198] \)

\[
b = \{1 - (1.3174)(6) - (0.1198)(7) + \ldots\} / 3 = -9.7425
\]

Thus \( \xi_4 = -2.3593 \)

**Figure 1 Anatomy of an SVM in two dimensions**

This is the classification plot for the data given in Table 1. Red points ( + ) indicate positive examples and blue points ( o ) are negatives. Coordinates \( d_1 \) and \( d_2 \) are the components of \( x \). The labeled points \( x_1, x_2, x_3, x_4 \) are the support vectors. The classifier is labeled as \( w \cdot x + b = 0 \) and the margin is labeled \( m \). One point, \( x_4 \), is misclassified. Since \( x_4 \) is in the positive set, its slack variable \( \xi_4 \) is the distance from the +1 margin line.
For linearly separable data (no misclassifications), we have $\xi_i = 0$, and we are in the first case, in which the values of $w$ and $b$ change to ensure that (3) is minimized subject to constraint (2). However, for data that is not linearly separable (e.g., with point $x_4$) $\alpha_i$ can become extremely large. In the Lagrangian formalization a constant $C$ becomes an upper bound for $\alpha_i$ (i.e., the constraint $0 \leq \alpha_i \leq C$ is used). By using $C$ as a bound for $\alpha_i$, we can limit the influence of single data points which cannot be classified correctly. Thus, in our example with $C = 5$, the multiplier for $x_4$ has value five (table 1).

Table 1 Data and parameters of the classifier

<table>
<thead>
<tr>
<th>examples</th>
<th>features</th>
<th>labels</th>
<th>Lagrange</th>
<th>calculate w vector</th>
<th>calculate b vector</th>
<th>Distance to hyperplane</th>
<th>Slack variables $\xi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>-1</td>
<td>3.3793</td>
<td>-20.2757</td>
<td>-23.656</td>
<td>-0.7425</td>
</tr>
<tr>
<td>2</td>
<td>7.7</td>
<td>5</td>
<td>1</td>
<td>1.7731</td>
<td>12.8620</td>
<td>8.8656</td>
<td>-0.7425</td>
</tr>
<tr>
<td>3</td>
<td>6.5</td>
<td>1.5</td>
<td>-1</td>
<td>3.3585</td>
<td>-22.0326</td>
<td>-5.8516</td>
<td>-0.7425</td>
</tr>
<tr>
<td>4</td>
<td>6.4</td>
<td>1</td>
<td>0</td>
<td>3.3585</td>
<td>30</td>
<td>30</td>
<td>-2.3653</td>
</tr>
<tr>
<td>5</td>
<td>9.3</td>
<td>1</td>
<td>0</td>
<td>-2.4731</td>
<td>0</td>
<td>0</td>
<td>3.7934</td>
</tr>
<tr>
<td>6</td>
<td>9.5</td>
<td>8.5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.2395</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.1916</td>
</tr>
<tr>
<td>8</td>
<td>9.9</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.1737</td>
</tr>
<tr>
<td>9</td>
<td>6.5</td>
<td>6.1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0959</td>
</tr>
<tr>
<td>10</td>
<td>7.6</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.4311</td>
</tr>
<tr>
<td>11</td>
<td>9.6</td>
<td>5.5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.4311</td>
</tr>
<tr>
<td>12</td>
<td>4.5</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.4541</td>
</tr>
<tr>
<td>13</td>
<td>2.8</td>
<td>4.5</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5.4602</td>
</tr>
<tr>
<td>14</td>
<td>5.7</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.3174</td>
</tr>
<tr>
<td>15</td>
<td>1.5</td>
<td>1.5</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7.5058</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5.7495</td>
</tr>
<tr>
<td>17</td>
<td>3.5</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.6922</td>
</tr>
<tr>
<td>18</td>
<td>3.6</td>
<td>8.5</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6.112</td>
</tr>
<tr>
<td>19</td>
<td>5.5</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2.5859</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.2755</td>
</tr>
</tbody>
</table>

It is evident from (7) that the first step in finding this maximal margin separator is to calculate all pairwise correlations between example vectors in the form of their inner (dot) products (also called the linear kernel function). Thus, given two data points $x_i$ and $x_j$, the kernel function $K(x_i, x_j) = x_i \cdot x_j$ is used to specify the inner products resulting in a complete kernel matrix $K_{ij} = K(x_i, x_j)$ involving every pair of data points. Since the data are represented as inner products rather than feature vectors it becomes possible to substitute different definitions of inner product for the above linear dot product. Several alternatives are given in table 2. These functions are inner products defined on feature spaces of different dimensionalities. Thus, defining a new inner product will implicitly map the data into a new feature space. This swapping of kernel functions to map data into different spaces is commonly referred to as the “kernel trick.” Biological features such as conservation or gene expression values can be correlated and may have complex, nonlinear relationships, highlighting the need for classification schemes which can accurately classify data that are not linearly separable.

Table 2. Common kernel functions

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Datasets
We have tested a variety of sequence and non-sequence based classifiers for predicting the association of TFs and genes. All together 18 separate data sources (each yielding a feature map and kernel) are combined to build classifiers for each transcription factor. The 18 data sources comprise a family of sequence-based methods (e.g., k-mer counts, TF motif conservation in multiple species, etc), expression data sets, phylogenetic profiles, and gene ontology (GO) functional profiles (see Table 3). For a more detailed description of data sets see (submitted for publication^26^).

Our positive and negative training sets are taken from ChIP-chip experiments^1^, 2, Transfac 6.0 Public^3^, and a list curated by Young et al. from which we have excluded indirect evidence such as sequence analysis and expression correlation (http://staffa.wi.mit.edu/cgi-bin/young_public/navframe.cgi?s=17&f=evidence). Only ChIP-chip interactions of p-value<=10^{-3} are considered, as recommended by the authors^1^.

The Transfac and curated list represent a manually annotated set which will later be used separately during SVM comparison to PSSM performance. For the purposes of SVM, however, all manually curated and high-throughput sets are grouped together, making a total of 9104 positive interactions.

Negative sets pose a greater challenge since no defined negatives exist in the literature; however, since a particular TF will regulate only a small fraction of the genome, a random choice of negatives seems acceptable. In fact, test cases with a few TFs show good classification performance with random negatives (unpublished work). Nevertheless, a safer set of negatives would be those showing no binding by experiment under some set of conditions. Along those lines, we have chosen for each TF 175 genes with the highest p-values (generally >0.8) under all conditions tested in genomic ChIP-chip analyses^1^, 2. Clearly all experimental conditions have not been sampled and this does not guarantee that our choices are truly never bound by the TF, but this choice of negatives will maximize our chances of selecting genes not regulated by the TF of interest.

Table 3 Dataset abbreviations and description

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MOT</td>
<td>Motif hits in S.cerevisiae</td>
</tr>
<tr>
<td>2 CONS</td>
<td>Motif hits conservation 18 organisms</td>
</tr>
<tr>
<td>Column 1</td>
<td>Column 2</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>PHY</td>
<td>Phylogenetic profile</td>
</tr>
<tr>
<td>EXP</td>
<td>Expression correlation</td>
</tr>
<tr>
<td>GO</td>
<td>GO term profile</td>
</tr>
<tr>
<td>KMER</td>
<td>K-mers – 4, 5, 6-mers</td>
</tr>
<tr>
<td>S1</td>
<td>Split 6-mer 1 gap _kkk _kkk</td>
</tr>
<tr>
<td>S2</td>
<td>Split 6-mer 2 gaps kkk _kkk</td>
</tr>
<tr>
<td>S3</td>
<td>Split 6-mer 3 gaps kkk ___kkk</td>
</tr>
<tr>
<td>S4</td>
<td>Split 6-mer 4 gaps kkk ____kkk</td>
</tr>
<tr>
<td>S5</td>
<td>Split 6-mer 5 gaps kkk _____kkk</td>
</tr>
<tr>
<td>S6</td>
<td>Split 6-mer 6 gaps kkk ______kkk</td>
</tr>
<tr>
<td>S7</td>
<td>Split 6-mer 7 gaps kkk ______kkk</td>
</tr>
<tr>
<td>S8</td>
<td>Split 6-mer 8 gaps kkk ________kkk</td>
</tr>
<tr>
<td>MM01</td>
<td>6-mer with 1 mismatch (count 0.1)</td>
</tr>
<tr>
<td>MM05</td>
<td>6-mer with 1 mismatch (count 0.5)</td>
</tr>
<tr>
<td>ENT</td>
<td>Condition specific TF-target correlation</td>
</tr>
<tr>
<td>SPAR</td>
<td>Nucleotide sparse binary encoding</td>
</tr>
</tbody>
</table>

All promoter sequences have been collected from RSA tools, Ensembl, or the Broad Institute’s Fungal Genome Anatomy Project. Sequences are then masked using the dust algorithm and the RepeatMasker software where appropriate to exclude low complexity sequences and known repeat DNA from further analysis. PSSM scans (for datasets 1 and 2, below) are performed with the MotifScanner algorithm. MotifScanner assumes a sequence model where regulatory elements are distributed within a noisy background sequence. The algorithm requires input of a background sequence model, which in this case is a transition matrix of a 3rd order Markov model generated from the masked upstream regions of each genome. MotifScanner only requires one parameter be set by the user; viz the threshold score for accepting a motif as a binding site. Several thresholds have been tested and the results we have used to create SVM kernels are all at a setting of 0.15, which has been found to be a reasonable middle ground, making approximately 560 predictions per TF. Settings beyond 0.2 produce too many false hits to be useful. The PSSMs themselves are obtained from Transfac 6.0 Public and from33, which are a mix of experimentally derived motifs and those generated by motif-discovery procedures.

Also datasets using k-mers rather than PSSMs are generated using the fasta2matrix program which delineates all possible k-mers and counts the occurrence of each within a set of promoters. Gapped k-mers are detected using custom scripts written as Matlab m-files.

The expression data used include 1011 microarray experiments compiled by Ihmels and co-workers, and can be downloaded with permission from the authors.

As mentioned above 18 different data kernels are used to construct a classifier for each transcription factor. The datasets fall into several distinct groups. All classifier construction and validation was performed in Matlab using the SPIDER machine learning library.
Methods

At first, each type of genomic data has been evaluated independently on each transcription factor. Several kernel functions are tested and parameters are optimized by a grid-selection technique. Each data set is normalized so that all attributes describing the data have mean zero and standard deviation one. The exceptions to this are the Gene Ontology, phylogenetic profile, and TF-target correlation data, which are not normalized since their data is binary.

A schematic representation of our method is shown in figure 4. Briefly, for a particular TF, four classifiers are produced for each type of genomic data, each from a different kernel function (linear, RBF, Gaussian, and polynomial). In order to make an appropriate choice of the \( C \) parameter a grid-selection technique is used to evaluate a range of choices. In the case of two parameter selections (e.g., when choosing the degree of the polynomial kernel), all possible combinations of parameter values within the pre-specified range are tested. A five-fold cross validation is used to choose the best parameters based on ROC score.

Once parameters are chosen for each kernel type, the parameter-optimized classifiers are tested using a leave-one-out cross validation procedure. Now, for each type of genomic data, there are four classifiers for a particular TF (one for each of the kernel functions). Of these four, we select the one with the best performance as measured by the \( F_1 \) statistic. Several common statistics, including accuracy, sensitivity, and specificity, can overstate the performance of a classifier depending on the relative size of the positive and negative training sets. The \( F_1 \) statistic is a more robust measure which in principle represents a harmonic mean between sensitivity (S) and positive predictive-value (PPV):

\[
F_1 = \frac{2 \times S \times PPV}{S + PPV} = \frac{2 \times TP}{2 \times TP + FP + FN}
\]

Each TF will now have only one classifier for each type of genomic data (18 classifiers in all). Before weighting and combining kernels, each kernel matrix is normalized according to

\[
\tilde{K}(x, y) = \frac{K(x, y)}{\sqrt{K(x, x)K(y, y)}}.
\]

This normalization adjusts all points to lie on a unit hypersphere in the feature space. This assures that no single kernel has matrix values that are comparatively larger or smaller than other kernels which would bias the combination.

Using a scheme with weights equal to the \( F_1 \) of each classifier, the underlying 18 kernels are scaled and added into one unified kernel for the transcription factor. This kernel represents the integration of all types of genomic data. Three simple weighting schemes were compared. In all cases the primary weight for a method is determined by computing its \( F_1 \) score ratio with that of the best performing method. Our first weighting scheme simply multiplies all kernel matrices by their primary weights (i.e., \( F_1 \) ratios ) and sums them. A second scheme squares the primary weights before multiplying. Our third scheme is the most nonlinear, taking the squared tangent of the primary weight.
PSSM Comparison

Using the same positive and negative sets as for the SVM procedure, PSSMs have been used to make predictions across the yeast genome at various score thresholds to serve as a comparison to predictions made by SVM. The data in figure 6 represent only one threshold, a value of 0.1 as the prior parameter in MotifScanner (low parameter values retain the best matches whereas values near 1 allow very loose hits). Other choices of threshold do not appear to improve performance. Loosening the threshold begins to dramatically increase false positive predictions beyond a prior 0.2. By making detection more strict false predictions are reduced along with sensitivity. Since the matrices for the 104 transcription factors are partly experimentally determined and partly computationally generated, the Transfac PSSMs for 17 TFs are then evaluated to determine whether the experimental matrices by themselves outperform SVM for target identification. Finally, since a large number of positive targets have been taken from high-throughput ChIP-chip experiments, the Transfac PSSMs are tested again on only the portion of the positives obtained from manually annotated sources.

Results and Discussion

Using the classification procedure described above, we have been able to accurately...
classify the known targets of many transcription factors in *S. cerevisiae*. Overall, the best single method achieves a sensitivity of 71% and a positive predictive value of 0.82. Our results have also been compared to a classifier generated from a random dataset. Many methods perform well, but the best classification is made with k-mer counts allowing 1-mismatch per k-mer. Our results show that combining datasets increases sensitivity incrementally and delivers a surprising leap in positive predictive value. This indicates that combining methods has the useful effect of reducing false positive classifications (submitted\textsuperscript{26}).

These results are significant since to our knowledge no other group has had such success classifying or validating results which include data from high-throughput experiments like ChIP-chip. This has led many to consider that as many as 50\%\textsuperscript{38} to 60\%\textsuperscript{39} of the targets produced by ChIP-chip are not biologically functional. Our ability to correctly classify large amounts of high-throughput data indicate that there is clearly a signal identifying ChIP-chip positives from other genes. It must be kept in mind, however, that for many classifiers the number of positives and negatives are almost equal, meaning that a random classifier would be expected to achieve \textasciitilde50\% sensitivity, as we observe with our control. In future work we will more closely compare our results to random classifiers for all TFs, and apply a post-processing method which will produce a probabilistic classification output rather than a simple binary result\textsuperscript{40}. This will allow us to attach a confidence to any prediction in the form of a class conditional probability (e.g., $P(\text{truly bound} \mid \text{classified as } 1)$).

In other work we search for evidence of such a signal in various individual datasets and extract the attributes which contribute most to a transcription factor’s classifier. The $\mathbf{w}$ vector described above can be used in this way to identify the features, in any particular data set, which are most important to classification. Features having large $\mathbf{w}$ components correspond to dimensions in the feature space where positives and negatives are more definitively separated. Thus by examining a single dataset, for example k-mer counts, it is possible to determine the k-mer(s) most responsible for the differences between positives and negatives. Our results on the k-mer count dataset has shown that the many k-mers having large $\mathbf{w}$ values are in fact elements of the known transcription factor binding site as taken from SGD (submitted\textsuperscript{26}).

To better judge the performance of SVM classifiers it is useful to compare them to standard PSSM scans for their ability to identify targets. We have found that support vector classification performs better than a simple weight matrix scan. Figure 5 shows such a comparison as a function of sensitivity, specificity, positive predictive value, and the $F_1$ statistic. The far left grouping of data uses only the Transfac PSSMs for 17 TFs on just the manually curated positives (with same negatives as all other analyses) from Transfac and literature sources. The second grouping uses the same Transfac PSSMs as grouping 1, but this time with the same high throughput positive sets used in the SVM classification. The third grouping is a result from scans using PSSMs for all 104 TFs against the positive and negative sets on which the SVMs were trained. Finally, the far right grouping restates the performance of the SVMs with 18 combined datasets on the full set of positives. The SVM classifiers consistently outperform PSSMs, even when the matrices are from a curated set such as Transfac. Although the PSSMs perform well, they suffer from a large
number of false positive predictions. Figure 5 shows data for only one threshold of PSSM
scan, but altering the threshold does not make PSSMs more competitive with SVM (see
Methods section). Support vector machine classifiers offer a reasonable balance between
sensitivity and false prediction.

Aside from the work shown here, we have also applied the combined classifier for
each TF to all promoters in the yeast genome in order to expand the binding repertoire of
each factor (submitted\textsuperscript{26}). This results in predictions of new regulatory roles for some TFs
and new identification of regulatory structures such as feed-forward loops in metabolic
pathways.

![Figure 5 SVM vs PSSM scan.](image)

Far left: Transfac PSSMs for 17 TFs scanned against manually curated positives (same
negatives). Second group: Transfac PSSMs for 17 TFs scanned against their positive
and negative sets. Third group: PSSMs for 104 TFs scanned against positive and
negative sets. Far right: Average results for SVM classifiers trained on weighted
combination of 18 datasets.
In conclusion, support vector machines can accurately classify and predict transcription factor binding sites using a wide range of genomic data types. Combining various information sources reduces false positives and increases sensitivity. Based on k-mer data, SVMs appear to be isolating appropriate features for classification. Finally, the flexibility of this approach allows easy inclusion of new types of genomic data. Our future work involves development of sophisticated dimension reduction techniques to discover biologically significant features in different datasets based on classifier performance. Also, new datasets can be included which leverage information about DNA structural features. Information of this type could consist of promoter melting temperature profiles, bend and curve features of promoters, or DNA accessibility predictions based on patterns of hydroxyl radical cleavage. Furthermore, it might be possible to capture more meaningful information from k-mer counts by instead measuring the likelihood that a certain k-mer occurs by chance in a gene’s promoter, thus attaching a p-value to all k-mers in each promoter region. Support vector machines show promise as a means to analyze regulatory relationships and will be increasingly useful for the analysis of higher mammalian genomes as more genomic data becomes available.

References
10. Zhu, Z., Pilpel, Y. & Church, G. Computational Identification of Transcription


30. Tatusov, R.L. & Lipman, D.J.
31. Smit, A. & Green, P. Repeatmasker.