Predicting an Aptamer's Target Binding Affinity Using its Nucleotide Sequence

Andrew Naber
naber@stanford.edu

June 6, 2016

1 Introduction

Aptamers and their Selection

Modern molecular and cellular biology depends on markers that can very selectively bind to a target molecule. Unfortunately, the process of finding such a marker is generally slow, through trial and error, and often *in vivo*. Aptamers (short nucleotide sequences) are a promising type of marker that can be generated and evaluated synthetically. Currently, the process of aptamer selection involves the generation of several thousand different nucleotide sequences and the measurement of their target affinity using fluorescence. Those aptamers with high affinity will fluoresce brightly. Through successively stronger washes, those aptamers that do not bind the target tightly are removed and the batch is refined down to those aptamers that very tightly bind the target. New technologies such as microfluidic chips and DNA arrays are now enabling millions of aptamers to be generated and evaluated in a single day, but it is still costly to run each batch, and only a small number of the possible aptamers of a fixed length can be evaluated.[1] A single batch generates large amounts of information about the target binding characteristics of many different aptamers. Using this information, it could be possible to automatically classify proposed aptamers as tightly or weakly binding and to create a model of the underlying function mapping the aptamer's nucleotide sequence to the its target affinity.

Project Goals

This project investigates how support vector machines (SVM) can be used to classify proposed aptamers after being trained on the fluorescence data from a previous batch. This will hopefully reduce the number of batches required to converge to a good aptamer by focusing resources on those aptamers with the most promising nucleotide motifs. Different types of string kernels for the SVM (*e.g.*, spectrum and mismatch kernels) will be built and evaluated. Further, by viewing the function mapping aptamers to fluorescences as a Gaussian process, the upper confidence bound reinforcement learning algorithm will be used to develop a model that can be used to suggest high-performing aptamers for future batches. Using the data from a single batch of aptamer selection, these algorithms could ideally be used to suggest an additional fifty or so aptamers that are likely to outperform any of the aptamers in the actual training set.

2 Methods

The dataset consists of approximately 3,000 nucleotide sequences of length 45 along with their associated fluorescences. The starting and stopping subsequences common to all of the aptamer nucleotide sequences were not used in any computation.

2.1 String Kernels

For many learning algorithms involving strings, kernels provide a computationally efficient means of finding the inner product of high-dimensional feature vectors without explicitly forming the feature vectors and taking their inner product. There are many possible kernels that could be used to measure the similarity of two aptamers. Some are based only on the underlying nucleotide sequence (*e.g.*, spectrum or mismatch kernels), but others are based on the structural properties of the folded aptamer in solution. For this project, all kernels were based upon the underlying nucleotide sequence of the aptamer.

2.1.1 Spectrum Kernel

The spectrum kernel as described in [2] is a commonly used kernel to compare two nucleotide strings. It has been used successfully to identify evolutionarily related genes that are distant on a chromosome. Consider the set $\mathcal{X}$ of all finite-length strings of characters from the alphabet $\mathcal{A} = \{\texttt{A}, \texttt{T}, \texttt{C}, \texttt{G}\}$. For $k \geq 1$, the k-spectrum feature vector of an $x \in \mathcal{X}$ is the set of all the k-length contiguous subsequences that it contains. This feature vector is indexed by all possible subsequences a of length k from $\mathcal{A}$. Concisely, $\Phi_k : \mathcal{X} \to \mathbb{R}^{4^k}$ with $\Phi_k(x) = (\phi_a(x))_{a \in \mathcal{A}}$, where $\phi_a(x)$ is the number of times that subsequence a occurs in x. Then, the k-spectrum kernel is defined as $K_k(x_1, x_2) = \Phi_k(x_1)^T \Phi_k(x_2)$. Using tree structures, this can be calculated directly in linear time in the length of the strings.

2.1.2 Mismatch Kernels

The mismatch kernels differ from the spectrum kernels by incorporating spatial information. The simplest example is the 1-mismatch kernel which simply compares each nucleotide in a string with the nucleotide at the same position in the other string. It returns the total number of such matches. For $k > 1$, there are several variations of this basic idea which instead compare substrings of length k. The sliding k-mismatch kernels use a sliding window of length k starting at each position within the string. The chunk k-mismatch kernels break the strings into non-overlapping substrings of length k and return the total number of these that match. Because of the structural and chemical similarities between `A` and `T` and those between `G` and `C`, I hypothesized that mismatches of `A` and `T` should be penalized less than those of `A` and `G` or `A` and `C` and similarly for the other nucleotides. So, for this project, I also created modified mismatch kernels return 0.1 for a mismatch between either of the above pairs rather than 0. Finally, the weighted mismatch kernel consisted of a sum of chunk k-mismatch kernels with weights of $1/k$ for $k = 1, ..., 5$. Note that the definitions of mismatch kernels given here differ from those in the literature.[3]

2.2 Support Vector Machines

An SVM learns a hyperplane from training data which can be used in binary classification. Using the kernel trick allows the SVM to learn and use this hyperplane in the feature vector space without explicitly computing feature vectors. Employing this kernel trick with the kernels described above, several SVM were trained. The dataset was divided into training (60%) and test (40%) subsets. The training data were then labeled using an arbitrary cutoff so that aptamers with fluorescences in the top 40% of the dataset were labeled as $+1$ (good) and those that were not were labeled as -1 (poor). The optimization was carried out using stochastic gradient descent. The logarithm of the fluorescence was used in these computations.

2.3 Gaussian Process Optimization using Upper Confidence Bound

For the Gaussian process upper confidence bound (GP-UCB) algorithm it is assumed that the underlying function f on the space of strings that determines the associated fluorescence is a sample from a Gaussian process with mean function $\mu(x) = \mathbb{E}f(x)$ and covariance function $k(x_1, x_2)$. For each step t in the algorithm, a new point $x_t \in \mathcal{X}$ that maximizes $\mu_{t-1}(x) + \sqrt{\beta_t}\sigma_{t-1}(x)$ is selected and then a noisy measurement $y_t = f(x_t) + \epsilon_t$ where ϵ_t is distributed $\mathcal{N}(0, \sigma^2)$ is collected. Then, including all the noisy samples up to iteration t', we calculated the updated

$$\mu_{t'}(x) = K^{(t')}(x)^T(K + \sigma^2 I)^{-1}y, \tag{1}$$

and

$$\sigma^2_{t'}(x) = k(x, x) - K^{(t')}(x)^T(K + \sigma^2 I)^{-1}K^{(t')}(x), \tag{2}$$

where K is the kernel matrix over all $x_1, \ldots, x_{t'}$ and $y = [y_1, \ldots, y_{t'}]^T$. The parameter β_t decreases with t and guarantees bounds on the regret.[4][5] This algorithm was implemented using the weighted mismatch kernel, as described above, and it was allowed to run for 200 iterations. The logarithm of the fluorescence was used in these computations.

3 Results

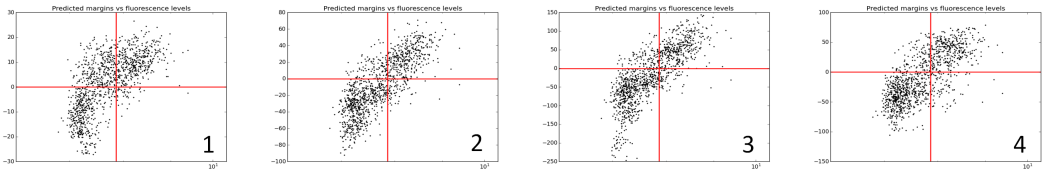

Figure 1: Predicted margins vs. fluorescence for four different kernels. See the following table for the labels. The vertical red line indicates the fluorescence cutoff below which aptamers are labeled as poor, and the horizontal red line indicates the zero margin below which aptamers were predicted to be poor. That is, dots in the upper left quadrant are misclassified as good aptamers, while dots in the lower right quadrant are misclassified as poor aptamers.

Label	Kernel	Training Accuracy	Test Accuracy
1	3-Spectrum	76.5%	71.9%
2	3-Mismatch (Chunk)	89.0%	84.5%
3	3-Mismatch (Sliding)	89.3%	85.5%
4	Modified 3-Mismatch (Sliding)	81.3%	83.8%

Figure 2: Training and test accuracy for the SVMs in figure 1.

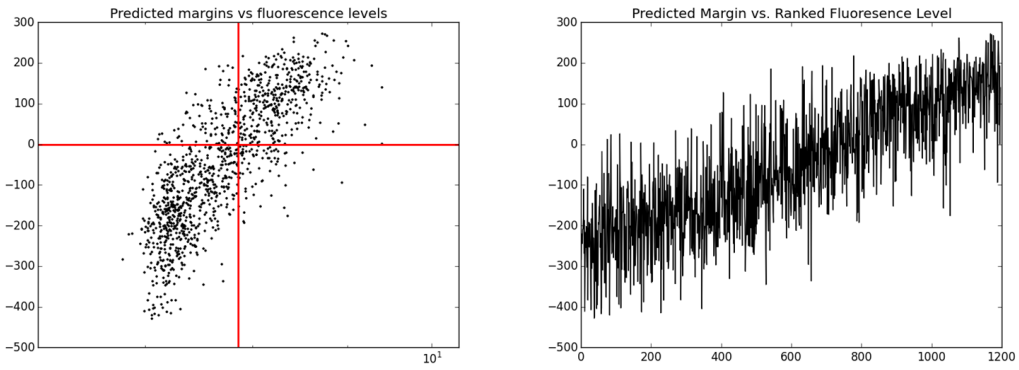

Figure 3: The left graph shows the predicted margins vs. fluorescence (similar to those in figure 1), and the right graph shows the predicted margins vs. ranked aptamer for the weighted mismatch kernel. This kernel performed the best with a training accuracy of 89.4% and a test accuracy of 85.6%.

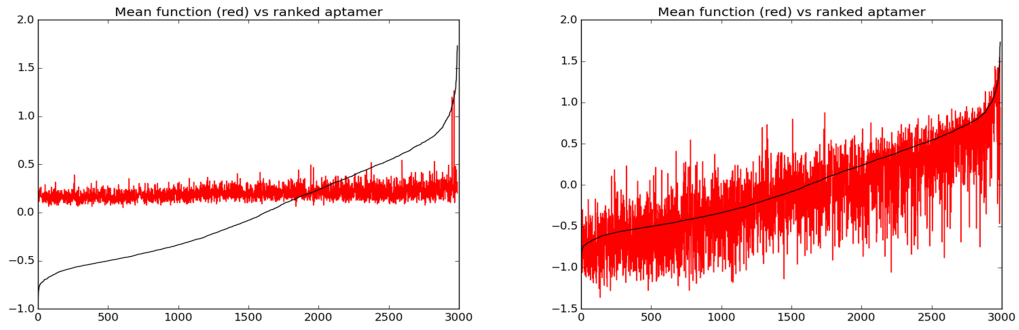

Figure 4: The left graph shows the actual zero-mean fluorescences in black and the expected mean function in red against the ranked aptamer following one iteration of the GP-UCB algorithm. The right graph shows the same following two hundred iterations.

4 Analysis

Support Vector Machines Five support vector machines were successfully trained and tested. The weighted mismatch kernel performed the best with training and test accuracies of 89.4% and 85.6%, respectively. However, the 3-mismatch (sliding) and 3-mismatch (chunk) performed nearly as well and were faster to compute. The modified 3-mismatch (sliding) did not perform as well as the unmodified mismatch kernels. This indicates that binding of the aptamers is highly dependent on the actual nucleotide at each position and not necessarily the cross-nucleotide binding characteristics of the nucleotide at each position. All mismatch kernels were significantly better than the spectrum kernel. There is an important explanation for this. The spectrum kernel completely disregards spatial information included in the nucleotide sequence. This information determines how the aptamer loops on itself in its secondary structure, which strongly determines the tertiary folding of the aptamer in solution. Because it is this tertiary structure that actually fits into the target molecule, it makes sense that the spatial information is important in predicting how tightly an aptamer binds the target. These results suggest that a more accurate kernel would weight different locations within the string depending on how strongly changing the nucleotide at that location changes the performance of the aptamer. It would be expected that certain locations within each string are very sensitive to changes, *e.g.* locations which are involved in closing the loops of the secondary structure. This weighting on the string could come from either prior physical knowledge of aptamer secondary folding or from hyper-parameter fitting.

GP Optimization The Gaussian process optimization using the upper confidence bound algorithm was successfully implemented on the entire dataset. As can be seen in figure 4, the mean function does converge to the actual underlying fluorescence after approximately two hundred iterations. After tuning the parameter β_t, the regret was minimized and the optimal aptamer was typically found following one hundred iterations. The performance of the UCB algorithm was highly dependent on the quality of the kernel used; using the spectrum kernel usually resulted in non-convergence or convergence to a local maximum far from the global maximum. These results suggest that it may be possible to use the model of the fluorescence function of the nucleotide sequences as a Gaussian process and then use it as a means of suggesting aptamers that could have higher fluorescences that those in the dataset. In future work, Monte Carlo search techniques combined with the Gaussian process model will be used to suggest new aptamers.

5 References and Acknowledgment

Acknowledgment This project was conducted under the guidance of Dr. John Duchi for EE391.

[1] Cho et al. (2013) Quantitative selection and parallel characterization of aptamers. *Proc Natl Acad Sci USA* 110(46):18460-18465.

[2] Leslie et al. (2002) The spectrum kernel: a string kernel for SVM protein classification. *Proc Pacific Symposium Biocomputing* 564-575.

[3] C. Leslie et al. (2002) Mismatch string kernels for SVM protein classification. *Neural Information Processing Systems.*

[4] Romero et al. (2012) Navigating the protein fitness landscape with Gaussian processes. *Proc Natl Acad Sci USA* Online:E193-E201.

[5] Srinivas et al. (2010) Gaussian process optimization in the bandit setting: no regret and experimental design. ICML.