Active Learning For DBSI: DNA Binding Site Identifier

Max Horowitz-Gelb

December 12, 2016

Abstract

DBSI is a structure based method for predicting the positions of protein interaction sites associated with DNA binding [5]. Here I present a method for applying active learning to the training of DBSI. This method optimizes the training of DBSI in a way that considers the batch style form of labelled data collection necessary when creating a training set. This method shows slight improvements in efficiency in comparison to naive methods.

Introduction

For area under an ROC curve, DBSI has been shown to achieve 88%, a high degree of separability. The score was achieved by training DBSI on a set of 263 unique proteins. The model as a result of this training is now accessible to anyone on a public server [3]. The quality of this model could be improved further by training with more labelled data. But to collect more data practically we need to address practical issues.

Necessity for Active Learning

Unlike learning on data generated from sensors or internet activity, aquiring labelled data for DBSI requires considerable time and energy. Experiments must be done to probe each protein for interaction sites.

Find out how protein complexes are aquired to emphasize lab work.

Neccessity for batch style active learning

Due to the way training data for DBSI is aquired, standard active learning methods are not appropriate. This is because each training point given to DBSI corresponds to one residue in a protein. Therefore each protein that is probed will give hundreds of new training points. Because of this we must query training points in batches as opposed to querying individual residues. This puts us in a unique situation where the batch with the most informative set training points as a whole may be different than the batch containing the one individually most informative training point. To address this issue I use a active learning querying method which scores batches rather than individual points in order to select the next query protein.

Methods

DBSI uses a support vector machine to classify residues. Such machine learning model has geometric properties that make applying active learning quite intuitive.

Non Batch-Style Active Learning

Let us consider a standard binary classifier SVM as DBSI uses. Then we have a set of n training examples $\{(x_1, y_1), ..., (x_n, y_n)\} \subset (X \times \{-1, 1\})^n$ where X denotes our input space.

as shown in [1], our decision function can be described by its training examples,

$$g(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$$

and classifcation is simply the sign,

$$f(x) = sign(g(x))$$

Where x_i is the set of support vectors, x is our vector we want to classify, α_i selects our support vectors, and k is our kernel function.

As shown in [2], if our kernel k satisfies Mercer's condition, then there exists a feature space F and a map ϕ from X to F, such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$, and we may rewrite our above equation as

$$f(x) = sign(\langle w_{svm}, \phi(x) \rangle)$$
$$w_{svm} = \sum_{i=1}^{n} \alpha_i \phi(x_i)$$

where w_{svm} is the normal vector of our hyperplane in feature space.

If we assume our training data is linearly separable in F then there is exists a non-empty set

$$V = \{w \in F | y_i \langle w, \phi(x_i) \rangle > 0 \text{ for } i = 1, ..., n \text{ and } \|w\| = 1$$

which we call our version space [4].

Now let $Q \in X^m$ be our query pool that our active learner has access to and $l : X \to \{-1, 1\}$ be a function giving the correct label of an instance of our query pool.

Then at each step our active learner removes one instance x_{n+1} from Q and then adds it to our training set resulting in a new training set

$$\{(x_1, y_1), ..., (x_n, y_n)\} \cup \{(x_{n+1}, l(x_{n+1})\}$$

For an efficient learning we want an active learner who most rapidly decays the cardinality of V as it adds individual examples to our training set. As shown in [2] we can guarantee a high decay in $|V|$ by querying the instance closest to the hyperplane in feature space. So at each active learning query step we query

$$x_{n+1} = \arg\min_{x' \in Q} |g(x')|$$

For active learning in a setting where we query one instance at a time, this works fairly well, but this is not the case in batch style active learning.

Batch-Style Active Learning

Batch-style active learning is very similar to standard active learning except now we remove multiple instances from the query pool at the same time. So for a batch size of p we remove a query batch $B \in X^p | B \subset Q$.

Intuitively one might extrapolate from the single instance case of active learning and select B to be the p instances in Q closest to the hyperplane. As shown in [2], this turns out to be a naive approach since the points may be close and contain redundant information. A better solution is one that not only considers distance from the hyperplane but also diversity of the points. Diversity in this case is measured using the angle between the induced hyperplanes of two instances. The induced hyperplane h of an instance x is the hyperplane whose normal vector is orthogonal to $\phi(x)$. From [2] we can calculate angles between the induced hyperplanes of two instances x_1 and x_2 as

$$|cos(\angle(h_1, h_2))| = \frac{|k(x_1, x_2)|}{\sqrt{k(x_1, x_1)k(x_2, x_2)}}$$

Then we can score each instance in Q for its redundancy to in comparison to all other instances as

$$r(x) = \max_{x' \in Q, x' \neq x} \frac{|k(x, x')|}{\sqrt{k(x, x)k(x', x')}}$$

Then for each instance x in the query pool we give it a score

$$s(x) = \lambda |g(x)| + (1 - \lambda) r(x)$$

where λ is a weight parameter between 0 and 1 which controls how much priority we give to each sub-score.

Then for as described in [2], we set our query batch B equal to the p instances in Q with lowest scores or

$$B = \underset{b \in X^p, b \subset Q}{\arg\min} \sum_{x \in b} s(x)$$

Active Learning Used for DBSI

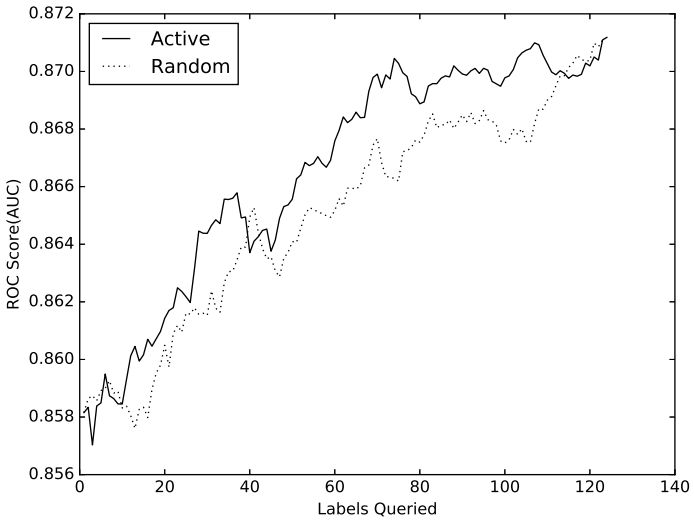
This method is slightly incompatible with DBSI since in [2] the method allows the active learner to create a query batch of any of the instances in the query pool. In the setting of DBSI, the batches are already predefined. Each instance is a member of a batch corresponding to the protein from which the instance came from. This is not a huge incompatibility. It only means that now the batches are pre-set and the active learner must score these batches instead of the individual instances. Each batch is given a score

$$s_{batch}(B) = \sum_{x \in I(B)} \lambda * (1 - |g(x)|) + (1 - \lambda) * (1 - r(x))$$

where $I(B)$ is the set of instances $\{x \in B : |g(x)| < 1\}$. Then at each query step we simply select the predefined batch B that maximizes $s_{batch}(B)$.

[2]

Results



Conclusion

References

[1] Scholkopf B and Smola AJ. Learning with kernels: Support vector machines, regularization, optimization, and beyond. 2002.

[2] Brinker K. Incorporating diversity in active learning with support vector machines. In *In Proceedings of the 20th International Conference on Machine Learning*, pages 59–66. AAAI Press, 2003.

[3] Sukamar S, Zhu X, Ericksen SS, and Mitchell JC. DBSI server: DNA binding site identifier. *Bioinformatics*, 2016.

[4] Mitchel TM. Generalization as search. *Artifical Intelligence*, 18:203–226, 1982.

[5] Zhu X, Ericksen E., and Mitchell JC. DBSI: DNA-binding site identifier. *Nucleic Acids Research*, 2013.