Writeup for Challange 1

CIHC Team

September 20, 2010

Preparation

We decided to use a number of different algorithms, and then blend the results to obtain the final classifier. In order to do this, the training data was split into two equal parts – odd rows formed a training set for our classifiers and even rows became a test set. The blender was trained on this set, the predictions of individual classifiers being its attributes.

Classifiers

Text algorithms

We classified each of the amino acids into one of the following 7 groups – small aliphatic [AGLIV], acid [DENQ], alkaline [KHR], hydrophilic [ST], aromatic [FYW], proline [P] and sulfuric [MC]. Amino acids belonging to the same group were not distinguished be the algorithms. More technically, each peptide sequence was encoded using just 7 letters (instead of the original 21).

k-NN

We considered normalised Levenshtein distance of two peptide sequences as a reasonable estimate of their similarity. Assuming this, we used k-NN. To make a prediction, 100 closest (in the sense of Levenshtein distance) sequences of the training set were found. Their average binding energy was the output of the classifier.

n-gram spectra

For a 7-letter alphabet, there are $7^3 = 343$ possible trigrams. Thus each sequence was represented as a 343-dimensional logical vector, where each entry indicated whether the respective trigram is contained in the sequence or not.

The Bernoulli event model was then used. We assume that a random peptide sequence is generated by first deciding if it will be binding or not (with prob. $p(B) = 1 - p(N)$), and then independently choosing the trigrams contained in it, where a trigram T is chosen with probability $p(T|B)$ if the sequence is binding and $p(T|N)$ otherwise. A sequence S is thought of as a set of trigrams T it contains.

For each sequence S, the output of the classifier was

$$p(B|S) = \frac{p(S|B)p(B)}{p(S)} = \frac{\left(\prod_{T\in S} p(T|B)\right)\cdot p(B)}{\left(\prod_{T\in S} p(T|B)\right)\cdot p(B) + \left(\prod_{T\in S} p(T|N)\right)\cdot p(N)}.$$

$p(B)$, $p(T|B)$ and $p(T|N)$ were fitted to the training set by taking the fraction of binding sequences and the fraction of the sequences containing T among the (not)binding sequences.

We used the same representation to run the Random Forest classifier.

For sake of completeness, an SVM with a text kernel was also run. It used 4-grams.

Profile-based approach

We chose 19 amino acid indices from the AAindex database which seemed important to our problem. These were:

1. Polar requirement
2. Normalized frequency of chain reversal S
3. Steric parameter
4. The number of atoms in the side chain labelled 3+1
5. Distance between C-alpha and centroid of side chain
6. Side chain torsion angle phi(AAAR)
7. Free energies of transfer of AcWl-X-LL peptides from bilayer interface to water
8. Hydropathy scale based on self-information values in the two-state model
9. Accessible surface area
10. Principal component III
11. Normalized positional residue frequency at helix termini C2
12. Avarage membrane preference: AMP07
13. Transfer free energy from oct to water
14. Normalized van der Waals volume
15. Isoelectric point
16. Polarity
17. Notmalized frequency of turn
18. Normalized frequency of alpha-helix
19. Free energy of solution in water, kcal/mole

All values were linearly rescaled to $[0, 1]$.

Given an amino acid index, we sought short patterns in positive training examples with respect to this index.

Let $v \in [0, 1]^n$ be a vector created by replacing each amino acid in a sequence with its value. Define $\sigma_i^{(3)}(v) = (v_{i-1}, v_i, v_{i+1})$, $\alpha_i^{(3)}(v) = (v_{i-3}, v_i, v_{i+3})$, $\beta_i^{(3)}(v) = (v_{i-2}, v_i, v_{i+2})$, for all sensible values of i. Now, for $p \in [0, 1]^3$, $v \in [0, 1]^n$, define

$$d_\sigma(v, p) = \min_i \|\sigma_i^{(3)}(v) - p\|^2,$$

and d_α, d_β analogously. We tried to find vectors p which were informative in the following sense.

Given p, find k vectors in the training set which are d_ζ-closest to p, $\zeta \in \{\sigma, \alpha, \beta\}$. The fraction of positive sequences among these k indicates how good p is. We set $k = 100$, and picked H best vectors for each amino acid index and each type of distance (α, β or σ). This gave a total of $57H$ vectors. Each of these vectors was made into an attribute of a classifier by calculating its distance from all the training examples. We repeated the procedure for $H \in \{5, 10, 20\}$, using then both SVM and random forest as a classifier, for both classification (binding/not binding) and regression (logarithm of reactivity). These classifiers hardly differed in performance and had both true positive rate and true negative rate $\approx 77\%$. However, we decided to use all 12 classifiers for blending.

Smith-Waterman algorithm

We calculated similarity scores of local sequence alignment using Smith-Waterman algoritm with three different substitution matrices: BLOSUM80, BLOSUM62 and BLOSUM45. Penalties were: 11 for opening a gap, 1 for extending it. For each sequence we found its maximum similarity score to positive sequences in our training set. These maximum similarities, for each of the substitution matrices, formed another 3 attributes of the blender.

Clustering

We calculated similarity scores of random sequences (permuted positive training sequences) using different BLOSUM matrices. For each probability value: 0.001, 0.01 and 0.1 we computed threshold, so that for each sequence meanly 0.001, 0.01, or 0.1 of scores were bigger than threshold. Two sequences with similarity score bigger than threshold were considered as neighbours.

Positive sequences from our training set were clustered. We took sequence with the biggest number of neighbours and made a cluster from that sequence and its neighbours. Then we removed these sequences from further clustering. This procedure was repeated until there were no clusters bigger than 2 elements.

Maximum similarity scores to clustered sequences were found for each probability value and substitution matrix, and formed another 9 attributes of the blender.

Blending

Random Forest algorithm was used as a blender, with 2000 voting trees. The proportion of votes was used as an estimate of the decision confidence.