Technical notes

Sequence transformation into a vector

Recall that each sequence is represented as a vector $x \in \Re^d$. Let G denote protein sequence space, than we define a function $\psi : G \rightarrow \Re^d$ which performs this transformation ($d = 545$). We further define $\psi(\{g_1, .., g_n\}) = \{\psi(g_1), .., \psi(g_n)\}$ where $\forall i = 1..n : g_i \in G$.

Training set construction

We begin with a set of all 924 proteins as 'ionic channel inhibitors'. Redundancy is removed so that no two proteins share more than 90% identity, leaving a set A of 604 proteins. To construct the training sets, we apply the following procedure:

for i=1 to 10 {
  randomly select a set T_1 of proteins from UniProt ($|T_1| = 2|A|$)
  $S_i := T_1$
  randomly select a set T_2 of proteins from UniProt ($|T_2| = 2|A|$)
  for each protein $a \in A$ {
    randomly remove two proteins $t_1, t_2 \in T_2$
    shorten* t_1, t_2 so that $length(t_1) = length(t_2) = length(a)$
    $S_i := S_i \cup \{t_1, t_2\}$
  }
  randomly select a set T_3 of proteins from UniProt ($|T_3| = 2|A|$)
  for each protein $a \in A$ {
    randomly remove two proteins $t_1, t_2 \in T_3$
    shorten** t_1, t_2 so that $length(t_1) = length(t_2) = length(a)$
    $S_i := S_i \cup \{t_1, t_2\}$
  }
  remove redundancy (90% identity) from S_i
}

* - Shorten sequences by trimming their tails (c-terminals), i.e. if $length(a) = l$, choose the subsequence $[1, l]$.
** - Shorten sequences by trimming at a random point i, i.e. if $length(a) = l$, choose the subsequence $[i, i + l]$.

The *training sets* $L_1, .., L_{10}$ are defined as:

$$\forall i = 1..10 : L_i = (\psi(S_i) \times \{-1\}) \cup (\psi(A) \times \{1\}).$$

Classifier construction

A *decision-stump* is a function $f : \Re^d \rightarrow \{1, -1\}$ of the form:

$$f_{i,j}(x) = \begin{cases} 1 : x_i > j \\ -1 : \text{ otherwise} \end{cases}$$

or vice-versa. Training a decision-stump classifier f on a training set L is defined as finding the decision-stump $f_{i,j}$ that maximizes $\sum_{(x,y)\in L} y f_{i,j}(x)$.
A *boosted stumps classifier* is a function $c : \Re^d \rightarrow \Re$ of the form $c(x) = \sum_{n=1}^{k} \alpha_n f_{i_n,j_n}(x)$ where f are decision-stump classifiers and $\alpha_n \in \Re$. c itself is constructed by applying the *AdaBoost*[1] algorithm to the training set (using the decision-stump as the weak learner), producing α_n and f_{i_n,j_n} as output.
Since *AdaBoost* requires a parameter input (the number of boosting rounds), we have developed the following parameter-tuning framework:

randomly divide the training set L into 3 equal subsets: P_1, P_2 and P_3
for $r \in$ {possible parameter values}{
  for i=1 to 3{
    train a boosted stumps classifier c on $L \backslash P_i$ with parameter value r
    use the classifier c to predict on P_i
    remember predictions in Q
  }
  calculate AUC (area under ROC curve) of Q
  if AUC is the highest yet, $best_r := r$
}
Train a boosted stumps classifier on L with parameter value $best_r$

We train 10 boosted stumps classifiers, $c_1, .., c_{10}$, one for each training set $L_1, .., L_{10}$.

The final classifier is defined as $\phi : \Re^d \rightarrow \Re$ where $\phi(x) = \frac{\sum_{n=1}^{10} c_n(x)/\beta_n}{10}$ where β_n is the normalization factor. The calculation of β_n is simply $\beta_n = \max_{x:(x,y)\in L_n} c_n(x)$. $\phi(x)$ is called the *(mean) prediction score of* x. We also define the *standard deviation of the score of* x as $sd(x) = \sqrt{\frac{\sum_{n=1}^{10}(c_n(x)/\beta_n)^2}{10} - \phi(x)^2}$.

AUC calculation

Given m positive instances and n negative instances, a classifier C outputs prediction for each instance. Let $x_1, .., x_m \in \Re$ and $y_1, .., y_n \in \Re$ be the predictions of C on the positive and negative instances, respectively. We define $ind(x)$ to be the indicator function (if x is true than $ind(x) = 1$, otherwise $ind(x) = 0$). The AUC of C on the instances is defined as:

$$AUC = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} ind(x_i > y_j)}{mn}$$

Note that this value is equal to the Wilcoxon Mann Whitney statistic.

References

[1] Freund Y. and Schapire, R. E. (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**: 119-139.