Machine learning methods for protein analyses

William Stafford Noble
Department of Genome Sciences
Department of Computer Science and Engineering
University of Washington
Outline

• Remote homology detection from protein sequences
• Identifying proteins from tandem mass spectra
  – Simple probability model
  – Direct optimization approach
Large-scale learning to detect remote evolutionary relationships among proteins

Iain Melvin  
Jason Weston  
Christina Leslie
History

• Smith-Waterman (1981)
  – Optimal pairwise local alignment via dynamic programming
• BLAST (1990)
  – Heuristic approximation of Smith-Waterman
• PSI-BLAST (1997)
  – Iterative local search using profiles
• Rankprop (2004)
  – Diffusion over a network of protein similarities
• HHSearch (2005)
  – Pairwise alignment of profile hidden Markov models
Supervised semantic indexing

• Data: 1.8 million Wikipedia documents
• Goal: given a query, rank linked documents above unlinked documents
• Training labels: linked versus unlinked pairs
• Method: ranking SVM (essentially)
  – Margin ranking loss function
  – Low rank embedding
  – Highly scalable optimizer

(Bai et al., ECIR 2009)
Key idea

• Learn an embedding of proteins into a low-dimensional space such that homologous proteins are close to one another.

• Retrieve homologs of a query protein by retrieving nearby proteins in the learned space.

This method requires
• A feature representation
• A training signal
• An algorithm to learn the embedding
Protein similarity network

- Compute all-vs-all PSI-BLAST similarity network.
- Store all E-values (no threshold).
- Convert E-values to weights via transfer function (weight = $e^{-E/\sigma}$).
- Normalize edges leading into a node to sum to 1.
Sparse feature representation

\[ \Phi(p') = (E(p', p_1), \ldots, E(p', p_\ell)) \]

\[ W(p', p_i) = \exp(-S_j(i)/\sigma) \]

\[ E(p', p_i) = \frac{W_{p'p_i}}{\sum_j W_{p'p_j}} \]

Probability that a random walk on the protein similarity network moves from protein \( p' \) to \( p_i \).

Query protein

Target protein

PSI-BLAST / HHSearch

E-value for query j, target i

Hyperparameter
Training signal

• Use PSI-BLAST or HHSearch as the teacher.

• Training examples consist of protein pairs.

• A pair \((q,p)\) is positive if and only if query \(q\) retrieves target \(p\) with E-value < 0.01.

• The online training procedure randomly samples from all possible pairs.
Learning an embedding

- Goal: learn an embedding
  \[ g(p) = W \Phi(p) \]
  where $W$ is an $n$-by-$\ell$ matrix, resulting in an $n$-dimensional embedding.

- Rank the database with respect to $q$ using
  \[ f(q, p_i) = \| g(q) - g(p_i) \|_1 = \| W \Phi(q) - W \Phi(p_i) \|_1 \]
  where small values are more highly ranked.

- Choose $W$ such that for any tuple
  \[ f(q, p^+) < f(q, p^-) \]
Learning an embedding

Good

Bad

• Minimize the margin ranking loss with respect to tuples \((q, p^+, p^-)\):

\[
\sum_{(q, p^+, p^-) \in \mathcal{R}} \max(0, 1 - f(q, p^-) + f(q, p^+))
\]

Negative examples should be further from the query than positive examples by a margin of at least 1.

Good

Bad

Training procedure

• Minimize the margin ranking loss with respect to tuples \((q, p^+, p^-)\):

\[
\sum_{(q,p^+,p^-) \in \mathcal{R}} \max(0, 1 - f(q, p^-) + f(q, p^+))
\]

• Update rules:

\[
\begin{align*}
\text{if} \ 1 - f(q, p^-) + f(q, p^+) & > 0 \\
W & \leftarrow W - \lambda \ \text{sign}(W \Phi(q) - W \Phi(p^-)) \Phi(q)^\top, \\
W & \leftarrow W + \lambda \ \text{sign}(W \Phi(q) - W \Phi(p^-)) \Phi(p^-)^\top, \\
W & \leftarrow W + \lambda \ \text{sign}(W \Phi(q) - W \Phi(p^+)) \Phi(q)^\top, \\
W & \leftarrow W - \lambda \ \text{sign}(W \Phi(q) - W \Phi(p^+)) \Phi(p^+)^\top,
\end{align*}
\]
Remote homology detection

- **Semi-supervised setting**: initial feature vectors are derived from a large set of unlabeled proteins.
- **Performance metric**: area under the ROC curve up to the 1\textsuperscript{st} or 50\textsuperscript{th} false positive, averaged over queries.
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>$\text{ROC}_1$</th>
<th>$\text{ROC}_{50}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI-BLAST</td>
<td>0.624</td>
<td>0.632</td>
</tr>
<tr>
<td>Rankprop</td>
<td>0.647</td>
<td>0.707</td>
</tr>
<tr>
<td>Protembed PSI-BLAST</td>
<td>0.689</td>
<td>0.739</td>
</tr>
<tr>
<td>HHHPred</td>
<td>0.771</td>
<td>0.836</td>
</tr>
<tr>
<td>Protembed HHHPred</td>
<td>0.777</td>
<td>0.853</td>
</tr>
</tbody>
</table>

Results are averaged over 100 queries.
Key idea #2

• Protein structure is more informative for homology detection than sequence, but is only available for a subset of the data.

• Use multi-task learning to include structural information when it is available.
Structure-based labels

• Use the Structural Classification of Proteins to derive labels

\[ y_i \in \{1, \ldots, C\} \]

• Introduce a centroid \( c_i \) for each SCOP category (fold, superfamily).

• Keep proteins in category \( i \) close to \( c_i \):

\[ f(p_i, c_{y_i}) < f(p_j, c_{y_i}), \quad \forall j : y_j \neq y_i \]
Structure-based ranks

- Use a structure-based similarity algorithm (MAMMOTH) to introduce additional rank constraints.
- Divide proteins into positive and negative with respect to a query by thresholding on the MAMMOTH E-value.

\[ f(q, p^+) < f(q, p^-) \]
<table>
<thead>
<tr>
<th>Method</th>
<th>$\text{ROC}_1$</th>
<th>$\text{ROC}_{50}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI-BLAST</td>
<td>0.624</td>
<td>0.632</td>
</tr>
<tr>
<td>Rankprop</td>
<td>0.647</td>
<td>0.707</td>
</tr>
<tr>
<td>Prottembed PSI-BLAST</td>
<td>0.689</td>
<td>0.739</td>
</tr>
<tr>
<td>Prottembed PSI-BLAST+SCOP</td>
<td>0.852</td>
<td>0.918</td>
</tr>
<tr>
<td>Prottembed PSI-BLAST+MAMMOTH</td>
<td>0.744</td>
<td>0.844</td>
</tr>
<tr>
<td>HHPredd</td>
<td>0.771</td>
<td>0.836</td>
</tr>
<tr>
<td>Prottembed HHPredd</td>
<td>0.777</td>
<td>0.853</td>
</tr>
<tr>
<td>Prottembed HHPredd+MAMMOTH</td>
<td>0.822</td>
<td>0.923</td>
</tr>
<tr>
<td>Prottembed HHPredd+SCOP</td>
<td>0.881</td>
<td>0.949</td>
</tr>
</tbody>
</table>
Protembed scores are well calibrated across queries.
Conclusions

- Supervised semantic indexing projects proteins into a low-dimensional space where nearby proteins are homologs.
- The method bootstraps from unlabeled data and a training signal.
- The method can easily incorporate structural information as additional constraints, via multi-task learning.
Calculation of exact protein posterior probabilities for identifying proteins from shotgun mass spectrometry data

Oliver Serang

Michael MacCoss
The protein ID problem

<table>
<thead>
<tr>
<th>Proteins</th>
<th>Peptides</th>
<th>Spectra</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EEAMPFk</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>CYCYGGLGK</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>CYCLLIGK</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>FTEILYCDLNR</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>VNILLGLPK</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>WGNEVNPILR</td>
<td>0.97</td>
</tr>
</tbody>
</table>
The protein ID problem

Input:
- Bipartite, many-to-many graph linking proteins to peptide-spectrum matches (PSMs)
- Posterior probability associated with each PSM.

Output:
- List of proteins, ranked by probability.
Existing methods

• ProteinProphet (2003)
  – Heuristic, EM-like algorithm
  – Most widely used tool for this task

• MSBayes (2008):
  – Probability model
  – Hundreds of parameters
  – Sampling procedure to estimate posteriors
Key idea

• Use a simple probability model with few parameters.
• Employ graph manipulations to make the computation tractable.
Three parameters

• The probability $\alpha$ that a peptide will be emitted by the protein.
• The probability $\beta$ that the peptide will be emitted by the noise model.
• The prior probability $\gamma$ that a protein is present in the sample.
Assumptions

2. Conditional independence of spectra given peptides.
3. Emission of a peptide associated with a present protein.
4. Creation of a peptide from the noise model.
5. Prior belief that a protein is present in the sample.
6. Independence of prior belief between proteins.
The probability model

\[ L(R^{(i)} = r^{(i)} | D) = \sum \prod \frac{\Pr(E_\epsilon^{(i)} = e_\epsilon^{(i)} | D_{\delta(\epsilon)}, Q)}{\Pr(E_\epsilon^{(i)} = e^{(i)}, Q)} \Pr(E_\epsilon^{(i)} = e_\epsilon^{(i)} | R^{(i)} = r^{(i)}) \]

- \( R \) = the set of present proteins
- \( D \) = the set of observed spectra
- \( E \) = the set of present peptides
- \( Q \) = peptide prior probability

**Computational challenge**: Exactly computing posterior probabilities requires enumerating the power set of all possible sets of proteins.
Speedup #1: Partitioning

- Identify connected components in the input graph.
- Compute probabilities separately for each component.
Speedup #2: Clustering

- Collapse proteins with the same connectivity into a super-node.
- Do not distinguish between "absent/present" versus "present/absent."
- Reduce state space from $2^n$ to $n$. 

**Proteins**

{1, 2}

**PSMs**

- 1.0
- 0.7

- 3
- 0.93

- 4
- 0.3

- 5
- 0.0

- 0.97
Speedup #3: Pruning

• Split zero-probability proteins in two.
• This allows the creation of two smaller connected components.
• When necessary, prune more aggressively.
## Effects of speedups

<table>
<thead>
<tr>
<th></th>
<th>H. Influenzae</th>
<th>Yeast</th>
<th>ISB 18</th>
<th>Sigma 49</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSMs</td>
<td>29,123</td>
<td>10,390</td>
<td>21,166</td>
<td>23,694</td>
</tr>
<tr>
<td>Proteins</td>
<td>32,748</td>
<td>3742</td>
<td>1777</td>
<td>392</td>
</tr>
<tr>
<td>Edges</td>
<td>60,844</td>
<td>12,202</td>
<td>21,720</td>
<td>24,392</td>
</tr>
<tr>
<td>Full problem</td>
<td>33,000</td>
<td>3700</td>
<td>1800</td>
<td>390</td>
</tr>
<tr>
<td>After partitioning</td>
<td>930</td>
<td>74</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>After clustering</td>
<td>170</td>
<td>47</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>After pruning</td>
<td>60</td>
<td>47</td>
<td>15</td>
<td>11</td>
</tr>
</tbody>
</table>

Numbers in the lower half of the table represent the log$_2$ of the size of problem.
Number of false positives

(A) *H. influenzae*

(B) Yeast

(C) ISB 18

(D) Sigma 49
Robustness to parameter choice

- Results from all ISB 18 data sets.
- Parameters selected using the *H. influenzae* data set.
Conclusions

• We provide a simple probability model and a method to efficiently compute exact protein posteriors.
• The model performs as well or slightly better than the state of the art.
Direct maximization of protein identifications from tandem mass spectra

Marina Spivak

Jason Weston

Michael MacCoss
The protein ID problem

Proteins | Peptides | Spectra
--- | --- | ---
EEAMPFK | 1.0 | ![Spectra](image)
CYCYGGLGK | 0.7 | ![Spectra](image)
CYCLLIGK | 0.93 | ![Spectra](image)
FTEILYCDLNR | 0.06 | ![Spectra](image)
VNILLGLPK | 0.3 | ![Spectra](image)
WGNEVNPILR | 0.97 | ![Spectra](image)
Key ideas

Previous methods:
• First compute a single probability per PSM, then do protein-level inference.
• First control error at peptide level, then at the protein level.

Our approach:
• Perform a single joint inference, using a rich feature representation.
• Directly minimize the protein-level error rate.
Features representing each PSM

- Cross-correlation between observed and theoretical spectra (XCorr)
- Fractional difference between 1\textsuperscript{st} and 2\textsuperscript{nd} XCorr.
- Fractional difference between 1\textsuperscript{st} and 5\textsuperscript{th} XCorr.
- Preliminary score for spectrum versus predicted fragment ion values (Sp)
- Natural log of the rank of the Sp score.
- The observed mass of the peptide.
- The difference between the observed and theoretical mass.
- The absolute value of the previous feature.
- The fraction of matched b- and y-ions.
- The log of the number of database peptides within the specified mass range.
- Boolean: Is the peptide preceded by an enzymatic (tryptic) site?
- Boolean: Does the peptide have an enzymatic (tryptic) C-terminus?
- Number of missed internal enzymatic (tryptic) sites.
- The length of the matched peptide, in residues.
- Three Boolean features representing the charge state.
PSM scoring

Input units: 17 PSM features

\[ f(E, s) = \sum_{i=1}^{\mathcal{HU}} w_i^O h_i(\phi(E, s)) + b, \]

\[ h_k(\phi(E, s)) = \tanh((w_k^H)^\top \phi(E, s) + b_k) \]
The Barista model

\[ F(R) = \frac{1}{|N(R)|^\alpha} \sum_{E \in N'(R)} g(E) \]

\[ g(E) = \max_{s: (E, s) \in M} f(\phi(E, s)) \]

Number of peptides in protein R

Proteins

Peptides

Spectra

Neural network score function
Model Training

repeat
    Pick a random protein \((R_i, y_i)\)
    Compute \(F(R_i)\)
    if \((1 – yF(R_i)) > 0\) then
        Make a gradient step to optimize \(L(F(R_i),y_i)\)
    end if
until convergence

• Search against a database containing real (target) and shuffled (decoy) proteins.
• For each protein, the label \(y \in \{+1, -1\}\) indicates whether it is a target or decoy.
• Hinge loss function: \(L(F(R),y) = \max(0, 1-yF(R))\)
• Goal: Choose parameters \(W\) such that \(F(R) > 0\) if \(y = 1\), \(F(R) < 0\) if \(y = -1\).
Target/decoy evaluation

- ProteinProphet
- Barista
- IDPicker
(A) Yeast trypsin

(B) Yeast elastase

(C) Yeast chymotrypsin

(D) Worm trypsin

- **ProteinProphet**
- **Barista**
- **IDPicker**
External gold standard

![Graph showing confirmed positive proteins against positive proteins with lines for ProteinProphet, Barista, and IDPicker markers.](Image)
PeptideProphet probability

Unmatched peptide

Proteins identified only by Barista

Proteins identified only by ProteinProphet

PeptideProphet probability
One-hit wonder

VEFLGGLDAIFGK

m/z  
0  10  20  30  40  50  60  70  80  90  100
Relative Abundance

y2  y3  y4  y5  y6  y7  y8  y9  y10  y11  y12
b2  b3  b4  b5  b6  b7  b8  b9  b10  b11  b12
Multi-task results

- At the peptide level, multi-tasking improves relative to either single-task optimization.
- At the protein level, multi-tasking improves only relative to peptide level optimization.
Conclusions

• Barista solves the protein identification in a single, direct optimization.
• Barista takes into account weak matches and normalizes for the total number of peptides in the protein.
• Multi-task learning allows for the simultaneous optimization of peptide- and protein-level rankings.
Take-home messages

• Generative models and discriminative, direct optimization techniques are both valuable.

• Developing application-specific algorithms often provides better results than using out-of-the-box algorithms.
Machine Learning in Computational Biology workshop

MLCB

- Affiliated with NIPS
- Whistler, BC, Canada
- December 11-12, 2009
- Unpublished or recently published work.
- 6-page abstracts due September 27.

http://www.mlcb.org