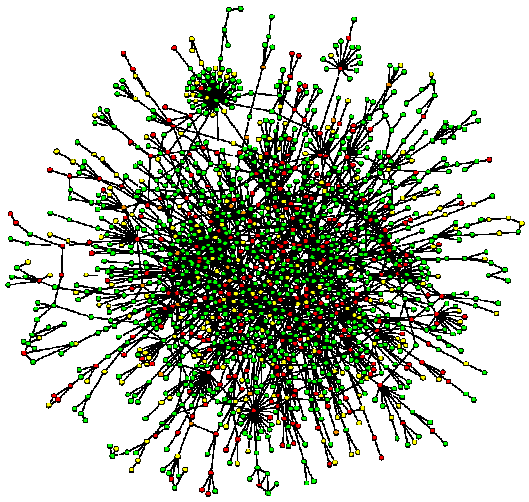
Structured prediction using the network perceptron

Ta-tsen Soong

Joint work with Stuart Andrews and Prof. Tony Jebara

Motivation



Protein-protein interaction network

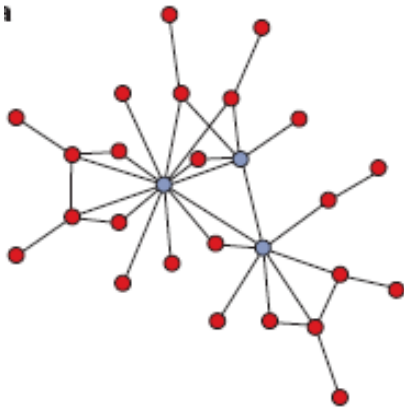
- A lot of network-structured data
  - Social networks
  - Citation networks
  - Biological networks
- Important to discover the network structure y given the node attributes x
  - Biological network structure = how cellular processes are regulated
- The network structure has dependency within y

  ➔ not i.i.d

  ➔ structured prediction

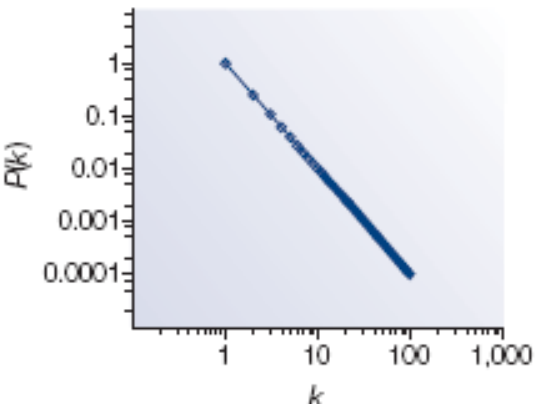
Properties of Networks

- What properties make edges in the network inter-dependent??



- Degree distribution
  - Power-law $\Pr[d=k] \sim k^{-\alpha}$

    $\alpha \sim 3$
  - many nodes with few neighbors
  - Few nodes with lots of neighbors



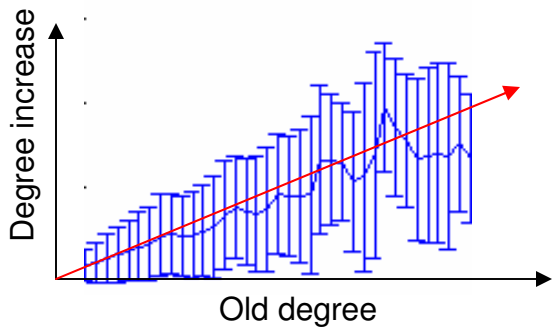
Degree (connectivity)

Barabasi & Oltvai. 2004. Nature Review: Genetics

Properties of Networks

- Network growth
  - Preferential attachment:
    New nodes prefer to connect to nodes that already have many links.

$$\Pr[\text{attach to } i] = \frac{d_i}{\sum_j d_j}$$



  - Infer incomplete networks

- Clustering coefficients, motifs, modules, diameter of network, etc.

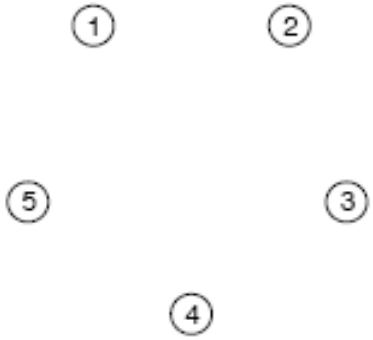
Barabasi & Oltvai. 2004. Nature Review: Genetics

Our Goals

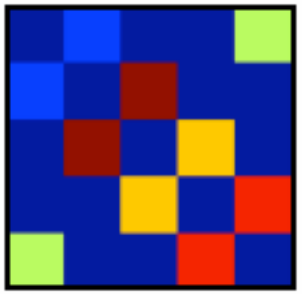
- Focus on protein-protein interaction network
- Given a set of nodes (proteins), find the network structure (edges) that indicates which proteins interact with which other proteins.
- Non-I.I.D. structured learning
- Network degree constraints
- Kernelization to take advantage of high-dimensional features
- Transductive learning

Overview

Protein 1



$$\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$$
$$\mathbf{x}_k \in \mathbb{R}^d$$



Similarity matrix S

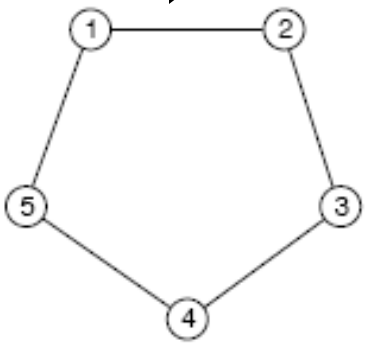
Maximum-weight b-matching problem

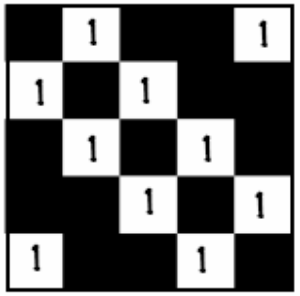
$$\max_{\mathbf{y} \in \mathcal{B}} \sum_{j,k} y_{j,k} s_{j,k} = \max_{\mathbf{y} \in \mathcal{B}} \mathbf{y}^T \mathbf{s}$$

Degree constraints

$$\sum_j y_{j,k} = b_k \quad \forall k$$

Interaction between proteins 1 and 2





Adjacency matrix Y

$$\mathbf{y} = (y_{j,k})$$
$$y_{j,k} \in \{0, 1\}$$

Learning the similarity

- From features x_i x_j to similarity S_{ij}
  - Correlation, Euclidean distance, etc.

- Want to learn a good transformation from x to S, such that the output Y is most compatible with the true network structure

Y= $\max_{\mathbf{y}\in\mathcal{B}} \sum_{j,k} y_{j,k} s_{j,k} = \max_{\mathbf{y}\in\mathcal{B}} \mathbf{y}^T \mathbf{s}$

s.t. degree constraints

$$\sum_j y_{j,k} = b_k \quad \forall k$$

$$s_{j,k} = w^T f_{j,k}$$

Perceptron learning

Algorithms

i.i.d perceptron

Input: $\{(x_i\ y_i),\ i=\ 1,\ \ldots\ ,\ n\}$,

$x \in R^n,\ y \in \{-1,+1\}$

$w^{(0)}=0$

for t=1 to max_iteration

for i=1 to num_examples

if sign($\left\langle w^{(t)}, x_i \right\rangle$) $\neq y_i$

$w^{(t+1)} = w^{(t)} + x_i y_i$

end if

end i

end t

network perceptron*

Input: $X = [x_{11}, x_{12}, ..., x_{ij}, ..., x_{nn}]$

$Y = [y_{11}, y_{12}, ...y_{ij}, ..., y_{nn}]$

$w^{(0)}=0$

for t=1 to max_iteration

$y^{(t)} = \underbrace{\arg\max}_{Y} (w^T X)Y$

s.t. Y satisfies degree constraints

(lower and upper bounds)

for all i, j

if $y^{(t)}_{ij} \neq y^{\text{true}}_{ij}$

$w^{(t+1)} = w^{(t)} + x_{ij} y_{ij}$

end if

end i,j

end t

*Algorithm can use transductive learning and be kernelized. Details not shown

Experiments

- Network data:
  - DIP database*
  - Protein-protein interaction network
  - 2,312 proteins
  - 5,299 interactions (y)
    - interactions labeled as +1
    - unseen protein pairs labeled as -1
- Features (x):
  - Microarray data:
    - 349 experiments,
    - 2,312 x 349 matrix
  - Used ICA / PCA to reduce dimensionality

*Xenarios I, et al. (2000) DIP: The Database of Interacting Proteins. Nucleic Acid Research

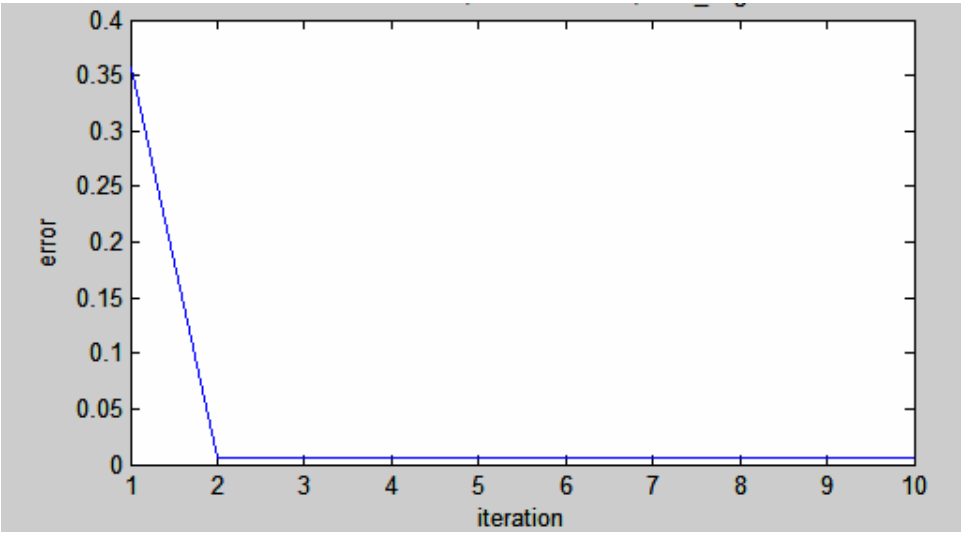
Results

- Exact degree constraints
- Non-exact degree constraints
- Effect of features
- Kernel diagonal to deal with unseparable data

Exact degree constraints

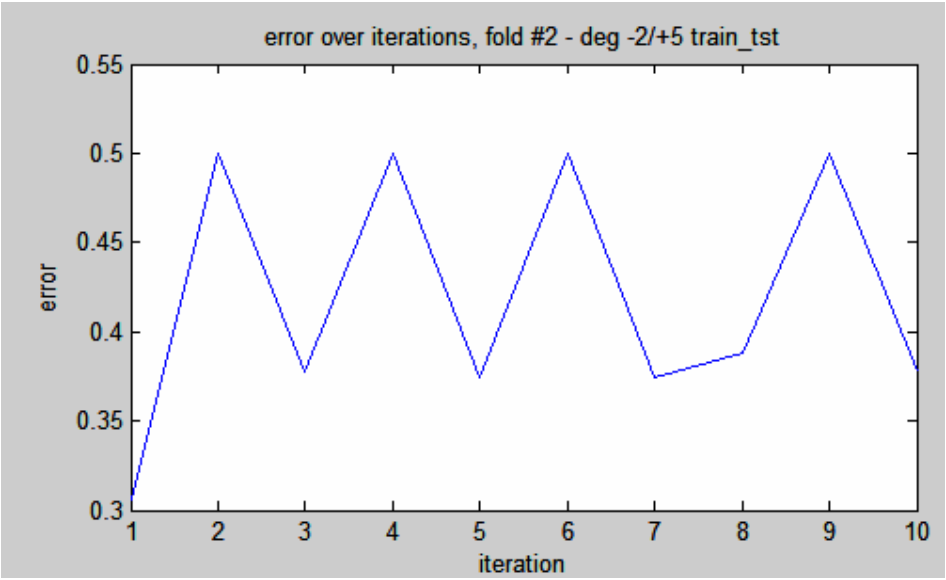
- 1,000 positives
- 1,000 negatives
- 5-fold cross-validation

- Can reach low error very quickly due to the tight degree constraints



Looser degree constraints

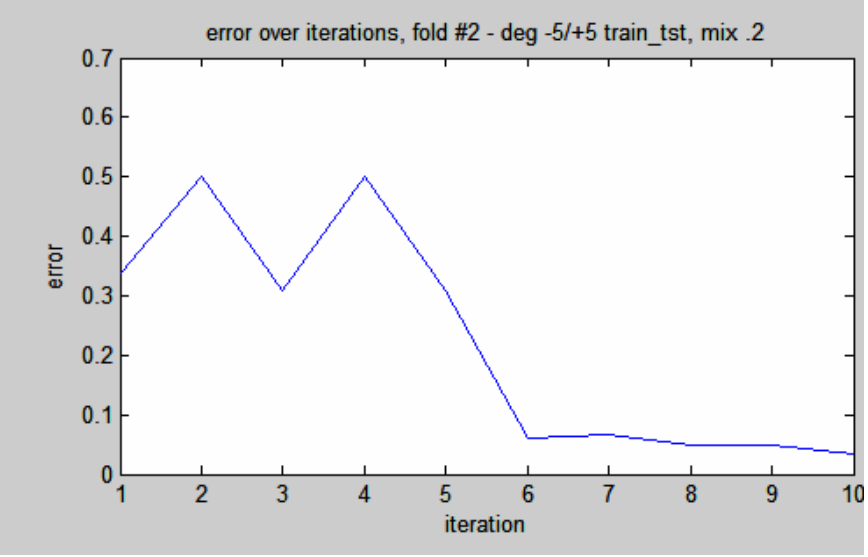
- More realistic in real-world problems.
- Ex. true degree -2/+5
- 1,000 positives
- 1,000 negatives
- Better than random. But the perceptron oscillates… Why??



Linear kernel

Features

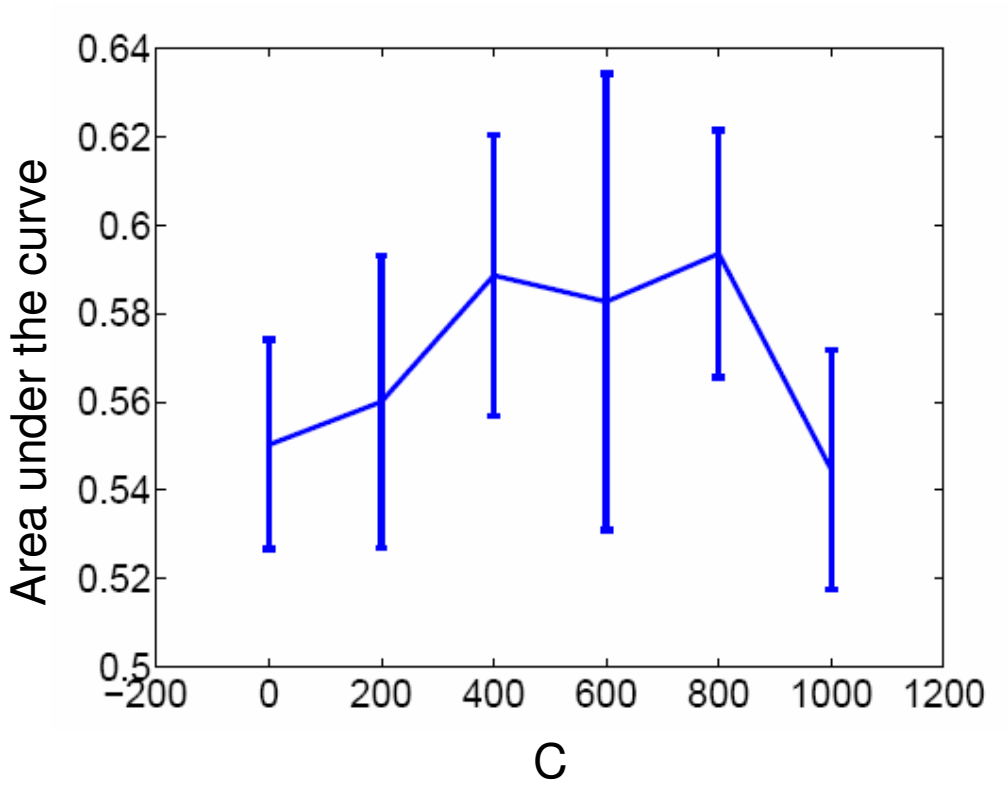
- Are microarray data good features?
- Try to add separable features to data to see if algorithm works
- Add two separable random Gaussian bumps of the same dimensions to the original positive and negative features respectively.



Algorithm works for separable data.
➔ Microarray features might not be that good

Kernel diagonal

- Adding a constant C to the kernel diagonal to deal with unseparable (microarray) data.
- Helps the algorithm converge.
- Cross-validate to find the C that gives the best performance



Future work

- Add more meaningful features.
  - Ex. function annotation, sub-cellular localization
- Add degree bound estimation
- Add topological constraints
- Memory-efficient kernel computation
- Take into account how close each node's degree is to the expected degree.