Integration of Multiple Networks for Robust Label Propagation

Tsuyoshi Kato
Univ of Tokyo, Japan
(currently, Ochanomizu univ)

Hisashi Kashima
TRL IBM, Japan

Masashi Sugiyama
Tokyo Tech, Japan

Classification from single dataset

Predict protein functions from gene expression data.

- SVM approach
  - Kernel function
  - Kernel matrix
  - SVM
  - Scores

- Gene expressions

Label Propagation approach

- Pick k-NN
- k-NN graph
- Label Propagation
- Scores

(Zhu et al., 2003)
What is label propagation

Input
- Adjacency matrix $A \in \mathbb{R}^{n \times n}$
- Class labels $y \in \{\pm 1\}^{n}$

Output
- Scores $f \in \mathbb{R}^{n}$

$n$: # of nodes

Classify the nodes with $f_i > \text{threshold}$ as positive nodes, and vice versa.

Label propagation is an algorithm for prediction of class labels of nodes.

Classification from multiple datasets

Predict protein functions from multiple datasets.

(Lanckriet et al. 2004)

An extension of SVM to use multiple kernel matrices
Classification from multiple datasets

Predict protein functions from multiple datasets.

Using TSS algorithm (Tsuda et al., 2005) A generalization of label propagation.

Efficient
Does not assume the existence of networks irrelevant to the task

We propose an alternative algorithm which works in the same setting.

Graph and class labels

Adjacency matrix

\[ A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
\end{bmatrix} \]

Class labels

\[ y = [+1 \ ? \ +1 \ ? \ ? \ ? \ -1 \ ? \ -1]^T \]

We wish to predict them.

Assume some labels are unknown.
Given multiple networks

e.g.
Three graphs are given.
Nodes are common.

Example: The 1st graph

\[
A_1 = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
y = [+1, 0, +1, 0, 0, 0, 0, -1, 0, -1]^T
\]

4th, 5th, 9th, and 10th nodes are not connected to any labeled nodes.
Label propagation cannot predict the labels of 4th, 5th, 9th, and 10th nodes, because no info can be conveyed from labeled nodes.
Example: The 2\textsuperscript{nd} graph

\[A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}\]

\[y = [+1 \ 0 \ +1 \ 0 \ 0 \ 0 \ 0 \ -1 \ 0 \ -1]^T\]

All the unknown nodes are not connected to any labeled nodes. Label propagation cannot predict the labels of those nodes, because no info can be conveyed from labeled nodes.

Example: The 3\textsuperscript{rd} graph

\[A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\]

\[y = [+1 \ 0 \ +1 \ 0 \ 0 \ 0 \ 0 \ -1 \ 0 \ -1]^T\]

The labels of adjacent nodes are assumed to be likely to be same. Label propagation wrongly predict the labels of 4\textsuperscript{th}, 5\textsuperscript{th}, 9\textsuperscript{th}, and 10\textsuperscript{th} nodes, because of the edges irrelevant to class labels.
If combining all the graphs simply

\[ A_{\text{int}} = A_1 + A_2 + A_3 \]

\[ y = \begin{bmatrix} +1 & 0 & +1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \end{bmatrix}^T \]

Label propagation wrongly predict the labels of 4th, 5th, 9th, and 10th nodes, because of the edges irrelevant to class labels.

We want to combine the graphs selectively

\[ A_{\text{int}} = A_1 + A_2 \]

\[ y = \begin{bmatrix} +1 & 0 & +1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \end{bmatrix}^T \]

If we can exclude 3rd graph, the labels are correctly predicted.
Objective function

Input

- Adjacency matrices \( A_1, \ldots, A_K \in S^n \)
- Class labels \( y \in \{ \pm 1, 0 \}^n \)

Output

- Scores \( f \in \mathbb{R}^n \)

Integrate multiple networks by linear combination with weights \( \bar{u} \in \mathbb{R}^K \)

\[
A_{\text{int}}(\bar{u}) = \sum_{k=1}^{K} \bar{u}_k A_k.
\]

Define the objective function of the standard label propagation

\[
J(f; A) = \beta_f \sum_{i \in E_{\text{int}}} (y_i - f_i)^2 + \beta_{\text{bias}} \sum_{i=1}^{n} f_i^2 + \beta_{\text{set}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} A_{ij} (f_i - f_j)^2
\]

where

\( \beta_f, \beta_{\text{bias}}, \beta_{\text{set}}, \nu \in \mathbb{R}_+ \): Constant.

\( E_{\text{tra}} \equiv \{ i \in \mathbb{N} | y_i \in \{ \pm 1 \} \} : \) Index set of training nodes.

\( \bar{u} \) and \( f \) will be nearby.

For stabilization.

Labels between adjacent nodes are likely to be same.
Algorithm: What does 2nd step do?

Integrate multiple networks by linear combination and apply the standard label propagation.

Repeat
\[ \hat{f} := \arg\min_f J(f; A_{\text{int}}(u)) \]
until convergence.

\[ \hat{u}_k := \frac{\nu + n}{\nu + \beta_{\text{int}} f^T L_k f} \]

\[ f^T L_k f = \sum_{i=1}^{n} \sum_{i \neq j+1} A^k_{ij} \left( f_i - f_j \right)^2 \]

In a network relevant to the task, adjacent nodes tend to have similar predictions.

\[ f^T L_k f \] will be small for relevant networks, \[ \hat{u}_k \] large.
\[ f^T L_k f \] will be large for irrelevant networks, \[ \hat{u}_k \] small.

i.e. the sum of differences between scores of adjacent nodes.

Contents

Introduction
Algorithm: Intuitively interpretable
Small weights are automatically assigned to networks irrelevant to the task

Relation to statistical inference
Proposed algorithm is reduced to an EM algorithm

Experimental results
Competitive prediction performance to the existing algorithms
Robust against the existence of irrelevant networks
Very efficient
Theorem: Reduced to an EM

Our algorithm is reduced to an EM algorithm for MAP estimation of the parameters in the following probability model:

- **Observations:** \( y_i, \quad i \in \mathcal{I}_q \)
- **Model parameter set:** \( f \in \mathbb{R}^n \)
- **Constant:** \( \beta_k, \beta_{\text{bias}}, \beta_{\text{bias}}^{-1}, \nu \in \mathbb{R}_+ \)
- **Pdf of observations:** \( p(y_i | f) \equiv \mathcal{N}(y_i; f_i, \beta_i^{-1}) \)
- **Prior of parameters:** \( p(f) \equiv \frac{1}{Z} \mathcal{N}(f; 0_n, \beta_{\text{bias}}^{-1}I_n) \prod_{k=1}^K T \left( f; 0_n, \beta_{\text{bias}}^{-1}I_k \right) \)

\( \hat{L}_k \equiv L_k + \epsilon I_n \) at \( \epsilon \to 0 \).

**Student-t density function**

\[
T(x; \mu, \Sigma, \nu) = \int_0^\infty du \gamma(u, 1/2, \nu) N(x; \mu, \Sigma u)
\]

**Gamma density function**

\[
\gamma(u; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} u^{\alpha-1} \exp(-\beta u)
\]

Proof: Reduced to an EM

Define

\[
h_k(f, u_k) = \gamma(u_k; 1/2, \nu, 2) N \left( f; 0_n, \frac{1}{u_k \beta_{\text{bias}}^{-1}} I_k \right).
\]

For all \( u_k \),

\[
\log \int_0^\infty du_k h_k(f, u_k) = \log \int_0^\infty du_k q(u_k) h_k(f, u_k) q(u_k)/q(u_k)
\]

\[
\geq \left( \log h_k(f, u_k) / q(u_k) \right) / q(u_k) = \int_0^\infty du_k q(u_k) \log h_k(f, u_k) + H(q(u_k))
\]

The objective function of MAP estimation is bounded from below as

\[
\mathcal{L}(f; u) = \log p(f) + \sum_{i=1}^N \log p(x_i | f_i)
\]

\[
= -\log Z + \log N(f; 0_n, \beta_{\text{bias}}^{-1}I_k) + \sum_{k=1}^K \log \int_0^\infty du_k h_k(f, u_k) + \sum_{i=1}^N \log p(x_i | f_i)
\]

\[
\geq -\log Z + \log N(f; 0_n, \beta_{\text{bias}}^{-1}I_k) + \sum_{k=1}^K \log h_k(f, u_k) q(u_k) + \sum_{i=1}^N \log p(x_i | f_i)
\]

\[
= \mathcal{L}(f, q(u))
\]

EM attempts to maximize the lower bound.
Proof: Reduced to an EM

EM algorithm repeats E-step and M-step until convergence.

**E-step**

\[ q(u_k) := \text{argmax}_{q} F(f, q(u)) \]

\[ = \text{gammal} \left( u_k; \frac{x + n}{2}, \frac{\nu + m}{2} \right) \]

Denote the expectation of \( u_k \) over \( q(u_k) \) by

\[ \hat{u}_k = \int_0^\infty du_k q(u_k) u_k = \frac{x + n}{\nu + \beta \text{diag}(y)} \]

**M-step**

\[ F(f, q(u)) = -\frac{1}{2} \left( \beta f - y \right)^T G(f - y) + f^T \left( \beta \text{diag}(y) I_K + \beta \sum_{k=1}^{K} \hat{u}_k L_k \right) f + \text{const.} \]

Hence,

\[ f := \text{argmax}_f F(f, q(u)) = \left( G + \frac{\beta \text{diag}(y)}{\nu} I_K + \frac{\beta \sum_{k=1}^{K} \hat{u}_k L_k}{\nu} \right)^{-1} G y \]

Take \( \epsilon \to 0 \).

**Algorithm**

Repeat

\[ f := \left( G + \frac{\beta \text{diag}(y)}{\nu} I_K + \frac{\beta \sum_{k=1}^{K} \hat{u}_k L_k}{\nu} \right)^{-1} G y \]

\[ \hat{u}_k := \frac{\nu + m}{\nu + \beta \text{diag}(y)} \]

until convergence.

* The proposed algorithm is an EM.
Experiments: Algorithms

**Proposed**
Gene expressions

**TSS** (Tsuda et al. 2006)
Gene expressions

**SDP/SVM** (Lanckriet et al. 2004)
Gene expressions

TSS works in the same setting.

An extension of SVM.
Demo: Prediction from multiple graphs

The shading of the edges represents the network weights.

Proposed gave the smallest weight to the 3rd graph.

TSS gave the largest weight to the 3rd graph.
### Experimental Setting

<table>
<thead>
<tr>
<th>Task</th>
<th>Protein function prediction</th>
<th>Handwritten digit classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ribosomal or not</td>
<td></td>
<td></td>
</tr>
<tr>
<td># of nodes</td>
<td>760</td>
<td>200</td>
</tr>
<tr>
<td>Positives</td>
<td>92</td>
<td>100</td>
</tr>
<tr>
<td>Negatives</td>
<td>668</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Networks</th>
<th>[e] 5-NN graph of gene expression</th>
<th>[n1] 1-NN graph</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[vm] PPI network</td>
<td>[n2] (2-NN graph) - [n1]</td>
</tr>
<tr>
<td></td>
<td>[n1] nodes of [e] shuffled</td>
<td>[n3] (3-NN graph) - [n2]</td>
</tr>
<tr>
<td></td>
<td>[n2] nodes of [vm] shuffled</td>
<td>[n4] (4-NN graph) - [n1]</td>
</tr>
<tr>
<td></td>
<td>[n3] nodes of [n1] shuffled</td>
<td>[n5] (5-NN graph) - [n4]</td>
</tr>
<tr>
<td></td>
<td>[n4] nodes of [n2] shuffled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[n5] nodes of [n3] shuffled</td>
<td></td>
</tr>
</tbody>
</table>

Some randomly generated networks are added to test the ability of selective integration of multiple networks.

### Experimental Results

#### Results on protein function prediction

<table>
<thead>
<tr>
<th>Method</th>
<th>ROC score</th>
<th>[n0]</th>
<th>[n1]</th>
<th>[n2]</th>
<th>[n3]</th>
<th>[n4]</th>
<th>[n5]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>0.668</td>
<td>0.279</td>
<td>0.408</td>
<td>0.16</td>
<td>0.154</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSS</td>
<td>0.721</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDP/SVM</td>
<td>0.596</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Results on handwritten digit classification

- Small weights were assigned to the decoy networks
- Much better than TSS
- Competitive to SDP/SVM
Robustness Test

<table>
<thead>
<tr>
<th>Task</th>
<th>Protein function prediction</th>
<th>Ribosomal or not</th>
</tr>
</thead>
<tbody>
<tr>
<td># of nodes</td>
<td>760</td>
<td></td>
</tr>
<tr>
<td>Positives</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>Negatives</td>
<td>668</td>
<td></td>
</tr>
<tr>
<td>Networks</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[e] and [vm]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[e],[vm],[r1], and [r2]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[e],[vm],[r1],[r2],[r3], and [r4]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[e],[vm],[r1],[r2],...,[r5], and [r6]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[e],[vm],[r1],[r2],...,[r7], and [r8]</td>
<td></td>
</tr>
</tbody>
</table>

5-fold CV

We varied the number of randomly generated networks in order to test the robustness of the algorithms.
Results of Robustness Test

- Robust against networks irrelevant to the tasks

Contents

Introduction
Algorithm: Intuitively interpretable
Small weights are automatically assigned to networks irrelevant to the task

Relation to statistical inference
Proposed algorithm is reduced to an EM algorithm

Experimental results
Competitive prediction performance to the existing algorithms
Robust against the existence of irrelevant networks
Very efficient
Computational Time

<table>
<thead>
<tr>
<th>Task</th>
<th>Handwritten digit classification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Odd vs even</strong></td>
<td></td>
</tr>
<tr>
<td><strong># of nodes</strong></td>
<td>200 400 600 800 1000</td>
</tr>
<tr>
<td><strong>Positives</strong></td>
<td>100 200 300 400 500</td>
</tr>
<tr>
<td><strong>Negatives</strong></td>
<td>100 200 300 400 500</td>
</tr>
<tr>
<td><strong>Networks</strong></td>
<td>[n1], [n2], ..., [n5]</td>
</tr>
<tr>
<td><strong>Unlabeled nodes</strong></td>
<td>5-fold CV</td>
</tr>
<tr>
<td><strong>Evaluation</strong></td>
<td>20%</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td></td>
</tr>
</tbody>
</table>

- Proposed algorithm was the most efficient.
Conclusion

Small weights are automatically assigned to networks irrelevant to the task

Proposed algorithm is reduced to an EM algorithm

Competitive prediction performance to the existing algorithms
Robust against the existence of irrelevant networks
Very efficient