Multitask Learning using Nonparametrically Learned Predictor Subspaces

Piyush Rai & Hal Daumé III

School of Computing, University of Utah

December 12, 2009
Learning to Learn

We wish:

- To exploit dependency structure between learning tasks

Why?

- Sharing statistical strength across models
- Improved overall generalization performance

How?

- Learning multiple tasks jointly
$M$ tasks defined by task parameters $\theta_1, \ldots, \theta_M \in \mathbb{R}^D$

Hierarchical Bayesian approaches: use prior knowledge about task-relatedness

We assume a linear shared subspace underlying the task parameters
A Subspace Model for Task-Relatedness

- $M$ tasks defined by task parameters $\theta_1, \ldots, \theta_M \in \mathbb{R}^D$

- **Hierarchical Bayesian approaches:** use prior knowledge about task-relatedness

- We assume a linear **shared subspace** underlying the task parameters

- The subspace is learned **nonparametrically** by automatically controlling its **complexity**
  - Without fixing the **intrinsic dimensionality** *a priori*
  - And automatically inferring its **sparsity**
A Subspace Model for Task-Relatedness

- $M$ tasks defined by task parameters $\theta_1, \ldots, \theta_M \in \mathbb{R}^D$
- Hierarchical Bayesian approaches: use prior knowledge about task-relatedness
- We assume a linear shared subspace underlying the task parameters

The subspace is learned nonparametrically by automatically controlling its complexity
- Without fixing the intrinsic dimensionality a priori
- And automatically inferring its sparsity

Extension: Nonparametrically learned nonlinear shared subspace
A Subspace Model for Task-Relatedness: Formally

Given: $M$ tasks having (unknown) parameters $\theta_1, \ldots, \theta_M \in \mathbb{R}^D$

The generative story: $\theta_m = Z a_m + \epsilon_m \quad \forall m \in [1, \ldots, M]$ 

$\theta_m \in \mathbb{R}^D, Z \in \mathbb{R}^{D \times K}, a_m \in \mathbb{R}^K, \epsilon_m \in \mathbb{R}^D$
Given: \( M \) tasks having (unknown) parameters \( \theta_1, \ldots, \theta_M \in \mathbb{R}^D \)

The generative story: \( \theta_m = Z a_m + \epsilon_m \quad \forall m \in [1, \ldots, M] \)

\( \theta_m \in \mathbb{R}^D, Z \in \mathbb{R}^{D \times K}, a_m \in \mathbb{R}^K, \epsilon_m \in \mathbb{R}^D \)

In matrix notation: \( \Theta = Z A_\theta + E \)

\( \Theta = [\theta_1 \ldots \theta_M] \in \mathbb{R}^{D \times M} \)
\( A_\theta = [a_1 \ldots a_M] \in \mathbb{R}^{K \times M} \)
\( E = [\epsilon_1 \ldots \epsilon_M] \in \mathbb{R}^{D \times M} \)

- \( Z \in \mathbb{R}^{D \times K} \): shared subspace consisting of \( K \) task basis vectors
- Akin to **Factor Analysis** or **Probabilistic PCA** but \( \Theta \) is a latent variable here
Our set-up: $\Theta = ZA_\theta + E$

- The tasks share a $K$-subspace defined by the $D \times K$ matrix $Z$
- How to select the “true” $K$?
A Nonparametric Bayesian Task-Subspace Model

Our set-up: $\Theta = ZA_\theta + E$

- The tasks share a $K$-subspace defined by the $D \times K$ matrix $Z$
- How to select the “true” $K$?
- **Solution**: Model $Z$ using the Indian Buffet Process (IBP)
- But $Z$ is real-valued and IBP is defined over binary matrices
A Nonparametric Bayesian Task-Subspace Model

Our set-up: $\Theta = ZA_\theta + E$

- The tasks share a $K$-subspace defined by the $D \times K$ matrix $Z$
- How to select the “true” $K$?
- **Solution**: Model $Z$ using the Indian Buffet Process (IBP)

But $Z$ is real-valued and IBP is defined over binary matrices

**Solution**: Express $Z$ as $\begin{pmatrix} B & \circ & V \end{pmatrix}$

- $B$ and $V$ are of same size: $D \times K$
- Place the IBP prior on the $B$ matrix
  - Automatically determines $K$ - the number of columns in $B$ and $V$
- .. and a Gaussian prior over $V$
The Indian Buffet Process

- Prior distribution for sparse, infinite binary matrices
- Can model latent features underlying observed data
- **Analogy:** Observations - customers, latent features - dishes
The Indian Buffet Process

- Prior distribution for sparse, infinite binary matrices
- Can model latent features underlying observed data
- **Analogy:** Observations - customers, latent features - dishes

Number of matrix columns determined automatically

**Our case:**
- Each task parameter $\theta_m$: a customer
- task-basis vectors (columns of $Z$): dishes
The Full Model

Given: Data from $M$ tasks. $\mathcal{D} = (\mathbf{X}, \mathbf{Y}) = [(X_1, Y_1) \ldots (X_M, Y_M)]$

\[
\mathbf{Y} \sim \mathcal{N}(\mathbf{X}^T \Theta, \rho^2 I) (\text{regression})
\]

\[
\mathbf{Y} \sim \mathcal{B}(1/(1 + e^{-\mathbf{X}^T \Theta}) (\text{classification})
\]

\[
\Theta = (\mathbf{B} \odot \mathbf{V}) \mathbf{A}_\theta + \mathbf{E}
\]

\[
\mathbf{B} \sim \text{IBP}(\alpha)
\]

\[
\mathbf{V} \sim \mathcal{N}(0, \sigma_v^2 I), \quad \sigma_v \sim \text{IG}(a, b)
\]

\[
\mathbf{A}_\theta \sim \mathcal{N}(0, \sigma_\theta^2 I), \quad \sigma_\theta \sim \text{IG}(c, d)
\]

\[
\mathbf{E} \sim \mathcal{N}(0, \Psi), \quad \Psi_D \sim \text{IG}(e, f)
\]
The basic model has only $\Theta$ as “data” for learning $Z$

More data should help!

Use inputs $X$ as well

Inference in both models via MCMC
Experiments

- Two multi-label datasets (Yeast and Scene, from UCI repository)
- Baselines: Logistic regression, pooling, task-clustering (yaxue)

<table>
<thead>
<tr>
<th>Model</th>
<th>Yeast</th>
<th>Scene</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>AUC</td>
</tr>
<tr>
<td>LR</td>
<td>0.5047</td>
<td>0.5049</td>
</tr>
<tr>
<td>pool</td>
<td>0.4983</td>
<td>0.5112</td>
</tr>
<tr>
<td>yaxue</td>
<td>0.5106</td>
<td>0.5105</td>
</tr>
<tr>
<td>Model-1</td>
<td>0.5212</td>
<td>0.5244</td>
</tr>
<tr>
<td>Model-2</td>
<td>0.5424</td>
<td>0.5406</td>
</tr>
</tbody>
</table>

Effect of varying data size (Scene dataset):

![Graphs showing the effect of varying data size on accuracy and AUC score for different models.](image)
Mixture of Subspaces Extension

- Natural extensions to more complex settings

- Nonlinear subspaces: Can be seen as mixture of linear subspaces

\[ p(\theta_m) = \sum_{i=1}^{L} \pi_i \mathcal{N}(\mu_i, Z_iZ_i^T + \Psi_i). \]

- Discovering manifold structure underlying task parameters
- Segregating outlier tasks (by allowing more than one linear subspace)

\( \mu_i \): component means, \( Z_i \): factor loadings, and \( \pi_i \): mixing proportions.
Mixture of Subspaces Extension

• Natural extensions to more complex settings

• Nonlinear subspaces: Can be seen as mixture of linear subspaces
  \[ p(\theta_m) = \sum_{i=1}^{L} \pi_i \mathcal{N}(\mu_i, Z_iZ_i^T + \Psi_i). \]
  • Discovering manifold structure underlying task parameters
  • Segregating outlier tasks (by allowing more than one linear subspace)

\( \mu_i \): component means, \( Z_i \): factor loadings, and \( \pi_i \): mixing proportions.

• A Dirichlet Process prior on mixing proportions \( \pi_i \) can determine the number of mixture components

• An IBP prior on \( Z_i \) can determine the dimensionality of each subspace
A nonparametric Bayesian framework for multitask learning

Based on learning a shared subspace of task parameters

Complexity (dimensionality and sparsity) of the shared subspace determined automatically

Natural extensions to more general nonparametric frameworks
A nonparametric Bayesian framework for multitask learning

Based on learning a shared subspace of task parameters

Complexity (dimensionality and sparsity) of the shared subspace determined automatically

Natural extensions to more general nonparametric frameworks

Thanks! Questions?