Multitask Learning using Nonparametrically Learned Predictor Subspaces

Piyush Rai & Hal Daumé III

School of Computing, University of Utah

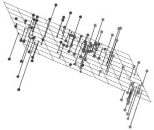
December 12, 2009

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**

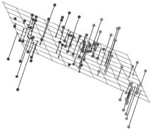
A Subspace Model for Task-Relatedness

- 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



A Subspace Model for Task-Relatedness

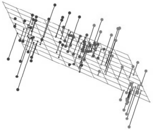
- M tasks defined by task parameters $\theta_1, \ldots, \theta_M \in \mathbb{R}^D$
- Hierarchical Bayesian approaches: use prior knowledge about task-relatedness
- We assume a linear shared subspace underlying the task parameters



- 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: $\theta_m = \mathbf{Z}\mathbf{a}_m + \epsilon_m \qquad \forall m \in [1, \ldots, M]$

$\theta_m \in \mathbb{R}^D, \mathbf{Z} \in \mathbb{R}^{D \times K}, \mathbf{a}_m \in \mathbb{R}^K, \epsilon_m \in \mathbb{R}^D$

A Subspace Model for Task-Relatedness: Formally

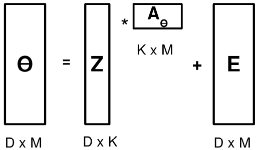
Given: M tasks having (unknown) parameters $\theta_1, \ldots, \theta_M \in \mathbb{R}^D$

The generative story: $\theta_m = \mathbf{Z}\mathbf{a}_m + \epsilon_m \quad \forall m \in [1, \ldots, M]$

$\theta_m \in \mathbb{R}^D, \mathbf{Z} \in \mathbb{R}^{D \times K}, \mathbf{a}_m \in \mathbb{R}^K, \epsilon_m \in \mathbb{R}^D$

In matrix notation: $\Theta = \mathbf{Z}\mathbf{A}_\theta + \mathbf{E}$

$\Theta = [\theta_1 \ldots \theta_M] \in \mathbb{R}^{D \times M}$
$\mathbf{A}_\theta = [\mathbf{a}_1 \ldots \mathbf{a}_M] \in \mathbb{R}^{K \times M}$
$\mathbf{E} = [\epsilon_1 \ldots \epsilon_M] \in \mathbb{R}^{D \times M}$



- $\mathbf{Z} \in \mathbb{R}^{D \times K}$: shared subspace consisting of K task basis vectors
- Akin to **Factor Analysis** or **Probabilistic PCA** but Θ is a latent variable here

A Nonparametric Bayesian Task-Subspace Model

Our set-up: $\Theta = \mathbf{Z}\mathbf{A}_\theta + \mathbf{E}$

- The tasks share a K-subspace defined by the $D \times K$ matrix $\mathbf{Z}$
- How to select the "true" K?

A Nonparametric Bayesian Task-Subspace Model

Our set-up: $\Theta = \mathbf{Z}\mathbf{A}_\theta + \mathbf{E}$

- The tasks share a K-subspace defined by the $D \times K$ matrix $\mathbf{Z}$
- How to select the "true" K?
- Solution: Model $\mathbf{Z}$ using the Indian Buffet Process (IBP)
- But $\mathbf{Z}$ is real-valued and IBP is defined over binary matrices

A Nonparametric Bayesian Task-Subspace Model

Our set-up: $\Theta = \mathbf{Z}\mathbf{A}_\theta + \mathbf{E}$

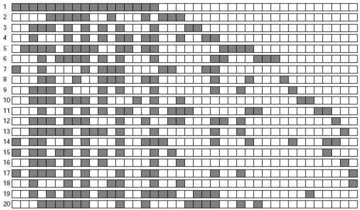
- The tasks share a K-subspace defined by the $D \times K$ matrix $\mathbf{Z}$
- How to select the "true" K?
- Solution: Model $\mathbf{Z}$ using the Indian Buffet Process (IBP)
- But $\mathbf{Z}$ is real-valued and IBP is defined over binary matrices

Solution: Express $\mathbf{Z}$ as $\underbrace{\mathbf{B}}_{binary} \odot \underbrace{\mathbf{V}}_{real}$

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

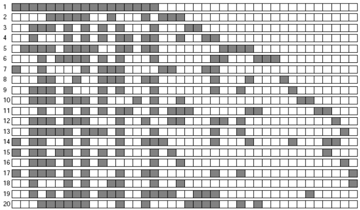
The Indian Buffet Process

- 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

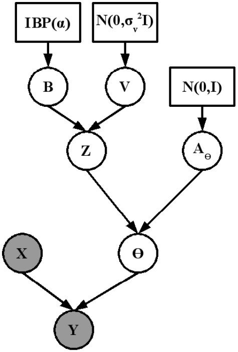


- Number of matrix columns determined automatically
- **Our case:**
  - Each task parameter θ_m: a customer
  - task-basis vectors (columns of $\mathbf{Z}$): dishes

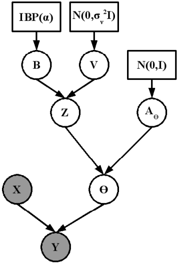
The Full Model

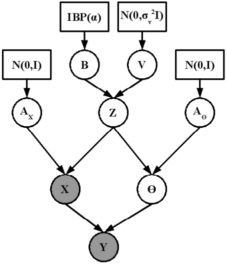
Given: Data from M tasks. $\mathcal{D} = (\mathbf{X}, \mathbf{Y}) = [(X_1, Y_1) \ldots (X_M, Y_M)]$

$$
\begin{aligned}
\mathbf{Y} &\sim \mathcal{N}or(\mathbf{X}^T\Theta, \rho^2 I)(regression) \\
\mathbf{Y} &\sim \mathcal{B}in(1/(1 + e^{-\mathbf{X}^T\Theta})(classification) \\
\Theta &= (\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta + \mathbf{E} \\
\mathbf{B} &\sim \mathcal{IBP}(\alpha) \\
\mathbf{V} &\sim \mathcal{N}or(0, \sigma_v^2\mathbf{I}), \qquad \sigma_v \sim IG(a, b) \\
\mathbf{A}_\theta &\sim \mathcal{N}or(0, \sigma_\theta^2\mathbf{I}), \qquad \sigma_\theta \sim IG(c, d) \\
\mathbf{E} &\sim \mathcal{N}or(0, \Psi), \qquad \Psi_D \sim IG(e, f)
\end{aligned}
$$



An Augmented Model





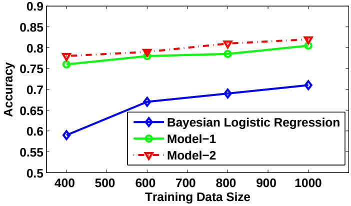
- The basic model has only Θ as "data" for learning $\mathbf{Z}$
- More data should help!
- Use inputs $\mathbf{X}$ as well
- Inference in both models via MCMC

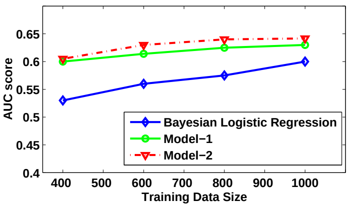
Experiments

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

Model	Yeast		Scene	
	Acc	AUC	Acc	AUC
LR	0.5047	0.5049	0.7362	0.6153
pool	0.4983	0.5112	0.7862	0.5433
yaxue	0.5106	0.5105	0.7765	0.5603
Model-1	0.5212	0.5244	0.7756	0.6325
Model-2	**0.5424**	**0.5406**	**0.7911**	**0.6416**

Effect of varying data size (Scene dataset):





Mixture of Subspaces Extension

- Natural extensions to more complex settings
- Nonlinear subspaces: Can be seen as mixture of linear subspaces

$$p(\theta_m) = \sum_{i=1}^{L} \pi_i \mathcal{N}or(\mu_i, \mathbf{Z}_i\mathbf{Z}_i^T + \Psi_i).$$

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

μ_i: component means, $\mathbf{Z}_i$: factor loadings, and π_i: mixing proportions.

Mixture of Subspaces Extension

- Natural extensions to more complex settings
- Nonlinear subspaces: Can be seen as mixture of linear subspaces

$$p(\theta_m) = \sum_{i=1}^{L} \pi_i \mathcal{N}or(\mu_i, \mathbf{Z}_i \mathbf{Z}_i^T + \Psi_i).$$

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

  μ_i: component means, $\mathbf{Z}_i$: factor loadings, and π_i: mixing proportions.

- A Dirichlet Process prior on mixing proportions π_i can determine the number of mixture components
- An IBP prior on $\mathbf{Z}_i$ can determine the dimensionality of each subspace

Conclusion

- 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

Conclusion

- 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?