Best Of NIPS 2005: Highlights on the 'Inductive Transfer : 10 Years Later' Workshop

Danny Silver, Goekhan Bakir, Kristin Bennett, Rich Caruana, Massimiliano Pontil, Stuart Russell, Prasad Tadepalli

February 21st 2006
Plan

1. Introduction
2. Workshop Motivations
4. Transfer Learning by Constructing Informative Priors
5. Conclusion
What is Inductive Transfer

**Definition**: Inductive transfer or transfer learning refers to the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task.
What is Inductive Transfer

**Definition:** Inductive transfer or transfer learning refers to the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task.

**Successes:**
- Medical Decision Making (Caruana, 1997)
- Recommendation Systems (Collaborative Filtering)
- Named Entity Recognition (Ando and Zhang, 2005)
What is Inductive Transfer

**Definition**: Inductive transfer or transfer learning refers to the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypothesis for a new task.

**Successes**:
- Medical Decision Making (Caruana, 1997)
- Recommendation Systems (Collaborative Filtering)
- Named Entity Recognition (Ando and Zhang, 2005)

**Algorithms**:
- ANN
- kNN
- Decision Trees
- SVM
- Hierarchical Bayesian models
Inductive Transfer really suffers from multiple personality disorder:

- Learning to Learn
- Lifelong Learning
- Multitask Learning
- Hints
- Continual Learning
- Speedup Learning
- Hierarchical Bayes
- and others...
Plan

1. Introduction
2. Workshop Motivations
4. Transfer Learning by Constructing Informative Priors
5. Conclusion
Workshop Motivations

- NIPS 1995: two-day workshop on “Learning to Learn”, organised by Danny Silver and Rich Caruana, that focused on the need for lifelong machine learning methods that retain and reuse learned knowledge
NIPS 1995: two-day workshop on “Learning to Learn”, organised by Danny Silver and Rich Caruana, that focused on the need for lifelong machine learning methods that retain and reuse learned knowledge.

NIPS 2005: “Inductive Transfer: 10 Years Later” workshop, to examine the progress that has been made in ten years, the questions and challenges that remain, and the opportunities for new applications of inductive transfer systems.
Workshop Motivations

- NIPS 1995: two-day workshop on “Learning to Learn”, organised by Danny Silver and Rich Caruana, that focused on the need for lifelong machine learning methods that retain and reuse learned knowledge.

- NIPS 2005: “Inductive Transfer : 10 Years Later” workshop, to examine the progress that has been made in ten years, the questions and challenges that remain, and the opportunities for new applications of inductive transfer systems.

- The next day, a related workshop was held, called “Interclass Transfer : why learning to recognize many objects is easier than learning to recognize just one”
In this talk, I’m going to describe two different ways doing inductive transfer:

1. Parallel transfer, i.e. all tasks are learned in parallel in: Sparsity Models for Multi-task Learning
2. Sequential transfer, i.e. auxiliary tasks are used to bias the learning of a new task in: Transfer Learning by Constructing Informative Priors
In this talk, I’m going to describe two different ways doing inductive transfer:

- parallel transfer, i.e. all tasks are learned in parallel in:
  **Sparsity Models for Multi-task Learning**
In this talk... 

- In this talk, I'm going to describe two different ways doing inductive transfer:
  - parallel transfer, i.e. all tasks are learned in parallel in: **Sparsity Models for Multi-task Learning**
  - sequential transfer, i.e. auxiliary tasks are used to bias the learning of a new task in: **Transfer Learning by Constructing Informative Priors**
Notation

Notation used for the rest of the talk:

- $\mathcal{X} = \mathbb{R}^N$ is the input space
Notation

Notation used for the rest of the talk:

- $\mathcal{X} = \mathbb{R}^N$ is the input space
- $\mathcal{Y} = \{1, \ldots, K\}$ is the output space for all tasks, and $\mathcal{Y}^{(k)} = \{0, 1\}$ is the output space for the binary classification task $k$
Notation used for the rest of the talk:

- $\mathcal{X} = \mathbb{R}^N$ is the input space
- $\mathcal{Y} = \{1, \ldots, K\}$ is the output space for all tasks, and $\mathcal{Y}^{(k)} = \{0, 1\}$ is the output space for the binary classification task $k$
- $K$ is the number of tasks

Both papers concentrate on logistic regression, so:

$$f^{(k)}(x) = P(Y^{(k)} = 1 | X = x, \theta_k) = \sigma(\theta_k^T x) = \frac{1}{1 + e^{-\theta_k^T x}}$$
Notation

Notation used for the rest of the talk:

- $\mathcal{X} = \mathbb{R}^N$ is the input space
- $\mathcal{Y} = \{1, \ldots, K\}$ is the output space for all tasks, and $\mathcal{Y}^{(k)} = \{0, 1\}$ is the output space for the binary classification task $k$
- $K$ is the number of tasks
- $f^{(1)}, f^{(2)}, \ldots, f^{(K)}$ are the predictors $f^{(k)} : \mathcal{X} \rightarrow [0, 1]$
Notation

Notation used for the rest of the talk:

- $\mathcal{X} = \mathbb{R}^N$ is the input space
- $\mathcal{Y} = \{1, \ldots, K\}$ is the output space for all tasks, and $\mathcal{Y}^{(k)} = \{0, 1\}$ is the output space for the binary classification task $k$
- $K$ is the number of tasks
- $f^{(1)}, f^{(2)}, \ldots, f^{(K)}$ are the predictors $f^{(k)} : \mathcal{X} \rightarrow [0, 1]$
- $\mathcal{D}_k = \{(x_1^{(k)}, y_1^{(k)}), \ldots, (x_{n_k}^{(k)}, y_{n_k}^{(k)})\}$ the training set for $k^{th}$ task
Notation

Notation used for the rest of the talk:

- $\mathcal{X} = \mathbb{R}^N$ is the input space
- $\mathcal{Y} = \{1, \ldots, K\}$ is the output space for all tasks, and $\mathcal{Y}^{(k)} = \{0, 1\}$ is the output space for the binary classification task $k$
- $K$ is the number of tasks
- $f^{(1)}, f^{(2)}, \ldots, f^{(K)}$ are the predictors $f^{(k)} : \mathcal{X} \rightarrow [0, 1]$
- $\mathcal{D}_k = \{(x_1^{(k)}, y_1^{(k)}), \ldots, (x_{n_k}^{(k)}, y_{n_k}^{(k)})\}$ the training set for $k^{th}$ task
- both papers concentrate on logistic regression, so:

$$f^{(k)}(x) = P(Y^{(k)} = 1|X = x, \theta_k) = \sigma(\theta_k^T x) = \frac{1}{1 + e^{-\theta_k^T x}}$$
Plan

1. Introduction
2. Workshop Motivations
4. Transfer Learning by Constructing Informative Priors
5. Conclusion
This paper by (Zhang, 2005) presents a hierarchical Bayesian model for multitask learning.
This paper by (Zhang, 2005) presents a hierarchical Bayesian model for multitask learning. The model is simply specified as follows:

\[
\theta_k = \Lambda s_k + e_k \\
e_k \sim \mathcal{N}(0, \lambda I) \\
y_k \sim \mathcal{B}(\sigma(\theta_k^T x))
\]

where \( s_k \in \mathbb{R}^{H \times 1} \), \( \Lambda \in \mathbb{R}^{n \times H} \), \( H \) is the number of “bases” and \( \sigma(\cdot) \) is the sigmoid function.
This paper by (Zhang, 2005) presents a hierarchical Bayesian model for multitask learning.

The model is simply specified as follows:

\[ \theta_k = \Lambda s_k + e_k \]
\[ e_k \sim \mathcal{N}(0, \lambda I) \]
\[ y_k \sim \mathcal{B}(\sigma(\theta_k^T x)) \]

where \( s_k \in \mathbb{R}^{H \times 1} \), \( \Lambda \in \mathbb{R}^{n \times H} \), \( H \) is the number of “bases” and \( \sigma(\cdot) \) is the sigmoid function.

To encourage sparsity in \( s_k \) or in the columns of \( \Lambda \), assume a Laplace prior:

\[ p(a) \propto \prod_i e^{-|a_i|} \]
This paper by (Zhang, 2005) presents a hierarchical Bayesian model for multitask learning. The model is simply specified as follows:

\[
\begin{align*}
\theta_k &= \Lambda s_k + e_k \\
e_k &\sim \mathcal{N}(0, \lambda I) \\
y_k &\sim \mathcal{B}(\sigma(\theta_k^T x))
\end{align*}
\]

where \( s_k \in \mathbb{R}^{H \times 1} \), \( \Lambda \in \mathbb{R}^{n \times H} \), \( H \) is the number of “bases” and \( \sigma(\cdot) \) is the sigmoid function.

To encourage sparsity in \( s_k \) or in the columns of \( \Lambda \), assume a Laplace prior:

\[
p(a) \propto \prod_i e^{-|a_i|}
\]

Here, we are going to applying this prior to the columns of \( \Lambda \).
Training algorithm for $\Lambda$ and the $\theta_k$

- The following procedure is used until convergence
  1. Compute, for $k = 1, \ldots, K$:
     \[
     (\hat{\theta}_k, \hat{s}_k) = \arg\max_{\theta_k, s_k} \left\{ \frac{1}{n_k} \sum_{i=1}^{n_k} \log P(y_i^{(k)}|x_i^{(k)}, \theta_k^T) + \log \mathcal{N}(\theta_k|\hat{\Lambda}s_k, \lambda I) \right\}
     \]
  2. Compute:
     \[
     \hat{\Lambda} = \arg\max_{\Lambda} \left\{ \sum_{k=1}^{K} \log \mathcal{N}(\theta_k|\hat{\Lambda}s_k, \lambda I) + \log p(\Lambda) \right\}
     \]
     which corresponds to a set of Lasso-style problems
     \[
     \hat{\Lambda} = \arg\min_{\Lambda} \left\{ \sum_{k=1}^{K} (\hat{\theta}_k - \Lambda\hat{s}_k)^T (\hat{\theta}_k - \Lambda\hat{s}_k) + \gamma \sum_{h=1}^{H} \sum_{f=1}^{F} |\Lambda_{f,h}| \right\}
     \]
     where $\gamma$ controls the sparsity of the solution $\hat{\Lambda}$
The following procedure is used until convergence

1. Compute, for $k = 1, \ldots, K$:

\[
(\hat{\theta}_k, \hat{s}_k) = \arg\max_{\theta_k, s_k} \left\{ \frac{1}{n_k} \sum_{i=1}^{n_k} \log P(y_i^{(k)} | x_i^{(k)}, \theta_k^T) + \log \mathcal{N}(\theta_k | \hat{\Lambda} s_k, \lambda I) \right\}
\]

2. Compute:

\[
\hat{\Lambda} = \arg\max_{\Lambda} \left\{ \sum_{k=1}^{K} \log \mathcal{N}(\theta_k | \hat{\Lambda} s_k, \lambda I) + \log p(\Lambda) \right\}
\]

which corresponds to a set of Lasso-style problems

\[
\hat{\Lambda} = \arg\min_{\Lambda} \left\{ \sum_{k=1}^{K} (\hat{\theta}_k - \Lambda \hat{s}_k)^T (\hat{\theta}_k - \Lambda \hat{s}_k) + \gamma \sum_{h=1}^{H} \sum_{f=1}^{F} |\Lambda_{f,h}| \right\}
\]

where $\gamma$ controls the sparsity of the solution $\hat{\Lambda}$

This is not as much Bayesian as we might have thought (or feared)
Results

Figure 1: LEFT: Text classification results on RCV1; RIGHT: Average sparsity rate of elements in $\hat{A}$. 
Plan

1. Introduction
2. Workshop Motivations
4. Transfer Learning by Constructing Informative Priors
5. Conclusion
This paper by (Raina, Ng and Koller, 2005) presents a method to learn a prior over the parameters of a logistic regressor, based on logistic regressors trained on auxiliary tasks.
Transfer Learning by Constructing Informative Priors

- This paper by (Raina, Ng and Koller, 2005) presents a method to learn a prior over the parameters of a logistic regressor, based on logistic regressors trained on auxiliary tasks.
- This prior is used to train a new logistic regressor on a new task.
Transfer Learning by Constructing Informative Priors

- This paper by (Raina, Ng and Koller, 2005) presents a method to learn a prior over the parameters of a logistic regressor, based on logistic regressors trained on auxiliary tasks.
- This prior is used to train a new logistic regressor on a new task.
- Particularly, in the following optimization problem:

\[
\theta_{\text{new}} = \operatorname{argmax}_{\theta} \left\{ \sum_{i=1}^{n_{\text{new}}} \log P(y_i^{(\text{new})} | x_i^{(\text{new})}, \theta) - \theta^T \Sigma^{-1} \theta \right\}
\]

instead of having \( \Sigma = \lambda I \), we are going to learn a good \( \Sigma \).
Training procedure for entries $\Sigma_{i,j}$

- Generate a training set $C_{i,j}$ from one of the training set $\mathcal{D}_k$, where only $m$ input components are visible, including $x_i$ and $x_j$.
Training procedure for entries \( \Sigma_{i,j} \)

- Generate a training set \( C_{i,j} \) from one of the training set \( \mathcal{D}_k \), where only \( m \) input components are visible, including \( x_i \) and \( x_j \)
- Train a logistic regressor on \( C_{i,j} \) to obtain \( \hat{\theta} \)

Two problems arise though:

1. To obtain the complete matrix \( \Sigma \), we need to do \( N(N+1)/2 \) times.
2. \( \Sigma \) may not be positive semidefinite.
Training procedure for entries $\Sigma_{i,j}$

- Generate a training set $C_{i,j}$ from one of the training set $\mathcal{D}_k$, where only $m$ input components are visible, including $x_i$ and $x_j$.
- Train a logistic regressor on $C_{i,j}$ to obtain $\hat{\theta}$.
- Use $\hat{\theta}_i$ and $\hat{\theta}_j$ as samples to estimate $\Sigma_{i,j} = \text{Cov}(\theta_i, \theta_j)$, based on a bootstrap estimator.
Training procedure for entries $\Sigma_{i,j}$

- Generate a training set $C_{i,j}$ from one of the training set $\mathcal{D}_k$, where only $m$ input components are visible, including $x_i$ and $x_j$
- Train a logistic regressor on $C_{i,j}$ to obtain $\hat{\theta}$
- Use $\hat{\theta}_i$ and $\hat{\theta}_j$ as samples to estimate $\Sigma_{i,j} = \text{Cov}(\theta_i, \theta_j)$, based on a bootstrap estimator
- Repeat until enough samples are gathered.
Training procedure for entries $\Sigma_{i,j}$

- Generate a training set $C_{i,j}$ from one of the training set $\mathcal{D}_k$, where only $m$ input components are visible, including $x_i$ and $x_j$
- Train a logistic regressor on $C_{i,j}$ to obtain $\hat{\theta}$
- Use $\hat{\theta}_i$ and $\hat{\theta}_j$ as samples to estimate $\Sigma_{i,j} = \text{Cov}(\theta_i, \theta_j)$, based on a bootstrap estimator
- Repeat until enough samples are gathered.

Two problems arise though:
Training procedure for entries $\Sigma_{i,j}$

- Generate a training set $C_{i,j}$ from one of the training set $D_k$, where only $m$ input components are visible, including $x_i$ and $x_j$
- Train a logistic regressor on $C_{i,j}$ to obtain $\hat{\theta}$
- Use $\hat{\theta}_i$ and $\hat{\theta}_j$ as samples to estimate $\Sigma_{i,j} = Cov(\theta_i, \theta_j)$, based on a bootstrap estimator
- Repeat until enough samples are gathered.

Two problems arise though:

1. To obtain the complete matrix $\Sigma$, we need to do $N(N + 1)/2$ times
Training procedure for entries $\Sigma_{i,j}$

- Generate a training set $C_{i,j}$ from one of the training set $\mathcal{D}_k$, where only $m$ input components are visible, including $x_i$ and $x_j$
- Train a logistic regressor on $C_{i,j}$ to obtain $\hat{\theta}$
- Use $\hat{\theta}_i$ and $\hat{\theta}_j$ as samples to estimate $\Sigma_{i,j} = \text{Cov}(\theta_i, \theta_j)$, based on a bootstrap estimator
- Repeat until enough samples are gathered.

Two problems arise though:
1. To obtain the complete matrix $\Sigma$, we need to do $N(N + 1)/2$ times
2. $\Sigma$ may not be positive semidefinite
Training procedure for $\Sigma$

**Solution to first problem**: parametrize the entries $\Sigma_{i,j}$ as follows:

$$\hat{\Sigma}_{i,j} = \Psi^T f_{i,j}$$

where $f_{i,j}$ is a feature vector for the pair $(i, j)$. This way, all elements can be predicted given $\Psi$ and $f_{i,j}$.
Training procedure for $\Sigma$

- **Solution to first problem**: parametrize the entries $\Sigma_{i,j}$ as follows:

$$\hat{\Sigma}_{i,j} = \Psi^T f_{i,j}$$

where $f_{i,j}$ is a feature vector for the pair $(i, j)$. This way, all elements can be predicted given $\Psi$ and $f_{i,j}$.

- **Solution to second problem**: solve the following optimization problem:

$$\min_{\Psi, \Sigma} \left\{ \sum_{(i,j) \in G} (e_{i,j} - \Psi^T f_{i,j})^2 + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} (\Sigma_{i,j} - \Psi^T f_{i,j})^2 \right\} \quad \text{s.t. } \Sigma \succeq 0$$

where $G$ is the set of pairs $(i, j)$ for which we have computed an estimate of $\Sigma_{i,j} e_{i,j}$. This can be optimized by alternating between the optimization of $\Psi$ with $\Sigma$ and vice versa.
Training procedure for $\Sigma$

- The obtained $\Sigma$ captures well the relative magnitude of $\Sigma_{i,j}$, but not their absolute value.
Training procedure for $\Sigma$

- The obtained $\Sigma$ captures well the relative magnitude of $\Sigma_{i,j}$, but not their absolute value.
- To solve this, a scaling parameter $q$ on $\Sigma$ is learned so as to maximize the local likelihood of the training cases in the different $D_k$. 
Results

Figure 1: Classification results on the 20 newsgroups dataset. Training set size is graphed on a log-scale. (a) Average test error for different training set sizes. Blue circles are for our SDP-based method, green triangles for SDP with the diagonal covariance constraint, red stars for the baseline diagonal prior. (b) Average percentage reduction in test error over the baseline: Blue circles are for the SDP-based method, green triangles for SDP with the diagonal covariance constraint. (c) Percentage reduction in test error over the baseline for each of the 10 runs of the SDP-based method. [Colors where available.]
Transfer Learning by Constructing Informative Priors

Results

<table>
<thead>
<tr>
<th></th>
<th>wave</th>
<th>mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>insurance</td>
<td>mile</td>
<td></td>
</tr>
<tr>
<td>rear</td>
<td>mile</td>
<td></td>
</tr>
<tr>
<td>honda</td>
<td>mile</td>
<td></td>
</tr>
<tr>
<td>brake</td>
<td>gear</td>
<td></td>
</tr>
<tr>
<td>meg</td>
<td>printer</td>
<td>wave</td>
</tr>
<tr>
<td>wheel</td>
<td></td>
<td>wave</td>
</tr>
<tr>
<td>bmw</td>
<td>seat</td>
<td></td>
</tr>
<tr>
<td>desktop</td>
<td>ram</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Word pairs from the classification problem `rec.motorcycles` versus `comp.os.ms-windows.misc` that were estimated to have the most positive (left) or most negative (right) bootstrap-corrected parameter covariance using auxiliary learning problems.
Plan

1. Introduction
2. Workshop Motivations
4. Transfer Learning by Constructing Informative Priors
5. Conclusion
Many other subjects were addressed during the workshop:

- Temporal Inductive Transfer
Conclusion

Many other subjects were addressed during the workshop
- Temporal Inductive Transfer
- Cross-Domain Knowledge Transfer
Conclusion

Many other subjects were addressed during the workshop:
- Temporal Inductive Transfer
- Cross-Domain Knowledge Transfer
- To Transfer or Not To Transfer
Conclusion

Many other subjects were addressed during the workshop:

- Temporal Inductive Transfer
- Cross-Domain Knowledge Transfer
- To Transfer or Not To Transfer
- Inductive Transfer in:
  - Kernel methods
Conclusion

Many other subjects were addressed during the workshop:

- Temporal Inductive Transfer
- Cross-Domain Knowledge Transfer
- To Transfer or Not To Transfer
- Inductive Transfer in:
  - Kernel methods
  - Bayes Nets
Conclusion

Many other subjects were addressed during the workshop:

- Temporal Inductive Transfer
- Cross-Domain Knowledge Transfer
- To Transfer or Not To Transfer
- Inductive Transfer in:
  - Kernel methods
  - Bayes Nets
  - Collaborative Filtering
Conclusion

Many other subjects were addressed during the workshop:

- Temporal Inductive Transfer
- Cross-Domain Knowledge Transfer
- To Transfer or Not To Transfer
- Inductive Transfer in:
  - Kernel methods
  - Bayes Nets
  - Collaborative Filtering
  - Reinforcement Learning
Conclusion

Many other subjects were addressed during the workshop:

- Temporal Inductive Transfer
- Cross-Domain Knowledge Transfer
- To Transfer or Not To Transfer
- Inductive Transfer in:
  - Kernel methods
  - Bayes Nets
  - Collaborative Filtering
  - Reinforcement Learning
  - Learning Curves
The organizers identified three major goals:

1. To summarize the work done thus far in the area
The organizers identified three major goals:

1. To summarize the work done thus far in the area
2. To share new theories, approaches and algorithms
The organizers identified three major goals:

1. To summarize the work done thus far in the area
2. To share new theories, approaches and algorithms
3. To discuss and organize a more formal inductive transfer community
Conclusion

The organizers identified three major goals:

1. To summarize the work done thus far in the area
2. To share new theories, approaches and algorithms
3. To discuss and organize a more formal inductive transfer community

For more details: http://iitrl.acadiau.ca/itws05/index.htm

