Transfer Learning

Jaromir Savelka

Intelligent Systems Program
University of Pittsburgh

jas438@pitt.edu

November 13, 2014

Lecture Outline

Intuition and Motivation

Framework Introduction

Categorization

Inductive Transfer Learning

Unsupervised Transfer Learning

Transductive Transfer Learning

Applications

Limitations

Lecture Outline

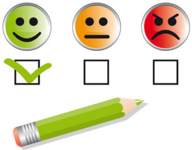
Transfer in Learning (in Theory of Learning)

- In theory of learning transfer of learning occurs when learning in one context enhances (positive transfer) or undermines (negative transfer) a related performance in another context.
- Transfer includes near transfer (to closely related contexts and performances) and far transfer (to rather different contexts and performances).
- Positive examples:
  1. Learning to drive a car helps a person later to learn more quickly to drive a truck.
  2. Learning mathematics prepares students to study physics.
  3. Learning to get along with one's siblings may prepare one for getting along better with others.
  4. Experience playing chess might even make one a better strategic thinker in politics or business.

[10] Perkins & Salomon 1992

Motivation for Transfer Learning

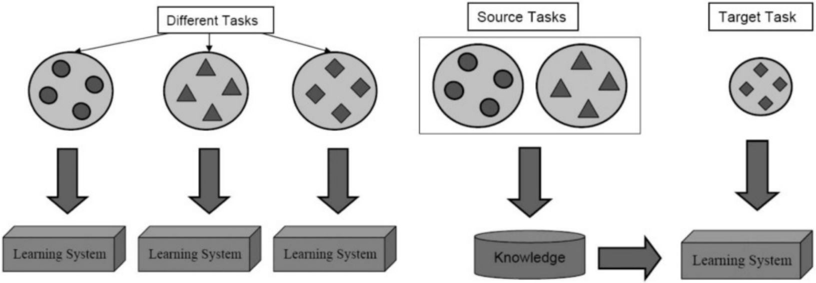
- Traditional machine learning methods assume that training and test data come from the same feature space and the same distribution.
- When the feature space or the distribution change the models need to be rebuilt from scratch using newly collected training data which is often expensive or impossible at all.
- Knowledge transfer or transfer learning between task domains would be desirable.



[8] Pan & Yang 2010

Transfer Learning (in Machine Learning)

- Transfer learning, in contrast to traditional ML framework, allows the domains, tasks, and distributions used in training and testing to be different.
- Transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task.



[8] Pan & Yang 2010

Lecture Outline

Domain and Task

Let us have feature space $\mathcal{X}$ and a marginal probability distribution $P(\boldsymbol{X})$, where $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots \boldsymbol{x}^{(n)}\} \in \mathcal{X}$.

Domain

$\mathcal{D} = \{\mathcal{X}, P(\boldsymbol{X})\}$

Additionally, let us have label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$ which is not observed.

Task

$\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$

The predictive function $f(\cdot)$ can be learned from the training data of the form $\{\boldsymbol{x}^{(i)}, y^{(i)}\}$, where $\boldsymbol{x}^{(i)} \in \boldsymbol{X}$ and $y_i \in \mathcal{Y}$, and used to predict label $y^{(i)}$ for data point $\boldsymbol{x}^{(i)}$ ($f(\boldsymbol{x}^{(i)}) = y^{(i)}$).

[8] Pan & Yang 2010

Source and Target Domain and Task

Let us consider the case where there are two domains:

1. Source domain $\mathcal{D}_S = \{\mathcal{X}_S, P_S(\boldsymbol{X})\}$ where $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots \boldsymbol{x}^{(n)}\} \in \mathcal{X}_S$
2. Target domain $\mathcal{D}_T = \{\mathcal{X}_T, P_T(\boldsymbol{X})\}$ where $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots \boldsymbol{x}^{(n)}\} \in \mathcal{X}_T$

And correspondingly, two tasks:

1. Source task $\mathcal{T}_S = \{\mathcal{Y}_S, f_S(\cdot)\}$ where $f_S(\cdot) \rightarrow y_i \in \mathcal{Y}_S$
2. Target task $\mathcal{T}_T = \{\mathcal{Y}_T, f_T(\cdot)\}$ where $f_T(\cdot) \rightarrow y_i \in \mathcal{Y}_T$

Note that source and target domains are connected to the source and target tasks respectively through predictive functions
$f_S(\boldsymbol{x}^{(i)}) = y_i$ where $\boldsymbol{x}^{(i)} \in \mathcal{X}_S$ and $y_i \in \mathcal{Y}_S$
$f_T(\boldsymbol{x}^{(i)}) = y_i$ where $\boldsymbol{x}^{(i)} \in \mathcal{X}_T$ and $y_i \in \mathcal{Y}_T$

[8] Pan & Yang 2010

Definition of Transfer Learning

Transfer Learning

Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{T}_T$ using the knowledge $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.

In case $\mathcal{D}_S \neq \mathcal{D}_T$ holds (the domains are different) at least one of the following is true:

1. $\mathcal{X}_S \neq \mathcal{X}_T$ (feature spaces are different)
2. $P_S(\boldsymbol{X}) \neq P_T(\boldsymbol{X})$ (probability distributions are different)

In case $\mathcal{T}_S \neq \mathcal{T}_T$ holds (the tasks are different) at least one of the following is true:

1. $\mathcal{Y}_S \neq \mathcal{Y}_T$ (label spaces are different)
2. $f_S(\cdot) \neq f_T(\cdot) \Leftrightarrow P_S(\boldsymbol{y}_S|\boldsymbol{X}_S) \neq P_T(\boldsymbol{y}_T|\boldsymbol{X}_T)$ (predictive functions are different)

[8] Pan & Yang 2010

Lecture Outline

Different Setups for Transfer Learning

$\mathcal{T}_S \neq \mathcal{T}_T$

1. Inductive Transfer Learning – Labeled data from $\mathcal{D}_T$ are required to induce the model $f_T(\cdot)$. Specific setups depend on the relatedness between $\mathcal{Y}_S$ and $\mathcal{Y}_T$ and availability of $\boldsymbol{y}_S$.
2. Unsupervised Transfer Learning – $\mathcal{T}_T$ is unsupervised task such as clustering, dimensionality reduction, or density estimation. We assume that $\mathcal{T}_S \neq \mathcal{T}_T$ but also $\mathcal{T}_S \propto \mathcal{T}_T$.

$\mathcal{T}_S = \mathcal{T}_T \wedge \mathcal{D}_S \neq \mathcal{D}_T$

3. Transductive Transfer Learning – We assume that $\boldsymbol{y}_T$ is not available while $\boldsymbol{y}_S$ is. Specific setups depend on the relatedness between $\mathcal{X}_S$ and $\mathcal{X}_T$.

[8] Pan & Yang 2010

Knowledge Transfer Approaches

1. Instance transfer approach – Assumes that certain parts of the data in $\mathcal{D}_S$ can be reused for learning in $\mathcal{D}_T$ by, e.g., instance reweighting or importance sampling.
2. Feature represantation transfer approach – The aim is to learn a good feature representation for the target domain. The knowledge that is transferred from $\mathcal{D}_S$ to $\mathcal{D}_T$ is encoded into the learned feature representation.
3. Parameter transfer approach – It is assumed that $\mathcal{T}_S$ and $\mathcal{T}_T$ share some parameters $\boldsymbol{\theta}$ or prior distributions of the hyperparameters of $f_S(\cdot)$ and $f_T(\cdot)$.
4. Relational knowledge transfer approach – Some relationship among the data in the source and target domains is similar. The knowledge to be transferred is the relationship among the data.

[8] Pan & Yang 2010

Different Approaches in Different Setups

	Inductive Transfer L'rning	Transductive Transfer L'rning	Unsupervised Transfer L'rning
Instance Transfer	SVM, TrAdaBoost	Sample Reweighting	
Feature Repr. Transfer	SVM, Sparse coding Spectral reg., Kernel-based approach	Structural Correspondence Learning (SCL), Kernel Map., co-Clustering, Bridged refinement, ...	Self-Taught Clustering (STC), Transferred Discriminative Analysis (TDA)
Parameter Transfer	Regularization, MT-IVM		
Data Relation Transfer	Transfer via Automatic Mapping and Revision (TAMAR)		

[8] Pan & Yang 2010

Lecture Outline

Definition

Inductive Transfer Learning

Given a source domain $\mathcal{D}_S$ and a learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a learning task $\mathcal{T}_T$, inductive transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{T}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{T}_S \neq \mathcal{T}_T$.

Intuition: Existence of labeled data from $\mathcal{D}_T$ is assumed, i.e., there is $\boldsymbol{D}_T = (\boldsymbol{X}_T, \boldsymbol{y}_T)$. While learning $f_T(\cdot)$ from $\boldsymbol{D}_T$ we try to use knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$ to enhance the performance in $\mathcal{T}_T$.

What is transferred in existing work?
Instances (1); Feature representation (2); Parameters (3); Relational Knowledge (4)

[8] Pan & Yang 2010

Instance Transfer via SVM

Let us have the following example of binary classification problem: $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, $\boldsymbol{x}^{(i)} \in \mathbb{R}^{d \times 1}$ and $y^{(i)} \in \{\pm 1\}$

We want to solve the problem with standard SVM model:

$$\begin{aligned} \underset{\boldsymbol{w}, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}||\boldsymbol{w}||_2^2 + \lambda \sum_{i=1}^{n} \xi^{(i)} \\ \text{subject to} \quad & y^{(i)} \boldsymbol{w}^T \boldsymbol{x}^{(i)} \geq 1 - \xi^{(i)},\ \xi^{(i)} \geq 0,\ i = 1, \ldots, n \end{aligned}$$

where $\boldsymbol{w} \in \mathbb{R}^{d \times 1}$ is the model parameter, $\boldsymbol{\xi} \in \mathbb{R}^{n \times 1}$ are the slack variables, and $\lambda > 0$ is the tradeoff parameter.

Solving the optimization problem we get the decision function:

$$f(\boldsymbol{x}^{(i)}) = \boldsymbol{w}^T \boldsymbol{x}^{(i)} = \sum_{j=1}^{d} w_j x_j^{(i)}$$

Instance Transfer via SVM (cont.)

We can reformulate the problem by adding transfer learning component:

$\mathcal{D}_S = \{\mathcal{X}_S, P_S(\boldsymbol{X}_S)\},\ \mathcal{T}_S = \{\mathcal{Y}_S, f_S(\cdot)\}$

$\mathcal{D}_T = \{\mathcal{X}_T, P_T(\boldsymbol{X}_T)\},\ \mathcal{T}_T = \{\mathcal{Y}_T, f_T(\cdot)\}$

Let us assume that we have both:

$\boldsymbol{D}_S = (\boldsymbol{X}_S, \boldsymbol{y}_S)$

$\boldsymbol{D}_T = (\boldsymbol{X}_T, \boldsymbol{y}_T)$

In addition we assume that $\mathcal{X}_S \approx \mathcal{X}_T$ and $\mathcal{Y}_S = \mathcal{Y}_T$ while $P_S(\boldsymbol{X}_S) \neq P_T(\boldsymbol{X}_T)$

We want to learn $f_T(\cdot)$ (SVM classifier), i.e. the decision function:

$$f(\boldsymbol{x}_T^{(i)}) = \boldsymbol{w}^T \boldsymbol{x}_T^{(i)} = \sum_{j=1}^{d} w_j x_j^{(i)}$$

Instance Transfer via SVM (cont.)

Remember, we want to learn $f_T(\cdot)$ (SVM classifier). What are our options?

1. We can ignore $\mathcal{D}_S$ and $\mathcal{T}_S$ and learn traditional SVM classifier from $\mathcal{D}_T$ and $\mathcal{T}_T$.
   - Is this a good approach? Why?
     - It might be. We do this all the time.
     - But we miss an opportunity to do even better by taking $\mathcal{D}_S$ and $\mathcal{T}_S$ into account.
     - – It is no fun.
2. We can take $\mathcal{D}_S$ and $\mathcal{T}_S$ into account and learn SVM classifier from both $\mathcal{D}_T$ and $\mathcal{T}_T$ and $\mathcal{D}_S$ and $\mathcal{T}_S$.
   - Is this a good approach? Why?
     - $+$ It might be.
     - $+$ It is fun.
     - – The question is how do we do that?

Instance Transfer via SVM (cont.)

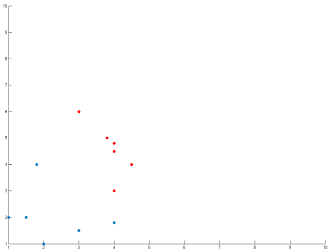
Remember, we want to learn $f_T(\cdot)$ (SVM classifier) and we want to take $\mathcal{D}_S$ and $\mathcal{T}_S$ into account and learn SVM classifier from both $\mathcal{D}_T$ and $\mathcal{T}_T$ and $\mathcal{D}_S$ and $\mathcal{T}_S$. How can we do that?

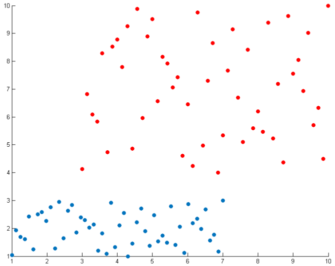
Really simple approach would be to use data from both domains,
$\boldsymbol{D}_S = (\boldsymbol{X}_S, \boldsymbol{y}_S)$ and
$\boldsymbol{D}_T = (\boldsymbol{X}_T, \boldsymbol{y}_T)$,
and train $f_T(\cdot)$ using the following:

$$\begin{array}{ll} \underset{\boldsymbol{w}, \boldsymbol{\xi_T}, \boldsymbol{\xi_S}}{\text{minimize}} & \frac{1}{2}||\boldsymbol{w}||_2^2 + \lambda_1 \sum_{i=1}^{n} \xi_T^{(i)} + \lambda_2 \sum_{i=1}^{m} \xi_S^{(i)} \\ \text{subject to} & y^{(i)} \boldsymbol{w}^T \boldsymbol{x}_S^{(i)} \geq 1 - \xi_S^{(i)},\ \xi_S^{(i)} \geq 0,\ i = 1, \ldots, n, \\ \text{subject to} & y^{(i)} \boldsymbol{w}^T \boldsymbol{x}_T^{(i)} \geq 1 - \xi_T^{(i)},\ \xi_T^{(i)} \geq 0,\ i = 1, \ldots, m \end{array}$$

Is this a good approach? Why?

Instance Transfer via SVM (cont.)





Remember, we want to learn $f_T(\cdot)$ (SVM classifier) and we want to take $\mathcal{D}_S$ and $\mathcal{T}_S$ into account and learn SVM classifier from both $\mathcal{D}_T$ and $\mathcal{T}_T$ and $\mathcal{D}_S$ and $\mathcal{T}_S$. Using $\mathcal{D}_S$ and $\mathcal{T}_S$ directly may not be such a good idea. Can we do something else?

Instance Transfer via SVM (cont.)

The key insight is that some $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$ are helpful for training $f_T(\cdot)$, while others may cause harm.

Thus, we need to select those $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$ that are useful and kick out those that are not.

Effective way to achieve this is to perform instance weighting on $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$ reflecting importance for learning $f_T(\cdot)$.

We can achieve this with only minor changes to the considered model:

$$\begin{aligned}
\underset{\boldsymbol{w},\boldsymbol{\xi_T},\boldsymbol{\xi_S}}{\text{minimize}} \quad & \frac{1}{2}||\boldsymbol{w}||_2^2 + \lambda \sum_{i=1}^{n} \xi_T^{(i)} + \lambda \sum_{i=1}^{m} \rho^{(i)} \xi_S^{(i)} \\
\text{subject to} \quad & y^{(i)} \boldsymbol{w}^T \boldsymbol{x}_S^{(i)} \geq 1 - \xi_S^{(i)},\ \xi_S^{(i)} \geq 0,\ i = 1, \ldots, n, \\
\text{subject to} \quad & y^{(i)} \boldsymbol{w}^T \boldsymbol{x}_T^{(i)} \geq 1 - \xi_T^{(i)},\ \xi_T^{(i)} \geq 0,\ i = 1, \ldots, m
\end{aligned}$$

[1] Aggarwal & Zhai 2012

Instance Transfer via SVM (cont.)

In the presented model $\rho^{(i)}$ is the weight on the data point $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$, which can be estimated via some heuristics or optimization techniques.

For example we can set $\rho^{(i)} = \sigma((\boldsymbol{x}_S^{(i)}, y_S^{(i)}), \boldsymbol{D}_T)$, where:

$$\sigma((\boldsymbol{x}_S^{(i)}, y_S^{(i)}), \boldsymbol{D}_T) = \frac{1}{|\boldsymbol{D}_T|} \sum_{j=1}^{|\boldsymbol{D}_T|} exp\left\{-\beta ||(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) - (\boldsymbol{x}_T^{(j)}, y_T^{(j)})||_2^2\right\}$$

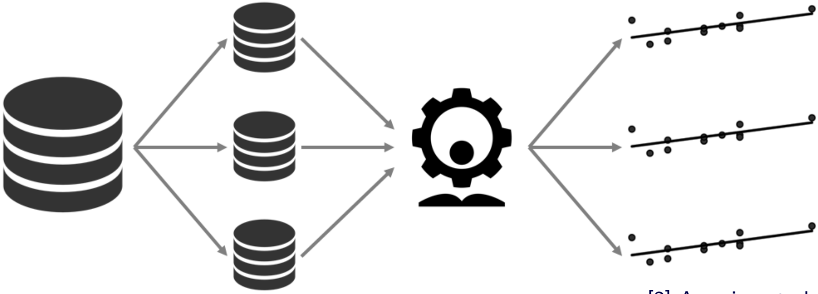
We can see that the only difference between the standard SVM and SVM with instance-based transfer is the loss function $\lambda \sum_{i=1}^{m} \rho^{(i)} \xi_S^{(i)}$ and its corresponding constraints.

The transferred instances $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$ can be support vectors of tranined SVM in $\mathcal{T}_S$ or the whole $\boldsymbol{D}_S$.

[1] Aggarwal & Zhai 2012, [7] Jiang et al. 2008

Feature Repre. Transfer via Supervised Feat. Construction

- Learning multiple related tasks simultaneously can significantly improve performance compared to learning each task independently.
- Sparse feature learning aims at learning a few features common accross the tasks by regularizing within the tasks while keeping them coupled to each other.



[2] Argyriou et al. 2007

Feature Representation Transfer via SFC (cont.)

Traditional Learning Setup

- one learning task $\mathcal{T}$
- dataset $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} \subset \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \subset \mathbb{R}^n$ which is used as $\{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots (\boldsymbol{x}^{(n)}, y^{(n)})\}$
- the goal is to learn function $f(\cdot)$ such that $f(\boldsymbol{x}^{(i)}) \approx y^{(i)}$

Multi-Task Learning Setup

- multiple learning tasks $\boldsymbol{\mathcal{T}} = (\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_t)$
- dataset $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} \subset \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \subset \mathbb{R}^n$ which is used as $\{(\boldsymbol{x}_{\mathcal{T}_j}^{(1)}, y_{\mathcal{T}_j}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_j}^{(2)}, y_{\mathcal{T}_j}^{(2)}), \ldots (\boldsymbol{x}_{\mathcal{T}_j}^{(m)}, y_{\mathcal{T}_j}^{(m)})\}$
- the goal is to learn functions $f_1(\cdot), f_2(\cdot), \ldots f_t(\cdot)$ such that $f_j(\boldsymbol{x}_{\mathcal{T}_j}^{(i)}) \approx y_{\mathcal{T}_j}^{(i)}$

[6] Evgeniou & Pontil 2004

Feature Representation Transfer via SFC (cont.)

- We already know that there is a dataset $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} \subset \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \subset \mathbb{R}^{n}$ which is used as $\{(\boldsymbol{x}_{\mathcal{T}_j}^{(1)}, y_{\mathcal{T}_j}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_j}^{(2)}, y_{\mathcal{T}_j}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_j}^{(m)}, y_{\mathcal{T}_j}^{(m)})\}$
- Specifically, this means that for each task $\mathcal{T}_j$ there is m data points sampled from a distribution P_j over $\boldsymbol{D}$.
- Now, as we have multiple learning tasks $\boldsymbol{\mathcal{T}_j} = \{\mathcal{T}_1, \mathcal{T}_2, \dots \mathcal{T}_t\}$, we also have multiple data sets $\{\boldsymbol{D}_{\mathcal{T}_1}, \boldsymbol{D}_{\mathcal{T}_2}, \dots \boldsymbol{D}_{\mathcal{T}_t}\}$.
- For each dataset $\boldsymbol{D}_{\mathcal{T}_j} = (\boldsymbol{X}_{\mathcal{T}_j}, \boldsymbol{y}_{\mathcal{T}_j})$ there is $\boldsymbol{X}_{\mathcal{T}_j} \subset \mathbb{R}^{m \times d}$ and $\boldsymbol{y}_{\mathcal{T}_j} \subset \mathbb{R}^{m}$.
- This means that the total data available is $\{\{(\boldsymbol{x}_{\mathcal{T}_1}^{(1)}, y_{\mathcal{T}_1}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_1}^{(2)}, y_{\mathcal{T}_1}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_1}^{(m)}, y_{\mathcal{T}_1}^{(m)})\},$ $\{(\boldsymbol{x}_{\mathcal{T}_2}^{(1)}, y_{\mathcal{T}_2}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_2}^{(2)}, y_{\mathcal{T}_2}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_2}^{(m)}, y_{\mathcal{T}_2}^{(m)})\}, \dots$ $\{(\boldsymbol{x}_{\mathcal{T}_t}^{(1)}, y_{\mathcal{T}_t}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_t}^{(2)}, y_{\mathcal{T}_t}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_t}^{(m)}, y_{\mathcal{T}_t}^{(m)})\}\}$

[6] Evgeniou & Pontil 2004

Feature Representation Transfer via SFC (cont.)

The common features are learned by solving an optimization problem, given as follows:

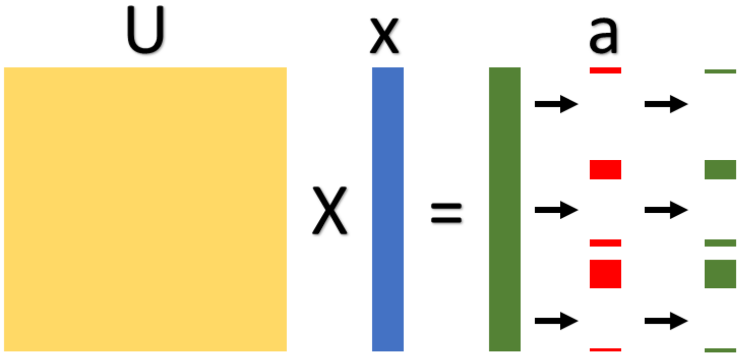
$$\begin{aligned} \underset{A,U}{\text{minimize}} \quad & \sum_{i=1}^{t} \sum_{j=1}^{n} \mathcal{L}(y_{\mathcal{T}_i}^{(j)}, \langle \boldsymbol{a}_{\mathcal{T}_i}, U^T \boldsymbol{x}_{\mathcal{T}_i}^{(j)} \rangle) + \gamma ||A||_{2,1}^2 \\ \text{subject to} \quad & U \in \boldsymbol{O}^d,\ A \in \mathbb{R}^{d \times t} \end{aligned}$$

U:	$d \times d$ matrix with orthonormal $\boldsymbol{u}^{(i)}$ in columns
A:	$d \times t$ matrix with entries $a_{\mathcal{T}_i}^{j}$
$\langle \cdot, \cdot \rangle$	standard inner product
$\|\|A\|\|_{2,1}$	$\|\|A\|\|_2$ over the rows of matrix A and then $\|\|a_{(i)}\|\|_1$ over the resulting elements

$W = UA$ where each column $w^{(i)}$ is a task specific weight vector

[2] Argyriou et al. 2007

Feature Representation Transfer via SFC (cont.)



Feature Rep. Transfer via Unsupervised Feat. Construction

- In self-taught learning we use unlabeled data for improving performance on supervised learning tasks.
- The key assumption is that unlabeled data contain basic patterns that are also present in the data we would like to classify.
- The goal is to learn higher-level feature representation of the inputs using unlabeled data without assuming that the unlabeled data can be assigned with class labels.
- The approach aims to make learning easier and cheaper.
- The inspiration is taken from human learning which is believed to be largely unsupervised.

[9] Raina et al. 2007

Feature Representation Transfer via UFC

Learning Setup

$\boldsymbol{D}_{\mathcal{T}_T} = (\boldsymbol{X}_{\mathcal{T}_T}, \boldsymbol{y}_{\mathcal{T}_T}) = \{(\boldsymbol{x}^{(1)}_{\mathcal{T}_T}, y^{(1)}_{\mathcal{T}_T}), (\boldsymbol{x}^{(2)}_{\mathcal{T}_T}, y^{(2)}_{\mathcal{T}_T}), \ldots (\boldsymbol{x}^{(n)}_{\mathcal{T}_T}, y^{(n)}_{\mathcal{T}_T})\}$
drawn i.i.d in $\mathcal{D}_T = (\mathcal{X}_T, \mathcal{P}_T(\boldsymbol{X}_T))$, where $\boldsymbol{x}^{(i)}_{\mathcal{T}_T} \in \mathbb{R}^d$ and $y^{(i)}_{\mathcal{T}_T} \in \{1, \ldots C\}$

$\boldsymbol{D}_{\mathcal{T}_S} = \boldsymbol{X}_{\mathcal{T}_S} = \{\boldsymbol{x}^{(1)}_{\mathcal{T}_S}, \boldsymbol{x}^{(2)}_{\mathcal{T}_S}, \ldots \boldsymbol{x}^{(n)}_{\mathcal{T}_S}\}$
drawn i.i.d in $\mathcal{D}_S = (\mathcal{X}_S, \mathcal{P}_S(\boldsymbol{X}_S))$, where $\boldsymbol{x}^{(i)}_{\mathcal{T}_S} \in \mathbb{R}^d$

We do not assume that $\mathcal{P}_T(\boldsymbol{X}_T) \approx \mathcal{P}_S(\boldsymbol{X}_S)$.

Our goal is to use $\boldsymbol{D}_{\mathcal{T}_S}$ to improve performance of $f_{\mathcal{T}_T(\cdot)} \in \mathcal{T}_T$.

[9] Raina et al. 2007

Feature Representation Transfer via UFC

First, we solve the following optimization problem on $\boldsymbol{D}_{\mathcal{T}_S}$:

$$\begin{aligned} \underset{\boldsymbol{b},\boldsymbol{a}}{\text{minimize}} \quad & \sum_{i=1}^{n} \left\| \boldsymbol{x}_{\mathcal{T}_S}^{(i)} - \sum_{j=1}^{s} a_j^{(i)} \boldsymbol{b}^{(j)} \right\|_2^2 + \beta \left\| \boldsymbol{a}^{(i)} \right\|_1 \\ \text{subject to} \quad & ||\boldsymbol{b}^{(j)}||_2 \leq 1,\ \forall j \in 1, \ldots s \end{aligned}$$

$\boldsymbol{b}^{(j)}$: basis vector $\boldsymbol{b}^{(j)} \in \mathbb{R}^d$
$\boldsymbol{a}^{(i)}$: activations vector $\boldsymbol{a}^{(i)} \in \mathbb{R}^s$

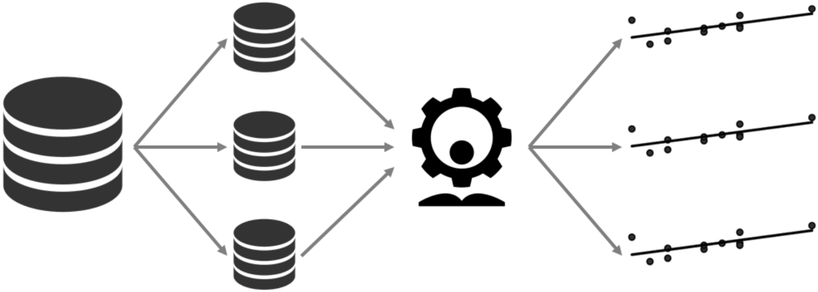
Second, for each training input $(\boldsymbol{x}_{\mathcal{T}_T}^{(i)}, y_{\mathcal{T}_T}^{(i)})$ we compute features $\hat{a}(\cdot) \in \mathbb{R}^d$ by solving the following optimization problem:

$$\underset{\boldsymbol{a}^{(i)}}{\text{minimize}} \quad \left\| \boldsymbol{x}_{\mathcal{T}_T} - \sum_{j=1}^{s} a_j^{(i)} \boldsymbol{b}^{(j)} \right\|_2^2 + \beta \left\| \boldsymbol{a}^{(i)} \right\|_1$$

[9] Raina et al. 2007

Parameter Transfer via Regularized Multi-Task Learning

- In certain situations it is necessary to create more than one statistical model.
- If the tasks are related it may be advantegous to learn the models simultaneously.



[6] Evgeniou & Pontil 2004

Param. Transfer via Regularized Multi-Task L'rning (cont.)

Traditional Learning Setup

- one learning task $\mathcal{T}$
- dataset $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} \subset \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \subset \mathbb{R}^{n}$ which is used as $\{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots (\boldsymbol{x}^{(n)}, y^{(n)})\}$
- the goal is to learn function $f(\cdot)$ such that $f(\boldsymbol{x}^{(i)}) \approx y^{(i)}$

Multi-Task Learning Setup

- multiple learning tasks $\boldsymbol{\mathcal{T}} = (\mathcal{T}_1, \mathcal{T}_2, \dots \mathcal{T}_t)$
- dataset $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} \subset \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \subset \mathbb{R}^{n}$ which is used as $\{(\boldsymbol{x}_{\mathcal{T}_j}^{(1)}, y_{\mathcal{T}_j}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_j}^{(2)}, y_{\mathcal{T}_j}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_j}^{(m)}, y_{\mathcal{T}_j}^{(m)})\}$
- the goal is to learn functions $f_1(\cdot), f_2(\cdot), \dots f_t(\cdot)$ such that $f_j(\boldsymbol{x}_{\mathcal{T}_j}^{(i)}) \approx y_{\mathcal{T}_j}^{(i)}$

[6] Evgeniou & Pontil 2004

Param. Transfer via Regularized Multi-Task L'rning (cont.)

- We already know that there is a dataset $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} \subset \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \subset \mathbb{R}^n$ which is used as $\{(\boldsymbol{x}_{\mathcal{T}_j}^{(1)}, y_{\mathcal{T}_j}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_j}^{(2)}, y_{\mathcal{T}_j}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_j}^{(m)}, y_{\mathcal{T}_j}^{(m)})\}$
- Specifically, this means that for each task $\mathcal{T}_j$ there is m data points sampled from a distribution P_j over $\boldsymbol{D}$.
- Now, as we have multiple learning tasks $\boldsymbol{\mathcal{T}_j} = \{\mathcal{T}_1, \mathcal{T}_2, \dots \mathcal{T}_t\}$, we also have multiple data sets $\{\boldsymbol{D}_{\mathcal{T}_1}, \boldsymbol{D}_{\mathcal{T}_2}, \dots \boldsymbol{D}_{\mathcal{T}_t}\}$.
- For each dataset $\boldsymbol{D}_{\mathcal{T}_j} = (\boldsymbol{X}_{\mathcal{T}_j}, \boldsymbol{y}_{\mathcal{T}_j})$ there is $\boldsymbol{X}_{\mathcal{T}_j} \subset \mathbb{R}^{m \times d}$ and $\boldsymbol{y}_{\mathcal{T}_j} \subset \mathbb{R}^m$.
- This means that the total data available is $\{\{(\boldsymbol{x}_{\mathcal{T}_1}^{(1)}, y_{\mathcal{T}_1}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_1}^{(2)}, y_{\mathcal{T}_1}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_1}^{(m)}, y_{\mathcal{T}_1}^{(m)})\}$, $\{(\boldsymbol{x}_{\mathcal{T}_2}^{(1)}, y_{\mathcal{T}_2}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_2}^{(2)}, y_{\mathcal{T}_2}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_2}^{(m)}, y_{\mathcal{T}_2}^{(m)})\}, \dots$ $\{(\boldsymbol{x}_{\mathcal{T}_t}^{(1)}, y_{\mathcal{T}_t}^{(1)}), (\boldsymbol{x}_{\mathcal{T}_t}^{(2)}, y_{\mathcal{T}_t}^{(2)}), \dots (\boldsymbol{x}_{\mathcal{T}_t}^{(m)}, y_{\mathcal{T}_t}^{(m)})\}\}$

[6] Evgeniou & Pontil 2004

Param. Transfer via Regularized Multi-Task L'rning (cont.)

Let us recall the standard SVM model:

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}||\boldsymbol{w}||_2^2 + \lambda \sum_{i=1}^{n} \xi^{(i)} \\ \text{subject to} \quad & y^{(i)}\boldsymbol{w}^T\boldsymbol{x}^{(i)} \geq 1 - \xi^{(i)},\ \xi^{(i)} \geq 0,\ i = 1, \ldots, n \end{aligned}$$

where $\boldsymbol{w} \in \mathbb{R}^{d\times 1}$ is the model parameter, $\boldsymbol{\xi} \in \mathbb{R}^{n\times 1}$ are the slack variables, and $\lambda > 0$ is the tradeoff parameter.

Solving the optimization problem we get the decision function:

$$f(\boldsymbol{x}^{(i)}) = \boldsymbol{w}^T\boldsymbol{x}^{(i)} = \sum_{j=1}^{d} w_j x_j^{(i)}$$

We are (again!) going to augment this model for the regularized multi-task learning problem.

Param. Transfer via Regularized Multi-Task L'rning (cont.)

For the base SVM model we had this binary classification problem:
$\mathcal{T} : \boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, $\boldsymbol{x}^{(i)} \in \mathbb{R}^{d \times 1}$ and $y^{(i)} \in \{\pm 1\}$

For multi-task learning we are seeking a single (!!!) model to handle multiple binary classification problems:

$\mathcal{T}_1 : \boldsymbol{D}_{\mathcal{T}_1} = (\boldsymbol{X}_{\mathcal{T}_1}, \boldsymbol{y}_{\mathcal{T}_1})$, where $\boldsymbol{X}_{\mathcal{T}_1} \in \mathbb{R}^{n \times d}$, $\boldsymbol{x}_{\mathcal{T}_1}^{(i)} \in \mathbb{R}^{d \times 1}$ and $y_{\mathcal{T}_1}^{(i)} \in \{\pm 1\}$

$\mathcal{T}_2 : \boldsymbol{D}_{\mathcal{T}_2} = (\boldsymbol{X}_{\mathcal{T}_2}, \boldsymbol{y}_{\mathcal{T}_2})$, where $\boldsymbol{X}_{\mathcal{T}_2} \in \mathbb{R}^{n \times d}$, $\boldsymbol{x}_{\mathcal{T}_2}^{(i)} \in \mathbb{R}^{d \times 1}$ and $y_{\mathcal{T}_2}^{(i)} \in \{\pm 1\}$

...

$\mathcal{T}_t : \boldsymbol{D}_{\mathcal{T}_t} = (\boldsymbol{X}_{\mathcal{T}_t}, \boldsymbol{y}_{\mathcal{T}_t})$, where $\boldsymbol{X}_{\mathcal{T}_t} \in \mathbb{R}^{n \times d}$, $\boldsymbol{x}_{\mathcal{T}_t}^{(i)} \in \mathbb{R}^{d \times 1}$ and $y_{\mathcal{T}_t}^{(i)} \in \{\pm 1\}$

Param. Transfer via Regularized Multi-Task L'rning (cont.)

Solving the optimization problem represented by the base SVM model we get the single decision function:

$$f(\boldsymbol{x}^{(i)}) = \boldsymbol{w}^T \boldsymbol{x}^{(i)} = \sum_{j=1}^{d} w_j x_j^{(i)}$$

In multi-task learning we expect to get multiple decision functions (one for each task):

$$\boldsymbol{F} = \{f_{\mathcal{T}_1}(\cdot), f_{\mathcal{T}_2}(\cdot), \ldots f_{\mathcal{T}_t}(\cdot)\}, \text{ where}$$

$$f_{\mathcal{T}_j}(\boldsymbol{x}_{\mathcal{T}_j}^{(i)}) = \boldsymbol{w}_{\mathcal{T}_j}^T \boldsymbol{x}_{\mathcal{T}_j}^{(i)} = \sum_{k=1}^{d} w_{\mathcal{T}_j,k} x_{\mathcal{T}_j,k}^{(i)}$$

Param. Transfer via Regularized Multi-Task L'rning (cont.)

Before we get to the proposed model for regularized multi-task learning we need to introduce the key insight:

- Frameworks and methods for multi-task learning are often based on some formal definition of the notion of relatedness of the tasks.
- The relatedness is formalized through the design of a multi-task learning method. For example, we can assume that all parameters $\boldsymbol{w}_{\mathcal{T}_i}$ come from a particular probability distribution such as a Gaussian.
- This implies that all $\boldsymbol{w}_{\mathcal{T}_i}$ are "close" to some mean parameter vector $\boldsymbol{w}_0$.
- Therefore, the assumption is that all $\boldsymbol{w}_{\mathcal{T}_i}$ can be written, for every $\mathcal{T}_i \in \boldsymbol{\mathcal{T}}$, as:

$$\boldsymbol{w}_{\mathcal{T}_i} = \boldsymbol{w}_0 + \boldsymbol{v}_{\mathcal{T}_i}$$

[6] Evgeniou & Pontil 2004

Param. Transfer via Regularized Multi-Task L'rning (cont.)

We can estimate all $\boldsymbol{v}_{\mathcal{T}_i}$ as well as the (common) $\boldsymbol{w}_0$ simultaneously by solving the following optimization problem (analogous to traditional SVM):

$$\begin{aligned} \underset{\boldsymbol{w}_0, \boldsymbol{v}_{\mathcal{T}_i}, \boldsymbol{\xi}_{\mathcal{T}_i}}{\text{minimize}} \quad & \sum_{i=1}^{t} \sum_{j=1}^{m} \xi_{\mathcal{T}_i}^{(j)} + \frac{\lambda_1}{t} \sum_{i=1}^{t} ||\boldsymbol{v}_{\mathcal{T}_i}||^2 + \lambda_2 ||\boldsymbol{w}_0||^2 \\ \text{subject to} \quad & y_{\mathcal{T}_i}^{(j)} (\boldsymbol{w}_0 + \boldsymbol{v}_{\mathcal{T}_i})^T \boldsymbol{x}_{\mathcal{T}_i}^{(j)} \geq 1 - \xi_{\mathcal{T}_i}^{(j)},\ \xi_{\mathcal{T}_i}^{(j)} \geq 0, \\ & i = 1, \ldots, t \text{ and } j = 1, \ldots, m \end{aligned}$$

Solving the optimization problem yields:

$$\boldsymbol{F} = \{ f_{\mathcal{T}_1}(\cdot), f_{\mathcal{T}_2}(\cdot), \ldots f_{\mathcal{T}_t}(\cdot) \}, \text{ where}$$

$$f_{\mathcal{T}_i}(\boldsymbol{x}_{\mathcal{T}_i}^{(j)}) = (\boldsymbol{w}_0 + \boldsymbol{v}_{\mathcal{T}_i})^T \boldsymbol{x}_{\mathcal{T}_i}^{(j)} = \boldsymbol{w}_{\mathcal{T}_i}^T \boldsymbol{x}_{\mathcal{T}_i}^{(j)} = \sum_{k=1}^{d} w_{\mathcal{T}_i,k} x_{\mathcal{T}_i,k}^{(j)}$$

[6] Evgeniou & Pontil 2004

Lecture Outline

Definition

Unsupervised Transfer Learning

Given a source domain $\mathcal{D}_S$ and a learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a learning task $\mathcal{T}_T$, unsupervised transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{T}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{T}_S \neq \mathcal{T}_T$ and $\mathcal{Y}_S$ and $\mathcal{Y}_T$ are not observable.

Intuition: It is assumed that no labeled data from $\mathcal{D}_S$ or $\mathcal{D}_T$ are available. While learning $f_T(\cdot)$ from $\boldsymbol{X}_T$ we try to use knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$ to enhance the performance in $\mathcal{T}_T$.

What is transferred in existing work?
Feature representation (2)

[8] Pan & Yang 2010

Feature Representation Transfer via Self-Taught Clustering

- Clustering aims at partitioning objects into groups, so that the objects in the same groups are relatively similar, while the objects in different groups are relatively dissimilar.
- Self-taught clustering uses unlabeled data from $\boldsymbol{D}_{\mathcal{T}_S}$ to enhance clustering performance on unlabeled data from $\boldsymbol{D}_{\mathcal{T}_T}$.
- Different objects may share some common features.
- Self-taught clustering exploits the commonality through co-clustering. It means that clustering operations on both, $\boldsymbol{D}_{\mathcal{T}_S}$ and $\boldsymbol{D}_{\mathcal{T}_T}$, are performed together.
- Co-clustering algorithm which minimizes loss in mutual information before and after co-clustering is used.

[5] Dai et al. 2008

Feature Rep. Transfer via Self-Taught Clustering (cont.)

Learning Setup

$$\boldsymbol{D}_{\mathcal{T}_T} = (\boldsymbol{X}_{\mathcal{T}_T}, \boldsymbol{Z}_{\mathcal{T}_T}) = \{(\boldsymbol{x}^{(1)}_{\mathcal{T}_T}, \boldsymbol{z}^{(1)}_{\mathcal{T}_T}), (\boldsymbol{x}^{(2)}_{\mathcal{T}_T}, \boldsymbol{z}^{(2)}_{\mathcal{T}_T}), \ldots (\boldsymbol{x}^{(n)}_{\mathcal{T}_T}, \boldsymbol{z}^{(n)}_{\mathcal{T}_T})\}$$

$$\boldsymbol{D}_{\mathcal{T}_S} = (\boldsymbol{X}_{\mathcal{T}_S}, \boldsymbol{Z}_{\mathcal{T}_S}) = \{(\boldsymbol{x}^{(1)}_{\mathcal{T}_S}, \boldsymbol{z}^{(1)}_{\mathcal{T}_S}), (\boldsymbol{x}^{(2)}_{\mathcal{T}_S}, \boldsymbol{z}^{(2)}_{\mathcal{T}_S}), \ldots (\boldsymbol{x}^{(n)}_{\mathcal{T}_S}, \boldsymbol{z}^{(n)}_{\mathcal{T}_S})\}$$

We assume that:

- $\boldsymbol{X}_{\mathcal{T}_T}$ is drawn i.i.d in $\mathcal{D}_T = (\mathcal{X}_T, \mathcal{P}_T(\boldsymbol{X}_T))$, where $\boldsymbol{x}^{(i)}_{\mathcal{T}_T} \in \mathbb{R}^{d_1}$
- $\boldsymbol{X}_{\mathcal{T}_S}$ is drawn i.i.d in $\mathcal{D}_S = (\mathcal{X}_S, \mathcal{P}_T(\boldsymbol{X}_S))$, where $\boldsymbol{x}^{(i)}_{\mathcal{T}_S} \in \mathbb{R}^{d_2}$
- $\boldsymbol{Z}_{\mathcal{T}_T}$ is drawn i.i.d in $\mathcal{Z}_T = (\mathcal{Z}_T, \mathcal{P}_T(\boldsymbol{Z}_T))$, where $\boldsymbol{z}^{(i)}_{\mathcal{T}_T} \in \mathbb{R}^{d_3}$
- $\boldsymbol{Z}_{\mathcal{T}_S}$ is drawn i.i.d in $\mathcal{Z}_S = (\mathcal{Z}_S, \mathcal{P}_T(\boldsymbol{Z}_S))$, where $\boldsymbol{z}^{(i)}_{\mathcal{T}_S} \in \mathbb{R}^{d_3}$

$\mathcal{X}_T \neq \mathcal{X}_S$ and $\mathcal{P}_T(\boldsymbol{X}_T) \neq \mathcal{P}_S(\boldsymbol{X}_S)$
$\mathcal{Z}_T = \mathcal{Z}_S$ and $\mathcal{P}_T(\boldsymbol{Z}_T) \neq \mathcal{P}_S(\boldsymbol{Z}_S)$

[5] Dai et al. 2008

Feature Rep. Transfer via Self-Taught Clustering (cont.)

Toy Example

$\boldsymbol{D}_{\mathcal{T}_T}^{n\times 2} = (\boldsymbol{X}_{\mathcal{T}_T}, \boldsymbol{Z}_{\mathcal{T}_T})$; $\boldsymbol{D}_{\mathcal{T}_S}^{m\times 2} = (\boldsymbol{X}_{\mathcal{T}_S}, \boldsymbol{Z}_{\mathcal{T}_S})$

$\boldsymbol{X}_{\mathcal{T}_T}^{n\times 1} \in \{1, 2, 3\}$; $\boldsymbol{X}_{\mathcal{T}_S}^{m\times 1} \in \{a, b, c\}$; $\boldsymbol{Z}_{\mathcal{T}_T}^{n\times 1}, \boldsymbol{Z}_{\mathcal{T}_S}^{m\times 1} \in \{\alpha, \beta, \gamma\}$

	p$(Z_{\mathcal{T}_T} = \alpha)$	p$(Z_{\mathcal{T}_T} = \beta)$	p$(Z_{\mathcal{T}_T} = \gamma)$
$p(X_{\mathcal{T}_T} = 1)$	0.2	0.0	0.2
$p(X_{\mathcal{T}_T} = 2)$	0.0	0.2	0.0
$p(X_{\mathcal{T}_T} = 3)$	0.0	0.2	0.2

	p$(Z_{\mathcal{T}_S} = \alpha)$	p$(Z_{\mathcal{T}_S} = \beta)$	p$(Z_{\mathcal{T}_S} = \gamma)$
$p(X_{\mathcal{T}_S} = a)$	0.4	0.0	0.0
$p(X_{\mathcal{T}_S} = b)$	0.0	0.1	0.1
$p(X_{\mathcal{T}_S} = c)$	0.0	0.4	0.0

[5] Dai et al. 2008

Feature Rep. Transfer via Self-Taught Clustering (cont.)

	$p(Z_{\mathcal{T}_T} = \alpha)$	$p(Z_{\mathcal{T}_T} = \beta)$	$p(Z_{\mathcal{T}_T} = \gamma)$
$p(X_{\mathcal{T}_T} = 1)$	0.2	0.0	0.2
$p(X_{\mathcal{T}_T} = 2)$	0.0	0.2	0.0
$p(X_{\mathcal{T}_T} = 3)$	0.0	0.2	0.2

Let us cluster the dataset described by the full-joint above in the following way:

- clustering on $X_{\mathcal{T}_T}$ is $\tilde{X}_{\mathcal{T}_T} = \{\tilde{x}_1 = \{x_1, x_2\}, \tilde{x}_2 = \{x_3\}\}$
- clustering on $Z_{\mathcal{T}_T}$ is $\tilde{Z}_{\mathcal{T}_T} = \{\tilde{z}_\alpha = \{z_\alpha, z_\beta\}, \tilde{z}_\beta = \{z_\gamma\}\}$

This gives us the new full-joint:

	$p(\tilde{Z}_{\mathcal{T}_T} = \alpha)$	$p(\tilde{Z}_{\mathcal{T}_T} = \beta)$
$p(\tilde{X}_{\mathcal{T}_T} = 1)$	0.4	0.2
$p(\tilde{X}_{\mathcal{T}_T} = 2)$	0.2	0.2

How can we evaluate the quality of clustering?

[5] Dai et al. 2008

Feature Rep. Transfer via Self-Taught Clustering (cont.)

One of the techniques used to measure how good is the clustering we obtained is loss in mutual information between instances and features before and after clustering.

$$\mathcal{J}(\tilde{\boldsymbol{X}}_{\mathcal{T}_T}, \tilde{\boldsymbol{Z}}_{\mathcal{T}_T}) = I(\boldsymbol{X}_{\mathcal{T}_T}, \boldsymbol{Z}_{\mathcal{T}_T}) - I(\tilde{\boldsymbol{X}}_{\mathcal{T}_T}, \tilde{\boldsymbol{Z}}_{\mathcal{T}_T})$$

where $I(\boldsymbol{X}, \boldsymbol{Z}) = \sum_{i=1}^{n} \sum_{j=1}^{m} p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(j)}) log \frac{p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(j)})}{p(\boldsymbol{x}^{(i)}) p(\boldsymbol{z}^{(j)})}$

In self-taught clustering we (co-)cluster on $\boldsymbol{X}_{\mathcal{T}_T}$ and $\boldsymbol{X}_{\mathcal{T}_S}$ simultaneously, while the two co-clusters share the same features clustering on $\boldsymbol{Z}_{\mathcal{T}_T}$ and $\boldsymbol{Z}_{\mathcal{T}_S}$.

$$\mathcal{J}(\tilde{\boldsymbol{X}}_{\mathcal{T}_T}, \tilde{\boldsymbol{X}}_{\mathcal{T}_S}, \tilde{\boldsymbol{Z}}) = \\ I(\boldsymbol{X}_{\mathcal{T}_T}, \boldsymbol{Z}_{\mathcal{T}_T}) - I(\tilde{\boldsymbol{X}}_{\mathcal{T}_T}, \tilde{\boldsymbol{Z}}) + \lambda \Big[I(\boldsymbol{X}_{\mathcal{T}_S}, \boldsymbol{Z}_{\mathcal{T}_S}) - I(\tilde{\boldsymbol{X}}_{\mathcal{T}_S}, \tilde{\boldsymbol{Z}}) \Big]$$

[5] Dai et al. 2008

Lecture Outline

Definition

Transductive Transfer Learning

Given a source domain $\mathcal{D}_S$ and a learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a learning task $\mathcal{T}_T$, transductive transfer learning aims to improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{T}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$. In addition, some unlabeled target-domain data must be available at training time.

Intuition: Since $\mathcal{T}_S$ and $\mathcal{T}_T$ are the same, in order to obtain $f_T(\cdot)$ we can adapt predictive function $f_S(\cdot)$ for use in $\mathcal{T}_T$ on the data from $\mathcal{D}_T$.

What is transferred in existing work?
Instances (1); Feature representation (2)

[8] Pan & Yang 2010

Instance Transfer via Sample Reweighting

Learning Setup

$\boldsymbol{D}_{\mathcal{T}_T} = \boldsymbol{X}_{\mathcal{T}_T} = \{\boldsymbol{x}^{(1)}_{\mathcal{T}_T}, \boldsymbol{x}^{(2)}_{\mathcal{T}_T}, \ldots \boldsymbol{x}^{(n)}_{\mathcal{T}_T}\}$ drawn i.i.d in $\mathcal{D}_T = (\mathcal{X}_T, \mathcal{P}_T(\boldsymbol{X}_T))$, where $\boldsymbol{x}^{(i)}_{\mathcal{T}_T} \in \mathbb{R}^d$ and $y^{(i)}_{\mathcal{T}_T} \in \{1, \ldots C\}$

$\boldsymbol{D}_{\mathcal{T}_S} = (\boldsymbol{X}_{\mathcal{T}_S}, \boldsymbol{y}_{\mathcal{T}_S}) = \{(\boldsymbol{x}^{(1)}_{\mathcal{T}_S}, y^{(1)}_{\mathcal{T}_S}), (\boldsymbol{x}^{(2)}_{\mathcal{T}_S}, y^{(2)}_{\mathcal{T}_S}), \ldots (\boldsymbol{x}^{(n)}_{\mathcal{T}_S}, y^{(n)}_{\mathcal{T}_S})\}$ drawn i.i.d in $\mathcal{D}_S = (\mathcal{X}_S, \mathcal{P}_S(\boldsymbol{X}_S))$, where $\boldsymbol{x}^{(i)}_{\mathcal{T}_S} \in \mathbb{R}^d$

We do not assume that $\mathcal{P}_T(\boldsymbol{X}_T) \approx \mathcal{P}_S(\boldsymbol{X}_S)$.

Our goal is to use $\boldsymbol{D}_{\mathcal{T}_S}$ to construct $f_{\mathcal{T}_T(\cdot)} \in \mathcal{T}_T$.

Instance Transfer via Sample Reweighting (cont.)

Let us recall the standard SVM model:

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}||\boldsymbol{w}||_2^2 + \lambda \sum_{i=1}^{n} \xi^{(i)} \\ \text{subject to} \quad & y^{(i)} \boldsymbol{w}^T \boldsymbol{x}^{(i)} \geq 1 - \xi^{(i)},\ \xi^{(i)} \geq 0,\ i = 1, \ldots, n \end{aligned}$$

where $\boldsymbol{w} \in \mathbb{R}^{d \times 1}$ is the model parameter, $\boldsymbol{\xi} \in \mathbb{R}^{n \times 1}$ are the slack variables, and $\lambda > 0$ is the tradeoff parameter.

Solving the optimization problem we get the decision function:

$$f(\boldsymbol{x}^{(i)}) = \boldsymbol{w}^T \boldsymbol{x}^{(i)} = \sum_{j=1}^{d} w_j x_j^{(i)}$$

We are (again!) going to augment this model for the regularized multi-task learning problem.

Instance Transfer via Sample Reweighting (cont.)

The key insight in inductive transfer learning was that some $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$ are helpful for training $f_T(\cdot)$, while others may cause harm.

Thus, we need to select those $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$ that are useful and kick out those that are not.

Effective way to achieve this is to perform instance weighting on $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$ reflecting importance for learning $f_T(\cdot)$.

We can achieve this with only minor changes to the base model:

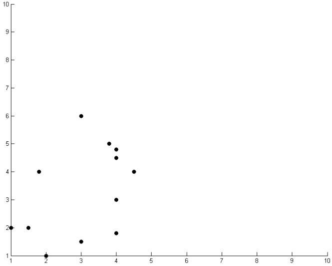
$$\begin{aligned} \underset{\boldsymbol{w}, \boldsymbol{\xi}_S}{\text{minimize}} \quad & \frac{1}{2}||\boldsymbol{w}||_2^2 + \lambda \sum_{i=1}^{n} \rho^{(i)} \xi_S^{(i)} \\ \text{subject to} \quad & y^{(i)} \boldsymbol{w}^T \boldsymbol{x}_S^{(i)} \geq 1 - \xi_S^{(i)},\ \xi_S^{(i)} \geq 0,\ i = 1, \ldots, n, \end{aligned}$$

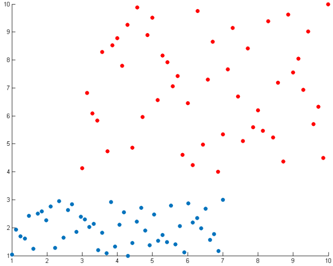
Instance Transfer via Sample Reweighting (cont.)

In the presented model $\rho^{(i)}$ is the weight on the data point $(\boldsymbol{x}_S^{(i)}, y_S^{(i)}) \in \boldsymbol{D}_S$.

For example we can set $\rho^{(i)} = \sigma(\boldsymbol{x}_S^{(i)}, \boldsymbol{D}_T)$, where:

$$\sigma(\boldsymbol{x}_S^{(i)}, \boldsymbol{D}_T) = \frac{1}{|\boldsymbol{D}_T|} \sum_{j=1}^{|\boldsymbol{D}_T|} exp\left\{-\beta ||\boldsymbol{x}_S^{(i)} - \boldsymbol{x}_T^{(j)}||_2^2\right\}$$





[7] Jiang et al. 2008

Lecture Outline

Applications of Transfer Learning

- learning text data across domains
- use of structural correspondence learning (SCL) for solving NLP problems
- use of transductive transfer learning methods to solve named entity recognition (NER) problems
- learning relational action models across domains in automated planning
- use of inductive transfer for learning multiple conceptually related classifiers for computer aided design (CAD)
- use of transfer learning in cross-language classification problem
- use of transfer learning for collaborative filtering
- use of transfer learning for estimation of Wi-fi client's location
- use of transfer learning for sentiment analysis

[8] Pan & Yang 2010

Datasets and Toolboxes

Publicly Available Datasets

- Text – 20 Newsgroups, SRAA and Reuters–21578.
- E-mail
- WiFi – Data collected inside a building for localization purposes in two different time periods.
- Sen – Product reviews from Amazon (four product types, i.e. domains) containing star ratings.

Transfer Learning Resources

There is a hub page pointing to a large number of research papers and code implementations related to transfer learning provided by the Hong Kong University of Science and Technology. It is available at `http://www.cse.ust.hk/TL/`.

[8] Pan & Yang 2010

Evaluation of Selected Applications

Data Set (reference)	Source v.s. Target	Baselines		TL Methods	
20 Newsgroups$_1$ ([6])		**SVM**		**TrAdaBoost**	
ACC (unit: %)	rec v.s. talk	87.3%		92.0%	
	rec v.s. sci	83.6%		90.3%	
	sci v.s. talk	82.3%		87.5%	
20 Newsgroups$_2$ ([84])		**SVM**		**TrAdaBoost**	**AcTraK**
ACC (unit: %)	rec v.s. talk	60.2%		72.3%	75.4%
	rec v.s. sci	59.1%		67.4%	70.6%
	comp v.s. talk	53.6%		74.4%	80.9%
	comp v.s. sci	52.7%		57.3%	78.0%
	comp v.s. rec	49.1%		77.2%	82.1%
	sci v.s. talk	57.6%		71.3%	75.1%
20 Newsgroups$_3$ ([49])		**SVM**	**LR**	**pLWE**	**LWE**
ACC (unit: %)	comp v.s. sci	71.18%	73.49%	78.72%	97.44%
	rec v.s. talk	68.24%	72.17%	72.17%	99.23%
	rec v.s. sci	78.16%	78.85%	88.45%	98.23%
	sci v.s. talk	75.77%	79.04%	83.30%	96.92%
	comp v.s. rec	81.56%	83.34%	91.93%	98.16%
	comp v.s. talk	93.89%	91.76%	96.64%	98.90%
Sentiment Classification ([8])		**SGD**		**SCL**	**SCL-MI**
ACC (unit: %)	DVD v.s. book	72.8%		76.8%	79.7%
	electronics v.s. book	70.7%		75.4%	75.4%
	kitchen v.s. book	70.9%		66.1%	68.6%
	book v.s. DVD	77.2%		74.0%	75.8%
	electronics v.s. DVD	70.6%		74.3%	76.2%
	kitchen v.s. DVD	72.7%		75.4%	76.9%
	book v.s. electronics	70.8%		77.5%	75.9%
	DVD v.s. electronics	73.0%		74.1%	74.1%
	kitchen v.s. electronics	82.7%		83.7%	86.8%
	book v.s. kitchen	74.5%		78.7%	78.9%
	DVD v.s. kitchen	74.0%		79.4%	81.4%
	electronics v.s. kitchen	84.0%		84.4%	85.9%
WiFi Localization ([67])		**RLSR**	**PCA**	**KMM**	**TCA**
AED (unit: meter)	Time **A** v.s. Time **B**	6.52	3.16	5.51	2.37

[8] Pan & Yang 2010

Lecture Outline

Negative Transfer

- Negative transfer happens when the transfer of knowledge from $\mathcal{D}_S$ or $\mathcal{T}_S$ contributes to the reduced performance in $\mathcal{T}_T$.
- If tasks $\mathcal{T}_S$ and $\mathcal{T}_T$ are too dissimilar, then brute-force transfer may hurt the performance in $\mathcal{T}_T$.
- It is important to analyze relatedness among $\mathcal{T}_S$ and $\mathcal{T}_T$ and $\mathcal{D}_S$ and $\mathcal{D}_T$.

  Example methods:
  - Similarity of $\mathcal{T}_S$ and $\mathcal{T}_T$ defined on the basis of similarity between the example generating distributions that underlie the tasks.
  - Similarity of $\mathcal{T}_S$ and $\mathcal{T}_T$ defined on the basis of introduction of higher level task characteristics, that is, features that are known beforehand.

[8] Pan & Yang 2010, [3] Bakker & Heskes 2003, [4] Ben-David & Schuller 2003

Lecture Summary

Intuition and Motivation

Framework Introduction

Categorization

Inductive Transfer Learning

Unsupervised Transfer Learning

Transductive Transfer Learning

Applications

Limitations

References I

- Aggarwal, Charu & Zhai, ChengXiang (eds.). *Mining Text Data*. Springer, 2012.
- Argyriou, Andreas, Evgeniou, Theodoros & Pontil, Massimiliano. Multi-task Feature Learning. *Advances in Neural Information Processing Systems*. MIT Press, 2007.
- Bakker, Bart & Heskes, Tom. Task Clustering and Gating for Bayesian Multitask Learning. *Journal of Machine Learning Research*, 4:83–99, 2003.
- Ben-David, Shai & Schuller, Reba. Exploiting Task Relatedness for Multiple Task Learning. *Lecture Notes in Computer Science*, 2777:567–580, 2003.
- Dai, Wenyuan, Yan, Qiang, Xue, Gui-Rong & Yu, Yong. Self-taught clustering. *ICML '08*. New York, NY, USA, 2008.

References II

- Evgeniou, Theodoros & Pontil, Massimiliano. Regularized Multi–Task Learning. *KDD'04*. Seattle, WSH, USA, 2004.
- Jiang, Wei, Zavesky, Eric, Chang, Shih-Fu & Loui, Alex. Cross-Domain Learning Methods for High-Level Visual Concept Classification. *15th IEEE International Conference on Image Processing*, San Diego, California, USA, 2008.
- Pan, Sinno Jialin & Yang, Qiang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- Raina, Rajat, Battle, Alexis, Lee, Honglak, Packer, Benjamin and Ng, Andrew Y. Self-taught Learning: Transfer Learning from Unlabeled Data. *ICML*. Corvallis, OR, USA, 2007.
- Perkins, David N. & Salomon, Gavriel. *Transfer of Learning*. 08-11-2014 [online]. 1992.

Thank you!

Questions, comments and suggestions are welcome now or any time at jas438@pitt.edu.