A Survey on Transfer Learning – summary

Matěj Nikl

August 6, 2016

As the title suggests, this paper focuses on transfer learning. A few definitions:

- a **Domain** $\mathcal{D} = \{\mathcal{X}, P(X)\}$
  - a feature space $\mathcal{X}$
  - a marginal probability distribution $P(X)$, where $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$
  - $\mathcal{D}_S$, $\mathcal{D}_T$ – source and target domains
- a **Task** $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ (given a domain $\mathcal{D}$)
  - a label space $\mathcal{Y}$
  - an objective predictive function $f(\cdot)$, which is not observed but can be learned from the training data
  - $\mathcal{T}_S$, $\mathcal{T}_T$ – source and target tasks
- **Training Data** consist of pairs $\{x_i, y_i\}$
  - $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$
  - D_S, D_T – source and target data with n_S and n_T instances each
  - in most cases $0 \leq n_T \ll n_S$
- **Transfer Learning** – given $\mathcal{D}_S$, $\mathcal{T}_S$, $\mathcal{D}_T$ and $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$
  - when $\mathcal{D}_S = \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$, the learning problem becomes a traditional machine learning problem

In other words, transfer learning aims to extract a knowledge from one or more source tasks (and their domains) and applies the knowledge to a target task.

One should not confuse transfer learning with multi-task learning – multi-task learning tries to learn all of the source and target tasks simultaneously (thus treating them all equally), transfer learning cares the most about the target task.

There are three main research issues:

1. **What** to transfer – which part of the knowledge can be transferred across domains or tasks – some of it may be specific for individual domains or tasks, some may be common
2. **How** to transfer – learning algorithms need to be developed to transfer the knowledge
3. **When** to transfer – in which situations should transferring be done as well as in which it should not be done – when source and target domains are not related to each other, transferring the knowledge may even hurt the performance – a phenomenon called *negative transfer*

Based on the definition of transfer learning, its relationship can be summarized into three sub-settings:

1. **Inductive Transfer Learning**
   - $\mathcal{T}_S \neq \mathcal{T}_T$
   - a few labeled data in $\mathcal{D}_T$ are required to *induce* $f_T(\cdot)$ for use in $\mathcal{D}_T$

   Further categorization:

   (a) A lot of labeled data in $\mathcal{D}_S$ – similar to the multi-task learning

   (b) No labeled data in $\mathcal{D}_S$ – similar to the self-taught learning
2. **Transductive Transfer Learning**
   - $\mathcal{D}_S \neq \mathcal{D}_T$, $\mathcal{T}_S = \mathcal{T}_T$
   - A lot of labeled data in $\mathcal{D}_S$, no labeled data in $\mathcal{D}_T$ (thus some unlabeled data in $\mathcal{D}_T$ must be available)

   Further categorization:

   (a) $\mathcal{X}_S \neq \mathcal{X}_T$

   (b) $\mathcal{X}_S = \mathcal{X}_T$, but $P(X_S) \neq P(X_T)$, is related to:
   - domain adaptation for knowledge transfer in text classification
   - sample selection bias
   - co-variate shift
3. **Unsupervised Transfer Learning**
   - $\mathcal{T}_S \neq \mathcal{T}_T$, $\mathcal{Y}_S$ and $\mathcal{Y}_T$ are not observable
   - different focus – clustering, dimensionality reduction, density estimation
   - the predicted labels are latent variables, such as clusters or reduced dimensions

Approaches to the above three different settings can be summarized into four cases based on what to transfer:

1. **Instance-based transfer learning** – assumes that certain parts of the data in $\mathcal{D}_S$ can be reused for learning in $\mathcal{D}_T$ by re-weighting or importance sampling

2. **Feature-representation-transfer** − learning a "good" feature representation for $\mathcal{D}_T$, thus (hopefully) improving the performance significantly
3. **Parameter-transfer** − assumes that $\mathcal{T}_S$ and $\mathcal{T}_T$ share some parameters or prior distributions of the hyper-parameters of the models
4. **Relational-knowledge-transfer** − assumes that relationship among the data in $\mathcal{D}_S$ and $\mathcal{D}_T$ are similar

1 Inductive Transfer Learning

1.1 Transferring Knowledge of Instances

$\mathcal{D}_S$ data cannot be used directly, however certain parts possibly can − together with a few labeled data in $\mathcal{D}_T$.

TrAdaBoost boosting algorithm (extension of AdaBoost) addresses this problem, assumes $\mathcal{X}_S = \mathcal{X}_T$ and $\mathcal{Y}_S = \mathcal{Y}_T$, but $P(X_S) \neq P(X_T)$. It iteratively re-weights the $\mathcal{D}_S$ data to reduce the effect of the "bad" source data while encourage the "good" source data to contribute more for $\mathcal{D}_T$.

There are also other algorithms, e.g. a heuristic method to remove "misleading" training examples from $\mathcal{D}_S$ based on the difference between conditional probabilities $P(y_S|x_S)$ and $P(y_T|x_T)$.

1.2 Transferring Knowledge of Feature Representations

There are different strategies for different types of $\mathcal{D}_S$ data for finding "good" feature representations to minimize domain divergence and classification or regression model error.

If a lot of labeled data in $\mathcal{D}_S$ are available, supervised learning can be used (similar to *common feature learning* in the field of multi-task learning). Otherwise, unsupervised learning must be used.

1.2.1 Supervised Feature Construction

The basic idea is to learn a low-dimensional representation that is shared across related tasks. The learned representation can reduce the classification or regression model error of each task as well. It is learned by solving an optimization problem, which can be transformed into an equivalent convex optimization formulation and thus solved efficiently.

1.2.2 Unsupervised Feature Construction

The proposed solution is to apply sparse coding − a unsupervised feature construction method, for learning *higher level* features for transfer learning. It consists of two (three) steps

1. learning higher-level basis vectors $b = \{b_1, b_2, \ldots, b_s\}$
2. learning higher level features based on the basis vectors b
3. (optional) applying discriminative algorithms to train classification or regression models for use in $\mathcal{D}_T$

1.3 Transferring Knowledge of Parameters

Most approaches in this section, like a regularization framework and a hierarchical Bayesian framework are designed to work under multi-task learning, however they can be easily modified for transfer learning by adjusting the weights in the loss functions − for example by assigning a larger weight to the loss function of $\mathcal{D}_T$ (thus making it more important to minimize than the loss function of $\mathcal{D}_S$).

An example for this task is a MT-IVM algorithm, which is based on Gaussian Processes (GP) – it tries to learn parameters of a GP over multiple tasks by sharing the same GP prior. Similar approach uses free-form covariance matrix over tasks to model inter-task dependencies.

Another approach tries to transfer parameters of SVMs under a regularization framework − separating a parameter w into two terms:

1. a common term over tasks
2. a task-specific term

1.4 Transferring Relational Knowledge

Differently from other three contexts, this approach deals with data that are not independent and identically distributed (i.i.d.) (or at least it is not assumed that they are) and can be represented by multiple relations, such as networked data or social network data. Statistical relational learning techniques are proposed to solve these problems.

A *TAMAR* algorithm transfers relational knowledge with Markov Logic Networks (MLNs). MLN uses predicates to represent entities in a relational domain, while their relationships are represented in first-order logic. The idea is that if two domains are related to each other, there may exist mappings to connect entities and their relationships from $\mathcal{D}_S$ to $\mathcal{D}_T$ (e.g. academic domain−professor and industrial management domain−manager). TAMAR tries to use an MLN learned for $\mathcal{D}_S$ to aid the learning of an MLN for $\mathcal{D}_T$:

1. a mapping from a source MLN to $\mathcal{D}_T$ based on weighted pseudo-loglikelihood measure (WPLL) is created
2. a revision of the mapped structure is done using the *FORTE* algorithm − revising the first order theories

The revised MLN can be used as a relational model for inference or reasoning in $\mathcal{D}_T$.

Another approach on this task is based on a form of second-order Markov logic − it tries to discover structural regularities in $\mathcal{D}_S$ in the form of Markov logic formulas with predicate variables, by instantiating these formulas with predicates from $\mathcal{D}_T$.

2 Transductive Transfer Learning

2.1 Transferring Knowledge of Instances

Most approaches are motivated by importance sampling. We can learn the optimal parameters θ by minimizing the empirical risk (ERM):

$$\theta = \arg\min_{\theta \in \Theta} \sum_{(x,y) \in D_T} P(D_T) l(x, y, \theta) \quad (1)$$

However, since no labeled data in D_T are observed and $P(D_S) = P(D_T)$, we may simply learn the model:

$$\theta = \arg\min_{\theta \in \Theta} \sum_{(x,y) \in D_S} P(D_S) l(x, y, \theta) \quad (2)$$

If $P(D_S) \neq P(D_T)$:

$$\theta = \arg\min_{\theta \in \Theta} \sum_{(x,y) \in D_S} \frac{P(D_T)}{P(D_S)} P(D_S) l(x, y, \theta) \quad (3)$$

$$\approx \arg\min_{\theta \in \Theta} \sum_{i=1}^{n_S} \frac{P_T(x_{T_i}, y_{T_i})}{P_S(x_{S_i}, y_{S_i})} l(x, y, \theta) \quad (4)$$

$$= \arg\min_{\theta \in \Theta} \sum_{i=1}^{n_S} \frac{P(x_{S_i})}{P(x_{T_i})} l(x, y, \theta) \quad (5)$$

Last simplification comes from $P(Y_S|X_S) = P(Y_T|X_T)$, thus the difference between $P(D_S)$ and $P(D_T)$ is caused by $P(X_S)$ and $P(X_T)$, hence the simplification. The remaining problem is the estimation of $\frac{P(x_{S_i})}{P(x_{T_i})}$.

Various ways to estimate the fraction exist. One way is to estimate the terms independently by constructing simple classification problems. The other way is to estimate the probability ratio directly:

- by using various classifiers
- a kernel-mean matching (KMM) algorithm – matches the means between D_S and D_T in a reproducing-kernel Hilbert space (RKHS), can be rewritten as a quadratic programming (QP) optimization problem, avoids performing density estimation of either $P(x_{S_i})$ or $P(x_{T_I})$, which is difficult when the size of the dataset is small
- a Kullback-Leibler divergence (KLIEP) – can estimate the weights of the D_S and thus train models on the re-weighted data to perform model selection automatically using cross-validation

2.2 Transferring Knowledge of Feature Representations

Most approaches are under unsupervised learning frameworks. A structural correspondence learning (SCL) algorithm is proposed. It makes use of the unlabeled data from $\mathcal{D}_T$ to extract relevant features that may reduce the difference between the domains:

1. define a set of m *pivot* features (which are domain specific and depend on prior knowledge) on the unlabeled data from both domains
2. remove the pivot features from the data and that each as a new label vector
3. construct m classification problems
4. by assuming each can be solved by linear classifier, learn a matrix $W = [w_1 w_2 \dots w_m]$ of parameters
5. SVD is applied $W = UDV^T$, then $\theta = U^T_{[1:h,:]}$ (h being the number of shared features) is the linear mapping whose rows are the top left singular vectors of W.
6. apply standard discriminative algorithms to the augmented feature vector to build models

The drawback is that the pivot features must be well designed, which is difficult and domain-dependent.

A heuristic for selecting pivot features for natural language processing (NLP) problems exists (e.g. tagging of sentences), the follow-up work proposes using Mutual Information (MI) to choose the pivot features. MI-SCL tries to find some pivot features that have high dependence on the labels in $\mathcal{D}_S$.

In domain adaptation (transfer learning in NLP domain) a kernel-mapping function was proposed – it maps data from both $\mathcal{D}_S$ and $\mathcal{D}_T$ to a high-dimensional feature space, where standard methods are used to train classifiers. However, the constructed mapping function is domain knowledge driven, thus hard to generalize to other domains.

Other approaches include:

- a co-clustering based algorithm to propagate the label information across domains
- a bridged refinement algorithm – it corrects the labels predicted by a shift-unaware classifier towards a target distribution and takes the mixture distribution of the training and test data as a bridge to better transfer from the training data to the test data
- a spectral classification framework, where the objective function is introduced to seek consistency between the in-domain supervision and the out-of-domain intrinsic structure
- dimensionality reduction – Maximum Mean Discrepancy Embedding (MMDE) algorithm and more efficient feature extraction algorithm known as Transfer Component Analysis (TCA)

3 Unsupervised Transfer Learning

A little research work on this setting has been done.

3.1 Transferring Knowledge of Feature Representations

Self-taught clustering (STC) algorithm is an instance of unsupervised transfer learning. It aims at clustering a small amount of unlabeled data in $\mathcal{D}_T$ with help of a large amount in the $\mathcal{D}_S$. It tries to learn common feature space

across domains. An iterative algorithm is used to solve the optimization function given by STC.

Transferred discriminative analysis (TDA) tries to solve the transfer dimensionality reduction problem. It runs iteratively:

1. apply clustering to generate pseudo-class labels for the D_T
2. apply dimensionality reduction methods

to find the best subspace for D_T.

Transfer bounds

It would be very useful to know the "right" amount of information to transfer. It has been done using conditional Kolmogorov complexity to measure relatedness between tasks.

Another approach is a graph-based method. It embeds a set of learned source models in a graph using transferability as a metric. Transferring to a new task proceeds by using this graph by mapping the problem into it and learning a function to automatically determine the parameters to transfer.