2

Instance-Based Transfer Learning

2.1 Introduction

Intuitively, instance-based transfer learning approaches aim to reuse labeled data from the source domain help to train a more precise model for a target learning task. If the source domain and the target domain are quite similar, we can directly merge the source domain data into the target domain. Then it becomes a standard machine learning problem in a single domain. However, in many cases, this "direct adoption" strategy of source domain instances cannot help to solve the target task.

A common motivation behind instance-based transfer learning approaches is that some source domain labeled data are still useful for learning a precise model for the target domain while some are useless or even may hurt the performance of the target model if used. We can use the bias-variance analysis to understand this motivation. When the target domain data set is small, the model may have a high variance level and thus the model's generalization error is large. By adding a part of the source domain data as an auxiliary data set, the model's variance can potentially be reduced. However, if the data distributions of the two domains are very different, the new learning model may have a high bias. Therefore, if we can single out those source domain instances that follow a similar distribution as those in the target domain, we can reuse them and have both the variance and bias of the target learning model reduced.

Briefly, there are two key issues to resolve in using instance-based transfer learning. The first issue is how to single out the source domain-labeled instances that are similar to the target domain ones, because these instances are useful to train the target domain model. The second issue is how to utilize the identified "similar" source domain-labeled instances in an algorithm to learn a more accurate target domain learning model.

Recall that a domain $\mathbb{D} = \{\mathscr{X}, \mathbb{P}^X\}$ has two components: a feature space $\mathscr{X}$ and a marginal probability distribution $\mathbb{P}^X$. Given $\mathbb{D}$, a task $\mathbb{T} = \{\mathscr{Y}, \mathbb{P}^{Y|X}\}$ has two components: the label space $\mathscr{Y}$ and the conditional probability distribution $\mathbb{P}^{Y|X}$. A common assumption behind most instance-based transfer learning approaches

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

is that the input instances of the source domain and the target domain have the same or very similar support, which means that the features for most instances have a similar range of values. Furthermore, the output labels of the source and target tasks are the same. This assumption ensures that knowledge can be transferred across domains via instances. According the definitions of a domain and a task, this assumption implies that, in instance-based transfer learning, the difference between domains/tasks is only caused by the differences of the marginal distribution of the features (i.e., $\mathbb{P}_s^X \neq \mathbb{P}_t^X$) or conditional probabilities (i.e., $\mathbb{P}_s^{Y|X} \neq \mathbb{P}_t^{Y|X}$).

When $\mathbb{P}_s^X \neq \mathbb{P}_t^X$ but $\mathbb{P}_s^{Y|X} = \mathbb{P}_t^{Y|X}$, we refer to the problem setting as noninductive transfer learning.[1] For example, suppose a hospital, either private or public, aims to learn a prediction model for a specific disease from its own patients' electronic medical records. Here we consider each hospital as a different domain. As the populations of patients of different hospitals are different, the marginal probabilities $\mathbb{P}^X$s are different across different domains. However, as the reasons that cause the specific disease are the same, the conditional probabilities $\mathbb{P}^{Y|X}$ across different domains remain the same. When $\mathbb{P}_s^{Y|X} \neq \mathbb{P}_t^{Y|X}$, we refer to the problem setting as inductive transfer learning. For instance, consider avian influenza virus as the specific disease in the previous example. As avian influenza virus has been evolving, the reasons causing avian influenza virus may change across different subtypes of avian influenza virus, for example, H1N1 versus H5N8. Here we consider learning a prediction model for each subtype of avian influenza virus for a specific hospital as a different task. As the reasons that cause different subtypes of avian influenza virus are different, the conditional probabilities $\mathbb{P}^{Y|X}$ are different across different tasks. In noninductive transfer learning, as the conditional probabilities across domains are the same, that is, $\mathbb{P}_s^{Y|X} = \mathbb{P}_t^{Y|X}$, it can be theoretically proven that, even without any labeled data in the target domain, an optimal predictive model can be learned from the source domain-labeled data and the target domain-unlabeled data. While in the inductive transfer learning case, as the conditional probabilities are different across tasks, a few labeled data in the target domain would then be required to exist to help transfer the conditional probability or the discriminative function from the source task to the target task. Since the assumptions of noninductive transfer learning and inductive transfer learning are different, the designs of instance-based transfer learning approaches for these two settings are different. In the following, we will review the motivations, basic ideas and representative methods for noninductive and inductive transfer learning in detail.

[1] Note that here we do not adopt the term "transductive transfer learning" used by Pan and Yang (2010) because the term "transductive" has been widely used to distinguish whether a model has an out-of-sample generalization ability, which may cause some confusion if used to define transfer learning problem settings.

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

2.2 Instance-Based Noninductive Transfer Learning

As mentioned earlier, in noninductive transfer learning, the source task and the target task are assumed to be the same, and the supports of the input instances across domains are assumed to be the same or very similar, that is, $\mathscr{X}_s = \mathscr{X}_t$. The only difference between domains is caused by the marginal distribution of input instances, that is, $\mathbb{P}_s^X \neq \mathbb{P}_t^X$. Under this setting, we are given a set of source domain-labeled data $\mathscr{D}_s = \{(\mathbf{x}_{s_i}, y_{s_i})\}_{i=1}^{n_s}$, and a set of target domain-unlabeled data $\mathscr{D}_t = \{(\mathbf{x}_{t_i})\}_{i=1}^{n_t}$. The goal is to learn a precise predictive model for the target domain unseen data.

In the following, we show that, under the assumptions in noninductive transfer learning, one is still able to learn an optimal predictive model for the target domain even without any target domain-labeled data. Suppose our goal is to learn a predictive model in terms of parameters θ_t for the target domain, based on the learning framework of empirical risk minimization (Vapnik, 1998), the optimal solution of θ_t can be learned by solving the following optimization problem.

$$\theta_t^* = \arg\min_{\theta_t \in \Theta} \mathbb{E}_{(\mathbf{x},y)\in\mathbb{P}_t^{X,Y}} [\ell(\mathbf{x}, y, \theta)], \tag{2.1}$$

where $\ell(\mathbf{x}, y, \theta)$ is a loss function in terms of the parameters θ_t. Since there are no target domain-labeled data, one cannot optimize (2.1) directly. It has been proven by Pan (2014) that, by using the Bayes' rule and the definition of expectation, the optimization (2.1) can be rewritten as follows,

$$\theta_t^* = \arg\min_{\theta_t \in \Theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}_s^{X,Y}} \left[\frac{P_t(\mathbf{x}, y)}{P_s(\mathbf{x}, y)} \ell(\mathbf{x}, y, \theta_t) \right], \tag{2.2}$$

which aims to learn the optimal parameter θ_t^* by minimizing the weighted expected risk over source domain-labeled data. In noninductive transfer learning, as $\mathbb{P}_s^{Y|X} = \mathbb{P}_t^{Y|X}$, by decomposing the joint distribution $\mathbb{P}^{X,Y} = \mathbb{P}^{Y|X}\mathbb{P}^X$, we obtain $\frac{P_t(\mathbf{x},y)}{P_s(\mathbf{x},y)} = \frac{P_t(\mathbf{x})}{P_s(\mathbf{x})}$. Hence, (2.2) can be further rewritten as

$$\theta_t^* = \arg\min_{\theta_t \in \Theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}_s^{X,Y}} \left[\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})} \ell(\mathbf{x}, y, \theta_t) \right], \tag{2.3}$$

where a weight of a source domain instance $\mathbf{x}$ is defined as the ratio of marginal distributions of input instances between the target domain and the source domain at the data point $\mathbf{x}$. Given a set of source domain-labeled data $\{(\mathbf{x}_{s_i}, y_{s_i})\}_{i=1}^{n_s}$, by defining $\beta(\mathbf{x}) = \frac{P_t(\mathbf{x})}{P_s(\mathbf{x})}$, an empirical approximation of (2.3) can be written as[2]

$$\theta_t^* = \arg\min_{\theta_t \in \Theta} \sum_{i=1}^{n_s} \beta(\mathbf{x}_{s_i}) \ell(\mathbf{x}_{s_i}, y_{s_i}, \theta_t), \tag{2.4}$$

Therefore, to properly reuse the source domain-labeled data to learn a target model, one needs to estimate the weight's $\{\beta(\mathbf{x}_{s_i})\}$. As shown in (2.4), to estimate $\{\beta(\mathbf{x}_{s_i})\}$,

[2] In practice, a regularization term is added to avoid model overfitting.

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

that is, density ratios, only input instances without labels from the source domain and the target domain are required. A simple solution to estimate $\{\beta(\mathbf{x}_{s_i})\}$ for each source domain instance is to first estimate $\mathbb{P}_t^X$ and $\mathbb{P}_s^X$, respectively, and then compute the ratio $\frac{P_t(\mathbf{x}_{s_i})}{P_s(\mathbf{x}_{s_i})}$ for each specific source domain instance $\mathbf{x}_{s_i}$. However, it is well known that density estimation itself is a difficult task (Tsuboi et al., 2009), especially when data are of high dimensions. In this way, the error caused by density estimation will be propagated to the density ratio estimation .

In the literature (Quionero-Candela et al., 2009), more promising solutions have been proposed to estimate $\frac{\mathbb{P}_t^X}{\mathbb{P}_s^X}$, directly bypassing the density estimation step. In the following sections, we introduce how to directly estimate the density ratio by reviewing several representative methods.

2.2.1 Discriminatively Distinguish Source and Target Data

One simple and effective approach to learn the weights is to transform the problem of estimating the marginal probability density ratio to the problem of distinguishing whether an instance is from the source domain or the target domain. This can be formulated as a binary classification problem with data instances from the source domain being labeled as 1 and those from the target domain being labeled as 0.

For example, Zadrozny (2004) proposes a rejection sampling-based method for correcting sample selection bias. The rejection sampling process is defined as follows. A binary random variable $\delta \in \{1,0\}$, which is called selection variable, is introduced. An instance $\mathbf{x}$ is sampled from the target marginal distribution $\mathbb{P}_t^X$ with probability $P_t(\mathbf{x})$, that is, $P_t(\mathbf{x}) = P(\mathbf{x}|\delta = 0)$. Similarly, $P_s(\mathbf{x})$ can be rewritten as $P_s(\mathbf{x}) = P(\mathbf{x}|\delta = 1)$. $\mathbf{x}$ is accepted by the source domain with probability $P(\delta = 1|\mathbf{x})$ or rejected with probability $P(\delta = 0|\mathbf{x})$. In mathematics, with the new variable δ, the density ratio for each data instance $\mathbf{x}$ can be formulated as

$$\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})} = \frac{P(\delta = 1)}{P(\delta = 0)} \frac{P(\delta = 0)}{P(\delta = 1)} \frac{P_t(\mathbf{x})}{P_s(\mathbf{x})}, \tag{2.5}$$

where $P(\delta)$ is the prior probability of δ in the union data set of the source domain and the target domain. By using the Bayes, rule and the equivalent forms of $P_s(\mathbf{x})$ and $P_t(\mathbf{x})$ in terms of δ, (2.5) can be further reformulated as

$$\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})} = \frac{P(\delta = 1)}{P(\delta = 0)} \left(\frac{1}{P(\delta = 1|\mathbf{x})} - 1 \right).$$

Therefore, the density ratio for each source domain data instance can be estimated as $\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})} \propto \frac{1}{P_{s,t}(\delta=1|\mathbf{x})}$. To compute the probability $P(\delta = 1|\mathbf{x})$, we regard it as a binary classification problem and train a classifier to solve it. After calculating the ratio for each source data instance, a model can be trained by either reweighting each source data instance or performing importance sampling on the source data set.

Following the idea of Zadrozny (2004), Bickel et al. (2007) propose a framework

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

to integrate the density ratio estimation step and the model training step with reweighted source data instances. Let $\mathbb{P}^X$ denote the probability density of $\mathbf{x}$ in the union data set of the source domain and the target domain. We can use any classifier to estimate the probability $P(\delta = 1|\mathbf{x})$. Suppose the classifier is parameterized by $\mathbf{v}$ and the parameters for the final learning model that is trained on the reweighted source domain data are denoted by $\mathbf{w}$. All the parameters can be optimized using the maximum a posterior (MAP) approach:

$$[\mathbf{w}, \mathbf{v}]_{MAP} = \underset{\mathbf{w},\mathbf{v}}{\operatorname{argmax}} P(\mathbf{w}, \mathbf{v}|\mathscr{D}_s, \mathscr{D}_t),$$

where $\mathscr{D}_s$ and $\mathscr{D}_t$ denote the source data set and the target data set, respectively. Note that $P(\mathbf{w}, \mathbf{v}|\mathscr{D}_s, \mathscr{D}_t)$ is proportional to $P(\mathscr{D}_s|\mathbf{w}, \mathbf{v})P(\mathscr{D}_s, \mathscr{D}_t|\mathbf{v})P(\mathbf{w})P(\mathbf{v})$. Therefore, the MAP solution can be found by maximizing $P(\mathscr{D}_s|\mathbf{w}, \mathbf{v})P(\mathscr{D}_s, \mathscr{D}_t|\mathbf{v})P(\mathbf{w})P(\mathbf{v})$.

2.2.2 Kernel Mean Matching

Another effective approach to estimate the density ratio is using the techniques of kernel embedding of distributions (Smola et al., 2007a). For instance, Huang et al. (2006) propose the Kernel Mean Matching (KMM) method to directly learn the density ratio by aligning the mean of source domain data instances to that of target domain data instances in a reproducing kernel Hilbert space (RKHS).

Specifically, we use β_i to denote $\frac{P_t(\mathbf{x}_i^s)}{P_s(\mathbf{x}_i^s)}$ for each source domain data instance $\mathbf{x}_i^s$ and define $\boldsymbol{\beta}$ as $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{n_s})$, where n_s is the size of the source domain data set. KMM makes use of the theory of Maximum Mean Discrepancy (MMD) (Gretton et al., 2007) between distributions. Given two samples, based on MMD, the distance between two sample distributions is simply the distance between the two mean elements in an RKHS. Therefore, KMM aims to learn the weights of source domain instances by matching the mean of the reweighted source domain instances to that of the target domain instances in an RKHS:

$$\min_{\boldsymbol{\beta}} \left\| \mu(\mathbb{P}_t^X) - \mathbb{E}_{\mathbb{P}_s^X}[\beta(\mathbf{x})\Phi(\mathbf{x})] \right\| \quad \text{s.t. } \beta(\mathbf{x}) \geq 0,\ \mathbb{E}_{\mathbb{P}_s^X}[\beta(\mathbf{x})\Phi(\mathbf{x})] = 1, \tag{2.6}$$

where Φ transforms each source domain data instance into the RKHS $\mathscr{F}$, and $\mu(\mathbb{P}_t^X)$ is the expectation of the target domain instances in the RKHS, that is, $\mu(\mathbb{P}_t^X) = \mathbb{E}_{\mathbb{P}_t^X}[\Phi(\mathbf{x})]$.

In practice, one can optimize the following empirical objective:

$$\min_{\boldsymbol{\beta}} \left\| \frac{1}{n_s}\sum_{i=1}^{n_s} \beta_i \Phi(\mathbf{x}_i^s) - \frac{1}{n_t}\sum_{i=1}^{n_t} \Phi(\mathbf{x}_i^t) \right\|^2 \quad \text{s.t. } \beta_i \geq 0,\ \left| \frac{1}{n_s}\sum_{i=1}^{n_s} \beta_i - 1 \right| \leq \epsilon, \tag{2.7}$$

where ϵ is a positive real number. After solving the optimal $\boldsymbol{\beta}$, that is, the weights, $\boldsymbol{\beta}$ can be incorporated into (2.4) with any specified loss function to learn a predictive model θ_t^* for the target domain.

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

2.2.3 Function Approximation

A third representative approach to estimate density ratio is to consider the density ratio as an unknown function, and learn a combination of a series of base functions to approximate it. This is also known as covariate shift methods (Sugiyama et al., 2008). To be specific, by defining $\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})}$ as a function $\omega(\mathbf{x})$, one could assume $\omega(\mathbf{x})$ is a linear combination of several base functions as

$$\widetilde{\omega}(\mathbf{x}) = \sum_{l=1}^{b} \alpha_l \phi_l(\mathbf{x}),$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_b)^T$ are coefficients to be learned and $\phi_l(\cdot)$ is the lth base function that can be linear or nonlinear. In this way, $P_t(\mathbf{x})$ can be approximated by $\widetilde{P}_t(\mathbf{x}) = \widetilde{\omega}(\mathbf{x}) P_s(\mathbf{x})$. The coefficients $\boldsymbol{\alpha}$ can be learned by minimizing a loss function between $P_t(\mathbf{x})$ and $\widetilde{P}_t(\mathbf{x})$. Different loss functions lead to different specific methods.

For instance, Sugiyama et al. (2008) propose to use Kullback–Leibler (KL) divergence as the loss function. The resultant method is known as KL Importance Estimation Procedure (KLIEP), whose objective is written as follows,

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}(\mathbb{P}_t^X, \widetilde{\mathbb{P}}_t^X) &= \int_{\mathscr{X}_t} P_t(\mathbf{x}) \log \frac{P_t(\mathbf{x})}{\widetilde{\omega}(\mathbf{x}) P_s(\mathbf{x})} \mathrm{d}\mathbf{x} && (2.8) \\ &= \int_{\mathscr{X}_t} P_t(\mathbf{x}) \log \frac{P_t(\mathbf{x})}{P_s(\mathbf{x})} \mathrm{d}\mathbf{x} - \int_{\mathscr{X}_t} P_t(\mathbf{x}) \log \widetilde{\omega}(\mathbf{x}) \mathrm{d}\mathbf{x}. && (2.9) \end{aligned}$$

Note that, in (2.9), the ground-truth marginal probability of the target domain data, $\mathbb{P}_t^X$, is used. However, it can shown that, empirically, minimizing the aforementioned KL divergence can be approximated by solving the following optimization problem, where the ground truth marginal probability of the target domain data is canceled out:

$$\max_{\boldsymbol{\alpha}} \frac{1}{n_t} \sum_{j=1}^{n_t} \log \left(\sum_{l=1}^{b} \alpha_l \phi_l(\mathbf{x}_j^t) \right) \quad \text{s.t.} \ \frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{l=1}^{b} \alpha_l \phi_l(\mathbf{x}_i^s) = 1, \ \ \alpha_l \geq 0 \ \ \forall l \in \{1, \ldots, b\}.$$

Another example of the loss function in discrepancy between $\omega(\mathbf{x})$ and $\widetilde{\omega}(\mathbf{x})$ is the squared loss (Kanamori et al., 2009). The resultant optimization problem can be written as follows,

$$\min_{\alpha} \int_{\mathscr{X}_s \cup \mathscr{X}_t} (\widetilde{\omega}(\mathbf{x}) - \omega(\mathbf{x}))^2 P_s(\mathbf{x}) \mathrm{d}\mathbf{x}.$$

Besides, using KL divergence and squared loss as the loss function, many other forms of loss functions can be used.

2.3 Instance-Based Inductive Transfer Learning

Different from noninductive transfer learning, in inductive transfer learning, the source task and the target task can be different in terms of conditional probabil-

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

ities, that is, $\mathbb{P}_s^{Y|X} \neq \mathbb{P}_t^{Y|X}$. As the conditional probability is changed across different tasks, if there are no labeled data in the target domain, then it is very difficult if not impossible to adapt $\mathbb{P}_s^{Y|X}$ to construct a precise $\mathbb{P}_t^{Y|X}$. Therefore, in most instance-based inductive transfer learning approaches, besides a set of source domain-labeled data $\mathscr{D}_s = \{(\mathbf{x}_{s_i}, y_{s_i})\}_{i=1}^{n_s}$, a small set of target domain-labeled data $\mathscr{D}_t = \{(\mathbf{x}_{t_i}, y_{t_i})\}_{i=1}^{n_t}$ is also required as inputs.[3] The goal is still to learn a precise predictive model for the target domain unseen data.

2.3.1 Integration of Source and Target Loss

An intuitive solution to make use of both source domain-labeled data and target domain-labeled data to train a model for the target domain is to decompose the loss function into two parts: one is for the source domain-labeled data, and the other is for the target domain-labeled data. A trade-off parameter is usually introduced to balance the impact of the two losses.

As an early representative work, Wu and Dietterich (2004) propose an instance-based K-nearest-neighbor (KNN) classifier to optimize the classification accuracy on both the source domain and the target domain. Specifically, in a traditional KNN classifier, the hypothesis $h(\mathbf{x})$ is defined by k training data instances that are closest to each test instance $\mathbf{x}$. In the proposed KNN-based inductive transfer learning method, K_s nearest source domain instances and K_t nearest target domain instances are first identified for a target domain test data instance $\mathbf{x}_i^t$. Then, for each class label y, the overall vote on the instance $\mathbf{x}_i^t$, denoted by $V(y)$, is computed as $V(y) = \theta(\frac{V_t(y)}{K_t}) + (1-\theta)(\frac{V_s(y)}{K_s})$, where $V_t(y)$ and $V_s(y)$ are the numbers of votes on class y from the K_t and the K_s nearest instances from the target domain and the source domain, respectively, and θ is a trade-off parameter to control the relative importance of the source domain nearest neighbors and the target domain nearest neighbors.

Such an idea can be applied to other base classifier beyond KNN. Wu and Dietterich (2004) also propose a support vector machine (SVM) based approach (Smola and Schölkopf, 2004) for instance-based inductive transfer learning methods. Recall that the objective function of SVMs is

$$\min \sum_j \alpha_j + C \sum_j \epsilon_j \text{ s.t. } y_i \left(\sum_j y_j \alpha_j K(\mathbf{x}_j, \mathbf{x}_i) + b \right) \geq 1 - \epsilon_i \ \ \forall i,\ \alpha_j \geq 0 \ \ \forall j,$$

where α_js are the model parameters of a SVM, ϵ_js are slack variables to absorb errors and C is a parameter to control how much penalty is conducted the misclassified examples. In the inductive transfer learning setting, Wu and Dietterich (2004) proposed to modify the objective function and constraints by considering the source domain-labeled data and the target domain-labeled data differently. Suppose α_j^s and ϵ_j^s denote the model parameters and slack variables for the source

[3] In some approaches, a set of target-unlabeled data is assumed to be given as well.

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

domain instance $\mathbf{x}_j^s$ for $j \in \{1,\ldots,n_s\}$, respectively. Similarly, α_j^t and ϵ_j^t denote the parameter and slack variable for the target domain instance $\mathbf{x}_j^t$, respectively, for $j \in \{1,\ldots,n_t\}$. The parameters C_s and C_t are trade-off parameters. Then the revised objective function of SVMs is formulated as

$$\begin{aligned}
\min\ & \sum_{j=1}^{n_s} \alpha_j^s + \sum_{j=1}^{n_t} \alpha_j^t + C_s \sum_{j=1}^{n_s} \epsilon_j^s + C_t \sum_{j=1}^{n_t} \epsilon_j^t, \\
\text{s.t.}\ & y_i^t \left(\sum_{j=1}^{n_s} y_j^t \alpha_j^t K\left(\mathbf{x}_j^t, \mathbf{x}_i^t\right) + \sum_{j=1}^{n_s} y_j^s \alpha_j^s K\left(\mathbf{x}_j^s, \mathbf{x}_i^t\right) + b \right) \geq 1 - \epsilon_i^t \quad i \in \{1,\ldots,n_t\}, \\
& y_i^s \left(\sum_{j=1}^{n_t} y_j^t \alpha_j^t K\left(\mathbf{x}_j^t, \mathbf{x}_i^s\right) + \sum_{j=1}^{n_s} y_j^s \alpha_j^s K\left(\mathbf{x}_j^s, \mathbf{x}_i^s\right) + b \right) \geq 1 - \epsilon_i^s \quad i \in \{1,\ldots,n_s\}, \\
& \alpha_j^t \geq 0 \quad j \in \{1,\ldots,n_t\},\ \alpha_j^s \geq 0 \quad j \in \{1,\ldots,n_s\}.
\end{aligned}$$

Generally speaking, the revised SVM jointly optimizes the losses on the labeled data of both the source domain and the target domain.

Liao et al. (2005) further extend this idea to logistic regression, and propose the "Migratory-Logit" algorithm. Migratory-Logit models the difference between two domains by introducing a new "auxiliary variable" μ_i for each source data instance $(\mathbf{x}_i^s, y_i^s)$. The parameter μ_i could be geometrically understood as a "intercept term" that makes $\mathbf{x}_i^s$ migrate toward class y_i^s in the target domain. It measures how mismatch the source data instance $\mathbf{x}_i^s$ is with respect to the target domain distribution $\mathbb{P}_t^X$ and thus controls the importance of source data instances. For a target domain data instance $(\mathbf{x}_i^t, y_i^t)$, the posterior probability of its label y_i^t is the same as the traditional logistic regression, that is, $P(y_i^t|\mathbf{x}_i^t;\mathbf{w}) = \delta(y_i^t \mathbf{w}^T \mathbf{x}_i^t)$, where $\mathbf{w}$ is the parameter vector and $\delta(a) = \frac{1}{1+\exp(-a)}$ is the Sigmoid function. For a source domain instance $(\mathbf{x}_i^s, y_i^s)$, the posterior probability of y_i^s is defined as:

$$P(y_i^s|\mathbf{x}_i^s;\mathbf{w},\mu_i) = \delta(y_i^s \mathbf{w}^T \mathbf{x}_i^s + y_i^s \mu_i).$$

By defining $\boldsymbol{\mu} = (\mu_1,\ldots,\mu_m)^T$, the log-likelihood is computed as

$$\mathscr{L}(\mathbf{w},\boldsymbol{\mu};\mathscr{D}_s \cup \mathscr{D}_t) = \sum_{i=1}^{n_t} \ln \delta(y_i^t \mathbf{w}^T \mathbf{x}_i^t) + \sum_{i=1}^{n_s} \ln \delta(y_i^s \mathbf{w}^T \mathbf{x}_i^s + y_i^s \mu_i).$$

Then, all the parameters can be learned by maximizing the log-likelihood with the optimization problem formulated as

$$\max_{\mathbf{w},\boldsymbol{\mu}}\ L(\mathbf{w},\boldsymbol{\mu};\mathscr{D}_s \cup \mathscr{D}_t) \quad \text{s.t.}\ \frac{1}{n_s} \sum_{i=1}^{n_s} y_i^s \mu_i \leq C,\ y_i^s \mu_i \geq 0,\ \forall i \in \{1,2,\ldots,n_s\},$$

where C is a hyper parameter to control the overall importance of the source domain data set.

The aforementioned approaches assume that, in the target domain, only labeled data are available as inputs for transfer learning algorithms. In many scenarios, plenty of unlabeled data may be available in the target domain as well.

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

Jiang and Zhai (2007) propose a general semi-supervised framework for instance-based inductive transfer learning, where both labeled and unlabeled data in the target domain are utilized with the source domain labeled data to train a target predictive model.

In the work by Jiang and Zhai (2007), a parameter α_i is introduced for each source domain instance $(\mathbf{x}_i^s, y_i^s) \in \mathscr{D}_s$ to measure how $P_s(y_i^s|\mathbf{x}_i^s)$ is different from $P_t(y_i^s|\mathbf{x}_i^s)$. Another parameter β_i is introduced for each source domain instance $(\mathbf{x}_i^s, y_i^s) \in \mathscr{D}_s$ to approximate the density ratio $\frac{P_t(\mathbf{x}_i^s)}{P_s(\mathbf{x}_i^s)}$. Then, for each target domain unlabeled instance $\mathbf{x}_i^{t,u} \in \mathscr{D}_t$ and each possible label y, a parameter $\gamma_i(y)$ is used to measure how likely the true label of $\mathbf{x}_i^{t,u}$ is y. Let $\mathscr{D}_t = \mathscr{D}_l \cup \mathscr{D}_u$ where $\mathscr{D}_l = \{(\mathbf{x}_j^{t,l}, y_j^{t,l})\}_{j=1}^{n_{t,l}}$ represents the subset of target domain-labeled instances and $\mathscr{D}_u = \{(\mathbf{x}_k^{t,u})\}_{k=1}^{n_{t,u}}$ represents the subset of target domain-unlabeled instances. To find an optimal classifier in terms of parameters θ, Jiang and Zhai (2007) propose to solve the following optimization problem:

$$\begin{aligned}\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \; & \frac{\lambda_s}{C_s} \sum_{i=1}^{n_s} \alpha_i \beta_i \log P(y_i^s|\mathbf{x}_i^s;\boldsymbol{\theta}) + \frac{\lambda_{t,l}}{C_{t,l}} \sum_{j=1}^{n_{t,l}} \log P(y_j^{t,l}|\mathbf{x}_j^{t,l};\boldsymbol{\theta}) \\ & + \frac{\lambda_{t,u}}{C_{t,u}} \sum_{k=1}^{n_{t,u}} \sum_{y \in \mathscr{Y}} \gamma_k(y) \log(P(y|\mathbf{x}_k^{t,u};\boldsymbol{\theta})) + \log P(\boldsymbol{\theta}),\end{aligned}$$

where $C_s = \sum_{i=1}^{n_s} \alpha_i \beta_i$, $C_{t,l} = n_{t,l}$, and $C_{t,u} = \sum_{k=1}^{n_{t,u}} \sum_{y \in \mathscr{Y}} \gamma_k(y)$ are normalization factors, the regularization parameters λ_s, $\lambda_{t,l}$ and $\lambda_{t,u}$ control the relative importance of each part with the sum equal to 1, and the prior $P(\boldsymbol{\theta})$ encodes the normal prior for $\boldsymbol{\theta}$. In this way, the source domain-labeled data, the target domain-labeled data and the target domain-unlabeled data are fully utilized to learn the optimal solution of $\boldsymbol{\theta}$.

2.3.2 Boosting-Style Methods

Another group of methods of instance-based inductive transfer learning is based on the boosting algorithm, which aims to identify misleading source domain instances by iteratively updating their weights. For instance, the TrAdaBoost algorithm proposed by Dai et al. (2007b) is the first boosting-style algorithm for an instance-based inductive transfer learning setting.

TraAdaBoost adopts a similar instance reweighting strategy used in AdaBoost to find useful data instances from the source domain. Specifically, TrAdaBoost first trains a model h on the union of $\mathscr{D}_s$ and $\mathscr{D}_t$. Then it uses h to make predictions on the target domain data and calculates the mean loss on the target domain as $\epsilon = \frac{\sum_{i=1}^{n_t} w_i^t l(h(\mathbf{x}_i^t), y_i^t)}{\sum_{i=1}^{n_t} w_i^t}$, where w_i^t is the weight for $\mathbf{x}_i^t$ and $l(\cdot,\cdot)$ is the loss function. For each target domain instance, its weight is updated as $w_i^t = w_i^t \beta^{-l(h(\mathbf{x}_i^t), y_i^t)}$, where $\beta = \epsilon/(1-\epsilon)$. This reweighting strategy is similar to AdaBoost in that, if a target

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

domain data instance has a higher loss, its weight should be increased in the next iteration.

For each source domain instance, if it has a higher loss, it may not be helpful to the target task and so its loss will be decreased in the next iteration. The rule to update the weight for each source domain instance is $w_i^s = w_i^s \theta^{l(h(\mathbf{x}_i^s), y_i^s)}$, where $\theta = 1/(1+\sqrt{2\ln n_s/(n_s+n_t)})$.

With these update rules, TraAdaBoost iteratively reweights both the source domain-labeled data and the target domain-labeled data to reduce the impact of misleading data instances in the source domain, and learn a series of classifiers to construct an ensemble classifier for the target domain.

2.3.3 Instance Generation Methods

Instead of reusing source domain-labeled data for the target domain, an alternative approach is to develop generative models to generate new instances for the target domain to be used to learn a precise target domain predictive model. Such generative models usually require sufficient source domain data and a few target domain data as inputs.

Instance-based transfer learning can also be used to adapt the style of instances in the target domain based on the source domain instances. For instance, Gatys et al. (2016) transfer image styles with a deep generative model to create new target images by preserving the semantic content of target images while synthesizing its texture from a source image. Basically, the overall loss function to generate a new image consists of two losses: the content loss $\mathscr{L}_{content}$ and the style loss $\mathscr{L}_{style}$:

$$\mathscr{L} = \alpha \mathscr{L}_{content}(G,T) + \beta \mathscr{L}_{style}(G,S), \tag{2.10}$$

where G is the output image, S is the source image providing style and T is the target image offering content. Here $\mathscr{L}_{content}$ is defined as

$$\mathscr{L}_{content}(G,T,l) = \frac{1}{2}\sum_{i,j}(G_{i,j}^l - T_{i,j}^l)^2, \tag{2.11}$$

where l stands for the lth layer of deep learning model, i stands for the feature mapping of the ith filter in the layer and j stands for the jth element of the vectorized feature mapping. In addition, the style loss is formulated as

$$\mathscr{L}_{style}(G,S) = \sum_l^L w_l E_l = \sum_l^L w_l \sum_{i,j}\left(Gamm(G)_{i,j}^l - Gamm(S)_{i,j}^l\right)^2, \tag{2.12}$$

where $Gamm(\cdot)_{i,j}^l$, the style representation, is defined as the inner product between the vectorized feature maps i and j in layer l, that is, $Gamm(G)_{i,j}^l = \sum_k F_{ik}^l F_{jk}^l$. Specifically, in the work by Gatys et al. (2016), a nineteen-layer Visual Geometry Group Network network is used as the base model and all of its max-pooling

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press

layers are replaced by the mean pooling layers. First, the style and content features are extracted from source images and target images. Then, a random white noise image G_0 is passed through the network and its style features G^l and content features F^l are computed. Gradients with respect to the pixel values can be computed using error back-propagation and is used to iteratively update the generated image G.

Although the task studied in Gatys et al.'s (2016) work is about style transfer for images, the idea of generating new instances in the target domain by capturing some important properties of the source domain can be applied to many other transfer learning applications. We will review more generative models for transfer learning later in Chapter 7.

https://doi.org/10.1017/9781139061773.004 Published online by Cambridge University Press