Transfer learning / Multi-task learning ideas in a nutshell

Traditional setting (ML): given data set D, learn the underlying distribution, then test it on test data coming from the SAME distribution

Transfer learning (TL): I have sourse D_S and target D_T, where:
① some relateness between D_S and D_T exists
② lot of info in D_S, little info available in D_T

Multi-task learning (MTL): I have several sources $D_1 \cdots D_K$, and/or different tasks (regression, classification, clustering ...)

Note: TL/MTL do not ask for different dataset and different tasks at the same time.

TL/MTL ideas

① use D_S, D_T to train model together:

1) naive approach: reweight loss on D_S:

$$\theta^* = \arg\min \mathcal{L}(\theta; D_S) + \alpha \mathcal{L}(\theta; D_T) + R(\theta),$$

θ is model parameter, α controls the weighting

$R(\theta)$ regularize the model

problem: how to choose α?

2) covariate shift: remember often we define $\mathcal{L}(\theta; D_S) = \mathbb{E}_{x\sim D_S}[l(\theta;x)]$ (same for $\mathcal{L}(\theta; D_T)$), then we first "shift" the datapoints in D_S to be approx. distributed like D_T, then train the model (see Shimodaira 2000 paper):

$$\theta^* = \arg\min \mathcal{L}(\theta; D_T) + \mathbb{E}_{x\sim D_S}\left[\frac{P_T(x)}{P_S(x)} l(\theta;x)\right]$$

where P_T, P_S are (empirical) distributions of D_T, D_S

(now we solve the problem of choosing α in the naive way)

now new problem: how to estimate $\frac{P_T(x)}{P_S(x)}$?

A) estimate P_T, P_S separately using whatever methods you like.

problem: estimator of P_T is often not reliable.

B) directly obtain IS weights is not optimal (because of problems of IS), so maybe we can accept some biases?

some ideas:

A) (kernel) feature mean matching (see Gretton's book chapter):

define $\beta(x) = \frac{P_T(x)}{P_S(x)}$ and feature embedding $\Phi(x)$, then solve the following problem:

$$\underset{\beta}{\text{minimize}} \; \| E_{x\sim P_T}[\Phi(x)] - E_{x\sim P_S}[\beta(x)\Phi(x)] \|$$

$$\text{subject to } \beta(x) \geq 0 \text{ and } E_{x\sim P_S}[\beta(x)] = 1$$

now $\|\cdot\|$ is of choose, or we can do it implicitely using kernel tricks!

B) train a classifier $f(x)$ to distinguish data from D_S or D_T:

def $p(z=1|x) = \frac{P_T(x)}{P_T(x)+P_S(x)}$ (prob. of x from D_T)

③

then $\dfrac{p_T(x)}{p_S(x)} = \dfrac{p(z=1|x)}{p(z=-1|x)}$, and if we define

$$p(z=1|x) = \frac{1}{1+e^{-f(x)}} \Rightarrow \frac{p_T(x)}{p_S(x)} = e^{f(x)}$$

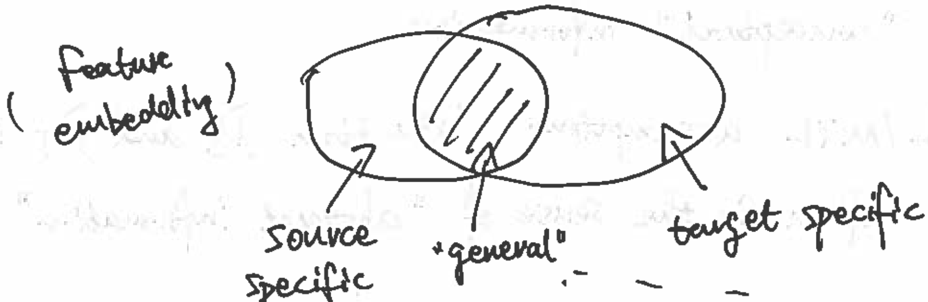
So the problem of covariate shift is solved in two steps:

① compute $f(x)$ using maximum likelihood / MAP ...

② use $e^{f(x)}$ as the weight to solve the original model training.

---

3) "bridging" via feature space (see Daumé III's EasyAdapt)



define two embeddings:

for $x \sim D_S$: $\Phi_S(x) = (\phi_S(x), \phi_g(x), 0)$

for $x \sim D_T$: $\Phi_T(x) = (0, \phi_g(x), \phi_T(x))$

then define model using input $\Phi_S(x)$ and $\Phi_T(x)$:

$$\theta^* = \arg\min \mathbb{E}_{x\sim D_S}[l(\theta; \Phi_S(x))] + \mathbb{E}_{x\sim D_T}[l(\theta; \Phi_T(x))] + R(\theta)$$

example: logistic regression: model ~~the~~

$$F(x;\theta) = \begin{cases} \text{sigmoid}(W_S^T \phi_S(x) + W_g^T \phi_g(x)), & x\sim D_S \\ \text{sigmoid}(W_g^T \phi_g(x) + W_T^T \phi_T(x)), & x \sim D_T \end{cases}$$

adding unlabelled data: we want the model to agree on unlabelled data no matter where it comes from:

(logistic regression) $\text{sigmoid}(W_S^T \phi_S(x) + W_g^T \phi_g(x)) = \text{sigmoid}(W_g^T \phi_g(x) + W_T^T \phi_T(x))$

$\Rightarrow$ constraint $W_S^T \phi_S(x) = W_T^T \phi_T(x)$ on unlabelled data

4) Joint representation (deep learning people like this!)

Learn a joint representation of data from D_S and D_T

$$x \sim D_S \longrightarrow h_S^1 \longrightarrow h_S^2 \cdots \longrightarrow h_S^{L_S}$$

low-level feature representation

$$h^1_{joint} \rightarrow \cdots$$

$$x \sim D_T \longrightarrow h_T^1 \longrightarrow h_T^2 \cdots \longrightarrow h_T^{L_T}$$

high-level feature

deep learning assumption: from low level to high level embedding, feature representation captures more "abstract", "conceptual" information.

TL/MTL assumption: data from D_S and D_T is similar often in the sense of "abstract information".

TL/MTL ideas

② adapt model trained on D_S to D_T:

1) fine-tuning: take a pre-trained model, e.g. VGG network, then fine-tuning (a subset of) parameters and/or add model to the previous one.

See Yoshuski et al. NIPS 2014 for an empirical study of transferability of different layers.

2) lazy TL (Liu & Fukumizu on arxiv, 2015)

assume we have trained a classifier $\hat{p}_S(y|x)$ on D_S, then the new classifier on D_T can be decomposed to

estimator of $p_S(y|x)$

data distribution

$$\hat{p}_T(y|x) = \frac{\hat{p}_T(y|x)}{\hat{p}_S(y|x)} \hat{p}_S(y|x)$$

now assume we already have $\hat{p}_S(y|x)$ and might not be able to modify it (e.g. for privacy reasons),

then we can parameterize $g(y, x; \theta) = \dfrac{\hat{p}_T(y|x)}{\hat{p}_S(y|x)}$

and minimize $KL[p_T(y|x) \| g(y, x; \theta)\hat{p}_S(y|x)]$

now if the true ratio $\dfrac{p_T(y, x)}{p_S(y, x)}$ is bounded, then

we can separate the problem to

$$KL[p_T(y|x) \| g(y, x; \theta)\hat{p}_S(y|x)]$$
$$\leq KL[p_T(y|x) \| g(y, x; \theta) p_S(y|x)]$$
$$+ (1+K)\underbrace{KL[p_S(y|x) \| \hat{p}_S(y|x)]}_{\text{only related to the pre-training step}}$$

and $KL[p_T(y|x) \| g(y, x; \theta) p_S(y|x)] \propto -\mathbb{E}_{p_T}[\log g(y, x; \theta)]$

use this objective to train model g

<u>after training</u>, use $g \cdot \hat{p}_S$ as the classifer on D_T

example: $g(y, x; \theta) \propto \exp[\theta^T \Phi(y, x)]$

and $\Sigma_y\, g(y, x; \theta) p_S(y|x) = 1$

and let $\hat{p}_S(y|x) \propto \exp[\beta^T \Phi(y, x)]$

then by selecting θ^T, we do feature selection on the target domain! (e.g. when $\theta_i + \beta_i = 0$, the ith feature $\Phi_i(y, x)$ is not used)

Comment: I generally like this idea although their explanation & experiments are not very convincing.