Semi-supervised Learning and Transfer Learning

Makoto Yamada
myamada@i.kyoto-u.ac.jp

Kyoto University

July/8/2019

Review: Supervised Learning

Problem formulation of supervised learning.

- Input vector: $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^\top \in \mathbb{R}^d$
- Output: $y \in \mathbb{R}$
- $(\boldsymbol{x}_i, y_i) \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}, y)$
- Labeled data: $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$
- Model: $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}$. (Linear model)

Risk: $R(\boldsymbol{w}) = \iint \text{loss}(y, f(\boldsymbol{x}; \boldsymbol{w}))p(\boldsymbol{x}, y)\mathrm{d}\boldsymbol{x}\mathrm{d}y$

Empirical Risk: $R_{emp}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \text{loss}(y_i, f(\boldsymbol{x}_i; \boldsymbol{w}))$

Empirical Risk Minimization (ERM): $\widehat{\boldsymbol{w}} = \mathrm{argmin}_{\boldsymbol{w}} \ R_{emp}(\boldsymbol{w})$

Semi-Supervised Learning

Problem formulation of semi-supervised learning.

- $(\boldsymbol{x}_i, y_i) \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}, y)$
- $\boldsymbol{x}_i \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x})$
- Labeled data: $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$
- Unlabeled data: $\{\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \ldots, \boldsymbol{x}_{n+m}\}$
- Usually $n \ll m$ and n is small
- If n is large, it is good to use supervised learning

Semi-supervised learning:

- We have both labeled and unlabeled samples.
- Semi-supervised learning uses both labeled and unlabeled samples.
- The unlabeled samples follow the same distribution of the marginal distribution of $p(\boldsymbol{x}, y)$

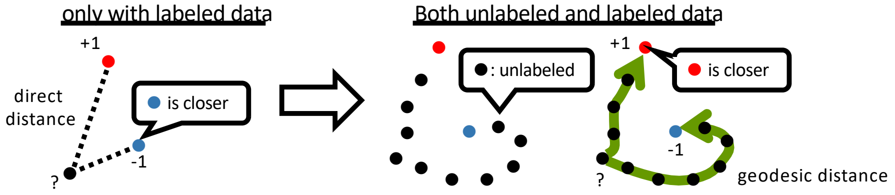
Role of unlabeled data

Data generation process

- Input $\boldsymbol{x}$ is generated by a distribution with probability density $p(\boldsymbol{x})$
- Output y for $\boldsymbol{x}$ is generated by conditional distribution with probability density $p(y|\boldsymbol{x})$.

Unlabeled data can be used for capturing $p(\boldsymbol{x})$

- input data distribution, input space metric, or better representation.



Semi-supervised learning problem: Learning with labeled and unlabeled data

We have both labeled and unlabeled instances (samples):

- Labeled data: $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$
- Unlabeled data: $\{\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \ldots, \boldsymbol{x}_{n+m}\}$

Estimate a deterministic mapping from $\boldsymbol{x}$ to y.

- Regression
- Classification

Typical approaches of semi-supervised learning

- Weighted maximum likelihood estimation
- Graph-based learning
- self-training
- Clustering
- Generative models

Weighted maximum likelihood

The original goal of ML estimation is to maximize:

$$\mathbb{E}_{\boldsymbol{x},y}[\log p(y|\boldsymbol{x})] = \iint \log P(y|\boldsymbol{x};\boldsymbol{w})p(\boldsymbol{x})p(y|\boldsymbol{x})\mathrm{d}\boldsymbol{x}\mathrm{d}y,$$
$$\approx \frac{1}{n}\sum_{i=1}^{n}\log(P(y_i|\boldsymbol{x}_i;\boldsymbol{w}))$$

where $P(y|\boldsymbol{x};\boldsymbol{w})$ is a model. Each training instance is equally weighted.

Note, ML is equivalent to maximize the negative log-likelihood function:

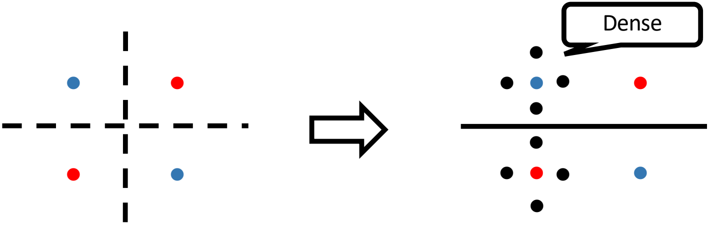
$$L(\boldsymbol{w}) = \log\left(\prod_{i=1}^{n} P(y_i|\boldsymbol{x}_i;\boldsymbol{w})\right)$$
$$\propto \frac{1}{n}\sum_{i=1}^{n}\log(P(y_i|\boldsymbol{x}_i;\boldsymbol{w}))$$

Weighted maximum likelihood

Weighted maximum likelihood:

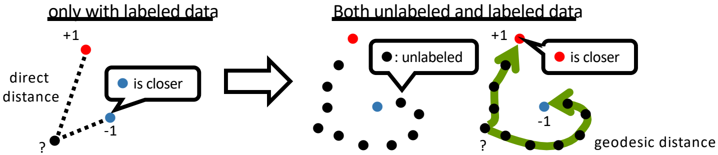
$$\max_{\boldsymbol{w}} \sum_{i=1}^{n} p(\boldsymbol{x}_i) \log(P(y_i|\boldsymbol{x}_i; \boldsymbol{w}))$$

- Each training data instance is weighted by $p(\boldsymbol{x}_i)$.
- $p(\boldsymbol{x})$ is estimated by using unlabeled data.
- Denser areas are largely weighted
- Training a classifier focusing on the dense areas



Graph-based method

- Basic idea: construct a graph capturing the intrinsic shape of input space, and make prediction on the graph.
- Assumption: Data lie on a manifold in the feature space
- The graph represent adjacency relationships among data
- K-nearest neighbor graph (e.g., $A_{ij} = 0, 1$)
- Edge-weighted graph with e.g., $A_{ij} = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2)$



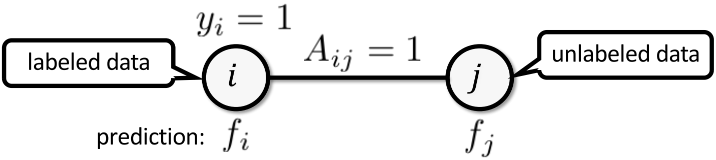
Label propagation

- Basic idea: Adjacent instances tend to have the same label
- Transductive setting (we have test instances)

$$\min_{\boldsymbol{f}\in\mathbb{R}^n} \sum_{i=1}^{n}(f_i - y_i)^2 + \lambda \sum_{i=1}^{n+m}\sum_{j=1}^{n+m} A_{ij}(f_i - f_j)^2,$$

where $\lambda > 0$ is the regularization parameter.

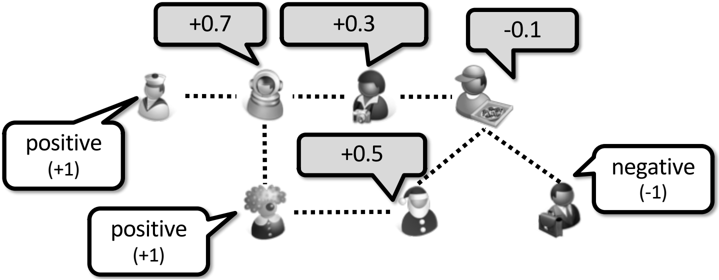
- 1st term: (squared) loss function to fit to labeled data.
- 2nd term: regularization function to make adjacent nodes to have similar predictions.



Illustrative example of label propagation

Predict if people are infected by some disease

- Test results are known for some people
- infections spread over social networks



Transfer Learning

Supervised Learning:

- Training $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$
- Test $(\boldsymbol{x}^{\mathrm{te}}, y^{\mathrm{te}}) \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x}, y)$ (Not observed during training)
- $p_{\mathrm{tr}} = p_{\mathrm{te}}$ (Training and test distributions are same)

Semi-supervised Learning:

- Training $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$, $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=n+1}^{n+m} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x})$.
- Test $(\boldsymbol{x}^{\mathrm{te}}, y^{\mathrm{te}}) \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x}, y)$ (Not observed during training)
- $p_{\mathrm{tr}} = p_{\mathrm{te}}$ (Training and test distributions are same)

If $p_{\mathrm{tr}} \neq p_{\mathrm{te}}$, supervised method and semi-supervised method do not perform well. A possible answer would be Transfer Learning!

Types of Transfer Learning

Key idea: Reduce generalization error in test data. (not in training data)

Unsupervised transfer learning

- $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \stackrel{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$,
- $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}} \stackrel{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x})$, $n_{\mathrm{tr}} \ll n_{\mathrm{te}}$

Supervised transfer learning

- $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \stackrel{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$
- $\{(\boldsymbol{x}_j^{\mathrm{te}}, y_j^{\mathrm{te}})\}_{j=1}^{n_{\mathrm{te}}} \stackrel{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x}, y)$, $n_{\mathrm{te}} \ll n_{\mathrm{tr}}$

Semi-supervised transfer learning

- $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \stackrel{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$
- $\{(\boldsymbol{x}_j^{\mathrm{te}}, y_j^{\mathrm{te}})\}_{j=1}^{n_{\mathrm{te}}} \stackrel{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x}, y)$, $n_{\mathrm{te}} \ll n_{\mathrm{tr}}$
- $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=n_{\mathrm{te}}+1}^{n_{\mathrm{te}}+n'_{\mathrm{te}}} \stackrel{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x})$, $n_{\mathrm{tr}} \ll n_{\mathrm{te}}$

Unsupervised Transfer Learning

Key idea: We assume

- It does not need to have test label
- Need some assumption

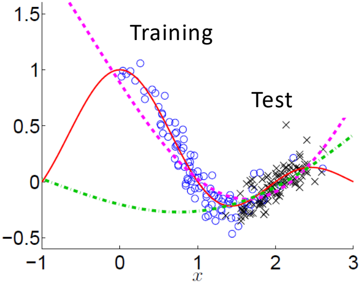
Standard approaches

- Importance weighted method (e.g., Covariate shift adaptation)
- Subspace based method.

Problem setup:

- $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$,
- $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x})$, $n_{\mathrm{tr}} \ll n_{\mathrm{te}}$

Key idea: Learning a function so that error in test data is minimized under the assumption $p_{\mathrm{tr}}(y|\boldsymbol{x}) = p_{\mathrm{te}}(y|\boldsymbol{x})$



Unsupervised Transfer Learning

The risk can be written as

$$\begin{aligned}
J(\boldsymbol{w}) &= \iint L(y, f(\boldsymbol{x})) p_{\mathrm{te}}(\boldsymbol{x}, y) \mathrm{d}\boldsymbol{x} \mathrm{d}y \\
&= \iint L(y, f(\boldsymbol{x})) \frac{p_{\mathrm{te}}(\boldsymbol{x}, y)}{p_{\mathrm{tr}}(\boldsymbol{x}, y)} p_{\mathrm{tr}}(\boldsymbol{x}, y) \mathrm{d}\boldsymbol{x} \mathrm{d}y \\
&= \iint L(y, f(\boldsymbol{x})) \frac{p_{\mathrm{te}}(y|\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(y|\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x})} p_{\mathrm{tr}}(y, \boldsymbol{x}) \mathrm{d}\boldsymbol{x} \mathrm{d}y \\
&= \iint L(y, f(\boldsymbol{x})) \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} p_{\mathrm{tr}}(y, \boldsymbol{x}) \mathrm{d}\boldsymbol{x} \mathrm{d}y \\
&\approx \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} L(y_i^{\mathrm{tr}}, f(\boldsymbol{x}_i^{\mathrm{tr}})) \frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})}
\end{aligned}$$

Actually, it is a weighted maximum likelihood problem. Note $\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})}$ is a ratio of probability densities (density-ratio)

Unsupervised Transfer Learning: Covariate shift adaptation

Exponentially-flattened Importance weighted empirical risk minimization (IW-ERM):

$$\min_{f \in \mathcal{F}} \quad \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} L(y_i^{\mathrm{tr}}, f(\boldsymbol{x}_i^{\mathrm{tr}})) \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\tau}$$

where $0 \leq \tau \leq 1$ is a tuning parameter for stabilizing the covariate shift adaptation.

- $\tau = 0 \rightarrow$ ERM
- $0 < \tau < 1 \rightarrow$ Intermediate
- $\tau = 1$ IW-ERM

Setting τ to $0 < \tau < 1$ is practically useful.

Unsupervised Transfer Learning: Covariate shift adaptation

Relative Importance weighted empirical risk minimization (RIW-ERM):

$$\min_{f \in \mathcal{F}} \quad \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} L(y_i^{\mathrm{tr}}, f(\boldsymbol{x}_i^{\mathrm{tr}})) \frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{(1-\alpha)p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}}) + \alpha p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})}$$

where $0 \leq \tau \leq 1$ is a tuning parameter for stabilizing the covariate shift adaptation.

- $\alpha = 0 \rightarrow$ ERM
- $0 < \alpha < 1 \rightarrow$ Intermediate
- $\alpha = 1$ IW-ERM

$$r_\alpha(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{(1-\alpha)p_{\mathrm{tr}}(\boldsymbol{x}) + \alpha p_{\mathrm{tr}}(\boldsymbol{x})} < \frac{1}{1-\alpha}$$

The density ratio is bounded above by $1/(1-\alpha)$.

Unsupervised Transfer Learning: Importance Weighted Least Squares

The importance weighted least squares problem can be written as

$$\min_{\boldsymbol{w}} \quad J(\boldsymbol{w}) = \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} r(\boldsymbol{x}_i^{\mathrm{tr}}) \| y_i^{\mathrm{tr}} - \boldsymbol{w}^\top \boldsymbol{x}_i^{\mathrm{tr}} \|_2^2,$$

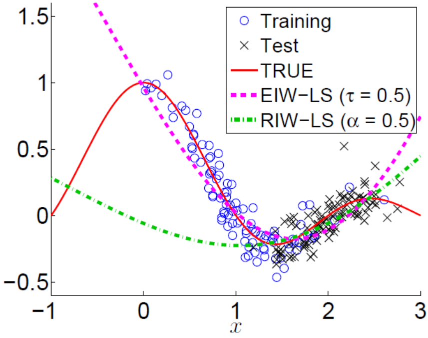
where $r(\boldsymbol{x})$ is a weight function (e.g., density-ratio).

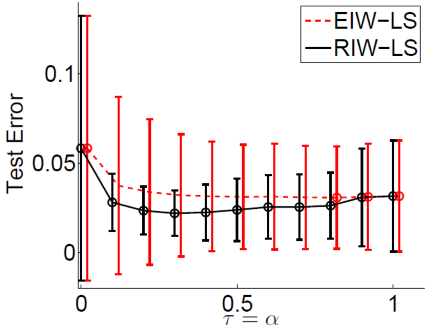
Take the derivative w.r.t. $\boldsymbol{w}$ and equating it to zero.

$$\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\frac{2}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} r(\boldsymbol{x}_i^{\mathrm{tr}})(y_i^{\mathrm{tr}} - \boldsymbol{w}^\top \boldsymbol{x}_i^{\mathrm{tr}}) \boldsymbol{x}_i^{\mathrm{tr}} = \boldsymbol{0}$$

$$\widehat{\boldsymbol{w}} = \left(\sum_{i=1}^{n_{\mathrm{tr}}} r(\boldsymbol{x}_i^{\mathrm{tr}}) \boldsymbol{x}_i^{\mathrm{tr}} {\boldsymbol{x}_i^{\mathrm{tr}}}^\top \right)^{-1} \sum_{i=1}^{n_{\mathrm{tr}}} r(\boldsymbol{x}_i^{\mathrm{tr}}) y_i^{\mathrm{tr}} \boldsymbol{x}_i^{\mathrm{tr}}$$

Comparison of EIW-LS and RIW-LS:





Supervised Transfer Learning

Problem formulation:

- $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$
- $\{(\boldsymbol{x}_j^{\mathrm{te}}, y_j^{\mathrm{te}})\}_{j=1}^{n_{\mathrm{te}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x}, y)$, $n_{\mathrm{te}} \ll n_{\mathrm{tr}}$

We assume to have a large number of training samples and a small number of paired target labeled samples.

- Frustratingly easy domain adaptation
- Multi-task Learning
- Fine-tuning (Deep Learning)

Supervised Transfer Learning: Importance Weight

Naive approach: Pooling training and test samples

$$
\begin{aligned}
J(\boldsymbol{w}) &= \iint \mathrm{loss}(y, f(\boldsymbol{x}; \boldsymbol{w})) p_{\mathrm{te}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \\
&= \alpha \iint \mathrm{loss}(y, f(\boldsymbol{x}; \boldsymbol{w})) p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \\
&\quad + (1-\alpha) \iint \mathrm{loss}(y, f(\boldsymbol{x}; \boldsymbol{w})) p_{\mathrm{te}}(\boldsymbol{x}, \boldsymbol{y}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \\
&\simeq \frac{\alpha}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \mathrm{loss}(y_i^{\mathrm{tr}}, f(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{w})) + \frac{(1-\alpha)}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} \mathrm{loss}(y_j^{\mathrm{te}}, f(\boldsymbol{x}_j^{\mathrm{te}}; \boldsymbol{w})),
\end{aligned}
$$

where $0 \leq \alpha \leq 1$ is a tuning parameter to control trade off between source and target errors.

Supervised Transfer Learning: Multi-task Learning

Problem formulation:

- Task1: $\{(\boldsymbol{x}_i^{(1)}, y_i^{(1)})\}_{i=1}^{n_1} \overset{\text{i.i.d.}}{\sim} p_1(\boldsymbol{x}, y)$
- Task2: $\{(\boldsymbol{x}_j^{(2)}, y_j^{(2)})\}_{j=1}^{n_2} \overset{\text{i.i.d.}}{\sim} p_2(\boldsymbol{x}, y)$
- ...
- Task$M$: $\{(\boldsymbol{x}_j^{(M)}, y_j^{(M)})\}_{j=1}^{n_M} \overset{\text{i.i.d.}}{\sim} p_M(\boldsymbol{x}, y)$
- Linear Models:
  $f_1(\boldsymbol{x}^{(1)}) = \boldsymbol{w}_1^\top \boldsymbol{x}^{(1)}, f_2(\boldsymbol{x}^{(2)}) = \boldsymbol{w}_2^\top \boldsymbol{x}^{(2)}, \ldots, f_M(\boldsymbol{x}^{(M)}) = \boldsymbol{w}_M^\top \boldsymbol{x}^{(M)}$

$$\min_{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M} \quad \sum_{m=1}^{M} \frac{1}{n_m} \sum_{i=1}^{n_m} \mathrm{loss}(y_i^{(m)}, f_m(\boldsymbol{x}^{(m)})) + \lambda R(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M).$$

where $R(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M)$ is a regularizer.

- $\lambda = 0$: Independently optimize $\boldsymbol{w}$s
- $\lambda > 0$: We share some information among models.

Supervised Transfer Learning: Multi-task Learning

Multi-task learning optimization (Graph-Laplacian).

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_M} \quad \sum_{m=1}^{M} \frac{1}{n_m} \sum_{i=1}^{n_m} \mathrm{loss}(y_i^{(m)}, f_m(\boldsymbol{x}_i^{(m)})) + \lambda \sum_{m=1}^{M} \sum_{m'=1}^{M} r_{m,m'} \|\boldsymbol{w}_m - \boldsymbol{w}_{m'}\|_2^2.$$

where $r_{m,m'} \geq 0$ is a model parameter (similarity between models). If $r_{m,m'} > 0$, we make $\boldsymbol{w}_m$ and $\boldsymbol{w}_{m'}$ close.

Supervised Transfer Learning: Multi-task Learning

Other approach: Explicitly including shared parameter. We decompose
$\boldsymbol{w}_m = \boldsymbol{w}_0 + \boldsymbol{v}_m$
That is

- $f_1(\boldsymbol{x}^{(1)}) = (\boldsymbol{w}_0 + \boldsymbol{v}_1)^\top \boldsymbol{x}^{(1)}$,
- $f_2(\boldsymbol{x}^{(2)}) = (\boldsymbol{w}_0 + \boldsymbol{v}_2)^\top \boldsymbol{x}^{(2)}$,
- ...
- $f_M(\boldsymbol{x}^{(M)}) = (\boldsymbol{w}_0 + \boldsymbol{v}_M)^\top \boldsymbol{x}^{(M)}$

where $\boldsymbol{w}_0$ is a common factor for all models.

For squared-loss, we can write the problem as

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_M} \quad \frac{1}{2}\sum_{m=1}^{M}\frac{1}{n_m}\sum_{i=1}^{n_m}\left(y_i^{(m)} - (\boldsymbol{w}_0+\boldsymbol{v}_m)^\top \boldsymbol{x}_i^{(m)}\right)^2 + \lambda(\|\boldsymbol{w}_0\|_2^2 + \sum_{m=1}^{M}\|\boldsymbol{v}_m\|_2^2)$$

Supervised Transfer Learning: Frustratingly easy domain adaptation

A frustratingly easy feature augmentation approach:

$$\boldsymbol{z}^{\mathrm{tr}} = (\boldsymbol{x}^{\mathrm{tr}\top} \;\; \boldsymbol{x}^{\mathrm{tr}\top} \;\; \boldsymbol{0}_{\mathrm{d}}{}^{\top})^{\top},$$
$$\boldsymbol{z}^{\mathrm{te}} = (\boldsymbol{x}^{\mathrm{te}\top} \;\; \boldsymbol{0}_{\mathrm{d}}{}^{\top} \;\; \boldsymbol{x}^{\mathrm{te}\top})^{\top},$$

The inner product of $\boldsymbol{z}$ in the same domain is give as

$$\boldsymbol{z}^{\mathrm{tr}\top}\boldsymbol{z}^{\mathrm{tr}} = 2\boldsymbol{x}^{\mathrm{tr}\top}\boldsymbol{x}^{\mathrm{tr}},$$
$$\boldsymbol{z}^{\mathrm{te}\top}\boldsymbol{z}^{\mathrm{te}} = 2\boldsymbol{x}^{\mathrm{te}\top}\boldsymbol{x}^{\mathrm{te}},$$

while we have

$$\boldsymbol{z}^{\mathrm{tr}\top}\boldsymbol{z}^{\mathrm{te}} = \boldsymbol{x}^{\mathrm{tr}\top}\boldsymbol{x}^{\mathrm{tr}},.$$

Then, we train a supervised learning method with the transformed vectors $\boldsymbol{z}$. Super easy!!!!

Supervised Transfer Learning: Multi-task Learning

Actually, supervised transfer learning can be regarded as a two-task learning problem. First task is for training and second task is for test. Let us denote the transformed vectors as

$$\boldsymbol{z}^{\mathrm{tr}} = (\boldsymbol{x}^{\mathrm{tr}\top} \;\; \boldsymbol{x}^{\mathrm{tr}\top} \;\; \mathbf{0}_{\mathrm{d}}{}^{\top})^{\top} \in \mathbb{R}^{3d},$$
$$\boldsymbol{z}^{\mathrm{te}} = (\boldsymbol{x}^{\mathrm{te}\top} \;\; \mathbf{0}_{\mathrm{d}}{}^{\top} \;\; \boldsymbol{x}^{\mathrm{te}\top})^{\top} \in \mathbb{R}^{3d},$$

where $\mathbf{0}_{\mathrm{d}} \in \mathbb{R}^{d}$ is the vector whose elements are all zero.
And, we consider a linear regression problem: The model parameter of the linear model can be written as

$$\boldsymbol{w} = (\boldsymbol{w}_0^{\top} \;\; \boldsymbol{v}_1{}^{\top} \;\; \boldsymbol{v}_2{}^{\top})^{\top} \in \mathbb{R}^{3d}$$

Supervised Transfer Learning: Multi-task Learning

$$
\begin{aligned}
J(\boldsymbol{w}) &= \frac{1}{2n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\|y_i^{\mathrm{tr}} - \boldsymbol{z}_i^{\mathrm{tr}\top}\boldsymbol{w}\|_2^2 + \frac{1}{2n_{\mathrm{te}}}\sum_{i=1}^{n_{\mathrm{te}}}\|y_i^{\mathrm{te}} - \boldsymbol{z}_i^{\mathrm{te}\top}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2 \\
&= \frac{1}{2}\sum_{m=1}^{M}\frac{1}{n_m}\sum_{i=1}^{n_m}\left(y_i^{(m)} - (\boldsymbol{w}_0+\boldsymbol{v}_m)^\top\boldsymbol{x}_i^{(m)}\right)^2 + \lambda(\|\boldsymbol{w}_0\|_2^2 + \sum_{m=1}^{M}\|\boldsymbol{v}_m\|_2^2),
\end{aligned}
$$

where we use

$$
\begin{aligned}
\boldsymbol{w}^\top\boldsymbol{z}^{\mathrm{tr}} &= (\boldsymbol{w}_0 + \boldsymbol{v}_1)^\top\boldsymbol{x}^{\mathrm{tr}}, \quad \boldsymbol{w}^\top\boldsymbol{z}^{\mathrm{te}} = (\boldsymbol{w}_0 + \boldsymbol{v}_2)^\top\boldsymbol{x}^{\mathrm{te}} \\
\boldsymbol{x}^{\mathrm{tr}} &= \boldsymbol{x}^{(1)}, \quad \boldsymbol{x}^{\mathrm{te}} = \boldsymbol{x}^{(2)}, \\
\|\boldsymbol{w}\|_2^2 &= \|\boldsymbol{w}_0\|_2^2 + \sum_{m=1}^{2}\|\boldsymbol{v}_m\|_2^2.
\end{aligned}
$$

Frustratingly easy domain adaptation is a multi-task learning.

Summary

- Semi-supervised learning. Use unlabeled samples and assume the data distribution of unlabeled data is same as training.
- Weighted Maximum Likelihood, Graph-based method.
- Transfer Learning. Use samples from test data. Training and test distributions are different.
- Covariate shift adaptation, frustratingly easy domain adaptation.