http://goo.gl/svgV0n

KYOTO UNIVERSITY

*Statistical Machine Learning Theory*

Lecture 10

# Semi-supervised, Active, and Transfer Learning

Hisashi Kashima
kashima@i.Kyoto-u.ac.jp

DEPARTMENT OF INTELLIGENCE SCIENCE
AND TECHNOLOGY

## Topics:
## Semi-supervised, active, and transfer learning

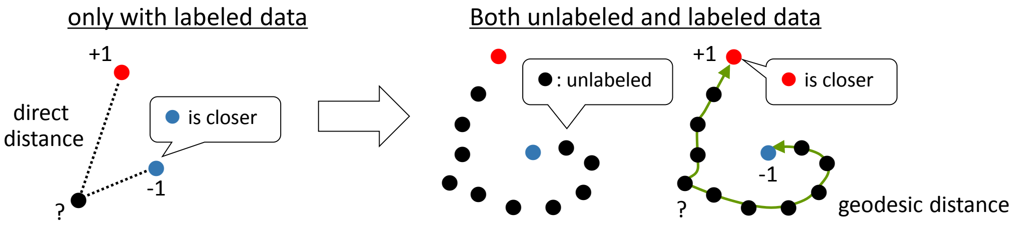
- Semi-supervised learning
  - Weighted maximum likelihood estimation
  - Graph-based methods (e.g. label propagation)
  - Self-training
- Active learning
  - Uncertainty sampling
  - Estimated model change
- Transfer learning
  - Covariate shift using with weighted ML estimation
  - Shared parameters and domain specific parameters

## Semi-supervised learning and active learning: Learning with labeled and unlabeled data

- We have both labeled and unlabeled instances
  - Labeled data: $\left\{ \left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \dots, \left(\boldsymbol{x}^{(N)}, y^{(N)}\right) \right\}$
  - Unlabeled data: $\left\{ \boldsymbol{x}^{(N+1)}, \dots, \boldsymbol{x}^{(N+M)} \right\}$
  - Usually, $N \ll M$
- Semi-supervised learning uses unlabeled data as well as labeled data
- Active learning
  - has accesses to an oracle to give labels to unlabeled data
  - has to choose which unlabeled data to query next

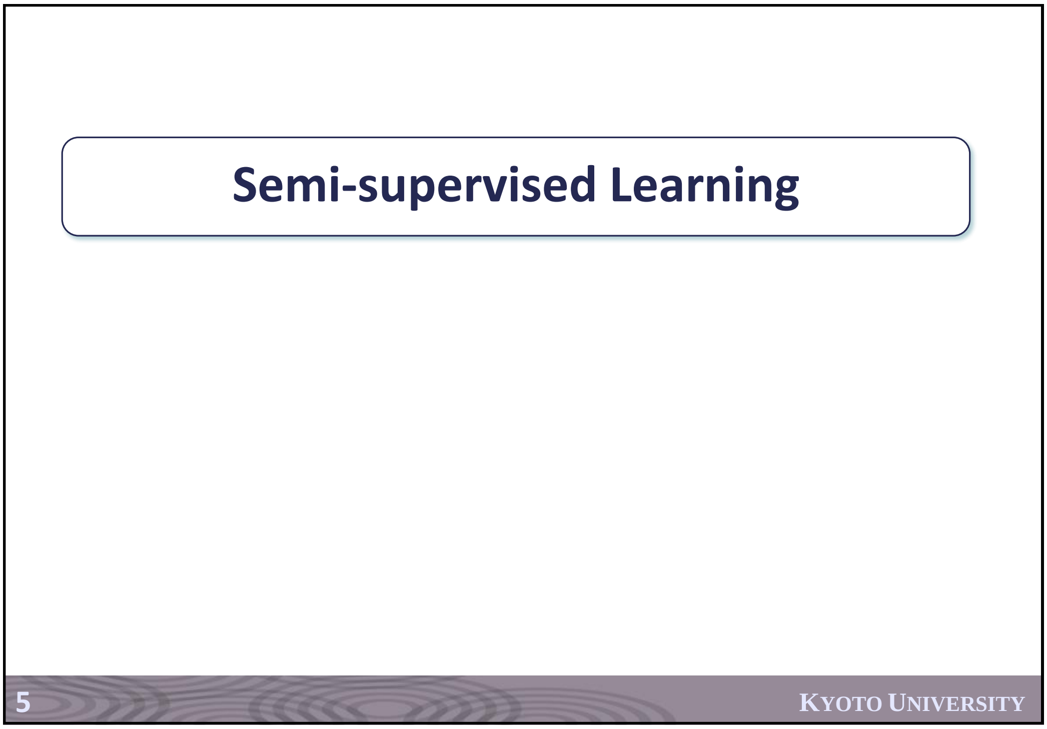
## Role of unlabeled data in supervised learning: Information of the input data distribution

- Data generation process
  - Input $\boldsymbol{x}$ is generated by input data distribution $\mathcal{D}_{\boldsymbol{x}}$
  - Output $y$ for $\boldsymbol{x}$ is generated by conditional distribution $\mathcal{D}_{y|\boldsymbol{x}}$
- Unlabeled data can be used for capturing $\mathcal{D}_{\boldsymbol{x}}$
  - Input data distribution, input space metric, or better representations

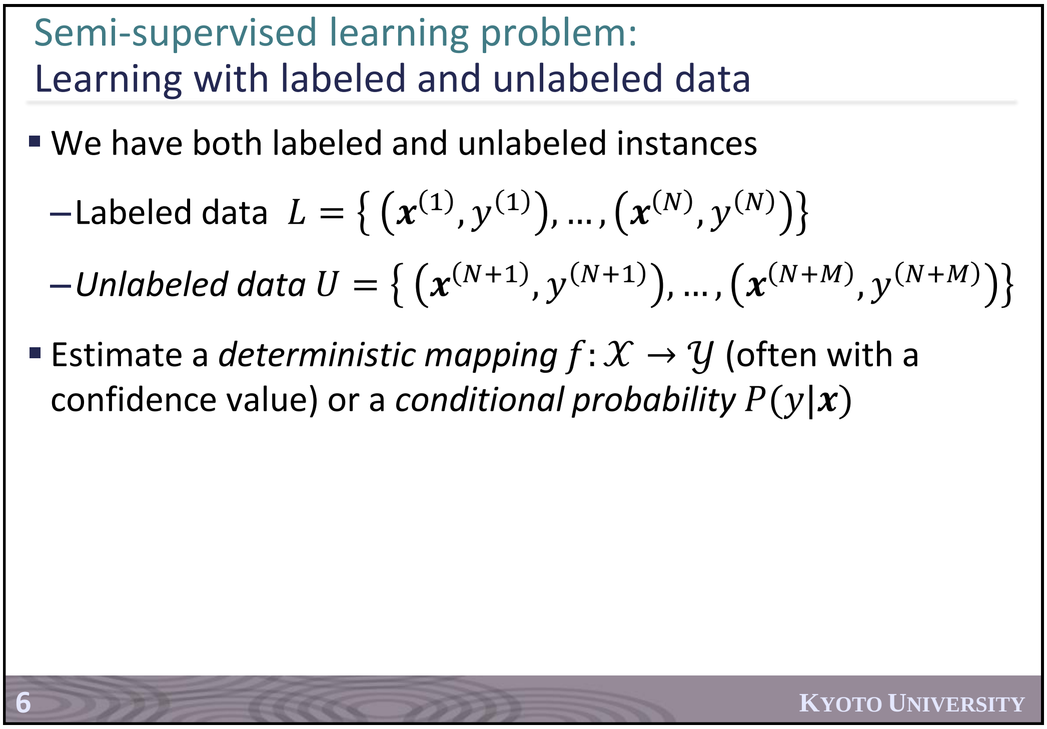


only with labeled data

Both unlabeled and labeled data

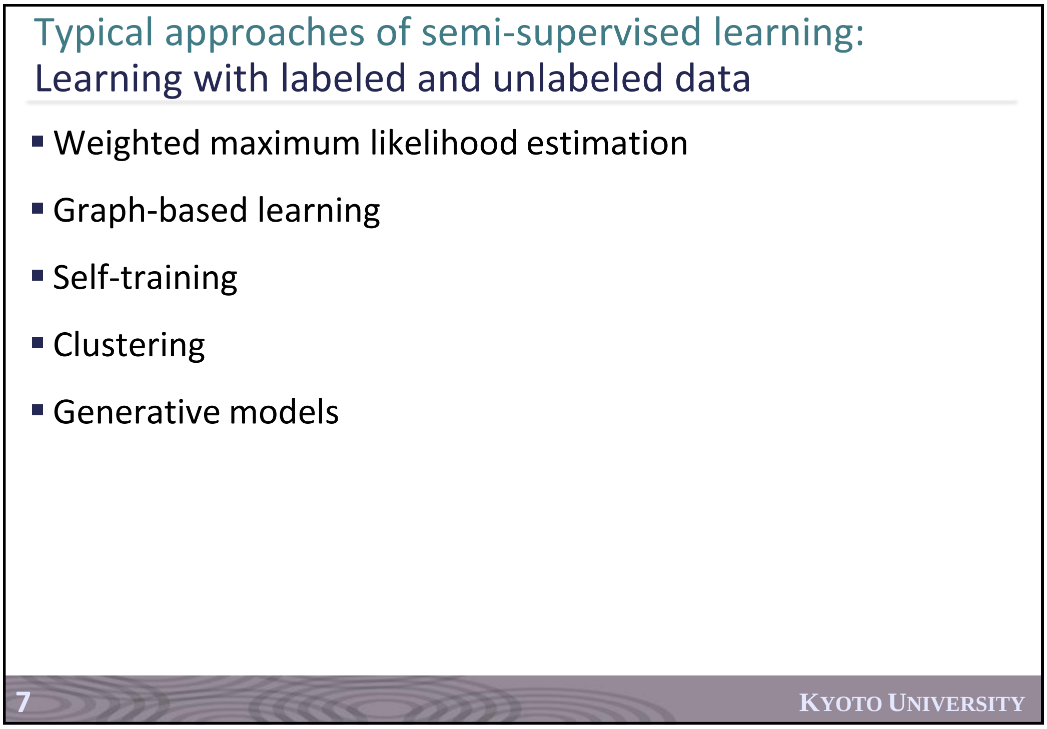


# Semi-supervised Learning



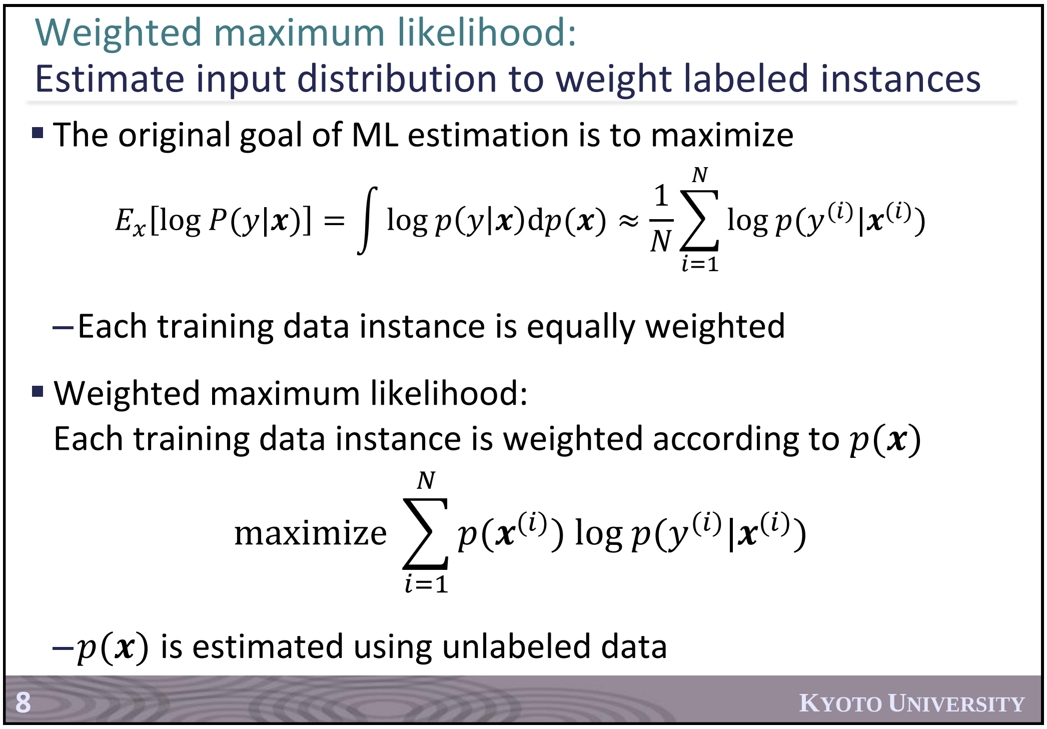
## Semi-supervised learning problem: Learning with labeled and unlabeled data

- We have both labeled and unlabeled instances
  - Labeled data $L = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \dots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$
  - *Unlabeled data* $U = \{(\boldsymbol{x}^{(N+1)}, y^{(N+1)}), \dots, (\boldsymbol{x}^{(N+M)}, y^{(N+M)})\}$
- Estimate a *deterministic mapping* $f: \mathcal{X} \to \mathcal{Y}$ (often with a confidence value) or a *conditional probability* $P(y|\boldsymbol{x})$



## Typical approaches of semi-supervised learning: Learning with labeled and unlabeled data

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



## Weighted maximum likelihood: Estimate input distribution to weight labeled instances

- The original goal of ML estimation is to maximize

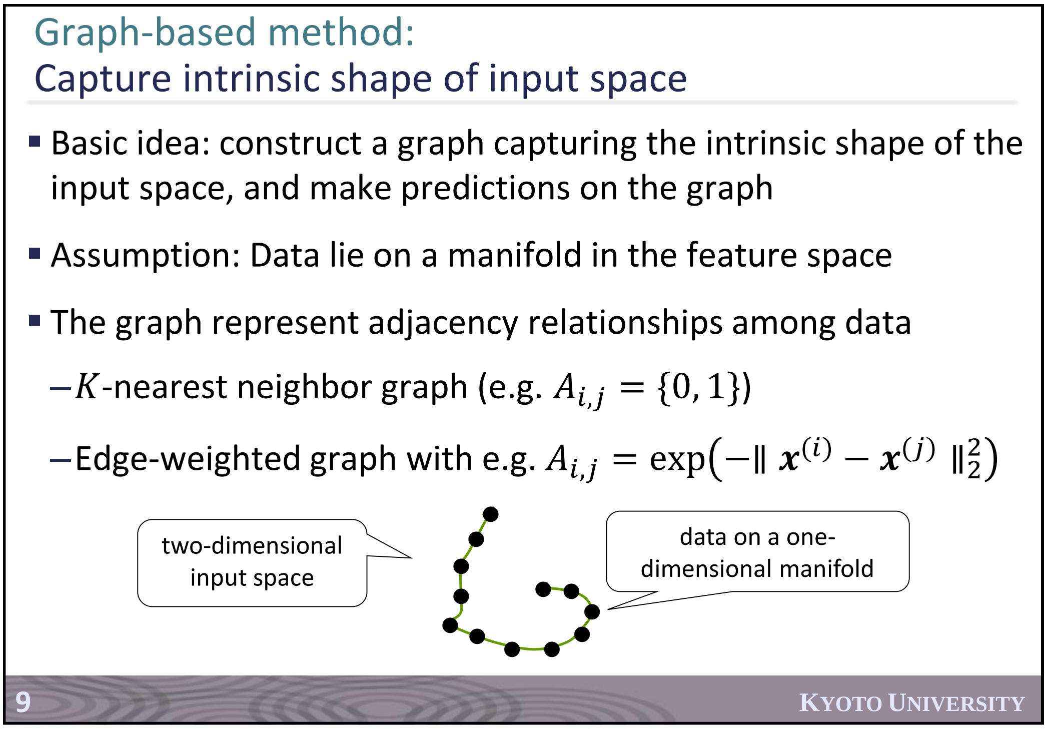
$$E_x[\log P(y|\boldsymbol{x})] = \int \log p(y|\boldsymbol{x}) \mathrm{d}p(\boldsymbol{x}) \approx \frac{1}{N}\sum_{i=1}^{N} \log p(y^{(i)}|\boldsymbol{x}^{(i)})$$

  – Each training data instance is equally weighted

- Weighted maximum likelihood:
  Each training data instance is weighted according to $p(\boldsymbol{x})$

$$\text{maximize} \sum_{i=1}^{N} p(\boldsymbol{x}^{(i)}) \log p(y^{(i)}|\boldsymbol{x}^{(i)})$$

  – $p(\boldsymbol{x})$ is estimated using unlabeled data

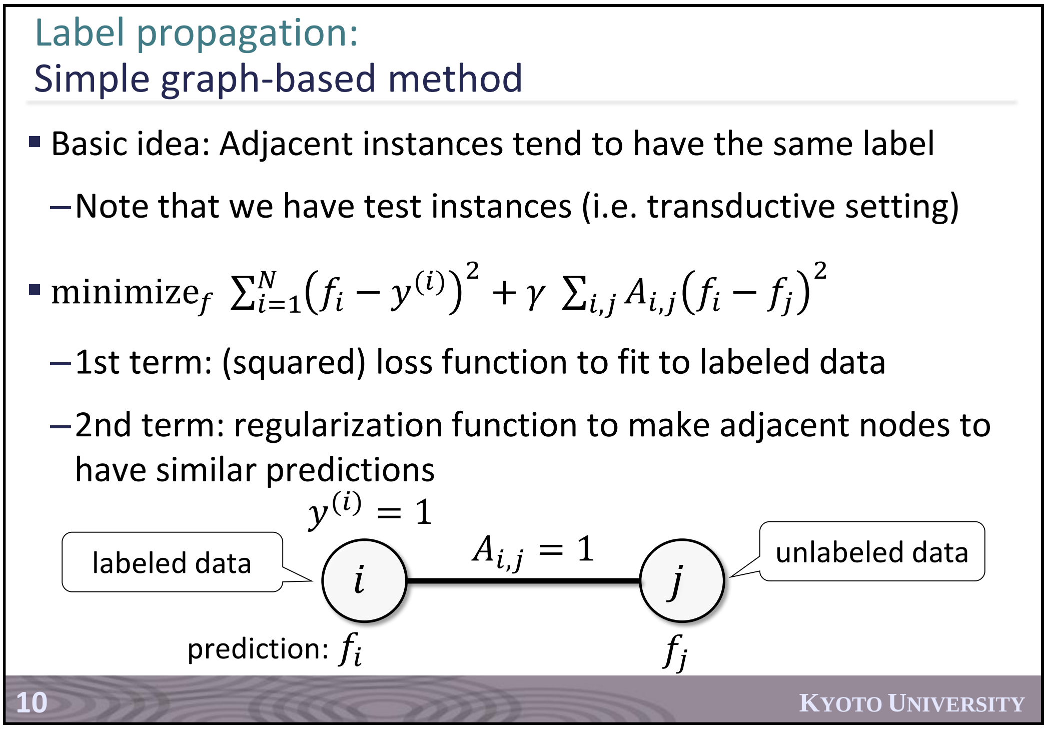


## Graph-based method: Capture intrinsic shape of input space

- Basic idea: construct a graph capturing the intrinsic shape of the input space, and make predictions 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. $A_{i,j} = \{0,1\}$)
  - Edge-weighted graph with e.g. $A_{i,j} = \exp\left(-\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \|_2^2\right)$

two-dimensional input space

data on a one-dimensional manifold



## Label propagation: Simple graph-based method

- Basic idea: Adjacent instances tend to have the same label
  - Note that we have test instances (i.e. transductive setting)
- minimize$_f$ $\sum_{i=1}^{N} \left(f_i - y^{(i)}\right)^2 + \gamma \sum_{i,j} A_{i,j} \left(f_i - f_j\right)^2$
  - 1st term: (squared) loss function to fit to labeled data
  - 2nd term: regularization function to make adjacent nodes to have similar predictions

$y^{(i)} = 1$

labeled data

$A_{i,j} = 1$

unlabeled data

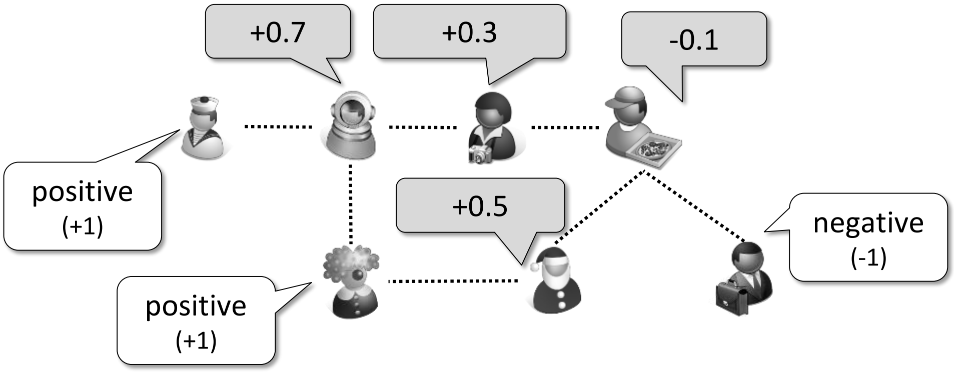
$i$ $j$

prediction: $f_i$

$f_j$

## Illustrative example of label propagation: Infection prediction on social network

- Predict if people are infected by some disease
  - Test results are known for some people
  - Infections spread over social networks

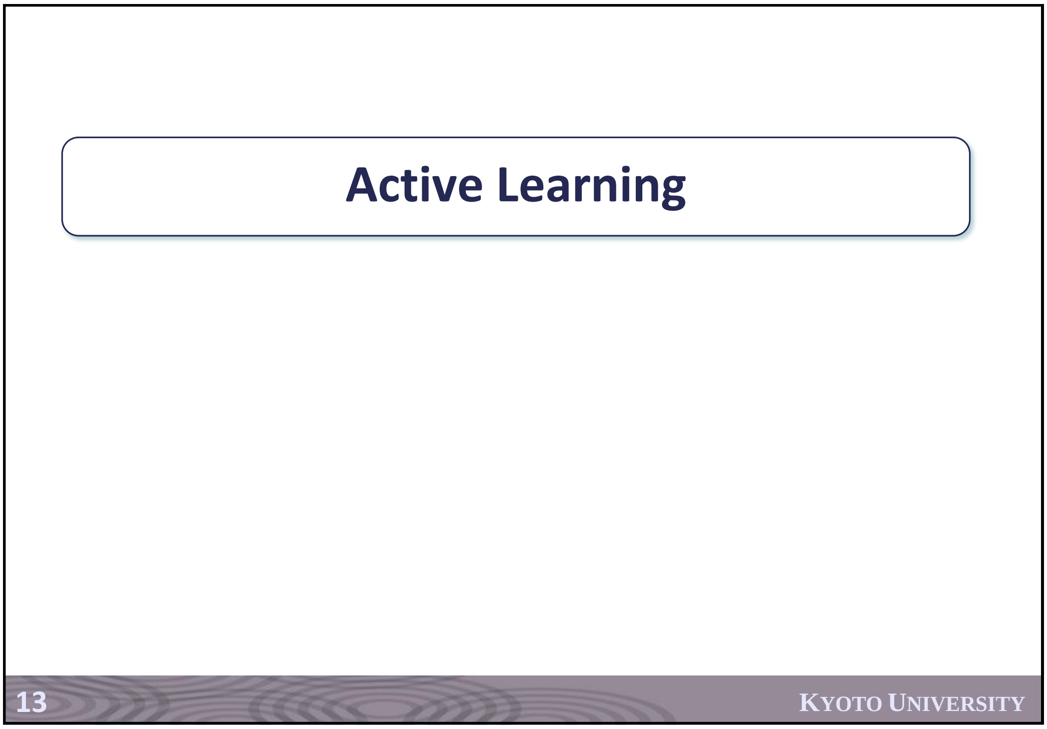


## Self-training: Believe what you believe

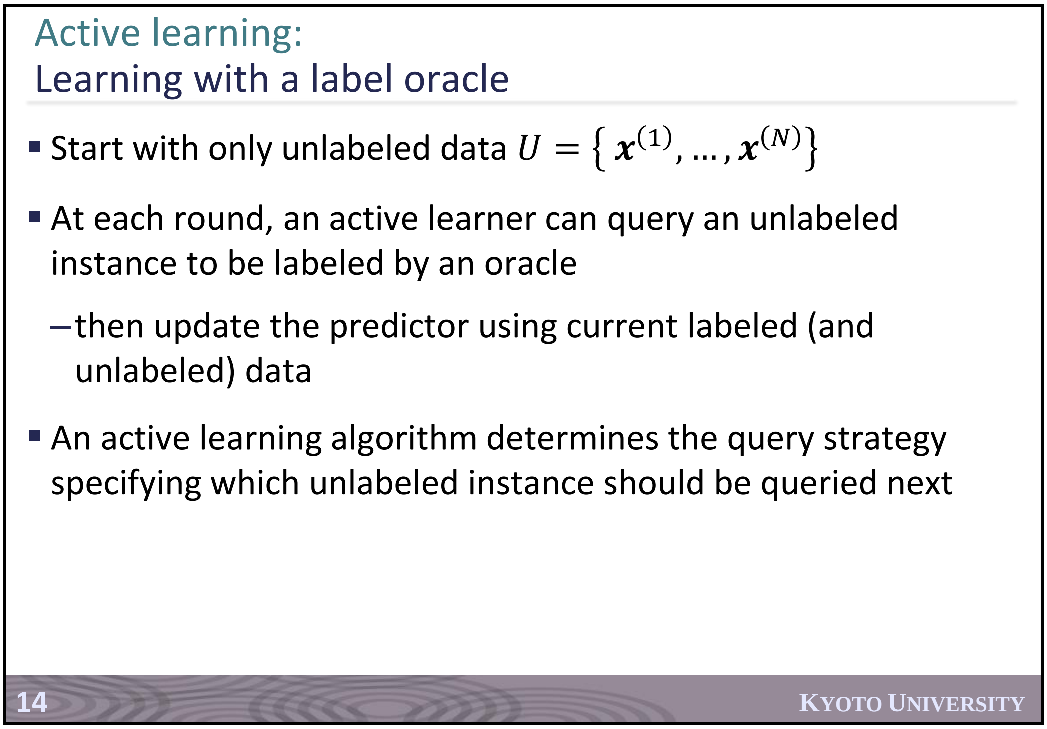
- Procedure:

1. Initialization: train a classifier using labeled dataset $L$
2. Use the classifier to assign temporary labels to unlabeled dataset $U$
3. Train a classifier using $L$ and $U$(with the temporary labels)
4. Return to Step 2

- For probabilistic classifier, use the weighted ML estimation

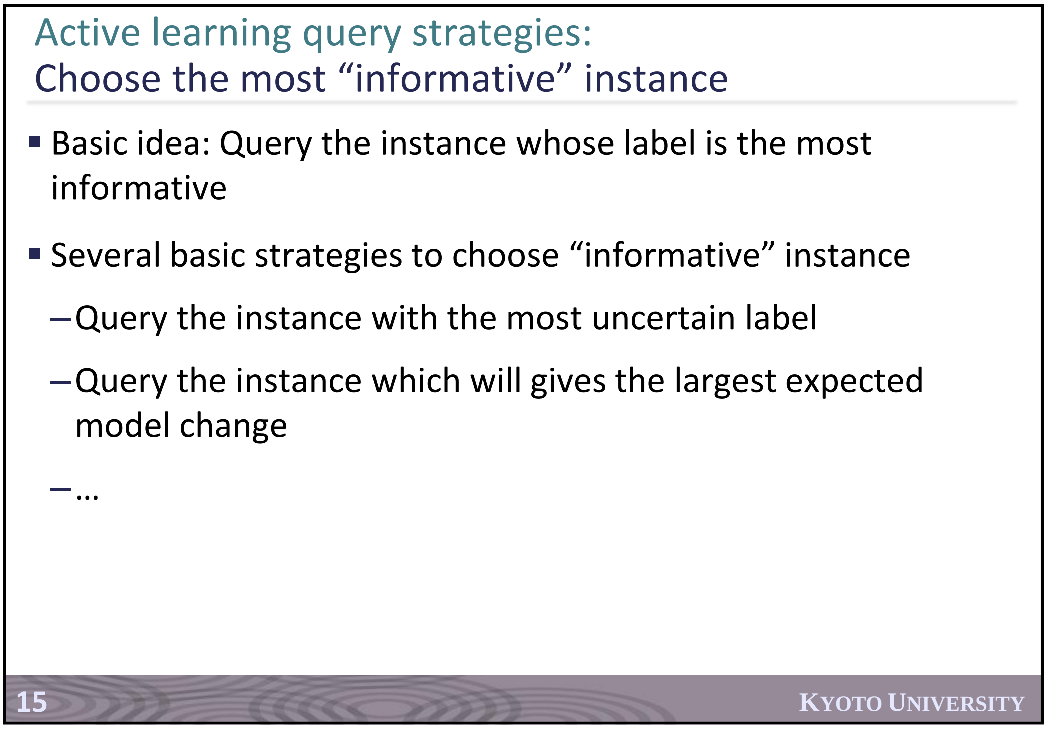


# Active Learning



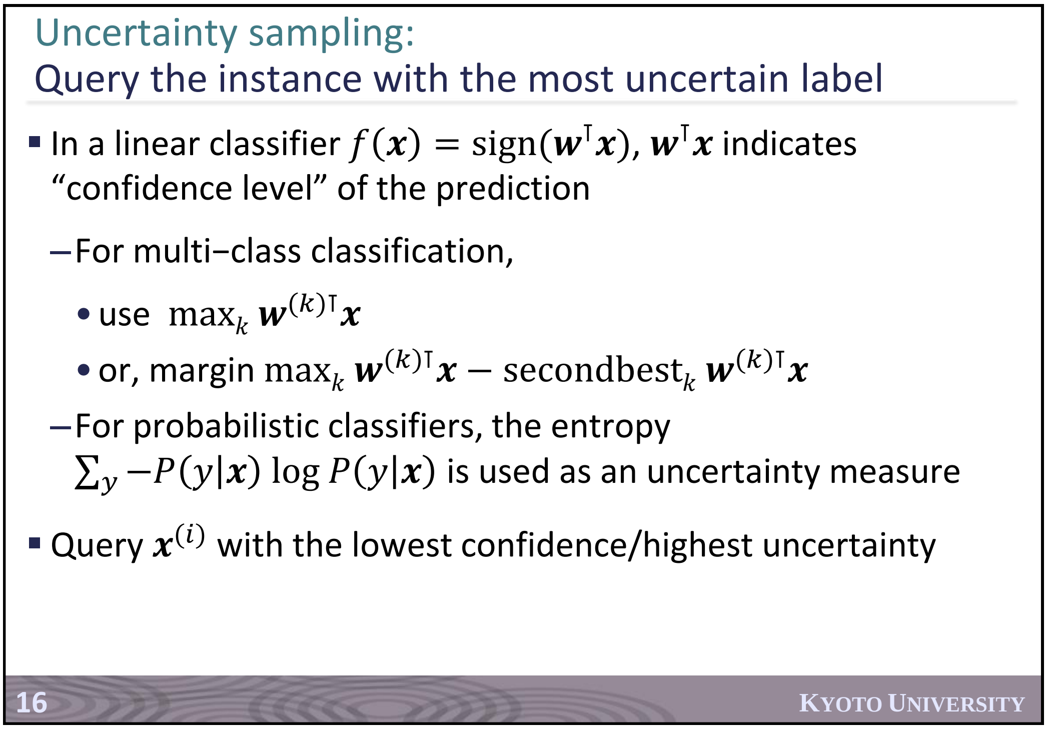
## Active learning:
## Learning with a label oracle

- Start with only unlabeled data $U = \{ \boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)} \}$
- At each round, an active learner can query an unlabeled instance to be labeled by an oracle
  - –then update the predictor using current labeled (and unlabeled) data
- An active learning algorithm determines the query strategy specifying which unlabeled instance should be queried next



## Active learning query strategies: Choose the most “informative” instance

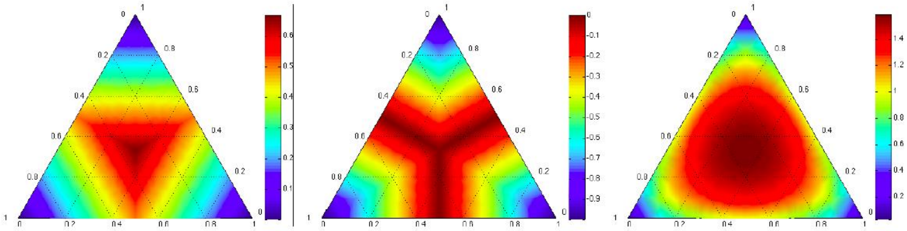
- Basic idea: Query the instance whose label is the most informative
- Several basic strategies to choose “informative” instance
  - Query the instance with the most uncertain label
  - Query the instance which will gives the largest expected model change
  - …



## Uncertainty sampling: Query the instance with the most uncertain label

- In a linear classifier $f(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x})$, $\boldsymbol{w}^\top \boldsymbol{x}$ indicates “confidence level” of the prediction
  - For multi−class classification,
    - use $\max_k \boldsymbol{w}^{(k)\top} \boldsymbol{x}$
    - or, margin $\max_k \boldsymbol{w}^{(k)\top} \boldsymbol{x} - \mathrm{secondbest}_k \boldsymbol{w}^{(k)\top} \boldsymbol{x}$
  - For probabilistic classifiers, the entropy $\sum_y -P(y|\boldsymbol{x}) \log P(y|\boldsymbol{x})$ is used as an uncertainty measure
- Query $\boldsymbol{x}^{(i)}$ with the lowest confidence/highest uncertainty

# Differences among confidence level, margin, and entropy [Settles, 2010. page 14]



(a) least confident (b) margin (c) entropy

Figure 5: Heatmaps illustrating the query behavior of common uncertainty measures in a three-label classification problem. Simplex corners indicate where one label has very high probability, with the opposite edge showing the probability range for the *other* two classes when that label has very low probability. Simplex centers represent a uniform posterior distribution. The most informative query region for each strategy is shown in dark red, radiating from the centers.

Settles, B. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. 2010.

# limitation of uncertainty sampling : Uncertainty sampling is based on local information

- Querying the least confident instance cares only about the local information
- Obtaining one labeled instance can make an impact on the whole model
- We should take the amount of the “impact” of a label into account

## Expected model change:
## Query the instance which gives the largest model change

- Choose instance $\boldsymbol{x}$ which gives the largest (expected) gradient of the objective function: $\sum_y -P(y|\boldsymbol{x}) \parallel \boldsymbol{\nabla}_{\boldsymbol{w}} J(L \cup (\boldsymbol{x}, y) \parallel$
  - Assume gradient-based learning methods are used
    - e.g. gradient descent $w^{\text{new}} \leftarrow w^{\text{old}} - \gamma\, \boldsymbol{\nabla}_{\boldsymbol{w}} J(L \cup (\boldsymbol{x}, y))$ when new labeled instance $(\boldsymbol{x}, y)$ is added to $L$
- Choose an instance which gives the largest information gain

$$\sum_{y} -P(y|\boldsymbol{x}) \sum_{i=N+1}^{N+M} \sum_{y'} P_{\boldsymbol{w}^{\text{new}}}\left(y'|\boldsymbol{x}^{(i)}\right) \log P_{\boldsymbol{w}^{\text{new}}}\left(y'|\boldsymbol{x}^{(i)}\right)$$

  - $P_{\boldsymbol{w}^{\text{new}}}$: model after update with new labeled data $(\boldsymbol{x}, y)$

# Transfer Learning

## Transfer learning: Training and test data come from different distributions

- In transfer learning, the training dataset and the test dataset are sampled from different distributions
- In transfer learning,
  - Training data come from $\mathcal{D}_{\boldsymbol{x}}^{\text{train}}$ and $\mathcal{D}_{y|\boldsymbol{x}}^{\text{train}}$
  - Test data come from $\mathcal{D}_{\boldsymbol{x}}^{\text{test}}$ and $\mathcal{D}_{y|\boldsymbol{x}}^{\text{test}}$
  - Previously we assumed an input $\boldsymbol{x}$ sampled from $\mathcal{D}_{\boldsymbol{x}}$, and an output $y$ from $\mathcal{D}_{y|\boldsymbol{x}}$
- Example: Domain adaptation
  - Text classification of general text documents and medical texts

## Covariate shift: Training and test input distributions are different

- Covariate shift: only the input distributions are different
  - $\mathcal{D}_{\boldsymbol{x}}^{\text{train}} \neq \mathcal{D}_{\boldsymbol{x}}^{\text{test}}$
  - $\mathcal{D}_{y|\boldsymbol{x}}^{\text{train}} = \mathcal{D}_{y|\boldsymbol{x}}^{\text{test}}$: conditional distributions are the same
  - Labeled training dataset and unlabeled test dataset are given
- Often occurs when label sampling is limited for some reason
  - Labels are obtained only from the targets to which some actions are taken (e.g. responses to direct mails)
  - Labels can only be taken in controlled environments (eg. in-vitro experiments)
  - Active learning controls the training distribution

## Maximum likelihood learning under covariate shift : Maximize likelihood for test input distribution

- The distribution on which we want to work well is the test input distribution $p^{\text{test}}(\boldsymbol{x})$
- In maximum likelihood estimation, we want to maximize

$$E_X^{\text{test}}[\log P(y|\boldsymbol{x})] = \int p^{\text{test}}(\boldsymbol{x}) \log p(y|\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

  – Note that the expectation is taken over $p^{\text{test}}(\boldsymbol{x})$

- However we do not have label information for test dataset

  – We can not evaluate the objective function directly

## Covariate shift learning only with training labels: Weighted maximum likelihood with density ratio

- Use of the importance sampling

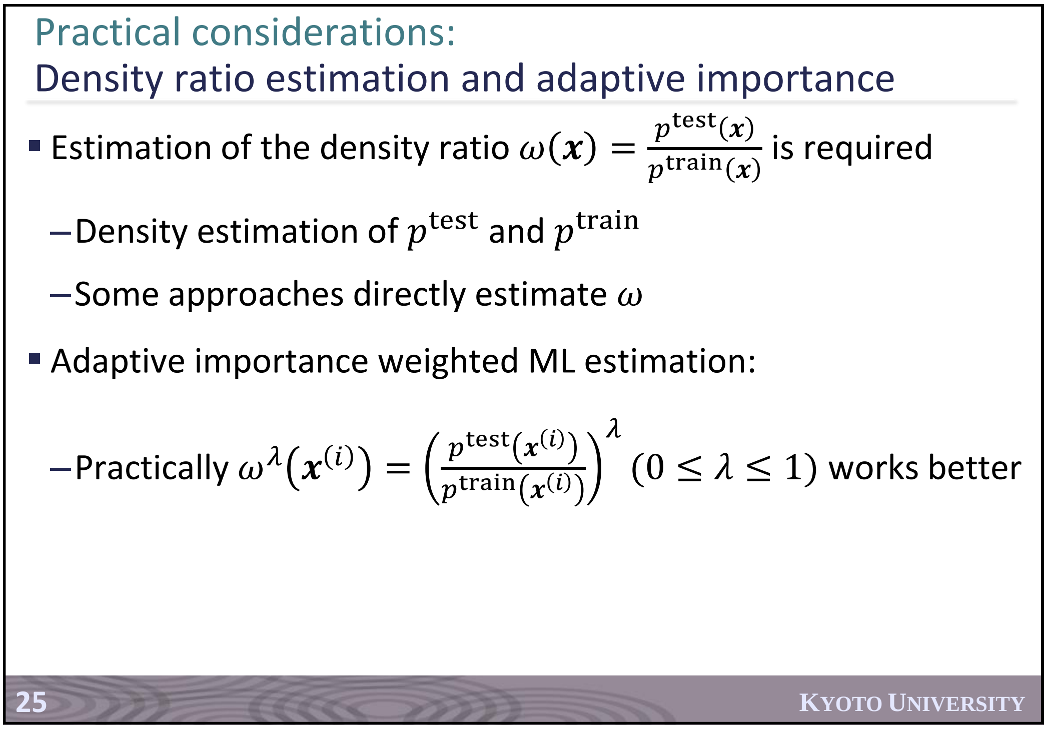
$$E_X^{\text{test}}[\log P(y|\boldsymbol{x})] = \int \frac{p^{\text{test}}(\boldsymbol{x})}{p^{\text{train}}(\boldsymbol{x})} p^{\text{train}}(\boldsymbol{x}) \log p(y|\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \frac{p^{\text{test}}\left(\boldsymbol{x}^{(i)}\right)}{p^{\text{train}}\left(\boldsymbol{x}^{(i)}\right)} \log p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}\right)$$

$$= \frac{1}{N} \sum_{i=1}^{N} \omega\left(\boldsymbol{x}^{(i)}\right) \log p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}\right)$$

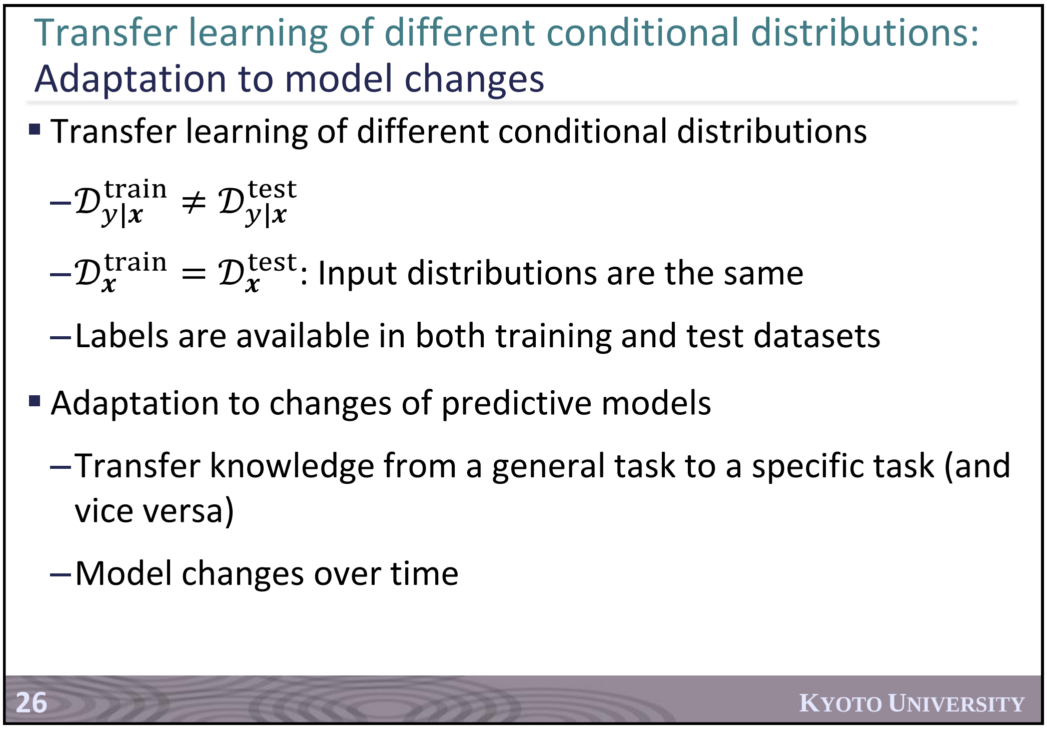
training data $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$ is weighted with $\omega\left(\boldsymbol{x}^{(i)}\right)$

  – Weighted ML estimation with weight $\omega\left(\boldsymbol{x}^{(i)}\right) = \frac{p^{\text{test}}\left(\boldsymbol{x}^{(i)}\right)}{p^{\text{train}}\left(\boldsymbol{x}^{(i)}\right)}$



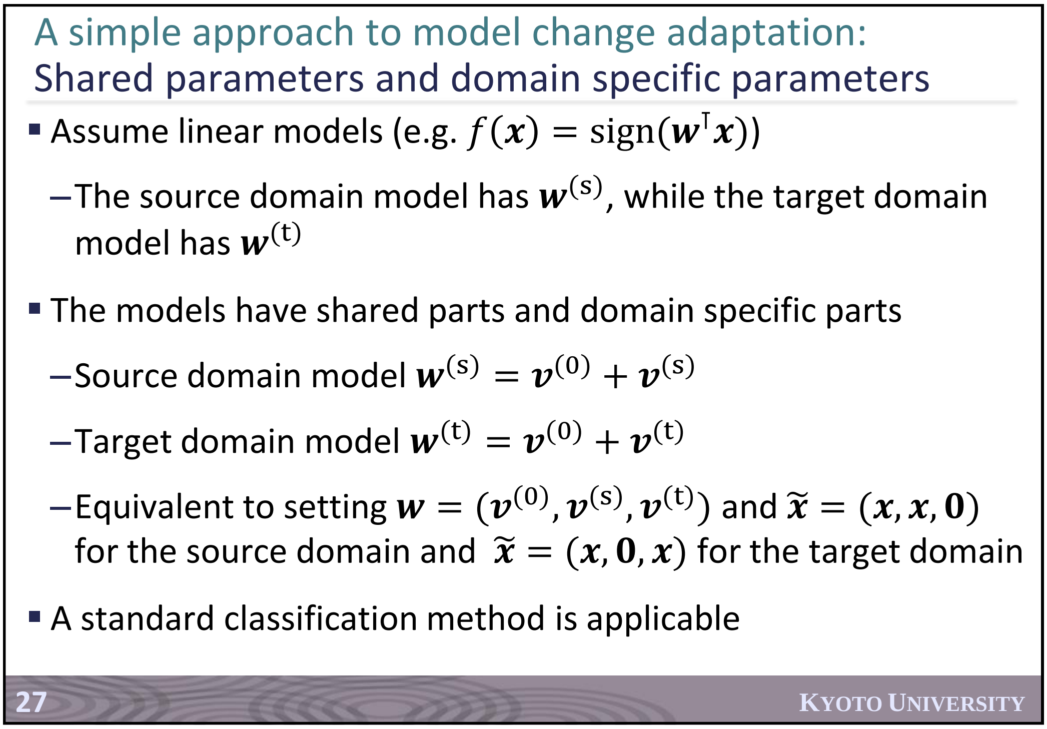
## Practical considerations:
## Density ratio estimation and adaptive importance

- Estimation of the density ratio $\omega(\boldsymbol{x}) = \frac{p^{\text{test}}(\boldsymbol{x})}{p^{\text{train}}(\boldsymbol{x})}$ is required
  - Density estimation of $p^{\text{test}}$ and $p^{\text{train}}$
  - Some approaches directly estimate $\omega$
- Adaptive importance weighted ML estimation:
  - Practically $\omega^{\lambda}\left(\boldsymbol{x}^{(i)}\right) = \left(\frac{p^{\text{test}}\left(\boldsymbol{x}^{(i)}\right)}{p^{\text{train}}\left(\boldsymbol{x}^{(i)}\right)}\right)^{\lambda}$ $(0 \leq \lambda \leq 1)$ works better



## Transfer learning of different conditional distributions:
## Adaptation to model changes

- Transfer learning of different conditional distributions
  - $\mathcal{D}_{y|\boldsymbol{x}}^{\text{train}} \neq \mathcal{D}_{y|\boldsymbol{x}}^{\text{test}}$
  - $\mathcal{D}_{\boldsymbol{x}}^{\text{train}} = \mathcal{D}_{\boldsymbol{x}}^{\text{test}}$: Input distributions are the same
  - Labels are available in both training and test datasets
- Adaptation to changes of predictive models
  - Transfer knowledge from a general task to a specific task (and vice versa)
  - Model changes over time



## A simple approach to model change adaptation: Shared parameters and domain specific parameters

- Assume linear models (e.g. $f(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x})$)
  - The source domain model has $\boldsymbol{w}^{(\mathrm{s})}$, while the target domain model has $\boldsymbol{w}^{(\mathrm{t})}$
- The models have shared parts and domain specific parts
  - Source domain model $\boldsymbol{w}^{(\mathrm{s})} = \boldsymbol{v}^{(0)} + \boldsymbol{v}^{(\mathrm{s})}$
  - Target domain model $\boldsymbol{w}^{(\mathrm{t})} = \boldsymbol{v}^{(0)} + \boldsymbol{v}^{(\mathrm{t})}$
  - Equivalent to setting $\boldsymbol{w} = (\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(\mathrm{s})}, \boldsymbol{v}^{(\mathrm{t})})$ and $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{0})$ for the source domain and $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{0}, \boldsymbol{x})$ for the target domain
- A standard classification method is applicable