https://goo.gl/kfxwEg

KYOTO UNIVERSITY

Statistical Machine Learning Theory

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

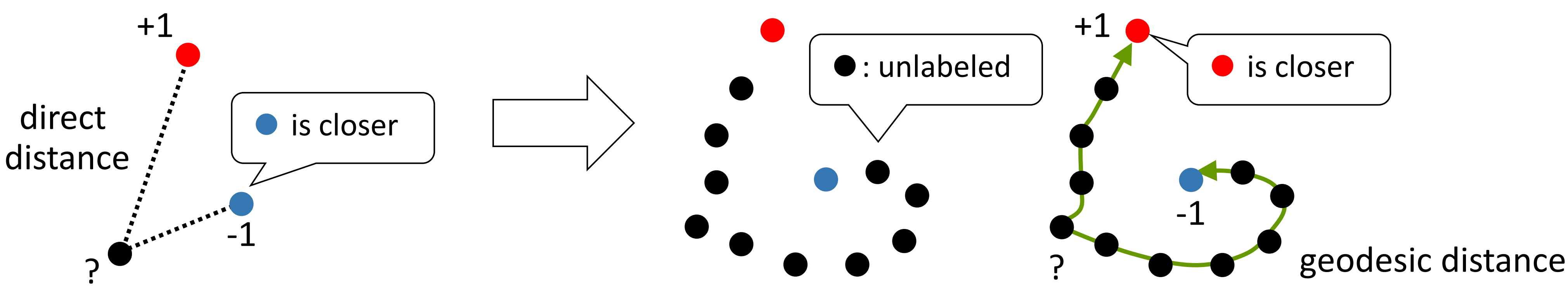
Semi-supervised learning and active learning: Learning with labeled and unlabeled data

- We have both labeled and unlabeled instances
  - Labeled data: $\left\{\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(N)}, y^{(N)}\right)\right\}$
  - Unlabeled data: $\left\{\mathbf{x}^{(N+1)}, \ldots, \mathbf{x}^{(N+M)}\right\}$
  - 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 $\mathbf{x}$ is generated by input data distribution $\mathcal{D}_{\mathcal{X}}$
  - Output y for $\mathbf{x}$ is generated by conditional distribution $\mathcal{D}_{\mathcal{Y}|\mathcal{X}}$
- Unlabeled data can be used for capturing $\mathcal{D}_{\mathcal{X}}$
  - 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 $L = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(N)}, y^{(N)}\right) \right\}$
  - *Unlabeled data* $U = \left\{\mathbf{x}^{(N+1)}, \ldots, \mathbf{x}^{(N+M)}\right\}$
- Estimate a *deterministic mapping* $f: \mathcal{X} \rightarrow \mathcal{Y}$ (often with a confidence value) or a *conditional probability* $P(y|\mathbf{x})$

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

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

  – Each training data instance is equally weighted

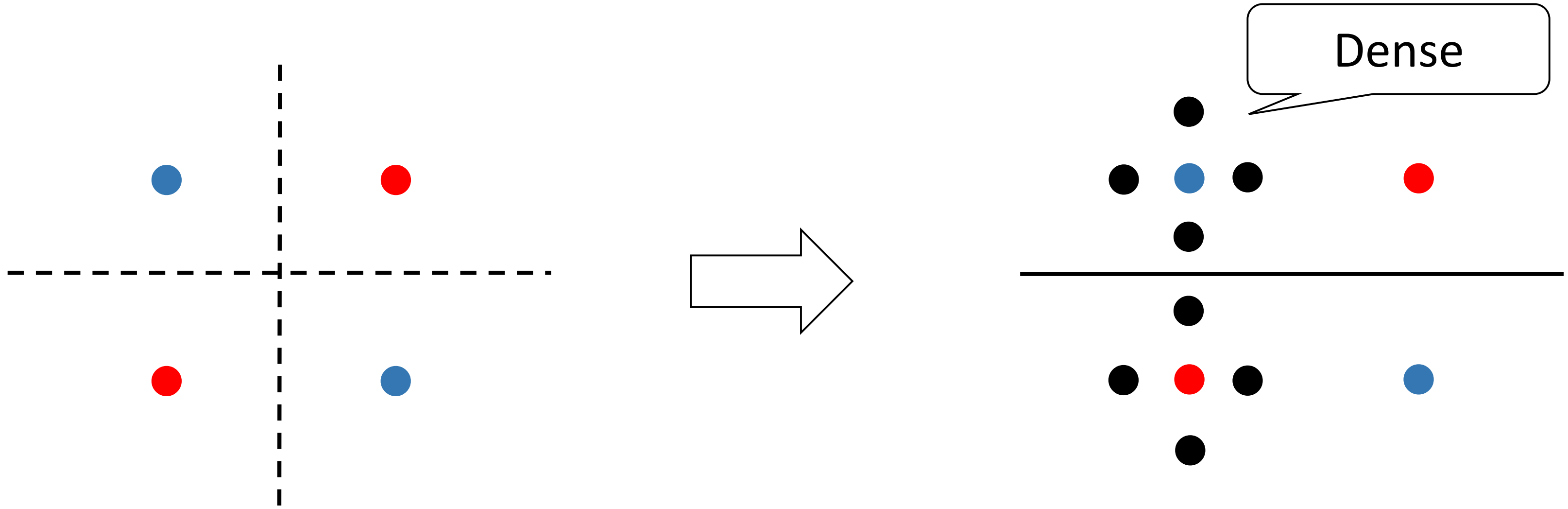
- Weighted maximum likelihood:
  Each training data instance is weighted according to $p(\mathbf{x})$

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

  – $p(\mathbf{x})$ is estimated using unlabeled data (but not practical)

Weighted maximum likelihood:
Densely distributed area are weighted larger

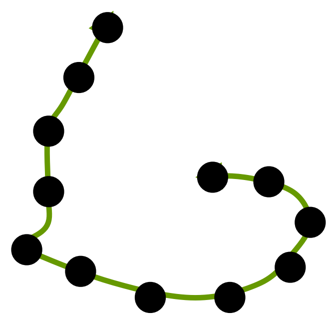
- Weighted maximum likelihood:
  - Each training data instance is weighted according to $p(\mathbf{x})$
  - Dense areas are largely weighted
  - Training a classifier focusing on the dense areas



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. $A_{i,j} = \exp\left(-\| \mathbf{x}^{(i)} - \mathbf{x}^{(j)} \|_2^2\right)$

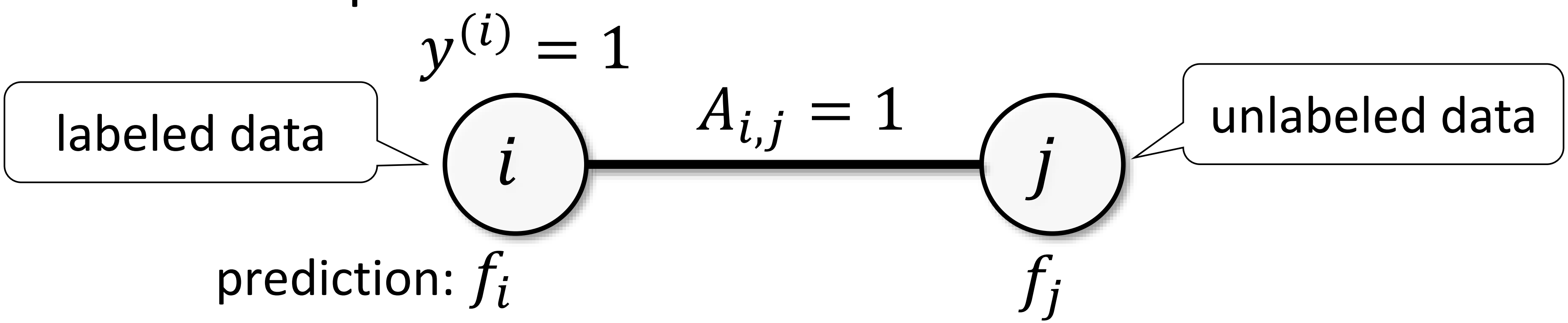
two-dimensional input space



data on a one-dimensional manifold

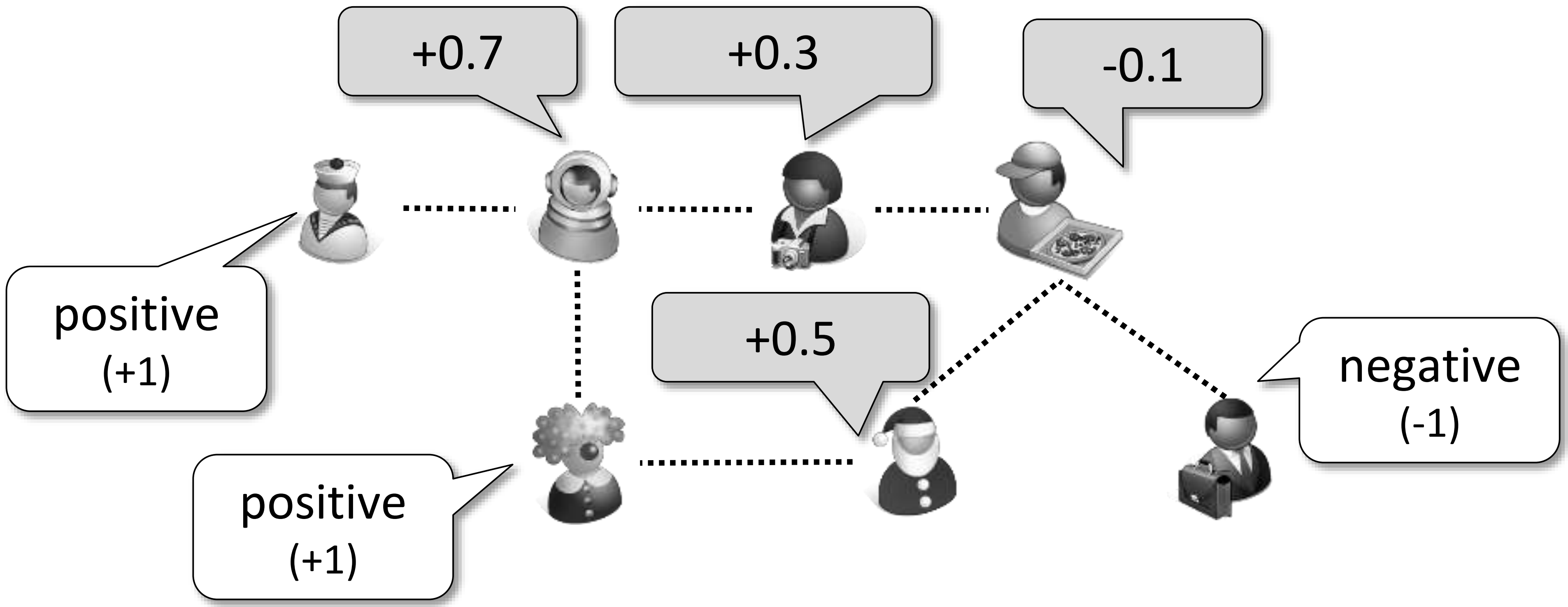
Label propagation: Simple graph-based method

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



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:

$$\text{maximize} \sum_{i \in L} \log p(y^{(i)} | \mathbf{x}^{(i)}) + \sum_{i \in U} \sum_{\hat{y}} p(\hat{y} | \mathbf{x}^{(i)}) \log p(\hat{y} | \mathbf{x}^{(i)})$$

Temporary label

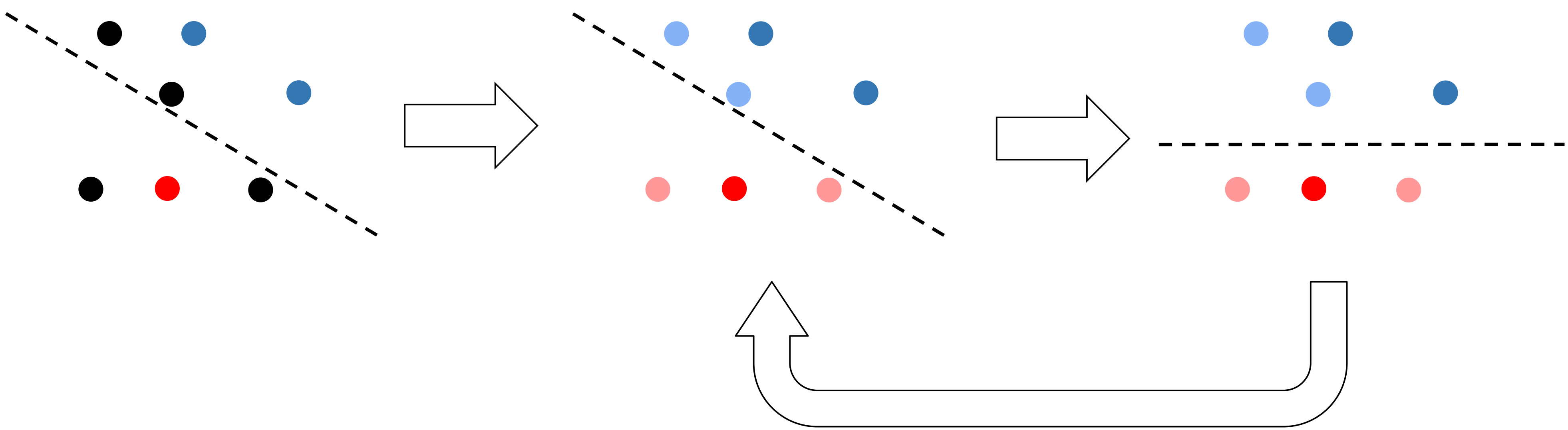
Self-training:
Believe what you believe

- Procedure:

1. Train a classifier using labeled dataset L

2. The classifier assigns temporary labels to unlabeled dataset U

3. Train a classifier using L and U



Active Learning

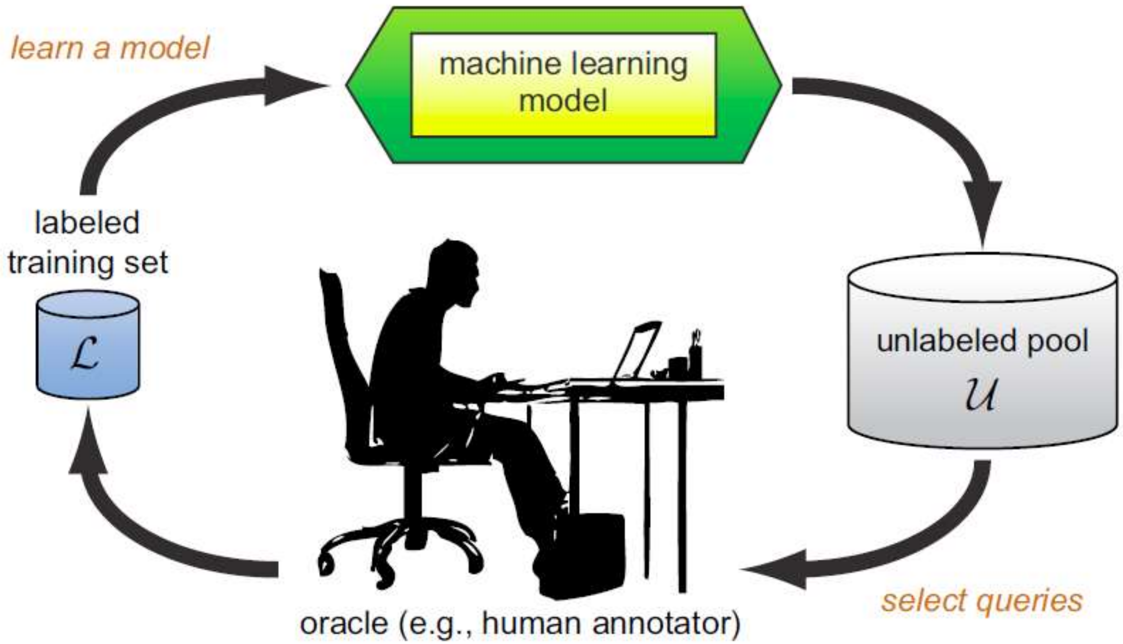


Figure 1: The pool-based active learning cycle.

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

Active learning:
Learning with a label oracle

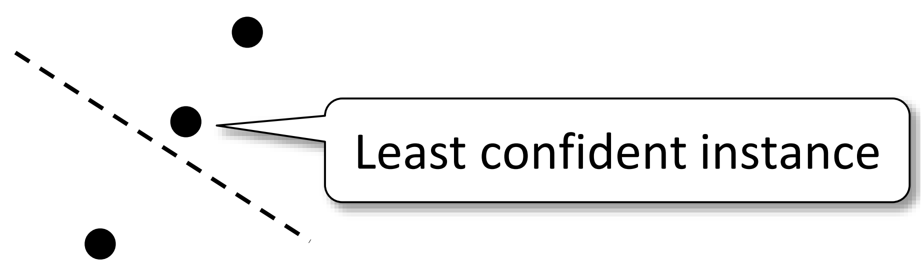
- Start with only unlabeled data $U = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$
- At each round, an active learner can query an unlabeled instance to be labeled by an oracle
  – then update the predictor using current labeled (and unlabeled) data
- An active learning algorithm determines the query 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 $f(\boldsymbol{x}) = \mathrm{sign}(\mathbf{w}^\top \mathbf{x})$, $|\mathbf{w}^\top \mathbf{x}|$ indicates "confidence level" of the prediction
  - For multi–class classification, use $\max_k \mathbf{w}^{(k)\top}\mathbf{x}$ (or, margin $\max_k \mathbf{w}^{(k)\top}\mathbf{x} - \mathrm{secondbest}_k\, \mathbf{w}^{(k)\top}\mathbf{x}$)
  - For probabilistic classifiers, the entropy $\sum_y -P(y|\mathbf{x}) \log P(y|\mathbf{x})$ is used as an uncertainty measure
- Query $\mathbf{x}^{(i)}$ with the lowest confidence/highest uncertainty



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

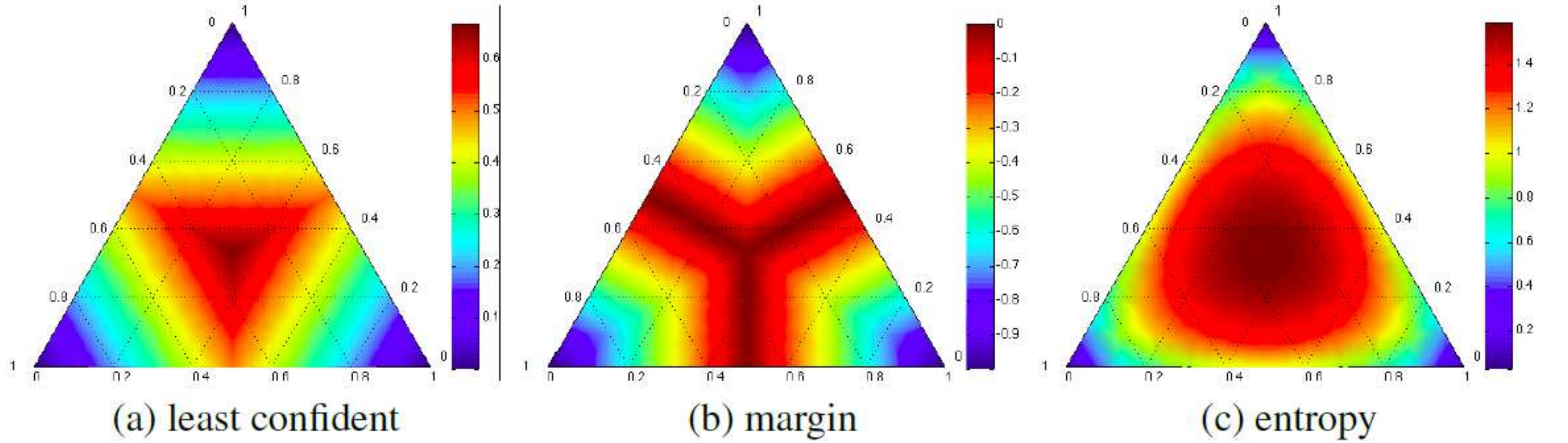


Figure 5: Heatmaps illustrating the query behavior of common uncertainty measures in a three-label classification problem. Simplex corners indicate where one label has very high probability, with the opposite edge showing the probability range for the *other* two classes when that label has very low 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

- How can we measure the impact of a labeled instance?
- We consider how much the label will change the model
- Assume gradient-based learning methods are used
  - Denote the loss function for L by $J(L)$
  - Gradient descent update $\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} - \gamma \, \boldsymbol{\nabla}_{\boldsymbol{w}} J(L \cup (\mathbf{x}, y))$ when a labeled instance $(\mathbf{x}, y)$ is newly added to L
  - The impact can be defined as $\| \boldsymbol{\nabla}_{\boldsymbol{w}} J(L \cup (\mathbf{x}, y) \|$
- Choose instance $\mathbf{x}$ that gives the largest (expected) gradient of the objective function: $\sum_y -P(y|\mathbf{x}) \| \boldsymbol{\nabla}_{\boldsymbol{w}} J(L \cup (\mathbf{x}, y) \|$

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

- Another definition of the model change
- $P_{\mathbf{w}^{\text{new}}}$: model after update with new labeled data $(\mathbf{x}, y)$
- Information gain about the unlabeled data:

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

- Choose an instance that gives the largest expected gain:

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

Transfer Learning

Transfer learning:
Training and test data come from different distributions

- Training dataset and test dataset are sampled from different distributions

- In the standard settings, an input $\mathbf{x}$ is sampled from $\mathcal{D}_{\mathcal{X}}$, and an output y is sampled from $\mathcal{D}_{\mathcal{Y}|\mathcal{X}}$ (in both training and test)

- In transfer learning,

$$\left.\begin{array}{l}\text{–Training data come from } \mathcal{D}_{\mathcal{X}}^{\text{train}} \text{ and } \mathcal{D}_{\mathcal{Y}|\mathcal{X}}^{\text{train}} \\ \text{–Test data come from } \mathcal{D}_{\mathcal{X}}^{\text{test}} \text{ and } \mathcal{D}_{\mathcal{Y}|\mathcal{X}}^{\text{test}}\end{array}\right\} \text{Different distributions}$$

- Example: Domain adaptation
  - Classification of general text documents and medical texts

Covariate shift: Input distributions are different

- Covariate shift: only the input distributions are different
  - $\mathcal{D}_{\mathcal{X}}^{\text{train}} \neq \mathcal{D}_{\mathcal{X}}^{\text{test}}$
  - $\mathcal{D}_{\mathcal{Y}|\mathcal{X}}^{\text{train}} = \mathcal{D}_{\mathcal{Y}|\mathcal{X}}^{\text{test}}$: conditional distributions are the same
  - Training dataset is labeled and test dataset is unlabeled
- Occurs when sampling of labeled data is constrained
  - 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 (e.g., 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 $p^{\text{test}}(\mathbf{x})$
- In maximum likelihood estimation, we want to maximize

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

  – Note that the expectation is taken over $p^{\text{test}}(\mathbf{x})$
- However, we can not directly evaluate the objective function
  – We do not have label information for test dataset

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

- Use the importance sampling

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

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

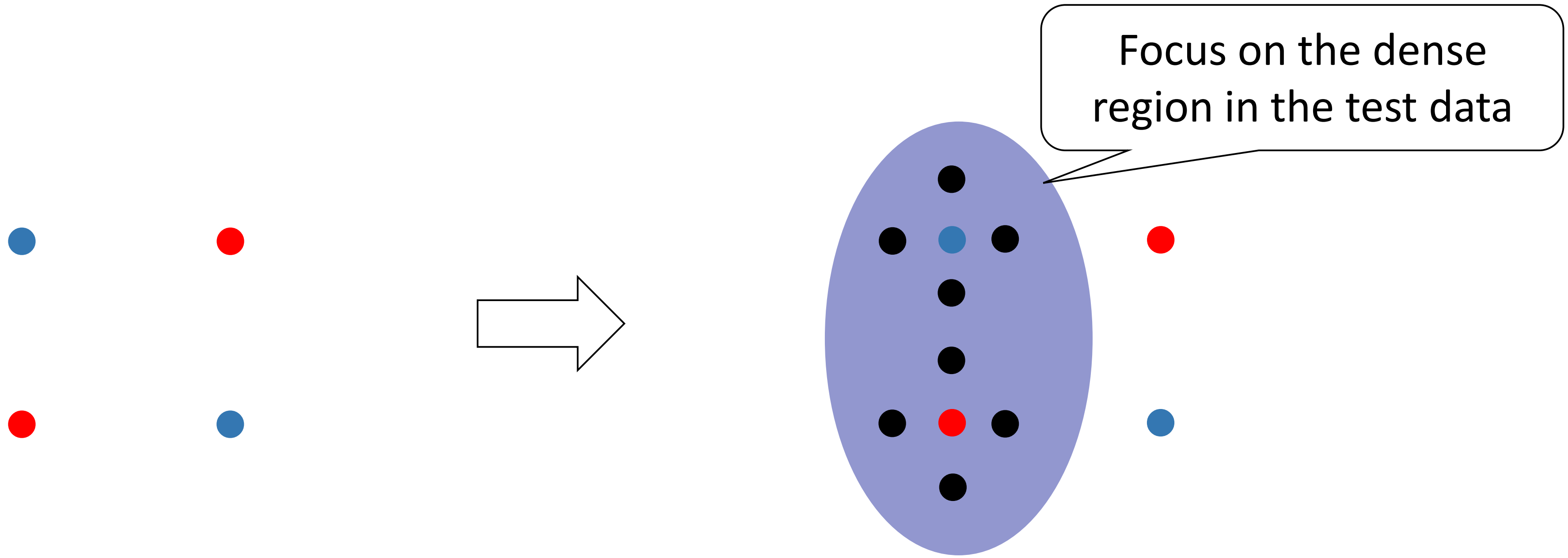
$$= \frac{1}{N} \sum_{i=1}^{N} \omega\left(\mathbf{x}^{(i)}\right) \log P\left(y^{(i)} \middle| \mathbf{x}^{(i)}\right)$$

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

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

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

- Focus on the training data in the dense region of the test data



$$\omega\left(\mathbf{x}^{(i)}\right) > 1$$

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

Transfer learning of different conditional distributions: Adaptation to model changes

- Transfer learning of different conditional distributions
  - $\mathcal{D}_{\mathcal{Y}|\mathcal{X}}^{\text{train}} \neq \mathcal{D}_{\mathcal{Y}|\mathcal{X}}^{\text{test}}$
  - $\mathcal{D}_{\mathcal{X}}^{\text{train}} = \mathcal{D}_{\mathcal{X}}^{\text{test}}$: 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. $f(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^\top \mathbf{x})$)
  - The source domain model has $\mathbf{w}^{(\mathrm{s})}$, while the target domain model has $\mathbf{w}^{(\mathrm{t})}$
- The models have shared parts and domain specific parts
  - Source domain model $\mathbf{w}^{(\mathrm{s})} = \mathbf{v}^{(0)} + \mathbf{v}^{(\mathrm{s})}$
  - Target domain model $\mathbf{w}^{(\mathrm{t})} = \mathbf{v}^{(0)} + \mathbf{v}^{(\mathrm{t})}$
- Ordinary classification methods can be used: $\tilde{f}(\mathbf{x}) = \mathrm{sign}(\widetilde{\mathbf{w}}^\top \tilde{\mathbf{x}})$
  - $\widetilde{\mathbf{w}} = (\mathbf{v}^{(0)}, \mathbf{v}^{(\mathrm{s})}, \mathbf{v}^{(\mathrm{t})})$
  - $\tilde{\mathbf{x}} = (\mathbf{x}^\top, \mathbf{x}^\top, \mathbf{0}^\top)^\top$ for source; $\tilde{\mathbf{x}} = (\mathbf{x}^\top, \mathbf{0}^\top, \mathbf{x}^\top)^\top$ for target