Crash-course in machine learning

Anders Søgaard

Center for Language Technology
University of Copenhagen
Njalsgade 140-142
DK-2300 Copenhagen S
Email: soegaard@hum.ku.dk

1. Classification
2. Semi-supervised learning
3. Sample selection bias

Running examples
Anyone?

Keywords

- observable discrete or continuous variables
- classes and clusters
- features and values
- conditional probability
- ...
- multinomial = discrete and multi-class

LOGIC: Are you sure you really know that?
STAT. METH.: C'mon! Let's shoot!

logic		**stat. meth.**	
goal	method	goal	method
proposition	propositional logic	discrete	classification
-	-	continuous	regression
structure	modal logic	structure	structured prediction

Exercise: What methods to use for ...

LOGIC: Are you sure you really know that?
STAT. METH.: C'mon! Let's shoot!

logic		**stat. meth.**	
goal	method	goal	method
proposition	propositional logic	discrete	classification
-	-	continuous	regression
structure	modal logic	structure	structured prediction

Exercise: What methods to use for ...

- predicting whether it rains now? (observations: there's thunder)

LOGIC: Are you sure you really know that?
STAT. METH.: C'mon! Let's shoot!

logic		**stat. meth.**	
goal	method	goal	method
proposition	propositional logic	discrete	classification
-	-	continuous	regression
structure	modal logic	structure	structured prediction

Exercise: What methods to use for ...

- predicting whether it rains now? (observations: there's thunder)
- predicting the amount of rain today? (observations: there's heavy rain, it started 7AM)

LOGIC: Are you sure you really know that?
STAT. METH.: C'mon! Let's shoot!

logic		**stat. meth.**	
goal	method	goal	method
proposition	propositional logic	discrete	classification
-	-	continuous	regression
structure	modal logic	structure	structured prediction

Exercise: What methods to use for ...

- predicting whether it rains now? (observations: there's thunder)
- predicting the amount of rain today? (observations: there's heavy rain, it started 7AM)
- predicting whether it rains tomorrow? (observations: it did not rain yesterday, it rains now, there's heavy wind)

	knowledge	labeled data	unlabeled data
logic	✓		
supervised		✓	
semi-supervised		✓	✓
ILP	✓	✓	✓
unsupervised			✓
*	✓		✓

*Minimally supervised, posterior regularization, generalized expectation.

Production system for POS tagging

Declarative memory and production rules (propositional logic):

- $v \vee n \vee d$ (all words are verbs, nouns or adjectives)

Conflict resolution:

(a) recency,
(b) specificity, or
(c) probability.

Note: This is similar to modern day grammar engineering.

Production system for POS tagging

Declarative memory and production rules (propositional logic):

- $v \vee n \vee d$ (all words are verbs, nouns or adjectives)
- $v \rightarrow (\neg(n \vee d))$
- $n \rightarrow (\neg(v \vee d))$
- $d \rightarrow (\neg(n \vee v))$

Conflict resolution:

(a) recency,
(b) specificity, or
(c) probability.

Note: This is similar to modern day grammar engineering.

Production system for POS tagging

Declarative memory and production rules (propositional logic):

- $v \vee n \vee d$ (all words are verbs, nouns or adjectives)
- $v \rightarrow (\neg(n \vee d))$
- $n \rightarrow (\neg(v \vee d))$
- $d \rightarrow (\neg(n \vee v))$
- $observe(can) \rightarrow label(can, v)$

Conflict resolution:

(a) recency,
(b) specificity, or
(c) probability.

Note: This is similar to modern day grammar engineering.

Production system for POS tagging

Declarative memory and production rules (propositional logic):

- $v \vee n \vee d$ (all words are verbs, nouns or adjectives)
- $v \rightarrow (\neg(n \vee d))$
- $n \rightarrow (\neg(v \vee d))$
- $d \rightarrow (\neg(n \vee v))$
- $observe(can) \rightarrow label(can, v)$
- $observe([a]\ can) \rightarrow label(can, n)$

Conflict resolution:

(a) recency,

(b) specificity, or

(c) probability.

Note: This is similar to modern day grammar engineering.

Production system for POS tagging

Declarative memory and production rules (propositional logic):

- $v \vee n \vee d$ (all words are verbs, nouns or adjectives)
- $v \rightarrow (\neg(n \vee d))$
- $n \rightarrow (\neg(v \vee d))$
- $d \rightarrow (\neg(n \vee v))$
- $\mathit{observe}(\mathit{can}) \rightarrow \mathit{label}(\mathit{can}, v)$
- $\mathit{observe}([a]\ \mathit{can}) \rightarrow \mathit{label}(\mathit{can}, n)$
- $\mathit{observe}(\mathit{can}) \rightarrow \mathit{label}(\mathit{can}, n)$

Conflict resolution:

(a) recency,
(b) specificity, or
(c) probability.

Note: This is similar to modern day grammar engineering.

Statistical methods

We introduce three cardinal statistical methods:

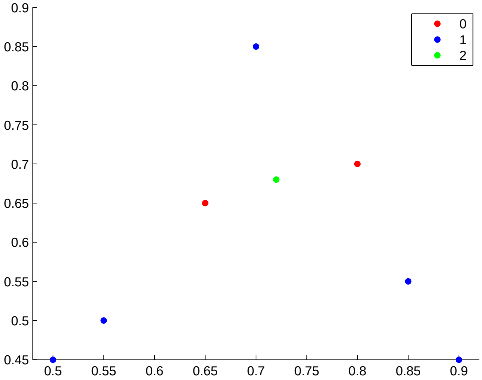
a. the nearest neighbor rule

b. Bayesian models

c. perceptron

- Intuition: If it walks like a duck and quacks like a duck, it is probably a duck.
- Formalized: duck-walking:+/-, quack:+/-, class:duck/not-duck.
- Example (Central Valley Naturalists): ”at first thought it might be a kind of goose, but it quacked”

Nearest neighbor



- Prediction: class of nearest neighbor.
- k-nearest neighbor: plurality vote of k nearest neighbors.
- See this demo:

`www.cs.cmu.edu/ zhuxi/courseproject/knndemo/KNN.html`

Nearest neighbor issues

- k increases robustness (but decreases expressivity; $k = N$ is equivalent to Rocchio)
- $$P(y|\mathbf{x}) = \frac{|\{\langle y', \mathbf{x}' \rangle \in T_k \mid y' = y\}|}{k}$$
  with T_k the k nearest neighbors
- If the optimal classifier has an error rate of ϵ, a nearest neighbor classifier has an error rate of at most 2ϵ as the amount of training data increases.

Major drawback:

- Nearest neighbor classifiers are extremely slow at test time on NLP-type problems.

Condensed nearest neighbor

$T = \{\langle \mathbf{x}_1, y_1 \rangle, \ldots, \langle \mathbf{x}_n, y_n \rangle\}$, $C = \emptyset$
for $\langle \mathbf{x}_i, y_i \rangle \in T$ **do**
  if $C(\mathbf{x}_i) \neq y_i$ **then**
    $C = C \cup \{\langle \mathbf{x}_i, y_i \rangle\}$
  end if
end for
return C

Bayesian models

- Bayesian models are widely used in cognitive science (as cognitive models) and machine learning (for prediction).
- In cognitive science, they link back to Stanford-style conceptual organization theories in the 1970s.

Bayes' rule

$$P(h|d) = \frac{P(d|h)P(h)}{P(d)}$$

follows from the fact that the joint probability $P(a, b) = P(a|b)P(b) = P(b|a)P(a)$.

Chain rule is the extension of $P(a, b) = P(a|b)P(b)$ to n variables:

$$P(a_1, \ldots, a_n) = P(a_1|a_2, \ldots, a_n) \ldots P(a_{n-1}|a_n)P(a_n)$$

Bayesian classifiers

It is typically not possible to compute:

$$P(a_1, \ldots, a_n) = P(a_1|a_2, \ldots, a_n) \ldots P(a_{n-1}|a_n)P(a_n)$$

however, if all observed variables are assumed to be only dependent on class, we get:

$$P(y, \mathbf{x}) = P(y) \prod_{x \in \mathbf{x}} P(x|y)$$

i.e. the product of the *likelihoods* of all features and the *prior probability* of class. In a dependency graph or **Bayesian network** this can be generalized to:

$$P(y, \mathbf{x}) = P(y) \prod_{x \in \mathbf{x}} P(x|\text{parents}(x))$$

The classifier in which all variables are only dependent on class is called the **naive Bayes** classifier.

Naive Bayes

- Naive Bayes is widely used in spam filtering, image classification, and bioinformatics.
- Bayesian classifiers are generative (model joint probability), and naive Bayes is linear.
- Training and testing are both linear in the size of data.

Major drawbacks:

- expressivity
- what happens when tornado = + never occurs in training data, but in test data?

Perceptron

- A perceptron consists of a weight vector $\mathbf{w}$ with a weight for each feature, a bias term and a learning rate α.
- $c(\mathbf{x}) = 1$ iff $\mathbf{w} \cdot \mathbf{x} + b > 0$, else 0

Perceptron learning:
For each datapoint $\langle y_j, \mathbf{x}_j \rangle$ with $|\mathbf{x}_j| = n$:

- $\forall 0 \leq i \leq n . w_i(t+1) = w_i(t) + \alpha(y_j - c(\mathbf{x}_j, t))\mathbf{x}_{j,i}$

There are applets demoing the perceptron here:

`http://intsys.mgt.qub.ac.uk/notes/perceptr.html`

`http://lcn.epfl.ch/tutorial/english/perceptron/html/index.html`

Example

y	x_1	x_2
0	1	0
0	0	0
1	1	1
0	0	1

Say $\alpha = .1$ and $b = 0$. The weight vector is initialized as $\langle 0, 0 \rangle$.

1. For the first data point $\mathbf{w} \cdot \mathbf{x} + b = 0$, which means that weights will remain the same.
2. Same goes for the second data point.
3. The third data point is positive, so there is an update such that the weight vector is now $\langle .1, .1 \rangle$.
4. ...

The input for b is set to -1 to uniformly update the bias.

Summary

	efficiency	**expressivity**	**stability**	SOA
NN		✓	✓	
NB	✓		(✓)	Bayesian networks
Perc	✓		(✓)	av. perc., SVMs

*Smoothed naive Bayes is relatively stable. The stability of perceptron depends on the underlying distribution.

Classification (notation)

From a sample

$$\mathcal{X} = \{x^t, r^t\}_{t=1}^N$$

we learn a model

$$g(x|\theta)$$

where θ picks out a hypothesis in the hypothesis class defined by $g()$. Our model's approximation error

$$E(\theta|\mathcal{X}) = \Sigma_t L(r^t, g(x^t|\theta))$$

is the sum of our losses. Supervised learning is thus about finding the θ^* that minimizes approximation error, e.g.:

$$\theta^* = \arg\min_\theta E(\theta|\mathcal{X})$$

Semi-supervised learning

- **self-training** and EM
- **co-training** (multi-view methods)
- graph-based methods
- feature-based methods

Human learning is semi-supervised (Zhu et al., 2007)

Experiment:

- 22 subjects.
- 1 labeled example for each of two classes (training). 21 test examples forming a continuum between the classes.
- 230 unlabeled examples sampled from two Gaussians around the two training examples.
- For 12 subjects, the Gaussians are shifted left. For 10 subjects, the Gaussians are shifted right.

Results:

- Unlabeled examples alter the decision boundary. Unlabeled examples help.
- Predictions are similar to that of EM.

Self-training

procedure `selfTrain` (L_0, U)

1 $c \leftarrow$ `train`(L_0)

2 **loop until** stopping criterion is met

3. $L \leftarrow L_0/L +$ `select(label(`U, c`))`

4. $c \leftarrow$ `train`(L)

5. **end loop**

6. **return** c

If L and not L_0 in line 3, this is called *indelibility*.

Parameters and variants

- Base learner: Naive Bayes, k nearest neighbor, classification tree, etc.,
- confidence measure: $P(h_1|d)$ or $\frac{P(h_1|d)}{P(h_2|d)}$,
- stopping criterion: fixed, convergence, cross-validation,
- seed: labeled data, initial classifier, dictionary, production system, etc.,
- throttling: instead of accepting all confident instances, only the k most confident instances are accepted,
- balancing: same number of instances of each class, and
- using a pool (preselection).

Co-training

procedure `coTrain` (L, U)
  loop until stopping criterion is met
    $c_1 \leftarrow \texttt{train}(\texttt{view}_1(L))$
    $c_2 \leftarrow \texttt{train}(\texttt{view}_2(L))$
    $L \leftarrow L + \texttt{select(label(}U, c_1\texttt{))} + \texttt{select(label(}U, c_2\texttt{))}$
  end loop
  $c \leftarrow \texttt{train}(L)$
  return c

Robust semi-supervised methods in WSD

Søgaard, A.; Johannsen, A. 2010. Robust semi-supervised and ensemble-based methods for word sense disambiguation. Int. Conf. on NLP. Reykjavik, Iceland.

learner	baseline	self-training	tri-training	Δ
LogitBoost	65.56	66.39	**66.43**	0.87
naive Bayes	**64.33**	64.09	62.74	-1.59
PART	60.37	60.57	**60.84**	0.47
DecisionStump	58.23	58.59	58.86	0.63

Semi-supervised condensed nearest neighbor

Søgaard, A. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. ACL. Portland, Oregon.

Condensed nearest neighbor:

$T = \{\langle \mathbf{x}_1, y_1\rangle, \ldots, \langle \mathbf{x}_n, y_n\rangle\}$, $C = \emptyset$
for $\langle \mathbf{x}_i, y_i\rangle \in T$ **do**
  if $C(\mathbf{x}_i) \neq y_i$ **then**
    $C = C \cup \{\langle \mathbf{x}_i, y_i\rangle\}$
  end if
end for
return C

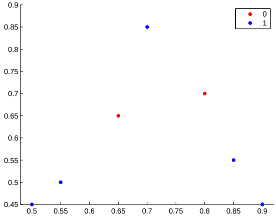
Semi-supervised condensed nearest neighbor

Generalized condensed nearest neighbor:

$T = \{\langle \mathbf{x}_1, y_1\rangle, \ldots, \langle \mathbf{x}_n, y_n\rangle\}$, $C = \emptyset$
for $\langle \mathbf{x}_i, y_i\rangle \in T$ **do**
  if $C(\mathbf{x}_i) \neq y_i$ or $P_C(\langle \mathbf{x}_i, y_i\rangle | \mathbf{x}_i) < 0.55$ **then**
    $C = C \cup \{\langle \mathbf{x}_i, y_i\rangle\}$
  end if
end for
return C

Semi-supervised condensed nearest neighbor

a)



b)

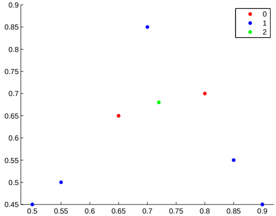
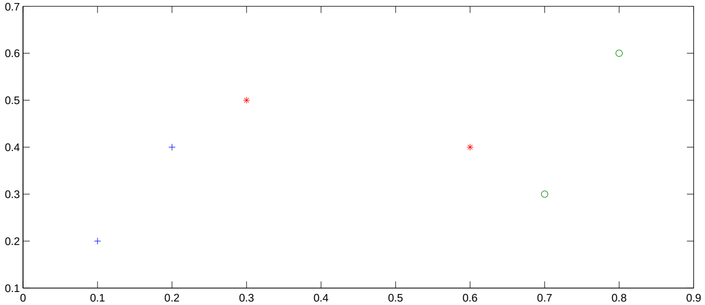
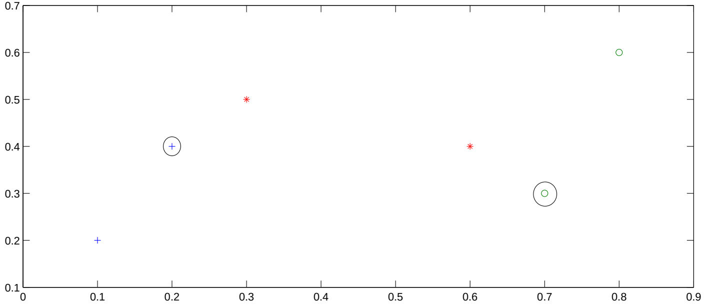


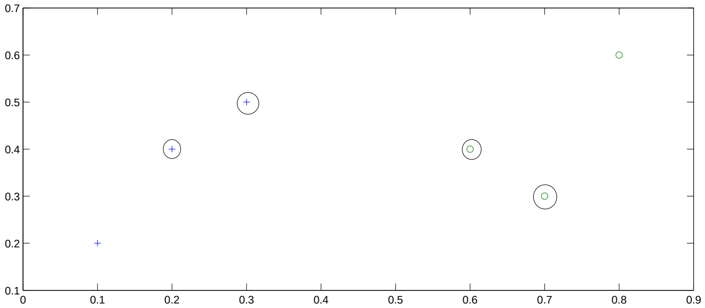
Figure: Unlabeled data may help find better representatives in condensed training sets.

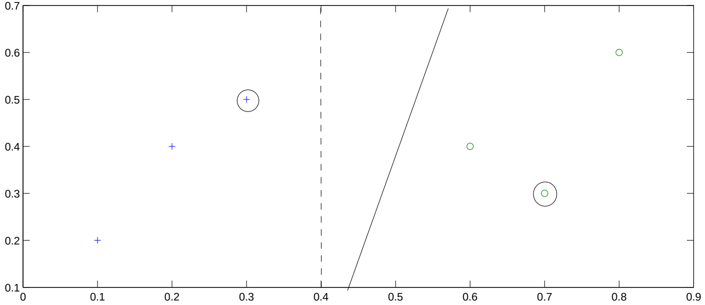
Semi-supervised condensed nearest neighbor

$T = \{\langle \mathbf{x}_1, y_1\rangle, \ldots, \langle \mathbf{x}_n, y_n\rangle\}$, $C = \emptyset$, $C' = \emptyset$
$U = \{\langle \mathbf{w}_1, z_1\rangle, \ldots, \langle \mathbf{w}_m, z_m\rangle\}$
for $\langle \mathbf{x}_i, y_i\rangle \in T$ **do**
    if $C(\mathbf{x}_i) \neq y_i$ or $P_C(\langle \mathbf{x}_i, y_i\rangle|\mathbf{x}_i) < 0.55$ **then**
        $C = C \cup \{\langle \mathbf{x}_i, y_i\rangle\}$
    end if
end for
for $\langle \mathbf{w}_i, z_i\rangle \in U$ **do**
    if $P_T(\langle \mathbf{w}_i, z_i\rangle|\mathbf{w}_i) > 0.90$ **then**
        $C = C \cup \{\langle \mathbf{x}_i, T(\mathbf{x}_i)\rangle\}$
    end if
end for
for $\langle \mathbf{x}_i, y_i\rangle \in C$ **do**
    if $C'(\mathbf{x}_i) \neq y_i$ **then**
        $C' = C' \cup \{\langle \mathbf{x}_i, y_i\rangle\}$
    end if
end for
return C'


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Classification

1/10	cnn		scnn		
	acc	dps	acc	dps	err.red
anneal	81.11	31	**83.33**	40	11.75%
balance-scale	69.84	28	**82.54**	49	42.11%
brown-selected	78.95	5	**89.47**	7	49.98%
bupa	57.14	23	**62.86**	17	13.35%
car	78.61	45	**87.28**	81	40.53%
crx	81.16	15	**84.06**	34	15.39%
ionosphere	**88.89**	11	72.22	13	-
lung	52.38	8	**90.48**	10	80.01%
monks-2	80.33	29	**81.97**	38	8.34%
mushroom	77.61	284	**82.17**	318	20.37%
primary-tumor	52.94	23	**58.82**	25	12.49%
shuttle-landing-control	80.77	8	**96.15**	12	79.98%
tic-tac-toe	34.38	3	**76.04**	24	63.49%
titanic	70.59	18	**76.92**	21	21.52%
wdbc	98.25	10	98.25	13	0
yeast	65.10	78	**69.13**	90	11.55%

Classification

1/20	cnn		scnn		
	acc	dps	acc	dps	err.red
anneal	66.67	11	**76.19**	23	28.56%
balance-scale	75.56	20	**81.11**	32	22.71%
brown-selected	78.95	5	78.95	5	0
bupa	48.57	11	**54.29**	10	11.12%
car	73.99	27	**77.46**	55	13.34%
crx	72.46	7	**88.41**	22	57.92%
ionosphere	69.44	9	**72.22**	11	9.10%
lung	52.34	5	**71.43**	6	40.05%
monks-2	68.85	18	68.85	23	0
mushroom	**73.31**	160	72.20	184	-
primary-tumor	47.06	12	47.06	12	0
shuttle-landing-control	88.46	8	88.46	8	0
tic-tac-toe	34.38	3	**68.75**	24	52.38%
titanic	82.35	16	82.35	16	0
wdbc	94.74	4	**96.49**	9	36.53%
yeast	58.39	52	**60.40**	57	4.83%

POS tagging

Features:

JJ	JJ	17*
NNS	NNS	1
IN	IN	428
DT	DT	425

POS tagging

Labeled: WSJ. Unlabeled: Brown.

	acc (%)	data points	err.red
cnn	95.79	3811*	
svmtool	97.15	-	
†S09	97.44	-	
scnn	**97.50**	2249*	40.6%

$*$: $3811/46451 \sim 8\%$. $2249/1217262 \sim 0.2\%$.

† : S09=Spoustova et al. (2009) (best published).

Err.red. relative to SVMTool is >12%; >2% relative to S09.

Conclusions

- Semi-supervised condensed nearest neighbor is a **robust** semi-supervised learning algorithm,
- and it does **better condensation** than supervised condensed nearest neighbor.

The code is available at:

`http://cst.dk/anders/scnn/`

Sample selection bias

The default assumption in machine learning is that training and test data are independently and identically (iid) drawn from the same distribution. Otherwise we talk of **sample selection bias**, transfer learning or in some cases domain adaptation. Sample selection biases can be biases in:

$P(y)$ (class imbalance)	WSD, SMT, parsing, NER
$P(\mathbf{x})$ (covariate shift)	SMT, parsing, NER
$P(y \mid \mathbf{x})$	parsing ('a can can melt down'), NER

In machine translation, for example, which can be seen as a structured learning problem of predicting target sentence y given a source sentence x, we typically see a bias in $P(y)$ and $P(\mathbf{x})$, but not in $P(y|\mathbf{x})$.

Sample selection bias

- instance weighting
- feature-based methods
- semi-supervised methods

Instance weighting

- In class imbalance, each data point $\langle y, \mathbf{x} \rangle$ should be weighted by $\frac{P_t(y)}{P_s(y)}$ where P_t is the target distribution, and P_s the source distribution.
- In covariate shift, each data point $\langle y, \mathbf{x} \rangle$ should be weighted by $\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})}$.

Example: co-variate shift In supervised learning with N labeled data points, we minimize the empirical risk to find a good model $\hat{\theta}$ for a loss function $l : \mathcal{X} \times \mathcal{Y} \times \Theta$:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{\langle \mathbf{x}, y \rangle \in \mathcal{X} \times \mathcal{Y}} \hat{P}(\langle \mathbf{x}, y \rangle) l(\mathbf{x}, y, \theta)$$

$$= \arg\min_{\theta \in \Theta} \sum_{i=1}^{N} l(\mathbf{x}_i, y_i, \theta)$$

In transfer learning, we can rewrite this as:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{\langle \mathbf{x}, y \rangle \in \mathcal{X} \times \mathcal{Y}} \frac{P_t(\langle \mathbf{x}, y \rangle)}{P_s(\langle \mathbf{x}, y \rangle)} \hat{P}_s(\langle \mathbf{x}, y \rangle) l(\mathbf{x}, y, \theta)$$

$$= \arg\min_{\theta \in \Theta} \sum_{i=1}^{N^s} \frac{P_t(\langle \mathbf{x}_i^s, y_i^s \rangle)}{P_s(\langle \mathbf{x}_i^s, y_i^s \rangle)} l(\mathbf{x}_i^s, y_i^s, \theta)$$

Under the covariate shift assumption $\frac{P_t(\langle \mathbf{x}, y \rangle)}{P_s(\langle \mathbf{x}, y \rangle)}$ for a pair $\langle \mathbf{x}, y \rangle$ can be replaced with $\frac{P_t(\mathbf{x})}{P_s(\mathbf{x})}$.

- Instance weighting **by distance**, weighting out-of-domain data points by their distance to in-domain data, e.g. input data.
- Instance weighting **by classification**, training a probabilistic classifier to distinguish between out-of-domain (training) and in-domain (test) data and for each out-of-domain data point use the probability that it belongs to the target domain as weight (Zadrovny, 2004).

Structural correspondence learning

- Select pivot features, i.e. common in source and target data, predictive in source data.
- Train a classifier to predict the occurrence pivot feature. Features that are predictive of pivot features are aligned with them.
- The target domain classifier is trained on pivot features and aligned features.

Sample selection bias for semantics

- Sample selection bias in WSD is typically thought of as class imbalance.
- Both instance weighting and SCL have been used for semantic parsing.
- Søgaard and Haulrich (2011) show how instance weighting can be incorporated in state-of-the-art dependency parsing.