Semi-supervised learning in natural language processing

Anders Søgaard

Course assumptions

"Much to learn you still have!"

Course outline

1. **Supervised learning**
2. Unsupervised learning and semi-supervised learning
3. Learning from weighted data and transfer learning
4. Applications to dependency parsing
5. Transfer learning in the blind

Materials:	http://esslli2012.pl
Personal website	http://cst.dk/anders/
Email:	soegaard@hum.ku.dk

Language as data points

In NLP, we represent language (documents, sentences, words) by arrays of numbers, most often 0s and 1s:

$$\langle 0, 1, 0, 0, 1 \rangle$$

could for example represent the text:

McCain just gave a cheap plug to Ed Kennedy.

The array may be a series of values for the attributes ⟨Obama, McCain, Malcolm X, Mary Poppins, Ed Kennedy⟩, where 1 means that a feature (word) is present in the text, and 0 means it isn't. It is often convenient to store data points as sparse matrices representing only non-zero values:

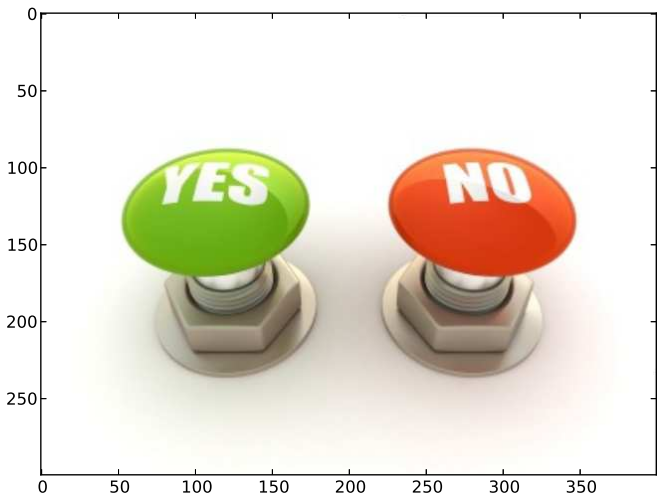
$$\langle 1 : 1, 4 : 1 \rangle$$

Labeled datapoints

Say we are interested in subsets (or classes) of documents, sentences, or words, e.g. positive user reviews, spam emails, sentences with relative clauses, or metaphors. The class of each datapoint is encoded by its associated class label.

$$\langle 1, \langle 0, 1, 0, 0, 1 \rangle \rangle \text{ or, in sparse format: } \langle 1, \langle 1:1, 4:1 \rangle \rangle$$

We write $\mathbf{x}$ for data points and y for class labels. For now class labels are assumed to be binary, i.e. $y \in \{0, 1\}$, but the observed variables can be both discrete (with values 0 and 1) and continuous (real-valued).



Visualization

- We can visualize data points with n observed variables in n-dimensional space.
- Say we represent texts by the relative frequency of the words *McCain* and *Obama*. The above text would then be represented by the array $\langle .2, 0 \rangle$.
- The figure below is a Pylab plot of 111 positive and 111 negative movie user reviews from the Imdb dataset (Maas et al., 2011) by the relative frequencies of the words 'love' (horizontal axis) and 'boring' (vertical axis). Note that the diagonal $f(x) = x$ discriminates positives from negatives fairly well.

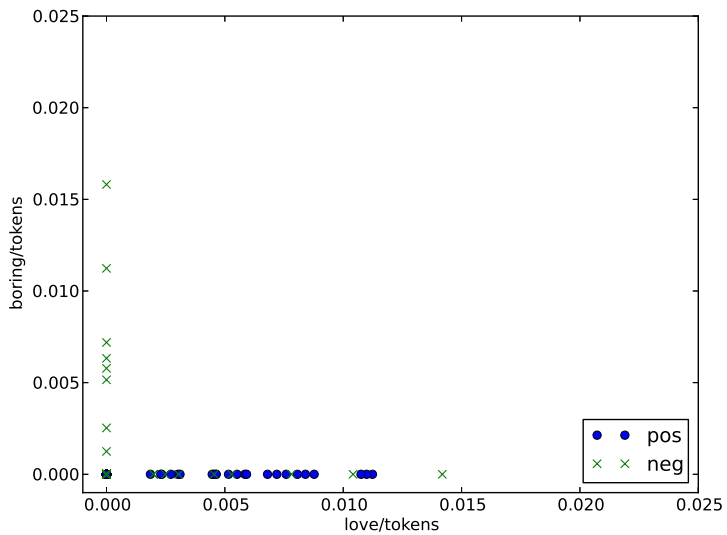


Figure: Imdb user reviews by keyword frequencies (toy example)

Linguistic structures

While classification and clustering algorithms find massive applications in NLP, classification or clustering is typically not the end goal in NLP. Here we are typically interested in *linguistic structures*.

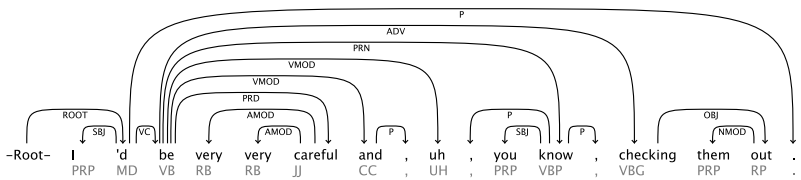


Figure: Syntactic analysis (Switchboard Corpus)

The answer is to combine classification or clustering algorithms with search or parsing algorithms, but initially we restrict ourselves to (text) classification problems:

20 Newsgroups

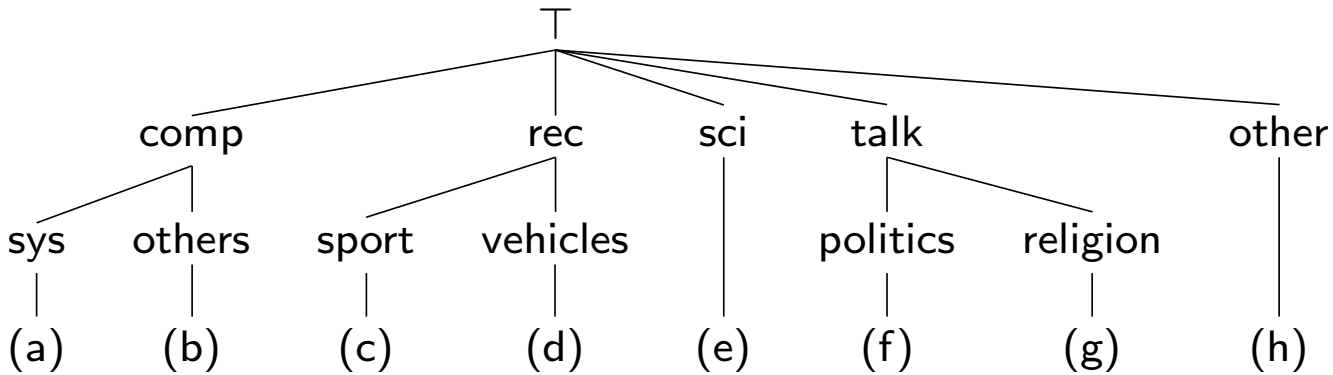


Figure: Hierarchical structure of 20 Newsgroups. (a) IBM, Mac, (b) Graphics, MS-Windows, X-Windows, (c) Baseball, Hockey, (d) Autos, Motorcycles, (e) Cryptography, Electronics, Medicine, Space, (f) Guns, Mideast, Miscellaneous, (g) Atheism, Christianity, Miscellaneous, (h) Forsale

Classification:

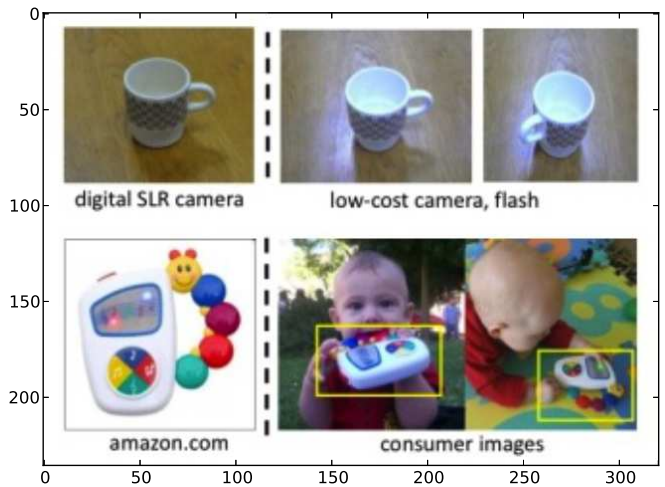
$$\arg\max_{y \in \mathcal{Y}} P(y|\mathbf{x})$$

Note: Instead of 20-way classification we will use 20 Newsgroups to perform domain adaptation experiments.

Motivation: Data is scarce and biased

This course introduces *semi-supervised learning* and *transfer learning* algorithms because:

1. Labeled data is scarce.
2. Labeled data is biased.



HIT-Baseline	LAS	POS
Wall Street Journal	91.88	97.76
Yahoo Answers!	80.75	90.99
BBC Newsgroups	85.26	92.32
Amazon (reviews)	81.60	90.46

Exercise: What could data bias be in your field?
Note: *inductive* bias vs. *data* bias.

Standard assumptions in supervised learning

- Smoothness assumption
- Independently and **identically distributed**
- Single cluster assumption
- Low-density separation

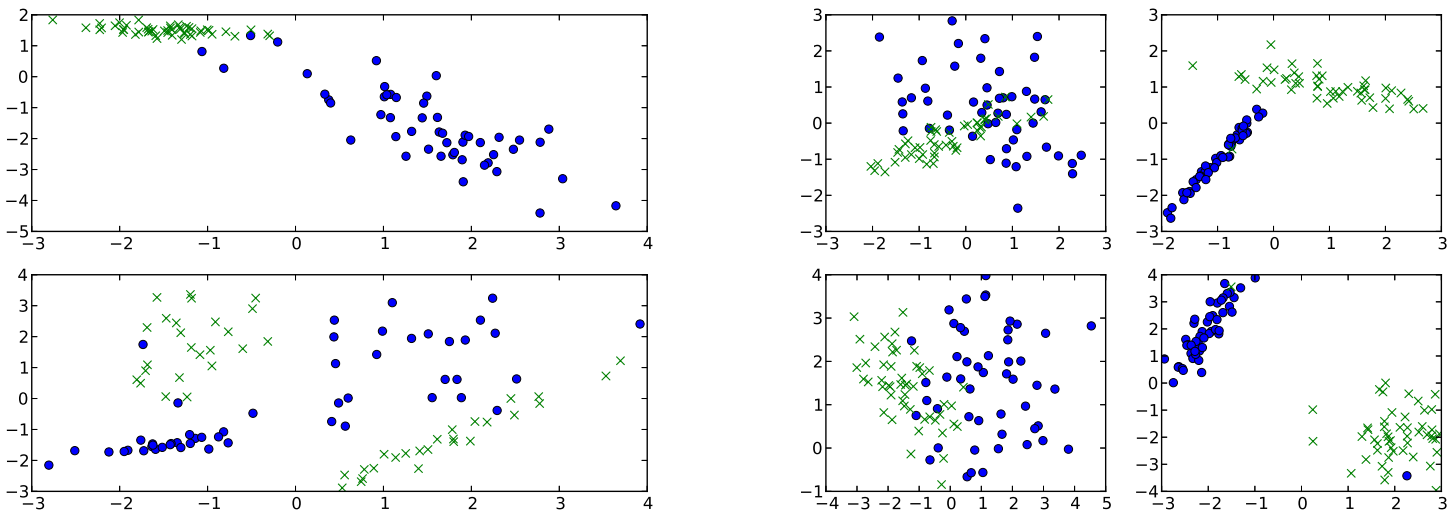


Figure: Binary classification problems where each class consists of one or two clusters (left) and binary classification problems with varying degrees of separability (right)

Standard assumptions in supervised learning

- Smoothness assumption
- Independently and **identically distributed**
- Single cluster assumption
- Low-density separation

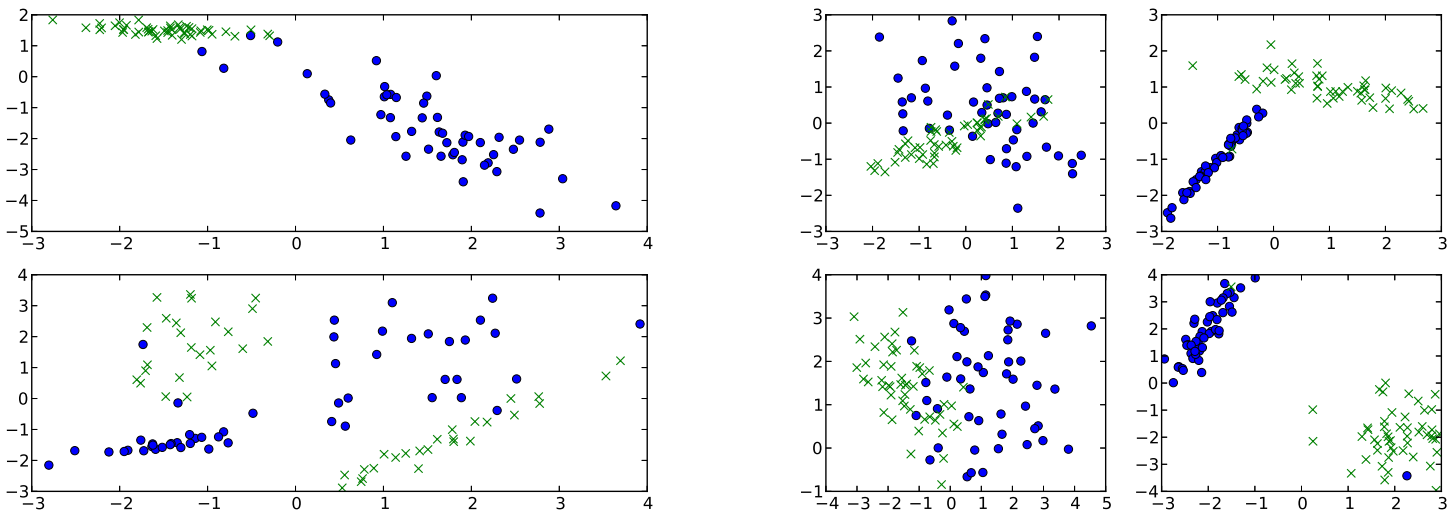


Figure: Binary classification problems where each class consists of one or two clusters (left) and binary classification problems with varying degrees of separability (right)

How to check whether the assumptions hold?

Identically distributed:

- *KL*-divergence: cross-entropy $(-\sum_{\mathbf{x}} P(\mathbf{x}) \log Q(\mathbf{x}))$ minus entropy $(-\sum_{\mathbf{x}} P(\mathbf{x}) \log P(\mathbf{x}))$:

$$\sum_{\mathbf{x}} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})}$$

- Jensen-Shannon divergence

$$D_{\mathrm{JS}}(P, Q) = \frac{1}{2} D_{\mathrm{KL}}(P, M) + \frac{1}{2} D_{\mathrm{KL}(Q,M)}$$

Coherence and separability:

- within-class scatter

$$\sum_{c} \sum_{i \in c} (\mathbf{x}_i - m_c)(\mathbf{x}_i - m_c)^T$$

- between-class scatter

$$\sum_{c} (m_c - \bar{\mathbf{x}}))(m_c - \bar{\mathbf{x}}))^T$$

KL-divergence in `numpy, scipy`

Approximate *KL*-divergence estimation using nearest neighbors (Pérez-Cruz, 2008).

```
import numpy as np
from scipy.spatial import cKDTree as KDTree
def KLDiv(X1,X2):
```

- n,d=X1.shape
- m,dy=X2.shape
- xtree=KDTree(X1)
- ytree=KDTree(X2)
- r=xtree.query(X1,k=2,eps=.01,p=2)[0][:,1]
- s=ytree.query(X1,k=1,eps=.01,p=2)[0]
- diff=r/s
- return -np.log(diff).sum() * d / n + np.log(m/(n-1))

Nearest neighbor

The nearest neighbor classifier, in its simplest form, just says that we should label an unseen data point by the label of its nearest neighbor in the previously seen data, i.e. our labeled data. **Example:**

y	zebra	viagra	venus
spam	0	1	0
non-spam	1	0	0
non-spam	0	0	1
?	0	1	1

Figure: Toy example

The Hamming distance of the unlabeled data point is 2 to the first data point, 3 to the second, and 1 to the third. The nearest neighbor is therefore the third of our labeled data points. Consequently, we predict the new email to be non-spam.

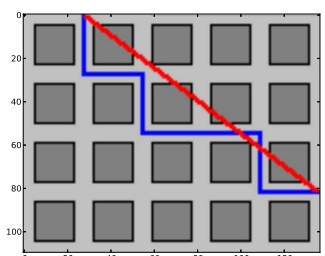


Figure: Manhattan vs. Euclidean distance

Nearest neighbor

Characteristics:

- Non-parametric.
- Very expressive, induces a Voronoi tessellation. Bias ↓.
- . . . *but:* variance ↑.

One way to decrease variance or instability is to increase the number of nearest neighbors used to determine the class of unseen data points. This leads us to k-nearest neighbor classification. In a k-nearest neighbor classification we simply let the k nearest neighbors of the unseen data point vote on its class.

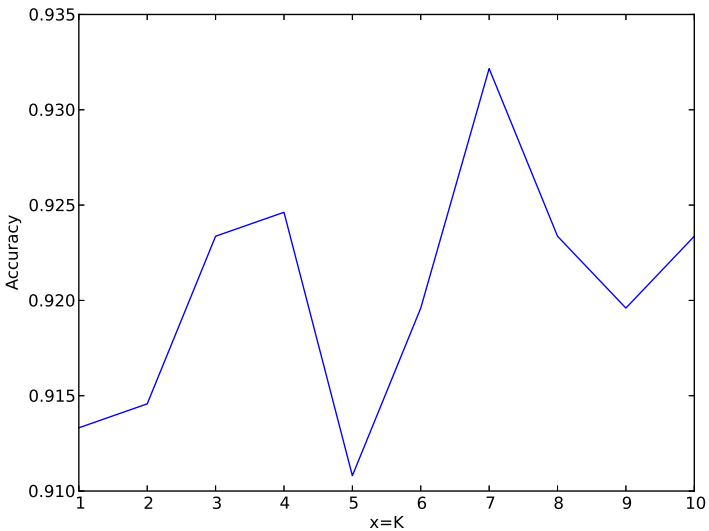


Figure: Performance of k-nearest neighbor with varying k on HOCKEY-BASEBALL

Nearest neighbor in `scikits`

- from sklearn.neighbors import KNeighborsClassifier as knn
- clf=knn(n_neighbors=1)
- clf.fit(X_train,y_train)
- print clf.score(X_test,y_test)

Note: `scikits` implements efficient search strategies, but in text classification *sparse* encoding is more important.
Note: that now we can estimate probabilities by taking the number of votes k_i on a particular class y_i over k:

$$\hat{P}(y_i|\mathbf{x}) = \frac{k_i}{k} \quad (1)$$

Naive Bayes

Assumptions:

- Chain Rule: $P(A, B) = P(A)P(B|A) = P(B)P(A|B)$
- If A and B independent, $P(A|B) = P(A)P(B)$

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(\mathbf{x})} \text{ (by Chain Rule)} \quad (2)$$

If independent:

$$P(y|\mathbf{x}) = \frac{P(y)\prod_i P(x_i|y)}{P(\mathbf{x})} \text{ (by independence of } A \text{ and } B) \quad (3)$$

Naive Bayes

y	zebra	viagra	venus
spam	0	1	0
non-spam	1	0	0
non-spam	0	0	1
?	0	1	1

Figure: Toy example

$$
\begin{aligned}
P(\text{spam}) & \quad 1/3 \\
P(\text{zebra} = 0 \mid \text{spam}) & \quad 1 \\
P(\text{viagra} = 1 \mid \text{spam}) & \quad 1 \\
P(\text{venus} = 1 \mid \text{spam}) & \quad 0 \\
\vdots & \\
P(\text{spam}) \prod_i P(x_i|\text{spam}) & \quad 0 \\
P(\text{non-spam}) \prod_i P(x_i|\text{non-spam}) & \quad 0
\end{aligned}
$$

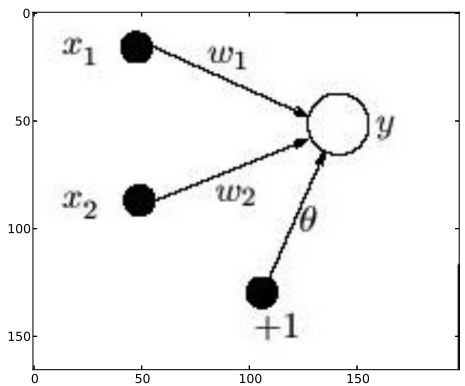
Laplace smoothing assumes every feature-value pair has been observed at least once. It now holds that $P(\text{spam}) \prod_i P(x_i|\text{spam}) \sim 6\%$ and $P(\text{non-spam}) \prod_i P(x_i|\text{non-spam}) \sim 4\%$.

Naive Bayes in scikits

- from sklearn.naive_bayes import BernoulliNB as nb
- clf=nb()
- clf.fit(X_train,y_train)
- print clf.score(X_test,y_test)

Perceptron

A perceptron c consists of a weight vector $\mathbf{w}$ with a weight for each feature, a bias term b and a learning rate α. For a data point $\mathbf{x}_j$, $c(\mathbf{x}_j) = 1$ iff $\mathbf{w} \cdot \mathbf{x} + b > 0$, else 0. The threshold for classifying something as positive is thus $-b$. The bias term is left out by adding an extra variable to our data with fixed value -1. The perceptron learning algorithm now works by maintaining $\mathbf{w}$ in several passes over the data:



$$\mathbf{w}^{i+1} \leftarrow \mathbf{w}^i + \alpha(y_j - \text{sign}(\mathbf{w}^i \cdot \mathbf{x}_j))\mathbf{x}_j \quad (4)$$

Perceptron

dataset $X = \{\langle y_i, \mathbf{x}_i \rangle\}_{i=1}^{N}$
$\mathbf{w}^0 = 0, i = 0$
for $k \in K$ **do**
  for $n \in N$ **do**
    $\mathbf{w}^{i+1} \leftarrow \mathbf{w}^i + \alpha(y_n - \text{sign}(\mathbf{w}^i \cdot \mathbf{x}_n))\mathbf{x}_n$
    $i \leftarrow i + 1$
  end for
end for
return $\mathbf{w}^{K \times N}$

Figure: Perceptron

Perceptron

y	zebra	viagra	venus
spam	0	1	0
non-spam	1	0	0
non-spam	0	0	1
?	0	1	1

Figure: Toy example

We initialize the model as $w = \langle 0, 0, 0, 0 \rangle$ where the 4th weight is the bias term which is encoded as a node always receiving the signal -1. Say $\alpha = 0.1$. The prediction for the first data point is that it belongs to the negative class, since $0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + -1 \cdot 0 = 0$. This is wrong so we update the weights, leading to:

$$w = \langle 0+0.1(1-0)0, 0+0.1(1-0)1, 0+0.1(1-0)0, 0+0.1(1-0)-1 \rangle = \langle 0, 0.1, 0, -0.1 \rangle$$

The next two datapoints are correctly classified as negative datapoints, so no additional updates are made in the first pass over the data.

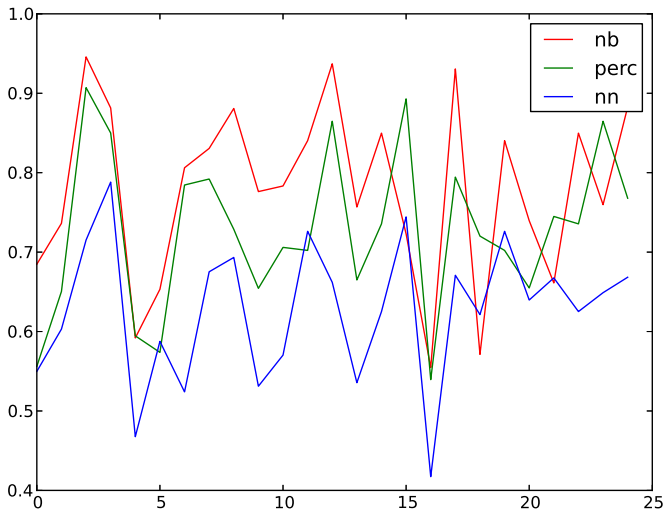
Perceptron in scikits

- from sklearn.linear_models import Perceptron as perc
- clf=perc()
- clf.fit(X_train,y_train)
- print clf.score(X_test,y_test)

Comparison

Evaluation over 25 randomly selected cross-domain datasets:

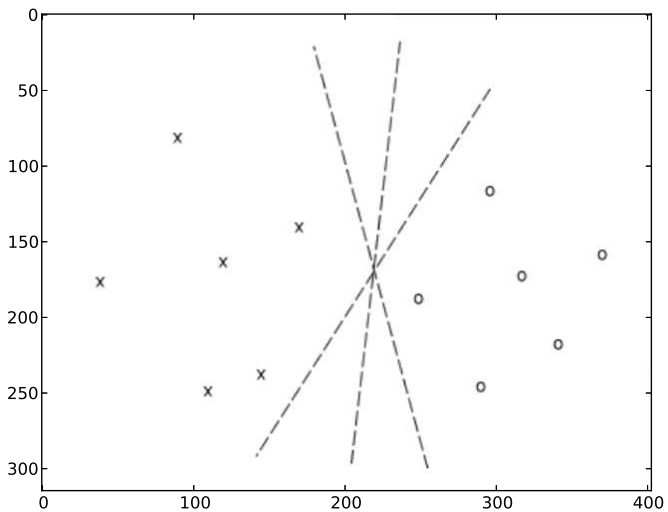
learner	acc	$\rho(KL)$
nb	0.778	-0.36
perc	0.727	-0.21
nn	0.627	-0.39



Exercise: Why might perceptron be less affected by *KL*-divergence?

Large-margin perceptrons

Large- and small-margin solutions are equally good to the standard perceptron. It turns out that larger margins can be obtained by **averaging** stored weights $w^0 \ldots \mathbf{w}^{K \times N}$. The **passive-agressive** perceptron is also *aggressive* in the sense that it forces $\mathbf{w}^{i+1}$ to classify the latest instance correctly regardless of the step-size required.

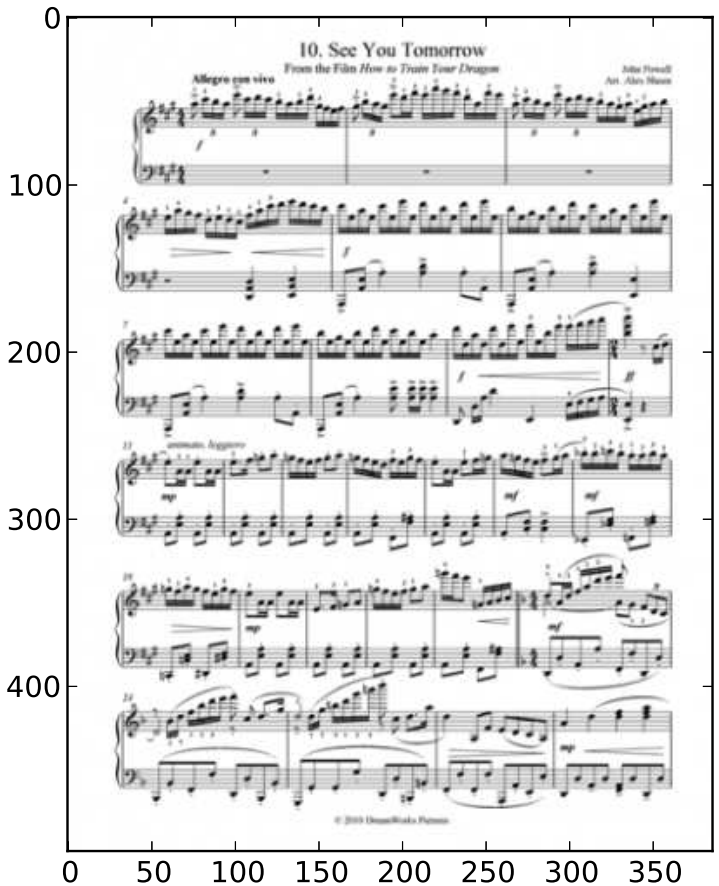


Large-margin perceptrons

dataset $X = \{\langle y_i, \mathbf{x}_i \rangle\}_{i=1}^{N}$
$\mathbf{w}^0 = 0, \mathbf{v} = 0, i = 0$
for $k \in K$ **do**
  for $n \in N$ **do**
    $\mathbf{w}^{i+1} \leftarrow \arg\min_{\mathbf{w} \in \mathbb{R}^n} ||\mathbf{w}^{i+1} - \mathbf{w}_i||$ s.t. $\text{sign}(\mathbf{w}^{i+1} \cdot \mathbf{x}_n) = y_n$
    $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{w}^{i+1}$
    $i \leftarrow i + 1$
  end for
end for
return $\mathbf{w} = \mathbf{v}/(N \times K)$

Figure: Passive agressive learning

Note: The step from here to MIRA, CW and SVM is small (for man, but . . .).


0
100
200
300
400
0
50
100
150
200
250
300
350
10. See You Tomorrow
From the Film How to Train Your Dragon
Allegro con vivo