Supervised Classification

- Supervised classification: Learn by examples to assign labels to objects
- The learning algorithm takes a training set as input and returns the learned classification function

• Some classical approaches:
  - Naïve Bayes
  - Rocchio
  - K-nearest neighbor

Problem Definition

- Assumptions:
  - Binary classification: Let’s assume there are only two classes (e.g. spam/non-spam or relevant/non-relevant)
  - Vector representation: Any item to be classified can be represented as a \(d\)-dimensional real vector
- Task:
  - Find a linear classifier (i.e. a hyperplane) that divides the space \(\mathbb{R}^d\) into two parts

Example

- A two-dimensional example training set
- Task: Separate it by a straight line!

Any of these linear classifiers would be fine… Which one is best?
**Idea:** Measure the quality of a linear classifier by its margin!

**Margin**

- Margin = The width that the boundary could be increased without hitting a data point

**Maximum Margin Classifiers**

- A maximum margin classifier is the linear classifier with a maximum margin

**Maximum Margin Classifiers (2)**

- The maximum margin classifier is the simplest kind of support vector machine, called a linear SVM
  - Let’s assume for now that there always is such a classifier, i.e. the training set is linearly separable!

**Maximum Margin Classifiers (3)**

- Why maximum margin?
  - It’s intuitive to divide the two classes by a large margin
  - The largest margin guards best against small errors in choosing the “right” separator
  - This approach is robust since usually only a small fraction of all data points are support vectors
  - There are some theoretical arguments why this is a good thing
  - Empirically, it works very well
Finding MM Classifiers

• How to formalize this approach?
• Training data:
  – Let there be n training examples
  – The i-th training example is a pair \((y_i, z_i)\),
    where \(y_i\) is a d-dimensional real vector and \(z_i \in \{-1, 1\}\)
  – "-1" stands for the first class and
    "1" stands for the second class

\[
\begin{align*}
  -1, -1 & & (1, 2), -1 & & (1, 0), -1 & & (1, 2), -1 & & (4, 1), 1 & & (5, -1), 1 \\
  x_1 - x_2 - 2 & < 0 & x_1 - x_2 - 2 & > 0
\end{align*}
\]

Finding MM Classifiers (2)

• What's a valid linear separator?
• Any hyperplane can be defined by a real row vector \(w\) and a scalar \(b\)
  – The set of points located on the hyperplane is given by
    \[ \{ x \in \mathbb{R}^d | w \cdot x + b = 0 \} \]
  – \(w\) is a normal vector of the hyperplane, i.e. \(w\) is perpendicular to it.
  – \(b\) represents a shift from the origin of the coordinate system

Finding MM Classifiers (3)

• Therefore, any valid separating hyperplane \((w, b)\) must satisfy the following constraints, for any \(i = 1, \ldots, n\):
  – If \(z_i = -1\), then \(w \cdot y_i + b < 0\)
  – If \(z_i = 1\), then \(w \cdot y_i + b > 0\)

\[
\begin{align*}
  -1, -1 & & (1, 2), -1 & & (1, 0), -1 & & (1, 2), -1 & & (4, 1), 1 & & (5, -1), 1 \\
  x_1 - x_2 - 2 & < 0 & x_1 - x_2 - 2 & > 0
\end{align*}
\]

Finding MM Classifiers (4)

• Furthermore, if \((w, b)\) is a valid separating hyperplane, then there are scalars \(r_+, r_- > 0\) such that
  \[ w \cdot x + b + r_+ = 0 \quad \text{and} \quad w \cdot x + b - r_- = 0 \]
  are the hyperplanes that define the boundaries to the "-1" class and the "1" class, respectively
  – The support vectors are located on these hyperplanes!

Finding MM Classifiers (5)

• Let \((w, b)\) be a valid separating hyperplane with scalars \(r_+\) and \(r_-\) as defined above
• Observation 1:
  Define \(b' = b + (r_- - r_+) / 2\). Then, the hyperplane \(w \cdot x + b' = 0\) is a valid separating hyperplane with equal shift constants \(r' = (r_- - r_+) / 2\) to its bounding hyperplanes (the margin width is the same)

\[
\begin{align*}
  w \cdot x + b + r_+ & = 0 & w \cdot x + b - r_- & = 0 \\
  w \cdot x + b' + r' & = 0 & w \cdot x + b' - r' & = 0
\end{align*}
\]

Finding MM Classifiers (6)

• Now, divide \(w, b', \text{ and } r' \) by \(r'\)
  – This does not change any of the three hyperplanes…
• Observation 2:
  Define \(w'' = w / r'\) and \(b'' = b' / r'\).
  Then, the hyperplane \(w'' \cdot x + b'' = 0\) is a valid separating hyperplane with shift constant 1 to each of its bounding hyperplanes

\[
\begin{align*}
  w \cdot x + b' + r' & = 0 & w'' \cdot x + b'' + 1 & = 0 \\
  w \cdot x + b' - r' & = 0 & w'' \cdot x + b'' - 1 & = 0 \\
  w \cdot x + b & = 0 & w'' \cdot x + b'' & = 0
\end{align*}
\]
The problem then becomes: Maximize \( \frac{2}{||\mathbf{w}||} \) over all \( \mathbf{w} \in \mathbb{R}^d \) and \( b \in \mathbb{R} \)
subject to the following constraints:

- \((\mathbf{w}', b')\) is a valid separating hyperplane
- \((\mathbf{w}, b)\) and \((\mathbf{w}', b')\) have equal margin widths
- the bounding hyperplanes of \((\mathbf{w}', b')\) are shifted away by 1

Therefore, to find a maximum margin classifier, we can limit the search to all hyperplanes of this special type.

Further advantage:
It seems to be a good idea to use a linear classifier that lies equally spaced between its bounding hyperplanes.

Further advantage:

We arrive at the following optimization problem:

Maximize \( \frac{2}{||\mathbf{w}||} \) over all \( \mathbf{w} \in \mathbb{R}^d \) and \( b \in \mathbb{R} \)
subject to the constraints from the previous slide.

Further advantage:

Since any optimal solution satisfies them anyway.

Further advantage:

Note that due to the “maximize the margin” goal, the last two constraints are not needed anymore.

Further advantage:

Therefore, the margin width is \( \frac{2}{||\mathbf{w}||} \).

Further advantage:

Consequently, our goal is to maximize the margin width subject to the constraints from the previous slide.

Further advantage:

Finding MM Classifiers (7)

Finding MM Classifiers (8)

Finding MM Classifiers (9)

Finding MM Classifiers (10)

Finding MM Classifiers (11)

Finding MM Classifiers (12)
Finally, minimize $0.5 \|w\|^2$ over all $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ subject to the following constraints:
- For any $i = 1, \ldots, n$:
  $$z_i \cdot (w \cdot y + b) - 1 \geq 0$$

This is a so-called quadratic programming (QP) problem.
- There are many standard methods to find the solution...

This problem is called the dual optimization problem and has the same optimal solutions as the original problem (if one ignores $\alpha$); but usually it is easier to solve.

Important property:
If $\alpha_i > 0$ in a solution of the above problem, then the corresponding data point $y_i$ is a support vector.
- Consequence: Usually, most $\alpha_i$ are zero, which makes things easy.

The classification function then becomes:
$$f(x) = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i z_i y_i^T x + b \right)$$

$b$ can be computed as follows, using any $i$ such that $\alpha_i > 0$:
$$b = z_i - \sum_{j=1}^{n} \alpha_j z_j y_j^T y_i$$

Note that $f$ can be directly expressed in terms of the support vectors.
Furthermore, computing $f$ basically depends on scalar products of vectors ($y_i^T \cdot x$), which is a key feature in advanced applications of SVMs.

So-called soft margins can be used to handle such cases.
- We allow the classifier to make some mistakes on the training data.
- Each misclassification gets assigned an error, the total classification error then is to be minimized.
We arrive at a new optimization problem

\[
\begin{aligned}
\text{Minimize} & \quad 0.5 \|w\|^2 + C \cdot (\beta_1 + \cdots + \beta_n) \\
\text{subject to} & \quad w \in \mathbb{R}^d, \quad b \in \mathbb{R}, \quad \beta \in \mathbb{R}^n
\end{aligned}
\]

over all \((w, b, \beta)\) satisfying \(w \in \mathbb{R}^d, b \in \mathbb{R}, \beta \in \mathbb{R}^n\) subject to the following constraints:

- For any \(i = 1, \ldots, n\):
  \[\beta_i \geq 0\]
  \[z_i \cdot (w \cdot y + b) - 1 \geq -\beta_i\]

If the \(i\)-th data point gets misclassified by \(\beta_i\), the price we pay for it is \(C \cdot \beta_i\).

\(C\) is a positive constant that regulates how expensive errors should be.

With soft margins, we can drop the assumption of linear separability

The corresponding dual problem is:

\[
\begin{aligned}
\text{Maximize} & \quad \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j z_i y_j (\langle w_i, w_j \rangle + b_i b_j)
\end{aligned}
\]

subject to \(C \geq \alpha_i \geq 0\), for any \(i\), and \(\alpha_1 z_1 + \cdots + \alpha_n z_n = 0\)

Note that only an upper bound on \(\alpha\) is added here.

- Still, it is possible to find solutions efficiently.

At the beginning, we also assumed that there are only two classes in the training set.

How to handle more than that?

Some ideas:

- One-versus-all classifiers:
  - Build an SVM for any class that occurs in the training set.
  - To classify new items, choose the greatest margin’s class.

- One-versus-one classifiers:
  - Build an SVM for any pair of classes in the training set.
  - To classify new items, choose the class selected by most SVMs.

- Multiclass SVMs:
  - (complicated, will not be covered in this course)

Now we are able to handle linearly separable data sets (perhaps with a few exceptions or some noise).

But what to do with this (one-dimensional) data set?

Obviously, it is not linearly separable, and the reason for that is not noise.

What we want to do:

Transform the data set into some higher-dimensional space and do a linear classification there...

Lecture 9: Support Vector Machines

1. Linear SVMs
2. Nonlinear SVMs
3. Support Vector Machines in IR
4. Overfitting

Nonlinear SVMs

Solution:
Transform the data set into some higher-dimensional space and do a linear classification there...
But…

When working in high-dimensional spaces, computing the transformation and solving the corresponding optimization problem will be **horribly difficult**.

What can we do about it?

**Observation:** There are no problems at all if we are able to compute scalar products in the high-dimensional space efficiently…

$$
\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j h(x_i)^T h(x_j)
$$

$$
f(x) = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i h(x_i)^T h(x) + b \right)
$$

The key technique here is called the **“kernel trick”**

Let $h: \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ be some function that maps our original $d$-dimensional data into some $d'$-dimensional space.

Typically $d' \gg d$ holds.

To deal with our optimization problem and be able to do classification afterwards, we must be able to quickly compute the following expressions:

$$
f(x) = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i h(x_i)^T h(x) + b \right)
$$

$$
b = z_i - \sum_{i=1}^{n} y_i \alpha_i h(x_i)^T h(x_i)
$$

Note that we only need to compute **scalar products** in the high-dimensional space.

If $h$ is some special type of mapping (e.g. polynomial or Gaussian), there are computationally simple **kernel functions** available, which correspond to the result of scalar products in $h$’s range.

A polynomial transformation of degree 2:

$$
h(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1x_2)^T
$$

A demo of nonlinear SVMs:

http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml

Another demo:

- Statistical Pattern Recognition Toolbox
  http://cmp.felk.cvut.cz/cmp/software/stpportool

Lecture 9: Support Vector Machines

1. Linear SVMs
2. Nonlinear SVMs
3. Support Vector Machines in IR
4. Overfitting
Text Classification

- An important application of SVMs in information retrieval is text classification.
- Typically, this means automatically assigning topics to new documents based on a training collection of manually processed documents.
  - But there are also many other applications, e.g., spam detection.
- In SVMs, document representations known from the vector space model can be used.
  - Plus additional features, e.g., document length.
- Although the dimensionality is very high, this usually is not a big problem since most document vectors are very sparse.

Learning to Rank

- A very recent application of SVM in information retrieval is called “Learning to Rank.”
- Here, a special type of SVMs is used: Ranking SVMs.
- The training set consists of $n$ pairs of documents $(y_i, y'_i)$.
- Each such pair expresses that document $y_i$ is preferred to $y'_i$ with respect to some fixed query shared by all training pairs.
- Example training set for query “Viagra”:
  - Wikipedia’s entry “Viagra” is preferred to some spam page.
  - Wikipedia’s entry “Viagra” is preferred to the manufacturer’s official page.
  - The manufacturer’s official page is preferred to some spam page.

Learning to Rank (2)

- The task in Learning to Rank: Find a ranking function that assigns a numerical score $s(d)$ to each document $d$ based on its vector representation such that $s(d) > s(d')$ if and only if document $d$ is preferred to document $d'$.
- A straightforward approach are linear ranking functions, i.e., $s(d) = w \cdot d$, for some row vector $w$.
- This reminds us of SVMs.

Learning to Rank (3)

- An SVM formulation of our task is...
  - Minimize $0.5 \|w\|^2$ over all $w \in \mathbb{R}^d$ subject to the following constraints:
    - For any $i = 1, \ldots, n$:
      - $w \cdot y_i \geq w \cdot y'_i + 1$
      - Enforces a standard margin of 1 between each pair of scores.
    - The constraint is equivalent to $w \cdot (y_i - y'_i) - 1 \geq 0$, which looks familiar.
    - Of course, we could also use a soft margin or nonlinear scoring functions here.

Learning to Rank (4)

- Where to get the preference pairs from?
  - Idea from Joachims (2002): Users tend to linearly read a search engine’s result lists down from its beginning.
  - If users click the $r$-th result but do not click the $(r-1)$-th, then document $r$ likely to be preferred to document $r-1$.

Text Classification (2)

- SVMs have been successfully applied in text classification on small and medium-sized document collections.
- Some results by Joachims (1998) from experiments on the Reuters-21578 data set (F-measure with $\alpha = 0.5$):
  - Categories | NB | Rocchio | Ave. Ters | kNN | C = 0.5 | C = 1.0 | kNN | $\alpha = 0.5$
  - euro | 96.4 | 96.1 | 96.3 | 95.8 | 95.5 | 96.0 | 95.6 | 98.1
  - eng | 96.7 | 92.1 | 95.3 | 95.4 | 95.8 | 95.0 | 95.6 | 96.7
  - money:fx | 79.6 | 76.7 | 75.4 | 75.4 | 76.0 | 75.3 | 76.5 | 76.4
  - sports | 69.9 | 75.9 | 74.6 | 76.3 | 76.1 | 74.9 | 76.5 | 76.2
  - trade | 77.2 | 76.2 | 75.7 | 76.2 | 76.2 | 75.7 | 76.5 | 76.4
  - health | 68.9 | 63.1 | 65.0 | 70.6 | 67.4 | 66.5 | 65.5 | 65.8
  - stock | 63.4 | 79.4 | 65.5 | 73.9 | 84.8 | 86.6 | 85.4 | 87.2
  - euro | 65.2 | 62.2 | 67.7 | 71.4 | 87.9 | 87.9 | 86.9 | 86.9
  - micro | 72.3 | 70.9 | 78.4 | 82.6 | 86.7 | 87.5 | 86.4 | 86.4

Information Retrieval and Web Search Engines — Wolf-Tilo Balke and Joachim Selke — Technische Universität Braunschweig
Learning to Rank (5)

- Then:
  1. Compute an initial result list using some retrieval algorithm
  2. Collect user clicks
  3. Learn a ranking function
  4. Incorporate the ranking function into the retrieval process, i.e. re-rank the result list

- Of course, one could use the ranking information already in computing the initial result list
  - ... if user feedback on similar queries is available
  - ... if feedback from different users on the same query is available

Applications:


Handwritten Digits

- Particularly popular: Recognition of handwritten digits

Results

- Taken from Decoste/Schölkopf:

<table>
<thead>
<tr>
<th>Digit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>SV</td>
<td>5</td>
<td>5</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>12</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>1.22%</td>
</tr>
<tr>
<td>USV</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>13</td>
<td>0.88%</td>
</tr>
<tr>
<td>SV12</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>11</td>
<td>8.06%</td>
</tr>
</tbody>
</table>

Lecture 9: Support Vector Machines

1. Linear SVMs
2. Nonlinear SVMs
3. Support Vector Machines in IR
4. Overfitting
One problem in using SVMs remains: if we use a mapping to a high-dimensional space that is "complicated enough," we could find a perfect linear separation in the transformed space, for any training set. So, what type of SVM is the "right" one? Example: How to separate this data set into two parts?

A perfect classification for the training set could generalize badly on new data. Fitting a classifier too strongly to the specific properties of the training set is called overfitting. What can we do to avoid it? Cross-validation:
- Randomly split the available data into two parts (training set + test set)
- Use the first part for learning the classifier and the second part for checking the classifier's performance
- Choose a classifier that maximizes performance on the test set.

Regularization:
- If you know how a "good" classifier roughly should look like (e.g., polynomial of low degree) you could introduce a penalty value into the optimization problem
- Assign a large penalty if the type of classifier is far away from what you expect, and a small penalty otherwise
- Choose the classifier that minimizes the overall optimization goal (original goal + penalty)
- An example of regularization is the soft margin technique since classifiers with large margins and few errors are preferred.

Usually, there is a tradeoff in choosing the "right" type of classifier:
- Ignoring specific characteristics of the training set leads to a systematic bias in classification
- Accounting for all individual properties of the training set leads to a large variance over classifiers when the training set is randomly chosen from some large "true" data set

What you want is small bias and small variance.
- Typically, you cannot have both!

Next Lecture
- Introduction to Web retrieval