Sources

- Previous KDD I lectures on LMU (Johannes Aßfalg, Christian Böhm, Karsten Borgwardt, Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Jörg Sander, Matthias Schubert, Arthur Zimek)

- Jiawei Han, Micheline Kamber and Jian Pei, *Data Mining: Concepts and Techniques, 3rd ed.*, Morgan Kaufmann, 2011.

- Tan P.-N., Steinbach M., Kumar V., *Introduction to Data Mining*, Addison-Wesley, 2006

- Boosting tutorial by Robert Schapire, Machine Learning Summer School (MLSS), Chicago 2005 (http://videolectures.net/mlss05us_schapire_b/)

Outline

• Introduction

• Support Vector Machines

• Ensembles of classifiers

• An overview of classification

• Things you should know

• Homework/tutorial
Support Vector Machines (SVM)

• A popular classification method

• Its roots are in statistical learning theory

• Promising results in many applications, e.g., handwritten text classification, text categorization

• The decision boundary is represented using a subset of the training examples, support vectors
Basic idea

Let's start with a simple 2 class problem

- Goal: find a hyperplane (decision boundary) that will separate the data based on their class
  - In 2D this is just a straight line
Finding a hyperplane I

One possible solution
Another possible solution
Finding a hyperplane III

Other possible solutions
Choosing a hyperplane I

- Which hyperplane is better?
- How do you define better?
Choosing a hyperplane II

Find hyperplane that maximizes the margin \[ \Rightarrow B_1 \text{ is better than } B_2 \]
A linear SVM searches for a hyperplane that maximizes the margin (maximal margin classifier).

Consider a simple 2 class problem. Let $D=(x_i)$ and $y_i={-1,1}$.

We can represent a linear classifier by: $\vec{w} \cdot \vec{x} + b = 0$

- $\vec{w}$ is a weight vector and $b$ a scalar (bias).

How to use it for prediction?

$$y(\vec{z}) = \begin{cases} 
1 & \text{if } \vec{w} \cdot \vec{z} + b \geq 1 \\
-1 & \text{if } \vec{w} \cdot \vec{z} + b \leq -1 
\end{cases}$$

Support vectors
• The margin of $B_1$ is given by the distance between the two hyperplanes $b_{11}$, $b_{12}$.
• Let $x_1$, $x_2$ be two points in $b_{11}$, $b_{12}$ respectively.

\[
\vec{w} \cdot \vec{x}_1 + b = +1
\]

\[
\vec{w} \cdot \vec{x}_2 + b = -1
\]

\[
\vec{w} \cdot (\vec{x}_1 - \vec{x}_2) = 2
\]

\[
=> \text{margin } d = \frac{2}{||\vec{w}||}
\]

• We want to maximize this margin

\[
||\vec{w}|| = \sqrt{\vec{w} \cdot \vec{w}}
\]
Linear SVM III

- We want to maximize
  \[ d = \frac{2}{\| \vec{w} \|} \]

  - This is equivalent to minimizing the following objective function:
  \[ \min_w \| \vec{w} \| \iff \min_w \frac{\| \vec{w} \|^2}{2} \]

  - but, subject to the following constraints

\[
\begin{align*}
y_i &= \begin{cases} 
1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 1 \\
-1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1
\end{cases} \\
y_i(\vec{w} \cdot \vec{x}_i + b) &\geq 1
\end{align*}
\]

This allows us to perform quadratic programming optimization latter on.
• This is a constrained quadratic optimization problem
  – The constraints are rewritten using a Lagrangian formulation
• The solution (trained SVM) consists of
  – The support vectors
  – The parameters $w, b$ of the decision boundary
• How can I classify a new instance?

$$y(z) = \text{sign}(wz + b) = \text{sign}\left( \sum_{i=1}^{N} \lambda_i y_i x_i z + b \right)$$

– $\lambda_i$: Lagrange multipliers
– $x_i$: is the support vector
– $y_i$: is the class of $x_i$
Linear SVM: nonseparable cases

What if the problem is not linearly separable?

• $B_1$ should be preferred over $B_2$ – it has a wider margin $\rightarrow$ less susceptible to overfitting
• but, the so far SVM formulation is error free $\rightarrow$ Soft margin approach
Soft margin approach I

• Learn a decision boundary that is tolerable to small training errors
• Allows SVM to construct a decision boundary even in cases where the classes are not linearly separable
• **Idea:** trade-off between the width of the margin and the misclassification errors committed by the linear decision boundary

**Original optimization problem**

\[
\begin{align*}
\text{min} & \quad \frac{\| \vec{w} \|^2}{2} \\
\text{subject to} & \quad y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 \\
\end{align*}
\]

**Idea:**

• Relax the constraints to accommodate nonlinearly separable data
• Introduce positive-valued slack variables \( \xi_i \)
Soft margin approach II

- Relaxing by introducing slack variables $\xi_i$, $\xi_i \geq 0$

$$y_i = \begin{cases} 
1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \\
-1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 
\end{cases}$$

$$y_i = \begin{cases} 
1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 - \xi_i \\
-1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 + \xi_i 
\end{cases}$$

- The slack variable $\xi_i$ measures the degree of misclassification of instance $x_i$
- Intuitively, data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it.
Updated definition

- **Need to minimize:** $\frac{||\vec{w}||^2}{2} + C \left( \sum_{i=1}^{N} \xi_i^k \right)$

- **Subject to the following constraints:**

  $y_i = \begin{cases} 
  1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq +1 - \xi_i \\
  -1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1 + \xi_i 
  \end{cases}$

- Can be solved used quadratic programming
  - This way we can learn the parameters $w$, $b$ of the decision boundary

If no constrains on # mistakes, we might end up with a very wide margin with many misclassification errors.

$C$, $k$ are user-specified parameters representing the penalty of misclassifying the training instances.
Nonlinear SVM

What if the decision boundary is not linear?

![Graph showing a nonlinear decision boundary](image)
Idea 1

• Trick: transform the data from its original space $x$ into a new space $\Phi(x)$ so that a linear decision boundary can be used to separate the instances in the transformed space

• In $\Phi(x)$, we can apply the same methodology as before to find a linear decision boundary
Idea II

- Intuitively, we extend the hypothesis space

![Diagram showing input and extended feature space with symbols a, b, c, and Φ]
Example I

Input space: \( \vec{x} = (x_1, x_2) \) (2 Attribute)

Extended space (6 Attributes)
\[
\phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2, \sqrt{2} \cdot x_1 \cdot x_2, 1)
\]
Example II

Elliptical boundary in the input space becomes linear in the transformed space

\[
\phi : [x_1, x_2]^T \rightarrow [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T
\]
Nonlinear SVM definition

Updated definition

• Need to minimize: \( \min_w \frac{||\vec{w}||^2}{2} \)

• Subject to the following constraints:

\[ y_i (\vec{w} \bullet \Phi(\vec{x}_i) + b) \geq 1 \]

• Can be solved used quadratic programming
  – This way we can learn the parameters \( w, b \) of the decision boundary

• Classifying a new instance \( z \) (through the transformed space)

\[ f(z) = \text{sign}(w \bullet \Phi(z) + b) = \text{sign}(\sum_{i=1}^{N} \lambda_i y_i \Phi(x_i)\Phi(z) + \beta) \]

Involves calculating of the dot product in the transformed space.
- computational problem (very large vectors)
- curse of dimensionality
The kernel trick is a method for computing similarity between two instances in the transformed feature space using the original attribute set.

- e.g., consider the mapping: \( \Phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1) \)
- The dot product between 2 input vectors \( u, v \) in the transformed space is:

\[
\Phi(u)\Phi(v) = (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1) * (v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1) \\
= u_1^2v_1^2 + u_2^2v_2^2 + 2u_1v_1 + 2u_2v_2 + 1 \\
= (uv + 1)^2
\]

- So, we can express the dot product in \( \Phi(x) \) in terms of a similarity function in the original feature space

\[
K(u, v) = \Phi(u)\Phi(v) = (uv + 1)^2
\]

A function that returns the dot product between the images of two vectors
The main requirement for a kernel function in nonlinear SVM:
There must exist a transformation such that the kernel function computed for two vectors is equivalent to the dot product between these vectors in the transformed space.

Mercer’s Theorem:
A kernel function $K$ can be expressed as:

$$K(u,v) = \Phi(u)\Phi(v)$$

if and only if, for any function $g(x)$ such that $\int g(x)^2 dx$ is finite, then

$$\int K(x,y)g(x)g(y)dxdy \geq 0$$

These functions are called positive definite kernel functions
Kernel functions

Popular kernel functions:

- **Linear** \( K(x, y) = \langle x, y \rangle \)

- **Polynomial** \( K(x, y) = \left( \langle x, y \rangle + c \right)^d \)

- **Gaussian kernel** \( K(x, y) = \exp\left( -\frac{\|x - y\|^2}{2\sigma^2} \right) \)

- **Radial basis function kernel** \( K(x, y) = \exp\left( -\gamma \cdot \|x - y\|^2 \right) \)

Choosing the right kernel depends on the problem at hand

- A linear kernel allows us to model hyperplanes / a polynomial kernel allows us to model feature conjunctions / radial basis functions allows us to model hyperspheres
- Parameter settings is also important!
Kernel Machines

Radial Basis Kernel

Polynomial kernel (degree 2)
SVM: overview

+ High accuracy classifiers
+ Relatively weak tendency to overfitting
+ Efficient classification of new objects
+ Compact models

– Costly implementation
– Sometimes long training times
– Found models difficult to interpret
Outline

• Introduction

• Support Vector Machines

• Ensembles of classifiers

• An overview of classification

• Things you should know

• Homework/tutorial
Ensemble of classifiers

- **Idea:**
  - Instead of a single model, use a combination of models to increase accuracy.
  - Combine a series of T learned models, $M_1, M_2, ..., M_T$, with the aim of creating an improved model $M^*$.
  - To predict the class of previously unseen records, aggregate the predictions of the ensemble.
How to generate ensembles of classifiers I

• By manipulating the training set
  – Multiple training sets are created by resampling the original training data
  – A classifier is built from each training set using some learning algorithm
  – e.g., bagging, boosting

• By manipulating the input features
  – A subset of features is chosen to form each training set (randomly or by domain experts)
  – A base classifier is built from each training set using some learning algorithm
  – e.g., random forests
How to generate ensembles of classifiers II

• By manipulating the class labels
  – Transform into a binary classification problem by randomly partitioning the class labels in two disjoint subsets $A_0, A_1$. Training examples who belong to $A_0$ are assigned to class 0, the rest to class 1.
  – The relabeled examples are used to train a base classifier.
  – Repeat the class-relabeling and model-building steps multiple times to derive the ensemble.
  – During testing, if the test instance is predicted as class 0 (1), all classes in $A_0$ ($A_1$) will receive a vote.

• By manipulating the learning algorithm
  – Many learning algorithms can be manipulated such that applying the same algorithm in the same data might result in different models.
  – e.g., insert randomness in the tree-growing process.
    o e.g., instead of choosing the best splitting attribute choose randomly.
Bagging/ Bootstrap aggregation  
(Breiman, 1996)

• Analogy: Diagnosis based on multiple doctors’ majority vote
• Training: Given a training set D of d tuples
  – In each iteration i: i=1, … , T
    • Randomly sample with replacement from D a training set $D_i$ of $d$ tuples (i.e., bootstrap)
      – On avg, the bootstrap sample contains approximately 63% of the original D
    • Train a chosen “base model” $M_i$ (e.g. neural network, decision tree) on the sample $D_i$
• Testing
  – For each test example
    • Get the predicted class from each trained base model $M_1, M_2, \ldots, M_T$
    • Final prediction by majority voting
### Bagging example I

**Training set**

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 1:**

- x <= 0.35 => y = 1
- x > 0.35 => y = -1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 2:**

- x <= 0.65 => y = 1
- x > 0.65 => y = 1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 3:**

- x <= 0.35 => y = 1
- x > 0.35 => y = -1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 4:**

- x <= 0.3 => y = 1
- x > 0.3 => y = -1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 5:**

- x <= 0.35 => y = 1
- x > 0.35 => y = -1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.6</th>
<th>0.6</th>
<th>0.6</th>
<th>0.6</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 6:**

- x <= 0.75 => y = -1
- x > 0.75 => y = 1

<table>
<thead>
<tr>
<th></th>
<th>0.2</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.7</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 7:**

- x <= 0.75 => y = -1
- x > 0.75 => y = 1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.4</th>
<th>0.4</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>0.9</th>
<th>0.9</th>
<th>0.9</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 8:**

- x <= 0.75 => y = -1
- x > 0.75 => y = 1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.2</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.7</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 9:**

- x <= 0.75 => y = -1
- x > 0.75 => y = 1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.3</th>
<th>0.4</th>
<th>0.4</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.8</th>
<th>0.8</th>
<th>0.9</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bagging Round 10:**

- x <= 0.05 => y = -1
- x > 0.05 => y = 1

<table>
<thead>
<tr>
<th></th>
<th>0.1</th>
<th>0.1</th>
<th>0.1</th>
<th>0.1</th>
<th>0.3</th>
<th>0.3</th>
<th>0.8</th>
<th>0.8</th>
<th>0.8</th>
<th>0.9</th>
<th>0.9</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Combining the predictions

<table>
<thead>
<tr>
<th>Round</th>
<th>x=0.1</th>
<th>x=0.2</th>
<th>x=0.3</th>
<th>x=0.4</th>
<th>x=0.5</th>
<th>x=0.6</th>
<th>x=0.7</th>
<th>x=0.8</th>
<th>x=0.9</th>
<th>x=1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sum</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-6</td>
<td>-6</td>
<td>-6</td>
<td>-6</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Sign</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>True Class</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Bagging overview

- The performance of bagging depends on the stability of base learners
  - If the base learner is unstable, bagging helps to reduce the errors associated with random fluctuations in the training data
  - If a base learner is stable, i.e., robust to minor perturbations of the training set, bagging may not be able to improve the performance of the base learners significantly.
    - It may even degrade the overall performance because the size of each dataset is ~37% smaller than the original data

- It is less susceptible to model overfitting when applied to noisy data
  - since it does not focus on any particular instance of the training data
Boosting

• An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  – Initially, all $N$ records are assigned equal weights
  – Unlike bagging, weights may change at the end of boosting round
    – Records that are wrongly classified will have their weights increased
    – Records that are classified correctly will have their weights decreased

• Adaptive boosting; each classifier is dependent on the previous one and focuses on the previous one’s errors

• Adaboost
Adaboost (Freund and Schapire, 1995)

• Given a training set $D$ of $d$ instances $(x_1, y_1), \ldots, (x_d, y_d)$

• Initially, all instances have the same weight: $1/d$

• A weak learner is trained and its error is computed

• The weights are updated based on the weak learner error
  – If a tuple is misclassified, its weight is increased, o.w. it is decreased

• The new weights are used in the next round

• The final decision (upon the arrival of a new test instance) is a linear combination of the weak learners decisions; the decision of each weak learner is by its error
Adaboost (Freund and Schapire, 1995)

Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)
Initialize \(D_1(i) = 1/m\).
For \(t = 1, \ldots, T\):

- Train weak learner using distribution \(D_t\).
- Get weak hypothesis \(h_t : X \rightarrow \{-1, +1\}\) with error
  \[\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].\]
- Choose \(\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)\).
- Update:
  \[D_{t+1}(i) = \frac{D_t(i) \times \left\{ \begin{array}{ll} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) = -y_i \end{array} \right\}}{Z_t},\]
  where \(Z_t\) is a normalization factor (chosen so that \(D_{t+1}\) will be a distribution).

Output the final hypothesis:
\[H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right).\]
Adaboost example

From: http://videolectures.net/mlss05us_schapire_b/

Round 1

Round 2

Round 3

Final classifier
Boosting overview

- Concentrates on more difficult examples
- Can be quite susceptible to overfitting
  - since it focuses on training examples that are wrongly classified

- Comparing to bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data
Ensembles: overview

• Pros
  – Better classification performance than individual classifiers
  – More resilience to noise

• Cons
  – Time consuming
  – Overfitting

• Necessary conditions
  – The base classifiers should be independent of each other
  – The base classifiers should do better than a classifier that performs random guessing
Outline

• Introduction
• Support Vector Machines
• Ensembles of classifiers

• An overview of classification

• Things you should know
• Homework/tutorial
Overview of the classification process

• **Model construction:**
  – Based on a training set
  – The class label for each training instance is known
  – The output of this step is a model:
    • e.g. a decision tree, Naïve Bayes etc

• **Model evaluation:**
  – Based on a test set
  – The class label for each testing instance is known and is compared with the model prediction
  – The output of this step are some quality measures:
    • e.g. accuracy

• **Model usage:**
  – If the quality is acceptable, use the model to classify data tuples whose class labels are not known

---

Class attribute: tenured={yes, no}

### Training set

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Assistant Prof</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>Mary</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Bill</td>
<td>Professor</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Jim</td>
<td>Associate Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Dave</td>
<td>Assistant Prof</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>Anne</td>
<td>Associate Prof</td>
<td>3</td>
<td>no</td>
</tr>
</tbody>
</table>

### Test set

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
<th>PREDICTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maria</td>
<td>Assistant Prof</td>
<td>3</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>John</td>
<td>Associate Prof</td>
<td>7</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Franz</td>
<td>Professor</td>
<td>3</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Known class label attribute

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
<th>PREDICTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeff</td>
<td>Professor</td>
<td>4</td>
<td>?</td>
<td>yes</td>
</tr>
<tr>
<td>Patrick</td>
<td>Associate Prof</td>
<td>8</td>
<td>?</td>
<td>yes</td>
</tr>
<tr>
<td>Maria</td>
<td>Associate Prof</td>
<td>2</td>
<td>?</td>
<td>no</td>
</tr>
</tbody>
</table>

### Unknown class label attribute

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
<th>PREDICTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeff</td>
<td>Professor</td>
<td>4</td>
<td>?</td>
<td>yes</td>
</tr>
<tr>
<td>Patrick</td>
<td>Associate Prof</td>
<td>8</td>
<td>?</td>
<td>yes</td>
</tr>
<tr>
<td>Maria</td>
<td>Associate Prof</td>
<td>2</td>
<td>?</td>
<td>no</td>
</tr>
</tbody>
</table>
Decision tree classifiers

- A partition-based method

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

- Selecting the best attribute for splitting
- Avoiding overfitting
Naïve Bayes classifiers

• A statistical method

• Maximum likelihood classification

\[ c = \arg \max_{c \in C} P(c \mid X) \]

• Bayes Rule

\[ c = \arg \max_{c \in C} \frac{P(X \mid c)P(c)}{P(X)} = \arg \max_{c \in C} P(X \mid c)P(c) \]

• Independency assumption:

\[ P(X \mid c) = P(A_1 A_2 \ldots A_n \mid c) = \prod P(A_i \mid c) \]

• Estimating:
  • \( P(c) \)
  • \( P(A_i \mid c) \)

• Dealing with 0 probabilities
kNN classifiers

- A similarity-based method
- Learning from your neighbors
- Lazy learner

- Distance function
- # of neighbors (k)

- Voting
  - Majority voting
  - Weighted voting
Support Vector machines

- A statistical method
- Maximizes the margin of the decision boundary

Linear separable

Linear nonseparable

Non linear

- Kernel functions
More methods

- Neural networks
- Ensembles of classifiers
  - Bagging
  - Boosting

http://en.wikibooks.org/wiki/Proteomics/Protein_Identification_-_Mass_Spectrometry/Data_Analysis/_Interpretation
evaluation of classifiers: quality measures

confusion matrix

<table>
<thead>
<tr>
<th>Actual class</th>
<th>C_1</th>
<th>C_2</th>
<th>totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_1</td>
<td>TP (true positive)</td>
<td>FN (false negative)</td>
<td>P</td>
</tr>
<tr>
<td>C_2</td>
<td>FP (false positive)</td>
<td>TN (true negative)</td>
<td>N</td>
</tr>
<tr>
<td>Totals</td>
<td>P'</td>
<td>N'</td>
<td></td>
</tr>
</tbody>
</table>

Different quality measures:

- Accuracy - Error rate
- Sensitivity - Specificity
- Precision - Recall
- $F_1$ score/ F-score/ F-measure
Evaluation of classifiers: train – test sets

- Hold-out method
  - Random sampling

- Cross-validation
  - Leave-one-out
  - Stratified cross-validation

- Bootstrap
  - .632 bootstrap
Outline

• Introduction

• Support Vector Machines

• Ensembles of classifiers

• An overview of classification

• Things you should know

• Homework/tutorial
Things you should know

• Support Vector Machines
  – Formulation
  – Linear separable case
  – Linear nonseparable cases
  – Kernel functions

• Ensemble methods
  – Boosting
  – Bagging
**Tutorial:** this Thursday tutorial on
- Decision trees/ Support Vector Machines

**Homework:**
- Repeat the classification methods learned

**Suggested reading:**
- Han J., Kamber M., Pei J. *Data Mining: Concepts and Techniques 3rd ed.*, Morgan Kaufmann, 2011 (Chapters 8, 9)
- Support Vector Machines tutorial by Chih-Jen Lin, Machine Learning Summer School (MLSS), Taipei 2006 ([http://videolectures.net/mlss06tw_lin_svm/](http://videolectures.net/mlss06tw_lin_svm/))
- Boosting tutorial by Robert Schapire, Machine Learning Summer School (MLSS), Chicago 2005 ([http://videolectures.net/mlss05us_schapire_b/](http://videolectures.net/mlss05us_schapire_b/))