Topics:

- Support vector machines (cont)
- ROC analysis
- Nonparametric methods

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square
Last lecture outline

Outline:

• Algorithms for linear decision boundary
• **Support vector machines**
• Maximum margin hyperplane
• Support vectors
• Support vector machines learning
• Extensions to the linearly non-separable case
• Kernel functions
Optimal separating hyperplane

**Problem:**
- There are multiple hyperplanes that separate the data points.
- Which one to choose?
Optimal separating hyperplane

• **Problem:**
  - There are multiple hyperplanes that separate the data points
  - Which one to choose?
  - The decision boundary that maximizes the distance of the +1 and -1 points from it
Maximum margin hyperplane

- For the maximum margin hyperplane only examples on the margin matter (only these affect the distances)
- These are called **support vectors**
Support vector machines: solution property

- Decision boundary defined by a set of support vectors SV and their alpha values
  - Support vectors = a subset of datapoints in the training data that define the margin

\[
\hat{\mathbf{w}}^T \mathbf{x} + \mathbf{w}_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + \mathbf{w}_0
\]

- Classification decision:

\[
\hat{y} = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + \mathbf{w}_0 \right]
\]

- Note that we do not have to explicitly compute \( \hat{\mathbf{w}} \)
  - This will be important for the nonlinear (kernel) case
Support vector machines: inner product

- Decision on a new $x$ depends on the **inner product** between two examples

- **The decision boundary:**
  \[
  \hat{w}^T x + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (x_i^T x) + w_0
  \]

- **Classification decision:**
  \[
  \hat{y} = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i (x_i^T x) + w_0 \right]
  \]

- Similarly, the optimization depends on $(x_i^T x_j)$

\[
J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
\]
Linearly non-separable case

- **Idea:** Allow some flexibility on crossing the separating hyperplane
Support vector machines: solution

• The solution of the linearly non-separable case has the same properties as the linearly separable case.
  – The decision boundary is defined only by a set of support vectors (points that are on the margin or that cross the margin)
  – The decision boundary and the optimization can be expressed in terms of the inner product in between pairs of examples

\[
\hat{w}^T x + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (x_i^T x) + w_0
\]

\[
\hat{y} = \text{sign} [\hat{w}^T x + w_0] = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i (x_i^T x) + w_0 \right]
\]

\[
J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
\]
Nonlinear decision boundary

So far we have seen how to learn a linear decision boundary

- But what if the linear decision boundary is not good.
- How we can learn a non-linear decision boundaries with the SVM?
The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors

\[ \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]

- **Example:** polynomial expansions
- Note that feature expansions are typically high dimensional

Given the nonlinear feature mappings, we can use the linear SVM on the expanded feature vectors

\[ (\mathbf{x}^T \mathbf{x}') \rightarrow \varphi(\mathbf{x})^T \varphi(\mathbf{x}') \]

**Kernel function (measures similarity)**

\[ K(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^T \varphi(\mathbf{x}') \]
Support vector machines: solution for nonlinear decision boundaries

- The decision boundary:

\[ \hat{w}^T x + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i K(x_i, x) + w_0 \]

- Classification:

\[ \hat{y} = \text{sign}(\hat{w}^T x + w_0) = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i K(x_i, x) + w_0 \right] \]

- Decision on a new \( x \) requires to compute the kernel function defining the similarity between the examples
- Similarly, the optimization depends on the kernel

\[ J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i, j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \]
Kernel trick

• **Feature mapping:**
  \[ x \rightarrow \varphi(x) \]

• **Kernel function** defines the inner product in the expanded high dimensional feature vectors and let us use the SVM
  \[ K(x, x') = \varphi(x)^T \varphi(x') \]

• **Problem:** after expansion we need to perform inner products in a very high dimensional \( \varphi(x) \) space

• **Kernel trick:**
  – If we choose the kernel function \( K(x, x') \) wisely we can compute linear separation in the high dimensional feature space implicitly by working in the original input space !!!!
Kernel function example

• Assume $\mathbf{x} = [x_1, x_2]^T$ and a feature mapping that maps the input into a quadratic feature set

$$\mathbf{x} \rightarrow \phi(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T$$

• Kernel function for the feature space:

$$K(\mathbf{x}', \mathbf{x}) = \phi(\mathbf{x}')^T \phi(\mathbf{x})$$

$$= x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_2x_1'x_2' + 2x_1x_1' + 2x_2x_2' + 1$$

$$= (x_1x_1' + x_2x_2' + 1)^2$$

$$= (1 + (\mathbf{x}^T \mathbf{x}'))^2$$

• The computation of the linear separation in the higher dimensional space is performed implicitly in the original input space
Kernel function example

Linear separator in the expanded feature space

Non-linear separator in the input space
Kernel functions

- **Linear kernel**
  \[
  K(x, x') = x^T x'
  \]

- **Polynomial kernel**
  \[
  K(x, x') = \left[1 + x^T x'\right]^k
  \]

- **Radial basis kernel**
  \[
  K(x, x') = \exp\left[-\frac{1}{2} \|x - x'\|^2\right]
  \]
Kernels

- ML researchers have proposed kernels for comparison of variety of objects.
  - Strings
  - Trees
  - Graphs
- **Cool thing:**
  - SVM algorithm can be now applied to classify a variety of objects
Evaluation of binary classifiers
ROC analysis
Evaluation

For any data set we use to test the classification model on we can build a *confusion matrix*:

- Counts of examples with:
  - class label \( \omega_j \) that are classified with a label \( \alpha_i \)

\[
\begin{array}{c|cc}
\text{predict} & \omega = 1 & \omega = 0 \\
\hline
\alpha = 1 & 140 & 17 \\
\alpha = 0 & 20 & 54 \\
\end{array}
\]
Evaluation

For any data set we use to test the model we can build a confusion matrix:

<table>
<thead>
<tr>
<th>predict</th>
<th>( \omega = 1 )</th>
<th>( \omega = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha = 1 )</td>
<td>140</td>
<td>17</td>
</tr>
<tr>
<td>( \alpha = 0 )</td>
<td>20</td>
<td>54</td>
</tr>
</tbody>
</table>

**Accuracy** = \( \frac{194}{231} \)
Evaluation

For any data set we use to test the model we can build a confusion matrix:

<table>
<thead>
<tr>
<th></th>
<th>$\omega = 1$</th>
<th>$\omega = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 1$</td>
<td>140</td>
<td>17</td>
</tr>
<tr>
<td>$\alpha = 0$</td>
<td>20</td>
<td>54</td>
</tr>
</tbody>
</table>

Accuracy = $\frac{194}{231}$
Error = $\frac{37}{231} = 1 - \text{Accuracy}$
Evaluation for binary classification

Entries in the confusion matrix for binary classification have names:

<table>
<thead>
<tr>
<th>predict</th>
<th>target</th>
<th>( \omega = 1 )</th>
<th>( \omega = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha = 1 )</td>
<td>( TP )</td>
<td>( FP )</td>
<td></td>
</tr>
<tr>
<td>( \alpha = 0 )</td>
<td>( FN )</td>
<td>( TN )</td>
<td></td>
</tr>
</tbody>
</table>

- **TP**: True positive (hit)
- **FP**: False positive (false alarm)
- **TN**: True negative (correct rejection)
- **FN**: False negative (a miss)
Additional statistics

- **Sensitivity (recall)**
  \[ SENS = \frac{TP}{TP + FN} \]

- **Specificity**
  \[ SPEC = \frac{TN}{TN + FP} \]

- **Positive predictive value (precision)**
  \[ PPT = \frac{TP}{TP + FP} \]

- **Negative predictive value**
  \[ NPV = \frac{TN}{TN + FN} \]
# Binary classification: additional statistics

- **Confusion matrix**

<table>
<thead>
<tr>
<th>predict</th>
<th>target</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>140</td>
<td>10</td>
<td>$PPV = 140/150$</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>180</td>
<td>$NPV = 180/200$</td>
</tr>
</tbody>
</table>

Row and column quantities:

- Sensitivity (SENS)  
  $SENS = 140/160$

- Specificity (SPEC)  
  $SPEC = 180/190$

- Positive predictive value (PPV)  
  $PPV = 140/150$

- Negative predictive value (NPV)  
  $NPV = 180/200$
Classifiers

Project datapoints to one dimensional space:
Defined for example by: \( w^T x \) or \( p(y=1|x,w) \)
Binary decisions: Receiver Operating Curves

- **Probabilities:**
  - *SENS*
  - *SPEC*

\[
p(x > x^* \mid x \in \omega_2)\]

\[
p(x < x^* \mid x \in \omega_1)\]
Receiver Operating Characteristic (ROC)

- **ROC curve plots:**

  \[ SN = p(x > x^* \mid x \in \omega_2) \]

  \[ 1-SP = p(x > x^* \mid x \in \omega_1) \]

  for different \( x^* \)

  \[ \text{SENS} \]

  \[ p(x > x^* \mid x \in \omega_2) \]

  \[ 1-\text{SPEC} \]

  \[ p(x > x^* \mid x \in \omega_1) \]
$p(x > x^* \mid x \in \omega_2)^{0.9}$

$p(x > x^* \mid x \in \omega_1)$
Receiver operating characteristic

- **ROC**
  - shows the discriminability between the two classes under different decision biases

- **Decision bias**
  - can be changed using different loss function

- **Quality of a classification model:**
  - Area under the ROC
  - Best value 1, worst (no discriminability): 0.5
Nonparametric Methods

• **Parametric distribution models** are:
  – restricted to specific forms, which may not always be suitable;
  – Example: modelling a multimodal distribution with a single, unimodal model.

• **Nonparametric approaches**:
  – make few assumptions about the overall shape of the distribution being modelled.
Nonparametric Methods
Nonparametric Density Methods

Problem:
• We have a set $D$ of data-points $x_i$ for $i = 1, 2, \ldots n$
• We want to calculate $p(x)$ for a target value of $x$

Parametric approach:
• represents $p(x)$ using a parametric density model with parameters $\theta$
• fits the parameters $\theta$ wrt the data

Nonparametric approach:
• Does not make any parametric assumption
• Estimates $p(x)$ from all datapoints in $D$, as if all $D$ are parameters
Nonparametric Density Methods

**Histogram methods:** partition the data space into distinct bins with widths $\Delta_i$ and count the number of observations, $n_i$, in each bin.

$$p_i = \frac{n_i}{N\Delta_i}$$

- Often, the same width is used for all bins, $\Delta_i = \Delta$.
- $\Delta$ acts as a smoothing parameter.
Nonparametric Density Methods

• Assume observations drawn from a density \( p(x) \) and consider a small region \( R \) containing \( x \) such that

\[
P = \int_R p(x) \, dx
\]

• The probability that \( K \) out of \( N \) observations lie inside \( R \) is \( \text{Bin}(K,N,P) \) and if \( N \) is large

\[
K \approx NP
\]

If the volume of \( R \), \( V \), is sufficiently small, \( p(x) \) is approximately constant over \( R \) and

\[
P \equiv p(x)V
\]

Thus

\[
p(x) = \frac{P}{V}
\]

\[
p(x) = \frac{K}{NV}
\]
Nonparametric Methods: kernel methods

Kernel Density Estimation:

Fix \( V \), estimate \( K \) from the data. Let \( R \) be a hypercube centred on \( x \) and define the kernel function (Parzen window)

\[
k\left( \frac{x - x_n}{h} \right) = \begin{cases} 
1 & \left| (x_i - x_{ni}) \right| / h \leq 1/2 \\
0 & \text{otherwise} 
\end{cases}, \quad i = 1, \ldots, D
\]

- It follows that

\[
K = \sum_{n=1}^{N} k\left( \frac{x - x_n}{h} \right)
\]

- and hence

\[
p(x) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^D} k\left( \frac{x - x_n}{h} \right)
\]
Nonparametric Methods: smooth kernels

To avoid discontinuities in $p(x)$ because of sharp boundaries use a **smooth kernel**, e.g. a Gaussian

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{D/2}} \exp \left\{ - \frac{\|x - x_n\|^2}{2h^2} \right\}$$

- Any kernel such that
  
  $$k(u) \geq 0,$$
  
  $$\int k(u) \, du = 1$$

- will work.
**Nonparametric Methods: kNN estimation**

**Nearest Neighbour Density Estimation:**

fix $K$, estimate $V$ from the data. Consider a hyper-sphere centred on $x$ and let it grow to a volume, $V^*$, that includes $K$ of the given $N$ data points. Then

$$p(x) \simeq \frac{K}{NV^*}.$$
Nonparametric vs Parametric Methods

Nonparametric models:
• More flexibility – no density model is needed
• But require storing the entire dataset
• and the computation is performed with all data examples.

Parametric models:
• Once fitted, only parameters need to be stored
• They are much more efficient in terms of computation
• But the model needs to be picked in advance
Nonparametric classification models

We have a set $D$ of $<x, y>$ pairs

We have a new data point $x$ and want to assign it a class $y$

How?

Algorithm 1

Step 1: Estimate $p(y=1)$ and $p(y=0)$

Step 2: Estimate $p(x | y=1)$ and $p(x | y=0)$ using nonparametric estimation methods and labels

Step 3: choose a class by comparing

$$p(x | y=1) \cdot p(y=1) \quad \text{with} \quad p(x | y=0) \cdot p(y=1)$$
Nonparametric classification models

We have a set $D$ of $<x, y>$ pairs
We have a new data point $x$ and want to assign it a class $y$

How?

**Algorithm 2 (K nearest neighbors)**

Step 1: Find the closest K examples to $x$
Step 2: choose a class by considering the majority of the class labels

A special case: *the nearest neighbour algorithm*