Support vector machines

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square
Outline:

- Fisher Linear Discriminant
- Algorithms for linear decision boundary
- **Support vector machines**
  - Maximum margin hyperplane.
  - Support vectors.
  - Support vector machines.

- Extensions to the non-separable case.
- Kernel functions.
Linear decision boundaries

- What models define linear decision boundaries?
Logistic regression model

- **Discriminant functions:**
  \[ g_1(x) = g(w^T x) \]
  \[ g_0(x) = 1 - g(w^T x) \]

- **where**
  \[ g(z) = \frac{1}{1 + e^{-z}} \] - is a logistic function

\[ f(x, w) = g_1(w^T x) = g(w^T x) \]
Linear discriminant analysis (LDA)

- When covariances are the same
  \[ x \sim N(\mu_0, \Sigma), \ y = 0 \]
  \[ x \sim N(\mu_1, \Sigma), \ y = 1 \]
Linear decision boundaries

- Any other models/algorithms?
Fisher linear discriminant

- Project data into one dimension
  \[ y = \mathbf{w}^T \mathbf{x} \]

Decision: \[ y = \mathbf{w}^T \mathbf{x} + w_0 \geq 0 \]

- How to find the projection line?
Fisher linear discriminant

How to find the projection line?

\[ y = w^T x \]
Fisher linear discriminant

Assume:

\[ m_1 = \frac{1}{N_1} \sum_{i \in C_1} x_i \quad m_2 = \frac{1}{N_2} \sum_{i \in C_2} x_i \]

Maximize the difference in projected means:

\[ m_2 - m_1 = w^T (m_2 - m_1) \]
Fisher linear discriminant

Problem 1: \( m_2 - m_1 = w^T (m_2 - m_1) \) can be maximized by increasing \( w \)

Problem 2: variance in class distributions after projection is changed

Fisher’s solution: \( J(w) = \frac{m_2 - m_1}{s_1^2 + s_2^2} \)

Within class variance \( s_k^2 = \sum_{i \in C_k} (y_i - m_k)^2 \)
Fisher linear discriminant

Error:

\[ J(w) = \frac{m_2 - m_1}{s_1^2 + s_2^2} \]

Within class variance after the projection

\[ s_k^2 = \sum_{i \in C_k} (y_i - m_k)^2 \]

Optimal solution:

\[ w \approx S_w^{-1}(m_2 - m_1) \]

\[ S_w = \sum_{i \in C_1} (x_i - m_1)(x_i - m_1)^T + \sum_{i \in C_2} (x_i - m_2)(x_i - m_2)^T \]
Linearly separable classes

There is a hyperplane that separates training instances with no error.

Hyperplane:
\[ w^T x + w_0 = 0 \]

<table>
<thead>
<tr>
<th>Class (+1)</th>
<th>[ w^T x + w_0 &gt; 0 ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class (-1)</td>
<td>[ w^T x + w_0 &lt; 0 ]</td>
</tr>
</tbody>
</table>
Algorithms for linearly separable set

- **Separating hyperplane** \[ \mathbf{w}^T \mathbf{x} + w_0 = 0 \]

- We can use **gradient methods** or Newton Rhapsone for sigmoidal switching functions and learn the weights
- Recall that we learn the linear decision boundary
Algorithms for linearly separable set

- **Separating hyperplane**

\[ \mathbf{w}^T \mathbf{x} + w_0 = 0 \]
Algorithms for linearly separable sets

**Perceptron algorithm:**
- Simple iterative procedure for modifying the weights of the linear model
- Works for inputs $x$ where each $x_i$ is in $[0,1]$

1. **Initialize** weights $w$
2. **Loop** through examples $(x, y)$ in the dataset $D$
   - 1. Compute $\hat{y} = w^T x$
   - 2. If $y \neq \hat{y} = -1$ then $w^T \leftarrow w^T + x$
   - 3. If $y \neq \hat{y} = +1$ then $w^T \leftarrow w^T - x$
3. **Until** all examples are classified correctly

**Properties:**
- guaranteed convergence if the classes are linearly separable
Algorithms for linearly separable sets

Linear program solution:
• Finds weights that satisfy the following constraints:

\[ w^T x_i + w_0 \geq 0 \quad \text{For all } i, \text{ such that } y_i = +1 \]

\[ w^T x_i + w_0 \leq 0 \quad \text{For all } i, \text{ such that } y_i = -1 \]

Together:
\[ y_i (w^T x_i + w_0) \geq 0 \]

Property: if there is a hyperplane separating the examples, the linear program finds the solution
Optimal separating hyperplane

- There are multiple hyperplanes that separate the data points
  - Which one to choose?
- **Maximum margin** choice: maximum distance of $d_+ + d_-$
  - where $d_+$ is the shortest distance of a positive example from the hyperplane (similarly $d_-$ for negative examples)
Maximum margin hyperplane

• For the maximum margin hyperplane only examples on the margin matter (only these affect the distances)
• These are called support vectors
Finding maximum margin hyperplanes

• **Assume** that examples in the training set are \((x_i, y_i)\) such that \(y_i \in \{+1, -1\}\)

• **Assume** that all data satisfy:

\[
\begin{align*}
    w^T x_i + w_0 &\geq 1 \quad \text{for} \quad y_i = +1 \\
    w^T x_i + w_0 &\leq -1 \quad \text{for} \quad y_i = -1
\end{align*}
\]

• The inequalities can be combined as:

\[
y_i (w^T x_i + w_0) - 1 \geq 0 \quad \text{for all} \quad i
\]

• Equalities define two hyperplanes:

\[
\begin{align*}
    w^T x_i + w_0 &= 1 \\
    w^T x_i + w_0 &= -1
\end{align*}
\]
Finding the maximum margin hyperplane

• **Geometrical margin:** \( \rho_{w,w_0}(x, y) = y(w^T x + w_0) / \|w\|_{L^2} \)
  – measures the distance of a point \( x \) from the hyperplane
  \( w \) - normal to the hyperplane  \( \|\cdot\|_{L^2} \) - Euclidean norm

For points satisfying:
\[
y_i (w^T x_i + w_0) - 1 = 0
\]

The distance is
\[
\frac{1}{\|w\|_{L^2}}
\]

**Width of the margin:**
\[
d_+ + d_- = \frac{2}{\|w\|_{L^2}}
\]
Maximum margin hyperplane

- We want to maximize $d_+ + d_- = \frac{2}{\|w\|_{L2}}$

- We do it by minimizing
  
  \[ \|w\|_{L2}^2 / 2 = w^T w / 2 \]

  \[ w, w_0 \] - variables

  - But we also need to enforce the constraints on points:

  \[ [y_i (w^T x + w_0) - 1] \geq 0 \]
Maximum margin hyperplane

- **Solution:** Incorporate constraints into the optimization
- **Optimization problem** (Lagrangean)

\[
J(w, w_0, \alpha) = \|w\|^2 / 2 - \sum_{i=1}^{n} \alpha_i [y_i (w^T x + w_0) - 1]
\]

\[
\alpha_i \geq 0 \quad - \text{Lagrange multipliers}
\]

- **Minimize** with respect to \(w, w_0\) (primal variables)
- **Maximize** with respect to \(\alpha\) (dual variables)

Lagrange multipliers enforce the satisfaction of constraints

If \(y_i (w^T x + w_0) - 1 > 0\) \(\iff\) \(\alpha_i \to 0\)

Else \(\iff\) \(\alpha_i > 0\) Active constraint
Max margin hyperplane solution

• Set derivatives to 0 (Kuhn-Tucker conditions)

\[ \nabla_w J(w, w_0, \alpha) = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \]

\[ \frac{\partial J(w, w_0, \alpha)}{\partial w_0} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \]

• Now we need to solve for Lagrange parameters (Wolfe dual)

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

Subject to constraints

\[ \alpha_i \geq 0 \quad \text{for all } i, \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \]

• Quadratic optimization problem: solution \( \hat{\alpha}_i \) for all i
Maximum margin solution

- The resulting parameter vector $\hat{\mathbf{w}}$ can be expressed as:

$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$$

$\hat{\alpha}_i$ is the solution of the dual problem

- The parameter $w_0$ is obtained through Karush-Kuhn-Tucker (KKT) conditions

$$\hat{\alpha}_i [y_i (\hat{\mathbf{w}}\mathbf{x}_i + w_0) - 1] = 0$$

Solution properties

- $\hat{\alpha}_i = 0$ for all points that are not on the margin
- $\hat{\mathbf{w}}$ is a linear combination of support vectors only
- The decision boundary:

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 = 0$$
Support vector machines

• The decision boundary:

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

• The decision:

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

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

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

- (!!!):
  - Decision on a new \( x \) requires to compute the inner product between the examples \( (x_i^T x) \)
  - Similarly, the optimization depends on \( (x_i^T x_j) \)

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

- **Idea:** Allow some flexibility on crossing the separating hyperplane
Extension to the linearly non-separable case

• Relax constraints with variables $\xi_i \geq 0$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \geq 1 - \xi_i \quad \text{for} \quad y_i = +1$$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \leq -1 + \xi_i \quad \text{for} \quad y_i = -1$$

• Error occurs if $\xi_i \geq 1$, $\sum_{i=1}^{n} \xi_i$ is the upper bound on the number of errors

• Introduce a penalty for the errors

$$\text{minimize} \quad \|\mathbf{w}\|^2 / 2 + C \sum_{i=1}^{n} \xi_i$$

Subject to constraints

$C$ – set by a user, larger $C$ leads to a larger penalty for an error
Extension to linearly non-separable case

- Lagrange multiplier form (primal problem)

\[ J(w, w_0, \alpha) = \|w\|^2 / 2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left[ y_i (w^T x + w_0) - 1 + \xi_i \right] - \sum_{i=1}^{n} \mu_i \xi_i \]

- Dual form after \( w, w_0 \) are expressed (\( \xi_i \) s cancel out)

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

Subject to: \( 0 \leq \alpha_i \leq C \) for all i, and \( \sum_{i=1}^{n} \alpha_i y_i = 0 \)

Solution: \( \hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i \)

The difference from the separable case: \( 0 \leq \alpha_i \leq C \)

The parameter \( w_0 \) is obtained through KKT conditions
Support vector machines

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

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

- (!!):
  - Decision on a new \( \mathbf{x} \) requires to compute the inner product between the examples \((\mathbf{x}_i^T \mathbf{x})\)
  - Similarly, the optimization depends on \((\mathbf{x}_i^T \mathbf{x}_j)\)

\[
J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)
\]
Nonlinear case

- The linear case requires to compute \( (x_i^T x) \)
- The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors
  \[ x \rightarrow \phi(x) \]
- It is possible to use SVM formalism on feature vectors
  \[ \phi(x)^T \phi(x') \]
- **Kernel function**
  \[ K(x, x') = \phi(x)^T \phi(x') \]

- **Crucial idea:** If we choose the kernel function wisely we can compute linear separation in the feature space implicitly such that we keep 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:

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

\[
= x_1^2 x'_1^2 + x_2^2 x'_2^2 + 2x_1 x_2 x'_1 x'_2 + 2x_1 x'_1 + 2x_2 x'_2 + 1
\]

\[
= (x_1 x'_1 + x_2 x'_2 + 1)^2
\]

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

- 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 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

- Kernels can be defined for more complex objects:
  - Strings
  - Graphs
  - Images
- Kernel – similarity between pairs of objects