Support vector machines

Outline

Outline:
- Algorithms for linear decision boundary
- Support vector machines
- Maximum margin hyperplane.
- Support vectors.
- Support vector machines.
- Extensions to the non-separable case.
- Kernel functions.
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>Class (-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w^T x + w_0 &gt; 0 )</td>
<td>( w^T x + w_0 &lt; 0 )</td>
</tr>
</tbody>
</table>

Logistic regression

- **Separating hyperplane:** \( w^T x + w_0 = 0 \)

  \[
  y = \sum w_i x_i + w_0
  \]

- We can use gradient methods or Newton Rhapson for sigmoidal switching functions and learn the weights.
- Recall that we learn the linear decision boundary.
Solving via LP

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: maximizes distance \( d_+ + d_- \)
  - 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**

![Diagram showing support vectors and maximum margin hyperplane]

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:
  \[
  w^T x_i + w_0 \geq 1 \quad \text{for} \quad y_i = +1 \\
  w^T x_i + w_0 \leq -1 \quad \text{for} \quad y_i = -1
  \]
- 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:
  \[
  w^T x_i + w_0 = 1 \quad \quad w^T x_i + w_0 = -1
  \]
Finding the maximum margin hyperplane

- **Distance of a point** $x$ with label 1 **from the hyperplane**: 
  
  $$d(x) = \frac{(w^T x + w_0)}{\|w\|_{L^2}}$$

  $w$ - normal to the hyperplane  $\|\cdot\|_{L^2}$ - Euclidean norm

- **Distance of a point** $x'$ with label -1:
  
  $$d(x') = -\frac{(w^T x' + w_0)}{\|w\|_{L^2}}$$

- **Distance of a point** $x$ with label $y$:
  
  $$\rho_{w,w_0}(x, y) = y(w^T x + w_0)/\|w\|_{L^2}$$

**Finding the maximum margin hyperplane**

- **Geometrical margin**: 
  
  $$\rho_{w,w_0}(x, y) = y(w^T x + w_0)/\|w\|_{L^2}$$

  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:
    \[
    \left[ y_i (w^T x + w_0) - 1 \right] \geq 0
    \]

---

Maximum margin hyperplane

- **Solution**: Incorporate constraints into the optimization
- **Optimization problem** (Lagrangian)
  \[
  J(w, w_0, \alpha) = \|w\|^2 / 2 - \sum_{i=1}^{n} \alpha_i [y_i (w^T x + w_0) - 1]
  \]
  \( \alpha_i \geq 0 \) - 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 \( \left[ y_i (w^T x + w_0) - 1 \right] > 0 \) \( \Rightarrow \alpha_i \rightarrow 0 \)
- Else \( \alpha_i > 0 \) Active constraint

---

CS 2750 Machine Learning
Max margin hyperplane solution

• Set derivatives to 0 (Karush-Kuhn-Tucker (KKT) 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)
\[ J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \]
\[ \text{maximize} \]

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 hyperplane solution

• The resulting parameter vector \( \hat{w} \) can be expressed as:
\[ \hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i y_i x_i \]
\( \hat{\alpha}_i \) is the solution of the dual problem

• The parameter \( w_0 \) is obtained through Karush-Kuhn-Tucker conditions
\[ \hat{\alpha}_i [y_i (\hat{w} x_i + w_0) - 1] = 0 \]

**Solution properties**

• \( \hat{\alpha}_i = 0 \) for all points that are not on the margin
• \( \hat{w} \) is a linear combination of support vectors only

• **The decision boundary:**
\[ \hat{w}^T x + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (x_i^T x) + w_0 = 0 \]
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:
  \[ \hat{y} = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i (x_i^T x) + w_0 \right] \]

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

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

• **Idea:** Allow some flexibility on crossing the separating hyperplane

![Graph showing linearly non-separable data points and support vectors]

Extension to the linearly non-separable case

• Relax constraints with variables $\xi_i \geq 0$
  \[
  w^T x_i + w_0 \geq 1 - \xi_i \quad \text{for} \quad y_i = +1 \\
  w^T x_i + w_0 \leq -1 + \xi_i \quad \text{for} \quad y_i = -1
  \]

• 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
  \[
  \text{minimize} \quad \|w\|^2 / 2 + C \sum_{i=1}^{n} \xi_i
  \]

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:

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

- 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 \( (x_i^T x_j) \)

\[ J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \]
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 \(x = [x_1, x_2]^T\) and a feature mapping that maps the input into a quadratic feature set
  \(x \rightarrow \phi(x) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T\)
- Kernel function for the feature space:
  \[K(x', x) = \phi(x')^T \phi(x)\]
  \[= x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_2x_1'x_2' + 2x_1x_1' + 2x_2x_2' + 1\]
  \[= (x_1x_1' + x_2x_2' + 1)^2\]
  \[= (1 + (x^T x'))^2\]
- 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

- The dot product $\mathbf{x}^T \mathbf{x}$ is a **distance measure**
- **Kernels** can be seen as distance measures
  - Or conversely express degree of similarity
- Design criteria - we want kernels to be
  - **valid** – Satisfy Mercer condition of positive semi-definiteness
  - **good** – embody the “true similarity” between objects
  - **appropriate** – generalize well
  - **efficient** – the computation of $k(x,x')$ is feasible

Kernels

- SVM 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