CS 3750 Machine Learning
Lecture 21

Support vector machines

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

Term Project proposals

Due: **Monday, November 10**

Proposal: **1-2 pages long**

1. Outline of a problem you want to address, type of data you have available. Why is the problem important?
2. Learning methods you plan to develop and implement for the problem. References to previous work.
3. How do you plan to test, compare learning approaches
4. Schedule of work (approximate timeline of work)

Projects:

- **Presentation**
- **Report: due on December 5, 2003**

Outline

Outline:

- **Support vector machines**
- Linearly separable classes. Algorithms.
- Maximum margin hyperplane.
- Support vectors.
- Support vector machines.

- Extensions to the non-separable case.
- Kernel functions.

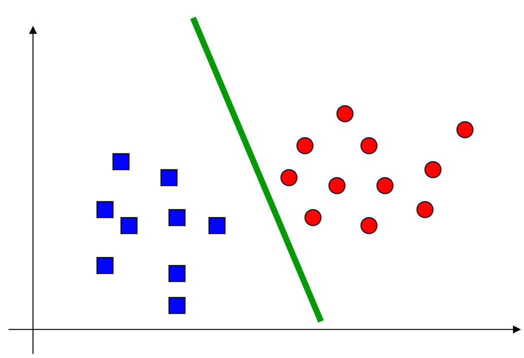
Linearly separable classes

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

Hyperplane:

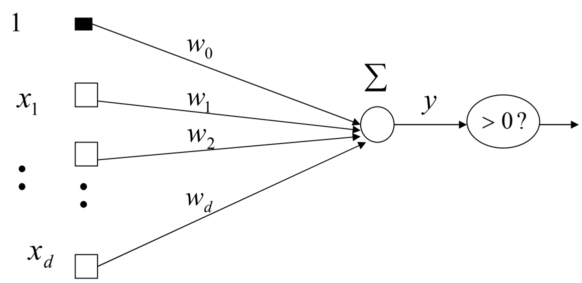
$$\mathbf{w}^T\mathbf{x} + w_0 = 0$$

Class (+1) $\mathbf{w}^T\mathbf{x} + w_0 > 0$
Class (-1) $\mathbf{w}^T\mathbf{x} + w_0 < 0$



Algorithms for linearly separable set

- **Hyperplane** $\mathbf{w}^T\mathbf{x} + w_0 = 0$



- We can use **gradient methods** for sigmoidal switching functions and learn the weights
- Recall that we learn the linear decision boundary

Algorithms for linearly separable sets

- **Linear program solution:**
  - Find weights that satisfy the following constraints:

$$\mathbf{w}^T\mathbf{x}_i + w_0 \geq 0 \quad \text{For all i, such that} \quad y_i = +1$$

$$\mathbf{w}^T\mathbf{x}_i + w_0 \leq 0 \quad \text{For all i, such that} \quad y_i = -1$$

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

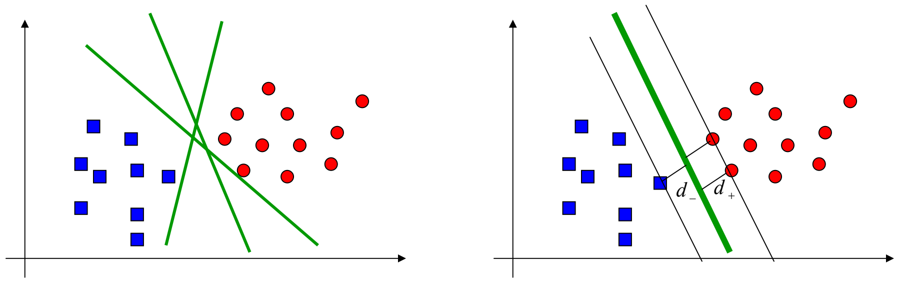
Other methods:

Fisher linear discriminant

Perceptron algorithm

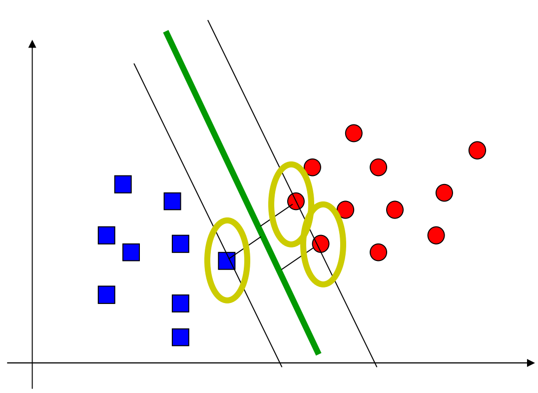
Optimal separating hyperplane

- There are multiple hyperplanes that separate the data points
  - Which one to choose?
- **Maximum margin** choice: the 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 $(\mathbf{x}_i, y_i)$ such that $y_i \in \{+1, -1\}$
- **Assume** that all data satisfy:

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

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

- The inequalities can be combined as:

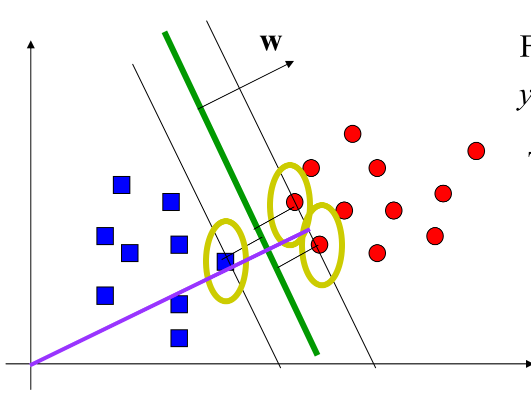
$$y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \geq 0 \quad \text{for all} \quad i$$

- Equalities define two hyperplanes:

$$\mathbf{w}^T \mathbf{x}_i + w_0 = 1 \qquad \mathbf{w}^T \mathbf{x}_i + w_0 = -1$$

Finding the maximum margin hyperplane

- **Geometrical margin:** $\rho_{\mathbf{w},w_0}(\mathbf{x}, y) = y(\mathbf{w}^T \mathbf{x} + w_0) / \|\mathbf{w}\|$
  - measures the distance of a point $\mathbf{x}$ from the hyperplane
    $\mathbf{w}$ - normal to the hyperplane $\|..\|$ - Euclidean norm



For points satisfying:

$$y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1 = 0$$

The distance is $\frac{1}{\|\mathbf{w}\|}$

Width of the margin:

$$d_+ + d_- = \frac{2}{\|\mathbf{w}\|}$$

Maximum margin hyperplane

- **We want to maximize** $d_+ + d_- = \frac{2}{\|\mathbf{w}\|}$
- We do it by **minimizing**

$$\|\mathbf{w}\|^2 / 2 = \mathbf{w}^T \mathbf{w} / 2$$

$\mathbf{w}, w_0$ - variables

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

$$\left[y_i (\mathbf{w}^T \mathbf{x} + w_0) - 1 \right] \geq 0$$

Maximum margin hyperplane

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

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

$\alpha_i \geq 0$ - **Lagrange multipliers**

- **Minimize** with regard to $\mathbf{w}, w_0$ (primal variables)
- **Maximize** with regard to $\boldsymbol{\alpha}$ (dual variables)

Lagrange multipliers enforce the satisfaction of constraints

If $\left[y_i (\mathbf{w}^T \mathbf{x} + w_0) - 1 \right] > 0 \implies \alpha_i \to 0$

Else $\implies \alpha_i > 0$ Active constraint

Max margin hyperplane solution

- Set derivatives to 0 (Kuhn-Tucker conditions)

$$\nabla_{\mathbf{w}} J(\mathbf{w}, w_0, \alpha) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = \bar{0}$$

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

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

$$J(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$ ⟸ **maximize**

Subject to constraints

$\alpha_i \geq 0$ for all *i*, and $\sum_{i=1}^{n} \alpha_i y_i = 0$

- **Quadratic optimization problem:** solution $\hat{\alpha}_i$ for all i

Maximum hyperplane solution

- The resulting parameter vector $\hat{\mathbf{w}}$ 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 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:**

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

- **The decision:**

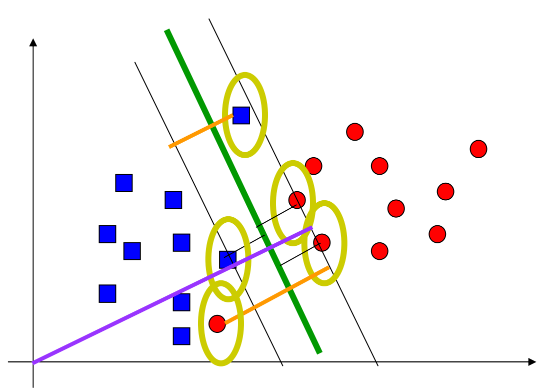
$$\hat{y} = \text{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

Note:

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

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

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

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)

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

- Dual form after $\mathbf{w}, w_0$ are expressed (ξ_i s cancel out)

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

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

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

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:**

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

Note:

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

Nonlinear case

- The linear case requires to compute $(\mathbf{x}_i^T \mathbf{x})$
- The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors

$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$$

- It is possible to use SVM formalism on feature vectors

$$\boldsymbol{\varphi}(\mathbf{x})^T \boldsymbol{\varphi}(\mathbf{x}')$$

- **Kernel function**

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

- **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

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

- Kernel function for the feature space:

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

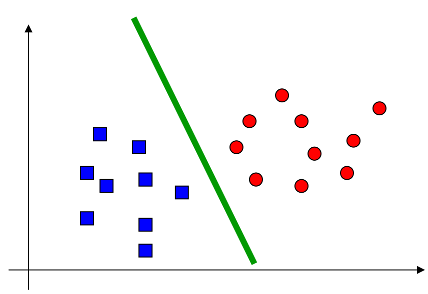
$$= x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_2 x_1' x_2' + 2x_1 x_1' + 2x_2 x_2' + 1$$

$$= (x_1 x_1' + x_2 x_2' + 1)^2$$

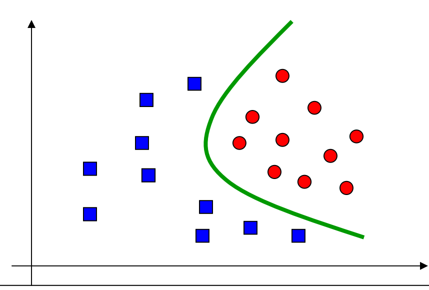
$$= (1 + (\mathbf{x}^T \mathbf{x}'))^2$$

- 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**

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

- **Polynomial kernel**

$$K(\mathbf{x}, \mathbf{x}') = \left[1 + \mathbf{x}^T \mathbf{x}'\right]^k$$

- **Radial basis kernel**

$$K(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{1}{2}\left\|\mathbf{x} - \mathbf{x}'\right\|^2\right]$$