CS 2750 Machine Learning
Lecture 12

Support vector machines II

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

Linearly separable classes

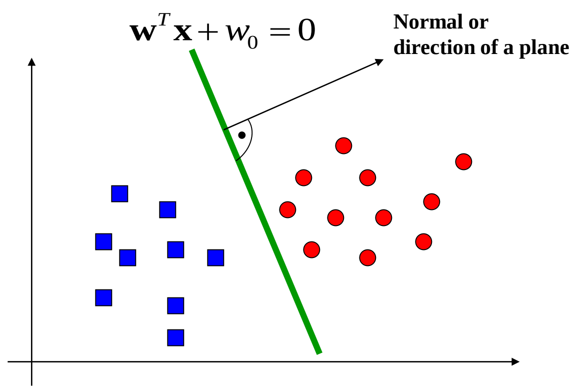
Linearly separable classes:
There is a **hyperplane** $\mathbf{w}^T\mathbf{x} + w_0 = 0$
that separates training instances with no error

Class (+1)

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

Class (-1)

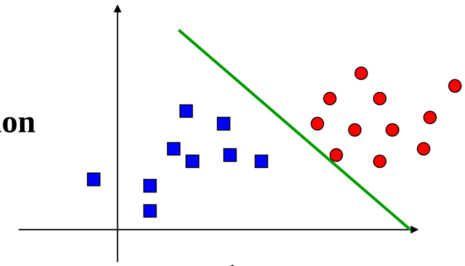
$\mathbf{w}^T\mathbf{x} + w_0 < 0$



Learning linearly separable sets

Finding weights for linearly separable classes:

- **Linear program (LP) solution**
- It finds weights that satisfy the following constraints:



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

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

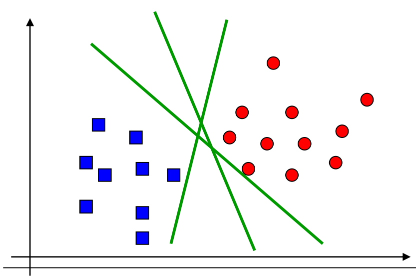
Together: $y_i (\mathbf{w}^T \mathbf{x}_i + w_0) \geq 0$

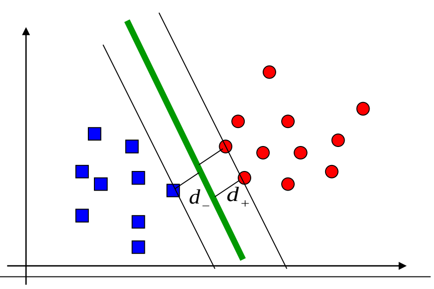
Property: if there is a hyperplane separating the examples, the linear program finds the solution

Optimal separating hyperplane

- **Problem:** multiple hyperplanes that separate the data exists
  - 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)

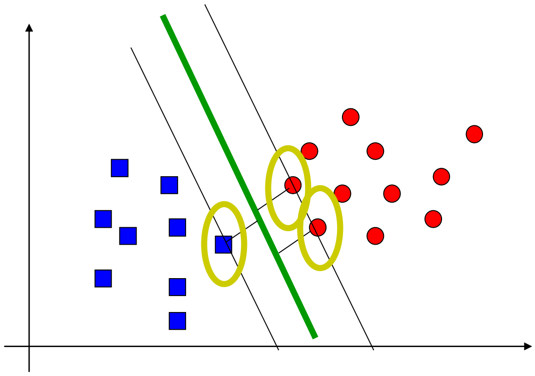
Note: a margin classifier is a classifier for which we can calculate the distance of each example from the decision boundary





Maximum margin hyperplane

- For the maximum margin hyperplane only examples on the margin matter (only these affect the distances)
- These are called **support vectors**



Maximum margin hyperplane

- **We want to maximize** $d_+ + d_- = \frac{2}{\|\mathbf{w}\|_{L2}}$
- We do it by **minimizing**

$$\|\mathbf{w}\|_{L2}^2 / 2 = \mathbf{w}^T \mathbf{w} / 2$$

$\mathbf{w}, w_0$ - variables

- But we also need to enforce the constraints on all data instances: $(\mathbf{x}_i, y_i)$

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

Maximum margin hyperplane

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

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

Data instances $(\mathbf{x}_i, y_i)$

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

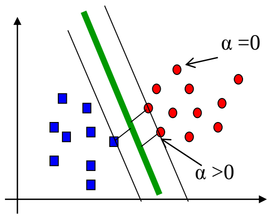
- **Minimize** with respect to $\mathbf{w}, w_0$ (primal variables)
- **Maximize** with respect to $\boldsymbol{\alpha}$ (dual variables)

What happens to α:

if $y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 > 0 \Longrightarrow \alpha_i \to 0$

else $\Longrightarrow \alpha_i > 0$

Active constraint



Max margin hyperplane solution

- Set derivatives to 0 (Kuhn-Tucker conditions)

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

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

- 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 margin solution

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

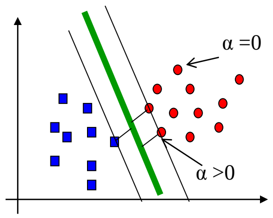
$$\hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$$

$\hat{\alpha}_i$ is the solution of the optimization

- The parameter w_0 is obtained from $\hat{\alpha}_i \left[y_i (\hat{\mathbf{w}}\mathbf{x}_i + w_0) - 1 \right] = 0$

Solution properties

- $\hat{\alpha}_i = 0$ for all points that are not on the margin
- **The decision boundary:**



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

The decision boundary defined by support vectors only

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**

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

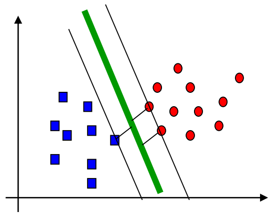
Lagrange multipliers

- **Classification decision for new x:**

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

- **Note that we do not have to explicitly compute** $\hat{\mathbf{w}}$
  - This will be important for the nonlinear (kernel) case

Support vector machines



- **The decision boundary:**

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

- **Classification decision:**

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

Support vector machines: inner product

- Decision on a new **x** depends on the **inner product between two examples**
- **The decision boundary:**

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

- **Classification decision:**

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

- Similarly, the optimization depends on $(\mathbf{x}_i{}^T \mathbf{x}_j)$

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

Inner product of two vectors

- **The decision boundary for the SVM and its optimization depend on the inner product of two datapoints (vectors):**

$$(\mathbf{x}_i^T \mathbf{x}_j)$$

$$\mathbf{x}_i = \begin{pmatrix} 2 \\ 5 \\ 6 \end{pmatrix} \qquad \mathbf{x}_j = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$$

$$(\mathbf{x}_i^T \mathbf{x}_j) = \quad \textbf{?}$$

Inner product of two vectors

- **The decision boundary for the SVM and its optimization depend on the inner product of two data points (vectors):**

$$(\mathbf{x}_i^T \mathbf{x}_j)$$

$$\mathbf{x}_i = \begin{pmatrix} 2 \\ 5 \\ 6 \end{pmatrix} \qquad \mathbf{x}_j = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$$

$$(\mathbf{x}_i^T \mathbf{x}_j) = \begin{pmatrix} 2 & 5 & 6 \end{pmatrix} * \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} = 2*2 + 5*3 + 6*1 = 25$$

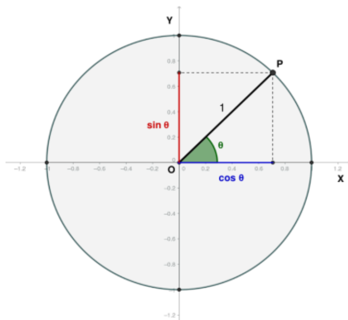
Inner product of two vectors

- **The decision boundary for the SVM and its optimization depend on the inner product of two data points (vectors):**

$$(\mathbf{x}_i^T \mathbf{x}_j)$$

- **The inner product is equal**

$$(\mathbf{x}_i^T \mathbf{x}_j) = \|\mathbf{x}_i\| * \|\mathbf{x}_j\| \cos\theta$$



If the angle in between them is 0 then:

$$(\mathbf{x}_i^T \mathbf{x}_j) = \|\mathbf{x}_i\| * \|\mathbf{x}_j\|$$

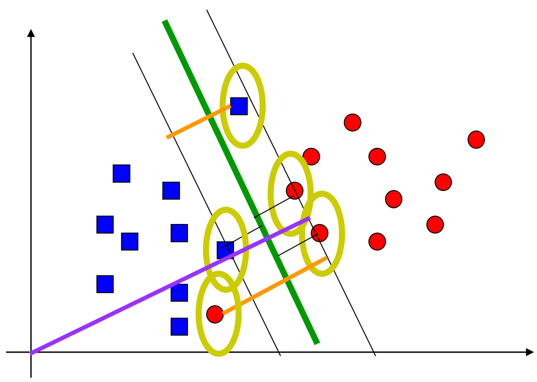
If the angle between them is 90 then:

$$(\mathbf{x}_i^T \mathbf{x}_j) = 0$$

The inner product measures how similar the two vectors are

Extension to a linearly non-separable case

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



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** (**soft margin**)

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

Linearly non-separable case

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

$$\mathbf{w}^T \mathbf{x}_i + w_0 \geq 1 - \xi_i \quad \text{for} \quad y_i = +1$$
$$\mathbf{w}^T \mathbf{x}_i + w_0 \leq -1 + \xi_i \quad \text{for} \quad y_i = -1$$
$$\xi_i \geq 0$$

- Rewrite $\xi_i = \max\left[0, \quad 1 - y_i(\mathbf{w}^T \mathbf{x}_i + w_0)\right]$ in $\|\mathbf{w}\|^2 / 2 + C \sum_{i=1}^{n} \xi_i$

$$\|\mathbf{w}\|^2 / 2 + C \sum_{i=1}^{n} \max\left[0, \quad 1 - y_i(\mathbf{w}^T \mathbf{x}_i + w_0)\right]$$

Regularization penalty

Hinge loss

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: $0 \le \alpha_i \le C$ 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: $0 \le \alpha_i \le C$

The parameter w_0 is obtained through KKT conditions

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

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

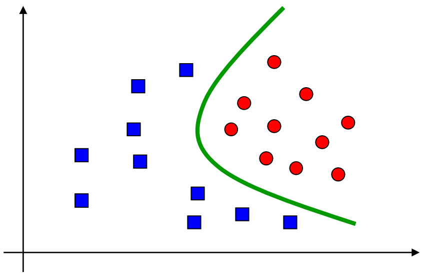
$$\hat{y} = \text{sign}\left[\hat{\mathbf{w}}^T\mathbf{x} + w_0\right] = \text{sign}\left[\sum_{i \in SV}\hat{\alpha}_i y_i (\mathbf{x}_i^T\mathbf{x}) + w_0\right]$$

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

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 non-linear decision boundaries with the SVM?**



Nonlinear decision boundary

- The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors

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

  - Note that feature expansions are typically high dimensional
    - Examples: polynomial expansions
- Given the nonlinear feature mappings, we can use the linear SVM on the expanded feature vectors

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

- **Kernel function**

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

Support vector machines: solution for nonlinear decision boundaries

- **The decision boundary:**

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

- **Classification:**

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

- Decision on a new $\mathbf{x}$ requires to compute **the kernel function defining the similarity between the examples**
- Similarly, the optimization depends on the kernel

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

Kernel trick

The non-linear case maps input vectors to (larger) feature space

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

- Note that feature expansions are typically high dimensional
  - Examples: polynomial expansions
- **Kernel function** defines the inner product in the expanded high dimensional feature vectors and let us use the SVM

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

- **Problem:** after expansion we need to perform inner products in a very high dimensional space
- **Kernel trick:**
  - If we choose the kernel function 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

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

- Kernel function for the feature space:

Kernel function example

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

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

- Kernel function for the feature space:

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

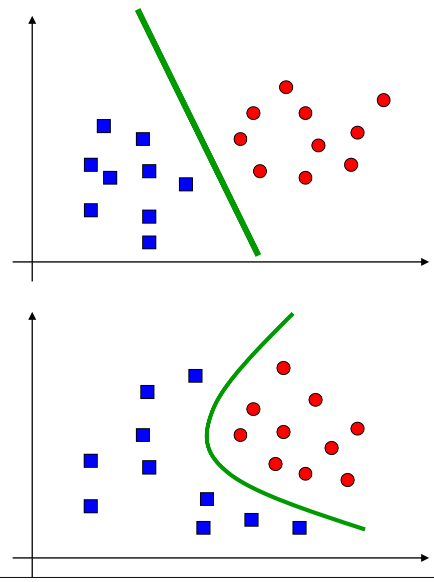
$$= x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1x_2x'_1\, x'_2 + 2x_1x'_1 + 2x_2x'_2 + 1$$

$$= (x_1x'_1 + x_2x'_2 + 1)^2$$

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

- The computation of the linear separation in the higher dimensional space is performed implicitly in the original input space

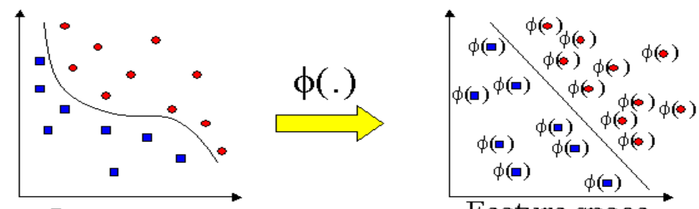
Kernel function example



Linear separator in the expanded feature space

Non-linear separator in the input space

Nonlinear extension



Kernel trick

- Replace the inner product with a kernel
- A well chosen kernel leads to an efficient computation

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**

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

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