CS 2750 Machine Learning
Lecture 9

Support vector machines

Milos Hauskrecht

milos@cs.pitt.edu

5329 Sennott Square

Outline

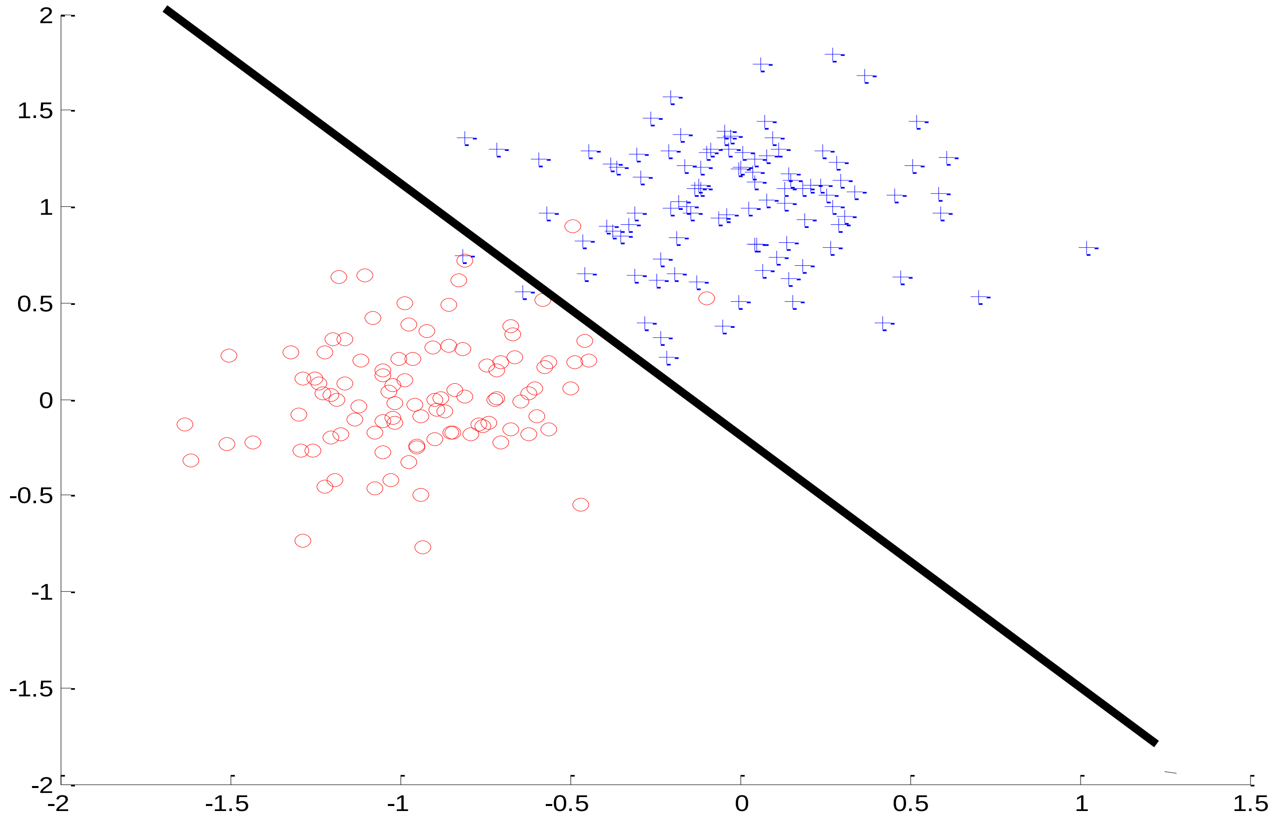
Outline:

- 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?



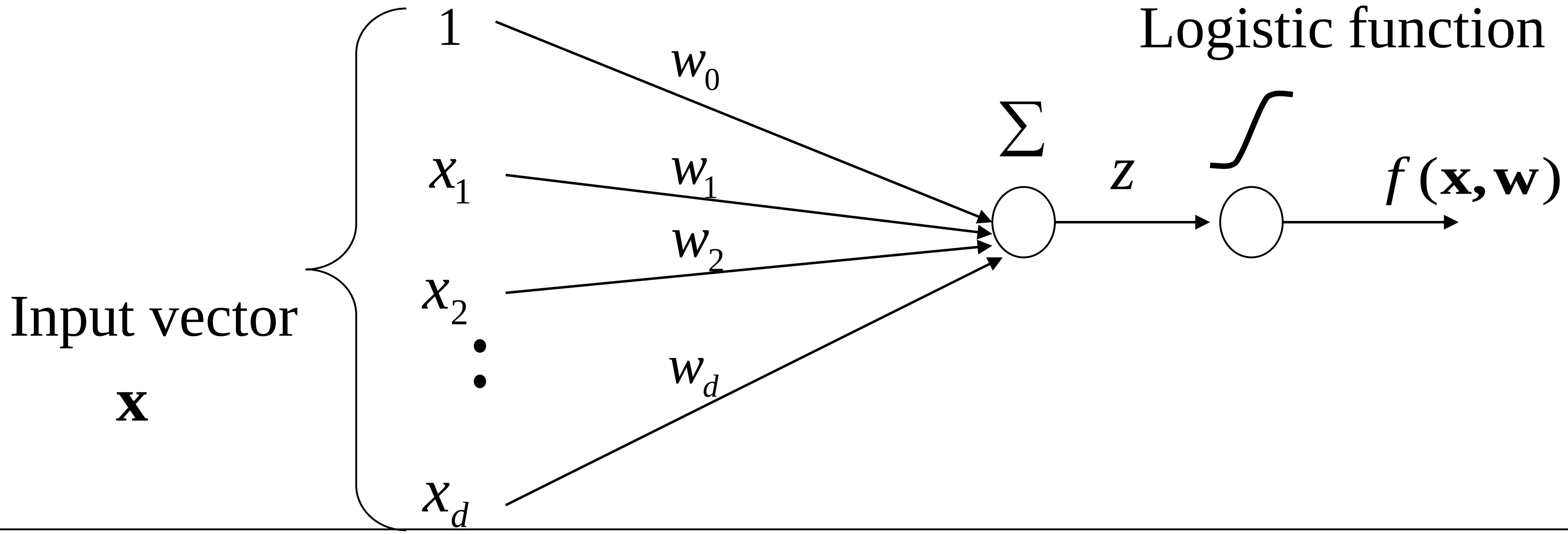
Logistic regression model

- **Discriminant functions:**

$$g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x}) \qquad\qquad g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T\mathbf{x})$$

- **where** $g(z) = 1/(1+e^{-z})$ - is a logistic function

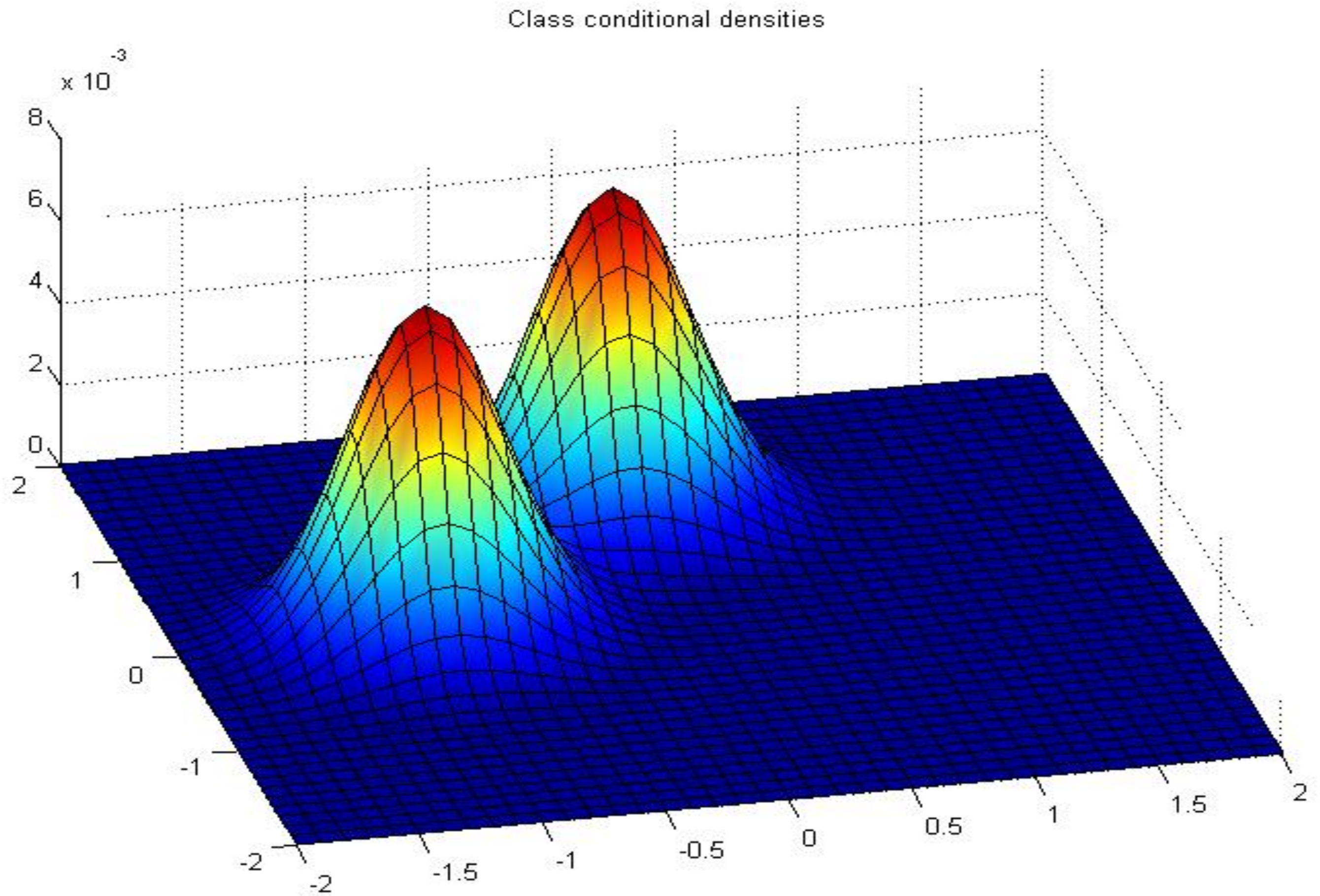
$$f(\mathbf{x},\mathbf{w}) = g_1(\mathbf{w}^T\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$$



Linear discriminant analysis (LDA)

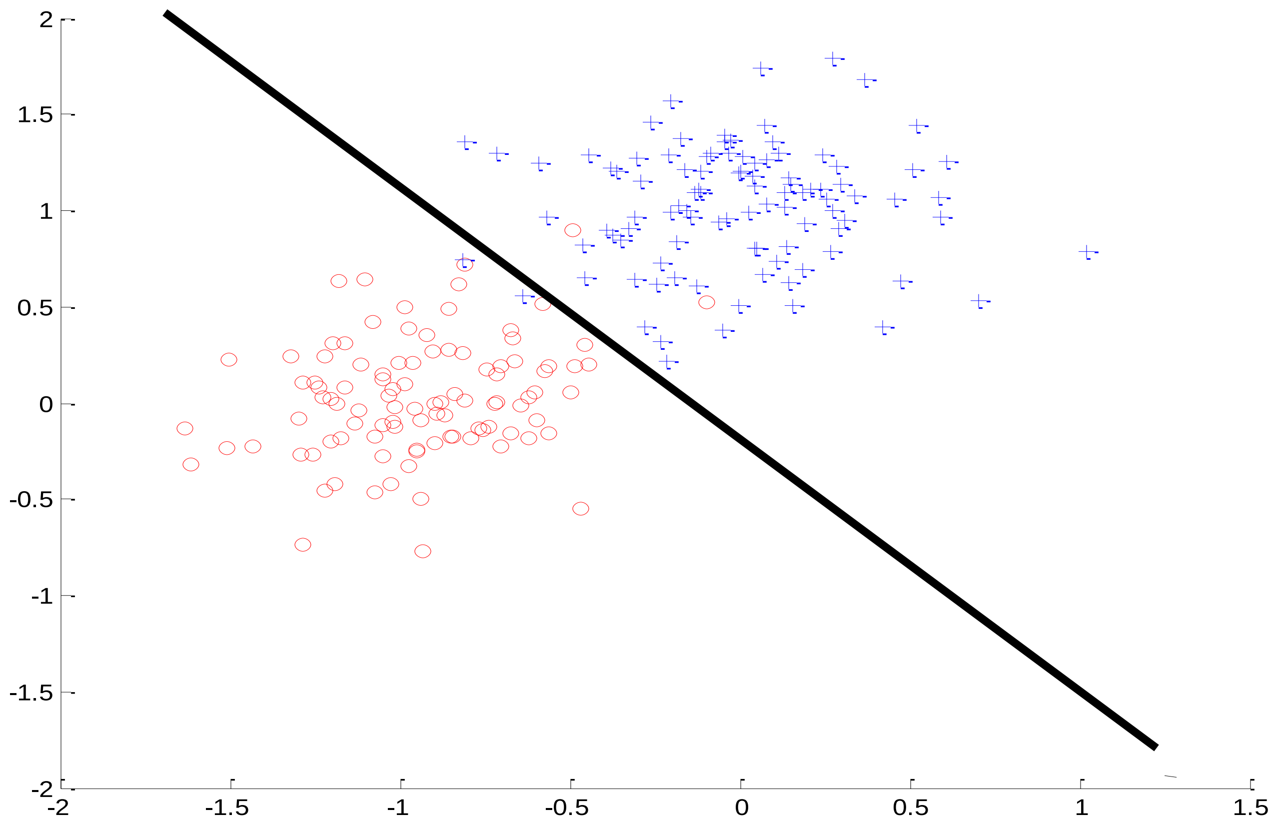
- When covariances are the same

$$\mathbf{x} \sim N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}),\ y = 0$$
$$\mathbf{x} \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}),\ y = 1$$



Linear decision boundaries

- Any other models/algorithms?

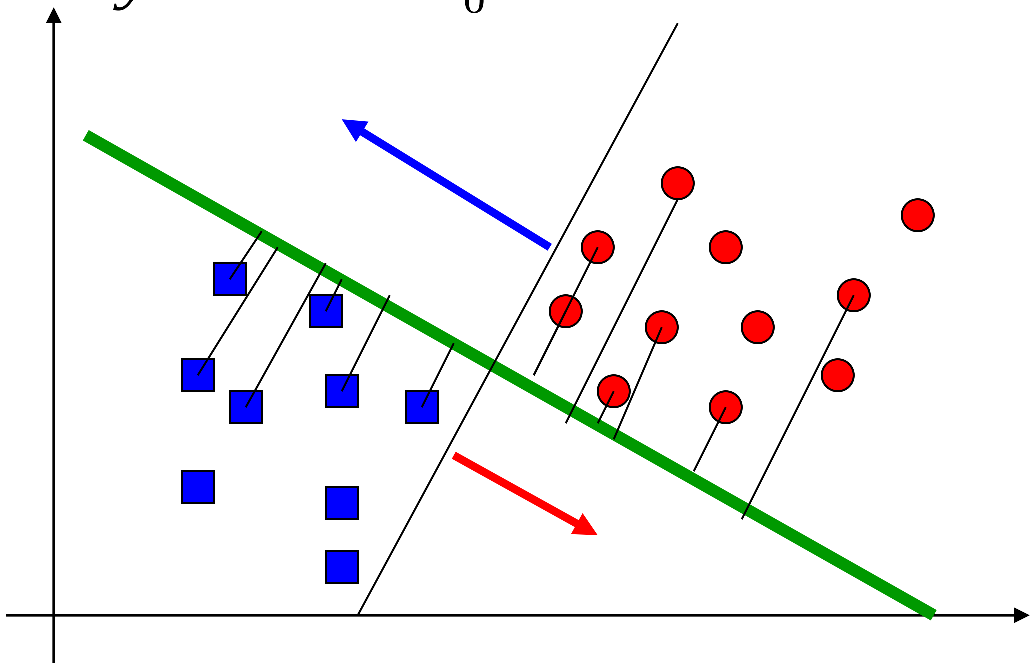


Fisher linear discriminant

- Project data into one dimension

$$y = \mathbf{w}^T \mathbf{x}$$

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

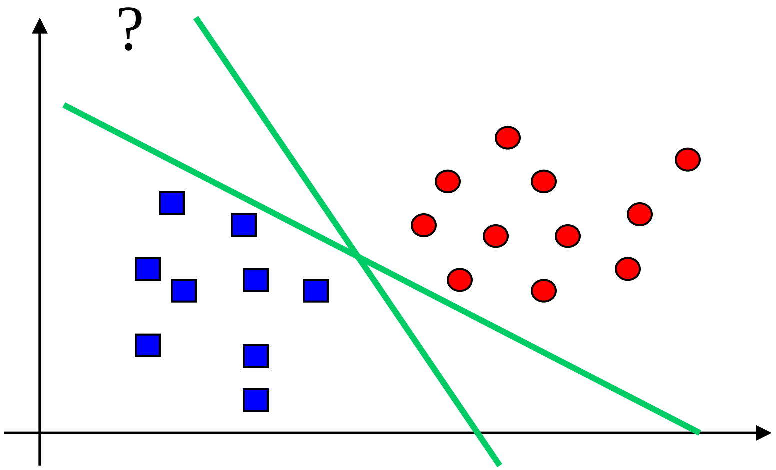


- How to find the projection line?

Fisher linear discriminant

How to find the projection line?

$$y = \mathbf{w}^T \mathbf{x}$$



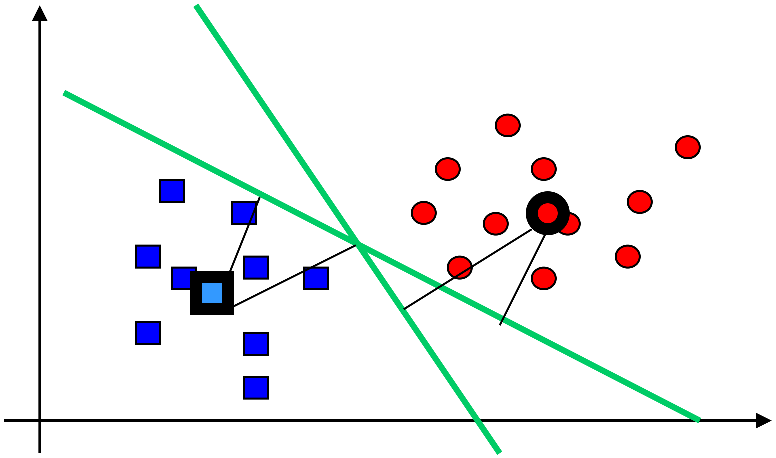
Fisher linear discriminant

Assume:

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{i \in C_1}^{N_1} \mathbf{x}_i \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{i \in C_2}^{N_2} \mathbf{x}_i$$

Maximize the difference in projected means:

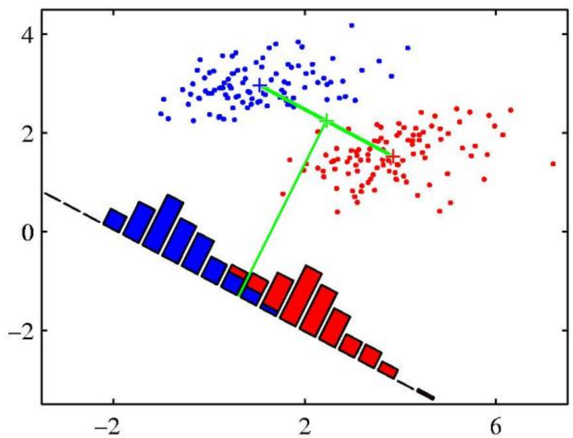
$$m_2 - m_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$$

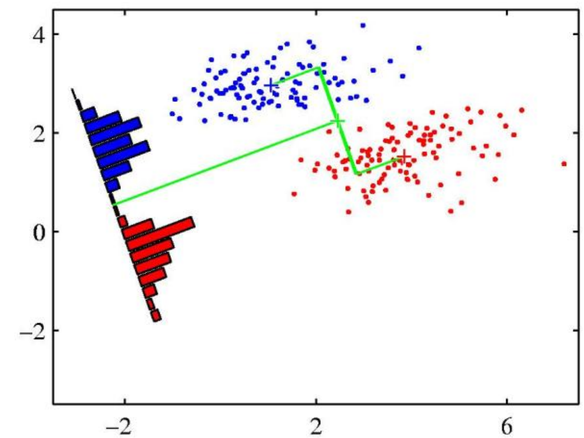


Fisher linear discriminant

Problem 1: $m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$ can be maximized by increasing $\mathbf{w}$

Problem 2: variance in class distributions after projection is changed





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

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

Fisher linear discriminant

Error:

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

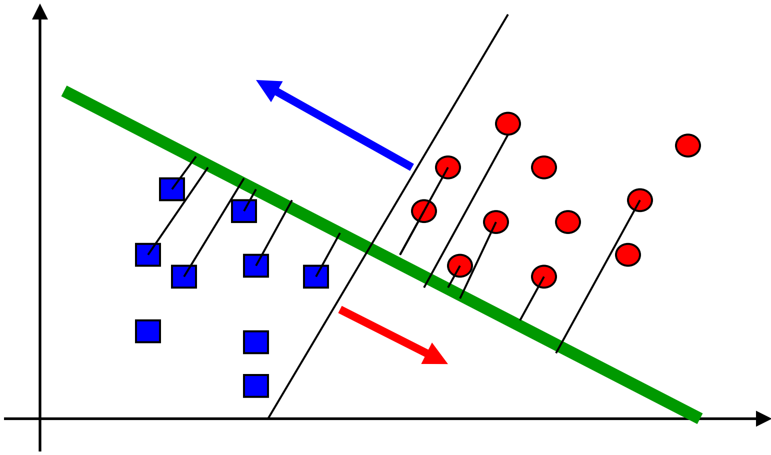
Within class variance after the projection

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

Optimal solution:

$$\mathbf{w} \approx \mathbf{S}_{\mathbf{w}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

$$\mathbf{S}_{\mathbf{w}} = \sum_{i \in C_1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^T + \sum_{i \in C_2} (\mathbf{x}_i - \mathbf{m}_2)(\mathbf{x}_i - \mathbf{m}_2)^T$$



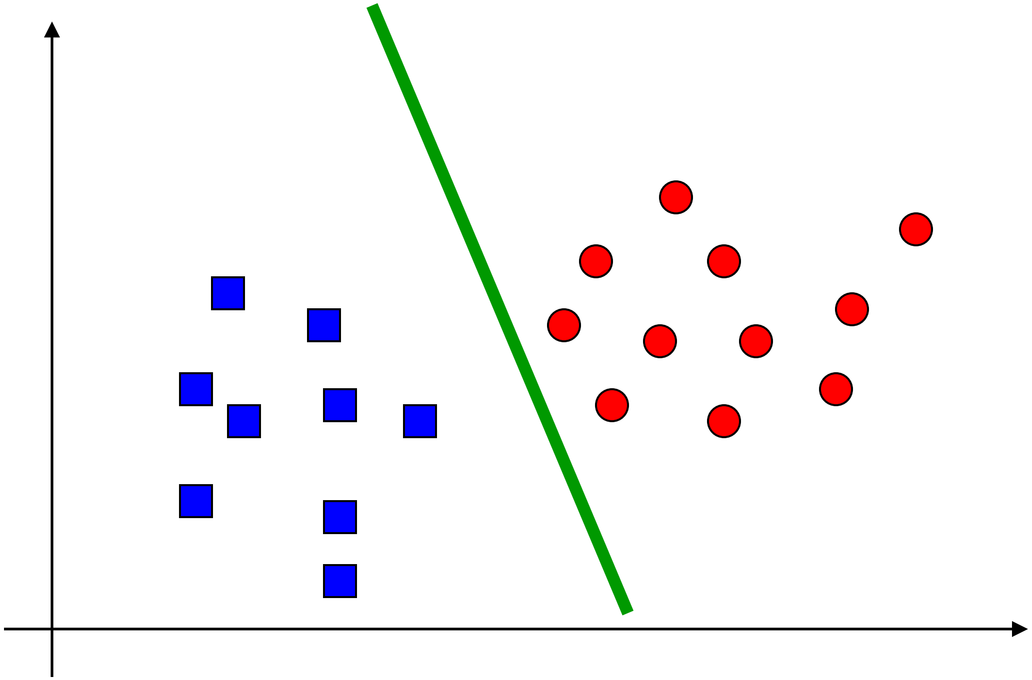
Linearly separable classes

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

Hyperplane:

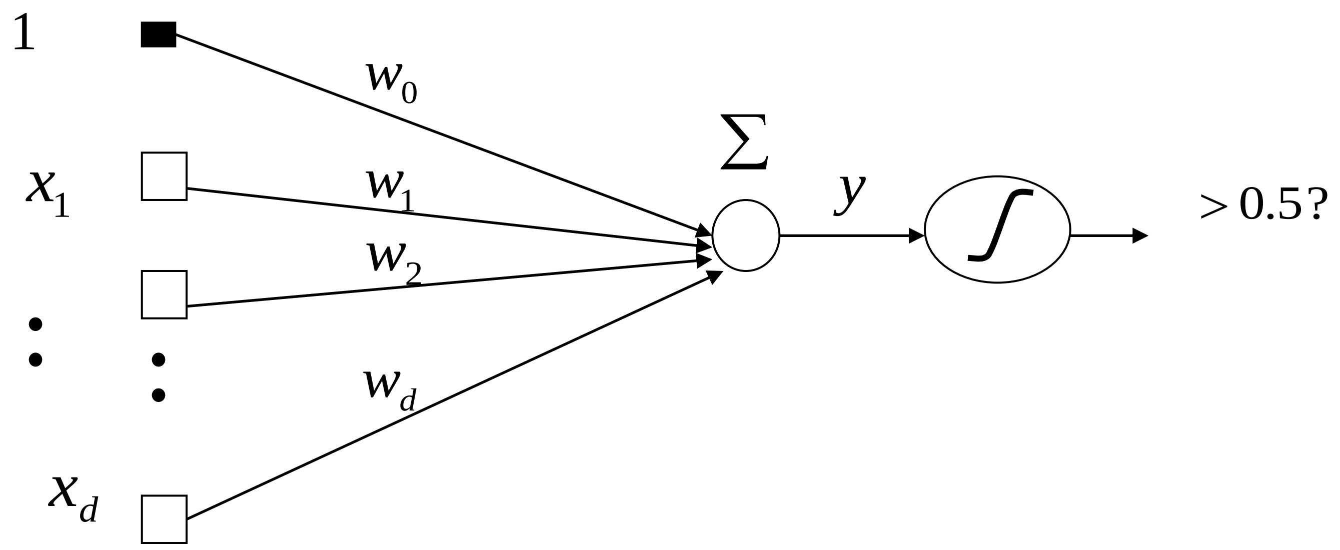
$$\mathbf{w}^T \mathbf{x} + w_0 = 0$$

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

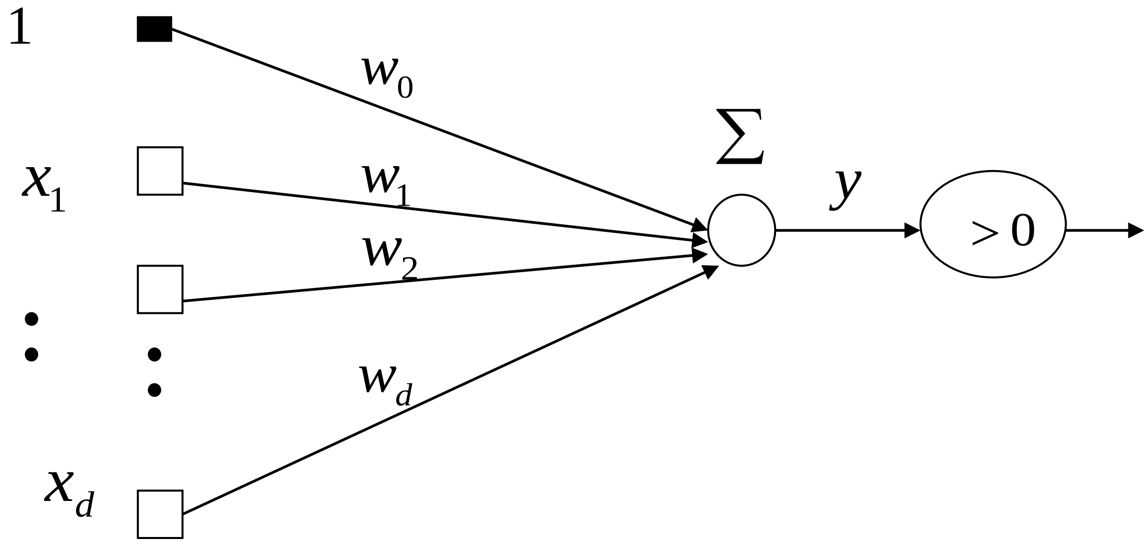
- **Separating hyperplane** $\mathbf{w}^T \mathbf{x} + w_0 = 0$



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

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 $\mathbf{x}$ where each x_i is in [0,1]

Initialize weights $\mathbf{w}$

Loop through examples ($\mathbf{x}$, y) in the dataset D

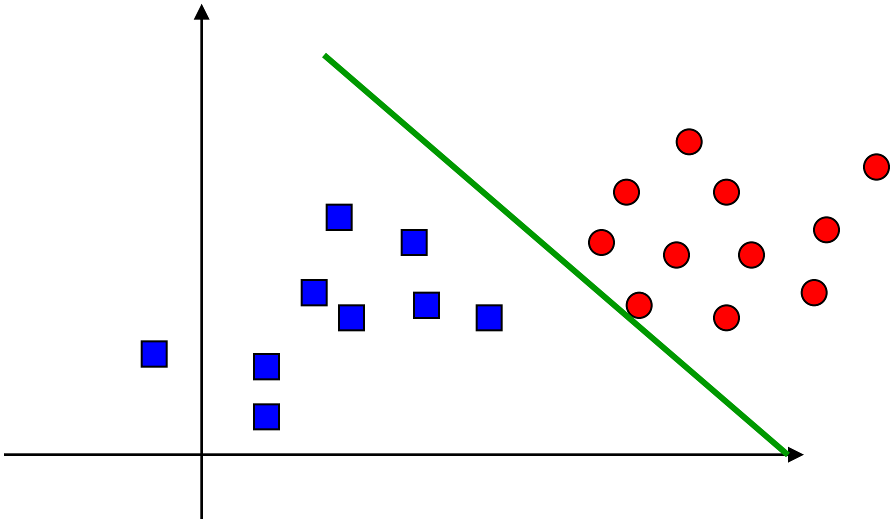
1. Compute $\hat{y} = \mathbf{w}^T \mathbf{x}$
2. If $y \neq \hat{y} = -1$ then $\mathbf{w}^T \leftarrow \mathbf{w}^T + \mathbf{x}$
3. If $y \neq \hat{y} = +1$ then $\mathbf{w}^T \leftarrow \mathbf{w}^T - \mathbf{x}$

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:

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

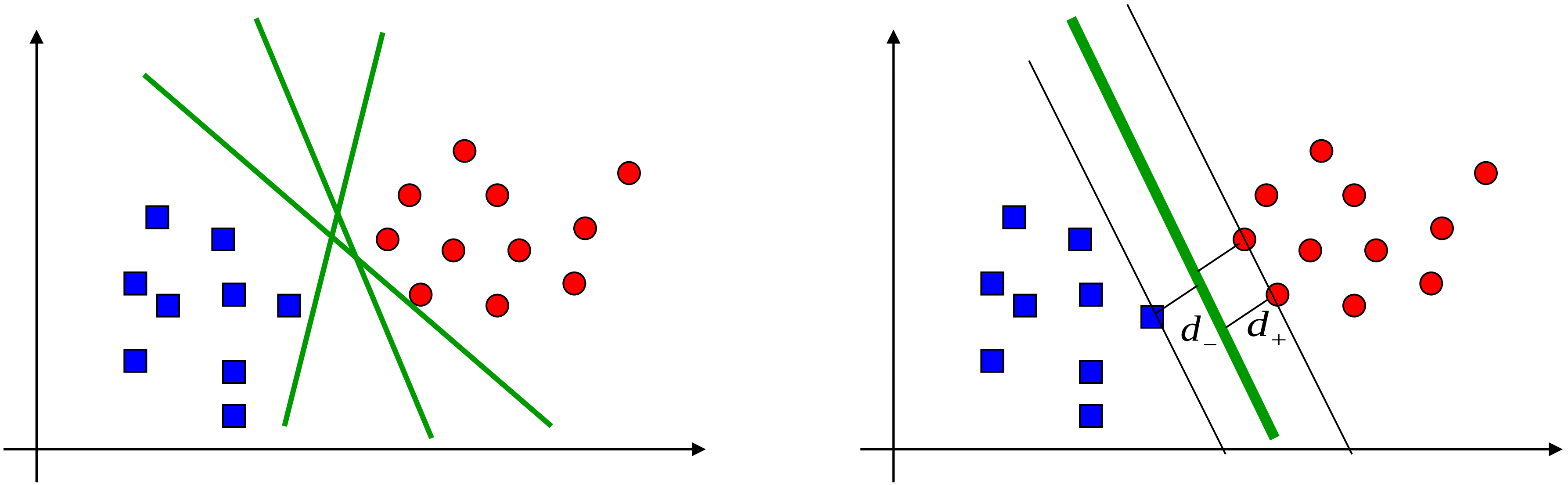
$$\mathbf{w}^T \mathbf{x}_i + w_0 \leq 0 \qquad \text{For all i, such that} \quad y_i = -1$$

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

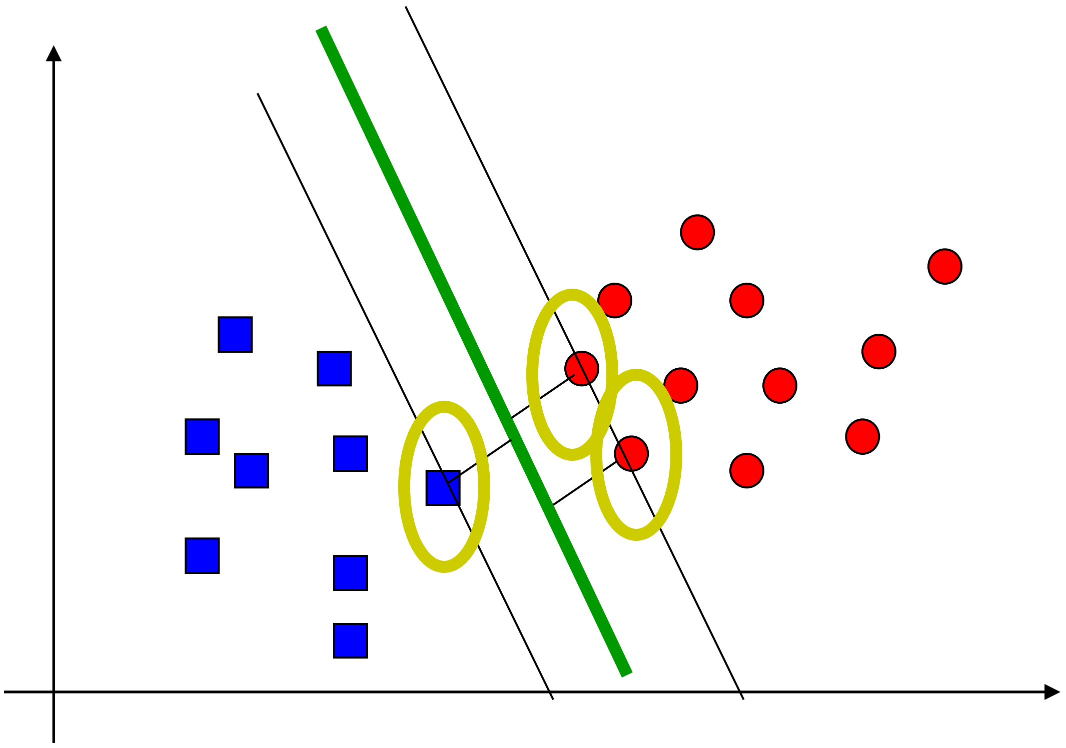
Optimal separating hyperplane

- 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 $(\mathbf{x}_i, y_i)$ 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:

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

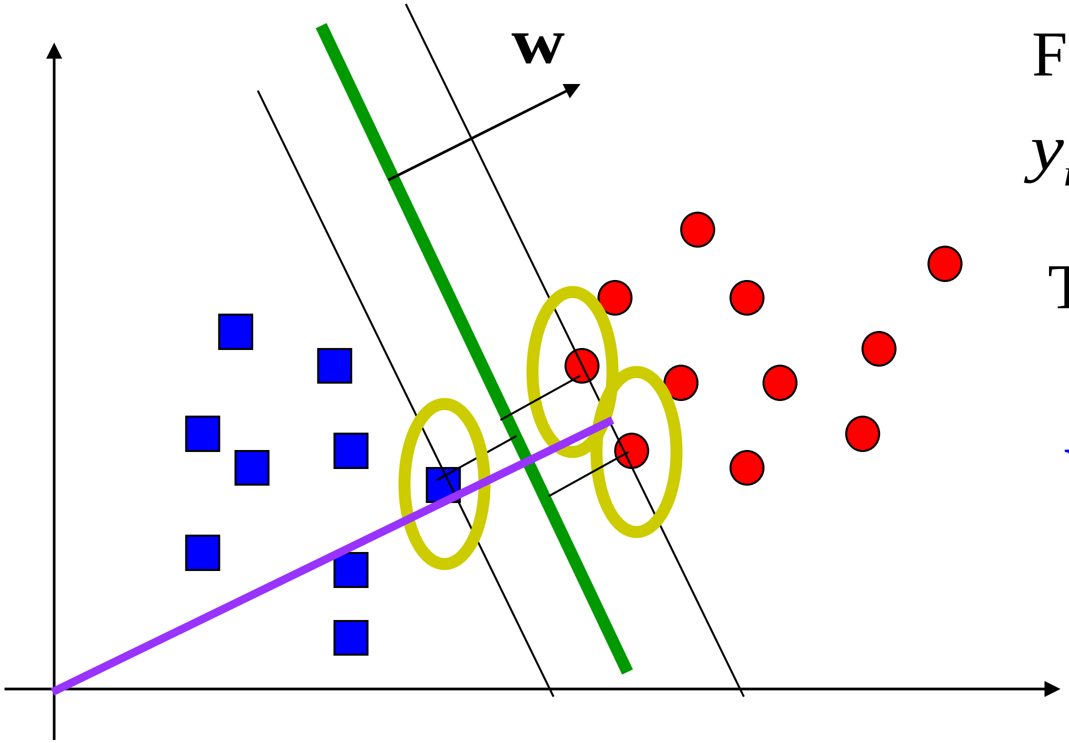
- Equalities define two hyperplanes:

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

Finding the maximum margin hyperplane

- **Geometrical margin:** $\rho_{\mathbf{w},w_0}(\mathbf{x}, y) = y(\mathbf{w}^T\mathbf{x} + w_0) / \|\mathbf{w}\|_{L2}$
  - measures the distance of a point **x** from the hyperplane

    $\mathbf{w}$ - normal to the hyperplane $\|..\|_{L2}$ - Euclidean norm



For points satisfying:

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

The distance is $\dfrac{1}{\|\mathbf{w}\|_{L2}}$

Width of the margin:

$$d_+ + d_- = \frac{2}{\|\mathbf{w}\|_{L2}}$$

Maximum margin hyperplane

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

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

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

Solution properties

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

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

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:**

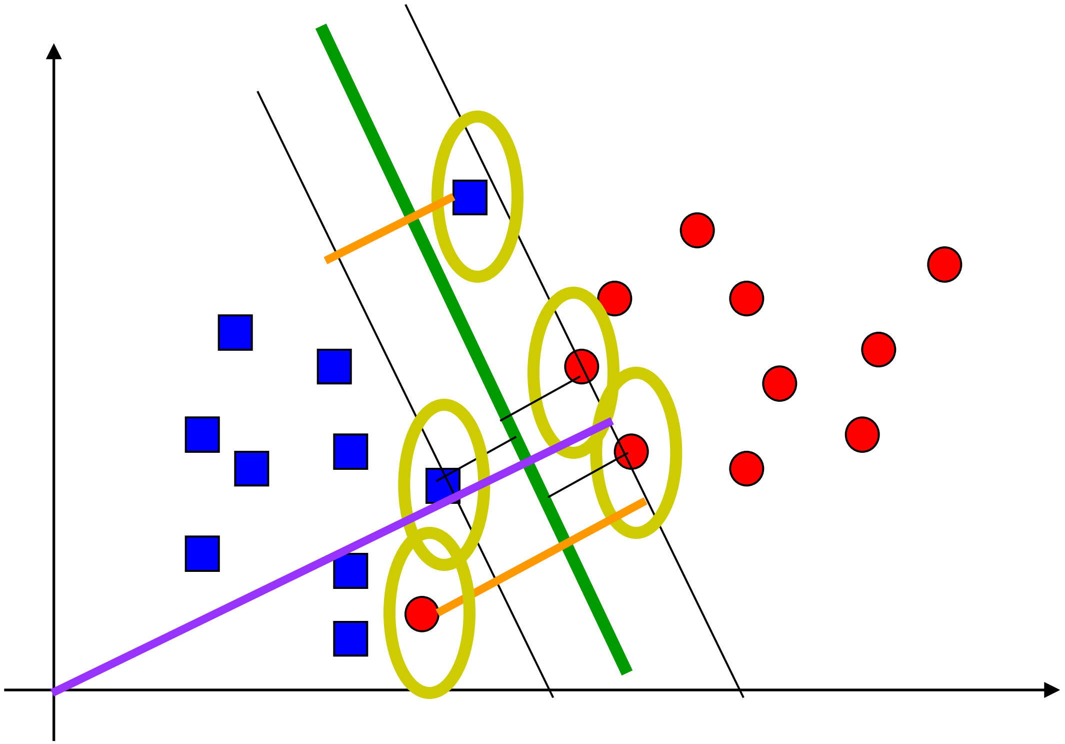
$$\hat{y} = \operatorname{sign}\left[\sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i{}^T \mathbf{x}) + w_0 \right]$$

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

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

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

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)

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

- 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

- **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:**

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

- **(!!):**
- Decision on a new **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 $(\mathbf{x}_i^T \mathbf{x})$
- 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

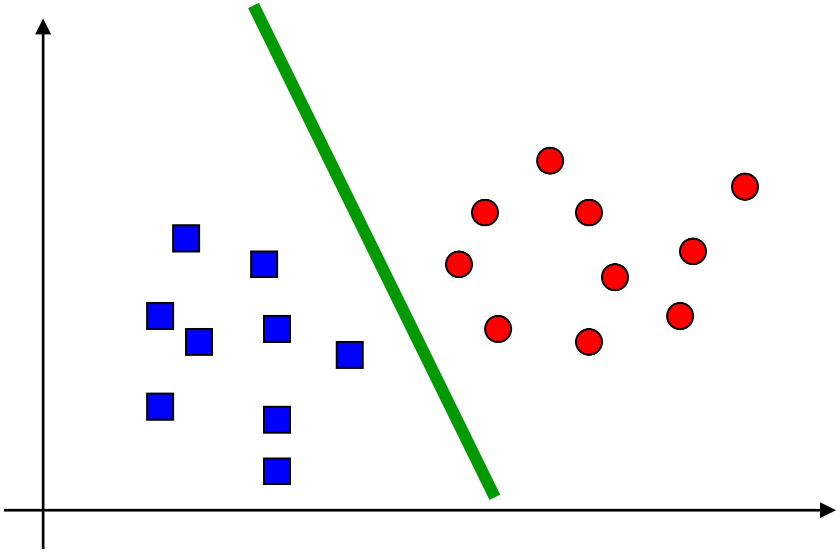
$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T$$

- Kernel function for the feature space:

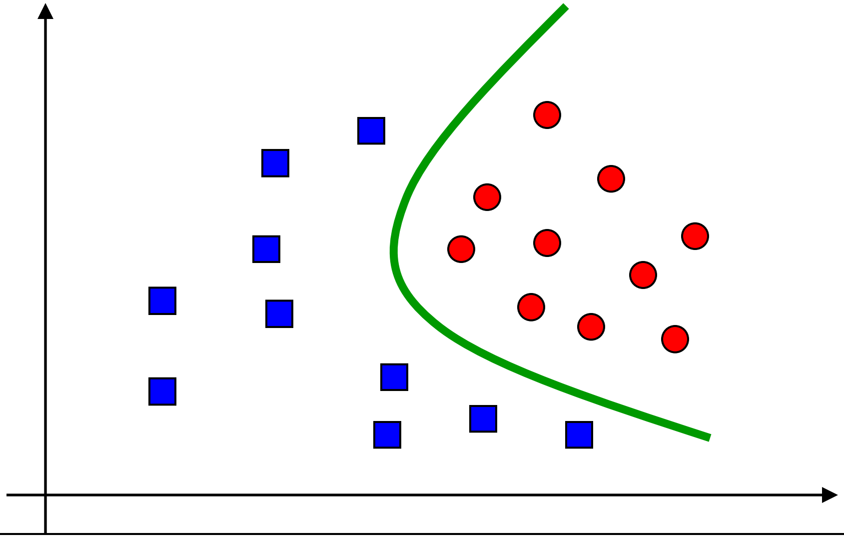
$$\begin{aligned} K(\mathbf{x}', \mathbf{x}) &= \boldsymbol{\varphi}(\mathbf{x}')^T \boldsymbol{\varphi}(\mathbf{x}) \\ &= x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1x_2x'_1\, x'_2 + 2x_1x'_1 + 2x_2x'_2 + 1 \\ &= (x_1x'_1 + x_2x'_2 + 1)^2 \\ &= (1 + (\mathbf{x}^T\mathbf{x}'))^2 \end{aligned}$$

- 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**

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

Kernels

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