Artificial Intelligence

Lecture 13: Support Vector Machines

Credit: Ansaf Salleb-Aouissi, and “Artificial Intelligence: A Modern Approach”, Stuart Russell and Peter Norvig, and “The Elements of Statistical Learning”, Trevor Hastie, Robert Tibshirani, and Jerome Friedman, and “Machine Learning”, Tom Mitchell.

Support Vector Machines (SVMs)

- Refer to a supervised learning algorithm that builds mainly on three ideas:
  - **large margin classification**
  - **regularization** (for data not linearly separable)
  - **feature transformation** and **kernels** (to go beyond linear classifiers)
- classification performance is often very good
- Given:
  - **training set** $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
  - $\mathbf{x}_i \in \mathbb{R}^d$: **input**
  - $y_i \in \{-1, +1\}$: **output (label)**

Outline

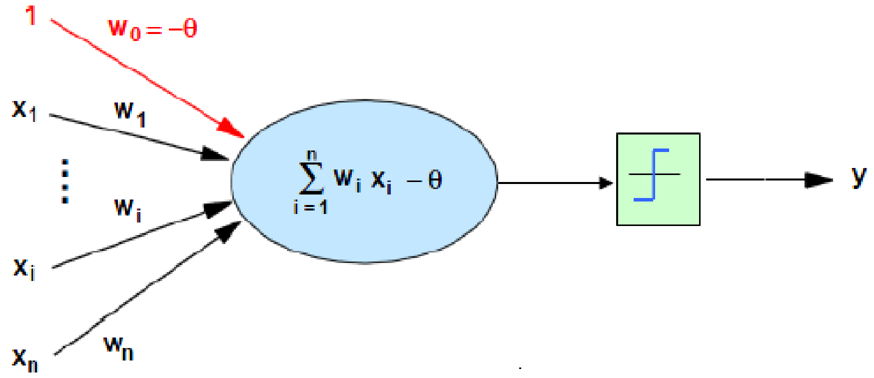
- Linear SVMs
- Data Not Linearly Separable/Regularization
- Non-Linearly SVMs/Kernels

Linear SVMs

Linear Models

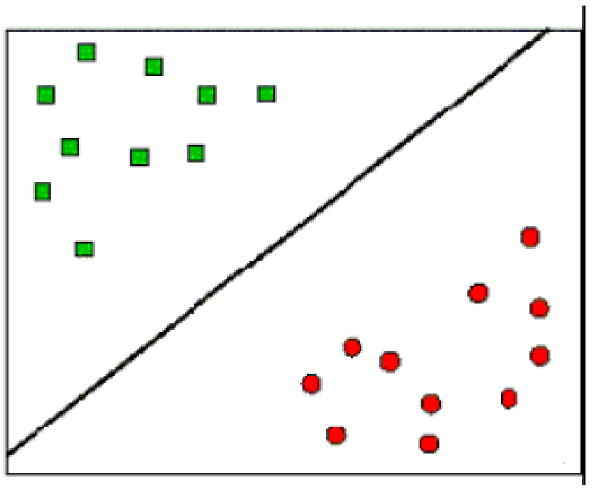
- $\mathbf{w} = (w_1, \ldots, w_d)$, $\mathbf{x} = (x_1, \ldots, x_d)$ and $b = -\theta$

$$y = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$



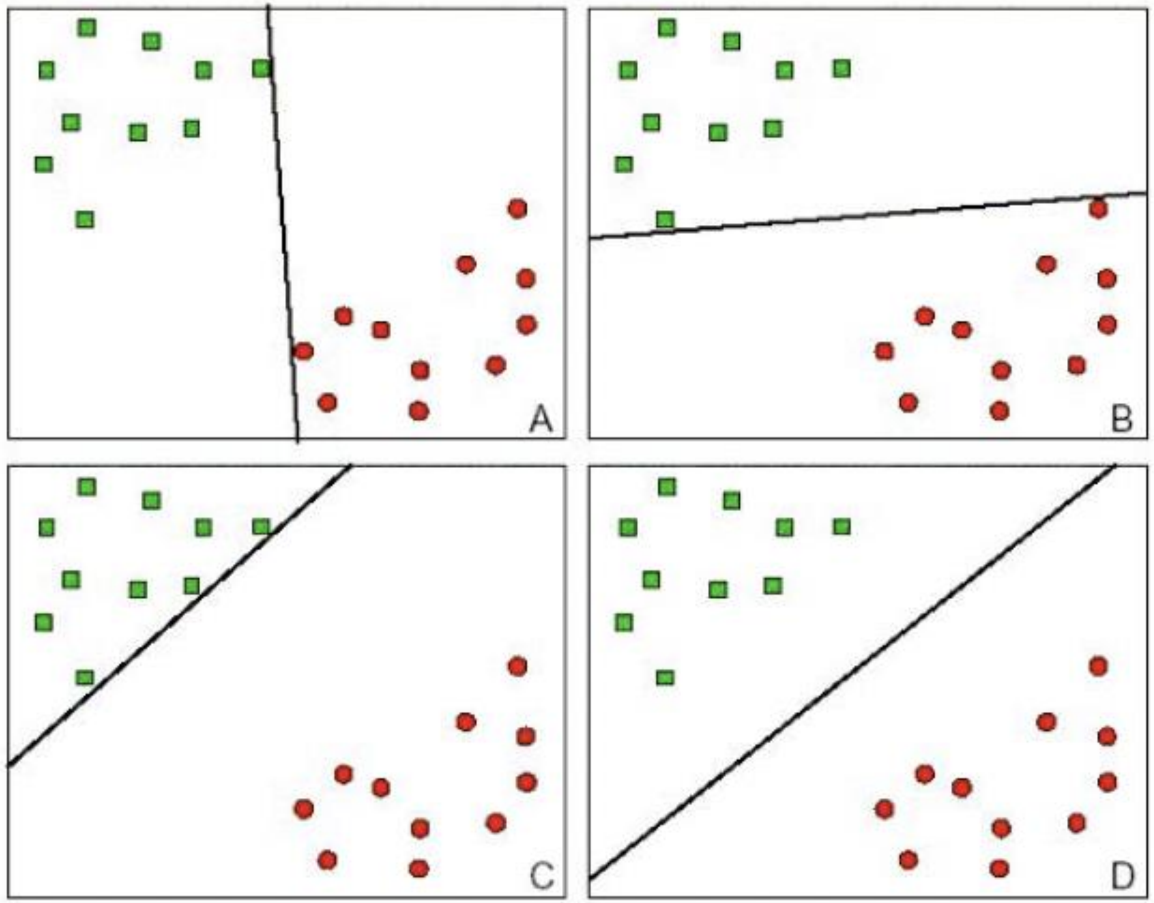
- the decision boundary is the hyperplane

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = 0$$



- decision rule: assign $\mathbf{x}$ to class 1 iff $f(\mathbf{x}) \geqslant 0$

Multiple Solutions

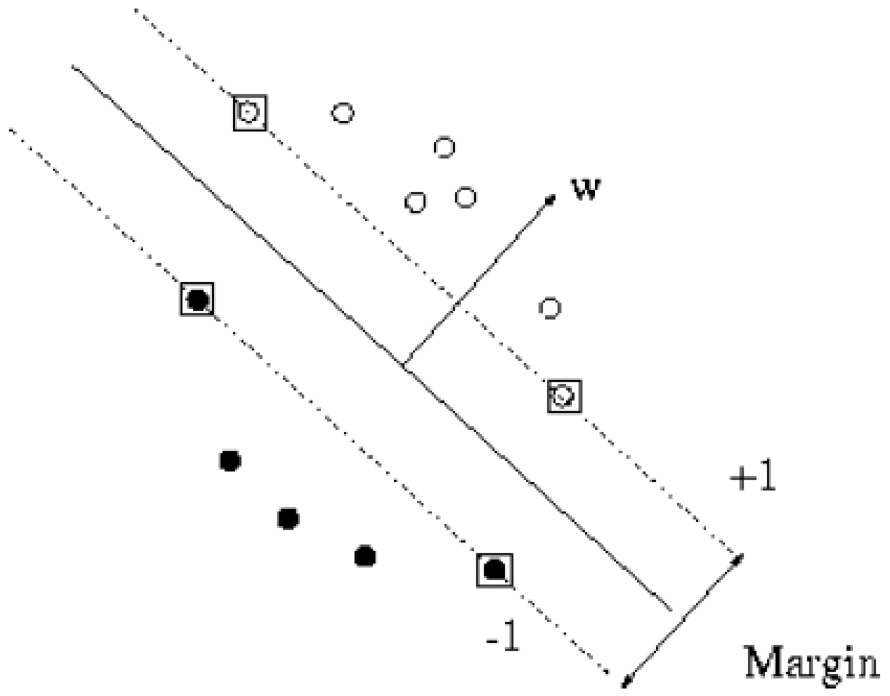


Which decision boundary to choose?

Optimal Margin Classifier

find classifier with the **maximum margin**

- the **minimum** distance between a data point to the decision boundary is maximized
- intuitively, the safest and most robust
- called **linear support vector machines**
- **support vectors**: datapoints the margin pushes up against



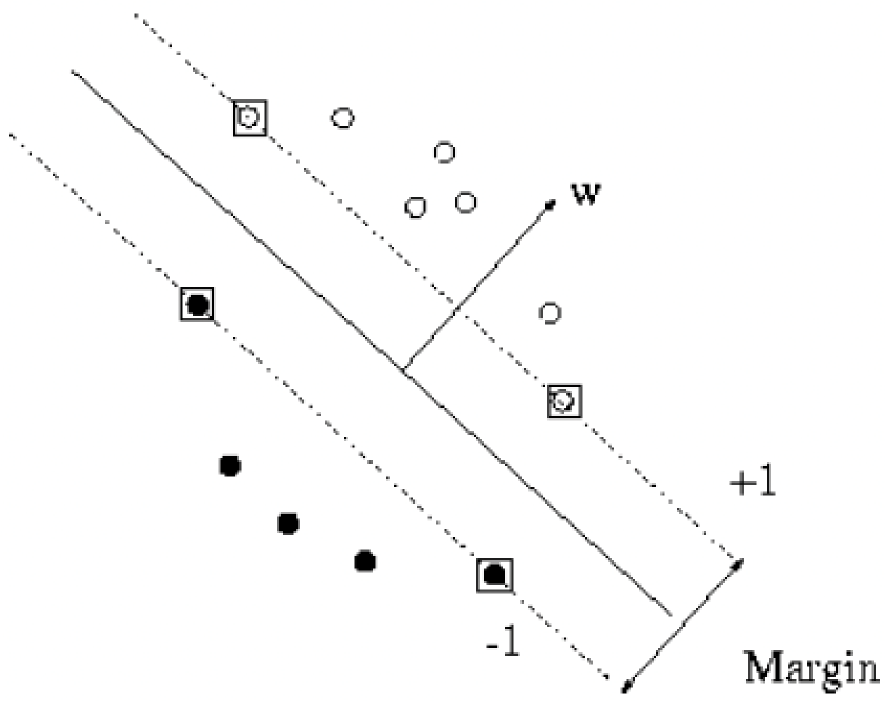
Mathematical Specification

- **decision boundary**: $<\mathbf{w}, \mathbf{x}>+b=0$
- **plus-plane**: hyperplane touching some positive examples, parallel to the decision boundary

  $$<\mathbf{w}, \mathbf{x}>+b=c \text{ for some constant } c$$

- **minus-plane**: hyperplane touching some negative examples, taking the form below since **decision boundary** is **half way** between **plus** and **minus** planes:

  $$<\mathbf{w}, \mathbf{x}>+b=-c$$

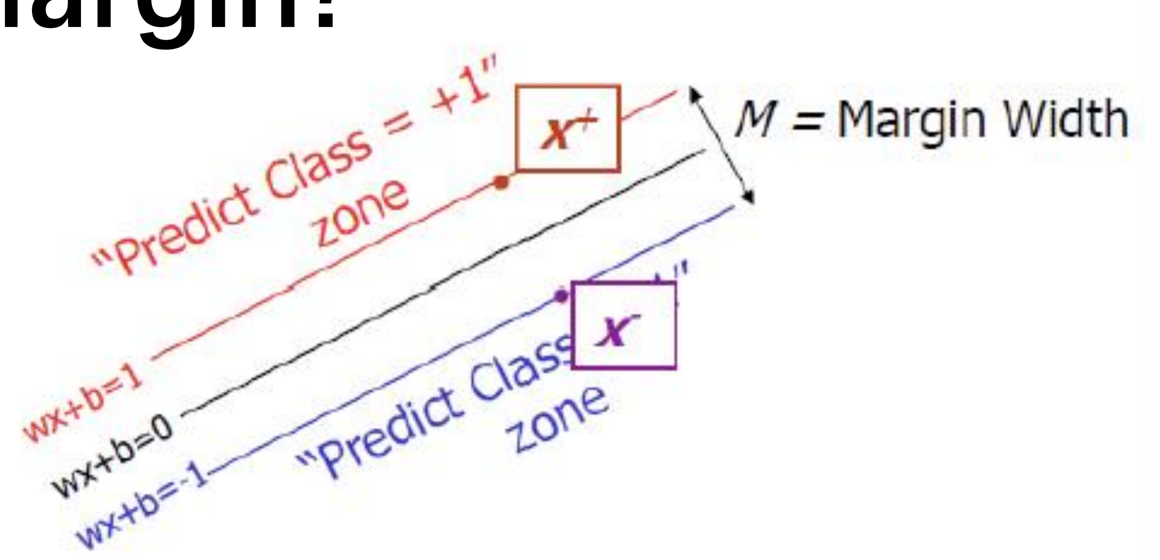


Mathematical Specification

- divide both sides by c, the planes remain the same
- rename $\mathbf{w}/c$ as $\mathbf{w}$ and b/c as b, we have
  - **decision boundary**: $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$
  - **plus-plane**: $\langle \mathbf{w}, \mathbf{x} \rangle + b = 1$
  - **minus-plane**: $\langle \mathbf{w}, \mathbf{x} \rangle + b = -1$
- $\mathbf{w}$ is **perpendicular to the 3 planes**, because for any two points $\mathbf{u}$ and $\mathbf{v}$ on the **decision boundary**, we have

$$\langle \mathbf{w}, \mathbf{u}-\mathbf{v} \rangle = 0$$

What is Margin?

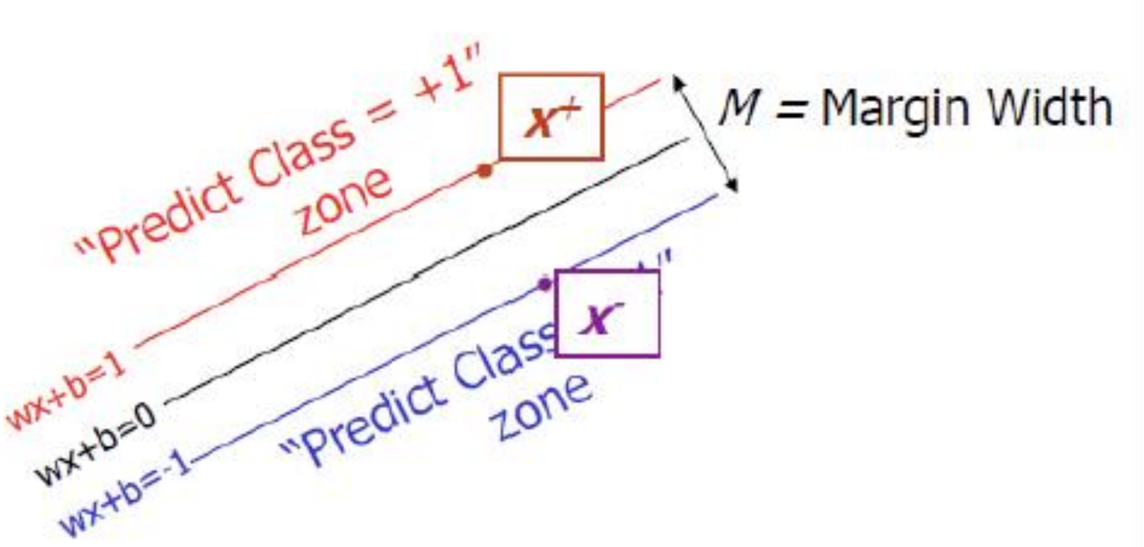


- a point $\mathbf{x}^-$ on **minus plane** and $\mathbf{x}^+$ on **plus plane** closest to $\mathbf{x}^-$ the line from $\mathbf{x}^-$ to $\mathbf{x}^+$ perpendicular to **3 planes**, so

$$\mathbf{x}^+ - \mathbf{x}^- = \lambda \mathbf{w} \text{ for some } \lambda \in \mathbb{R}$$

- by <**w**, $\mathbf{x}^+$>+b=1 and <**w**, $\mathbf{x}^-$>+b=-1, we have $\lambda = \frac{2}{\|\mathbf{w}\|_2^2}$
- the distance: $M := \|\mathbf{x}^+ - \mathbf{x}^-\|_2 = \|\lambda \mathbf{w}\|_2 = \dfrac{2}{\|\mathbf{w}\|_2}$

Optimal Margin Classifier



- **training set** $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
- find **w** and b to

$$\max \frac{2}{\|\mathbf{w}\|_2} \quad \text{s.t. } \langle \mathbf{w}, \mathbf{x}_i \rangle + b \begin{cases} \geq 1, & \text{if } y_i = 1 \\ \leq -1, & \text{if } y_i = -1. \end{cases} \qquad (i = 1, \ldots, n)$$

The Primal Optimization Problem

equivalent constrained optimization problem: find $\mathbf{w}$, b to

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|_2^2}{2} \quad \text{subject to } y_i\big(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\big) \geq 1, \ \forall i$$

- one constraint for each data point
- a **quadratic programming** (QP) problem
  - there are commercial softwares for solving it
- however, we will study the **dual optimization problem**
  - allow SVM to work efficiently with **high dimensional** data
  - which are necessary when dealing with data sets that are not **linearly separable**

Lagrangian

- The **Lagrangian** of the primal problem is

$$\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{\|\mathbf{w}\|_2^2}{2} - \sum_{i=1}^{n} \alpha_i \big(y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1\big)$$

- where $\alpha = (\alpha_1, \dots, \alpha_n) \geq 0$ are the **Lagrangian multipliers**
- **strong duality**: if data **linearly separable**, e.g, there is a **w** and *b* satisfying all constraints, then

$$\max_{\alpha:\alpha_i \geq 0} \min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \alpha) = \min_{\mathbf{w},b} \max_{\alpha:\alpha_i \geq 0} \mathcal{L}(\mathbf{w}, b, \alpha)$$

`the min and max operators are swapped!`

The Dual Optimization Problem

- for a given α, define $\mathcal{L}_d(\alpha) = \min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \alpha)$
- the **dual optimization problem**:

$$\max_{\alpha:\alpha_i \geq 0} \mathcal{L}_d(\alpha) = \max_{\alpha:\alpha_i \geq 0} \min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \alpha)$$

- for **fixed** α, first solve $\min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \alpha)$:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = 0 \\ \frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^n \alpha_i y_i = 0 \end{cases} \Longrightarrow \begin{cases} \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \\ \sum_{i=1}^n \alpha_i y_i = 0 \end{cases}$$

The Dual Optimization Problem

- plug **optimal** **w** and **constraint** for **fixed** α to **Lagrangian**, we get **dual problem** in terms of dual variables

$$\max_{\alpha} \; W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

$$\text{s.t.} \;\; \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- **quadratic programming** problem
- can be solved numerically by any general purpose optimization packages, e.g., **MATLAB** optimization toolbox
- finds **global optimal** (convex)

Support Vectors

- **KKT complementarity condition**: $\alpha_i \left[y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 \right] = 0$

- patterns for which $y_i\big(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\big) > 1$

  $\alpha_i = 0$ (**inactive** constraints): $\mathbf{x}_i$ irrelevant

- patterns that have $\alpha_i > 0$ (**active** constraints)

  $y_i\big(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\big) = 1$: lie either on margins

- solutions are **determined** by examples on the margin (**support vectors**): if all other training points are removed and training was repeated, the same hyperplane is found

How to Find b?

- if we solve the **QP** problem on page 14, we get optimal value for α^* and $\mathbf{w}$

$$\mathbf{w}(\alpha^*) = \sum_{i \in S} \alpha_i^* y_i \mathbf{x}_i$$

- where $S = \{i : \alpha_i^* > 0, i = 1, \ldots, n\}$ is the set of support vectors.
- how about optimal value for b?
- use again the **KKT complementarity condition**:
- any support vector ($\mathbf{x}_s$, y_s) satisfies $y_s\big(\langle \mathbf{w}(\alpha^*), \mathbf{x}_s \rangle + b(\alpha^*)\big) = 1$
- from which we know

$$b(\alpha^*) = \frac{1}{|S|} \sum_{s \in S} \left(\frac{1}{y_s} - \sum_{i \in S} \alpha_i^* y_i \langle \mathbf{x}_i, \mathbf{x}_s \rangle \right)$$

Prediction

- new instance **x** in which class?
- answer:

$$\text{sign}\big(\langle \mathbf{w}(\alpha^*), \mathbf{x}\rangle + b(\alpha^*)\big)$$

- recall that $\mathbf{w}(\alpha) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$, then

$$\text{sign}\big(\langle \mathbf{w}(\alpha^*), \mathbf{x}\rangle + b(\alpha^*)\big) = \text{sign}\Big(\sum_{i \in S} \alpha_i^* y_i \langle \mathbf{x}_i, \mathbf{x}\rangle + b(\alpha^*)\Big)$$

Data Not Linearly Separable/Regularization

When Data Not Linearly Separable

- if data **linearly separable**, find a plane that separates the two class with 0 error

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|_2^2}{2} \quad \text{s.t. } y_i\big(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\big) \geq 1, \ \forall i$$

- if data **not linearly separable**, try to find a plane separating two classes with **minimal** errors
- introduce positive **slack variables** ξ_i, the summation of which is an upper bound on the number of **training errors**

$$y_i\big(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\big) \geq 1 - \xi_i \quad \xi_i \geq 0 \ \forall i$$

- **penalize** $\sum_i \xi_i$ in the objective function

$$\min_{\mathbf{w}, \xi_i \geq 0} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \xi_i \quad \text{s.t. } y_i\big(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\big) \geq 1 - \xi_i$$

- the larger the constant C, the more we want to minimize error, the more complex the **decision boundary**

Lagrangian

- **Lagrangian**: with dual variables $\alpha_i \geq 0, \mu_i \geq 0$

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha, r) = \frac{\|\mathbf{w}\|_2^2}{2} + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \Big(y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 + \xi_i \Big) - \sum_{i=1}^{n} \mu_i \xi_i$$

- Solving the dual: $\min_{\mathbf{w},b,\xi} \mathcal{L}(\mathbf{w}, b, \xi, \alpha, \mu)$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \quad \Longrightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \quad \Longrightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \quad \Longrightarrow \quad C - \alpha_i - \mu_i = 0$$

Dual Problem

- **Dual**: still a **QP** problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

$$\text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C, \ \forall i$$

- **KKT condition**

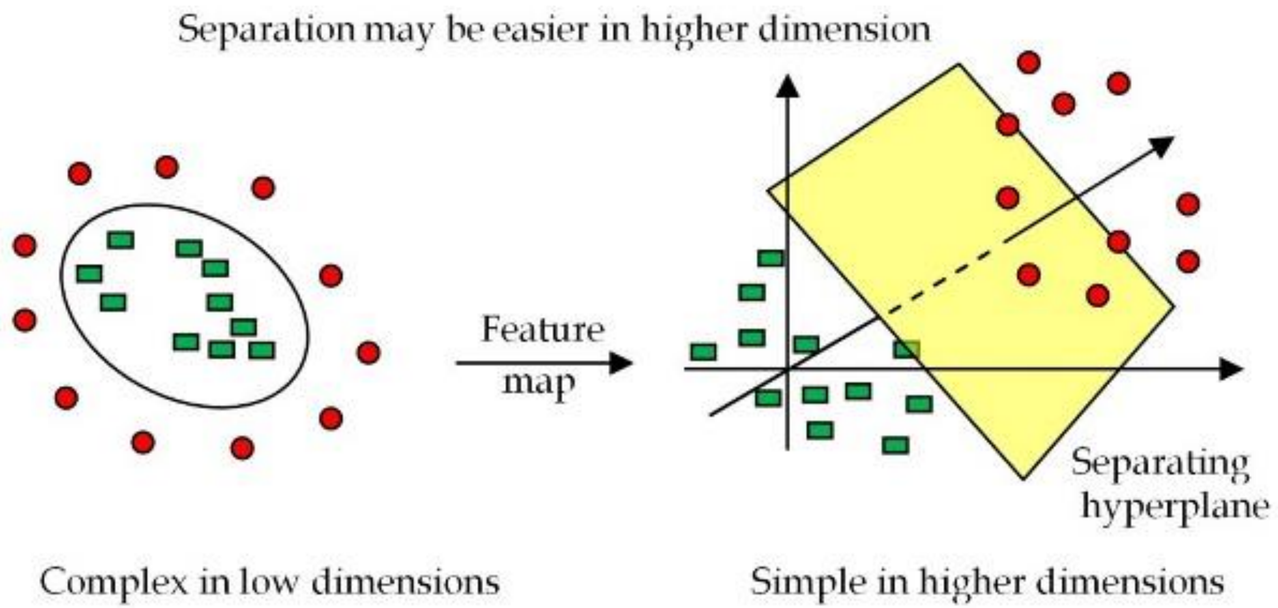
$$\begin{cases} \alpha_i \geq 0, \mu_i \geq 0 \\ \xi_i \geq 0, \mu_i \xi_i = 0, \\ y_i f(\mathbf{x}_i) - 1 + \xi_i \geq 0, \\ \alpha_i (y_i f(\mathbf{x}_i) - 1 + \xi_i) = 0. \end{cases}$$

- ▶ $\forall (\mathbf{x}_i, y_i)$, either $\alpha_i = 0$ or $y_i f(\mathbf{x}_i) = 1 - \xi_i$
- ▶ $\alpha_i = 0 \Rightarrow (\mathbf{x}_i, y_i)$ has no influence on f
- ▶ $\alpha_i > 0 \Rightarrow y_i f(\mathbf{x}_i) = 1 - \xi_i$ **support vector**
- ▶ $\alpha_i < C \Rightarrow \mu_i > 0$ and $\xi_i = 0$ (lie on margin)
- ▶ $\alpha_i = C \Rightarrow \mu_i = 0$. moreover, if $\xi_i \leq 1$, $(\mathbf{x}_i, y_i)$ lie within margin, otherwise misclassified

the prediction model only depend on support vectors!

Non-Linearly SVMs/Kernels

Nonlinear Decision Boundary & Feature Transformation

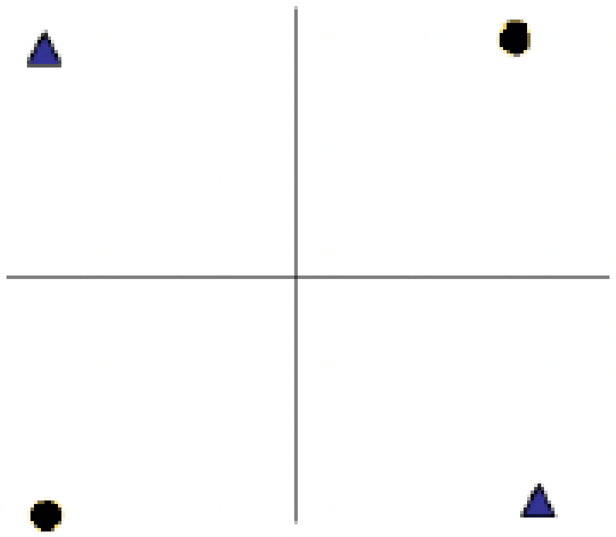


- mapping from the input space $\mathbb{R}^d$ (**attributes**) to a feature space $\mathcal{H}$ (**features**) ψ: $\mathbb{R}^d \to \mathcal{H}$, $\mathbf{x} \to \psi(\mathbf{x})$
- transform the data with the mapping

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \longrightarrow (\psi(\mathbf{x}_1), y_1), \ldots, (\psi(\mathbf{x}_n), y_n)$$

- we have **linear decision boundary** on feature space
- in general, the **higher** the dimension the feature space, the more likely data becomes **linearly separable**

Example



- data

$$(x_1, y_1; y) : (-1, -1; -1), (-1, 1; +1), (1, -1; +1), , (1, 1; -1)$$

- the data set is **not** linearly separable
- however, if we transform the data using

$$(x_1, x_2; y) \rightarrow (x_1, x_2, (x_1 x_2); y)$$

$$(-1, -1, 1; -1), (-1, 1, -1; +1)(+1, -1, -1; +1), (1, 1, 1; -1)$$

- **linearly separable**: $x_1 x_2 > 0 \Rightarrow -1, x_1 x_2 \leq 0 \Rightarrow +1$

Apply SVM After Feature Transformation

- Dual Problem on Features:

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

$$\text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C, \ \forall i$$

$\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ replaced by $\langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle$!

Kernel Trick

- Define $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle$ called **kernel function**
- Rewrite the problem as

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

$$\text{s.t.} \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C, \ \forall i$$

- **kernel trick**
  - **no need to explicitly calculate** ψ
  - **dot product** $\langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle$ realized by the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$
  - k is cheaper to calculate
  - allow one to use very high dimensional feature space

Common Kernels

- **linear kernel**: $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle$, **identity mapping**
- **polynomial kernel**: $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle^m$, corresponding to feature transformation
$$\psi(\mathbf{x}) = (x_1 x_1, x_1 x_2, \ldots, x_1 x_n, \ldots, x_n x_1, x_n x_2, \ldots, x_n x_n)$$
- **inhomogeneous polynomial**: $k(\mathbf{x}, \tilde{\mathbf{x}}) = \big(\langle \mathbf{x}, \tilde{\mathbf{x}} \rangle + 1\big)^m$
- **Gaussian kernel**: $k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp\big(-\|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 / (2\sigma^2)\big)$
  - **radial basis function** (RBF) network
  - corresponding to an **infinite-dimensional feature space**

Any algorithm that depends only on dot products can use the kernel trick!

Kernels

- Intuitively, $k(\mathbf{x}, \tilde{\mathbf{x}})$ represents our notion of **similarity** between data $\mathbf{x}$ and $\tilde{\mathbf{x}}$ and this is from our prior knowledge
- $k(\mathbf{x}, \tilde{\mathbf{x}})$ needs to satisfy a technical condition **(Mercer condition)** in order for ψ to exist
- **Mercer condition** for k to be a kernel function
  - there is a Hilbert space $\mathcal{F}$ for which k defines a **dot product**
  - the above is true if k is a **positive semidefinite function**: K is positive semi-definite for any $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_j) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_i, \mathbf{x}_1) & \cdots & k(\mathbf{x}_i, \mathbf{x}_j) & \cdots & k(\mathbf{x}_i, \mathbf{x}_n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_j) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Classification with SVM

- choose nonlinear transformation $\psi : \mathbb{R}^d \to \mathcal{H}$

$$\mathbf{x} \to \psi(\mathbf{x})$$

(**implicitly** via $k(\mathbf{x}, \tilde{\mathbf{x}}) =< \psi(\mathbf{x}), \psi(\tilde{\mathbf{x}}) >$)

- solve the dual optimization problem on features, get α^*
- calculate $\mathbf{w}(\alpha^*)$ and $b(\alpha^*)$ from α^*
- classify future examples as follows

$$\text{sign}\left(\sum_{i=1}^{n} \alpha_i^* y_i k(\mathbf{x}_i, \mathbf{x}) + b(\alpha^*)\right)$$

calculation of k suffices for training and prediction!

To be continued