

SUPPORT VECTOR MACHINES

Concha Bielza, Pedro Larrañaga

Computational Intelligence Group
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid



Machine Learning

Outline

1 Introduction

2 “Hard-margin” linear SVM

3 “Soft-margin” linear SVM

4 Nonlinear SVM

5 Conclusions

Outline

1 Introduction

2 “Hard-margin” linear SVM

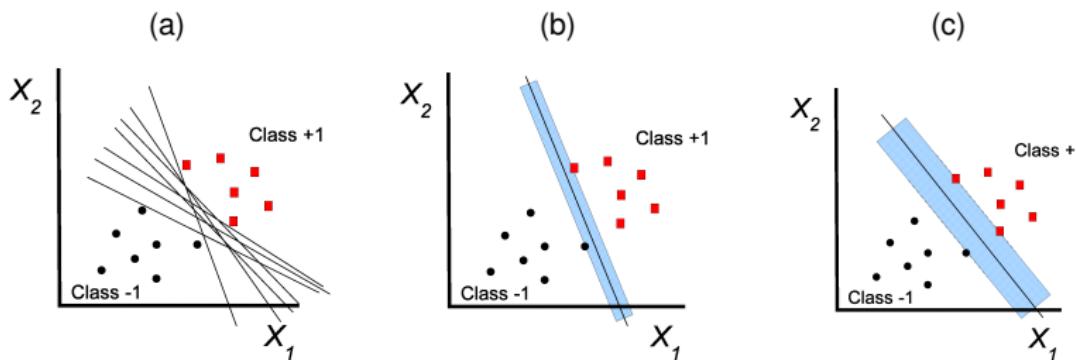
3 “Soft-margin” linear SVM

4 Nonlinear SVM

5 Conclusions

Motivation: linear SVM

- Data set $\mathcal{D} = \{(\mathbf{x}^1, c^1), \dots, (\mathbf{x}^N, c^N)\}$
- Assume $c^i \in \Omega_C = \{-1, 1\}$ and data is linearly separable:



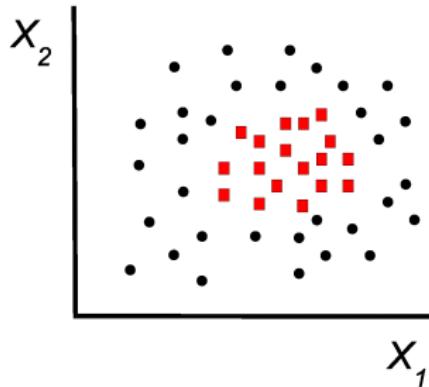
- (a) Infinite possible separating lines
 - (b) Margin → width that the line can be increased before hitting a data point
 - (c) Simplest (linear) SVM → the linear classifier with the maximum margin
- Choose the line as far as possible from the closest points –called support vectors– of both classes

Motivation: nonlinear SVM

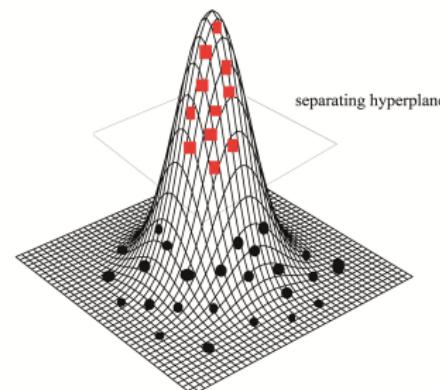
(a) If such linear decision boundary does **not exist...??**

- ⇒ Solution 1) Allow a **few** points to be on the **wrong side** (by using **slack variables**)
- ⇒ Solution 2) Map the data to a much **higher dimensional space** where a linear decision rule is found, *see (b)*
Construct the new space via clever mathematical projection known as "**kernel trick**"

(a)



(b)



Outline

1 Introduction

2 “Hard-margin” linear SVM

3 “Soft-margin” linear SVM

4 Nonlinear SVM

5 Conclusions

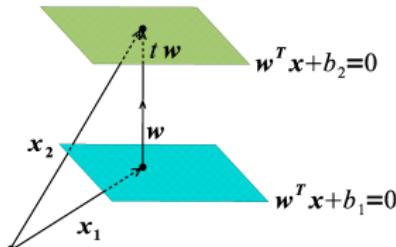
Linearly separable data: “Hard-margin” linear SVM

- Assume linear SVM trained on linearly separable data, i.e.,
 - There exists a hyperplane $\mathcal{H}: \mathbf{w}^T \mathbf{x} + b = 0$ which separates positive from negative instances
- The distance d between two parallel hyperplanes, $\mathbf{w}^T \mathbf{x} + b_1 = 0$ and $\mathbf{w}^T \mathbf{x} + b_2 = 0$, is $d = |b_1 - b_2| / \|\mathbf{w}\|$: see figure

$$d = |t| \cdot \|\mathbf{w}\|;$$

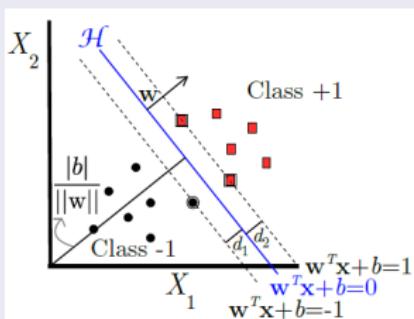
take \mathbf{x}_1 and \mathbf{x}_2 , a point on each hyperplane, $\mathbf{x}_2 = \mathbf{x}_1 + t\mathbf{w} \Rightarrow$

$$0 = \mathbf{w}^T \mathbf{x}_2 + b_2 = \mathbf{w}^T (\mathbf{x}_1 + t\mathbf{w}) + b_2 = \mathbf{w}^T \mathbf{x}_1 + t\|\mathbf{w}\|^2 + b_2 = -b_1 + t\|\mathbf{w}\|^2 + b_2$$



Linearly separable data: “Hard-margin” linear SVM

- Let d_2 (d_1) be the distance from \mathcal{H} to the closest positive (negative) instance
- Margin** of \mathcal{H} is defined as $d_1 + d_2$
- Linear SVM looks for \mathcal{H} with **largest margin** \Rightarrow how to find it?
- Points above (below) \mathcal{H} should have label 1 (-1); decision rule is: $\phi(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$
- Assume $\mathbf{w}^T \mathbf{x}^i + b \geq 1$ for $c^i = +1$ and $\mathbf{w}^T \mathbf{x}^i + b \leq -1$ for $c^i = -1$
 $\Leftrightarrow c^i(\mathbf{w}^T \mathbf{x}^i + b) \geq 1, \quad i = 1, \dots, N$
- For the **support vectors** the equality holds
- Dist. from support hyperplanes to \mathcal{H} is $d_1 = d_2 = 1/\|\mathbf{w}\| \Rightarrow$ Margin is $2/\|\mathbf{w}\|$



- No training points fall between the three hyperplanes
- \mathcal{H} must be as far from the support vectors (+ difficult points to classify) as possible
- Support vectors:** influence on the optimum location of the decision boundary; if removed, the solution changes

Primal and dual forms

- Max $\frac{2}{||\mathbf{w}||} \Leftrightarrow \text{Min } \frac{1}{2} ||\mathbf{w}||^2$ (allows quadratic programming optimization)

$$\begin{aligned} \text{Primal: } & \underset{\mathbf{w}, b}{\min} \frac{1}{2} ||\mathbf{w}||^2 \\ & \text{subject to} \end{aligned} \tag{1}$$

$$1 - \mathbf{c}^i(\mathbf{w}^T \mathbf{x}^i + b) \leq 0, \quad \forall i = 1, \dots, N$$

- ⇒ Solved by allocating each constraint a Lagrange multiplier or **dual variable** $\lambda_i \geq 0$ and then obtaining the dual form (with N variables rather than n)

$$\begin{aligned} \text{Dual: } & \underset{\lambda}{\max} L_D(\lambda) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j \mathbf{c}^i \mathbf{c}^j (\mathbf{x}^i)^T \mathbf{x}^j + \sum_{i=1}^N \lambda_i \\ & \text{subject to} \\ & \lambda_i \geq 0, \quad \forall i = 1, \dots, N \end{aligned} \tag{2}$$

$$\sum_{i=1}^N \lambda_i \mathbf{c}^i = 0$$

- Solving dual \Leftrightarrow Solving primal
- Problem (2) is a convex quadratic optimization problem: A **global maximum** can always be found (many methods available, like Seq. Min. Optimization (SMO))

Procedure

- Points \mathbf{x}^i for which $\lambda_i > 0$ are called support vectors
- All other points have $\lambda_i = 0$

Algorithm 1: Classification with hard margin linear SVM: Linearly sep. data

Input : A data set $\mathcal{D} = \{(\mathbf{x}^1, c^1), \dots, (\mathbf{x}^N, c^N)\}$ with $c^i \in \{-1, +1\}$

Output: Classification c^* of each new observation \mathbf{x}

- 1 Solve problem (1) by solving problem (2) and obtain $\lambda_i, i = 1, \dots, N$
- 2 Find set S , the indices i of support vectors such that $\lambda_i > 0$
- 3 Calculate $\mathbf{w} = \sum_{i \in S} \lambda_i c^i \mathbf{x}^i$
- 4 Calculate $b = \frac{1}{|S|} \sum_{s \in S} (c^s - \sum_{i \in S} \lambda_i c^i (\mathbf{x}^i)^T \mathbf{x}^s)$
- 5 Classify each new point \mathbf{x} as $c^* = \phi(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$

Outline

1 Introduction

2 “Hard-margin” linear SVM

3 “Soft-margin” linear SVM

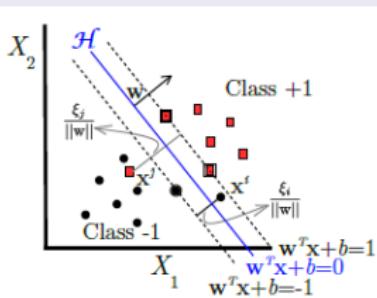
4 Nonlinear SVM

5 Conclusions

Non-linearly sep. data: “Soft-margin” linear SVM

- Assume non-linearly separable data, e.g., there are outliers, noise or slightly non-linear data
- Aim: handle this case without changing the family of decision functions
 - Relax the constraints slightly to allow for misclassifications, although with a cost
 - Can be done by introducing slack variables $\xi_i \geq 0, i = 1, \dots, N$ in the constraints

$$c^i(\mathbf{w}^T \mathbf{x}^i + b) \geq 1 - \xi_i$$



- Points on the wrong side of \mathcal{H} will have a penalty that increases with the distance from it
- ξ_i can be thought of as the distance from the support hyperplane for misclassified instances and 0 for correct classifications
- ξ_i thereby measures the degree of misclassification of \mathbf{x}^i

Primal and dual

$$\text{Primal: } \min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + M \sum_{i=1}^N \xi_i$$

subject to

(3)

- $M > 0$: fixed by the user. A larger M assigns a higher penalty to errors. M controls the trade-off between the slack variable penalty or errors and the size of the margin

$$\text{Dual: } \max_{\lambda} L_D(\lambda) = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j c^i c^j (\mathbf{x}^i)^T \mathbf{x}^j + \sum_{i=1}^N \lambda_i$$

subject to

$$0 \leq \lambda_i \leq M, \quad \forall i = 1, \dots, N \quad (4)$$

$$\sum_{i=1}^N \lambda_i c^i = 0$$

Procedure

Algorithm 2: Classification with soft margin linear SVM: Non-linearly sep. data

Input : A data set $\mathcal{D} = \{(\mathbf{x}^1, c^1), \dots, (\mathbf{x}^N, c^N)\}$ with $c^i \in \{-1, +1\}$, a constant
 $M > 0$

Output: Classification c^* of each new observation \mathbf{x}

- 1 Solve problem (3) by solving problem (4) and obtain $\lambda_i, i = 1, \dots, N$
 - 2 Find set S , the indices i of support vectors such that $0 < \lambda_i < M$
 - 3 Calculate $\mathbf{w} = \sum_{i \in S} \lambda_i c^i \mathbf{x}^i$
 - 4 Calculate $b = \frac{1}{|S|} \sum_{s \in S} (c^s - \sum_{i \in S} \lambda_i c^i (\mathbf{x}^i)^T \mathbf{x}^s)$
 - 5 Classify each new point \mathbf{x} as $c^* = \phi(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$
-

Outline

1 Introduction

2 “Hard-margin” linear SVM

3 “Soft-margin” linear SVM

4 Nonlinear SVM

5 Conclusions

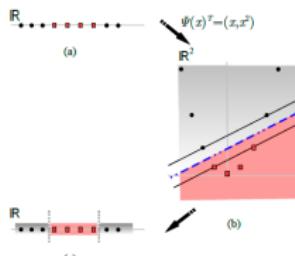
Non linearly separable data: nonlinear SVM

- Idea from **Cover's theorem**: given a set of training data that is **non-linearly** separable, it can be transformed with high probability into **linearly** separable by projecting it into a **higher-dimensional space** via some non-linear transformation
- Assume we map the data to some other (possibly infinite dimensional) **feature space \mathcal{F}** via a **non-linear mapping**

$$\begin{aligned}\psi : \mathbb{R}^n &\mapsto \mathcal{F} \\ \mathbf{x} &\mapsto \psi(\mathbf{x})\end{aligned}$$

- Data can be linearly separable in the new feature space. When the classifier is seen from the original input space it is non-linear

Example: map data x from \mathbb{R} to \mathbb{R}^2 by using $\psi(x) = (x, x^2)^T$



- We obtain the points in (b)
- (bottom): Original space with the classification and the associated nonlinear boundaries

Primal and dual forms

- Replace \mathbf{x} everywhere by their mapped versions $\psi(\mathbf{x})$
- A new instance is assigned the label $c^* = \phi(\mathbf{x}) = \text{sign}(\mathbf{w}^T \psi(\mathbf{x}) + b)$, with

$$\begin{aligned}\mathbf{w} &= \sum_{i \in S} \lambda_i c^i \psi(\mathbf{x}^i) \\ b &= \frac{1}{|S|} \sum_{s \in S} (c^s - \sum_{i \in S} \lambda_i c^i \psi(\mathbf{x}^i)^T \psi(\mathbf{x}^s))\end{aligned}$$

where S contains the indices i such that $0 < \lambda_i < M$

- Dual problem only requires the inner products $\psi(\mathbf{x}^i)^T \psi(\mathbf{x}^j)$ of mapped points
- A smart way to compute such inner product is known as **kernel**
 \Rightarrow **kernel trick** enters into play, allowing to recast the problem efficiently **without** using \mathbf{w} or even $\psi(\mathbf{x})$ alone

The kernel mapping

- A **kernel function** or simply a **kernel** K is a symmetric function of two arguments that returns in \mathbb{R} the value of the inner product of the mapped two arguments
$$K(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x})^T \psi(\mathbf{x}')$$
- Computing $K(\mathbf{x}^i, \mathbf{x}^j)$ directly can be less costly than $\psi(\mathbf{x}^i)^T \psi(\mathbf{x}^j)$ and avoids computing ψ explicitly

Example

- $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$
- $\psi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2)^T \in \mathbb{R}^6$
- $\Rightarrow \psi(\mathbf{x})^T \psi(\mathbf{x}') = 1 + 2x_1x'_1 + 2x_2x'_2 + 2x_1x'_1x_2x'_2 + x_1^2x'^2_1 + x_2^2x'^2_2$
- If we define $K(\mathbf{x}, \mathbf{x}') = (1 + x_1x'_1 + x_2x'_2)^2 = (1 + \mathbf{x}^T \mathbf{x}')^2 \dots = \psi(\mathbf{x})^T \psi(\mathbf{x}')$
- ...it operates on the lower dimension vectors \mathbf{x} and \mathbf{x}' to produce a value equivalent to the inner product of the higher-dimensional vectors
- Neither the mapping ψ nor the space \mathcal{F} are **unique** for a given kernel:
 \Rightarrow Example: $\psi_2(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1x_2, x_1x_2, x_1^2, x_2^2)^T \in \mathbb{R}^7$ is such that $\psi_2(\mathbf{x})^T \psi_2(\mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^2$ too

The kernel mapping

- A **kernel function** or simply a **kernel** K is a symmetric function of two arguments that returns in \mathbb{R} the value of the inner product of the mapped two arguments

$$K(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x})^T \psi(\mathbf{x}')$$
- Computing $K(\mathbf{x}^i, \mathbf{x}^j)$ directly can be less costly than $\psi(\mathbf{x}^i)^T \psi(\mathbf{x}^j)$ and avoids computing ψ explicitly

Example

- $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$
- $\psi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2)^T \in \mathbb{R}^6$
- $\Rightarrow \psi(\mathbf{x})^T \psi(\mathbf{x}') = 1 + 2x_1x'_1 + 2x_2x'_2 + 2x_1x'_1x_2x'_2 + x_1^2x'^2_1 + x_2^2x'^2_2$
- If we define $K(\mathbf{x}, \mathbf{x}') = (1 + x_1x'_1 + x_2x'_2)^2 = (1 + \mathbf{x}^T \mathbf{x}')^2 \dots = \psi(\mathbf{x})^T \psi(\mathbf{x}')$
- ...it operates on the lower dimension vectors \mathbf{x} and \mathbf{x}' to produce a value equivalent to the inner product of the higher-dimensional vectors
- Neither the mapping ψ nor the space \mathcal{F} are **unique** for a given kernel:
 \Rightarrow Example: $\psi_2(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1x_2, x_1x_2, x_1^2, x_2^2)^T \in \mathbb{R}^7$ is such that $\psi_2(\mathbf{x})^T \psi_2(\mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^2$ too

Procedure

Algorithm 3: Classification with nonlinear SVM

Input : A data set $\mathcal{D} = \{(\mathbf{x}^1, c^1), \dots, (\mathbf{x}^N, c^N)\}$ with $c^i \in \{-1, +1\}$, a constant $M > 0$

Output: Classification c^* of each new observation \mathbf{x}

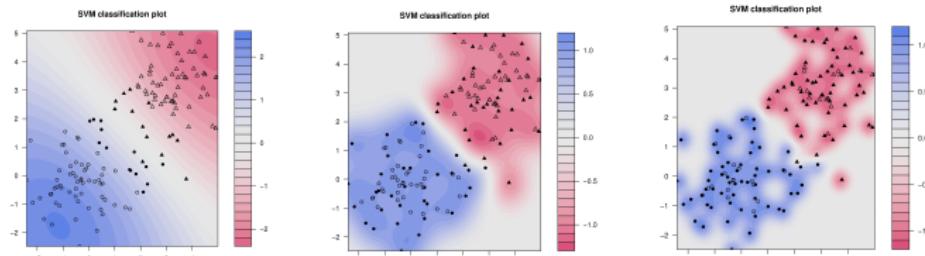
- 1 Solve problem (4) with $K(\mathbf{x}^i, \mathbf{x}^j)$ instead of $(\mathbf{x}^i)^T \mathbf{x}^j$ in the objective function, and obtain $\lambda_i, i = 1, \dots, N$
- 2 Find set S , the indices i of support vectors such that $0 < \lambda_i < M$
- 3 Calculate $b = \frac{1}{|S|} \sum_{s \in S} (c^s - \sum_{i \in S} \lambda_i c^i K(\mathbf{x}^i, \mathbf{x}^s))$
- 4 Classify each new point \mathbf{x} as $c^* = \phi(\mathbf{x}) = \text{sign}(\sum_{i \in S} \lambda_i c^i K(\mathbf{x}^i, \mathbf{x}) + b)$

Which is a valid kernel?

- It needs to satisfy the **Mercer's condition**: the $N \times N$ kernel matrix (also called the Gram matrix) \mathbf{K} in which the (i, j) entry is $K(\mathbf{x}^i, \mathbf{x}^j)$ is always positive-semidefinite, i.e., $\mathbf{x}^T \mathbf{K} \mathbf{x} \geq 0, \forall \mathbf{x} \neq 0$ in the training set
- Popular kernels:

Name	$K(\mathbf{x}, \mathbf{x}')$	Parameters
Polynomial	$(\mathbf{x}^T \mathbf{x}')^p$	degree $p \in \mathbb{N}$
Inhomogeneous polynomial	$(1 + \mathbf{x}^T \mathbf{x}')^p$	degree $p \in \mathbb{N}$
Gaussian RBF	$e^{-\frac{1}{2\sigma^2} \mathbf{x} - \mathbf{x}' ^2}$	width $\sigma > 0$
Sigmoidal	$\tanh(\kappa \mathbf{x}^T \mathbf{x}' - \theta)$	κ, θ

Gaussian RBF: As σ decreases (center and right) the flexibility of the decision boundary increases and can lead to overfitting (color gradient is the decision value)



Points in solid black are the support vectors. Note that a smaller σ demands more support vectors

Outline

1 Introduction

2 “Hard-margin” linear SVM

3 “Soft-margin” linear SVM

4 Nonlinear SVM

5 Conclusions

Conclusions

Summary

- Linear (hard and soft) SVM
- Kernel trick for the nonlinear SVM
- Multi-class SVM: $\Omega_C = \{1, \dots, R\}$: build many binary SVMs and then combine them

In Weka

SMO within *Functions*

Bibliography

Texts

- Bielza, C., Larrañaga, P. (2021) *Data-Driven Computational Neuroscience. Machine Learning and Statistical Models*, Cambridge University Press [Chap. 7]
- Burges, C.J.C. (1998) A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, 2, 2, 121-167
- Cristianini, N., Shawe-Taylor, J. (2000) *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press
- Platt, J.C. (1999) Fast training of support vector machines using sequential minimal optimization, In *Advances in Kernel Methods - Support Vector Learning*, pp. 185-208, The MIT Press
- Vapnik, V.N. (1998) *Statistical Learning Theory*, Wiley