Introductory Course: Machine Learning (WWI15B4)

Support Vector Machines

Fabio Ferreira, David Bethge

DHBW Karlsruhe

Overview

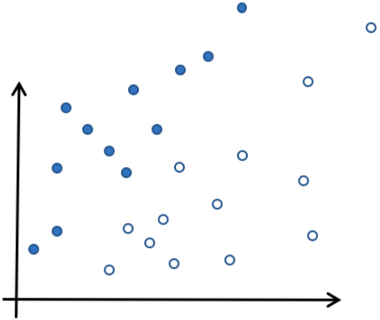
Table of Contents

Recommended Literature

- V.N. Vapnik: "Statistical Learning Theory", Wiley, 1998
- B. Schoelkopf: "Support Vector Learning"
- Patrick Winston, MIT 6.034 Artificial Intelligence, Fall 2010, `https://www.youtube.com/watch?v=_PwhiWxHK8o`

Introduction

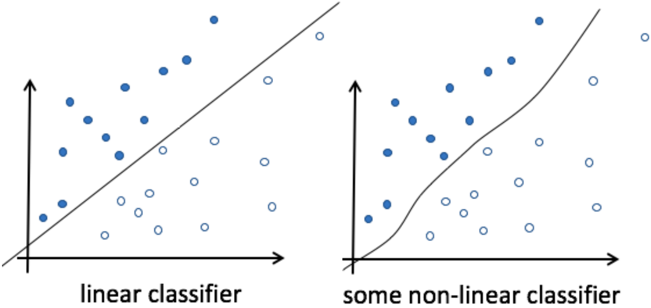
How to separate this space?



filled samples: positive, blank samples: negative

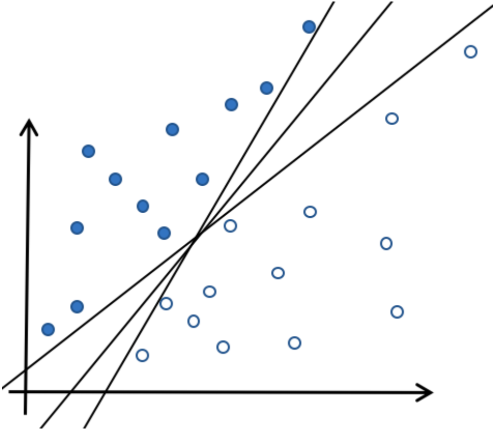
Introduction

Approaches we know so far would do something like this:

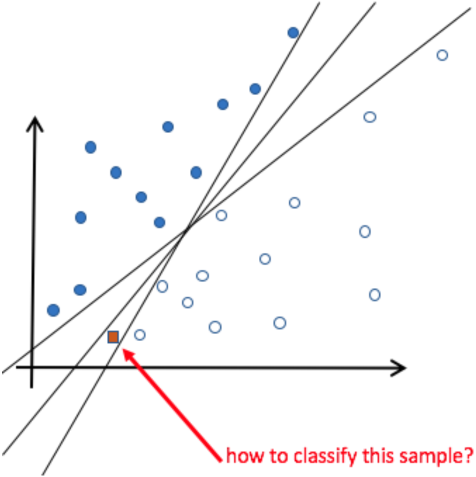


Introduction

There exist many hyperplanes that would correctly classify the data. Which one is the best?



Introduction



Introduction

Let's choose a hyperplane so that it represents the largest separation (margin) between both classes.
This yields the task: maximize the distance from the *middle line* to the nearest data point on each side.

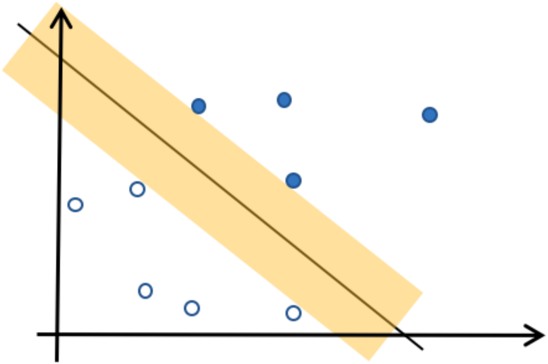
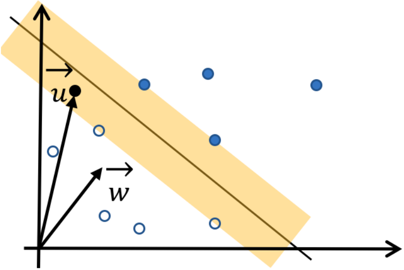


Table of Contents

Linear Maximum Margin Classifier: Decision Rule

Considering a vector pointing to an unknown sample $\vec{u}$ and a vector $\vec{w}$ of arbitrary length constrained to be perpendicular to the middle line. **Task:** Determine if $\vec{u}$ is on the right or left side of the hyperplane



Idea: project $\vec{u}$ onto $\vec{w}$ with some constant $c \in \mathbb{R}$:

$$\vec{w} \cdot \vec{u} \geq c$$

or

$$\vec{w} \cdot \vec{u} + b \geq 0, c = -b$$

Decision rule: if this inequality holds then $\vec{u}$ is a positive sample

Linear Maximum Margin Classifier: Constraints

Remember the decision rule: $\vec{w} \cdot \vec{u} + b \geq 0 \Rightarrow$ positive sample
Idea: if some unknown sample is a positive sample, we insist the decision rule yields ≥ 1 (otherwise ≤ -1.)

Linear Maximum Margin Classifier: Constraints

Remember the decision rule: $\vec{w} \cdot \vec{u} + b \geq 0 \Rightarrow$ positive sample
Idea: if some unknown sample is a positive sample, we insist the decision rule yields ≥ 1 (otherwise ≤ -1.)
Mathematically:

- for positive samples: $\vec{w} \cdot \vec{x}_+ + b \geq 1$
- for negative samples: $\vec{w} \cdot \vec{x}_- + b \leq -1$

Linear Maximum Margin Classifier: Constraints

Remember the decision rule: $\vec{w} \cdot \vec{u} + b \geq 0 \Rightarrow$ positive sample
Idea: if some unknown sample is a positive sample, we insist the decision rule yields ≥ 1 (otherwise ≤ -1.)
Mathematically:

- for positive samples: $\vec{w} \cdot \vec{x}_+ + b \geq 1$
- for negative samples: $\vec{w} \cdot \vec{x}_- + b \leq -1$

For convenience we introduce a variable y_i s.t.:

- $y_i = 1$ for positive samples and $y_i = -1$ for negative samples

The comfort we gain: only one inequality that holds for x_i laying outside of the margin boundaries

$$\boxed{y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1}$$

Linear Maximum Margin Classifier: Constraints

Remember the decision rule: $\vec{w} \cdot \vec{u} + b \geq 0 \Rightarrow$ positive sample
Idea: if some unknown sample is a positive sample, we insist the decision rule yields ≥ 1 (otherwise ≤ -1.)
Mathematically:

- for positive samples: $\vec{w} \cdot \vec{x}_+ + b \geq 1$
- for negative samples: $\vec{w} \cdot \vec{x}_- + b \leq -1$

For convenience we introduce a variable y_i s.t.:

- $y_i = 1$ for positive samples and $y_i = -1$ for negative samples

The comfort we gain: only one inequality that holds for x_i laying outside of the margin boundaries

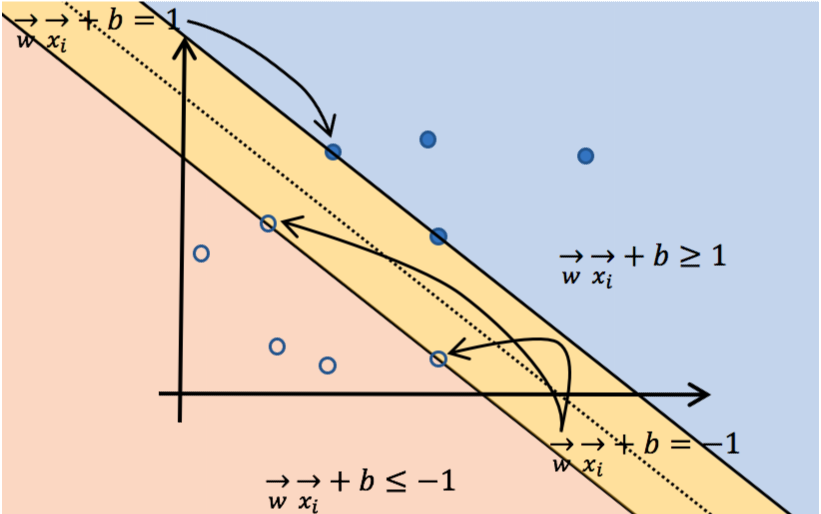
$$\boxed{y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1}$$

and we add one additional constraint for x_i placed on the margin boundaries:

$$\boxed{y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0}$$

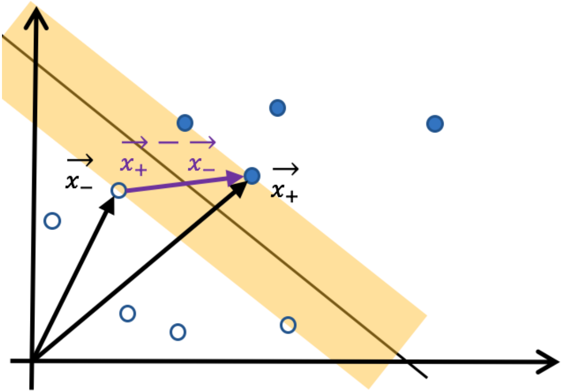
Linear Maximum Margin Classifier: Constraints

Geometrically this gives us:



Linear Maximum Margin Classifier: Margin Width

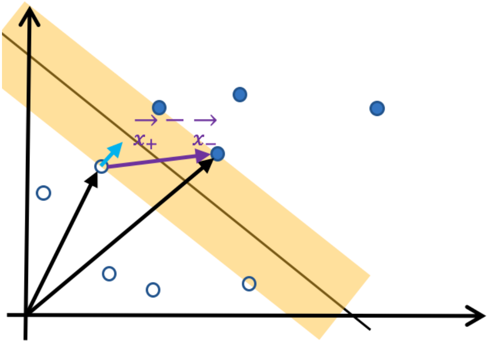
Recall: we want to maximize the distance between points of two different classes. This raises the question: how to express the distance between the two margin boundaries?



Linear Maximum Margin Classifier: Margin Width

How to express the distance between the two margin boundaries?

One solution: compute the width with a unit vector (light blue) and project the purple vector on that unit vector



$$width = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{||\vec{w}||}$$

Linear Maximum Margin Classifier: Margin Width

$$width = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

now use $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0$ from before for to get:

Linear Maximum Margin Classifier: Margin Width

$$width = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

now use $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0$ from before for to get:

$$width = \frac{(1 - b + 1 + b)}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$

Linear Maximum Margin Classifier: Margin Width

$$width = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

now use $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0$ from before for to get:

$$width = \frac{(1 - b + 1 + b)}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$

our goal now is:

$$\max \frac{2}{\|\vec{w}\|} = \max \frac{1}{\|\vec{w}\|} = \min \|\vec{w}\| \rightsquigarrow \min \frac{1}{2}\|\vec{w}\|^2$$

Linear Maximum Margin Classifier: Margin Width

$$width = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

now use $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0$ from before for to get:

$$width = \frac{(1 - b + 1 + b)}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$

our goal now is:

$$\max \frac{2}{\|\vec{w}\|} = \max \frac{1}{\|\vec{w}\|} = \min \|\vec{w}\| \rightsquigarrow \min \frac{1}{2} \|\vec{w}\|^2$$

find extremum of a function with constraints

$\rightarrow$ use Lagrangian optimization (method of Lagrange multipliers)

$\rightarrow$ yields a new (closed) expression with the constraints included

Linear Maximum Margin Classifier: Lagrangian Multipliers

Recall: we had defined a constraint for x_i placed directly on the margin boundaries: $y_i(\vec{x_i} \cdot \vec{w} + b) - 1 = 0 \rightarrow$ re-use it for the Lagrangian for m samples:

$$L(\vec{w}, \vec{\alpha}, b) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\vec{w} \cdot \vec{x_i} + b) - 1] \qquad (1)$$

Linear Maximum Margin Classifier: Lagrangian Multipliers

Recall: we had defined a constraint for x_i placed directly on the margin boundaries: $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0 \rightarrow$ re-use it for the Lagrangian for m samples:

$$L(\vec{w}, \vec{\alpha}, b) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1] \tag{1}$$

compute the minimum/first partial derivatives of L w.r.t. $\vec{w}$ and b:

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i = 0 \Rightarrow \boxed{\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i} \tag{2}$$

Linear Maximum Margin Classifier: Lagrangian Multipliers

Recall: we had defined a constraint for x_i placed directly on the margin boundaries: $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0 \rightarrow$ re-use it for the Lagrangian for m samples:

$$L(\vec{w}, \vec{\alpha}, b) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + b) - 1] \tag{1}$$

compute the minimum/first partial derivatives of L w.r.t. $\vec{w}$ and b:

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i = 0 \Rightarrow \boxed{\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i} \tag{2}$$

$$\frac{\partial L}{\partial \vec{b}} = -\sum_{i=1}^{m} \alpha_i y_i = 0 \Rightarrow \boxed{\sum_{i=1}^{m} \alpha_i y_i = 0} \tag{3}$$

Linear Maximum Margin Classifier: Lagrangian Multipliers

now we plug eq. 2 into eq. 1, simplify it and get:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j} \tag{4}$$

[1] note that W is a quadratic function $\Rightarrow$ convex problem

Linear Maximum Margin Classifier: Lagrangian Multipliers

now we plug eq. 2 into eq. 1, simplify it and get:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j} \tag{4}$$

- W is only dependent on α_i

[1]note that W is a quadratic function $\Rightarrow$ convex problem

Linear Maximum Margin Classifier: Lagrangian Multipliers

now we plug eq. 2 into eq. 1, simplify it and get:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j} \tag{4}$$

- W is only dependent on α_i
- Now maximize W to get $\vec{\alpha}$ with subject to eq. 3 (we call this procedure *finding the saddle point (minimax point)*)[1]

[1] note that W is a quadratic function $\Rightarrow$ convex problem

Linear Maximum Margin Classifier: Lagrangian Multipliers

now we plug eq. 2 into eq. 1, simplify it and get:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j} \tag{4}$$

- W is only dependent on α_i
- Now maximize W to get $\vec{\alpha}$ with subject to eq. 3 (we call this procedure *finding the saddle point (minimax point)*)[1]
- After optimization we will observe that most $\alpha_i = 0$

[1] note that W is a quadratic function $\Rightarrow$ convex problem

Linear Maximum Margin Classifier: Lagrangian Multipliers

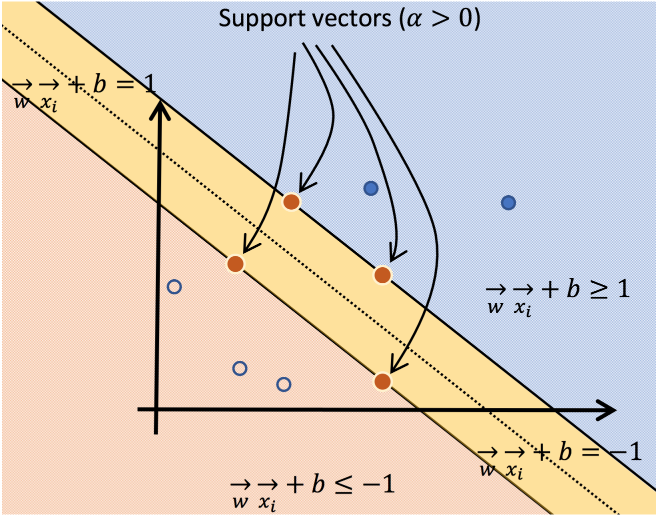
now we plug eq. 2 into eq. 1, simplify it and get:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j} \tag{4}$$

- W is only dependent on α_i
- Now maximize W to get $\vec{\alpha}$ with subject to eq. 3 (we call this procedure *finding the saddle point (minimax point)*)[1]
- After optimization we will observe that most $\alpha_i = 0$
- Those $\vec{x_i}$ with $\alpha_i > 0$ we call **support vectors** which all lie perpendicular to the margin line

[1] note that W is a quadratic function $\Rightarrow$ convex problem

Linear Maximum Margin Classifier: Support Vectors



Linear Maximum Margin Classifier: Lagrangian Multipliers

Recall the the **decision rule** $\vec{w} \cdot \vec{u} + b \geq 0$ for positive samples, insert $\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x_i}$ (eq. 2) and we get:

$$\sum_{i=1}^{m} \alpha_i y_i \vec{x_i} \cdot \vec{u} + b \geq 0$$

for a positive (unknown) sample $\vec{u}$. The decision rule now also **only depends on α_i and on the the dot product between $\vec{x_i}$ and $\vec{u}$**

Linear Maximum Margin Classifier: Lagrangian Multipliers

Recall the the **decision rule** $\vec{w} \cdot \vec{u} + b \geq 0$ for positive samples, insert $\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x_i}$ (eq. 2) and we get:

$$\sum_{i=1}^{m} \alpha_i y_i \vec{x_i} \cdot \vec{u} + b \geq 0$$

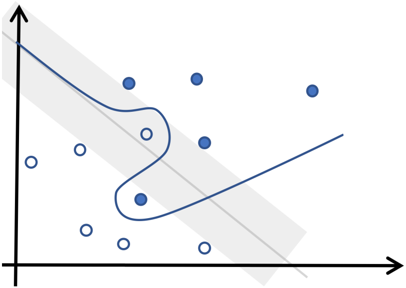
for a positive (unknown) sample $\vec{u}$. The decision rule now also **only depends on α_i and on the the dot product between $\vec{x_i}$ and $\vec{u}$**
This lets us specify a classification rule:

$$f(\vec{u}) = sgn(\vec{w} \cdot \vec{u} + b) = \boxed{sgn\left(\sum_{i=1}^{m} \alpha_i y_i \vec{x_i} \cdot \vec{u} + b\right)}$$

Table of Contents

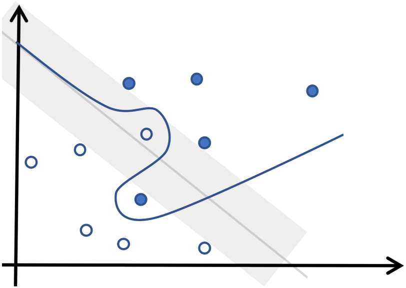
Non-linear separable data

What if the data is not linearly separable (as in most practical cases)?



Non-linear separable data

What if the data is not linearly separable (as in most practical cases)?



⇒ linear SVM won't converge. Two common solutions:

- adjust SVM specification to use a soft margin
- apply kernel methods
- (or both)

Table of Contents

Soft Margin

Solution: instead of specifying a hard margin, specify a soft margin with a slack variable $\xi \geq 0 \Rightarrow$ demand that not all samples must be correctly classified:

Soft Margin

Solution: instead of specifying a hard margin, specify a soft margin with a slack variable $\xi \geq 0 \Rightarrow$ demand that not all samples must be correctly classified:

- slightly change constraint from
  $y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1$
  to
  $y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1 - \xi_i$

Soft Margin

Solution: instead of specifying a hard margin, specify a soft margin with a slack variable $\xi \geq 0 \Rightarrow$ demand that not all samples must be correctly classified:

- slightly change constraint from
  $y_i(\vec{x_i} \cdot \vec{w} + b) \geq 1$
  to
  $y_i(\vec{x_i} \cdot \vec{w} + b) \geq 1 - \xi_i$
- now the optimal hyperplane is given by:

$$\boxed{\min \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^{m} \xi_i}$$

Soft Margin

Solution: instead of specifying a hard margin, specify a soft margin with a slack variable $\xi \geq 0 \Rightarrow$ demand that not all samples must be correctly classified:

- slightly change constraint from
  $y_i(\vec{x_i} \cdot \vec{w} + b) \geq 1$
  to
  $y_i(\vec{x_i} \cdot \vec{w} + b) \geq 1 - \xi_i$
- now the optimal hyperplane is given by:
  $$\boxed{\min \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m}\xi_i}$$
- re-apply maximization of $W(\vec{\alpha})$ w.r.t. $0 \leq \alpha_i \leq C$, $\xi_i \geq 0$

Soft Margin

Solution: instead of specifying a hard margin, specify a soft margin with a slack variable $\xi \geq 0 \Rightarrow$ demand that not all samples must be correctly classified:

- slightly change constraint from
  $y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1$
  to
  $y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1 - \xi_i$
- now the optimal hyperplane is given by:
  $$\boxed{\min \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m} \xi_i}$$
- re-apply maximization of $W(\vec{\alpha})$ w.r.t. $0 \leq \alpha_i \leq C$, $\xi_i \geq 0$
- C is a regularization parameter (usually $C \in [0, 1]$)

Soft Margin

Solution: instead of specifying a hard margin, specify a soft margin with a slack variable $\xi \geq 0 \Rightarrow$ demand that not all samples must be correctly classified:

- slightly change constraint from
  $y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1$
  to
  $y_i(\vec{x}_i \cdot \vec{w} + b) \geq 1 - \xi_i$
- now the optimal hyperplane is given by:
  $$\boxed{\min \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m}\xi_i}$$
- re-apply maximization of $W(\vec{\alpha})$ w.r.t. $0 \leq \alpha_i \leq C$, $\xi_i \geq 0$
- C is a regularization parameter (usually $C \in [0, 1]$)
- if C large $\Rightarrow$ enforce only few misclassified samples

Soft Margin

Solution: instead of specifying a hard margin, specify a soft margin with a slack variable $\xi \geq 0 \Rightarrow$ demand that not all samples must be correctly classified:

- slightly change constraint from
  $y_i(\vec{x_i} \cdot \vec{w} + b) \geq 1$
  to
  $y_i(\vec{x_i} \cdot \vec{w} + b) \geq 1 - \xi_i$
- now the optimal hyperplane is given by:

$$\boxed{\min \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m}\xi_i}$$

- re-apply maximization of $W(\vec{\alpha})$ w.r.t. $0 \leq \alpha_i \leq C$, $\xi_i \geq 0$
- C is a regularization parameter (usually $C \in [0, 1]$)
- if C large $\Rightarrow$ enforce only few misclassified samples
- if C small $\Rightarrow$ more misclassified samples allowed

Soft Margin Example

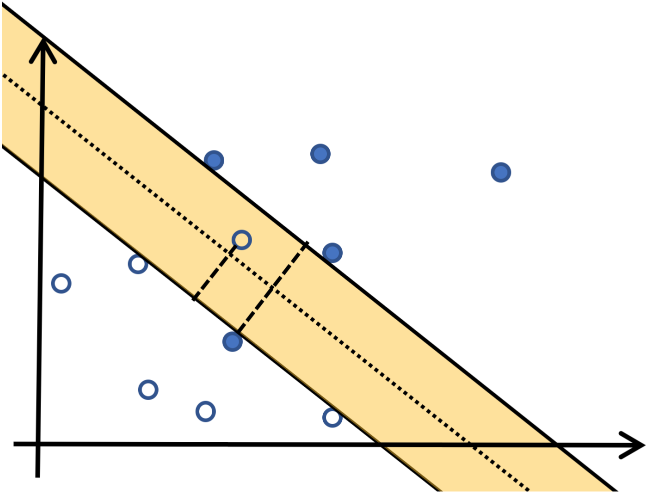


Table of Contents

Example

Often data samples are

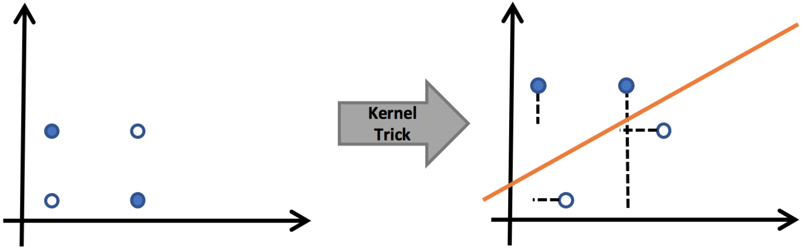
- not linearly separable in the original space
- but linearly separable in a higher-dimensional space

Example

Often data samples are

- not linearly separable in the original space
- but linearly separable in a higher-dimensional space

use the *kernel trick* for projecting into such higher-dimensional space, for example:



Kernel Trick

Kernel Trick

The approach of transforming data into an **implicitly** higher-dimensional space without computing coordinates of the data in that space, but rather by computing pairwise inner products of the samples. Typically $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$ [Wikipedia]

Kernel Trick

Kernel Trick

The approach of transforming data into an **implicitly** higher-dimensional space without computing coordinates of the data in that space, but rather by computing pairwise inner products of the samples. Typically $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$ [Wikipedia]

Mapping $\phi : \mathbb{R}^n - > \mathbb{R}^m$, usually $n > m$

Kernel Trick

Kernel Trick

The approach of transforming data into an **implicitly** higher-dimensional space without computing coordinates of the data in that space, but rather by computing pairwise inner products of the samples. Typically $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$ [Wikipedia]

Mapping $\phi : \mathbb{R}^n - > \mathbb{R}^m$, usually $n > m$
Some clarification:

- the kernel trick **does not** produce a **mapping** from low to high-dimensional space
- it does provide a solution to compute inner products of samples in high-dim. space **without knowing the mapping**

Kernel Trick

Kernel Trick

The approach of transforming data into an **implicitly** higher-dimensional space without computing coordinates of the data in that space, but rather by computing pairwise inner products of the samples. Typically $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$ [Wikipedia]

Mapping $\phi : \mathbb{R}^n -> \mathbb{R}^m$, usually $n > m$
Some clarification:

- the kernel trick **does not** produce a **mapping** from low to high-dimensional space
- it does provide a solution to compute inner products of samples in high-dim. space **without knowing the mapping**
- Advantages: low-cost computation, operating in infinite spaces (e.g. Gaussian kernel) possible

Kernel Trick Example

Example for $K(\vec{x}, \vec{z}) = (\vec{x} \cdot \vec{z})^2$
without using the kernel trick (explicit mapping):

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_{\mathbb{R}^3} \qquad \phi(x) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_3 x_3 \end{bmatrix}_{\mathbb{R}^9}$$

$\Rightarrow$ 18 multiplications (project x and $z : \mathbb{R}^3 \rightarrow \mathbb{R}^9$) + 9 multiplications + 8 additions (inner product) = 35 operations

Kernel Trick Example

Example for $K(\vec{x}, \vec{z}) = (\vec{x} \cdot \vec{z})^2$
using the kernel trick:

$$\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_1 \\ z_1 \end{bmatrix} \right)^2 = (x_1 z_1 + x_2 z_2 + x_3 z_3)^2$$

$\Rightarrow$ 3 multiplications + 2 additions + 1 multiplication $((\cdot)^2)$
$=$ 6 operations

Kernel Trick in the SVM

Where the kernel trick is used in the SVM:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \boxed{\vec{x_i} \cdot \vec{x_j}}$$

[2] must fulfill the Mercer theorem

[3] also called Gaussian kernel

Kernel Trick in the SVM

Where the kernel trick is used in the SVM:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \boxed{\vec{x_i} \cdot \vec{x_j}}$$

Note: not all functions $\phi : \mathrm{I\!R}^{\mathrm{n}} \rightarrow \mathrm{I\!R}^{\mathrm{m}}, (n, m) \in \mathrm{I\!R}$ are valid kernel functions[2].

[2] must fulfill the Mercer theorem

[3] also called Gaussian kernel

Kernel Trick in the SVM

Where the kernel trick is used in the SVM:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \boxed{\vec{x_i} \cdot \vec{x_j}}$$

Note: not all functions $\phi : \mathbb{R}^{\mathrm{n}} \to \mathbb{R}^{\mathrm{m}}, (n, m) \in \mathbb{R}$ are valid kernel functions[2].

Popular kernel functions are:

- inner product: $K(\vec{x}, \vec{z}) = \vec{x} \cdot \vec{z}$
- degree-d polynomial: $K(\vec{x}, \vec{z}) = (\vec{x} \cdot \vec{z} + c)^d, c \geq 0$

---

[2] must fulfill the Mercer theorem

[3] also called Gaussian kernel

Kernel Trick in the SVM

Where the kernel trick is used in the SVM:

$$W(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \boxed{\vec{x_i} \cdot \vec{x_j}}$$

Note: not all functions $\phi : \mathbb{R}^n \to \mathbb{R}^m, (n, m) \in \mathbb{R}$ are valid kernel functions[2].

Popular kernel functions are:

- inner product: $K(\vec{x}, \vec{z}) = \vec{x} \cdot \vec{z}$
- degree-d polynomial: $K(\vec{x}, \vec{z}) = (\vec{x} \cdot \vec{z} + c)^d, c \geq 0$
- Gaussian radial basis function[3]: $K(\vec{x}, \vec{z}) = \exp(-\frac{\|\vec{x}-\vec{z}\|^2}{2\sigma^2})$

---

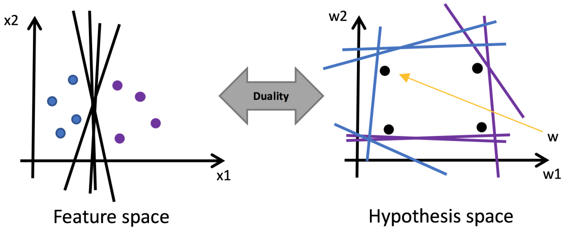
[2] must fulfill the Mercer theorem

[3] also called Gaussian kernel

Table of Contents

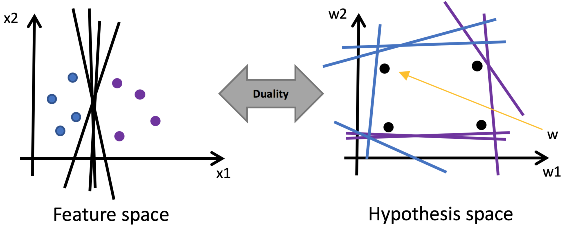
Duality of feature and hypothesis space

Points in the feature space correspond to hyperplanes in the hypothesis space and vice versa ("Statistical Learning Theory", Vapnik, 1998).



Duality of feature and hypothesis space

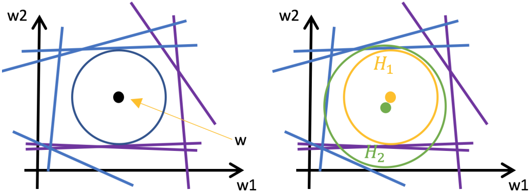
Points in the feature space correspond to hyperplanes in the hypothesis space and vice versa ("Statistical Learning Theory", Vapnik, 1998).



Implications:

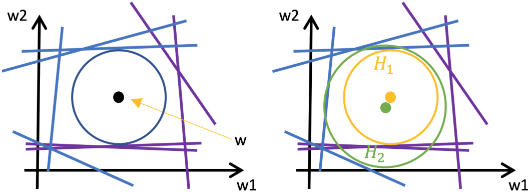
- the more data points, the more the hypothesis space will be constrained
- maximum margin search means searching for hyper planes with largest distance to data points $\Rightarrow$ center point of hyper sphere

Structural Risk Minimization



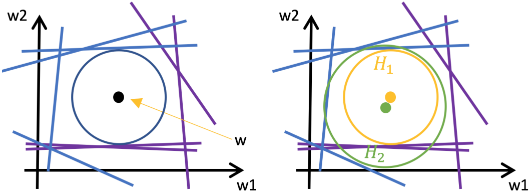
- during the saddle point search in the SVM (Lagrange optimization) more and more data samples are considered

Structural Risk Minimization



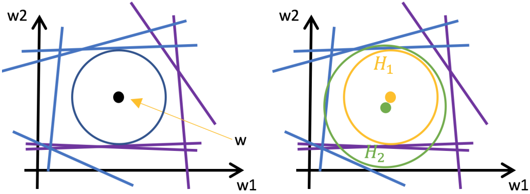
- during the saddle point search in the SVM (Lagrange optimization) more and more data samples are considered
- this successively constrains the hypothesis search space

Structural Risk Minimization



- during the saddle point search in the SVM (Lagrange optimization) more and more data samples are considered
- this successively constrains the hypothesis search space
- then, the best hyper plane with the smallest empirical error is chosen (center point of hyper sphere)

Structural Risk Minimization



- during the saddle point search in the SVM (Lagrange optimization) more and more data samples are considered
- this successively constrains the hypothesis search space
- then, the best hyper plane with the smallest empirical error is chosen (center point of hyper sphere)
- recall from concept learning lecture: this is **Structural Risk Minimization** e.g. $\ldots H_3 \subset H_2 \subset H_1$

Table of Contents

Advantages

- SVM optimization problem is convex (no local minima)
- can handle high-dimensional data well
- fast test time execution (usually few $\alpha_i > 0 \Rightarrow$ few inner products, if linear SVM: $\vec{w}$ can always be pre-computed [use eq. 2], if non-linear SVM: no pre-computation of $\vec{w}$ guaranteed [e.g. Gaussian kernel] but computing inner products between support vector train samples and a new sample is still relatively cheap)

Advantages

- SVM optimization problem is convex (no local minima)
- can handle high-dimensional data well
- fast test time execution (usually few $\alpha_i > 0 \Rightarrow$ few inner products, if linear SVM: $\vec{w}$ can always be pre-computed [use eq. 2], if non-linear SVM: no pre-computation of $\vec{w}$ guaranteed [e.g. Gaussian kernel] but computing inner products between support vector train samples and a new sample is still relatively cheap)

Disadvantages

- data samples have to be stored (space complexity not negligible, however, SVM is still not a lazy-learner since it learns a decision boundary $\Rightarrow$ eager-learner)
- number of support vectors depend on problem
- no pre-processing of the data in the SVM approach included
- finding optimal kernel can be tedious

Reading Assignment

Use the Internet to gain knowledge about the following topics:

- multi-class SVM (one-vs-all and one-vs-one)
- where the kernel trick is further applied (in addition to the SVM)