Machine Learning

Support Vector Machines

Fabio Vandin

December 2nd, 2022

Duality

We now present (Hard-)SVM in a different way which is very useful for *kernels*.

We want to solve

data is linearly separable

$$\mathbf{w}_0 = \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } \forall i : y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1$$

Duality

We now present (Hard-)SVM in a different way which is very useful for *kernels*.
We want to solve

$$\mathbf{w}_0 = \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } \forall i : y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1$$

One can prove (details in the book!) that $\mathbf{w}$ that minimizes the function above is equivalent to find $\boldsymbol{\alpha}$ that solves the *dual problem*:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq \mathbf{0}} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \right)$$

Duality

We now present (Hard-)SVM in a different way which is very useful for *kernels*.
We want to solve

$$\mathbf{w}_0 = \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } \forall i : y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1$$

One can prove (details in the book!) that $\mathbf{w}$ that minimizes the function above is equivalent to find $\boldsymbol{\alpha}$ that solves the *dual problem*:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq \mathbf{0}} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \right)$$

Note:

- solution is the vector $\boldsymbol{\alpha}$ which defines the support vectors $= \{\mathbf{x}_i : \alpha_i \neq 0\}$
- $\mathbf{w}_0$ can be derived from $\boldsymbol{\alpha}$ (see previous slides!)

Duality

We now present (Hard-)SVM in a different way which is very useful for *kernels*.
We want to solve

$$\mathbf{w}_0 = \min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } \forall i : y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1$$

One can prove (details in the book!) that $\mathbf{w}$ that minimizes the function above is equivalent to find $\boldsymbol{\alpha}$ that solves the *dual problem*:

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq \mathbf{0}} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \right)$$

Note:

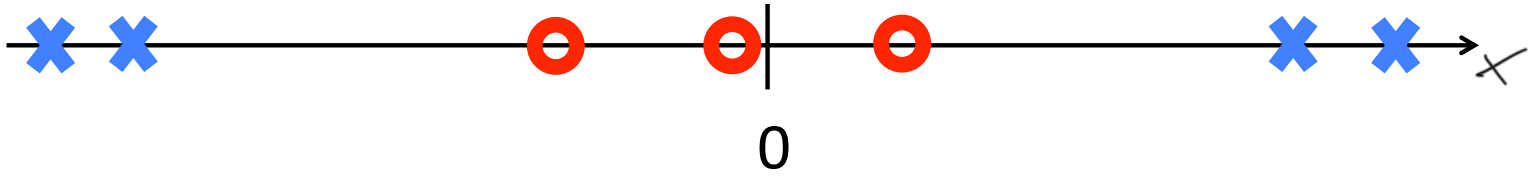
- solution is the vector $\boldsymbol{\alpha}$ which defines the support vectors $= \{\mathbf{x}_i : \alpha_i \neq 0\}$
- $\mathbf{w}_0$ can be derived from $\boldsymbol{\alpha}$ (see previous slides!)
- dual problem requires only to compute inner products $\langle \mathbf{x}_j, \mathbf{x}_i \rangle$, does not need to consider $\mathbf{x}_i$ by itself

SVM is a powerful algorithm, but still limited to linear models...
and linear models cannot always be used (directly)!

$\vec{x} = [x]$

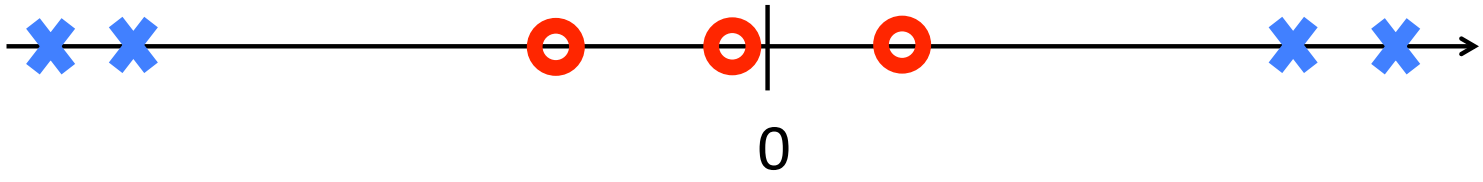
$\mathcal{X} = \mathbb{R}$

Example



SVM is a powerful algorithm, but still limited to linear models...
and linear models cannot always be used (directly)!

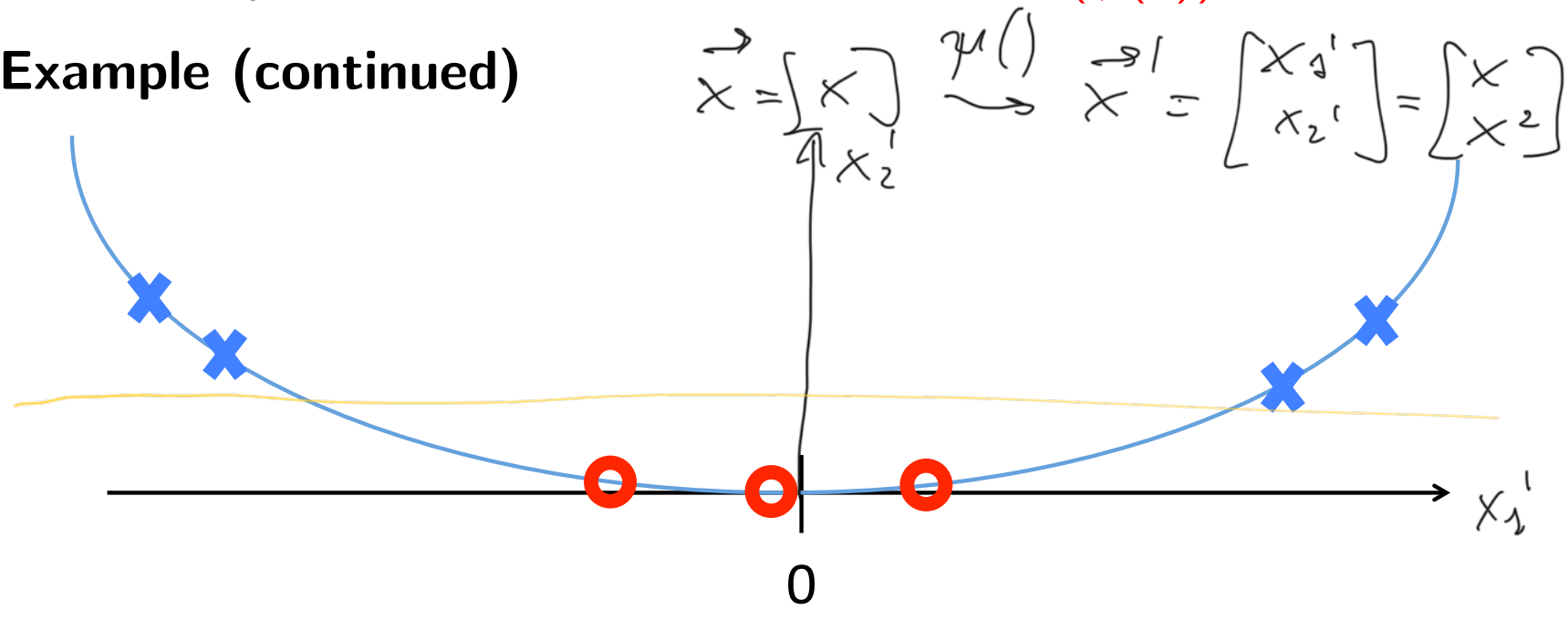
Example



We can:

- apply a nonlinear transformation $\psi()$ to each point in training set S first: $S' = ((\psi(\mathbf{x}_1), y_1), \ldots, (\psi(\mathbf{x}_m), y_m))$;
- learn a linear predictor $\hat{h}$ in the transformed space using S';
- make prediction for a new instance $\mathbf{x}$ as $\hat{h}(\psi(\mathbf{x}))$

Example (continued)



Kernel Trick for SVM

What if we want to apply a nonlinear transformation before using SVM?

Let $\psi()$ be the nonlinear transformation

i) Given training set S: obtain training S'

$$S = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}$$

$$\downarrow$$

$$S' = \{(\psi(\vec{x}_1), y_1), \ldots, (\psi(\vec{x}_m), y_m)\}$$

ii) Learn a model with SVM using S', let h_{SVM} be the model we learn

iii) Given $\vec{x} \in X$, the prediction is $h_{SVM}(\psi(\vec{x}))$

Kernel Trick for SVM

What if we want to apply a nonlinear transformation before using SVM?

Let $\psi()$ be the nonlinear transformation

Considering the dual formulation $\Rightarrow$ we only need to be able to compute $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ for some $\mathbf{x}$,$\mathbf{x}'$.

Definition

A *kernel function* is a function of the type:

$$K_\psi(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$$

where $\psi(\mathbf{x})$ is a transformation of $\mathbf{x}$.

Kernel Trick for SVM

What if we want to apply a nonlinear transformation before using SVM?

Let $\psi()$ be the nonlinear transformation

Considering the dual formulation $\Rightarrow$ we only need to be able to compute $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ for some $\mathbf{x}$,$\mathbf{x}'$.

Definition

A *kernel function* is a function of the type:

$$K_\psi(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$$

where $\psi(\mathbf{x})$ is a transformation of $\mathbf{x}$.

Intuition: we can think of K_ψ as specifying *similarity* between instances and of ψ as mapping the domain set $\mathcal{X}$ into a space where these similarities are realized as dot products.

Kernel Trick for SVM(2)

$$K_\psi(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$$

It seems that to compute $K_\psi(\mathbf{x}, \mathbf{x}')$ requires to be able to compute $\psi(\mathbf{x})$...

Not always... sometimes we can compute $K_\psi(\mathbf{x}, \mathbf{x}')$ without computing $\psi(\mathbf{x})$!

Kernel: Example

Consider $\mathbf{x} \in \mathbb{R}^d$

$$\psi(\mathbf{x}) = (1, x_1, x_2, \ldots, x_d, x_1x_1, x_1x_2, x_1x_3, \ldots, x_dx_d)^T$$

What is the dimension of $\psi(\vec{x})$? $1 + d + d^2$

Kernel: Example

Consider $\mathbf{x} \in \mathbb{R}^d$

$$\psi(\mathbf{x}) = (1, x_1, x_2, \ldots, x_d, x_1x_1, x_1x_2, x_1x_3, \ldots, x_dx_d)^T$$

The dimension of $\psi(\mathbf{x})$ is $1 + d + d^2$.

$$K_\psi(\vec{x}, \vec{x'}) = \left\langle \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \\ x_1x_1 \\ \vdots \\ x_dx_d \end{bmatrix}, \begin{bmatrix} 1 \\ x_1' \\ x_2' \\ \vdots \\ x_d' \\ x_1'x_1' \\ \vdots \\ x_d'x_d' \end{bmatrix} \right\rangle = 1 + (x_1x_1' + x_2x_2' + \ldots + x_dx_d') + x_1x_1x_1'x_1' + x_1x_2x_1'x_2' + \ldots + x_dx_dx_d'x_d'$$

$$= 1 + \sum_{i=1}^{d} x_i x_i' + \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j x_i' x_j'$$

$$\left(\sum_{i=1}^{d} x_i x_i'\right)\left(\sum_{j=1}^{d} x_j x_j'\right)$$

$$\langle \vec{x}, \vec{x'} \rangle^2 = \langle \vec{x}, \vec{x'} \rangle \cdot \langle \vec{x}, \vec{x'} \rangle$$

$$\langle \vec{x}, \vec{x'} \rangle$$

Kernel: Example

Consider $\mathbf{x} \in \mathbb{R}^d$

$$\psi(\mathbf{x}) = (1, x_1, x_2, \ldots, x_d, x_1x_1, x_1x_2, x_1x_3, \ldots, x_dx_d)^T$$

The dimension of $\psi(\mathbf{x})$ is $1 + d + d^2$.

$$\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = 1 + \sum_{i=1}^{d} x_i x_i' + \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j x_i' x_j'$$

Note that

$$\sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j x_i' x_j' = \left(\sum_{i=1}^{d} x_i x_i' \right) \left(\sum_{j=1}^{d} x_j x_j' \right) = \left(\langle \mathbf{x}, \mathbf{x}' \rangle \right)^2$$

therefore

$$K_\psi(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = 1 + \langle \mathbf{x}, \mathbf{x}' \rangle + \left(\langle \mathbf{x}, \mathbf{x}' \rangle \right)^2$$

We have:

$$\psi(\mathbf{x}) = (1, x_1, x_2, \ldots, x_d, x_1x_1, x_1x_2, x_1x_3, \ldots, x_dx_d)^T$$

$$K_\psi(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = 1 + \langle \mathbf{x}, \mathbf{x}' \rangle + \left(\langle \mathbf{x}, \mathbf{x}' \rangle\right)^2$$

Computation of $\psi(\vec{x})$: how many operations? $\Theta(d^2)$
(starting from $\vec{x}$)

Computation of $K_\psi(\vec{x}, \vec{x}')$: how many operations?
If I first compute $\psi(\vec{x})$ and $\psi(\vec{x}')$, and then their $\langle , \rangle$: $\Theta(d^2)$

If I use $K_\psi(\vec{x}, \vec{x}') = 1 + \langle \vec{x}, \vec{x}' \rangle + \left(\langle \vec{x}, \vec{x}' \rangle\right)^2$: $\Theta(d)$

$\Theta(1)$ $\Theta(d)$ $\rightsquigarrow$ $\Theta(1)$

We have:

$$\psi(\mathbf{x}) = (1, x_1, x_2, \ldots, x_d, x_1x_1, x_1x_2, x_1x_3, \ldots, x_dx_d)^T$$

$$K_\psi(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle = 1 + \langle \mathbf{x}, \mathbf{x}' \rangle + \left(\langle \mathbf{x}, \mathbf{x}' \rangle \right)^2$$

Observation

Computing $\psi(\mathbf{x})$ requires $\Theta\left(d^2\right)$ time; computing $K_\psi(\mathbf{x}, \mathbf{x}')$ from the last formula requires $\Theta\left(d\right)$ time

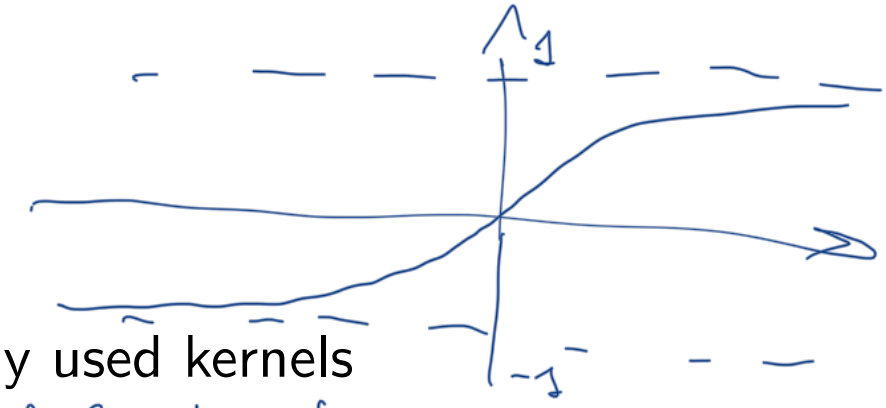
When $K_\psi(\mathbf{x}, \mathbf{x}')$ is efficiently computable, we don't need to explicitly compute $\psi(\mathbf{x})$
$\Rightarrow$ *kernel trick*

Some Kernels

The following are the most commonly used kernels

- linear kernel: $\psi(\mathbf{x}) = \mathbf{x}$ (standard SVM)

Some Kernels



The following are the most commonly used kernels

- linear kernel: $\psi(\mathbf{x}) = \mathbf{x}$
- sigmoid: $K(\mathbf{x}, \mathbf{x}') = \tanh(\gamma\langle \mathbf{x}, \mathbf{x}' \rangle + \zeta)$ (for γ, $\zeta > 0$)

hyperbolic tangent

Some Kernels

The following are the most commonly used kernels

- linear kernel: $\psi(\mathbf{x}) = \mathbf{x}$
- sigmoid: $K(\mathbf{x}, \mathbf{x}') = \tanh(\gamma\langle \mathbf{x}, \mathbf{x}' \rangle + \zeta)$ (for γ, $\zeta > 0$)
- degree-Q polynomial kernel
- Gaussian-radial basis function (RBF) kernel

Degree-Q polynomial kernel

Definition

For given constants $\gamma > 0, \zeta > 0$ and for $Q \in \mathbb{N}$, the *degree-Q polynomial kernel* is

$$K(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \langle \mathbf{x}, \mathbf{x}' \rangle)^Q$$

time? (1) $O(d)$ (2) $O(Q+d)$ (3) $O(d + \log Q)$

Example

For $Q = 2$:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$\psi(\mathbf{x}) = [\zeta, \sqrt{2\zeta\gamma}x_1, \sqrt{2\zeta\gamma}x_2, \ldots, \sqrt{2\zeta\gamma}x_d, \gamma x_1 x_1, \gamma x_1 x_2, \ldots, \gamma x_d x_d]^T \in \mathbb{R}^{1+d+d^2}$$

In general: $\psi(\vec{x}) \in \mathbb{R}^{\binom{d+Q}{Q}}$

Gaussian-RBF Kernel

Definition

For a given constant $\gamma > 0$ the *Gaussian-RBF kernel* is

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma ||\mathbf{x}-\mathbf{x}'||^2}$$

What is $\psi(\mathbf{x})$? Assume $\gamma = 1$ and $\mathbf{x} = x \in \mathbb{R}$ for simplicity, then

$$\begin{aligned} K(x, x') &= e^{-||x-x'||^2} \\ &= e^{-x^2} e^{2xx'} e^{-(x')^2} \\ &= e^{-x^2} \underbrace{\left(\sum_{k=0}^{+\infty} \frac{2^k (x)^k (x')^k}{k!} \right)}_{\text{Taylor expansion of } e^{2xx'} = \sum_{k=0}^{+\infty} \frac{(2xx')^k}{k!}} e^{-(x')^2} \end{aligned}$$

Gaussian-RBF Kernel

Definition

For a given constant $\gamma > 0$ the *Gaussian-RBF kernel* is

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma ||\mathbf{x} - \mathbf{x}'||^2}$$

What is $\psi(\mathbf{x})$? Assume $\gamma = 1$ and $\mathbf{x} = x \in \mathbb{R}$ for simplicity, then

$$\begin{aligned} K(x, x') &= e^{-||x - x'||^2} \\ &= e^{-x^2} e^{2xx'} e^{-(x')^2} \\ &= e^{-x^2} \left(\sum_{k=0}^{+\infty} \frac{2^k (x)^k (x')^k}{k!} \right) e^{-(x')^2} = \langle \psi(x), \psi(x') \rangle \end{aligned}$$

$$\Rightarrow \psi(x) = e^{-x^2} \left(1, \sqrt{\frac{2}{1!}} x, \sqrt{\frac{2^2}{2!}} x^2, \sqrt{\frac{2^3}{3!}} x^3, \dots \right)^T$$

Gaussian-RBF Kernel

Definition

For a given constant $\gamma > 0$ the *Gaussian-RBF kernel* is

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma ||\mathbf{x} - \mathbf{x}'||^2}$$

What is $\psi(\mathbf{x})$? Assume $\gamma = 1$ and $\mathbf{x} = x \in \mathbb{R}$ for simplicity, then

$$\begin{aligned} K(x, x') &= e^{-||x - x'||^2} \\ &= e^{-x^2} e^{2xx'} e^{-(x')^2} \\ &= e^{-x^2} \left(\sum_{k=0}^{+\infty} \frac{2^k (x)^k (x')^k}{k!} \right) e^{-(x')^2} \end{aligned}$$

$$\Rightarrow \psi(x) = e^{-x^2} \left(1, \sqrt{\frac{2}{1!}} x, \sqrt{\frac{2^2}{2!}} x^2, \sqrt{\frac{2^3}{3!}} x^3, \dots \right)^T$$

$\Rightarrow \psi(x)$ has infinite number of dimensions!

Choice of Kernel

Notes

- polynomial kernel: usually used with $Q \leq 10$
- Gaussian-RBF kernel: usually $\gamma \in [0, 1]$
- many other choices are possible!

Mercer's condition

$K(\mathbf{x}, \mathbf{x}')$ is a valid kernel function if and only if the kernel matrix

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) \dots & K(\mathbf{x}_1, \mathbf{x}_m) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) \dots & K(\mathbf{x}_2, \mathbf{x}_m) \\ \vdots & \vdots & \vdots \\ K(\mathbf{x}_m, \mathbf{x}_1) & K(\mathbf{x}_m, \mathbf{x}_2) \dots & K(\mathbf{x}_m, \mathbf{x}_m) \end{bmatrix}$$

is always symmetric positive semi-definite for any given $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$.

Support Vector Machines for Regression

$y_i \in \mathbb{R}$

SVMs can be also used for regression. The function to be minimized will be

l_2-regularization ← $\frac{\lambda}{2}||\mathbf{w}||^2 + \sum_{i=1}^{m} V_\varepsilon(y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle - b)$ ← loss function; observed value; predicted value (by model $\vec{w}, b$)

where

$$V_\varepsilon(r) = \begin{cases} 0 & \text{if } |r| < \varepsilon \\ |r| - \varepsilon & \text{otherwise} \end{cases}$$

One can prove that the solution has the form:

$$\mathbf{w} = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i)\mathbf{x}_i$$

$\vec{x}_i$ in the training set ($\vec{x}_i \in S$)

and that the final model produced in output is

the prediction for $\vec{x} \in \mathbb{R}^d$ — training set points and $\vec{x}$

$$h(\mathbf{x}) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i)\langle \mathbf{x}_i, \mathbf{x} \rangle + b$$

where $\alpha_i^*, \alpha_i \geq 0$ and are the solution to a suitable QP.

Definition

Support vector: $\mathbf{x}_i$ such that $\alpha_i^* - \alpha_i \neq 0$

One can define kernels, similarly to SVM for classification.

Exercise 4

Assuming we have the following dataset ($x_i \in \mathbb{R}^2$) and by solving the SVM for classification we get the corresponding optimal dual variables:

i	x_i^T	y_i	α_i^*
1	[0.2 -1.4]	-1	0
2	[-2.1 1.7]	1	0
3	[0.9 1]	1	0.5
4	[-1 -3.1]	-1	0
5	[-0.2 -1]	-1	0.25
6	[-0.2 1.3]	1	0
7	[2.0 -1]	-1	0.25
8	[0.5 2.1]	1	0

Answer to the following:

(A) What are the support vectors?

(B) Draw a schematic picture reporting the data points (approximately) and the optimal separating hyperplane, and mark the support vectors. Would it be possible, by moving only two data points, to obtain the SAME separating hyperplane with only 2 support vectors? If so, draw the modified configuration (approximately).

Bibliography [UML]

SVM: Chapter 15

- no sections 15.1.2, 15.2.1, 15.2.2,15.2.3,

Kernels: Chapter 16

- no section 16.3