Machine Learning

Support Vector Machines

Fabio Vandin

November 29^{th}, 2022

Linearly Separable Training Set

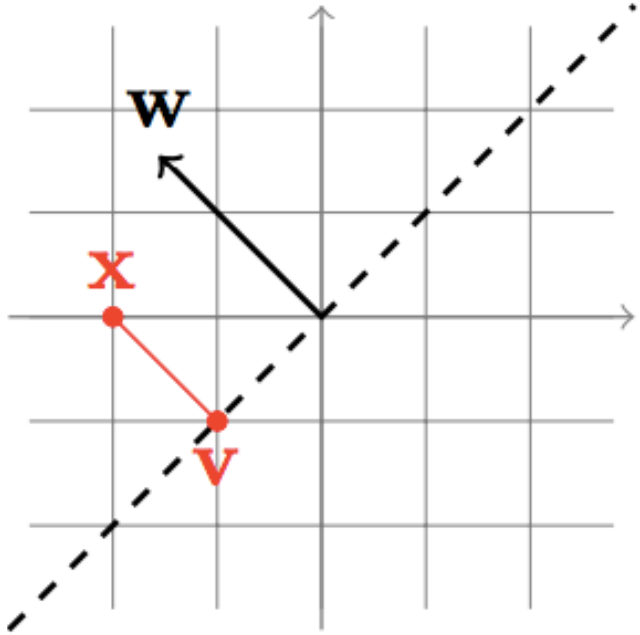
Training set $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ is *linearly separable* if there exists a halfspace $(\mathbf{w}, b)$ such that $y_i = \text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ for all $i = 1, \ldots, m$.

Equivalent to:

$$\forall i = 1, \ldots, m : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$$

Informally: *margin* of a separating hyperplane is its minimum distance to an example in the training set S

Separating Hyperplane and Margin



Given hyperplane defined by $L = \{\mathbf{v} : \langle \mathbf{w}, \mathbf{v} \rangle + b = 0\}$, and given $\mathbf{x}$, the distance of $\mathbf{x}$ to L is

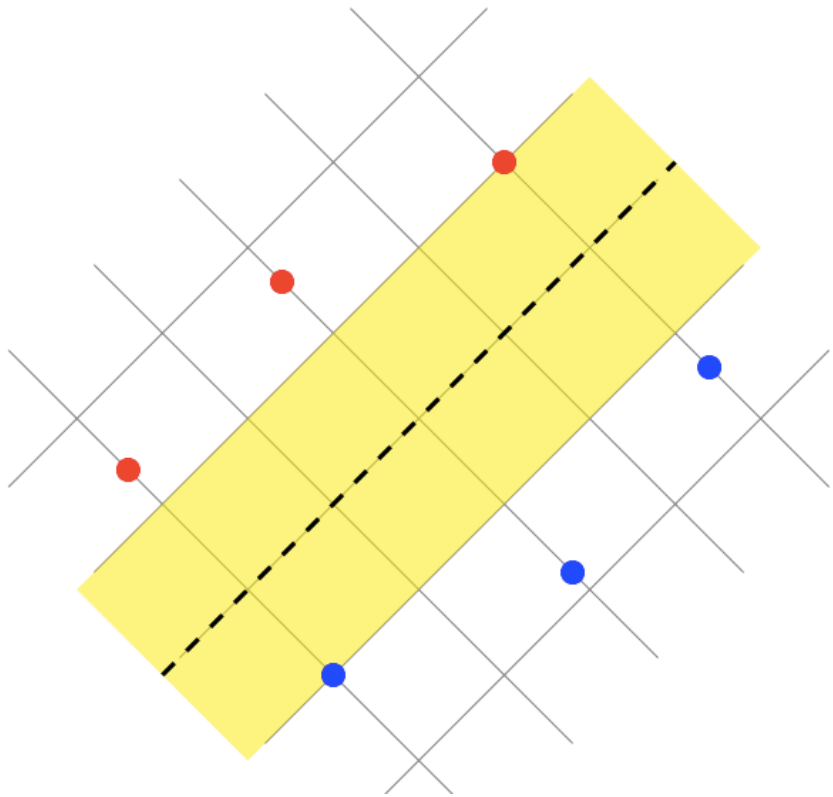
$$d(\mathbf{x}, L) = \min\{||\mathbf{x} - \mathbf{v}|| : \mathbf{v} \in L\}$$

Claim: if $||\mathbf{w}|| = 1$ then $d(\mathbf{x}, L) = |\langle \mathbf{w}, \mathbf{x} \rangle + b|$ (Proof: Claim 15.1 [UML])

Margin and Support Vectors

The *margin* of a separating hyperplane is the distance of the closest example in training set to it. If $||\mathbf{w}|| = 1$ the margin is:

$$\min_{i \in \{1,\ldots,m\}} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b|$$



The closest examples are called *support vectors*

Support Vector Machine (SVM)

Hard-SVM: seek for the separating hyperplane with largest margin (only for linearly separable data)

Computational problem:

$$\arg \max_{(\mathbf{w},b):||\mathbf{w}||=1} \min_{i\in\{1,\ldots,m\}} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b|$$

subject to $\forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$

Equivalent formulation (due to separability assumption):

$$\arg \max_{(\mathbf{w},b):||\mathbf{w}||=1} \min_{i\in\{1,\ldots,m\}} y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$$

Hard-SVM: Quadratic Programming Formulation

- **input**: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
- **solve**:

$$(\mathbf{w}_0, b_0) = \arg \min_{(\mathbf{w}, b)} ||\mathbf{w}||^2$$

  subject to $\forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$
- **output**: $\hat{\mathbf{w}} = \frac{\mathbf{w}_0}{||\mathbf{w}_0||}, \hat{b} = \frac{b_0}{||\mathbf{w}_0||}$

Proposition

The output of algorithm above is a solution to the *Equivalent Formulation* in the previous slide.

How do we get a solution? Quadratic optimization problem: objective is convex quadratic function, constraints are linear inequalities $\Rightarrow$ Quadratic Programming solvers!

Equivalent Formulation and Support Vectors

Equivalent formulation (homogeneous halfspaces): assume first component of $\mathbf{x} \in \mathcal{X}$ is 1, then

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

"Support Vectors" $=$ vectors at minimum distance from $\mathbf{w}_0$

The support vectors are the only ones that matter for defining $\mathbf{w}_0$!

Proposition

Let $\mathbf{w}_0$ be as above. Let $I = \{i : |\langle \mathbf{w}_0, \mathbf{x}_i \rangle| = 1\}$. Then there exist coefficients $\alpha_1, \ldots, \alpha_m$ such that

$$\mathbf{w}_0 = \sum_{i \in I} \alpha_i \mathbf{x}_i$$

"Support vectors" $= \{\mathbf{x}_i : i \in I\}$

Note: Solving Hard-SVM is equivalent to find α_i for $i = 1, \ldots, m$, and $\alpha_i \neq 0$ only for support vectors

Soft-SVM

Hard-SVM works if data is linearly separable.

What if data is not linearly separable? $\Rightarrow$ soft-SVM

Idea: modify constraints of Hard-SVM to allow for some violation, but take into account violations into objective function

Soft-SVM Constraints

Hard-SVM constraints:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

Soft-SVM constraints:

- *slack variables*: $\xi_1, \ldots, \xi_m \geq 0 \Rightarrow$ vector $\boldsymbol{\xi}$
- for each $i = 1, \ldots, m$: $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$
- ξ_i: how much constraint $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ is violated

Soft-SVM minimizes combinations of

- norm of $\mathbf{w}$
- average of ξ_i

Tradeoff among two terms is controlled by a parameter
$\lambda \in \mathbb{R}, \lambda > 0$

Soft-SVM: Optimization Problem

- **input**: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$, parameter $\lambda > 0$
- **solve**:

$$\min_{\mathbf{w},b,\xi} \left(\lambda ||\mathbf{w}||^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right)$$

  subject to $\forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$
- **output**: $\mathbf{w}, b$

Equivalent formulation: consider the *hinge loss*

$$\ell^{\text{hinge}}((\mathbf{w}, b), (\mathbf{x}, y)) = \max\{0, 1 - y(\langle \mathbf{w}, \mathbf{x} \rangle + b)\}$$

Given $(\mathbf{w}, b)$ and a training S, the empirical risk $L_S^{\text{hinge}}((\mathbf{w}, b))$ is

$$L_S^{\text{hinge}}((\mathbf{w}, b)) = \frac{1}{m} \sum_{i=1}^{m} \ell^{\text{hinge}}((\mathbf{w}, b), (\mathbf{x}_i, y_i))$$

Soft-SVM as RLM

Soft-SVM: solve

$$\min_{\mathbf{w},b,\xi} \left(\lambda ||\mathbf{w}||^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right)$$

subject to $\forall i : y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

Equivalent formulation with hinge loss:

$$\min_{\mathbf{w},b} \left(\lambda ||\mathbf{w}||^2 + L_S^{\text{hinge}}(\mathbf{w}, b) \right)$$

that is

$$\min_{\mathbf{w},b} \left(\lambda ||\mathbf{w}||^2 + \frac{1}{m} \sum_{i=1}^{m} \ell^{\text{hinge}}((\mathbf{w}, b), (\mathbf{x}_i, y_i)) \right)$$

Note:

- $\lambda ||\mathbf{w}||^2$: ℓ_2 regularization
- $L_S^{\text{hinge}}(\mathbf{w}, b)$: empirical risk for hinge loss

Soft-SVM: Solution

We need to solve:

$$\min_{\mathbf{w},b} \left(\lambda ||\mathbf{w}||^2 + \frac{1}{m} \sum_{i=1}^{m} \ell^{\text{hinge}}((\mathbf{w}, b), (\mathbf{x}_i, y_i)) \right)$$

where

$$\ell^{\text{hinge}}((\mathbf{w}, b), (\mathbf{x}, y)) = \max\{0, 1 - y(\langle \mathbf{w}, \mathbf{x} \rangle + b)\}$$

How?

- standard solvers for optimization problems
- **Stochastic Gradient Descent**

SGD for Solving Soft-SVM

We want to solve

$$\min_{\mathbf{w}} \left(\frac{\lambda}{2} ||\mathbf{w}||^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}_i \rangle\} \right)$$

Note: it's standard to add a $\frac{1}{2}$ in the regularization term to simplify some computations.

SGD algorithm:

$\boldsymbol{\theta}^{(1)} \leftarrow \mathbf{0}$;

for $t \leftarrow 1$ *to* T **do**

  $\eta^{(t)} \leftarrow \frac{1}{\lambda t}$; $\mathbf{w}^{(t)} \leftarrow \eta^{(t)} \boldsymbol{\theta}^{(t)}$;

  choose i uniformly at random from $\{1, \ldots, m\}$;

  if $y_i \langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_i \rangle < 1$ **then** $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} + y_i \mathbf{x}_i$;

  else $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)}$;

return $\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}^{(t)}$;

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*:

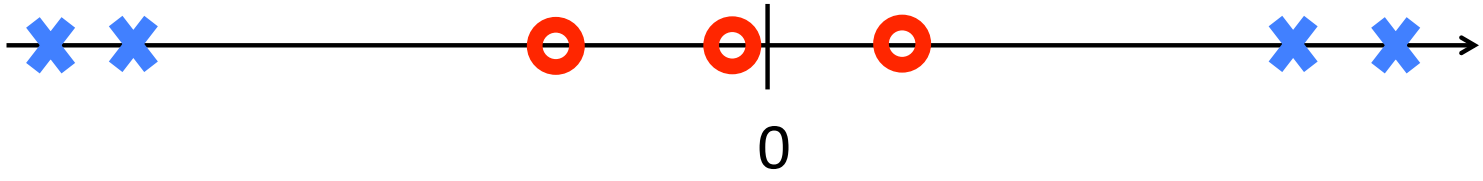
$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq \mathbf{0}} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \right)$$

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)!

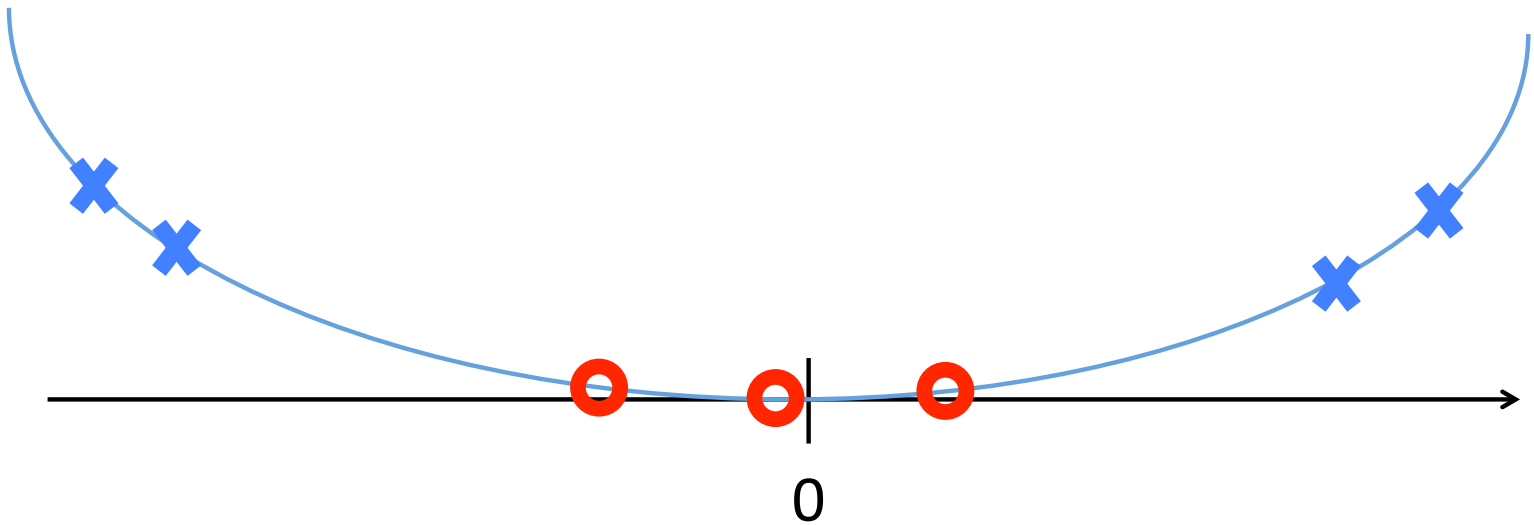
Example



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

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

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

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:

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

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

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: $\psi(\mathbf{x}) = \mathbf{x}$
- 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

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

Example

For $Q = 2$:

$$\psi(\mathbf{x}) = [\zeta, \sqrt{2\zeta\gamma}x_1, \sqrt{2\zeta\gamma}x_2, \ldots, \sqrt{2\zeta\gamma}x_d, \gamma x_1 x_1, \gamma x_1 x_2, \ldots, \gamma x_d x_d]^T$$

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

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

$$\frac{\lambda}{2}||\mathbf{w}||^2 + \sum_{i=1}^{m} V_\varepsilon(y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle - b)$$

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:

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

and that the final model produced in output is

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

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