Pattern Analysis and Recognition

Lecture 9: Support Vector Machine, Kernels

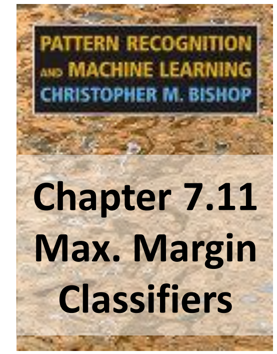
Resources

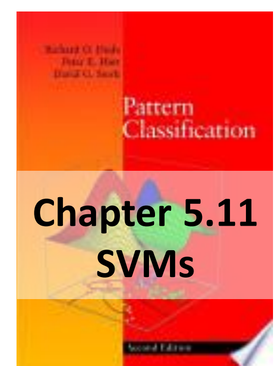
See the following sources for further information:

C. Bishop, “Pattern Recognition and Machine Learning”, Springer, 2006
Some related material available:
http://research.microsoft.com/en-us/um/people/cmbishop/prml/index.htm

D. MacKay, “Information Theory, Inference and Learning Algorithms”, Cambridge University Press, 2003.
Book available online:
http://www.inference.phy.cam.ac.uk/mackay/

R.O. Duda, P.E. Hart, D.G. Stork, “Pattern Classification”, Wiley & Sons, 2000
Have a look inside at selected chapters:
http://books.google.es/books/about/Pattern_Classification.html?id=Br33IRC3PkQC&redir_esc=y

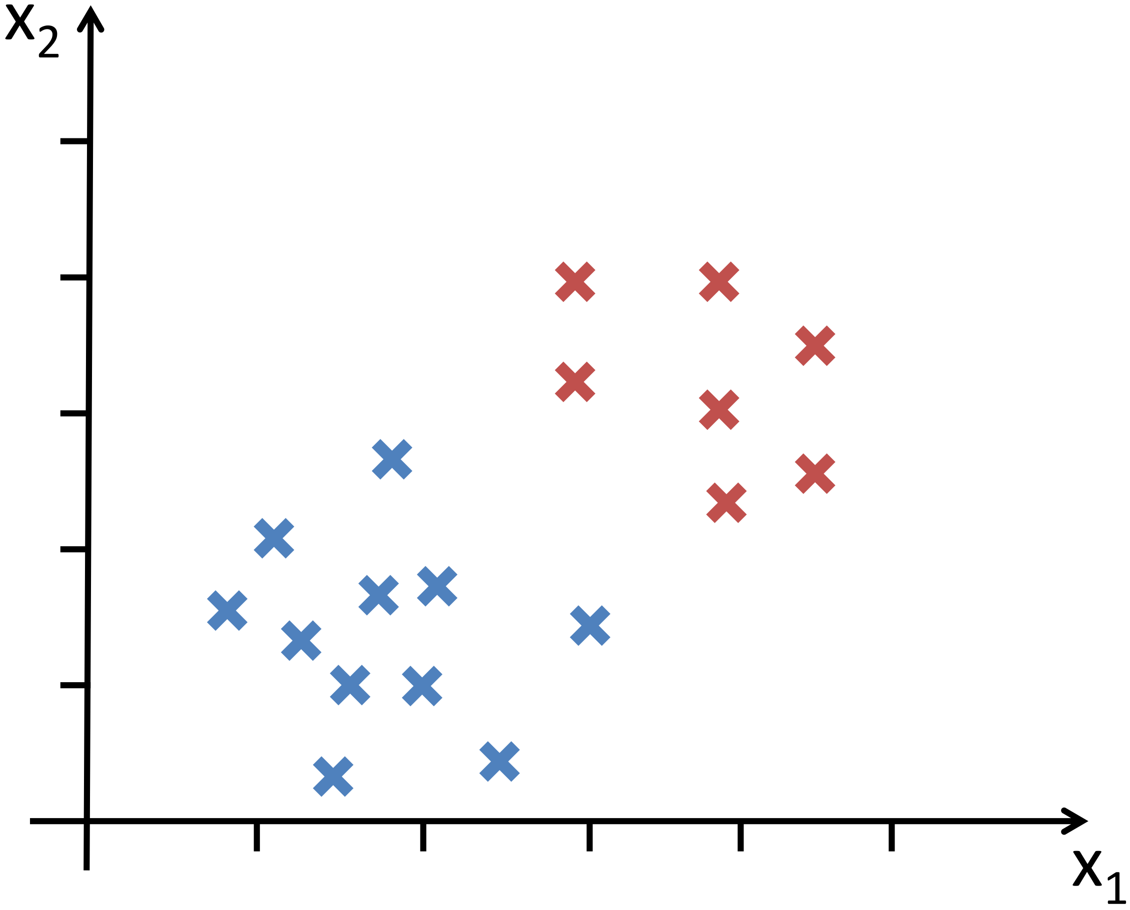




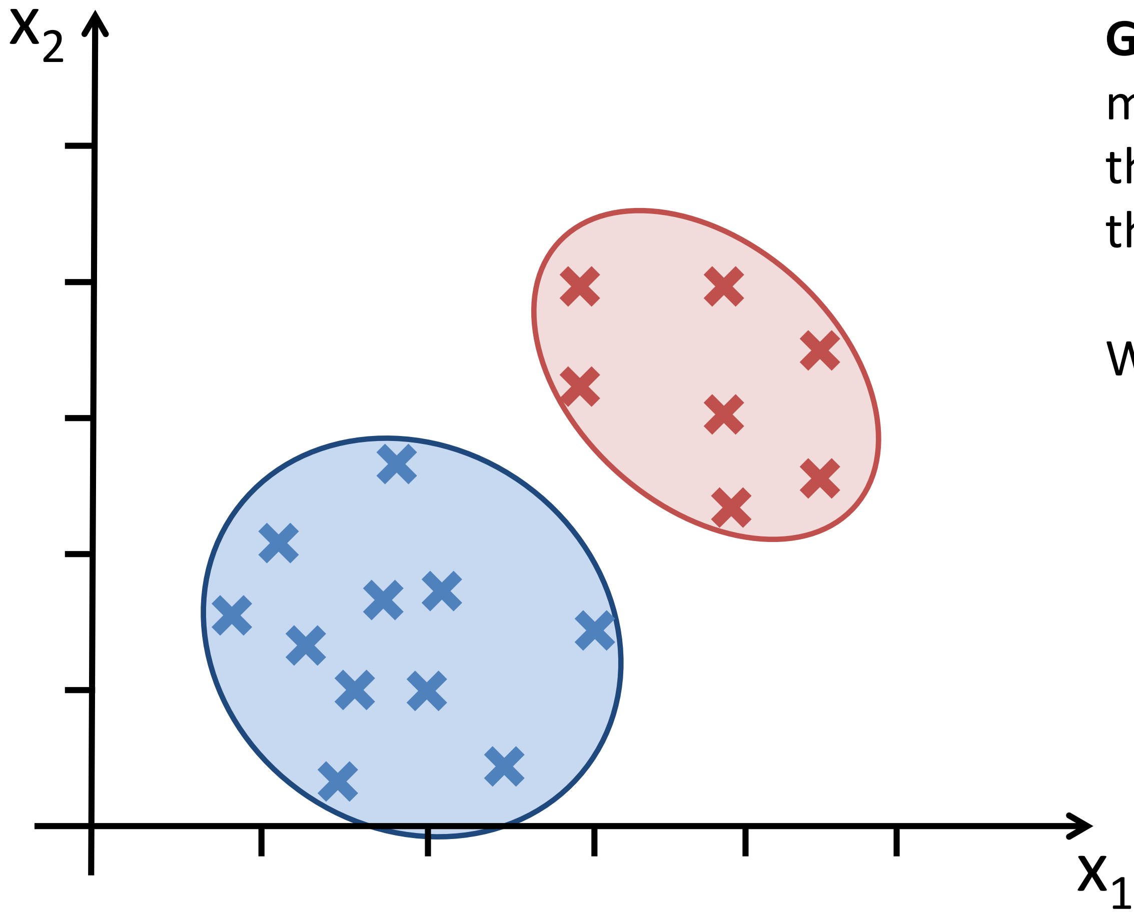
A pictorial introduction

SUPPORT VECTOR MACHINES

Classification



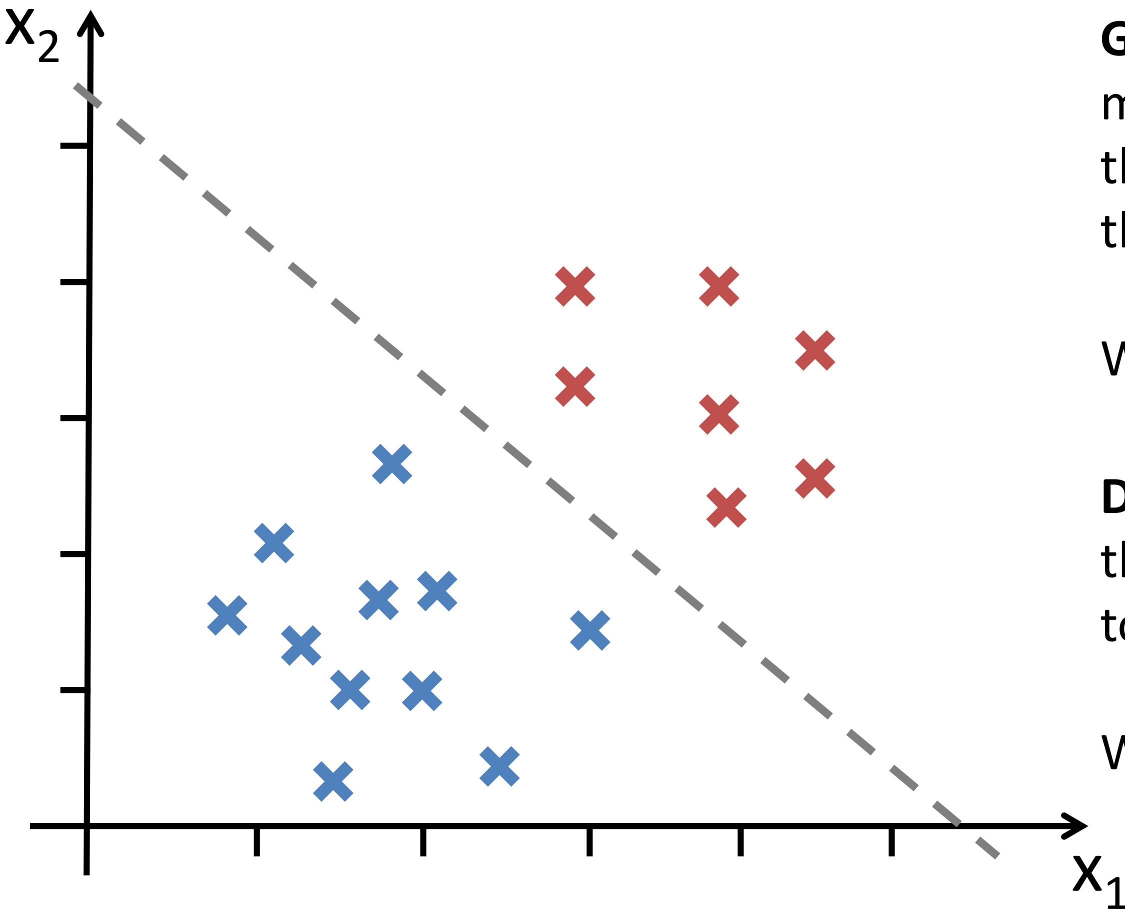
Classification



Generative models: try model the class, e.g. estimate the PDF that gave rise to these points.

We learn: p(x | y)

Classification



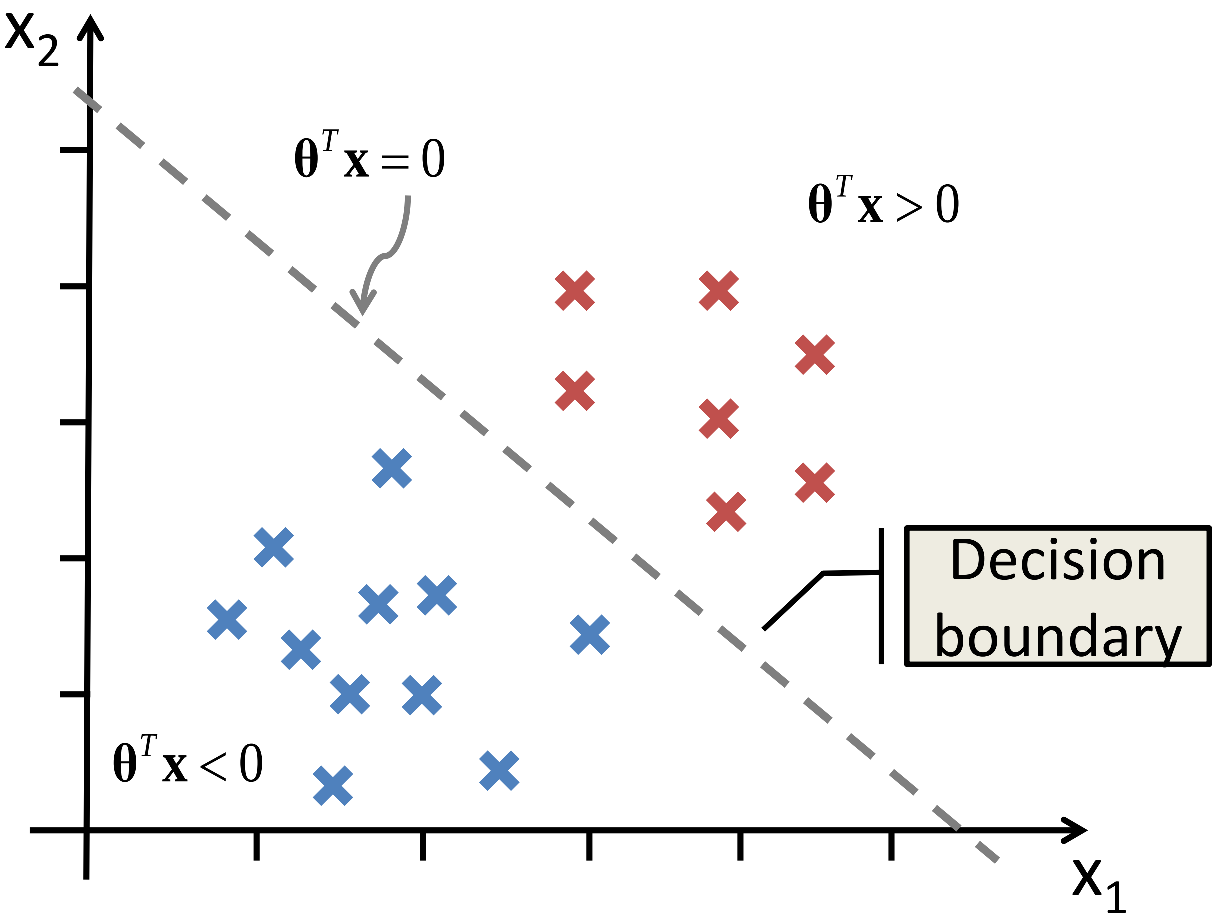
Generative models: try model the class, e.g. estimate the PDF that gave rise to these points.

We learn: p(x | y)

Discriminative Models: use the data to find the best way to separate the classes.

We learn: p(y | x; θ)

Linearly Separable Classes



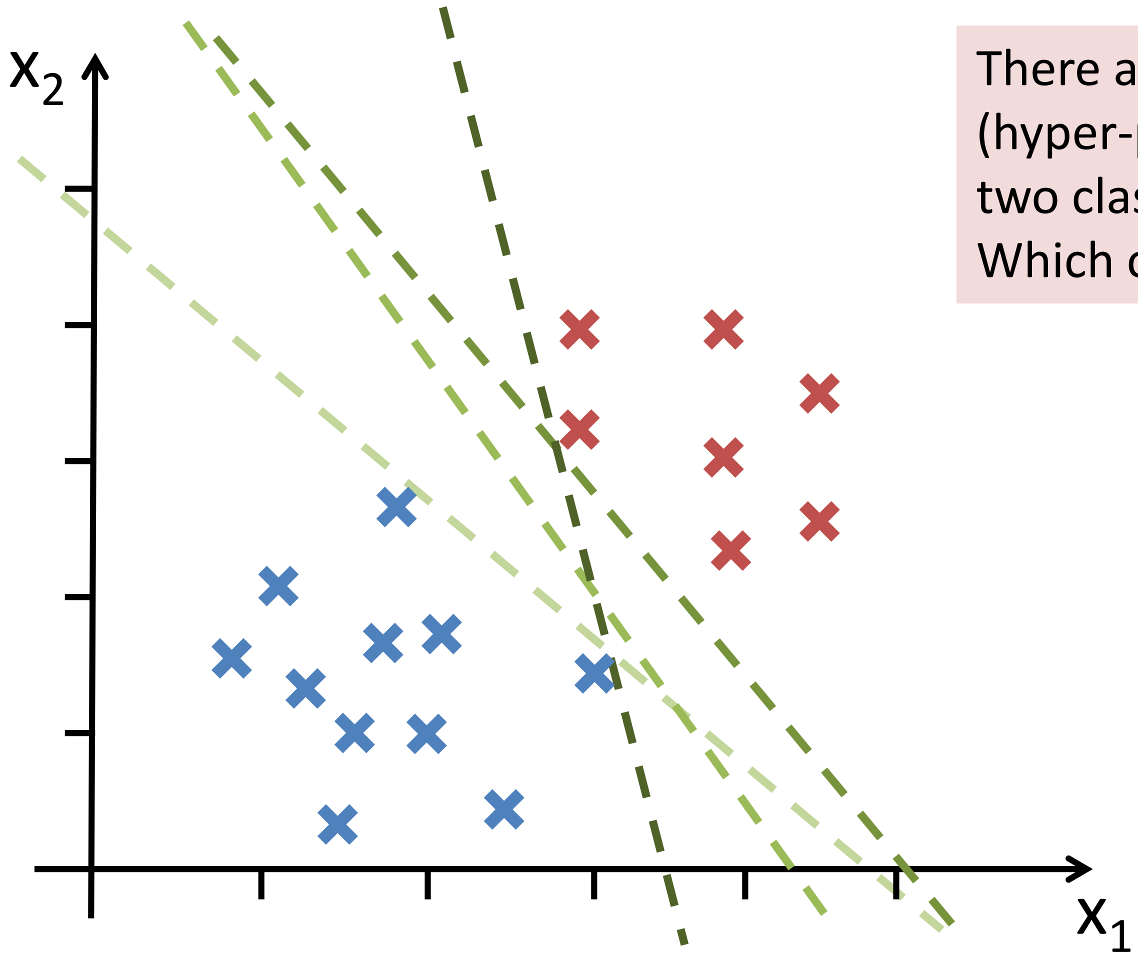
$\boldsymbol{\theta}^T\mathbf{x} > 0 \rightarrow$ ✖ (red)

$\boldsymbol{\theta}^T\mathbf{x} < 0 \rightarrow$ ✖ (blue)

$$\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_n]^T$$

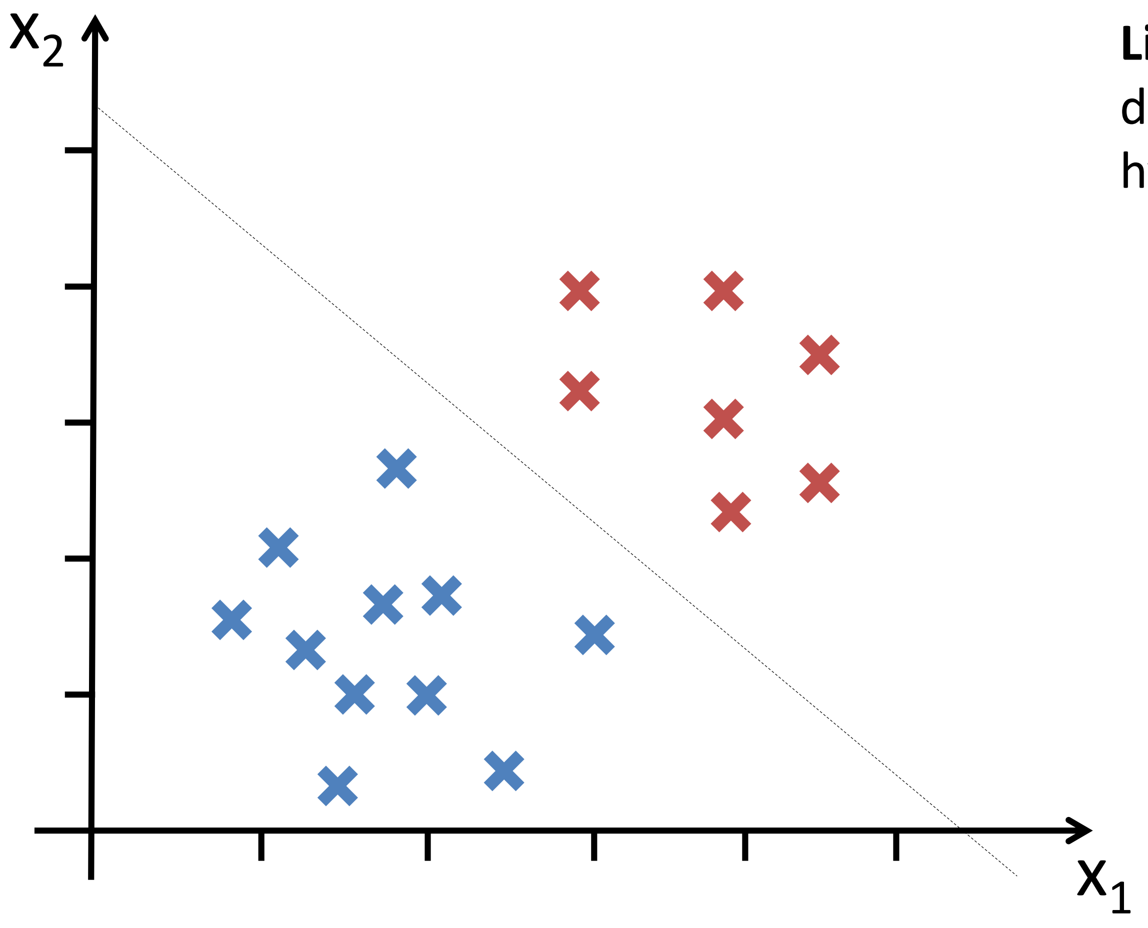
$$\mathbf{x} = [x_0 = 1, x_1, \ldots, x_n]^T$$

Linearly Separable Classes



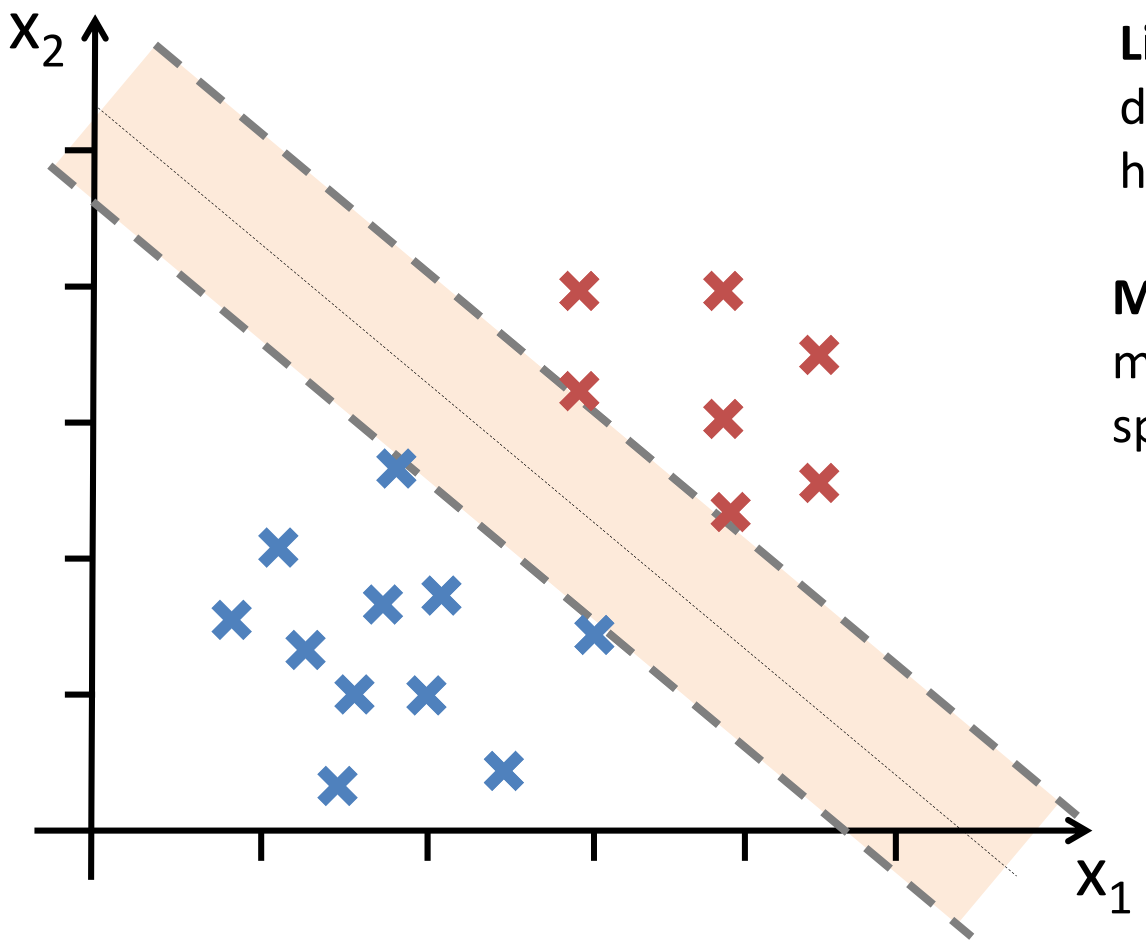
There are multiple valid solutions (hyper-planes that separate the two classes).
Which one shall we choose?

Support Vector Machines



Linear classifiers: the decision boundary is an hyper-plane

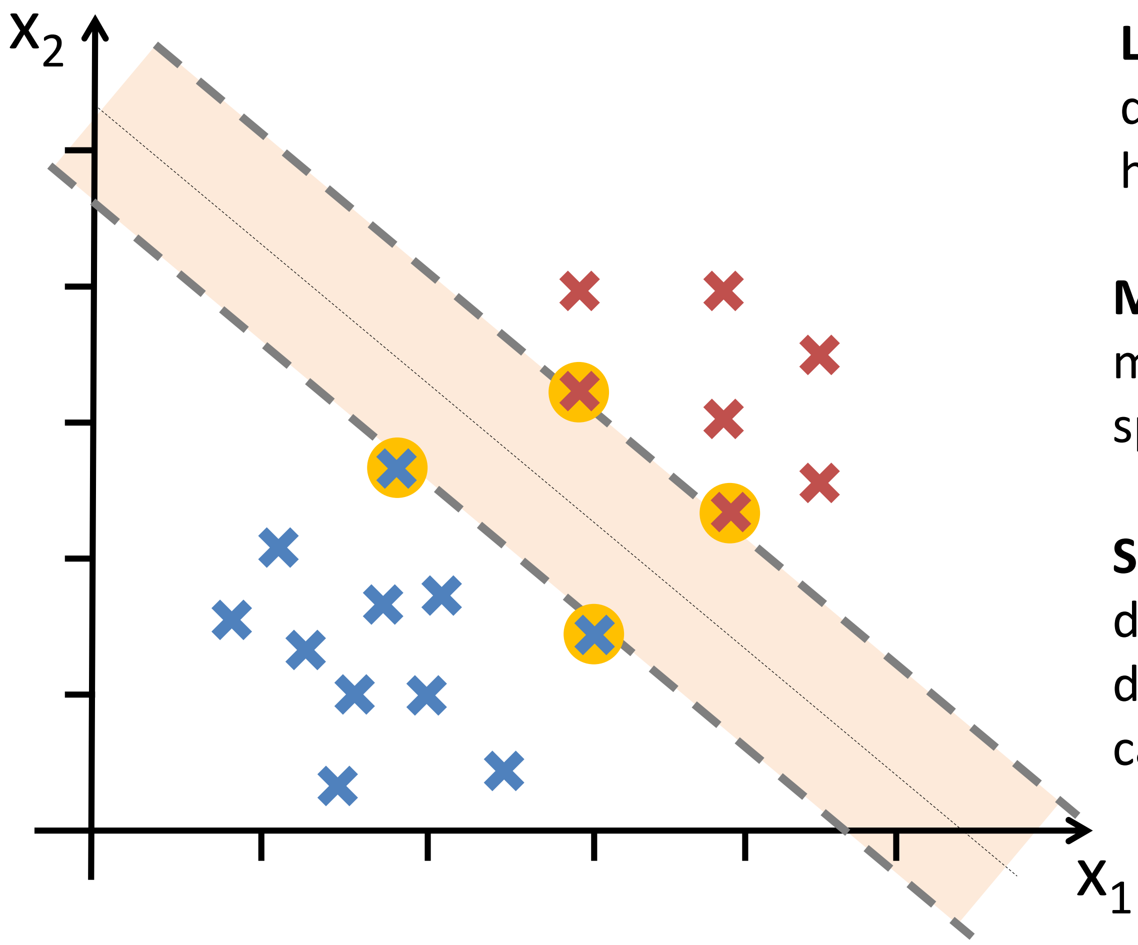
Support Vector Machines



Linear classifiers: the decision boundary is an hyper-plane

Maximal margin: the solution maximises the margin (empty space) between the classes

Support Vector Machines

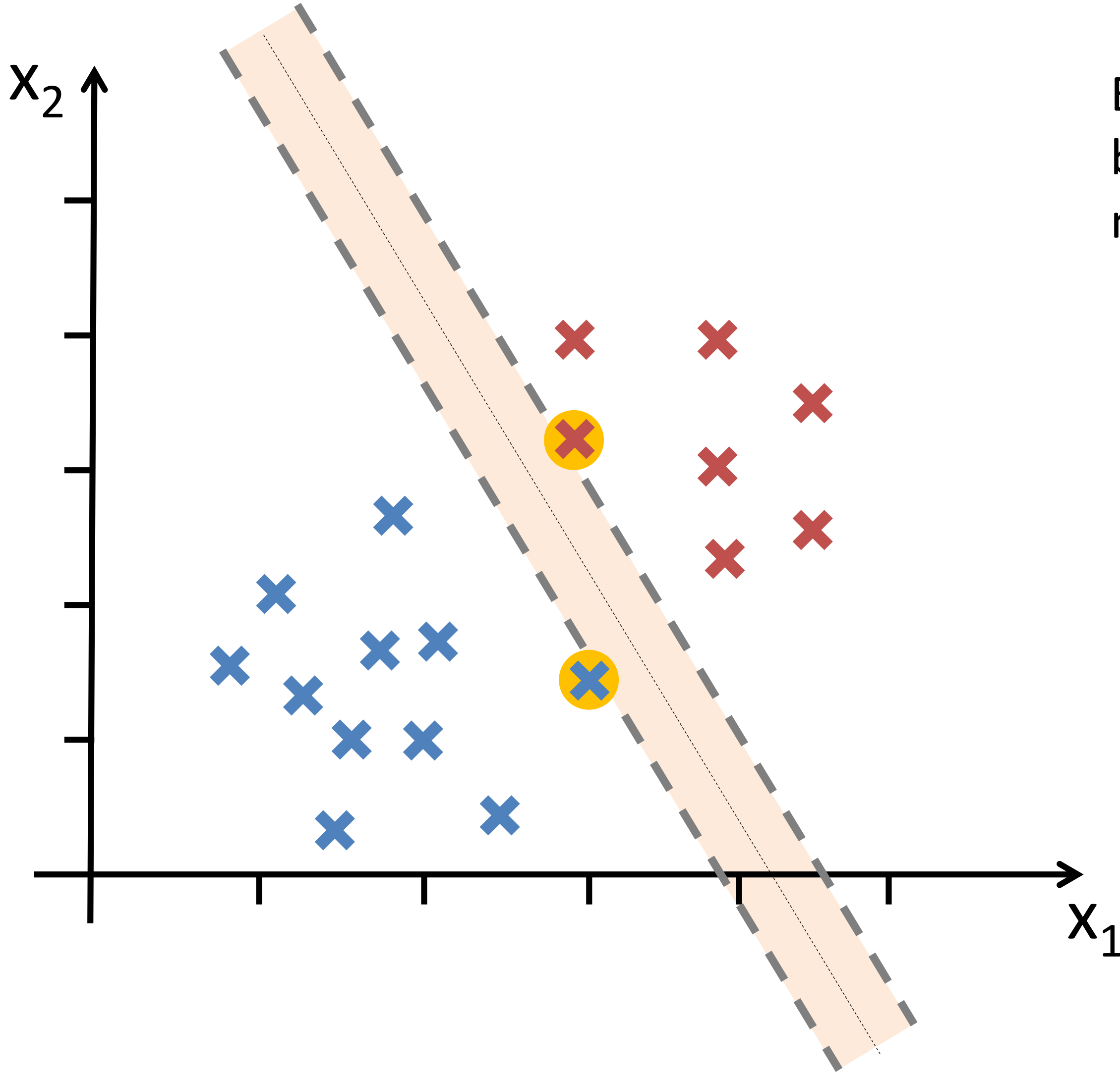


Linear classifiers: the decision boundary is an hyper-plane

Maximal margin: the solution maximises the margin (empty space) between the classes

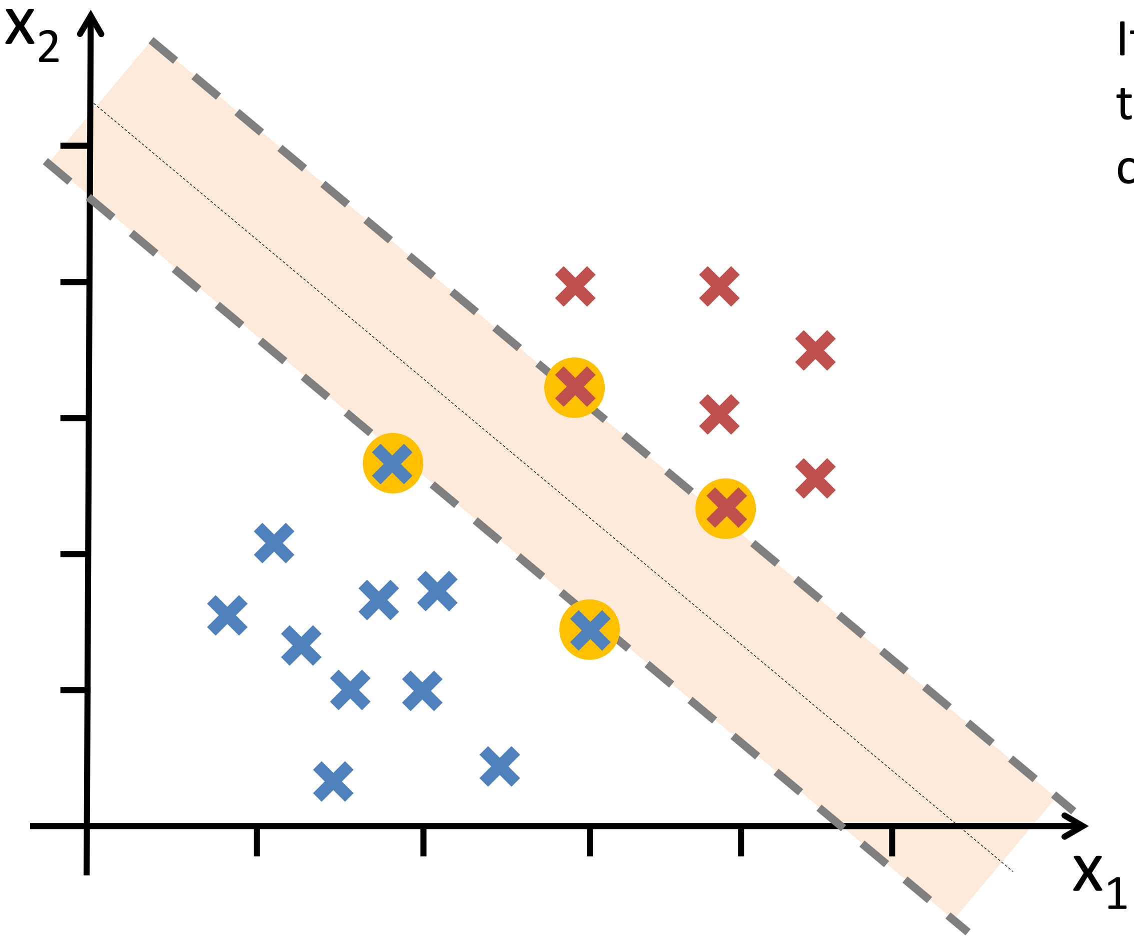
Support vectors: only a few data points are needed to define this margin. They are called support vectors

Support Vector Machines



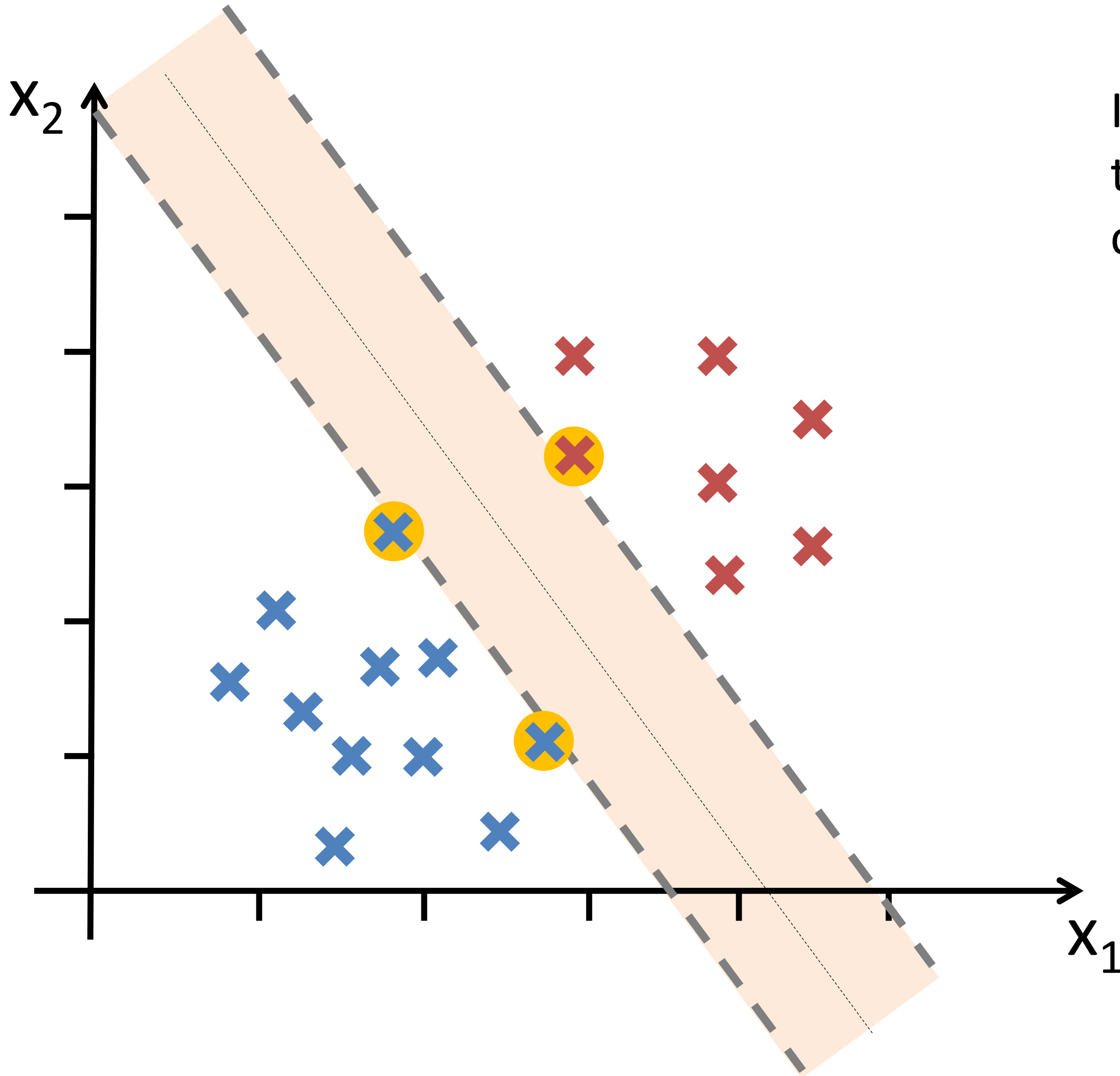
Every other decision boundary, has a smaller margin

Support Vector Machines



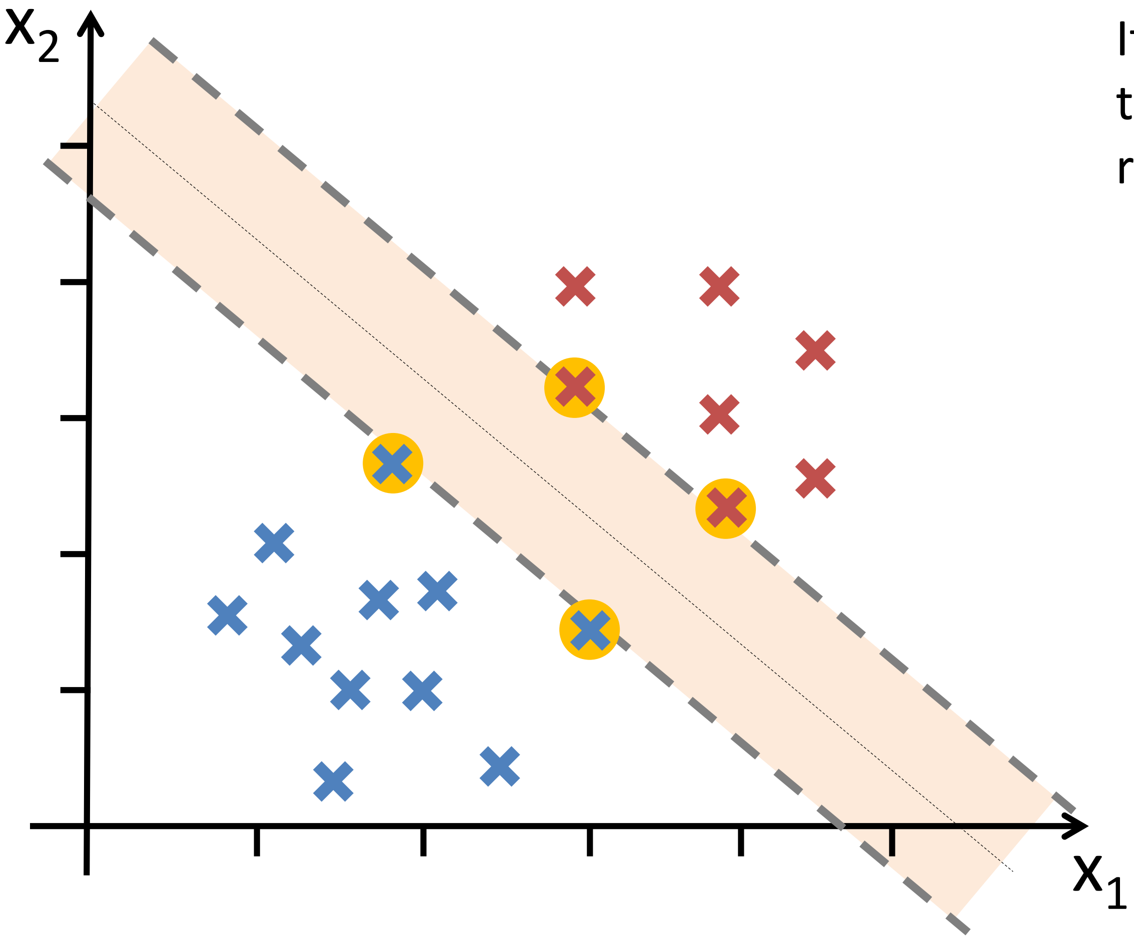
If the support vectors move, the decision boundary changes

Support Vector Machines



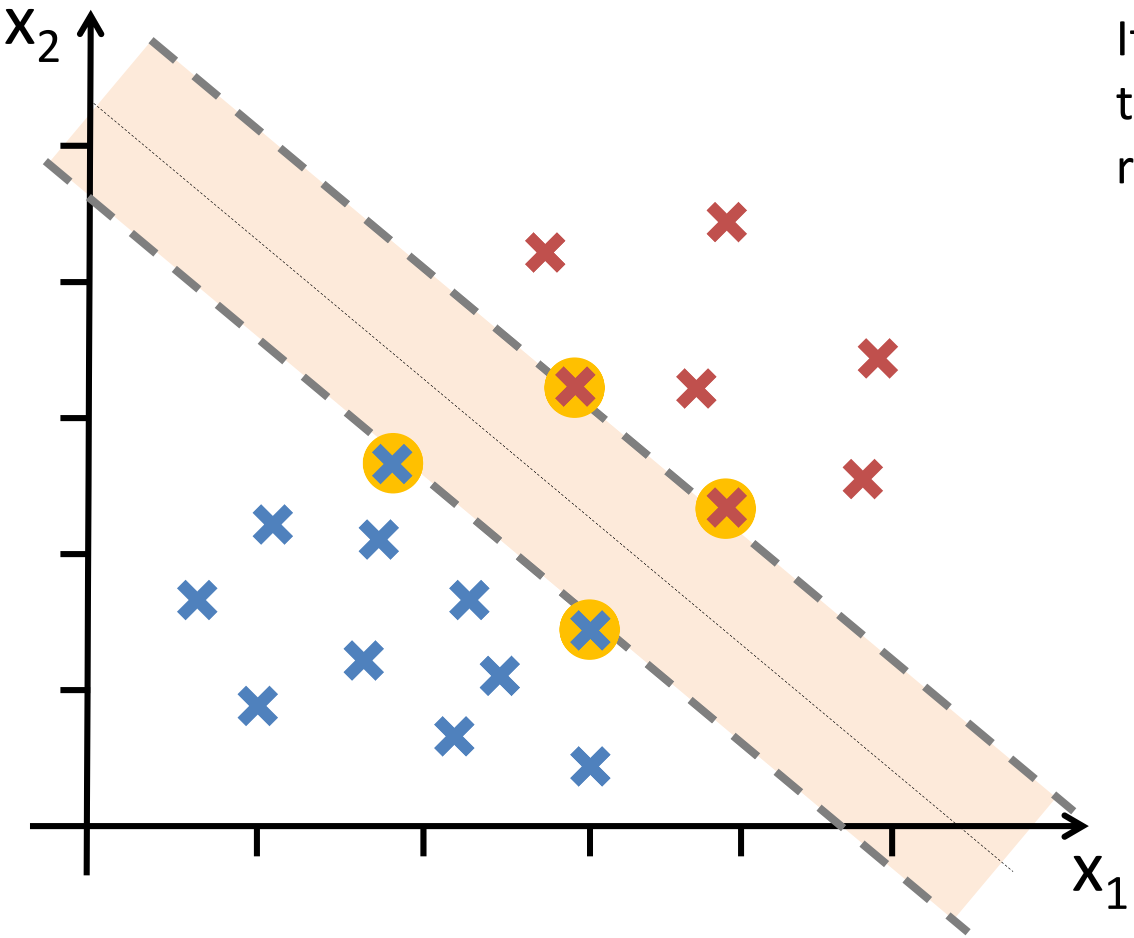
If the support vectors move, the decision boundary changes

Support Vector Machines



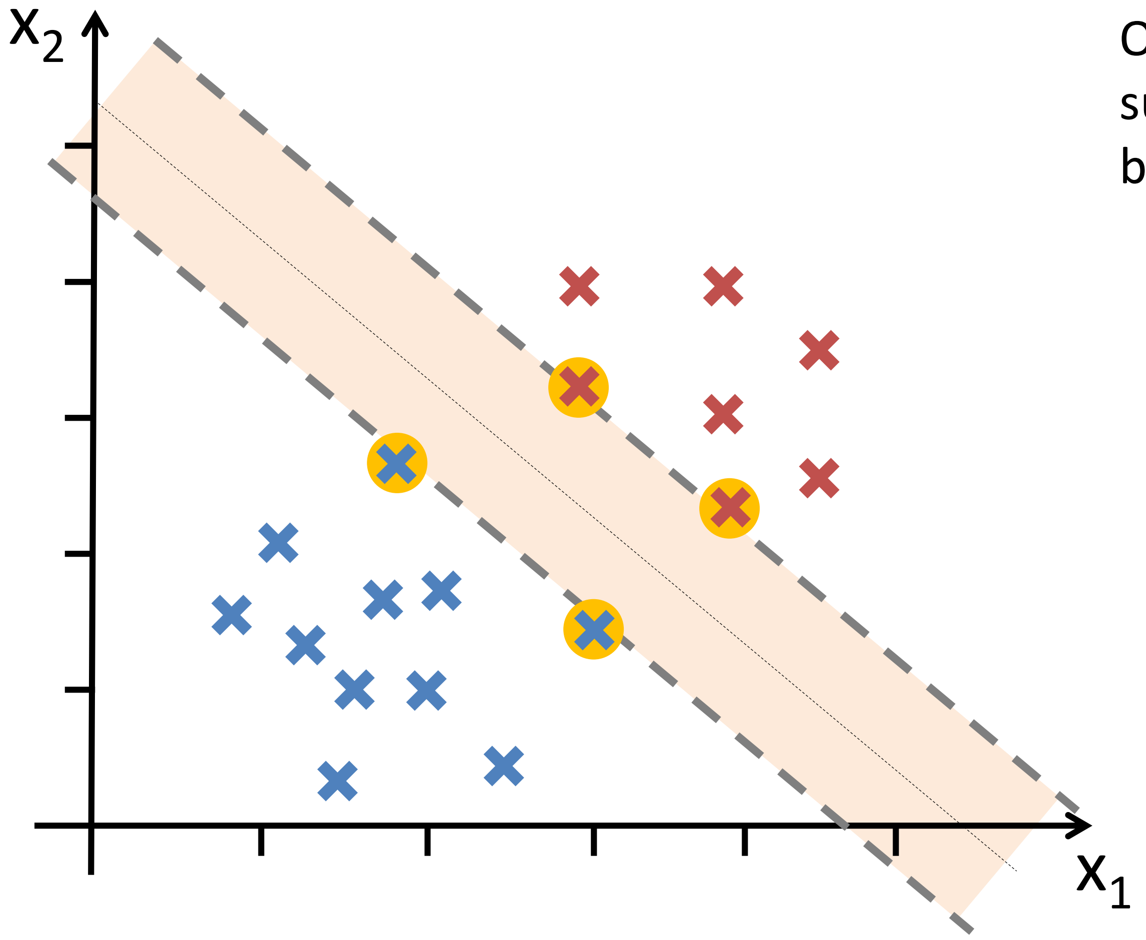
If the rest of the points move, the decision boundary remains stable

Support Vector Machines



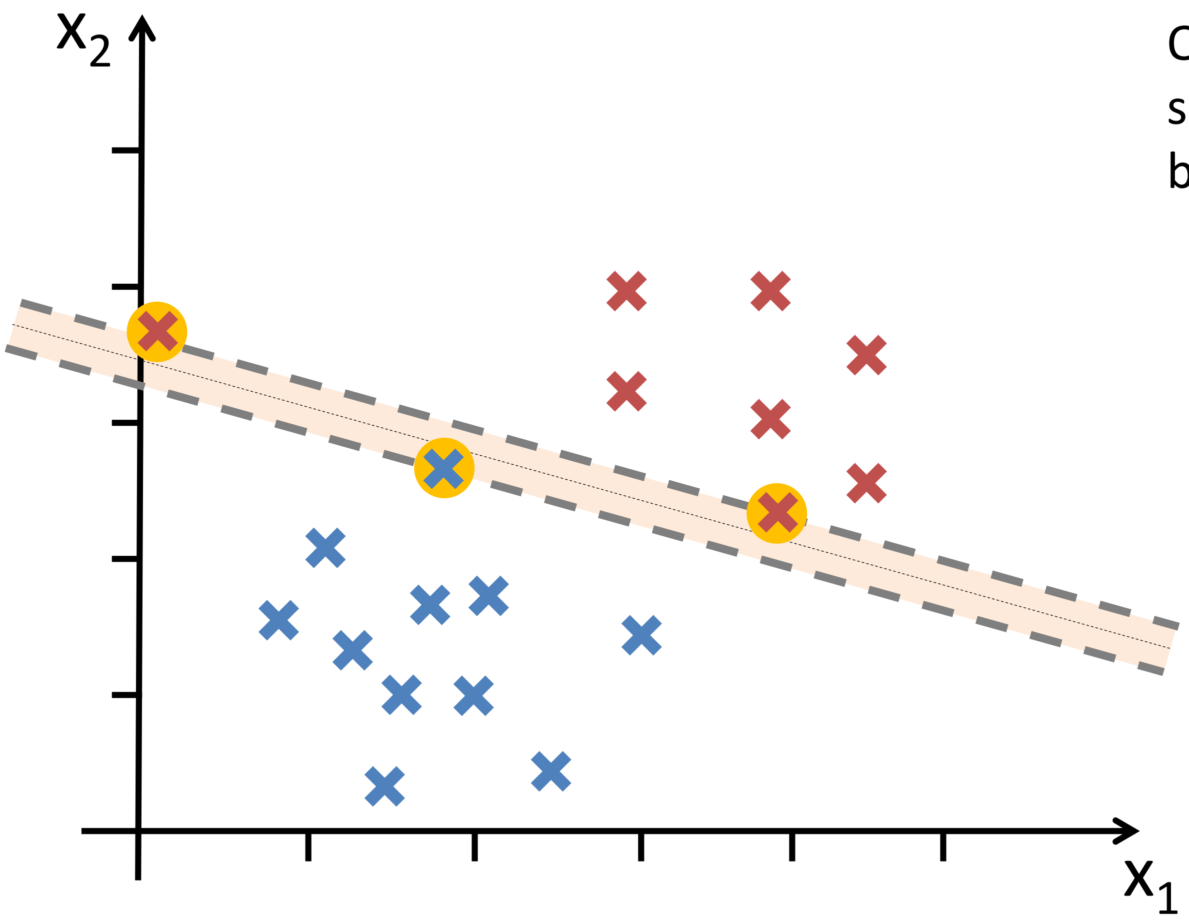
If the rest of the points move, the decision boundary remains stable

Outliers - Soft Margin



Outliers (if they happen to be support vectors) might have a big effect to the solution

Outliers - Soft Margin

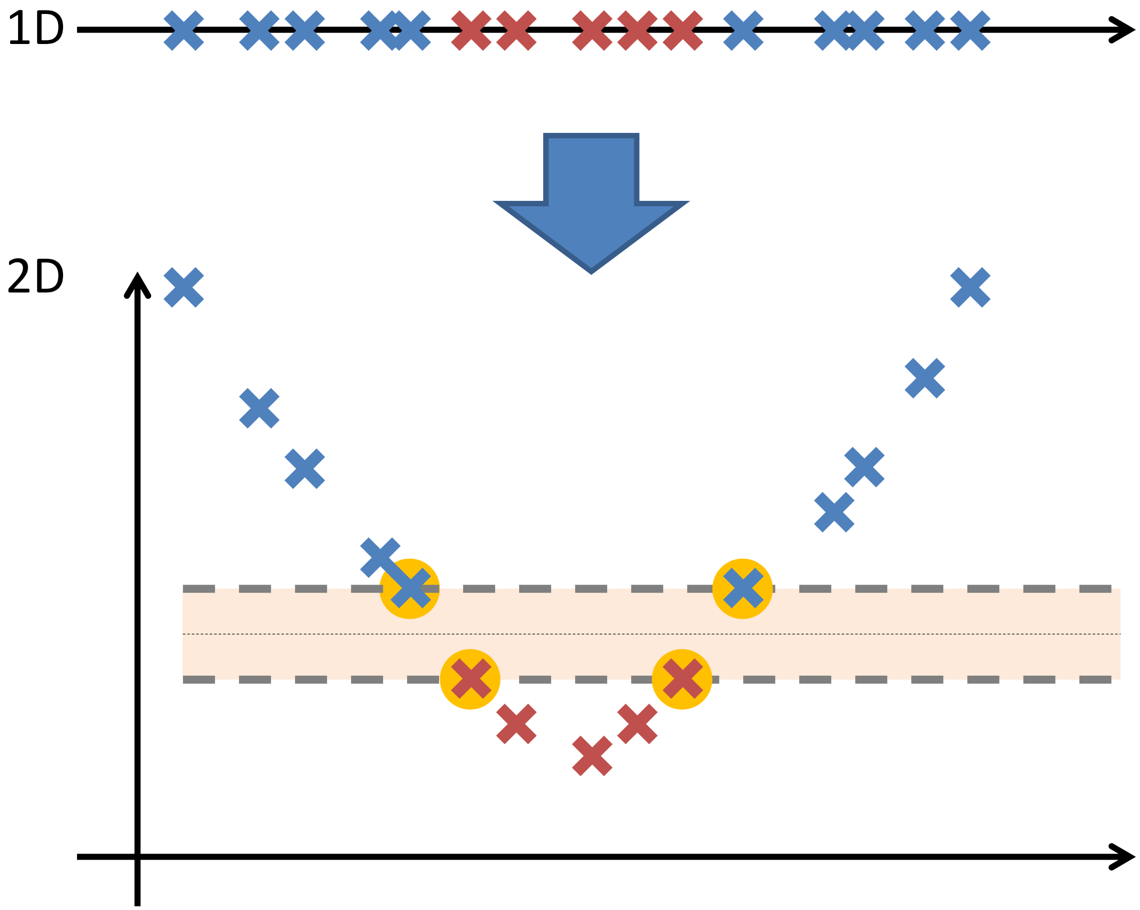


Outliers (if they happen to be support vectors) might have a big effect to the solution

In this case, we can relax the maximum margin condition (**soft margin**)

This implies adding a tolerance to errors, controlled by what is called **slack variables**

Non-Linearly Separable Classes – The Kernel Trick



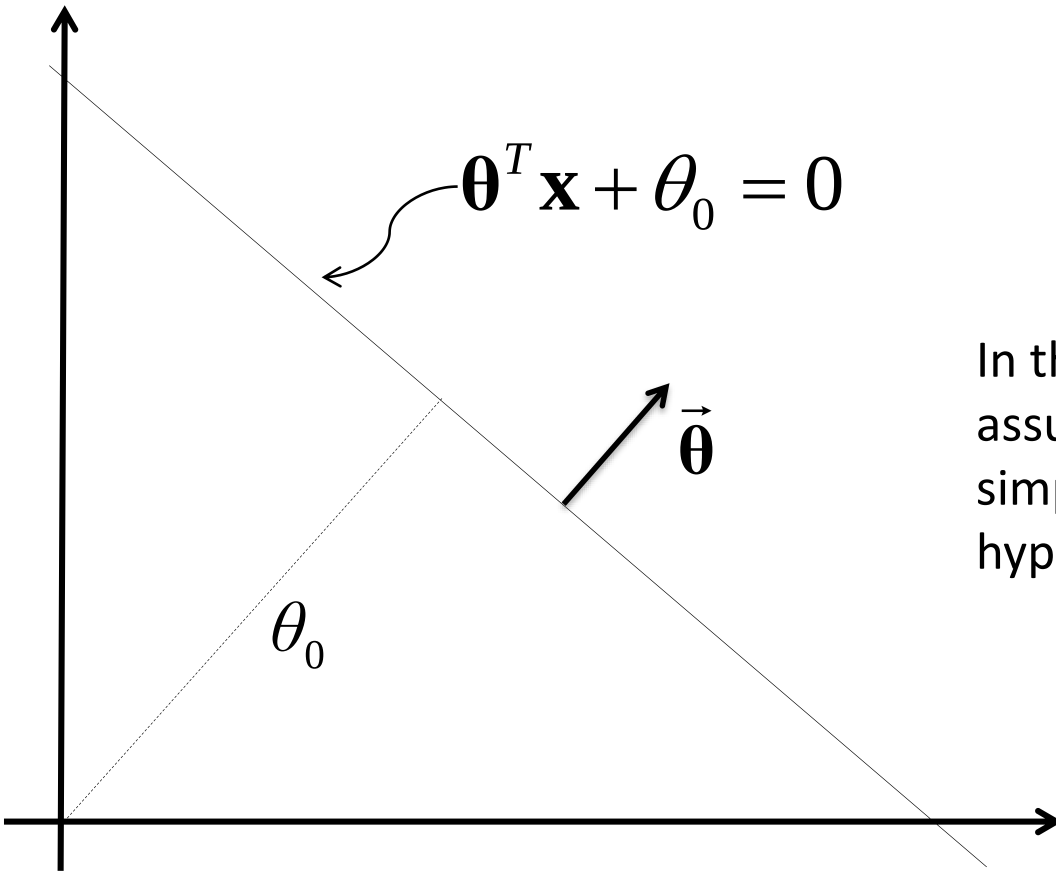
If classes are not linearly separable, SVM will not work

The solution is to transform the feature space into another one (of higher dimension), in which our classes are linearly separable

It is not necessary to define this transformation explicitly. It is only necessary to define a similarity function that calculates the dot-product in this new space. This is called a **kernel**

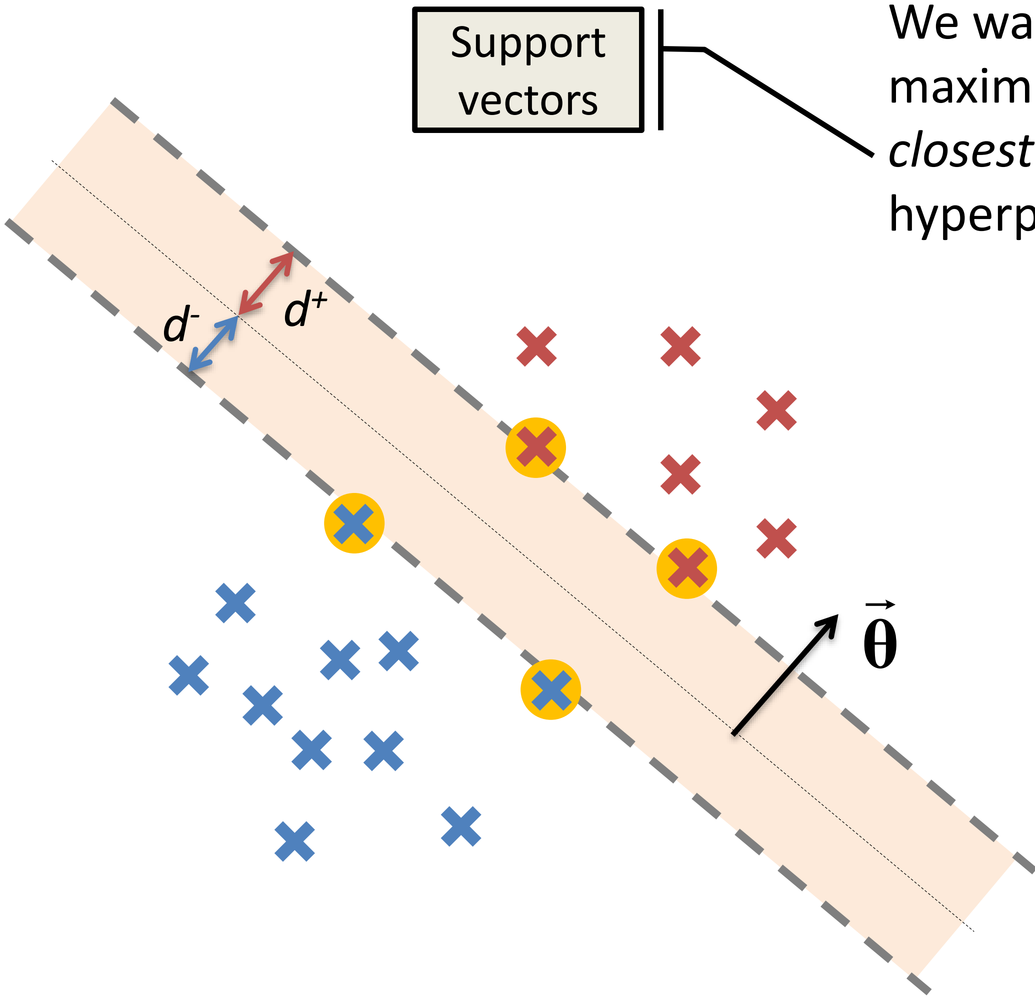
INTUITION AND OPTIMISATION OBJECTIVE

A Line – Just in Case



In the rest of the discussion we will assume that θ_0 is equal to zero for simplicity, this means that the hyperplane crosses the origin

Intuition



We want to find the hyperplane that maximises the distance between the *closest vectors* (of both classes) to the hyperplane and the hyperplane itself

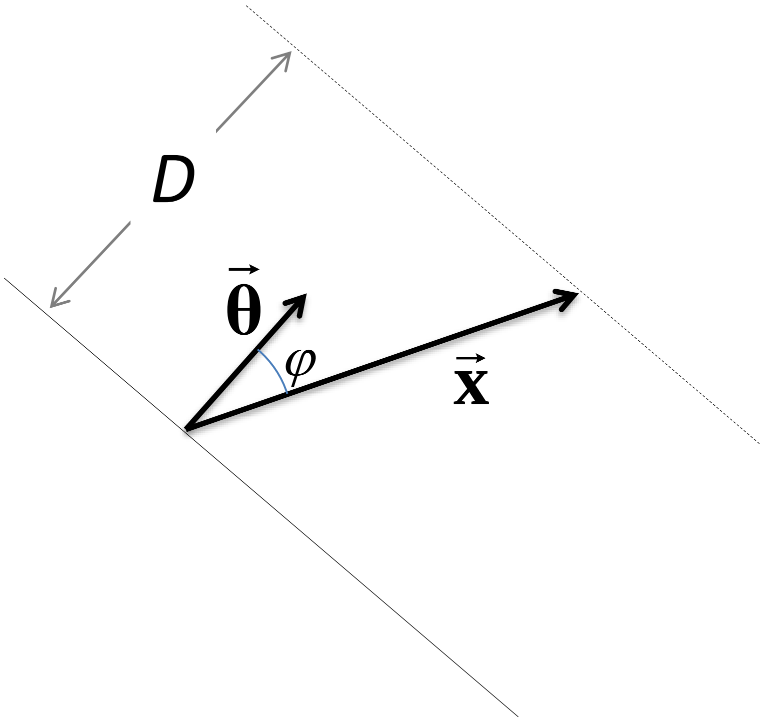
Hyperplane is defined by vector: $\boldsymbol{\theta}$

d^+ : distance from hyperplane to support vectors of class (+)

d^- : distance from hyperplane to support vectors of class (-)

$d^+ = d^- = D$

A bit of Geometry



Given a hyperplane **θ** the distance of a point **x** to the hyperplane is given by:

$$D = |\vec{\mathbf{x}}| \cos(\varphi)$$

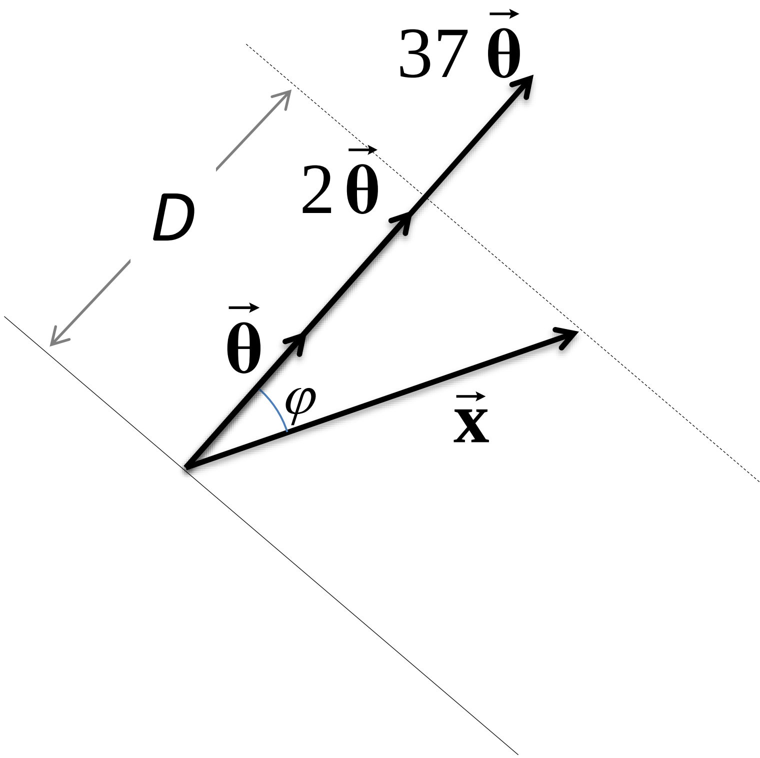
The definition of the dot product between two vectors x and θ is:

$$\vec{\boldsymbol{\theta}} \cdot \vec{\mathbf{x}} = \boldsymbol{\theta}^T \mathbf{x} = |\vec{\boldsymbol{\theta}}| \, |\vec{\mathbf{x}}| \cos(\varphi)$$

Distance can be written:

$$D = \frac{\vec{\boldsymbol{\theta}} \cdot \vec{\mathbf{x}}}{|\vec{\boldsymbol{\theta}}|} = |\vec{\mathbf{x}}| \cos(\varphi)$$

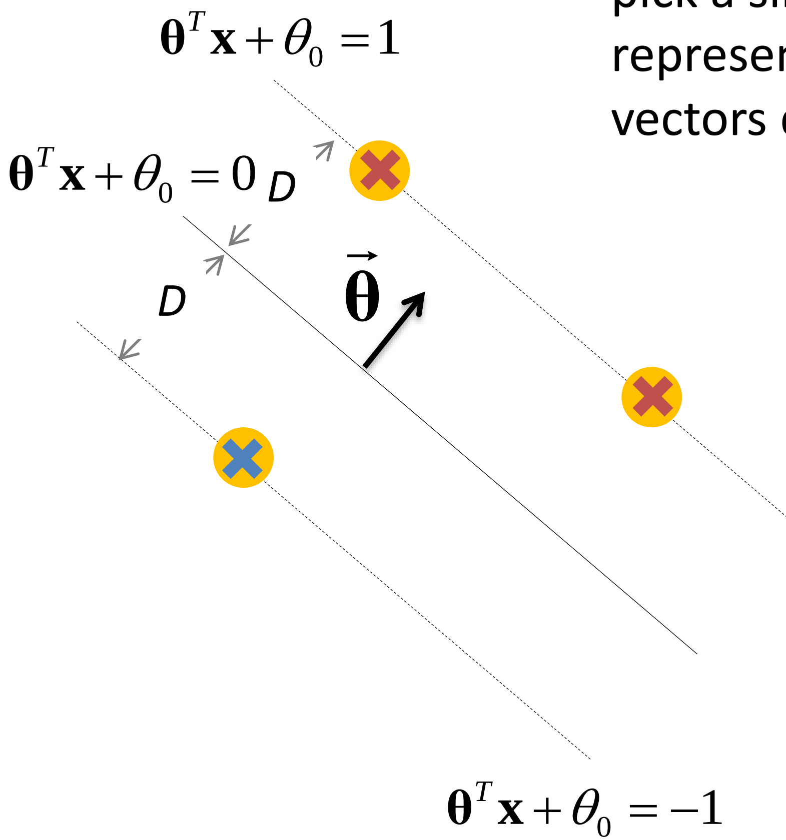
A bit of Geometry



The same hyperplane can be defined by a whole family of vectors (same direction, different lengths), and the distance calculation would be the same:

$$D = \frac{\vec{\boldsymbol{\theta}}}{|\vec{\boldsymbol{\theta}}|}\cdot\vec{\mathbf{x}} = \frac{2\vec{\boldsymbol{\theta}}}{|2\vec{\boldsymbol{\theta}}|}\cdot\vec{\mathbf{x}} = \frac{400\vec{\boldsymbol{\theta}}}{|400\vec{\boldsymbol{\theta}}|}\cdot\vec{\mathbf{x}}$$

A bit of Geometry



When searching for the best hyperplane, we want to pick a single representative between all equivalent representations for a given hyperplane (the family of vectors defining the same hyperplane)

The convention for SVM is to pick the vector **θ** that makes (we introduce θ_0 back):

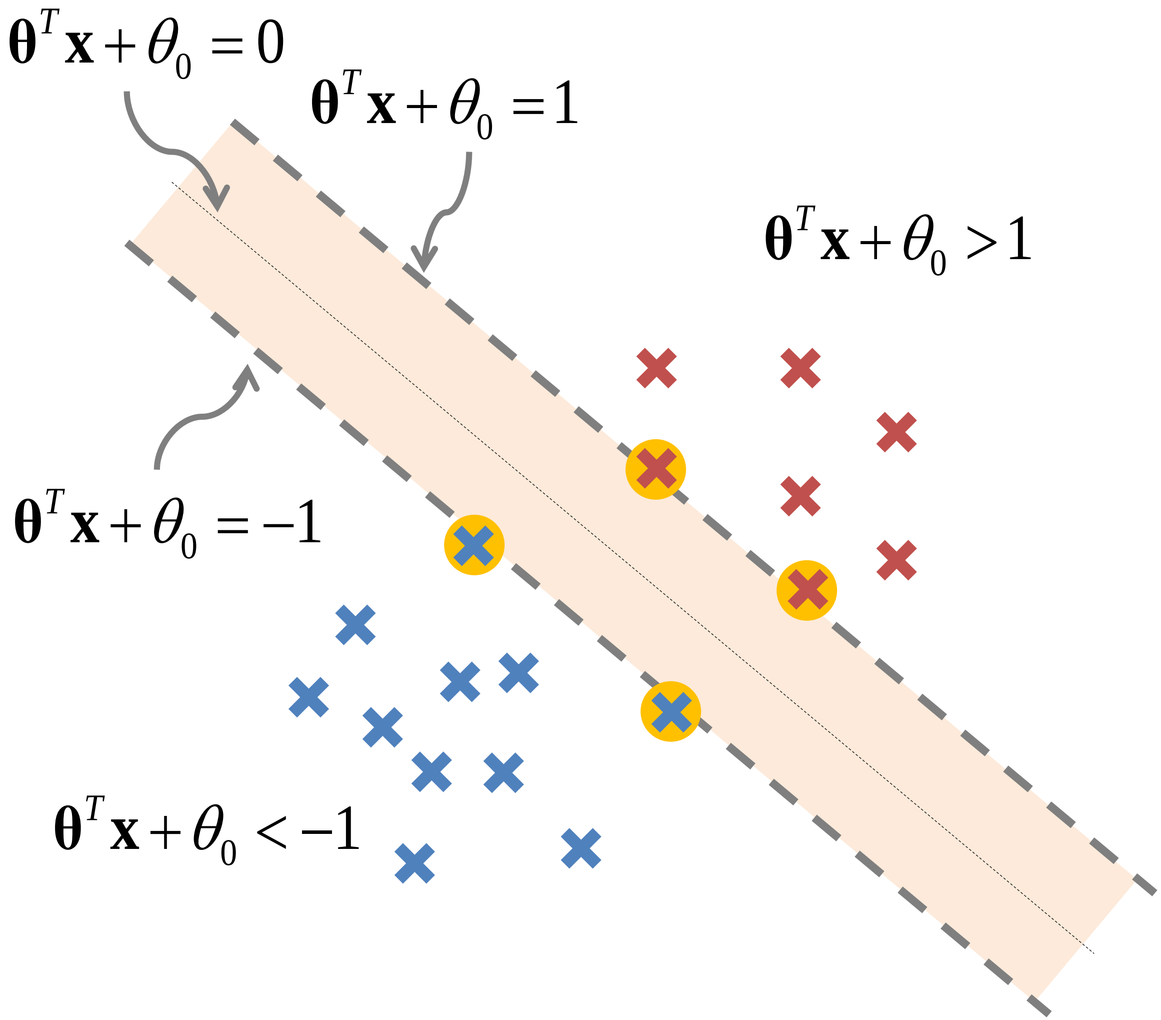
$$\boldsymbol{\theta}^T\mathbf{x}+\theta_0=1 \quad \text{for all support vectors of class}(+)$$
$$\boldsymbol{\theta}^T\mathbf{x}+\theta_0=-1 \quad \text{for all support vectors of class}(-)$$

Therefore the distance we are trying to maximise is given by:

$$D=\frac{\vec{\boldsymbol{\theta}}\cdot\vec{\mathbf{x}}+\theta_0}{|\vec{\boldsymbol{\theta}}|}=\frac{1}{|\vec{\boldsymbol{\theta}}|}$$

Which is **equivalent to minimising |θ|**

Optimisation Function



Maximise margin

Minimise $|\boldsymbol{\theta}|$:

$$\min|\boldsymbol{\theta}| \Leftrightarrow \min \frac{1}{2}|\boldsymbol{\theta}|^2$$

Subject to:

$$y_i(\boldsymbol{\theta}^T\mathbf{x}_i+\theta_0)\geq 1$$

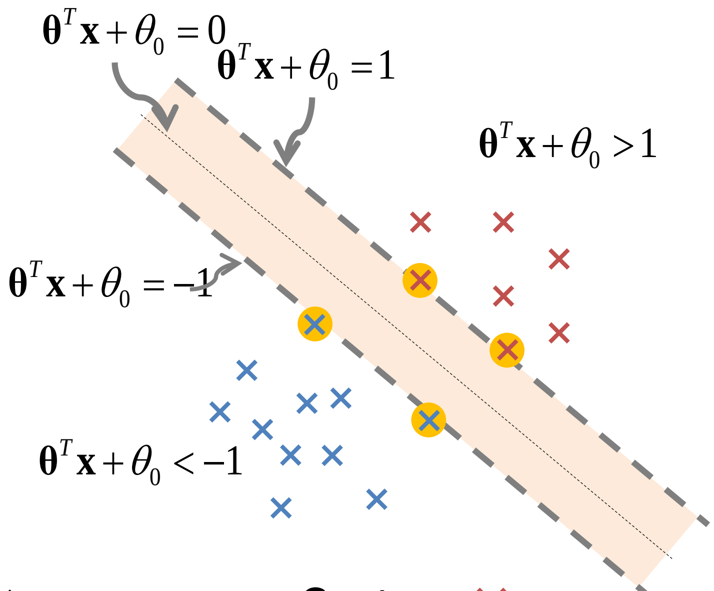
Classify correctly

× $y=1$

× $y=-1$

This is a **quadratic optimisation problem**

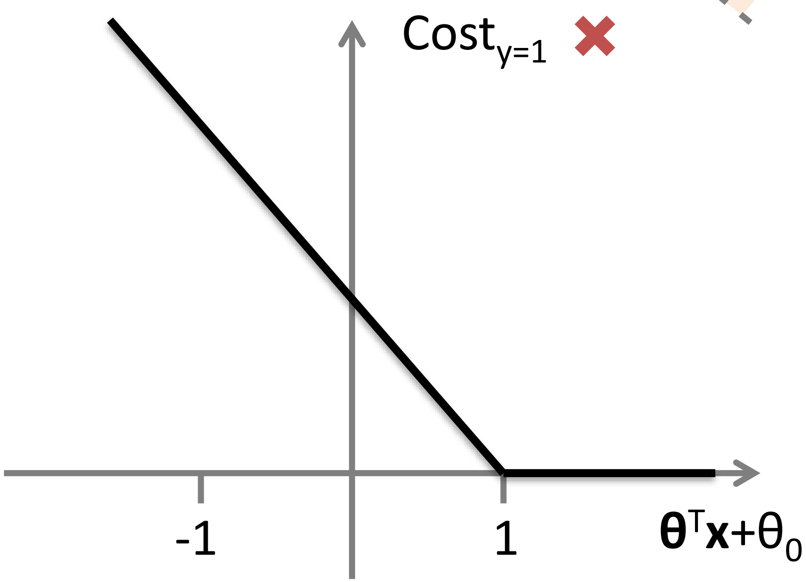
Cost Functions

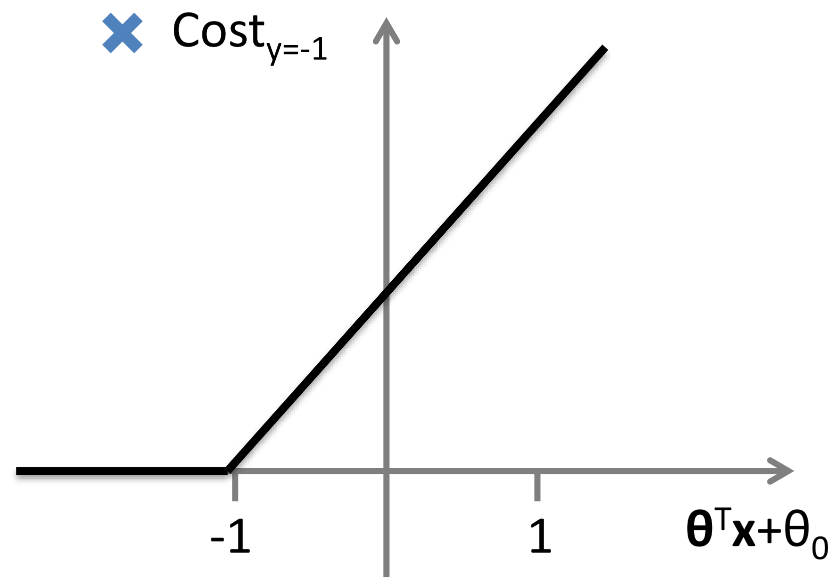


This means setting the costs so that we do not penalise (cost = zero) anything classified correctly, but we do penalise if things fall on the wrong side

✖ $y=1$

✖ $y=-1$



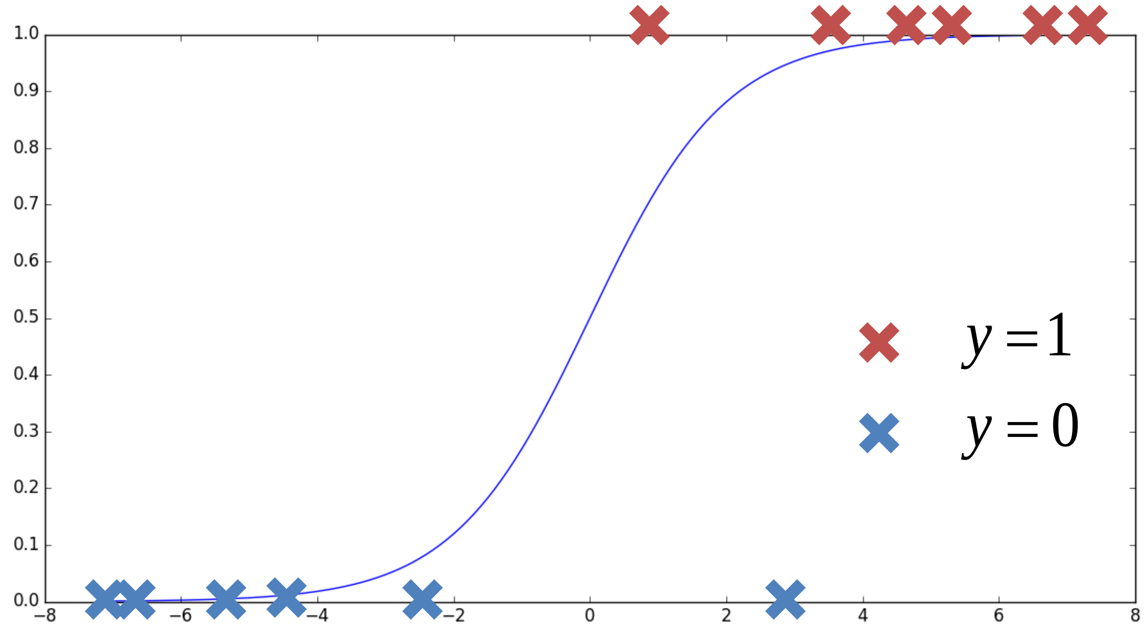


COMPARISON TO LOGISTIC REGRESSION

Logistic Regression Reminder

$$h_{\boldsymbol{\theta}}(x) = \frac{1}{1+e^{-\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}}}$$

$$h_{\boldsymbol{\theta}}(x) = g(z)$$



$$z = \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}$$

If $y=1$, we want $h_{\boldsymbol{\theta}} \approx 1$, $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} \gg 0$

If $y=0$, we want $h_{\boldsymbol{\theta}} \approx 0$, $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} \ll 0$

Simplified cost function

$$J(\vartheta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\vartheta\left(x^{(i)}\right), y^{(i)}\right)$$

$$Cost\left(h_\vartheta(x), y\right) = \begin{cases} -\log\left(h_\vartheta(x)\right) & ,\text{if } \; y = 1 \\ -\log\left(1 - h_\vartheta(x)\right) & ,\text{if } \; y = 0 \end{cases}$$

(Note that y is always either 0 or 1)

$$\boxed{Cost\left(h_\vartheta(x), y\right) = -y\log\left(h_\vartheta(x)\right) - (1-y)\log\left(1 - h_\vartheta(x)\right)}$$

$$\text{If } \; y = 1 : Cost\left(h_\vartheta(x), y\right) = -\log\left(h_\vartheta(x)\right)$$

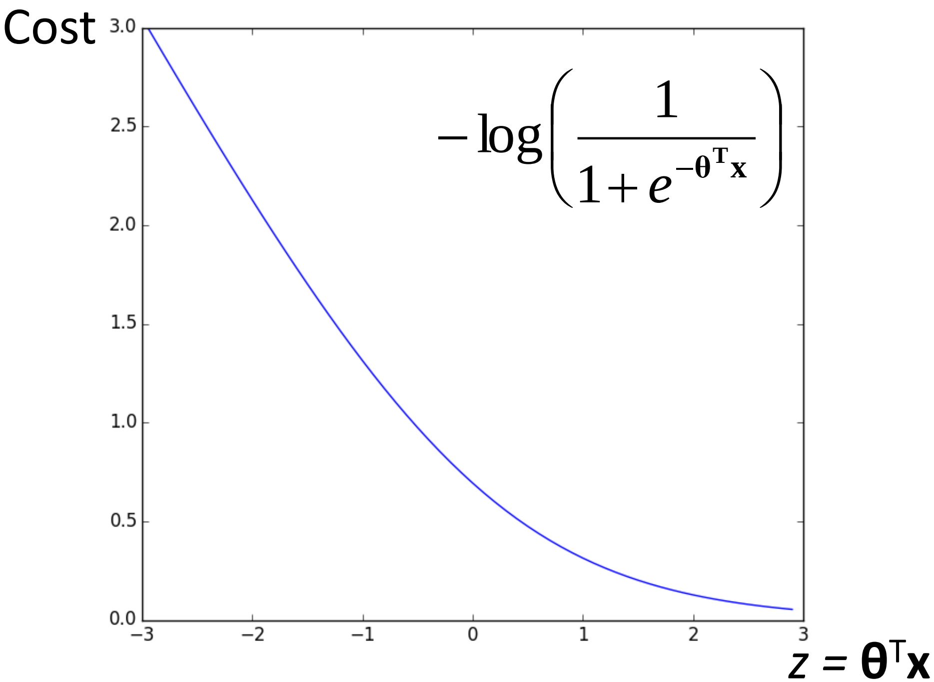
$$\text{If } \; y = 0 : Cost\left(h_\vartheta(x), y\right) = -\log\left(1 - h_\vartheta(x)\right)$$

The Cost of a Sample

$$\text{Cost of sample } = -y\log\left(h_\vartheta(x)\right) - (1-y)\log\left(1-h_\vartheta(x)\right)$$

$$= -y\log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}}\right) - (1-\underbrace{y}_{0})\log\left(1-\frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}}\right)$$

If y=1, we want $\boldsymbol{\theta}^T\mathbf{x} \gg 0$



The Cost of a Sample

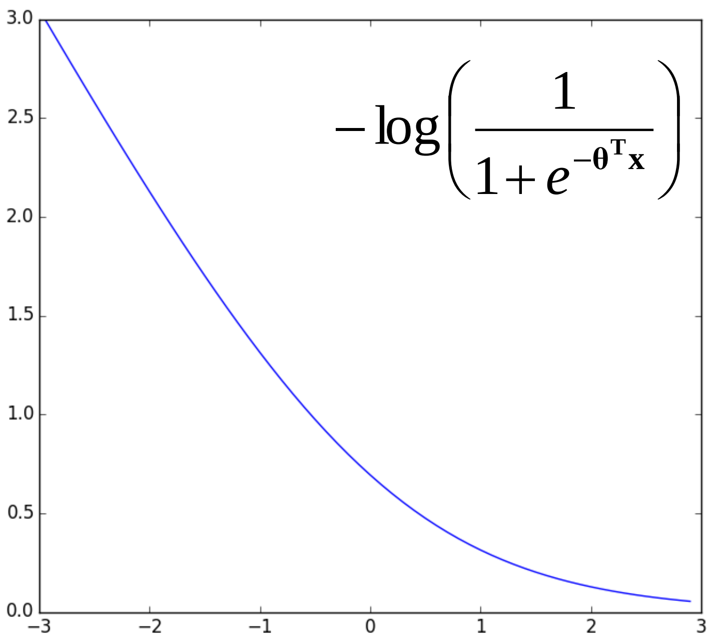
Cost of sample $= -y\log\left(h_\vartheta(x)\right) - (1-y)\log\left(1-h_\vartheta(x)\right)$

0

$$= -y\log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}}}\right) - (1-y)\log\left(1-\frac{1}{1+e^{-\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}}}\right)$$

If y=1, we want $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} \gg 0$

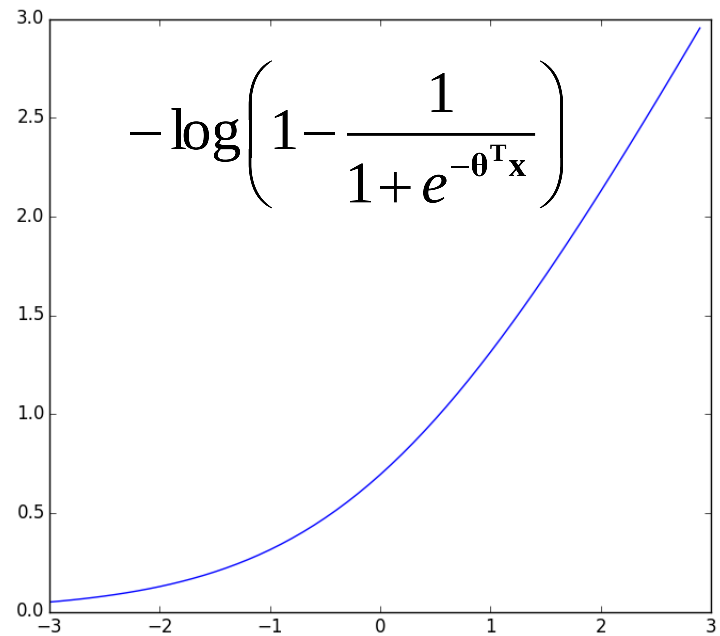
Cost



$z = \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}$

If y=0, we want $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x} \ll 0$

Cost

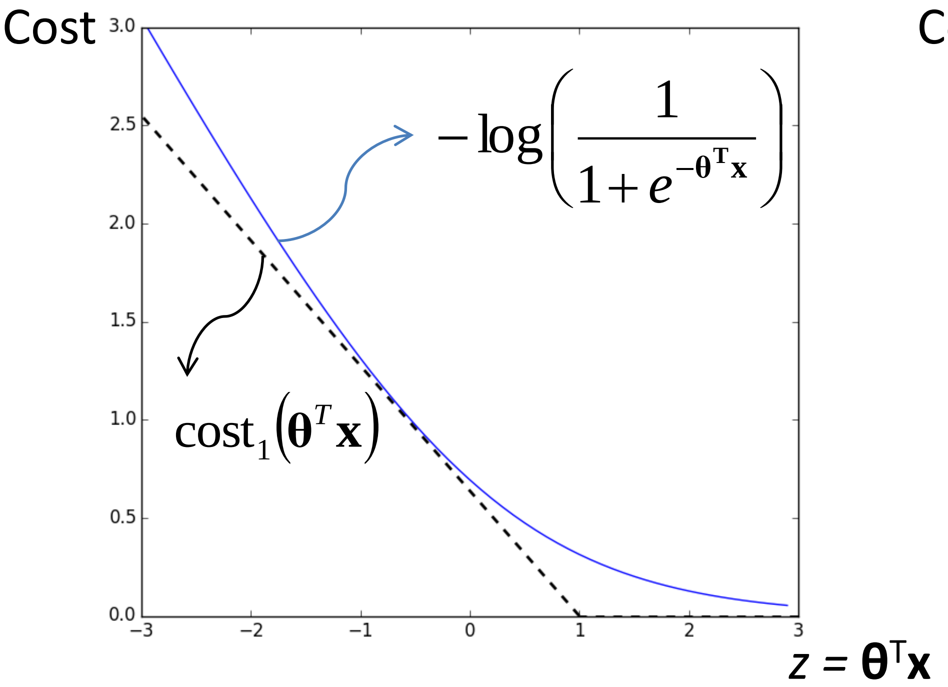


$z = \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}$

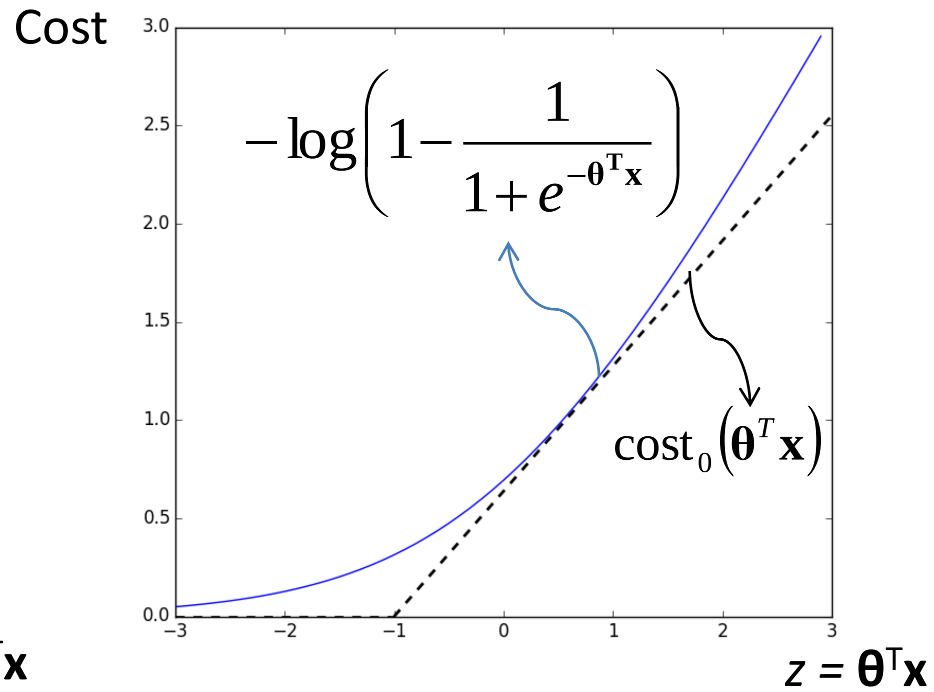
Towards SVM

Cost of sample $= -y\,\mathrm{cost}_1\left(\boldsymbol{\theta}^T\mathbf{x}\right) - (1-y)\,\mathrm{cost}_0\left(\boldsymbol{\theta}^T\mathbf{x}\right)$

If y=1, we want $\boldsymbol{\theta}^T\mathbf{x} \geq 1$



If y=0, we want $\boldsymbol{\theta}^T\mathbf{x} \leq -1$



Optimization Objective

Logistic Regression:

$$\min_{\boldsymbol{\theta}} \frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\left(-\log h_\theta\left(x^{(i)}\right)\right)+\left(1-y^{(i)}\right)\left(-\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right)\right]+\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

SVM:

$$\min_{\boldsymbol{\theta}} \frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1\left(\boldsymbol{\theta}^T\mathbf{x}\right)+\left(1-y^{(i)}\right)\text{cost}_0\left(\boldsymbol{\theta}^T\mathbf{x}\right)\right]+\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

By convention: we do not divide by m, and we use a weight C (equivalent to 1/λ) to control the relative weight of the regularization factor

$$\min_{\boldsymbol{\theta}} C\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1\left(\boldsymbol{\theta}^T\mathbf{x}\right)+\left(1-y^{(i)}\right)\text{cost}_0\left(\boldsymbol{\theta}^T\mathbf{x}\right)\right]+\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

SVM Hypothesis Function

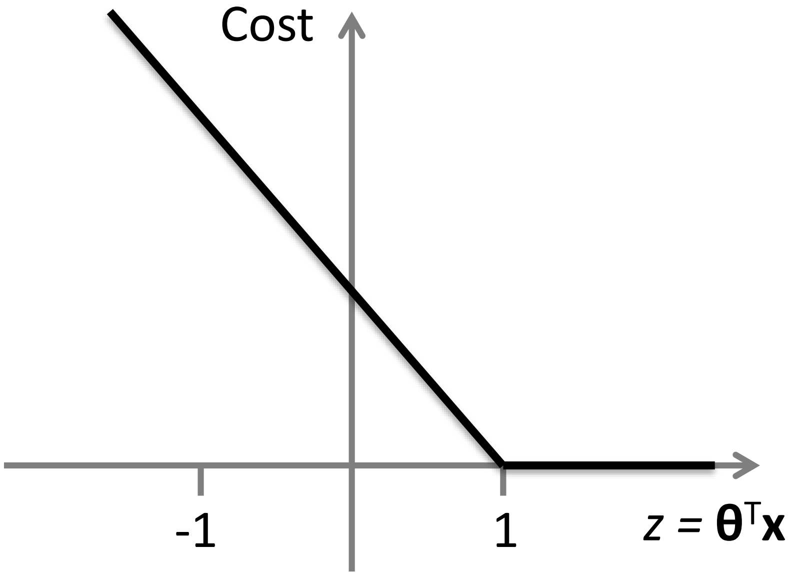
$$\min_{\boldsymbol{\theta}} \; C\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1\left(\boldsymbol{\theta}^T\mathbf{x}\right)+\left(1-y^{(i)}\right)\text{cost}_0\left(\boldsymbol{\theta}^T\mathbf{x}\right)\right]+\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

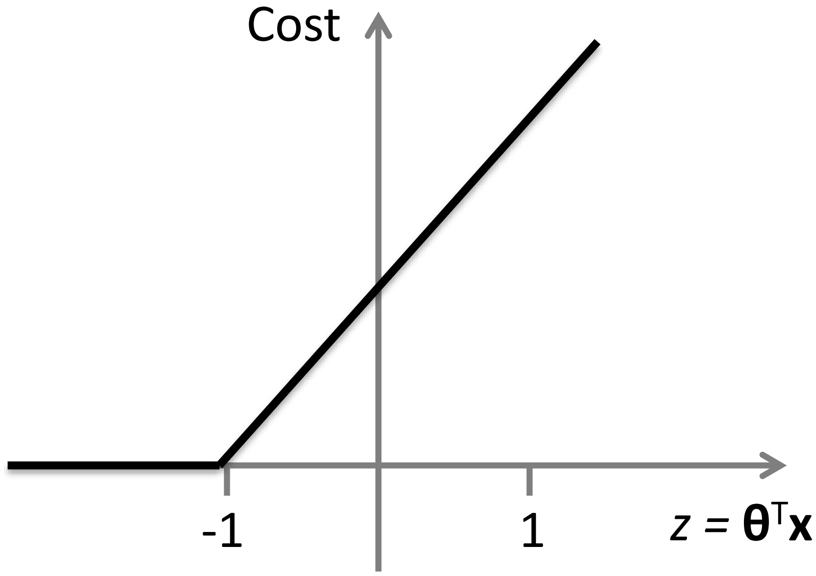
Unlike logistic regression, the SVM output cannot be treated as a probability

$$h_{\boldsymbol{\theta}}(x) = sign\left(\boldsymbol{\theta}^T\mathbf{x}\right)$$

Large Margin Classifier

$$\min_{\boldsymbol{\theta}} C\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1\left(\boldsymbol{\theta}^T\mathbf{x}\right)+\left(1-y^{(i)}\right)\text{cost}_0\left(\boldsymbol{\theta}^T\mathbf{x}\right)\right]+\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$





If y=1, we want $\boldsymbol{\theta}^T\mathbf{x} \geq 1$
(not just $\boldsymbol{\theta}^T\mathbf{x} \geq 0$)

If y=0, we want $\boldsymbol{\theta}^T\mathbf{x} \leq -1$
(not just $\boldsymbol{\theta}^T\mathbf{x} < 0$)

Large Margin Classifier

$$\min_{\boldsymbol{\theta}} \cancelto{0}{C\sum_{i=1}^{m}\left[y^{(i)}\mathrm{cost}_1\left(\boldsymbol{\theta}^T\mathbf{x}\right)+\left(1-y^{(i)}\right)\mathrm{cost}_0\left(\boldsymbol{\theta}^T\mathbf{x}\right)\right]}+\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

Suppose that we are exploring the set of (many possible) decision boundaries that classify correctly our data (C is set to a very large value):

If y=1, $\boldsymbol{\theta}^T\mathbf{x} \geq 1$

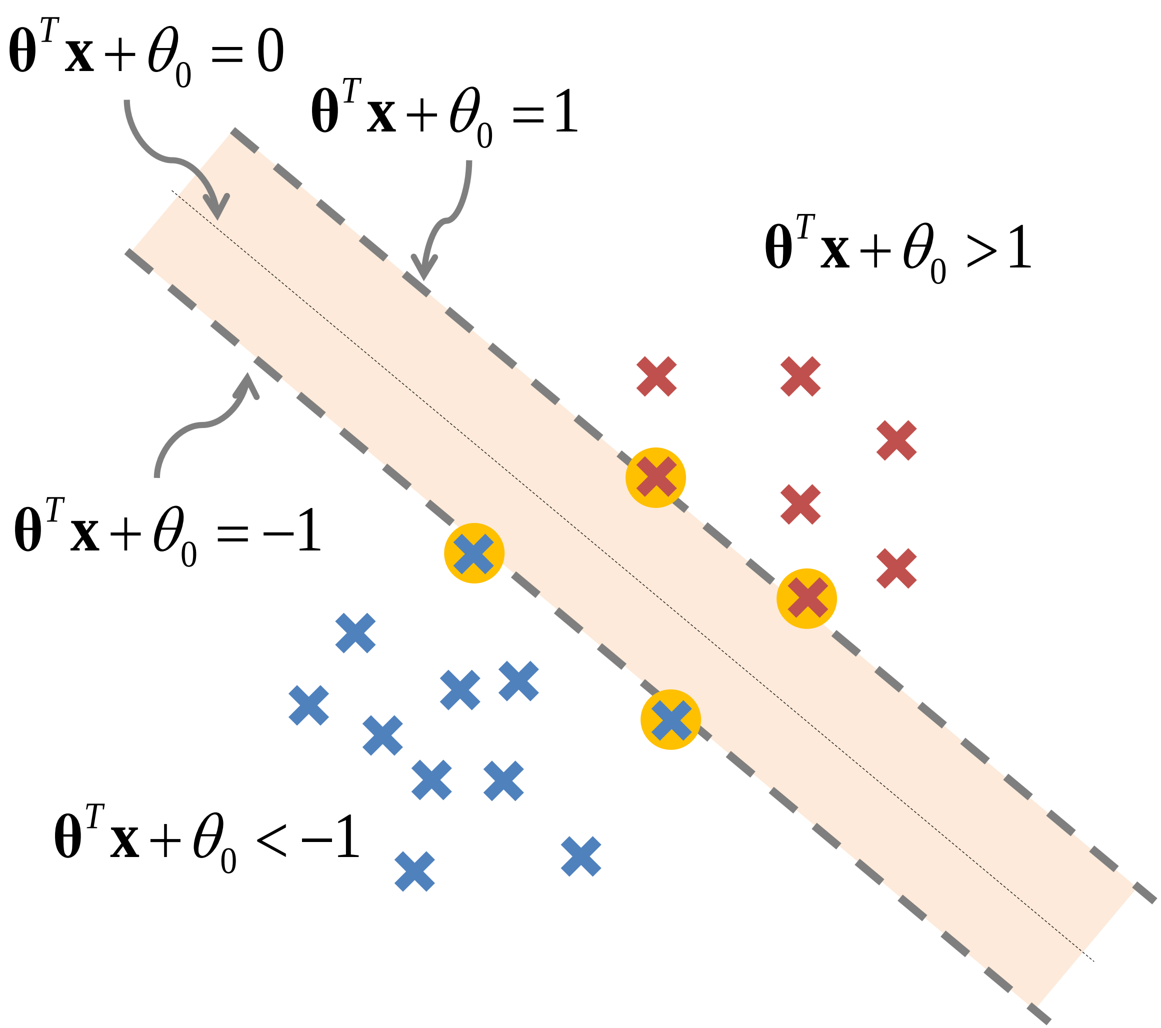
If y=0, $\boldsymbol{\theta}^T\mathbf{x} \leq -1$

The first term of the objective function is zero

Minimizing the second term has the effect of **selecting the solution with the maximum margin**!

DUAL REPRESENTATION AND KERNELS

Optimisation Function



Maximise margin

Minimise:

$$\frac{1}{2}\left|\boldsymbol{\theta}\right|^2 = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$$

Subject to:

$$y_i(\boldsymbol{\theta}^T \mathbf{x}_i + \theta_0) \geq 1$$

Classify correctly

× $y = 1$

× $y = -1$

This is a **quadratic optimisation problem**

Dual Representation

Minimizing theta is a non-linear optimization task.

For solving this type of quadratic optimization problems we use the **Karush-Kuhn-Tucker (KKT)** conditions. The KKT approach generalises the method of **Lagrange multipliers**, a strategy for finding local minima/maxima of a function subject to equality constraints to allow inequality constraints.

Minimise:

$$\frac{1}{2}\left|\boldsymbol{\theta}\right|^2 = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$$

Subject to:

$$y_i(\boldsymbol{\theta}^T\mathbf{x}_i + \theta_0) \geq 1$$

The weights can be estimated as:

$$\boldsymbol{\theta} = \sum_{i=0}^{N} \alpha_i y_i \mathbf{x}_i \qquad \sum_{i=0}^{N} \alpha_i y_i = 0$$

So the decision function can be re-written as:

$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x} + \theta_0 = \sum \alpha_i y_i \mathbf{x}_i{}^T\mathbf{x} + \theta_0$$

Dual Representation

Applying this approach we derive an equivalent representation for our problem.

Primal Form

Minimise:

$$\frac{1}{2}|\boldsymbol{\theta}|^2 = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$$

Subject to:

$$y_i(\boldsymbol{\theta}^T\mathbf{x}_i + \theta_0) \geq 1$$

Dual Form

Maximise:

$$\tilde{L}(\alpha) = -\frac{1}{2}\sum_i\sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i$$

Data appears only inside dot products

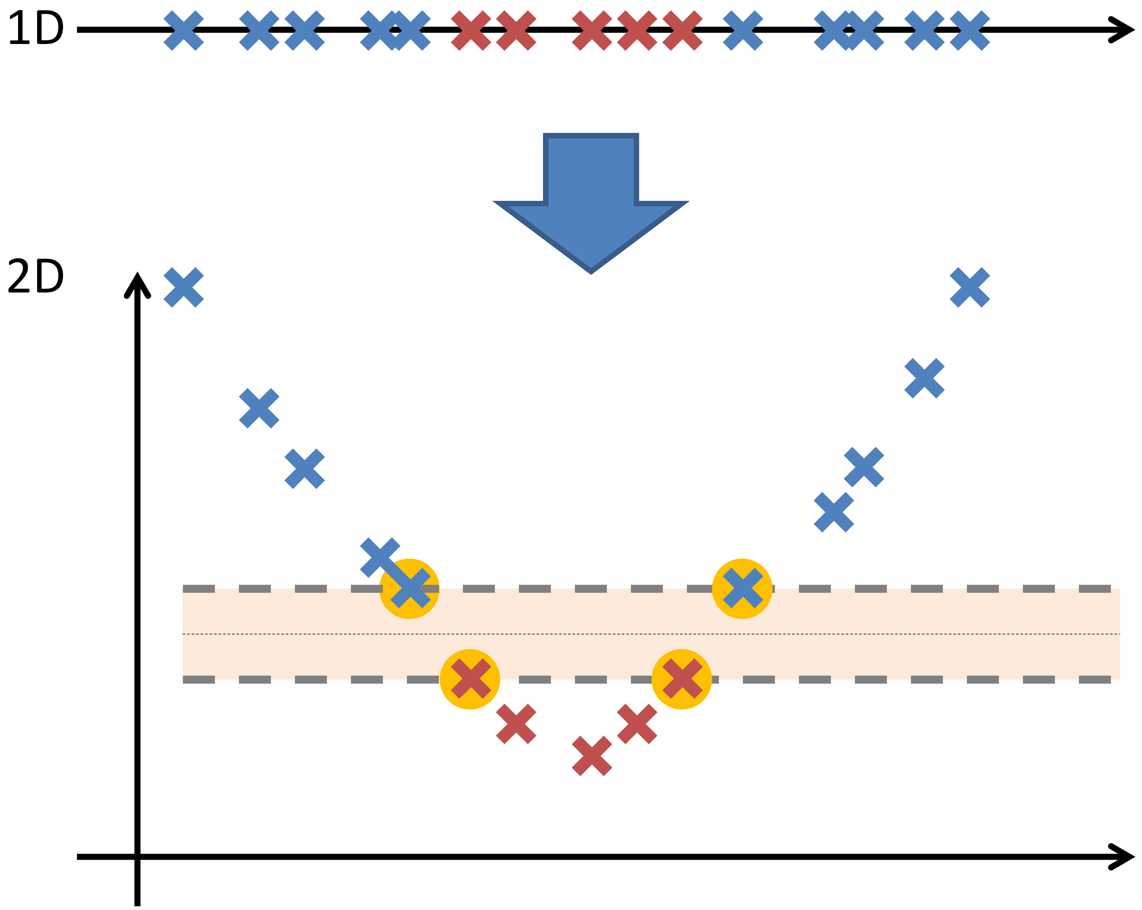
Subject to:

$$\sum_{i=0}^{N} \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Only a few α_i will be greater than zero. The corresponding x_i are exactly the *support vectors*

Non-Linearly Separable Classes – The Kernel Trick

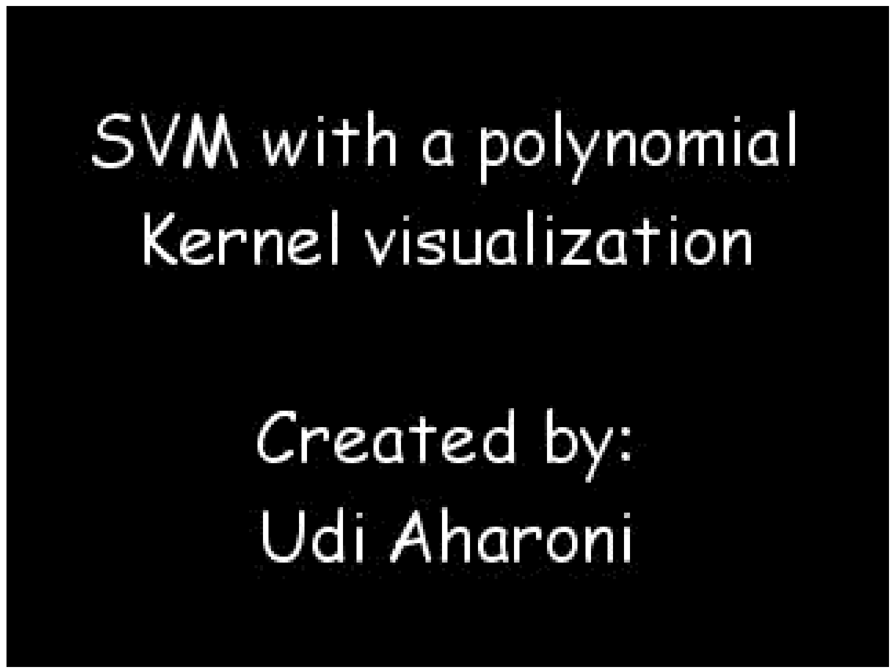


If classes are not linearly separable, SVM will not work

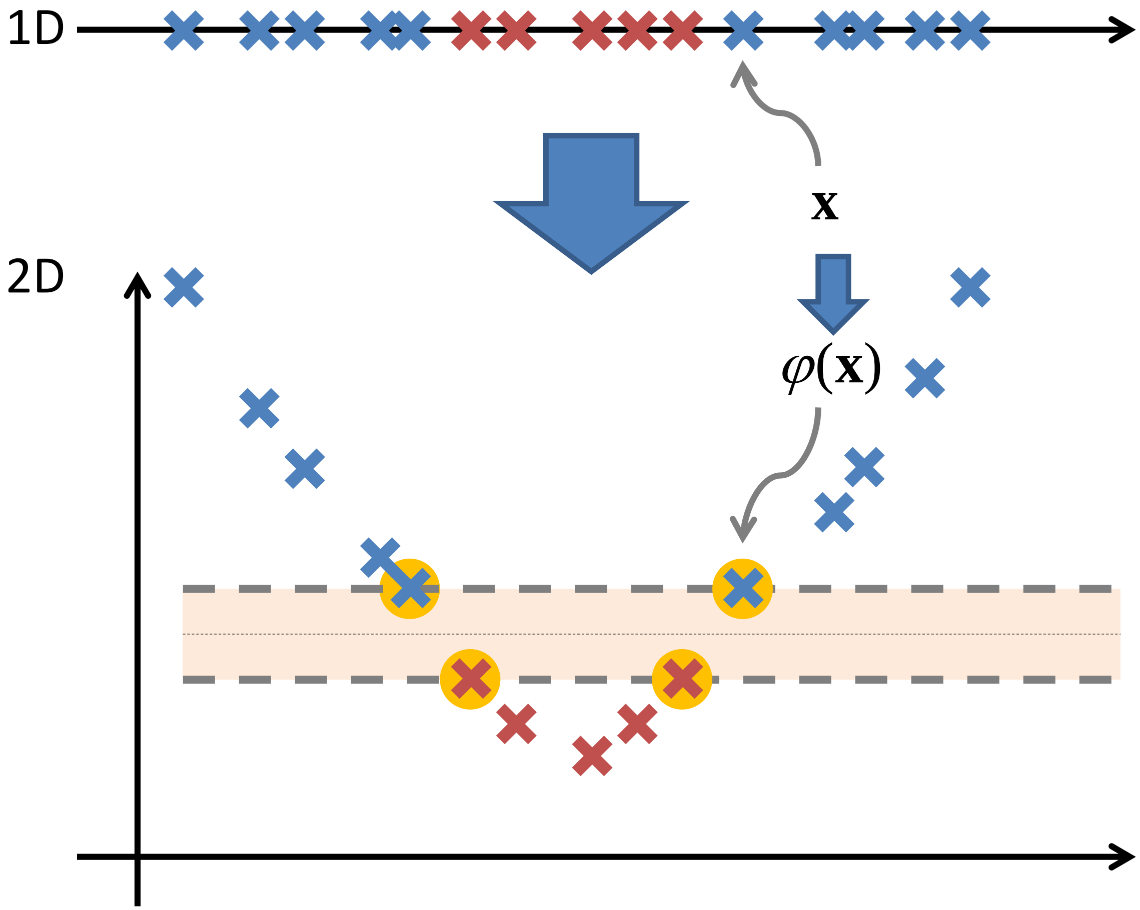
The solution is to transform the feature space into another one (of higher dimension), in which our classes are linearly separable

It is not necessary to define this transformation explicitly. It is only necessary to define a similarity function that calculates the dot-product in this new space. This is called a **kernel**

Non-Linearly Separable Classes – The Kernel Trick



Non-Linearly Separable Classes – The Kernel Trick



Mapping function:

$$\mathbf{x} \mapsto \varphi(\mathbf{x})$$

Kernel function:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \varphi(\mathbf{x}_1)^T \varphi(\mathbf{x}_2)$$

The Kernel Trick

Dual Form

Maximise:

$$\tilde{L}(\alpha) = -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_i \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i$$

Subject to:

$$\alpha_i \geq 0 \qquad \sum_{i=0}^{N} \alpha_i y_i = 0$$

Kernel SVM

Maximise:

$$\tilde{L}(\alpha) = -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_i K\left(\mathbf{x}_i, \mathbf{x}_j\right) + \sum_i \alpha_i$$

Subject to:

$$\alpha_i \geq 0 \qquad \sum_{i=0}^{N} \alpha_i y_i = 0$$

$$\theta_0 = y_i - \boldsymbol{\theta}^T \varphi(\mathbf{x}_i) = y_i - \sum_j y_j \alpha_i K(\mathbf{x_j}, \mathbf{x_i})$$

$$h_\theta(\mathbf{x}) = \text{sign}\,(\sum_i y_i \alpha_i K(\mathbf{x}, \mathbf{x_i}) + b)$$

Example Kernel Functions

Kernel functions introduce their own parameters, the value of which is generally fixed through cross-validation

Polynomial: $K(\mathbf{x},\mathbf{z}) = \left(\mathbf{x}^T\mathbf{z}\right)^d$

Radial Basis Functions: $K(\mathbf{x},\mathbf{z}) = e^{-\|x-z\|^2/2\sigma}$

Sigmoid: $K(\mathbf{x},\mathbf{z}) = \tanh(\kappa\langle\mathbf{x},\mathbf{z}\rangle - \delta)$

Inverse Multiquadric: $K(\mathbf{x},\mathbf{z}) = (\|\mathbf{x}-\mathbf{z}\|^{1/2} 2\sigma + c^2)^{-1}$

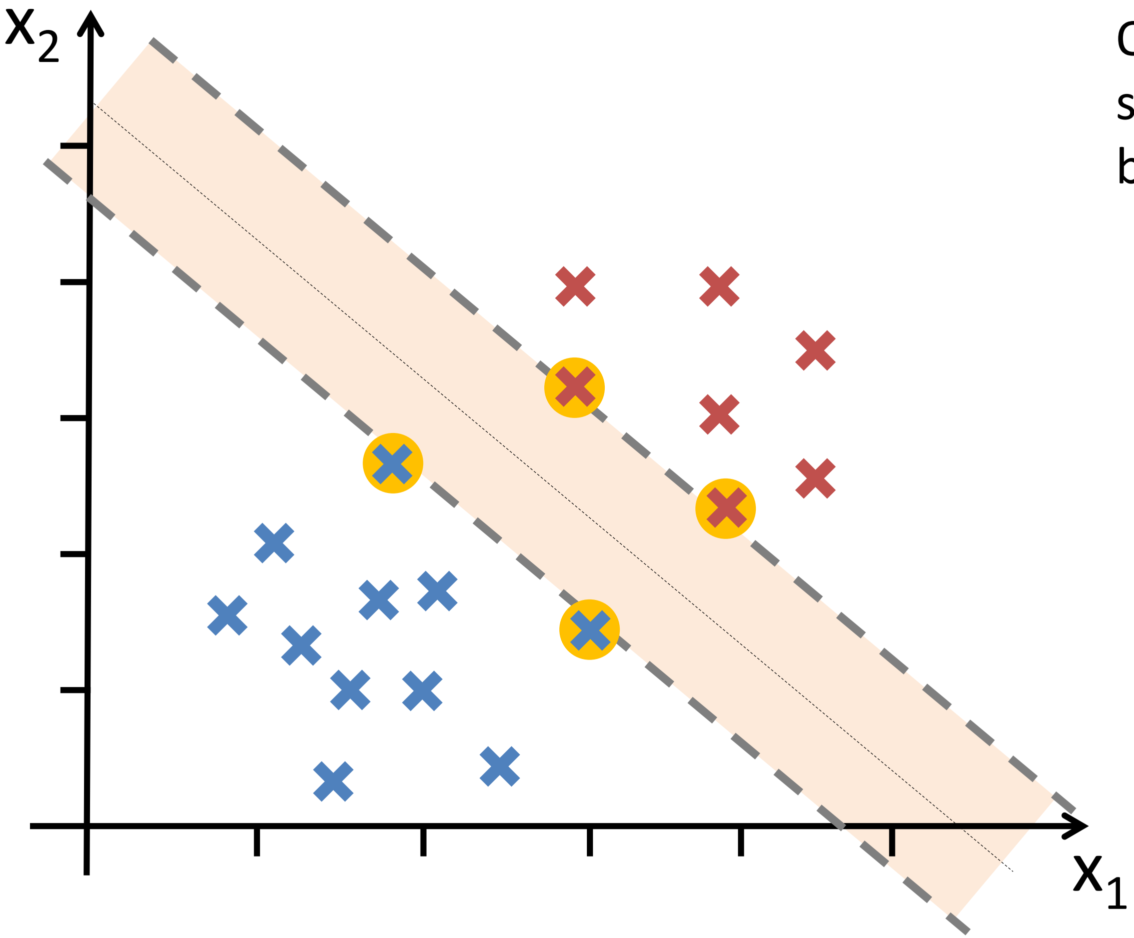
Intersection Kernel: $K(\mathbf{x},\mathbf{z}) = \sum_{i=1}^{n} \min(x(i), z(i))$

The small print

Kernel functions must be continuous, symmetric, and most preferably should have a positive (semi-) definite Gram matrix. Kernels which are said to satisfy the Mercer's theorem are positive semi-definite, meaning their kernel matrices have only non-negative Eigen values. The use of a positive definite kernel insures that the optimization problem will be convex and solution will be unique.

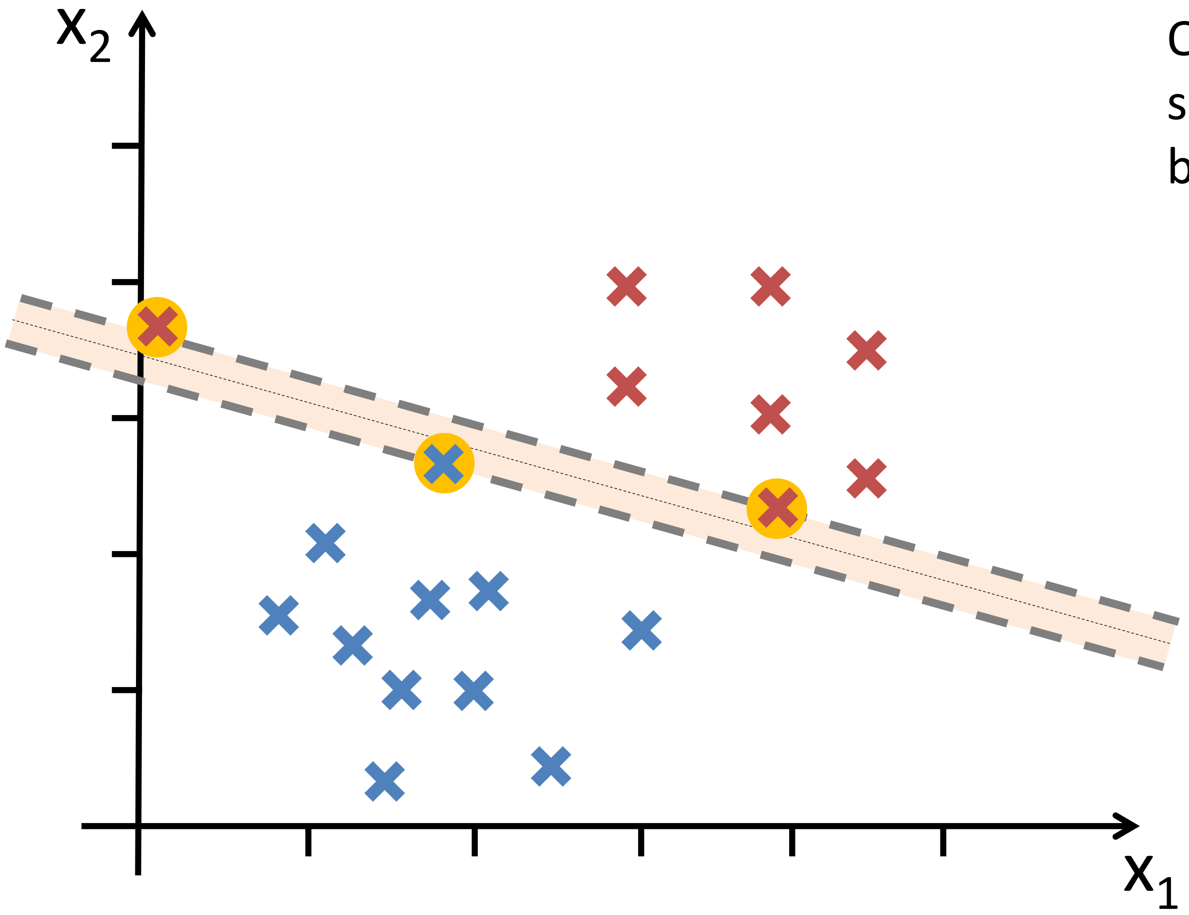
SOFT MARGIN, SLACK VARIABLES

Outliers - Soft Margin



Outliers (if they happen to be support vectors) might have a big effect to the solution

Outliers - Soft Margin



Outliers (if they happen to be support vectors) might have a big effect to the solution

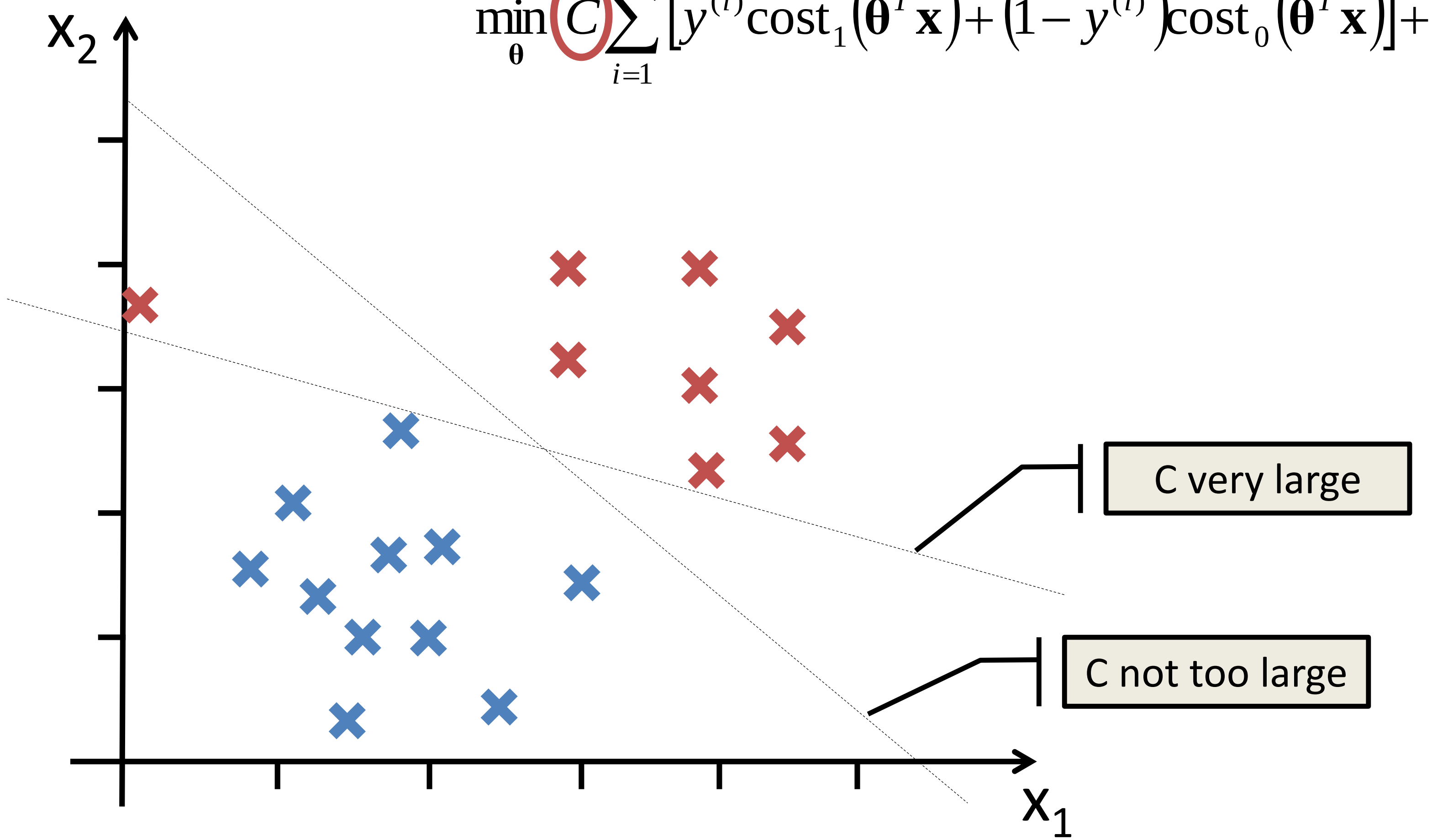
In this case, we can relax the maximum margin condition (**soft margin**)

This implies adding a tolerance to errors, controlled by what is called **slack variables**

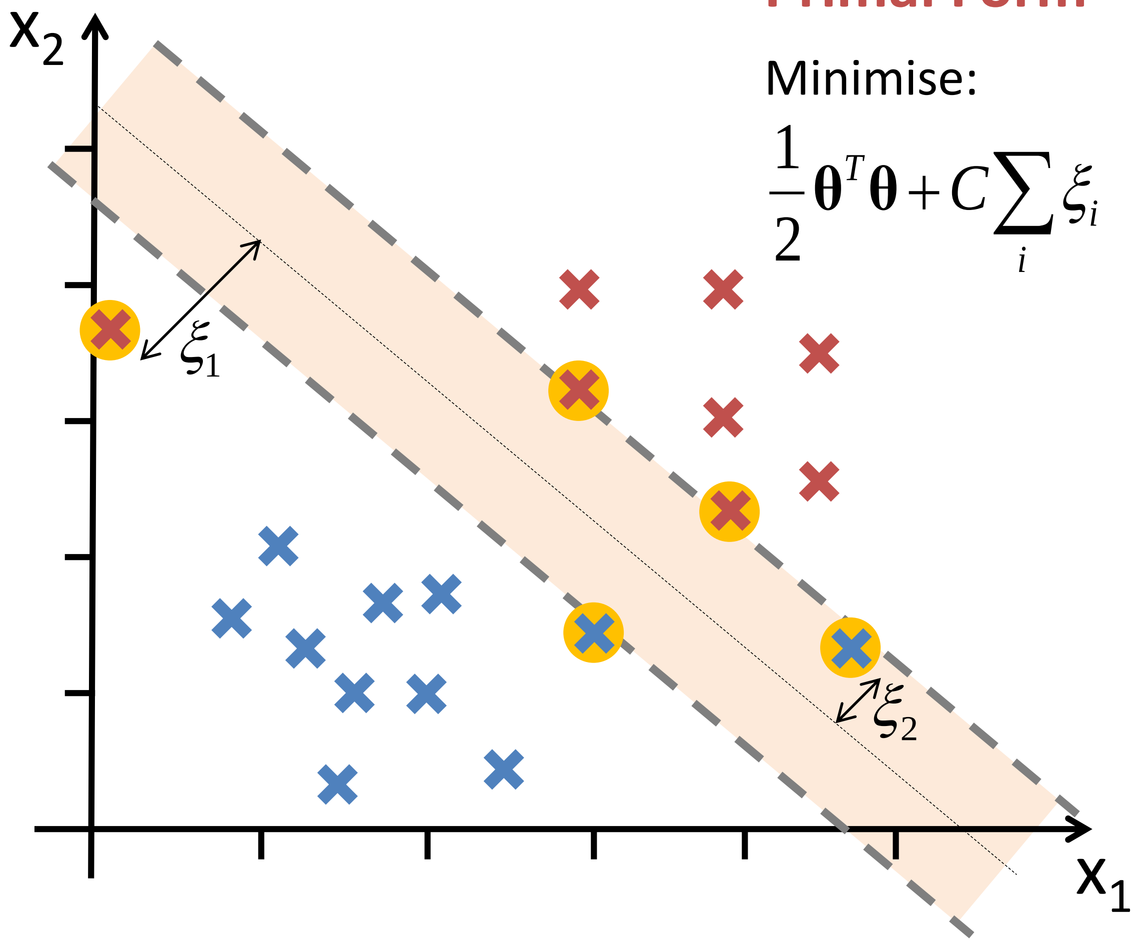
Outliers - Soft Margin

$$\min_{\boldsymbol{\theta}} C \sum_{i=1}^{m} \left[y^{(i)} \mathrm{cost}_1\left(\boldsymbol{\theta}^T \mathbf{x}\right) + \left(1 - y^{(i)}\right) \mathrm{cost}_0\left(\boldsymbol{\theta}^T \mathbf{x}\right) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

$$C = \frac{1}{\lambda}$$



Outliers - Soft Margin



Primal Form

Minimise:

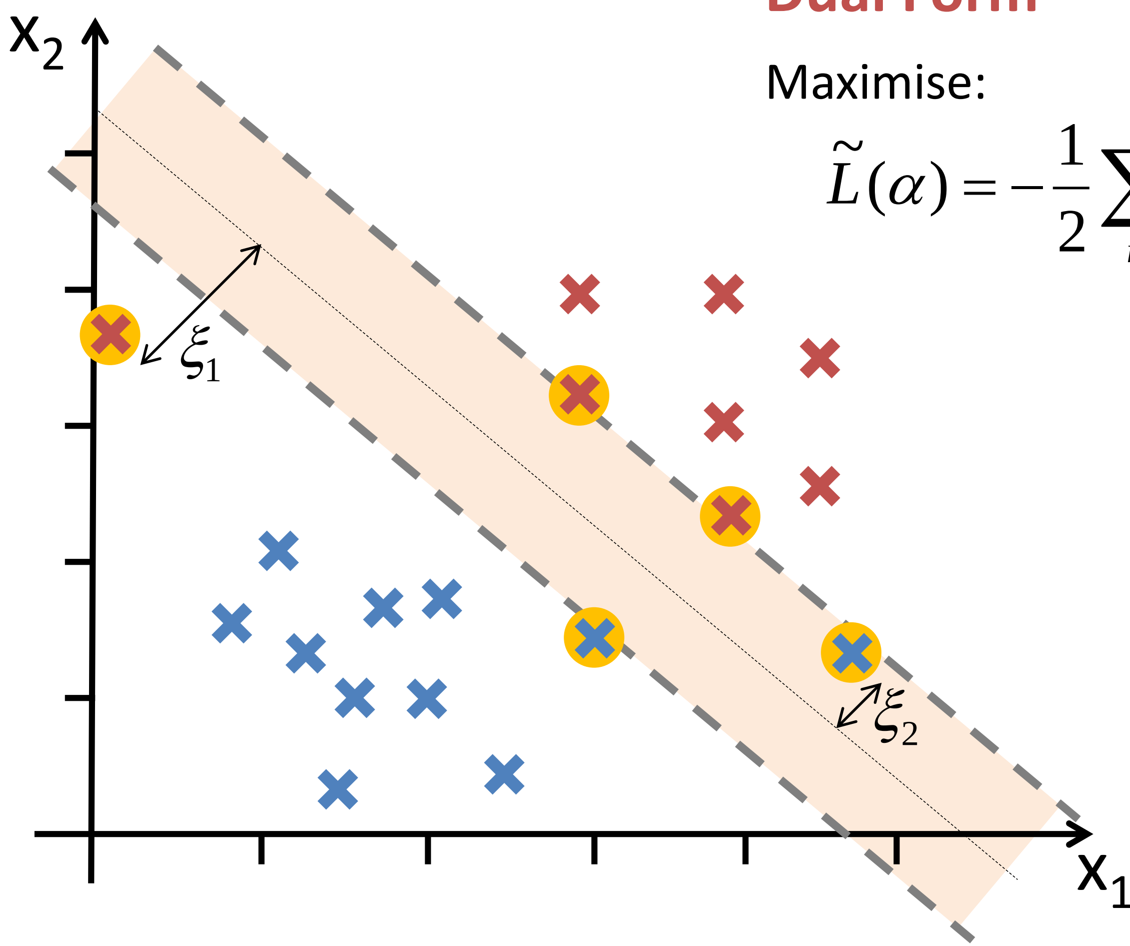
$$\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta} + C\sum_i \xi_i$$

Subject to:

$$y_i(\boldsymbol{\theta}^T\mathbf{x}_i + \theta_0) \geq 1 - \xi_i$$

The impact of slack variables is **modulated** in the optimization problem by a parameter C, that can be interpreted as a regularization parameter (maximization of margin vs minimization of the accepted error)

Outliers - Soft Margin



Dual Form

Maximise:

$$\tilde{L}(\alpha) = -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i$$

Subject to:

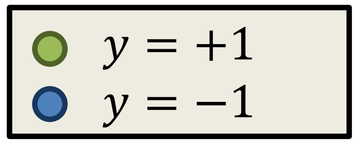
$$\sum_{i=0}^{N} \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

In the dual representation slack variables imply **a new inequality** that regulates the contribution of the vectors

RELATION TO K-NN AND PARZEN WINDOWS

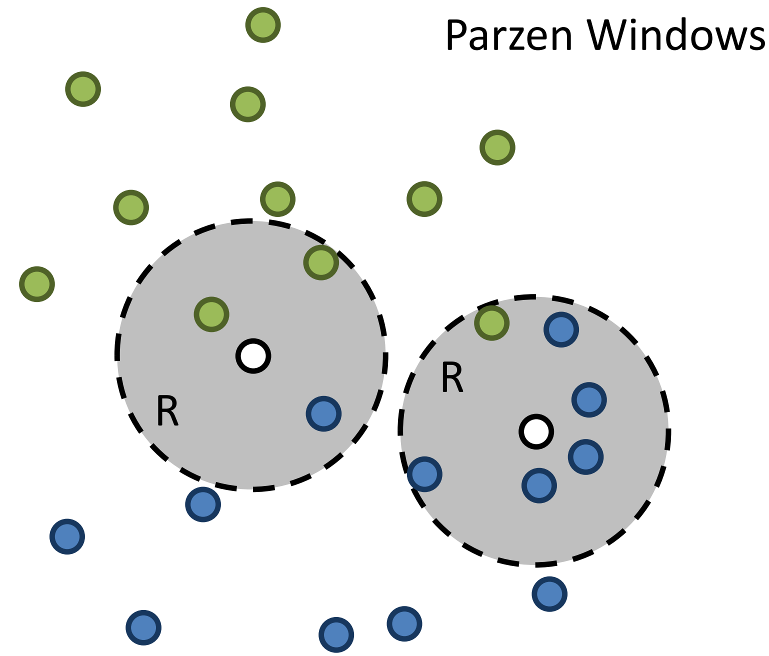
Parzen Windows and Kernels



$$f(x) = sgn\left[\sum_{i:x_i \in R} y_i\right]$$

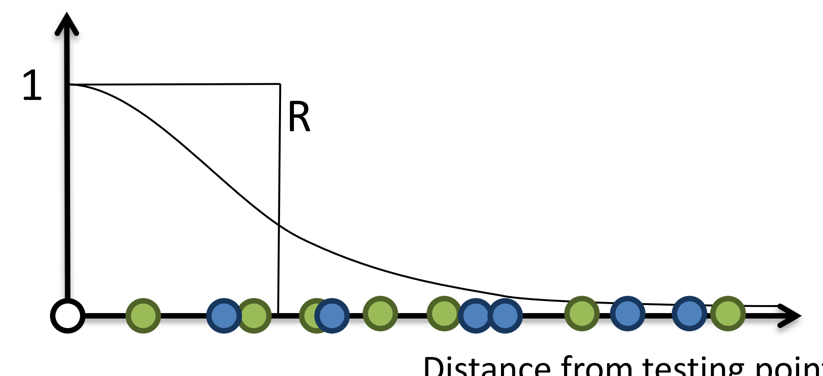
$$= sgn\left[\sum_i y_i \cdot 1_{\|x_i - x\| \leq R}\right]$$

$$= sgn\left[\sum_i y_i K(x_i, x)\right]$$



Compare to kernelized SVM's:

$$sgn\left[\sum_i a_i y_i K(x_i, x)\right]$$



What’s Next

		Mondays	**Tuesdays**				
		16:00 - 18:00	**15:00 - 17:00**				
Practical Sessions		**M**	**T**	**W**	**T**	**F**	**Lectures**
	Feb	8	9	10	11	12	Introduction and Linear Regression
P0. Introduction to Python, Linear Regression		15	16	17	18	19	Logistic Regression, Normalization
P1. Text non-text classification (Logistic Regression)		22	23	24	25	26	Regularization, Bias-variance decomposition
	Mar	29	1	2	3	4	Subspace methods, dimensionality reduction
		7	8	9	10	11	Probabilities, Bayesian inference
Discussion of intermediate deliverables / project presentations		14	15	16	17	18	Parameter Estimation, Bayesian Classification
		21	22	23	24	25	Easter Week
	Apr	28	29	30	31	1	Clustering, Gausian Mixture Models, Expectation Maximisation
P2. Feature learning (k-means clustering, NN, bag of words)		4	5	6	7	8	Nearest Neighbour Classification
		11	12	13	14	15	
		18	19	20	21	22	Kernel methods
Discussion of intermediate deliverables / project presentations		25	26	27	28	29	Support Vector Machines, Support Vector Regression
P3. Text recognition (multi-class classification using SVMs)	**May**	2	3	4	5	6	Neural Networks
		9	10	11	12	13	Advanced Topics: Deep Nets
		16	17	18	19	20	Advanced Topics: Metric Learning, Preference Learning
Final Project Presentations		23	24	25	26	27	Advanced Topics: Structural Pattern Recognition
	Jun	30	31	1	2	3	Revision

LEGEND	
	Project Follow Up
	Project presentations
	Lectures
	Project Deliverable due date
	Vacation / No Class