MATH60026/MATH70026
Methods for Data Science
Lecture 4

Barbara Bravi, Imperial College London

Department of Mathematics, Academic year 2024-2025

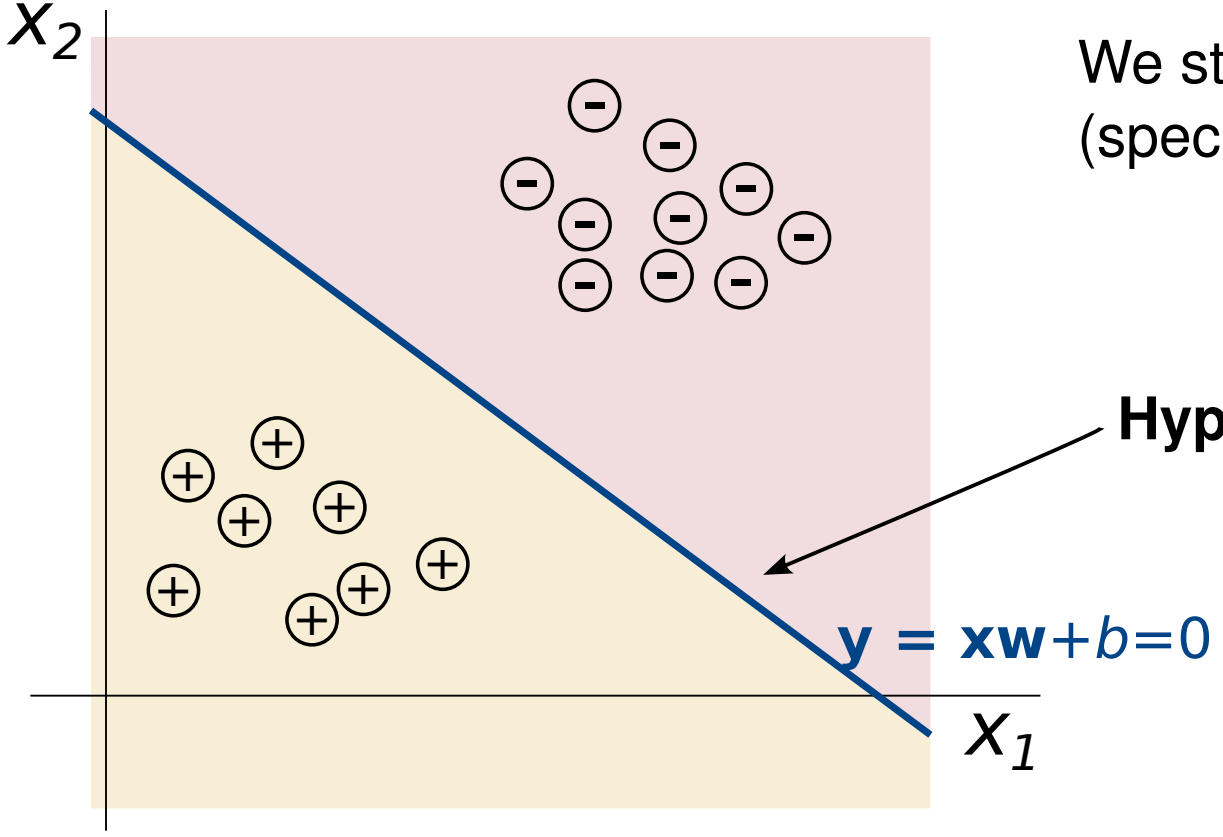
IMPERIAL

Support Vector Machines

Important methods for classification relying on the *geometric* idea of constructing **optimal class-separating hyperplanes**.

Support Vector Machines

Important methods for classification relying on the *geometric* idea of constructing **optimal class-separating hyperplanes**.



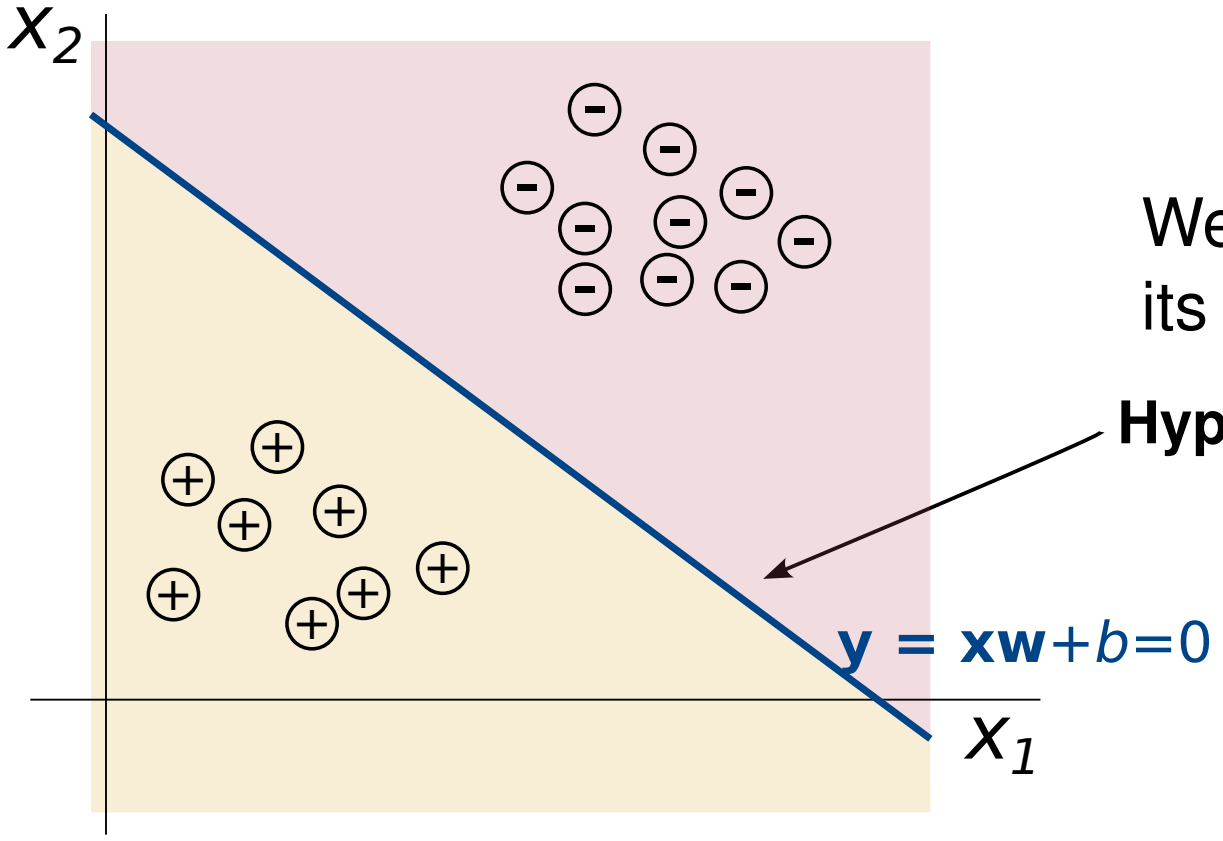
We study it for **binary** classification (specifically, **hard** classification)

The optimisation step here is a geometric construction.

Hard-margin SVM

Binary classification set-up:

$$\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_p^{(i)}) \in \mathbb{R}^p \quad \rightarrow \quad \boxed{y^{(i)} \in \{-1, +1\},} \quad i = 1, \ldots, N$$



We need to learn its parameters

Hyperplane

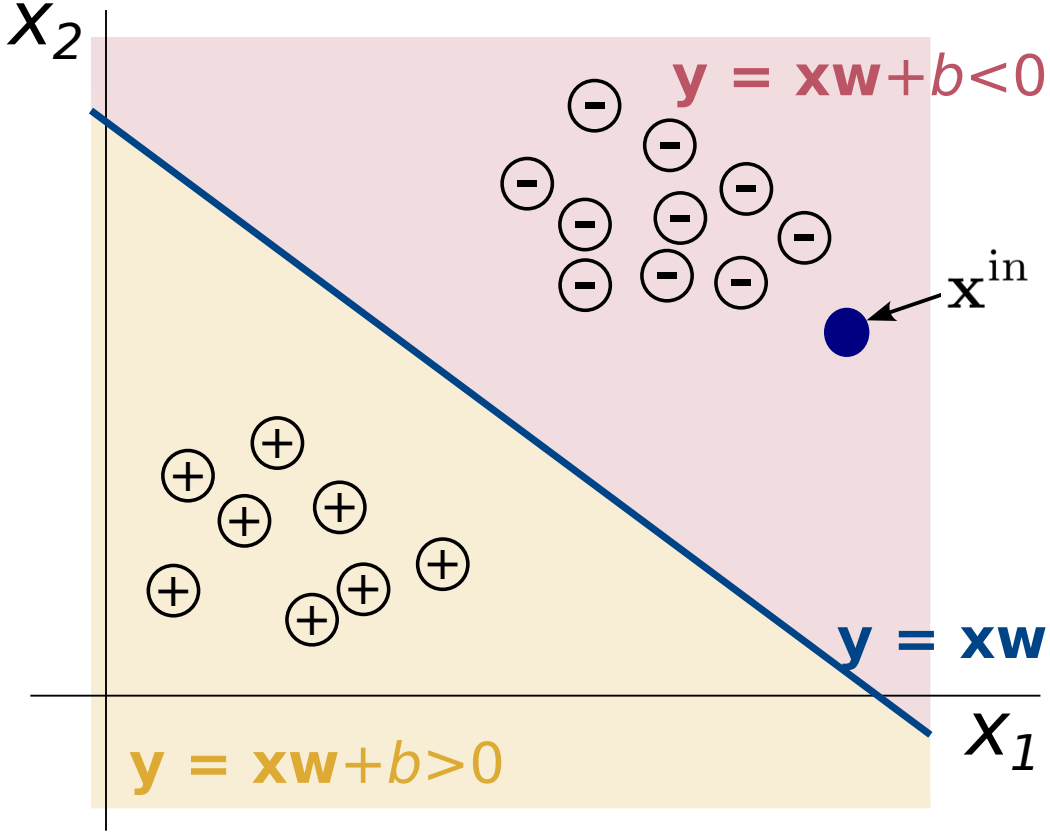
$$\mathbf{x} \cdot \mathbf{w} + b = 0$$

$$\mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix}$$

Hard-margin SVM

Binary classification set-up:

$$\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_p^{(i)}) \in \mathbb{R}^p \quad \rightarrow \quad \boxed{y^{(i)} \in \{-1, +1\},} \quad i = 1, \ldots, N$$



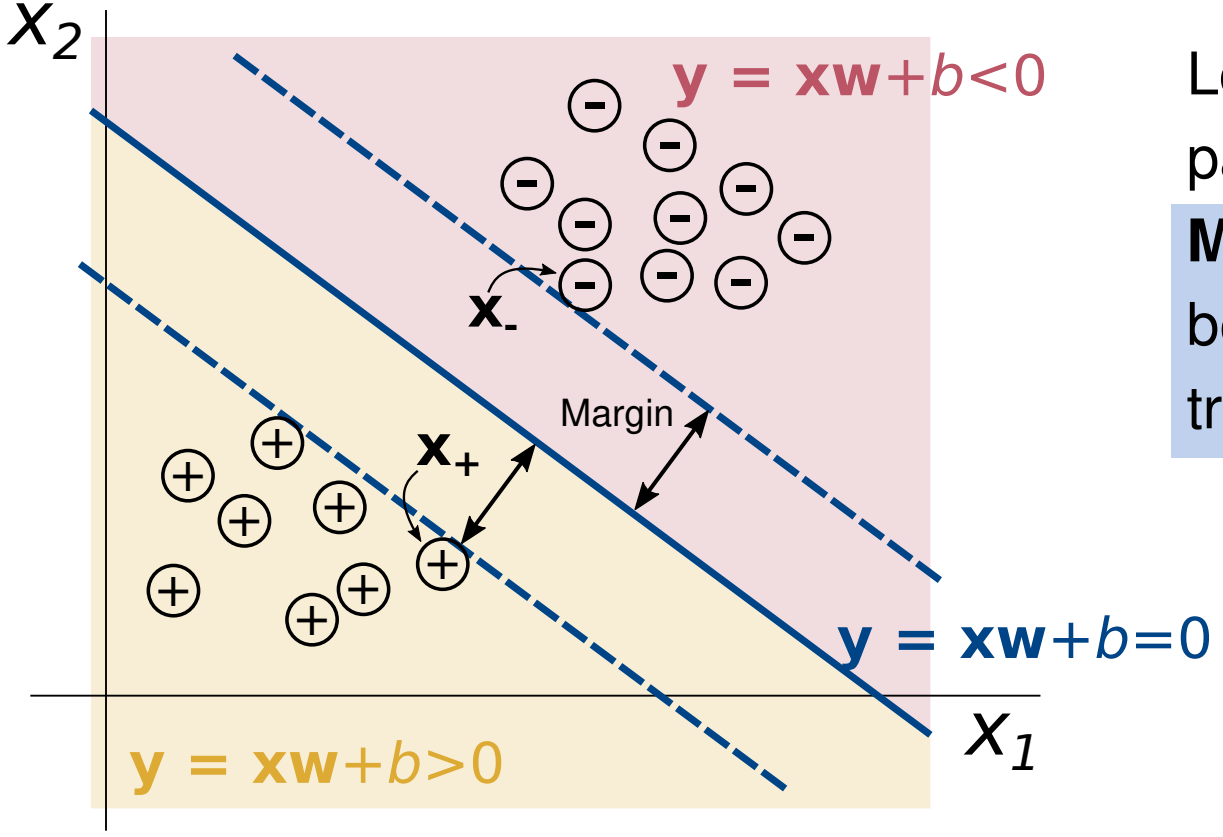
Once learnt, we have a decision boundary, hence the SVM prediction for a given $\mathbf{x}^{\text{in}}$:

$$\begin{cases} \mathbf{x}^{\text{in}} \cdot \mathbf{w} + b \geq 0 & \text{then } \hat{y} = +1 \\ \mathbf{x}^{\text{in}} \cdot \mathbf{w} + b < 0 & \text{then } \hat{y} = -1 \end{cases}$$

Hard-margin SVM

Binary classification set-up:

$$\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_p^{(i)}) \in \mathbb{R}^p \quad \rightarrow \quad \boxed{y^{(i)} \in \{-1, +1\},} \quad i = 1, \ldots, N$$



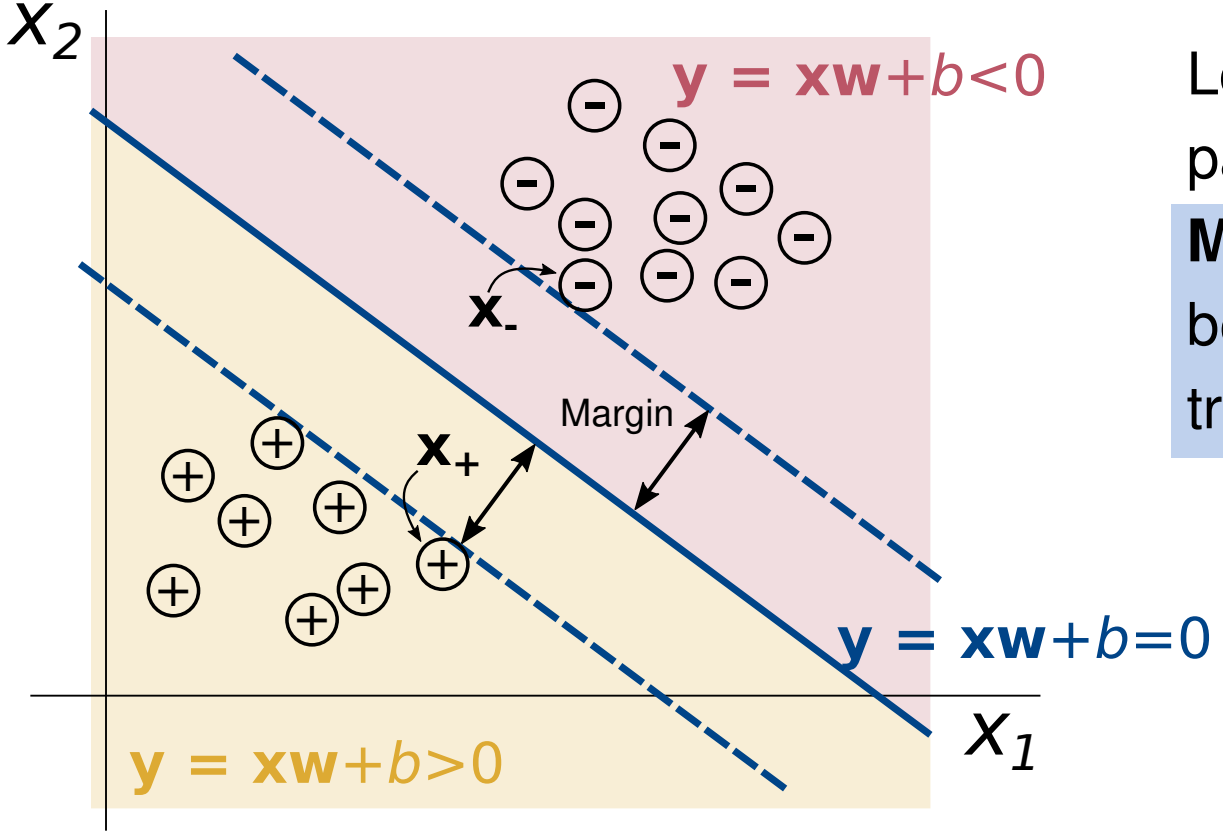
Learning the hyperplane parameters is based on:

MARGIN: smallest distance between the hyper-plane and training data points

Hard-margin SVM

Binary classification set-up:

$$\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_p^{(i)}) \in \mathbb{R}^p \quad \rightarrow \quad \boxed{y^{(i)} \in \{-1, +1\},} \quad i = 1, \ldots, N$$

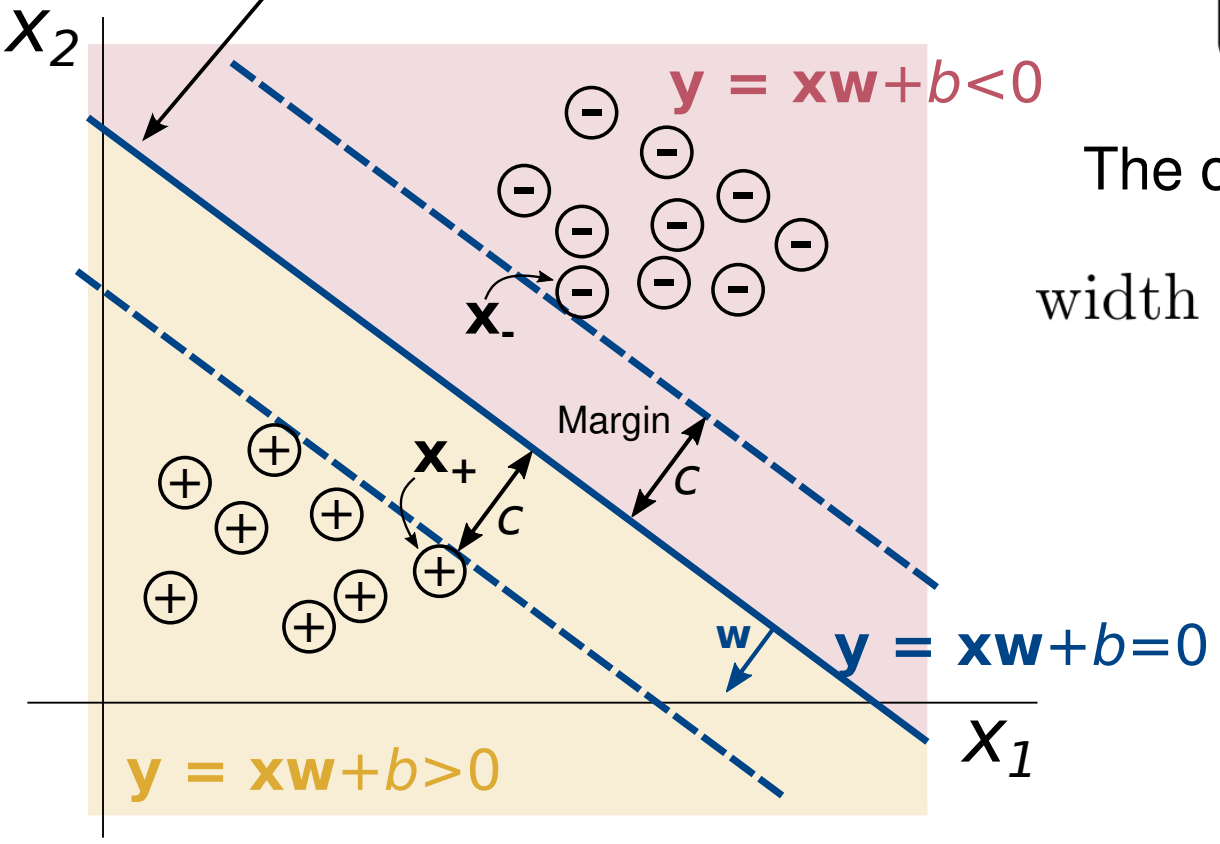


Learning the hyperplane parameters is based on:

MARGIN: smallest distance between the hyper-plane and training data points

The optimal parameters are the ones **maximising the margin.**

Hard-margin SVM



For the points on the margin one has:

$$\begin{cases} \mathbf{x}_+ \cdot \mathbf{w} + b = c \\ \mathbf{x}_- \cdot \mathbf{w} + b = -c \end{cases}$$

The overall width of the margin is:

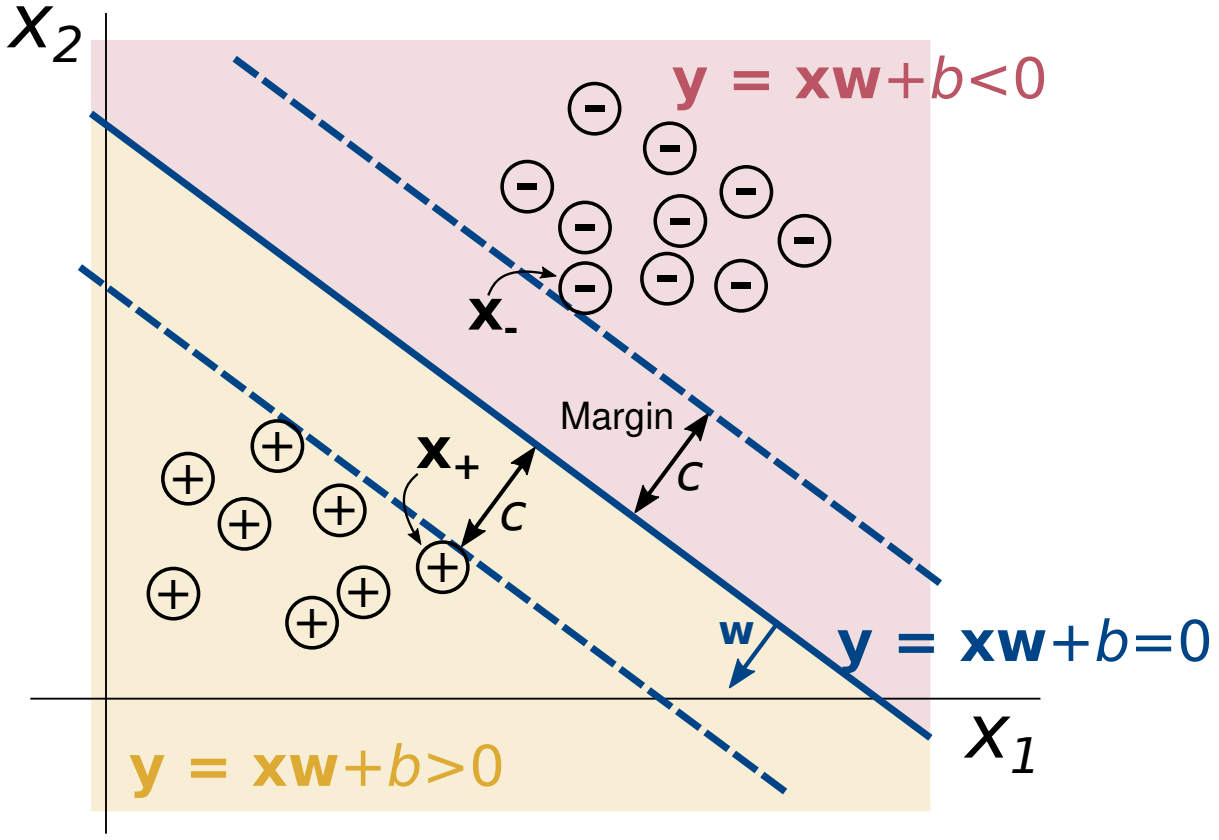
$$\text{width} = (\mathbf{x}_+ - \mathbf{x}_-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2c}{\|\mathbf{w}\|}$$

distance between margin points projected on the direction orthogonal to the hyperplane

$c := 1$ via a simple rescaling

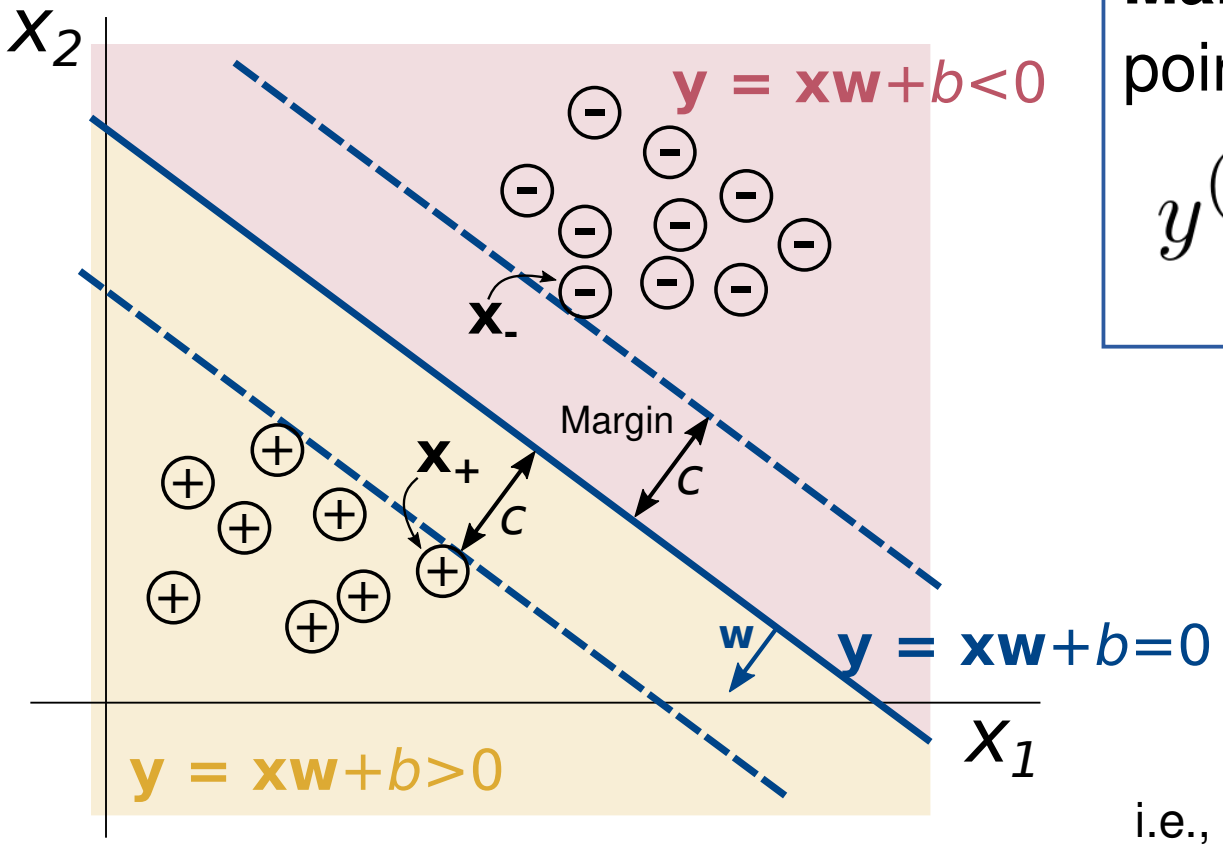
Hard-margin SVM

Finally we get: maximising the margin is about minimising $\|\mathbf{w}\|$



Hard-margin SVM

Finally we get: maximising the margin is about minimising $\|\mathbf{w}\|$



Margin constraint: training points *outside* of the margin

$$y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \geq 1$$

Summarising compactly that:

$(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \geq 1 \quad \text{if} \quad \mathbf{x}^{(i)} \in \{1\}$

$(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \leq -1 \quad \text{if} \quad \mathbf{x}^{(i)} \in \{-1\}$

i.e., all training points should be at least at a distance 1 from the hyperplane.

Hard-margin SVM

Key message: the hard classification SVM optimisation problem is:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \geq 1, \qquad i = 1, \ldots, N \tag{6.4}$$

Hard-margin SVM

Key message: the hard classification SVM optimisation problem is:

$$(6.4) \qquad \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \geq 1, \qquad i = 1, \ldots, N$$

Primal problem

Duality principle: optimisation tells that it can be formulated in 2 specular ways – the primal and dual.

If the primal is a minimisation, like here, the dual problem is a maximisation (and viceversa).

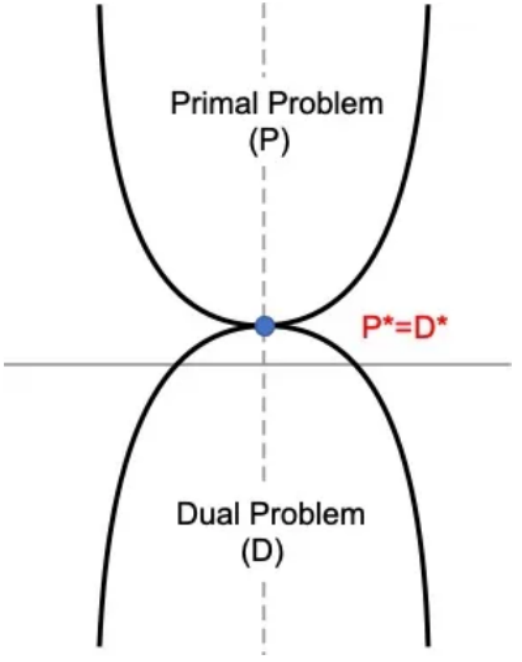


Figure from wikipedia

Hard-margin SVM

Primal problem - a minimisation:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } \underbrace{1 - y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b)}_{g_i(\mathbf{w},b)} \leq 0$$

Inequality constraint

Dual problem - a maximisation:

$$\max_{\boldsymbol{\alpha}} \left\{ \inf_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b; \boldsymbol{\alpha}) \right\} \quad \text{subject to} \quad \alpha_i \geq 0, \forall i$$

where we have defined the Lagrangian function:

$$\mathcal{L}(\mathbf{w}, b; \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \alpha_i \underbrace{\left(1 - y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b)\right)}_{g_i(\mathbf{w},b)}$$

Lagrange multipliers to enforce the constraints

Hard-margin SVM

Dual problem - a maximisation:

$$\max_{\boldsymbol{\alpha}} \left\{ \inf_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b; \boldsymbol{\alpha}) \right\}$$

Hard-margin SVM

Dual problem - a maximisation:

$$\max_{\boldsymbol{\alpha}} \left\{ \inf_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b; \boldsymbol{\alpha}) \right\}$$

First minimise wrt the hyperplane parameter:

$$\left\{ \begin{array}{c} \dfrac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{N} y^{(i)} \alpha_i \\ \\ \nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)} \end{array} \right\} \longrightarrow \left\{ \begin{array}{c} \sum_{i=1}^{N} y^{(i)} \alpha_i = 0 \\ \\ \mathbf{w}^\star = \sum_{i=1}^{N} \alpha_i y^{(i)} \mathbf{x}^{(i)} \end{array} \right\}$$

By substitution in Lagrangian does simplify it:

$$\mathcal{L}(\boldsymbol{\alpha}) \coloneqq \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$$

Hard-margin SVM

Dual problem - a maximisation:

$$\max_{\boldsymbol{\alpha}} \left\{ \inf_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b; \boldsymbol{\alpha}) \right\}$$

Solution stated by the Karush-Kuhn-Tucker (KKT) conditions (beyond this course).
In summary:

$$\begin{aligned} g_i(\mathbf{z}^\star) &\leq 0 \quad \forall i \quad \text{(primal costraints)} \\ \alpha_i &\geq 0 \quad \forall i \quad \text{(dual constraints)} \\ \alpha_i \cdot g_i(\mathbf{z}^\star) &= 0 \quad \forall i \quad \text{(complementary slackness conditions)} \end{aligned}$$

Hard-margin SVM

Dual problem - a maximisation:

$$\max_{\boldsymbol{\alpha}} \left\{ \inf_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b; \boldsymbol{\alpha}) \right\}$$

Solution stated by the Karush-Kuhn-Tucker (KKT) conditions (beyond this course).
In summary:

$$\begin{aligned} g_i(\mathbf{z}^\star) &\leq 0 \quad \forall i \quad \text{(primal costraints)} \\ \alpha_i &\geq 0 \quad \forall i \quad \text{(dual constraints)} \\ \alpha_i \cdot g_i(\mathbf{z}^\star) &= 0 \quad \forall i \quad \text{(complementary slackness conditions)} \end{aligned}$$

Key result: non-zero Lagrange multipliers are there enforce the constraint only for:

$$\alpha_i = 0 \text{ if } \mathbf{x}^{(i)} \neq \mathbf{x}_+ \text{ or } \mathbf{x}_-$$

Recall: $\begin{cases} \mathbf{x}_+ \cdot \mathbf{w} + b = 1 \\ \mathbf{x}_- \cdot \mathbf{w} + b = -1 \end{cases}$

i.e. the training data **on the margin - the *support vectors***

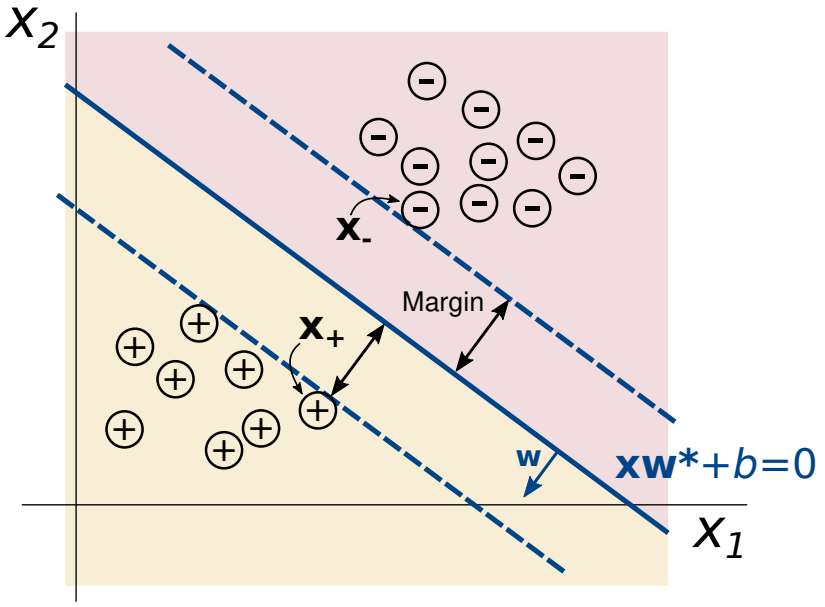
Hard-margin SVM

Key message: Only the two minimally distant points $\mathbf{x}_+$ and $\mathbf{x}_-$ define the vector $\mathbf{w}^\star$ and b of the margin-maximising hyperplane of a hard-margin SVM, that is:

$$\mathbf{w}^\star = \sum_{i=1}^{N} \alpha_i \, y^{(i)} \mathbf{x}^{(i)} = \alpha_+ \mathbf{x}_+ - \alpha_- \mathbf{x}_-$$

$$b = 1 - \mathbf{x}_+ \cdot \mathbf{w}^\star = 1 - \mathbf{x}_+ \cdot [\alpha_+ \mathbf{x}_+ - \alpha_- \mathbf{x}_-] = 1 - \alpha_+ \mathbf{x}_+ \cdot \mathbf{x}_+ + \alpha_- \mathbf{x}_+ \cdot \mathbf{x}_-$$

where α_+ and α_- are determined via the dual-problem formulation. The two points $\mathbf{x}_+$ and $\mathbf{x}_-$ are called the *support vectors* of the hyperplane (hence the name SVM).



Soft-margin SVM

Problem: hard-margin SVM (i.e., where the margin constraint is strictly enforced) can be found only if data are *linearly separable* into classes.

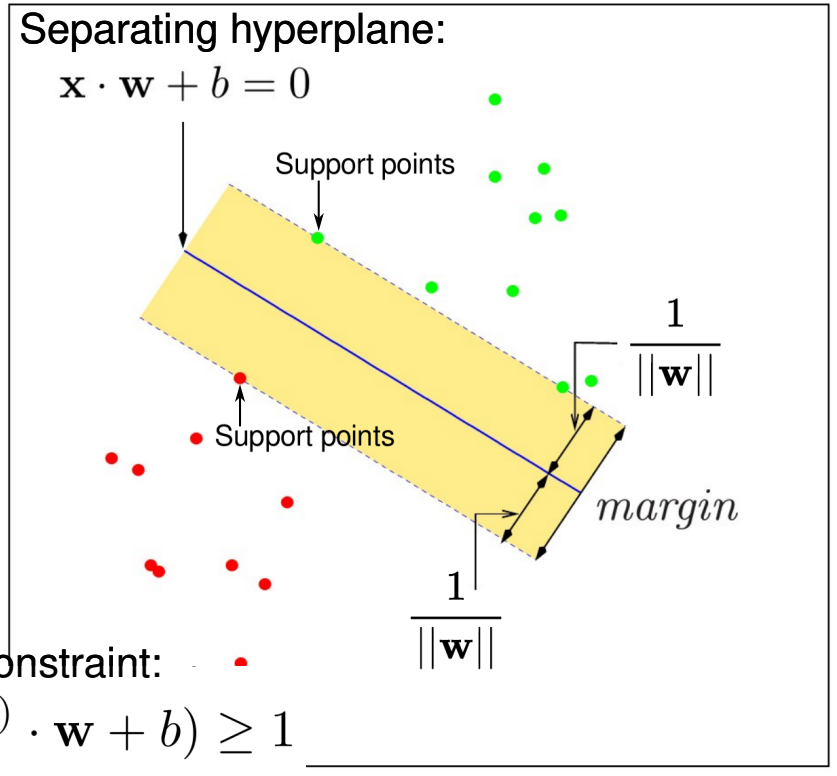
The soft-margin SVM formulation allows us to deal with **imperfect separation** (i.e., non-linearly separable data).

Soft-margin SVM

Problem: hard-margin SVM (i.e., where the margin constraint is strictly enforced) can be found only if data are *linearly separable* into classes.

The soft-margin SVM formulation allows us to deal with **imperfect separation** (i.e., non-linearly separable data).

Linear hard-margin SVM



Margin constraint:

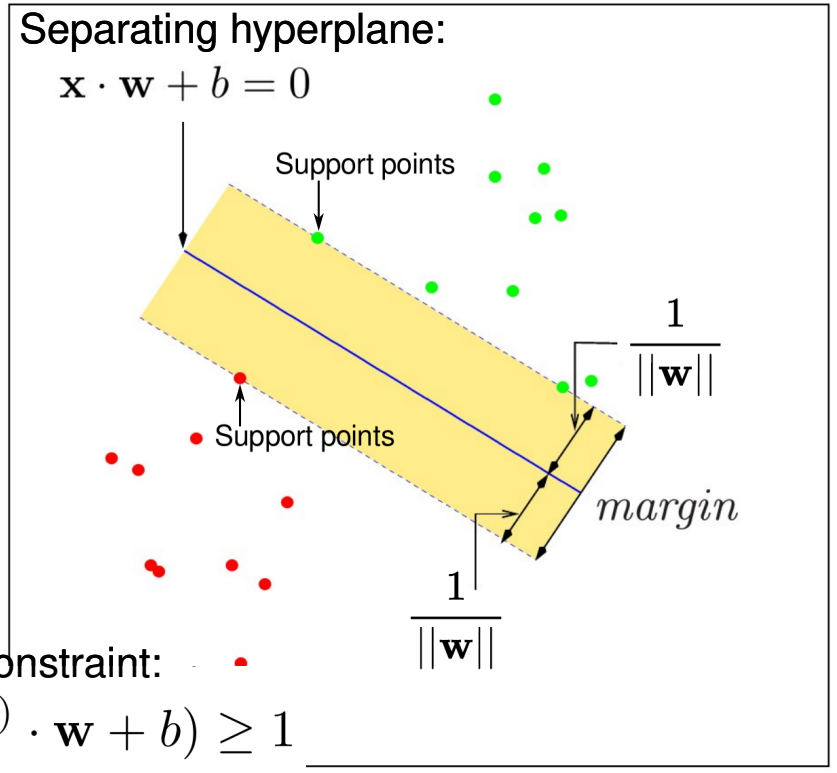
$$y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \geq 1$$

Soft-margin SVM

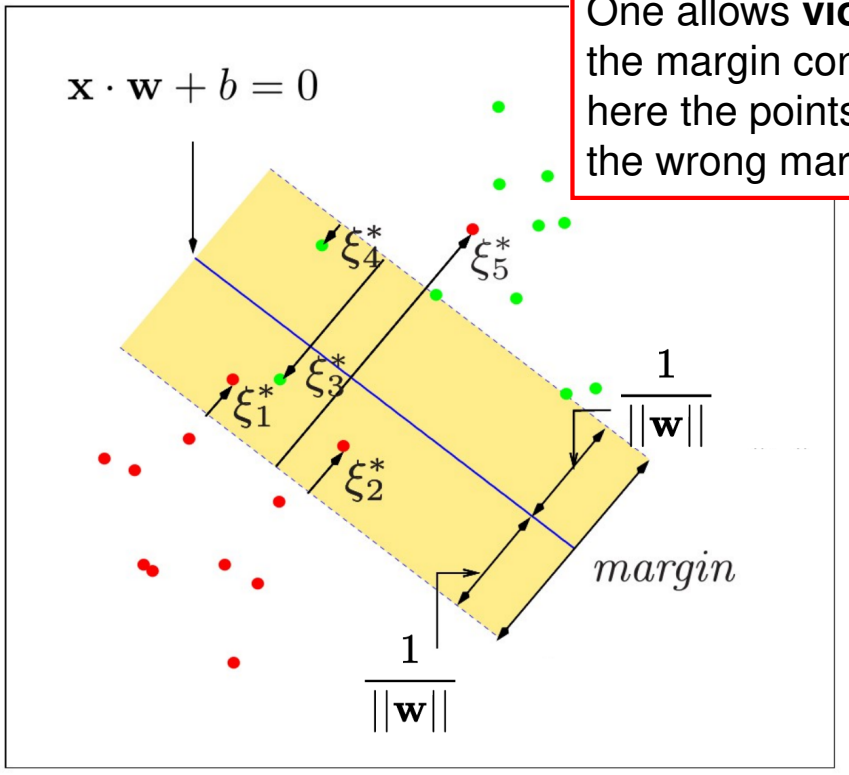
Problem: hard-margin SVM (i.e., where the margin constraint is strictly enforced) can be found only if data are *linearly separable* into classes.

The soft-margin SVM formulation allows us to deal with **imperfect separation** (i.e., non-linearly separable data).

Linear hard-margin SVM



Linear soft-margin SVM



One allows **violations** of the margin constraint: here the points ξ_j^*, on the wrong margin side

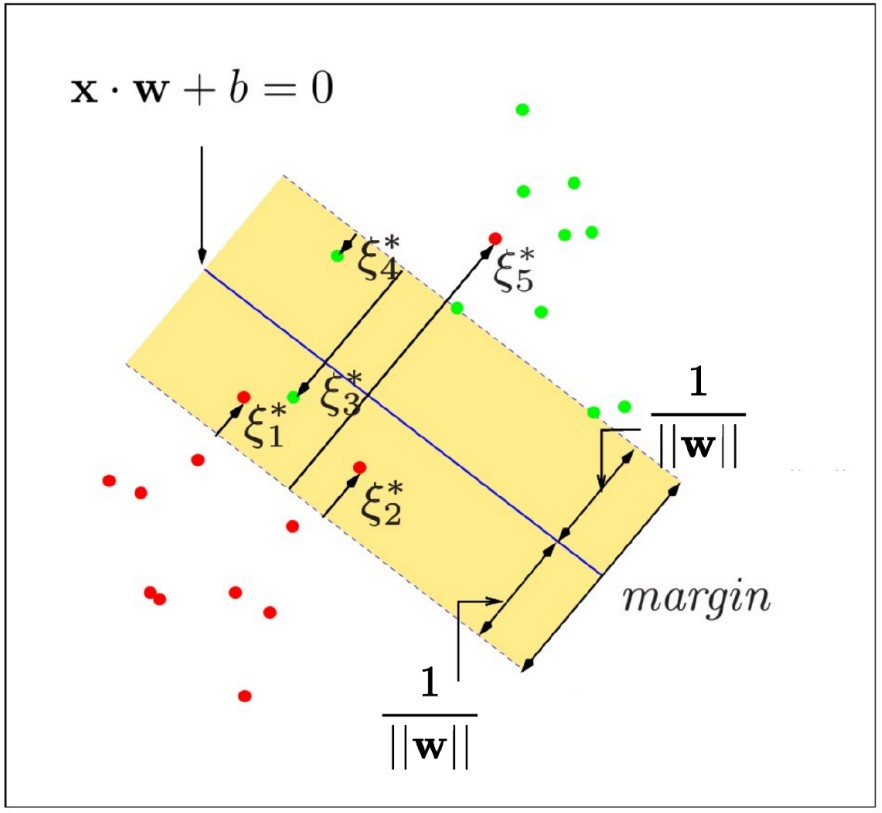
Margin constraint:

$$y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \geq 1$$

Figures adapted from Hastie Tibshirani Friedman, Elements of Statistical Learning, chap. 12

Soft-margin SVM

Linear soft-margin SVM



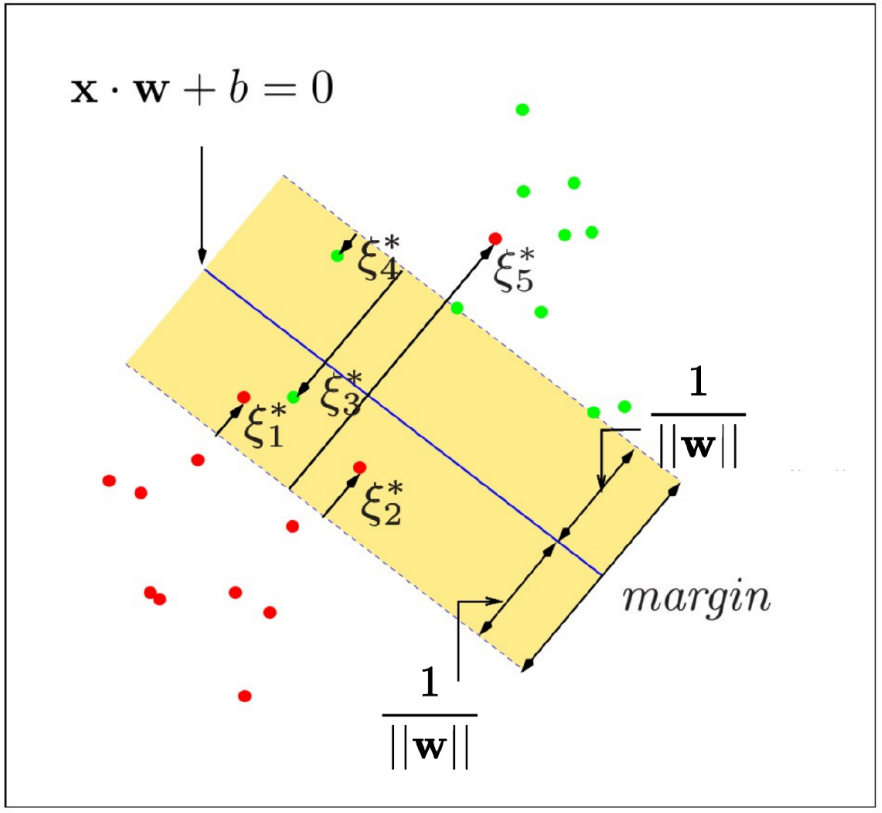
Mathematically, allowing violations = **relaxed margin constraint**

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

Size of the violations

Soft-margin SVM

Linear soft-margin SVM



Mathematically, allowing violations = **relaxed margin constraint**

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

Size of the violations

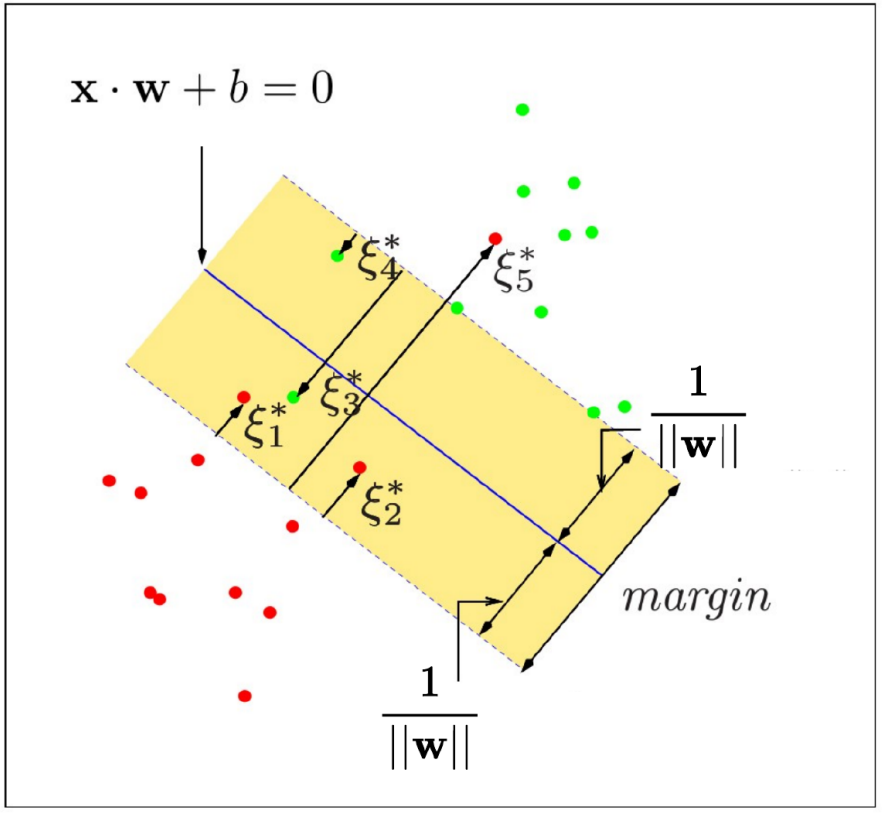
In the optimisation one allows violations of the margin constraint while setting a penalty on the size of such violations:

Key message: The optimisation to learn the soft-margin SVM is:

$$\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{N} \xi_i \right) \text{ subject to } 1 - y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \leq \xi_i \text{ for } i = 1, \ldots, N$$

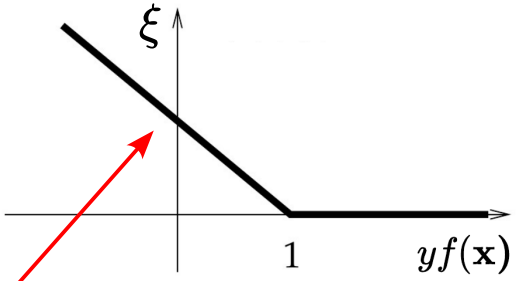
Soft-margin SVM

Linear soft-margin SVM



The violations are modelled through the HINGE LOSS:

$$\xi_i = \max\left\{0, \underbrace{1 - (\mathbf{x}^{(i)} \cdot \mathbf{w} + b)y^{(i)}}_{\text{size of violation}}\right\}$$



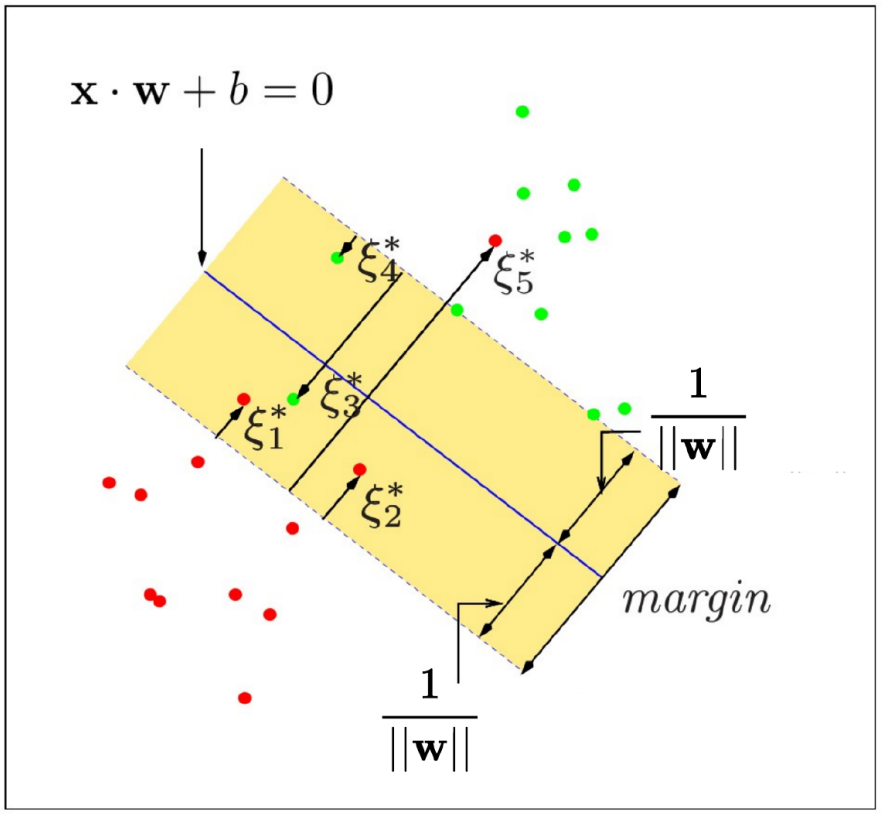
it's non-zero only for violation points

Key message: The optimisation to learn the soft-margin SVM is:

$$\min_{\mathbf{w},b}\left(\frac{1}{2}\|\mathbf{w}\|^2 + \lambda\sum_{i=1}^{N}\xi_i\right) \text{ subject to } 1 - y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \leq \xi_i \text{ for } i = 1, \ldots, N$$

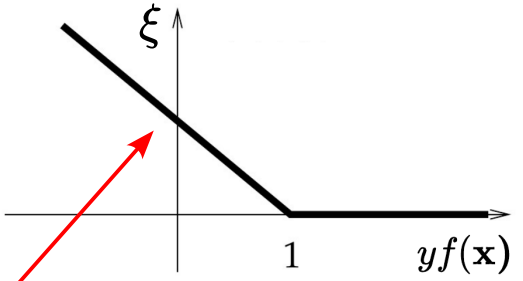
Soft-margin SVM

Linear soft-margin SVM



The violations are modelled through the HINGE LOSS:

$$\xi_i = \max\left\{0,\ \underbrace{1 - (\mathbf{x}^{(i)} \cdot \mathbf{w} + b)y^{(i)}}_{\text{size of violation}}\right\}$$



it's non-zero only for violation points

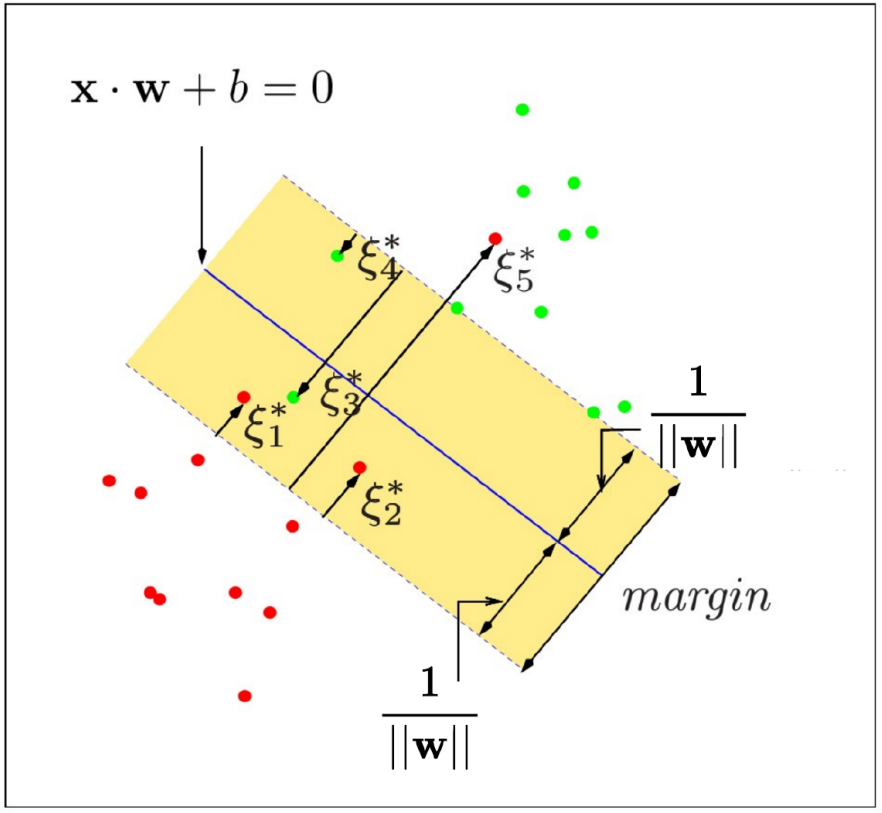
Hardness hyperparameter: controls how large violations are

Key message: The optimisation to learn the soft-margin SVM is:

$$\min_{\mathbf{w},b}\left(\frac{1}{2}\|\mathbf{w}\|^2 + \lambda\sum_{i=1}^{N}\xi_i\right) \text{ subject to } 1 - y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \leq \xi_i \text{ for } i = 1, \ldots, N$$

Soft-margin SVM

Linear soft-margin SVM



To solve the optimisation (not shown): dual formulation + KKT conditions.

Result: one has support vectors - points with non-zero Lagrange multipliers - when

$$y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) = 1 - \xi_i$$

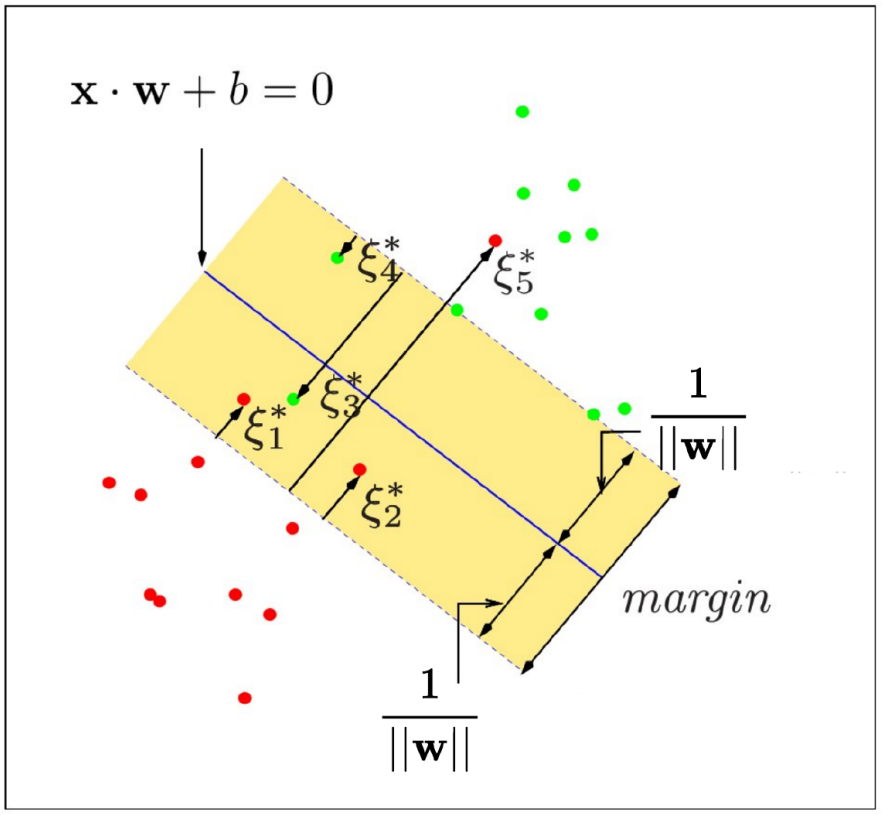
One has:

$\xi_i = 0$ **On-margin support vectors**

Correctly classified

Soft-margin SVM

Linear soft-margin SVM



To solve the optimisation (not shown): dual formulation + KKT conditions.

Result: one has support vectors - points with non-zero Lagrange multipliers - when

$$y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) = 1 - \xi_i$$

One has:

$\xi_i = 0$ **On-margin support vectors**

Correctly classified

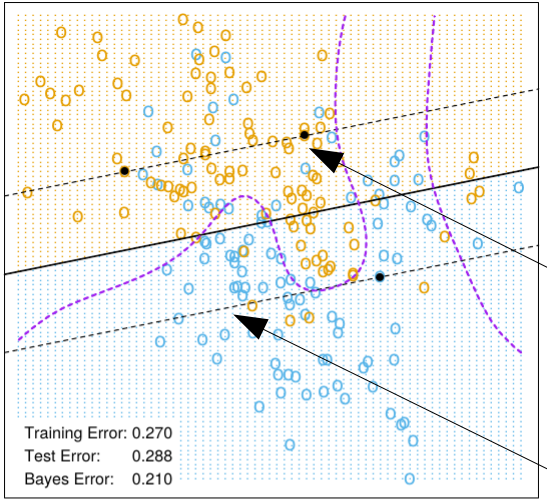
Non-margin support vectors:

$0 < \xi_i < 1$ $\qquad$ $\xi_i > 1$

Correctly classified
Like points 1, 2 in figure

Incorrectly classified
Like point 5 in figure

Linear soft-margin SVM

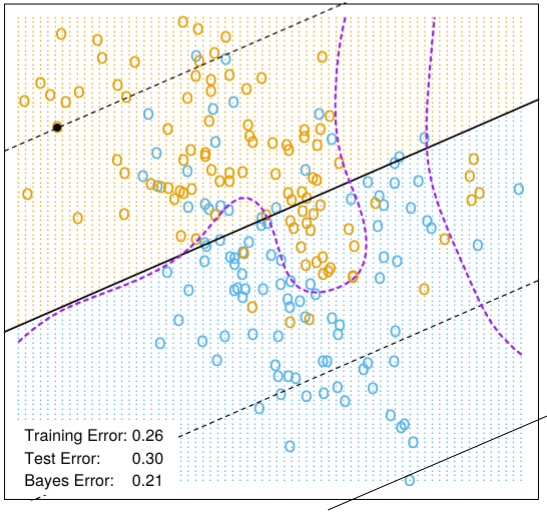


$C = 10000$

All points on the wrong side of the margin are support vectors (62%)

On-margin support vectors

Margin



$C = 0.01$

85% are support vectors

hardness hyperparameter

Figures from Hastie Tibshirani Friedman, Elements of Statistical Learning, chap. 12

Remark on notation

We arrived at the optimisation problem by a **geometric construction**:

See Hastie Tibshirani Friedman, Elements of Statistical Learning, chap. 12

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \quad \text{subject to} \quad \xi_i \geq 0,\ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i$$

Put a constraint on the size of the violation

Find the hyper-plane maximising the margin

But we can see this as a **penalised-loss method**:

$$\min_{\beta_0,\,\beta} \sum_{i=1}^{N}[1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2$$

Loss function (hinge)

Penalising large values of the parameters (like in Ridge)

The two formulations are equivalent (with $\lambda = 1/C$), they simply come from a different derivation

Kernelised SVM

To extend further SVMs to non-linearly separable data.

They rely on *kernel* functions, the intuition being:

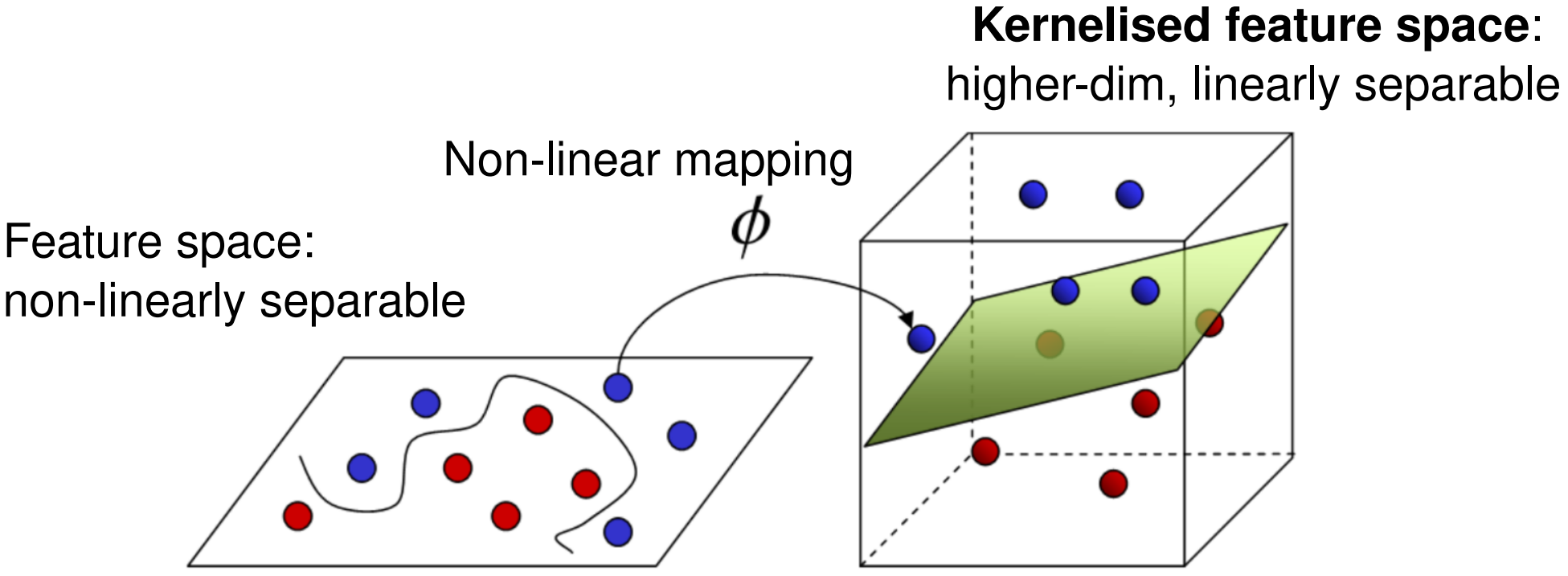


Illustration from towardsdatascience.com

Kernel functions

A kernel function *k* serves to map a pair of *d*-dim. vectors into the dot product of non-linearly transformed, *D*-dim. vectors, *D>>d*.

$$k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y})$$

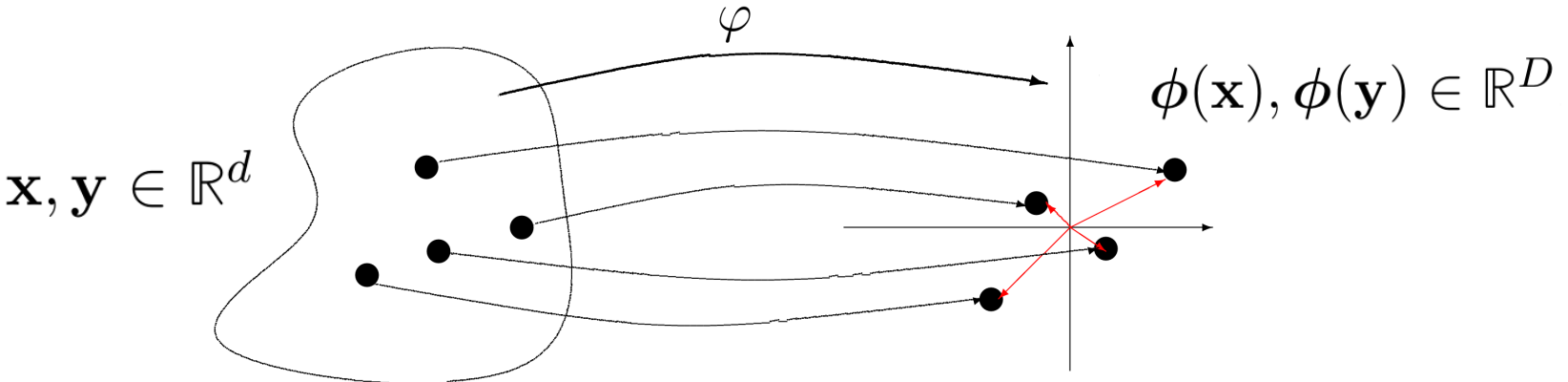
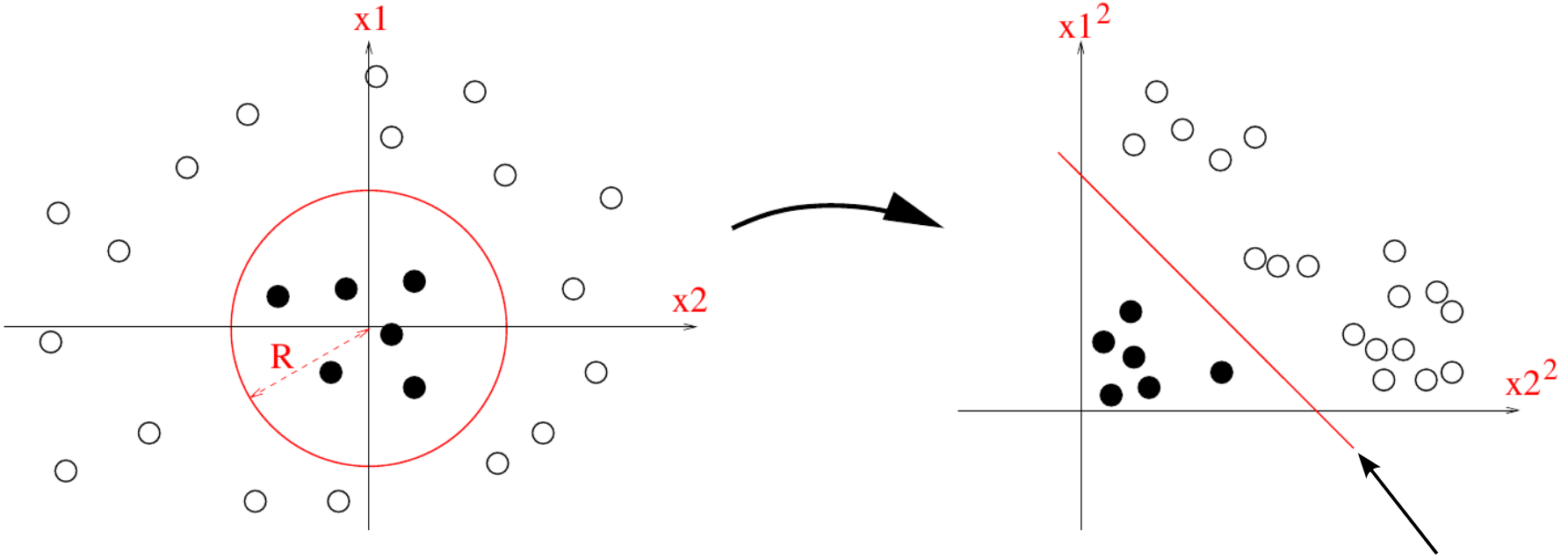


Figure from Julien Mairal & Jean-Philippe Vert, Machine Learning with Kernel Methods

In other words, it stands for the dot product in the *D*-dim. space: as such, it has to be positive semi-definite.

Example: the polynomial kernel

Figure from Julien Mairal & Jean-Philippe Vert, Machine Learning with Kernel Methods



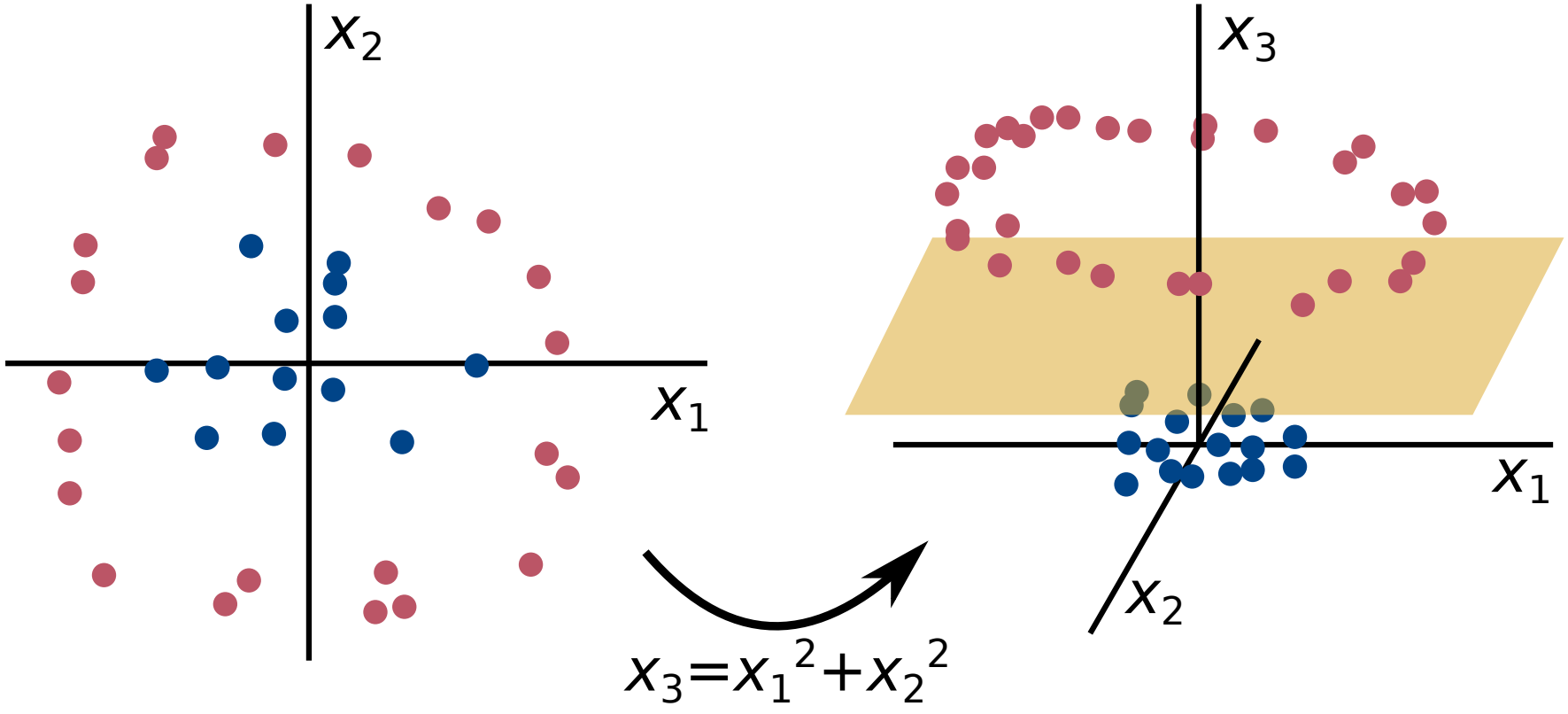
Data lifted in the higher-dimensional space become linearly separable!

$$\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$$

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \in \mathbb{R}^3$$

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) &:= (\mathbf{x} \cdot \mathbf{y})^2 = (x_1y_1 + x_2y_2)^2 \\ &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y}) \end{aligned}$$

Example: the polynomial kernel



$$\phi(\mathbf{x}) = (x_1, x_2, x_3) \text{ with } x_3 = x_1^2 + x_2^2$$

See also this nice animation:
https://www.youtube.com/watch?v=OdlNM96sHio

Kernel trick

We don't need to compute the non-linear mapping ϕ to lift the data vectors to *D*-dimensions, if they feature only in dot products.

Kernel trick (or *kernel substitution*): an algorithm defined for d-dimensional vectors that can be formulated only in terms of pairwise dot products can be applied to non-linearly transformed, D-dimensional vectors (with $D >> d$ and potentially infinite) by replacing each dot product by a kernel function.

Kernel trick

We don't need to compute the non-linear mapping ϕ to lift the data vectors to *D*-dimensions, if they feature only in dot products.

> **Kernel trick** (or *kernel substitution*): an algorithm defined for d-dimensional vectors that can be formulated only in terms of pairwise dot products can be applied to non-linearly transformed, D-dimensional vectors (with $D >> d$ and potentially infinite) by replacing each dot product by a kernel function.

SVM is such an algorithm, both in the primal and dual formulation:

$$\frac{1}{2}\underbrace{\mathbf{w}\cdot\mathbf{w}}_{\star}+\lambda\sum_{i=1}^{N}\max\left\{0,1-y^{(i)}(\underbrace{\mathbf{x}^{(i)}\cdot\mathbf{w}}_{\star}+b)\right\}$$

$$\mathcal{L}(\boldsymbol{\alpha})=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y^{(i)}y^{(j)}\underbrace{\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)}}_{\star}$$

Inner products

Kernelised SVM

If we apply the kernel trick, we obtain, for a hard-margin SVM:

$$\min_{\mathbf{w}_\phi} \frac{1}{2}\|\mathbf{w}_\phi\|^2 \text{ subject to } 1 - y^{(i)}(\phi(\mathbf{x}^{(i)}) \cdot \mathbf{w}_\phi + b) \leq 0 \text{ for } i = 1, \ldots, N$$

Parameters of the kernelised SVM

Transformed data point

To work only with the kernel, we introduce a *N*-dim vector $\mathbf{u}$:

p x N transpose data matrix

$$\mathbf{w}_\phi \cdot \phi(\mathbf{X}^T) = \mathbf{u}^{\mathrm{T}}\mathbf{K}$$

Gram matrix, containing the kernel evaluations:

$$\mathbf{w}_\phi \cdot \mathbf{w}_\phi = \mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u} \qquad \mathbf{K}_{ij} = \phi(\mathbf{x}^{(i)}) \cdot \phi(\mathbf{x}^{(j)}) \equiv k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

The optimisation over $\mathbf{w}_\phi$ becomes an optimisation over $\mathbf{u}$

Kernelised SVM

Key message: The optimisation to learn a kernelised soft-margin SVM can be formulated as:

$$(6.9)\quad \min_{\mathbf{u},b}\left(\frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u}+\lambda\sum_{i=1}^{N}\xi_i\right) \text{ subject to } 1-y^{(i)}(\mathbf{K}^{(i)}\mathbf{u}+b)\leq\xi_i \text{ for } i=1,\ldots,N$$

where $\mathbf{K}^{(i)}$ denotes row i of the Gram matrix $\mathbf{K}$, and the hinge loss reads:

$$\xi_i = \max\left\{0, 1-y^{(i)}(\mathbf{K}^{(i)}\mathbf{u}+b)\right\}$$

In the notebook, we will see the solution of this optimisation (in the primal formulation) by numerical method we know - **gradient descent.**
However, many implementations work with the dual formulation, through quadratic programming, where the kernel trick is applicable even more neatly as:

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y^{(i)}y^{(j)}\underbrace{\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)}}_{\bigstar}$$

and which is effiicient for very large datasets thanks to the solution's sparsity.

GD and Minibatch SGD

To train an SVM, we need to minimise a loss:

$$L(\mathbf{w}, b; \mathcal{S}) := \left(\frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{|\mathcal{S}|} \xi_i \right) \text{ subject to } 1 - y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \leq \xi_i$$

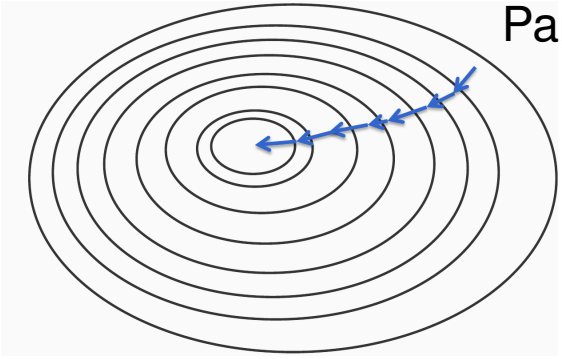
$$\text{Training set: } \mathcal{S} = \{\mathbf{x}^{(i)}, y^{(i)}\},\ |\mathcal{S}| = N^{\text{training}}$$

Gradient Descent (GD):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t; \mathcal{S})$$

Parameters to learn

$$\boldsymbol{\theta} = \{\mathbf{w}, b\}$$



Figures by Vivek Srikumar: https://svivek.com/teaching/lectures/

GD and Minibatch SGD

To train an SVM, we need to minimise a loss:

$$L(\mathbf{w}, b; \mathcal{S}) := \left(\frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{|\mathcal{S}|} \xi_i \right) \text{ subject to } 1 - y^{(i)}(\mathbf{x}^{(i)} \cdot \mathbf{w} + b) \leq \xi_i$$

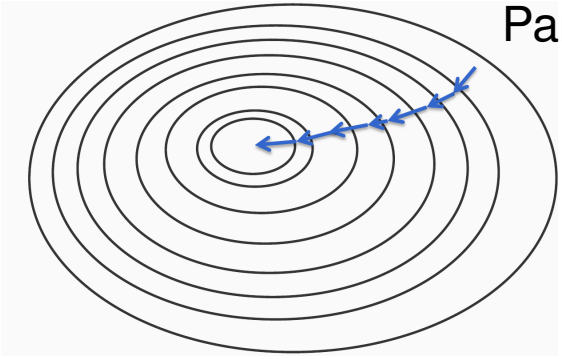
$$\text{Training set: } \mathcal{S} = \{\mathbf{x}^{(i)}, y^{(i)}\},\ |\mathcal{S}| = N^{\text{training}}$$

Gradient Descent (GD):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t; \mathcal{S})$$

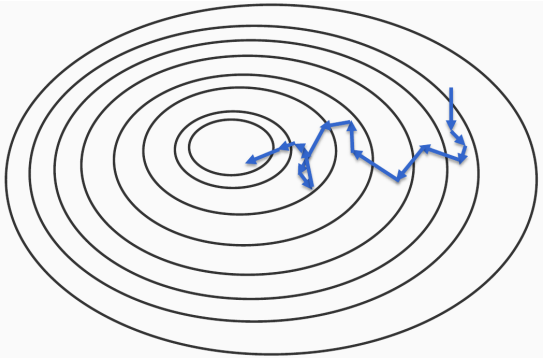
Parameters to learn

$$\boldsymbol{\theta} = \{\mathbf{w}, b\}$$



Minibatch Stochastic GD (SGD):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t; \mathcal{S}_m)$$



Minibatch of training data:

$$|\mathcal{S}_m| \ll |\mathcal{S}|$$

Simple SGD:

$$|\mathcal{S}_m| = 1$$

It is a **stochastic approximation** of GD: minibatch SGD involves more iterations, but each less computationally expensive

Figures by Vivek Srikumar: https://svivek.com/teaching/lectures/

Standard Kernels

- Polynomial (of degree up to n):

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^n$$

- Polynomial of degree n:

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^n$$

- Gaussian kernel (or radial basis function kernel):

$$k(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma}}$$

- Sigmoid:

$$k(\mathbf{x}, \mathbf{y}) = \tanh\left(\beta(\mathbf{x} \cdot \mathbf{y}) + c\right)$$

+ one can construct new ones by combination

Standard Kernels

- Polynomial (of degree up to n):

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^n$$

- Polynomial of degree n:

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^n$$

- Gaussian kernel (or radial basis function kernel):

$$k(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma}}$$

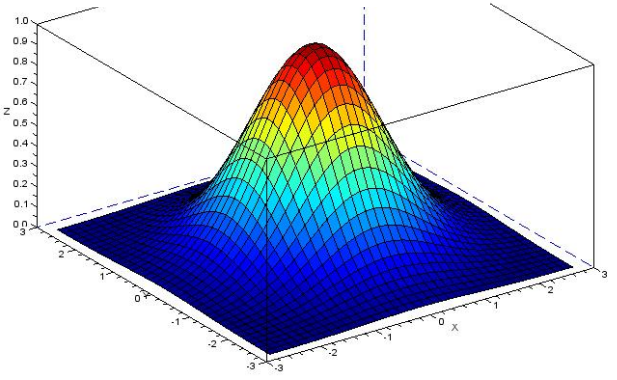
- Sigmoid:

$$k(\mathbf{x}, \mathbf{y}) = \tanh\left(\beta(\mathbf{x} \cdot \mathbf{y}) + c\right)$$

+ one can construct new ones by combination

with hyperparameters to set, e.g.

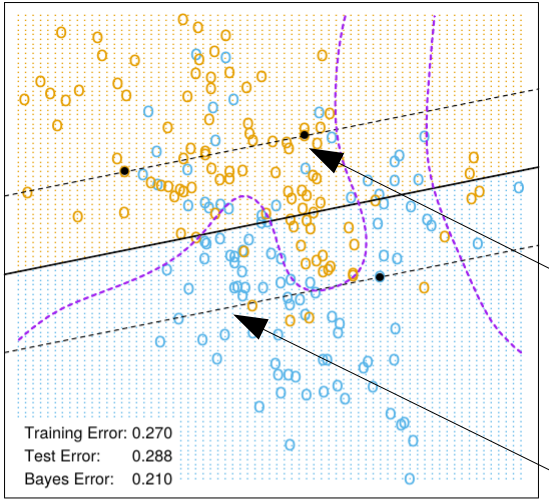
$$k(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\sigma}}$$



$\sigma << 1$ weight the x-y distance

$\sigma >> 1$ flatten all elements to 1

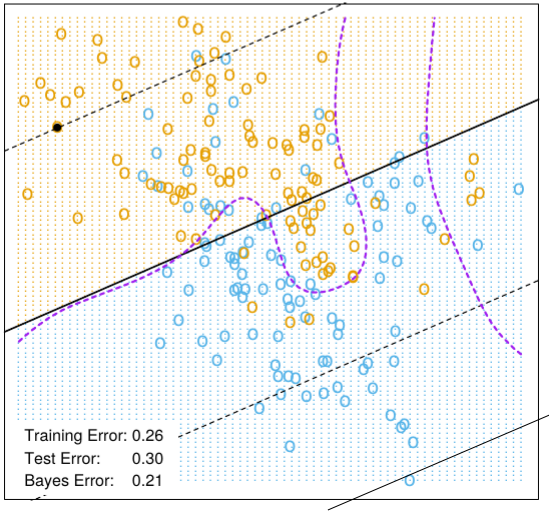
Linear soft-margin SVM



$C = 10000$

On-margin support vectors

Margin



$C = 0.01$

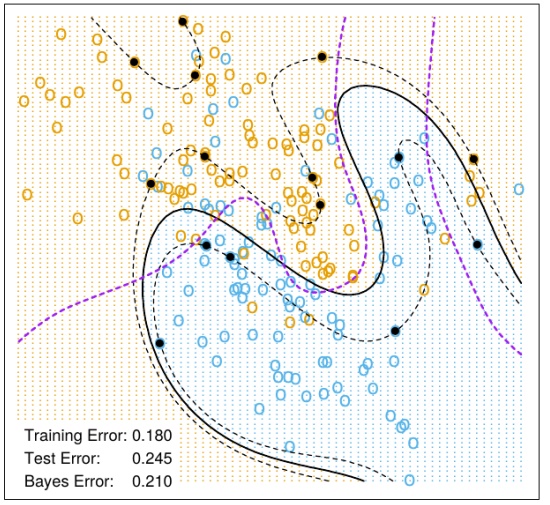
hardness hyperparameter

Kernelised soft-margin SVM

(both with: $C = 1$)

Polynomial

SVM - Degree-4 Polynomial in Feature Space

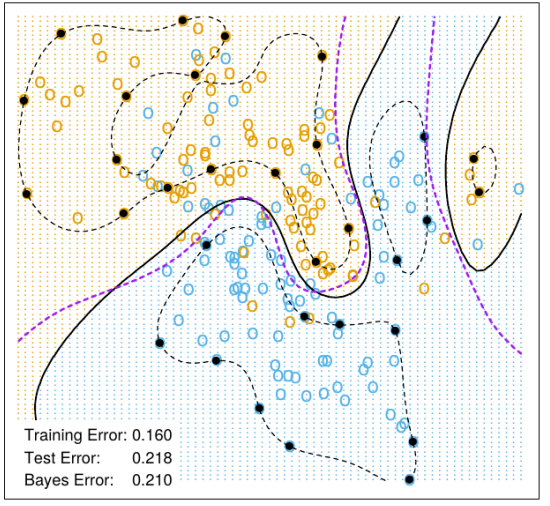


Produce highly non-linear decision boundaries!

Gaussian

$\sigma = 1$

SVM - Radial Kernel in Feature Space



Figures from Hastie Tibshirani Friedman, Elements of Statistical Learning, chap. 12

Quick recap

Basic idea: classification = finding the optimally separating hyperplane.

The optimisation problem is formulated in terms of maximising the ‘width of the margin’. Its solution (parameters of the optimal hyperplane) are written in terms of the support vectors, i.e., the data points closest to the hyperplane on both sides.

Quick recap

Basic idea: classification = finding the optimally separating hyperplane.

The optimisation problem is formulated in terms of maximising the ‘width of the margin’. Its solution (parameters of the optimal hyperplane) are written in terms of the support vectors, i.e., the data points closest to the hyperplane on both sides.

Soft-margin SVM: deals with imperfect separation by allowing yet penalising margin violations.

Quick recap

Basic idea: classification = finding the optimally separating hyperplane.

The optimisation problem is formulated in terms of maximising the ‘width of the margin’. Its solution (parameters of the optimal hyperplane) are written in terms of the support vectors, i.e., the data points closest to the hyperplane on both sides.

Soft-margin SVM: deals with imperfect separation by allowing yet penalising margin violations.

Kernel trick (by which dot products are replaced by kernel functions)
$\rightarrow$ kernelised SVM, which produces nonlinear boundaries by constructing a linear boundary in a higher-dimensional, transformed version of the feature space.

Quick recap

Basic idea: classification = finding the optimally separating hyperplane.

The optimisation problem is formulated in terms of maximising the ‘width of the margin’. Its solution (parameters of the optimal hyperplane) are written in terms of the support vectors, i.e., the data points closest to the hyperplane on both sides.

Soft-margin SVM: deals with imperfect separation by allowing yet penalising margin violations.

Kernel trick (by which dot products are replaced by kernel functions)
$\rightarrow$ kernelised SVM, which produces nonlinear boundaries by constructing a linear boundary in a higher-dimensional, transformed version of the feature space.

There are extensions of many well-known algorithms by the kernel trick, e.g., we will see PCA (provided that the data vectors enter only via dot products).