Lecture 8. Support Vector Machines

COMP90051 Statistical Machine Learning

Lecturer: Jean Honorio

THE UNIVERSITY OF MELBOURNE

This lecture

- Support vector machines (SVMs) as maximum-margin classifiers
- The hard-margin SVM objective
- SVM objective as regularised loss function
- The soft-margin SVM

Maximum-Margin Classifier: Motivation

A new twist to binary linear classification.

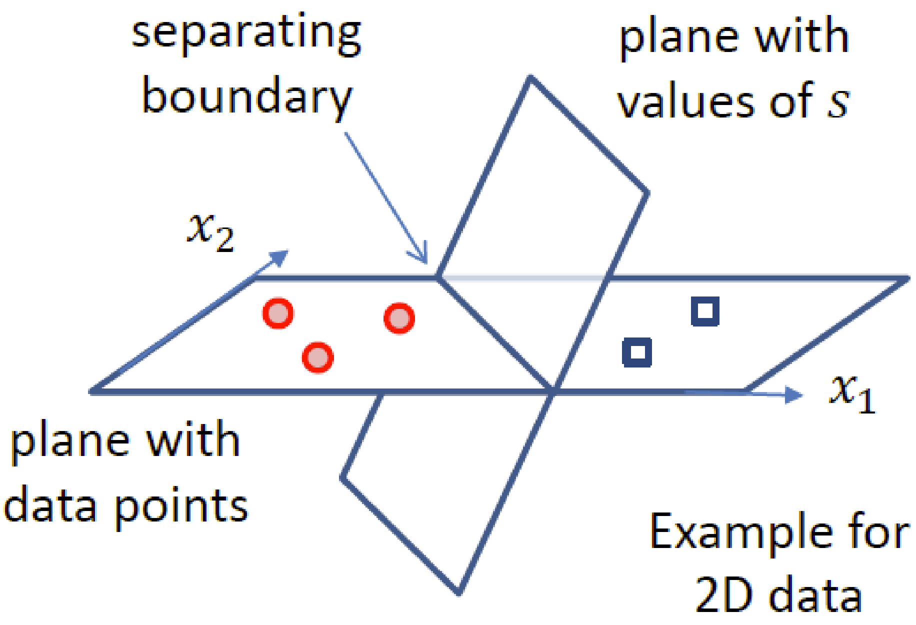
Beginning: linear SVMs

- In the first part, we will consider a basic setup of SVMs, something called linear *hard-margin SVM*.
- Keep in mind: SVMs are more powerful than they initially appear
- For now, we model the data as linearly separable, i.e., there exists a hyperplane perfectly separating the classes

SVM is a linear binary classifier

Predict class A if $s \geq 0$
Predict class B if $s < 0$
where $s = b + \sum_{i=1}^{m} x_i w_i$

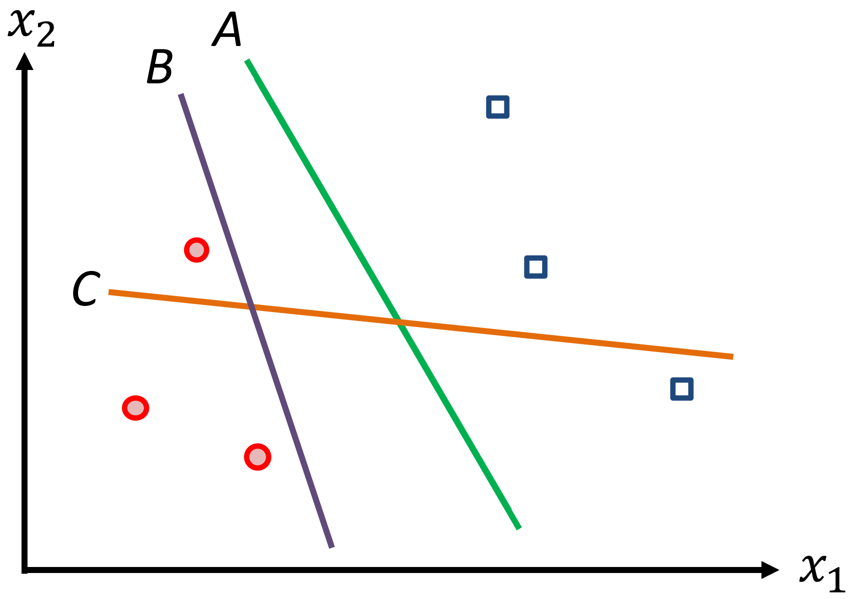
SVM is a linear classifier: s is a linear function of inputs, and the separating boundary is linear



Example for 2D data

Choosing separation boundary

- An SVM is a linear binary classifier: choosing parameters means choosing a separating boundary (hyperplane)
- In 2D:

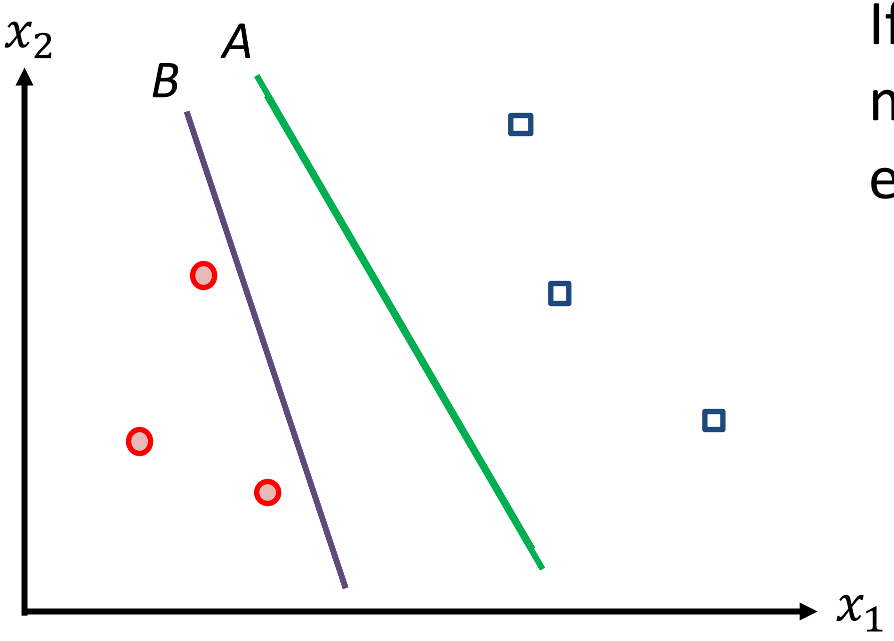


Which boundary would you choose?

A (Green)
B (Purple)
C (Orange)

Which boundary should we use?

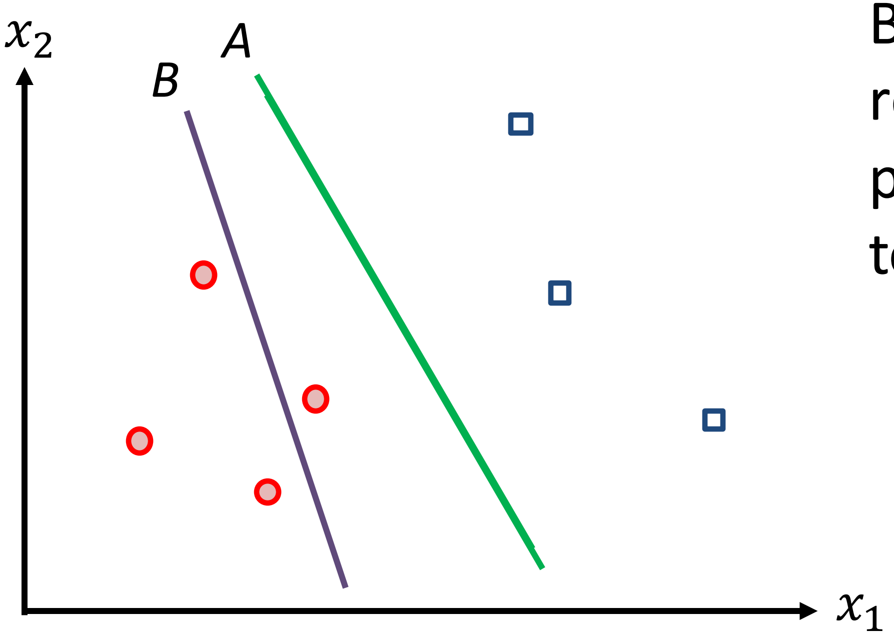
- Provided the dataset is linearly separable, classifiers like logistic regression might find boundaries that separate classes perfectly. Many such boundaries exist (infinite!)



If defining loss as 0-1 mistakes, A and B are equally good.

Which boundary should we use?

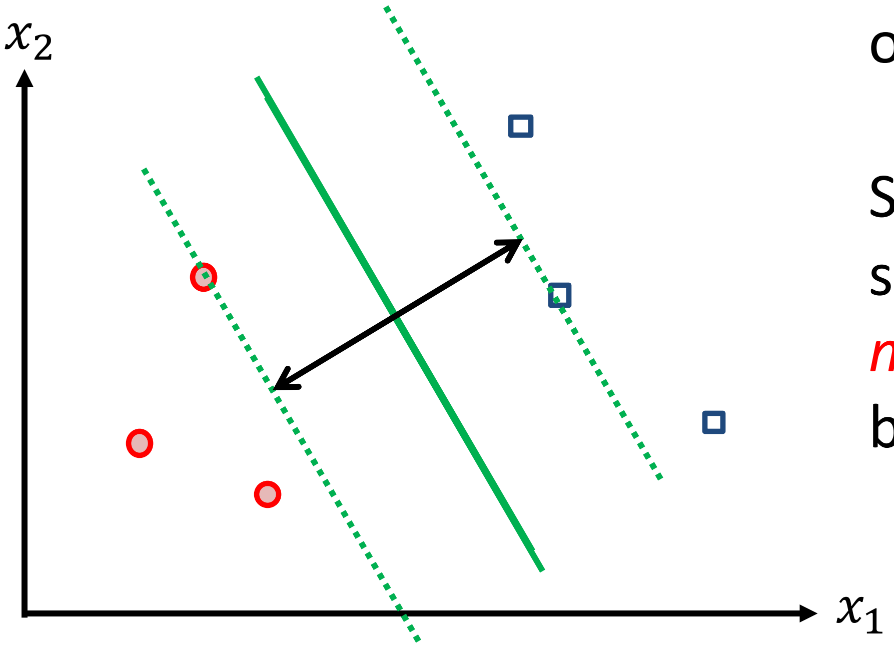
- Provided the dataset is linearly separable, classifiers like logistic regression might find boundaries that separate classes perfectly. Many such boundaries exist (infinite!)



But... line A seems more reliable. When new data point arrives, line B is likely to misclassify it

Aiming for the safest boundary

- Intuitively, the most reliable boundary would be the one that is between the classes and as far away from both classes as possible



SVM objective captures this observation

SVMs aim to find the separation boundary that *maximises the margin* between the classes

Maximum-margin classifier

- An SVM is a linear binary classifier. SVM training aims to find the separating boundary that maximises margin
- For this reason, SVMs a.k.a *maximum-margin classifiers*
- The training data is fixed, so the margin is defined by the location and orientation of the separating boundary
- Our next step is to formalise our objective by expressing *margin width* as a function of parameters (and data)

Mini Summary

- Many linear classifiers seem equally good for linearly separable data if you just care about training error
- Max-margin classifier is far from data and therefore robust against training data sampling effects

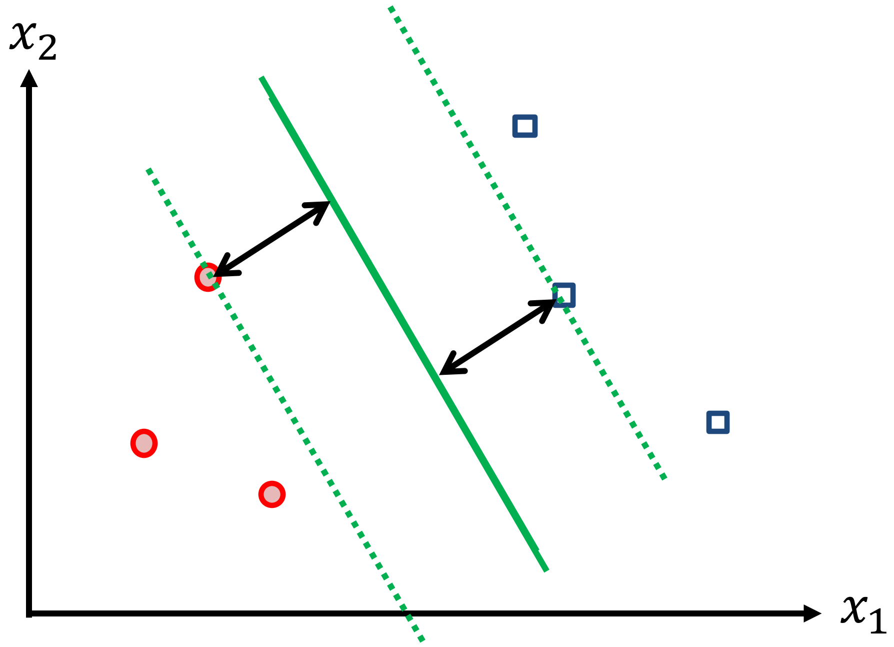
Next: Deriving the max-margin objective

Maximum-Margin Classifier: Derivation

A geometric derivation of
the (hard-margin) SVM's objective

Margin width

- While the margin can be thought as the space between two dashed lines, it is more convenient to define margin width as the distance between the separating boundary and the nearest data point(s)

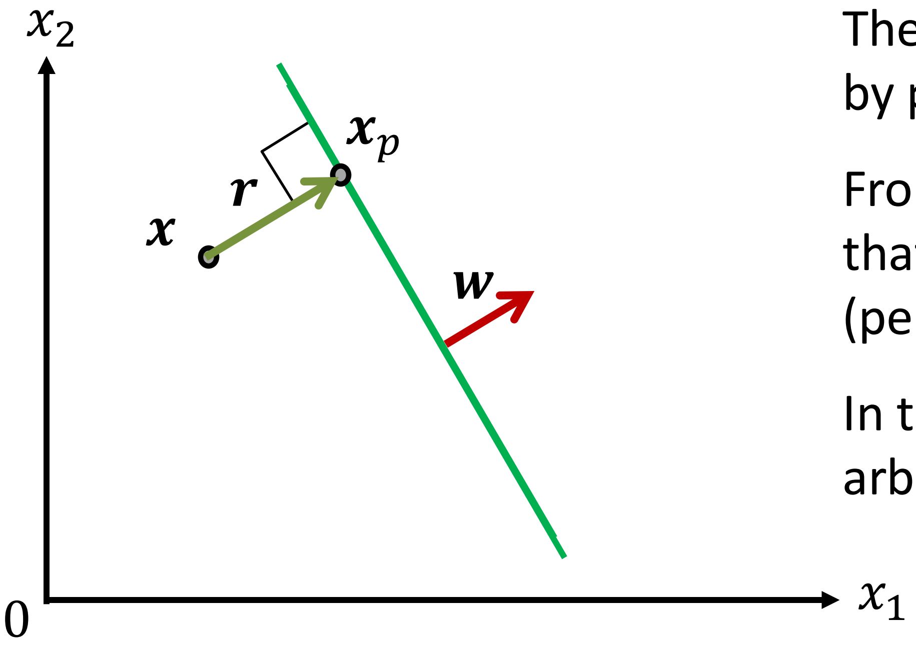


Point(s) on margin boundaries called *support vectors*

We want to maximise the distance to support vectors

Distance from point to hyperplane

- Consider an arbitrary point $\boldsymbol{x}$ (from either of the classes, and not necessarily the closest one to the boundary), and let $\boldsymbol{x}_p$ denote the projection of $\boldsymbol{x}$ onto the separating boundary
- Now, let $\boldsymbol{r}$ be a vector $\boldsymbol{x}_p - \boldsymbol{x}$. Note that $\boldsymbol{r}$ is perpendicular to the boundary, and also that $\|\boldsymbol{r}\|$ is the required distance



The separation boundary is defined by parameters $\boldsymbol{w}$ and b.

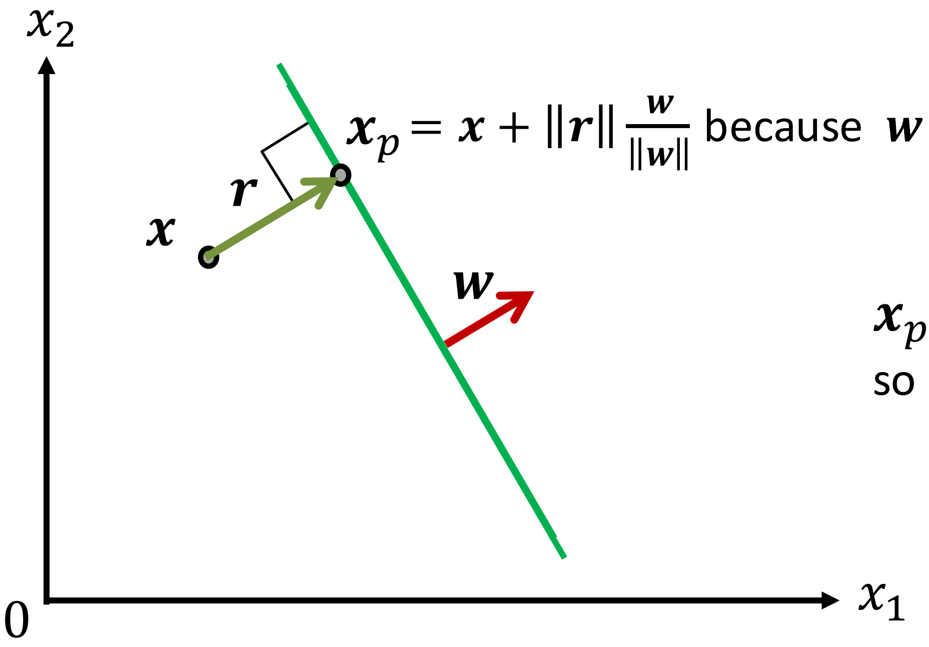
From our linear algebra slides, recall that $\boldsymbol{w}$ is a vector normal (perpendicular) to the boundary

In the figure, $\boldsymbol{w}$ is drawn from an arbitrary starting point on boundary

Remember that $\|\boldsymbol{w}\| = \sqrt{w_1^2 + \cdots + w_m^2}$

Distance from point to hyperplane

- Distance is $\|\boldsymbol{r}\| = -\frac{\boldsymbol{w}'\boldsymbol{x}+b}{\|\boldsymbol{w}\|}$, or more generally $\|\boldsymbol{r}\| = \pm\frac{\boldsymbol{w}'\boldsymbol{x}+b}{\|\boldsymbol{w}\|}$



$\boldsymbol{x}_p = \boldsymbol{x} + \|\boldsymbol{r}\|\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ because $\boldsymbol{w} \parallel \boldsymbol{r}$.

$\boldsymbol{x}_p$ is on the decision boundary, so we know $0 = \boldsymbol{w}'\boldsymbol{x}_p + b$.

Distance from point to hyperplane

- Distance is $\|\boldsymbol{r}\| = -\frac{\boldsymbol{w}'\boldsymbol{x}+b}{\|\boldsymbol{w}\|}$, or more generally $\|\boldsymbol{r}\| = \pm\frac{\boldsymbol{w}'\boldsymbol{x}+b}{\|\boldsymbol{w}\|}$

$$\boldsymbol{x}_p = \boldsymbol{x} + \|\boldsymbol{r}\|\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|},$$

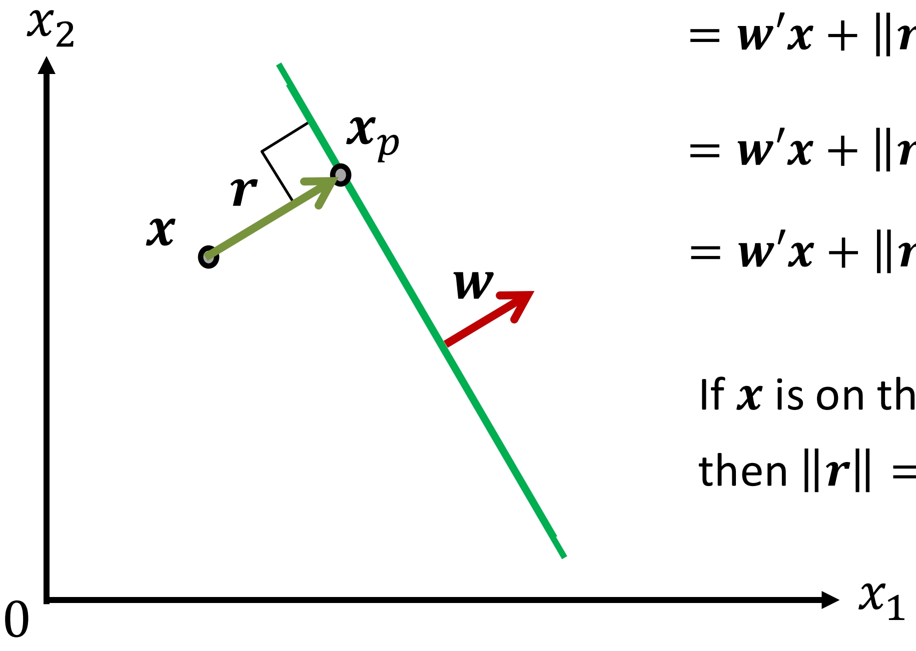
$$\Big\downarrow$$

$$0 = \boldsymbol{w}'\boldsymbol{x}_p + b \Rightarrow 0 = \boldsymbol{w}'\left(\boldsymbol{x} + \|\boldsymbol{r}\|\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) + b$$

$$= \boldsymbol{w}'\boldsymbol{x} + \|\boldsymbol{r}\|\frac{\boldsymbol{w}'\boldsymbol{w}}{\|\boldsymbol{w}\|} + b$$

$$= \boldsymbol{w}'\boldsymbol{x} + \|\boldsymbol{r}\|\frac{\|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} + b$$

$$= \boldsymbol{w}'\boldsymbol{x} + \|\boldsymbol{r}\|\|\boldsymbol{w}\| + b \Rightarrow \|\boldsymbol{r}\| = -\frac{\boldsymbol{w}'\boldsymbol{x} + b}{\|\boldsymbol{w}\|}$$



If $\boldsymbol{x}$ is on the right side of the green line, then $\|\boldsymbol{r}\| = \frac{\boldsymbol{w}'\boldsymbol{x}+b}{\|\boldsymbol{w}\|}$.

Encoding the side using labels

- Training data is a collection $\{\boldsymbol{x}_i, y_i\}$, $i = 1, \dots, n$, where each $\boldsymbol{x}_i$ is an m-dimensional instance and y_i is the corresponding binary label encoded as -1 or 1
- Given a **perfect** separation boundary, y_i will encode the side of the boundary each $\boldsymbol{x}_i$ is on
- Thus the distance from the i-th point to a perfect boundary can be encoded as

$$\|\boldsymbol{r}_i\| = \frac{y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|}$$

Maximum margin objective

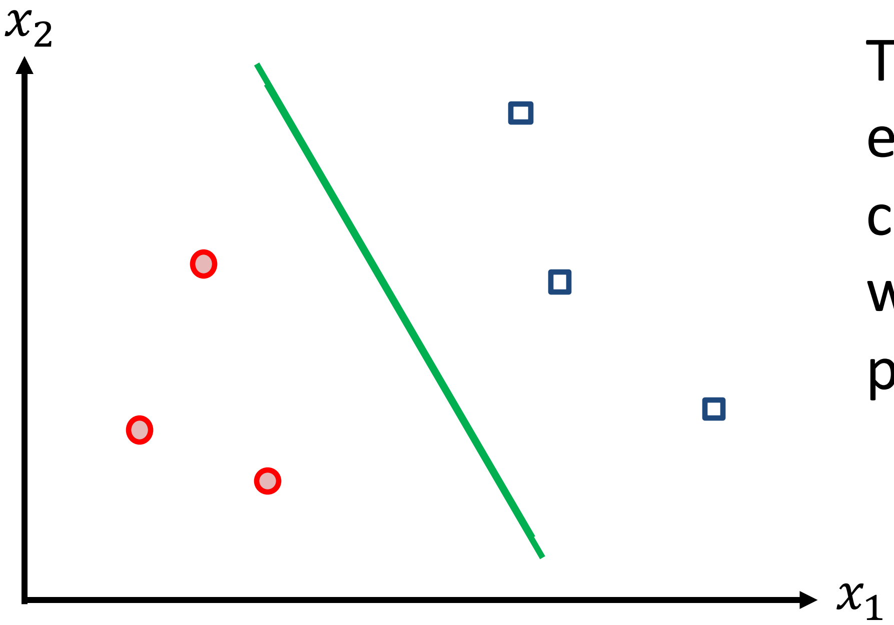
- The distance from the i-th point to a perfect boundary can be encoded as $\|\boldsymbol{r}_i\| = \frac{y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|}$
- The margin width is the distance to the closest point
- Thus SVMs aim to maximise $\left(\min_{i=1,\ldots,n} \frac{y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|}\right)$ as a function of $\boldsymbol{w}$ and b

But the objective function is non-unique and non-convex

art: OpenClipartVectors at pixabay.com (CC0)

Non-unique representation

- A separating boundary (e.g., a line in 2D) is a set of points that satisfy $\boldsymbol{w}'\boldsymbol{x} + b = 0$ for some given $\boldsymbol{w}$ and b
- However, the same set of points will also satisfy $\tilde{\boldsymbol{w}}'\boldsymbol{x} + \tilde{b} = 0$, with $\tilde{\boldsymbol{w}} = \alpha\boldsymbol{w}$ and $\tilde{b} = \alpha b$, where $\alpha > 0$ is arbitrary



The same boundary, and essentially the same classifier can be expressed with infinitely many parameter combinations

Constraining the objective for uniqueness

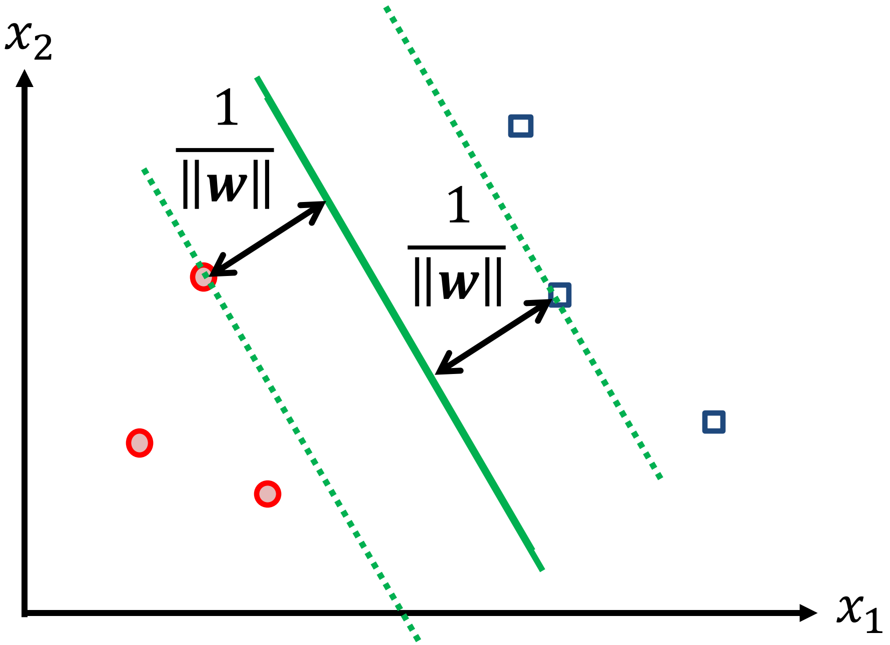
- SVMs aim to maximise $\left(\min_{i=1,\dots,n} \frac{y_i(\boldsymbol{w}'\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|}\right)$
- Introduce (arbitrary) extra requirement $\min_{i=1,\dots,n} \frac{y_i(\boldsymbol{w}'\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$
- Instead of maximising margin $\frac{1}{\|\boldsymbol{w}\|}$, can minimise $\|\boldsymbol{w}\|$
- Ensure classifier makes no errors: constrain $y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1$
  - $\min_{i=1,\dots,n} \frac{y_i(\boldsymbol{w}'\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$ equivalent to $\frac{y_i(\boldsymbol{w}'\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|} \geq \frac{1}{\|\boldsymbol{w}\|}$ for all i
  - $\frac{y_i(\boldsymbol{w}'\boldsymbol{x}_i+b)}{\|\boldsymbol{w}\|} \geq \frac{1}{\|\boldsymbol{w}\|}$ equivalent to $y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1$
  - Data points where $y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) = 1$ are called support vectors

Hard-margin SVM objective

We now have a major result: SVMs aim to find

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t. } y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 \text{ for } i = 1, \dots, n$$


x_2
x_1
$\frac{1}{\|\boldsymbol{w}\|}$
$\frac{1}{\|\boldsymbol{w}\|}$

Note 1: parameter b is optimised indirectly by influencing constraints

Note 2: all points are enforced to be on or outside the margin

Therefore, this version of SVM is called *hard-margin SVM*

*Changed $\|\boldsymbol{w}\|$ to $\frac{1}{2}\|\boldsymbol{w}\|^2$ - monotonic increasing transform. Modified objective yields same solution.

Mini Summary

- Derived expression for margin, towards formulating an objective to optimise for training an SVM
- Chose a “canonical scale” to ensure uniqueness
- Converted max margin to min norm of ***w***
- Constraints needed to ensure perfect accuracy

Next: SVM objective as regularised loss

SVM Objective as Regularised Loss

Relating the resulting objective function to that of other machine learning methods

Previously in COMP90051 …

1. Choose/design a model
2. Choose/design loss function
3. Find parameter values that minimise discrepancy on training data

How do SVMs fit this pattern?

SVM as Regularised ERM

- Recall ridge regression objective

$$\text{minimise } \left(\sum_{i=1}^{n}(y_i - \boldsymbol{w}'\boldsymbol{x}_i)^2 + \lambda\|\boldsymbol{w}\|^2\right)$$

- Hard margin SVM objective

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t. } y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 \text{ for } i = 1, \ldots, n$$

data-dependent training error

data-independent regularisation term

- The constraints can be interpreted as loss

$$l_\infty = \begin{cases} 0 & 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \leq 0 \\ \infty & 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) > 0 \end{cases}$$

Hard margin SVM loss

- The constraints can be interpreted as loss

$$l_\infty = \begin{cases} 0 & 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \leq 0 \\ \infty & 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) > 0 \end{cases}$$

- In other words, for each point:
  - If it’s on the right side of the boundary and at least $\frac{1}{\|\boldsymbol{w}\|}$ units away from the boundary, we’re OK, the loss is 0
  - If the point is on the wrong side, or too close to the boundary, we immediately give infinite loss thus prohibiting such a solution altogether

Mini Summary

- Very helpful to view hard-margin SVM as minimising regularised loss
  - Connects this topic to rest of COMP90051
  - Prepares us for more important soft-margin SVM
- Regularisation: Objective function $\boldsymbol{w}$ norm
- Loss: Found in the constraints

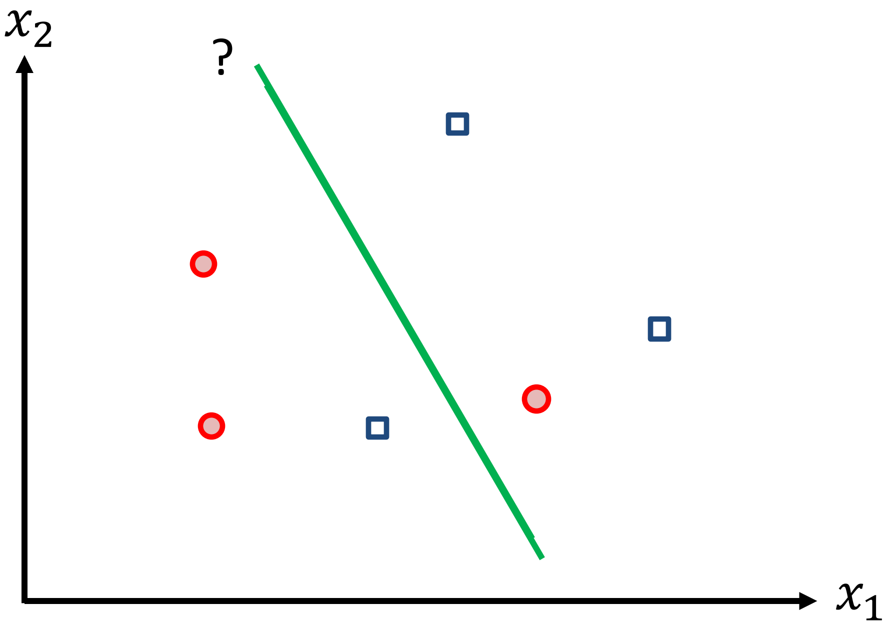
Next: Soft-margin SVM

Soft-Margin SVMs

Addressing linear inseparability

When data is not linearly separable

- Hard-margin loss is too stringent (*hard*!)
- Real data is unlikely to be linearly separable
- If the data is not separable, hard-margin SVMs are in trouble

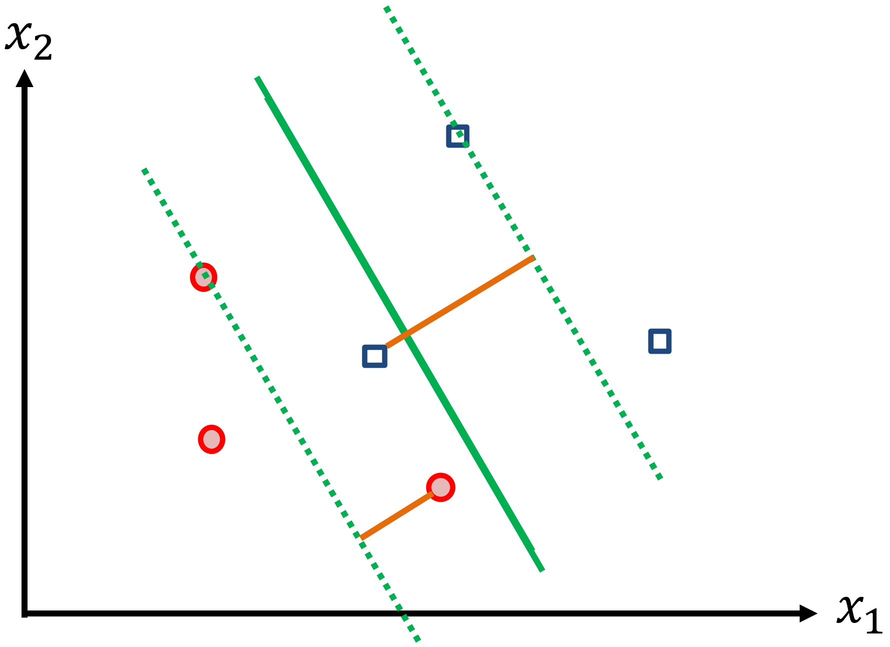


SVMs offer 3 approaches to address this problem:

1. *Still use hard-margin SVM, but transform the data (next lecture)*
2. *Relax the constraints (next slide)*
3. *The combination of 1 and 2 ☺*

Soft-margin SVM

- Relax constraints to allow points to be inside the margin or even on the wrong side of the boundary



However, we penalise boundaries by the extent of “violation”

In the figure, the objective penalty will take into account the orange distances

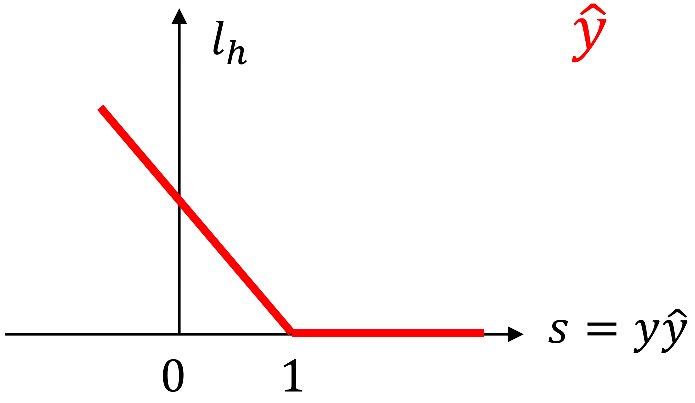
Hinge loss: soft-margin SVM loss

- Hard-margin SVM loss

$$l_\infty = \begin{cases} 0 & 1 - y(\boldsymbol{w}'\boldsymbol{x} + b) \leq 0 \\ \infty & otherwise \end{cases}$$

- Soft-margin SVM loss (hinge loss)

$$l_h = \begin{cases} 0 & 1 - y(\boldsymbol{w}'\boldsymbol{x} + b) \leq 0 \\ 1 - \underbrace{y(\boldsymbol{w}'\boldsymbol{x} + b)}_{\hat{y}} & otherwise \end{cases}$$



Soft-margin SVM objective

- Soft-margin SVM objective

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}}\left(\sum_{i=1}^{n} l_h(\boldsymbol{x}_i, y_i, \boldsymbol{w}, b) + \lambda\|\boldsymbol{w}\|^2\right)$$

  - ∗ Reminiscent of ridge regression
  - ∗ Hinge loss $l_h = \max\left(0,1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)\right)$

- We are going to re-formulate this objective to make it more amenable to analysis

Re-formulating soft-margin objective

- Introduce slack variables as an upper bound on loss

$$\xi_i \geq l_h = \max\left(0, 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)\right)$$

or equivalently $\xi_i \geq 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)$ and $\xi_i \geq 0$

- Re-write the soft-margin SVM objective as:

$$\underset{\boldsymbol{w}, b, \boldsymbol{\xi}}{\operatorname{argmin}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i\right)$$

s.t. $\xi_i \geq 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)$ for $i = 1, \dots, n$

$\xi_i \geq 0$ for $i = 1, \dots, n$

Side-by-side: Two variations of SVM

- Hard-margin SVM objective*:

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t. } y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 \text{ for } i = 1, \dots, n$$

- Soft-margin SVM objective:

$$\underset{\boldsymbol{w},b,\boldsymbol{\xi}}{\operatorname{argmin}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i\right)$$

$$\text{s.t. } y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1, \dots, n$$

$$\xi_i \geq 0 \text{ for } i = 1, \dots, n$$

- In the second case, the constraints are relaxed ("softened") by allowing violations by ξ_i. Hence the name "soft margin"

Mini Summary

- Support vector machines (SVMs) as maximum margin classifiers
- Deriving hard margin SVM objective
- SVM as regularised ERM
- Soft-margin SVM

Next time: Kernel methods