Lecture 13. Introduction to Support Vector Machines

COMP90051 Statistical Machine Learning

Semester 2, 2015
Lecturer: Andrey Kan

THE UNIVERSITY OF MELBOURNE

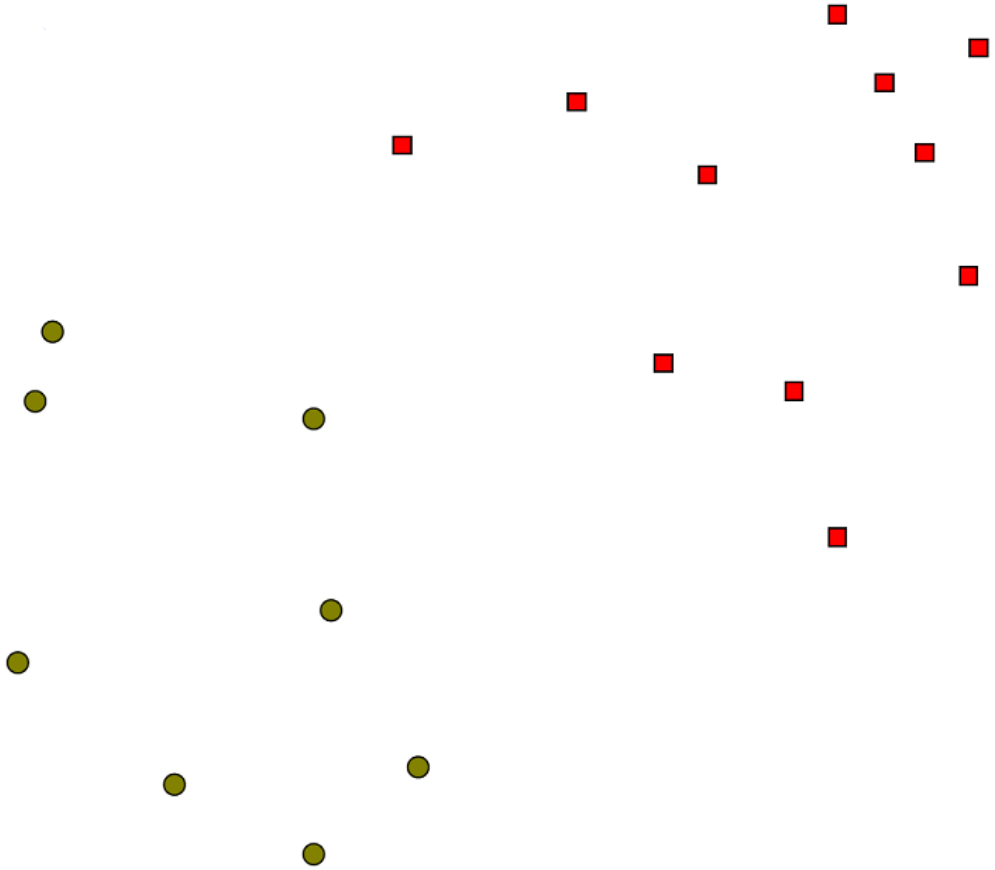
Content is based on slides provided by Ben Rubinstein

Linear Support Vector Machines (SVM)

Yet another linear model that can be regularised

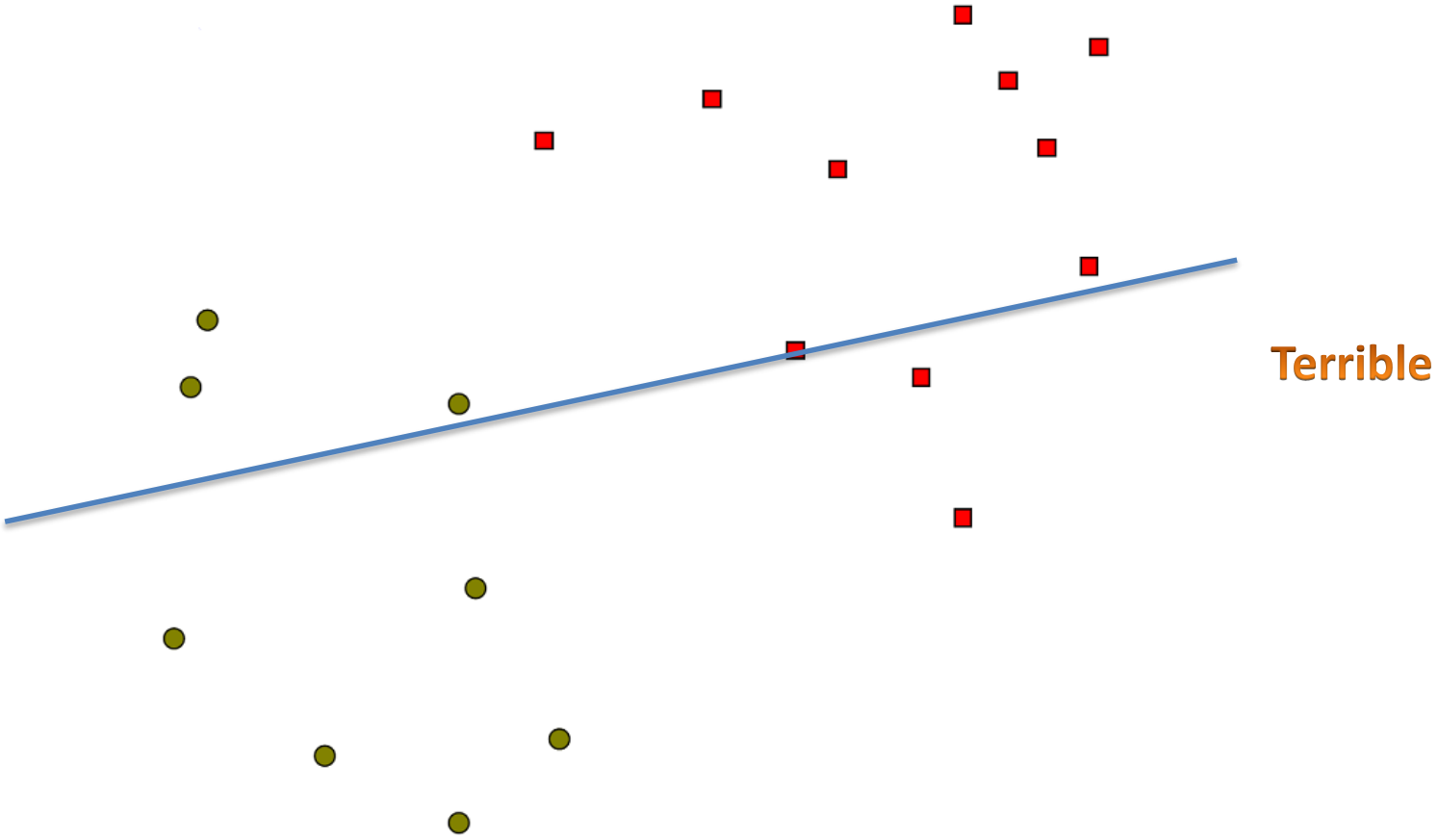
Support Vector Machines: Intuition

- What linear classifier to choose?



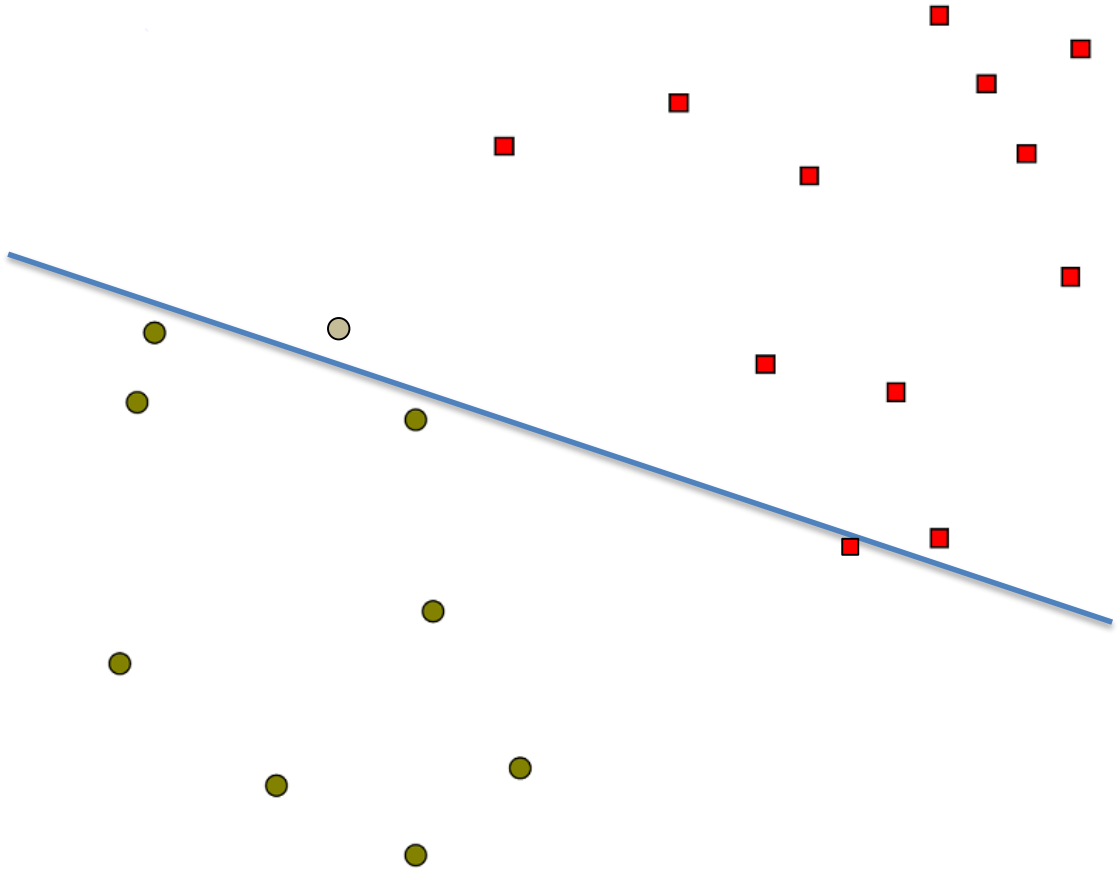
Support Vector Machines: Intuition

- What linear classifier to choose?



Support Vector Machines: Intuition

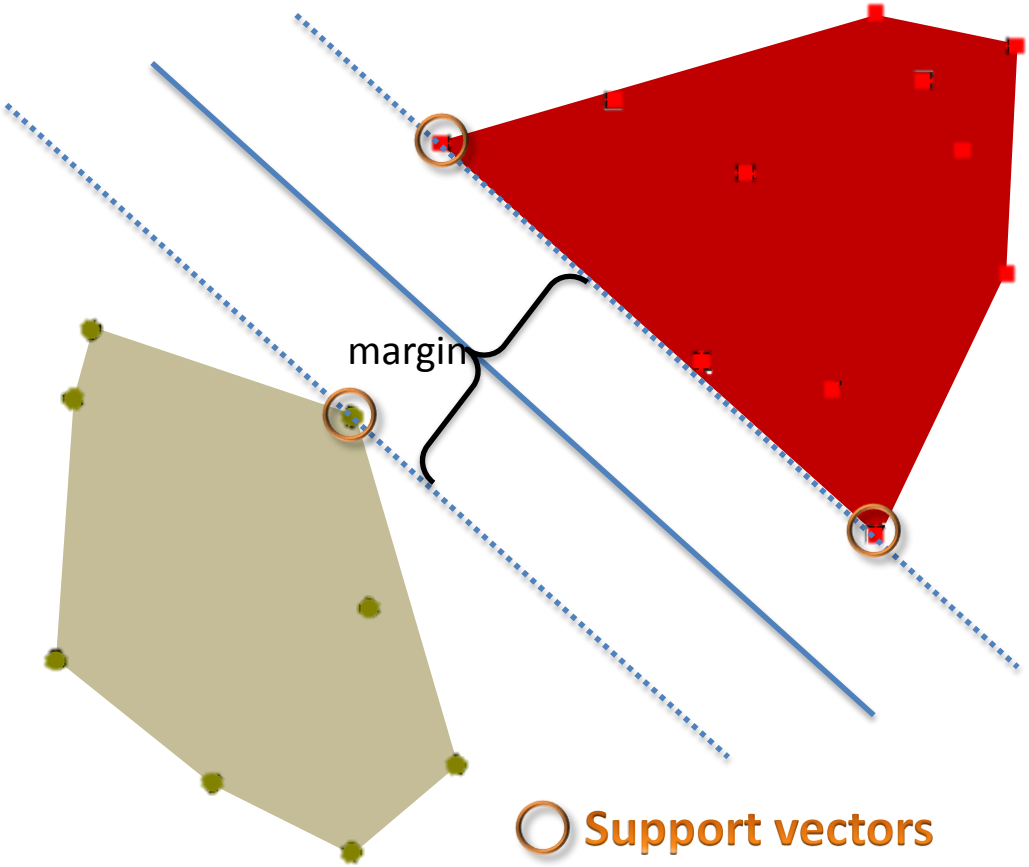
- What linear classifier to choose?



Good train error
Bad test error

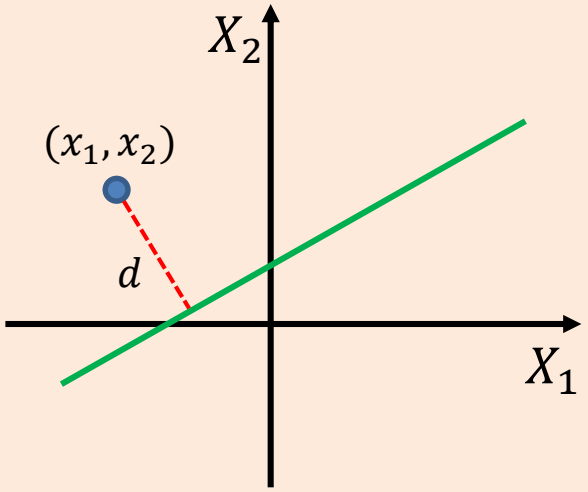
Support Vector Machines: Intuition

- What linear classifier to choose?



(very brief) Revision of geometry

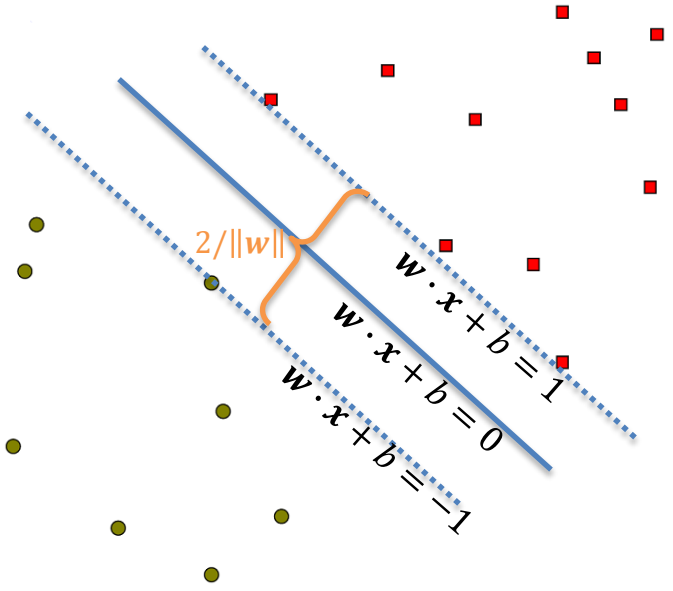
- Line = set of points (x_1, x_2) that satisfy $w_1 x_1 + w_2 x_2 + b = 0$
- Plane ➔ $w_1 x_1 + w_2 x_2 + w_3 x_3 + b = 0$
- Hyperplane ➔ $\sum_{i=1}^{n} w_i x_i + b = 0$



- Note: we can multiply w_i, b by a constant without changing the (hyper-)plane
- Distance from point $\boldsymbol{x}$ to (hyper-)plane is $d = \frac{|\boldsymbol{w}^T \boldsymbol{x} + b|}{\|\boldsymbol{w}\|}$
- We can rescale so that $\boldsymbol{w}^T \boldsymbol{x} + b = 1$, and get $d = \frac{1}{\|\boldsymbol{w}\|}$

Linear SVM parameters

- Goal: max margin hyperplane
- Hyperplane parameters
  - Normal vector $\boldsymbol{w}$
  - Offset b
- Scale parameters so that
  - Negative data: $\boldsymbol{w} \cdot \boldsymbol{x} + b \leq -1$
  - Positive data: $\boldsymbol{w} \cdot \boldsymbol{x} + b \geq +1$
- Then margin is $2/\|\boldsymbol{w}\|$



Linear SVM algorithm

- If linearly separable data: the optimal classifier is

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|^2$$

subject to $y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1, \ \forall i$

Hard-margin SVM

- In general, relax to allow training errors
  - Intuitively minimise: $\frac{1}{2}\|\boldsymbol{w}\|^2 + C \cdot average\big(loss(y_i, \hat{y}_i)\big)$
  - Formally, introduce "slacks" $\xi_i \geq 0$'s

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

subject to $y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i, \ \forall i$

Soft-margin SVM

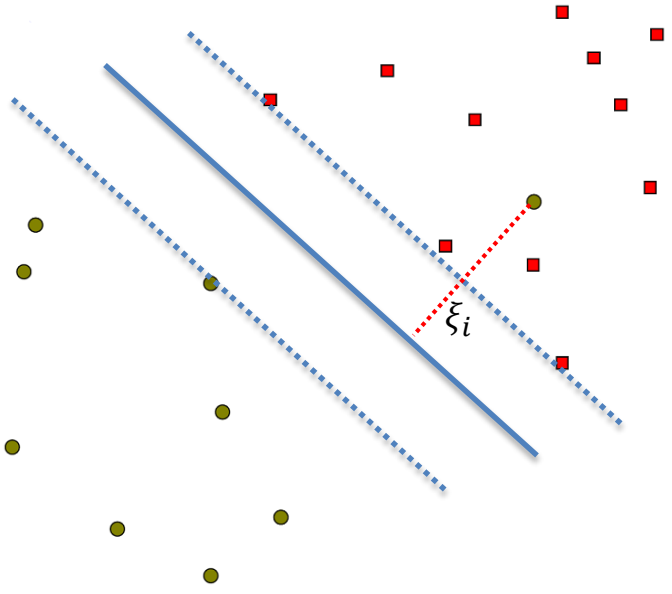
Soft-margin SVM

"Slack" variables: it is OK for some points to be on the wrong side

Still, closer they are to the margin the better

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

subject to $y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i, \ \forall i$



Déjà vu?

Soft-margin SVM: $\min_{\mathbf{w}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C \cdot average\big(loss(y_i, \hat{y}_i)\big) \right)$

model class complexity

training error

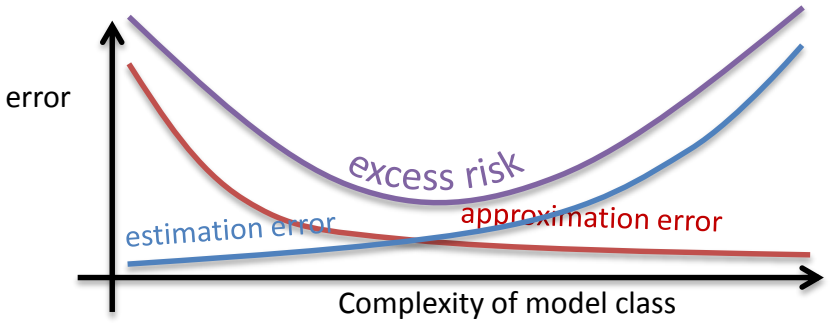
Ridge regression: $\min_{\boldsymbol{w}} \left(\lambda\|\boldsymbol{w}\|^2 + \sum_i (y_i - \boldsymbol{X}_i.\boldsymbol{w})^2 \right)$

model class complexity

training error

Soft-margin parameter *C*

- Varies effective complexity
- Theory: should grow like $O(\sqrt{n})$
- Practice: set with cross-validation



Training the SVM

- Loss function
  - Must be convex to make minimisation tractable
  - Typically: linear (hinge), squared
- Solving the minimisation
  - Convex → unique global optimum guaranteed
  - Could use gradient descent, but there are better ways
  - Constrained Quadratic Program → standard
  - Many tricks to improve speed or memory (chunking, SMO)
- Non-linear SVM coming up (“kernelisation”)

SVM summary (so far)

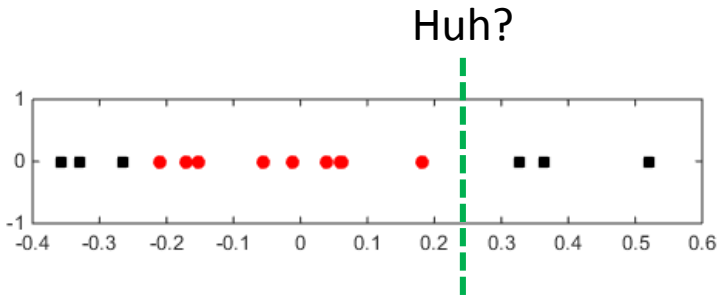
Non-parametric	There are essentially parameters per support vector (next time)
Discriminative	SVM models P(Y|X) – the decision boundary – nothing about P(X)
Regularised, frequentist	Extent of regularisation controlled by varying the soft-margin parameter; Justification is via theoretical bounds on risk
Computational efficiency	Mediocre cubic polynomial in #data, or #features, for training
Accuracy (aka statistical efficiency)	Typically very good empirically; Exists nice theory on error bounds; Much better if non-linear in some cases.
Packages	http://svmlight.joachims.org http://www.csie.ntu.edu.tw/~cjlin/libsvm Pyhton, R packages, MatLab, Weka (Java), etc.

The Kernel Trick

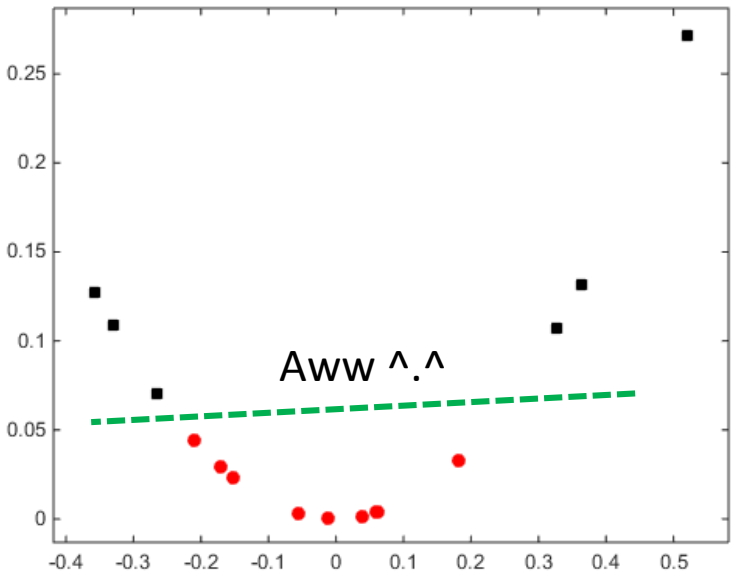
Very powerful method that enables effective use of linear methods for non-linear data

Feature space mapping: example 1

- Consider a binary classification problem
- Each example has one feature $[x_1]$
- Not linearly separable

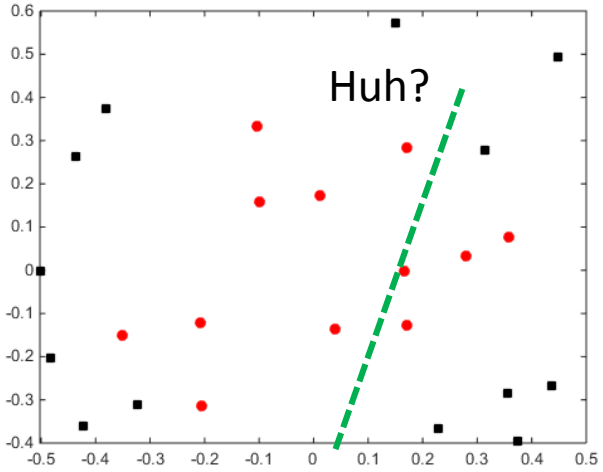


- Now 'add' a feature $x_2 = x_1^2$
- Each point is now $[x_1, x_1^2]$
- Linearly separable!

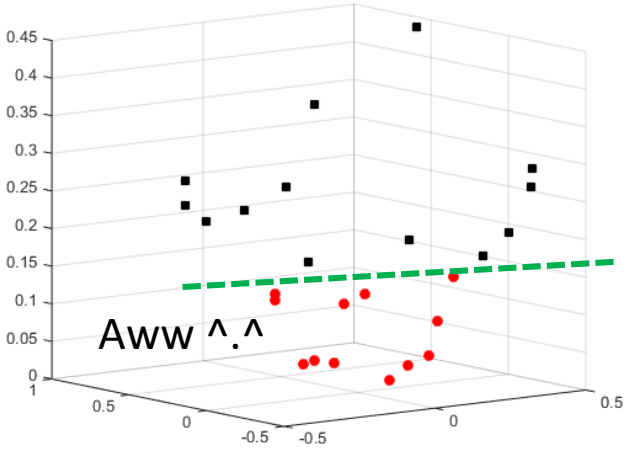


Feature space mapping: example 2

- Consider a binary classification problem
- Each example has features $[x_1, x_2]$
- Not linearly separable



- Now 'add' a feature $x_3 = x^2 + x_2^2$
- Each point is now $[x_1, x_2, x_1^2 + x_2^2]$
- Linearly separable!



Observation 1

- A non-linear boundary in original feature space can become a linear boundary in a transformed space
- Denote this transformation as $\varphi(\boldsymbol{x})$
- For example, $\varphi([x_1, x_2]) = [x_1, x_2, x_1^2 + x_2^2]$
- Note $\varphi(\boldsymbol{x})$ has more dimensions than $\boldsymbol{x}$
  - Commonly used $\varphi(\boldsymbol{x})$ often have very large dimensionality
  - Sometimes $\varphi(\boldsymbol{x})$ have infinite dimensionality!

Naïve workflow

- Choose/design a linear model
- Choose/design a transformation $\varphi(\boldsymbol{x})$
- For each training example, and for each new instance compute $\varphi(\boldsymbol{x})$
- Train classifier/Do predictions

- Problem: impractical/impossible to compute $\varphi(\boldsymbol{x})$ for high/infinite-dimensional $\varphi(\boldsymbol{x})$

Observation 2

- For a large class of linear models, both parameter estimation and computing predictions depend on data only in a form of dot product
- Dot product $\boldsymbol{xy} = \sum_{i=1}^{d} x_i y_i$
- In transformed space $\varphi(\boldsymbol{x})\varphi(\boldsymbol{y}) = \sum_{i=1}^{d'} \varphi(\boldsymbol{x})_i \varphi(\boldsymbol{y})_i$
- Kernel is a function that can be expressed as a dot product in some feature space $K(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x})\varphi(\boldsymbol{y})$

Kernel trick: example

- For many transformations there is a shortcut computation of the dot product!
- For example, consider two points in original space $\boldsymbol{x} = [x_1]$ and $\boldsymbol{y} = [y_1]$
- Consider transformation $\varphi(\boldsymbol{x}) = [x_1^2, \sqrt{2c}x_1, c]$
- Then $\varphi(\boldsymbol{x})\varphi(\boldsymbol{y}) = (x_1^2 y_1^2 + 2cx_1 y_1 + c^2)$
- This simplifies to $\varphi(\boldsymbol{x})\varphi(\boldsymbol{y}) = (x_1 y_1 + c)^2$
- Here $K(\boldsymbol{x}, \boldsymbol{y}) = (x_1 y_1 + c)^2$ is called kernel

Kernel trick

- For a pair of data points in original space kernel $K(\boldsymbol{x}, \boldsymbol{y})$ can be used to compute dot product in the transformed space
- Computing the transformation $\varphi(\boldsymbol{x})$ as such is not necessary!
- Kernel can be computed in $O(d)$, whereas computing $\varphi(\boldsymbol{x})$ requires $O(d')$, where $d' \gg d$ or $d' = \infty$

Summary

- Linear SVM aims to maximise a separating margin
- Soft-margin SVM allows some instances to be misclassified
- Soft-margin SVM includes regularisation parameter

- When data is not linearly separable, transforming the feature space can make it linearly separable
- Many linear methods only require data in form of dot products
- Kernel trick – fast computation of dot products in very high dimensional feature spaces