Classification Methods part 2

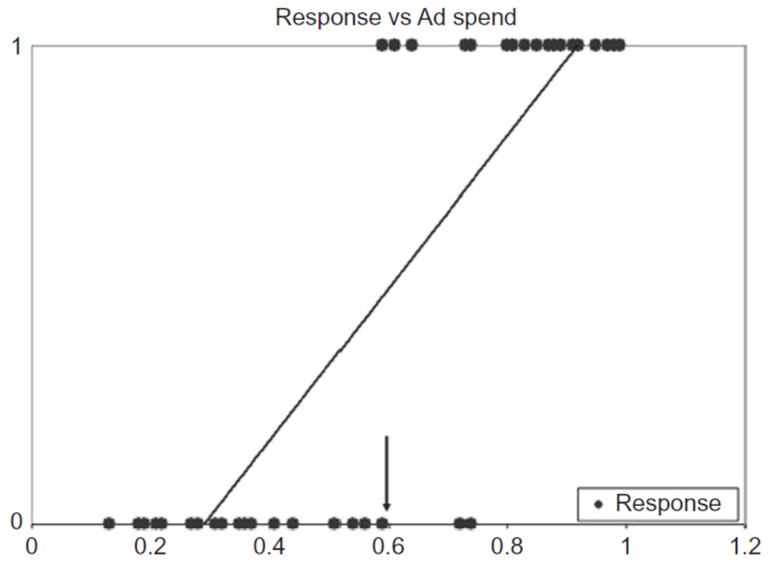
Module #3

THE UNIVERSITY of ADELAIDE

Lecture Summary

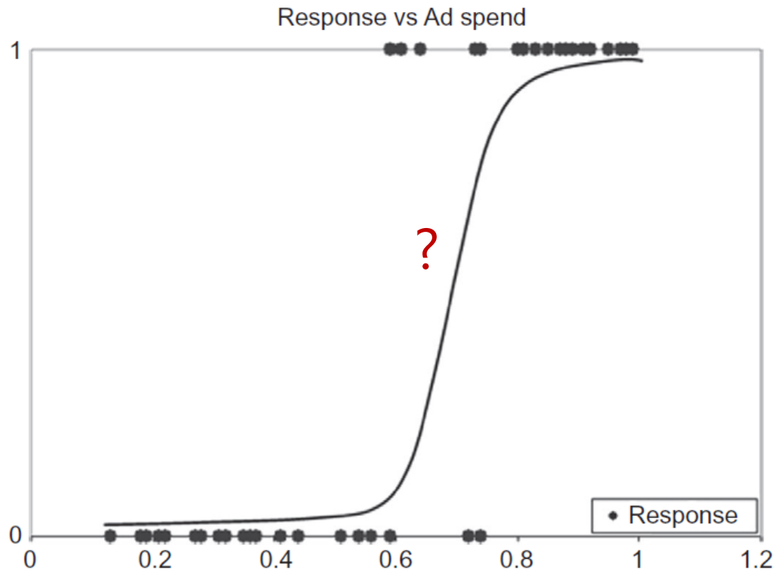
- Classification with Logistic Regression and Neural Networks
- Bayes Theory Naïve Bayes Recap
- Laplace smoothing
- SVM

Recap: Logistic Regression



Linear Fit for a Binary outcome: Although we can make an intuitive assessment that increase in *Ad spend increases Response*, the switch is abrupt - around 0.6. Using the straight line, we cannot really predict outcome.

FIGURE 5.17
Fitting a linear model to discrete data.



Logistic Regression Model. The S-shaped curve is clearly a better fit for *most* of the data. we can state *Ad spend* increases *Sales*, **and**. we may also be able to predict using this model.

FIGURE 5.18
Fitting a nonlinear curve to discrete data.

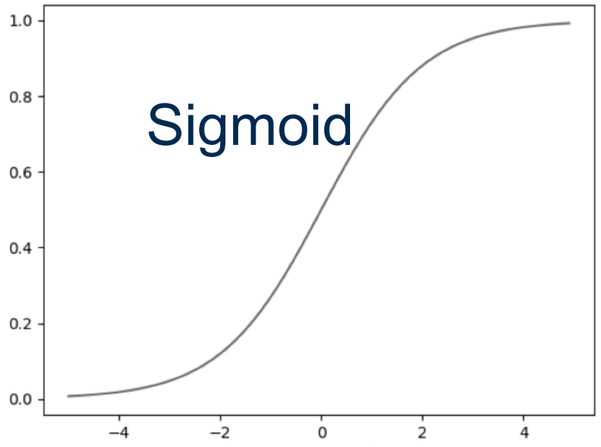
$$\text{logit} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \qquad p = e^{\text{logit}} / (1 + e^{\text{logit}})$$

How to classify if the test point is close to middle?

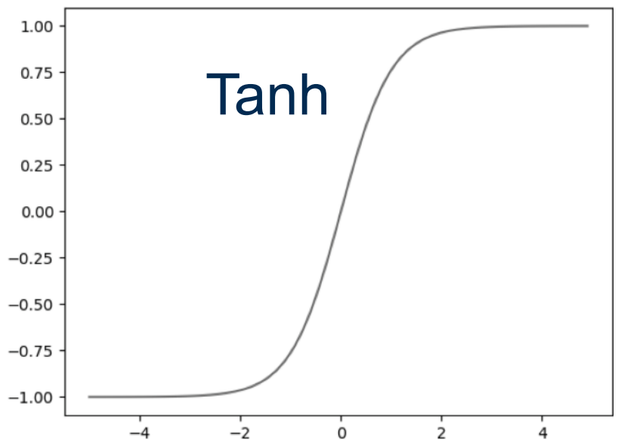
Rebala et al. ch. 5.2

THE UNIVERSITY of ADELAIDE

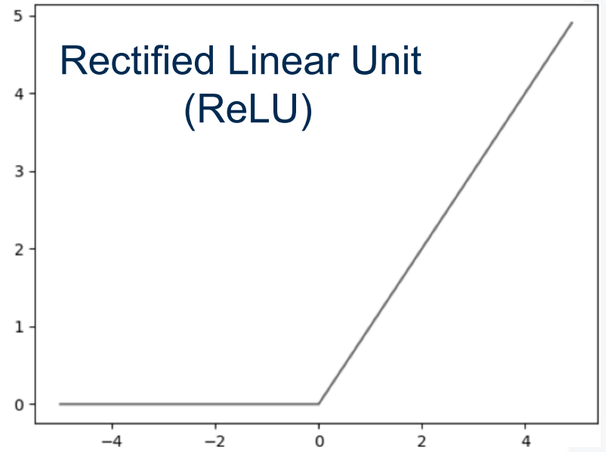
Logistic Regression and Neural Networks



$$f(x) = \frac{1}{1 + e^{-z}}$$



$$f(x) = \frac{e^{2z} - 1}{e^{2z} + 1}$$



$$f(x) = max\{0, z\}$$

- Let $z = a + xb$, a=-1.0, b=1.0. input x=1.05, If we use Sigmoid, what would be the $f(x)$ value?

THE UNIVERSITY of ADELAIDE

Classification with Neural Networks

- Remember what NN does? (forward and backward pass, sigmoid activation)
- **Binary:** NN classifies binary targets in very similar way to logistic regression.
- **k-class classification**:
  - Neural Network with k outputs: choose class with highest output score: max arg(output_score)
- **k-label multi-label categorisation:**
  - Neural Network with k outputs, set threshold θ (same or different for each k),
    - For all k
      - If k_output > θ, set k class to 1, else 0
  - Many other methods available
    (see http://jmread.github.io/talks/Tutorial-MLC-Porto.pdf)

THE UNIVERSITY of ADELAIDE

Naïve Bayes for Classification

THE UNIVERSITY of ADELAIDE

Questions

- Why Naïve Bayes classifier is called “naïve”
- What to do if a feature appears in test set that never existed in training set?

Laplace Smoothing for classification

If a test case has an attribute that does not occur in every class? Then the probability value for that attribute is 0, so all is 0!

In this case use Laplace smoothing:

$$P(x|y) = \frac{n_{yx}}{n_y} \Longrightarrow \hat{P}(x|y) = \frac{\gamma + n_{yx}}{\gamma \cdot d + n_y}, \; x \in (x_1, \ldots, x_d)$$ d-dimensional multinomial distribution

n_y no. of instances from training in class y

n_{yx} no. of instances from class y with value x for attribute X

d no. of distinct features for that attribute (number of dimensions),

γ is usually taken as 1.

Every feature must be represented in each class with non-zero probability.

Using Naïve Bayes with Laplace Smoothing

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

We have a new instance: $\langle Outlk$ = *ovc*, $Temp$ = $cool$, $Humid$ = $high$, $Wind$ = $true\rangle$, *Play or not*?

Let's recalculate with smoothing by adding 10 (#features)

P(play=yes) = 9/14, P(play=no) = 5/14 (same as before)

P(ovc | y=(4+1)/(9+10), P(cool | y)=(3+1)/(9+10),
P(high | y)=(3+1)/(9+10),
P(windy=true | y) =(3+1)/(9+10)

P(ovc | n=(0+1)/(5+10), P(cool | n)=(1+1)/(5+10),
P(high | n)=(4+1)/(5+10),
P(windy=true | n)=(3+1)/(5+10)

Therefore

arg max(9/14*5/19*4/19*4/19*4/19,
5/14*1/15*2/15*5/15*4/15) =

arg max(y: 0.0016,n:0,00028) = yes (same result)

THE UNIVERSITY of ADELAIDE

Support Vector Machines

THE UNIVERSITY of ADELAIDE

Finding the optimal hyperplane

- The primal optimization problem for linear support vector machine

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i$$

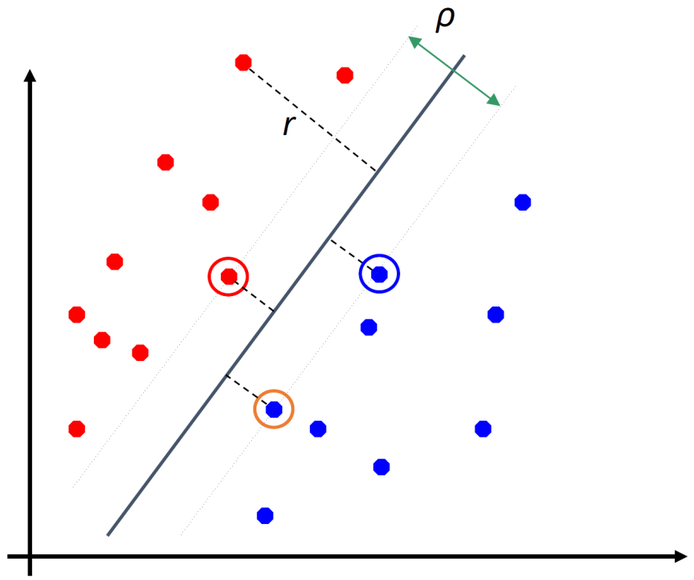
- Decision rule

$$\hat{y} = 1 \quad if \quad \mathbf{w}^\top \mathbf{x} + b > 0$$

$$\hat{y} = -1 \quad if \quad \mathbf{w}^\top \mathbf{x} + b < 0$$

- Margin

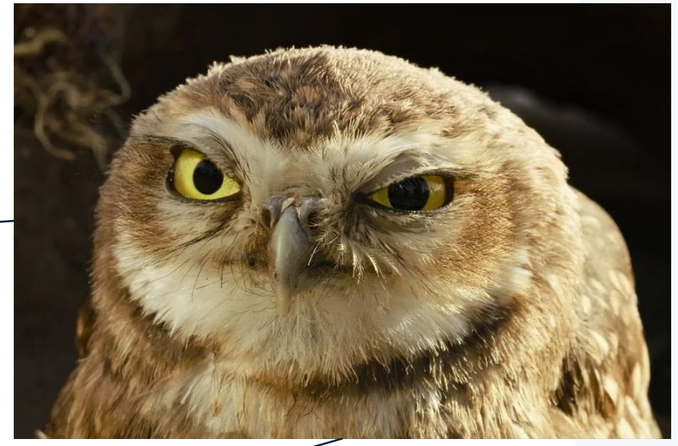
$$\rho = \frac{1}{\|\mathbf{w}\|_2}$$

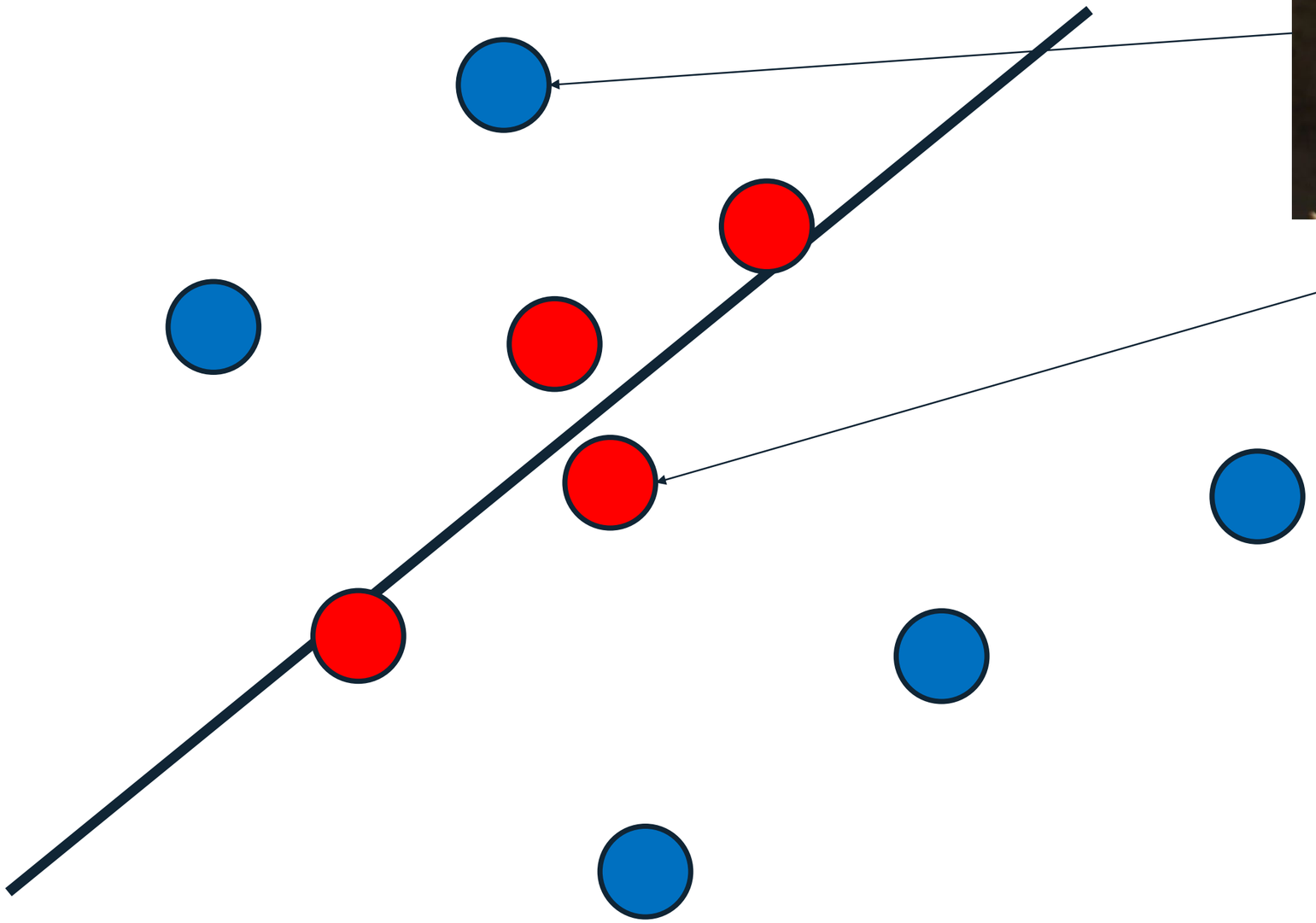


- All the support vectors are in

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$$

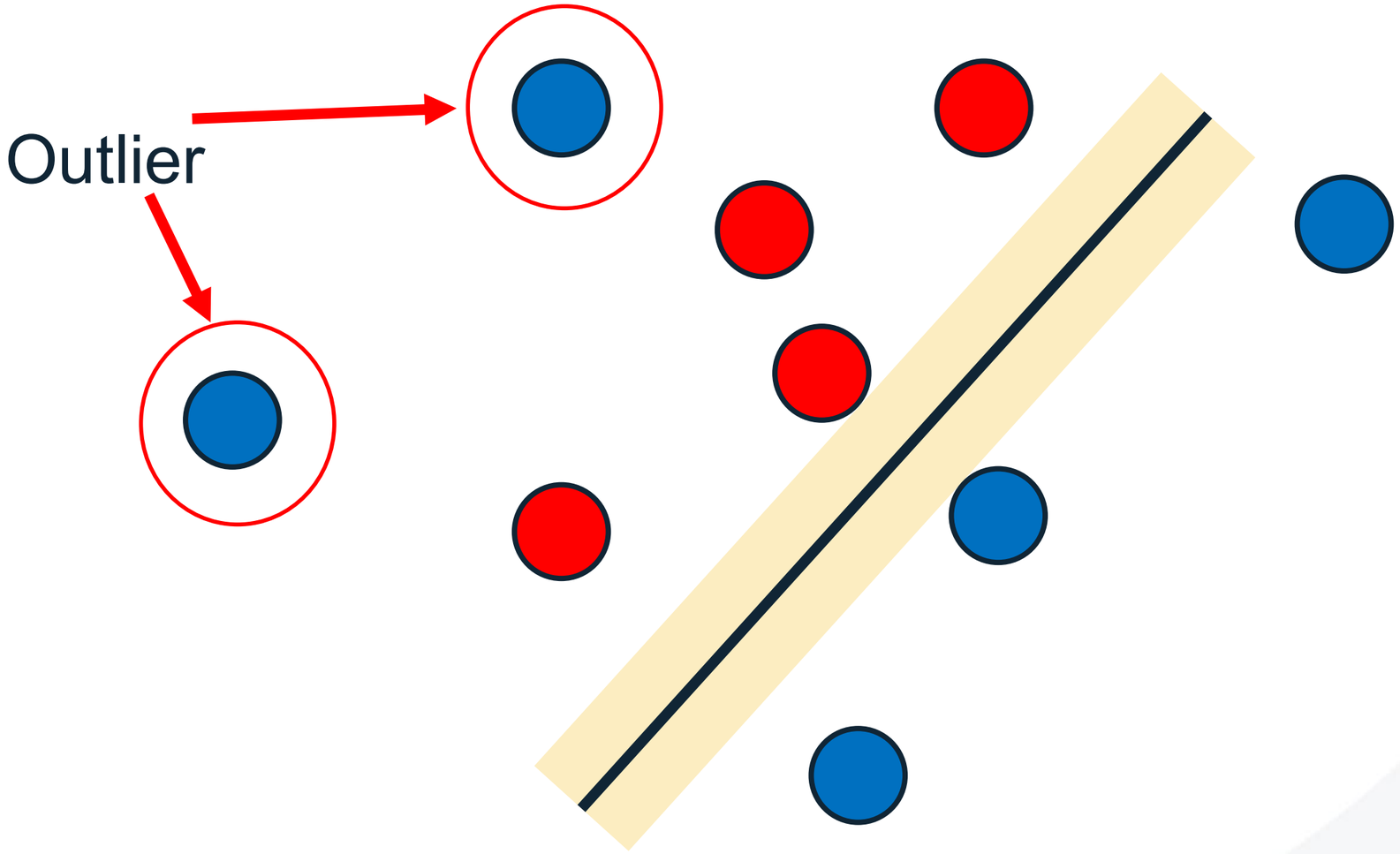
What if we have this?





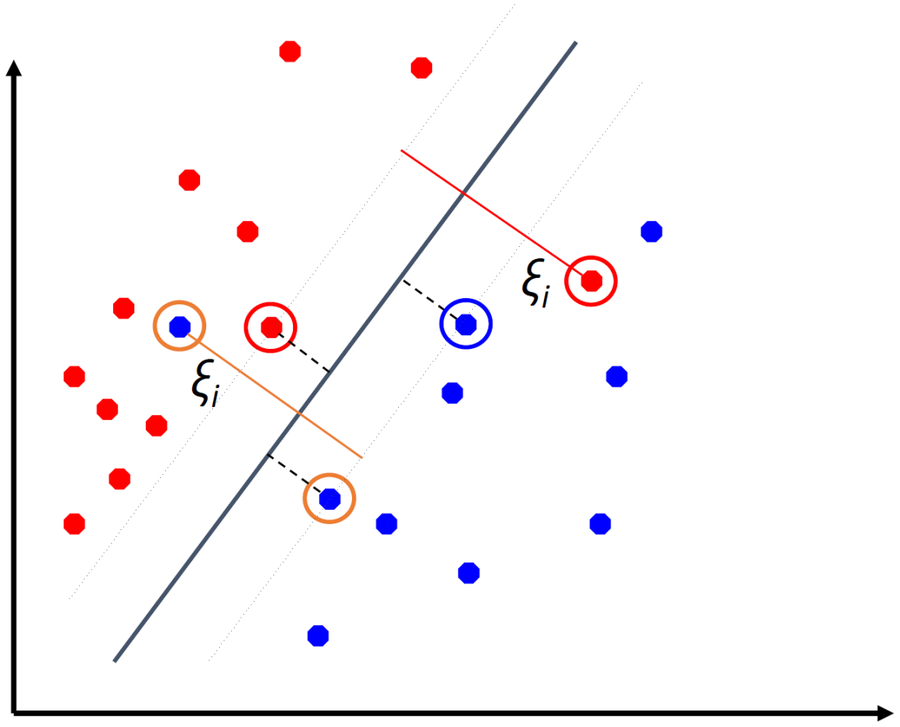
THE UNIVERSITY of ADELAIDE

Option 1, Treat as outliers



Option 2: Soft margin

Sometimes it is better to allow some instances to be present in the margin space or even cross the boundary
This is called soft-margin SVM



$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i$$

In that case, not all inequality constraints can be satisfied

Slack variables and soft-margin SVM

Slack variables indicates how much a sample violates the inequality constraint

$$\min_{\mathbf{w},b,\{\xi_i\}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_i \xi_i$$

$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i$$

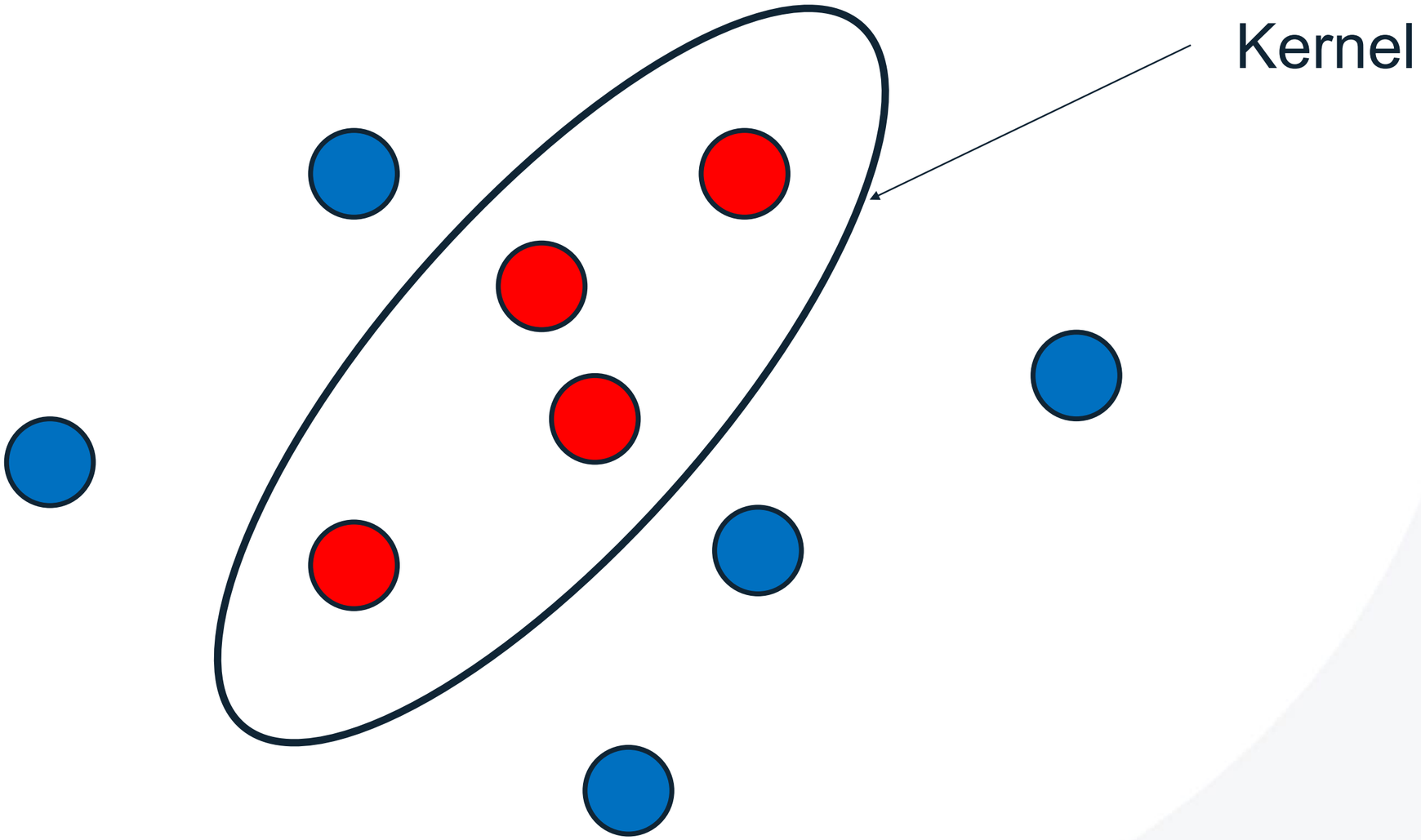
$$\xi_i \geq 0$$

And we want to minimize the total violation

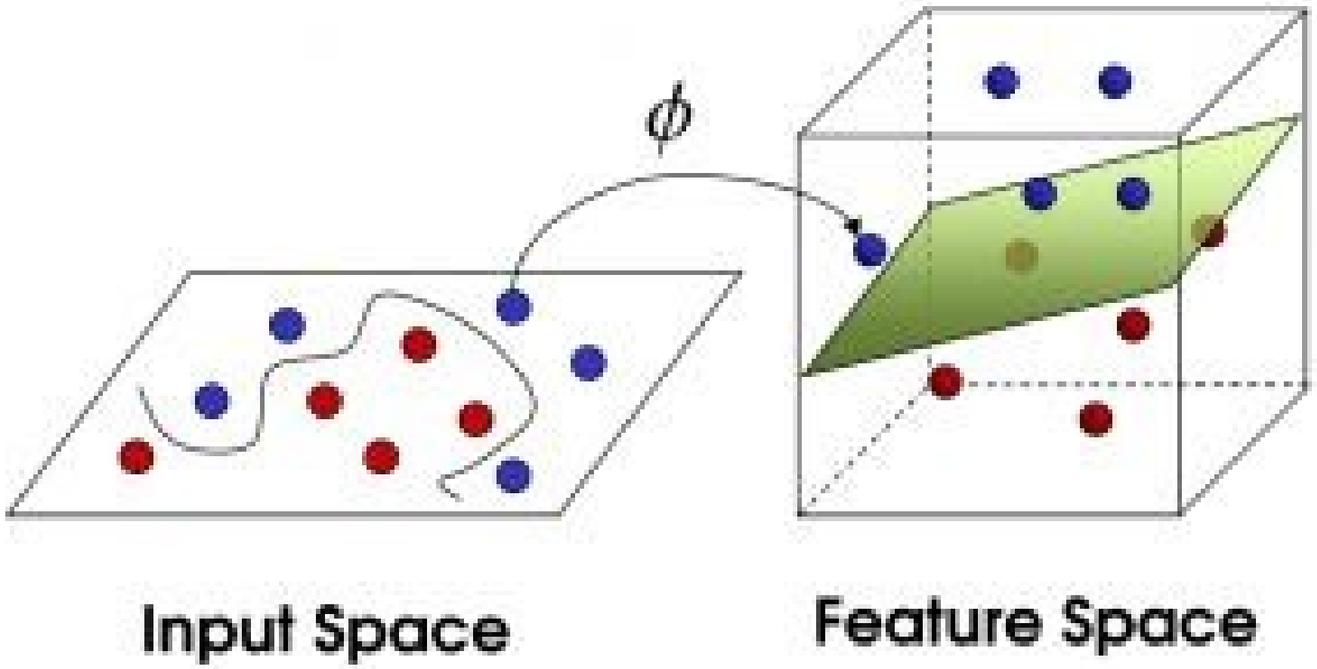
C is a hyper-parameter represents how much violation allowed.

Question: How the margin change if we increase C?

Option 3, Kernelizing



Kernel ideas



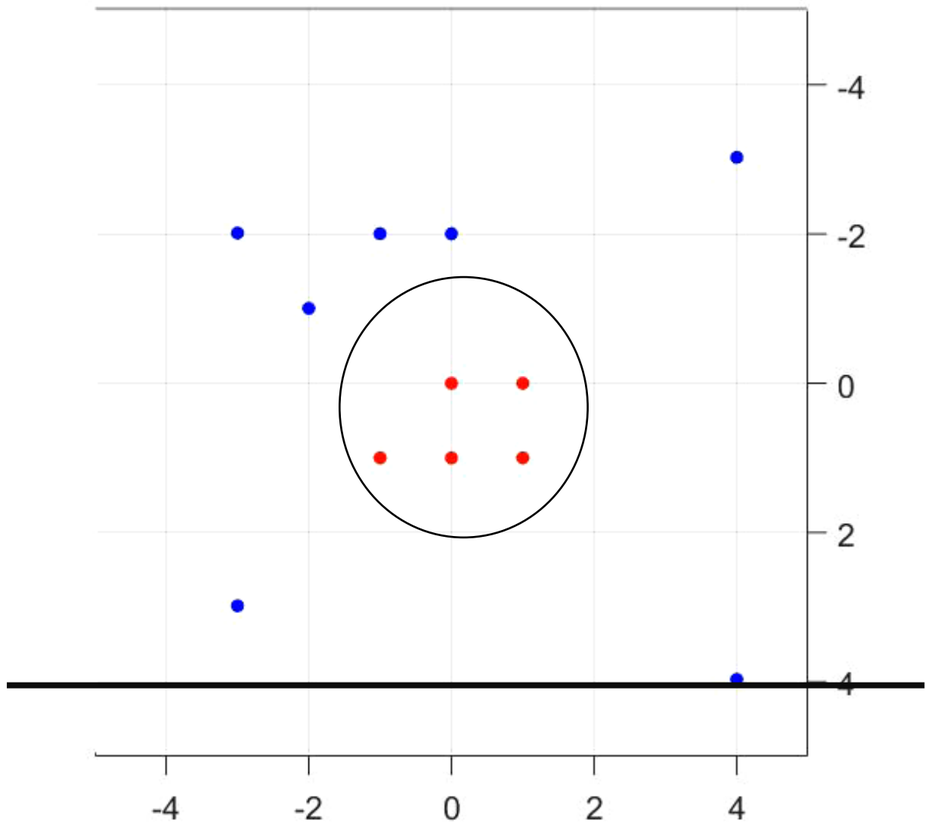
Instances may not be linearly separable.
Transform into a higher dimension space
This is called kernel trick

Kernelizing cont.

We have these points:

- We notice we could draw a circle around them

- We define a **z** as $\mathbf{x^2 + y^2}$



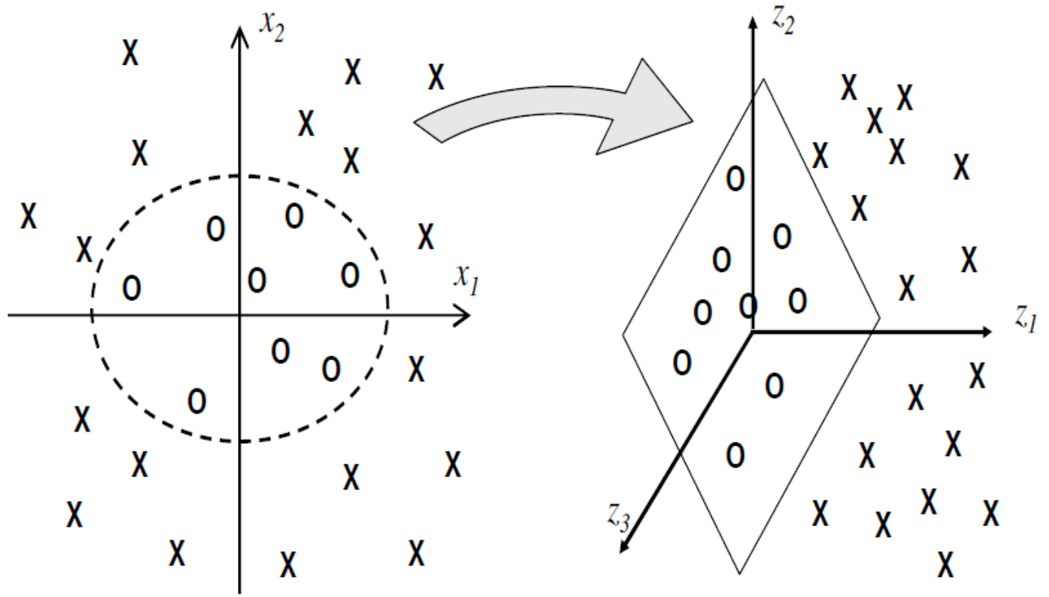
Example with circle (squared) kernel

Consider transformation:

$$(x_1 \cdot x_2) \times (x_1 \cdot x_2) = (x_1^2 + x_2^2) \times (x_1^2 + x_2^2) = x_1^4 + 2x_1^2 x_2^2 + x_2^4$$
$$= \left(x_1^2, x_2^2, \sqrt{2}x_1 x_2\right) \cdot \left(x_1^2, x_2^2, \sqrt{2}x_1 x_2\right)$$

Therefore, we have new 3D coordinates: $x^2, y^2, \sqrt{2}x_1 x_2$
from 2D coordinates

And all we need to calculate is the dot product $(x_1 \cdot x_2)$



Kernelizing cont.

How could this work?

- We pick a function $K(\mathbf{x}_i, \mathbf{x}_j)$ that is easy to compute
- We check that it is the dot-product of something? $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$
- We stop caring about what the transform represents
- We start caring about how well it classifies our data

Commonly used Kernels

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (X_i \cdot X_j + c)^d$$

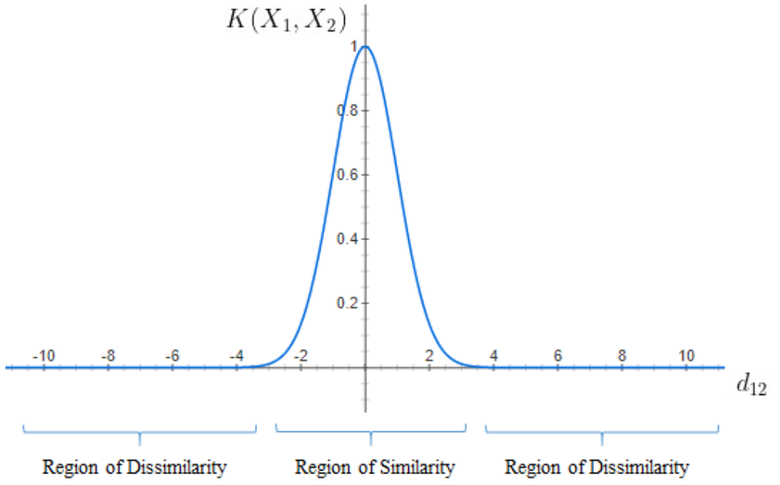
(Polynomial Kernel)

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma(\boldsymbol{x}_i - \boldsymbol{x}_j)^2}$$

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}}$$

(Radial Basis Function or Gaussian Kernel)

(looking to make boundaries in circular region)



Advance topic: Classifying Big Data

- Neural Networks can use Map-Reduce (http://infolab.stanford.edu/~ullman/mmds/ch12n.pdf ch. 12.2.8)
  - Forward pass (Map): a batch of instances for each process
    - If output = target, do noting
    - If output ≠ target, store $(i, \eta y x_i)$
  - Backward pass (Reduce): update weights using stored pairs $(i, \eta y x_i)$

- SVM (http://infolab.stanford.edu/~ullman/mmds/ch12n.pdf ch. 12.3.6)
  - If hyperplane **w** is found using stochastic gradient descent, use procedure similar to NN above.
  - If another optimisation is used, there are variety of methods available, see https://dl.acm.org/doi/pdf/10.1145/3280989

- Note: the algorithms are parallelised already in some libraries (e.g. Spark), no need to rewrite the algorithms in most cases

- Stream classification is another example of Big Data: in Module 11

THE UNIVERSITY
of ADELAIDE

Derivations

$$\sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$

… Math derivation …

$$\boldsymbol{w} = \sum \alpha_i y_i \boldsymbol{x}_i$$

MINIMIZE => $$\sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$

dependent on pairs of samples

w is a weighted sum of the input vectors ($\mathbf{x}_i$)

The optimisation is dependent on dot products of pairs of vectors ($\mathbf{x}_i \cdot \mathbf{x}_j$)

Full math derivations are available

THE UNIVERSITY of ADELAIDE

Kernelizing cont.

The cunning plan…

- What if I apply some transform to my x, y etc
- Then I apply SVM?

Let us call our transformation:

$\phi(\mathbf{x})$

Our old optimisation problem

$$\sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j}$$

Our new optimisation problem

$$\sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x_i}) \cdot \phi(\boldsymbol{x_j})$$

Kernelizing cont.

Example

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma(\boldsymbol{x}_i - \boldsymbol{x}_j)^2}$$

$\phi(x_i) \cdot \phi(x_j)$

Easy to compute

Looks a bit like a **Gaussian**

Assuming $\gamma = 1$ and **x** is a single dimension:

$\phi(x_i)$

Hard to compute

$$e^{-x_i{}^2} e^{-x_j{}^2} \underbrace{\sum_{k=0}^{\infty} \frac{2^k x_i{}^k x_j{}^k}{k!}}_{e^{2x_i x_j}}$$