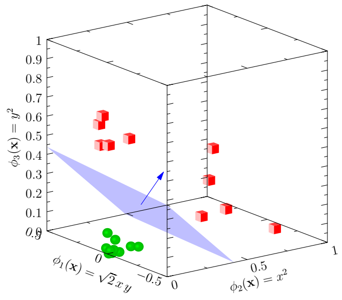
# Advanced Machine Learning

## *Support Vector Machines*

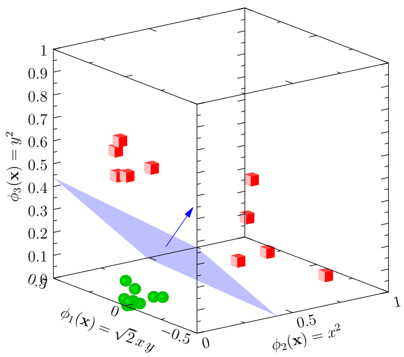



*Support Vector Machines, maximum margins*



# Outline

1. The Big Picture
2. Practice
3. Maximum Margins





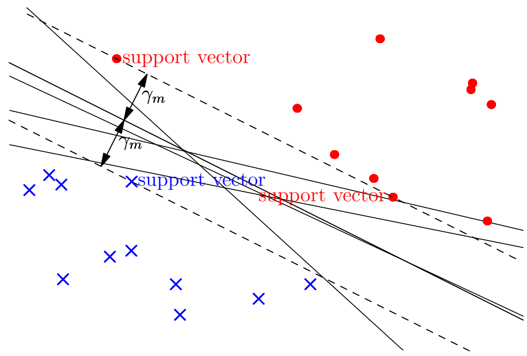

# Support Vector Machines

- Support vector machines, when used right, often have the best generalisation results
- They are typically used on numerical data, but can and have been adapted to text, sequences, etc.
- Although not as trendy as deep learning, they will often be the method of choice on small data sets
- They subtly regularise themselves, choosing a solution that generalises well from a host of different problems



# Linear Separation of Data

- SVMs classify linearly separable data




- Finds maximum-margin separating plane



# Extended Feature Space

- To increase the likelihood of linear-separability we often use a high-dimensional mapping

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_p) \rightarrow \vec{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_r(\boldsymbol{x}))$$

$r \gg p$

- Finding the maximum margin hyper-plane is time consuming in "primal" form if $r$ is large
- We can work in the "dual" space of patterns, then we only need to compute dot products

$$\vec{\phi}(\boldsymbol{x}_i) \cdot \vec{\phi}(\boldsymbol{x}_j) = \sum_{k=1}^{r} \phi_k(\boldsymbol{x}_i)\, \phi_k(\boldsymbol{x}_j)$$



# Kernel Trick

- If we choose a **positive semi-definite** kernel function $K(\boldsymbol{x}, \boldsymbol{y})$ then there exists functions $\phi_k(\boldsymbol{x})$, such that

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \vec{\phi}(\boldsymbol{x}_i) \cdot \vec{\phi}(\boldsymbol{x}_j)$$

  (like an eigenvector decomposition of a matrix)

- Never need to compute $\phi_k(\boldsymbol{x}_i)$ explicitly as we only need the dot-product $\vec{\phi}(\boldsymbol{x}_i) \cdot \vec{\phi}(\boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to compute maximum margin separating hyper-plane
- Sometimes $\vec{\phi}(\boldsymbol{x}_i)$ is an infinite dimensional vector so its good we don't have to compute it!



# Kernel Functions

- Kernel functions are symmetric functions of two variable
- Strong restriction: *positive semi-definite*
- Examples

$$\text{Quadratic kernel:} \quad K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left(\boldsymbol{x}_1^{\mathsf{T}} \boldsymbol{x}_2\right)^2$$

$$\text{Gaussian (RBF) kernel:} \quad K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathrm{e}^{-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$$

- Consider the mapping

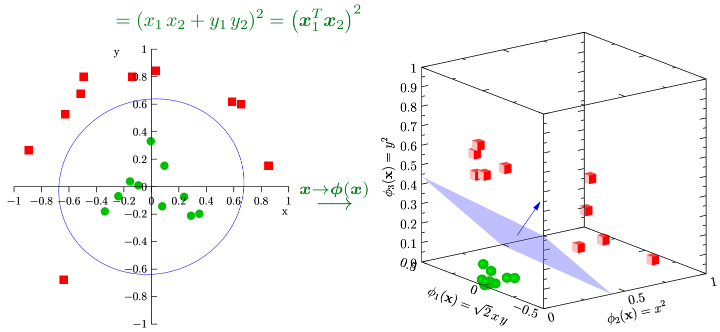
$$\boldsymbol{x}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \rightarrow \boldsymbol{\phi}(\boldsymbol{x}_i) = \begin{pmatrix} x_i^2 \\ y_i^2 \\ \sqrt{2}\, x_i\, y_i \end{pmatrix}$$



# Non-linearly Separation of Data

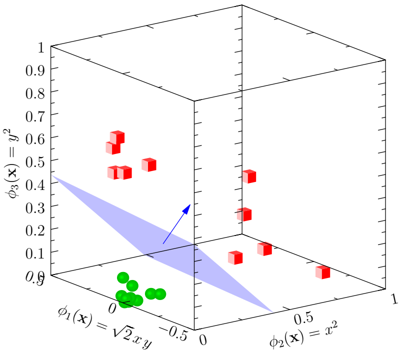
$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{pmatrix} x_1^2 & y_1^2 & \sqrt{2}\, x_1\, y_1 \end{pmatrix} \begin{pmatrix} x_2^2 \\ y_2^2 \\ \sqrt{2}\, x_2\, y_2 \end{pmatrix} = x_1^2\, x_2^2 + y_1^2\, y_2^2 + 2\, x_1\, y_1\, x_2\, y_2$$

$$= (x_1\, x_2 + y_1\, y_2)^2 = \left(\boldsymbol{x}_1^T \boldsymbol{x}_2\right)^2$$



# Outline

1. The Big Picture
2. **Practice**
3. Maximum Margins





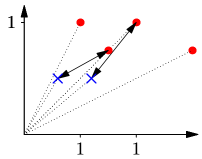

# Computing the Maximum-Margin Hyper-plane

- We will derive the formula for the minimum-margin hyper-plane in the next lecture
- This gives us a quadratic programming problem
- Through a neat trick we can represent this problem in a "dual form" where we
- Never need to compute $\boldsymbol{\phi}_i(\boldsymbol{x})$ only need to compute $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- When we use the kernel trick the time to compute the solution to the quadratic programming problem is $p\,m^3$ where $m$ is the number of training examples and $p$ is the number of features



# Getting SVMs to Work Well

- SVMs rely on distances between data points
- These will change relative to each other if we rescale some features but not other—giving different maximum-margin hyper-planes



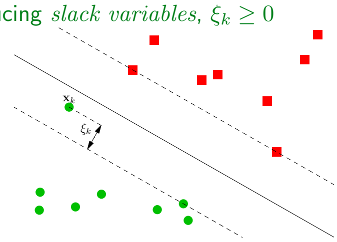
- If we don't know what features are important (most often the case), then it is worth scaling each feature (for example, so their range is between 0 and 1 or their variance is 1)



# Soft Margins

- Sometimes the margin constraint is too severe
- Relax constraints by introducing *slack variables*, $\xi_k \ge 0$

$$y_k(\boldsymbol{x}_k^{\mathsf{T}}\boldsymbol{w} - b) \ge 1 - \xi_k$$




- Minimise $\frac{\|\boldsymbol{w}\|^2}{2} + C\sum_{k=1}^{n}\xi_k$ subject to constraints
- Large $C$ punishes slack variables



# Optimising C

- In practice it can make a huge difference to the performance if we change $C$
- Optimal $C$ values changes by many orders of magnitude e.g. $2^{-5}$–$2^{15}$
- Typically optimised by a grid search (start from $2^{-5}$ say and double until you reach $2^{15}$)



# Choosing the Right Kernel Function

- There are kernels design for particular data types (e.g. string kernels for text or biological sequences)
- For numerical data people tend to look at using no kernel (linear SVM), a radial basis function (Gaussian) kernel or polynomial kernels
- Kernel's often come with parameters, e.g. the popular radial basis function kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\gamma\,\|\boldsymbol{x}-\boldsymbol{y}\|^2}$$

- Optimal $\gamma$ values range over $2^{-15}$–$2^{3}$



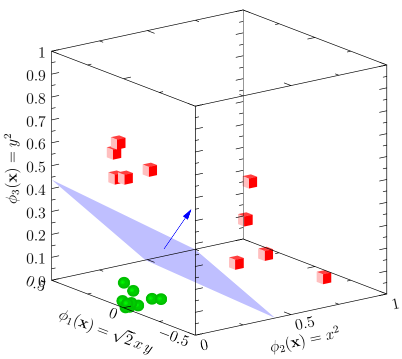
# SVM Libraries

- Although SVMs have unique solutions, they require very well written optimisers
- If you have a large data set they can be very slow
- There are good libraries out there, svmlib, svm-lite, etc.
- These will often automate normalisation of data and grid search for parameters



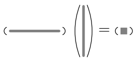
# Outline

1. The Big Picture
2. Practice
3. **Maximum Margins**



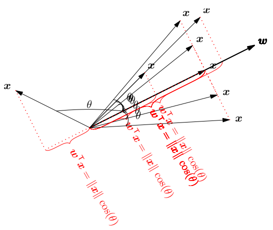
## Dot Product

- Recall the dot product



$$\boldsymbol{x} \cdot \boldsymbol{y} = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{y} = \sum_{i=1}^{n} x_i\, y_i = \|\boldsymbol{x}\|\, \|\boldsymbol{y}\|\, \cos(\theta)$$

- If $\|\boldsymbol{w}\| = 1$ then $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w} = \|\boldsymbol{x}\|\, \cos(\theta)$





## Maximise Margin

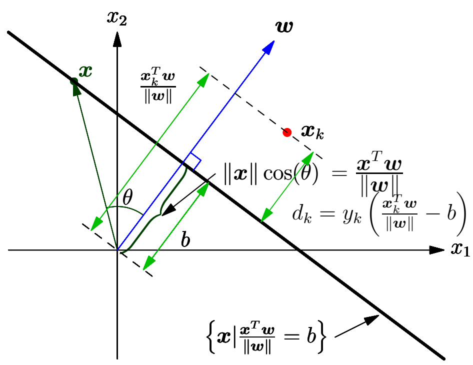
- Consider a linearly separable set of data
  - ⋆ $\mathcal{D} = \{(\boldsymbol{x}_k,\, y_k)\}_{k=1}^{P}$
  - ⋆ $y_k \in \{-1, 1\}$
- Our task is to find a separating plane defined by the orthogonal vector $\boldsymbol{w}$ and a threshold $b$ such that

$$y_k \left( \frac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_k}{\|\boldsymbol{w}\|} - b \right) \geq \gamma_m$$

where $\gamma_m$ is the margin



## Distance to hyperplanes






## Constrained Optimisation

- Wish to find $\boldsymbol{w}$ and $b$ to maximise $\gamma_m$ subject to constraints

$$y_k \left( \frac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_k}{\|\boldsymbol{w}\|} - b \right) \geq \gamma_m \quad \text{for all } k = 1, 2, \ldots, P$$

- If we divide through by $\gamma_m$

$$y_k \left( \frac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_k}{\gamma_m\, \|\boldsymbol{w}\|} - \frac{b}{\gamma_m} \right) \geq 1 \quad \text{for all } k = 1, 2, \ldots, P$$

- Define $\hat{\boldsymbol{w}} = \boldsymbol{w}/(\gamma_m \|\boldsymbol{w}\|)$ and $\hat{b} = b/\gamma_m$

$$y_k \left( \hat{\boldsymbol{w}}^{\mathsf{T}}\boldsymbol{x}_k - \hat{b} \right) \geq 1$$



## Quadratic Programming Problem

- Note that as $\hat{\boldsymbol{w}} = \boldsymbol{w}/(\gamma_m \|\boldsymbol{w}\|)$

$$\|\hat{\boldsymbol{w}}\| = \left\| \frac{\boldsymbol{w}}{m\, \|\boldsymbol{w}\|} \right\| = \frac{1}{\gamma_m}$$

- Minimising $\|\hat{\boldsymbol{w}}\|^2$ is equivalent to maximising the margin $\gamma_m$
- Can write the optimisation problem as a *quadratic programming problem*

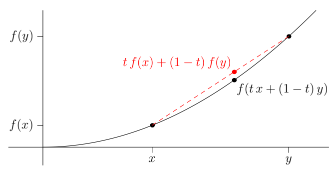
$$\min_{\hat{\boldsymbol{w}}, \hat{b}} \frac{\|\hat{\boldsymbol{w}}\|^2}{2} \quad \text{subject to } y_k \left( \hat{\boldsymbol{w}}^{\mathsf{T}}\boldsymbol{x}_k - \hat{b} \right) \geq 1 \text{ for all } k = 1, 2, \ldots, P$$



## Convexity

- The quadratic function $f(x) = x^2$ is an example of a *convex* function satisfying

$$f(t x + (1-t) y) \leq t f(x) + (1-t) f(y) \quad \text{for } 0 \leq t \leq 1$$

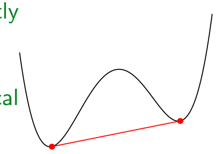



- This extends to high dimensions
  $f(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq t f(\boldsymbol{x}) + (1-t) f(\boldsymbol{y})$



## Unique Minimum

- Convex function have a unique minimum
- The existence of a local minimum would break convexity
  - ⋆ The line connecting a local minimum to a global minimum would be strictly decreasing
  - ⋆ Thus there are points next to the local minimum with lower values
  - ⋆ This is a contradiction

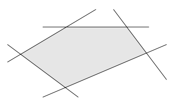


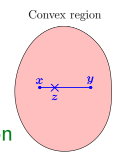
- This remains true if we consider convex functions that are constrained to live in a **convex region**



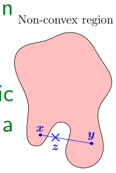
## Convex Regions

- Convex regions are familiar



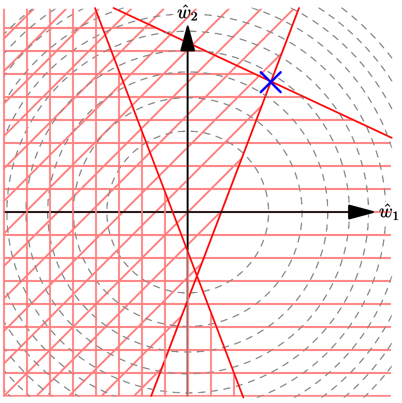



- Convex function constrained to lie in a convex region will also have unique minima as minima can't hide in corners unlike concave regions




- Quadratic programming problems involving a quadratic function and linear constraints are convex and have a unique minimum

## Quadratic Programming in SVMs






## Quadratic Programming

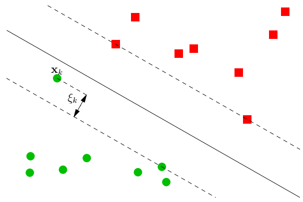
- We have a quadratic programming problem for the weights $\hat{\boldsymbol{w}} = (\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_n)$ and bias $\hat{b}$ and $m$ constraints
- This is a classic but fiddly optimisation problems
- It can be solved in $O(m^3)$ time (it involves inverting matrices) (phew it is not NP-complete!)
- We will see that there is an equivalent dual problem which allows us to use the kernel trick



## Soft Margins

- Can relax constraints by introducing *slack variables*, $\xi_k \geq 0$

$$y_k(\hat{\boldsymbol{w}}^\mathsf{T} \boldsymbol{x}_k - \hat{b}) \geq 1 - \xi_k$$




- Minimise $\frac{\|\hat{\boldsymbol{w}}\|^2}{2} + C \sum_{k=1}^{m} \xi_k$ subject to constraints
- We've added a linear function that leaves our objective convex



## Convexity of Lasso

- Recall that in Lasso we are asked to minimise

$$E = \sum_{j=1}^{m} \left(\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_j - y_j\right)^2 + \nu \sum_{i=1}^{p} |w_i|$$

- We can rewrite this (using $w_i = w_i^+ - w_i^-$) as

$$E = \sum_{n=1}^{N} \left((\boldsymbol{w}^+ - \boldsymbol{w}^-)^\mathsf{T} \boldsymbol{x}_n - y_n\right)^2 + \nu \sum_{i=1}^{p} (w_i^+ + w_i^-)$$

$$\text{subject to} \quad w_i^+, w_i^- \geq 0 \quad \text{for all } i$$

- Again it Lasso has a unique minima



## Conclusions

- We've seen how SVMs work
- We've learnt how to use them
- We've seen that we can find the maximum margin hyper-plane by solving a quadratic programming problem (with a unique solution)
- This is a convex optimisation problem with a unique optimum
- Next we will look at the **dual problem**