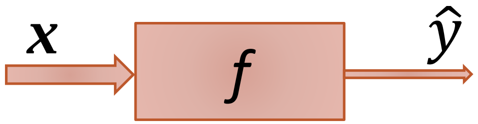
Support Vector Machines

CMT307 Session 5

Yuhua Li
liy180@cardiff.ac.uk

Support Vector Machines

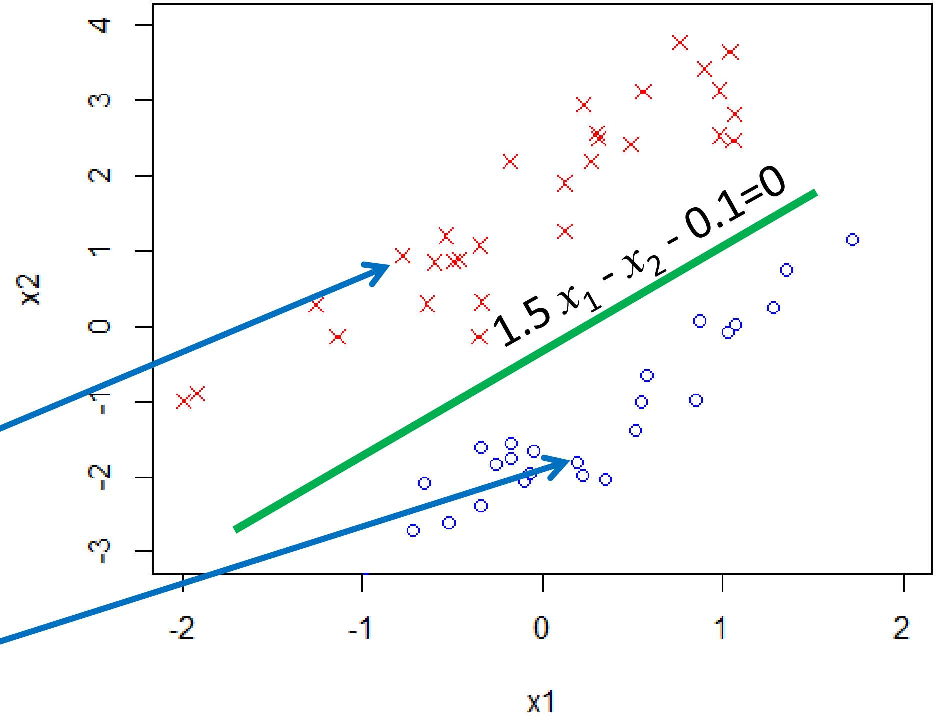
- Linear classifiers for binary classes



$$f(x,w,b) = sign(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

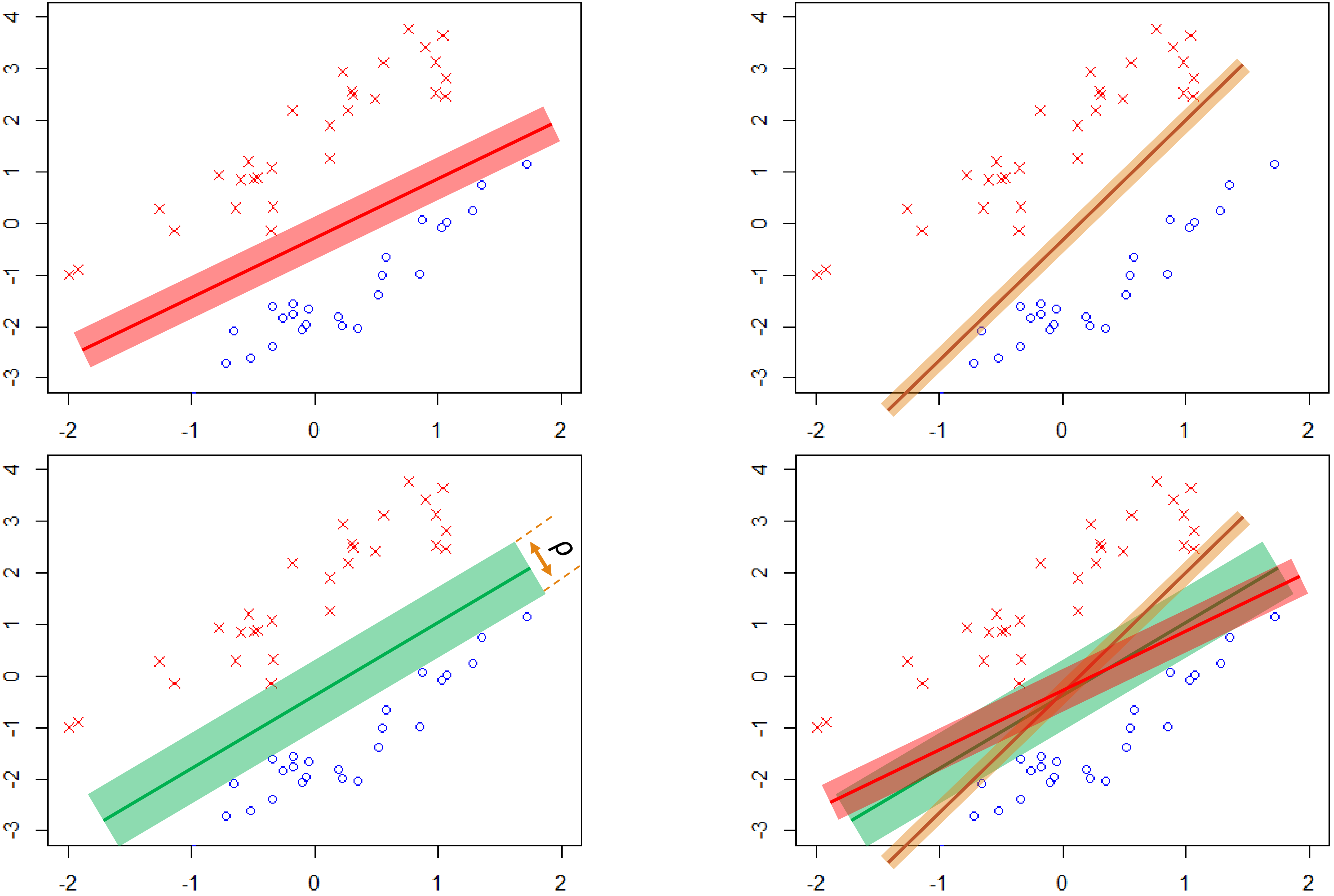
$$1.5x_1 - x_2 - 0.1 = 0$$

$\boldsymbol{w}$=(1.5, -1), $\boldsymbol{b}$=−0.1



For a point (-0.8, 1), *sign*(1.5*(-0.8) - 1 – 0.1) = *sign*(-2.3) = -1

For a point (0.1, -2), *sign*(1.5*0.1 - (-2) – 0.1) = *sign*(2.05) = +1



-2
-1
0
1
2
3
4

Support Vector Machines

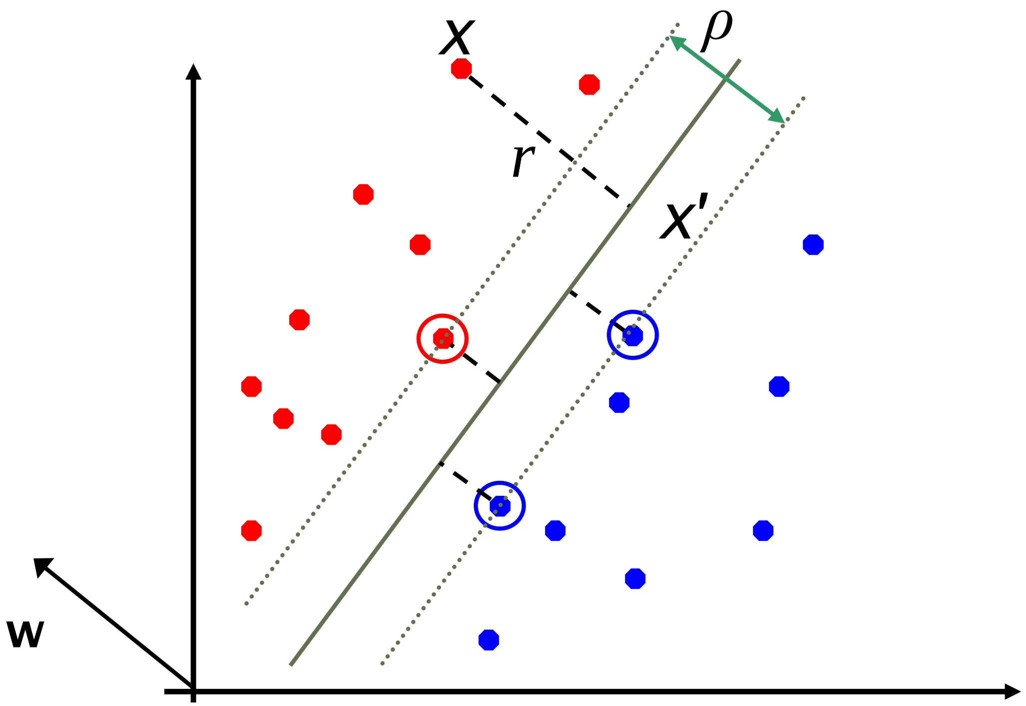
- As the 2-d example shown on previous slide, the two classes can be separated by different lines.
- The points nearest to the separation/decision line (or plane in 3-d or hyperplane in higher dimension) define the margin(i.e., gap) between classes.
- Those points are called support vectors.
- Support vector machine aims to maximise the margin.

Maximum Margin: Formalization

- $\boldsymbol{w}$: decision hyperplane normal vector
- $\boldsymbol{x_i}$: data point i
- y_i: class of data point i (+1 or -1) NB: Not 1/0
- Classifier is: $f(\boldsymbol{x}_i) = sign(\mathbf{w}^T \boldsymbol{x}_i + b)$
- Functional margin of $\boldsymbol{x}_i$ is: $y_i(\mathbf{w}^T \boldsymbol{x}_i + b)$
  - But note that we can increase this margin simply by scaling ***w***, ***b***….
- Functional margin of dataset is **twice** the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin

Geometric Margin

- Distance from example to the separator is $r = y\dfrac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** *ρ* of the separator is the width of separation between support vectors of classes.



Derivation of finding *r*:

Dotted line **x'**−**x** is perpendicular to decision boundary so parallel to **w**.

Unit vector is **w**/|**w**|, so line is *r* **w**/|**w**|.

x' = **x** – *yr***w**/|**w**|.

x' satisfies $\mathbf{w}^T\mathbf{x}' + b = 0$.

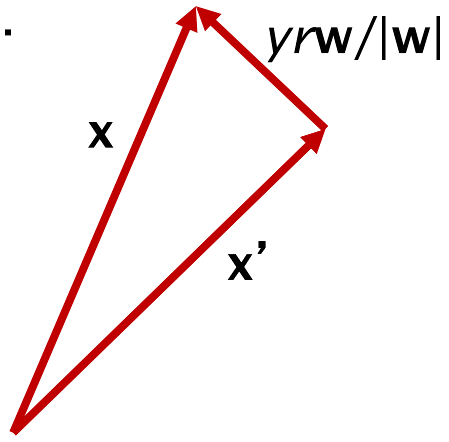
So $\mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$

Recall that $|\mathbf{w}| = \mathrm{sqrt}(\mathbf{w}^T\mathbf{w})$.

So $\mathbf{w}^T\mathbf{x} - yr|\mathbf{w}| + b = 0$

So, solving for *r* gives:

$r = y(\mathbf{w}^T\mathbf{x} + b)/|\mathbf{w}|$



Linear SVM Mathematically

The linearly separable case

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w^T x_i} + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example's distance from the hyperplane is

$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

$$r = \frac{2}{\|\mathbf{w}\|}$$

Linear Support Vector Machine (SVM)

- **Hyperplane**

  $\mathbf{w}^T \mathbf{x} + b = 0$

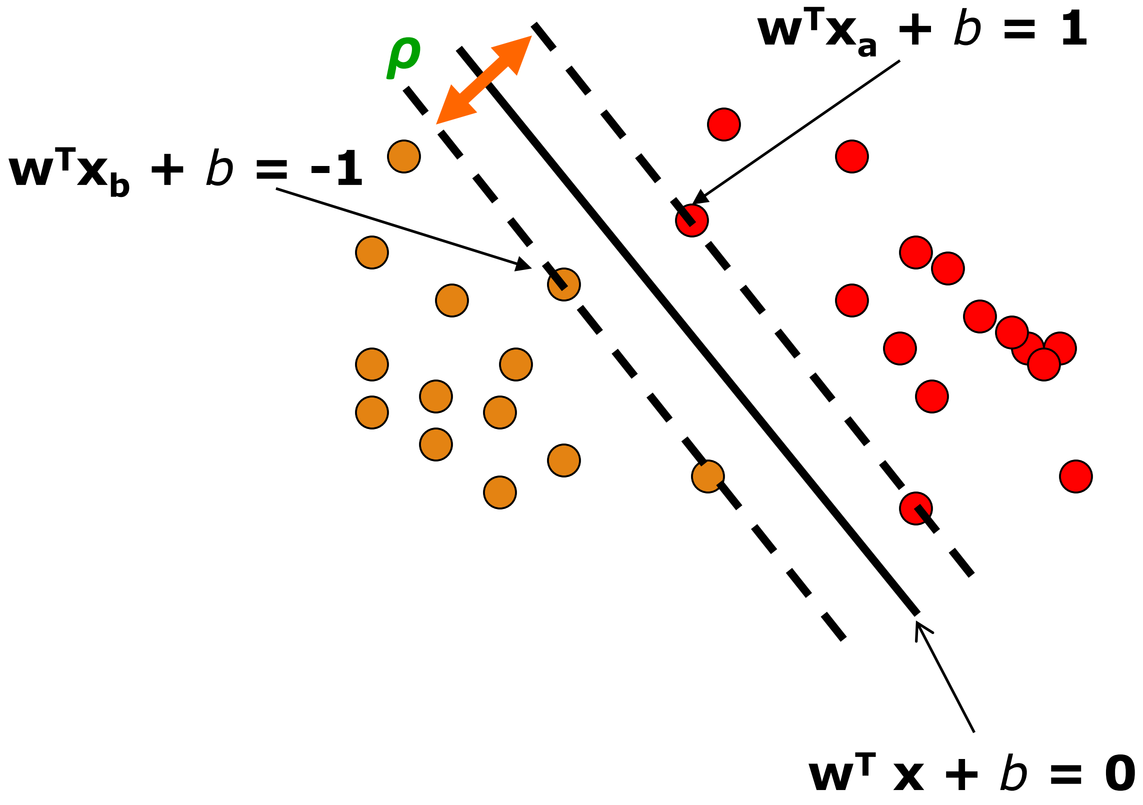
- **Extra scale constraint**:

  $\mathbf{min_{i=1,\ldots,n} |w^T x_i + \mathit{b}| = 1}$

- This implies:

  $w^T(x_a - x_b) = 2$

  $\rho = ||x_a - x_b||_2 = \mathbf{2/||w||_2}$



Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

> Find **w** and *b* such that
>
> $r = \frac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$
>
> $\mathbf{w}^T\mathbf{x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w}^T\mathbf{x_i} + b \leq -1$ if $y_i = -1$

- A better formulation (min **||w||** = max 1/ **||w||**):

> Find **w** and *b* such that
>
> $L(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w}$ is minimized;
>
> and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i (\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$L(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - ½\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

$$\boxed{\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^{\mathrm{T}}\mathbf{x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0}$$

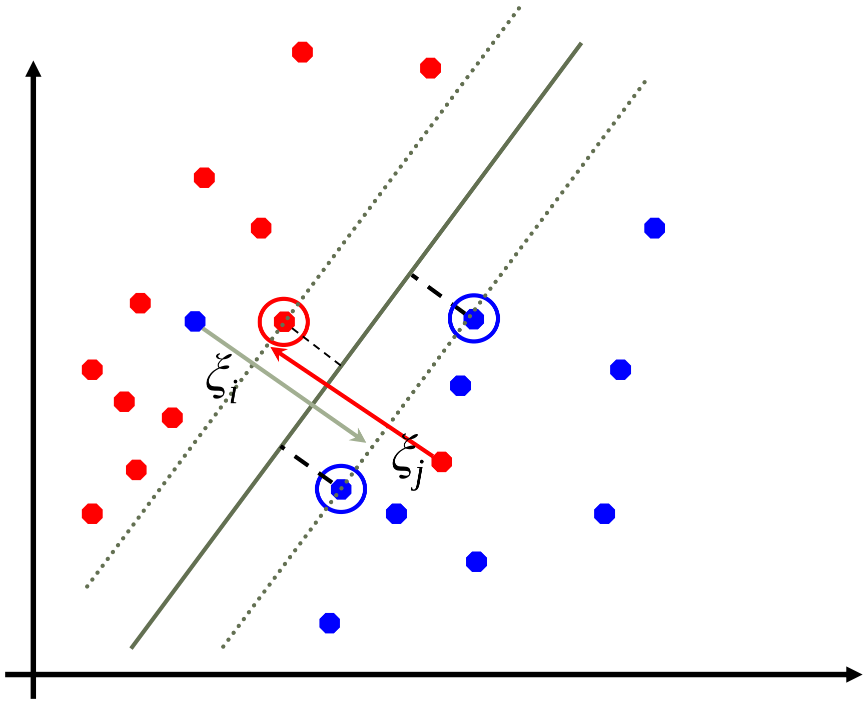
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$\boxed{f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^{\mathrm{T}}\mathbf{x} + b}$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x_i}$
  - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i}^{\mathrm{T}}\mathbf{x_j}$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

> Find **w** and b such that
>
> $L(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
>
> $y_i\,(\mathbf{w^T x_i} + b) \geq 1$

- The new formulation incorporating slack variables:

> Find **w** and b such that
>
> $L(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
>
> $y_i\,(\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter *C* can be viewed as a way to control overfitting
  - A regularization term

Regularisation

- Support vector machines
  - Training SVM involves regularization by default, its strength of regularization is determined by the penalty parameter *C*>0.

$$\min \quad \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i$$

  - Larger values of *C*: less regularization
    - Fit the training data as well as possible
    - Each individual data point is important to classify correctly
  - Smaller values of *C*: more regularization
    - More tolerant of errors on individual data points

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

$$
\boxed{
\begin{array}{l}
\text{Find } \alpha_1 \ldots \alpha_N \text{ such that} \\
\quad \mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \tfrac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^{\mathbf{T}}\mathbf{x_j} \text{ is maximized and} \\
\quad (1)\ \ \Sigma\alpha_i y_i = 0 \\
\quad (2)\ \ 0 \leq \alpha_i \leq C \text{ for all } \alpha_i
\end{array}
}
$$

- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!
- Again, $\mathbf{x_i}$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$$
\boxed{
\begin{array}{l}
\mathbf{w} \ = \Sigma\alpha_i y_i \mathbf{x_i} \\
b \ = y_k(1 - \xi_k) - \mathbf{w}^{\mathbf{T}}\mathbf{x}_k \text{ where } k = \operatorname{argmax}\ \alpha_{k'}
\end{array}
}
$$

k'

w is not needed explicitly for classification!

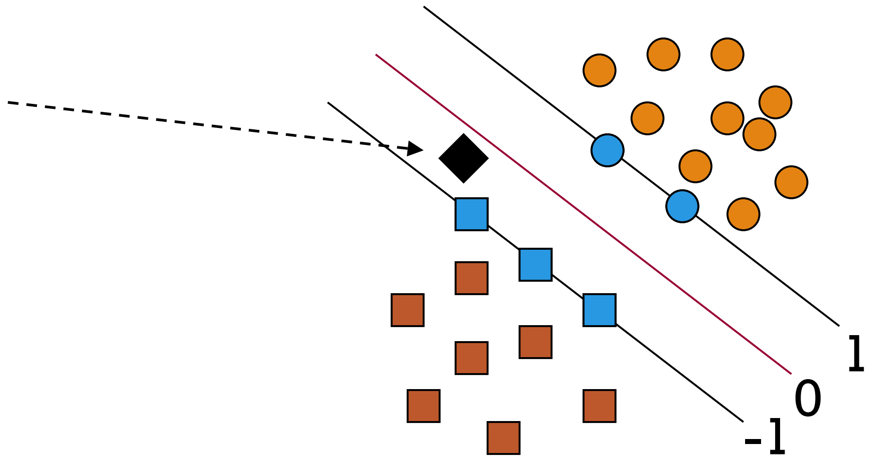
$$
\boxed{f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^{\mathbf{T}}\mathbf{x} + b}
$$

Classification with SVMs

- Given a new point **x**, we can score its projection onto the hyperplane normal:
  - I.e., compute score: $\mathbf{w}^T\mathbf{x} + b = \Sigma \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$
    - Decide class based on whether < or > 0

  - Can set confidence threshold *t*.

Score > *t*:	yes
Score < -*t*:	no
Else:	don't know



Linear SVMs: Summary

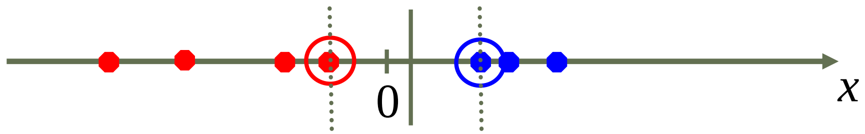
- The classifier is a *separating hyperplane.*
- The most "important" training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x_i}^T\mathbf{x_j}}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

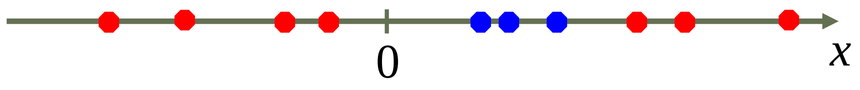
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x_i}^T\mathbf{x}} + b$$

Non-linear SVMs

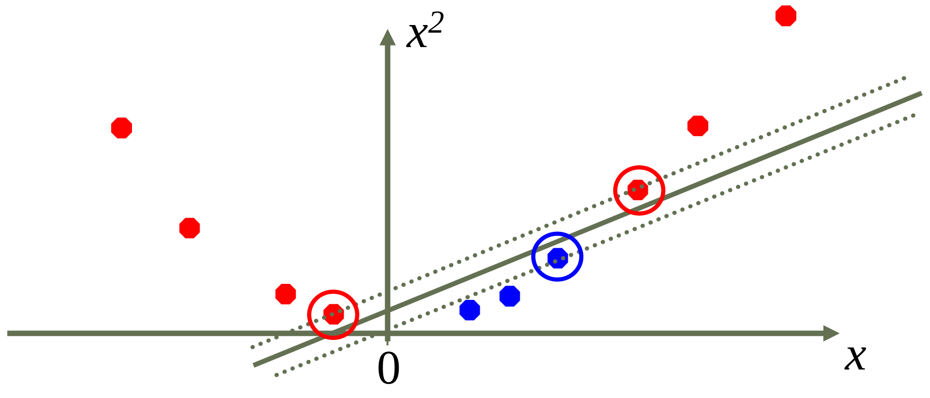
- Datasets that are linearly separable (with some noise) work out great:

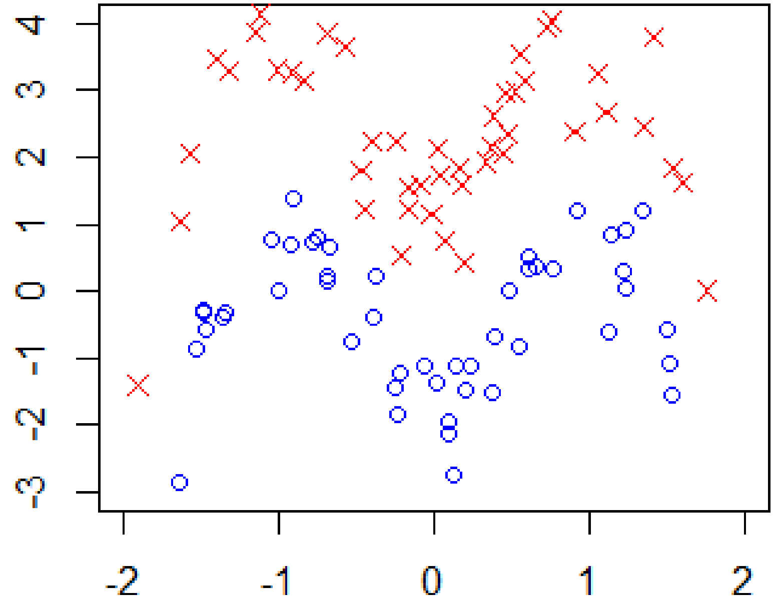


- But what are we going to do if the dataset is just too hard?



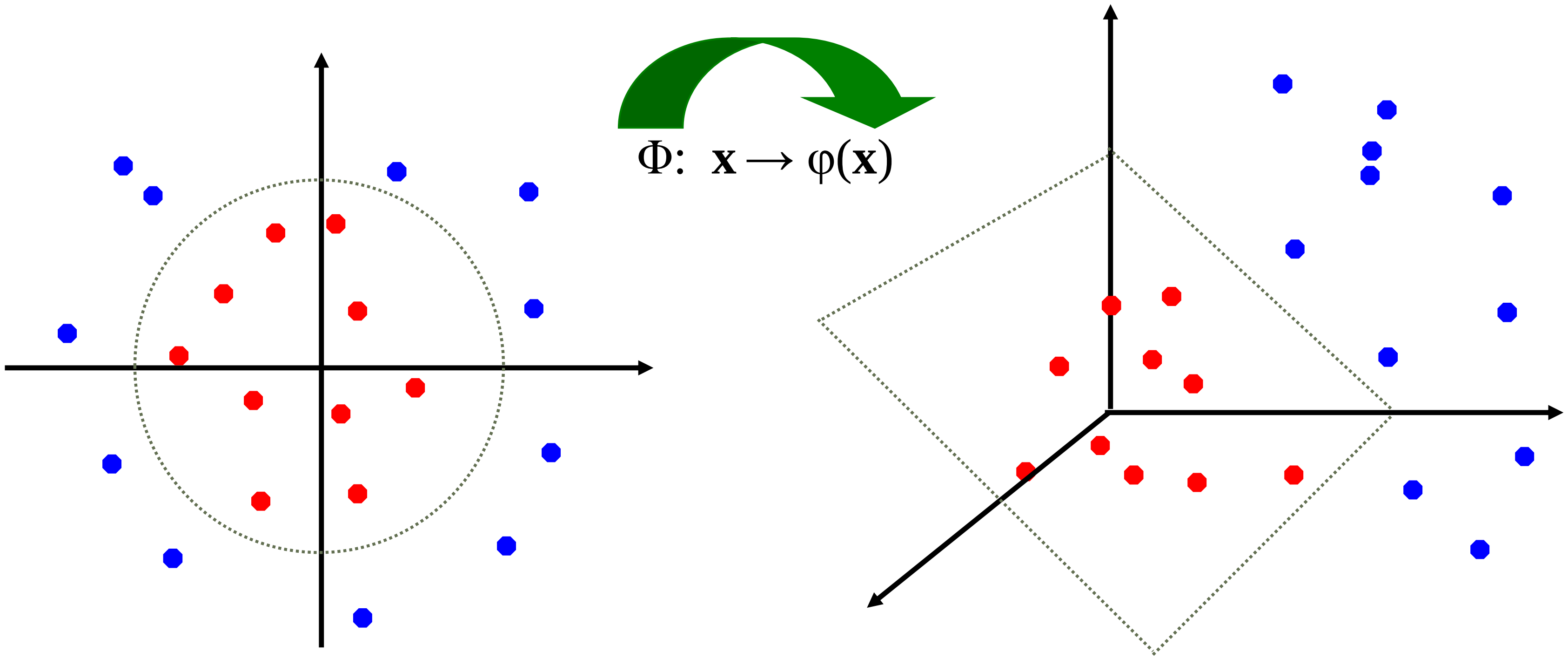
- How about … mapping data to a higher-dimensional space:





Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The “Kernel Trick”

- The linear classifier relies on an inner product between vectors $K(\mathbf{x_i},\mathbf{x_j})=\mathbf{x_i}^\mathbf{T}\mathbf{x_j}$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^\mathbf{T}\varphi(\mathbf{x_j})$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$; let $K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^\mathbf{T}\mathbf{x_j})^2$,

Need to show that $K(\mathbf{x_i},\mathbf{x_j})= \varphi(\mathbf{x_i})^\mathbf{T}\varphi(\mathbf{x_j})$:

$$K(\mathbf{x_i},\mathbf{x_j})=(1 + \mathbf{x_i}^\mathbf{T}\mathbf{x_j})^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^\mathbf{T} [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$$

$$= \varphi(\mathbf{x_i})^\mathbf{T}\varphi(\mathbf{x_j}) \quad \text{where } \varphi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$$

Non-linear SVMs: Feature spaces

- SVM uses a kernel function (which is a similarity measure between data points) to map data from input feature space to a higher-dimensional feature space
- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^d$
  - Radial basis function $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$

Reading

- Aurélien Géron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems. O'Reilly, 2017. **Chapter 5, Appendix C** for a rudimentary introduction.

- Christopher Bishop. Pattern Recognition and Machine Learning. Springer 2006. https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf
Section 7.1 for a more rigorous introduction.