Lecture 16-17: Support Vector Machines (SVMs)

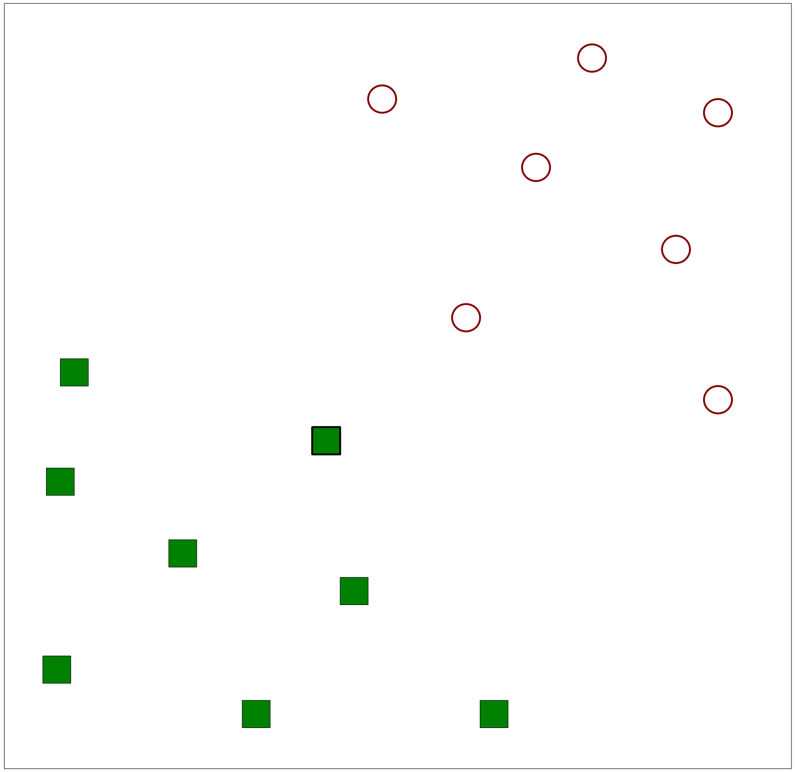
COMP90049
Knowledge Technology

Sarah Erfani and Vinh Nguyen, CIS

Semester 2, 2017

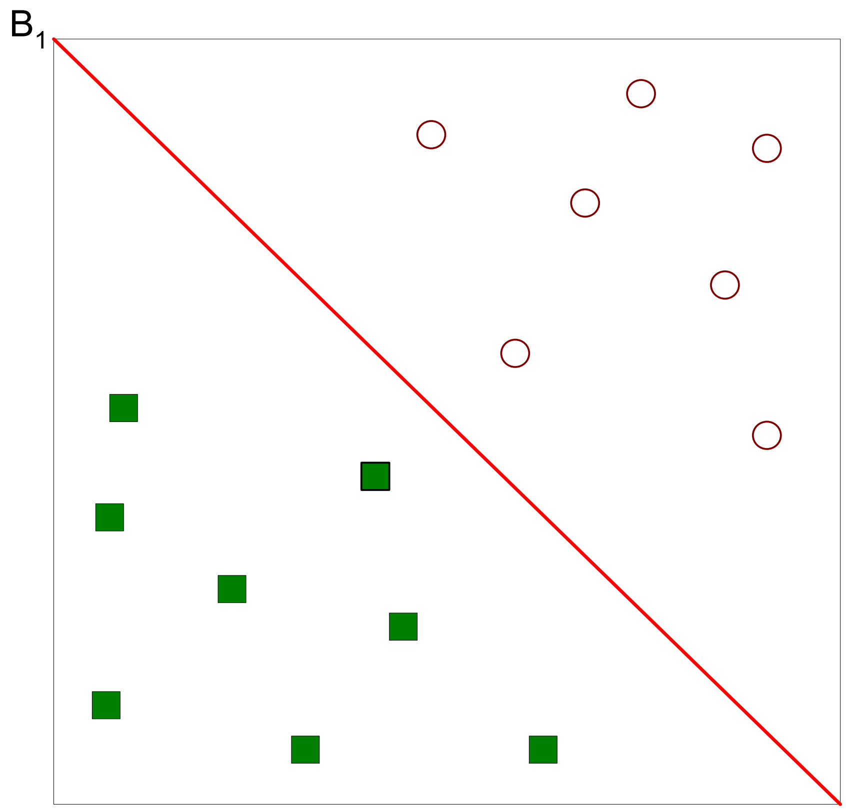
Support Vector Machines: Intuition

- Assuming the data is linearly separable
- Aim: find a linear hyperplane (decision boundary) that will separate the data



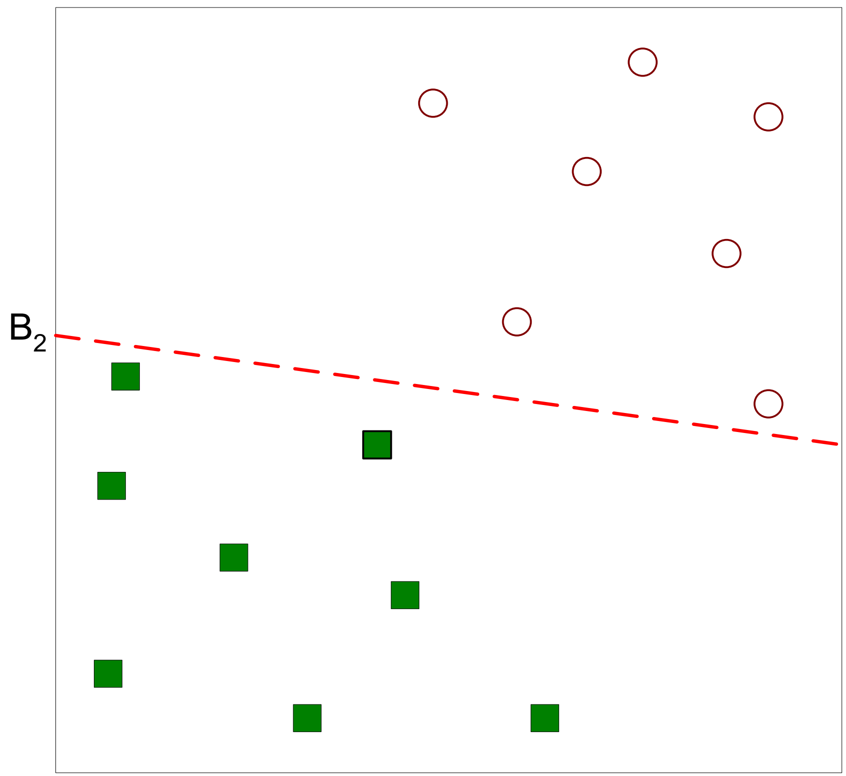
Support Vector Machines: Intuition

- One Possible Solution



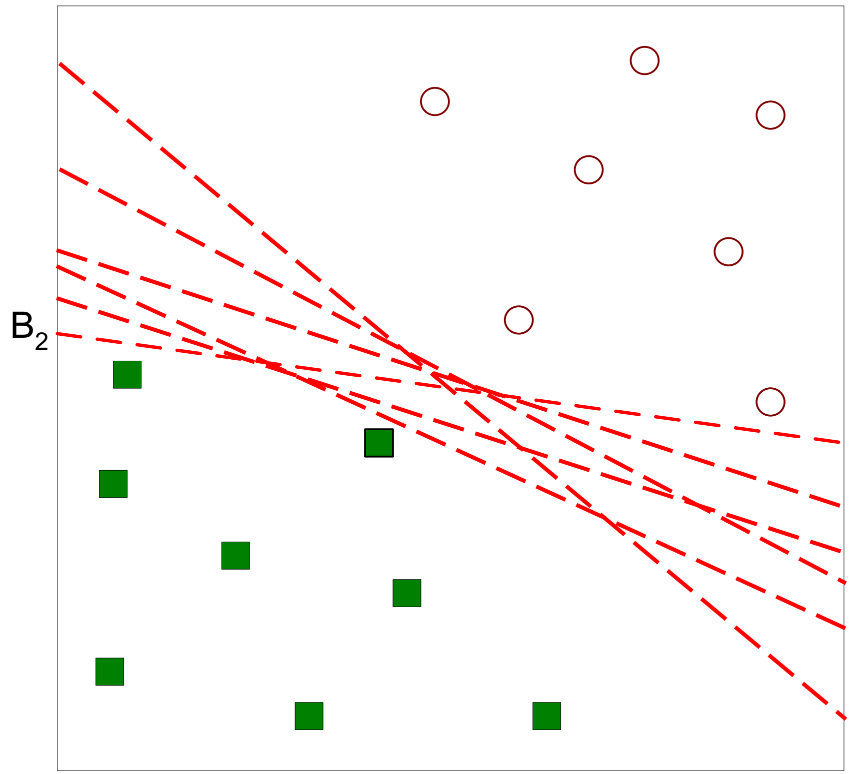
Support Vector Machines: Intuition

- Another Possible Solution



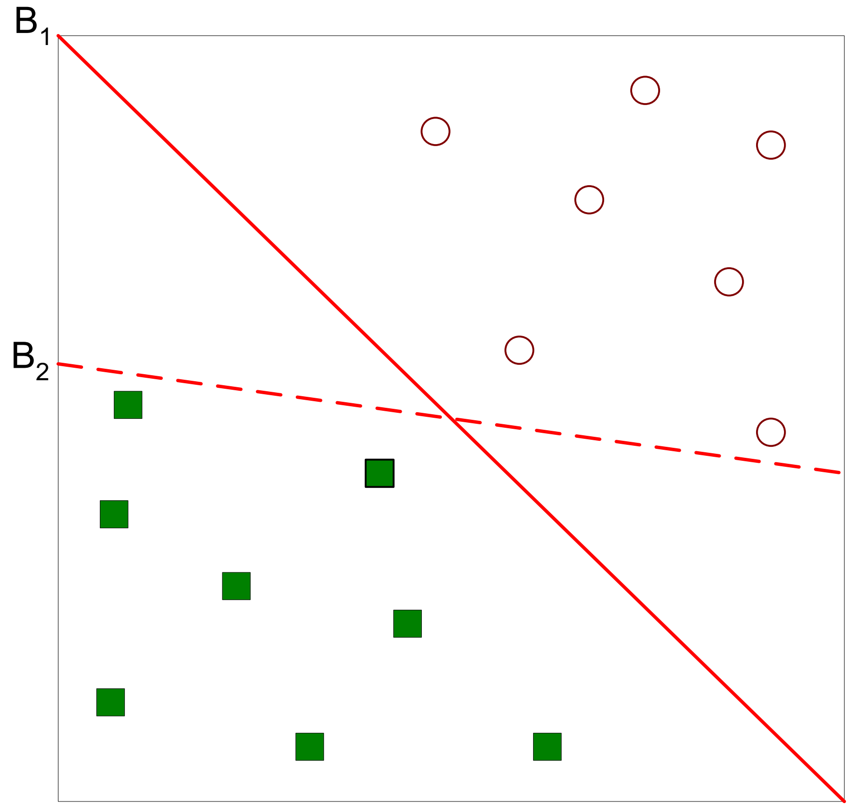
Support Vector Machines: Intuition

- Other Possible Solutions



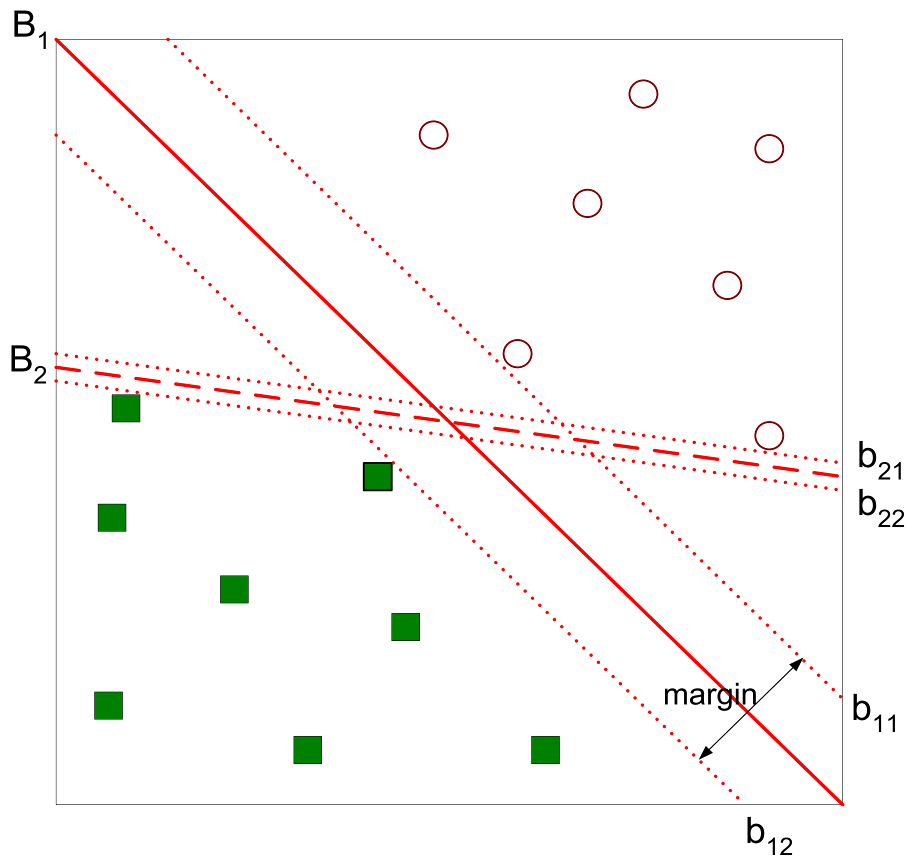
Support Vector Machines: Intuition

- Which one is better? B1 or B2?
- How do you define better?

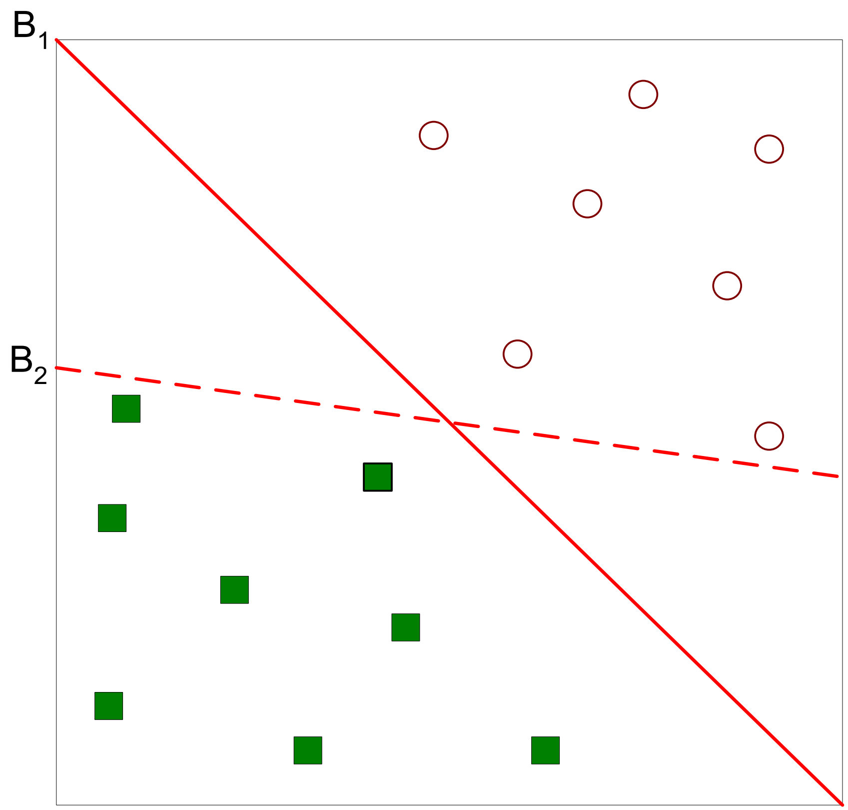


Support Vector Machines: Large Margin Classifiers

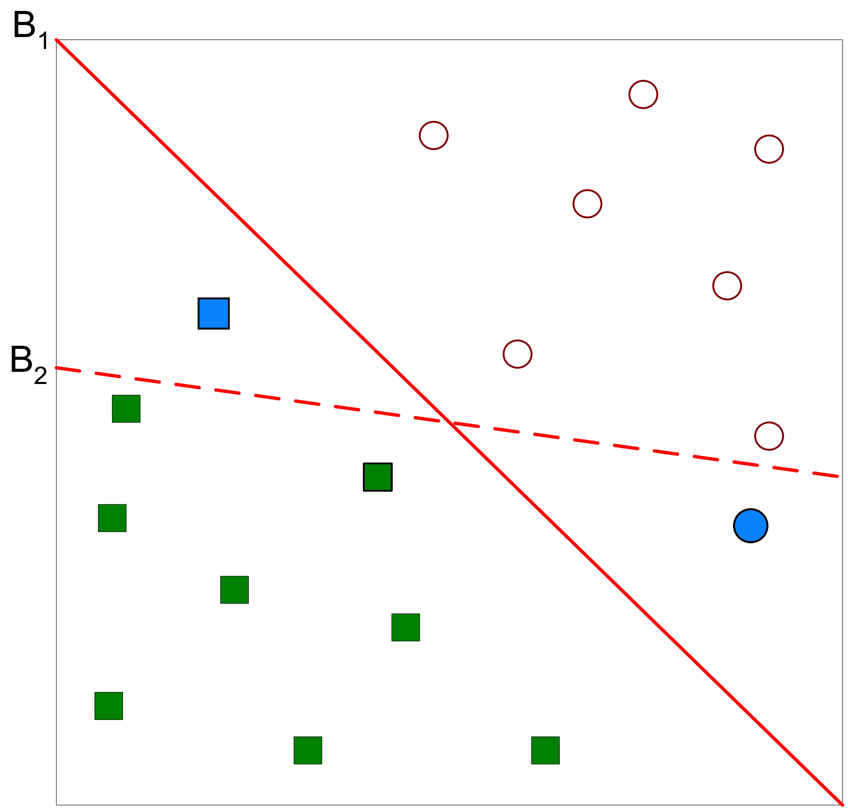
- Find hyperplane maximises the margin => B1 is better than B2
- Margin: sum of shortest distances from the planes to the positive/negative samples



Why Large Margin?



Why Large Margin?



Why Large Margin?

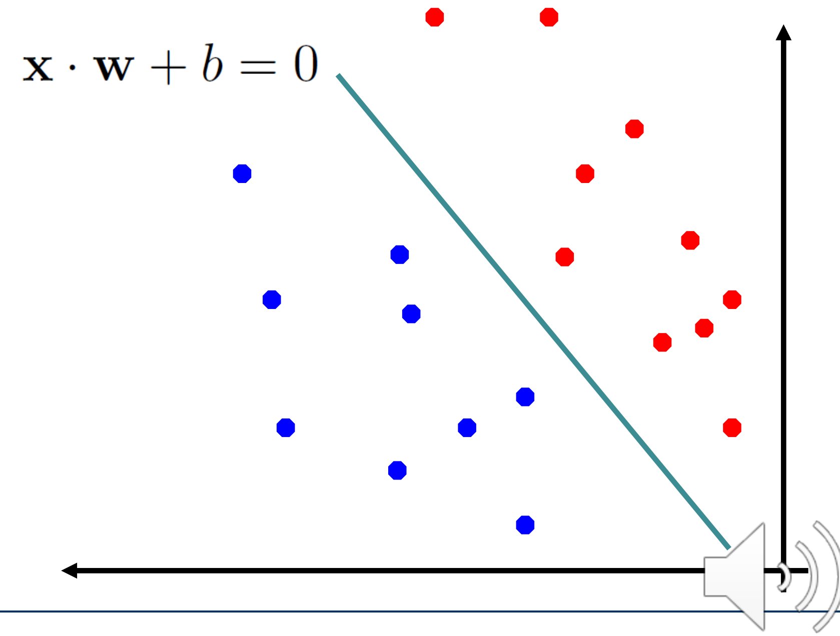
- Small margin separating planes:
  - are more fragile to noise
  - may over-fit the data
- Large margin separating planes:
  - are more robust to noise
  - From statistical learning theory: large margin planes generalises better to unseen data

Linear Classifiers Formulation

$\{\mathbf{x}_i, y_i\}$ where $i = 1 \ldots L, y_i \in \{-1, 1\}, \mathbf{x}_i \in \mathbb{R}^D$

This hyperplane can be described by $\mathbf{x} \cdot \mathbf{w} + b = 0$ where:

- $\mathbf{w}$ is normal to the hyperplane.
- $\frac{b}{\|\mathbf{w}\|}$ is the perpendicular distance from the hyperplane to the origin.



Linear Classifiers Formulation

$$\{\mathbf{x}_i, y_i\} \text{ where } i = 1 \ldots L, y_i \in \{-1, 1\}, \mathbf{x}_i \in \mathbb{R}^D$$

This hyperplane can be described by $\mathbf{x} \cdot \mathbf{w} + b = 0$ where:

- $\mathbf{w}$ is normal to the hyperplane.
- $\frac{b}{\|\mathbf{w}\|}$ is the perpendicular distance from the hyperplane to the origin.

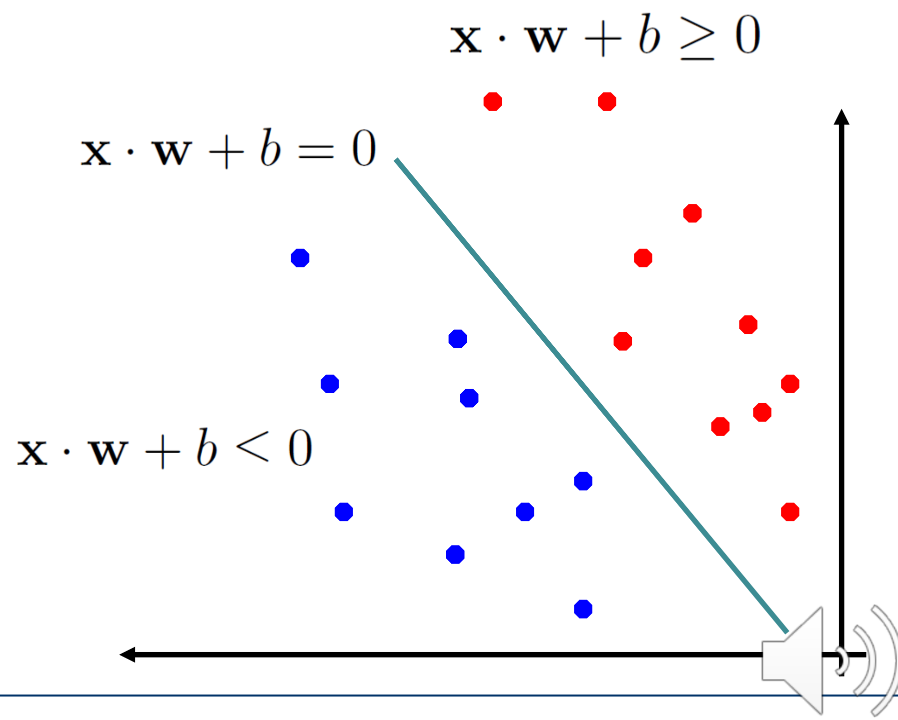
Classification rule

$$f(\mathbf{x}) = \text{sign}(\mathbf{x} \cdot \mathbf{w} + b) = \begin{cases} +1 \text{ if } \mathbf{x} \cdot \mathbf{w} + b \geq 0 \\ -1 \text{ if } \mathbf{x} \cdot \mathbf{w} + b < 0 \end{cases}$$

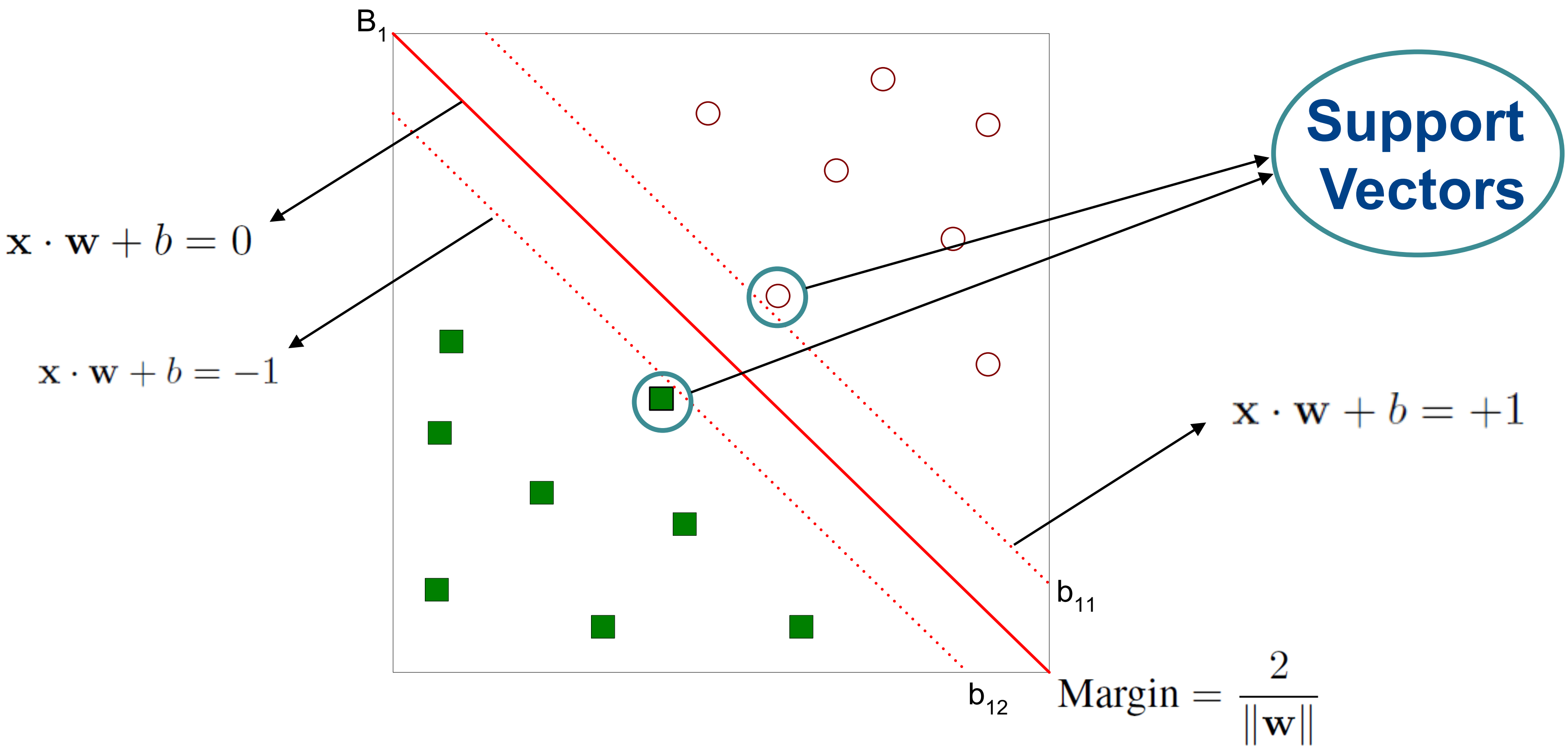
Find $\mathbf{w}$ and b such that:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq 0 \text{ for } y_i = +1$$
$$\mathbf{x}_i \cdot \mathbf{w} + b < 0 \text{ for } y_i = -1$$
$$\text{for all } i = 1 \ldots L$$

Training objective



Linear Support Vector Machines: Need to Consider Margin



Requirement for margin:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1$$

Linear Support Vector Machines Formulation

$$\max \frac{2}{\|\mathbf{w}\|}$$

Margin

Subject to:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \text{ for } y_i = +1$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \text{ for } y_i = -1$$

Note that:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1$$

These equations can be combined into:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \ \forall_i$$

Linear Support Vector Machines Equivalent Formulations

(1) $$\max \frac{2}{\|\mathbf{w}\|} \quad \text{s.t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i$$

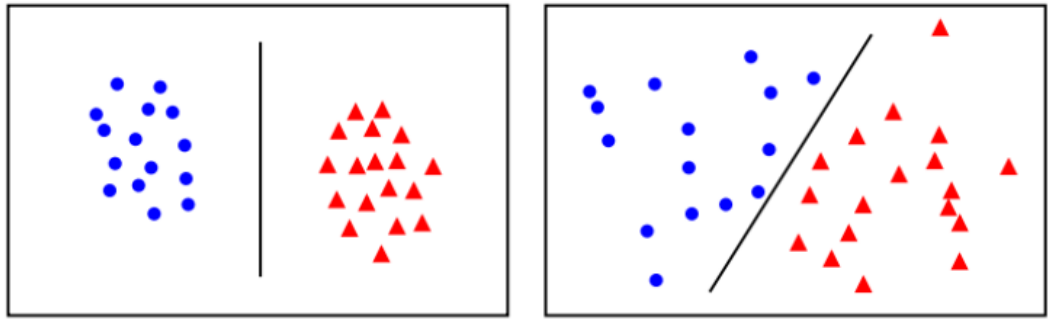
(2) $$\min \|\mathbf{w}\| \quad \text{s.t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall_i$$

(3) $$\min \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall_i$$

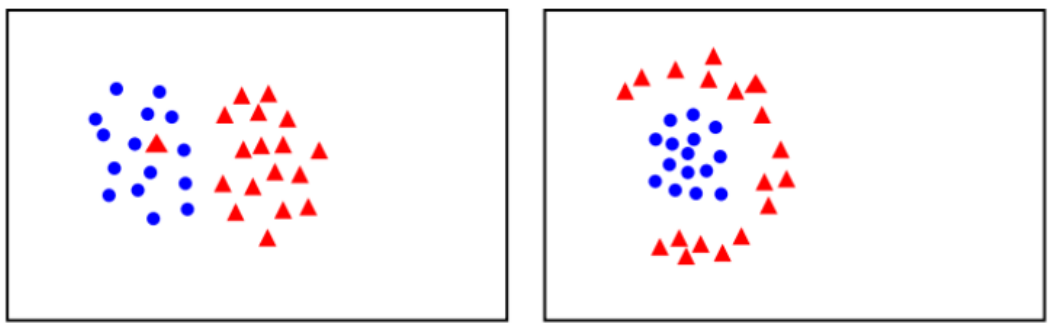
Linear SVM Feasibility

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t.} \qquad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \;\; \forall_i$$

linearly separable



not linearly separable

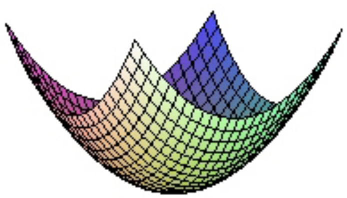


- For linearly separable data: a max-margin solution is **guaranteed** to exist
- For non- linearly separable data: a solution does not exist

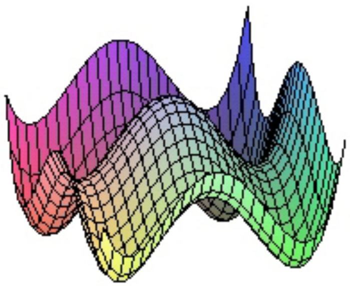
Solving the Optimization Problem

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t.} \qquad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \;\; \forall_i$$

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Convex quadratic optimization problem
- Convex objective: any local minimum is also a global minimum



Convex



Non Convex

Solving the Optimization Problem: Duality Formulation

***Primal problem**: solve for **w** and b*

$$\min \frac{1}{2} \|\mathbf{w}\|^2 \qquad \text{s.t.} \qquad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \ \ \forall_i$$

Solving the Optimization Problem: Duality Formulation

***Primal problem**: solve for **w** and b*

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t.} \qquad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall_i$$

*Equivalent **dual problem** formulation: solve for $\alpha_1 \ldots \alpha_L$*: Lagrange multipliers for each data point

$$\max_{\alpha} \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

s.t.

$$\alpha_i \geq 0$$

$$\sum_{i=1}^{L} \alpha_i y_i = 0$$

More convenient to solve

See Ref. [1] for derivation

Solution: Dual to Primal

- Given a solution $\alpha_1 \ldots \alpha_L$ to the dual problem, solution to the primal is:

$$\mathbf{w} = \Sigma \alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma \alpha_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_k \quad \text{for any } \alpha_k > 0$$

- Each non-zero α_i indicates that corresponding $\mathbf{x}_i$ is a support vector.
- Then the classifying function is (note that we don't need **w** explicitly):

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

Classification rule

- Notice that it relies on an *inner product* between the test point **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 training points.

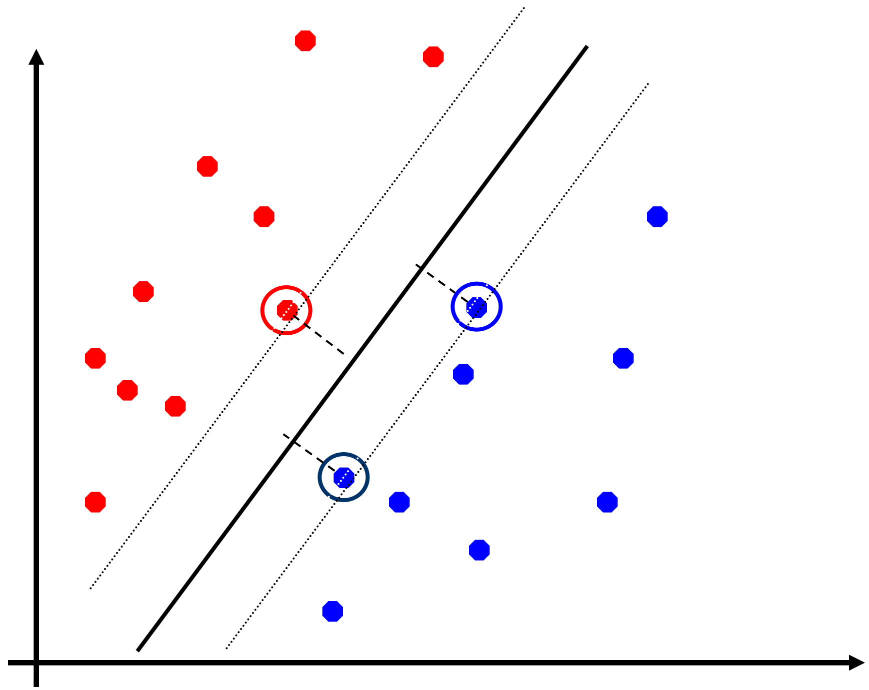
Solution: Support Vectors

- Classification function:

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

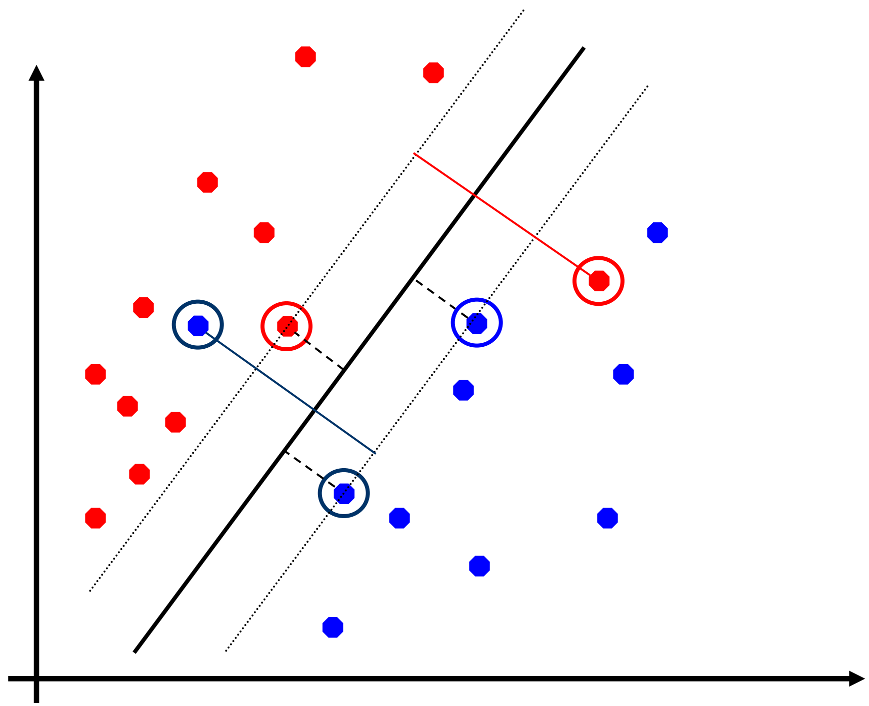
Linear SVM

- Only support vectors matter; other training examples are ignorable.



Soft Margin Classification

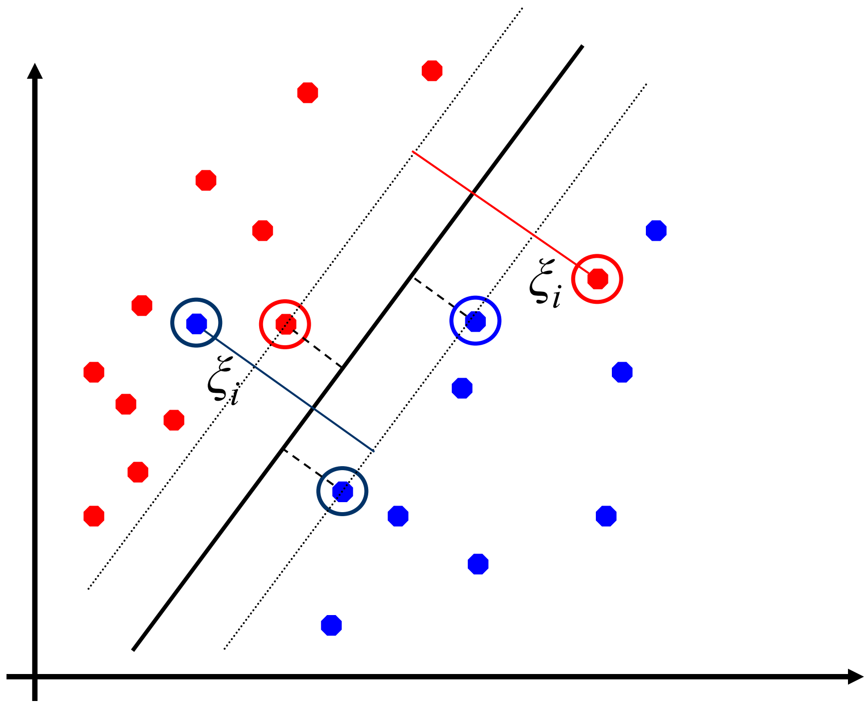
- What if the training set is mostly, but not exactly, linearly separable?



*The (hard) linear SVM problem is **infeasible** here.*

Soft Margin Classification

- ***Slack variables*** ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification Mathematically

- The old formulation (hard SVM):

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..L$: $\quad y_i (\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1$

- Modified formulation incorporates slack variables (soft SVM):

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..L$: $\quad y_i (\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq 1 - \xi_{i,}\ , \quad \xi_i \geq 0$

- **Parameter *C*** can be viewed as a way to control overfitting: it “trades off” the relative importance of maximizing the margin and fitting the training data.

Soft Margin Classification – Solution

- Dual problem is identical to separable case:

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

- Again, $\mathbf{x}_i$ with non-zero α_i will be support vectors.
- Solution to the primal problem is:

$$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i$$
$$b = y_k(1 - \xi_k) - \Sigma\alpha_i y_i \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_k \quad \text{for any } k \text{ s.t. } \alpha_k > 0$$

- Again, we don't need to compute **w** explicitly for classification:

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

Linear SVMs: Review

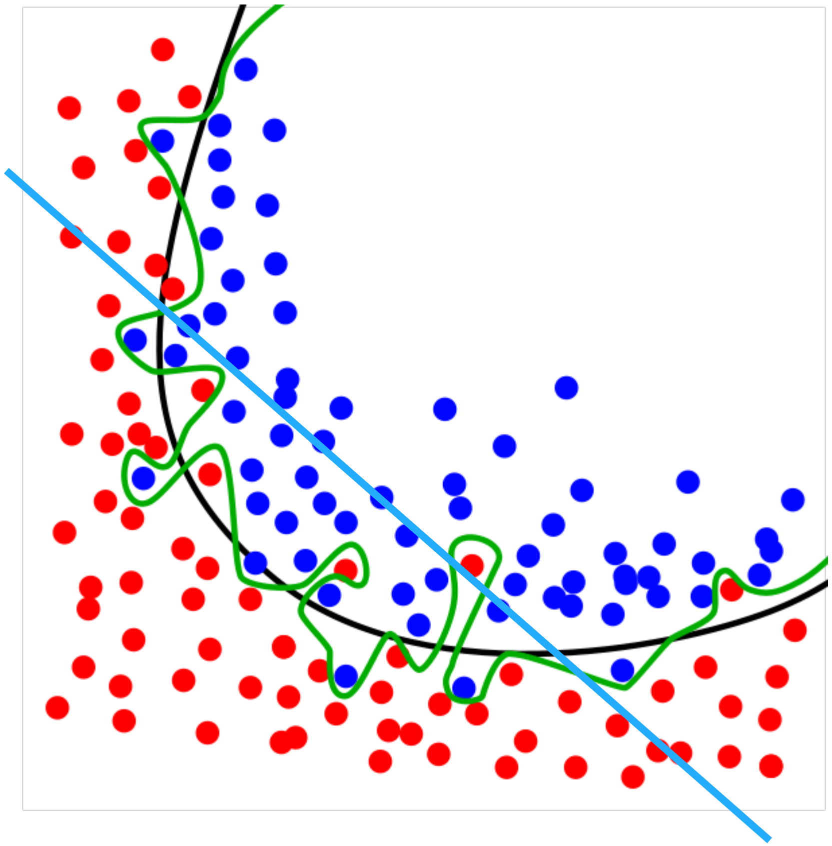
- The classifier is a *separating hyperplane*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Model complexity depends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

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

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

THE UNIVERSITY OF MELBOURNE

Overfitting - Underfitting

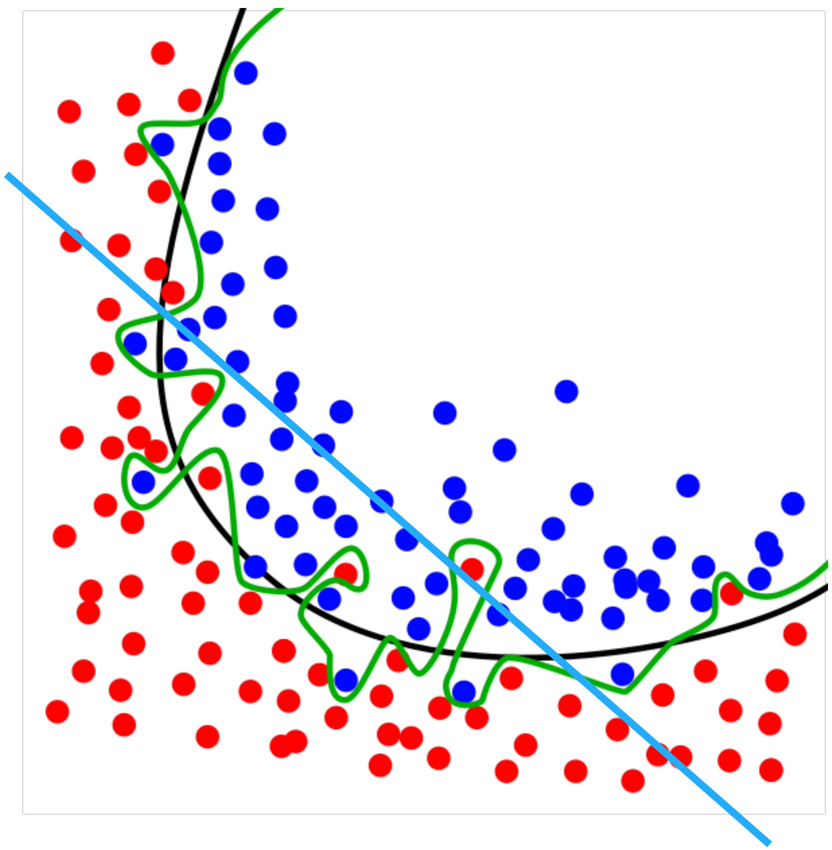


Underfitting: model not expressive enough to capture patterns in the data

Overfitting: model too complicated; capture noise in the data

Just right: model captures essential patterns in the data

Non-Linear SVM Motivation



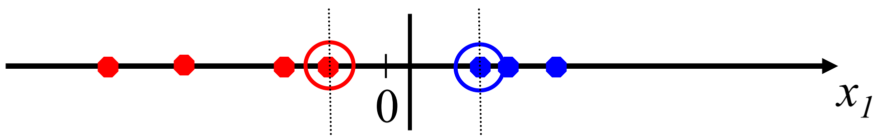
Linear model underfitting: model not expressive enough to capture patterns in the data

Soft-Margin (linear) SVM can cater for a small number of training errors

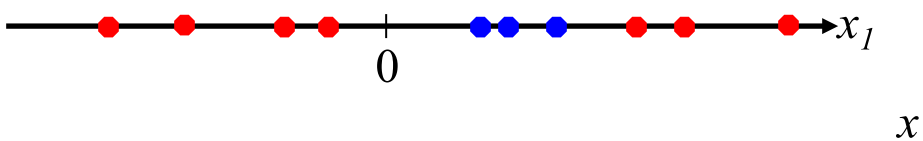
But is still a linear model

Non-Linear SVM Motivation–

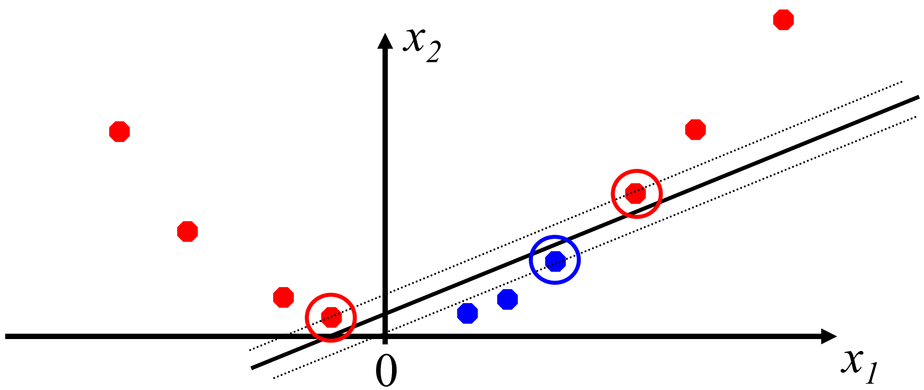
- Datasets that are linearly separable with some noise work out great:



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

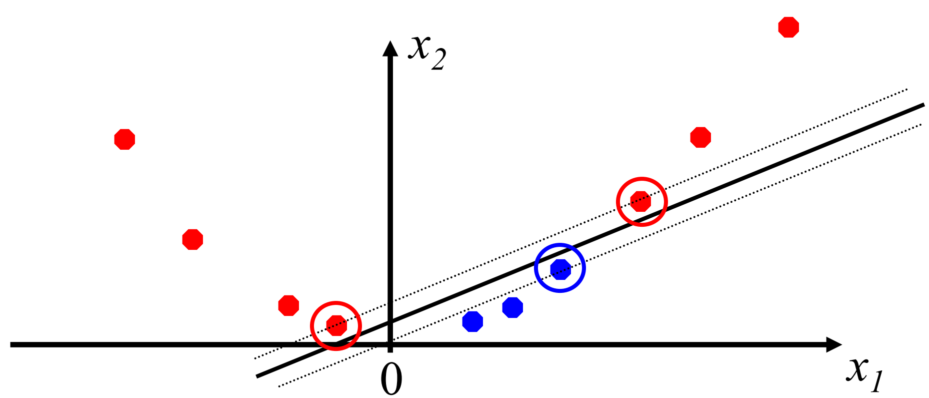


- How about… mapping data to a higher-dimensional space:



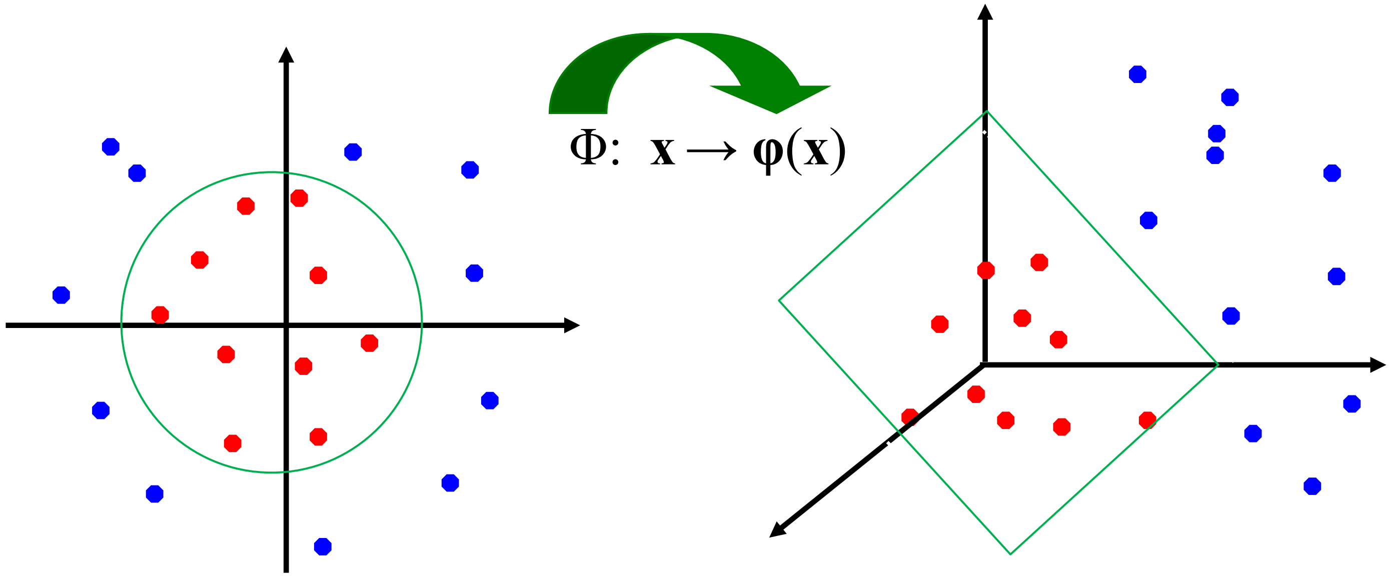
Non-linear SVMs Overview

- Turn linear SVM into a non-linear model
- By mapping the original data into a high dimensional space where the data is hopefully linearly separable



Non-linear SVMs Overview

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



- Higher-dimensional space still has *intrinsic* dimensionality *d*, but linear separators in it correspond to *non-linear* separators in original space.

Turning Linear SVM into Non-linear SVM

Find $\alpha_1 \ldots \alpha_L$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - ½\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

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

- The linear SVM classifier relies on inner product between vectors $\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$

(pair-wise dot products between all data points)

- If every data point is mapped into high-dimensional space via some transformation Φ : $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$\boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$$

Turning Linear SVM into Non-linear SVM

- Explicit mapping & Plug

$$\boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j)$$

In place of

$$\mathbf{x}_i^T\mathbf{x}_j$$

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

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)} + b$$

What if we can by-pass this explicit mapping step?

The "Kernel Trick": The Dot Product

- SVM does not need direct access to the original feature space, i.e., original data representation **x**
- It only requires access to the dot products $\mathbf{x}_i^T\mathbf{x}_j$
- The inner products

$$K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$$

$$K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i)^T\,\varphi(\mathbf{x}_j)$$

Can be regarded as a measure of similarity between data points (think cosine similarity)

Find $\alpha_1 \ldots \alpha_L$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\sigma\alpha_i\alpha_j y_i y_j \boxed{K(\mathbf{x}_i, \mathbf{x}_j)}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{K(\mathbf{x}_i, \mathbf{x})} + b$$

THE UNIVERSITY OF MELBOURNE

The “Kernel Trick”: Implicit Mapping

- What if we have a function that compute the inner product $K(\mathbf{x}_i,\mathbf{x}_j)$ directly without explicitly performing the mapping Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$

Find $\alpha_1 \ldots \alpha_L$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\sigma\alpha_i\alpha_j y_i y_j \boxed{K(\mathbf{x}_i, \mathbf{x}_j)}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{K(\mathbf{x}_i, \mathbf{x})} + b$$

Kernel Functions

- A *kernel function* is a function that is equivalent to an inner product in some feature space.
- Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).
- Why implicit mapping?
  - Save computational cost
  - The target space can have very high dimensionality

Kernel Example

- 2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$
- Let: $K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^\mathrm{T}\mathbf{x}_j)^2$
- What mapping is this?
- Need to show that $K(\mathbf{x}_i,\mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i)^\mathrm{T}\boldsymbol{\varphi}(\mathbf{x}_j)$ for some $\boldsymbol{\varphi}$

$$
\begin{aligned}
K(\mathbf{x}_i,\mathbf{x}_j) &= (1 + \mathbf{x}_i^\mathrm{T}\mathbf{x}_j)^2 = 1 + x_{i1}^2 x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} \\
&= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^\mathrm{T} [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}] \\
&= \boldsymbol{\varphi}(\mathbf{x}_i)^\mathrm{T}\boldsymbol{\varphi}(\mathbf{x}_j),
\end{aligned}
$$

where $\boldsymbol{\varphi}(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

Kernel Functions

- Not all 'similarity' measures are proper kernels

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2$$

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^3 \; ???$$

- For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^{\mathbf{T}}\varphi(\mathbf{x}_j)$ can be cumbersome.

What Functions are Kernels?

- Mercer's theorem:

  Every positive semi-definite symmetric function is a kernel

- Positive semi-definite symmetric functions correspond to a positive semi-definite symmetric Gram matrix:

K=

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	$\ldots$	$K(\mathbf{x}_1,\mathbf{x}_n)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_n)$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$K(\mathbf{x}_n,\mathbf{x}_1)$	$K(\mathbf{x}_n,\mathbf{x}_2)$	$K(\mathbf{x}_n,\mathbf{x}_3)$	$\ldots$	$K(\mathbf{x}_n,\mathbf{x}_n)$

Non examinable

Examples of Kernel Functions

- Linear:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$$

  – Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

- Polynomial of power *p*:

$$K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^p$$

  – Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

Examples of Kernel Functions

- Gaussian (Radial-Basis Function (RBF)):

$$K(\mathbf{x}_i,\mathbf{x}_j) = e^{\frac{-\left\|x_i-x_j\right\|^2}{2\sigma^2}}$$

  – Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian)

Non-linear SVMs Mathematically

- Dual problem formulation:

Find $\alpha_1 \ldots \alpha_L$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $C \geq \alpha_i \geq 0$ for all α_i

- The classifier function is:

$$f(\mathbf{x}) = \Sigma\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- Optimization techniques for finding α_i's remain the same!

In Practice

- Are we guaranteed that the kernel trick will make the data linearly separable?
  - No
  - But usually work

- How to find the suitable kernel function and its parameters?
  - Method: Using M-fold cross-validation error rate

Multi-class Extension

- SVM is inherently a binary classifier
- Extension to multiclass:
  - One-versus-all: build M classifiers for M classes. Choose class with largest margin for test data
  - One-versus-one: one classifier per pair of classes (M(M-1)/2 classifiers in total), choose class selected by most classifiers

SVM Applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* ’97], principal component analysis [Schölkopf *et al.* ’99], etc.
- Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of α_i’s at a time, e.g. SMO [Platt ’99] and [Joachims ’99]
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.

References

- [1]https://static1.squarespace.com/static/58851af9ebbd1a30e98fb283/t/58902fbae4fcb5398aeb7505/1485844411772/SVM+Explained.pdf
- [2] A Tutorial on Support Vector Machines for Pattern Recognition
- [3] Demo: http://cs.stanford.edu/people/karpathy/svmjs/demo/
  – (Note: C is the inverse penalty in this app)
- [4] Demo: http://www.cmsoft.com.br/download/SVMDemo.zip

Summary

- What is the intuition of Support Vector Machines (SVMs)?
- How to formulate and solve SVM?
- What is linear and non-linear SVM?