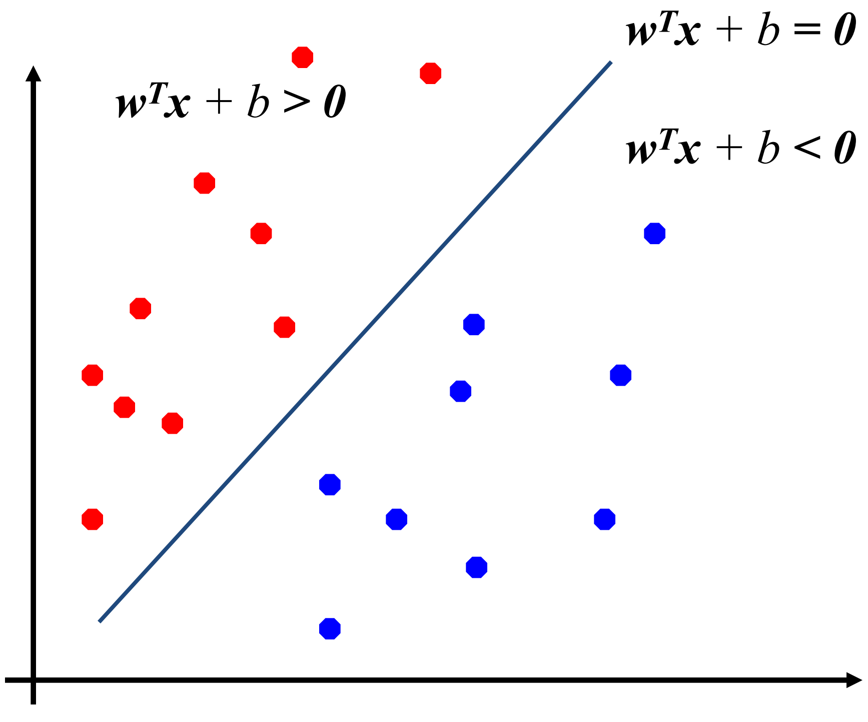
COMP3055
Machine Learning

Topic 11 – SVM

Dr. Zheng LU
2018 Autumn

Perceptron Revisited: Linear Separators

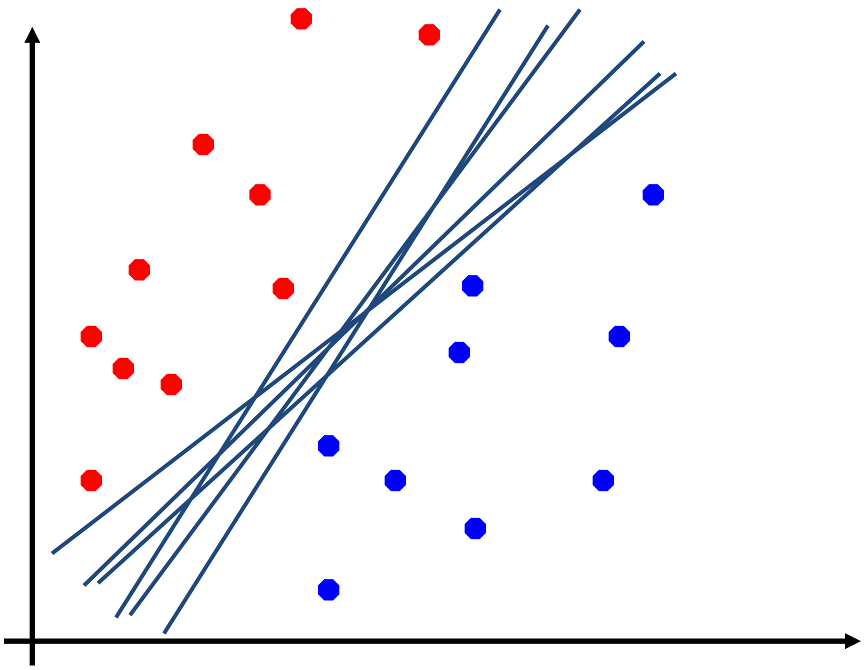
Binary classification can be viewed as the task of separating two classes in feature space:



$$f(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$$

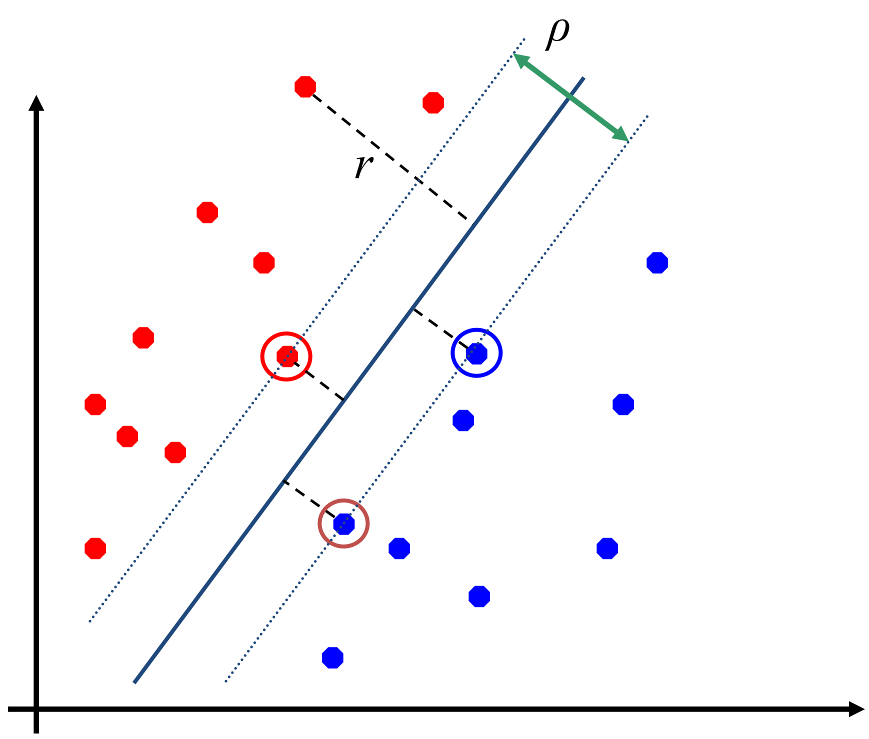
Linear Seperator

Which of the linear separators is optimal?



Classification Margin

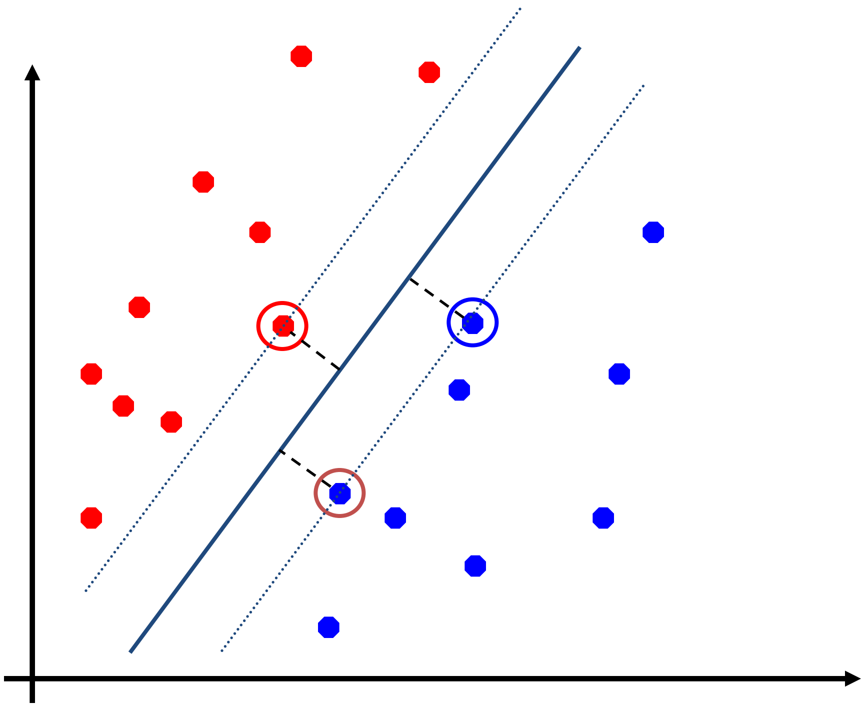
- Distance from example $\boldsymbol{x}_i$ to the separator is $r = \frac{\boldsymbol{w}^T\boldsymbol{x}_i + b}{\|\boldsymbol{w}\|}$.
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



Maximum Margin Classification

- Maximizing the margin is good according to intuition.

- Implying that only support vectors matter; other training examples are ignorable.



Linear SVM Mathematically

- Let training set $\{(x_i, y_i)\}_{i=1..w}, x_i \in R^d, y_i \in \{-1,1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\boldsymbol{x}_i, y_i)$:

$$\begin{aligned} \boldsymbol{w}^T\boldsymbol{x}_i + b \leq -\rho/2 \quad & if\ y_i = -1 \\ \boldsymbol{w}^T\boldsymbol{x}_i + b \geq \rho/2 \quad & if\ y_i = 1 \end{aligned} \qquad \Leftrightarrow \qquad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq \rho/2$$

- For every support vector $\boldsymbol{x}_s$ the above inequality is an equality. After rescaling $\boldsymbol{w}$ and b by $\rho/2$ in the equality, we obtain that distance between each $\boldsymbol{x}_s$ and the hyperplane is

$$r = \frac{y_s(\boldsymbol{w}^T\boldsymbol{x}_s + b)}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$$

- Then the margin can be expressed through (rescaled) $\boldsymbol{w}$ and b as:

$$\rho = 2r = \frac{2}{\|\boldsymbol{w}\|}$$

Linear SVM Mathematically

- Then we can formulate the quadratic optimization problem:

$$\boxed{\begin{array}{l}\text{Find } \boldsymbol{w} \text{ and } b \text{ such that} \\ \rho = \dfrac{2}{\|\boldsymbol{w}\|} \text{ is maximized,} \\ \text{and for all } (\boldsymbol{x_i}, y_i), i = 1..n\text{: } y_i(\boldsymbol{w^T x_i} + b) \geq 1\end{array}}$$

Which can be reformulated as:

$$\boxed{\begin{array}{l}\text{Find } \boldsymbol{w} \text{ and } b \text{ such that} \\ \boldsymbol{\Phi}(\boldsymbol{w}) = \|\boldsymbol{w}\|^2 = \boldsymbol{w^T w} \text{ is minimized,} \\ \text{and for all } (\boldsymbol{x_i}, y_i), i = 1..n\text{: } y_i(\boldsymbol{w^T x_i} + b) \geq 1\end{array}}$$

Solving the Optimization Problem

Find $\boldsymbol{w}$ and b such that
$\boldsymbol{\Phi}(\boldsymbol{w}) = \|\boldsymbol{w}\|^2 = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$ is minimized,
and for all $(\boldsymbol{x_i}, y_i), i = 1..n$: $y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x_i} + b) \geq 1$

- Need to optimize a ***quadratic*** **function subject to** ***linear*** **constraints**.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.
- The solution involves constructing a ***dual problem*** where a ***Lagrange multiplier*** α_i is associated with every inequality constraint in the primal (original) problem:

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

The Optimization Problem Solution

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

$$\boxed{\boldsymbol{w} = \sum \alpha_i y_i \boldsymbol{x_i}\ , \quad b = y_k - \sum \alpha_i y_i \boldsymbol{x_i^T x_k}, \ \text{for any } \alpha_k > 0}$$

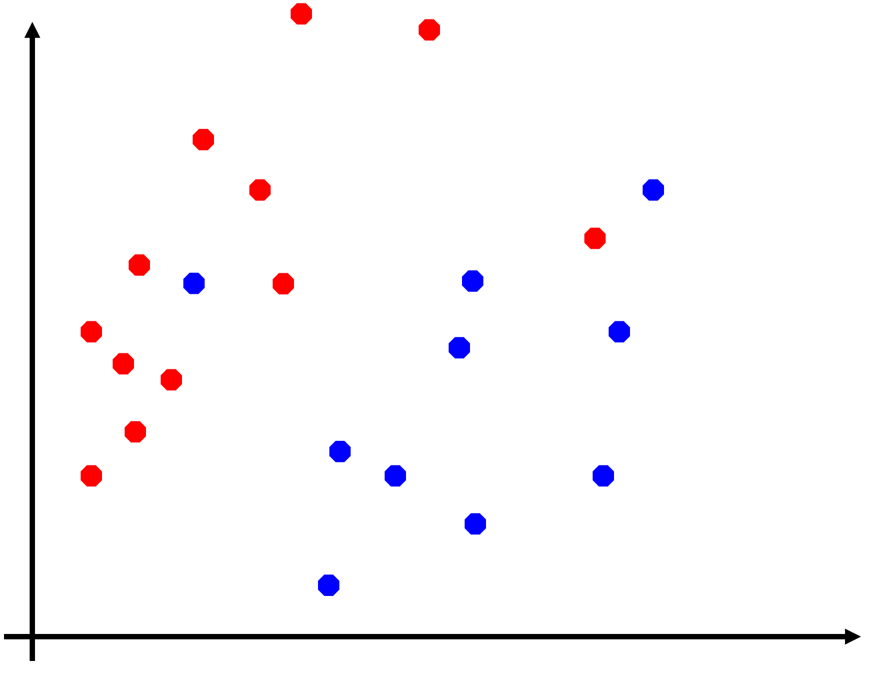
- Each non-zero α_i indicates that corresponding $\boldsymbol{x_i}$ is a support vector.
- Then the classifying function is (note that we don't need $\boldsymbol{w}$ explicitly):

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

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x}_i$ – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\boldsymbol{x}_i^T\boldsymbol{x}_j$ between all training points.

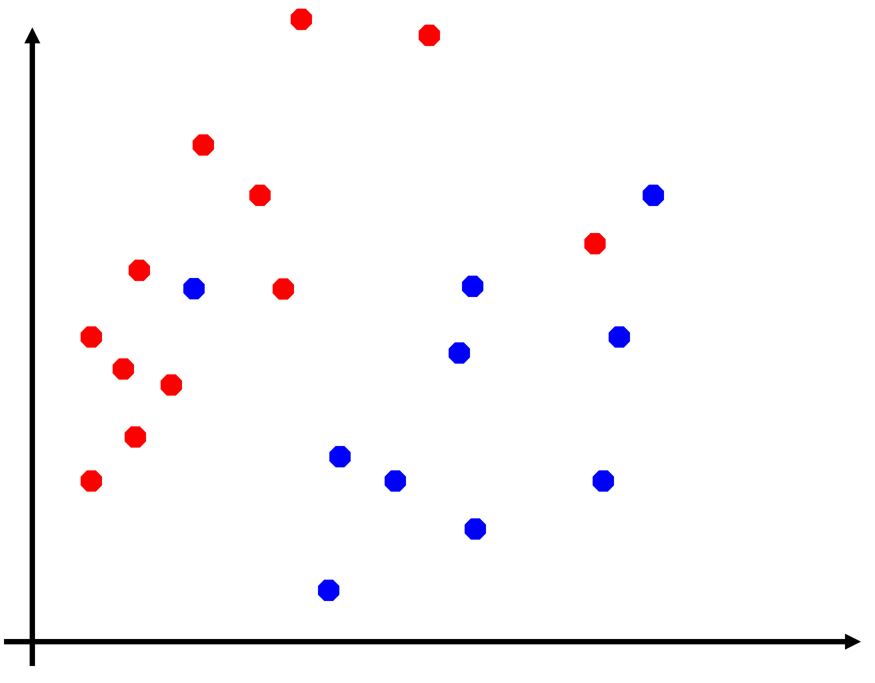
Soft Margin Classification

- What if the training set is not linearly separable?



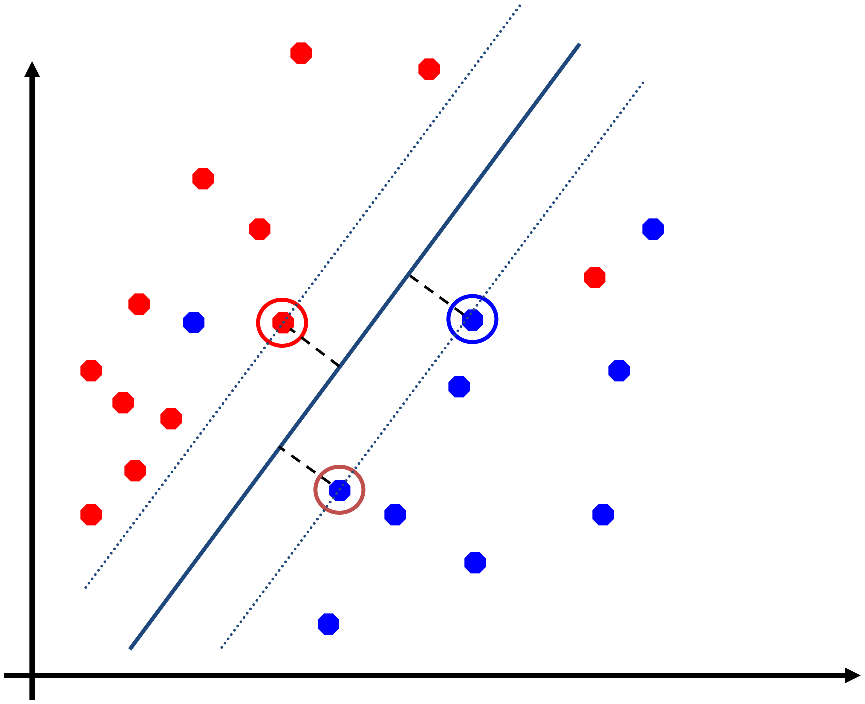
Soft Margin Classification

- What if the training set is not linearly separable?
- ***Slack variables*** ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called ***soft***.



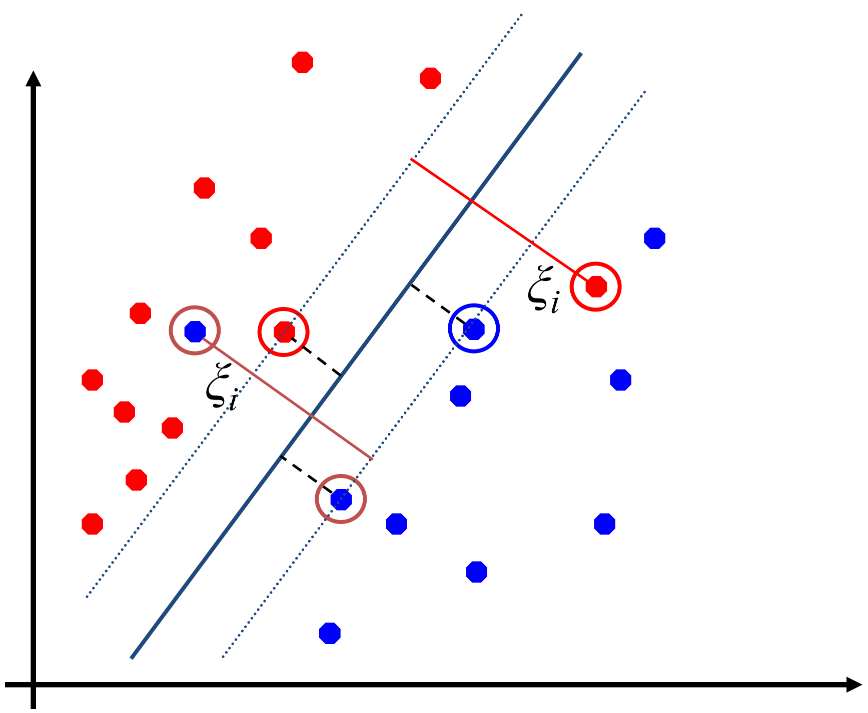
Soft Margin Classification

- What if the training set is not linearly separable?
- ***Slack variables*** ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called ***soft***.



Soft Margin Classification

- What if the training set is not linearly separable?
- ***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:

Find $\boldsymbol{w}$ and b such that
$\boldsymbol{\Phi}(\boldsymbol{w}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}$ is minimized,
and for all $(\boldsymbol{x_i}, y_i), i = 1..n$: $\quad y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x_i} + b) \geq 1$

- Modified formulation incorporates slack variables:

Find $\boldsymbol{w}$ and b such that
$\boldsymbol{\Phi}(\boldsymbol{w}) = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} + C\sum \xi_i$ is minimized,
and for all $(\boldsymbol{x_i}, y_i), i = 1..n$: $\quad y_i\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x_i} + b\right) \geq 1 - \xi_i, \quad \xi_i > 0$

- 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:

Find $\alpha_1 \dots \alpha_n$ such that
$\boldsymbol{Q}(\alpha) = \sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i\alpha_j y_i y_j \boldsymbol{x_i^T x_j}$ is maximized and
(1)$\sum \alpha_i y_i = 0$
(2)$0 \leq \alpha_i \leq C$ for all α_i

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

$$\boldsymbol{w} = \sum \alpha_i y_i \boldsymbol{x_i}\,, \quad b = y_k(1-\xi_k) - \sum \alpha_i y_i \boldsymbol{x_i^T x_k}, \quad \text{for any } \alpha_k > 0$$

- Again, we do not need to compute $\boldsymbol{w}$ explicitly for classification:

$$f(x) = \sum \alpha_i y_i \boldsymbol{x_i^T x} + b$$

Linear SVM - Overview

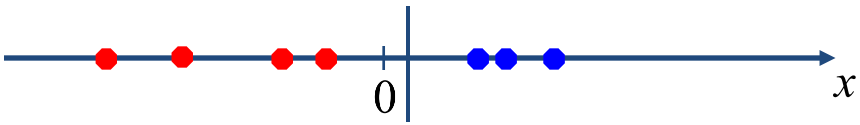
- The classifier is a ***separating hyperplane***.
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\boldsymbol{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the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

Find $\alpha_1 \dots \alpha_n$ such that
$\boldsymbol{Q}(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \boldsymbol{x_i^T x_j}$ is maximized and
(1) $\sum \alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

$$f(x) = \sum \alpha_i y_i \boldsymbol{x_i^T x} + b$$

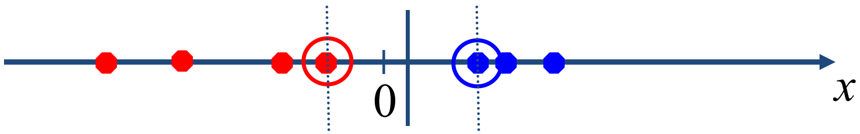
Non-Linear SVM

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



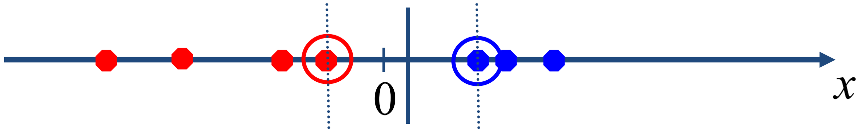
Non-Linear SVM

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

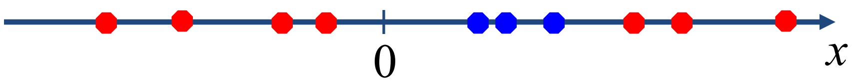


Non-Linear SVM

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

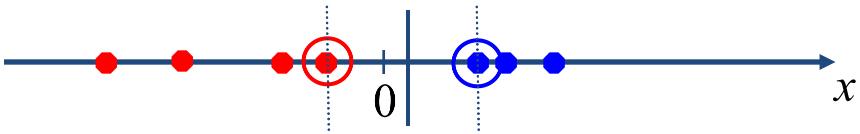


- But what are we going to do if the dataset is just too hard?

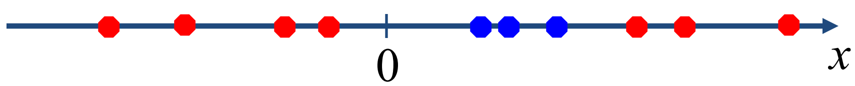


Non-Linear SVM

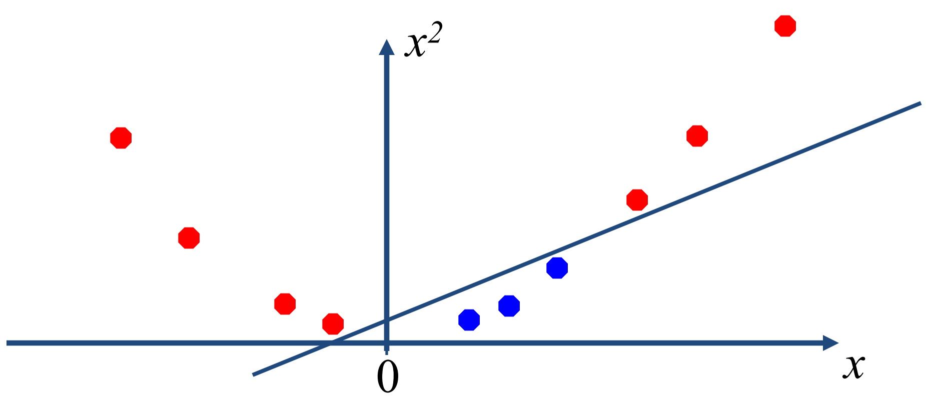
- Datasets that are linearly separable with some noise work out great:



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

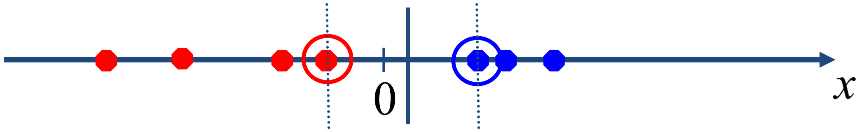


- How about… mapping data to a higher-dimensional space:

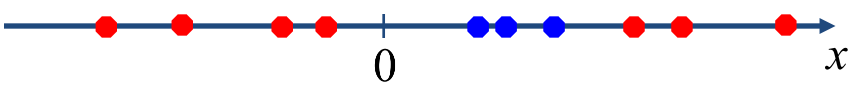


Non-Linear SVM

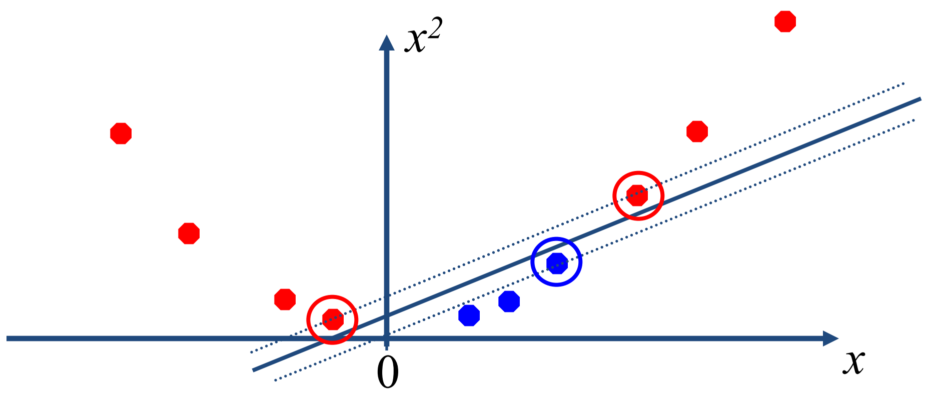
- Datasets that are linearly separable with some noise work out great:



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

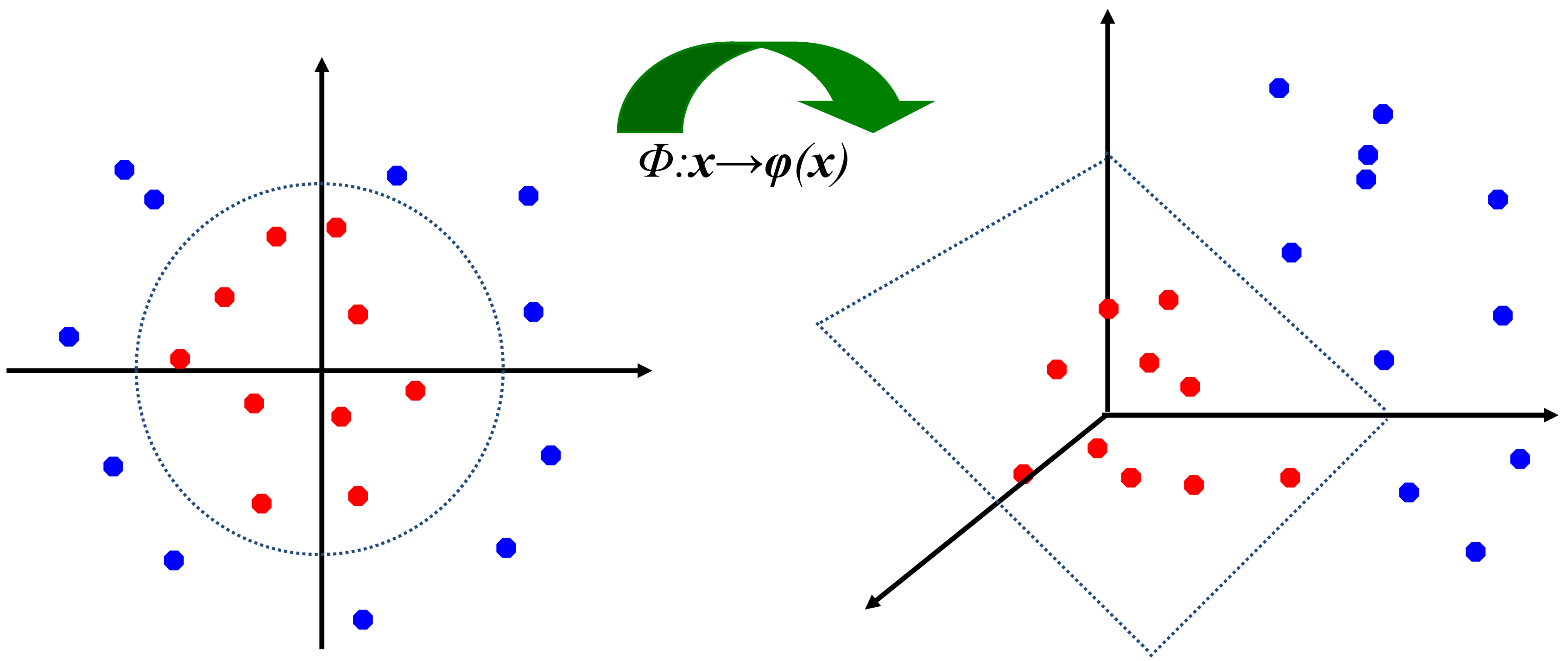


- How about… mapping data to a higher-dimensional space:



Non-Linear SVM: 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 inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

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

- A ***kernel function*** is a function that is equivalent to an inner product in some feature space.
- Example: 2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,
we need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x}_j)$:

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

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

- Thus, a kernel function ***implicitly*** maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

Kernel Functions

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

Kernel Functions

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

- Mercer's theorem
  - ***Every semi-positive definite symmetric function is a kernel.***

Examples of Kernel Function

- Linear: $K(\boldsymbol{x}_i,\boldsymbol{x}_j)= \boldsymbol{x}_i^{\boldsymbol{T}}\boldsymbol{x}_j$
  - Mapping Φ*:* $\boldsymbol{x}\rightarrow\varphi(\boldsymbol{x})$, where $\varphi(\boldsymbol{x})$ is $\boldsymbol{x}$ itself.

- Polynomial of power p: $K(\boldsymbol{x}_i,\boldsymbol{x}_j)= (1+ \boldsymbol{x}_i^{\boldsymbol{T}}\boldsymbol{x}_j)^p$
  - Mapping Φ*:* $\boldsymbol{x}\rightarrow\varphi(\boldsymbol{x})$, where $\varphi(\boldsymbol{x})$ has $\binom{d+p}{p}$ dimensions

- Gaussian (radial-basis function): $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ = $e^{-\frac{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2}{2\sigma^2}}$
  - Mapping Φ*:* $\boldsymbol{x}\rightarrow\varphi(\boldsymbol{x})$, where $\varphi(\boldsymbol{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

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

Non-Linear SVM Mathematically

- Dual problem formulation:

Find $\alpha_1 \dots \alpha_n$ such that
$\boldsymbol{Q}(\alpha) = \sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i \alpha_j y_i y_j K(\boldsymbol{x_i}, \boldsymbol{x_j})$ is maximized and
(1) $\sum \alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- The solution is:

$$f(x) = \sum \alpha_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b$$

- Optimization for finding α_i remains the same!

SVM and Kernel Methods

- Are explicitly based on a theoretical model of learning.
- Come with theoretical guarantees about their performance.
- Have a modular design that allows one to separately implement and design their components.
- Are not affected by local minima.
- Do not suffer from the curse of dimensionality.

SVM Software and Resourses

- http://www.svms.org/tutorials/
- http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - LIBSVM -- A Library for Support Vector Machines by Chih-Chung Chang and Chih-Jen Lin