

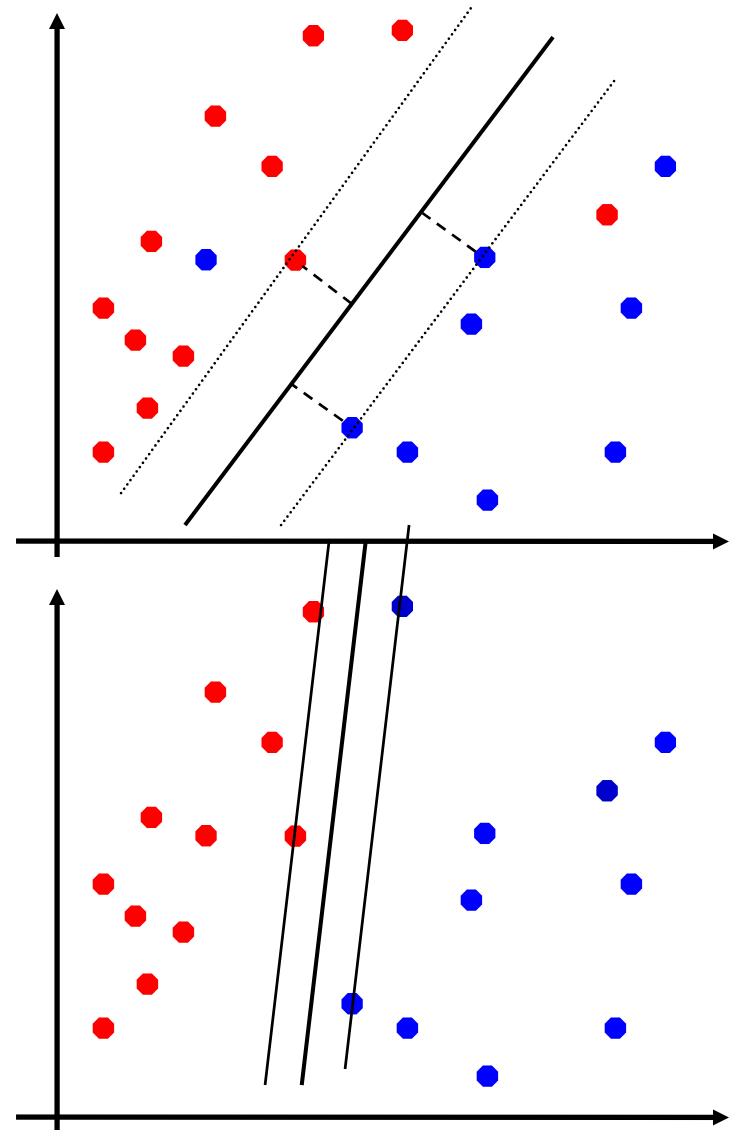
Support Vector Machine (cont.)

Summarization So Far

- We demonstrated that we prefer to have linear classifiers with large margin.
- We formulated the problem of finding the maximum margin linear classifier as a quadratic optimization problem
- This problem can be solved by solving its dual problem, and efficient QP algorithms are available.
- Problem solved?

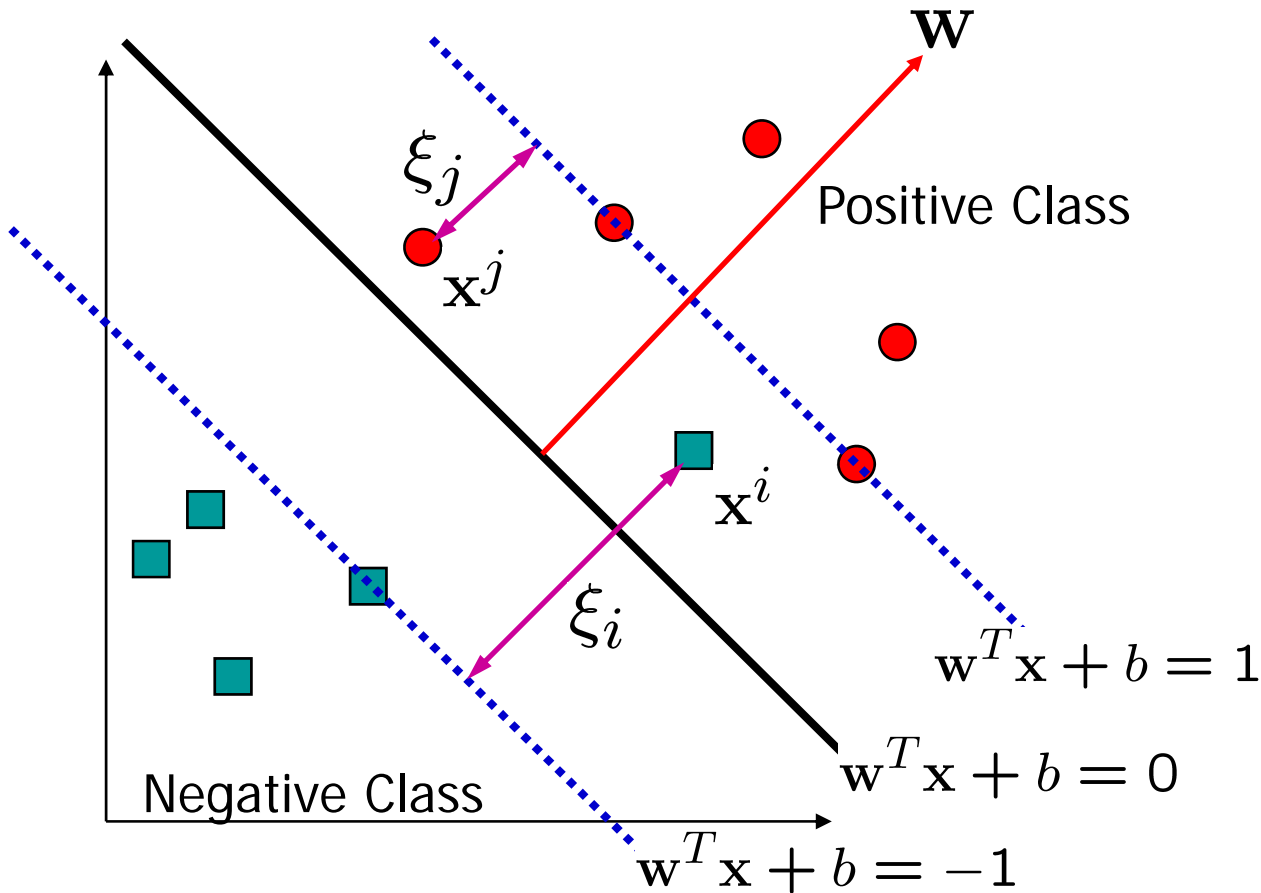
Non-separable Data and Noise

- What if the data is not linearly separable?
- Even when linearly separable, we may have noise in data, and maximum margin classifier is not robust to noise!



Soft Margin

- Allow functional margins to be less than 1
 - But will charge a penalty



Soft-Margin Maximization

Hard margin

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

subject to: $y^i (\mathbf{w} \cdot \mathbf{x}^i + b) \geq 1, \quad i = 1, \dots, N$



Soft margin

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{i=1}^N \xi_i$$

subject to: $y^i (\mathbf{w} \cdot \mathbf{x}^i + b) \geq 1 - \xi_i, \quad i = 1, \dots, N$
 $\xi_i \geq 0, \quad i = 1, \dots, N$

- Introduce **slack variables** ξ_i to allow functional margins to be smaller than 1
- Parameter c controls the tradeoff between maximizing the margin and fitting the training example

Dual Formulation of Soft Margin

$$\max \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y^i y^j \langle \mathbf{x}^i \cdot \mathbf{x}^j \rangle$$

$$\text{Subject to: } \sum_{i=1}^N \alpha_i y^i = 0$$

$$0 \leq \alpha_i \leq c \quad i=1, \dots, N$$

- The dual problem is almost identical to the separable case, except for that α_i 's are now bounded by c
- c controls the tradeoff between maximizing margin and fitting training data
- It's effect is to put a **box constraint** on α , the weights of the support vectors
- It limits the influence of outliers

Dual Formulation of Soft Margin

$$\max \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y^i y^j \langle \mathbf{x}^i \cdot \mathbf{x}^j \rangle$$

Subject to: $\sum_{i=1}^N \alpha_i y^i = 0$

$$0 \leq \alpha_i \leq c \quad i=1, \dots, N$$

- We now have also have support vectors for data that have functional margin less than one (in addition to those that equal 1), but there α_i 's will only equal c

support vectors ($\alpha_i > 0$)

$$c > \alpha_i > 0: \quad y^i(w \cdot x^i + b) = 1, \text{ i.e., } \xi_i = 0$$

$$\alpha_i = c: \quad y^i(w \cdot x^i + b) \leq 1, \text{ i.e., } \xi_i \geq 0$$

The optimal \mathbf{w} can then be computed:

$$\mathbf{w} = \sum \alpha_i y^i \mathbf{x}^i$$

Linear SVMs: Overview

- So far our 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 Lagrange multipliers α_i .
- For both training and classification, we see training data appear only inside inner products:

Find $\alpha_1 \dots \alpha_N$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y^i y^j \langle \mathbf{x}^i \cdot \mathbf{x}^j \rangle$ is maximized
and

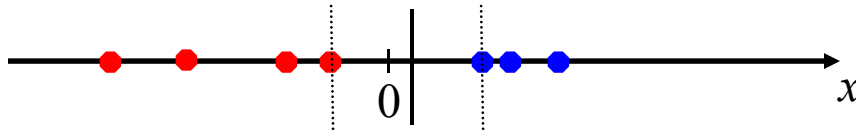
(1) $\sum \alpha_i y^i = 0$

(2) $0 \leq \alpha_i \leq c$ for all α_i

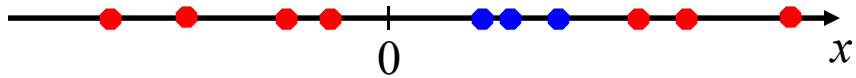
$$f(\mathbf{x}) = \sum \alpha_i y^i \langle \mathbf{x}^i \cdot \mathbf{x} \rangle + b$$

Non-linear SVMs

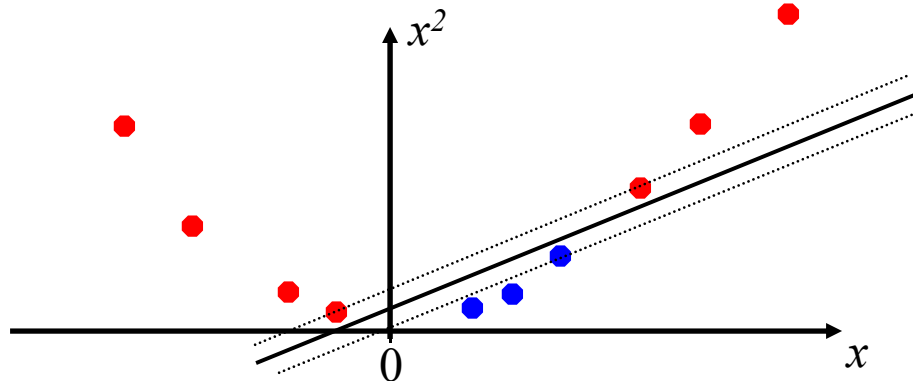
- Datasets that are linearly separable with some noise work out great:



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

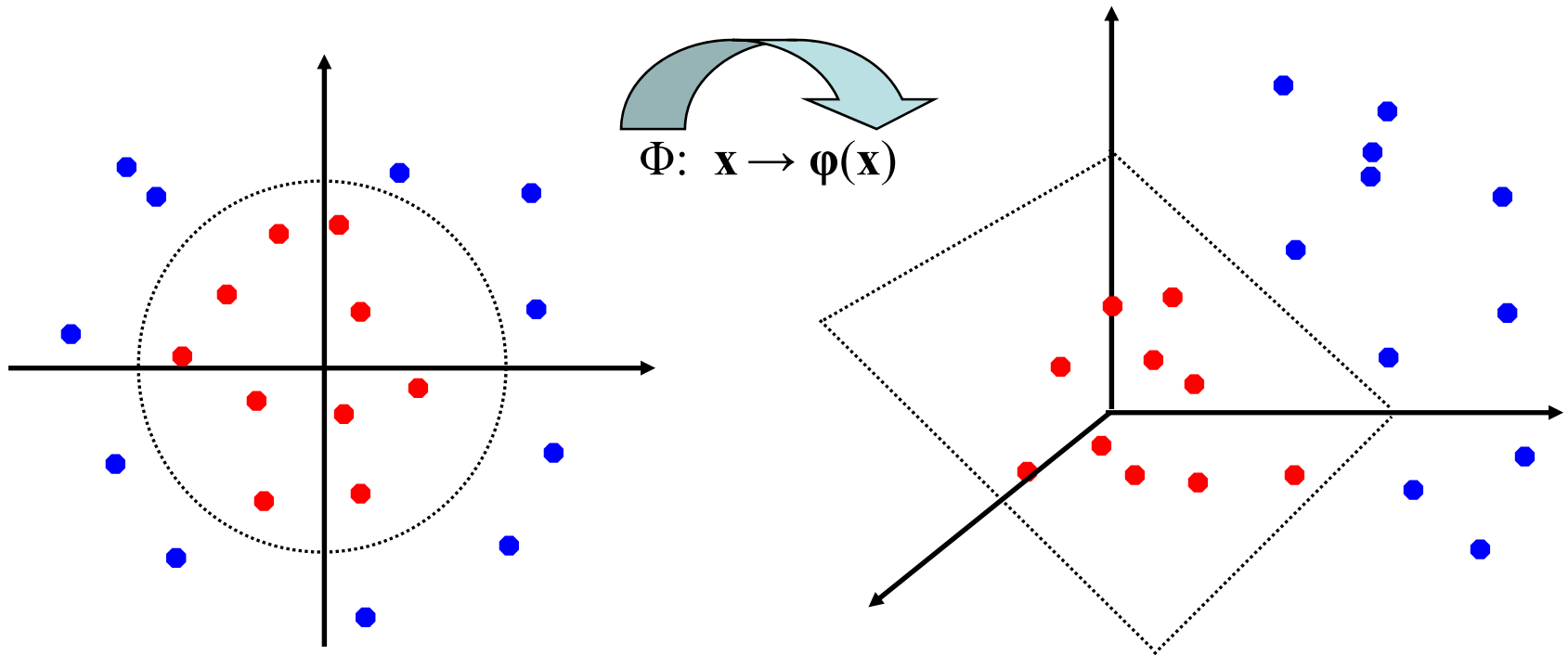


- How about... mapping data to a higher-dimensional space:



Non-linear SVMs: Feature Spaces

- General idea: For any data set, the original input space can always be mapped to some higher-dimensional feature space such that the data is linearly separable:



Example: Quadratic Space

- Assume m input dimensions

$$\mathbf{x} = (x_1, x_2, \dots, x_m)$$

- Number of quadratic terms:

$$(m+2)\text{-choose-2} = \frac{(m+2)(m+1)}{2}$$

- The number of dimensions increase rapidly - expensive to compute!

You may be wondering about the $\sqrt{2}$'s
You will find out why they are there soon!

$$\Phi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

Diagram illustrating the structure of the quadratic space $\Phi(\mathbf{x})$. The terms are grouped into four categories:

- Constant Term (1)
- Linear Terms ($\sqrt{2}x_1, \sqrt{2}x_2, \dots, \sqrt{2}x_m$)
- Pure Quadratic Terms ($x_1^2, x_2^2, \dots, x_m^2$)
- Quadratic Cross-Terms ($\sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \dots, \sqrt{2}x_1x_m, \sqrt{2}x_2x_3, \dots, \sqrt{2}x_{m-1}x_m$)

An arrow points from the text box to the $\sqrt{2}x_1x_2$ term.

Kernel Function

- The linear classifier relies on inner product between vectors $K(\mathbf{x}^i, \mathbf{x}^j) = \langle \mathbf{x}^i \cdot \mathbf{x}^j \rangle$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}^i) \cdot \phi(\mathbf{x}^j) \rangle$$

- A **kernel function** is a function that is equivalent to an inner product in some feature space.
- Example: we can define a kernel as

$$K(\mathbf{x}^i, \mathbf{x}^j) = (\mathbf{x}^i \cdot \mathbf{x}^j + 1)^2$$

This is equivalent to mapping to the quadratic space!

Example: Quadratic Kernel

Consider a 2-d input space: (generalizes to n-d)

$$\begin{aligned}K(\mathbf{x}^i, \mathbf{x}^j) &= (\mathbf{x}^i \cdot \mathbf{x}^j + 1)^2 \\&= (x_1^i x_1^j + x_2^i x_2^j + 1)^2 \\&= x_1^{i2} x_1^{j2} + 2x_1^i x_2^i x_1^j x_2^j + x_2^{i2} x_2^{j2} + 2x_1^i x_1^j + 2x_2^i x_2^j + 1 \\&= (x_1^{i2}, \sqrt{2} x_1^i x_2^i, x_2^{i2}, \sqrt{2} x_1^i, \sqrt{2} x_2^i, 1) \cdot \\&\quad (x_1^{j2}, \sqrt{2} x_1^j x_2^j, x_2^{j2}, \sqrt{2} x_1^j, \sqrt{2} x_2^j, 1) \\&= \Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{x}^j)\end{aligned}$$

nonlinear mapping of \mathbf{x}^i
and \mathbf{x}^j to quadratic space

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

Computing inner product of quadratic features is $O(m^2)$ time vs. $O(m)$ time for kernel

Non-linear SVMs

- Dual problem formulation:

Find $\alpha_1 \dots \alpha_N$ such that
 $\sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j \rangle$ is maximized and
(1) $\sum \alpha_i y^i = 0$
(2) $0 \leq \alpha_i \leq c$ for all α_i

$K(\mathbf{x}^i, \mathbf{x}^j)$

- To classify a given new data point \mathbf{x} , we compute

$$f(\mathbf{x}) = \sum \alpha_i y^i \langle \mathbf{x}^i, \mathbf{x}^j \rangle + b$$

$K(\mathbf{x}^i, \mathbf{x}^j)$

- Optimization techniques for finding α_i 's remain the same!
- This shows the utility of the dual formulation.

Kernel Functions

- In practical, the user specifies the kernel function K , without explicitly stating the transformation $\phi(\cdot)$
- Given a kernel function, finding its corresponding transformation can be very cumbersome
 - This is why people only specify the kernel function without worrying about the exact transformation
- Another view: a kernel function computes some kind of measure of similarity between objects
- If you have a reasonable measure of similarity for your application, can we use it as the kernel in an SVM?

What Functions are Kernels?

- Consider some finite set of m points, let matrix K be defined as follows:

$$K = \begin{array}{|c|c|c|c|c|} \hline K(\mathbf{x}^1, \mathbf{x}^1) & K(\mathbf{x}^1, \mathbf{x}^2) & K(\mathbf{x}^1, \mathbf{x}^3) & \dots & K(\mathbf{x}^1, \mathbf{x}^m) \\ \hline K(\mathbf{x}^2, \mathbf{x}^1) & K(\mathbf{x}^2, \mathbf{x}^2) & K(\mathbf{x}^2, \mathbf{x}^3) & & K(\mathbf{x}^2, \mathbf{x}^m) \\ \hline & & & & \\ \hline \dots & \dots & \dots & \dots & \dots \\ \hline K(\mathbf{x}^m, \mathbf{x}^1) & K(\mathbf{x}^m, \mathbf{x}^2) & K(\mathbf{x}^m, \mathbf{x}^3) & \dots & K(\mathbf{x}^m, \mathbf{x}^m) \\ \hline \end{array}$$

- This is called the **Kernel Matrix**
- Mercer's theorem:
A function K is a kernel function iff for any finite sample $\{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^m\}$, its corresponding kernel matrix is symmetric and semi-definite.

Examples of Kernel Functions

- Linear: $K(\mathbf{x}^i, \mathbf{x}^j) = \langle \mathbf{x}^i, \mathbf{x}^j \rangle$
 - Mapping $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, where $\phi(\mathbf{x})$ is \mathbf{x} itself
- Polynomial of power p : $K(\mathbf{x}^i, \mathbf{x}^j) = (1 + \mathbf{x}^i \cdot \mathbf{x}^j)^p$
 - Mapping $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, where $\phi(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions
- Gaussian (radial-basis function): $K(\mathbf{x}^i, \mathbf{x}^j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$
 - Mapping $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, where $\phi(\mathbf{x})$ 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.

Critical Steps for Using SVM

- Select the kernel function to use (important but often trickiest part of SVM)
 - In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try and usually support by off-the-shelf software
- Select the parameter of the kernel function and the value of c
 - You can use the values suggested by the SVM software
see www.kernel-machines.org/software.html for a list of available software
 - You can set apart a validation set to determine the values of the parameter

SVM Summary

- Advantages of SVMs
 - polynomial-time exact optimization rather than approximate methods
 - unlike decision trees and neural networks
 - Kernels allow very flexible hypotheses
 - Can be applied to very complex data types, e.g., graphs, sequences
- Disadvantages of SVMs
 - Must choose a good kernel and kernel parameters
 - Very large problems are computationally intractable
 - quadratic in number of examples
 - problems with more than 20k examples are very difficult to solve exactly