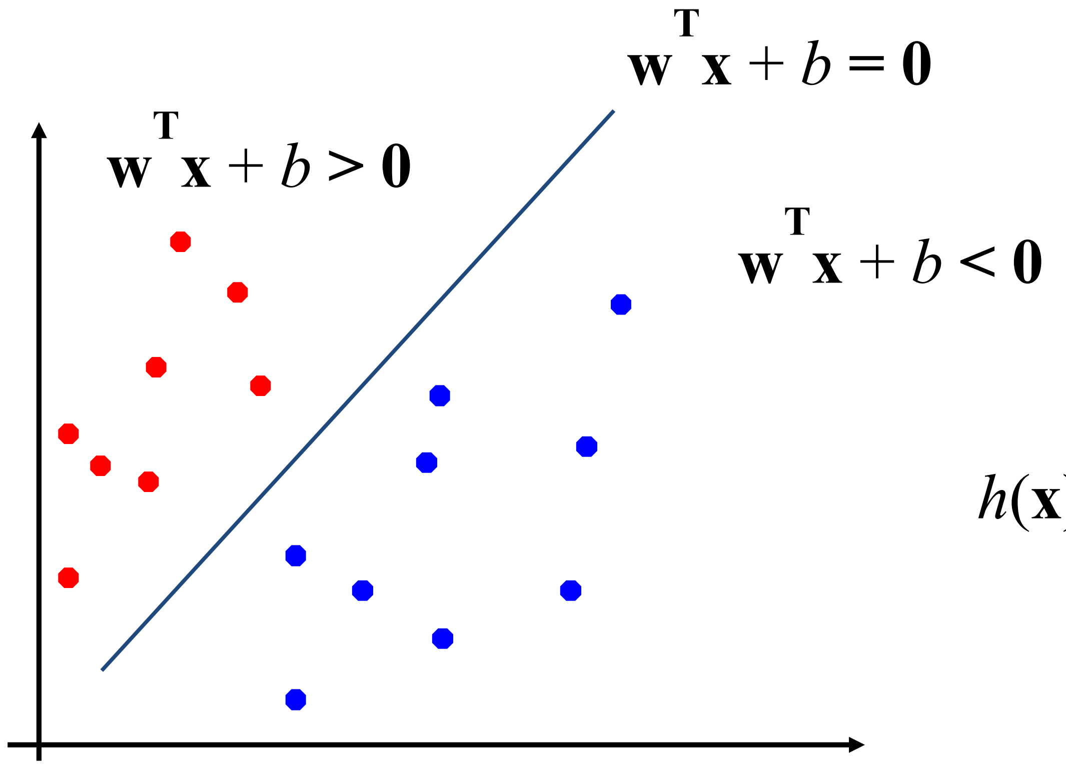
Support Vector Machines (SVM)

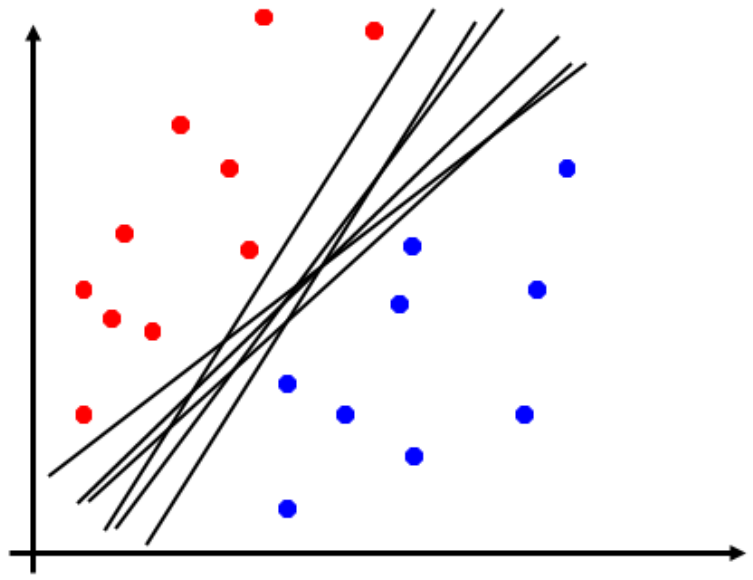
- Binary classification can be viewed as the task of separating classes in a “feature” space


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

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathbf{T}}\mathbf{x} + b)$$

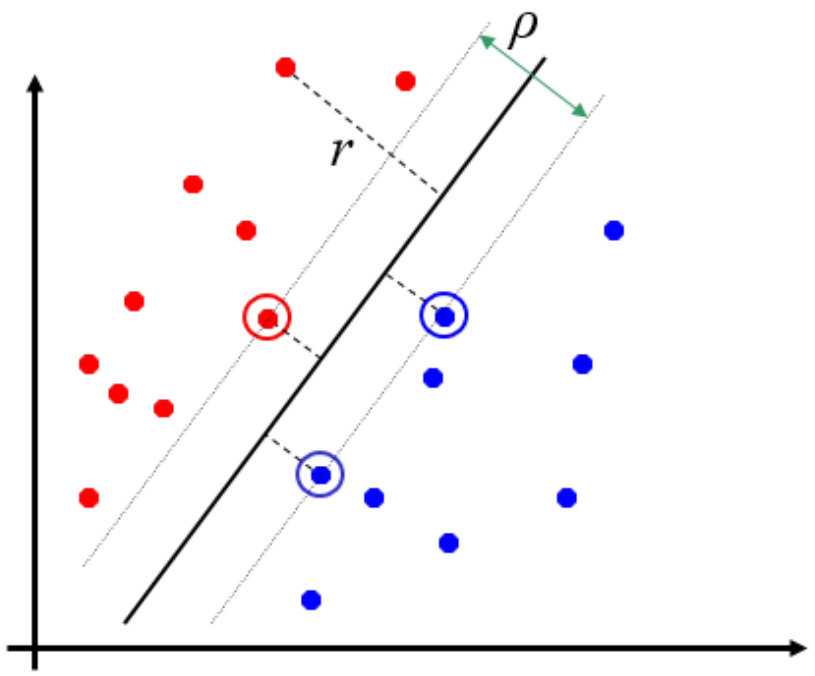
Linear Separators

- Which of the linear separators is optimal?



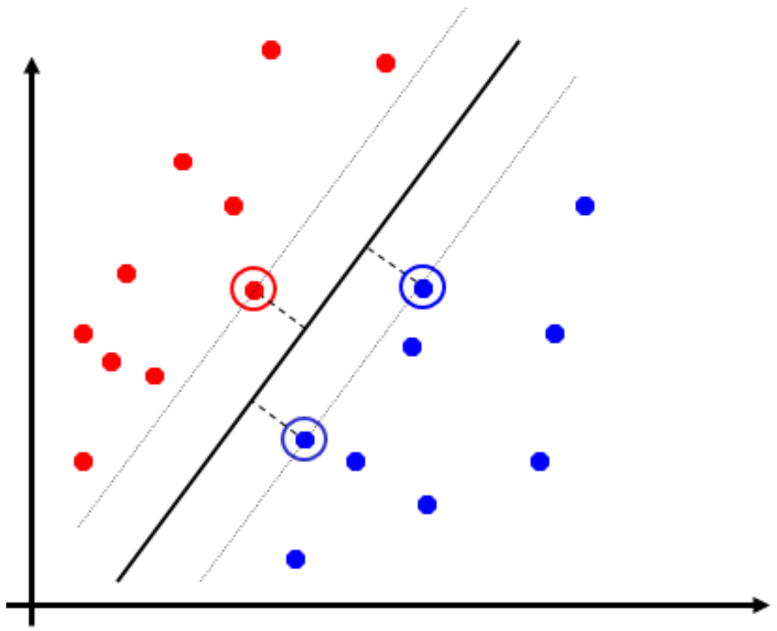
Classification Margin

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



The Maximal Margin Classifier

- Maximizing the margin is good according to intuition. In a sense, the maximal margin hyperplane represents the mid-line of the widest “slab” that we can insert between the two classes



- This implies that only the support vectors matter – other training observations are ignorable. This gives robustness of the method

Linear SVM Mathematically

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

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

- For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is $r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$

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

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

Linear SVMs Mathematically (cont.)

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

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

$$\rho = \frac{2}{\|\mathbf{w}\|} \text{ is maximized}$$

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $\quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Which can be reformulated as:

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

$\mathbf{\Phi}(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $\quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Solving the Optimization Problem

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

- 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
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Non-Separable Case

- When a separating hyperplane does not exist we modify the constraints of the above problem to get:

$$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}^T\mathbf{w} \to min$$

subject to

$$y_i(\mathbf{w}^T x_i + b) \geq 1 - \varepsilon_i$$

$$\varepsilon_i \geq 0, i = 1,2, \dots, n; \quad \sum_{i=1}^{n} \varepsilon_i = C$$

- Here ε_i (called slack variables) allow the margins to be violated and the fixed constant $C > 0$ controls by how much this can happen.

- Certain observations can be on the wrong side of the margin ($\varepsilon_i > 0$), even of the hyperplane ($\varepsilon_i > 1$), but this has a "price/cost" that we cap by C. In this sense C is the budget (the total) for the penalties we wish to pay for the margin violations.

- Equivalent formulation of the SVM optimization problem is

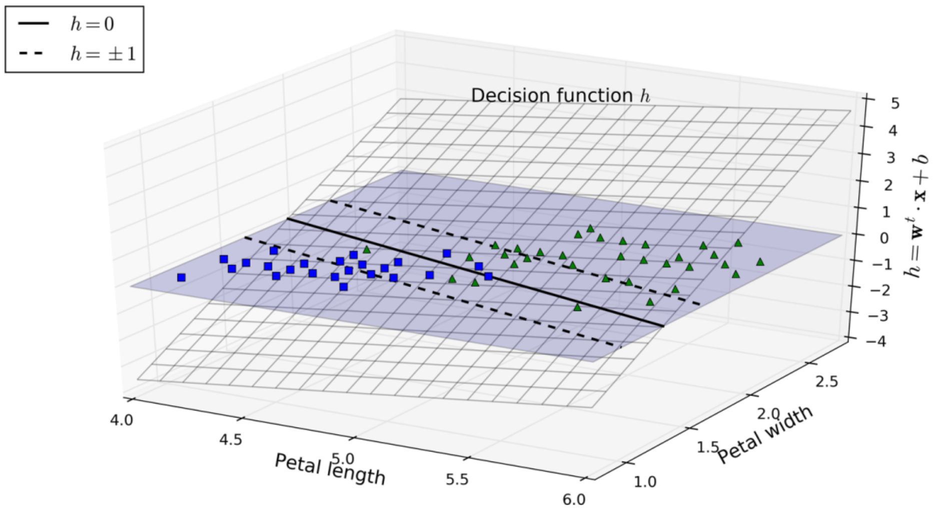
$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \varepsilon_i \ \to min$$

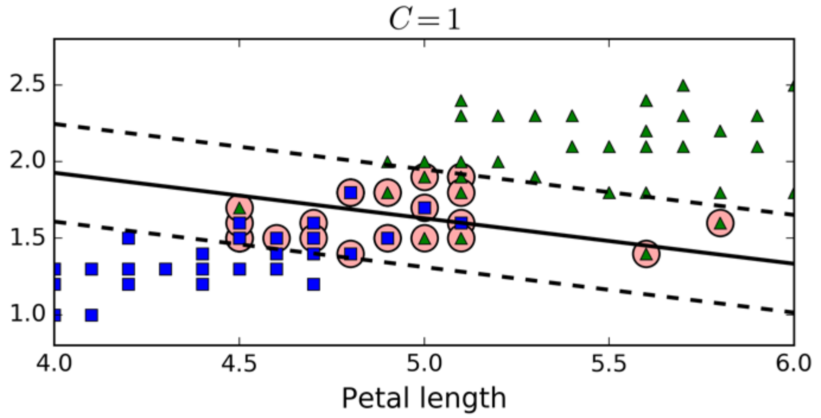
subject to

$$y_i(\mathbf{w}^T x_i + b) \geq 1 - \varepsilon_i$$
$$\varepsilon_i \geq 0, i = 1,2, \dots, n$$

Here C has the interpretation of “cost” (of violating the boundary) not budget.

- The plot below shows the decision function that corresponds to the model to the right. It is a 2-dimensional plane since this dataset has two features (petal width and petal length).



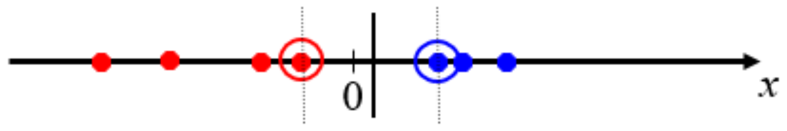


- The decision boundary is the set of points where the decision function is equal to 0: it is the intersection of two planes, which is a straight line (represented by the thick solid line).

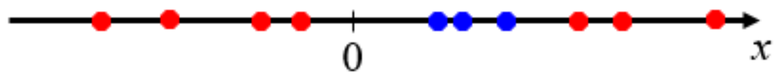
- Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors. These points are circled in red on the last plot.
- To re-iterate, only the support vector observations are used to build the classifier and this makes it more robust
- Sometimes even slack variables are not enough – see next

Non-linear SVMs

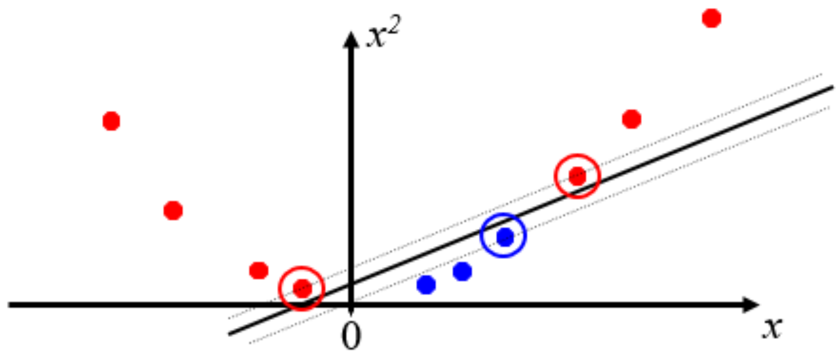
- Datasets that are linearly separable with some noise work out great:



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

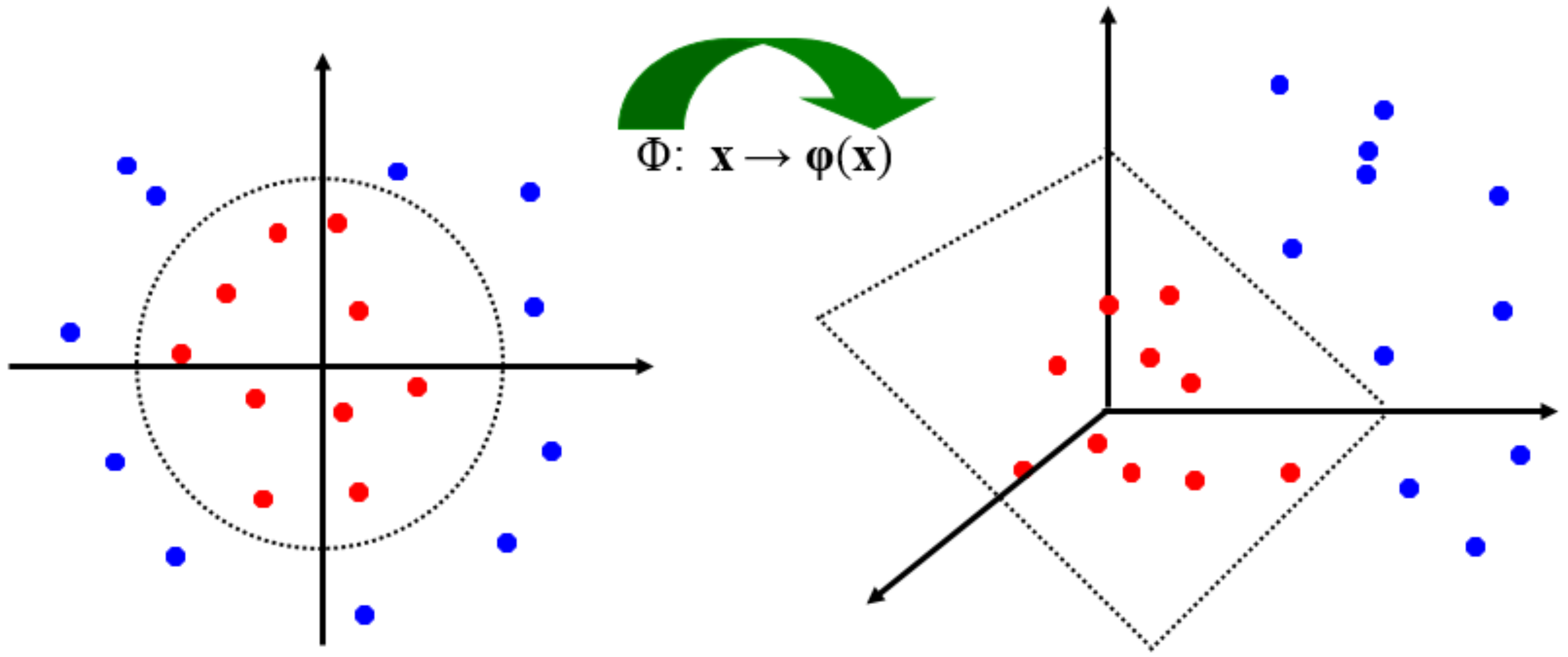


- How about… mapping data to a higher-dimensional space:



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 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:

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

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

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

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_\mathbf{j})= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$:

  $$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2_{,}= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$$
  $$= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^{\mathbf{T}}\, [1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}] =$$
  $$= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\varphi}(\mathbf{x}_j), \quad \text{where } \boldsymbol{\varphi}(\mathbf{x}) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$$
- Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

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
- Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$
  - 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); combination of functions for support vectors is the separator.

- In the radial-basis case there is 1 more tuning parameter (σ)

Example (from the ISLR book)

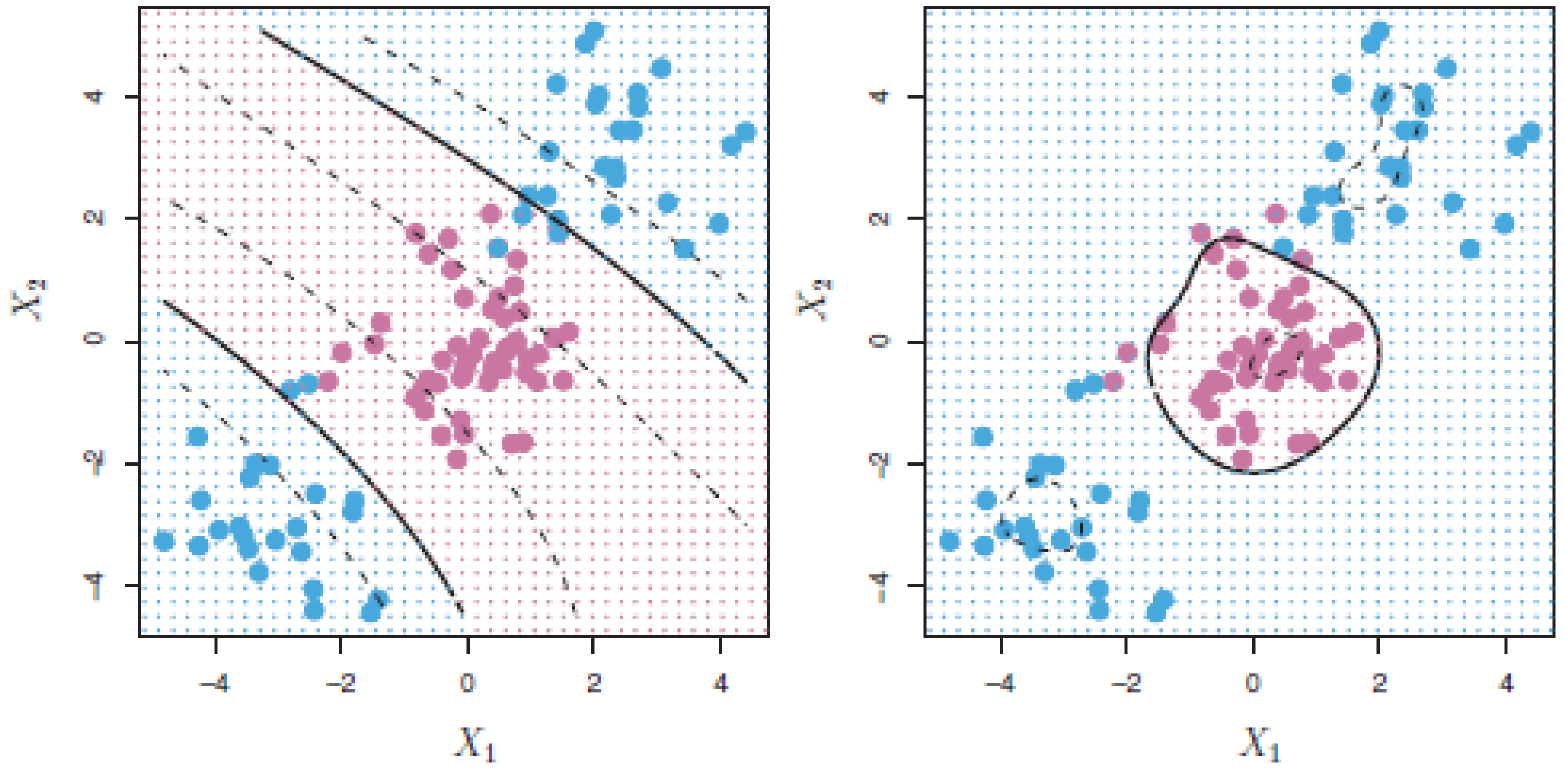


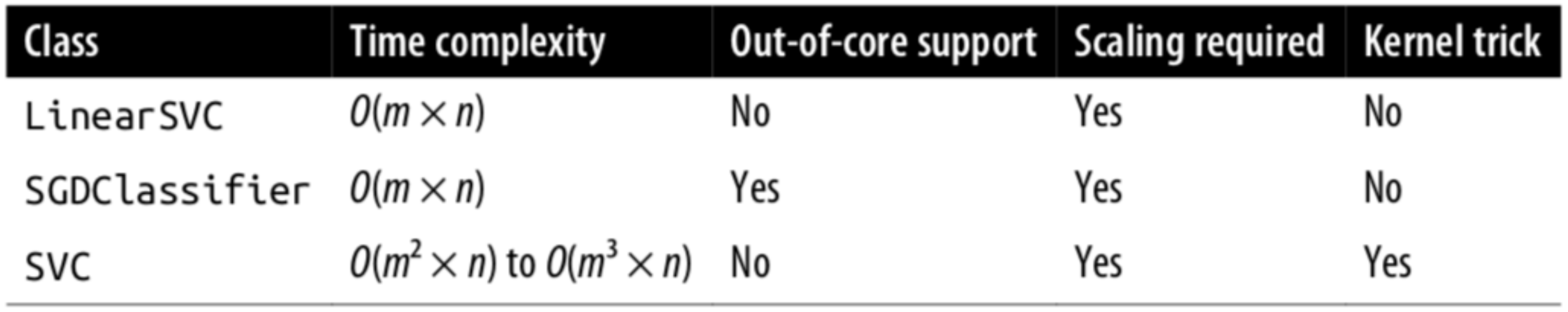
FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

SVM with more than two classes

- There is no organic extension of the method to classify into $K > 2$ classes
- One-versus-one classification is one option – we classify for all the “K choose 2” pairs and then take a majority vote
- One-versus-all classification – we build K SVMs where each time one class is compared to the remaining $K - 1$. Then we pick the class where the “margin” is the largest

Computational Cost

Table 5-1. Comparison of Scikit-Learn classes for SVM classification



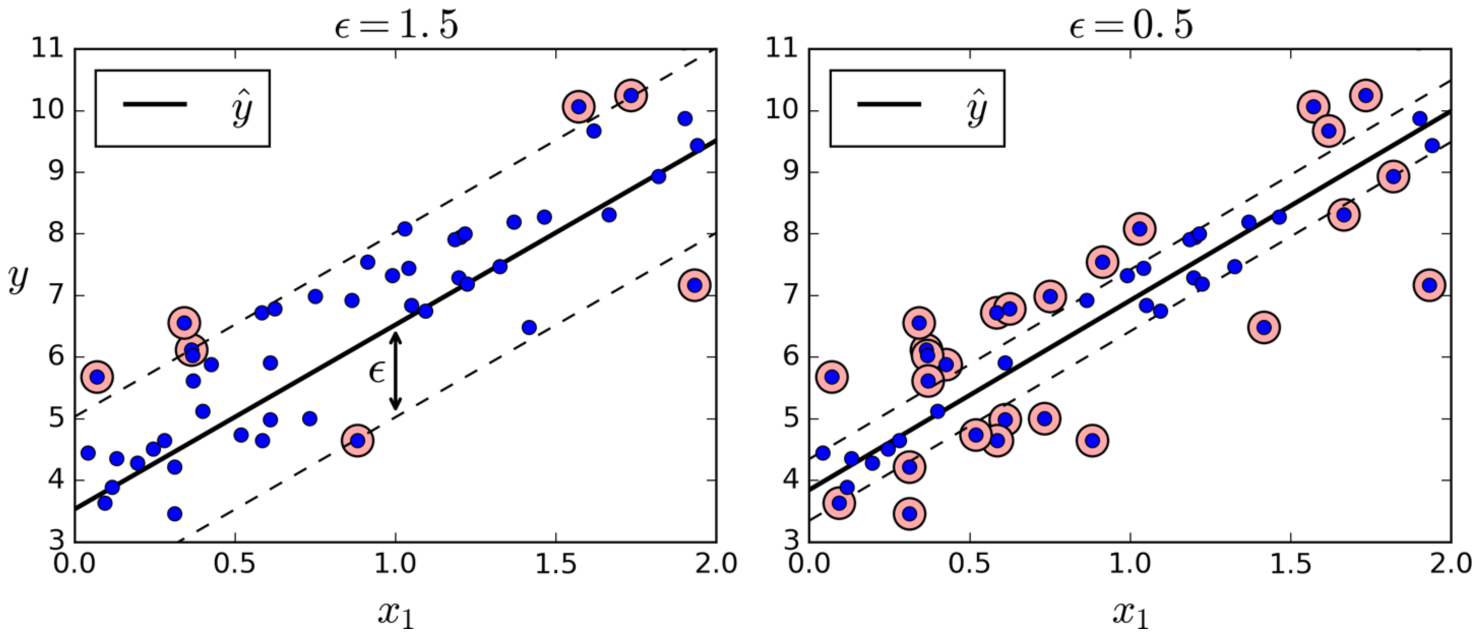
Class	Time complexity	Out-of-core support	Scaling required	Kernel trick
`LinearSVC`	$O(m \times n)$	No	Yes	No
`SGDClassifier`	$O(m \times n)$	Yes	Yes	No
`SVC`	$O(m^2 \times n)$ to $O(m^3 \times n)$	No	Yes	Yes

SVM Regression

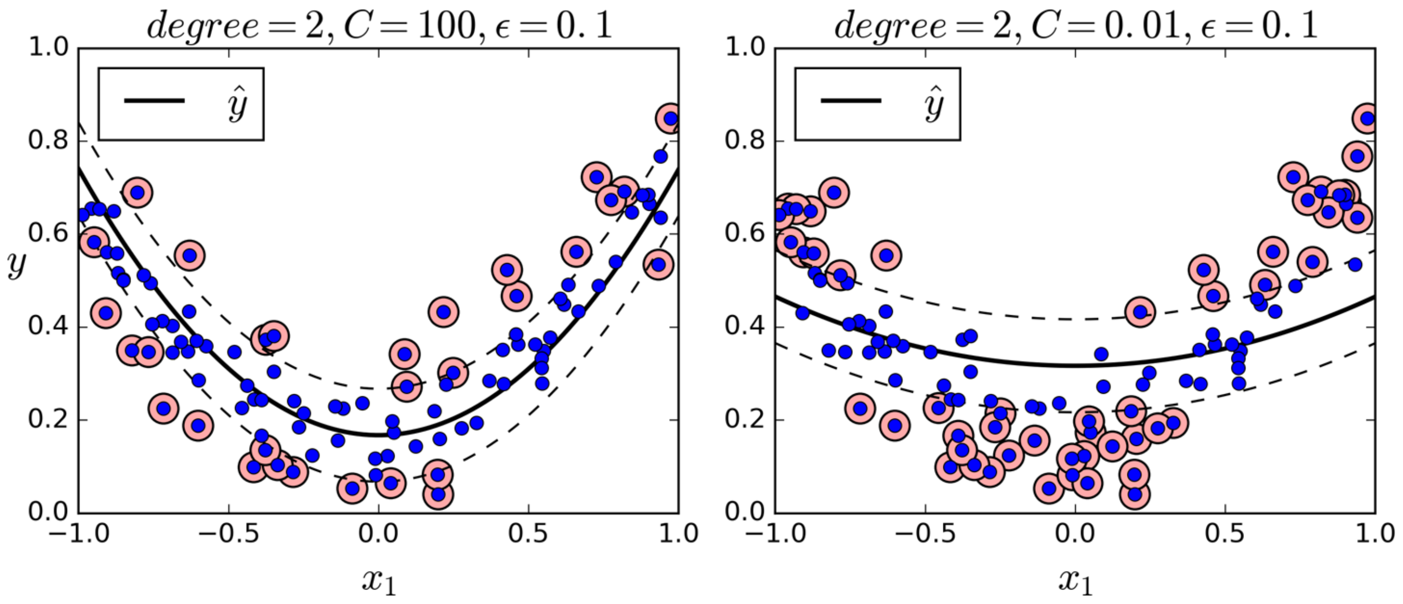
- The SVM algorithm is quite versatile: not only does it support linear and nonlinear classification, but it also supports linear and nonlinear regression.
- The trick is to reverse the objective: instead of trying to fit the largest possible street between two classes while limiting margin violations, SVM Regression tries to fit as many instances as possible on the street while limiting margin violations (i.e., instances off the street).

- The width of the street is controlled by a hyper-parameter ε. Next plot shows two linear SVM Regression models trained on some random linear data, one with a large margin ($\varepsilon = 1.5$) and the other with a small margin ($\varepsilon = 0.5$)

  (Python code)

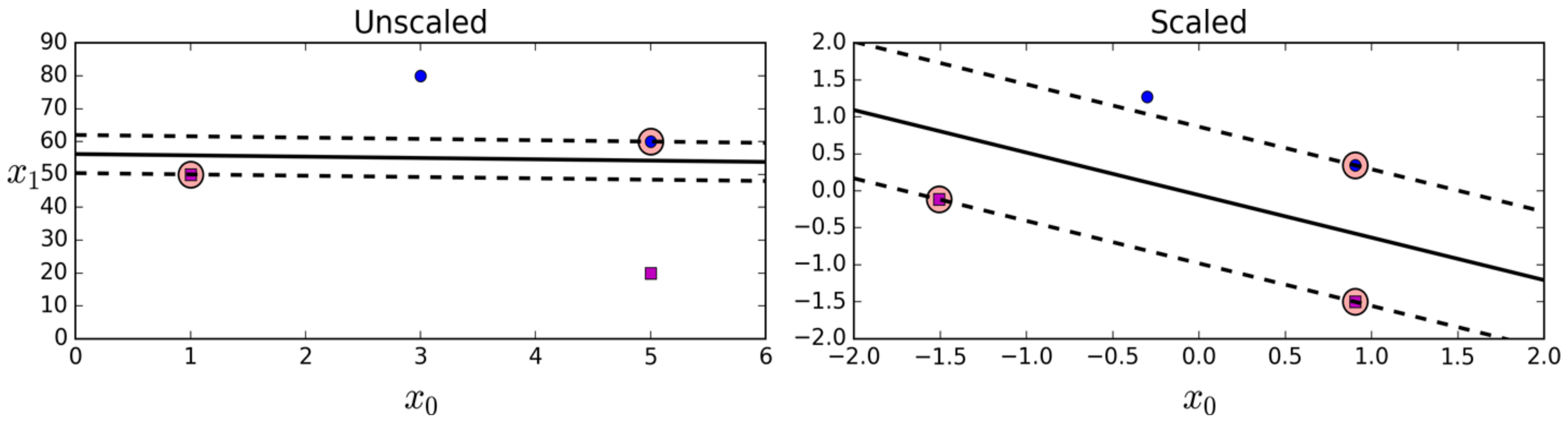


- To tackle nonlinear regression tasks, you can use a kernelized SVM model. Below you can see a SVM Regression on a random quadratic training set, using a 2nd-degree polynomial kernel. There is little regularization on the left plot (i.e., a large C value), and much more regularization on the right plot (i.e., a small C value)



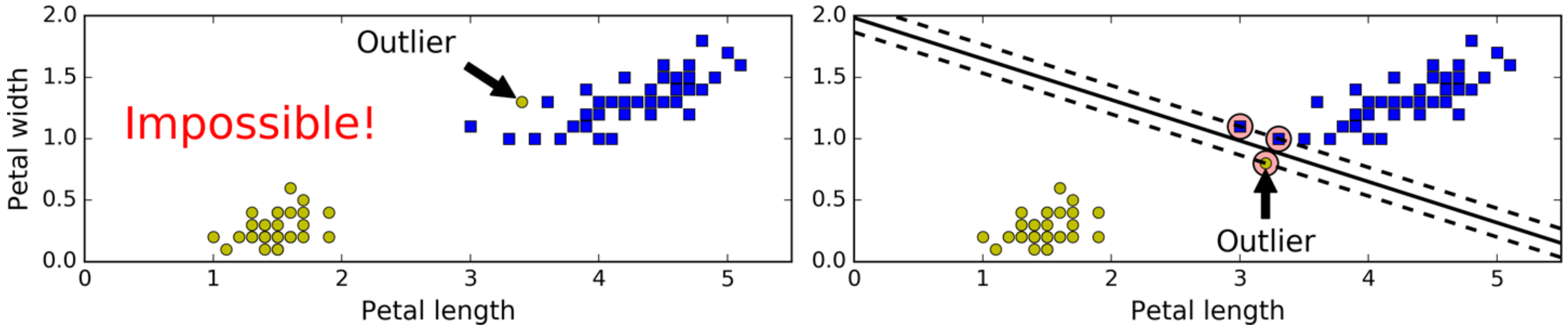
Some things to remember about SVM:

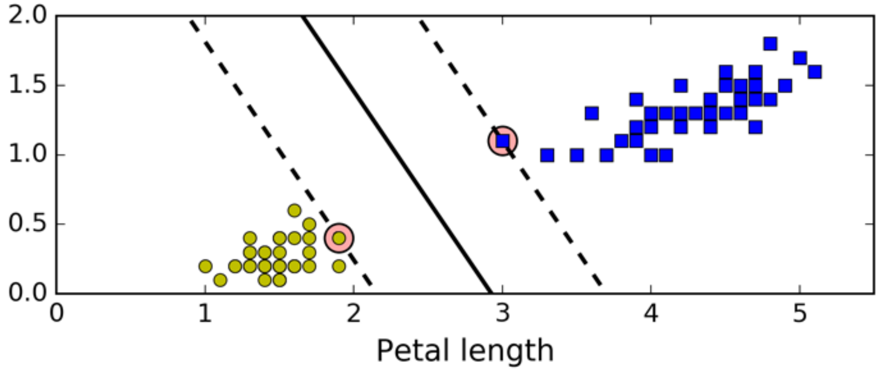
- SVMs are sensitive to the feature scales: on the left plot below, the vertical scale is much larger than the horizontal scale, so the widest possible street is close to horizontal. After feature scaling, the decision boundary looks much better (on the right plot)



Reading:
ISLR: 9.1-9.5 (Lab: 9.6)
Hands-on-ML: Chapter 5 (SVM)

- Below is the `iris` dataset with just one additional outlier: on the left, it is impossible to find a hard margin. On the right the decision boundary ends up very different from the one we have without the outlier (at the bottom) – this fit will probably not generalize as well.


Outlier
Impossible!
Petal width
Petal length
Outlier
Petal length


Petal length