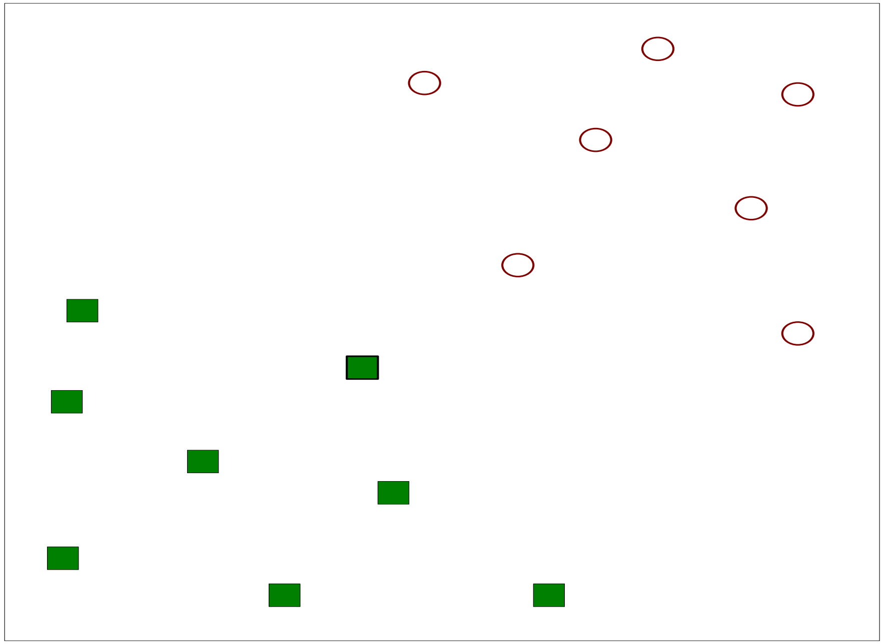
Support Vector Machine

Support Vector Machines

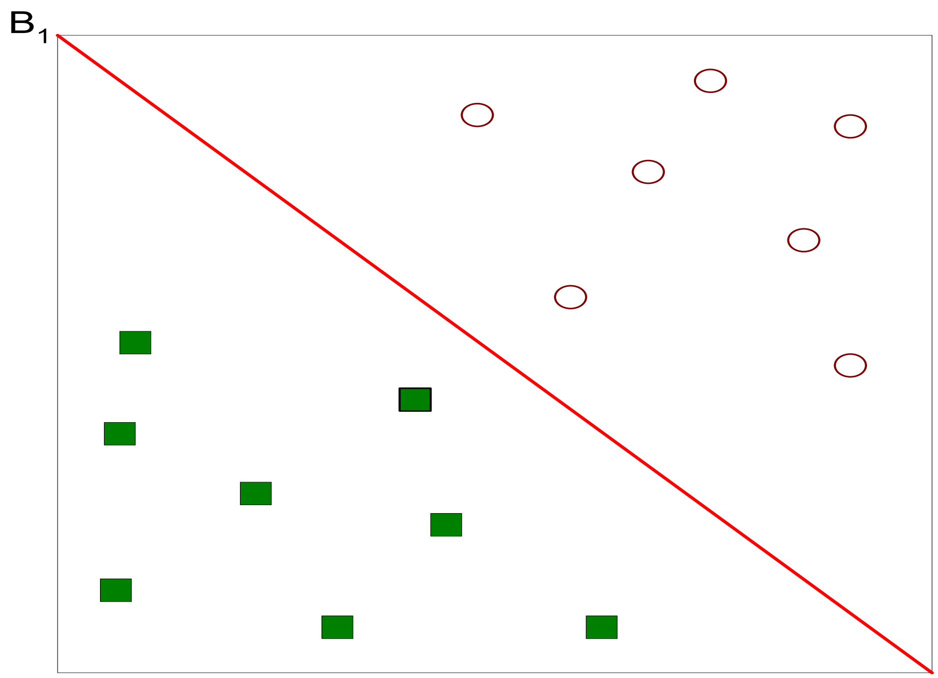
- Linear models can be used to implement nonlinear class boundaries using space transformations
  - The instance space is transformed into a new space using a nonlinear mapping
  - A straight line in the new space can represent a nonlinear decision boundary in the original space
- What kind of transformations?
  - Polynomials of sufficiently high degree can approximate arbitrary decision boundaries to any required accuracy

Support Vector Machines



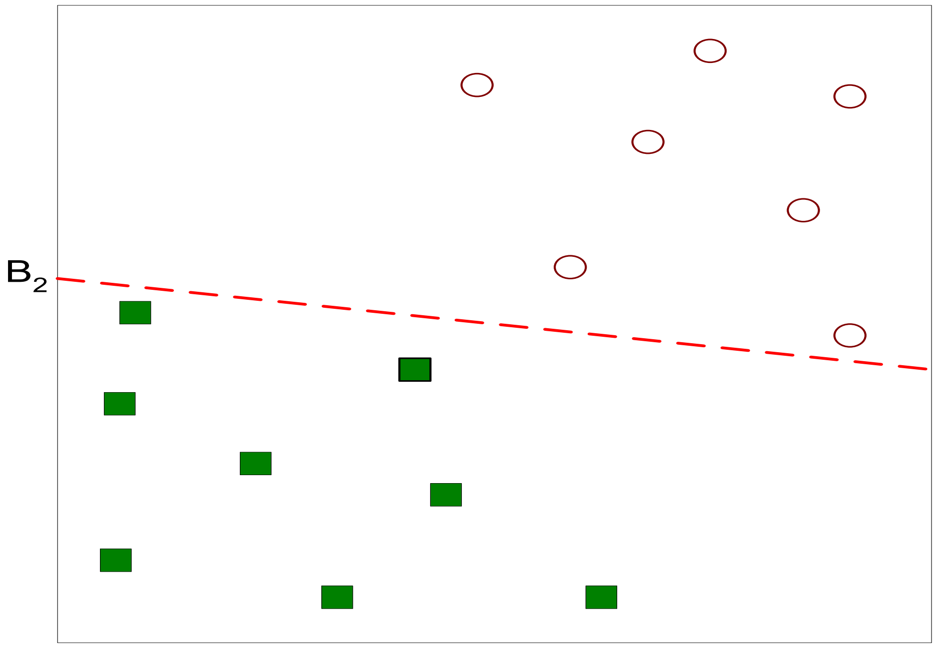
Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



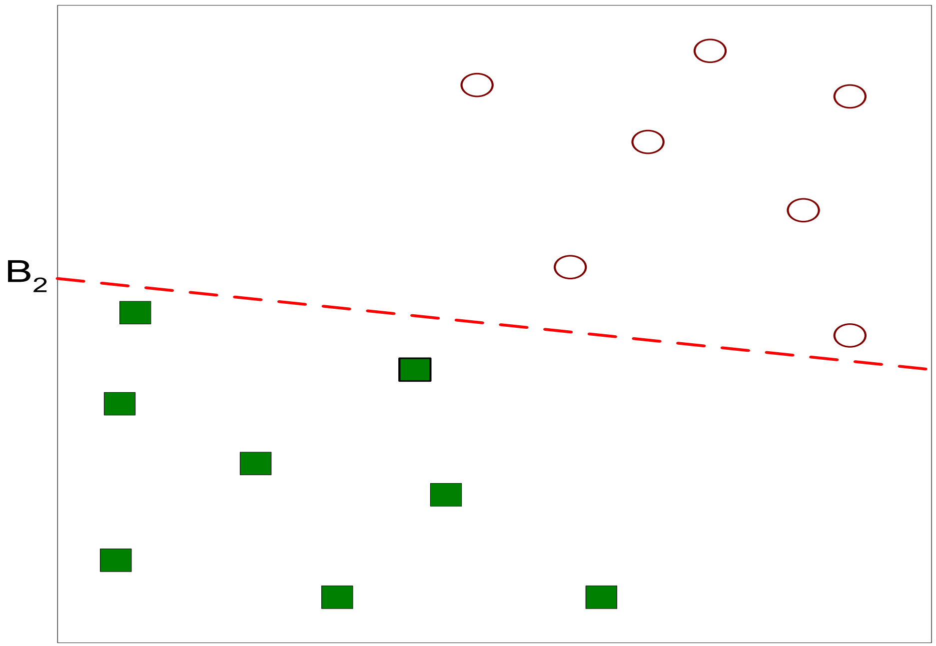
One possible solution

Support Vector Machines



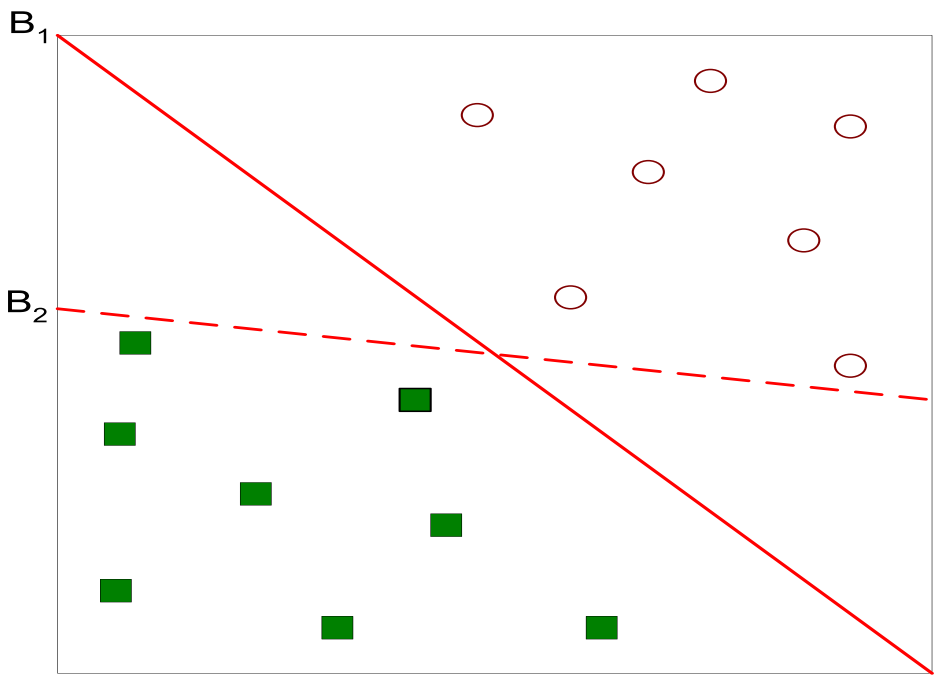
Another solution

Support Vector Machines



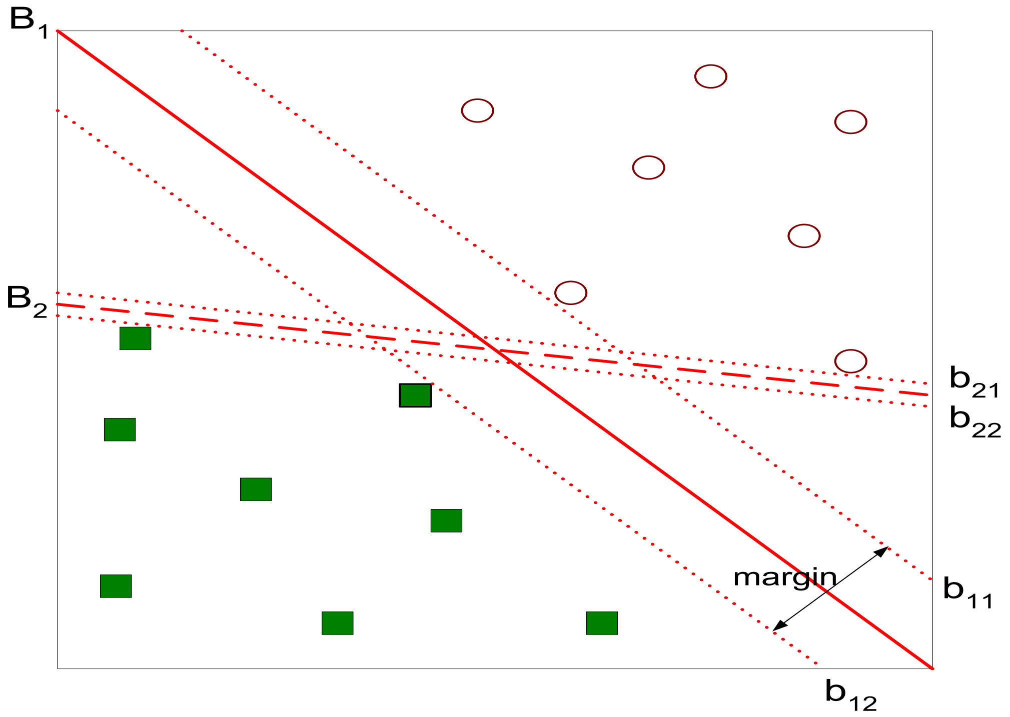
Other possible solutions

Support Vector Machines



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

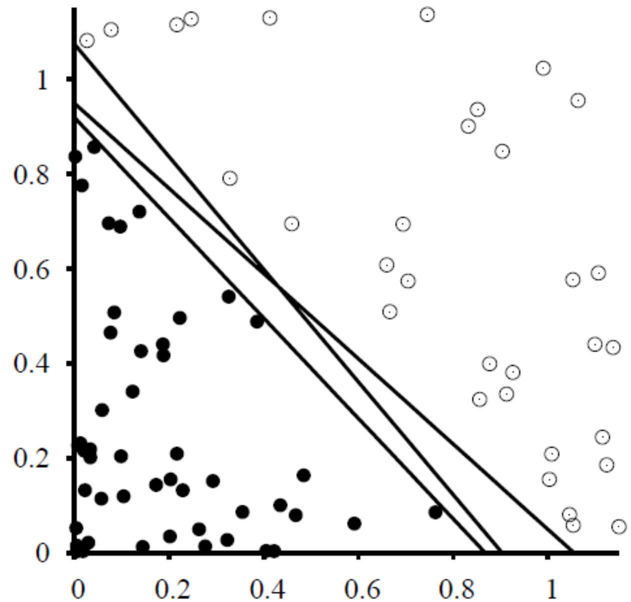
Support Vector Machines



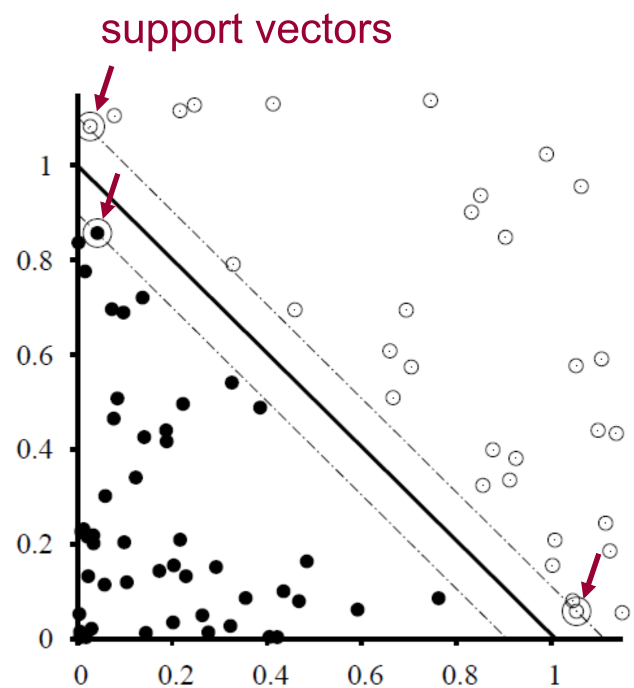
Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines

- Attractive properties:
  - SVMs construct maximum margin separator
    - Minimize expected generalization loss rather than empirical loss



(a) candidate linear separators



(b) maximum margin separator

Support Vector Machines

- Attractive properties:
  - SVMs construct maximum margin separator ☞
    - Minimize expected generalization loss rather than empirical loss
  - SVMs map the data into a higher-dimensional space in which the data become linearly separable
  - SVMs are a nonparametric method retaining only a small fraction of training examples
    - Can represent complex functions but resistant to overfitting
- Key insight: maximum margin separator
  - Some examples are more important than the others
  - Only support vectors matter → better generalization

Support Vector Machines

- In the two-attribute case, a hyperplane separating the two classes (1 or –1) might be written as

$$y = w_0 + w_1 x_1 + w_2 x_2$$

  where x_1 and x_2 are the attribute values, and there are three weights w_i to be learned

- In general, for a training set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, we want to find $\mathbf{w}$ and w_0 such that

$$\mathbf{w}^T\mathbf{x}_i + w_0 \geq m \quad \text{if } y_i = 1$$

$$\mathbf{w}^T\mathbf{x}_i + w_0 \leq -m \quad \text{if } y_i = -1$$

  where m is the smallest margin of any positive or negative example

- The above can be rewritten as the following constraint

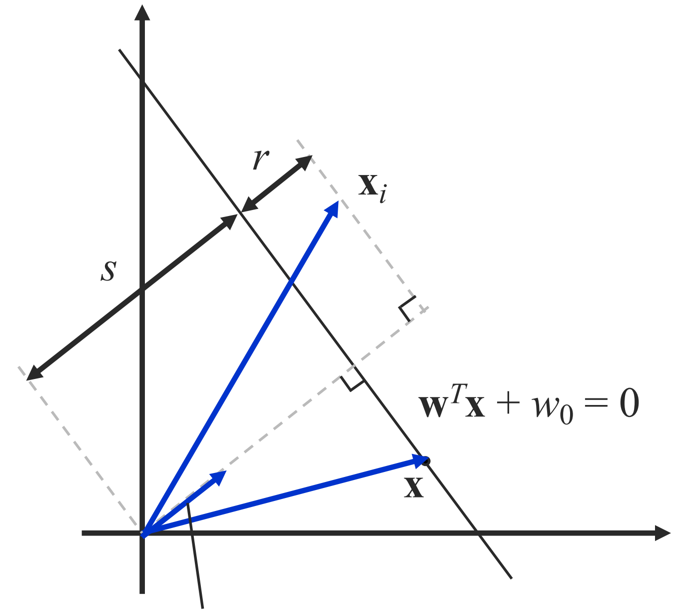
$$y_i\,(\mathbf{w}^T\mathbf{x}_i + w_0) \geq m$$

Support Vector Machines

- To calculate the distance from an example to the hyperplane, we need to know the distance from the origin to the hyperplane
- The distance s from the origin to the hyperplane can be obtained by the inner product of a point $\mathbf{x}$ on the hyperplane and the unit normal vector

$$s = \frac{\mathbf{w}^T}{\|\mathbf{w}\|} \cdot \mathbf{x} = \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = \frac{-w_0}{\|\mathbf{w}\|}$$

because $\mathbf{x}$ satisfies $\mathbf{w}^T\mathbf{x} + w_0 = 0$, i.e., $\mathbf{w}^T\mathbf{x} = -w_0$



$\frac{\mathbf{w}^T}{\|\mathbf{w}\|}$: unit normal vector

Support Vector Machines

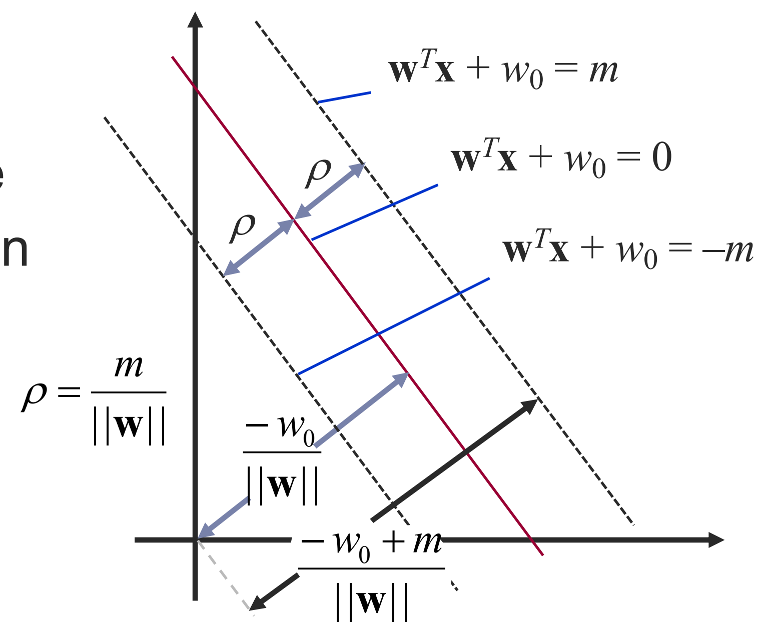
- Thus, the distance r from an example $\mathbf{x}_i$ to the hyperplane is

$$r = \left| \frac{\mathbf{w}^T \mathbf{x}_i}{\|\mathbf{w}\|} - s \right| = \left| \frac{\mathbf{w}^T \mathbf{x}_i}{\|\mathbf{w}\|} - \frac{-w_0}{\|\mathbf{w}\|} \right| = \frac{\left| \mathbf{w}^T \mathbf{x}_i + w_0 \right|}{\|\mathbf{w}\|} = \frac{y_i(\mathbf{w}^T \mathbf{x}_i + w_0)}{\|\mathbf{w}\|}$$

- We want this distance to be at least some value ρ for all i:

$$\frac{y_i(\mathbf{w}^T \mathbf{x}_i + w_0)}{\|\mathbf{w}\|} \geq \rho, \quad \forall i$$

- We want to maximize ρ but there are an infinite number of solutions we can get by scaling $\mathbf{w}$



Support Vector Machines

- For a unique solution, we fix $\rho\,||\mathbf{w}|| = 1$ (i.e., $m = 1$) and thus, to maximize ρ, we minimize $||\mathbf{w}||$, or more conveniently:

$$\min \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1,\ \forall i$$

  - This is a standard quadratic optimization problem and can be solved directly to find $\mathbf{w}$ and w_0 (numerical approaches can solve it)
  - The complexity depends on the input dimensionality d which can be quite large especially in the transformed space
  - Note that the instances are at least $\rho = 1/||\mathbf{w}||$ away from the hyperplane and the total margin becomes $2/||\mathbf{w}||$

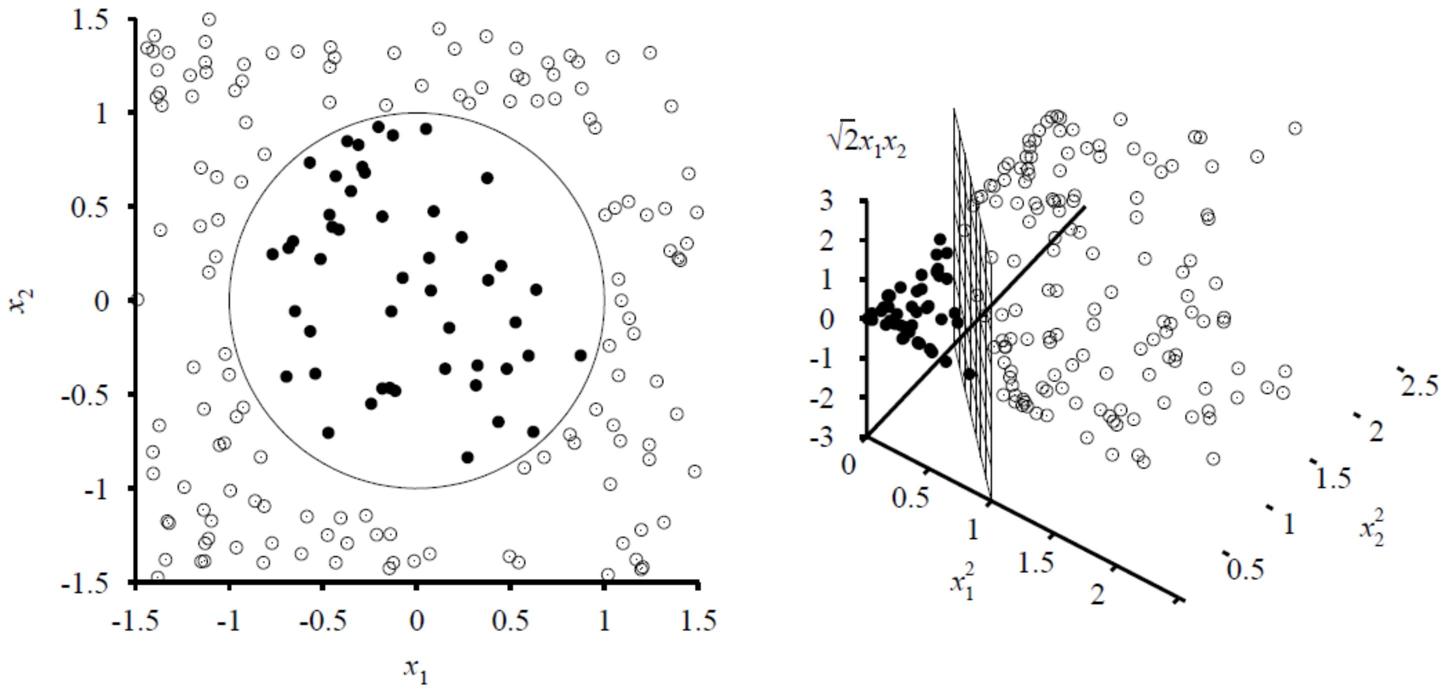
Support Vector Machines

- Now the classifying function becomes

$$g(\mathbf{x}) = \operatorname{sign}\left(\sum_j \alpha_j y_j (\mathbf{x} \cdot \mathbf{x}_j) + b \right)$$

- Properties:
  - The data enter the expression only in the form of dot products of pairs of points
  - $\alpha_j = 0$ except for the support vectors
    $\rightarrow$ SVMs gain some advantages of parametric models
- Nonlinear decision boundaries:
  - $\mathbf{x}_j \cdot \mathbf{x}_k$ can be replaced by a kernel function $K(\mathbf{x}_j \cdot \mathbf{x}_k)$ that is equivalent to $F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)$
  - To find a linear separator not in the input space $\mathbf{x}$ but in the high-dimensional feature space $F(\mathbf{x})$

Support Vector Machines



(a) true decision boundary:

$$x_1^2 + x_2^2 \le 1$$

(b) after mapping to the space

$$(x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$F(\mathbf{x}_j) \cdot F(\mathbf{x}_k) = (\mathbf{x}_j \cdot \mathbf{x}_k)^2 = K(\mathbf{x}_j \cdot \mathbf{x}_k)$$

$$(a^2, b^2, \sqrt{2}ab) \cdot (c^2, d^2, \sqrt{2}cd) = a^2c^2 + b^2d^2 + 2acbd = (ac + bd)^2 = ((a, b) \cdot (c, d))^2$$

Support Vector Machines

- Kernel trick:
  - Polynomial kernel, $K(\mathbf{x}_j \cdot \mathbf{x}_k) = (1 + \mathbf{x}_j \cdot \mathbf{x}_k)^d$, corresponds to a feature space whose dimension increases fast with d
  - The dot product can be calculated before the nonlinear mapping is performed in the original feature space
    $\longrightarrow$ efficient computation

- Any algorithms for learning linear models can be upgraded by applying the kernel trick:
  - Reformulate to work only with dot products of pairs of data points
  - Replace the dot product by a kernel function

Support Vector Machines

- Soft margin classifier:
  - For noisy data, we want to find a decision surface in a lower-dimensional space that do not cleanly separate the classes
  - Examples on the wrong side are penalized