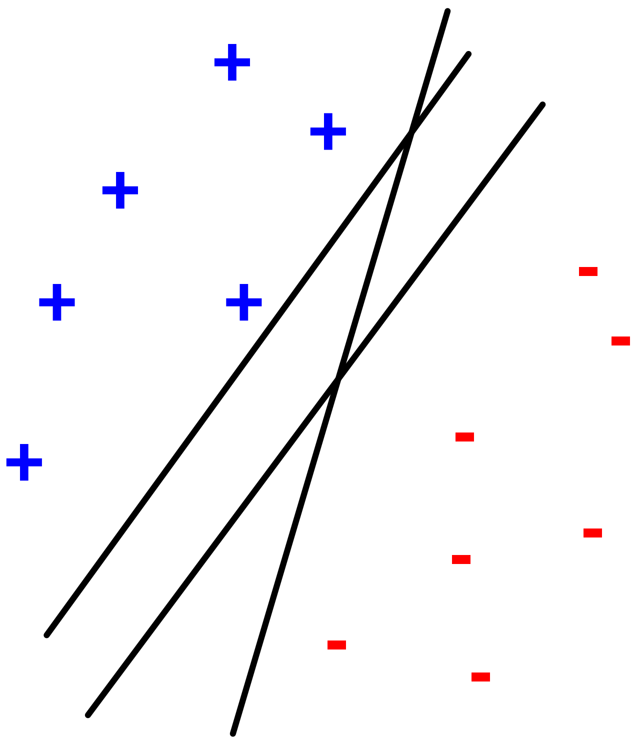
Support Vector Machine

Reference: Mining Massive Datasets (J. Leskovec, A. Rajaraman, J. Ullman)

Support Vector Machines

- Want to separate "+" from "-" using a line



Data:

- **Training examples:**
  - $(x_1, y_1) \dots (x_n, y_n)$
- **Each example *i*:**
  - $x_i = (x_i^{(1)}, \dots , x_i^{(d)})$
    - $x_i^{(j)}$ is real valued
  - $y_i \in \{ -1, +1 \}$
- **Inner product:**

$$\boldsymbol{w} \cdot \boldsymbol{x} = \sum_{j=1}^{d} w^{(j)} \cdot x^{(j)}$$

Which is best linear separator (defined by *w*)?

Non Overlapping Class

- SVM is a binary classifier
  - Considered by many as one of the best supervised learning algorithms
  - Decision boundary is placed at the optimal that maximizes the distance to the nearest data points

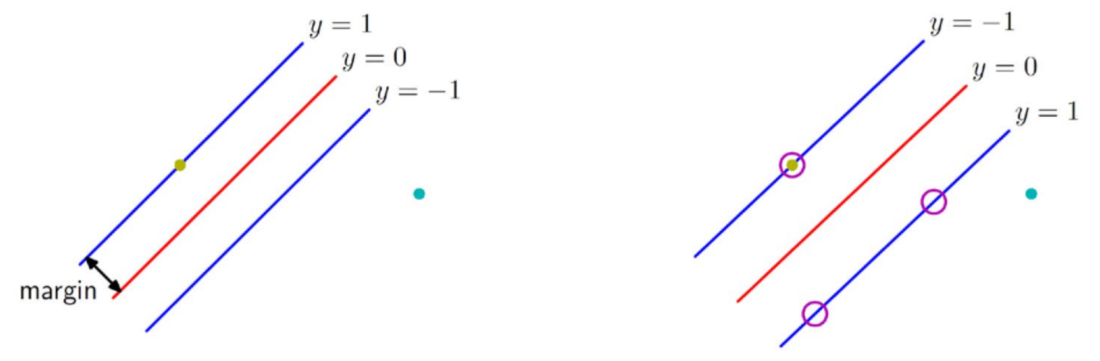


Figure 7.1 The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

How to place the decision boundary

- Solve the following convex optimization problem:

$$\min_w \tfrac{1}{2}\|w\|^2$$
$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1$$

  - $\boldsymbol{w}.\boldsymbol{x} + \boldsymbol{b}$ defines the decision boundary

- This is called the primal form.
- It is a quadratic programming problem.
- The solution yields an optimal margin classifier.

How to find *w* and *b*?

- **Standard way:** Use a quadratic solver!
  - **Solver:** software for finding solutions to "common" optimization problems

Lagrangian Duality

- Primal Problem:

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

$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- Dual Problem:

$$\max_{\alpha} J(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \left(\mathbf{x}_i^T \mathbf{x}_j\right)$$

$$\text{subject to } \alpha_i \geq 0, i = 1, \ldots, N \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0$$

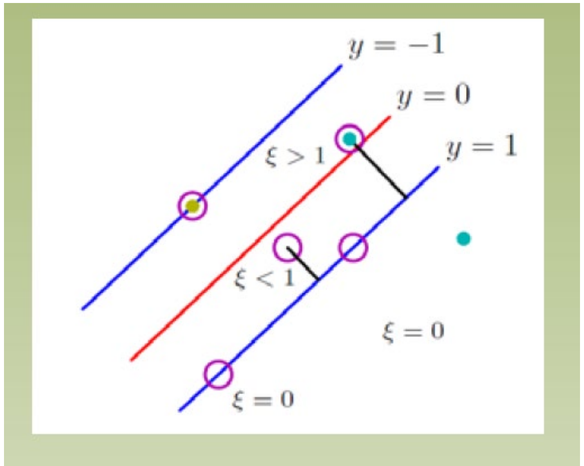
where $\left(\mathbf{x}_i^T \mathbf{x}_j\right)$ is the *kernel*

Classification

- To classify new data $\mathbf{z}$
  - Compute $h = \mathbf{w}^T\mathbf{z} + b = \sum_{i=1}^{M} \alpha_i y_i \left(\mathbf{x}_i^T \mathbf{z}\right) + b$
    - *M* (support vectors) << *N* (training examples)
  - If $h \geq 0$ → positive label, else negative label

Overlapping Class Distribution

- **Introduce** **slack variables** ξ_i
  - Correct classification: $\xi_i = 0$
  - Inside the band: $0 < \xi_i \leq 1$
  - Misclassified: $\xi_i > 1$
  - Make the margin as large as possible while keeping the number points with $\xi_i > 1$ as small as possible



$$\min_{w,b,\xi_i \geq 0} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} \xi_i$$

$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i > 0$$

Overlapping Class Distribution

- **The role of regularization parameter *C*:**
  - Controls the influence of the two terms in the cost function $\min_{w,b,\xi_i \geq 0} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} \xi_i$

    $$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i > 0$$

  - Large *C*: don’t want misclassified points but will accept narrow margin
  - Small *C*: accept some misclassified points but want big margin
  - *C* is usually obtained via **cross validation**

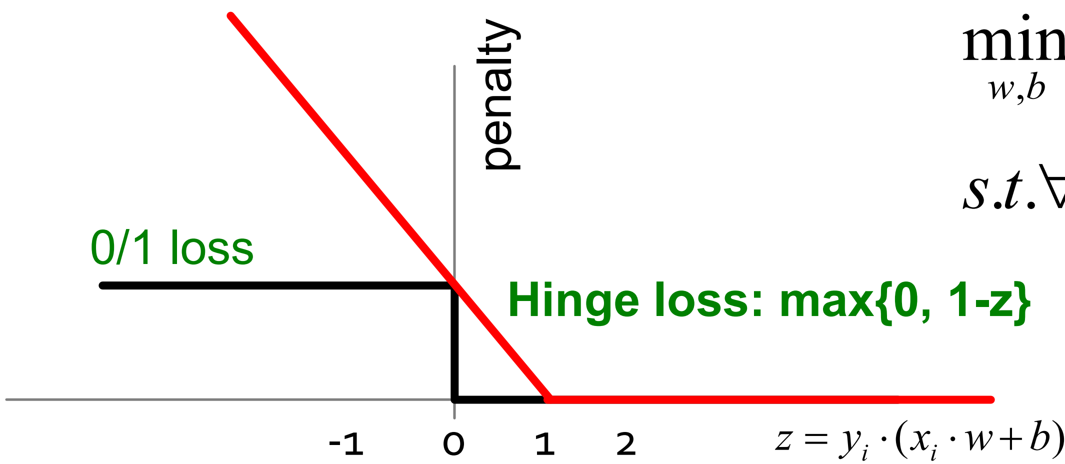
Support Vector Machine

- **SVM in the "natural" form**

$$\arg\min_{w,b}\ \frac{1}{2}\underbrace{w\cdot w}_{\text{Margin}} + C\cdot\sum_{i=1}^{n}\underbrace{\max\{0,1-y_i(w\cdot x_i+b)\}}_{\text{Empirical } \textbf{loss L} \text{ (how well we fit training data)}}$$

Regularization parameter

- **SVM uses "Hinge Loss":**

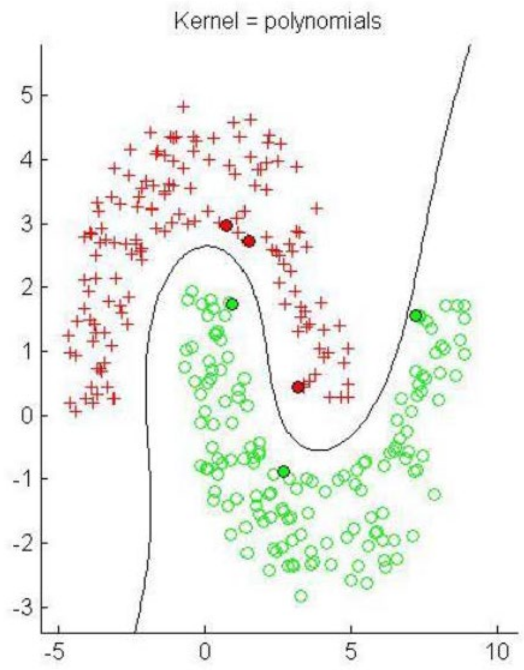


$$\min_{w,b}\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$s.t.\forall i,\ y_i\cdot(w\cdot x_i+b)\geq 1-\xi_i$$

Non Linear SVM

- Complex non-linear cases need non-linear classifier



Kernel Function

- Assume a mapping function to transform the data to higher dimensional space where it is linearly separable
  - $\mathbf{x} \in \mathbb{R}^d \rightarrow \mathbf{y} \in \mathbb{R}^k$
  - *k* is generally much higher than *d*

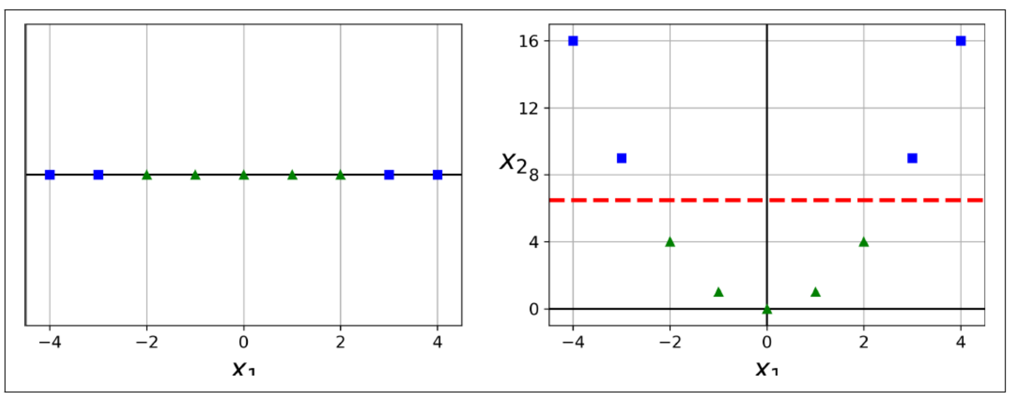
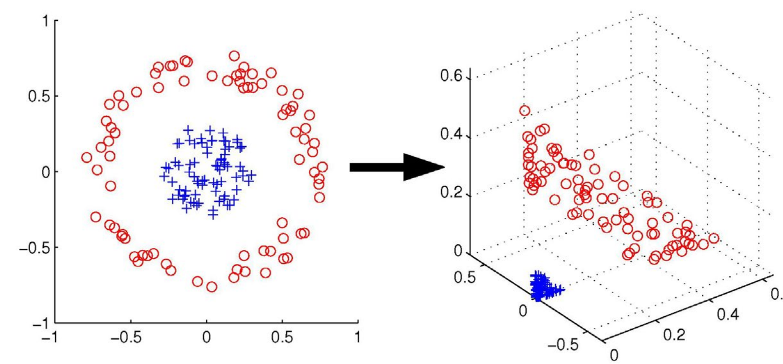


Figure 5-5. Adding features to make a dataset linearly separable



Reference: M.I. Jordan

How to apply a kernel function

- Since SVM can be written in terms of the inner products, they can be replaced with a kernel function
- Optimization : replace $\left(\mathbf{x}_i^T\mathbf{x}_j\right)$ with $K(\mathbf{x}_i, \mathbf{x}_j)$
  - $\max_\alpha J(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \left(\mathbf{x}_i^T\mathbf{x}_j\right)$
  - subject to $\alpha_i \geq 0, i = 1, \dots, N$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$
- Classification : replace $\left(\mathbf{x}_i^T\mathbf{z}\right)$ with $K(\mathbf{x}_i, \mathbf{z})$
  - $h = \mathbf{w}^T\mathbf{z} + b = \sum_{i=1}^{M} \alpha_i y_i \left(\mathbf{x}_i^T\mathbf{z}\right) + b$

Kernel Trick

- Mercer's Theorem
  - Let $\mathbf{x} \in \mathbb{R}^d$ and a mapping function ϕ
    - $\mathbf{x} \rightarrow \phi(\mathbf{x}) \in H$ where H is a Hilbert space
  - Let the inner product operation have equivalent kernel representation
    - $K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ where $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$

- The theorem does not disclose how to find the space or the dimensionality of $\phi(\mathbf{x})$

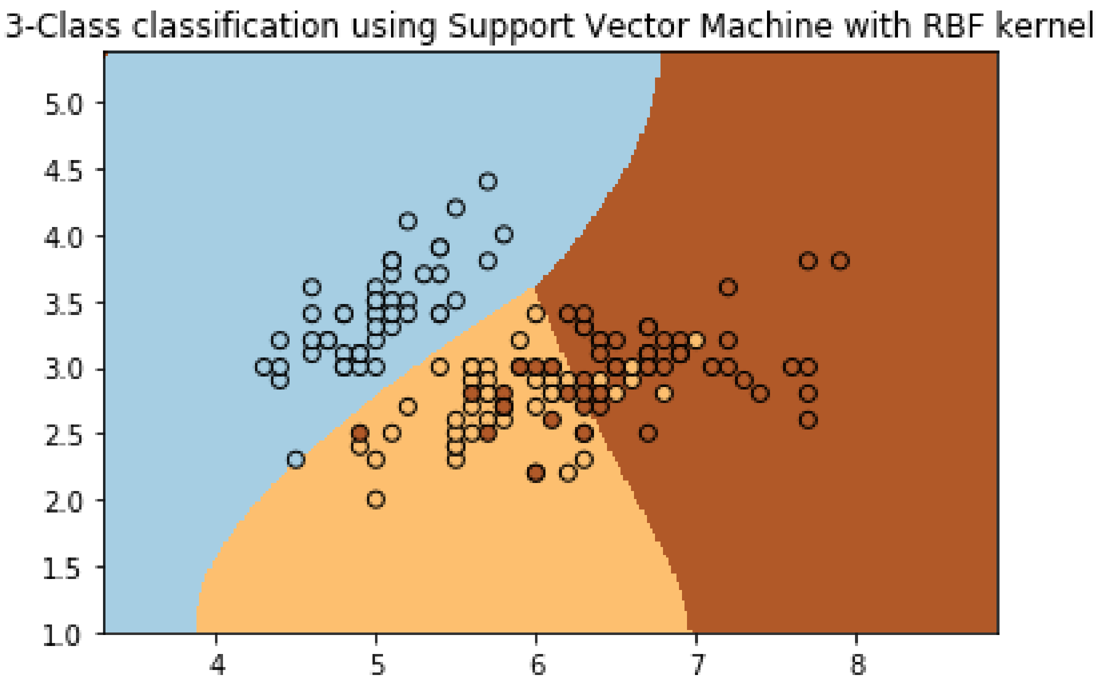
- It is not necessary, i.e., computing $K(\mathbf{x}, \mathbf{z})$ is sufficient, hence called *a kernel trick*

Typical Kernels

- Dot product : $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$
- Polynomials : $K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + 1)^p$ where $p > 0$
- Radial Basis Function : $K(\mathbf{x}, \mathbf{z}) = \exp\left[-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma^2}\right]$
- Hyperbolic Tangent : $K(\mathbf{x}, \mathbf{z}) = \tanh(\beta\, \mathbf{x}^T \mathbf{z} + \gamma)$

Example from scikit-learn

http://scikit-learn.org/stable/index.html



To see the code go to

http://scikit-learn.org/stable/auto_examples/svm/plot_iris.html#sphx-glr-auto-examples-svm-plot-iris-py

Multiclass SVM

One-Vs-All (OVA)

- One-Vs-Rest, One-Against-Rest

- Train *K* classifiers, each to distinguish its own label from the remaining classes
  - *K* = the number of classes

- Apply new data to all *K* classifiers. Choose the label based on "$h = \boldsymbol{w}^T \boldsymbol{z} + b$" that produces the largest (most positive)

All-Vs-All (AVA)

- All_vs_all / one_vs_one / all_pairs
- Train *K(K-1)/2* classifiers to distinguish each pair of labels
- Apply new data to all *K* classifiers. Determine the final class label by majority voting