SUPPORT VECTOR MACHINES

Today

- Reading
  - AIMA 18.9
- Goals
  - (Naïve Bayes classifiers)
  - Support vector machines

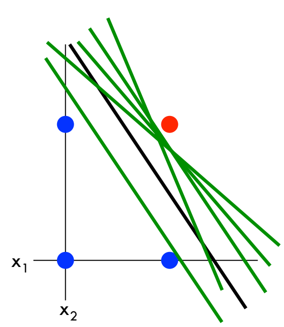
Support Vector Machines (SVMs)

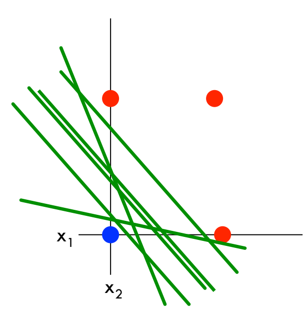
- SVMs are probably the most popular off-the-shelf classifier!
- Software Packages
  - LIBSVM (LIBLINEAR) – on the Resources page
  - SVM-Light

Linearly Separable

x_1	x_2	x_1 and x_2
0	0	0
0	1	0
1	0	0
1	1	1

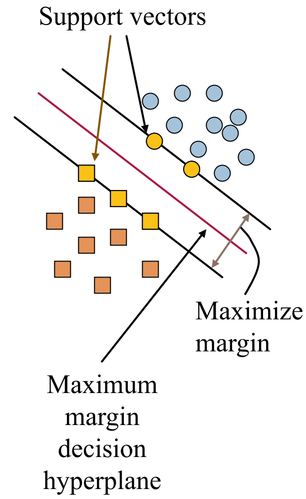
x_1	x_2	x_1 or x_2
0	0	0
0	1	1
1	0	1
1	1	1



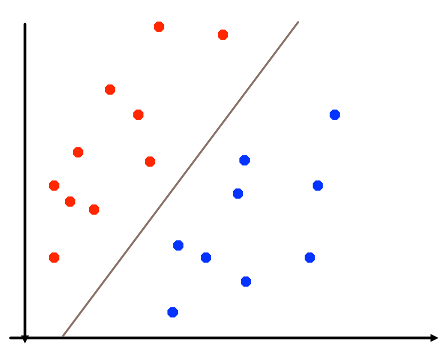


Support Vector Machines

- A support vector machine (SVM) is a linear classifier that finds the decision boundary btw. two classes that is *maximally far from any point in the training set*
- The margin is the distance from the decision boundary to the closest data point
- The support vectors are a subset of the training examples that fully determine the decision boundary



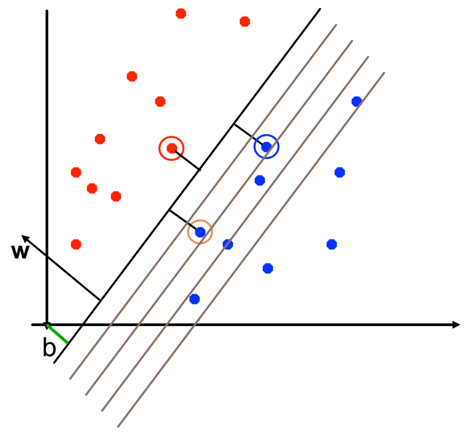
What defines a hyperplane?



What defines a hyperplane?

A hyperplane is defined by:

- A vector w
  - Perpendicular to the hyperplane
  - Often called the “weight” vector
- A scalar b
  - Selects the hyperplane that is distance b from the origin from among all possible hyperplanes

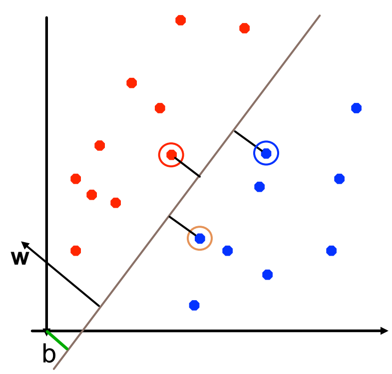


How do we classify an example?

$$D = \{(x_i, y_i) | i = 1 \dots N\}$$

$$y_i \in \{-1, 1\}$$

$w^\mathsf{T} x + b = 0$ x on the decision boundary
$w^\mathsf{T} x + b < 0$ x “below” the decision boundary
$w^\mathsf{T} x + b > 0$ x “above” the decision boundary



$$g(x_i) = \text{sign}(w^\mathsf{T} x + b)$$

The hyperplane that maximizes the margin

- We know how to specify a hyperplane (w and b).
- Given the hyperplane, we know how to predict.
- But how do we find the hyperplane with the maximum margin?

(Derivation on board)

Solving the Optimization Problem

$$\min_{w,b} \frac{1}{2}||w||^2 \text{ such that } y^{(i)}\left(w^\intercal x^{(i)} + b\right) \geq 1 \quad \forall i$$

- Need to optimize a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem and many algorithms exist for solving them
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* (a scalar value) is associated with every constraint in the primary problem

Solving the Optimization Problem

$$\min_{w,b} \frac{1}{2}||w||^2 \text{ such that } y^{(i)}\left(w^\intercal x^{(i)} + b\right) \geq 1 \quad \forall i$$

$$\downarrow$$

$$\underbrace{\max_{\alpha}}_{\text{Lagrange multipliers}} \min_{w,b} \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i \left[y^{(i)}\left(w^\intercal x^{(i)} + b\right) - 1\right] \quad \text{Dual}$$

$$\downarrow$$

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} x^{(j)}$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_i \alpha_i y^{(i)} = 0$$

Solving the Optimization Problem

- The solution has the form:

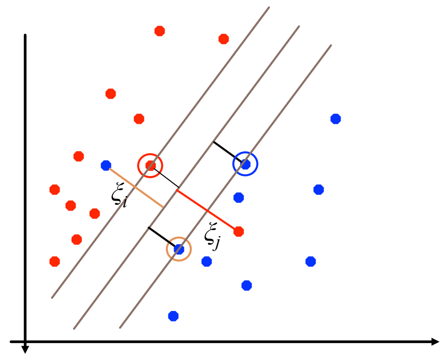
$$w = \sum_{i=1}^{N} \alpha_i y^{(i)} x^{(i)} \text{ and } b = y^{(i)} - w^\intercal x^{(i)} \text{ for any } x^{(i)} \text{ s.t. } \alpha_i \neq 0$$

- Each non-zero alpha indicates corresponding x_i is a support vector
- The classifying function has the form: $g(x_i) = \text{sign}\left(\sum_i \alpha_i y^{(i)} x^{(i)} x + b\right)$
- Relies on an inner product between the test point x and the support vectors x_i

Soft-margin Classification

If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.

Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



How many support vectors?

- Determined by alphas in optimization
- Typically only a small proportion of the training data
- The number of support vectors determines the run time for prediction

How fast are SVMs?

Training

- Time for training is dominated by the time for solving the underlying quadratic programming problem
- Slower than Naïve Bayes
- Non-linear SVMs are worse

Testing (Prediction)

- Fast - as long as we don't have too many support vectors

Multi-label classification

- SVMs are inherently two-class classifiers
- Given C classes, common techniques are:
  - One-versus-all
    - Train C different SVMs where each SVM learns one class versus all the other classes
  - One-versus-one
    - Train C(C-1)/2 SVMs where each SVM learns to distinguish one class from another

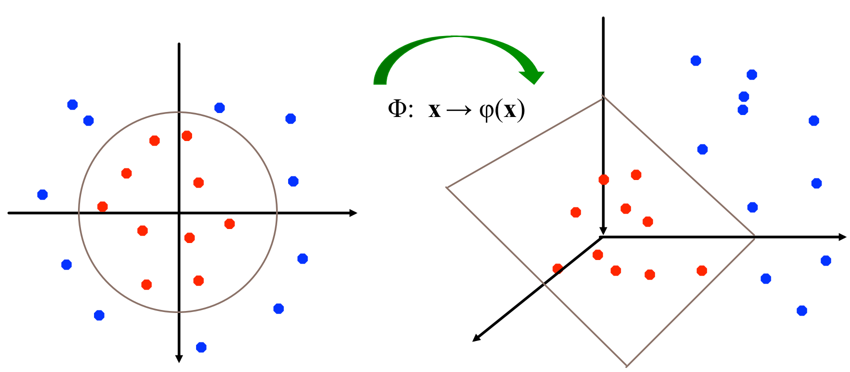
- Multi-class SVMs
- Transductive SVMs

Linear SVMs Summary

- The classifier is a decision boundary (separating hyperplane)
- Most "important" training points are support vectors which define the hyperplane
- Quadratic optimization algorithms can identify which training points are support vectors (vectors with non-zero Lagrange multipliers)
- In the dual formation and in classifying an example, the training points appear only inside inner products

Non-linear SVMs

- 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 an inner product between vectors $x_i^T x_j$

$$g(x_i) = \text{sign}\left(\sum_i \alpha_i y^{(i)} x^{(i)} x + b\right)$$

- If every example is mapped into a high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$ then the inner product becomes:

$$g(x_i) = \text{sign}\left(\sum_i \alpha_i y^{(i)} \varphi(x^{(i)})^\mathsf{T} \varphi(x) + b\right)$$

- A kernel function is some function that corresponds to a dot product in some transformed feature space:

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

The "Kernel" trick

- The kernel K may be cheaper to compute then the transformation φ
  - Implictly do the transformation

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^n x_i z_i\right)\left(\sum_{j=1}^n x_i z_i\right) \\ &= \sum_{i=1}^n \sum_{j=1}^n x_i x_j z_i z_j \\ &= \sum_{i,j=1}^n (x_i x_j)(z_i z_j) \end{aligned}$$

*

Kernels

Why use kernels?

- Make non-separable problem separable.
- Map data into better representational space

Common kernels

- Linear
- Polynomial $\mathbf{K(x,z) = (1+x^Tz)^d}$
- Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2}$$

Summary

- Support Vector Machines (SVMs)
  - Choose hyperplane based on support vectors
  - Support vectors are critical points close to the decision boundary
  - Often among the best performing classifiers

*