TIM 245 Lecture 11 (5/10/17)

Agenda

1) Support Vector Machines

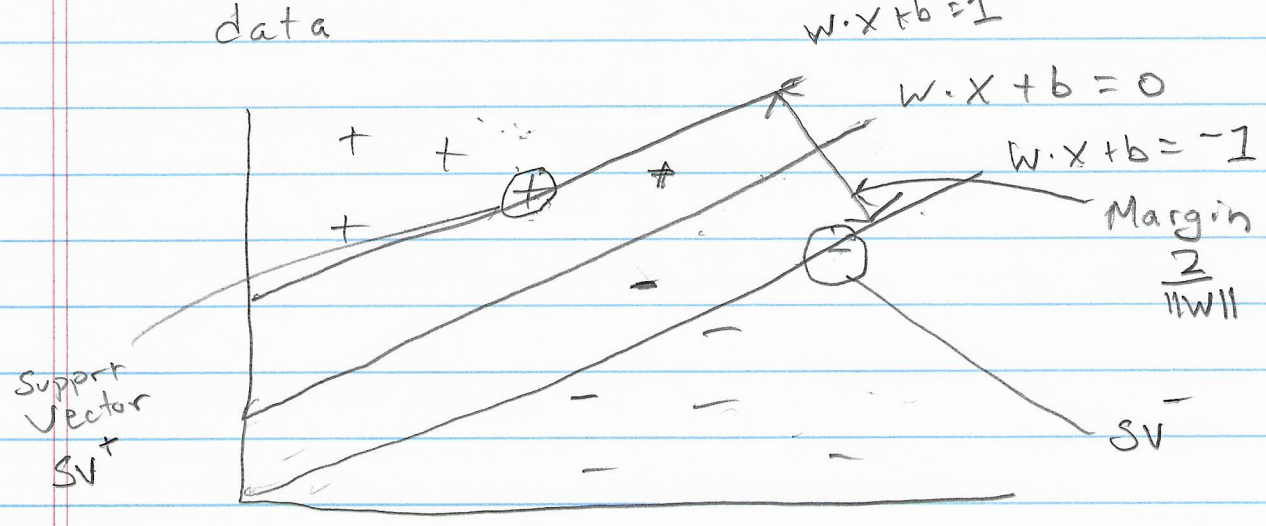
2) Kernel trick

3) Neural Networks

4) Midterm exam

(1) Support Vector Machines

Given a binary classification problem $y = \{+1, -1\}$ with linearly seperable data



The support vector machine algorithm finds the hyperplane $w \cdot x + b = 0$ that maximizes the margin between the two classes.

Margin = distance between $w \cdot x + b = 1$ and $w \cdot x + b = -1$

$$= \frac{|w \cdot sv^{+1}| + |w \cdot sv^{-1}|}{||w||}$$

$$= \frac{2}{||w||}$$

Maximizing the margin $\frac{2}{\|w\|}$ is equivalent to minimizing $\frac{1}{2}\|w\|^2$

Solve the optimization problem

Minimize $\frac{1}{2}\|w\|^2$

Subject to

$$w \cdot x + b \geq 1 \quad \text{if } y = +1$$
$$w \cdot x + b \leq -1 \quad \text{if } y = -1$$

Solve the convex optimization problem

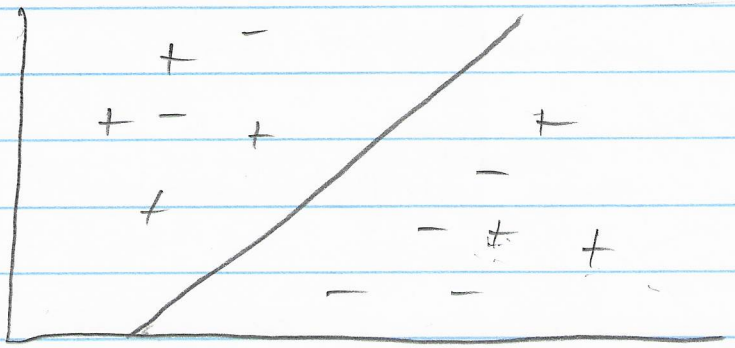
Lagrange Multipliers (closed form)

Gradient Descent

New instances can be classified using the weight vector w

$$\hat{y} = \begin{cases} +1 & \text{if } w \cdot x + b > 0 \\ -1 & \text{if } w \cdot x + b < 0 \end{cases}$$

How do handle cases when the data is not linearly seperable?



Soft Constraint with a penalty for misclassified instances in the training data set

Penalty : Hinge loss function

$$L(x, y) = \begin{cases} \max(1 - w \cdot x + b, 0) \text{ if } y = +1 \\ \max(1 + w \cdot x + b, 0) \text{ if } y = -1 \end{cases}$$

New optimization problem with a slack variable ξ_i to enforce the hinge loss penalty for (x_i, y_i)

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

Subject to

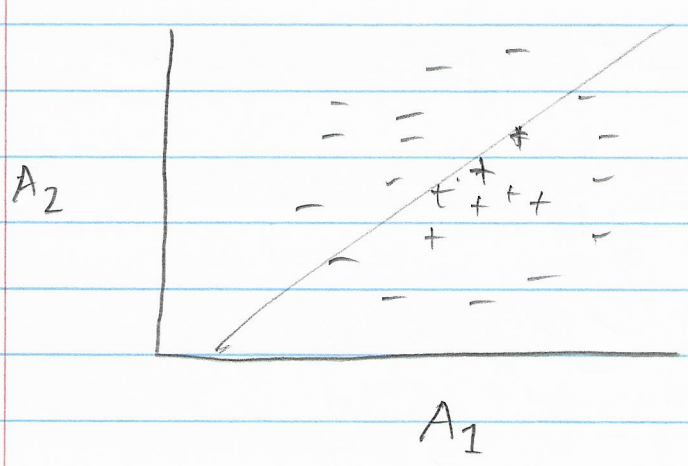
$$w \cdot x + b \geq 1 - \xi_i \quad \text{if } y = +1$$
$$w \cdot x + b \leq -1 + \xi \quad \text{if } y = -1$$
$$\xi_i \geq 0$$

C controls the balance between the conflicting objectives of maximizing the margin and separating the data cleanly (underfitting vs overfitting)

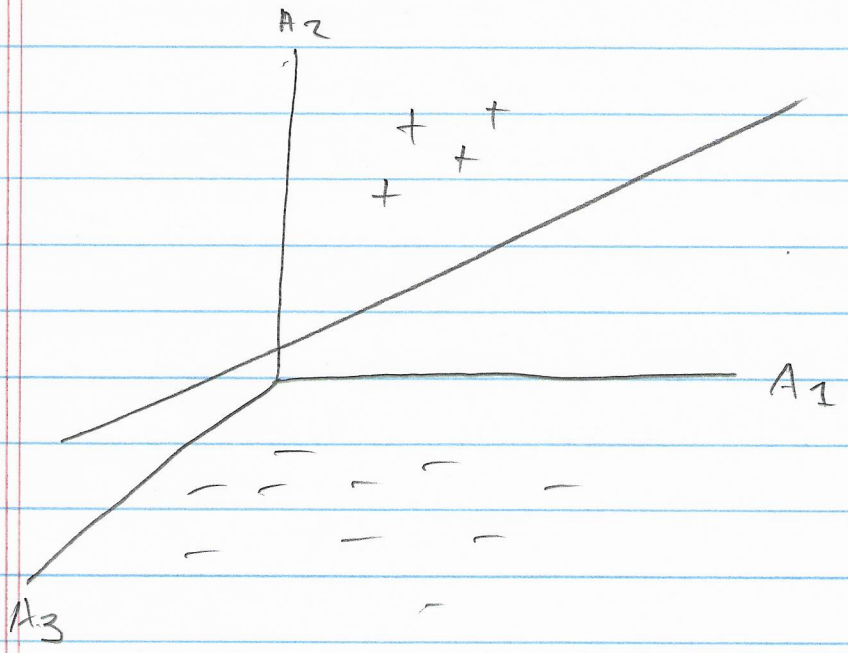
Typically C is set to 1. Increasing C will drive overfitting. Decreasing C will drive underfitting.

② Kernel Trick

What if we have data that is non-linearly separable?



Transform into a higher dimensional space



and find the separating hyperplane

Example

$$\phi(A_i, A_j) = (A_i, A_j^2, (A_i + A_j)^2)$$

All data needs to be transformed into the higher dimensional space.

Becomes extremely computationally expensive to solve the SVM optimization problem.

Kernel Function: Computes the dot product in the higher dimensional space without having to transform the data

$$K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j)$$

Popular Kernels

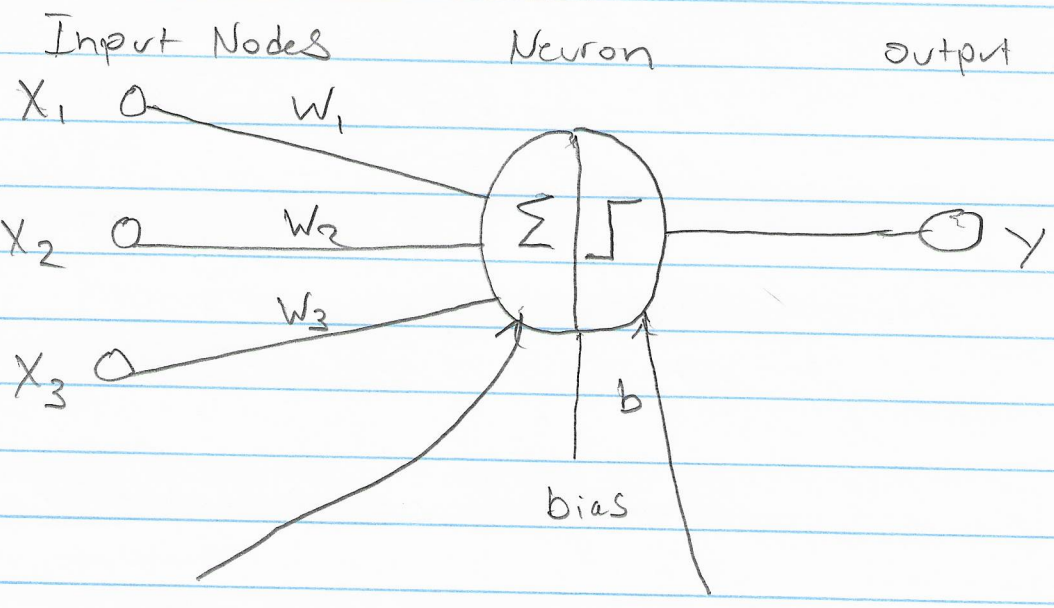
Polynomial Kernel: $K(X_i, X_j) = (X_i \cdot X_j + 1)^h$

Radial Bias Kernel: $\exp(-\|X_i - X_j\|) / 2\sigma^2$

Sigmoid Kernel: $K(X_i, X_j) = \tanh((x_i \cdot x_j) + r)$

③ Neural Networks

Simple Neural Network: Perceptron



$$\sum_{i=1}^{3} w_i \cdot x_i$$

$$\hat{y} = \begin{cases} +1 & \text{if } \sum_{i=1}^{3} w_i \cdot x_i - b > 0 \\ -1 & \text{if } \sum_{i=1}^{3} w_i \cdot x - b < 0 \end{cases}$$

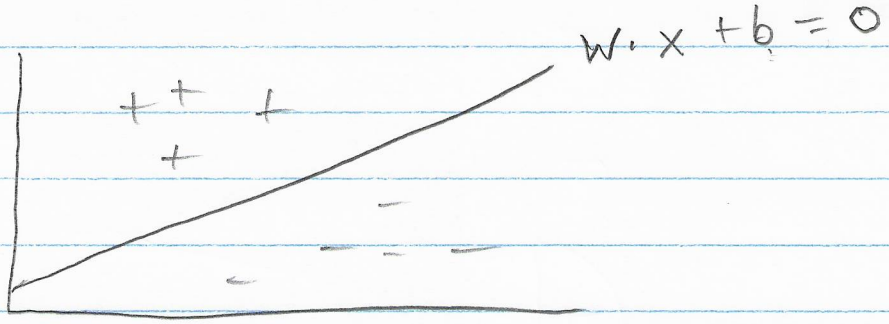
Learn the weights that minimize the sum of squared error

$$SSE(\theta) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

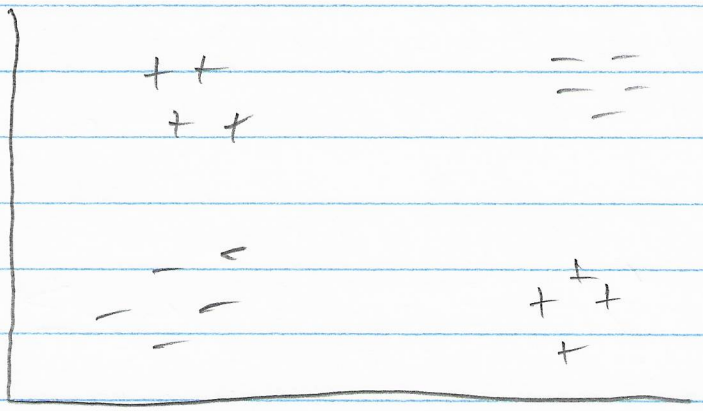
Where $\theta = \{w, b\}$

Solved using gradient descent

Perceptron is equivalent to a linear hyperplane



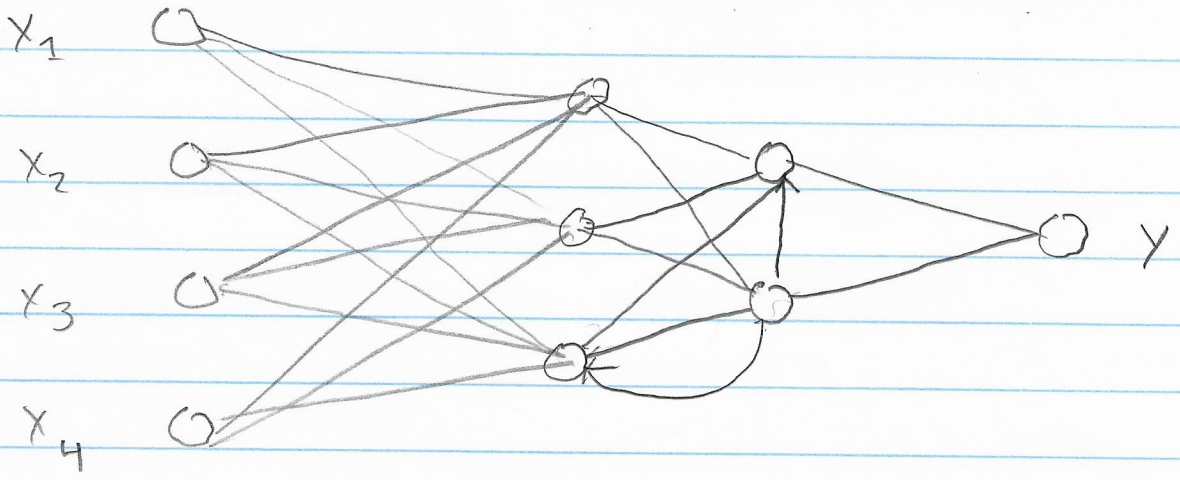
How do we extend our classifier to non-linear problems?



Add more neurons to the network

Multilayer Neural Network

Input Layer Hidden Layer output layer



Two different kinds of networks

1) <u>Feed forward</u> : nodes are only connected to nodes in the next layer (no loops)

2) <u>Recurrent</u> : nodes can be connected in the same layer or in previous layers (loops allowed)