CSC345/M45:
Big Data & Machine Learning
(support vector machine)

Prof. Xianghua Xie

x.xie@swansea.ac.uk
http://csvision.swan.ac.uk
224 Computational Foundry, Bay Campus

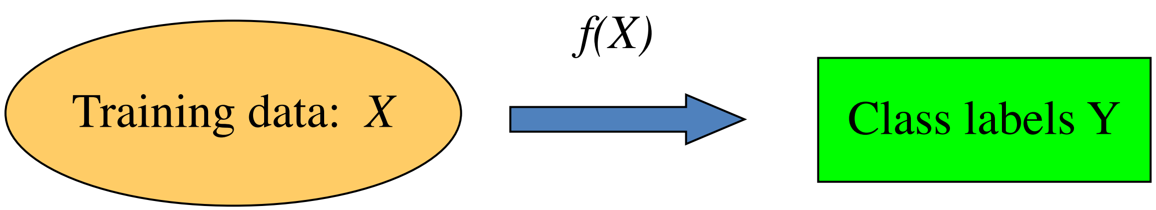
Clustering vs. Classification

- Clustering: unsupervised learning
  - Class labels of the data are unknown
  - Given data, the task is to establish the existence of classes or clusters in the data

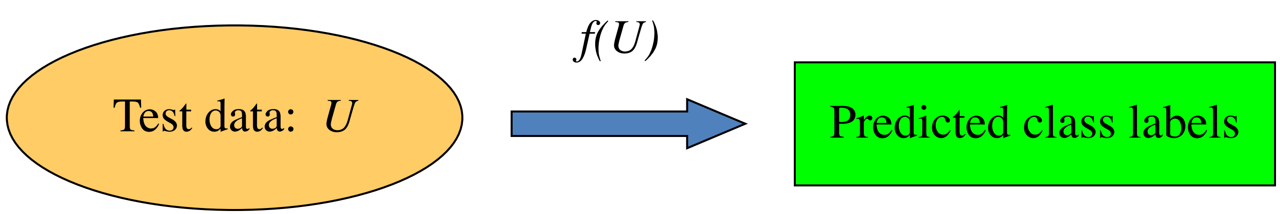
- Classification: supervised learning
  - Supervision: data (observations) are labelled with pre-defined classes
  - The input data (training set) consists of multiple records, each of which has multiple attributes or features
  - Given training data, the task is
    - to develop an accurate description or model for each class using the features
    - and to predict categorical class label for unseen data (test data)

Classification

- Supervised learning
  - Training data (X_i, Y_i), X_i is typically a feature vector and Y_i is the corresponding class label
  - The task of training is to find a good mapping function f
  - The derived function is then evaluated on test data (unseen data)



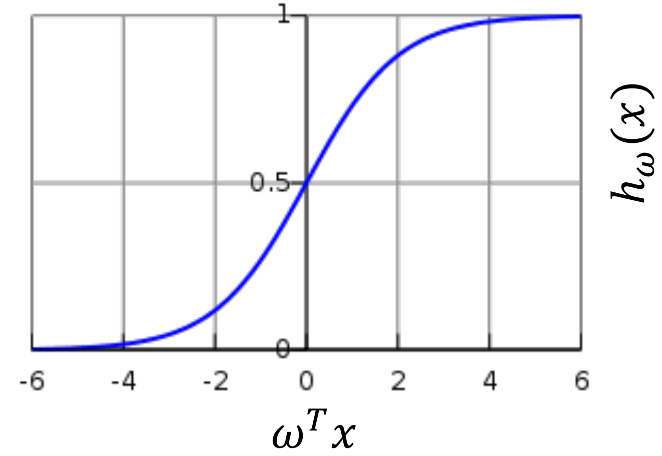
A classifier, a mapping, a hypothesis



From Logistic Regression to SVM

- Logistic regression cost function:

$$E(\omega) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log h_\omega(x_i) + (1-y_i)\log(1-h_\omega(x_i))] + \frac{\lambda}{2N}\omega^T\omega$$

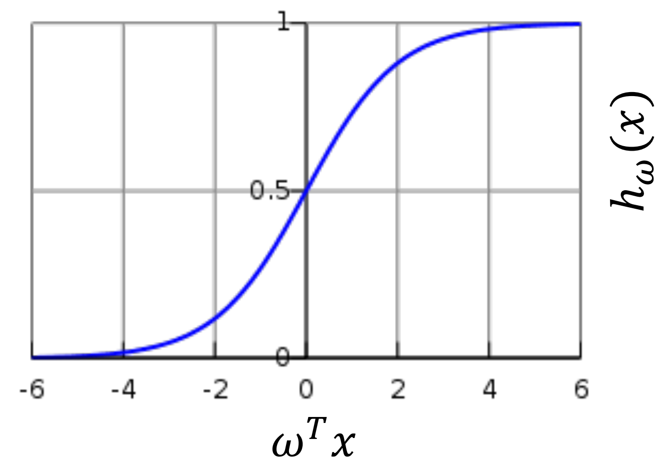


$$h_\omega(x) = \frac{1}{1+e^{-\omega^T x}}$$

From Logistic Regression to SVM

- Logistic regression cost function:

$$E(\omega) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log h_\omega(x_i) + (1-y_i)\log(1-h_\omega(x_i))\right] + \frac{\lambda}{2N}\omega^T\omega$$



- If y=1, we want $h_\omega(x) \approx 1$, $\omega^T x \gg 0$
- If y=0, we want $h_\omega(x) \approx 0$, $\omega^T x < 0$

$$h_\omega(x) = \frac{1}{1+e^{-\omega^T x}}$$

From Logistic Regression to SVM

- Logistic regression cost function:

$$E(\omega) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log h_\omega(x_i) + (1-y_i)\log(1-h_\omega(x_i))] + \frac{\lambda}{2N}\omega^T\omega$$

$$\min_{\omega} -\frac{1}{N}\sum_{i=1}^{N}[y_i \log h_\omega(x_i) + (1-y_i)\log(1-h_\omega(x_i))] + \frac{\lambda}{2N}\omega^T\omega$$

$$\min_{\omega} \frac{1}{N}\sum_{i=1}^{N}[y_i\,(-\log h_\omega(x_i)) + (1-y_i)(-\log(1-h_\omega(x_i)))] + \frac{\lambda}{2N}\omega^T\omega$$

$$\min_{\omega} \sum_{i=1}^{N}[y_i\,(-\log h_\omega(x_i)) + (1-y_i)(-\log(1-h_\omega(x_i)))] + \frac{\lambda}{2}\omega^T\omega$$

$$\min_{\omega}\ C\sum_{i=1}^{N}[y_i\,(-\log h_\omega(x_i)) + (1-y_i)(-\log(1-h_\omega(x_i)))] + \frac{1}{2}\omega^T\omega$$

$$\min_{\omega}\ C\sum_{i=1}^{N}[y_i \mathrm{cost}_1(\omega^T x_i) + (1-y_i)\mathrm{cost}_0(\omega^T x_i)] + \frac{1}{2}\omega^T\omega$$

From Logistic Regression to SVM

- Logistic regression cost function:

$$\min_{\omega} \; C \sum_{i=1}^{N} [y_i \text{cost}_1 (\omega^T x_i) + (1 - y_i) \text{cost}_0 (\omega^T x_i)] + \frac{1}{2} \omega^T \omega$$

$$\begin{cases} \text{cost}_1(\omega^T x_i) = -\log h_\omega(x_i) & \text{if } y = 1 \\ \text{cost}_0(\omega^T x_i) = -\log(1 - h_\omega(x_i)) & \text{if } y = 0 \end{cases}$$

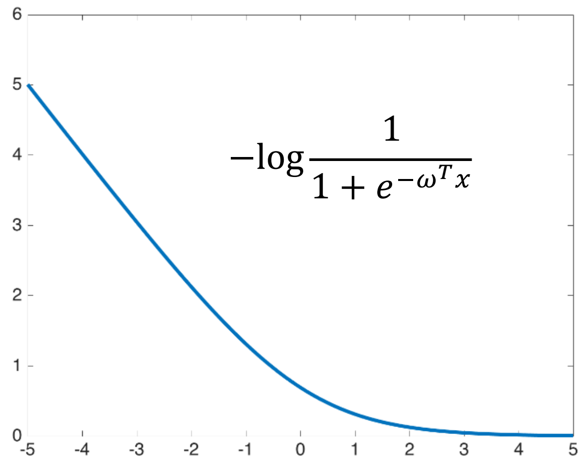
From Logistic Regression to SVM

- Logistic regression cost function:

$$\min_{\omega} \; C \sum_{i=1}^{N} [y_i \text{cost}_1 (\omega^T x_i) + (1 - y_i) \text{cost}_0 (\omega^T x_i)] + \frac{1}{2} \omega^T \omega$$

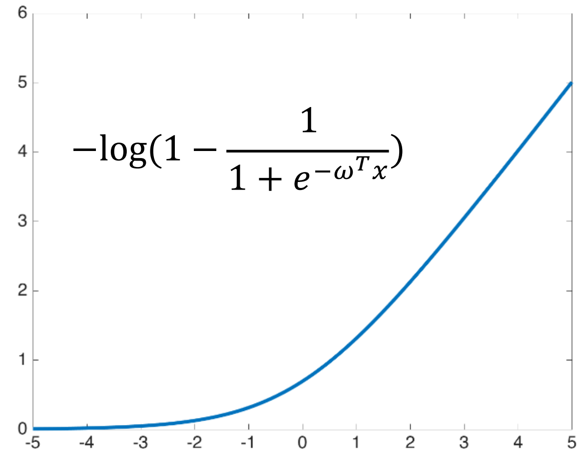
$$\begin{cases} \text{cost}_1(\omega^T x_i) = -\log h_\omega(x_i) & \text{if } y = 1 \\ \text{cost}_0(\omega^T x_i) = -\log(1 - h_\omega(x_i)) & \text{if } y = 0 \end{cases}$$

If y=1, we want $h_\omega(x) \approx 1$, $\omega^T x \gg 0$



$\omega^T x$

If y=0, we want $h_\omega(x) \approx 0$, $\omega^T x \ll 0$



$\omega^T x$

From Logistic Regression to SVM

- SVM cost function (linear case):

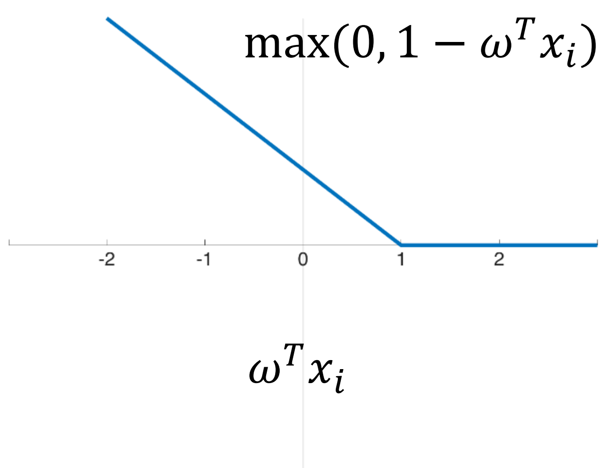
$$\min_{\omega}\ C\sum_{i=1}^{N}[y_i \mathrm{cost}_1\,(\omega^T x_i) + (1 - y_i)\mathrm{cost}_0(\omega^T x_i)] + \frac{1}{2}\omega^T \omega$$

From Logistic Regression to SVM

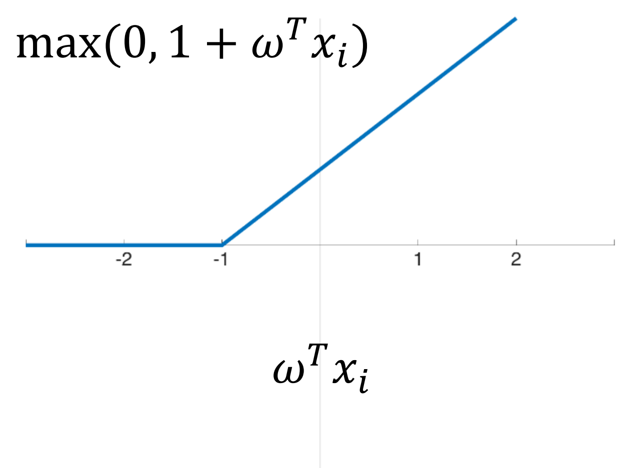
- SVM cost function (linear case):

$$\min_{\omega} \; C \sum_{i=1}^{N} [y_i \mathrm{cost}_1 \,(\omega^T x_i) + (1 - y_i)\mathrm{cost}_0(\omega^T x_i)] + \frac{1}{2}\omega^T \omega$$

If y=1, we want $\omega^T x \geq 1$



If y=0, we want $\omega^T x \leq -1$



From Logistic Regression to SVM

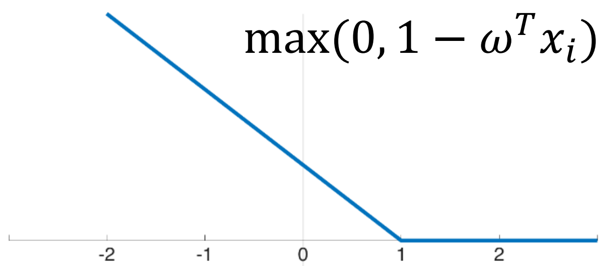
- SVM cost function (linear case):

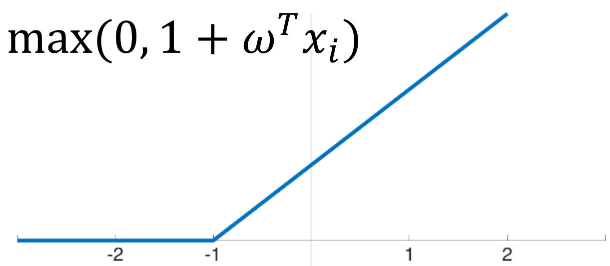
$$\min_{\omega} \; C \sum_{i=1}^{N} [y_i \mathrm{cost}_1 \, (\omega^T x_i) + (1 - y_i) \mathrm{cost}_0(\omega^T x_i)] + \frac{1}{2} \omega^T \omega$$

$$\begin{cases} \mathrm{cost}_1(\omega^T x_i) = \max(0, 1 - \omega^T x_i) & \text{if } y = 1 \\ \mathrm{cost}_0(\omega^T x_i) = \max(0, 1 + \omega^T x_i) & \text{if } y = 0 \end{cases}$$

If y=1, we want $\omega^T x \geq 1$

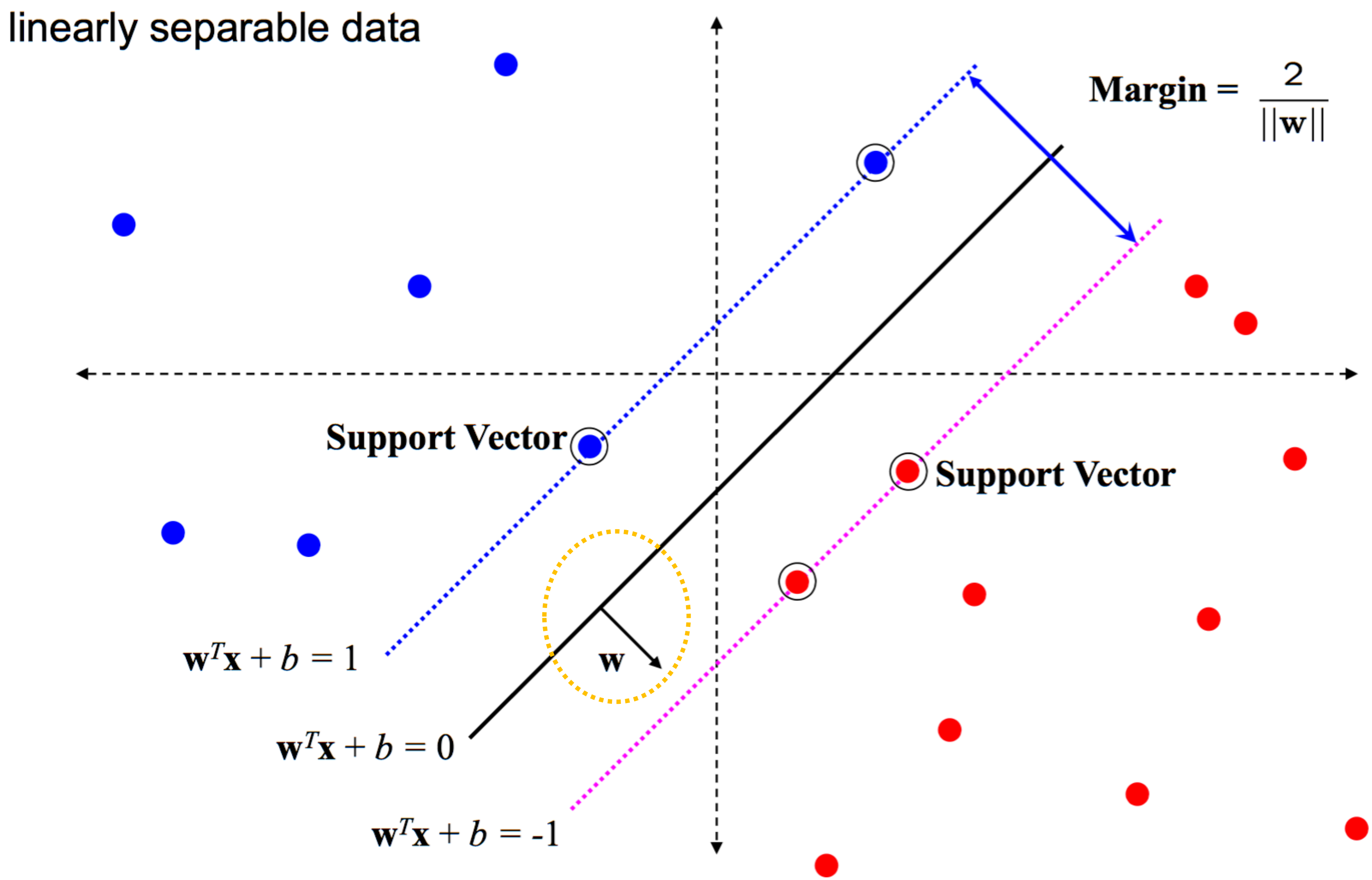
If y=0, we want $\omega^T x \leq -1$





Enforces a larger margin!

SVM Decision Boundary: large margin classifier



SVM Decision Boundary

- Recap: cost function

$$\min_{\omega} \; C \sum_{i=1}^{N} [y_i \mathrm{cost}_1 \,(\omega^T x_i) + (1 - y_i) \mathrm{cost}_0(\omega^T x_i)] + \frac{1}{2} \omega^T \omega$$

- Equivalent to:

$$\min_{\omega} \frac{1}{2} \omega^T \omega = \min_{\omega} \frac{1}{2} \sum_{i=1}^{N} \omega_i^2$$

$$s.t. \qquad \omega^T x_i \geq 1, \qquad if \; y_i = 1$$

$$\omega^T x_i \leq -1, \qquad if \; y_i = 0$$

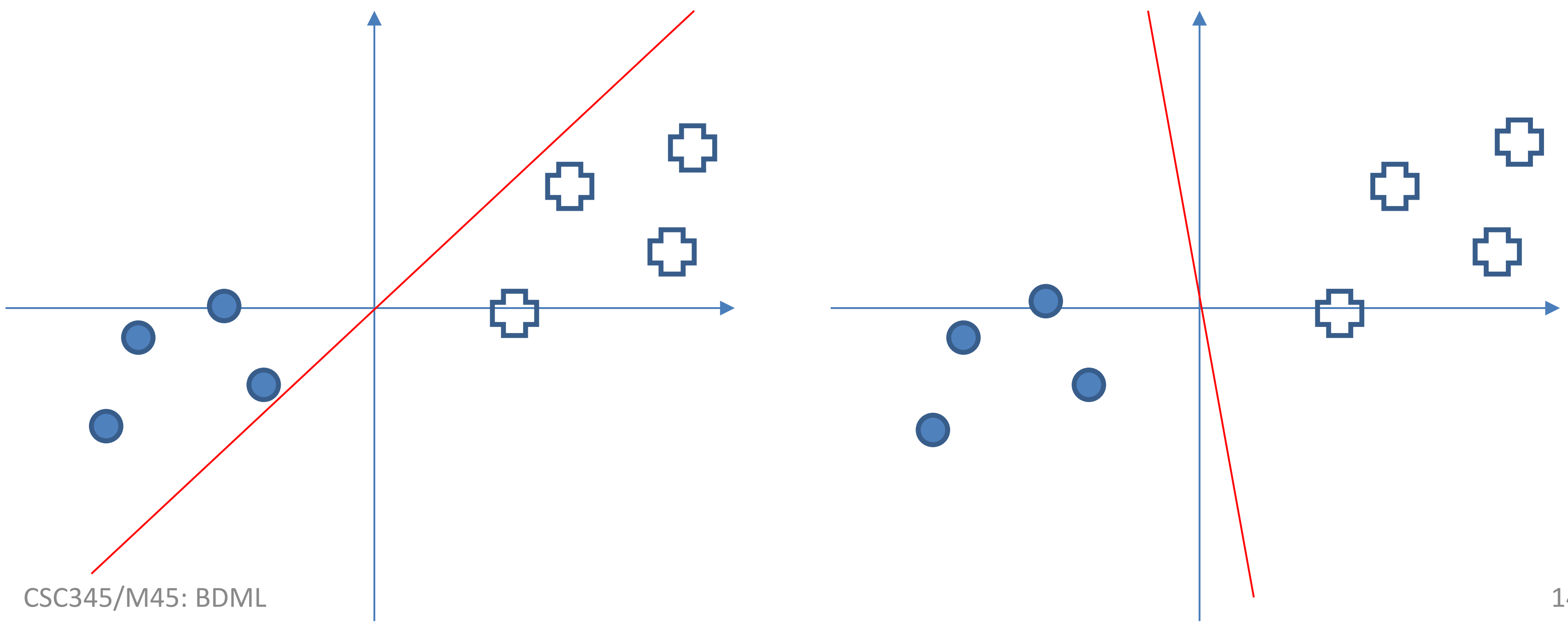
SVM Decision Boundary

$$\min_{\omega} \frac{1}{2} \omega^T \omega = \min_{\omega} \frac{1}{2} \sum_{i=1}^{N} \omega_i^2$$

$$s.t. \qquad \omega^T x_i \geq 1, \qquad if\ y_i = 1$$

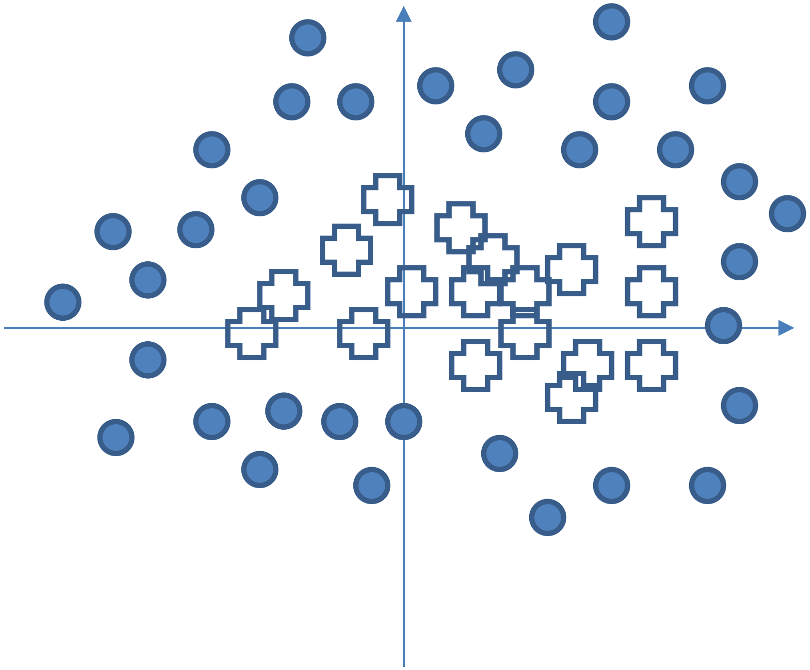
$$\omega^T x_i \leq -1, \qquad if\ y_i = 0$$

Which one of the decision boundaries is better?



Non-linear Decision Boundary

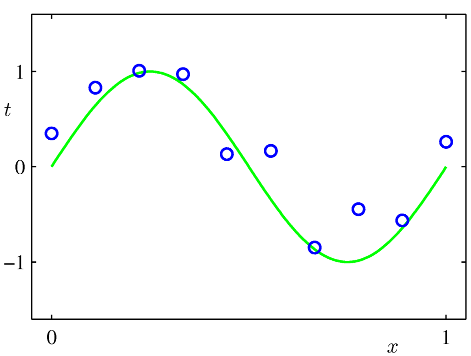
- How to represent the decision boundary?
  - Use the linear regression technique; but note the data is nonlinear
    - Use polynomials
    - Use kernel functions



Non-linear Decision Boundary

- How to represent the decision boundary?
  - Use the linear regression technique; but note the data is nonlinear
    - ~~Use polynomials~~
    - Use kernel functions

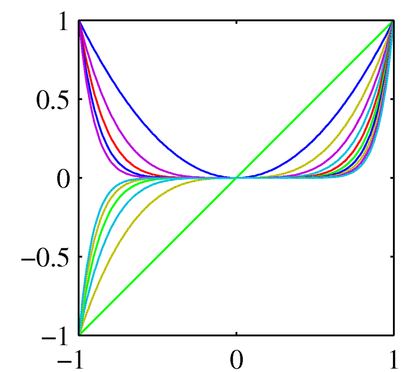
Recap: linear regression



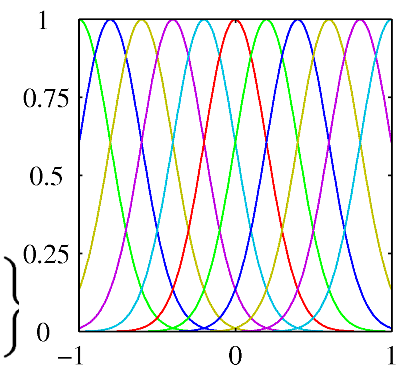
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

$$\phi_j(x) = x^j.$$



$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$



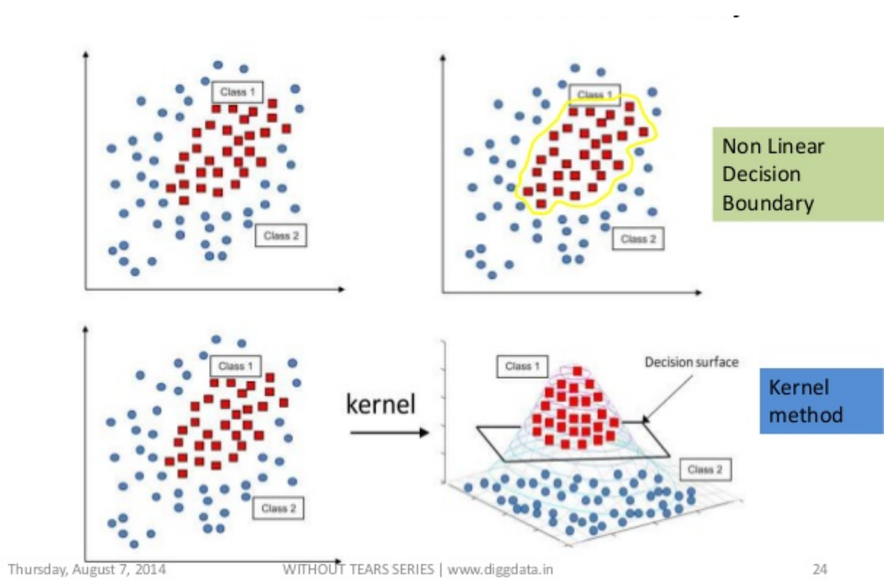
Decision Boundary: from linear to non-linear

- Linear case:

$$\min_{\omega} \; C \sum_{i=1}^{N} [y_i \mathrm{cost}_1 \left(\omega^T x_i\right) + (1 - y_i) \mathrm{cost}_0 (\omega^T x_i)] + \frac{1}{2} \omega^T \omega$$

- Non-linear case:

$$\min_{\omega} \; C \sum_{i=1}^{N} [y_i \mathrm{cost}_1 \left(\omega^T \phi(x_i)\right) + (1 - y_i) \mathrm{cost}_0 (\omega^T \phi(x_i))] + \frac{1}{2} \omega^T \omega$$



SVM Parameters

- C
  - Large C: lower bias, higher variance
  - Small C: higher bias, lower variance

$$\min_{\omega} \; C \sum_{i=1}^{N} [y_i \mathrm{cost}_1 (\omega^T \phi(x_i)) + (1 - y_i) \mathrm{cost}_0 (\omega^T \phi(x_i))] + \frac{1}{2} \omega^T \omega$$

- σ (kernel width)
  - Large σ :
    - kernel varies more smoothly
    - Higher bias, lower variance
  - small σ:
    - Kernel is sharper
    - Lower bias, higher variance

Classification

- Evaluation
  - Measure performance on independent blind test data
  - LOOCV: leave one out cross validation
    - Take one sample from the dataset as the test data and the remaining for training
    - Rotate the test data across the whole dataset
    - Performance is measured as the average across the rounds
  - K-fold cross validation:
    - Divide the dataset into K even parts
    - K-1 parts used for training, and one for testing
    - Rotating the test set
    - Performance is measured as the average across the rounds
  - 2-fold cross validation
    - Simplest variation of K-fold
    - Training and testing has equal number of samples

Classification

- Evaluation
  - Positive: data sample that belongs to the class of interest
  - Negative: data sample that does not belong to the class

  - True positive: a positive sample correctly identified as positive
  - False positive: a negative sample incorrectly classified as positive

  - True negative: a negative sample correctly identified as negative
  - False negative: a positive sample incorrectly classified as negative

  - True positive rate: sensitivity
    - Percentage of correct prediction of positive samples $\frac{TP}{TP + FN}$
  - True negative rate: specificity
    - Percentage of correct prediction of negative samples $\frac{TN}{TN + FP}$

Classification

- Evaluation
  - Overall accuracy
    - Percentage of correct prediction

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{P + N}$$

  - Error rate
    - Percentage of incorrect prediction
  - 2-class confusion matrix

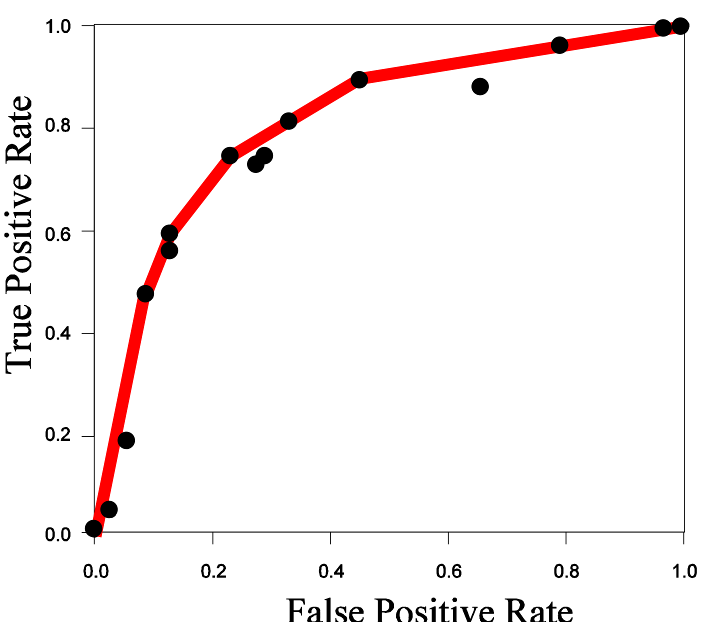
True class	Predicted class	
	positive	negative
positive	TP	?
negative	?	TN

ROC Curve

- Receiver operating characteristic (ROC), or ROC curve
  - a graphical plot that illustrates the performance of a binary classifier as its decision boundary is varied
  - X axis: false positive rate
  - Y axis: true positive rate
  - Each classifier represented by a point in ROC space corresponding to its (FP,TP) pair

$$\frac{TP}{TP+FN}$$

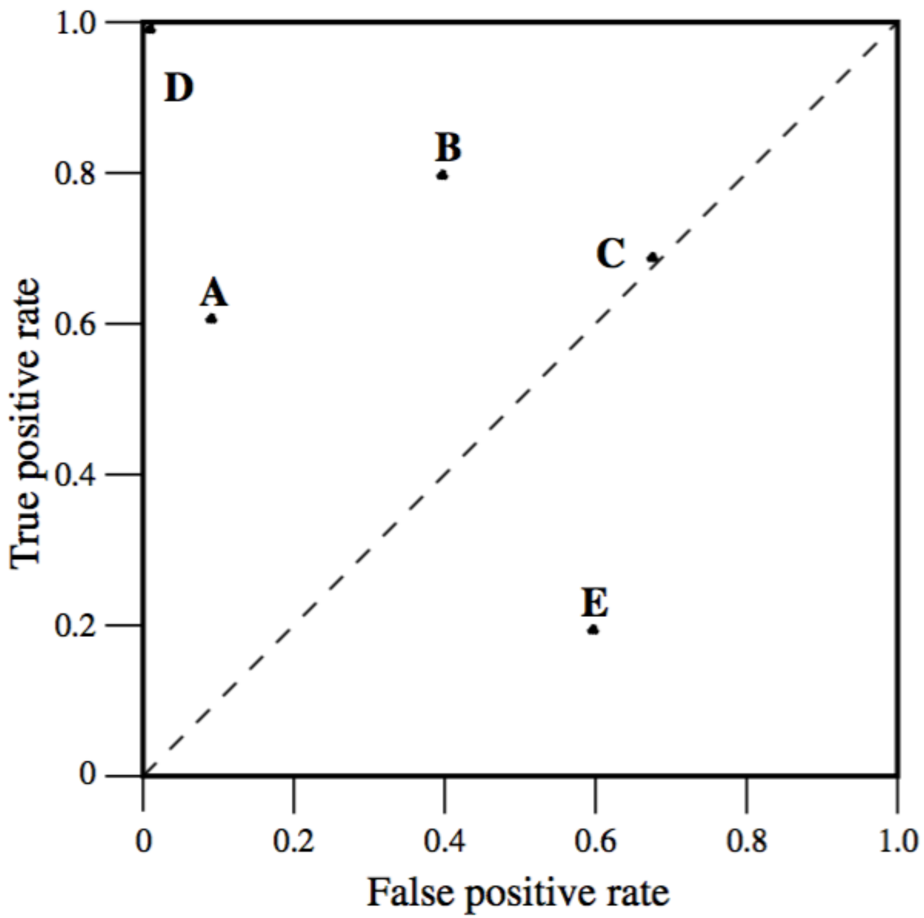
- Varying parameters
- Varying class distribution



$$\frac{FP}{FP+TN}$$

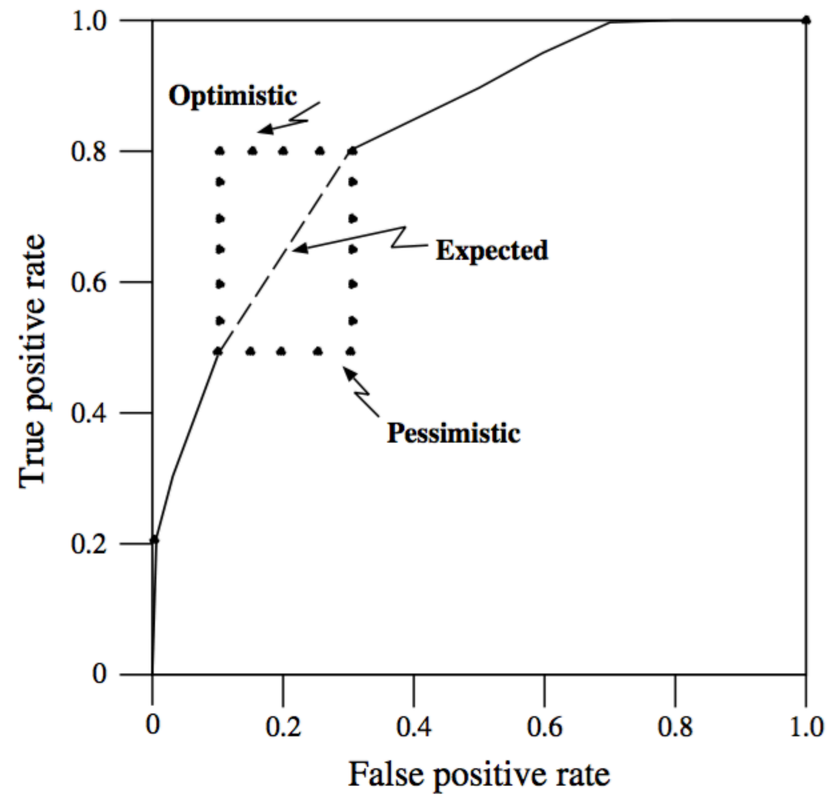
ROC Curve

- A basic ROC curve showing 5 discrete classifiers
  - Which one is the best classifier?
  - What is the dashed line?



Creating an ROC Curve

- A classifier produces a single ROC point.
- If the classifier has a “sensitivity” parameter, varying it produces a series of ROC points (confusion matrices).
- Alternatively, if the classifier is produced by a learning algorithm, a series of ROC points can be generated by varying the class ratio in the training set.
  - Class ratio (class distribution): number of positives samples versus number of negative samples



Example ROC Curve

- Learner L1 dominates as L2’s ROC curve is beneath L1’s curve
  - i.e. L1 is better than L2 for all possible costs and class distributions
- L2 & L3: neither dominates
  - Perhaps switch the classifiers at the intersection point on ROC graph