Midterm Review

Nan Li

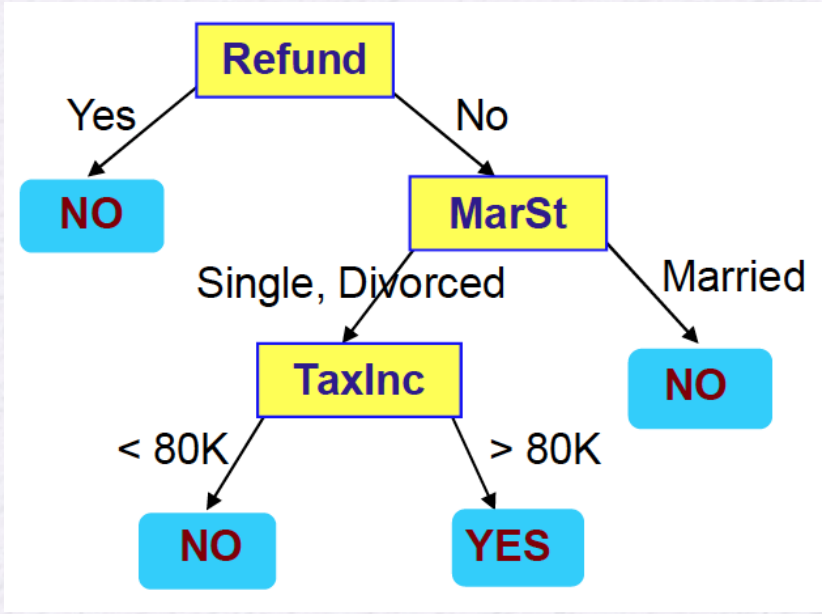
2011.10.24

Things to Think about…

- How to train it?
  - What is the objective function?
  - Which algorithm could be used?
    - E.g. Gradient descent, EM
  - Optimality?
- What are the assumptions?
  - Linear boundary?
- What are the pros and cons comparing with other algorithms?

Decision Tree

- Input: a vector of attributes
- Output: a class, e.g. cheating or not
- Express any function of the input variables
- Overfitting
- Prefer compact tree
- Learning:
  - Greedily select the most informative attribute
  - Information gain

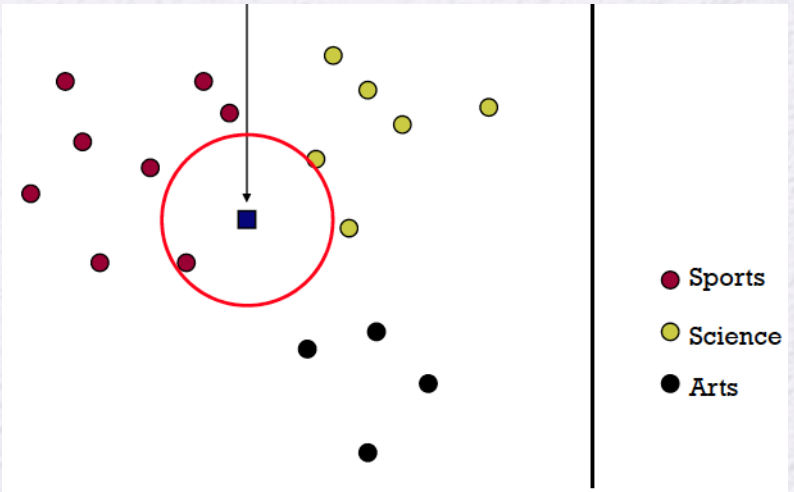


$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

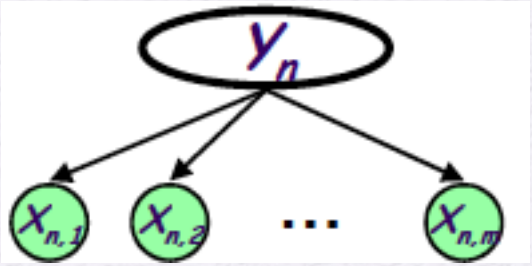
Parent Node, p is split into k partitions; n_i is number of records in partition i

K-Nearest Neighbour

- Learning is just storing all training examples
- Bayes classifier is the best classifier which minimizes the probability of classification error
- The error-rate of 1NN is less than twice the Bayes error – Error rate of classifier knowing model of generated data
  - What if the density estimates converge to the true densities?



Naïve Bayes



- Generative
  - Model P(X, Y)
- P(Y|X) = P(X|Y)P(Y)/P(X)
- Conditional Independence Assumption
  - P(X1, X2 … Xn|Y) = P(X1|Y)P(X2|Y)… P(Xn|Y)
- Why do we want to make this assumption?
- Training:
  - MLE
- Connection to logistic regression

Logistic Regression

- Discriminative
  - Directly model P(Y|X)

The condition distribution: a Bernoulli

$$p(y \mid x) = \mu(x)^y (1 - \mu(x))^{1-y}$$

where μ is a logistic function

$$\mu(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- Training:
  - MCLE
  - Gradient ascent
  - Concave -> Global optimum

$$l(\theta) \equiv \ln \prod_i P(y_i|x_i;\theta) = \sum_i \ln P(y_i|x_i;\theta)$$

- Linear decision boundary

Naïve Bayes vs Logistic Regression

- When model assumptions correct
  - NB = LR
- When model assumptions incorrect
  - LR is less biased
- Convergence rate
  - NB order log m (m = # attributes in X)
  - LR order m

Linear Regression

- Assume that Y (target) is a linear function of X (features)

$$\hat{y} = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \ldots + \theta_k x^k$$

  - Can be with non-linear basis functions

- Training
  - Least Mean Square (LMS)

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

    1. Gradient descent, global optimum
    2. Set derivative to zero

$$\theta^* = \left(X^T X\right)^{-1} X^T \vec{y}$$

- Equivalence of LMS and MLE

Ridge Regression vs Lasso

$$\min_{\beta}(\mathbf{X}\beta - \mathbf{Y})^T(\mathbf{X}\beta - \mathbf{Y}) + \lambda \text{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \text{pen}(\beta)$$

Prior belief that β is Gaussian with zero-mean biases solution to "small" β

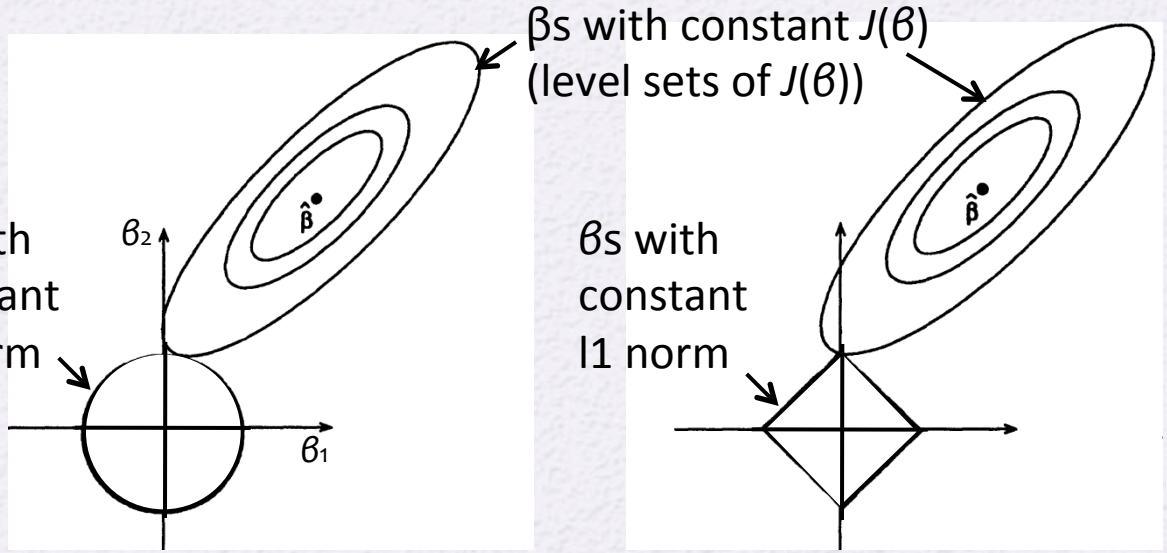
Ridge Regression:

$$\text{pen}(\beta) = \|\beta\|_2^2$$

Lasso:

$$\text{pen}(\beta) = \|\beta\|_1$$

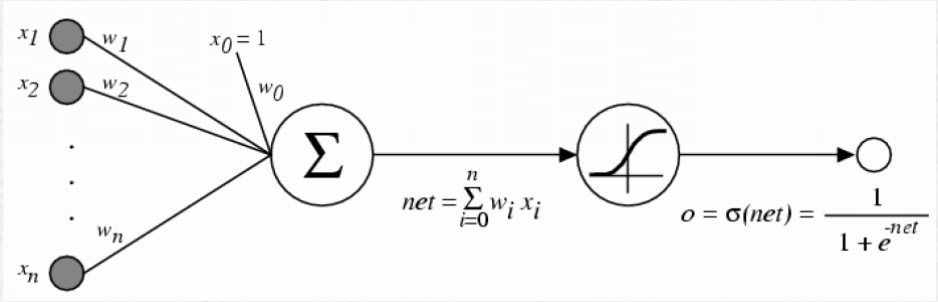
Prior belief that β is Laplace with zero-mean biases solution to "small" β



Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates
Good for high-dimensional problems – don't have to store all coordinates!

Neural Networks

- Perceptron



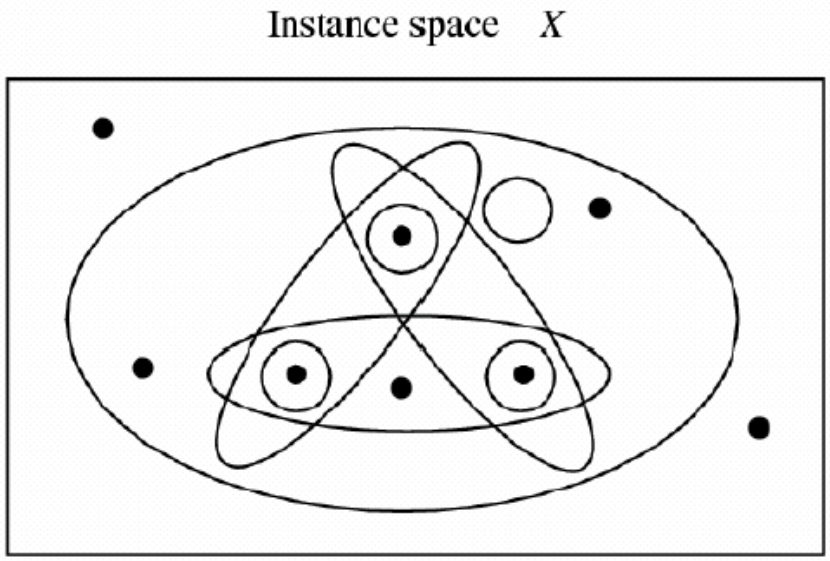
  - Training: Gradient descent
  - Decision boundary?
    - Linear if using sigmoid.
    - Not true in general.

- Neural Networks
  - Highly expressive
  - Training:
    - Backpropagation
    - Minimizing sum of squared training errors
  - May stuck in local optimum

Learning Theory

- A lot of definitions…
- And theorems…

- *Definition*: The **Vapnik-Chervonenkis dimension**, *VC*(*H*), of hypothesis space *H* defined over instance space *X* is the size of the *largest finite subset* of *X* shattered by *H* . If arbitrarily large finite sets of *X* can be shattered by *H* , then $VC(H) \equiv \infty$.



***Definition*:**

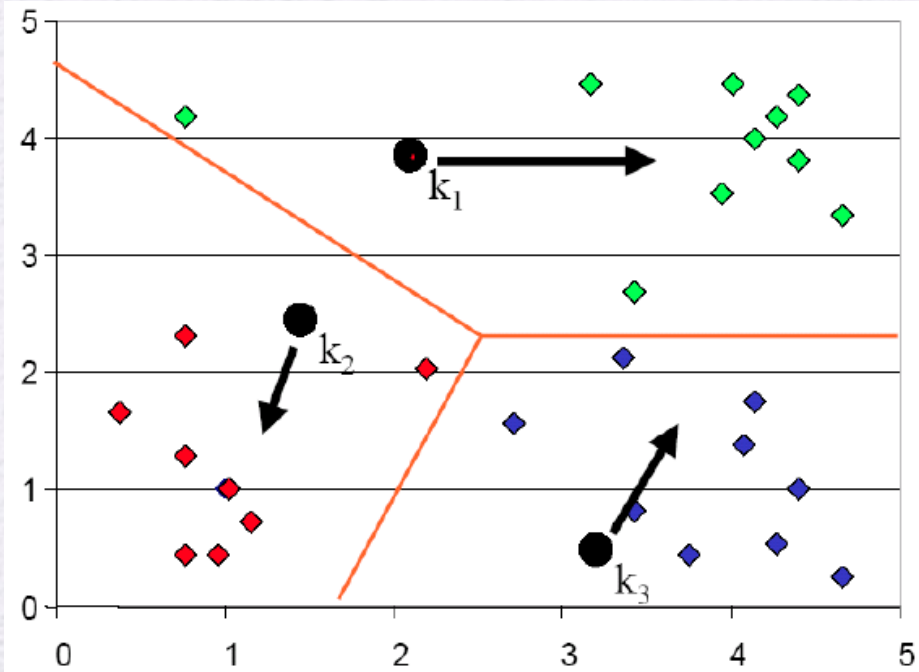
Given a set *S* = {*x*(1), … , *x*(*d*)} of points *x*(*i*)Î X, we say that *H* shatters *S* if *H* can realize any labeling on *S*.

Overfitting and Model Selection

- Training Error vs Testing Error
- Cross Validation
  - Enough to run it once?
- Regularization
  - L1, L2
- Feature Selection
  - Filter: direct feature ranking
  - Wrapper: determine feature based on performance under the learning algorithm
  - Simultaneous learning and feature selection

Clustering

- Unsupervised learning
- Various distance metrics
  - E.g. Euclidean distance, Manhattan distance …
- Hierarchical clustering
  - Bottom-up
  - Top-down
- K-Means
  - Known to converge
  - Sensitive to initial points



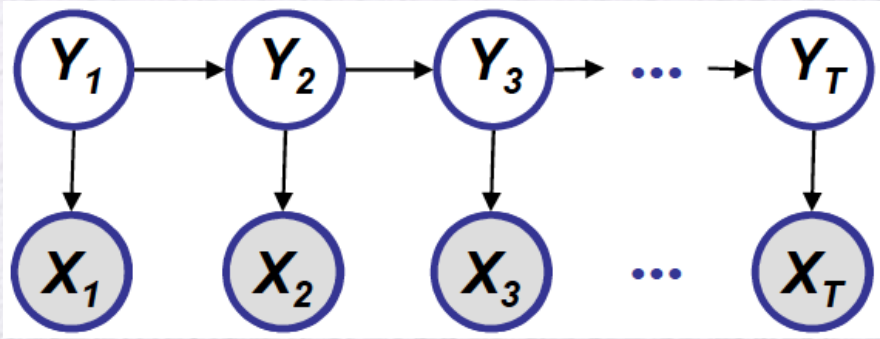
Expectation Maximization

- Why do we need it?
  - Used when there are hidden variables
- Complete log likelihood $\ell_c(\theta; x, z)$
- Incomplete log likelihood $\ell_c(\theta; x)$
- Lower bound $F(q,\theta) \stackrel{\text{def}}{=} \sum_z q(z \mid x) \log \frac{p(x, z \mid \theta)}{q(z \mid x)} \le \ell(\theta; x)$
- Algorithm (May refer to the Bishop book for details)
  - E-step: filling in the latent variables using the best guess
  - M-step: updating the parameters based on this guess
- Local optimum
- A soft version of K-means

Hidden Markov Model

- Assumption
  - How many parameters needed?

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{y}) &= p(x_1 \ldots\ldots x_T, y_1, \ldots\ldots, y_T) \qquad \text{(Joint probability)} \\
&= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T) \\
&= p(y_1)\, P(y_2 \mid y_1) \ldots p(y_T \mid y_{T-1}) \times p(x_1 \mid y_1)\, p(x_2 \mid y_2) \ldots p(x_T \mid y_T)
\end{aligned}
$$



HMM

- Forward $\alpha(y_t^k = 1) = \alpha_t^k \overset{\text{def}}{=} P(x_1, \ldots, x_t, y_t^k = 1)$

$$\alpha_t^k = p(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

- Backward $\beta_t^k = P(x_{t+1}, \ldots, x_T \mid y_t^k = 1)$

$$\beta_t^k = \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

- Viterbi $V_t^k = \max_{\{y_1, \ldots y_{t-1}\}} P(x_1, \ldots, x_{t-1}, y_1, \ldots, y_{t-1}, x_t, y_t^k = 1)$

$$V_t^k = p(x_t \mid y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$$

- Learning
  - Baum-Welch (EM)
  - Please refer to the paper if you are interested in knowing the details

Good Luck ☺