CS412 Final Review
Classification (Chapters 8, 9)

Jia Wang

12/6/2015

Decision tree

- Tree structure
  - **Leaf** nodes: outcome
  - **Non-leaf** nodes: decision attribute
  - **Branches** below non-leaf nodes: predicate on the decision attribute, by which data is divided into disjoint sets
- Construction
  - Starting with a single node (root), which involves set of all data
  - At each node: select best attribute by some **merit measure**
    - E.g., information gain, Gini index
  - Stopping condition: all attributes used by parent nodes, or data have pure classes
- Easily works for both nominal and numerical attributes
  - For numerical attribute x_i, branches can correspond to, e.g., $x_i > 1.5$ and $x_i \leq 1.5$
- Leads to interpretable decision rules
  - Each root-to-path corresponds can be converted into a rule

Naïve Bayes

- Class predicted by $P(H|X) = \frac{P(H)P(X|H)}{P(X)}$

Hypothesis (prediction)

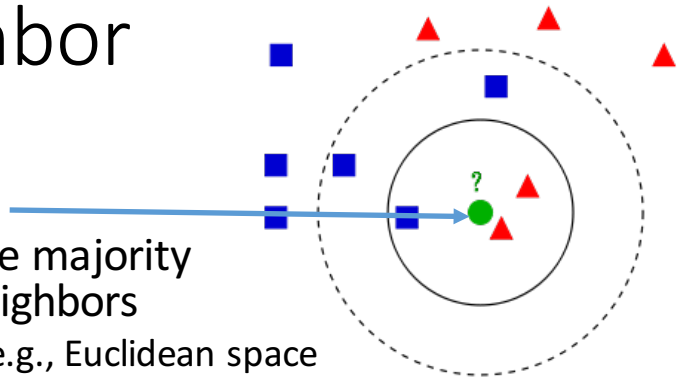
Evidence

Computed by **total probability** formula

$P(X) = P(X|H)P(H)|P(X|\neg H)P(\neg H)$

- Not to confuse $P(H|X)$ with $P(X|H)$
  - The former is **posterior probability** (of class given observations)
  - The latter is **likelihood** (of observations given class)
- Assumption: attribute values X_k **conditionally independent** given class H
  - Key assumption underlying Naïve Bayes
  - $P(X|H) = \prod_k P(X_k|H)$

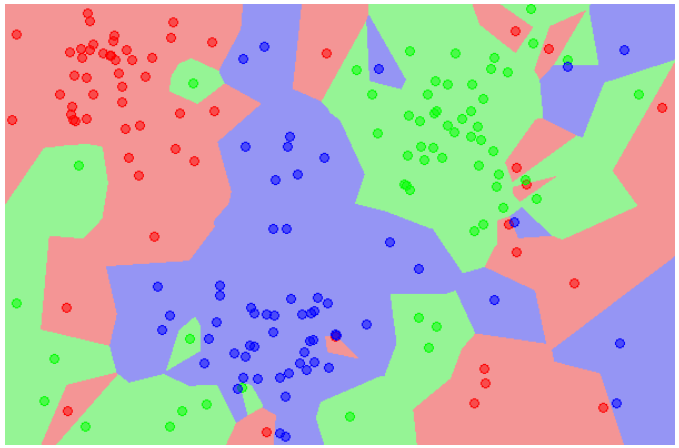
K-Nearest Neighbor



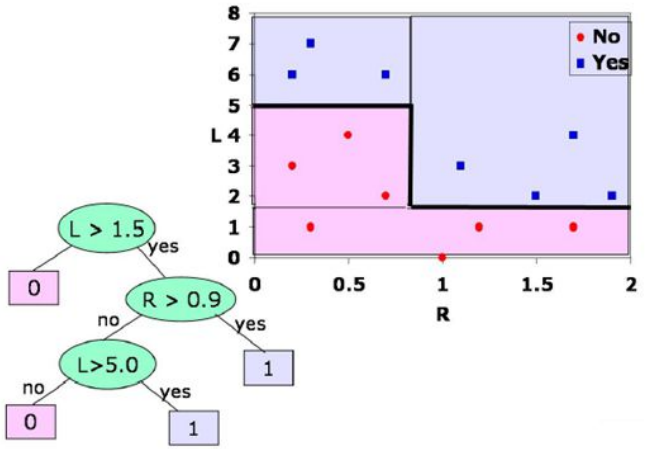
- Given querying point q
  - Its class predicted by the majority class of its k-nearest neighbors
    - Distance measured by, e.g., Euclidean space
    - Optionally, contributions of nearest neighbors weighted by distance
      - The further the neighbor is, the less it contributes to the class of q
- Curse of dimensionality (CoD)
  - In high dimensional space, nearest neighbors far away from q, hence **less representative** of q (premise of kNN classifier broken)
  - CoD also an issue for many other classification methods
- "Lazy" classifier
  - No overhead for training phase; higher cost for prediction

K-Nearest Neighbor

- Decision boundary of classifiers
  - Hypersurfaces that partitions feature space into subsets, one for each class
    - E.g., Curves in 2-D spaces, surfaces for 3-D spaces



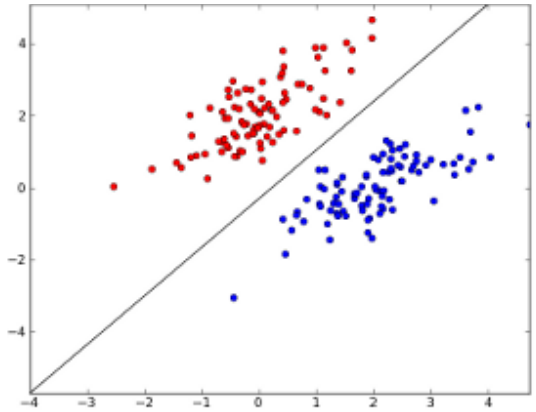
A 1NN example



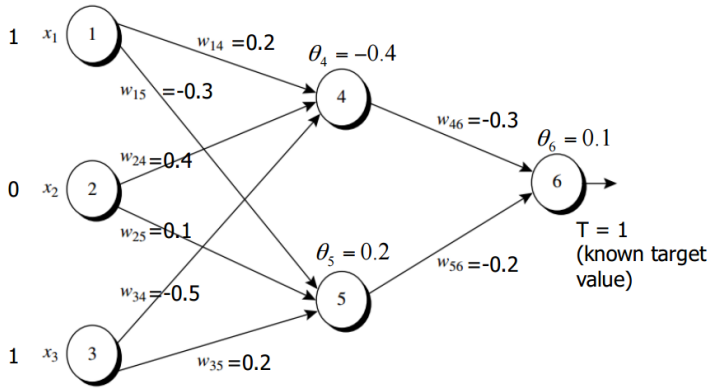
A decision tree example

Perceptron

- Predict class $+1$ if $\boldsymbol{w} \cdot \boldsymbol{x} + b > 0$; otherwise class -1
  - $\boldsymbol{w}$ and b are parameters, $\boldsymbol{x}$ is input vector
  - Parameters w and b updated iteratively for each training tuple
- Linear decision boundary only



Multilayer Neural Network

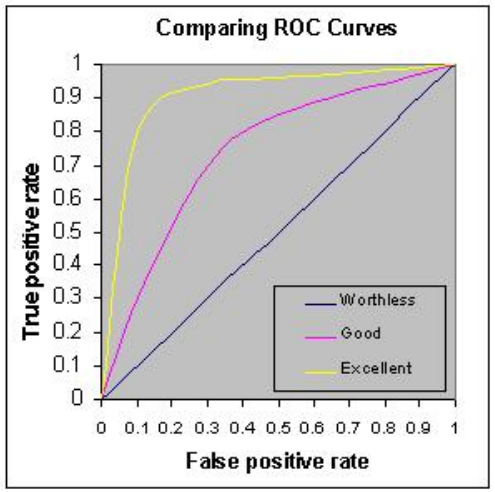


- Topology
  - **Input layer**, **hidden layer(s)**, **output layer**
  - Each layer consists of **units** (neurons)
  - Units linked to some units in the next layer
- Prediction by **feedforwarding**
  - Each neuron takes input as the **biased weighted sum** of outputs from neurons in the previous layer, and calculates output with the **activation function**
    - Examples of activation function: sign, threshold (step), sigmoid, tanh, etc.
- Parameter determined by **backpropagation**
  - Parameters include weights on the connections and biases on the units

Evaluating classifiers

- Training/testing data
  - Model built on training data
  - Evaluate classifier on the testing data
- $precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$
- ROC curve
  - Larger the area under the curve
    → better accuracy

		Actual +	Actual -
Predicted	Y	True positives	False positives
Predicted	N	False negatives	True negatives



AdaBoost

- Ensemble classifier combines a set of weak (base) classifiers h_i
- Algorithm in iterations
  - A data weight distribution $\boldsymbol{w}_i = (w_1, \ldots, w_N)$ in iteration i
  - h_i to minimize the weighted error ϵ_i on the training tuples
    - Weighted error rate: sum of weights on misclassified instances
  - Adjust weight distribution for next iteration
    - Decrease weights on correctly classified instances by factor $\frac{1-\epsilon_i}{\epsilon_i}$
    - Normalize weight distribution: weights sum to 1
- Final classifier has form $h(x) = \prod_i \alpha_i h_i(x)$
  - $\alpha_i = \ln \frac{1-\epsilon_i}{\epsilon_i}$ is the weight of weak classifier h_i