Chapter 6 Classification and Prediction (2)

Outline

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers

Bayesian Classification: Why?

- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem: Basics

- Let **X** be a data sample (*evidence*) with class label unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine P(H|**X**), (*posteriori probability*), the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*), the initial probability, which is independent of **X.**
  - E.g., **A customer** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (*likelihood*), the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X' age is 31..40, medium income

Bayesian Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H*,* P(H|**X**)*,* follows the **Bayes' theorem**

$$P(H\,|\,\mathbf{X}) = \frac{P(\mathbf{X}\,|\,H)P(H)}{P(\mathbf{X})}$$

- Informally, this can be written as

  posteriori = likelihood x prior/evidence
- Predicts X belongs to Ci iff the probability P(Ci|X) is the highest among all the P(C_k|X) for all the k classes

Towards Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal P(C_i|**X**)
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only needs to be maximized $P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$

Derivation of Naïve Bayes Classifier

- **Naïve**:
  - A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If Attribute A_k is categorical, P(x_k|C_i) is the # of tuples in C_i having value x_k for A_k divided by |$C_{i, D}$| (# of tuples of C_i in D)
- If A_k is continous-valued, P(x_k|C_i) is usually computed based on Gaussian distribution with a mean μ and standard deviation σ: probability density function

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and P(x_k|C_i) is

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	_comp
28	high	no	fair	no
25	high	no	excellent	no
35	high	no	fair	yes
45	medium	no	fair	yes
46	low	yes	fair	yes
47	low	yes	excellent	no
35	low	yes	excellent	yes
25	medium	no	fair	no
27	low	yes	fair	yes
59	medium	yes	fair	yes
26	medium	yes	excellent	yes
32	medium	no	excellent	yes
32	high	yes	fair	yes
50	medium	no	excellent	no

Converted Dataset

- Age *
  - •1 : <=30
  - •2 : 31 – 40
  - •3 : >40
- Income
  - •1 : low
  - •2 : medium
  - •3 : high
- Student
  - •0 : no
  - •1 : yes
- Credit_rating
  - •1 : fair
  - •2 : excellent
- Buy_computer (class label)
  - •0 : no
  - •1 : yes

ID	Age	Income	Student	Credit Rating	Buy Computer
1	1	3	0	1	0
2	1	3	0	2	0
3	2	3	0	1	1
4	3	2	0	1	1
5	3	1	1	1	1
6	3	1	1	2	0
7	2	1	1	2	1
8	1	2	0	1	0
9	1	1	1	1	1
10	3	2	1	1	1
11	1	2	1	2	1
12	2	2	0	2	1
13	2	3	1	1	1
14	3	2	0	2	0

Train the Naïve Bayesian Classifier

- $P(C_i)$
  - P(buys_computer = “yes”) = 9/14 = 0.643
  - P(buys_computer = “no”) = 5/14= 0.357

- Compute $P(X|C_i)$ for each class (part of classifiers are listed)
  - P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  - P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6
  - P(age =“31 - 40”| buys_computer = “yes”) …
  - P(age=“31-40” | buys_computer = “no”) …
  - P(age=“>40” | buys_computer=“yes”) …
  - P(age=“>40” | buys_computer=“no”) …

Train the Naïve Bayesian Classifier

- Compute $P(X|C_i)$ for each class (part of classifiers are listed)
  - P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  - P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  - P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  - P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  - P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  - P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

Predict The Class Label

- A new record
  - X = (age <= 30 , income = medium, student = yes, credit_rating = fair)
- Predict the probability
  - **P(X|C$_i$)**
    - P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
    - P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
  - **P(X|C$_i$)*P(C$_i$)**
    - P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028
    - P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007
- Therefore, X belongs to class (“buys_computer = yes”)

Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts

Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

    Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

Outline

- Classification and Prediction
- Decision Tree
- Naïve Bayes Classifier
- Support Vector Machines (SVM)
- K-nearest Neighbors
- Accuracy and Error Measures
- Feature Selection Methods
- Ensemble Methods
- Applications
- Summary

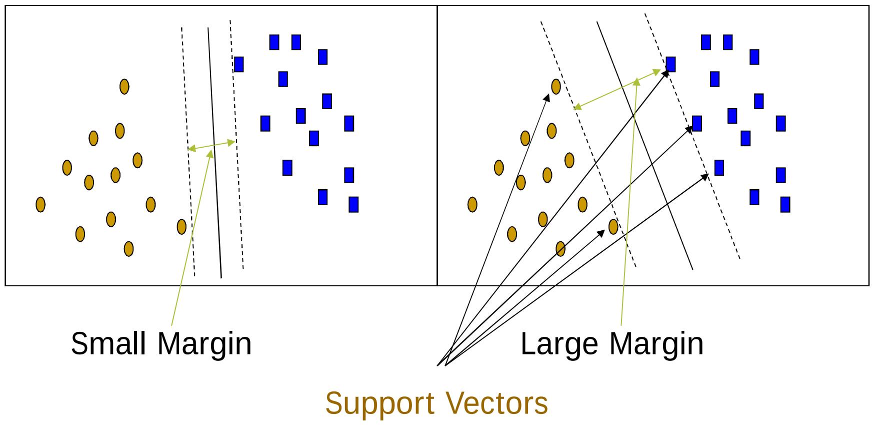
SVM—Support Vector Machines

- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)

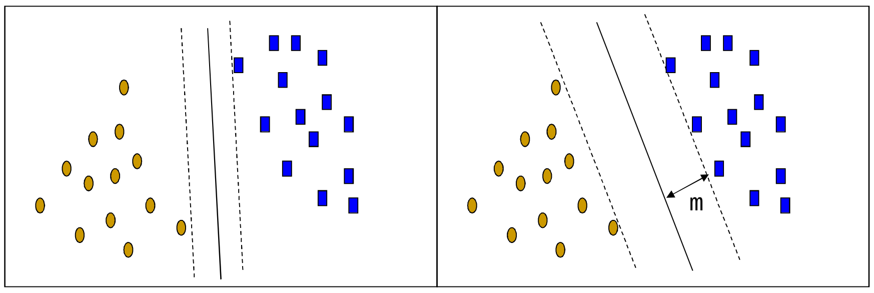
SVM—History and Applications

- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used both for classification and prediction
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

SVM—General Philosophy



SVM—When Data Is Linearly Separable



Let data D be (X_1, y_1), …, ($X_{|D|}$, $y_{|D|}$), where X_i is the set of training tuples associated with the class labels y_i

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., maximum marginal hyperplane (MMH)

SVM—Linearly Separable

- A separating hyperplane can be written as

  $W \bullet X + b = 0$

  where $W=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as

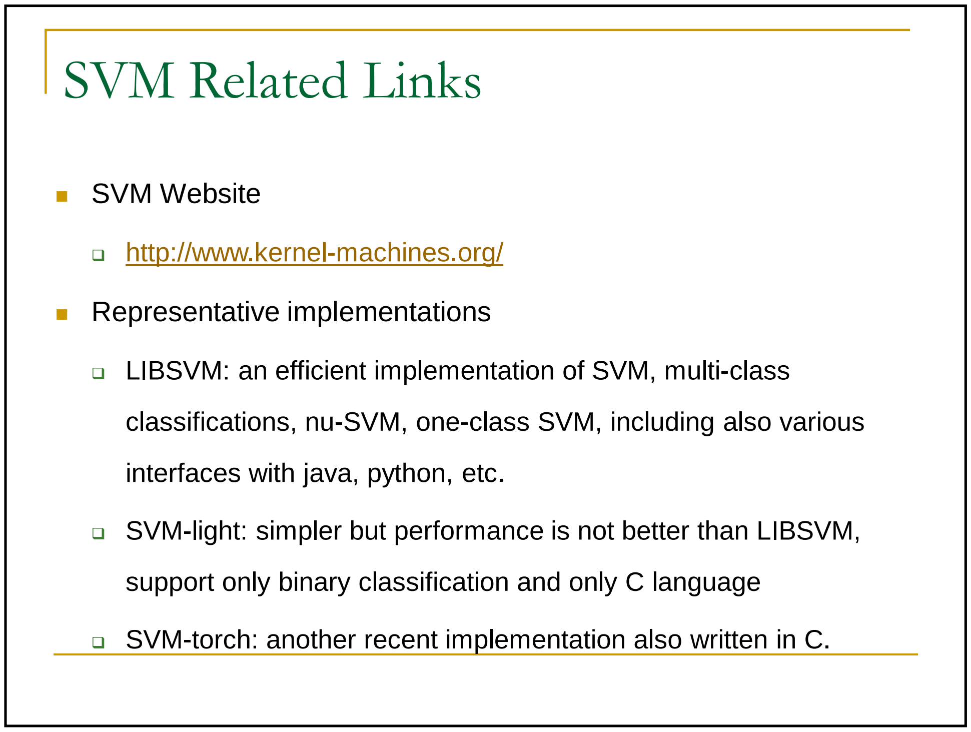
  $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:

  $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  $H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are support vectors
- This becomes a constrained (convex) quadratic optimization problem: Quadratic objective function and linear constraints $\rightarrow$ Quadratic Programming (QP) $\rightarrow$ Lagrangian multipliers

Why Is SVM Effective on High Dimensional Data?

- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples —they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high



SVM Related Links

- SVM Website
  - http://www.kernel-machines.org/
- Representative implementations
  - LIBSVM: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - SVM-light: simpler but performance is not better than LIBSVM, support only binary classification and only C language
  - SVM-torch: another recent implementation also written in C.