Università degli Studi di Milano Master Degree in Computer Science

Information Management course

Teacher: Alberto Ceselli

Lecture 20: 10/12/2015

Data Mining: Concepts and Techniques (3rd ed.)

- Chapter 8, 9 -

Jiawei Han, Micheline Kamber, and Jian Pei University of Illinois at Urbana-Champaign & Simon Fraser University © 2011 Han, Kamber & Pei. All rights reserved.

Classification methods

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Support Vector Machines
- Model Evaluation and Selection
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods

Classification: A Mathematical Mapping

- Classification: predicts categorical class labels
 - E.g., Personal homepage classification
 - $x_i = (x_1, x_2, x_3, ...), y_i = +1 \text{ or } -1$
 - x₁: # of word "homepage"
 - x₂: # of word "welcome"
- Mathematically,
 - $x \in X = \Re^n, y \in Y = \{+1, -1\},\$
 - We want to derive a function f: $X \rightarrow Y$
- Linear Classification
 - Binary Classification problem
 - Data above the red line belongs to class 'x'
 - Data below red line belongs to class 'o'
 - Examples: SVM, Perceptron, Probabilistic Classifiers



Perceptron: finding a separating hyperplane

Hyperplane: wx = b

- Mathematical model: find w
 - s.t. $wx_k b \ge 0$ (forall k: $y_k = 1$) $wx_k - b < 0$ (forall k: $y_k = -1$) ||w|| = 1
- Mathematical model: minimize $\sum_{i=1}^{m} d_k$

s.t.
$$wx_k - b + d_k \ge 0_k$$
 (forall k: $y_k = 1$)
 $wx_k - b - d_k < 0$ (forall k: $y_k = -1$)
 $|| w || = 1$



SVM—Support Vector Machines

- A classification method for both <u>linear and</u> <u>nonlinear</u> data
- Use a <u>nonlinear mapping</u> to map the original training data into a higher dimensional space
- In the new space, search for the linear optimal separating hyperplane (i.e. a "decision boundary")
- Speedup by using support vectors ("essential" training tuples) and margins (defined by the support vectors)
- Theoretically, with an appropriate mapping to a sufficiently high dimensional space, data from two classes can always be separated by a hyperplane

SVM—History and Applications

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

SVM—General Philosophy



Support Vectors

SVM—Margins and Support Vectors



SVM—When Data Is Linearly Separable



Let data D be $(\mathbf{X}_1, \mathbf{y}_1)$, ..., $(\mathbf{X}_{|D|}, \mathbf{y}_{|D|})$, where \mathbf{X}_i is the set of training tuples associated with the class labels \mathbf{y}_i

There are infinite lines (<u>hyperplanes</u>) separating the two classes but we want to <u>find the best one</u> (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 hyperplane: $\mathbf{wx} = b$ where $\mathbf{w} = \{w_1, w_2, ..., w_n\}$ 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:

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

 $H_2: w_0 + w_1 x_1 + w_2 x_2 \le -1$ for $y_i = -1$

- Any training tuples that fall on hyperplanes H₁ or H₂ (i.e., the sides defining the margin) are support vectors
- This becomes a constrained (convex) quadratic optimization problem: Quadratic objective function and linear constraints → Quadratic Programming (QP) → Lagrangian multipliers

SVM - A QP model

A hyperplane: wx = bwhere $w = \{w_1, w_2, ..., w_n\}$ is a weight vector and b a scalar (bias)

- Separating margin: $D = \frac{2}{\|w\|} \qquad \|w\| = \sqrt{\sum_{i=1}^{n} (w_i)^2}$
- Find an optimal hyperplane (linearly separable):

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

s.t. $y_{k}(wx_{k}-b) \ge 1 \forall k=1...m$

Find an optimal hyperplane (general):

$$\min \frac{1}{2} ||w||^2 + C \sum_{k=1}^m d_k$$

s.t. $y_k (w x_k - b) + d_k \ge 1 \forall k = 1 \dots m$
 $d_k \ge 0 \forall k = 1 \dots m$

12

SVM - A QP model

• Find an optimal hyperplane (general):

$$\min \frac{1}{2} \|w\|^2 + C \sum_{k=1}^m d_k$$

s.t. $y_k (w x_k - b) + d_k \ge 1 \forall k = 1 \dots m$
 $d_k \ge 0 \forall k = 1 \dots m$

Langrangean (dual) function:

$$L = \min \frac{1}{2} \|w\|^2 + C \sum_{k=1}^m d_k - \sum_{k=1}^m \alpha_k (y_k (w x_k - b) + d_k - 1) - \sum_{k=1}^m \mu_k d_k$$

Derivatives:

$$\frac{\partial L}{\partial w} = w - \sum_{k=1}^{m} \alpha_k y_k x_k$$
$$\frac{\partial L}{\partial b} = \sum_{k=1}^{m} \alpha_k y_k$$
$$\frac{\partial L}{\partial d_k} = C - \alpha_k - \mu_k$$

13

SVM - A QP model

Langrangean (dual) function:

$$L = \min \frac{1}{2} \|w\|^2 + C \sum_{k=1}^m d_k - \sum_{k=1}^m \alpha_k (y_k (w x_k - b) + d_k - 1) - \sum_{k=1}^m \mu_k d_k$$

Optimality conditions:

$$\frac{\partial L}{\partial w} = w - \sum_{k=1}^{m} \alpha_k y_k x_k = 0$$
$$\frac{\partial L}{\partial b} = \sum_{k=1}^{m} \alpha_k y_k = 0$$
$$\frac{\partial L}{\partial d_k} = C - \alpha_k - \mu_k = 0$$

- Dual problem: ... (blackboard discussion)
- Interpretation of KKT conditions: ... (blackboard discussion)

Why Is SVM Effective on High Dimensional Data?

- The complexity of trained classifier is characterized by the <u># of</u> <u>support vectors</u> rather than the dimensionality of the data
- The support vectors are the <u>essential or critical training examples</u> —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 <u>(upper) bound on the expected error rate</u> 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—Linearly Inseparable

Transform the original input data into a higher dimensional space

Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space Z using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3, \phi_4(X) = (x_1)^2, \phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{WZ} + b$, where W and Z are vectors. This is linear. We solve for W and b and then substitute back so that we see that the linear decision hyperplane in the new (Z) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$d(Z) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 (x_1)^2 + w_5 x_1 x_2 + w_6 x_1 x_3 + b$$

= $w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5 + w_6 z_6 + b$

 Search for a linear separating hyperplane in the new space

A,

SVM: Different Kernel functions

- Instead of computing the dot product on the transformed data, it is math. equivalent to applying a kernel function K(X_i, X_j) to the original data, i.e., K(X_i, X_j) = Φ(X_i) Φ(X_j)
- Typical Kernel Functions

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

Gaussian radial basis function kernel : $K(X_i, X_j) = e^{-||X_i - X_j||^2/2\sigma^2}$

Sigmoid kernel : $K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$

 SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters)

"geometric" Classifiers

- Advantages
 - Prediction accuracy is generally high
 - As compared to Bayesian methods in general
 - Robust, works when training examples contain errors
 - Fast evaluation of the learned target function
 - Bayesian networks are normally slow
- Weaknesses
 - Long training time
 - Difficult to understand the learned function (weights)
 - Bayesian networks can be used easily for pattern discovery
 - Not easy to incorporate domain knowledge
 - Easy in the form of priors on the data or distributions

SVM vs. Neural Network

SVM

- Deterministic algorithm
- Nice generalization properties
- Hard to learn learned in batch mode using quadratic programming techniques
- Using kernels can learn very complex functions

Neural Network

- Nondeterministic algorithm
- Generalizes well but doesn't have strong mathematical foundation
- Can easily be learned in incremental fashion
- To learn complex functions —use multilayer perceptron (nontrivial)

SVM Related Links

- SVM Website: http://www.kernel-machines.org/
- Representative implementations
 - LIBSVM: an efficient implementation of SVM, multiclass 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 in C
 - SVM-torch: another recent implementation also written in C

Classification methods

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Support Vector Machines
- Model Evaluation and Selection
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy?
 Other metrics to consider?
- Use test set of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
 - Holdout method, random subsampling
 - Cross-validation
 - Bootstrap
- Comparing classifiers:
 - Confidence intervals
 - Cost-benefit analysis and ROC Curves

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C ₁	$\neg C_1$
C ₁	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	6954	46	7000
buy_computer = no	412	2588	3000
Total	7366	2634	10000

- Given *m* classes, an entry, *CM_{i,j}* in a confusion matrix indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

	Ρ	Ν	All
¬C	FN	TN	Ν′
С	ТР	FP	Ρ'
A\P	С	¬C	

 Classifier Accuracy, or recognition rate: percentage of test set tuples that are correctly classified

Accuracy = (TP + TN)/All

Error rate: 1 – accuracy, or
 Error rate = (FP + FN)/All

Class Imbalance Problem:

- One class may be rare, e.g. fraud, or HIV-positive
- Significant majority of the negative class and minority of the positive class
- Sensitivity: True Positive recognition rate
 - Sensitivity = TP/P
- Specificity: True Negative recognition rate
 - Specificity = TN/N

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: coherence what % of tuples that the classifier labeled as positive are actually positive TP precision $\overline{TP + FP}$
- Recall: completeness what % of positive tuples did the classifier label as positive? TPrecall
 - Perfect score is 1.0
- Inverse relationship between precision & recall
- **F measure (F**₁ or **F-score)**: harmonic mean of $2 \times precision \times recall$ precision and recall,

precision + recall

- **F**_B: weighted measure of precision and recall
 - assigns ß times as much weight to recall as to precision

$$F_{\beta} = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	90	210	300	30.00 (<i>sensitivity</i>
cancer = no	140	9560	9700	98.56 (<i>specificity)</i>
Total	230	9770	10000	96.40 (<i>accuracy</i>)

Precision = 90/230 = 39.13%
 Recall = 90/300 = 30.00%

 $precision = \frac{TP}{TP + FP}$

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

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

Holdout method

- Given data is randomly partitioned into two independent sets
 - Training set (e.g., 2/3) for model construction
 - Test set (e.g., 1/3) for accuracy estimation
- Random sampling: a variation of holdout
 - Repeat holdout k times
 - Accuracy = avg. of the accuracies obtained
- Cross-validation (k-fold, where k = 10 is most popular)
 - Randomly partition the data into k mutually exclusive subsets, each approximately equal size
 - At *i*-th iteration, use D_i as test set and others as training set
 - Leave-one-out: k -fold with k = # of tuples (small sized data)
 - <u>Stratified cross-validation</u>: folds are clustered so that class dist. in each class is approx. the same as that in the initial data

Evaluating Classifier Accuracy: Bootstrap

Bootstrap

- Works well with small data sets
- Samples the given training tuples uniformly with replacement
 - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set

Several bootstrap methods, and a common one is .632 boostrap

- A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set.
 About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since (1 1/d)^d ≈ e⁻¹ = 0.368)
- Repeat the sampling procedure k times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k} \sum_{i=1}^{k} (0.632 \times Acc(M_i)_{test_set} + 0.368 \times Acc(M_i)_{train_set})$$

Model Selection: ROC Curves

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test subsets in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

Issues Affecting Model Selection

Accuracy

- classifier accuracy: predicting class label
- Speed
 - time to construct the model (training time)
 - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
 - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Classification methods

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Support Vector Machines
- Model Evaluation and Selection
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
 - Use a combination of models to increase accuracy
 - Combine a series of k learned models, M₁, M₂, ..., M_k, with the aim of creating an improved model M*
- Popular ensemble methods
 - Bagging: averaging the prediction over a collection of classifiers
 - Boosting: weighted vote with a collection of classifiers
 - Ensemble: combining a set of heterogeneous classifiers

Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
 - Given a set D of d tuples, at each iteration i, a training set D_i of d tuples is sampled with replacement from D (i.e., bootstrap)
 - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample X
 - Each classifier M_i returns its class prediction
 - The bagged classifier M* counts the votes and assigns the class with the most votes to X
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
 - Often significantly better than a single classifier derived from D
 - For noise data: not considerably worse, more robust
 - Proved improved accuracy in prediction

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
 - Weights are assigned to each training tuple
 - A series of k classifiers is iteratively learned
 - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i
 - The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

- Given a set of d class-labeled tuples, (X₁, y₁), ..., (X_d, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
 - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
 - Each tuple's chance of being selected is based on its weight
 - A classification model M_i is derived from D_i
 - Its error rate is calculated using D_i as a test set
 - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: err(X_j) is the misclassification error of tuple X_j. Classifier
 M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{i}^{a} w_i \times err(X_i)$$

• The weight of classifier M_i 's vote is $\log \frac{1 - error(M_i)}{error(M_i)}$

Random Forest (Breiman 2001)

- Random Forest:
 - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
 - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
 - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
 - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting