CS6220: DATA MINING TECHNIQUES

Chapter 8&9: Classification: Part 4

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

March 18, 2013

Chapter 8&9. Classification: Part 4

- Frequent Pattern-based Classification 
- Ensemble Methods
- Other Topics
- Summary

Associative Classification

- Associative classification: Major steps
  - Mine data to find strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Association rules are generated in the form of

    $P_1 \wedge p_2 \ldots \wedge p_l \rightarrow$ "$A_{class} = C$" (conf, sup)
  - Organize the rules to form a rule-based classifier
- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time
  - Associative classification has been found to be often more accurate than some traditional classification methods, such as C4.5

General Framework for Associative Classification

- Step 1:
  - Mine frequent itemsets in the data, which are typically attribute-value pairs
    - E.g., age = youth
- Step 2:
  - Analyze the frequent itemsets to generate association rules per class
- Step 3:
  - Organize the rules to form a rule-based classifier

Typical Associative Classification Methods

- **CBA** (Classification Based on Associations: Liu, Hsu & Ma, KDD'98)
  - Mine possible association rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support
- **CMAR** (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Classification: Statistical analysis on multiple rules
- **CPAR** (Classification based on Predictive Association Rules: Yin & Han, SDM'03)
  - Generation of predictive rules (FOIL-like analysis) but allow covered rules to retain with reduced weight
  - Prediction using best k rules
  - High efficiency, accuracy similar to CMAR

Discriminative Frequent Pattern-Based Classification

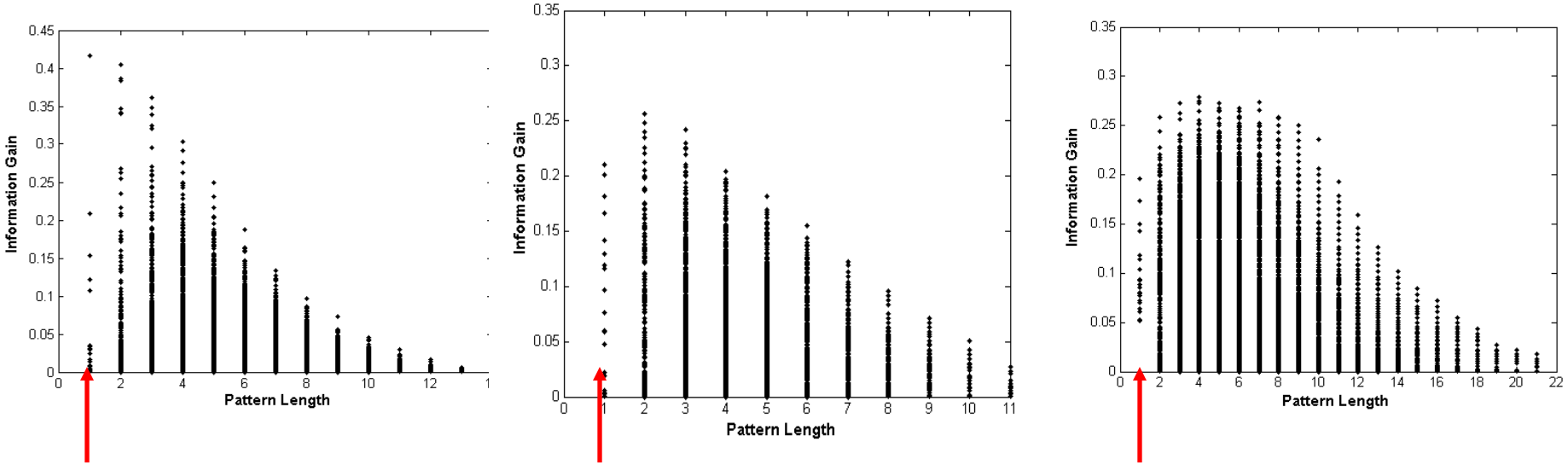
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, "Discriminative Frequent Pattern Analysis for Effective Classification", ICDE'07
  - Use combined features instead of single features
    - E.g., age = youth and credit = OK
- Accuracy issue
  - Increase the discriminative power
  - Increase the expressive power of the feature space
- Scalability issue
  - It is computationally infeasible to generate all feature combinations and filter them with an information gain threshold
  - Efficient method (DDPMine: FPtree pruning): H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct Discriminative Pattern Mining for Effective Classification", ICDE'08

Discriminative Frequent Pattern-Based Classification

- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, "Discriminative Frequent Pattern Analysis for Effective Classification", ICDE'07
- Accuracy issue
  - Increase the discriminative power
  - Increase the expressive power of the feature space
- Scalability issue
  - It is computationally infeasible to generate all feature combinations and filter them with an information gain threshold
  - Efficient method (DDPMine: FPtree pruning): H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct Discriminative Pattern Mining for Effective Classification", ICDE'08

Frequent Pattern vs. Single Feature

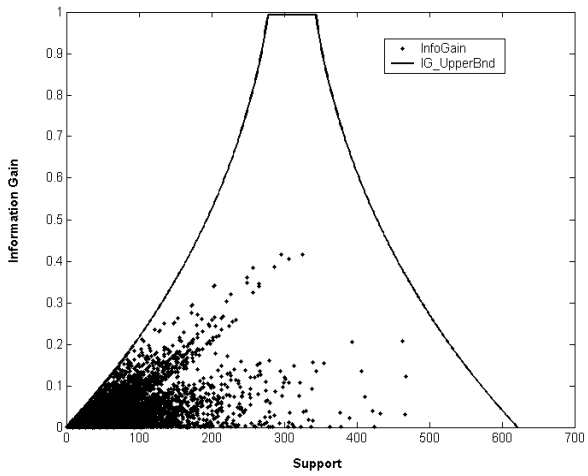
The discriminative power of some frequent patterns is higher than that of single features.



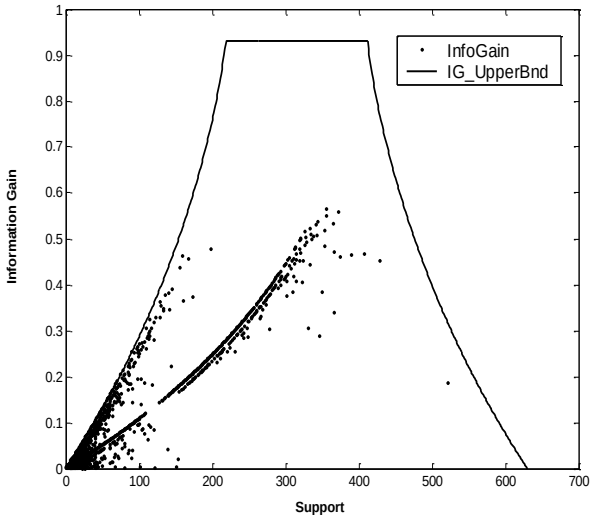
(a) Austral (b) Cleve (c) Sonar

Fig. 1. Information Gain vs. Pattern Length

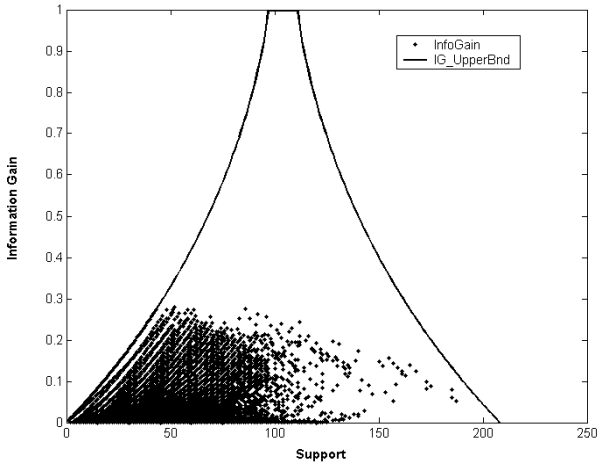
Empirical Results



(a) Austral



(b) Breast



(c) Sonar

Fig. 2. Information Gain vs. Pattern Frequency

Feature Selection

- Given a set of frequent patterns, both non-discriminative and redundant patterns exist, which can cause overfitting
- We want to single out the discriminative patterns and remove redundant ones
- The notion of Maximal Marginal Relevance (MMR) is borrowed
  - A document has high marginal relevance if it is both relevant to the query and contains minimal marginal similarity to previously selected documents

General Framework for Discriminative Frequent Pattern-based Classification

- Step 1:
  - Find the frequent patterns for the data set D, which are considered as feature candidates
- Step 2:
  - Select the best set of features by feature selection, and prepare the transformed data set D' with new features
- Step 3:
  - Build classification models based on the transformed data set

Experimental Results

Table 1. Accuracy by SVM on Frequent Combined Features vs. Single Features

Data	Single Feature			Freq. Pattern	
	Item_All	*Item_FS*	*Item_RBF*	*Pat_All*	*Pat_FS*
anneal	**99.78**	**99.78**	99.11	99.33	99.67
austral	85.01	85.50	85.01	81.79	**91.14**
auto	83.25	84.21	78.80	74.97	**90.79**
breast	97.46	97.46	96.98	96.83	**97.78**
cleve	84.81	84.81	85.80	78.55	**95.04**
diabetes	74.41	74.41	74.55	77.73	**78.31**
glass	75.19	75.19	74.78	79.91	**81.32**
heart	84.81	84.81	84.07	82.22	**88.15**
hepatic	84.50	89.04	85.83	81.29	**96.83**
horse	83.70	84.79	82.36	82.35	**92.39**
iono	93.15	94.30	92.61	89.17	**95.44**
iris	94.00	**96.00**	94.00	95.33	**96.00**
labor	89.99	91.67	91.67	94.99	**95.00**
lymph	81.00	81.62	84.29	83.67	**96.67**
pima	74.56	74.56	76.15	76.43	**77.16**
sonar	82.71	86.55	82.71	84.60	**90.86**
vehicle	70.43	72.93	72.14	73.33	**76.34**
wine	98.33	99.44	98.33	98.30	**100**
zoo	97.09	97.09	95.09	94.18	**99.00**

Table 2. Accuracy by C4.5 on Frequent Combined Features vs. Single Features

Dataset	Single Features		Frequent Patterns	
	Item_All	*Item_FS*	*Pat_All*	*Pat_FS*
anneal	98.33	98.33	97.22	**98.44**
austral	84.53	84.53	84.21	**88.24**
auto	71.70	77.63	71.14	**78.77**
breast	95.56	95.56	95.40	**96.35**
cleve	80.87	80.87	80.84	**91.42**
diabetes	**77.02**	**77.02**	76.00	76.58
glass	75.24	75.24	76.62	**79.89**
heart	81.85	81.85	80.00	**86.30**
hepatic	78.79	85.21	80.71	**93.04**
horse	83.71	83.71	84.50	**87.77**
iono	92.30	92.30	92.89	**94.87**
iris	**94.00**	**94.00**	93.33	93.33
labor	86.67	86.67	**95.00**	91.67
lymph	76.95	77.62	74.90	**83.67**
pima	75.86	75.86	76.28	**76.72**
sonar	80.83	81.19	**83.67**	**83.67**
vehicle	70.70	71.49	**74.24**	73.06
wine	95.52	93.82	96.63	**99.44**
zoo	91.18	91.18	95.09	**97.09**

Scalability Tests

Table 3. Accuracy & Time on Chess Data

min_sup	#Patterns	Time (s)	SVM (%)	C4.5 (%)
1	N/A	N/A	N/A	N/A
2000	68,967	44.703	92.52	97.59
2200	28,358	19.938	91.68	97.84
2500	6,837	2.906	91.68	97.62
2800	1,031	0.469	91.84	97.37
3000	136	0.063	91.90	97.06

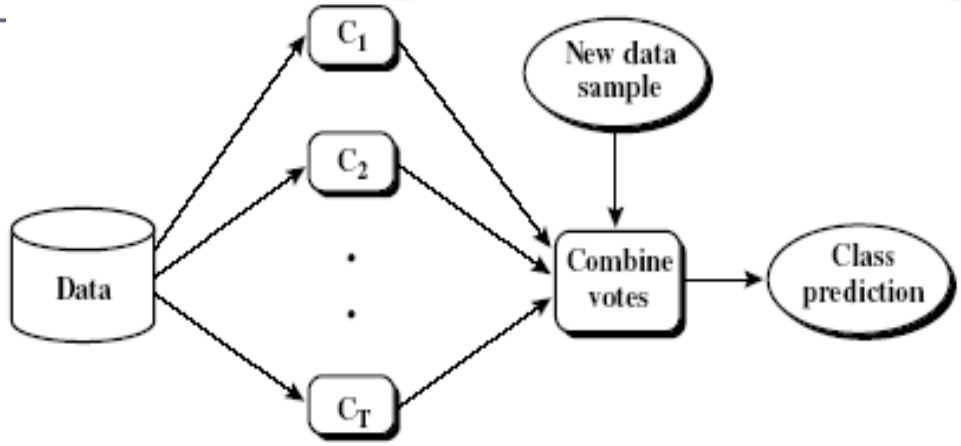
Table 4. Accuracy & Time on Waveform Data

min_sup	#Patterns	Time (s)	SVM (%)	C4.5 (%)
1	9,468,109	N/A	N/A	N/A
80	26,576	176.485	92.40	88.35
100	15,316	90.406	92.19	87.29
150	5,408	23.610	91.53	88.80
200	2,481	8.234	91.22	87.32

Chapter 8&9. Classification: Part 4

- Frequent Pattern-based classification
- Ensemble Methods 
- Other Topics
- Summary

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, ..., 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

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

Performance of Bagging

- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction
- Example
  - Suppose we have 5 completely independent classifiers…
  - If accuracy is 70% for each
    - The final prediction is correct, if at least 3 classifiers make the correct prediction
      - 3 are correct: $\binom{5}{3} \times (.7\text{^}3)(.3\text{^}2)$
      - 4 are correct: $\binom{5}{4} \times (.7\text{^}4)(.3\text{^}1)$
      - 5 are correct: $\binom{5}{5} \times (.7\text{^}5)(.3\text{^}0)$
    - In all, 10 (.7^3)(.3^2)+5(.7^4)(.3)+(.7^5)
    - **83.7% majority vote accuracy**
  - 101 Such classifiers
    - **99.9% majority vote accuracy**

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_t is learned, the weights are updated to allow the subsequent classifier, M_{t+1}, to **pay more attention to the training tuples that were misclassified** by M_t
  - 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, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round t,
  - Tuples from D are sampled (with replacement) to form a training set D_t of the same size based on its weight
  - A classification model M_t is derived from D_t
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
    - $w_{t+1,j} \propto w_{t,j} \times \exp(-\alpha_t)$ if j is correctly classified
    - $w_{t+1,j} \propto w_{t,j} \times \exp(\alpha_t)$ if j is incorrectly classified

$$\alpha_t : weight\ for classifier\ t\ , the\ higher\ the\ better$$

AdaBoost

- Error rate: err($\mathbf{X}_j$) is the misclassification error of tuple $\mathbf{X}_j$. Classifier M_t error rate ($\epsilon_t = \text{error}(M_t)$) is the sum of the weights of the misclassified tuples:

$$error(M_t) = \sum_{j}^{d} w_{tj} \times err(\mathbf{X}_{tj})$$

- The weight of classifier M_t's vote is

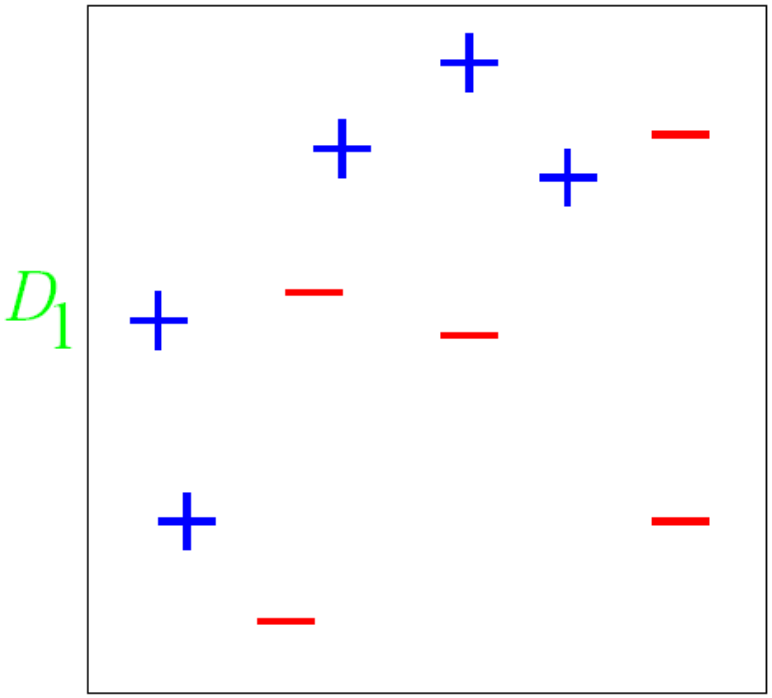
$$\alpha_t = \frac{1}{2} \log \frac{1 - error(M_t)}{error(M_t)}$$

- Final classifier M*

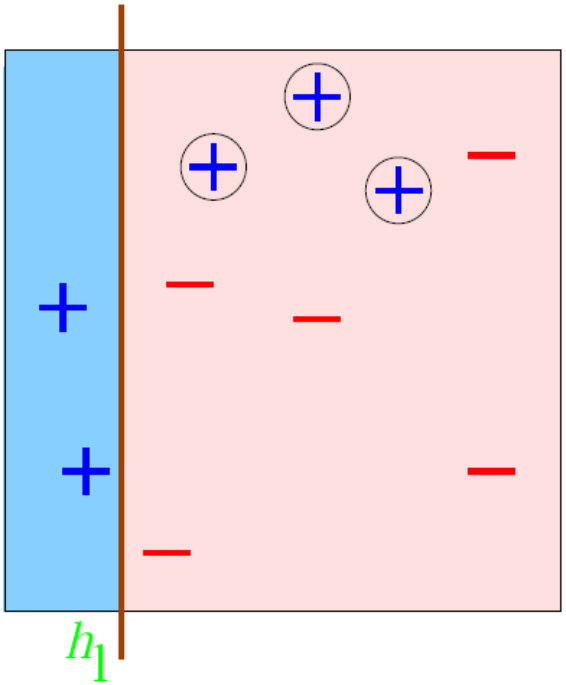
$$M^*(x) = sign(\sum_{t} \alpha_t M_t(x))$$

AdaBoost Example

- **From "A Tutorial on Boosting"**
  - By Yoav Freund and Rob Schapire
- Note they use h_t to represent classifier instead of M_t

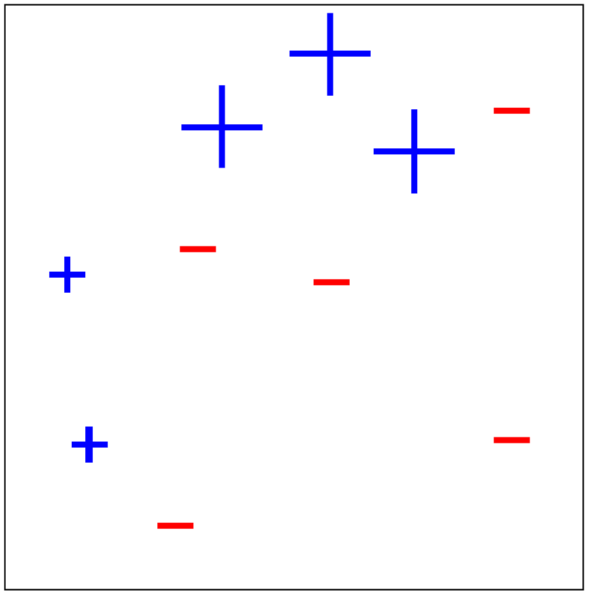


Round 1

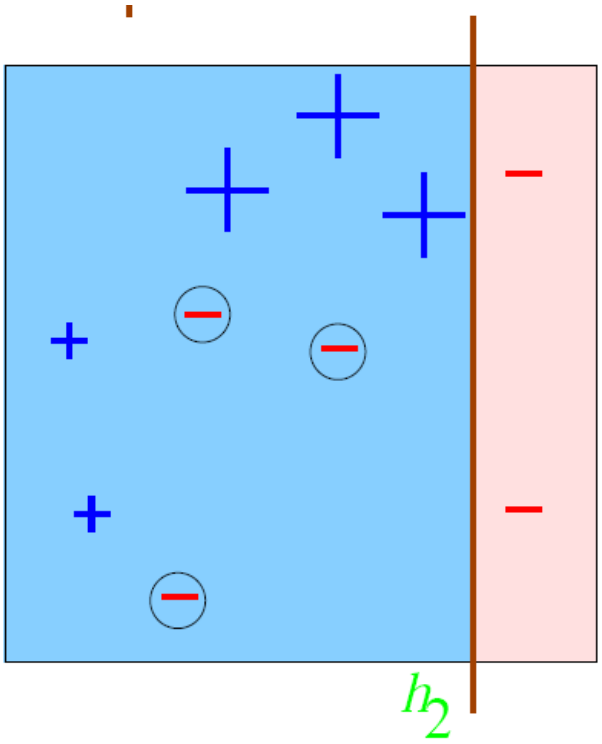


$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

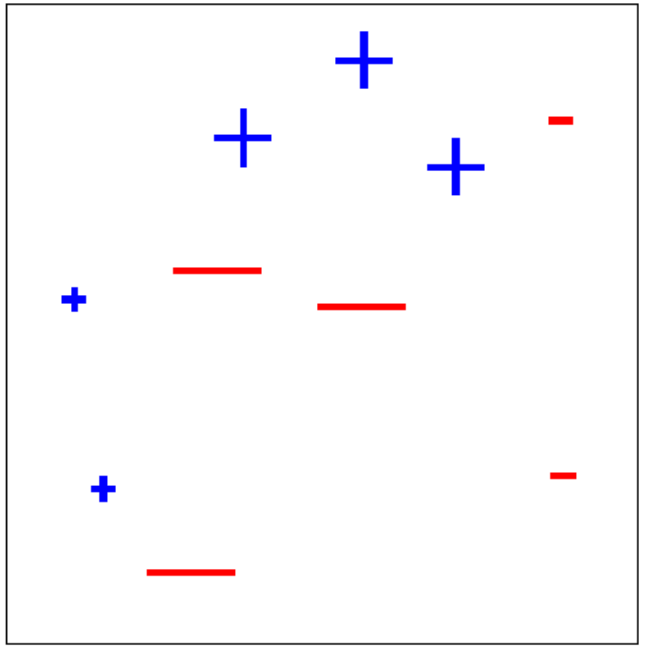


Round 2

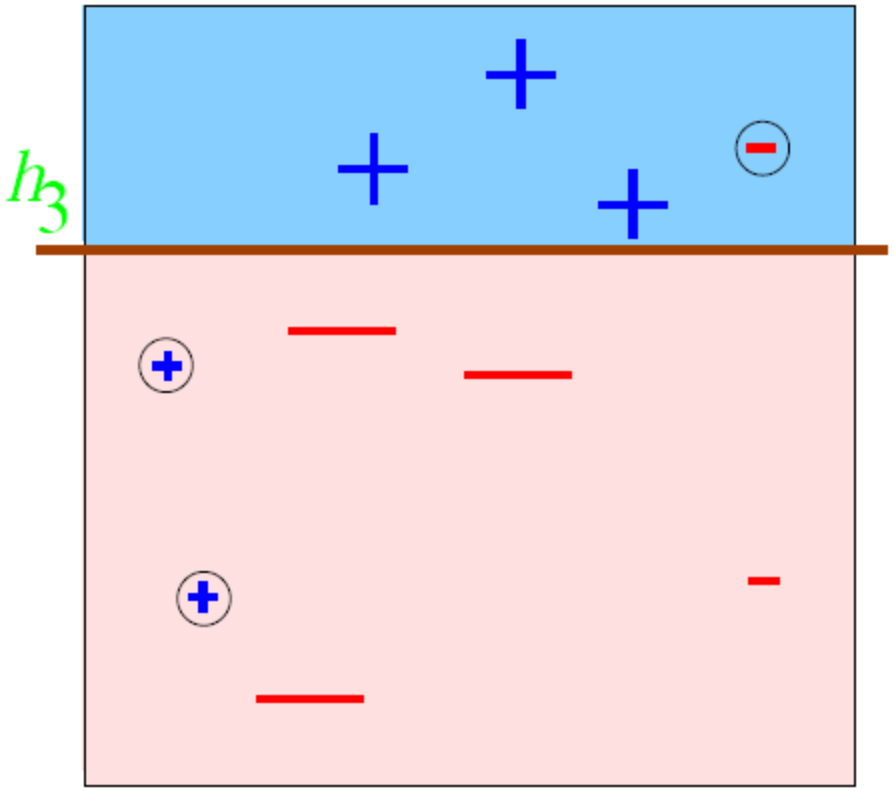


$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$



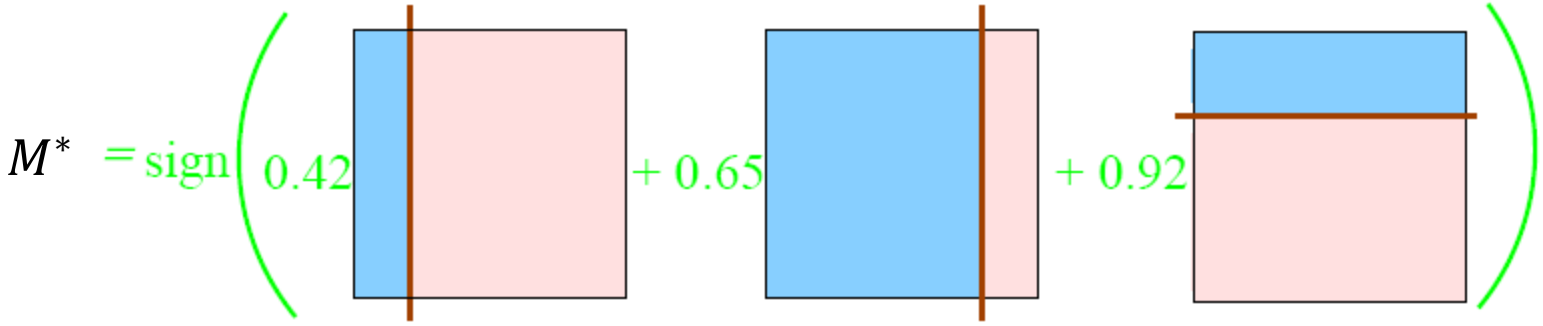
Round 3

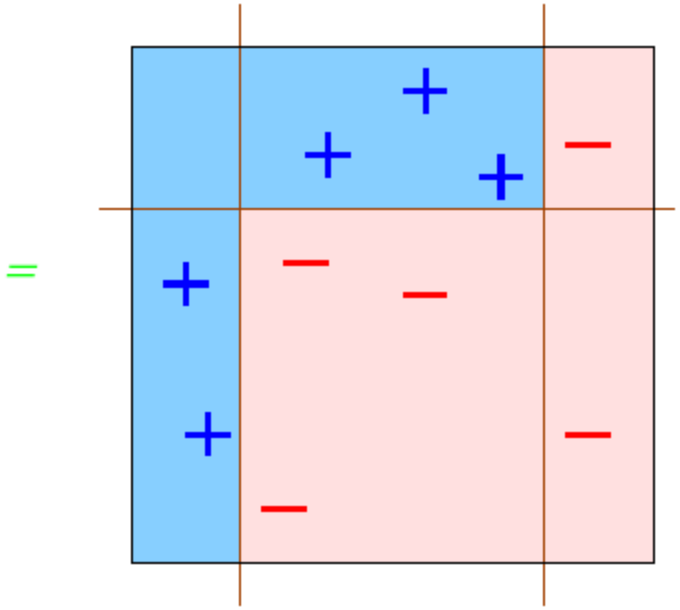


$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Final Model





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

Chapter 8&9. Classification: Part 4

- Frequent Pattern-based classification
- Ensemble Methods
- Other Topics 
- Summary

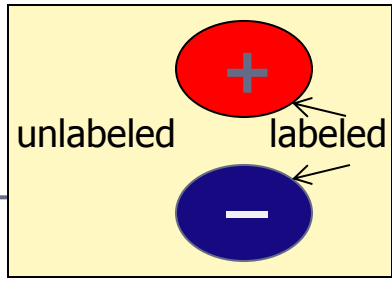
Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods for imbalance data in 2-class classification:
  - **Oversampling:** re-sampling of data from positive class
  - **Under-sampling:** randomly eliminate tuples from negative class
  - **Threshold-moving:** moves the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - Ensemble techniques: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

Multiclass Classification

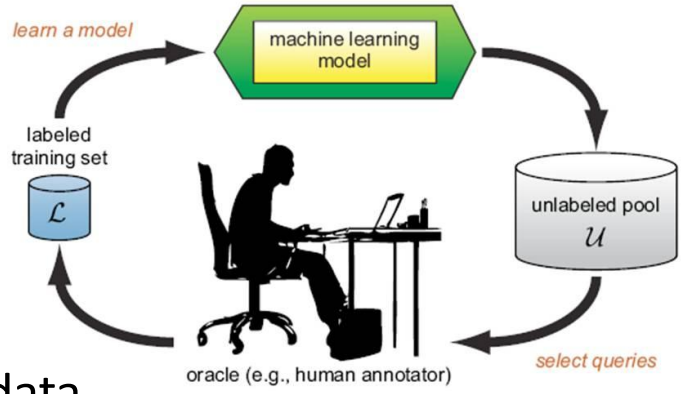
- Classification involving more than two classes (i.e., > 2 Classes)
- Method 1. **One-vs.-all** (OVA): Learn a classifier one at a time
  - Given m classes, train m classifiers: one for each class
  - Classifier j: treat tuples in class j as *positive* & all others as *negative*
  - To classify a tuple **X**, the set of classifiers vote as an ensemble
- Method 2. **All-vs.-all** (AVA): Learn a classifier for each pair of classes
  - Given m classes, construct m(m-1)/2 binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple **X**, each classifier votes. X is assigned to the class with maximal vote
- Comparison
  - All-vs.-all tends to be superior to one-vs.-all
  - Problem: Binary classifier is sensitive to errors, and errors affect vote count

Semi-Supervised Classification



- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Self-training:
  - Build a classifier using the labeled data
  - Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - Repeat the above process
  - Adv: easy to understand; disadv: may reinforce errors
- Co-training: Use two or more classifiers to teach each other
  - Each learner uses a mutually independent set of features of each tuple to train a good classifier, say f_1
  - Then f_1 and f_2 are used to predict the class label for unlabeled data X
  - Teach each other: The tuple having the most confident prediction from f_1 is added to the set of labeled data for f_2, & vice versa
- Other methods, e.g., joint probability distribution of features and labels

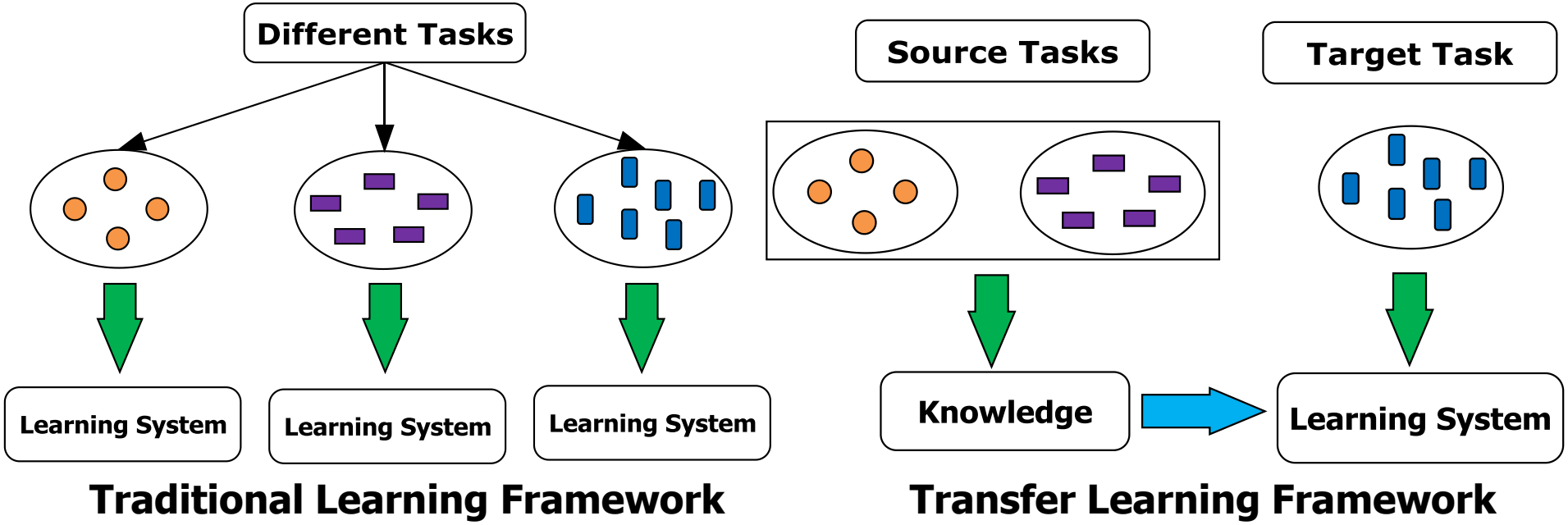
Active Learning



- Class labels are expensive to obtain
- Active learner: query human (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
  - L: a small subset of D is labeled, U: a pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to L, and learn a model
  - Goal: Achieve high accuracy using as few labeled data as possible
- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)
- Research issue: How to choose the data tuples to be queried?
  - Uncertainty sampling: choose the least certain ones
  - Reduce *version space*, the subset of hypotheses consistent w. the training data
  - Reduce expected entropy over U: Find the greatest reduction in the total number of incorrect predictions

Transfer Learning: Conceptual Framework

- Transfer learning: Extract knowledge from one or more source tasks and apply the knowledge to a target task
- Traditional learning: Build a new classifier for each new task
- Transfer learning: Build new classifier by applying existing knowledge learned from source tasks



Transfer Learning: Methods and Applications

- Applications: Especially useful when data is outdated or distribution changes, e.g., Web document classification, e-mail spam filtering
- *Instance-based transfer learning*: Reweight some of the data from source tasks and use it to learn the target task
- TrAdaBoost (Transfer AdaBoost)
  - Assume source and target data each described by the same set of attributes (features) & class labels, but rather diff. distributions
  - Require only labeling a small amount of target data
  - Use source data in training: When a source tuple is misclassified, reduce the weight of such tupels so that they will have less effect on the subsequent classifier
- Research issues
  - Negative transfer: When it performs worse than no transfer at all
  - Heterogeneous transfer learning: Transfer knowledge from different feature space or multiple source domains
  - Large-scale transfer learning

Chapter 8&9. Classification: Part 4

- Frequent Pattern-based classification
- Ensemble Methods
- Other Topics
- Summary 

Summary

- Frequent Pattern-based classification
  - Associative classification
  - Discriminative frequent pattern-based classification
- Ensemble Methods
  - Bagging; Boosting; AdaBoost
- Other Topics
  - Class imbalanced data; multi-class classification; semi-supervised learning; active learning; transfer learning