
Knowledge Discovery & Data Mining

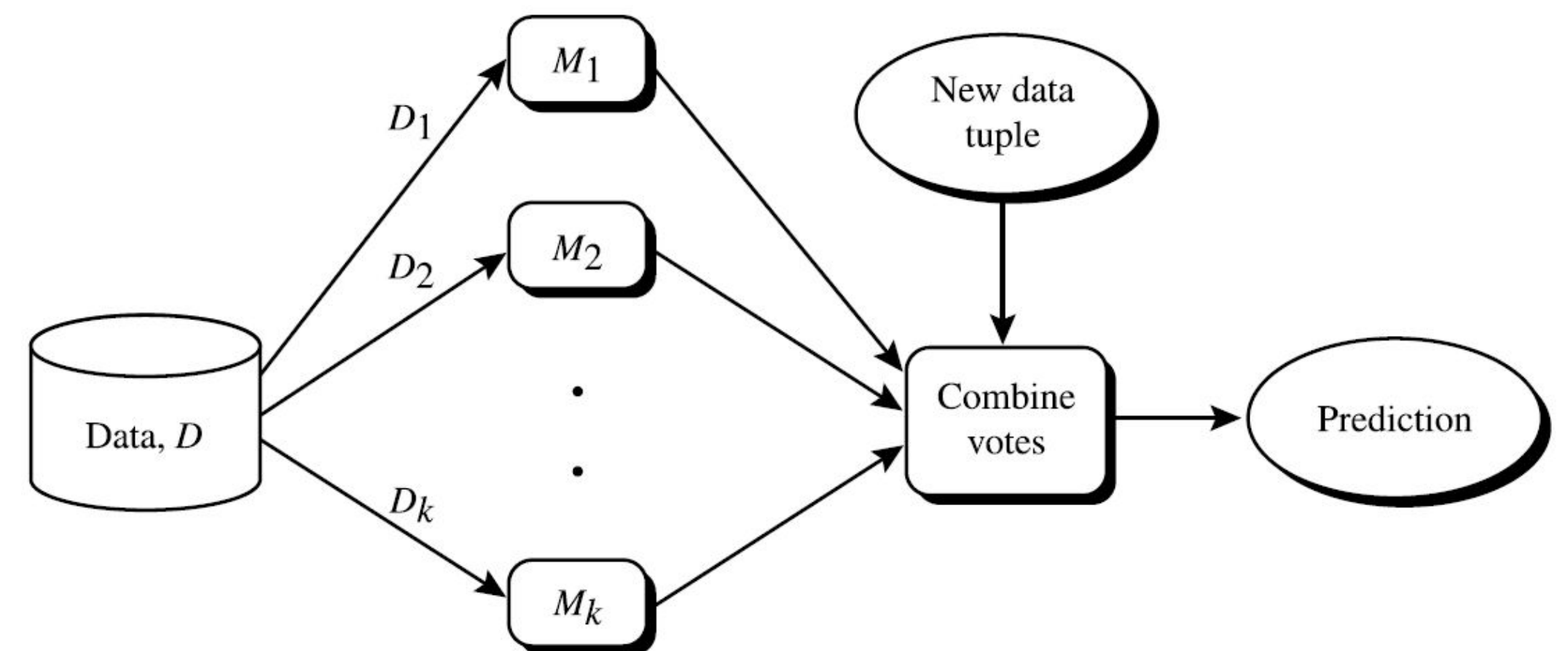
— Ensemble Methods —

Instructor: Yong Zhuang

yong.zhuang@gvsu.edu

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, \dots, 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: Bootstrap 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, $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_d, y_d)$
- Initially, all the weights of tuples are set the same ($1/d$)
- Generate k classifiers in k rounds. At round 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: $err(\mathbf{X}_j)$ is the misclassification error of tuple \mathbf{X}_j . Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j=1}^d w_j \times err(\mathbf{X}_j)$$

- The weight of classifier M_i 's vote is

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

Gradient boosting

- An iterative ensemble method that builds a series of weak learners (typically decision trees) in a stage-wise fashion.
- Take a regression task as example. Starts with a simple regression model $F(x)$ which outputs a constant
 - Suppose the loss function $L(y_i, F(x_i)) = \frac{1}{2}(y_i - \hat{y}_i)^2$, then compute the the negative gradient r_i .
$$r_i = y_i - \hat{y}_i$$
 - Fit a decision tree $M_t(x)$ for the training set $\{(x_1, r_1), \dots, (x_n, r_n)\}$
 - update the composite regression model. $F(x) \leftarrow F(x) + M_t(x)$
 - Repeat the process until a pre-defined number of trees are added or the residuals are small.
- Variant: XGBoost has been successfully used by data scientists in many data mining challenges, often leading to top competitive results.

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

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