CPE/EE 695: Applied Machine Learning

Lecture 5-2: Ensemble Learning

Dr. Shucheng Yu, Associate Professor
Department of Electrical and Computer Engineering
Stevens Institute of Technology

Classification formulation

- Given
  - an input space $\Re$
  - a set of classes $\omega = \{\omega_1, \omega_2, \dots \omega_n\}$
- the ***Classification Problem*** is
  - to define a mapping $f: \Re \rightarrow \omega$ where each x in $\Re$ is assigned to one class
- This mapping function is called a Decision Function

Single Classifier

- Most popular single classifiers:
  - Minimum Distance Classifier
  - Bayes Classifier
  - K-Nearest Neighbor
  - Decision Tree
  - Neural Network
  - Support Vector Machine

Drawback of Single Classifier

The “best” classifier not necessarily the ideal choice

- Problems:
  - Which one is the best?
    - Maybe more than one classifiers meet the criteria (e.g. same training accuracy), especially in the following situations:
      - » Without sufficient training data
      - » The learning algorithm leads to different local optima easily
  - Potentially valuable information may be lost by discarding the results of less-successful classifiers
    - E.g., the discarded classifiers may correctly classify some samples
  - The trained classifier may not be complex enough to handle the problem

Motivations for classifier ensemble

- Combining a number of trained classifiers lead to a better performance than any single one
  - Errors can be complemented by other correct classifications
  - Different classifiers have different knowledge regarding the problem
- To decompose a complex problem into sub-problems for which the solutions obtained are simpler to understand, implement, manage and update

Classifier ensemble

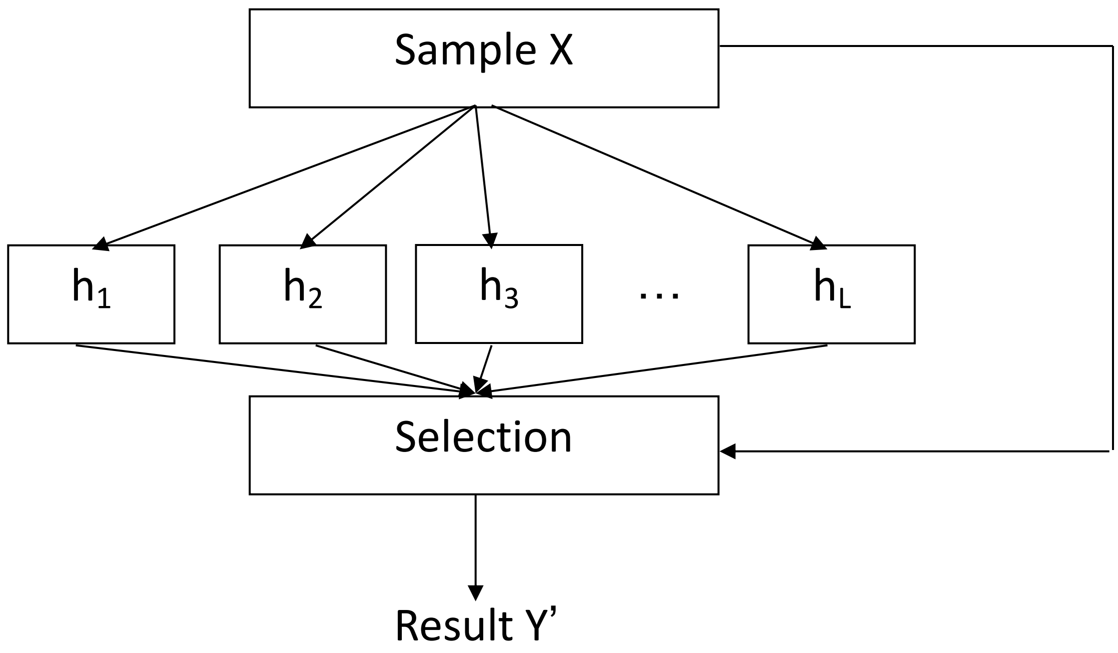
- Classifier ensemble consists of
  - a set of individual classifiers
  - a fusion/selection method:

    to combine/select individual classifier outputs to give a final decision
- Types of ensemble:
  - Classifier Selection
  - Classifier Fusion

Classifier Selection

- The input sample space is partitioned into smaller areas and each classifier learns the sample in each area
- For each sample, identify a single classifier which is most likely to produce the correct classification label
- Only the output of the selected classifier is taken as a final decision
- It is similar to the “Divide and Conquer” approach

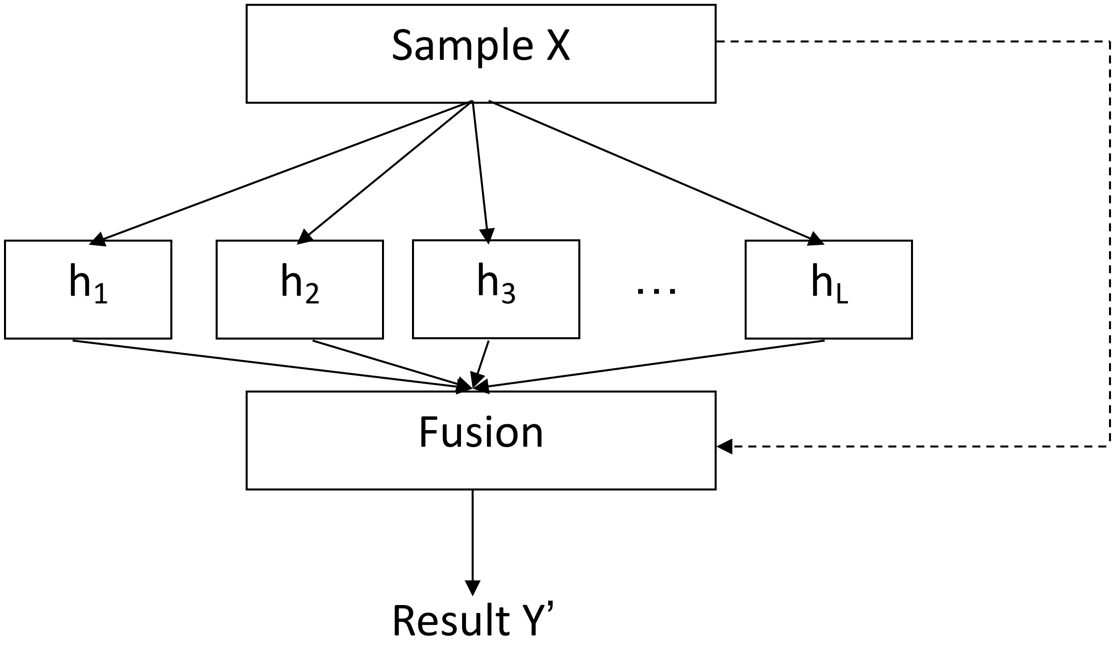
Classifier Selection



Classifier Fusion

- Mixing many different classifiers instead of extracting a single best classifier
- Each individual classifier tries to solve the same classification problem using different methods
  - E.g. Different training sets, different classifiers or different parameters
- The final output of a classifier ensemble system is determined by fusing the outputs of the individual classifiers

Classifier Fusion



Classifier ensemble MUST be better than Single?

- In all cases? NO!
- But in many cases, a ensemble learning can have a better performance than a single classifier

Affecting Factors

Three factors affecting the accuracy:

- Accuracy of individual classifiers

  How good are the individual classifiers?

- Fusion Methods

  How to combine classifiers?

- Diversity among classifiers

  What are the differences among different classifiers?

Accuracy of individual classifier

- The performance of an individual classifier is affected by
  - Training Dataset (sample and feature)
  - Learning Model (types of classifier)
  - Model's Parameters (e.g. the number of neurons in NN)

Fusion method

- A method to combine individual classifier outputs to reach the final decision for the classifier ensemble learning
- Since different fusion methods may have different final outputs for the same individual classifier outputs, MCS error is affected by its fusion method
- For Example

Sample **x**	Class1	Class2
Classifier 1	0.2	0.8
Classifier 2	0.7	0.3
Classifier 3	0.7	0.3

average Class 1: 0.53
Class 2: 0.47

max Class 1: 0.7
Class 2: 0.8

Fusion method

- Two popular fusion methods based on individual classifier outputs types:
  - Soft Output: soft type rules
  - Class Label: hard type rules

Soft Output

- For each sample, the classifier outputs a value representing the **confidence** of this sample belonging to each class
- The output of a classifier ***i*** is a c-dimensional vector $[d_{i,1},d_{i,2},...,d_{i,c}]^T$, where c is number of classes
- It is called decision profile

	ω_1	ω_2
Classifier 1	0.5	0.5
Classifier 2	0.2	0.8
Classifier 3	0.6	0.4

Class Label

- A classifier outputs only one class label

Classifier	output
Classifier 1	ω_1
Classifier 2	ω_2
Classifier 3	ω_1

Fusion method for Soft Output

- Popular methods:
  - Simple Summary Function (SSF)
  - Weighted Average (WA)

Simple Summary Function (SSF)

- Average (AVR), Maximum (MAX), Minimum (MIN), Product (PRO)
  - These fusion methods calculate the support ($\mu_j(x)$) for class j

AVR: $$\mu_j(x) = \frac{1}{L}\sum_{i=1}^{L} d_{i,j}(x)$$

MIN: $$\mu_j(x) = \min_{i} \{d_{i,j}(x)\}$$

MAX: $$\mu_j(x) = \max_{i=1,..,L} \{d_{i,j}(x)\}$$

PRO: $$\mu_j(x) = \prod_{i=1}^{L} d_{i,j}(x)$$

Simple Summary Function (SSF)

- An example:
  - Average
    - **<u>Class 1: 0.5</u>**
    - Class 2: 0.5
  - Minimum
    - Class 1: 0.2
    - **<u>Class 2: 0.3</u>**
  - Maximum
    - Class 1: 0.7
    - **<u>Class 2: 0.8</u>**
  - Product
    - Class 1: 0.084
    - **<u>Class 2: 0.096</u>**

Sample x	Class1	Class2
Classifier 1	0.2	0.8
Classifier 2	0.6	0.4
Classifier 3	0.7	0.3

Weighted Average (WA)

- The Weighted Average (WAVR)

  The support for class j :

$$\mu_j(x) = \frac{1}{L}\sum_{i=1}^{L} w_i d_{i,j}(x)$$

where $\sum_{i=1}^{L} w_i = 1$

Usually, the weight is calculated using the accuracy of the individual classifier

Weighted Average (WA)

- An example:
  - Case 1:
    - Assume that the weight of
      - Classifier 1: 0.7
      - Classifier 2: 0.2
      - Classifier 3: 0.1
    - Class 1: 0.33
    - Class 2: 0.67
  - Case 2:
    - Assume that the weight of
      - Classifier 1: 0.2
      - Classifier 2: 0.3
      - Classifier 3: 0.5
    - Class 1: 0.57
    - Class 2: 0.43

x	Class1	Class2
Classifier 1	0.2	0.8
Classifier 2	0.6	0.4
Classifier 3	0.7	0.3

Fusion method for Class Label

- Popular methods :
  - Majority Vote (MV)
  - Weighted Majority Vote (WMV)
  - Naïve Bayes (NB)

Majority Vote (MV)

- This method may be the oldest and the most well known strategy for decision making
- Assume that the label outputs of the classifiers are given as c-dimensional binary vectors
  $[d_{i,1}, d_{i,2}, \ldots, d_{i,c}]^T$ in $\{0,1\}^c$
- $i = 1,\ldots,L$ where $d_{i,j} = 1$ if label x in class j, and 0 otherwise
- The majority vote results in an ensemble decision for class k if

$$\sum_{i=1}^{L} d_{i,k} = \max_{j=1}^{c} \sum_{i=1}^{L} d_{i,j}$$

Majority Vote (MV)

- An example:
  - Class 1: 2 votes
  - **Class 2: 3 votes**

Sample x	Result
Classifier 1	1
Classifier 2	2
Classifier 3	1
Classifier 4	2
Classifier 5	2

Weighted Majority Vote (WMV)

- Similar to majority vote method but the influence of each vote to the final decision is not the same
- The weighted majority vote results in an ensemble decision for class k if

$$\sum_{i=1}^{L} w_i d_{i,k} = \max_{j=1}^{c} \sum_{i=1}^{L} w_i d_{i,j}$$

Weighted Majority Vote (WMV)

- An example:
  - Case 1:
    - Assume that the weight of
      - Classifier 1: 0.1
      - Classifier 2: 0.2
      - Classifier 3: 0.2
      - Classifier 4: 0.3
      - Classifier 5: 0.2
    - Class 1: 0.3
    - <u>Class 2: 0.7</u>
  - Case 2:
    - Assume that the weight of
      - Classifier 1: 0.4
      - Classifier 2: 0.2
      - Classifier 3: 0.2
      - Classifier 4: 0.1
      - Classifier 5: 0.1
    - <u>Class 1: 0.6</u>
    - Class 2: 0.4

Sample x	Result
Classifier 1	1
Classifier 2	2
Classifier 3	1
Classifier 4	2
Classifier 5	2

Naïve Bayes (NB)

- The term "**naïve**" is used since this method relies on the assumption that the classifiers are mutually independent but this situation does not occur normally

$$\mu_j(x) \propto \prod_{i=1}^{L} \hat{P}\left(\omega_j \mid d_{i,j}(x) = 1\right)$$

- $\hat{P}\left(\omega_j \mid d_{i,j}(x) = 1\right)$ is learned from training samples

Naïve Bayes (NB)

- An example:
  - The outputs of individual classifiers are **1 2 1**.

$$\hat{P}(\omega_1 \mid d_{1,1}(x)=1)=\frac{40}{70} \qquad \hat{P}(\omega_2 \mid d_{1,1}(x)=1)=\frac{30}{70}$$

$$\hat{P}(\omega_1 \mid d_{2,2}(x)=1)=\frac{30}{60} \qquad \hat{P}(\omega_2 \mid d_{2,2}(x)=1)=\frac{30}{60}$$

$$\hat{P}(\omega_1 \mid d_{3,1}(x)=1)=\frac{50}{90} \qquad \hat{P}(\omega_2 \mid d_{3,1}(x)=1)=\frac{40}{90}$$

- **Class 1: 0.16**
- Class 2: 0.10

Classifier1 Decision

True	Class1	Class2
Class1	40	10
Class2	30	20

Classifier2 Decision

True	Class1	Class2
Class1	20	30
Class2	20	30

Classifier3 Decision

True	Class1	Class2
Class1	50	0
Class2	40	10

Major methods

- Bootstrap aggregating (Bagging)
- Boosting (AdaBoost)
- Random subspace method
- Stacked generalization
- Mixture of experts

Bootstrap aggregating - Bagging

- Several classifiers are trained independently by using different bootstrap sample sets from the original dataset
- Combine outputs by voting (e.g., majority vote)
- Key Insight:

  Decreases error by decreasing the variance in the results due to ***unstable learners***, algorithms (like decision trees and neural networks) whose output can change dramatically when the training data is slightly changed.

L. Breiman, "Bagging predictors," Machine Learning,.24, pp. 123-140, 1996

Bagging

Input: sequence of m examples $\langle(x_1, y_1), \ldots, (x_m, y_m)\rangle$ with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

- Do t=1, 2, …T
  - Obtain bootstrap sample S_t by randomly drawing *m* instances, with replacement, from the original training set;
  - Call the WeakLearn based on the S_t to build a hypothesis;
- Using majority voting to combine these T individual hypothesis

Adaptive boosting - AdaBoost

- Similar to bagging method, each individual classifier is also trained using a different training set
- But the training set is selected based on the error of the trained hypothesis: more weights given to the "difficult" examples!
- Finally, Weight Majority Voting (WMV) will be used as a fusion method

- Key Insights
  - Instead of sampling (as in bagging) re-weigh examples!
  - Examples are given weights. At each iteration, a new hypothesis is learned (weak learner) and the examples are reweighted to focus the system on examples that the most recently learned classifier got wrong.
  - Final classification based on weighted vote of weak classifiers

Yoav Freund and Robert E. Schapire, "Experiments with a new boosting algorithm," In the *ICML'96,* pages 148--156, 1996.

AdaBoost.M1

Algorithm AdaBoost.M1

Input: sequence of m examples $\langle(x_1, y_1), \ldots, (x_m, y_m)\rangle$ with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i.

Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \to Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$. If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
5. Update distribution D_t: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$.

AdaBoost.M2

Algorithm AdaBoost.M2

Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Let $B = \{(i, y) : i \in \{1, \ldots, m\}, y \neq y_i\}$
Initialize $D_1(i, y) = 1/|B|$ for $(i, y) \in B$.
Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, providing it with mislabel distribution D_t.
2. Get back a hypothesis $h_t : X \times Y \to [0, 1]$.
3. Calculate the pseudo-loss of h_t: $\epsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update D_t: $D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \cdot \beta_t^{(1/2)(1 + h_t(x_i, y_i) - h_t(x_i, y))}$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) h_t(x, y)$.

Random Subspace method

- To choose a group of features for each classifier
- Each individual classifier is then trained on the randomly selected group of features
- Their results are combined by a fusion rule.

- Sample: Random forests .

T. K. Ho, "The random subspace method for constructing decision forests", IEEE Trans on pattern analysis and machine intelligence, 1998.

What is a Random Forest

A random forest is a collection of CART-like trees following specific rules for

- Tree growing
- Tree combination
- Self-testing
- Post-processing

Tree growing/Split

- Binary partitioning
- Each tree is grown at least partially *at random*
  - growing each tree on a different random subsample of the training data
  - splitting at any node from the eligible random subset

Prediction Mechanism

- Grow many trees.
- Each tree casts a vote at its terminal nodes. For a binary target the vote will be YES or NO
- Count up the YES votes. The percent YES votes received is the predicted probability

Acknowledgement

Part of the slide materials were based on Dr. Rong Duan's Fall 2016 course CPE/EE 695A Applied Machine Learning at Stevens Institute of Technology.

STEVENS

INSTITUTE of TECHNOLOGY

THE INNOVATION UNIVERSITY®

1870

stevens.edu