Ensemble Learning

Hong Chang

Institute of Computing Technology,
Chinese Academy of Sciences

Machine Learning Methods (Fall 2012)

Outline I

Rationale

- There is no single learning algorithm that in any domain always induces the most accurate learner.
- Each learning algorithm dictates a certain model with a set of assumptions, leading to the corresponding model bias.
- If the assumptions do not hold for the data, the model bias leads to error.
- Ensemble learning:
  - We construct a group of base learners which, when combined, has higher accuracy than the individual learners.
  - The base learners are usually not chosen for their accuracy, but for their simplicity.
  - The base learners should be accurate on different instances, specializing in different subdomains of the problem, so that they can complement each other.

Differences between Base Learners

- Different learning algorithms: different algorithms make different assumptions about the data and lead to different classifiers.
- Different hyperparameters of the same algorithm: e.g., number of hidden units in a multilayer perceptron, K in K-nearest neighbor classifier, error threshold in a decision tree, initial state of an iterative procedure, etc.
- Different representations of the same input object or event: multiple sources of information are combined, e.g., both acoustic input and video sequence of lip movements for speech recognition.
- Different training sets: multiple base learners are trained either in parallel or serially using different training sets.
- Different subtasks: the main task is defined in terms of a number of subtasks solved by different base learners.

Combining Base Learners

- Multiexpert combination methods (parallel style):
  - The base learners work in parallel.
  - Given an instance, they all give their decisions which are then combined to give the final decision.
  - E.g., voting, mixture of experts, stacked generalization.
- Multistage combination methods (sequential style):
  - The base learners work serially.
  - The base learners are sorted in increasing complexity: a complex base learner is not used unless the preceding simpler base learners are not confident.
  - E.g., cascading.

Why Ensembles Superior to Singles

- The generalization ability of an ensemble is usually much stronger than that of a single learner.
- The reasons:
  - The training data might not provide sufficient information for choosing a single best learner.
  - The search processes of the learning algorithms might be imperfect.
  - The hypothesis space being searched might not contain the true target function, while ensembles can give some good approximation.

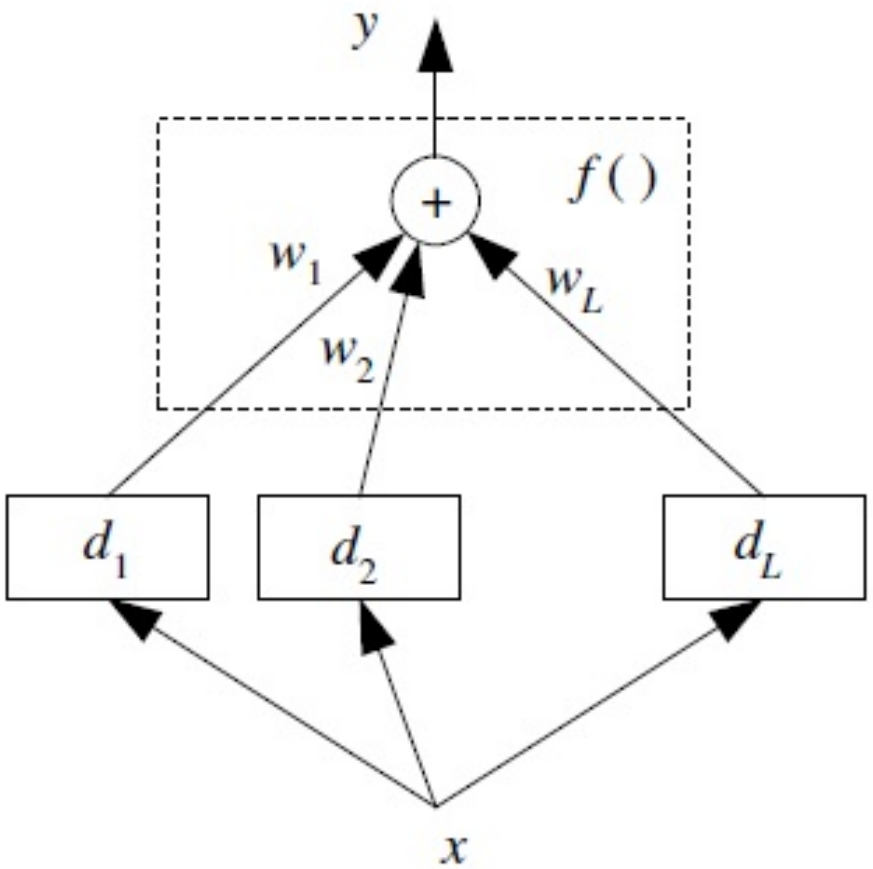
Model Selection vs. Model Averaging

- Model selection: works better if one model is significantly more accurate than other models
  - No ambiguity of which single model is better
- Equally weighted averaging: works better if all models have similar prediction accuracy, but are different
  - Some ambiguity of which single model is better
- Key to the success of model ensemble (averaging):
  - All models are reasonably accurate
  - Models are diverse (they have different predictions)

More Comments

- Empirical studies on popular ensemble methods: [BK99] [TW99] [OM99] ... ...
- Zhou et al. [ZWT02]: "Many could be better than all"
  - selective ensembles
- Ensemble methods are designed for classification, regression, clustering, and many kinds of machine learning tasks.
- Unsatisfactory points:
  - the comprehensibility of ensembles [ZJC03]
  - measures for diversity [KW03]
- A good reference book by Prof. Zhihua Zhou: Ensemble Methods: Foundations and Algorithms, Boca Raton, FL: Chapman & Hall/CRC, 2012.

Voting



Voting (2)

- Voting takes a convex combination of the base learners:

$$y = f(d_1, \ldots, d_L | \mathbf{\Phi}) = \sum_{j=1}^{L} w_j d_j$$

where w_j and d_j are the weight and prediction of learner j with

$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1,$$

$\mathbf{\Phi} = (w_1, \ldots, w_L)^T$ are the parameters and y is the final prediction.

Voting for Classification

- For class C_i:

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$

where d_{ji} is the vote of learner j for C_i and w_j is the weight of its vote.

- Simple voting (a.k.a. plurality voting, majority voting for 2 classes):

$$w_j = \frac{1}{L}$$

- Bayesian model selection:

$$P(C_i|x) = \sum_{\text{all models } M_j} P(C_i|x, M_j)P(M_j)$$

So the weights w_j can be seen as approximating the prior model probabilities $P(M_j)$.

Analysis

- Let there be L independent two-class classifiers, where $E[d_j]$ and $\text{var}(d_j)$ are the expected value and variance of d_j for classifier j.
- Expected value and variance of output:

$$E[y] = E[\sum_j \frac{d_j}{L}] = \frac{1}{L} L E[d_j] = E[d_j]$$

$$\text{var}(y) = \text{var}(\sum_j \frac{d_j}{L}) = \frac{1}{L^2}\text{var}(\sum_j d_j) = \frac{1}{L}\text{var}(d_j)$$

  As L increase, the expected value (and hence the bias) does not change but the variance (and hence the mean squared error) decreases, leading to an increase in accuracy.
- General case (non-independent classifiers):

$$\text{var}(y) = \frac{1}{L^2}\text{var}(\sum_j d_j) = \frac{1}{L^2}[\sum_j \text{var}(d_j) + 2\sum_j \sum_{i<j} \text{cov}(d_j, d_i)]$$

Stacking

- In typical stacking [Wol92] implementation:
  - A number of first-level individual learners are generated from the training data set by employing different learning algorithms.
  - The individual learners are then combined by a second-level learner which is called as meta-learner.
- It is closely related to information fusion methods.

Stacking Algorithm

- Input: Data set $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$
  First-level learning algorithms $\mathcal{L}_1, \ldots, \mathcal{L}_T$
  Second-level learning algorithm $\mathcal{L}$
- Process:
  For $t = 1, \ldots, T$
  $h_t = \mathcal{L}_t(\mathcal{D})$ %Train first-level individual learner h_t
  End
  $\mathcal{D}' = \emptyset$ %Generate a new data set
  For $i = 1, \ldots, N$
  For $t = 1, \ldots, T$
  $z_{it} = h_t(\mathbf{x}^{(i)})$
  End
  $\mathcal{D}' = \mathcal{D}' \cup \{((z_{i1}, \ldots, z_{iT}), y^{(i)})\}$
  End
  $h' = \mathcal{L}(\mathcal{D}')$ %Train the second-level learner h'
- Output: $H(\mathbf{x}) = h'(h_1(\mathbf{x}), \ldots, h_T(\mathbf{x}))$

Bagging

- Bagging [Bre96], a short form for bootstrap aggregating, is a voting method whereby the base learners are made different by training on slightly different training sets.
- Different training sets are generated by bootstrap, which draws N instances randomly from a training set $\mathcal{X}$ of size N with replacement.
- Bagging can be seen as a special case of model averaging which helps to reduce variance and hence improve accuracy.
- Unstable algorithms (e.g., decision trees and multilayer perceptrons) that cause large changes in the generated learner (i.e., high variance) with small changes in the training set can particularly benefit from bagging.

Bagging Algorithm

- **Input**: Data set $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$
  Base learning algorithm $\mathcal{L}$
  Number of learning rounds T
- **Process**:
  For $t = 1, \ldots, T$
  $\mathcal{D}_t = \mathit{Bootstrap}(\mathcal{D})$ %Generate a bootstrap sample from $\mathcal{D}$
  $h_t = \mathcal{L}(\mathcal{D}_t)$ %Train a base learner h_t from the bootstrap sample
  End
- **Output**: $H(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} 1(y = h_t(\mathbf{x}))$

Analysis

- The bootstrap samples usually overlap more than the cross validation samples and hence their estimates are more dependent.
- Probability that an instance is not chosen after N random draws:

$$(1 - \frac{1}{N})^N \approx e^{-1} = 0.368$$

  So each bootstrap sample contains only approximately 63.2% of the instances.
- Multiple bootstrap samples are used to maximize the chance that the system is trained on all the instances.
- Majority voting is usually used to predict the most-voted class.
- A variant of bagging, Random Forests [Bre01], is a powerful ensemble method.

Boosting

- In bagging, generating complementary base learners is left to chance and to the instability of the learning algorithm.
- In boosting [Sch90][FR97], complementary base learners are generated by training the next learner on the mistakes of the previous learners.
- Boosting combines weak learners (learners with accuracy just required to be better than random guessing, i.e., $> 1/K$ for K-class classification problems; weak but not too weak) to generate a strong learner.

AdaBoost

- AdaBoost [FR97] (a short form for adaptive boosting) is an iterative procedure that generates a sequence of base learners each focusing on the errors of previous ones.
- The original algorithm is AdaBoost.M1, but many variants of AdaBoost have also been proposed.
- AdaBoost modifies the probabilities of drawing instances for classifier training as a function of the error of the previous base learner.
- Initially all N instances have the same probability of being drawn.
- Moving from one iteration to the next iteration, the probability of a correctly classified instance is decreased and that of a misclassified instance is increased.
- The success of AdaBoost is due to its property of increasing the margin, making the aim of AdaBoost similar to that of SVM.

AdaBoost Algorithm

- Input: Data set $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$
  Base learning algorithm $\mathcal{L}$
  Number of learning rounds T
- Process:
  $W_1(i) = 1/N$ %Initial the weight distribution
  For $t = 1, \ldots, T$
  $h_t = \mathcal{L}(\mathcal{D}, W_t)$
  $\epsilon_t =$ the error of h_t
  $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$ %Determine the weight of h_t
  $W_{t+1}(i) = \frac{W_t(i) \exp(-\alpha_t y^{(i)} h_t(\mathbf{x}^{(i)}))}{Z_t}$ %Update the weight distribution,
  %where Z_t is a normalization factor
  End
- Output: $H(\mathbf{x}) = sign(f(\mathbf{x})) = sign \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

E. Bauer and R. Kohavi.

An empirical comparison of voting classification algorithms: Bagging, boosting, and variants.
Machine Learning, 36(1-2):105–139, 1999.

L. Breiman.

Bagging predictors.
Machine Learning, 24(2):123–140, 1996.

L. Breiman.

Random forests.
Machine Learning, 45(1):5–32, 2001.

Y. Freund and R.E.Schapire.

A decision-theoretic generalization of on-line learning and an application to boosting.
Journal of Computer and System Sciences, 55(1):119–139, 1997.

L.I. Kuncheva and C.J. Whitaker.

Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy.
Machine Learning, 51(2):181–207, 2003.

D. Opitz and R. Maclin.

Popular ensemble methods: An empirical study.
Journal of Artificial Intelligence Research, 11:169–198, 1999.

R.E. Schapire.

The strength of weak learnability.
Machine Learning, 5(2):197–227, 1990.

K.M. Ting and I.H. Witten.

Issues in stacked generalization.
Journal of Artificial Intelligence Research, 10:271–289, 1999.

D.H. Wolpert.

Stacked generalization.
Neural Networks, 5(2):241–260, 1992.

Z.H. Zhou, Y. Jiang, and S.F. Chen.

Extracting symbolic rules from trained neural network ensembles.
AI Communications, 16(1):3–15, 2003.

Z.H. Zhou, J. Wu, and W. Tang.

Ensembling neural networks: Many could be better than all.
Artificial Intelligence, 137(1-2):239–263, 2002.