Creating ensembles of classifiers via fuzzy clustering and deflection

Fuzzy Sets And Systems - Elsevier (JCR: 1.875)

Huaxiang Zhang, Jing Lu (Shandong, China)

Ion Marqués, Grupo de Inteligencia Computacional, UPV/EHU

27-01-2012

Introduction

- Two novel ensemble construction approaches:

1. The first method, **FuzzyBoost**, applies fuzzy clustering to the training data.
2. The second proposed approach, **Deflected FuzzyBoost Algorithm (DFA)**, builds an ensemble using deflection techniques with the FoozyBoost method .

Introduction

- Main idea:
  - A data point located far away from the class boundaries is different from that of a data point located near the class boundaries.
  - This characteristic represents the **fuzziness** of the training data and can be used when creating ensembles with diverse classifiers.

Fuzzy-clustering training data.

- Use **Fuzzy-clustering method (FCM)** to calculate fuzzy membership for each training instance. Fuzzily partitioning the training data is carried out through iteratively optimizing the objective function J_o:

$$J_o = \sum_{i=1}^{d} \sum_{j=1}^{c} \mu_{ij}^{m} \|\mathbf{x}_i - \mathbf{v}_j\|^2$$

Figure: m is any real number greater than one and represents the fuzziness degree, μ_{ij} is the degree of membership of the ith instance $\mathbf{x}_i$ with respect to class j, c is the number of classes of the training data, and d denotes the training data set size. $\mathbf{x}_i$ is the ith training instance, and $\mathbf{v}_i$ is the center of class j. $\|*\|$ denotes the Euclidean distance representing the similarity between an instance and the class center.

Fuzzy-clustering training data.

- During each iteration step, we update μ_{ij} membership and $\mathbf{v}_i$ class center.:

$$\mu_{ij} = \left[\sum_{k=1}^{c} \left(\frac{\|\mathbf{x}_i - \mathbf{v}_j\|^2}{\|\mathbf{x}_i - \mathbf{v}_k\|^2} \right)^{1/(m-1)} \right]^{-1}$$

$$\mathbf{v}_j = \frac{\sum_{i=1}^{d} \mu_{ij}^m \mathbf{x}_i}{\sum_{i=1}^{d} \mu_{ij}^m}$$

Figure: m is any real number greater than one and represents the fuzziness degree, μ_{ij} is the degree of membership of the ith instance $\mathbf{x}_i$ with respect to class j, c is the number of classes of the training data, $\mathbf{x}_i$ is the ith training instance, and $\mathbf{v}_i$ is the center of class j. $\|*\|$ denotes the Euclidean distance representing the similarity between an instance and the class center.

- The iteration stops when function J_o converges.

Generating training data set for component classifiers

- They employ the concept of entropy commonly used in information theory to characterize the fuzziness of each training instance, and calculate the **fuzzy information** it contains:

$$\text{Info}(\mathbf{x}_i) = -\sum_{j=1}^{c} \mu_{ij} \log_2 \mu_{ij}$$

- If one μ_{ij} equals 0 and the rest equal 1 $\rightarrow \text{Info}(\mathbf{x}_i) = 0$. This may imply that $\mathbf{x}_i$ can be labeled easily.
- If each μ_{ij} equals $1/c$ $\rightarrow \text{Info}(\mathbf{x}_i)$ is maximized. Labeling $\mathbf{x}_i$ is difficult.

Generating training data set for component classifiers

- The **reciprocal of the fuzzy information** is

$$\overline{\text{Info}}(\mathbf{x}_i) = \frac{1}{\text{Info}(\mathbf{x}_i)}$$

- Calculating the weight for each training instance according to $\text{Info}(\mathbf{x}_i)$ and $\overline{\text{Info}}(\mathbf{x}_i)$, we build build two individual classifiers h_1 and h_1^* on weighted bootstrap samples from the original data set.

Algorithm description

1. Use FCM to build training set D
2. For each training instance $\mathsf{x}_i \in D$, calculate $\mathrm{Info}(\mathsf{x}_i)$ and $\overline{\mathrm{Info}}(\mathsf{x}_i)$.
3. Calculate two weights for each training instance:

$$w(\mathbf{x}_i) = \frac{\mathrm{Info}(\mathbf{x}_i)}{\sum_{\mathbf{x}_i \in D} \mathrm{Info}(\mathbf{x_i})}$$

$$\overline{w}(\mathbf{x}_i) = \frac{\overline{\mathrm{Info}}(\mathbf{x}_i)}{\sum_{\mathbf{x}_i \in D} \overline{\mathrm{Info}}(\mathbf{x}_i)}$$

Algorithm description

We form two weights $W_1 = \{w(\mathbf{x}_i)|\forall \mathbf{x}_i \in D\}$ and $\overline{W}_1 = \{\overline{w}(\mathbf{x}_i)|\forall \mathbf{x}_i \in D\}$, which satisfy:

$$\sum_{i=1}^{d} w(\mathbf{x}_i) = 1, \quad \forall w(\mathbf{x}_i) \in [0, 1]$$

$$\sum_{i=1}^{d} \overline{w}(\mathbf{x}_i) = 1, \quad \forall \overline{w}(\mathbf{x}_i) \in [0, 1]$$

4 Feed the learning algorithms h_1 and h_1^* with a training data set that consists of d examples drawn from the original data set based on the weights W_1 and $\overline{W}_1$, respectively, using the weighted bootstrap approach.

Algorithm description

5 **Iterative process** (for a given base classifier number r), for $t = 1, \dots, r-1$:

- Update the fuzzy memberships for the data correctly classified by mvh_t (the majority vote result of classifiers h_1 ...h_t) and mvh_t^* by

$$\begin{cases} \mu_{ij}(t+1) = \mu_{ij}(t) + \alpha(1 - \mu_{ij}(t)) \\ \mu_{ik}(t+1) = \mu_{ik}(t) - \alpha\mu_{ik}(t), \quad k \neq j \end{cases}$$

- Update the miss-classified ones by:

$$\mu_{ik}(t+1) = \mu_{ik}(t) - \alpha\left(\mu_{ik}(t) - \frac{1}{c}\right) \quad (k = 1, 2, \dots, j, \dots, c)$$

The constant α is a reward parameter that lies in (0,1).

- With the new fuzzy memberships, recalculate $\text{Info}(\mathbf{x}_i)$ and $\overline{\text{Info}}(\mathbf{x}_i)$, obtain new weight sets W_{t+1} and $\overline{W}_{t+1}$ and training classifiers h_{t+1} and h^*_{t+1}.

Ojo: As we want to make the constructed classifiers gradually focus on those not easily classified, the weights of the correctly classified data should decrease. We adopt the following approach to update $\overline{\text{Info}}(\mathbf{x}_i)$ of the data correctly classified according to the majority vote result of the t classifiers h^*_1 ...h^*_t

$$\overline{\text{Info}}(\mathbf{x}_i) = \overline{\text{Info}}(\mathbf{x}_i)/(e^{I(y_i = mvh^*(\mathbf{x}_i))\cdot\overline{\text{Info}}(\mathbf{x}_i)})$$

where y_i is the real class label, $\overline{\text{Info}}(\mathbf{x}_i)$ of the right is $1/\text{Info}(\mathbf{x}_i)$ and I is 0 or 1.

Algorithm description

6. Calculate the error rate and parameter alpha for each base classifier c_i acording to $\varepsilon_i = (1/N) \sum_j I(c_i(x_j) \neq y_j)$ and $\alpha_i = \frac{1}{2} \ln (1 - \varepsilon_i)/\varepsilon_i$, and outputting the final decision of the ensemble according to $C^*(x) = \text{argmax}_{y \in C} \sum_{i=1}^{r} \alpha_i I(c_i(x) = y)$.

Here ends the explanation of FoozyBoost.

Deflected FuzzyBoost Algorithm (DFA) variation

- FCM converges to one minimum of the objective function J_o which may have multiple minimizers.
- They adopt a technique to get more minimizers of J_o: After a minimum of J_o has been obtained, they reformulate it with a transformation, such that the reformulated objective function will not obtain minima at previous minimal points but keeps all the other minima of the original objective function locally unchanged. This property is called the **deflection property.**

Deflected FuzzyBoost Algorithm (DFA) variation

- Given the kth reformulated function $J_{o,k}$ and the fuzzy memberships learned using the FCM, the function transformation is obtained

$$J_{o,k+1} = J_{o,k} \cdot \mathrm{Tanh}(\lambda_k \|\boldsymbol{\mu} - \boldsymbol{\mu}_k\|)$$

Figure: $\boldsymbol{\mu}$ is the fuzzy membership vector, $\boldsymbol{\mu}_k$ is the kth group of fuzzy memberships learned by applying FCM to $J_{o,k}$, and λ_k is a relaxing parameter.

Deflected FuzzyBoost Algorithm (DFA) variation

Algorithm (Given the deflection number s and the base classifier number r)**:**

1. Use FCM to get a group of fuzzy memberships of the objective function. *(as in basic FoozyBoost)*
2. Obtain $s + 1$ groups of fuzzy membership transforming the objective function with the deflection technique, applying FCM to the transformed objective function, and calculating a new group of fuzzy memberships.

Deflected FuzzyBoost Algorithm (DFA) variation

Algorithm (cont.):

3. Use FuzzyBoost approach to generate $\lfloor r/(s+1) \rfloor$ classifiers based on each group of fuzzy Memberships. If $(s+1) * \lfloor r/(s+1) \rfloor < r$, we train $r - s * \lfloor r/(s+1) \rfloor$ classifiers using the finally obtained group of memberships. r base classifiers can be constructed in the end.
3. Calculate the error rate and parameter alpha for each base classifier c_i acording to $\varepsilon_i = (1/N) \sum_j I(c_i(x_j) \neq y_j)$ and $\alpha_i = \frac{1}{2} \ln (1 - \varepsilon_i)/\varepsilon_i$, and outputting the final decision of the ensemble according to $C^*(x) = \operatorname{argmax}_{y \in C} \sum_{i=1}^{r} \alpha_i I(c_i(x) = y)$. *(as in basic FoozyBoost).*

Experimental results

- **Databases**: 20 from UCI repository. (In order to process data with non-numerical attributes, we define the nominal attribute difference between two instances to be 0 if the attribute values are the same or 1 if they are different. When calculating the class center, we process the data set and count the number for each value of the nominal attribute, and denote the numbers as nominal attribute value numbers.)
- **Evaluation approach**: Only one classifier tested: C4.5. Evaluation using 10-fold cross-validation 3 times on each database.
- **Parameters**: The same for all the data bases.

Experimental results

Besides the typical tables, they present the following results table:

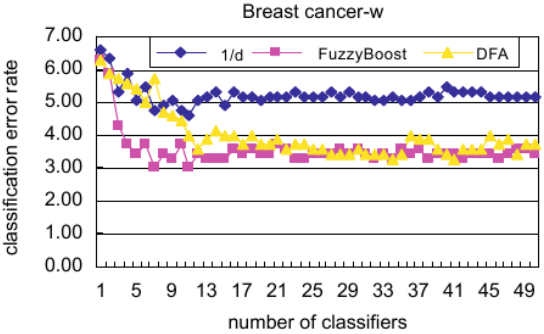
DFA/FuzzyBoost (Win/lose/draw)	FuzzyBoost/AdaBoost (Win/lose/draw)	FuzzyBoost/Bagging (Win/lose/draw)
8/6/6	14/5/1	15/4/1

DFA/AdaBoost (Win/lose/draw)	DFA/Bagging (Win/lose/draw)	FuzzyBoost/Alle (Win/lose/draw)	FuzzyBoost/Alld (Win/lose/draw)
13/7/0	17/3/0	19/0/1	18/0/2

Experimental results

Weight initialization evaluation

- Over 6 databases, test algorithms with different classifier number. The results are usually similar to this one:



Experimental results

The influence of the parameters

- They fix the deflection parameter and evaluate the influence of the reward parameter on the classification performance of the ensemble by implementing experiments on the 20 data sets and calculating the mean classification error rates.
- The results show that the classification performance of the ensemble can be improved through concurrently selecting an optimal reward parameter and a deflection number for a specific data set.

Main contributions:

1. Adopt FCM to cluster the training data in order to get the distribution attribute of the training data, and propose fuzzy entropy to evaluate the labeling difficulty of each training example.
2. FCM converges only to one local optima of the objective function, and each local optimal point corresponds to an optimal fuzzy labeling of the training data. They employ a deflection technique to learn different optimal points of the objective function, and get different optimal fuzziness of the training data.
3. Experimental results show that there is an optimal learning rate for each data set. The optimal learning rate is different from one data set to another, and it reveals the internal closeness of the data.

Conclusions and discussions

Other points:

- The proposed approaches perform better than those of AdaBoost and Bagging.
- Selection of the parameters need further discussion and theoretical analysis.
- They only test the base learning algorithm C4.5 in this work, and other base learning algorithms will be tested in our future work.