On Error Correlation and Accuracy of NN Ensemble Classifiers

Carlotta Domeniconi, Bojun Yan
SIAM International Conference on Data Mining, 2005

Ensembles

➢ An ensemble is a collection of learning machines (or agents) that operate to solve a machine learning problem (supervised/unsupervised);

➢ T. Dietterich: “The task of improving classification accuracy by learning ensembles of classifiers is one of the most important directions in machine learning research.” (AI Magazine, 1997);

➢ No unified theory on ensembles: growing interest within the machine learning community.
Why Ensembles?

- In many domains it has been shown that an ensemble is often more accurate than any of the single components;
- Combining predictors can lead to significant reductions in generalization error.

When do Ensembles Work?

- An ensemble succeeds in improving the accuracy of the whole when the components are diverse and accurate;
- Diversity: To ensure that the agents make uncorrelated errors;
- Accuracy: To avoid poor components to obtain the majority of votes.
When do Ensembles Work? (contd.)

- To obtain the required properties: Train the individual components on different sets of data, acquired by sampling from the original training set;

- **Bagging** [Breiman, 96] and **Boosting** [Freund & Schapire, 96] are successful ensemble iterative methods for improving the predictive power of classifier learning systems.

Bagging

- Uses sampling with replacement;

- Generates multiple classifiers trained on the different bootstrapped training sets;

- To classify an instance:
  - A vote for each class $j$ is recorded by every classifier that chooses it;
  - The class with the most votes is chosen by the aggregating scheme.
Boosting

- Uses adaptive sampling;
- Uses all instances at each iteration;
- Maintains a weight for each instance, that reflects its importance as a function of the errors made by previously generated hypotheses;
- Aggregation is done by voting, but with different voting strengths to classifiers based on their accuracy.

Bagging vs. Boosting

- Experimental evidence proved that both bagging and boosting are effective in reducing generalization errors (e.g., with CART, C4.5);
- Boosting provides in general higher improvements;
- This behavior can be explained in terms of the bias-variance components of the generalization error.
Bagging vs. Boosting (contd.)

\[ E[(\hat{\theta} - \theta)^2] = E[(\hat{\theta} - E(\hat{\theta}))^2] + (E(\hat{\theta}) - \theta)^2 \]

- The objective of combination is to reduce variance, that is what both bagging and boosting achieve.

- In addition, boosting challenges the weak learner algorithm to perform well on the harder examples, thereby reducing also the bias.

Nearest Neighbor Ensemble

- Bootstrapping the data is not effective for stable classifiers;

- NN methods are very robust with respect to variations of the training data;

- As a consequence, bootstrapping the data is not effective with NN classifiers.
Nearest Neighbor Ensemble

- Suppose the weak learner is the NN classifier;
- It has been shown [Breiman, 96] that the probability that any given training point is included in a data set bootstrapped by bagging is approximately 63.2%;
- It follows: the nearest neighbor will be the same in 63.2% of the classifiers.
- Thus: errors are highly correlated. Bagging becomes ineffective!

Nearest Neighbor Ensemble

- In contrast, NN methods are very sensitive to input features (i.e., highly intolerant to irrelevant features), and to the chosen distance function.
- Then, the idea is to exploit the instability of NN classifiers with respect to different choices of features to generate a diverse set of NN classifiers with (possibly) uncorrelated errors.
Basic Idea

To design an effective NN ensemble:

- Use different feature subsets to build the component classifiers;
- To achieve both diversity and accuracy, we perform adaptive sampling over the feature space;

Related Work

- Each nearest neighbor classifier has access only to a random subset of features [Ho98,Bay99];
- **Pros**: Can increase diversity without increasing error rates. Thus: accuracy improvement;
- **Cons**: No guarantee that discriminant features are selected. Thus: voting can increase the generalization error.
Our Solution

- To reduce the risk of discarding discriminant information, we perform adaptive sampling over the feature space;
- To keep the bias of individual classifiers low, we use feature relevance to guide the sampling process;
- This approach can lead to accurate classifiers in disagreement with each other;
- Effective for problems in high dimensions.

Learning Feature Relevance

- We use the ADAMENN algorithm [Domeniconi et al., PAMI 02]:
- It uses the Chi-squared distance to estimate to which extent each dimension can predict class posterior probabilities;
- Features are weighted according to their estimated local relevance;
- Provides a local flexible metric for computing neighborhoods.
Modified “weighted” neighborhoods

Red=Class1, Green=Class2, Black=Query’s neighbors

Initially: \( w_x = w_y \)

Finally: \( w_x < w_y \)

Chi-Squared Distance

\[
D(x, x_0) = \sum_{j \in \{+, -\}} (P(j \mid x) - P(j \mid x_0))^2
\]

\[
D(x, x_0) = \sum_{j \in \{+, -\}} \frac{(P(j \mid x) - P(j \mid x_0))^2}{P(j \mid x_0)}
\]

\[
P(+ \mid x) = 1 \quad P(+ \mid x_0) = 0
\]

Minimize: \( E[(r^*(x_0) - r(x_0, x))^2] \)
Adaptive Sampling

- The weights credited to features by ADAMENN are values in \((0, 1)\) and their sum equals 1;
- Thus: they define a probability distribution over the feature space that can be employed in our adaptive sampling mechanism;
- For each test point and each classifier of the ensemble, any given feature has a non zero probability to be selected;
- A certain level of diversity among classifiers is guaranteed.

Putting All Together

- **Input**: Number-of-Classifiers \((NoC)\), Number-of-Features \((NoF)\), \(k\), test point \(x\);
  - Compute the weight vector \(w\) reflecting feature relevance at \(x\);
    - For 1 to \(NoC\):
      - Sample \(NoF\) features with or without replacement, according to the probability distribution given by the weight vector \(w\) (adaptive sampling);
      - Use selected features only (and their weights) to compute the \(k\) closest neighbors;
      - Classify test point using kNN rule;
    - Apply the voting scheme in use to the \(NoC\) classifiers.
Voting Methods

- **Simple** majority vote;
- **Count**: Delay the class membership decision until the aggregation phase: select the class with the *largest expected posterior probability* in the ensemble;
- **Borda**: Positional-scoring technique. Each candidate class gets 0 points for each last place vote received, ..., and so on up to $C-1$ points for each first place vote. The class with the largest point total wins.

Experiments

- We compare Random and Weight-Driven feature subset methods;
- $NoC = 200$; $NoF = 1,...,DIM$; $k=1,...,5$;
- Leave-One-Out cross-validation was used to generate training and test data in each classifier;
- Average error rates (over 10 runs).
## Error rates

<table>
<thead>
<tr>
<th>Dim-N-C</th>
<th>liver (6-345-2)</th>
<th>ionosphere (34-351-2)</th>
<th>spectf-test (44-267-2)</th>
<th>lung (54-32-3)</th>
<th>sonar (60-208-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>kNN</td>
<td>32.5</td>
<td>13.7</td>
<td>23.6</td>
<td>50.0</td>
<td>12.5</td>
</tr>
<tr>
<td>ADAMENN</td>
<td>30.7</td>
<td>7.1</td>
<td>19.1</td>
<td>37.5</td>
<td>9.1</td>
</tr>
<tr>
<td>Random (S)</td>
<td>29.4 (0.5)</td>
<td>5.8 (0.2)</td>
<td>20.2 (0.4)</td>
<td>45.0 (0.5)</td>
<td>10.5 (0.3)</td>
</tr>
<tr>
<td>Random (C)</td>
<td>28.6 (0.5)</td>
<td>5.7 (0.2)</td>
<td>19.9 (0.4)</td>
<td>45.3 (0.5)</td>
<td>10.3 (0.3)</td>
</tr>
<tr>
<td>Random (B)</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weight (S)</td>
<td>29.3 (0.5)</td>
<td>6.3 (0.2)</td>
<td>17.6 (0.4)</td>
<td>35.0 (0.5)</td>
<td>8.3 (0.3)</td>
</tr>
<tr>
<td>Weight (C)</td>
<td>29.9 (0.5)</td>
<td>6.3 (0.2)</td>
<td>17.7 (0.4)</td>
<td>32.5 (0.5)</td>
<td>8.3 (0.3)</td>
</tr>
<tr>
<td>Weight (B)</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td></td>
<td>30.9 (0.5)</td>
</tr>
</tbody>
</table>

## Ionosphere Data (34–351–2)

![Graph showing error rates vs. number of selected features](image-url)

- Random-simple: minimum at 5 features: 0.058
- Random-counting: minimum at 5 features: 0.057
- Weight-simple: minimum at 8 features: 0.063
- Weight-counting: minimum at 7 features: 0.063
- ADAMENN: error rate 0.071
- kNN: error rate 0.137
Spectf-test Data (44-267-2)

Lung Data (54-32-3)
Sonar Data (60-208-2)

Measure of Diversity

Kappa statistic $\kappa$ [Margineantu et al, 1997]:

$h_a, h_b$: two classifiers;

$N_{ij} =$ number of examples $x$ for which $h_a(x) = i$ and $h_b(x) = j$

$$\Theta_1 = \frac{\sum_{i=1}^{C} N_{ii}}{n}$$

Probability that the two classifiers agree

$$\Theta_2 = \sum_{i=1}^{C} \left( \frac{\sum_{j=1}^{C} N_{ij}}{n} \right) \left( \frac{\sum_{j=1}^{C} N_{ji}}{n} \right)$$

Probability that the two classifiers agree by chance

$$\kappa = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}$$
Kappa-error: spectf-test

Kappa-error: lung
**Results**

- **Our Weight-Driven approach offers accuracy improvements for the data sets with a larger number of dimensions** *(spectf-test, lung, sonar)*;

- **Bootstrapping features using an “intelligent” distance metric takes advantage of the high dimensionality of the data**;

- **The Weight-Driven approach shows a robust behavior as the number of selected features increases**.
Results (cont.)

- **Drawback of the Random approach:** as the fraction of selected features *not* carrying discriminant information increases, poor classifiers are generated, and the voting increases the generalization error (*ionosphere, spectf-test, sonar*).

- The Weight-driven technique offers a lower diversity. However, the “intelligent” metric employed by the Weight-driven technique allows to reduce bias, and thus achieve a better error rate.

Reduction of Error Correlations

- We explore the possibility of decorrelating errors by introducing new elements of diversification among the NN classifiers;

- We face the challenge of reaching a trade-off between error decorrelation and accuracy in the context of NN classifiers.
Reduction of Error Correlations

**Technique 1:**
- Each classifier customizes the number of selected features at each query point:

  Sort the weight components of \( w_0 \) in non-increasing order: \( w_{01} \geq \cdots \geq w_{oq} \);
  Number of selected features at \( x_0 \) is \( NoF_0 \) such that:

  \[
  \sum_{i=1}^{NoF_0} w_{oi} \leq f \quad \text{and} \quad \sum_{i=1}^{NoF_0+1} w_{oi} > f \quad f \in (0,1)
  \]

  We used \( f = 0.6, 0.8, 0.9 \)

Reduction of Error Correlations

**Technique 2:**
- Ensemble of a mixture of Random and Weight-driven classifiers;

- Two percentage combinations were tested: 50% of each kind; 60% Weight-driven and 40% Random.
Measure of Error Correlation

Correlation of errors of two classifiers (1 and 2) on each class $i$:
\[
\delta^{i}_{1,2} = \frac{\text{cov}(\eta_{j}^{i}(x), \eta_{j}^{i}(x))}{\sigma_{\eta_{j}^{i}} \sigma_{\eta_{j}^{i}}}
\]

$\eta_{j}^{i}(x)$: error value on $x \in C_i$ of classifier $j$

$\sigma_{\eta_{j}^{i}}$: standard deviation of $\eta_{j}^{i}(x)$ $\forall x \in C_i$

To account for all classes:
\[
\delta_{1,2} = \sum_{i=1}^{C} \delta^{i}_{1,2} P(i)
\]

Equal priors:
\[
\delta_{1,2} = 1/C \sum_{i=1}^{C} \delta^{i}_{1,2}
\]

Total error correlation between classifiers 1 and 2

Average Error Correlation and
Error Rates: Liver Data

<table>
<thead>
<tr>
<th></th>
<th>Error Correlation</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.12</td>
<td>29.4</td>
</tr>
<tr>
<td>Weight</td>
<td>0.23</td>
<td>29.3</td>
</tr>
<tr>
<td>Weight-C ($f=0.9$)</td>
<td>0.74</td>
<td>30.3</td>
</tr>
<tr>
<td>Weight-C ($f=0.8$)</td>
<td>0.41</td>
<td>31.4</td>
</tr>
<tr>
<td>Weight-C ($f=0.6$)</td>
<td>0.21</td>
<td>31.6</td>
</tr>
<tr>
<td>Mixture</td>
<td>0.11</td>
<td>30.8</td>
</tr>
</tbody>
</table>
### Average Error Correlation and Error Rates: Sonar Data

<table>
<thead>
<tr>
<th></th>
<th>Error Correlation</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.34</td>
<td>10.5</td>
</tr>
<tr>
<td>Weight</td>
<td>0.69</td>
<td>8.3</td>
</tr>
<tr>
<td>Weight-C (f=0.9)</td>
<td>0.72</td>
<td>8.7</td>
</tr>
<tr>
<td>Weight-C (f=0.8)</td>
<td>0.66</td>
<td>10.2</td>
</tr>
<tr>
<td>Weight-C (f=0.6)</td>
<td>0.42</td>
<td>11.4</td>
</tr>
<tr>
<td>Mixture</td>
<td>0.43</td>
<td>8.1</td>
</tr>
</tbody>
</table>

### Conclusions

- We have introduced a mechanism to generate an effective and diverse ensemble of NN classifiers;

- Results show the potential of combining ensembles with locally adaptive metrics to effectively dodge the sparsity of high dimensional data;

- To reach a good balance between error decorrelation and accuracy, multiple adaptive mechanisms for sampling in feature space will be considered.