Ensembling learning applied to ACLR: Artificial Characters Learning Problem

Wangzhihui Mei 2019124044 6603385

CCNU-UOW JI

1 Dataset

This database has been artificially generated by using a first order theory which describes the structure of ten capital letters of the English alphabet and a random choice theorem prover which accounts for etherogeneity in the instances. The capital letters represented are the following: A, C, D, E, F, G, H, L, P, R. Each instance is structured and is described by a set of segments (lines) which resemble the way an automatic program would segment an image.

Each instance is stored in a separate file whose format is the following:

CLASS	OBJNUM	TYPE	XX1	YY1	XX2	YY2	SIZE	DIAG
1	0	line	0	0	0	13	13.00	45.28
1	1	line	20	0	22	15	15.13	45.28
1	2	line	0	13	22	15	22.09	45.28
1	3	line	0	13	0	27	14.00	45.28
1	4	line	22	15	23	39	24.02	45.28
1	5	line	0	27	23	39	25.94	45.28

Figure 1: The generated characters

where CLASS is an integer number indicating the class as described below, OBJNUM is an integer identifier of a segment (starting from 0) in the instance and the remaining columns represent attribute values. [3].

The generated character image is like Figure 1

2 Model construction and parameter optimization

2.1 Data pre-process

The character described by segments is represented as the vertex pair, we transform them to binary grid (12×8), like Figure 2.

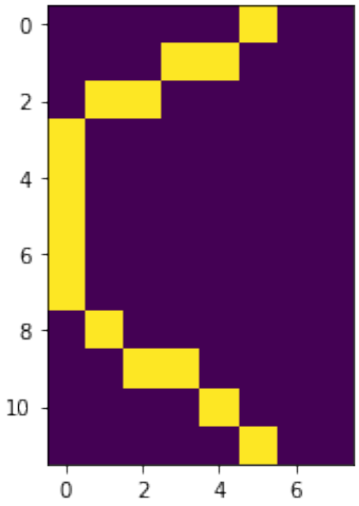

Figure 2: The representation of "C"

Then the problem is transformed into one like the MNIST classification problem. The feature is the flattened 0-1 pixel, whose dimension is 96×1, the label is "A", "C", "D", "E", "F", "G", "H", "L", "P", "R", which correspond to the numbers 0 through 9.

We got 6000 training data from the primitive dataset. We shuffled them and use 70% of the data as training data, 30% as test data. The label distribution of training data is like Figure 3

2.2 Model construction

I applied linear models(Linear Regression, SGD, SVM), bayes model, neural network model(MLP), decision tree model, random forest model, extra tree model, and them ensemble them as voting model.

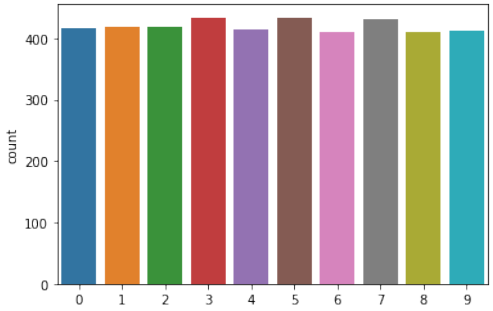

Figure 3: The label distribution of training data

2.2.1 Logistic Regression Model

Regarding logistic regression, it can be summarized in one sentence: Logistic regression assumes that the data obey Bernoulli distribution, and uses the gradient likelihood method to solve the parameters through the maximum likelihood function method to achieve the purpose of classifying the data.

Our parameter tuning:

- multi_class: In this case, the pattern only belong to one class, we set it to 'multinomial'
- max_iter: Set the maximum number of iterations taken for the solvers to converge to 1000.
- solver: For multiclass problems, only newton-cg, sag, saga and lbfgs handle multinomial loss; liblinear is limited to one-versus-rest schemes, we tried and selected 'saga'.
- n_jobs: Set to -1 to use all CPU cores to calculation.
- penalty: Set to 'l1' to get better weight sparsity.

2.2.2 SGD

This model is Linear classifiers with SGD training. Stochastic gradient descent(SGD) uses only one sample at a time to calculate the gradient of the objective function. The formula is:

$$\theta = \theta - \alpha \nabla_\theta J\left(\theta; x^i, j^i\right)$$

The cost function is:

$$Jtrain(\theta) = \frac{1}{m}\sum i = 1^m \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{m}\sum i = 1^m cost(\theta; x^i, j^i) \quad (1)$$

Where cost $\left(\theta; x^i, j^i\right) = \frac{1}{2}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

Use the loss function of each sample to derive the partial gradient of θ to get the corresponding gradient to update θ:

$$\theta_j = \theta_j + \alpha(y^i - h_\theta(x^i))xj^i$$

Stochastic gradient descent is iteratively updated once by each sample, the calculation is very fast and suitable for updating the model online. However, the problem with SGD is that there is more noise than BGD, so that SGD does not go to the overall optimization direction every iteration, and the search process

in the solution space seems blind.

When using SGD we can choose to start with a learning rate of 1.0 or 0.1. Check it on the verification set at intervals.If the cost does not drop, the learning rate is halved.

Our parameter tuning:

- loss: Set to log loss to give logistic regression, a probabilistic classifier.
- penalty: Set to 'l2'
- n_job: Set to -1 to use all processors.
- learning_rate: optimal: $eta = 1.0/(alpha * (t + t0))$ where $t0$ is chosen by a heuristic proposed by Leon Bottou[1].

2.2.3 SVM

We applied grid search to find the optimized C and $gamma$. We applied binary search method and narrow the search iteratively. The we got the optimized $C = 4.5$ and $gamma = 0.06$, see in Figure 4.

```
C 4.5
break_ties False
cache_size 200
class_weight None
coef0 0.0
decision_function_shape ovr
degree 3
gamma 0.06
kernel rbf
max_iter -1
probability False
random_state None
shrinking True
tol 0.001
verbose False
[0.97738095 0.97261905 0.98333333 0.97619048 0.98214286]
Accuracy: 0.98 (+/- 0.01)
precition is :  97.66666666666667 %
```

Figure 4: Grid search SVM parameters

My parameter tuning:

- C: Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive.
- kernel: 'rbf'
- gamma: 0.06, Kernel coefficient for rbf, poly and sigmoid.

This is the number of support vectors of each class accordingly: [389 90 109 207 285 203 308 44 236 199].

2.2.4 Decision Tree

Decision tree is a machine learning method using statistical probability analysis. It represents a kind of mapping between object attributes and object values, and each one in the tree represents a judgment

condition indicating an object attribute, which corresponds to an object that meets the condition. The leaves of the tree represent the prediction results of the object.

We use grid search to find the best parameters, see in Figure 5.

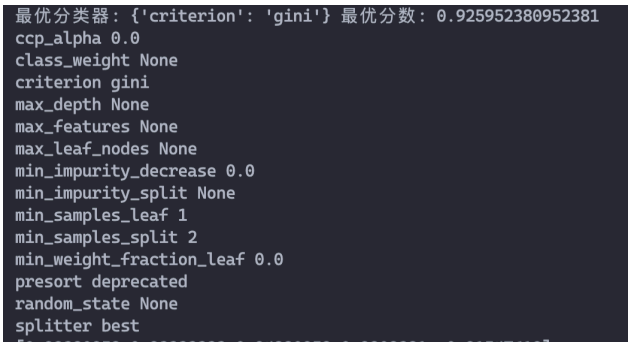

Figure 5: Grid search decision tree parameters

My parameter tuning:

- criterion: 'gini'. Supported criteria are gini for the Gini impurity.
- min_samples_leaf: 1. The minimum number of samples required to be at a leaf node.
- min_samples_split: 2. The minimum number of samples required to split an internal node.

2.2.5 Random Forest

Random forest is to build a forest in a random way. There are many decision trees in the forest. There is no relationship between each decision tree in the random forest. After getting the forest, when a new input sample enters, let each decision tree in the forest make a judgment to see which category this sample should belong to, and then see which category is selected the most , Predict which category this sample is.

The main focus parameter is n_estimators. Generally, the larger the n_estimator, the better the model performs, but n_estimator is too large, the boundary revenue is little, i.e., the model convergence is met. We tried 10,50,100,1000, the n_estimator setting to 100 can perform well enough and realize the balance of the performance-calculation trade-off.

My parameter tuning:

- n_estimators: 100. The number of trees in the forest.
- oob_score: True. The score outside the bag reflects the generalization ability of a model after fitting.
- criterion: gini. The function to measure the quality of a split

2.2.6 Extratree

ET or Extra-Trees (Extremely Randomized Trees) were proposed by Pierre Geurts et. al. in 2006[2]. This algorithm is very similar to the random forest algorithm, and is composed of many decision trees. But there are two main differences between this algorithm and random forest:

1. The random forest uses the Bagging model, and ET uses all training samples to obtain each decision tree, that is, each decision tree applies the same all training samples;

2. Random forest obtains the best bifurcation attribute in a random subset, and ET obtains the bifurcation value completely randomly, so as to bifurcate the decision tree.

The tuning process is similar to Random Forest:

- n_estimators: 100. The number of trees in the forest.
- oob_score: True. Whether to use out-of-bag samples to estimate the generalization accuracy.
- criterion: gini. The function to measure the quality of a split

2.2.7 MLP

We use a 4-layer MLP as classifier, with the hidden layer of 96-node and 192-node.

We applied some tricks to it: Integer multiple number of hidden layer may perform better.

Parameter tuning:

- solver='sgd',
- activation='relu',
- learning_rate_init=.1,

2.3 Model Ensembling

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction. As long as the base models are diverse and independent, the prediction error of the model decreases when the ensemble approach is used. The approach seeks the wisdom of crowds in making a prediction. Even though the ensemble model has multiple base models within the model, it acts and performs as a single model. Most of the practical data mining solutions utilize ensemble modeling techniques. Chapter 4 Classification covers the approaches of different ensemble modeling techniques and their implementation in detail.

In this task, I applied Voting method. Voting is a combination strategy for classification problems in integrated learning. The basic idea is to select the most output class among all machine learning algorithms.

There are two types of output of classification machine learning algorithms: one is to directly output class labels, and the other is to output class probabilities. Using the former to vote is called hard voting (majority / hard voting), and using it to classify is called soft voting (Soft voting). VotingClassifier in sklearn is the implementation of voting method.

2.3.1 Hard mode

The final result is determined by the minority obeying the majority; I applied all classifiers above to ensemble to one VotingClassifier. The structure is like Figure 6

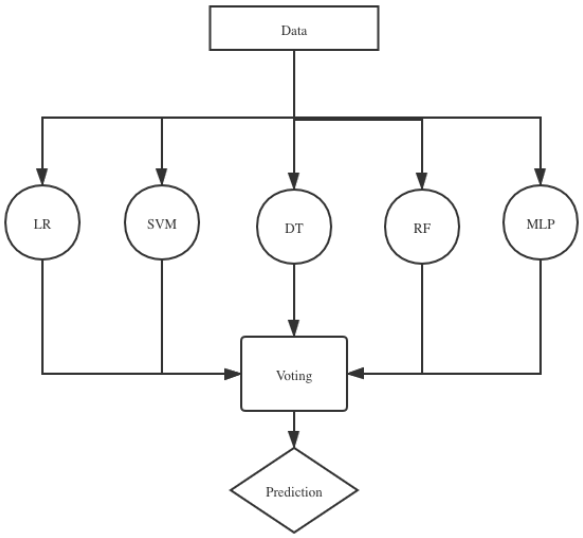

Figure 6: The voting classifier

2.3.2 Soft mode

All the model prediction samples are the estimated value of the probability of certain categories, and the corresponding type with the highest probability is the final prediction result; This require that each model of the set can estimate the probability. We have to enable the probability output of SVM.

I applied elimination strategy to the model, that is, minimizing the weight of poor performed classifier and turn up the weight of well performed classifier. Here, I use the average f1-score if classifiers as the weight vector.

3 Evaluation

The dataset is divided into training and testing sets, train:test = 7:3.

The traditional F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall. The number of positive sample is P, The number of negative sample is N

The recognition rate $Acc = \frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{P+N}$

Recall $= \frac{TP}{TP+FN} = \frac{TP}{P}$

Precision $= \frac{TP}{TP+FP}$

$F_1 = \frac{2TP}{2TP+FN+FP} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Recall reflects the classification model H's ability to identify positive samples. The higher the recall, the stronger the model's ability to identify positive samples. Precision reflects the model's ability to distinguish negative samples. The higher the precision, the better the model's discrimination of negative samples. The stronger the ability. F1-score is a combination of the two. The higher the F1-score, the more robust the classification model.

Logistic Regression Classifier

Accuracy: 0.94 (+/- 0.01)
The classification report:

	precision	recall	f1-score	support
A	0.90	0.92	0.91	183
C	0.96	1.00	0.98	182
D	0.95	0.97	0.96	182
E	0.97	0.93	0.95	167
F	0.92	0.82	0.87	185
G	0.99	0.95	0.97	166
H	0.96	0.92	0.94	190
L	1.00	1.00	1.00	168
P	0.82	0.92	0.87	190
R	0.96	0.98	0.97	187
accuracy			0.94	1800
macro avg	0.94	0.94	0.94	1800
weighted avg	0.94	0.94	0.94	1800

Figure 7: The LR performance

SVM

Accuracy: 0.98 (+/- 0.00) [SVM]
The classification report:

	precision	recall	f1-score	support
A	0.96	0.97	0.97	183
C	0.99	0.99	0.99	182
D	0.98	0.99	0.99	182
E	0.99	0.99	0.99	167
F	0.94	0.92	0.93	185
G	0.99	0.99	0.99	166
H	0.98	0.97	0.98	190
L	1.00	1.00	1.00	168
P	0.93	0.93	0.93	190
R	0.99	1.00	0.99	187
accuracy			0.98	1800
macro avg	0.98	0.98	0.98	1800
weighted avg	0.98	0.98	0.98	1800

Figure 8: The SVM performance

SGD

```
Accuracy: 0.93 (+/- 0.01) [SGD]
The classification report:
              precision    recall  f1-score   support

           A       0.95      0.89      0.92       183
           C       0.94      1.00      0.97       182
           D       0.94      0.96      0.95       182
           E       0.98      0.95      0.96       167
           F       0.90      0.76      0.82       185
           G       0.99      0.93      0.96       166
           H       0.95      0.95      0.95       190
           L       0.99      1.00      1.00       168
           P       0.78      0.93      0.85       190
           R       0.97      0.98      0.97       187

    accuracy                           0.93      1800
   macro avg       0.94      0.93      0.93      1800
weighted avg       0.94      0.93      0.93      1800
```

Figure 9: The SGD performance

Decision Tree

```
Accuracy: 0.93 (+/- 0.01) [Decision Tree]
The classification report:
              precision    recall  f1-score   support

           A       0.87      0.87      0.87       183
           C       0.95      0.99      0.97       182
           D       0.92      0.96      0.94       182
           E       0.95      0.92      0.94       167
           F       0.93      0.90      0.91       185
           G       0.99      0.94      0.96       166
           H       0.94      0.89      0.91       190
           L       0.96      0.99      0.98       168
           P       0.89      0.93      0.91       190
           R       0.97      0.96      0.97       187

    accuracy                           0.94      1800
   macro avg       0.94      0.94      0.94      1800
weighted avg       0.94      0.94      0.93      1800
```

Figure 10: The Decision Tree performance

Random Forest

```
Accuracy: 0.96 (+/- 0.01) [Random Forest]
The classification report:
              precision    recall  f1-score   support

           A       0.96      0.95      0.95       183
           C       0.96      0.99      0.98       182
           D       0.98      0.99      0.99       182
           E       0.99      0.99      0.99       167
           F       0.96      0.93      0.95       185
           G       0.99      0.95      0.97       166
           H       0.96      0.95      0.96       190
           L       1.00      0.99      1.00       168
           P       0.92      0.95      0.94       190
           R       0.98      0.98      0.98       187

    accuracy                           0.97      1800
   macro avg       0.97      0.97      0.97      1800
weighted avg       0.97      0.97      0.97      1800
```

Figure 11: The Random Forest performance

Extratree

```
Accuracy: 0.96 (+/- 0.01) [Extra Trees]
The classification report:
              precision    recall  f1-score   support

           A       0.91      0.93      0.92       183
           C       0.97      0.99      0.98       182
           D       0.98      0.99      0.99       182
           E       0.99      0.98      0.99       167
           F       0.95      0.95      0.95       185
           G       0.99      0.96      0.97       166
           H       0.95      0.93      0.94       190
           L       0.98      1.00      0.99       168
           P       0.95      0.94      0.95       190
           R       0.98      0.99      0.99       187

    accuracy                           0.97      1800
   macro avg       0.97      0.97      0.97      1800
weighted avg       0.97      0.97      0.97      1800
```

Figure 12: The Extratree performance

MLP

```
Accuracy: 0.97 (+/- 0.00) [MLP]
The classification report:
              precision    recall  f1-score   support

           A       0.94      0.96      0.95       183
           C       0.99      1.00      0.99       182
           D       0.98      0.98      0.98       182
           E       0.98      0.96      0.97       167
           F       0.93      0.93      0.93       185
           G       1.00      0.98      0.99       166
           H       0.98      0.95      0.96       190
           L       0.99      1.00      0.99       168
           P       0.93      0.93      0.93       190
           R       0.98      0.99      0.99       187

    accuracy                           0.97      1800
   macro avg       0.97      0.97      0.97      1800
weighted avg       0.97      0.97      0.97      1800
```

Figure 13: The MLP performance

Ensembling (Hard voting)

```
Accuracy: 0.97 (+/- 0.00) [Ensemble(hard voting)]
The classification report:
              precision    recall  f1-score   support

           A       0.97      0.97      0.97       183
           C       0.98      1.00      0.99       182
           D       0.98      0.99      0.99       182
           E       0.99      0.98      0.98       167
           F       0.96      0.93      0.94       185
           G       1.00      0.98      0.99       166
           H       0.98      0.97      0.98       190
           L       1.00      1.00      1.00       168
           P       0.93      0.95      0.94       190
           R       0.98      0.99      0.99       187

    accuracy                           0.98      1800
   macro avg       0.98      0.98      0.98      1800
weighted avg       0.98      0.98      0.98      1800
```

Figure 14: The Ensembling (Hard voting) performance

Ensembling (Soft voting)

```
Accuracy: 0.97 (+/- 0.00) [Ensemble(soft voting)]
The classification report:
              precision    recall  f1-score   support

           A       0.96      0.97      0.96       183
           C       0.99      1.00      0.99       182
           D       0.98      0.98      0.98       182
           E       0.98      0.96      0.97       167
           F       0.95      0.94      0.94       185
           G       0.99      0.99      0.99       166
           H       0.98      0.97      0.97       190
           L       1.00      1.00      1.00       168
           P       0.94      0.95      0.94       190
           R       0.98      0.99      0.99       187

    accuracy                           0.97      1800
   macro avg       0.97      0.97      0.97      1800
weighted avg       0.97      0.97      0.97      1800
```

Figure 15: The Ensembling (Soft voting) performance

3.1 Analysis

Generally, The SVM and Hard voting Ensembling classifier have the best performance. Hard voting classifier perform better when precision/recall bias of each label is taken into concern. The Soft voting classifier, MLP and Random Forest also have ideal performance. The SGD, LR and Decision tree model is too simplified to perform well in this task.

4 Conclusion

We showed the Ensembling can improve the performance of the classifiers, with which the classification can resist the disturbance of some noise. The voting ensembling is a kind of bagging, which mean that we can perform parallel learning in FPGA or GPU to accelerate the computing.

References

[1] Bottou, L. (2003). Stochastic learning. In *Summer School on Machine Learning*, pages 146–168. Springer.

[2] Geurts, P., Wehenkel, L., and d'Alché Buc, F. (2006). Kernelizing the output of tree-based methods. In *Proceedings of the 23rd international conference on Machine learning*, pages 345–352.

[3] Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.