

Comparative Experimentation of Machine Learning Classifiers

Installation

```
sh $ pip3 install -r requirements.txt
```

Running

Custom hyperparameters in a textfile i.e. `"/configs/config.txt"`.

```
sh $ python3 experiments.py ./configs/config.txt
```

A *results* folder will contain a timestamp directory with the latest results.

This work is a experiment with a number of algorithms on several datasets. The aim is to get a feeling of how well each of these algorithms works, and whether there are differences depending on the dataset.

Datasets

- Iris (http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html)
 - Handwritten digits (http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html)
 - Music (by George Tzanetakis, called "gtzan")
-

Classifiers

- k-NN (with 3 different values for k)
- Naive Bayes
- Perceptron
- Decision Trees
- SVC, LinearSVC
- Random Forests: one limited by max depth
- Full Decision Tree
- Pruned Decision Tree

For each dataset, each classifier is trained and evaluated (with parameter variations), and then evaluation metrics are computed.

Metrics

- Effectiveness: accuracy, precision
 - Efficiency: runtime for training and testing
-

Splitting technique

The holdout method with 2/3 training and the rest for testing is used once, and cross validation with 5 folds also used once.

Results

□
□
□
□
□
□
□
□
□

Description and analysis

Which classifiers work best?

Regarding the Iris dataset and the 66-33 training & test split, Naive Bayes has turned out to be the best classifier in terms of accuracy. As to the 5-folds split, k-NN (3-NN) is the best one in both mean and standard deviation. The largest k-NN and Perceptron classifiers were outperformed by the rest.

Concerning the handwritten digits dataset and the 66-33 training & test split, k-NN (3-NN) resulted as the best in terms of accuracy. As to the 5-folds split, k-NN (3-NN) is once again the best undoubtedly, in both mean and standard deviation.

Are there differences between the datasets?

On the one hand, the iris dataset has 3 different types of irises (plant with showy flowers) petal and sepal length and width, with 50 samples for each type: * Setosa * Versicolour * Virginica

Its dimensionality is 4 real, positive values.

On the other hand, the digits dataset has 1797 samples of handwritten digits from 0 to 9, a class for each digit. Each class has around 180 samples. As explained in scikit-learn user guide, normalized bitmaps of handwritten digits were extracted from a pre-printed form. 43 people contributed, 30 to the training set and 13 to the test set, with no overlapping between sets. 32x32 bitmaps are divided into non-overlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0 to 16. It also reduces dimensionality and gives invariance to small distortions. Its dimensionality is 64 integers from 0 to 16.

The music dataset contains 1000 songs, 100 songs for 10 genres, and the task is therefore to predict the genres of a song; to limit file size, the songs are only 30 second snippets, and sampled with 22 khz only. As this is audio data, feature extraction is a requirement before learning. This extraction generates different features, very simple ones containing just bpm, and more advanced ones based on advanced signal processing.

Are there differences in the efficiency measurements?

In all experiments, k-NN takes more time predicting than training. This is because k-NN is also called as lazy learner since during fitting it does nothing but save input data (there is no learning). During prediction or testing time it is actually when distance calculation happens for each test data point. The opposite observation can be seen for Perceptron times, where training calculations must take place, and prediction is rather straightforward. With this in mind, Perceptron classifier training time is seen to be significantly greater than the rest. Concerning the music dataset, there are distinctions between the four features. *data_bpm* is the only feature where pruning does not make a difference in terms of training time. For the rest of the features, training time is about three times as much as the unpruned experiment, but accuracy and precision are not significantly improved. However, testing time is no different between these two settings for all music features. Predictably, Random Forests take at least an order of magnitude more in terms of runtime than the rest of the classifiers.

How is the runtime changing with the different datasets?

It is clear that the dimensionality plays a major role in the difference in runtime between the datasets. The greater computation times for the digits dataset with respect to the ones from the iris dataset can be distinguished easily; roughly an order of magnitude greater in most cases, independently of the split chosen. With regards to the music dataset, *data_bpm* takes the least time in runtime compared to the other features throughout both splits. This is expected since it only contains one feature, the tempo, whereas *data_bpm_statistics* contains 7 other features. The comparison can be seen clearly with *data_mfcc*, which is composed of 7 values per each of the 12 coefficients (mean, median, var, min, max, skewness and kurtosis), or with *data_chroma*, which also takes these 7 values for each of the 12 chroma bins, and takes the most runtime.

Why are accuracy and precision equal?

According to sklearn's documentation:

Note that for "micro"-averaging in a multiclass setting with all labels included will produce equal precision, recall and F, while "weighted" averaging may produce an F-score that is not between precision and recall.

From <https://simonhessner.de/why-are-precision-recall-and-f1-score-equal-when-using-micro-averaging-in-a-multi-class-problem/>:

Micro averaging scheme is not prone to inaccurate values due to an unequally distributed test set (e.g. 3 classes and one of these has 98% of the samples). This is why I prefer this scheme over the macro averaging scheme. Besides micro averaging, one might also consider weighted averaging in case of an unequally distributed data set.

If there is a false positive, there will always be a false negative and vice versa, because always one class is predicted. If class A is predicted and the true label is B, then there is a FP for A and a FN for B. If the prediction is correct, i.e. class A is predicted and A is also the true label, then there is neither a false positive nor a false negative but only a true positive.

An example. Confusion matrix: $\begin{bmatrix} 13 & 0 & 0 \\ 3 & 6 & 12 \\ 0 & 0 & 16 \end{bmatrix}$

Accuracy calculation: $(13+6+16)/50 = 0.7$

Micro-precision calculation: $((13/13)*(13/50) + (6/21)*(21/50) + (16/16)*(16/50)) = 0.7$

Experiments with *data_bpm*

For both splits, the results are improved for an increasing number of NN. Naive Bayes does not improve this, nor the Perceptron, which performed worse and took longer than the previously mentioned. The Unpruned decision tree was around the levels of performance of the k-NNs, but was outperformed by the pruned classifier, for which we can conclude that the complete tree probably was overfitting. As

to the Random Forests classifiers, there are not any differences between the scores, albeit the larger training time for the 100-tree forest. There were practically no differences between the SVC and LinearSVC classifiers.

Experiments with *data_bpm_statistics*

For both splits, the results of the k-NNs, NB and Perceptron are quite like the *data_bpm* experiments. The Unpruned decision tree, which took considerably more time than the others, was around the levels of performance of the k-NNs, but this time it was not outperformed by the pruned classifier. As to the Random Forests classifiers, these were the best performing classifiers, though predictably took longer. Having 10 times the number of trees did not show considerable improvements. Once again, there were practically no differences between the SVC and LinearSVC classifiers.

Experiments with *data_chroma*

With this feature extraction, the performance in terms of accuracy and precision increase considerably compared to the last two. For both splits, the results of the k-NNs are around the same, if not worse, for an increasing amount of NN. Naive Bayes performed quite well concerning both these measures and the training time. The Perceptron classifier was the second worst, followed by the pruned decision tree which took a fifth of the training time of its unpruned version, but performed poorly. As to the Random Forests classifiers, these were once again the best performing classifiers, though predictably took longer. Having 10 times the number of trees this time did show considerable improvements, without much more training time. Once again, there were practically no differences between the SVC and LinearSVC classifiers, but outperformed all classifiers except for the Random Forests.

Experiments with *data_mfcc*

With this feature extraction, the performance in terms of accuracy and precision is the greatest considering all extractions. With mel-frequency cepstral coefficients (MFCCs), taking 12 coefficients, and for each coefficient, around 40 values per second, seems to be the best feature extraction on the music dataset. For both splits, the results of the k-NNs are worse for an increasing amount of NN, being 3 NN the best performer. Naive Bayes performed once again quite well concerning both these measures and the training time. The Perceptron classifier was the second worst, with the greatest standard deviation, followed by the SVC and LinearSVC classifiers, which presented the worse results, without a clear distinction between them. The pruned decision tree took about a fourth of the training time of its unpruned version, but performed considerably worse. As to the Random Forests classifiers, these were once again the best performing classifiers, though predictably took longer. Having 10 times the number of trees this time did show considerable improvements (the best, 0.69 accuracy, in the entire experimentation), without much more training time.