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)

Classifiers

- k-NN (with 3 different values for k)
- Naive Bayes
- Perceptron
- Decision Trees

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

See README.md

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.

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.

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.

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: `[[ 13 0 0 ] [ 3 6 12 ] [ 0 0 16 ]]`

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`