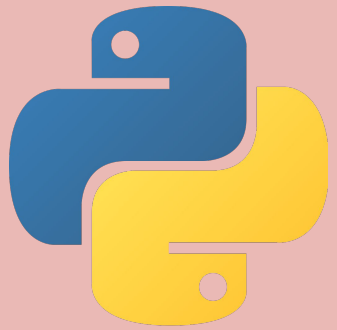


Python vs. Go

A machine learning experiment

Chiara Mezzavilla
Berlin Python Pizza
August 2019

Hello!

I’m Chiara :-)

- Coding in Python since 2012
- Coaching at OpenTechSchool since 2013
- Paid developer since 2015
- Started learning Go to have a better grasp of CS concepts
- First time conference speaker!

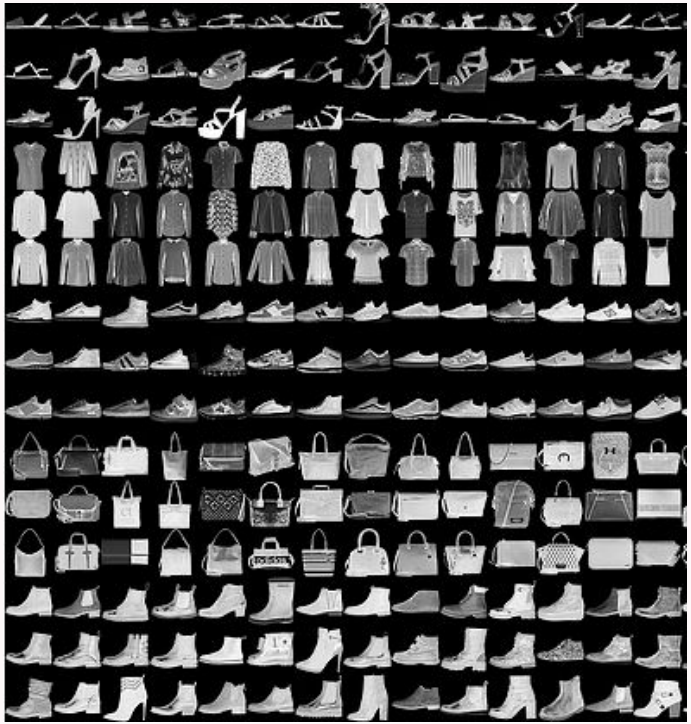
How do Python and Go compare, in classification?

Objectives:

- Get to know the Go machine learning stack (conference)
- Compare ease of use and performance with Python
- Try the Fashion MNIST dataset (contributed to by a former colleague)
- Give a talk at work :-)

Fashion MNIST replaces the digits MNIST dataset

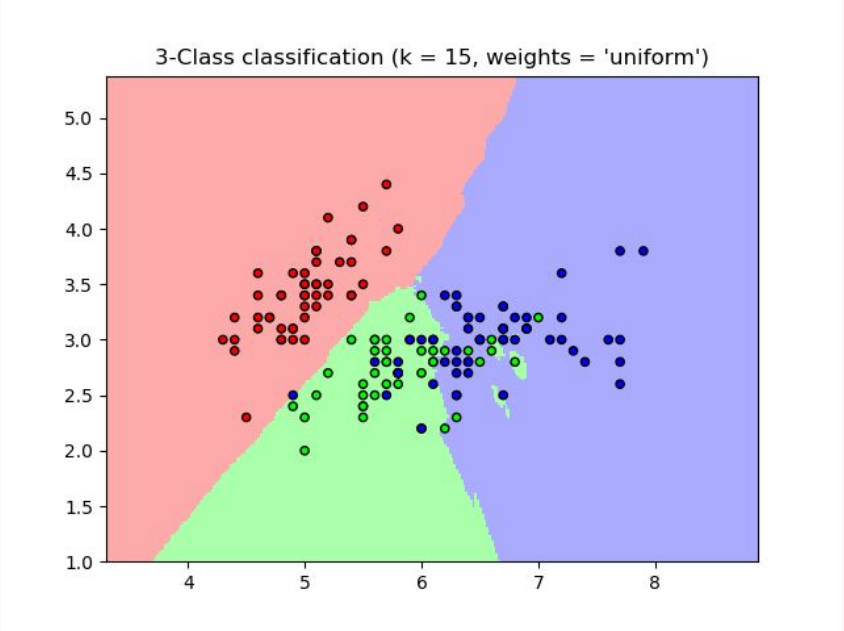
- 60000 records in the training dataset, 10000 in the test dataset (but I used a smaller one to speed up)
- Images are represented by pixel color (greyscale)
- 10 labels, corresponding to categories of clothing, shoes, and accessories (t-shirts, pullovers, ankle boots, bags, etc.)
- Replaces the digits datasets, considered “too easy”



Source: https://github.com/zalandoresearch/fashion-mnist

KNN is a simple classification algorithm

- KNN looks at the “neighbors” to predict the label
- Easier to understand in a 2-d representation like that --->
- Doesn’t make assumption on the distribution of data
- Doesn’t do any actual learning, simply saves the training data data and compares the test data with it



Source: scikit-learn.org

First finding (surprise!): Go was easier to write

- It was tricky to load the data because of different assumptions
- Golearn provides a function to load data, which assumes that the rightmost column is the label
- Scikit-learn accepts numpy arrays - you need to parse the csv files properly (unless one loads a built-in dataset)

main.go

```go
package main

import (
  "fmt"
  "github.com/pkg/profile"
  "github.com/sjwhitworth/golearn/base"
  "github.com/sjwhitworth/golearn/evaluation"
  "github.com/sjwhitworth/golearn/knn"
)

func main() {

  fmt.Println("Loading data...")
  trainData, err := base.ParseCSVToInstances("data/fashion-mnist_train.csv", true)
  if err != nil {
    panic(err)
  }
  testData, err := base.ParseCSVToInstances("data/fashion-mnist_testsmall.csv", true)
  if err != nil {
    panic(err)
  }
  fmt.Println("Very data, much mnist, wow")

  classifier := knn.NewKnnClassifier("euclidean", "kdtree", 5)
  fmt.Println("Many training...")
  err = classifier.Fit(trainData)
  if err != nil {
    panic(err)
  }

  fmt.Println("Very predict...")
  predictions, err := classifier.Predict(testData)
  if err != nil {
    panic(err)
  }

  confusionMat, err := evaluation.GetConfusionMatrix(testData, predictions)
  if err != nil {
    panic(fmt.Sprintf("Unable to get confusion matrix: %s", err.Error()))
  }
  fmt.Println(evaluation.GetSummary(confusionMat))
}
```

py_fashion_ml.py

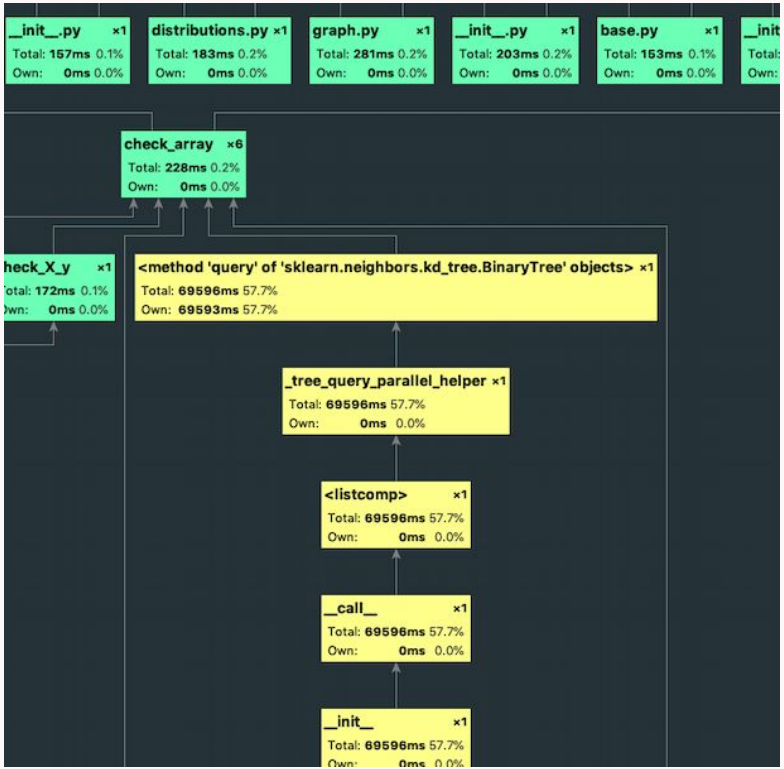
```python
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

print("Loading data...")
train_data = np.loadtxt(fname='data/fashion-mnist_train.csv',
                        delimiter=',',
                        skiprows=1)
test_data = np.loadtxt(fname='data/fashion-mnist_testsmall.csv',
                       delimiter=',',
                       skiprows=1)
print("Very data, much mnist, wow")

# euclidean is the default distance
neigh = KNeighborsClassifier(n_neighbors=5, algorithm='kd_tree')

print("Many training...")
neigh.fit(train_data[:, :-1], train_data[:, -1])

print("Very predict...")
prediction = neigh.predict(test_data[:, :-1])

print("Classification summary")
print(classification_report(y_pred=prediction, y_true=test_data[:, -1]))

```

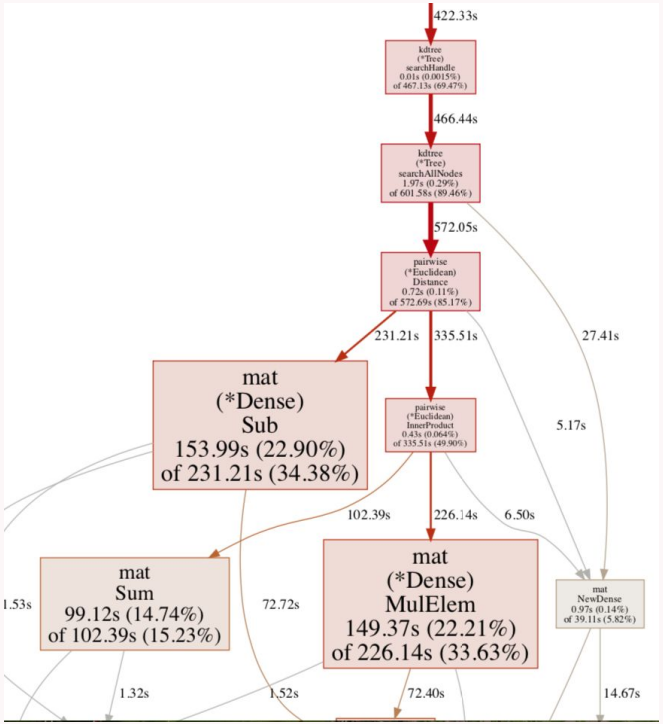
Second finding: accuracy was similar

```
time ./main
Loading data...
Very data, much mnist, wow
Many training...
Very predict...
Reference   True        False     True      Precision Recall  F1 Score
Classn      Positives   Positives Negatives
---------   ----------  --------- --------- --------- ------  --------
7           51          4         443       0.9273    0.9808  0.9533
9           45          3         449       0.9375    0.9574  0.9474
0           41          15        431       0.7321    0.7736  0.7523
8           40          4         452       0.9091    0.9302  0.9195
4           45          7         437       0.8654    0.8182  0.8411
6           34          20        430       0.6296    0.6939  0.6602
5           45          1         446       0.9783    0.8654  0.9184
1           39          0         459       1.0000    0.9750  0.9873
2           50          9         427       0.8475    0.7937  0.8197
3           41          5         449       0.8913    0.9111  0.9011
Overall accuracy: 0.8637
./main  554,05s user 2,83s system 103% cpu 9:00,66 total
```

```
time python py_fashion_ml.py
Loading data...
Very data, much mnist, wow
Many training...
Very predict...
Classification summary
              precision    recall  f1-score   support

         0.0       0.75      0.84      0.79        98
         1.0       1.00      0.96      0.98        89
         2.0       0.76      0.82      0.79       111
         3.0       0.90      0.90      0.90       104
         4.0       0.80      0.79      0.80        96
         5.0       0.99      0.86      0.92        96
         6.0       0.68      0.60      0.64       100
         7.0       0.91      0.98      0.94       102
         8.0       0.95      0.94      0.94        98
         9.0       0.93      0.95      0.94       105

    accuracy                           0.86       999
   macro avg       0.87      0.86      0.86       999
weighted avg       0.87      0.86      0.86       999

python py_fashion_ml.py  105.02s user 1.67s system
99% cpu 1:46.78 total
```

Third finding (surprise?): C makes Python faster





Thank you!

Find me on Twitter @ChiaraM_87

See also:

- Scikit-learn - https://github.com/scikit-learn/scikit-learn
- Golearn - https://github.com/sjwhitworth/golearn
- Fashion MNIST - https://github.com/zalandoresearch/fashion-mnist
- My code - https://github.com/samurang87/go-fashion-ml