wu1

October 1, 2023

[1]:
```
from numpy import *
from pylab import *
import dumbClassifiers, datasets, runClassifier
```

[ ]:
```
```

[2]:
```
h = dumbClassifiers.AlwaysPredictOne({})
h.train(datasets.TennisData.X, datasets.TennisData.Y)
h.predictAll(datasets.TennisData.X)
```

[2]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

[3]:
```
mean((datasets.TennisData.Y > 0) == (h.predictAll(datasets.TennisData.X) > 0))
```

[3]:
```
0.6428571428571429
```

1: why is this computation equivalent to computing classification accuracy?

Ans: - we have `(datasets.TennisData.Y > 0)` as y, aka ground truth. - we have `(h.predictAll(datasets.TennisData.X) > 0)` as y_hat, aka predictions.

using `==`, we get a boolean array where array[i] is true iff `y[i] == y_hat[i]`, basically if correct prediction.

`mean()` sums up the input then divides by its length. In python, `True` is `1` and `False` is `0`.

this gives us accuracy because `Accuracy = (True positives + True Negatives)/ (True positives + True negatives + False positives + False negatives)`

[4]:
```
runClassifier.trainTestSet(h, datasets.TennisData)
```

```
Training accuracy 0.642857, test accuracy 0.5
```

[ ]:
```
```

[5]:
```
h = dumbClassifiers.AlwaysPredictMostFrequent({})
runClassifier.trainTestSet(h, datasets.TennisData)
```

```
Training accuracy 0.642857, test accuracy 0.5
```

[6]:
```
runClassifier.trainTestSet(dumbClassifiers.AlwaysPredictOne({}), datasets.
  ↪SentimentData)
runClassifier.trainTestSet(dumbClassifiers.AlwaysPredictMostFrequent({}),␣
  ↪datasets.SentimentData)
```

```
Training accuracy 0.504167, test accuracy 0.5025
Training accuracy 0.504167, test accuracy 0.5025
```

[11]:
```
runClassifier.trainTestSet(dumbClassifiers.FirstFeatureClassifier({}), datasets.
  ↪TennisData)
runClassifier.trainTestSet(dumbClassifiers.FirstFeatureClassifier({}), datasets.
  ↪SentimentData)
```

```
[3, 2] [2, 7]
Training accuracy 0.714286, test accuracy 0.666667
[1, 3] [594, 602]
Training accuracy 0.504167, test accuracy 0.5025
```

[ ]:
```

```

[ ]:
```

```