thomasnilsson / ml02450

A series of functions that will speed up a lot of exercises found in exam sets, that normally are calculated by hand

**15** commits **1** branch **0** packages **0** releases **1** contributor MIT

Branch: master New pull request Find file Clone or download

Thomas Nilsson Readme Latest commit `f2d7972` on Dec 29, 2018

| | | |
|---|---|---|
| .idea | Readme | 11 months ago |
| .gitattributes | Initial commit | 11 months ago |
| .gitignore | Gitignore | 11 months ago |
| LICENSE | Initial commit | 11 months ago |
| README.md | Readme | 11 months ago |
| bin_classifier_ensemble.py | Added toolbox | 11 months ago |
| categoric2numeric.py | Added toolbox | 11 months ago |
| non_latex_README.md | Readme | 11 months ago |
| setup.py | Added toolbox | 11 months ago |
| similarity.py | Added toolbox | 11 months ago |
| tmgsimple.py | Added toolbox | 11 months ago |
| toolbox_02450.py | Added toolbox | 11 months ago |
| toolbox_extended.py | Added toolbox | 11 months ago |

README.md

# ml02450

This is a library which extends the toolbox provided in the 02450-course at The Technical University of Denmark and contains a series of functions that will speed up a lot of exercises found in exam sets, that normally are calculated by hand.

Tip: *The functions can also be used as a result-checking mechanism if you still prefer to do the questions by hand.*

## Install

Install this package with pip:

```
!pip install --ignore-installed ml02450
```

Disclaimer: *The author of most of this package is Tue Herlau*

## Usage

Import library in python with:

```
import toolbox_extended as te
import toolbox_02450 as tb
import numpy as np
```

Most library functions require numpy vectors as inputs, so numpy needs to be imported as well.

## Usage of the main toolbox

Whenever you would normally use toolbox functions such as

```
density, log_density = gausKernelDensity(X,w)
```

Instead, simply use the `tb.` as the module prefix:

```
density, log_density = tb.gausKernelDensity(X,w)
```

This accomplishes two things:

- No namespace pollution
- The toolbox being much easier to use in Jupyter Notebooks

## Usage of the extended toolbox

### AdaBoost

Given a classification problem with 25 observations in total, with 5 of them being misclassified in round 1, the weights can be calculated as:

```
miss = np.zeros(25)
miss[:5] = 1
te.adaboost(miss, rounds=1)
```

The weights are as follows:

```
w[0]: 0.100000
w[1]: 0.100000
w[2]: 0.100000
w[3]: 0.100000
w[4]: 0.100000
w[5]: 0.025000
w[6]: 0.025000
w[7]: 0.025000
w[8]: 0.025000
w[9]: 0.025000
w[10]: 0.025000
w[11]: 0.025000
w[12]: 0.025000
w[13]: 0.025000
w[14]: 0.025000
w[15]: 0.025000
w[16]: 0.025000
w[17]: 0.025000
w[18]: 0.025000
w[19]: 0.025000
w[20]: 0.025000
w[21]: 0.025000
w[22]: 0.025000
w[23]: 0.025000
w[24]: 0.025000
```

### Naive Bayes

Example from exam, fall 2018.

Given the following transaction matrix, where the rows correspond to a feature $f_i$, and the classes being given by the vector $y$, what is $P(y=1 \mid f_1 = 1, f_2 = 1. f_6 = 0)$?

```
X = np.array([
    [1,1,0,0,0,1,0,0,0,1],
    [1,0,0,0,0,0,0,0,0,0],
    [1,1,0,0,0,1,0,0,0,1],
    [0,1,1,1,0,0,0,1,1,0],
    [1,1,0,0,0,1,0,0,0,1],
    [0,1,1,1,0,0,1,1,1,0],
    [1,1,1,0,0,1,1,1,1,0],
    [0,1,1,1,0,1,1,0,0,1],
    [0,0,0,0,1,1,1,0,1,1],
    [1,0,0,0,0,1,1,1,1,0],
])

labels = ['f%i' %i for i in range(1,11)]
f1,f2,f3,f4,f5,f6,f7,f8,f9,f10 = X.T

y = [1,1,1,2,2,2,2,2,3,3]
x = [1,1,0]

te.naive_bayes(y, x, f1, f2, f6)
```

The probabilities $P(y=c \mid f_1 = 1, f_2 = 1. f_6 = 0)$ for $c=1,2,3$ is given by the following vector:

```
array([0.45454545, 0.54545455, 0.        ])
```

The answer is therefore $y=0.45$.

## Support

Given the transaction matrix, the support of the item-set $\{f_1, f_3, f_8, f_9, f_2, f_6, f_7\}$ can be calculated using the `supp(I)` function:

```
te.supp([f1,f3,f8,f9,f2,f6,f7])
```

Output:

```
0.1
```

## Confidence

Given the transaction matrix, what is the confidence of the rule $\{f_1, f_3, f_8, f_9\} \rightarrow \{f_2, f_6, f_7\}$

```
a = [f1,f3,f8,f9]
b = [f2,f6,f7]
te.conf(a,b)
```

Output:

```
1.0
```

## Confusion Matrix

Given a 2x2 confusion matrix with:

- TP: 18
- FN: 12
- TN: 15
- FP: 9

Calculating the accuracy, error rate, recall, and much more, is done as follows:

```
M = [[18,12],[9,15]]
te.confusion_matrix(M)
```

Alternatively, by providing the attributes directly:

```
te.confusion_matrix(tp=18, fn=12, tn=15, fp=9)
```

Output:

```
TP: 18 FN: 12 TN: 15 FP: 9
Accuracy: 0.6111111111111112
Error rate: 0.38888888888888884
Recall: 0.6
Precision: 0.6666666666666666
FPR: 0.375
TPR: 0.6
```

## ARD

Using K=2 nearest neighbours given the following distances, the avergae relative density becomes:

```
o1 = np.array([68.1, 165.4])
o3 = np.array([68.1, 111.1])
o4 = np.array([44.74, 32.5])

te.density(o1) / np.mean([te.density(o3), te.density(o4)])
```

Output:

```
0.46231460448232553
```