Perceptron update rule with numpy

This notebook implements the perceptron update rule with all misclassified training instances used in one shot rather than one at a time.

In [1]:

```
import numpy as np

# This import is needed so that we can display full output in Jupyter, not onl
y the last result.
# Importing modules is explained later in this tutorial.
# For the moment just execute this cell.

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [2]:

```
X = np.array([
[3,0.2,1],
[1,0.3,1],
[4,0.5,1],
[2,0.7,1],
[0,1.0,1],
[1,1.2,1],
[1,1.7,1],
[6,0.2,1],
[7,0.3,1],
[6,0.7,1],
[3,1.1,1],
[2,1.5,1],
[4,1.7,1],
[2,1.9,1]
])

y = np.array([[-1,-1,-1,-1,-1,-1,-1,1,1,1,1,1,1,1,]])
y=y.T

X,y
```

Out[2]:

```
(array([[ 3. ,  0.2,  1. ],
        [ 1. ,  0.3,  1. ],
        [ 4. ,  0.5,  1. ],
        [ 2. ,  0.7,  1. ],
        [ 0. ,  1. ,  1. ],
        [ 1. ,  1.2,  1. ],
        [ 1. ,  1.7,  1. ],
        [ 6. ,  0.2,  1. ],
        [ 7. ,  0.3,  1. ],
        [ 6. ,  0.7,  1. ],
        [ 3. ,  1.1,  1. ],
        [ 2. ,  1.5,  1. ],
        [ 4. ,  1.7,  1. ],
        [ 2. ,  1.9,  1. ]]), array([[-1],
        [-1],
        [-1],
        [-1],
        [-1],
        [-1],
        [-1],
        [ 1],
        [ 1],
        [ 1],
        [ 1],
        [ 1],
        [ 1],
        [ 1]]))
```

In [3]:

```
w = np.array([[0.3,0.8,-2.2]])
```

In [4]:

```
X@w.T
y*X@w.T
y*X@w.T<0
(y*X@w.T<0).reshape(X.shape[0])
XX = X[(y*X@w.T<0).reshape(X.shape[0]), :]
yy = y[(y*X@w.T<0).reshape(X.shape[0]), :]
XX
yy
Z = np.sum(yy*XX, axis=0, keepdims=True)
Z
```

```
Out[4]:  array([[-1.14],
                [-1.66],
                [-0.6 ],
                [-1.04],
                [-1.4 ],
                [-0.94],
                [-0.54],
                [-0.24],
                [ 0.14],
                [ 0.16],
                [-0.42],
                [-0.4 ],
                [ 0.36],
                [-0.08]])
```

```
Out[4]:  array([[ 1.14],
                [ 1.66],
                [ 0.6 ],
                [ 1.04],
                [ 1.4 ],
                [ 0.94],
                [ 0.54],
                [-0.24],
                [ 0.14],
                [ 0.16],
                [-0.42],
                [-0.4 ],
                [ 0.36],
                [-0.08]])
```

```
Out[4]:  array([[False],
                [False],
                [False],
                [False],
                [False],
                [False],
                [False],
                [ True],
                [False],
                [False],
                [ True],
                [ True],
                [False],
                [ True]], dtype=bool)
```

```
Out[4]:  array([False, False, False, False, False, False, False,  True, False,
                False,  True,  True, False,  True], dtype=bool)
```

```
Out[4]:  array([[ 6. ,  0.2,  1. ],
                [ 3. ,  1.1,  1. ],
                [ 2. ,  1.5,  1. ],
                [ 2. ,  1.9,  1. ]])
```

```
Out[4]:  array([[1],
                [1],
                [1],
                [1]])
```

```
Out[4]:  array([[ 13. ,   4.7,   4. ]])
```

In [5]:

```
eta =0.01
w = w + eta*Z
w

# Now we need to put this in a loop and execute several times until there is n
ot classification error.
```

Out[5]:

```
array([[ 0.43 ,  0.847, -2.16 ]])
```