Program 1 - Perceptron

Logan Grosz

Last updated: September 24, 2020

1 Description

Contained in this package are two items:

- Pilot program, `prog1.py`
- `myml` module

I have opted to move `ML.py` to a full python module since we will be adding to it over the semester and I could already start to see it getting bloated when adding the plotting functions.

1.1 myml.util

This part of the module simply contains utility functions `plot_decision_regions` and `plot_decision_regions_3d`. The former will show a scatter plot with the fitted line. The 3d variant will do the same but with a plane instead of a line.

1.2 myml.Perceptron

The perceptron follows the data fitting algorithm presented on Wikipedia closely. It initializes weights to 0. For each sample, it calculates the actual output,and then changes the weight to reflect the difference between this output and the desired output. This process repeats until the specified number of iterations is reached or there is an iteration error rate of 0. In order to preserve data, both the number of errors and weight at each step is recorded. These can be accessed through *Perceptron*`.weights` and *Perceptron*`.errors`.

Perceptron`.__init__` :: The initilizer takes 2 arguments: the first is the learning rate, a float between 0 and 1. The second is the max number of iterations. These values can be accessed again with *Perceptron*`.rate` and *Perceptron*`.niter`.

Perceptron`.fit` :: Takes arguments `X` (numpy array with shape (*nSamples*, *nFeatures*)) and `d` (also a numpy array with shape (*nSamples*)). This function applies the steps described above and "validates" the errors and weights arrays.

Perceptron`.net_input` :: Returns the weighted values of some given input `X` (of type numpy array with shape (*nSamples, nFeatures*)) as a numpy array with shape (*nSamples*).

Perceptron.`predict` :: Returns labels (-1 or 1) for a given `X` of type numpy array with shape (*nSamples, nFeatures*) as a numpy array of shape (*nSamples*).

`Perceptron.f` :: a static method which returns the label given `w` (a numpy array of shape (*nFeatures*+1)), and `x`, a sample with shape (*nFeatures*). Note, they're not the same size but that's because `w[0]` acts as a bias.

2 Importing the Module

Due to the heavy reliance on numpy arrays, obviously the numpy module should be installed. This is also a design decision as numpy arrays are much faster than python lists.

```
import Perceptron from myml

import myml.util as util
```

3 Usage

All the functions described above can be used how you'd expect. For example usage please see `prog1.py` included.

4 Testing

Several manual tests were used in the Iris dataset. These tests included all combinations of flowers with varying features. Some converged and some did not. I also did some 3 feature learning, to test that functionality. Although there much more manual tests, two can be found in the example `prog.py`. The first is a 2 feature test which takes almost 800 iterations to converge while the second is a 3 feature test which converges quickly. Both of these use the first two flower species as they were the easiest to get to converge.