Linear Least Squares Classifier

Requirements

- Python - tested with version 2.7.6
- Numpy - tested with version 1.8.2

Usage

```
./LLS.py data_file [--head]
```

`data_file` is a file containing data attributes and classes on each line, deliminated by a comma.

- Data must be formatted like those in the UCI Repository.
- Example data sets from UCI are included in the repo: iris.csv and wine.csv
- Classes can be words or numbers. **Attributes must be numbers at this time**

`--head` (optional) Explicitly state the location of the class label is at the head of each line. Without this option, default to the tail of the line. If you are getting terrible accuracy, you may have forgot to enable this flag.

You can also run all the data sets in this repo by executing `run_data.sh`

Description

This classifier works much like the libsvm classifier. Data must be seperated into training and testing data, were the class of the training data is explictly known.

The linear least squares function used during training is

$$E(\mathbf{w}) = \sum_i trace\left[(\mathbf{y}_i - \mathbf{W}^T x_i)^T(\mathbf{y}_i - \mathbf{W}^T\mathbf{x}_i)\right]$$

where $\mathbf{x}_i \in \mathbb{R}^D, \mathbf{y}_i \in \mathbb{R}^K, i \in 1, 2, ..N$.

We can minimize this function with respect to W to obtain

$$\hat{\mathbf{W}} = (\sum_i \mathbf{x}_i\mathbf{x}_i^T)^{-1}(\sum_i \mathbf{x}_i\mathbf{y}_i^T)$$

During testing, we find the class by solving $\hat{\mathbf{W}}\mathbf{x} = \mathbf{y}$

API Reference

Note: matrix referrs to a numpy matrix

`predict(W, x)` Predict the class y of a single set of attributes

- `matrix W` DxK Least squares weight matrix
- `matrix x` 1xD matrix of attributes for testing
- `return` List of 0's and 1's. Index with 1 is the class of x

`train(x, y)` Build the linear least weight matrix W using a training set of size N

- `matrix x` NxD matrix containing N attributes vectors for training
- `matrix y` NxK matrix containing N class vectors for training
- `Return` Weight matrix, as outlined in the description

`test(a, b, split)` Helper method that splits data into training and testing sets, trains the classifier using the training set, then predicts each of the testing data. Then it will compare the predicted result with the actual label and print the accuracy of the predicitons.

- `matrix a` All the attribute data
- `matrix b` All the classes that belong to each attribute
- `int split` Percent of data you want to train with