Homework 3 : Supervised Classification

Alexandre Boucher

January 25, 2010

Due date: January 28th 2010

We need two classes to model the nearest neighbor (nn) and the quadratic discriminant analysis (qda) classifiers. Each class implements the same interface

Classifier interface

The training phase is performed with the constructor (`__init__()`) and the classification with the member function `classify`.

```
__init__(self,dataTraining, classID, ...)
classify(self,data)
```

- `dataTraining` is a 2D scipy array of size L by M; L is the number of observations and M is the length of each observation.
- `classID` is a vector of size L, the entries can take the integer value $0, 1, ...K-1$ which defined the class labels of the observations in `dataTraining`.
- `classify(self,data)` returns a 1D vector of size N with the classification label $(0, 1, ...K-1)$ for each observation. You must check that the input variable is a scipy array:

  ```
  type(data) is scipy.ndarray
  ```

  and that each observation has the correct number of variables (M in this case)
- `data` is a 2D scipy array of size N by M; N is the number of observations and M is the length of each observation.

Classifiers

The library should two classifiers:

1. `NearestNeighborClassifier`

2. `QuadraticDiscriminantAnalysis`
   Recall that the qda discriminant function for the class label k is:

   $$g(z) = -\frac{1}{2}\underline{z}\Gamma_k^{-1}\underline{z}^T + \underline{z}\Gamma_k^{-1}\underline{\mu}_k^T - \frac{1}{2}\underline{\mu}_k\Gamma_k^{-1}\underline{\mu}_k^T + \log P(c_k) - \frac{1}{2}\log\left(|\Gamma_k|\right)$$

   where $\underline{z}$ and $\underline{\mu}_k$ are row vectors of size $1 \times M$ and Γ_k is the variance/covariance matrix of size $M \times M$. The determinant $|\Gamma_k|$ can be calculated with

   ```
   d = scipy.linalg.det(A)
   ```

   where `A` is a square array. The namespace `linalg` refers to the numpy linear algebra module. The matrix multiplication can be done with the `scipy.dot()` command. The scipy.matrix object is also an alternative for linear algebra.

Usage

The user of your function should be able to classify a data set by first calling the constructor `__init__` function followed by the `classify()` function. For example :

```
#Some code to get the training data ...
nnClassifier =  \
NearestNeighborClassifier(dataTraining, dataLabel)
qdaClassifier = \
QuadraticDiscriminantAnalysis(dataTraining, dataLabel)

#some code to get the to be classified data ...
nnLabels = nnClassifier.classify( data )
qdaLabels = qdaClassifier.classify( data )

#Do something with the classification
```

Tip:

Write a function to compute the accuracy of your classifier so you know if it works.

Bonus

Write a k-nn classifier.