Exercício 1 - MO444 - Aprendizado de máquina e reconhecimento de padrões

Renato Lopes Moura - 163050

Importando os módulos que serão utilizados no exercício

```
In [1]: import numpy as np
        from sklearn.decomposition import PCA
        import pandas as pd
        from sklearn.linear_model import LogisticRegression
        from sklearn.lda import LDA
        from sklearn import metrics
```

Leitura dos dados do arquivo csv (utilizando o pandas) e conversão para arrays do numpy (necessário para aplicar o PCA do scikit-learn)

```
In [3]: data = pd.read_csv('data1.csv')
        array = data.values
```

Separando os dados entre:

- Atributos: primeiras 166 colunas
- Classes: última coluna

```
In [4]: X = array[:,0:165]
        Y = array[:,166]
```

Inicialmente, é aplicado o PCA considerando todos os atributos de cada amostra e então é verificada a variância acumulada.

```
In [6]: pca = PCA()
        pca.fit(X)

        var=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)

        print var
```

```
[ 31.37  45.37  53.01  58.15  62.86  66.96  70.15  73.2   75.2   76.93
  78.48  79.9   81.23  82.49  83.59  84.58  85.45  86.27  87.03  87.75
  88.43  89.07  89.7   90.29  90.85  91.37  91.87  92.36  92.83  93.27
  93.67  94.07  94.44  94.77  95.06  95.34  95.61  95.87  96.1   96.3   96.5
  96.69  96.87  97.04  97.2   97.34  97.48  97.62  97.74  97.86  97.97
```

```
  98.08  98.18  98.28  98.37  98.45  98.53  98.61  98.68  98.75  98.81
  98.87  98.93  98.98  99.03  99.08  99.13  99.17  99.21  99.25  99.28
  99.31  99.34  99.37  99.4   99.43  99.46  99.49  99.51  99.53  99.55
  99.57  99.59  99.61  99.63  99.65  99.67  99.69  99.7   99.71  99.72
  99.73  99.74  99.75  99.76  99.77  99.78  99.79  99.8   99.81  99.82
  99.83  99.84  99.85  99.86  99.87  99.88  99.89  99.9   99.91  99.92
  99.93  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94
  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94
  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94
  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94
  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94  99.94
  99.94  99.94  99.94  99.94]
```

Pela variância acumulada acima, podemos ver que para manter pelo menos 80% da variância original precisamos considerar **13 componentes** na transformação do PCA. Aplicando o PCA novamente, agora com o número de componentes desejado e salvando os dados transformados.

```
In [7]: pca = PCA(n_components=13)
        X_transf = pca.fit_transform(X)
```

Agora os dados do conjunto original e os dados transformados pelo PCA serão utilizados para treinar dois modelos de regressão logística. Apenas os 200 primeiros dados de cada conjunto serão utilizados no treinamento.

```
In [8]: model = LogisticRegression()
        model = model.fit(X[:200,:],Y[:200])

        model_transf = LogisticRegression()
        model_transf = model_transf.fit(X_transf[:200,:],Y[:200])
```

Os modelos treinados agora serão testados no restante dos dados dos respectivos conjuntos.

```
In [9]: predicted = model.predict(X[200:])

        predicted_transf = model_transf.predict(X_transf[200:])
```

E então podemos verificar a acurácia de cada modelo na classificação:

```
In [10]: print "Acuracia da regressao logistica no conjunto de dados original: "+ \
             str(metrics.accuracy_score(Y[200:], predicted))
         print "Acuracia da regressao logistica no conjunto de dados transformado: "+ \
             str(metrics.accuracy_score(Y[200:], predicted_transf))

Acuracia da regressao logistica no conjunto de dados original: 0.797101449275
Acuracia da regressao logistica no conjunto de dados transformado: 0.800724637681
```

Repetindo o procedimento para o LDA e verificando a acurácia dos modelos:

```
In [11]: model_LDA = LDA()
         model_LDA = model_LDA.fit(X[:200],Y[:200])

         model_LDA_transf = LDA()
         model_LDA_transf = model_LDA_transf.fit(X_transf[:200],Y[:200])

         predicted_LDA = model_LDA.predict(X[200:])

         predicted_LDA_transf = model_LDA_transf.predict(X_transf[200:])

         print "Acuracia do LDA no conjunto de dados original: "+ \
             str(metrics.accuracy_score(Y[200:], predicted_LDA))
         print "Acuracia do LDA no conjunto de dados transformado: "+ \
             str(metrics.accuracy_score(Y[200:], predicted_LDA_transf))

Acuracia do LDA no conjunto de dados original: 0.677536231884
Acuracia do LDA no conjunto de dados transformado: 0.79347826087
```

Baseado nos resultados das acurácias, verificamos que o melhor modelo para a classificação dos dados é o que utiliza **Regressão Logística com PCA**.