LogicticRegression

January 9, 2020

1 Logistic Regression on Titanic data set

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        from sklearn.linear_model import LogisticRegression
        from sklearn import metrics
        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeClassifier
```

First we clean all the data.

```
In [2]: def normalisation(feature):
            return (feature - feature.mean()) / feature.std()
```

```
In [3]: titanic = pd.read_csv('./titanic.csv', index_col=0)

        titanic['Age'].fillna(titanic['Age'].mean(), inplace=True) # replace Null value with t
        titanic['Age'] = normalisation(titanic['Age'])

        # replace the sex with a binary value
        titanic['Sex'].replace(to_replace=['female','femme'],value=0, inplace=True)
        titanic['Sex'].replace(['male', 'homme'], 1 , inplace=True)

        # replace O with 0 to have a binary value
        titanic['Survived'].replace(['O'], 0, inplace=True)
        titanic['Survived'].fillna(0, inplace=True)
        titanic['Survived'] = titanic['Survived'].astype('int')

        titanic['isKid']=0
        titanic.loc[titanic.Age<16,'isKid']=1
        titanic['isAlone']=0
        titanic.loc[(titanic.SibSp==0)&(titanic.Parch==0),'isAlone']=1

        titanic = pd.concat([titanic, pd.get_dummies(titanic['Pclass'], prefix='Pclass'), pd.g
        titanic.drop(columns=['Pclass', 'Sex', 'Cabin', 'Embarked'], inplace=True)
```

```
titanic.head(20)
```

Out[3]:

PassengerId	Survived	Name	Age	SibSp	Parch	Ticket	Fare	isKid
1	0	Braund, Mr. Owen Harris	-5.921480e-01	1	0	A/5 21171	7.2500	1
2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	6.384304e-01	1	0	PC 17599	71.2833	1
3	1	Heikkinen, Miss. Laina	-2.845034e-01	0	0	STON/O2. 3101282	7.9250	1
4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	4.076970e-01	1	0	113803	53.1000	1
5	0	Allen, Mr. William Henry	4.076970e-01	0	0	373450	8.0500	1
6	0	Moran, Mr. James	4.371893e-15	0	0	330877	8.4583	1
7	0	McCarthy, Mr. Timothy J	1.869009e+00	0	0	17463	51.8625	1
8	0	Palsson, Master. Gosta Leonard	-2.130371e+00	3	1	349909	21.0750	1
9	1	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	-2.075923e-01	0	2	347742	11.1333	1
10	1	Nasser, Mrs. Nicholas (Adele Achem)	-1.207437e+00	1	0	237736	30.0708	1
11	1	Sandstrom, Miss. Marguerite Rut	-1.976549e+00	1	1	PP 9549	16.7000	1
12	1	Bonnell, Miss. Elizabeth	2.176654e+00	0	0	113783	26.5500	1
13	0	Saundercock, Mr. William Henry	-7.459703e-01	0	0	A/5. 2151	8.0500	1
14	0	Andersson, Mr. Anders Johan	7.153416e-01	1	5	347082	31.2750	1
15	0	Vestrom, Miss. Hulda Amanda Adolfina	-1.207437e+00	0	0	350406	7.8542	1
16	1	Hewlett, Mrs. (Mary D Kingcome)	1.945920e+00	0	0	248706	16.0000	1
17	0	Rice, Master. Eugene	-2.130371e+00	4	1	382652	29.1250	1
18	1	Williams, Mr. Charles Eugene	4.371893e-15	0	0	244373	13.0000	1
19	0	Vander Planke, Mrs. Julius (Emelia Maria Vande...	1.000524e-01	1	0	345763	18.0000	1
20	1	Masselmani, Mrs. Fatima	4.371893e-15	0	0	2649	7.2250	1

PassengerId	isAlone	Pclass_1	Pclass_2	Pclass_3	Sex_0	Sex_1	Embarked_C	Embarked_Q	Embarked_S
1	0	0	0	1	0	1	0	0	1
2	0	1	0	0	1	0	1	0	0
3	1	0	0	1	1	0	0	0	1
4	0	1	0	0	1	0	0	0	1
5	1	0	0	1	0	1	0	0	1
6	1	0	0	1	0	1	0	1	0
7	1	1	0	0	0	1	0	0	1
8	0	0	0	1	0	1	0	0	1
9	0	0	0	1	1	0	0	0	1
10	0	0	1	0	1	0	1	0	0
11	0	0	0	1	1	0	0	0	1
12	1	1	0	0	1	0	0	0	1
13	1	0	0	1	0	1	0	0	1
14	0	0	0	1	0	1	0	0	1
15	1	0	0	1	1	0	0	0	1
16	1	0	1	0	1	0	0	0	1
17	0	0	0	1	1	0	0	0	1
18	1	0	1	0	0	1	0	0	1
19	0	0	0	1	1	0	0	0	1
20	1	0	0	1	1	0	1	0	0

In [4]:
```
titanic.columns
```

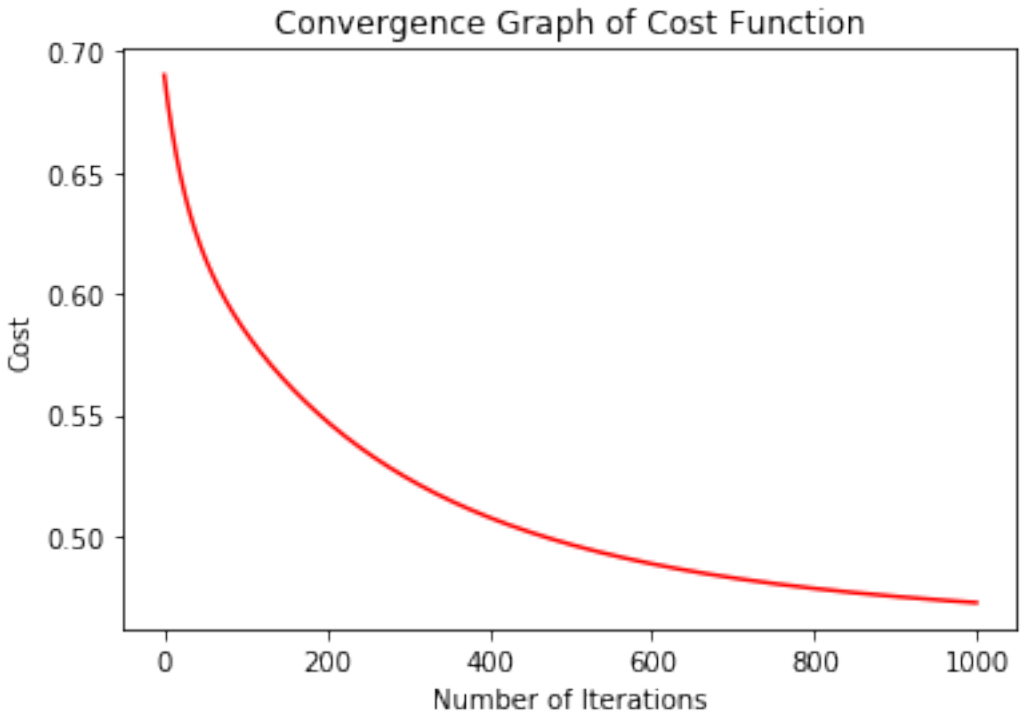
```
Out[4]: Index(['Survived', 'Name', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'isKid',
               'isAlone', 'Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_0', 'Sex_1',
               'Embarked_C', 'Embarked_Q', 'Embarked_S'],
              dtype='object')
```

1.1 From Scratch

```
In [5]: def sigmoid(x, theta):
            z = np.dot(x, theta)
            return 1 / (1 + np.exp(-z))

In [6]: def cost_function(X, y, theta):
            h = sigmoid(X, theta)
            loss = (-y * np.log(h) - (1-y)* np.log(1-h)).mean()
            return loss

In [7]: def gradient_descent(X, y, params, learning_rate=0.01, iterations=1000):
            m = len(y)
            cost_history = np.zeros((iterations, 1))

            for i in range(iterations):
                pred = sigmoid(X,params)
                loss = pred - y

                grad = np.dot(X.T, loss)
                params = params - learning_rate * grad * 1/m

                params = params - (learning_rate/m) * grad
                cost_history[i] = cost_function(X,y,params)

            return (cost_history, params)

In [8]: def predict(X, y, params):
            costs, w = gradient_descent(X,y, params)
            y_pred=sigmoid(X,w)
            classify=[1 if i > 0.5 else 0 for i in y_pred]
            return classify

In [31]: X = titanic[['Age', 'Sex_0','Sex_1', 'Pclass_1', 'Pclass_2', 'Pclass_3', 'Embarked_C'
         y = np.array(titanic[['Survived']])

         X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)

         m = len(y_train)
         n = len(y_test)

         X_train = np.hstack((np.ones((m,1)),X_train))
         X_test = np.hstack((np.ones((n,1)),X_test))
         params = np.zeros((X_train.shape[1],1))
```

```
initial_cost = cost_function(X_train, y_train, params)

(cost_history, params_optimal) = gradient_descent(X_train, y_train, params)
print(round(metrics.accuracy_score(y_train, predict(X_train,y_train, params_optimal))*
print(round(metrics.accuracy_score(y_test, predict(X_test,y_test, params_optimal))* 10

final_cost = cost_history[-1][0]

print("Initial Cost is:", initial_cost)
print("Final Cost is:", final_cost)

plt.figure()
plt.plot(range(len(cost_history)), cost_history, 'r')
plt.title("Convergence Graph of Cost Function")
plt.xlabel("Number of Iterations")
plt.ylabel("Cost")
plt.show()
```

```
78.33 % regression train.
81.34 % regression test.

Initial Cost is: 0.6931471805599454
Final Cost is: 0.47246519783840224
```



1.2 Classification librairies

1.2.1 Scikit Learn

```
In [73]: logisticRegScikit = LogisticRegression()
         X = titanic[['Age', 'Sex_0','Sex_1', 'Pclass_1', 'Pclass_2', 'Pclass_3', 'Embarked_C'
         Y = titanic[['Survived']]

         X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.3)

         X_train
         logisticRegScikit.fit(X_train,Y_train)
         logisticRegScikit.predict(X_train)


         print(round(logisticRegScikit.score(X_train, Y_train) * 100, 2),'% regression train.\
         print(round(logisticRegScikit.score(X_test,Y_test) * 100, 2),'% regression test.\n')
```

```
79.78 % regression train.

76.87 % regression test.
```

```
/Users/romane/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:433: Futu
  FutureWarning)
/Users/romane/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:761: DataConve
  y = column_or_1d(y, warn=True)
```

1.2.2 Decision tree

```
In [74]: tree = DecisionTreeClassifier()
         tree.fit(X_train, Y_train)
         prediction_survived_tree = tree.predict(X_train)
         print(round(tree.score(X_train, Y_train) * 100, 2),'% tree train.\n')
         print(round(tree.score(X_test,Y_test) * 100, 2),'% tree test.\n')
```

```
93.42 % tree train.

72.76 % tree test.
```

```
In [ ]:
```