maximum likelihood estimation and logistik regression

July 13, 2022

[27]:
```
import math
#n = 100, p = 0.5
def p(n,x,p):
    result = math.comb(n,x)*p**(x)*(1-p)**(n-x)
    return result
```

[28]:
```
p(100,56,0.5)
```

[28]:
```
0.03895255978909614
```

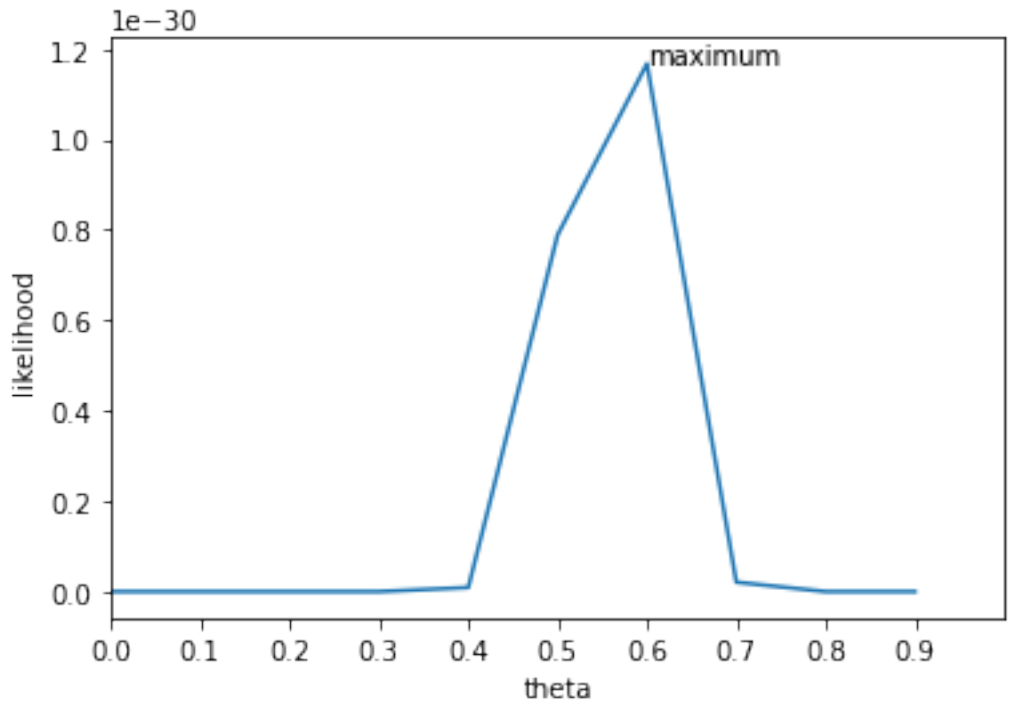
[39]:
```
def likelihood(n,x,p):
    return (p**x*((1-p)**(n-x)))
```

[40]:
```
import numpy as np

mylist = np.arange(0, 1, 0.1)
res = []
for i in mylist:
    res.append(likelihood(100,56,i))
print(res)
```

```
[0.0, 9.697737297875277e-59, 3.923188584616697e-44, 8.000258592525047e-37,
8.995276787107377e-33, 7.888609052210118e-31, 1.167104221439726e-30,
2.0836532533509102e-32, 6.58201822928478e-37, 2.7389274499533857e-47]
```

[43]:
```
import matplotlib.pyplot as plt
plt.plot(mylist, res)
plt.xticks(mylist)
plt.xlabel('theta')
plt.ylabel('likelihood')
plt.xlim(0,1)
plt.annotate('maximum',(0.6,res[6]))
plt.show()
```



[51]:
```
import pandas as pd
from sklearn import datasets

data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns = data.feature_names)
df = df[['mean radius', 'mean texture', 'mean area', 'mean symmetry']]

df['target'] = data.target
df.head()
```

[51]:

	mean radius	mean texture	mean area	mean symmetry	target
0	17.99	10.38	1001.0	0.2419	0
1	20.57	17.77	1326.0	0.1812	0
2	19.69	21.25	1203.0	0.2069	0
3	11.42	20.38	386.1	0.2597	0
4	20.29	14.34	1297.0	0.1809	0

[52]:
```
from sklearn.model_selection import train_test_split

# train, test
X = df[['mean radius', 'mean texture', 'mean area', 'mean symmetry']]
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

[53]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(penalty ='l2')
model.fit(X_train,y_train)
```

[53]:
```
LogisticRegression()
```

[54]:
```
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
accuracy_score(y_pred, y_test)
```

[54]:
```
0.9122807017543859
```

[64]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

[65]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'C' : [0.001, 0.01, 0.1, 1, 10, 100],
    'max_iter' : [100,500,1000]
}

grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5)

grid_search.fit(X_train_scaled, y_train)
print('test accuracy : ', grid_search.score(X_test_scaled, y_test))
print('  parameters : ', grid_search.best_params_)
```

```
test accuracy :  0.9210526315789473
  parameters :  {'C': 10, 'max_iter': 100}
```

[66]:
```
model = LogisticRegression(C = 10, max_iter = 100 ,penalty ='l2')
model.fit(X_train_scaled,y_train)
y_pred = model.predict(X_test_scaled)
accuracy_score(y_pred, y_test)
```

[66]:
```
0.9210526315789473
```

[ ]:
```

```