Bayesian Model for Machine Learning

Loïc Tetrel

Keywords: Generative model; Bayes; Prediction

1 Introduction

Thomas Bayes is a well known mathematician, with work focused in probability. Bayes theory[1] is central in machine learning and has numerous applications in automatic classification.

2 Theory

Consider two event a and b, the Bayesian rule define the probability P that a occurs, knowing the result of the event b :

$$P(a|b) = \frac{P(b|a) \times P(a)}{P(b)} \tag{1}$$

It can be interpreted as is : if we know a priori the probabilities of the events a, b, and the result of the event b (knowing result of a), then we can deduct the result a (knowing result of b).
Using this rule will allow us to infer things about the income of events, when you have some aprioris. This is the base in Machine Learning, with training data, you infer the result for testing data.

3 Code example

Let's do a practical example of what Bayes is. Imagine you are in charge of a production line, this line sort strawberries from grapes. To do this, you bought cameras and measure the radius of the fruits coming from the farmers. Your hypothesis is that the radius is a good predictor, and you want to use a bayesian rule to classify the fruits !

$$P(\omega_i|X) = \frac{P(X|\omega_i) \times P(\omega_i)}{P(X)} \tag{2}$$

Where $\omega_i = \omega_g, \omega_c$ are the events for ω_g : "The fruit is a grape", and ω_c "The fruit is a cherry". The measured radius is called X, $P(\omega_i)$ and $P(X)$ are called marginal probabilities, and $P(X|\omega_i)$ is known as the likelihood. One can define the likelihhod as : *The probabilities of the incoming data, as if they belong to the classes.*

First, you want to model the distribution of your fruits. In Machine learning, we usually refer the probability as a function called pdf (probability density function). A gaussian prior is often used. So you start acquiring many data while knowing what fruit it is, and calculate the parameters (μ, σ) of the associated gaussian [1].

```
# Creation of data
RadiusCherries = np.random.normal(4, 0.5, 1000)
RadiusGrapes = np.random.normal(2, 0.5, 1000)

# Estimators for gaussian mean and deviation
MeanCherries = np.median(RadiusCherries)
DevCherries = np.std(RadiusCherries)
MeanGrapes = np.median(RadiusGrapes)
DevGrapes = np.std(RadiusGrapes)
```

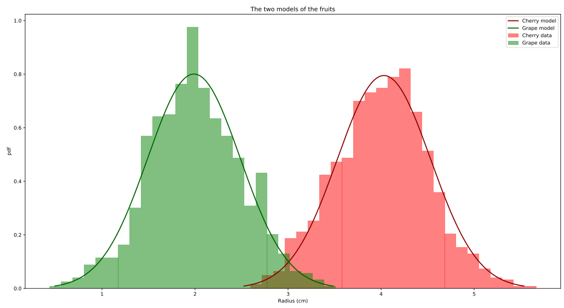

Fig. 1: The two models for the fruits.

Now that we have our models, we can evaluate the likelihood of the data for our classes.

```
#New radius data coming in
Data = np.array([1.52, 2.68, 3, 4, 6])

#Bayes
# P(w_c) = P(w_g)
PAprioriCherry = PAprioriGrape = 0.5
PData = 1
LikelihoodsCherry = stats.norm(MeanCherries, DevCherries).pdf(Data)
LikelihoodsGrape = stats.norm(MeanGrapes, DevGrapes).pdf(Data)
```

[1] Usually you want to calculate the likelihood of your data to verify if it's truly gaussian, or use a test like Student's t-test.

For example, the likelihood of $x = 3$ gives the figure 2.

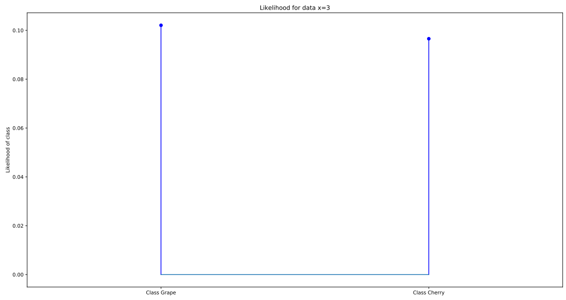

Fig. 2: Likelihood for one data.

Now, we can evaluate the posterior probability (predict what type of fruit it is), given the marginals and the likelihood [2].

```
PCherry = LikelihoodsCherry * PAprioriCherry/PData
PGrape = LikelihoodsGrape * PAprioriGrape/PData
```

After we calculated the posterior for every data, the MAP (maximum aposteriori prediction) can be used to find to which class our measure comes from 3. So if $P(\omega_g|X) > P(\omega_c|X)$ that means that the data X is more likely to be a grape.

```
for i in range(0, len(Data)):
if PGrape[i] > PCherry[i]:
plt.plot(Data[i], 0.02, 'o', color='darkgreen', linewidth=10, markers
else:
plt.plot(Data[i], 0.02, 'o', color='darkred', linewidth=10, markersiz
```

Be careful if the two probabilities are really close (in our example the 3rd fruit). In this case, you could use an uncertainty factor and decide to not predict it (it's ok, just throw this fruit !).

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

# Creation of data
RadiusCherries = np.random.normal(4, 0.5, 1000)
RadiusGrapes = np.random.normal(2, 0.5, 1000)
```

[2] Sometimes, it can be simpler and faster to evaluate just the likelihoods to do predictions.

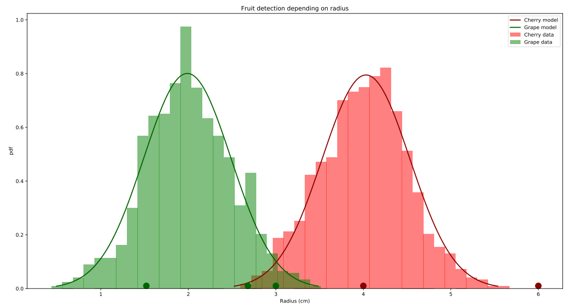

Fig. 3: The prediction of 5 fruits.

```
# Estimators for gaussian mean and deviation
MeanCherries = np.median(RadiusCherries)
DevCherries = np.std(RadiusCherries)
MeanGrapes = np.median(RadiusGrapes)
DevGrapes = np.std(RadiusGrapes)

#pdf of data
xCherries = np.linspace(MeanCherries − 3∗DevCherries, MeanCherries + 3∗DevCh
fCherries = stats.norm(MeanCherries, DevCherries).pdf(xCherries)
xGrapes = np.linspace(MeanGrapes − 3∗DevGrapes, MeanGrapes + 3∗DevGrapes, 100
fGrapes = stats.norm(MeanGrapes, DevGrapes).pdf(xGrapes)

# Probability density functions
plt.figure()
plt.hist(RadiusCherries, normed = 1, bins = 25, color = 'red', alpha = 0.5, l
plt.hist(RadiusGrapes, normed = 1, bins = 25, color = 'green', alpha = 0.5, l
plt.plot(xCherries, fCherries, color = 'darkred', linewidth = 2, label = 'Che
plt.plot(xGrapes, fGrapes, color = 'darkgreen', linewidth = 2, label = 'Grape


#New radius data coming in
Data = np.array([1.52, 2.68, 3, 5, 6])

#Bayes
# P(w_c) = P(w_g)
PAprioriCherry = PAprioriGrape = 0.5
PData = 1
PCherry = stats.norm(MeanCherries, DevCherries).pdf(Data) ∗ PAprioriCherry/PD
PGrape = stats.norm(MeanGrapes, DevGrapes).pdf(Data) ∗ PAprioriGrape/PData
```

```
for i in range(0, len(Data)):
    if PGrape[i] > PCherry[i]:
        plt.plot(Data[i], 0.02, 'o', color='darkgreen', linewidth=10, markers
    else:
         plt.plot(Data[i], 0.02, 'o', color='darkred', linewidth=10, markersi
plt.legend()
plt.title('Fruit␣detection␣depending␣on␣radius')
plt.xlabel('Radius␣(cm)')
plt.ylabel('Probability')
```

4 Study case

5 Conclusion

Example of Monty Hall problem, resolved using bayesian rule : `https://www.quora.com/How-do-I-solve-the-Monty-Hall-Problem-using-Bayes-Theorem`

Acknowledgment

Thanks

References

1. Bayes, T.: Essay towards solving a problem in the doctrine of chances. University Press (1958)