Bayes Decision Theory and Its Application on Edge Detection

Ye Li

These notes briefly describe how to make decisions with uncertain observations by using the Bayes Decision Theory in the context of an edge detection example.

1. Bayes Decision Theory

To find the Bayes rule (by using the Bayes Decision Theory), we need to have the likelihood distribution $P(x|y)$ and the prior distribution $P(y)$ to calculate the posterior distribution. The **likelihood** distribution specifies how likely the observed is under the assumed class and the **prior** allows you to specify your domain knowledge, i.e., which class is more likely. The observer data, x, can be a specific pixel value present in the image like 3.2 or 1.9 or it can be an image feature vector.

Mathematically, we can write the posterior probability as follows:

$$P(x|y)P(y) \approx P(y|x) = \frac{P(x,y)}{P(x)}$$

The next thing is to specify a **loss** function. A loss function is defined as

$$L(\alpha(x), y) = \begin{cases} K\_1 \; if \; \alpha(x) = y \\ K\_2 \; if \; \alpha(x) \neq y \end{cases}$$

where $\alpha(x)$ is the label/class estimated by the model and y is the true label of the data. For example, the loss function is defined as follows:

$$L(\alpha(x), y) = \begin{cases} 0 \; if \; \alpha(x) = y \\ 1 \; if \; \alpha(x) \neq y \end{cases}$$

We can then compute the risk for this $\alpha(x)$:

$$\begin{aligned} R(\alpha) &= \sum\nolimits_{x,y} L(\alpha(x), y)P(x,y) \\ &= \textstyle\sum_x (L(\alpha(x), 0)P(x,0) + L(\alpha(x), 1)P(x,1)) \end{aligned}$$

Now assume that z is one of the x's and we want to minimize the risk for one of the terms in the above sum. And by taking the log of that we have:

$$L(\alpha(z), 0)P(z,0) + L(\alpha(z), 1)P(z,1)$$

When $\alpha(x) = 0$, the risk for that is as follows:

$$logP(z,1) = \log P(z|y=1) + logP(y=1)$$

When $\alpha(x) = 1$, the risk for that is as follows:

$$logP(z,0) = \log P(z|y=0) + logP(y=0)$$

Now if we want the classifier, $\alpha(z)$, to choose 1, we would like to have the following risk relationship:

$$logP(z,y=1) > logP(z,y=0)$$

which is

$$log\frac{P(z|y=1)}{P(z|y=0)} > log\frac{P(y=0)}{P(y=1)} = \mathrm{T}$$

This is an analytical way to derive Bayes rule's threshold. Notice that our loss function is simply 0 and 1 we didn't include any T_f, T_n, F_n, F_p. But if we do the analytical threshold should be as follows:

$$log\frac{P(x|y=1)}{P(x|y=0)} > log\frac{P(y=0)}{P(y=1)} + \log\frac{T_n - F\_p}{T_p - F\_n}$$

Of course it's the best to specify the threshold by defining the prior and loss function at the beginning but if we don't have a specific loss/prior in mind we can use the following estimators to find a threshold using the development dataset.

Maximum Likelihood estimation and **Maximum a posteriori** estimation are methods that can be used for model parameter estimation. For a binary decision task the parameter to be estimated can be the threshold for classifying the observed data into a certain class. By using these methods we can find the optimal threshold for $\alpha(x)$ using the development set (a set of data that's separate from training and testing).

There are three approaches to select a model: 1) Baye's Decision Theory, that is we can penalize the errors by different amounts by taking the rareness of the error into account by minimizing the expectation of the loss function above. 2) MAP, which is we can penalize all errors by the same amount by simply maximizing the posterior probability above and get a MAP estimator $\alpha(x) = argmaxP(y|x)$. 3) ML, which is that we are assuming the prior distribution is uniform (no prior). For the first approach, the total loss function or the risk is:

$$R(\alpha) = \sum_{x,y} L(\alpha(x), y)P(x,y)$$

To select the best decision rule $\alpha(x)$ among all αs with different thresholds, we can minimize the above total loss function and then we will get the **Bayes Risk** and **Bayes Decision Rule** as follows:

$$\mathrm{R}(\hat{\alpha}) = \arg min_{\alpha}\, R(\alpha)$$
$$\hat{\alpha} = \arg min_{\alpha}\, R(\alpha)$$

For a binary decision task, the Bayes rule can be expressed as the log-likelihood ratio vs. a threshold value. A generic form of the Bayes rule can be written as follows:

$$log\,\frac{P(x|y=1)}{P(x|y=-1)} > T$$

The threshold is dependent on the **loss function** and **prior**. The loss function is defined at the beginning by the user by assigning user preferred cost to different types of mistakes (i.e., $L(\alpha(x), y)$, FP=3, FN=30) and the prior is specified by the domain knowledge. Once the prior and loss function is defined, conditional distributions ($P(x|y=1)\ and\ P(x|y=-1)$) are learned from the training data, we can then use the threshold ($log\,\frac{P(y=0)}{P(y=1)} + \log\,\frac{T_n - F\_p}{T_p - F\_n}$) and these conditional distributions to calculate the likelihood ratio of the testing data and classify the testing data. The distribution of $P(x|y=1)$ for the testing data is obtained from fitting the learned conditional distributions to a parametric model or simply using the original histogram generated from the training data.

2. False positives, false negatives, and their relation to the threshold

What are false positives and false negatives? What are the formula for these when the likelihood functions/distributions are Gaussians of one variable with the same variance σ^2, but different means μT, μD. Express the false positives and false negatives in terms of the error function (integrals of Gaussians). How do they vary with the threshold? Give a formula for how the threshold depends on the prior and the loss function?

False positive is when the estimated class is positive but the true class is actually negative
False negative is when the estimated class is negative but the true class if actually positive

$$FP = \int_{T}^{ub\ of\ uT} P(x|y=-1)dx$$
$$FN = \int_{lb\ of\ uB}^{T} P(x|y=1)dx$$

where $P(x|y=-1) \sim N(u_T, \sigma)$ and $P(x|y=1) \sim N(u_D, \sigma)$

To find the upper and lower bounds, we can use the CDF of the Gaussian distribution. The upper bound of the Gaussian that has mean uT is equal to:

$$1 = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x-uT}{\sigma\sqrt{2}}\right)\right]$$

$$x = u_T + \sigma\sqrt{2}erf^{-1}(1) = \infty$$

and the lower bound of the Gaussian that has mean uD is equal to:

$$0 = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-uD}{\sigma\sqrt{2}}\right)\right]$$
$$x = u_T + \sigma\sqrt{2}erf^{-1}(-1) = -\infty$$

So the integrals above become:

$$FP = \int_T^{\infty} P(x|y=-1)dx$$
$$= 1 - \int_{-\infty}^{T} P(x|y=-1)dx$$
$$= 1 - \frac{1}{2}\left[1 + \text{erf}\left(\frac{T-uT}{\sigma\sqrt{2}}\right)\right]$$

$$FN = \int_{-\infty}^{T} P(x|y=1)dx$$
$$= \frac{1}{2}\left[1 + \text{erf}\left(\frac{T-uD}{\sigma\sqrt{2}}\right)\right]$$

From the relations above, we can see that as T increases, FP rate decreases and FN increases.
For Bayes Decision Theory, the threshold is dependent on the loss function and prior as follows:

$$log\frac{P(x|y=1)}{P(x|y=0)} > log\frac{P(y=0)}{P(y=1)} + \log\frac{T_n - F\_p}{T_p - F\_n}$$

3. Bayes Decision Theory for Edge Detection

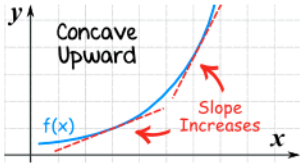
How can Bayes Decision Theory be applied to edge detection? Why is a first order derivative filter good for edge detection? And why is the second order 1 derivative filter less good? Given the hierarchical nature of visual processing, and the difficulty of edge detection, what is a good loss function for edge detection? What should be the trade-off between false positives and false negatives?

For edge detection, there are two classes to be set for Bayes Decision Theory: y=0 for no edge and y=1 for edge. Once the classes are chosen, we can then define the prior and loss function to make the threshold as described at the end of question 2. Then we will need to learn the conditional probability distributions from the training data and then from that we can calculate the likelihood of the testing data, from which we can obtain the class/inference result by comparing it with the threshold.

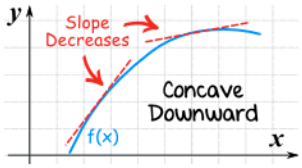
There are few advantages for 1st order derivatives: 1) fast to compute, 2) good at detecting strong edges. First order derivatives are less sensitive to noise than second order derivative

in detecting edges. Also, 2nd derivatives require strong changes to be useful. In other words, not so significant edges can't be detected by the 2nd derivative of the image. A good illustration of the 2nd derivatives can be seen in the following picture. It's not hard to see that the rate of the increase or decrease on slope is not significant between two neighboring points (i.e., pixel values) on the curve so the 2nd derivative in this case won't serve as a good maker for edges as the 1st derivatives.

Concave upward is when the slope increases:



Concave downward is when the slope decreases:



A good loss function for edge detection should have a relatively small penalty for false positive so that all possible edges can be captured by the classifier and if necessary these edges can be further processed for more accurate result. So we should have a smaller value for the cost of false positives than that of the false negatives.