Q1. Suppose that in answering a question in a multiple choice test, an examinee either knows the answer, with probability p, or he/she guesses with probability 1 − p. Assume that the probability of answering a question correctly is 1 for an examinee who knows the answer and 1/m for the examinee who guesses, where m is the number of multiple choice alternatives. What is the probability that an examinee actually knew the answer to a question, given that he has correctly answered it?

Answer:

P(knew) = p

P(guess) = 1 – p

P(correct | knew) = 1

P(correct | guess) = 1/m

From Bayes’Rule,

P(knew | correct) = /

And,

P(correct) = P(correct | knew) \* P(knew) + P(correct | guess) \* P(guess)

So,

P(knew | correct) =

Q2. A prize is hidden behind one of three doors A, B, and C. The contestant picks a door, say A, but it is left closed. The host (who knows the actual location of the prize, but will never reveal the prize at this stage of the game) opens door C and shows that there is no prize behind it. Should the contestant change his mind and select door B? Formulate this problem as Bayes inference. What is the prior probability of the prize being behind each door (i.e., A, B, or C) before door C was opened? What is the posterior probability after door C has been opened? What should be the contestants decision?

Answer:

Use C# replace The host opens dorr C.

A, B, C means the prize being behind each door(A, B, C)

The prior probability of the prize being behind each door (i.e., A, B, or C) before door C was opened:

P(A) = P(B) = P(C) = 1/3.

Posterior probability after door C has been opened:

P(A | C#) =

If the A have prize, the host will open door B or C equally. So P(C# | A) is 1/2.

So, P(A | C#) =

P(B | C#) = =

If the B have prize, the host will definitly open door B. So P(C# | B) is 1.

So, P(B | C#) =

And, P(C#) = P(C# | A) \* P(A) + P(C# | B) \* P(B) + P(C# | C) \* P(C)

= 1/2 \* 1/3 + 1 \* 1/3 + P(C# | C) \* 1/3

Because the host will never reveal the prize at this stage of the game, So P(C# | C) = 0.

So, P(C#) = 1/2.

P(A | C#) = 1/3

P(B | C#) = 2/3

So, the contestants should pick door B, after the host opens the door C.

Q3. Your spaceship has landed on an unknown planet. On this planet, it is possible with prior probability P1, that the sun rises every day, but it is also possible, with prior probability 1 − P1, that the sun only rises 50% of the time. The sun has risen for the past three days. Your commander asks you to determine, with minimum probability of error, whether or not this is a planet on which the sun always rises. For what values of P1 would you answer yes?

Answer:

Use 3d replace The sun has risen for the past three days.

use A replace The sun has risen for the past three days

use B replace The sun has risen for the past three days

P(A | 3d) =

P(B | 3d) =

P(A) = P1

P(B) = P1

P(3d | A) = 1

P(3d) = P1 \* 1 + (1 - P1)\*(0.5 \* 0.5 \* 0.5) = 0.125 + 0.875 P1

So, P(A | 3d) =

P(B | 3d) =

When P(A | 3d) > P(B | 3d), we can say “yes, this is a planet on which the sun always rises”.

So, P1 > 0.125/1.125 (P1 > 0.1111111) is enough.

Q4: Background Recovery

1. Compare the two methods. Which method gives a better result? Why?

The averaging gives a better result, it looks have more details on the background. And from the video capture, there are not too many cars. So, on a particular part which you choose randomly, there are not too many cars passed.

So averaging method is a better choice, looks smoothly, although some part’s color looks strange. K-means try to find clusters and try to find higher peak. But k-means is focuses on most frequently occurring color, maybe it will proper for other kinds of video which need correct color images.

And the averaging method is faster, only takes seconds of time, but the k-means takes minutes of time.

1. How would you use SVD to recover the background?

先得到一个新的矩阵，每列代表一帧，有300列（count）。对整个矩阵做SVD，取出奇异值最大的所对应的矩阵，用该矩阵还原原来的图像即可。