HOMEWORK 4

Jackson Hellmers
9075444662

10/16/2021

Solution 1

Part 1

$E[\mathbb{I}[\hat{x} \neq x]] = 1 * P(\hat{x} \neq x) = 1 - P(\hat{x} = x)$
Since we are looking only at the θ_i that is maximum we get
$1 - P(\hat{x} = x) = 1 - \frac{max(\theta)}{\sum_{i=1}^{k} \theta_i} = 1 - \frac{max(\theta)}{1} = \mathbf{1 - max(\theta)}$

Part 2

When independent we know that $E[\hat{x}x] = E[\hat{x}]E[x]$
Again we have, $E[\mathbb{I}[\hat{x} \neq x]] = 1 * P(\hat{x} \neq x) = 1 - P(\hat{x} = x)$
$1 - P(\hat{x} = x) = 1 - \sum_{i=1}^{k} P(\hat{x} = 1, x = 1) = 1 - \sum_{i=1}^{k} P(\hat{x} = i)P(x = 1)$
This simplifies down to $\mathbf{1 - \sum_{i=1}^{k} \theta_i^2}$

Solution 2

$\mathbb{E}_{Cx\hat{x}} = \sum_{i=1}^{k}(P(x = i) * C_{i\hat{x}})$
Since we already know θ_i is equal to $P(x = i)$ and are looking to minimize our FP/FN cost we get
$\hat{x} = \arg\min_{\hat{x}} \sum_{i=1}^{k}(\theta_i * C_{i\hat{x}})$

Solution 3

Part 1

With no other information about the distribution except for μ and σ^2 our best guess/optimal strategy will be picking the mean such that $x_t = \mu$
If the distribution was know so that we know then we could find the expected payment/cost of the above strategy by taking the sum:
cost $= \sum_{t=1}^{T} \mathbb{E}[(\mu - y_t)^2]$
The expectation of the square of a variable subtracted by its means is equivalent to the variable's variance σ^2.
cost $= \sigma^2 * T$

Part 2

This is exactly what the MSE measures.
MSE $= \frac{1}{T} * \sum_{t=1}^{T}(x_t - \mu)^2$
This will give us the average square distance of that our guess is away from the true mean.

Solution 4

Part 1

$\hat{p}(y = \text{english}) = \mathbf{0.3333}$

$\hat{p}(y = \text{spanish}) = \mathbf{0.3333}$
$\hat{p}(y = \text{japanese}) = \mathbf{0.3333}$

Part 2

$$\theta_{i,e} = \hat{p}(c_i|y = e) = \frac{(\sum_{i=1}^{N}\sum_{j=1}^{M}\mathbb{I}[x_j^{(i)} = c_i, y^{(i)} = e]) + 0.5}{(\sum_{i=1}^{N}\sum_{j=1}^{M}\mathbb{I}[y^{(i)} = e]) + (\mathrm{K}_S * 0.5)}$$

```
a: 0.060170
b: 0.011130
c: 0.021510
d: 0.021970
e: 0.105370
f: 0.018930
g: 0.017480
h: 0.047220
i: 0.055410
j: 0.001420
k: 0.003730
l: 0.028980
m: 0.020520
n: 0.057920
o: 0.064460
p: 0.016750
q: 0.000560
r: 0.053820
s: 0.066180
t: 0.080130
u: 0.026660
v: 0.009280
w: 0.015500
x: 0.001160
y: 0.013840
z: 0.000630
 : 0.179250
```

Figure 1: English Conditional Probabilities

Part 3

```
a: 0.104560
b: 0.008230
c: 0.037530
d: 0.039750
e: 0.113810
f: 0.008600
g: 0.007180
h: 0.004530
i: 0.049860
j: 0.006630
k: 0.000280
l: 0.052940
m: 0.025810
n: 0.054180
o: 0.072490
p: 0.024270
q: 0.007680
r: 0.059300
s: 0.065770
t: 0.035610
u: 0.033700
v: 0.005890
w: 0.000090
x: 0.002500
y: 0.007860
z: 0.002680
 : 0.168260
```

(a) Spanish

```
a: 0.131770
b: 0.010870
c: 0.005490
d: 0.017230
e: 0.060200
f: 0.003880
g: 0.014010
h: 0.031760
i: 0.097030
j: 0.002340
k: 0.057410
l: 0.001430
m: 0.039800
n: 0.056710
o: 0.091160
p: 0.000870
q: 0.000100
r: 0.042800
s: 0.042170
t: 0.056990
u: 0.070620
v: 0.000240
w: 0.019740
x: 0.000030
y: 0.014150
z: 0.007720
 : 0.123450
```

(b) Japanese

Figure 2: Conditional Probabilities for Non-English Languages

Part 4

I received the following counts for characters in 'e10.txt'

a: 164 b: 032 c: 053 d: 057 e: 311 f: 055 g: 051 h: 140 i: 140 j: 003
k: 006 l: 085 m: 064 n: 139 o: 182 p: 053 q: 003 r: 141 s: 186 t: 225
u: 065 v: 031 w: 047 x: 004 y: 038 z: 002 : 498

Part 5

$\log_{10}(\hat{p}\,(x \mid y = e)) = -3405.67889$ which yields $\hat{p}\,(x \mid y = e) = \mathbf{10^{-3405.67889}}$
$\log_{10}(\hat{p}\,(x \mid y = s)) = -3677.29386$ which yields $\hat{p}\,(x \mid y = s) = \mathbf{10^{-3677.29386}}$
$\log_{10}(\hat{p}\,(x \mid y = j)) = -3809.38498$ which yields $\hat{p}\,(x \mid y = j) = \mathbf{10^{-3809.38498}}$

Here we can see that the most probabilistic outcome is the language 'english'. However, this is a maximum likelihood estimate, not a Naive Bayes or MAP estimate.

Part 6

When doing Naive Bayes we take advantage of Baye's rule to calculate $\hat{y} = \arg\max_y(p(y|x)) = \arg\max_y(\frac{p(x|y)*p(y)}{p(x)})$
We can easily find the $p(x)$ as it was already solved for in the last problem.

$p(x) = \frac{1}{3} * (10^{-3405.67889} + 10^{-3677.29386} + 10^{-3809.38498}$

We also have already found our values for $p(y)$ in part one and all are equally probable such that $p(y) = \frac{1}{3}$.
We can now just plug everything together and get:

$$p(y = e \mid x) = \frac{10^{-3405.67889} * \frac{1}{3}}{p(x)} \approx 1$$

$$p(y = s \mid x) = \frac{10^{-3677.29386} * \frac{1}{3}}{p(x)} \approx 0$$
$$p(y = j \mid x) = \frac{10^{-3809.38498} * \frac{1}{3}}{p(x)} \approx 0$$

We can clearly see that the most probable language is English.

Part 7

Confusion matrix with true labels representing columns and predicted labels representing rows. The rows and columns go as [English, Spanish, Japanese].

10	0	0
0	10	0
0	0	10

Test Data Confusion Matrix

Clearly the Naive Bayes classifier functions well on the test set as it has a 100% accuracy.

Part 8

I chose to scramble the characters in the file "e10.txt" so that they would be compared to part 5. After scrambling, the calculated probabilities were as follows:

$\log_{10}(\hat{p}(x \mid y = e)) = -3405.67889$ which yields $\hat{p}(x \mid y = e) = \mathbf{10^{-3405.67889}}$
$\log_{10}(\hat{p}(x \mid y = s)) = -3677.29386$ which yields $\hat{p}(x \mid y = s) = \mathbf{10^{-3677.29386}}$
$\log_{10}(\hat{p}(x \mid y = j)) = -3809.38498$ which yields $\hat{p}(x \mid y = j) = \mathbf{10^{-3809.38498}}$

We can see they are exactly the same as before meaning that scrambling did not impact the prediction. This is because Naive Bayes assumes that all features are independent of each other. This makes computations much more efficient but could cause the loss of some important information. In English, you would not expect to find the letter 'T' immediately after the letter 'Z' but the classifier thinks that the string 'ZTI' and 'ZIT' and both equally probable despite one being an actual english word and the other being nonsense..

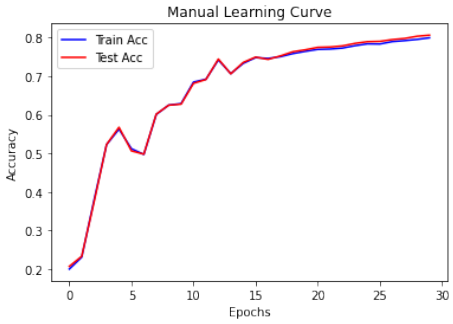
Solution 5

One of the most difficult portions of this problem was finding the correct learning rate that would lead to convergence without causing instability. Due to the long training time of my manual implementation it there was not a lot of opportunity to try a large range of hyper parameters or train on a large number of epochs.

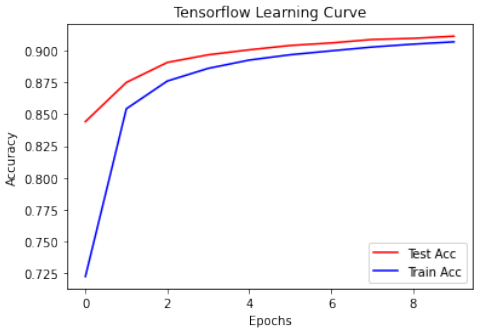
I ended up settling on a learning rate of 0.01 with a batch-size of 10. I chose a batch size of 10 as it would usually allow for each class to be represented in the batch. Batch sizes under 10 began to see a lot of ripple in the learning curve. My weights were initialized using a normal distribution of with a mean of 0 and variance of 0.1. After 30 epochs I was able to obtain a testing accuracy of roughly 0.80

For my chosen framework I used Tensorflow and was able to play with the hyper parameters here more as training was much faster. Using a similar batch-size, learning rate and number of epochs I found tensorflow to outperform my own model, reaching a testing accuracy of over 0.90.

Included below are the learning curves from the two implementations.



(a) Backprop Learning Curve



(b) Tensorflow Learning Curve