CSCI 635

Hw3

Prakhar Gupta(pg9349)

Q1a.

(Code in the q1a.py)

Following parameter and initialization were tried-

Initial hyper parameters-

Experiment 1

beta = 1e-4 # regularization coefficient

alpha = 0.01 # step size coefficient

n_epoch = 10000 # number of epochs (full passes through the dataset)

eps = 0.00001# controls convergence criterion

In the start weights were initialized with zero and this was the results which i got

```
w =  [[0. 0.]
 [0. 0.]]
b =  [[0. 0.]]
Model initial epochs set at  10000
Convergence happened at epoch  1
-------
```

Accuracy = 50.0%

Error = 50.0%

The model started with a loss of 0.6931471805599453 and the loss remained constant for the next iteration suggesting that the derivatives become zero and hence halting happened. The accuracy is low at 50 percent.

I then tried with random initialization of weights as model is not able to break symmetry

217 L = 0.6951594532515735

w = [[0.48223651 0.65486208]

[0.06529103 0.23708999]]

b = [[0.07027227 -0.07027227]]

Model initial epochs set at 10000

Convergence happened at epoch 217

Accuracy = 75.0%

Error = 25.0%

This time model did train and the loss did appear to reduce the epoch until 217 after which halting happened. This increase the accuracy to 75%

So weights were randomly and following hyper parameters were change

Experiment 2

```
beta = 1e-4 # regularization coefficient

alpha = 0.1 # step size coefficient

n_epoch = 10000 # number of epochs (full passes through the dataset)

eps = 0.00001# controls convergence criterion
```

Following is the result-

```
 77 L = 0.6938022600656999

w =  [[0.51986388 0.61659583]

 [0.10294301 0.19926812]]

b =  [[ 0.05697633 -0.05697633]]

Model initial epochs set at  10000

Convergence happened at epoch  77

-------

Accuracy = 25.0%

Error = 75.0%
```

Convergence did happen earlier but accuracy reduced to 25 percent

Experiment 3

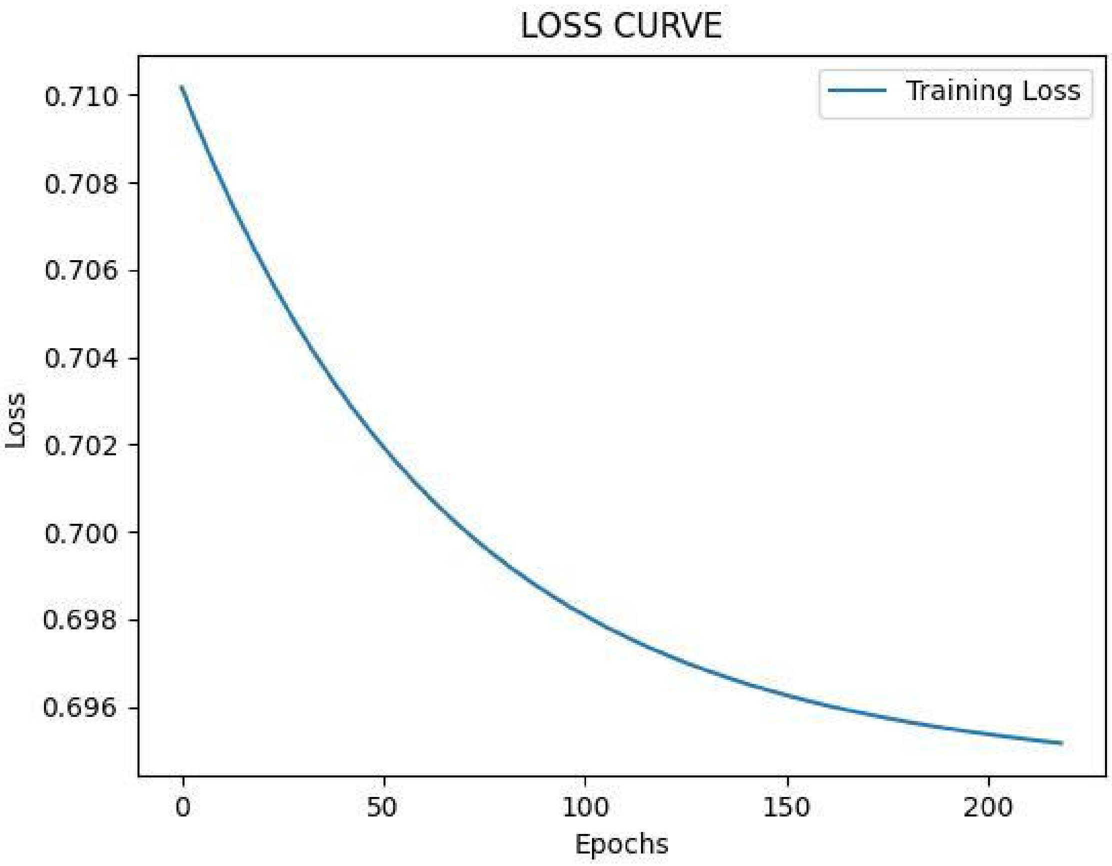
```
beta = 1e-4 # regularization coefficient

alpha = 0.1 # step size coefficient

n_epoch = 10000 # number of epochs (full passes through the dataset)

eps = 0# controls convergence criterion
```

Ran the same experiment but with no halting.

```
 9999 L = 0.6931755297227945

w =  [[0.51455658 0.51455658]

 [0.13683259 0.13683259]]

b =  [[-1.28450948e-15 -1.28985816e-15]]

-------

Accuracy = 50.0%

Error = 50.0%
```

Accuracy remained low becoming 50 percent

Observations-



SO its clear from above experiments that our model hyperparameters are -

beta = 1e-4 # regularization coefficient

alpha = 0.01 # step size coefficient

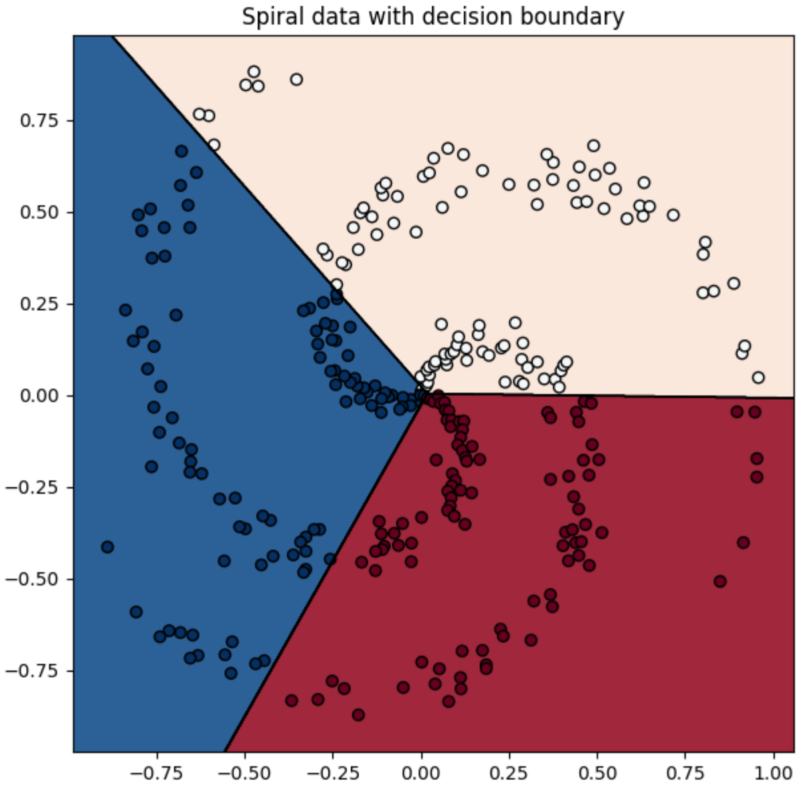
n_epoch = 10000 # number of epochs (full passes through the dataset)

eps = 0.00001# controls convergence criterion

With initializing weights randomly to achieve maximum accuracy.

Well the problem is that XOR is not a linear operation and trying to fit a linear model is causing us to not be able to achieve better results.

Q1b.



Experiment 1:

beta = 1e-3 # regularization coefficient

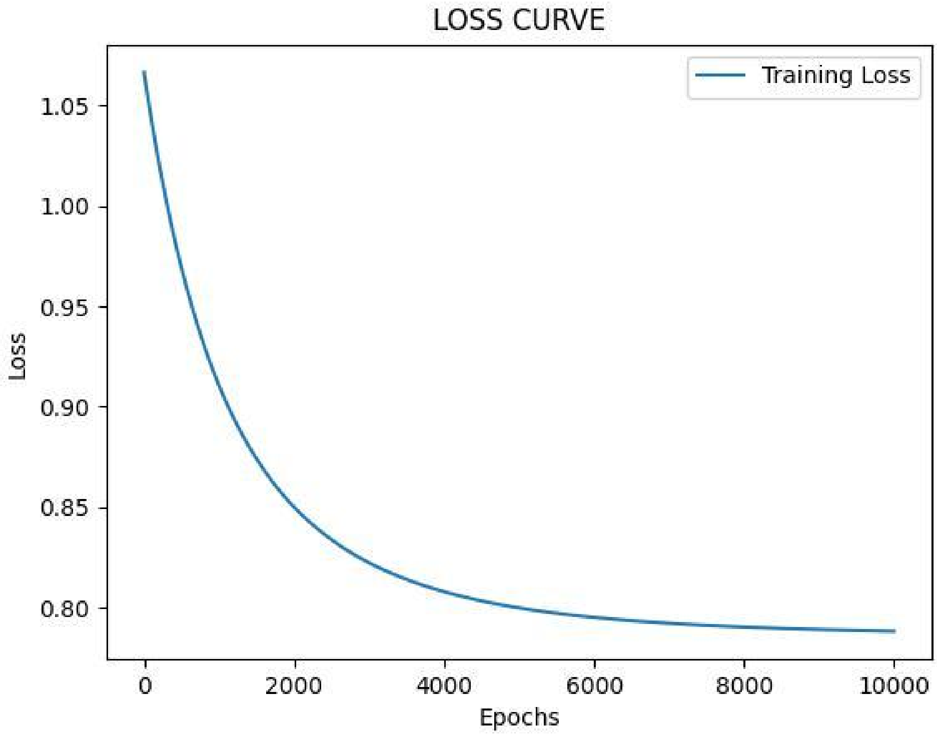
alpha = 0.1 # step size coefficient

n_epoch = 1000 # number of epochs (full passes through the dataset)

eps = 0# controls convergence criterion

The following results were obtained-

```
 999 L = 0.7882119708088751

w =  [[ 1.32542078  1.378404   -1.67461276]

 [-1.94902644  2.6213217  -0.18239425]]

b =  [[-0.00959839 -0.03017578  0.03977418]]

-------

Accuracy = 49.0%

Error = 51.0%
```



Training loss vs epochs

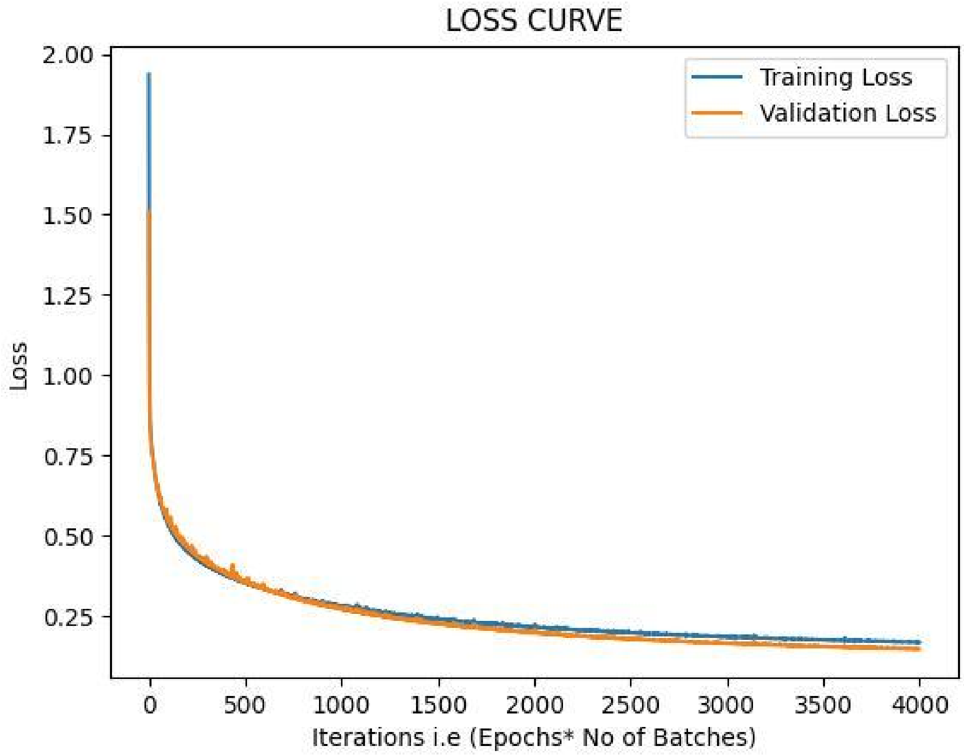
Experiment 2:

```
beta = 1e-3 # regularization coefficient

alpha = 0.01 # step size coefficient

n_epoch = 10000 # number of epochs (full passes through the dataset)

eps = 0# controls convergence criterion
```

Similar results were observed

```
 9999 L = 0.7882207344667038

w =  [[ 1.3252142   1.37821386 -1.67421141]

 [-1.94840135  2.62059758 -0.18229301]]

b =  [[-0.00957902 -0.03014512  0.03972415]]

-------

Accuracy = 49.0%

Error = 51.0%
```

Q1c:



Parameters-

beta = 1e-3 # regularization coefficient

alpha = 0.02 # step size coefficient

n_epoch = 1000 # number of epochs (full passes through the dataset)

eps = 0# controls convergence criterion

Following is the result I got

After Complete epoch 999 Loss = 0.1675019991995233

```
w =  [[ 1.12909032  0.95823972 -1.03732216]
 [ 2.00763728 -0.00947893 -1.49835858]
 [-2.23617024  0.23190005  2.86146465]
 [-0.61886804 -0.5160315   2.6517915 ]]
b =  [[ 0.39731041  0.49250502 -0.88981544]]
```

```
Train Accuracy
```

```
Accuracy = 98.18181818181819%
Error = 1.8181818181818188%
```

```
Validation Accuracy
```

```
Accuracy = 100.0%
Error = 0.0%
```

As we increase the number of epochs the validation accuracy will start decreasing; this is due to overfitting of the model.The following experiment with show this if we keep the parameters same just increase the epochs to 2000 validation accuracy drops

```
After Complete epoch 1999 Loss = 0.14041219476524036
w =  [[ 1.29563681  0.96543703 -1.29179517]
 [ 2.38042175  0.16349689 -2.08254567]
 [-2.77641489  0.1371047   3.43059977]
```

[-0.90937341 -0.87793017 3.18757022]]

b = [[0.52717394 0.85718917 -1.38436311]]

Train Accuracy

Accuracy = 98.18181818181819%

Error = 1.8181818181818188%

Validation Accuracy

Accuracy = 97.5%

Error = 2.500000000000002%

Methods to reduce overfitting are **early stopping which we did ,regularization and dropouts which is common in deep networks.**

Q2

(The functions are implemented in q2.py)

(A description of the code and functions is present in the docstring of each function)

(The paramaters are also present in the csv files with are attached with the submission)

Part A

Likelihood Parameters for the model each cell represents

$P(f_i = x | is\ spam = y)\ where\ x = |f_i|\ and\ y = \{True, False\}\ i.e\ class$

is spam	in html_False	in html_True	has emoji_False	has emoji_True	sent to list_False	sent to list_True	from .com_False	from .com_True	has my name_False	has my name_True	has sig_False	has sig_True
FALSE	0.4130	0.5870	0.8527	0.1473	0.6884	0.3116	0.7246	0.2754	0.3986	0.6014	0.6763	0.3237
TRUE	0.2442	0.7558	0.8023	0.1977	0.9302	0.0698	0.2558	0.7442	0.6512	0.3488	0.3372	0.6628

Following can be used as a lookup to get the likelihood of a feature expressing this value and can be multiplied with respective prior probabilty to get the bayesian probabilty (P(y|f)) using bayes formula

For numerical features mean and variance are given below. This is used to get the gaussian probabilty density with is multiplied with the above terms

Mean of the columns

is spam	# sentences	# words
FALSE	6.1908	70.7705
TRUE	3.9767	68.8372

Variance of the columns

is spam	# sentences	# words
FALSE	6.4163	914.9763
TRUE	3.7642	80.2791

Accuracy of training data

0.89

Part B

Following is the results obtained on the validation data q2b.csv

Accuracy of New data using all columns

0.875

Error

0.125

Part C

Since I created the NB model using OOPS design pattern it was easy to retrain the model by creating a new object of it.

For identifying the feature, I used a process similiar to grid search. I created subsets which have atleast n features(6 in our case) and search across all the combinations of the features to find the combination which maximizes accuracy. Following accuracy on validation data was achieved-

Best Subset

[' has emoji', ' sent to list', ' from .com', ' has my name', ' # sentences', ' # words']

Best Accuracy

0.91

Error

0.08999999999999997