CS 725 Assignment 2 Report

November 3, 2017

1 Implement Fully Connected Neural Networks From Scratch:

Arijit Mukherjee 17305t002

1.1 Kaggle link:

https://www.kaggle.com/c/assignment-2-cs-725

Use Python version 3.x only. Packages allowed:

```
All inbuilt libraries
Numpy
Pandas
```

1.2 Files

```
template.py         : All the Codes
best_model.net      : best model with highest accuracy
report.csv          : 5 Fold Cross Validation Rerpot
output.csv          : Predictions for the test dataset
dataset/            : contains train.csv and test.csv
```

1.3 The Neural Net Configuration

```
Type                : Fully Connected
Input Layer         : 85 Neurons
Hidden Layer 1      : 20 Neurons
Hidden Layer 2      : 10 Neurons
Output Layer        : 10 Neurons
Learning Rate       : 0.1
Momentum Alpha      : 0.3
Regularizer Lambda  : 0.0001
Loss Function       : Categorical Cross Entropy with 2 Norm Regularizer
Optimizer           : Stochastic Gradient Descent with Momentum
Accuracy            : 98.8%
```

1.4 Normalization or Embeding

The variables s1,c1 to s5,c5 are basically Card Color 1-4 and Card Number 1-13 . We will treat these as Categorical Variables and Encode our input vector from 10x1 to a 85x1 vector .

1.5 Hyper Parameter Tuning

K fold Cross validation was performed

No.	Hidden Layers	Lambda	Learning Rate	Test Loss	Training Loss
1	2	10	0.1	6.14328057322281	6.17416545919407
2	2	1	0.1	3.67821983628067	3.61844199996527
3	2	0.1	0.1	2.32941869884646	2.14510858266194
4	3	10	0.1	6.16042021402955	6.17425508138314
5	3	1	0.1	3.63980574670977	3.60276305078353
6	3	0.1	0.1	2.143093238911	2.15280701979785
7	4	10	0.1	6.14620655405346	6.16822812086785
8	4	1	0.1	3.61480378100672	3.59725539296367
9	4	0.1	0.1	2.21845094797518	2.14883564279614
10	2	10	0.01	6.14638992352594	6.15325620206048
11	2	1	0.01	3.51727669437741	3.52516757198236
12	2	0.1	0.01	2.00303320271562	1.98983582533624
13	3	10	0.01	6.15220625069803	6.15255192697429
14	3	1	0.01	3.51427997273981	3.52194283486631
15	3	0.1	0.01	1.99057094453086	1.98775560002518
16	4	10	0.01	6.14476332875311	6.15613260748345
17	4	1	0.01	3.51717839311572	3.52129927548716
18	4	0.1	0.01	1.97546219396641	1.98850101700008
19	2	10	0.001	6.15031759683169	6.15149648484453
20	2	1	0.001	3.51377909669325	3.51779895862937
21	2	0.1	0.001	1.96892279972755	1.97323905522031
22	3	10	0.001	6.1515126786939	6.15163402089916
23	3	1	0.001	3.51573478845444	3.51605191046406
24	3	0.1	0.001	1.9628361527365	1.97006182173045
25	4	10	0.001	6.15201291232243	6.15146127880081
26	4	1	0.001	3.51435328079368	3.51599270131193
27	4	0.1	0.001	1.96495580558206	1.96891592958524
28	2	10	0.0001	6.15133470140591	6.15113294213481
29	2	1	0.0001	3.5154691234683	3.51570697342565
30	2	0.1	0.0001	2.00117099838658	2.00194838249932
31	3	10	0.0001	6.15127999055386	6.15117944083468
32	3	1	0.0001	3.5139780164648	3.51402875298356
33	3	0.1	0.0001	2.00213907248009	2.0034907051758
34	4	10	0.0001	6.15092671014372	6.15139984186751
35	4	1	0.0001	3.51433580030055	3.51438771214769
36	4	0.1	0.0001	1.9876431410847	1.98930018898316
37	2	10	1E-05	6.15033177755154	6.15025068124822
38	2	1	1E-05	3.88649736635822	3.88749737953762

39	2	0.1	1E-05	2.53973839209852	2.53940467174116
40	3	10	1E-05	6.15166054158602	6.15161635107562
41	3	1	1E-05	3.82010252931798	3.82172721027488
42	3	0.1	1E-05	2.88633739676967	2.88709632626891
43	4	10	1E-05	6.15157912381427	6.15155974529636
44	4	1	1E-05	3.79815771965742	3.79923099634328
45	4	0.1	1E-05	2.63664685080926	2.63908254178225

1.6 Training Procedure

1.6.1 Categorical Cross Entropy Loss

As we are doing multiclass classification categorical cross enropy loss with 2 norm regularizer to check the growth of weights is used without softmax .

$$E(\mathbf{w}) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\sigma_k^L\left(\mathbf{x}^{(i)}\right)\right) + \left(1 - y_k^{(i)}\right)\log\left(1 - \sigma_k^L\left(\mathbf{x}^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L}\sum_{i=1}^{s_{l-1}}\sum_{j=1}^{s_l}\left(w_{ij}^l\right)^2$$

1.6.2 Momentum

Here we used momentum which is nothing but adds a fraction of previous ΔW so our current step update becomes

$$W_k = W_{k-1} - \Delta W_k - \alpha \Delta W_{k-1}$$

Here we take α as 0.3 Adding momentum helps the Gradient Descent Converge faster .

1.6.3 Stochastic Gradient Descent

Here we used the Stochastic variant of the gradient descent . Each epoch we iterate over the dataset one by one point .

1.7 Activation Function

Here we used sigmoid Activation function , as we were dealing with class probabilities in the output layer and in the hidden layers we used Sigmoid as it gives better results than ReLu

$$\sigma_j = \frac{1}{1 + e^{-sum_j}}$$

Sigmoid helps us smooth the output and give it a probabilistic interpretation , while it is also differentiable .

1.8 Comments

Though the dataset was 8lac long using only 10000 samples achived good accuracy of 98.8% . Prediction of rare lables , labels with 7,8,9 is the most challenging part as there are very few examples of these classes. To improve the model further data augmentation and synthetic data generation techniques can be used to improve perfomance for the rare classes . The code can be optimized further and exploration work need to be done to see if the model can be trained in parrarel . Exploring needs to done to see if GPUs can be used to train these model fast .