Yue Zhou

6 April 2020

Assignment 2 Report

DD2424

1.Gradient check

I update finite difference method to fit two-layer network and compare the difference between analytical gradient and numerical gradient. Also, I reduce the dimensionality of the input vector to 20*2, this is my results.

Step3:check gradients

```
xmini=Xtr[1:20,1:10]
Ymini=Ytr[:,1:10]

W1,W2, b1,b2= ini_parameters(xmini, Ymini)

gnw1,gnw2, gnb1, gnb2=ComputeGradsNum(X=xmini, Y= Ymini,y=ymini,  W1=W1, W2=W2, b1=b1,b2=b2, h=1e-5,lamda=0)
gaw1,gaw2, gab1,gab2=compute_gradient(X=xmini, Y= Ymini,  W1=W1,W2=W2,b1=b1,b2=b2,lamda=0,svm=False)
print(gnb2-gab2)
print(gnb2-gab2)
print(gnw1-gaw1)
print(gnw2-gaw2)
```

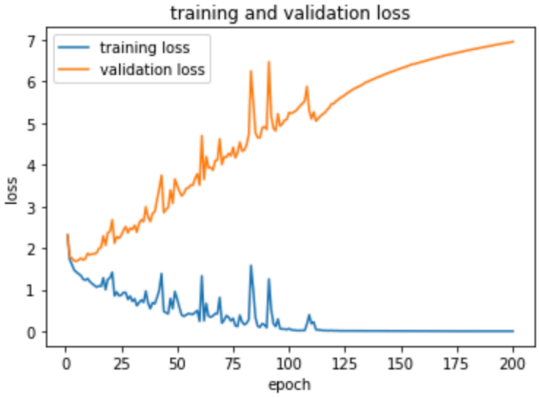
```
[[ 6.10168166e-12]
 [-7.59638186e-12]
 [-2.23240593e-11]
 [-3.85316223e-11]
 [-3.49435064e-12]
 [-1.78358162e-11]
 [-5.65278380e-12]
 [ 1.65535606e-11]
 [ 9.10183040e-12]
 [ 4.14735468e-11]]
[[ 6.10168166e-12]
 [-7.59638186e-12]
 [-2.23240593e-11]
 [-3.85316223e-11]
 [-3.49435064e-12]
 [-1.78358162e-11]
 [-5.65278380e-12]
 [ 1.65535606e-11]
 [ 9.10183040e-12]
 [ 4.14735468e-11]]
[[4.54702506e-08 3.65959045e-08 5.79087417e-08 7.09985609e-08
  6.89043662e-08 6.89961162e-08 6.81714184e-08 6.80846869e-08
  6.56205524e-08 6.65172036e-08 6.58079902e-08 6.67281631e-08
  6.85183355e-08 6.63658246e-08 6.42806711e-08 5.17800209e-08
  4.93928332e-08 5.22924275e-08 4.43946050e-08]
 [1.48786179e-08 1.29880702e-08 1.80442277e-08 2.13515105e-08
  2.13506107e-08 1.99441666e-08 1.91876454e-08 2.20422931e-08
  2.11832268e-08 2.16858404e-08 1.91832365e-08 1.86668178e-08
  1.89733301e-08 1.77962561e-08 1.60025238e-08 1.47228905e-08
  1.35074834e-08 1.65258082e-08 2.48975823e-08]
 [1.90640762e-07 2.28564104e-07 2.37593815e-07 2.02131428e-07
  2.60671201e-07 3.07486432e-07 2.86550326e-07 2.15072312e-07
  2.08775911e-07 1.92792038e-07 1.53265375e-07 1.64587846e-07
  1.39768771e-07 1.25437707e-07 1.52372223e-07 1.54433793e-07
  1.60670416e-07 1.51302870e-07 1.08843402e-07]
 [1.84249856e-07 2.06729751e-07 2.11940070e-07 1.93100676e-07
  2.32372130e-07 2.62512234e-07 2.47892404e-07 1.73367598e-07
  1.47139794e-07 1.36831660e-07 1.17301007e-07 1.24483779e-07
  1.10401940e-07 9.83811142e-08 1.03232305e-07 1.05686267e-07
```

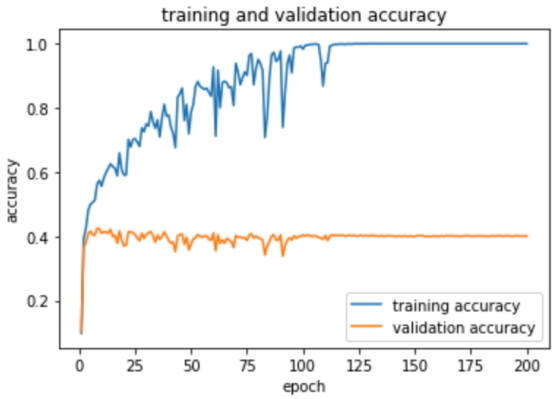
```
 1.40603398e-08  2.43767989e-06  2.81328614e-07  4.15515101e-10
 3.94434031e-10  3.53190276e-09  8.01320112e-08  1.30553412e-07
 3.74046597e-09  7.16869948e-08  7.61527362e-08  1.49771906e-08
 1.05615810e-07  9.08134206e-09  1.25883193e-07  7.80417770e-10
 9.51667407e-08  4.47561989e-08  1.17062059e-09  6.36431845e-08
 3.16818130e-08  1.76614562e-08  2.14516822e-07  3.04753906e-07
 1.12688671e-08  2.73194733e-07  2.48111405e-07  2.20720406e-09
 2.04321944e-09  4.46402432e-07  1.02380650e-06  8.99053630e-09
 1.41602069e-08  8.88903395e-07  3.35543575e-09  4.03184334e-07
 4.11478793e-08  3.85081374e-07  1.13587251e-08  7.11708186e-09
 3.16644822e-07  1.17409489e-09  5.15708745e-09  2.67946403e-08
 1.72136089e-07  7.53942698e-08]
[ 1.17892751e-09  1.68939197e-07  4.12741531e-07  3.66052470e-08
 1.06426808e-07  1.08777178e-06  1.18544189e-07  1.95106137e-08
 6.43129529e-10  1.20712010e-08  1.08099869e-06  2.42597294e-07
 1.71450891e-08  6.54194516e-07  1.29697813e-06  6.31277175e-08
 1.37434148e-06  3.15442627e-08  2.50810049e-08  3.23761461e-09
 1.72509495e-07  1.58254012e-07  3.20413803e-08  1.38403663e-07
 4.95698622e-07  7.07748556e-08  2.55208198e-06  1.08085743e-07
 3.89040195e-08  1.80347092e-07  5.28506333e-08  4.61224727e-10
 8.56885361e-09  1.85202516e-07  2.49733181e-07  6.58236025e-08
 1.09976040e-07  2.02941330e-07  4.22434581e-08  8.88788798e-08
 2.26028859e-08  1.97139673e-07  1.04886856e-07  1.90086318e-07
 1.22823989e-07  4.73651434e-08  1.32872802e-07  4.55332544e-07
 1.30152465e-07  2.95095656e-07]
[ 6.53885180e-10  9.69571508e-08  3.37249601e-07  2.70069741e-08
 5.26426961e-08  6.31395360e-07  6.25320936e-08  6.01314929e-09
 9.72494713e-10  9.18499677e-09  4.92090980e-07  1.35295408e-07
 1.67126580e-08  3.29723201e-07  5.00854114e-07  4.91035691e-08
 6.67942215e-07  2.13077672e-08  2.86600155e-09  2.40025934e-09
 1.48668888e-07  1.39339223e-07  1.04524074e-08  1.14660063e-07
 2.00218726e-07  6.78063197e-08  1.15909401e-06  6.43843972e-08
 1.15373515e-08  1.35068382e-07  8.09656179e-09 -2.27823365e-12
 6.35409377e-09  9.68219877e-08  6.76093046e-08  3.98632236e-08
 4.92223797e-08  4.04254787e-08  2.37112842e-08  1.71828003e-08
 2.02774643e-08  1.15395453e-07  5.32655703e-08  6.42257568e-08
 6.17227270e-08  1.48908139e-08  4.37733194e-08  1.76265224e-07
 9.25231577e-08  2.79820330e-07]
[ 1.97094007e-09  2.73772353e-07  1.86732617e-06  1.85564645e-07
```

The difference is very small, around 1e-7.

Also, when I move to sanity check, with lambda=0 and epochs=200, I can achieve 100% accuracy on training data although overfit on validation data, which indicates that my gradient computation and mini-batch gradient descent algorithms are okay.

```
the 197 epoch for training data,  the loss is 0.0021302735624896853 the accuracy is  1.0
the 197 epoch for validation data,  the loss is 6.939105581699407 the accuracy is  0.4005
the 198 epoch for training data,  the loss is 0.0020910124533825014 the accuracy is  1.0
the 198 epoch for validation data,  the loss is 6.9485010658449795 the accuracy is  0.4004
the 199 epoch for training data,  the loss is 0.00206898231817072 the accuracy is  1.0
the 199 epoch for validation data,  the loss is 6.959201042179715 the accuracy is  0.4005
```

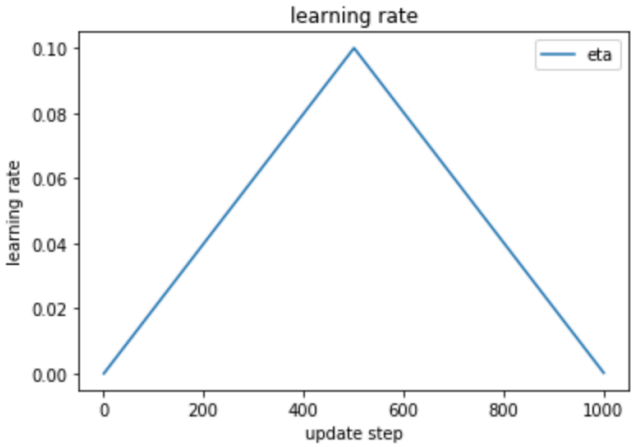


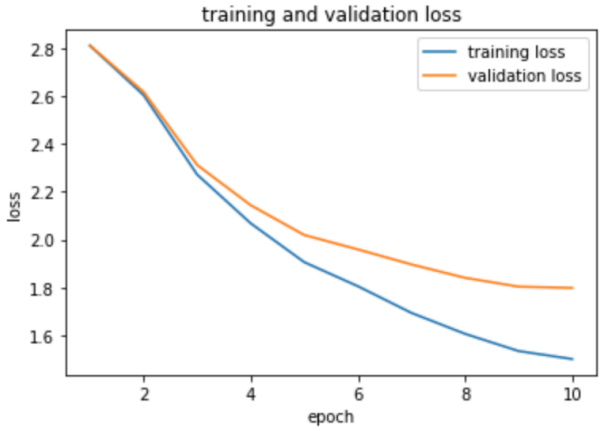


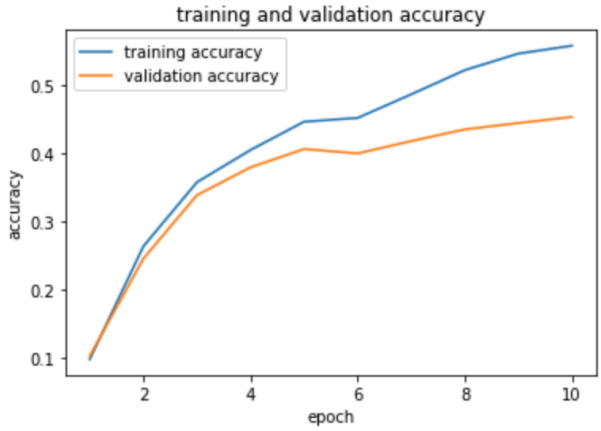
2.Cyclical learning rates

2.1 One cycle of training

-eta min=1e-5, eta_max=1e-1, lamda=0..1, ns=500



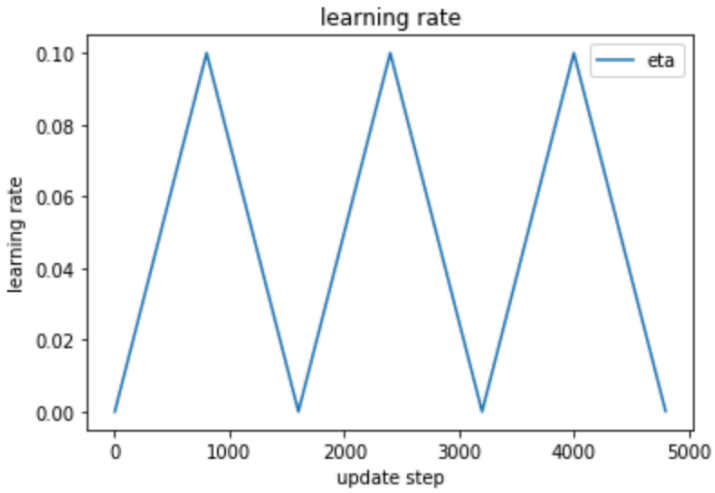


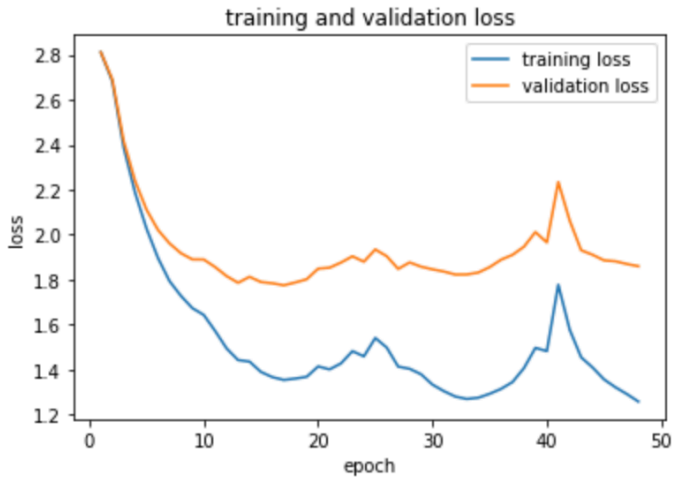


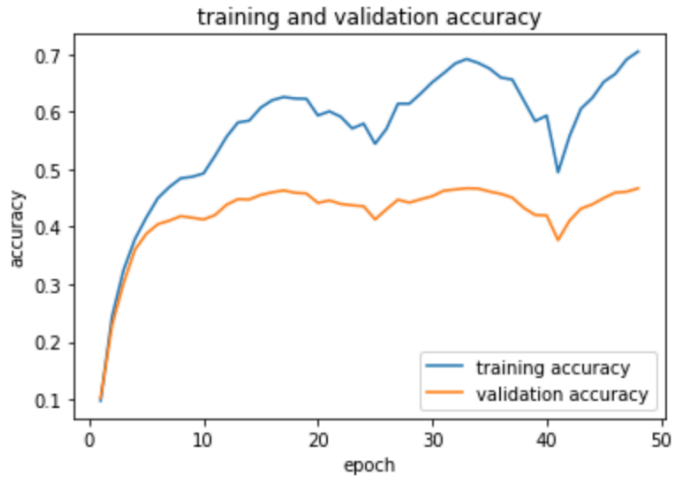
At the end the test accuracy is 45.26%, I plot the evaluation scores every update step.

2.2 Three cycles of training

-eta min=1e-5, eta_max=1e-1, lamda=0..1, ns=800







At the end the test accuracy is 46.69%, I plot the evaluation scores every update step.

3. Finding the best lambda

3.1 Coarse search for lambda

The range of lambda values I search for is [-5,-1]. I calculate test accuracy after training with 2 cycles and ns=200, eta_min=1e-5, eta_max=1e-1.

```
for i in range(8):
    l1=find_lamda(-1,-5,i)
    c=one_hidden_layer_classifier(lamda=10**l1, n_epochs=8, n_batch=100, etamin=1e-5,etamax=0.1,k=2)
    c.fit(Xtr, Ytr, ytr, Xva, Yva, yva)
    print("when l is",l1,"the test accuracy is",c.predict(Xte, Yte, yte))
    i+=1
```

The followings are the results:

when l is -2.804745984290701 the test accuracy is 0.5114
when l is -2.8263802328361383 the test accuracy is 0.5138
when l is -1.2094709725933188 the test accuracy is 0.4257
when l is -3.195509725199781 the test accuracy is 0.5132
when l is -2.3250843646633013 the test accuracy is 0.5121
when l is -2.2252818919795283 the test accuracy is 0.511
when l is -4.870530899288644 the test accuracy is 0.5116
when l is -4.373814541471651 the test accuracy is 0.5102

The three best networks are:

lambda=1e-2.83, test accuracy= 51.38%;

lambda=1e-3.10, test accuracy=51.32%;

lambda=1e-2.33, test accuracy=51.21%.

3.2 Fine search for lambda

The range of lambda values I search for is [-2,-4]. I calculate test accuracy after training with 2 cycles and ns=200, eta_min=1e-5, eta_max=1e-1.

```
for i in range(10):
    l1=find_lamda(-2,-4,i)
    c=one_hidden_layer_classifier(lamda=10**l1, n_epochs=8, n_batch=100, etamin=1e-5,etamax=0.1,k=2)
    c.fit(Xtr, Ytr, ytr, Xva, Yva, yva)
    print("when l is",l1,"the test accuracy is",c.predict( Xte, Yte, yte))
    i+=1
```

The following are the results:

when l is -2.9023729921453505 the test accuracy is 0.5118
when l is -2.913190116418069 the test accuracy is 0.5104
when l is -2.1047354862966596 the test accuracy is 0.5095
when l is -3.0977548625998903 the test accuracy is 0.5121
when l is -2.6625421823316504 the test accuracy is 0.5103
when l is -2.612640945989764 the test accuracy is 0.5114
when l is -3.935265449644322 the test accuracy is 0.5107
when l is -3.6869072707358255 the test accuracy is 0.5137
when l is -3.837242449441949 the test accuracy is 0.5079
when l is -3.1335969431465314 the test accuracy is 0.509

So I choose l=-2.82638 to run the final results.

4.Final results

For my best found lambda setting, I trained the network on all the training data. The training and validation cost and accuracy is below. The test accuracy is 51.46% in the end.

-lamda=10** (2.82638), n_epochs=20, n_batch=100, etamin=1e-5, etamax=0.1, ns=500