Write up Project 3

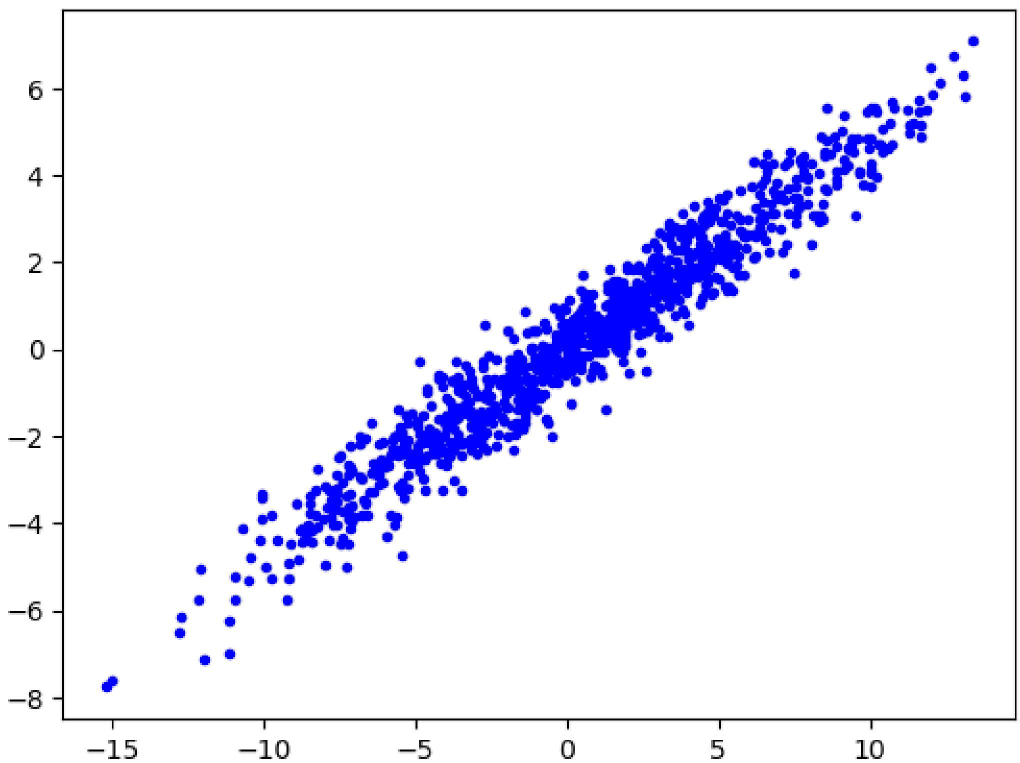
December 11, 2023

[1]:
```
from numpy import *
from matplotlib.pyplot import *
import matplotlib.pyplot as plt
import util, dr, datasets, softmax, runClassifier
from utils import *
from softmax import *
from runClassifier import *
```

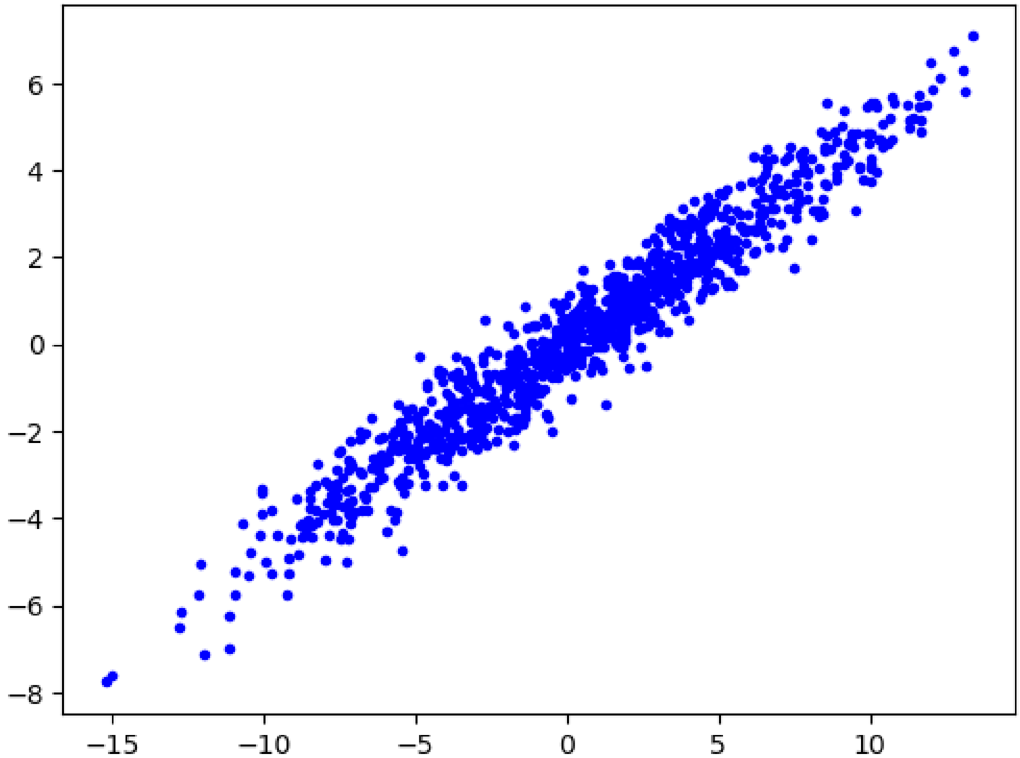
Part 1: PCA

Qpca1: Implement PCA

[9]:
```
Si = util.sqrt(array([[3,2],[22,4]]))
x = dot(random.randn(1000,2), Si)
plot(x[:,0], x[:,1], 'b.')
show(False)
dot(x.T,x)/real(x.shape[0] - 1)
```



[9]:
```
array([[26.18828362, 12.38687024],
       [12.38687024,  6.24247454]])
```



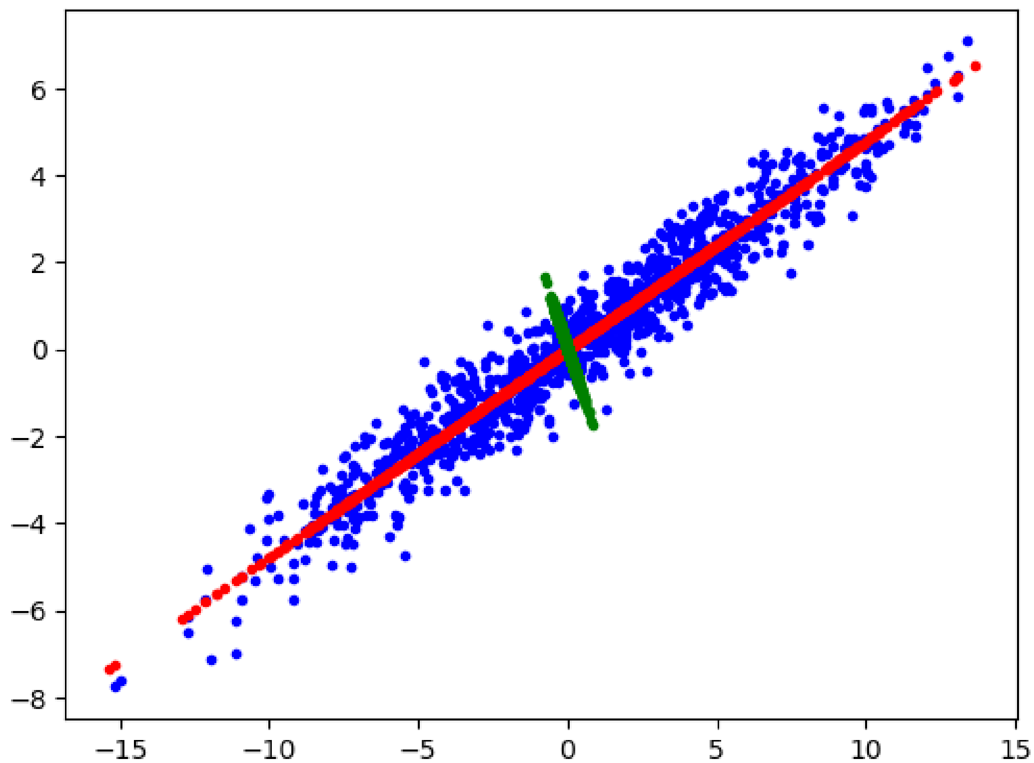
[10]:
```
(P, Z, evals) = dr.pca(x,2)
Z
```

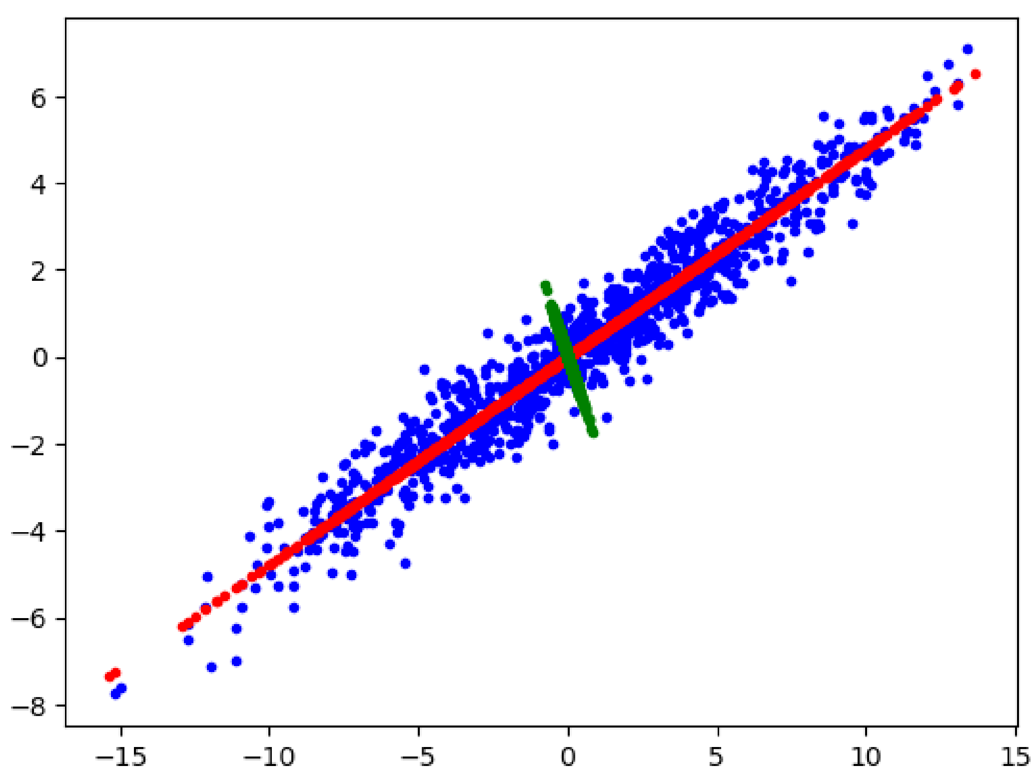
[10]:
```
array([[ 0.90201048, -0.43171413],
       [ 0.43171413,  0.90201048]])
```

[11]:
```
evals
```

[11]:
```
array([31.83153594,  0.31273565])
```

[12]:
```
x0 = dot(dot(x, Z[:,0]).reshape(1000,1), Z[:,0].reshape(1,2))
x1 = dot(dot(x, Z[:,1]).reshape(1000,1), Z[:,1].reshape(1,2))
plot(x[:,0], x[:,1], 'b.', x0[:,0], x0[:,1], 'r.', x1[:,0], x1[:,1], 'g.')
show(False)
```


6
4
2
0
−2
−4
−6
−8
−15
−10
−5
0
5
10
15


6
4
2
0
−2
−4
−6
−8
−15
−10
−5
0
5
10
15

[14]:
```
(X,Y) = datasets.loadDigits()
(P1,Z1,evals1) = dr.pca(X, 784)
evals
```

[14]:
```
array([ 5.47145902e-02,  4.32457401e-02,  3.91832356e-02,  3.07589801e-02,
        2.97240699e-02,  2.48634922e-02,  1.98552656e-02,  1.75982431e-02,
        1.57833269e-02,  1.31208707e-02,  1.27918802e-02,  1.18266157e-02,
        1.12559221e-02,  1.00752373e-02,  9.36845932e-03,  8.73601732e-03,
        8.27005062e-03,  7.92899274e-03,  7.23521657e-03,  6.85924774e-03,
        6.42706546e-03,  6.14996142e-03,  5.91821028e-03,  5.67737906e-03,
        5.40356132e-03,  5.30190215e-03,  5.02203749e-03,  4.59637089e-03,
        4.42538322e-03,  4.36981194e-03,  4.16187116e-03,  4.11031674e-03,
        3.77459052e-03,  3.62187854e-03,  3.41085954e-03,  3.28497607e-03,
        3.23280900e-03,  3.19587689e-03,  3.02982002e-03,  2.87978872e-03,
        2.83078667e-03,  2.74927351e-03,  2.62589089e-03,  2.54642734e-03,
        2.47927464e-03,  2.37555612e-03,  2.29383593e-03,  2.28594556e-03,
        2.20977922e-03,  2.08730066e-03,  1.99999581e-03,  1.96943065e-03,
        1.92821820e-03,  1.88309395e-03,  1.84886396e-03,  1.76088832e-03,
        1.67020455e-03,  1.61444912e-03,  1.59805234e-03,  1.57744026e-03,
        1.48050938e-03,  1.44561925e-03,  1.43009511e-03,  1.40344365e-03,
        1.37760752e-03,  1.34557952e-03,  1.31369874e-03,  1.29776443e-03,
        1.26802185e-03,  1.20748247e-03,  1.19311640e-03,  1.13277494e-03,
        1.10668220e-03,  1.08849526e-03,  1.06129719e-03,  1.04324556e-03,
        1.01409526e-03,  1.00687447e-03,  9.81023790e-04,  9.38761970e-04,
        9.14566689e-04,  9.11766083e-04,  8.98271843e-04,  8.61728296e-04,
        8.51322737e-04,  8.37576333e-04,  8.07377310e-04,  7.89436310e-04,
        7.77157242e-04,  7.65601500e-04,  7.43889898e-04,  7.36837174e-04,
        7.23782179e-04,  7.16183052e-04,  6.88886751e-04,  6.76881361e-04,
        6.73658879e-04,  6.57964557e-04,  6.31789458e-04,  6.14600872e-04,
        6.06852214e-04,  5.97638691e-04,  5.82295974e-04,  5.75874171e-04,
        5.69167770e-04,  5.66582916e-04,  5.56170605e-04,  5.46447592e-04,
        5.35887081e-04,  5.29480645e-04,  5.12646375e-04,  5.10076375e-04,
        4.98901433e-04,  4.92492964e-04,  4.83279552e-04,  4.77823523e-04,
        4.74356288e-04,  4.58393177e-04,  4.47510129e-04,  4.41138118e-04,
        4.34427000e-04,  4.26534001e-04,  4.22078706e-04,  4.20811833e-04,
        4.16611215e-04,  4.07636898e-04,  4.01008493e-04,  3.88768112e-04,
        3.84665028e-04,  3.82008801e-04,  3.70120096e-04,  3.68936048e-04,
        3.61495769e-04,  3.52564002e-04,  3.50413432e-04,  3.43551550e-04,
        3.42456916e-04,  3.34070452e-04,  3.31285706e-04,  3.25466266e-04,
        3.23319301e-04,  3.21575937e-04,  3.13465865e-04,  3.09027164e-04,
        3.02370057e-04,  2.97280504e-04,  2.94665985e-04,  2.92498807e-04,
        2.91135494e-04,  2.87403980e-04,  2.83721082e-04,  2.77237998e-04,
        2.73678546e-04,  2.70133830e-04,  2.69075520e-04,  2.65467924e-04,
        2.60293374e-04,  2.58426988e-04,  2.54266891e-04,  2.53192164e-04,
```

2.49764944e-04, 2.42950419e-04, 2.42151942e-04, 2.41786314e-04,
2.40184890e-04, 2.34384637e-04, 2.29678760e-04, 2.25975527e-04,
2.24052785e-04, 2.22648492e-04, 2.21183849e-04, 2.18103133e-04,
2.16772283e-04, 2.11643742e-04, 2.10557240e-04, 2.06598941e-04,
2.03725106e-04, 2.03002825e-04, 2.01737235e-04, 1.99838248e-04,
1.97020900e-04, 1.95882771e-04, 1.90553874e-04, 1.90035654e-04,
1.86202179e-04, 1.84195282e-04, 1.82248919e-04, 1.81374403e-04,
1.78634663e-04, 1.77003084e-04, 1.76734926e-04, 1.74827021e-04,
1.73861570e-04, 1.69074913e-04, 1.68301018e-04, 1.65926793e-04,
1.64763869e-04, 1.63508376e-04, 1.60460373e-04, 1.58903225e-04,
1.57550898e-04, 1.57069298e-04, 1.54714752e-04, 1.53382367e-04,
1.52196112e-04, 1.51851924e-04, 1.49036629e-04, 1.47581845e-04,
1.47015964e-04, 1.45691497e-04, 1.44753088e-04, 1.41471392e-04,
1.39746586e-04, 1.39175790e-04, 1.37987629e-04, 1.36408508e-04,
1.33085110e-04, 1.32389884e-04, 1.32027803e-04, 1.29974745e-04,
1.28716136e-04, 1.28319783e-04, 1.27568188e-04, 1.25324466e-04,
1.24347102e-04, 1.22744275e-04, 1.22300425e-04, 1.20760186e-04,
1.19779984e-04, 1.17751428e-04, 1.16606506e-04, 1.15922466e-04,
1.15017447e-04, 1.12928827e-04, 1.11734652e-04, 1.11213415e-04,
1.10236654e-04, 1.08918378e-04, 1.08252462e-04, 1.06430084e-04,
1.04553397e-04, 1.04281598e-04, 1.04054083e-04, 1.01576056e-04,
1.01343340e-04, 1.01267465e-04, 1.00522423e-04, 9.85053164e-05,
9.78816815e-05, 9.70690379e-05, 9.62925456e-05, 9.55812780e-05,
9.45559635e-05, 9.36019509e-05, 9.24278471e-05, 9.19656175e-05,
9.14391679e-05, 8.96372447e-05, 8.93024179e-05, 8.75651715e-05,
8.69322559e-05, 8.54624033e-05, 8.52465643e-05, 8.45712126e-05,
8.42354339e-05, 8.35126534e-05, 8.26637951e-05, 8.16250614e-05,
8.10797669e-05, 7.96140119e-05, 7.94147014e-05, 7.88650653e-05,
7.76851426e-05, 7.69168134e-05, 7.59915224e-05, 7.54955439e-05,
7.46249502e-05, 7.41006358e-05, 7.36929743e-05, 7.23523711e-05,
7.20429520e-05, 7.07902291e-05, 7.06927386e-05, 7.01269353e-05,
6.96346942e-05, 6.87641980e-05, 6.82166674e-05, 6.76223162e-05,
6.70368715e-05, 6.65200035e-05, 6.56722876e-05, 6.51808836e-05,
6.42906211e-05, 6.36604944e-05, 6.29414925e-05, 6.27787655e-05,
6.12849305e-05, 6.09263075e-05, 6.01884006e-05, 5.95464554e-05,
5.80776976e-05, 5.78205279e-05, 5.75292392e-05, 5.69274446e-05,
5.65771276e-05, 5.64412985e-05, 5.57036459e-05, 5.50499316e-05,
5.49942491e-05, 5.36323282e-05, 5.30991777e-05, 5.26568756e-05,
5.23712794e-05, 5.19583484e-05, 5.18381924e-05, 5.15401960e-05,
5.04554852e-05, 4.90569702e-05, 4.86487355e-05, 4.83589125e-05,
4.76003331e-05, 4.70510781e-05, 4.59385460e-05, 4.57307845e-05,
4.52785302e-05, 4.48145357e-05, 4.46440287e-05, 4.40717511e-05,
4.36747605e-05, 4.32923298e-05, 4.29148619e-05, 4.26332825e-05,
4.19585597e-05, 4.16284803e-05, 4.11736824e-05, 4.06097570e-05,
4.04030818e-05, 4.01164886e-05, 3.95221559e-05, 3.92048548e-05,
3.88512263e-05, 3.85824261e-05, 3.82123548e-05, 3.75291417e-05,
3.72292299e-05, 3.69259350e-05, 3.65790506e-05, 3.62293599e-05,

3.61688812e-05,	3.57178110e-05,	3.52504156e-05,	3.46303776e-05,
3.43365755e-05,	3.37485856e-05,	3.36655102e-05,	3.30809444e-05,
3.29371687e-05,	3.22457276e-05,	3.17451188e-05,	3.11110595e-05,
3.10819608e-05,	3.06394735e-05,	3.01036807e-05,	2.99597482e-05,
2.94869075e-05,	2.92286816e-05,	2.89553410e-05,	2.86577988e-05,
2.81775649e-05,	2.78537419e-05,	2.76491103e-05,	2.74490176e-05,
2.69949143e-05,	2.66711158e-05,	2.63043530e-05,	2.59990966e-05,
2.55178324e-05,	2.54420060e-05,	2.49967869e-05,	2.47386118e-05,
2.44478402e-05,	2.42861628e-05,	2.37778737e-05,	2.34678050e-05,
2.29924394e-05,	2.27444158e-05,	2.25710993e-05,	2.21925530e-05,
2.16898469e-05,	2.12574642e-05,	2.10710698e-05,	2.06125368e-05,
2.04801867e-05,	2.03528951e-05,	2.01852754e-05,	2.00109236e-05,
1.96358067e-05,	1.95357198e-05,	1.88062148e-05,	1.84740717e-05,
1.82343730e-05,	1.79672046e-05,	1.78883200e-05,	1.77158139e-05,
1.75571151e-05,	1.72617510e-05,	1.70780959e-05,	1.67246044e-05,
1.65874069e-05,	1.64043907e-05,	1.61572993e-05,	1.59549823e-05,
1.56642274e-05,	1.55508890e-05,	1.52751700e-05,	1.50113639e-05,
1.45628889e-05,	1.43395732e-05,	1.40217384e-05,	1.37922572e-05,
1.37149459e-05,	1.34724106e-05,	1.34004024e-05,	1.31395277e-05,
1.28611756e-05,	1.27115880e-05,	1.25654559e-05,	1.23064597e-05,
1.21871093e-05,	1.20404242e-05,	1.19082819e-05,	1.17445668e-05,
1.14430627e-05,	1.13639877e-05,	1.09683346e-05,	1.07615238e-05,
1.06384075e-05,	1.05582384e-05,	1.03033757e-05,	1.01843376e-05,
9.99360386e-06,	9.88765753e-06,	9.45538175e-06,	9.43379699e-06,
9.33984633e-06,	9.25979331e-06,	9.00558539e-06,	8.88594799e-06,
8.76931369e-06,	8.67576620e-06,	8.44589675e-06,	8.42047973e-06,
8.29299911e-06,	8.18500338e-06,	8.10426232e-06,	7.86968087e-06,
7.80326767e-06,	7.61047354e-06,	7.54069273e-06,	7.32839510e-06,
7.02790061e-06,	6.93803992e-06,	6.86839425e-06,	6.81302794e-06,
6.72974568e-06,	6.61403985e-06,	6.50509538e-06,	6.33303381e-06,
6.22118912e-06,	5.97738030e-06,	5.81367159e-06,	5.66976470e-06,
5.57648468e-06,	5.54174989e-06,	5.42658176e-06,	5.19783419e-06,
5.13442795e-06,	5.06890822e-06,	4.98959410e-06,	4.82534928e-06,
4.66035688e-06,	4.56198566e-06,	4.51001805e-06,	4.36820868e-06,
4.29161273e-06,	4.24292681e-06,	4.13953158e-06,	4.08394845e-06,
3.84565097e-06,	3.81561572e-06,	3.68911200e-06,	3.65366772e-06,
3.62286115e-06,	3.48300688e-06,	3.41704003e-06,	3.26430511e-06,
3.21058430e-06,	3.14052045e-06,	3.06495818e-06,	3.01867376e-06,
2.89121200e-06,	2.82203991e-06,	2.76596354e-06,	2.68021341e-06,
2.57159092e-06,	2.48812905e-06,	2.43153349e-06,	2.37842654e-06,
2.26778695e-06,	2.20089413e-06,	2.12019860e-06,	2.03927844e-06,
2.00474900e-06,	1.96567566e-06,	1.86980680e-06,	1.85320127e-06,
1.80781363e-06,	1.75081389e-06,	1.69244461e-06,	1.64184031e-06,
1.60386521e-06,	1.57252147e-06,	1.49681811e-06,	1.39316172e-06,
1.32824152e-06,	1.27965819e-06,	1.22198261e-06,	1.16411084e-06,
1.14888666e-06,	1.07544038e-06,	1.03222807e-06,	9.91356743e-07,
9.64565963e-07,	9.55180458e-07,	8.87585979e-07,	8.67379177e-07,

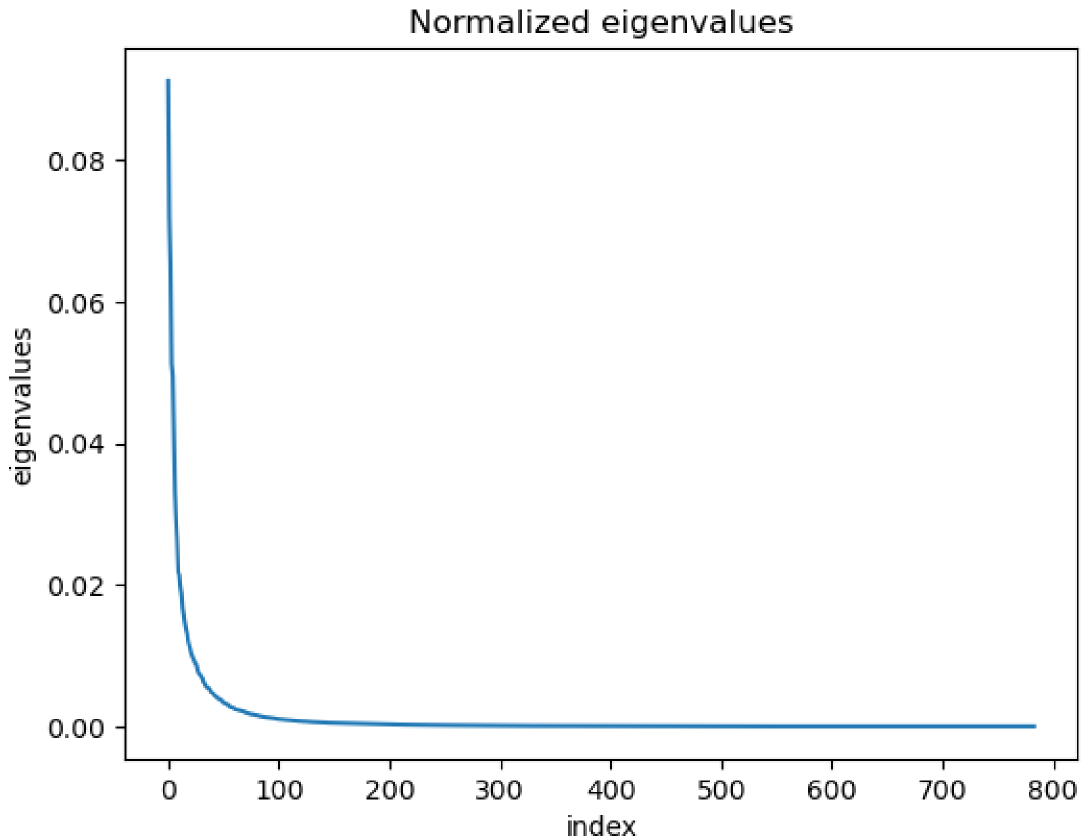
```
8.06333967e-07,  7.77153861e-07,  7.59038994e-07,  6.98664082e-07,
6.45956164e-07,  6.45187032e-07,  6.35692489e-07,  5.85869419e-07,
4.82994400e-07,  4.25013771e-07,  4.13274222e-07,  3.15553719e-07,
3.05291620e-07,  2.60417758e-07,  2.57564150e-07,  2.44620688e-07,
2.30361353e-07,  2.06364199e-07,  1.87638129e-07,  1.63753931e-07,
1.52234408e-07,  1.07680580e-07,  1.05669696e-07,  9.59688618e-08,
7.11282118e-08,  7.00776131e-08,  6.30297801e-08,  5.68270450e-08,
4.68948543e-08,  4.03696499e-08,  2.95752817e-08,  2.80017795e-08,
2.45719490e-08,  2.03688879e-08,  1.49558280e-08,  1.32228778e-08,
4.82729785e-09,  4.06940783e-09,  3.34976851e-09,  1.46713596e-09,
4.57968796e-10,  3.48585266e-10,  1.93364069e-10,  8.14009864e-20,
8.14009864e-20,  7.95882817e-20,  5.63474128e-20,  4.36301215e-20,
4.19216815e-20,  1.26791379e-20,  1.26791379e-20,  3.11508896e-21,
3.11508896e-21,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
```

```
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
 0.00000000e+00,  0.00000000e+00, -1.30009069e-22, -4.78099872e-21,
-4.78099872e-21, -5.03227736e-21, -5.03227736e-21, -8.00168618e-21,
-1.39413532e-20, -1.39413532e-20, -3.56875226e-20, -3.56875226e-20,
-4.05460168e-20, -4.87670768e-20, -4.87670768e-20, -7.37491520e-20])
```

Qcpa2

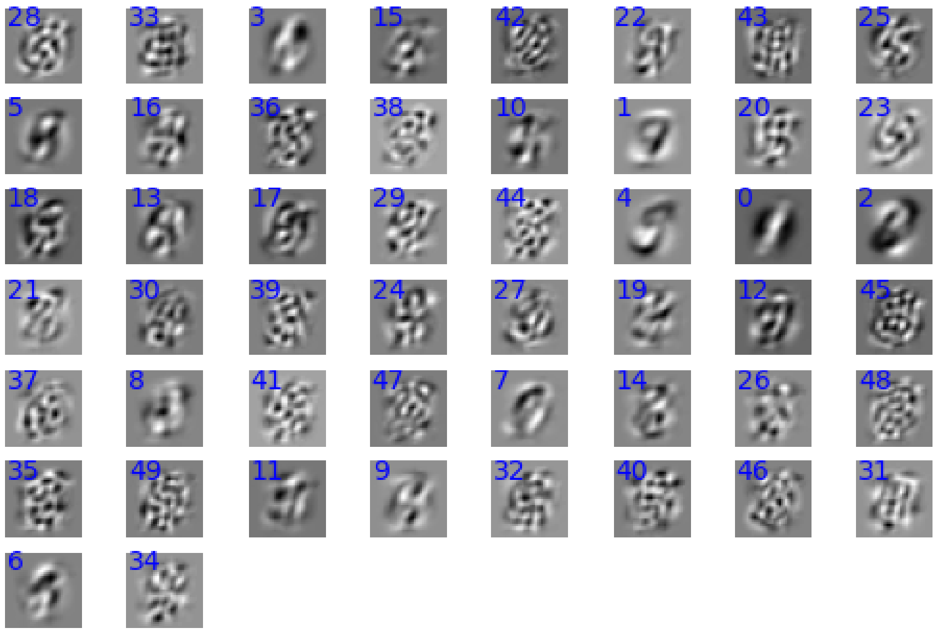
[18]:
```
#code for Qpca2:
normal_evals = evals1/sum(evals)
index = range(0, len(normal_evals))
plot(index, normal_evals)
xlabel('index')
ylabel('eigenvalues')
title('Normalized eigenvalues')

cum_sum = cumsum(normal_evals)
print(argmax(cum_sum > 0.9))
print(argmax(cum_sum > 0.95))
```

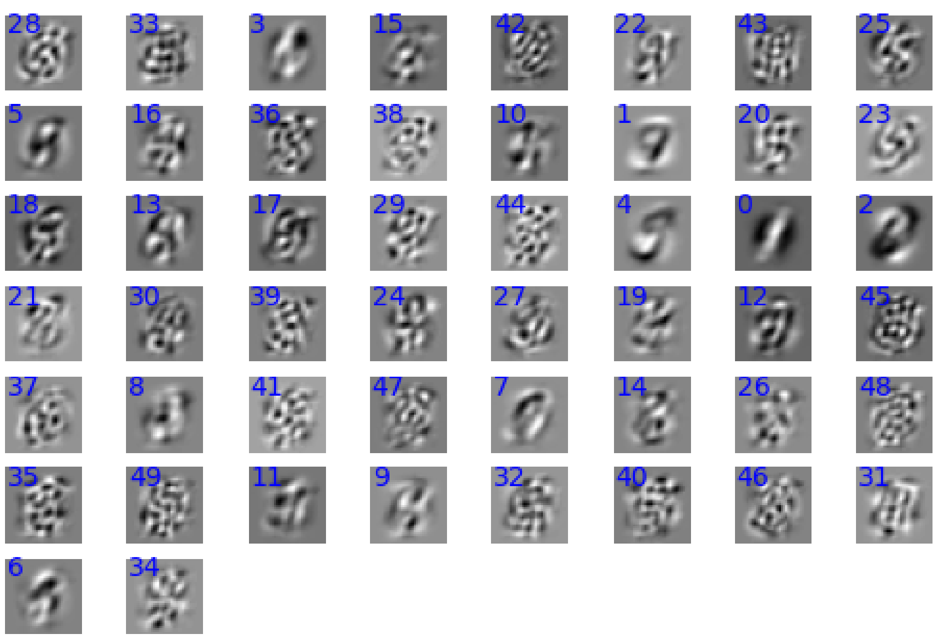
```
81
135
```



There are 81 eigenvectors we need to inlcuded before we've accounted for 90% of the variance. For 95% of the variance, the number goes up to 135.

[17]:
```
#Qpca3:
util.drawDigits(Z.T[:50,:], arange(50))
show(False)
```





Qpca3: These graphs do not exactly look like digit. Some of them somewhat resemble the digits but most of them are not very clear. They are being blurry due to the fact that projecting the eigenvectors onto the top 50 eigenvectors yeilds 82.7% accuracy of the total variance of the dataset. Hence, we compress the image to a lower dimension, which explains why the graphs are not clearly defined.

Part 2: Softmax Regression.

Qsr1:

1) Given the probability: $P = [y = i] = \frac{e^{\vec{w_i} \cdot \vec{x}}}{\sum_j e^{\vec{w_j} \cdot \vec{x}}}$

Sum of the probabilities:

$$\sum_i P = [y = i] = \sum_i \frac{e^{\vec{w_i} \cdot \vec{x}}}{\sum_j e^{\vec{w_j}} \cdot \vec{x}} = \frac{\sum_i e^{\vec{w_i} \cdot \vec{x}}}{\sum_j e^{\vec{w_j}} \cdot \vec{x}} = 1$$

Hence, the probability sum to 1.

2) The dimension of W is M x C, where C is the number of possible classification and M is the number of input features. The dimension of X is N x M where N is number of examples. The dimension of WX is C x N.

Qsr3:

1) We have: $P = [y = i] = \frac{e^{\vec{w_i} \cdot \vec{x}}}{\sum_j e^{\vec{w_j} \cdot \vec{x}}}$

Now, let $max(\vec{w_i}x) = WX_{max}$. We divide the $e^{WX_{max}}$ to the probability, result:

$$P = [y = i] = \frac{\frac{e^{\vec{w_i} \cdot \vec{x}}}{e^{WX_{max}}}}{\frac{\sum_j e^{\vec{w_j} \cdot \vec{x}}}{e^{WX_{max}}}} = \frac{e^{\vec{w_i}\vec{x} - WX_{max}}}{\sum_j e^{\vec{w_j}\vec{x} - WX_{max}}}$$

This is still the same ratio, which means it does not affected the predicted accuracy.

2) This might be an optimization over using W_X because when we calculate the cost, a large value x can create a wrong value for total cost. The exponent of a large number is very significant. Hence, when divide with the exponent of a large number, the calculation might go wrong. Therefore, using this function might be better.

[4]:
```
#code for Qrs4:
# MNIST images are 28 * 28
exSize = 28*28
# 10 digits
numClasses = 10
# Regularizer coefficient
X, Y = loadMNIST('data/train-images.idx3-ubyte', 'data/train-labels.idx1-ubyte')
sm = SoftmaxRegression(numClasses, exSize)
testX, testY = loadMNIST('data/t10k-images.idx3-ubyte', 'data/t10k-labels.
 ↪idx1-ubyte')
predictions = sm.predict(X)
(data_size, train_acc, test_acc) = learningCurve(sm, numClasses, exSize, X, Y,␣
 ↪testX, testY)
print(data_size)
```

```
60000
16
1
(784, 2)
(2,)
Training classifier on 2 points...
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  3.45388D+00    |proj g|=  4.50000D-01

At iterate    1    f=  7.54791D-01    |proj g|=  2.56845D-01

At iterate    2    f=  7.35419D-01    |proj g|=  2.49947D-01

At iterate    3    f=  6.93397D-01    |proj g|=  2.49922D-01

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840      4     59      2     0     0   2.499D-01   6.934D-01
  F =  0.69339723797348007

ABNORMAL_TERMINATION_IN_LNSRCH
Training accuracy 1, test accuracy 0.1139
2
(784, 4)
(4,)
Training classifier on 4 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds
```

```
At iterate    0    f=  1.40360D+00    |proj g|=  4.37973D-01

At iterate    1    f=  3.61703D-01    |proj g|=  1.24020D-01

At iterate    2    f=  3.46576D-01    |proj g|=  1.24999D-01

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840      3     51      2     0     0   1.250D-01   3.466D-01
  F =  0.34657562925118829

ABNORMAL_TERMINATION_IN_LNSRCH
Training accuracy 1, test accuracy 0.1437
3
(784, 8)
(8,)
Training classifier on 8 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.77493D+01    |proj g|=  2.09057D-01

At iterate    1    f=  1.53034D+01    |proj g|=  1.95530D-01

At iterate    2    f=  2.03323D+00    |proj g|=  1.25000D-01

At iterate    3    f=  3.48882D-01    |proj g|=  1.16768D-01

...

At iterate   12    f=  2.39576D-05    |proj g|=  2.38353D-05
```

```
At iterate   13    f=  1.20328D-05    |proj g|=  1.19597D-05

At iterate   14    f=  6.05174D-06    |proj g|=  6.00330D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     14     15      1     0     0   6.003D-06   6.052D-06
  F =   6.0517408407289446E-006

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.2692
4
(784, 15)
(15,)
Training classifier on 15 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  5.11229D+00    |proj g|=  1.76880D-01

At iterate    1    f=  3.35493D+00    |proj g|=  1.26666D-01

At iterate    2    f=  2.27328D+00    |proj g|=  1.55596D-01

At iterate    3    f=  1.24779D-01    |proj g|=  6.98760D-02

...
At iterate   12    f=  2.29767D-05    |proj g|=  1.01193D-05

At iterate   13    f=  1.22311D-05    |proj g|=  5.25885D-06
```

```
           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     13     14      1     0     0   5.259D-06   1.223D-05
  F =   1.2231068946561243E-005

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.3502
5
(784, 30)
(30,)
Training classifier on 30 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  5.69010D+00    |proj g|=  9.86926D-02

At iterate    1    f=  4.51942D+00    |proj g|=  9.86915D-02

At iterate    2    f=  1.44265D+00    |proj g|=  1.19933D-01

At iterate    3    f=  2.35187D-01    |proj g|=  5.25363D-02

At iterate    4    f=  1.11300D-01    |proj g|=  3.17428D-02
...
At iterate   12    f=  1.24037D-04    |proj g|=  2.94642D-05

At iterate   13    f=  6.69782D-05    |proj g|=  1.52075D-05

At iterate   14    f=  3.69401D-05    |proj g|=  8.24258D-06
```

```
           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     14     15      1     0     0   8.243D-06   3.694D-05
  F =   3.6940080143739086E-005

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.4004
6
(784, 59)
(59,)
Training classifier on 59 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  2.63305D+00    |proj g|=  8.05740D-02

At iterate    1    f=  1.77657D+00    |proj g|=  4.96180D-02

At iterate    2    f=  1.08953D+00    |proj g|=  6.26937D-02

At iterate    3    f=  6.01203D-01    |proj g|=  2.53500D-02

...
At iterate   13    f=  2.41959D-04    |proj g|=  3.23871D-05

At iterate   14    f=  1.29249D-04    |proj g|=  1.74530D-05

At iterate   15    f=  8.02618D-05    |proj g|=  3.50358D-05

At iterate   16    f=  3.05624D-05    |proj g|=  4.88372D-06
```

```
          * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

          * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     16     17      1     0     0   4.884D-06   3.056D-05
  F =   3.0562412316286158E-005

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.5819
7
(784, 118)
(118,)
Training classifier on 118 points...
RUNNING THE L-BFGS-B CODE

          * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.99178D+00    |proj g|=  4.68696D-02

At iterate    1    f=  1.44924D+00    |proj g|=  3.09946D-02

At iterate    2    f=  9.85280D-01    |proj g|=  6.24452D-02

At iterate    3    f=  6.45540D-01    |proj g|=  3.73655D-02

At iterate    4    f=  4.11109D-01    |proj g|=  2.28940D-02

...



At iterate   19    f=  5.57072D-05    |proj g|=  1.95780D-05

At iterate   20    f=  2.69800D-05    |proj g|=  4.44389D-06
```

```
           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     20     21      1     0     0   4.444D-06   2.698D-05
  F =   2.6980015531539907E-005

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.7018
8
(784, 235)
(235,)
Training classifier on 235 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.47343D+00    |proj g|=  3.22743D-02

At iterate    1    f=  1.18133D+00    |proj g|=  2.45324D-02

At iterate    2    f=  8.43887D-01    |proj g|=  2.67739D-02

At iterate    3    f=  5.21751D-01    |proj g|=  4.18095D-02

At iterate    4    f=  2.89418D-01    |proj g|=  1.23715D-02

At iterate    5    f=  2.26249D-01    |proj g|=  8.73027D-03

...

At iterate   20    f=  1.33032D-04    |proj g|=  1.12012D-05
```

```
At iterate   21    f=  9.21129D-05    |proj g|=  2.01157D-05

At iterate   22    f=  5.19530D-05    |proj g|=  5.48372D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     22     23      1     0     0   5.484D-06   5.195D-05
  F =   5.1952951981449447E-005

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.793
9
(784, 469)
(469,)
Training classifier on 469 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.04250D+00    |proj g|=  2.10990D-02

At iterate    1    f=  8.31139D-01    |proj g|=  1.53740D-02

At iterate    2    f=  6.07346D-01    |proj g|=  2.75524D-02

At iterate    3    f=  4.19349D-01    |proj g|=  1.48126D-02

At iterate    4    f=  2.75766D-01    |proj g|=  1.25437D-02

...
At iterate   27    f=  1.15651D-04    |proj g|=  1.31001D-05
```

```
At iterate   28    f=  9.89895D-05    |proj g|=  1.00974D-05

At iterate   29    f=  7.66305D-05    |proj g|=  6.20370D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     29     30      1     0     0   6.204D-06   7.663D-05
  F =   7.6630544212003617E-005

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.815
10
(784, 938)
(938,)
Training classifier on 938 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.84480D-01    |proj g|=  8.93479D-03

At iterate    1    f=  7.71158D-01    |proj g|=  7.56693D-03

At iterate    2    f=  6.27666D-01    |proj g|=  3.31254D-02

...

At iterate   36    f=  1.73244D-04    |proj g|=  3.16258D-05

At iterate   37    f=  1.24814D-04    |proj g|=  1.19187D-05

At iterate   38    f=  1.10069D-04    |proj g|=  7.07214D-06
```

```
           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     38     39      1     0     0   7.072D-06   1.101D-04
  F =   1.1006897943093058E-004

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.8428
11
(784, 1875)
(1875,)
Training classifier on 1875 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.50555D-01    |proj g|=  8.62377D-03

At iterate    1    f=  6.70055D-01    |proj g|=  4.48365D-03

At iterate    2    f=  6.25365D-01    |proj g|=  4.71615D-03

At iterate    3    f=  5.23282D-01    |proj g|=  1.02471D-02



...

At iterate   60    f=  1.88922D-04    |proj g|=  1.53696D-05

At iterate   61    f=  1.68409D-04    |proj g|=  1.06619D-05
```

```
At iterate   62    f=  1.35620D-04    |proj g|=  6.19396D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840     62     66      1     0     0   6.194D-06   1.356D-04
  F =   1.3562009626588901E-004

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.859
12
(784, 3750)
(3750,)
Training classifier on 3750 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  9.47158D-01    |proj g|=  8.23798D-03

At iterate    1    f=  9.05725D-01    |proj g|=  5.58973D-03

At iterate    2    f=  8.71385D-01    |proj g|=  5.17801D-03

...

At iterate    7    f=  5.51246D-01    |proj g|=  3.91548D-03

At iterate    8    f=  5.04666D-01    |proj g|=  8.69213D-03
 This problem is unconstrained.

At iterate    9    f=  4.65868D-01    |proj g|=  8.26349D-03
```

```
At iterate   10    f=  4.44192D-01    |proj g|=  3.42502D-03

...

At iterate  133    f=  1.78186D-04    |proj g|=  1.13796D-05

At iterate  134    f=  1.65219D-04    |proj g|=  7.48054D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840    134    144      1     0     0   7.481D-06   1.652D-04
  F =   1.6521861880642840E-004

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.8471
13
(784, 7500)
(7500,)
Training classifier on 7500 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

 This problem is unconstrained.

At iterate    0    f=  1.23260D+00    |proj g|=  4.40204D-03

At iterate    1    f=  1.19851D+00    |proj g|=  4.23543D-03

At iterate    2    f=  1.17325D+00    |proj g|=  3.22072D-03
```

```
At iterate    3    f=  1.08138D+00    |proj g|=  6.82248D-03

At iterate    4    f=  1.01066D+00    |proj g|=  4.20362D-03
...

At iterate  341    f=  2.99889D-04    |proj g|=  1.24309D-05

At iterate  342    f=  2.82909D-04    |proj g|=  9.70383D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840    342    362      1     0     0   9.704D-06   2.829D-04
  F =   2.8290915858524717E-004

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Training accuracy 1, test accuracy 0.8501
14
(784, 15000)
(15000,)
Training classifier on 15000 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.83205D+00    |proj g|=  3.84989D-03

At iterate    1    f=  1.79612D+00    |proj g|=  2.49122D-03

...
At iterate    7    f=  1.54689D+00    |proj g|=  1.98857D-03

At iterate    8    f=  1.51754D+00    |proj g|=  3.01389D-03
```

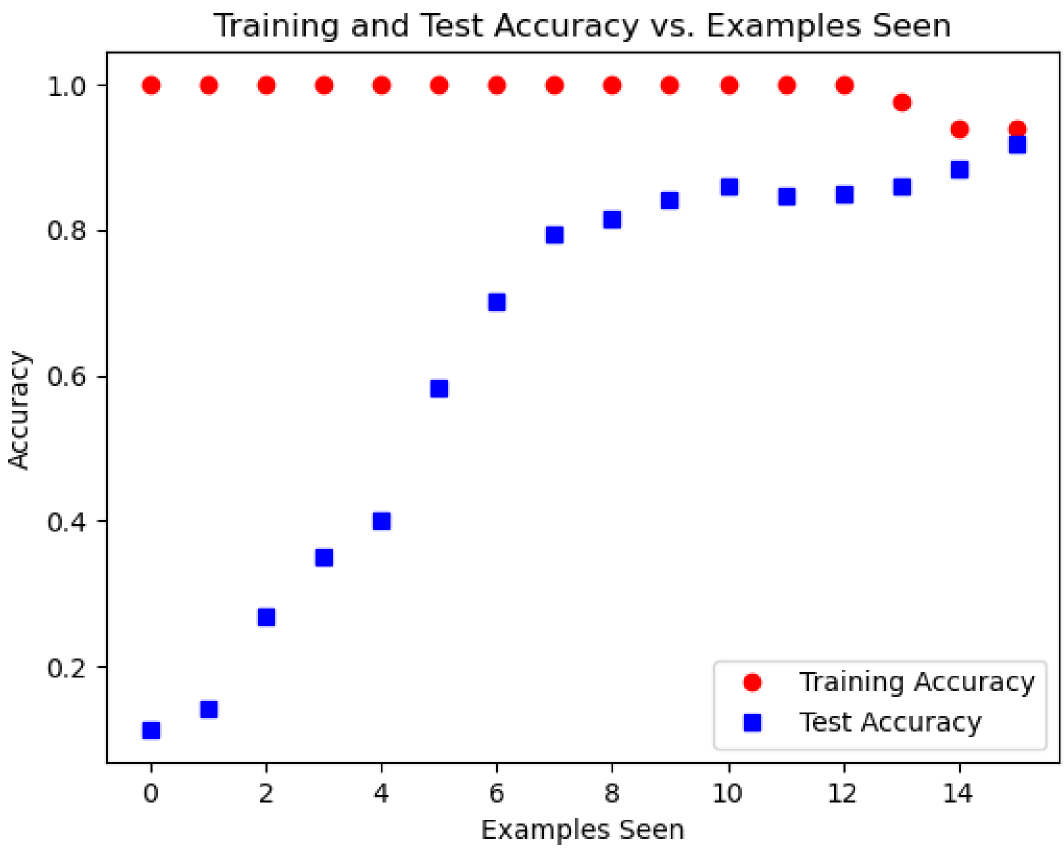
```
At iterate    9    f=  1.47373D+00    |proj g|=  4.94200D-03
 This problem is unconstrained.

At iterate   10    f=  1.42990D+00    |proj g|=  3.76280D-03

At iterate   11    f=  1.40955D+00    |proj g|=  2.11316D-03

...

At iterate  398    f=  9.84071D-02    |proj g|=  2.75509D-04

At iterate  399    f=  9.81845D-02    |proj g|=  4.41805D-04

At iterate  400    f=  9.79706D-02    |proj g|=  3.73465D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840    400    427      1     0     0   3.735D-04   9.797D-02
  F =   9.7970595972840205E-002

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT
Training accuracy 0.9766, test accuracy 0.861
15
(784, 30000)
(30000,)
Training classifier on 30000 points...
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds
```

```
At iterate    0    f=  1.00383D+00    |proj g|=  3.19584D-03

At iterate    1    f=  1.00016D+00    |proj g|=  2.96458D-03

At iterate    2    f=  9.96332D-01    |proj g|=  2.63122D-03

At iterate    3    f=  9.77399D-01    |proj g|=  3.05127D-03

At iterate    4    f=  9.70435D-01    |proj g|=  4.43586D-03
 This problem is unconstrained.

At iterate    5    f=  9.61265D-01    |proj g|=  2.73159D-03

At iterate    6    f=  9.50354D-01    |proj g|=  1.62143D-03

...

At iterate  398    f=  2.24659D-01    |proj g|=  1.30028D-03

At iterate  399    f=  2.24426D-01    |proj g|=  6.81391D-04

At iterate  400    f=  2.24219D-01    |proj g|=  3.18458D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840    400    434      1     0     0   3.185D-04   2.242D-01
  F =  0.22421920094254763

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT
Training accuracy 0.9388, test accuracy 0.8832
16
(784, 60000)
(60000,)
Training classifier on 60000 points...
RUNNING THE L-BFGS-B CODE
```

```
           * * *

Machine precision = 2.220D-16
 N =         7840     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  5.63363D-01    |proj g|=  7.75851D-03

At iterate    1    f=  5.43949D-01    |proj g|=  6.49524D-03
 This problem is unconstrained.

At iterate    2    f=  5.31404D-01    |proj g|=  4.74214D-03

At iterate    3    f=  5.02349D-01    |proj g|=  3.67615D-03

At iterate    4    f=  4.98815D-01    |proj g|=  1.23197D-02

At iterate    5    f=  4.84652D-01    |proj g|=  4.27436D-03

At iterate    6    f=  4.78740D-01    |proj g|=  1.73815D-03

...
At iterate  398    f=  2.26960D-01    |proj g|=  4.39289D-04

At iterate  399    f=  2.26853D-01    |proj g|=  2.77452D-04

At iterate  400    f=  2.26781D-01    |proj g|=  3.58551D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
 7840    400    424      1     0     0   3.586D-04   2.268D-01
  F =  0.22678116337494983

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT
Training accuracy 0.93805, test accuracy 0.9172
```

```
[2.000e+00 4.000e+00 8.000e+00 1.500e+01 3.000e+01 5.900e+01 1.180e+02
 2.350e+02 4.690e+02 9.380e+02 1.875e+03 3.750e+03 7.500e+03 1.500e+04
 3.000e+04 6.000e+04]
```

[6]:
```
plt.plot(train_acc, 'ro')
plt.plot(test_acc, 'bs')
plt.ylabel('Accuracy')
plt.xlabel('Examples Seen')
plt.legend(['Training Accuracy', 'Test Accuracy'])
plt.title('Training and Test Accuracy vs. Examples Seen')
plt.show()
```



QRS4: When the examples is low, the overfit occurs. When there are more examples, the overfit dereases. When it passes 14, the accuracy of training and testing are the same.

Part 3: NN

Qnn1.1: Submitted as code

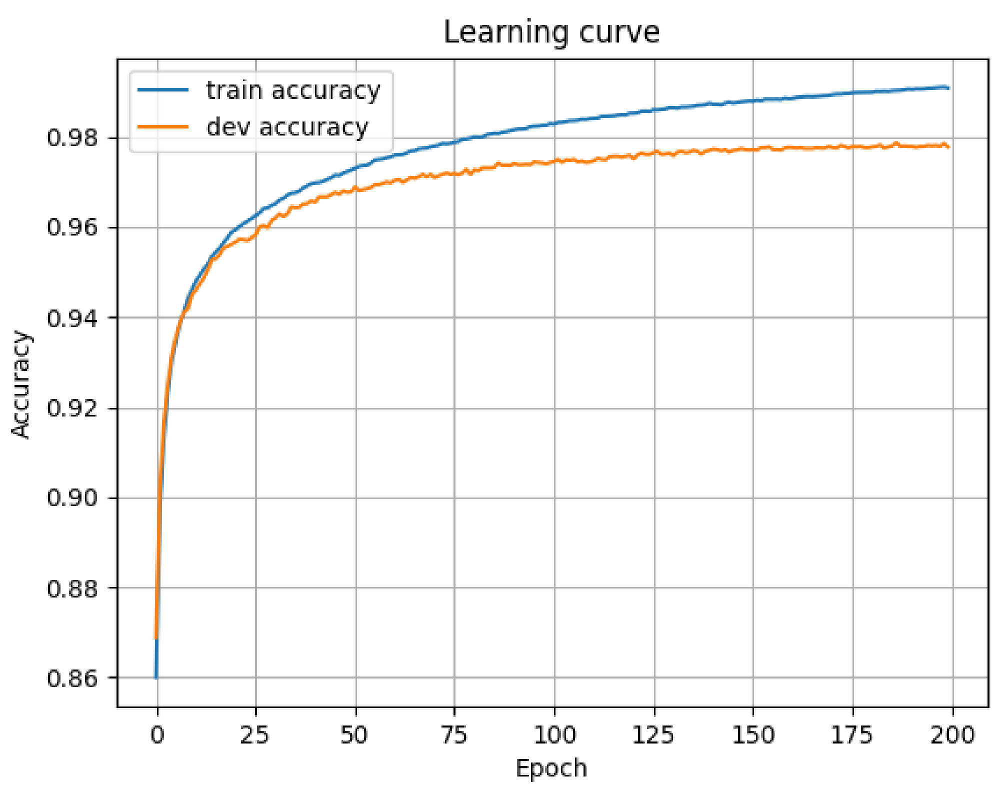
Qnn1.2: Submitted as code

Qnn1.3:

With the following settings, the final dev accuracy for 200/200 epoch is **0.979**.
Settings:

1. Number of layers: 2
2. Size of layers 1 and 2: [196, 196]
3. Activation function: ReLU
4. Loss function: Squared Loss
5. Learning rate: 0.01
6. batch size: 64
7. Max epochs: 200

The Training and Dev accuracy for NN with the settings above is:



Qnn1.4:
In some cases it may be appropriate to initialize the entries of the weight matrix as small random numbers rather than all zeros because:

- **We need to avoid dead neurons.** If all weights are initialized to zero, the neurons in a layer will all produce the same output, and during backpropagation, they will all receive the same gradients. For example, we have the first 2 layers of multi-layer perceptron. During forward propagation, each unit in the hidden layer gets a signal $\sum_i^N W_{i,j} \cdot x_i$. It does not matter what input we get, if the weights are the same, all the hidden units will be the same. This can lead to a situation where neurons become "dead" (i.e., they always output the same values) and do not contribute meaningfully to the learning process. Therefore, random initialization helps prevent this by ensuring that neurons start with different values, encouraging them to learn distinct features.

Extra Credit

NOTE 1: We implemented the new optimizer, the **AMSGrad**, in **direction 3**. The modified implementation is in:

1. **nn_extra.py**: modified update rules
2. **run_nn_extra.py**: run experiments

NOTE 2:: In the implementation, we first run a comparative study between AMSGrad and SGD on different training set sizes (e.g: 1/100 or 1/2 original size) to test its convergence ability with different training data available. The dev set size is kept the same. It can take about 5 minutes to train 2 models on all different sizes.

Qnn2.1:

- What I did was re-implementing the **update()** function in the NN class by adding the 1^{st} and 2^{nd} order momentum and using them by using the following update rule (**AMSGrad()**):

$$\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{v}_t &= \max(\hat{v}_{t-1}, v_t) \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} m_t
\end{aligned}$$

- Our experiments show that **AMSGrad requires less sample to converge compared to SGD**. The experiments are shown below:

Optimizer Type	1/100 train set	1/10 train set	1/5 train set	1/2 train set	full training set
SGD	0.7561	0.9184	0.9507	0.9619	0.9725
AMSGrad	0.8592	0.94	0.9568	0.9686	0.9781

- Our experiments below show that AMSGrad has **significantly faster convergence rate** and **better overall performance** compared to SGD with the same architecture, loss function, and optimal hyperparameters. The comparative study on the convergence rate between AMSGrad and SGD (in terms of training loss, training accuracy, and dev accuracy) is shown in the 2 plots below: