178 HW4

February 28, 2020

0.0.1 Zhiyuan Liu's Solution

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml

np.random.seed(0)
%matplotlib inline
```

1 Problem 1: Setting up the data and Linear Classifier

[5]:
```
# Data Loading
X = np.genfromtxt('data/X_train.txt', delimiter=None)
Y = np.genfromtxt('data/Y_train.txt', delimiter=None)
X,Y = ml.shuffleData(X,Y)

print(X.shape)
print(Y.shape)
```

```
(200000, 14)
(200000,)
```

1.1 Q1

[8]:
```
print('min = ', np.min(X, axis = 0))
print('max = ', np.max(X, axis = 0))
print('mean = ', np.mean(X, axis = 0))
print('var = ', np.var(X, axis = 0))
```

```
min =  [ 1.9350e+02  1.5250e+02  2.1425e+02  1.5250e+02  1.0000e+01  0.0000e+00
  0.0000e+00  0.0000e+00  8.7589e-01  0.0000e+00  0.0000e+00  0.0000e+00
  9.9049e-01 -9.9990e+02]
max =  [2.5300e+02 2.4900e+02 2.5250e+02 2.5250e+02 3.1048e+04 1.3630e+04
 9.2380e+03 1.2517e+02 1.9167e+01 1.3230e+01 6.6761e+01 7.3902e+01
 9.7504e+02 7.9720e+02]
mean =  [2.41601104e+02 2.27376571e+02 2.41554150e+02 2.32826768e+02
 3.08992337e+03 9.28259020e+02 1.38093830e+02 3.24857933e+00
 6.49865290e+00 2.09713912e+00 4.21766041e+00 2.69171845e+00
 1.02715905e+01 5.78148050e+00]
var =  [8.34991711e+01 9.26255931e+01 3.52863398e+01 9.76257317e+01
 1.56515138e+07 3.08176182e+06 4.43951746e+05 8.21948502e+00
 6.40504819e+00 4.36344047e+00 4.08637188e+00 2.19877847e+00
 4.04646245e+02 3.40652055e+03]
```

1.2 Q2

[23]:
```
Xtr, Xva, Ytr, Yva = ml.splitData(X, Y)
Xt, Yt = Xtr[:5000], Ytr[:5000] # subsample for efficiency (you can go higher)
XtS, params = ml.rescale(Xt) # Normalize the features
XvS, _ = ml.rescale(Xva, params) # Normalize the features

print(Xtr.shape)
print(Yva.shape)
print(XtS.shape)
print(XvS.shape)
```

```
(160000, 14)
(40000,)
(5000, 14)
(40000, 14)
```

[20]:
```
print('Training:')
print('min = ', np.min(XtS, axis = 0))
print('max = ', np.max(XtS, axis = 0))
print('mean = ', np.mean(XtS, axis = 0))
print('var = ', np.var(XtS, axis = 0))
```

```
Training:
min =  [ -4.44388405  -3.97613598  -4.50170696  -2.82054574  -0.79098017
  -0.52341183  -0.198389    -1.13489794  -2.11086267  -0.99993405
  -2.12660763  -1.89052581  -0.47756894 -16.92686545]
max =  [ 1.25457608  1.78066768  1.77730571  1.95354925  7.36935003  7.61180462
 13.74708956  8.91028803  4.27660806  4.52186562  6.98719275 13.47745335
 49.48790281 12.52637108]
mean =  [-3.20894422e-15  2.51438870e-15 -5.42410783e-14 -2.71676903e-14
 -6.75903777e-17  1.39488421e-16 -3.14159809e-16 -4.05797618e-16
 -9.99333949e-16  7.18980431e-16 -8.49142978e-16 -3.18949311e-15
  1.15223386e-15  4.02078371e-16]
var =  [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

[21]:
```
print('Validation:')
print('min = ', np.min(XvS, axis = 0))
print('max = ', np.max(XvS, axis = 0))
print('mean = ', np.mean(XvS, axis = 0))
print('var = ', np.var(XvS, axis = 0))
```

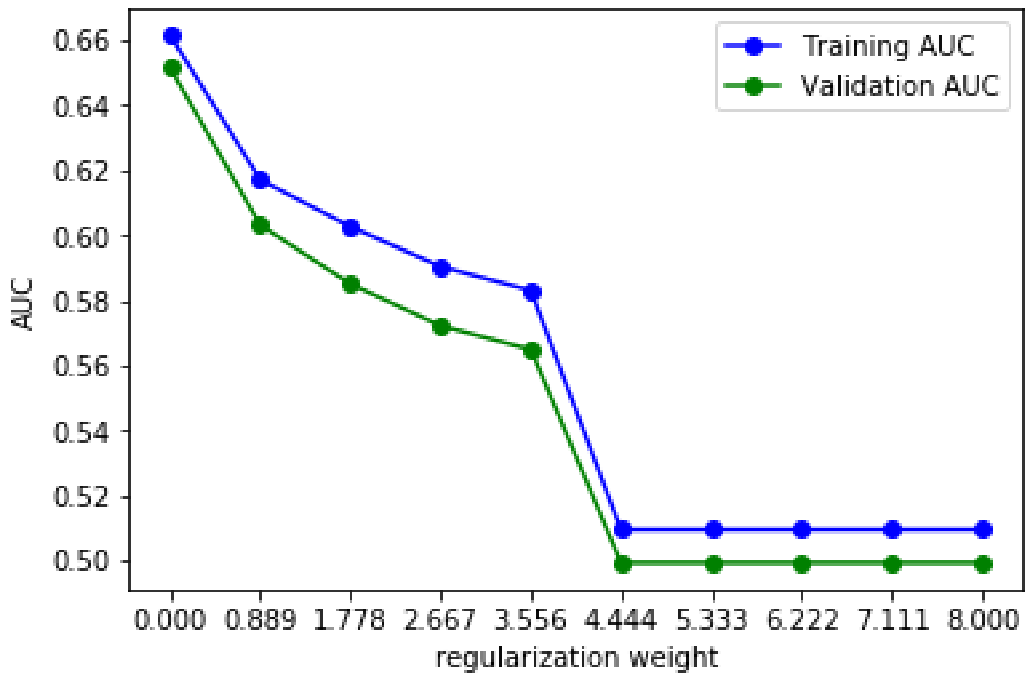
```
Validation:
min =  [ -5.32908174  -3.97613598  -4.55792167  -2.92268726  -0.7912431
  -0.52341183  -0.198389    -1.13489794  -2.22290153  -0.99993405
  -2.12660763  -1.89052581  -0.47939007 -16.92686545]
max =  [ 1.25457608  2.25600009  1.83522389  1.98827737  7.36935003  7.61180462
 13.74708956  8.91028803  4.78082507  5.4896644  13.55449008 38.16432873
 49.48790281 13.17796901]
mean =  [-0.01046472 -0.03135781 -0.03678486 -0.02251483  0.01750273  0.03520212
  0.01888531 -0.00737174  0.03571879  0.03331622  0.03494385  0.04670989
 -0.00817653  0.00486745]
var =  [1.03312196 1.05095611 1.04891268 1.02666459 1.08238603 1.1212983
 1.06447592 0.96514949 1.04667444 1.0686109  1.07520744 1.25634665
 0.92717743 1.00017888]
```

1.3 Q3

[32]:
```python
reg_value = np.linspace(0,8,10)

training_auc = np.zeros(10)
validation_auc = np.zeros(10)

for i, r in enumerate(reg_value):
    learner = ml.linearC.linearClassify()
    learner.train(XtS, Yt, reg=r, initStep=0.5, stopTol=1e-6, stopIter=100)
    training_auc[i] = learner.auc(XtS, Yt) # train AUC
    validation_auc[i] = learner.auc(XvS, Yva)

print(training_auc)
print(validation_auc)
```

```
[0.66151794 0.61708233 0.60264192 0.59046973 0.58310286 0.50953539
 0.50953539 0.50953539 0.50953539 0.50953539]
[0.65146811 0.60312207 0.58513866 0.57224215 0.56496986 0.49933836
 0.49933836 0.49933836 0.49933836 0.49933836]
```

[43]:
```python
plt.plot(reg_value, training_auc, color = 'blue', label = 'Training AUC',␣
 ↪marker = 'o')
plt.plot(reg_value, validation_auc, color = 'green', label = 'Validation AUC',␣
 ↪marker = 'o')

plt.xticks(reg_value)
plt.xlabel('regularization weight')
plt.ylabel('AUC')

plt.legend()
plt.show()
```



1.4 Q4

[45]:
```
degree = 2
XtrP = ml.transforms.fpoly(Xt, degree, False)

print(XtrP.shape)
```

```
(5000, 119)
```

[46]:
```
print('Number of features = ', XtrP.shape[1])
```

```
Number of features =  119
```

1.5 Q5

[59]:
```
XtrP,params = ml.transforms.rescale(XtrP)
XvaP,_ = ml.transforms.rescale(ml.transforms.fpoly(Xva,degree,False), params)
print(XvaP.shape)
```

```
(40000, 119)
```

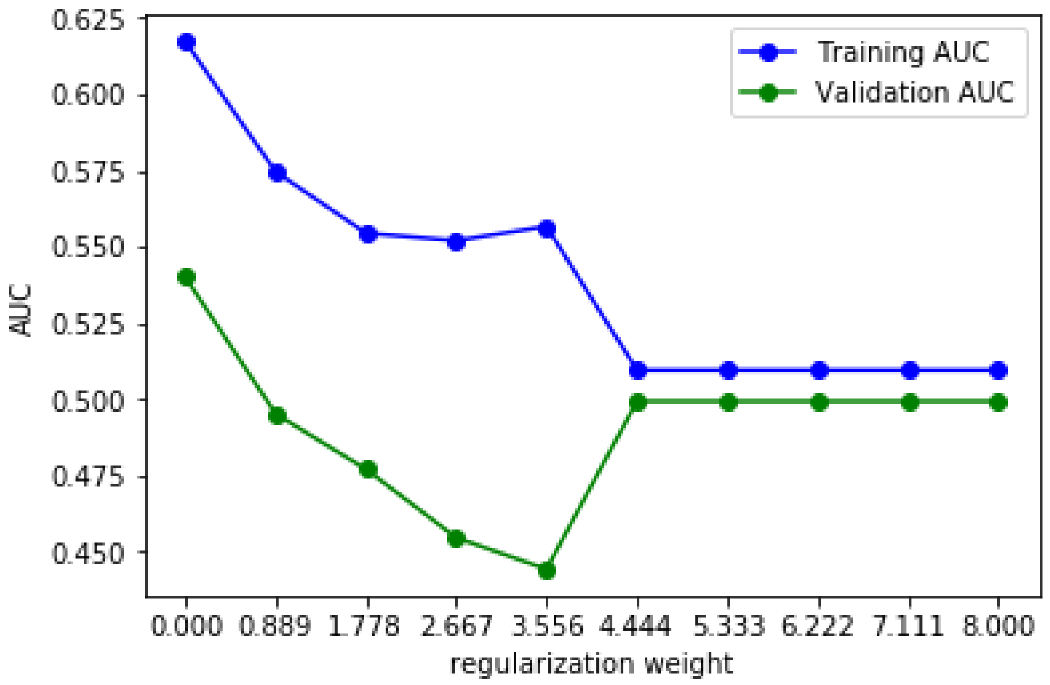
[55]:
```
reg_value = np.linspace(0,8,10)

training_auc = np.zeros(10)
```

```python
validation_auc = np.zeros(10)

for i, r in enumerate(reg_value):
    learner = ml.linearC.linearClassify()
    learner.train(XtrP, Yt, reg=r, initStep=0.5, stopTol=1e-6, stopIter=100)
    training_auc[i] = learner.auc(XtrP, Yt) # train AUC
    validation_auc[i] = learner.auc(XvaP, Yva)

print(training_auc)
print(validation_auc)
```

```
/Users/zhiyuanliu/Documents/Classes/CS 178/hw178/178HW4/mltools/base.py:96:
RuntimeWarning: divide by zero encountered in log
  return - np.mean( np.log( P[ np.arange(M), Y ] ) ) # evaluate
/Users/zhiyuanliu/Documents/Classes/CS 178/hw178/178HW4/mltools/linearC.py:134:
RuntimeWarning: invalid value encountered in double_scalars
  done = (it > stopIter) or ( (it>1) and (abs(Jsur[-1]-Jsur[-2])<stopTol) )
/Users/zhiyuanliu/Documents/Classes/CS 178/hw178/178HW4/mltools/linearC.py:84:
RuntimeWarning: invalid value encountered in true_divide
  prob /= prob + 1.0       # logistic transform (binary classification; C=1)
```

```
[0.61734116 0.57458994 0.55444619 0.55201911 0.55666876 0.50953539
 0.50953539 0.50953539 0.50953539 0.50953539]
[0.54007041 0.49515319 0.47705685 0.4545484  0.44423999 0.49933836
 0.49933836 0.49933836 0.49933836 0.49933836]
```

[56]:
```python
plt.plot(reg_value, training_auc, color = 'blue', label = 'Training AUC',␣
 ↪marker = 'o')
plt.plot(reg_value, validation_auc, color = 'green', label = 'Validation AUC',␣
 ↪marker = 'o')

plt.xticks(reg_value)
plt.xlabel('regularization weight')
plt.ylabel('AUC')

plt.legend()
plt.show()
```



2 Problem 2: Nearest Neighbor

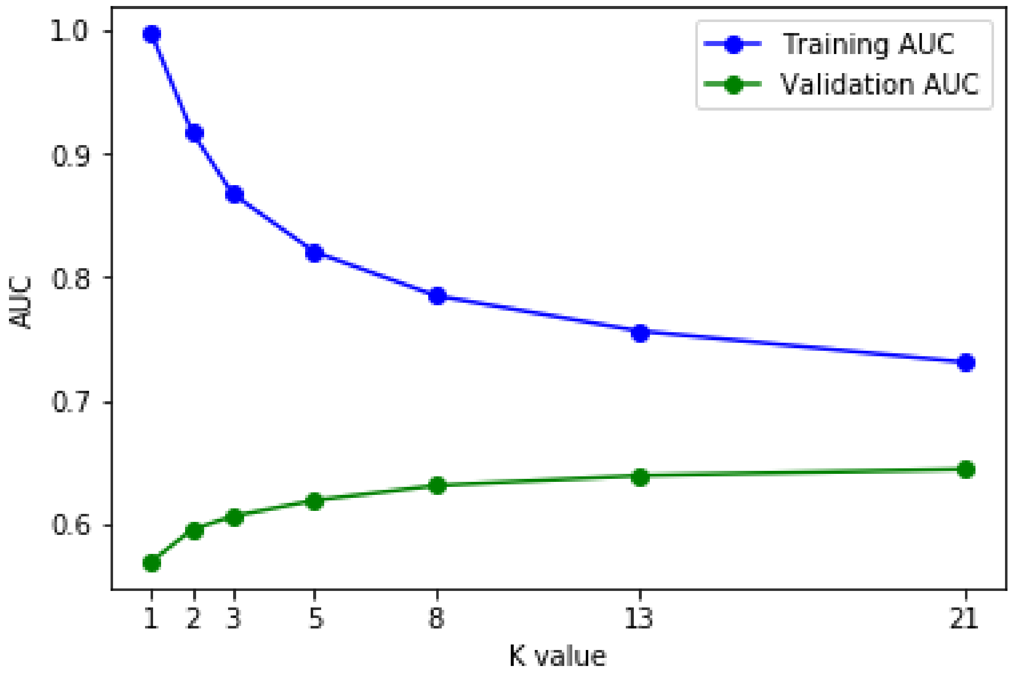
2.1 Q1

[61]:
```
K_num = np.array([1,2,3,5,8,13,21])
training_auc = np.zeros(7)
validation_auc = np.zeros(7)

learner = ml.knn.knnClassify()
for i, k in enumerate(K_num):
    learner.train(XtS, Yt, K=k, alpha=0.0)
    training_auc[i] = learner.auc(XtS, Yt) # train AUC
    validation_auc[i] = learner.auc(XvS, Yva)

print(training_auc)
print(validation_auc)
```

```
[0.99717752 0.9178101  0.86756235 0.82087064 0.78516166 0.75659512
 0.73135687]
[0.56998478 0.59581769 0.60664543 0.61946638 0.63150742 0.63960391
 0.64472525]
```

[63]:
```
plt.plot(K_num, training_auc, color = 'blue', label = 'Training AUC', marker =␣
 ↪'o')
```

```
plt.plot(K_num, validation_auc, color = 'green', label = 'Validation AUC',␣
 ↪marker = 'o')

plt.xticks(K_num)
plt.xlabel('K value')
plt.ylabel('AUC')

plt.legend()
plt.show()
```

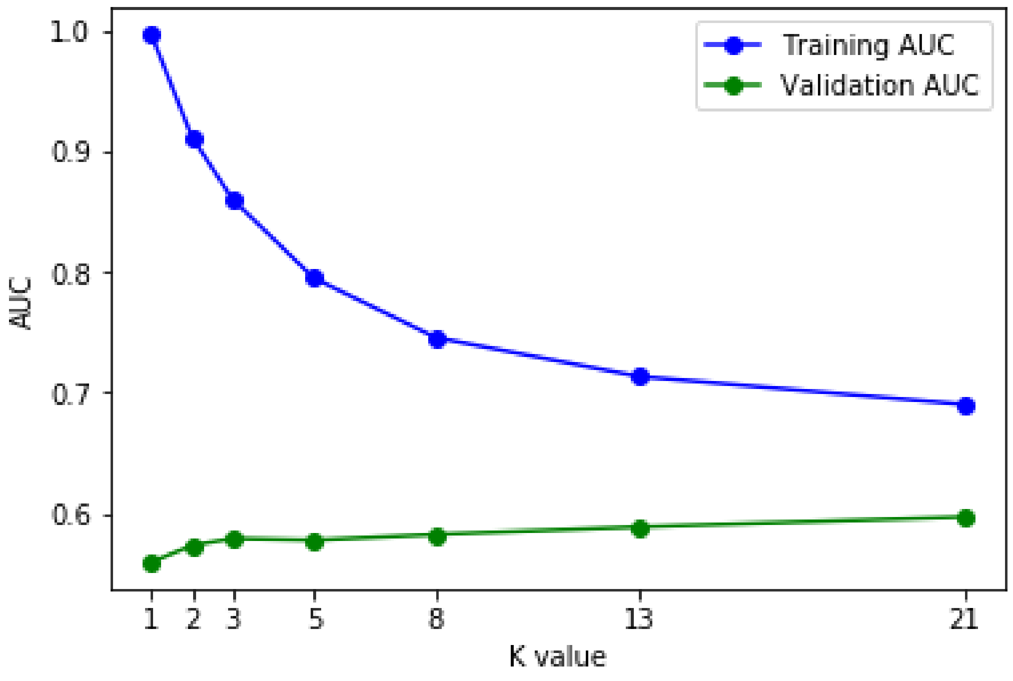


2.2 Q2

[64]:
```
K_num = np.array([1,2,3,5,8,13,21])
training_auc = np.zeros(7)
validation_auc = np.zeros(7)

learner = ml.knn.knnClassify()
for i, k in enumerate(K_num):
    learner.train(Xt, Yt, K=k, alpha=0.0)
    training_auc[i] = learner.auc(Xt, Yt) # train AUC
    validation_auc[i] = learner.auc(Xva[:5000], Yva[:5000])

print(training_auc)
print(validation_auc)
```

```
[0.99704076 0.91211207 0.86035488 0.79513526 0.74566345 0.71339296
 0.69019381]
[0.55932082 0.5731827  0.57892934 0.57757238 0.58228176 0.58856514
 0.59681686]
```

[65]:
```
plt.plot(K_num, training_auc, color = 'blue', label = 'Training AUC', marker =␣
 ↪'o')
plt.plot(K_num, validation_auc, color = 'green', label = 'Validation AUC',␣
 ↪marker = 'o')

plt.xticks(K_num)
plt.xlabel('K value')
plt.ylabel('AUC')

plt.legend()
plt.show()
```



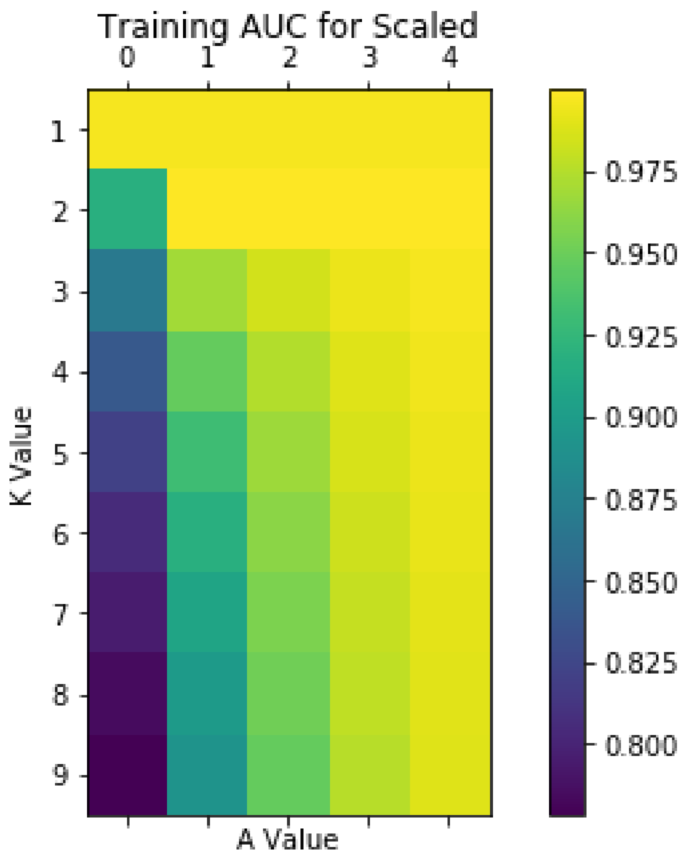
2.3 Q3

[69]:
```
K = range(1,10,1) # Or something else
A = range(0,5,1) # Or something else

tr_auc = np.zeros((len(K),len(A)))
va_auc = np.zeros((len(K),len(A)))
```

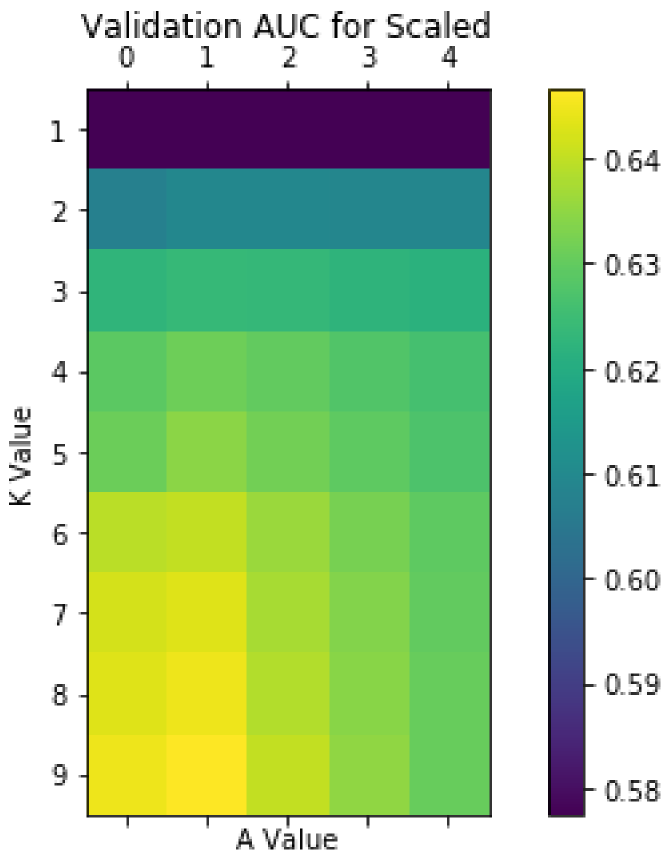
```
learner = ml.knn.knnClassify()
for i,k in enumerate(K):
    for j,a in enumerate(A):
        learner.train(XtS, Yt, K=k, alpha=a)
        tr_auc[i][j] = learner.auc(XtS, Yt) # train learner using k and a
        va_auc[i][j] = learner.auc(XvS[:5000], Yva[:5000])

print(tr_auc)
print(va_auc)
```

```
[[0.99717752 0.99717752 0.99717752 0.99717752 0.99717752]
 [0.9178101  0.99998266 0.99998266 0.99998266 0.99998266]
 [0.86756235 0.9693971  0.9845296  0.99323004 0.9968849 ]
 [0.84047808 0.9472222  0.97464259 0.98944692 0.99528401]
 [0.82087064 0.93133676 0.96761947 0.98622514 0.99380377]
 [0.80532956 0.91786831 0.96143651 0.98322804 0.99249089]
 [0.79516003 0.90770196 0.95638952 0.98067907 0.99127815]
 [0.78516166 0.89935676 0.95200241 0.97857114 0.99032299]
 [0.77838728 0.89248101 0.94795338 0.97658617 0.98939614]]
[[0.57773161 0.57773161 0.57773161 0.57751119 0.57751119]
 [0.60751944 0.60960162 0.60961928 0.60936746 0.60936826]
 [0.62280795 0.62381362 0.62358053 0.6225836  0.62164536]
 [0.62911594 0.63138446 0.62991493 0.62785068 0.62638035]
 [0.63107653 0.63457516 0.63198061 0.62957462 0.62727345]
 [0.63934583 0.64017542 0.63625744 0.63253286 0.62948792]
 [0.64227223 0.64334882 0.63765151 0.63376645 0.6300879 ]
 [0.64323892 0.64476358 0.63860063 0.63443538 0.63050314]
 [0.64497882 0.64663051 0.64025106 0.63510075 0.63064889]]
```

[77]:
```
# Now plot it
f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax = ax.matshow(tr_auc, interpolation='nearest')
f.colorbar(cax)
ax.set_xticklabels(['']+list(A))
ax.set_yticklabels(['']+list(K))
ax.set_title('Training AUC for Scaled')
ax.set_xlabel('A Value')
ax.set_ylabel('K Value')
plt.show()
```



[75]:
```
# Now plot it
f, ax = plt.subplots(1, 1, figsize=(8, 5))
cax = ax.matshow(va_auc, interpolation='nearest')
f.colorbar(cax)
ax.set_xticklabels(['']+list(A))
ax.set_yticklabels(['']+list(K))
ax.set_title('Validation AUC for Scaled')
ax.set_xlabel('A Value')
ax.set_ylabel('K Value')
plt.show()
```



k = 9, a = 1

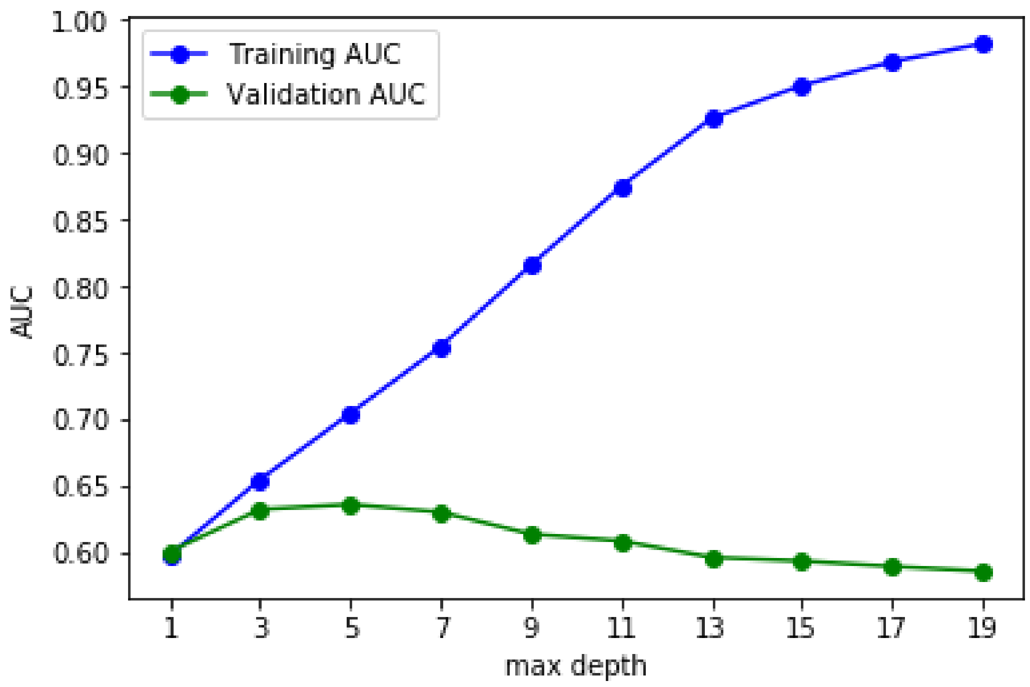
3 Problem 3: Decision Trees

3.1 Q1

[81]:
```
max_depth = np.array(range(1,20,2))

training_auc = np.zeros(max_depth.shape[0])
validation_auc = np.zeros(max_depth.shape[0])

learner = ml.dtree.treeClassify()
for i, d in enumerate(max_depth):
    learner.train(XtS, Yt, maxDepth=d, minParent=2, minLeaf=1)
    training_auc[i] = learner.auc(XtS, Yt) # train AUC
    validation_auc[i] = learner.auc(XvS, Yva)

print(training_auc)
print(validation_auc)
```

```
[0.59802421 0.65497753 0.70465519 0.75494991 0.81603585 0.87513063
 0.92641218 0.95107768 0.96863956 0.98242034]
[0.60093708 0.6325335  0.63628392 0.63045846 0.61395721 0.60884357
 0.59644889 0.59372132 0.58965459 0.58608085]
```

[82]:
```
plt.plot(max_depth, training_auc, color = 'blue', label = 'Training AUC',␣
 ↪marker = 'o')
plt.plot(max_depth, validation_auc, color = 'green', label = 'Validation AUC',␣
 ↪marker = 'o')

plt.xticks(max_depth)
plt.xlabel('max depth')
plt.ylabel('AUC')

plt.legend()
plt.show()
```



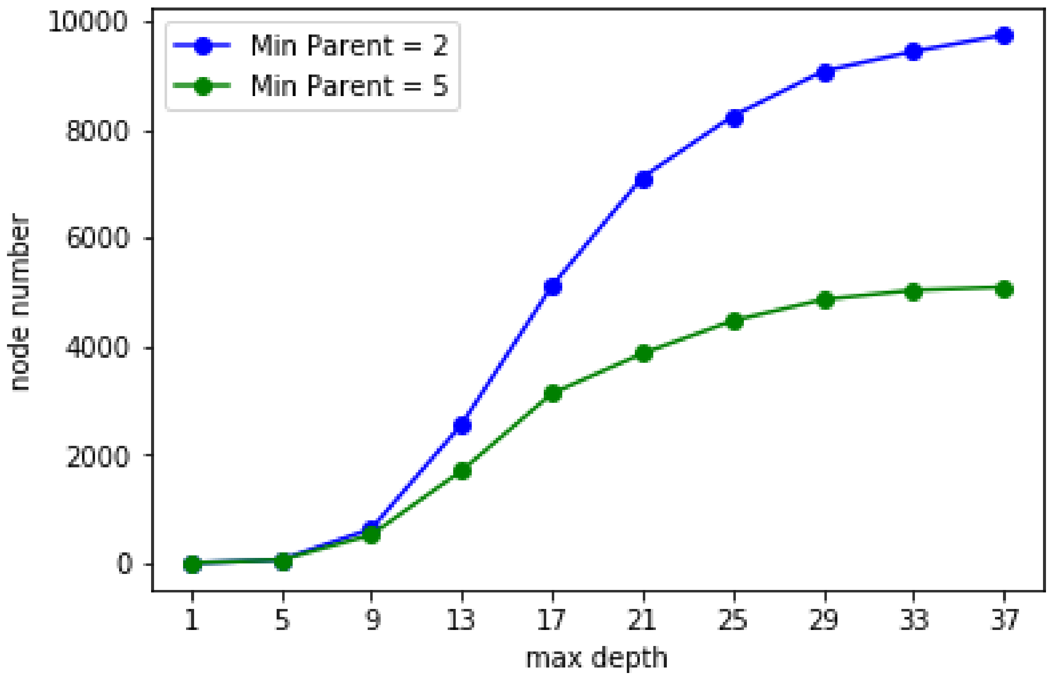
3.2 Q2

[89]:
```
## Plot the number of nodes in the tree as maxDepth is varied:
max_depth = np.array(range(1,40,4))
node_num = np.zeros(maxd.shape[0])

learner = ml.dtree.treeClassify()
```

```
for i, d in enumerate(max_depth):
    learner.train(XtS, Yt, maxDepth=d, minParent=2)
    node_num[i] = learner.sz

node_num1 = np.zeros(maxd.shape[0])

for i, d in enumerate(max_depth):
    learner.train(XtS, Yt, maxDepth=d, minParent=5)
    node_num1[i] = learner.sz

print(node_num)
print(node_num1)
```

```
[3.000e+00 6.300e+01 6.350e+02 2.563e+03 5.115e+03 7.115e+03 8.245e+03
 9.073e+03 9.435e+03 9.739e+03]
[3.000e+00 6.300e+01 5.230e+02 1.709e+03 3.129e+03 3.867e+03 4.471e+03
 4.863e+03 5.035e+03 5.095e+03]
```

[90]:
```
plt.plot(max_depth, node_num, color = 'blue', label = 'Min Parent = 2', marker␣
 ↪= 'o')
plt.plot(max_depth, node_num1, color = 'green', label = 'Min Parent = 5',␣
 ↪marker = 'o')

plt.xticks(max_depth)
plt.xlabel('max depth')
plt.ylabel('node number')

plt.legend()
plt.show()
```



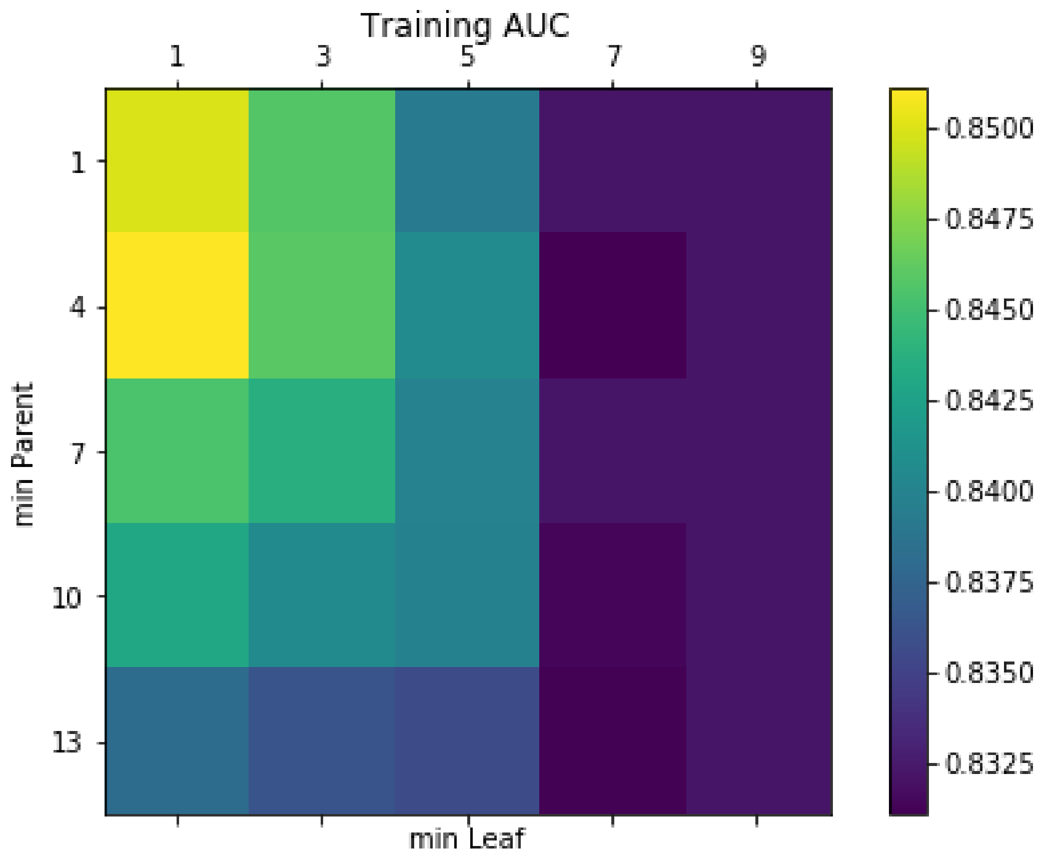
3.3 Q3

```
[103]: P = range(1,16,3) # Or something else
L = range(1,11,2) # Or something else

tr_auc = np.zeros((len(P),len(L)))
va_auc = np.zeros((len(P),len(L)))

learner = ml.dtree.treeClassify()
for i,p in enumerate(P):
    for j,l in enumerate(L):
        learner.train(XtS, Yt, maxDepth=10, minParent=p, minLeaf=l)
        tr_auc[i][j] = learner.auc(XtS, Yt) # train learner using k and a
        va_auc[i][j] = learner.auc(XvS[:5000], Yva[:5000])

print(tr_auc)
print(va_auc)
```

```
[[0.85099004 0.84650165 0.83969453 0.83198277 0.83195358]
 [0.85222622 0.84668988 0.84112072 0.83074774 0.83194527]
 [0.84610155 0.8442774  0.84028251 0.83197464 0.83195199]
 [0.8436406  0.84100449 0.84020192 0.83109838 0.83194792]
 [0.83830063 0.83630699 0.83573529 0.83084424 0.83195871]]
[[0.6223283  0.61911708 0.62102201 0.62442011 0.61879435]
```

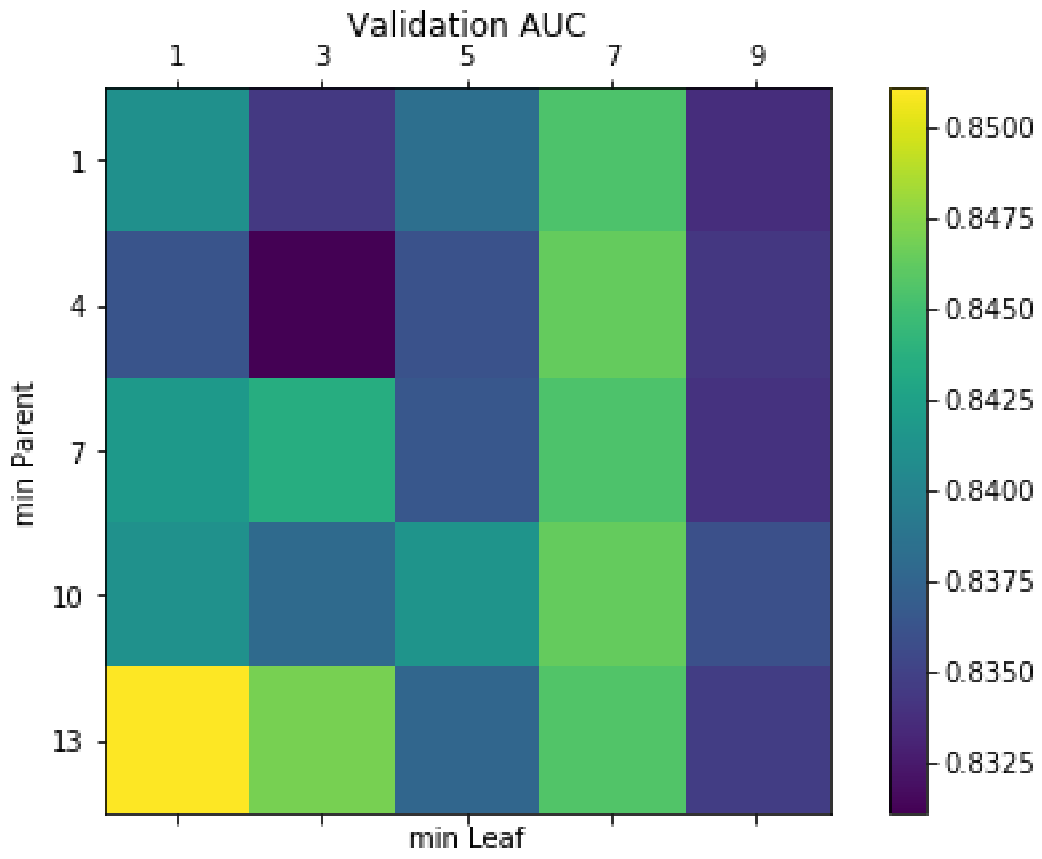
```
 [0.62000902 0.61753533 0.61994052 0.6248333  0.61906713]
 [0.62266406 0.6234866  0.62012945 0.62439398 0.61893734]
 [0.62234525 0.62083638 0.62246434 0.62482741 0.61987335]
 [0.62707471 0.62513454 0.62069803 0.62449469 0.61927899]]
```

[106]:
```python
f, ax = plt.subplots(1, 1, figsize=(8, 5))

cax = ax.matshow(tr_auc, interpolation='nearest')
f.colorbar(cax1)
ax.set_xticklabels(['']+list(L))
ax.set_yticklabels(['']+list(P))
ax.set_xlabel('min Leaf')
ax.set_ylabel('min Parent')
ax.set_title('Training AUC')

plt.show()
```



[107]:
```python
f, ax = plt.subplots(1, 1, figsize=(8, 5))

cax = ax.matshow(va_auc, interpolation='nearest')
```

```
f.colorbar(cax1)
ax.set_xticklabels(['']+list(L))
ax.set_yticklabels(['']+list(P))
ax.set_xlabel('min Leaf')
ax.set_ylabel('min Parent')
ax.set_title('Validation AUC')

plt.show()
```



min parent = 13, min leaf = 1

4 Problem 4: Neural Networks and Conclusion

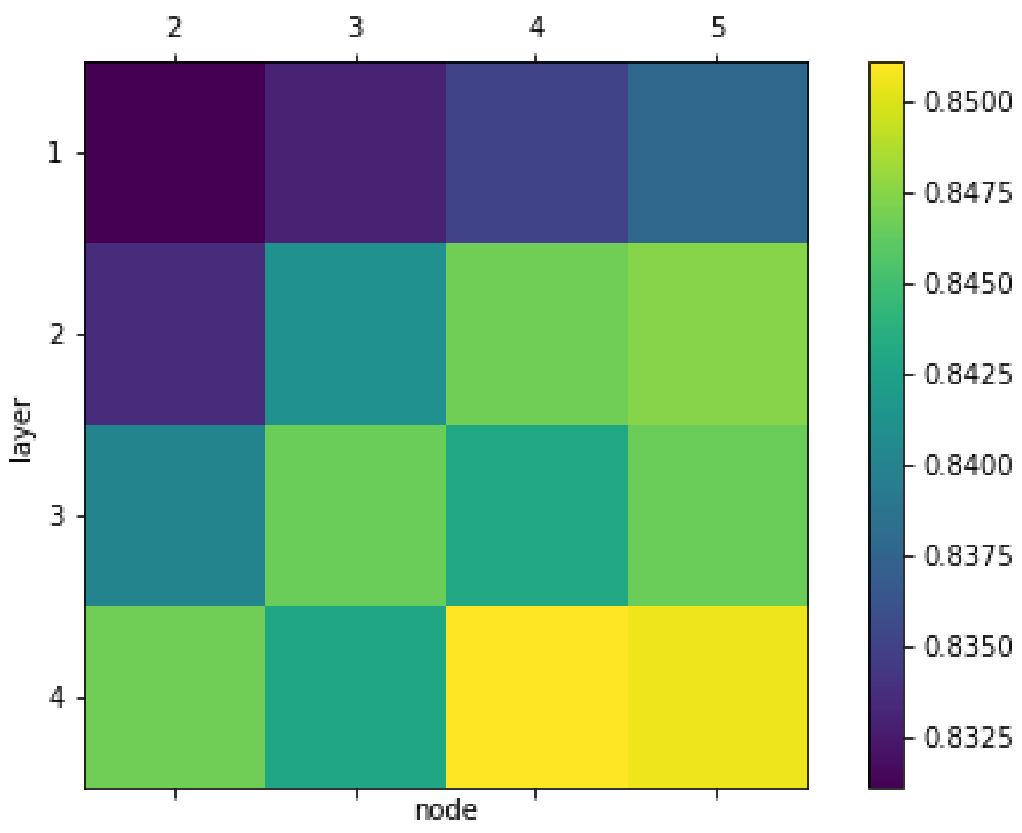
4.1 Q1

[108]:
```
L = range(1,5,1) # Or something else
N = range(2,6,1) # Or something else

tr_auc = np.zeros((len(L),len(N)))
va_auc = np.zeros((len(L),len(N)))
```

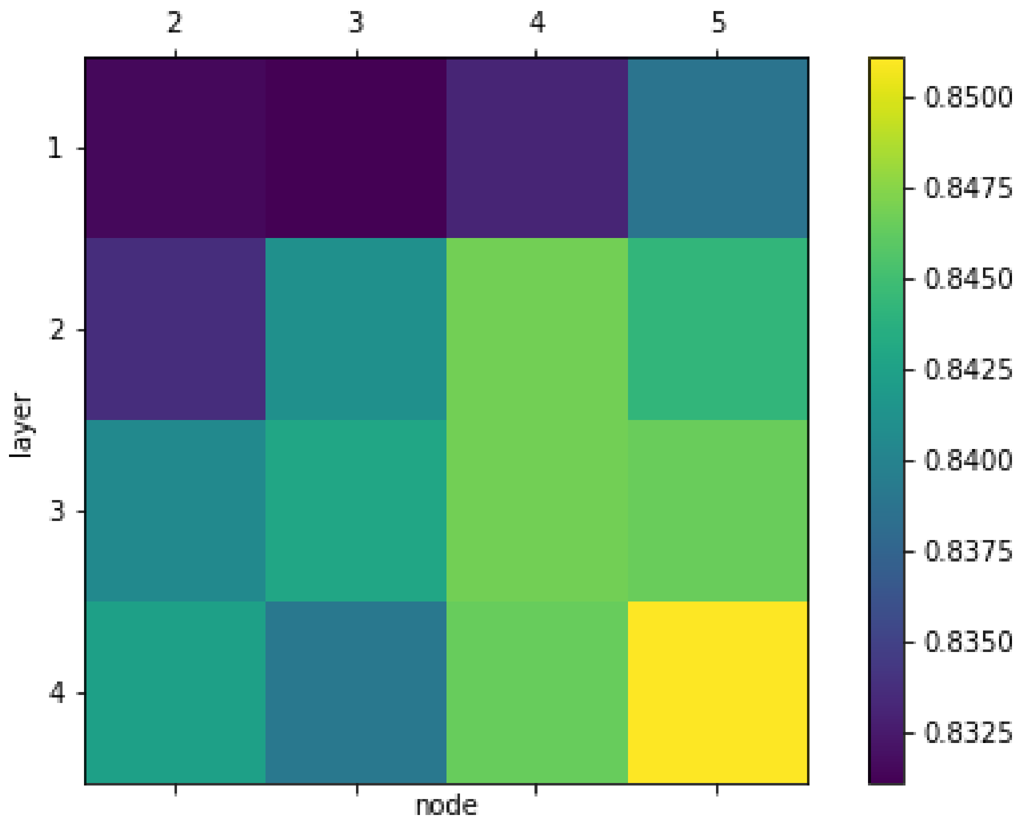
```
nn = ml.nnet.nnetClassify()
for i,l in enumerate(L):
    for j,n in enumerate(N):
        nn.init_weights([XtS.shape[1], l*n, 2], 'random', XtS, Yt) # as many␣
↪layers nodes you want
        nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
        tr_auc[i][j] = nn.auc(XtS, Yt) # train learner using k and a
        va_auc[i][j] = nn.auc(XvS[:5000], Yva[:5000])

print(tr_auc)
print(va_auc)
```

```
it 1 : Jsur = 0.4321604749706381, J01 = 0.3454
it 2 : Jsur = 0.42612752933713643, J01 = 0.3454
it 4 : Jsur = 0.42592009333627046, J01 = 0.3296
it 8 : Jsur = 0.4263831239356833, J01 = 0.3368
it 16 : Jsur = 0.4267765674944996, J01 = 0.338
it 32 : Jsur = 0.4273177167834803, J01 = 0.3454
it 64 : Jsur = 0.42775216846794284, J01 = 0.3454
it 128 : Jsur = 0.42820687644452726, J01 = 0.3454
it 256 : Jsur = 0.42852511530903203, J01 = 0.3454
it 1 : Jsur = 0.4307475631414062, J01 = 0.3454
it 2 : Jsur = 0.42430830054724983, J01 = 0.3454
it 4 : Jsur = 0.4219517541867341, J01 = 0.322
it 8 : Jsur = 0.4223957821420998, J01 = 0.3244
it 16 : Jsur = 0.42292342977437947, J01 = 0.3264
it 32 : Jsur = 0.42320276924057926, J01 = 0.3262
it 64 : Jsur = 0.4237229929132943, J01 = 0.3266
it 128 : Jsur = 0.4243747440654962, J01 = 0.3284
it 256 : Jsur = 0.42501063864346017, J01 = 0.33
it 1 : Jsur = 0.43013860525847786, J01 = 0.3454
it 2 : Jsur = 0.42272367750468565, J01 = 0.3454
it 4 : Jsur = 0.41903620452733115, J01 = 0.3166
it 8 : Jsur = 0.4192627290163874, J01 = 0.3246
it 16 : Jsur = 0.4200343525753111, J01 = 0.3246
it 32 : Jsur = 0.4199317119293608, J01 = 0.3234
it 64 : Jsur = 0.4197834196405704, J01 = 0.3218
it 128 : Jsur = 0.4198355084897309, J01 = 0.3216
it 256 : Jsur = 0.4199091668369239, J01 = 0.3228
it 1 : Jsur = 0.43076055618371273, J01 = 0.3454
it 2 : Jsur = 0.4230513677579499, J01 = 0.3288
it 4 : Jsur = 0.41818798433151366, J01 = 0.3108
it 8 : Jsur = 0.4167313242279575, J01 = 0.3152
it 16 : Jsur = 0.4154618569977592, J01 = 0.3134
it 32 : Jsur = 0.41407262530711925, J01 = 0.3116
it 64 : Jsur = 0.41317872965745756, J01 = 0.308
it 128 : Jsur = 0.41419238355198285, J01 = 0.3088
it 256 : Jsur = 0.41492469919353697, J01 = 0.3128
it 1 : Jsur = 0.4299124808483749, J01 = 0.3454
it 2 : Jsur = 0.423296512465319, J01 = 0.3454
it 4 : Jsur = 0.4199092907943008, J01 = 0.3156
it 8 : Jsur = 0.41952083693374986, J01 = 0.324
it 16 : Jsur = 0.41881486765086373, J01 = 0.3212
it 32 : Jsur = 0.41817268385180917, J01 = 0.3172
it 64 : Jsur = 0.41773638680099173, J01 = 0.3158
it 128 : Jsur = 0.41737409456956664, J01 = 0.313
it 1 : Jsur = 0.4290205600140115, J01 = 0.3454
it 2 : Jsur = 0.4213892682189672, J01 = 0.3238
it 4 : Jsur = 0.4169599832336241, J01 = 0.3126
it 8 : Jsur = 0.41590344762084297, J01 = 0.3154
it 16 : Jsur = 0.4131385489936268, J01 = 0.3122
it 32 : Jsur = 0.4097448456236623, J01 = 0.31
it 64 : Jsur = 0.4085060856806732, J01 = 0.3064
it 128 : Jsur = 0.4089147547842748, J01 = 0.3068
it 256 : Jsur = 0.40943704760760763, J01 = 0.3116
it 1 : Jsur = 0.42960903519435867, J01 = 0.3454
it 2 : Jsur = 0.4217363294937411, J01 = 0.3248
it 4 : Jsur = 0.4168051356717824, J01 = 0.31
it 8 : Jsur = 0.41526246709663756, J01 = 0.3132
it 16 : Jsur = 0.41251994797409436, J01 = 0.307
it 32 : Jsur = 0.40866450226224504, J01 = 0.301
it 64 : Jsur = 0.4046881983760139, J01 = 0.3024
it 128 : Jsur = 0.4020803965278983, J01 = 0.2988
it 256 : Jsur = 0.4001914516815521, J01 = 0.297
it 1 : Jsur = 0.429032601832447, J01 = 0.3454
it 2 : Jsur = 0.42221217911985826, J01 = 0.3264
it 4 : Jsur = 0.4168947449480689, J01 = 0.3102
it 8 : Jsur = 0.41524153596847346, J01 = 0.3118
it 16 : Jsur = 0.4135576409827646, J01 = 0.309
it 32 : Jsur = 0.41047590590486255, J01 = 0.303
it 64 : Jsur = 0.4061548186045488, J01 = 0.2982
it 128 : Jsur = 0.4024562896609561, J01 = 0.2968
it 256 : Jsur = 0.3992907208562406, J01 = 0.2936
it 1 : Jsur = 0.42945112610405195, J01 = 0.3454
it 2 : Jsur = 0.42157849910185724, J01 = 0.3258
it 4 : Jsur = 0.41685747850422405, J01 = 0.3108
it 8 : Jsur = 0.4153801992343824, J01 = 0.315
it 16 : Jsur = 0.4128807669022357, J01 = 0.3102
it 32 : Jsur = 0.40997367775211685, J01 = 0.3088
it 64 : Jsur = 0.40956682256942945, J01 = 0.3102
it 128 : Jsur = 0.410896758278292, J01 = 0.3136
it 256 : Jsur = 0.4114630064984376, J01 = 0.312
it 1 : Jsur = 0.4287058863708692, J01 = 0.3454
it 2 : Jsur = 0.4225349064023047, J01 = 0.3272
it 4 : Jsur = 0.41727393907028554, J01 = 0.31
it 8 : Jsur = 0.41540047652407286, J01 = 0.313
it 16 : Jsur = 0.4134350496908952, J01 = 0.3108
it 32 : Jsur = 0.41048126997818507, J01 = 0.307
it 64 : Jsur = 0.4079779998949332, J01 = 0.3022
it 128 : Jsur = 0.4051533525198143, J01 = 0.3004
it 256 : Jsur = 0.40055563842542047, J01 = 0.2972
it 1 : Jsur = 0.43012299980520263, J01 = 0.3454
it 2 : Jsur = 0.4215108981906131, J01 = 0.3256
it 4 : Jsur = 0.4158925628908766, J01 = 0.309
it 8 : Jsur = 0.4147444086517229, J01 = 0.3088
it 16 : Jsur = 0.41284973404113406, J01 = 0.3062
it 32 : Jsur = 0.4098315787017069, J01 = 0.303
it 64 : Jsur = 0.4068888904658235, J01 = 0.3006
it 128 : Jsur = 0.4054234335229372, J01 = 0.2998
it 256 : Jsur = 0.4037844748197587, J01 = 0.2972
it 1 : Jsur = 0.4290581753048704, J01 = 0.3454
it 2 : Jsur = 0.4223595863838358, J01 = 0.3276
it 4 : Jsur = 0.41665084511898415, J01 = 0.31
it 8 : Jsur = 0.4145451380933673, J01 = 0.3132
it 16 : Jsur = 0.4127079093284079, J01 = 0.3082
it 32 : Jsur = 0.4096538174105677, J01 = 0.3034
it 64 : Jsur = 0.4064210849250925, J01 = 0.3008
it 128 : Jsur = 0.4035745363328767, J01 = 0.3012
it 256 : Jsur = 0.40031365901052757, J01 = 0.3002
it 1 : Jsur = 0.43136817830882035, J01 = 0.3454
it 2 : Jsur = 0.4229477513379143, J01 = 0.3284
it 4 : Jsur = 0.41625589944121055, J01 = 0.3084
it 8 : Jsur = 0.4144531613145371, J01 = 0.3092
it 16 : Jsur = 0.4122357926602912, J01 = 0.3084
it 32 : Jsur = 0.40831142026931083, J01 = 0.3022
it 64 : Jsur = 0.4053196284834055, J01 = 0.2974
it 128 : Jsur = 0.4021243959754152, J01 = 0.2952
it 256 : Jsur = 0.3992903824577717, J01 = 0.293
it 1 : Jsur = 0.4297249764875419, J01 = 0.3454
it 2 : Jsur = 0.42291715420790743, J01 = 0.3294
it 4 : Jsur = 0.4165515074165744, J01 = 0.3084
it 8 : Jsur = 0.4141186070973521, J01 = 0.3134
it 16 : Jsur = 0.41211117304284955, J01 = 0.3092
it 32 : Jsur = 0.40908605870549675, J01 = 0.3036
it 64 : Jsur = 0.4068585426712193, J01 = 0.2976
it 128 : Jsur = 0.4050502831777462, J01 = 0.2978
it 256 : Jsur = 0.40372471799692905, J01 = 0.2984
it 1 : Jsur = 0.4291941500281872, J01 = 0.3454
it 2 : Jsur = 0.4227894265618673, J01 = 0.327
it 4 : Jsur = 0.4172577799486331, J01 = 0.3108
it 8 : Jsur = 0.41482534593028575, J01 = 0.315
it 16 : Jsur = 0.4124046564386779, J01 = 0.3094
it 32 : Jsur = 0.4092856328868244, J01 = 0.3054
it 64 : Jsur = 0.4060123918479182, J01 = 0.303
it 128 : Jsur = 0.40151550338399844, J01 = 0.2974
it 256 : Jsur = 0.396566584770366, J01 = 0.2928
it 1 : Jsur = 0.42829726173222443, J01 = 0.3456
it 2 : Jsur = 0.42234942992536534, J01 = 0.3238
it 4 : Jsur = 0.4174027848117599, J01 = 0.3132
it 8 : Jsur = 0.41565889148053886, J01 = 0.3156
it 16 : Jsur = 0.41304566228477435, J01 = 0.312
it 32 : Jsur = 0.40850339529430024, J01 = 0.306
it 64 : Jsur = 0.40482904669914976, J01 = 0.3002
it 128 : Jsur = 0.4016288333702511, J01 = 0.2984
it 256 : Jsur = 0.39756319424223846, J01 = 0.2956
[[0.65440451 0.65899436 0.66392645 0.67039212]
 [0.66035704 0.67844612 0.69159435 0.69308326]
 [0.67581152 0.69114959 0.68262553 0.69118585]
 [0.6915556  0.68245799 0.70177927 0.7004848 ]]
[[0.65386077 0.65334419 0.65563091 0.66190099]
 [0.65621876 0.6644349  0.67077188 0.66794486]
 [0.66383313 0.66647568 0.67074886 0.67044022]
 [0.6659303  0.66221927 0.67033728 0.67550713]]
```

[111]:
```python
f, ax = plt.subplots(1, 1, figsize=(8, 5))

cax = ax.matshow(tr_auc, interpolation='nearest')
f.colorbar(cax1)
ax.set_xticklabels(['']+list(N))
ax.set_yticklabels(['']+list(L))
ax.set_xlabel('node')
ax.set_ylabel('layer')

plt.show()
```



[113]:
```
f, ax = plt.subplots(1, 1, figsize=(8, 5))

cax = ax.matshow(va_auc, interpolation='nearest')
f.colorbar(cax1)
ax.set_xticklabels(['']+list(N))
ax.set_yticklabels(['']+list(L))
ax.set_xlabel('node')
ax.set_ylabel('layer')

plt.show()
```



node = 4, layer = 2

4.2 Q2

I did this homework alone.