Setting up the data (10 points)

The following is the snippet of code to load the datasets, and split it into train and validation data:

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import mltools as ml
import warnings
# warnings.filterwarnings("ignore")
np.random.seed(0)

# 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)

def print_info(X, name):
    for i in range(X.shape[1]):
        print(i + 1)
        print(name + " min is:", np.min(X[:, i]), name + " max is:", np.max(X[:, i])
        print(name + " mean is:",np.mean(X[:,i]), name + " variance is:",np.var(X[:,
```

In [2]:

```
# 1.1
print_info(X, '')
```

```
1
 min is: 193.5  max is: 253.0
 mean is: 241.6011037  variance is: 83.4991711498463
2
 min is: 152.5  max is: 249.0
 mean is: 227.37657130000002  variance is: 92.62559312501628
3
 min is: 214.25  max is: 252.5
 mean is: 241.55415049999996  variance is: 35.28633980334975
4
 min is: 152.5  max is: 252.5
 mean is: 232.82676815000005  variance is: 97.6257317486456
5
 min is: 10.0  max is: 31048.0
 mean is: 3089.923365  variance is: 15651513.756432075
6
 min is: 0.0  max is: 13630.0
 mean is: 928.25902  variance is: 3081761.8169486397
7
 min is: 0.0  max is: 9238.0
 mean is: 138.09383  variance is: 443951.7464459313
8
 min is: 0.0  max is: 125.17
 mean is: 3.2485793303000015  variance is: 8.2194850249125
9
 min is: 0.87589  max is: 19.167
 mean is: 6.498652902749999  variance is: 6.40504819135735
10
 min is: 0.0  max is: 13.23
 mean is: 2.09713912048  variance is: 4.36344047061341
11
 min is: 0.0  max is: 66.761
 mean is: 4.21766040935  variance is: 4.086371884226908
12
 min is: 0.0  max is: 73.902
 mean is: 2.69171845215  variance is: 2.198778474358265
13
 min is: 0.99049  max is: 975.04
 mean is: 10.271590475899998  variance is: 404.6462450411812
14
 min is: -999.9  max is: 797.2
 mean is: 5.781480500000001  variance is: 3406.52055097812
```

In [3]:

```
# 1.2
Xtr, Xva, Ytr, Yva = ml.splitData(X, Y)
Xt, Yt = Xtr[:5000], Ytr[:5000]
Xv, Yv = Xva[:2000], Yva[:2000]
XtS, params = ml.rescale(Xt)
XvS, _ = ml.rescale(Xv, params)

print('---XtS-----')

print_info(XtS, 'XtS')

print('---XvS-----')

print_info(XvS, 'XvS')
```

```
---XtS-----
1
XtS min is: -4.422165731512143 XtS max is: 1.2446776414093952
XtS mean is: 1.0618350643198937e-14 XtS variance is: 0.999999999999991
8
2
XtS min is: -3.837995400836122 XtS max is: 1.8142505916980844
XtS mean is: 8.15703060652595e-16 XtS variance is: 0.99999999999999974
3
XtS min is: -4.599184593067695 XtS max is: 1.8066817941254465
XtS mean is: -3.058886477447231e-14 XtS variance is: 0.999999999999999
7
4
XtS min is: -2.910816429908188 XtS max is: 1.9544977425417718
XtS mean is: -1.1679901490424526e-14 XtS variance is: 1.00000000000000
09
5
XtS min is: -0.7795113781142542 XtS max is: 7.300953888425734
XtS mean is: -3.1974423109204506e-17 XtS variance is: 1.00000000000000
09
6
XtS min is: -0.5162351009819977 XtS max is: 7.373421397063304
XtS mean is: 7.105427357601002e-18 XtS variance is: 1.0000000000000244
7
XtS min is: -0.20010710502010892 XtS max is: 13.767196827061854
XtS mean is: -4.263256414560601e-18 XtS variance is: 0.999999999999948
4
8
XtS min is: -1.1381986913324114 XtS max is: 7.353078467636559
XtS mean is: 1.581668129801983e-15 XtS variance is: 1.0000000000000004
9
XtS min is: -2.1005892848226817 XtS max is: 4.726589902128516
XtS mean is: -1.8900436771218666e-15 XtS variance is: 1.00000000000000
22
10
XtS min is: -0.9895913731224477 XtS max is: 5.432144742570911
XtS mean is: -1.8900436771218666e-15 XtS variance is: 0.99999999999999
82
11
XtS min is: -2.1053692164096467 XtS max is: 7.417399913270711
XtS mean is: 1.3642420526593924e-15 XtS variance is: 1.00000000000000
4
```

```
12
XtS min is: -1.9498143483595054 XtS max is: 6.112879769015834
XtS mean is: 3.182520913469489e-15 XtS variance is: 1.0000000000000004
13
XtS min is: -0.3759977761619259 XtS max is: 37.41876648085912
XtS mean is: -2.0889956431346945e-16 XtS variance is: 0.99999999999995
27
14
XtS min is: -16.304214604124 XtS max is: 12.784747661515619
XtS mean is: -1.3145040611561854e-17 XtS variance is: 1.00000000000001
4
---XvS-----
1
XvS min is: -4.836812807579572 XvS max is: 1.2446776414093952
XvS mean is: -0.030869645519057385 XvS variance is: 1.0307656154330378
2
XvS min is: -3.733324178752155 XvS max is: 2.128264257949985
XvS mean is: -0.027971814033608045 XvS variance is: 1.039536075629955
3
XvS min is: -4.038062281679677 XvS max is: 1.7058813789060432
XvS mean is: -0.03185561922043485 XvS variance is: 1.0500728964161328
4
XvS min is: -2.5801049161011202 XvS max is: 1.8936305927613308
XvS mean is: -0.01856620525229013 XvS variance is: 1.049917668055387
5
XvS min is: -0.7782096729034578 XvS max is: 7.300953888425734
XvS mean is: 0.00512007520793826 XvS variance is: 1.0619426788172626
6
XvS min is: -0.5162351009819977 XvS max is: 7.373421397063304
XvS mean is: 0.02314158387158499 XvS variance is: 1.0115360233082396
7
XvS min is: -0.20010710502010892 XvS max is: 13.767196827061854
XvS mean is: 0.0024845713954957916 XvS variance is: 0.9265435227832375
8
XvS min is: -1.1381986913324114 XvS max is: 7.914294457086004
XvS mean is: -0.025752751689473944 XvS variance is: 1.0073911704210738
9
XvS min is: -2.0671703869753504 XvS max is: 3.8863546676534666
XvS mean is: 0.03318811297166039 XvS variance is: 1.0463805999920182
10
XvS min is: -0.9895913731224477 XvS max is: 4.029847264246034
XvS mean is: 0.038881262397865615 XvS variance is: 1.047157501525062
11
XvS min is: -2.1053692164096467 XvS max is: 5.804757332478634
XvS mean is: -0.0008946835323864928 XvS variance is: 0.985364608468011
3
12
XvS min is: -1.9498143483595054 XvS max is: 6.018195917592274
XvS mean is: 0.015797086546689036 XvS variance is: 1.0045748809274226
13
XvS min is: -0.3759977761619259 XvS max is: 37.41876648085912
XvS mean is: -0.01914690592286119 XvS variance is: 1.0380157496184723
14
XvS min is: -16.304214604124 XvS max is: 9.626536010860384
XvS mean is: 0.0330106922645041 XvS variance is: 0.5645822302844906
```

Linear Classifiers (20 points)

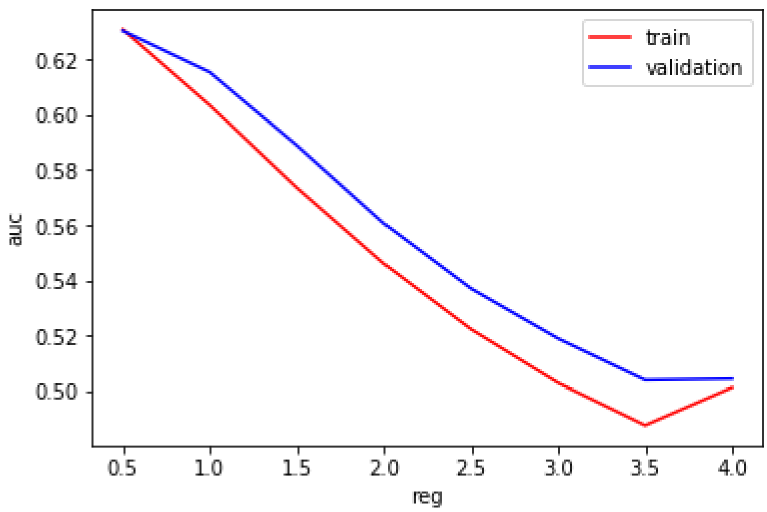
In [4]:

```
def plot(xlist, tr_auc, va_auc, xname):
    plt.plot(xlist, tr_auc, c='r', label='train')
    plt.plot(xlist, va_auc, c='b', label='validation')
    plt.xlabel(xname)
    plt.ylabel('auc')
    plt.legend()
    plt.show()

def linear_classfier_print(learner, XtS, Yt, XvS, Yv):
    reg = [0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0]
    tr_auc = []
    va_auc = []
    for r in reg:
        learner.train(XtS, Yt, reg=r, initStep=0.5, stopTol=1e-6, stopIter=100)
        tr_auc.append(learner.auc(XtS, Yt))
        va_auc.append(learner.auc(XvS, Yv))
    plot(reg, tr_auc, va_auc, 'reg')
```

In [5]:

```
# 2.1
learner = ml.linearC.linearClassify()
linear_classfier_print(learner, XtS, Yt, XvS, Yv)
```

```
/home/zhangjitao0405/uci-cs273/HW4/mltools/linearC.py:122: RuntimeWarn
ing: invalid value encountered in true_divide
  sigx  = np.exp(respi) / (1.0+np.exp(respi))
```



In [6]:

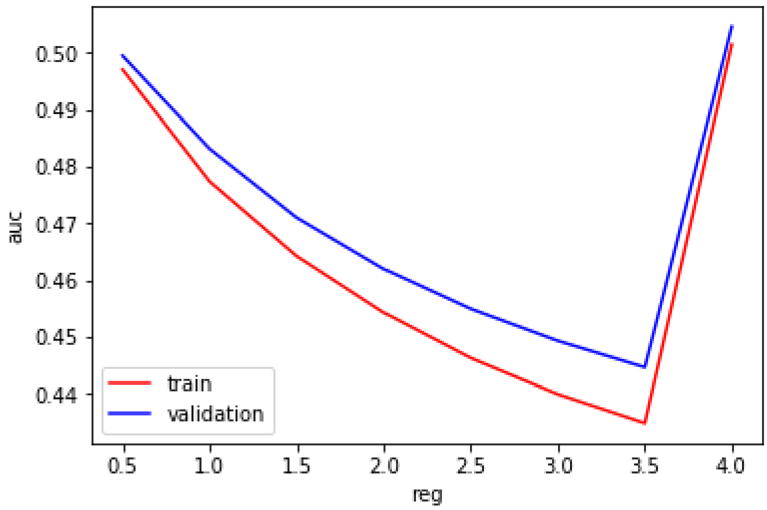
```
# 2.2
Xt2 = ml.transforms.fpoly(Xt, 2, bias=False)
Xv2 = ml.transforms.fpoly(Xv, 2, bias=False)
print(Xt2.shape[1])

# We originally have 14 features from x1 -> x14
# we pick 2 different from them to combine a new xi * xj feature, it will be 14 * 1
# we convert every feature to its square  x1 -> x1 * x1 , it will be 14

# so, totally 14 + 14 + 91 = 119
```

```
119
```

In [7]:

```
# 2.3
XtS2, params = ml.rescale(Xt2)
XvS2, _ = ml.rescale(Xv2, params)
learner_trans = ml.linearC.linearClassify()
linear_classfier_print(learner_trans, XtS2, Yt, XvS2, Yv)
```

```
/home/zhangjitao0405/uci-cs273/HW4/mltools/base.py:96: RuntimeWarning:
divide by zero encountered in log
  return - np.mean( np.log( P[ np.arange(M), Y ] ) ) # evaluate
/home/zhangjitao0405/uci-cs273/HW4/mltools/linearC.py:134: RuntimeWarn
ing: invalid value encountered in double_scalars
  done = (it > stopIter) or ( (it>1) and (abs(Jsur[-1]-Jsur[-2])<stopT
ol) )
/home/zhangjitao0405/uci-cs273/HW4/mltools/linearC.py:122: RuntimeWarn
ing: invalid value encountered in true_divide
  sigx  = np.exp(respi) / (1.0+np.exp(respi))
```



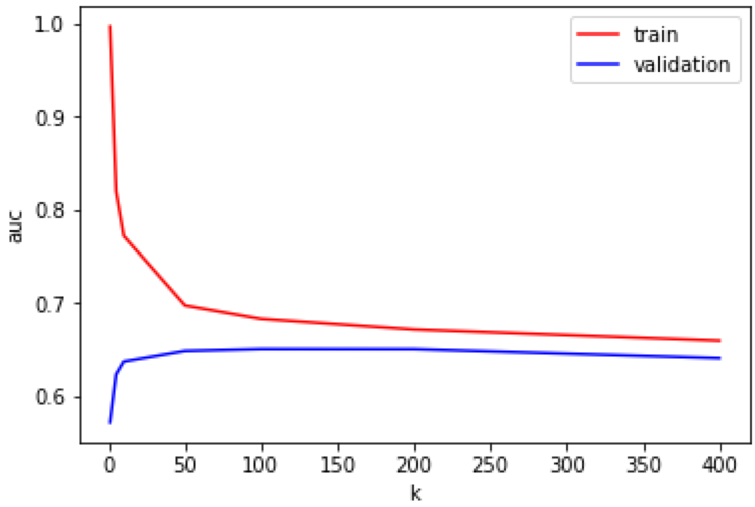
Nearest Neighbors (20 points)

In [8]:

```
def nearest_neighbors_print(XtS, Yt, XvS, Yv):
    klist = [1, 5, 10, 50, 100, 200, 400]
    tr_auc = []
    va_auc = []
    for k in klist:
        learner = ml.knn.knnClassify()
        learner.train(XtS, Yt, K=k, alpha=0.0)
        tr_auc.append(learner.auc(XtS, Yt))
        va_auc.append(learner.auc(XvS, Yv))

    plot(klist, tr_auc, va_auc, 'k')
```

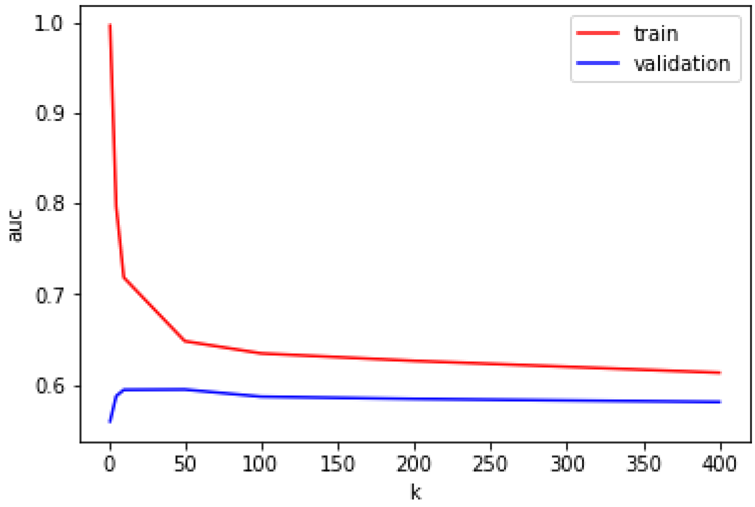
In [9]:

```
# 3.1
nearest_neighbors_print(XtS, Yt, XvS, Yv)
```



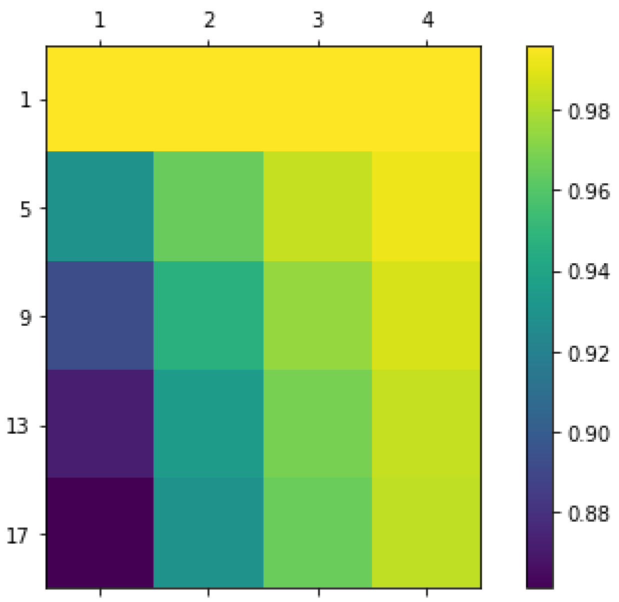
In [10]:

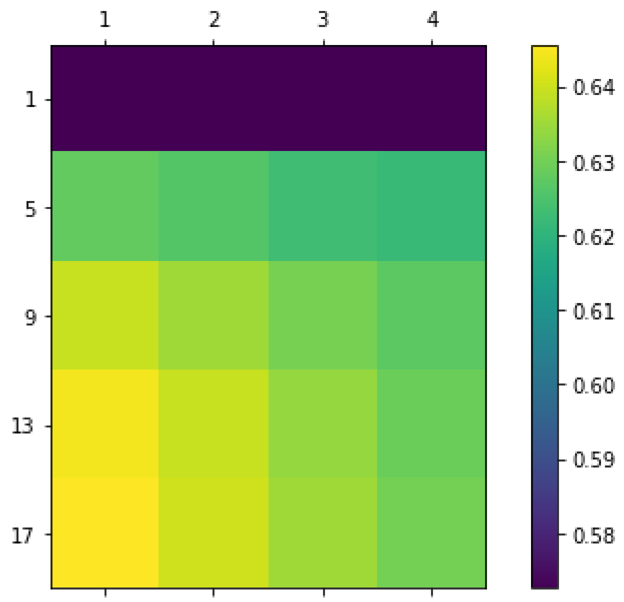
```
# 3.2
nearest_neighbors_print(Xt, Yt, Xv, Yv)
```



In [18]:

```
# 3.3
def plot_2d_auc(_2d_auc, x_list, y_list):
    f, ax = plt.subplots(1, 1, figsize=(8, 5))
    cax = ax.matshow(_2d_auc, interpolation='nearest')
    f.colorbar(cax)
    ax.set_xticklabels(['']+list(x_list))
    ax.set_yticklabels(['']+list(y_list))
    plt.show()

K = range(1,20,4)
A = range(1,5,1) # Or something else
tr_auc = np.zeros((len(K),len(A)))
va_auc = np.zeros((len(K),len(A)))
for i,k in enumerate(K):
    for j,a in enumerate(A):
        learner = ml.knn.knnClassify()
        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, Yv)

plot_2d_auc(tr_auc, A, K)
plot_2d_auc(va_auc, A, K)
```



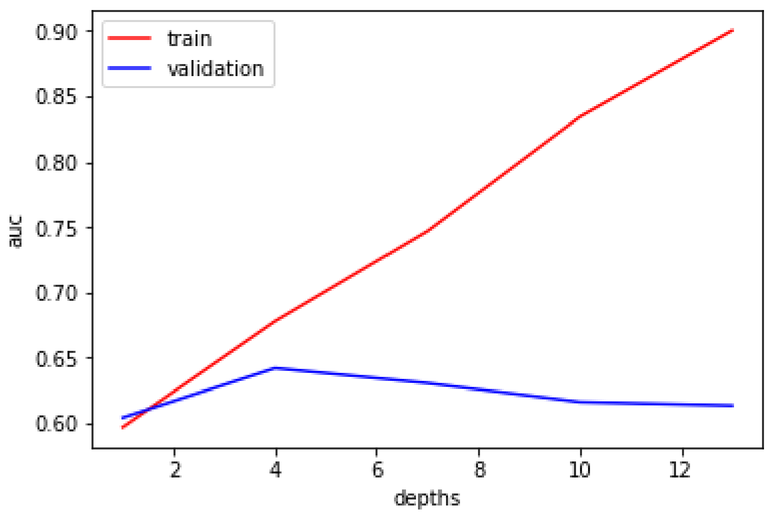


I would recommand the K is 17 and a is 1

Decision Trees (20 points)

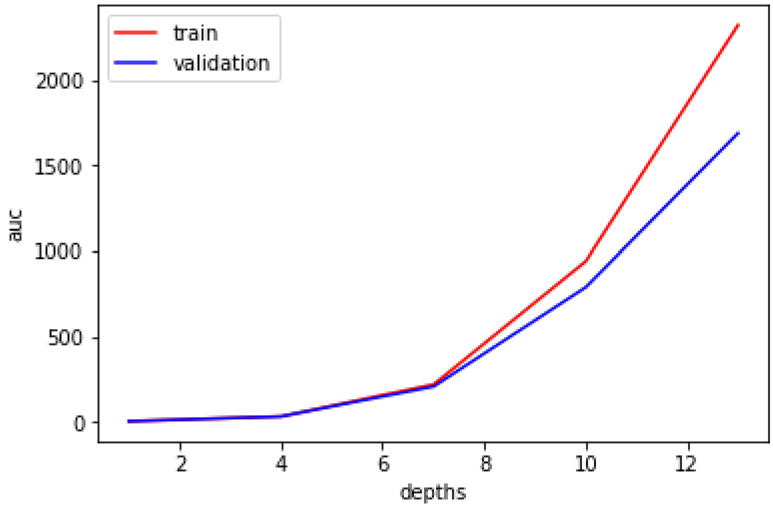
In [12]:

```
# 4.1
depths = range(1,15,3)
tr_auc = []
va_auc = []

for d in depths:
    learner = ml.dtree.treeClassify(XtS, Yt, maxDepth=d)
    tr_auc.append(learner.auc(XtS, Yt))
    va_auc.append(learner.auc(XvS, Yv))

plot(depths, tr_auc, va_auc, 'depths')
```



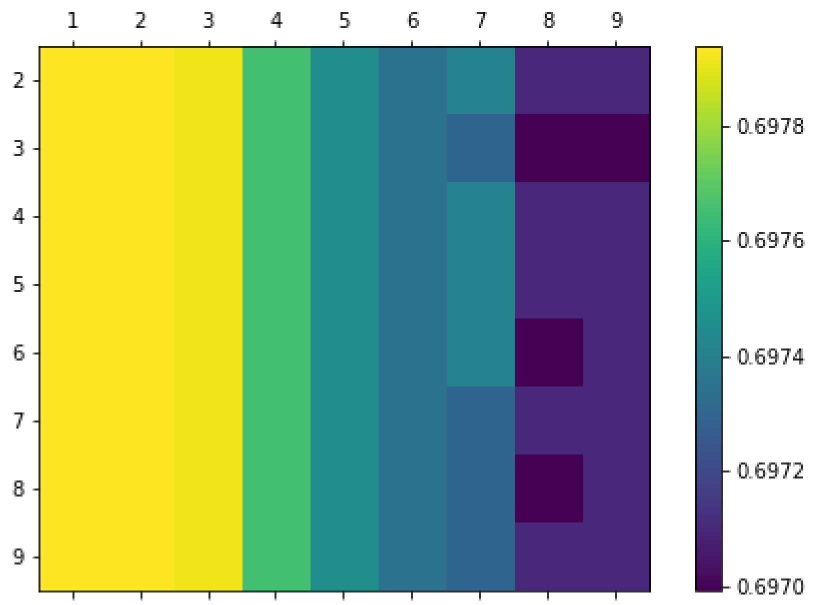
In [13]:

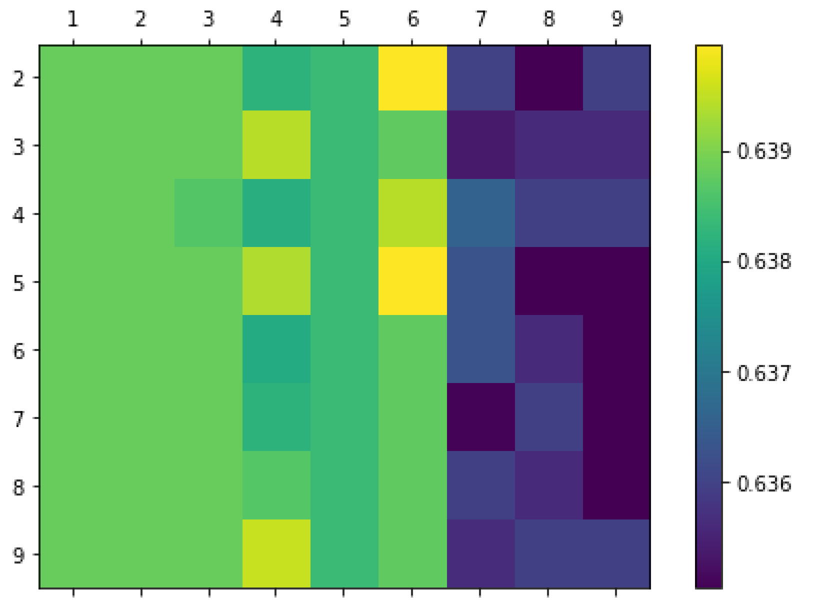
```
# 4.2
node_minParent_2 = []
node_minParent_4 = []

for d in depths:
    learner = ml.dtree.treeClassify(XtS, Yt, minParent=2, maxDepth=d)
    node_minParent_2.append(learner.sz)

    learner = ml.dtree.treeClassify(XtS, Yt, minParent=4, maxDepth=d)
    node_minParent_4.append(learner.sz)

plot(depths, node_minParent_2, node_minParent_4, 'depths')
```



In [14]:

```
# 4.3
minParents = range(2,10,1)
minLeaves = range(1,10,1)

tr_auc = np.zeros((len(minParents),len(minLeaves)))
va_auc = np.zeros((len(minParents),len(minLeaves)))
for i,p in enumerate(minParents):
    for j,l in enumerate(minLeaves):
        learner = ml.dtree.treeClassify(XtS, Yt, maxDepth=5, minParent=p, minLeaf=l)
        tr_auc[i][j] = learner.auc(XtS, Yt)
        va_auc[i][j] = learner.auc(XvS, Yv)

plot_2d_auc(tr_auc, minLeaves, minParents)
plot_2d_auc(va_auc, minLeaves, minParents)
```





I would recommand the minParent is 5 and minLeaf is 6

Neural Networks (20 points)

In [15]:

```
# 5.1
nodes = range(1,5,1)
layers = range(1,10,1)
tr_auc = np.zeros((len(nodes),len(layers)))
va_auc = np.zeros((len(nodes),len(layers)))
for i,n in enumerate(nodes):
    for j,l in enumerate(layers):
        nn = ml.nnet.nnetClassify()
        nn.init_weights([XtS.shape[1]] + [n for x in range(1,l+1)] + [2], 'random',
        nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=100) # 100 is fast to
        tr_auc[i][j] = nn.auc(XtS, Yt)
        va_auc[i][j] = nn.auc(XvS, Yv)

plot_2d_auc(tr_auc, layers, nodes)
plot_2d_auc(va_auc, layers, nodes)
```

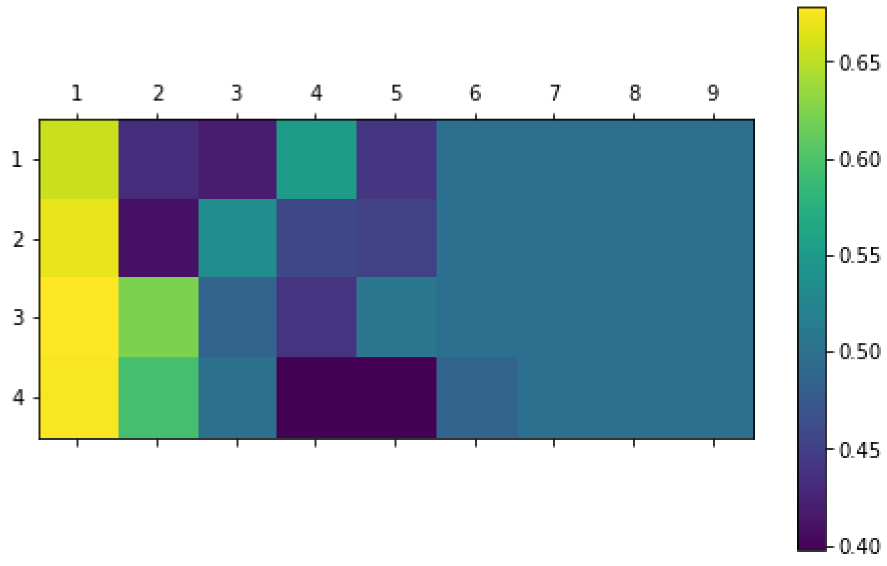
```
it 1 : Jsur = 0.4288143676366621, J01 = 0.3498
it 2 : Jsur = 0.4239632755196131, J01 = 0.3456
it 4 : Jsur = 0.4187788218961281, J01 = 0.3382
it 8 : Jsur = 0.419628948554538, J01 = 0.3402
it 16 : Jsur = 0.4201414410165782, J01 = 0.3358
it 32 : Jsur = 0.4209076331382794, J01 = 0.3358
it 64 : Jsur = 0.4217410449933179, J01 = 0.3358
it 1 : Jsur = 0.4583770472186758, J01 = 0.3358
it 2 : Jsur = 0.45199486546999634, J01 = 0.3358
it 4 : Jsur = 0.4481873398697302, J01 = 0.3358
it 8 : Jsur = 0.4467273769713889, J01 = 0.3358
it 16 : Jsur = 0.4462173647552678, J01 = 0.3358
it 32 : Jsur = 0.44609837124420865, J01 = 0.3358
it 64 : Jsur = 0.44608000171899403, J01 = 0.3358
it 1 : Jsur = 0.4583770471951559, J01 = 0.3358
it 2 : Jsur = 0.45199486533768085, J01 = 0.3358
it 4 : Jsur = 0.4481873398897029, J01 = 0.3358
it 8 : Jsur = 0.4467273771086681, J01 = 0.3358
it 16 : Jsur = 0.4462173651163337, J01 = 0.3358
it 32 : Jsur = 0.446098371861367, J01 = 0.3358
it 64 : Jsur = 0.44608000265079883, J01 = 0.3358
it 1 : Jsur = 0.45837704706104776, J01 = 0.3358
it 2 : Jsur = 0.4519948653077009, J01 = 0.3358
it 4 : Jsur = 0.4481873398941484, J01 = 0.3358
it 8 : Jsur = 0.44672737711079336, J01 = 0.3358
it 16 : Jsur = 0.4462173651178179, J01 = 0.3358
it 32 : Jsur = 0.4460983718620366, J01 = 0.3358
it 64 : Jsur = 0.4460800026509385, J01 = 0.3358
it 1 : Jsur = 0.4583770471818206, J01 = 0.3358
it 2 : Jsur = 0.45199486530977895, J01 = 0.3358
it 4 : Jsur = 0.4481873398946014, J01 = 0.3358
it 8 : Jsur = 0.4467273771119766, J01 = 0.3358
it 16 : Jsur = 0.44621736511833315, J01 = 0.3358
it 32 : Jsur = 0.44609837186248985, J01 = 0.3358
it 64 : Jsur = 0.44608000265101116, J01 = 0.3358
it 1 : Jsur = 0.4583770472264963, J01 = 0.3358
it 2 : Jsur = 0.4519948653494041, J01 = 0.3358
it 4 : Jsur = 0.448187339891822, J01 = 0.3358
it 8 : Jsur = 0.44672737710844324, J01 = 0.3358
it 16 : Jsur = 0.4462173651162204, J01 = 0.3358
it 32 : Jsur = 0.44609837186116835, J01 = 0.3358
```

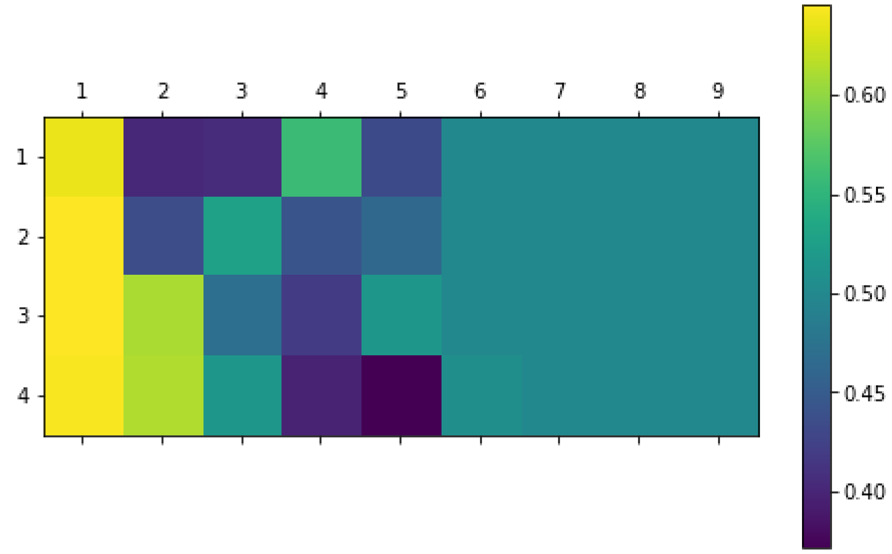
```
it 64 : Jsur = 0.4460800026508039, J01 = 0.3358
it 1 : Jsur = 0.458377047052691, J01 = 0.3358
it 2 : Jsur = 0.4519948653114232, J01 = 0.3358
it 4 : Jsur = 0.44818733989369813, J01 = 0.3358
it 8 : Jsur = 0.44672737711031807, J01 = 0.3358
it 16 : Jsur = 0.4462173651175439, J01 = 0.3358
it 32 : Jsur = 0.44609837186186424, J01 = 0.3358
it 64 : Jsur = 0.4460800026509114, J01 = 0.3358
it 1 : Jsur = 0.45837704723105854, J01 = 0.3358
it 2 : Jsur = 0.4519948653393525, J01 = 0.3358
it 4 : Jsur = 0.4481873398954589, J01 = 0.3358
it 8 : Jsur = 0.4467273771098618, J01 = 0.3358
it 16 : Jsur = 0.44621736511714355, J01 = 0.3358
it 32 : Jsur = 0.44609837186155044, J01 = 0.3358
it 64 : Jsur = 0.44608000265086156, J01 = 0.3358
it 1 : Jsur = 0.45837704747776936, J01 = 0.3358
it 2 : Jsur = 0.45199486534599864, J01 = 0.3358
it 4 : Jsur = 0.44818733989114434, J01 = 0.3358
it 8 : Jsur = 0.44672737711090466, J01 = 0.3358
it 16 : Jsur = 0.4462173651173838, J01 = 0.3358
it 32 : Jsur = 0.4460983718622308, J01 = 0.3358
it 64 : Jsur = 0.4460800026509748, J01 = 0.3358
it 1 : Jsur = 0.427068239661609, J01 = 0.336
it 2 : Jsur = 0.4195837685301663, J01 = 0.3254
it 4 : Jsur = 0.4149883137526034, J01 = 0.3172
it 8 : Jsur = 0.4128629134760127, J01 = 0.317
it 16 : Jsur = 0.4118340802502113, J01 = 0.3138
it 32 : Jsur = 0.41112391819670246, J01 = 0.3118
it 64 : Jsur = 0.4108421789518941, J01 = 0.3108
it 1 : Jsur = 0.4583770471826313, J01 = 0.3358
it 2 : Jsur = 0.4519948654830231, J01 = 0.3358
it 4 : Jsur = 0.4481873398904133, J01 = 0.3358
it 8 : Jsur = 0.4467273769178282, J01 = 0.3358
it 16 : Jsur = 0.44621736454765704, J01 = 0.3358
it 32 : Jsur = 0.4460983707823414, J01 = 0.3358
it 64 : Jsur = 0.44608000086508254, J01 = 0.3358
it 1 : Jsur = 0.4583770476186535, J01 = 0.3358
it 2 : Jsur = 0.45199486538377, J01 = 0.3358
it 4 : Jsur = 0.44818733989674336, J01 = 0.3358
it 8 : Jsur = 0.4467273771103943, J01 = 0.3358
it 16 : Jsur = 0.4462173651169538, J01 = 0.3358
it 32 : Jsur = 0.44609837186166634, J01 = 0.3358
it 64 : Jsur = 0.44608000265091924, J01 = 0.3358
it 1 : Jsur = 0.45837704720162553, J01 = 0.3358
it 2 : Jsur = 0.4519948653432444, J01 = 0.3358
it 4 : Jsur = 0.44818733989149206, J01 = 0.3358
it 8 : Jsur = 0.4467273771086309, J01 = 0.3358
it 16 : Jsur = 0.4462173651163641, J01 = 0.3358
it 32 : Jsur = 0.446098371861273, J01 = 0.3358
it 64 : Jsur = 0.4460800026508201, J01 = 0.3358
it 1 : Jsur = 0.4583770473778968, J01 = 0.3358
it 2 : Jsur = 0.45199486535510297, J01 = 0.3358
it 4 : Jsur = 0.448187339893428, J01 = 0.3358
it 8 : Jsur = 0.4467273771097696, J01 = 0.3358
it 16 : Jsur = 0.4462173651168199, J01 = 0.3358
it 32 : Jsur = 0.4460983718616104, J01 = 0.3358
it 64 : Jsur = 0.4460800026508742, J01 = 0.3358
it 1 : Jsur = 0.4583770473587085, J01 = 0.3358
it 2 : Jsur = 0.451994865356642, J01 = 0.3358
it 4 : Jsur = 0.44818733989288995, J01 = 0.3358
it 8 : Jsur = 0.44672737710937854, J01 = 0.3358
```

```
it 16 : Jsur = 0.44621736511660237, J01 = 0.3358
it 32 : Jsur = 0.44609837186148177, J01 = 0.3358
it 64 : Jsur = 0.44608000265085396, J01 = 0.3358
it 1 : Jsur = 0.45837704776468674, J01 = 0.3358
it 2 : Jsur = 0.45199486540657396, J01 = 0.3358
it 4 : Jsur = 0.44818733989995146, J01 = 0.3358
it 8 : Jsur = 0.4467273771102557, J01 = 0.3358
it 16 : Jsur = 0.4462173651167793, J01 = 0.3358
it 32 : Jsur = 0.44609837186137286, J01 = 0.3358
it 64 : Jsur = 0.4460800026508348, J01 = 0.3358
it 1 : Jsur = 0.45837704705861626, J01 = 0.3358
it 2 : Jsur = 0.45199486532808403, J01 = 0.3358
it 4 : Jsur = 0.4481873398894935, J01 = 0.3358
it 8 : Jsur = 0.4467273771081764, J01 = 0.3358
it 16 : Jsur = 0.44621736511623455, J01 = 0.3358
it 32 : Jsur = 0.4460983718612206, J01 = 0.3358
it 64 : Jsur = 0.4460800026508124, J01 = 0.3358
it 1 : Jsur = 0.4583770470813062, J01 = 0.3358
it 2 : Jsur = 0.45199486531473065, J01 = 0.3358
it 4 : Jsur = 0.4481873398932109, J01 = 0.3358
it 8 : Jsur = 0.4467273771102895, J01 = 0.3358
it 16 : Jsur = 0.44621736511746557, J01 = 0.3358
it 32 : Jsur = 0.4460983718618767, J01 = 0.3358
it 64 : Jsur = 0.44608000265091396, J01 = 0.3358
it 1 : Jsur = 0.42694600454169196, J01 = 0.3314
it 2 : Jsur = 0.4178277467689155, J01 = 0.32
it 4 : Jsur = 0.41283424277922476, J01 = 0.3148
it 8 : Jsur = 0.4087957892485247, J01 = 0.3126
it 16 : Jsur = 0.4068326135719936, J01 = 0.3078
it 32 : Jsur = 0.4055558009479352, J01 = 0.307
it 64 : Jsur = 0.40431464837207404, J01 = 0.3056
it 1 : Jsur = 0.4583770479723966, J01 = 0.3358
it 2 : Jsur = 0.45199486602273664, J01 = 0.3358
it 4 : Jsur = 0.4481873403473441, J01 = 0.3358
it 8 : Jsur = 0.4467273764699388, J01 = 0.3358
it 16 : Jsur = 0.44621736210718066, J01 = 0.3358
it 32 : Jsur = 0.4460983636153127, J01 = 0.3358
it 64 : Jsur = 0.4460799814216327, J01 = 0.3358
it 1 : Jsur = 0.45837704771503923, J01 = 0.3358
it 2 : Jsur = 0.45199486540519285, J01 = 0.3358
it 4 : Jsur = 0.44818733989714765, J01 = 0.3358
it 8 : Jsur = 0.44672737710936955, J01 = 0.3358
it 16 : Jsur = 0.4462173651162747, J01 = 0.3358
it 32 : Jsur = 0.4460983718611741, J01 = 0.3358
it 64 : Jsur = 0.44608000265079656, J01 = 0.3358
it 1 : Jsur = 0.4583770473152115, J01 = 0.3358
it 2 : Jsur = 0.4519948653007534, J01 = 0.3358
it 4 : Jsur = 0.4481873399013095, J01 = 0.3358
it 8 : Jsur = 0.4467273771150975, J01 = 0.3358
it 16 : Jsur = 0.4462173651201465, J01 = 0.3358
it 32 : Jsur = 0.44609837186338736, J01 = 0.3358
it 64 : Jsur = 0.4460800026511482, J01 = 0.3358
it 1 : Jsur = 0.45837704787817646, J01 = 0.3358
it 2 : Jsur = 0.4519948653903264, J01 = 0.3358
it 4 : Jsur = 0.44818733990239434, J01 = 0.3358
it 8 : Jsur = 0.4467273771133392, J01 = 0.3358
it 16 : Jsur = 0.44621736511841936, J01 = 0.3358
it 32 : Jsur = 0.4460983718625046, J01 = 0.3358
it 64 : Jsur = 0.44608000265101383, J01 = 0.3358
it 1 : Jsur = 0.45837704738990454, J01 = 0.3358
it 2 : Jsur = 0.45199486535119193, J01 = 0.3358
```

```
it 4 : Jsur = 0.44818733989232096, J01 = 0.3358
it 8 : Jsur = 0.4467273771099466, J01 = 0.3358
it 16 : Jsur = 0.44621736511691307, J01 = 0.3358
it 32 : Jsur = 0.4460983718617564, J01 = 0.3358
it 64 : Jsur = 0.44608000265089814, J01 = 0.3358
it 1 : Jsur = 0.4583770471923546, J01 = 0.3358
it 2 : Jsur = 0.4519948653424887, J01 = 0.3358
it 4 : Jsur = 0.44818733989155773, J01 = 0.3358
it 8 : Jsur = 0.44672737710863814, J01 = 0.3358


it 16 : Jsur = 0.4462173651163739, J01 = 0.3358
it 32 : Jsur = 0.44609837186127216, J01 = 0.3358
it 64 : Jsur = 0.4460800026508202, J01 = 0.3358
it 1 : Jsur = 0.4583770473468667, J01 = 0.3358
it 2 : Jsur = 0.45199486533725913, J01 = 0.3358
it 4 : Jsur = 0.4481873398992363, J01 = 0.3358
it 8 : Jsur = 0.44672737711192706, J01 = 0.3358
it 16 : Jsur = 0.4462173651182413, J01 = 0.3358
it 32 : Jsur = 0.44609837186217355, J01 = 0.3358
it 64 : Jsur = 0.4460800026509587, J01 = 0.3358
it 1 : Jsur = 0.4583770475154636, J01 = 0.3358
it 2 : Jsur = 0.451994865373, J01 = 0.3358
it 4 : Jsur = 0.448187339893602, J01 = 0.3358
it 8 : Jsur = 0.44672737710966287, J01 = 0.3358
it 16 : Jsur = 0.44621736511658383, J01 = 0.3358
it 32 : Jsur = 0.4460983718615376, J01 = 0.3358
it 64 : Jsur = 0.44608000265086367, J01 = 0.3358
it 1 : Jsur = 0.42602123889920185, J01 = 0.3308
it 2 : Jsur = 0.41822972189116353, J01 = 0.3226
it 4 : Jsur = 0.4139285790624123, J01 = 0.318
it 8 : Jsur = 0.410495040550343, J01 = 0.3118
it 16 : Jsur = 0.408082248981815, J01 = 0.3096
it 32 : Jsur = 0.40681285732326533, J01 = 0.3058
it 64 : Jsur = 0.40598201367135645, J01 = 0.308
it 1 : Jsur = 0.45837704801535, J01 = 0.3358
it 2 : Jsur = 0.4519948657114581, J01 = 0.3358
it 4 : Jsur = 0.44818733994407195, J01 = 0.3358
it 8 : Jsur = 0.44672737653219763, J01 = 0.3358
it 16 : Jsur = 0.4462173628005607, J01 = 0.3358
it 32 : Jsur = 0.4460983652032588, J01 = 0.3358
it 64 : Jsur = 0.44607998393814124, J01 = 0.3358
it 1 : Jsur = 0.45837704759272685, J01 = 0.3358
it 2 : Jsur = 0.4519948653465322, J01 = 0.3358
it 4 : Jsur = 0.4481873398965801, J01 = 0.3358
it 8 : Jsur = 0.446727377112952, J01 = 0.3358
it 16 : Jsur = 0.44621736511854787, J01 = 0.3358
it 32 : Jsur = 0.4460983718627125, J01 = 0.3358
it 64 : Jsur = 0.4460800026510087, J01 = 0.3358
it 1 : Jsur = 0.4583770472286663, J01 = 0.3358
it 2 : Jsur = 0.4519948652747764, J01 = 0.3358
it 4 : Jsur = 0.4481873399109659, J01 = 0.3358
it 8 : Jsur = 0.44672737711848803, J01 = 0.3358
it 16 : Jsur = 0.4462173651222871, J01 = 0.3358
it 32 : Jsur = 0.44609837186422063, J01 = 0.3358
it 64 : Jsur = 0.4460800026512712, J01 = 0.3358
it 1 : Jsur = 0.45837704745435676, J01 = 0.3358
it 2 : Jsur = 0.45199486532415384, J01 = 0.3358
it 4 : Jsur = 0.4481873398966707, J01 = 0.3358
it 8 : Jsur = 0.4467273771134918, J01 = 0.3358
it 16 : Jsur = 0.4462173651190209, J01 = 0.3358
it 32 : Jsur = 0.44609837186299084, J01 = 0.3358
```

```
it 64 : Jsur = 0.4460800026510909, J01 = 0.3358
it 1 : Jsur = 0.4583770478810873, J01 = 0.3358
it 2 : Jsur = 0.45199486538595873, J01 = 0.3358
it 4 : Jsur = 0.4481873399178541, J01 = 0.3358
it 8 : Jsur = 0.4467273771162187, J01 = 0.3358
it 16 : Jsur = 0.44621736512045035, J01 = 0.3358
it 32 : Jsur = 0.4460983718627822, J01 = 0.3358
it 64 : Jsur = 0.4460800026510456, J01 = 0.3358
it 1 : Jsur = 0.4583770478723736, J01 = 0.3358
it 2 : Jsur = 0.451994865395244, J01 = 0.3358
it 4 : Jsur = 0.44818733990346027, J01 = 0.3358
it 8 : Jsur = 0.446727377112851, J01 = 0.3358
it 16 : Jsur = 0.44621736511822946, J01 = 0.3358
it 32 : Jsur = 0.4460983718622436, J01 = 0.3358
it 64 : Jsur = 0.44608000265097114, J01 = 0.3358
it 1 : Jsur = 0.4583770477824212, J01 = 0.3358
it 2 : Jsur = 0.4519948654025167, J01 = 0.3358
it 4 : Jsur = 0.448187339894827, J01 = 0.3358
it 8 : Jsur = 0.4467273771098169, J01 = 0.3358
it 16 : Jsur = 0.4462173651164303, J01 = 0.3358
it 32 : Jsur = 0.4460983718615072, J01 = 0.3358
it 64 : Jsur = 0.4460800026508601, J01 = 0.3358
it 1 : Jsur = 0.45837704738064783, J01 = 0.3358
it 2 : Jsur = 0.45199486535406375, J01 = 0.3358
it 4 : Jsur = 0.44818733989310056, J01 = 0.3358
it 8 : Jsur = 0.44672737710977417, J01 = 0.3358
it 16 : Jsur = 0.44621736511683074, J01 = 0.3358
it 32 : Jsur = 0.44609837186163515, J01 = 0.3358
it 64 : Jsur = 0.4460800026508783, J01 = 0.3358
```





I would recommand node in each layer is 4 and layer is 1

In [16]:

```
# 5.2
def sig(z): return np.atleast_2d(z)
def dsig(z): return np.atleast_2d(1)

def activation_switch(name):
    nn = ml.nnet.nnetClassify()
    nn.init_weights([XtS.shape[1],5,2], 'random', XtS, Yt)
    nn.setActivation(name, sig, dsig)
    nn.train(XtS, Yt, stopTol=1e-8, stepsize=.25, stopIter=300)
    print(name + " trian auc:",nn.auc(XtS,Yt))
    print(name + " validation auc:",nn.auc(XvS,Yv))
    print('---------------------------------------')

activation_switch('custom')
activation_switch('logistic')
activation_switch('htangent')
```

```
it 1 : Jsur = 0.4282136543634937, J01 = 0.3156
it 2 : Jsur = 0.42093814223664777, J01 = 0.3118
it 4 : Jsur = 0.4144330867408389, J01 = 0.308
it 8 : Jsur = 0.41066736183976166, J01 = 0.3038
it 16 : Jsur = 0.40858058227351185, J01 = 0.3026
it 32 : Jsur = 0.4078519729171966, J01 = 0.3028
it 64 : Jsur = 0.4076626319507943, J01 = 0.3066
it 128 : Jsur = 0.40762231430005724, J01 = 0.3068
custom trian auc: 0.6689316582134123
custom validation auc: 0.647759861076656
---------------------------------------
it 1 : Jsur = 0.42766468258506174, J01 = 0.3332
it 2 : Jsur = 0.4190564835364335, J01 = 0.3184
it 4 : Jsur = 0.4160874228735016, J01 = 0.3114
it 8 : Jsur = 0.4157725910679841, J01 = 0.3114
it 16 : Jsur = 0.41636097122693916, J01 = 0.3104
it 32 : Jsur = 0.4171513858306903, J01 = 0.3358
it 64 : Jsur = 0.4179114980966551, J01 = 0.3358
it 128 : Jsur = 0.41858944576311168, J01 = 0.3358
it 256 : Jsur = 0.4191892440683062, J01 = 0.3358
logistic trian auc: 0.6613389732600258
logistic validation auc: 0.6444003417955291
---------------------------------------
it 1 : Jsur = 0.4265493246412589, J01 = 0.3296
it 2 : Jsur = 0.4174722907085884, J01 = 0.3216
it 4 : Jsur = 0.41287632446740336, J01 = 0.3146
it 8 : Jsur = 0.4088740942561583, J01 = 0.31
it 16 : Jsur = 0.4064165761599706, J01 = 0.3052
it 32 : Jsur = 0.4049430632143297, J01 = 0.3048
it 64 : Jsur = 0.403532169063381, J01 = 0.3064
it 128 : Jsur = 0.40148266364878765, J01 = 0.3072
it 256 : Jsur = 0.3993478355626767, J01 = 0.3018
htangent trian auc: 0.687201878636482
htangent validation auc: 0.6516232531216406
---------------------------------------
```

In this case, the custom activation is better than the logistic activation and a little worse than the htangent activation.

Conclusions (5 points)

In [3]:

```
# I prefer the decision tree, my name is Jitao,
# and my leaderboard is 10 with score 0.72677
Xte = np.genfromtxt('data/X_test.txt', delimiter=None)
learner = ml.dtree.treeClassify(X, Y, maxDepth=30, minParent=10, minLeaf=10)
Yte = np.vstack((np.arange(Xte.shape[0]), learner.predictSoft(Xte)[:,1])).T
np.savetxt('Y_submit.txt', Yte, '%d, %.2f', header='ID,Prob1', comments='', delimite
print('finished')
```

```
finished
```

Statement of Collaboration (5 points)

I did my homework independently.