Machine Learning Exercises: Set 4

Roger Garriga Calleja

March 14, 2017

Problem 13: Let $(x_1, y_1), \ldots, (x_n, y_n)$ be data in $\Re^d \times \{-1, 1\}$. Suppose that the data is linearly separable, that is, there exists a $w \in \Re^d$ such that $y_i w^T x_i > 0$ for all $i = 1, \ldots, n$. The margin of such vector is

$$\gamma(w) = \min_{i=1,\ldots,n} \frac{y_i w^T x_i}{\|w\|}.$$

Formulate a convex optimization problem whose solution is a vector w^* that classifies the data correctly (i.e., $y_i w^T x_i > 0$ for all $i = 1, \ldots, n$) and maximizes the margin. Show that the optimal solution w^* lies in the vector space spanned by the examples x_i for which the margin $\frac{y_i {w^*}^T x_i}{\|w^*\|}$ is minimal among all examples. (These are called the support vector).

The aim is to maximize $\gamma(w)$ with the constraints $y_i w^T x_i > 0$. So,

$$\max_{w \in U_{\mathcal{D}_n}} \left\{ \min_{i=1,\ldots,n} \frac{y_i w^T x_i}{\|w\|} \right\}, \; U_{\mathcal{D}_n} = \{w : y_i w^T x_i > 0, \|w\| = 1 \forall i = 1, \ldots, n\}, \tag{1}$$

observe that we should put a condition $\|w\| = 1$ to avoid the solution $w = \infty$, and the margin does not change because it is scale invariant $\frac{kw^T}{\|kw\|} = \frac{w^T}{\|w\|}$. However, the function to maximize is not convex, so we have manipulate it. Since $\gamma(w)$ is the minimum $\frac{y_i w^T x_i}{\|w\|}$ over $i = 1, \ldots, n$, $\frac{y_i w^T x_i}{\|w\|} \geqslant \gamma(w)$. So we can write the problem as

$$\max_{w \in U_{\mathcal{D}_n}, \gamma \in \Re^d} \gamma, \; U_{\mathcal{D}_n} = \{\gamma \in \Re^d, w : \frac{y_i w^T x_i}{\|w\|} \geqslant \gamma, \gamma \geqslant 0, \|w\| = 1, \forall i = 1, \ldots, n\}, \tag{2}$$

using γ as a dummy variable. We still have the non-convex constrain $\|w\| = 1$, in order to avoid it we will optimize over β where $\gamma = \frac{\beta}{\|w\|}$. And without loss of generality (because the it is still scale invariant), we can fix $\beta = 1$. Then, the optimization problem can be written as

$$\max_{w \in U_{\mathcal{D}_n}} \frac{1}{\|w\|}, \; U_{\mathcal{D}_n} = \{w : y_i w^T x_i \geqslant 1, \forall i = 1, \ldots, n\}. \tag{3}$$

Equivalently, we can put this into the convex optimization framework as

$$\min_w \frac{1}{2} \|w\|^2, \tag{4}$$

$$\text{such that } y_i w^T x_i \geqslant 1, \; \forall i = 1, \ldots, n. \tag{5}$$

In a more standardized manner, the constrains can be written as $-y_i w^T x_i + 1 \leqslant 0$. Now, in order to solve this problem we formulate the Lagrangian and minimize it

$$\mathcal{L}(w, \beta) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i w^T x_i - 1), \tag{6}$$

$$0 = \frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^{n} \beta_i y_i x_i, \tag{7}$$

which leads to the solution on the primal $w^* = \sum_{i=1}^{n} \alpha_i y_i x_i$. Now, imposing the Karush-Kuhn-Tucker (KKT) conditions:

$$\frac{\partial \mathcal{L}(\sqsupseteq^*, \alpha^*)}{\partial w} = 0, \tag{8}$$

$$\alpha_i^* (y_i (w^*)^T x_i - 1) = 0, \ i = 1, \ldots, n, \tag{9}$$

$$y_i (w^*)^T x_i - 1 \leqslant 0, \ i = 1, \ldots, n, \tag{10}$$

$$\alpha_i^* \geqslant 0, \ i = 1, \ldots, n. \tag{11}$$

Since w^* is a linear combination of the examples x_i that have a $\alpha_i > 0$ and $\alpha_i > 0 \iff y_i (w^*)^T x_i - 1 = 0$ by KKT conditions, by the definition of γ, that will only hold when the x_i have the minimum margin.

Problem 14: Let $\mathcal{H}$ be the Hilbert space of all sequences $s = \{s_n\}_{n=0}^{\infty}$ satisfying $\sum_{n=0}^{\infty} s_n^2 < \infty$ with inner product $\langle s, t \rangle = \sum_{n=0}^{\infty} s_n t_n$. Consider the feature map $\Phi : \Re \to \mathcal{H}$ that assigns to each real number x, the sequence $\Phi(x)$ whose n-th element equals

$$(\Phi(x))_n = \frac{1}{\sqrt{n!}} x^n e^{-\frac{x^2}{2}}, \ n = 0, 1, 2, \ldots.$$

Determine the kernel function $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$ for $x, y \in \Re$.
Can you generalize the kernel so that it is defined on $\Re^d \times \Re^d$ instead of $\Re \times \Re$? What is the corresponding feature map?

Since $(\Phi(x))_n = \frac{1}{\sqrt{n!}} x^n e^{-\frac{x^2}{2}}$ maps the the space $\mathcal{H}$ with the inner product $\langle s, t \rangle = \sum_{n=0}^{\infty} s_n t_n$, the kernel $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$ is

$$K(x, y) = \langle \Phi(x), \Phi(y) \rangle = \sum_{n=0}^{\infty} \frac{x^n}{\sqrt{n!}} e^{-\frac{x^2}{2}} \frac{y^n}{\sqrt{n!}} e^{-\frac{y^2}{2}} = e^{-\frac{x^2+y^2}{2}} \sum_{n=0}^{\infty} \frac{(xy)^n}{n!} = e^{-\frac{x^2+y^2}{2}} e^{xy}, \tag{12}$$

where the last equality is due to the fact that $\sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x$ (Taylor expansion). Then,

$$K(x, y) = e^{-\frac{x^2+y^2}{2}+xy} = e^{-\frac{x^2+y^2-2xy}{2}} = e^{-\frac{(x-y)^2}{2}}. \tag{13}$$

This kernel can be generalized as $K(x, y) = e^{-\frac{\|x-y\|^2}{2}}$, where $x, y \in \Re^d$. Its feature map can be derived from

$$\langle \Phi(x), \Phi(y) \rangle =) e^{-\frac{\|x-y\|^2}{2}} = e^{-\frac{\|x\|^2}{2}} e^{-\frac{\|y\|^2}{2}} e^{x^T y} = e^{-\frac{\|x\|^2}{2}} e^{-\frac{\|y\|^2}{2}} \sum_{n=0}^{\infty} \frac{(x^T y)^n}{n!} = e^{-\frac{\|x\|^2}{2}} e^{-\frac{\|y\|^2}{2}} \sum_{n=0}^{\infty} \frac{\left(\sum_{i=1}^{d} x_i y_i\right)^n}{n!}. \tag{14}$$

So, the sequence will be such that for each n, there will be a "vector" of $m_n(d)$ numbers corresponding to the different combinations of the d components of a d-dim vector x that multiplied give a polynomial of degree n. For example, for $d = 2$ and $n = 2$, we need to get $\frac{(x1y_1+x_2y_2)^2}{2!}$, as

$$\frac{(x1y_1 + x_2y_2)^2}{2!} = \frac{(x_1^2y_1^2)}{2!} + \frac{(x_2^2y_2^2)}{2!} + \frac{2x_1x_2y_1y_2}{2!}, \tag{15}$$

we will need to multiply $(\frac{1}{\sqrt{2!}}x_1^2, \frac{1}{\sqrt{2!}}x_2^2, x_1x_2)^T(\frac{1}{\sqrt{2!}}y_1^2, \frac{1}{\sqrt{2!}}y_2^2, y_1y_2)$. For $d = 3$, $n = 3$, we need to get $\frac{(x1y_1+x_2y_2+x_3y_3)^3}{2!}$, as

$$\frac{(x1y_1 + x_2y_2 + x_3y_3)^3}{3!} = \frac{x_1^3y_1^3 + 3x_1^2x_2y_2y_1^2 + 3x_1^2x_3y_3y_1^2 + 3x_1x_2^2y_2^2y_1 + 3x_1x_3^2y_3^2y_1}{3!} + \tag{16}$$

$$+ \frac{6x_1x_2x_3y_2y_3y_1 + x_2^3y_2^3 + x_3^3y_3^3 + 3x_2x_3^2y_2y_3^2 + 3x_2^2x_3y_2^2y_3}{3!}, \tag{17}$$

we will need to multiply

$$(\frac{x_1^3}{\sqrt{3!}}, \frac{\sqrt{3}x_1^2x_2}{\sqrt{3!}}, \frac{\sqrt{3}x_1^2x_3}{\sqrt{3!}}, \frac{\sqrt{3}x_1x_2^2}{\sqrt{3!}}, \frac{\sqrt{3}x_1x_3^2}{\sqrt{3!}}, \frac{\sqrt{6}x_1x_2x_3}{\sqrt{3!}}, \frac{x_2^3}{\sqrt{3!}}, \frac{x_3^3}{\sqrt{3!}}, \frac{\sqrt{3}x_2x_3^2}{\sqrt{3!}}, \frac{\sqrt{3}x_2^2x_3}{\sqrt{3!}})^T \tag{18}$$

$$(\frac{y_1^3}{\sqrt{3!}}, \frac{\sqrt{3}y_1^2y_2}{\sqrt{3!}}, \frac{\sqrt{3}y_1^2y_3}{\sqrt{3!}}, \frac{\sqrt{3}y_1y_2^2}{\sqrt{3!}}, \frac{\sqrt{3}y_1y_3^2}{\sqrt{3!}}, \frac{\sqrt{6}y_1y_2y_3}{\sqrt{3!}}, \frac{y_2^3}{\sqrt{3!}}, \frac{y_3^3}{\sqrt{3!}}, \frac{\sqrt{3}y_2y_3^2}{\sqrt{3!}}, \frac{\sqrt{3}y_2^2y_3}{\sqrt{3!}}). \tag{19}$$

This would keep growing for each n as d increases. In general the mapping would be

$$(\Phi(x))_n = e^{-\frac{\|x\|^2}{2}} \left(\frac{x_1^{n_1}x_2^{n_2}\cdots x_d^{n_d}}{\sqrt{n_1!n_2!\cdots n_d!}} \right)_{\sum_{i=1}^{d} n_i = n}. \tag{20}$$

Problem 15: Let $K_1, K_2 : \mathcal{X} \times \mathcal{X} \to \Re$ be kernels. Prove that $K_1 + K_2$ and K_1K_2 are also kernels.

Given K_1, K_2 kernels, let us denote by $\Phi_1(x)$ and $\Phi_2(x)$ their respective feature spaces. Then,

$$K_3(x, y) = (K_1 + K_2)(x, y) = \langle \Phi_1(x), \Phi_1(y) \rangle + \langle \Phi_2(x), \Phi_2(y) \rangle. \tag{21}$$

If $\Phi_3(x) = (\Phi_1(x), \Phi_2(x)$ (concatenation of the two feature spaces), then

$$\langle \Phi_3(x), \Phi_3(y) \rangle = \langle (\Phi_1(x), \Phi_2(x)), (\Phi_1(y), \Phi_2(y)) \rangle = \langle \Phi_1(x), \Phi_1(y) \rangle + \langle \Phi_2(x), \Phi_2(y) \rangle, \tag{22}$$

which clearly satisfies the properties of an inner product.

For the multiplication,

$$K_3(x, y) = (K_1K_2)(x, y) = \langle \Phi_1(x), \Phi_1(y) \rangle \langle \Phi_2(x), \Phi_2(y) \rangle. \tag{23}$$

Let us denote $\varphi_i^1(x)$ and $\varphi_i^2(x)$, $\forall i = 1, 2, \ldots$ the (possibly infinite) components of the feature map Φ_1 and Φ_2 respectively. Then, we can write the inner product as

$$\langle \Phi_1(x), \Phi_1(y) \rangle \langle \Phi_2(x), \Phi_2(y) \rangle = \sum_{i=1}^{\infty} \varphi_i^1(x)\varphi_i^1(y) \sum_{j=1}^{\infty} \varphi_j^2(x)\varphi_j^2(y) = \sum_{i,j=1}^{\infty} \varphi_i^1(x)\varphi_j^2(x)\varphi_i^1(y)\varphi_j^2(y), \tag{24}$$

so the feature map of the product would be the one having as components $\varphi_{ij}^3(x) = \varphi_i^1(x)\varphi_j^2(x)$. Properties of the inner product:

- $\langle \Phi_3(x), \Phi_3(y) \rangle = \sum_{i,j=1}^{\infty} \varphi_i^1(x)\varphi_j^2(x)\varphi_i^1(y)\varphi_j^2(y) = \sum_{i,j=1}^{\infty} \varphi_i^1(y)\varphi_j^2(y)\varphi_i^1(x)\varphi_j^2(x) = \langle \Phi_3(y), \Phi_3(x) \rangle$.
- $\langle a\Phi_3(x)+b\Phi_3(y), \Phi_3(z) \rangle = \sum_{i,j=1}^{\infty} (a\varphi_i^1(x)\varphi_j^2(x)+b\varphi_i^1(y)\varphi_j^2(y))\varphi_i^1(z)\varphi_j^2(z) = a \sum_{i,j=1}^{\infty} \varphi_i^1(x)\varphi_j^2(x)\varphi_i^1(z)\varphi_j^2(z)+ b \sum_{i,j=1}^{\infty} \varphi_i^1(y)\varphi_j^2(y)\varphi_i^1(z)\varphi_j^2(z) = a\langle \Phi_3(x), \Phi_3(z) \rangle + b\langle \Phi_3(y), \Phi_3(z) \rangle$.
- $0 = \langle \Phi_3(x), \Phi_3(x) \rangle = \sum_{i,j=1}^{\infty} (\varphi_i^1(x))^2(\varphi_j^2(x))^2$, since it is a sum of positive numbers, in order to be 0 all the terms have to be 0. That will happen if and only if $x = 0$.

Problem 16: Write a program that generates n independent pairs of random variables (X_i, Y_i) such that $\mathbb{P}\{Y_i = 0\} = \mathbb{P}\{Y_i = 1\} = \frac{1}{2}$ and, conditionally on $Y_i = 0$, X is multivariate normal with mean $(0, 0, \ldots, 0)$ and identity covariance matrix, while, conditionally on $Y_i = 1$, X is multivariate normal with mean $(1, 1, 0, 0, \ldots, 0)$ and identity covariance matrix. Train a decision-tree classifier that greedily splits each cell by minimizing the number of misclassified points until it has k cells and assigns a majority vote to each cell.

- **Test the performance of the classifier on independent test data for a wide range of the parameters n, d, and k.**
- **Implement bagging for the decision-tree classifier above (by training the classifier of many subsamples and taking a majority vote) and, again, test its performance for a wide range of the parameters n, d, and k.**
- **Implement the random-subspace method that chooses two of the d components at random, builds the decision-tree classifier above, repeats this many times and takes a majority vote of the obtained classifiers. Test the performance for a wide range of the parameters n, d, and k.**

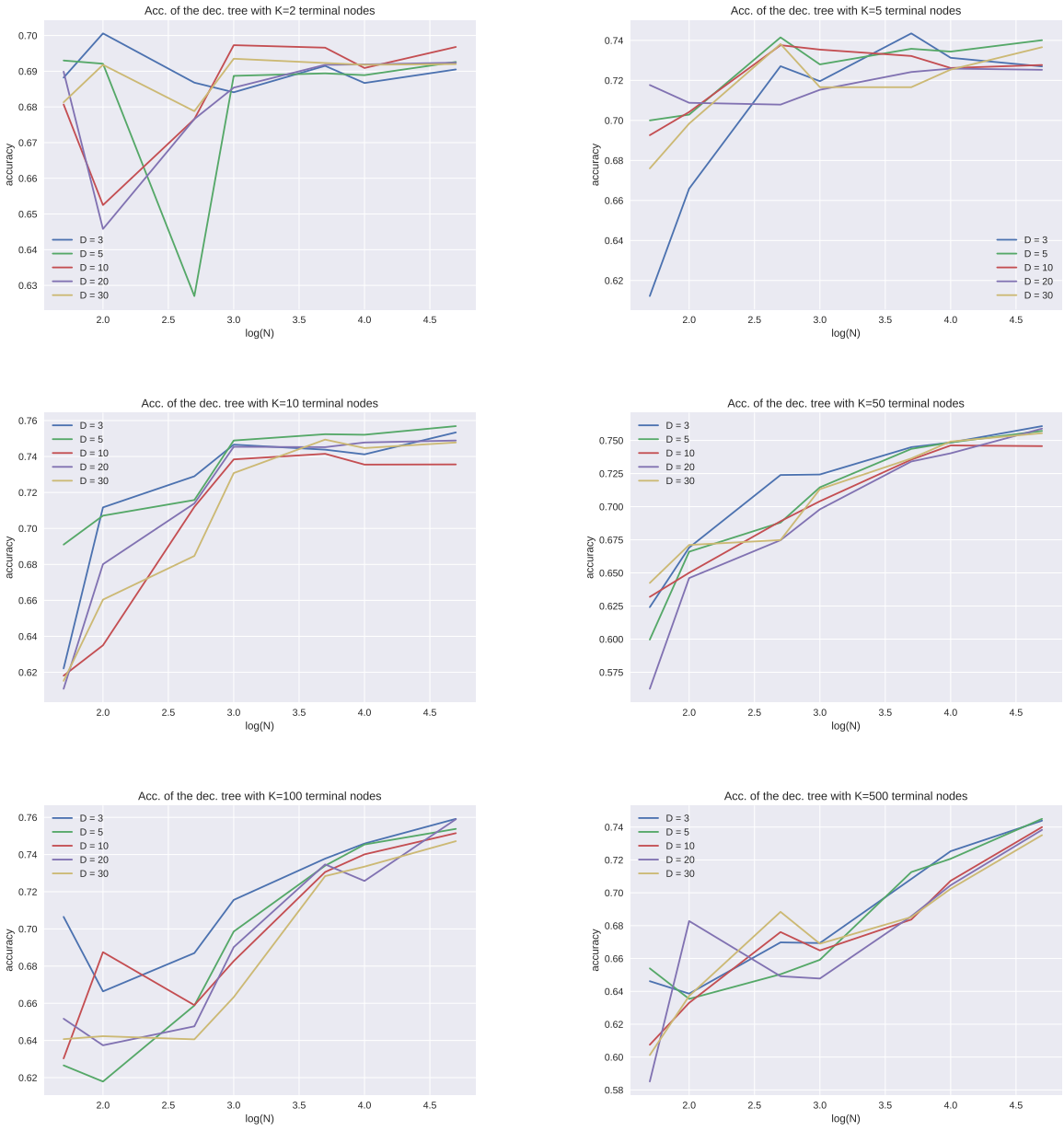

Figure 1: Decision tree classifier for $K = \{2, 5, 10, 50, 100, 500\}$ terminal nodes, with ranges of $d = \{3, 5, 10, 20, 30\}$ dimensions and $N = \{50, 100, 500, 1000, 5000, 10000, 50000\}$ sizes of training sets.

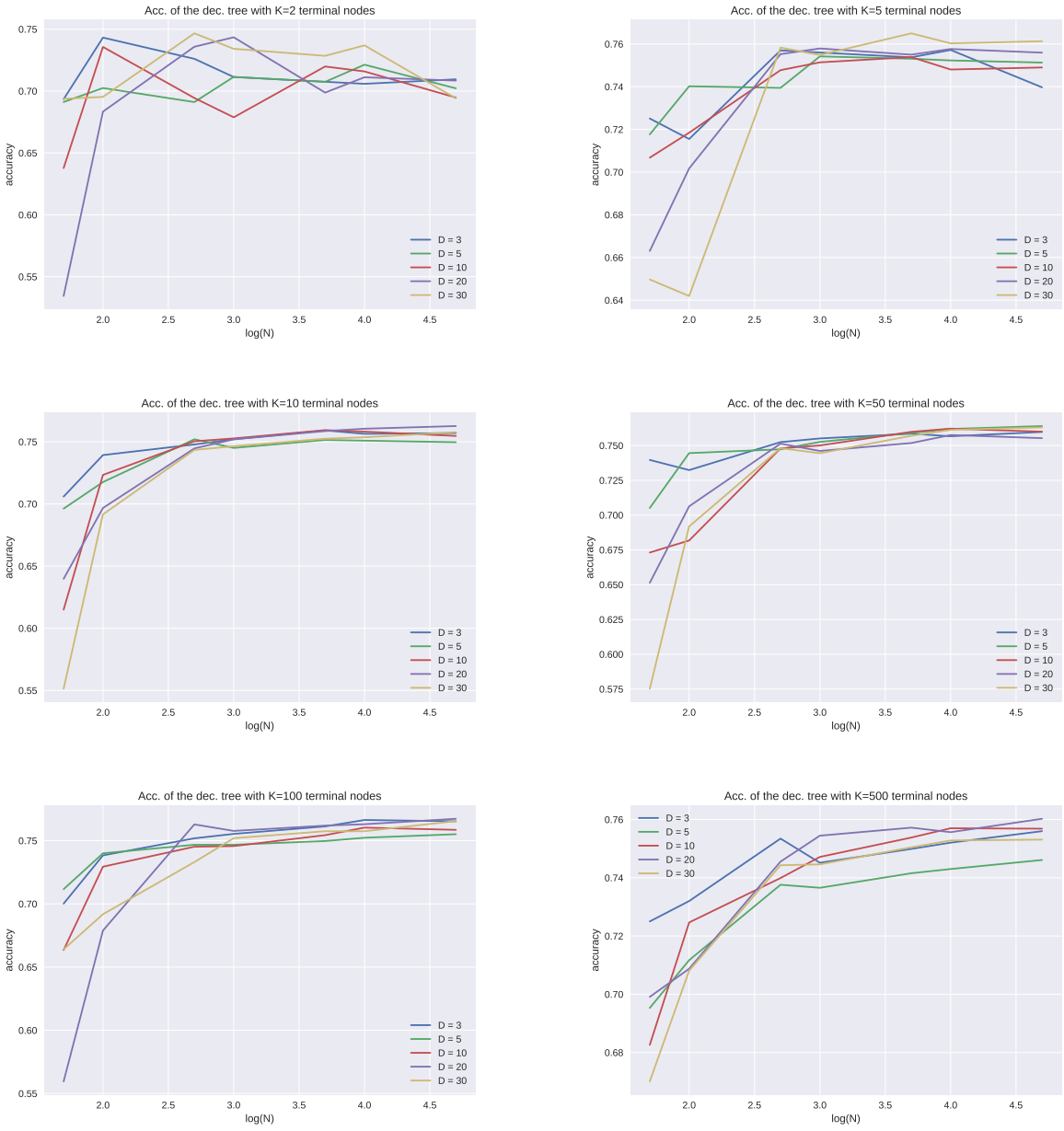

Figure 2: Decision tree classifier with bagging for $K = \{2, 5, 10, 50, 100, 500\}$ terminal nodes, with ranges of $d = \{3, 5, 10, 20, 30\}$ dimensions and $N = \{50, 100, 500, 1000, 5000, 10000, 50000\}$ sizes of training sets.

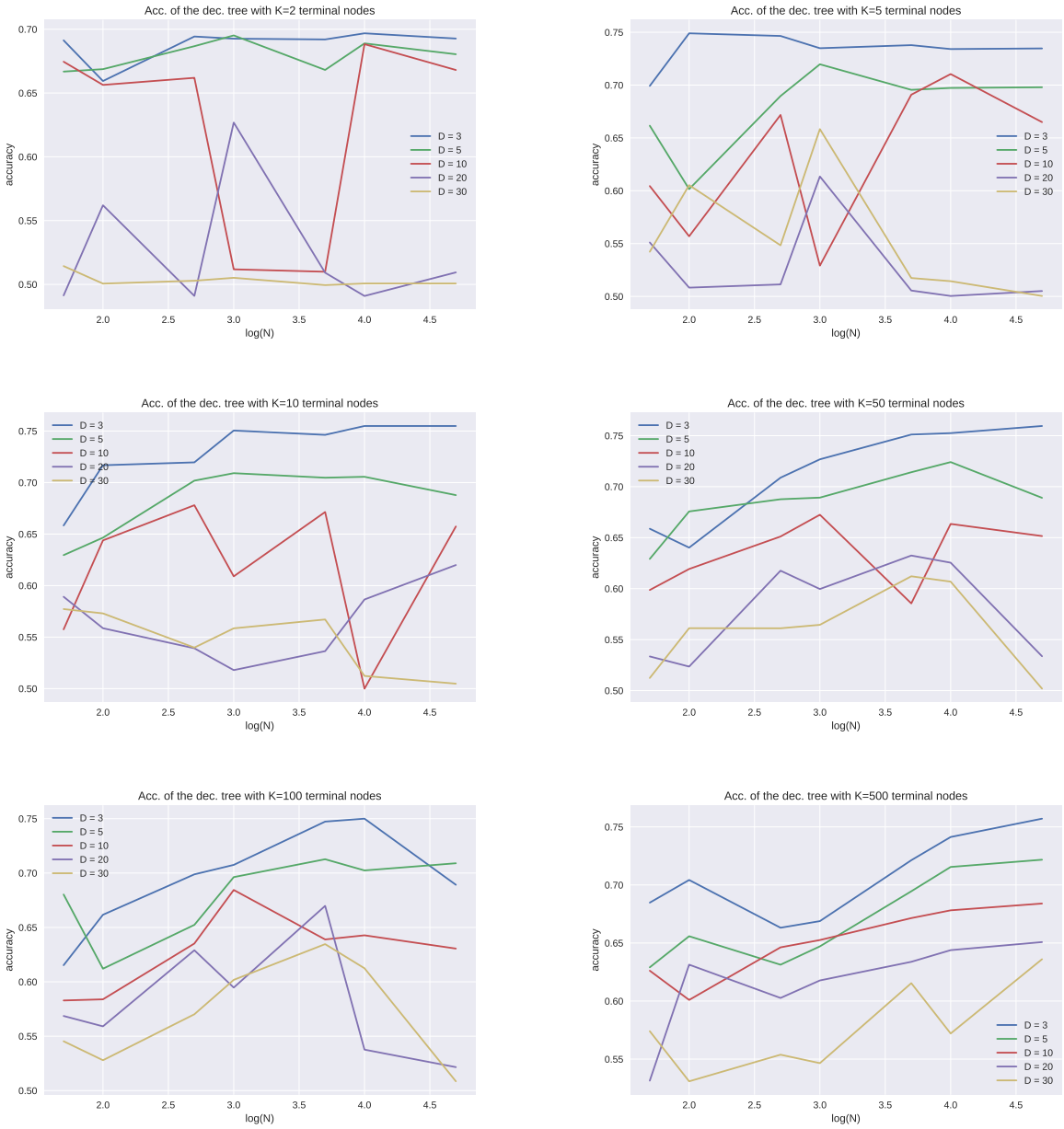

Figure 3: Decision tree classifier with random sub-sampling of two columns for $K = \{2, 5, 10, 50, 100, 500\}$ terminal nodes, with ranges of $d = \{3, 5, 10, 20, 30\}$ dimensions and $N = \{50, 100, 500, 1000, 5000, 10000, 50000\}$ sizes of training sets.

Appendix

Ex 16:

```
import sklearn as sk
import numpy as np
from numpy import random as rand
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import math as mat
import pylab



#Generate the data
def gen_data(n,d):
Y=rand.binomial(1,1/2,n)

X=np.zeros((n,d))
cov=np.identity(d)
for i in range(0,n):
if Y[i]==0:
vmean=[0]*d
else:
if d<=2:
vmean=[1]*d
else:
vmean=[1,1]+[0]*(d-2)
X[i]=rand.multivariate_normal(vmean,cov,1)
return X,Y



#Decision tree
m=10000
ns = [50,100,500,1000,5000,10000,50000]
logns = [mat.log(x,10) for x in ns]
ds = [3, 5, 10, 20, 30]
ks=[2,5,10,20,50,100,500]
j=0
for k in ks:

scores_list = []

for d in ds:
```

```
accuracy = np.zeros(len(ns))
X_test,Y_test=gen_data(m,d)

for i in range(len(ns)):

n=ns[i]
X_train,Y_train=gen_data(n,d)
dec_tree = DecisionTreeClassifier(max_leaf_nodes=k)
dec_tree.fit(X_train,Y_train)
Y_pred = dec_tree.predict(X_test)
accuracy[i] = 1-sum(abs(Y_test - Y_pred))/m

scores_list.append(accuracy)

plt.figure(j)
j+=1
for z in range(len(ds)):

plt.plot(logns, scores_list[z], label = 'D = %s'%ds[z])

plt.xlabel('log(N)')
plt.ylabel('accuracy')
plt.legend(loc="upperright")
plt.title('Acc. of the dec. tree with K='+str(k)+' terminal nodes')

pylab.savefig('/home/roger/Desktop/BGSE/14D005 Machine Learning/problemsets/Proble
```

```
#Bagging

def bagging_dec_tree(train_set,test_set,max_leaf_nodes,times_bag,size_bag):
dec_tree=DecisionTreeClassifier(max_leaf_nodes=max_leaf_nodes)
X_test=test_set[:,:(-1)]
Y_test=test_set[:,(-1)]
Y_array=Y_test
for i in range(times_bag):
bag=train_set[rand.randint(len(train_set),size=size_bag)]
bag_X=bag[:,:(-1)]
bag_Y=bag[:,(-1)]
dec_tree.fit(bag_X,bag_Y)
Y_array=np.column_stack((Y_array,dec_tree.predict(X_test)))

Y_pred=np.median(Y_array[:,1:],axis=1)
return Y_pred


m=10000
```

```
ns = [50,100,500,1000,5000,10000,50000]
logns = [mat.log(x,10) for x in ns]
ds = [3, 5, 10, 20, 30]
ks=[2,5,10,20,50,100,500]
num_bags=100
j=0
for k in ks:

scores_list = []

for d in ds:

accuracy = np.zeros(len(ns))
X_test,Y_test=gen_data(m,d)
test_set=np.column_stack((X_test,Y_test))

for i in range(len(ns)):

n=ns[i]
X_train,Y_train=gen_data(n,d)
train_set=np.column_stack((X_train,Y_train))
Y_pred = bagging_dec_tree(train_set,test_set,k,num_bags,mat.floor(0.2*n))
accuracy[i] = 1-sum(abs(Y_test - Y_pred))/m

scores_list.append(accuracy)

plt.figure(j)
j+=1
for z in range(len(ds)):

plt.plot(logns, scores_list[z], label = 'D = %s'%ds[z])

plt.xlabel('log(N)')
plt.ylabel('accuracy')
plt.legend(loc="upperright")
plt.title('Acc. of the dec. tree with K='+str(k)+' terminal nodes')

pylab.savefig('/home/roger/Desktop/BGSE/14D005 Machine Learning/problemsets/Proble



#random sub-space
def rsubspace_dec_tree(train_set,test_set,max_leaf_nodes,times_rspace):
dec_tree=DecisionTreeClassifier(max_leaf_nodes=max_leaf_nodes)
X_test=test_set[:,:(-1)]
Y_test=test_set[:,(-1)]
train_X=train_set[:,:(-1)]
train_Y=train_set[:,(-1)]
```

```
Y_array=Y_test
for i in range(times_rspace):
subid=rand.choice(len(train_X[0]),2,replace=False)
subsp_train=train_X[:,subid]
dec_tree.fit(subsp_train,train_Y)
subsp_test=X_test[:,subid]
Y_array=np.column_stack((Y_array,dec_tree.predict(subsp_test)))

Y_pred=np.median(Y_array[:,1:],axis=1)
return Y_pred



m=10000
ns = [50,100,500,1000,5000,10000,50000]
logns = [mat.log(x,10) for x in ns]
ds = [3, 5, 10, 20, 30]
ks=[2,5,10,20,50,100,500]
num_subsp=100
j=0
for k in ks:

scores_list = []

for d in ds:

accuracy = np.zeros(len(ns))
X_test,Y_test=gen_data(m,d)
test_set=np.column_stack((X_test,Y_test))

for i in range(len(ns)):

n=ns[i]
X_train,Y_train=gen_data(n,d)
train_set=np.column_stack((X_train,Y_train))
Y_pred = rsubspace_dec_tree(train_set,test_set,k,num_subsp)
accuracy[i] = 1-sum(abs(Y_test - Y_pred))/m

scores_list.append(accuracy)

plt.figure(j)
j+=1
for z in range(len(ds)):

plt.plot(logns, scores_list[z], label = 'D = %s'%ds[z])

plt.xlabel('log(N)')
plt.ylabel('accuracy')
```

```
plt.legend(loc="upperright")
plt.title('Acc. of the dec. tree with K='+str(k)+' terminal nodes')

pylab.savefig('/home/roger/Desktop/BGSE/14D005 Machine Learning/problemsets/Problem
```