Question 1)

27/April/2018 Machine Learning HW4 Solutions.

(a) without using the calculator

$D_1 = 3+, 5-$ & $D2 = 6+, 2-$

$$\text{Entropy}(D_1) = -\frac{3}{8}\log\frac{3}{8} - \frac{5}{8}\log\frac{5}{8}$$

$$\text{\& Entropy}(D2) = -\frac{6}{8}\log\frac{6}{8} - \frac{2}{8}\log\frac{2}{8}$$

Here entropy $(D_1) >$ entropy $(D2)$.
This makes sense as there is more uncertainity in D1 than in D2 (3+, 5- vs 6+, 2-)

(b) $$\text{Entropy}(S) = -\frac{3}{6}\log\frac{3}{6} - \frac{3}{6}\log\frac{3}{6}$$

$$= -\log\frac{3}{6} = -\log\frac{1}{2} = \log_2 2 = 1 \text{ —— (i)}$$

$$\text{Entropy}(S_{x_1} = T) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} \text{ —— (ii)}$$

$$\text{Entropy}(S_{x_1} = F) = -\frac{1}{3}\log\frac{1}{3} + \frac{2}{3}\log\frac{2}{3} \text{ —— (iii)}$$

$$\text{Information gain} = \text{Entropy}(S) - \sum_{v \in V} \frac{|S_v|}{|S|}\text{Entropy}(S_v)$$

$$\cancel{\Rightarrow (i) - ((ii) + (iii))}$$

$$\cancel{\Rightarrow 1 + \left(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3} + \frac{2}{3}\left(\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right)\right)}$$

$$= \cancel{\text{deg}}\ 1 - \left[\frac{3}{6}\left(\frac{-2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right) + \frac{3}{6}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right)\right]$$

After solving

$$= 1 + \frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}$$

$$= 0.0817 \quad] \text{ Ans.}$$

(c) Decision tree algorithm to produce ~~test~~ 0% training error. is as follows.

first calculating the entropy of each attribute & before splitting

for x_1

$$E(S_{x_1} = \text{True}) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} = 0.918$$

$$E(S_{x_1} = \text{False}) = -\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3} = 0.91$$

for x_2

$$E(S_{x_2} = T) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$$

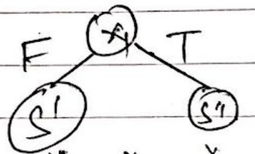
$$E(S_{x_2} = f) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$$

for x_3

$$E(S_{x_3} = T) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$$

$$E(S_{x_3} = F) = -\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4} = 1$$

$\therefore$ we split on the 'x_1' attribute.



S' :

	x_1	x_2	x_3	y
$x^{(1)}$	T	F	F	+
$x^{(3)}$	T	T	F	−
$x^{(4)}$	T	F	T	+

S'' :

	x_1	x_2	x_3	y
$x^{(2)}$	F	T	F	+
$x^{(5)}$	F	F	T	−
$x^{(6)}$	F	F	F	−

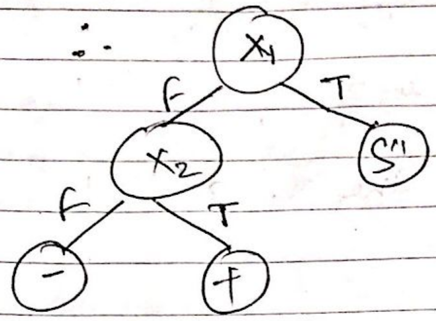
Now, $\text{Entropy}(S'_{x_2} = \text{True}) = -\frac{0}{1}\log\frac{0}{1} - \frac{1}{1}\log\frac{1}{1} = 0$

$\text{Entropy}(S'_{x_2} = \text{False}) = -\frac{2}{2}\log\frac{2}{2} - \frac{0}{2}\log\frac{0}{2} = 0$

$\text{Entropy}(S'_{x_3} = \text{True}) = -\frac{1}{1}\log\frac{1}{1} - \frac{0}{1}\log\frac{0}{1} = 0$

$\text{Entropy}(S'_{x_3} = \text{false}) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$

Now, we split on x_2



For S''

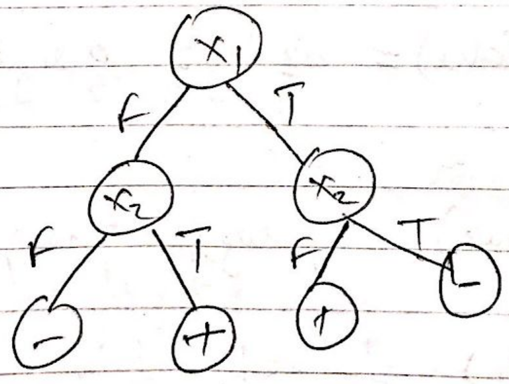
$$\text{Entropy}(S''_{x_2=T}) = -\frac{0}{1}\log\frac{0}{1} + \frac{1}{1}\log\frac{1}{1} = 0$$

$$\text{Entropy}(S''_{x_2=F}) = -\frac{2}{2}\log\frac{2}{2} + \frac{0}{2}\log\frac{0}{2} = 0$$

$$\text{Entropy}(S''_{x_3=T}) = -\frac{0}{1}\log\frac{0}{1} - \frac{1}{1}\log\frac{1}{1} = 0$$

$$\text{Entropy}(S''_{x_3=F}) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$$

Again, split on x_2



Ans.

(d) $H(X) = -\frac{3}{6}\log\frac{3}{6} - \frac{3}{6}\log\frac{3}{6} = 1$

$\Rightarrow H(Y|X) =$

for true $= \frac{1}{2}\left(-\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right)$

Illy for false $= \frac{1}{2}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right)$

Now,

$H(X) - H(Y|X) =$ mutual Information

$= 1 + \frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3} = 0.0817$

Ans

e)

The 9 digit driver license number will have higher gain will be higher compared to the other because the second term in the information gain formula will be smaller.

It is a bad idea to use the driving license number for information gain, because we want to minimize the entropy (maximize information gain) and splitting based on the first attribute does not decrease the entropy much as the second attribute. Also every license will have a different number and not all the combinations are used, thus it will not generalise well and overfit.

Question 2)

Q2

(a) The pseudocode use stochastic gradient descent.
there are 500 egs : , so for every eg in the dataset, weights are adjusted ~~on~~ once
so in one epoch, weights are updated 500 times
$\therefore$ In 100 epochs, weights will be upgraded

$100 \times 500 = 50{,}000$ times] Ans.

(b) ~~For...~~

(i)

$$\Delta v_{ih} = -\eta \frac{\partial E(w,v|X)}{\partial v_{ih}}$$

$$= -\eta \frac{\partial}{\partial v_{ih}} \frac{1}{2}\left[(r_1-y_1)^2+(r_2-y_2)^2+\delta(r_3-y_3)^2\right]$$

for $i=1$ & $i=2$

$$\Delta v_{1h} = -\eta(r_1-y_1) z_h$$

$$\Delta v_{2h} = -\eta(r_2-y_2) \dot{z}_h$$

for $i=3$.

$$\Delta v_{3h} = 5\eta(r_3-y_3)z_h$$

(ii)

$$\Delta w_{hj} = -\eta\frac{\partial E}{\partial w_{hj}} = -\eta\sum_t \frac{\partial E^{(t)}}{\partial y^{(t)}}\frac{\partial y^{(t)}}{\partial z_h^{(t)}}\frac{\partial z_h^{(t)}}{\partial w_{hj}}$$

$$= \eta\sum_t\left[\sum_i (r_i^t-y_i^t)v_{ih}\right]z_h^t(1-z_h^t)x_j^t$$

$\therefore$ for the new Err function.

$$\Delta w_{hj} = \eta\sum_t\left[(r_1^t-y_1^t)+(r_2^t-y_2^t)+(r_3^t-y_3^t)\right]v_h z_h^t(1-z_h^t)x$$

Ans

Question 3)

A) We should use NNRZeroOne instead of NNRK, beacuse NNRK is used for regression and it might produce output not within the boundaries of [0,1]

B) We should not use NNCK instead of NNZeroOne, because we are interested in doing regression and obtain K different ouputs for every input. If we use NeuralNetCK, the sum of the K output values will be 1 as they represent probabilities. Because of this being a regression problem and not a classification one, NNZeroOne is most suitable.

C)

epoch 10: err 0.175075
epoch 20: err 0.168263
epoch 30: err 0.166156
epoch 50: err 0.162933
The error is decreasing.

Error on the final epoch
epoch 4999: err 0.000030

y0 = [[0.00703212]] y1 = [[0.99271479]] y2 = [[0.99265531]] y3 = [[0.00899179]]

ii)
epoch 0: err 0.112072
epoch 9999: err 0.000012

y0 = [[9.91966531e-01 1.47821022e-07 7.23203838e-08 6.50772946e-03
5.31736594e-11 6.99113147e-03 1.04577555e-06 4.66374014e-03]]

y1 = [[2.08993607e-07 9.92216490e-01 3.52264933e-08 9.75060047e-11
6.77437251e-03 6.89870534e-03 1.96352193e-06 4.89800321e-03]]

y2 = [[1.03094610e-06 8.22719895e-07 9.91547095e-01 7.13061616e-03
7.57335952e-03 1.35441284e-10 5.24051984e-07 4.81343215e-03]]

y3 = [[4.62402066e-03 5.29286647e-12 4.50614855e-03 9.90795079e-01
8.83816411e-07 3.59849712e-07 8.19804466e-03 2.62445307e-08]]

y4 = [[9.23311545e-12 4.54073060e-03 4.23385328e-03 1.69270005e-06
9.90613702e-01 2.16974882e-07 8.56402667e-03 5.57427738e-08]]

y5 = [[4.52666833e-03 4.49026217e-03 1.88240308e-13 2.87285457e-07
1.71865614e-07 9.90585916e-01 8.36551703e-03 1.57570770e-07]]

y6 = [[1.15405174e-07 1.69116934e-07 1.41156240e-08 4.57298201e-03
4.11456616e-03 4.52928545e-03 9.87368532e-01 7.18040067e-13]]

y7 = [[5.72604120e-03 5.44784030e-03 6.75743745e-03 1.66537390e-06
1.95761315e-06 8.01135849e-07 3.10732539e-10 9.91143568e-01]]

iii)
epoch 0: err 0.030797 epoch 3999: err 0.000167

iv) The code is updating stochastic gradient descent as weights are being updated for each iteration of the training examples.

v)
The input values are scaled b/w -pi to pi wirth 0.1 interval.
Input is scaled between -1 to 1 dividing each value by pi

X = X/pi

The output is scaled
For every input, cosine is taken, 1 is added and then divided by 2.

X = (Cos(X) + 1) / 2

NeuralNetRZeroOne is designed for regression problems with K outputs where each output is either in 0 or 1 or anything in between.
Cos(x) has values from -1 to 1 , so this NN is not suitable for data unless it is first preprocessed and scaled.

Question 4)
a)

Q4

(a) $\begin{bmatrix} 0 & 14 \\ 2 & 3 \end{bmatrix}$ 28

$A = \begin{bmatrix} 0-\lambda & 14 \\ 2 & 3-\lambda \end{bmatrix}$

$\text{Det}(A) = (0-\lambda)(3-\lambda) - 14 \times 2 = 0$

$= (\lambda - 3)\lambda = 28$

$\lambda^2 - 3\lambda - 28 = 0$

Eigen Values $= \boxed{\lambda = 7 \ \& \ \lambda = -4}$

for 7

$\left[\begin{array}{cc|c} 0-7 & 14 & 0 \\ 2 & 3-7 & 0 \end{array}\right]$

Using Gauss-Jordan elimination

$\left[\begin{array}{cc|c} -7 & 14 & 0 \\ 2 & -4 & 0 \end{array}\right]$ $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$-7x + 14y = 0$

$2x - 4y = 0$

$x = 2y$

$-14y + 14y = 0$

$y = 0,$

$x = 0$

$x = 2y$

$\therefore$ Infinite Solutions.

let $y = t$, ~~they~~ then $x = 2t$.

L_2 Norm $= \sqrt{t^2 + 4t^2} = 1$

$= t^2 + 4t^2 = 1$

$= 5t^2 = 1$

$t^2 = \frac{1}{5}$

$= t = \pm \frac{1}{\sqrt{5}}$

$\therefore$ eigenvectors are of the form $= \begin{bmatrix} t \\ 2t \end{bmatrix}$

Now, for $\lambda = -4$

$A = \begin{bmatrix} 4 & 14 \\ 2 & 7 \end{bmatrix}$

$4x + 14y = 0$
$2x + 7y = 0$ $\Big| \quad x = \frac{-7}{2} y$

Again, ∞ no. of Solutions with $\lambda = -4$

L_2 Norm $= \sqrt{t^2 + \frac{49}{4} t^2} = 1$

$\Rightarrow t = \pm \frac{2}{\sqrt{53}}$

∞ no. of eigenvectors of the form

$$\begin{bmatrix} t \\ -\frac{7}{2}t \end{bmatrix}$$

d)

```
mat = np.array([[0,14],[2,3]])
eigenVal, eigenVec = np.linalg.eig(mat)
print eigenVal
print eigenVec
```

```
[-4.  7.]
[[-0.96152395 -0.89442719]
 [ 0.27472113 -0.4472136 ]]
```

Question 5:

a) and b)

Q5

$$A = \begin{bmatrix} 4 & 1 & 3 \\ 8 & 5 & 3 \\ 6 & 0 & 1 \\ 1 & 4 & 5 \end{bmatrix}$$

$S_1 = 4.75$

$S_2 = 2.5$

$S_3 = 3.$

$$B = \begin{bmatrix} -0.75 & -1.5 & 0 \\ 3.25 & 2.5 & 0 \\ 1.25 & -2.5 & -2 \\ -3.75 & +1.5 & 2 \end{bmatrix}$$

(b) $$\text{Covariance} = \frac{\sum_t (x_i^t - c_i) \times (x_j^t - c_j)}{N-1}$$

$$S_{1,2} = \sum_t (x_i^t - s_i)(x_j^t - s_j) \Big/ N-1$$

$$= \frac{(-0.75)(-1.5) + (3.25)(2.5) + (1.25)(-2.5) + (-3.75)(1.}{3}$$

$$= \cancel{(1.125 + 8.125 - 3.75 - 5.625) / 3}$$

$$= \cancel{-0.125/3}$$

$$= \cancel{-0.04166} \Big] \text{Ans.}$$

$$= \frac{0.5}{3} = \frac{1}{6} \Big] \text{Ans}$$

c)

```
B = np.array([[-0.75,-1.5,0],[3.25,2.5,0],[1.25,-2.5,-2],[-3.75,1.5,2]])
C = np.cov(B,rowvar = False)
print C
```

```
[[ 8.91666667  0.16666667 -3.33333333]
 [ 0.16666667  5.66666667  2.66666667]
 [-3.33333333  2.66666667  2.66666667]]
```

```
eigenVal, eigenVec = np.linalg.eig(C)
print eigenVal
```

```
[  0.07634809  10.55611233   6.61753958]
```

Largest Value = 10.55611233

d) Value of Z

```
pca = skd.PCA(n_components = 3)
skd.PCA.fit(pca,B)
W1 = pca.components_
W = W1.transpose()
Z = pca.transform(B)
print Z
```

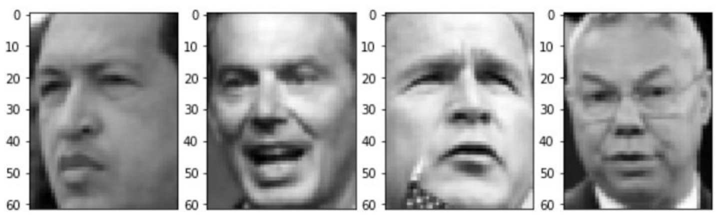
```
[[ 0.33956422 -1.598666    0.3761159 ]
 [-2.31212882  3.38610763  0.02890423]
 [-2.49664663 -2.34865846 -0.25111557]
 [ 4.46921123  0.56121683 -0.15390457]]
```

Question 6)

Question 6

In [46]:
```
lfw_people = fetch_lfw_people(min_faces_per_person=70)
n_samples, h, w = lfw_people.images.shape
npix = h*w
fea = lfw_people.data
def plt_face(x):
    global h,w
    plt.imshow(x.reshape((h, w)), cmap=plt.cm.gray)
    plt.xticks([])
plt.figure(figsize=(10,20))
nplt = 4
for i in range(nplt):
    plt.subplot(1,nplt,i+1)
    plt_face(fea[i])
plt.show()
```

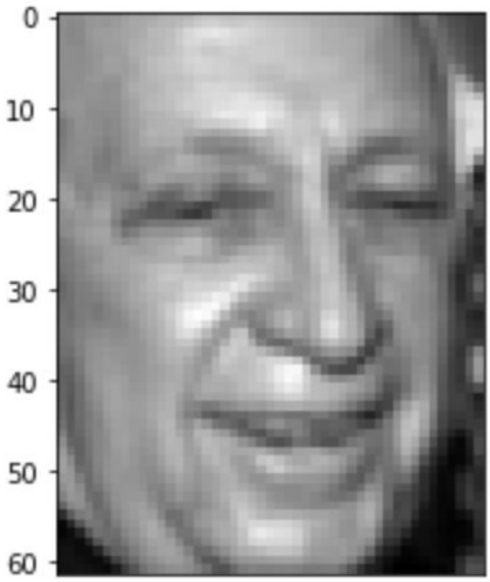
```
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
```



This is the replicated code from the assignment pdf and its result.

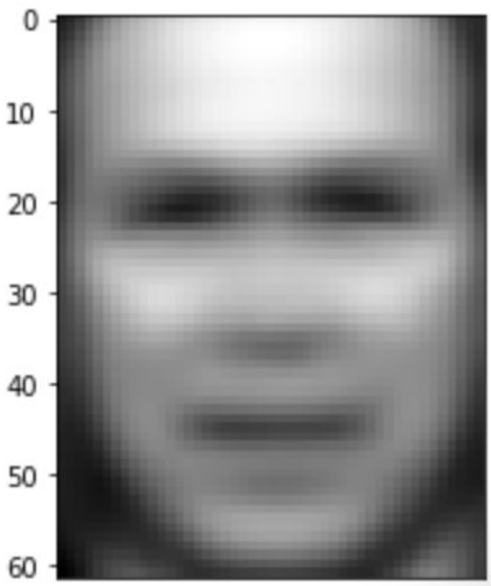
The fifth face.
Along with the mean face.

```
plt_face(fea[4])
plt.show()
```




```
mean_face = np.mean(fea,axis = 0)
plt_face(mean_face)
print mean_face.shape
```

```
(2914,)
```



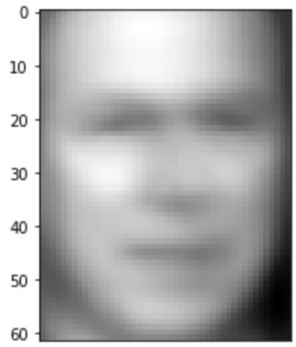
Using the PCA code. The values of Z are.

```
]: pca = skd.PCA(n_components = 6)
   pca.fit(fea)
   W1 = pca.components_
   W = W1.transpose()
   Z = pca.transform(fea)
   print Z[4:5]

   [[ 547.9161377   811.66046143  592.57885742  268.08358765  171.35772705
      -17.44544983]]
```

```
]: reconstruction = np.matmul(Z,np.transpose(W)) + mean_face #np.transpose(mean_face)
```

```
]: plt_face(reconstruction[4])
```



For 100 iterations.

```
: pca = skd.PCA(n_components = 100)
  pca.fit(fea)
  W1 = pca.components_
  W = W1.transpose()
  Z = pca.transform(fea)
  reconstruction = np.matmul(Z,np.transpose(W)) + mean_face
```

```
: plt_face(reconstruction[4])
```