EE 379K: Data Science Lab

Lab 1 - 9/11/17

Rachel Chen and Kevin Yee

rjc2737 and kjy252

Question 1:

1. Create 1000 samples from a Gaussian distribution with mean -10 and standard deviation 5. Create another 1000 samples from another independent Gaussian with mean 10 and standard deviation 5. (a) Take the sum of 2 these Gaussians by adding the two sets of 1000 points, point by point, and plot the histogram of the resulting 1000 points. What do you observe? (b) Estimate the mean and the variance of the sum.

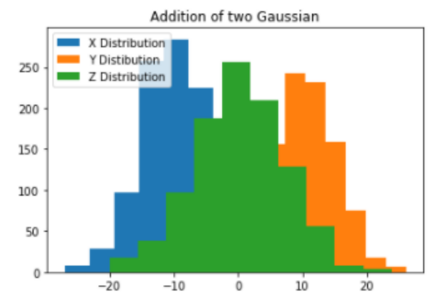
In [2]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [3]:
```
#Problem 1
X = np.random.normal(-10, 5, 1000)
Y = np.random.normal(10, 5, 1000)
Z = np.add(X,Y)
#https://stackoverflow.com/questions/38747612/whats-the-difference-between-numpy-adda-b-and-ab
```

In [4]:
```
plt.title("Addition of two Gaussian")

plt.hist(X, label ='X Distribution')
plt.hist(Y, label = 'Y Distibution')
plt.hist(Z, label = 'Z Distribution')
plt.legend(loc = 'upper left')
```

Out[4]:
```
<matplotlib.legend.Legend at 0x2136358>
```

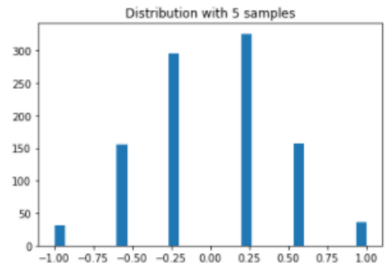


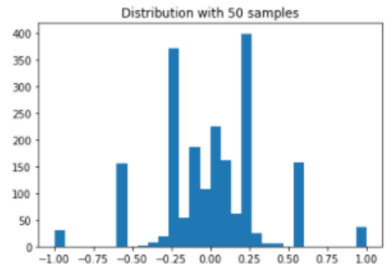
Question 2:

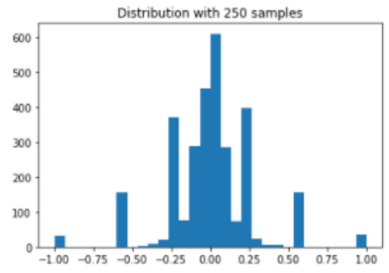
1. Central Limit Theorem. Let Xi be an iid Bernoulli random variable with value {-1,1}. Look at the random variable Zn = 1 n PXi . By taking 1000 draws from Zn, plot its histogram. Check that for small n (say, 5-10) Zn does not look that much like a Gaussian, but when n is bigger (already by the time n = 30 or 50) it looks much more like a Gaussian. Check also for much bigger n: n = 250, to see that at this point, one can really see the bell curve.

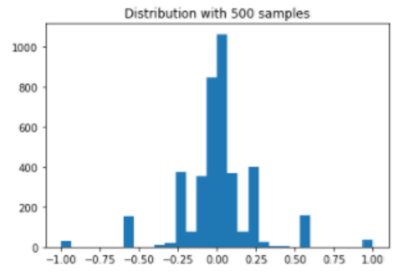
In [6]:
```
#Problem 2

#n-trials
numsamples = [5,50,250,500]
trials = []
for k in range(len(numsamples)):
    for j in range(1000):
        x = np.random.binomial(1, 0.5, numsamples[k]); #bernoulli
        for i in range(len(x)):
            if x[i] == 0: x[i] = -1 #spans -1 to 1
        trials.append(float(sum(x))/numsamples[k]) #average of sum
    plt.figure(k)
    plt.title("Distribution with " + str(numsamples[k]) + " samples")
    plt.hist(trials, bins = 30)
```









Question 3:

1. Estimate the mean and standard deviation from 1 dimensional data: generate 25,000 samples from a Gaussian distribution with mean 0 and standard deviation 5. Then estimate the mean and standard deviation of this gaussian using elementary numpy commands, i.e., addition, multiplication, division (do not use a command that takes data and returns the mean or standard deviation).

In [7]:

```
k = np.random.normal(0, 5, 25000)
mean = k.sum()/len(k)
std = np.sqrt(np.sum(np.square(k-mean))/len(k))
print mean
print std
```

```
0.0149204618984
5.00275383556
```

Question 4:

1. Estimate the mean and covariance matrix for multi-dimensional data: generate 10,000 samples of 2 dimensional data from the Gaussian distribution Then, estimate the mean and covariance matrix for this multi-dimensional data using elementary numpy commands, i.e., addition, multiplication, division (do not use a command that takes data and returns the mean or standard deviation).

In [8]:

```
mean = [-5, 5]
cov = [[20, .8], [.8, 30]]
x = np.random.multivariate_normal(mean, cov, 10000)
print x
mean = np.sum(x, axis = 0)/len(x)

print "The mean matrix is",
print mean

#https://stackoverflow.com/questions/27448352/how-numpy-cov-function-is-implemented
```

```
[[ -8.37707136   8.48436319]
 [ -7.13527547   4.65723797]
 [ -0.04755268  10.57112753]
 ...,
 [ -4.25173414   7.16760325]
 [-10.03460056   9.46320271]
 [ -8.60942973   0.85905407]]
The mean matrix is [-4.98667696  4.97735342]
```

For reference, the Covariance matrix is defined below:

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{pmatrix}.$$

In [9]:

```
varX = np.sum(np.square(x[:,0]-mean[0]))/len(x)
varY = np.sum(np.square(x[:,1]-mean[1]))/len(x)
CovXY = np.sum((x[:,0]-mean[0]) * (x[:,1]-mean[1]))/len(x)

print "The covariance matrix is"
print np.array([[varX, CovXY], [CovXY, varY]])
```

```
The covariance matrix is
[[ 19.8701742    0.43441176]
 [  0.43441176  29.58796425]]
```

Question 5:

Each row is a patient and the last column is the condition that the patient has. Do data exploration using Pandas and other visualization tools to understand what you can about the dataset. For example:

(a) How many patients and how many features are there?

```
452 patients.
280 features.
```

(b) What is the meaning of the first 4 features? See if you can understand what they mean.

```
Feature 1: Age
Feature 2: Sex
Feature 3: Systolic blood presure
Feature 4: Diastolic blood presssure
```

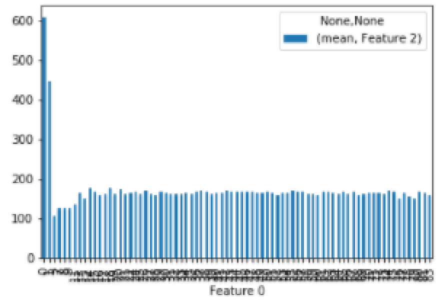
In [10]:

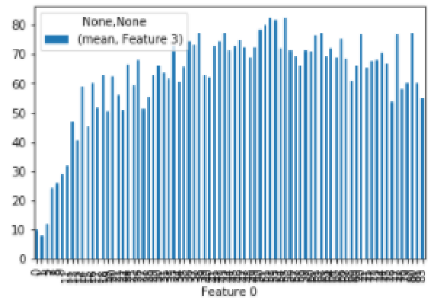
```
df0 = pd.read_csv('PatientData.csv')
colNames = [];
#define column names
for i in range(len(df0.columns)):
    colNames.append("Feature " + str(i))
#import into pandas dataframe
df0 = pd.read_csv('PatientData.csv',names = colNames)

#take the average "blood pressure" for each age and plot
dfpiv = pd.pivot_table(df0,index=['Feature 0'], values = ['Feature 2'], aggfunc=[np.mean])
dfpiv.plot(kind = 'bar')
dfpiv = pd.pivot_table(df0,index=['Feature 0'], values = ['Feature 3'], aggfunc=[np.mean])
dfpiv.plot(kind = 'bar')
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xaec3f28>
```





Looking at the bar graph above, the x axis is of age and y axis is of Feature 3. We can infer that Feature 3 correlates to blood pressure. Younger people <10 have lower blood pressure and older people have higher blood pressure.

Our observation is made in regards to the normal healthy 120/80 measurement for blood pressure

(c) Are there missing values? Replace them with the average of the corresponding feature column

Yes. We can verify that we correctly replaced all the missing values by looking at Column 13, where originally many values were missing. These have been replaced with the average of the corresponding feature column

In [11]:

```
#replace ? with nan so the mean will ignore nan
newdata = df0.replace('?',np.nan)

#replace all missing values by iterating through columns
for col in newdata.columns:
    newdata[col] = newdata[col].fillna(newdata[col].map(float).mean())

#print to show soln is correct
newdata.ix[:,'Feature 13':].head()
```

Out[11]:

	Feature 13	Feature 14	Feature 15	Feature 16	Feature 17	Feature 18	Feature 19	Feature 20	Feature 21	Feature 22	...	Feature 270	Feature 271	Feature 272	Feature 273	Feature 274	Featur 27
0	-13.5921	63	0	52	44	0	0	32	0	0	...	0.0	9.0	-0.9	0.0	0.0	0.9
1	-13.5921	53	0	48	0	0	0	24	0	0	...	0.0	8.5	0.0	0.0	0.0	0.2
2	23	75	0	40	80	0	0	24	0	0	...	0.0	9.5	-2.4	0.0	0.0	0.3
3	-13.5921	71	0	72	20	0	0	48	0	0	...	0.0	12.2	-2.2	0.0	0.0	0.4
4	-13.5921	74.4634	0	48	40	0	0	28	0	0	...	0.0	13.1	-3.6	0.0	0.0	-0.1

5 rows × 267 columns

(d) How could you test which features strongly influence the patient condition and which do not?

Method 1

1. Create a correlation matrix using pandas .corr() function
2. Compare the correlation coefficient of all other columns in comparison to the "patient's condition column" (Last column)
3. Sort the data in descending order. The columns with the highest correlation coefficient will be most related to the patient's condition. (This is demonstrated in the code below)

Method 2: Because there are 280 features. We run into the 'curse of dimensionality.' We could alternatively run PCA (Principal Component Analysis) on our dataset to determine which features minimizes the mean squared distance between the original dataset to find out which features are indeed most important.

PCA is probably the most effective, however, considering this is Data Science Lab 1, I will tackle it with Method 1 (naiive solution).

List what you think are the three most important features.

The three most important features are

Feature 90

Feature 4

Feature 92

(Data shown below)

In [14]:

```
corr_matrix = newdata.corr()
corr_matrix.iloc[:,-1].sort_values(ascending=False).head()
```

Out[14]:

```
Feature 279    1.000000
Feature 90     0.368876
Feature 4      0.323879
Feature 92     0.313982
Feature 102    0.282523
Name: Feature 279, dtype: float64
```

Written Questions

1. Consider two random variables X,Y that are not independent. Their probabilities of are given by the following table:

	X=0	X=1
Y=0	1/4	1/4
Y=1	1/6	1/3

(a) What is the probability that $X = 1$?
(b) What is the probability that $X = 1$ conditioned on $Y = 1$?
(c) What is the variance of the random variable X?
(d) What is the variance of the random variable X conditioned that $Y = 1$?
(e) What is $E[X^3 + X^2 + 3Y^7|Y = 1|]$?

1(a):

Marginal pmf from joint pmf:

$$P(X = x) = \sum_y P(X = x, Y = y)$$
$$\therefore P(X = 1) = \frac{1}{4} + \frac{1}{3} = \frac{7}{12}$$

1(b):

Conditional Probability:

$$P(X|Y) = \frac{P(X,Y)}{P(Y)}$$
$$\therefore P(X = 1|Y = 1) = \frac{1/3}{1/6 + 1/3}$$
$$P(X = 1|Y = 1) = \frac{2}{3}$$

1(c):

Bernoulli Equation

$$P(X = 1) = \frac{7}{12}$$
$$P(X = 0) = \frac{5}{12}$$
$$\therefore pq = \frac{7}{12} * \frac{5}{12} = \frac{35}{72}$$

1(d):

Bernoulli Equation

$$P(X = 1|Y = 1) = \frac{2}{3}$$
$$P(X = 1|Y = 0) = \frac{3}{7}$$
$$\therefore pq = \frac{2}{3} * \frac{1}{3} = \frac{2}{9}$$

1(e):

$$E[X^3 + X^2 + 3Y^7|Y = 1]$$
$$= E[X^3|Y = 1] + E[X^2|Y = 1] + 3 * E[Y^7|Y = 1]$$
$$= \frac{1}{3} + \frac{1}{3} + 3 * 1$$
$$= 3\frac{2}{3}$$

2. Consider the vectors $\mathbf{v}_1 = [1, 1, 1]$ and $\mathbf{v}_2 = [1, 0, 0]$. These two vectors define a 2-dimensional subspace of $\mathbb{R}^3$. Project the points $P1 = [3, 3, 3]$, $P2 = [1, 2, 3]$, $P3 = [0, 0, 1]$ on this subspace. Write down the coordinates of the three projected points. (You can use numpy or a calculator to do arithmetic if you want).

Let W be defined as the

$$Span\ (v_1, v_2)$$

```
In [ ]: def orthog(v1, v2, point):
            u1 = (np.dot(v1,point)/np.dot(v1,v1))*v1;
            u2 = (np.dot(v2,point))/np.dot(v2,v2)*v2;
            return u1 + u2;
```

```
In [ ]: v1 = [1, 1, 1];
        v2 = [1, 0, 0];
        P1 = [3, 3, 3];
        P2 = [1, 2, 3];
        P3 = [0, 0, 1];

        print("Point 1 projection", orthog(np.transpose(v1),np.transpose(v2),np.transpose(P1)))
        print("Point 2 projection", orthog(np.transpose(v1),np.transpose(v2),np.transpose(P2)))
        print("Point 3 projection", orthog(np.transpose(v1),np.transpose(v2),np.transpose(P3)))
```

3. Consider a coin such that probability of heads is 2/3. Suppose you toss the coin 100 times. Estimate the probability of getting 50 or fewer heads. You can do this in a variety of ways. One way is to use the Central Limit Theorem. Be explicit in your calculations and tell us what tools you are using in these.

Central Limit Theorem:

$$\frac{S_n - n * E[X]}{\sqrt{n}\theta}$$
$$P(Head) = \frac{2}{3}$$

vb
Let

$$X_i$$

be the random variable that a head is flipped. where X = 1 if heads and 0 otherwise. Then, let

$$S_n = X_1 + X_2 + \dots X_n$$

Variance =

$$\frac{2}{3} * \frac{1}{3}$$

Using CTL:

$$\frac{50 - (100 * \frac{2}{3})}{\sqrt{100} * \sqrt{2/9}}$$
$$P(Z_{100} < -3.54) = \phi(-3.54)$$

```
In [ ]: import scipy.stats
        scipy.stats.norm.cdf(-3.54)
```

$$P(Z_{100} < -3.54) = 0.0002$$

Probability of getting fewer than 50 heads is .02%