DSE 210

Homework 5

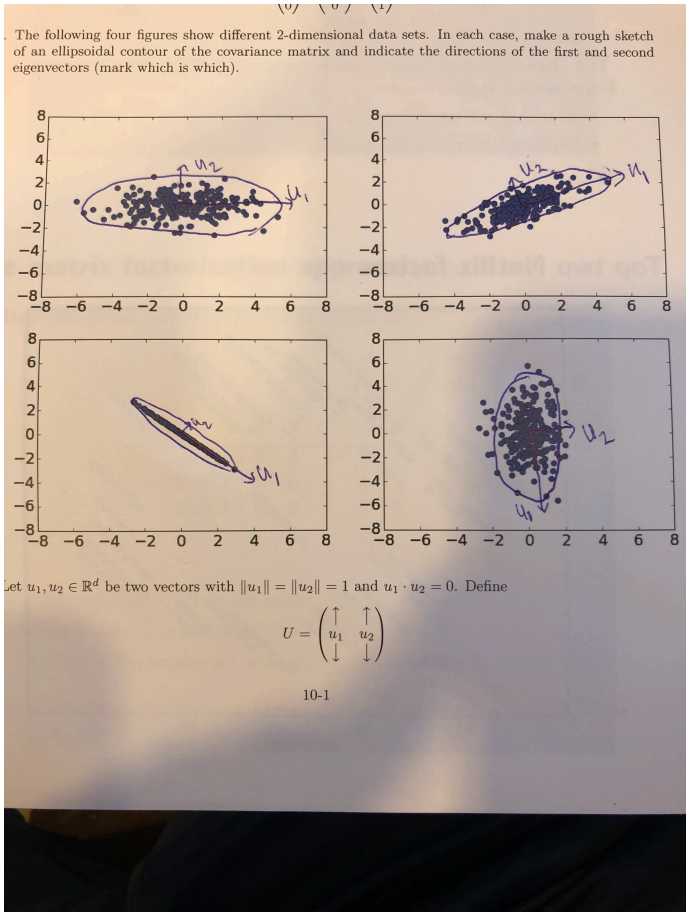
Kevin Kannappan
March 11, 2019

1 PCA and SVD

1.1 Worksheet 10

1. The following set of vectors do not form an orthonormal basis of $\mathbb{R}^3$. The reasoning for why they do not is, while they are orthogonal and the dot products between each vector pair is 0, the first two vectors are not unit vectors. To be an orthonormal set of vectors, the vectors need to be both unit vectors and orthogonal.

2. Drew a rough sketch of ellipsoidal contours of the covariance matrices and the directions of the first and second eigenvectors directly on the plots. They can be viewed in the image below:



5. Please refer to Jupyter notebook for Problem 5.

Day 5 Homework

March 11, 2019

1 Worksheet 10: PCA & SVD

Problem 5:
Begin by leveraging previous work:

In [6]:
```
# Imported Code from Professor:
from struct import unpack
import numpy as np
import pandas as pd
import matplotlib.pylab as plt

def loadmnist(imagefile, labelfile):

    # Open the images with gzip in read binary mode
    images = open(imagefile, 'rb')
    labels = open(labelfile, 'rb')

    # Get metadata for images
    images.read(4)  # skip the magic_number
    number_of_images = images.read(4)
    number_of_images = unpack('>I', number_of_images)[0]
    rows = images.read(4)
    rows = unpack('>I', rows)[0]
    cols = images.read(4)
    cols = unpack('>I', cols)[0]

    # Get metadata for labels
    labels.read(4)
    N = labels.read(4)
    N = unpack('>I', N)[0]

    # Get data
    x = np.zeros((N, rows*cols), dtype=np.uint8)  # Initialize numpy array
    y = np.zeros(N, dtype=np.uint8)  # Initialize numpy array
    for i in range(N):
        for j in range(rows*cols):
            tmp_pixel = images.read(1)  # Just a single byte
```

```
                tmp_pixel = unpack('>B', tmp_pixel)[0]
                x[i][j] = tmp_pixel
            tmp_label = labels.read(1)
            y[i] = unpack('>B', tmp_label)[0]

        images.close()
        labels.close()
        return (x, y)

    def displaychar(image):
        plt.imshow(np.reshape(image, (28,28)), cmap=plt.cm.gray)
        plt.axis('off')
        plt.show()
```

In [101]:
```
# Train
train_images = "/Users/kkannapp/Documents/DSE/DSE210-homework/day_3/train-images-idx3-
train_labels = "/Users/kkannapp/Documents/DSE/DSE210-homework/day_3/train-labels-idx1-
# Test
test_images = "/Users/kkannapp/Documents/DSE/DSE210-homework/day_3/t10k-images-idx3-ub
test_labels = "/Users/kkannapp/Documents/DSE/DSE210-homework/day_3/t10k-labels-idx1-ub

train_x,train_y = loadmnist(train_images,train_labels)
test_x,test_y = loadmnist(test_images,test_labels)
```

In [102]:
```
# Preview the data
train_x[0],train_y[0] # First entry is a 28 x 28 matrix of pixels, and is classified a
```

Out[102]:
```
(array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   3,  18,  18,  18,
        126, 136, 175,  26, 166, 255, 247, 127,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170, 253,
        253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253,
        253, 253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  18, 219, 253,
        253, 253, 253, 253, 198, 182, 247, 241,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         80, 156, 107, 253, 253, 205,  11,   0,  43, 154,   0,   0,   0,
```

```
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,  14,   1, 154, 253,  90,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0, 139, 253, 190,   2,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190, 253,  70,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,  81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,  45, 186, 253, 253, 150,  27,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,  16,  93, 252, 253, 187,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 249,
253, 249,  64,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  46, 130,
183, 253, 253, 207,   2,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39, 148,
229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114,
221, 253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  23,  66,
213, 253, 253, 253, 253, 198,  81,   2,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  18, 171,
219, 253, 253, 253, 253, 195,  80,   9,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  55, 172,
226, 253, 253, 253, 253, 244, 133,  11,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
136, 253, 253, 253, 212, 135, 132,  16,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0], dtype=uint8), 5)
```

In [103]:
```
# Part a)
# Leverage covariance matrix to gather eigenvalues & eigenvectors
cov=np.cov(train_x.T)
lamda, evectors = np.linalg.eig(cov)
lamda = np.float64(lamda)
evectors = np.float64(evectors)
total_var = sum(lamda)
```

```
/Users/kkannapp/anaconda/lib/python3.6/site-packages/ipykernel_launcher.py:5: ComplexWarning: Ca
  """
/Users/kkannapp/anaconda/lib/python3.6/site-packages/ipykernel_launcher.py:6: ComplexWarning: Ca
```

In [104]:
```
# Provides the sum of lost variance
sum(lamda[25::])
```

Out[104]:
```
1056647.6637139525
```

In [105]:
```
# Let us compute a fraction of lost variance, for each value of k defined by F(k):
def F(k):
    lost_variance = sum(lamda[k+1::])
    return lost_variance/total_var
```

In [106]:
```
k_var = [(i, F(i)) for i in [200, 150, 100, 50, 25]]

k_lost_var = pd.DataFrame(data = k_var, columns = ['k value', 'Variance Lost'])
k_lost_var
```

Out[106]:

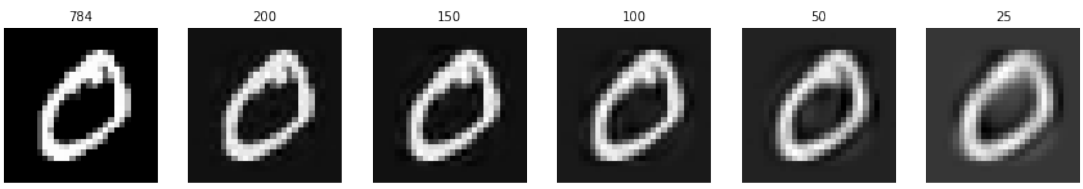
	k value	Variance Lost
0	200	0.033271
1	150	0.051177
2	100	0.084387
3	50	0.172163
4	25	0.299802

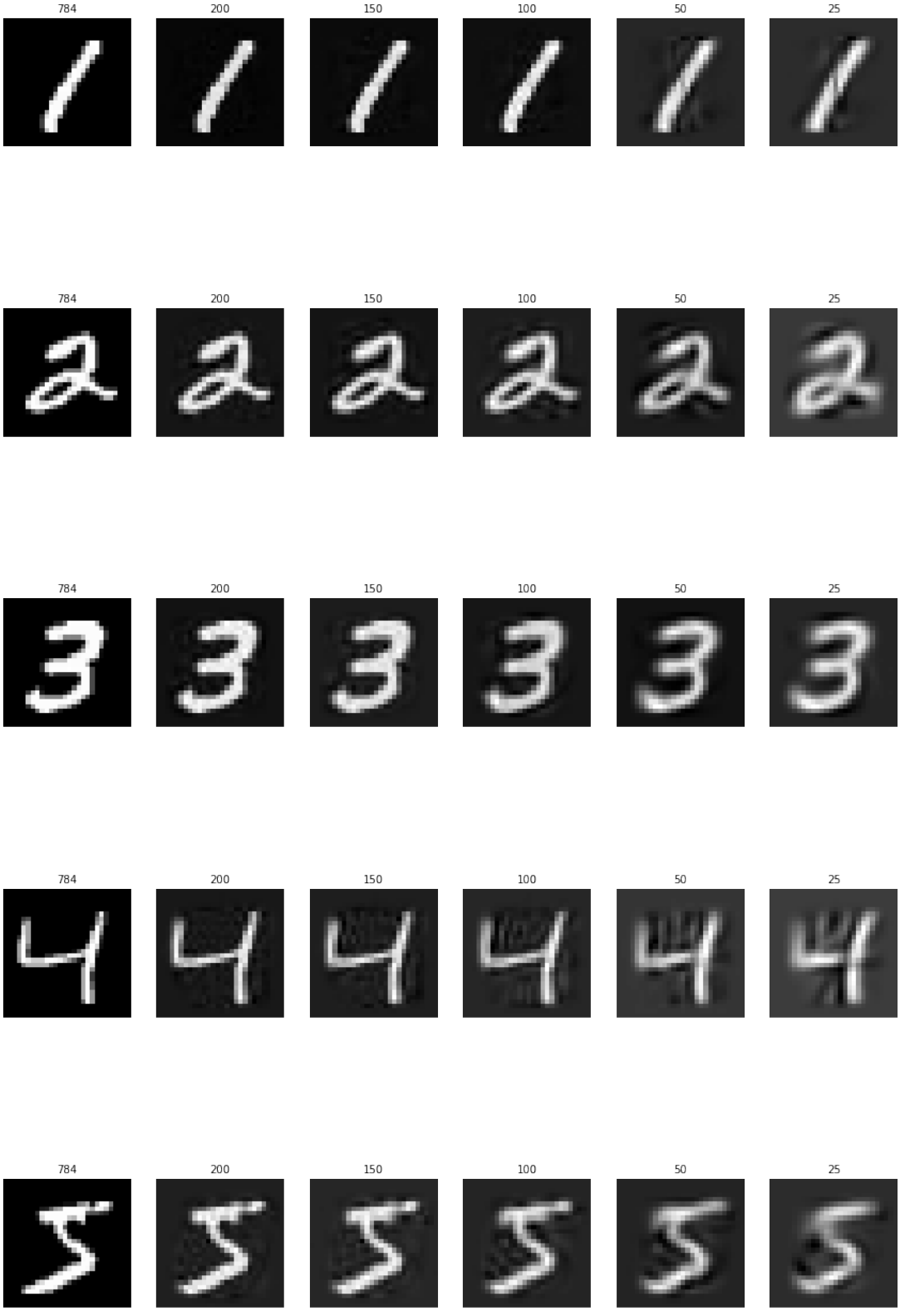
In [144]:
```
# Part b)
# Apply PCA formula to re-generate images, calculate class probabilities

class PCA_fx(object):

    def __init__(self,train_x,train_y):
        self.train_x = train_x
        self.train_y = train_y
        self.dict={}
        self.dict = {num:{} for num in range(10)}

    def F(self, k):
        lost_variance = sum(self.lamda[k+1::])
        return lost_variance/self.total_var

    def reshape_proj(self, k, image, image_id):
        U_matrix = self.evectors[:,:k+1]
        transform_U = np.dot(U_matrix,U_matrix.T)
        image = image.T
        X = np.dot(transform_U,image).T[image_id]
        X = X.reshape(28,28)
```

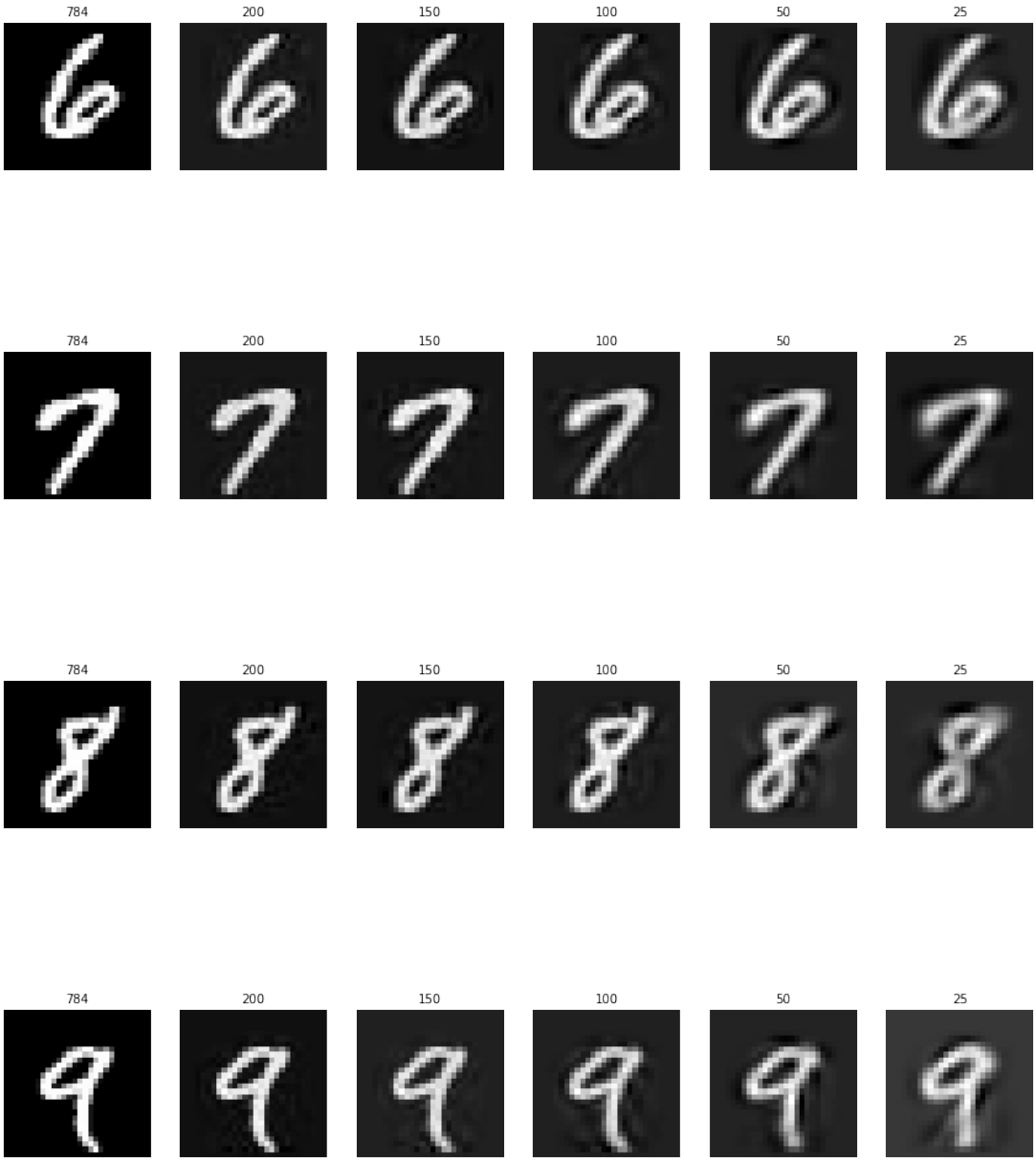
```
            return X

        def show_digit(self,X,k):
            plt.imshow(X, cmap=plt.cm.gray)
            plt.title('%i' % k, fontsize = 10)
            plt.axis('off')

        def computation(self):
            for num in range(10):
                x_train = []
                y_train = []
                for i, image in enumerate(self.train_x):
                    if train_y[i] == num:
                        x_train.append(image)
                        y_train.append(num)
                x_train = np.array(x_train)
                cov = np.cov(x_train.T)
                lamda, evectors = np.linalg.eig(cov)
                self.lamda = np.float64(lamda)
                self.evectors = np.float64(evectors)
                self.total_var = np.sum(self.lamda)
                index = 0
                plt.figure(figsize=(15,5))
                for k in [784, 200, 150, 100, 50, 25]:
                    self.dict[num][k]=self.F(k)
                    image_reduced = self.reshape_proj(k, x_train[:10], 0)
                    index += 1
                    plt.subplot(1, 6, index)
                    self.show_digit(image_reduced, k)
                plt.show();
```

In [145]:
```
PCA_Class = PCA_fx(train_x, train_y)
PCA_Class.computation()
```

```
/Users/kkannapp/anaconda/lib/python3.6/site-packages/ipykernel_launcher.py:20: ComplexWarning: C
/Users/kkannapp/anaconda/lib/python3.6/site-packages/ipykernel_launcher.py:21: ComplexWarning: C
```




784
200
150
100
50
25
784
200
150
100
50
25
784
200
150
100
50
25
784
200
150
100
50
25
784
200
150
100
50
25



In [146]:
```
lost_var_table = pd.DataFrame(PCA_Class.dict)
lost_var_table
```

Out[146]:

	0	1	2	3	4	5	6
25	0.201817	0.127616	0.276346	0.268917	0.246501	0.256309	0.216404
50	0.117534	0.071766	0.158965	0.156145	0.145469	0.147014	0.120032
100	0.058764	0.030534	0.077597	0.075243	0.072825	0.070057	0.057224
150	0.034165	0.014634	0.045578	0.043139	0.042213	0.039591	0.032374
200	0.020834	0.006765	0.028082	0.026002	0.025398	0.023609	0.018940
784	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

	7	8	9
25	0.216041	0.291628	0.218670
50	0.126723	0.164755	0.121179
100	0.062837	0.076202	0.056898
150	0.035918	0.042682	0.030738
200	0.021194	0.024901	0.016958
784	0.000000	0.000000	0.000000

Based on this table, I would expect 1 to be the digit most amenable to dimensionality reduction. Intuitively, this makes the most sense.

2 Sampling

2.1 Worksheet 11

1. The distribution of the number of red marbles seen, can be computed using the standard formula for n tosses of a coin of bias p: $N(np, np(1-p))$. Substituting $n = 900$ and $p = \frac{9}{10}$, we have:
   $N(\frac{9}{10} * 900, \frac{9}{10} * \frac{1}{10} * 900)$
   hence: $N(810, 81)$, with $\mu = 810$ and $\sigma = 9$.

2. The distribution of the population of left-handedness, can be computed using the standard formula for n tosses of a coin of bias p: $N(np, np(1-p))$. Substituting $n = 200$ and $p = \frac{1}{100}$, we have:
   $N(\frac{1}{100} * 200, \frac{1}{100} * \frac{99}{100} * 200)$
   hence: $N(2, 1.98)$, with $\mu = 2$ and $\sigma \approx 1.41$. To find the 99% confidence interval for the number of people in the sample that are left handed, we take the mean and add/subtract 3 times the standard deviation: $2 \pm 3 \times 1.41$. The 99% confidence interval is -2.23 to 6.23.

4. Considering we have a population with 1% colorblindness, then we have n people at random, with the probability that none are colorblind at 0.99^n. Since we want this probability of a sample to contain at least one colorblind person to be at least 95%, then we just need to take the value for which $0.99^n < 0.05$. Computing on wolfram alpha, we find the real value to be 298. Hence, we must have a sample of at least 298 people to contain at least one person with probability 95%.

5. Using the true fraction of San Diegans liking sushi of 0.5, we can leverage that $p = 0.5$ for the distribution of the observed fraction of n tosses of a coin of bias p: $N(np, \frac{p(1-p)}{n})$. The standard deviation is the square root of the variance, so we are solving for the second term, specifically: $\sqrt{\frac{p(1-p)}{n}}$:

   (a) For $n = 100$: $\sqrt{\frac{0.5^2}{100}} = 0.05$. $\sigma = 0.05$.

   (b) For $n = 2500$: $\sqrt{\frac{0.5^2}{2500}} = 0.01$. $\sigma = 0.01$.

6. For this problem, p is not provided. However, the good news is that p at its largest variance (0.5) is also a reasonable estimate for the fraction of gender between males and females. For this problem we will set $p = 0.5$ to calculate the sample size so that we can provide a confidence interval with confidence at least 99% estimated within 0.01. Hence, we can solve for the value of n such that $3 * \sigma \leq 0.01$. Inserting $p = 0.5$ into this forumula, we have:
   $3 * \sqrt{\frac{0.5^2}{n}} = 0.01$

$\sqrt{\frac{0.25}{n}} = \frac{0.01}{3}$ which is solved for n by wolfram alpha to be $\approx 22,500$, so maybe they could sample a small town in Wyoming.

8. Using the observed proportion of people enrolled in college from ages 18-24, we have $p = 0.388$. We can use this to solve for the estimate of the fraction of all persons age 18 to 24 in the city who are enrolled in college, we will also provide the 95.5% confidence interval for the estimate, which is taken by multiplying by adding/subtracting $2 * \sigma$ from the mean:
   Since we are estimating the fraction of people 18 to 24 who are enrolled in college, we use the observed proportion to determine the mean. It is $p = 0.388$. Or $\mu = 38.8\%$. To find the confidence interval, we need to solve for the standard deviation:
   $0.388 \pm 2 * (\sqrt{\frac{0.388*0.612}{500}})$, or
   $0.388 \pm 2 * 0.02179$ which yields the 95% confidence interval for the estimate of the mean between 34.4% to 43.2%.

10. Using the observed average test score on the math component of the test for 17-year old students, we have $p = 0.307$. We can use this to estimate the nationwide average score on the math test and compute the standard deviation of this estimate, using the sample standard deviation which was provided to be $\sigma = 30$.
    Since we are estimating the nationwide average score on the math test, we use the observed proportion to determine the mean. It is $p = 0.307$. Or $\mu = 307$. To find the standard deviation, we use the sample standard deviation:
    The standard deviation is approximately $\frac{30}{\sqrt{1000}} \approx 0.95$. Hence, $\sigma = 95$.

3 Hypothesis Testing

3.1 Worksheet 12

2. (a) Observational study.
   (b) To avoid confounding effects.
   (c) This should be tested in a controlled experiment, right now this is an observed effect that has yet to be studied further.

4. No, it was not wise to let teachers to use their judgement in this way. Teachers are unable to randomly select students, hence the experiment was no longer random and thus the results are not valid.

5. Potentially, though I would expect it to be independent. The good news is there are ways to assess the validity and scope of this effect!

6. (a)

   Null Hypothesis 1. *The coin is fair*

   Hypothesis 1. *The coin is biased*

   (b) Calculate mean: $\mu = 0.5 * 10000 = 5000$. Calculate standard deviation: $\sigma = \sqrt{10000 * 0.5^2} = 50$. Hence, $Z = \frac{5400-5000}{50} = 8$. Leveraging a Z-score table, we find the p-value to be < 0.01 which is statistically significant at the 99% level.

   (c) Since the p-value is statistically significant, we can safely reject the null hypothesis that the coin is fair and conclude that the coin is biased.

7. This can be explained as chance variation. Using mean and variance summation principles, we are able to conclude that the expected value is the same as the sum of the expected values, which is the same for the variance. Given that $\mu_1 = 3.5$ and $\sigma_1^2 \approx 2.7$, we just need to multiply the forumals by the amount of dice rolls (100). We can then determine the expected number of spots to be 350, which was given, and the variance to be ≈ 2742. Hence, $\sigma \approx 52$, which would yield a $Z \approx 1$, which is not significant to reject the null hypothesis that the dice is fair.

9. (a) $\sigma_1 = \sqrt{\frac{0.219*0.781}{700}} \approx 0.01563$, $\sigma_2 = \sqrt{\frac{0.11*0.89}{700}} \approx 0.01183$. Hence, $\sigma = \sqrt{0.01563^2 + 0.01183^2} \approx 0.01960$. From here, we calculate $Z = \frac{0.219-0.11}{0.01960} \approx 5.6$. The difference in marijuana usage percentage is real.

   (b) $\sigma_1 = \sqrt{\frac{0.369*0.631}{700}} \approx 0.01824$, $\sigma_2 = \sqrt{\frac{0.11*0.89}{700}} \approx 0.01762$. Hence, $\sigma = \sqrt{0.01824^2 + 0.01762^2} \approx 0.02536$. From here, we calculate $Z = \frac{0.369-0.319}{0.02536} \approx 1.97$. The difference in cigarette smoking percentage is real.

10. $\sigma_{pu} = \frac{10.5}{\sqrt{1000}} \approx 0.3320$, $\sigma_{pr} = \frac{9.9}{\sqrt{1000}} \approx 0.3131$. Hence, $\sigma = \sqrt{0.3320^2 + 0.3131^2} \approx 0.4564$. From here, we calculate $Z = \frac{12.2-9.2}{0.4564} \approx 6.6$. The difference in working hours between public and private universities is extremely unlikely to be due to chance.