CS 361 Spring 2018
Homework 9

Nathaniel Murphy (njmurph3)

10.1

(a)

$$\begin{aligned}\text{mean}(\{f\}) &= \text{mean}(a^T\{x\}) \\ &= a^T\text{mean}(\{x\})\end{aligned}$$

because a^T is a constant.

(b)

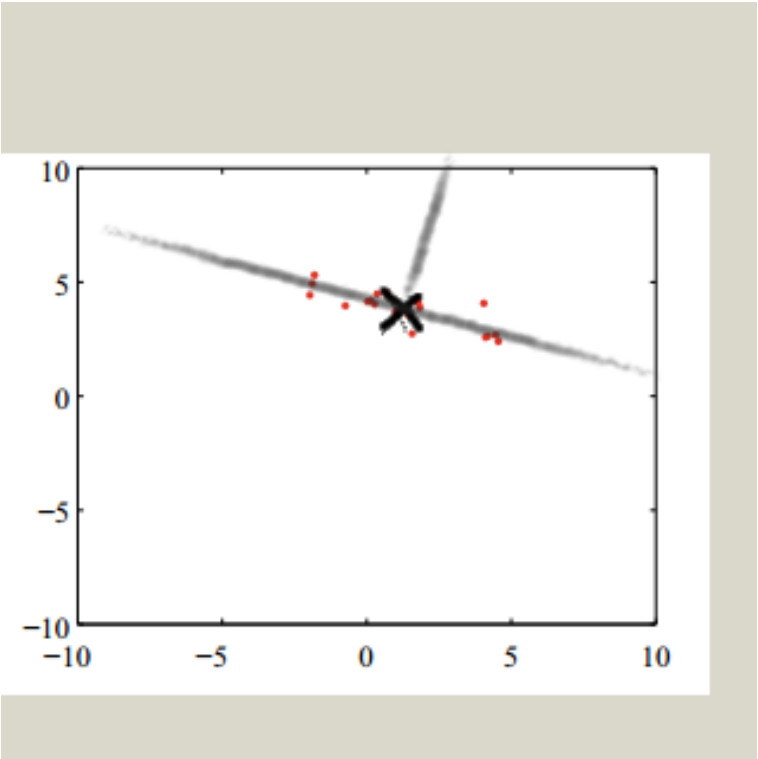
$$\begin{aligned}\text{var}(\{f\}) &= \text{var}(a^T\{x\}) \\ &= \frac{\sum_i \left(a^T x_i - \text{mean}(a^T\{x\})\right)\left(a^T x_i - \text{mean}(a^T\{x\})\right)^T}{N} \\ &= \frac{\sum_i \left(a^T x_i - a^T\text{mean}(\{x\})\right)\left(a^T x_i - a^T\text{mean}(\{x\})\right)^T}{N} \\ &= \frac{\sum_i \left(a^T (x_i - \text{mean}(\{x\}))\right)\left(a^T (x_i - \text{mean}(\{x\}))\right)^T}{N} \\ &= \frac{\sum_i a^T (x_i - \text{mean}(\{x\}))(x_i - \text{mean}(\{x\}))^T a}{N} \qquad \text{because } (AB)^T = B^T A^T \\ &= a^T \left(\frac{\sum_i (x_i - \text{mean}(\{x\}))(x_i - \text{mean}(\{x\}))^T}{N}\right) a \\ &= a^T \text{Covmat}(\{x\}) a\end{aligned}$$

(c)

Suppose that for some a, $a^T\text{Covmat}(\{x\})a = 0$.

Because $a^T\text{Covmat}(\{x\})a = 0$, we know that a^T is an eigenvector of $\text{Covmat}(\{x\})$ with eigenvalue 0. because the eigenvalue is 0, we can say that this eigenvector preserves no variance of the data. Consider the hyperplane that is orthogonal to a^T. This hyperplane must preserve all variance of the data simply by the fact that a^T preserves none. Since 100% of the data can be expressed with this hyperplane, it must follow that the dataset lies in a hyperplane.

10.2



The black X drawn on the data represents the mean of the dataset. The longer horizontal line represents the first principle component because most of the variance lies on this line. Finally, the second principle component is perpendicular to the first principle component and this represents the variance not captured by the first principle component.

10.3

(a)

Because the covariance matrix has one non-zero eigenvalue, we may say that 100% of the variance of the data can be explained using one dimension. This implies that all of the data points lie on a line in d-dimensional space. This means that any data point is a basis for the data set. i.e.

$$x_i = a \cdot x_j \text{ for some } a \in \mathbb{R} \text{ and } j < N$$

It clearly follows that

$$x_i = a \cdot x_j \Rightarrow x_i = x_1 + t_i(x_2 - x_1)$$

because $x_2 - x_1$ lies on the same line formed by x_1 and x_2 continuing infinitely (because they are linearly dependent). Call this value x_k. $x_k + x_1$ is also on the infinite line formed by x_1 and x_2 because of the same linear dependence between all vectors in the dataset. Let $x_\ell = x_k + x_1$. Because the infinite line formed by x_1 and x_2 si precisely $a \cdot x_j$, $a \in \mathbb{R}$, $j < N$, we can say that such an $a \in \mathbb{R}$ exists such that $x_i = a \cdot x_\ell$.

(b)

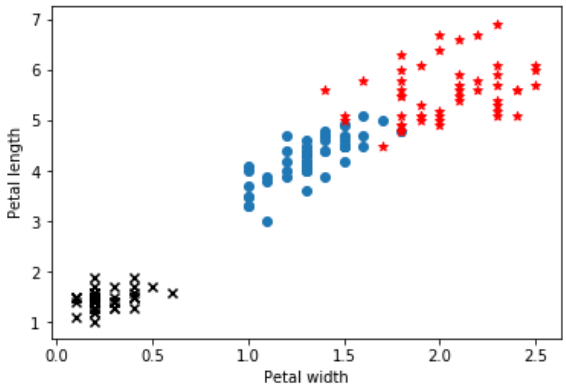
Because all the variance is explained using 1 dimension, the t values are precisely the dataset projected on the line $a \cdot x_j$, $a \in \mathbb{R}$, $j < N$. Because no variance is lost, std($\{t\}$) =std($\{x\}$).

10.4

a)

For this, we change all the class labels to numbers so that we can more conveniently identify the classes. Since we have 4 features, but can only plot 2 given the graphical tools that we have, I have chosen to plot petal width and petal length, as those separated the classes very well in my opinion.

```
plt.scatter(data[labels['class']==0]['petal width'], data[labels['class']==0]['petal length'], marker='x', c='k')
plt.scatter(data[labels['class']==1]['petal width'], data[labels['class']==1]['petal length'])
plt.scatter(data[labels['class']==2]['petal width'], data[labels['class']==2]['petal length'], marker='*', c='r')
plt.xlabel('Petal width')
plt.ylabel('Petal length')
plt.show()
```



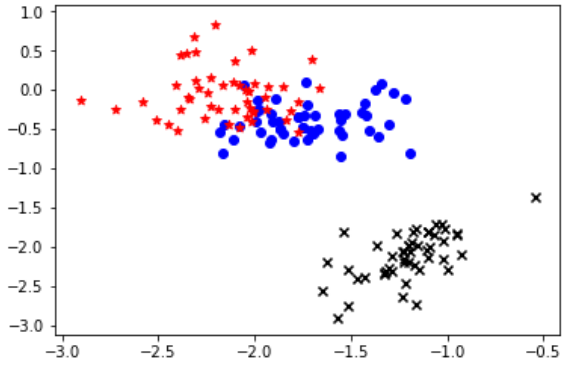
b)

For this problem, we first find the covariance matrix using a numpy library. We can then determine all of the eigenvalues and eigenvectors associated with this covariance matrix, order them, and choose the two eigenvectors with the highest corresponding eigenvalues. This will ensure that the most variance is retained in the dataset. We can then create our change of base matrix from these two eigenvectors and plot the data using the new base, which is 2-dimensional rather than 4-dimensional. We cannot label the axes of the graph because we the principal components will have parts of each feature in them.

```
covmat = np.matrix(data.cov())
eigenvalues, eigenvectors = np.linalg.eig(covmat)
first_component = eigenvectors[np.argsort(eigenvalues)[-1]]
second_component = eigenvectors[np.argsort(eigenvalues)[-2]]
```

```
Q = np.vstack((first_component, second_component))
x = np.matrix(data)
A = np.array(x * Q.T)
```

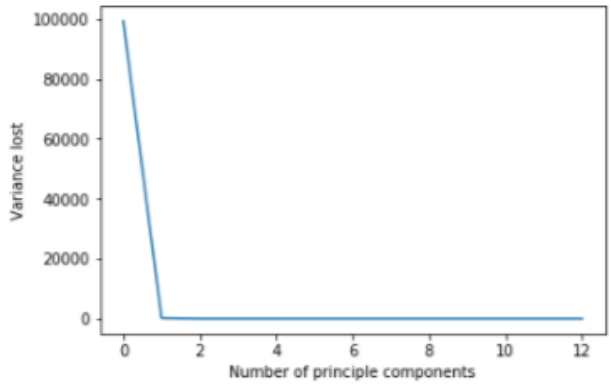
```
plt.scatter(A[labels['class']==0][:,0].ravel(), A[labels['class']==0][:,1].ravel(), marker='x', c='k')
plt.scatter(A[labels['class']==1][:,0].ravel(), A[labels['class']==1][:,1].ravel(), marker='o', c='b')
plt.scatter(A[labels['class']==2][:,0].ravel(), A[labels['class']==2][:,1].ravel(), marker='*', c='r')
plt.show()
```

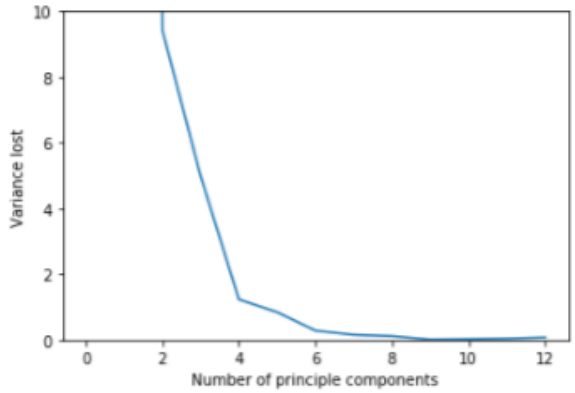


10.5

a)

```
covmat = np.matrix(data.cov())
eigenvalues, eigenvectors = np.linalg.eig(covmat)
plt.plot(np.arange(len(eigenvalues)), eigenvalues)
plt.xlabel('Number of principle components')
plt.ylabel('Variance lost')
plt.show()
plt.plot(np.arange(len(eigenvalues)), eigenvalues)
plt.ylim(0,10)
plt.xlabel('Number of principle components')
plt.ylabel('Variance lost')
plt.show()
```



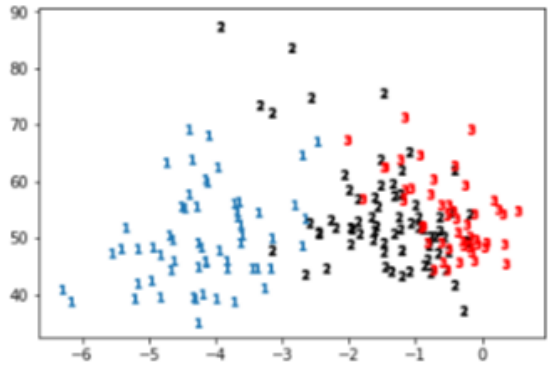


It looks like if we decided to use 6 principle components, we would retain much of the variance of the dataset becuase the sixth eigenvalue is less than 1.

b)

```
first_component = eigenvectors[np.argsort(eigenvalues)[-1]]
second_component = eigenvectors[np.argsort(eigenvalues)[-2]]
Q = np.vstack((first_component, second_component))
x = np.matrix(data)
A = np.array(x * Q.T)
```

```
plt.scatter(A[labels['class']==1][:,0].ravel(), A[labels['class']==1][:,1].ravel(), marker='$1$')
plt.scatter(A[labels['class']==2][:,0].ravel(), A[labels['class']==2][:,1].ravel(), marker='$2$', c='k')
plt.scatter(A[labels['class']==3][:,0].ravel(), A[labels['class']==3][:,1].ravel(), marker='$3$', c='r')
plt.show()
```



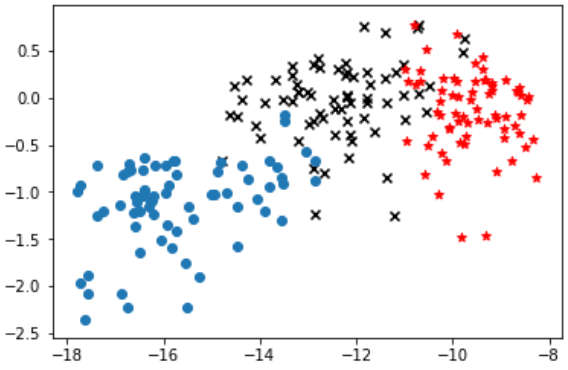
The resulting plot is the 2-dimensional plot that retains the most variance from the original dataset.

10.6

a)

```
plt.scatter(A[labels['class']==1][:,0], A[labels['class']==1][:,1], marker='x', c='k')
plt.scatter(A[labels['class']==2][:,0], A[labels['class']==2][:,1])
plt.scatter(A[labels['class']==3][:,0], A[labels['class']==3][:,1], marker='*', c='r')
```

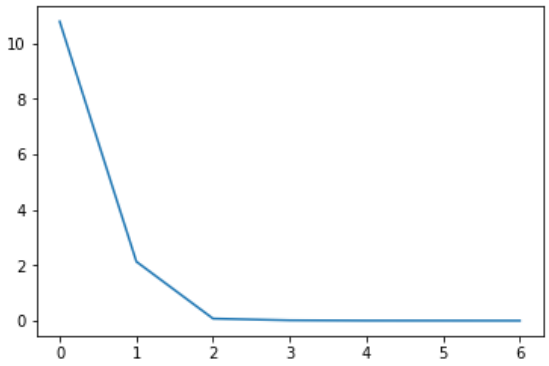
```
<matplotlib.collections.PathCollection at 0x7f4179fcb908>
```



It seems as if the data exhibits a positive linear correlation. The data is well separated into classes.

b)

```
plt.plot(np.arange(len(eigenvalues)), np.flip(np.sort(eigenvalues), 0))
plt.show()
```



It looks as if only using the first 2 eigenvectors as a basis for the dataset will retain much of the variance of the dataset.