# Intro to ML

## Ex0

### 1.a)

An operator $L$ is linear if it obeys:

$$L(x + y) = Lx + Ly$$

where $x, y$ are eg. functions

and

$$L(\lambda x) = \lambda Lx$$

where $\lambda \in \mathfrak{R}$

Show that the operator $E$ is linear:

$E[f(\omega)] = \sum_{\omega} P(\omega) f(\omega)$

$$E[f(\omega) + g(\omega)] = \sum_{\omega} P(\omega) \left(f(\omega) + g(\omega)\right)$$

$$E[f(\omega) + g(\omega)] = \sum_{\omega} P(\omega) f(\omega) + \sum_{\omega} P(\omega) g(\omega)$$

$$E[f(\omega) + g(\omega)] = E[f(\omega)] + E[g(\omega)]$$

$$E[cf(\omega)] = \sum_{\omega} P(\omega) c f(\omega)$$

$$E[cf(\omega)] = c \sum_{\omega} P(\omega) f(\omega)$$

$$E[cf(\omega)] = cE[f(\omega)]$$

**1.c)**

$$\begin{aligned}
\mathrm{Var}[f(\omega)] &= E[(f(\omega) - E[f(\omega)])^2] \\
\mathrm{Var}[f(\omega)] &= E[f(\omega)^2 - 2f(\omega)E[f(\omega)] + E[f(\omega)]^2] \\
\mathrm{Var}[f(\omega)] &= E[f(\omega)^2] - 2E[f(\omega)]E[f(\omega)] + E[f(\omega)]^2 \\
\mathrm{Var}[f(\omega)] &= E[f(\omega)^2] - E[f(\omega)]^2
\end{aligned}$$

**2.**

Eigenfunction $\mathrm{A}x = \lambda x$

Matrix $\mathrm{A} \in \mathfrak{R}^{n \times n}$
Eigenvalue $\lambda \in \mathfrak{R}$
Eigenvector $x \in \mathfrak{R}$

$\mathrm{A}$ has $n$ orthonormal eigenvectors $x_i$

A matrix $\mathrm{A}$ can be expressed in the basis of it's eigenvectors
$X = [\lambda_1 + \lambda_2 + \ldots + \lambda_n]$ and $X^\top$

$$\mathrm{A} = X \Lambda X^\top$$

where $\Lambda$ is a diagonal matrix with $\mathrm{Tr}(\Lambda) = \sum_i^n \lambda_i$

$$\mathrm{A} = \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ x_1 & x_2 & \ldots & x_n \\ \downarrow & \downarrow & & \downarrow \end{pmatrix} \begin{pmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{pmatrix} \begin{pmatrix} \leftarrow & x_1 & \rightarrow \\ \leftarrow & x_2 & \rightarrow \\ \leftarrow & \vdots & \rightarrow \\ \leftarrow & x_n & \rightarrow \end{pmatrix}$$

Multiplying the matrices yields $\mathrm{A} = \sum_{i=1}^n \lambda_i x_i x_i^\top$ Hence $\mathrm{A} = \mathrm{B}$ and the eigenvectors and -values are the same for $\mathrm{B}$ as well.

## 3.a)

$$f(x) = ax^4 + bx + c$$
$$f'(x) = 4ax^3 + b$$

The extreme values are found when $f'(x) = 0$

$$4ax^3 + b = 0$$
$$4ax^3 = -b$$
$$x^3 = \frac{-b}{4a}$$
$$x = \sqrt[3]{\frac{-b}{4a}}$$

## b)

Conditions: $a > 0$ and none of them can go to the infinity (and beyond)

## 4.

Pseudocode for producing fibonacci numbers

```
func fibo(n) {
    i = 0
    j = 1
    for k in [1,n + 1] {
        i = j
        j = i + j
        print j
```

The time compexity is $\mathcal{O}(n)$ for there's only one for loop and it goes through all the numbers $[1, n + 1]$.

## 5.

> Sorry I do this first in Python and maybe later if I have time in R

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [5]:

```
x_data = pd.read_csv("x.csv")
x_data.sample(4)
```

Out[5]:

| | V1 | V2 | V3 | V4 | V5 | V6 |
|---|---|---|---|---|---|---|
| **246** | -5.449209 | -4.839936 | 7.791066 | -4.113574 | 3.116813 | -1.612643 |
| **937** | -5.580300 | -5.008709 | 7.743093 | -3.919815 | 3.864029 | -1.348257 |
| **242** | -5.904165 | -5.309392 | 8.032675 | -1.519781 | 3.971225 | 0.910792 |
| **367** | -5.347353 | -4.902344 | 5.388195 | -4.786115 | 4.851407 | -2.117089 |

4 rows × 32 columns

In [14]:

```
np.where(x_data.describe().loc["std",:] > 2)[0]
```

Out[14]:

```
array([12, 20])
```

In [16]:

```
use = x_data.iloc[:,np.where(x_data.describe().loc["std",:] > 2)[0]]
```

In [17]:

```
use.sample(3)
```

Out[17]:

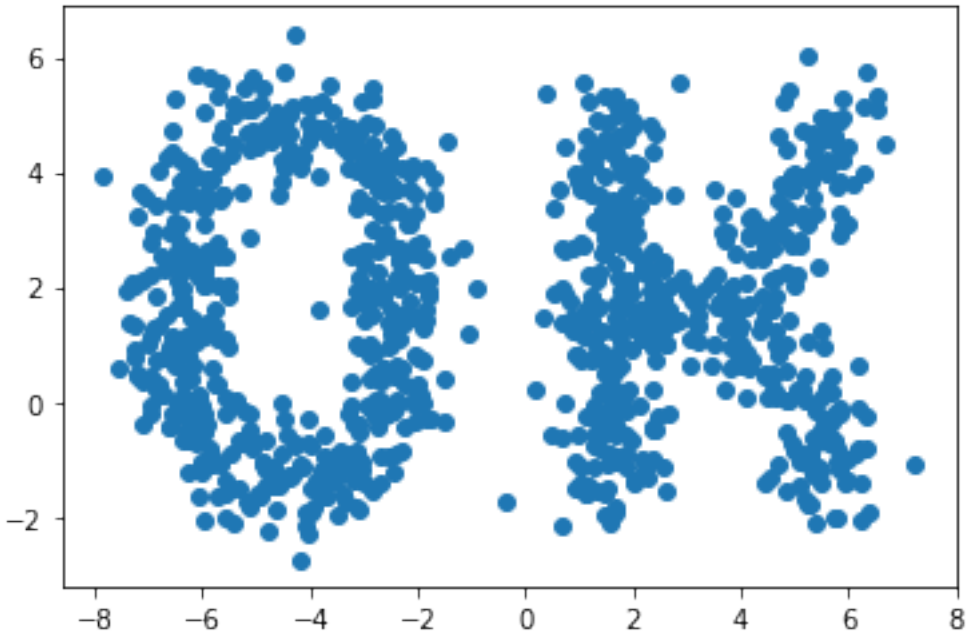
| | V13 | V21 |
|---|---|---|
| **289** | 3.115115 | 1.729768 |
| **870** | -5.590086 | -1.346247 |
| **482** | 5.310008 | -1.757451 |

In [27]:

```
plt.scatter(use.V13,use.V21)
```

Out[27]:

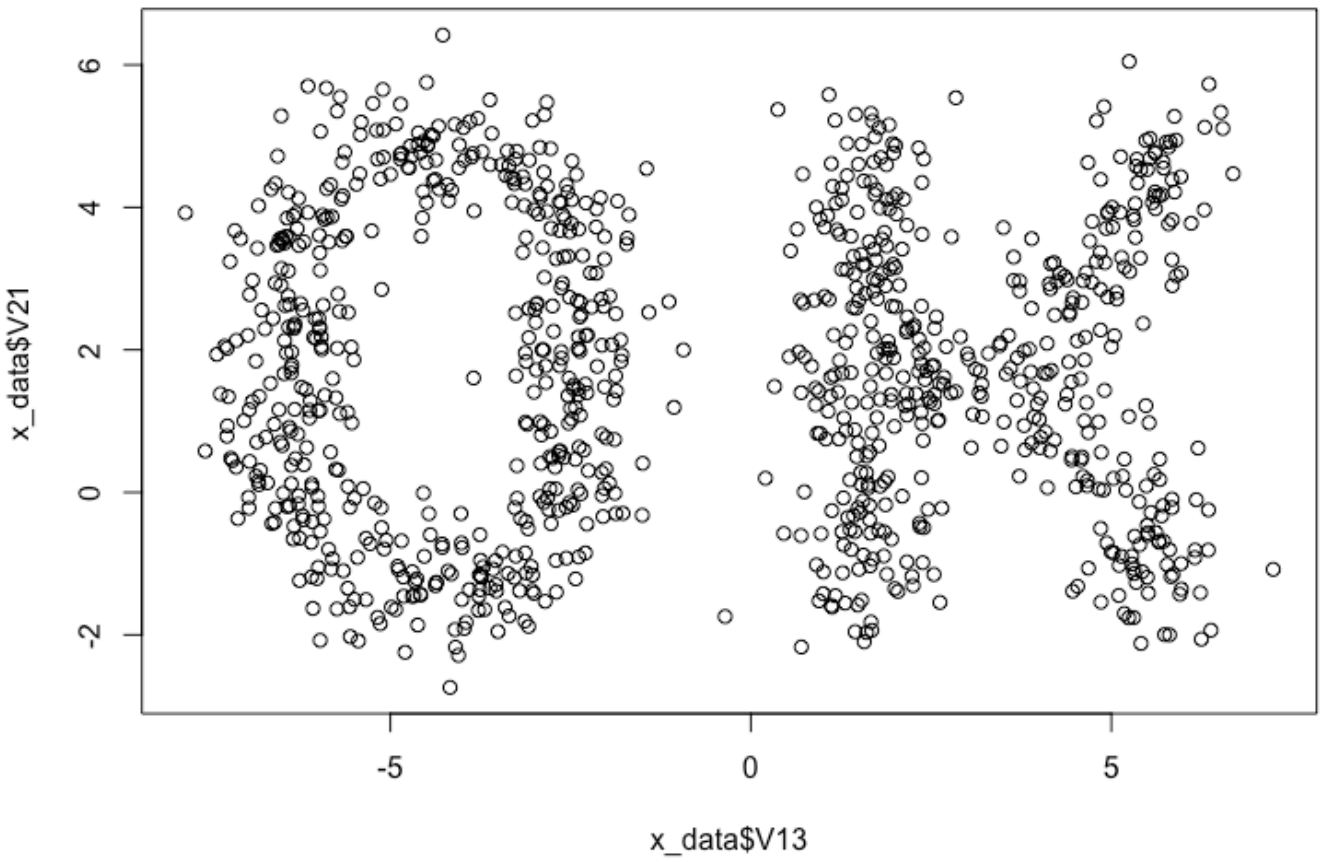
```
<matplotlib.collections.PathCollection at 0x11b443
9b0>
```




Aaaand here's the R code:

```r
# Read data in

x_data = read.csv(file = "~/Documents/GitHub/IntroML/Ex
0/x.csv", header = TRUE, sep = ",")

?sapply
?var

vars = sapply(x_data, var)
vars

?which.max

imax1 = which.max(vars)
imax1

vars[imax1] = 0
vars

imax2 = which.max(vars)
imax2

plot(x_data[, imax1], x_data[, imax2])
```

And Here's the plot:




In [ ]: