LINEAR ALGEBRA AND PROBABILITY TAKE HOME EXAM.

Student Name - Chetana Sharma
Program / Year - MSDS / Spring 2020
NUID - 001055979

1.

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$$

Assuming $f(x) = x^T P x$ is a convex function

$$\Rightarrow f(\lambda x + (1-\lambda)y) < \lambda(x^T P x) + (1-\lambda)(y^T P y)$$

$$\Rightarrow (\lambda x + (1-\lambda)y)^T P(\lambda x + (1-\lambda)y) < \lambda x^T P x + (1-\lambda) y^T P y$$

$$\Rightarrow \lambda x^T P(\lambda x + (1-\lambda)y) + (1-\lambda) y^T P(\lambda x + (1-\lambda)y) < \lambda x^T P x + (1-\lambda) y^T P y$$

$$\Rightarrow \lambda^2 x^T P x + \lambda(1-\lambda) x^T P y + \lambda(1-\lambda) y^T P x + (1-\lambda)^2 y^T P y < \lambda x^T P x + (1-\lambda) y^T P y$$

Taking $x^T P x$ and $y^T P y$ terms from L.H.S to R.H.S

$$\Rightarrow \lambda(1-\lambda) x^T P y + \lambda(1-\lambda) y^T P x < \lambda(1-\lambda) x^T P x + \lambda(1-\lambda) y^T P y$$

Removing $\lambda(1-\lambda)$ from both L.H.S and R.H.S

$$\Rightarrow x^T P y + y^T P x < x^T P x + y^T P y$$

$$\Rightarrow x^T P(y-x) + y^T P(x-y) < 0$$

$$\Rightarrow x^T P(y-x) - y^T P(y-x) < 0$$

$$\Rightarrow (x^T P - y^T P)(y-x) < 0$$

$\Rightarrow (x^T P - y^T P)(x - y) > 0$

$\Rightarrow (x^T - y^T) P (x - y) > 0$

$\Rightarrow (x - y)^T P (x - y) > 0$

Let $x - y = z$

$\Rightarrow z^T P z > 0$

$\Rightarrow f(x)$ is convex function when $z^T P z > 0$

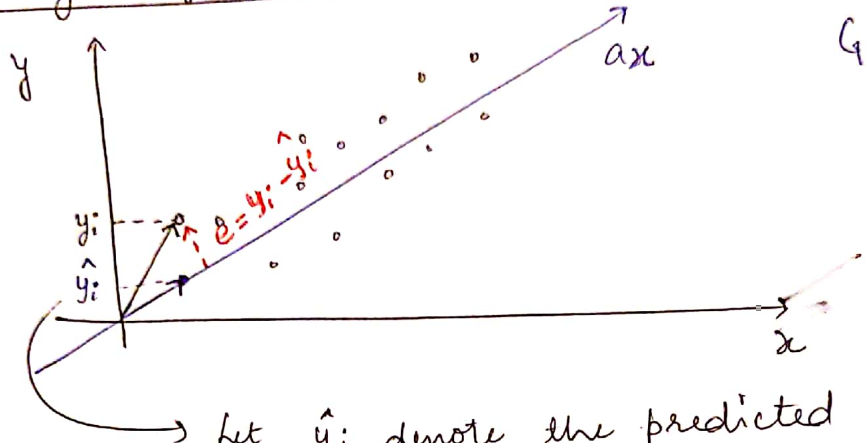
And $z^T P z > 0$ is a property when P is a positive-definite matrix.

Q

DETERMINISTIC APPROACH

Given: sets of data $y_i \in R^{n_y}$, $x_i \in R^{n_x}$ where $i = 1, 2, \ldots n$

Using Projection



Let $\hat{y}_i$ denote the predicted value

Matrix General form:

$$\hat{Y} = XA$$

We need to minimise the distance between y_i and $\hat{y}_i$. That is possible only when 'e' is perpendicular to the line ax.

The projected value is nothing but a projection of the actual point on the linear line / plane or hyperplane.

So if $\hat{Y} = XA$

error $E = Y - \hat{Y}$

And $\langle X, E \rangle = 0$

$\Rightarrow \quad X^T(Y - \hat{Y}) = 0 \quad \Rightarrow X^T(Y - XA) = 0$

$\Rightarrow \quad X^T Y = X^T X A$

$\Rightarrow \quad \boxed{A = (X^T X)^{-1} X^T Y}$

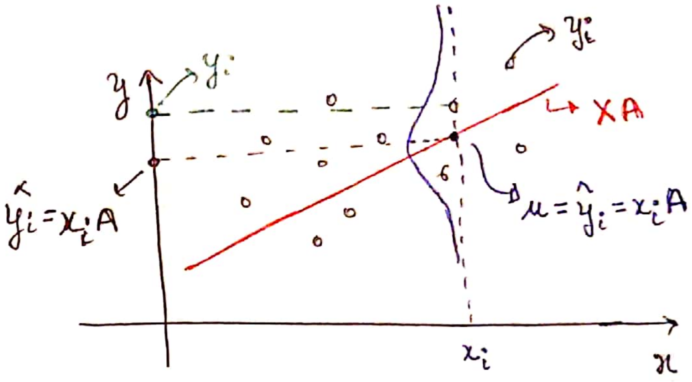
2 PROBABILISTIC APPROACH

Given: $y_i \in R^{n_y}$, $x_i \in R^{n_x}$, $i \in 1, 2, 3, \dots n$

$$y = Ax$$

For a scalar case:

$$y_i = x_i a \quad \text{or} \quad Y = XA$$



Assuming that each y_i is Gaussian distributed with mean $x_i a$.

$$y_i = N(x_i a, \sigma^2)$$

$$P(Y|X, A, \sigma) = \prod_{i=1}^{n} p(y_i | x_i, A, \sigma)$$

$$= \prod_{i=1}^{n} (2\pi\sigma^2)^{-1/2} e^{-1/2\sigma^2 (y_i - x_i A)^2}$$

$$= (2\pi\sigma^2)^{-n/2} e^{-1/2\sigma^2 \sum_{i=1}^{n} (y_i - x_i a)^2}$$

$$= (2\pi\sigma^2)^{-n/2} e^{-1/2\sigma^2 (Y - XA)^T (Y - XA)}$$

Taking log on both sides

$$\log(P(Y|X, A, \sigma)) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(Y - XA)^T(Y - XA)$$

We need to find A that maximises the likelihood of seeing the training data Y given X.

So we differentiate w.r.t. A & equate to zero.

Scanned by CamScanner

$$l(A) = \frac{-n}{2}(2\pi\sigma^2) - \frac{1}{2\sigma^2}(Y - XA)^T(Y - XA)$$

$$\frac{\partial l}{\partial A} = \frac{-1}{2\sigma^2}\frac{\partial}{\partial A}\left(Y^TY - Y^TXA - A^TX^TY + A^TX^TXA\right)$$

$$= \frac{-1}{2\sigma^2}\left(-Y^TX - X^TY + 2X^TXA\right)$$

$$\frac{\partial l}{\partial A} = 0$$

$$\Rightarrow \frac{-1}{2\sigma^2}\left(-2X^TY + 2X^TXA\right) = 0$$

$$\Rightarrow \quad A = \frac{X^TY}{X^TX}$$

$$\Rightarrow \boxed{A = (X^TX)^{-1}X^TY}$$

3. $x_0 \in R^n$, $\boxed{x_{k+1} = Ax_k \quad -(1)}$ → Given

Using (1)

$x_1 = Ax_0$

$x_2 = Ax_1 \Rightarrow x_2 = A(Ax_0) \Rightarrow x_2 = A^2 x_0$

$\Rightarrow \boxed{x_k = A^k x_0}$

Expectation of x_1 and x_k.

$x_1 = Ax_0$

Applying expectation operator both sides.

$E[x_1] = E[Ax_0]$

$\Rightarrow \boxed{E[x_1] = A\,E[x_0]}$

For $x_k = A^k x_0$.

$\boxed{E[x_k] = A^k E[x_0]} \quad - (2)$

Considering $n=1$ and writing in terms of eigen values

$Ax_0 = \lambda x_0$ where λ = eigen value and
x_0 = eigen vector of A.

$x_1 = Ax_0$ [using (1)]

$E[x_1] = E[\lambda x_0]$

$\Rightarrow E[x_1] = \lambda E[x_0]$

Similarly, we can write

$\boxed{E[x_k] = \lambda^k E[x_0]}$ So, if $|\lambda| < 1$ and $k \to \infty$
$\lambda^k \to 0$

$\therefore$ Expectation of $x_k \to 0$ for $k \to \infty$ if Eigen value of A between (-1) and (1).

For general 'n'

Let S be a matrix where all columns are eigen vectors of A.

$$S = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{bmatrix}$$

Let Λ be the eigen value matrix of A.

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & & 0 \\ 0 & \lambda_2 & \cdots & | \\ | & | & & | \\ 0 & 0 & & \lambda_n \end{bmatrix}$$

We know that, $Av_1 = \lambda_1 v_1$, $Av_2 = \lambda_2 v_2$ -- $Av_n = \lambda_n v_n$

$\therefore$ $AS = S\Lambda$

Multiplying with S^{-1} both sides

$$\boxed{A = S\Lambda S^{-1}}$$

Also, $A^2 = (S\Lambda S^{-1})(S\Lambda S^{-1}) = S\Lambda^2 S^{-1}$

$\therefore$ $\boxed{A^k = S\Lambda^k S^{-1}}$ — (3)

Substituting the above value in (2)

$$\boxed{E[x_k] = S\Lambda^k S^{-1} E[x_0]}$$

If $|\lambda_i| < 1$ and $k \to \infty$

then each element of $E[x_k] \to 0$

Else $E[x_k] \to \infty$

VARIANCE

$$var(x_0) = E[x_0^2] - (E[x_0])^2$$

For a vector x_0

$$var(x_0) = \Sigma_{x_0} = E[x_0 x_0^T] - E[x_0]E[x_0]^T \quad - (4)$$

Variance expression for x_1

$$x_1 = Ax_0$$

$$var(x_1) = E[x_1 x_1^T] - E[x_1]E[x_1]^T$$

$$\Rightarrow var(x_1) = E[(Ax_0)(Ax_0)^T] - E[Ax_0]E[Ax_0]^T$$

$$\Rightarrow var(x_1) = E[Ax_0 x_0^T A^T] - AE[x_0]E[x_0]^T A^T$$

$$\Rightarrow var(x_1) = AE[x_0 x_0^T]A^T - AE[x_0]E[x_0]^T A^T$$

$$\Rightarrow var(x_1) = A\left(E[x_0 x_0^T] - E[x_0]E[x_0]^T\right)A^T$$

Using (4)

$$\boxed{var(x_1) = A\Sigma_{x_0}A^T}$$

Variance expression for x_k

$$x_k = A^k x_0$$

$$var(x_k) = E[x_k x_k^T] - E[x_k]E[x_k]^T$$

$$\Rightarrow var(x_k) = E[A^k x_0 x_0^T (A^k)^T] - E[A^k x_0]E[A^k x_0]^T$$

$$\Rightarrow var(x_k) = A^k E[x_0 x_0^T](A^k)^T - A^k E[x_0]E[x_0]^T (A^k)^T$$

$$\Rightarrow \boxed{var(x_k) = A^k \Sigma_{x_0} (A^k)^T}$$

Considering $n=1$.

$$var(x_k) = A^k(var(x_0))(A^k)^T$$

for $n=1$ $\quad Ax_0 = \lambda x_0$ where λ = eigen value

and $A^k x_0 = \lambda^k x_0$

$$\therefore \boxed{var(x_k) = \lambda^{2k} var(x_0)}$$

For $|\lambda| < 1$ and $k \to \infty$

$var(x_k)$ tends to zero faster than the expectation.

Otherwise, $var(x_k) \to \infty$ with $k \to \infty$

For general 'n'

$$var(x_k) = A^k \Sigma_{x_0} (A^k)^T$$

Using (3) i.e. $A^k = S\Lambda^k S^{-1}$

$$var(x_k) = (S\Lambda^k S^{-1}) \Sigma_{x_0} (S\Lambda^k S^{-1})^T$$

If each $|\lambda_i| < 1$ and $k \to \infty$

then $var(x_k) \to 0$

Else $var(x_k) \to \infty$

LINEAR ALGEBRA AND PROBABILITY

PROBLEM-4

Let X & Y be two random variables with uniform distribution.

$$Z = \frac{X+Y}{2}$$

If $X = k$, then $Z = z$ only if $Y = 2z - k$

$$\boxed{P(Z=z) = \sum_{-\infty}^{\infty} P(X=k) \cdot P(Y=2z-k)}$$ ← FOR DISCRETE DISTRIBUTION OF X & Y.

Consider, throw of a fair six-sided dice.

$P(X=x) = 1/6$ for $x \in [1, 6]$

Then $P(Z=z) = \sum_{k=1}^{6} P(X=k)\, P(Y=2z-k)$

$\Rightarrow P(Z=z) = \frac{1}{6} \sum_{k=1}^{6} P(Y=2z-k)$

$P(Z=1) = \frac{1}{6} \sum_{k=1}^{6} P(Y=2-k) = \frac{1}{6}\left(P(Y=2-1)\right) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$

$P(Z=1.5) = \frac{1}{6} \sum_{k=1}^{6} P(Y=3-k) = \frac{1}{6}\left(P(Y=2) + P(Y=1)\right) = \frac{2}{36}$

Similarly, we can find for other values.

$P(Z=2) = \frac{3}{36}$, $P(Z=2.5) = \frac{4}{36}$, $P(Z=3) = \frac{5}{36}$

$P(Z=3.5) = \frac{6}{36}$, $P(Z=4) = \frac{5}{36}$, $P(Z=4.5) = \frac{4}{36}$

$P(Z=5) = \frac{3}{36}$, $P(Z=5.5) = \frac{2}{36}$, $P(Z=6) = \frac{1}{36}$

FOR A CONTINUOUS UNIFORM DISTRIBUTION

$Z = \dfrac{X+Y}{2}$. Let $f(x)$ be the PDF for X and Y.

Let $f(x) = \begin{cases} 1 & 0 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$

then $f(z) = \displaystyle\int_{-\infty}^{\infty} f(x)\, f(2z-x)\, dx$

$= \displaystyle\int_{0}^{1} f(2z-x)\, dx.$

To get the limits:

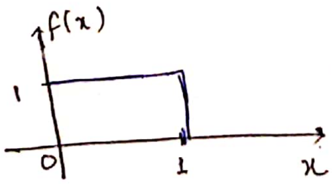
$\Rightarrow \quad 0 \le 2z-x \le 1 \quad \Rightarrow -2z \le -x \le 1-2z$

$\Rightarrow \boxed{2z-1 \le x \le 2z.}$

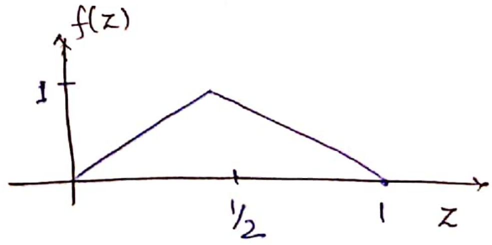
$f(z) = \begin{cases} \displaystyle\int_{0}^{2z} dx & 0 \le z \le 1/2 \\ \displaystyle\int_{2z-1}^{1} dx. & 1/2 < z \le 1 \\ 0 & \text{otherwise} \end{cases}$

$\Rightarrow f(z) = \begin{cases} 2z, & 0 \le z \le 1/2 \\ 2(1-z) & , 1/2 < z \le 1 \\ 0 & \text{otherwise} \end{cases}$

Average of



$\Longrightarrow$



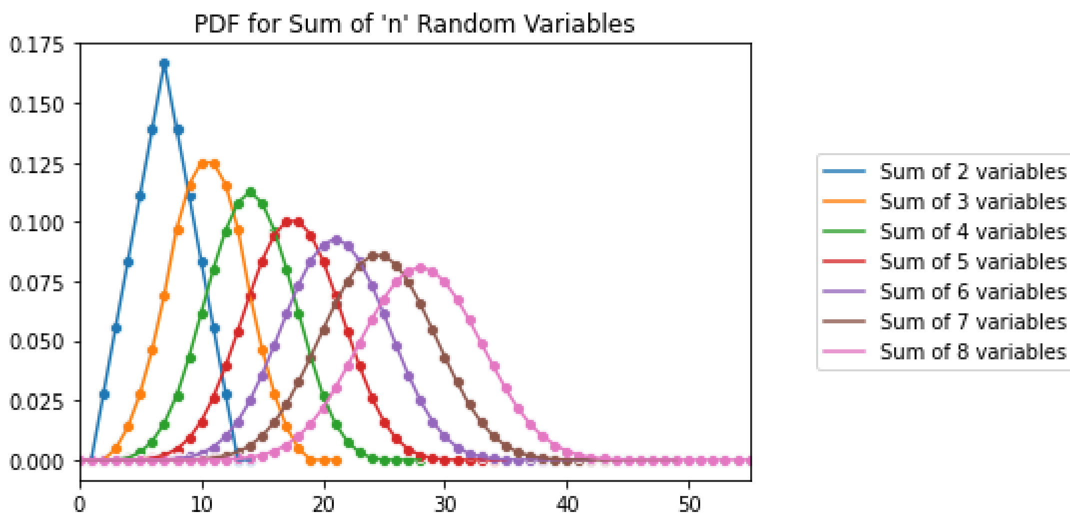
PDF for the average of 2 independent random variables with distribution $f(x)$

Problem-4 Part-B:

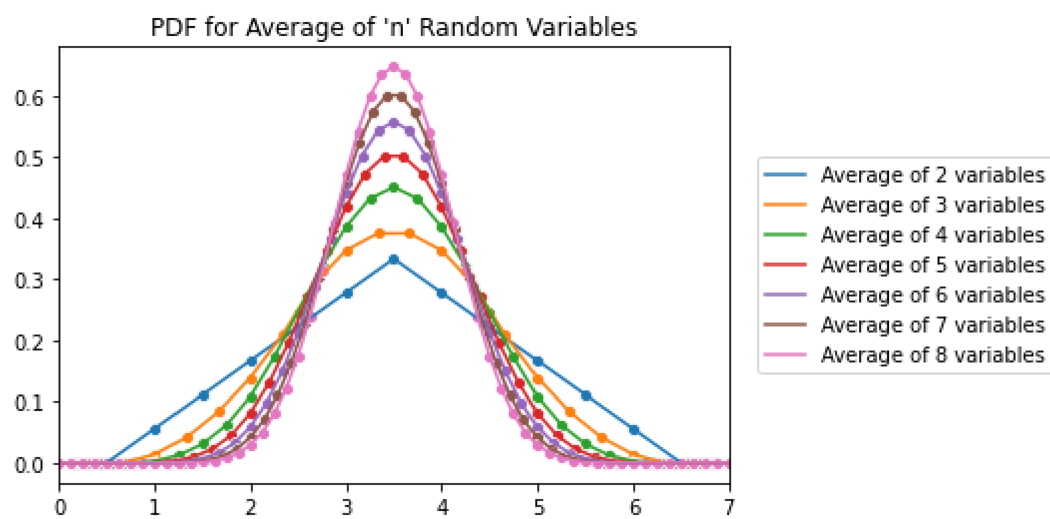
In [74]:
```
from matplotlib import pyplot as plt
from functools import reduce
import seaborn as sns

import numpy as np
import pandas as pd
import scipy

from scipy import signal
```

In [75]:
```
data_sample = list([0,1/6,1/6,1/6,1/6,1/6,1/6,0])
```

In [76]:
```
def n_convolve_sum(n, distribution):
    return np.array(reduce((lambda x, y: signal.convolve(x, y)), [data_s
ample]*n))

n =9
for i in range(2,n):
    conv = n_convolve_sum(i, data_sample)
    sns.lineplot(data = conv ,label = "Sum of " + str(i) +" variables")
    sns.scatterplot(data = conv )
    plt.xlim(0, 6*(n)+1)
    plt.legend(loc='center right', bbox_to_anchor=(1.5, 0.5), ncol=1)
    plt.title("PDF for Sum of 'n' Random Variables")
```



In [77]:
```
def n_convolve_average(n, distribution):
    y_convolve = np.array(reduce((lambda x, y: signal.convolve(x, y)), [
data_sample]*n))
    average_values = {key*(1/n):value/(1/n) for key,value in enumerate(y
_convolve)}
    return list(average_values.keys()),list(average_values.values())


n =9
for i in range(2,n):
    average_values ,average_convolve = n_convolve_average(i, data_sample
)
    sns.scatterplot(x = average_values, y = average_convolve)
    sns.lineplot(x = average_values, y = average_convolve,label = "Avera
ge of " + str(i) +" variables")

    plt.xlim(0, 7)
    plt.legend(loc='center right', bbox_to_anchor=(1.5, 0.5), ncol=1)
    plt.title("PDF for Average of 'n' Random Variables")
```



Inference:

It can be observed that as the number of random variables increases the PDF of average of random variables follow a Gaussian distribution irrespective of the initial distribution. Also the variance of the distribution decreases as the number of random variables increases. This inference aligns with the Central Limit Theorem as well.

PROBLEM-5

$$y = a_1 f_1(x) + a_2 f_2(x) + a_3 f_3(x)$$

$x, y \in$ scalars, Given: (x_i, y_i), $i \in \{1, 2, --n\}$

(a) find parameter vector $(\hat{a}_1, \hat{a}_2, \hat{a}_3)$ that best fits.

In vector notation, Let $f_1(x) = X_1$ for $x_i, i \in \{1, .., n\}$

$f_2(x) = X_2$

$f_3(x) = X_3$

$$\hat{Y} = \hat{a}_1 X_1 + \hat{a}_2 X_2 + \hat{a}_3 X_3$$

$$\Rightarrow \hat{Y} = [X_1 \quad X_2 \quad X_3] \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \end{bmatrix}$$

$$\Rightarrow \boxed{\hat{Y} = XA}$$

The best A would be the one that reduces the error i.e. difference between actual 'Y' and predicted '$\hat{Y}$'.

So we need to minimise $(Y - \hat{Y})^2$.

Squared error $= (Y - XA)^T (Y - XA)$

$$E = Y^T Y - A^T X^T Y - Y^T X A + A^T X^T X A$$

To find A that minimises error, we differentiate.

$$\frac{\partial E}{\partial A} = -X^T Y - Y^T X + 2X^T X A = 0$$

$$\Rightarrow -2X^T Y + 2X^T X A = 0$$

$$\Rightarrow X^T X A = X^T Y$$

$$\Rightarrow \boxed{A = (X^T X)^{-1} X^T Y}$$

Problem-5

PART - (b) , (c) , (d)

Picking f_1 , f_2, f_3 as following: f_1(x) = x , f_2(x) = x^3 , f_3(x) = e^{-x} .

Using a_1 = 11.3, a_2 = 8.7, a_3 = 5.2 randomly to create the training dataset.

Data Simulation

Using small deltas from a normal distribution[N(0,1)] with a mean of 0 and std. deviation as 1, we add some random noise in the training data.

In [224]:
```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
from numpy.linalg import inv
```

In [225]:
```
# Simulation to create training dataset of 200 rows
np.random.seed(12)
x = 10 * np.random.sample(200)
deltax = np.random.normal(0,1,200)
deltay = np.random.normal(0,1,200)
```

In [226]:
```
def functions(x):
    fx1 = x
    fx2 = x*x*x
    fx3 = np.exp(-x)
    return fx1,fx2,fx3
```

In [227]:
```
def get_training_data(x, deltay):
    fx1,fx2,fx3 = functions(x)
    train_y = 11.3*fx1 + 8.7*fx2 + 5.2*fx3 + deltay
    Train_X = np.matrix([[fx1[i],fx2[i],fx3[i]]for i in range(len(x))] )
    Train_Y = np.matrix([[train_y[i]] for i in range(len(train_y))])
    return Train_X, Train_Y
```

In [228]:
```
def calculate_coeff(X, Y):
    first = np.dot(X.T, X)
    second = np.dot(X.T, Y)
    return np.dot(inv(first), second)
```

Below in line 220, added the delta_x to the training data.

In [229]:
```
Train_X, Train_Y = get_training_data(x+deltax, deltay)
plt.scatter(x, np.array(Train_Y), label = "Training data")
plt.title("Training data")
plt.xlabel("x + deltax")
plt.ylabel("y values")
```

Out[229]:
```
Text(0, 0.5, 'y values')
```

Predicting using the coefficients calculated from the training data

The coefficients are calculated using A = (X^{T}$X)^{-1}X^T$Y

In [230]:
```
coeff_A = calculate_coeff(Train_X, Train_Y)
print("Coefficients calculated are: ", coeff_A)
```

```
Coefficients calculated are:  [[11.28469767]
 [ 8.7003012 ]
 [ 5.16718308]]
```

In [231]:
```
fx1,fx2,fx3 = functions(x)
y_predicted = float(coeff_A[[0]])*fx1 + float(coeff_A[[1]])*fx2 + float(
coeff_A[[2]])*fx3
plt.scatter(x, np.array(Train_Y), label = "Training data")
plt.scatter(x, y_predicted, color = "red", label = "Predicted data")
plt.legend()
plt.xlabel("x")
plt.ylabel("y")
```

Out[231]:
```
Text(0, 0.5, 'y')
```

Inference:

The coefficients calculated from the training data are very close to the actual coefficients chosen. The slight variation is due to the small randomness introduced in each x_i and y_i in the training dataset. The curve for the predicted values follow a generic trend of the training data. Hence we can say that the model proposed for calculation is good.