Linear Regression

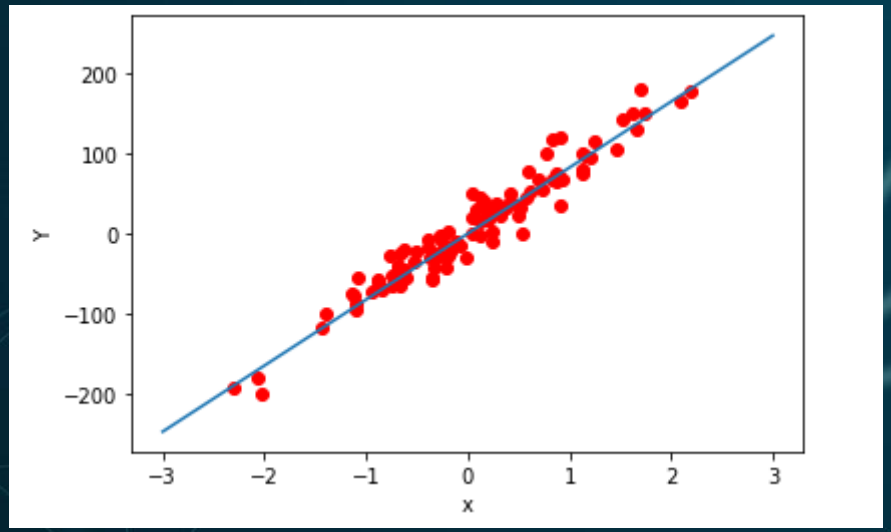
גלעד מרקמן

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

קרוב ע"י קו ישר - רגרסיה לינארית

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- ברגרסיה לינארית אנחנו מבקשים למצור את הקשר בין X ל- Y ומניחים כי הקשר הוא לינארי – קו ישר. לדוגמה בנתונים הבאים

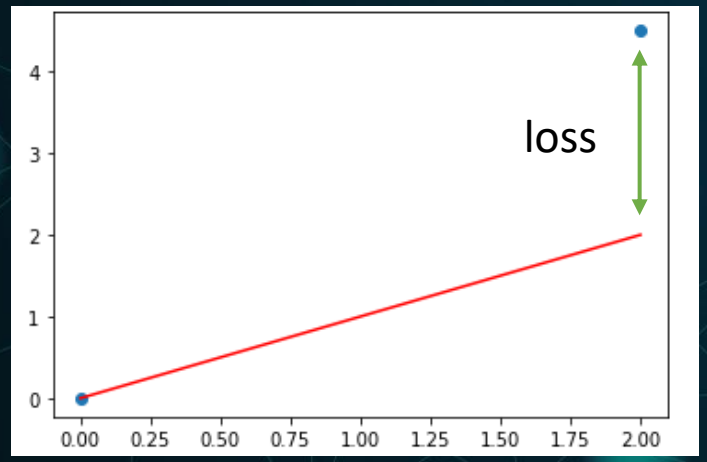


- רגרסיה לינארית היא אלגוריתם שנועד למצוא את הקו הישר המתאר בצורה טובה ביותר את היחס בין משתנה X ל Y.

רגרסיה לינארית בנקודה אחת

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- נדגים את האלגוריתם של רגרסיה לינארית במקרה פשוט שניתן לפתור בקלות באמצעות חישוב אלגברי.
- השאלה: נתונה הנקודה (2, 4.5) ועלינו למצוא את פונקציית קו הישר העוברת בראשית הצירים ובנקודה זו: $y = w \cdot x$.
- למעשה אנחנו צריכים למצוא את w.

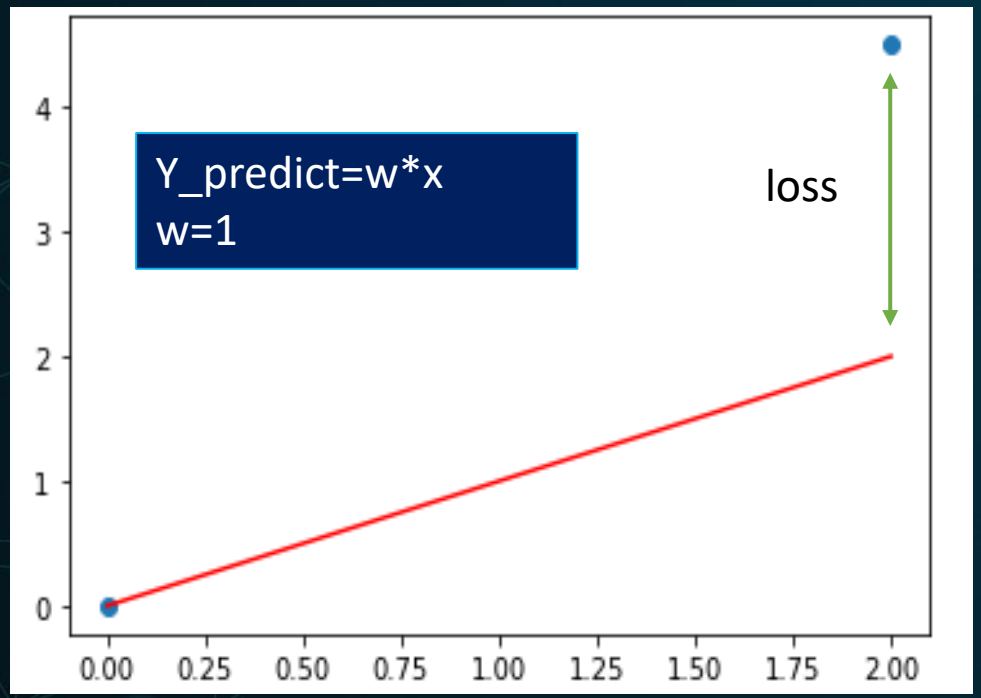


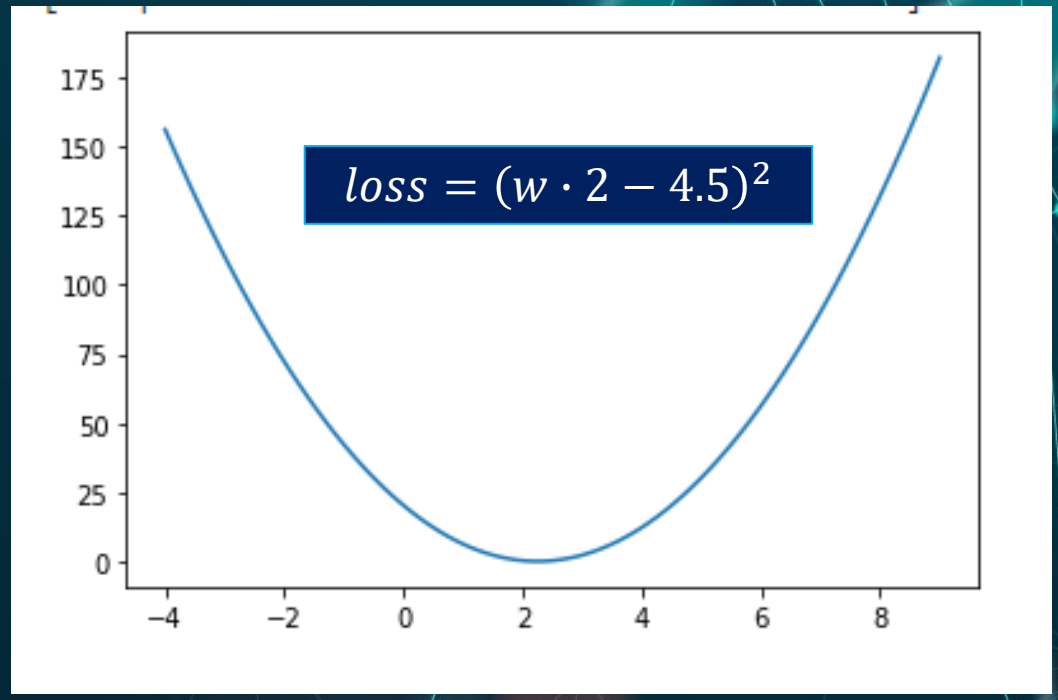
- האלגוריתם של Gradient descent:
  - נתחיל מ w שרירותי למשל 1.
  - נבדוק את הטעות בין y = w*2 = 1*2 לבין התוצאה המצופה שהיא 4.5.
  - הטעות תחושב כ: $loss = \left(y_{predict} - y\right)^2 = (w \cdot 2 - 4.5)^2$
  - נמצא את w שיקטין את הטעות למינימום.

רגרסיה לינארית בנקודה אחת

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- פונקציית הטעות לנקודה (2,4.5)
- אנו מחפשים את ערך ה w שיקטין את הטעות למינימום.





האלגוריתם

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- האלגוריתם בנוי מן השלבים הבאים:

- Prepare data
- Design Model: model = w * x
- Build loss and optimizer
- Training loop:
  - Forward pass : compute y_predict
  - Backward pass: compute loss and gradients
  - Update weight (then zero grads)
- Print results

האלגוריתם - המשך

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- הכנת הנתונים:

```
x = 2
y = 4.5
```

- בניית המודל:

Design Model

```
X = torch.tensor(x)
Y = torch.tensor(y)
W =  torch.tensor(1.2, dtype=torch.float32, requires_grad=True)
learning_rate = 0.01

def model (X):
  return W * X
```

- בניית פונקציית המחיר והאופטימיזר:

construct loss and optimizer

```
[30] def loss (Y_predict, Y):
       return (Y_predict - Y)**2

     optim = torch.optim.SGD([W], lr=learning_rate)
```

אימון המודל

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

Training loop

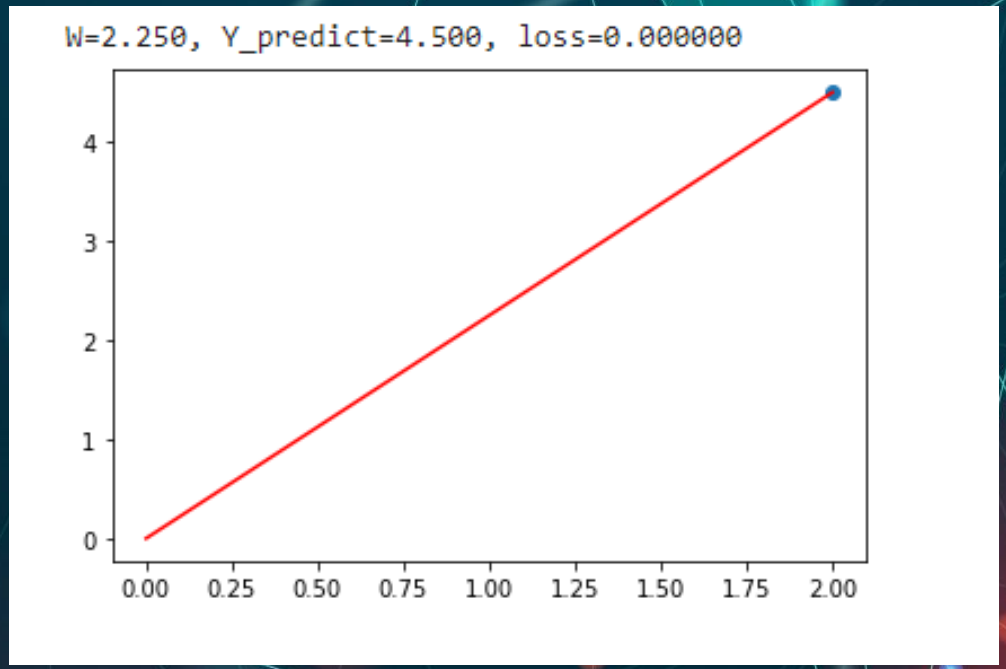
```
[54] for epoch in range(120):
       # Forward
       Y_predict = model(X)

       # Backward - zero grads + calculate new grads
       l = loss(Y_predict, Y)

       l.backward()

       # optimize wights
       optim.step()

       if (epoch+1) % 10 == 0:
         print (f"epoch={epoch} W={W:.3f}, Y_predict={Y_predict:.3f}, loss={l:.6f} ")

       # zero grads
       optim.zero_grad()
```

```
epoch=9 W=1.794, Y_predict=3.508, loss=0.983149
epoch=19 W=2.052, Y_predict=4.069, loss=0.185514
epoch=29 W=2.164, Y_predict=4.313, loss=0.035005
epoch=39 W=2.213, Y_predict=4.419, loss=0.006605
epoch=49 W=2.234, Y_predict=4.465, loss=0.001246
epoch=59 W=2.243, Y_predict=4.485, loss=0.000235
epoch=69 W=2.247, Y_predict=4.493, loss=0.000044
epoch=79 W=2.249, Y_predict=4.497, loss=0.000008
epoch=89 W=2.249, Y_predict=4.499, loss=0.000002
epoch=99 W=2.250, Y_predict=4.499, loss=0.000000
epoch=109 W=2.250, Y_predict=4.500, loss=0.000000
epoch=119 W=2.250, Y_predict=4.500, loss=0.000000
```

הדפסת התוצאות

- כפי שניתן לראות. האלגוריתם מצא את w=2.5. והגרף עובר בדיוק בנקודה.
- המחיר כאשר w=2.5 הוא 0 מינימלי.

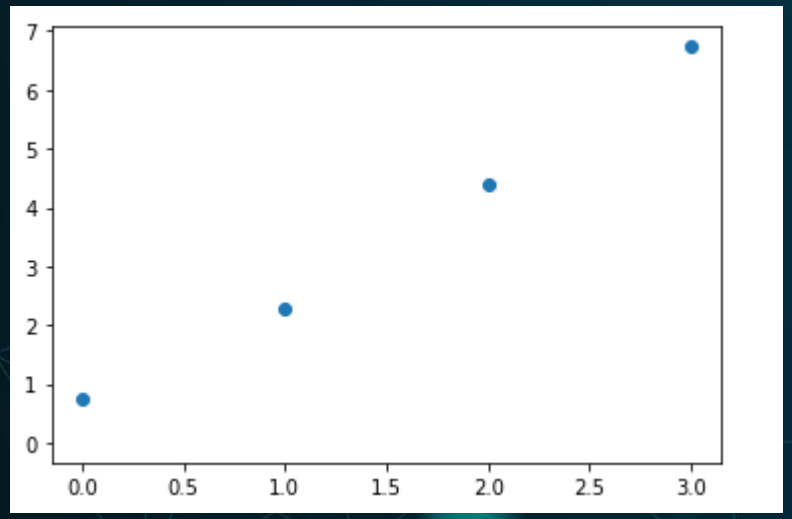
```
Y_predict = model(X)
print (f"W={W:.3f}, Y_predict={Y_predict:.3f}, loss={l:.6f}")

plt.scatter(x,y)
plt.plot([0,x], [0,Y_predict.item()], color = 'red')
# plt.plot([0, 2], [0, 2.4], color='b')
plt.show()
```

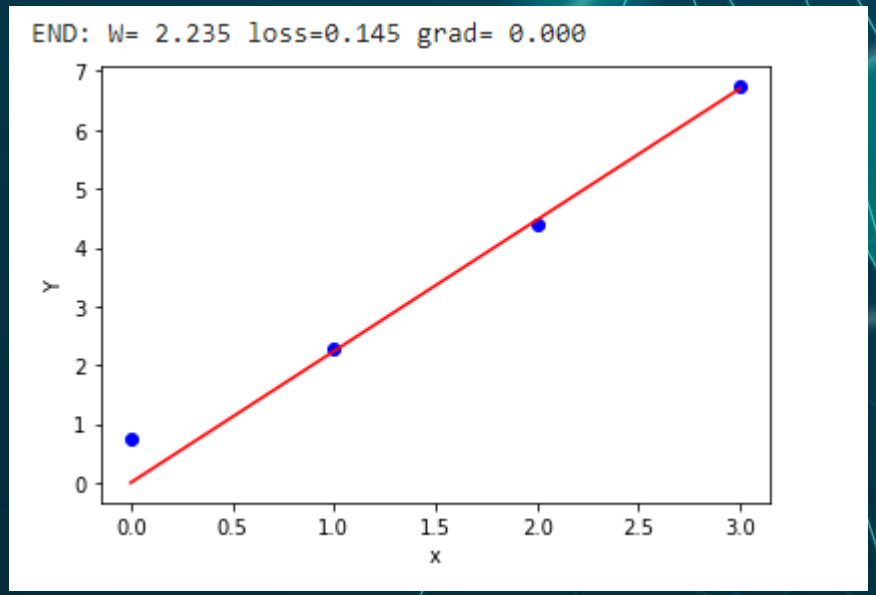


רגרסיה לינארית עם ארבע נקודות

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- נתונים לנו הנתונים הבאים:
- X = 0, 1, 2, 3 Y = 0.7576, 2.2793, 4.4031, 6.7347

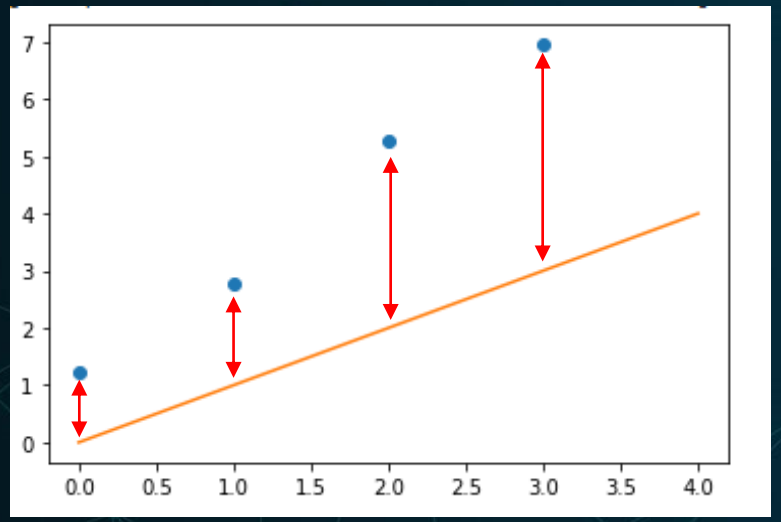




- עלינו למצוא את נוסחת הקו הישר (המתחיל בראשית הצירים) שמתאר בצורה הטובה ביותר את הנקודות הללו.

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

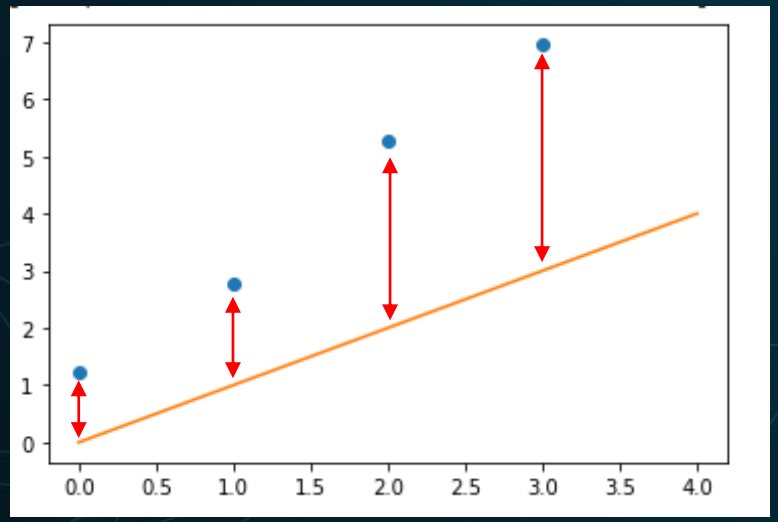
הרעיון



- ידוע לנו כי נוסחת הקו (העובר בראשית הצירים) הוא $y = wx$.
- נבחר רנדומלית ב w=1.0. נקבל: y_predict = 1.0 * X.
- נחשב את ממוצע הטעויות לפי ההנחה שלנו.
  - נגדיר את הטעות: $(y_predict - y)^2$
  - נעלה בריבוע כיוון שאנחנו רוצים את הטעות כמספר חיובי בלבד.
  - ממוצע הטעות - סכום הטעויות לכל הנקודות חלקי מספר הנקודות.
- פונקציית הטעות היא אם כן:
  - $loss = \frac{1}{n}(y_{predict} - y)^2$
  - $loss = \frac{1}{n}(wx - y)^2$
- נמצא את ה w שיביא את פונקציית הטעות למינימום בעזרת Gradient Descent.
- שימו לב – המשתנה שלנו הוא w ולא X.
- אם מצאנו את ה w שיקטין את ה loss למינימום, מצאנו את נוסחת הקו שהכי קרובה לנקודות שלנו.

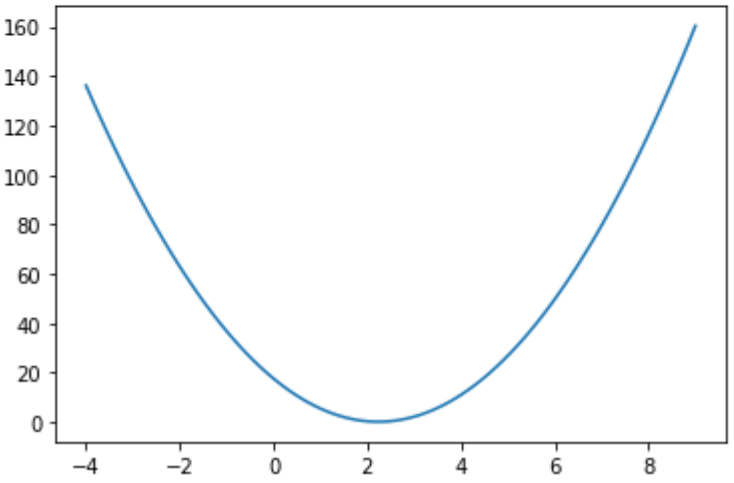
פונקציית המחיר

- נתונים לנו הנתונים הבאים:
  - X = 0, 1, 2, 3 Y = 0.7576, 2.2793, 4.4031, 6.7347




```
W = torch.tensor(np.linspace(-4, 9, 100))
loss = torch.zeros(100)
for i in range(W.size()[0]):
  loss[i] = ((W[i]*X - Y)**2).mean()

plt.plot(W.numpy(), loss.numpy())
```

```
[<matplotlib.lines.Line2D at 0x7f03f21caac0>]
```



הקוד לרגרסיה לינארית

האלגוריתם שלנו יהיה בנוי בהתאם לשלבים הבאים:

- הכנת הנתונים
  - בחירת פרמטרים אקראיים - w
  - בחירת קצב הלמידה learning_rate
- בניית המודל. במקרה זה הוא מודל לינארי $y_{predict} = wx$
- בניית פונקציית המחיר והאופטמיזציה
  - פונקציית המחיר היא $loss = \frac{1}{n}\left(y_{predict} - y\right)^2$
  - האופטמיזציה נעשית באמצעות Gradient descent = SGD
- לולאה (למציאת המינימום)
  - חלחול קדימה – לחישוב הערכים הצפויים.
  - חלחול לאחור וחישוב הנגזרת.
  - עדכון הפרמטרים

הקוד א'

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- הכנת הנתונים:

prepare Data

```
torch.manual_seed(1)
X = torch.tensor([0, 1, 2, 3])
Y = X * 2 + torch.rand(4,)
```

- בניית המודל הלינארי:

Design Model

```
[118] W = torch.tensor(0.0, requires_grad=True)
      learning_rate = 0.01

      def Model (X):
        return W * X
```

- בניית פונקציית המחיר והאופטימיזר:

construct loss and optimizer

```
[119] def Loss (Y_predict, Y):
        return ((Y_predict - Y)**2).mean()

      optimizer = torch.optim.SGD([W], lr=learning_rate)
```

הקוד ב'

- לולאת האימון:

Training loop

```
[59] for epoch in range(200):

       # forward
       Y_predict = Model(X)

       # backward
       loss = Loss(Y_predict, Y)
       loss.backward()

       # update wights
       optimizer.step()

       if epoch % 10 == 0:
         print(f"epoch= {epoch} W= {W.item():.3f} loss={loss.item():.3f} grad= {W.grad.item():.3f}")

       # zero grads
       optimizer.zero_grad()
```

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

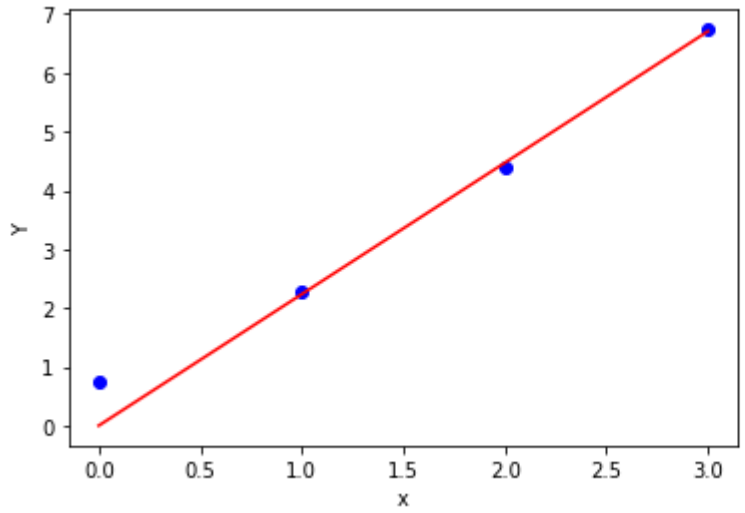
הקוד ג'

- הדפסת תוצאות החישוב:

print results

```
[121]
print(f"END: W= {W.item():.3f} loss={loss.item():.3f} grad= {W.grad.item():.3f}")
plt.scatter(X, Y, color='b')
plt.xlabel("x")
plt.ylabel("Y")
with torch.no_grad():
  plt.plot (X, Model(X), color='red')
  pass
plt.show()
```

```
END: W= 2.235 loss=0.145 grad= 0.000
```



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

תוצאות האימון

```
epoch= 0 W= 0.156 loss=17.628 grad= -15.645
epoch= 10 W= 1.229 loss=4.240 grad= -7.572
epoch= 20 W= 1.748 loss=1.105 grad= -3.665
epoch= 30 W= 1.999 loss=0.370 grad= -1.774
epoch= 40 W= 2.121 loss=0.198 grad= -0.858
epoch= 50 W= 2.180 loss=0.158 grad= -0.415
epoch= 60 W= 2.208 loss=0.148 grad= -0.201
epoch= 70 W= 2.222 loss=0.146 grad= -0.097
epoch= 80 W= 2.229 loss=0.145 grad= -0.047
epoch= 90 W= 2.232 loss=0.145 grad= -0.023
epoch= 100 W= 2.233 loss=0.145 grad= -0.011
epoch= 110 W= 2.234 loss=0.145 grad= -0.005
epoch= 120 W= 2.235 loss=0.145 grad= -0.003
epoch= 130 W= 2.235 loss=0.145 grad= -0.001
epoch= 140 W= 2.235 loss=0.145 grad= -0.001
epoch= 150 W= 2.235 loss=0.145 grad= -0.000
epoch= 160 W= 2.235 loss=0.145 grad= -0.000
epoch= 170 W= 2.235 loss=0.145 grad= -0.000
epoch= 180 W= 2.235 loss=0.145 grad= -0.000
epoch= 190 W= 2.235 loss=0.145 grad= -0.000
```

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

סיכום – האלגוריתם לרגרסיה לינארית

קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- הכנת הנתונים
  - במקרה שלנו קבלת 4 הנקודות
- בניית המודל.
  - במקרה זה הוא מודל לינארי $y_{predict} = wx$
  - בחירת פרמטרים אקראיים - w
  - בחירת קצב הלמידה learning_rate
- בניית פונקציית המחיר והאופטמיזציה
  - פונקציית המחיר היא $loss = \frac{1}{n}\left(y_{predict} - y\right)^2$
  - האופטמיזציה נעשית באמצעות Gradient descent = SGD
- לולאה (למציאת המינימום)
  - חלחול קדימה – לחישוב הערכים הצפויים
  - חלחול לאחור - חישוב ההפסד loss וחישוב הנגזרת.
  - עדכון הפרמטרים