



Linear Regression

ספריית חח

גלעד מרקמן



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות



תזכורת - הקוד לרגרסיה לינארית

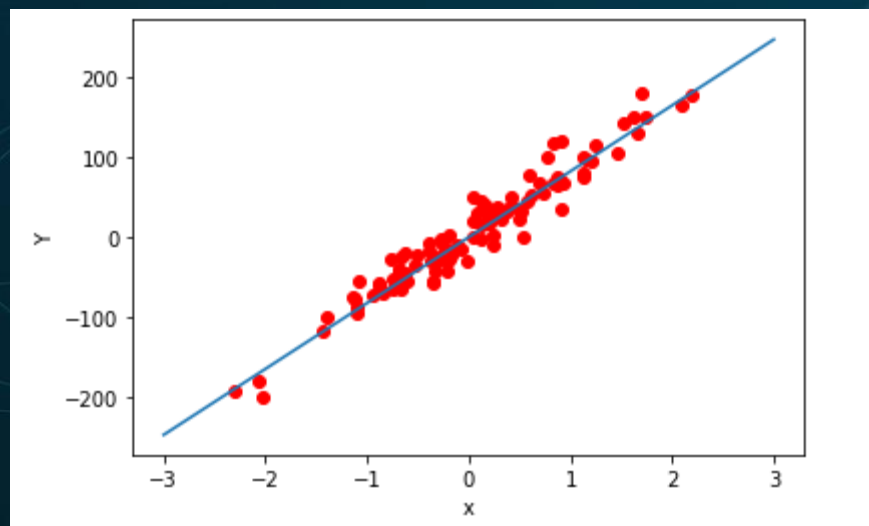
- הכנת הנתונים
 - בחירת פרמטרים אקראיים - w
 - בחירת קצב הלמידה $learning_rate$
- בניית המודל. במקרה זה הוא מודל לינארי $y_{predict} = wx$
- בניית פונקציית המחיר והאופטימיזציה
 - פונקציית המחיר היא $loss = (y_{predict} - y)^2.mean()$
 - האופטימיזציה נעשית באמצעות Gradient descent = SGD
- לולאה (למציאת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

תרגיל רגרסיה לינארית



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

- נבצע את האלגוריתם לחישוב רגרסיה לינארית עם מספר רב של פרמטרים.
- נתונים לנו 100 נקודות (x, y) ועלינו למצוא את הקו הישר שמתאר בצורה הטובה ביותר את אוסף הנקודות.





הכנת הנתונים

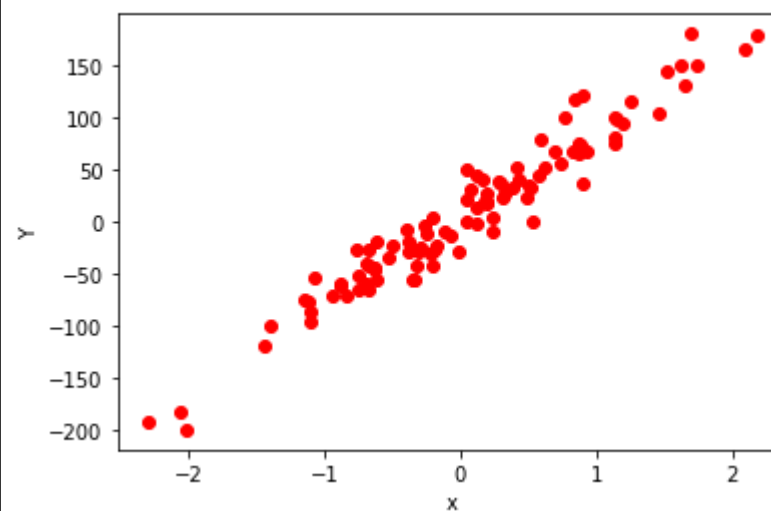
- ניצור 100 נקודות באמצעות פונקציה היוצרת נקודות המתאימות לרגסיה לינארית. הפונקציה מחזירה מערך של X ומערך של Y .

Prepare data

[96]

```
x_np, y_np = datasets.make_regression(n_samples=100, n_features=1, noise=20, random_state=1)
print(x_np.shape, y_np.shape)
plt.scatter(x_np, y_np, color='red')
plt.xlabel("x")
plt.ylabel("Y")
plt.show()
```

(100, 1) (100,)





המרת הנתונים ל Tensors

- עלינו להמיר את הנתונים ל- `torch.tensor`, במבנה תואם של 100 שורות.
- `X_np` מתקבל במבנה המתאים.
- `Y_np` מתקבל במבנה של שורה אחת עם 100 ערכים, ויש לשנות אותו ל- 100 שורות.

```
# prepare tensors

X = torch.from_numpy(x_np.astype(np.float32))
Y = torch.from_numpy(y_np.astype(np.float32))
Y = Y.view(Y.shape[0],1)

print (X.shape, X.dtype)
print(Y.shape, Y.dtype)

torch.Size([100, 1]) torch.float32
torch.Size([100, 1]) torch.float32
```


פרמטרים, מודל, פונקציית טעות, ואופטימיזר ללא torch.nn



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

```
W = torch.tensor([0.0], requires_grad=True)
learning_rate = 0.1
epochs = 100
losses = []

# design model
def Model(X):
    return W * X

#construct loss and optimizer
def Loss (Y_predict, Y):
    return ((Y_predict-Y)**2).mean()

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

רשימה לשמירת תוצאות

מודל לינארי העובר בראשית
הצירים

לולאת האימון



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

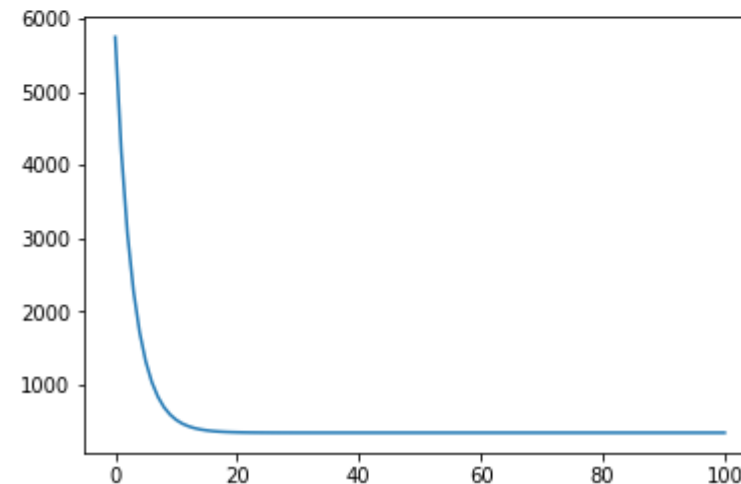
training loop

```
[139] for epoch in range(epochs+1):  
    # 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} loss={loss.item():.3f} ")  
  
    losses.append(loss.item())  
  
    # zero wights  
    optimizer.zero_grad()
```

```
epoch= 0 loss=5745.195  
epoch= 10 loss=524.382  
epoch= 20 loss=354.631  
epoch= 30 loss=349.112  
epoch= 40 loss=348.932  
epoch= 50 loss=348.926  
epoch= 60 loss=348.926  
epoch= 70 loss=348.926  
epoch= 80 loss=348.926  
epoch= 90 loss=348.926  
epoch= 100 loss=348.926
```

```
plt.plot (losses)
```

```
[<matplotlib.lines.Line2D at 0x7ff822987d00>]
```

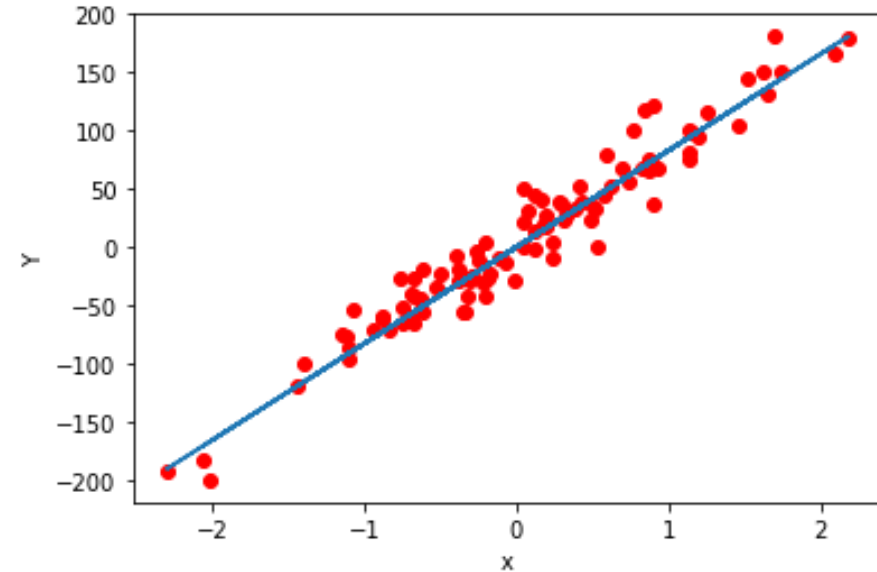


הדפסת התוצאות

print result

```
[151] print(f"END: loss={loss.item():.4f}")  
      plt.scatter(x_np, y_np, color='red')  
      plt.xlabel("x")  
      plt.ylabel("Y")  
  
      with torch.no_grad():  
          predicted = Model(X).detach().numpy()  
          plt.plot(x_np, predicted)  
          plt.show()
```

END: loss=348.9261



$$Y = 82.7965 \cdot X$$

```
[152] print(W)  
      # for name, param in Model.named_parameters():  
      #     print(name, param)  
  
      tensor([82.7965], requires_grad=True)
```




קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

שימוש ב torch.nn

פרמטרים, מודל, פונקציית טעות, ואופטימיזר ללא torch.nn



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

```
[158] learning_rate = 0.1
      epochs = 100
      losses = []

# design model
Model = nn.Linear(1, 1, bias=False)

#construct loss and optimizer
Loss = nn.MSELoss()

# init optimizer
optimizer = torch.optim.SGD(Model.parameters(), lr=learning_rate)
```

```
print wights
```

```
[162] # print(W)
      for name, param in Model.named_parameters():
      .. print(name, param)
```

```
weight Parameter containing:
tensor([[82.7965]], requires_grad=True)
```



Bias

- נוסחת פונקציה לינארית היא: $y = wx + b$.
- המשתנה b נקרא $bias$, והוא מהווה את הסטיה מראשית הצירים.
- עד כה חיפשנו את המשקל בלבד ללא התחשבות ב $bias$, והנחנו כי הגרף המבוקש עובר בראשית הצירים.
- ניתן להגדיר את המודל כל שיחושב בנוסף ה $bias$, בשינוי קטן בלבד.

```
# design model  
Model = nn.Linear(1, 1, bias=True)
```

$$Y = 82.4845 \cdot X + 4.0540$$

```
print wights
```

```
[172] # print(w)  
for name, param in Model.named_parameters():  
    print(name, param)
```

```
weight Parameter containing:  
tensor([[82.4845]], requires_grad=True)  
bias Parameter containing:  
tensor([4.0540], requires_grad=True)
```

