



רגרסיה לינארית ונרמול נתונים גלעד מרקמן



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות



דוגמה מחירי דירות

- בסקר שנערך בעיר נס ציונה נבדקו מחירי הדירות לפי שטח הדירה והקומה. התוצאות נתונות בטבלה למטה:
- עליכם למצוא את הנוסחה לחישוב מחיר הדירה בהתאם לשטחה (בשלב זה אנו מתעלמים מהקומה).

מחיר הדירה במיליוני ₪	קומה	שטח הדירה
2.462	3	90
2.661	2	100
3.177	9	105
2.972	3	110
3.321	7	115
3.226	3	120
3.648	6	130
3.447	1	133
3.848	5	140



תזכורת - האלגוריתם לרגרסיה לינארית

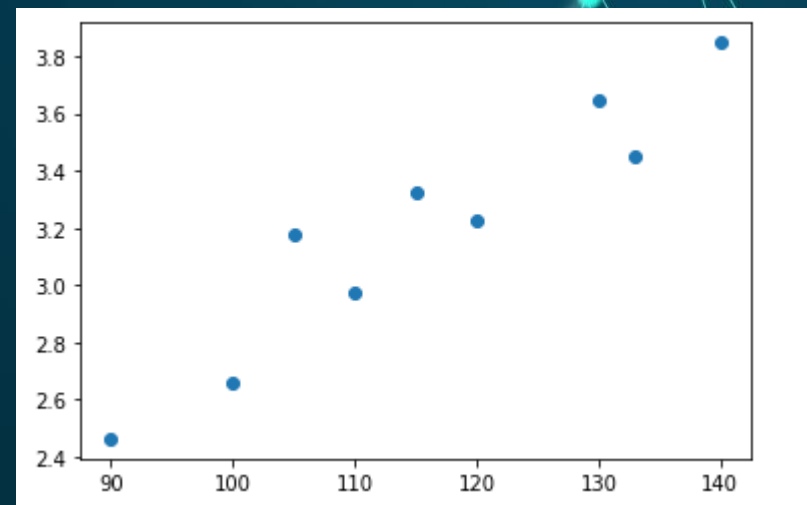
- הכנת הנתונים
 - בחירת פרמטרים אקראיים – w (נעשה אוטמטית על ידי `nn.Linear`)
 - בחירת קצב הלמידה `learning_rate`
 - בניית המודל. במקרה זה הוא מודל לינארי `nn.Linear`
 - בניית פונקציית המחיר והאופטימיזציה
 - פונקציית המחיר היא `nn.MSELoss()`
 - האופטימיזציה נעשית באמצעות `Gradient descent = SGD`
 - לולאה (למציאת המינימום)
 - חלחול קדימה – לחישוב הערכים הצפויים.
 - חלחול לאחור וחישוב הנגזרת.
 - עדכון הפרמטרים.
 - הדפסת התוצאות



הכנת הנתונים

- הנתונים התקבלו בתקבלו במערכים של numpy.

```
X_np = np.array([90,100,105,110,115,120,130,133,140])  
F_np = np.array([3,2,9,3,7,3,6,1,5])  
P_np = np.array([2.462,2.661,3.177,2.972,3.321,3.226,3.648,3.447,3.848])  
# P_np = np.around((X_np * 25.5 + F_np * 55.5)/1000, decimals=3)  
plt.scatter(X_np, P_np)
```



- המרה לטנסורים ועיצוב הטנסורים. יש לשים לב כי כל דירה צריכה להיות עמודה אחת.

```
X = torch.from_numpy(X_np.astype(np.float32))  
Y = torch.from_numpy(P_np.astype(np.float32))  
X = X.view(X.shape[0],1)  
Y = Y.view(Y.shape[0],1)
```

```
X: torch.Size([9, 1]) torch.float32  
Y: torch.Size([9, 1]) torch.float32
```

פרמטרים, מודל, פונקציית המחיר, והאופטימיזר



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

```
learning_rate = 0.1 #0.000001
epochs = 100
losses = []

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

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

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


לולאת האימון



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

```
for epoch in range(epochs):
    # forward
    Y_predict = Model(X)

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

    # update wights
    optim.step()

    if epoch % 10 == 0:
        print(f"epoch={epoch} loss={loss.item():.4f} ")
        # print (Y_predict)

    losses.append(loss.item())

    # zero wights
    optim.zero_grad()
```

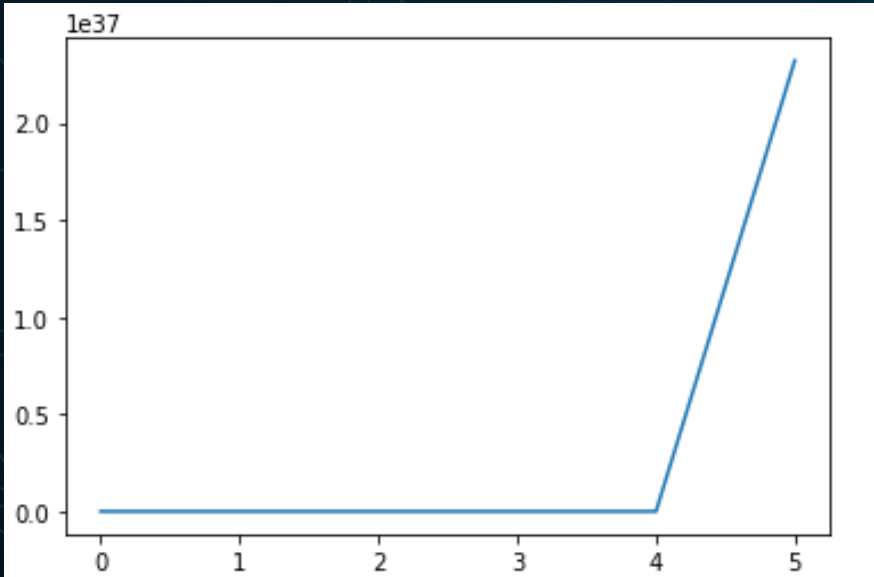
תוצאות האימון – המודל נכשל



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

```
epoch= 0 loss=1505.2563  
epoch= 10 loss=inf  
epoch= 20 loss=nan  
epoch= 30 loss=nan  
epoch= 40 loss=nan  
epoch= 50 loss=nan  
epoch= 60 loss=nan  
epoch= 70 loss=nan  
epoch= 80 loss=nan  
epoch= 90 loss=nan
```

```
epoch= 0 loss=4558.2754  
epoch= 1 loss=34056773632.0000  
epoch= 2 loss=254453636644470784.0000  
epoch= 3 loss=1901139458635015871528960.0000  
epoch= 4 loss=14204277544339544332097739030528.0000  
epoch= 5 loss=inf  
epoch= 6 loss=inf  
epoch= 7 loss=inf  
epoch= 8 loss=inf  
epoch= 9 loss=inf  
epoch= 10 loss=inf  
epoch= 11 loss=inf  
epoch= 12 loss=nan  
epoch= 13 loss=nan  
epoch= 14 loss=nan  
epoch= 15 loss=nan  
epoch= 16 loss=nan  
epoch= 17 loss=nan  
epoch= 18 loss=nan  
epoch= 19 loss=nan
```



vanishin / exploding gradients



קריית החינוך
פארק המדע
בית לערכים
למצוינות ולחדשנות

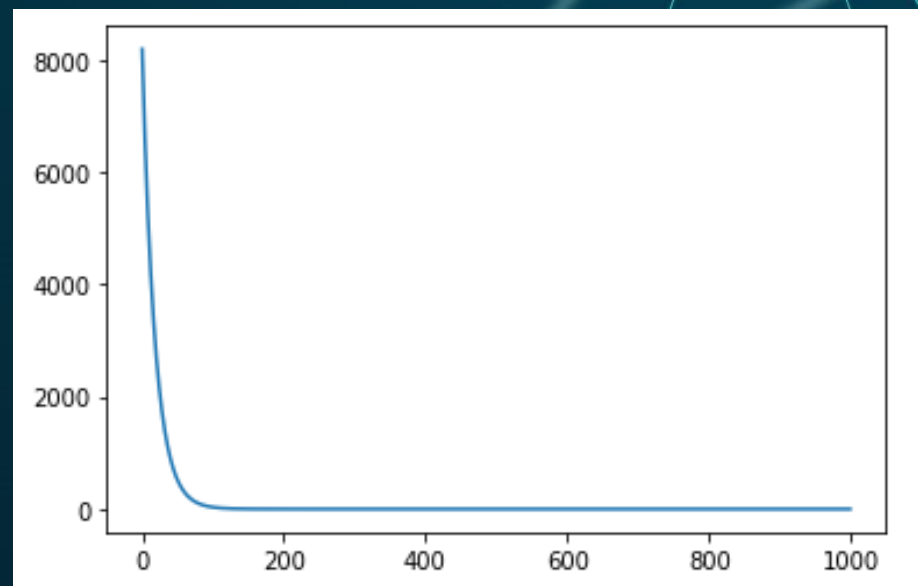
- נתקלנו בתופעה המכונה העלמות / פיצוץ של גרדיאנים. כפי שניתן לראות הערכים של הגרדיאנים היו גדולים מידי או קרובים מידי לאפס, מה שהביא לערכים גדולים מידי או ערכים קרובים לאפס של פונקציית המחיר.
- המודל נכשל כיוון שלא הצליח להתמודד עם ערכים גדולים כל כך או קרובים מידי לאפס.
- הסיבה לכך הינה שערכי שטח הדירה היו גדולים (בין 90 – 140) מה שהביא לנגזרות מאוד גבוהות.



פתרונות שינוי פרמטרים

- הקטנת קצב הלמידה והגדלת מספר האימונים (epochs).
- שינוי קצב הלימוד: $\text{learning_rate} = 0.000001$
- שינוי מספר האימונים: $\text{epochs} = 1000$.

```
epoch= 0 loss=8185.0425  
epoch= 100 loss=32.0105  
epoch= 200 loss=0.1555  
epoch= 300 loss=0.0311  
epoch= 400 loss=0.0306  
epoch= 500 loss=0.0306  
epoch= 600 loss=0.0306  
epoch= 700 loss=0.0306  
epoch= 800 loss=0.0306  
epoch= 900 loss=0.0306
```





שינוי האופטימיזר

- ל- PyTorch ישנם אופטימזרים נוספים מלבד SGD. אחד מהם נקרא Adam, שהינו שיפור של Gradient Descent.
- במרבית המקרים האופטימיזר של Adam נותן תוצאות טובות יותר.
- יחד עם זאת, הוא לא פותר תמיד את הבעיה שהצגנו לעיל.

```
learning_rate = 0.1 # 0.000001
epochs = 200
losses = []

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

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

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

```
epoch= 0 loss=6299.0166
epoch= 10 loss=359.4584
epoch= 20 loss=134.9965
epoch= 30 loss=158.4764
epoch= 40 loss=11.8144
epoch= 50 loss=3.3893
epoch= 60 loss=6.6565
epoch= 70 loss=2.7275
epoch= 80 loss=0.5347
epoch= 90 loss=0.0610
epoch= 100 loss=0.0269
epoch= 110 loss=0.0328
epoch= 120 loss=0.0319
epoch= 130 loss=0.0292
epoch= 140 loss=0.0274
epoch= 150 loss=0.0266
epoch= 160 loss=0.0262
epoch= 170 loss=0.0261
epoch= 180 loss=0.0260
epoch= 190 loss=0.0260
```



נרמול נתונים

- הדרך הטובה ביותר היא לנרמל את הנתונים כך שיהיו בטווח שבין 0-1 או בין 0.5 – 0.5.

- נרמול הנתונים שומר על היחס בין הנתונים ולכן לא פוגע בתוצאות המודל. יחד עם זאת, כאשר אנחנו מבקשים לבדוק את המודל עלינו לנרמל גם את נתוני הבדיקה (שכן המודל אומן על נתונים מנורמלים).

- קיימות שתי דרכים עיקריות לנרמול הנתונים:

- minMax : לנתונים שידוע שנמצאים בטווח מסויים נשתמש בנוסחה הבאה.

```
def Normalize_minMax(X , max, min):  
    return (X - min) / (max - min)
```

- Z-normalization : הנעזר בממוצע ובסטיית תקן.

```
def Normalize_z(X, mean, std):  
    return (X - mean) / std
```



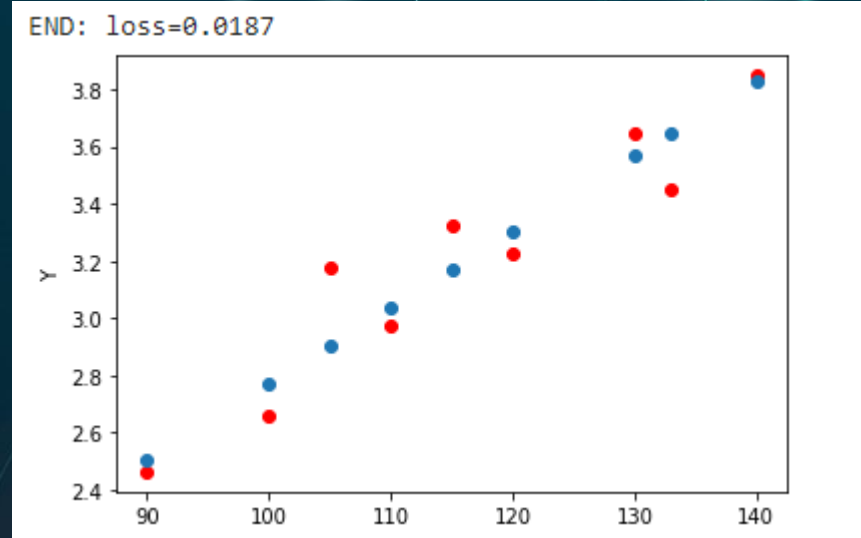
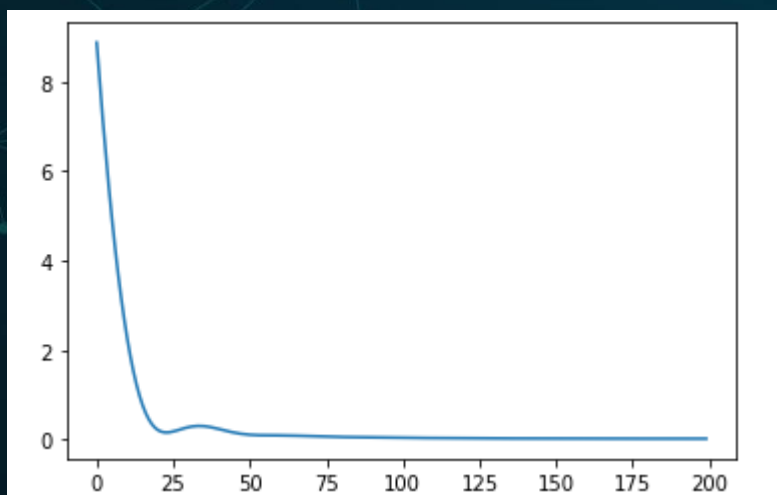

שימוש בנרמול של הנתונים

• נרמול הנתונים עשוי למונע את כשלון המודל ומייעל את הזמן הנדרש לאמן את המודל. לכן יש לנרמל תמיד את הנתונים.

- נרמול הנתונים נעשה בעת הכנת הנתונים:
- והתוצאות הן טובות יותר:

```
X = torch.from_numpy(X_np.astype(np.float32))  
Y = torch.from_numpy(P_np.astype(np.float32))  
  
# Normalize data  
max = X.max()  
min = X.min()  
X = Normalize_minMax(X, max, min)
```

```
epoch= 0 loss=8.8723  
epoch= 20 loss=0.2117  
epoch= 40 loss=0.2335  
epoch= 60 loss=0.0926  
epoch= 80 loss=0.0569  
epoch= 100 loss=0.0388  
epoch= 120 loss=0.0283  
epoch= 140 loss=0.0228  
epoch= 160 loss=0.0202  
epoch= 180 loss=0.0191
```





סיכום

- תרגלנו רגרסיה לינארית על דוגמה פרקטית.
- הבנו את הצורך בנרמול הנתונים לפני הכנסתם למודל.
- עדיין, המודל שלנו התייחס אך ורק לשטח הדירה ולא התייחס לקומה. בכך, בוודאי שהוא לא יכול להיות מדוייק מאוד.
- בשיעור הבא נלמד כיצד לבצע רגרסיה לינארית למספר משתנים.



קריית החינוך
פארק המדע
בית לערכים
למציאות ולחדשנות