לא לשכוח להפעיל הקלטה!

מה נלמד בשיעור זה?

- ✓ רגרסיה לינארית **עם יותר ממשתנה אחד**
- ✓ gradient descent עבור רגרסיה ליניארית יותר ממשתנה אחד
- ✓ רגרסיה לינארית עם ספריית scikit-learn
- ✓ נירמול הנתונים

תזכורת:
רגרסיה לינארית עם משתנה אחד

שלבים למציאת המשקולות באמצעות gradient descent

- ✓ 1. איתחול W, b - ראשית אנו מאתחלים את פרמטרי המודל בערכים רנדומלים
- ✓ 2. חישוב הנגזרת החלקית ל - W, b
- ✓ 3. עידכון b – הערך החדש של b שווה לערך הקודם של b פחות הנגזרת של b כפול גודל הצעד
- ✓ 4. עידכון w – הערך החדש של w שווה לערך הקודם של w פחות הנגזרת של w כפול גודל הצעד
- ✓ 5. בדיקת ביצועים – חישוב r^2

אנו חוזרים על שלבים 2-4 עד שפונקציית העלות תתכנס לערך המינימלי

פונקציית המחיר J עם משתנה אחד

- בהינתן סט של m דוגמאות $(x^{(i)},y^{(i)})$
  - x_i - ערכי דגימות של המשתנה הבלתי תלוי
  - y_i – ערכי דגימות המשתנה התלוי בהתאמה

- ובהינתן מודל רגרסיה $\hat{y} = w \cdot x + b$
- נגדיר את שגיאת החיזוי על כל סט נתוני האימון:

$$J = \frac{1}{m}\sum_{i=1}^{m} l^{(i)} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{m}\sum_{i=1}^{m}(w \cdot x^{(i)} + b - y^{(i)})^2$$

פונקציית המחיר J היא פונקציה של W,B

$$J(w,b)=\frac{1}{m}\sum_{i=1}^{m}(\underbrace{w\cdot x^{(i)}+b}_{\text{הערך החזוי}}-\underbrace{y^{(i)}}_{\text{הערך האמיתי}})^{2}$$

המטרה – למזער את הטעות עבור ערכי w,b שונים

- תשובות המודל w*x+b
- הנתונים האמיתיים y
- m – מספר הדגימות
- i – אינדקס שרץ על כל הדגימות

Cost Function

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(\underbrace{w\cdot x^{(i)} + b}_{\text{הערך החזוי } \hat{y}} - \underbrace{y^{(i)}}_{\text{הערך האמיתי}})^2$$

MSE (MEAN SQUARED ERROR)

אלגוריתם GRADIENT DESCENT עבור פונקציה רב ממדית

- על מנת למצוא את נקודת המינימום של פונקציה J(w,b):
  - נגריל נקודת התחלה w_0,b_0
  - נקדם את ערכי w,b לפי הנוסחאות:

$$w \leftarrow w - \alpha \frac{\partial J(w,b)}{\partial w}$$

$$b \leftarrow b - \alpha \frac{\partial J(w,b)}{\partial b}$$

פונקציית המחיר ונגזרותיה

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(w \cdot x^{(i)} + b - y^{(i)})^2$$

$$\frac{\partial}{\partial w}J(w,b) = \frac{1}{m}\sum_{i=1}^{m}2\left(w \cdot x^{(i)} + b - y^{(i)}\right)x^{(i)}$$

$$\frac{\partial}{\partial b}J(w,b) = \frac{1}{m}\sum_{i=1}^{m}2\left(w \cdot x^{(i)} + b - y^{(i)}\right)$$

```
def J(w,b,X,y):
  y_hat = X*w+b
  return np.sum((y_hat - y)**2) / len(y)
```

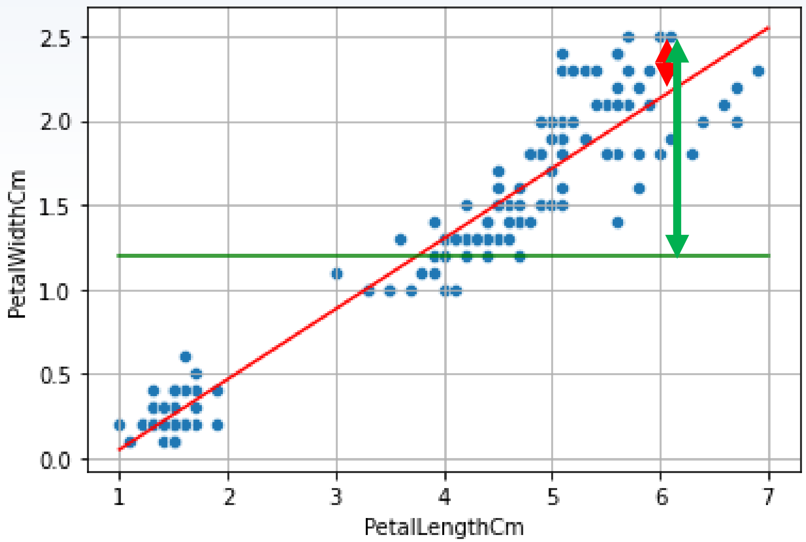
```
def dw_(w,b,X,y):
  y_hat = X*w+b
  return 2 * np.sum((y_hat - y)*X) / len(y)
```

```
def db_(w,b,X,y):
  y_hat = X*w+b
  return 2 * np.sum(y_hat - y) / len(y)
```

הערכת ביצועים של רגרסיה לינארית
מדד השונות המוסברת R^2

$$SS_{tot} = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \bar{y}\right)^2$$

$$SS_{error} = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$



$$R^2 = 1 - \frac{SS_{error}}{SS_{tot}}$$

ערכים נעים (לרוב) בין 0-1

1: התאמה מלאה

0: אין התאמה

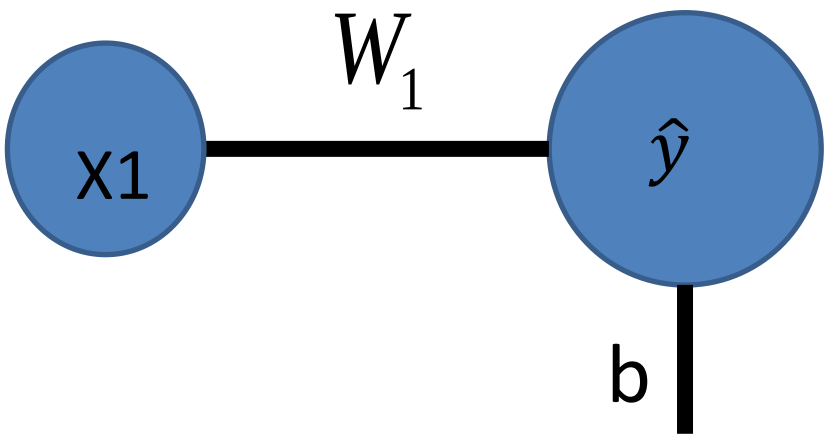
ערכים שליליים: חיזוי יותר גרוע מאשר מדד הממוצע

רגרסיה לינארית כנוירון

הנוירון מקבל קלט יחיד x1

הפלט של הנוירון – הקלט המשוקלל ועוד $W_1 * X_1 + b$

אין הפעלה של פונקציית מעבר



$\hat{y} = W_1 * X_1 + b$

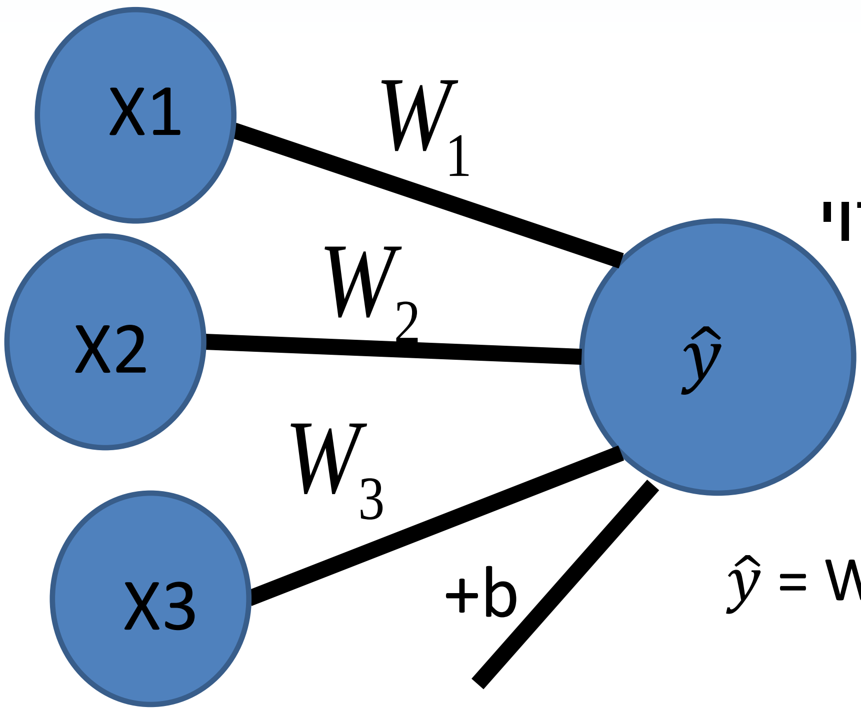
רגרסיה לינארית עם יותר משתנה אחד

רגרסיה ליניארית עם יותר ממשתנה אחד

נניח כעת שמחיר הבית תלוי לא רק בשטחו במ"ר, אלא גם במידת השיפוץ שלו, במדד של איכות החינוך בסביבתו וכו'

מספר נתוני הקלט: 3

אנחנו נקרא לקלטים השונים הללו בשם features (בעברית – מאפיינים, או תכונות).



כל התכונות משפיעות על חיזוי של הפלט

$\hat{y} = W_1 * X_1 + W_2 * X_2 + W_3 * X_3 + b$

המטרה – למזער את הטעות עבור <u>כל</u> המקדמים של התכונות X

$$J = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$\hat{y} = W_1 * X_1 + W_2 * X_2 + \ldots + W_n * X_n + b$$

i – אינדקס שרץ על כל הדגימות

j – אינדקס שרץ על כל התכונות / הקלטים בכל דגימה

m - מספר הדגימות

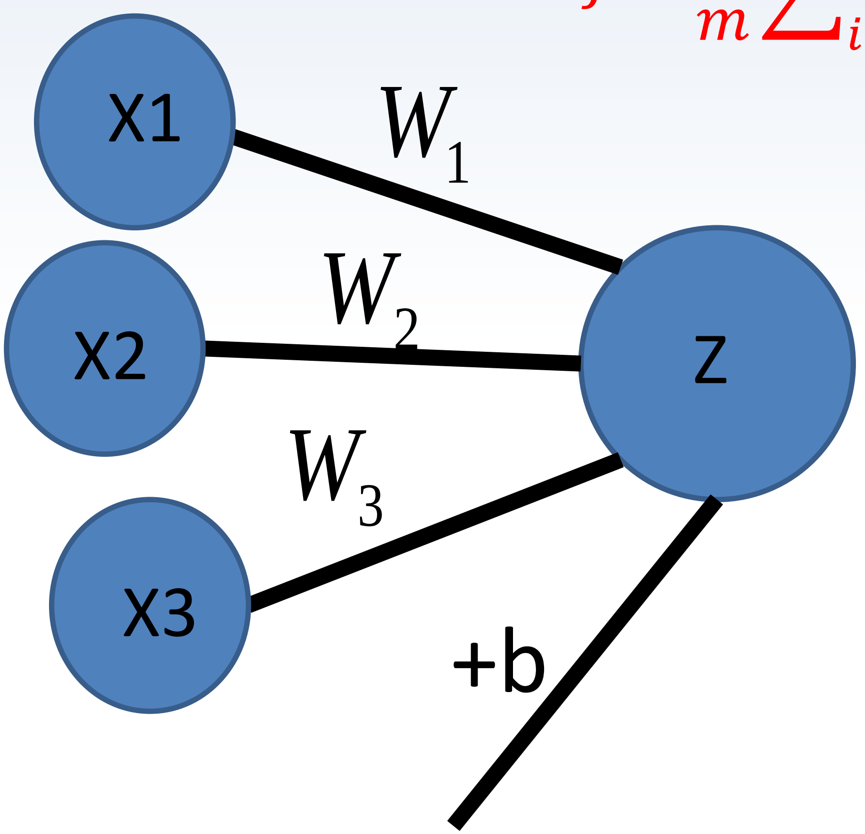
n - מספר התכונות

J - סכום כל הפרשי הריבועים בין הערך המצוי לערך הרצוי ונחלק במספר הדוגמאות לקבלת ה-MSE.

המטרה – למזער את הטעות עבור <u>כל</u> המקדמים של התכונות X – <u>רישום וקטורי</u>

$$J = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$\hat{y} = W_1 * X_1 + W_2 * X_2 + \dots + W_n * X_n + b$$



$$\underset{(m,\ )}{\hat{y}} = \underset{(m * n)}{X} * \underset{(n,)}{w} + b$$

J – סקלר

b - סקלר

w -מימדים (n,) הוא וקטור הפרמטרים של המודל

X - מימדים (m * n) - היא מטריצת התכונות

תרגול

נתונה הטבלה הבאה ובה מדדים של מדידות אולטרסאונד של עוברים:

משקל (gr)	אורך (cm) X2	היקף ראש (cm) X1
2000	20	10
2300	22	15
2200	21	17

המודל שלך צריך לחזות את משקל העובר על סמך שתי התכונות: היקף הראש ואורך כללי:

$y=w1*X1+w2*X2+b$

איך תראה מטריצה X?

A. X = [10 20 2000; 15 22 2300; 17 21 2200]

B. X = [10 15 17; 20 22 21]

C. X = [10 15 17; 20 22 21; 2000 2300 2000]

D. X = [10 20; 15 22; 17 21]

תרגול

נתונה הטבלה הבאה ובה מדדים של מדידות אולטרסאונד של עוברים:

משקל (gr)	אורך (cm) X2	היקף ראש (cm) X1
2000	20	10
2300	22	15
2200	21	17

המודל שלך צריך לחזות את משקל העובר על סמך שתי התכונות: היקף הראש ואורך כללי:

y=w1*X1+w2*X2+b

איך תראה מטריצה X?

A. X = [10 20 2000; 15 22 2300; 17 21 2200]

B. X = [10 15 17; 20 22 21]

C. X = [10 15 17; 20 22 21; 2000 2300 2000]

D. X = [10 20; 15 22; 17 21]

תרגול

נתונה הטבלה הבאה ובה מדדים של מדידות אולטרסאונד של עוברים:

משקל (gr)	אורך (cm) X2	היקף ראש (cm) X1
2000	20	10
2300	22	15
2200	21	17

המודל שלך צריך לחזות את משקל העובר על סמך שתי התכונות: היקף הראש ואורך כללי:

y=w1*X1+w2*X2+b

מטריצה X היא:

X = [10 20; 15 22; 17 21]

כמה איברים יש בוקטור w:

A. 3
B. 1
C. 2
D. 6

תרגול

נתונה הטבלה הבאה ובה מדדים של מדידות אולטרסאונד של עוברים:

היקף ראש (cm) X1	אורך (cm) X2	משקל (gr)
10	20	2000
15	22	2300
17	21	2200

המודל שלך צריך לחזות את משקל העובר על סמך שתי התכונות: היקף הראש ואורך כללי:

y=w1*X1+w2*X2+b

מטריצה X^T היא:

X^T = [10 15 17; 20 22 21]

כמה איברים יש בוקטור w:

A. 3

B. 1

C. 2

D. 6

שלבים למציאת המשקולות באמצעות gradient descent

- ✓ 1. איתחול W, b - ראשית אנו מאתחלים את פרמטרי המודל בערכים רנדומלים
- ✓ 2. חישוב הנגזרת החלקית **לכל אחת מהמשקולות - W**, b
- ✓ 3. עידכון b – הערך החדש של b שווה לערך הקודם של b פחות הנגזרת של b כפול גודל הצעד
- ✓ 4. עידכון **כל ערכי w** – הערך החדש של w שווה לערך הקודם של w פחות הנגזרת של w כפול גודל הצעד
- ✓ 5. בדיקת ביצועים – חישוב r^2

אנו חוזרים על שלבים 2-4 עד שפונקציית העלות תתכנס לערך המינימלי

חישוב הנגזרות ועדכון המשקולות

נעדכן כל משקולת לפי הנגזרת החלקית של פונקצית העלות לפי אותה משקולת:

w_0(new) = w_0(old)-learning rate*dE/dW_0

:

:

w_n(new) = w_n(old)-learning rate*dE/dW_n

b (new) = w (old)-learning rate*dE/db

חישוב הנגזרות ועדכון המשקולות

<u>רישום וקטורי</u>

נעדכן כל משקולת לפי הנגזרת החלקית של פונקצית העלות לפי אותה משקולת:

$$\underset{(n,)}{w} = \underset{(n,)}{w} - (lr / m) * \underset{n*m}{(X^T)} * \underset{(m,)}{sum(\hat{y} - y)}$$

$$b = b - (lr / m) * sum(\hat{y} - y)$$

$$\underset{(m,)}{\hat{y}} = \underset{m*n}{X} * \underset{(n,)}{w}$$

כללים במציאת המשקלות שיתאימו לטעות המינימלית – קצב הלמידה

איך נדע שקצב הלמידה שלנו גדול או קטן מדי?

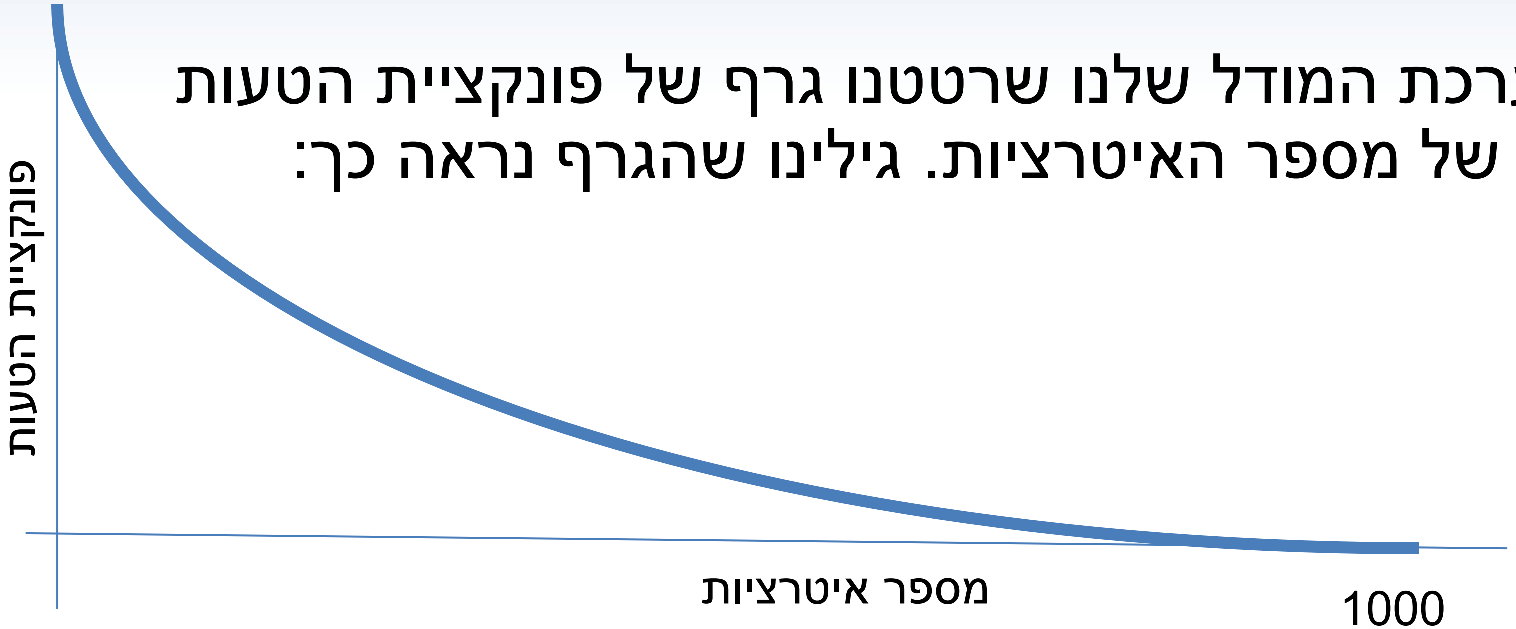
נצייר את ה-MSE לעומת מספר האיטרציות.

אם MSE עולה – קצב למידה גדול מדי.

אם MSE יורד – אפשר לנסות להגדיל את קצב הלמידה, בשביל לא להיות בקצב קטן מדי שלא מאפשר לנו להגיע לטעות המינימלית עבור מספר איטרציות נתון.

תרגול

לצורך הערכת המודל שלנו שרטטנו גרף של פונקציית הטעות כפונקציה של מספר האיטרציות. גילינו שהגרף נראה כך:

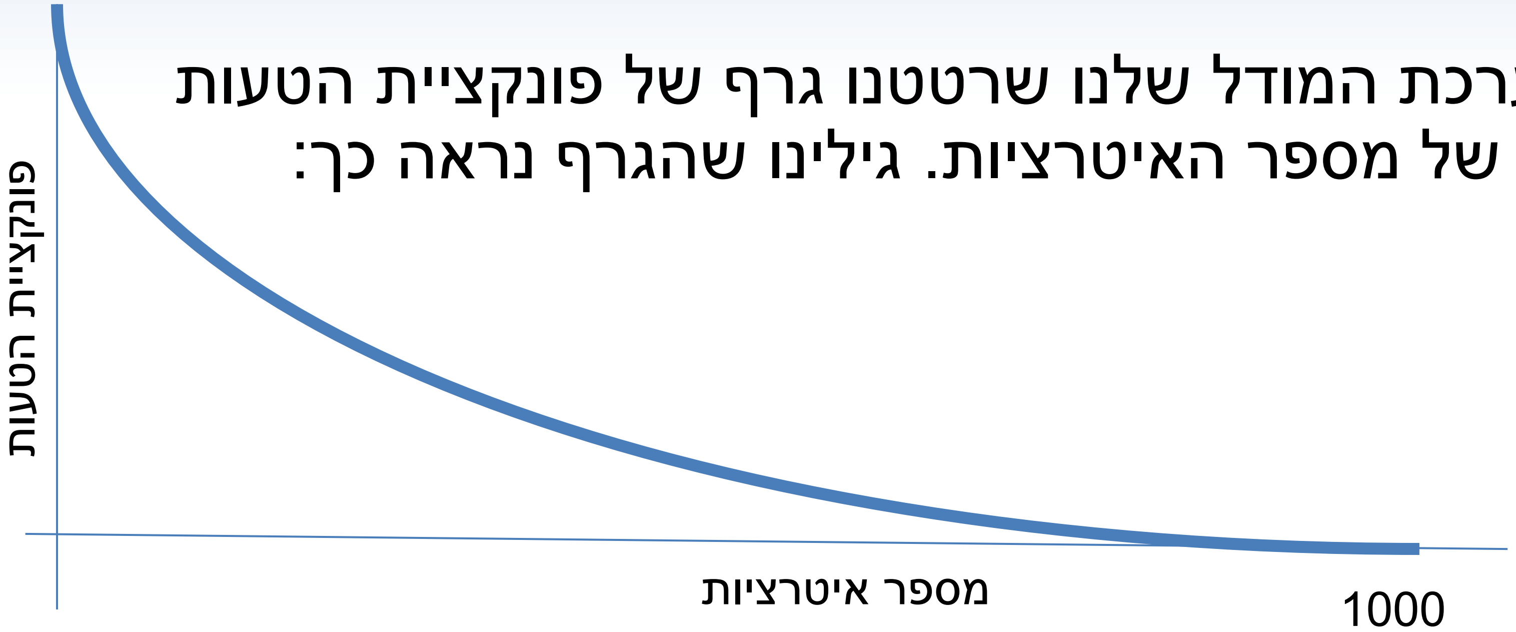


מה עלינו לעשות בכדי לקצר את מספר האיטרציות:

A. להגדיל את learning rate
B. להקטין את learning rate

תרגול

לצורך הערכת המודל שלנו שרטטנו גרף של פונקציית הטעות כפונקציה של מספר האיטרציות. גילינו שהגרף נראה כך:



מה עלינו לעשות בכדי לקצר את מספר האיטרציות:

A. $\boxed{\text{להגדיל את learning rate}}$
B. להקטין את learning rate

כללים במציאת המשקלות שיתאימו לטעות המינימלית – ערך הטעות

ה-gradient descent עוצר לאחר מספר איטרציות מסוים, או לאחר שהגענו לטעות שמקובלת עלינו – error goal.

הטעות error goal אינה מובנת מאליה במקרה כזה – יכול להיות ש**<u>לא ניתן</u>** להגיע לטעות של אפס, כי הנקודות לא יושבות כולן על קו ישר.

ולכן כדאי לנו להסתפק ב-MSE גבוה יותר, שיתן לנו עדיין פתרון טוב. מצד שני – לא נרצה לקבוע error goal גבוה מדי, אם אפשר להגיע לביצועים טובים יותר.

נורמליזציה של הנתונים

כללים במציאת המשקלות שיתאימו לטעות המינימלית – נורמליזציה

- gradient descent היא שיטה נומרית, אם עושים לתכונות "scaling" או "נורמליזציה" – מתאימים את הטווח של כל התכונות לערכים בעלי אותם סדרי גודל.
- נמצא שטווחים הקרובים לטווח של בין כ-1- ל-1 (לא בהכרח זהים לו, אבל קרובים) הם טובים להתכנסות מהירה של ה-gradient descent

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

where x is an original value, x' is the normalized value.

נניח שאתם רוצים להעריך את מחיר הבית באמצעות רגרסיה לינארית

תכונה 1 – Xi - גיל הבית. בדגימות שלכם גילאי הבתים נעים בין 30 ל 50, והממוצע הוא 38. מה יהיה החישוב לצורך נירמול התכונה של גיל הבית:

A. Xi
B. Xi / 50
C. (Xi-38) / (50-30)
D. (Xi-38) / (50-38)

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

תרגול

נניח שאתם רוצים להעריך את מחיר הבית באמצעות רגרסיה לינארית

תכונה 1 – Xi - גיל הבית. בדגימות שלכם גילאי הבתים נעים בין 30 ל 50, והממוצע הוא 38. מה יהיה החישוב לצורך נירמול התכונה של גיל הבית:

A. Xi
B. Xi / 50
C. $\boxed{\text{(Xi-38) / (50-30)}}$
D. (Xi-38) / (50-38)

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

שיטה נוספת לנירמול המאפיינים – Z-SCORE

z-score normalization

After z-score normalization, all features will have a mean of 0 and a standard deviation of 1.

To implement z-score normalization, adjust your input values as shown in this formula:

$$x_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j} \tag{4}$$

where j selects a feature or a column in the $\mathbf{X}$ matrix. μ_j is the mean of all the values for feature (j) and σ_j is the standard deviation of feature (j).

$$\mu_j = \frac{1}{m} \sum_{i=0}^{m-1} x_j^{(i)} \tag{5}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=0}^{m-1} (x_j^{(i)} - \mu_j)^2 \tag{6}$$

רגרסיה לינארית עם ספריית scikit-learn

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDRegressor
from sklearn.preprocessing import StandardScaler
```

#Scale/normalize the training data

```python
scaler = StandardScaler()
X_norm = scaler.fit_transform(X_train)
```

#Create and fit the regression model

```python
sgdr = SGDRegressor(max_iter=1000)
sgdr.fit(X_norm, y_train)
number_of_iterations_completed = sgdr.n_iter_
Number_of_weight_updates = sgdr.t_

b_norm = sgdr.intercept_
w_norm = sgdr.coef_
```

 # make a prediction using sgdr.predict()

```python
y_pred_sgd = sgdr.predict(X_norm)
```

 # make a prediction using w,b.

```python
y_pred = np.dot(X_norm, w_norm) + b_norm
```

#The R-squared (R^2) value

```python
print(sgdr.score(X_norm, y_train))
```