פרק 9: מימד, קשר ותבנית – יסודות ליניאריים לאינטליגנציה מלאכותית

Chapter 9: Dimension, Correlation, and Pattern – Linear Foundations for Artificial Intelligence

ד"ר יורם סגל

ספטמבר 2025

תוכן העניינים

1 מבוא: עולמו הנסתר של הווקטור

Introduction: The Hidden World of the Vector

בפרק זה נחקור את הבסיס הפילוסופי והמתמטי של הייצוג הווקטורי בבינה מלאכותית. נראה כיצד כל יחידת מידע – תמונה, מילה, או חולה במרפאה – הופכת לנקודה במרחב רב-ממדי, וכיצד מושג זה מהווה את היסוד של כל AI מודרני.

1.1 מבט ראשון: מה רואים כשמביטים בתמונה?

כשאתם מביטים בתמונה של חתול, מה המוח שלכם באמת רואה? האם הוא רואה יצור פרוותי חמוד עם שפם? או שמא הוא רואה משהו עמוק יותר – רשת מורכבת של 1000000 נקודות אור, כל אחת מהן נושאת מספר המייצג עוצמת בהירות?

זוהי השאלה המרכזית שמניעה את כל תחום הבינה המלאכותית המודרני: **כיצד ניתן להפוך מידע לווקטור, ואיך ווקטורים הופכים למשמעות?**

1.2 דקארט והמצאת השפה המספרית של המרחב

ב-1637, כשרנה דקארט (René Descartes) פרסם את ספרו "La Géométrie", הוא לא יכול היה לדמיין שהרעיון הפשוט שלו – לייצג נקודה במרחב באמצעות מספרים – יהפוך ליסוד של מהפכה טכנולוגית שתגיע 400 שנה מאוחר יותר [1].

דקארט המציא את מערכת הצירים הקרטזית, ועמה את הרעיון המהפכני שכל נקודה במרחב ניתנת לתיאור מספרי. אך רק במאה ה-20, כשמדעני המחשב החלו לחשוב על דרכים לייצג מידע, הבינו שהרעיון של דקארט הוא לא רק כלי גיאומטרי – **הוא שפה אוניברסלית לתיאור המציאות**.

שאלה למחשבה: אם דקארט יכול היה לייצג נקודה במרחב תלת-ממדי עם 3 מספרים, כמה מספרים נדרשים כדי לייצג תמונת חתול?

1.3 הווקטור: לא רק חץ במרחב

בקורסים מתמטיים מסורתיים, וקטור מוגדר לרוב כ"בחרמב ץח" – יש לו כיוון ואורך. אך בעולם מדעי הנתונים ולמידת המכונה, הווקטור הוא משהו עמוק יותר בהרבה: **הוא ייצוג של ישות במרחב מופשט** [1].

דוגמה מהחיים – חולה במרפאה:

אם נתון לכם חולה במרפאה, איך תתארו אותו כווקטור? התשובה: כל תכונה נמדדת של החולה – גיל, משקל, לחץ דם, רמת סוכר, טמפרטורה – הופכת ל-tnenopmoc אחד בווקטור. אם יש לנו 50 תכונות נמדדות, החולה מיוצג כנקודה במרחב בעל 50 ממדים:

$$\vec{p}_{\text{patient}} = \begin{bmatrix} \text{age} \\ \text{weight} \\ \text{blood\_pressure} \\ \text{glucose} \\ \vdots \\ \text{feature}_{50} \end{bmatrix} \in \mathbb{R}^{50}$$

הסימון $\mathbb{R}^{50}$ משמעותו "םידממ 50 לש ידילקוא בחרמ" – מרחב שבו כל נקודה מוגדרת על ידי 50 מספרים ממשיים. זהו המרחב שבו **כל החולים שלנו חיים**, במובן מתמטי.

1.4 מתמונת חתול לווקטור: תהליך הוקטוריזציה

כפי שהדגיש ד"ר יורם סגל בהרצאתו, תמונה של 1000×1000 פיקסלים בצבע RGB מכילה 3×10^6 ערכים מספריים (כל פיקסל מיוצג על ידי 3 ערכים: אדום, ירוק, כחול).

המתמטיקה מאחורי הוקטוריזציה:

כדי להפוך אותה לייצוג וקטורי הניתן לניתוח על ידי מודל למידת מכונה, אנו מבצעים תהליך של "Flattening" (השטחה). המטריצה המבנית מפורקת לרצף של מספרים, היוצר וקטור באורך של שלושה מיליון.

אם $\mathbf{I} \in \mathbb{R}^{h \times w \times c}$ היא תמונה בגובה h, רוחב w ו-c ערוצי צבע, אזי הוקטוריזציה מוגדרת כ:

$$\text{vec}(\mathbf{I}) = \mathbf{v} \in \mathbb{R}^{h \cdot w \cdot c} \quad (1)$$

כאשר $\mathbf{v} = [I_{1,1,1}, I_{1,1,2}, \ldots, I_{h,w,c}]^T$ – רצף של כל הפיקסלים בזה אחר זה.

משמעות עמוקה: תמונת החתול הפכה לנקודה אחת ויחידה במרחב בעל שלושה מיליון ממדים. כל תמונה שונה במעט תהיה נקודה אחרת באותו מרחב עצום.

1.5 מכפלה סקלרית: מדד הדמיון האולטימטיבי

אחת הפעולות החשובות ביותר על וקטורים היא **מכפלה סקלרית** (Dot Product, Inner Product). היא מוגדרת כך:

הגדרה 1.1 – מכפלה סקלרית (אלגברית):

$$\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n \quad (2)$$

שאלה מהותית שהעלה המרצה: מה משמעות המכפלה הסקלרית? למה היא כל כך חשובה?

התשובה מגיעה מנוסחה חלופית, גיאומטרית, שגילה המתמטיקאי הגרמני Hermann von Helmholtz במאה ה-19:

$$\langle\vec{u},\vec{v}\rangle = \|\vec{u}\| \cdot \|\vec{v}\| \cdot \cos(\theta) \tag{3}$$

כאשר θ היא הזווית בין שני הווקטורים, ו-$\|\vec{u}\|$ היא **הנורמה** (Norm) או אורך הווקטור:

$$\|\vec{u}\| = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2} = \sqrt{\sum_{i=1}^{n} u_i^2} \tag{4}$$

1.6 הוכחה: שתי ההגדרות זהות

משפט 1.1: ההגדרה האלגברית (משוואה 1.2) וההגדרה הגיאומטרית (משוואה 1.3) של מכפלה סקלרית שוות.

הוכחה:

צעד 1: משפט הקוסינוסים.

במשולש עם צלעות $\|\vec{u}\|$, $\|\vec{v}\|$ וזווית θ ביניהן, הצלע השלישית היא $\|\vec{u}-\vec{v}\|$. לפי משפט הקוסינוסים:

$$\|\vec{u}-\vec{v}\|^2 = \|\vec{u}\|^2 + \|\vec{v}\|^2 - 2\|\vec{u}\|\|\vec{v}\|\cos(\theta)$$

צעד 2: פיתוח אלגברי של האגף השמאלי.

נפתח את $\|\vec{u}-\vec{v}\|^2$ באמצעות ההגדרה הא לגברית:

$$\begin{aligned}
\|\vec{u}-\vec{v}\|^2 &= \sum_{i=1}^{n}(u_i - v_i)^2 \\
&= \sum_{i=1}^{n}(u_i^2 - 2u_iv_i + v_i^2) \\
&= \sum_{i=1}^{n} u_i^2 + \sum_{i=1}^{n} v_i^2 - 2\sum_{i=1}^{n} u_iv_i \\
&= \|\vec{u}\|^2 + \|\vec{v}\|^2 - 2\langle\vec{u},\vec{v}\rangle \qquad (5)
\end{aligned}$$

צעד 3: השוואת הביטויים.

מצעד 1: $\|\vec{u}-\vec{v}\|^2 = \|\vec{u}\|^2 + \|\vec{v}\|^2 - 2\|\vec{u}\|\|\vec{v}\|\cos(\theta)$

מצעד 2: $\|\vec{u}-\vec{v}\|^2 = \|\vec{u}\|^2 + \|\vec{v}\|^2 - 2\langle\vec{u},\vec{v}\rangle$

ביטול $\|\vec{u}\|^2 + \|\vec{v}\|^2$ משני האגפים מוביל ל:

$$2\langle\vec{u},\vec{v}\rangle = 2\|\vec{u}\| \cdot \|\vec{v}\| \cdot \cos(\theta)$$

חלוקה ב-2 נותנת:

$$\blacksquare \quad \langle \vec{u}, \vec{v} \rangle = \|\vec{u}\| \cdot \|\vec{v}\| \cdot \cos(\theta)$$

משמעות עמוקה: מכפלה סקלרית היא **מדד דמיון גיאומטרי**.

- אם $\theta = 0°$ (וקטורים באותו כיוון): $\cos(\theta) = 1$ והמכפלה מקסימלית ⎕ **דמיון מלא**
- אם $\theta = 90°$ (וקטורים אורתוגונליים): $\cos(\theta) = 0$ והמכפלה מתאפסת ⎕ **אין דמיון כלל**
- אם $\theta = 180°$ (וקטורים מנוגדים): $\cos(\theta) = -1$ והמכפלה מינימלית ⎕ **ניגוד מלא**

1.7 קשר ל-IA: Cosine Similarity ומנועי חיפוש

מנועי חיפוש מודרניים כמו Google ומערכות המלצה כמו Netflix משתמשים בווריאציה על מכפלה סקלרית שנקראת **Cosine Similarity** – דמיון קוסינוס:

$$\text{similarity}(\vec{u}, \vec{v}) = \frac{\langle \vec{u}, \vec{v} \rangle}{\|\vec{u}\| \cdot \|\vec{v}\|} = \cos(\theta) \tag{6}$$

חישוב זה **מנרמל** את המכפלה הסקלרית כך שהתוצאה תמיד בין -1 ל-$+1$, ללא תלות באורכי הווקטורים.

למה נורמליזציה חשובה?

בלי נורמליזציה, וקטור ארוך יקבל מכפלה סקלרית גדולה יותר רק בגלל האורך, לא בגלל הדמיון. Cosine Similarity פותר זאת על ידי מדידת הזווית בלבד.

דוגמה מהחיים – מנוע חיפוש:

כשאתם מקלידים שאילתה "machine learning tutorial", מנוע החיפוש:

1. **ממיר את השאילתה לווקטור** $\vec{q}$: כל מילה מקבלת משקל (למשל, באמצעות TF-IDF או Word2Vec [2])
2. **משווה את** $\vec{q}$ **לכל מסמך** $\vec{d_i}$ במאגר באמצעות Cosine Similarity
3. **מחזיר את המסמכים** עם הדמיון הגבוה ביותר (ציון קוסינוס הכי קרוב ל-1)

פונקציות NumPy למכפלה סקלרית ונורמה:

טבלה 1: פונקציות NumPy למכפלה סקלרית, נורמה ו-Cosine Similarity

Function	תפקיד	שימוש והסבר
np.dot(u, v)	מכפלה סקלרית	מחשבת $\sum u_i v_i$ – הליבה של כל חישוב דמיון
np.linalg.norm(u)	נורמה (L2)	מחשבת $\sqrt{\sum u_i^2}$ – אורך הווקטור
@ (אופרטור)	מכפלת מטריצות	תחביר קצר: u @ v זהה ל-np.dot(u, v)

הערה חשובה: פונקציות אלה הן הבסיס של כל אלגוריתם AI. נשתמש בהן שוב ושוב לאורך הספר.

פסאודו-קוד – Cosine Similarity:

Cosine Similarity – מימוש ממוקד

```
import numpy as np

def cosine_similarity(u, v):
    """
␣␣␣␣מחשבת␣Cosine␣Similarity␣בין␣שני␣וקטורים.

␣␣␣␣Args:
␣␣␣␣␣␣␣␣u,␣v:␣וקטורי␣NumPy␣בעלי␣אותו␣אורך

␣␣␣␣Returns:
␣␣␣␣␣␣␣␣float:␣ציון␣דמיון␣בין␣-1␣ל-1
␣␣␣␣"""
    # מכפלהסקלרית  - מונה
    dot_product = np.dot(u, v)

    # נורמות - מכנה
    norm_u = np.linalg.norm(u)
    norm_v = np.linalg.norm(v)

    # Cosine Similarity
    return dot_product / (norm_u * norm_v)

# דוגמה: שנימסמכים וקטוריים  ( TF-IDF מדומים)
doc1 = np.array([1, 2, 3, 0, 0])
doc2 = np.array([1, 1, 0, 4, 5])

print(f"Similarity:␣{cosine_similarity(doc1,␣doc2):.3f}")
# תוצאה: 0.277 - דמיון חלש
```

1.8 Word Embeddings: המהפכה של Word2Vec

אחת ההצלחות המרשימות ביותר של ייצוג וקטורי היא **Word2Vec**, שפותחה על ידי Tomas Mikolov ועמיתיו ב-elgooG ב-2013 [2].

הרעיון המהפכני: לייצג כל מילה בשפה כווקטור בן ~ 300 ממדים, כך **שמילים דומות במשמעות יהיו קרובות במרחב הווקטורי**.

הפלא המתמטי של Word2Vec:

אחד הממצאים המפורסמים ביותר הוא שחישובים אריתמטיים על וקטורי מילים מניבים תוצאות משמעותיות סמנטית:

$$\vec{king} - \vec{man} + \vec{woman} \approx \vec{queen}$$

שאלה שמעלה פלא: איך זה אפשרי? איך חיבור וחיסור של מספרים יכול לבטא יחסים סמנטיים?

התשובה: הווקטורים לומדים **יחסים**. הווקטור $\vec{king} - \vec{man}$ מייצג את המושג המופשט ”תירכז תוכלמ”, וכשמוסיפים $\vec{woman}$ מקבלים ”תישנ תוכלמ” = $\vec{queen}$.

זה עובד בגלל **הנחת הדיסטריבוציה** (Distributional Hypothesis) שניסח הבלשן Zelling Harris ב-1954 [3]:

> *”Words that occur in similar contexts tend to have similar meanings”*
>
> ”מילים שמופיעות בהקשרים דומים נוטות לשאת משמעות דומה”

אלגוריתם Word2Vec לומד וקטורים שמנבאים את הקונטקסט של מילה, ובכך **מקודדים משמעות סמנטית כמרחק גיאומטרי**.

1.9 תרגיל תכנות עצמי 1.1 – חיפוש מילים דומות

מטרה: להבין איך Cosine Similarity משמש למציאת מילים דומות במודל Word2Vec.

משימה:

1. טענו וקטורי Word2Vec מוכנים (למשל, מ-misneG או מודל GloVe)
2. בחרו מילה (למשל, ”computer”)
3. חשבו Cosine Similarity בינה לבין כל המילים האחרות במאגר
4. הציגו את 10 המילים הדומות ביותר

פסאודו-קוד:

תוצאה צפויה:

- laptop: 0.82
- software: 0.78
- technology: 0.75
- hardware: 0.72
- PC: 0.70

חיפוש מילים דומות ב-ceV2droW

```python
from gensim.models import KeyedVectors

# טנרטניאהמ ( דירוהלןכומ Word2Vec לדומתניעט
model = KeyedVectors.load_word2vec_format(
    'GoogleNews-vectors-negative300.bin',
    binary=True
)

# תומודםילימתאיצמ   - בתשמתשמהיצקנופה  -Cosine Similarity
similar_words = model.most_similar('computer', topn=10)

# הספדה
print("המילים␣הדומות␣ביותר␣ל-'computer':")
for word, similarity in similar_words:
    print(f"␣␣{word}:␣{similarity:.3f}")
```

1.10 אזהרה: הסכנה בווקטורים – Bias ודעות קדומות

ב-2016, חוקרים מאוניברסיטת Princeton גילו תופעה מדאיגה: מודלי Word2Vec שאומנו על טקסטים מהאינטרנט הטמיעו **דעות קדומות חברתיות** [4].

דוגמאות להטיות שנמצאו:

- $\vec{\text{man}}$: $\vec{\text{programmer}}$ כמו $\vec{\text{woman}}$: $\vec{\text{homemaker}}$
- $\vec{\text{man}}$: $\vec{\text{doctor}}$ כמו $\vec{\text{woman}}$: $\vec{\text{nurse}}$
- קשרים חזקים יותר בין שמות אירופאים למילים חיוביות לעומת שמות אפרו-אמריקאים

המודל למד שגברים קשורים למקצועות טכנולוגיים, ונשים למקצועות ביתיים – לא משום שזו אמת, אלא כי **כך התייחסו הטקסטים שממנו למד**.

שאלה מוסרית שהעלה ד"ר סגל: אם AI לומד מהעולם, והעולם מוטה – האם AI מחזק את המוטיות?

התשובה המחקרית: כן, אלא אם כן מתערבים באופן פעיל. זו אחת המשימות המרכזיות של תחום **Fairness in AI**. חוקרים כמו Tolga Bolukbasi ועמיתיו פיתחו שיטות ל**ניטרול הטיות** (Debiasing) בווקטורי מילים [5].

שיטת הניטרול:

השיטה מבוססת על זיהוי **כיווני הטיה** במרחב הווקטורי. למשל, הכיוון $\vec{d}_{\text{gender}} = \vec{\text{he}} - \vec{\text{she}}$ מייצג את ממד המגדר. לאחר מכן, מסירים את המרכיב הזה ממילים שאמורות להיות ניטרליות:

$$\vec{\text{doctor}}_{\text{debiased}} = \vec{\text{doctor}} - \text{proj}_{\vec{d}_{\text{gender}}}(\vec{\text{doctor}})$$

כאשר $\text{proj}_{\vec{d}}(\vec{v})$ היא ההטלה של $\vec{v}$ על הכיוון $\vec{d}$.

מסקנה: ייצוג וקטורי הוא כלי עוצמתי, אך הוא משקף את הנתונים שממנו למד. **אחריות החוקר היא להבטיח שהמודל לא מנציח אפליה.**

1.11 מרחבים וקטוריים: הפורמליזם המתמטי

עד כה דיברנו על וקטורים בצורה אינטואיטיבית. אך מה באמת הופך אוסף של מספרים לווקטור? התשובה מגיעה מ**תורת המרחבים הווקטוריים** (Vector Space Theory), שפותחה במאה ה-19 על ידי מתמטיקאים כמו Hermann Grassmann (1844) ו-onaeP eppesuiG (1888).

הגדרה 1.2 – מרחב וקטורי:

מרחב וקטורי V מעל שדה $\mathbb{F}$ (בדרך כלל $\mathbb{R}$ או $\mathbb{C}$) הוא קבוצה עם שתי פעולות:

1. **חיבור וקטורים:** $\vec{u} + \vec{v} \in V$ לכל $\vec{u}, \vec{v} \in V$
2. **כפל בסקלר:** $\alpha\vec{u} \in V$ לכל $\alpha \in \mathbb{F}$ ו-$\vec{u} \in V$

שמקיימות 8 אקסיומות:

טבלה 2: אקסיומות מרחב וקטורי

אקסיומה	תכונה
1	קומוטטיביות: $\vec{u} + \vec{v} = \vec{v} + \vec{u}$
2	אסוציאטיביות: $(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$
3	קיום איבר אפס: קיים $\vec{0}$ כך ש-$\vec{u} + \vec{0} = \vec{u}$
4	קיום איבר נגדי: לכל $\vec{u}$ קיים $-\vec{u}$ כך ש-$\vec{u} + (-\vec{u}) = \vec{0}$
5	כפל באחד: $1 \cdot \vec{u} = \vec{u}$
6	דיסטריבוטיביות: $\alpha(\vec{u} + \vec{v}) = \alpha\vec{u} + \alpha\vec{v}$
7	דיסטריבוטיביות: $(\alpha + \beta)\vec{u} = \alpha\vec{u} + \beta\vec{u}$
8	אסוציאטיביות כפל: $(\alpha\beta)\vec{u} = \alpha(\beta\vec{u})$

למה זה חשוב ל-AI?

ברגע שהוכחנו שאוסף של ישויות (תמונות, מסמכים, חולים) יוצר מרחב וקטורי, אנחנו יכולים להשתמש ב**כל הכלים של אלגברה ליניארית**:

- פירוק SVD (Singular Value Decomposition)
- ערכים עצמיים (Eigenvalues) ווקטורים עצמיים (Eigenvectors)
- הטלות (Projections) והטרנספורמציות ליניאריות

- PCA (Principal Component Analysis) להפחתת ממדיות

כל אלו הם כלים שנשתמש בהם לאורך הספר לניתוח ועיבוד נתונים.

1.12 המעבר למימד גבוה: ברכה או קללה?

ב-1961, הסטטיסטיקאי והמתמטיקאי Richard Bellman טבע את המונח המפורסם **"Curse of Dimensionality"** – קללת המימדיות [6].

הבעיה המתמטית:

Bellman הבין שככל שמוסיפים תכונות (ממדים) לנתונים, **נפח המרחב גדל באופן אקספוננציאלי**, והנתונים הופכים ספרסיים (Sparse) יותר ויותר.

דוגמה מספרית שהדגים המרצה:

נניח שאנחנו רוצים לדגום 10 נקודות לאורך כל ממד כדי לכסות את המרחב בצפיפות סבירה:

- ב-1 ממד: $10^1 = 10$ נקודות
- ב-2 ממדים: $10^2 = 100$ נקודות
- ב-3 ממדים: $10^3 = 1000$ נקודות
- ב-10 ממדים: $10^{10} = 10000000000$ נקודות (עשרה מיליארד!)
- ב-100 ממדים: 10^{100} – מספר גדול יותר מכמות האטומים ביקום!

שאלת המפתח שהעלה ד"ר סגל:

אם יש לנו תמונת חתול 1000×1000 פיקסלים (מיליון תכונות), כמה דוגמאות אימון נדרשות כדי לאמן מודל Brute Force?

התשובה המתמטית:

לפי כלל האצבע שנידון בהרצאה:

- **למידת מכונה קלאסית:** מינימום 30 דוגמאות לכל תכונה
- **למידה עמוקה:** מינימום 100 דוגמאות לכל תכונה

עבור מיליון תכונות: $100 \times 10^6 = 100000000$ דוגמאות (מאה מיליון!)

הפתרון – רשתות CNN:

זו הסיבה שרשתות Convolutional Neural Networks (CNN) חיוניות [7], [8]. הן מפחיתות באופן דרמטי את מספר הפרמטרים באמצעות:

1. **שיתוף משקלים** (Weight Sharing): אותו פילטר משמש בכל התמונה
2. **קישוריות מקומית** (Local Connectivity): כל נוירון מחובר רק לאזור קטן
3. **היררכיה של תכונות:** למידה הדרגתית מתכונות פשוטות (קצוות) למורכבות (פנים)

הישג מפורסם: רשת AlexNet של Krizhevsky, Sutskever ו-notniH (2012) אימנה על 1.2 מיליון תמונות עם $60 \sim$ מיליון פרמטרים, והשיגה פריצת דרך בסיווג תמונות [8].

זו תהיה נקודת המוצא לפרק הבא, שבו נצלול לעומקה של קללת המימדיות ונראה כיצד היא משפיעה על כל אלגוריתם למידה.

1.13 תרגיל תכנות עצמי 1.2 – השוואת מרחקים בממדים שונים

מטרה: להמחיש את קללת המימדיות באופן מעשי.

משימה:

1. צרו 100 נקודות אקראיות במרחבים בממדים שונים: 2, 10, 100, 1000
2. חשבו את המרחק האוקלידי בין כל זוג נקודות
3. חשבו את ממוצע המרחקים ואת סטיית התקן
4. הציגו גרף: ציר X = מימד, ציר Y = יחס סטיית תקן/ממוצע

פסאודו-קוד:

תוצאה צפויה: היחס שואף ל-0 ככל שהממד גדל – כל הנקודות נעשות "באותו מרחק" זו מזו, ומדדי מרחק מאבדים משמעות.

1.14 סיכום ומבט קדימה

מה למדנו בפרק זה?

1. **וקטור הוא ייצוג מופשט** – לא רק חץ, אלא כל ישות הניתנת לתיאור מספרי (תמונה, מילה, חולה)
2. **מכפלה סקלרית היא מדד דמיון גיאומטרי** – הבסיס לחיפוש, המלצות, ו-PLN
3. **Cosine Similarity הוא הכלי המרכזי** – מנרמל מרחקים ומודד זווית בלבד
4. **Word2Vec הוא דוגמה מבריקה** – מילים הופכות לווקטורים והסמנטיקה הופכת לגיאומטריה
5. **הטיות ב-IA הן בעיה אמיתית** – מודלים לומדים מהעולם, כולל דעות קדומות
6. **מרחבים וקטוריים הם הפורמליזם** – מבטיחים שנוכל להשתמש באלגברה ליניארית
7. **ממד גבוה = אתגר עצום** – קללת המימדיות דורשת ארכיטקטורות חכמות כמו CNN

מבט קדימה – פרק 2:

בפרק הבא נצלול לעומק **הדיכוטומיה של המידע** – מדוע יותר תכונות לא תמיד טוב יותר, ואיך להתמודד עם מרחבים בעלי ממד גבוה. נראה:

```
import numpy as np
import matplotlib.pyplot as plt

def curse_of_dimensionality_demo(dims, n_points=100):
    """
␣␣␣␣ממחישה␣את␣קללת␣המימדיות␣על␣ידי␣חישוב␣מרחקים.
␣␣␣␣"""
    results = []

    for d in dims:
        # תויארקאתודוקנתריצי
        points = np.random.rand(n_points, d)

        # םיקחרמהלכבושיח
        distances = []
        for i in range(n_points):
            for j in range(i+1, n_points):
                dist = np.linalg.norm(points[i] - points[j])
                distances.append(dist)

        # תוקיטסיטטס
        mean_dist = np.mean(distances)
        std_dist = np.std(distances)
        ratio = std_dist / mean_dist  # כשהוזשואפל  -0, הקללהזהק

        results.append(ratio)
        print(f"Dim={d}:␣Mean={mean_dist:.3f},␣Std={std_dist:.3f},␣Ratio
={ratio:.3f}")

    return results

# הצרה
dimensions = [2, 5, 10, 50, 100, 500, 1000]
ratios = curse_of_dimensionality_demo(dimensions)

# ףרג
plt.plot(dimensions, ratios, marker='o')
plt.xlabel('Number␣of␣Dimensions')
plt.ylabel('Std/Mean␣Ratio')
plt.title('Curse␣of␣Dimensionality')
plt.grid(True)
plt.show()
```

- הוכחה מתמטית מלאה של קללת המימדיות
- השפעה על אלגוריתמים: K-Nearest Neighbors, SVM, Decision Trees
- יחס פיצ'ר-דגימה: כמה נתונים באמת צריך?
- פתרונות: Feature Selection, PCA, Regularization

שאלת מחשבה לסיום:

אם תמונת חתול 1000×1000 היא נקודה במרחב מיליון-ממדי, וכמעט כל "הזזה" קטנה היא אקסטרפולציה – איך מודלים גנרטיביים כמו DALL-E מצליחים ליצור תמונות חדשות שנראות הגיוניות?

התשובה תחכה לפרק 5 על קונוולוציה ורשתות CNN.

מטלות וקריאה מורחבת

תרגיל 1.1: מצאו מילים דומות באמצעות Word2Vec (ראו פסאודו-קוד).

תרגיל 1.2: המחישו את קללת המימדיות (ראו פסאודו-קוד).

תרגיל 1.3: חשבו ידנית מכפלה סקלרית, נורמה, ו-ytiralimiS enisoC בין:

$$\vec{u} = [1, 2, 3], \quad \vec{v} = [4, 5, 6]$$

פתרון: $\langle \vec{u}, \vec{v} \rangle = 32$, $\|\vec{u}\| = \sqrt{14}$, $\|\vec{v}\| = \sqrt{77}$, דמיון $= 0.975 =$

קריאה מורחבת:

- [1] – Linear Algebra and Learning from Data, פרקים 1-2: יסודות אלגברה ליניארית
- [2] – המאמר המקורי על Word2Vec: "Efficient Estimation of Word Representations in Vector Space"
- [4] – מחקר על הטיות במודלי שפה: "Semantics Derived Automatically from Language Corpora Contain Human-like Biases"
- [9] – Deep Learning, פרק 5: למידת מכונה בסיסית

שאלות להעמקה:

1. מדוע Cosine Similarity עדיף על מרחק אוקלידי במרחבים רב-ממדיים?
2. האם ניתן להסיר לחלוטין הטיות ממודלי Word2Vec? מה המחיר?
3. איך רשתות CNN "מרמות" את קללת המימדיות?

סיום פרק 1

2 הדיכוטומיה של המידע: קללת המימדיות

The Information Dichotomy: Curse of Dimensionality

בפרק זה נחקור את אחד האתגרים המרכזיים של למידת מכונה: הפרדוקס שבו הוספת מידע (תכונות) עלולה להחליש את המודל במקום לחזק אותו. נבחן את היסודות המתמטיים של קללת המימדיות, נוכיח את השפעותיה, ונראה כיצד היא משפיעה על אלגוריתמים שונים.

2.1 הפרדוקס: מתי "יותר" זה "פחות"?

דמיינו רופא שמנסה לאבחן מחלה. בתחילה, יש לו שלוש תכונות: טמפרטורה, דופק, ולחץ דם. המודל שלו עובד היטב. לפתע, הוא מקבל גישה למעבדה מתקדמת שמודדת 1000 תכונות ביוכימיות. אינטואיטיבית, המידע הנוסף אמור לשפר את האבחנה. אך במציאות, דיוק המודל יורד.

למה זה קורה?

התשובה טמונה בתופעה מתמטית מפתיעה שגילה Richard Bellman ב-1957 [6]: ככל שמספר הממדים גדל, המרחב "מתנפח" באופן אקספוננציאלי, והנתונים הופכים דלילים (Sparse) – כמו כוכבים ביקום המתרחב.

2.2 בלמן והולדת המושג

Richard Ernest Bellman (1920–1984) היה מתמטיקאי אמריקאי שתרם תרומות מהפכניות לתחומי תכנות דינמי, תורת הבקרה, ולמידת מכונה. בספרו "Dynamic Programming" (1957), הוא זיהה בעיה מהותית: **מורכבות חישובית גדלה באופן אקספוננציאלי עם מספר המשתנים**.

התובנה המרכזית של בלמן:

במרחב חד-ממדי, אם רוצים לכסות קטע באורך 1 ברשת של נקודות במרווח ϵ, נדרשות $1/\epsilon \sim$ נקודות. אך במרחב d-ממדי, נדרשות $(1/\epsilon)^d \sim$ נקודות – גידול אקספוננציאלי [6].

דוגמה מספרית:

טבלה 3: מספר הנקודות הנדרש לכיסוי מרחב [0,1] בצפיפות $\epsilon = 0.1$

ממד	נקודות נדרשות	סדר גודל
1	$10^1 = 10$	עשרות
2	$10^2 = 100$	מאות
3	$10^3 = 1000$	אלפים
10	10^{10}	עשרה מיליארד
100	10^{100}	גוגול (יותר מאטומי היקום)
1000000	$10^{1000000}$	בלתי נתפס

מסקנה: תמונות חתול 1000×1000 פיקסלים (מיליון ממדים) דורשת מספר דגימות שהוא מעבר ליכולת חישובית של כל מחשבי העולם ביחד.

2.3 הוכחה מתמטית: התרחקות מהמרכז

נוכיח תופעה מפתיעה: במרחבים רב-ממדיים, **כמעט כל הנפח מרוכז בקליפה החיצונית**, רחוק מהמרכז.

משפט 2.1 – ריכוז הנפח בקליפה:

יהי כדור יחידה $B_d = \{\vec{x} \in \mathbb{R}^d : \|\vec{x}\| \leq 1\}$ במימד d. נפח הכדור הוא:

$$V_d(r) = \frac{\pi^{d/2}}{\Gamma(d/2+1)} r^d \tag{7}$$

כאשר Γ היא פונקציית הגמא, ו-r הוא הרדיוס.

הוכחה – יחס הנפחים:

נחשב את יחס הנפח של הכדור הפנימי $B_d(0.9)$ (רדיוס 0.9) לכדור המלא $B_d(1)$:

$$\begin{aligned} \frac{V_d(0.9)}{V_d(1)} &= \frac{(0.9)^d \cdot \pi^{d/2}/\Gamma(d/2+1)}{1^d \cdot \pi^{d/2}/\Gamma(d/2+1)} \\ &= (0.9)^d \end{aligned} \tag{8}$$

נחשב עבור ממדים שונים:

טבלה 4: יחס הנפח הפנימי (90% מהרדיוס) לנפח המלא

ממד d	$(0.9)^d$	**אחוז מהנפח**
2	0.81	81%
3	0.729	73%
10	0.349	35%
100	2.66×10^{-5}	0.003%
1000	$\sim 10^{-46}$	כמעט אפס

מסקנה מדהימה: ב-100 ממדים, רק 0.003% מהנפח נמצא ב-90% המרכזיים! כמעט כל הנפח מרוכז בקליפה הדקה בין $r = 0.9$ ל-$r = 1$.

משמעות ל-IA:

אם דגימות האימון שלנו הן "נקודות במרכז", המודל שלנו יצטרך לעשות אקסטרפולציה עצומה כדי לחזות במרחב האמיתי (הקליפה החיצונית) שבו נמצאות רוב ה"תמונות האפשריות".

2.4 הוכחה: התכנסות המרחקים

תופעה נוספת של קללת המימדיות: במרחבים רב-ממדיים, **כל המרחקים בין נקודות הופכים להיות דומים**.

משפט 2.2 – התכנסות יחס המרחקים:

יהיו $\{\vec{x}_i\}_{i=1}^n$ נקודות בלתי-תלויות במרחב $\mathbb{R}^d$. יהיו $d_{\max}$ ו-$d_{\min}$ המרחקים המקסימלי והמינימלי מנקודת מבחן $\vec{q}$. אזי:

$$\lim_{d\to\infty} \frac{d_{\max} - d_{\min}}{d_{\min}} \to 0 \tag{9}$$

הוכחה (סקיצה):

צעד 1: מרחק אוקלידי במימד d בין $\vec{q}$ ל-$\vec{x}_i$:

$$d_i = \|\vec{q} - \vec{x}_i\| = \sqrt{\sum_{j=1}^{d}(q_j - x_{ij})^2}$$

צעד 2: אם המרכיבים $(q_j - x_{ij})$ הם משתנים אקראיים בלתי-תלויים עם תוחלת μ ושונות σ^2, אזי לפי חוק המספרים הגדולים:

$$\frac{1}{d}\sum_{j=1}^{d}(q_j - x_{ij})^2 \xrightarrow{P} \mu^2 + \sigma^2$$

צעד 3: לכן:

$$d_i = \sqrt{d\cdot(\mu^2+\sigma^2+o(1))} = \sqrt{d}\cdot\sqrt{\mu^2+\sigma^2}\cdot(1+o(1))$$

צעד 4: כל המרחקים d_i גדלים כ-$\sqrt{d}$, אך ההבדלים ביניהם גדלים לאט יותר (רק כ-$\sqrt{d\cdot \text{Var}}$). לכן:

$$\frac{d_{\max} - d_{\min}}{d_{\min}} \sim \frac{O(\sqrt{d})}{O(\sqrt{d})}\cdot\frac{\text{Var}}{\text{Mean}^2} \to 0 \quad \text{כאשר } d\to\infty$$

■

משמעות מעשית:

במרחב בעל 1000 ממדים, אם הנקודה הקרובה ביותר נמצאת במרחק 100, הנקודה הרחוקה ביותר נמצאת במרחק ~ 100.01 – **כמעט אותו דבר**. מדדי מרחק מאבדים יכולת הבחנה [10], [11].

2.5 השפעה על אלגוריתמים: K-Nearest Neighbors

אלגוריתם K-Nearest Neighbors (KNN), שפותח על ידי Evelyn Fix ו-segdoH hpesoJ ב-1951 [12], הוא אחד האלגוריתמים הפשוטים והאינטואיטיביים ביותר בלמידת מכונה.

הרעיון: כדי לסווג נקודה חדשה $\vec{q}$, מצא את k הנקודות הקרובות ביותר במערך האימון, והצבע לפי רוב.

למה KNN קורס במימד גבוה?

מכיוון שכל המרחקים הופכים דומים (משפט 2.2), **אין משמעות ל"קרוב ביותר"**. הנקודה ה-1 הקרובה ביותר והנקודה ה-1000 הקרובה ביותר נמצאות כמעט באותו מרחק!

ניסוי מספרי – סימולציה:

תוצאה צפויה: הדיוק יורד מ-95% (במימד 2) ל-55% (במימד 1000) – כמעט אקראי.

2.6 יחס פיצ'ר-דגימה: כמה נתונים צריך?

כפי שהדגיש ד"ר יורם סגל בהרצאתו, קיימים כללי אצבע לקביעת מספר הדגימות הנדרש:

כלל 1 – למידת מכונה קלאסית:

$$n_{\text{samples}} \geq 30 \times n_{\text{features}} \tag{10}$$

כלל 2 – למידה עמוקה:

$$n_{\text{samples}} \geq 100 \times n_{\text{features}} \tag{11}$$

הצדקה מתמטית:

כללים אלו נובעים מתורת הלמידה הסטטיסטית. לפי **גבול VC (Vapnik-Chervonenkis)** [13], שגיאת ההכללה של מודל עם VC-dimension d מוגבלת ע"י:

$$\epsilon \leq \sqrt{\frac{d \log(n/d) + \log(1/\delta)}{n}} \tag{12}$$

כאשר n הוא מספר הדגימות, δ רמת הביטחון, ו-ϵ שגיאת ההכללה.

פירוש: כדי לקבל שגיאה קטנה ϵ, נדרש $n \sim O(d/\epsilon^2)$ – **מספר הדגימות צריך לגדול ליניארית עם המימד.**

דוגמה מהחיים – תמונת חתול:

תמונה 1000×1000 פיקסלים BGR:

- מספר תכונות: $1000 \times 1000 \times 3 = 3000000$

- דגימות נדרשות (כלל 2): $100 \times 3000000 = 300000000$ (שלוש מאות מיליון תמונות!)

למה זה בלתי אפשרי?

- ImageNet, אחד ממאגרי התמונות הגדולים בעולם, מכיל רק ~ 1.2 מיליון תמונות אימון [14]

הדגמת קריסת KNN במימד גבוה

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

def knn_curse_demo(dims, n_samples=1000):
    """
␣␣␣␣בוחן␣ביצועי␣KNN␣כפונקציה␣של␣מספר␣הממדים.
␣␣␣␣"""
    results = []

    for d in dims:
        # יצירתנתוניםסינתטיים
        X, y = make_classification(
            n_samples=n_samples,
            n_features=d,
            n_informative=min(d, 10),  # רק 10 תכונותאינפורמטיביות
            n_redundant=0,
            random_state=42
        )

        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.3, random_state=42
        )

        # אימון KNN
        knn = KNeighborsClassifier(n_neighbors=5)
        knn.fit(X_train, y_train)

        # דיוק
        accuracy = knn.score(X_test, y_test)
        results.append(accuracy)

        print(f"Dim={d}:␣Accuracy={accuracy:.3f}")

    return results

# הרצה
dimensions = [2, 5, 10, 50, 100, 500, 1000]
accuracies = knn_curse_demo(dimensions)
```

- גם Google לא צילמה 300 מיליון תמונות חתולים

אז איך Deep Learning מצליח?
התשובה: **שיתוף משקלים והנחות אינדוקטיביות** [7], [9].

2.7 הפתרון: רשתות CNN ומבנה מרחבי

רשתות Convolutional Neural Networks (CNN), שפותחו על ידי Yann LeCun ב-1989 [15], מתגברות על קללת המימדיות באמצעות שלושה עקרונות:

עיקרון 1 – קישוריות מקומית (Local Connectivity):
במקום לחבר כל פיקסל לכל נוירון (שיצטרך $1000^2 \times 1000^2 = 10^{12}$ משקלים), כל נוירון מחובר רק **לאזור קטן** (למשל, 3×3 פיקסלים).

עיקרון 2 – שיתוף משקלים (Weight Sharing):
אותו פילטר (קרנל) משמש **בכל מקום בתמונה**. במקום 10^{12} משקלים, נדרשים רק $3 \times 3 \times 64 = 576$ משקלים לשכבה!

עיקרון 3 – היררכיה של תכונות:
השכבות הראשונות לומדות תכונות פשוטות (קצוות, צבעים), והשכבות העמוקות לומדות תכונות מורכבות (עיניים, אוזניים, פנים).

השוואת מספר הפרמטרים:

טבלה 5: השוואת מספר פרמטרים: Fully Connected מול CNN

דגימות נדרשות (כלל 2)	מספר פרמטרים	ארכיטקטורה
10^{14} (בלתי אפשרי)	$\sim 10^{12}$	Fully Connected
6×10^9	$\sim 60 \times 10^6$	CNN (AlexNet)
2.5×10^9	$\sim 25 \times 10^6$	CNN (ResNet-50)

הצלחה מעשית:
רשת AlexNet [8] אומנה על 1.2 מיליון תמונות בלבד (הרבה פחות מהנדרש תיאורטית), והשיגה פריצת דרך בתחרות ImageNet ב-2012, עם הפחתה של 10% בשגיאה לעומת השיטות הקודמות.

2.8 פתרונות נוספים: הפחתת ממדיות

מלבד CNN, קיימות שיטות נוספות להתמודדות עם קללת המימדיות:

שיטה 1 – Feature Selection (בחירת תכונות):
בחירת תת-קבוצה של התכונות החשובות ביותר. שיטות נפוצות:

- **Filter Methods**: מיון לפי קורלציה, mutual information, chi-square

- **Wrapper Methods**: בחירה לפי ביצועי המודל (forward selection, backward elimination)

- **Embedded Methods**: רגולריזציה L1 (Lasso) שמאלצת משקלים לאפס

שיטה 2 – PCA (Principal Component Analysis):
המצאה של Karl Pearson (1901) [16] ו-Harold Hotelling (1933) [17]. PCA מוצא את הכיוונים בעלי השונות המקסימלית ומקרין עליהם.
רעיון PCA:
אם $\mathbf{X} \in \mathbb{R}^{n \times d}$ היא מטריצת הנתונים, PCA מוצא את הווקטורים העצמיים של מטריצת הקוואריאנס $\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$.
הווקטורים העצמיים $\{\vec{v}_1, \ldots, \vec{v}_d\}$ עם הערכים העצמיים הגדולים ביותר $\{\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d\}$ מהווים כיוונים של שונות מקסימלית.
הקרנה על k מרכיבים עיקריים:

$$\mathbf{Z} = \mathbf{X}\mathbf{V}_k \tag{13}$$

כאשר $\mathbf{V}_k = [\vec{v}_1, \ldots, \vec{v}_k]$ ו-$\mathbf{Z} \in \mathbb{R}^{n \times k}$ הוא הייצוג המופחת ($k \ll d$).
כמה מרכיבים לבחור?
כלל אצבע: בחר k כך ש**השונות המוסברת** היא לפחות 95%:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq 0.95 \tag{14}$$

פסאודו-קוד – PCA ב-NumPy:
הערה על SVD: Singular Value Decomposition הוא שיטה יעילה יותר לחישוב PCA מאשר חישוב ישיר של ערכים עצמיים, במיוחד כאשר $n \ll d$ [18].

2.9 תרגיל תכנות עצמי 2.1 – הפחתת ממדיות עם PCA

מטרה: להמחיש את יכולת PCA לשמר מידע תוך הפחתת ממדים.
משימה:

1. טענו את מערך הנתונים MNIST (תמונות ספרות כתובות ביד, $28 \times 28 = 784$ פיקסלים = 784 תכונות)
2. הפחיתו ל-50 מרכיבים עיקריים באמצעות PCA
3. שחזרו תמונות מהייצוג המופחת
4. השוו את התמונות המקוריות למשוחזרות
5. חשבו את אחוז השונות המוסברת

פסאודו-קוד מורחב:
תוצאה צפויה: עם 50 מרכיבים (במקום 784), ניתן לשמר ~85% מהמידע, והתמונות המשוחזרות יהיו קריאות למדי.

```python
import numpy as np

def pca_manual(X, n_components=2):
    """
␣␣␣␣מבצע␣PCA␣ידני␣באמצעות␣SVD.

␣␣␣␣Args:
␣␣␣␣␣␣␣␣X:␣מטריצת␣נתונים␣(n_samples,␣n_features)
␣␣␣␣␣␣␣␣n_components:␣מספר␣רכיבים␣עיקריים

␣␣␣␣Returns:
␣␣␣␣␣␣␣␣X_reduced:␣נתונים␣מופחתים␣(n_samples,␣n_components)
␣␣␣␣␣␣␣␣components:␣הווקטורים␣העצמיים
␣␣␣␣␣␣␣␣explained_var_ratio:␣שיעור␣השונות␣המוסברת
␣␣␣␣"""
    # ריכוז הנתונים (חיסור ממוצע)
    X_centered = X - np.mean(X, axis=0)

    # SVD - פירוק ערכים סינגולריים
    # U: eigenvectors של XX^T
    # S: ערכים סינגולריים (שורש של eigenvalues)
    # Vt: eigenvectors של X^TX (רכיבים עיקריים)
    U, S, Vt = np.linalg.svd(X_centered, full_matrices=False)

    # בחירת k רכיבים עיקריים
    components = Vt[:n_components]

    # הקרנה
    X_reduced = X_centered @ components.T

    # חישוב שונות מוסברת
    explained_variance = (S**2) / (len(X) - 1)
    explained_var_ratio = explained_variance[:n_components] / np.sum(
explained_variance)

    print(f"Explained␣variance␣ratio:␣{explained_var_ratio}")
    print(f"Total:␣{np.sum(explained_var_ratio):.3f}")

    return X_reduced, components, explained_var_ratio

# דוגמה
X = np.random.rand(1000, 100)  # 1000 דגימות, 100 תכונות
X_reduced, components, var_ratio = pca_manual(X, n_components=10)
print(f"Original␣shape:␣{X.shape},␣Reduced␣shape:␣{X_reduced.shape}")
```

PCA על MNIST – דוגמה מלאה

```python
from sklearn.datasets import fetch_openml
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np

# טעינת MNIST
mnist = fetch_openml('mnist_784', version=1, parser='auto')
X = mnist.data.to_numpy()[:1000]  # 1000 דגימותראשונות
y = mnist.target.to_numpy()[:1000]

# נרמולחשוב (!)
X = X / 255.0

print(f"Original␣shape:␣{X.shape}")  # (1000, 784)

# PCA להפחתהל -50 מרכיבים
pca = PCA(n_components=50)
X_reduced = pca.fit_transform(X)

print(f"Reduced␣shape:␣{X_reduced.shape}")  # (1000, 50)
print(f"Explained␣variance:␣{np.sum(pca.explained_variance_ratio_):.3f}"
)

# תונומתרוזחש
X_reconstructed = pca.inverse_transform(X_reduced)

# הצגה
fig, axes = plt.subplots(2, 5, figsize=(12, 5))
for i in range(5):
    # תירוקמהנומת
    axes[0, i].imshow(X[i].reshape(28, 28), cmap='gray')
    axes[0, i].set_title(f'Original:␣{y[i]}')
    axes[0, i].axis('off')

    # תרזחושמהנומת
    axes[1, i].imshow(X_reconstructed[i].reshape(28, 28), cmap='gray')
    axes[1, i].set_title(f'Reconstructed')
    axes[1, i].axis('off')

plt.tight_layout()
plt.show()
```

2.10 רגולריזציה: מניעת Overfitting במימד גבוה

כאשר מספר התכונות גדול ביחס למספר הדגימות, המודל נוטה ל**התאמת יתר** (Overfitting) – הוא "לומד בעל-פה" את נתוני האימון ונכשל בנתונים חדשים.

רגולריזציה (Regularization) היא טכניקה שמוסיפה "קנס" על מורכבות המודל, ובכך מעודדת פשטות.

שתי שיטות עיקריות:

רגולריזציה L2 (Ridge):

$$\mathcal{L}_{\text{Ridge}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{d} w_j^2 \tag{15}$$

קנס על **גודל המשקלים** – מעודד משקלים קטנים, אך לא מאפס אותם.

רגולריזציה L1 (Lasso):

$$\mathcal{L}_{\text{Lasso}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{d} |w_j| \tag{16}$$

קנס על **סכום ערכי המשקלים** – מאלץ חלק מהמשקלים להיות **בדיוק אפס**, ובכך מבצע בחירת תכונות אוטומטית.

פרמטר הקנס λ:

- $\lambda = 0$: אין רגולריזציה (סיכון ל-gnittifrevO)
- λ גדול מאוד: המודל פשוט מדי (Underfitting)
- λ אופטימלי: נקבע באמצעות Cross-Validation

מי המציא?

Ridge: Arthur Hoerl ו-treboR dranneK (1970) [19]

Lasso: Robert Tibshirani (1996) [20]

פסאודו-קוד – השוואת Ridge ו-ossaL:

תוצאה צפויה: Lasso יאפס ~ 490 מתוך 500 המשקלים, וישמור רק את התכונות האינפורמטיביות.

2.11 מחקר עדכני: אקסטרפולציה מול אינטרפולציה

מחקר פורץ דרך של Anadi Chaman ו-navI cinamkoD (2021) [21] גילה תוצאה מדהימה: **במרחבים עם יותר מ-100 ממדים, דגימה חדשה כמעט אף פעם אינה אינטרפולציה – היא כמעט תמיד אקסטרפולציה.**

הגדרות:

- **אינטרפולציה:** דגימה חדשה נמצאת בתוך המעטפת הקמורה (Convex Hull) של נתוני האימון

```python
from sklearn.linear_model import Ridge, Lasso
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
import numpy as np

# יצירת נתונים סינתטיים : n << d (קללת המימדיות!)
X, y = make_regression(
    n_samples=100,      # 100 דגימות בלבד
    n_features=500,     # 500 תכונות!
    n_informative=10,   # רק 10 רלוונטיות
    noise=10,
    random_state=42
)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Ridge
ridge = Ridge(alpha=1.0)  # alpha = lambda
ridge.fit(X_train, y_train)
ridge_score = ridge.score(X_test, y_test)

# Lasso
lasso = Lasso(alpha=1.0)
lasso.fit(X_train, y_train)
lasso_score = lasso.score(X_test, y_test)

print(f"Ridge␣R²:␣{ridge_score:.3f}")
print(f"Lasso␣R²:␣{lasso_score:.3f}")

# כמה משקלים Lasso אפס?
zero_weights = np.sum(np.abs(lasso.coef_) < 1e-5)
print(f"Lasso␣zeroed␣out␣{zero_weights}/{len(lasso.coef_)}␣features")
```

- **אקסטרפולציה:** דגימה חדשה נמצאת מחוץ למעטפת הקמורה

המעטפת הקמורה (Convex Hull):

$$\text{Conv}(\mathcal{X}) = \left\{ \sum_{i=1}^{n} \alpha_i \vec{x}_i \mid \vec{x}_i \in \mathcal{X}, \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i = 1 \right\} \tag{17}$$

התוצאה המפתיעה:

בניסויים על מערכי נתונים אמיתיים (MNIST, CIFAR-10), Chaman ו-cinamkoD מצאו ש-99.9% מדגימות הבדיקה הן אקסטרפולציה במרחב המקורי!

משמעות ל-IA:

כל פעם שמודל Deep Learning מסווג תמונה חדשה, הוא למעשה **משער מעבר לנתונים שראה**. ההצלחה של מודלים מודרניים נובעת מה**הנחות האינדוקטיביות** שלהם (כמו מבנה ה-NNC) ולא מכיסוי מלא של המרחב.

שאלת מחשבה שהעלה המרצה:

"אם כל תחזית היא אקסטרפולציה, האם AI באמת 'מבין' את העולם, או שהוא רק מנחש בצורה חכמה?"

זוהי אחת השאלות הפילוסופיות המרכזיות של AI מודרני.

2.12 סיכום ומבט קדימה

מה למדנו בפרק זה?

1. **קללת המימדיות היא אמיתית ומוכחת** – נפח המרחב גדל אקספוננציאלית, והנתונים הופכים דלילים

2. **מרחקים מתכנסים** – במימד גבוה, כל הנקודות נעשות "באותו מרחק", ומדדים כמו KNN קורסים

3. **יחס פיצ'ר-דגימה הוא קריטי** – נדרשות לפחות 30-100 דגימות לכל תכונה

4. **רשתות CNN מתגברות על הקללה** – שיתוף משקלים וקישוריות מקומית מפחיתים דרמטית את מספר הפרמטרים

5. **הפחתת ממדיות עובדת** – PCA, Feature Selection, Regularization מאפשרים עבודה יעילה

6. **רוב התחזיות הן אקסטרפולציה** – AI מצליח בזכות הנחות אינדוקטיביות, לא כיסוי מלא

מבט קדימה – פרק 3:

בפרק הבא נעבור מהבעיות לפתרונות. נחקור את **מקדם הקביעה** R^2 – הכלי המרכזי להערכת מודלי רגרסיה. נראה:

- הגדרה מתמטית מלאה של R^2
- הוכחה: למה R^2 תמיד בין 0 ל-1?
- סכנות: מתי R^2 מטעה?
- Adjusted R^2 – הגרסה המתוקנת
- קשר לקורלציה ולמכפלה סקלרית

שאלת מחשבה לסיום:
אם מודל רגרסיה ליניארית השיג R^2 = 0.95 על נתוני האימון, אך רק R^2 = 0.60 על נתוני הבדיקה – מה קרה? האם זו קללת המימדיות, Overfitting, או משהו אחר?

מטלות וקריאה מורחבת

תרגיל 2.1: הפחיתו את MNIST עם PCA (ראו פסאודו-קוד).
תרגיל 2.2: השוו Ridge ו-ossaL על נתונים עם $n < d$ (ראו פסאודו-קוד).
תרגיל 2.3: הוכיחו בעצמכם את משפט 2.1 (ריכוז הנפח בקליפה) עבור $d = 10$.
רמז: חשבו $V_d(r) = \frac{\pi^{d/2}}{\Gamma(d/2+1)}r^d$ עבור $r = 0.9$ ו-$r = 1$.
קריאה מורחבת:

- [6] – Dynamic Programming: המקור המקורי של "קללת המימדיות"
- [10] – "When Is 'Nearest Neighbor' Meaningful?": ניתוח מתמטי של קריסת KNN
- [21] – "Truly Generative or Just Extrapolation?": המחקר על אינטרפולציה מול אקסטרפולציה
- [9] – Deep Learning, פרק 7: רגולריזציה והפחתת Overfitting
- [22] – The Elements of Statistical Learning, פרק 3: רגרסיה ליניארית ורגולריזציה

שאלות להעמקה:

1. מדוע Lasso מאפס משקלים אך Ridge לא? רמז: חשבו על הגיאומטריה של הקנס (L_1 מול L_2)
2. האם קללת המימדיות משפיעה גם על רשתות Transformer ב-PLN?
3. אם 99.9% מהתחזיות הן אקסטרפולציה, איך מסבירים את ההצלחה של GPT-4?

סיום פרק 2

3 הערכת מודלים: מקדם הקביעה R^2

Model Evaluation: The Coefficient of Determination R^2

בפרק זה נחקור את אחד הכלים המרכזיים להערכת מודלי רגרסיה: **מקדם הקביעה** R^2 (R-squared). נבין את המשמעות המתמטית שלו, נוכיח מדוע הוא תמיד בין 0 ל-1, ונראה מתי הוא עלול להטעות אותנו.

3.1 השאלה המרכזית: מה זה מודל "טוב"?

דמיינו שבניתם מודל רגרסיה ליניארית לחיזוי מחיר דירה על סמך שטח, מיקום וגיל. המודל נותן תחזיות, אך **כיצד נדע אם הוא טוב?**

שאלה זו מעסיקה סטטיסטיקאים ומדעני נתונים מאז ראשית המדע. התשובה המודרנית, שהתגבשה במאה ה-20, היא **מקדם הקביעה** – מספר יחיד שמסכם את "טיב ההתאמה" של המודל לנתונים.

3.2 היסטוריה: מי המציא את R^2?

המושג פותח בהדרגה על ידי מספר סטטיסטיקאים:

Francis Galton (1822–1911) היה הראשון לחקור קורלציה ורגרסיה. בעבודתו "Regression towards Mediocrity in Hereditary Stature" (1886) [23], הוא גילה שילדים של הורים גבוהים נוטים להיות נמוכים יותר מהוריהם (רגרסיה לממוצע).

Karl Pearson (1857–1936), תלמידו של Galton, פיתח את **מקדם הקורלציה** r (1896) [24], שהוא השורש של R^2 במקרה של רגרסיה פשוטה.

Ronald Fisher (1890–1962) הרחיב את המושג ל**רגרסיה מרובה** (Multiple Regression) וניתח את חלוקת השונות [25].

Sewall Wright (1889–1988) טבע את הסימון R^2 ב-1921 [26].

3.3 הגדרה: מהו R^2?

הגדרה 3.1 – מקדם הקביעה:

R^2 הוא מדד סטטיסטי המצביע על **שיעור השונות במשתנה התלוי** Y **שמוסברת על ידי המשתנים הבלתי תלויים** X במודל רגרסיה.

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{18}$$

כאשר:

- SS_{res} – **סכום ריבועי השאריות** (Residual Sum of Squares)
- SS_{tot} – **סכום הריבועים הכולל** (Total Sum of Squares)

פירוט המשתנים:

$$SS_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{19}$$

$$SS_{\text{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{20}$$

כאשר:

- y_i – הערך האמיתי של התצפית ה-i
- $\hat{y}_i$ – הערך החזוי על ידי המודל עבור התצפית ה-i
- $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ – ממוצע הערכים האמיתיים

3.4 משמעות אינטואיטיבית: מהו "הסבר שונות"?

כדי להבין את R^2, נתחיל מ**מודל הבסיס** – המודל הפשוט ביותר האפשרי.
מודל הבסיס (ledoM enilesaB): תחזה תמיד את הממוצע $\bar{y}$.
זהו המודל הטיפש ביותר – הוא מתעלם מכל התכונות X ופשוט אומר "התשובה היא הממוצע". שגיאת מודל זה היא SS_{tot} – **השונות הכוללת** בנתונים.
המודל שלנו: משתמש ב-X כדי לחזות $\hat{y}_i$.
שגיאת המודל שלנו היא SS_{res} – **השונות שנותרה** אחרי שהמודל עשה את עבודתו.
הפרשנות:

$$R^2 = 1 - \frac{\text{שגיאת המודל שלנו}}{\text{שגיאת מודל הבסיס}} = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{21}$$

- אם $R^2 = 0$: המודל שלנו לא טוב יותר מהממוצע (חסר תועלת!)
- אם $R^2 = 1$: המודל מושלם – אין שגיאות כלל
- אם $R^2 = 0.8$: המודל מסביר 80% מהשונות

דוגמה מספרית:
נניח שאנחנו מנבאים מחירי דירות. הממוצע הוא 500000 ש"ח.

טבלה 6: דוגמה: חישוב R^2 למחירי דירות

תצפית	y_i **(אמיתי)**	$\hat{y}_i$ **(חזוי)**	$(y_i - \bar{y})^2$	$(y_i - \hat{y}_i)^2$
1	600000	580000	10000000000	400000000
2	450000	470000	2500000000	400000000
3	700000	680000	40000000000	400000000
סה"כ			$SS_{\text{tot}} = 52500000000$	$SS_{\text{res}} = 1200000000$

חישוב:

$$R^2 = 1 - \frac{1200000000}{52500000000} = 1 - 0.023 = 0.977$$

פרשנות: המודל מסביר 97.7% מהשונות במחירים – מודל מצוין!

3.5 הוכחה: R^2 תמיד בין 0 ל-1

משפט 3.1 – תחום R^2:

עבור רגרסיה ליניארית עם איבר חופשי, מתקיים: $0 \leq R^2 \leq 1$.

הוכחה:

חלק א': $R^2 \leq 1$

נוכיח ש-$SS_{\text{res}} \leq SS_{\text{tot}}$.

צעד 1: מודל הרגרסיה הליניארית ממזער את SS_{res} על ידי פתרון:

$$\min_{\vec{w}} \sum_{i=1}^{n} (y_i - \vec{w}^T \vec{x}_i)^2$$

צעד 2: מודל הבסיס (חיזוי הממוצע) הוא מקרה פרטי של רגרסיה ליניארית עם $\vec{w} = \vec{0}$ (למעט איבר חופשי = $\bar{y}$).

צעד 3: מכיוון שהרגרסיה ממזערת, היא לא יכולה להיות גרועה יותר ממודל הבסיס:

$$SS_{\text{res}} \leq SS_{\text{tot}}$$

צעד 4: לכן:

$$\frac{SS_{\text{res}}}{SS_{\text{tot}}} \leq 1 \quad \Rightarrow \quad 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \leq 1 \quad \Rightarrow \quad R^2 \leq 1 \quad \blacksquare$$

חלק ב': $R^2 \geq 0$

זה נכון **רק לרגרסיה ליניארית עם איבר חופשי**. במקרים מיוחדים (כמו רגרסיה דרך הראשית), R^2 יכול להיות שלילי!

עבור רגרסיה רגילה, המודל תמיד יכול לחזות את הממוצע (במקרה הגרוע ביותר), כך ש-$SS_{\text{res}} \leq SS_{\text{tot}}$ ולכן $R^2 \geq 0$.

$\blacksquare$

3.6 ייצוג גרפי: חלוקת השונות

ניתן לחשוב על השונות הכוללת כ"פאי" שמתחלק לשניים:

$$\underbrace{SS_{\text{tot}}}_{\text{שונות כוללת}} = \underbrace{SS_{\text{reg}}}_{\text{שונות מוסברת}} + \underbrace{SS_{\text{res}}}_{\text{שונות לא מוסברת}} \tag{22}$$

כאשר $SS_{\text{reg}} = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$ – השונות שהמודל "לכד".

קשר ל-R^2:

$$R^2 = 1 - \frac{\text{שגיאת המודל שלנו}}{\text{שגיאת מודל הבסיס}} = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{23}$$

זו הגרסה החיובית של ההגדרה – במקום "אחד פחות שאריות", זה "שיעור המוסבר".

3.7 קשר למקדם הקורלציה

במקרה של **רגרסיה ליניארית פשוטה** (משתנה בלתי תלוי אחד), מתקיים:

$$R^2 = r^2 \tag{24}$$

כאשר r הוא **מקדם הקורלציה של פירסון** בין X ו-Y:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{25}$$

הוכחה (סקיצה):

ברגרסיה פשוטה, $\hat{y}_i = a + bx_i$ כאשר:

$$b = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = r \cdot \frac{\sigma_y}{\sigma_x}$$

ניתן להוכיח (באלגברה מייגעת אך ישירה) ש-$SS_{\text{reg}} = r^2 \cdot SS_{\text{tot}}$, ולכן $R^2 = r^2$.

■

משמעות: ברגרסיה פשוטה, R^2 הוא ריבוע הקורלציה. אם $r = 0.9$ (קורלציה גבוהה), אזי $R^2 = 0.81$.

3.8 סכנה 1: R^2 עולה תמיד כשמוסיפים תכונות

הבעיה הקריטית של R^2: **הוא אף פעם לא יורד כשמוסיפים תכונות**, גם אם הן חסרות תועלת!

משפט 3.2 – מונוטוניות R^2:

אם מוסיפים תכונה למודל רגרסיה, R^2 לא יורד (ובדרך כלל עולה).

הוכחה:

צעד 1: נניח מודל עם p תכונות מושג R_p^2. נוסיף תכונה נוספת (סה"כ $p+1$).
צעד 2: המודל החדש יכול תמיד לבחור משקל $w_{p+1} = 0$ לתכונה החדשה, ובכך להשיג את אותה שגיאה כמו המודל הישן.
צעד 3: מכיוון שהרגרסיה ממזערת את SS_{res}, היא תמצא משקלים טובים או שווים:

$$SS_{\text{res}}^{(p+1)} \leq SS_{\text{res}}^{(p)}$$

צעד 4: לכן:

$$R_{p+1}^2 = 1 - \frac{SS_{\text{res}}^{(p+1)}}{SS_{\text{tot}}} \geq 1 - \frac{SS_{\text{res}}^{(p)}}{SS_{\text{tot}}} = R_p^2 \quad \blacksquare$$

המשמעות המסוכנת:
אם נוסיף 1000 תכונות אקראיות (רעש טהור), R^2 יעלה! זה יוביל אותנו לחשוב שהמודל השתפר, אך באמת הוא פשוט **התאים יתר** (Overfitted).
דוגמה מספרית – סימולציה:
תוצאה צפויה:

טבלה 7: עליית R^2 עם הוספת תכונות אקראיות

R^2 **(אימון)**	**מספר תכונות**
0.850	1
0.852	2
0.870	6
0.895	11
0.930	21
0.975	51

R^2 עלה מ-0.85 ל-0.975 למרות שהתכונות הנוספות היו רעש טהור!

3.9 הפתרון: Adjusted R^2

Adjusted R^2 (מקדם הקביעה המתוקן) פותר את הבעיה על ידי הטלת **"קנס"** על הוספת תכונות.
הגדרה 3.2 – Adjusted R^2**:**

$$R_{\text{adj}}^2 = 1 - \frac{SS_{\text{res}}/(n-p-1)}{SS_{\text{tot}}/(n-1)} \tag{26}$$

כאשר:

- n – מספר התצפיות

הדגמת בעיית R^2 עם תכונות אקראיות

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# נתוניםפשוטים : Y = 2X + רעש
np.random.seed(42)
n = 100
X_real = np.random.rand(n, 1)
y = 2 * X_real.flatten() + np.random.randn(n) * 0.5

# נבדוק R² כפונקציהשלמספרתכונותאקראיות
r2_scores = []

for num_random_features in [0, 1, 5, 10, 20, 50]:
    # הוספתתכונותאקראיותרעש   (!)
    X_random = np.random.randn(n, num_random_features)
    X_combined = np.hstack([X_real, X_random])

    # אימון
    model = LinearRegression()
    model.fit(X_combined, y)

    # R² עלנתוניהאימון
    y_pred = model.predict(X_combined)
    r2 = r2_score(y, y_pred)
    r2_scores.append(r2)

    print(f"Num␣features:␣{1␣+␣num_random_features},␣R²:␣{r2:.4f}")

# תוצאה: R² עולהעםכלתכונה   , גםאםהיארעש   !
```

- p – מספר התכונות (לא כולל איבר חופשי)

ניסוח חלופי:

$$R^2_{\text{adj}} = 1 - (1 - R^2) \cdot \frac{n-1}{n-p-1} \tag{27}$$

משמעות:

המונה והמכנה מחולקים במספר **דרגות החופש** (Degrees of Freedom). ככל שיש יותר תכונות p, דרגות החופש קטנות יותר, והקנס גדל.

תכונות של Adjusted R^2:

- **יכול לרדת** כשמוסיפים תכונה חסרת תועלת
- **יכול להיות שלילי** (אם המודל גרוע מאוד)
- תמיד $R^2_{\text{adj}} \leq R^2$

מתי להשתמש?

- R^2: להערכת התאמה על נתוני אימון בודדים
- **Adjusted** R^2: להשוואה בין מודלים עם מספר תכונות שונה

חזרה על הסימולציה עם Adjusted R^2:

תוצאה צפויה:

טבלה 8: השוואה: R^2 מול Adjusted R^2

תכונות	R^2	Adjusted R^2
1	0.850	0.849
2	0.852	0.849
6	0.870	0.862
11	0.895	0.880
21	0.930	0.900
51	0.975	0.895

Adjusted R^2 יורד כשמוסיפים תכונות חסרות תועלת – בדיוק כמו שצריך!

3.10 סכנה 2: R^2 לא מתאים לסיווג

R^2 מיועד **לרגרסיה** (תחזית של ערך מספרי). עבור **סיווג** (תחזית של קטגוריה), הוא לא מתאים.

למה?

בסיווג, $y_i \in \{0, 1\}$ אך $\hat{y}_i$ יכול להיות כל מספר (למשל, 0.73 = הסתברות לקטגוריה 1). חישוב SS_{res} לא משקף את דיוק הסיווג.

מונע עליה מלאכותית – Adjusted R^2

```python
def adjusted_r2(r2, n, p):
    """
␣␣␣␣חישוב␣Adjusted␣R².

␣␣␣␣Args:
␣␣␣␣␣␣␣␣r2:␣R²␣רגיל
␣␣␣␣␣␣␣␣n:␣מספר␣דגימות
␣␣␣␣␣␣␣␣p:␣מספר␣תכונות␣(לא␣כולל␣intercept)
␣␣␣␣"""
    return 1 - (1 - r2) * (n - 1) / (n - p - 1)

# חישוב␣מחדש
for i, num_random in enumerate([0, 1, 5, 10, 20, 50]):
    r2 = r2_scores[i]
    n = 100
    p = 1 + num_random
    adj_r2 = adjusted_r2(r2, n, p)

    print(f"Features:␣{p},␣R²:␣{r2:.4f},␣Adj␣R²:␣{adj_r2:.4f}")
```

מדדים נכונים לסיווג:

טבלה 9: מדדי הערכה למשימות שונות

משימה	מדד	הסבר
רגרסיה	R^2, RMSE, MAE	R^2 מודד שיעור שונות מוסברת
סיווג	Accuracy, Precision, Recall, F1	מודדים אחוז סיווגים נכונים
סיווג (התפלגות)	Log-Loss, AUC-ROC	מודדים איכות הסתברויות

3.11 קשר לאלגברה ליניארית: הקרנות

ניתן לראות רגרסיה ליניארית כ**הטלה** של $\vec{y}$ על המרחב שנפרש על ידי עמודות המטריצה $\mathbf{X}$.

נוסחת הרגרסיה במטריצות:

$$\vec{\hat{y}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\vec{y} = \mathbf{P}\vec{y} \tag{28}$$

כאשר $\mathbf{P} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ היא **מטריצת ההטלה** (Projection Matrix).

משמעות גיאומטרית של R^2:

$$R^2 = \frac{\|\vec{\hat{y}} - \bar{y}\vec{1}\|^2}{\|\vec{y} - \bar{y}\vec{1}\|^2} = \cos^2(\theta)$$

כאשר θ היא הזווית בין $\vec{y} - \bar{y}\vec{1}$ (הנתונים הממורכזים) ל-$\vec{\hat{y}} - \bar{y}\vec{1}$ (החיזוי הממורכז).

פרשנות: R^2 הוא ריבוע הקוסינוס של הזווית בין הנתונים לחיזוי במרחב הממורכז – **מדד גיאומטרי של התאמה**.

3.12 תרגיל תכנות עצמי 3.1 – חישוב R^2 ידני

מטרה: להבין את המשמעות של R^2 על ידי חישוב ידני.

משימה:

1. צרו נתונים סינתטיים: שער + $y = 3x + 5$
2. אמנו מודל רגרסיה ליניארית
3. חשבו את R^2 בשלוש דרכים:
   - נוסחה ישירה: $1 - SS_{\text{res}}/SS_{\text{tot}}$
   - באמצעות קורלציה: r^2
   - באמצעות sklearn
4. השוו את התוצאות (צריכות להיות זהות!)

פסאודו-קוד מלא:

תוצאה צפויה: כל שלוש השיטות יתנו $R^2 \approx 0.95$ (תלוי ברעש).

3.13 תרגיל תכנות עצמי 3.2 – השוואת R^2 ו-detsujdA R^2

מטרה: להמחיש את ההבדל בין R^2 ל-detsujdA R^2 כאשר מוסיפים תכונות.

משימה:

1. צרו נתונים עם תכונה אחת רלוונטית
2. הוסיפו בהדרגה תכונות אקראיות (רעש)
3. עקבו אחרי R^2 ו-detsujdA R^2 כפונקציה של מספר התכונות
4. הציגו גרף: ציר X = מספר תכונות, ציר Y = ציון

תוצאה צפויה: R^2 עולה בהתמדה, אך Adjusted R^2 יורד אחרי כמה תכונות.

חישוב R^2 בשלוש דרכים

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# 1. יצירת נתונים
np.random.seed(42)
n = 100
X = np.random.rand(n, 1) * 10  # X בין 0 ל-10
y = 3 * X.flatten() + 5 + np.random.randn(n) * 2  # y = 3x + 5 + רעש

# 2. אימון מודל
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

print(f"המודל שנלמד: y = {model.coef_[0]:.3f}x + {model.intercept_:.3f}"
)

# 3. דרך 1: נוסחה ישירה
y_mean = np.mean(y)
SS_tot = np.sum((y - y_mean)**2)
SS_res = np.sum((y - y_pred)**2)
r2_manual = 1 - (SS_res / SS_tot)

print(f"\nדרך 1 נוסחה (ישירה):")
print(f"  SS_tot = {SS_tot:.2f}")
print(f"  SS_res = {SS_res:.2f}")
print(f"  R² = {r2_manual:.6f}")

# 4. דרך 2: קורלציה בריבוע  (לרגרסיה פשוטה!)
correlation = np.corrcoef(X.flatten(), y)[0, 1]
r2_from_corr = correlation**2

print(f"\nדרך 2 (קורלציה):")
print(f"  r = {correlation:.6f}")
print(f"  R² = r² = {r2_from_corr:.6f}")

# 5. דרך 3: sklearn
r2_sklearn = r2_score(y, y_pred)

print(f"\nדרך 3 (sklearn):")
print(f"  R² = {r2_sklearn:.6f}")

# 6. השוואה
print(f"\nהכל זהה? {np.allclose([r2_manual, r2_from_corr, r2_sklearn], 
r2_sklearn)}")
```

גרף השוואה – Adjusted R^2 מול R^2

```
import matplotlib.pyplot as plt

def adjusted_r2(r2, n, p):
    return 1 - (1 - r2) * (n - 1) / (n - p - 1)

# נתונים
np.random.seed(42)
n = 100
X_real = np.random.rand(n, 1)
y = 2 * X_real.flatten() + np.random.randn(n) * 0.5

# מעקב
num_features_list = []
r2_list = []
adj_r2_list = []

for num_random in range(0, 51):
    # הוספתתכונותאקראיות
    if num_random == 0:
        X_combined = X_real
    else:
        X_random = np.random.randn(n, num_random)
        X_combined = np.hstack([X_real, X_random])

    # אימון
    model = LinearRegression()
    model.fit(X_combined, y)
    y_pred = model.predict(X_combined)

    # חישוב
    r2 = r2_score(y, y_pred)
    p = X_combined.shape[1]
    adj_r2 = adjusted_r2(r2, n, p)

    num_features_list.append(p)
    r2_list.append(r2)
    adj_r2_list.append(adj_r2)

# גרף
plt.figure(figsize=(10, 6))
plt.plot(num_features_list, r2_list, label='R²', marker='o', markersize
=3)
plt.plot(num_features_list, adj_r2_list, label='Adjusted␣R²', marker='s'
, markersize=3)
plt.xlabel('Number␣of␣Features')
plt.ylabel('Score')
plt.title('R²␣vs␣Adjusted␣R²:␣Adding␣Random␣Features')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

3.14 מקרי קצה: מתי R^2 מטעה?

מקרה 1: יחסים לא-ליניאריים

אם הקשר בין X ל-Y הוא לא-ליניארי (למשל, ריבועי), רגרסיה ליניארית תיכשל ו-R^2 יהיה נמוך – אך זה לא אומר שאין קשר!

דוגמה: $y = x^2$

רגרסיה ליניארית תיתן R^2 נמוך, אך הקשר מושלם (רק לא ליניארי).

פתרון: השתמש ב**פולינומים** או מודלים לא-ליניאריים (כמו רשתות נוירונים).

מקרה 2: Outliers (ערכי חריגים)

ערך חריג אחד יכול להוריד דרמטית את R^2, גם אם המודל טוב ל-99% מהנתונים.

פתרון: זהה וטפל ב-sreiltuO לפני אימון (באמצעות Z-score, IQR, או שיטות עמידות כמו RANSAC).

מקרה 3: מתאם אינו סיבתיות

R^2 גבוה לא אומר ש-X גורם ל-Y. ייתכן שיש משתנה שלישי Z שגורם לשניהם.

דוגמה קלאסית: צריכת גלידה וטביעות מים מתואמות – לא כי גלידה גורמת לטביעות, אלא כי **קיץ** גורם לשניהם.

3.15 סיכום ומבט קדימה

מה למדנו בפרק זה?

1. R^2 **מודד שיעור שונות מוסברת** – מספר יחיד שמסכם את איכות המודל
2. **תחום: 0 עד 1** – הוכחנו מתמטית למה זה תמיד נכון (לרגרסיה עם intercept)
3. **קשר לקורלציה** – ברגרסיה פשוטה, $R^2 = r^2$
4. **בעיה: עולה תמיד** – הוספת תכונות מעלה את R^2 גם אם הן חסרות תועלת
5. **פתרון: Adjusted R^2** – מנרמל לפי מספר התכונות ומונע Overfitting
6. **לא לסיווג!** – R^2 מיועד לרגרסיה, לא לסיווג
7. **מגבלות** – מתאם אינו סיבתיות, רגיש ל-sreiltuO, מניח ליניאריות

מבט קדימה – פרק 4:

בפרק הבא נעבור מהערכת מודלים למדידת **קשרים בין משתנים**. נחקור:

- **קו-וריאנס** (Covariance) – מדד גולמי של קשר ליניארי
- **קורלציה** (Correlation) – הגרסה המנורמלת של קו-וריאנס
- **ייצוג אלגברי ליניארי** – קו-וריאנס כמכפלה סקלרית, קורלציה כ-ytiralimiS enisoC
- **סכנות** – "Correlation is not Causation", קורלציות מזויפות

- **מטריצת קורלציה** – כלי לזיהוי Multicollinearity

שאלת מחשבה לסיום:
אם שני משתנים X ו-Y בעלי קורלציה $r = 0$ (אורתוגונליים), האם זה אומר שהם בלתי תלויים סטטיסטית?
רמז: התשובה היא **לא!** קורלציה אפס משמעה רק אין קשר ליניארי, אך עדיין יכול להיות קשר לא-ליניארי (למשל, $Y = X^2$).

מטלות וקריאה מורחבת

תרגיל 3.1: חשבו R^2 ידנית בשלוש דרכים (ראו פסאודו-קוד).
תרגיל 3.2: השוו R^2 ו-Adjusted R^2 עם תכונות אקראיות (ראו פסאודו-קוד).
תרגיל 3.3: הוכיחו בעצמכם שברגרסיה פשוטה $R^2 = r^2$.
רמז: התחילו מהביטוי $SS_{\text{reg}} = \sum(\hat{y}_i - \bar{y})^2$ והשתמשו ב-$\hat{y}_i = a + bx_i$.
קריאה מורחבת:

- [23] – "Regression towards Mediocrity in Hereditary Stature": המאמר המקורי של Galton
- [24] – "Mathematical Contributions to the Theory of Evolution": Pearson על קורלציה
- [25] – "Statistical Methods for Research Workers": Fisher על ניתוח שונות
- [26] – "Correlation and Causation": Wright טובע את R^2
- [22] – The Elements of Statistical Learning, פרק 3: רגרסיה ליניארית ברזולוציה גבוהה

שאלות להעמקה:

1. למה Adjusted R^2 יכול להיות שלילי, אך R^2 לא?
2. מה יקרה ל-R^2 אם ננרמל את X ו-Y (נחסיר ממוצע ונחלק בסטיית תקן)?
3. איך R^2 מתייחס למודלים לא-ליניאריים (כמו רשתות נוירונים)?

סיום פרק 3

4 קו-וריאנס וקורלציה: מדידת קשרים בין משתנים

Covariance and Correlation: Measuring Relationships Between Variables

בפרק זה נחקור את הכלים המתמטיים למדידת קשרים בין משתנים. נראה כיצד קו-וריאנס וקורלציה מהווים הרחבה טבעית של מכפלה סקלרית ו-ytiralimiS enisoC, ונבין מדוע "**מתאם אינו סיבתיות**" הוא אחד העקרונות החשובים ביותר במדע הנתונים.

4.1 השאלה המרכזית: איך מודדים "יחד-משתנות"?

דמיינו שאנחנו בוחנים את הקשר בין גובה ומשקל של אנשים. אינטואיטיבית, אנחנו מצפים שאנשים גבוהים יותר יהיו בדרך כלל כבדים יותר – הם "משתנים יחד". אך **כיצד נכמת את הקשר הזה?**

התשובה המתמטית היא **קו-וריאנס** (Covariance) – מדד שמצביע על מידת ה"יחד-משתנות" של שני משתנים.

4.2 היסטוריה: מי המציא את הקו-וריאנס?

המושג פותח במקביל להתפתחות הסטטיסטיקה המודרנית:

Francis Galton (1822–1911) היה הראשון לחקור קשרים בין משתנים במחקריו על תורשה. בעבודתו על גובה הורים וילדים (1886) [23], הוא גילה שהמשתנים "משתנים יחד" באופן שיטתי.

Karl Pearson (1857–1936) פיתח את **מקדם הקורלציה** r ב-1896 [24], והגדיר באופן פורמלי את הקו-וריאנס כבסיס למקדם זה.

Ronald Fisher (1890–1962) הרחיב את התורה ל**מטריצות קו-וריאנס** (Covariance Matrices) ולניתוח רב-משתני [25].

4.3 הגדרה: מהי קו-וריאנס?

הגדרה 4.1 – קו-וריאנס:

עבור שני משתנים X ו-Y עם n תצפיות, הקו-וריאנס מוגדרת כ:

$$\text{Cov}(X, Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{29}$$

כאשר $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ ו-$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ הם הממוצעים.

הערה על הגדרות שונות:

חלק מהספרים משתמשים בחלוקה ב-$(n-1)$ במקום n (קו-וריאנס מדגמית):

$$\text{Cov}_{\text{sample}}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{30}$$

החלוקה ב-$(n-1)$ נותנת **אומד לא מוטה** (Unbiased Estimator) לקו-ווריאנס האוכלוסייה. לצרכי למידת מכונה, ההבדל זניח כאשר n גדול.

4.4 משמעות אינטואיטיבית: מה אומר הסימן?

פרשנות לפי סימן הקו-ווריאנס:

טבלה 10: פרשנות הסימן של קו-ווריאנס

משמעות	ערך
קשר חיובי: כאשר X גדל, Y נוטה לגדול	$\mathrm{Cov}(X,Y) > 0$
אין קשר ליניארי (לא בהכרח בלתי תלויים!)	$\mathrm{Cov}(X,Y) = 0$
קשר שלילי: כאשר X גדל, Y נוטה לקטון	$\mathrm{Cov}(X,Y) < 0$

דוגמה מספרית:

נניח שיש לנו נתוני גובה ומשקל של 5 אנשים:

טבלה 11: דוגמה: גובה ומשקל

משקל (ק"ג)	גובה (ס"מ)	אדם
55	160	1
65	170	2
75	180	3
70	175	4
60	165	5
65	170	**ממוצע**

חישוב:

$$\begin{aligned}\mathrm{Cov}(X,Y) &= \frac{1}{5}\big[(160-170)(55-65)+(170-170)(65-65)\\ &\quad+(180-170)(75-65)+(175-170)(70-65)\\ &\quad+(165-170)(60-65)\big]\\ &= \frac{1}{5}\big[(-10)(-10)+(0)(0)+(10)(10)+(5)(5)+(-5)(-5)\big]\\ &= \frac{1}{5}\big[100+0+100+25+25\big] = \frac{250}{5} = 50 \end{aligned} \tag{31}$$

$\mathrm{Cov}(X,Y) = 50 > 0 \rightarrow$ קשר חיובי בין גובה למשקל.

4.5 קשר לאלגברה ליניארית: קו-ווריאנס כמכפלה סקלרית

נזכור מפרק 1 את ההגדרה של מכפלה סקלרית:

$$\langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{n} u_i v_i \tag{32}$$

קו-ווריאנס היא מכפלה סקלרית של וקטורים ממורכזים!
אם נגדיר:

- $\vec{x}_c = [x_1 - \bar{x}, x_2 - \bar{x}, \ldots, x_n - \bar{x}]^T$ – וקטור X ממורכז
- $\vec{y}_c = [y_1 - \bar{y}, y_2 - \bar{y}, \ldots, y_n - \bar{y}]^T$ – וקטור Y ממורכז

אזי:

$$\text{Cov}(X, Y) = \frac{1}{n} \langle \vec{x}_c, \vec{y}_c \rangle = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \tag{33}$$

משמעות גיאומטרית:
קו-ווריאנס מודדת את **"ההטיה המשותפת"** של שני משתנים מהממוצע שלהם – זו מכפלה סקלרית של הסטיות!

4.6 הבעיה עם קו-ווריאנס: חוסר סקלה

הבעיה המרכזית של קו-ווריאנס: **הערך שלה תלוי ביחידות המדידה.**
דוגמה בעייתית:

- $\text{Cov}(\text{גובה בס"מ}, \text{משקל בק"ג}) = 50$
- $\text{Cov}(\text{גובה במטרים}, \text{משקל בק"ג}) = 0.005$

אותו קשר, אבל ערכים שונים לחלוטין! **לא ניתן להשוות קו-ווריאנסים מזוגות משתנים שונים.**
זו הסיבה שאנחנו זקוקים ל**קורלציה** – גרסה מנורמלת של קו-ווריאנס.

4.7 קורלציה: הגרסה המנורמלת

הגדרה 4.2 – מקדם הקורלציה של פירסון:
מקדם הקורלציה בין X ו-Y מוגדר כ:

$$r = \text{Cor}(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y} \tag{34}$$

כאשר:

- $\sigma_X = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}$ – **סטיית התקן של** X
- $\sigma_Y = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2}$ – **סטיית התקן של** Y

נוסחה מפורשת:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{35}$$

הכרת הנוסחה?

זו **בדיוק Cosine Similarity** של וקטורים ממורכזים! (ראו פרק 1)

4.8 קשר לאלגברה ליניארית: קורלציה כ-Cosine Similarity

נזכור מפרק 1:

$$\cos(\theta) = \frac{\langle \vec{u}, \vec{v} \rangle}{\|\vec{u}\| \cdot \|\vec{v}\|} \tag{36}$$

אם $\vec{x}_c$ ו-$\vec{y}_c$ הם וקטורים ממורכזים, אזי:

$$r = \frac{\langle \vec{x}_c, \vec{y}_c \rangle}{\|\vec{x}_c\| \cdot \|\vec{y}_c\|} = \cos(\theta) \tag{37}$$

פרשנות גיאומטרית מרתקת:

קורלציה היא הקוסינוס של הזווית בין שני משתנים ממורכזים במרחב התצפיות!

- $r = 1$ (זווית $0°$) $\rightarrow$ קשר ליניארי חיובי מושלם
- $r = 0$ (זווית $90°$) $\rightarrow$ אורתוגונליים, אין קשר ליניארי
- $r = -1$ (זווית $180°$) $\rightarrow$ קשר ליניארי שלילי מושלם

4.9 משפט: תחום הקורלציה

משפט 4.1 – תחום מקדם הקורלציה:

לכל שני משתנים X ו-Y, מתקיים: $-1 \leq r \leq 1$.

הוכחה:

שיטה 1: אי-שוויון קושי-שוורץ

אי-שוויון קושי-שוורץ (Cauchy-Schwarz Inequality) קובע שלכל וקטורים $\vec{u}, \vec{v}$:

$$|\langle \vec{u}, \vec{v} \rangle| \leq \|\vec{u}\| \cdot \|\vec{v}\| \tag{38}$$

כאשר השוויון מתקיים אם ורק אם הווקטורים פרופורציונליים ($\vec{v} = c\vec{u}$).

צעד 1: נישם $\vec{u} = \vec{x}_c$ ו-$\vec{v} = \vec{y}_c$ (ווקטורים ממורכזים).

צעד 2: לפי קושי-שוורץ:

$$|\langle \vec{x}_c, \vec{y}_c \rangle| \leq \|\vec{x}_c\| \cdot \|\vec{y}_c\|$$

צעד 3: חלוקה ב-$\|\vec{x}_c\| \cdot \|\vec{y}_c\|$ נותנת:

$$\left| \frac{\langle \vec{x}_c, \vec{y}_c \rangle}{\|\vec{x}_c\| \cdot \|\vec{y}_c\|} \right| \leq 1$$

צעד 4: אך זה בדיוק $|r|$, לכן:

$$|r| \leq 1 \quad \Rightarrow \quad -1 \leq r \leq 1 \quad \blacksquare$$

מתי מתקיים $r = \pm 1$?
השוויון מתקיים כאשר $\vec{y}_c = c\vec{x}_c$ – כלומר, Y ו-X קשורים ליניארית: $Y = a + bX$.

4.10 דוגמאות ויזואליות: קורלציות שונות

פסאודו-קוד – יצירת דוגמאות קורלציה:

תוצאה צפויה: גרפים המראים כיצד הנקודות "מסתדרות" בצורה ליניארית ככל שהקורלציה חזקה יותר.

4.11 תרגיל תכנות עצמי 4.1 – חישוב קורלציה ידני

מטרה: להבין את הקשר בין קו-וריאנס, קורלציה, ומכפלה סקלרית.
משימה:

1. צרו שני משתנים X ו-Y עם קורלציה ידועה
2. חשבו קו-וריאנס בשלוש דרכים:
   - נוסחה ישירה
   - מכפלה סקלרית של וקטורים ממורכזים
   - באמצעות NumPy
3. חשבו קורלציה בשלוש דרכים:
   - מקו-וריאנס וסטיות תקן
   - Cosine Similarity של וקטורים ממורכזים
   - באמצעות NumPy
4. השוו את כל התוצאות

תוצאה צפויה: כל השיטות יתנו אותה תוצאה, מה שמדגים את הקשר העמוק בין קורלציה ומכפלה סקלרית.

דוגמאות קורלציה עם ויזואליזציה

```python
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)
n = 100
x = np.random.randn(n)

# קורלציותשונות
correlations = [1.0, 0.8, 0.5, 0.0, -0.5, -0.8, -1.0]

fig, axes = plt.subplots(2, 4, figsize=(16, 8))
axes = axes.flatten()

for i, r_target in enumerate(correlations):
    # יצירת Y עםקורלציהרצויה
    noise = np.random.randn(n)
    y = r_target * x + np.sqrt(1 - r_target**2) * noise

    # חישובקורלציהבפועל
    r_actual = np.corrcoef(x, y)[0, 1]

    # ציור
    axes[i].scatter(x, y, alpha=0.6)
    axes[i].set_title(f'r␣=␣{r_actual:.2f}')
    axes[i].set_xlabel('X')
    axes[i].set_ylabel('Y')
    axes[i].grid(True, alpha=0.3)

axes[-1].axis('off')
plt.tight_layout()
plt.show()
```

חישוב קורלציה וקו-וריאנס ידני

```
import numpy as np

# נתונים
x = np.array([160, 170, 180, 175, 165])
y = np.array([55, 65, 75, 70, 60])

print("===␣דרך␣1:␣נוסחה␣ישירה␣===")
# ממוצעים
x_mean = np.mean(x)
y_mean = np.mean(y)

# קווריאנס-
cov_manual = np.mean((x - x_mean) * (y - y_mean))
print(f"Cov(X,Y)␣=␣{cov_manual:.2f}")

# סטיותתקן
sigma_x = np.std(x)
sigma_y = np.std(y)

# קורלציה
r_manual = cov_manual / (sigma_x * sigma_y)
print(f"r␣=␣{r_manual:.4f}")

print("\n===␣דרך␣2:␣אלגברה␣לינארית␣===")
# וקטוריםממורכזים
x_c = x - x_mean
y_c = y - y_mean

# קווריאנס- כמכפלהסקלרית
cov_dot = np.dot(x_c, y_c) / len(x)
print(f"Cov(X,Y)␣=␣{cov_dot:.2f}")

# קורלציהכ -Cosine Similarity
r_cosine = np.dot(x_c, y_c) / (np.linalg.norm(x_c) * np.linalg.norm(y_c)
)
print(f"r␣=␣{r_cosine:.4f}")

print("\n===␣דרך␣3:␣NumPy␣===")
# מטריצתקווריאנס -
cov_matrix = np.cov(x, y, ddof=0)  # ddof=0 לחלוקהב -n
cov_numpy = cov_matrix[0, 1]
print(f"Cov(X,Y)␣=␣{cov_numpy:.2f}")

# מטריצתקורלציה
corr_matrix = np.corrcoef(x, y)
r_numpy = corr_matrix[0, 1]
print(f"r␣=␣{r_numpy:.4f}")

print("\n===␣השוואה␣===")
print(f"Cov␣-␣כל␣השיטות␣זהות?␣{np.allclose([cov_manual,␣cov_dot,␣
```

4.12 מטריצת הקורלציה

כאשר יש d משתנים, ניתן לארגן את כל הקורלציות במטריצה.

הגדרה 4.3 – מטריצת קורלציה:

עבור d משתנים $X_1, X_2, \ldots, X_d$, מטריצת הקורלציה $\mathbf{R} \in \mathbb{R}^{d\times d}$ מוגדרת כ:

$$\mathbf{R}_{ij} = \text{Cor}(X_i, X_j) \tag{39}$$

תכונות מטריצת הקורלציה:

1. **סימטרית:** $\mathbf{R} = \mathbf{R}^T$ (כי $\text{Cor}(X_i, X_j) = \text{Cor}(X_j, X_i)$)
2. **האלכסון כולו 1:** $\mathbf{R}_{ii} = 1$ (כי $\text{Cor}(X_i, X_i) = 1$)
3. **חיובית חצי-מוגדרת** (Positive Semi-Definite): כל הערכים העצמיים ≥ 0

דוגמה – מטריצת קורלציה לשלושה משתנים:

$$\mathbf{R} = \begin{bmatrix} 1.00 & 0.85 & 0.62 \\ 0.85 & 1.00 & 0.71 \\ 0.62 & 0.71 & 1.00 \end{bmatrix}$$

פרשנות: המשתנים הראשון והשני בעלי קורלציה חזקה (0.85).

שימוש למעשי – זיהוי Multicollinearity:

ברגרסיה ליניארית, אם שני משתנים בלתי תלויים קשורים מאוד ($|r| > 0.9$), זו בעיה שנקראת **Multicollinearity**. המודל לא יכול להפריד בין השפעתם, והמשקלים הופכים לא יציבים.

פתרון: הסר אחד מהמשתנים או השתמש ב-noissergeR egdiR/ACP.

פסאודו-קוד – ויזואליזציה של מטריצת קורלציה:

4.13 הסכנה הגדולה: מתאם אינו סיבתיות

זהו אחד העקרונות החשובים ביותר במדע הנתונים: **"Correlation is not Causation"**.

קורלציה גבוהה בין X ו-Y לא אומרת ש-X גורם ל-Y!

שלוש אפשרויות לקורלציה:

1. **סיבתיות ישירה:** X גורם ל-Y (למשל: עישון $\rightarrow$ סרטן ריאות)
2. **סיבתיות הפוכה:** Y גורם ל-X (הפוך ממה שחשבנו!)
3. **משתנה מבלבל** (Confounding Variable): משתנה שלישי Z גורם גם ל-X וגם ל-Y

Heatmap של מטריצת קורלציה

```python
import seaborn as sns
import pandas as pd

# דוגמה: Iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# מטריצתקורלציה
corr_matrix = df.corr()

# Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0,
            square=True, linewidths=1, cbar_kws={"shrink": 0.8})
plt.title('Correlation Matrix - Iris Dataset')
plt.tight_layout()
plt.show()

# זיהוי Multicollinearity
high_corr = (corr_matrix.abs() > 0.9) & (corr_matrix != 1)
if high_corr.any().any():
    print("Warning: High correlation detected (Multicollinearity)")
    print(corr_matrix[high_corr])
```

דוגמאות קלאסיות לקורלציות מזויפות:

טבלה 12: דוגמאות לקורלציות מזויפות

משתנה מבלבל Z	משתנה Y	משתנה X
חום הקיץ	מספר טביעות מים	צריכת גלידה
גודל השריפה	נזק משריפות	מספר כבאים
גיל (ילדים)	יכולת קריאה	גודל נעליים
מקריות סטטיסטית	טביעות בבריכות	מכירות Nicolas Cage

אתר מפורסם לקורלציות מזויפות: tylervigen.com/spurious-correlations

4.14 מחקר מקרה: עישון וסרטן

ההוכחה שעישון גורם לסרטן ריאות לקחה עשרות שנים, למרות קורלציה ברורה.

הבעיה: חברות הטבק טענו שזו "רק קורלציה" – אולי אנשים עם נטייה גנטית לסרטן גם נוטים לעשן?

ההוכחה הסיבתית דרשה:

1. **מחקרים אפידמיולוגיים רבים** – קורלציה עקבית בכל האוכלוסיות
2. **מחקרים פרוספקטיביים** – מעקב לאורך זמן: עישון $\rightarrow$ סרטן (ולא להפך)
3. **מנגנון ביולוגי** – זיהוי החומרים הקרצינוגניים בעשן
4. **מינון-תגובה** – יותר עישון $\rightarrow$ יותר סרטן
5. **הפסקת עישון** – מורידה את הסיכון

רק שילוב של כל אלה הוכיח **סיבתיות**.

קריטריונים של ברדפורד היל (Bradford Hill Criteria) (1965) **5691llih**:

טבלה 13: קריטריונים לזיהוי סיבתיות

קריטריון	הסבר
חוזק	קורלציה חזקה יותר מרמזת על סיבתיות
עקביות	התוצאות חוזרות במחקרים שונים
ספציפיות	החשיפה גורמת לתוצאה ספציפית
זמניות	הגורם קודם לתוצאה בזמן
גרדיאנט ביולוגי	מינון גבוה יותר $\rightarrow$ השפעה חזקה יותר
סבירות	קיים מנגנון ביולוגי סביר
קוהרנטיות	התוצאות עולות בקנה אחד עם הידע הקיים
ניסוי	התערבות משנה את התוצאה
אנלוגיה	קיימות תופעות דומות

4.15 כלים לזיהוי סיבתיות: Causal Inference

תחום ה**Causal Inference** מציע כלים מתמטיים לזיהוי סיבתיות מנתונים תצפיתיים.

שיטה 1 – ניסויים אקראיים מבוקרים (Randomized Controlled Trials - RCT):

זהו תקן הזהב. חלוקה אקראית לקבוצת ביקורת וטיפול מבטלת משתנים מבלבלים.

שיטה 2 – גרפים סיבתיים (Causal Graphs / DAGs):

פותחה על ידי Judea Pearl (זוכה פרס טיורינג 2011) **9002lraep**.

גרף סיבתי הוא גרף מכוון אציקלי (Directed Acyclic Graph - DAG) שבו:

- צמתים = משתנים
- חצים = יחסי סיבה-תוצאה

דוגמה – גלידה, טביעות, וחום:

טביעות מיסצריכת גלידה$[dr][dl]$חום הקיץ$[columnsep = large, rowsep = large]$

הגרף מראה ש"חום הקיץ" גורם לשניהם – אין קשר סיבתי ישיר בין גלידה לטביעות.

שיטה 3 – Instrumental Variables:

כאשר RCT לא אפשרי, ניתן להשתמש ב**משתנה אינסטרומנטלי** – משתנה שמשפיע על X אך לא ישירות על Y (רק דרך X).

דוגמה: מחקר השפעת השכלה על הכנסה. משתנה אינסטרומנטלי: מרחק מבית הספר התיכון הקרוב. משפיע על שנות לימוד (מי שגר רחוק נוטה ללמוד פחות), אך לא ישירות על ההכנסה.

4.16 קורלציה חלקית: בידוד השפעה

קורלציה חלקית (Partial Correlation) מודדת את הקשר בין X ו-Y **לאחר שליטה** על משתנה שלישי Z.

הגדרה 4.4 – קורלציה חלקית:

הקורלציה החלקית בין X ו-Y בשליטה על Z מוגדרת כ:

$$r_{XY\cdot Z} = \frac{r_{XY} - r_{XZ}\cdot r_{YZ}}{\sqrt{(1-r_{XZ}^2)(1-r_{YZ}^2)}} \tag{40}$$

פרשנות:

$r_{XY\cdot Z}$ היא הקורלציה בין X ו-Y **אחרי הוצאת ההשפעה של** Z.

דוגמה – גלידה וטביעות:

- $r_{\text{הדילג, תועיבט}} = 0.85$ (קורלציה גבוהה!)
- $r_{\text{הדילג, תועיבט}\cdot\text{סוח}} = 0.02$ (אחרי שליטה על חום – כמעט אפס!)

זה מוכיח שהקשר מזויף והוא נובע מהחום.

פסאודו-קוד – חישוב קורלציה חלקית:

תוצאה צפויה: הקורלציה הרגילה גבוהה (~ 0.7), אך הקורלציה החלקית כמעט אפס (~ 0.05).

4.17 קורלציה אפס ≠ בלתי תלויים

שאלה קריטית: אם $r = 0$, האם X ו-Y בלתי תלויים סטטיסטית?

תשובה: לא!

קורלציה אפס משמעה רק **אין קשר ליניארי**. עדיין יכול להיות קשר לא-ליניארי חזק.

דוגמה קלאסית – קשר ריבועי:

תוצאה: $r \approx 0$, אך $Y = X^2$ – תלות מושלמת!

הסיבה: קורלציה מודדת רק קשרים ליניאריים. $Y = X^2$ הוא קשר סימטרי (עבור X חיובי ושלילי), לכן הקורלציה הליניארית מתאפסת.

פתרון – מדדי תלות לא-פרמטריים:

טבלה 14: מדדי תלות: פרמטריים ולא-פרמטריים

מה הוא מודד	סוג	מדד
קשר ליניארי בלבד	פרמטרי	קורלציה של פירסון
קשר מונוטוני (לא בהכרח ליניארי)	לא-פרמטרי	קורלציה של ספירמן
קשר מונוטוני (עמיד יותר ל-sreiltuo)	לא-פרמטרי	קורלציה של קנדל
כל סוג תלות (כולל לא-מונוטונית)	לא-פרמטרי	Distance Correlation
כל סוג תלות (גם לא-דטרמיניסטית)	תאוריית מידע	Mutual Information

קורלציה של ספירמן (Spearman's Rank Correlation):

במקום הערכים עצמם, משתמשים ב**דירוגים** (Ranks):

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{41}$$

כאשר d_i הוא ההפרש בין דירוגי X ו-Y עבור תצפית i.

יתרון: מזהה קשרים מונוטוניים (עולים/יורדים) גם אם לא ליניאריים.

תוצאה צפויה: - עבור $Y = X^2$: פירסון ≈ 0, ספירמן ≈ 0 (לא מונוטוני) - עבור $Y = e^X$: פירסון ≈ 0.8, ספירמן $= 1.0$ (מונוטוני מושלם)

4.18 תרגיל תכנות עצמי 4.2 – גילוי קורלציות מזויפות

מטרה: להבין כיצד משתנה מבלבל יוצר קורלציה מזויפת.

משימה:

קורלציה חלקית ב-Python

```python
from scipy.stats import pearsonr

def partial_correlation(x, y, z):
    """
␣␣␣␣מחשב␣קורלציה␣חלקית␣r_{xy.z}

␣␣␣␣Args:
␣␣␣␣␣␣␣␣x,␣y,␣z:␣וקטורי␣נתונים

␣␣␣␣Returns:
␣␣␣␣␣␣␣␣float:␣קורלציה␣חלקית
␣␣␣␣"""
    # קורלציות זוגיות
    r_xy, _ = pearsonr(x, y)
    r_xz, _ = pearsonr(x, z)
    r_yz, _ = pearsonr(y, z)

    # נוסחת קורלציה חלקית
    numerator = r_xy - r_xz * r_yz
    denominator = np.sqrt((1 - r_xz**2) * (1 - r_yz**2))

    return numerator / denominator

# דוגמה: גלידה, טביעות, חום
np.random.seed(42)
n = 100

# חום הקיץ משתנה (מבלבל)
heat = np.random.randn(n)

# גלידה וטביעות תלויים בחום
ice_cream = 0.8 * heat + np.random.randn(n) * 0.2
drownings = 0.7 * heat + np.random.randn(n) * 0.3

# קורלציה רגילה
r_regular, _ = pearsonr(ice_cream, drownings)
print(f"קורלציה␣רגילה:␣{r_regular:.3f}")

# קורלציה חלקית (שליטה על חום)
r_partial = partial_correlation(ice_cream, drownings, heat)
print(f"קורלציה␣חלקית␣שליטה(␣על␣חום):␣{r_partial:.3f}")

# התוצאה: הקורלציה נעלמת!
```

קורלציה אפס עם תלות מושלמת

```
import matplotlib.pyplot as plt

# יעוביררשקתריצי
x = np.linspace(-3, 3, 100)
y = x**2

# היצלרוקבושיח
r, _ = pearsonr(x, y)

print(f"היצלרוק:␣{r:.6f}")
print("אבל␣Y␣תלוי␣לחלוטין␣ב-X!")

# רויצ
plt.figure(figsize=(8, 6))
plt.scatter(x, y, alpha=0.6)
plt.xlabel('X')
plt.ylabel('Y␣=␣X²')
plt.title(f'Perfect␣Dependency,␣Zero␣Correlation␣(r␣=␣{r:.3f})')
plt.grid(True, alpha=0.3)
plt.show()
```

1. צרו סימולציה של "גלידה, טביעות, חום"
2. חשבו קורלציה רגילה וקורלציה חלקית
3. הציגו גרפית איך הקורלציה נעלמת לאחר שליטה
4. נסו עם משתנים מבלבלים שונים

תוצאה צפויה: הגרף השמאלי מראה קורלציה חזקה, אך הגרף הימני (אחרי שליטה) מראה שאין קשר אמיתי.

4.19 סיכום ומבט קדימה

מה למדנו בפרק זה?

1. **קו-ווריאנס היא מכפלה סקלרית** של וקטורים ממורכזים – הרחבה של מושגי פרק 1
2. **קורלציה היא Cosine Similarity** של וקטורים ממורכזים – גרסה מנורמלת ללא תלות ביחידות
3. **תחום הקורלציה:** $[-1, 1]$ – הוכחנו באמצעות אי-שוויון קושי-שוורץ
4. **מתאם אינו סיבתיות!** – משתנים מבלבלים יוצרים קורלציות מזויפות

השוואת קורלציית פירסון וספירמן

```python
from scipy.stats import spearmanr

# קשרריבועי
x = np.linspace(-3, 3, 100)
y = x**2

# פירסון
r_pearson, _ = pearsonr(x, y)

# ספירמן
r_spearman, _ = spearmanr(x, y)

print(f"Pearson:␣{r_pearson:.4f}")
print(f"Spearman:␣{r_spearman:.4f}")

# קשראקספוננציאלימונוטוני   ()
x_exp = np.linspace(0, 5, 100)
y_exp = np.exp(x_exp)

r_pearson_exp, _ = pearsonr(x_exp, y_exp)
r_spearman_exp, _ = spearmanr(x_exp, y_exp)

print(f"\nExponential␣relationship:")
print(f"Pearson:␣{r_pearson_exp:.4f}")
print(f"Spearman:␣{r_spearman_exp:.4f}")
```

סימולציה: גילוי קורלציות מזויפות

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# פרמטרים
np.random.seed(42)
n = 200

# משתנהמבלבל : חוםהקיץ
temperature = np.random.uniform(15, 35, n)  # טמפרטורהבצלזיוס

# גלידהתלויהבחום   + רעש
ice_cream = 2 * temperature + np.random.randn(n) * 5

# טביעותתלויותבחום   + רעש
drownings = 1.5 * temperature + np.random.randn(n) * 3

# קורלציות
r_naive, _ = pearsonr(ice_cream, drownings)
r_partial = partial_correlation(ice_cream, drownings, temperature)

print(f"קורלציה␣נאיבית␣ללא␣(שליטה):␣{r_naive:.3f}")
print(f"קורלציה␣חלקית␣עם␣(שליטה␣על␣חום):␣{r_partial:.3f}")

# ויזואליזציה
fig = plt.figure(figsize=(15, 5))

# גרף 1: קורלציהנאיבית
ax1 = fig.add_subplot(131)
ax1.scatter(ice_cream, drownings, c=temperature, cmap='hot', alpha=0.6)
ax1.set_xlabel('Ice␣Cream␣Sales')
ax1.set_ylabel('Drownings')
ax1.set_title(f'Naive␣Correlation:␣r␣=␣{r_naive:.3f}')
ax1.grid(True, alpha=0.3)

# גרף 2: 3D - רואיםאתהמבלבל
ax2 = fig.add_subplot(132, projection='3d')
ax2.scatter(temperature, ice_cream, drownings, c=temperature, cmap='hot'
, alpha=0.6)
ax2.set_xlabel('Temperature')
ax2.set_ylabel('Ice␣Cream')
ax2.set_zlabel('Drownings')
ax2.set_title('True␣Relationship␣(3D)')

# גרף 3: לאחרהסרתהמבלבלשאריות   ()
from sklearn.linear_model import LinearRegression
model_ice = LinearRegression().fit(temperature.reshape(-1, 1), ice_cream
```



```
model_drown = LinearRegression().fit(temperature.reshape(-1, 1),
drownings)
```

5. **קורלציה חלקית** – כלי לבידוד השפעה ושליטה על משתנים מבלבלים

6. **קורלציה אפס ≠ בלתי תלויים** – קיימים קשרים לא-ליניאריים

7. **מדדים לא-פרמטריים** – ספירמן, קנדל, Mutual Information למצבים מורכבים

מבט קדימה – פרק 5:
בפרק הבא נעבור מיחסים בין משתנים ל**אופטימיזציה**. נחקור:

- **רגרסיה ליניארית כבעיית אופטימיזציה** – מזעור פונקציית ה-ESM

- **פתרון אנליטי** – נוסחת ה-noitauqE lamroN

- **פתרון איטרטיבי** – Gradient Descent וגרסאות שלו

- **התכנסות ויציבות** – מתי האלגוריתם מצליח?

- **קשר למכפלה סקלרית** – גרדיאנט כווקטור אורתוגונלי

שאלת מחשבה לסיום:
אם X ו-Y בעלי קורלציה $r = 0.9$, ו-Y ו-Z בעלי קורלציה $r = 0.9$, האם בהכרח X ו-Z יהיו בעלי קורלציה גבוהה?
רמז: לא! קורלציה אינה טרנזיטיבית. אפשר לבנות דוגמה נגדית שבה $r_{XZ} = 0$.

מטלות וקריאה מורחבת

תרגיל 4.1: חשבו קו-וריאנס וקורלציה ידנית (ראו פסאודו-קוד).
תרגיל 4.2: סמלצו קורלציה מזויפת עם משתנה מבלבל (ראו פסאודו-קוד).
תרגיל 4.3: הוכיחו שמטריצת קורלציה היא תמיד חיובית חצי-מוגדרת.
רמז: השתמשו בעובדה שלכל וקטור $\vec{v}$, מתקיים $\vec{v}^T\mathbf{R}\vec{v} \geq 0$.
קריאה מורחבת:

- [24] – "Mathematical Contributions to the Theory of Evolution": המאמר המקורי על קורלציה

- **5691llih** – "The Environment and Disease: Association or Causation?": הקריטריונים המפורסמים לסיבתיות

- **9002lraep** – "Causality: Models, Reasoning, and Inference": הספר המקיף על Causal Inference

- **5102negiv** – "Spurious Correlations": אוסף משעשע של קורלציות מזויפות

שאלות להעמקה:

1. מדוע קורלציה של ספירמן עמידה יותר ל-sreiltuo מקורלציה של פירסון?

2. האם שתי קורלציות חלקיות $r_{XY\cdot Z}$ ו-$r_{XY\cdot W}$ יכולות לתת תוצאות שונות?

3. כיצד Mutual Information יכולה לזהות תלות שקורלציה מפספסת?

סיום פרק 4

5 English References

1 G. Strang, *Linear Algebra and Learning from Data*. Wellesley, MA, USA: Wellesley-Cambridge Press, 2019.

2 T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

3 Z. S. Harris, “Distributional structure,” *Word*, vol. 10, no. 2-3, 146–162, 1954.

4 A. Caliskan, J. J. Bryson, and A. Narayanan, “Semantics derived automatically from language corpora contain human-like biases,” 6334, 356, 2017, 183–186.

5 T. Bolukbasi, K.-W. Chang, J. Zou, V. Saligrama, and A. Kalai, “Man is to computer programmer as woman is to homemaker? debiasing word embeddings,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

6 R. E. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1957.

7 Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” 11, 86, 1998, 2278–2324.

8 A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 25, 2012.

9 I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

10 K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is ”nearest neighbor” meaningful?” *Lecture Notes in Computer Science*, vol. 1540, 217–235, 1999.

11 C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” *Lecture Notes in Computer Science*, vol. 1973, 420–434, 2001.

12 E. Fix and J. L. Hodges, “Discriminatory analysis: Nonparametric discrimination: Consistency properties,” *Technical Report, USAF School of Aviation Medicine*, 1951.

13 V. N. Vapnik and A. Y. Chervonenkis, “On the uniform convergence of relative frequencies of events to their probabilities,” *Theory of Probability and Its Applications*, vol. 16, no. 2, 264–280, 1971.

14 J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, 248–255.

15 Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, 541–551, 1989.

16 K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 11, 559–572, 1901.

17 H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, 417–441, 1933.

18 G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, 403–420, 1970.

19 A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, 55–67, 1970.

20 R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B*, vol. 58, no. 1, 267–288, 1996.

21 A. Chaman and I. Dokmanic, "Truly generative or just extrapolation? on the ability of generative models to create truly novel content," *arXiv preprint arXiv:2109.09018*, 2021.

22 T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd. New York, NY, USA: Springer, 2009.

23 F. Galton, "Regression towards mediocrity in hereditary stature," *Journal of the Anthropological Institute of Great Britain and Ireland*, vol. 15, 246–263, 1886.

24 K. Pearson, "Mathematical contributions to the theory of evolution. iii. regression, heredity, and panmixia," *Philosophical Transactions of the Royal Society of London. Series A*, vol. 187, 253–318, 1896.

25 R. A. Fisher, *Statistical Methods for Research Workers*. Edinburgh, UK: Oliver and Boyd, 1925.

26 S. Wright, "Correlation and causation," *Journal of Agricultural Research*, vol. 20, no. 7, 557–585, 1921.