

1. Introduction

1.1 What is Machine Learning?

1.1.1 The Basic Concept

Artificial Intelligence (AI)

בינה מלאכותית היא נתחם מחשב או מנגנון טכנולוגי אחר מחקה מנגן חישבה אנוש. בתחום רחב זה יש רמות שונות של בינה מלאכותית – יש מערכות שמסוגלות ללמידה דפואית התנהגות ולהתאים את עצמן לשינויים, ואילו יש מערכות שאמנים מנגנון חישבה אנושי אך הן לא מתוחכמתות מעבר لما שתכננו אותן בהתחלה. שואב רוחוט הידוע לחשב את גודל החדר ואת מסלול הינוקי האופטימלי פועל לפי פרוצדורה ידועה מראש, ואין בו תחכם מעבר לתכנות הראשוני שלו. לעומת זאת תוכנה היודעת לסנן רעשים באופן מסתגל, או להמליץ על Shirims בגין מזיקה בהתאם לסוגן של המש坦ש, משתמשת במבנה מלאכותית ברמה גבוהה יותר, כיוון שהן לומדות עם הזמן דברים חדשים.

המונה בינה מלאכותית מתייחס בדרך כלל למערכת שמחקה התנהגות אנושית, אך היא שగרתית, לא לומדת ממשו חדש, ועולה את אותו הדבר כל הזמן. מערכת זו יכולה להיות משוכלת ולחשב דברים מסוימים ואף להסיק מסקנות על דוגמאות חדשות שהיא מעולמת לא ראתה, אך תמיד עברו אותה הקלט (Input), יהיה אותה הפלט (Output).

נראה לדוגמא מערכת סטרימינג של סרטיים, למשל Netflix. חלק משיפור המערכת והגדלת זמני הצפייה, ניתן לבנות מנגנון המלצות הבני על ההיסטוריה השימוש של הלוקחות של המערכת – איזה סרטיים הם רואים, איזה צ'אנרים ומתי. כדי שמש צופים ומעט סרטיים, ניתן לעשות זאת באמצעות ידע – למלא טבלאות של הנתונים, לנתח אותם ידנית ולבנות מערכת חוקים שמהווה מנוע המלצות מבוסס AI. נראה לדוגמא אדם שצופה ב"פארק היורה" וב"אינדיינה ג'ונס" – סביר שהמערכת תמליץ לו לצפות גם ב- "פולטרגיסט". אדם שצופה לעומת זאת ב- "אהבה בין הכרמים" ו"הבית על האגם", ככל הנראה כדי להמליץ לו על "הגשים של מחוז מדיסון".

מערכת זו יכולה לעבוד טוב, אך במקרה מסוימת כבר לא ניתן לנוכח אותה כפרוצדורה מسودרת וכוסף של חוקים ידוע מראש. מאגר הסרטים גדול, נוספים סוגים נוספים של סרטים (כמו למשל סדרות, תוכניות ריאליטי ועוד) ובנוסף רוצים להתייחס לpermeters נוספים – האם הצופה ראה את כל הסרט או הפסיק במאצל, מה גיל הצופה ועוד. מערכת הבניה באופן קלאסי אינה מסוגלת להתמודד עם כמיות המידע הקיימות, וכמויות הכללים שנדרש לחושב עליהם מראש היא עצומה ומורכבת לחישוב.

נתבונן על דוגמא נוספת – מערכת לניניות רכב. ניתן להגיד כלל פשוט בו אם משתמש יצא מטל איביב ורוצה להגיע לפתח תקווה, אז האפליקציה תיקח אותו דרך מסלול ספציפי שנבחר מראש. מסלול זה לא מתחשב בpermeters קרייטיים כמו מה השעה, האם יש פקקים או חסימות ועוד. כמויות הpermeters שיש להתייחס אליהם איננה ניתנת לטיפול על ידי מערכת כללים ידועה מראש, וגם הפונקציונליות המתאפשרת היא מוגבלת מאוד – למשל לא ניתן לחזות מה תהיה שעת ההגעה וכדומה.

Machine Learning (ML)

למייד מוכנה הוא תחום בינה מלאכותית, הבא להתמודד עם שני האתגרים שתוארו קודם – יכולת לתוכנת מערכת על בסיס מסוות של נתונים וpermeters, וחיזוי דברים חדשים כתלות בpermeters רבים שאינם להשנות עם הזמן. מנגנוני ML מנהנים כמיות אידיות של DATA ומנשות להציג לagency לאיזו תוכאה. אם מדובר באפליקציה ניוט, המערכת תנתח את כל אוטם הפקטורים ותנסה לחשב את מרחק הנסיעה המשוער. נניח והיא חזתה 20 דקות נסעה. אם בסופו של דבר הנסעה ארוכה 30 דקות, האלגוריתם ינסה להבין פקטור השתנה במהלך הדרך ומדווח הוא נכשל בחיזוי (למשל: הcabesh בין ארבעה נתבים, אבל במקטע מסוים הוא מצטמצם לאחד וזה מייצר עיכוב, וזה עיכוב קבוע ברוב שעות היום ולא פקק אקראי). בהינתן מספיק מקרים אלה, האלגוריתם "ambil" שהוא טועה, והוא פשוט יתקן את עצמו ויכניס למרכז החישובים גם פקטור של מספר נתבים יורייד אויל את המשקל של הטמפרטורה בחו"ל. וכך באפין חזרתי האלגוריתם שוב ושוב מקבל קלט, מוציא פלט ובודק את התוצאה הסופית. לאחר מכן הוא בודק היכן הוא טעה, משנה את עצמו, מתקן את המשקל שהוא נותן לפקטורים שונים ומשתכל מנסעה לנסעה.

במערכות אלה הקלט נשאר לכוארה קבוע, אבל הפלט משתנה – עבור זמני יציאה שונים, האלגוריתם יעריך זמני נסעה שונים, כתלות במגון permeters הרלוונטיים.

מערכות ML משמשות את כל רשות הpermeters הגדולים. כל אחת מנסה בדרכה שלה לחזות למשל, איזה משתמש שהקליק על המודעה צפי שיבצע ריכשה. הפלטpermeters השונות מננות לזיהות כוונה (Intent) על ידי למידה

מניסיון. בהתחלה הן פשוט ניחשו על פי כמה פקטוריים שהזemoו להם על ידי בני אדם. נניח, גוגל החלטה שמי שצופה בסרטוני יוטיוב של *Unboxing* הוא הבומרן Intent��値. במשמעותו של רכישה. בהמשך הדריך, בהנחה והמשתמש מבצע רכישה כלשהו, האלגוריתם מקבל "נקודה טוביה". אם הוא לא קנה, האלגוריתם מקבל "נקודה רעה". ככל שהוא מקבל יותר נקודות טובות ורעות, האלגוריתם יודע לשפר את עצמו, לחתך משקל גדול יותר לpermeters טוביים ולהזניח permeters פחות משמעותיים. אבל רגע, מי אמר למערכת להסתכל בכל בסרטוני *Unboxing*?

האמת שהיא שאף אחד. מישחו, בנאדים, אמר למערכת לזהות את כל הסרטונים שימוש צופה בהם ביווטיב, לזהות מתוך הסרטון, האודיו, תיאור הסרטון ומילוט המפתח וכו' – איזה סוג סרטון זה. יתרון שאחרי מיילארדי ציפוי הסרטונים, האלגוריתם מתייחס למצואו קשר בין סוג מסוים של סרטונים לבין פעולות כמו רכישה באתר. באופן זה, גוגל מזינה את האלגוריתם בכל הפעולות שהמשתמש מבצע. המיללים שהוא קורא, המיקומות שהוא מסתובב בהם, התמונות שהוא מעלה לענן, ההודעות שהוא שולח, כל מידע שיש אליו גישה. הכל נשפר תוך מאגר הנתונים העוזם בו מוסה גוגל לבנות פרופילים ולמצוא קשר בין הסיכוי שלו לרכוש או כל פעולה אחרת שבאה לה לזהות.

המכונה המופלאה זו לומדת כל הזמן החדש ומנסה כל הזמן למצוא הקשיים, להזנת תוצאה, לבדוק אם היא הצליחה, ואם לא לתקן את עצמה שוב ושוב עד שהיא פוגעת במטרה. חשוב לציין שלמכונה אין סנטימנטים, כל המידע קביל ואם היא תמצא קשר מוכח בין מידת הנעלים של בנאדים לבין סרטונים של בייבי שארק, אז היא משתמש בו גם אם זה לא נשמע הגיוני.

חשוב לשים לב לעניין המטרה – המטרה היא לא המצאה של האלגוריתם. הוא לא קם בבודוק ומחייב מה האפליקציה שלכם צריכה לעשות. המטרה מוגדרת על ידי היוצר של המערכת. למשל – חישוב זמן נסעה, בניית מסלול אופטימי בין A ל-B וכו'. המטרה של גוגל – שימוש יבצע רכישה, והכל מתנקז לזה בסוף, כי גוגל בראש ובראשונה היא מערכת פרסום. אגב, גם ההגדרה של מסלול "אופטימי" היא מעשה ידי אדם. המכונה לא יודעת מה זה אופטימי, זו רק מילה. אז צריך לעזרה לה ולהגיד לה שאופטימי זה מינימום זמן, מעט עצירות, כמה שפוחות רמזורים וכו'. לסיכום, המטרה מאופיינת על ידי האדם ולא על ידי המכונה. המכונה רק חותרת למטרה שהוגדר לה.

יש לנו ML המבוססים על דата מסודר ומתויג כמו Netflix, עם כל המאפיינים של הסרטים אבל גם עם המאפיינים של הצופים (מדינה, גיל, שעת צפייה וכו'). לעומת זאת יש לנו ML שמקבלים טיפה יותר חופש וمتבוססים על מידע חלק מאד (יש להם מידע על כל הסרטים, אבל אין להם מידע על הצופה). מנגנונים אלו לא בהכרח מנסים לבנות מנוע המלצות אלא מנסים למצוא חוקיות בנתונים, חריגות וכו'.

כך או כך, המערכת הסובב זהה הקורי ML בנייני אלגוריתמים שונים המيونנים בניתוח טקסט, אלגוריתמים אחרים המתמקדים בעיבוד אודיו,أكلה המנתחים היסטורית גליישה או זיהוי מטור דף Web בו אtmp צופים ועוד. عشرות או מאות מנגנונים כאלה מסתובבים ורצים ובונים את המפה השלמה. ככה רוב רשותם הפרסום הגדולה עובדות. ככל שהמכונה של גוגל/פייסבוק תהיה חכמה יותר, ככה היא תדע להציג את המודעה המתאימה למשתמש הנכוון, בזמן הנכוון ועל ה-device המתאים.

1.1.2 Data, Tasks and Learning

כאמור, המטרה הבסיסית של למידת מכונה היא יכולת להכליל מטור הניסיון, ולבצע שימושים באופן מדויק ככל הנitin על דטה חדש שעדיין לא נצפה, על בסיס צבירת ניסיון מדטה קיים. באופן כללי ניתן לדבר על שלושה סוגים של למידה:

למידה מונחית (supervised learning) – הדטה הקיים הינו אוסף של דוגמאות, וכל דוגמא יש תווית (label). מטרת האלגוריתמים במרקחה זה היא לסייע דוגמאות חדשות שלא נצפו בתהילך הלמידה. באופן פורמלי, עבור דטה $\mathbb{R}^{n \times d} \in x$, יש אוסף $\mathbb{R}^{1 \times d}$ – labels $y \in u$, ומ Chapman את האלגוריתם שמבצע את המיפוי $y \rightarrow X: g$: בצורה הטובה ביותר, כתוב בהינתן דוגמא חדשה $x \in \mathbb{R}^n$, המטרה היא למצוא עבורה את ה- u הנכוון. המיפוי נמדד ביחס לפונקציות מחייר, כדי שיווסף בהמשך בוגר לתהילך הלמידה.

למידה לא מונחית (unsupervised learning) – הדטה הקיים הינו אוסף של דוגמאות מרחב, בלי שננתן עליהם מידע כלשהו המבחן ביניהם. במקרה זה, בדרך כלל האלגוריתמים יחפשו מודל המסביר את התפלגות הנקודות – למשל חלוקה לקבוצות שונות וצדומה.

למידה באמצעות חיזוקים (reinforcement learning) – הדטה בו נעדרים איהם מצוי בתחום התוכנית אלא נאוסף עם הזמן. ישנו סוכנים הנמצאים בסביבה מסוימת ומעבירים מידע למשתמש, והוא בתורו למד אסטרטגייה בה הסוכנים ינקטו בצעדים הטוביים עבורה.

האלגוריתמים השונים של הלמידה מתחלקים לשתי קבוצות – מודלים דיסקרימנטיביים המוצאים פלט על בסיס מידע נתון, אך לא יכולים ליצור מידע חדש בעצם, ומודלים גנרטיביים, שלא רק לומדים להכלי את הדעתה הנלמד גם עבור דוגמאות חדשות, אלא יכולים גם להבין את מה שהם רואו וליצור מידע חדש על בסיס הדוגמאות שנלמדו.

כאמור, בשביל לבנות מודל יש צורך בדעתה. מודל טוב הוא מודל שמציל להכלי את הדעתה הקיימם גם לדעתה חדשה. המודל למשה מנסה למציא דפוסים בדעתה הקיימם, מהם הוא יכול להסיק מסקנות גם על דוגמאות חדשות. כדי לוודא שהמודל אכן מציל להכלי גם על דוגמאות חדשות, בדרך כלל מחלקים את הדעתה הקיימם לשניים – קבוצת אימון (training set) וקבוצת מבחן (test set). סט האימון מאפשר למדד לצלחת המודל – אם המודל מציל למציא דפוסים בסט האימון שניכנים גם עבור סט המבחן, זה סימן שהמודל הצליח למציא כלים שיכולים להיות ניכנים גם לדוגמאות חדשות שיבאו. לעיתים סט האימון מוחולקת בעצמה לשניים – קבוצת דוגמאות עליה המודל מתאמן, וקבוצת ולידציה (validation set) המשמשת להימנע מ-*overfitting* (המשמעות להימנע מ-*validation set*) שיפור בהמשך.

מגון התחומיים בהם משתמשים בכלים של למידה הוא עצום, עד כדי כך שכמעט אין תחום בו לא נכנס השימוש באלגוריתמים לומדים. דוגמאות בולטות למשימות באלגוריתמים לומדים: סיוג, רגסיה (מציאת קשר בין משתנים), חלוקה לקבוצות, מערכת המלצות, הורדת ממד, ראייה ממוחשבת, עיבוד שפה טבעיות ועוד.

1.2 Applied Math

האלגוריתמים של למידת מכונה נסמכים בעיקר על שלושה ענפים מתמטיים; אלגברה לינארית, חישוב דיפרנציאלי והסתברות. פרק זה נציג את העקרונות הנדרשים בלבד, ללא הרחבה, על מנת להבין את הנושאים הנדרשים בספר זה.

1.2.1 Linear Algebra

וקטורים ומרחבים וקטוריים

באופן מתמטי מופשט, וקטורים, המנסונים בדרך כלל ע"י \vec{x} או על ידי x , הינם אובייקטים הנמצאים במרחב וקטורי $(+, \cdot)$ מעל שדה \mathbb{F} . מהו אותו מרחב וקטורי?

ראשית, השדה, \mathbb{F} , הוא קבוצת מספרים המקיימים תכונות מתמטיות מסוימות. לדין בספר זה, השדה הוא קבוצת המספרים המשיים – \mathbb{R} , או קבוצת המספרים המרוכבים – \mathbb{C} . שנית, נשים לב כי המרחב הווקטורי דורש גם הגדרת פעולה חיבור $(+)$.

כעת, $(+, \cdot)$ היא מרחב וקטורי אם הוא מקיים את התכונות הבאות:

- (I) קיימ איבר אפס (וקטור אפס) כך שכל \vec{x} בקבוצה V מקיים: $\vec{x} = \vec{x} + \vec{0} = \vec{x}$.
- (II) לכל איבר בשדה a ולכל \vec{x} ו- \vec{y} בקבוצה V , גם $\vec{y} + \vec{x} \cdot a$ הינו איבר בקבוצה V .

הערה: קיימות דרישות נוספות למרחב וקטורי, אך הם מעבר לנדרש בספר זה.

דוגמאות:

A. וקטורים גאומטריים:
מערך חד ממדי (x_1, x_2, \dots, x_n) נקרא וקטור גאומטרי a ממד', כאשר רכיבי הווקטור הם איברים בשדה \mathbb{F} . האיבר x_i המיצג על ידי האינדקס i מציין את מיקום האיבר. מרחב זה מסומן ע"י \mathbb{F}^n .
נראה שמרחב זה הוא אכן מרחב וקטורי:

חיבור וקטוריים:

$$\vec{x} = (x_1, x_2, \dots, x_n), \quad \vec{y} = (y_1, y_2, \dots, y_n) \rightarrow \vec{x} + \vec{y} = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n)$$

וקטור אפס:

$$\vec{0} = (0, 0, \dots, 0)$$

כפל בסקלר:

$$\vec{x} = (x_1, x_2, \dots, x_n) \rightarrow a \vec{x} = (a x_1, a x_2, \dots, a x_n)$$

הערה: לשם פשוטות, בהמשך, נenna וקטור גאומטרי כ"וקטור" בלבד.

B. מטריצות:

מערך זו מגדיר $\begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nm} \end{pmatrix}$, אשר רכיביו הם איברים בשדה \mathbb{F} , נקרא מטריצה מסדר $m \times n$, כאשר n הוא מספר השורות ו- m הוא מספר העמודות במערך. האיברים במטריצה A_{ij} מיוצגים ע"י שני אינדקסים – i, j , המתארים את השורה והעמודה בהתאם. מרכיב זה מסומן בדרך כלל על ידי $\mathbb{F}^{n \times m}$.
וכlich שמרחב זה הוא אכן מרחב וקטורי:

חיבור מטריצות:

$$\hat{A} = \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nm} \end{pmatrix}, \hat{B} = \begin{pmatrix} B_{11} & \dots & B_{1m} \\ \vdots & \ddots & \vdots \\ B_{n1} & \dots & B_{nm} \end{pmatrix} \rightarrow \hat{A} + \hat{B} = \begin{pmatrix} A_{11} + B_{11} & \dots & A_{1m} + B_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} + B_{n1} & \dots & A_{nm} + B_{nm} \end{pmatrix}$$

מטריצת אפס:

$$\hat{0} = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{pmatrix}$$

כפל בסקלר:

$$\hat{A} = \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nm} \end{pmatrix} \rightarrow a \hat{A} = \begin{pmatrix} a A_{11} & \dots & a A_{1m} \\ \vdots & \ddots & \vdots \\ a A_{n1} & \dots & a A_{nm} \end{pmatrix}$$

ניתן לבדוק כי הווקטורים הגיאומטריים שהוגדרו בדוגמה א', הם בעצם מטריצות במד $1 \times n$.

ג. פולינומים:

פולינומים מסדר n הינם ביוטיים מהסוג $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$, כאשר n מייצג את החזקה הגדולה ביותר ו- a_i הם איברים בשדה. מרכיב זה מסומן בדרך כלל על ידי $P_n(x)$.

בכל הדוגמאות לעיל קל לראות שהן אכן מהוות מרחב וקטורי. רשימה חלקית לדוגמאות נוספות לווקטורים (ולמרחבים וקטוריים) כוללת למשל מרחבי פונקציות או אפילו אוטות אלקטромגנטיים. כאן בחרנו רק את הדוגמאות הרלוונטיות למספר זה.

פעולות חשבון על מטריצות וקטורים:

כמו שנצכר לעיל, הווקטורים הגיאומטריים שהוגדרו בדוגמה א', הם בעצם מטריצות במד $1 \times n$. לכן, פעולות החשבון מוגדרות באופן זהה.

• חיבור וחיסור בין שתי מטריצות:

$\mathbb{F}^{n \times m} \in \hat{A}, \hat{B}, \hat{A}_{ij}, \hat{B}_{ij}$ הם האיברים בשורה i בעמודה j של המטריצות \hat{A}, \hat{B} בהתאם. אז, האיבר בשורה i בעמודה j של מטריצת הסכום (או ההפרש) הינו

$$(A \pm B)_{ij} = A_{ij} \pm B_{ij}$$

(הגדרת חיבור המטריצות בעצם כבר ניתנה בדוגמה א' לעיל).

שים לב: ניתן לחסר ולחסור מטריצות רק בעלות אותו הממד.

• כפל בין שתי מטריצות:

$\mathbb{F}^{k \times m} \in \hat{A}, \mathbb{F}^{m \times k} \in \hat{B}$ הן שתי מטריצות, כאשר מסדר העמודות במטריצה \hat{A} שווה למספר השורות של מטריצה \hat{B} (אך שתי המטריצות אינן בהכרח בעלות אותן ממד). במקרה זה, מכפלת המטריצות מוגדרת על ידי:

$$\hat{A} \cdot \hat{B} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} + \dots + A_{1k}B_{k1} & \dots & A_{11}B_{1n} + \dots + A_{1k}B_{kn} \\ \vdots & \ddots & \vdots \\ A_{m1}B_{11} + \dots + A_{mk}B_{k1} & \dots & A_{m1}B_{1n} + \dots + A_{nk}B_{kn} \end{pmatrix}$$

למעשה כל איבר בתוצאה הינו סכום של מכפלת שורה i ממטריצה A בעמודה j ממטריצה B :

$$(\hat{A} \cdot \hat{B})_{ij} = \sum_r A_{ir} B_{rj}$$

שים לב: על מנת שכפל המטריצות יהיה מוגדר מספר העמודות ב- \hat{A} שווה למספר השורות ב- \hat{B} .
 $\hat{A}\hat{B} \neq \hat{B}\hat{A}$ גם הכפל $\hat{B}\hat{A}$ אולם יתכן ש- $\hat{A}\hat{B}$.

- **שחלוף (transpose)**:

החלפת שורות בעמודות, או 'סיבוב' המטריצה. נניח מטריצה $\hat{A} \in \mathbb{F}^{n \times m}$, אז השחלוף שלה, המסומן \hat{A}^T הוא:

$$(\hat{A}^T)_{ij} = A_{ji}$$

ובאופן מפורש:

$$\hat{A} = \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nm} \end{pmatrix} \rightarrow \hat{A}^T = \begin{pmatrix} A_{11} & \dots & A_{n1} \\ \vdots & \ddots & \vdots \\ A_{1m} & \dots & A_{nm} \end{pmatrix}$$

שים לב שהמטריצה החדשה \hat{A}^T הינה בממ"ד $n \times m$. בנוסף ניתן להוכיח כי מתקיימ:

שחלוף של וקטור שורה, נותן וקטור עמודה ולהפך.

- **מטריצת יחידה:**

מטריצת יחידה, הינה מטריצה ריבועית (מסדר $n \times n$), המסומנת על ידי \mathbb{I}_n ומוגדרת כך שכל איבריה אפס מלבד איברי האלכסון הראשי המקבלים את הערך 1:

$$(\mathbb{I}_n)_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

ובאופן מפורש:

$$\mathbb{I}_n = \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{pmatrix}$$

מטריצת זו מקיימת $\hat{A} = \hat{A} \cdot \mathbb{I}_n = \mathbb{I}_m \cdot \hat{A}$ לכל מטריצה \hat{A} מסדר $m \times n$.
הערה: לעיתים סדר מטריצת היחידה אינו משנה או טריויאלי, ולכן המטריצה מסומנת רק על ידי \mathbb{I} ללא ציון הממד.

- **מטריצת הופכית:**

למטריצות ריבועיות (מטריצות עם מספר זהה של שורות ועמודות; מסדר $n \times n$) יתכן שיש מטריצה הופכית⁻¹ שמקיימת את הקשר:

$$\hat{A} \cdot \hat{A}^{-1} = \hat{A}^{-1} \cdot \hat{A} = \mathbb{I}_n$$

דוגמא: $\hat{A} = \hat{A}^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. לכן, במקרה זה $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathbb{I}_2$

- **מטריצת צמודה/hermitian:**

עבור מטריצה A , המטריצה $A^* = A^\dagger$ נקראת הצמוד הרמייטי של A , ומתקיימ:

$$(A^*)_{ij} = \overline{A_{ji}}$$

הצמוד הרמייטי הוא שחלוף של A , כאשר לכל איבר במטריצה המשוחלפת לוקחים את הצמוד המרוכב.
אם A מטריצה ממשית, המטריצה הצמודה שלה היא למעשה המטריצה המשוחלפת של A .

• מטריצה אוניטרית:

מטריצה אוניטרית היא מטריצה ריבועית מעל המספרים המרוכבים המקיימת את התנאי:

$$A^* A = AA^* = \mathbb{I}$$

מערכת משוואות לינאריות:

מערכת משוואות לינאריות מוצגת באופן כללי באופן הבא:

$$\begin{array}{ccccccccc} A_{11}x_1 + A_{12}x_2 + & \dots & + A_{1n}x_n & = & b_1 \\ \vdots & \ddots & \vdots & & \vdots \\ A_{m1}x_1 + A_{m2}x_2 + & \dots & + A_{mn}x_n & = & b_m \end{array}$$

נשים לב כי מערכת משוואות לינארית ניתנת לייצוג באופן קומפקטי על ידי הפרדה בין רשימת המשתנים, המקדמים של משתנה, והאיבר החופשי, באופן הבא:

$$\hat{A} \vec{x} = \vec{b} = \begin{pmatrix} A_{11} & \dots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \dots & A_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

מטריצה \hat{A} הינה מטריצת המקדמים מסדר $m \times n$, כאשר n הוא מספר המשתנים, ו- m הוא מספר המשוואות במערכת.

קטגור \vec{x} , הינו וקטור عمودה (לעתים גם מסומן על ידי $(x_1, x_2, \dots, x_n)^T$), המיצג את וקטורי המשתנים.

קטגור \vec{b} , הינו וקטור عمודה, שאיבריו הם האיבר החופשי.

פתרונות של מערכת המשוואות הלינארית, $\vec{b} = \hat{A} \vec{x}$, אם הם קיימים ויחדים, נתונים ע"י $\vec{b} = \hat{A}^{-1} \vec{b}$.

מכפלה פנימית, נורמה, אורטוגונליות

מרחב מכפלה פנימית, מוגדר על ידי מרחב וקטורי V (המוגדר על גבי שדה \mathbb{F}) ועל ידי פעולה "מכפלה פנימית". מכפלה פנימית, הנקראת לעתים רק מכפלה, הינה בעצם פונקציה המתקבלת שני וקטורים מרחב וקטורי V ומחזירה סקלר (=מספר) בשדה \mathbb{F} . מכפלה זו, מסומנת בדרך כלל ע"י $\langle \cdot, \cdot \rangle$ (או ע"י $\mathbb{F} \rightarrow V \times V : \langle \cdot, \cdot \rangle$), חיבת לkiem מספר תכונות.

לכל $v \in V$ ו- \vec{w}, \vec{v} (כל שלושה וקטורים למרחב הווקטורי V), ולכל $\lambda \in \mathbb{F}$ (סקלר בשדה \mathbb{F}):

- $\langle \vec{v} + \vec{w}, \vec{u} \rangle = \langle \vec{v}, \vec{u} \rangle + \langle \vec{w}, \vec{u} \rangle$
- $\langle \lambda \vec{v}, \vec{u} \rangle = \lambda \langle \vec{v}, \vec{u} \rangle$
- $\overline{\langle \vec{v}, \vec{u} \rangle} = \langle \vec{u}, \vec{v} \rangle$
- $\langle \vec{v}, \vec{v} \rangle \geq 0$

ההגדרה עצמה של המכפלה משתנה כתלות במרחב הווקטורי הנutan. לדוגמה:

א. מכפלה סקלרית על מרחב הווקטורים הגיאומטריים:

נתונים $\vec{v}, \vec{u} \in \mathbb{C}^n$ וקטורים גיאומטריים מסדר n מעל שדה המספרים המרוכבים. מכפלה פנימית בין שני וקטורים אלו, נקראת גם מכפלה סקלרית, המוגדרת על ידי:

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

כאשר \vec{u} הינו הצמוד המרוכב של u .

ב. מרחב הילברט – מרחב מכפלה פנימית על מרחב הפונקציות:

נניח שתי פונקציות מרוכבות $\mathbb{C} \rightarrow \mathbb{C}$: f, g אינטגרביליות בתחום כלשהו I (כמו שהוזכר לעיל, גם מרחב הפונקציות הוא מרחב וקטורי), אז המכפלה פנימית מוגדרת על ידי:

$$\langle f(x), g(x) \rangle = \int_I f^*(x)g(x)dx$$

כאשר f^* הינו הצמוד המרוכב של f .

ניתן להגדיר גם מרחבי מכפלה פנימית נוספים, נניח עבור מרחב המטריצות.

נורמה:

נורמה, מוגדרת על ידי מכפלה פנימית של וקטור בעצמו, ומסומנת ע"י $\| \cdot \|$, זאת אומרת:

$$\| \vec{v} \| = \sqrt{\langle \vec{v}, \vec{v} \rangle} \geq 0$$

שווין מתקיים אריך עבור וקטור האפס: $0 = \vec{v} = 0 \Leftrightarrow \| \vec{v} \| = 0$.

תמונה נוספת, נקראת א-שוויון המשולש, מתוארת על ידי:

$$\| \vec{u} + \vec{v} \| \leq \| \vec{u} \| + \| \vec{v} \|$$

א-שוויון נוסף הקשור לנורמות נקרא א-שוויון קושי שוורץ (Cauchy-Schwarz inequality):

$$\langle x, y \rangle \leq \|x\| \cdot \|y\|$$

כאשר $\langle y, x \rangle$ הינה המכפלה הפנימית בין שני הווקטורים, המוגדרת מעל הטבעיים כר: $y_i \cdot x_i$, והביטוי $\|x\| \cdot \|y\|$ הוא מכפלת הנורמות.

דוגמה:

א. במרחב הווקטורים הגיאומטריים, הגדרת הנורמה היא בעצם הגדרת אורך (או גודל הווקטור). נניח עבור הווקטורים הגיאומטריים התלת-ממדים, $V = \mathbb{R}^3$, אז עבור $V \in \mathbb{R}^3$, הנורמה מוגדרת ע"י $\| \vec{v} \| = \sqrt{x^2 + y^2 + z^2}$.

ב. במרחב הילברט נורמה של פונקציה $\mathbb{C} \rightarrow \mathbb{C}$: f הינה $\|f\| = \int_I |f(x)|^2 dx$.

אורותוגונליות

הגדרת מכפלה פנימית מאפשרת לנו להגדיר אורותוגונליות (או אנכיות) של שני וקטורים למרחב מכפלה פנימית מסוים. שני וקטורים $V \in \mathbb{R}^n$ נקראים אורותוגונליים זה לזה אם ורק אם המכפלה הפנימית שלהם הינה אפס:

$$\langle \vec{u}, \vec{v} \rangle = 0 \Leftrightarrow \vec{u} \perp \vec{v}$$

כאשר מתייחסים למרחב הווקטורים הגיאומטריים, קל להבין את מושג האורתוגונליות.

אורותוגונליות היא הכללה של תכונת הניצבות המוכרת מגאומטריה. בגאומטריה, שני ישרים במשור האוקלידי ניצבים זה זה אם הזרויות הנוצרת בנקודת החיתוך שלהם היא זוית ישרה (בת 90 מעלות). מושג האורתוגונליות כללית תכונה זו גם למרחבים וקטוריים n -ממדים. על מנת להכליל את מושג הניצבות יש ראשית להגדיר זוית בין שני וקטורים:

$$\cos(\theta) = \frac{\langle x, y \rangle}{\|x\| \cdot \|y\|}$$

לפי א-שוויון קושי שוורץ מתקיים: $\|y\| \cdot \|x\| \leq \langle y, x \rangle$, ולכן ימ' תמיד קטן או שווה בערכו המוחלט ל-1. כיוון שכן, תמיד ניתן לחשב זוית בין שני וקטורים באמצעות מכפלה פנימית.

לוקטוריים אורתוגונליים חשובות רבה כאשר חוקרים מרחב וקטורי יש מספר תכונות נוחות כאשר הוא אורתונורמלי (כל אבריו אורתוגונליים זה לזה ובعلילו). יתר על כן, מתרבר שבהינתן בסיס כלשהו למרחב וקטורי ניתן לקבל ממנו בסיס חדש שכל אבריו אורתוגונליים זה לזה, כך שתמיד ניתן למצאו בסיס נוח שכזה. דבר זה נעשה על ידי תהליך גרם-שמידט (gram-schmidt).

שני וקטורים אורתוגונליים יסומנים על ידי \mathbf{v} . עבור וקטורים אורתוגונליים מתקיימות התכונות הבאות:

- אם $\mathbf{v} \perp \mathbf{u}$, אז $\mathbf{u} \perp \mathbf{v}$.
- אם $\mathbf{v} \perp \mathbf{u}$, אז לכל סקלר λ גם $\mathbf{u} \perp \lambda\mathbf{v}$.
- אם $\mathbf{v} \perp \mathbf{u}$ וגם $\mathbf{u} \perp \mathbf{w}$, אז $\mathbf{v} \perp (\mathbf{u} + \mathbf{w})$.
- אם וקטור אורתוגונלי לקבוצה של וקטורים איז הוא גם אורתוגונלי לכל צירוף לינארי שלהם (נובע ממשת התכונות הקודמות).

וקטורים עצמיים וערכים עצמיים

תהי $\mathbb{F}^n \in A$ מטריצה ריבועית, וקטור $\mathbb{F}^n \in v \in \mathbb{F}^n$ ארך עצמי של A ו- $0 \neq v$ נקרא הווקטור העצמי המתאים אם מתקיים:

$$A \cdot v = \lambda \cdot v$$

ניתן להראות שעבור מטריצה A , הווקטורים העצמיים המתאים לסקלר λ הם כל פתרונות המשוואה ההומוגנית $0 = v(\lambda I_n - A)$.

אם נסמן $[v_1, \dots, v_n] = V$, אז מתקיים:

$$A = V \operatorname{diag}(\Lambda) V^{-1}$$

כאשר (Λ) הוא ערכי האלכסון של המטריצה A .

פירוק לערכים סינגולריים

ניתן לפרק מטריצה $\mathbb{R}^{m \times n} \in M$ למכפלה של שלוש מטריצות באופן הבא:

$$M = U \Sigma V^*$$

כאשר $\mathbb{C}^{m \times m} \in U$ היא מטריצה אוניטרית מרוכבת (או ממשית), $\mathbb{R}^{m \times n} \in \Sigma$ היא מטריצה אלכסונית שכל אברי האלכסון שלה ממשיים או-שליליים, $-1^{n \times n} \mathbb{C} \in V^*$ היא מטריצה אוניטרית מרוכבת (או ממשית). פירוק זה נקרא פירוק לערכים סינגולריים (Singular value decomposition - SVD).

ערכים האלכסון של $\Sigma = \sum_{ii}$ – מסודרים מהגדול לקטן, והם נקראים הערכים הסינגולריים של M . בנוסף, m העמודות של U נקראות הווקטורים הסינגולריים השמאליים של M , ובהתאם n העמודות של V הן הווקטורים הסינגולריים הימניים של M . שלוש המטריצות מקיימות את התכונות הבאות:

- הווקטוריים הסינגולריים השמאליים של M הם וקטורים עצמיים של $M^* M$.
- הווקטוריים הסינגולריים הימניים של M הם וקטורים עצמיים של $M M^*$.
- הערכים הסינגולריים (אברי האלכסון של Σ) שאינם אפס הם שורשים ריבועיים של הערכים עצמיים השונים מאפס של $M^* M$ ושל $M M^*$.

לפירוק SVD יש שימושים בתחוםים רבים, ואף ניתן להציג בעזרתו נורמות חדשות.

1.2.2 Calculus

פונקציה

פונקציה הינה התאמה (או העתקה), המתאימה לכל איבר x (בתחום מסוים), ערך ייחיד y , ומסומנת באופן הבא: $(x) f = y$. קבוצת הא-ים, נקראת תחום, וקבוצת הע-ים נקראת טווח. קבוצות התחום והטווח יכולות להיות רציפות (למשל מספרים ממשיים חיוביים) או בדידות (למשל קבוצה $\{1, 0\}$). בדרך כלל הסימון מופיע כך: $Y \rightarrow X$, כאשר X ו- Y הינם התחום והטווח בהתאם.

דוגמא: $\mathbb{R}^+ \rightarrow \mathbb{R}^2$: $f(x) = (x^2, x)$, הינה פונקציה, הולוקחת וקטורים גיאומטריים דו-ממדים, ומחזירה מספר ממשי אי שלילי. הפונקציה עצמה f היא הנורמה של הווקטור, כפי שהוגדרה בפרק הקודם.

נגזרת

עבור פונקציות ממשיות, נגזרת מוגדרת על ידי מידת השתנות של הפונקציה $(x) f$ על ידי שינוי קטן (אינפיניטסימלי) בא. באופן גיאומטרי, הנגזרת הינה השיפוע של הפונקציה בנקודה x . נגזרת מסומנת בדרך כלל ע"י $f'(x) = \frac{df}{dx}(x)$.

נגזרות של פונקציות אלמנטריות ניתן לחשב באמצעות כללים ידועים. לדוגמה:

- לכל $n \neq 0$ מתקיים: $\frac{d(x^n)}{dx} = nx^{n-1}$.
- חיבור או חיסור פונקציות: $\frac{d(f(x)+g(x))}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$.
- מכפלת שתי פונקציות: $\frac{d(f(x) \cdot g(x))}{dx} = \frac{f(x)dg(x)}{dx} + \frac{g(x)df(x)}{dx}$.
- כלל שרשרת: $\frac{df(g(x))}{dx} = \frac{df}{dg} \frac{dg}{dx}$.

כיוון שנגזרת של פונקציה ממשית מכמתת את קצב שינוי הפונקציה, אז בתחום שבו הפונקציה יורדת הנגזרת שם תהיה שלילית, ובתחום שבו היא עולה הנגזרת תהיה חיובית. ככל שקצב ההשתנות גדול יותר כך ערכה המוחלט של הנגזרת גדול.

הערה: לא לכל פונקציה מוגדרת נגזרת. למספר זהணיה שהפונקציה אנליטית ולכן גדרה.

הערה נוספת: כיוון שנגזרת של פונקציה היא גם פונקציה, ניתן גם להגדיר נגזרת שנייה או נגזרת מסדרים גבוהים יותר. בדרך כלל הסימון הינו $f''(x) = \frac{d^2f}{dx^2}(x)$ לנגזרת מסדר שני וכו'.

נקודות אקסטרום

נקודות אקסטרום של פונקציה, הן נקודות שבהם הפונקציה מקבלת ערך מקסימום או מינימום באופן מקומי. במקרה אליו, הנגזרת של הפונקציה "משנה ציווין" (מפונקציה עולה לפונקציה יורדת או להפך) ולכן מקבלת את הערך אפס. יש לשים לב שהתאפסות הנגזרת בנקודות המינימום והמקסימום היא תנאי הכרחי אך לא מספיק. יתכן שהנגזרת מתאפסת בנקודה מסוימת, אך נקודה זו אינה מינימום או מקסימום מקומי, אלא נקודת פיתול.

לדוגמה: $x^3 = f(x)$. נגזרת הפונקציה הינה $f'(x) = 3x^2$ והיא מתאפסת בנקודה $0 = x$.

גרדיאנט, יעקוביאן והסיאן

עבור פונקציה מרובת משתנים, נגזרת חלקית מוגדרת להיות הנגזרת של הפונקציה לפי אחד המשתנים בלבד, והיא מסומנת ב- $\frac{\partial f(x_1, \dots, x_n)}{\partial x_i}$. כאשר גוזרים לפי משתנה מסוים, שאר המשתנים הם קבועים ביחס לנגזרת. בהינתן הפונקציה $f(x_1, \dots, x_n)$, וקטור הנגזרות לפי כל המשתנים נקרא גרדיאנט:

$$\nabla f(x_1, \dots, x_n) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \leftrightarrow [\nabla f]_i = \frac{\partial f}{\partial x_i}$$

עבור m פונקציות הבלתיות ב- n משתנים, הייעקוביאן הוא מטריצת הנגזרות החלקיים:

$$\mathcal{J}_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{n \times m} \leftrightarrow [\mathcal{J}_f]_{ij} = \frac{\partial f_i}{\partial x_j}$$

עבור פונקציה $f(x_1, \dots, x_n)$, מטריצת הנגזרות מסדר שני נקראת הסיאן:

$$\mathcal{H}_f = \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}_{n \times n} \leftrightarrow [\mathcal{H}_f]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

שני כללים חשובים בחישוב נגזרות של מטריצות:

$$\nabla_x(a^T x) = a$$

$$\nabla_x(x^T A x) = (A + A^T)x$$

1.2.3 Probability

תורת ההסתברות היא תחום המספק כל ניתוח למאורעות המכילים ממד של אקראיות ואינטראקטיביות. הסתברות של מאורע הוא ערך מסוים למידת הסבירות שהוא יתרחש, כאשר ערך זה נע בין 0 ל-1 – מאורע בלתי אפשרי הוא בעל הסתברות 0, ומאורע ודאי הוא בעל הסתברות 1.

הגדרות בסיסיות

Ω = מרחב המדגם – מכלול האפשרויות השונות של ניסוי. לדוגמה עבור הטלת קובייה: $\{\Omega, 1, 2, 3, 4, 5, 6\}$.

קבוצה – חלק ממכלול המדגם. לדוגמה עבור הטלת קובייה: $A = \{2, 4, 6\}$ = even number
מאורע – תוצאה אפשרית של ניסוי.

הסתברות – סיכוי של מאורע להתרחש. עבור תת קבוצה A של מרחב המדגם Ω , ההסתברות לקיום מאורע מקובצת A שווה לחלק היחסי של מספר איברי הקבוצה מתוך קבוצת המדגם:

$$p(A) = \frac{\#A}{\#\Omega}, 0 \leq p(A) \leq 1$$

$A \cup B$ = איחוד – איחוד של שתי קבוצות הוא אוסף האברים של שתיהן הקבוצות. איחוד של הקבוצות A ו- B הוא אוסף האברים המופיעים לפחות פעם אחת משתי הקבוצות A או B . לדוגמה עבור הטלת קובייה:

$$A = \text{even number} = \{2, 4, 6\}, B = \text{lower than } 4 = \{1, 2, 3\}$$

$$\rightarrow A \cup B = \{1, 2, 3, 4, 6\}, \quad p(A \cup B) = \frac{5}{6}$$

$A \cap B$ = חיתוך – חיתוך של שתי קבוצות הוא אוסף האברים המופיעים בשתי הקבוצות. חיתוך של הקבוצות A ו- B הוא אוסף האברים המופיעים גם ב- A וגם ב- B . עבור הדוגמא הקודמת:

$$A \cap B = \{2\}, p(A \cap B) = \frac{1}{6}$$

מאורעות זרים – מאורעות שהחיתוך שלהם ריק, כלומר אין להם אברים משותפים:

$$A \cap B = \emptyset, p(A \cap B) = 0$$

מאורע משלים – מאורע המכיל את כל האברים שאינם נמצאים בקבוצה מסוימת:

$$A \cup A^c = \Omega, p(A \cup A^c) = 1, p(A) = 1 - p(A^c)$$

מאורעות בלתי תלויים: $P(A \cap B) = P(A) \cdot P(B)$. באופן אינטואיטיבי ניתן לחשב על כך שבמקרה זה ידיעת אחד אינה משפיעה על הסיכוי של השני.

אם המאורעות זרים (והם בעלי סיכוי שונה מ-0), הם בהכרח תלויים:

$$P(A \cap B) = 0 \neq P(A) \cdot P(B) > 0$$

$p(A|B) = \text{הסתברות מותנית} - \text{בгинتن מידע מסוים, מה ההסתברות של מאורע כלשהו:}$

$$p(A|B) = \frac{p(A \cup B)}{p(B)} \leftrightarrow p(A|B) \cdot p(B) = p(A \cup B) = p(B|A) \cdot p(A)$$

בעזרת ההגדרה של הסתברות מותנית ניתן לתת הגדרה נוספת למאורעות בלתי תלויים:

$$A, B \text{ תלויים} \Leftrightarrow p(A|B) = p(A)$$

נשים לב שהמשמעות של שתי ההגדרות זהה – המידע על B לא משנה את חישוב ההסתברות של A .

נוסחת ההסתברות השלמה וחוק ב'י'

נוסחת ההסתברות השלמה היא נוסחה פשוטה המאפשרת לחשב מאורעות מסוימים. ניתן לפרק מרחב הסתברות לאייריים זרים, ואז לחשב את ההסתברות של כל אייר בפni עצמו. אם ניקח את כל ההסתברויות המתקבלות, ונכפיל כל אחת מהן במשקל של אותו אייר, נקבל את נוסחת ההסתברות השלמה:

$$P(B) = \sum_i P(B|A_i) \cdot P(A_i)$$

מתוך נוסחה זו מגאים בקלות לחוק ב'י', המאפשרת לחשב הסתברות מותנית באמצעות ההתניתה ההפוכה:

$$p(A|B) = \frac{p(A \cap B)}{p(B)} = \frac{p(B|A)p(A)}{p(B)}$$

משפט הcalcul וההדחה

כדי לספור עצמים בקבוצה, אפשר לכלול ולהוציא את אותו עצם שוב ושוב, כל עוד בסוף ההליך נספר כל עצם פעמי אחת. עקרון פשוט זה מתרגם לנוסחה הבאה:

$$\left| \bigcup_{i=1}^n A_i \right| = \sum_{i=1}^n |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n-1} |A_1 \cap \dots \cap A_j|$$

עבור 2 קבוצות הנוסחה נהיית יותר פשוטה:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

במקרה זה, כאשר A, B זרות, אז $0 = |A \cap B|$.

עבור שלוש קבוצות מתקובלת הנוסחה:

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|$$

משתנים אקראיים

$\mathbb{R} \rightarrow \Omega$: משתנה מקרי – פונקציה המתאימה לכל מאורע השיר למרחב ההסתברות ערך מסווני, המהווה את הסיכוי של המאורע להתרחש.

פונקציית ההסתברות של משתנה מקרי X נותנת את הסיכוי של כל x אפשרי:

$$f_X: \mathbb{R} \rightarrow [0,1] = p(X = x)$$

פונקציה זו מקיימת שלוש אקסiomות:

- הסתברות של כל מאורע למרחב המדגם גדולה או שווה ל-0.
- סכום ההסתברויות של כל המאורעות למרחב שווה ל-1: $\sum p(X = x) = 1$
- סכום ההסתברויות של שני מאורעות זרים שווה להסתברות של איחוד המאורעות.

עבור משתנה מקרי רציף יש אינסוף מאורעות אפשריים, לכן ההסתברות של כל מאורע יחיד היא 0. لكن עבור משתנה מקרי רציף מכללים את פונקציית ההסתברות לפונקציה הנקראת פונקציית ההתפלגות (או פונקציית הצפיפות המצטברת), המחשבת את ההסתברות שמאורע יהיה קטן מערך מסוים:

$$F_X(a) = p(X \leq a) = \int_{-\infty}^a f_X(x)dx$$

ניתן לחשב בעזרת פונקציה זו את ההסתברות שמאורע $'\text{יה}'$ בטווח מסוים:

$$p(a \leq X \leq b) = F_X(b) - F_X(a) = \int_a^b f_X(x)dx$$

פונקציית ההתפלגות מקיימת את התכונות הבאות:

- $\lim_{a \rightarrow -\infty} F_X(a) = 0$
- $\lim_{a \rightarrow \infty} F_X(a) = 1$
- $\int_{-\infty}^{\infty} f_X(x)dx = 1$
- הפונקציה מונוטונית עולה במובן החלש: לכל $b \leq a$ מתקיים $F_X(a) \leq F_X(b)$
- $p(X \geq a) = 1 - F_X(a)$

תכונות ופרמטרים עבור משתנה מקרי

תוחלת – ממוצע משוקל של כל הערכים האפשריים, כל אחד מוכפל בהסתברות שלו:

$$\mathbb{E}[X] = \sum_i x_i P(X = x_i) = \int_{-\infty}^{\infty} xf(x)dx$$

תכונות:

- $\mathbb{E}[c] = c$
- $\mathbb{E}[\mathbb{E}[X]] = \mathbb{E}[X]$
- $\mathbb{E}[aX + b] = a\mathbb{E}[X] + b$, $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$

שונות – ממד פיזור הערכים ביחס לממוצע המשוקל (-התוחלת):

$$Var[x] = E[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = \int_{-\infty}^{\infty} x^2 f(x)dx - (\mathbb{E}[X])^2$$

סטיית תקן מוגדרת להיות שורש השונות: $\sigma = \sqrt{Var[X]}$

תכונות:

- אי שליליות: $Var[x] \geq 0$
- $Var[aX + b] = a^2 Var[X]$

שונות משותפת – ממד ליחס אפשרי בין שני משתנים מקרים:

$$cov(X, Y) = \mathbb{E}[X \cdot Y] - \mathbb{E}[X] \cdot \mathbb{E}[Y]$$

כאשר: $\mathbb{E}[X \cdot Y] = \sum_j \sum_i x_i y_j P(X = x_i \cap Y = y_j)$

מקדם המתאים – נרמול של השונות המשותפת: $\rho(X, Y) = \frac{cov(X, Y)}{\sqrt{V(X)V(Y)}}$. המקדם מקיים: $1 \leq |\rho|$.

שני משתנים מקרים מוגדרים בלתי מתואמים אם $cov(X, Y) = 0$. אם המשתנים בלתי תלויים אז הם בהכרח בלתי מתואמים.

בازURRENT השונות המשותפת ניתן לכתוב: $V[X + Y] = V[X] + V[Y] + 2 \cdot cov(X, Y)$

פונקציה יוצרת מומנטים (התמרת לפילס של פונקציית הצפיפות):

$$M_X(t) = \mathbb{E}[e^{tX}] = \begin{cases} \sum_{i=0}^n e^{t \cdot x_i} p_X(x_i) \\ \int_S e^{t \cdot x} f_X(x) dx \end{cases}$$

בעזרת פונקציה זו ניתן ליצור מומנטים, שימושיים ללמידה על המשתנים:

$$\frac{\partial^n M_X(t)}{\partial t^n} \Big|_{t=0} = \mathbb{E}[X^n]$$

המומנט הראשון הוא התוחלת והמומנט השני הוא השונות.

התפלגות מיוחדות (בדיד)

ישן כל מיני התפלגות מיוחדות, שופיעות בטבע בכל מיני מקרים ויש להן נוסחאות ידועות.

התפלגות ברנולי: ($X \sim Ber(p)$)

ניסוי בעל שתי תוצאות אפשריות "הצלחה" או "כשלון". המשתנה המקרי מקבל שני ערכים בלבד – 0 או 1, בהתאם להצלחה וכשלון.

$$P(X = k) = \begin{cases} 1, & k = 1 \\ 0, & k = 0 \end{cases}, \mathbb{E}[X] = p, V[X] = pq = p(1-p)$$

התפלגות בינומית: ($X \sim B(n, p)$)

בהתפלגות בינומית חוזרים על אותו ניסוי ברנולי n פעמים באופן בלתי תלוי זה בזה. מגדירים את X להיות מספר ההצלחות שהתקבלו בסה"כ. מסמן $b-k$ סיכוי להצלחה בניסוי בודד וב- k סיכוי לכישלון בניסוי בודד.

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \mathbb{E}[X] = np, Var[X] = npq$$

ציריך לוודא 3 דברים: 1) חוזרים על אותו ניסוי באופן בלתי תלוי. 2) חוזרים על הניסוי n פעמים. 3) X מוגדר כמספר ההצלחות המתקבלות בסה"כ.

התפלגות גיאומטרית: ($X \sim G(p)$)

חוזרים על ניסוי ברנולי. כאשר X מבטא את מספר הניסויים שבוצעו עד ההצלחה הראשונה. k מסמן את הסתירות ההצלחה בניסוי בודד.

$$P(X = k) = pq^{k-1}, \mathbb{E}[X] = \frac{1}{p}, Var[X] = \frac{q}{p^2}$$

להתפלגות זו יש שתי תכונות נוספות:

1) "תכנת חוסר זיכרון": ($P[X = (n+k)|X > k] = P(n|X > k)$).

2) ההסתברות שייעברו k ניסויים ללא הצלחה: ($P(X > k) = q^k$).

כמו כן, אם מעוניינים לדעת את מספר הניסיונות הממוצע הנדרש עד להצלחה ראשונה – יש לחשב את התוחלת של המשתנה המקרי X .

התפלגות אחדית: ($X \sim U[a, b]$)

בהתפלגות זו לכל תוצאה יש את אותה הסתברות. הערכים המתקבלים בהתפלגויות החל מ- a ועד b הינם בקפיצות של יחידה אחת (לדוגמא הגרלה של מספר שלם בין 1-100):

$$P(X = k) = \frac{1}{b-a+1}, k = a, a+1, \dots, b, \mathbb{E}[X] = \frac{a+b}{2}, Var[X] = \frac{(b-a+1)^2 - 1}{12}$$

התפלגות פואסונית: ($X \sim poi(\lambda)$)

התפלגות זאת מתאפיינת במספר אירועים ליחידת זמן כאשר לא פרמטר המיצג את קצב האירועים ליחידת זמן הנבחרת.

$$P(X = k) = \frac{e^{-\lambda} \lambda^k}{k!}, k = 1 \dots \infty, \mathbb{E}[X] = Var[X] = \lambda$$

יש לשים לב שכאן ההתפלגות נמצדת ליחידת זמן.

התפלגות היפר גאומטרית: $X \sim H(N, D, n)$

נתונה אוכלוסייה שמכילה N פריטים סה"כ, מתוךה D פריטים "מיוחדים" בעלי תכונה מסוימת. בוחרים מאותה אוכלוסייה n פריטים ללא החזרה. מגדירים את X להיות מספר הפריטים ה-"מיוחדים" שנדרגו.

$$P(X = k) = \frac{\binom{D}{k} \binom{N-D}{n-k}}{\binom{N}{n}}, \mathbb{E}[X] = \frac{nD}{N}, Var[X] = \frac{nD}{N} \left(1 - \frac{D}{N}\right) \frac{N-n}{N-1}$$

התפלגותBINOMIALE SHLILIT: $X \sim NB(r, p)$

חווזרים על אותו ניסוי ברנולי זה אחר זה באופן בלתי תלוי עד אשר מצליחים בפעם ה- r . כלומר, מבצעים את הניסוי עד שמבצעים r פעמים. מגדירים את X להיות מספר החזרות עד שהתקבלו r הצלחות.

$$P(X = k) = \binom{k-1}{r-1} p^r (1-p)^{k-r}, k = r, r+1, \dots, \infty \quad \mathbb{E}[X] = \frac{r}{p}, Var[X] = \frac{r(1-p)}{p^2}$$

התפלגותים מיוחדות (רציף)

התפלגות מעריכית: $(\lambda) X \sim exp(\lambda)$

התפלגות רציפה המאפיינת את הזמן עד להתרחשות מאורע מסוים. λ הוא ממוצע מספר האירועים המתרחשים ביחסית זמן (אותו פרמטר מההתפלגות הפואסונית). $(\lambda) X \sim exp(\lambda) < 0$.

גם בהתפלגות זו יש את תכונות חוסר הזיכרון: $P(X > (a+b)|X > a) = P(X > b|a)$

התפלגות אחידה: $X \sim U(a, b)$

זו ההתפלגות שפונקציית הצפיפות שלה קבועה בין a ל- b .

פונקציית הצפיפות: $b < x < a \Rightarrow f(x) = \frac{1}{b-a}$. פונקציית ההתפלגות המצתברת:

$$\mathbb{E}[X] = \frac{a+b}{2}, Var[X] = \frac{(b-a)^2}{12}$$

התפלגות נורמלית: $(\mu, \sigma^2) X \sim \mathcal{N}$

התפלגות נורמלית היא ההתפלגות חשובה מאוד כיון שהיא מופיעה בהמוני מקרים. פונקציית הצפיפות של ההתפלגות הנורמלית נראה כmo פעמוני, כאשר לעקומה קוראים גם עקומת גאות. ההתפלגות הנורמליות נבדלות אחת מהשניה באמצעות הממוצע וסטיית התקן (-הפרמטרים שמאפיינים את ההתפלגות). התפלגות נורמלית סטנדרטית היא התפלגות נורמלית בעלת תוחלת 0 ושונות 1:

$$X \sim \mathcal{N}(0,1)$$

עבור תוחלת ושונות σ, μ , פונקציית הצפיפות של משתנה נורמלי הינה:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

ניתן להשתמש במומנטים כדי למצוא קשרים בין ההתפלגות. למשל עבור שני משתנים המתפלגים נורמלית:

$$X \sim \mathcal{N}(\mu_x, \sigma_x^2), Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$$

המומנטים מקיימים:

$$M_X(t) \cdot M_Y(t) = e^{\mu_x t + \frac{1}{2} \sigma_x^2 t^2} \cdot e^{\mu_y t + \frac{1}{2} \sigma_y^2 t^2} = e^{(\mu_x + \mu_y)t + \frac{1}{2}(\sigma_x^2 + \sigma_y^2)t^2} = M_{X+Y}(t)$$

ולכן ניתן לחשב את ההתפלגות של $X + Y$:

$$X + Y \sim \mathcal{N}(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$$

אי שיוגנים

מרקוב

בහינתן $0 \leq X \leq \mathbb{E}[X]$, התוחלת $\mathbb{E}[X]$, עבור פרמטר $0 > a$ מתקיים:

$$p(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$$

צ'בישוב

בහינתן התוחלת $\mathbb{E}[X]$ והשונות $Var[X]$, עבור פרמטר $0 > a$ מתקיים:

$$p(|X - \mathbb{E}[X]| \geq a) \leq \frac{Var[X]}{a^2}$$

צ'רנוフ

בහינתן התוחלת $\mathbb{E}[X]$, עבור שני פרמטרים $0 > a, t > a$ מתקיים:

$$p(X \geq a) \leq \frac{\mathbb{E}[e^{tx}]}{e^{ta}}$$

ינט'

עבור משתנה מקרי X בעל תוחלת, עבור פונקציה g : $\mathbb{R} \rightarrow \mathbb{R}$ מתקיים:

$$g(E[x]) \leq \mathbb{E}[g(x)]$$

התפלגות דו ממדית

$$F_{x,y}(a, b) = P(x \leq a, y \leq b)$$

תכונות:

$$\lim_{a,b \rightarrow \infty} F_{x,y}(a, b) = 1$$

$$\lim_{a \rightarrow -\infty} F_{x,y}(a, b) = \lim_{b \rightarrow -\infty} F_{x,y}(a, b) = 0$$

$$\begin{aligned} P(c < x < a, d < y < b) &= P(x < a, y < b) - P(x < a, y < d) - P(x < c, y < b) + P(x < c, y < d) \\ &= F_{x,y}(a, b) - F_{x,y}(a, d) - F_{x,y}(c, b) + F_{x,y}(c, d) \end{aligned}$$

אם y, x בלתי תלויים אז מתקיים:

$$\forall a, b \ F_{x,y}(a, b) = F_x(a) \cdot F_y(b)$$

זוג משתנים נקרא דו-ממדי רציף אם קיימת פונקציית צפיפות דו-ממדית $f_{x,y}(s, t)$, כך שמתקיים:

$$P(x, y \in A) = \int f_{x,y}(s, t) ds dt$$

באופן שקול מתקיים:

$$f_{x,y}(s, t) = \frac{\partial^2}{\partial s \partial t} F_{x,y}(s, t) = \frac{\partial^2}{\partial t \partial s} F_{x,y}(s, t)$$

התפלגות שולית:

$$F_x(s) = P(x \leq s) = P(x \leq s, y \leq \infty) = \int_{-\infty}^s \int_{-\infty}^{\infty} f_{x,y}(x, y) dx dy$$

נוסחת ההסתברות השלמה לצפיפות (באופן שקול גם ל- (t)):

$$f_x(s) = \frac{d}{ds} F(x_s) = \int_{-\infty}^{\infty} f_{x,y}(s, y) dy$$

כעת ניתן גם לכתוב תנאי שקול למשתנים בלתי תלויים – y, x בלתי תלויים אם מתקיים:

$$\forall x, y f_{x,y}(X, Y) = f_x(X) \cdot f_y(Y)$$

סטטיסטייה היסקית

אם ידועים את סוג ההתפלגות אבל לא ידועים את מרכיביה, ניתן לאמוד את המרכיבים בעזרת מדגם. המדגם מאפשר לנו להשתמש באמצעות מספר מאורעות שווים ההתפלגות. דוגמא – נניח רוצם למדוד גובה של קבוצה מסוימת – כלל התלמידים בבית ספר מסוים. ידוע שגובה מתפלג נורמלית, אבל לא ידועים כאן את התוחלת והשונות. לשם כך ניתן להשתמש באומד – פונקציה שמנה לנתח את המאורעות ומתורן כרך להסביר את התוחלת והשונות.

בנитוח נמצא מנקודת הנחה שידוע כי הערכים במדגם נלקחים כולם מתוך ההתפלגות X , השיכת המשפחה של ההתפלגיות שתלויות בפרמטר אחד או יותר שאינם ידועים. (למשל במקרה: $X \sim N(\mu, \sigma^2)$ כאשר μ, σ לא ידועים). בפועל נתונות n דגימות בלתי תלויות מתוך ההתפלגות: X_1, X_2, \dots, X_n , ורוצים לאמוד את הפרמטרים הלא ידועים (פונקציה של הערכים שדגמוני).

אומד בלתי מוטה: אומד מוגדר להיות בלתי מוטה אם התוחלת של האומד שווה לפרמטר אותו אנו מנסים לאמוד, כלומר, אם $\theta = \hat{\theta}$ אז האומד הוא חסר הטיה. במקרים אחרים – אומד יהיה חסר הטיה אם התוחלת של המשתנה המקרי המוחושב לפי θ שווה $-\theta$ עבור כל θ .

דוגמאות לאמדים בלתי מוטים:

אומד בלתי מוטה לתוחלת – ממוצע חשבוני:

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\mathbb{E}(\hat{\theta}) = \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^n x_i\right] = \frac{1}{n} \sum_{i=1}^n \mathbb{E}[x_i] = \mathbb{E}[x_i] = \theta$$

אומד בלתי מוטה לשונות:

$$\mathbb{E}[s^2] = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

הוכחה:

$$\mathbb{E}[s^2] = \mathbb{E}\left[\frac{1}{n-1} \cdot \sum_i (x_i - \bar{x})^2\right] = \frac{1}{n-1} \sum_i \mathbb{E}(x_i - \bar{x})^2$$

$$\frac{1}{n-1} \sum_i \mathbb{E}[(x_i - \mu) - (\bar{x} - \mu)]^2$$

$$\frac{1}{n-1} \sum_i \mathbb{E}[(x_i - \mu)^2 - 2(x_i - \mu)(\bar{x} - \mu) + (\bar{x} - \mu)^2]$$

$$\begin{aligned}
& \frac{1}{n-1} \sum_i \mathbb{E}[(x_i - \mu)^2] - \mathbb{E}[2(x_i - \mu)(\bar{x} - \mu)] + \mathbb{E}[(\bar{x} - \mu)^2] \\
& = \frac{1}{n-1} \sum_i \sigma^2 - 2 \left(\frac{1}{n} \sum_j \mathbb{E}[(x_i - \mu)(x_j - \mu)] \right) + \frac{1}{n^2} \sum_j \sum_k \mathbb{E}[(x_j - \mu)(x_k - \mu)] \\
& = \frac{1}{n-1} \sum_i \left[\sigma^2 - \frac{2\sigma^2}{n} + \frac{\sigma^2}{n} \right] \\
& = \frac{1}{n-1} \sum_i \left[\frac{(n-1)\sigma^2}{n} \right] = \frac{n-1}{n(n-1)} \sum_i \sigma^2 = \sigma^2 \blacksquare
\end{aligned}$$

אומד נראות מרבית – (MLE)

בاهינתן סדרת דגימות מתוך התפלגות עם פרמטר לא ידוע, נגדיר את פונקציית הנראות שלhn כמכפלת ההסתברויות של כל הדגימות, או "הנראות של המדגם":

$$L(x_1, x_2 \dots x_n | p(\theta)) = \prod_i P_\theta(x_i)$$

זהוי פונקציה hn של הדגימות והן של הפרמטר.

אם ההתפלגות רציפה מגדים במקומות זאת את פונקציית הנראות להיות מכפלת הצפיפות:

$$L(x_1, x_2 \dots x_n | p(\theta)) = \prod_i f_\theta(x_i)$$

אומדן הנראות המקסימלית הוא פשוט הערך של הפרמטר שמקסם את פונקציית הנראות. כלומר, $\hat{\theta}$ הוא אומדן נראות מקסימלי עבור θ אם $\hat{\theta} = \arg \max_{\theta} L(x_1, x_2 \dots x_n | p(\theta))$.

מכoon *-log* הינה מונוטונית, למקסם את L שקול למקסם את $(\log(L))$, זה לרוב יותר קל, מכיוון שהמכפלה הופכת לסכום:

$$\log L(x_1, x_2 \dots x_n | p(\theta)) = \sum_{i=1}^n \log f_\theta(x_i)$$

נראה מספר דוגמאות לחישוב ה-MLE:

א. מציאת הפרמטר λ בהתפלגות פואסונית:

$$X \sim poi(\lambda)$$

שלב א' – נגדיר את אומדן הנראות – $L = (x_1, x_2 \dots x_n | p_\lambda) = \prod_i P_\lambda(x_i)$. בהתפלגות פואסונית מקיימת: $P(X=k) = \frac{e^{-\lambda} \lambda^k}{k!}$

$$\prod_i P_\lambda(x_i) = \prod_i \frac{e^{-\lambda} \lambda^{x_i}}{x_i!}$$

מוסבר למצוא להזיה מקסימום, لكن נוציא לאו:

$$\begin{aligned}
& \ln \left(\prod_i \frac{e^{-\lambda} \lambda^{x_i}}{x_i!} \right) = \sum_i \ln \left(\frac{e^{-\lambda} \lambda^{x_i}}{x_i!} \right) \\
& = \sum_i \ln(e^{-\lambda} \lambda^{x_i}) - \ln(x_i!) = \sum_i \ln(e^{-\lambda}) + \ln(\lambda^{x_i}) - \ln(x_i!)
\end{aligned}$$

$$= \sum_i x_i \ln(\lambda) - \lambda - \ln(x_i!)$$

כעת נגזר:

$$\frac{\partial L}{\partial \lambda} = \sum_i \frac{x_i}{\lambda} - 1 = \sum_i \frac{x_i}{\lambda} - \sum_i 1 = \sum_i \frac{x_i}{\lambda} - n$$

וכשנשווה ל-0 נקבל:

$$\sum_i \frac{x_i}{\lambda} = n$$

ובודד את הפרמטר אותו מנוטים לאמוד:

$$\lambda = \frac{\sum x_i}{n}$$

וקיבלנו אומד עבור הפרמטר λ , וכך אשר נתון סט התוצאות, פשוט נציב אותן, ונמצא מפורשות את הערך של האומד. זה בעצם תהליכי מציאת-*MLE*. כעת נבדוק האם האומד הוא מוטה או לא, כאשר משתמש בעובדה שעבור התפלגות פואסונית $\lambda = \mathbb{E}(x)$:

$$\mathbb{E}(\lambda) = \mathbb{E}\left(\frac{\sum_i x_i}{n}\right) = \frac{1}{n} \sum_i \mathbb{E}[x_i] = \frac{n\lambda}{n} = \lambda$$

קיבלנו שתווחלת האומד שווה לפרמטר, ולכן הוא בלתי מוטה.

ב. התפלגות נורמלית:

$$X \sim (\mu, \sigma^2)$$

פה יש שני פרמטרים לאמוד – התוחלת והשונות. ראשית נגדיר את הנראות:

$$f(X) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

לכן הנראות תהיה (נשים לב שהמכפלה תעבור לסכום במעירך של האקספוננט):

$$\prod_i f(x) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x_i-\mu)^2} = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{-\frac{1}{2\sigma^2}\sum_i (x_i-\mu)^2}$$

נוציא לוג:

$$\begin{aligned} \ln(L) &= \ln\left(\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n\right) + \ln\left(e^{-\frac{1}{2\sigma^2}\sum_i (x_i-\mu)^2}\right) \\ &= n \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \ln\left(e^{\frac{1}{2\sigma^2}\sum_i (x_i-\mu)^2}\right) \end{aligned}$$

נשים לב שבביטויו הראשון מה שיש בתוך ה- $- \ln(2\pi)^{-\frac{1}{2}} + (\sigma^2)^{-\frac{1}{2}}$, וזה בעצם $(2\pi)^{-\frac{1}{2}} + (\sigma^2)^{-\frac{1}{2}}$, ואז המעריך יכול לרדת מוחז ל- $-\ln$:

$$= -\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln(\sigma^2) - \frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2$$

כעת בשביל לאמוד את התוחלת יש לגזור לפי μ , וכדי לאמוד את השונות יש לגזור לפי σ^2 :

$$\frac{\partial L}{\partial \mu} = -\frac{(-2)}{2\sigma^2} \sum_i (x_i - \mu) = \frac{1}{\sigma^2} \sum_i (x_i - \mu)$$

וכשנשווה ל-0 נקבל:

$$\hat{\mu} = \frac{\sum_i x_i}{n}$$

ניתן להוכיח כי עבור התוחלת האומד הוא בעצם הממוצע של המדגם. אפשר לבצע תהליכי דומה על השונות, ומקבל הביטוי:

$$\hat{\sigma}^2 = \frac{\sum_i (x_i - \hat{\mu})^2}{n}$$

1 References

Intro:

<https://www.analytics.org.il/2019/12/ai-vs-deep-learning-vs-machine-learning/>

2. Machine Learning

2.1 Supervised Learning Algorithms

2.1.1 Support Vector Machines (SVM)

(SVM) Support Vector Machine הינו מודל למידה מונחית המשמש לניטוח נתונים לצורך סיווג, חיזוי ורגסציה. המודל מקבל אוסף של דוגמאות מתוויות במרחב d -ממדי, ומנסה למצוא משור המפריד בצורה טובה כמה שניתן בין דוגמאות האימון השתייכות לקטגוריות השונות.

המסוג הנוצר באמצעות מודל SVM הינו ליניארי, כאשר חלוקת הדוגמאות במרחב הווקטורי נעשית באופן צזה שיוציא מרוחק גדול ככל האפשר בין המשור המפריד לבין הנקודות הממוקמות היכי קרוב אליו. מרוחק זה מכונה שולים (margin), כאשר בצד אחד של השולים נמצאות דוגמאות עם label אחד, ובצד השני נמצאות הדוגמאות עם ה-label השני. את המשור המפריד ניתן לייצג באמצעות אוסף הנקודות $\vec{x} \cdot \vec{w} + b = 0$, כאשר \vec{w} הוא וקטור נורמלי של המשור.

נסח את האלגוריתם באופן פורמלי: נתון אוסף של n נקודות (x_i, y_i) , כאשר $y_i \in \{-1, 1\}$ מייצג את התוויג המתאים לדוגמא i , ו- $x_i \in \mathbb{R}^d$ הוא וקטור המאפיינים המתוארים את דוגמא i . מודל-SVM מייצר משור המפריד את המרחב לשני מרחבים אחד מהם אמרור להכליל עיקרי דוגמאות מסווג תיוג אחד. בנוסף, המודל מייצר שני משורים מקבילים לו, אחד מכל צד, במרחב צזה וגדול ככל האפשר:

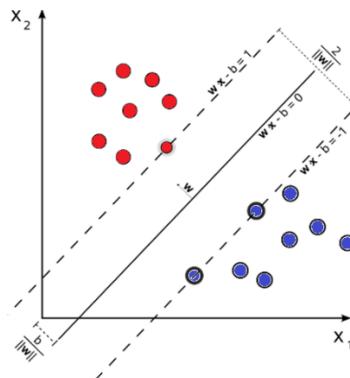
$$w^* = \operatorname{argmin}_w \left(\frac{1}{2} \|w\|^2 \right), \text{ s.t. } y_i (x_i \cdot w + b) \geq 1 \quad \forall i$$

כלומר, רוצים למצאו את וקטור המשקלות w המיציר שולים מרוחק $\frac{1}{2} \|w\|^2$ מהדוגמאות מתוויות נכון $y_i(x_i \cdot w + b) > 0$ ואין בתוך השולים (לא מתקיים: $y_i(x_i \cdot w + b) \leq 0$).

ישנו מספר גישות למציאת המפריד, ונפרט על כמה מהן.

Hard-Margin (hard SVM)

במצב הפשוט ביותר, המשווה עבר כל אחד מצדדיו של המפריד הינה פונקציה ליניארית של המאפיינים וכל הדוגמאות אשר סווגו נcona. מצב זה מכונה " הפרדה קשיחה " בו האלגוריתם מוצא את המשור עם השול הרחב ביותר האפשרי, ולא מאפשר לדוגמאות להיות בין הווקטורים התומכים. זהוי למעשה הפרדה מושלמת, והווקטורים התומכים הם למעשה הנקודות בקצוות השולים, כפי שניתן לראות באירוע:



איור 2.1 סיווג באמצעות אלגוריתם SVM עם מפריד בעל השולים הרחבים ביותר. הקווים המקווקווים מייצגים את משורי השולים. דוגמאות האימון המתלכחות עם משורי השולים נקראות וקטורי תומכים (support vectors), ומכאן נגזר שם האלגוריתם.

את המשורי בקצוות השולים ניתן לייצג באמצעות $y_i(x_i \cdot w + b) = 1$ or $y_i(x_i \cdot w + b) = -1$. גאומטרית, המרחק בין שני המשוריים הוא $\frac{2}{\|w\|}$, ולכן מנת למקסם את המרחק זהה, יש מהביא למינימום את $\|w\|$. על מנת שדוגמאות האימון לא יכללו בשולים המפרידים, יש להוסיף אילוץ לכל דוגמא i , באופן הבא:

$$y_i(x_i \cdot w + b) \geq 1$$

ailouz זה מחייב שכל דוגמא תימצא בצד הנכון של המפריד. לכן, במקרה זה יש לקיים את הדרישה הבאה:

$$\min_{w,b} \|w^2\|$$

$$s.t \quad y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 \quad \forall i = 1 \dots n$$

Soft-Margin (soft SVM)

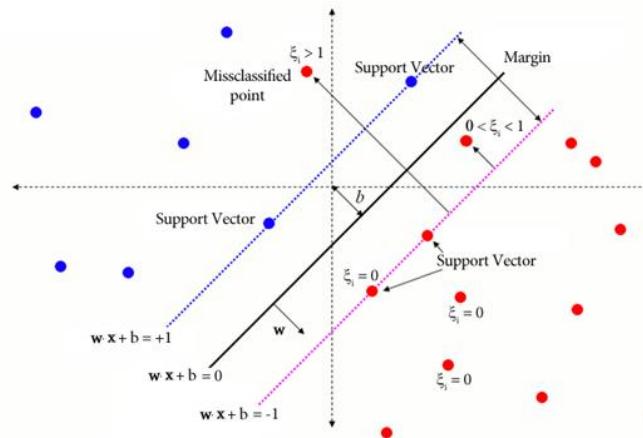
הפרדה מושלמת באמצעות מישור LINEAR לעתים קרובות אינה אפשרית, ולכן ניתן להרחיב את המודל כך שיאפשר לנקודות מסוימות לא להיות בצד המותאים להן. הרחבה זו, היוצרת "הפרדה רכה", מאפשרת לטפל בעיות שבהן אין הפרדה LINEAR בין הקבוצות, כמו למשל שיש נקודות חריגות. משמעות ההרחבה היא שכל וקטור ממוקם לפחות אחד מהאלצים, אך עם זאת, נרצה להגיע למצב בו האלצים מופרים "כמה שפחות". הפרדה רכה יוצרת מצב בו יש trade-off בין רוחב השולץ לבין השגיאות ומיציאת המשקלים האופטימליים של המסוווג. בגרסת זו יש לרשום באופן מועט שונה את בעיית האופטימיזציה, כאשר מתווסף משתנה המתיחס לנקודות שאין נמצאות בסיווג המתאים להן לפני המפריד:

$$\min ||w^2|| + C \sum_{i=1}^n \xi_i$$

$$s.t \quad y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1 \dots n \quad \xi_i \geq 0$$

לשם קבלת אינטואיציה, נשים לב לתפקיד המשתנים:

אם $\xi_i = 0$, מתקיים התנאי שנדרש בהפרדה קשיחה, כלומר הנקודה x_i גם נמצאת בצד הנכון של המפריד וגם מתקיימת הדרישה לשמירה על השולדים. אם $1 < \xi_i < 0$ אז הנקודה x_i נמצאת בצד הנכון של המפריד המסווג, אבל המסווג קרוב אליה, כך שהנקודה נמצאת בתחום השולדים. C הינו קבוע שאחראי על "ענישה" של דוגמאות שאינן בצד הנכון של המפריד. ערך C גבוה פירושו העדפת הסיווג הנכון על פני שלויים רחבים, ואילו C נמוך מעדייף הכללה (שלויים רחבים), גם במקרים האימון הספציפיות אין מסוגות נכונן.



איור 2.2 סיווג באמצעות אלגוריתם SVM עם הפרדה רכה. המשתנה ξ שווה לאפס אם הנקודה ממוקמת בצד הנכון של המפריד. גודול מאפס כאשר הנקודות נמצאות בצד הלא נכון של המפריד.

Non-linear Separation

מסוגים LINEAR מוגבלים ביכולת הכללה שלהם בגלל הפשטות שלהם. לכן, כאשר לא ניתן להפריד אוסף דוגמאות באמצעות מפריד LINEAR, משתמשים ב"הפרדה ALINEARIT". גישה זו מאפשרת להשתמש ב-SVM ל逶וג לא LINEAR, על ידי טרנספורמציה לא LINEARIT, כמו למשל "תעלול הגערין" (Kernel Trick). בגישה זו מבצעים מיפוי לדאטה מרחב אחר, בו ניתן למצוא עבורו הפרדה LINEARIT, וממילא יהיה אפשר להשתמש באלגוריתם SVM. כך למשל, קיימת אפשרות ליצור מאפיינים חדשים על ידי העלאת ערכי המאפיינים הקיימים בחזקה מסוימת, המכונים בפונקציות טריגונומטריות וכו'.

באופן פורמלי', נחפש פונקציית מיפוי להעתקת מרחב $F \rightarrow \chi$: ψ כך שבמרחב F ניתן יהיה להפריד את הנתונים $\{y_i(x_i)\}_{i=1}^N$ באמצעות מסוג לינארי. לשם כך, משתמשים בטריק קרNEL שמקבל כקלט וקטורים למרחב המקורי ומחזיר את המכפלה הפנימית (dot product) של הווקטורים למרחב החדש (נקרא גם מרחב התכונות – feature space):

$$K(\vec{x}_i, \vec{x}_j) = \psi(\vec{x}_i)^T \psi(\vec{x}_j)$$

דוגמאות של פונקציות קרNEL נפוצות:
קרNEL לינארי:

$$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$$

זהה הפונקציה הcy פשוטה, המוגדרת על ידי מכפלה פנימית של הווקטורים. במקרה זה מרחב התכונות ומרחב הקלט זהים ונחזר לפתרון בעזרת SVM לינארי:
קרNEL פולינומי:

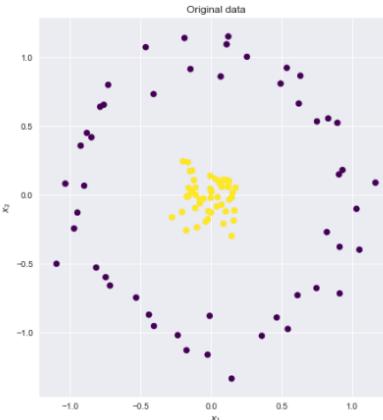
$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + c)^d$$

העתקה מהמרחב המקורי למרחב שמהווה פולינום ממעלה 2 $\geq d$. $0 \geq c$ הוא פרמטר חופשי המשפיע על היחס בין סדר גובה לעומת סדר נמוך בפולינום. כאשר $c = 0$, הkrnel נקרא הומוגני.
קרNEL גאוסיאני:

$$K(\vec{x}_i, \vec{x}_j) = \exp(-\gamma(\vec{x}_i - \vec{x}_j)^2), \gamma > 0$$

הפרמטר γ מלא תפקיד חשוב, יש לבחור אותו בהתאם לביעיה העומדת בפנים. אם הערך שלו קטן מאוד, האקספוננט ינהג כמעט כמעט-original לינארי וההטלה למרחב מממד גבוה יותר יתחליל לאבד מכוחו הלא לינארי. מצד שני, אם נעריך אותו יתר על המידה, הפונקציה לא תהיה סדירה וגבול ההחלה יהיה רגש מאוד לרעש בתנויי האימון.

המהות של טרייק קרNEL היא שניתן לבצע את ההעתקה גם מבלי לדעת מהי הפונקציה ψ , אלא הידעה של K מספקיה.
לצורך קבלת אינטואיציה והמחשה נביא דוגמא. נתון מערך הנתונים הבא:



ניתן לראות שלא ניתן להפריד בין הנקודות הצהובות לשגולות על ידי מישור הפרדה לינארי. لكن נחפש מרחב אחר, מאותו ממד או בעל ממד גבוה יותר, בו ניתן יהיה להפריד בין נקודות אלה באופן לינארי. לצורך כך נבצע את הפעולות הבאות:

- א. נמפה את התכונות המקורי למרחב הגובה יותר (מייפוי תכונות).
- ב. נבצע SVM לינארי למרחב החדש.
- ג. נמצא את קבועות המשקלות התואמות את מישור גבול ההחלה.
- ד. נמפה את מישור המפריד בחזרה למרחב הדו-ממדי המקורי כדי לקבל גבול החלטה לא לינארי.

ישנם הרבה מרחבים מממדים גבוהים יותר בהם ניתן להפרדה לינארית. נציג דוגמא אחת:

$$x_1, x_2 \rightarrow z_1, z_2, z_3$$

$$z_1 = \sqrt{2}x_1x_2 \quad z_2 = x_1^2 \quad z_3 = x_2^2$$

למעשה נעזרנו בטריק קרナル. כאמור, בהינתן שקיימות פונקציה שסופה $\mathbb{R}^n \rightarrow \mathbb{R}^m$: φ את הווקטורים ממרחב \mathbb{R}^n למרחב תכונות כלשהו \mathbb{R}^m , אז המכפלה הפנימית של x_1 ו- x_2 במרחב זהה היא $(x_2)^T \varphi(x_1) \varphi(x_2)$. קרナル היא פונקציה K שהייכת למכפלה פנימית זו, כלומר $K(x_1, x_2) = \varphi(x_1)^T \varphi(x_2)$. אם נוכל למצוא פונקציית קרナル המקבילה למפת התכונות שלעיל, נוכל להשתמש בפונקציה ייחד עם SVM לינארי וכך לבצע את החישובים בעילוות.

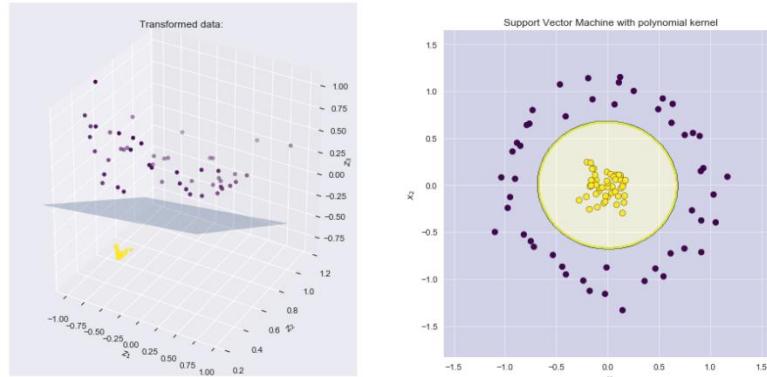
מתברר שמרחב התכונות שלעיל תואם את השימוש בקרナル פולינומי ידוע: $(x^T x')^d = (x' x)^d = K(x, x')$. נבחר $d=2$. נסמן $(x_1, x_2)^T = \alpha$ ונקבל:

$$K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix}\right) = (x_1 x'_1 + x_2 x'_2)^2 =$$

$$2x_1 x'_1 x_2 x'_2 + (x_1 x'_1)^2 + (x_2 x'_2)^2 = (\sqrt{2}x_1 x_2 x_1^2 x_2^2) \begin{pmatrix} \sqrt{2}x'_1 x'_2 \\ x'_1^2 \\ x'_2^2 \end{pmatrix}$$

$$K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} x'_1 \\ x'_2 \end{pmatrix}\right) = \varphi(\alpha)^T \varphi(\alpha')$$

$$\varphi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} \sqrt{2}x_1 x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$$



איור 2.3 שימוש ב-SVM לצורכי הפרדה לאחר ביצוע Kernel trick. המעגל באירור הימני מומפה למשור הפרדה לינארי למרחב מממד גבוה יותר, כפי שנitinן לראות באירור השמאלי. ניתן לראות שאחרי המיפוי שנעשה בעזרת kernel trick, המוקודות אכן מופרדות בצורה לינארית.

2.1.2 Naive Bayes

סיווג בייסיאני הוא מודל המשמש בחוק ביסוס על מנת לסתור אובייקט $\mathbb{R}^n \in x$ בעל n מאפיינים לאחת מ- K קטגוריות אפשריות. יחד עם השימוש בחוק ביסס, המודל מניח "נאיביות" – בהינתן סיווג של אובייקט מסוים, אין תלות בין המאפיינים השונים שלו.

נניח שיש מודל המקובל וקטור מאפיינים בינהירים {cab, ראש, משתעל, חום גבורה}, ומסווג האם בעל תכונות אלה חוליה בשפעת או לא. באופן כללי ניתן לומר שש תלות בין שייעול לבין חום גבורה, ככלומר העובדה שיש לאדם חום מעלה את ההסתברות שהוא גם משתעל. למרות זאת, ניתן להניח באופן "נאיבי" שאם כבר יודעים שאדם חוליה בשפעת, אז כבר אין יותר תלות בין היותו משתעל להיותו בעל חום. באופן פורמלי, אמן סביר להניח שמתפקידים (משתעל) $k > (\text{חום} | \text{משתעל})^k$, אך ניתן להניח נאיביות ולקבל: $(\text{שפעת} | \text{משתעל})^k = (\text{שפעת}, \text{חום} | \text{משתעל})^k$.

באופן כללי, סיווג בייסיאני נאיבי מניח שההינתן הסיווג של אובייקט מסוים, המאפיינים שלו בלתי תלויים. ההנחה זו כמובן לא תמיד מדויקת, וממילא גם ערכי ההסתברויות הנובעים ממנה ומשמשים לשיווג אינם מדויקים, אך ההנחה

מקלה מאוד על חישוב ההסתברויות של הסיווג הביסיאני, ובמקרים רבים תחת ההנחה זו התקבלו תוצאות סיווג. הסיבה להצלחת המודל נעה בכך שבבבליות סיווג העיקר הוא למצוא את הסיווג הסביר ביותר לאובייקט (שפעת או לא-פעת לנבדק בדוגמה), ולא דוחק לקבל הסתברות מדויקת לכל סיווג. במקרים רבים למרות שההסתברות הנובעת מההנחה הנאייבית אינה מדויקת עבור שני סיווגים אפשריים, היא בכל זאת שומרת על סדר הסתברות שלהם.

נtablename בוקטור מאפיינים $(y_1, \dots, y_k) = y \in \mathbb{R}^n$, היכל להיות שיר לאחת מ- K קטגוריות $(x_1, \dots, x_n) = x$, והסתברות הפריאוריות $p(y_k|y_i)$ ידועה, ובנוסף ידועות התפלגות המותנות של המאפיינים בהינתן הסיווג – $p(x_i|y_k)$. בעזרה הנתונים האלה רוצים לסwoג את x לאחת מהקטגוריות, כלומר למצוא את y_k שבעורו הביטוי $p(y_k|x)$ הוא מקסימלי. באופן פורמלי ניתן לנוכח זאת כך:

$$y = \arg \max_k p(y_k|x), k = 1 \dots K$$

בשביל למצוא את y_k האופטימלי ניתן להיעזר בחוק ביטוי:

$$p(y_k|x) = \frac{p(y_k, x)}{p(x)}$$

המכנה לא תלוי ב- k , ולכן מספיק למצוא את y_k שעבור המונה מקסימלי. לפי כלל השרשרת מתקיים:

$$\begin{aligned} p(y_k, x) &= p(y_k, x_1, x_2, \dots, x_n) = p(y_k, x_1 | y_k, x_2, \dots, x_n) \cdot p(y_k, x_2 | y_k, x_3, \dots, x_n) \cdots \\ &= p(x_1 | y_k, x_2, \dots, x_n) \cdot p(x_2 | y_k, x_3, \dots, x_n) \cdots = p(x_1 | y_k) p(x_2 | y_k) \cdots p(x_n | y_k) \\ \text{כעת ניתן להשתמש בהנחה הנאייבית, לפי בהינתן הסיווג } y_k, \text{ אין תלות בין המאפיינים. לפי הנחה זו נוכל לפשט את הביטוי:} \\ &= p(x_1 | y_k) \cdot p(x_2 | y_k) \cdots p(x_{n-1} | y_k) \cdot p(x_n | y_k) \\ &= p(y_k) \prod_{i=1}^n p(x_i | y_k) \end{aligned}$$

בביטוי זה כל האיברים ידועים, ולכן קל שנתרז זה רק להציב את הנתונים ולקבל את y_k שעבורו ביטוי זה他会很大：

$$y = \arg \max_k p(y_k) \prod_{i=1}^n p(x_i | y_k)$$

בדוגמה שהובאה לעיל, המאפיינים קיבלו ערכים בדים, ולכן ניתן לחשב את ההסתברות המותנית של כל מאפיין $(y_k|y_i)$ על ידי ספירת כמהם הפעמים שמופיע כל מאפיין באוכטוסייה הנדגמת ולחולק בגודל המדגם. לעומת זאת, רציפים (כמו למשל מחיר מניה, גובה של אדם וכדו'), אין אפשרות לחשב כך את ההסתברות המותנית. במקרים כאלה יש להניח התפלגות מסוימת עבור המדגם, ולהשתמש הפרמטרים של התפלגות שיטות שונות (למשל בשיטת מינימום נורמלית – MLE). עבור מדגם המתפלג נורמלית, ההסתברות המותנית היא גאומטרית:

$$p(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}}$$

כאשר σ_y^2, μ_y הם הפרמטרים של התפלגות, ואומרם הם משוערים בשיטת שערוך אחרת. אם התפלגות היא לא נורמלית, ניתן להשתמש באלגוריתם [Kernel density estimation](#) עבור שערוך התפלגות. ניתן אחרת להתמודדות עם מאפיינים היכולים לקבל ערכים רציפים היא לבצע דיסקרטיזציה לערכים אוטם המאפיינים יכולים לקבל.

במקרה [המולטיבומי](#), בו התפלגות היא רב ממדית ומיצינית תוצאה של סדרה בלתי תלואה, יש לחשב את הנראות בהתאם למתחאים להתפלגות מולטינומית. כדי להבין את החישוב נביא קודם קודם בדוגמה – נניח ורוצים לבנות מודל סיווג ביסיאני המזהה הودעות ספאם. נתונות 12 הודעות, מתוכן 8 אמיתיות ו-4 ספאם. כעת נניח וכל ההודעות מורכבות מארבע של ארבע מילים, בהתפלגות הבא:

Real (R) – {Dear, Friend, Lunch, Money} = {8, 5, 3, 1}.

Spam (S) – {Dear, Friend, Lunch, Money} = {2, 1, 0, 4}.

נחשב את הנראות – ההסתברות של כל מילה בהינתן הסיווג:

$$p(\text{Dear}|R) = \frac{8}{17}, p(\text{Friend}|R) = \frac{5}{17}, p(\text{Lunch}|R) = \frac{3}{17}, p(\text{Money}|R) = \frac{1}{17}$$

$$p(\text{Dear}|S) = \frac{2}{7}, p(\text{Friend}|S) = \frac{1}{7}, p(\text{Lunch}|S) = 0, p(\text{Money}|S) = \frac{4}{7}$$

כעת נבחן מה ההסתברות שהצירוף "Dear friend" הוא אמיתי (הצירוף הוא למעשה התפלגות מולטינומית, כיוון שהוא מכיל שתי מיללים שאין בין ההסתברויות שלhn קשור ישירות):

$$p(\text{Dear friend is R}) = p(R) \cdot p(\text{Dear}|R) \cdot p(\text{Friend}|R) = 0.67 \cdot 0.47 \cdot 0.29 = 0.09$$

$$p(\text{Dear friend is S}) = p(S) \cdot p(\text{Dear}|S) \cdot p(\text{Friend}|S) = 0.33 \cdot 0.29 \cdot 0.14 = 0.01$$

מספרים אלה ניתן להסיק שהצירוף "Dear friend" אינו ספאם.

באופן כללי, עבור וקטור מאפיינים $(x_1, \dots, x_n) \in \mathbb{R}^n$, הנראות מחושבת באופן הבא:

$$p(x|y_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p(y_{ki})^{x_i}$$

על הציר הלוגריטמי, בעזרתו נסוכה זו ניתן לבנות מסווג לינארי:

$$p(y_k|x) = \frac{p(y_k, x)}{p(x)} \propto p(y_k) \cdot \prod_i p(y_{ki})^{x_i}$$

$$\rightarrow \log p(y_k|x) \propto \log p(y_k) + \sum_i x_i \cdot \log p(y_{ki}) \equiv b + w^T x$$

היחסון בשימוש במסווג בייסיאני נאבי בעיות מולטינומיות נועז בכך שיש הרובה צירופים שלא מופיעים יחד בסע האימון, ולכן הנראות שלהם תמיד תהיה 0, מה שפוגם באמינות התוצאות.

מקרה דומה להתפלגות מולטינומית הוא מקרה בו המאפיינים הם משתני ברנולי, המקבלים ערכים בינאריים. במקרה זו הנראות הינה:

$$p(x|y_k) = \prod_{i=1}^n p_i^{x_i} (1 - p(y_{ki}))^{1-x_i}$$

עבור דатаה לא AMAZON, ניתן להשתמש באלגוריתם שנקרא CNB (complement naive Bayes). לפי אלגוריתם זה, במקום ללקחת את $p(x|y_k)$ נלקחים את המינימום של הפונקציה ההופכית:

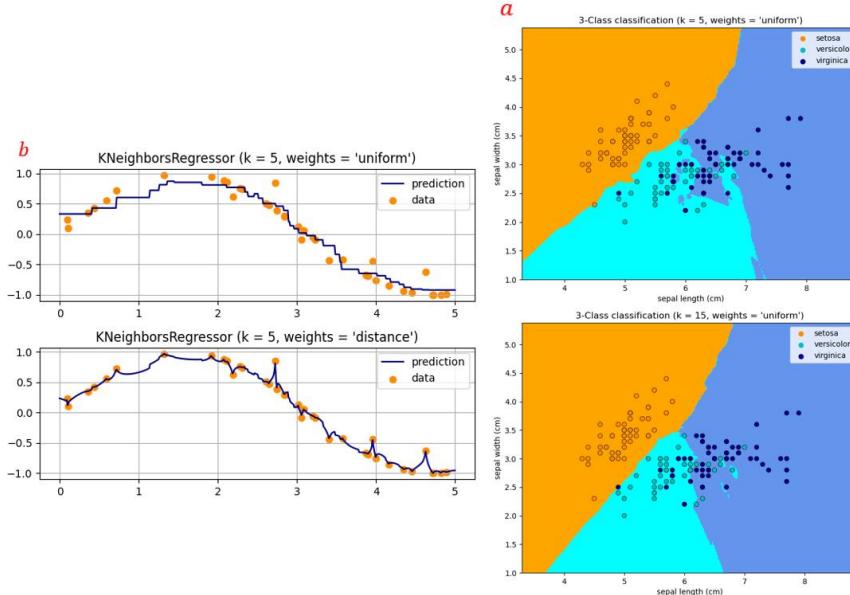
$$\arg \min_k p(y_k) \prod_{i=1}^n \frac{1}{p(x_i|y_k)}$$

שימוש באלגוריתם זה הוכח כיעיל במקרים בהם הדטה אינם AMAZON והביצועים של מסווגים בייסיאנים אחרים (גיאויסיאני או מולטינומי) היה לא מספיק טוב.

2.1.3 K-Nearest Neighbors (K-NN)

אלגוריתם השכן הקרוב הינו אלגוריתם של למידה מונחית, בו נתנות מספר דוגמאות ובונספ' ידוע ה-label של כל אחת מהן. אלגוריתם זה מתאים הן לבעיות סיווג (שיוך נקודה חדשה למחלקה מסוימת) והן לבעיות רגרסיה (נתינת ערך מסוין לנקודה חדשה). האלגוריתם הינו מודל חסר פרמטרים, והוא מבצע סיווג לנ נתונים בעזרת הcrcutta הרוב. עבור כל נקודה במדגם, המודל בוחן את ה-label של K nearest הנקודות הקרובות אליו ביותר, ומסווג את הנקודה לפי ה-label שקיבל את מרבית הקווות. מספר הנקודות הקרובות, K, הוא היפר-פרמטר שנקבע מראש.

אלגוריתם השכן הקרוב הוא אחד המודלים הנפוצים והפשוטים ביותר בלמידת מכונה, ואומר בכך שהוא מתאים גם לבעיות גרגסיה. המודל יפעל בצורה דומה בשני המקדים, כאשר ברגרסיה תבצע שקלול של מוצע בין השכנים הקרובים, ולא הכרעת הרוב, כלומר, התוצאה לא תהיה סיווג-label מסוים לפי הערך הנפוץ ביותר בקרוב K השכנים הקרובים, אלא חישוב ממוצע של כל labels השכנים. התוצאה המתקבלת היא ערך רציף, המיצג את הערכים בסביבת התצפית. ניתן להתחשב במרקח של שכן מהצפית בזורה שווה (uniform), וכן לתת משקל שונה של שכן בהתאם למרקח שלו מהנקודה אותה רצימן לחשב, כך שככל שכן מסוים קרוב יותר לנקודה אותה רצימן לחשב כך הוא יותר ישפיע עליה, ביחס לשכן בין הנקודה (distance).



איור 2.4 (a) סיווג בעזרת אלגוריתם N-NN-K: מסוגים את המרחק לאזוריים בהתאם ל-K השכנים הקרובים ביותר, כך שאם תבוא נקודה חדשה היא תהיה מסוגת בהתאם לצבע של האזור שלאה, הנקבע כאמור לפי השכנים הקרובים ביותר. ניתן לראות שישandel הבדל בין ערכי K שונים, וככל ש-K יותר גבוהה ככל האזוריים יותר חלקיים ויש פוחות מובלעות. (b) גרגסיה בעזרת אלגוריתם NN-K: קביעת ערך הע-בהתאם ל-K השכנים הקרובים ביותר. ניתן לתת משקלים שווים לכל השכנים, או לתת משקל ביחס למראק של שכן מהנקודה אותה רצימן לחשב.

לעתים נאמר על המודל שהוא "עצלן". הסיבה לכך היא שבשלב האימון לא מבצע תהליכי ממשמעותי, מלבד השמה של המשתנים וה-labels (variables) של המחלקה, כמו גם כל נקודה מסוימת למחלקה מסוימת. עקב לכך, כל מבחן האימון (או רובו) נדרש לצורך התחזית, מה עשוי להפוך את המודל לאיטי כאשר יש הרבה נתונים. למורדות זאת, המודל נוחש לאחד המודלים הקלאסיים הבוטניים, בזכות היתרונות שלו. הוא פשוט וקל לפירוש, עובד היטב עם מספר רב של מחלקות, ומתאים לפחות גרגסיה וסיווג. בנוסף הוא נוחש אמין במיוחד במקרה של נחחות הנקודות לגבי התפלגות הנתונים (כמו גרגסיה ליניארית למשל).

מנגד, יש לו מספר חסרונות. עקב העבודה שהוא דורש את כל נתונים האימון לצורך התחזית, הוא עשוי להיות איטי כאשר מדובר על דатаה עשיר. מסיבה זו הוא גם יעיל מבחינת זיכרון. מכיוון שהמודל דורש את כל נתונים האימון לצורך המבחן, כושר ההכללה שלו עשוי להיות לヒיגם (Generalization). ניקח לדוגמא מורה של כתה בבית ספר, המנסה לסwoוג את התלמידים למספר קבוצות. אם יעשה זאת לפי צבע שער, עיניים, לדוגמאות, סביר להניח שלא יתקשה בכך; אם לעומת זאת הוא ינסה לסwoוג לפי צבע שער, עיניים, חולצה, מכנסיים, נעליים, וכו' – סביר שיתתקל בקושי. במצב זה, כל תלמיד רחוק מרעהו באופן שווה כיון שאין שני תלמידים שזהם לחוטין בכל הפרמטרים, מה שמקשה על חישוב המרחק. בעיה זו מכונה קלתת הממדיות (Curse of dimensionality), ולכן מומלץ להיעזר באמצעות להורדת הממד (Dimensionality reduction).

קושי נוסף הקיים במודל הוא הצורך בחירת K-הנקון, מטלה שעשויה להיות לא קללה לעיתים. בכל שימוש של אלגוריתם השכן הקרוב, K הינו היפר-פרמטר שצריך להיקבע מראש. היפר פרמטר זה קובע את מספר הנקודות אשר האלגוריתם יתחשב בהן בעת בחרית סיווג התצפית. בחירת היפר-פרמטר קטן מדי, לדוגמה K=1, יכולה לגרום למצב בו המודל מותאם יתר על המידה לנכוני האימון, מה שוביל לדיווק גבוה נתונים האימון, ודיווק נמוך נתונים המבחן. מן העבר השני, כאשר K גבוהה מדי, למשל K=100, נוצר מצב ההופך – מודל שמתחשב יותר מדי בDATAה ולא מצליח למצא הכללה נכונה לסיווג. מומלץ לבחור K איזוגי לבגלאוון הפעולה של האלגוריתם – הכרעת

הרוב. כאשר בוחרים K זוגי, עלולים להיתקל במצב של שווין אשר עשוי להוביל לתוכאה מוטעית, ולכן כדי להימנע מתיקו כדאי לבחור K אי-זוגי.

כמו אלגוריתמים רבים מבוססי מרחק, אלגוריתם השכן הקרוב רגש לערכים קיצוניים (Outliers) ושימוש 알고יריהם ללא טיפול בערכים קיצוניים עשוי להוביל לתוכאה מוטעית. מלבד זאת, חשוב לנרמל את הנתונים לפני שימוש במודל. הסיבה לכך היא שהאלגוריתם מבוסס מרחק; במצב זה, יתרכו מרחקים בין תכיפות אשר עשויים להשפיע על החלטת המודל, למراتות שמרחוקים אלו הם חסרי משמעות לצורך הסיג. דוגמא לכך היא משתנה שעשויה לשמש ביחסות מידת שונות (מיילומטרים). ההחלטה האם להשתמש בקילומטרים או במילים עלולה להטוט את תוצאות את המודל, למراتות שבפועל לא השתנה דבר.

השיטה הנפוצה ביותר למדידת מרחק בין משתנים רציפים היא מרחק אוקלי – עבור שתי נקודות במרחב, המרחק ביניהם יחושב לפי הנוסחה: $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} = d$. במידה ומדובר במשתנים בדידים, כגון טקסט, ניתן להשתמש במטריקות אחרות כגון מרחק המיניג, המודד את מספר השינויים הדרושים כדי להפוך מחרוזת אחת למחרוזת שנייה, ובכך למדוד את הדמיון ביניהן.

לפני שימוש באלגוריתם השכן הקרוב, יש הכרח לוודא שהמחלקות מאוזנות. במידה ומספר דוגמאות האימון באחת המחלקות גבוהה מאשר בשאר המחלקות, האלגוריתם ייטה לטעוג למחלקה זאת. הסיבה לכך היא שבשל מסגרן הגדל, מחלוקת זו צפואה להיות נפוצה הרבה יותר בקרב K השכנים של כל תכיפות. הדבר עשוי להוביל לתוכאה מוטעת, ולכן יש לוודא מראש שכן יש איזון בין המחלקות השונות.

2.1.4 Quadratic\Linear Discriminant Analysis (QDA\LDA)

Quadratic Discriminant Analysis הינו מודל נוסף לסיווג של נתונים לקבוצות שונות, המניח שבהתאם סיווג של אובייקט מסוים – מתקבלת התפלגות נורמלית, ככלומר בהינתן $\{y_k, k \in \{1, \dots, K\}, y_k, \text{ מתקיים}:$

$$x | y_k \sim N(\mu_k, \Sigma_k)$$

ובאופן מפורש, עבור $x \in \mathbb{R}^{n \times d}$ הפילוג המותנה הוא:

$$p(x|y = k; \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)\right)$$

בעזרת הנחה זו, ניתן למצוא מסווג אופטימלי עבור $(x|y_k) = y$. לפי חוק ב'יוס:

$$p(y_k|x) = \frac{p(x|y = k)p(y)}{p(x)}$$

המכנה קבוע ביחס ל- y_k , ואת $(y|k)$ ניתן לשער בקבילות על פי השכיחות של כל label במדגם – $\frac{\mathbb{1}_{y=k}}{n}$. את הביטוי הנוסף במונה – $(x|y = k)p(y)$ – שכאמור מתפלג נורמלית, ניתן לשער בעזרת הנראות מרבית (MLE). נסמן את הפרמטרים של המודל ב- $\theta = \{\mu_1, \Sigma_1, \dots, \mu_K, \Sigma_K\}$, ונקבל:

$$\theta_{MLE} = \arg \max_{\theta} p(x|y) = \arg \max_{\theta} \log p(x|y; \theta)$$

$$= \arg \max_{\theta} \log \sum_{i=1}^n p(x_i|y_i; \theta)$$

ניתן לפרק את הסכום לפי ה-label של כל דגימה:

$$= \arg \max_{\theta} \log \sum_{i:y_i=1} p(x_i|y_i = 1; \theta) + \log \sum_{i:y_i=2} p(x_i|y_i = 2; \theta) + \dots + \log \sum_{i:y_i=K} p(x_i|y_i = K; \theta)$$

כעת בשביל לחשב פרמטרים עבור כל y מספיק להסתכל על הדגימות עבורן $k = y$, ככלומר:

$$\theta_{k_{MLE}} = \arg \max_{\theta_k} \log \sum_{i:y_i=k} p(x_i|y_i = k; \theta_k)$$

על ידי גזירה והשווואה ל-0 ניתן לחשב את הפרמטרים האופטימליים:

$$\mu_k = \frac{\sum_{i \in y_i=k} x_i}{\sum_i \mathbb{1}_{y_i=k}}$$

$$\Sigma_k = \frac{\sum_{i \in y_i=k} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i \mathbb{1}_{y_i=k}}$$

ניתן לשים לב שהתוחלת μ היא למעשה ממוצע הדגימות עבורן $k = y$. בעזרה הפרמטרים המשוערים ניתן לבנות את המסויוג:

$$y = \arg \max_k p(y_k | x; \mu_k, \Sigma_k) = \arg \max_k \log p(x|y=k)p(y)$$

$$= \arg \max_k -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log p(y)$$

עבור המקרה בו מטריצת covariance היא אלכסונית, כלומר אין תלות בין משתנים שונים, מתקבל המסויוג הביסיאני הגאוסיאני (תוצאה זו הגיונית כיוון שהמסויוג הביסיאני מניח שבاهינתן סיווג של אובייקט מסוים אין יותר תלות בין המשתנים).

עבור המקרה הבינארי, בו $\{0, 1\} \in y$, מתקבל סיווג בצורה של משווהה ריבועית:

$$y = 1 \Leftrightarrow -\frac{1}{2} \log |\Sigma_1| - \frac{1}{2} (x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1) + \log p(y=1) > -\frac{1}{2} \log |\Sigma_0| - \frac{1}{2} (x - \mu_0)^T \Sigma_0^{-1} (x - \mu_0) + \log p(y=0)$$

נוסף:

$$a = \frac{1}{2} (\Sigma_1^{-1} - \Sigma_0^{-1})$$

$$b = \Sigma_1^{-1} \mu_1 - \Sigma_0^{-1} \mu_0$$

$$c = \frac{1}{2} (\mu_0^T \Sigma_0^{-1} \mu_0 - \mu_1^T \Sigma_1^{-1} \mu_1) + \log \frac{p(y=1)}{p(y=0)} - \log \frac{\sqrt{|\Sigma_1|}}{\sqrt{|\Sigma_0|}}$$

ונקבל:

$$y = 1 \Leftrightarrow x^T a x + b^T x + c > 0$$

וזהו משטח הפרדה ריבועי.

Linear Discriminant Analysis הינו מקרה פרטי של Quadratic Discriminant Analysis, בו מניחים כי מטריצת covariance זהה לכל ה-labels, $\Sigma_k = \Sigma$. במקרה זה מתקבל:

$$\log p(x|y=k)p(y) = -\frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log p(y)$$

הביטוי $(\mu_k - x)^T \Sigma^{-1} (\mu_k - x)$ נקרא מרחק מהלוביס, והוא מבטא את מידת הקשר בין x לבין μ תוך כדי התחשבות בשונות של כל משתנה. למעשה ניתן להסתכל על מסויוג LDA כמסויוג המשיר אובייקט-label המרחק על פי מטריקת מהלוביס הוא הערך קטן. על ידי גזירה והשווהה ל-0 מתקבל השערור:

$$\mu_k = \frac{\sum_{i \in y_i=k} x_i}{\sum_i \mathbb{1}_{y_i=k}}$$

$$\Sigma_k = \frac{1}{n} \sum_i (x_i - \mu_k)(x_i - \mu_k)^T$$

ומסווג המתקבל הינו:

$$y = \arg \max_k p(y_k | x; \mu_k, \Sigma_k)p(y) = \arg \max_k -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log p(y=k)$$

$$= \arg \max_k -x^T \Sigma^{-1} \mu_k + \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log p(y = k)$$

ניתן לסמך:

$$a = \Sigma^{-1} \mu_k$$

$$b = \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log p(y = k)$$

ומתתקבל מטוגן לינארי (ומכאן השם של האלגוריתם):

$$y = \arg \max_k a x^T + b$$

מטוגן זה מחלק כל שני אזורים בעזרת מישור לינארי, כאשר בסך הכל יש K קווים הפרדה. עבור המקרה הבינארי מתקיים:

$$a = \Sigma^{-1} (\mu_1 - \mu_0)$$

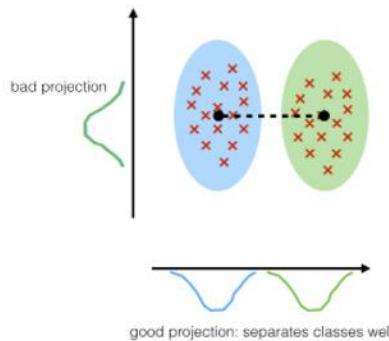
$$b = \frac{1}{2} (\mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1) + \log \frac{p(y = 1)}{p(y = 0)}$$

והסיג הינו:

$$y = 1 \Leftrightarrow a^T x + b > 0$$

אלגוריתם LDA פשוט יותר מאשר QDA יכול שמש פרמטרים לשערק, אך יש לו שני חסרונות עיקריים – הוא לא גמיש אלא לינארי, ובנוסף הוא מניח שטחית ה covariance (labels, covariation) מה שיכל לגרום לשגיאות בסיג. כדי להתמודד עם הבעיה השנייה ניתן להשתמש באלגוריתמים המנסים למצאו את מטריצת covariance- שתביאו לשגיאת הטוב ביותר האפשרי (למשל Oracle Shrinkage Approximating Ledoit-Wolf estimator).

באופן גרפי ניתן להסתכל על אלגוריתם LDA כמציאת כיוון ההפרדה בו יש את השונות הגדולה ביותר בין שתי התפלגיות נורמליות, ובנוסף יש בו את ההפרדה המקסימלית בין הקבוצות השונות. לאחר מציאת הקו האופטימלי ניתן לחשב את התפלגיות של הקבוצות השונות כהתפלגיות נורמליות על הישר המאונך לקו ההפרדה:



איור 2.5 אלגוריתם LDA באופן גרפי: מציאת הכיוון של התפלגיות והטלת המידע על הציר האנכי לכיוון ההפרדה.

2.1.5 Decision Trees

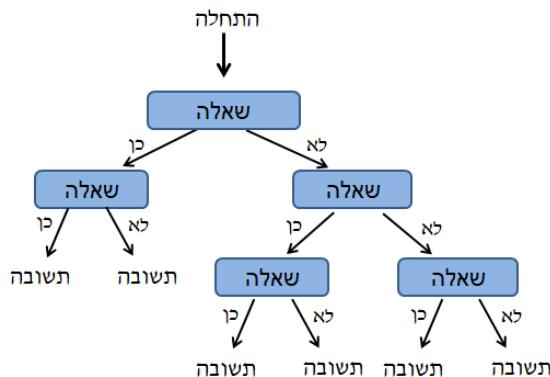
1. הקדמה

עוז החלטה הינו אלגוריתם לומד היכול לשמש הן לביעות סיווג והן לביעות רגסיה. באופן כללי, עוז החלטה לוקחים את המשתנה שברצוננו להסביר (משתנה המטרה/החיזוי), ומחלקים את המרחב שלו לקבוצות (segments). לכל קבוצה של סט האימון מחשבים "ממוצע" (Mean) או "שכיח" (Mode), וכאשר מתקבלת תצפית חדשה משייכים אותה לקבוצה בעלת הממוצע או השכיח הדומה ביותר. לאחר מכן הסיווג לקבוצות השונות יכולם להציג בצורת עץ, אלגוריתם זה נקרא "עוז החלטה".

הגישות השונות בעוז החלטה פשוטות וឥדיות להבנה, אולם הן לא מצלילות להתחרות במדדי הדיקוק של מודלים אחרים של Supervised Learning. לכן, בפרקם הבאים נציג שיטות ensembles, בהן מתבצעת בנייה של כמה עוז-החלטה במקביל, אשר משולבים בסופו של דבר למודל יחיד. ניתן להראות ששימוש במספר גדול של עצים

יכול לשפר דרמטית את מדרדי הדיק של המודל. עם זאת, ככל שימושים נוספים ביותר עיצים כך יכולת הפרשנות של המודל נהיית מורכבת יותר ופחות אינטואיטיבית לצופה שאין מעורב במבנה המודל (למשל גורם עסק'י בארגון שאנו מועוניים להסביר לו את תוצאות המודל).

ראשית נקבע מבנה בסיסי של עץ ונגיד עבורי את המושגים הרלוונטיים:



איור 2.6 דיאגרמה של עץ החלטה.

על מנת להבין את מבנה העץ וליצור שפה משותפת נציג את השמות המקובלים בעבודה עם עצים:

- Root (שורש) – נקודת הכניסה לעץ (חלקו העליון ביותר של העץ).
- (צומת) – נקודת ההחלטה/פיקול של העץ – השאלות.
- (עלים) – הקצות של העצים – התשובות. נקראים גם terminal nodes.
- (ענף) – חלק מתוך העץ המלא (תת-עץ)
- Depth (עומק) – מספר ה-nodes במסלול הארוך ביותר בעץ.

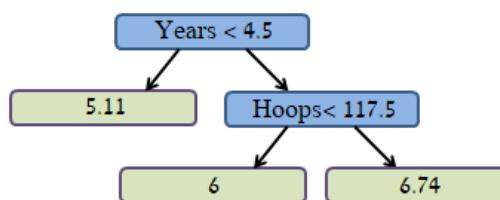
יסודות של עצי החלטה

עצים רגסיה:

כאמור, עצי החלטה יכולים לשמש הן עבור סיווג והן עבור רגסיה, ונתחיל עם דוגמא פשוטה לבניית עץ עבור בעיית רגסיה – חיזוי שכר (באלפי שקלים) שחקני כדורסל באמצעות עצי החלטה. נרצה לחזות את השכר של שחקן בהתאם על הנתונים הבאים:

- שנים - מספר העונות שאותו שחקן שיחק בリגת העל.
- סלים - מספר הסלים שהוא קלע בשנה הקודמת.

תחילה נסיר תכיפות ללא ערכים עבור המשתנה המוסבר שלנו, הלא הוא "שכר" ובנוסף נבצע על אותו משתנה Log transform בכך שהוא ככל הנין בקירוב להטפלגות נורמלית (עקומת גאוס/פעמוני).



איור 2.7 דוגמא של עץ החלטה עבור בעיית רגסיה של עץ החלטה.

כאמור, האירור מתאר עץ המנסה לחזות את Log השכר (באלפי שקלים) של שחקן בהינתן מספר עונות הותק שלו וכמויות הסלים שהקלע בעונה הקודמת. ניתן לראות שהחלק העליון של העץ (Root) מחלק ל-2 ענפים. הענף השמאלי מתייחס לשחקנים בעלי ותק של יותר מ-4.5 שנים ($years < 4.5$), והענף הימני מתייחס לשחקנים פחות ותיקים. לעץ יש 2 צמתים (=שאלות/Internal nodes) ו-3 עלים (=תשבות/Terminal nodes). המספר בכל עלה שבתחלת העץ הוא הממוצע של כל התכיפות ששווגו לעלה זהה. לאחר שבנית העץ הסט'ימה, כל תכיפות חדשה

שתסועג לעלה מסוים תקבל את הערך הממוצע של התצפויות שלל בסיסם נבנה העץ. בהמשך הפרק יסביר כיצד לבנות את העץ וכייד לקבוע את כללי הפייצול בהתאם לדאטה הקיימ.

בסופו של דבר העץ מחלק את השחקנים לשלווש קבוצות:

- שחוקנים ששחיקו 4 עונות או פחות:

$$R_1 = \{X | Years < 4.5, Hoops\} = 5.11$$

- שחוקנים ששחיקו 5 עונות או יותר וקלעו פחות מ- 118 סלים בעונה שחלפה:

$$R_2 = \{X | Years > 4.5, Hoops < 117.5\} = 6$$

- שחוקנים ששחיקו 5 עונות או יותר וקלעו יותר מ- 118 סלים בעונה שחלפה:

$$R_3 = \{X | Years > 4.5, Hoops > 117.5\} = 6.74$$

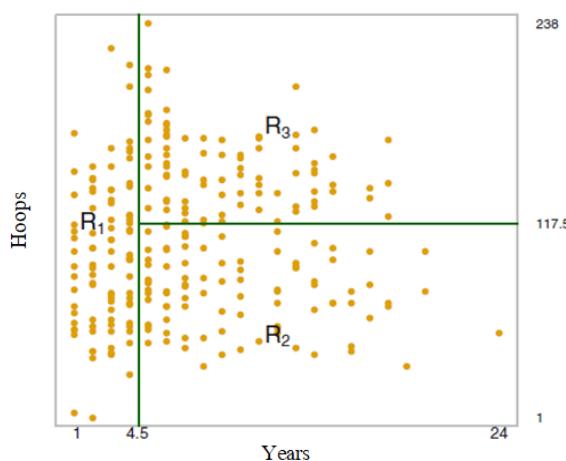
בתואם לדוגמה של העץ, הקבוצות R_1, R_2, R_3 מכונות בשם nodes terminal, או "עלים" של העץ.

אפשר לפרש את עץ הרגרסיה שבדוגמה הצורה נוספת: $Years$ הוא הפרמטר המרכזי ביותר בקביעה מה יהיה גובה השכר, וחקוקנים עם פחות שנות ניסיון ירוויחו פחות משחקנים מנוסים יותר. עבור שחוקן יחסית חדש (עד 5 שנים בלבד) הפרמטר של כמות הסליםאותה הוא קלע בעונה הקודמת מתברר כפחות משפייע על השכר. לעומת זאת, ככל מר, כל עוד לשחקן אין 5 שנות ניסיון, כמות הסלים היא יחסית שלית ביחס לחומר הניסיון עבור קביעת שכרכ. לעומת זאת, בקרוב לשחקנים עם 4.5 שנות ניסיון או יותר, כמות הסלים שהם קלעו בשנה הקודמת כן משפיעה על השכר, וחקוקנים שקלעו הרבה סלים בשנה הקודמת כנראה ירוויחו יותר כסף מאשר שחוקנים עם אותן ניסיון אך קלעו פחות סלים.

עץ הרגרסיה שהובא בדוגמה מפשט קצת "יתר על המידה" את הקשר "האמת" בין $Salary$, $Years$ ו- $Hoops$, והיחסיו של העץ לא מספיק טוב ביחס למודלים אחרים של רגרסיה, כמו למשל רגרסיה ליניארית שתואיר בפרק 3. עם זאת, מודל זה פשוט יותר להבנה ופרשנותו וקל יחסית לייצוג באמצעות גרפים.

2. חיזוי באמצעות ריבוד (Stratification) של מרחב המשתנים:

עד כה הוסבר באופן אינטואיטיבי מהו עץ החלטה וכייד הוא פועל. האתגר המרכזי הוא לבנות את העץ בצורה טובה כך שהיחסיו שלו אכן יהיה תואם למציאות. בכך להבין כיצד בונים עץ בצורה ייעילה, נציג את הדוגמא הקודמת באופן נוספת:



איור 2.8 דוגמא של עץ החלטה עבור בעיית רגרסיה של עץ החלטה.

באיור זה ניתן לראות שהנתונים נפרשו על פני מרחב דו ממדי, וחולקו בעזרת קו ההחלטה בהתאם ל- $nodes$. כתעת נגידיר את תהליך החלוקה והיחסיו בשני שלבים:

1. מחלקים את מרחב הערכים האפשריים של המשתנה אותו רוצים לחזות ל- J אזורים נפרדים $R_j, j = 1, \dots, J$. כתלות בפרמטרים השונים X_n, X_1, \dots, X_J .
2. לאחר החלוקה, כל תצפית שנופלת באזור R_i מקבלת ערך הממוצע של הנקודות מטט האימון שמרכזיות את הקבוצה R_i .

באופן תיאורתי, לכל אזור יכולה להיות כל צורה שהיא. אולם לטובת הפשטות, אנו בוחרים לחלק את המשטנה ל- "אזורים מרובעים". המטרה היא למצוא את האזורים שمبאים למינימום את סכום ריבועי ההפרשיות בין התוויות של הנתונים בסט האימון לבין הערכים של כל אזור. מגד זה נקרא residual sum of squares (RSS), שמשמעותו:

$$RSS = \sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

כאשר \hat{y} הוא ממוצע המשתנים של תציפות האימון באזורי j , $-y_i - \hat{y}_{R_j}$ הוא המרחק בין כל תצפית לבין הערך הממוצע באזורי j . כאמור, המטרה של מגד זה היא למצוא את מקבץ התציפות בכל אזור, כך שריבוע המרחק בין הערך הממוצע לבין כל תצפית יהיה מינימלי. כיוון שבדיקת כל החלוקות השונות האפשריות אינה ריאלית מבחינה חישובית, לרבות משתמשים באלגוריתם חמדן (greedy) אשר עוזב "מלמעלה למטה". גישה זו ביחס לעץ החלטה נקראה "פיזול ביניארי רקורסיבי" (recursive binary splitting), והוא נקראת "מלמעלה למטה" כיון שהיא מתחילה מראש העץ (root), היכן שככל התציפות עדין שייכות לקבוצה אחת גדולה. לאחר סימון נקודת ההתחליה, האלגוריתם מפצל את מרחב הערכים של המשטנה אותו רוצים לחזות, כאשר כל פיזול מסומן באמצעות שני ענפים חדשים בהמשך העץ.

האלגוריתם כאמור הוא חמדן, וזה מתבטא בכך שbullet שלב בתהליכי בניית העץ בוחרים לבצע את הפיזול הטוב ביותר עבור השלב הנוכחי, מבלי להתחשב כמה שלבים קדימה. בגיןה זו יתכן וbullet שלב הנוכחי יש פיזול יותר יעיל לטוויה ארוך, אך הוא לא יבחר אם הוא לא הפיזול האופטימלי בשלב זה. ניתן להמחיש את דרך הפעולה של אלגוריתם חמדן לעמידה בפקק תנועה, ומתרן ניסיון לעקוב את הפקק נבחרת הפניה הראשונה שנראית פעות פוקוקה, מבלי להתחשב בפקק שעשו לבוא בהמשך. כמובן שפניה זו לא בהכרח טוביל לדרך יותר מהירה, ויתכן שבראייה יותר ארוכת טווח דזוקא היה מוטב להימנע מפניה זו כיון שהיא מובילת לפיקק גדול יותר בהמשך.

כדי לבצע את אותו "פיזול ביניארי רקורסיבי", יש לבצע את התהליך האיטרטיבי הבא:

עבור כל פרמטר אפשרי X וקו חלוקה s , נקבל שתי קבוצות:

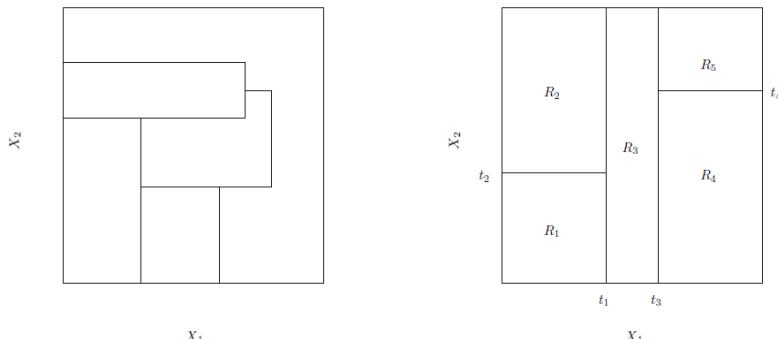
$$\begin{aligned} R_1(j, s) &= \{X | X_j < s\} \\ R_2(j, s) &= \{X | X_j \geq s\} \end{aligned}$$

עבור שתי קבוצות אלה נחפש את הערכים של j ו- s שמבאים למינימום את המשוואה הבאה:

$$\sum_{i: x_i \in R_1(j, s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j, s)} (y_i - \hat{y}_{R_2})^2$$

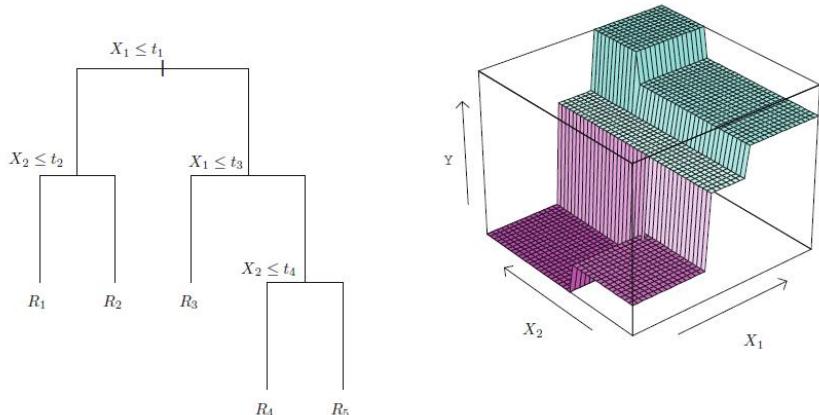
מציאת ערכי s ו- j שמבאים למינימום את המשוואה יכולה להתבצע די בקלות באמצעות הפרמטרים לא גודלה מייד, אך זה תהליך שעלול להיות די סבוך כשייש הרבה פרמטרים.

למעשה תהליך זה מייצר ענף אחד שיצוא מה-root, ולו 2 עליים. בעת ממצאים את התהליךשוב, מתוך מטרה למציאת האלגוריתם הבא שمبיא למינימום את RSS בכל קבוצה, ובכך לקבל 3 קבוצות. ובאופן דומה, נרצה לפצל את אחת מהותן 3 קבוצות שכבר יש לנו כדי לצמצם עוד את RSS. התהליך הזה נמשך איטרטיבית עד שmagics "לקרייטריוון עצייה" מסוים, כמו למשל מספר פיצולים, מספר מקסימלי של תציפות בכל אזור וכו'. אחריו שהתבצעה החלוקה לקבוצות R_1, \dots, R_J , ניתן להפוך את הקבוצות לעץ, ולהשתמש בו כדי לחזות נקודות חדשות.



איור 2.9 חלוקה של משתנים לאזורים.

באילו המצויר ניתן לראות שתי חלוקות – החלוקה הימנית הינה חלוקה דו-ממדית של שני משתנים באמצעות פיצול ביןארי רקורסיבי. החלוקה השמאלית היא גם חלוקה דו-ממדית של שני משתנים, אך היא לא יכולה להוורר באמצעות פיצול ביןארי רקורסיבי, כיוון שהיא מורכבת מידי וaina תואמת את הגישה החמדנית שרצה "חלוקת פשוטה ומהירה".



איור 2.10 תיאור אזרחי חלוקה בעץ ביןארי. מצד שמאל מוצג העץ התואם לחלוקה מהאיור הקודם, מצד ימין ישנה פרספקטיביה רחבה כיצד חיזויים מתבצעים באמצעות העץ.

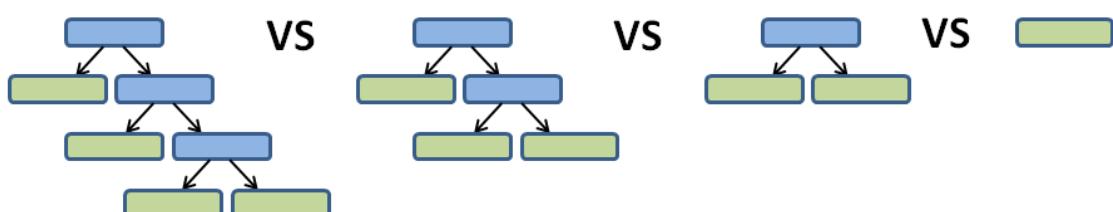
3. גיזום (pruning):

התהlik' שתואר משקף את התהlik' הקלאסי של יצירת עץ גרסיה, והוא מסוגל לנפק חיזויים טובים (מבחינת ממד RSS ושונות נמכה). עם זאת העץ עשוי לבצע התאמת-יתר (Over-fitting) על הנתונים, מה שיוביל לביצועים לא טובים עבור דאטה חדש. בעיה זו עלולה לנבוע מכך שהעץ שנבנה בתהlik' האימון עשוי להיות "מורכב מידי" ומוטאם יתר על המידה לנ庭וני האימון, מה שפוגע ביכולת ההכללה שלו. לעומת עץ מורכב, עץ דליל יותר בעל פחות פיצולים (כלומר פחות קבוצות R_j , ..., R_1) יתרכז בעלי הטיה (Bias) גדולה יותר בהשוואה לאלטרנטיבה הראשונה, אך נראה יוביל לשונות קטנה יותר בין סט המבחן, ובנוסך מודל זה יהיה קל יותר לפרשנותו.

גישה פשוטית בכך שהמודד עם בעיה זו היא לקבוע רף כלשהו, כך שבכל פיצול הירידה ב-RSS תעלה על רף זה. ככלומר, פיצול שימושי לירידה שלoit יחסית RSS-לא יתבצע. באופן זה העצים שיתקבלו יהיו קטנים יותר ויש פחות over-fitting. החיסרונו בגישה זו נועז בקשר שהוא קצרת-رأיה. יתרן מצב בו יש פיצול בעל תרומה קטנה יחסית אך הוא יכול להוביל לפיצול אחר בעל תרומה גדולה,זכה שיביא לירידה גדולה ב-RSS בהמשך. שיטה זו מدلגת על אותו פיצול בעל ערך קטן, מאחר והוא לא עבר את הרף שהוגדר.

גישה אחרת, שמתבגר והיא טובה יותר, הולכת בכיוון הפוך. בשלב הראשון יבנה עץ גדול מאוד (T_0), ולאחר מכן נגוזם ממנו כל מיינן פיצולים בצד להימנע over-fitting. את מקומם של הענפים שהסרנו יתפוז עליה בודד שמקבל ערך ממוצע המתחשב במספר גדול יותר של צפויות. כמובן שצעד זה יגרום לירידה בביטויים על פני סט האימון, אך השαιפה היא שזה ישתלם בבדיקה השגיאה בסט המבחן.

השאלה הגדולה היא כמובן מהי הדרכ הטובה ביותר לגוזם את העץ. אינטואיטיבית, המטרה שלנו היא לבחור subtree מתוך העץ המקורי שימושו לשגיאה הקטנה ביותר. אומדן זה יכול להתבצע על ידי בדיקת השגיאה של העץ החדש ביחס לסט המבחן. כיוון שאינו אפשר לבדוק את כל תתי העצים האפשריים, נהוג להשתמש בשיטה הנקראת subtree pruning (ידועה גם בשם Cost complexity pruning). בגישה זו, במקום להתחשב בכל subtree אפשרי, אנו מסתכלים על רצף מסוים של subtrees שהם למעשה חלקים שונים המרכיבים את T_0 – העץ המקורי שנבנה בהתאם:

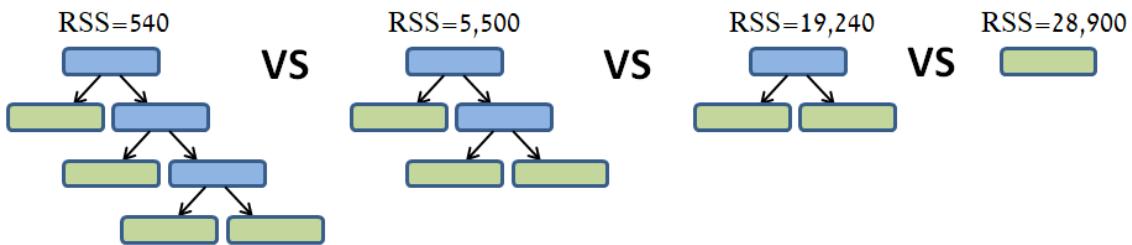


איור 2.11 חיפוש תת עץ אופטימי ביחס ל- T_0 . מתחילה מ- T_0 (העץ השמאלי) וכל פעם מורידים פיצול יחיד עד למגעים ל-root.

cut מהשנים עברו כל אחד מרכז העצים שהתקבל את RSS שלו. לפני שנסביר באיזה עץ לבחור, נסכם את השלבים שבוצעו עד כה:

- בנית עץ רגסיה גדול מכל הדאטה (ולא רק מסט האימון) בגישה ה- recursive binary splitting (הגישה החמדנית שתוארה לעיל). העץ ימשיך להבנות עד קритריון עצירה מסוים (כמו למשל – הגיע למינימום של צפיפות). עץ זה יסומן על ידי T_0 .
- כל עלה בעץ מקבל את הערך הממוצע של צפיפות הפורט שבאותו עלה, ועבור כל עלה פיצול מחושב ה-RSS.
- סכום כל ערכי RSS שקיבלו בכל העליים. הסכום שהתקבל הוא RSS של כל העץ T_0 .
- cut מביצים את שלבים א-ג ל-subtree הבा ברצף. זהו עץ-משנה שכמעט זהה לעץ המקורי, למעט שני עלים שנגזרים מהעץ המקורי. כך חוזרים על התהליך עבור כל subtree, עד ל-subtree האחרון שמורכב רק מה-root של העץ המקורי.

התוצר של השלב האחרון הוא RSS לכל subtree העצים. ניתן לבדוק בכך כי בכל פעם שענף מסוים הוסר, RSS שהתקבל נהיה גדול יותר לעומת subtree המקורי. תוצאה זו הגיונית, שהרי כל פיצול במקור נועד להקטין את השגיאה באמצעות הקטנת RSS, ואילו גיזום עשו את ההיפך – הוא נועד למנוע over-fitting, גם במחיר של שגיאה גדולה יותר.



איור 2.12 ביצוע גיזום וחישוב RSS לכל תת עץ (subtree) שהתקבל.

cut יש לבחור אחד אחד מה-subtrees, ואומר זה נעשה בגישה cost complexity pruning. גישה זו משקלת עבור כל cut את מד RSS שלו יחד עם מספר העלים בעץ. באופן פורמלי:

$$\text{Tree score} = \text{RSS} + \alpha T$$

כאשר T הוא מספר העלים בעץ (Terminal nodes) ו- α הוא פרמטר רגולריזציה הקובע את היחס בין מספר העלים לבין מד RSS. פרמטר זה מחושב באמצעות cross-validation trade-off בין הרצון להגדיל את העץ ובכך להקטין את RSS. RSS בין הרצון להימנע עד כמה שניתן מ-overfitting. המוחזר αT מכונה Tree Complexity Penalty, ואומרו הוא נועד "לפצות" על ההפרש במספר העלים שבין subtrees השונים.

משלב ד' כבר קיבלנו RSS לכל subtree, וcut נשאר רק לחשב לכל אחד מהם את ה-Tree score.

נשים לב כי:

- כאשר $\alpha = 0$, העץ המקורי (T_0) יהיה בהכרח בעל Tree score הנמוך ביותר, כיוון שכאשר $0 = \alpha$ אין כל הריביב αT בנוסחה שווה ל-0. במקרה זה ה-Tree score יהיה לפחות למדד RSS:

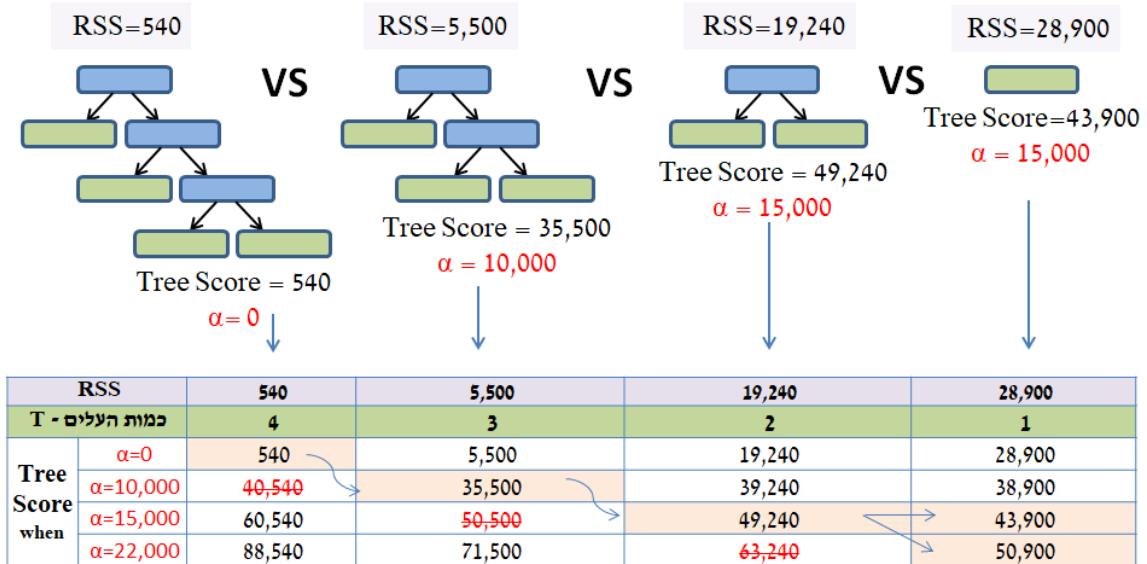
$$\text{Tree score} = \text{RSS} + 0 \cdot T = \text{RSS}$$

ממילא שאר שלבי החישוב מיותרים, כי אנחנו כבר יודעים שהעץ T_0 הוא בעל RSS הנמוך ביותר מכל subtrees, וכן בעל Tree Score הנמוך ביותר. לכן, נרצה לקבוע $0 < \alpha$. נתבונן מה קורה עבור ערכי α שונים:

- עבור $\alpha = 0$ הציון של T_0 יהיה 40,540. וכך גם ל-2 עליים ולקבל עבור אותו α ציון נמוך יותר מ-40,540 (העץ השני משמאלי עם ציון של 35,500). ושוב, אם נשאר ב- $\alpha = 0,000 = \alpha$ ובמקביל נציג 2 עלים נוספים (אנו כעת בעץ השלישי משמאלי), נקבל ציון של 39,240 – שזה עדיין גבוה יותר מאשר 35,500. וכך זה לא טוב לנו.

עבור $\alpha = 15,000$ ניתן לראות שהציון המתקבל יהיה נמוך יותר מה-subtree הקודם. כעת נשים לב שבמעבר ל-subtree האחרון (ה-root) לא משנה אם α יגדל או ישאר אותו דבר, בכל מקרה הציון של ה-root יהיה נמוך יותר מה-subtree הקודם.

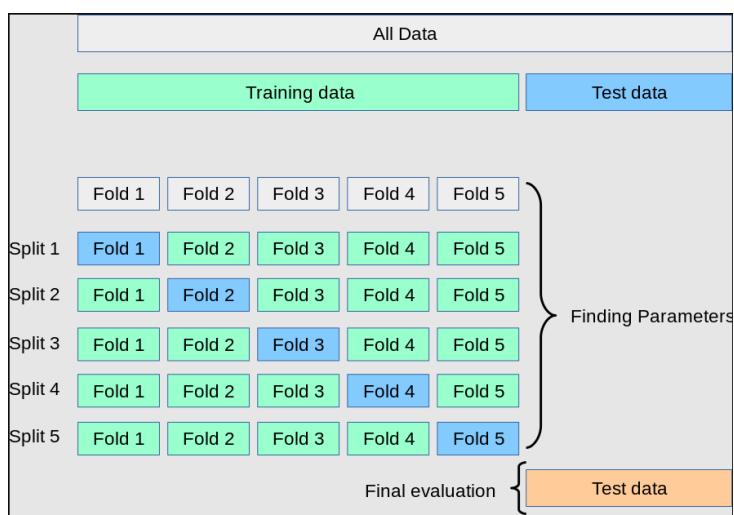
למעשה חזרנו על אותם צעדים עד שאין יותר מה לגוזם. בסופו של תהליך ערכיהם שונים של α יתנו רצף של עצים, מעץ מלא ועד לעלה בודד. יוצא מכך של ערך של α קיים subtree מתאים $T_0 \subset T$ כך שה-tree score המתקבל יהיה קטן ככל האפשר.



איור 2.13 חישוב ערך α מותאם לכל עץ משנה כך שיתקבל Tree score קטן ככל האפשר.

icut, נבחר ב-base score העץ הנמוך ביותר – כולומר העץ השני משמאלי עם ציון של 35,500. במקביל יש לזכור מהם ערכי α של העצים השונים שהתקבלו (ב顺序 הקודם), כיוון שהם יהיו שימושיים לחישוב ערך α אופטימלי:

- o ערכי α חשובים, כיוון שלכל α עשוי להתקבל עץ אחר בעל ציון מינימלי. עד לשלב זה העץ נבנה ביחס **לכל הדadata** (בלי חלוקה ל-Test ו-Train), אך המטריה היא לבחון מהו ערך ה- α שייתן את התוצאה האופטימלית בעזרת העץ הגוזם עבור DATA חדש.
- ט. כדי לבחור את ערך ה- α האופטימלי ניתן להשתמש ב-cross-validation. לשם כך, יש לקחת את הדadata המלא (ממנו נבנה העץ הראשון – T_0), ולחילק אותו באופן הבא:

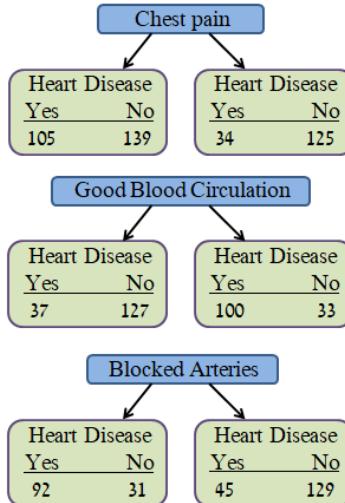


איור 2.14 $K_{fold} = 5$ ב-cross-validation 2.14

- ב-cross-validation המודל מתאמן תחילה לפי 1 split (איור 2.14 לעיל) על התצפויות שבחלקים הירוקים, ובוחן את החיזויים על סט התצפויות הכהול.
 - התהילך זהה נעשה K פעמים, כאשר בכל איטרציה האימון נעשה את התצפויות שבחלקים הירוקים, ובוחינת החיזויים (וחישוב RSS) נעשה על התצפויות שבחלק הכהול.
 - . בכל אחד מה-splits ב-cross-validation המודל משתמש רק ב-data Training כדי לבנות עץ מלא (נסמן אותו הפעם ב- T_1) ורץ (sequence) חדש של subtrees שמבאים לMINIMUM את-h-Tree Score (אותו סדר פעולה כמו בסעיפים א'-ד'), בעזרתו ערכיו α שהתקבלו בסעיף ז'.
 - א. כעת יש לחשב את RSS לכל subtree באמצעות שימוש ב-data Test בלבד, ולזכור מיהו-h-Subtree שקיבל את RSS הנמור ביותר. נניח שבאיטרציה הראשונה העץ שקיבל את RSS הנמור ביותר ב-Testing data הוא דוקא העץ שבו $\alpha = \alpha$.
 - ב. את התהילך של סעיפים יי' יש לבצע K פעמים – פעם אחת עבור כל split, כאשר בכל איטרציה האימון מtabצע על התצפויות שבחלקים הירוקים, ובוחינת החיזויים (וחישוב RSS) מtabצע על התצפויות שבחלק הכהול.
 - ג. בסופו של התהילך, כל איטרציה תניב subtrees בעלי RSS מסוים, וכן ערכיו α שנקבעו כבר מראש בסעיף ז'. לאחר K האיטרציות יש לבדוק מיהו העץ עם RSS המינימלי מבין כל האיטרציות, ומהו ערך α של העץ הזה, וזה יהיה הערך הסופי של α .
 - ד. לבסוף, יש לחזור לעץ המקורי T_0 וה-subtrees שנבנו מה-set data המלא (שמכיל גם את-h-Tree וגם את-h-Test), ולבחר את העץ שתואם לערך α הנבחר. ה-h-subtree הזה יהיה "העץ הגוזם הסופי" שיבחר.
- לסיכום:**
- אחרי כל החישובים מצאנו שהעץ T_0 הוא בעל RSS הנמור ביותר, אך יתכן והוא יסבול מ-Overfitting.
 - כדי לפצות על כך, נוסף רכיב שנועד להתחשב גם ביתר העצים שהווים בעלי RSS גבוהה יותר, ו- "מעניש" את העץ המקורי (T_0) עקב ריבוי העלים שבו.
 - באופן הזה התקבל ציון המאפשר להשוות בין העצים ולבחור את העץ הטוב ביותר ביותר. העץ הזה מהווה מעין איזון בין הרצון ל RSS מינימלי לבין שונות נוכחית בין Train-ל-Test .
 - בכדי למצוא את α האופטימלי, שמצד אחד נותן עץ בעל RSS אופטימלי ומצד שני נמנע כמה שניתן מ-cross-validation Overfitting.

4. עץ סיווג

עץ סיווג ד' דומה לעץ רגרסיה, רק שהמטרה היא שונה – במקומם לקבל תשובה כמותית כמו ברגרסיה, עץ סיווג יtan תווית (label) לתצפית המבוקשת. כאמור לעיל, עץ רגרסיה מספק חיזוי לתצפית מסוימת בהתאם לערך המומוצע של תצפויות האימון ששhicות לאותו node terminal. בעץ סיווג לעומת זאת, כל תצפית תשאיר לקבוצה (class) בעל תווית משותפת. למשל, נניח ומעוניינים לסוג מטופל מסוים האם יש לו מחלת לב או לא. אנו יכולים לבנות עץ החלטה על בסיס מאפיינים של חוליות שאובחנו בעבר ואנו יודעים להגיד מי מהם באמת חוליה לב ומ' לא, ועל בסיס העץ הזה להחליט עבור כל מטופל חדש האם הוא דומה במאפיינים שלו למטופלים שאובחנו בעבר חוליה לב או לא. קר שההתשובה שהעץ נותן היא לא "ערך ממוצע" כמו שראינו בעץ רגרסיה, אלא פשוט החלטה - "כן חוליה לב" או "לא חוליה לב". מלבד החיזוי של התווית, עץ סיווג מספק גם יחסים בין הקבוצות השונות בקרבת תצפויות האימון שנoplים באותו אזור. נתבונן בדוגמא שתמחיש את העניין:



איור 2.15 השפעת פרמטרים שונים על הסיכוי לחולות במחלה לב.

באיר לעיל ניתן לראות שלושה פרמטרים (שבמקרה זה הם סימפטומים של מטופל) בעזרתם מנסים לסוג האם למטופל יש מחלת לב או לא. כפי שניתן לראות אף אחד מהמשתנים אינו יכול לענות על שאלת זו בפני עצמו, כיון שבאף אחד מהמעלים אין איחודות בתכפיות. משتنים כאלה, אשר אינם יכולים בפני עצם לספק סיווג מושלם, נקראים משתנים לא הומוגניים (impure) – לא טהורם). כיון שברוב המקרים כל המשתנים אינם הומוגניים, יש למצאו דרך כיצד לבחור באחד מהמשתנים להיות המשטנה שבראש העץ (Root node). כמובן, יש ליצור מدد הבוחן ומשווה את רמת ה-*impurity* של כל משתנה. ישנו מספר מדדים, ונתמקד בשניים מהם – Entropy ו-Gini index.

א. ממד Gini index :

נסמן את ההסתברות לשירותתו מסוימת לקבוצה j ב- $-r_d$, ונגדיה:

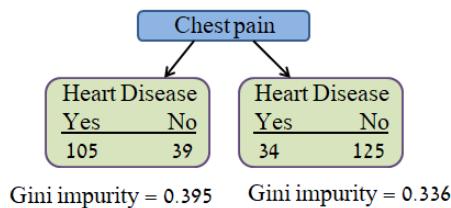
$$Gini = 1 - \sum_{j=1}^J p_j^2$$

באופן אינטואיטיבי, ממד Gini מיצג את הסיכוי לקבלת סיווג שגוי עבור בחירה רנדומלית של נקודה מהדאטה בהתאם לפרופורציות של כל class בדאטה. נדגים זאת על אחד הפרמטרים שבדוגמא הקודמת – pain. עבור הענף הימני מתקיים:

$$Gini = 1 - \left(\frac{34}{34 + 125} \right)^2 - \left(\frac{125}{34 + 125} \right)^2 = 0.336$$

ועבור הענף השמאלי מתקיים:

$$Gini = 1 - \left(\frac{39}{39 + 105} \right)^2 - \left(\frac{105}{39 + 105} \right)^2 = 0.395$$



איור 2.16 חישוב ממד Gini עבור הפרמטר Chest pain.

אחרי שהיחסנו את ה- Gini Impurity לשני העליים, נחשב את ממד Gini הכלול של כל המשתנה Chest Pain. בשבייל חישוב זה יש לאזן בין מספר התכפיות שככל עלה, באופן הבא:

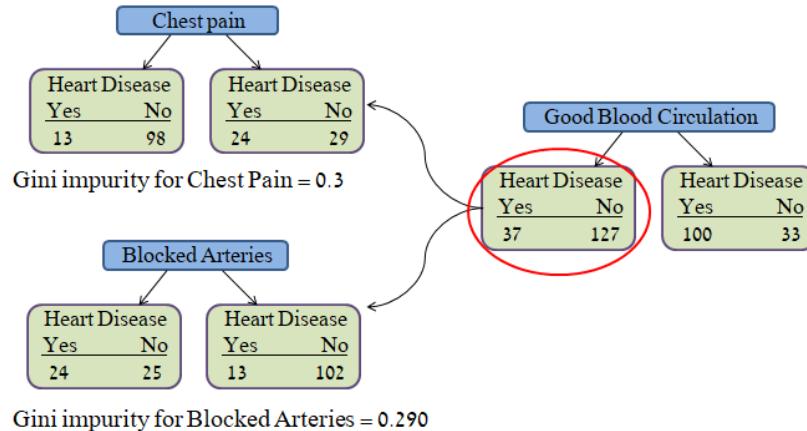
$$\text{Gini impurity for chest pain} = \left(\frac{144}{144 + 159} \right) \times 0.395 + \left(\frac{159}{144 + 159} \right) \times 0.336 = 0.364$$

באופן דומה ניתן לחשב את מדד Gini גם עבור יתר המשתנים ונקבל:

$$\text{Gini impurity for good blood circulation} = 0.36$$

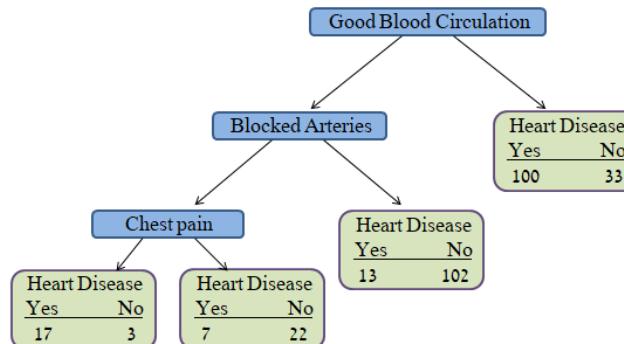
$$\text{Gini impurity for blocked arteries} = 0.381$$

למשתנה Good blood Circulation יש את הציון הכי נמוך, מה שאומר שהוא מסוג הכי טוב את המטופלים עם ובל' מחלת לב, ולכן נשתמש בו כמשתנה המסוג הראשון בראש העץ (ה-root). בצד לקבוע את הפיצול הבא, יש להתבונן כיצד שאר הפרמטרים מסוגים את התכיפות של ה-root. נניח למשל ומתקיים:



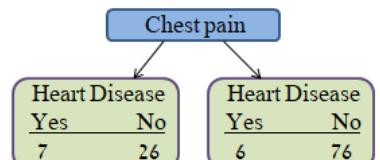
איור 2.17 חישוב מדד Gini עבור שאר הפרמטרים לאחר קביעת ה-root.

כפי שניתן לראות, למשתנה Blocked arteries יש ציון Gini נמוך יותר, ולכן זה שנבחר להיות הפיצול הבא. לבסוף נבחן כיצד הפרמטר האחרון מסביר את התכיפות של הפיצול שלוני, ונקבל את העץ הבא:



איור 2.18 חישוב מדד Gini עבור פיצול לפי הפרמטר Chest pain ביחס לשאר העץ.

נשים לב שניתן לפצל גם את הולה 13/102 לפי הפרמטר Chest pain, באופן הבא (המספרים כמפורט בתוצאות האמיתיות):



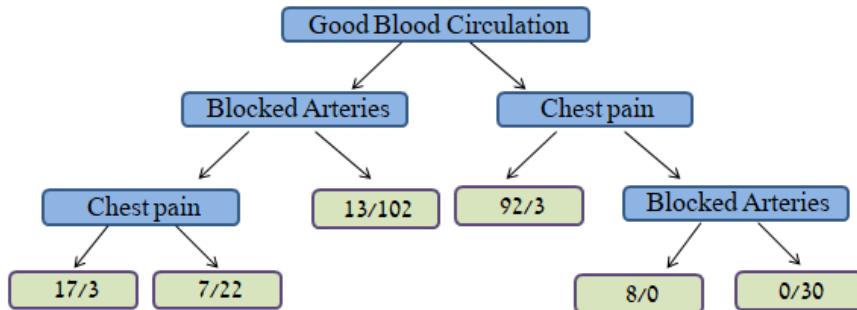
$$\text{Gini impurity for Chest Pain} = 0.29$$

איור 2.19 חישוב מדד Gini עבור פיצול לפי הפרמטר Chest pain ביחס לעלה 13/102.

מדד Gini שהתקבל הינו 0.29, בעוד שבלי הפיצול הממד של העלה היה 0.2, ולכן במקרה זה עדיף להשאיר אותו כפי שהוא לפני ניסיון הפיצול.Cut-but בואנו נבנה גם את הענף ימני של העץ:

1. נחשב את מדדי Gini.
2. אם לענף הקימ שצין Gini נמוך יותר, אז אין טעם לפצל עוד, והוא הופך להיות terminal node.
3. אם פיצול הענף הקימ מביא לשיפור, בוחרים את המשטנה המפצל בעל ציון Gini הנמוך ביותר.

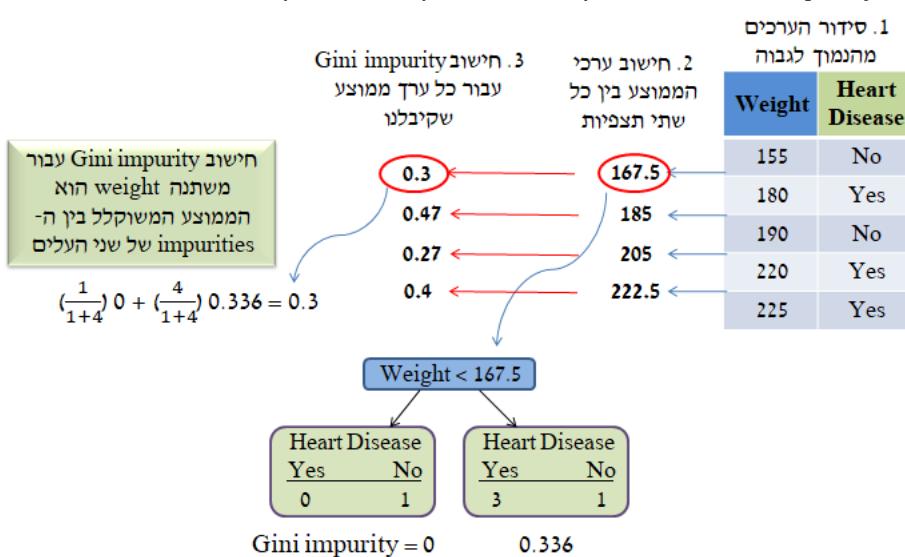
ע"ז טיפוסי לאחר סיום התהילה נראה כך:



איור 2.20 חישוב מדדי Gini עבור פיצול לפי הפרמטר Chest pain ביחס לעלה 13/102.

מדד Gini שהתקבל הינו 0.29, בעוד שבלי הפיצול הממד של העלה היה 0.2. התהילה שתואר מטהים בהם הפרמטרים מתפצלים באופןBINARI, כלומר הפיצול של כל פרמטר נקבע על ידי שאלה שעליה יש תשובה של כן או לא. במקרה בהם ישנים משתנים רציפים, הפיצול דורש כמה שלבים מקדימים:

1. סידור ערכי הפרמטרים מהערך הנמוך ביותר לערך הגבוה ביותר.
2. חישוב הערך הממוצע בין כל 2 תציפות.
3. לחישוב Gini impurity עברו כל ערך ממוצע שהתקבל בשלב הקודם.

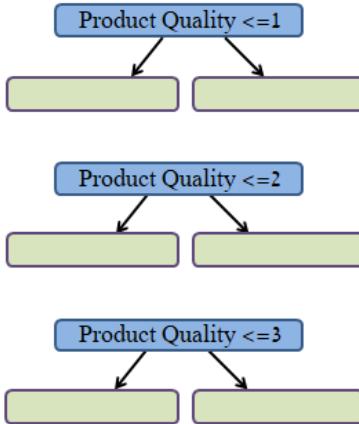


איור 2.21 פיצול פרמטרים רציפים לפי מדדי Gini.

אנחנו מקבלים את ה-Gini הנמוך ביותר מתי שאנו חנו קבועים weight<205 threshold של. אז זהו ה-cutoff וערך ה-Gini שנשתמש בהם כשאנו ממשווים את המשטנה weight ליתר המשתנים. עד עכשיו דיברנו על איך מחלקים משתנה רציף ואיך מחלקים משתנה BINARIO (שאלות כן/לא).Cut-but בואנו נבנה בו את ה-Gini העדיף (מכל ערכיהם: אדום, ירוק, כחול וכו').

משטנה מדורג מאוד דומה למשטנה רציף, חוץ מזה שאנו צריכים ליחס בו את הציון עבור כל חלוקה אפשרית.

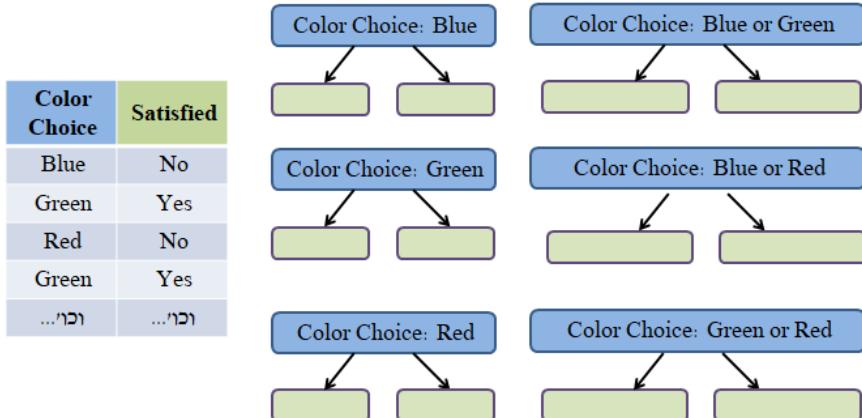
Product Quality	Satisfied
1	No
1	Yes
3	No
1	Yes
... וכו'...	... וכו'...



שימוש לב: אנחנו לא צריכים לחשב את ה-
impurity score ל-
prod. quality <=4
שזה יכלול בעצם את כלם

איור 2.22 פיצול פרמטרים קטגוריאליים לפי מדד Gini.

כשיש משתנה קטגוריאלי עם כמה קטגוריות, אפשר לחשב את ציון ה-Gini עבור כל קטגוריה, כמו גם עבור כל קומבינציה אפשרית:



איור 2.23 פיצול פרמטרים קטגוריאליים לפי מדד Gini.

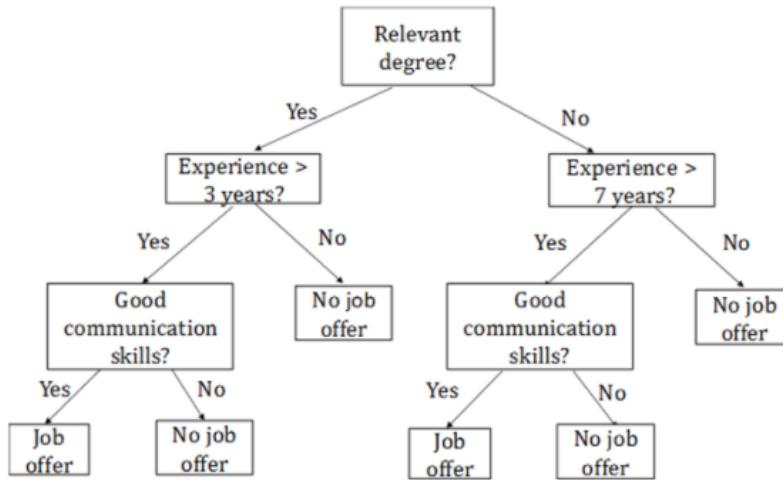
ב. מדד Entropy & Information Gain

מדד נוסף לפיצול צמתי העז מtabased על האנתרופיה של העלים, בעזרתה ניתן לבחון את ה-information gain המקיים מכל פיצול. אנטרופיה באה לממדוד את השגיאה של התפלגות המשתנה הנבחן מול משתנה המטרה. נניח וישנו n תוצאות אפשריות, כל אחת מהן בעלת הסתברות p_i , אז האנתרופיה מוגדרת באופן הבא:

$$H(x) = - \sum_{i=1}^n p_i \log p_i$$

בדומה למדד Gini, הפיצול האופטימלי נבחר על ידי המשתנה בעל מדד האנתרופיה הנמוך ביותר. אם כל התוצאות בעלי מסוים מסוימות לאוטו-class, אז מדד האנתרופיה יהיה 0. מайдך, כאשר בעלי מסוים יש התפלגות שווה בין 2-class-ים של המשתנה המוסבר, מדד האנתרופיה יהיה 1 (שהה ערך המקיים שמדד אנטרופיה יכול לקבל).

לעיל פירטנו שלב אחריו שלב את חישוב מדד Gini ל-Use-Case של חולץ לב. כעת ניקח דוגמא אחרת פשוטה יותר הנוגעת לקבלת מועד לתפקיד מסוים, כאשר מטרת העז היא להחזיר "Yes" אם רוצים לקבל את המועד, אחרת "No".



איור 2.23 עץ סיווג עבור קבלת מועדן לתקפיך מסוים.

הדוגמא מתארת את אחד המאפיינים העיקריים של עצי החלטה – ההחלטה מתמקדת על ידי הסתכילות היררכית על הפרמטרים השונים, כאשר בכל פעם מתמקד בפרמטר אחד ולא על כולם בבבב אחת. תחילה, נלקח בחשבון המאפיין החשוב ביותר (תוואר רלוונטי במקורה שלפנינו), לאחר מכן נלקחים שנות הניסיון, ורק הלאה. נניח שהסיכוי בקרוב כלל העובדים לקבל עבודה הוא 20%, והסתוכוי לא לקבל עבודה הוא 80%. במקרה זה האנטרופיה תחשב באופן הבא (הלוגריתם יכול להיות מחושב בכל בסיס, פה נלקח הבסיס הטבעי):

$$H = -0.2 \ln 0.2 - 0.8 \ln 0.8 = 0.5004$$

כעת נניח בនוסף ש-30% מהמועמדים בעלי תואר רלוונטי מקבלים הצעת עבודה ואילו מtower אלה שאינם בעלי תואר רלוונטי רק 10% מקבלים הצעת עבודה. האנטרופיה עבור מועדן בעל תואר הינה:

$$H = -0.3 \ln 0.3 - 0.7 \ln 0.7 = 0.61$$

ועבור מועדן ללא תואר רלוונטי בתחום:

$$H = -0.1 \ln 0.1 - 0.9 \ln 0.9 = 0.32$$

נניח ומדוברים מתפלגים באופן שווה בין בעלי תואר לכלה שאינם בעלי תואר, כלומר ל-50% מהמועמדים יש תואר רלוונטי ול-50% אין, הרי שתווחלת האנטרופיה במקורה זה הינה:

$$\mathbb{E}_{H(x)} = 0.5 \times 0.61 + 0.5 \times 0.32 = 0.46$$

לאחר כל החישובים, נוכל לבחון את רמת *impurity* שמתיקבלת מהידיעה האם למועדן מסוים יש תואר רלוונטי או לא. כאמור, כמה הידיעה שלמועדן מסוים יש תואר רלוונטי מפחיתה מחוור ה odds שלו לקבל את המשרה. אם אי ה odds נמדדת באמצעות מדד ה-*Entropy*, Entropy, הרי שהרווח מהמידע הוא:

$$\text{Information gain} = 0.5004 - 0.4680 = 0.0324$$

הן עבור מועמדים בעלי תואר והן עבור כלה ללא תואר, המשטנה שמקסם את הרוחות מהמידע הצפוי (ירידה באנטרופיה הצפוי) הוא מספר שנות הניסיון. כאשר למועדן יש תואר רלוונטי, הסוף עבור "שנות ניסיון" הממקסם את הרוחות מהמידע הצפוי הוא 3 שנים. עבור הענף התואם למועדן שאין לו תואר רלוונטי, הסוף של שנות ניסיון הממקסם את הרוחות מהמידע הצפוי הוא 7 שנים. לפיכך, שני הענפים הבאים הם: "nisyon < 7" ו- "nisyon ≥ 7". באותו הרגע בונים את יתר העץ.

Misclassification rate

לאחר בניית עץ הסיווג, יש לבדוק את רמת הדיווק שלו על דатаה חדש. בעץ רגסיה זה נעשה בעזרת מדד RSS ובבעיות סיווג מקובל למדוד את *misclassification rate*. מדד זה בא לכמת את היחס בין כמות התוצאות שהמודול סיווג באופן שגוי לבין כמות ההצלחות של התוצאות. במקרה זה פונקציית המבחן תהיה:

$$\mathcal{L}(\tilde{y}, y) = I\{\tilde{y} \neq y\}$$

פונקציית מחיר זו נקראת zero-one loss, כאשר תחת פונקציה זו חישויים נכונים יקבלו ציון 0 ושגיאות יקבלו ציון 1, ללא תלות בגודל השגיאה. פונקציית ה misclassification rate נראית כך:

$$R(h) = \mathbb{E}[I\{h(x) \neq y\}]$$

סיכום

עż החלטה (Decision Tree) הינו אלגוריתם לשיווג או לחיזוי ערכו של משתנה, כאשר המאפיינים מסודרים לפי סדר החשיבות. לצורך סיווג, קיימים שני מדרדים אלטרנטיביים לאז וודאות: ממד אנטרופיה (Entropy) וממד ג'יני (Gini). כאשר ערכו של משתנה מסוים נחזה, אי הוודאות נמדדת באמצעות RSS. חשיבותו של מאפיין הינו הרוח מההמיעץ הצפוי שלו (Expected Information Gain). הרוח מההמיעץ הצפוי נמדד על ידי הירידה באז הוודאות הצפואה אשרתרחש כאשר יתקבל מידע אודות המאפיין.

במקרה של חלוקת **משתנה קטgorיאלי**, המידע המתkeletal הינו על פי רוב אוזות קטgorיה (Label) של המשתנה למשל: צבע מועדף (מכיל ערכים: אדום, ירוק, כחול וכו'). במקרה של **משתנה רציף** יש לקבוע ערך סף (Threshold) של המשתנה. ערכי סף אלו נקבעים באופן שמקסם את ה- **אוזן information gain** הצפוי.

אלגוריתם עż ההחלטה קובע תחיליה את צומת השורש (Root Node) האופטימלי של העץ באמצעות קרייטריון "מרקזום הרוח מההמיעץ" שהוגדר לעיל. לאחר מכן הוא ממשיך לעשות אותו הדבר עבור הצמתים הבאים. הקצוות של הענפים הסופיים של העץ מכונים **צמתים עליים** (Leaf Nodes) או **Terminal Nodes**.

במקרים בהם עż ההחלטה משמש לסייע, צמתים העליים כוללים בתוכם את ההסתברויות של כל אחת מהקטגוריות להיות הקטgorיה הנכונה. כאשר עż ההחלטה משמש לחיזוי ערך נומירי לעומת צאת, אז צמתים העליים מספקים את ערך התוצאה של היעד. הגיאומטריה של העץ נקבעת באמצעות סט האימון (Training Set), אך הסטטיסטיקה שעוסקת ברמת הדיק של העץ צריכה כמו תמיד בלמידת מכונה לבוא מtower סט הבדיקה (Test Set) ולא רק מtower סט האימון.

אחרית דבר:

מדד RSS וממד RSS misclassification rate שהוצגו בפרק זה הינם רק חלק מהמדרדים המקובלים. לכל ממד יתרונות וחסרונות, והמדד הרלוונטי יבחר בהתאם לסוג הנתונים והבעיה העסקית.ណון מעט בחזקות ובחולשות של עż ההחלטה:

יתרונות:

- בהשוואה לאלגוריתמים אחרים, עż ההחלטה דורשים פחות השקעה בתהילך הכנת הנתונים (-pre processing).
- עż ההחלטה לא דורש גרמול של הדטה.
- ערכים חסרים בDATA לא משפיעים על תהליך בניית העץ.
- עż ההחלטה תואם לאופן שבו מרבית בני האדם חושבים על בעיה מסוימת והוא פשוט להסביר למי שאינם מומחים.
- אין שום דרישת שהקשר בין המשתנה המוסבר והמשתנים המסבירים יהיה LINEAR.
- העץ בוחר אוטומטית במשתנים הטובים ביותר על מנת לבצע את החיזוי.
- עż ההחלטה רגש פחות לנסיבות חריגות מאשר רגסיה.

חסרונות:

- שינוי קטן נתונים יכול לגרום לשינוי גדול במבנה עż ההחלטה ולגרום לחוסר יציבות.
- לעיתים החישוב בעż ההחלטה יכול להיות מורכב מאוד ביחס לאלגוריתמים אחרים.
- עż ההחלטה לעתים תוצאות דורשים יותר זמן הרצה לאימון המודל, ומשכך מדובר באלגוריתם "יקר" במשאבים.
- האלגוריתם של עż ההחלטה אינו מספק ליישום רגסיה ולניבוי ערכים רציפים.
- עż ההחלטה נתה לעתים תוצאות לנוטות ל-Overfitting.

2.2 Unsupervised Learning Algorithms

2.2.1 K-means

אלגוריתם K-means הינו אלגוריתם של למידה לא מונחית, בו מתבצעת תחזית על נתונים כאשר ה-label אינו נתון. אלגוריתם זה מתאים לביעות של חלוקה לאשכולות (Clustering), ובנוסף יכול לשמש בשלב הצגת וניקוי הנתונים (EDA). עבור כל נקודה במדגם, המודל ממציע את סכום ריבוע המרחקים (WCSS) מכל מרכז אשכול (סנטרואיד - centroid), ולאחר תהליך של התכנסות – נקבעים האשכולות והסנטרואידים הסופיים. מספר האשכולות הנדרש הוא היפר-פרמטר שנקבע מראש. כמו כן האלגוריתם השיכים למידה הבלתי-מוניית, ב-K-means לא מתבצע אימון, ולמעשה התחזית מתבצעת על כל הדadataה הנתון.

סנטרואיד הוא מונח מתחום הגיאומטריה, והוא מתאר את הממוצע האריתמטי של כל הנקודות שמתפרסות על פני צורה כלשהי. באופן אינטואיטיבי ניתן לחשב על סentrואיד נקודות איזון של צורה גיאומטרית כלשהיא, כך שאם נספה להניח צורה, משולש לדוגמה, באופן מסוין, הסentrואיד הוא הנקודה שבה המשולש יתאזן ולא ייפול לאחד הצדדים.

בפועל, סביר שהוצאות איתן מתמודדים במצבים יותר מורכבים מאשר פשוט. במצב זה, הסentrואיד יהיה הנקודה בה סכום המרחקים של כל נקודה באשכול מהסentrואיד יהיה מינימלי. כמובן, המודל ימוך את מרכזו של כל אשכול כך שסכום המרחקים של כל הנקודות מהסentrואיד יהיה נמוך ככל האפשר. למעשה, זהו ההגדרה הבסיסית של K-means: אלגוריתם מבוסס סentrואידים המציג את סכום ריבוע המרחק של כל הנקודות באשכול. מدد זה נקרא WCSS, והוא מدد משמעותי ביותר בקרב אלגוריתמים שמבצעים חלוקה לאשכולות, K-means. הסיבה לחזקה במשווה היא שאנו רוצים להגבר את ההשפעה של המרחק, מעין "עונש" לתוצאות רחוקות מהמרכז.

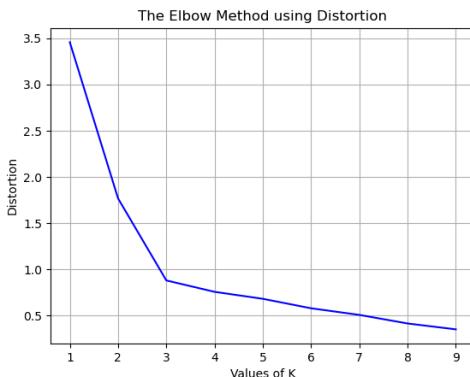
מדד WCSS הוא אחד הדריכים המקבילים ביוטר להעיר את תוצאות החלוקת לאשכולות ב-K-means. היתרון של ממד זה הוא האפשרות לראות את מידת ההצלחה של המודל, כמובן לקבל מספר ממש שמאמת את הצלחת המודל. מנגד, WCSS הוא מסוף ללא תחום מסוים והוא דרוש פרשנות, כיוון שהערך והמשמעות שלו משתנים ממודל למודל. ערך מסוים יכול להיחשב תוצאה טובאה במקורה מסוים, ובמקרה אחר עשויה להיחשב תוצאה רעה מאוד. ניתן להשוות WCSS בין מודלים אך ורק כאשר יש להם את אותן אשכולות ואיתם מספר תוצאות. באופן פורמלי, ערך זה מחושב באופן הבא:

$$WCSS = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$

כאשר K הוא מספר האשכולות, ו- n הוא מספר הנקודות במדגם.

ישנו trade-off בין השאייה למצער את ממד WCSS ובין מספר האשכולות הרצוי: ככל שמספר האשכולות גדול יותר, כך ה-WCSS יקטן. הדבר מתיישב עם הריגיון – פיזור סentrואידים רבים (כלומר, חילקה ליותר אשכולות) על פני הנתונים יוביל לכך שבכרכרה סכום המרחקים של התוצאות מהסentrואידים יקטן או לא ישנה. כיוון שתוצאות המשיכת לסentrואיד הקרוב אליה ביוטר, אם התווסף סentrואיד שקרוב לנקודה מסוימת – ה-WCSS קטן. ואם הסentrואיד רחוק מכל שאר הנקודות במדגם יותר מהסentrואידים הקיימים – חילקת התוצאות לאשכולות לא תשנה, וערך ה-WCSS לא ישנה.

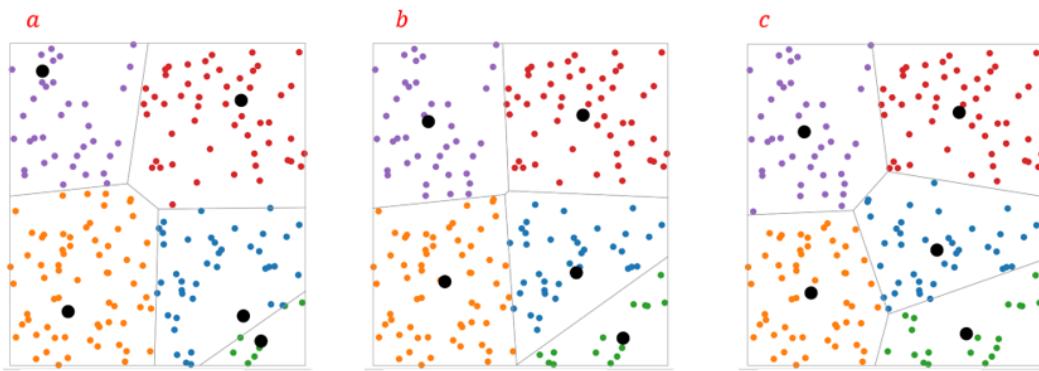
לכן מצד אחד, נרצה לבחור K גדול שימזר את ה-WCSS; מצד שני, הסיבה שהשתמשנו ב-K-means מלכתחילה היא בכך לפשט את הנתונים למספר סביר של אשכולות, כזה שיאפשר לנו לעורך אנליזה נוחה. שיטת המפרק (Elbow method) היא טכניקה שימושת לפתרון סוגיה זו. הרעיון הוא לבחור את ה- K הקטן ביותר שסמן השיפור במדד ה-WCSS הוא מטען במידה סבירה. שיטה זו היא היוריסטיבית ואני דרך חד שמעית לקבוע שה- K הנבחר הוא האופטימלי. בדרך זו ניתן לשכנע מדוע ה- K שנבחר הוא הנכון, אך ההחלטה הסופית נתונה לשיקול דעתו של המשתמש.



איור 2.6 – שיטה היוריסטית למציאת מספר האשכולות האופטימלי. בדוגמא זו ניתן לראות שבמעבר של K מ-2 ל-3 יש ירידת משמעותית בערך של-WCSS. המעבר מ-3 ל-4 לעומת זאת מוביל לשינוי זניח ב-WCSS (וכך גם במערכות הבאים). לכן ניתן להסיק שבמקרה זה בחירה של $K = 3$ הינהבחירה טובה.

כאמור, האלגוריתם מחלק את הנתונים לאשכולות בדרך שמצוירת את סך ריבועי המרחקים של כל תצפית ממרכז האשכול. באופן פורמלי האלגוריתם מתבצע ב-4 שלבים:

- א. **אתחלו:** המודל מציב את הסנטרואידים באופן רנדומלי.
- ב. **שייר:** כל תצפית משוכנת לסנטרואיד הקרוב אליה ביותר.
- ג. **עדכן:** הסentrואיד מוזץ שכר שה-WCSS של המודל יموער.
- ד. חוזרת על שלבים ב, ג עד אשר הסentrואידים לא צדים לאחר העדכון, כלומר יש התכנסות.



איור 2.7 (a) **אתחלו** K -means. (b) **שייר** כל נקודה לסentrואיד הקרוב ביותר אליו, ועדכן הסentrואידים לפי מודד-WCSS. (c) **חוזה על b עד להתכנסות.**

K-means ידוע בכך שהוא אלגוריתם פשוט ומהיר. לרוב, הבחירה הראשונה בפתרון בעיות של חלוקה לאשכולות תהיה ב- K -means. עם זאת, לאלגוריתם ישנו גם חסרונות. ראשית, בחירת- K הנכון עשויה להיות מרבבית המקרים. בנוסף, האלגוריתם רגיש מאוד לערכים קיצוניים (Outliers). באופן הפעולה של האלגוריתם מאפשר לו ליצור אשכולות רק בצורה של ספירות, והדבר אינו אופטימלי בחלוקת מן המקרים.

בעיה נוספת להתייעזר בבחירה המינימום הראשוני של הסentrואידים – כיוון שהבחירה היא רנדומלית, ניתן להיקלע להתקנסות במינימום מקומי שהוא אינו המינימום הגלובלי. כדי להתמודד עם בעיה זה ניתן להשתמש באלגוריתם +++. בשלב ראשון האלגוריתם בוחן למקם סentrואיד אחד באופן רנדומלי. ככל תצפית, האלגוריתם מחשב את המרחק בין התצפית לnearestoid הקרוב אליו ביותר. לאחר מכן, כך יכולת נבחרת להיות הסentrואיד החדש. התצפית נבחרת בהתאם להתפלגות משוקלת של המרחקים, כך שכל שתצפית יותר רוחקה – כך גובר הסיכוי שהיא תבחר. שני השלבים האחרונים נמשכים עד שנבחרו K סentrואידים. כאשר כל הסentrואידים מוקמו, מבצעים K -means עם דילוג על שלב האתחול (השלב בו ממקמים את הסentrואידים). +++ מוביל להתקנסות מהירה יותר, ומוריד את הסיכוי להתקנס לאופטימום מקומי.

2.2.2 Mixture Models

אלגוריתם K -means מחלק n נקודות ל- K קבוצות על פי מרחק של כל נקודה ממרכז מסוים. בדומה ל- K -means גם אלגוריתם mixture model הוא אלגוריתם של clustering, אך במקומם להסתכם על כל קבוצה של נקודות כשייכות למרכז מסוים, המודל מ Shirley נקודות להתפלגויות שונות. המודל מניח שכל קבוצה היא למעשה דגימות של התפלגות מסוימת, וכל הדאטה הוא עריכוב דגימות ממספר התפלגויות. הקושי בשיטה זה הוא האתחול של כל קבוצה – כיצד

ניתן לדעת על איזה דוגמאות לנסות ולמצוא התפלגות מסוימת? עקב בעיה זו, לעיתים משתמשים קודם באלגוריתם K-means על מנת לבצע חלוקה ראשונית לקבוצות, ולאחר מכן למצוא לכל קבוצה של נקודות התפלגות מסוימת.

ראשית נניח שיש k אשכולות, אז נוכל לרשום את ההסתברות לכל אשכול:

$$p(y = i) = \alpha_i, i = 1, \dots, k$$

וכמוון לפי חוק ההסתברות השלמה מתקיים $\sum_i \alpha_i = 1$.

בנוסף נניח שכל אשכול מתפלג נורמלית עם פרמטרים (μ_i, σ_i) , אז נקודה השויכת לאשכול i מקיימת:

$$x|y = i \sim \mathcal{N}(\mu_i, \sigma_i), i = 1, \dots, k$$

אם מגיעה נקודה חדשה ורוצים לשיר אותה לאחד האשכולות, אז צריך למשה למצוא את האשכול i שעבורו הביטוי $p(x|y = i)$ הוא הכי גדול. לפי חוק ביביס מתקיים:

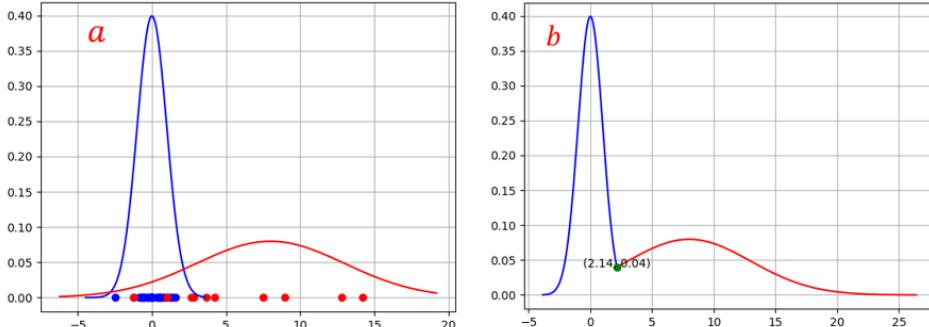
$$p(y = i|x) = \frac{p(y = i) \cdot p(x|y = i)}{p(x)}$$

המכנה למשה נתון, כיוון שההתפלגות של כל אשכול ידועה ונותר לחשב את המכנה:

$$f(x) = f(x; \theta) = \sum_i p(y = i) f(x|y = i) = \sum_i \alpha_i \mathcal{N}(x; \mu_i, \sigma_i)$$

ובסוף הכל:

$$p(y = i|x) = \frac{\alpha_i \cdot \mathcal{N}(x; \mu_i, \sigma_i)}{\sum_j \alpha_j \mathcal{N}(x; \mu_j, \sigma_j)}$$

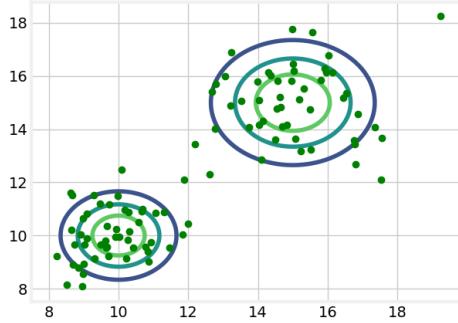


איור 2.8 (a) תערובת של שני גaussיאנים בממד אחד: בשלב ראשון מחלקים את הנקודות לשני אשכולות ומתאיםים לכל אשכול התפלגות מסוימת. במקרה זה אשכול אחד (מוסמן בכחול) הותאם להתפלגות $\mathcal{N}(0, 1)$, ואשכול אחד (מוסמן באדום) הותאם להתפלגות $\mathcal{N}(8, 5)$. (b) נקודה חדשה x תסוג לאשכול הכחול אם $2.14 < x < 2.14 + 2.14 \sqrt{0.04 + 0.04} > 8.5$. באופן דומה, הנקודה x תסוג לאשכול האדום אם $x > 2.14 + 2.14 \sqrt{0.04 + 0.04} < 0.1$.

כאמור, כדי לשיר נקודה חדשה x לאחד מה气colonות, יש לבדוק את ערך ההתפלגות בנקודה החדש. ההתפלגות שבעזרה ההסתברות (x) היא הגדולה ביותר, היא זאת שאליה תהיה משוייכת הנקודה. ההתפלגות יכולות להיות בחד ממד, אך הן יכולות להיות גם במדידות נוספות. למשל אם מסתכלים על מישור, ניתן להתאים לכל אשכול ההתפלגות נורמלית דו-ממדית. במקרה ה- n -ממד, ההתפלגות נורמלית $(\Sigma, \mu) \sim \mathcal{N}(\Sigma, \mu)$ היא בעלת הצפיפות:

$$f_X(x_1, \dots, x_n) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1}(X-\mu)}$$

כאשר $|\Sigma|$ הוא הדטרמיננטה של מטריצת ה-covariance.



איור 2.9 טרורובות של שני גאוסיאנים בדו-ממד: אשלול אחד מתאים לגאוסיאן עם וקטור תוחלות $\mu_1 = [10, 10]$ ומטריצת covariance $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, והאשלול השני מתאים לגאוסיאן עם וקטור תוחלות $\mu_2 = [15, 15]$ ומטריצת covariance $\Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$.

כיוון שהאלגוריתם mixture model מספק התפלגיות, ניתן להשתמש בו כמודל גנרטיבי, כלומר מודל שיעד לייצר דוגמאות חדשות. לאחר התאמת התפלגות לכל אשלול, ניתן לדגום מההתפלגיות השונות ובכך לקבל דוגמאות חדשות.

2.2.3 Expectation–maximization (EM)

אלגוריתם מיקסום התוחלת הינו שיטה איטרטיבית למציאת הפרמטרים האופטימליים של התפלגיות שונות, במקרים בהם אין נוסחה סגורה למציאת הפרמטרים. נתבונן על מקרה של Mixture of Gaussians, ונניח שיש אשלול מסוים המתפלג נורמלית עם תוחלת ושותות (μ, σ^2) , ומשויכות אליו n נקודות. כדי לחשב את ההתפלגות של אשלול זה ניתן להשתמש בלוג הנראות המרבית:

$$L(\theta|x_1, \dots, x_n) = \log \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} = \sum_{i=1}^n \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(x_i-\mu)^2}{2\sigma^2}$$

כדי למצוא את הפרמטרים האופטימליים ניתן לגזר ולהשוו ל-0:

$$\frac{\partial L(\theta)}{\partial \mu} = \sum_{i=1}^n \frac{x_i - \mu}{\sigma^2} \rightarrow \mu_{MLE} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\frac{\partial L(\theta)}{\partial \sigma^2} = \frac{1}{2\sigma^2} \left(-n + \frac{1}{\sigma^2} \sum_{i=1}^n (x_i - \mu)^2 \right) \rightarrow \sigma_{MLE}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

כעת נניח שיש k אשלולות וכל אחד מתפלג נורמלית.icut סט הפרמטרים אותם צריך לעריך הינו:

$$\theta = \{\mu_1, \dots, \mu_k, \sigma_1^2, \dots, \sigma_k^2, \alpha_1, \dots, \alpha_k\}$$

עבור מקרה זה, הלוג של פונקציית הנראות המרבית יהיה:

$$L(\theta|x_1, \dots, x_n) = \log \prod_{i=1}^n \sum_{j=1}^k \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2) = \sum_{i=1}^n \log \left(\sum_{j=1}^k \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2) \right)$$

אם נגזר ונשווה ל-0 נקבל בדומה למקרה הפשוט:

$$\sum_{i=1}^n \frac{1}{\sum_{j=1}^k \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2)} \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2) \frac{(x_i - \mu_j)}{\sigma_j^2} = 0$$

נוסחה זו אינה ניתנת לפתרון אנליטי, ולכן יש הכרח למצוא דרך אחרת כדי לחשב את הפרמטרים האופטימליים של ההתפלגיות הרצויות. נתבונן בחלוקת מהביתי שקיבילנו:

$$\frac{1}{\sum_{j=1}^k \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2)} \alpha_j \mathcal{N}(x_i, \mu_j, \sigma_j^2) = \frac{p(y_i = j) \cdot p(x_i|y = j)}{p(x_i)} = p(y_i = j|x_i) \equiv w_{ij}$$

קייבלו למשה את הפוסטוריור y_i (האשכול אליו רוצים לשיר את x_i), אך הוא לא נתן אלא הוא חבו. כדי לחשב את המבוקש ננחש ערך התחלתי θ_0 ובעזרתו נחשב את y_i , ואז בהינתן y_i נבצע עדכון לפרמטרים – נבחן מהו סט הפרמטרים שסביר בצורה הטובה ביותר את האשכולות שהתקבלו בחישוב ה- y_i . באופן פורמלי שני השלבים מנוסחים כך:

E-step – בהינתן אוסף נקודות x וערך עבור פרמטר θ נחשב את האשכול המתאים לכל נקודה, כלומר כל נקודה x_i תוחאמ לאשכול מסוים y_i . עבור כל הנקודות y_i נחשב תוחלת ובעזרתה נגדר את הפונקציה $Q(\theta, \theta_0)$, כאשר θ הוא פרמטר חדש ו- θ_0 הוא סט הפרמטרים הנוכחיים.

$$Q(\theta, \theta_0) = \sum_{i=1}^n \sum_{j=1}^k p(y_i = j | x_i; \theta_0) \log p(y_i = j, x_i; \theta) = \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p(y_i = j, x_i; \theta)$$

$$\sum_{i=1}^n \mathbb{E}_p(y_i | x_i; \theta_0) \log p(y_i = j, x_i; \theta)$$

M-step – מחשבים את הפרמטר θ שיביא למקסימום את $Q(\theta, \theta_0)$ ואז מעדכנים את θ_0 להחדר:

$$\theta = \arg \max_{\theta} Q(\theta, \theta_0)$$

$$\theta_0 \leftarrow \theta$$

חוזרים על התהילך באופן איטרטיבי עד להתקנות.

עבור Mixture of Gaussians נוכל לחשב באופן מפורש את הביטויים:

$$Q(\theta, \theta_0) = \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p(y_i = j, x_i; \theta)$$

$$= \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p(y_i = j; \theta) + \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log p(x_i | y_i = j; \theta)$$

$$= \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log \alpha_j + \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log \mathcal{N}(\mu_j, \sigma_j^2)$$

$$= \sum_{i=1}^n \sum_{j=1}^k w_{ij} \log \alpha_j - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^k w_{ij} \left(\log \sigma_j^2 + \frac{(x_i - \mu_j)^2}{\sigma_j^2} \right)$$

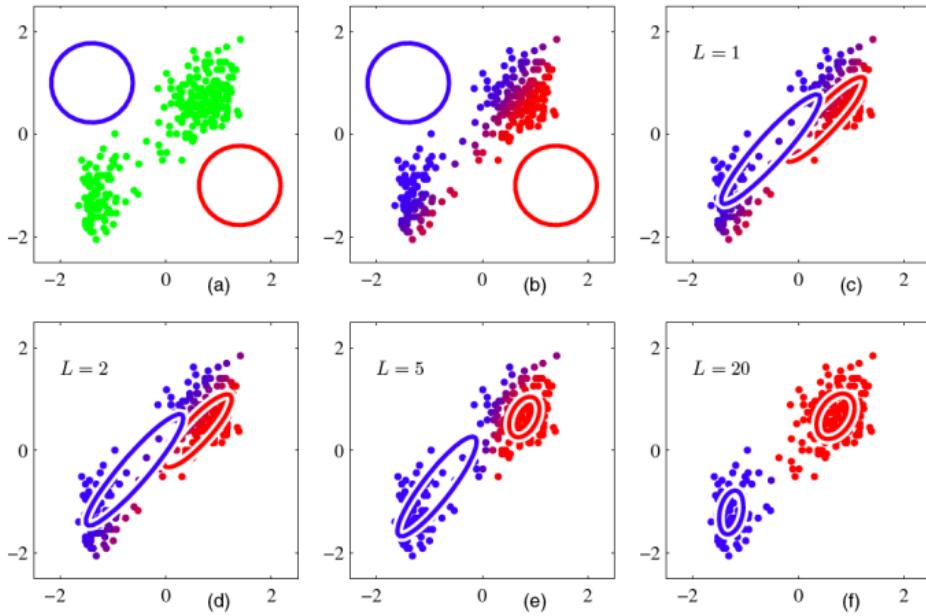
וכעת ניתן לגזר ולמצוא אופטימום:

$$\hat{\alpha}_j = \frac{1}{n} \sum_{i=1}^n w_{ij}$$

$$\hat{\mu}_j = \frac{\sum_{i=1}^n w_{ij} x_i}{\sum_{i=1}^n w_{ij}}$$

$$\hat{\sigma}_j^2 = \frac{\sum_{i=1}^n w_{ij} (x_i - \mu_j)^2}{\sum_{i=1}^n w_{ij}}$$

עבור התפלגיות שונות שאין בהכרח נורמליות יש לחזור לביטוי של $Q(\theta, \theta_0)$ ולבצע עבורו את האלגוריתם.



איור 2.10 איטרציות של אלגוריתם EM. מתחילה מינוחש אקראי של ההטפלוגיות, ובכל איטרציה יש שיפור כך שההטפלוגיות מייצגות בצורה יותר טובה את הדadataה המקורי.

ונכון שהאלגוריתם משתמש בכל איטרציה, ככלمر שעבור כל (θ_0, θ) מתקיים: $\log p(x; \theta) \geq \log p(x; \theta_0)$

$$\begin{aligned} \log p(x; \theta) &= \sum_y p(y|x; \theta_0) \log p(x; \theta) = \sum_y p(y|x; \theta_0) \frac{\log p(x, y; \theta)}{\log p(y|x; \theta)} \\ &= \sum_y p(y|x; \theta_0) (\log p(x, y; \theta) - \log p(y|x; \theta)) \\ &= \sum_y p(y|x; \theta_0) \log p(x, y; \theta) - p(y|x; \theta_0) \log p(y|x; \theta) \end{aligned}$$

נשים לב שהאיבר הראשון הוא בדיק $Q(\theta, \theta_0)$. האיבר השני לפני הגדרה הוא האנטרופיה של ההטפלוגיות $p(y|x; \theta_0)$:

$$H(\theta, \theta_0) = - \sum_y p(y|x; \theta_0) \log p(y|x; \theta_0)$$

cut עבור שני ערכים שונים של θ מתקיים:

$$\begin{aligned} \log p(x; \theta) - \log p(x; \theta_0) &= Q(\theta, \theta_0) + H(\theta, \theta_0) - Q(\theta_0, \theta_0) - H(\theta_0, \theta_0) \\ &= Q(\theta, \theta_0) - Q(\theta_0, \theta_0) + H(\theta, \theta_0) - H(\theta_0, \theta_0) \end{aligned}$$

לפי [איסויון גיבוב](#) מתקיים $Q(\theta, \theta_0) \geq H(\theta_0, \theta_0)$, לכן:

$$\log p(x; \theta) - \log p(x; \theta_0) \geq Q(\theta, \theta_0) - Q(\theta_0, \theta_0)$$

ולכן עבור כל עדכון של θ שمبיא לאופטימום את (θ, θ_0) , הביטוי $Q(\theta, \theta_0) - Q(\theta_0, \theta_0)$ יהיה חיובי וממילא יהיה שיפור ב- $\log p(x; \theta)$.

2.2.4 Hierarchical Clustering

אלגוריתם נוסף של למידה לא מונחית עבור חלוקת n נקודות ל- K אשכולות נקרא Hierarchical Clustering, והוא מחולק לשתי שיטות שונות:

ובכך מורדים את מספר האשכולות ב-1, עד למוגעים ל-K אשכולות. האיחוד בכל שלב נעשה על ידי מציאת שני agglomerative clustering – בשלב הראשוני מגדרים כל נקודה האשכול, וזה בכל פעם מאחדים שני אשכולות

האשכולות הקרובים ביותר זה לזה ואיחודם לאשכול אחד. ראשית יש לבחור מטריקה לחישוב מרחק בין שתי נקודות (למשל מרחק אוקלידי, מרחק מנהטן ועוד), ולאחר מכן לחשב מרחק בין האשכולות, כאשר יש מספר דרכים להגדיר את המרחק הזה, למשל:

complete-linkage clustering: $\max\{d(a, b) : a \in A, b \in B\}$.

single-linkage clustering: $\min\{d(a, b) : a \in A, b \in B\}$.

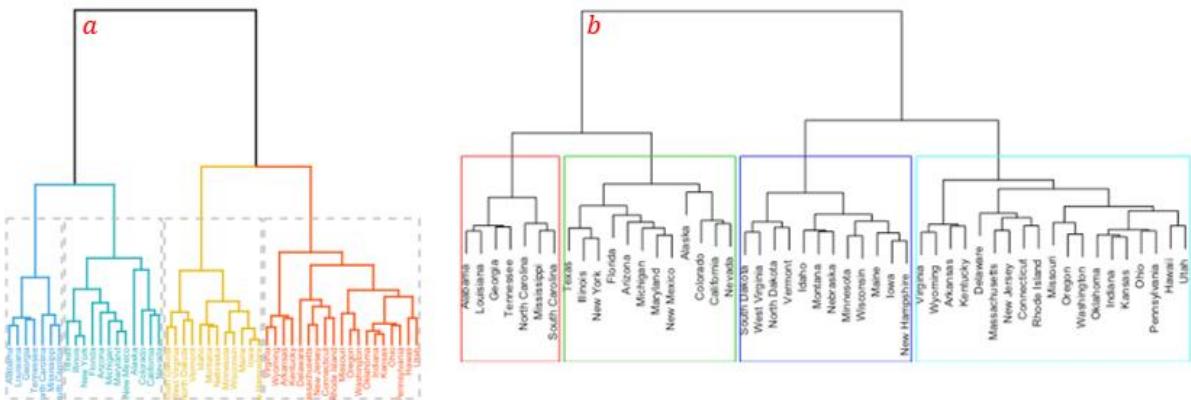
Unweighted average linkage clustering (UPGMA): $\frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$.

Centroid linkage clustering (UPGMC): $\|c_s - c_t\|$ where c_s, c_t are centroids of clusters s, t, respectively.

עם התקדמות התהילה יש פחת אשכולות, כאשר האשכולות כבר לא מכילים נקודה אחת בלבד אלא הם הולכים וגדלים. שיטה זו מכונה "top-bottom" ("bottom-top") שתחילתה כל נקודה הינה אשכול עצמאי ובכל צעד של האלגוריתם מסטר האשכולות קטן באחד. במקרים אחרים, האלגוריתם בונה את האשכולות ממצב שבו אין למעשה חלוקה לאשכולות למשך שבועות או חודשים. שיטה זו מכונה "hierarchical clustering" (DIANA) (Divisive ANAlysis Clustering).

בשיטתה זו מבצעים פעולה הפוכה – מסתכלים על כל הנקודות כאשכול אחד, ואז בכל שלב מבצעים חלוקה של אחד האשכולות לפי כלל חלוקה שנקבע מראש, עד שמגיעים ל-K אשכולות. כיוון שיש 2^n דרכים לחלק את המדגם, יש הכרח לנוקוט בשיטות היוריסטיות כדי לקבוע את כלל החלוקה המתואם בכל שלב. שיטה מקובלת לביצוע חלוקה נקראת DIANA (Divisive ANAlysis Clustering). ולפיה בכל שלב בוחרים את האשכול בעל השונות היכי גדולה ומחלקים אותו לשניים. שיטה זו מכונה "top-down" ("down-top") כיוון שבהתחלתה יש אשכול יחיד ובכל צעד של האלגוריתם מתווסף עוד אשכול.

את התצוגה של האלגוריתם ניתן להראות בצורה נוחה באמצעות dendrogram – דיאגרמה הבנוייה כעץ המציג קשרים בין קבוצות.



איור 2.11 תצוגה של Hierarchical Clustering (a) – התחלת האשכול הראשון (divisive) ו(b) – התחלת האשכול האחרון (agglomerative). צבעים שונים מציינים את מספר האשכולות הרצוי (במקרה זה K=4).

2.2.5 Local Outlier Factor (LOF)

אלגוריתם Local Outlier Factor הינו אלגוריתם של מנתה לא מונחית למציאת נקודות חריגות (Outliers). האלגוריתם מחשב לכל נקודה ערך הנקרוא (LOF), ועל פי ערך זה ניתן לקבוע עד כמה הנקודה היא חילוק מקבוצה או לחילופין חריגה ויצאת דופן.

בשלב ראשון בוחרים ערך k מסוים. עבור כל נקודה x_i , נסמן את k השכנים הקרובים ביותר שלה – $N_k(x_i)$.Cutננו את k -distance של כל נקודה כמרחק שלה מהשכן הרחוק ביותר מבין השכנים – $N_k(x_i)$. אם למשל $k=3$, אז $N_k(x_i)$ הוא סט המכיל את שלושת השכנים הקרובים ביותר ל- x_i , וה- k -distance שלה הוא המרחק מהשכן השליishi היכי קרוב. חישוב המרחק בין שני שכנים נתון לבחירה – זה יכול להיות למשל מרחק אוקלידי, מרחק מנהטן ועוד. עבור בחירה של מרחק אוקלידי, ניתן להסתמך על k -distance כמעגל – הרדיוס של מעגל המינימלי המכיל את כל הנקודות השוכנות ל- x_i הוא $N_k(x_i)$.

לאחר חישוב ה- k -distance של כל נקודה, מחשבים לכל נקודה (LRD) Local Reachability Density באופן הבא:

$$LRD_k(x_i) = \frac{1}{\sum_{x_j \in N_k(x_i)} \frac{RD(x_i, x_j)}{k}}$$

כאשר $RD(x_i, x_j) = \max(k - \text{distance}(x_i, x_j))$. הגודל LRD מחשב את ההופכי של ממוצע המרחקים בין x לבין k השכנים הקרובים אליו. ככל שנוקודה יותר קרובה ל- k השכנים שלה כך ה-LRD שלה גדול יותר, ו-LRD קטן משמעותית כשהנקודה רחוקה מascal הקרוב אליה.

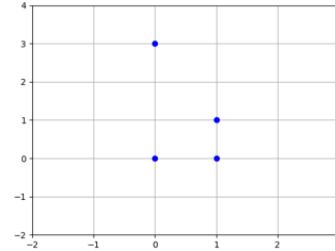
בשלב האחרון בוחנים עבור כל נקודה x את היחס בין ה-LRD שלה וה-LRD של $N_k(x)$. היחס הזה הוא ה-LOF, והוא מחושב באופן הבא:

$$LOF_k(x_i) = \frac{\sum_{x_j \in N_k(x_i)} LRD(x_j)}{k} \times \frac{1}{LRD(x_i)}$$

הביטוי הראשון במכפלה הוא ממוצע ה-LRD של k השכנים של נקודה x , ולאחר חישוב הממוצע מחלקים אותו ב-LRD של הנוקודה x עצמה. אם הערכים קרובים, אז ה-LOF יהיה שווה בקירוב ל-1, ואם הנוקודה x באמת לא שייכת לשכבו של נקודות, אז ה-LOF שלה יהיה נמוך מהתמוצע מהרחקים שלה, וממילא ה-LOF יהיה גובה. אם עברו נקודה x מתקבל $LOF \approx 1$, אז סביר שהוא חלק מסכום מסוים.

כדי להמחיש את התהליך נסתכל על האוסף הבא: $\{A = (0,0), B = (1,0), C = (1,1), D = (0,3)\}$, ונקבע $k = 2$. נחשב את ה- k -distance של כל נקודה במנוחים של מרחק מנהטו:

$$\begin{aligned} k(A) &= \text{distance}(A, C) = 2 \\ k(B) &= \text{distance}(B, A) = 1 \\ k(C) &= \text{distance}(C, A) = 2 \\ k(D) &= \text{distance}(D, C) = 3 \end{aligned}$$



נחשב את ה-LRD:

$$LRD_2(A) = \frac{1}{\frac{RD(A, B) + RD(A, C)}{k}} = \frac{2}{1+2} = 0.667$$

$$LRD_2(B) = \frac{1}{\frac{RD(B, A) + RD(B, C)}{k}} = \frac{2}{2+2} = 0.5$$

$$LRD_2(C) = \frac{1}{\frac{RD(C, B) + RD(C, A)}{k}} = \frac{2}{1+2} = 0.667$$

$$LRD_2(D) = \frac{1}{\frac{RD(D, A) + RD(D, C)}{k}} = \frac{2}{3+3} = 0.334$$

ולבסוף נחשב את ה-LOF:

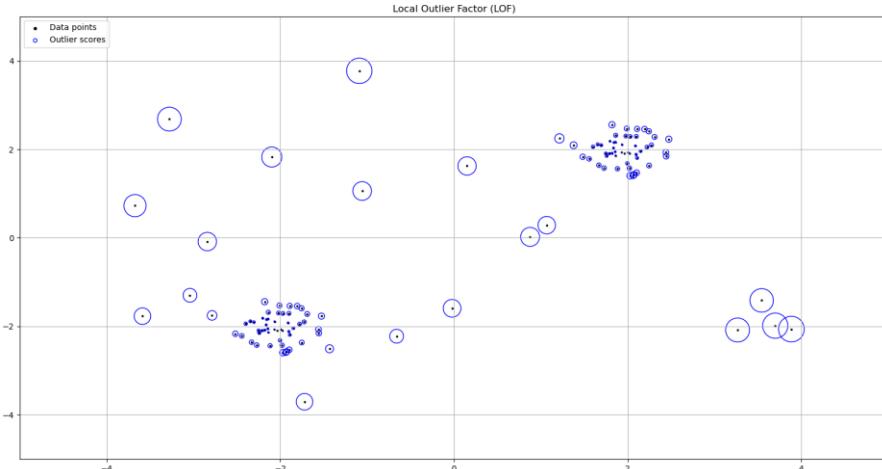
$$LOF_2(A) = \frac{LRD_2(B) + LRD_2(C)}{k} \times \frac{1}{LRD_2(A)} = 0.87$$

$$LOF_2(B) = \frac{LRD_2(A) + LRD_2(C)}{k} \times \frac{1}{LRD_2(B)} = 1.334$$

$$LOF_2(C) = \frac{LRD_2(B) + LRD_2(A)}{k} \times \frac{1}{LRD_2(C)} = 0.87$$

$$LOF_2(D) = \frac{LRD_2(A) + LRD_2(C)}{k} \times \frac{1}{LRD_2(D)} = 2$$

כיוון ש- $1 \gg LOF_2(D)$ באופן יחסי לשאר הנקודות, נסיק כי נקודה D היא outlier.



איור 2.12 Local Outlier Factor (LOF) – מציאת נקודות חריגות על ידי השוואת ערך ה-LRD של כל נקודה לממוצע ה-LRD של השכנים שלה. ככל שה-LOF גדול יותר (העוגל הכהול), כהה הנקודה יותר רוחקה משכניה.

יש שני אתגרים מרכזיים בשימוש באלאגוריתם זה – ראשית יש לבחור k מתאימים, כאשר k ייחסית קtan יהיה טוב עבור נקודות רועשות, אך יכול להיות בעיתני במקרים בהם יש הרבה מאוד נקודות חמודות אחת לשניה, ונקודה שמעט רחוקה מאוד מהתוצאות שהייא באמת כן שייכת אליו. k גדול לעומת זאת יתגבר על בעיה זו, אך הוא לא יזהה נקודות חריגות שנמצאות בקרוב לאשכולות של נקודות. מלבד אתגר זה, יש צורך לחתת פרשנות לתוצאות המתקבלות, ולהחליט על סף מסוים ש- LOF , שהחול ממנו נקודה מסווגת כחריגה. LOF קtan מ-1 הוא בוודאי לא outlier או עבר ערכי LOF גדולים מ-1 אין כלל חד משמעי עבור איזה ערך הנקודה היא outlier ועבור איזה ערך היא לא. כדי להתמודד עם אתגרים אלו הוצעו הרחבות לשיטה המקורית, כמו למשל שימוש סטטיסטיות שונות המורידות את התלות בבחירה הערך k (LoOP – Local Outlier Probability), או שיטות סטטיסטיות העוזרות לתת פרשנות לערכים המתקבלים (Interpreting and Unifying Outlier Scores).

2.3 Dimensionally Reduction

הורדת ממד (Dimensionality Reduction) הינה טרנספורמציה של נתונים מממד גבוה למדוד נמוך, כאשר נרצה שהורדת הממד לא תנסה באופן מהותי את מאפייני הדatta המקורי. הורדת הממד של נתונים נתן דרישת משתי סיבות עיקריות; הראשונה טכנית וקשורה לסבירויות גבואה במערכות מרובות ממדים, ואילו הסיבה השנייה יותר עקרונית ומהותית – הורדת הממד של הדatta קשורה לניסוי להבין מהם המשנים העיקריים והמשנים המשניים, הפחות חשובים להבנת הדatta (אלו שפחות מאפיינים דוגמא נתונה ביחס לדוגמאות אחרות). לעיתים התחשבות במשנים המשניים משפיעה לרעה על ביצועי המודל, למשל על ידי הוספת רעש ולא מידע. תופעה זו נקראת **קללת הממדיות** (curse of dimensionality). יתרון נוסף של הורדת ממד טמון בויזואלייזציה של המידע, כך שניתן להציגו על ידי 2 או 3 ממדים עיקריים, בعزيزת גרפ' דו-ממדי או תלת-ממדי בהתאם.

דוגמה למערכת מרובת ממדים יכולה להיות מדידת רמות חלבונים (proteins) של גנים (genes) המבוטאים בתא ח', כאשר כל ממד, או מאפיין (פיצר), מתייחס לגן אחר. באופן כללי, יתכן ונמדדים בכל ניסוי מאות תאים, כאשר לכל תא נמדדות רמות ביוטי של מאות או אלפי גנים. כמה עצמה זו של מידע בממד גבוה (אלפי תאים אלפי גנים בכל תא) מאגירה את המחקר – הן מבחינות זיהוי המאפיינים, או רמות הגנים המבוטאים, הרלוונטיים והמשמעותיים ביותר, והן מבחינות ניסויים למדל את הדatta בצורה כמה שייתר פשוטה. במחקר משנת 2007 נלקחו 105 דגימות של תא סרטן שד, כאשר לכל דגימה (או דוגמא) נמדדו רמות התבניות של 27,648 גנים שונים. כמובן שלבתה את המידע בצורה הגלומית זו משימה בלתי אפשרית, ויש הכרח לבצע עליה מניפולציה כלשהיא כדי שהיא אפשר לעבוד איתה.

ישנן שיטות מרובות להורדת ממד לדatta נתון, כאשר ניתן לסוגן לשני חלקים עיקריים: בחירת מאפיינים (feature selection), והטלת מאפיינים (features projection). השיטה הראשונה היא ניסוי לבחור את המאפיינים (המשנים) המתארים באופן מספק את המידע הנתון. השנייה, שבה עוסקת פרק זה, נוקטת בגישה של הטלה, טרנספורמציה, של המאפיינים הקיימים לסת של מאפיינים חדשים. חשוב להציג שבשיטות בחירת המאפיינים אנו בעצם משמשים מאפיינים פחות רלוונטיים. בנויגוד לכך, בשיטה שנדון בעט, שיטת הטלת המאפיינים, כל מאפיין חדש

הוא צירוף לינארי של כל האחרים, ולא רק של חלקם. כך, המאפיינים החדשניים מقلילים, או לוקחים בחשבון, כל אחד מהמאפיינים הנמדדים המקוריים, ללא השמטה.

ניתן לבצע הטלה מאפיינים באמצעות טרנספורמציה לינארית או לא-LINIARITY. בפרק זה עוסוק בטרנספורמציה LINIARITY, הנקראת ניתוח גורמים ראשיים (principal component analysis) ובשתי טרנספורמציות לא-LINIARITY (t-SNE, UMAP).

2.3.1 Principal Components Analysis (PCA)

כפי שהזכר לעיל, ניתוח גורמים ראשיים מבוסס על טרנספורמציה לינארית של המאפיינים המקוריים. הגורם הראשי הראשון, PCA_1 (first principal component) הינו הצירוף לינארי של המאפיינים הנתונים בעל השונות הגדולה ביותר. הגורם הראשי השני (PCA_2) הוא גם צירוף לינארי של המאפיינים הנתונים, השונות שלו היא השניה הגדולה ביותר, ובנוסף דורשים ש- PCA_1 יהיה אורתוגונלי ל- PCA_2 . הגורם השלישי, הוא צירוף לינארי בעל השונות השלישית הגדולה ביותר, ומאונך לשני הגורמים הראשונים – PCA_3 . וכך וגו' $PCA_1 \perp PCA_2 \perp PCA_3 \perp \dots \perp PCA_j \perp \dots \perp PCA_i$. הורדת הממד מתבצעת על ידי לקיחת מספר גורמים ראשיים ראשוניים, והזנתה הקטנים ביותר.

לאחר שאפינו את הגורמים הראשיים בהם אנו מעוניינים, עולה השאלה כיצד ניתן לבצע טרנספורמציה LINIARITY שבuzzرتה ניתן למצוא את הגורמים הראשיים הללו. נניח שבידינו DATA $\mathbb{R}^{M \times N}$, כלומר נתונות M דוגמאות שונות, שכל אחת מהן היא בעל N מאפיינים [למשל, עבור הדוגמא של תא סרטן השד, נתון מידע $M = 105$ תאים שונים, כאשר עבור כל תא נמדדו רמות ביוטי של $N = 27,648$ גנים שונים]. נסמן את מטריצת המאפיינים על ידי:

$$\vec{X} = \begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_M \end{bmatrix} \in \mathbb{R}^{M \times N}, \text{ כאשר } \vec{X}_m, m \in \{1, \dots, M\}, \text{ הינו נתוני המדידות של המאפיינים השונים בדוגמא מס' } m, \text{ וקטור שורה, } \{1, \dots, N\} \in \mathbb{R}^n, \text{ שימנו לב לשינוי סימונו, אינדקס עלון עבור וקטור עמודה}, \text{ הינו נתוני המדידות של מאפיין מסוים על כל הדוגמאות. נניח שסכום המדידות עבור כל מאפיין הוא אפס, זאת אומרת שסכום מאפיין } i \text{-י מתקיים:}$$

$$mean(\vec{X}^n) = \sum_{m=1}^M X_{m,n} = 0$$

מכיוון שכל עמודה של המטריצה מסמלת ערכים של מאפיין מסוים במדידות שונות, סכום כל עמודה במטריצה \hat{X} הוא אפס. עתה, נרצה לבצע הטלה (טרנספורמציה) LINIARITY, זאת אומרת נכפיל את מטריצה \hat{X} במטריצת משקלים \hat{W} :

$$\hat{T} = \hat{X} \cdot \hat{W}$$

אם נסמן את השורה ה- m -ית במטריצה \hat{T} על ידי \vec{T}_m , נקבל:

$$\vec{T}_m = \vec{X}_m \cdot \hat{W}$$

כאשר המטריצה $\hat{W} \in \mathbb{R}^{N \times K}$, כך ש- $\hat{W} \in \mathbb{R}^{M \times K}$. הטלה זו מביאה לכך שלאחר הטרנספורמציה נשאים רק K מאפיינים. כיוון שאנו מעוניינים בהורדת הממד, קרי הורדת מספר המאפיינים, נדרש $N \leq K$.

את תהליך מציאת מטריצת המשקלים ניתן לנתח באופן פורמלי על ידי שלושה תנאים:

$$(1) \text{ כל עמודה של מטריצת המשקלים הינה מנורמלת: } \|\hat{W}^k\|^2 = \sum_{m=1}^M (W_{m,k})^2 = 1$$

$$(2) \text{ השונות עבור המאפיין } k-th, \text{ המוגדרת על ידי } s_k^2 = (\vec{T}^k)^T \vec{T}^k = \sum_{m=1}^M (T_{mk})^2, \text{ מקיימת: } s_k^2 > s_{k+1}^2$$

$$(3) \text{ העמודות של } \hat{W} \text{ אורתוגונליות זו לזו, זאת אומרת } \hat{W}^k \perp \hat{W}^{k'}, \text{ לכל שתי עמודות } k, k'.$$

נראה זאת באופן מפורש: נתחילה במציאת העמודה הראשונה \hat{W}^1 . נדרש:

$$\hat{W}^1 = \underset{\|\hat{W}\|=1}{\operatorname{argmax}}(s_1^2)$$

זאת אומרת:

$$\begin{aligned}\widehat{W}_1 &= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}(s_1^2) = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\vec{T}^1\right)^T \cdot \vec{T}^1\right) = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{X}\widehat{W}^1\right)^T \cdot \widehat{X}\widehat{W}^1\right) \\ &= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{W}^1\right)^T\left(\widehat{X}\right)^T \cdot \widehat{X}\widehat{W}^1\right)\end{aligned}$$

ולכן העמודה הראשונה של מטריצת המשקלים \widehat{W} נתונה על ידי:

$$\widehat{W}^1 = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{W}^1\right)^T \cdot \widehat{S} \cdot \widehat{W}^1\right)$$

כאשר מטריצה $\widehat{S} = (\widehat{X})^T \cdot \widehat{X}$ הינה מטריצת השונות המשותפת (covariance), המוגדרת על ידי \widehat{X} .
 $S_{v_1, v_2} = \sum_{m=1}^M X_{v_1, m} X_{m, v_2}$, כאשר מגדירה את השונות המשותפת בין שני מאפיינים, כאשר ניתן לשים לב כי מטריצה זו סימטרית ומשנית (ולכן הרミיטית).

לפי משפט המינימום-מקסימום (קורנט-פישר-ויל): עבור \widehat{S} מטריצה הרמייטית ($S_{ij} = S_{ji}^*$), בעלת ערכים עצמיים $\lambda_K \geq \dots \geq \lambda_1$, מתקיים:

$$\lambda_1 = \max_{\|\widehat{W}\|=1}\left(\left(\widehat{W}^1\right)^T \cdot \widehat{S} \cdot \widehat{W}^1\right)$$

כאשר \widehat{W}^1 הינו הווקטור העצמי המתאים לערך העצמי המקסימלי של \widehat{S} : λ_1 .

כעת, כדי למצוא את הווקטור העצמי הבא, \widehat{W}^2 , והערך העצמי המתאים לו λ_2 , נגדיר מטריצה חדשה \tilde{X} :

$$\tilde{X} = \widehat{X} - \widehat{X}\widehat{W}^1(\widehat{W}^1)^T$$

$$\begin{aligned}\widehat{W}^2 &= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}(s_2^2) = \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\vec{T}^2\right)^T \cdot \vec{T}^2\right) \\ &= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{W}^2\right)^T\left(\tilde{X} + \widehat{X}\widehat{W}^1(\widehat{W}^1)^T\right)^T \cdot \left(\tilde{X} + \widehat{X}\widehat{W}^1(\widehat{W}^1)^T\right) \widehat{W}^2\right) \\ &= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{W}^2\right)^T\left(\tilde{X}\right)^T \cdot \left(\tilde{X}\right) \widehat{W}^2\right)\end{aligned}$$

כאשר \widehat{W}^2 הינו הווקטור העצמי המתאים לערך העצמי המקסימלי של \tilde{X} , ובפועל הוא הערך העצמי השני בגודלו. עבור מטריצה $\widehat{X} = \widehat{X}^T \widehat{W}^1$. (בчисוב השתמשנו בעובדה כי $\widehat{W}^1 \perp \widehat{W}^2$).

באופן כללי, כדי למצוא את \widehat{W}^k והערך העצמי המתאים לו λ_k , נגדיר מטריצה חדשה \tilde{X} באופן הבא:

$$\begin{aligned}\tilde{X} &= \widehat{X} - \sum_{i=1}^{k-1} \widehat{X}\widehat{W}^i(\widehat{W}^i)^T \\ \widehat{W}^k &= \underset{\|\widehat{W}\|=1}{\operatorname{argmax}}\left(\left(\widehat{W}^k\right)^T\left(\tilde{X}\right)^T \cdot \left(\tilde{X}\right) \widehat{W}^k\right)\end{aligned}$$

כך ש λ_k הינו הערך העצמי המקסימלי ה- k -י של מטריצת השונות המשותפת \widehat{X} . $\widehat{S} = \widehat{X}^T \widehat{X}$

ניתן גם, באופן פשוט יותר, להשתמש בשיטת פירוק לערכים סיגנוריים, כאשר נמצא את הפירוק המתאים למטריצת השונות המשותפת:

$$\widehat{S} = \widehat{W} \cdot \widehat{\Lambda} \cdot \widehat{W}^T$$

כאשר $\widehat{\Lambda}$ הינה מטריצה אלכסונית, $\Lambda_{ii} = \lambda_i$ הינם הערכים העצמיים של \widehat{S} המסודרים לפי גודלם מהגדול לקטן - $\lambda_M \geq \dots \geq \lambda_2 \geq \lambda_1$, ומטריצה \widehat{W} מרכיבת מוקטור עמודה שהינם הווקטורים העצמיים המתאים לערכים

העצמיים. הוקטורים העצמיים בהגדלתם הינם אורתוגונליים זה לזה, וכך \hat{W}^k לכל k , הם בעצם אורתוגונרמליים.

לטיכום, על מנת למצוא את הגורמים הראשיים עבור המידע הנוכחי \hat{X} :

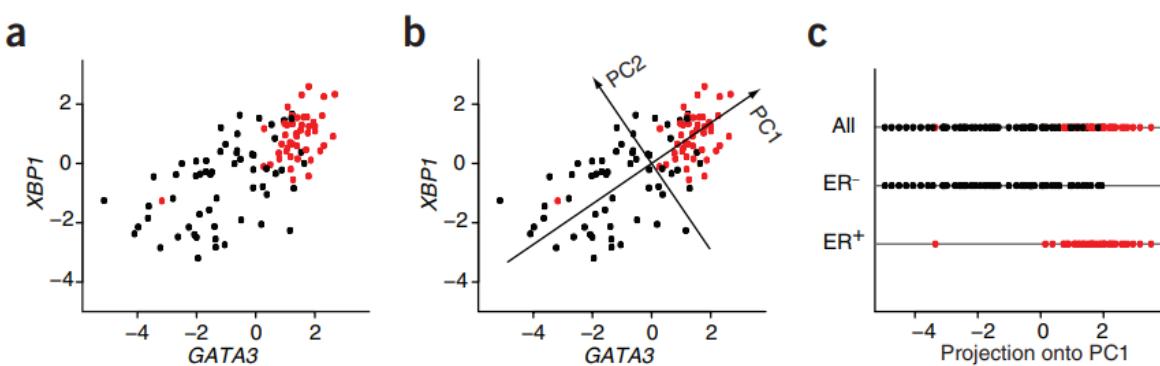
- א. "מרכז" את הנתונים כך שהממוצע עבור כל מאפיין הוא אפס: $\hat{X}^m = \hat{X}^m - mean_n(\hat{X}^m)$
- ב. מצא את מטריצת השונות המשותפת $\hat{S} = (\hat{X})^T \cdot \hat{S}$.
- ג. מצא את $\hat{W}^T \cdot \hat{L} \cdot \hat{S}$.
- ד. חשב $\hat{W}^T \cdot \hat{L} \cdot \hat{X}$.

הגורמים הראשיים נתונים על ידי וקטורי העמודה $\vec{W}^k \equiv PCA_k$, וההטלה של מדידה m למערכת המאפיינים החדשה נתונה על ידי $\vec{T}^m = \hat{X}^m \cdot \hat{W}^k$.

מצין שלשיטת הבניתו של גורמים ראשיים יש מספר מגבלות. ראשית, היא נותנת "משקל יתר" על מאפיינים שהשונות בהם גדולה, ללא קשר לחשיבותם, או ליחידות שבנה המאפיין נמדד (זאת אומרת לדוגמה שגובה שנמדד בסנטימטרים ינתן "משקל" גבוה יותר מאשר גובה הנמדד במטרים). שנית, שיטת זו מניחה כי הממד החשוב הוא השונות המשותפת שהיא בעצם קורלציה בין שני משתנים, אולם יתכן במערכות מסוימות שדווקא הקורלציה הלא-LINEARITATE היא החשובה יותר. כמו כן, לעתים "מרכז" המידע גורם לתוצאות לא-בד ממשמעותן.

כדי להתגבר על המגבילות בשיטת-PCA שהציגו לעיל פוטחו שיטות נוספות או משלימות. לדוגמה, ניתן למזער את השפעת יחידות המדידה על המאפיינים על ידי הפיכתם לחסרי יחידות. בנוסף, יש שיטות הלוקחות בחשבון קורלציות לא-LINEARITIES, לדוגמה שיטת PCA kernel, או שיטות להערכתם עם בעית המרכז על ידי דרישת משתנים חיוביים (NMF).

לצורך המחשה ניתן שתי דוגמאות. ראשית נזכיר דוגמא שהזכרנו בתחילת פרק זה – מחקר שפורסם בשנת 2007 בו נלקחו 105 דגימות של תא סרטן אחד, כאשר לכל דגימה נמדד רמות התבטאות של 27,648 גנים שונים. לשם הדגמה, נשתמש בניתוח שפורסם כמנה לאחר מכן (ב-2008) על ידי אחד מעורכי המאמר המקורי. שם, החוקרים מציג רמות של שני חלבונים; האחד בשם GATA3, והשני בשם XBP1, כאשר הוא מסווים את דגימות תא הסרטן לפי סוג קולטני האסטרוגן שלהם (+ או -). עתה, על ידי "סיבוב" מערכת הATTRACTים – באמצעות טרנספורמציה ליניארית PCA כפי שהסביר לעיל – נמצא כי ניתן לסווג, ללא אי-בוד מדיע רב, את מצב קולטני האסטרוגן בתאי סרטן השד על ידי הגורם הראשי PCA_1 , כפי שניתן לראות באירוע. יש לשים לב שהגורם הראשי הראשון, PCA_1 , מכיל מידע משני החלבונים.



איור 2.13 (a) רמות ביוטי של שני חלבונים GATA3 (ציר-X) ו-XBP1 (ציר-Y). קולטני אסטרוגן חיוביים או שליליים מסומנים באדום ושחור בהתאם. (b) מציאת הגורמים הראשיים, וסיבוב מערכת הATTRACTים. בהתאם לתיאוריה, ניתן להבחין כי השונות של המידע על גבי הציר החדש PCA_1 הינה מksamלית. (c) הצגת תוצאות המדידה כפונקציה של PCA_1 בלבד. בגרף זה ניתן לראות בבירור כיצד הורדת הממד מס'יעת למציאת הבדיקה פשוטה (במדד אחד) בין קולטני האסטרוגן.

נבייא בנוסף דוגמא חשיבות מפורטת. נניח וננתן המערכת הדו-מדדי הבא:

$$X = \begin{pmatrix} -0.5 & -0.4 \\ -0.4 & -0.1 \\ 0.1 & 0 \\ 0.3 & 0.3 \\ 0.5 & 0.2 \end{pmatrix}$$

מערך הנתונים מכיל 5 דוגמאות, ולכל דוגמא נמדדו שני מאפיינים. זאת אומרת $M = 5, N = 2$, כך שורות המטריצה מציגות את המדידות השונות, והעמודות מייצגות את מאפייניהם.

מערך זה כבר ממורכז, כלומר מתקיים עבור המאפיין הראשון:

$$mean_1(X^m) = \sum_{m=1}^5 X_{m1} = -0.5 - 0.4 + 0.1 + 0.3 + 0.5 = 0$$

ועבור המאפיין השני:

$$mean_2(X^m) = \sum_{m=1}^5 X_{m2} = -0.4 - 0.1 + 0 + 0.3 + 0.2 = 0$$

נחשב את מטריצת השונות המשותפת:

$$\begin{aligned} S &= (\hat{X})^T \hat{X} = \begin{pmatrix} -0.5 & -0.4 \\ -0.4 & -0.1 \end{pmatrix} \begin{pmatrix} 0.5 & 0.3 & 0.5 \\ 0.3 & 0.2 & 0.2 \end{pmatrix} \begin{pmatrix} -0.5 & -0.4 \\ -0.4 & -0.1 \\ 0.1 & 0 \\ 0.3 & 0.3 \\ 0.5 & 0.2 \end{pmatrix} \\ &= \begin{pmatrix} 0.5^2 + 0.4^2 + 0.1^2 + 0.3^2 + 0.5^2 & 0.5 \cdot 0.4 + 0.4 \cdot 0.1 + 0.1 \cdot 0 + 0.3^2 + 0.5 \cdot 0.2 \\ 0.5 \cdot 0.4 + 0.4 \cdot 0.1 + 0.1 \cdot 0 + 0.3^2 + 0.5 \cdot 0.2 & 0.4^2 + 0.1^2 + 0^2 + 0.3^2 + 0.2^2 \end{pmatrix} \\ &= \begin{pmatrix} 0.76 & 0.43 \\ 0.43 & 0.3 \end{pmatrix} \end{aligned}$$

על מנת ללקסן מטריצה זו, נפתחו:

$$0 = |\hat{S} - \lambda \hat{I}| = \begin{vmatrix} 0.76 - \lambda & 0.43 \\ 0.43 & 0.3 - \lambda \end{vmatrix} = (0.76 - \lambda)(0.3 - \lambda) - 0.43^2 \approx (\lambda - 1.02)(\lambda - 0.04)$$

כאשר לפולינום אופיני זה שתי פתרונות: $\lambda_1 \approx 1.02, \lambda_2 \approx 0.04$ (שים לב שהחרנו $\lambda_1 > \lambda_2$, התוצאות המובאות בחלק זה מקובלות ולכן הסימן \approx).

נמצא את הווקטור העצמי המתאים לערך העצמי הגדל מבין השניים – λ_1 . וקטור זה, המסומן על ידי \hat{W}^1 , מקיים:

$$\hat{S}\hat{W}^1 = \lambda_1 \hat{W}^1$$

כך ש:

$$0 = (\hat{S} - \lambda_1 \hat{I})\hat{W}^1 \approx \begin{pmatrix} -0.83 & 0.107 \\ 0.107 & -0.94 \end{pmatrix} \begin{pmatrix} W_{11} \\ W_{21} \end{pmatrix} \Rightarrow \hat{W}^1 \approx \begin{pmatrix} 0.86 \\ 0.51 \end{pmatrix}$$

הווקטור העצמי השני, \hat{W}^2 , המתאים לערך העצמי $\lambda_2 \approx 0.04$, מחושב באותו אופן, ומתקיים:

כך שמטריצת המשקלים נתונה על ידי:

$$\hat{W} = (\hat{W}^1 \quad \hat{W}^2) = \begin{pmatrix} 0.86 & 0.51 \\ 0.51 & -0.86 \end{pmatrix}$$

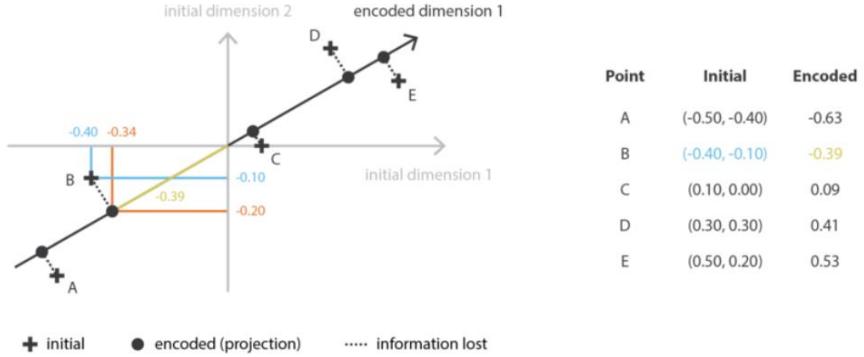
הטלה המדידות למערכת המאפיינים החדשה נתונה על ידי:

$$\hat{T} = \hat{X} \cdot \hat{W}$$

לכן, המדידות של הגורם הראשי הראשון, נתונות על ידי

$$\hat{T}^1 = \hat{X} \cdot \hat{W}^1 \approx \begin{pmatrix} -0.5 & -0.4 \\ -0.4 & -0.1 \\ 0.1 & 0 \\ 0.3 & 0.3 \\ 0.5 & 0.2 \end{pmatrix} \begin{pmatrix} 0.86 \\ 0.51 \end{pmatrix} = \begin{pmatrix} -0.5 \cdot 0.86 - 0.4 \cdot 0.51 \\ -0.4 \cdot 0.86 - 0.1 \cdot 0.51 \\ 0.1 \cdot 0.86 \\ 0.3 \cdot 0.86 + 0.51 \cdot 0.3 \\ 0.5 \cdot 0.86 + 0.2 \cdot 0.51 \end{pmatrix} \approx \begin{pmatrix} -0.63 \\ -0.39 \\ 0.09 \\ 0.41 \\ 0.53 \end{pmatrix}$$

נראה זאת באופן גרפי:



איור 2.14 הורדת ממד של נתונים דו-ממדית לממד אחד.

נספח: משפט המינימום- מקסימום (קורנט-פישר-ויל):

עבור $\hat{S} \in \mathbb{R}^{M \times M}$ מטריצה הרミיטית ($S_{ij} = S_{ji}^*$) מסדר M עם ערכי עצמיים $\lambda_1, \lambda_2 \geq \dots \geq \lambda_M$, מתקיים:

$$\begin{aligned} \lambda_m &= \min_U \left\{ \max_{\substack{x \in U, \\ \|x\|=1}} \{x^\dagger S^\dagger x \mid x \in U, x \neq 0\} \middle| \dim(U) = M - m + 1 \right\} \\ &= \min_U \left\{ \max_{\substack{x \in U, \\ \|x\|=1}} \left\{ \frac{x^\dagger S^\dagger x}{x^\dagger x} \right\} \mid x \in U, x \neq 0 \right\} \mid \dim(U) = M - m + 1 \end{aligned}$$

הערך העצמי המקסימלי מקיים:

$$\lambda_1 = \max_{\|x\|=1} ((\hat{W}^1)^T \cdot \hat{S} \cdot \hat{W}^1)$$

כאשר \hat{W}^1 , הינו הערך העצמי המתאים ל- λ_1 – ערך העצמי המקסימלי של \hat{S} .

2.3.2 t-distributed Stochastic Neighbors Embedding (t-SNE)

אלגוריתם הורדת הממד PCA פועל באופן LINEAR, מה שמקל על תהליכי החישוב שלו, אך מגביל את יכולות ההכללה שלו. אלגוריתם אחר, לא LINEAR, נקראת t-SNE, והוא מנשה לקחת את הדטה בממד גובה ועל ידי מיצד סטטיסטי למפותאות אותו למערכת דו-ממדית או תלת-ממדית. לשם כך, נשתמש באותו מערכת נתונים, $\hat{X} \in \mathbb{R}^{M \times N}$, כאשר M הוא מספר הדוגמאות, ו- N הוא מספר המאפיינים (או המשתנים). חשוב לשים לב כי כל מדידה מיוצגת על ידי וקטור שורה \vec{X}_m . הרעיון הכללי של השיטה הוא למפותאות את סט המדידות באמצעות צזה שמדידות דומות יותר, קרי מדידות "קרובות" יותר במרחב ה- N ממד, יוצגו על ידי נקודות קרובות יותר במרחב חדש K -ממד, כאשר לרוב $3 \leq K$. נסמן את המרחב המקורי ה- X ואת המרחב החדש ה- Y , כאשר בשני המרחבים המדידות מוצגות על ידי נקודות בגרף פיזור (scatter plot). המטריקה המשמשת לממדית דמיון (similarity) בין שתי נקודות במרחב המקורי X הינה הסתברותית. עבור שתי מדידות m_1, m_2 במרחב המקורי ה- N -ממד, ההתפלגות הנורמלית המשותפת P_{m_1, m_2} הינה:

$$P_{m_1, m_2} = \frac{\mathcal{Z}_1^{-1}}{2N} \exp \left(-\frac{\|\vec{X}_{m_1} - \vec{X}_{m_2}\|^2}{2\sigma_1^2} \right) + \frac{\mathcal{Z}_2^{-1}}{2N} \exp \left(-\frac{\|\vec{X}_{m_1} - \vec{X}_{m_2}\|^2}{2\sigma_2^2} \right)$$

כאשר i נקרא פרפלקסיות (perplexity) והוא פרמטר שנקבע מראש, ו- Z הינו קבוע הנורמליזציה, המוגדר על ידי $Z_i = \sum_{k \neq i} \exp \left(-\frac{\|\vec{X}_i - \vec{X}_k\|^2}{2\sigma_i^2} \right)$. עבור נקודות קרובות יותר, עבור הביטוי $\|\vec{X}_{m_1} - \vec{X}_{m_2}\|$ קטן, ההסתברות

שהנקודה $\|\vec{X}_{m_1} - \vec{X}_{m_2}\|$ שכנה של \vec{X}_{m_1} גדולה. לעומת זאת כאשר הנקודות רחוקות זו מזו, כלומר, ככלمر השתקה שנקודה \vec{X}_{m_1} שכנה של \vec{X}_{m_2} קטנה מאוד עד אפסית.

כעת, כפי שהוזכר לעיל, נרצה למפות את סט המדידות: $\begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_M \end{bmatrix} \rightarrow \begin{bmatrix} \vec{Y}_1 \\ \vdots \\ \vec{Y}_M \end{bmatrix}$, כך שהמדד של \vec{Y}_m הינו נמור (2 או 3 ממדים). בנוסף, נדרש שנקודות דומות ("שכנות") במרחב \mathcal{X} , ישארו שכןות לאחר המיפוי למרחב \mathcal{Y} . מתרבר שפונקציית ההסתברות המותנית, המתאימה לתיאור דמיון בין נקודות שכןות למרחב החדש \mathcal{Y} , הינה התפלגות \mathcal{t} , הניקראת גם התפלגות סטודנט עם דרגת חופש אחת (נדון ברענון לבחור בפונקציות הסתברות אלו בהמשך). כך נכתם את הדמיון בין m_1 לבין m_2 , על ידי ההסתברות המשותפת Q_{m_1, m_2} המוגדרת באופן הבא:

$$Q_{m_1, m_2} = 3^{-1} \frac{1}{1 + \|\vec{Y}_{m_1} - \vec{Y}_{m_2}\|^2}$$

כאשר $3 = \sum_{k \neq j} \left(1 + \|\vec{Y}_k - \vec{Y}_j\|^2\right)^{-1}$ הינו קבוע נורמלייזציה.

המיפוי בין מרחב המקור \mathcal{X} לבין המרחב החדש \mathcal{Y} הוא מיטבי אם הוא "משמר" את השכניםות של נקודות (מדידות) קרובות. לשם כך נגידר את פונקציית המחיר על ידי Kullback-Leibler divergence, הבוחן מרחק בין שתי התפלגיות:

$$C = \mathcal{D}_{KL}(P||Q) \equiv \sum_{m_1} \sum_{m_2} P_{m_1, m_2} \log \left(\frac{P_{m_1, m_2}}{Q_{m_1, m_2}} \right)$$

נרצה למצוא את הווקטור $\begin{bmatrix} \vec{Y}_1 \\ \vdots \\ \vec{Y}_M \end{bmatrix}$ עבורו פונקציית המחיר מינימלית, ולשם כך נשתמש בגרדיינט לפי:

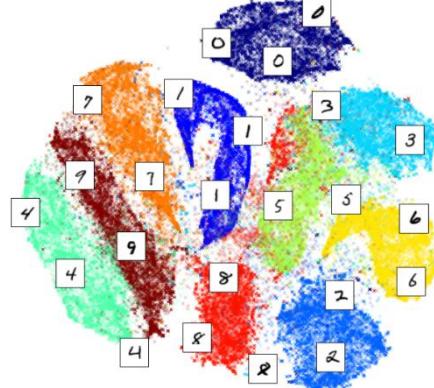
$$\begin{aligned} \frac{\delta C}{\delta \vec{Y}_{m_i}} &= \frac{\delta}{\delta \vec{Y}_{m_i}} \left[\sum_{m_1} P_{m_1, m_i} \log \left(\frac{P_{m_1, m_i}}{Q_{m_1, m_i}} \right) + \sum_{m_2} P_{m_1, m_2} \log \left(\frac{P_{m_i, m_2}}{Q_{m_i, m_2}} \right) \right] \\ &= 4 \sum_{m_1} (P_{m_1, m_i} - Q_{m_1, m_i}) \left(1 + \|\vec{Y}_{m_1} - \vec{Y}_{m_i}\|^2 \right)^{-1} (\vec{Y}_{m_1} - \vec{Y}_{m_i}) \end{aligned}$$

чисוב המינימום באופן אנלטי לא תמיד אפשרי או לא תמיד יעיל, ולכן מקובל להשתמש בשיטת gradient descent שaining שיטה איטרטיבית למציאת המינימום של פונקציה (פירוט על שיטה זו ווריאציות שונות שלה מופיע בחלק 4.3.5). עבור הורדת הממד, חישוב המינימום בעזרת שיטה זו יעשה באופן הבא:

- א. אתחול: * נתון $X \in \mathbb{R}^{M \times N}$.
- * פרמטר לפונקציית הדמיון: בחירת השונות σ^2 .
- * בחירת פרמטרים לאופטימיזציה: קצב הלמידה η , מומנטום $\alpha(t)$.
- ב. חשב את P_{m_1, m_2} .
- ג. אתחל את המיפוי ($s \hat{I}_M \sim N(0, s \hat{I}_M)$) $\psi^{(0)} = \{\vec{Y}_1, \vec{Y}_2, \dots, \vec{Y}_M\}$ [ז"א בחר את הערכים ההתחלתיים לפני התפלגות גausianית עם ממוצע 0 וסטיית תקן s (s נבחר להיות קטן, נניח $s = 10^{-4}$). \hat{I}_M מטריצת יחידה].
- ד. עברו איטרציה t : * חשב את Q_{m_1, m_2} .
- * חשב את הגרדיינט של פונקציית המחיר $\frac{\delta C}{\delta \psi}$.
- * עדכן: $\psi^{(t+1)} = \psi^{(t-1)} + \eta \frac{\delta C}{\delta \psi} + \alpha(t)[\psi^{(t-1)} - \psi^{(t-2)}]$.

נעיר כי בשפות תכנות רבות, האלגוריתם עצמו כבר מוגדר על ידי פונקציות מובנות, ויש רק להגדיר את הפרמטרים הדרושים.

במאמר המקורי שהציג את השיטה הובאה דוגמא של שימוש באלגוריתם עבור הטלה של הספרות 0 עד 9, המיצגות על ידי תמונות במד גובה 28×28 , למרחב דו-ממדי, בדגם איזומטרי. בדגם זה נלקחו 6,000 תמונות של ספרות ומיפוי אותן למרחב דו-ממדי. במרחב זה ניתן לראות בבירור כיצד כל תמונה מופתעת לאחור אחר, כיוון שבפועל נוצרו עשרה אשכולות שונים, המובחנים בצורה ברורה אחד מהשני. ביצוג הדו-ממדי אין משמעות לציריהם, כיוון שבאלגוריתם זה יש חשיבות רק למוחק היחס בין הנקודות.



איור 2.15 הצגה (יזואלייזציה) דו-ממדית של מערך נתונים עבור כתבי-יד של ספרות (MNIST) על ידי שיטת SNE-t. כל דגם \vec{X}_m מאופיינית על ידי $784 \times 28 = 784 \times 28$ ערכים (פיקסלים בגווני אפור) ומסוגת להיות ספרה בין 0 ל-9. באירז מוצגות 6000 נקודות (מדידות) כלו, כאשר צבעים שונים מייצגים ספרות שונות. מלבד ההבחנה בין הספרות, ניתן לראות שספרות דומות קרובות זו לזו גם במרחב החדש (למשל הספרה 1 קרוביה לספרה 7, שבturnora קרוביה לספרה 9).

כאמור, פונקציית הדמיון בין שתי נקודות למרחב המקורי הינה הפילוג הנורמלי המשותף של שתי הנקודות, ואילו במרחב החדש פונקציית הדמיון הינה התפלגות t . שתי העורות חשובות על בחירות אלו:

א. סימטריה:

fonkciyah ddimion gaoosianit bin shati nukodot bermachb χ hinya fonkciyah simetrik, kolomr $P_{m_1, m_2} = P_{m_2, m_1}$. Aolim nitn lehagidir gem fonkciyah ddimion a-simetrik, mboosut ul htaplagot motnt (bmkom htaplagot meshutaf). fonkciyah motnta ntuna ul id:

$$P_{m_1|m_2} = Z_2^{-1} \exp\left(-\frac{\|\vec{X}_{m_1} - \vec{X}_{m_2}\|^2}{2\sigma_2^2}\right)$$

kr sh:

$$P_{m_1, m_2} = \frac{P_{m_1|m_2} + P_{m_2|m_1}}{2N}$$

ב. בחירת Fonkciyah ddimion bmkom Fonkciyah t:

balgoritam shatوار, fonkciyah ddimion bin shati nukodot bermachb χ ntuna ul id htaplagot t . Nitn lehagidir gem fonkciyah achra, lmasl at fonkciyah ddimion gaoosianit ubor shati mdidot bermachb χ . Shita zo nkrat SNE, vahgerdianut shel fonkciyah mchir bmkraha zo ntuna ul id:

$$\frac{\delta C}{\delta \vec{Y}_{m_i}} = 4 \sum_{m_1} (P_{m_1, m_i} - Q_{m_1, m_i}) (\vec{Y}_{m_1} - \vec{Y}_{m_i})$$

g. Aolim, fonkciyah ddimion gaoosianit bermachb χ ycolah lagrom lcr shnkodot la maod krivot bermachb χ , ymopo lnkodot krivot bermachb χ , cion shagaoosian butzem gorom latrktor (mshicha) ychisit chzq bin shati nukodot, gem bmkrim bhem hnukodot ainin maod krivot. loumat zat, casher fonkciyah ddimion hinya htaplagot stodont t , shinya htaplagot um znb cbd yoter, shati nukodot shainin maod krivot ymopo bztora roaya lermachb χ kr shainin "nmeskot" ao matkrivot zo lzo. Shita achra, nkrat SNE-UNI, mtsia lahstems htaplagot achida, ark gem la chsrn domha l-SNE, casher shati nukodot la maod domot zo lzo, ain "dzhoot" achta et

השנייה. באופן אינטואיטיבי, ניתן לחשב על גרדיאנט פונקציית המחיר כבדה כוח, ועל פונקציית המחיר בתור פוטנציאל, כך שהכוח הפועל הוא בעצם כוח קפיץ.

לשיטת t-SNE יש שלוש מגבלות עיקריות

א. הורדת ממד: השיטה משמשת לויזואליזציה של מידע מממד גבוה בדו-ממד או תלת-ממד. אולם, באופן עקרוני, יתכן ונרצה להורד את הממד לא לשם הצגתו, אלא לצרכים אחרים, כאשר הממד החדש הינו גדול מ-3. יתכן ובממד גבוה פונקציית התפלגות סטודנט t עם דרגת חופש אחת, אשר לה משקל גבוה יחסית במרקחים גבוהים, לא תשמיר את המבנה של המידע המקורי. לכן, כאשר נרצה להורד לממד גבוה מ-3, פונקציית התפלגות t עם יותר מדרגת חופש אחת מתאימה יותר.

ב. קלילת הממדיות: t-SNE מבוססת על מאפיינים מקומיים בין נקודות. השיטה, המבוססת על טריצית מרחק אוקלידי, וכר מנicha לנאריות מקומית על גבי היריעה המתמטית בה מתקיימות הנקודות. אולם, במרחב נתונים בו הממד הפנימי גבוה, שיטת t-SNE עלולה להיכשל כיוון שהנחה הלינארית לא מתקיימת. למרות שישנן מספר שיטות למצער תופעה זו, עדין, בהגדלה, כאשר הממד הפנימי גבוה, לא ניתן להורד ממד הכר שמבנה המידע ישמר באופן מלא.

ג. פונקציית מחיר לא קמורה: הרבה שיטות למידה מבוססות על פונקציית הפסד קמורה, כך שתיאורטית מציאות אופטימיזציה (יחידה) לפונקציה זו אפשרית תמיד. אולם, בשיטת t-SNE, פונקציית המחיר אינה קמורה, והפתרון המתתקבל על ידי האופטימיזציה משתנה בהתאם לפרמטרים הנבחרים.

2.4 Ensemble Learning

2.4.1 Introduction to Ensemble Learning

נכיה ויש בידינו אוסף נתונים מסוים, ורוצים לבנות מודל המנתה את הנתונים הללו שמתבסס על אלגוריתם מסוים. כמעט תמיד, המודל לא יהיה מדויק במאה אחוז, והוא יהיה בעל שונות או בעל הטיה. ניתן להשתמש במכול (Ensemble) של מודלים שונים המבוססים על אותו אלגוריתם רצוי, ובכך לקבל מודל משוקל בעלי שונות/טיה נמוכים יותר מאשר מודל שמתבסס על אותו אלגוריתם אך נבנה באופן פשוט.

בכדי להבין את יותר טוב את החשיבות של שימוש ב-ensembles, Trade off בין שנות המודל להטיה שבו. מודל אופטימי יתאפשר בשונות נמוכה ובhetiya נמוכה. לעומת, השוני בין התוצאות לא יהיה מהותי, ובממוצע התוצאות תהיה קרובה מאוד לערך האמתי. מודל כזה יהיה מודל אמין, וכן יכול לבסס עליו את עצמנו. למרבה הצער, מודל שכזה לרובינו אפשרי. סוג אחר של מודל יהיה המודל הגרוע, הפהיר למודל האופטימי. זהו מודל עם שנות גבוהה והטיה גדולה. מודל שכזה יציג טוויה רחבה של תוצאות על אותם נתונים, ובממוצע יהיה רחוק מאוד מהערך האמתי. מודל זה כלל אינו שימושי.

בפועל, המודלים במציאות ינוויל לארוך SCI קצחות: מודלים עם שנות גבוהה והטיה נמוכה, ומודלים עם שנות נמוכה והטיה גבוהה. הדיזיון של המקום שלנו לאורך ציר זה קרייטי, כיוון שהוא מאפשר לנו לבחור את דרך ההסתמודות הטובה ביותר. יש מספר משפחות של t-SNE models, ושני העקרורים שבהם נקראים "לומדים". Bagging and Boosting. כאשר ניתקל במודלים עם שנות גבוהה, ככל מודל הסובל מ-Overfitting, לרוב נרצה להשתמש באנסמל מסווג Bagging על מנת להורד את השונות במודל הסופי. אלגוריתם מסווג Boosting יטפל במקרה השני, בו הטיה גבוהה והשונות נמוכה.

2.4.2 Bootstrap aggregating (Bagging)

Bagging היא משפחת אלגוריתמים אשר פועלת כ-ensemble, כלומר – מספר אלגוריתמים שפועלים ביחד, על-מנת להציגו לתוצאה משופרת. כאמור, אלגוריתמים מסווג bagging נועד להגדיל את יציבות המודל והעלאת הדיק שלו, זאת תוך הורדת השונות והימנענות מ-overfitting. Bagging מרכיב מספר רב של אלגוריתמים, המכונים "לומדים" ("Weak learners"), כאשר כל אחד מהם מבצע למידה ותחזית על חלק מן הנתונים, מתוך מטרה להציג לתוצאה אינטואיטיבית. אלגוריתם bagging הינו שיטה נפוצה ו פשוטה יחסית לשיפור ביצועים, אם כי הוא עשוי להיות קרה מבחינה חשיבותית.

מודל "פשוט" יקבל את הנתונים, יתאמן עליהם ויבצע תחזית על נתונים חדשים. זהו תהליך הלמידה והבחן אשר ידוע לנו ממודלים כגון עץ החלטה (Decision Tree), רגסטי לינארית וכו'. כפי שהסביר לעיל, מצב זהה עשוי להוביל להסתמודות יתר של המודל לנטיון האימון, דבר שעשוי להוביל למודל בעל שונות גבוהה. בכך לסתמודות עם בעיה זו, אלגוריתמים מסווג bagging מפרקם את התהליך לשני שלבים: Bootstrapping and Aggregating.

בשלב ה-Bootstrapping, יוצרים מהנתונים המקוריים קבוצות חדשות, כאשר כל קבוצה נוצרת על ידי דוגמה (עם חוזרות) של איברים מהקבוצה המקורי, באופן כזה שגודל כל קבוצה חדשה הוא בגודל של הדאטה המקורי. בשלב

השני, ה-*Aggregating*, הקבוצות החדשות נכנסות כקלט ל"לומדים חלשים", אלגוריתמים פשוטים יותר, אשר עובדים במקביל על תחזית, כמו למשל, יוצרים מודל נפרד לכל קבוצה של נתונים. בשלב הסופי, יבוצע איחוד של כל המודלים על מנת ליצור מודל משוקל בעל שונות קטנה יותר מאשר מודל המסתמך על הדאטה המקורי כפי שהוא.

אופן חיבור המודלים המתבצע ב-*bagging* מבוסס על אותו רעיון של NN-K, כאשר מודלים אלו יכולים לשמש הן למטרות סיוג והן למטרות גרסיה. כאמור לעיל (פרק 2.1.3), באlgorigithם השכן הקרוב כל "שכן" העיד על התוויות שלו, ולאחר הכרעת הרוב נקבעת התוויות של התצפית החדשת. במקרה שבו נספור את תדירות כל התוויות השכנות, התוויות הנבחרת תהיה של התצפית הנפוצה ביותר; נעשה זאת כאשר NN-K יעבד כמסוג. במקרים בהם NN-K יעבד כגרסיה, יבוצע ממוצע של כל התוויות השכנות, וזאת גם תהה התחזית. כאשר *bagging* יעבד כמסוג, כל *weak learner* יבצע תחזית, והתוויות השכיחה ביותר תהה התוצאה של האנסמבל (-הכרעת הרוב). כאשר *bagging* יעבד כגרסיה, כל מודל יבצע תחזית, אבל התוצאה של האנסמבל תהה הממוצע של כל המודלים.

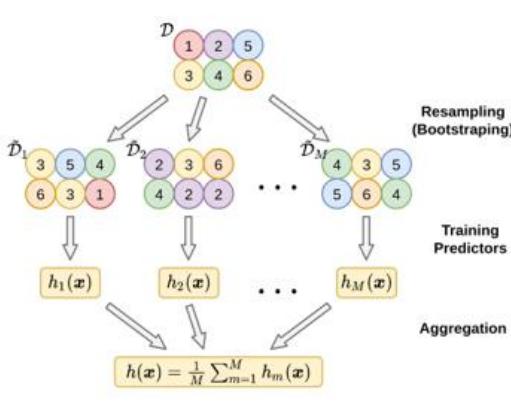
באופן פורמלי, עבור דאטה $\mathcal{R} \in \mathbb{R}^{n \times d}$, ניצור M קבוצות חדשות באותו גודל של הדאטה המקורי – עבור כל קבוצה $m \in \mathbb{R}^{n \times d}_{m=1}$, נבנה מודל $(x)_m$. עבור עיות סיוג ההחלטה תתקבל על פי הצבעת הרוב:

$$C(x) = \text{majority}(\{c_1(x), \dots, c_M(x)\})$$

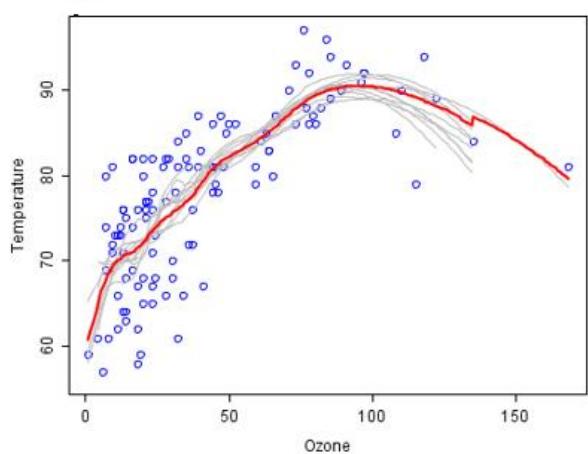
ועבור עיות גרסיה ההחלטה תבוצע בעזרת מיצוע כל המודלים:

$$C(x) = \frac{1}{M} \sum_{m=1}^M c_m(x)$$

a



b



איור 2.16 (a) אלגוריתם Bagging: בשלב ראשון יוצרים הרבה מחלקות שונות שנוצרות מהדאטה המקורי (Bootstrapping), ולאחר מכן מודלים המתאימים לכל מחלקה (Aggregating), ולבסוף יוצרים מודל יחיד המבוסס על כל המודלים המקוריים. (b) דוגמא לבניית מודל גרסיה בעזרת אלגוריתם Bagging. ניתן לראות שהמודל המשוקל הוא בעל שונות קטנה יותר מכל שאר המודלים.

בין אם משתמשים בהכרעת הרוב ובין אם משתמשים במיצוע של המודלים, המודל המשוקל שנוצר הופך להיות חלק יותר ובעל פחות שיפועים חדים, מה שמקטין את ה-*overfitting*, ומילא מפחית את השונות. ניתן להבין זאת על ידי דוגמא פשוטה – נניח שיש התפלגות נורמלית (σ^2, μ), אז השונות של n דגימות בלתי תלויות הינה $\frac{\sigma^2}{n}$. בעת נניח ומבצעים n ניסויים שככל אחד מהם דוגמים n נקודות, ובין כל שתי קבוצות יש קורלציה ρ . השונות הממוצעת של הניסויים הינה:

$$\text{Var} \left(\frac{1}{m} \sum_{i=1}^m \text{single cycle} \right) = \frac{1}{m} (1 - \rho) \sigma^2 + \rho \sigma^2$$

אם נבצע הרבה מאוד ניסויים, כמו m גדול מאוד, נקבל:

$$\lim_{m \rightarrow \infty} \frac{1}{m} (1 - \rho) \sigma^2 + \rho \sigma^2 = \rho \sigma^2$$

ובסך הכל השונות הסופית הינה בקירוב ² סק, וביתוי זה לרוב קטן מאשר השונות של מודל המבוסס על הדאטה המקורי ללא שימוש-b-ensembles. ניתן לשים לב שכלל שהקורסציה בין הקבוצות קטנה, אך השונות של המודל המשוקל גם כן קטנה יותר.

מודל נפוץ מאוד מסוג bagging הוא Random Forest. אלגוריתם זה משלב בין עצי החלטה לבין הרעיון הבסיסי של bagging, כאשר הוא מפצל את הנתונים ואת המשתנים לעצי החלטה רבים, וכל אחד מהם מקבל חלק מסוים מן השלם. העצים הם בעלי שונות גבוהה, ככלمر – כל אחד מהם הוא overfitting עצמו, אך עם זאת הקורסציה ביןיהם נמוכה, מה שמקל על הורדת השונות והימנענות מ-overfitting. לבסוף, השקלול של כל המודלים ביחיד מצליח לייצר מודל בעל שונות נמוכה, ומוביל לתוצאות טובות.

ל-bagging יתרונות רבים. הוא מוריד את השונות, והוא גם חסין לעריצים קיצוניים (Outliers). יכולת העבודה שלו במקביל עשויה לאפשר לו להגיע לתוצאות באופן מהר יותר.

עם זאת, ל-bagging יש גם חסרונות. הוא אינו מוריד את ההטייה, ולכן עשוי לא להתאים במרקם רבים. במודלים של בינה מלאכותית יש חשיבות רבה ליכולת הפרשנות של המודל; לחוב, ידרש הסבר פחות טכני של תוצאות המודל למבקלי ההחלטה או לצרכנים. הם עשויים לא לקבל כלל החלטות של מודל שראה "קופסה שחורה". יש קושי רב לתת פרשנות להחלטות של מודלים מבוססי bagging, והדבר מתקשה על השימוש בו. מעבר לכך,bagging עשוי להיות יקר מבחינה חישובית. עקב כך, הוא שימושי מאוד במרקם בהן שיפור זעיר עשוי להוביל להצלחה, אך לרוב תינען עדיפות למודלים פשוטים יותר של ensembles.

2.4.3 Boosting

כאמור, המושג boosting מתיחס למשפחת אלגוריתמים המשתמשים באוסף של מודלים "חלשים" על מנת ליצור מודל אחד "חזק", כאשר מודלים אלו מתמקדים בניסיון להפחית את ההטייה שיש למודל. מבחינה אינטואטיבית, מודל חלש הוא זהה שתוצאותיו מעט טובות יותר מNICHOש אקראית בעוד שאחד חזק מתקרב לביצועים אופטימליים. בינויג לטכניקות ensemble שפועלות במקביל, העקרון המנחה כאן הוא לשרשר את המודלים באופן כזה שכל מודל שמתווסף יטפל בשגיאות שקדמי פספסו. היפוי נעז בכך ש-boosting מוכיח כי למידה חלה בהכרח מצבעה על קיום של שיטת למידה חזקה, לרוב, מודלים מבוססי boosting מתמקדים בעיות סיווג בינהי.

באופן פורמלי, המושגים "לומד חלש" ו-"לומד חזק" עברו בעיתות סיווג בינהי מוגדרים כך: אלגוריתם נקרא לומד חזק אם $\text{אם } \underset{\epsilon}{\text{לכל }} 0 < \delta, \epsilon \text{ האלגוריתם מסוגל (בעור אוסף נתונים גדול מספיק) לבנות מסוג } (x)_c \text{ שמקיים } < (y) \neq c$ בהסתברות גדולה מ- $\delta - \epsilon$. לומד חלש הינו אלגוריתם שלכל $0 > \delta \text{ קיימים } 0 > \gamma \text{ כך שעור אוסף נתוני מספיק}$ גדול, האלגוריתם מסוגל לבנות מסוג שמקיים $\gamma - \frac{1}{2} < (y) \neq (x)_c$ בהסתברות גדולה מ- $\delta - \epsilon$. כאמור, המטרה של boosting הינה לחתת אוסף של מודלים חלשים ובעזרתם ליצור מודל חזק, כאשר הצליחו להוכיח שnitן להפוך כל לומד חלש לומד חזק על ידי בניית קומבינציה ליניארית של מסוגים אשר נוצרו בעזרת הלומד החלש.

נמחיש את הרעיון של boosting בעזרת דוגמא: נניח ויש בידינו אוסף נתונים X , המכולק באופן אקראי לשולש קבוצות שוות (כל אחת מכילה שליש מהנתונים) – x_1, x_2, x_3 .icut בונים מודל לצורך סיווג בינהי, המסומן ב- h . נמצא כי h מתאים בצורה טובה רק לקבוצות x_2, x_3 אך מסוג בצורה לא טובה את פריטי הקבוצה x_1 . כיוון ש- x_3 מכיל שליש מסך הנתונים, שגיאת הסיווג גדולה $-h(x)$ הוא מודל חלש, ונרצה לשפר אותו. כדי לעשות זאת ניקח רק חלק מהנתונים, $X \in X'$, ונdagאג לכך ש- X' יכול הרבה מאיברי x_1 .icut בונה מודל נוסף $(x)_c$ על בסיס X' , מתוך כוונה שמודל זה יתמקד גם בקבוצה x_1 ויסוג את איבריה בצורה טובה.icut בונה מודל זה אכן מסוג בצורה נאותה את איברי x_1 , אך הפעם המודל שוגה בצורה גסה בסיווג איברי x_2 . undercut השגיאה בסיווג x_2 המודל השני גם אוסף הנתונים המקורי X . אם נמצא דרך הולמת לחבר את שני המסוגים, יוכל ליצור מודל בעל פוטנציאל להצלח לסוג את X כמו שצרכן.

באופן כללי, אם רוצים לאמן מודל $(x)_C$ בעזרת אלגוריתם L על אוסף הנתונים \mathcal{D} , יש לבצע את השלבים הבאים:

$$1. \text{ אתחול הנתונים: } \mathcal{D}_1 = \mathcal{D}.$$

$$2. \text{ עבור } t = 1, \dots, T:$$

$$\text{אימון מודל חלש על: } c_t(x) = L(\mathcal{D}_t).$$

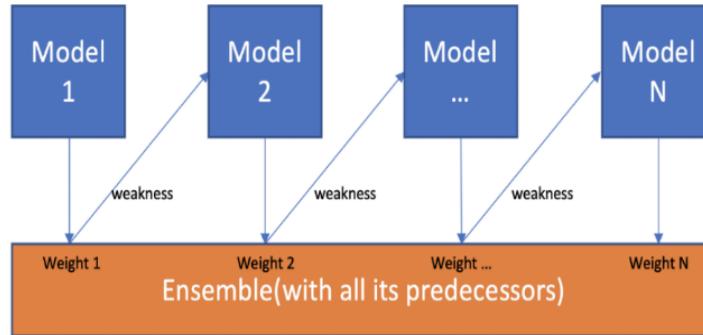
$$\text{чисוב שגיאת המסוג: } \epsilon_t = P_{x \sim \mathcal{D}_t}(c_t(x) \neq f(x)).$$

התאמת הנתונים עבור האיטרציה הבא: $\mathcal{D}_{t+1} = \text{Adjust Distribution}(\mathcal{D}_t, \epsilon_t)$

$$3. \text{ איחוד המודלים החלשים: } C(x) = \text{combine outputs}\{c_1(x), c_2(x), \dots, c_T(x)\}.$$

יש כל מיני שיטות כיצד לבצע את השלבים השונים באלגוריתם boosting, ונפרט את המרכזיות שבהן.

Model 1,2,..., N are individual models (e.g. decision tree)



איור 2.17 – סכמת כללית של boosting. המודלים (במקרה זה מוחבר בעץ החלטה רדו, אך זה נכון לכל מודל חלש) מוחברים אחד לשני באופן שכל אחד לומד מהתפלגות המשוקלת בהתאם לשגיאות של המודלים הקודמים.

Adaptive Boosting (AdaBoost)

AdaBoost היא אחת הטכניקות הראשונות של boosting, ועל אף שהקימות טכניקות נוספות נוספות, היא בין הפופולריות ביותר בתחום (אם כי יש לה מספר לא מבוטל של וריאנטים). העצמה הכלומת בטכניקה זו נובעת מכך שגם בהינתן מספר מאפיינים רב, האלגוריתם מצליח להיפגע פחות מ"קלחת המדדיות" ולשמור על יכולות ניבוי טובות, בניגוד לאלגוריתמים אחרים של סיוג, כמו למשל SVM או אפילו רשתות נירונים.

זכור, תחת ההנחה שהקיים אלגוריתם לומד חלש (x), המטרה היא למצוא דרך להפוך אותו למודל חזק (C). באופן אינטואיטיבי היה ניתן לחשב שאפשר פשוט לאמן מספר מודלים על תת קבוצות של הדאטה המקורי (עם אפשרות לחיפויות בין תת-קבוצות), להשתמש ב-vote-majority, ובכך לשרשר את ההיפותזות של כל המודלים לפולט אחד. גישה זו מבוסנת נאיבית ופשטנית, ואינה לוקחת בחשבון מקרה בו מרבית המודלים טוענים. גישה טובה יותר תהיה לבנות מודל על בסיס חלק מהדאטה, לבחון את מידת הצלחה של המודל על יתר הדאטה, ולפי ההצלחה שלו במשימה זו לחתול מ-vote של המודל. ניתן להויסף תחכום לרענון זה, כך שבכל שלב ינתן יותר דגש איברים בדאטה שהמודלים הקודמים שגוי בסיווג שלהם, ובכך בכל שלב בו מאמנים מודל נוסף נוסף תהיה הצלחה יותר גדולה מאשר המודל הקודם. חלק זה הינו החלק האדפטיבי (Adaptive) באלגוריתם, על שמו נקרא האלגוריתם AdaBoost.

כעת, נסביר כיצד ניתן להרכיב מסווג חזק באמצעות אוסף של מסוגים חלשים עבור אוסף נתונים $\mathbb{R}^N \in X$.

1. ראשית יש לאותל משקלות באופן אחד עבור כל אחת מ- N הדוגמאות בסט הנתונים – $w_i^{t=0} = \frac{1}{N}$
2. לאחר מכן יש לבצע איטרציות באופן הבא:

בנייה מסווג אופטימלי (x) c_t ביחס לאוסף הנתונים המשוקלל.

חישוב שגיאת הסיווג של (x) c_t : $c_t(x) \neq y_i$

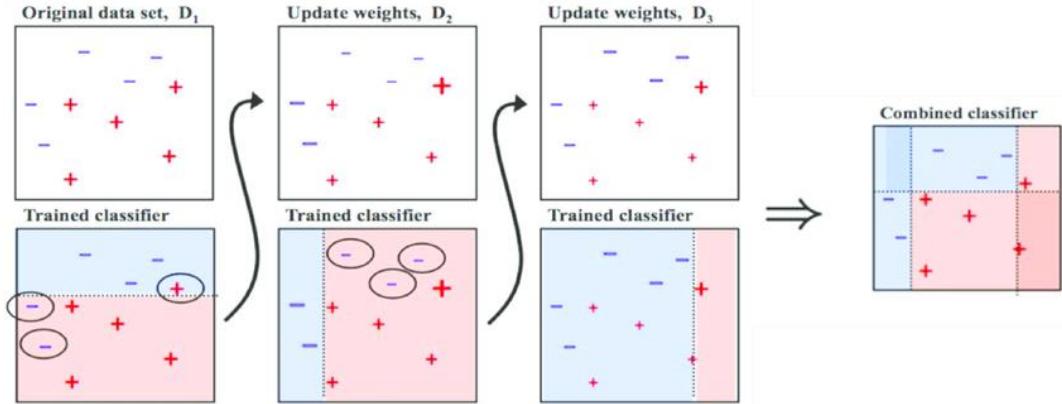
$$\text{חישוב משקל עבור מסווג זה: } \alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon}{\epsilon} \right)$$

עדכון המשקלים: $w_i^{t+1} = w_i^t \exp(-\alpha_t y_i c_t(x_i))$

$$\text{נرمול המשקלים בהתאם לסכום הכלול: } N_{t+1} = \sum_i w_i^t \rightarrow w_i^{t+1} = \frac{w_i^t}{N_{t+1}}$$

3. חישוב המסווג המשוקלל, שהוא קומבינציה לינארית של המסוגים החלשים:

$$C(x) = \text{sign} \left(\sum_t \alpha_t c_t(x) \right)$$



איור 2.18 – דוגמא לשימוש ב-AdaBoost עבור מודל סיווג בינארי

2. References

SVM:

https://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png

<https://svm.michalhaltuf.cz/support-vector-machines/>

<https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781>

https://xavierbourretsicotte.github.io/Kernel_feature_map.html

Naïve Bayes:

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://scikit-learn.org/stable/modules/naive_bayes.html

K-NN:

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

EM:

https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/13_mog.pdf

https://stephens999.github.io/fiveMinuteStats/intro_to_em.html

Hierarchical Clustering:

<https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/>

LOF:

<https://towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843>

PCA:

Saal, L.H. et al. (2007). *Proc. Natl. Acad. Sci. USA* 104, 7564–7569.

3. Linear Neural Networks

פרק זה עוסק בבעיות רגראטיה – כיצד ניתן בעזרת אוסף דוגמאות נתון לבנות מודל המסוגל לספק מידע על נקודות חדשות שיגיעו ויחסר עליהן מידע. המודלים שיוצגו בפרק זה מתמחים לדאטה שניית למצוא עבורה הפרדה לינארית, ככלומר, ניתן למצוא קווים לינאריים המחלקים את הדאטה לקבוצות שונות. החלק הראשון של הפרק עוסק ברגראטיה לינארית (Linear regression) והחלק השני עוסוק ברגראטיה לוגיסטיות (Logistic regression). לבסוף יוצג מבנה שקול לביעות הרגראטיה בעזרת רשת נירונים פשוטה, ומבנה זה יהיה הבסיס לפרך הבא העוסק ברשתות נירונים עמוקות, הבאות להתמודד עם דאטה שאינו ניתן לבצע עבורה הפרדה לינארית.

3.1 Linear Regression

3.1.1 The Basic Concept

המודל פשוט ביותר הינו linear regression. מודל זה מנסה למצוא קשר לינארי בין מספר משתנים או מספר מאפיינים. בהנחה שמתיקים יחס לינארי בין סט משתנים בלתי תלויים $\mathbb{R}^d \in x$ לבין משתנה תלוי $\mathbb{R} \in y$, ניתן לכתוב את הקשר ביניהם בצורה הבאה:

$$\hat{y} = w^T x + b = w_1 x_1 + w_2 x_2 + \dots + w_d x_d + b$$

כאשר $\mathbb{R}^d \in w$ הם המשקלים ו- $\mathbb{R} \in b$ נקרא bias.

דוגמה: ניתן לטעון כי מחיר הבתים באזורי מסוימים נמצא ביחס למספר פרמטרים: גודל הדירה,อายזה קומה היא נמצאת, וכמה שנים הבניין קיימ. תחת הנחה זו, יש לבדוק את המודל עבור דוגמאות ידועות ובכך למצוא את המשקלים וה-bias. לאחר מכן ניתן היה לקחת את המודל ולנחש את מחיר הדירה עבור בתים שונים לא ידוע, אך הפרמטרים שלהם כן נתונים.

בכדי לבנות מודל המאפשר לשער בaczורה טובה את y בהינתן סט מאפיינים, יש לדעת את המשקלים וה-bias. כיוון שהם לא ידועים, יש לחשב אותם בעזרת אוסף של דוגמאות ידועות. ראשית יש להגדיר פונקציה מחיר (Loss), הקובעת עד כמה הביצועים של מודל מסוים טובים. פונקציית המחיר היא פונקציה של הפרמטרים הנלמדים - $(b, w)^T$, והבאתה למיניהם תספק את העריכים האופטימליים של המשקלים וה-bias. פונקציית מחיר מקובלת הינה השגיאה הריבועית ממוצעת (MSE) – המחשבת את ריבוע ההפרש בין החיזוי לבין הפלט האמתי:

$$L^{(i)}(w, b) = \frac{1}{2}(y_i - \hat{y}_i)^2$$

כאשר נתונות n דוגמאות ידועות, יש לסקום את כל ההפרשים הללו:

$$L(w, b) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2}(y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n \frac{1}{2}(y_i - w^T x_i - b)^2$$

כעת בשביל למצוא את הפרמטרים האופטימליים, יש למצוא את b ו- w שմבאים את פונקציית המחיר למינום:

$$\hat{w}, \hat{b} \equiv \hat{\theta} = \arg \min L(w, b)$$

עבור המקירה הסקלרי בו $d = 1$, ככלומר יש מאפיין יחיד ומונוטם למצוא קשר בין פלט מסוים, הקשר הלינאר הוא $b + ax = \hat{y}$. עבור המקירה זהה, פונקציית המחיר תהיה:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2}(y_i - w^T x_i - b)^2$$

ובכדי למצוא אופטימום יש לגזר ולהשווות ל-0:

$$\frac{\partial L}{\partial w} = \frac{1}{n} \sum_{i=1}^n (y_i - w^T x_i - b) \cdot (-x_i) = 0$$

$$\frac{\partial L}{\partial b} = \frac{1}{n} \sum_{i=1}^n (y_i - w^T x_i - b) \cdot (-1) = 0$$

מתகבלות סט משוואות לינאריות:

$$\begin{aligned} w \sum x_i^2 + b \sum x_i &= \sum y_i x_i \\ w \sum x_i + bn &= \sum y_i \end{aligned}$$

ובכתיב מטריציוני:

$$\begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{pmatrix} \begin{pmatrix} w \\ b \end{pmatrix} = \begin{pmatrix} \sum y_i x_i \\ \sum y_i \end{pmatrix}$$

על ידי הצבה של הדוגמאות הנתונות ניתן לקבל את הפרמטרים של הקשר הlienear.

לשם הנוחות ניתן לסמן את \hat{y} -as \hat{w} כפרמטר נוספים:

$$\hat{y} = w^T x + b = (w^T b) \begin{pmatrix} x \\ 1 \end{pmatrix} = \tilde{w}^T \tilde{x}, \quad \tilde{w}, \tilde{x} \in \mathbb{R}^{d+1}$$

עבור המקרה הווקטורי יש n דוגמאות, כלומר יש n מאפיינים בלתי תלויים ומנסים למצוא את הקשר ביןם לבין פלט מסוים. במקרה זה $(x_1, x_2, \dots, x_n)^T, Y = (y_1, \dots, y_n)^T$:

$$L(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (y_i - w^T x_i)^2$$

המינימום של הביטוי הזה שקול למינימום של $\|Y - Xw\|^2$:

$$\frac{\partial L}{\partial w} = \frac{1}{n} \sum_{i=1}^n (y_i - w^T x_i) \cdot (-x_i) = 0$$

$$\rightarrow X^T (Xw - Y) = 0$$

$$\hat{w} = (X^T X)^{-1} X^T Y$$

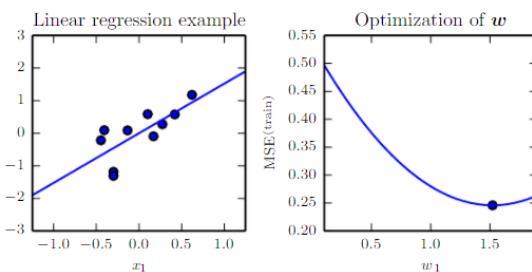
ובהינתן אוסף דוגמאות:

$$X = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}, \quad \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

אזי הפתרון של הרגרסיה הלינארית הינו:

$$\hat{w} = (X^T X)^{-1} X^T Y = \begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{pmatrix}^{-1} \begin{pmatrix} \sum y_i x_i \\ \sum y_i \end{pmatrix}$$

דוגמה למציאת קו הרגרסיה והמשקל האופטימלי עבור בעיה סקלרית:



איור 3.1 רגרסיה לינארית אופטימלית עבור אוסף דוגמאות נתון (שמאל) ואופטימיזציה עבור המשקל w ביחס לפונקציית המחיר (ימין).

3.1.2 Gradient Descent

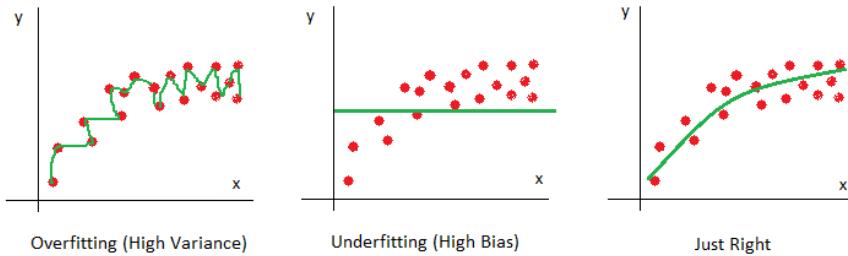
הרבה פעמים מציאת המינימום של פונקציית המחיר היא משימה קשה. דרך מקובלת להתמודד עם חישוב הפרמטר האופטימלי היא שיטת gradient descent (GD). בשיטה זו מתחילה מינוחש מסוים עבור הפרמטרים, וכל פעם מבצעים צעד לכיוון הגרדיאנט השילוי. הגרדיאנט הוא הנגזרת של הפונקציה, והוא מגדיר את הכוון שערך הפונקציה עולה בו בצורה מקסימלית. אם לוקחים את הכוון השילוי של הגרדיאנט, בעצם הולכים לכיוון בו יש את הירידה הכי גדולה, וכן כדי להגיע למינימום יש לבצע את הצעד בכיוון הגרדיאנט השילוי. בכך להימנע מהיקלעות לנקודת אוכף, מושגים איבר הנקרה (lr) (learning rate) (מוסמן באות ϵ). מבצעים את הגזירה ושינוי הפרמטרים באופן איטרטיבי עד נקודת עצירה מסוימת. באופן פורמלי, עבור ניחוש התחלתי θ_0 , בכל צעד יבוצע הקידום באופן הבא (העדרון מתבצע באופן סימולטני עבור כל θ_j):

$$\hat{\theta}_{j+1} = \hat{\theta}_j - \frac{\partial}{\partial \theta_j} L(\hat{\theta}_j)$$

קידום זה יבוצע שוב ושוב עד התכנסות לערך מסוים. כיוון שהבעיה קמורה מובטח שתהיה התכנסות למינימום, אך היא יכולה להיות איטית עקב לכך עדכונים גדולים או קטנים מדי. פרמטר ה- ϵ , learning rate, קובע את קצב ההתכנסות, אך רצוי לבחור פרמטר לא קטן מדי כדי לא להאט את ההתכנסות ולא גדול מדי כדי למנוע ההתכנסות.

3.1.3 Regularization and Cross Validation

אחד האתגרים המרכזיים של בעיית הרגרסיה (ושאר בעיות הלמידה) הוא לפתח מודל שייהיה מוצלח לא רק עבור אוסף הדוגמאות הידוע (5ט האימון), אלא שייהיה מספיק טוב גם עבור דוגמאות חדשות ולא מוכחות (קבוצת מבחן). כל מודל יכול לסביר מהטיה לשוני כיוונים – Overfitting ו-Underfitting. Underfitting הוא מצב בו יתנתה הערכת יתר לכל נקודה בסט האימון, מה שגורר מודל מסדר גובה בעל שונות גדולה. במצב זה המודל מתאים רק ל5ט האימון, אך הוא לא מצליח להסביר גם נקודות חדשות. Underfitting – מודל שלא מצליח למצואו קו מגמה המכיל מספיק מידע על הדוגמאות הנוכחיות, ויש לו רעש חזק.



איור 3.2 – נתינת משקל יתר לכל נקודה גורמת למצב בו המודל הוא מסדר גובה ובעל שונות גבוהה (שמאל). – מודל בעל רעש חזק מייצג בצורה מספיק טובה את המידוע (אמצע). מצב מאוזן – מודל בעל שעיה מינימלית, המתאר בצורה טובה את המידוע, ובנוסף נמנע משגיאת יתר עבור דוגמאות חדשות (ימין).

בכדי להימנע מהטויות אלו, יש לבצע regularization – regularization היא שיטת אילוץ המונע מהמודל להיות מוטה באופן הפגע בתוצאות. לאחר הוספת האילוץ, פונקציית המחיר תהיה בצורה:

$$\text{Regularized Loss} = \text{Loss Function} + \text{Constraint}$$

יש מספר דרכים לבצע את ה-regularization:

Ridge Regression / L2 Regularization

דרך אחת לבצע את ה-regularization היא להוסיף איבר נוסף המתיחס לריבוע הפרמטרים:

$$L(\theta) = \text{MSE}_{\text{train}} + \lambda w^T w = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (y_i - w^T x_i)^2 + \lambda \|w\|^2$$

cutet האופטימום של הביטוי הינו:

$$\hat{w} = (X^T X + \lambda I)^{-1} X^T Y$$

הוספת האילוץ גורמת לכך שבנוסף לחיזוי מדויק של משתנה המטריה, המודל מנסה למצער את ריבוע הפרמטרים, וכך לנסות להקטין עד כמה שנייתן את הערך של כל פרמטר ולהימנע מ מצב בו נתונים משקל יתר לחלק מהפרמטרים. למעשה האילוץ מקטין את השונות של המודל ובכך עשוי למנוע overfitting.

Lasso / L1 Regularization

דרך נוספת לבצע את ה-*on-hoing* היא להוסיף אילוץ המתיחס לערך המוחלט של הפרמטרים:

$$L(\theta) = \text{MSE}_{\text{train}} + \lambda |w| = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (y_i - w^T x_i)^2 + \lambda |w|$$

הוספת האילוץ מחייבת את סכום הפרמטרים להיות כמו שיותר קטן, כדי למצער כמה שנייתן את פונקציית המהיר. בפועל אילוץ זה מביא ל"רידוד משקלים" (sparse), כלומר כפיה חלק מהמקדים להיות אפס, וכך למעשה יש מעין selection – בחירת הפרמטרים המשמעותיים יותר.

ניתן לשים לב כי עבור L_2 השפעה של המשקלים על פונקציית המהיר היא ריבועית. לכן במקרה זה הרגולרייזציה תשאיר להקטין את הפרמטרים הגדולים, ובאופן כללי תנסה לדאוג לכך שככל הפרמטרים יהיו קטנים, ובאותו סדר גדול. L_1 לעומת זאת שואף להקטין את כל האיברים כמה שיותר ללא קשר לגודלם, ולהקטין פרמטר מסוים מ-10-9 יש את אותה השפעה כמו הקטנה של פרמטר מ-1000 ל-999. לכן במקרה זה הרגולרייזציה תגרום לפרמטרים הפחות חשובים להתאפס, והמודל נהייה פשוט יותר.

Elastic Net

ניתן לשלב בין Ridge Regression לבין Lasso, וכך לנסות לכון את המודל עבור היתרונות של כל שיטה – גם להימנע מנתינית משקל יתר לפרמטרים וגם ניסיון לאפס פרמטרים, וכך לקבל מודל פשוט ככל הניתן. פונקציית המהיר במקרה זה תהיה מהצורה:

$$L(\theta) = \text{MSE}_{\text{train}} + \lambda \|w\|^2 + \lambda_2 |w|$$

עבור כל אחת מהדריכים, יש למצוא את הפרמטר λ האופטימלי עבור ה-*on-hoing* (במקרה של Elastic Net – Cross validation). שיטה מקובלת למציאת λ האופטימלי היא – *leave-one-out cross validation*. הפרמטר λ הוא למעשה λ_2 , והוא נקבע על סט האימון ל- k קבוצות, אימון כל תתי הקבוצות מלבד אחת, ואז בדיקת הפרמטרים שהתקבלו בשלהי האימון על הקבוצה שנותרה. בכל איטרציה מוצאים חלק מסוים הדוגמאות והופכים אותן לקבוצה מבחון, וכך מוצאים את הפרמטר λ האופטימלי המונע מהמשקלים להציג fitting (בדרך כלל לווקחים את הממוצע של כל ה- λ מכל האיטרציות). נפוץ להשתמש ב- $k=5$, ולמעשה עבור בחירה טיפוסית זו יהיו 5 איטרציות, שבסך אחת מהן האימון יבוצע על 80% מסט האימון, ולאחר מכן תבוצע הבדיקה של הפרמטרים שנלמדו על ה-20% הנדריכים.

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

איור 3.3 Cross validation עם חלוקה ל-5 קבוצות ($k=5$). בכל פעם קבוצה אחת משמשת ל-validation (הקבוצה הכהולה).

בחירה של $n = k$ נקראת *leave-one out cross validation* כיון שלמעשה בכל איטרציה יש דוגמא אחת בלבד שלא נכללת בסט האימון ועליה מתבצעת הבדיקה של הפרמטרים שנלמדו.

3.1.4 Linear Regression as Classifier

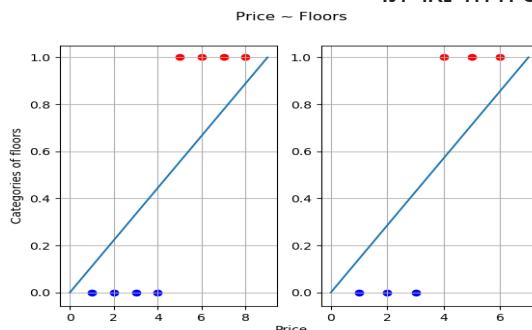
משימת סיווג מוגדרת באופן הבא: בהינתן סט פרמטרים מסוימים $\{x_1, \dots, x_m\}$ X השיר לתצפית מסוימת, יש לסתור אותו לאחת מתוך m קטגוריות אפשריות: $\{1, \dots, y\}$. לדוגמה: נתונה תמונה בעלת m פיקסלים המייצגת חייה, ויש לקבוע איזו חייה היא, כאשר הבחירה נעשית מתוך הווקטור u . לרוב הווקטור u מורכב ממספרים שלמים, שכן

אחד מהם מייצג בחירה מסוימת. בדוגמה של החיות, ניתן לחתך לדוגמא 3 = m , כלומר $\{1,2,3\} \in \mathcal{Y}$, כאשר המספרים מייצגים את סט החיות $\{\text{dog, cat, chicken}\}$.

ניתן להשתמש במודל של רגסיה לינארית למשימות של סיוג. עבור המקרה של $2 = m$, יש שתי קטגוריות אפשריות, ולבסוף יש צורך למפות כל נקודה לאחת משתי הקטגוריות. בעזרת רגסיה לינארית ניתן לבצע מיפוי מ- \mathbb{R}^l , $\{0,1\}$, כלומר כל נקודה במרחב ממופה לאחד משני ערכי אפסריים: קבועים ערך סף $T = 0.5$, ועבור נקודה חדשה x_n בזדוקים מה היחס בין הביטוי $b + w^T x_n < 0.5$ לבין ערך הסף. אם הנקודה החדשה מקיימת: $b + w^T x_n < 0.5$ אז הנקודה החדשה תתויג בקטgorיה 1. אחרת, הנקודה החדשה תתויג בקטgorיה 0. באופן פורמלי:

$$y = \text{sign}(w^T x_{\text{new}} + b - 0.5) = \begin{cases} 1 & w^T x_{\text{new}} + b > 0.5 \\ 0 & w^T x_{\text{new}} + b < 0.5 \end{cases}$$

הבחירה בערך הסף $T = 0.5$ נובעת מכך שיש שתי קטגוריות $\{0,1\}$, וערך הסף נקבע להיות נקודת המיצע ביניהן. בדוגמה: נתונים z בתים ועבור כל אחד מהם ידוע מה מחירו והאם יש בו קומה אחת או שתיים. כעת רוצים לבדוק את היחס בין המחיר למספר הקומות ולקבוע עבור מחיר בית נתון מה מספר הקומות שלו. במקרה זה יש 2 קטגוריות: $\{1,0\} = \{1 \text{ floor}, 2 \text{ floors}\}$, ויש להיעזר במידע על z הבטים בכך לבנות מודל מסווג. הדרך לעשות זאת היא לבצע רגסיה לינארית, ואז כשבוחנים מחיר של בית, יש לבדוק אם $b + w^T x \cdot \text{price} < 0.5$ או קטן ממנו, כאשר (w, b) הם הפרמטרים של הרגסיה הלינארית.



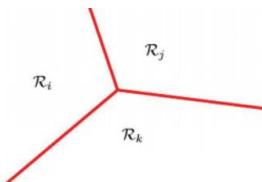
איור 3.4 רגסיה לינארית כמסוג ביבנאי: מיפוי הנקודות בהתאם למיקומן ביחס לקו ההפרדה של הרגסיה הלינארית. בדוגמה הימנית ערך הסף מתkeletal עבור $x_T = 3.5$, כלומר $3.5 + b = 0.5$. בדוגמה השاملית ערך הסף מתkeletal עבור $x_T = 4.5$. עבור כל בית חדש, בהינתן מחיר ניתן יהיה לסוג אותו לאחת משתי הקטגוריות, בהתאם ליחסו לערך הסף 0.5.

עבור m נקודות ידועות – $(x_1, y_1), \dots, (x_n, y_n)$, $y_i \in \{0,1\}$, פונקציית המחיר הינה:

$$L(\theta) = \sum_{i=1}^n 1_{\{y_i \neq \text{sign}(w^T x_i + b + 0.5)\}}$$

הfonקציה $L(\theta)$ מכילה סט של פרמטרים – $(b, w) = \theta$. כיוון שהנגזרת של הפונקציה לפי כל אחד מהפרמטרים שלה w לא תלויה רק באותו פרמטר, קשה למצאו את θ המבאים למינימום את $L(\theta)$.

ניתן להרחיב את המսוג גם עבור מקרים בהם יש יותר משתי קטגוריות (multi-class). סט האימון תראה כמו בדוגמה הביבנאי, ואילו y מכילCutת m קטגוריות: $\{m, \dots, 1, 0\} = \{y_1, \dots, y_m\}$. במקרים אלו יש ליצור מספר קווים לינאריים, המפרידים בין אזורים שונים. כדי לחשב את הקווים מבצעים התהליך שנקרא all versus one, בו בכל פעם לוקחים קטגוריה אחת ובזוקרים מהו קו ההפרדה בין לביון שאר הקטגוריות. הפרמטרים הנלמדים של קווי ההפרדה יהיו הסט המורכב מכל הפרמטרים של הרגסיה: $\{w_1, b_1, w_2, b_2, \dots, w_m, b_m\} = \theta$.



איור 3.5 רגסיה לינארית מרובה – הפרדה בין מספר אזורים שונים על ידי מספר קווים לינאריים.

במקרה זהה, נקודה חדשה תסוג לקטgorיה לפי הביטוי הבא:

$$y(x) = \arg \max_i (w_1^T x + b_1, \dots, w_m^T x + b_m)$$

וכל אזכור יוגדר לפיה:

$$R_i = \{x | y(x) = i\}$$

בדומה ל McKenna ה binnari, פונקציית המחיר תהיה:

$$L(\theta) = \sum_{i=1}^n 1_{\{y_i \neq \hat{y}_i\}} \text{ s.t. } \hat{y}_i = \arg \max_i (w_i^T x + b_i)$$

המסוג האופטימלי יהיה וקטור הפרמטרים המביא את פונקציית המחיר למינימום:

$$\hat{\theta} = \arg \min_{\theta} L(\theta)$$

גם במקרה זה, כיוון שהנגזרת של פונקציית המחיר לפי כל פרמטר אינה תלולה רק באותו פרמטר, בפועל קשה למצאו את θ האופטימלי המביא את $L(\theta)$ למינימום.

3.2 Softmax Regression

3.2.1 Logistic Regression

המסוג הנוצר מהרגסיה הלינארית הינו "מסוג קשה" – כל דוגמא חדשה שמתקבלת מסווגת לקטגוריה מסוימת, ואין שום מידע עד כמה הדוגמאacea זו דומה לקטגוריות האחרות. מסוג קשה אינו מספיק טוב עבור מגוון בעיות, בהן מעוניינים לדעת לא רק את הקטgorיה, אלא גם מודיע נוסף על היחס בין הדוגמא החדשיה לבין כל הקטגוריות. לדוגמה: בהינתן מידע של גידול מסוים רוצים לדעת אם הוא ממאייר או שפיר. במקרה זה ההכרעה היא לא תמיד חד משמעות, ויש עניין לדעת מה הסיכוי של הגידול להיות ממאייר או שפיר, שהרי יתכן שהטיפול יהיה שונה בין מקרה בו יש 1% שהגידולacea זהה הוא מסווג בין מקרה בו יש 40% שהגידולacea הוא מסווגacea זהה. כדי להימנע מהסתואוג הקטgorיו, יש ליצור מודל הסתברותי, בו כל קטgorיה מקבלת הסתברות מסוימת. אחד המודלים הבסיסיים הינו רגסיה לוגיסטיבית (Logistic regression). עבור המסוגacea זהה ראשית יש להגדיר את פונקציית הסיגמוואיד:

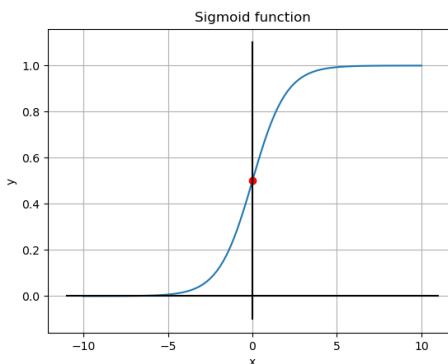
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

פונקציה זו רציפה על כל הישר, ובעצמותה ניתנת להגדיר מסווג עבור המקרה ה binnari:

$$p(y = 1|x; \theta) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

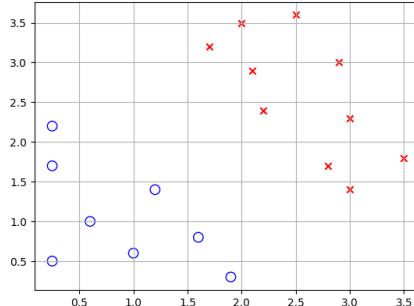
$$p(y = 0|x; \theta) = 1 - \sigma(w^T x + b) = 1 - \frac{1}{1 + e^{-(w^T x + b)}} = \frac{e^{-(w^T x + b)}}{1 + e^{-(w^T x + b)}}$$

המסוג לוקח את קן החלטה הלינארית, ומעבר אליו בפונקציה המחזירה ערך בטוחה $[0, 1]$, כאשר הערך המוחזר הוא ההסתברות להיות בקטgorיה מסוימת. בכך להבין יותר טוב את משמעות המסוג, יש להסתכל על גרף הסיגמוואיד:



איור 3.6 גרף הפונקציה: $y = \frac{1}{1+e^{-x}}$. הנקודה $(0, 0.5)$ מודגשת באדום.

כאשר הפונקציה $b = w^T x + b = \theta$ שווה בדיק ל-0, אז $w^T x + b = 0.5\sigma$. המשמעות של התוצאה זו היא שאם עבור סט פרמטרים x_n מתקיים $w^T x_n + b > 0$, אז ההסתברות של הנקודה זו להיות משוכנת לקטגוריה 1 גדולה מחצי, כיוון $0.5 < w^T x_n + b < 0.5\sigma$. באופן סימטרי אם $w^T x_n + b < 0$, אז ההסתברות של הנקודה x_n להיות משוכנת לקטגוריה 1 קטנה מחצי. כעת עליה השאלה מתי $w^T x + b = 0$, והתשובה היא זהה תלוי בכך ה הפרדה שבין הקטגוריות. לשם המראה נניח ונתונות מדידות על שני פרמטרים - x_1, x_2 , ועבור כל נקודה (x_1, x_2) נתון גם מה הקטgorיה שלה ($y \in \{0,1\} = \{\text{blue o}, \text{red x}\}$):



איור 3.7 דוגמא למספר מדידות התלויות בשני פרמטרים x_1, x_2 , ומושכות לאחת משתי קטגוריות: $\{y \in \{0,1\} = \{\text{blue o}, \text{red x}\}$.

כיוון שנתנו נתונת הנקודות, ניתן לייצר בעזרתו קו רגסצי. לצורך הדוגמא נניח שיש שלושה פרמטרים והם מקיימים: $w^T = [1, 1, -3]^T = [w_1, w_2, b]$. הפרמטרים האלו מרכיבים את הקו הלינארי $3x_2 - x_1 + b = 0$, כלומר, עבור כל נקודה אם מתקיים $3x_2 - x_1 > 0$ היא תהיה מסווגת כ-' red x ', אחרת היא תהיה מסווגת כ-' o blue '. קו זה הוא למעשה חישוב הפרדה בין שתי נקודות חדשות. קו זה מקיים את המשוואה $w^T x + b = 0$, וכן את המשוואה $w^T x + b = 1$, כלומר קו הפרדה נמצא בפערו של 1. שגם מקיימת $w^T x + b = 0$, המשמעות היא שנקודה זו נמצאת בבדיקה על קו ה הפרדה. נקודה צוואר קבל הסתברות של 50% להיות משוכנת לכל אחת מהקטגוריות. ככל שהנקודה החדרה תתרחק מקו ה הפרדה, כך הביטוי $w^T x + b$ יתרחק מה-0, וכך גם $(w^T x + b)^2$ יתרחק מהערך חצי ויתקרב לאחד מערכי הקצה 0 או 1, והמשמעות היא כמובן שיש יותר סיכוי שנקודה זו שייכת לקטגוריה אחת ולא אחרת.

כמובן שניתן לקחת גם את המשוואה ההסתברותי זהה ולהשתמש בו כמסוג קשה: עבור דוגמא חדשה לוקחים את ההסתברויות שלה לכל אחת מהקטגוריות, ומוסוגים את הדוגמא לקטגוריה בעלת ההסתברות הגבוהה ביותר. במקרה הבינארי וקטור ההסתברויות הינו $[p(y=1|x), p(y=0|x)]$, והקטגוריה של \hat{y} תהיה $\hat{y} = \arg \max_i \hat{y}_i$ שזהו ה- \hat{y} בעל ההסתברות הגדולה ביותר.

3.2.2 Cross Entropy and Gradient descent

בכדי למצוא את הפרמטרים $(w, b) = \theta$ האופטימליים בהינתן n דוגמאות, ניתן להחליף את קרייטריו השגיאה הריבועית הממוצעת בקרייטריו אחר למazure פונקציית המחיר – Cross entropy. קרייטריו זה אומר שיש לחייב לפחות מינימום את מינוס הלוג של סך הדוגמאות (הביטוי נובע משערוך הנראות המרבית – [Maximum likelihood](#)):

$$-\log P(Y|X; \theta) = -\frac{1}{n} \sum_{i=1}^n \log p(y_i|x_i; \theta) = L(\theta)$$

למעשה, יש למצוא את סט הפרמטרים $\hat{\theta}$ המביא את הביטוי \hat{y} למינימום: $\hat{\theta} = \arg \min_{\theta} L(\theta)$.

בכדי לחשב את הביטוי יש לפתח קודם את הביטוי עבור נגזרת הסיגמודואיד:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \rightarrow \frac{\partial \sigma(z)}{\partial z} = \frac{-1}{(1 + e^{-z})^2} \cdot e^{-z} \cdot (-1) = \frac{1}{1 + e^{-z}} \cdot \frac{e^{-z}}{1 + e^{-z}} = \sigma(z)(1 - \sigma(z))$$

זכור, $1 - \sigma(z) = p(y=0|x; w, b)$, ולכן יש לחשב גם את הנגזרת עבור (z) :

$$\frac{\partial(1 - \sigma(z))}{\partial z} = -\sigma(z)(1 - \sigma(z))$$

בהתאם, הנגזרות של לוג סיגמודואיד הן:

$$\frac{\partial \log \sigma(z)}{\partial z} = \frac{1}{\sigma(z)} \cdot \frac{\partial \sigma(z)}{\partial z} = (1 - \sigma(z))$$

$$\frac{\partial \log(1 - \sigma(z))}{\partial z} = \frac{1}{1 - \sigma(z)} \cdot \frac{\partial(1 - \sigma(z))}{\partial z} = -\sigma(z)$$

כעת יש לשים לב שהנגזרת של $\log p(y=1|z)$ הינה $1 - \sigma(z) = y - \sigma(z)$, והנגזרת של $\log p(y=0|z)$ הינה $\sigma(z) - y_i$. לכן אם $y \in \{0,1\}$, אז ניתן לרשום בקיצור: $\frac{\partial}{\partial \theta} \log p(y_i|z) = y_i - \sigma(z)$. במקרה של רגסיה לוגיסטי, מוחפשים את הנגזרת של $\frac{\partial}{\partial \theta} \log p(y_i|x_i; \theta)$, ולפי הפיתוח המקדים ניתן לרשום את זה כך:

$$\frac{\partial}{\partial \theta} \log p(y_i|x_i; \theta) = (y_i - \sigma(w^T x + b)) \cdot \frac{\partial}{\partial w} (w^T x + b) = (y_i - p(y_i = 1|x_i; \theta)) \cdot x_i$$

כעת לאחר הפיתוח ניתן לחזור חזרה לביטוי $\arg \min_{\theta} L(\theta)$ ולהציב:

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{1}{n} \sum_{i=1}^n \frac{\partial}{\partial \theta} \log p(y_i|x_i; \theta) = -\frac{1}{n} \sum_{i=1}^n (y_i - \sigma(w^T x + b)) x_i = -\frac{1}{n} \sum_{i=1}^n (y_i - p(y_i = 1|\theta; x)) x_i$$

3.2.3 Optimization

בדומה לרגסיה לינארית, גם כאן חישוב הערך האופטימלי של $\hat{\theta}$ יהיה איטרטיבי בשיטת

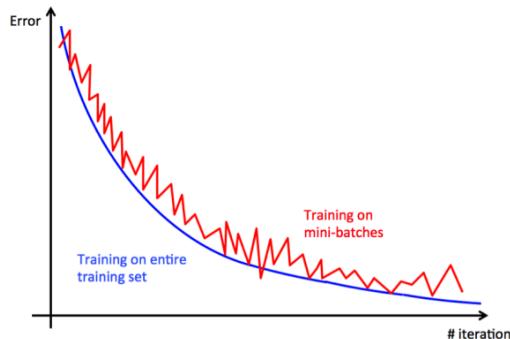
$$\hat{\theta}_{j+1} = \hat{\theta}_j - \epsilon \cdot \frac{\partial}{\partial \theta_j} L(\theta)$$

כאשר ϵ הוא הפרמטר של ה-rate learning. כיוון שפונקציית המחר $L(\theta)$ קעורה, מובהק שתהיה התוכנות ל- $\hat{\theta}$.

במקרים רבים הدادה סט הוא גדול, ולחשב את הגרדיינט עבור כל הدادה צורר הרבה חישוב. בכל צעד של קידום ניתן לחשב את הגרדיינט עבור חלק מהدادה, וביצוע את הקידום לפי הכוון של הגרדיינט הנוכחי. למשל ניתן לבחור באופן אקראי נקודה אחת ולחשב עליה את הגרדיינט. בחירה זו נקראת Stochastic Gradient Descent (SGD), כיוון שבכל צעד יש בחירה אקראיית של נקודה. חישוב בשיטת SGD יכול לגרום לשונות גדולות מתקדם, ולכן עדיף לחתך מספר נקודות. חישוב הגרדיינט בשיטה זו נקרא mini-batch learning (לעומת חישוב המתבצע על כל הدادה הנקרא batch learning). באופן פורמלי, הגרדיינט בשיטת mini-batch הינו:

$$\frac{\partial L}{\partial \theta} = \frac{\partial}{\partial \theta} \left[-\frac{1}{|V|} \sum_{i \in v} \log p(y_i|x_i; \theta) \right] \approx \frac{\partial}{\partial \theta} \left[-\frac{1}{n} \sum_{i=1}^n \log p(y_i|x_i; \theta) \right]$$

אמנם כל צעד הוא קירוב לגרדיינט, אך החישוב מאד מהיר ביחס לגרדיינט המדויק, וזה יתרון ממשמעותי שיש לשיטה זו על פני שיטות אחרות. בנוסף, ניתן להוכיח שבשיטה זו מתקבל משערך חסר הטיה לגרדיינט האמיתי.



איור 3.8 השגיאה של $\hat{\theta}$ כפונקציה של האיטרציות בשיטת gradient descent בשיטת batch learning. הגרף הכהול מייצג את השגיאה בשיטת בה הגרדיינט בכל צעד מחושב על כל הدادה, והגרף האדום מייצג את השגיאה בשיטת mini-batch learning, בה בכל צעד הגרדיינט מחושב רק על חלק מהدادה הנבחר באופן אקראי.

בדומה ל-linear regression, גם ב-logistic regression קיימים עניין הרגולרייזציה, שנועד למנוע מהמודל לתת משקל יתר לכל נקודה (Overfitting) או לא ליצג את הדadata בצורה מספיק טובה (Underfitting). ניתן להויסף למשל אילוץ ביחס לריבוע הפרמטר:

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^n \log p(y_i|x_i; \theta) + \lambda \|\theta\|^2$$

ואז הנגזרת הינה:

$$\frac{\partial L}{\partial \theta} = -\frac{1}{n} \sum_{i=1}^n \frac{\partial}{\partial \theta} \log p(y_i|x_i; \theta) + 2\lambda \|\theta\|$$

הפרמטר λ האופטימלי מחושב על ידי ביצוע Cross validation על הדadata.

3.2.4 SoftMax Regression – Multi Class Logistic Regression

בדומה ל-linear regression, גם ב-logistic regression ניתן להרחב את המסוג גם עבור multi-class (מקורה בו יש יותר משתי קטגוריות). גם בהכללה למקרה מרובה קטגוריות יש מיפוי של כל קטgorיה להסתברות בתחום $[0, 1]$. רק כעת הפונקציה בה משתמשים היא SoftMax במקומ סיגמואיד. SoftMax היא פונקציה המופעלת על סדרה, והיא מוגדרת כך:

$$\text{SoftMax}(z_1, \dots, z_n) = \left(\frac{e^{z_1}}{\sum_{j=1}^n e^{z_j}}, \dots, \frac{e^{z_n}}{\sum_{j=1}^n e^{z_j}} \right)$$

המונה מחשב אקספוננט בחזקת z_i , והמננה מנורמל את התוצאה, כך שסך כל האיברים לאחר הפונקציה הוא 1. במקרה בו יש מספר קטגוריות – יש מספר קווי הפרדה, ולכל אחד מהם יש סט פרמטרים θ . בהינתן נקודה חדשה, ניתן בעזרת SoftMax לחתה הסתברות לכל קטgorיה:

$$p(y = i|x; \theta) = \text{SoftMax}(w_1^T x + b, \dots, w_n^T x + b_n)$$

ואם מעוניינים לקבל סיווג קשה, לוקחים את האיבר בעל ההסתברות הגבוהה ביותר. גם במקרה זה פונקציית המבחן תהיה ה-Cross entropy:

$$L(\theta) = -\frac{1}{n} \sum_{i=1}^n \log p(y_i|x_i; \theta)$$

נחשב את הנגזרת של הביטוי בתחום הסכום לפ' θ_i :

$$\begin{aligned} \frac{\partial}{\partial \theta_i} \log p(y_i = s|x_i; \theta) &= \frac{\partial}{\partial \theta_i} \log \frac{\exp(w_s^T x + b)}{\sum_{j=1}^n \exp(w_j^T x + b)} = \frac{\partial}{\partial \theta_i} \left(w_s^T x + b - \log \sum_{j=1}^n \exp(w_j^T x + b) \right) \\ &= 1_{\{i=s\}} x - \frac{\exp(w_i^T x + b) x}{\sum_{j=1}^n \exp(w_j^T x + b)} = (1_{\{i=s\}} - p(y = i|x)) x \end{aligned}$$

כאשר הסימון $1_{\{i=s\}}$ הינו 1 אם $i = s$ ו-0 אחרת. כעת ניתן להציב את הביטוי האחרון בנגזרת של $L(\theta)$:

$$\frac{\partial L}{\partial \theta_i} = -\frac{1}{n} \sum_{t=1}^n (1_{\{y_t=k\}} - p(y_t = i|x_t; \theta)) x$$

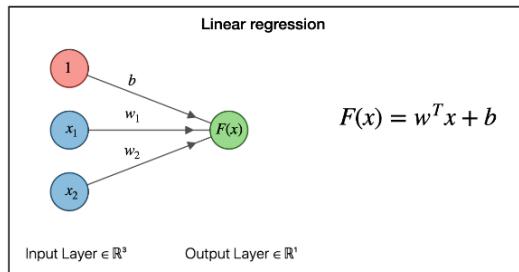
כעת ניתן לחשב את θ האופטימלי בשיטת gradient descent

$$\theta_{i+1} = \theta_i - \epsilon \frac{\partial L}{\partial \theta}$$

3.2.5 SoftMax Regression as Neural Network

לשיטת logistic regression יש מספר יתרונות: היא יחסית קלה לאימון, מספקת דיוק טוב לדאטה-5טיטים פשוטים, יציבה ל-overfitting, מציעה סיווג הסתברותי ומתחילה גם למקורה בו יש יותר משתי קטגוריות. עם זאת, יש לה חסרוןמשמעותי – קווי הפרדה של המודל הינם לינאריים, וזו הפרדה שאינה מספקת טוביה עבור בעיות מורכבות. יש מגוון בעיות בהן על מנת לבנות מודל המסוגל להפריד בין קטגוריות שונות, יש צורך במנגנון הפרדה לא לינארית.

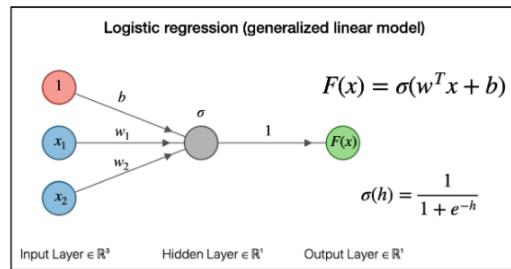
דרך מקובלת לבניית מודלים לא לינאריים היא שימוש ברשות נירונים עמוקות, ובכך להבין את הקונספט של זה. היטב, ראשית יש ליצג את המודלים הלינאריים כשכבה של נירונים, כאשר המודל הזה שקול לחילוטין לכל מה שהוא גוד כה. בעיית Linear regression לוקחת סט של מאפיינים ומכפילה כל אחד מהם במשקל, ולאחר מכן סוכמת את כל האלמנטים (בצירוף bias) לכדי משתנה יחיד הקובלע מה הקטgorיה של סט זה. ניתן ליצג את המודל על ידי התיאור הגרפי הבא:



איור 3.9 יציג רגסיה לינארית כרשת נירונים עם שכבה אחת.

בתיאור זה יש 2 מאפיינים המהווים את ה-*setpoint*, וכל אחד מהם מחובר למצאה בתוספת הכפלת במשקל. בנוסף יש bias, ובצירוף המאפיינים המכופלים במשקלים וה- bias מתקובל למצאה: $b = w_1x_1 + w_2x_2 + b = w^T x + b$. כל עיגול באIOR נקרא ניירון מלאכותי – אלמנט היכול לקבל קלט, לבצע פעולה חישובית ולהוציא קלט.

רגסיה לוגיסטיבית ניתנת לתיאור באופן דומה, כאשר הנירונים של סט ה-*setpoint* לא מחוברים ישירות למצאה אלאüberירים דרך סיגМОואיד במקורה הבינארי או דרך SoftMax במקורה בו יש יותר משתי קטגוריות:



איור 3.10 יציג רגסיה לוגיסטיבית כרשת נירונים עם שכבה אחת.

מלבד המעבר לפונקציית הסיגמוואיד, יש הבדל נוסף בין היצוג של הרגסיה הלינארית ליצוג של הרגסיה הלוגיסטיבית: בעוד הרגסיה הלינארית מספקת למצאה מספר יחיד במוחא (מוסוג קשה), הרגסיה הלוגיסטיבית מספקת למצאה וקטור באורך של מספר הקטגוריות, באופן צזה שלכל קטgorיה יש הסתברות מסוימת שה-*setpoint* שייר לאותה קטgorיה.

בפרק הבא יציג מבנה בעל מספר שכבות של נירונים, כאשר בין שכבה לשכבה יש פונקציה לא לינארית. באופן זה המודל שיתקיים יהיה מיפוי של סט מאפיינים באופן לא לינארי לוקטור הסתברויות למצאה. הगמישות של המודל מאפשר להתמודד עם משימות בעלות דатаה מורכב.

References

[https://www.deeplearningbook.org/](https://www deeplearningbook org/)

Fitting:

<https://www.calloftechies.com/2019/08/solving-overfitting-underfitting-in-machine-learning.html>

Cross validation:

https://scikit-learn.org/stable/_images/grid_search_cross_validation.png

linear regression:

מצגות מהקורס של פרופ' יעקב גולדברג

<https://joshuagoings.com/2020/05/05/neural-network/>

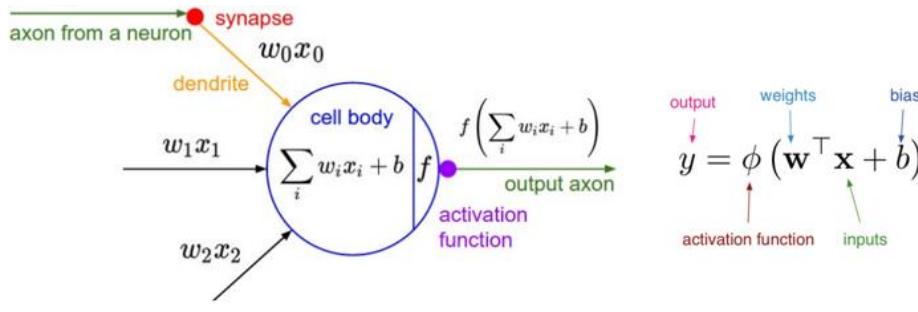
4. Deep Neural Networks

פרק זה עוסק ברשתות נירונים עמוקות. רשת נירונים הינה חיבור של יחידות עיבוד בסיסיות (נירונים מלאכותיים) על ידי משקלים ופונקציות לא לינאריות. רשת נירונים מורכבת ממספר יחידות עיבוד אחד. לאחר הצגת הבסיסי הרעיוני והפורמלי, יסביר כיצד ניתן לחשב את המשקלים של הרשת בצורה ישרה בעזרת מבנה המוכר Computational Graph הלמידה ושיטות לבחון עד כמה המודל המתאים אכן מדויק.

4.1 MLP – Multilayer Perceptron

4.1.1 From a Single Neuron to Deep Neural Network

ראשית יש לתאר את המבנה של יחידת העיבוד הבסיסית – נירון מלאכותי. יחידת עיבוד זו נקראת כך עקב הדמיון שלה לנירון פיזיולוגי – יחידת העיבוד הבסיסית במוח האדם האנושי. הנירון יכול לקבל מספר קלטים ולחבר אותם, ואז להעביר את התוצאה בפונקציית הפעלה (activation function) שאינה בהכרח לינארית. באופן סכמטי ניתן לתאר את הנירון הבא:

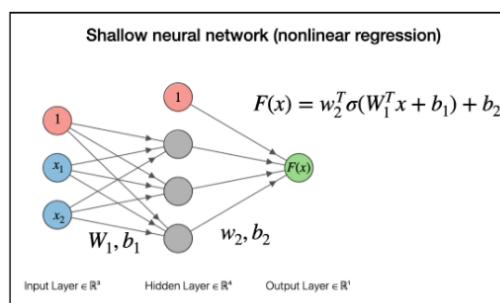


איור 4.1 יציג של נירון מלאכותי, המתקבל מוסף סום איבר ומעביר את התוצאה בפונקציית הפעלה.

הקלט של הנירון הוא סט input מוכפל במשקלים: $x^T w \in \mathbb{R}^d$ כאשר $w \in \mathbb{R}^d$, $x \in \mathbb{R}^d$, w , b איבר בדרכם, ומתקבל הביטוי $b + \sum_{i=1}^d w_i x_i$. לאחר מכן הסכום עובר דרך פונקציית הפעלה, ומתקבל המוצא ($f(\sum_{i=1}^d w_i x_i + b)$). במקרה הפרטי בו פונקציית הפעלה היא סיגמודיד/SoftMax והmoza'a לא מחובר לשכבה נוספת, אז למעשה מקבלים את הרגรสיה הלוגיסטיות.

במקרה בו הנירונים המוחברים ל-*שכבת* אינם מהווים את המוצא אלא הם מוכפלים במשקלים ומחברים לשכבה נוספת נירונית, אז השכבה המוחברת ל-*שכחת* נקראת שכבה חבויה (hidden layer). אם יש יותר שכבה חבויה אחת, הרשת מכונה רשת נירונים עמוקה. במקרה בו יש לפחות שכבה חבויה אחת, הקשר בין הכוונה למוצא אינו לינארי, וזה הינו שף למודל זה. נתבונן במקרה של שכבה חבויה וnochesh את הקשר בין הכוונה למוצא: נסמן את המשקלים בין הכוונה לבין השכבה החבויה ב- b_1, w_1 ואת המשקלים בין השכבה החבויה לבין המוצא ב- b_2, w_2 , וכן את היחס בין השכבה החבויה ל- y : $y = f_1(w_1^T x + b_1) + b_2$. ביטוי זה עובר בפונקציית הפעלה נוספת ומתקבל המוצא:

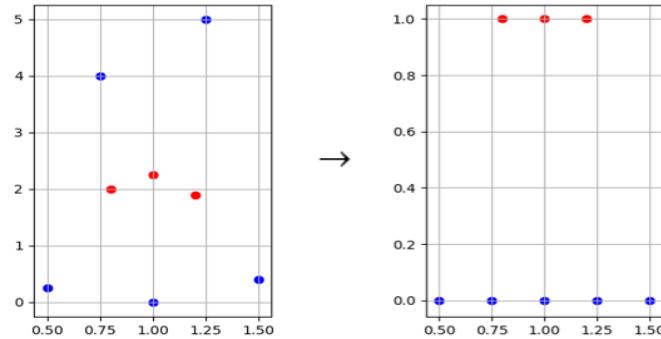
$$\hat{y} = f_2(w_2 \cdot f_1(w_1^T x + b_1) + b_2)$$



איור 4.2 רשת נירונים בעלת שכבה חבויה אחת.

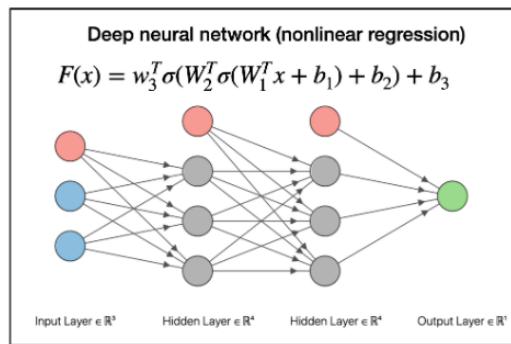
חשוב להזכיר שמטרת הרשת היא לבצע פעולות לא לינאריות על ה-*שכחת* כך שהיא יסודר באופן חדש הניתן להפרדה לינארית. למעשה מבדילים את הפרדה הלינארית הנעשה באמצעות הרגרסיה, אלא מבצעים פניה שלב מקדים של העתקה לא לינארית. תהליך זה נקרא למדידת "יצוגים" (representation learning), כאשר בכל שכבה

מנסם ללמידה יציג פשטוט יותר לדאטה על מנת שהוא יוכל להיות מופרד באופן לינארי. המיקוד של הרשות הוא אינו במשימת סיווג אלא במשימת ייצוג, כך שבסתומו של דבר ניתן יהיה לסייע את הדאטה באמצעות סיווג לינארי פשוט (רגרסיה לינארית או לוגיסטיבית).



איור 4.3 העתקה לא לינארית של דוגמאות על ידי המשוואה $\hat{y} = \begin{cases} 1, & \text{if } 3 \leq (x^2 + y^2) \leq 8 \\ 0, & \text{else} \end{cases}$. העתקה זו מאפשרת להבחין בין הדוגמאות באמצעות קו הפרדה לינארי.

כאשר מחברים יותר שכבה חביה אחת, מקבלים רשת עמוקה. החיבור בין השכבות נעשה באופן זהה – הכפלה של משקלים, סכימה והעברה בפונקציית הפעלה.



איור 4.4 רשת נירונים בעלת שתי שכבות חביות.

רשת נירונים בעלת לפחות שכבה חביה אחת הינה *universal approximation*, כלומר, ניתן לייצג בקירוב כל התפלגות מותנית באמצעות הארכיטקטורה הזאת. ככל שהרשת יותר עמוקה, כך היכולת שלה להשיג דיוק טוב יותר גדלה.

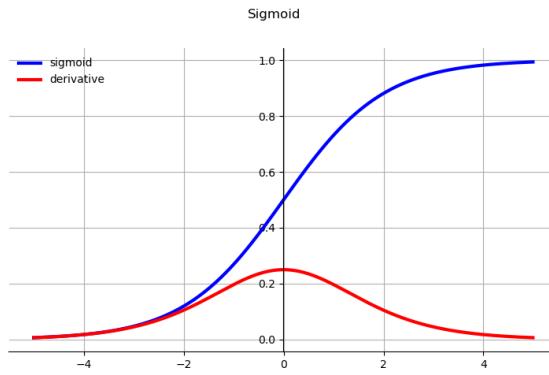
4.1.2 Activation Function

האלמנט המרכזי בכל ניירון הוא פונקציית הפעלה, ההופכת אותו ליחידת עיבוד לא לינארית. יש מספר פונקציות הפעלה מקובלות – Sigmoid, tanh, ReLU –

Sigmoid

פונקציית הסיגמאיד הוצגה בפרק של רגרסיה לוגיסטיבית, ועתנו נרחיב עליה. הפונקציה והנגזרת שלה הן מהצורה:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad \frac{\partial}{\partial z} \sigma(z) = \sigma(z)(1 - \sigma(z))$$



איור 4.5 פונקציית סיגמואיד והנגזרת שלה.

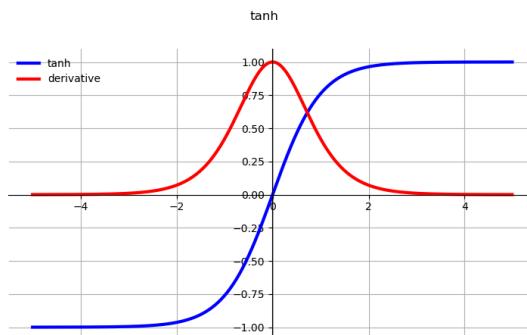
יש לפונקציה זו שלושה חסרונות:

- א. עבור ערכים גדולים, הנגזרת שואפת ל-0. זה כמובן יוצר בעיה בחישוב הפרמטר האופטימלי בשיטת Gradient descent, שהרי בכל צעד התוספת תליה בגרדיינט, ואם הוא מתאפס – לא ניתן לחשב את הפרמטר האופטימלי.
- ב. הסיגמואיד לא ממורכז סביב ה-0, וזה יוצאה בעור דатаה שאינו מנורמל.
- ג. הן הפונקציה והן הנגזרת דורשות חישוב של אקספוננט, ובאופן יחס' זו פעולה יקרה לחישוב.

tanh

פונקציית טנגנס היפרבולי הינה מהצורה:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad \frac{\partial}{\partial z} \tanh(z) = 1 - (\tanh(z))^2$$



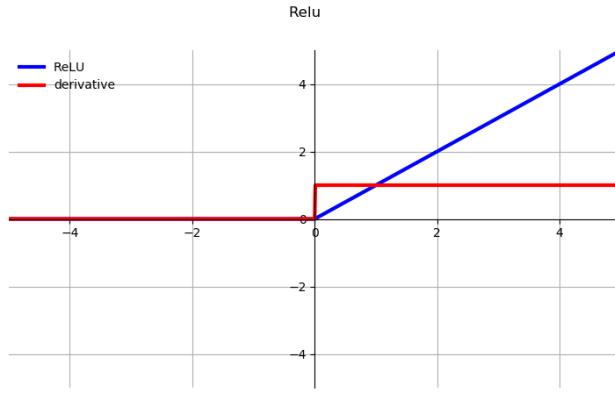
איור 4.6 פונקציית טנגנס היפרבולי והנגזרת שלה.

גם בפונקציה זו יש את הבעיות של חישוב אקספוננט והתאפסות הגרדיינט עבור ערכים גדולים, אך היתרון שלה הוא שהיא ממורכצת סביב 0.

ReLU (Rectified Linear Unit)

פונקציית ReLU מאפסת ערכים שליליים ואディשה כלפי ערכים חיוביים. הפונקציה מחזירה את המקסימום מבין המספר שהוא מקבלת וביין 0. באופן פורמלי צורת המשוואה הינה:

$$ReLU(z) = \max(0, z), \quad \frac{\partial}{\partial z} ReLU(z) = 1_{\{z>0\}} = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$$



איור 4.7 פונקציית ReLU והנגזרת שלה.

פונקציית ReLU ייעלה יותר לחישוב מהפונקציות הקודמות, כיוון שיש בה רק בדיקה של סימן המספר, ואין בה כפל או אקספוננט. בנוסף, בפונקציה זו הגרדיינט לא מתאפשר בערכים גבוהים. יתרון נוסף שיש לפונקציה זו – היא מתכנסת יותר מהפונקציות הקודמות (αx). פונקציה יש שני חסרונות עיקריים: היא לא ממורצת סביב 0, ועבור אתחול משקלים לא טוב מרבית הנוירונים מתאפסים וזה יחסית בזבזני. כדי להתגבר על הבעיה האחורונה ניתן להשתמש בורותיות של הפונקציה, כמו למשל PReLU ו-ELU:

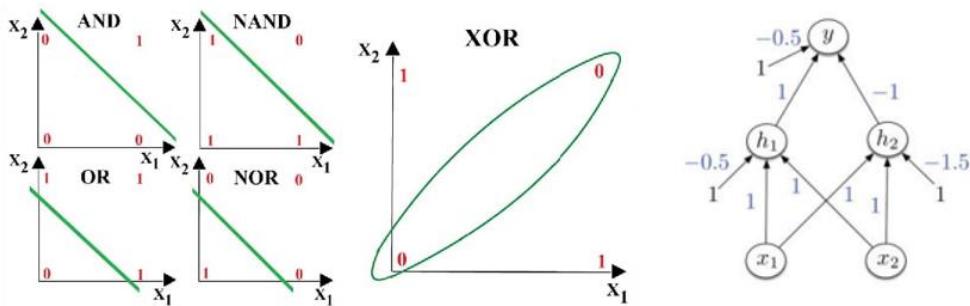
$$PReLU(z) = \max(\alpha z, z), ELU(z) = \begin{cases} z & z > 0 \\ \alpha(e^z - 1) & z \leq 0 \end{cases}$$

בפונקציית PReLU, המקירה הפרטி בו $\alpha = 0.1$ נקרא Leaky ReLU. בפונקציית ELU, הפרמטר α הוא פרמטר נלמד.

ישנן עוד פונקציות, אך אלה הן העיקריות, כאשר לרוב מקובל להשתמש בReLU ובורותיות שלו.

4.1.3 Xor

אחת הדוגמאות הידועות ביותר שאין ניתנות להפרדה לינארית היא בעיית ה-XOR. יש שתי כניסה - x_1, x_2 והמוצא הוא 0 אם הכניסות שוות ו-1 אם הן שונות. פונקציה זו מיפה שתי כניסה ליציאה, כאשר יש שתי קטגוריות במוחא, ואין אפשרות להסביר קו לינארי שיבחין בין הדוגמאות השונות. לעומת זאת, ניתן לבצע שלב מקדים של הפרדה לא לינארית, ולאחריה ניתן יהיה לבנות מסוווג על בסיס קו הפרדה לינארית.



איור 4.8 אופרטור XOR אינו ניתן להפרדה לינארית, בשונה משאר האופרטורים הלוגיים. באמצעות רשת נוירונים בעלי שכבה חבויה אחת ניתן לייצר מודל שקל לאופרטור XOR.

בדוגמא המובאת באיזור הכניסות עוברות דרך שכבה חבויה אחת בעלי שתי נוירונים - h_1, h_2 , המקבלים בנוסף גם bias. פונקציית הפעלה של נוירונים אלו היא פונקציית הסימן, ונitin לכתוב את המוצא של שכבה זו כך:

$$h_1 = sign(x_1 + x_2 - 0.5), h_2 = sign(x_1 + x_2 - 1.5)$$

לאחר השכבה החבויה הנוירונים מחוברים למוצא, שגם לו יש bias, והסכום של הכניסות והbias עוביים במסווג:

$$y = sign(h_1 - h_2 - 0.5) = \begin{cases} 1 & \text{if } h_1 - h_2 - 0.5 > 0 \\ 0 & \text{if } h_1 - h_2 - 0.5 < 0 \end{cases}$$

נבחן את המשמעות של הנוירונים: הנeuron h_1 יהיה 0 אם שתי הכניסות שוות 0, אחרת הוא יהיה שווה 1. הנeuron h_2 יהיה שווה 1 אם שתי הכניסות שוות 1, ובכל מצב אחר הוא יהיה שווה 0. באופן זה לאחר השכבה החבויה הראשונה,

אם גם h_1 שונה מ-0 אז יש לפחות כניסה אחת שווה 1, וצריך לבדוק בעזרת h_2 את המצב של הכניסה השנייה. אם גם הכניסה השנייה שווה 1, אז כניסה של 0 (יחד עם ה-bias) יתקבל מספר שלילי, ובמוצא יתקבל 0. אם הכניסה השנייה היא 0, אז $h_1 = h_2 = sign(0.5) = 0$. במצב בו שתי ה כניסות הן 0, יתקיים $h_1 = h_2 = 0$, ואז רק ה bias ישפייע, כיון שהוא שלילי שוב יתקבל 0 במקרה.

נרשום בפירוט את הערכים בכל שלב, עבור על הנסיבות האפשריות:

x_1	x_2	h_1	h_2	$h_1 - h_2 - 0.5$	y
0	0	0	0	-0.5	0
0	1	1	0	0.5	1
1	0	1	0	0.5	1
1	1	1	1	-1.5	0

4.2 Computational Graphs and propagation

4.2.1 Computational Graphs

כפי שהסביר לעיל, רשת נוירונים عمוקה היא רשת בעלת לפחות שכבה עמוקה אחת, והמטרה של כל שכבה היא ללמידה יצוג פשוט יותר של המידע שנכנס אליה, כך שבסתור של דבר ניתן היה להבחין בין קטגוריות שונות בעזרת הפרדה ביןארית. מה שקובע את השינוי של הדadata במעבר שלו בראשת הם המשקלים והנוירונים המבצעים פעולות לא לינאריות. בעוד הפעולות אותן מבצעים הנוירונים קבועות (סכימה ולאחר מכן פונקציית הפעלה), המשקלים קבועים בהתחלה באופן אקרטי, ובעדת הדוגמאות הידועות ניתן לאמן את הרשות ולשנות את המשקלים כך שיביצעו את למידת הייצוג החדש בצורה אופטימלית.

התליך האימון מתבצע בשני שלבים – ראשית מכנים דוגמא ידועה לתחילת הרשות ו"疎傳" (Forward propagation) (Forward propagation), כלומר, מחשבים את השינוי שהיא עוברת כאשר היא מוכפלת במשקלים וועוברת בנוירונים החבויים. לאחר שמגיעים למקום, משווים את מה שהתקבל למאה שאמור להיות במקומות לפי מה שידוע על דוגמא זו, ועוד מבצעים פעוף לאחר מכן (Backward propagation), שמרתתו לתקן את המשקלים בהתאם למה שהתקבל במקומות. השלב השני הוא למשה חישוב עיל של GD על פני כל שכבות הרשות – מחשבים את הנגזרת בין המשקל w_i לבין פונקציית המחיר (L), ועוד מבצעים עדכון בשיטת $\frac{\partial L}{\partial w_i} \epsilon - w_{i+1}$. כיון שהרשות יכולה להכיל מילוני משקלים, יש למצאו דרך יעילה לחישוב הגרדיאנט עבור כל משקל.

נحو לעשות את התהליך הדו-שלבי הזה בעזרת Computational Graphs, שזהו למעשה גרף הבניי מצמתים המיצגים את התהליך שהדאטה עובר בטור הרשות. הגרף יכול ליצג כל רשות, וכן ניתן באמצעותו לחשב נגזרות מורכבות באופן פשוט יחסית. לאחר השלב הראשון בו מעריכים דוגמא בכל חלק הגרף, ניתן למשל לחשב את השגיאה הריבועית הממוצעת ($y - \hat{y}$), להגדיר אותה כפונקציית המחיר, ולמצוא את הנגזרת של כל משקל לפי פונקציה זו – $\frac{\partial L}{\partial w_i}$, כאשר הנגזרות החלקיות מחושבות בעזרת כלל השרשרת.

4.2.2 Forward and Backward propagation

באופן פורמלי, עבור N משקלים התהליך מנוטה כך:

Forward pass:

For i in 1 ... N:

Compute w_i as function of $w_0 \dots w_{i-1}$

Backward pass:

$$\overline{w_N} = 1$$

For i in N - 1 ... 1:

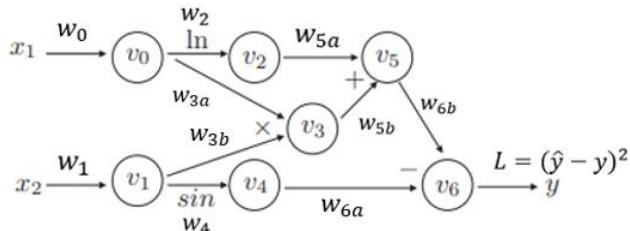
$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial w_N} \cdot \frac{\partial w}{\partial w_{N-1}} \dots \frac{\partial w_{i+1}}{\partial w_i}$$

$$\overline{w_i} = w_i - \epsilon \frac{\partial L}{\partial w_i}$$

בשלב הראשון מחשבים כל צומת על סמך הצמתים הקדמים לו, ובשלב השני בו חוזרים אחורה, מחשבים את הנגזרת של כל משקל בעזרת כל השרשרת החל מהmozo עד לאותו משקל, ומעדכנים את המשקל. נסתכל למשל בדוגמה הבאה:

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

לפונקציה זו שתי כניסה, העוברות כל אחת בנפרד דרך פונקציה לא לינארית, ובנוסף מוכפלות אחת בשנייה. באופן גרפי ניתן לאייר את הפונקציה כך:



איור 4.9 הפונקציה $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$ מתוארת באופן גרפי.

בגרף זה יש 7 צמתים:

$$v_0 = x_1, v_1 = x_2$$

$$v_2 = \ln(v_0), v_3 = v_0 \cdot v_1, v_4 = \sin(v_1)$$

$$v_5 = v_2 + v_3$$

$$\hat{y} = v_6 = v_5 - v_4$$

לאחר שבוצע החישוב עבור \hat{y} , ניתן לחשב את הנגזרות החלקיות, בעזרת כל השרשרת:

$$\frac{\partial L}{\partial w_{6a}} = -1, \frac{\partial L}{\partial w_{6b}} = -1$$

$$\frac{\partial L}{\partial w_{5a}} = \frac{\partial L}{\partial w_{6b}} \frac{\partial w_{6b}}{\partial w_{5a}} = -1 \cdot 1 = 1, \quad \frac{\partial L}{\partial w_{5b}} = \frac{\partial L}{\partial w_{6b}} \frac{\partial w_{6b}}{\partial w_{5b}} = -1 \cdot 1 = -1$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial w_{6a}} \frac{\partial w_{6a}}{\partial w_4} = -1 \cdot (-\cos w_4) = \cos w_4$$

$$\frac{\partial L}{\partial w_{3a}} = \frac{\partial L}{\partial w_{6b}} \frac{\partial w_{6b}}{\partial w_{5b}} \frac{\partial w_{5b}}{\partial w_{3a}} = -1 \cdot 1 \cdot w_{3b}, \quad \frac{\partial L}{\partial w_{3b}} = \frac{\partial L}{\partial w_{6b}} \frac{\partial w_{6b}}{\partial w_{5b}} \frac{\partial w_{5b}}{\partial w_{3b}} = -1 \cdot 1 \cdot w_{3a}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial w_{6b}} \frac{\partial w_{6b}}{\partial w_{5a}} \frac{\partial w_{5a}}{\partial w_2} = -1 \cdot 1 \cdot \frac{1}{\ln w_2}$$

המשקלים בכניסה, $w_0, w_1, w_2, w_3, w_4, w_5, w_6$, רק מעבירים ללא שינוי את הכניסות לצמתים $v_0, v_1, v_2, v_3, v_4, v_5, v_6$, שכן הם שווים 1.

לאחר שכל הנגזרות החלקיות חושבו, ניתן לעדכן את המשקלים לפי העיקרון של GD: $w_{i+1} = w_i + \epsilon \frac{\partial L}{\partial w_i}$

היתרון הגדול של חילוקת הרשת לגרף עם צמתים נובע בכך שאפשר כותבים את הנגזרת של $L(\theta)$ בעזרת כל השרשרת, אז כל איבר בשרשראת בפני עצמו הוא יחסית פשוט לחישוב. למשל – נגזרת של חיבור היא 1, נגזרת של כפל היא המקדם של המשתנה לפיו גוזרים, וכן באותו אופן עבור כל אופרטור שימושיים בצומת מסויים. לשיטה זו קוראים **backpropagation** והוא מודד נפוצה בשיטות עמוקות עוקב ייעולותה בחישוב המשקלים. בשונה מבועית רגристיה, חישוב האופטימום ברשומות עמוקות היא לא בעיה קמורה, ולכן לא תמיד יש לה בהכרח מינימום גלובלי.

עם זאת, עדכון הממשקלים בשיטת backpropagation הוכיח את עצמו, למרות שהמשקלים לא בהכרח הגיעו לאופטימום שלהם.

4.3 Optimization

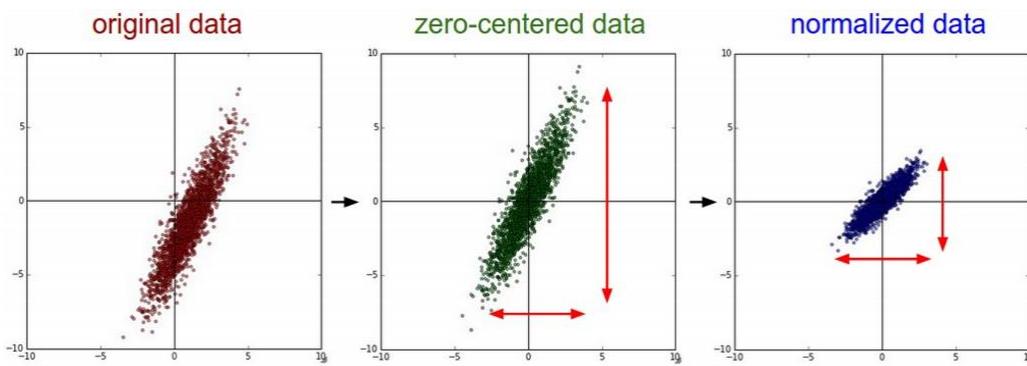
מציאת אופטימום למשקלים על פני כל העומק של הרשת היא בעיה לא קמורה, ולכן אין לה בהכרח מינימום גלובלי. לכן מלבד עדכון הממשקלים בשיטת backpropagation יש לבצע אופטימיזציות נוספת על הרשת על מנת לשפר את הביצועים שלה.

4.3.1 Data Normalization

חלק מפונקציות הפעולה אין ממורכבות סיבוב-0, ועבור ערכים גבוהים הן קבועות בקריב גרדיאנט בערכים אלו מטאפס, דבר שאינו מאפשר לעדכן את המשקלים בשיטת GD. כדי להימנע מהגעה לתחום ה"רויה" בו הגרדיאנט מטאפס, ניתן לנורמל את הדadata כך שהיא בעל תוחלת 0 ושונות 1, ובכך הוא יהיה ממורכב סיבוב-0:

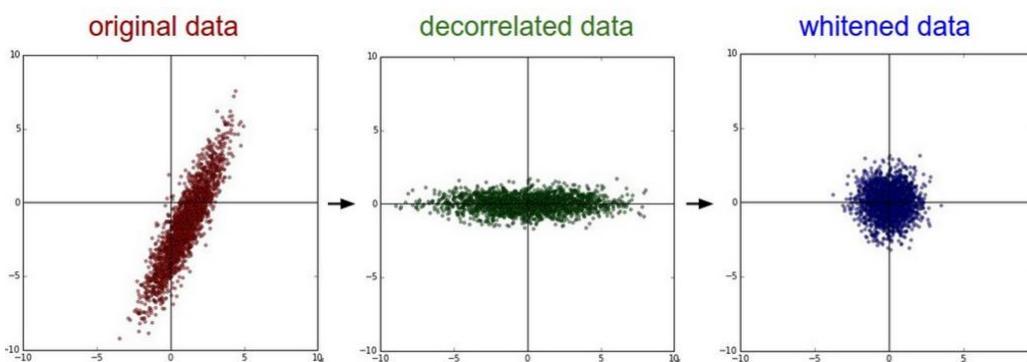
$$X_i = \frac{X_i - \mu_i}{\sigma_i}$$

ובאופן חזותי:



איור 4.10 נרמול דadata בשני שלבים – איפוס התוחלת (ירוק) ונרמול השונות -1 (כחול).

שלב זה הוא למעשה שלב pre-processing הנועד להcin את הדadata לפני כניסה לרשת, כדי לשפר את האימון הרשת. ישנו אופנים נוספים לנרמול את הדadata – ללקסן את מטריצת covariance של הדadata או להפוך אותה למטריצת היחידה:



איור 4.11 דרכים נוספים לנרמל את הדadata – ללקסן את מטריצת covariance (ירוק) או להפוך אותה למטריצת היחידה (כחול).

4.3.2 Weight Initialization

ענין נוסף שיכול להשפיע על האימון ניתן להתייחס אליו עוד בשלב ה-pre-processing הוא אתחול הממשקלים. אם כל המשקלים מאותחלים ב-0, אז המוצא וכל הגרדיאנטים יהיו גם כן 0, ולא יבוצע עדכון למשקלים. לכן יש לבחור את המשקלים ההתחלתיים בצורה מושכלת, למשל, להגריל אותם מהתפלגות מסוימת שתאפשר אימון טוב של הרשת.

אפשרות אחת להגריל היא לאותחול עבור כל משקל ערך קטן מהתפלגות נורמלית עם שונות קטנה – $(\alpha, 0, N)$, כאשר $\alpha = 0.01$ or 0.1 .

בערכיים קטנים גורם לאיפוס הגרדיאנט מהר מדי. כדי להתמודד עם בעיה זו, ניתן לבחור $\alpha = 1$, אך זה יכול לגרום להתקדרות הגרדיאנט. שיטה נוספת יותר נקראת Xavier Initialization, הלוקחת בחשבון את הגודל של השכבות – האתחול יבוצע באמצעות התפלגות נורמלית, אך השונות לא תהיה מספר ללא משמעות, אלא תהיה תלולה במספר השכבות – $\alpha = \frac{1}{\sqrt{n}}$. שיטה זו טובה גם לרשות עם הרבה שכבות, אך היא בעייתית במקרים בו פונקציית הפעלה הינה ReLU, כיוון שהאתחול מניח שפונקציית הפעלה ממורכצת סביב 0 (כמו למשל \tanh). כדי לאפשר גמישות גם מבחינת פונקציית הפעלה, ניתן לבחור $\alpha = \sqrt{\frac{2}{n}}$, ואז האתחול יתאים גם ל-ReLU.

Xavier-Initialization היא להಗיל מהתפלגות איחודית, כאשר באופן דומה ל- $\alpha = \frac{1}{\sqrt{n}}$, גם כאן הגבולות יהיו תלויים בגודל השכבות – $U \left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right]$.

4.3.3 Batch Normalization

כאשר מביצעים normalization, למשה דוגמים לכך שבכינסה לרשת הדטה היה מנורמל סביב ה-0. באופן זה נמנעים מהגעה לUMB ציבויים גבוהים בעומק הרשת, הגורמים להתאפסות או להתקדרות של הגרדיאנט. בפועל, הנרמול הזה לא תמיד מספיק טוב עבור כל השכבות, ואחרי כמה שכבות של הכפלה במשקלים ומעבר בפונקציות הפעלה הרבה פעמים מתקבלים ערכים גבוהים. באופן דומה ל- $\alpha = \frac{1}{\sqrt{n}}$ Data normalization המבוצע לפני האימון, ניתן תוך כדי האימון לבצע Batch normalization שודואג לנרמול הערכים שנכנסים לנירונים בשכבות החבויות. התהליך נעשה בשלושה שלבים:

- א. עברו כל ניירון בעל פונקציית הפעלה לא לינארית, מחשבים את התוחלת והשונות של כל הערכים היוצאים ממנו.
 - ב. מנורמלים את כל היציאות – מחסרים מכל יציאה את התוחלת ומחלקים את התוצאה בשונות (בתוספת אפסיון, כדי להימנע מחילוקה ב-0).
 - ג. הנרמול יכול לגרום לאיובן מידע, לכן מבצעים לתוצאה המנורמלת scale and shift – scale – הזרה ושינוי קנה המידה. תיקון מתבצע בעזרת פרמטרים נלמדים.
- עבור שכבות גדולות חישוב התוחלת והשונות יקר כיוון שלמיירון יש הרבה יציאות, לכן לוקחים רק חלק מהיציאות – $\{x_1 \dots x_m\}$ Mini Batch: \mathcal{B}

באופן פורמלי ניתן לנסח את ה- $\text{BN}_{\gamma, \beta}$ (Mini) Batch Normalizing transform כך:

$$\begin{aligned} \mu_{\mathcal{B}} &= \frac{1}{m} \sum_{i=1}^m x_i, \quad \sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \\ \hat{x}_i &= \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \\ y_i &= \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \end{aligned}$$

כאשר β, γ הם פרמטרים נלמדים (עבור כל ניירון יש פרמטרים שונים).

בשלב המבחן, השונות והתוחלת שבעזרתם מביצעים את הנרמול אין נלקחים מהיציאות של הנירונים, אלא לוקחים ממוצע של כמה מה- Mini Batch האخرين.

יש כמה יתרונות לשימוש ב- BN : האימון נעשה מהר יותר, יש פחות ריגשות לאתחול של המשקלים, מאפשר שימוש ב- learning rate גדול יותר (מוני מגרדיאנט להתקדר או להתאפס), מאפשר שימוש במוגן פונקציות הפעלה (אם אלה שאין ממורכצות סביב 0) ומספק באופן חלקי גם רגולריזציה (שונות נמוכה במקרה).

4.3.4 Mini Batch

במקרים רבים הדטה-ט גודל, ולחשב את הגרדיאנט עבור כל הדטה נדרש הרבה חישוב. בכל צעד של קידום ניתן לחשב את הגרדיאנט עבור חלק מהדטה, ובוצעו את הקידום לפי הכוון של הגרדיאנט המתיקבל. למשל, ניתן לבחור באופן אקראי נקודה אחת ולחשב עליה את הגרדיאנט. בחירה כזו נקראת Stochastic Gradient Descent (SGD).

כיוון שבכל צעד יש בחירה אקראיית של נקודה. בחירה אקראיית של נקודה בודדת יכולה לגרום לשונות גדולות ככל שהחישוב מתקדם, ולכן כל מבצעים mini-batch learning – חישוב הגרדיאנט על חלק מהדטה. באופן זה גם יש הפחתה של כמות החישובים, וגם אין שונות גבוהה. אם מבצעים את החישוב בשיטה זו יש לדאוג שהדטה מעורבב כדי שהמשקלים אכן יתעדכו בצורה נכונה, ובנוסף שה-mini-batch יהיה מספיק גדול כך שהיא בו יציג לכל הדטה. כל מעבר על פני כל הדטה-טט נקרא Epoch (אם הדטה הוא בגודל N, והגודל של כל-mini-batch הוא S, אז כל Epoch הוא S/N איטרציות).

אנו כל צעד הוא קירוב לגראינט, אך החישוב מאד מהיר ביחס לגראינט המדויק, וזה יתרון משמעותי שיש לשיטה זו על פני learning batch. בנוסף, המשקלים שמתקבלים קרובים מאד לאלו שהיו מתקבלים באמצעות 3.8. batch learning

4.3.5 Gradient Descent Optimization Algorithms

בשיטת GD, עדכון המשקלים בכל צעד הוא: $w_{i+1} = w_i - \epsilon \frac{\partial L}{\partial w}$, כאשר ϵ הוא פרמטר שנקרא (lr) Learning Rate והוא קבוע עד כמה יש לשנות המשקל בכיוון הגרדיאנט. בגין לביעות גרגוטה, אופטימיצית רשותנו היה לרוב בעיה שאינה קמורה, שכן לא מובטחת התכנסות למינימום הגלובלי. משום כך, אם בכלל צעד הולכים יותר מדי לכיוון הגרדיאנט השילי, ניתן להתכנס לנקודות אוכף או למינימום לוקאלי שהוא אינו בהכרח המינימום הגלובלי. מצד שני אם מתקדמים מעט מדי לכיוון הגרדיאנט, המשקל יatkesh בקצבו מתעדכן. פרמטר ה- ϵ -זנו נדרש לעילו, שכן צריך שהוא לא יהיה גדול מדי (אחרת תהיה התבדרות של המשקלים או התכנסות למינימום לוקאלי) ולאחר מכן קטן מדי (אחרת לא תהיה התקדמות או שהיא תיה מאוד איטית). כיוון שאין ערך אבסולוטי שמתאים לכל הביעות, יש מגוון שיטות המנסות למצוא את העדכון האופטימלי בכל צעד. יש שיטות משתמשות בפרמטר משתנה – lr, ויש שיטות שימושיות פרמטרים אחרים לביטוי של העדכון.

Momentum

ישנו מצבים בהם יש כל מיני פיתולים בדרך לנקודת מינימום. במקרה זה, בכל צעד הגרדיאנט יפנה לכיוון אחר, וההתכנסות לנקודת מינימום תהיה איטית. הדבר דומה לנחל שזרם לים, אך הוא לא זורם ישר אלא יש לו הרבה פיתולים. כדי להאיץ את ההתכנסות במקורה זה, ניתן לנסות לבחון את הכוון הכללי של הגרדיאנט על סמך כמה צעדים, ולהוסיף התקדמות גם לכיוון זהה. שיטה זו נקראת מומנטום, כיון שהיא מחפש את המומנטום הכללי של הגרדיאנט. החישוב של המומנטום מתבצע בהתאם רקורסיבית:

$$m_{i+1} = \mu m_i - \epsilon \frac{\partial L}{\partial w}$$

ואז העדכון הינו:

$$w_{i+1} = w_i + m_{i+1}$$

הפרמטר μ הינו פרמטר דעיכה עם ערך טיפוסי בתווך [0.9, 0.99]. ניתן להבין את משמעותו על ידי פיתוח של עוד איבר בנוסחת המומנטום:

$$m_{i+1} = \mu m_i - \epsilon \frac{\partial L(w_i)}{\partial w} = \mu^2 m_{i-1} - \mu \epsilon \frac{\partial L(w_{i-1})}{\partial w} - \epsilon \frac{\partial L(w_i)}{\partial w}$$

ניתן לראות שככל שהולכים אחורה בצעדים, כך החזקה של μ גדלה. אם $1 < \mu$, אז עם הזמן הביטוי μ^n יLOUR ויקטן, וכך תהיה פחות השפעה לצעדים שכבר היו לפני הרבה עדכנים. תחת הנחה שהגרדיאנט זהה לכל הפרמטרים, ניתן לפתח נוסחה סגורה לרקורסיבית:

$$m_{i+1} = \mu m_i - \epsilon \frac{\partial L(w)}{\partial w} = \mu^2 m_{i-1} - \mu \epsilon \frac{\partial L(w)}{\partial w} - \epsilon \frac{\partial L(w)}{\partial w} = \dots = -\epsilon \frac{\partial L}{\partial w} (1 + \mu + \mu^2)$$

הביטוי שמתබול הוא סדרה הנדסית מתכנסת, ובסך הכל מתබול הביטוי:

$$w_{i+1} = w_i - \frac{\epsilon}{1 - \mu} \frac{\partial L}{\partial w}$$

היעילות של המומנטום תלויה בבעיה – לעיתים היא מאייצה את ההתכנסות ולפעמים כמעט ואין לה השפעה, אך היא לא יכולה להזיק.

וריאציה של שיטת המומנטום נקראת Nesterov Momentum. בשיטה זו לא מחשבים את הגרדיינט על הצעד הקודם, אלא על המומנטום הקודם:

$$m_{i+1} = \mu m_i - \epsilon \frac{\partial L}{\partial w} (w_i + \mu m_i)$$

$$w_{i+1} = w_i + m_{i+1} = (w_i + \mu m_i) - \epsilon \frac{\partial L}{\partial w} (w_i + \mu m_i)$$

שיטה זו עובדת טוב יותר מאשר בMETHODS, כמובן היא מצליחה להתכנס יותר טוב מאשר המומנטום הרגיל, אך היא איטית יותר.

learning decay

באימון רשותות עמוקות בדרך כלל כדאי להקטין את ה- ϵ -ן עם הזמן. הסיבה לכך היא שיכל שמתקדמים לכיוון המינימום, יש צורך בצעדים יותר קטנים כדי להצליח להתכנס אליו ולא לחוץ מסביבו מצד לצד. עם זאת, קשה לקבוע כיצד לבדוק להקטין את ה- ϵ -ן: הקטנה מהירה שלו תימנע הגעה לאזור של המינימום, והקטנה איטית שלו לא תעזר להתכנס למינימום כאשר מגעים לאזור שלו. ישנו שלושה סוגים נפוצים של שינוי הפרמטר:

א. שינוי הפרמטר בכל כמה Epochs. מספרים טיפוסיים הם הקטנה בחצי כל 5 epochs או חלוקה ב-10 כל 20 epochs. באופן כללי ניתן לומר שאשר גרפ' הלמידה של ה-validation משתפר, יש להקטין את ה- ϵ -ן.

ב. דעיכה אקספוננציאלית של ה- ϵ -ן: $\epsilon_0 e^{-kt}$, כאשר k הם היפר-פרמטרים, ו- t יכול להיות צעד או epoch.

ג. דעיכה לפי $1/t$: $\epsilon = \frac{\epsilon_0}{1+kt}$, כאשר k הם היפר-פרמטרים, ו- t הינו צעד של עדכון.

Adagrad and RMSprop

בעוד השיטה הקודמת מעדכנת את ה- ϵ -ן בצורה קבועה מראש, ניתן לשנות אותו גם באופן מסתגל לפי ההתקדמות בכיוון הגרדיינט. בכל צעד ניתן לבדוק עד כמה גדול היה השינוי בצעדים הקודמים, ובהתאם לכך אפשר לנקוט פעולה מתאימים, מותק מגמה להקטין אותו ככל שמתקדמים לכיוון המינימום. באופן פורמלי, אלגוריתם Adagrad מוגדר כך:

$$w_{i+1} = w_i - \epsilon_i \frac{\partial L}{\partial w}, \epsilon_i = \frac{\epsilon}{\sqrt{\alpha_i + \epsilon_0}}, \alpha_i = \sum_{j=1}^i \left(\frac{\partial L}{\partial w_j} \right)^2$$

כאשר ϵ הוא מספר קטן הנועד למנוע חלקה ב-0. כיוון ש- α הולך וגדל, הביטוי $\frac{\epsilon}{\sqrt{\alpha_i + \epsilon_0}}$ הולך וקטן, וקצב הדעיכה הוא ביחס ישיר לקצב ההתקדמות בכיוון הגרדיינט. בכך מרווחים דעיכה של ה- ϵ -ן, בקצב המשתנה לפי ההתקדמות. באופן ייחודי, הדעיכה של ה- ϵ -ן מהרירה, כיוון שהסכום $\sum_{j=1}^i \left(\frac{\partial L}{\partial w_j} \right)^2$ גדל במהירות. כדי להאט את קצב הדעיכה יש שיטות בהן נתונים יותר משקל לצעדים האחרונים ופחות לצעדים שכבר עברו זמן. השיטה הפופולרית נקראת RMSprop, ובשיטה זו במקומם לסכום את ריבוע הגרדיינט של כל הצעדים הקודמים באופן שווה, מבצעים moving average, וככל שעברו יותר צעדים מצעדים מסוים עד לצעד הנוכחי, כך תהיה לו פחות השפעה על דעיכת ה- ϵ -ן:

$$w_{i+1} = w_i - \epsilon_i \frac{\partial L}{\partial w}, \epsilon_i = \frac{\epsilon}{\sqrt{\alpha_i + \epsilon_0}}, \alpha_i = \beta \alpha_{i-1} + (1 - \beta) \left(\frac{\partial L}{\partial w} \right)^2$$

Adam

ניתן לשלב בין הרעיון של מומנטום לבין adaptive learning rate:

$$\alpha_i = \beta_1 \alpha_{i-1} + (1 - \beta_1) \left(\frac{\partial L}{\partial w} \right)^2, m_i = \beta_2 m_{i-1} + (1 - \beta_2) \frac{\partial L}{\partial w}$$

$$\hat{\alpha}_i = \frac{\alpha_i}{1 - \beta_1^i}, \hat{m}_i = \frac{m_i}{1 - \beta_2^i}$$

$$w_{i+1} = w_i - \frac{\epsilon}{\sqrt{\hat{\alpha}_i + \epsilon_0}} \hat{m}_i$$

מספרים טיפוסיים: 5^{-4} or 5^{-2} , $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-2}$. האלגוריתם למעשה גם מוסיף התקדמות בכיוון המומנטום (הכיוון הכללי של הגרדיאנט), וגם מביא לדעיכה אדפטיבית של ה- ϵ -ו. זה האלגוריתם ה- ϵ -פופולרי ברטשות عمוקות, אך הוא לא מושלם ויש לו שתי בעיות עיקריות: האימון הראשון לא יציב, כיון שבתחלת האימון יש מעט נקודות לחישוב הממוצע עבור i . בנוסף, המודל המתkeletal נוטה ל-overfitting עם מומנטום.

יש הרבה וריאציות חדשות על בסיס Adam שמדוודר להתגבר על בעיות אלו. ניתן למשל להתחילה לאמן בקצב נורא, וכאשר המודל מתגבר על בעיית ההתייצבות הראשונית, להגבר את הקצב (ups-warm). במקביל, ניתן להתחילה עם Adam ולהחליף ל-SGD כאשר קритריונים מסוימים מתקיימים. כך ניתן לנצל את התכונות המהירה של Adam בתחלת האימון, ואת יכולת ההכללה של SGD.

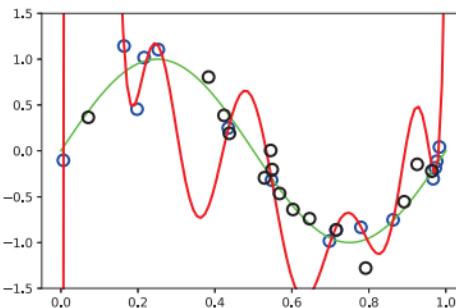
4.4 Generalization

כל מודל שנבנה נסمر על דатаה קיימ, מתוך מגמה שהמודל יתאים גם לדאטה חדש. לכן יש חשיבותגדולה שהמודל ידע להכליל כמה שיותר טוב, על מנת שהוא יתאים בצורה טובה לא רק לדאטה הקיימ אלא גם לדאטה חדש. במקרים אחרים, יש לוודא שהמודל לא מצליח להכליל גם לדוגמאות שהוא רואה, אלא שינסה להבין מטר הדוגמאות מה החקיקות הכללית, שמתאיימה גם לדוגמאות אחרות.

4.4.1 Regularization

כפי שהסביר בפרק 3.1.3, מודל יכול לסבול מהטיה לשני ציוונים – Overfitting ו-underfitting. בהתחלת הערכת יתר לככל נקודה בסט האימון, מה שגורר מודל מסדר גבוה בעל שונות גדולות. במצב זה המודל מתאים רק לסט האימון, אך הוא לא מצליח להכליל גם נקודות חדשות. Underfitting הוא מצב ההפוך – מודל שלא מצליח למצאו קן מגמה המכיל מספיק מידע על הדוגמאות הנתונות, ויש לו רושץ.

ברשת נוירונים, ככל שמספר הפרמטרים גדול, כך השגיאה של ה-test training קטנה. לגבי ה-test training קטנה. עד נקודה מסוימת, ושם היא גדלה בחזרה. בהתאם השגיאה יודעת כיוון שמציאות לבנות מודל יותר מדויק ונמנעים מ-underfitting, אך בנקודה מסוימת יש יותר מדי פרמטרים והם נהנים מواتאמים יותר מידי לסט האימון. overfitting. למעשה צריך למצוא את היחס הנכון בין מספר הפרמטרים (סדר המודל) לבין גודל הדאטה. כיוון שאין אפשרות לזהות Overfitting באמצעות ה-h-Loss בלבד, שחרי ה-h-Loss קtan ככל שיש יותר פרמטרים, ניתן לחלק את הדאטה לשני חלקים – training and validation. בשלב ראשון בונים מודל בהינתן ה-training, ולאחר מכן בוחנים את המודל על ה-validation – אם המודל לא מתאים ל-validation – אז מודל לא מותאם. overfitting סימן שיש מודל בהינתן ה-training ולאחר מתאים רק לדוגמאות שהוא רואה והוא נתן להם הערכת יתר. ככל שהגרף של ה-validation accuracy קרוב יותר ל-training accuracy, כך יש פחות overfitting.



איור 4.12 בדיקת overfitting set validation. הנקודות הכחולות שייכות ל-training והשחורות שייכות ל-validation. המודל האדום מתאים רק לנקודות הכהולות, אך אפשר לומר שהוא נוטה ל-overfitting. המודל הירוק לעומת זאת מותאם גם לנקודות השחורות, אותן הוא לא ראה בשלב האימון, ככל שהוא הצליח להכליל טוב גם לדוגמאות חדשות.

האפשרות ה- ϵ פשוטה להימנע מ-overfitting היא פשוט להוריד פרמטרים, כלומר להקטין את גודל הרשת. בנוסף ניתן לבצע בכל Epoch את גרפ Epoch של ה-h-Loss של ה-validation, וכאשר הוא מתחילה לעלות להפסיק את האימון. שיטות אלה פשוטות מאוד לשימוש, אך ישן שיטות מסוימות ביצועים יותר טובים, ובוחן אותן כעת.

4.4.2 Weight Decay

בדומה לרגולריזציה של linear regression, גם ברשת נוירונים ניתן להוסיף איבר ריבועי לפונקציית המבחן, מה שמכונה L2 Regularization:

$$Cost(w; x, y) = L(w; x, y) + \frac{\lambda}{2} \|w\|^2$$

ההוספה של הביטוי האחרון דואגת לכך שהמשקל לא יהיה גדול מדי, שהרי רצים למזער את פונקציית המבחן, שכן לאף לכך שהביטוי הריבועי יהיה כמוה שיותר קטן. בתוספת האיבר ערךן של המשקלים יהיה:

$$w_{i+1} = w_i - \epsilon \left(\frac{\partial L}{\partial w} + \lambda w - 1 \right)$$

הביטוי הזה דומה מאוד ל-GD רגיל, כאשר נוסף איבר של λ . אם $\lambda < 0$, אז ללא קשר לגראדיאנט המשקל יורד בכל צעדים, וזה נקרא "Weight decay".

ניתן לבצע רגולריזציה עם איבר לא ריבועי, מה שמכונה L1 Regularization:

$$Cost(w; x, y) = L(w; x, y) + \lambda \sum_i |w_i|$$

ואז הערךן יהיה:

$$w_{i+1} = w_i - \epsilon \left(\frac{\partial L}{\partial w} + \lambda \cdot sign(w) \right)$$

בעוד L2 regularization מושך יחיד ונiska להקטין אותו, ניתן לבנות מושך חדש בודקים אותו על כל המodelים ולוקחים את המושצע. סט המodelים נקרא ensemble. ניתן לבנות מodelים שונים במספר דרכים:

4.4.3 Model Ensembles and Drop Out

עבור דאטה קיימן ניתן לבנות מספר מודלים, ואז כשבאים לבחון דאטה חדש בודקים אותו על כל המodelים ולוקחים את המושצע. סט המodelים נקרא ensemble. ניתן לבנות מodelים שונים במספר דרכים:

א. לאמן רשות עם אתחולים שונים למשקלים.

ב. לאמת מספר רשותות על חלקים שונים של הדאטה.

ג. לאמן רשות במספר ארכיטקטורות.

יצירת ensemble בדרכים אלה יכולה לעזור בהכללה, אך יקר ליצור את ה-ensemble ולפעמים קשה לשלב בין מodelים שונים.

יש דרך נוספת ליצורensemble – באמצעות Dropout. קלומר למחוק באופן אקראי נוירון אחד או יותר. אם יש רשות מסוימת ומוחקים את אחד הנוירונים – למעשה מקבלים רשות אחרת, ובפועל אפשר לקבלensemble בעזרת רשות אחת בלבד פעמיים מוחקים ממנו נוירון אחד או יותר. היתרון של יצירת ensemble בדרך זו הוא שההרשעות חולקות את אותן פרמטרים ולבסוף מתקבלים מקרים שונים כל הנוירונים והמשקלים. בפועל עבור כל דוגימה מגරילים רשות (מווחקם כל נוירון בהסתברות $0.5 = k$) וכך לומדים במקביל הרבה רשותות שונות עם אותן פרמטרים. באופן זה כל נוירון מוכrho להיות יותר משמעוני בלי' אפשרות להסתמך על נוירונים אחרים שיעשו את הלמידה, כיון שלא תמיד הם קיימים. אמנם כל ריצה יחידה יכולה להיות בעלת שונות גבוהה אך המושצע של המשקלים מביא לשונות נמוכה.

בשלב המבחן, לא מפעילים את dropout אלא לוקחים את כל הנוירונים, כאשר מחלקים את כל המשקלים בחצי. הסיבה לכך היא שניתן להניח שבשלב האימון חצי מהפעמים המשקל היה 0 כיון שהנוירון הקשור אליו נמחק, ובחצי מהפעמים היה משקל שנלמד. ניתן גם לקחת הסתברות אחרת למחיקת נוירונים, למשל $0.25 = k$, ואז כמשמעותם את כל הרשותות השונות יש לחלק בהסתברות המתאימה. החיסרונו של שיטה זו הוא שלווח לה זמן לה恬נו.

4.4.4 Data Augmentation

שיטה אחרת להימנע מ-overfitting היא להגדיל את סט האימון, וכך המודל שנוצר יתאים ליותר דוגמאות. ניתן לעשות זאת על ידי ייצור וריאציות של הדוגמאות המקוריות. שיטה זו נקראת **Data Augmentation**, והרעיון הוא לבצע עיבوت קטן לכל דוגמא כך שהיא עדין תשמור על המשמעות המקורית שלה, אך תהיה מספקת שונה מהמקורה כדי להיות דוגמא נוספת משמעותית בסט האימון. בדומין של תמונה האוגמנטציות הנפוצות הן:

- סיבוב תמונה בزاوية מסוימת (rotate), הנבחרת מהתפלגות אחידה מהתחום $[0, 2\pi]$.
- הוספת רעש לכל פיקסל, כאשר הרעש משתנה מפיקסל לפיקסל, והוא קטן מ-1%.
- שינוי הגודל (rescaling) של התמונה בפקטור מסוים – בדרך כלל הפקטור שייר לתוחם $\left[\frac{1}{1.6}, 1.6\right]$.
- שיקוף התמונה (flip).
- מתיחה ומריצה של התמונה (shearing and stretching).

References

MLP:

מצגות מהקורס של פרופ' יעקב גולדברגר

<https://joshuagoings.com/2020/05/05/neural-network/>

Xor:

<https://www.semanticscholar.org/paper/Simulations-of-threshold-logic-unit-problems-using-Chowdhury-Ayman/ecd5cb65f0ef50e855098fa6e244c2b6ce02fd48>

5. Convolutional Neural Networks (CNNs)

הרשאות שתוארו עד כה הין (FC), Fully-Connected, כל נוירון מחובר לכל הנוירונים בשכבה שלפניו וכל הנוירונים בשכבה לאחריו. גישה זו יקרה מבינה חישובית, ופעמים רבות אין צורך בכל הקשרים בין הנוירונים. לדוגמה, תמונה בגווני אפור (grayscale) בעלת $1 \times 256 \times 256$ פיקסלים, המזונת לרשת FC עם $N = 1000$. קטגוריות במציאות, מכילה יותר מ-65 מיליון קשרים בין נוירונים, כאשר כל קשר הינו משקל המתעדכן במהלך הלמידה. אם יש מספר שכבות רב במספר נהיה עצום ממש, ולכן מושך הרבה הפעמים והרמיטרים גדלה, באופן צזה שבתי מעשי לתחזק את הרשת. מלבד בעיות הגדל, בפועל לא תמיד יש צורך בכל הקשרים, וכך גם במקרה איננו ראוי הכניסה. למשל, עבור תמונה המזונת לרשת, שימושות רבותקשר בין פיקסלים רוחקים בתמונה איננו שימושותי, שכן אין חשיבות לחבר את הכניסה לכל הנוירונים בשכבה הראשונה ולאחר מכן כל שתי שכבות סמוכות באופן מלא. כדי להימנע מבעיות אלו, לרוב יהיה כדאי להשתמש בשכבות קונבולוציה, שאינן מושכות בין שני נוירונים, אלא רק בין איברים קרובים, כפי שיפורט. רשתות מודרניות רבות מבוססות על שכבות קונבולוציה, כאשר על גבי המבנה הבסיסי נבנו ארכיטקטורות מתקדמות.

5.1 Convolutional Layers

5.1.1 From Fully-Connected Layers to Convolutions

האלמנט הבסיסי ביותר ברשתות קונבולוציה הינו שכבת קונבולוציה לינארית על פני דאטה בכדי לקבל ייצוג אחר ופשטוט יותר שלו. לרוב, שכבת קונבולוציה מבצעת פעולה קרוס-קורולציה בין וקטור המשקלים לבין K מסויים (וקטור הכניסה או וקטור היוצא משכבה חבויה). וקטור המשקלים נקרא גרעין הקונבולוציה (convolution kernel) או מסנן (filter), ובעצרתו מבצעת פעולה הקרוס-קורולציה הבאה:

$$y[n] = \sum_{m=1}^{K-1} x[n-m]w[m]$$

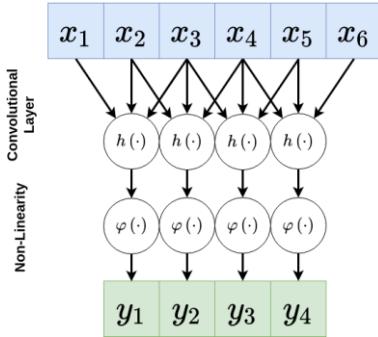
כאשר $x \in \mathbb{R}^n$ הוא וקטור הכניסה ואילו $w \in \mathbb{R}^K$ הוא וקטור המשקלים אשר נלמדים במהלך האימון. וקטור המשקלים w זהה לכל הכניסות בשכבה וכן מספר הפרמטרים הנלמדים לעומת שכבת FC הינו קטן בהרבה – שכבת FC מכילה $N_{inputs} \times N_{outputs}$ משקלים ואילו שכבת קונבולוציה מכילה K משקלים בלבד (לרוב מתקיים $K \ll N_{inputs} \times N_{outputs}$).

מלבד הקטנות כמות המשקלים, השימוש בגרעין קונבולוציה מסיע לזרחי דפוסים ולמציאות מאפיינים. יכולות אלו נובעות מאפיי פעולה הקונבולוציה, הבודקת חיפויה בין חלקים מוקטור הכניסה לבין גרעין הקונבולוציה. הקונבולוציה יכולה למצוא מאפיינים בסיגナル, ושניהם גרעיני קונבולוציה שיכולים לבצע אוסף פעולות שימושיות, כמו למשל החלקה, נגזרת ועוד. אם מתייחסים על תמונה הרבה גרעינים שונים, ניתן למצוא בה כל מיני מאפיינים – למשל אם הגרעין הוא בצורה של עין או אף, אז הוא מסוגל למצוא את האזוריים בתמונה המקוריים הדומים לעין או אף.



איור 5.1 (a) קונבולוציה חד ממדית בין שתי פונקציות: x_1 הינו מלבן בגובה 1 עם רעש קטן (כחול), $-x_2$ הינו גרעין קונבולוציה מלבני שרע על פני כל הישר (כתום). פעולה הקונבולוציה (שחזור) בודקת את החיפויה בין הסיגナル לבין הגרעין, וניתן לראות שנקסיב $= x$ $\pm 0.5 \pm 0.1$ יש אזכור עם הרביה חיפויה. (b) קונבולוציה דו ממדית למציאת קווי מתאימים בתוך תמונה.

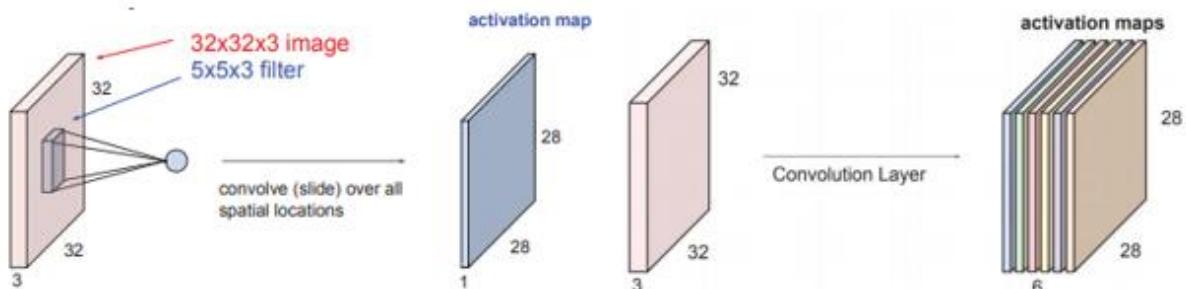
המוצא של שכבת הקונבולוציה עובר בפונקציית הפעלה לא לינארית (בדרך כלל \tanh או $ReLU$), והוא מכונה מפת הפעלה או מפת מאפיינים (feature map). הקונבולוציה יחד עם האקטיבציה נראה כ:



איור 5.2 דואטה x עובר דרך שכבה קונבולוציה ולאחריה פונקציית הפעלה, ובמוצא מתקבלת מפת אקטיבציה y .

לרוב בכל שכבה קונבולוציה יהיה כמה מסכנים, אשר כל אחד מהם אמור ללמידה מאפיין אחר בתמונה. ככל שהרשות הולמת ועמוקה, כך המאפיינים בתמונה אמורים להיות מובחנים באופן יותר אחד מאשר השני, וכן המסכנים בשכבות העמוקות אמרורים להבדיל בין דברים מסוימים יותר. למשל, פעמים רבות ניתן לבדוק אם המסכנים בשכבות הראשונות יזהו את שפות האלמנטים שבתמונה או בצורות אבסטרקטיות, ואילו מסכנים בשכבות העמוקות יותר יזהו אלמנטים מסוימים יותר כמו איברים או חפצים שלמים בעלי צורה ומוגדרת.

הקלט של שכבה הקונבולוציה יכול להיות רב ערכיו (למשל, תמונה צבעונית המייצגת לרוב בעזרת ערכי RGB). במקרה זה הקונבולוציה יכולה לבצע פעולה על כל הערכים יחד ולספק פלט חד ערכיו והוא יכול לבצע פעולה על כל ערך בנפרד ובכך לספק פלט רב ערכיו. גרעין הקונבולוציה יכול להיות חד ממדי, כolumn וקטור שפועל על קלט מסוים, אך הוא יכול להיות גם מממד גבוה יותר. לרוב, מסכנים הפעילים על תמונות הינם דו ממדיים, ופעולה הקונבולוציה מבוצעת בכל שלב כפל בין המסלן לבין איזור דו ממד אחר בתמונה.



איור 5.3 מסנן $\in \mathbb{R}^{5 \times 5 \times 3}$ פועל על קלט $\in \mathbb{R}^{32 \times 32 \times 3}$ ומטבלת מפת אקטיבציה $\in \mathbb{R}^{28 \times 28}$ y (שמאל). הקלט יכול לעבור דרך מסנן ולייצר מפת אקטיבציה עם מספר שכבות – עברו שיש מהסן המודד של המפה הינו $\in \mathbb{R}^{28 \times 28 \times 6}$ y (ימין).

5.1.2 Padding, Stride and Dilatation

כמו ברשת FC, גם ברשת קונבולוציה יש היפר-פרמטרים הנקבעים מראש וקובעים את אופן פעולות הרשת. ישנו שני פרמטרים של שכבה הקונבולוציה – גודל המסלן ומספר ערוצי הקלט וכן שלושה פרמטרים עיקריים נוספים הקובעים את אופן פעולות הקונבולוציה:

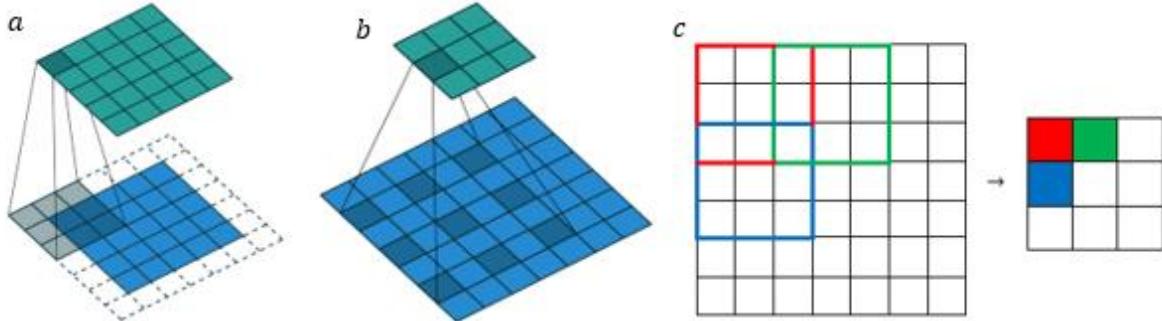
ריפורד (Padding): פעולות הקונבולוציה המוגדרת באמצעות המסלן הינה פעולה מרחבית, כלומר, המסלן פועל על מספר איברים בכל פעולה. בנוסף, נשים לב כי פעולה הקונבולוציה לא מוגדרת על איברי הקצוות لكن לא נוכל להפעיל את המסלן במקומות אלו. באירור 5.2 ניתן לראות את כיצד פעולה על תמונה בממד של 32×32 מוגדרת על 28×28 , דבר הנבע מכך שהקונבולוציה לא מוגדרת על הפיקסלם בקצוות התמונה ולכן לא מופעלת עליהם. אם רוצים לבצע את הקונבולוציה גם על הקצוות, ניתן לרדוף את שולי הקלט (באפסים או שכפול של ערכי הקצה). עבור מסנן בגודל $K \times K$, גודל הריפורד הנדרש הינו: $\text{Padding} = \frac{K-1}{2}$.

תרחבות (Dilation): על מנת לצמצם עוד במספר החישובים, אפשר לפעול על אזוריים יותר גדולים מתוך הנחה שערכאים קרובים גיאוגרפית הם בעלי ערך זהה. לשם כך ניתן להרחיב את פעולה הקונבולוציה תוך השמטה של ערכים קרובים. התרחבות טיפוסית הינה בעלת פרמטר $d = d$.

גודל צעד (Stride): ניתן להניח שלרוב הקשר המרחבי נשמר באזוריים קרובים, שכן על מנת לקטין בחישוביות ניתן לדלג על הפלט ולהפעיל את פעולה הקונבולוציה באופן יותר דليل. לעומת זאת, אין צורך להטיל את המסלן על כל האזוריים

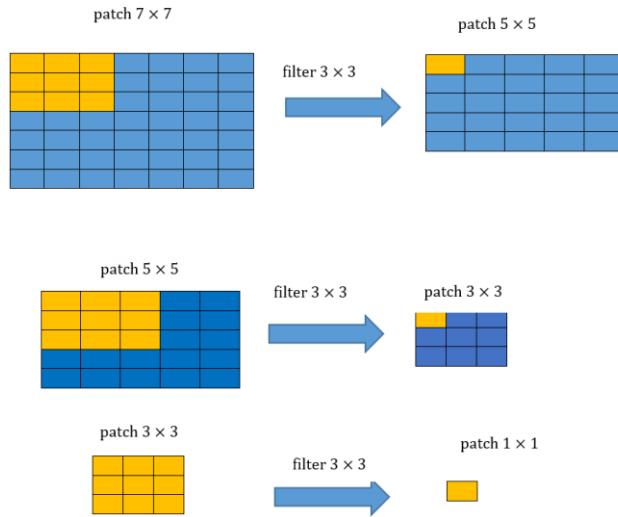
האפשרים ברשות, אלא ניתן לבצע דילוגים, כך שלאחר כל חישוב קונבולוציה יבוצע דילוג בגודל הצעד לפני הקונבולוציה הבאה. גודל צעד טיפוסי הינו $s = 2$.

גודל שכבת הפלט לאחר ביצוע הקונבולוציה תלוי בגודלים של הכניסה והמשן, בריפוד באפסים ובגודל הצעד. באופן פורמלי ניתן לחשב את גודל שכבת הפלט לפי הנוסחה: $\frac{W-K+2P}{s} + 1 = 0$, כאשר W הוא גודל הכניסה, K הוא גודל המשן, P זה הריפוד באפסים ו- s זה גודל הצעד. מספר שכבות הפלט הינו כמספר המשנים (כasher שכבת פלט יכולה להיות רב ערכית). יש לשימושם לב שערבי ההיפר-פרמטרים (padding, dilation and stride) וכן גודל הגרעין נדרשם להיות מספרים טבעיים אשר מקיימים את נוסחת גודל שכבת הפלט (0) הנ"ל, כך שגם 0 הינו מספר טבעי.



איור 5.4 (a) ריפוד באפסים על מנת ביצוע קונבולוציה גם על הקצוות של הדואטה. (b) התוצאות ($2 = d$): ביצוע הקונבולוציה וטור השמטת איברים סמוכים מתוך הנחה שכנהה הם דומים. (c) הצעת המשן בצעד של $2 = s$.

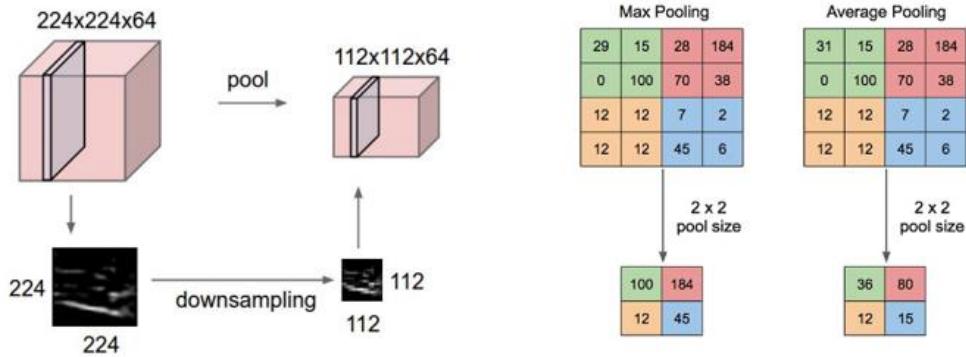
תמרק (Receptive field) של איבר ברשת מוגדר להיות כל התחומים בכניסה אשר משפיע על אותו איבר לאורך השכבות.



איור 5.5 של ערך מסוים בManagedObject של שלוש שכבות קונבולוציה רצופות עם מסנן בגודל 3×3 .

5.1.3 Pooling

פעמים רבות דאטה מרחבית מאופיין בכך שאיברים קרובים דומים אחד לשני, למשל – פיקסלים סמוכים לרוב יהיו בעלי אותו ערך. ניתן לנצל עבודה זו בצדדי להויריד את מספר החישובים הדרושים בעזרת דילוגים (Strides) או הרחבת (dilation) כדי שתואר לעיל. שיטה אחרת לניצול עבודה זו היא לבצע Pooling – אחרי כל ביצוע קונבולוציה, דגימת ערך ייחיד מהאזור בעל ערכים מרובים, המציג את האזור. את צורת חישוב הערך של תוצאה ה-*pooling* ניתן לבחור בכמה דרכים, כאשר המקבילות הן בחירת האיבר הגדול ביותר באיזור שלו (max pooling) או את הממוצע של האיברים (average pooling).



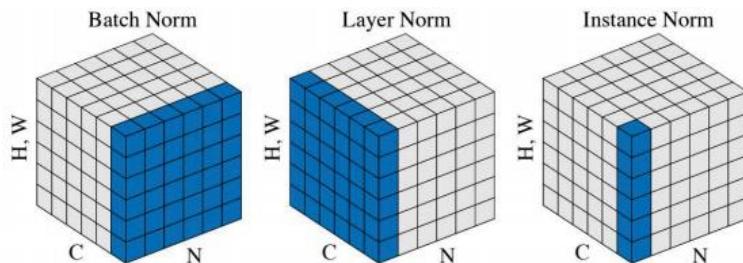
איור 5.6 הקטנת הממד של הדadata בעזרת Pooling (שמאל), והמחשה מספרית של ביצוע max/average pooling בגדול של 2×2 .

5.1.4 Training

ככל, תהליכי האימון של רשת קומבולוציה זהה לאימון של רשת FC, כאשר ההבדל היחיד הוא בארכיטקטורה של הרשת. יש לשים לבש המנסנים מופעלים על הרבה אזורים שונים, כאשר המשקלים של המנסנים בכל צעד שוויים, וכן אותם משקלים פועלם על אזורים שונים. לשם הפשטות נניח ויש מסנן יחיד, ככלומר מטריצה אחת נלמדת של משקלים. מטריצה זו מוכפלת בכל אחד מהאזורים השונים של הדadata, וכדי לבצע עדכון למשקלים שלא ישקלל את הגרדיינטס של כל האזורים השונים. בפועל, הגרדיינט בכל צעד יהיה הסכום של הגרדיינטס על פני כל הדadata, ועבור המקרה הכללי בו יש N אזורים שונים עליהם מופעל המסנן הגרדיינט יהיה:

$$\frac{\partial L}{\partial w_k} = \sum_{i=1}^N \frac{\partial L}{\partial w_k(i)}$$

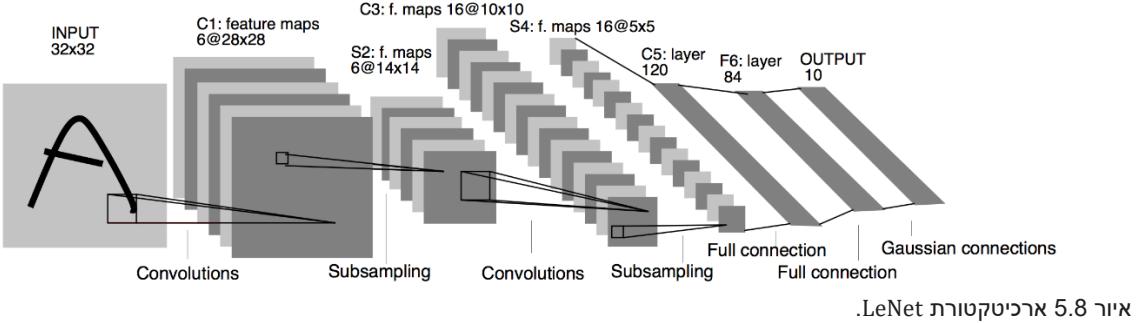
בדומה ל-FC, גם ב-CNN ניתן לבצע normalization של הנרמול על סט של וקטורים מסוימים (לשם הנוחות נתיחס לווקטורים של הדadata כתמונות). אפשרות פשוטה ונפוצה היא לנרמל כל מסנן בפני עצמו על פני כמה תמונות (Batch Norm), ככלומר לקחת את כל הפיקסלים בסט של תמונות ולנרמל בתוחלת ובשונות שלהם. אפשרות נוספת היא לקחת חלק מהמידע של סט תמונות, אך לנרמל אותו ביחס לאותו מידע על פני מסננים אחרים (Layer Norm). יש וריאציות של הנרמולים האלה, כמו למשל Normalization, Instance Norm, הלוקח מסנן אחד ותמונה אחת ומנרמל את הפיקסלים של אותה תמונה.



איור 5.7 נרמול שכבות של רשת קומבולוציה.

5.1.5 Convolutional Neural Networks (LeNet)

בעזרת שרשור של שכבות וחיבור כל האלמנטים השיכים לקומבולוציה ניתן לבנות רשת שלמה עבו מגוון ממשימות שונות. לרוב במאזא שכבות הקומבולוציה יש שכבה אחת או מספר שכבות FC. מטרת ה-FC היא לאפשר חיבור של המידע המוכל במאזאים שנאספו במהלך שכבות הקומבולוציה. ניתן להסתמך על הרשת הכוללת כשי שלבים – בשלב הראשון מבצעים קומבולוציה עם מסננים שונים, שכל אחד מהם נועד לזרות מאפיין אחר, ובשלב השני מוחברים חוזרת את כל המידע שנאנסף על ידי חיבור כל הננוירונים באמצעות FC. לראשונה השתמשו בארכיטקטורה זו בשנת 1998, בראשת הנקרatte LeNet (על שם Yann LeCun), ומוצגת באיור 5.8. רשת זו השיגה דיוק של 98.9% בזיהוי ספורות, כאשר המבנה שלה הוא שתי שכבות של קומבולוציה ושלוש שכבות FC, כאשר לאחר כל אחת משכבות הקומבולוציה מבצעים pooling.



איור 5.8 ארכיטקטורת LeNet.

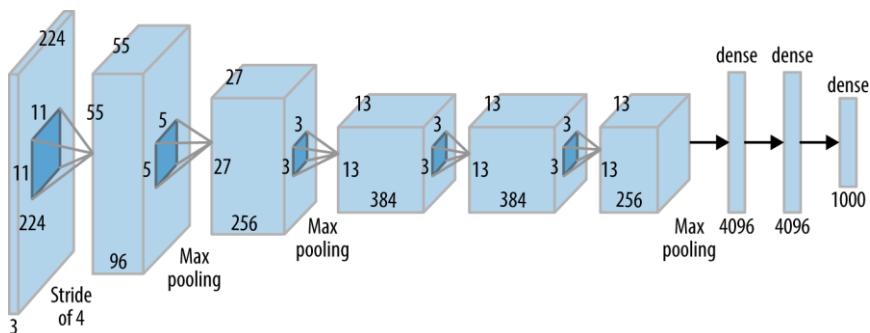
5.2 CNN Architectures

בשנים שלאחר LeNet העיסוק ברשתות נירוניים عمוקות דיברנו, עקב חוסר המשאבים לבצע חישובים רבים בעיליות ובמהירות. בשנת 2012 רשות בשם AlexNet המבוססת על שכבות קונבולוציה ניצחה בתחרות ImageNet (תחרות ליזיינו תמונות), כאשר היא הציגה שיפור משמעותית מהותוצה הци טובה בשנה שלפני. יחד עם התפתחות יכולות החישוב, העיסוק ברשתות عمוקות חזר להיות מרכזי ופותחו הרבה מאוד ארכיטקטורות מתקדמות.

5.2.1 AlexNet

רשת AlexNet שאבה את ההשראה למבנה שלה מארכיטקטורת LeNet, כאשר היכולת שלה להתמודד עם משימות יותר מורכבות מאשר LeNet נובעת מכך שנהיי דאטא-סיטים גדולים מאוד שנitin לאمن עליהם את הרשות, ובנוסף כבר היה קיים GPU שבעדרטו ניתן לבצע חישובים מורכבים. הארכיטקטורה של הקונבולוציה מתבצע pooling ו- convolution ושלוש שכבות FC, כאשר לאחר שתי השכבות הראשונות של הקונבולוציה מתקבל pooling ו- normalization. ה-input הוא מממד $3 \times 224 \times 224$, ומופעלים עליו 96 מסננים בגודל 11×11 , עם גודל צעד 4 max-pooling. ולאחר מכן הוא מממד $55 \times 55 \times 96$. לאחר מכן מתקבל max-pooling עם גודל צעד 4. ולאחר מכן יש 256 מסננים בגודל 5×5 עם גודל צעד 1 = s וריפורד באפסים = k, שכן הממד הוא השני יש $27 \times 27 \times 256$ max-pooling שמחזית את שני הממדים הראשונים, ומתקבלת שכבה בממד $27 \times 27 \times 2$. בשכבת הקונבולוציה השנייה יש 256 מסננים בממד 3×3 , עם גודל צעד 1 = s וריפורד באפסים = k, ואז שכבת הקונבולוציה השנייה עם 256 מסננים בממד 3×3 , עם 1 = k = s. במושג הקונבולוציות יש עוד max-pooling, ואז שלוש שכבות FC, כאשר המוצא של השכבה الأخيرة הוא וקטור באורך 1000, המציג 1000 קטגוריות שונות שיש בDATA-SET. ImageNet

פונקציית האקטיבציה של הרשת הינה ReLU (בשונה מ- LeNet שהשתמשה ב- \tanh), וההיפר פרמטרים הם: $\text{learning rate} = 1e-2$, $\text{SGD+momentum} = 0.9$, $\text{batch size} = 128$, $\text{Dropout} = 0.5$, $\text{屢次} = 60$ מיליון.

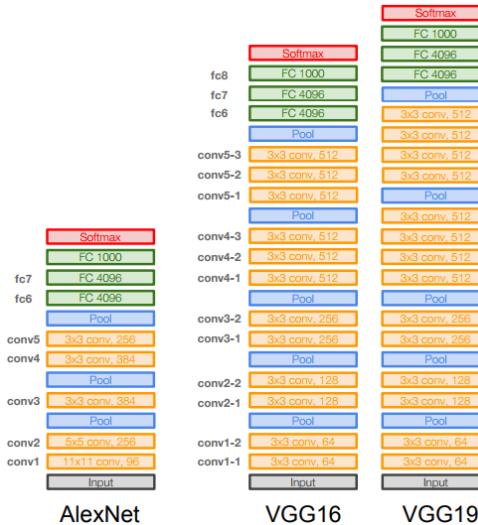


איור 5.9 ארכיטקטורת AlexNet.

שנה לאחר הפרסום של רשת AlexNet, פותחה רשת דומה בשם ZFNet, הבנוהה באותה ארכיטקטורה עם הבדלים קטנים בהיפר-פרמטרים ובמספר הפילטרים: השכבה הראשונה של הקונבולוציה הפכה מ: $4 \times 11 \times 11$ ל: $5 \times 7 \times 7$, ובשכבות 3-4-5 מס' הפילטרים הוא 512, 1024, 512, 1024, 512. הרשת השיגה שיפור של כ-5% על פוי AlexNet. הממד של השכבות בשתי הארכיטקטורות אינםינו נובע מסיבה מסוימת אלא מניסוי וטעיה – במסו תוצאות רבות ומגוונות נבחרה זו בעלת הביצועים הטובים ביותר. לאחר שהרשתות מבוססות קונבולוציה הוכחו את כוחן, השלב הבא היה לבנות רשתות עמוקות, ובועלות ארכיטקטורה הנשענת לא רק על ניסויים אלא גם על היגיון מסוים.

5.2.2 VGG

שנה לאחר ZFNet הוצגה בתקנות רשת עמוקה – בעלת 19 שכבות, המנצלת יותר טוב את שכבות הקונבולוציה. מפתח הרשת הראו כי ניתן להחליף שכבת פילטרים של 7×7 בשולש שכבות של 3×3 ולקבל את אותו תمر (receptive field), כאשר מרוחאים חסכו משמעותי במספר הfilטרים הנלמדים. לפילטר בגודל d הפעול על c ערוצי קלט ופלט יש $c^2 d^2$ פילטרים נלמדים, لكن לפילטר של 7×7 יש $49c^2$ פילטרים נלמדים ואילו לשולש שכבות של 3×3 יש $c^2 \cdot 3^2 = 27c^2$ פילטרים נלמדים – חיסכון של 45%. הרשת המקורית שפיתחו נקראה VGG16 והיא מכילה 138 מיליון פילטרים, ויש לה וריאציה המוסיפה עוד שתי שכבות קונבולוציה ומוכנה VGG19.



איור 5.10 ארכיטקטורת VGG (ימין) ביחס לארכיטקטורת AlexNet (שמאל).

5.2.3 GoogleNet

המודלים הקודמים היו יקרים חישובית עקב מספר הfilטרים הגדל. כדי להציגו להגעה לאותם ביצועים עם אותו עומק אבל עם הרבה פחות פילטרים, בוצעת מפתחים מוגול הציגו קונספט שנקרא inception module. בлок המבצע הרבה פעולות פשוטות במקביל, במקומות לבצע פעולה אחת מורכבת. כל בлок מקבל input ומבצע עליו ארבעה חישובים במקביל, כאשר המגדים של מוצאי כל הענפים שוויםacr שnitן לשרשר אותם יחד. ארבעת הענפים הם: קונבולוציה 1×1 , קונבולוציה 1×1 ולאחריה קונבולוציה 3×3 עם padding 1, קונבולוציה 1×1 ולאחריה קונבולוציה 5×5 עם padding 2, ולאחריה קונבולוציה 3×3 עם max pooling 3, ולאחריה קונבולוציה 1×1 . לבסוף, הפלטים של ארבעת הענפים משוררים יחד ומהווים את פלט הבלוק.

המבנה זה שקול למספר רשותות במקביל, כאשר היתרון של המבנה זה הוא כפול: כמה פילטרים נמוכה ביחס לרשותות קודמות וחישובים יחסית מהירים כיוון שהם נעשים במקביל. ניתן לחבר שכבות קונבולוציה רגילים עם בלוקים כאלה, ולקבל רשת עמוקה. נעשו הרבה ניסויים כדי למצוא את היחס הנכון בין הרכיבים והמלדים בכל שכבה המביאים לביצועים אופטימליים.



איור 5.11 Inception Block 5.11 (ימין), וארכיטקטורת GoogleNet מלאה (שמאל).

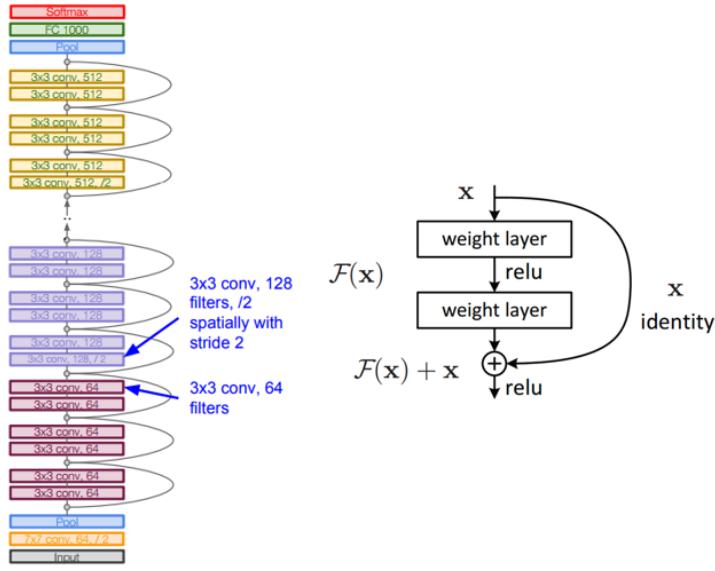
5.2.4 Residual Networks (ResNet)

לאחר שראו שככל שהרשת عمוקה יותר כך היא משיגה תוצאות טובות יותר, ניסו לבנות רשתות עם מאות שכבות, אך הן השיגו תוצאות פחות טובות מהרשתות הקודומות שהיו בעלות סדר גודל של 20 שכבות. הבעיה המרכזית של הרשתות העמוקות נבעה מכך שללאור מספר שכבות מסוים התקבל ייצוג מספק טוב, ולכן השכבות היו צרכיות לא לשנות את הקטל אלא להניבר את היצוג כמו שהוא. בשביל לבצע זאת המשקלים בשכבות אלו צריכים להיות 1. הסתבר לשכבות קשה ללמידה את פונקציית הזהות והן מעשנה פגעו בתוצאה. אתגר נוסף ברשתות עמוקות נבע מהקשה לבצע אופטימיזציה כמו שצריך למשקלים בשכבות עמוקות.

ניתן לנוכח את הבעיה המרכזית באופן מעט שונה – בהינתן רשת עם N שכבות, יש טעם להוסיף שכבה נוספת רק אם היא תוסיף מידע שלא קיים עד עכשווי. כדי להבטיח ששכבת התוסיף מיידע, או לכל הפחות לא תפגע במידע קיימים, בנו רשת חדשה בעזרת Residual Blocks – יצירת בלוקים של שכבות קובולוציה, כאשר בנווסף למעבר של המידע בתוך הבלוק, מחברים גם בין הכניטה למצאה שלו. כעת אם בלוק מבצע פונקציה מסוימת $(x)\mathcal{F}$, אז הימצא הינו $x + (x)\mathcal{F}$. באופן זהה כל בלוק ממוקד בלמידה משווה שונה ממה שנלמד עד עכשווי, ואם אין מה להוסיף – הפונקציה $(x)\mathcal{F}$ פשוט נשארת 0. בנוסף, המבנה של הבלוקים מונע מהגדידיאנט בשכבות העמוקות להתבדר או להתאפס, והאימון מצליח להתכנס.

באופן זהה פותחה רשת בעלת 152 שכבות אשר הציגה ביצועים מעולים ביחס לכל שאר הרשתות באותה תקופה. השכבות היו מורכבות משלשות של בלוקים, כאשר בכל בלוק יש שתי שכבות קובולוציה. בין כל שלשה יש הכפלה Batch normalization והורדה של הממד פי שניים בעזרת pooling. ההיפר-פרמטרים הם: lr=0.1 ,SGD+momentum=0.9 ,Xavier initialization בשיטת siontion .weight decay=1e-5 ו-batch size=256 ומוחולק ב-10 בכל פעם שה-validation error מת>'ישר,

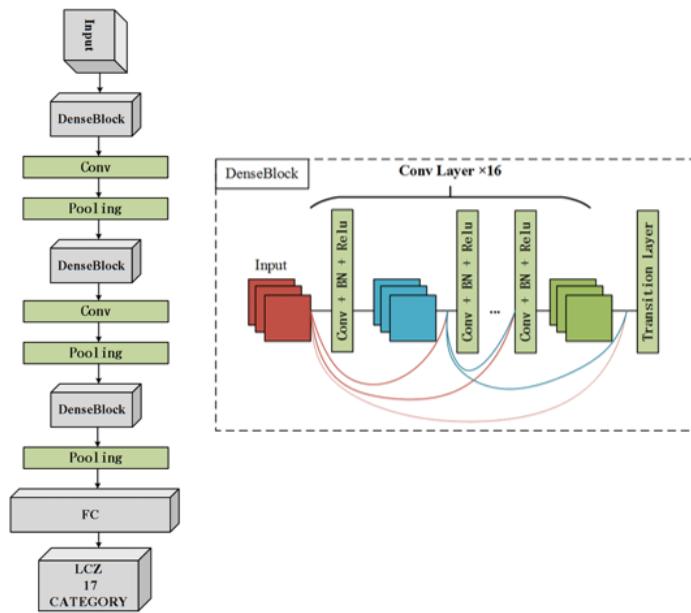
רשתות מתקדמות יותר שילבו את גישת ה-inception יחד עם ResNet על מנת לשלב בין היתרונות של שתי השיטות.



איור 12 Residual Block 5.12 יחיד (ימין), וארQUITקטורת ResNet מלאה (שמאל).

5.2.5 Densely Connected Networks (DenseNet)

ניתן להרחיב את הרעיון של Residual Block כך שלא רק מ לחברים את הכניסה של כל בלוק למוצא שלו, אלא גם שומרים את הכניסה בפניה עצמה, ובודקים את היחס שלה לשכבות יותר עמוקות. Dense block הוא בלוק בעל כמה שכבות, הבניי כך שכניסה של כל שכבה מחוברת לכל הENSIONS של השכבות אחרות. ניתן כמובן לשרשור כמה בלוקים כאלה יחד ולבצע ביניהם כל מיני פעולות כמו pooling או אפילו שכבת קונבולוציה עצמאית. כיוון שהשכבות כמה כניסה של בלוקים שונים, יש בעיה של התאמת ממדים, משום שכל בלוק מגדיל את מספר העורצים, חיבור של כמה בלוקים יכולם ליצור מודל מורכב מדי. כדי להתגבר על בעיה זו הוספו שכבות transition בסוף כל dense block המבצעות קונבולוציות 1×1 עם רוחב צעד $= 2$, וכך מספר העורצים נותר סביר ומהודל לא נעשה מורכב מדי.

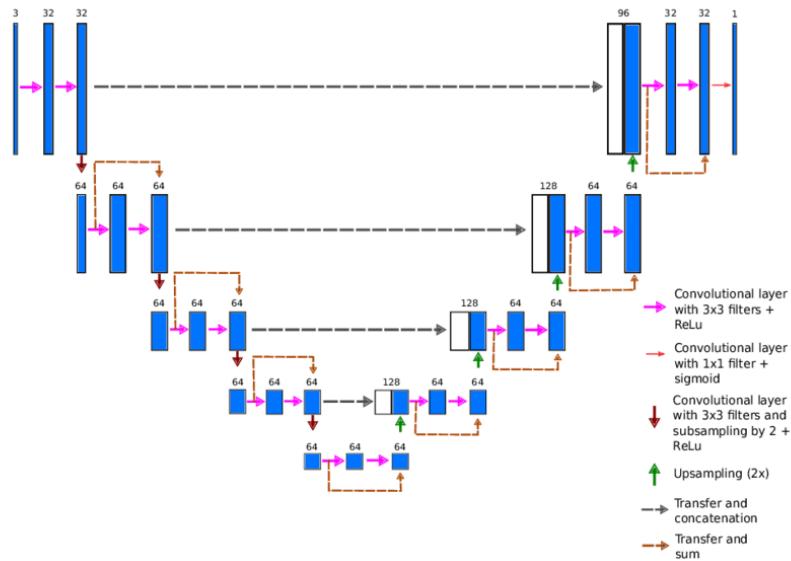


איור 13 Dense Block 5.13 יחיד (ימין), וארQUITקטורת DenseNet מלאה (שמאל).

5.2.6 U-Net

ברשותות קונבולוציה המיפויות לסיווג, בסוף התהילה מתקובל וקטור של הסטברויות, כאשר כל איבר הוא הסטברות של label מסוים. במשימת סגמנטציה זה בעייתי, כיוון שצריך בסוף התהילה לא רק ללמידה את המאפיינים שבתמונה ועל פיהם לקבוע מה יש בתמונה, אלא צריך גם לשחזר את מיקומי הפיקסלים והתיוגים שלהם ביחס לתמונה המקורית עם הסגמנטציה המתאימה. כדי להתמודד עם בעיה זו הציעו את ארכיטקטורת U-Net, המכילה שלושה חלקים עיקריים: כיווץ, צוואר בקבוק והרחבה (contraction, bottleneck, and expansion section).

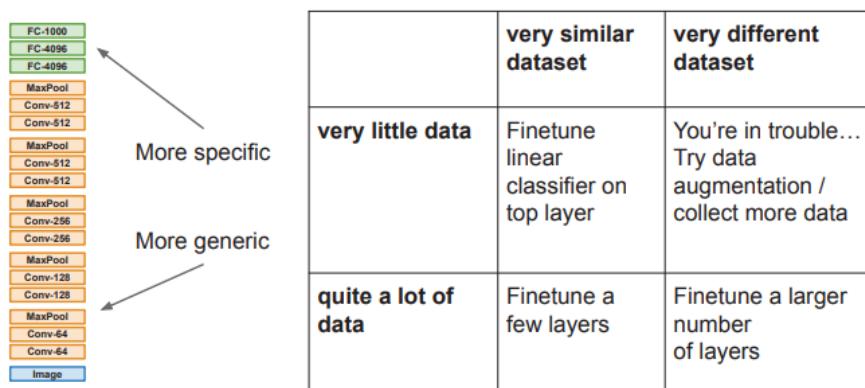
באיור, בחלק הראשון יש טופולוגיה רגילה של רשת קונבולוציה, המבוצעת בעזרת שכבות קונבולוציה וביצוע pooling. השני בין השלב הזה לבין רשת קונבולוציה קלאסית הוא החיבור שיש בין כל שלב בתהילר לבין חלקים בהמשך התהילר. לאחר המעבר בצדואר הבקבוק יש למשה שחזור של התמונה עם הסגמנטציה. השחזור נעשה בעזרת sampling על הווקטור שהתקבל במצוא צדוර הבקבוק יחד עם המידע שנשאר מהחלק הראשון של התהילר. פונקציית המבחן שמשתמשים ברשת זו נקראת **pixel-wise cross entropy loss**, הבודקת כל פיקסל ביחס ל-label האמתי אליו הוא שייך.



איור 5.14 ארכיטקטורת Net-U.

5.2.7 Transfer Learning

כאשר נתקלים במשימה חדשה, אפשר לתקן עבורה ארכיטקטורה מסוימת ולאמן רשת עמוקה. בפועל זה יקר ומסובך להתאים רשת מיוחדת לכל בעיה ולאמן אותה מהתחלתה, ולכן ניתן להשתמש ברשתות הקיימות שאומנו כבר ולהתאים אותן לביעות אחרות. גישה זו נקראת **Transfer Learning**, וההגיאון מאחוריה טוון שעבור שימוש כל סוג דאטה השכבות הראשונות למודדות אותו דבר (ziehi שיפות, קווים וצורות כלליות, מאפיינים כללים וכו') וכן ניתן להשתמש בהן פעמים רבות ללא שינוי כלל. משום כך, בפועל בדרך כלל לוקחים רשת קיימת ומחליפים בה את השכבות האחרונות או מוסיפים לה עוד שכבות בסופה, ואז מאמנים את השכבות החדשות על הדאטה החדש כך שהן תהיה מוכנות לדאטה הספציפי של המשימה החדשה. ככל שיש יותר דאטה חדש ניתן להוסיף יותר שכבות ולקיים דיקון יותר טוב, וככל שהמשימה החדשה דומה יותר למשימה המקורית של הרשת כך יש צורך בפחות שכבות חדשות. כמו כן, משום שבשיטה זו נדרש שכבות קטן יותר, קטן ה-**overfitting** הנובע ממוחוסר בדאטה.



איור 5.15 Transfer Learning 5.15

5. References

Convolutional:

<https://github.com/technion046195/technion046195>

מצגות מהקורס של פרופ' יעקב גולדברגר

AlexNet:

<https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecc96>

VGG

<https://arxiv.org/abs/1409.1556>

GoogleNet

http://d2l.ai/chapter_convolutional-modern/vgg.html

ResNet

<https://arxiv.org/abs/1512.03385>

DenseNet

<https://arxiv.org/abs/1608.06993>

<https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803>

U-Net

<https://arxiv.org/abs/1505.04597>

6. Recurrent Neural Networks

היתרון של שכבות קונבולוציה על פניהם FC הוא ניצול הקשר המרחבי שיש בין איברים שונים בדאטא, כמו למשל פיקסלים בתמונה. יש סוג מסוימן נתונים יוצרים סדרה שיש לסדר האיברים חשיבות, כמו למשל טקסט, גלי קול, רצף DNA ועוד. כמובן שדאטא מהסוג זהה דורש מודל הנוטן חשיבות לסדר של האיברים, מה שלא קיים ברשותות קונבולוציה. בנוסף, הרבה פעמים הממד של הקלט לא ידוע או משתנה, כמו למשל מספר המילים במשפט, וגם לך יש לתת את הדעת. כדי להתמודד עם אטגרים אלו יש לבנות ארכיטקטורה שמקבלת סדרה של וקטורים וממציאות וקטור יחיד, כאשר הווקטור היחיד מקודד קשרים על הדאטא המקורי שנכנס אליו. את וקטור המוצא ניתן להעביר בשכבה FC או בכל מסוג אחר, תלוי באופן המשימה.

6.1 Sequence Models

6.1.1 Vanilla Recurrent Neural Networks

רשתות רקורסיביות הן הכללה של רשתות נוירונים עמוקות, כאשר הן מכילות משקלות המאפשרות להן לחת משמעות לסדר של איברי הכניסה. ניתן להסתכל על משקלות אלה כרכיב זיכרון פיני, כאשר כל איבר שוכנו משקל בלבד ביחס לפונקציה קבועה בתוספת רכיב משתנה שתלו בעבר ההפוך. כאשר נכנס וקטור x , הוא מוכפל במשקל w_{xh} ונכנס לרכיב זיכרון h_t , אשר h_t הוא פונקציה של h_{t-1} :

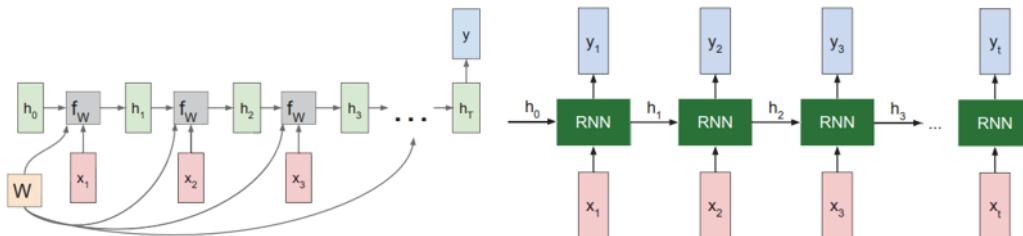
$$h_t = f(h_{t-1}, x_t)$$

מלבד המשקלים הפעילים על וקטור הכניסה, יש גם משקלים שפועלים על המשקלות הפנימיות (רכיב הזיכרון) – w_{hh} , ומשקלים הפעילים על המוצא של רכיב זה – w_{hy} . המשקלים w_{hx}, w_{hh}, w_{hy} זהים לכל השלבים, והם מתעדכנים ביחיד. כמו כן, הפונקציה f היא קבועה לכל האיברים, למשל \tanh , ReLU או sigmoid . באופן פרטני התהיליך נראה כך:

$$h_t = f_w(w_{hh}h_{t-1} + w_{xh}x_t), f_w = \tanh/\text{ReLU}/\text{sigmoid}$$

$$y_t = w_{hy}h_t$$

באופן סכמתי התהיליך נראה כך:



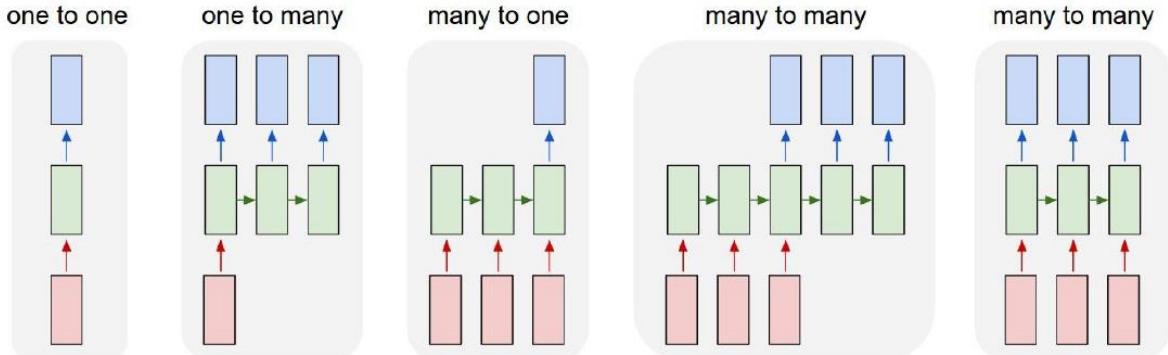
איור 6.1 ארכיטקטורות RNN בסיסיות: Many to One (מימין) – One to many (משמאל) (Many to Many). על כל חץ יש משקל מתאים עליו מתבצע הלמידה.

כמובן שניתן גם לשרשן שכבות ח比亚ות ולקבל רשת עמוקה, כאשר פלט של שכבה מסוימת הופך להיות הקלט של השכבה הבאה. ישנם מודלים שונים של RNN, המתאים לביעיות שונות:

One to many – יש קלט יחיד ורוצים להוציא סדרת פלטים, למשל מכנים תמונה לרשת ורוצים משפט שיתאר אותה (Image captioning).

Many to one – רוצים לקבוע משהו ייחיד עבור קלט סדרתי, למשל מקבלים משפט ורוצים לסוג את הסנטימנט שלו – האם הוא חיובי או שלילי.

Many to many – עבור כל סדרת קלט יש סדרת פלט, למשל תרגום משפה אחת לשפה אחרת – מקבלים משפט ומוציאים משפט.



איור 6.2 מודלים שונים של RNN.

6.1.2 Learning Parameters

$x = (x_1, \dots, x_n), (y_1, \dots, y_n)$. עבור דוגמה לרשומות שבפרקם הקודמים. עבור דוגמה (\hat{y}_i, y_i) נגידר את פונקציית המחר:

$$L(\theta) = \frac{1}{n} \sum_i L(\hat{y}_i, y_i, \theta)$$

כאשר הפונקציה $L(\hat{y}_i, y_i, \theta)$ תותאם למינימיזציה – עבור משימות סיווג cross entropy ועבור בעיות רגראטיות המשמש בקורסירטוריון MSE. האימון יבוצע בעזרת GD, אך לא ניתן להשתמש ב-backpropagation כיוון שכל משקל מופיע מספר פעמים – למשל w_{hx} פועל על כל הכניסות ו- w_{hh} פועל על כל רכיבי הזיכרון. כדי לבצע את עדכון המשקלים משתמשים ב-(BPTT) backpropagation through time (BPTT) – מסתכלים על הרשות הנפרשת קרשת אחת גדולת, מחשבים את הגרדיאנט עבור כל משקל, ואז סוכמים או ממצאים את כל הגרדיאנטים. אם הדואטה בכניסה הוא בגודל a , כלומר יש a דוגמאות בזמן, אז יש a רכיבי זיכרון, ו- $a - 1$ – a משקלים w_{hh} . لكن הגרדיאנט המשוקל יהיה:

$$\frac{\partial L}{\partial w_{hh}} = \sum_{n=1}^a \frac{\partial L}{\partial w_{hh}(t)} \quad \text{or} \quad \frac{\partial L}{\partial w_{hh}} = \frac{1}{a} \sum_{n=1}^a \frac{\partial L}{\partial w_{hh}(t)}$$

כיוון שהמשקלים זהים לאורך כל הרשות, $w_{hh} = (t)$ והשני בזמן יהיה רק לאחר ביצוע ה-BPTT יהיה רלוונטי רק לוקטור הבא.

הצורה הפשוטה של ה-BPTT יוצרת בעיה עם הגרדיאנט. נניח שרכיב הזיכרון מיוצג באמצעות הפונקציה הבאה:

$$h_t = f(z_t) = f(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$$

לפי כלל השרשרת:

$$\frac{\partial h_n}{\partial x_1} = \frac{\partial h_n}{\partial h_{n-1}} \times \frac{\partial h_{n-1}}{\partial h_{n-2}} \times \dots \times \frac{\partial h_2}{\partial h_1} \times \frac{\partial h_1}{\partial x_1}$$

כיוון ש- w_{hh} קבוע ביחס בזמן עבור קטור כניסה יחיד, מתקבל:

$$\frac{\partial h_t}{\partial h_{t-1}} = f'(z_t) \cdot w_{hh}$$

אם נציב זאת בכלל השרשרת, נקבל שעבור חישוב הנגזרת $\frac{\partial h_n}{\partial x_1}$ מכפילים 1 – a פעמים ב- w_{hh} . לכן אם מתקיים $1 > \|w_{hh}\|$ אז הגרדיאנט יתבדר, ואם $1 < \|w_{hh}\|$ הגרדיאנט יתאפס. לעומת זאת, של התבדורות או התאפסות הגרדיאנט, יכולה להופיע גם ברשומות אחרות, אבל בגלל המבנה של RNN והLINEARITY של ה-BPTT ברשומות רקורסיביות זה קורה כמעט תמיד.

עבור הבעיה של התבדורות הגרדיאנט ניתן לבצע clipping – אם הגרדיאנט גדול מקבוע מסוים, מנורמלים אותו:

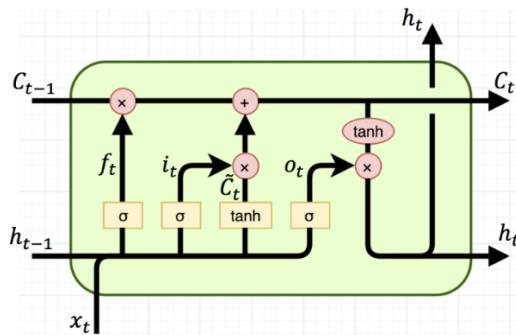
$$\text{if } \|g\| > c, \text{ then } g = \frac{cg}{\|g\|}$$

הבעיה של התאפסות הגראדיינט אמונה לא גורמת לחישובים של מספרים עצומים, אך היא בעצם מבטלת את ההשפעה של איברים שנמצאים רחוק אחד מהשני. אם למשל יש משפט ארוך, אז במרקבה בו הגראדיינט דוער במהלך ה-TT-BPTT, כמעט ואין השפעה של המילה הראשונה על המילה الأخيرة. במילים אחרות – התאפסות הגראדיינט גוררת בעיה של Long-term, ככלומר קשה ללמידה דاطה בעל תלות בטוויה ארוך, כמו משפט ארוך או תופעות שימושיות לאט. בגלל הבעיה זו לא משתמשים ב-RNN הקלסי (שנקרא גם Vanilla RNN), אלא מבצעים עליון שיפורים, כפי שיווסף בפרק הבא.

6.2 RNN Architectures

6.2.1 Long Short-Term Memory (LSTM)

כדי להתגבר על בעיית דעיכת הגראדיינט המונעת מהרשת לשימוש בזכרון ארוך טווח, ניתן להוסיף אלמנטים לריכב הזכרון כך שהוא יוכל רק מידע על העבר, אלא יהיה גם בעל שליטה על איך וכמה לשמש בميدע. ב-RNN הפשט לריכב הזכרון יש שתי כניסה – x_t ו- h_{t-1} , ובעזרתן מחשבים את המוצא על ידי שימוש בפונקציה $(x_t, h_{t-1}) \cdot f_w(h_t)$. למעשה ריכב הזכרון הוא קבוע והלמידה מתבצעת רק במשקלים. ב-LSTM יש שני שינויים עיקריים – מלבד הכניסות הרגולריות יש עוד כניסה הנקראת memory cell state ומסומנת ב- c_{t-1} , ובנוסף לכך h_t מחושב בצורה מורכבת יותר. באופן זה האלמנט c_t דואג לזכרון ארוך טווח של דברים, ו- h_t אחראי על הזיכרון של הטוויה הקצר. נתבונן בארכיטקטורה של תא זיכרון:



איור 6.3 תא זיכרון בארכיטקטורת LSTM.

הצמד $[x_t, h_{t-1}]$ נכנס לתא ומוכפל במשקל w , ולאחר מכן עובר בנפרד דרך ארבעה שערים (יש לשים לב שלא מבצעים פעולה בין x_t לבין h_{t-1} אלא הם נשאים בנפרד ואת כל הפעולות עושים על כל איבר בנפרד). **שער הראשון** מביא פועלתו בין x_t לבין h_{t-1} והוא שער שכחה והוא אחראי על מחיקת חלק מהזיכרון. אם למשל יש משפט ומופיע בו נושא חדש, אז שער זה אמור למחוק את הנושא שהוא שומר בזיכרון. **שער השני** הוא שער זיכרון והוא אחראי על כמה יש לזכור את המידע החדש לטוויה ארוך. אם לדוגמה אכן יש במשפט מוסרים נושא חדש, אז השער יחליט שיש לזכור את המידע הזה. אם לעומת זאת המידע החדש הוא תיאור שלRALONTO להמשך אז אין טעם לזכור אותו. **שער השלישי** הוא שער זיכרון אחד והוא אחראי על כמה מהמידע RALONTO לדאטה הנוכחי x_t , כלומר מהי הפלט של התא בהינתן מידע העבר. שלושת השערים הללו נקראים מסכות (Masks), והם מקבלים ערכים בין 0 ל-1 כיוון שהם עוביים דרך סיגמודoid. **שער רביעי** c_t (לפעמים מסומן ב- \tilde{c}_t) שאחריו על השאלה כמה מהזיכרון להعبر לата הבא. שער זה משקל את המידע המתkeletal יחד עם c_{t-1} שאומר עד כמה יש לזכור להמשך את המידע החדש.

באופן זהה מקבלים הוא את h_t שאחריו על הזיכרון לטוויה קצר $Vanilla RNN$, והן את c_t שאחריו על זיכרון של כל העבר. ארכיטקטורת הריכב מאפשרת להתייחס לאלמנטים נוספים במספרים הקשורים לזכרון – ניתן לשכוח חלקים לא רלוונטיים של התא הקודם (f_t), להתייחס באופן סלקטיבי ל- c_t ולהוציא רק חלק מהמידע המשווקל הקיים (o_t). באופן פורמלי ניתן לנסח את פעולות התא כך:

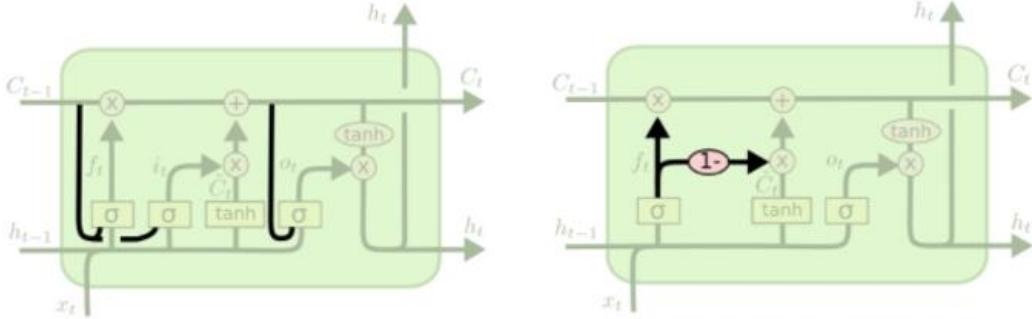
$$\begin{pmatrix} i \\ f \\ o \\ \tilde{c} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} = \begin{pmatrix} \sigma(w_i \cdot [x_t, h_{t-1}] + b_i) \\ \sigma(w_f \cdot [x_t, h_{t-1}] + b_f) \\ \sigma(w_o \cdot [x_t, h_{t-1}] + b_o) \\ \tanh(w_{\tilde{c}} \cdot [x_t, h_{t-1}] + b_{\tilde{c}}) \end{pmatrix}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, h_t = o_t \odot \tanh(c_t)$$

כאשר האופרטור Θ מסמל כפל איבר (כיוון שלשערים נכנס הזוג $[x_t, h_{t-1}]$, אם במצב מוצאים מכפלה מסוימת, יש לבצע אותה על כל אחד מהאיברים).

יש וריאציות שונות של רכבי LSTM – ניתן למשל לחבר את c_{t-1} לא רק למצאו h_t אלא גם לשאר השערים. חיבור כזה נקרא peephole, כיוון שהוא מאפשר לתשעים להתובן ב- c_{t-1} באופן ישיר. יש ארכיטקטורות שמחברות את c_{t-1} לכל השערים, ויש ארכיטקטורות שמחברות אותו רק לחלק המשערם. חיבור כל השערם $\Theta c_{t-1} \cdot w$ משנה מבנה את משוואות השערם. במקום $(b + b)$, המשווה החדש תהיה $(b + w) \cdot [c_{t-1}, x_t, h_{t-1}] \cdot w$.

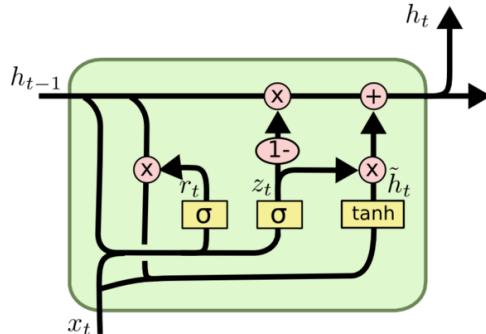
וריאציה אחרת של LSTM מתחדשת בין שער השכחה f_t לבין שער הזיכרון i_t , והחלה עד כמה יש למחוק מידע מהזיכרון מתקבלת יחד עם החלטה כמה מידע חדש יש לכתוב. שינוי זה ישפייע על memory cell, כאשר במקום $c_t = f_t \Theta c_{t-1} + (1 - f_t) \Theta \tilde{c}_t$, משווהות העדכון תהיה: $c_t = f_t \Theta c_{t-1} + i_t \Theta \tilde{c}_t$.



איור 6.4 וריאציות של LSTM – LSTM – (ימין) ו- i -gates (שמאל).

6.2.2 Gated Recurrent Units (GRU)

ישנה ארכיטקטורה נוספת של תא זיכרון הנקראת (GRU), והיא כוללת מספר שינויים ביחס ל-LSTM:



איור 6.5 תא זיכרון בארכיטקטורת GRU.

השינוי המשמעותי מ-LSTM הוא העבודה שאינו memory cell state, וכל השערם מתבססים רק על הקלט והמצוא של התא הקודם. כדי לאפשר זיכרון הן לטווח קצר והן לטווח ארוך, יש שני שערם – Update gate ו-Reset gate – והם מחושבים על פי הנוסחאות הבאות:

$$\text{Update: } z_t = \sigma(w_z \cdot [x_t, h_{t-1}])$$

$$\text{Reset: } r_t = \sigma(w_r \cdot [x_t, h_{t-1}])$$

בעזרת שער reset מחשבים Candidate hidden state

$$\tilde{h}_t = \tanh(w \cdot [x_t, r_t \Theta h_{t-1}])$$

ראשית יש לשים לב Ci $\in [0, 1]$ כי Ci כיוון שהוא תוצאה של סיגמואיד.Cut געט נתובן על \tilde{h}_t ביחס לרכיב זיכרון פשוט של Vanilla RNN: $r_t = f_W(w_{hh}h_{t-1} + w_{xh}x_t)$

$$\tilde{h}_t = \tanh(w \cdot [x_t, r_t \Theta h_{t-1}]) \approx \tanh(w[x_t, h_{t-1}]) = \tanh(w_{hx}x_t + w_{hh}h_{t-1})$$

המשמעות היא שעבור $1 \rightarrow r_t$ מתקובל רכיב הזיכרון הקלאסי, השומר על זיכרון לטוח קצר. אם לעומת זאת $0 \rightarrow r_t$, אז מתקובל $\tanh(w_{xh}x_t + b_{xh}) = \tanh(w_{xh}x_t)$, ולמעשה הזיכרון של הטווח הקצר ממתפס (reset).

לאחר החישוב של \tilde{h}_t מחשבים את המוצא של המצב החבוי בעזרת z_t , שגם הוא מקבל ערכים בין 0 ל-1:

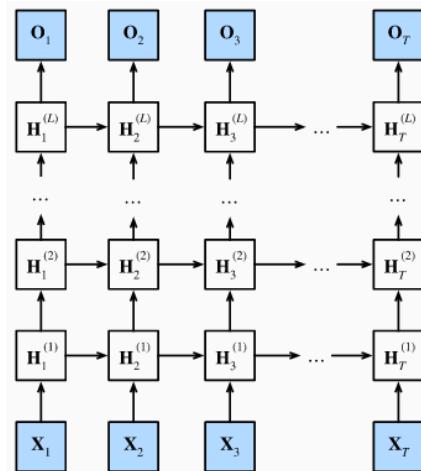
$$h_t = (1 - z_t)\Theta h_{t-1} + z_t\Theta \tilde{h}_t$$

אם $0 \rightarrow z_t$, אז $h_t \approx h_{t-1}$, כלומר לא מתחשבים ב- \tilde{h}_t , ולמעשה מעבירים את המצב הקודם כמו שהוא. אם לעומת זאת $1 \rightarrow z_t$, אז $h_t \approx \tilde{h}_t$, כלומר מתחברים המוצב הקודם כמו שהוא ולוקחים את ה- h_t .Candidate hidden state שפהו מעשה שקלול של המצב הקודם עם איבר הקלט הנוכחי. עבור כל ערך אחר של z_t , מקבלים שקלול של המצב החבוי הקודם וה- \tilde{h}_t .Candidate hidden state

ארכיטקטורה זו מאפשרת גם לזכור דברים לאורק זמן, וגם מצילה להתמודד עם בעיות הגראדיינט. אם שעור העדכון קרוב ל-1 כל הזמן, אז בעצם מעבירים את המצב החבוי כמו שהוא, ולמעשה הזיכרון נשמר לאורק זמן. בנוסף, אין בעיה של התבדרות הגראדיינט, כיוון שאין שאמם השינוי בין תא לתא לא גדול, אז הגראדיינט קרוב ל-1 כל הזמן ולא מתבדה.

6.2.3 Deep RNN

עד כה דובר על רכיבי זיכרון ייחדים, שניתן לשרשר אותם יחד ולקבל שכבה שיכולה לנתח. ניתן להרחיב את המודל הפשוט לרשת בעל מספר שכבות עומוקות.



איור 6.6 ארכיטקטורת Deep RNN

נתאר את הרשת באופן פורמלי. בכל נקודת זמן t יש וקטור כניסה $x_t \in \mathbb{R}^{n \times d}$ (וקטור בעל n איברים, כאשר כל איבר הוא מממד d). איברי הסדרה נוכנים לרשת בעלי L שכבות ו- T איברים בכניסה, כאשר עבור כל נקודת זמן t יש L שכבות (או מצבים חבויים). כל שכבה מכילה h מצבים חבויים, כאשר השכבה ה- ℓ -הׂה בנקודת זמן t מסומנת בתור $H_t^{(\ell)} \in \mathbb{R}^{n \times h}$. בכל נקודת זמן t יש גם וקטור מצוי באורך $n - q$ $o_t \in \mathbb{R}^{q \times 1}$. נסמן: $x_t = H_t^{(0)}$, ונניח שבין שכבה אחת לשניה משתמשים באקטיבציה לא ליניארית ϕ . בעזרת סימונים אלה נקבל את הנוסחה הבאה:

$$H_t^{(\ell)} = \phi_\ell \left(H_t^{(\ell-1)} w_{xh}^{(\ell)} + H_{t-1}^{(\ell)} w_{hh}^{(\ell)} + b_h^{(\ell)} \right)$$

כאשר $w_{xh}^{(\ell)} \in \mathbb{R}^{h \times n}$, $w_{hh}^{(\ell)} \in \mathbb{R}^{h \times h}$, $b_h^{(\ell)} \in \mathbb{R}^{1 \times h}$ הם הפרמטרים של השכבה החבוייה ה- ℓ . הפלט t תלוי באופן ישיר רק בשכבה ה- L , והוא מחושב על ידי:

$$o_t = H_t^{(L)} w_{hq}^{(L)} + b_q^{(L)}$$

כאשר $w_{hq}^{(L)} \in \mathbb{R}^{q \times h}$, $b_q^{(L)} \in \mathbb{R}^{1 \times q}$ הם הפרמטרים של שכבת הפלט.

ניתן כמובן להשתמש במצבים החבויים ברכיבי זיכרון LSTM או GRU, וכך לקבל Deep Gated RNN.

6.2.4 Bidirectional RNN

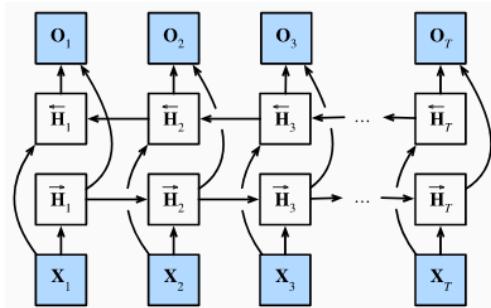
כל הרכיבים והרטות שנידונו עד כה עוסקו בסדרות סיבתיות, כלומר, סדרות בהן כל איבר מושפע מקודמי או אף לא מallow הבאים אחריו. למשל, ערך מניה ביום מסוים קשור לערכה ביום הקודמים, אך הערך שלא ביום המחרת (שכלל עד לא ידוע) לא משפיע בשום צורה על ערכה ביום הנוכחי. דוגמא נוספת – התפתחות של גל בזמן תלויה בערכי הקודמים של הגל אך אינה מושפעת ממצבי הגל בעבר. זהו אמונם המצב הייתר מצוי, אך ישנו מצבים בהם יש סדרה לאו דווקא סיבתית, כך שניתן לבדוק את הקשר בין איבריה משני הכוונים. ניקח לדוגמא את משימת ההשלמה הבאה:

I am _.

I am _ hungry.

I am _ hungry, and I can eat a big meal.

כעת נניח שבכל אחד מהמשפטים צריך לבחור את אחת מהמילות שבסט $\{ \text{happy, not, very} \}$. כמובן שסוף הביטוי, במקורה וק"י, תורם מידע שימושוטי על אייזו מילה לבחור. מודל שאינו מסוגל לנצל את המידע לאחר המילה החסורה יכול לפפסס מידע חשוב, ולרוב יכול לנחש מילה שאינה מסתדרת עם המשך המשפט הן מבחינה תחבירית והן מבחינה המשמעות. כדי לבנות מודל שמתיחס לכל חלקו המשפט, יש לתכנן ארכיטקטורה שמאפשרת לנתח סדרה משני הכוונים שלה. ארכיטקטורה כזו נקראת RNN, Bidirectional RNN, והוא למעשה מבצעת נתח של סדרה משני הכוונים שלה במקביל. באופן זה כל מצב חבוי נקבע בו זמן על ידי הנתחים של שני מצבים חבויים אחרים – זה שלפניו וזה לאחריו.



איור 6.6 ארכיטקטורת RNN Bidirectional RNN

עבור כל כניסה $x_t \in \mathbb{R}^{n \times d}$ נחשב במקביל שני מצבים חבויים – $\vec{H}_t \in \mathbb{R}^{n \times h}, \bar{H}_t \in \mathbb{R}^{n \times h}$, כאשר h זה מספר רכיבי הזיכרון בכל מצב חבוי. כל מצב מחושב באופן הבא:

$$\vec{H}_t = \phi(x_t w_{xh}^{(f)} + \vec{H}_{t-1} w_{hh}^{(f)} + b_h^{(f)})$$

$$\bar{H}_t = \phi(x_t w_{xh}^{(b)} + \bar{H}_{t+1} w_{hh}^{(b)} + b_h^{(b)})$$

כאשר $w_{xh}^{(b)} \in \mathbb{R}^{d \times h}, w_{hh}^{(b)} \in \mathbb{R}^{h \times h}, b_h^{(b)} \in \mathbb{R}^{1 \times h-1}$ $w_{xh}^{(f)} \in \mathbb{R}^{d \times h}, w_{hh}^{(f)} \in \mathbb{R}^{h \times h}, b_h^{(f)} \in \mathbb{R}^{1 \times h}$ הם הפרמטרים של המודל. לאחר החישוב של \vec{H}_t ו- \bar{H}_t מושרים אותם יחד ומתקבלים את $H_t \in \mathbb{R}^{n \times 2h}$, ובעזרתו מחשבים את המוצא:

$$o_t = H_t w_{hq} + b_q$$

כאשר $w_{hq} \in \mathbb{R}^{2h \times q}, b_q \in \mathbb{R}^{1 \times q}$ הם הפרמטרים של שכבת הפלט.

6.2.5 Sequence to Sequence Learning

ב

<https://buomssoo-kim.github.io/blog/tags/#attention-mechanism>

6. References

Vanilla:

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

LSTM, GRU:

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Deep RNN, Bidirectional RNN:

http://d2l.ai/chapter_recurrent-modern/index.html

7. Deep Generative Models

המודלים שהוצגו בפרקם הקודמים הינם מודלים דיסקרטוניים, קרי הם מאומנים לבצע פעולות על בסיס נתונים, אך לא יכולים ליצור פיסות מידע או דוגמאות חדשות בעצמן. בנויגוד אליהם קיימים מודלים גנרטיביים, المسؤولים לצור פיסות חדשות על בסיס הדוגמאות שמלמדו. באופן פורמלי, בהינתן אוסף דוגמאות $\mathcal{X} \in \mathbb{R}^{n \times d}$ ותגיות $\mathcal{Y} \in \mathbb{Z}$, מודל דיסקרטוני מאמן לשערר את ההסתברות $(x|y)Pr$. מודל גנרטיבי לעומת זאת לימד את ההסתברות $(y|x)Pr$ (או את $(x|z)$ במקרה שהתגיות אינן נתונות), כאשר y, x הן צמד נתונים של דוגמא-label.

ישנם שני סוגי עיקרים של מודלים גנרטיביים: סוג אחד של מודלים מאמן למצואו באופן מפורש את פונקציית הפילוג של הדטה הנתון, ובאזור הפילוג ליצר דוגמאות חדשות (על ידי דוגימה מההתפלגות שמלמדה). סוג שני של מודלים אינם עוסקים בשערור הפילוג של הדטה המקורי, אלא מסוגל ליצר דוגמאות חדשות בדרכים אחרות. בפרק זה נדון במודלים הפופולריים בתחום – VAE, GANs, והשייכים לשוג הראשון והשני של המודלים הגנרטיביים.

7.1 Variational AutoEncoder (VAE)

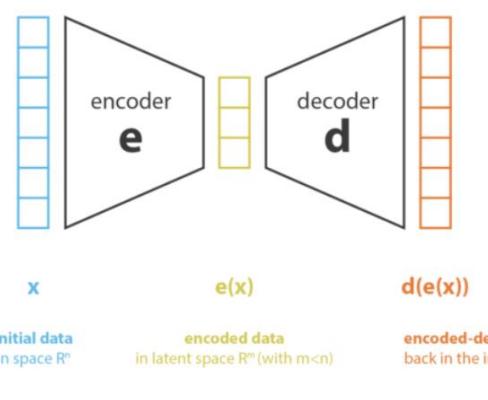
המודל הראשון הינו VAE, וכדי להבין כיצד ניתן בעזרתו לייצר מידע חדש, יש להסביר קודם כמה Autoencoders יוצרים מודדים ומה החסרונות שלהם. Autoencoder הינו רשת המאומנת לבצע הורדת מידע לדטה, תוך איבוד מידע של מידע. כדי להבין את המבנה ואופיו הפעולה שלו, נקדים מעט על הורדת מידע באופן כללי.

7.1.1 Dimensionality Reduction

במקרים רבים, הדטה אותו רצים לנתח הוא בעל ממד גובה, למשל, כל דוגמה יש מספר רב של מאפיינים (features). לרוב, לא כל המאפיינים ממשמעותיים באותה מידת. לדוגמה – מחיר מניה של חברה מסוימת מושפע ממספר רב של גורמים, אך ככל הנראה גובה ההכנסות של החברה משפיע על מחיר המניה יותר מאשר הגיל הממוצע של העובדים. דוגמא נוספת – במקרים רבות חיזוי גיל של אדם על פי תכונת הפנים שלו, לא כל הפיקסלים בתמונה הפנים יהיו בעלי אותה חשיבות לצורך החיזוי. כיוון שקשה לנתח דטה מממד גובה ולבנות מודלים עבור דטה כזו, במקרים רבים מנסים להוריד את הממד של הדטה תוך איבוד מידע מינימלי עד כמה שניתן. בהתאם לכך, במקרים רבים יציג חישוב חדש של הדטה בעל ממד יותר נמוך, כאשר הייצוג הזה מורכב מהמאפיינים העיקריים של הדטה. יש מגוון שיטות להורדת הממד כאשר הרעיון המשותף לכלן הוא לייצג את הדטה בממד נמוך יותר, בו באים לידי ביטוי רק המאפיינים המשמעותיים של הדטה.

הייצוג של הדטה בממד נמוך נקרא **היצוג הלטני** (latent representation) או **הקוד הלטני** (latent code). בcodespace, יותר קל לבנות מודלים למשימות שונות על סמך הייצוג הלטני של הדטה מאשר לעבד עם הדטה המקורי. כדי לקבל ייצוג לטני איקוטי, ניתן לאמן אותו באמצעות מנגנון הנקרא **decoder**, הבודק את יכולת השחזור של הדטה מהייצוג הלטני שלו. ככל שנינו לשחזר בצורה מדויקת יותר את הדטה מהייצוג הלטני, ככלmore אובדן המידע בתהליך הוא קטן, כך הקוד הלטני אכן מייצג בצורה אמינה את הדטה המקורי.

תהליך האימון הוא דו שלבי: פיסת מידע המוצג על ידי קטגור המאפיינים $\mathcal{R}^n \in x$ עובר דרך encoder, שמטרטטו להפיך מהדטה את הייצוג הלטני שלו $\mathcal{R}^m \in e(x)$, כאשר $n < m$. לאחר מכן התוצאה מוכנסת ל-decoder בצד השני לשחזר את הדטה המקורי, ולבסוף מתקיים מתקובל וקטורי $\mathcal{R}^m \in d(e(x))$. אם מתקיים השוויון $d(e(x)) = x$ אז למעשה לא אבד שום מידע בתהליך, אך אם לעומת זאת $d(e(x)) \neq x$ אז איבוד עקב הורדת הממד ולא היה ניתן לשחזר אותו במלואה בפועל. באופן אינטואיטיבי, אם אנו מצליחים לשחזר את הדטה המקורי מהייצוג שלו בממך הנמוך בבדיקה טוב מספיק, נראה שהייצוג הלטני הצליח להפיך את המאפיינים המשמעותיים של הדטה המקורי.



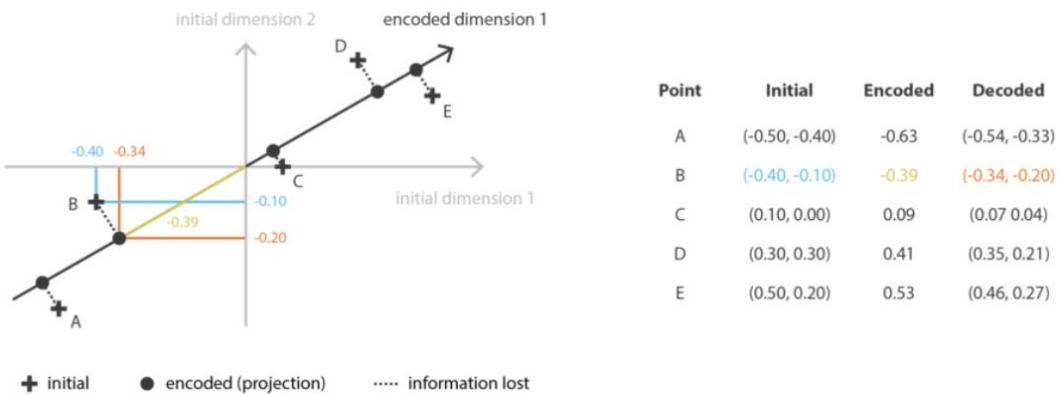
איור 7.1 ארכיטקטורת encoder-decoder.

כאמור, המטרה העיקרית של השיטות להורדת ממד הינה לקבל ייצוג לטנסי איזוטי עד כמה שניתן. הדרך לעשות זאת היא לאמן את זוג encoder-decoder השומרים על מקסימום מידע בעת הקידוד, וمبאים למינימום את שגיאת שחזור בעת הפענוח. אם נסמן בהתאם E ו-D את כל הזוגות של encoder-decoder האפשריים, ניתן לנסח את בעיית הורדת הממד באופן הבא:

$$(e^*, d^*) = \arg \min_{(e,d) \in E \times D} \epsilon(x, d(e(x)))$$

כאשר $\epsilon(x, d(e(x)))$ הוא שגיאת השחזור שבין הדטה המקורי לבין הדטה המשוחזר.

אחת השיטות השימושיות להורדת ממד שאפשר להסתמך עליה בצורה זו היא Principal Components Analysis (PCA). בשיטה זו מטילים (בצורה לינארית) דאטה מממד n לממד $m < n$, כך שהמאפיינים של הייצוג הלטנטי של הדוגמאות המקוריות יהיו אורתוגונליים. תהליך זה נקרא גם feature decorrelation, והמטרה שלו היא לסייע את המרחק האוקלידי בין הדטה המקורי לדאטה המשוחזר, בצורה לינארית גם כן, מהויצוג החדש במרחב ה- m -ממדי.

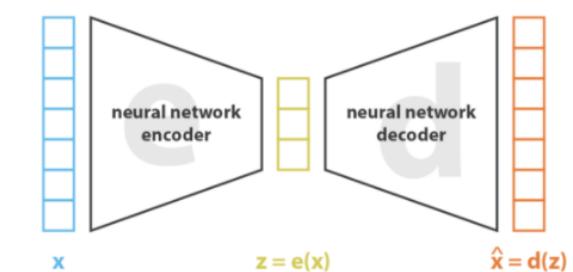


איור 7.2 דוגמא להורדת ממד בשיטת PCA.

במנוחים של encoder-decoder, ניתן להראות כי אלגוריתם PCA מ Chapman את הזוג של שמיים שני תנאים: א. ה-encoder מבצע טרנספורמציה לינארית על הדטה כך שהמאפיינים החדשים (בממד נמוך) של הדטה יהיו אורתוגונליים. ב. ה-decoder הלינארי המתאים יביא לשגיאה מינימלית במרחב אוקלידי בין הדטה המקורי לבין זה המשוחזר מהויצוג החדש. ניתן להוכיח שה-encoder האופטימלי מכיל את הווקטורים העצמיים של מטריצת covariance של מטריצת $\text{cov}(x)$, וה-decoder הוא השולוף של ה-encoder.

7.1.2 Autoencoders (AE)

ניתן לקחת את המבנה של ה-AE המtauור בפרק הקודם ולהשתמש ברשת נירונים עבור בניית הייצוג החדש ועבור השחזור. מבנה זה נקרא Autoencoder:



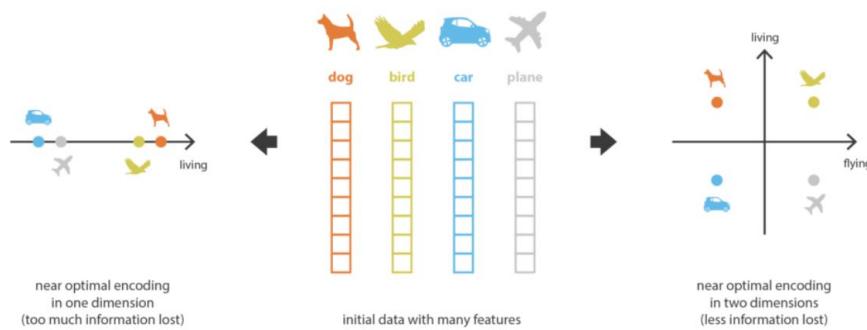
$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$

איור 7.3 – שימוש ברשתות נירונים עבור הורדת הממד והשחזור.

באופן זהה, הארכיטקטורה יוצרת לדטה צואר בקבוק מידע (information bottleneck), שמביטה שרק המאפיינים החשובים של הדטה, שבאמצעותם ניתן לשחזר אותה בדיק טוב, יושמשו לייצוג במרחב הלטנטי. במקרה הפוטוט בוככל רשת יש רק שכבה חבויה אחת והיא לא משתמשת בפונקציות הפעלה (activation functions) לא לינאריות, ניתן לראות כי ה-AE יחשוף טרנספורמציה לינארית של הדטה, שבאמצעותו ניתן לשחזרו

באופן לינארי גם כן. בדומה ל-PCA, גם רשות צזו תחפש להוריד את הממד באמצעות טרנספורמציות לינאריות של המאפיינים המקוריים אך היצוג בממד נמוך המופק על ידה לא יהיה בהכרח זהה לזה של PCA, כיון שהבדיל מ-PCA המאפיינים החדשניים (לאחר הורדת ממד) עשויים לצאת לא אורתוגונליים (קורלציה שונה מ-0).

כעת נניח שהרשאות של ה-encoder וה-decoder הן רשותות עמוקות ומשתמשות בפונקציות הפעלה לא לינאריות. במקרה זה, ככל שהארקיטקטורה של הרשותות מורכבת יותר, כך רשות ה-encoder יכולת להוריד יותר ממדים תוך יכולת באמצעות ה-decoder לבצע שחזור ללא כל איבוד מידע. באופן תיאורטי, אם ל-encoder ול-decoder יש מספיק דרגות חופש (למשל מספיק שכבות ברשת נירונית), ניתן להפחית ממד של כל דאטה לחד-מדד ללא כל איבוד מידע. עם זאת, הפקחת ממד דרסטית שכזו עלולה לגרום לדאטה המשוחזר לאבד את המבנה שלה. לכן יש חשיבותגדולה בבחירה מספר הממדים של המרחב הלטני, כך שמצד אחד אכן יבוצעיפוי של מאפיינים פחות משמעותיים ומצד שני המידע עדין יהיה בעל שימושים לשונות. כדי להמחיש את המתואר לעיל, ניקח לדוגמה מערכת שמקבלת תמונות של כלב, ציפור, מכונית ומטוס ומנסה למצוא את הפרמטרים העיקריים המבוחנים ביניהם:



.איור 7.4 דוגמא לשימוש ב-Autoencoder.

לפריטים אלו יש הרבה מאפיינים, וקשה לבנות מודל שمبוחן ביניהם על סמך כל המאפיינים. רשות נירונים מורכבת מספיק מאפשרת לבנות ייצוג של כל הדוגמאות על קו ישר, כך שיכל שפרט מסוים נמצא יותר ימינה, כך הוא יותר "חי". באופן זה אמן מתקבל ייצוג חד-ממדי, אבל הוא גורם לאיבוד המבנה הסמנטי של הדוגמאות ולא באמנת ניתנת להבין את ההפרדה ביניהן. לעומת זאת ניתן להוריד את הממד של תמונות אלו לדוו-ממד ולהתychס רק לפרמטרים "חי" ו"עף", וכך לקבל הבחנה יותר ברורה בין הדוגמאות. כזכור שהפרדה זו היא הרבה יותר פשוטה מאשר הסתכילות על כל הפרמטרים (הפיקסלים) של הדוגמאות. דוגמא זו מראה את החשיבות שיש בבחירה הממדים של ה-encoder.

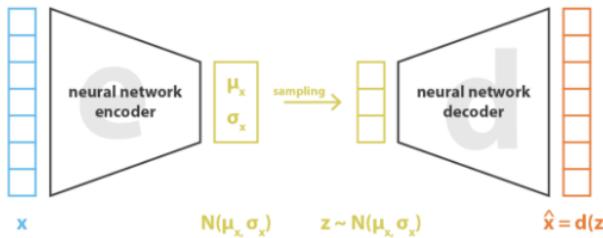
7.1.3 Variational AutoEncoders (VAE)

ניתן לקחת את AE ולהפוך אותו למודל גנרטיבי, כלומר מודל שמוסgal לייצר בעצמו דוגמאות חדשות שאכן מתפלגות כמו הפולוג של הדאטה המקורי. אם מדובר בדומין של תמונות למשל, אז נרצה שהמודול יהיה מסוגל לייצר תמונות שנראות אוטנטיות ביחס לדאטה עלי'ו אומן. AE מאמן לייצג את הדאטה בממד נמוך, שליקח בחשבון את המאפיינים העיקריים, ולאחר מכן משחזר את התוצאה לממד המקורי. אולם, מגננון זה אינו מאפשר להשפיע על האופן בו הדאטה מייצג במרחב הלטני. אם יוגרך וקטור כלשהו מהמרחב הלטני – קרוב לוודאי שהוא לא יהווה ייצוג שdomה לדאטא המקורי. אם היינו מכנים אותו ל-decoder, סביר להניח שההתוצאה לא תהיה דומה בכלל לדאטא המקורי. למשל אם AE אומן על אוסף של תמונות של כלבים ודוגמים באקריא וקטור מהמרחב הלטני שלו, הסיכוי לקבל תמונה כלב כלשהו לאחר השחזור של ה-decoder הינו אפסי.

כדי להתמודד עם בעיה זו, ניתן להשתמש ב-VAE (Variational AutoEncoder). בשונה מ-AE שליקח דאטא ורק בונה לו ייצוג מממד נמוך, VAE קובע התפלגות פרוירית למרחב הלטני z – למשל התפלגות נורמלית עם תוחלת 0 ומטריצת covariance \mathbb{I} . בהינתן התפלגות זו, רשות ה-encoder מאומנת לקובל דאטא x ולהוציא פרמטרים של התפלגות פואיסטרורית $x|z$, מתוך מטרה למזער כמה שניתן את המרחק בין התפלגות z ו- $x|z$. לאחר מכן וקטורים מההתפלגות הפואיסטרורית $x|z$ (הנתונה על ידי הפרמטרים המוחשיים ב-encoder), וمبرירים אותן דרך decoder כדי לייצר פרמטרים של התפלגות $x|z$. חשוב להבהיר שאם הדאטה המקורי הוא תמונה המורכבת מאוסף של פיקסלים, אז בmoץ יתקבל $x|z$ לכל פיקסל $z|z$ בנפרד ומההתפלגות זו דוגמים נקיים שתיציג את ערך הפיקסל בתמונה המשוחזרת.

באופן זהה, הלמידה דואגת לא רק להורדת הממד של הדאטה, אלא גם להתפלגות המשוררת על המרחב הלטנטי. כאשר ההתפלגות המותנית בmoza $x|z$ טובה, קרי קרובה להתפלגות המקורית של x , ניתן בעזרתה גם ליצור דוגמאות חדשות, ובעומם מתקבל מודל גנרטיבי.

כאמור, ה-encoder מנסה לייצג את הדאטה המקורי באמצעות התפלגות במרחב נורמלי – covariance ומטריצת covariance: $\text{covariance} = \text{cov}(x, z) = N(\mu_x, \sigma_x)$. חשוב לשים לב להבדל בתפקיד של ה-decoder – בעוד שב-AE הוא מעד לתהילך האימון בלבד ובפועל מה שחווסף זה הייצוג הלטנטי, ב-VAE ה-decoder חשוב לא פחות מאשר הייצוג הלטנטי, כיון שהוא שמש ליצירת דאטה חדש לאחר תהילך האימון, או במקרים אחרות, הוא הופך את המערכת למודל גנרטיבי.



איור 7.5 ארכיטקטורה של VAE.

לאחר שהציג המבנה הכללי של VAE, ניתן לתאר את תהילך האימון, ולשם כך נפריד בשלב זה בין שני החלקים של ה-VAE. encoder מאמין רשות שמקבלת דוגמאות מסט האימון, ומנסה להפיק מהן פרמטרים של התפלגות $x|z$ הקוראים כמה שניתן לפרמטרים של התפלגות הפרויוית z , שכאמרנו נקבעה מראש. מההתפלגות הנלמדת זו דוגמים וקטורים לטנטים חדשים ומעבירים אותם ל-decoder. decoder מבצע את הפעולה ההפוכה – לוקח וקיטור שנדגם מהמרחב הלטנטי $x|z$, ומיציר באמצעותו דוגמא חדשה שאמורה להיות דומה לדאטה המקורי. תהילך האימון היה כזה שימזעր את השגיאה של שני חלקים VAE – גם $x|z$ שבmoza ה-VAE יתגלו כמיידר קרוב לא-המקור, וגם התפלגות $x|z$ תהיה כasma שיותר קרובה להתפלגות הפרויוית z .

ונתאר באופן פורמלי את בעיית האופטימיזציה ש-VAE מנסה לפתור. נסמן את הווקטורים של המרחב הלטנטי $-z$, את הפרמטרים של ה-decoder θ , ואת הפרמטרים של ה-encoder λ . כדי למצאו את הפרמטרים האופטימליים של שתי הרשומות, נרצה להביא למקסימום את $L(\theta; \lambda)$, כלומר למקסם את הנראות המרבית של סט האימון תחת θ . כיון שפונקציית $\log p$ מונוטונית, יוכל לקחת את לוג ההסתברות:

$$L(\theta) = \log p(x; \theta)$$

אם נביא למקסימום את הביטוי הזה, נקבל את θ האופטימלי. כיון שלא ניתן לחשב במפורש את $p(x; \theta)$, יש להשתמש בקירוב. נניח והפלט של ה-encoder הוא בעל התפלגות $(\lambda; x|z)q$ (מה ההסתברות לקבל את z בהינתן x בכנסה), וננסה לייצג את התפלגות זו בעזרת רשות נוירונים עם סט פרמטרים λ . כתע ניתן לחלק ולהכפיל את $L(\theta; \lambda) = q(z; \lambda)$

$$\log p(x; \theta) = \log \sum_z p(x, z; \theta) = \log \sum_z q(z; \lambda) \frac{p(x, z; \theta)}{q(z; \lambda)} \geq \sum_z q(z; \lambda) \log \frac{p(x, z; \theta)}{q(z; \lambda)}$$

כאשר אי השווין האחרון נובע מאי-שוויון יונסן, והביטוי שמיין לאי השוויון נקרא Evidence Lower Bound (ELBO). ניתן להוכיח שההפרש בין ELBO(θ, λ) לבין הערך שלפני הקירוב הוא המהלך בין שתי התפלגות $(z|x, q, p)$, שנקרא Kullback-Leibler divergence ומסומן ב- $\mathcal{D}_{KL}(q(z|x, \theta) \| p(z|x, \theta))$:

$$\log p(x; \theta) = ELBO(\theta, \lambda) + \mathcal{D}_{KL}(q(z; \lambda) \| p(z|x; \theta))$$

אם שתי התפלגות זהות, אז מרחיק \mathcal{D}_{KL} ביןיה הוא 0 ומתקיים שוויון: $\log p(x; \theta) = ELBO(\theta, \lambda)$. כזכור, אנחנו מחפשים למקסם את פונקציית המבחן $\log p(x; \theta)$, וכעת בעזרת הקירוב ניתן לרשום:

$$L(\theta) = \log p(x; \theta) \geq ELBO(\theta, \lambda)$$

$$\rightarrow \theta_{ML} = \arg \max_{\theta} L(\theta) = \arg \max_{\theta} \max_{\lambda} ELBO(\theta, \lambda)$$

כעת ניתן בעזרת שיטת Gradient Descent (GD) למצוא את האופטימום של הביטוי, וממנו להפיק את הפרמטרים האופטימליים של ה-encoder ושל ה-decoder. נפתח יותר את ה-ELBO(θ, λ) עבור VAE, ביחס לשתי התפלגות:

$p(x|z; \theta)$ – ההסתברות ש-decoder עם סט פרמטרים θ יוציא x בהינתן z .
 $q(z|x; \lambda)$ – ההסתברות ש-encoder עם סט פרמטרים λ יוציא את z_i בהינתן x בכניסה.

לפי הגדרה:

$$ELBO(\theta, \lambda) = \sum_z q(z|x; \lambda) \log p(x, z; \theta) - \sum_z q(z|x; \lambda) \log q(z|x; \lambda)$$

את הביטוי $p(x, z) = p(x|z) \cdot p(z)$ ניתן לפתח לפי חוק ב'יוס

$$\begin{aligned} &= \sum_z q(z|x; \lambda) (\log p(x|z; \theta) + \log p(z; \theta)) - \sum_z q(z|x; \lambda) \log q(z|x; \lambda) \\ &= \sum_z q(z|x; \lambda) \log p(x|z; \theta) - \sum_z q(z|x; \lambda) (\log q(z|x; \lambda) - \log p(z; \theta)) \\ &= \sum_z q(z|x; \lambda) \log p(x|z; \theta) - \sum_z q(z|x; \lambda) \frac{\log q(z|x; \lambda)}{\log p(z; \theta)} \end{aligned}$$

הביטוי השני לפי הגדרה שווה ל- $-\mathcal{D}_{KL}(q(z|x; \lambda) \| p(z; \theta))$, לכן מתקבל:

$$= \sum_z q(z|x; \lambda) \log p(x|z; \theta) - \mathcal{D}_{KL}(q(z|x; \lambda) \| p(z))$$

הביטוי הראשון הוא בדיק התוחלת של $\log p(x|z; \theta)$. תחת ההנחה ש- z מתפלג נורמלית, ניתן לרשום:

$$= \mathbb{E}_{q(z|x; \lambda)} \log N(x; \mu_\theta(z), \sigma_\theta(z)) - \mathcal{D}_{KL}(N(\mu_\lambda(x), \sigma_\lambda(x)) \| N(0, I))$$

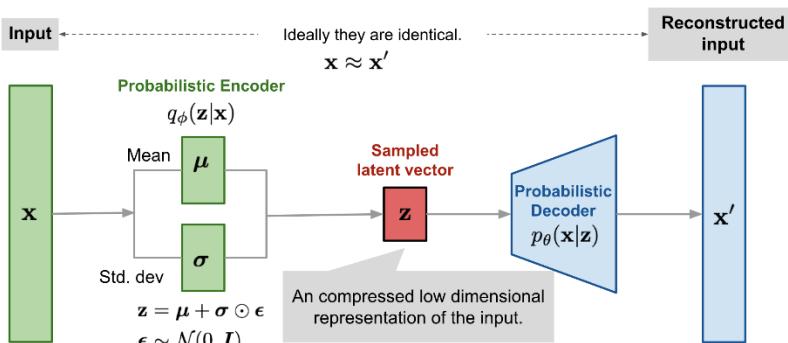
כדי לחשב את התוחלת ניתן פשוט לדוגמ דוגמאות מההתפלגות $(x|z)$ ולקבל:

$$\mathbb{E}_{q(z|x; \lambda)} \log N(x; \mu_\theta(z), \sigma_\theta(z)) \approx \log N(x; \mu_\theta(z), \sigma_\theta(z))$$

עבור הביטוי השני יש נוסחה סגורה:

$$\mathcal{D}_{KL}(N(\mu, \sigma^2) \| N(0, I)) = \frac{1}{2}(\mu^2 + \sigma^2 - \log \sigma^2)$$

cut משיש בידינו נוסחה לחישוב פונקציית המחיר, נוכל לבצע את תהליכי הלמידה. יש לשם לב שפונקציית המחיר המקורית הייתה תליה רק ב- θ , אך באופן שיפתחנו אותה היא למעשה דואגת גם למצער הפרש בין הכנסה של VAE לבין המוצא שלו, וגם למצער המרחק בין ההתפלגות הפרוירית של z לבין ההתפלגות $x|z$ שבוצעה ה-encoder.



$$\begin{aligned} x_t &\rightarrow \mu_\lambda(x_t), \Sigma_\lambda(x_t) \rightarrow z_t \sim \mathcal{N}(\mu_\lambda(x_t), \Sigma_\lambda(x_t)) \rightarrow \mu_\theta(z_t), \Sigma_\theta(z_t) \\ ELBO &= \sum_t \log \mathcal{N}(x_t; \mu_\theta(z_t), \Sigma_\theta(z_t)) - \mathcal{D}_{KL}(\mathcal{N}(\mu_\lambda(x_t), \Sigma_\lambda(x_t)) \| \mathcal{N}(0, I)) \end{aligned}$$

איור 7.6 תהליכי הלמידה של VAE

כאשר נתון אוסף דוגמאות X , ניתן להעביר כל דוגמא x ב-encoder ולקבל עבורה את σ_x . לאחר מכן דוגמים מוקטור לנטוי z מההתפלגות עם פרמטרים אלו, מעבירים אותם ב-decoder ומקבלים את σ_z . לאחר התהילר ניתן להציג את הפרמטרים המתקבלים ב-ELBO ולחשב את ערך פונקציית המחר. ניתן לשים לבשה ELBO מורכב משני איברים – האיבר הראשון משער את הדמיון בין הדוגמא שבסכינסה לבין ההתפלגות שמתקבלת במצוא, והאיבר השני מבצע רגולרייזציה להתפלגות הפרוריות במרחב הlatent. הרגולרייזציה גורמת לכך שההתפלגות במרחב הלטנטי z תהיה קרובה עד כמה שנitin לההתפלגות הפרורית z . אם ההתפלגות במרחב הlatent קרובה להתפלגות הפרורית, אז ניתן באמצעות decoder ליצור דוגמאות חדשות, ובמובן זהה ה-VAE הוא מודל גנרטיבי.

הדגימה של z מההתפלגות במרחב הlatent יוצרת קשיי בחישוב הגדריאנט של ה-ELBO, שכן בדרך כלל מוצעים Reparameterization trick – דוגמים σ_0 מההתפלגות נורמלית סטנדרטית, ואז כדי לקבל את ערך הדגימה של z משתמשים בפרמטרים של ה-encoder: $(x) = \mu + \sigma_0 z$. בגישה זו כל התהילר יהיה דטרמיניסטי – מגרילים σ_0 מראש ואז רק נשאר לחשב באופן סכמטי את התפשטות הערך ברשות.

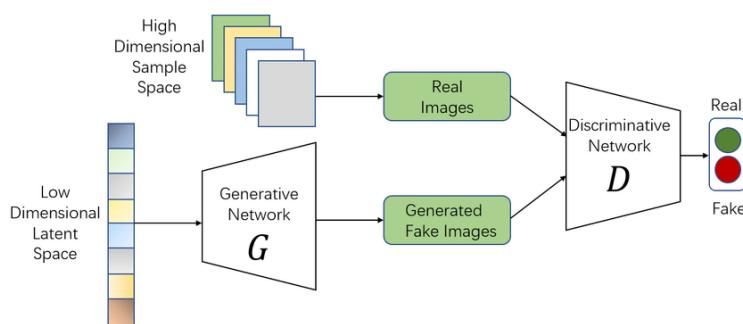
7.2 Generative Adversarial Networks (GANs)

גישה אחרת של מודל גנרטיבי נקראת GANs או בקיצור Generative Adversarial Networks, ובשונה מ-VAE בגישה זו לא מנוטים לשער התפלגות של>Data בצורה מפורשת (על ידי מציאת הפרמטרים הממנסים את הנראות המירבית של סט האימון), אלא יוצרים>Data באופן אחר. הרעיון הוא לאמן שתי רשותות במקביל – רשות אחת שלומדת ליצור דוגמאות, ורשת שנייה שלומדת להבחן בין דוגמא אמיתית מסט האימון לבין תמונה סינטטית שנוצרה על ידי הרשות הראשונה. הרשות הראשונה מאומנת ליצור דוגמאות שיגרמו לרשות השנייה לחושם שהן אמיתיות, בזמן שהמטרה של הרשות השנייה היא לא לחת לרשת הראשונה לבלב אותה. באופן זה הרשות הראשונה מהווה למעשה מודל גנרטיבי, שלאחר שלב האימון היא מסוגלת לייצר>Data סינטטי שלא ניתן להבין בין>Dataจรיא.

7.2.1 Generator and Discriminator

בפרק זה נסביר את המבנה של ה-GAN הקלאסי שהומצא בשנת 2014 על ידי Ian Goodfellow ושותפיו. נזכיר כי קיימים מאות רבות של וריאנטים שונים של GAN שהוצעו מאז, ועודין תחום זה פעיל מאוד מחקרים.

כאמור, GAN מבוסס על שני אלמנטים מרכזיים – רשות שיצירת>Data (generator) ורשת שמכריעה האם>Data ה-*real* ה-*fake* סינטטי או אמיתי (discriminator), כאשר האימון נעשה על שתי הרשותות יחד. ה-*discriminator* מקבל כקלט *real* או *fake* *sample* מה-*generator* ופידבק מדויק לאם>Data הוא אמיתי או סינטטי. ה-*generator* מיציר דוגמאות ומתקבל פידבק מה-*discriminator* וכך לומד ליצור דוגמאות שנראות אמיתיות. נסמן את ה-*generator* ב-*G* ואת *discriminator* ב-*D*, ונקבל את הסכמה הבאה:



איור 7.7 ארכיטקטורת GAN.

ה-*discriminator* *D* הוא למעשה מושג שהפלט שלו הוא היחסותיות שהקלט היוו דוגמא אמיתית, ונסמן ב- (x) את היחסותיות זו. כדי לאמן את ה-*discriminator* נרצה להשיג שני דברים: א. למקסם את (x) עבור *D* מסט האימון, כלומר, לטעות כמה שפחות בדוחוי DATA אמית. ב. למזער את (x) עבור DATA סינטטי, כלומר, להחות כמה שיותר דוגמאות סינטטיות שייצרו על ידי ה-*generator*. באופן דומה נרצה לאמן את ה-*generator* כך שהדוגמאות שהוא מייצר תהינה כמה שיותר דומות לדוגמאות אמיתיות, כלומר generator-*D* מועוני לגרום לא-*discriminator* להוציא ערכים כמה שיותר גבוהים עבור הDATA הסינטטי שהוא מייצר. בשבייל לאמן ייחד את שני חלקים המודל, נבנה פונקציית מבחן בעלת שני איברים, באופן הבא:

$$V(D, G) = \min_G \max_D \mathbb{E}_{x \sim Data} \log D(x) + \mathbb{E}_{z \sim Noise} \log (1 - D(G(z)))$$

נסביר את הביטוי המתkeletal: ה-discriminator מעוניין למקסם את פונקציית המחיר, כך שהערך של $(x) D$ יהיה כמו שיטור קרוב ל-1 ו- $(z) D$ יהיה כמו שיטור קרוב ל-0. ה-generator לעומת זאת רוצה להביא למינימום את פונקציית המחיר, כך ש- $(z) D$ יהיה כמו שיטור קרוב ל-1, כלומר ה-generator יחשוב ש- $(z) G$ הוא דата אמיתית.

כעת האימון נעשה באופן איטרטיבי, כאשר פעם אחת מקבעים את G ומאמנים את D , ופעם אחרת מקבעים את D ומאמנים את G . אם מקבעים את G , אז למעשה מאמנים מסווג בינהר, כאשר מփשים את האופטימום התלוי בוקטור הפרמטרים ϕ_d :

$$\max_{\phi_d} \mathbb{E}_{x \sim Data} \log D_{\phi_d}(x) + \mathbb{E}_{z \sim Noise} \log \left(1 - D_{\phi_d}(G_{\theta_g}(z)) \right)$$

אם לעומת זאת מקבעים את D , אז ניתן להתעלם מהאיבר הראשון כיון שהוא פונקציה של ϕ_d בלבד וקבוע ביחס ל- θ_g . לכן נשאר רק לבדוק את הביטוי השני, שמחפש את ה-generator שמייצר דата שנראה אמיתית בצורה הטובה ביותר:

$$\min_{\theta_g} \mathbb{E}_{z \sim Noise} \log \left(1 - D_{\phi_d}(G_{\theta_g}(z)) \right)$$

כאמור, המטרה היא לאמן את G בעזרת D (במצבו הנוכחי), כדי שהיא מסוגל ליצור דוגמאות הנראות אוטנטיות. האימון של ה-generator נעשה באמצעות Gradient Descent (מצער פונקציית המחיר ביחס ל- θ_g), והאימון של ה-discriminator נעשה באמצעות Gradient Ascent (מקסום פונקציית המחיר ביחס ל- ϕ_d). האימון מתבצע במספר מוסים של Epochs, כאשר כאמור מאמנים לסירוגין את G ו- D . בפועל דוגמים mini-batch בגודל m מסט האימון (x_m, z_1, \dots, z_m) ו- m דוגמאות של רעש (x_1, \dots, z_1), ומכוונים את הקלט ל- G . הגרדיאנט של פונקציית המחיר לפि הפרמטרים של ה-generator במהלך האימון מחושב באופן הבא:

$$\nabla_{\theta} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\theta} \sum_{i=1}^m \log \left(1 - D_{\phi}(G_{\theta}(z_i)) \right)$$

וכאשר מאמנים את ה-generator, הגרדיאנט נראה כך:

$$\nabla_{\phi} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\phi} \sum_{i=1}^m \log D_{\phi}(x_i) + \log \left(1 - D_{\phi}(G_{\theta}(z_i)) \right)$$

נוהג לבצע מודיפיקציה קטנה על פונקציית המטרה של ה-generator. כיון שבהתחלת הדגימות המיוצרות על ידי ה-generator לא דומות לחולוטן לאלו מסט האימון, ה-generator מזיהה אותן בקלות כمزיפות. כתוצאה לכך הביטוי $(z) D$ מקבל ערכים מאד קרובים ל-0, ומילא גם הביטוי $(1 - D(G(z))) \log(D(G(z)))$ עניין ל-0. עניין זה גורם לכך שהgradian של ה-generator גם מאד קטן, ולכן כמעט ולא מתבצע שיפור ב-generator. לכן במקומות לחפש מינימום של הביטוי $\mathbb{E}_{z \sim Noise} \log(1 - D(G(z)))$ מփשים מינימום לביטוי $(D(G(z)) \log(D(G(z)))$. הביטויים לא שווים לגמרי אך שניהם מוביילים לאוותו פתרון של בעיית האופטימיזציה אותה הם מייצגים, והביטוי החדש עובד יותר טוב נומריית ומצליח לשפר את ה-generator בצורה עיליה יותר.

הערכים האופטימליים של G ו- D :

כזכור, פונקציית המחיר הינה:

$$V(D, G) = \min_G \max_D \mathbb{E}_{x \sim Data} \log D(x) + \mathbb{E}_{z \sim Noise} \log \left(1 - D(G(z)) \right)$$

כעת נרצה לחשב מה הערך האופטימי של discriminator generator עבור נתון, ובוורו לחשב את הערך של פונקציית המחיר. לשם הנוחות נסמן את התפלגות הדטה האמיתית ב- r_g , ואת התפלגות הדטה הסינטטי המייצר על ידי ה-generator ב- r_g . עבור G קבוע, ניתן לרשום את פונקציית המחיר כך:

$$V(D, G) = \int_x p_r(x) \log D(x) + p_g(x) \log(1 - D(x)) dx$$

כדי להביא את הביטוי זהה למקסימום, נרצה למקסם את האינטגרד עבור כל ערך x האפשרי. لكن הפונקציה לה מעוניינית למצוא אופטימום הינה:

$$f(D(x)) = p_r(x) \log D(x) + p_g(x) \log(1 - D(x))$$

נזכור את הביטוי האחרון ונשווה ל-0 כדי למצוא את הערך האופטימלי של $D(x)$ עבור x נתון:

$$\begin{aligned} \frac{\partial f(D(x))}{\partial D(x)} &= \frac{p_r(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} = 0 \\ \rightarrow p_r(x)(1 - D(x)) - p_g(x)D(x) &= 0 \\ D(x)_{opt} &= \frac{p_r(x)}{p_r(x) + p_g(x)} \end{aligned}$$

הביטוי שהתקבל הינו הערך האופטימלי של discriminator קבוע (ביחס לקלט x נתון). נשים לב שבעור המקרה בו-h-GAN מצליח לייצר דוגמאות שנראות אמיתיות לחלווטין, כלומר ($p_g(x) = p_r(x)$, אז $\text{מתק}' = \frac{1}{2}D$). הסתברות זו משמעותה שהיא discriminator לא יודע להחליט לגבי הקלט המתתקבל, והוא קובל שהסתברות שהקלט אמיתי זהה לזו שהקלט סינטטי.

כעת נבחן מהו ערך פונקציית המחיר כאשר D אופטימלי:

$$\begin{aligned} V(G, D) &= \mathbb{E}_{x \sim Data} \log D(x) + \mathbb{E}_{z \sim Noise} \log(1 - D(G(z))) \\ &= \mathbb{E}_{x \sim Data} \log \left(\frac{p_r(x)}{p_r(x) + p_g(x)} \right) + \mathbb{E}_{z \sim Noise} \log \left(1 - \left(\frac{p_r(x)}{p_r(x) + p_g(x)} \right) \right) \\ &= \mathbb{E}_{x \sim Data} \log \left(\frac{p_r(x)}{p_r(x) + p_g(x)} \right) + \mathbb{E}_{z \sim Noise} \log \left(\frac{p_g(x)}{p_r(x) + p_g(x)} \right) \\ &= \mathbb{E}_{x \sim Data} \log \left(\frac{p_r(x)}{\frac{(p_r(x) + p_g(x))}{2}} \right) + \mathbb{E}_{z \sim Noise} \log \left(\frac{p_g(x)}{\frac{(p_r(x) + p_g(x))}{2}} \right) - \log 4 \end{aligned}$$

הביטוי המתתקבל הינו המרחק בין התפלגיות p_r ו- p_g , והוא נקרא Jensen-Shannon divergence ומוסומן ב- \mathcal{D}_{JS} . מרחק זה הינו גרסה סימטרית של Kullback–Leibler divergence (\mathcal{D}_{KL}), ובעור שתי התפלגיות P , Q הוא מוגדר באופן הבא:

$$\mathcal{D}_{JS} = \frac{1}{2} \mathcal{D}_{KL}(P||M) + \frac{1}{2} \mathcal{D}_{KL}(Q||M), M = \frac{1}{2}(P + Q)$$

קיים שבעור D אופטימלי, פונקציית המחיר שווה למרחק \mathcal{D}_{JS} בין p_g עד כדי קבוע, ובאופן מפורש:

$$V(G, D_{opt}) = \mathcal{D}_{JS}(p_r, p_g) - \log 4$$

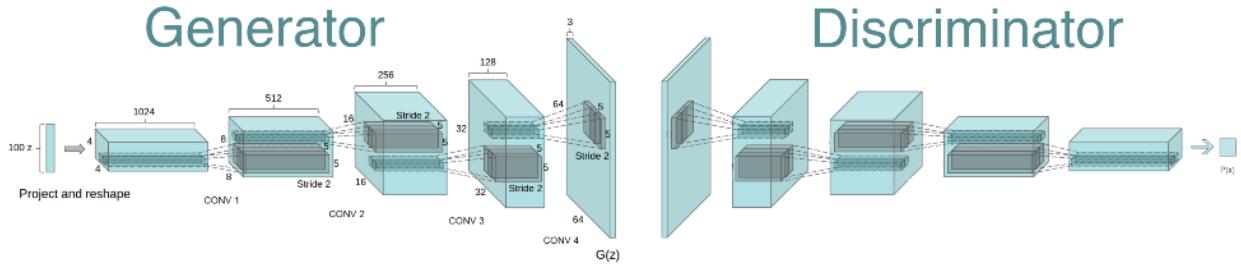
כאשר G אופטימלי ומתקיים ($p_g(x) = p_r(x)$, אז המרחק בין התפלגיות שווה 0, כלומר $\mathcal{D}_{JS}(p_r, p_g) = 0$), אך מתקובל:

$$V(G_{opt}, D_{opt}) = -\log 4$$

יש משמעות גדולה לביטוי שהתקבל – ככל שנצליח למזרע יותר את ($\mathcal{D}_{JS}(p_r, p_g)$, כך נצליח לקבל GAN יותר טוב).

7.2.2 Deep Convolutional GAN (DCGAN)

כפי שהוסבר בפרק 5, רשותות קונבולוציה יעילות יותר בהדמיין של תמונות מאשר רשותות FC. לכן היה טבעי לחקות רשותות קונבולוציה ולבנות בעזרתן generator ו-discriminator עבור דומיין של תמונות. ה-generator מקבל וקטור אקראי ומעביר אותו דרך רשת קונבולוציה על מנת ליצור תמונה, וה-discriminator מקבל תמונה ומעביר אותה דרך רשותות קונבולוציה שועשה סיווג ביןארי אם התמונה אמיתי או סינטטי. DCGAN הומצא ב-2015 ומאז פותחו רשותות שמייצרות תמונות יותר איכותיות הן מבחינת הדמיון שלהן לתמונות אמיתיות, אך החישבות של המאמר נעצה בשימוש ברשותות קונבולוציה עבור GAN שמיועד לדומיין של תמונות.



איור 7 ארכיטקטורת DCGAN.

7.2.3 Conditional GAN (cGAN)

לעתים מודל גנרטיבי נדרש ליצור דוגמא בעלת מאפיין ספציפי ולא רק דוגמא שנראית אוטנטית. למשל, עבור אוסף תמונות המציגות את הספורות 0-9, ונרצה שה-GAN יוצר תמונה של ספרה מסוימת. במקרה אחד, במקרה מסוים של הפלט הכניסה z , ה-GAN מקבל תנאי נוסף על הפלט אותו הוא צריך ליצור, כמו למשל ספרה ספציפית אחרת רוצים לקבל. GAN כזה נקרא conditional GAN (או בקיצור cGAN), ופונקציית המחר שלו דומה מאוד לפונקציית המחר של GAN רגיל למעט העבודה שהביטויים הופכים להיות מותניים:

$$\mathcal{L}_c(D, G) = \min_G \max_D \mathbb{E}_{x \sim Data} \log D(x|y) + \mathbb{E}_{z \sim Noise} \log (1 - D(G(z|y)))$$

7.2.4 Pix2Pix

כפי שראינו, ה-GAN הקלאסי שתוואר לעיל מסוגל ליצור דוגמאות חדשות מוקטור אקראי z , המוגרל מהתפלגות מסוימת (בדרכו כלל התפלגות גאוסית סטנדרטית), אך זה לא מוכחה. ישן גישות נוספת לצור דאטה חדש, כמו למשל ייצור תמונה חדשה על בסיס קווי מתרח כללים שלה. סט האימון במקורה זה בניית מזגות של תמונות והסקציות שלהן.

שיטת Pix2Pix משתמש בארכיטקטורה של GAN אך במקומם לדגם את וקטור z מהתפלגות ממשה, Pix2Pix מקבלת סקיצה של תמונה בתור הקלט, וה-generator לומד להפוך את הסקיצה לתמונה אמיתי. הארכיטקטורה של ה-discriminator נשארת ללא שינוי ביחס למאה שתואר קודם לכן (פרט להתאמתה לבינה הקלט), אך ה-discriminator מקבל זוג תמונות – אם משתנה – במקומם לקבל תמונת ולבצע עליה סיווג ביןארי, הוא מקבל זוג תמונות – את הסקיצה ואת התמונה (פעם שנייה מסט האימון המתאימה לסקיצה S ופעם זאת שמיוצרת על ידי ה-generator על בסיס S). על ה-discriminator לקבוע האם התמונה היא אכן תמונה של הסקיצה או תמונה סינטטית. ווריאציה זו של ה-GAN משנה גם את פונקציית המחר – כתעת ה-generator צריך ללמוד שני דברים – גם ליצור תמונות טובות כרשה-discriminator יאמין שהן אמיתיות, וגם למזער את המרחק בין תמונה שנוצרת לבין תמונה אמיתיות השיכת לסקיצה.

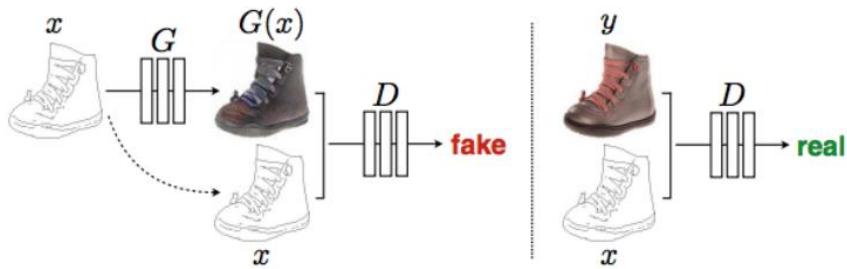
כעת נסמן תמונה אמיתיות השיכת לסקיצה ב- y , ונרשום את פונקציית המחר כשי חלקיים נפרדים – cross entropy ומודרך אוקלידי L_1 בין תמונת המקור לבין הפלט:

$$V(D, G) = \min_G \max_D (\mathbb{E}_{x,y} (\log D(x,y) + \log (1 - D(x,G(x)))))$$

$$\mathcal{L}_{L1}(G) = \min_{\theta_g} \mathbb{E}_{x,y} \|G(x) - y\|_1$$

$$\mathcal{L}(G,D) = V(D,G) + \lambda \mathcal{L}_{L1}(G)$$

ניתן להסתכל על pix2pix המפה תמונה לתמונה (image-to-image translation). נציין שבמקרה זה הקלט והפלט של pix2pix שייכים לתחומים (domains) שונים (סקיצה ותמונה רגילה).

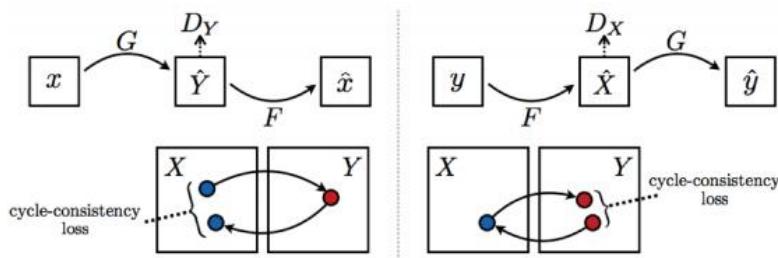


.Image-to-Image Translation - Pix2Pix

7.2.5 CycleGAN

ב-Pix2Pix הدادה המקורי הגיע בזוגות – סקיצה ואיתה תמונה אמיתית. זוגות של תמונות זה לא דבר כל כך זמין, ולכן שיפרו את תהליכי האימון כך שהיה ניתן לבצע אותו על שני סטים של>Data מאוחדים שונים. הארכיטקטורה עבורי המשימה זו מורכבת משני generators – בהתחלה מכניים דוגמא מהדומין הראשוני x ל- G , G -generator שמנסה להפוך אותו לדוגמא מהדומין השני y , והפלט נכנס ל- D_y discriminator השני שמנסה לשחרר את המקור x . המוצא של G -generator נכנס לא רק ל- D_y אלא גם ל- D_x discriminator שמנס את התמונה שהתקבלת הינה אמיתית או לא (עבור הדומין של y). ניתן לבצע את התהליכי זהה באופן דו-אי עבור y – מכניים את y ל- F על מנת לקבל את x ואת המוצא של F -generator ל- D_x discriminator בכך לבער סיווג ביןאי ול- G על מנת לנס את המוקור. ה- G -generator השני F נדרש לשפר את תהליכי הלמידה – לאחר ש- x הופך ל- y , G , ניתן לקבל חזרה את x אם נעביר את y דרך F מתוך ציפייה לקבל $x \approx (G(y))$. התהליכי של השוואת הכנסה למוצא נקרא cycle-consistency, והוא מוסיף עוד איבר לפונקציית המבחן, שמטרתו לסייע עד כמה שניתן את המרחק בין התמונה המקורית לתמונה המשוחזרת:

$$V(D_x, D_y, G, F) = \mathcal{L}_{GAN}(G, D_y, x, y) + \mathcal{L}_{GAN}(F, D_x, x, y) \\ + \lambda (\mathbb{E}_x \|F(G(x)) - x\|_1 + \mathbb{E}_y \|G(F(y)) - y\|_1)$$

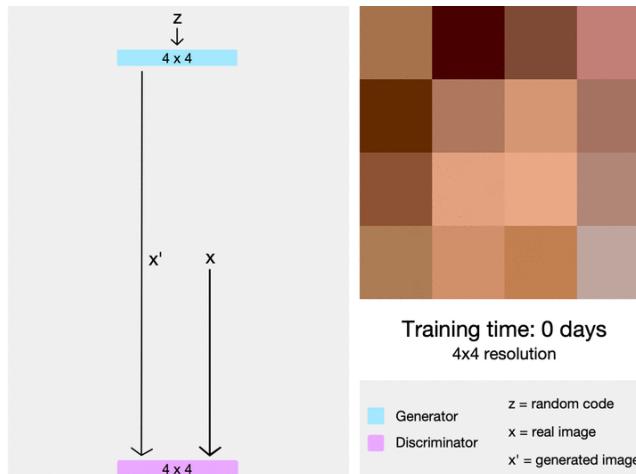


.CycleGAN

7.2.6 Progressively Growing GAN (ProGAN)

כאמור לעיל, עבור דומיין של תמונות, הגיוני להשתמש קונבנצייה עבור יצירת תמונות חדשות, וזה הרעיון הבסיסי שמאחורי DCGAN. למרות היכולת המרשימה של DCGAN ביצירה של תמונה באיכות גבוהה, יכולת זאת מוגבלת לתמונות בגודל מסוים. ככל שהרזולוציה של תמונה גבוהה יותר, כך יותר קל להבחין אם תמונה זו אמיתית או נזירה על ידי רשת גברטיבית. בעוד Sh-GAN מצליח ליצור תמונות שנראות אוטנטיות יותר, כמו למשל רזולוציה של 64×64 , 32×32 , 16×16 , 8×8 , 4×4 , והוא היה הראשון שפרק את מחסום הרזולוציה והצליח ליצור תמונות אינטלקטואליות מאוד (במאמר המקורי של ProGAN – עד רזולוציה של 1024×1024) בלי שהיה ניתן להבחין שתמונות אלה סינטטיות. אולם עוד לפני ProGAN היו GANs שהצליחו ליצור תמונה בעלת רזולוציה גבוהה מתמונה אחרת ברזולוציה גבוהה (az2x2k), אך זו ממשאה אחרת, מכיוון שבשבילו צריך רק ללמוד לשנות תכונות של תמונות קלות, ולא ליצור תמונה חדשה לגמרי מאפס.

הרעיון העיקרי מאחורי ProGAN, שהוצע ב-2017 על ידי חוקרים מחברת Nvidia, הינו ליצור תמונות ברזולוציה הולכת וגדלה בצורה הדרגתית. כלומר, מקום לנסות לאמן את כל השכבות של ה- G -generator במת אחת, כפי שנעשה בכל ה-GANs לפניו, ניתן לאמן אותו ליצור תמונות ברזולוציה משתנה – בהתחלה הוא מתאים לייצר תמונות ברזולוציות מאוד נמוכה (4×4), לאחר מכן המשיכו ליצור תמונות ברזולוציה 8×8 , אחר כן 16×16 וכך הלאה עד יצירת תמונה ברזולוציה של 1024×1024 .



איור 7.11 ארכיטקטורת ProGAN.

כדי לאמן GAN לייצר תמונות בגודל 4×4 , התמונות מוסט האימון הוקטנו לגודל זה (down-sampling). אחרי שה-GAN לומד לייצר תמונות בגודל 4×4 , מושגים לו עוד שכבה המאפשרת להכפיל את גודל התמונות המיצירות, קרי לייצור תמונות בגודל 8×8 . יש לציין שהאימון של הרשת עם השכבה הנוספת מתחילה עם המשקלים שאומנו קודם לכן, אך לא "מקפאים" אותם, ככלומר הם מעודכנים גם כן תוך כדי אימון הרשת בשבייל לייצור תמונה ברוחולציה כפולה. הגדלה הדרגתית של הרוחולציה מאפשרת את הרשותה להתמקדש תחילה בפרטים ("הגים") של התמונה (דפוסים בתמונה מוטשטשת מאוד). לאחר מכן הרשת "לומדת" לבצע sampling (להכפיל את הרוחולציה) של התמונות המוטשטשות האלה. תהליך זה משפר את איכות התמונה הסופית כיון שבאופן זה הסבירות שהרשות תלמד דפוסים שגויים קטנה משמעותית.

7.2.7 StyleGAN

StyleGAN, שיצא בשלהי שנת 2018, מציע גרסה משודרגת של ProGAN generator. עם דגש על רשת-ה-*generator*. המאמר שמו לב כי היתרונו הפטונצייאלי של שכבות ProGAN המייצרות תמונה בצורה הדרגתית נובע מיכולתו לשולוט בתוכנות (מאפיינים) ויזואליות שונות של התמונה, אם משתמשים בהן כראוי. ככל שהשכבה והרוחולציה נਮוכה יותר, כך התוכנות שהיא משפיעה עליה גסות יותר.

למעשה,_styleGAN הינו GAN הראשון שנותן יכולת לשולוט במאפיינים ויזואליים (אומנם לא בצורה מלאה) של התמונה הנוצרת. מחברי StyleGAN חילקו את התוכנות הוויזואליות של תמונה ל-3 סוגים:

- **גס:** משפיע על תנוכה, סגןן שיער כללי, צורת פנים וכו'.
- **אמצעית:** משפיע על תוכו פנים עדים יותר, סגןן שיער, עיניים פקוחות/עצומות ועוד'.
- **רוחולציה דקה:** משפיעה על צבע (עיניים/שיער/עור) ועל שאר תכונות המיקרו של תמונה.

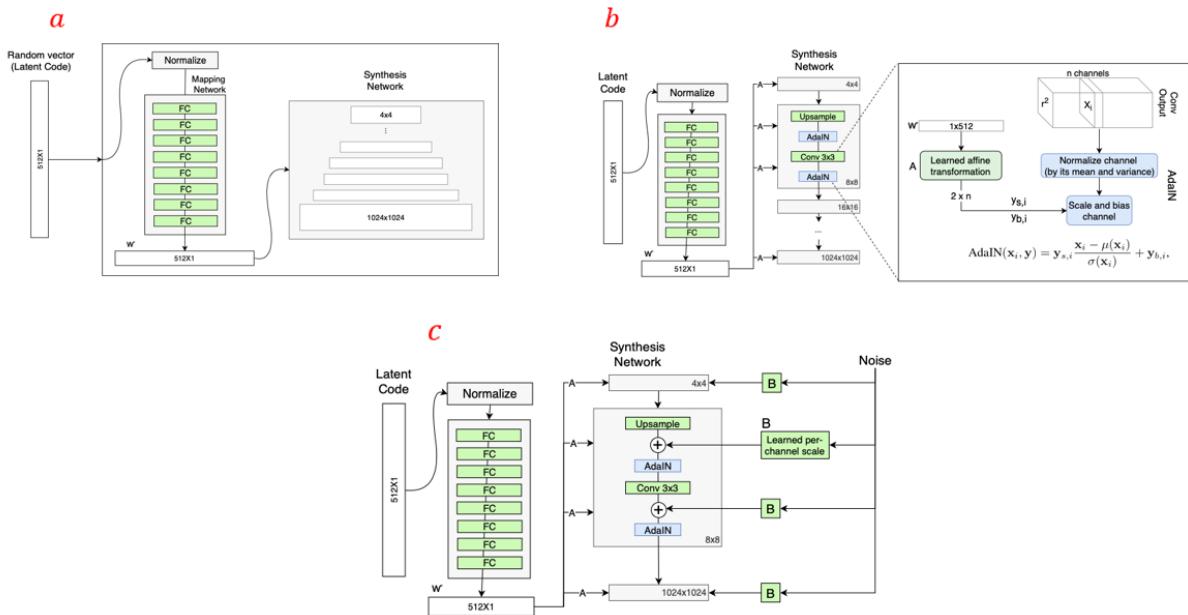
כדי להעניק ל-AN StyleGAN את היכולות האלו, נדרש מספר שינויים ביחס לארכיטקטורה של ProGAN (נתאר רק את שלושת השינויים החשובים ביותר כאן):

- **הוספת רשת מיפוי:** מטרת רשת המיפוי היא קידוד וקטור הקלט לווקטור ביןיהם w (הנקרא וקטור סגןון) אשר האיברים השונים שלו שולטים בתוכנות ויזואליות שונות של התמונה הנוצרת. זהו תהליך לא טריוויאלי מכיוון שהיכולת של הרשת לשולוט בתוכנות ויזואליות באמצעות וקטור הקלט הינה מוגבלת. הסיבה לכך טמונה בעובדה שווקטור הקלט נאלץ "לעקוב אחר צפיפות ההסתברות של סט האימון" שגורם לתופעה הנקראת (FE) feature entanglement (ערובם מאפיינים). FE בין תכונות צבע השיער והמגדר יכול להופיע אם למשל בסט האימון יש מגמה כללית של גברים עם שיער קצר ונשים בעלות שיער ארוך. במקרה זה הרשת תלמד שגברים יכולים להיות בעלי שיער קצר בלבד ולהיפך אצל נשים. כתוצאה לכך, אם "נשחק" עם רכיבי וקטור הקלט כדי לייצר תמונה של גבר בעל שיער ארוך, בסופו של דבר מגדרו ישתנה גם כן ונקבל תמונה של אישה.

רשת המיפוי שההווסף לארכיטקטורה הופכת את וקטור הקלט לווקטור ביןיהם w שאינו צריך לעקוב אחר התפלגות של סט האימון, וכך יש פחות ערבוב המאפיינים. במילים אחרות, רשת זו מאפשרת את היכולת לשולוט במאפיינים ויזואליים של התמונה הנוצרת באמצעות שני רכיביו של וקטור w . רשת המיפוי מורכבת משמונה שכבות FC וגודל הפלט שלו זהה לגודל הקלט.

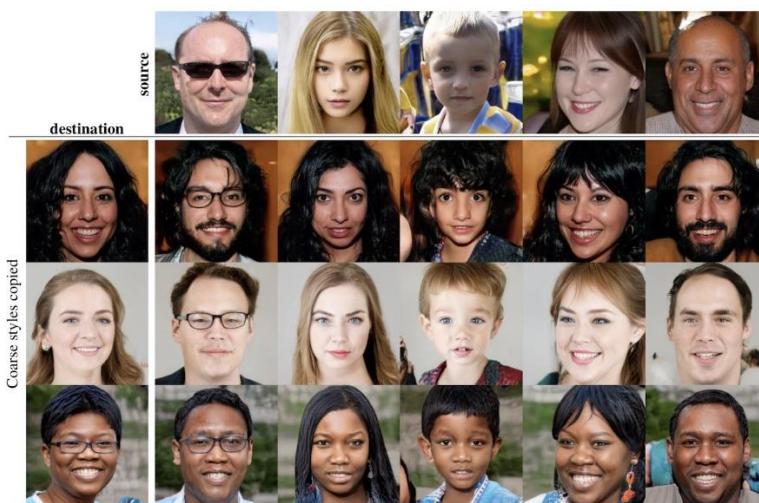
- **החלפת BN ב-AdaIN:** רשותת הקונבולוציה של ה-generator, שנוועדו לייצור תמונות ברוחולציות שונות משתמשות במנגנון שנקרא AdaIN (במקום BN). בשונה מ-BN, הפרמטרים של המוצע ושל השונות בגישת AdaIN נלמדים מוקטור הסגן w (המוצע טרנספורמציה לינארית של w עם משקלים נלמדים). להבדיל מ-AdaIN, במנגנון BN סטנדרטי פרמטרים אלו נלמדים כמו המשקלים האחרים ולא תלויים במצבם של שכבה כלשהי.

- **ויתור על אקריאות של קטורי קלט:** ב-StyleGAN וקטורי הקלט אינם וקטורי המוגREL מהתפלגות גausית אלא וקטורי דטרמיניסטיים רכיבים נלמדים. וקטורי הרעש מתווספים ישירות לפלאטים של ערוצי קונבולוציה ברשתות ה-generator כאשר העוצמה שלהם נŁmdת לכל ערוץ בנפרד. שימוש בוקטור קלט דטרמיניסטי במקומם בוקטור אקריאי מקל כל הנראה על הפרדת המאפיינים על ידי רשת המיפוי (ויתר קל לעשوت זאת על וקטורי קבוע מאשר להתאים את משקליו רשת המיפוי לווקטורי כניסה אקריאים).



איור 12. השינויים העיקריים בארכיטקטורת StyleGAN. (a) שימוש רשת מיפוי. (b) שימוש ב-AdaIN. (c) שימוש ב-ProGAN.

יש עוד כמה שינויים יותר מינוריים ב-StyleGAN ייחסית ל-ProGAN, כמו שינוי של היפר פרמטרים של הרשתות, פונקציית מחיר וכו'. התוצאות הן לא פחות ממרשימות – StyleGAN יוצר תמונות שנראות ממש אמיתיות ובונספ מקנה יכולת לשילוט בחלק מהתכונות החזויות של התמונה.



איור 13. תמונות שיוצרו באמצעות StyleGAN.

7.2.8 Wasserstein GAN

אחד סוגי GAN החשובים ביותר הינו Wasserstein GAN, והוא נוגע בבעיה שיש בפונקציית המחר בבה משתמשים הרבה וריאנטים של GAN-ים. כאמור, תהליך הלמידה של הרשת המיצרת דאטה – generator – generator – נעשה באמצעות משוב המתקבל מה-generator. בעוד שה-generator מאמין להבחן בין דאטה אמיתית לדאטה סינטטי הון discriminator מאמין לא מסתמן על דוגמאות אמיתיות אלא רק על המשוב מה-generator. משום כך, בתחלת הלמידה, כאשר generator-ו עוד לא מאמין, הדוגמאות הסינטטיות שהוא מייצר אין דומות כלל לדאטה האמיתית, וה-generator discriminator מבחין بكلות ביניהם. במקרים אחרים, בתחלת תהליך הלמידה generator-ו יוצר טוב יותר מאשר generator-ו העובר ל-"זען" העוברgenerator-ל-generator, כיון שהשיפור מtaboo על ה-generator. generator-ו, כדי להבין מדוע תהליכי הגרדיינט של פונקציית המחר (loss) שלו, התלו依 בערכיהם אותם מוצאים discriminator-ו. כדי להבין מדוע תהליכי הגרדיינט העברת המידיע באופן הזה בעייתי, יש להרחיב מעט על תהליכי היצירה של הדאטה על ידי generator-ו ורק generator-ו מסתכל על דאטה זו.

הנחה היסודית ברוב המודלים הגנרטיביים, ובפרט ב-GANs, הינה שהדאטה הרבה ממד (למשל תמונה) "חיה" במשטח מממד נמוך בתוכו. אפשר להסתכל על משטח בתווך הכללה של תחת-מרחב וקטורי מממד נמוך הנפרש על ידי תחת-קבוצה של וקטורי בסיס של מרחב וקטורי מממד גבוה יותר. גם המשטח נוצר מחת-קבוצה של וקטורי הבסיס של "מרחב האם", אך ההבדל בין תחת-מרחב וקטורי מתבטא בכך שלמשטח עשוי להיות צורה מאוד מרכיבתיחסית לתחת-מרחב וקטורי. משתמש מכך שניתן לייצר דאטה רב ממד על ידי טרנספורמציה של וקטור מרחב בעל מממד נמוך (וקטור לטנסי). למשל, ניתן בצערת רשת נירונים לייצר תמונה בגודל $k \times 3 > 64 \times 64$ פיקסלים מוקטור באורך 100 בלבד. זאת אומrette, שגם התפלגות התמונות של הרשת הגנרטיבית וגם התפלגות של הדאטה האמיתית נמצאים ב"משטח בעל ממד נמוך" ב远处 מרחב בעל ממד גבוה של הדאטה המקורי. באופן פורמלי יותר, משטח זה נקרא ירעה (manifold), וההשערה שתוארכה מעלה מהויה הנחתה יסוד בתחום הנקרא למידת ייעות manifold learning. מכיוון שמדובר במשטחים בעלי ממד נמוך בתוך מרחב בעל ממד גבוה, קיימת סבירות גבוהה שלא יהיה שום חיתוך בין המשטח בו "חיה" הדאטה האמיתית לבין זה של הדאטה הסינטטי (לכל הפחות בתחלת תהליכי האימון של ה-GAN), ויתרה מכך, המרחק בין משטחים אלה עשוי להיות די גדול. מכך נובע שה-generator D עשוי ללמוד להבחין בין הדאטה האמיתית לסינטטי לבסוף, כיון שבמרחב מממד גבוה יש מרחק גדול בין ירעה אמיתית לבין הירעה של הדאטה הסינטטי. בנוסף, D יתגאה יותר ממד טובי יחסית ל-G. אתגר זה בא לידי ביטוי גם בצורה של ממש קרובים לאפס כי אכן קל למצא "משטח הפרדה" בין שתי הירעות – זה של הדוגמאות האמיתיות והאלא הסינטטיות, כיון שהם נוטים להיות רחוקים מאוד מהשני.

רקע זה מסייע להבין מדוע הפער שיש בין ה-z-generator וה-generator discriminator מחייב מהויה בעיה. כאמור, ה-generator מעדכן את המשקלים שלו על סמך היצוניים שהוא מקבל מה-generator discriminator (דרך פונקציית המחר של ב-GAN). אבל אם ה-generator discriminator כל הזמן מוציא ציונים מאד נמוכים (עקב מרחק גדול בין הירעות שתואר מעלה) לדוגמאות המיצרות על ידי ה-generator, generator-ו פשטוט לא יכול לשפר את איכות התמונות שהוא מייצר. במקרה פשוט, D "פשטוט הרבה יותר מידי טובי יחסית ל-G". אתגר זה בא לידי ביטוי גם בצורה של פונקציית המחר, שלא מאפשרת "העברה עילאה של ידע" מה-generator discriminator-ל-generator.

יש מספר לא קטן של שיטות הבאות לשפר את תהליכי האימון של GAN, אך אף אחת מהן אינה מטפלת בבעיה זו באמצעות שינוי של פונקציית המחר. השיטות הבולטות הן:

- [התאמת פיצרים \(feature matching\)](#)
- [minibatch discrimination](#)
- [virtual batch normalization](#)
- [מציע היסטורי](#)

כפי שהסביר, הבעיה של המרחק בין הירעות מתקפת במבנה של פונקציית המחר, וכיון שכך, ניתן לנוטות ולפתרן את הבעיה מהשורש על ידי שימוש בפונקציית מחר יותר מתאימה. לשם כך ראשית נסמן את התפלגות הדאטה האמיתית p_{real} , ואת התפלגות הדאטה הסינטטי המיצר על ידי ה-generator p_{fake} . לעיל הראינו שפונקציית המחר האופטימלית המגדעת את המרחק בין התפלגות D_{JS} – Jensen-Shannon divergence על ידי $D_{JS} = \frac{1}{2} \text{KL}(p_{\text{real}} || p_{\text{fake}}) + \frac{1}{2} \text{KL}(p_{\text{fake}} || p_{\text{real}})$.

ניתן להוכיח כי מרחק D_{JS} בין התפלגות p_{real} ו- p_{fake} לא ניתן לשינויים ב- p_{fake} כאשר המשטחים שבהם "חינם" r ו- $-r$ רוחקים אחד מהשני. לעומת זאת, מרחק D_{JS} כמעט ולא ישנה אחריו עדכון המשקלים של ה-generator, וממילא לא ישקף את המרחק המעודכן בין שתי התפלגות p_{real} ו- p_{fake} . זו למעשה הבעיה המהותית ביותר עם פונקציית המחר המקורית של ה-generator, שעדכון המשקלים לא משפיע כמעט על D_{JS} , וכיון שטראש התפלגות r ו- $-r$ רוחקות אחת מהשנייה.

ב� להתמודד עם בעיה זו, ולשם כך הוא משתמש בפונקציית מחיר אחרת, בה עדכון המשקלים generator בין התפלגיות p ו- q . פונקציית המחיר החדשה מבוססת על מרחק הנקרא (EM) (Earth Mover), המהווה מקרה פרטי של מרחק ורסטיין המשומן ב- \mathcal{W}_p . מרחק ורסטיין מסדר 1 $\geq p \geq 1$ בין שתי מידות הסתברות μ ו- ν , על מרחב M מוגדר באופן הבא:

$$W_p(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|] = \left(\inf_{\gamma \in \Gamma(\mu, \nu)} \int_{M \times M} d(x, y)^p d\gamma(x, y) \right)^{\frac{1}{p}}$$

כאשר (μ, ν) הן כל מידות הסתברות על מרחב המכפלה (product space) של M עם עצמו (זהו למעשה מרחב המכיל את כל הזוגות האפשריים של האלמנטים M -ים). עם פונקציות שליליות (marginal) השווות ל- μ ו- ν בהתאם ל- μ ו- ν , ובאופן מפורש: כאשר $1 = k$,

$$EM = W_1(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \int_{M \times M} d(x, y) d\gamma(x, y)$$

הגדרה זו נראהית מאוד מוסובכת וננסת לתת עברורה אינטואיציה, ולהבין מדוע עבור $1 = k$, מרחק ורסטיין נקרא EM. לשם הפשטות נניח שהמרחב M הינו חד ממדי, כלומר קוו ישר, ועליו عشر משקלות של 0.1_{kg} כל אחת המפוזרות באופן הבא: 6 משקלות 0.6_{kg} בנקודת $x = 0$, ו-4 משקלות 0.4_{kg} בנקודת $x = 1$. כעת נרצה להזיז את המשקלות כך שתהיינה מפוזרות באופן הבא: בנקודת $x = 4$ יהיה משקל של 0.3_{kg} , בנקודת $x = 5$ יהיה משקל של 0.3_{kg} , ושאר המשקלות 0.2_{kg} בנקודת $x = 8$.

כמובן שיש הרבה דרכים לבצע את היזזת המשקלות, ונרצה למצאו את הדרך היעילה ביותר. לשם כך נגדיר באמצעות מכפלה של משקל במרחב אותו מודים את המשקל (בפיזיקה מושג זה נקרא עבודה - כוח המופעל על גופו לאורכו מסלול). בדוגמה המובאת, המאמץ המינימלי מתקיים על ידי היזזת המשקלות באופן הבא:

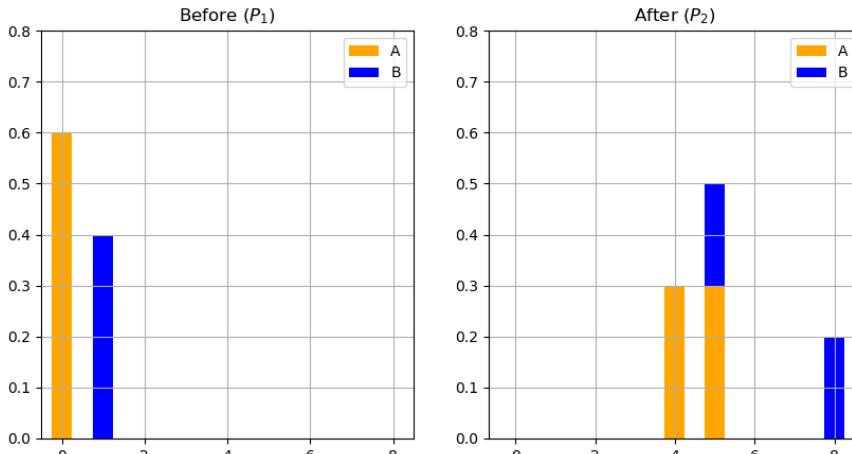
$$\text{מעברים } m=0 \text{ : } x = 4 - 0 = 4, \text{ כאשר המאמץ הנדרש לכך הינו } 0.3_{kg}.$$

$$\text{מעברים } m=0 \text{ : } x = 5 - 0 = 5, \text{ כאשר המאמץ הנדרש לכך הינו } 0.3_{kg}.$$

$$\text{מעברים } m=1 \text{ : } x = 5 - 1 = 4, \text{ כאשר המאמץ הנדרש לכך הינו } 0.2_{kg}.$$

$$\text{מעברים } m=1 \text{ : } x = 8 - 1 = 7, \text{ כאשר המאמץ הנדרש לכך הינו } 0.2_{kg}.$$

$$\text{סך המאמץ המינימלי שווה במקרה זה ל: } 1.2 + 1.5 + 0.8 + 1.4 = 4.9.$$



איור 7.14 העברת משקלות באופן אופטימלי. P_1 מייצג את המצב ההתחלתי, ו- P_2 הינו המצב לאחר היזזת המשקלות.

כעת, במקום להסתכל על משקלים, נתיחס לתפלגיות p_1, p_2 , המוגדרות באופן הבא:

$$p_1(x) = \begin{cases} 0.6, & x = 0 \\ 0.4, & x = 1 \\ 0, & \text{else} \end{cases} \quad p_2(x) = \begin{cases} 0.3, & x = 4 \\ 0.5, & x = 5 \\ 0.2, & x = 8 \\ 0, & \text{else} \end{cases}$$

השאלה כיצד ניתן להעביר מסה הסתברותית מ- p כרך שתתקבל ההתפלגות \mathcal{D}_2 , שקופה לדוגמא של הזרת המשקלות. מרחק EM בין שתי התפלגויות p_1, p_2 , או מוגדר להיות ה"אמץ" המינימלי הנדרש בשבייל להעביר את המסנה הסתברותית מ- p_1 ל- p_2 , או במילוי אחריות – מרחק EM מוגדר מהי כמות ה"עבודה" (מאםץ) המינימלית הנדרשת בשבייל להפוך p_1 ל- p_2 . אם נחזור לדוגמא של המשקלות, נוכל להבין מדוע D_w עבור $1 = k$ נקרא מרחק Earth Mover – מרחק בין שתי התפלגויות שקול לכמה מאםץ נדרש להעביר כמות אדמה במשקל מסוים כדי לעבור מחלקה מסוימת של אדמה לחולקה אחרת. באופן יותר פורמלי – מידת הסתברות על מרחב המכפלה בנוסחה של מרחק EM מוגדרת את האופן שבו אנחנו מעבירים את המסנה הסתברותית (משקל מסוים של אדמה), כאשר הביטוי (y, x) מציין כמה מסה הסתברותית מועברת מנוקודה x לנוקודה y .

לאחר שהסבירנו מהו מרחק ורשותן D_w ומהו מרחק EM, ניתן להבין כיצד אפשר להשתמש במסוגים אלו עבור פונקציית מחיר של GAN. נזכיר כי מרחק D_w בין מידות הסתברות מתחשב בתכונות של הקבוצות עליהם מידות אלו מוגדרות בצורה מפורשת, על ידי המתחשב במרקםם בין קבוצות אלו. תכונה זו היא למעשה בדיקת מה שציבור בשבייל למדוד את המרחק בין התפלגות האמיתית של דאטה r לבין התפלגות של הדאטה הסינטטי g . מרחק EM ייעד לשערך בצורה טוביה את המרחק בין היריעות שבנה "חוות" שתי התפלגויות, ככלומר אם מזידם את היריעות של הדאטה הסינטטי, נוכל לדעת בערך מרחק EM עד כמה השתנה המרחק בין היריעות. נזכיר שזה לא קורה כאשר משתמשים בפונקציית המחיר המקורי הנמדדת באמצעות J_S . כתוב, בעזרת פונקציית המחיר החדשה המבוססת על מרחק EM, ניתן לדעת עד כמה עדכון המשקלים מקרוב או מרחק את g מ- r .

באופן תיאורתי זה מצוין, אך עדין זה לא מספיק, כיון שצריך למצוא דרך לחשב את D_w , או לכל הפחות את המקרה הפרטני שלו עבור $1 = r$, ככלומר את מרחק EM. במקור מרחק זה מוגדר כבעית אופטימיזציה של מידות הסתברות על מרחב המכפלה, וצריך למצוא דרך להשתמש בו כפונקציית מחיר. בשבייל לבצע זאת, ניתן להשתמש בצורה דואלית של D_w עבור $1 = r$ – שיווין RK (Rubinstein-Kantorovich) לפוי נិતן לחשב את $1 = r$ – שיווין D_w באופן הבא:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L} E_{x \sim p_r}[f(x)] - E_{x \sim p_g}[f(x)]$$

כאשר (x, f) הינה **פונקציית K-לייפשיץ רציפה** (כלומר, פונקציה רציפה עם קצב השתנות החסום על ידי K). כתוב נניח ש- (x, f) הינה פונקציית K-לייפשיץ רציפה המתארת discriminator בעל סט הפרמטרים w . ה- z -generator מחשב באופן הקרוב את המרחק בין התפלגויות באופן הבא:

$$L(p(r), p(g)) = W(p(r), p(g)) = \max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))]$$

פונקציית מחיר זו מודדת את המרחק D_w בין התפלגויות p_r, p_g , וככל שפונקציה זו מקבל ערכים יותר נמוכים כהה-generator יצליח ליצור דוגמאות שמתפלגות באופן יותר דומה לדאטה המקורי. בשונה מ-GAN קלאסי בו ה-discriminator מוציא הסתברות עד כמה הדוגמא אותה הוא מקבל אמיתית, פה ה- z -generator לא מאמין להבחין בין דוגמא אמיתית לסינטטית, אלא מאמין ללמידה פונקציית K-לייפשיץ רציפה המודדת את D_w בין התפלגויות p_r, p_g . ה- z -generator לועמת זאת מאמין ללמידה $L(p_g, p_r)$ (כאשר רק האיבר השני שתלי ב- g_θ), וככל שפונקציית המחיר הולכת וקטנה, כך g מתקרב יותר ל- r .

כאמור, תנאי הכרחי לשימוש במרקם זה בפונקציית המחיר הינו שהפונקציה תהיה K-לייפשיץ רציפה. מסתבר שקיים תנאי זה אינו ממשימה קלה כלל. כדי להבטיח את קיומו, המאמר המקורי הציג לבצע קטימה של משקלי ה-discriminator לטווח סופי מסוים, נניח $[0.01, 0.01]$. ניתן להראות כי קטימה זו מבטיחה את ש- w f_w K-לייפשיץ רציפה. אולם, כמו שכותבי המאמר מודים בעצמם, ביצוע קטימה בכדי לדאוג לפחות לקיום K-לייפשיץ יכול לגזור לביעות אחרות. למעשה, כאשר חלון הקטימה של המשקלים צר מדי, הגדריאנטים של Wasserstein GAN עלולים להתאפס, מה שיאט את תהליכי הלמידה. מצד שני, כאשר חלון זה רחב מדי, התכנסות עלולה להיות מאוד איטית. נזכיר שיש עוד מספר דרכי לכפות על w להיות K-לייפשיץ-רציפה למשל gradient penalty.

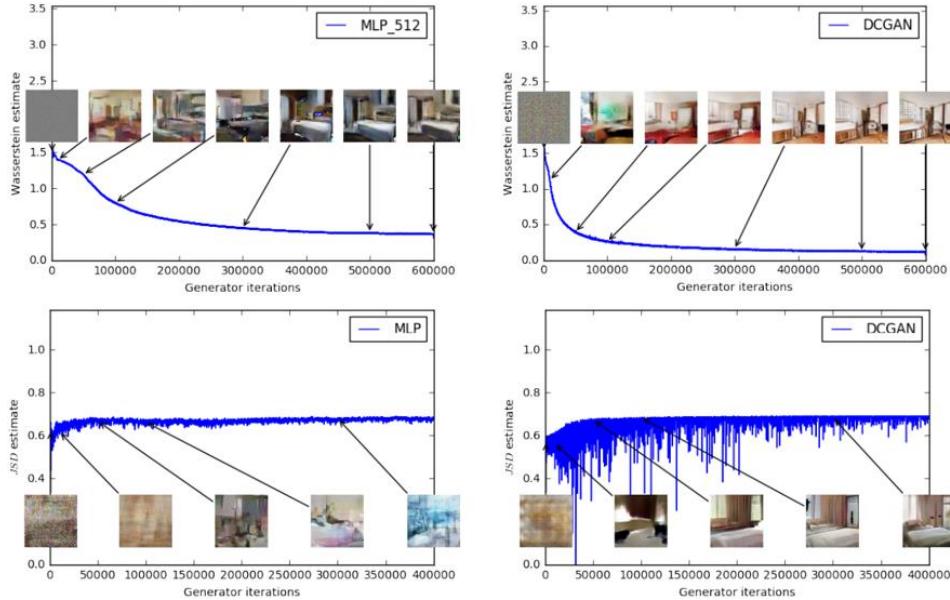
הإيمان של Wasserstein GAN דומה לאימון של ה-GAN המקורי, למעט שני הבדלים עיקריים:

א. קיצוץ טווח המשקלים על מנת לשמור על רציפות-לייפשיץ.

ב. פונקציית מחיר המסתמכת על D_w במקומם על J_S .

תהליכי הלמידה מתבצע באופן הבא – לאחר כל עדכון משקלים של ה- z -generator (gradient ascent) ובאמצעות discriminator (gradient descent) מצלח לגזור לכך שהקורסיצה בין איות התמונה הנוצרת על ידי ה- z -generator לבין ערך של מנגנונים את טווח המשקלים. לאחר מכן מבצעים עדכון רגיל של משקלי ה- z -generator תוך שימוש של איטרצייה שלgradient descent.

Wasserstein GAN מצליח לגזור לכך שהקורסיצה בין איות התמונה הנוצרת על ידי ה- z -generator לבין ערך של פונקציית לוס תהיה הרבה יותר בולטת מאשר ב-GAN רגיל בעל אותה ארכיטקטורה. ניתן להמחיש זאת היטב באמצעות גרפים הבוחנים את היחס בין D_w לבין D_{JS} .



איור 7.15 שערור מרחק W בין \mathcal{D}_g ל- \mathcal{D}_g כפונקציה של מספר האיטרציות (בגרפים העליונים), לעומת שערור מרחק JS בין \mathcal{D}_g ל- \mathcal{D}_g כפונקציה של מספר האיטרציות (בגרפים התחתונים).

ניתן לראות בבירור כי ככל ש懿יות התמונהות שה-generator מיציר עולה, כך \mathcal{D} הולך וקטן, ואילו מרחק JS לא מראה שום סימן של ירידה. הצלחה זו נובעת מהשינוי בפונקציית המחזר, שגרם לאימון להיות יותר יעיל, והביא לכך שהדוגמאות הסינטטיות תהינה דומות הרבה יותר לדאטה המקורי.

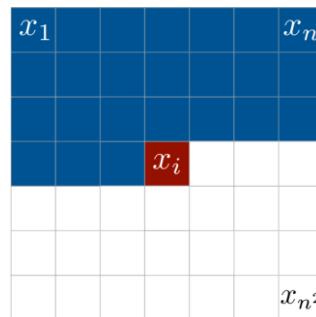
נוקודה נוספת שיכולה להסביר את ההצלחה הייחסית של השימוש ב- W נובעת מכך שמטריקת JS חלשה יחסית למטריקת JS , וננסה להבהיר נוקודה זו.

באופן אידיאלי, כאשר אנו מאמנים מודל הינו רצים להיות בטוחים שאם ננגב בצורה נאותה ובכל צעד נעדן המודל לבדוק על פי הוראות הגרדיינט, נסימם את האימון בנקודה בנקודה כמעט ללא אופטימיליזציה. אולם בפועל זה לא תמיד כך, כיון שישנן בעיות שעבורן מטריקות מסוימות יגיעו לנקודת קצה זרואה ולא. ניקח לדוגמא שני אנשים שעומדים על סף תחום ורוצחים להגיע עמוק. האחד מodd את הגובה ומתקדם על פי, ולכן הוא יגיע למיטה בקלות ייחסית. לאחר מתענין במיקומו על ציר צפוני דרום, ולכן הוא עשוי להגיע בקצבים ממהלך הירידה, וגם אם הוא אכן יגיע למיטה, זה בהכרח יהיה בתהליך איטי יותר. באופן דומה, כאשר לוחכים זוג מטריקות, באופן פורמלי ניתן להגיד שאם התכונות של סדרת התפלגיות תחת מטריקה אחת גוררת התכונות של הסדרה תחת מטריקה אחרת, אז המטריקה הראשונה חזקה יותר ממתירה השנייה. העובדה ש- W חלש יותר מ- JS בעצם אומerta שיתכן ויש בעיות שעבורן תתקבל תוצאה אופטימלית עבור W אך לא עבור JS .

7.3 Auto-Regressive Generative Models

משפחה נוספת של מודלים גנרטיביים נקראת Auto-Regressive Generative Models, VAE, ובודומה לו גם מודלים אלו מוצאים התפלגות מפורשת של מרחב מסוים ובעזרת התפלגות זו מייצרים דאטה חדש. עם זאת, בעוד VAE מוצא קירוב להתפלגות של המרחב הלטנטי, שיטות AR מנוטות לחשב במדויק התפלגות מסוימת, וממנה לדגום וליצור דאטה חדש.

תמונה $a \times a$ היא למעשה רצף של a^2 פיקסלים. כאשר רוצים ליצור תמונה, ניתן ליצור כל פעם כל פיקסל בהתאם לזה שהוא יהיה תלוי בכל הפיקסלים שלפניו.



איור 15.7 תמונה כרצף של פיקסלים.

כל פיקסל הוא בעל התפלגות מותנית:

$$p(x_i | x_1 \dots x_{i-1})$$

כאשר כל פיקסל מורכב משלושה צבעים (RGB), لكن ההסתברות המדויקת היא:

$$p(x_{i,R} | x_{<i}) p(x_{i,G} | x_{<i}, x_{i,R}) p(x_{i,B} | x_{<i}, x_{i,R}, x_{i,G})$$

כל התמונה השלמה היא מכפלת ההסתברויות המותניות:

$$p(x) = \prod_{i=1}^{n^2} p(x_i) = \prod_{i=1}^{n^2} p(x_i | x_1 \dots x_{i-1})$$

הביטוי (x) הוא ההסתברות של DATA מסוים לייצג תמונה אמיתית, אך נרצה למקסם את הביטוי הזה כדי לקבל מודל שמייצג תמונות שנראות אוטנטיות עד כמה שניתן.

7.3.1 PixelRNN

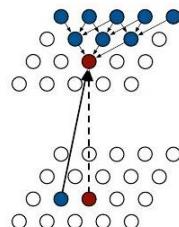
אפשרות אחת לחשב את (x) היא להשתמש ברכיבי זיכרון כמו LSTM עבור כל פיקסל. באופן טבעי הינו רצימם הקשור כל פיקסל לשכנים שלו:

$$\text{Hidden State } (i, j) = f(\text{Hidden State } (i-1, j), \text{Hidden State } (i, j-1))$$

הבעיה בחישוב זה היא הזמן שהוקח לבצע אותו. כיוון שכל פיקסל דרוש לדעת את הפיקסל שלפניו – לא ניתן לבצע אימון מקבילי לרכיבי-h-LSTM. כדי להתגבר על בעיה זו הוצעו כמה שיטות שונות לאפשר חישוב מקבילי.

Row LSTM

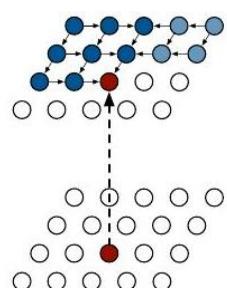
במקרה להשתמש במצב החבוי של הפיקסל הקודם, ניתן להשתמש רק בשורה שמעל הפיקסל אותו רצימם לחשב. שורה זו בעצם מחושבת לפני כן על ידי השורה שמעליה, ובכך למעשה לכל פיקסל יש receptive field של משולש. בשיטה זו ניתן לחשב באופן מקבילי כל שורה בנפרד, אך יש לכך מחיר של איבוד הקשר בין פיקסלים באותה שורה (loss context).



איור 16 Row LSTM 7.16 – כל פיקסל מחושב על ידי $\geq k$ פיקסלים בשורה שמעליה.

Diagonal BiLSTM

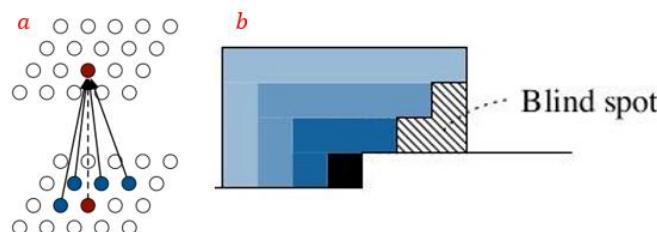
כדי לאפשר גם חישוב מקבילי וגם שמירה על קשר עם כל הפיקסלים, ניתן להשתמש ברכיבי זיכרון דו כיווניים. בכל שלב מחשבים את רכיבי הזיכרון משני הצדדים, וכך כל פיקסל מחושב גם באמצעות הפיקסל שלידו וגם על ידי זה שמעליו. באופן זה receptive field גודל יותר ואין loss context, אך החישוב יותר איטי מהשיטה הקודמת, כיוון שהשורות לא מחושבות בפעם אחת אלא כל פעם שני פיקסלים.



איור 7.17 Diagonal BLSTM – כל פיקסל מחושב על ידי $3 \geq k$ פיקסלים בשורה שמעלינו. כדי לשפר את השיטות המשתמשות ברכיבי זיכרון ניתן להוסיף עוד שכבות, כמו למשל Residual blocks שיעזרים להאיץ את ההתכנסות ו-*Masked convolutions* כדי להפריד את התלות של הערכים השונים של כל פיקסל.

7.3.2 PixelCNN

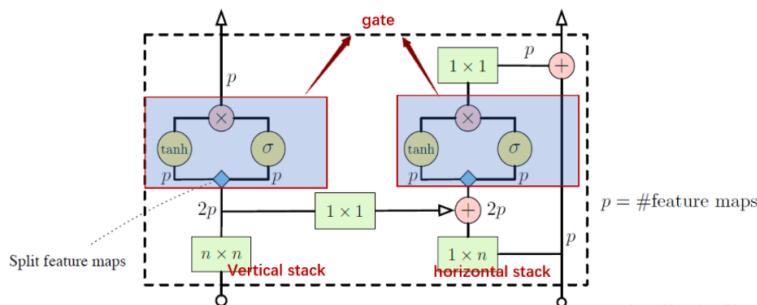
הчисIRON העיקרי של PixelRNN נובע מהAIMON האיטי שלו. במקום רכיבי זיכרון ניתן להשתמש ברשת קונבולוציה, ובכך להאיץ את תהליכי הלמידה ולהגדיל את *the receptive field*. גם בשיטה זו מתחילה מפהיקסל הפינתי, רק כעת הלמידה היא לא בעדרת רכיבי זיכרון אלא באמצעות שכבות קונבולוציה. היתרון של שיטה זו על פני PixelRNN מתקבל בקיצור משמעותי של תהליכי האימון, אך התוצאות פחות טובות. חיסרון נוסף בשיטה זו נובע מהמבנה של המנסנים ו-*the receptive field* – כל פיקסל מtabssס על שלושה פיקסלים שמעליו, והם בתורם כל אחד תלוי בשלושה פיקסלים בשורה שמעל. מבנה זה מנתק את התלות בין פיקסלים קרובים יחסית אך אינם ב-*receptive field*.



איור 7.18 החיסרון של receptive field (a) PixelCNN. (b) החיסרון של receptive field – ניתוק בין פיקסלים יחסית קרובים.

7.3.3 Gated PixelCNN

בכדי להתגבר על בעיות אלו – ביצועים לא מספיק טובים והתעלמות מפיקסלים יחסית קרובים שאינם ב-*receptive field* – נעשה שימוש ברכיב זיכרון הדומה ל-LSTM, המשלב את רשותות הקונבולוציה בתוך RNN.



איור 7.19 שכבה של Gated PixelCNN.

כל רכיב זיכרון בני משני חלקים – horizontal stack and vertical stack – כאשר כל אחד מהם הוא למעשה שכבת קונבולוציה. ה-*vertical stack* בנייתו מזכיר של כל השורות שבו עד כה בתמונה, וה-*horizontal stack* הוא מסנן יחיד על הקטל הנוכחי. ה-*horizontal stack* עובר דרך שער של אקטיבציות לא לינאריות ובנוסף מתחבר ל-*vertical stack*, כאשר גם החיבור ביניהם עובר דרך שער של אקטיבציות לא לינאריות. פנוי כל כניסה של stack לתוך שער, המנסנים מטאפים – חצי עוביים דרך \tanh וחצי דרך סיגמאoid. בסך הכל המוצא של כל שער הינו:

$$y = \tanh(w_f * x) \odot \sigma(w_g * x)$$

7.3.4 PixelCNN++

שיפור אחר של PixelCNN הוצע על ידי OpenAI, והוא מבוסס על מספר מודיפיקציות:

- שכבה ה-*SoftMax* שקובעת את צבע הפיקסל צורכת הרבה זיכרון, כיוון שיש הרבה צבעים אפשריים. בנוסף, היא גורמת לגרדיינט להתפס מהר. כדי להתגבר על כך ניתן לבצע דיסקרטיזציה לצבעים, ולאפשר טווח צבעים קטן יותר. באופן זה קל יותר לקבוע את ערכו של כל פיקסל, ובנוסף תהליכי האימון יותר יעיל.
- במקרה לביצוע בכל פיקסל את ההתניתה על כל צבע בנפרד (כפי שהראינו בפתחה), ניתן לבצע את ההתניתה על כל הצבעים יחד.
- אחד האתגרים של PixelCNN הוא יכולת המוגבלת למצוא תלויות בין פיקסלים רחוקים. כדי להתגבר על כך ניתן לבצע *sampling down*, ובכך להפחית את מספר הפיקסלים בכל מסנן, מה שמאפשר לשמור את הקשרים בין פיקסלים בשורות רחוקות.

- בדומה ל-Net-U, ניתן לבצע חיבורים בעזרת Residual blocks ולשמור על יציבות במהלך הלמידה.
- שימוש ב-Dropout לצורף רגולרייזציה והימנעות מ-fitting.

7. References

VAE:

<https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>

<https://jaan.io/what-is-variational-autoencoder-vae-tutorial/>

<https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html>

GANs:

<https://arxiv.org/abs/1406.2661>

<https://arxiv.org/pdf/1511.06434.pdf>

<https://phillipi.github.io/pix2pix/>

<https://junyanz.github.io/CycleGAN/>

<https://arxiv.org/abs/1710.10196>

<https://arxiv.org/abs/1812.04948>

<https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431>

<https://arxiv.org/abs/1701.07875>

AR models:

<https://arxiv.org/abs/1601.06759>

<https://arxiv.org/abs/1606.05328>

<https://arxiv.org/pdf/1701.05517.pdf>

<https://towardsdatascience.com/auto-regressive-generative-models-pixelrnn-pixelcnn-32d192911173>

https://wiki.math.uwaterloo.ca/statwiki/index.php?title=STAT946F17/Conditional_Image_Generation_with_PixelCNN_Decoders#Gated_PixelCNN

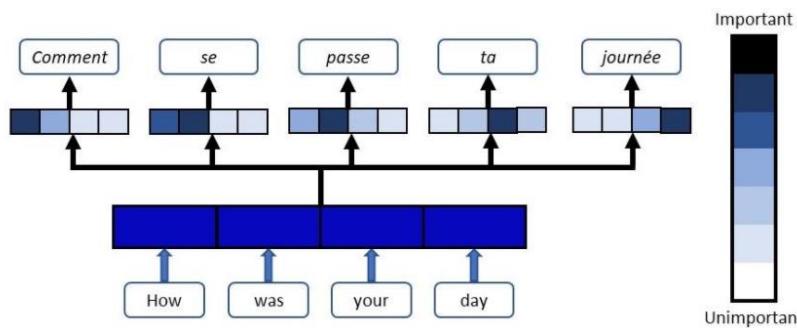
8. Attention Mechanism

8.1 Sequence to Sequence Learning and Attention

8.1.1 Attention in Seq2Seq Models

ניתוח סדרות בהן יש קשר בין האיברים יכול להיעשות בעזרת רשות עם רכיבי זיכרון, כפי שהואר בפרק 6. ברשותת אלו הסדרה הנכנת לרשף עוברת דרך יוצר וקטור בגודל ידוע מושך המציג את הסדרה המקורי, תוך התחשבות בסדר של איברי הסדרה ובקשר ביניהם. לאחר מכן וקטור זה עובר ב-decoder שיכל לפענן את המידע שיש בווקטור ולהציג אותו בצורה אחרת. למשל בתרגום משפה לשפה – מודל של seq2seq מקודד משפט בשפה אחת לווקטור מסוים ולאחר מכן מפענן את הווקטור לשפה השנייה.

הדרך המקובלת ליצור את הווקטור ולפענן אותו הייתה שימוש בארכיטקטורות שונות של RNN, כמו למשל רשת عمוקה מסוג LSTM או GRU המכילה רכיבי זיכרון. מודלים אלו נתקלו בבעיה בסדרות ארוכות, כיון שהווקטור מוגבל ביכולת שלו להכיל קשרים בין מספר רב של איברים. כדי להתמודד עם בעיה זו ניתן לנוקוט בגישה שונה – במקומם ליצור וקטור בموقع encoder, ניתן להשתמש במצבים החכبيים של ה-decoder, וכך למצוא תלויות בין איברי סדרת הקלט לאיברי סדרת הפלט (general attention) וקשרים בין איברי סדרת הפלט עצם (self-attention). ניקח לדוגמה תרגום של המשפט "How was your day" מאנגלית לשפה אחרת – במקרה זה מנגנון attention מייצר וקטור חדש עבור כל מילה בסדרת הפלט, אשר כל רכיב בווקטור מכמת עד כמה המילה הנוכחית בموقع קשורה לכל אחת מהמלילים במשפט המקורי. באופן זה כל איבר בסדרת הפלט משקל כל אחד מאיברי סדרת הפלט. מנגנון זה נקרא attention.



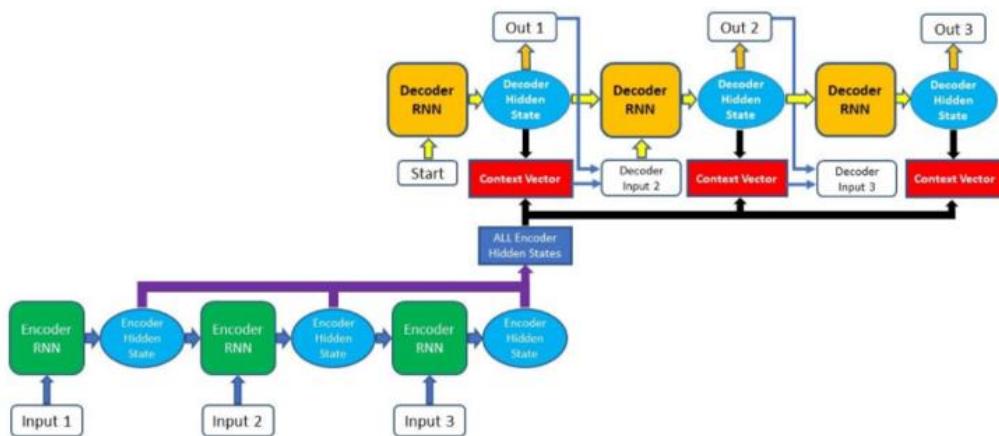
איור 8.1 מנגנון attention – נתינת משקל לכל אחת ממלילות הקלט בהתאם לכל אחת ממלילות הפלט.

במאמר משנת 2017 שנקרא "Attention is All You Need", הוצע להשתמש ב-attention בלבד ללא רשות מסוג LSTM או GRU, ומאמר זה פרץ דרך לשימושים רבים במנגן זה תוך קבלת ביצועים מעולים.

בחלק זה יוצגו הגישות המשלבות בין רשות NN לביון attention ולאחר מכן יוסבר על ה-transformer שמשתמש ב-attention ועוד דרך self-attention הרגיל, ובכך מיותר את הצורך ברכיבי זיכרון.

8.1.2 Bahdanau Attention and Luong Attention

הגישה הראשונה שהוצעה נקראה Bahdanau attention על שם הממציא שלה – Dzmitry Bahdanau.



איור 8.2 ארכיטקטורת Bahdanau Attention

הרעין של גישה זו היא לבנות ארכיטקטורה בה משתמשים בכל הממצבים החבויים של רכיבי הזיכרון ב-encoder ומעבירים אותם ל-decoder. כתוצאה לכך ה-decoder מחשב את המוצא לא רק על סמך מצביו הקודמים, אלא משקלל אותם יחד עם הממצבים החבויים של ה-encoder. עבור כל אחד מאיברי סדרת הפלט מחשבים alignment score בין המצב החבוי של רכיב הזיכרון הקודם בסדרת הפלט לבין כל הממצבים החבויים של ה-encoder, וכך יוצרים context vector שבעזרתו מחשבים את הפלט עבור האיבר הנוכחי ב-decoder. ביצוע הפעולה זו הוא הלב של מנגנון ה-attention, כיוון שהוא קשור בין הפלט לפלאט, ובנוסף מחשב עבור כל איבר של סדרת הפלט כמה משקל יש לתת לכל אחד מאיברי הפלט האחרים.

ביצוע פעולה זו יוצרת לכל אחד מאיברי הפלט context vector ייחודי משלו הנבנה גם מה המצב הקודם וגם מאיברי ה-encoder, בשונה מהארQUITקטורות הקודומות של seq2seq בהן לא היה ניתן להעיבר מידע באופן ישיר מהמצבים החבויים של ה-encoder ל-decoder. את ה-attention vector מחברים למוצא של האיבר הקודם ב-encoder, ויחד עם המצב החבוי הקודם יוצרים את המצב החבוי הבא, שבעזרתו מוצאים את הפלט של האיבר הנוכחי.

באופן פורמלי, אם נסמן ב- H_d , H_e את הממצבים החבויים של ה-encoder וה-decoder, יתקבל:

$$\text{alignment score} = w_{\text{alignment}} \times \tanh(w_d H_d + w_e H_e)$$

כאשר w_d , w_e הם המשקלים הנלמדים של ה-encoder, ה-decoder והחיבור ביניהם. את התוצאה העבירים דרך SoftMax ב- H_e , מכפילים ב- H_d ומתקבלים את ה-

$$\text{context vector} = H_e \times \text{SoftMax}(\text{alignment score})$$

הווקטור המתתקבל מכיל משקל של כל אחד מאיברי הפלט בהתאם לאיבר הפלט הנוכחי. את התוצאה כאמור מחברים לפלאט של האיבר הקודם, ובעזרת המצב החבוי הקודם מחשבים את המצב החבוי הנוכחי, שמננו מחלצים את הפלט של האיבר הנוכחי.

ישנו שיפור של Bahdanau attention הנקרא Loung attention. שני הבדלים העיקריים בין שני המנגנונים: חישוב alignment score מתבצע באופן שונה, ובנוסף בכל שלב לא משתמשים במצב החבוי הקודם אלא alignment score. שהוא אלא יוצרים מצב חבוי חדש ובעזרתו מחשבים את ה-

8.2 Transformer

לאחר שמנגנון ה-attention התחיל לצבור תאוצה, הומצאה ארכיטקטורת המבוססת על attention בלבד ללא שום רכיבי זיכרון. ארכיטקטורה זו הנקראת transformer מציעה שני אלמנטים חדשים על מנת למצוא קשרים בין איברים בסדרה מסוימת – self-attention – positional encoding.

8.2.1 Positional Encoding

ארQUITקטורות מבוססות RNN משתמשות ברכיבי זיכרון לצורך לקחת בחשבון הסדר של האיברים בסדרה. גישה אחרת לייצוג הסדר בין איברי הסדרה נקראת positional encoding, בה מושגים לכל אחד מאיברי הפלט פיסות מידע לגבי המיקום שלו בסדרה, והוספה זו כאמור באה כתחליף לרכיבי הזיכרון ברשותת RNN. באופן פורמלי, עבור סדרת קלט $\mathbb{R}^d \in \alpha$, מחשבים וקטור מממד $1 \times d$ באופן הבא:

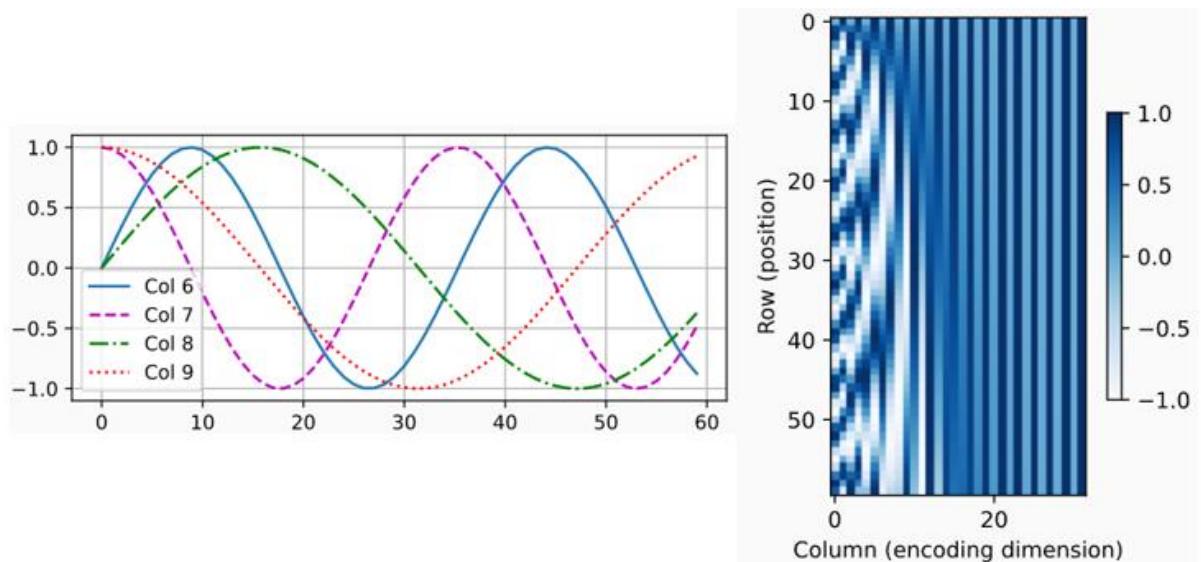
$$p_t(i) = \begin{cases} \sin(\omega_k t), & i \text{ is even} \\ \cos(\omega_k t), & i \text{ is odd} \end{cases}, \omega_k = \frac{1}{10000^{\frac{2k}{d}}} \rightarrow p_t = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \vdots \\ \sin\left(\omega_{\frac{d}{2}} t\right) \\ \cos\left(\omega_{\frac{d}{2}} t\right) \end{bmatrix}_{d \times 1}$$

בכדי להבין כיצד וקטור זה מכיל מידע של סדר בין דברים, נציג את הרעיון שהוא מייצג בצורה יותר פשוטה. אם נרצה לקחת רצף של מספרים וליצג אותם בצורה בינארית, נוכל לראות שככל שהיבט יש משקל גדול יותר, כך הוא משתנה בתדריות נמוכה יותר, ולמעשה תדריות שינוי הביט היא אינדיקציה למיקום שלו.

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

איור 8.3 ייצוג בינארי של מספרים. ה-MSB משתנה בתדרות הcy נמוכה, ואילו ה-LSB משתנה בתדרות הcy גבוהה.

כיוון שמתפקידם במספרים שאינם בהכרח שלמים, הייצוג הבינארי של מספרים שלמים הוא יחסית פשוט, ולכן רק חת גרסה רציפה של אותו רעיון – פונקציות טריגונומטריות עם תדרות הולכת וגדלה. זהו בעצם הווקטור \mathbf{z} – הוא מכיל הרבה פונקציות טריגונומטריות בעלות תדרות הולכת וקטנה, ולפי התדרות שמתווסףת לכל איבר בסדרה המקורית ניתן לקבל אינדיקציה על מיקומו.



איור 8.4 Positional encoding. דוגמא למספר פונקציות בעלות תדרות הולכת וקטנה, בהתאם לאיבר אותו הן מייצגות (שמאלו). המuschא לקבץ השני של כל פונקציה בהתאם למיקום של האיבר אותו היא מייצגת – מען גרסה רציפה לקבץ שניי הביטים בייצוג בינארי של מספרים שלמים (ימין).

ישנו יתרון נוסף שיש לשימוש בפונקציות הטריגונומטריות – עבור כל צמד פונקציות בעלות אותו תדר ניתן לבצע טרנספורמציה ליניארית ולקבל תדר אחר (Relative Positional Information):

$$M \cdot \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k t + \phi) \\ \cos(\omega_k t + \phi) \end{bmatrix}, M = \begin{bmatrix} \cos(\omega_t \phi) & \sin(\omega_t \phi) \\ -\sin(\omega_t \phi) & \cos(\omega_t \phi) \end{bmatrix}$$

באופן זהה מקבלים באופן מיידי ייחוס בין כל ה-positions, מה שיכל לעזור בניתוח הקשרים שבין איברים שונים.

8.2.2 Self-Attention Layer

בנוסף ל-positional encoding, עליה הרעיון לבצע attention לא רק בין איברי הקלט לאיברי הפלט, אלא גם בין איברי הקלט עצמם. הרעיון הוא ליצר ייצוג חדש של סדרת הקלט באותו אורך כמו הסדרה המקורית, כאשר כל איבר בסדרה החדשה יציג איבר בסדרה המקורית בתוספת מידע על הקשר שלו לשאר האיברים. הרעיון הכללי אומר שיש לקחת כל איבר בסדרה, ולהחשב את הדמיון שלו לאור האיברים בסדרה. איברים דומים (קרובים) בסדרה יקבלו ערכי דמיון גבוהים, ואילו איברים שונים (רחוקים) בסדרה ינתנו ערכים נמוכים (ב-Transformer זה יכול להיות מילים שסביר שיופיעו בסמכיות, ובתמונה זה יכול להיות פיקסליהם דומים). דמיון בין איברים נמדד על פי הקשר שיש ביניהם, והוא מחושב באמצעות מכפלה פנימית בין וקטוריו ייצוג של האיברים. כל מכפלה פנימית בין שני איברים נותנת מוקדם שהוא מספר ממשי, ונור נור לסכום את מכפלת כל המוקדים באיברים המקוריים, ולקבל ייצוג חדש לאיבר המקורי.

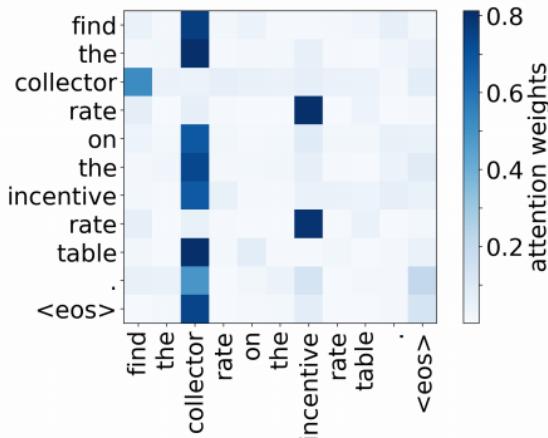
המכיל גם קשר בין האיבר הנוכחי לאיברים דומים בסדרה. ניתן להסתכל על וקטור המכיל את הקשרים של איבר מסוים בסדרה כדיו הבודד self-attention המבוסס על מטריצות של מקדים עבור סדרת הכוונה.

באופן פורמלי, בשביל לחשב את self-attention המבוסס על מטריצות self-attention נקראות Query, Key, Value, כאשר כל אחת מהן נוצרת על ידי הכפלת מטריצת משקלים באיבר הקלט. בעזרה מטריצות אלו מחשבים את attention score:

$$\text{Attention}(\text{Query}, \text{Key}, \text{Value}) = \text{SoftMax}\left(\frac{\text{Query} \cdot \text{Key}}{\sqrt{d_k}}\right) \cdot \text{Value}$$

כדי להבין כיצד הנוסחה זו מסייעת במציאת קשר בין איברים, נבחן כל איבר שלא בנפרד. עבור סדרת קלט x מקבלים שלוש מטריצות, כאשר כל איבר בסדרה המקורי x יוצר שורה בכל אחת מהמטריצות. כאשר לוקחים את השורה i , $q_i = Q$, ומתקבלים אותה בכל אחת מהשורות במטריצה K , מקבלים וקטור חדש, שכל איבר j בוקטור אומר עד כמה יש קשר בין האיברים i, j , בסדרה המקורי. ביצוע ההכפלת הזו עבור כל סדרת הקלט יוצר מטריצה חדשה בה כל שורה מייצגת את הקשר בין איבר מסוים לשאר איברי הסדרה. ההכפלת הזו היא בעצם $K \cdot Q$, כאשר כל מכפלה $q_i^T k_j$ מייצגת את הקשר בין האיבר i לאיבר j . את התוצאה מחלקים בשורש של ממד embedding כדי לשמר על יציבות הגדרי-אנט, ולאחר מכן מנורמלים על ידי Ax. SoftMax. באופן זהה מקבלים מטריצה של מספרים בטוחווות $[0, 1]$, המייצגים כאמור את הקשר בין כל שני איברים בסדרה המקורי. נסמן כל איבר במטריצה w_{ij} , ונוכל לקבל אותו יישור על ידי הנוסחה:

$$w_{ij} = \text{SoftMax}\left(\frac{q_i \cdot k_j}{\sqrt{d_k}}\right) = \frac{\exp\left(\frac{q_i^T k_j}{\sqrt{d_k}}\right)}{\sum_{s=1}^n \exp\left(\frac{q_i^T k_s}{\sqrt{d_k}}\right)}$$

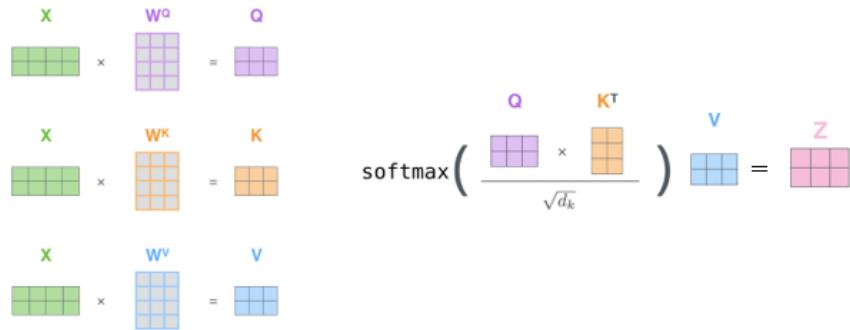


איור 8.5 מטריצת משקלים של המשפט "Find the collector rate on the incentive rate table". כל קשר בין שתי מילים חזק יותר-כך המשקל ביניהם גבוה יותר. כМОון שיש גם משמעות לסדר – המשקל בין "collector" ל-"Find" שונה מאשר בין "collector" ל-"Find".

כעת בעזרה משקלים אלו בונים ייצוג חדש לסדרה המקורי, על ידי הכפלתם בוקטור v :

$$z_i = \sum_{j=1}^n w_{ij} v_j = \frac{\sum_{j=1}^n \exp(q_i^T k_j)}{\sum_{s=1}^n \exp(q_i^T k_s)} v_j$$

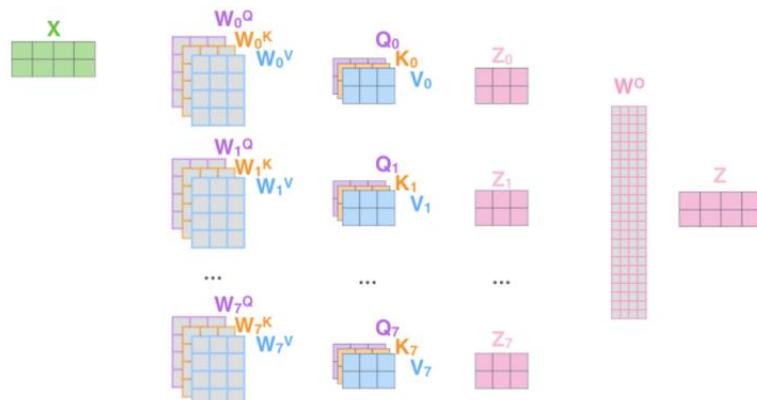
סדרה המתקבלת z היא למעשה ייצוג חדש של סדרה המקורי, כאשר כל איבר z_i מייצג איבר בסדרה המקורי יחד עם מידע על הקשרים ביןו לבין שאר איברי הסדרה. את סדרה המתקבלת ניתן להעביר ב-decoder המכיל שכבות נוספות, ובכך לבצע כל מיני משימות, כפי שיואר בהמשך.



איור 8.6 ביצוע Self-attention – ייצור מטריצות Query, Key, Value (ימין) ויחסם ה-attention score (שמאל).

8.2.3 Multi Head Attention

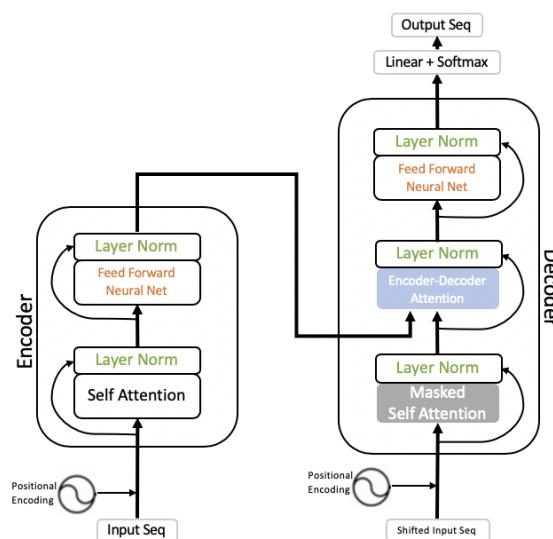
ניתן להשתמש במנגנון self-attention מספר פעמים במקביל. כל פעם מקבלים שלוש מטריצות (Q_r, K_r, V_r), ובעזרתיה מחשבים את הייצוגים החדשניים של איברי הסדרה (attention score). כל מנגנון זהה נקרא head, וסביר במקביל של כמה attention heads נקרא Multi-head attention. באופן זהה לכל איבר כניסה x_i יש כמה ייצוגים שונים Z_i , אותן ניתן להכפיל במטריצת משקלים W ולקבל את הייצוג המשוקל של אותו איבר באמצעות attention heads רבים.



איור 8.7 Self-attention with 8 heads

8.2.4 Transformer End to End

בעזרת מנגנוני multi head attention ו-positional encoding ניתן לבנות Transformer – ארכיטקטורה עבורה סדרות המבוססת רק על attention ללא רכיבי זיכרון.



איור 8.8 Transformer

כפי שניתן לראות באירור ה-transformer מורכב משני חלקים – encoder ו-decoder.encoder residual block סדרה מסוימת x (לרוב אחרי שבערבה embedding מסוים) ומצע עלייה positional encoding. לאחר מכן הסדרה עוברת דרך residual block של self-attention התוצאה $(x) + \text{attention}(x)$. על תוצאה זו מבצעים normalization (כפי שהוסבר בפרק 5.1.4). לאחר מכן יש residual block נוסף, המכיל שכבת fully connected normalization (כפי שהוסבר בפרק 5.1.4).encoder residual block סדרה נוספת מוצפנת x , ומשם יצא הפלט לכיוון ה-decoder.

ה-decoder בניו בדומה מודול דומה, עם שני הבדלים העיקריים: הקלט שלו הוא איברי הפלט שהו עד כה, ובנוסף יש בתחלת ה-decoder שכבה של Masked self-attention. שכבה זו מקבלת את כל איברי הפלט שהו עד כה, ומטרת decoder ה-decoder היא ללמידה עצמה מה האיבר הבא של הפלט. בשלב הראשון ה-decoder מבצע self-attention על איברי הסדרה שהתקבלו עד כה, וכך לומד ייצוג חדש שלהם, המכיל גם את הקשר בין איברי סדרה זו.

לאחר השכבה הראשונה יש שכבת multi head attention נוספת הנקראת Encoder-Decoder Attention Query, Key, Value מה-shaiah שילוב של ה-decoder וה-encoder: המטריצות Key , Value נלקחות מה-encoder, Query מගיע מה-decoder. כתם כשמבאים את המכפלת $K \cdot Q$, לא מוחפשים דמיון בין איברים של סדרת הפלט (בייצוג שלהם לאחר ה-encoder) לבין איברי סדרת הקלט (בייצוג שלהם לאחר שכבת masked). $Q \cdot K$ שלב זה דומה מאוד למקרה, רק שהיצוגים שהתקבלו לא נעזרים ברכיבי זיכרון. כאמור, המכפלת $K \cdot Q$ מייצרת מטריצת משקלים שכל איבר בה אומר מה היחס בין איבר בסדרה המקורית לבין איבר בסדרת הפלט. את המטריצה הזאת מכפילים ב-V, וכך מקבלים וקטור מסוים שהוא ייצוג חדש של איבר הפלט הבא. וקטור זה עובר בשכבות FC ו-SoftMax, וכך מתקבל איבר הפלט.

ניקח דוגמא ממאמר שנקרא DETR המראה כיצד ניתן להשתמש ב-transformer בשביל זיהוי אובייקטים בתמונה. בשלב הראשון לוקחים כל פיקסל בתמונה ומשווים אותו לשאר הפיקסלים (זהו בעצם המכפלת $K \cdot Q$). באופן זה ניתן למצוא אזוריים דומים ושונים בתמונה, כאשר דמיון ושווא זה לאו דווקא פיקסלים עם ערכים קרובים, אלא זה יכול להיות למשל שני אזוריים שונים בפונים של אדם. לאחר מכן מיצרים ייצוג חדש לתמונה, בעזרת המשקלים והכפלתם ב-V. שלב זה למעשה מאפשר לבצע זיהוי של אובייקטים, בלי לדעת מה הם אותם אובייקטים. בשביל לבצע סיוג לכל אובייקט שזוהה, מעבירים את הייצוג החדש של התמונה ב-decoder, וכך ה-Query שמנסים זה כל מיני ליבליים אפשריים, ומחפשים מבין כל ה-Query את הפלט של ה-decoder שמצילח ליצור תמונה שהכי דומה ל-Query.

אם למשל יש תמונה גדולה ויש אזור מסוים בו יש חתול, אז ה-encoder מוצא איפה החתול בתמונה, וה-decoder משווה את האזור הזה לכל מיני חיות אפשריות. כל Query שלא יהיה חתול, המכפלת $K \cdot Q$ תהיה קרובה ל-0, וה-decoder יזהה שה-Query הנוכחי לא תואם לאובייקט שזוהה. אך כאשר ה-Query היה חתול, אז ייקום ש- $K \cdot Q$ אחד לשני, המכפלת $K \cdot Q$ תביא לכך שהייצוג החדש $\sum_{j=1}^n z_j w_{ij}$ כן היה דומה לחתול. ייצוג זה עובר בשכבה FC, ולאחר מכן SoftMax יסוויג את התמונה הזאת חתול.

8.2.5 Transformer Applications

ה-transformer הציג ביצועי state-of-the-art בмагון שימושות, והוא הינו השימוש הSharah להמוני ישומים הנשענים על attention בלבד. מלבד הרמה הגבוהה של הביצועים, תהליכי האימון של transformer הוא הרבה יותר מהיר מרשותות קונבולוציה או רשותות רקורסיביות. כמו במקרים אחרים, גם עם transformer ניתן לבצע transfer learning, כלומר לחתוך transformer שאומן על משימה מסוימת, ולהתאים אותו למשימה חדשה שדומה למשימה המקורית. בפועל לא כל היחסים משתמשים בכל ה-transformer, אלא בהתאם למשימה לוקחים חלקים מסוימים שלו ובונים בעזרתם מודל עבור משימה מסוימת. נביא מספר דוגמאות:

Machine Translation – תרגום משפטים בין שפות שונות הוא יישום טריוויאלי של ה-transformer. המשימה היא לחתוך משפט ולהוציא משפט בשפה אחרת, וזה נעשה באמצעות ייצוג המשפט המקורי באמצעות self-attention encoderDecoder Attention על השפה אחרת.

Bidirectional Encoder Representations from Transformers (BERT) – מודל שפה מבוסס encoder. מודל שפה פונקציית המכבלת קלט טקסט ומחזירה את ההתפלגות למילה הבאה על פי כל המילים במילון. השימוש הכי מוכר אינטואיטיבי של מודל שפה הוא השלמה אוטומטית, שמצויה את המילה או המילים היכי סבירות בהינתן מה שהמשתמש הקליד עד כה. כאשר מבאים self-attention על משפטים, למעשה מקבלים ייצוגים חדשים שלהם יחד עם הקשרים בין המילים השונים. לכן ה-transformerencoderencoder יכול ליצור מודל שפה, אם מאמנים אותו בצורה מתאימה. המפתחים של BERTencoderencoder מבנו makbet כל מיני משפטים בשני כשי הכוונים – גם מההתחלת לסוף וגם מהסוף להתחלה, וכך הייצוגים שנלמדו קיבלו קונטקט שלם יותר. בנוסף, הם אימנו את המודל על משפטים בהם כל פעם באופן רנדומלי עושים masking למילים מסוימות, ומטרת המודל הוא לחזות את המילים החסרות.

— מודל לחריזה המילה הבאה במשפט. ניתן לקחת משפט שקטוע באמצעותו, ולבוחן מהי המילה הבאה באמצעות decoder בלבד. מכניםים משפט קטוע-ל-decoder ואז עוברים על המונחים בודקים את ההתאמנה שלחן למשפט הנתון, והמילה שהכי מתאימה נבחרת להיות המילה הבאה. המשפט הקטוע הוא למעשה ה-*Query*, וה-*Key*-*Query* שנכנסו הוא כל פעם מילה אחרת במילון, וכך בעזרת attention בוחנים איזה מילים בצוותה יתאים ביותר ביותר ל-*Key*-*Query* הנתון.

References

<https://arxiv.org/abs/1409.0473>

<https://arxiv.org/abs/1706.03762>

<https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263>

<https://towardsdatascience.com/transformer-attention-is-all-you-need-1e455701fdd9>

<https://arxiv.org/abs/1810.04805>

9. Computer Vision

להכין Region Based CNNs (R-CNN Family)

להכין מטריקות

https://github.com/taldatech/ee046746-computer-vision/blob/spring20/ee046746_tut_05_deep_semantic_segmentation.ipynb

9.1 Object Detection

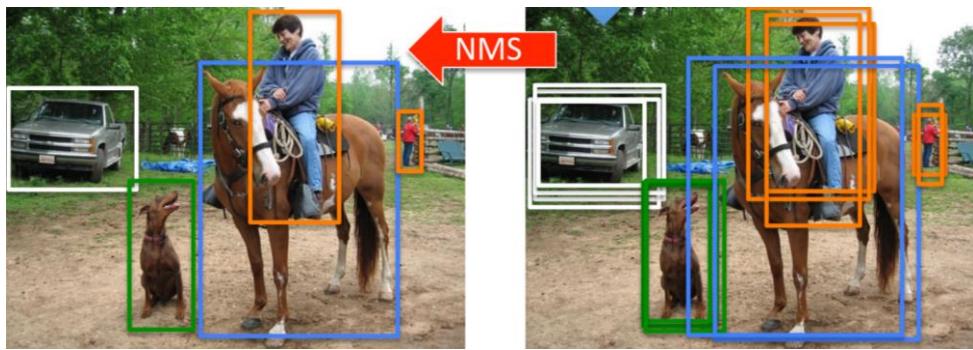
ה

9.1.2 You Only Look Once (YOLO)

עד שנת 2016 כל הgalais (detectors) המבוססים על מידע عمוקה (כמו למשל R-CNN family) היו galais דו-שלביים – השלב הראשון יצר אלפי הצעות (proposals) למסגרות מלכניות החשודות כמכילות אובייקטים, ואלו הוזנו dazu אחר זו לשלב השני אשר דיבק את המסגרות וביצע סיווג לאובייקטים המוכלים בהן. ארכיטקטורת YOLO, הנקראת מראשי התיבות של ONLY LOOK ONCE, הוצגה על ידי ג'וזף רדמן ב-2016, והיתה galai הראשון שמקורב משלב יחיד, ובו הרשות מבאת את המסגרות וגם מסוגת את האובייקטים שבתוכן בבת אחת. בנווסף, ארכיטקטורת YOLO מתאפיינת במיעוט פרמטרים וכמעט פועלות אrettmatiyot. היא אמנת משלמת על המבנה הרזה והאלגנטית שלה בדיק נמור יותר, אך מהירות הגבואה (שנובעת בעיקר מהיעדר האילוץ לעבד מסגרת ייחודה בשלב השני בכל מעבר של הלולאה) הפכה את גישת השלב היחיד לאטרקטיבית מאוד, במיוחדם, במיוחד למעבדים קטנים כדוגמת מכשירי mobile phone. בעקבות עובדה זו פותחו galais רבים על בסיס שלב יחיד, כולל גרסאות מתקדמות יותר של YOLO (הגרסה המתקדמת ביותר כיום היא 5).

NMS (Non-Maximum Suppression)

כמעט כל אלגוריתם של זיהוי אובייקטים מייצר מספר רב של מסגרות חשודות, כאשר רובן מיותרות ויש צורך לדלול את מספן. הסיבה לייצרת מספר גדול של מסגרות נובע מאופי פועלות galais. בזכות התכונות lokalkaliot של פועלות הקונבולוציה, מפת הפיצרים במקומות galai יונתנת לתיאור ממתריצת משbezות כאשר כל משbez שකולה לריבוע של הרבה פיקסלים בתמונה המקורית. רוב galais פועלם בשיטת עוגנים (anchors), כאשר כל משbez במקומות galai מנבאת מספר קבוע של מסגרות שעשוות להכיל אובייקט (למשל, ב-2 YOLOv2 המספר הוא 5, ובגרסאות המתקדמות יותר המספר הוא 3). השיטה זו יוצרת אלפי מסגרות שרק מיעוט מהן הן ממשמעותיות. בנווסף – יש ריבוי של מסגרות דומות בסביבת כל אובייקט (למשל – YOLOv3 מנבאת יותר 7000 מסגרות לכל תמונה). אחת הדרכים הפופולריות לסנן את אלפי המסגרות ולהשאר רק את המשמעותיות נקראת NMS. בשיטה זו מטבחעת השוואה בין זוגות של קופסאות מסוות מהחלון (למשל – חתול), ובמידה שיש ביןיהם חפיפה גבוהה – מוחקים את המסגרת בעלת הווודאות הנמוכה היותר ונשארים רק עם המסגרת בעלת רמת הווודאות הגבוהה. שיטה זו בוצנית בחישוב (סיבוכיות פרופורציונלית לריבוע מספר המסגרות) ואני חלק מהמודול המתאם, אך עם זאת הינה אינטואטיבית יחסית למימוש, ומשום כך נמצאת בשימוש נפוץ galais, כולל ב-YOLO.

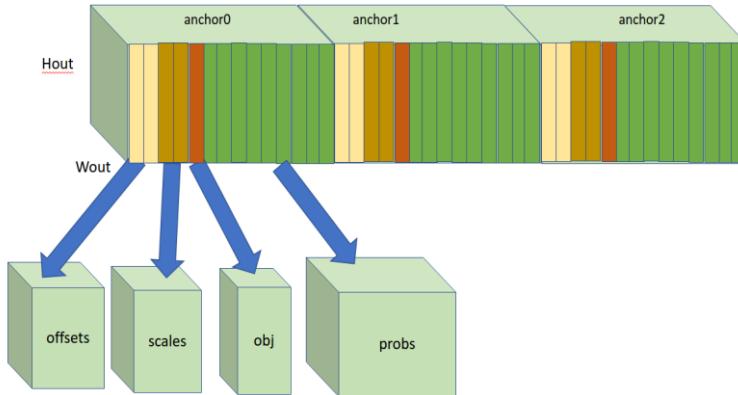


אייר 9.1 אלגוריתם NMS.

YOLO Head

כמו רוב הгалאים, YOLO הינו מבוסס-עוגנים (anchor-based), שהין תיבות מלכניות קבועות ושותות זו מזו בצורתן. לכל עוגן מוקצה מقطع של פיצרים במתה המוצא של הרשת וכל הניבויים במקטע זהה מקודדים כסתות (offsets) ביחס לממד העוגן. כפי שניתן לראות באIOR 9.2, הפיצרים של כל תא מרחבי במתה המוצא מחולקים למקטעים על פי העוגנים (שלושה עוגנים במרקחה זהה).

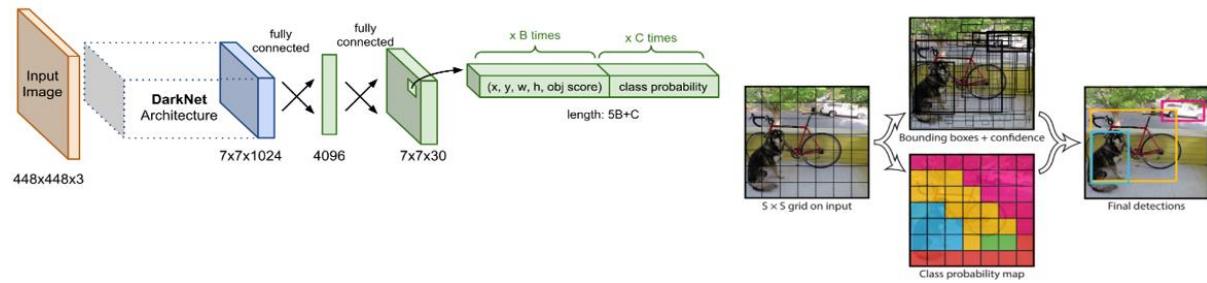
ניבוי הוא מסגרת שמרצתה נמצא במרקחה כלשהי בשטח התא (השקלול לריבוע של מספר פיקסלים בתמונה המקורית). ההסתה המדוקיקת של מרכז המסגרת ביחס לנקודה נתנת על ידי שני הפיצרים הראשונים ברצף.elog הממדים של הקופסה (ביחס לממד העוגן) ניתן באמצעות שני הפיצרים הבאים ברצף. הפיצר החמישי לומד את מידת ה-objectness, כפי שהסבירה לעיל. שאר הפיצרים ברצף של העוגן הנו כל הם הסתברויות המוותנות לכל מחלוקת (אם אוסף הנתונים מכיל 80 מחלוקות, יהיו 80 פיצרים כאלה). על מנת לקבל רמת ודאות סופית, יש להכפיל את מדד ה-objectness במדד הסתברויות המוותנות לכל מחלוקת.



AIOR 9.2 ראש YOLO.

YOLOv1

מודל YOLO מבוסס על גרסאות Backbone הנקראות Darknet ומשמשות לעיבוד פיצרים מתוך התמונה, וראש detection המקבל את הפיצרים האלה ומתאם לייצר מהם ניבויים למסגרות סביב אובייקטים. המודל מחלק את התמונה לרשת בעלת $S \times S$ משבצות, כאשר כל משבצת מנבאת N מסגרות של אובייקטים בשיטת העוגנים, כאמור לעיל. כל ניבוי כולל את מספר ערכיהם: הסט הקואורדינטות x, y , של מרכז המסגרת ביחס למשבצת, הגובה והרוחב של המסגרת, ורמת ה-objectness, כפי שהסבירה לעיל. בנוסף, כל מסגרת מבוצעת גם סיווג, ככלומר מנבאת את רמת הוודאות של השתייכות האובייקט לכל אחת מהמחלקות האפשריות. החידוש באלגוריתם נעוץ בעובדה שחייבי המסגרות וסיווג לאובייקטים נעשו במקביל, ולא באופן דו-שלבי. הרעיון הוא להתייחס לסוג האובייקט כדי פיצר שהרשת מנסה לחזות בנוסף למיקום וגודל של המסגרת.



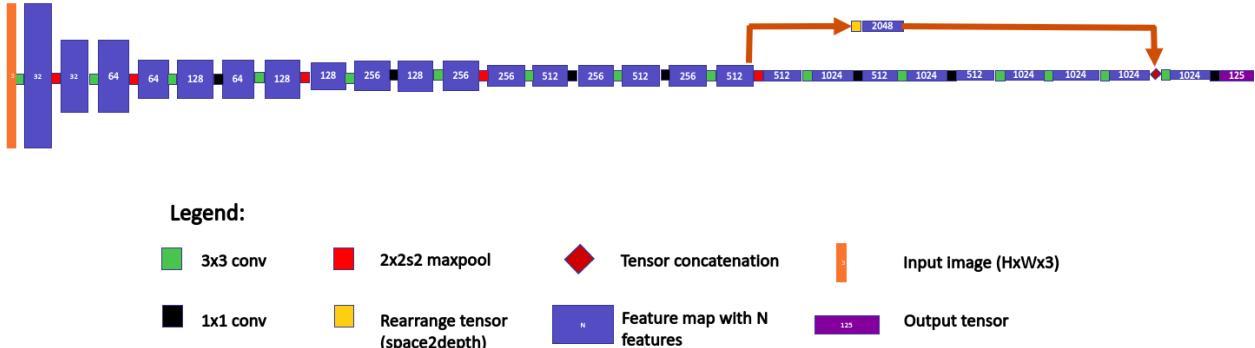
AIOR 9.3 ארכיטקטורת YOLO.

הגרסה הראשונה של ארכיטקטורת YOLO כללה שכבה fully connected שהוסרה בגרסאות הבאות. בונוס, פונקציית LOSS וסדר התוכנות של המסגרות במצב הרשת השטנו, אבל הרעיון נותר זהה.

YOLOv2

גרסה זו משתמשת ברשת Backbone הנקראות Darknet19 ובها 19 קונволוציות. המודל כולל 22 קונבולוציות (מלבד 5 שכבות ה-MAXPOOL המקבילות על מנת למצוא את הפיצרים), ועוד מסלול עוקף בסופו לשונו הרשת

המחזק את יכולת העיבוד. הכותב של המאמר המקורי, רדמן, נהג לפתח גם גרסאות "Tiny" לכל מודל. הווריאנט Tiny YOLOv2 מכיל רשת Backbone ישרה יותר ולא מסלול עוקף ומבנהו לינארי ופשוט מאוד. ביצועיו אמנים נמוכים יותר אך הוא מהיר מאד (207 תמונות לשנייה לעומת 67 של מודל 2 YOLOv2). על מעבד Titan X (מעניין לציין ש-2 YOLOv2 הוא המודל הראשון שאונן על תמונות במדים משתנים, תהליך המשפר את דיק המודל).



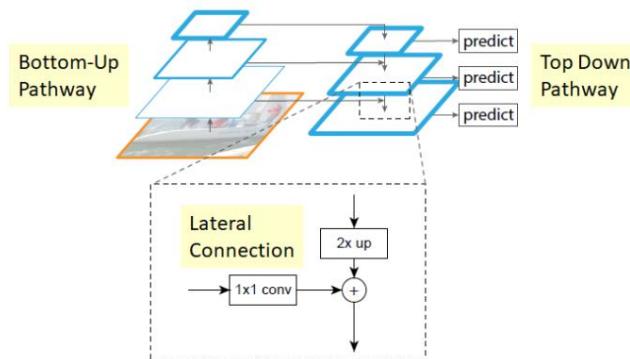
איור 9.4 ארכיטקטורת YOLOv2.

YOLOv3

כל דור נוסף של YOLO הציג חידושים ארכיטקטוניים שהגדילו את מורכבות החישוב וגודל המודל ושיפרו את ביצועיו. גרסה מספר 3 מבוססת על רשת Backbone גדולה בהרבה שנקראה Darknet53 ובה, כפי שעהלה ממשמה, 53 קונולציות. כמו כן הרשת מכילה צואואר של ארכיטקטורת Feature Pyramid Network (FPN) בעלת שלושה ראש גilioי, כאשר כל אחד מהם הוא בעל רוחולזיה שונה (38×38 , 19×19 , 76×76).

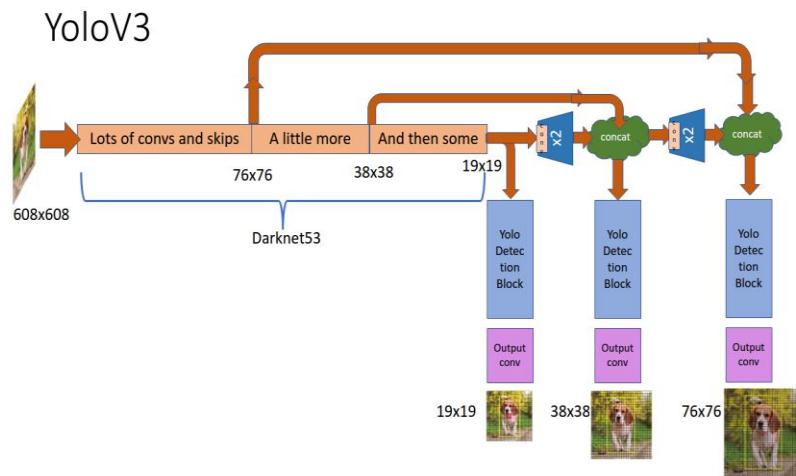
ארQUITקטורת FPN היא תוספת בקצה-h Backbone, אשר מגדילה חזרה באופן הדרגתית את מפת הפיצרים. תוספת זו נועדה ליצור מסלול Top-bottom במקביל למסלול Backbone-h Feed Forward ברשת Backbone-h, שבינוי למעשה בצורת Up, Bottom, בו ככל שמתקדמי בשכבות Backbone-one מבצע עיבוד ברמה גבוהה יותר והרוחולזיה המרחבית של מפת הפיצרים הולכת וקטנה. בעזרת השימוש ברשת-h FPN המודל לומד לנצל את המיטב משני עולמות: הוא משתמש במידע שטחוני במפת הפיצרים הגדולה, שהוא אمنם פחות מעובדת אך בעלת פיצרים ברוחולזיה מרחבית גבוהה, ובנוסף הוא מנצל גם את המידע ממפת הפיצרים הקטנה, שהוא אمنם בעלת פיצרים ברוחולזיה מרחבית נמוכה, אך עם זאת היא מעובדת יותר.

לאחר כל הגדלה של מפת הפיצרים מתבצע חיבור בין התוצאה בין מפת פיצרים קדומה יותר במדים זרים (מתוך Backbone-h). זאת בדומה לחיבורים העקיפים ברשת ResNet המסייעים להתקנסות האימון. השכבות השונות של FPN מאפשרות לאלי למצוא מיקום מדויק יותר של האובייקט ברוחולזיות השונות, מה שמעניק לרשת זו את היכולת להבחן באובייקטים קטנים בתמונה גדולה.



איור 9.5 ארכיטקטורת FPN (FPN), המשלבת מסלול top down לאחר ה-bottom up.

לרأس המודל של YOLOv3 יש מספר ענפי detection, אשר כל אחד מהם פועל על מפת פיצרים ברוחולציה שונה ובאופן טבעי מתמחה בגילוי אובייקטים בגודל שונה (הענף בעל הרוחולציה הנמוכה מתמחה בגילוי אובייקטים גדולים, והענף בעל הרוחולציה הגבוהה מתמחה בגילוי אובייקטים קטנים).



איור 9.6 ארכיטקטורת YOLOv3.

YOLOv4

רשת YOLOv4 היא בעלת ראש זהה של שתי הגרסאות הקודמות, אך ה-*Backbone* שונה. הוא נקרא CSPDarknet53 כאשר הוא קיזור של CSPNet. רשת זו מפצלת מפות פיצרים לטובות קונבולוציה בחלקים ואיחוד מחדש. פיצול זה אפשרי, כמו במאמר המקורי, חילול טוב יותר של הגרדיאנטים בשלב האימון. בנוסף, נעשה ברשת זו שימוש בפונקציית אקטיבציה הנקראת Mish (ולא Leaky ReLU) כמו בגרסאות הקודמות).

נקודות מענית – החל מגרסת זו התכורות אינן של יוצר המקורי ג'וזף רדמן, שהחליט לפרש מחקר ראייה ממוחשבת בגליל שיקולים אתיים של שימושים צבאים או שימושים הפוגעים בפרטיות.

YOLOv5

רשת YOLOv5 מוסיפה עוד שכליים על רשת Backbone של הדור הקודם, ומציגת אופרטור חדש המארגן פיקסלים סטטיסטיים בתמונה בממד הפייצרים. אופרטור זה דואג לכך שהכניסה לרשת היא לא עמוקה 3 פיצרים מכוקבל (RGB), אלא 12 פיצרים, תוך הקטנת הממד המרחב. באופן זה הרשת מתאמת לעיבוד תמונות ברוחולציה גבוהה, ואף לזרחות אובייקטים גדולים בקצב הרבה יותר, שכן שדה התמך (receptive field) של הקונבולוציות מכל מידע משטח תמונה גדול יותר.

9.1.4 Spatial Pyramid Pooling (SPP-net)

Spatial Pyramid Pooling (SPP) הינה שכבת pooling ברשת נוירונים, שמטרתה להטייר את האילוץ של רשתות קונבולוציה, הדורש שכבת הכניסה לרשת תכיל תמונה בגודל קבוע (כמו למשל רשת VGG, המתקבלת רק תמונות בגודל 224×224).

קיים קיימים מכשרי צילום רבים – החל מצלמות ניידות, מצלמות אבטחה ואף מצלמות טלפונים סלולריים ור呼iphones. מצלמות שונות עשויות להוציא כפלט תמונות בגודלים שונים, מגוון סיבובות (למשל אינט הtmpuna או מטרת המכשיר). אם נרצה לבצע סיווג באמצעות רשת נוירונים לכל אותן תמונות, נאלץ לבצע שלב נוסף בתחלת הדרך – התאמת התמונה בכניסה לרשת.

נשים לב כי על מנת להתאים תמונה כלשהי לגודל מסוים, לרוב יבוצע חיתוך (crop), או שינוי גודל (resize/wrap). פעולה אלה עלולות לפגוע ביזיה עקב שינוי היחס בתמונה (מתיחה/כיווץ), החסירה של פרטים או שילוב של השניים. מוטיבציה זו היא שהובילה לשימוש בטכניקת SPP.

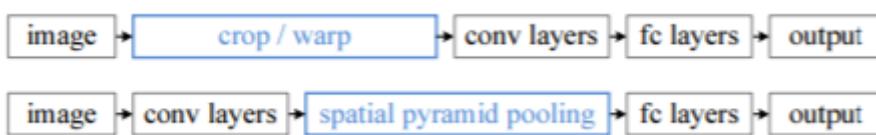
ניתן לראות דוגמא לשינוי גודל באמצעות מתיחה ולחיתוך באירור הבא:



איור 9.7: מימין - שינוי פרופורציה. משמאלי – חיתוך.

למעשה, הדרישה לתמונה קלט בגודל קבוע בכניסה לרשותת אלה אינה הכרחית, שכן שכבות הקונבולוציה יכולות לחלק מאפייני קלט (feature maps) בכל גודל. לעומת זאת, שכבות ה-FC בעומק הרשות הן השכבות שדורשות בכניסה להן קלט בגודל קבוע.

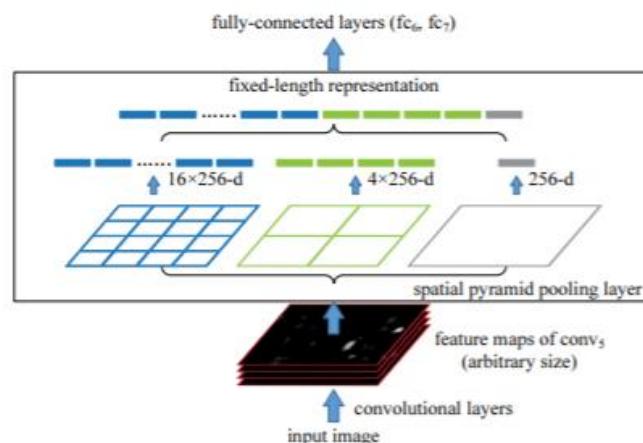
כעת, לאחר שהובאה המוטיבציה ליצירת רשת זו, ניתן להבין את אופן פועלתה. על בסיס שכבת ה-SPP מתבססת רשת SPP-Net. החידוש במבנה הרשת, הוא שבמקום שכבות בכניסה לרשות (לפניהם שכבות הקונבולוציה) תבצע התאמת תמונה הקלט לגודל הנדרש ברשות. ההתאמנה המבוצעת בשכבת SPP תתבצע לאחר מציאת המאפיינים בשכבות הקונבולוציה, אך לפני שכבת ה-FC, כמפורט באירור הבא:



איור 9.8 – מבנה של רשת CNN קלאסית (למעלה) לעומת מבנה של רשת SPP-Net (למטה).

הרעין מאחורי שכבת SPP הוא חלוקה של הפלט של שכבות הקונבולוציה, ביצוע max-pooling בכל חלק ורשור של התוצאות לווקטור שגודלו אחד. במילים אחרות, עבור כל ה-features המתקבלים לאחר שכבות הקונבולוציה, מייצרים שלושה וקטורים לכל feature:

- וקטור בגודל 1 המתקבל באמצעות ביצוע max-pooling על כל הערכים באותו ה-feature.
 - חלוקה של ה-feature ל-4 תת-חלקים (2×2) וביצוע max-pooling בכל חלק מתוכם. מתקבל וקטור בגודל 4.
 - חלוקה של ה-features ל-16 תת-חלקים (4×4) וביצוע max-pooling בכל חלק מתוכם. מתקבל וקטור בגודל 16.
 - שרשור כל הווקטורים יחד לוקטור בעל 21 ערכים.
 - בסיום התהליך, מקבל שכבה ביצוג אחד בגודל 21, בהכפלה במספר ה-features שהתקבלו משכבות הקונבולוציה. שכבה זו "מחליפה" את שכבת ה-pooling הממוקמת לאחר שכבת הקונבולוציה האחרונה ותוצרה הוא היקלט לשכבת ה-FC.
- ניתן לראות את מבנה הרשת באירור הבא, בו התקבלו 256 features:



איור 9.9: ארכיטקטורת SPP-Net.

9.2 Segmentation

אחד האתגרים הכי משמעותיים בעולם הריאיה הממוחשבת הוא זיהוי אובייקטים בתמונה והבנת המתרחש בה. אחת הטכניקות הקלסיות להתחממות עם משמעותה זו הינה ביצוע סגמנטציה, כלומר, התאמת label לכל פיקסל בתמונה. בהתאם הסגמנטציה מבצעים חילוק/בידול בין עצמים שונים בתמונה המצלמת באמצעות סיווג ברמה הפיקסל, כלומר כל פיקסל בתמונה יסוווג וישויר למחלקה מסוימת.

ישנם שימושים מגוונים באלגוריתמים של סגמנטציה – הפרדה של עצמים מסוימים מהרקע שמאחוריהם, מציאת קשרים בין עצמים ועוד. לדוגמה, תוכנות של שיחות וUID, zoom, skype, teams וכו', מאפשרות בחירת רקעים שונים עבור המשטח, כאשר מלבד הרקע הנבחר רק הגוף של המשטח מוצג בוודיאו. הפרדת גוף האדם מהרקע והטמעת רקע אחר מtbodyות באמצעות אלגוריתמים של סגמנטציה. דוגמא נוספת – ניתן לזהות בתמונה אדם, כלב, וביניהם רצעה, ומכך ניתן להסיק שתוך התמונה הוא אדם מחזיק כלב בעזרת רצעה. במקרה זה, הסגמנטציה מועדה למצאו קשר בין עצמים ולהבין את המתרחש.

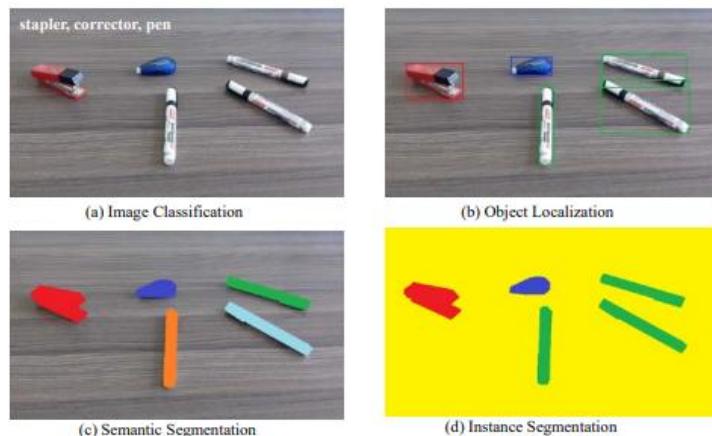
9.2.1 Semantic Segmentation vs. Instance Segmentation

קיימים שני סוגי עיקריים של סגמנטציה:

Semantic segmentation (חלוקת סמנטיבית) – חלוקה של כל פיקסל בתמונה למחלקה אליה העצם אותו הוא מייצג שיר. למשל, פיקסל יכול להיות משיר לכלי רכב, בן אדם, מבנה וכו'.

Instance segmentation (חלוקת מופעית) – חלוקה של פיקסל בתמונה למופע של אותה מחלקה אליה העצם אותו הוא מייצג שיר. במקרה זה, בתמונה בה מופיעים מספר כלי רכב, תבצע חלוקה של כל פיקסל לאיזה כלי רכב אותו פיקסל מייצג – מכונית 1, מכונית 2, אופנוע 1, משאית 1 וכו'.

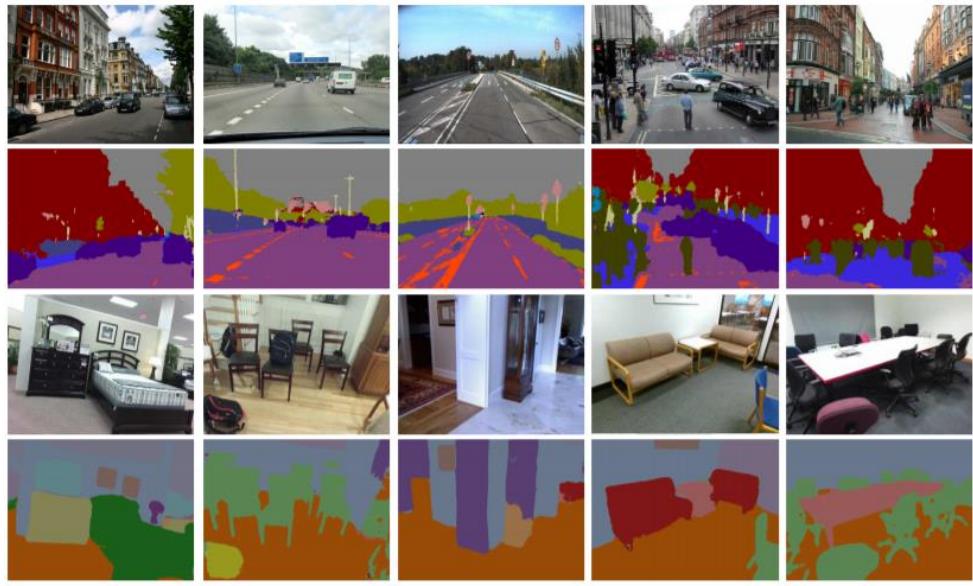
הבדל העיקרי בין שני סוגי אלו הוא ברמת עומק המידע של פיקסל – המידע עשוי לסווג את הפיקסל למחלקה כלשהי, או לעצם ספציפי בתמונה. עומק המידע משליך גם על עלות המידע.חלוקת הסמנטיבית מבצעת שירותים, בעוד שהחלוקת המופעית דורשת בנוסף ביצוע של זיהוי אובייקטים כדי לסווג מופעים שונים של המחלקות.



איור 9.7 שימושות שונות תחת התחום של Computer Vision. ניתן להבחין בהבדל שבין Semantic segmentation (התאמת כל פיקסל למחלקה מסוימת) לבין Instance segmentation (התאמת כל פיקסל למופע של מחלקה מסוימת).

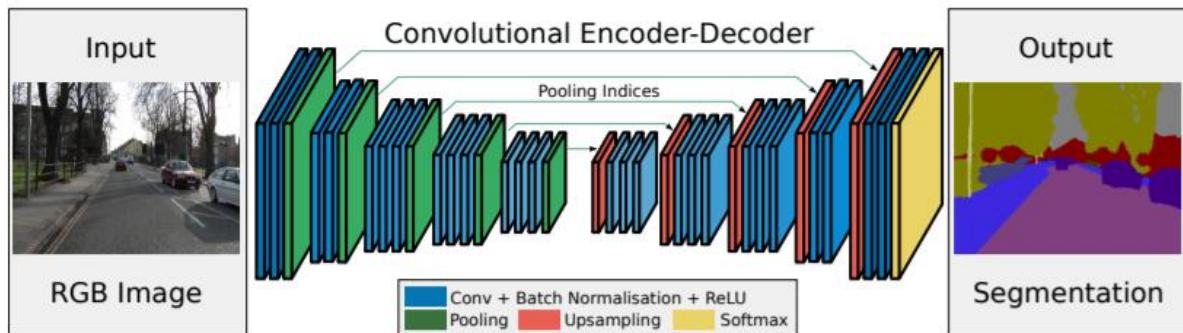
9.2.2 SegNet neural network

רשת SegNet הינה רשת קוונבולוציה عمוקה, שמטרתה לבצע חלוקה סמנטיבית (Semantic segmentation) לתמונה הקלה. בתחילת הרשת פותחה להבנה של תמונות חזות (למשל כביש עם מכוניות ובצדדים בתים והולכי רגל) ותמונות פנים (למשל חדר עם מיטה וכיסאות). הרשת נבנתה מtower מטרה להיות יעילה בהיבטי זיכרון וזמן חישוב, תוך שמירה על דיוק מעשי.



איור 9.7 סיווג סמנטי באמצעות רשת SegNet עבור תמונות פנים וחוץ.

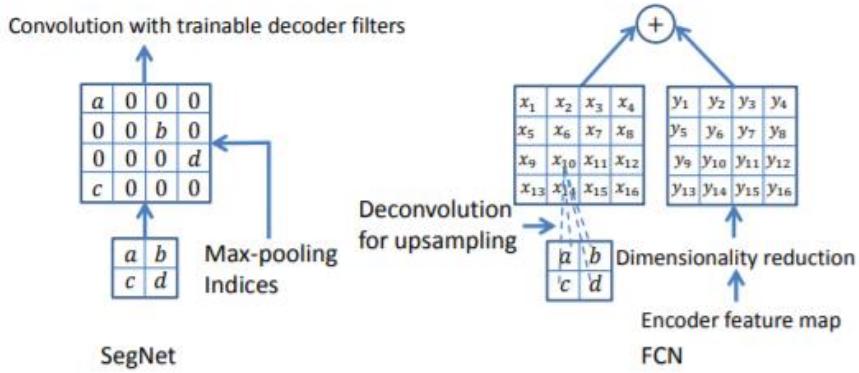
הרשת בנוייה כארQUITקטורת מקודד-מפענה (encoder-decoder). המקודד מורכב מ-13 השכבות הראשונות של רשת VGG16, ומטרתו לחלץ את מפת המאפיינים (feature maps). לכל שכבה קידוד יש שכבת פיענוח תואמת ומכאן שגם רשת המפענה מכילה 13 שכבות. מטרת המפענה היא לבצע sampling-קע וליצור תמונות מאפיינים בגודל המקורי. את מוצא המפענה מעבירים דרך מסווג SoftMax, המתאים את המחלקה בעלת ההסתברות הגבוהה ביותר לכל פיקסל בנפרד.



איור 9.8 ארכיטקטורת רשת SegNet

המקודד מורכב משכבות קוונטולזיה, אחריהן שכבות נרמול (batch normalization) ושכבות אקטיבציה מסוג ReLU. לאחר שכבות אלו, ישנה שכבת max-pooling המבצעת subsampling, בעלת גודל חלון 2×2 ומרוחה בגודל 2.

ההידוש ברשת זו הוא אופן הפעולה של המפענה. בשונה מרשתות אחרות, בהן תהליכי sampling-קע גורר ביצוע חישובים עבור הפיענוח, כמתואר באյור הבא, הרעיון ברשת זו הוא שמירת מיקומי ערכי המקסימום הנבחרים מכל רבייה. רק הערכים שנבחרו כמקסימום יושחזו בתהליך ואילו הערכים יתרפסו.



איור 9.9 שכבה פענוח ברשת SegNet לעומת רשת FCN.

ארQUITטורה זו מביאה את הרשת לビיצועים טובים בהיבטי זמן חישוב, על חשבון פגיעה מסוימת בדיק הרשות. למקרה זאת, ביצוע הרשת מתאים לשימושים פרקטיים והפגעה בדיק קטנה מאוד.

בדומה למקודם, לאחר כל שכבה sampling-skip בפענוח, יופיעו שכבות קונבולוציה, שכבות נרמול ושכבות אקטיבציה מסוג ReLU. את מוצא המפענה מעבירים דרך שכבת SoftMax המבצעת סיווג ברמת הפיקסל. מוצא הרשת, שהוא גם מוצא שכבת h-Max, הינו מטריצת הסתברויות, כאשר עבור כל פיקסל יש קטור באורך K, כאשר K הוא מספר המחלקות לסיווג. כמובן שהסיווג מתבצע בהתאם לגובהה ביותר המתאימה לכל פיקסל.

אימון הרשת בוצע על בסיס מידע ייחסי של 600 תמונות דרך צבעוניות בגודל 480×360 , שנלקחו מבסיס המידיע CamVid road scene dataset. סט האימון הכיל 367 תמונות וסיט הבדיקה הכליל את 233 התמונות הנותרות. המטריה הייתה להזדהות בתמונות אלה 11 מחלקות (דרך, בניין, מכונית, הולכי רגל וכדומה). כל תמונה עברה נרמול מקומי לערך הניגודיות של תמונה הקלט לפני הכנינה לרשת. האימון בוצע בשיטת SGD, עם קצב למידה קבוע שערכו 0.1 ומומנטום שערכו 0.9. האימון נמשך עד שהשגיאת התכנסה. לפני כל epoch סט האימון עורב וחולק ל- mini-batch של 12 תמונות. פונקציית המחר הינה cross-entropy:

לעתים, נדרש לבצע איזון-מחלקות (class balancing). מונח זה מתייחס מצבי בו קיימים שונים גודל בין כמות הפיקסלים המשווים לכל מחלוקת, למשל כאשר קיימת הטיה מסוימת - סצינה שבה רובו מכילה בניינים / דרכי. במצב זה, יבוצע משקל מוחודש לפונקציית השגיאה, באמצעות תהליך "איזון התדריות החציניות" (median frequency balancing). התהליך משקל מחדש את המחלקות בפונקציית המחר, באופן יחסית ליחסן של תדריות הופעות המחלקות בכל סט האימון, תוך חלוקה בתדריות הופעת המחלוקת:

$$\alpha_c = \frac{\text{median freq}}{\text{freq}(c)}$$

משקל זה משנה את היחסים בפונקציית המחר כך שהתרומה של כל המחלקות לפונקציית המחר תהיה שווה. לכן, הוא מעניק למחלקות הגדולות יותר משקל נמוך יותר ולמחלקות הקטנות משקל גבוה יותר.

9.2.3 Atrous Convolutions (Dilated Convolutions)

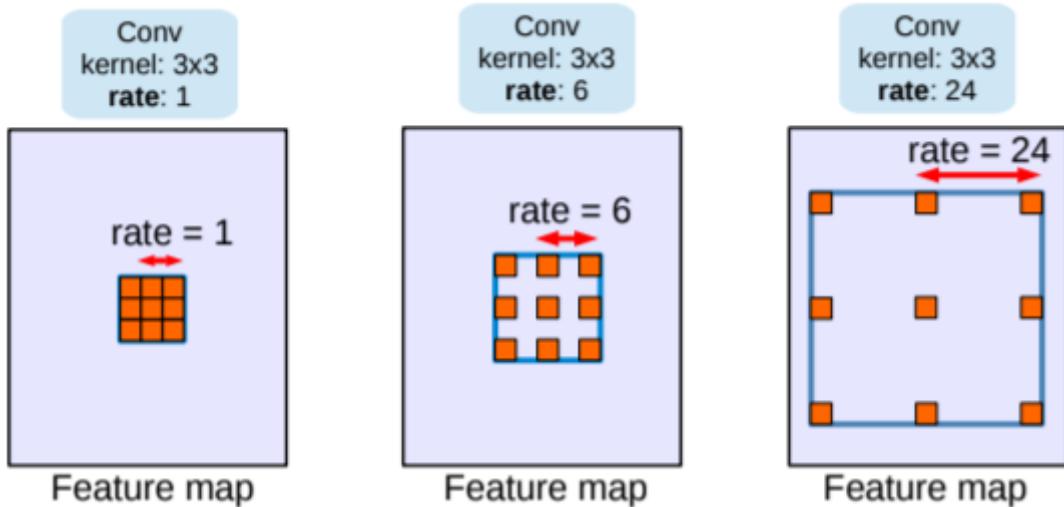
המונח Atrous מזכיר בשפה הצרפתית – "à", שם שמעוטו "עם חורים". אך, ניתן לתרגם את המונח convolution Atrous כ- "קונבולוציה מחוררת", ובמשמעותו מעט יותר מתאימה – קונבולוציה מרוחקת (או בשם אחר – Dilated convolution – קונבולוציה מרווחת).

בטכניקת קונבולוציה זו, יש שימוש בפרמטר נוסף בנוסף לנוסחת הקונבולוציה – dilation rate. פרמטר זה מסמן את המרווח בין כל איבר בגרעין הקונבולוציה (הרחבת על פרמטר זה – בפרק 5.1.2). נוסחת הקונבולוציה עברו במקרה החד ממד' יחד עם פרמטר ההתרחבות r ניתנת לתיאור באופן הבא:

$$y[i] = \sum_{k=1}^K x[i + r \cdot k]w[k]$$

עבור $1 = r$ מתקבלת הקונבולוציה הרגילה, כאשר ישנו dilation של $1 > r$, מתקבלת קונבולוציה מרוחקת בפקטור r . יתרונה של קונבולוציה נעוץ בכך שuber או kernel וuber אותה כמות חישובים, מרחיבים את ה- (FoV) field of view של הקונבולוציה.

ניתן לראות את הרחבת field of view באירור הבא:



איור 10.9: קונבולוציה מרוחקת דו-ממדית, עם kernel בגודל 3×3 ופרמטר התחרבות $1, 6, 24 = r$. בהתאם לכל פרמטר מתקבל - field-of-view בגודל שונה – $3, 16, 49 \times 49 \times 3$ בהתאם.

9.3 Face Recognition and Pose Estimation

9.3.1 Face Recognition

אחד מהיישומים החשובים בראיה ממוחשבת הינו זיהוי פנים, כאשר ניתן לחלק משימה זו לשולש שלבים:

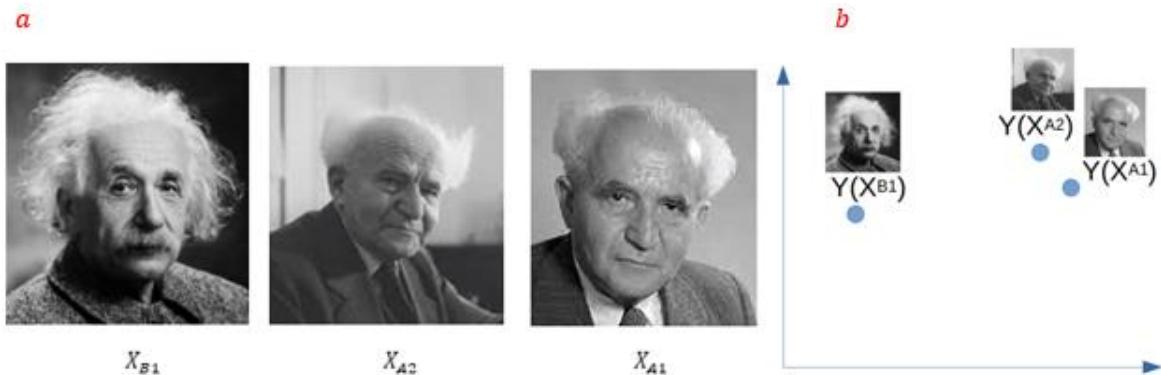
1. Detection – מציאת הרצופים בתמונה.
2. Embedding – מיפוי כל הרצוף למרחב חדש, בו המאפיינים שאינם קשורים לתיאור הפנים (למשל: זווית, מיקום, תארה וכדו) אינם משפיעים על הייצוג.
3. Searching – חיפוש במאגר של תמונות למציאת תמונה פנים הקרובה לתמונה הפנים שהוצאה מהתמונה המקורית.

גישה פשוטה, כמו למשל בניית מילוי המכיל מספר יציאות כמספר הפנים אותם רצים לזהות, הינה בעייתית מושתת סיבות עיקריות: ראשית יש צורך באלפי דוגמאות לכל אדם (שלא ניתן בהכרח להציג). כמו כן, נוצרר למד את המערכת מחדש בכל פעם שהוא משוחח מחדש. כדי להתגבר על בעיות אלו מוצעים "מידת מטריקה" (metric learning) בה מזקקים מאפיינים של פנים ויוצרים וקטור יחסית קצר, למשל באורך 128, המכיל את האלמנטים המרכזיים בתמונה הפנים. כתע נפרט את שלושת השלבים:

1. מציאת פנים.
2. תיאור פנים.
3. למציאת פנים בimensional המשמש ברשות המבצעות detection, כפי שתואר בפרק 9.1. שיטה מקובלת למשימה זו הינה SIFT, המבוססת על חלוקת התמונה למשבצות, אשר עבר כל משבצת בוחנים האם יש בה אובייקט מסוים, מהו אותו אובייקט, ומה ה- bounding box שלו.

כאמור, המשימה בתיאור פנים נעשית בעזרת metric learning, כאשר הרעיון הוא לזרק פנים לוקטור שאינו מושפע ממאפיינים שלא שייכים באופן מהותי לפנים הספציפיות האלה, כגון זווית צילום, רמת תאורה ועוד. בכך לעשוות זאת יש לבנות רשת המקבלת פנים של בנאים ומחזירה וקטור, כאשר הדרישת היא שעבור שתי תמונות של אותו אדם יתקבלו וקטורים מאוד דומים, ועבור הרצופים של אנשים שונים יתקבלו וקטורים שונים. למעשה, פונקציית h-loss מקבל בכל פעם \min , ותעניש בהתאם לקרבה בין וקטורים של אנשים שונים וריחוק בין וקטורים של אותו אדם.

cutת נניח שיש לנו קלט X , המכיל אוסף פרצופים. כל איש יסומן באות אחרת – A,B,C. ותמונהות שונות של אותו אדם יסומנו על ידיאות ומספר, כך למשל X_{A1} זהה התמונה הראשונה של אדם A בסט הקלט X , וכמוון ש- X_{A1} - X_{A2} - X_{B1} הן שתי תמונהות של אותו אדם. באופן גרפי, בוד-ממד ניתן לתאר זאת כך (בפועל הוקטוריהם המיצגים פניהם יהיו במד גובה יותר):



איור 9.10 (a) דוגמאות מסט הפרצופים X . (b) איך נרצה שהדעתה ימפה לממד חדש Y .

כאמור, נרצה לבנות פונקציית loss שמעודדת קירבה בין X_{A1} ו- X_{A2} , וריחוק בין X_{A1} ו- X_{B1} . פונקציית loss מרכיבת משני איברים, המודדים מרחק אוקלידי בין וקטורים שונים:

$$L = \sum_X \|Y(X^{Ai}) - Y(X^{Aj})\| - \|Y(X^{Ai}) - Y(X^{Bj})\|$$

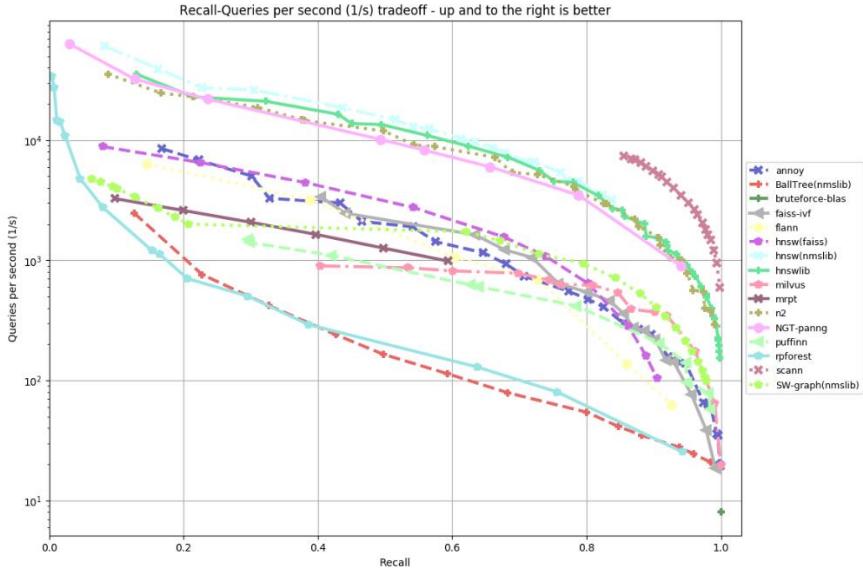
כאשר האיבר הראשון ינסה להביא למינימום וקטורים של אותו אדם, והאיבר השני ינסה להביא למינימום וקטורים של פרצופים שאינם שייכים לאותו אדם. כיוון שנרצה להימנע מקבלת ערכים שליליים, נוסיף פונקציית מקסימום. בנוסף, ניתן 'להרחיק' תוצאות של פרצופים שונים על ידי הוספה קבוע k , כך שהפרש בין המרחק של פרצופים של אנשים שונים לבין המרחק של פרצופים של אותו איש יהיה לפחות k :

$$L = \sum_X \max(\|Y(X^{Ai}) - Y(X^{Aj})\| - \|Y(X^{Ai}) - Y(X^{Bj})\| + k, 0)$$

loss זה נקרא triplet loss, כיוון שיש לו שלושה איברי קלט – שתי תמונהות של אותו אדם ואחת של מישחו אחר. כאמור, הפלט של הרשת הנלמדת צריך להיות וקטור המאפיין פנים של אדם, ומטרת הרשת היא למפות פרצופים שונים של אותו אדם לווקטורים דומים עד כמה שניתן, ואילו פרצופים של אנשים שונים יקבלו וקטורים רחוקים זה מזה.

3. מציאת האדם

בשלב הקודם, בו ה被执行 האימון, יצרנו למשה מאגר של פרצופים במרחב חדש. cutת כsigmoid פרצוף חדש, כל שנוטר זה למפות אותו למרחב החדש, ולהפץ במרחב זה את הוקטור הקרוב ביותר לווקטור המיצג את הפנים החדשות. בכך לעשות זאת ניתן להשתמש בשיטות קלאליסיות של machine learning, כמו למשל חיפוש שכן קרוב (כפי שהסביר בחלק 2.1.3). שיטות אלו יכולות להיות איטיות עבור מאגרים המכילים מילוני וקטורים, וישן שיטות חיפוש מהירות יותר (ובדרך כלל המהירות באה על חשבון הדיווק). בעזרת השיטה המובילה CarGAN (SCANN) ניתן להגיע לכמה מאות חיפושים שלמים בשניה (הchiposh ב-100 ממדים מתוך מאגר של 10000 דוגמאות).



איור 9.11 השוואת ביצועים של שיטות חיפוש שונות. עבור פרצוף נתון, מהפשים עברו וקטור תואם במדד החדש המכיל ייצוג וקטורי של הפרצופים הידועים. בכל שיטה יש טריד-אוף בין מהירות החיפוש לבין הדיקט.

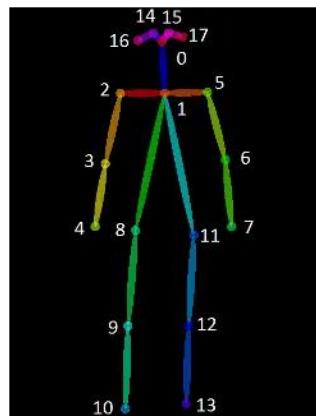
מלבד זההו וסיווג פנים, יש גם שיטות של מציאת אלמנטים של פנים הוכולות אף, עיניים וכו'. אחת השיטות המקובלות משתמשת בשערוך הצורה של פנים אנושיות, וניסיון למצוא את איברי הפנים לפי הצורה הסטנדרטית. בשיטה זו ראשית מבצעים יישור של הפנים והתאמאה לסקירה אוניות (על פי מרחק בין האיברים השונים בפנים), ולאחר מכן מטילים 68 נקודות עניין מרכזיות על התמונה המיושרת, מתוך ניסיון להתאים בין הצורה הידועה לבין התמונה המבוקשת.



איור 9.12 זההו אזוריים בפנים של אדם על ידי התאמת פנים לסקירה אונשית והשוואה למבנה של פנים המכיל 68 נקודות מרכזיות.

9.3.2 Pose Estimation

ישום פופולרי נוספת של אלגוריתמים השייכים לראיה מוחשבת הינו קביעת תנוחה של אדם – האם הוא עומד או יושב, מה התנוחה שלו, באיזה זווית האיברים נמצאים וכו'. ניתן להשתמש בניתוח התנוחה עבור מגוון תחומיים – ספורט, פיזiotרפיה, משחקים שונים ועוד. לרוב, תנוחה מיוצגת על ידי המיקומים של חלק גוף עיקריים כגון ראש, כתפיים, מפרקים וכו'. ישם כמה סטנדרטים מקובלים, למשל COCO, posenet, COCO הנקודות מיוצגות באמצעות מערכת מערך של 17 נקודות (בדו-מיד):



איור 9.13 מיפוי תנוחה ל-17 נקודות מרכזיות.

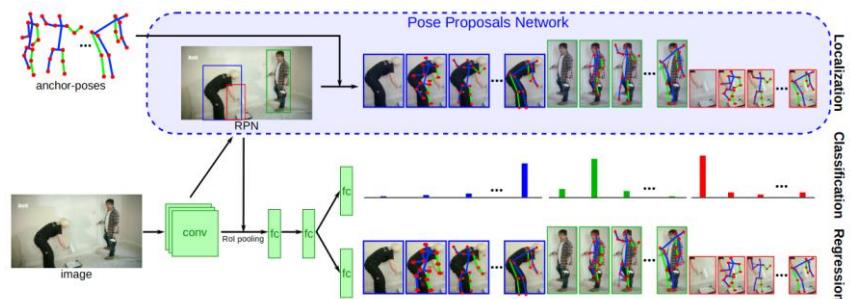
שאר הפורמטים דומים; מוטיפים עוד מידע (למשל מיקום הפה), משתמשים בתלת ממד במקום בדו ממד, משתמשים בסידור אחר וכך. כאשר רוצים לאוסף נתונים על מנח גוף שונים, ניתן לשים חישנים על אותן נקודות מרכזיות וככה לקבל מידע על מנה הגוף לאורך זמן.

רשות לצורך קביעת תנוחה יכולה לטפל במקרה כללי, בו יש מספר אנשים בתמונה, או במקרה הפרטי של גוף יחיד. במקרה השני כמובן יותר פשוט, מכיוון שישנו פלט יחיד, אותו ניתן להזוז באמצעות גרגסיה (למשל, 34 מספרים שמטארים את המיקומים של 17 הנקודות בפורטט COCO בדו-ממד). במקרה זה ניתן לעשות למידה סטנדרטית לחולstein של בעיית גרגסיה בעלי 34 יציאות.

ישנן מספר גישות כיצד להכילה את הרשות כך שתוכל לטפל גם במקרה הכללי בו יש יותר מגוף אחד בתמונה. באופן נאיבי ניתן לבצע תהליכי מוקדים של מציאת כל האנשים בתמונה, ואז להפעיל על כל אחד מהם בנפרד את הרשות שמבצעת גרגסיה, כפי שתואר לעיל. שיטה נוספת פועלת בכיוון הפוך – ראשית כל הרשות מוצאת את כל האיברים בתמונה נתונה, ולאחר מכן משיכת אותן לאנשים שונים. השיטה השנייה נקראת "מלמטה למעלה" (top-bottom), כלומר שקדם כל היא מוצאת את הפרטים ולאחר מכן מכך מכך אותם. גישה זו עיליה ל蹶ה בו יש הרבה חפיפה בין האנשים בתמונה, כיון שאין לה צורך לבצע תהליכי מוקדים של הפרדת האנשים. השיטה הראשונה, הנקראת "מלמטה למטה" (bottom-top), תהיה פשוטה יותר עבור מקרים בהם אין חפיפה בין האנשים בתמונה וכל אחד מהם נמצא באזור שונה בתמונה, כיון שאין צורך לשירות איברים לאנשים.

רשות פופולרית לקביעת תנוחה נקראת *pose estimation*, המורכבת למעשה משתי תת-רשתות ופעולות בשיטת top-bottom. הרשות שאחראית על שייר חלק גוף לאדם מסוים, נקראת (PAF, part affinity fields) והרעיון שלו הוא לייצג כל איבר כשדה וקטורי. ביצוג זה הווקטוריים השונים מצביעים לכיוון איבר הגוף ה'בא בתו' (למשל זרוע מצביעה ליד), וככה ניתן לשירות איברים שונים אחד לשני, ואת כל ייחד לגוף מסוים.

רשות פופולרית אחרת, הפעולת בגישה down-top, נקראת LCR-NET, והוא מבוססת על רעיון של 'מיקום-סיווג-רגסיה' (Localization-Classification-Regression). בשלב הראשון יש תת-רשת המיצרת עוגנים עבור אנשים, אזורים בהם הרשות חושבת שנמצא באדם, ולאחר מכן הרשות משרכת את התנוחות שלהם. בשלב השני ככלומר, אזורים בהם הרשות חושבת שנמצא באדם, ככלומר כל עוגן מקבל ציון המיציג את טיב השערור של העוגן והתנוחה של האדם הנמצא בתוכו. השלב השלישי מפשט את העוגנים ומשקל את השערור הסופי בעזרת מיצוע של הרבה עוגנים. שלושת השלבים משתמשים ברשת קונבולוציה משותפת, כמפורט באיור.



.איור 9.14 Localization-Classification-Regression 9.14

9.5 Few-Shot Learning

יכולת הצלחתם של אלגוריתמי למידה עמוקה נעה על כמות ואיכות הדעתה לאימון. עבור משימת סיווג תמונות (Image Classification), נדרש שבעור כל קטגורית סיווג תהיה כמות גדולה של תמונות מוגנות (עם הבדלי רקעים, בהיות, זווית וכו'), ובנוסף יש צורך בכמות דומה של דוגמאות בכל קטגוריות הסיווג. חוסר איזון בין כמות התמונות בקטגוריות השונות משפייע על יכולת הלמידה של האלגוריתם את הקטגוריות השונות ועל כן עלול ליצור הטיה בתוצאות הסיווג לטובות הקטגוריות להן יש יותר דוגמאות. פרק זה עוסק בשיטות כיצד ניתן להתמודד עם מצבים בהם הדעתה אינה מושגת.

9.5.1 The Problem

התחום של למידה ממיינט דוגמאות (Few-Shot Learning) נוצר על מנת להתמודד עם מצב של חוסר איזון קיצוני בין כמות הדוגמאות של כל קטgorיה לאימון הרשות. באופן פורמלי, קיימות קטגוריות הנקראות קטגוריות בסיס (base classes), עבורן יש כמות גדולה של דוגמאות, ובנוסף ישן קטגוריות חדשניות (novel classes), עבורן יש כמות קטנה מאוד של דוגמאות. כדי להגדיר את היחס, משתמשים בשני פרמטרים: פרמטר k המיצג את מספר ה-shots, הכולמר מספר הדוגמאות הקיימות בסיס האימון מכל קטגוריה חדשה, ופרמטר n שמייצג את מספר הקטגוריות החדשניות הקיימות סך הכל. כל בעיה מוגדרת על ידי "k-shot n-way learning" (ולמשל "k-shot 5-way learning" מתייחס לבן יש חמישה קטגוריות חדשות, ומכל אחת מהן יש רק דוגמא אחת לאימון הרשות. בכלל, בקטגוריות הבסיס תהייה כמות גדולה של דוגמאות. למשל בסט התמונות האופני לעיבוט藻, mini-ImageNet, יש 600 דוגמאות לכל קטגוריה בסיס ולרוב 5-10 דוגמאות עבור הקטגוריות החדשניות).

האתגר בלמידה ממיינט דוגמאות נובע מה הצורך להכניס לרשות כמות ידועה קטנה נוספת על הידע הנרחב הקיים, תוך הימנעות מ-overfitting כתוצאה מכמות הפרמטרים הגדולה של הרשות לעומת כמות המועיטה של הדעתה. לכן, גישה נאיבית כמו אימון מחדש של רשות על מעט דוגמאות נוספת ליצור הטיה בתוצאות.

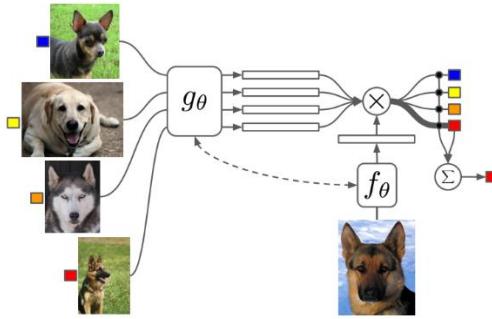
יש לציין כי בכל בעית הלמידה ממיינט דוגמאות עבור האימון, אך השאיפה היא להצליח באופן זהה בזיהוי כל הקטגוריות בשלב המבחן, בו לא יהיה חוסר איזון. لكن בעיות אלוロンיות לשימושים רבים כמו: זיהוי חיות נדירות באופן זהה לחיות יותר נפוצות, מערכת זיהוי טילים שצריכה להתמודד גם עם אינויים נדירים יותר (ניתן לחשב למשל על פצת אוטום), מערכות זיהוי פנים שצרוכות לעבוד טוב עבור כל אדם ללא תלות בדתתו שהיא קיימת באימון הרשות.

פרק זה נתאר את שלוש הגישות העיקריות לפתרון בעיות למידה ממיינט דוגמאות. עבור כל גישה נציג את האלגוריתמים המשמעותיים ביותר שנקטו בגישה זו. לאחרונה, מפותחים יותר וייתר אלגוריתמי למידה ממיינט דוגמאות שימושיים יחד רענון השאבים במספר גישות יחיד אך נשענים על האלגוריתמים המשמעותיים מהעברית. לבסוף, נציג את התחום של Zero-Shot Learning, כלומר יכולת למידה של קטgorיה חדשה כאשר לא קיימת אף דוגמא שליה לאימון.

9.5.2 Metric Learning

שיטות להתמודדות עם למידה ממיינט דוגמאות הנוקטות בגישת למידת מטריקה, שואפות לייצג את הדוגמאות כוקטוריים של מאפיינים מרוחב רב-ממד', כך שניתן היה למצאו בקהלות את השיר הקטגוריה של דוגמא חדשה, גם אם היא תהיה מקטגוריה חדשה. שיטות אלו מבוססות על עיקנון הגדלת המרחק בין יצוגים וקטורים של דוגמאות מקטגוריות שונות (inter-class dissimilarity), בד בבד שמרה על מרחק קטן בין הייצוג הווקטוריאי של דוגמאות מאותה הקטגוריה (intra-class similarity).

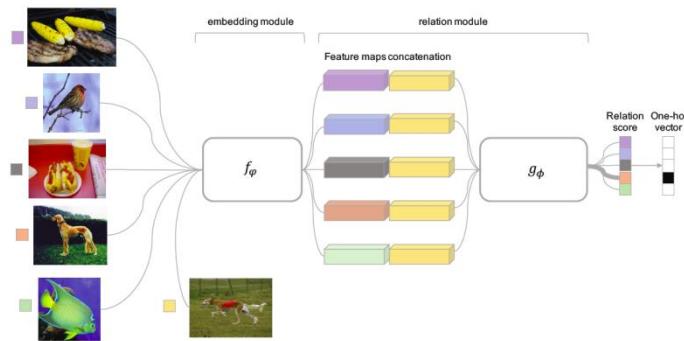
התקדמות משמעותית של שיטות אלו הוצאה במאמר Matching Networks for One Shot Learning בשנת 2016. שיטה זו משתמשת בזיכרון שהגישה אליו נעשית באמצעות מנגנון Attention, על מנת לחשב את ההסתברות של דוגמא להיות שייכת לכל קטגוריה, בדומה לשיטות השכן הקרוב (Nearest Neighbors).



איור 9.15 אילוסטרציה של שיטת Matching Networks

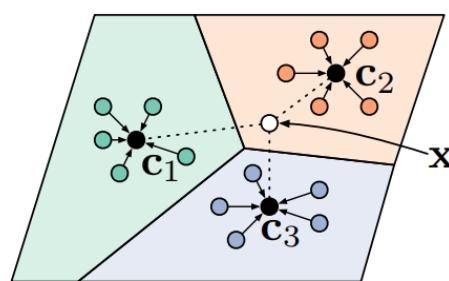
ההידוש המשמעותי בשיטת Matching Networks נועד בשיטת האימון המבוצעת באפיוזודות (Episodes). בשיטה זו האימון מכיל כמות של שימושים, כאשר כל שימוש היא למעשה מבחן של הדאטה שבו יש קטגוריות מסוימות שהן חדשות ואחרות מהן קטגוריות בסיס. על ידי דוגמאות רבות ויצירת שימושים מלאכותיים כאלה, בהן בכל פעם נלקחות קטגוריות אחרות ליצג את החדשניות, מתבצע אימון המתאים לבעה של מיעוט דוגמאות. שיטת אימון באפיוזדות הפכה לנפוצה ביותר בלמידה ממינית דוגמאות, גם בגין הatrixה ווגם בגישות שנראה בהמשך.

שיטות רבות מtabssoot על הרעיון של מאמר זה. למשל שיטת Relation Network משרשרת וקטורי מאפיינים של דוגמת מבחן לבין כל דוגמא של קטגוריות האימון. אלו ננסים למודל המשערך מdad דמיון בעזרתו ניתן לסואג את דוגמת המבחן.



איור 9.16 Learning to Compare: Relation Network for Few-Shot Learning

שיטה נוספת המשמשת נוספת הנקראת בגישה למידת מטריקה קראט Prototypical Networks. בגישה זו כל קבוצת דוגמאות של קטgorיה מסוימת במרחב וקטורי המאפיינים מקבלת נקודת אב-טיפוס אופיינית המוחשבת על ידי המוצע של הדוגמאות בקטgorיה זו. בכר מחשבים מסווג לנארה המפריד בין הקטגוריות. בעת המבחן נסואג דוגמא חדשה על סמך מרחק אוקלידי מנקודות האב-טיפוס.



איור 9.17 Prototypical Networks for Few-Shot Learning

בטבלה הבאה ניתן לראות השוואת ביצועים של שיטות למידת המטריקה שהזיכרנו על הקטגוריות החדשניות. יש להציג כי כל השיטות מגיעות לאחיזה דיק נמוכים משמעותית מאשר הדיק המדוחים במקרים של איזון בין כמות הדוגמאות בקטגוריות השונות (לרוב מעל 90% דיק).

Method	5-way 1-Shot	5-way 5-Shot
Matching Networks	46.6%	60.0%
Prototypical Networks	49.42%	68.20%
Learning To Compare	50.44%	65.32%

איור 9.18 השוואת ביצועי דיוק של שיטות למידת מטריקה על קטגוריות חדשות עבור mini-ImageNet.

9.5.3 Meta-Learning (Learning-to-Learn)

גישה שנייה להתמודדות עם מיעוט דוגמאות וחוסר איזון בין הקטגוריות נקראת מטא-למידה (או: למדוד איך למדוד). באופן כללי בלמידה מכונה, כאשר מדובר על מטא-למידה, מתכוונים לרשת שלומדת על סמך התוצאות של רשת אחרת. בלמידה ממיעוט דוגמאות הרוין הוא שהרשת תלמד בעצמה איך להתמודד עם מיעוט הדאטה על ידי עדכון הפרמטרים שלה לאופטימיזציה של בעיה של סיווג ממיעוט דוגמאות. לשם כך משתמשים באפיוזות של משימות למידה ממיעוט דוגמאות.

שיטה חשובה בגישה זו היא (MAML) Model-Agnostic Meta-Learning. בשיטה זו, שאינה מיועדת ספציפית לסיווג תמונות ממיעוט דוגמאות, בעזרת מספר צעדים מיטים בכיוון הגראדיינט ניתן למדוד את הרשות התאמת מהירה (fast adaptation) לשימה חדשה. כאמור, כל משימה באימון היא אפיוזה שבה קטגוריות מסוימות נבחרות רנדומלית לדוחות את הקטגוריות החדשיות. בכל משלים פרמטרים של המודל האגנוטטי כך שעದכון בכיוון הגראדיינט יוביל להתאמאה לשימה החדש. הכותבים מצינים שמנקודת מבט של מערכות דינמיות, ניתן להתבונן על תהליך הלמידה שלהם כenza שמנקסס את ריגושים פונקציית המחיר של משימות חדשות ביחס לפרמטרים. כאשר הריגשות גבוהה, שניוי פרמטרים קתנים יכולים להוביל לשיפור משמעותו במחair של המשימה. מתמטית, פרמטרי המודל, המיצגים על ידי θ , משתנים עבור כל משימה i להיות θ'_i , כאשר:

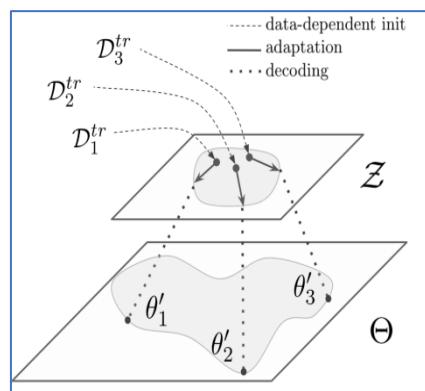
$$\theta'_i = \theta - \alpha \nabla_{\theta} L_{T_i}(f_{\theta})$$

עבור פונקציית מחיר L והיפרפרמטר α . כאשר מבצעים מטא-למידה לעדכון הפרמטרים, מחשבים למשה SGD:

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{T_i \sim p(T)} (L_{T_i}(f_{\theta}))$$

כאשר המשימותxDוגמאות מטור (T) $k - \beta$ הוא גודל הצעד של המטא-למידה.

שיטה מעניינת נוספת בשם (LEO) Latent Embedding Classification מימושת את הגישה של מטא-למידה במרחב ייצוג לטוני בעל ממדים נמוכים, ולאחר מכן עובר בחזרה למרחב המאפיינים הרוב ממד.



איור 9.19 שיטת (LEO) Latent Embedding Classification

השימוש במרחב מאפיינים בעל ממדים נמוכים המשמרים את המאפיינים החשובים לייצוג הקטגוריות, שיפור באופן ניכר את תוצאות הסיווג, כפי שניתן לראות בטבלה הבאה:

Method	5-way 1-Shot	5-way 5-Shot
MAML	48.7%	63.11%
LEO	61.76%	77.59%

איור 9.20 השוואת ביצועי דיק של שיטות מטא-למידה על קטגוריות חדשותיות עבור mini-ImageNet.

9.5.4 Data Augmentation

גישה שונה להטמודדות עם מיעוט דוגמאות נוקטת ביצירת דוגמאות כדי להימנע מהטיה. שיטות אוגמנטציה למשה יוצרות דאטה חדש על סמך הדאטה המקורי. השיטות פשוטות יותר מייצרות מהתמונה הקיימות תМОנות ראי, שינוי תאוריה וקונטרסט, שינוי סקללה, שינוי צוויות, ואף הוספת רעש רנדומלי. כל אלו הראו שיפורים ביכולות הרשותת ללמידה קטגוריות שהו במצב חוסר איזון. דרך נוספת היא שימוש ברטובות גנרטיביות (GANs) על מנת ליצור דוגמאות רלוונטיות, למשל דוגמאות של אותו האובייקט מזוויות שונות. שיטה מעניינת של אוגמנטציות היא CutMix, בה פאצ'ים של תМОנות נחככים ומודבקים בתМОות האימון וגם התוצאות מעורבבים בהתחם. שיטה זו הגיעה לביצועים מרשימים בסיווג תМОות וגם בזיהוי אובייקטים, ככל הנראה בגלל שהיא מאפשרת למודל להיות גנרי יותר בהתייחסות לחלקים שונים מהתמונה המשפיעים על הסיווג לקטgorיה.

9. References

Detection:

<https://arxiv.org/pdf/1406.4729.pdf>

Segmentation:

<https://arxiv.org/ftp/arxiv/papers/2007/2007.00047.pdf>

SegNet:

<https://arxiv.org/pdf/1511.00561.pdf>

<https://mi.eng.cam.ac.uk/projects/segnets/#demo>

<https://arxiv.org/pdf/1409.1556.pdf>

<https://arxiv.org/pdf/1502.01852.pdf>

Face recognition:

https://docs.opencv.org/master/d2/d42/tutorial_face_landmark_detection_in_an_image.html

<http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html>

10. Natural Language Processing

עיבוד שפה טבعتית (NLP) הוא תחום העוסק בפיתוח אלגוריתמים לעיבוד וניתוח של דата טקסטואלי. המטרה העיקרית היא לפתח שיטות ומודלים שיאפשרו למכונות "להבין" את התוכן הטקסטואלי, ואת הニアנסים והקשרים של השפה. עלם ה-NLP מורכב ממספר רב של תמי משימות, כגון ניתוח סנטימנט של טקסט (Sentiment analysis), תמצאות/סיקום אוטומטי של טקסט (Text summarization), מציאת תשובה עבור שאלות מושיות (Question answering) ועוד. יחד עם יכולות רבות אחרות. פיתוח כלים המסייעים להtauוד עם מושיות אלה יאפשר לנו (בין היתר) לפחות אפליקציות שימושיות כגון עזרות קוליות, מערכות תרגום, מערכות אוטומטיות לבדיקת דקדוק ועוד. כמה דוגמאות לאפליקציות אלה כולנו מכירים הן Siri, העוזרת הקולית של אפל, ההשלה האוטומטית בתיבת החיפוש ב-Google וGrammarly, מערכת לתיקון תחבירי לטקסט באנגלית.

10.1 Language Models and Word Representation

בפרקים הקרובים נדונם בשניים מהנושאים הבסיסיים ביותר בתחום ה-NLP – מודלי שפה ויצוג מילים: נראה כיצד מייצרים מודל שפה מדatta טקסטואלי (שנקרא גם "Corpus"), וכייד מיצגים את הטקסט כך שייהי אפשר לאמן בעזרתו מודל. כדי להקנות למcona יכולת הבנה של דата טקסטואלי יש לבנות מודל הסתברותי הקובע את התפלגות של המילים בשפה. המודל מנשה לכמת את הסיכוי להופעה של סדרות שונות של מילים, ובאופן יותר כללי הוא קובע מה הסתברות של כל רצף אפשרי להופיע. מודל זה מאפשר לבנות בעזרת אימון על גבי דата טקסטואלי, והוא נקרא "מודל שפה" (Language Model). לפני ביצוע האימון, יש לבצע מיפוי של הטקסט לייצוג מסויים (Word Representation), המאפשר למcona ללקחת את הדטה הקים, לעבד אותו ולנסות לבנות בעזרתו את מודל השפה.

ראשית נגידר מהו מודל שפה: מודל שפה הינו מודל סטטיסטי המגדיר התפלגות מעלה המרכיב מכל הסדרות האפשרות של מילים (כלומר משפטים, פסקאות וכדומה). מודל זה מקבל כקלט סדרה של מילים ותפקידו הוא לחזות מה היא המילה הבאה שתنبيיב את ההסתברות המרבית לרצף ביחיד עם המילה הנוספת.

כעת נתאר בצורה מתמטית מהו מודל שפה. נניח ונთן משפט עם n מילים: $[w_1, \dots, w_n]$, אז ההסתברות לקבל את המשפט זהה הינה:

$$P(w_1, \dots, w_n) = \prod_{i=1}^n P(w_i | w_1, \dots, w_{i-1})$$

ניקח למשל את המשפט הבא:

Take a big corpus

ההסתברות של משפט זה ניתנת לחישוב באופן הבא:

$$P(\text{Take}, a, \text{big}, \text{corpus}) = P(\text{Take}) \cdot P(a|\text{Take}) \cdot P(\text{big}|\text{Take}, a) \cdot P(\text{corpus}|\text{Take}, a, \text{big})$$

במילים, נוכל לתאר את המשמעות הזה כך: ההסתברות לקבל את המשפט הנתון שווה להסתברות של המילה להופיע כפולה ההסתברות של המילה a להופיע אחרי המילה big להופיע אחרי הצירוף Take a big , כפולה ההסתברות של המילה corpus להופיע אחרי הציגוף Take a big corpus .

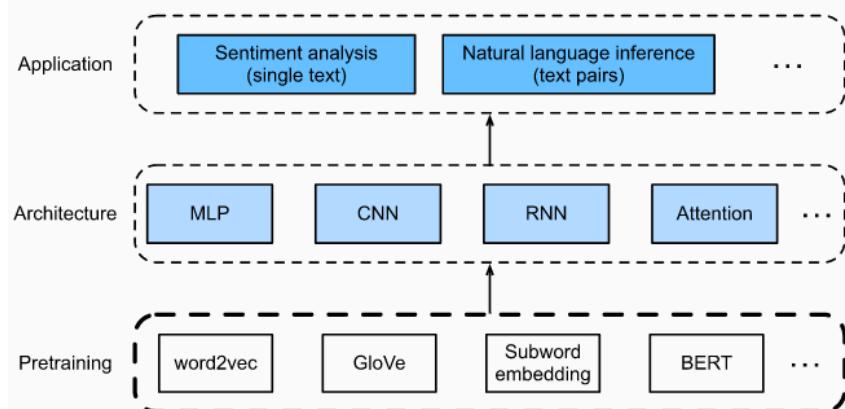
באופן הפנטני ביותר נוכל לשערך את ההסתברויות האלו באופן הבא: ניקח corpus מספק גדול ועшир בעל N מילים שונות, כמו למשל כל ערבי וקיידי, ופושט נספור את כל המילים והצירופים שמופיעים בו. ההסתברות של כל מילה תהיה אחות הפעמים שהיא מופיעה בטקסט, וההסתברות המותנית תחושב באופן דומה – על ידי מנתית מספר המופיעים של צירוף כלשהו וחלוקת במספר הפעמים שהמילה עצמה מופיעה. באופן פורמלי נוכל לרשום זאת כך:

$$P(W_i) = \frac{\#W_i}{N}, P(W_i | W_j) = \frac{\#W_i, W_j}{\#W_i}$$

בין אם נחליט ללקחת את מודל השפה זה או שנייך מודלים מתוחכמים יותר כדי שנראה בהמשך, המודל הוא אחד הדברים היסודיים ביותר במשימות שפה, כיון שבעזרתו ניתן לבצע מגוון משימות.

כאמור, במקרה שנווכל לבנות מודל שפה או לאמן כל מודל אחר, נצטרך קודם כל ליצג את הטקסט בצורה כלשהיא. בשיטה שהוזגה לעיל, כל מילה (Token) מיצגת באמצעות צירוף אותיות. כך למשל המילה הראשונה במשפט

שראינו מרכיבת מצירוף של האותיות e, k, a, T. כפי שיפורט בהמשך, נוכל גם לדבר על ייצוג למלילים עצמןvruck כר' שכל חידה אוטומטית תהיה מילה וצירוף של היחידות האלה ירכיבו ייצוג של משפט. באופן סכמטי, ניתן לתאר את משימת עיבוד השפה מקצתו להזזה באופן הבא:



איור 11.1 תהליך פיתוח אלגוריטם של מודל שפה: א. מייצגים את הטקסט בזורה כלשהו (ניתן מבון לקחת ייצוג קיימש שנבנה על בסיס דאטסה אחר). ב. מאמנים מודל שמקבל כתובות טקסט מסוימות יציגו אותה בדרכו כלשהיא בשלב הראשוני, והמודל מוציא output מסויים. למודל זה יכולות להיות ארכיטקטורות שונות. ג. באמצעות המודל המאומן ניתן לבצע משימות קיצה שונות.

בהמשך פרק זה נתמקד בשכבה התחכונה של התרשימים: נתאר מספר שיטות מרכזיות לייצוג טקסטים, ונראה כיצד ניתן לאמן מודלי שפה שונים היכולים לבצע כל מיני משימות.

10.1.1 Basic Language Models

מודל השפה הראשון אותו נציג הינו N-Grams, שהוא מודל סטטיסטי המניח שהסתברות למילה הבאה תליה ארורק ב- $1 - N$ המילים שקדמו לה בסדרה. הנחה זאת נקראת "הנחה מרקוב" (Markov assumption), ובאופן כללי יותר, מודלי מרקוב (או שרשרות מרקוב במקורה הדיסקרט) מסדר n הם מודלי הסתברות המניחים שנייתן לחזות הסתברות של אירוע עתידי T בהתאם על האירועים שהתרחשו ב- n קודמות הזמן שקדמו למועד T מוביל להתחשב באירועי עבר רצוקים מדי.

מודל-h-N-Gram הפשט ביוטר נקרא unigram. במודל זה אנחנו חוזים את המילה הבאה לפי התדריות של המילה עצמה ב-skipgrams מוביל להתחשב بما שקדם לה. מבון שהיחס כזה הינו בעיתוי, מכיוון שהמילה הבאה **חייבת להיות תליה במילים שקדמו לה**, וכך מילים כמו the שמופיעות באופן תדרי בטקסט שאין בהכרח משפיעות על ההקשר. לכן, נסתכל על מודל קצר יותר מרכיב הנקרא bigram, המתיחס למילה האחרונה הקודמת למילה הנחוצה. במודל bigram אנחנו מניחים את המשווואה הבאה:

$$P(w_n | w_{n-1}, w_{n-2}, \dots, w_1) = P(w_n | w_{n-1})$$

כלומר, רק למילה האחרונה יש השפעה על החיזוי של המילה הבאה, וכל המילים שלפניה הן חסרות השפעה על התפלגות של המילה הנחוצה (ונמיהר גם על המשפט המשפט). באופן כללי, מודל N-grams מניח את המשווואה הבאה:

$$P(w_n | w_{n-1}, w_{n-2}, \dots, w_1) = P(w_n | w_{n-1}, w_{n-2}, \dots, w_{n-N}), \quad n \leq N$$

ניקח לדוגמא מספר משפטים וננתן אותם בשיטת bigram:

- I know you
- I am happy
- I do not know Jonathan

נכיה ונרצה לבדוק את ההסתברות למילה Jonathan היא המילה הבאה אחרי הסדרה do I. ראשית נגדיר את המילון, המכיל את כל המילים האפשרות בשפה:

$$V = \{I, know, you, am, happy, do, not, Jonathan\}$$

כעת נוכל להעריך את ההסתברות לכך על ידי ספירת כמה הפעמים שהצמד ($know|Jonathan$) מופיע בטקסט, ולנורמל בכמות הפעמים ש- $know$ מופיע בטקסט עם מילה כלשהי (כולל הפעמים ש- $know$ עם $Jonathan$). באופן פורמלי נגידר את המשוואה הבאה:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_{w \in V} C(w_{n-1}w_n)}$$

כאשר האות C מסמנת את מספר הפעמים שצמוד מסוים בטקסט. ניתן לשים לב שהביטוי במכנה למעשה סופר את כמה הפעמים ש- w_{n-1} קיימ בטקסט, וכך נוכל לפשט את המשוואה האחורונה ורשום במקומה:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

אם כך, ההסתברות לכך שהמילה $know$ תופיע אחרי המילה $know$ היא:

$$P(Jonathan|know) = \frac{1}{2}$$

באופן דומה ניתן לנסוט לשערך את ההסתברות של המילה הבאה על סמך יותר ממילה אחת אחרת, למשל לחתות 2 מילים אחרת. מודל זה נקרא *trigram*, כפי שנאמר לעיל, באופן כללי מודל המסתמן על $1 - N$ מילים לצורך חישוב ההסתברות של המילה הבאה בטקסט נקרא *N-gram*.

המודל המركובי סובל ממספר בעיות:

1. טבלת ההסתברויות המתקבלת מאוד דיליה. כיוון שה-*n-grams* (שהוא סט האימון) הינו בגודל מוגבל, לעומת זאת נוכל לראות את כל הקומבינציות הקיימות, מה שיוביל להערכתה של הסתברות 0 עבור צמדים שלא מופיעים בטקסט האימון.

2. כאשר נרצה לחזות בעזרת מודל זה את ההסתברויות על טקסט חדש, כנראה שנתקל במילים שלא נתקלנו בהן בטקסט האימון וכן לא נוכל להגיד את ההסתברות עבור *-N-grams* המכילים מילים אלו.

בשביל להתמודד עם בעיות אלו – באופן מלא או חלק – נוכל להפעיל שיטות החלקה (Smoothing) על ההסתברויות של צמדים שהופיעו מעט/כלל לא הופיעו בטקסט האימון. שיטות החלקה במהותן לוקחות קצת מסת הסתברות מהאירועים השכיחים (כלומר, *-N-grams* המופיעים כי הרבה) ו"טורמות" לאיירועים פחות שכיחים. ישנן מגוון שיטות להחלהת הסתברות ונסקור כעת חלק מהן.

Laplace Smoothing

הדרך הפשוטה ביותר לבצע החלקה היא להוסיף מספר קבוע \mathcal{K} לכל האירועים. באופן זהה אנו דואגים למתן משמעות גם לצירופים נדירים שמאפיעים מספר מועט של פעמים (או אפילו לא מופיעים בכלל). אם למשל יש צירוף שמאפיע רק פעם אחת ולעומתו יש צירוף שמאפיע 1000 פעמים, אז הוספה הקבוע \mathcal{K} משמעותה שעכשו נמינה את הצירוף הראשון ככזה המופיע $(\mathcal{K} + 1)$ פעמים וביחס לצירוף השני נתיחס ככזה שהופיע $(\mathcal{K} + 1000)$ פעמים. הוספה זו כמעט ואינה משפיעה על הצירוף השני, אך היא יכולה להכפיל פי כמה את ההסתברות של הצירוף הראשון. כמובן שכאשר אנחנו מתעסקים עם הסתברויות נרצה לאחר הוספת הקבוע במונה לתקן גם את מקדם הנורמל. באופן פורמלי, החלקת לפסל מוגדרת באופן הבא:

$$P_{Laplace}(w_n|w_{n-1}) = P_{Add-\mathcal{K}}(w_n, w_{n-1}) = \frac{C(w_{n-1}w_n) + \mathcal{K}}{\sum_w C(w_{n-1}w_n) + \mathcal{K}} = \frac{C(w_{n-1}w_n) + \mathcal{K}}{C(w_{n-1}) + \mathcal{K} \cdot |V|}$$

כאשר $|V|$ הוא גודל המילון (כמות המילים המופיעות ב-*n-grams*). היתרון בשיטה זאת היא הפשטות שלה, אך עם זאת יש לה חסרן בולט הנובע מכך שהיא משנה באופן משמעותי משמעותי את ההסתברויות של מאורעות תדיירים ובכך בעצם משבשת את ההסתברויות שנלמדות מהtekst. כפועל יוצא יותר מדי מסת הסתברות עוברת מהמאורעות השכיחים למאורעות עם הסתברות נמוכה. השיטה הבאה מנסה לתקן בדיק את זה.

כਮון שניתן לבחור כל ערך \mathcal{K} שרצים ואותו להוסיף למיניה. באופן זה כן ניתן לשולוט בrama מסויימת עד כמה להגדיל את ההסתברויות לקומבינציות שאינן מופיעות בסט האימון על חישוב הקומבינציות השכיחות (כתלות-ב- \mathcal{K}). בחירה למשל של $\mathcal{K} = 0.5$ מאפשרת לפחות את העיונות יכול להתקבל משינוי הספרה.

Backoff and Interpolation

שיטת החלוקת בסעיף הקודם נותנת פתרון לקביעת הסტברות של מאורעות המקבילים הסტברות 0, וישן גישות נוספת לשונייה מענה למוגבלות האלה. נניח ואנחנו משתמשים במודל trigram, מודל המניח שהסתברות למילה הבאה תלולה בשתי המילים שבאותה לפניה. אם נבצע את השערוך של כל שלישיה לפי הגדרה (=ספירת המופיעים שלהם בטקסט), כל שלישית מילים שאינה מופיעה כרץ בטקסט תקבל הסტברות 0, ובuczם תקיים את המשוואה:

$$P(w_n | w_{n-1}, w_{n-2}) = 0$$

בכדי להימנע מלחת הסტברות 0 בaczא מצב, ניתן להיעזר גם בהסתברויות של bigram וה-unigram:

$$P(w_n | w_{n-1}), P(w_n)$$

שיטת backoff מציעה לחת את כל המקרים בהם קיבלנו 0 ולשערך אותם מחדש באמצעות מודל bigram. לאחר מכן את כל המילים שעדיין נותרו עם הסტברות 0 (כלומר, צמדי מילים שאין מופיעים בטקסט) ונשערך אותם באמצעות unigram. באופן זה מקווים להגיע לפחות ממספר המילים בעלי הסტברות 0 היא קטנה (עד אפסית), כדי ששיעור השימוש במודלים יותר פשוטים עשוי שימוש מגוון רחב יותר של קומבינציות הקיימות בטקסט. שיטה דומה נקראת Interpolation, ובזה במקומם לחת את הרצפים בעלי הסტברות 0 ולשערך אותם במודל הנמצא בדרגה אחת מתחת (למשל – לשערך בעזרת bigram במקומם trigram), המודול הראשון מתיחס לקומבינציה כלשהי של המילים – למשל קומבינצייה ליניארית). בשיטה זו גם מקרים שאינם בעלי הסტברות 0 נעצרים ב-unigram, bigram and trigram ב-gram unigram לשערוך ההסתברות ובניית מודל השפה.

10.1.2 Word representation (Vectors) and Word Embeddings

עד כה, המילים השונות היו מייצגות בעזרת אותיות. כך לדוגמה, המילה כלב תוצג על ידי צירוף האותיות DOG בעוד שהמילה חתול תוצג על ידי הצירוף CAT. יציג זה מכיל מאפיינים סינקטטיים (=תחביריים) של השפה, קרי אין המילה נכתבת. עם זאת, יציג זה חסר מאד, כיון שהוא יתקשה בלמידת יציג של מאפיינים סמנטיים. דוגמא להבנה – סמנטית של שפה היא ההבנה שכלב וחתול הן לא מילים בלבד, אך הן כן קשורות אחת לשנייה באופן כלשהו – שתיהן מייצגות חייה מחמד שאנשים מגדלים בדירותם. מודל המבוסס על יציג טקסטואלי של השפה לא יכול להציג להבנה סמנטית שלה, ולכן נרצה שהייצוג שלנו יהיה מספיק עשיר ויכיל הן מאפיינים תחביריים והן מאפיינים סמנטיים של השפה.

בפועל, נוכל למפות את הייצוג הטקסטואלי לייצוג נומרי בצורה פשוטה בעזרת One-Hot vectors – מערכת בגודל המילון שלנו, המציג כל מילה במלון בעזרת 1 באיבר המתאים במערך. לדוגמה נתון המילון הבא:

Index	Word
0	Dog
1	Cat
2	Lion

נוכל לייצג את המילים השונות בעזרת וקטוריים באורך 3, באופן הבא:

$$\text{Dog} \rightarrow [1, 0, 0]$$

$$\text{Cat} \rightarrow [0, 1, 0]$$

$$\text{Lion} \rightarrow [0, 0, 1]$$

כך, לכל מילה יהיה יציג וקטורי ייחודי. עם זאת, יציג פשטני זה הינו בעיתי מכיוון שהיא קשה ללמידה ממנו מאפיינים סמנטיים. כדי להבין את הסיבה לכך רצוי שלהגדיר את מושג הדמיון בעולם של וקטורים.

Cosine similarity

מעבר לייצוג וקטורי של מילים דרוש מאייתנו להגדיר דמיון בין וקטורים למרחב שנוצר. אחת מההגדרות הפופולריות לדמיון בין וקטורים היא ה-Cosine similarity. כמו רוב השיטות לחישוב דמיון בין וקטורים, גם פונקציית דמיון זו מבוססת על מכפלה פנימית של וקטורים. נניח וקטורים w, v , שניהם בעלי אותו הממד N . המכפלה הפנימית (dot product) בין הווקטורים האלה מוגדרת באופן הבא:

$$v \cdot w = v^T w = \sum_{i=1}^N v_i w_i$$

באמצעות הגדרה זו ננסה לאמוד את הדמיון בין הייצוג של המילים כלב וחתול במרחב הוקטורי שנוצר בעקבות ייצוג בעזרת One-Hot vectors. כאמור, הייצוג של המילה חתול במרחב זה הוא: $[0, 1, 0] \rightarrow [0, 1, 0]$, בעוד שהייצוג של המילה כלב באותו מרחב הינה: $[0, 0, 1] \rightarrow [1, 0, 0]$. לכן, לכן המכפלה הפנימית במרחב זה תהיה:

$$[0, 0, 1] \cdot [1, 0, 0] = 0 \cdot 0 + 0 \cdot 1 + 1 \cdot 0 = 0$$

תוצאה זו מדגימה את הביעיותו בייצוג פשוטי זה, מכיוון שהimson רוצים שוואקטורים אלה כן יהיו דומים במובן מסוים, ולא שהחטואה תהיה 0, המלמדת על כך שככל אין קשר בין שתי המילים. למעשה בשיטה זו הדמיון בין ייצוגים של כל זוג מילים יהיה אפס.

Bag-of-words

דרך אחת לייצר קשר בין מילים בעלות סמנטיקת הדמיון לא רק למילים בודדות אלא גם להקשרים בתוך המשפט עצמו. באופן זה נוכל להגיד ייצוג וקטורי של מילה על ידי ספירה של כמות הפעמים שלמילהachaרבת נמצאת אליה באותו הקשר. שיטה זאת נקראת Bag-of-Words, ולפניהם נוכל להציג אותה, נctrar להגדיר מהו הקשר של מילה. באופן פשוט הקשר של מילה זה המשפט בו היא מופיעה, אך ניתן גם לנקוט רק חלון של מספר מילים מתוך המשפט (לרוב אורך החלון קטן מאורך המשפט). לדוגמה, נניח ונ נתונים לנו הטקסט והmillion הבאים:

טקסט:

- [The dog is a domestic mammal, not wild mammal], is a domesticated descendant of the wolf, characterized by an upturning tail.
- [The cat is a domestic species of small mammal], It is the only domesticated species in the family.

million:

1. The	5. Mammal	9. Animal	13. Tail
2. Is	6. Not	10. Descendant	14. Cat
3. A	7. Natural	11. Dog	15. Species
4. Domestic	8. Wild	12. Wolf	16. Small

כעת נרצה לייצג את המילים Dog ו-Cat בשיטת Bag-of-words. חלון זה מסומן בטקסט בסוגרים מרובעות בצלע אדום. נבנה מטריצה עם מספר עמודות כאורך המילון ומספר שורות כמספר המילים אותן נרצה לייצג (לרוב מספר השורות יהיה כמספר המילים במלון, אך לשם הדוגמא נציג כאן טבלה קטנה יותר). עבור כל מילה, נבדוק כמה פעמים היא נמצאת בטקסט באותו חלון יחד עם מילים אחרות:

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Dog	1	1	1	1	2	1	1	1	0	0	0	0	0	0	0	0
Cat	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1

כעת נוכל לראות שהמכפלה הפנימית בין שני הווקטורים המייצגים את המילים Dog ו-Cat אינה 0. עם זאת, ישנו שתי בעיות נוספתש הנובעות מפשטות פתרון זה:

1. מילים פחות משמעותיות כמו is, the, to, a, in ועוד, נכללות בספירה למرات שאין לה בהכרח מושיפות מידע משמעותי לייצוג.
2. מילים המופיעות בטקסט לעיתים רחוקות יהיו בעלות ייצוג וקטורי מאד דיליל, מה ש慷慨דיל שוב את הסיכוי למכפלה הפנימית בעלת ערך קטן מאוד (עד אפס) עם רוב הווקטורים האחרים.

TF-IDF

הדרך הטבעית לפתרון של הבעיה הראשונה – רוש שנווצר ממילאים בעלות תדירות גבוהה שאינן תורמות בייצוג, היא סיכון של המילים האלה מהילוֹן. אולם, פתרון זה אינו יעיל, כיון שהתדרות של מילים משתנה בין תחומיים/דומיניים שהם נלקח הטקסט. לכן פותחה שיטת ייצוג הנקראת TF-IDF, ומטרתה להפחית את רוש זה באופן אוטומטי.

בשיטת TF-IDF, מגדירים פונקציה ניקוד אחורית בעלת שני רכיבים. במקרה להסתכל על חלון יחיד מסביב לכל מילה, ניתן להסתכל על כל המילים בחולנות הנוצרים מסביב למילה המייצגת, ולתת לה ניקוד לפי התדרות של אותה מילה. באופן פורמלי, tf מוגדר בצורה הבאה:

$$tf = \log(C(w, c) + 1)$$

משמעות הביטוי היא שאנו סופרים כמה המילה c מופיעה בקונטקסט (=בחילוֹן) של המילה המיוצגת w (על התוצאה מפעלים \log בשבייל rescaling של ערכים גבוהים מאוד).

עד כה השיטה אינה שונה במהותה מספירה כמו שראינו בסדום-words, אך מה שעשו את TF-IDF שונה הוא הביטוי השני. הביטוי idf מוגדר בצורה הבאה:

$$df = \text{count}(w \in \text{Context})$$

$$idf = \log\left(\frac{N}{df}\right)$$

המונח df מייצג את כמות הפעמים שהמילה w מופיעה בקונטקסטים אחרים, בעוד N מייצג את כמות המילים במילוֹן. נשים לב שגם מילה מסוימת, למשל the, מופיעה בכל הקונטקסטים של כל המילים במילוֹן, אך הביטוי בתוך הלוג יהיה 1 וכן ה- idf יהיה 0. לעומת זאת, אם מילה מסוימת מופיעה אך ורק בקונטקסט אחד, אז הערך שהוא בתוך הלוג יהיה N .

לבסוף, TF-IDF מוגדר באופן הבא:

$$TF - IDF = tf \cdot idf$$

מדד משקל זה מציין עבור ייצוג של מילה מסוימת לתחם משקל גדול למילים אחרות הנמצאות אליה בקונטקסט באופן תקין אך אין נמצאות בקונטקסט של מילים אחרות.

PPMI - Positive Pointwise Mutual Information

כעת נרצה לפתור את הבעיה השנייה – בעית הייצוג הדليل למילים שאינן תדרות בטקסט. שיטת PPMI מגדירה פונקציית ניקוד המחשבת את היחס בין הסיכוי של שתי המילים להימצא יחד לעומתון בנפרד – $\frac{P(x,y)}{P(x)P(y)}$. כעת בשבייל לחשב את הערך של התא המייצג את המילה y בוקטור הייצוג של המילה x השתמש בהסתברות הנ"ל ונפעיל לוג. אם ההסתברות לראות את המילים x, y ביחד שווה לכפל ההסתברויות לראות כל אחת לחוד, נקבל שערך הביטוי הוא $\log(1)$ כלומר 0. לעומת זאת, אם הסיכוי לראות את המילים הללו ביחד גדול מהסתוכי שיראו אותם לחוד אז נקבל ערך גדול מ-1. ישנו מקרה נוסף, בו הסיכוי לראות את הביטויים ביחד קטן מהסתוכי לראות אותם לחוד. במקרה זה הביטוי שנתקבל יהיה קטן מ-1 וכאן הלוג יהיה שלילי, אך מכיוון שהערכים החלקיים נוטים להיות לא אמינים אלא אם הטקסט שלנו גדול ממספריק), נוסף עוד אלמנט קטן לפונקציית החישוב:

$$PPMI = \max\left(\log\frac{P(x,y)}{P(x)P(y)}, 0\right)$$

בפועל זה נוכל לנורמל את הערך הנמוך עבור מילים נדירות בטקסט.

Word2Vec

השיטות שראינו עד כה לחישוב וקטורי הייצוג של המילים מאפשרות לנו לקודד מאפיינים סמנטיים בייצוג המילים. עם זאת, יש כמה חסרונות לשיטות אלו: ראשית, הן יוצרות וקטורים מאד דילילים, ובנוסף לכך גודל הווקטורים תלוי בכמות המילים שיש לנו במילוֹן, מה שיוצר וקטורים גדולים שמכבידים על החישובים במשימות השפה השונות. למשל – ראיינו קודם שנדעתן ליציג מילה באמצעות וקטור שכלוֹן אפסים למעט תא אחד עם הערך 1 במקומות ייחודיים למילה. עבור שפה עם אלפי מילים ואף יותר מכך, כל וקטור המייצג מילה הוא באורך עצום, ועם זאת הוא מאד דיליל כיון

שיש בו רק מספר תאים מועט שערכם שונה מ-0. לכן, נרצה לפתח שיטה שתיצור וקטורי ייצוג דחוסים (dense) בעלי ממד קטן יותר.

שיטת Word2Vec הינה שיטת Self-Supervised שפותחה למטרת יצרה של וקטורי ייצוג דחוסים של מיללים. הפרודיגמה של למידה מונחית Self-Supervised learning ("דונה" ללמידה מונחית (supervised learning) רגילה, אך התוצאות אינם נתונים אלא נוצרים באופן אוטומטי מתוך הדאטה ללא מתואג. באופן זה ניתן לאמן מודלים עם כמות גדולה של דאטה לא מותיג בצורה עיליה ולא צריך בתיאוג (שעלול להיות מודע לך). בהקשר זה, אלגוריתם Word2Vec משתמש ב-SGNS – Skip-Grams-With-Negative-Sampling Self-supervision באופן שומרת מה ההסתברות של מילים שונות להיות בكونטקסט של מילה נתונה. בסוף הוא להגדיר בעיית סיווג שומרת מה ההסתברות של מילים שונות להיות בكونטקסט של מילה נתונה. בכך האימון לוקחים את המשקלים שנוצרו בעקבות תהליכי אימון המשימה הראשית, והם יהיו הייצוג של המילה. בכך האימון שלנו:

Folklore, legends, myths, and fairy tales have followed childhood through the ages.

ראשית נקבע את אורך החולון (=מספר המילים עליהם מסתכמים בסביבות כל מילה) – 3. בעת עברו על הטקסט וניצור תיוגים בין כל מילה במלון ליתר המילים. למשל עבור המילה tales נוסף לדאטה סט שלנו דוגמאות חיוביות של המילים שנמצאות בكونטקסט עם tales. בנוסף, במקרה התנונות של כל הייצוגים לוקטור בודד, נדרש "להראות" למודל איך נראות דוגמאות שליליות וכן נשתמש בשיטה הנקראת negative sampling. בשיטה זו דוגמים מהמלון בהסתברות פרופורציונלית לתזדיות המילה (עם תיקון קטן שנตอน קצת יותר סיכוי למילים נדירות) את המילים ששימשו אותנו כדוגמאות שליליות. הרעיון מאחורי תהליך זה הוא שכאשר יש לנו מיליון גדול המילים שנגריל לא יהיו קשורות למילה שעבורה אנחנו יוצרים את הדוגמאות שליליות.

Folklore, legends, [myths and fairy [tales] have followed childhood] through the ages.

word	context	Label
tales	myths	+
tales	and	+
tales	fairy	+
tales	have	+
tales	followed	+
tales	childhood	+
tales	great	-
tales	April	-
tales	the	-
tales	young	-
tales	orphan	-
tales	dishes	-

כך נוכל ליצור מתואג עבור משימת הסיווג, ונוכל להשתמש בתיוגים אלה בשביל לאמן את המודל. הכוונה במשימת סיווג בהקשר זה מעט שונה מסיווג במובן הפשוט של המילה: מטרת המודל היא שבהינתן מילה *ש* (במקרה שלנו), נרצה שהיא ייצוג של מילים המופיעות באותו קונטקסט עם כל מילה (במקרה שלנו – אלו שופיעו עם

(tales באותו חלון) יהיה קרוב לייצוג של tales בעוד שמלים שאינן מופיעות באותו קונטקסט יהיו בעלות ייצוג "שונה" (mbhinit פנימית או cosine similarity). נניח שהחරנו שהייצוג של כל מילה יהיה וקטור בגודל 100. נתחיל את התהיליך כך שכל מילה מקבלת וקטור רנדומלי. נסמן את וקטורי הייצוג של המילה w - e_w ואת הווקטור של מילת קונטקסט c - e_c . השאייפה היא שהמכללה הפנימית של וקטורי הייצוג של המילים בkontekst של w יהיה גבוהה יחסית, בעוד שהמכללה הפנימית של וקטורי הייצוג של מילים שאינן מופיעות באותו kontekst יהיה נמוך. כאמור לעיל, Cosine similarity (המטריקה המגדירה דמיון בין וקטורים) היא בעצם מכפלת פנימית של הווקטורים (=מכפלת dot עם נרמול). כתע נוכל להגיד בעית logistic regression באופן הבא:

$$p(+|w, c) = \sigma(e_w, e_c) = \frac{1}{1 + e^{-e_w \cdot e_c}}$$

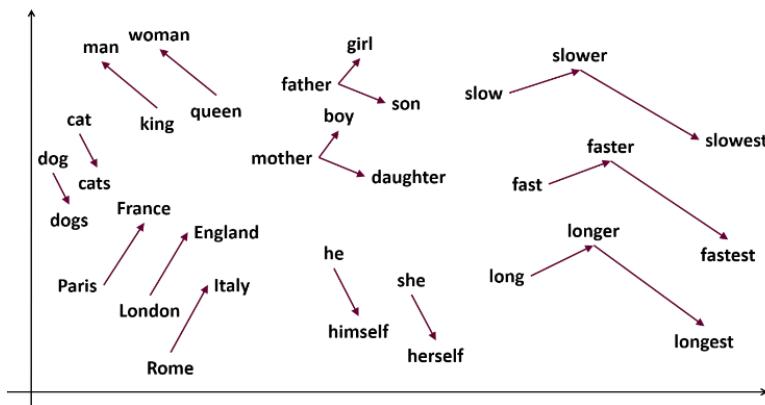
$$p(+|w, c) = 1 - p(+|w, c)$$

ובהתאם לכך פונקציית המטרה (Loss) תוגדר באופן הבא:

$$\begin{aligned} L &= -\log[p(+|w, c_{pos}) \prod_{i=1}^k p(-|w, c_{neg_i})] \\ &= -[\log(p(+|w, c_{pos})) + \log(\prod_{i=1}^k p(-|w, c_{neg_i}))] \\ &= -\left[\log(\sigma(e_w \cdot e_{c_{pos}})) + \sum_{i=1}^k \log(1 - \sigma(e_w \cdot e_{c_{neg_i}}))\right] \end{aligned}$$

נשים לב שאנו מניחים כי תלות בייצוג של הדוגמאות השליליות. בכך שנבצע מינימיזציה לפונקציית מטרה זו נגרים למכללה הפנימית בין הייצוג של מילה לבין מילת הקונטקסט להיות גבוהה ובו בזמן למכללה הפנימית בין וקטורים שאינם בkontekst להיות נמוכה. וכך הייצוג של המילה tales יהיה "דומה" (קרוב במנוחים של similarity cosine) לייצוג של המילים בkontekst ושוונה מיצוגן של המילים שאינם בkontekst. את התהיליך המינימיזציה במשך האימון נוכל לבצע באמצעות stochastic gradient descent.

אחת התוצאות היפות והחשובות של שיטת Word2Vec ניתנת להמחשה על ידי פריסת וקטורי הייצוג במרחב נמוך. בעזרת שיטות מתקדמות להורדת ממד (כפי שהוסבר בהרחבה בפרק 2), ניתן לצייר בדו-ממד או תלת ממד את וקטורי הייצוג של המילים לאחר האימון.



איור 11.2 וקטורי הייצוג של מילים לאחר ביצוע embedding באמצעות word2vec. צמד מילים בעלי משמעות דומה מוצגים על ידי וקטורים באותו כיוון.

נוכל לבדוק אם המרחב הווקטורי מקיים מאפיינים סמנטיים. לדוגמה, ניתן לראות שהווקטור המחבר בין וקטורי הייצוג של המילים King, Man מקביל ובעל אורכו דומה לווקטור המחבר בין הייצוג של Queen, Woman. דוגמה נוספת – הווקטור בין שם של ארצות הברית לביירה שלו מקביל ובעל אורכו דומה לווקטור שבין ארצות הברית ועיר הבירה המתאימה. כמובן שניתן למודד מכך על קשרים סמנטיים, כמו למשל שהיחס בין King, Man זהה לזה שבין Queen, Woman.

10.1.3 Contextual Embeddings

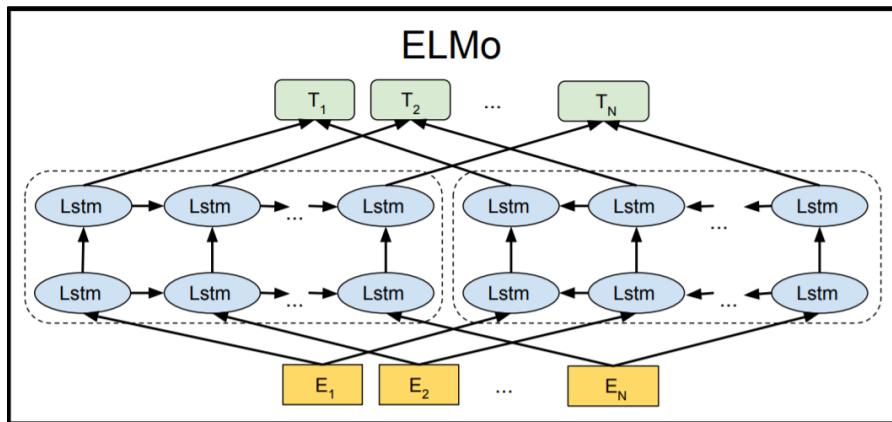
מנגנון בנייה יציגי מילים (embedding) שראינו עד כה למדו ייצוג סטטי עבור כל מילה. אך דבר זה יכול להיות בעיתוי מסוון לשפה טבעית היא דינמית ותלויה בקשר, ולאותה מילים יכולה להיות ככמה פירושים. בשיביל להבין את הבעייתיות נסתכל על המשפטים הבאים:

1. We need to **book** the flight as soon as possible
2. I read the **book** already

למילה **book** יש כפל משמעותים המשפטים הבאים. במשפט הראשון המילה משמשת כפועל ובמשפט השני עצם עם תפקיד סמנטי שונה במשפט. אם כך ברור שההקשר שבו המילה מופיעה משפיע על המשמעות שלה אך מוגנוינה כמו **word2vec** כמו **ELMo** יציג את המילה באותו וקטור יציג עבור שני המפעעים. لكن נרצה לפתח מנגנון **embedding** עבור המילים, שבמשמעותו וקטור המיצג מילא יהיה תלוי בהקשר בו היא מופיעה.

Embeddings from Language Models (ELMo)

אחד השיטות הראשונות שהציגה טכניקה למידת ייצוג תלי הקשר למילים הינה **Embeddings from Language Models**, או בקיצור **ELMo** – ארכיטקטורה הבונה לכל מילה ייצוג שתלי בהקשר שללה בתוך המשפט. הרעיון במודל זה הוא ללקחת ייצוג של מילה, להוסיף לו מידע נוסף מההקשר של המילה במשפט ולקבל ייצוג חדש התלי גם בהקשר שלו. בניסוח אחר ניתן לומר **ELMo** הינה פונקציה המתקבלת משפט שבו כל מילה מיוצגת בדרך כלל לשאה (למשל – **Word2Vec**), ומוסףיה ליצוג זה גם את ההקשר של המילה בתוך כל המשפט. בפועל זה נעשה על ידי מושימת אימון מודל שפה דו-כיווני – המודל לומד לחזות גם את המילה הבאה בטקסט וגם את המילה הקודמת, ובכך הוא לומד לתת למילה גם את ההקשר שלה. ארכיטקטורת הרשת נראה כך:



איור 11.3 ארכיטקטורת **ELMo**. הקלט הינו משפט המיצג כלשהו, והפלט הוא אותו משפט אך כל מילה קיבלה מידע נוספת על ההקשר שלו וכעת מיוצגת באופן חדש. תהליך האימון והוספת ההקשר בין המילים נעשו באמצעות שכבות של רכיבי **LSTM**.

כפי שמתואר בפרק 6.2.1, כל בלוק של **LSTM** מקבל קלט שני רכיבים המיצגים את ההיסטוריה של המשפט עד הנקודה בה מופיעה המילה של timestamp t (c_t, h_t), וקלט נוסף של האיבר הנוכחי בסדרה, שבמקרה שלנו זה המילה הנוכחיית (x_t). המוצא של **h-LSTM** הינו ייצוג חדש המשקיל את רכיבי ההיסטוריה יחד עם הייצוג הנוכחי של המילה.

בדומה לשיטות אחרות לייצרת ייצוג וקטורי למילים, אנחנו מאמנים את המודול בעזרת מושימת מידול שפה וחוזים את המילה הבאה בהינתן המילים הקודמות. אך בשונה מאלגוריתמים אחרים, **ELMo** משתמש בארכיטקטורת דו-כיוונית, כך שבתהליך האימון משלבת משימת שפה נוספת המנסה לחזות את המילה הקודמת בהינתן הסוף של המשפט. הארכיטקטורת של **ELMo** בנויה ממספר שכבות של **LSTM** שמורכבות זו על גבי זו, ולפי כתבי המאמר השכבות הת酣נות מצלחות ללמידה פיצ'רים פשוטים (למשל מאפיינים סינטקטיים למיניהם), בעוד שהשכבות העליונות לומדות פיצ'רים מורכבים (למשל מאפיינים סמנטיים, כמו משמעות המילה בהקשר).

לאחר תהליכי האימון ניתן להקפיא את הפרמטרים של המודול ולהשתמש בו עבור מושימות אחרות. הכותבים מציעים לשרשר את הייצוג הווקטורי של **LSTM** בכל שכבה ככה שיכיל אינפורמציה גם מתחילת המשפט עד המילה הנבדקת וגם מסוף המשפט עד המילה הנבדקת. מה שקרה בפועל זה שהשכבות החבויות (hidden layers) של **LSTM** הם עצם מהווים את ייצוגי המילים בשיטת ייצוג צוז, כלומר כל מילה במשפט מיוצגת על ידי התא המקביל בשכבה ה-**LSTM** שמעליה. בנוסף הם מוסיפים מספר פרמטרים קטן שמאפשר ציול (Fine tune) עבור מושימה ספציפית. כך לדוגמא יוכל להתאים את הייצוג של המילים למשימת סיוג של משפט לעומת משימת תיאוג של ישויות במשפט.

פה חשוב להזכיר נקודת מרכזית – בסופו של דבר התוצר של **ELMo** הינו **מודל שפה הлокח ייצוג של טקסט והוא قادر ליצוג תלי הקשר**. שכבות **-LSTM** השונות מאمدنות מודל שפה על מנת ליצור ייצוג חדש עבור המילים,

המתיחס גם להקשר. לאחר סיום האימון של מודל השפה (pre-training), ניתן לנקח אותו ולבצע transfer learning, כלומר להשתמש ביצוגים שהוא מפיק גם למשימות אחרות על ידי הוספת שכבות בקצה. לאחר פרסום המאמר, Sebastian Ruder (חוקר NLP מפורסם) טען כי:

"It is very likely that in a year's time NLP practitioners will download pretrained language models rather than pre-trained word embeddings"

כמובן, כתת מי שירצה לבצע משימת שפה כבר לא יתבסס רק על ייצוג סטטי של המילים אלא הוא יסתמך על מודל שפה מאומן שיאפשר לנקח ייצוג התחלתי של מילים ולהפוך אותם ליצוגים קונטקטואליים. עם עודamarim רבים אחרים אימנו מודלי שפה מאומנים שניים לנקח אותם ולהשתמש בהם עבור משימות קצה שונות על ידי הוספה של כמה שכבות וכייל המודל.

Bidirectional Encoder Representations from Transformers (BERT)

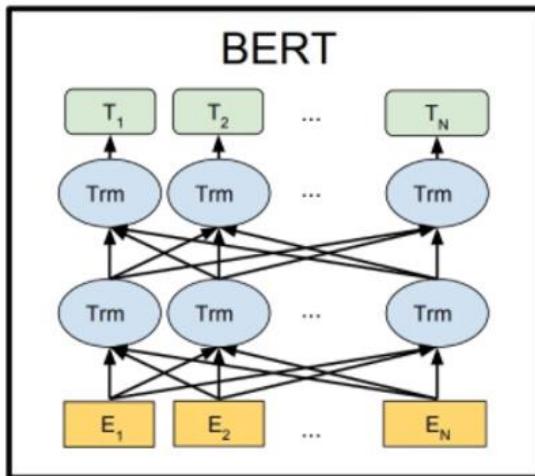
כאמור, הרעיון של ELMo הוא לייצר contextual embedding, ובכך לקבל מודל שפה המאפשר לנקח ייצוג וקטורי של מילים ולהעшир אותו במידע על הקשרן של כל מילה בטקסט. למרות-ShMo מעתה בשני היכיונים של המשפט (חוזה את המשך המשפט מתחילה ואת תחילתו המשפט מסופו) הוא אינו לומד משנה היכיונים בתהילר אחד, אלא צריך לחלק את הלמידה לשני חלקיים שונים. בנוסף, כפי שהסביר בהקדמה לפרק 8, כאשר מתעניינים עם סדרות ארוכות של מילים, וקטור הייצוג של כל איבר נהיה בעייתי כיון שהוא מוגבל ביכולת שלו להכיל קשרים בין מספר רב של איברים. במילים אחרות, כאשר רוצים להוציא פוקטור הייצוג של מילה מסוימת קשור למילים רחוקות, אנו מלאצים את הייצוג "לזכור" מידע רב, אך הייצוג הוקוטורי של המילים ב-ELMo אינו מצליח לעשות זאת בaczora מספיק טוב. לכן, על אף הצלחת גישה זו במשימות שונות, היא עדין התקשה במשימות בהן נדרשת יכולת לנתח טקסטים ארוכים (כמו למשל משימה של summarization). בנוסף לכך, האלגוריתם 'יחסית איטי', כיון שככל פעם הוא מסתכל על מילה אחת בלבד.

בדי להתמודד עם בעיות אלו וליצור ייצוג המסוגל להכיל מידע איקוני גם ברכפים ארוכים, ניתן להשתמש ב-self attention (–묘설 בחרחה בפרק 8). אחד השימושים הראשונים במנגנון ה-self attention עבור משימת עיבוד שפה היה בטרנספורמים, ובפרט בארכיטקטורת רשת הנקראט BERT, המבוססת על ה-encoder של הטרנספורמר המקורי. שימוש זה היווה פריצת דרך בתחום, וכךים ברוב המוחלט של המחבר והפייטוח בתחום ה-NLP משתמשים בארכיטקטורת רשת מבוססת self-attention. למעשה, BERT מציע שיטה לבניית ייצוג קונטקטואלי של מילים הבא להתמודד עם החולשות הקיימות ב-ELMo. נתאר בקצרה את העקרונות של מנגנון ה-self attention, שהוא הלב של BERT:

באופן הכל פשוט, בהקשר של עיבוד שפה self-attention הוא מנגנון שמשערק את הקשרים של כל מילה בטקסט כלשהו ביחס לשאר המילים באותו טקסט. כאשר מבצעים self-attention על קטע טקסט, מקבלים ייצוגים חדשים של המילים הולקים בחשבון גם את הקשרים בין המילים השונות באותו טקסט. בזאת אופיו של מנגנון ה-self attention ניתן לבנות ייצוג של מילה שתלויה בקשרים שלה עם מילים הנמצאות רחוק ממנה בקטע טקסט, ככלומר ההקשרים המתקברים בין המילים יכולים להיות מיזגים בצורה טובה גם עבור רצפים ארוכים ומילים שאין נמצאות בסמכותיחסית (שכאמור זה היה אחד החסרונות הגדולים של ELMo). בנוסף, מנגנון זה מיותר את הצורך לעבור מילה אחר מילה בקטע טקסט לצורך בניית ייצוג המילים שבו. במקרים מעבר זה, ה-encoder מקבל הקלט את כל קטע הטקסט כמקשה אחת, מה שעשויה להקטין את הזמן הנדרש עבור בניית הייצוג של המילים. لكن ה-encoder בטרנספורמר יכול לשמש מודל שפה, אם מאמנים אותו בצורה מתאימה.

בשונה ממודל LSTM ששומר את המצב בכל נקודת זמן ובעצם מקודד את המיקום של כל מילה בכך שהקלט מתקבל כמילה בזדמנות בכל פעם, מודל הטרנספורמר מקבל את כל הקלט בבת אחת. لكن בשайл לנקח בחשבון את המיקום של כל מילה במשפט אנחנו משתמשים באלמנט נוסף שנקרא Positional Embedding. אלמנט זה מקודד וקוטור ייחודי לכל מיקום במשפט ובסיום מבצעים חיבור של הוקטור שנוצר מהקלט והוקטור שנוצר מהמיקום.

הפותחים של BERT ימיצאו מארכיטקטורת הטרנספורמר המקורית את ה-encoder, והגדירו משימת אימון חדשה בכך להפוך אותו למודל שפה. בכך לבנות מודל מוצלח, תהליך האימון של BERT כולל שתי משימות: 1. Masked Language Model (MLM) – באופן רנדומלי עושים masking למילים מסוימות, ומטרת המודל הוא לחזות את המילים החסרות. 2. Next Sentence Prediction (NSP) – המודל מקבל קלט זוגות של משפטים מקטע טקסט, ומטרת המודל היא לחזות האם המשפט השני הוא המשכו של המשפט הראשון במסגר המקורי. ארכיטקטורת הרשת נראה כך:



איור 11.4 ארכיטקטורת BERT. הקלט הינו משפט המיצג כלשהו, והפלט הוא אותו משפט אך כל מילה קיבלה מידע נוסף על ההקשר שלו וכעת מיוצגת באופן חדש. תהליכי האימון והוספת ההקשר בין המילים נעשו באמצעות self-attention.

גם BERT, בדומה ל-ELMo, מציע בסופו של דבר מודל שפה מאומן הידוע לקחת טקסט השפה המוצג באופן מסוים ולהוסיף לו מידע על היחס בין המילים השונות שבtekst. תהליכי יצירתיות המודל היה אמם יקר, אך ניתן ללקחת אותו ויחסית בקלות לכילו אותו ואך להוסיף שכבות בקצבה עבור שימושים שפה שונים.

GPT: Generative Pre-trained Transformer

עם הכניסה של מנגנון attention-attention וטרנספורמרים לעולם ה-NLP, הוצעו יותר ויותר מודלים שפה מבוססי attention. לצורך ההמחשה ניתן לציין שבשנים האחרונות שבעברו מאז יצא BERT, הוא צוטט כבר בעשרות אלפי מאמרים. אחד המודלים הייתר מפורטים הינו Generative Pre-Training (GPT). מודל ה-GPT הינו מודל שעבוד בשיטת auto-regression, כלומר, כאשר המודל חוצה את המילה הבא הוא מוסיף את המילה לקלט עבור האיטרציה הבאה. כך הוא יכול בעצם ליצור משפטים מהתחילה של מילה בודדת. אם נרצה לדijk, המודלים הללו לא תמיד משתמשים במילים מיוחדות, לפעמים אנחנו נעבד עם חלקים מילים ואפיו אוטיות להם נקרא Tokennים או אסימונים. דבר זה יכול לעזור לנו בהכללה ולהקטיין את הסיסוי לtoken שלא נמצא במילון (Out of Vocabulary).

הארQUITקטורה של GPT בונה מ-transformers מה שמאפשר לבנות ארכיטקטורה עמוקה שמתחשבת בקונטקסט של המשפט עבור כל מילה (Contextual embeddings). ארכיטקטורת הינה היחידה המרכזית של GPT, אשר בשונה מ-BERT ה-GPT משתמש רק ב-decoder (מנגן ה-self-attention) שמקודד את הפיצרים, והפלט שלו הינו token הבא.

השכבה הראשונה בארכיטקטורה של GPT היא שכבה הנקראת Input encoding והוא הופכת את המילים (או ליתר דיוק הטוקנים) לוקטורים, כלומר היא מבצעת word embedding.

לאחר קידוד הקלט נשתמש במודול-h-Transformer בכדי לקודד פיצרים מהם נסיק את הטוקן הבא. התהיליך זהה מתבצע באמצעות רכיב הנקרא Masked Self attention. בשונה ממנגן self-attention רגיל שמקודד כל token בעזרת הkowskiט של כל שאר הטקסט, GPT צריך לקודד כל token רק בעזרת הטוקנים שקדמו לו, כיוון שבשלב זה המידע היחיד שקיים זה הטוקנים שנוצרו עד כה (וכmodoן שאין גישה לטוקנים שעדיין לא נוצרו). כאשר מקודדים את הייצוג עבור token מסוים, רכיב Masked Self attention מופיע כל וקטור של tokenuba אחריו, כך שהמודול לא יכול ללמידה ייצוג התלויה מילים שבאות לאחר הטוקן המוצג, אלא עליו להפוך את המירב מהtokנים הקודמים לו.

כיוון ש-GPT פועל בצורה של auto-regressive, ניתן לאחר האימון ליצור טקסט באמצעותו – ניתן למודל התחילה קצראה של טקסט, ובבקש ממנו ליצור את המילים הבאות. כך בכל שלב ניתן לו לקרוא את הטקסט הראשון ואת הטוקנים שייצור בשלבים הקודמים, והוא ימשיך וייצר עוד ועוד טקסט.

Perplexity

לאחר בניית מודל שפה, נרצה "למדוד" עד כמה הוא מצליח. לצורך זה יש להגדיר מטריקה מתאימה. המטריקה היכי נפוצה למדידת "עווצמה" של מודל שפה הינה perplexity, שזהו מושג הלקו מהטורת האינפורמציה והוא מודד כמה טוב מודל השפה חוזה את השפה ב-Corpus שאותו ניסינו למודל.

לפנינו שנסביר את המושג באופן פורמלי. ניתן אינטואיציה למה אנו מצפים לקבל מהמטריקה שבחרנו. נניח וננו מבצעים את הפעולה הבאה: ראשית לוקחים משפט שלם וחותכים ממנו את התחלה, ואז לוקחים את אותה התחלה ומכניםים כקילט למודל שפה ומקשימים מהמודל לחזות את המשך המשפט. כמובן שנרצה לקבל חיזוי שדומה ככל האפשר לשפט המקורי, וכן למדוד הצלחה של מודל על ידי השוואת הפלט שלו לשפט האמתי. באופן יותר כללי ניתן למשפט טקסט המכיל כמה משפטים כרצוננו, להכניס חלקים ממנו למודל השפה, ולהשוו את הפלט המתkeletal לטקסט המקורי. כיוון שמודל שפה הינו סטטורי, השוואת הפלט לשפט המקורי בלבד לאינה מספקת, כיון שהיא אינה משקפת בצורה מספיק טובה את מידת הצלחה שלה. אם למשל במשפט המקורי היתה כתובה המילה "לבנה" ואילו המודל חזה את המילה "רוח", השוואת שתי המילים כשלעצמה מראה לאורה שהמודל שגה לחוץין, אך בפועל אנו יודעים שמלים אלו נרדפות ולכן הפלט של המודל במרקחה זה הוא דווקא כן טוב. לכן, נרצה לבחור מدد המסוגל לבדוק עד כמה סביר לקבל את הפלט של המודל בהינתן חלק מהמשפט המקורי.

מדד perplexity בא להגדיר עם אתגר זה, והוא אכן פועל בצורה שונה מהאומן בו תיארנו את ההשוואה הפשטית בין טקסט המקורי לבין הפלט של מודל השפה. מדד זה מסתכל רק על הטקסט המקורי, והוא עובר מילה-מילה בטעסט זה ובודק **מה ההסתברות שמודל השפה ינביא את המילה הבאה בטקסט בהינתן כל המילים שלפנייה**. ככל שהסתברות יותר גבוהה, כך המודל יותר מוצלח. באופן פורמלי, perplexity מוגדר באופן הבא:

$$PP(W) = P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}}$$

ככל שהמודל מנבא בהסתברות גבוהה יותר את המילים של המשפט המקורי, כך המונה שבתוך השורש יהיה יותר גדול, וממילא כל הביטוי עצמו של perplexity נהיה קטן יותר. לעומת זאת, ככל שערך perplexity קטן יותר, כך המודל מוצלח יותר. נפתח מעט את הביטוי האחרון בעזרת כלל השרשרת:

$$= \sqrt[N]{\frac{1}{\prod_{i=1}^N P(w_i | w_{i-1}, w_{i-2}, \dots, w_1)}}$$

למשל עבור מודל מבוסס bigram, המדד יהיה פשוט יותר ויראה כך:

$$= \sqrt[N]{\frac{1}{\prod_{i=1}^N P(w_i | w_{i-1})}}$$

כאמור לעיל, ככל שערך מדד perplexity נמוך יותר, כך מודל השפה איקוטי יותר.

10. References

<http://d2l.ai/>

ELMo, BERT:

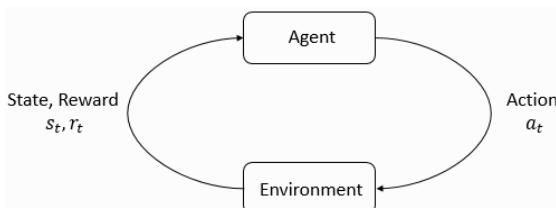
<https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html>

11. Reinforcement Learning (RL)

רוב האלגוריתמים של עולם הלמידה הינם מבוססי דата, כלומר, בהינתן מידע מסוים הם מנוסים למצוא בו חוקיות מסוימת, ועל בסיסה לבנות מודל שיוכל להתאים למקדים נוספים. אלגוריתמים אלה מחולקים לשניים:

1. אלגוריתמים של למידה מונחת, המבוססים על דата $\{x = S, \text{ כאשר } x \in \mathbb{R}^d \text{ ו } y \in \mathbb{R}^n\}$ הינו אוסף של אובייקטים (למשל נקודות במרחב, אוסף של תמונות וכו'), ו- $y \in \mathbb{R}^n$ הינו אוסף של labels. לכל אובייקט $x \in \mathbb{R}^d$ יש label מתאים $y \in \mathbb{R}^n$.
2. אלגוריתמים של למידה לא מונחת עבורם הדטה $\{x \in \mathbb{R}^d\}$ הוא אוסף של אובייקטים ללא labels, ומנסים למצוא כלליים מסוימים על דטא זה (למשל – חלוקה לאשכולות, הורדת ממד ועוד).

למידה מבוססת חזוקים הינה פרדיגמה נוספת תחת התחום של למידת מכונה, כאשר במקורה זה הלמידה לא מסתמכת על דטה קיים, אלא על חקירה של הסביבה ומציאת המדיניות/הסטרטגיה הטובה ביותר לפועלה. שמו סוכן שנמצא בסביבה שאינה מוכרת, ועלוי לבצע צעדים כך שהתגמול המתקבל אותו הוא יקבל יהיה מקסימלי. בלמידה מבוססת חזוקים, בנויגוד לפרדיוגמות האחרות של למידת מכונה, הסביבה לא ידועה מראש מבעוד מועד. הסוכן נמצא באירועים וainו יודע בשום שלב מה הצעד הנוכחי לעשות, אלא הוא רק מקבל פידבק על הצעדים שלו, וכך הוא לומד מה כדי לעשות ומה כדי להימנע. באופן כללי ניתן לומר שמטרת הלמידה היא לייצר אסטרטגיה כך שככל מני מצבים לא ידועים הסוכן יבחר בפעולות שבאותן מצבים יהיה הכי עיליות עבורו. נתאר את תהליך הלמידה באופן גרפי:



איור 11.1 מודל של סוכן וסביבה.

בכל צעד הסוכן נמצא במצב s_t ובוחר פעולה a_t המעבירו אותו למצב s_{t+1} , ובהתאם לכך הוא מקבל מהסביבה תגמול r_t . האופן בה מתבצעת הלמידה היא בעזרת התגמול, כאשר נזיה שהסוכן יבצע פעולות המזקחות אותו בתגמול חיובי (-חזוק) וימנע מפעולות עבורן הוא מקבל תגמול שלילי, ובמצטבר הוא ימסס את כלל התגמולים עבור כל הצעדים שהוא בחר לעשות. כדי להבין כיצד האלגוריתמים של למידה מבוססת חזוקים עובדים ראשית יש להגדיר את המושגים השונים, ובנוסף יש לנסח באופן פורמלי את התיאור המתמטי של חלקו הבלתי השוני.

11.1 Introduction to RL

בפרק זה נגדיר באופן פורמלי תהליכי מרקוב, בעזרתם ניתן לתאר בעיות של למידה מבוססת חזוקים, ונראה כיצד ניתן למצוא אופטימום לבעיות אלו בהינתן מודל וכל הפרמטרים שלו. לאחר מכן נדון בקשרה במספר שיטות המנסות למצוא אסטרטגיה אופטימלית עבור תהליכי מרקוב כאשר לא כל הפרמטרים של המודל נתונים, ובפרק הבא נדבר על שיטות אלה בהרחבה. שיטות אלה הן למעשה הלב של למידה מבוססת חזוקים, כיון שהן מנוסות למצוא אסטרטגיה אופטימלית על בסיס תגמולים ללא ידיעת הפרמטרים של המודל המרковי עבורו רוצים למצוא אופטימום.

11.1.1 Markov Decision Process (MDP) and RL

המודל המתמטי העיקרי העיקרי עליו מבוסים האלגוריתמים השונים של RL הינו תהליכי מרקובי, כלומר תהליכי שבה המעברים בין המצבים מקיימים את תכונת מרקוב, לפיה ההתפלגות של מצב מסוים תלויה רק במצב הקודם לו:

$$P(s_{t+1} = j | s_t = i, s_1, \dots, s_t) = P(s_{t+1} = j | s_t = i)$$

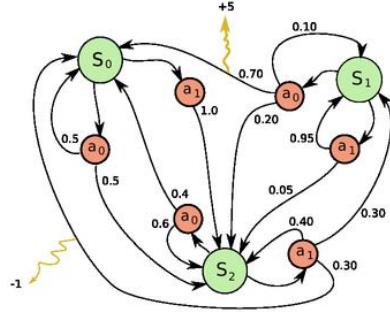
תהליכי קבלת החלטות מרקובי מתואר על ידי סט הפרמטרים $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}\}$:

- (\mathcal{S}) – מרחב המצבים של המערכת. המצב ההתחלתי מסומן ב- S_0 .
- (\mathcal{A}) – מרחב הפעולות. A_s הוא מרחב הפעולות האפשריות במצב S .
- (\mathcal{T}) – הbijection: $[0, 1] \rightarrow [0, 1]$ הינו פונקציית מעבר, המחשבת את ההסתברות לעבור בזמן t במצב s_t למצב s_{t+1} על ידי הפעולה a : $T(s_t, a) = s_{t+1}$.
- מעשה מייצג את המודל – מה ההסתברות שבחירה הפעולה a במצב s תביא את הסוכן למצב s' .
- (\mathcal{R}) – הbijection: $\mathbb{R} \rightarrow (\mathbb{R}, s, R_a)$ הינו פונקציה הנונגנת תגמול/רווח לכל פעולה a הגורמת למעבר ממצב s למצב s' , כאשר בדרך כלל $[0, 1] \in \mathcal{R}_a$. לעיתים מנסים את התגמול של הצעד בזמן t ב- R_t .

המרקזיות של התהילך באה ידי ביטוי בכך שמצב s_t מכיל בתוכו את כל המידע הנחוץ כדי לקבל החלטה לגבי a_t , או במקרה אחר – כל ההיסטוריה עצמה שמורה בתחום המצב s_t .

ריצה של MDP מואפינית על ידי הרביעיה הסדרה $\{s_t, a_t, r_t, s_{t+1}\}$ – פעולה a_t המתבצעת בזמן t וגורמת למעבר ממצב s_t למצב s_{t+1} , ובנוסף מקבלת תגמול מיידי r_t , כאשר $r_t \sim \mathcal{R}(s_t, a_t)$.

מסלול (trajectory) הינו סט של שלשות $\{s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{t-1}, a_{t-1}, r_{t-1}, s_t\}$, כאשר המצב ההתחלתי מוגדר מהתפלגות כלשהיא $(\cdot | s_0, a_0)$, והמעבר בין הממצבים יכול להיות דטרמיניסטי $s_{t+1} = f(s_t, a_t)$ או סטוכסטי $s_{t+1} \sim p(\cdot | s_t, a_t)$.



איור 11.2 תהליך קבלת החלטות מركז. ישנו שלושה מצבים – $\{s_0, s_1, s_2\}$, ובכל אחד מהם יש שתי פעולות אפשריות (עם הסתברויות מעבר שונות) – $\{a_0, a_1\}$. עבור חלק מהפעולות יש תגמול שונה מ-0. מסלול יהיה מעבר על אוסף של מצבים דרך אוסף של פעולות, שלכל אחד מהם יש תגמול.

אסטרטגיה של סוכן, המסומנת ב- π , הינה בחירה של אוסף מהלכים. בעיות של למידה מבוססת חיזוקים, נרצה למצוא **אסטרטגיה אופטימלית** (Optimal Policy) $\pi: S \rightarrow A$ הממקסמת את **התגמול המצטבר** ($\sum_{t=0}^{\infty} \mathbb{E}[R(s_t, a_t) | \pi]$). כיוון שלא תמיד אפשר לחשב באופן ישיר את האסטרטגיה האופטימלית, ניתן להגדיר ערך החזרה (Return) המבטא סכום של תגמולים, ומנסים למקסם את התוחלת שלו $\mathbb{E}[Return | \mathcal{S}, \mathcal{A}]$. ערך ההחזרה הינו נפוץ ונקרא **Return** או **discount return**, והוא מוגדר באופן הבא: עבור פרמטר $\gamma \in (0, 1)$, ה- γ -return הינו הסכום הבא:

$$Return = \sum_{t=1}^T \gamma^t r_t$$

אם $\gamma = 1$, אז מתעניינים רק בתגמול המיידי, וככל ש- γ גדול כך יותר נתונים יותר לתגמולים עתידיים. כיוון ש- $[0, 1] \ni r_t$, הסכום חסום על ידי $\frac{1}{1-\gamma}$.

התוחלת של ערך ההחזרה נקראת **Value function**, והוא נותן לכל מצב ערך מסוים המשקף את תוחלת התגמול שנitin להישיג דרך מצב זה. באופן פורמלי, כאשר מתחילה במצב s , ה-value function מוגדר להיות:

$$\mathcal{V}^\pi(s) = \mathbb{E}[Return | s_0 = s]$$

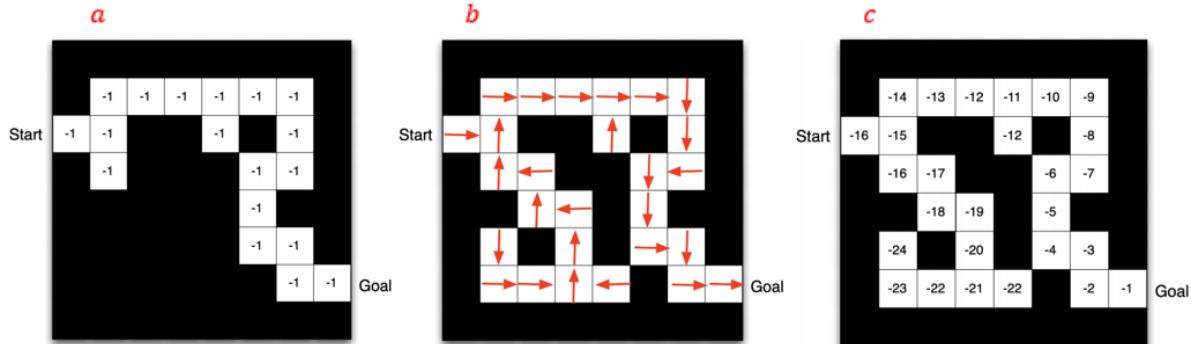
בעזרת ביטוי זה ניתן לחשב את **האסטרטגיה האופטימלית**, כאשר ניתן לנוקוט בגישה ישירה ובגישה עקיפה. הגישה הישירה מנסה למצאו בכל מצב מה פעולה הכי מתאים. בהתאם לכך, חישוב האסטרטגיה האופטימלית יעשה באופן הבא:

$$\pi(s) = \arg \max_a \sum_{s'} p_a(s, s') (\mathcal{R}_a(s, s') + \gamma \mathcal{V}^\pi(s'))$$

לעתים החישוב הישיר מסובך, כיוון שהוא צריך ללקח בחשבון את כל הפעולות האפשרות, ולכן מסתכלים רק על ה-value function. לאחר שלכל מצב יש ערך מסוים, בכל מצב הסוכן יעבור למצב בעל הערך הכי גדול מבין כל הממצבים האפשריים אליו הם ניתן לעבור. חישוב הערך של כל מצב נעשה באופן הבא:

$$\mathcal{V}(s) = \sum_{s'} p_\pi(s, s') (\mathcal{R}_\pi(s, s') + \gamma V(s'))$$

ניתן לשים לב שבעוד הגישה הראשונה מתמקדת במציאת אסטרטגיה/מדיניות אופטימלית על בסיס הפעולות האפשריות בכל מצב, הגישה השנייה לא מסתכלת על הפעולות אלא על הערך של כל מצב, המשקף את תוחלת התגמול שnitן להשיג כאשר נמצאים במצב זה.



איור 11.3 a) מודל: המצב של הסוקן הוא המשבצת בו הוא נמצא, הפעולות האפשריות הן ארבעת הכיוונים, כל פעולה גוררת תגמול של -1, והסתברויות המעבר קבועות לפני הצבעים של המשבצות (או אפשר ללחוץ למשבצות שחומות). b) מדיניות – החלטה בכל מצבizia עד לבצע. c) Value של כל משבצת.

לסיכום, ניתן לומר שכל התחומים של RL מבוסס על שלוש אבני יסוד:

- מודל: האופן בו אנו מתארים את מרחב המצבים והפעולות. המודל יכול להיות נתון או שנוצר לערך אותו, והוא מורכב מהסתברויות מעבר בין מצבים ותגמול עבור כל צד:

$$\mathcal{P}_{ss'}^a = p_\pi(s, s' | s_t = s, a_t = a)$$

$$\mathcal{R}_{ss'}^a = \mathcal{R}_\pi(s, s' | s_t = s, a_t = a, s_{t+1} = s')$$
- פונקציה המתארת את התוחלת של התגמולים העתידיים: Value function

$$V^\pi(s) = \mathbb{E}[R(\tau) | s_0 = s]$$

- מדיניות/אסטרטגיה (Policy) – בחירה (דטרמיניסטיית או אקראית) של צעד בכל מצב נתון:

$$\pi(s|a)$$

11.1.2 Bellman Equation

לאחר שהגדכנו את המטרה של למידה מבוססת חיזוקים, ניתן לדבר על שיטות לחישוב אסטרטגיה אופטימלית. בפרק זה נתייחס למקרה הספציפי בו נתון מודל מركובי עם כל הפרמטרים שלו, כמו ראות המצבים, הפעולות והסתברויות המעבר ידועים. כאמור, Action-value function היא הינה התוחלת של ערך ההחזרה עבור אסטרטגיה נתונה π , כאשר מתחילה במצב s :

$$V^\pi(s) = \mathbb{E}[R(\tau) | s_0 = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right]$$

ביטוי זה מסתכל על הערך של כל מצב, בלי להתייחס לפעולות המעבירות את הסוקן ממצב אחד למצב אחר. נתינת ערך לכל מצב יכולה לשיער במציאות אסטרטגיה אופטימלית, כיוון שהיא מדרגת את המצבים השונים של המודל. באופן דומה, ניתן להגיד את ה-Action-Value function – התוחלת של ערך ההחזרה עבור אסטרטגיה נתונה π , כאשר במצב s מבצעים את פעולה a , ולאחר מכן ממשיכים לפי האסטרטגיה π :

$$Q^\pi(s, a) = \mathbb{E}[R(\tau) | s_0 = s, a_0 = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right]$$

ביטוי זה מסתכל על הזוג (s_t, a_t) , כלומר בכל מצב יש התייחסות למצב הנוכחי ולפעולות האפשריות במצב זה. בדומה ל-Value function, גם ביטוי זה יכול לשיער במציאות אסטרטגיה אופטימלית, כיוון שהוא מדרג עבור כל מצב את הפעולות האפשריות.

ונכל לסמן ב- $-(s, a)$ את הערך של האסטרטגיה האופטימלית π^* – Optimal Value function ו- $Q^*(s, a)$ את הערך של האסטרטגיה זו מתקין: Optimal Action-Value function

$$\mathcal{V}^*(s) = \max_{\pi} \mathbb{E}[R(\tau)|s_0 = s], \mathcal{Q}^*(s, a) = \max_{\pi} \mathbb{E}[R(\tau)|s_0 = s, a_0 = a]$$

הרבה פעמים מתעניינים ביחס שבין \mathcal{V} -ו- \mathcal{Q} , וניתן להיעזר במערכות הבאים:

$$\mathcal{V}^\pi(s) = \mathbb{E}[\mathcal{Q}^\pi(s, a)]$$

$$\mathcal{V}^*(s) = \max_{\pi} \mathcal{Q}^*(s, a)$$

באופן קומפקטי ניתן לרשום את (s^*, \mathcal{V}) כך:

$$\forall s \in S \quad \mathcal{V}^*(s) = \max_{\pi} \mathcal{V}^\pi(s)$$

כלומר, האסטרטגיה π^* הינה האופטימלית עבור כל מצב s .

עת נתון מודל מרקובי עם כל הפרמטרים שלו – אוסף המצבים והפעולות, הסתברויות המעבר והתגמול עבור כל פעולה, ומעוניינים למצאו דרך פעולה אופטימלית עבור מודל זה. ניתן לעשות זאת בשתי דרכים עיקריות – מציאת האסטרטגיה $(s|a)\pi^*$ האופטימלית, או חישוב-value של כל מצב ובחרת מצבים בהתאם לערך זה. משימות אלו יכולות להיות מסובכות מאוד עבור משימות מורכבות וגדולות, ולכן לעיתים קרובות משתמשים בשיטות איטרטיביות ובקירובים על מנת לדעת כיצד להנוג בכל מצב. הדרך הפешוטה לחישוב $(s|\pi^*)\mathcal{V}$ משתמשת ב-Bellman equation, המבוססת על תכונות דינמי. נפתח את הביטוי של $(s|\pi^*)\mathcal{V}$ מתוך ההגדרה שלו:

$$\mathcal{V}^\pi(s) = \mathbb{E}[R(\tau)|s_0 = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right]$$

נפצל את הסכום שבתוכלה לשני איברים – האיבר הראשון ויתר האיברים:

$$= \mathbb{E}_\pi \left[r_{t+1} + \gamma \cdot \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s \right]$$

cut נשותמש בהגדרת התוכחת ונקבל:

$$\begin{aligned} &= \sum_{a,s'} \pi(a|s) p_\pi(s, s') \left(\mathcal{R}_\pi(s, s') + \gamma \cdot \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s \right] \right) \\ &= \sum_{a,s'} \pi(a|s) p_\pi(s, s') (\mathcal{R}_\pi(s, s') + \gamma \cdot \mathcal{V}^\pi(s')) \end{aligned}$$

הביטוי המתkeletal הוא מערכת משוואות לינאריות הניתנות לפתרון באופן אנליטי, אם כי סיבוכיות החישוב יקרה. נסמן:

$$V = [V_1, \dots, V_n]^T, R = [r_1, \dots, r_n]^T$$

$$T = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

ונקבל משווהה מטריצионаית:

$$V = R + \gamma T V \rightarrow V = R + \gamma T V$$

$$\rightarrow \mathcal{V}^\pi(s) = (\mathbb{I}_n - \gamma T)^{-1} R$$

בגלל שהערכים העצמיים של T חסומים על ידי 1, בהכרח יהיה ניתן להפוך את $\gamma T - \mathbb{I}_n$ מה שمبرטיח שייהי פתרון למשווהה, ופתרון זה הוא אף ייחיד עבור π^* . כמשמעותם את V ניתן למצוא גם את \mathcal{Q}^π על ידי הקשר:

$$\mathcal{Q}^\pi(s, a) = \sum_{s'} p_\pi(s, s') (\mathcal{R}_\pi(s, s') + \gamma \mathcal{V}^\pi(s')) = \sum_{s'} p_\pi(s, s') \left(\mathcal{R}_\pi(s, s') + \gamma \sum_{a'} \pi(a'|s') \mathcal{Q}^\pi(a'|s') \right)$$

Iterative Policy Evaluation

הסיבוכיות של היפוך מטריצה הינה $O(a^3)$, ועבור a גדול החישוב נהיה מאוד יקר ולא עיל. כדי לחשב את הפתרון באופן עיל, ניתן כאמור להשתמש בשיטות איטרטיביות. שיטות אלו מבוססות על אופרטור בלמן, המוגדר באופן הבא:

$$BO(V) = R^\pi + \gamma T^\pi \cdot V$$

ניתן להוכיח שאופרטור זה הינו העתקה מכווצת (contractive mapping), כלומר הוא מקיים את התנאי:

$$\forall x, y: \|f(x) - f(y)\| < \gamma \|x - y\| \text{ for } 0 < \gamma < 1$$

במילים: עבור שני וקטורים במרחב, אופרטור $(\cdot)f$ ומספר γ החסום בין 0 ל-1, אם נפעיל את האופרטור על כל אחד מהווקטורים ונחשב את נורמת ההפרש, נקבל מספר קטן יותר מאשר הנורמה בין הווקטורים כפול הפקטור γ . אופרטור המקיים תכונה זו הינו העתקה מכווצת, כיוון שנורמת ההפרש של האופרטור על שני וקטורים קטנה מnorמת ההפרש בין הווקטורים עצמו. הוכחה:

$$\|f(u) - f(v)\|_\infty = \|R^\pi + \gamma T^\pi \cdot u - (R^\pi + \gamma T^\pi \cdot v)\|_\infty = \|\gamma T^\pi(u - v)\|_\infty$$

מטריקת אינסוף מוגדרת לפיה: $\|(s)v - u\|_\infty = \max_{s \in S} |u(s) - v(s)|$. לכן נוכל לרשום:

$$\|\gamma T^\pi(u - v)\|_\infty \leq \|\gamma T^\pi\|_\infty \|u - v\|_\infty$$

הביטוי $\|\gamma T^\pi\|_\infty$ למעשה סוכם את כל ערכי מטריצת המעברים, שכן הוא מסתכם ל-1, ונקבל:

$$= \gamma \|u - v\|_\infty$$

ובכך הוכחנו את הדרוש.

לפי משפט נקודת השבת של בנך, להעתקה מכווצת יש נקודת שבת (fixed point) יחידה המקיים $(x)f = x$ וסדרה $(x_t)f = x_{t+1}$ המתכנסת לאוותה נקודת שבת. לכן נוכל להשתמש באלגוריתם איטרטיבי עבור T^π שיביא אותנו לנקודת שבת, ולפי המשפט זהוי נקודת השבת היחידה ומילא הגענו להתכנסות. בפועל, נשתמש באלגוריתם האיטרטיבי הבא:

$$V_{k+1} = BO(V_k) = R^\pi + \gamma T^\pi \cdot V_k$$

נסתכל על הדוגמא הבאה:

$$T^\pi = \begin{pmatrix} 0.8 & 0.1 & 0.1 & 0 & 0 \\ 0.1 & 0.8 & 0.1 & 0 & 0 \\ 0 & 0.1 & 0.8 & 0.1 & 0 \\ 0 & 0 & 0.1 & 0.8 & 0.1 \\ 0 & 0 & 0.1 & 0.1 & 0.8 \end{pmatrix}, \mathcal{R}^\pi = \begin{pmatrix} 0.1 \\ 1.3 \\ 3.4 \\ 1.9 \\ 0.4 \end{pmatrix}, \gamma = 0.9$$

באמצעות השיטה האיטרטיבית נקבל:

$$V_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, V_1 = \begin{pmatrix} 0.1 \\ 1.3 \\ 3.4 \\ 1.9 \\ 0.4 \end{pmatrix}, V_2 = \begin{pmatrix} 0.6 \\ 2.6 \\ 6.1 \\ 3.7 \\ 1.2 \end{pmatrix}, \dots, V_{10} = \begin{pmatrix} 7.6 \\ 10.8 \\ 18.2 \\ 3.7 \\ 12.0 \end{pmatrix}, \dots, V_{50} = \begin{pmatrix} 14.5 \\ 17.1 \\ 26.4 \\ 26.8 \\ 18.4 \end{pmatrix}, V^\pi = \begin{pmatrix} 14.7 \\ 17.9 \\ 26.6 \\ 27.1 \\ 18.7 \end{pmatrix}$$

ניתן לשים לב שאחרי 50 איטרציות הפתרון המתkeletal בצורה האיטרטיבית קרוב מאוד לפתרון המתkeletal בצורה האנליטית.

Policy Iteration (PI)

חישוב π -Value function מאפשר לחשב את ערכו של $(s)\pi$ עבור כל s , אך הוא אינו מבטיח שנגע לאסטרטגיה האופטימלית. נניח והצלהנו לחשב את $(s)\pi$ וממנו אנו יודעים לגזר אסטרטגיה, עדין יתכן שקיימות פעולות a שייתר משלמת מאשר הפעולה המוצעת לפי האסטרטגיה הנגזרת מ- $(s)\pi$. באופן פורמלי ניתן לתאר זאת בצורה פשוטה – נניח שהיחסנו את $(s)\pi$ ואת $(s)\pi^*$ יתכן וקייםת פעולה עבורה:

for such s, a : $\mathcal{Q}^\pi(s, a) > \mathcal{V}^\pi(s)$

אם קיימת פעולה כזו, אז ישתלם לבחור בה ורק לאחר מכן לוחזר לפעול בהתאם לאסטרטגיה $(s|a)\pi$ הנגזרת מחישוב ה-Value function. למעשה, ניתן לחפש את כל הפעולות עבורן כדי לבצע פעולה מסוימת עבורה התגמול יהיה גבוה יותר מאשר האסטרטגיה של $(s)\pi'$. באופן פורמלי יותר, נרצה להגדיר אסטרטגיה דטרמיניסטית, עבורה בהסתברות 1 ננקוט בכל מצב s בפעולה הכי כדאי a :

$$\pi'(s) = \arg \max_{a'} \mathcal{Q}^\pi(s, a')$$

נשים לב שרגע זה הוא בעצם להשתמש באסטרטגיה גרידית – בכל מצב לננקוט בפעולה הכי משלימה בטוחה של צעד אחד. השאלה העולה היא כמובן – מדוע זה בהכרח נכון? כאמור, אם הרעיון של לבחור **בכל** צעד את a האופטימלי בהכרח תוביל ל渴בלת אסטרטגיה אופטימלית עבור כל הזרים כולם יחד? בכך להוכיח זאתணסח זאת ממשפט:

בاهינתן 2 אסטרטגיות π' , π , כאשר π' דטרמיניסטית, אז כאשר $\mathcal{V}^\pi(s) > \mathcal{V}^{\pi'}(s)$ בהכרח לכל s יתקיים: $\mathcal{Q}^\pi(s) > \mathcal{Q}^{\pi'}(s)$. ראשית נפתח לפני הגדרה:

$$\mathcal{V}^\pi(s) < \mathcal{Q}^\pi(s, \pi'(s)) = \mathbb{E}_\pi[r_{t+1} + \gamma \cdot \mathcal{V}^\pi(s_{t+1}) | s_t = s, a_t = \pi'(s)]$$

כיוון שהאסטרטגיה הינה דטרמיניסטית, הפעולה הנבחרת אינה רנדומלית ביחס ל- π' , ולכן נוכל לרשום:

$$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot \mathcal{V}^\pi(s_{t+1}) | s_t = s]$$

cutת לפני אותו אי שווין שבנהנה נוכל לבצע את אותו חישוב גם לצעד הבא: s_{t+2} :

$$< \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot \mathcal{Q}^\pi(s_{t+1}, \pi'(s_{t+1})) | s_t = s]$$

זה שווה ל:

$$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot \mathcal{V}^\pi(s_{t+2}) | s_t = s]$$

וכך הלאה, ולמעשה הוכחנו את הדריש – נקיית הפעולה הכי עיליה בכל מצב תמיד תהיה יותר טובה מהפתרון של $(s)\pi'$. cutת יש בידינו שתי טכניקות שאנו יודעים לבצע:

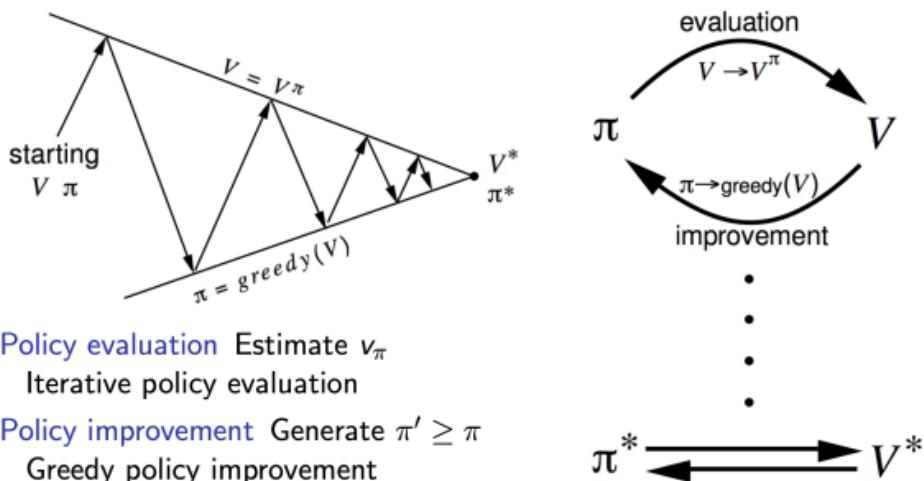
Evaluation (E) – בהינתן אסטרטגיה מסוימת נוכל לפתור את משוואות בלמן ולקבל את $\mathcal{V}^\pi(s)$ ו- $\mathcal{Q}^\pi(s, a)$.

Improve (I) – בהינתן הערך של value function של π , ניתן להתחיל מאסטרטגיה רנדומלית, ואז לבצע איטרציות המורכבות משתי הטכניקות האלה באופן הבא:

$$\pi_0 \xrightarrow{E} \pi_1 \xrightarrow{I} \pi_2 \xrightarrow{E} \pi_3 \xrightarrow{I} \dots$$

תהליך זה נקרא Policy iteration – בכל צעד בו יש לנו אסטרטגיה נפתרו עבורה משוואות בלמן ובכך נחשב את ה- Value function שלה, ולאחר מכן נשפר את האסטרטגיה באמצעות policy improvement, שכך מוצע בחירה גרידית שבתוך הקצר טוביה יותר מאשר ה-Value function שחישבנו. ניתן להוכיח שאחרי מספר סופי של איטרציות האסטרטגיה תתכנס לנקודת שבת (fixed point), ואז הפעולה הבאה לפיה האסטרטגיה תהיה לבחירה הגדית:

$$\pi(s) = \arg \max_a \mathcal{Q}^\pi(s, a) = \pi'(s)$$



איור 11.4 – ביצוע איטרציות של Policy improvement ו-Policy evaluation על מנת למצוא בכל שלב את ה-value function ולשפר אותו באמצעות בחירה גרידית.

Bellman optimality equations

השלב הבא בשימוש ב-*policy iteration* הוא **להוכיח** שהאסטרטגיה אליה מתכנסים הינה אופטימלית. נסמן את נקודת השבת ב- π^* ונקבל את הקשר הבא:

$$V^{\pi^*}(s) \equiv V^*(s) = \max_a Q^*(s, a) = \max_a \sum_{s'} p_\pi(s, s') \left(R_\pi(s, s') + \gamma \cdot V^*(s') \right)$$

ובאופן דומה:

$$Q^*(s, a) = \sum_{s'} p_\pi(s, s') \left(R_\pi(s, s') + \gamma \cdot \max_{a'} Q^*(s', a') \right)$$

משוואות אלה נקראות Bellman optimality equation. ניתן לשים לב שהן מאוד דומות למשוואות בלמן מהן ייצנו, אך במקומם התוחלת שהייתה לנו בהתחלה, כעת יש *max*. נרצה להראות שהפתרון של משוואות אלה הוא *the Value* של האסטרטגיה האופטימלית. ננסח את הטענה באופן הבא:

אסטרטגיה הינה אופטימלית אם ורק אם היא מקיימת את Bellman optimality equation. כיוון אחד להוכחה הוא טריויאלי – אם האסטרטגיה הינה אופטימלית אז היא בהכרח מקיימת את משוואות האופטימליות, כיוון שהראינו שהן מתקבלות מנקודת השבת אליה האיטרציות מתכנסות. אם האסטרטגיה לא הייתה אופטימלית אז היה ניתן לשפר עוד את האסטרטגיה ולא היינו מגאים עדין לנקודת השבת. בשביל להוכחה את הכיוון השני השתמש שוב ברעיון של העתקה מכווצת. נגדיר את האופרטור הבא:

$$BV(s) = \max_a \sum_{s'} p_\pi(s, s') \left(R_\pi(s, s') + \gamma \cdot V(s') \right)$$

ניתן להראות שאופרטור זה הינו העתקה מכווצת, וממילא לפי המשפט של בנך יש לו נקודת שבת יחידה. כיוון שהראינו שימוש ב-*policy iteration* מביא את האסטרטגיה לנקודת שבת מסוימת, נוכל לצרף לכך את העבודה שהאופרטור שהגדכנו הינו העתקה מכווצת וממילא קיבל שאותה נקודת שבת הינה יחידה, וממילא אופטימלית.

Value Iteration

הראנו שבעזרת שיטת *policy iteration* ניתן להגיע לאסטרטגיה אופטימלית, אך התהליך יכול להיות איטי. ניתן לנוקוט גם בגישה יותר ישירה ולנסות לחשב באופן ישיר את הפתרון של משוואות האופטימליות של בלמן (ופתרון הינו אופטימלי כיוון שהראינו שהפתרון הוא נקודת שבת יחידה). נתחיל עם פתרון רנדומלי V_0 ולאחר מכן נקבע איטרציות באופן הבא עד שנגיע להתקנסות:

$$\mathcal{V}_{k+1} = \max_a \sum_{s'} p_\pi(s, s') (\mathcal{R}_\pi(s, s') + \gamma \cdot \mathcal{V}_k(s'))$$

נשים לב שבשיטת זו אין לנו מידע לגבי האסטרטגיה אלא רק חישבנו את ה-Value function, אך ממנה ניתן לגוזר את Q ואז לבחור באסטרטגיה גרידית, שהינה במקורה זה גם אופטימלית:

$$\pi(s) = \arg \max_a Q^\pi(s, a)$$

ניתן להראות כי בשיטה זו ההתקנסות מהירה יותר ודרושים פחות איטרציות מהשיטה הקודמת, אך כל איטרציה יותר מורכבת.

Limitations

לשתי השיטות – Policy iteration ו-Value iteration – יש שני חסרונות מרכזיים:

1. הן דורשות לדעת את המודול והסיבבה באופןשלם ומדויק.

2. הן דורשות לעדכן בכל שלב את כל המ מצבים בו זמן. עבור מערכות עם הרבה מצבים, זה לא מעשי.

11.1.3 Learning Algorithms

בפרק הקודם הוסבר כיצד ניתן לחשב את האסטרטגיה האופטימלית וערך ההצעה **בהינתן** מודל מרקובי. השתמשנו בשתי הנחות עיקריות על מנת להתמודד עם הבעיה:

1. Tabular MDP – הנחנו שהבעיה סופית ולא גדולה מדי, כך שנוכל ליציג אותה בזכרון ולפתרו אותה.

2. Known environment – הנחנו שהמודול ידוע לנו, כלומר נתונה לנו מטריצת המעברים שקובעת מה הסיכוי לעבור מצב s' במצב s למצב s' כשלוקטים בפועלה a (סימנו את זה בתור $(s', s) = \mathcal{P}_{ss'}^a$, ובנוסף נתון לנו מה ה- Q המתקבל עבור כל action a סימנו את זה בתור $(s', s) = \mathcal{R}_{ss'}^a$).

בעזרת שתי הנחות פיתחנו את המשוואות בלבד, כאשר הוי לנו שני צמדים של משוואות. משוואות בלבד מעבור אסטרטגיה נתונה כתובות באופן הבא:

$$\mathcal{V}^\pi(s) = \sum_{a, s'} \pi(a|s) \mathcal{P}_{ss'}^a (\mathcal{R}_{ss'}^a + \gamma \cdot \mathcal{V}^\pi(s'))$$

$$Q^\pi(s, a) = \sum_{s'} \mathcal{P}_{ss'}^a \left(\mathcal{R}_{ss'}^a + \gamma \sum_{a'} Q^\pi(s', a') \right)$$

ובנוסף פיתחנו את המשוואות עבור הפתרון האופטימלי:

$$\mathcal{V}^*(s) = \max_a \sum_{s'} \mathcal{P}_{ss'}^a (\mathcal{R}_{ss'}^a + \gamma \cdot \mathcal{V}^*(s'))$$

$$Q^*(s, a) = \sum_{s'} \mathcal{P}_{ss'}^a \left(\mathcal{R}_{ss'}^a + \gamma \max_a Q^*(s', a') \right)$$

הראינו שתי דרכי להגעה לפתרון האופטימלי:

1. Policy improvement המורכב מ-Policy evaluation ולאחריו Policy iteration.

2. Value iteration – פתרון משוואות בלבד באופן ישיר באמצעות איטרציות על ה-Value function.

כאמור, דרכי פתרון אלו מניחים שהמודול ידוע, ובנוסף שמדובר במצבים איננו גדול מדי וכי יכול להיות מוצג בזכרון. האתגר האמייתי מתחילה בנקודה בה לפחות אחת מהנחות אלה אינה תקיפה, ולמעשה פה מתחילה התפקיד של אלגוריתמי RL. עיקר ההתקנות של אלגוריתמים אלו יהיה למצאו באופן יעיל את האסטרטגיה האופטימלית כאשר לא נתונים הפרמטרים של המודול, וזאת לשערך אותם (Model-based learning) או למצוא דרך אחרת לחישוב האסטרטגיה האופטימלית ללא שימוש במודל (Model free learning). אם למשל יש משחק בין משתמש לבין המחשב, אלגוריתמים השייכים ל-Model based learning ינסו ללמידה את המודל של המשחק או להשתמש במודל

קיים, ובעזרת המודל הם ינסו לבחון כיצד יגיב המשתמש לכל תור שהמחשב יבחר. לעומת זאת אלגוריתמים מסווג Model free learning לא יתעניינו בכך, אלא ינסו ללמידה ישירות את האסטרטגיה הטובה ביותר עבור המחשב.

היתרון המשמעותי של אלגוריתמים המשתיכים על המודל של הבעיה (Model-based) נובע מהיכולת לתקן מספר צעדים קדימה, כאשר עברו כל בחירה של פעולה המודל בוחן את התוצאות האפשריות, את הפעולות המתאימות לכל תגובה, וכן הלאה. דוגמא מפורסמת לכך היא תוכנת המחשב AlphaZero שאומנה לשחק משחקי לוח כגון שחמט או גו. במקרים אלו המודל הוא המשחק והחוקים שלו, והתוכנה משתמשת בידע זהה בכך כדי לבחון את כל הפעולות וההתוצאות למשך מספר צעדים רב ובחירה של הצעד הטוב ביותר.

עם זאת, בדרך כלל אף בשלב האימון אין לסוקן מידע חיצוני מהו הצעד הנכון באופן אולטימטיבי, ועלוי ללמידה רק מהניסיון. עובדה זו מציבה כמה אתגרים, כאשר העיקרי ביניהם הוא הסenna שהאסטרטגיה הנלמדת תהיה טובה רק עבור המקרים אותם ראה הסוקן, אך לא תתאים למקרים חדשים שיבואו. אלגוריתמים שמחפשים באופן ישיר את האסטרטגיה האופטימלית אמNONם לא משתמשים בידע שיכל להגעה מבחינה צעדים עתידיים, אך הם הרבה יותר פשוטים למימוש ולאימון.

באופן מעט יותר פורמלי ניתן לנוכח את ההבדל בין הגישות כך: גישה Model-based learning מנסה למצוא את הפרמטרים המגדירים את המודל $\{A, \mathcal{T}, \mathcal{R}\}$ ואז בעזרתם לחשב את האסטרטגיה האופטימלית (למשל בעזרת משוואות בלמן). הגישה השנייה לעומת זאת לא מעוניינת לחשב במפורש את הפרמטרים של המודל אלא למצוא באופן ישיר את האסטרטגיה האופטימלית $(s_t | a_t) \pi$ שעבור כל מצב קבוע באיזה פעולה לנ��וט. ההבדל בין הגישות נוגע גם לפונקציית המחיר לה נרצה למצוא אופטימום.

בכל אחד משני סוג הלמידה יש אלגוריתמים שונים, כאשר הם נבדלים אחד מהשני בשאלת מהו האובייקט אותו מעוניינים ללמידה.

Model-free learning

בגישה זו יש שתי קטגוריות מרכזיות של אלגוריתמים:

- א. Policy Optimization – ניסוח האסטרטגיה כבעית אופטימיזציה של מציאת סט הפרמטרים θ הממקסם את $\mathbb{E}[\pi(a|s)]$. פתרון בעיה זו יכול להיעשות באופן ישיר על ידי שיטת Gradiant Ascent עבור פונקציית המחיר $\mathbb{E}[R(\tau)|\pi_\theta] = Q_\theta(s, a)$, או בעזרת קירוב פונקציה זו ומיציאת מקסימום עבורה.
- ב. Q-learning – שערוך $Q^*(s, a)$ על ידי $Q_\theta(s, a)$. מציאת המשערך האופטימלי יכולה להתבצע על ידי חיפוש θ שיספק את השערוך הטוב ביותר ביותר שנייתן למצוא, או על ידי מציאת הפעולה שתמקסם את המשערך:

$$a(s) = \arg \max_a Q_\theta(s, a)$$

השיטות המנסות למצוא אופטימום לאסטרטגיה הן לרוב policy-ho, כלומר כל פעולה נקבעת על בסיס האסטרטגיה המעודכנת לפי הפעולה הקודמת. Q-learning – שיטות זאת הוא לרוב אלגוריתם off-policy, כלומר בכל פעולה ניתן להשתמש בכל המידע שנცבר עד כה. היתרון של שיטות האופטימיזציה נובע מכך שהן מנסות למצוא ישר את האסטרטגיה הטובה ביותר ביזה, בעוד שאלגוריתם Q-learning רק משערך את $Q^*(s, a)$, ולעתים השעריך לא מספיק ואז התוצאה המתבקשת אינה מספקת טובה. מצד שני, כאשר השעריך מצליח, הביצועים של Q-learning טובים יותר, כיון שהשימוש במידע על העבר מנוצל בצורה יעילה יותר מאשר אלגוריתמים המבצעים אופטימיזציה של האסטרטגיה. שתי הגישות האלה אינן זרות לחלווטין, וישנם אלגוריתמים שמנוסים לשלב בין הרעיונות ולנצל את החזקוות והיתרונות שיש לכל גישה.

Model-based learning

גם בגישה זו יש שתי קטגוריות מרכזיות של אלגוריתמים:

- א. Model-based RL with a learned model – אלגוריתמים המנסים ללמידה אין את המודל עצמו והן את ה-Value function או את האסטרטגיה π .
- ב. Model-based RL with a known model – אלגוריתמים המנסים למצוא את ה-Value function ו/או את האסטרטגיה כאשר המודל עצמו נתון.

ההבדל בין הקטגוריות טמון באתגר אותו מנסים להתמודד. במקרים בהם המודל ידוע, הממד של אי הווודאות לא קיים, ולכן ניתן להתמקד בביטויים אסימפטומטיים. במקרים בהם המודל אינו ידוע, הדגש העיקרי הוא על למידת המודל.

11.2 Common Approaches

לאחר שסקרנו בפרק המבוא את הבסיס המתמטי של בעיות RL והציגו את משוואות בלמן ופתרון, בפרק זה נציגGISות שונות להתמודדות עם בעיות RL עבורן פתרונות אלה אינן מספקים – או מפני שהמודל אינו ידוע או מפני שהואBig-Scale גדול יותר מזה שניין לפתור באמצעות משוואות בלמן.

11.1.2 Monte-Carlo Methods

האלגוריתם הראשון אותו נציג הינו Monte Carlo, והוא מציע דרך לשערך את ה-Value function (בל' לדעת את המודל (כלומר האלגוריתם הינו **Model-Free**). ראשית נסביר בקצרה מהו אלגוריתם Monte Carlo ואז נראה כיצד ניתן לישם אותו בבעיות RL.