

1. Introduction

1.1 What is Machine Learning?

1.1.1 The Basic Concept

Artificial Intelligence (AI)

בינה מלאכותית הינה תחום בו תוכנות מחשב או מנגנון טכנולוגי אחר מחקה מנגנון חשיבה 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 חשיבה אנושי אך הן לא מתוחכמתות מעבר لما שתכננו אותן בהתחלה. שואב רובוטי הידוע לחשב את גודל החדר ואת מסלול הינוקי האופטימלי פועל לפי פרוצדורה ידועה מראש, ואין בו תחכם מעבר לתכונות הראשוני שלו. לעומת זאת תוכנה היודעת לסנן רישומים באופן מסתגל, או להמליץ על Shirims בגין מזיקה בהתאם לסוגנון של המשתמש, משתמשת במבנה מלאכותית ברמה גבוהה יותר, כיוון שהן לומדות עם הזמן דברים חדשים.

המונה בינה מלאכותית מתייחס בדרך כלל למערכת שמקהה 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 מעולם 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 סרטים הם רואים, איזה צ'אנרים ומתי. כדי שמש צופים ומעט סרטים, 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 במאצע, מה גיל הצופה ועוד. מערכת הבניה באופן קלאסי אינה מסוגלת להתמודד עם כמויות המידע הקיימות, וכמויות הכללים שנדרש לחושב עליהם מראש היא עצומה ומורכבת לחישוב.

נתבונן על דוגמא נוספת – מערכת לניווט רכב. ניתן להגדיר כלל פשוט בו אם משתמש יצא מTEL אביב 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 תחומי בינה מלאכותית, הבא להתמודד עם שני האתגרים שתוארו קודם – יכולת לתוכנת מערכת על בסיס מסוות של נתונים וpermeters, וחיזוי דברים חדשים כתלות בpermeters רבים שיכולים להשנותם עם הזמן. מנגנוני ML מנהלים כמויות אדירות של DATA ומנשות להציגו לתצאה. 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 פקטורי השתנה במהלך הדרך ומדווח הוא נכשל בחיזוי (למשל: הcabש בן ארבעה נתבים, אבל במקטע מסוים הוא מצטמצם לאחד וזה מייצר עיכוב, וזה עיכוב קבוע ברוב שעות היממה 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 אוリー את המשקל של הטערטורה בחוץ. וכך באפין חזרתי האלגוריתםשוב ושוב מקבל קלט, מוציא פלט ובודק את התוצאה הסופית. לאחר מכן הוא בודק היכן הוא טעה, משנה את עצמו, מתקן את המשקל שהוא נותן לפקטוריים 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 שהוזנו להם על ידי בני אדם. נניח, גוגל החלטה שמי שצופה בסרטוני יוטיוב של *Unboxing* הוא Intent��度 של רכישה. בהמשך הדרך, 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, לחת משקל גדול יותר לפרמטרים טובים ולהזניח פרמטרים 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 ביוטיוב, לזהות מתוך הסרטון, האודיו, תיאור הסרטון ומילוי המפתח וכו' – איזה סוג סרטון זה. יתרון שאחרי מיליארדי ציפוי הסרטונים, האלגוריתם מצליח למצאו קשר בין סוג מסוים של סרטונים לבין פעולות כמו רכישה באתר. באופן זהה, גוגל מזינה את האלגוריתם בכל הפעולות שהמשתמש מבצע. המיללים שהוא קורא, המקומות שהוא מסתובב בהם, התמונות שהוא מעלה לענן, ההודעות שהוא שולח, כל מידע שיש אליו גישה. הכל נשפר תוך מאגר הנתונים העצום בו מנסה גוגל לבנות פרופילים ולמצוא קשר בין הסיכוי שלו לרכושו או כל פעולה אחרת שהוא לה. לזהות.

המכונה המופלאה זו לומדת כל הזמן דברים חדשים ומנסה כל הזמן למצוא הקשיים, להזות תוצאה, לבדוק אם היא הצלחה, 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 אינטלקט פרטום. 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 מנגנוני ML המבוססים על דатаה מסויד ומתיויג כמו Netflix, עם כל המאפיינים של הסרטים אבל גם עם המאפיינים של הצופים (מדינה, גיל, שעת צפייה וכו'). לעומת זאת יש מנגנוני ML שמקבלים טיפה יותר חופש ומתבססים על מידע חלקי מאד (יש להם מידע על כל עלי הסרטים, אבל אין להם מידע על הצופה). מנגנונים אלו לא בהכרח מנסים לבנות מנגנון המלצות אלא מנסים למצוא חוקיות בנתונים, חריגות וכו'.

כך או כך, המערכת הבוחר הזה הקורי ML בנייני אלגוריתמים שונים המيونנים בניתוח טקסט, אלגוריתמים אחרים המתמקדים בעיבוד אודיו,أكلת המנתחים היסטורית גלישה או זיהוי מטור דף ה-Web בו אtmp צופים ועוד. عشرות או מאות מנגנונים כאלה מסתובבים ורצים ובונים את המפה השלמה. ככה רוב רשות הפרסום הגדולה עובדות. ככל שהמכונה של גוגל/פייסבוק תהה 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 משימות באופן מדויק ככל הנitin על דטה חדשה 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 , ומ Chapman את האלגוריתם שמבצע את המיפוי $Y \rightarrow X: g$: בצורה הטובה ביותר, ככלומר בהינתן דוגמא חדשה x , המטרה היא למצוא עבורה את ה- y 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 עלייהן מידע כלשהו המבחן ביןיה. 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 נאסף עם הזמן. ישנם סוכנים הנמצאים בסביבה מסוימת ומעבירים מידע למשתמש, והוא בתורו למד אסטרטגיה בה הסוכנים ינקטו בצעדים הטוביים עבורם.

האלגוריתמים השונים של הלמידה מתחלקים לשתי קבוצות – מודלים דיסקרימנטיביים המוצאים פلت 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 מה שהם רואו וליצור מידע חדש על בסיס הדוגמאות שנלמדו.

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

מגון התחומיים בהם משתמשים בכללים של למידה הוא עצום, עד כדי כך שכמעט אין תחום בו לא נכנס השימוש באלגוריתמים לומדים. דוגמאות בולטות למשימות באלגוריתמים לומדים: סיווג, רגרסיה (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 בעיקרם על שלושה ענפים מתמטיים; אלגברה לינארית, חישוב דיפרנציאלי והסתברות. פרק זה נציג את העקרונות הנדרשים בלבד, ללא הרחבת, 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, $(+, +)$ היא מרחב וקטורי אם הוא מקיים את התכונות הבאות:

- (I) קיימים 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 הן 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 מתאר את מיקום האיבר. מרחב זה מסומן ע"י \mathbb{F}^n .
נראה שמרחב זה הוא אכן מרחב וקטורי:

חיבור וקטוריים:

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

וקטור אפס:

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

כפל בסקלר:

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

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. פולינומיים:

פולינומיים 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 להראות 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:

$\hat{A}, \hat{B} \in \mathbb{F}^{n \times m}$ כאשר A_{ij}, 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 לא לעיל).

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

• כפל בין שתי מטריצות:

$\hat{A}, \hat{B} \in \mathbb{F}^{k \times m}$ 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:

$$\hat{A} \cdot \hat{B} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} + \dots + A_{1k}B_{k1} & \dots & A_{11}B_{1n} + \dots + A_{1k}B_{kn} \\ \vdots & \ddots & \vdots \\ A_{m1}B_{11} + \dots + A_{mk}B_{kn} & \dots & A_{n1}B_{1m} + \dots + A_{nk}B_{km} \end{pmatrix}$$

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 $\hat{B}\hat{A}$ וגם הכפל $\hat{A}\hat{B}$, אולם יתכן ש- $\hat{A}\hat{B} = \hat{B}\hat{A}$.

- **שחלוף (transpose)**:

החלפת שורות בעמודות, או 'סיבוב' המטריצה. נניח מטריצה $\hat{A} \in \mathbb{F}^{n \times m}$ איזה 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 היא בממ"ד $n \times m$. בנוסף ניתן להוכיח כי מתקיים:
 \hat{A}^T של וקטורי שורה, נתונים וקטורי 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:

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

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

- **מטריצה הופכית:**

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

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

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

- **מטריצה צמודה/hermitian:**

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

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

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 המורכבים 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.

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

קטgor \vec{b} , הינו וקטור عمودה, שアイבוריו הם האיבר החופשי.

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

מכפלה פנימית, נורמה, אורותוגונליות

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 המקבלת 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 ליקים מספר תכונות.

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

$$\begin{aligned} \langle \vec{v} + \vec{u}, \vec{w} \rangle &= \langle \vec{v}, \vec{w} \rangle + \langle \vec{u}, \vec{w} \rangle && \bullet \\ \langle \lambda \vec{v}, \vec{u} \rangle &= \lambda \langle \vec{v}, \vec{u} \rangle && \bullet \\ \overline{\langle \vec{v}, \vec{u} \rangle} &= \langle \vec{u}, \vec{v} \rangle && \bullet \\ \langle \vec{v}, \vec{v} \rangle &\geq 0 && \bullet \end{aligned}$$

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

א. מכפלה סקלרית על מרחב הווקטורי הגיאומטריים:

נתונים $\vec{v}, \vec{u} \in \mathbb{F}$, וקטורים גיאומטריים מסדר n מעל שדה המספרים המורכבים. מכפלה פנימית בין שני וקטורים אלו, הנקראת 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 v .

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:

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

שוויון מתקיים אך ורק עבור וקטור האפס; $0 = \vec{v} \Leftrightarrow \| \vec{v} \| = 0$.

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):

$$\| u \| \cdot \| v \| \leq | \langle u, v \rangle |$$

כאשר $\langle u, v \rangle$ הינה המכפלה הפנימית בין שני הווקטורים, המוגדרת מעל הטבעיים כר: $y_i \cdot u_i, v_i$, והbij'וי $\| u \| \cdot \| v \|$ 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 $v \in V$ ($x, y, z \in \mathbb{R}$) הנורמה מוגדרת ע"י $\| v \| = \sqrt{x^2 + y^2 + z^2}$.

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

אורותוגונליות

הגדרת מכפלה פנימית מאפשרת לנו להגדיר אורותוגונליות (או אנכיות) של שני וקטורים למרחב מכפלה פנימית מסוים. שני וקטורים $v, w \in \mathbb{C}^n$ 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 הזווית 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: $\| u \| \cdot \| v \| \leq | \langle u, v \rangle |$, ולכן $\cos(\theta) \leq 1$. כיוון שכך, 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 לכך. דבר זה נעשה על ידי תהליך גרם-שמידט (gram-schmidt).

שני וקטורים אורתוגונליים יסומנים על ידי \perp . עבור וקטורים אורתוגונליים מתקיימות התכונות הבאות:

- אם $v \perp u$, אז $uv = 0$.
- אם $v \perp u$, אז לכל סקלר λ גם $v \perp \lambda u$.
- אם $v \perp u$ וגם $v \perp w$, אז $v \perp (u + w)$.
- אם וקטור אורתוגונלי לקבוצה של וקטורים אזי הוא גם אורתוגונלי לכל צירוף לינארי שלהם (נובע ממשת התכונות הקודמות).

וקטורים עצמיים וערכים עצמיים

תהי $A \in \mathbb{F}^{n \times n}$ מטריצה ריבועית, וקטור $v \in \mathbb{F}^n$ נקרא ערך עצמי של A אם $Av = \lambda v$ נקרא הווקטור העצמי המתאים לו מתקיים:

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

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

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

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

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

ערכים האלכסון של $\Sigma = \sum_{ii} \sigma_i$ – 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:

- הווקטורים הסינגולריים השמאליים של M הם וקטורים עצמיים של $M M^*$.
- הווקטורים 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 להציגו 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: $f(x) = y$. 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 כר: $Y \rightarrow X$, 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 גזירה.

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. בנקודות אלו, 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 שלו, 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 מממרחב 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 שתי הקבוצות. איחוד של הקבוצות A ו- B הוא אוסף האיברים המופיעים לפחות פעם אחת בקבוצות 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 האיברים המופיעים בשתי הקבוצות. חיתוך של הקבוצות A ו- B הוא אוסף האיברים 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 איברים משותפים:

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

מאורע משלים – מאורע המכיל את כל האיברים 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:

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

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

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

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 מתורגם 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 נהיה 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 השירט 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 במרחב המדגם גדולה או שווה ל-0.
- סכום ההסתברויות של כל המאורעות במרחב 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 $\{Y \leq a\}$ 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:

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

תכונות ופרמטרים עבור משתנה מקרי

תוחלת – ממוצע משוקל של כל הערכים האפשריים, כאשר כל ערך מוכפל בהסתברות שלו (במקרה הבדיקה – סכום ובמקרה הרציף – אינטגרל):

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

תכונות:

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

שונות – ממד פיזור הערכים בגין ממוצע 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 לבחירתם 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: $\rho(X, Y) = \frac{cov(X, Y)}{\sqrt{Var(X)Var(Y)}}$. המקדם מקיים: $1 \leq |\rho| \leq 1$.

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

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

פונקציה יוצרת מומנטים (התמורה לפולס של פונקציית הצפיפות – $M_X(t) = \mathcal{L}\{f_X(-t)\}$)

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

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

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

המומנט הראשון הוא התוחלת והמומנט השני הוא כפעת זיהה לשונות (מומנט השני הוא $[X^2] - E[X]^2$, לעומת זאת המומנט השלישי הוא התוחלת והמומנט הרביעי הוא כפעת זיהה לשונות (מומנט הרביעי הוא $E[X^4] - 3E[X]^2 + 2E[X]^2$).

התפלגות מיוחדות (בדיד)

ישנן כל מיני התפלגות מיוחדות, שמשמשות בפועל בכל מיני מקרים ויש להן נוסחאות ידועות.

התפלגות ברנולי: ($X \sim Ber(p)$)

ניסוי בעל שתי תוצאות אפשריות "הצלחה" או "כשלון". המשטנה 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. נסמן $b-k$ סיכוי להצלחה בניסוי בודד וב- k סיכוי לכישלון בניסוי בודד.

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

צריך 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. n מסמן את הסתברות 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 מיוחדות:

$$1) \text{ "תכונת חוסר זיכרון": } P[X = (n+k)|X > k] = P(n)$$

$$2) \text{ ההסתברות שיעברו } k \text{ ניסויים ללא הצלחה: } P(X > k) = q^k$$

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

התפלגות אחידה: ($X \sim U[a, b]$)

בהתפלגות זו לכל תוצאה יש את אותה הסתברות. הערכים המתקבלים בהתפלגות החל מ- a ועד b הינם בקפיצות של יחידה אחת (לדוגמה 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: (λ) $X \sim poi(\lambda)$

התפלגות זאת מתאפיינת במספר אירועים ליחידת זמן ליצב האירועים ליחידת זמן הנבחרת.

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

יש לשימוש לב שכאן ההתפלגות נמצדת ליחידת זמן.

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

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

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

התפלגותBINOMIAL שלילית: (r, p) $X \sim NB(r, p)$

חוזרים 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.

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

התפלגות מיוחדות (רצף)

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

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

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 .

$$F(t) = \frac{t-a}{b-a}, f(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0, & \text{ אחרת} \end{cases} \quad \text{פונקציית הצפיפות: } b < x < a$$

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

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

התפלגות נורמלית היא ההתפלגות חשובה מאוד כיון שהיא מופיעה בהמוני מקרים. פונקציית הצפיפות של ההתפלגות הנורמלית נראה כמו פעמן, כאשר לעקומה קוראים גם עקומת גאוס. ההתפלגות הנורמלית נבדלות אחת מהשניה באמצעות הממוצע וסטיית התקן (-הפרמטרים שמאפיינים את ההתפלגות). ההתפלגות נורמלית סטנדרטית היא ההתפלגות 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:

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

המוננטים 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 שוויוןים

מרקוב

בاهינתן $0 \leq X \leq \mathbb{E}[X]$, עברו פרמטר $0 > a$ מתקיים:

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

צ'בישוב

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

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

צ'רנוף

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

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

כאשר $(t) M$ היא פונקציה יוצרת מומנטים של X .

ינט'

עבור משתנה מקרי X בעל תוחלת, עברו פונקציה קמורה $g: \mathbb{R} \rightarrow \mathbb{R}$ 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-ممדי רציף אם קיימת פונקציית צפיפות דו-מממית $f_{x,y}(s, t)$, כך שמתקיים:

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

באופן שקול מתקיימ:

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

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- $f_y(t)$):

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

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

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

סטטיטיקה היסקית

אם ידועים את סוג ההתפלגות אבל לא ידועים את מרכיביה, ניתן לאמוד את המרכיבים בעזרת מדגם. המדגם מאפשר לנו להשתמש באומדן עבור מספר מאורעות שניי ההתפלגות. דוגמא – נניח רוצם למדוד גובה של קבוצה מסוימת – כל התלמידים בבית ספר מסוים. ידוע שהגובה מתפלג נורמלית, אבל לא ידועים כאן את התוחלת והשונות. לשם כך ניתן להשתמש באומדן – פונקציה שמנסה לנתח את המאורעות ומתוך כך להסיק את התוחלת והשונות.

בנитוח נמצא מנגנון הנחה שידוע כי הערכים במדגם נלקחים כולם מתוך ההתפלגות X , השיכת המשפחה של ההתפלגות שתלויות בפרמטר אחד או יותר שאינם ידועים. (למשל בדוגמא: (μ, σ^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 (כפונקציה של הערכים שדגםנו).

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 אחרות – אומד יהיה חסר הטיה אם התוחלת של המשתנה המקרי המוחשב לפי θ שווה ל- θ 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:

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

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

אומד נראות מרבית – (MLE)

בහינתן סדרת דגימות מתוך התרפוגות עם פרמטר לא ידוע, נגדיר את פונקציית הנראות שלהן כמכפלת ההסתברויות של כל הדגימות, או "הנראות של המדגם":

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

זהו פונקציה הן של הדגימות והן של הפרמטר.

אם התרפוגות רציפה מוגדרים במקום 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 אומד נראות מקסימלי עבור θ אם $\hat{\theta} = \arg \max_{\theta} L(x_1, x_2 \dots x_n | p(\theta))$.

מכיוון \log הינה מונוטונית, למקסם את L שקול למקסם את $\log(L)$, וזה 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 בהתרפוגות פואסונית:

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

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

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

מוסבך למצוא לזה מקסימום, لكن נוציא לוג:

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

cut 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 מקבל:

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

ובודד את הפרמטר אותו מנוטים לאמוד:

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

וקיבילנו אומד עבור הפרמטר λ , וכאשר נתון סט התוצאות, פשוט נציב אותן, ונמצא מפורשות את הערך של האומד. זה בעצם תהליכי מציאת ה-*MLE*. cut 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 שבביטוי הראשון מה שיש בתוך ה- \ln זה בעצם $(2\pi)^{-\frac{1}{2}} + (\sigma^2)^{-\frac{1}{2}}$, ואז המעריך יכול לרדת מוחץ ל- $-\infty$:

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

כעת בשביל לאמוד את התוחלת יש לגזר לפי μ , וכך לאמוד את השונות יש לגזר 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 לבדוק אם עבור התוחלת האומד הוא בעצם הממוצע של המדגם. אפשר לבצע תהליכי דומה על השונות, ומקבלים 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 מתקבל אוסף של דוגמאות מתוויות במרחב d -ממדי, ומנסה למצוא משור המפריד בצורה טובה כמה שניתן בין דוגמאות האימון השויות לקטגוריות השונות.

המשמעות הנוצר באמצעות מודל SVM הינו לינארי, 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 הצד אחד של השולים נמצא דוגמאות עם label אחד, ובצד השני נמצא דוגמאות 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.

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

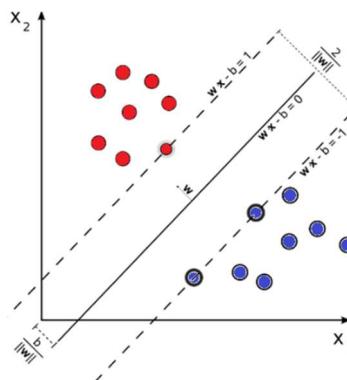
$$w^* = \operatorname{argmin}_w \left(\frac{1}{2} \|w\|^2 \right), \text{s.t. } y_i (x_i \cdot w + b) \geq 1 \quad \forall i$$

כלומר, רוצים למצאו את וקטור המשקלות w המיציר שולים $\|w\|^2$ שהדוגמאות מתוויות נכון. $\text{margin} = \frac{1}{2} \|w\|^2$ (לא מתקיים: $y_i (x_i \cdot w + b) < \frac{1}{2} \|w\|^2$) ≥ 0 .

ישנו מספר גישות למציאת המשור, ונפרט על כמה מהן.

Hard-Margin (hard SVM)

במצב הפשוט ביותר, המשווה עבר כל אחד מצדדיו של המפריד הינה פונקציה לינארית של המאפיינים וכל הדוגמאות אשר סווים נכוна. מצב זה מכונה " הפרדה קשה " בו האלגוריתם מוצא את המשור עם השול הרחב ביותר האפשרי, ולא אפשר 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 המקווקים 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 $1 = -b = \vec{x} \cdot \vec{w}$. גאומטרית, המרחק בין שני המשוריים הוא $\frac{2}{\|\vec{w}\|}$, ולכן מנת למסס את המרחק הזה, יש מהביא למינימום את $\|\vec{w}\|$. על מנת שדוגמאות האימון לא יכללו בשולים המפרידים, יש להוסיף אילוץ לכל דוגמא i , באופן הבא:

$$y_i (\vec{x}_i \cdot \vec{w} - b) \geq 1$$

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

$$\begin{aligned} \min_{w,b} & \|w^2\| \\ \text{s.t. } & y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 \quad \forall i = 1 \dots n \end{aligned}$$

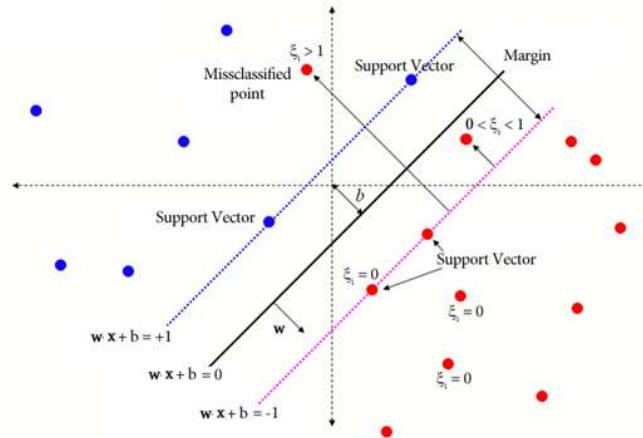
Soft-Margin (soft SVM)

הפרדה מושלמת באמצעות מישור לנארו 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 לנארו בין הקבוצות, כמו למשל שיש נקודות חרגות. 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 המסובג. בגרסת זו יש לרשום באופן מעט שונה את בעיית האופטימיזציה, כאשר מתווסף משתנה המתיחס לנקודות שאין נמצאות בסיווג המתאים להן לפני המפריד:

$$\begin{aligned} \min & \|w^2\| + C \sum_{i=1}^n \xi_i \\ \text{s.t. } & y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1 \dots n \quad \xi_i \geq 0 \end{aligned}$$

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 לשמרה 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 שאחריו 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 של הדוגמאות האימון הספציפיות אינן מסוגות נכון.



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 לנארים מוגבלים ביכולת הכללה שלהם בגלל הפשטות שלהם. לכן, כאשר לא ניתן להפריד אוסף דוגמאות באמצעות מפריד לנאר, משתמשים ב"הפרדה אל-لينארית". גישה זו מאפשרת להשתמש ב-SVM לשיווג לא לנאר, על ידי טרנספורמציה לא לנארית, כמו למשל "תעלול הגערין" (Kernel Trick). בגישה זו מבצעים מיפוי לדאטה למרחב אחר, בו ניתן למצוא עבורו הפרדה לנארית, ומילא אליה אפשר להשתמש באlgוריית 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 מסויוג לינארי. לשם כך, משתמשים בטריק קרNEL שמקבל כקלט וקטורים למרחב המקורי ומחזיר את המכפלה הפנימית (dot product) של הווקטוריים במרחב 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 היכי פשוטה, המוגדרת על ידי מכפלה פנימית של הווקטוריים. במקרה זה מרחב התכונות ומרחב הקלט זהים ונחזר לפתרון בעזרת SVM לינארי:

קרNEL פולינומי:

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

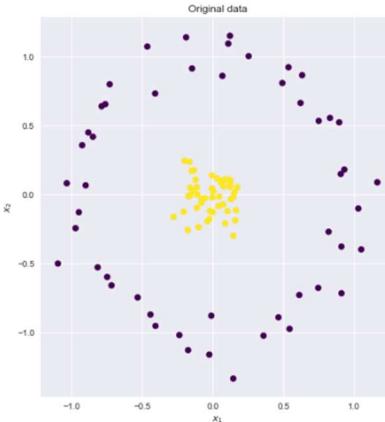
העתקה מהמרחב המקורי למרחב שמהווה פולינום ממעלה 2 $\geq d$. $0 \geq c$ הוא פרמטר חופשי המשפיע על היחס בין סדר גובה לעומת סדר נמוך בפולינום. כאשר $c = 0$, ה الكرNEL נקרא הומוגני.

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 אחר 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 בתנאי 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 מממד או בעל ממד גבוה יותר, בו ניתן יהיה להפריד בין נקודות אלה באופן לינארי. לצורך כך נבצע את הפעולות הבאות:

- א. נמפה את התכונות המקוריות למרחב הגובה יותר (מיפוי 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 נקודות אלה ניתנות 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 שקיימת 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 $(x_2)^T \varphi(x_1)x_2$. קרナル היא פונקציה K שהייכת למכפלה פנימית זו, כלומר $K(x_1, x_2) = (x_1)^T \varphi(x_2)\varphi(x_1)$. אם נוכל למצוא פונקציית קרナル המקבילה למפת התכונות שלעיל, נוכל להשתמש בפונקציה ייחד עם SVM לינארי וכך לבצע את החישובים בעילוות.

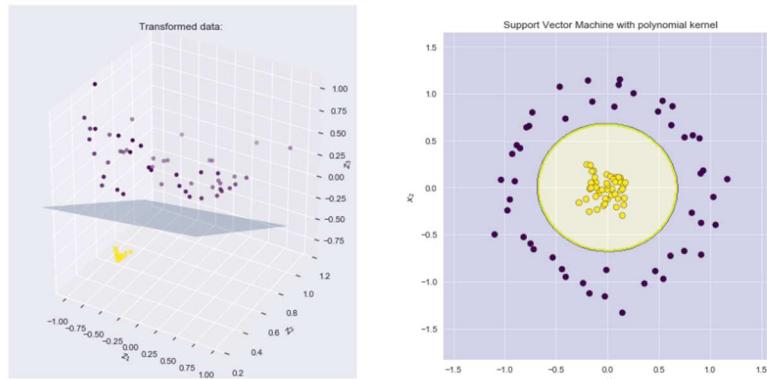
מתברר שמרחב התכונות שלעיל תואם את השימוש בקרナル פולינומי ידוע: $(x^T x')^d = (x' x)^d = K(x, x')$. נבחר $d=2$. נסמן $(x_1, x_2)^T = \alpha$ ונקבל:

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

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

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

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



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

2.1.2 Naive Bayes

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

נניח שיש מודל המקביל וקטור מאפיינים בינהירים {cab, ראש, משתעל, חום גבוה}, ומסווג האם בעל תכונות אלה חוליה בשפעת או לא. באופן כללי ניתן לומר שיש תלות בין שייעול לבין חום גבוה, כלומר 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 ההסתברות שלהם.

נתבונן בוקטור מאפיינים $(x_1, \dots, x_n) = x \in \mathbb{R}^n$, היכל להיות שיר לאחת מ-K קטגוריות $(y_k | x)$. הסתברות הפרIORיות $p(y_k | x)$ 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 שעבורו הביטוי $p(y_k | x)$ הוא מקסימלי. באופן פורמלי ניתן לנסח 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 שעבורו המונה מקסימלי. לפי כלל השרשרת מתקיים:

$$\begin{aligned} p(y_k, x) &= p(y_k, x_1, x_2, \dots, x_n) = p(y_k, x_1 | y_k, x_2, \dots, x_n) \cdot p(y_k, x_2 | y_k, x_3, \dots, x_n) \cdot \dots \\ &= p(x_1 | y_k, x_2, \dots, x_n) \cdot p(x_2 | y_k, x_3, \dots, x_n) \cdot \dots = p(x_1 | y_k, x_2, \dots, x_n) \cdot p(x_2 | y_k, x_3, \dots, x_n) \cdots p(x_{n-1} | y_k, x_n) \cdot p(x_n | y_k) \\ &= p(x_1 | y_k) \cdot p(x_2 | y_k) \cdots p(x_{n-1} | y_k) \cdot p(x_n | y_k) \\ &= p(y_k) \prod_{i=1}^n p(x_i | y_k) \end{aligned}$$

בביטוי זה כל האיברים ידועים, ולכן ככל שנוטר זה רק להציב את הנתונים ולקבל את y_k עבורו ביטוי זה הינו גדול:

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

בדוגמה שהובאה לעיל, המאפיינים קיבלו ערכים בדים, ולכן ניתן לחשב את ההסתברות המותנית של כל מאפיין $(x_i | y_k)$ על ידי ספירת כמות הפעמים שמופיע כל מאפיין באוכלוסייה הנדגמת ולהלך בגודל המדגם. עבור ערכים רציפים (כמו למשל מחיר מניה, גובה של אדם וכדו'), אין אפשרות לחשב כך את ההסתברות המותנית. במקרים כאלה יש להניח התפלגות מסוימת עבור המדגם, וליחסב את הפרמטרים של התפלגות שיטות שונות (למשל בשיטת מינימום שגיאה מרבית – 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 שגרור אחרית. אם התפלגות היא לא נורמלית, ניתן להשתמש באלגוריתם Kernel density estimation עבור שגרור התפלגות. גישה אחרת להערכתם 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 של תליות, 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.Cutet 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 שלهن קשר ישיר):

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

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

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:

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

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

הчисרונות בשימוש במסווג בייסיאני נאיבי בעיות מולטינומיות נועז בכך שיש הרובה צירופים שלא מופיעים יחד בסע האימון, ולכן הנראות שלהם תמיד תהיה 0, מה שפוגם באמינותות 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 שנקרוא CNB (complement naive Bayes). לפי אלגוריתם זה, במקומות ללקחת את $p(x_i|y_k)$ ניקחים את המינימום של הפונקציה ההופכית:

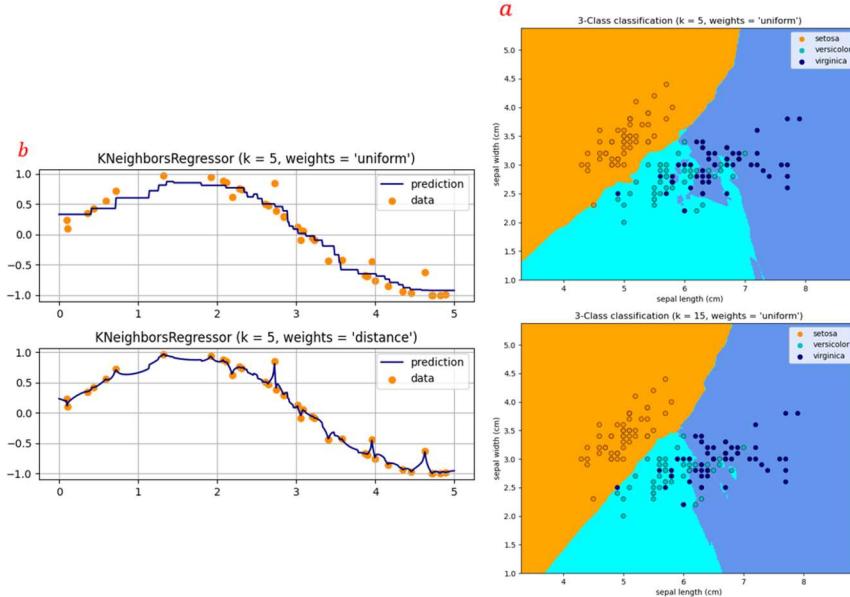
$$\arg \min_k p(y_k) \prod_{i=1}^n \frac{1}{p(x_i|y_k)}$$

שימוש באלגוריתם זה הוכח כיעיל במקרים בהם הדטה אינם AMAZON והביצועים של מסווגים בייסיאנים אחרים (גיאוסיאני 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 להעיות סיוג (שיוך נקודה חדשה למחלקה מסוימת) והן לעביעות רגסיה (נתינת ערך מאפיין לנקודה חדשה). האלגוריתם הינו מודל חסר פרמטרים, והוא מבצע סיוג לנ נתונים בעזרת הכרעת הרוב. עבור כל נקודה במדגם, המודל בוחן את ה-label של K nearest הנקודות הקרובות אליו ביותר, ומסווג את הנקודתה לפי ה-label שקיבל את מרבית הקולות. מספר הנקודות הקרובות, K, הוא היפר-פרמטר שנקבע מראש.

אלגוריתם השן הקרוב הוא אחד המודל הנפוצים והפושים ביותר בלמידה מוכנה, ואומר בכך שהוא מתאים גם לבעיות גרגסיה. המודל יפעל בצורה דומה בשני המקדים, כאשר ברגסיה 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 שן מהצפית בצורה שווה (uniform), וכן לתת משקל שונה שונה לכל שן בהתאם למראק של מהנוקודה אותה רצים לחשב, כך שכל ששן מסויים קרוב יותר לנוקודה אותה רצים לחשב כך הוא יותר ישפיע עליה, ביחס של הופכי המראק בין השן לנוקודה (distance).



איור 2.4 (a) סיווג בעזרת אלגוריתם N-NN-K: מסוגים את המראק לאזוריים בהתאם ל-K השכנים הקרובים ביותר, כך שאם תבוא נוקודה 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 כל שן מהנוקודה אותה רצים לחשב.

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

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

קושי נוסף הקיים במודל הוא הצורך בבחירה K-הנקון, מטלה שעשויה להיות לא קללה לעתים. בכל שימוש של אלגוריתם השן הקרוב, K הינו היפר-פרמטר שצריך להיקבע מראש. היפר פרמטר זה קובע את מספר הנוקודות אשר האלגוריתם יתחשב בהן בעת בחירת סיווג התצפית. בחירת היפר-פרמטר קטן מדי, לדוגמה K=1, יכולה לגרום למצב בו המודל מותאם יתר על המידה לנוקדי האימון, מה שmobivalent לדיווק גבוה נתונים האימון, ודיווק נמוך נתונים המבחן. מן העבר השני, כאשר K גבוהה מדי, למשל K=100, נוצר מצב ההופך – מודל שמתהשך 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 לתוצאה מוטעית, ולכן כדאי להימנע מתיקו כדי לבחור K אי-זוגי.

כמו אלגוריתמים רבים מבוססי מרחק, אלגוריתם השכנן הקרוב רגיש לערכים קיצוניים (Outliers) ושימוש 알고יריהם ללא טיפול בערכים קיצוניים עשוי להוביל לתוצאות מוטעות. מלבד זאת, חשוב לנормל את הנתונים לפני שימוש במודל. הסיבה לכך היא שהאלגוריתם מבוסס מרחק; במצב זה, יתרה מכך קיימת בין-תצפויות אשר עשויים להשפיע על החלטת המודל, למרות שהmarkerים אלו הם חסרי משמעות לצורך הסיווג. דוגמא לכך היא משתנה שעשויה לשמש ביחסות מידת שונות (מיילומטרים). ההחלטה האם להשתמש בקילומטרים או במילים עלולה להטוט את תוצאות את המודל, למרות שבפועל לא השתנה דבר.

השיטה הנפוצה ביותר לממדית 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 לכך להפוך מחרוזת אחת לחרוזת שנייה, ובכך למדוד את הדמיון ביניהן.

לפני שימוש 알고יריתם השכנן הקרוב, יש הכרח לוודא שהמחלקות מאוזנות. במידה ומספר דוגמאות האימון באחת המחלקות גבוהה מאשר בשאר המחלקות, האלגוריתם יטה לטעוג למחלקה זאת. הסיבה לכך היא שבשל מסגרן הגדל, מחלוקת זו צפואה להיות נפוצה הרבה יותר בקרב K השכנים של כל תצפית. הדבר עשוי להוביל לתוצאות מוטעות, ולכן יש לוודא מראש שacky 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 – $p(y = k) = \frac{\mathbb{1}_{y=k}}{n}$. 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:

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

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

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

על ידי גזירה והשוויה 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 המקירה בו מטריצת covariance היא אלכסונית, כלומר אין תלות בין משתנים שונים, מקבל המסויוג הביסיאני הגאוסיאני (תוצאה זו הינוית כיוון שהמסויוג הביסיאני מניח שבاهינתן סיווג של אובייקט מסוים אין יותר תלות בין המשתנים).

עבור המקירה הבינארי, בו $\{0, 1\}$, מקבל 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 מקירה פרטיה של Quadratic Discriminant Analysis, בו מניחים כי מטריצת covariance זהה לכל ה-labels, $\Sigma = \Sigma_k$. במקירה זה מקבל:

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

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 ל-label המרחק על פי מטריקת מהלוביס הוא הערך קטן. על ידי גזירה והשווהה ל-0 מקבל 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:

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

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

ניתן לסמך:

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

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

ומתתקבל מטוג לינארי (ומכאן השם של האלגוריתם):

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

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

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

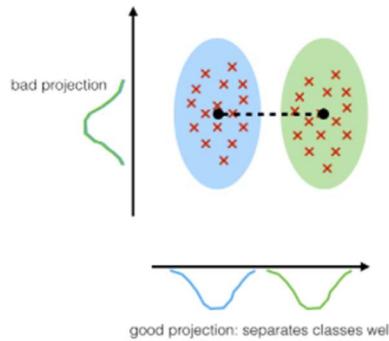
$$b = \frac{1}{2} (\mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1) + \log \frac{p(y = 1)}{p(y = 0)}$$

והסיגו 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 שטחיתת ה covariance (covariance) זאה לכל ה-labels, מה שיכל לגרום לשגיאות בסיגו. כדי להתמודד עם הבעיה השנייה ניתן להשתמש באלגוריתמים המנסים למצאו את מטריצת covariance- שתביאו לסיגו 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

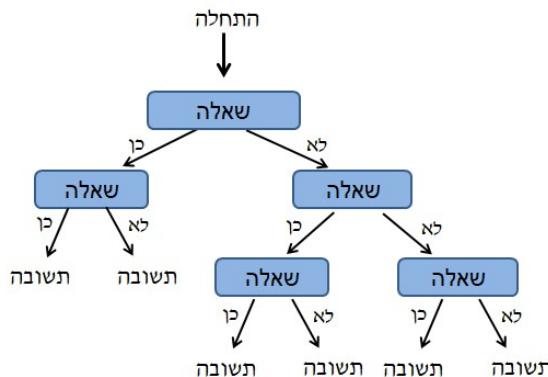
1. הקדמה

עדי החלטה הינו אלגוריתם לומד היכול לשמש הן לביעות סיגו והן לביעות רגסיה. באופן כללי, עדי החלטה לוקחים את המשטנה שברצוננו להסביר (משתנה המטריה/הхиיזי), ומחלקים את המרחב שלו לקבוצות (segments). לכל קבוצה של סט האימון מחשבים "ממוצע" (Mean) או "שכיח" (Mode), וכאש מתקבלת תצפית חדשה משיערים אותה לקבוצה בעלת הממוצע או השכיח הדומה ביותר. לאחר וכליל הסיגו לקבוצות השונות יכולים להציגו בצורה עז, אלגוריתם זה נקרא "עדי החלטה".

הגישות השונות בעדי החלטה פשוטות וrintoaitiviyot להבנה, אולם הן לא מצלילות להתחרות במדדי הדיקוק של מודלים אחרים של Supervised Learning. לכן, בפרקים הבאים נציג שיטות ensembles, בהן מתבצעת בנייה של כמה עדי-החלטה במקביל, אשר משלבים 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 עבורו 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) – הקצוות של העצים – התשובות. נקראים גם terminal nodes.
- (ענף) – חלק מתוך העץ המלא (תת-עץ)
- Depth (עומק) – מספר ה-nodes במסלול הארוך ביותר בעץ.

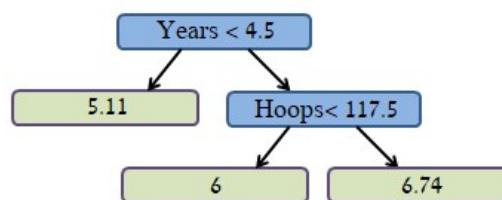
יסודות של עצי החלטה

עצים רגסיה:

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

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

תחילה נסיר תכיפות לא ערכים עבור המשתנה המוסבר שלנו, הלא הוא "שכר" ובנוסף נבצע על אותו משתנה Log transform בכך שהוא ככל הנין בקרוב להתפלגות נורמלית (עקומת גאוס/פעמון).



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

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

שתסוג לעלה מסוים תקבל את הערך הממוצע של התכפיות שלם בסיסם נבנה העץ. בהמשך הפרק יסביר כיצד לבנות את העץ וכייד לקבוע את כללי הפייטול 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:

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

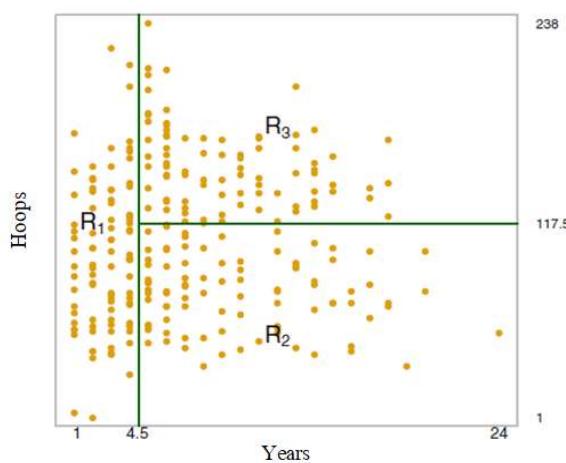
בתואם לדוגמה של העץ, הקבוצות R_1, R_2, R_3 מכונות בשם nodes terminal, או "עלים" של העץ.

אפשר לפרש את עץ הרגרסיה שבדוגמה הצורה נוספת: $Years$ הוא הפרמטר המרכזי ביותר בקביעת 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 משפיע על השכר. כאמור, כל עוד לשחקן אין 5 שנות ניסיון, כמות הסלים היא יחסית שלית ביחס לחומר הניסיון עבור קביעת שכרו. לעומת זאת, בקרב שחוקנים 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 5 שנות ניסיון אך קלעו פחות סלים.

עץ הרגרסיה שהובא בדוגמה מפשט קצת "יתר על המידה" את הקשר "האמת" בין $Salary$, $Years$ ו- $Hoops$, והחיזוי 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 שהחיזוי 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 נפרסו על פני מרחב דו ממדי, וחולקו בעזרת קו החלטה בהתאם ל- $Years$. כתעת נגידיר את תהליך החלוקה והחיזוי בשני שלבים:

1. מחלקים את מרחב הערכים האפשריים של המשתנה אותו רוצים לחזות ל- J אזורים נפרדים R_1, \dots, R_J . כתלות בפרמטרים השונים X_1, \dots, X_n .
2. לאחר החלוקה, כל תכפית 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 המשטנים של תציפות האימון באזורי j , $-y_i - \hat{y}_{R_j}$ הוא המרחק בין כל תצפית לבין הערך הממוצע באזורי זה. 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, לרוב משתמשים באלגוריתם חמדן (greedy recursive binary splitting) אשר עובד "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 ההתחלת, 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 בנייה הטוב ביותר עבור השלב הנוכחי, מבלי להתחשב כמה שלבים קדימה. בגין זה 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 תנעה, ומתרן ניסיון לעקוב את הפקק נבחרת הפניה הראשונה שנראית פעות פוקוקה, מבלי להתחשב בפקק שעשו לבוא בהמשך. כਮובן שפניה זו לא בהכרח תוביל לדרכו 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:

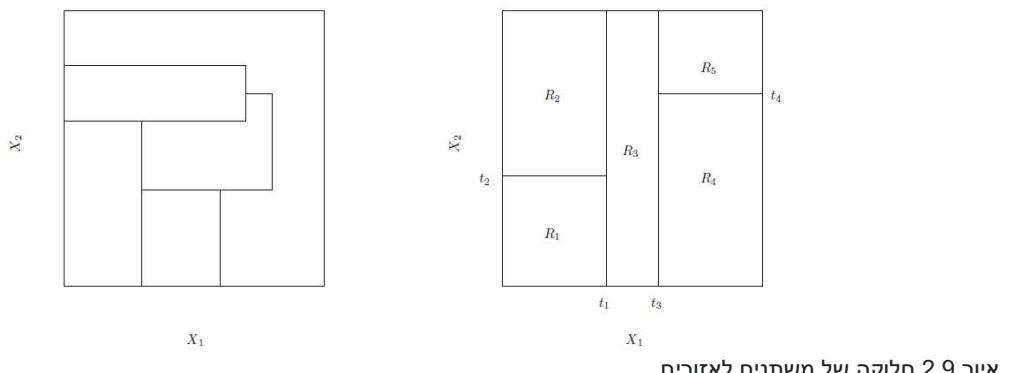
$$\begin{aligned} R_1(j, s) &= \{X | X_j < s\} \\ R_2(j, s) &= \{X | X_j \geq s\} \end{aligned}$$

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

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

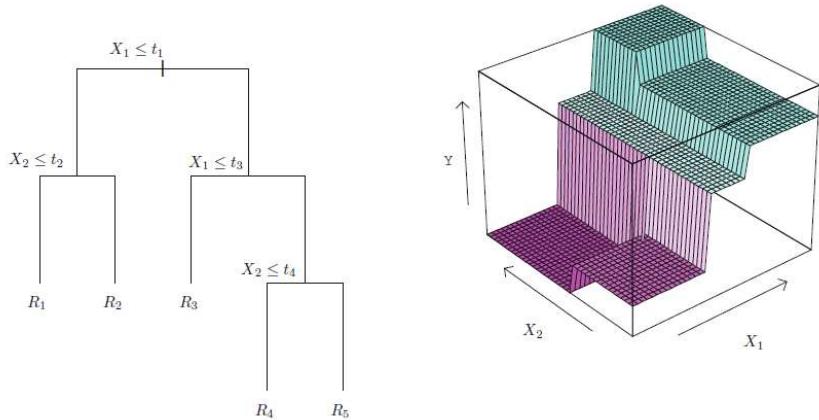
מציאת ערכי s ו- j שמבאים למינימום את המשוואה יכולה להתבצע די בקלות באמצעות הפרמטרים לא גודלה מייד, אך זה תהליך שעלול להיות די סבוך כשייש הרבה פרמטרים.

למעשה תהליך זה מייצר ענף אחד ש יוצא מה-root, ולו 2 עליים. בעת מבצעים את התהליך שוב, מתוך מטרה למציאת האלגוריתם הבא שמביא 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.

באזור המצויר ניתן לראותו 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 להוור במערכות פיצול ביןארי רקורסיבי, כיוון שהיא מורכבת מידי ואינה תואמת את הגישה החמדנית שרצה "חלוקת פשוטה ו מהירה".



איור 10.2 תיאור אזרחי חלוקה בעץ ביןארי. מצד שמאל מוצג העץ התואם לחלוקה מהאיור הקודם, מצד ימין ישנה פרספקטיביה רחבה כיצד חיזויים מתבצעים באמצעות העץ.

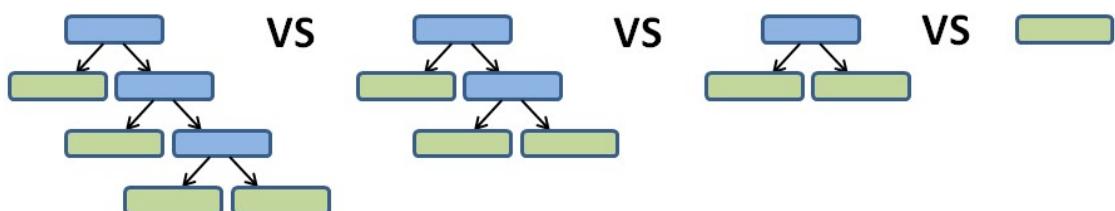
3. גיזום (pruning):

התהילך שתואר משקף את התהילך הקלאסי של ייצור 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 בתהילך האימון עשוי להיות "מורכב מידי" ומוטאם יתר על המידה לנ庭וני האימון, מה שפוגע ביכולת ההכללה שלו. לעומת זאת מודל יותר בעל פחות פיצולים (כלומר פחות קבוצות R_j , ..., R_1) יתנו חיזויים בעלי הטיה (Bias) גדול יותר בהשוואה לאלטרנטיבתה הראשונה, אך כנראה יוכל לשנות קטנה יותר בין סט המבחן, ובנוסף מודל זה יהיה קל יותר לפרשנותו.

גישה פשוטנית בכך כי התמודד 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 יבוצע. באופן זהה העצים שיתקבלו יהיו קטנים יותר ויש פחות Chsh�-over-fitting. החיסרונו בגישה זו נועז בכך שהוא קצרת-ראיה. יתרון מכך בו יש פיצול בעל תרומה קטנה יחסית אך הוא יכול לסייע אחר בעל תרומה גדולה RSS-ב-RSS בהמשך. שיטה זו מדגלא 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 ב כדי להימנע מ-over-fitting. את מקומם של הענפים שהסרנו יתפוז עליה בודד שמקבל ערך ממוצע המתחשב במספר גדול יותר של ציפוי. כמובן שצד 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 בגישה הנקראת subtree pruning (ידועה גם בשם Cost complexity pruning). בגישה זו, במקום להתחשב בכל subtree כלם שונים המרכיבים את T_0 – העץ המקורי, אנו משתמשים על רצף מסוים של subtrees שהם למעשה חלקים שונים המרכיבים את T_0 – העץ המקורי – שנבנה בהתאם:

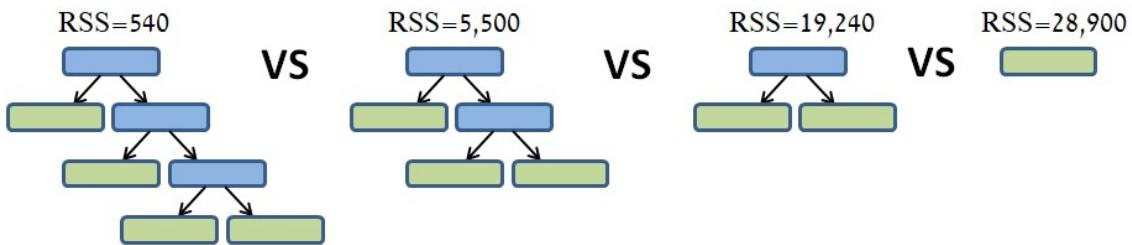


איור 11.2 חיפוש תת עץ אופטימי ביחס ל- T_0 . מתחילה מ- T_0 (העץ השמאלי) וכל פעם מורידים פיצול יחיד עד שmaguisim ל-root.

cut ממחשבים עבור כל אחד מרכץ העצים שהתקבל את ה-RSS שלו. לפני שנסביר באיזה עץ לבחור, נסכם את השלבים שבוצעו עד כה:

- a. בניית עץ רגסיבי 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 .
- b. כל עלה בעץ מקבל את הערך הממוצע של צפיפות הפורט שבאותו עלה, ועבור כל עלה פיצול מחושב ה-RSS.
- c. סכימת כל ערכי RSS שקיבלו בכל העליים. הסכום שהתקבל הוא RSS של כל העץ T_0 .
- d. cut מביצעים את שלבים א-ג ל-subtree הבא ברצף. זהו עץ-משנה שכמעט זהה לעץ המקורי, למעט שני עלים שנגזרמו מהעץ הקודם. 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 לבדוקין כי בכל פעם שענף מסויים הוסר, ה-RSS שהתקבל נהיה גדול יותר לעומת subtree המקורי. תוצאה זו הגיונית, שהרי כל פיצול במקור נועד להקטין את השגיאה באמצעות הקטנת ה-RSS, ואילו גיזום עשויה 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.

ה. cut יש לבחור את 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 רגולרייזציה 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 שהוא מ- α -fitting. 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:

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

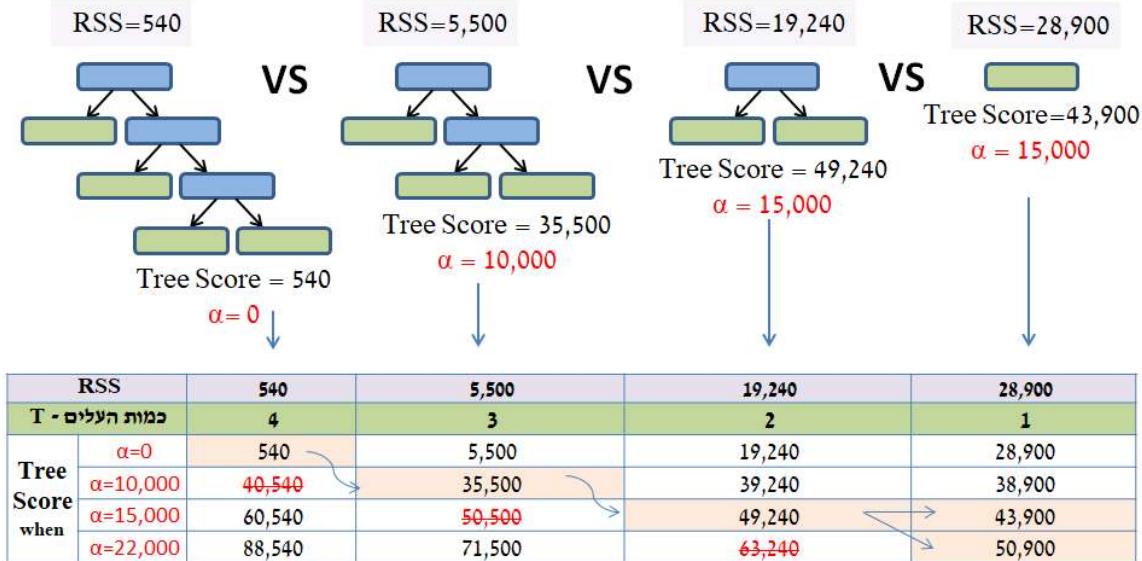
$$\text{Tree score} = \text{RSS} + 0 \cdot T = \text{RSS}$$

滿ילא 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:

- i. עבור $\alpha = 0$ הציון של T_0 יהיה 40,540. וכך שווה לנו לגוזם 2 עליים ולקבל עבור אותו α ציון נמוך יותר מ-40,540 (העץ השני ממשאל עם ציון של 35,500). ושוב, אם נשאר ב- $\alpha = 0,000 = \alpha$ ובמקביל נציגם 2 עליים נוספים (אנו נcut בעץ השלישי משמאלי), נקבל ציון של 39,240 – שהוא גבוה יותר מ-35,500 ולכן זה לא טוב לנו.

עבור $\alpha = 15,000$ ניתן לראות שהציוון 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 הציוון של ה-root יהיה נמוך יותר מה-subtree הקודם.

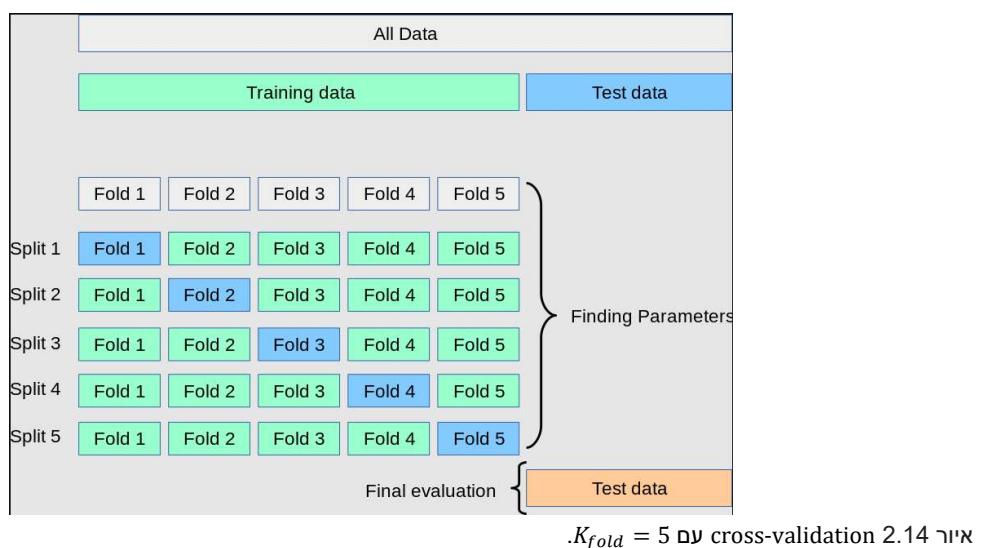
למעשה חזרנו על אותם צעדים עד שאין יותר מה לאציג. בסופו של תהליך ערכיהם שונים של α יתנו רצף של עצים, מעץ מלא ועד לעלה בודד. יוצא מכך של α קיימים subtree מתאימים $T_0 \subset T$ כך שה-tree score המתקבל יהיה קטן ככל האפשר.



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 subtrees השונים שהתקבלו (בסעיף הקודם), כיוון שהם יהיו שימושיים לחישוב ערך α אופטימלי:

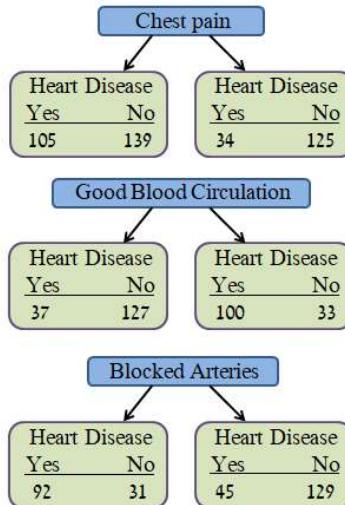
- o ערכי α חשובים, כיוון שלכל α עשוי להתקבל עץ אחר בעל ציון מינימלי. עד לשלב זה העץ נבנה ביחס **לכל הדadata** (בלי חלוקה ל-Test ו-Train), אך המטריה היא לבחון מהו ערך ה- α שייתן את התוצאה האופטימלית בעזרת העץ הגזום עבור DATA חדש.
- ט. כדי לבחור את ערך ה- α האופטימלי ניתן להשתמש ב-cross-validation. לשם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



- ב-cross-validation המודל מתאמן תחילה לפי 1 split (איור 2.14 לעיל) על התוצאות שבחלקים הירוקים, ובוחן את החיזויים על סט התוצאות הכלול.
 - התהילך זהה נעשה K פעמים, כאשר בכל איטרציה האימון נעשה את התוצאות שבחלקים הירוקים, ובוחינת החיזויים (וחישוב RSS) נעשה על התוצאות שבחלק הכלול.
 - . 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 ערכו α שהתקבלו בסעיף ז'.
 - . כתעת 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 דואק העץ שבו $= \alpha$.
 - . את התהילך של סעיפים יי' יש לבצע K פעמים – פעם אחת עבור כל split, כאשר בכל איטרציה האימון מtabצע על התוצאות שבחלקים הירוקים, ובוחינת החיזויים (וחישוב RSS) מtabצע על התוצאות שבחלק הכלול.
 - . בסופו של התהילך, כל איטרציה תניב subtrees בעלי RSS מסוים, וכן ערכו α שנקבעו כבר מראש בסעיף ז'. לאחר K האיטרציות יש לבדוק מיהו העץ עם RSS המינימלי מבין כל האיטרציות, ומהו ערך α של העץ זהה, וזה יהיה הערך הסופי של α .
 - . לבסוף, יש לחזור לעץ המקורי T_0 וה-subtrees שנבנו מה-data set 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 הוא בעל RSS הנמור ביותר, אך יתכן והוא יסבול מ-Overfitting.
 - כדי לפצות על כך, נוסף רכיב שנוןד להתחשב גם ביתר העצים שהנים בעלי RSS גבוהה יותר, ו- "מעניש" את העץ המקורי (T_0) עקב ריבוי העלים שבו.
 - באופן זהה התקבל ציון המאפשר להשוות בין העצים ולבחור את העץ הטוב ביותר. העץ הזה מהווה מען איזון בין הרצון ל RSS מינימלי לבין שונות נוכחית בין-h-Train ל-h-Test.
 - בכדי למצאו 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 רגסיה, רק שהמטרה היא שונה – במקומם לקבל תשובה כמותית כמו ברגסיה, עץ סיווג יtan תווית (label) לתצפית המבוקשת. כאמור לעיל, עץ רגסיה מספק חיזוי לתצפית מסוימת בהתאם לערך המוצע של תצפיות 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 חולים 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. וכך שהתשובה שהעץ נותן היא לא "ערך ממוצע" כמו שראינו בעץ רגסיה, אלא פשוט החלטה - "כן חוליה לב" או "לא חוליה לב". מלבד החיזוי של התווית, עץ סיווג מספק גם יחסים בין הקבוצות השונות בקרב תצפיות האימון שנoplים באותו אזור. נתבונן בדוגמא שתמחיש את העניין:



איור 2.15 השפעת פרמטרים שונים על הסיכוי לחולות במחלה לב.

באיור 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 אחידות בתוצאות. משתנים 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 המשנה שבראש העץ (Root node). כמובן, יש ליצור ממדד הבוחן ומשווה את רמת ה-*impurity* של כל משתנה. ישנו מספר מדדים, ונתמקד בשני מהם – Entropy ו-Gini index.

A. ממד Gini index :

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

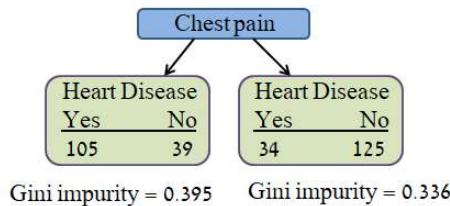
$$Gini = 1 - \sum_{j=1}^J p_j^2$$

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 התוצאות בכל עלה, באופן הבא:

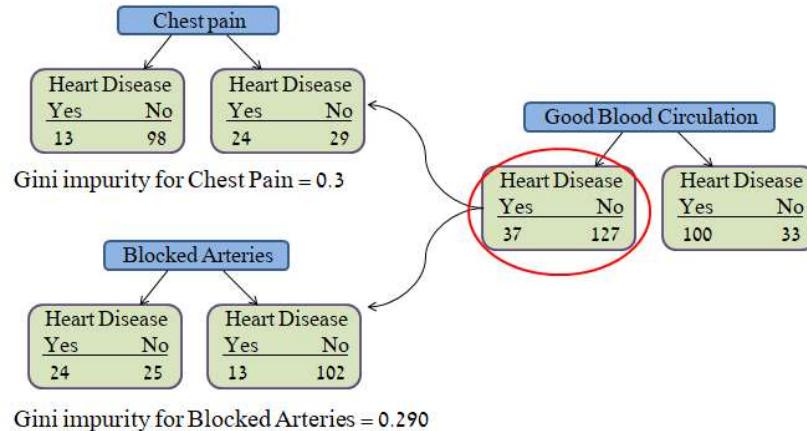
$$\text{Gini impurity for chest pain} = \left(\frac{144}{144 + 159} \right) \times 0.395 + \left(\frac{159}{144 + 159} \right) \times 0.336 = 0.364$$

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

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

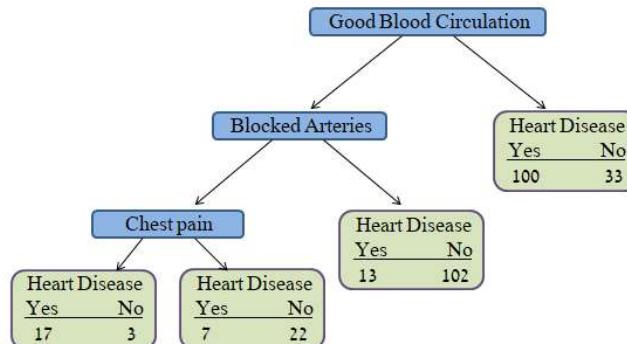
$$\text{Gini impurity for blocked arteries} = 0.381$$

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:



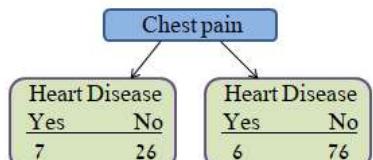
איור 2.17 חישוב מדד Gini עבור שאר הפרמטרים לאחר קביעת ה-root.

כפי שניתן לראות, למשתנה Blocked arteries יש ציון גINI 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 pain, באופן הבא (המספרים כמפורט בתכיפות האמיתיות):



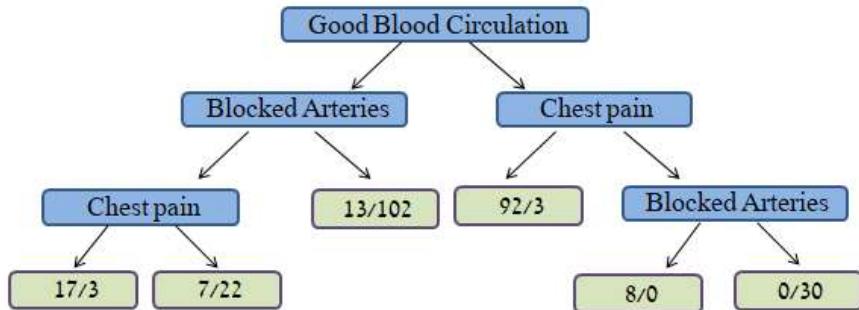
$$\text{Gini impurity for Chest Pain} = 0.29$$

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

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

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

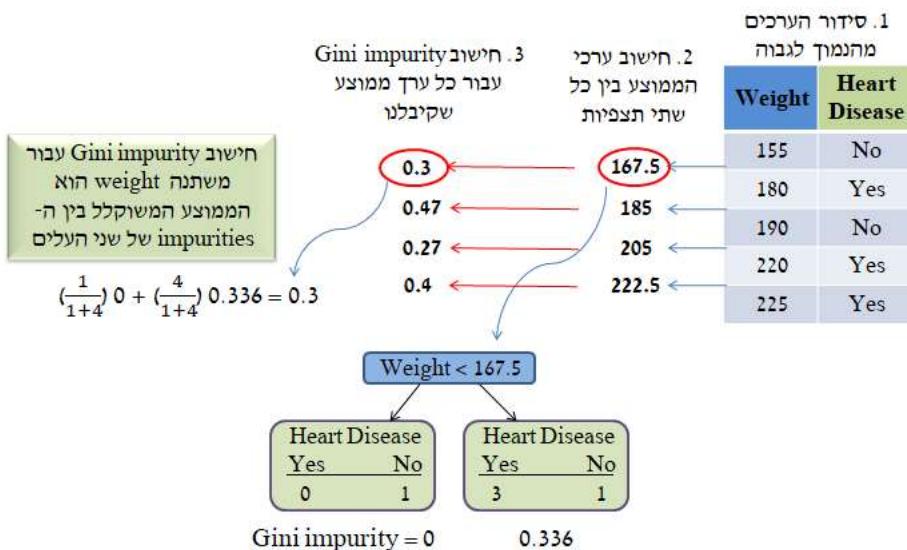
ע"ז טיפוסי לאחר סיום התהילה נראית כך:



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

מדד Gini שהתקבל הינו 0.29, בעוד שבלי הפייזל המדד של העלה היה 0.2.התהיליך שתוואר מטאדים למצבים בהם הפרמטרים מתפצלים באופן בין-ארט, כלומר הפייזל של כל פרמטר נקבע על ידי שאלה שעליה יש תשובה של כן או לא. במקרה בהם ישנו משטנים רציפים, הפייזל דורש כמה שלבים מקדימים:

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

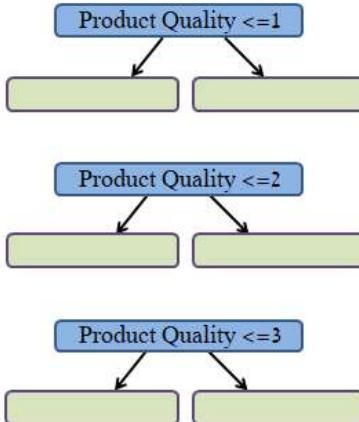


איור 2.21 פיצול פרמטרים רציפים לפי מדדי Gini.

אנחנו מקבלים את ה-Gini הנמוך ביותר מתי שאחננו קובעים threshold של weight<205. אז זהו ה-cutoff וערך ה-Gini שנשתמש בהם כשאחננו משווים את המשטנה weight ליתר המשטנים. עד עכשו דיברנו על איך מחלקים משתנה רציף ואיך מחלקים משתנה בינארי (שאלות כן/לא).Cut-but בואפין דומה נבנה גם את הענף ימני של העץ:

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

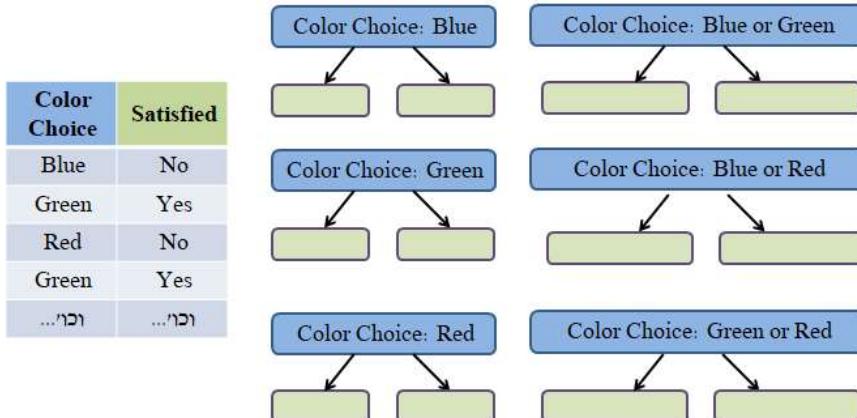
Product Quality	Satisfied
1	No
1	Yes
3	No
1	Yes
... וכך...	... וכך...



שימוש בפ: אנחנו לא צריכים לחשב את ה- impurity score ל- prod. quality <= 4- השוואת colum בעצם את colum

איור 2.22 פיזול 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 פיזול פרמטרים קטגוריאליים לפי מדד Gini.

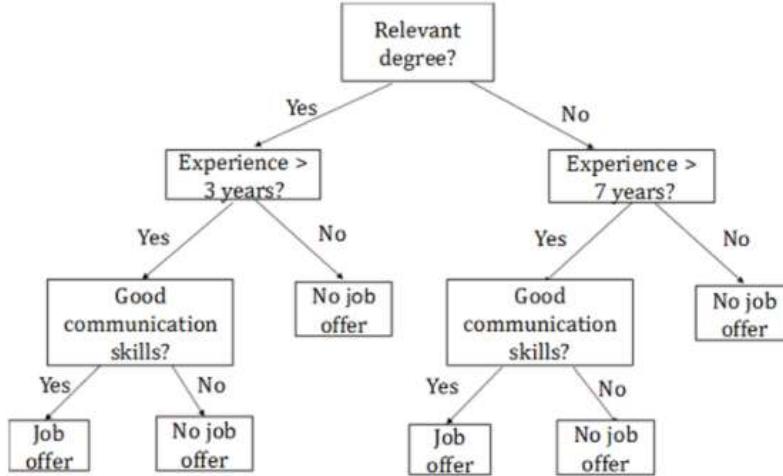
ב. מדד Entropy & Information Gain

מדד נוסף לפיזול צמתי העץ מtabased על האנתרופיה של העלים, בעזרתה ניתן לבחון את ה-information gain המקיים מכל פיזול. אנטרופיה באה לממדוד את השגיאה של התפלגות המשנה הנבחן מול משתנה המטרה. נניח וישנן 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, הפיזול האופטימלי נבחר על ידי המשנה בעל מדד האנתרופיה הנמוך ביותר. אם כל התוצאות בעלי מסוים משויות לאותו class, אז מדד האנתרופיה יהיה 0. מайдך, כאשר בעלי מסוים יש התפלגות שווה בין 2-class-ים של המשנה המוסבר, מדד האנתרופיה יהיה 1 (זהה הערך המקיים שמדד אנטרופיה יכול לקבל).

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



איור 2.23 עץ סיווג עבור קבלת מועדן לתקפ'יד מסוים.

הדוגמא מתארת את אחד המאפיינים העיקריים של עצי החלטה – ההחלטה מת金陵ת על ידי הסתכילות היררכית על הפרמטרים השונים, כאשר בכל פעם מתחקדים 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 להתקבל לעבודה הוא 20%, והסתוכוי לא להתקבל 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 בונוסף 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 ומუלדים 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* שמת金陵ת מהידיעה האם למועדן מסוים יש תואר רלוונטי או לא. כאמור, כמה הידיעה שלמועדן מסוים יש תואר רלוונטי מפחיתה מוחסן ה odds שלו לקבל את המשרה. אם אי ה odds נמדדת באמצעות מדד ה-*Entropy*, Entropy, הרי שהרווח מה מידע הוא:

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

הן עבור מועמדים בעלי תואר והן עבור כלה ללא תואר, המשטנה שמקנס את הרוח מהמידע הצפוי (ירידה באנטרופיה הצפוי) הוא מספר שנות הניסיון. כאשר למועדן יש תואר רלוונטי, הסוף עבור "שנות ניסיון" הממקנס את הרוח מהמידע הצפוי הוא 3 שנים. עבור הענף התואם למועדן שאין לו תואר רלוונטי, הסוף של שנות ניסיון הממקנס את הרוח מהמידע הצפוי הוא 7 שנים. לפיכך, שני הענפים הבאים הם: "ניסיון < 7" ו- "ניסיון ≥ 7". באותו האופן 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 הכלולות של התוצאות. במקורה 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 לשינוע או לחיזוי ערכו של משתנה, כאשר המאפיינים מסודרים לפי סדר החשיבות. לצורך סיווג, קיימים שני מדרדים אלטרנטיביים לאז וודאות: מדד אנטרופיה (Entropy) ומדד 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 אשר יתרחש כאשר יתקבל מידע אודות המאפיין.

במקרה של חלוקת **משתנה קטgorיאלי**, המידע המתkeletal הינו על פי רוב אוזחות קטגוריה (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- **אוזחות information gain** הצפוי.

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

במקרים בהם עż ההחלטה משמש לשינוע, 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 הדיקוק של העץ צריכה כמו תמיד בלמידת מכונה לבוא מtower סט הבדיקה (Test Set) ולא רק מtower סט האימון.

אחרית דבר:

מדד RSS ומדד RSS misclassification rate שהוצגו בפרק זה הינם רק חלק מהמדרדים המקובלים. לכל מדד 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 לתוצאות חריגות מאשר רגסיה.

חרונות:

- 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 תכופות דורשים יותר זמן הרצה לאימון המודל, ומשכך מדובר באלגוריתם "יקר" במשאבים.
- האלגוריתם של עż ההחלטה אינו מספק ליישום רגסיה ולניבוי ערכים רציפים.
- עż ההחלטה נוטה לעתים תכופות לננות 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 אינם נתונים. אלגוריתם זה מתאים לביעות של חלוקה לאשכולות (Clustering), ובנוסף יכול לשמש בשלב הצגת ניקוי הנתונים (EDA). עבור כל נקודה במדגם, המודל ממציע את סכום ריבוע המרחקים (WCSS) מכל מרכז אשכול (סנטרואיד - centroid), ולאחריה 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 היגיומטריה, 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 מורכבים ממשולש. 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. מכך זה נקרא WCSS, והוא ממד משמעותי ביחס לקרוב אלגוריתמים שմבצעים חלוקה לאשכולות, K-means. הסיבה להזקה במשווה היא שאנו רוצים להגבר את ההשפעה של המרחק, מעין "עונש" לתוצאות רחוקות מהמרכז.

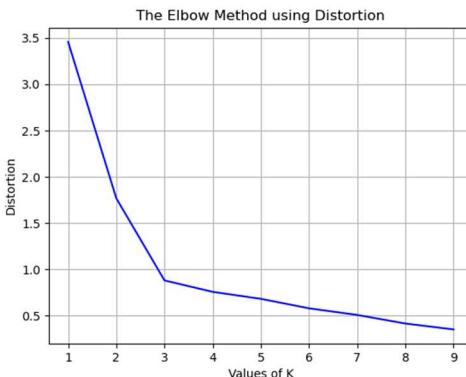
מדד WCSS הוא אחד הדריכים המקבילים ביוטר להערכת תוצאות החלוקה לאשכולות-ב-K. היתרונו של מדד זה הוא האפשרות לראות אם מידת הצלחה של המודל, כאמור לקבל מספר ממש שמאמת את הצלחת המודל. מנגד, WCSS הוא מסוף ללא תחום מסוים והוא דרוש פרשנות, כיון שהערך והמשמעות שלו משתנים ממודל למודל. ערך מסוים יכול להיחשב תוצאה טובאה במקורה מסוים, ובמקרה אחר זאת עשויה להיחשב תוצאה רעה מאוד. ניתן להשוות WCSS בין מודלים אך ורק כאשר יש להם את אותו מספר אשכולות ואיתו 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 מדד 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 שבhartoon סכום המרחקים של התוצאות מהסentrואידים יקטן או לא ישתנה. כיון 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 תשתנה, וערך ה-WCSS לא ישתנה.

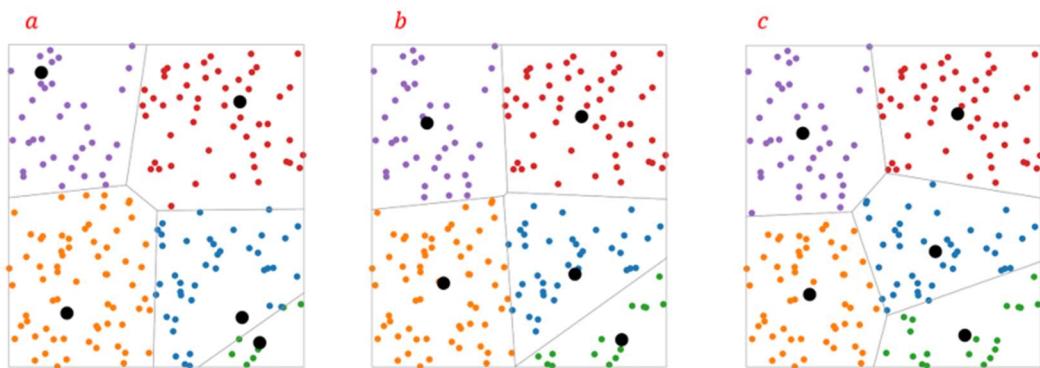
לכן מצד אחד, נרצה לבחור K גדול שימזר את ה-WCSS; מצד שני, הסיבה שהשתמשנו ב-K-means מלכתחילה היא בכך לפשט את הנתונים למספר סביר של אשכולות, כזה שיאפשר לנו לעורך אנליזה נוחה. שיטת המפרק (Elbow method) היא טכניקה שימושת לפתור סוגיה זו. הרעיון הוא לבחור את ה- K הקטן ביותר שמננו השיפור במדד ה-WCSS הוא מטען במידה סבירה. שיטה זו היא היוריסטיבית ואינו דרך חד משמעית לקבוע שה- K הנבחר הוא האופטימלי. בדרך זו ניתן לשכנע מודיעע 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 ירידה 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 בחרה של $K = 3$ הינה בירה 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. **אתחול:** המודל מציב את הסנטראידים באופן רנדומלי.
- ב. **шиיר:** כל תצפית משוכנת לסנטראיד הקרוב אליה ביותר.
- ג. **עדכן:** הסנטראיד מוחזק שכ- WCSS של המודול יمواזר.
- ד. חוזרת על שלבים ב, ג עד אשר הסנטראידים לא זרים לאחר העדכון, כלומר יש התכנסות.



איור 2.7 (a) **אתחול** 6 סנטראידים באופן רנדומלי. (b) **שייר** כל נקודה לסנטראיד הקרוב ביותר אליו. (c) **עדכן** הסנטראידים לפי מודד-WCSS. (d) **חזרה 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 להוות אתגר במרבית המקרים. בנוסף, האלגוריתם רגיש מאוד לערכים קיצוניים (Outliers). אופן הפעולה של האלגוריתם מאפשר לו ליצור אשכולות רק בצורה של ספירות, והדבר אינו אופטימלי בחלוקת מן המקרים.

בעיה נוספת להתייעזר בבחירה המוקדם הריאשוני של הסנטראידים – 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 למקום סנטראיד אחד באופן רנדומלי. לכל תצפית, האלגוריתם מחשב את המרחק בין התצפית לסנטראיד הקרוב אליו ביותר. לאחר מכן, תצפית רנדומלית נבחרת להיות הסנטראיד החדש. התצפית נבחרת בהתאם להסתפלות משקלת של המרחקים, כך שכל שתצפית יותר רוחקה – כך גובר הסיכוי שהיא תבחר. שני השלבים האחרונים נמשכים עד שנבחרו K סנטראידים. כאשר כל הסנטראידים מוקמו, מבצעים K-means עם דילוג על שלב האתחול (השלב בו ממקמים את הסנטראידים). +++ מוביל להתקנסות מהירה יותר, ומוריד את הסיכוי להתקנס לאופטימום מקומי.

2.2.2 Mixture Models

אלגוריתם K-means מחלק א-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 מניה 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 לשיך אותה לאחד האשכולות, אז צריך למשה למצוא את האשכול i שעבורו הביטוי $p(x|y = i)$ הוא הכי גדול. לפי חוק ביסוס מתקיים:

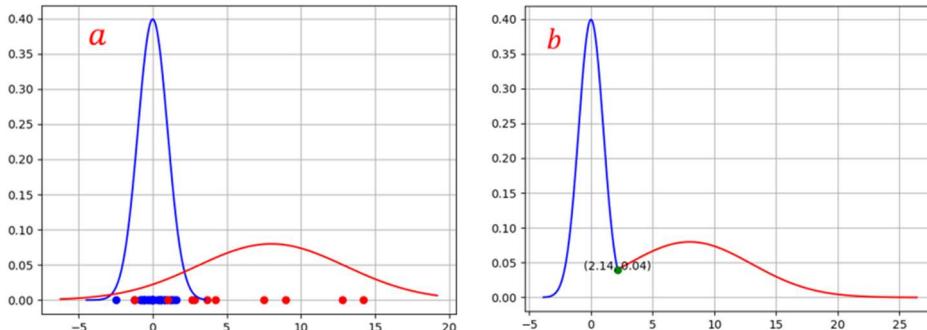
$$p(y = i|x) = \frac{p(y = i) \cdot p(x|y = i)}{p(x)}$$

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:

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

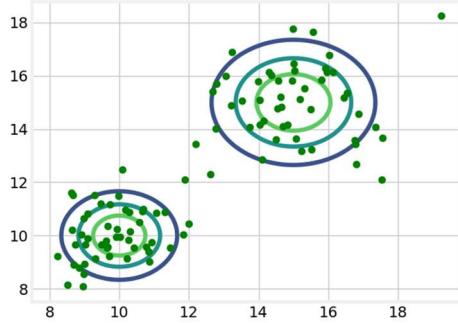


איור 2.8 (a) תערובת של שני גאוסיאנים במדוד אחד: בשלב ראשון מחלקים את הנקודות לשני אשכולות ומתייחסים לכל אשכול התפלגות מסוימת. במקרה זה אשכול אחד (מוסמן בכחול) הותאם להתפלגות $\mathcal{N}(0, 1)$, ואשכול אחד (מוסמן באדום) הותאם להתפלגות $\mathcal{N}(8.5, 1)$. (b) נקודה חדשה x תסוג לאשכול הכחול אם $2.14 < x < 2.14 + 2.14 \cdot 0.04 > 2.14 + 0.04 = 2.18$. במקרה דומה, הנקודה x תסוג לאשכול האדום אם $x > 2.14 + 2.14 \cdot 0.04 = 2.18$.

כאמור, כדי לשיך נקודה חדשה x לאחד מהאשכולות, יש לבדוק את ערך ההתפלגות בנקודה החדש. ההתפלגות שבעזרה ההסתברות (x) היא הגדולה ביותר, היא זאת שאליה תהיה משוכנת הנקודה. ההתפלגיות יכולות להיות בחד ממד, אך הן יכולות להיות גם בממד יותר גבוה. למשל אם מסתכלים על מישור, ניתן להתאים לכל אשכול ההתפלגות נורמלית דו-ממדית. במקרה ה- n -ממדי, 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 תערובת של שני גאוסיאנים דו-ממד: אשכול אחד מתאים לגאוסיאן עם וקטור תוחולות $\mu_1 = [10, 10]$ ומטריצת covariance $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, וה אשכול השני מתאים לגאוסיאן עם וקטור תוחולות $\mu_2 = [15, 15]$ ומטריצת covariance $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$.

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

2.2.3 Expectation–maximization (EM)

אלגוריתם מקסום 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 אשכול מסוים המתפלג נורמלית עם תוחלת ושותות (μ, σ^2) , ומשויכות אליו n נקודות. כדי לחשב את ההתפלגות של אשכול 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 אשכולות 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 הפשטוט:

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

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 - 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 הפרמטר θ נחשב את האשכול המתאים לכל נקודה, כלומר כל נקודה x_i תוויתם 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 התהליך באופן איטרטיבי עד להתכנסות.

עבור Mixture of Gaussians נוכל לחשב באופן מפורש את הביטויים:

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

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

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

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

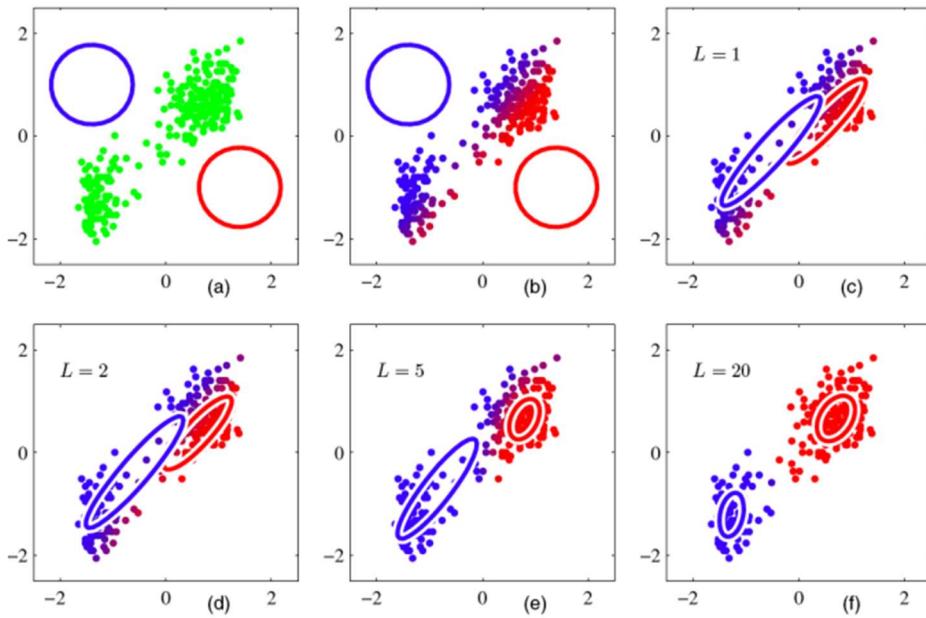
וכעת ניתן לגזר ולמצוא אופטימום:

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

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

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

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 אקריאי של ההתפלגות, ובכל איטרציה יש שיפור כך שההתפלגות מייצגתו בצורה יותר טובה את הדadataה המקורית.

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 בדיקת $Q(\theta, \theta_0)$. האיבר השני לפני הגדרה הוא האנטרופיה של ההתפלגות $p(y|x; \theta_0)$:

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

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 גיבס](#) 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 מצאת שני agglomerative clustering – בשלב הראשוני מגדרים כל נקודה כ האשכול, ואז 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: $\min\{d(a, b) : a \in A, b \in B\}$.

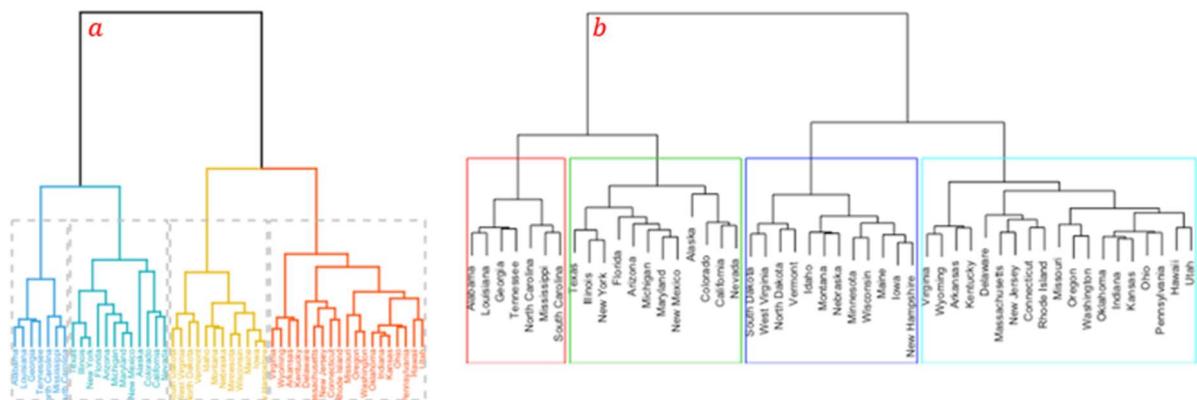
Unweighted average linkage clustering (UPGMA): $\frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$.

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") – היא שתחילה כל נקודה הינה אשכול עצמאי ובכל צעד של האלגוריתם מסpter האשכולות קטן באחד. במקרים אחרים, האלגוריתם בונה את האשכולות ממצב שבו אין למעשה חלוקה לאשכולות למשך זמן קצר. שיטה זו מכונה "bottom-top" ("top-down") – היא שתחילה כל אשכול ייחיד ובכל צעד של האלגוריתם מתוויף עוד אשכול.

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 n^2 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 "bottom-top" ("top-down") – היא שתחילה כל אשכול ייחיד ובכל צעד של האלגוריתם מתוויף 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) – התחילה מאשכול יחיד ופיצול עד שמגיעים למספר האשכולות המקורי (במקרה זה 4). (b) – בהתחילה כל נקודה הינה אשכול, ובכל צעד מחברים שני אשכולות עד שמגיעים למספר האשכולות המקורי.

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 מקובוצה או לחילופין חריגה ויצאת דופן.

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

לאחר חישוב השיעיות $LOF(x_i)$ 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 ההופci 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.

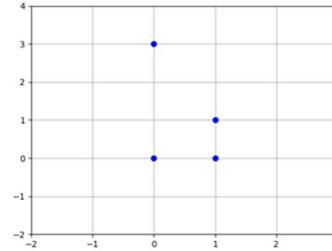
בשלב האחרון בוחנים עבור כל נקודה x את היחס בין ה-LRD שלה וה-LRD של $N_k(x)$. היחס הזה הוא ה-LOF, והוא מחושב באופן הבא:

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

הביטוי הראשון במכפלה הוא ממוצע ה-LRD של k השכנים של נקודה x , ולאחר חישוב הממוצע מחלקים אותו ב-LRD של הנקודה x עצמה. אם הערכים קרובים, אז ה-LOF יהיה שווה בקירוב ל-1, ואם הנקודה 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 האותיב 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:

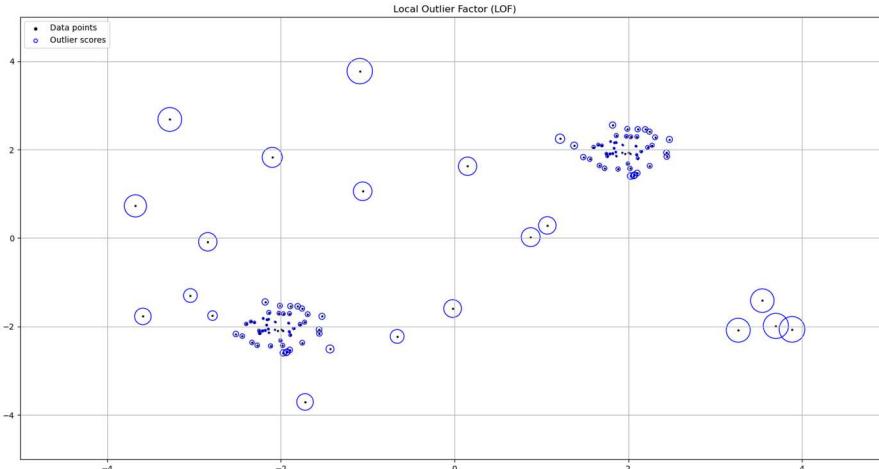
$$LOF_2(A) = \frac{LRD_2(B) + LRD_2(C)}{k} \times \frac{1}{LRD_2(A)} = 0.87$$

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

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

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

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 מרכזים בשימוש באlgorigthms זה – ראשית יש לבחור k מתאים, 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 שהייה 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 אך עבור ערכי LOF גדולים מ-1 אין כלל חד משמעי עבור איזה ערך הנקודה היא outlier או עבור איזה ערך היא לא. כדי להתמודד עם אתגרים אלו הוצעו הרחבות לשיטה המקורית, כמו למשל שימוש סטטיסטיות שונות המורידות את התלות בבחירת הערך k (LoOP – Local Outlier Probability), או שיטות סטטיסטיות העוזרות לחתתפרשנות לערכים המתקבלים (Interpreting and Unifying Outlier Scores).

2.3 Dimensionally Reduction

הורדת ממד (Dimensionality Reduction) הינה טרנספורמציה המפה דатаה לממד נמוך יותר מהמדד המקורי, כאשר נרצה שההורדת הממד לא תנסה באופן מהותי את מאפייני הדטה המקורי. הורדת הממד של דטה נתון נדרשת משתי סיבות עיקריות – אחת טכנית והשנייה מהותית:

א. ביצוע חישובים ופעולות על מערכת ממדים הינה בעלת סיבוכיות גבוהה, ולעתים אף בלתי ניתנת לביצוע.

ב. הורדת הממד של הדטה קשורה לניסיון להבין מהם המשתנים העיקריים ומהם המשתנים המשניים, הפחות חשובים להבנת הדטה (אלו שפחות מאפיינים דוגמא נתונה ביחס לדוגמאות אחרות). לעיתים התחשבות במסתנים 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 בויזואлизציה של המידע, כך שניתן להציגו על ידי 2 או 3 ממדים עיקריים, בעזרה גרף דו-ממדי או תלת-ממדי בהתקאה.

דוגמה למערכת מרובת ממדים יכולה להיות מדידת רמות חלבונים (פרוטאין) של גנים (genes) המבוטאים בתא ח', כאשר כל ממד, או מאפיין (פייצ'), מטאים לאחר. באופן כללי, יתכן ונמדדים בכל ניסוי מאות תאים, כאשר לכל תא נמדדות רמות ביוטי של מאות או אלפי גנים. כמות עצומה זו של מידע בממד גבוה (אלפי תאים ואלפי גנים בכל תא) מאייגרת את המחקר – הן מבחינות זיהוי המאפיינים, או רמות הגנים המבוטאים, הרלוונטיים והמשמעותיים ביותר, והן מבחינות ניסיון למדל את הדטה בצורה כמה שיוטר פשוטה. במחקר משנת 2007 נלקחו 105 דגימות של תא סיוטן שד, כאשר לכל דגימה (או דוגמא) נמדדו רמות התבניות של 27,648 גנים שונים. כמובן שלgentה את המידע בצורה הגלומית זו משימה בלתי אפשרית, ויש הכרח לבצע עליו מניפולציה כלשהיא כדי שהיא אפשר לעבוד איתו.

ישן שיטות רבות לביצוע הורדת ממד לדעתה, כאשר ניתן לסייע לשתי קטגוריות עיקריות: בחירת מאפיינים (feature selection), והטלה מאפיינים (feature projection). השיטות השתייכות לקטגוריה הראשונה מנוסת לבחור את המאפיינים (המשתנים) המתאים באופן מספק את המידע הנוכחי. שיטות מוקטגוריה השנייה, המתווארות בפרק זה, נוקטות בגישה של הטלה, טרנספורמציה, של המאפיינים המקוריים לייצוג על ידי סט של מאפיינים חדשים במרחב אחר (ולרוב פשוט יותר). חשוב להציג בשיטות מוקטגוריה הראשונה, מאפיינים פחות רלוונטיים מושפעים. בנויג לבך, בשיטות שנדרן בהן בפרק זה, המבוססות על הטלת המאפיינים, כל מאפיין חדש הינו טרנספורמציה של כל האחרים, ולא רק של חלקם. כך, המאפיינים החדשניים מقلילים, או לוקחים בחשבון, כל אחד מהמאפיינים הנמדדים המקוריים, ללא השמטה.

ניתן לבצע הטלת מאפיינים באמצעות טרנספורמציות ליניאריות. בפרק זה עוסק בטרנספורמציה ליניארית אחת, הנקראת ניתוח גורמים ראשיים (Principle Component Analysis) ובסתי טרנספורמציות לא-ליניאריות (UMAP, t-SNE-t). נציין כי קיימות עוד טרנספורמציות, ליניאריות ולא-ליניאריות, המשמשות להורדת ממד של דата שאין יזכרו כאן.

2.3.1 Principal Components Analysis (PCA)

כפי שהוזכר לעיל, ניתוח גורמים ראשיים מבוסס על טרנספורמציה ליניארית של המאפיינים המקוריים. המטרה של אלגוריתם זה היא לבנות ייצוג חדש ובעל ממד נמוך יותר מאשר הממד המקורי של הדעתה, מתוך מטרה לשמר כמה שיותר את השונות של המאפיינים (features) של הדעתה המקורי. מדוע השונות כה משמעותית? ניקח למשל מאפיין המופיע בכל הדוגמאות בדעתה הנוכחי, ובכל מאפיין זה הוא בעל אותו ערך. מאפיין זה הוא בעל שונות אפס, ולמעשה הוא לא מכיל שום מידע על הדעתה ונרצה להיפטר ממנו. בדומה לכך, אם יש שני מאפיינים בעלי תלות ליניארית – אין טעם לשמר את שניהם כיוון שידיעת האחד מאפשרת לדעת גם את השני. במקורה שיצוג שני המאפיינים הוא 1, והשארת המאפיין השני לא תתרום "לשונות הכלולות" של ייצוג הדעתה. נראה בהמשך שיצוג שנבנה באמצעות PCA לא יכול פיצ'רים כאלה. עוד נעיר כי האלגוריתם דואג לכך שהמקדמים דואג לכל צירוף לינארי היו וקטורים בעלי אורקל של 1, בכך לא לנפח באופן מלאכותי את השונות של המאפיינים החדשניים (לאחר הורדת ממד).

לאחר הקדמה זו, נפרט כיצד מחושבים המאפיינים החדשניים, הנקראים למעשה "הגורמים העיקריים". הגורם הראשי הראשון (first principal component, PCA_1) הינו צירוף הליניארי של המאפיינים המקוריים בעל השונות הגדולה ביותר. הגורם הראשי השני (second principle component, PCA_2) הוא גם צירוף ליניארי של המאפיינים הנוכחיים, השונות שלו היא השניה הגדולה ביותר, ובנוסף דורשים ש- PCA_2 יהיה אורתוגונלי ל- PCA_1 : $PCA_2 \perp PCA_1$. הגורם השלישי, הוא צירוף ליניארי בעל השונות השלישית הגדולה ביותר, ומאונך לשני הגורמים המקוריים $PCA_2 \perp PCA_3$ וכן $PCA_3 \perp PCA_1$, וכן הלאה כך שהגורם הראשי מסדר i , הוא בעל השונות ה- i -ית הגדולה ביותר תחת אילוץ של גורמים מאונכים – $j < i \Rightarrow PCA_j \perp PCA_i$. הורדת הממד מתבצעת על ידי לקיחת מספר גורמים ראשיים המקוריים, והזנתה הקטנים ביותר.

לאחר שאפינו את הגורמים הראשיים בהם אנו מעוניינים, עולה השאלה כיצד ניתן לבצע טרנספורמציה ליניארית שבוצرتה ניתן למצוא את הגורמים הראשיים הללו. נניח שבידינו דעתה $\vec{X} \in \mathbb{R}^{M \times N}$, הכולרת נתונות M דוגמאות שונות, שכל אחת מהן היא עלת N מאפיינים [למשל, עבור הדוגמא של תא סרטן השד, נתון מידע-מ- $M = 105$ תאים שונים, כאשר עבור כל תא נמדד רמות ביוטי של $N = 27,648$ גנים שונים]. מטריצה זו לעתים נקראת ה-*design matrix* של הדעתה, ונסמן אותה באופן הבא:

$$\hat{\vec{X}} = \begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_M \end{bmatrix} = [\vec{X}^1, \dots, \vec{X}^M] \in \mathbb{R}^{M \times N}$$

כאשר כל וקטור שורה, $\{1, \dots, M\} \in \vec{X}_m$, הינו נתוני המדידות של המאפיינים השונים בדוגמא מספר m , ובהתאם, וקטור عمودה $\{N, \dots, 1\} \in \vec{X}^n$ (שימו לב לשינוי סימון, אינדקס עליון עבור וקטורי عمودה), הינו נתוני המדידות של מאפיין מסויים על כל הדוגמאות. נניח שסכום המדידות עבור כל מאפיין הוא אפס, זאת אומרת שילם מאפיין n -י מתקיים:

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

מכיוון שכל עמודה של המטריצה מסמלת ערכים של מאפיין מסוים במדידות שונות, סכום כל עמודה במטריצה \hat{X} הוא אפס.icut, נרצה לבצע הטלה (טראנספורמציה) לינארית, 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 $K \times N \in \mathbb{R}^{N \times K} \in \hat{W}$, כך ש- \vec{T}_m זו מביאה לכך שלאחר הטרנספורמציה נשארים רק K מאפיינים. כיוון שאנו מעוניינים בהורדת הממד, קרי הורדת מספר המאפיינים, נדרש $N \leq K$.

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

(1) כל עמודה של מטריצת המשקלים הינה בעל נורמה השווה ל-1, כלומר:

$$\|\hat{W}^k\|^2 = \sum_{m=1}^M (W_{m,k})^2 = 1$$

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

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

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

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

זאת אומרת:

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

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

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

כאשר מטריצה $\hat{S} = (\hat{X})^T \cdot \hat{X} \in \mathbb{R}^{(N \times N) \times (N \times N)}$ הינה מטריצת השונות המשותפת (covariance), המוגדרת על ידי \hat{X} מטריצה זו, מסדר $N \times N$, מגדרה את השונות המשותפת בין צמד מאפיינים, כאשר $S_{v_1, v_2} = \sum_{m=1}^M X_{v_1, m} X_{v_2, m}$ ניתן לשימם לב כי מטריצה זו סימטרית וממשית (ולכן הרミיטית).

לפי משפט המינימום-מקסימום (קורנט-פישר-ויל, מובא כנספח לפרק): עבור \hat{S} מטריצה הרמייטית בעלת ערכים עצמיים $\lambda_K \geq \dots \geq \lambda_1$, מתקיים:

$$\lambda_1 = \underset{\|\hat{W}\|=1}{\operatorname{max}}((\hat{W}^1)^T \cdot \hat{S} \cdot \hat{W}^1)$$

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

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

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

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

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

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

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

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

ניתן גם, באופן פשוט יותר, להשתמש בשיטת פירוק לערכים סינגולריים (SVD), כאשר נמצא את הפירוק המתאים למטריצת השונות המשותפת:

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

כאשר $\hat{\Lambda}$ הינה מטריצה אלכסונית, $-\lambda_i = \Lambda_{ii}$ הינם הערכים העצמיים של \hat{S} המסודרים לפי גודלם מהגדול לקטן - $\lambda_M \geq \dots \geq \lambda_2 \geq \lambda_1$, ומטריצה \hat{W} מורכבת מעמודה שהינם הווקטורים העצמיים המתאים לערכים העצמיים. הווקטורים העצמיים בהגדرتם הינם אורתוגונליים זה לזה, וכיון ש- $1 = \|\hat{W}\|$ לכל k , הם בעצם אורתונורמליים.

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

- א. "מרכז" את הנתונים כך שהממוצע עבור כל מאפיין הוא אפס: $\hat{X}^m = \hat{X}^m - \text{mean}_n(\hat{X}^m)$.
- ב. מצא את מטריצת השונות המשותפת $\hat{X}^T \hat{X} = \hat{S}$.
- ג. מצא את $\hat{W}^T \hat{W} \cdot \hat{\Lambda} \cdot \hat{W} = \hat{S}$ – ה-SVD של \hat{S} .
- ד. חשב $\hat{W} \cdot \hat{X} = \hat{T}$.

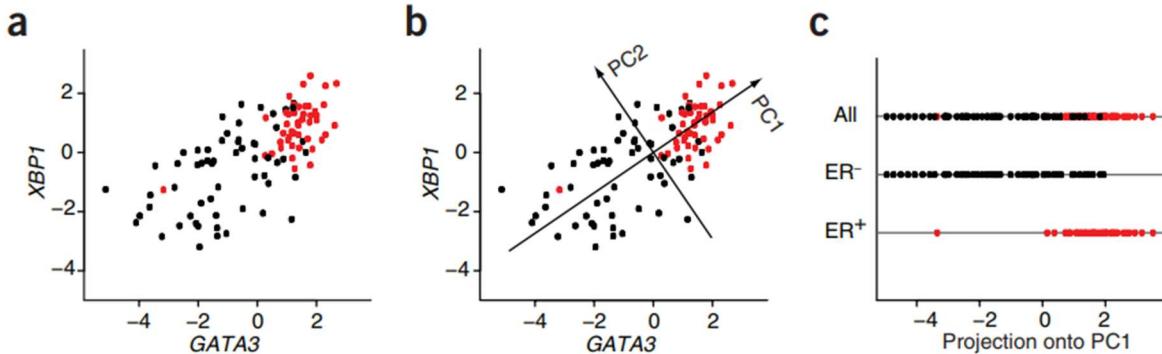
ה. הגורמים הראשיים נתונים על ידי וקטורי העמודה $\vec{W}^k \equiv PCA_k$, וההטלה של מדידה m למערכת המאפיינים החדשה נתונה על ידי $\vec{T}^m = \hat{X}^m \hat{W}^m$.

נעין שלשิตת הניתוח של גורמים ראשיים יש מספר מגבלות. ראשית, היא נוטנת "משקל יתר" על מאפיינים שהשונות בהם גדולה, ללא קשר לחישובתם, או ליחידות שבahn 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 לנארית בין שני משתנים, אולי יתרן במערכות מסוימות שדוקא הקורלציה הלא-لينארית היא החשובה יותר. כמו כן, לעיתים "מרכז" המידע גורם לתוצאות לאבד ממשמעוון.

כדי להתגבר על המגבילות בשיטת PCA שהציגו לעיל פותחו שיטות נוספות או משלימות. לדוגמה, ניתן למצער את השפעת יחידות המדידה על המאפיינים על ידי הפיכתם לחסרי יחידות. בנוסף, יש שיטות הולוקחות בחישוב קורלציות לא-لينאריות, לדוגמה שיטת PCA kernel, או שיטות להתמודדות עם בעיית המרכז על ידי דרישת משתנים חיוביים (NMF).

לצורך המבנה של תהליך חישוב הגורמים העיקריים ניתן שתי דוגמאות. ראשית נחזר לדוגמא שהזכרנו בתחילת פרק זה – מחקר שפורסם בשנת 2007 ובו נלקחו 105 דגימות של תאי סרטן שד, כאשר לכל דגימה נמדד רמות התבטאות של 27,648 גנים שונים. לשם הדגמה, נשתמש בניתוח שפורסם כמנה לאחר מacen (ב-2008) על ידי אחד מעורכי המחבר המקורי. שם, החוקר מציג רמות של שני חלבוניים; האחד בשם GATA3, והשני בשם XBP1, כאשר

דגימות תא הסרטן מסווגות לפי סוג קולטני האסטログן שלהם (+ או -). כתע, על ידי "סיבוב" מערכת הצירים – בעזרת טרנספורמציה לינארית PCA כפי שהוסבר לעיל – נמצא כי ניתן לסוווג, ללא איבוד מידע רב, את מצב קולטני האסטログן בתאי סרטן השד על ידי הגורם הראשי הראשון, PCA₁, כפי שניתן לראות באIOR. יש לשים לב שהגורם הראשי הראשון, PCA₁, מכיל מידע משני החלבונים.



איור 2.13 (a) רמות ביוטי של שני חלבונים GATA3 (ציר ה-X) ו-XBP1 (ציר ה-Y). קולטני אסטログן חיוביים או שליליים מסומנים באדום ושחור בהתאם. (b) מציאת הגורמים הראשיים, וסיבוב מערכת הצירים. בהתאם לתיאוריה, ניתן להבחין כי השונות של המידע על גבי הציר החדש PCA₁ הינה מקסימלית. (c) הצגת תוצאות המדידה כפונקציה של PCA₁ בלבד. בגרף זה ניתן לראות בבירור כיצד הורדת הממד מסיעת למציאו הבדיקה פשוטה (בממד אחד) בין קולטני האסטログן.

נבייא בסוף דוגמא חישובית מפורטת. נניח ונთנו המערכת הדו-ממדי הבאה:

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

מערכת הנתונים מכיל 5 דוגמאות, וכל דוגמא נמדדו שני מאפיינים ($2 = M$). שורות המטריצה מציגות את המדידות השונות, והעמודות מייצגות את מאפייניהם.

מערכת זה כבר ממורכז, כלומר המאפיין הראשון מקיים:

$$\text{mean}_1(X^m) = \sum_{m=1}^5 X_{m1} = -0.5 - 0.4 + 0.1 + 0.3 + 0.5 = 0$$

ובבור המאפיין השני:

$$\text{mean}_2(X^m) = \sum_{m=1}^5 X_{m2} = -0.4 - 0.1 + 0 + 0.3 + 0.2 = 0$$

נחשב את מטריצת השונות המשותפת:

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

על מנת ללקסן מטריצת זו, נמצא את הערכים העצמיים שלה המהווים שורשים של המשוואה האופיינית הבאה:

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

כאשר לפולינום אופיני זה שני שורשים: $\lambda_1 \approx 1.02$, $\lambda_2 \approx 0.04$ ($\lambda_1 > \lambda_2$). נמצא כעת 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:

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

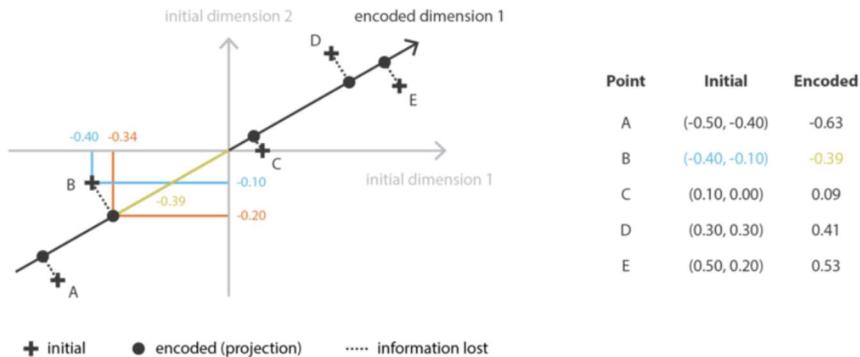
הטלה המדידות למערכת המאפיינים החדשה נתונה על ידי:

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

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 מידע של דאטא דו-ממדי לממד אחד.

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

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

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

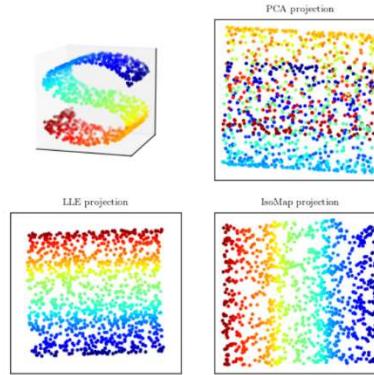
הערך העצמי המקסימלי מקיים:

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

כאשר \hat{W} , הינו הערך העצמי המתאים ל- $\hat{\lambda}_1$ – ערך העצמי המקסימלי של \hat{S} .

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

אלגוריתם הורדת הממד PCA פועל באופן לינארי, מה שמקל על תהליכי החישוב שלו, אך מגביל את יכולות ההכללה שלו. אלגוריתם אחר, לא לינארי, נקראת t-SNE, והוא מנסה לקחת את הדטה במרחב גבוה ובאמצעות שימוש בכלים סטטיסטיים למפות אותו למערכת דו-ממדית או תלת-ממדית.



איור 2.15 הדגמה של מגבלת שיטת PCA ויתרונות השיטות הלא-LINARIOT להורדת ממד. באור השמאלי העלון מוצג דטה תלת ממד' בעל מבנה בצורת האות S. בנוסף, לדטה ישנו מאפיין נוסף המבטא במצבים שונים ובעבר שלהם מכחול לאדם. טרנספורמציה LINARIOT PCA (סיבוב/מתיחה) אינה מצליחה לשמר את מבנה הדטה כגרדיאנט בעמ' מכחול (ראה אייר עליון ימינו). אולם, שיטות לא LINARIOT יכולות ומצליחות להוריד את הממדיות של הבעה ולשמור את מבנה הירעה הטופולוגית (ראה שני איורים תחתונים).

לשם כך, השתמש באותו מערכן נתונים, $\mathbb{R}^{M \times N} \in \hat{X}$, כאשר M הוא מספר הדוגמאות, ו- N הוא מספר המאפיינים (או המשתנים/פיצרים). חשוב לשים לב כי כל מדידה מיוצגת על ידי וקטור שורה \vec{x}_m . הרעיון הכללי של השיטה הוא לmaps את סט המדידות באופן צזה שמדוות יותר, קרי מדידות "קרובות" יותר במרחב ה- N ממד', יוצגו על ידי נקודות קרובות יותר במרחב חדש K -ממד', כאשר לרובה $3 \leq K$. נסמן את המרחב המקורי \mathcal{X} ואת המרחב החדש ב- \mathcal{Y} , כאשר בשני המרחבים המדידות מוצגות על ידי נקודות בגרף פיזור (scatter plot). המטריקה המשמשת למדידת דמיון (similarity) בין שתי נקודות במרחב המקורי \mathcal{X} הינה הסתברותית. עברו שתי מדידות m_1, m_2 במרחב המקורי ה- N -ממד', ההתפלגות הנורמלית המשותפת P_{m_1, m_2} הינה:

$$P_{m_1, m_2} = \frac{Z_1^{-1}}{2N} \exp\left(-\frac{\|\vec{x}_{m_1} - \vec{x}_{m_2}\|^2}{2\sigma_1^2}\right) + \frac{Z_2^{-1}}{2N} \exp\left(-\frac{\|\vec{x}_{m_1} - \vec{x}_{m_2}\|^2}{2\sigma_2^2}\right)$$

כאשר σ נקרא פרפלקסיות (perplexity) והוא פרמטר שקבע מראש, ו- Z הינו קבוע הנורמלייזציה, המוגדר על ידי $Z = \sum_{k \neq i} \exp\left(-\frac{\|\vec{x}_i - \vec{x}_k\|^2}{2\sigma_i^2}\right)$. עבור נקודות קרובות יותר, עבור הביטוי $\|\vec{x}_{m_1} - \vec{x}_{m_2}\|$ קטן, ההסתברות שהנקודה \vec{x}_{m_1} שכנה של \vec{x}_{m_2} גדולה. לעומת זאת כאשר הנקודות רחוקות זו מזו, ככלומר $\|\vec{x}_{m_1} - \vec{x}_{m_2}\|$ גדול, ההסתברות שהנקודה \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 שהמרחב של \vec{y}_m הינו נמוך (2 או 3 ממדים). בנוסף, נדרש שנקודות דומות ("שכנות") במרחב \mathcal{X} , ישארו שכנות לאחר המיפוי למרחב \mathcal{Y} . מתברר שפונקציית ההסתברות המותנית, המתאימה לתיאור דמיון בין נקודות שכנות במרחב החדש \mathcal{Y} , הינה התפלגות t, הנקראת גם ההתפלגות סטודנט עם דרגת חופש אחת (נדון בראען לבחור בפונקציות הסתברות אלו בהמשך). כך, נכמת את הדמיון בין m_1 לבין m_2 , על ידי ההסתברות המשותפת Q_{m_1, m_2} המוגדרת באופן הבא:

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

$$\text{כאשר } \Sigma_{k \neq j} \left(1 + \left\| \vec{Y}_k - \vec{Y}_j \right\|^2 \right)^{-1}$$

המיופיע בין מרחב המקורי \mathcal{X} לבין המרחב החדש \mathcal{Y} הוא מיטבי אם הוא "שמר" את השכנות של נקודות (מדידות) קרובות. לשם כך נגדיר את פונקציית המחריף על ידי Kullback-Leibler divergence, הבוחן מרחק בין שתי התפלגיות:

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

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

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

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

א. אתחול: * נתון $X \in \mathbb{R}^{N \times M}$.

* היפר-פרמטר לפונקציית הדמיון: בחירת השונות σ^2 .

* בחירת שיטת אופטימיזציה (למשל: ADAM, SGD), והיפר פרמטרים כגון: קצב הלמידה ϵ , מומנטום $(t)\alpha$, גודל ה-batch וכו'.

ב. חשב את P_{m_1, m_2} .

ג. אתחל את המיפוי $(s \hat{f}_M) \sim N(0, s \hat{f}_M) \sim \mathcal{Y}_1, \mathcal{Y}_2, \dots, \mathcal{Y}_M$ [ז"א בחר את הערכים ההתחלתיים לפי התפלגות גausianית עם ממוצע 0 וסטייה תקן s (נבחר להיות קטן, נניח $10^{-4} s = \hat{f}_M$ מטריצת יחידה.)]

ד. עברו איטרציה t :

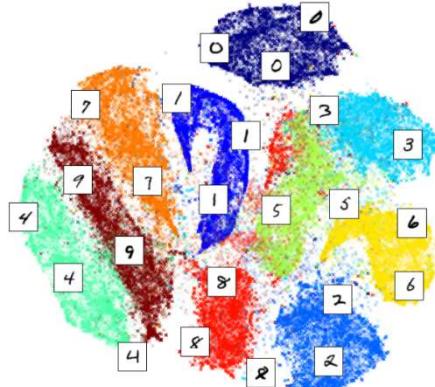
* חשב את Q_{m_1, m_2} עבור ה-batch הנבחר.

* חשב את הגרדיינט של פונקציית המחריף $\frac{\delta C}{\delta \mathcal{Y}}$.

* עדכן: $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)$

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 הספורות 0 עד 9, המיצגות על ידי תמונות בממד גובה 28×28 , למרחב דו-ממדי. בדוגמה זו נלקחו 6,000 תמונות של ספרות ומיפוי אותן למרחב דו-ממדי. במרחב זה ניתן לראות בבירור כיצד כל תמונה מופתת לאזור אחר, כיוון שבפועל נוצרו עשרה אשכולות שונות, המובחנים בצורה ברורה אחד מהשני. ביצוג הדו-ממד אין משמעות לציריהם, כיוון שבאלגוריתם זה יש חשיבות רק למורקח היחס בין הנקודות.



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

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

א. סימטריה:

פונקציית הדמיון הגאומטרית בין שתי נקודות במרחב X הינה פונקציה סימטרית, כלומר $P_{m_1, m_2} = P_{m_2, m_1}$. אולם ניתן להגיד גם פונקציית דמיון א-סימטרית, המבוססת על התפלגות מותנית (במקום התפלגות משותפת). הפונקציה המותנית נתונה על ידי:

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

כך ש:

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

ב. בחירת פונקציית הדמיון במקומות פונקציית t :

באלגוריתם שתואר, פונקציית הדמיון בין שתי נקודות במרחב ה- U נתונה על ידי התפלגות t . ניתן להגיד גם פונקציה אחרת, למשל את פונקציית דמיון גאומטרית עברו שתי מדידות במרחב U . שיטה זו נקראת SNE, והגדיאנט של פונקציית המהיר במקורה זה נתונה על ידי:

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

אולם, פונקציית דמיון גאומטרית במרחב U יכולה לגרום לכך שנקודות לא מאוד קרובות במרחב X , ימופו לנקודות קרובות במרחב U . זה קורה מכיוון שההתפלגות גאומטרית במרחב U גורמת לאטרקטטור (משיכה) יחסית חזק בין שתי נקודות, גם במקרים בהם הנקודות אין ממש קרובות. לעומת זאת, כאשר פונקציית הדמיון הינה התפלגות סטודנט t , שההינה התפלגות עם זנב כבד יותר, שתי נקודות שאין ממש קרובות ימופו彼此ה רואיה למרחב U כך שאין "נשכות" או מתקרבות זו לזו. שיטה אחרת, הנקראת t-SNE, מציעה להשתמש בתפלגות איחודית, אך גם לה חסרן דומה ל-SNE, כאשר שתי נקודות לא מאוד דומות זו לזו, אין "דוחות" אוחת את השניה.

לשיטת SNE-t יש שלוש מגבלות עיקריות

א. הורדת מד: השיטה משתמש לויזואלייזציה של מידע מממד גובה דו-ממד או תלת-ממד. אולם, באופן עקרוני, ניתן להוריד את הממד לא שם הציגו, אלא לצרכים אחרים, כאשר הממד החדש הינו גדול מ-3. ניתן ובממד גובה פונקציית התפלגות סטודנט t עם דרגת חופש אחת, אשר לה משקל גובה יחסית

- במרחקים גבוהים, לא תשמר את המבנה של המידע המקורי. לכן, כאשר נרצה להוריד לממד גובה מ-3, פונקציית התפלגות t עם יותר מדרגת חופש אחת מתאימה יותר.
- ב. קלילת המדדיות: t-SNE- t מבוססת על מאפיינים מקומיים בין נקודות. השיטה, המבוססת על מטריקת מרחק אוקלידי, וכר מנicha לנאריות מקומית על גבי הירעה המתמטית בה מתקיימות הנקודות. אולם, במקרים מסוימים בו הממד הפנימי גבוה, שיטת t-SNE- t עלולה להיכשל כיוון שהනחת הלינאריות לא מתקיימת. לעומת זאת, ישנן מספר שיטות למצוור תופעה זו, עדין, בהגדלה, כאשר הממד הפנימי גבוה, לא ניתן להוריד ממד אחד שמבנה המידע ישמר באופן מלא.
- ג. פונקציית מחיר לא קמורה: הרבה שיטות למידה מבוססות על פונקציית הפסד קמורה, כר שתיאורטית מציאות אופטימיזציה (יחידה) לפונקציה זו אפשרית תמיד. אולם, בשיטת t-SNE- t , פונקציית המחיר אינה קמורה, והפתרון המתתקבל על ידי האופטימיזציה משתנה בהתאם לפרמטרים הנבחרים.

2.3.3 Uniform Manifold Approximation (UMAP)

UMAP הינה שיטה לא לנארית נוספת עבור הורדת ממד, המבוססת על קירוב של ירעה טופולוגית (manifold). השיטה מבוססת על מספר הנחות בסיס: ראשית, מנייחים כי הנקודות/הדוגמאות מתפלגות באופן אחיד על גבי ירעה טופולוגית כלשהיא. בנוסף, מנייחים כי ירעה זו קשירה באופן מקומי. מטרת קירוב זה היא לשמר את המבנה של הדוגמאות על גבי הירעה. נעיר כי הוכחה ריגורוזית של השיטה מעורבת מתמטיקה מתקדמת (מעבר לנלמד בתואר הראשון), אך ניתן להבין העיקרי גם ללא הוכחה המדוקת, ועל כן נציג ראשית את הרעיון העיקרי, ולאחר מכן הבסיס המתמטי העומד מאחוריו. השיטה מחולקת לשני שלבים עיקריים – בשלב הראשון מחשבים משקל של קשר בין כל שתי דוגמאות במרחב, ולאחר מכן מבצעים את הורדת הממד על סמך משקל הקשרות:

1. ראשית, נרצה לתאר את הירעה הטופולוגית בה נמצאות כל הדוגמאות במרחב המקורי, \hat{X} , על ידי גרפם מושקל-קשיר. על מנת ליצור גרף זה יש לבצע את השלבים הבאים עבור כל דוגמא X_{m_i} (שהיא כאמור נקודה במרחב המקורי N -ממדי):

- מציאת את k השכנים הקרובים ביותר לעצמו במרחב האוקלידי (\hat{X}). $d_{i,j} \equiv d(X_{m_i}, X_{m_j})$.
- чисוב המרחק לנכודה הקרובה ביותר – נסמן אותו ב- $P_{i|j}$.
- чисוב הגודל σ_i על ידי פתרון המשוואה:

$$\log_2(k) = \sum_{j=1}^k \exp\left(-\frac{\max\{0, d_{ij} - \rho_i\}}{\sigma_i}\right)$$

המשקל (הסימטרי) של קשר בין שתי נקודות – X_{m_j} ו- X_{m_i} הינו:
 $P_{ij} \equiv P_{i|j} + P_{j|i} - P_{i|j} \cdot P_{j|i}$
 כאשר:

$$P_{i|j} = \exp\left(-\frac{\max\{0, d_{ij} - \rho_i\}}{\sigma_i}\right)$$

כלומר: עבור X_{m_j} שאין בקבוצת k השכנים הקרובים של X_{m_i} : $P_{i|j} = 0$, ואםשתי הנקודות הן i "שכנות", אז מתקיים: $P_{i|j} = \exp\left(\frac{\rho_i - d_{ij}}{\sigma_i}\right)$

2. שנית, על מנת לבצע הורדת ממד, נרצה ליצור הצגה של הגרף המושקל בממד נמוך ממרחב \hat{X} , כאשר לגוף החדש יש "מבנה דומה" לגוף המקורי. לשם כך, נגדיר פונקציית דמיון באופן הבא:

$$q_{i,j} \equiv \left(1 + a \|\vec{Y}_{m_1} - \vec{Y}_{m_2}\|^{2b}\right)^{-1}$$

כאשר פונקציית המחיר הינה:

$$C = \sum_{i \neq j} P_{ij} \log\left(\frac{P_{ij}}{q_{ij}}\right) + (1 - P_{ij}) \log\left(\frac{1 - P_{ij}}{1 - q_{ij}}\right)$$

נעיר כי מספר השכנים הקרובים k , והמרחב המינימלי האפקטיבי (הנקבע על ידי a, b) הינם היפר-פרמטרים הנבחרים על ידי המשתמש.

הבדלים העיקריים בין שיטת t-SNE- t לבין UMAP:

- א. בשיטת SNE-t המרחקים האוקלידיים מחושבים כל זוג נקודות. בשיטת UMAP, נלקחים בחשבון רק קבוצת שכנים קרובים.
- ב. הסימטריזציה בשיטת SNE-t נתונה על ידי: $P_{i,j} = \frac{P_{i|j} + P_{j|i}}{2N}$. לעומת זאת, הסימטריזציה בשיטת UMAP נתונה על ידי $P_{i,j} = P_{i|i} + P_{j|i} - P_{i,j}$.
- ג. פונקציית הדמיון במרחב המקורי בשיטת SNE-t הינה גאומטרית. לעומת זאת, בשיטת UMAP פונקציית הדמיון במרחב \mathcal{X} הינה אקספוננטית.
- ד. עבור UMAP פונקציית המשקל (הדמיון) במרחב החדש \mathcal{Y} הינה $q_{i,j} \equiv \left(1 + a \left\| \vec{Y}_{m_i} - \vec{Y}_{m_j} \right\|^{2b}\right)^{-1}$ כאשר $a = b = 1$ ובנוסף פונקציה זו מונומלת (מכפלת במקדם $^1/3$), אנו מקבלים את פונקציית המשקל של מרחב \mathcal{Y} עבור SNE-t.
- ה. לפונקציית המחיר (או הפוטנציאלי) עבור שיטת UMAP נוסף איבר מהצורה $\log \left(\frac{1-P_{ij}}{1-q_{ij}} \right)$. איבר זה מוריד את סיכוי לקבלת שתי נקודות קרובות מאוד במרחב החדש (זאת אומרת $1 \sim q_{ij}$) כאשר הם אינם קרובות במרחב המקורי.

תיאור מתמטי של השיטה:

נניח שהיריעה הטופולוגית בה המידע (data) נמצא, אינה ידועה מראש, אולם בכל זאת נרצה להגדיר מרחקים גאודזיים על גבי יריעה זו. בנוסף, נניח כי הדוגמאות מפוזרות באופן אחיד על גבי יריעה זו. באופן כללי, הגדרת מטריקת רימן (מטריקה, פונקציה מסוימת, שמאפשרת לנו הגדרת מרחקים, ראה דוגמא למטה), תלויה בסביבת הנקודה. המטריה הראשונית שלנו היא להגדיר את היריעה הטופולוגית בה הנקודות נמצאות על ידי גרפם משקל קשור.

עתה, נניח שבסבירות נקודה x המטריקה g אלכסונית וקבועה. לכן, ניתן להגדיר את המרחק בין נקודה x לנקודה y שנמצאת בסביבת x על ידי המטריקה הקיימת במרחב \mathcal{M} .

מתוך ההנחה שהנקודות מפוזרות בצורה אחידה על גבי היריעה, מתקיים הדרישה הבאה. רדיוס של כדור סביב נקודה x (הdogsma ϵ -ית של מערכ הנטוינום הרב ממד') כר שהכדור מכיל k שכנים קרובים ל x , הוא קבוע ואינו משתנה בין הנקודות.

אם כך, בידנו משפחת מטריקות מקומיות, אחת לכל דוגמא i -ית, שעיל ידה ניתן להגדיר את המרחקים ל k שכנים קרובים. נרצה "לאחד" את משפחת המטריקות המקומיות הללו, למרחב גלובלי. לשם כך, השתמש ב-*fuzzy simplicial sets* [לצורך הדיוון קבוצות אלו מכילות סימפלקס – מבנה מסוים של גרפף/רשת – כר שלכל איבר יש דרגת שייכות לקבוצה]. נשים לב כי על ידי איחוד הקבוצות הללו, יוצרים גרף ממושקל-קשיר. עבור גרפם זה, כל נקודה מקושרת אר ורך לקבוצת k השכנים הקרובים לה, כאשר עבור שכנים קרובים ביותר, משקל הקשת המחברת גבוי יותר משקל קשת עבור שכנים רחוקים יותר. נשים לב כי חישוב המשקלים ו"קישור" עבור שכנים קרובים בלבד מאפשרת לשזרור את בניית היריעה הטופולוגית ביותר דיק (ראה איור).

נניח, בדומה לפרק הקודם, כי מערכ נתוניים המקורי מסומן על ידי, $\hat{X} \in \mathbb{R}^{M \times N}$, כאשר M הוא מספר הדוגמאות, ו- N הוא מספר המאפיינים (או המשתנים). חשוב לשים לב כי כל מדידה מייצגת על ידי וקטור שורה \hat{x}_m . נרצה למפות מדידות אלו ל- $\mathbb{R}^{M \times D}$ $\hat{Y} = \{\hat{y}_1, \dots, \hat{y}_M\}$, כר שמדידה i במרחב המקורי \mathcal{X} : \hat{x}_i , מייצגת על ידי וקטור חדש \hat{y}_i במרחב החדש \mathcal{Y} , כר שמתאים $N \ll D$, זאת אומרת הממד של הווקטור החדש הרבה יותר קטן מהමמד המקורי.

עתה, על מנת להשוות בין ייצוג ב-*fuzzy simplicial sets* עבור \hat{X} ו- \hat{Y} , השתמש ב-*cross entropy*. נניח שדרגת השיכיות של איבר a בקבוצה A נתון על ידי פונקציה $\mu: A \rightarrow [0,1]$. הסימון המקורי עבור a הוא (A, μ) . פונקציית cross entropy עבור (A, μ) מוגדרת להיות:

$$C[(A, \mu), (B, \nu)] \equiv \sum_{a \in A} \left[\mu(a) \log \left(\frac{\mu(a)}{\nu(a)} \right) + (1 - \mu(a)) \log \left(\frac{1 - \mu(a)}{1 - \nu(a)} \right) \right]$$

עבור קבוצת המדידות במרחב המקורי: $\{ \vec{X}_1, \vec{X}_2, \dots, \vec{X}_M \} \subseteq \mathbb{R}^+$, נסמן מטריקה $\{0\}$ על $\hat{X}: \hat{X} \times \hat{X} \rightarrow \mathbb{R}^+$: המכמתת את אי-הדמיון בין שתי מדידות; קרי מדידות דומות יציגו מרחק קטן ביניהן, לעומת זאת מדידות שאינן דומות בהן המרחק המוגדר על ידי המטריקה ימצא להיות גדול. נגדיר תת-קבוצה של k השכנים הקרובים, על פי המטריקות d , של

מדידה \vec{X}_i , ונסמנה באופן הבא: $\{\vec{X}_{i_1}, \vec{X}_{i_2}, \dots, \vec{X}_{i_k}\}$. עבור כל מדידה i , נחפש את המרחק הקטן ביותר בינה לבין השכנים שלה, זאת אומרת:

$$\rho_i = \min \left\{ d(\vec{X}_i, \vec{X}_{i_j}) : 1 \leq j \leq k, d(\vec{X}_i, \vec{X}_{i_j}) > 0 \right\}$$

בנוסף, נחפש σ_i כר שמתקיים:

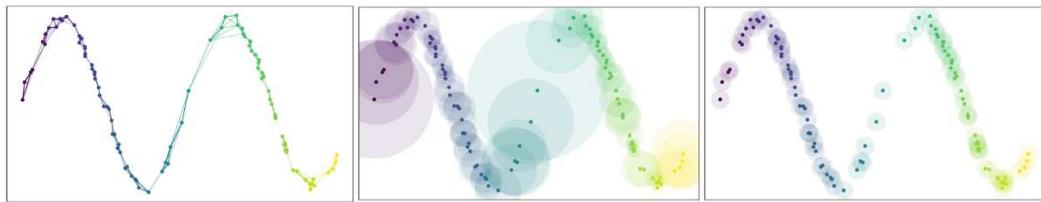
$$\sum_{j=1}^k \exp \left[-\frac{\max(0, d(\vec{X}_i, \vec{X}_{i_j}) - \rho_i)}{\sigma_i} \right] = \log_2 k$$

כעת, נוכל להגיד גראף מכוון ומ משקל (weighted directed graph) (V, E, w) (weighted directed graph) הינה קבוצת הקווים E הינה קבוצת המדידות \vec{X} , קבוצת הקשתות (edges) נקבעת על ידי k השכנים הקרובים לכל נקודת מדידה:

$$E = \left\{ (\vec{X}_i, \vec{X}_{i_j}) : 1 \leq i \leq M, 1 \leq j \leq k \right\}$$

והמשקל עבור כל קשת מוגדר על ידי:

$$w[(\vec{X}_i, \vec{X}_{i_j})] = \sum_{j=1}^k \exp \left[-\frac{\max(0, d(\vec{X}_i, \vec{X}_{i_j}) - \rho_i)}{\sigma_i} \right] \equiv p_{i,j}$$



איור 2.16 לפניו נקודות הנמצאות על גבי יריעה טופולוגית שהינה בקירוב בצורת סינוס. באור מיימן ניתן לראות כי הנקודות אינן מפוזרות באופן אחיד על גבי יריעה זו, אך אם נקשר רק נקודות שכנות (הנמצאות באותו סיבוב כל נקודה) נקבל גראף / יריעה שאינה קשירה. אם לעומת זאת נדרש שסיבוב הנקודה המכילה k שכנים הקרובים הינם כדור בעל רדיוס מסוים, נמצא כי רדיוס כדור זה אינו קבוע בין הנקודות (ראה אוור אמצע). לשם כך, ההנחה הבסיסית של שיטת MAP/MAPU הינה שהנקודות מפוזלות בצורה אחדה על גבי יריעת. כעת, לאחר שהחנו התפלגות אחדה של הנקודות על גבי היריעה, נוכל להגיד את משקל הקשתות בין שכנים הקרובים (ראה איור שמאל). נשים לב כי באופן זה מבנה היריעה המתואר על ידי הגרף הקשור אכן נראה בצורה סינוסי כנדרש.

סוף מתמטית:

בהינתן מטריקת רימן $(g_{\mu,\nu})$ עבור סביבה נקודה x על גבי היריעת, האורך של העקום $\gamma(x_i, x_f)$ נתון על ידי:

$$\|\gamma(x_i, x_f)\|_g \equiv \int_{x_i}^{x_f} \sqrt{\sum_{\mu} \sum_{\nu} g_{\mu,\nu}(p) dx_{\mu} dx_{\nu}} = \int_{x_i}^{x_f} \sqrt{\sum_{\mu} \sum_{\nu} g_{\mu,\nu}(p) \frac{\partial x_{\mu}}{\partial t} \frac{\partial x_{\nu}}{\partial t}} dt$$

כאשר הסכימה הכפולה (n, μ) על וקטורי הבסיס של העקומה. נבחר את העקומה שנוננת מרחק מינימלי.

דוגמאות:

דוגמה לירעה דו-ממדית הינו משטח אוקלידי דו-ממדי \mathbb{R}^2 . מטריקת רימן עבור יריעת זו, אינו תלוי בנקודה x , ומוגדר על ידי: $\delta_{\mu,\nu} = g_{\mu,\nu}$. עבור עקום במישור הדו-ממד: $(x(t), y(t)) = (x, f(t))$. פרמטריזציה של עקום זה נתונה על ידי $\gamma(t) = (t, f(t))$. אורך עקום זה נתון על ידי:

$$\|\gamma(x_i, x_f)\| \equiv \int_{x_i}^{x_f} \sqrt{\sum_{\mu=\{x,y\}} \sum_{\nu=\{x,y\}} \delta_{x_{\mu}, x_{\nu}} \frac{\partial x_{\mu}}{\partial t} \frac{\partial x_{\nu}}{\partial t}} dt = \int_{x_i}^{x_f} \sqrt{\left(\frac{\partial x}{\partial t} \right)^2 + \left(\frac{\partial f(t)}{\partial t} \right)^2} dt$$

עקומה $\gamma(t)$ שנוננת מרחק מינימלי בין נקודות מוגדרת על ידי $\gamma(t) = (t, mt + n)$, כאשר $m, n \in \mathbb{R}$, כך ש:

$$\int_{x_i}^{x_f} \sqrt{1 + m^2} dt = \sqrt{1 + m^2} (x_f - x_i) = \sqrt{(x_f - x_i)^2 + m^2 (x_f - x_i)^2} = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$$

מצופה.

גם ספירה דו-ממדית (מסומנת לעתים על ידי \mathbb{S}^2 , לדוגמה גלובוס) מהויה ירעה דו ממדית. שם, מטריקת רימן מוגדרת על ידי $g = \begin{pmatrix} 1 & 0 \\ 0 & \sin^2 \theta \end{pmatrix}$. לכן, עבר עקום על גבי גלובוס; $(t) \gamma = (\phi, \theta)$ נקבע:

$$\|\gamma\| \equiv \int_{x_i}^{x_f} \sqrt{\left(\frac{\partial \theta}{\partial t}\right)^2 + \sin^2 \theta \left(\frac{\partial \phi}{\partial t}\right)^2} dt$$

2.4 Ensemble Learning

2.4.1 Introduction to Ensemble Learning

נניח ויש בידינו אוסף נתונים מסוים, ורוצים לבנות מודל המנתה את הנתונים האלו שמתבסס על אלגוריתם מסוים. כמעט תמיד, המודל לא יהיה מדויק במידה כלשהי, והוא יהיה בעל שונות או בעל הטיה. ניתן להשתמש במכולול (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 בננה באופן פשוט.

ב כדי להבין את יותר טוב את החשיבות של שימוש ב-ensembles, יש להרחב על ה-Trade off בין שונות המודל להטיה שלו. מודל אופטימי יתאפשר בשונות נמוכה ובהטיה נמוכה. לעומת זאת, השוני בין התוצאות לא יהיה מהותי, ובמוצע התוצאות תהיה קרובה מאוד לערך האמתי. מודל זה יהיה מודל אמין, ונוכל לבסס עליו את צעדנו. למרבה הצער, מודל שכזה לרוב איינו אפשרי. סוג אחר של מודל יהיה המודל הגרוע, ההפור למודל האופטימי. זה מודל עם שונות גבוהה והטיה גדולה. מודל שכזה יציג טווח רחב של תוצאות על אותן נתונים, ובמוצע יהיה הרחוק מאוד מהערך האמתי. מודל זה כלל איינו שימושי.

בפועל, המודלים במציאות יונעו לאורך חיי קצאות: מודלים עם שונות גבוהה והטיה נמוכה, ומודלים עם שונות נמוכה והטיה גבוהה. הדיזהו של המקום שלמו לאורך זמן זה קרייטי, כיוון שהוא מאפשר לנו לבחור את דרך ההתמודדות הטובה ביותר. יש מספר משפחות של ensembles model, ושני העיקרים בהם נקראים Bagging and Boosting. כאשר ניתקל במודלים עם שונות גבוהה, ככלומר מודל הסובל-*Overfitting*, לרוב נרצה להשתמש באנסטבל מסווג Bagging על מנת להוריד 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 שפעילים ביחד, על-מנת להגעה לתוצאה משופרת. כאמור, אלגוריתמים מסוג bagging נועד להגדיל את יציבות המודל והעלאת הדיק שלו, זאת תוך הורדת השונות והימנעות מ-*overfitting*. Bagging מורכב ממספר רב של אלגוריתמים, המכונים "לומדים חלשים" (Weak learners), כאשר כל אחד מהם מבצע למידה ותחזית על חלק מן הנתונים, מתוך מטרה להגעה לתוצאה אינטואטיבית. אלגוריתם bagging הינו שיטה נפוצה ופושטה יחסית לשיפור ביצועים, אם כי הוא עשוי להיות יקרה מבחינה חישובית.

מודל "פשוט" יקבל את הנתונים, יתאמן עליהם ויבצע תחזית על נתונים חדשים. זה תהליך הלמידה והבחן אשר ידוע לנו ממודלים כגון עץ החלטה (Decision Tree), רגראטיון ליניארית וכו'. כפי שהסביר לעיל, מצב זה עשוי להוביל להתקמת יתר של המודל לנ庭ו האימון, דבר שעשוי להוביל למודל בעל שונות גבוהה. ב כדי להתמודד עם בעיה זו, אלגוריתמים מסוג 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 חזורת) 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, ככלומר, יוצרים מודל נפרד לכל קבוצה של נתונים. בשלב הסופי, יבצעו יחדו של כל המודלים על מנת ליצור מודל משוקל בעל שונות קטנה יותר מאשר מודל המסתמך על הדאטה המקורי כפי שהוא.

אוף חיבור המודלים המתבצע ב-*bagging* מבוסס על אותו רעיון של NN-K, אשר מודלים אלו יכולים לשמש הן למטרות סיוג והן למטרות רגראטייה. כאמור לעיל (בפרק 2.1.3), אלגוריתם השכן הקרוב כל "שכן" העיד על התווית שלו, ולאחר הכרעת הרוב נקבעה התווית של התצפית החדשה. במקרה שבו נספור את תדיות כל התוויות השכנות, התווית הנבחרת תהיה של התצפית הנפוצה ביותר; נעשה זאת כאשר NN-K יעבד כמסוג. במקרים בהם NN-K יעבד כרגראטייה, יבצע מסווג של כל התוויות השכנות, וזאת גם תהיה התחזית. כאשר *bagging* יעבד כמסוג, כל

weak learner יבצע תחזית, והתווית השכיחה ביותר תהיה התוצאה של האנסמבל (-הכרעת הרוב). כאשר בעוד כרגסיה, כל מודל יבצע תחזית, אבל התוצאה של האנסמבל תהיה הממוצע של כל המודלים.

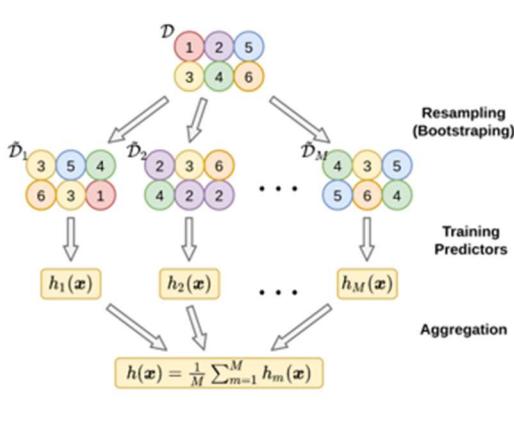
באופן פורמלי, עבור דאטה $\mathcal{D} \in \mathbb{R}^{n \times d}$, ניצור M קבוצות חדשות באוטו גדול של הדאטה המקורי – עבור כל קבוצה $X_m \in \mathbb{R}^{n \times d}_{m=1} \{X\}$, ועבור כל קבוצה c_m נבנה מודל $c_m(x)$. עבור עיות סיווג ההחלטה מתקבל על פי הצביעת הרוב:

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

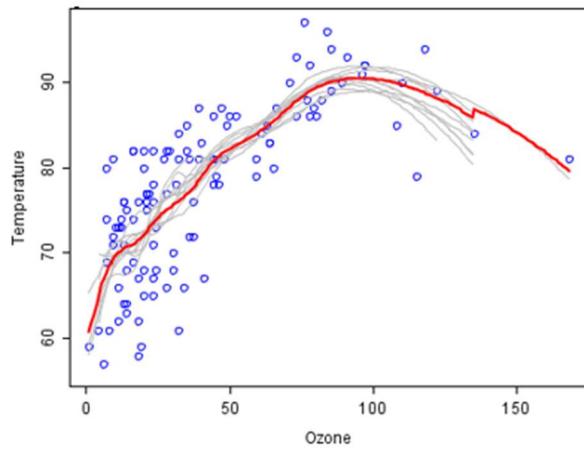
ועבור עיות רגסיה ההחלטה תבוצע באמצעות מיצוע כל המודלים:

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

a



b



איור 2.17 (a) אלגוריתם Bagging: בשלב ראשון יוצרים הרבה מחלקות שונות שנוצרות מהדאטה המקורי (Bootstrapping), ולאחר מכן מודלים המתאימים לכל מחלקה (Aggregating). (b) דוגמא לבניית מודל רגסיה באמצעות 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 שיפורים חדים, מה שמקטין את ה-overfitting, וממייל מפחית את השונות. ניתן להבין זאת על ידי דוגמא פשוטה – נניח שיש התפלגות נורמלית (σ, μ) , אז השונות של n דגימות בלתי תלויות הינה $\frac{\sigma^2}{n}$.icut גענשטיין ומנצחים *o* ניסויים שככל אחד מהם דוגמים *a* 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, כלומר ניקח m גדול מאוד, נקבל:

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

ובכך הכל השונות הסופית הינה בקשר $\rho \sigma^2$, וביתוי זה לרוב קטן מאשר השונות של מודל המבוסס על הדאטה המקורי ללא שימוש ב-ensembles. ניתן לשאול שהקורסיב בין הקבוצות קטנה, כך השונות של המודל המשוקל גם כן קטנה יותר.

מודל נפוץ מאוד מסווג bagging הוא Random Forest. אלגוריתם זה משלב בין עצי החלטה לבין הרעיון הבסיסי של bagging, כאשר הוא מפצל את הנתונים ואת הממשנים לעצי החלטה רבים, וכל אחד מהם מקבל חלק מסוים מן השלם. העצים הם בעלי שונות גבוהה, כלומר – כל אחד מהם overfitting הוא בפני עצמו, אך עם זאת הקורלציה ביןיהם נמוכה, מה שמקל על הורדת השונות והימנע 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 לערבים קיצוניים (Outliers). יכולת העבודה שלו במקביל עשויה לאפשר לו להגיע לתוצאות באופן מהיר יותר.

עם זאת, ל-*bagging* יש גם חסרונות. הוא אינו מוריד את ההתיה, ולכן ניתן לא להתאים במרקם רבים. במקרים של בינה מלאכותית יש חשיבות רבה לכלי הפרשנות של המודל; לרוב, ידרש הסבר פחות טכני של תוצאות המודל למוקבי החלטות או לצרכנים. הם עשויים לא לקבל כלל החלטות של מודל שראה כ"קופסה שחורה". יש קושי רב לתת פרשנות החלטות של מודלים מבוסס *bagging*, והדבר מתקשה על השימוש בו. מעבר לכך, ישנו *bagging* עשוי להיות יקר מבחינה חישובית. עקב כך, הוא שימושי מאוד במקרים בהן שיפור זעיר עשוי להוביל להצלחה, אך לרוב תינון עדיף למודלים פשוטים יותר של *ensembles*.

2.4.3 Boosting

כאמור, המושג *boosting* מתייחס למשפחה של אלגוריתמים המשמשים באוסף של מודלים "חלשים" על מנת ליצור מודל אחד "חזק", כאשר מודלים אלו מתמקדים אלו בŃביסיון להפחית את התהיה שיש למודל. מבחינה אינטואטיבית, מודל חלש הוא כזה שתוצאותיו מעט טובות יותר מאשר אקדמי בעוד שאחד חזק מתקדם לביצועים אופטימליים. בינווד לטכניקות *ensemble* אחרות שפועלות במקביל, העקרון המנחה כאן הוא לשרשר את המודלים באופן כזה שכל מודל שמתווסף יטפל בשגיאות שקדמי פספסו. היפי נוצע בכך ש-*boosting* מוכיח כי למידה חלה בהכרח מצביע על קיום של שיטת למידה חזקה, לרוב, מודלים מבוסס *boosting* מתמקדים בבעיות סיווג בגיןari.

באופן פורמלי, המושגים "לומד חלש" ו-"לומד חזק" עברו בעייה סיווג בגיןari מוגדרים כך: אלגוריתם נקרא לומד חזק אם לכל $0 < \delta, \epsilon$ האלגוריתם מסוגל ($\text{עבור אוסף נתונים גדול מספיק}$) לבנות מסוווג $(x)_c$ שמקיים $< (y) \neq (x)_c$ בהסתברות גדולה מ- $\delta - \epsilon$. לומד חלש הינו אלגוריתם שלכל $0 > \delta$ קיים $0 < \gamma$ כך שעבור אוסף נתונים מספיק גדול, האלגוריתם מסוגל לבנות מסוווג שמקיים $\gamma - \frac{1}{2} < (y) \neq (x)_c$ בהסתברות גדולה מ- $\delta - \epsilon$. כאמור, המטרה של *boosting* הינה לחתך אוסף של מודלים חלשים ובעזרתם ליצור מודל חזק, כאשר הצלחו להוכיח שניתן להפור כל לומד חלש לומד חזק על ידי בניית קומבינציה ליניארית של מסוווגים אשר נוצרו בעזרת הלומד החלש.

נמחיש את הרעיון של *boosting* בעזרת דוגמא: נניח שיש לנו נתונים X , המוחלק באופן אקראי לשולש קבוצות שוות (כל אחת מכילה שליש מהנתונים) – x_1, x_2, x_3 .icut בונים מודל לצורך סיווג בגיןari, המסומן b_1-h . נמצאו כי b_1 מתאים בצורה טובה רק לקבוצות x_1, x_2 אך מסוווג בצורה לא טובאה את פריטי הקבוצה x_3 . כיוון ש- x_3 מכיל שליש מסך הנתונים, שגיאת הסיווג גדולה $-b_1(x)$ הוא מודל חלש, ונרצה לשפר אותו. כדי לעשות זאת ניקח רק חלק מהנתונים, $X \in X'$, ונdag לחק- X' יכול הרבה מאברי x_3 .icut בונה מודל נוסף $(x)_c$ על בסיס X' , מתוך כוונה שמודל זה יתמקד גם בקבוצה x_3 ויסוווג את איבריה בצורה טובאה.icut בונה מודל זה אכן מסוווג בצורה נאותה את איברי x_3 , אך הפעם המודל שוגה בצורה גסה בסיווג איברי x_2 . עקב השגיאה בסיווג x_2 המודל השני גם הוא מודל חלש, אךicut בונה שני מודלים חלשים שהחולשה בכל אחד נבעת מקבוצת איברים אחד של אוסף הנתונים המקורי X . אם נמצא דרך הולמת לחבר את שני המסוווגים, יוכל ליצור מודל בעל פוטנציאל להצלח לסוווג את X כמו שציר.

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

1. אתחול הנתונים: $D = \mathcal{D}_1$
2. עבור $t = 1, \dots, T$:

$$c_t(x) = L(\mathcal{D}_t)$$

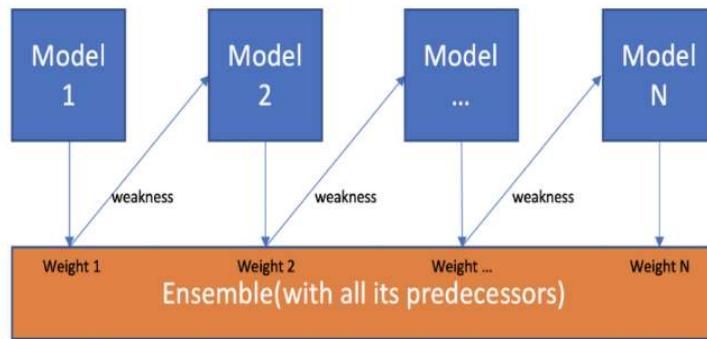
חישוב שגיאת המסוווג: $\epsilon_t = P_{x \sim D_t}(c_t(x) \neq f(x))$

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

3. 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 באlgorigithm *boosting*, ונפרט את המרכזיות שבהן.

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



איור 2.18 – סכמת כללית של boosting. המודלים (במקרה זה מזוהה בעץ החלטה רדווד, אך זה תקין לכל מודל חלש) מחוברים אחד לשני באופן שכל אחד לומד מהתפלגות המשווקלת בהתאם לשגיאות של המודלים הקודמים.

Adaptive-Boosting (AdaBoost)

AdaBoost היא אחת הטכניקות הראשונות של boosting, ועל אף שהיימורות הטכנית של boosting נוספות, היא בין הפופולריות ביותר בתחום (אם כי יש לה מטוטל של וריאנטים). העוצמה הגלומה בטכניקה זו נובעת מכך שגם בהינתן מספר מאפיינים רב, האלגוריתם מצליח להיפגע פחות מ"קלחת הממדיות" ולשמור על יכולות ניבוי טובות, בניגוד לאלגוריתמים אחרים של סיווג, כמו למשל SVM או אפילו רשתות נeurונים.

זכור, תחת ההנחה שקיים 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 ההיפופוזזות של כל המודלים לפטל אחד. גישה זו מבוסנת נאיבית ופשטנית, ואני לוקחת בחשבון מקרה בו מרבית המודלים שוגים. גישה טובה יותר תהיה לבנות מודל על בסיס חלק מהדאטה, לבחון את מידת הצלחה של המודל על יתר הדאטה, ולפי ההצלחה שלו במשימה זו לחת משקל ל-vote של המודל. ניתן להוציא תחכום לרענון זה, כך שבכל שלב ינתן יותר דגש איברים בדאטה שהמודלים הקודמים שגו בסיווג שלהם, ובכך בכל שלב בו מאמנים מודל נוסף תוך הzilla הצלחה יותר גדולה מאשר המודל המקורי. חלק זה הינו החלק האדפטיבי (Adaptive) באlgorigthm, 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 משלכות 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 אופטימי (x) c_t ביחס לאוסף הנתונים המשווקל.

чисוב שגיאות הסיווג של (x) c_t : $c_t(x) \neq y_i \Rightarrow \epsilon_t = \sum_i w_i^t \{c_t(x) \neq y_i\}$

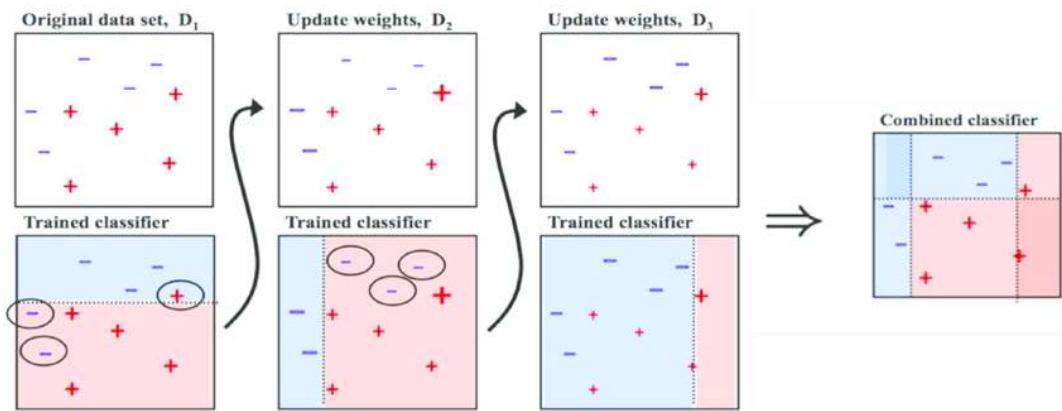
чисוב משקל עבור מסווג זה: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon}{\epsilon} \right)$.

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

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 המשווקל, שהינו קומבינציה לינארית של המסוגים החלשים:

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



איור 2.19 – 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 רגראסיה – 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 ברגראסיה לינארית (Linear regression) והחלק השני עוסוק ברגראסיה לוגיסטיית (Logistic regression). לבסוף יוצג מבנה שקול לביעות הרגראסיה 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 ק"מ. תחת הנחה זו, יש לבחון את המודל עבור דוגמאות ידועות ובכך למצוא את המשקלים וה-bias. לאחר מכן ניתן יהיה 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 למינימום תספק את הערכות האופטימליים של המשקלים וה-bias. פונקציית מחיר מקובלת הינה השגיאה הריבועית הממוצעת (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 לסכום 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 למינימום:

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

עבור המקירה הסקלרי בו $d = 1$, ככלומר יש מאפיין יחיד ומונוטם למצוא קשר בין פלט מסוים, הקשר הלינארי הוא $b + wx = \hat{y}$. 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 משווהות 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 הילינארי.

לשם הנוחות ניתן לסמן את ה-*bias* כפרמטר נוסף:

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

עבור המקרה הוקטוריש יש n דוגמאות, כלומר יש n מאפיינים בלתי תלויים ומנסים למצוא את הקשר ביןיהם 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 $\|Y - Xw\|^2$:

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

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

$$\hat{w} = (X^T X)^{-1} X^T Y$$

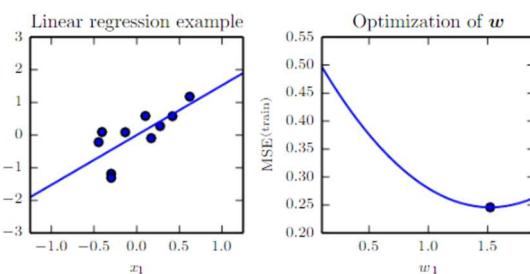
ובה ניתן אוסף דוגמאות:

$$X = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}, \quad \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

אזי הפתרון של הרגרסיה הילינארית 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

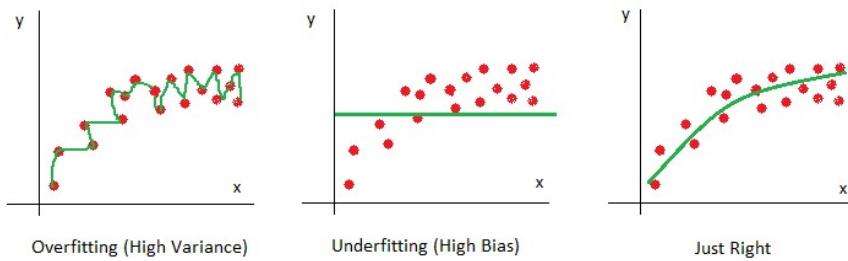
הרבה פעמים מציאת המינימום של פונקציית המחיר היא משימה קשה. דרך מקובלת להתמודד עם חישוב הפרמטר האופטימלי היא שיטת gradient descent (GD). בשיטה זו מתחילה מינוחש מסוים עבור הפרמטרים, וכל פעם מבצעים צעד לכיוון הגראדינט השילבי. הגראדינט הוא הנגזרת של הפונקציה, והוא מגדר את הכוון שערך הפונקציה עולה בו בצורה מקסימלית. אם לוקחים את הכוון השילבי של הגראדינט, בעצם הולכים לכיוון בו יש את הירידה הכי גדולה, וכך כל פעם ל הגיעו למינימום יש לבצע את הצעד בכיוון הגראדינט השילבי. כדי להימנע מהיקלעות לנקודת אוכף, מוסיפים איבר הנקרא (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 (העדכן 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 שתיה 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 בעית הרגסיה (ושאר בעית הלמידה) הוא לפתח מודל שייהה מוצלח לא רק עבור אוסף הדוגמאות הידוע (אימון), אלא שייהה 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 גבוה בעל שונות גדולה. במצב זה המודל מתאים רק לסט האימון, אך הוא לא מצליח להסביר גם דוגמאות חדשות. Overfitting – מודל שלא מצליח למצואן קוו מגמה המכיל מספיק מידע על הדוגמאות הנתונות, ויש לו רעש חזק.



איור 3.2 – נתינת משקל יתר לכל נקודה גורמת למצב בו המודל הוא מסדר גבוה ובעל שונות גבוהה (שמאל). – מודל בעל רעש חזק מייצג ביצורה מספיק טובה את המידע (אמצע). מצב מאוזן – מודל בעל שגיאה מינימלית, המתאים ביצורה טובה את המידע, ובנוסף מננו 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 ביצורה:

$$\text{Regularized Loss} = \text{Loss Function} + \text{Constraint}$$

יש מספר דרכים לבצע את ה-regularization:

Ridge Regression / L2 Regularization

דרך אחת לבצע את 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, ובכך 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 עשו למונע overfitting.

Lasso / L1 Regularization

דרך נוספת לבצע את ה-*on-hoing* היא ליחס 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 המחייר. בפועל אילוץ זה מביא ל"רידוד משקלים" (sparse), כלומר כופה חלק מהמקדים להיות אפס, וכך למעשה יש מעין feature selection – בחירת הפרמטרים המשמעותיים יותר.

ניתן לשים לב כי עבור L_2 השפעה של המשקלים על פונקציית המחייר היא ריבועית. לכן במקרה זה הרגולרייזציה תשאיר להקטין את הפרמטרים הגדולים, ובאופן כללי תנסה לדאוג לכך שככל הפרמטרים יהיו קטנים, ובאותו סדר גודל. 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, ובכך לנסות לכון את המודל עבור היתרונות של כל שיטה – גם להימנע מנתינית משקל יתר לפרמטרים וגם ניסיון לאפס פרמטרים, ובכך לקבל מודל פשוט ככל הנניתן. פונקציית המחייר 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 בלבד 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 להגיע מ-*fitting* (בדרכן כל לווקחים את המוצע 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% הנטרים.

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

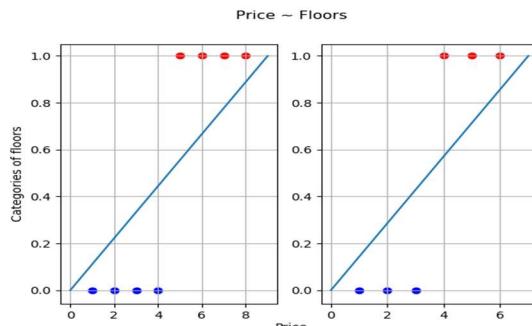
משימת סיווג מוגדרת באופן הבא: בהינתן סט פרמטרים מסוימים $\{x_1, \dots, x_n\}$ X השיר לתצפית מסוימת, יש לסתור אותו לאחת מתוך m קטגוריות אפשריות: $\{1, \dots, m\} \in y$. לדוגמה: נתונה תמונה בעלת n פיקסלים המייצגת חייה, ויש לקבוע איזו חייה היא, כאשר הבחירה נעשית מתוך הווקטור y . לרוב הווקטורים y מורכב 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 סיווג. עבור המקרה של $2 = m$, יש שתי קטגוריות אפשריות, ולמעשה יש צורך למפות כל נקודה לאחת משתי הקטגוריות. בעזרת רגסיה לינארית ניתן לבצע מיפוי מ- \mathbb{R} ל- $\{0,1\}$, כלומר כל נקודה במרחב ממופה לאחד משני ערכי אפסריים: קבועים ערך סף $T = 0.5$, ועבור נקודה חדשה x בזדוקים מה היחס בין הביטוי $b + w^T x_n < 0.5$ לבין ערך הסף. אם הנקודה החדשה מקיימת: $w^T x_n + b < 0.5$ 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 צ בתים ובער 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: $\{1 \text{ floor}, 2 \text{ floors}\} \in \{0,1\}$, ויש להיעזר במידע על צ הבתים כדי 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.



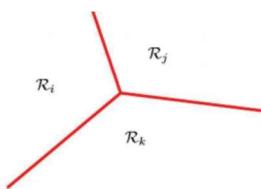
איור 3.4 רגסיה לינארית מסווג בינהר: מיפוי הנקודות בהתאם למיקומן ביחס לקו ההפרדה של הרגסיה הלינארית. בדוגמה הימנית ערך הסף מתkeletal עבור $x_T = 3.5$, כלומר $3.5 + b = 0.5 \cdot 3.5 + b = 0.5$. בדוגמה השاملית ערך הסף מתkeletal עבור $x_T = 4.5$. עבור כל בית חדש, בהינתן מחיר ניתן ישותו לאחת משתי הקטגוריות, בהתאם ליחסו לערך הסף 0.5.

עבור צ 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 במונה הבינהר, ואילו y מכיל צעת m קטגוריות: $\{m, \dots, 1, 0\} \in \mathcal{Y}$. במקרים אלו יש ליצור מספר קווים לינאריים, המפרידים בין אזורים שונים. כדי לחשב את הקווים מבצעים התהילה'ן שנקרא all versus one, בו בכל פעם לוקחים קטגוריה אחת ובזוקרים מהו קו ההפרדה בין לביון אחר הקטגוריות. הפרמטרים הנלמדים של קווי ההפרדה יהיו הסט המורכב מכל הפרמטרים של הרגסיה: $\{w_1, b_1, w_m, b_m\} = \theta$.



איור 3.5 רגסיה לינארית מרובה – הפרדה בין מספר אזורים שונים על ידי מספר קווים לינאריים.

במקרה זהה, נקודה חדשה תסואג 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 מהרגression ה binnari הינו "מסוג קשה" – כל דוגמא חדשה שמתקבלת מסווגת לקטגוריה מסוימת, ואין שום מידע עד כמה הדוגמא הזה דומה לקטגוריות האחרות. מסוג זה אינו מספיק טוב עבור מגוון בעיות, בהן מעוניינים לדעת לא רק את הקטgorיה, אלא גם מידע נוסף על היחס בין הדוגמא החדש לבין כל הקטגוריות. לדוגמה: בהינתן מידע של גידול מסוים רוצים לדעת אם הוא ממאייר או שפיר. במקרה זה ההכרעה היא לא תמיד חד משמעותית, ויש עניין לדעת מה הסיכוי של הגידול להיות ממאייר או שפיר, שהרי יתכן שהטיפול יהיה שונה בין מקרה בו יש 1% שהגידול הזה הוא מסווג בין מקרה בו יש 40% שהגידול הוא מסווג הזה. כדי להימנע מהסיווג הקטgorio, יש ליצור מודל הסתברותי, בו כל קטgorיה מקבלת הסתברות מסוימת. אחד המודלים הבסיסיים הינו רגرسיה לוגיסטיבית (Logistic regression). עבור המסוג הזה ראשית יש להגדיר את פונקציית הסיגמוואיד:

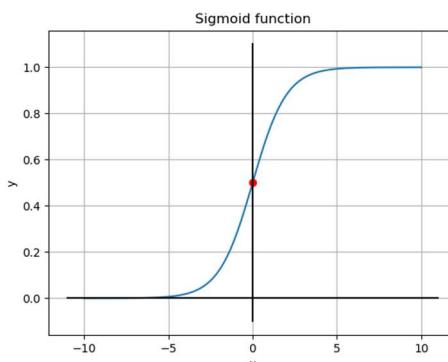
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

פונקציה זו רציפה על כל הישר, ובעצמותה ניתנת להגדיר מסווג עבור המקרה ה binnari:

$$p(y = 1|x; \theta) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

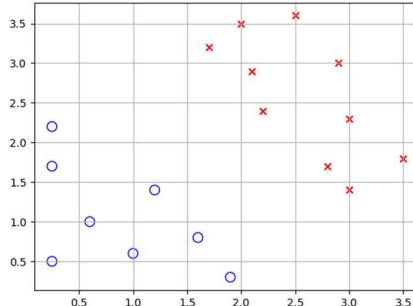
$$p(y = 0|x; \theta) = 1 - \sigma(w^T x + b) = 1 - \frac{1}{1 + e^{-(w^T x + b)}} = \frac{e^{-(w^T x + b)}}{1 + e^{-(w^T x + b)}}$$

המסוג לוקח את קוו החלטה ה binnari, ומעברו 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 בדיקן-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 נקודה זו להיות משוכנת לקטגוריה 1 גדולה מחצי, כיוון $w^T x_n + b > 0.5\sigma$. באופן סימטרי אם $w^T x_n + b < 0$, אז ההסתברות של נקודה x_n להיות משוכנת לקטגוריה 1 קטנה מחצי. עתול השאלה מתי $w^T x + b = 0$, והתשובה היא שזה תלוי בכך אם 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 , ומושיכות לאחת משתי קטגוריות: $y \in \{0,1\} = \{\text{blue o}, \text{red x}\}$.

כיוון שננתנות הנקודות, ניתן ליצור בעזרתן קו רגסצייה. לצורך הדוגמא נניח שיש שלושה פרמטרים והם מקיימים: $w^T = [1, 1, -3] = [w_1, w_2, b]$. הפרמטרים האלו מרכיבים את הקו הלינארי $3 = x_2 - x_1$, כלומר, עבור כל נקודה אם מתקיים $3 > x_2 - x_1$ היא תהיה מסווגת כ-' $\text{red x}'$, אחרת היא תהיה מסווגת כ-' o blue '. קו זה הוא למעשה חישוב הפרדה בין שתי נקודות חדשות. קו זה מקיים את המשוואה $0 = w^T x + b$, ולכן הוא יתחלק מ-50% של הנקודות בפערתנו. וכך נקבע את המשוואה $0 = w^T x + b$, המשמעות היא שנקודה זו נמצאת בבדיקה על קו הפרדה. נקודה צויה לקבל הסתברות שוגם מקיימת $0 = w^T x_n + b = 0$, המשמעות היא שנקודה זו נמצאת בבדיקה על קו הפרדה. ככל שהנקודה תתרחק מקו הפרדה, כך הביטוי $w^T x + b$ יתרחק מ-0, וכך גם $(w^T x + b)^2$ יתרחק מ-0, וכך נקבע את הקטגוריה אחורית ולא לאחרת.

כਮון 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 $[y=1|x], p(y=0|x) = \hat{y}$, והקטגוריה של \hat{y} תהיה שזו ה- \hat{y} בעל ההסתברות הגבוהה ביותר.

3.2.2 Cross Entropy and Gradient descent

בכדי למצוא את הפרמטרים $(b, w) = \theta$ האופטימליים בהינתן סט דוגמאות, ניתן להחליף את קרייטריון השגיאה הריבועית המומוצעת בקריטריון אחר למazure פונקציית המחיר – Cross entropy. קרייטריון זה אומר שיש לחייב לפחות מינימום את מינוס הלוג של סך הדוגמאות (הביטוי נבע 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 למינימום: $\hat{\theta} = \arg \min_{\theta} L(\theta)$.

בכדי לחשב את הביטוי יש לפתח קודם את הביטוי עבור נגזרת הסיגמודואיד:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \rightarrow \frac{\partial \sigma(z)}{\partial z} = \frac{-1}{(1 + e^{-z})^2} \cdot e^{-z} \cdot (-1) = \frac{1}{1 + e^{-z}} \cdot \frac{e^{-z}}{1 + e^{-z}} = \sigma(z)(1 - \sigma(z))$$

כך, $1 - \sigma(z) = \sigma(-z)$, כלומר $\sigma(-z) = 1 - \sigma(z)$, וכך 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 $L(\theta)$ 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

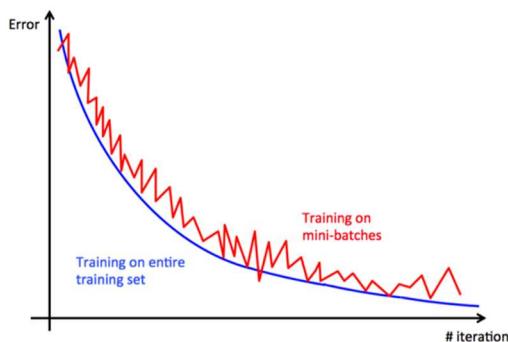
$$\hat{\theta}_{j+1} = \hat{\theta}_j - \epsilon \cdot \frac{\partial}{\partial \theta_j} L(\theta)$$

כאשר ϵ הוא הפרמטר של ה-rate learning. כיוון שפונקציית המחר $L(\theta)$ קעורה, מובהק שתיהה התוכנות ל- $\hat{\theta}$.

במקרים רבים הدادה סט הוא גדול, ולחשב את הגרדיינט עבור כל הدادה צריכה הרבה קלידום ניתן לחשב את הגרדיינט עבור חלק מהدادה, וביצוע את הקיימות לפי הכוון של הגרדיינט המתקבל. 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 לקחת 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:

$$\frac{\partial L}{\partial \theta} = \frac{\partial}{\partial \theta} \left[-\frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \log p(y_i|x_i; \theta) \right] \approx \frac{\partial}{\partial \theta} \left[-\frac{1}{n} \sum_{i=1}^n \log p(y_i|x_i; \theta) \right]$$

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 מתקובל משער חסר הטיה לגרדיינט האמיתי.



איור 3.8 השגיאה של $\hat{\theta}$ כפונקציה של האיטרציות בשיטת gradient descent בשיטת batch learning בכל צעד מחושב על כל הدادה, והgraf הכהול 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 , והמכנה מנורמל את התוצאה, כך שסך כל האיברים לאחר הפונקציה הוא 1. במקרה בו יש מספר קטגוריות – יש מספר קווים 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 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 הסימן $1_{\{i=s\}}$ הינו 1 אם $i = s$ ו-0 אחרת. כעת ניתן להציב את הביטוי האחרון בנגזרת של $L(\theta)$:

$$\frac{\partial L}{\partial \theta_i} = -\frac{1}{n} \sum_{t=1}^n (1_{\{y_t=k\}} - p(y_t = i|x_t; \theta)) x$$

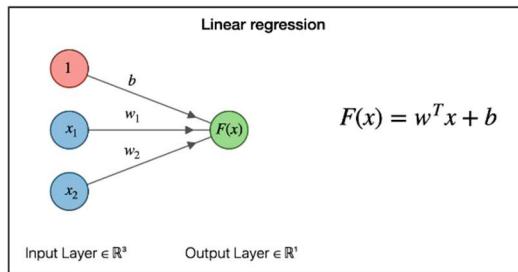
כעת ניתן לחשב את θ האופטימלי בשיטת gradient descent

$$\theta_{i+1} = \theta_i - \epsilon \frac{\partial L}{\partial \theta}$$

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 לקרה בו יש יותר משתי קטגוריות. עם זאת, יש לה חסרוןמשמעותי – קווי הפרדה של המודל הינם לינאריים, וזו הפרדה שאינה מספקת טובہ עבור בעיות מורכבות. יש מגוון בעיות בהן על מנת לבנות מודל המסוגל להפריד בין קטגוריות שונות, יש צורך במנגנון הפרדה לא לינארית.

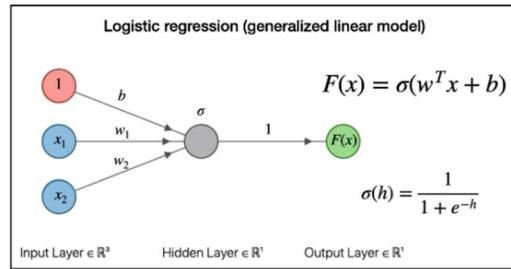
דרך מקובלת לבניית מודלים לא לינאריים היא שימוש ברשות נוירונים עמוקות, ובכך להבין את הקונספט של זה. ראייה יש לייצג את המודלים הלינאריים כשכבה של נוירונים, כאשר המודל הזה שקול לחלוטין לכל מה שהוא גוד כה. בעיית Linear regression לוקחת סט של מאפיינים ומכלילה כל אחד מהם במשקל, ולאחר מכן סוכמת את כל האלמנטים (בצירוף bias) לכדי משתנה יחיד הקובל מה הקטgorיה של סט זה. ניתן לייצג את המודל על ידי התיאור הגרפי הבא:



איור 3.9 ייצוג רגסיה לינארית כרשת נוירונים 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 באיר נקרא נוירון מלאכותי – אלמנט היכל לקבל קלט, לבצע פעולה חישובית ולהוציא קלט.

רגסיה לוגיסטיבית ניתנת לתיאור באופן דומה, כאשר הנוירונים של סט ה-*setpoint* לא מחוברים ישירות למצאה אלאüberים דרך סיגמוואיד במקורה הבינארי או דרך SoftMax במקורה בו יש יותר משתי קטגוריות:



איור 3.10 ייצוג רגסיה לוגיסטיבית כרשת נוירונים עם שכבה אחת.

מלבד המעבר לפונקציית הסיגמוואיד, יש הבדל נוסף בין הייצוג של הרגסיה הלינארית לייצוג של הרגסיה הלוגיסטיבית: בעוד הרגסיה הלינארית מספקת למצאה מספר יחיד במוחא (מושג קשה), הרגסיה הלוגיסטיבית מספקת למצאה וקטור באורך של מספר הקטגוריות, באופן צזה שלכל קטgorיה יש הסתברות מסוימת שה-*setpoint* שייר לאותה קטgorיה.

בפרק הבא יוצג מבנה בעל מספר שכבות של נוירונים, כאשר בין שכבה לשכבה יש פונקציה לא לינארית. באופן זה המודל שיתקיים יהיה מיפוי של סט מאפיינים באופן לא לינארי לוקטור הסתברויות למצאה. הגמישות 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/>

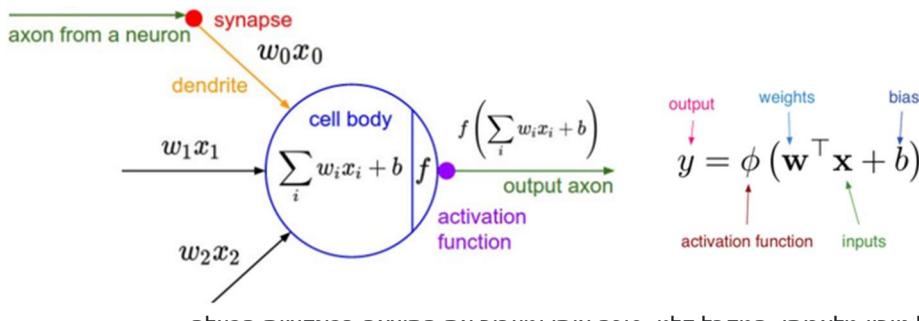
4. Deep Neural Networks

פרק זה עוסק ברשתות נירוניים عمוקות. רשת נירונים הינה חיבור של יחידות עיבוד בסיסיות (נירונים מלאכותיים) על ידי משקלים ופונקציות לא לינאריות. רשת נירונים נקראת עומקה אם היא מכילה יותר שכבה חבוייה אחת. לאחר הצגת הבסיסי הרעיוני והפורמלי, יסביר כיצד ניתן לחשב את המשקלים של הרשת בצורה ישרה בעזרת מבנה המוכנה Computational Graph. לאחר מכן יוצג שני תחומים העוסקים בשיפור הרשת – שיטות אופטימיזציה לתהילר הלמידה ושיטות לבחון עד כמה המודל המתאים אכן מוביל בצורה טובה את הדאטה עליו הוא מאומן.

4.1 Multilayer Perceptron (MLP)

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 כך:

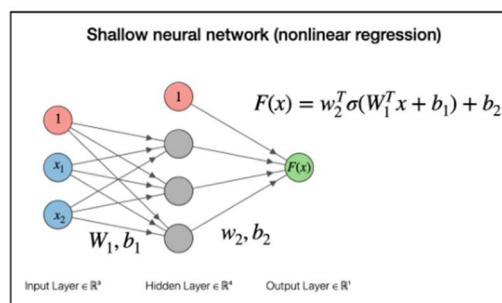


איור 4.1 יציג של נירון מלאכותי, המתקבל קלות, סוכם אותו וublisher את התוצאה בפונקציית הפעלה.

הקלט של הנירון הוא סט input מוכפל במשקלים: $x^T \mathbf{w} \in \mathbb{R}^d$ כאשר $\mathbf{w} \in \mathbb{R}^d$, $x \in \mathbb{R}^d$, ואיבר דרכ סוכם, ומתקבל הביטוי $b + \sum_{i=1}^d w_i x_i$. לאחר מכן הסכם עובר דרך פונקציית הפעלה, ומתקיים המוצא ($f(\sum_{i=1}^d w_i x_i + b)$). במקרה הפרטי בו פונקציית ההפעלה היא סיגמויד/SoftMax והמושך לא מחובר לשכבה נוספת, אז למעשה מקבלים את הרגסית הלוגיסטיות.

במקרה בו הנירונים המוחברים ל-*שכבת* אינם מהווים את המושך אלא הם מוכפלים במשקלים ומחברים לשכבה נוספת נירונית, אז השכבה המוחברת ל-*שכחת* נקראת שכבה חבוייה (hidden layer). אם יש יותר שכבה חבוייה אחת, הרשת מכונה רשת נירונים عمוקה. במקרה בו יש לפחות שכבה חבוייה אחת, הקשר בין הכניסה למושך אינו לינאר, וזה הינו שיפר מודול זה. נתבונן במקרה של שכבה חבוייה וnochesh את הקשר בין הכניסה למושך: נסמן את המשקלים בין הכניסה לבין השכבה החבוייה ב- w_1, b_1 ואת המשקלים בין השכבה החבוייה לבין המושך ב- w_2, b_2 , וכן את השכבה החבוייה מתתקבל הביטוי: $y = f_1(w_1^T x + b_1) + b_2$. ביטוי זה עובר בפונקציית הפעלה נוספת ומתקבל המושך:

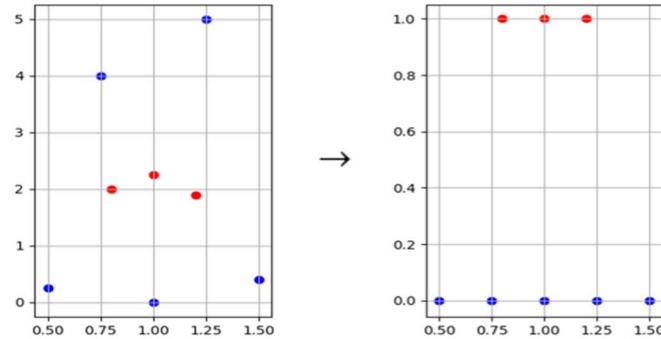
$$\hat{y} = f_2(w_2 \cdot f_1(w_1^T x + b_1) + b_2)$$



איור 4.2 רשת נירונים בעלת שכבה חבוייה אחת.

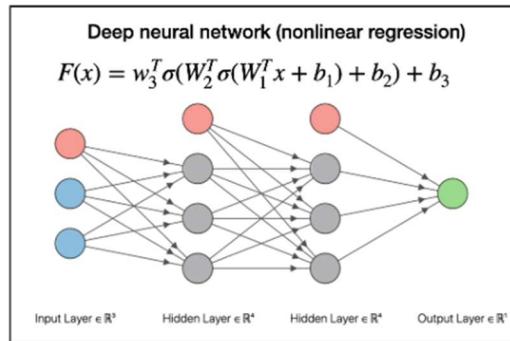
חשוב להזכיר שמטרת הרשת היא לבצע פעולות לא לינאריות על ה-*שכחת* כך שהיא יסודר באופן חדש הניתן להפרדה לינארית. למעשה מבדילים את הפרדה הלינארית הנעשית באמצעות הרגסית, אלא מבצעים פניה שלב מקדים של העתקה לא לינארית. תהילר זה נקרא למדידת "יצוגים" (representation learning), כאשר בכל שכבה

מנסם ללמידה יציג פשוט יותר לדאטה על מנת שהוא יוכל להיות מופרד באופן לינארי. המיקוד של הרשות הוא אינו במשימת סיווג אלא במשימת יציג, 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 להבחן בין הדוגמאות באמצעות קו הפרדה לינארי.

כאשר מחברים יותר משכבה 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 גדולה.

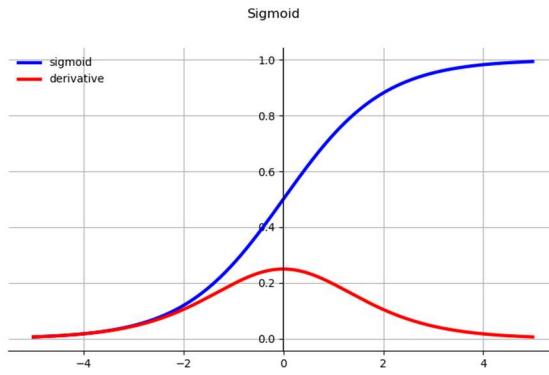
4.1.2 Activation Function

האלמנט המרכזי בכל ניירון הוא פונקציית הפעלה, ההופכת אותו ליחידת עיבוד לא לינארית. יש מספר פונקציות הפעלה מקובלות – Sigmoid, tanh, ReLU –

Sigmoid

פונקציית הסיגמודайд 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.

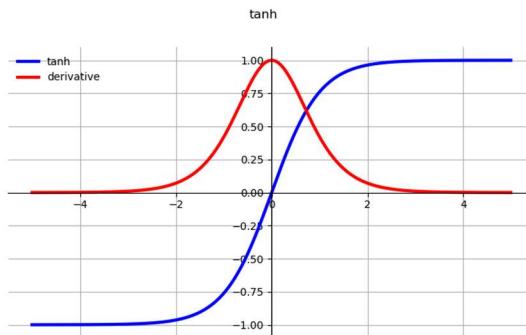
יש לפונקציה זו שלושה חסרונות:

- עבור ערכים גדולים, הנגזרת שואפת ל-0. זה כמובן יוצר בעיה בחישוב הפרמטר האופטימלי בשיטת Gradient descent, שהרי בכל צעד התוספת תלויות בגרדיינט, ואם הוא מתאפס – לא ניתן לחשב את הפרמטר האופטימלי.
- הסיגמואיד לא ממורכז סביב ה-0, וזה יוצאה בעיה עבור דאטה שאינו מנוון.
- הן הפונקציה והן הנגזרת דורשות חישוב של אקספוננט, ובאופן יחס' זו פעולה יקרה לחישוב.

tanh

פונקציית טנגנס היפרבולי הינה מהצורה:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad \frac{\partial}{\partial z} \tanh(z) = 1 - (\tanh(z))^2$$



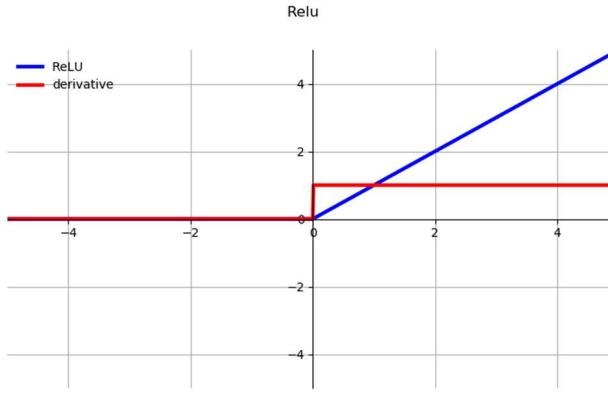
איור 4.6 פונקציית טנגנס היפרבולי והנגזרת שלה.

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 מארפסת ערכים שליליים ואディשה כלפי ערכים חיוביים. הפונקציה מחזירה את המקסימום מבין המספר שהוא מקבלת ובין 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 עיליה 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 מהר מהפונקציות הקודמות ($6x$). לפונקציה יש שני חסרונות עיקריים: היא לא ממורצת סביבה 0, ועבור אתחול משקלים לא טוב מרבית הנוירונים מתאפסים וזה יחסית בזבזני. כדי להתגבר על הבעיה האחורונה ניתן להשתמש בורותיות של הפונקציה, כמו למשל ELU ו-PReLU:

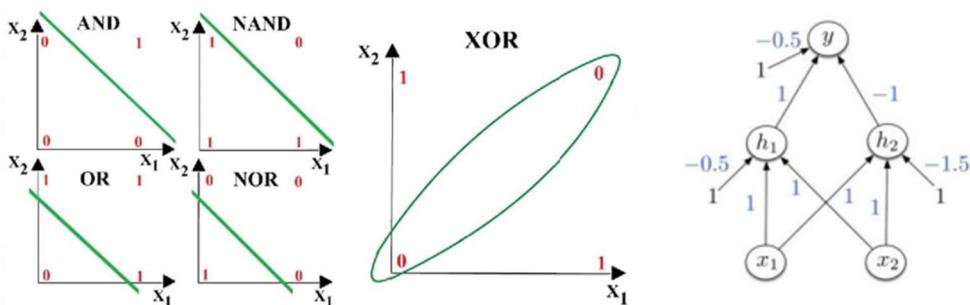
$$PReLU(x) = \max(ax, x), ELU(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}$$

בפונקציית PReLU, המקירה הפרטி בו $\alpha = 0.01$ 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 לפחות ליציאה, כאשר יש שתי קטגוריות במצוא, ואין אפשרות להעביר 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 שקלול לאופרטור XOR.

בדוגמא המובאת באյור הכניסות עוברות דרך שכבה חבויה אחת בעלת שני 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 y (יחד עם ה-bias) יתקבל מספר שלילי, ובמוצא יתקבל 0. אם הכניסה השנייה היא 0, אז $1 = sign(0.5) = y$. במצב בו שתי ה כניסות הן 0, יתקיים $0 = h_2 = h_1$, וזה 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 יציג פשוט יותר של המידע שנכנס אליה, כך שבסופו של דבר ניתן היה להבחין בין קטגוריות שונות בעזרת הפרדה ביןארית. מה שקובע את השינוי של הדadata במעבר לשולן בראשת הממשקלים 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.

התהילך האימון מתבצע בשני שלבים – ראשית מכנים דוגמא ידועה לתחילת הרשות ו"מפעעים" אותה עד למוצא (Forward propagation), כמובן, מחשבים את השינוי שהיא עוברת כאשר היא מוכפלת במסקלים וועוררת בנוירונים החבויים. לאחר שמגיעים למוצא, משווים את מה שהתקבל למה שאמור להיות במצבו לפי מה שידוע על דוגמא זו, ואז מבצעים פעוף לאחר (Backward propagation), שמטרתו לתקן את המשקלים בהתאם למה שהתקבל במצבו. השלב השני הוא למשה חישוב עיל של GD על פני כל שכבות הרשות – מחשבים את הנגזרת בין המשקל w_i לבין פונקציית המחיר ($\theta(L)$, ואז מבצעים עדכון בשיטת $GD - \frac{\partial \theta}{\partial w_i} \epsilon - w_i = w_{i+1}$. כיון שהרשות יכולה להכיל מיליארדי משקלים, יש למצאו דרך יעילה לחישוב הגרדיאנט עבור כל משקל.

נحو לעשות את התהילך הדו-שלבי הזה בעזרת Computational Graphs, שזהו למעשה גרף הבניי מצמתים המיצגים את התהילך שהדadata עובר בתוך הרשות. הגרף יכול ליצג כל רשות, וכן ניתן באמצעותו לחשב נגזרות מורכבות באופן פשוט יחסית. לאחר השלב הראשון בו מעבירים דוגמא בכל חלקו הgraf, ניתן למשל לחשב את השגיאה הריבועית הממוצעת ($y - \hat{y}$), להגדיר אותה כפונקציית המחיר, ולמצוא את הנגזרת של כל משקל לפי פונקציה זו – $\frac{\partial \theta}{\partial w_i}$, כאשר הנגזרות החלקיות מחושבות בעזרת כלל השרשרת.

4.2.2 Forward and Backward propagation

באופן פורמלי, עבור N משקלים התהילך מנוט כה:

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:

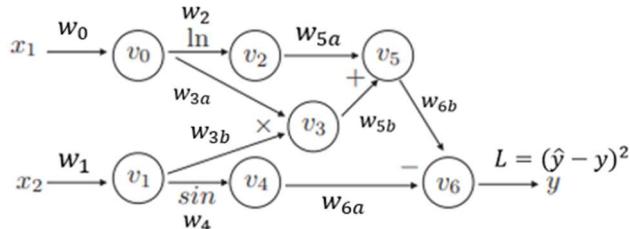
$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial w_N} \cdot \frac{\partial w}{\partial w_{N-1}} \dots \frac{\partial w_{i+1}}{\partial w_i}$$

$$\overline{w_i} = w_i - \epsilon \frac{\partial L}{\partial w_i}$$

בשלב הראשון מחשבים כל צומת על סמך הצמתים הקודמים 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 מודול נפוצה ברשומות עמוקות עוקב ייעולותה בחישוב המשקלים. בשונה מבועית גרסיה, חישוב האופטימום ברשומות עמוקות היא לא בעיה קמורה, ולכן לא תמיד יש לה בהכרח מינימום גלובלי.

עם זאת, עדכון המשקלים בשיטת Back propagation הוכיח את עצמו, למרות שהמשקלים לא בהכרח הגיעו לאופטימום שלהם.

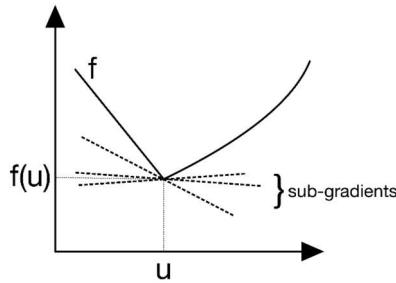
4.2.3 Back Propagation and Stochastic Gradient Descent

כיוון שהנושא של backpropagation הוא מאוד בסיסי בראשות נירונים, נרحب עליו את הדיבור וنبסס את העקרונות המתמטיים שלו בצורה יותר עמוקה.

הקדמה – סאב גרדיאנט: נאמר ש- $V \in g$ הוא סאב-גרדיינט של פונקציה $\mathbb{R} \rightarrow V$: אם לכל $V \in \mathcal{U}$ מתקיים:

$$f(v) \geq f(u) + \langle g, v - u \rangle$$

קובוצת כל הסאב-גרדיינטים של f בנקודה u מסומנת ב- $\{\partial f\}$. באופן גאומטרי, $\{u\}$ היא קבוצת כל הישרים שנמצאים מתחת לגרף f בנקודה u . בפרט, עבור פונקציה f קמורה וגדירה בנקודה u ישנו רק ישר אחד מתחת לגרף – זהו השר המשיק ל- f , דהיינו $\{\nabla f\}(u) = \{\partial f\}$.

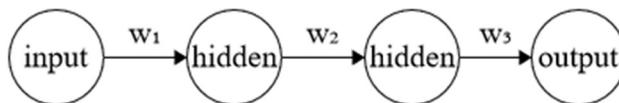


הסאב-גרדיינט משמש כהכללה למושג הגרדיינט כאשר אנו עוסקים בפונקציות שאין בהכרח גזירות, ולעתים שימושי גם באיטרציה של (SGD) Stochastic Gradient Descent כאשר עוסקים בפונקציה שאינה גדרה אנליטית. את הסאב-גרדיינט של פונקציית מחיר l התלויה במשקלות w ומשועך בנקודה (y, x) נסמן ב- $\{\partial l(w, x)\}$.

כדוגמה קצרה לפני שנמשיך, ניקח למשל את פונקציית הערך המוחלט: $|v| = (v)_+$. לכל $0 > \mathcal{U}$ הסאב-גרדיינט $\{+1\} = (0)_+$ וככל' עברו הנקודות שמקיימות $0 < v$, עברו $\{-1\} = (-v)_+$. בנקודה $0 = \mathcal{U}$ הפונקציה אינה גדרה, אך נוכל לרשום מפורשות ש- g הוא סאב-גרדיינט של f אם מתקיים $g \geq |u|$, מה שכן רק אם $[-1, 1] \in g$. משום כך נסיק $[1, -1] \in (0)_+$.

אלגוריתם backpropagation: נוהג לסמן איטרציה SGD בצורתה הכללית על ידי $\theta^{(t+1)} = \theta^{(t)} - \eta_{n+1} g^{(t)}$ כאשר g הוא סאב-גרדיינט של פונקציית הלוס l , דהיינו (x_t, y_t) $\in \mathcal{U}^{(t)}$ למען הפרשנות אנו נניח כרגע ש $\nabla l = g^{(t)}$ (כלומר, אנו נניח שלפונקציית הלוס יש גרדיינט) ונתעמק בדרך החישוב המהירה של הביטוי הנ'ל, הידועה בתור אלגוריתם backpropagation.

המסגרת הכללית שלנו היא רשת נירונים בסיסית (MLP) ופונקציית מחיר l (כלומר ריבוע ההפרש בין הפלט של הרשת לבין הפלט האמתי). הצעד הראשון בתהליך יהיה להתחיל מרשת בסיסית עם שכבה קלה בעלת נירון יחיד, 2 שכבות נסתרות המכילות כל אחת נירון יחיד, ושבכבה פلت המכילה גם היא נירון בודד. לכל נירון (פרט לנירון הקלה) יש גם bias (אליה הם b_i), וכל זוג נירונים מחוברים באמצעות משקל (אליה הם w_{ij}), כך שבסך הכל, פונקציית המחיר שלנו L היא פונקציה של המשתנים הבאים: $L(w_1, b_1, w_2, b_2, w_3, b_3) = L$. כאמור, נמצא של כל נירון יש פונקציית אקטיבציה (בדרכ כליל לא לינארית), והמטרה הכללית היא למצוא את ערכי b , w שambilאים את השגיאה של L למינימום.



איור 4.10 רשת נירונים עם קלט יחיד, שתי שכבות חביות בעלות נירון בודד בכל אחת מהן, ומוצא בעל נירון יחיד. לכל אחד מהנירונים פרט לכינסה יש גם bias.

נתמך בקשר בין 2 הנירונים האחרונים, כאשר נסמן את האקטיבציה של הנירון בrama i - $a^{(i)}$. השגיאה של הרשת בנקודה ה-0 בדата (למשל התמונה הראשונה מתוך 50,000) היא:

$$L_0(\dots) = (a^{(o)} - y)^2, (o \text{ is output})$$

כאשר נוכל לסמן את המוצא של האקטיבציה בrama i באמצעות האקטיבציה של הרמה הקודמת והערכים של הנירון הנוכחי:

$$a^{(i)} = \sigma(w^{(i)}a^{(i-1)} + b^{(i)}) = \sigma(z^{(L)})$$

המטרה כעת היא להבין כמה פונקציית המחיר משתנה ביחס למשקל, כדי שנוכל לבדוק את המשקלים בהתאם:

$$\frac{\partial L_0}{\partial w^{(o)}} = \frac{\partial z^{(o)}}{\partial w^{(o)}} \cdot \frac{\partial a^{(o)}}{\partial z^{(o)}} \cdot \frac{\partial L_0}{\partial a^{(o)}}$$

נחשב את הביטוי באופן מפורש:

$$L = (a^{(o)} - y)^2 \rightarrow \frac{\partial L}{\partial a^{(o)}} = 2(a^{(o)} - y)$$

$$a^{(o)} = \sigma(z^{(o)}) \rightarrow \frac{\partial a^{(o)}}{\partial z^{(o)}} = \sigma'(z^{(o)})$$

$$z^{(o)} = w^{(o)}a^{(o-1)} + b^{(o)} \rightarrow \frac{\partial z^{(o)}}{\partial w^{(o)}} = a^{(o-1)}$$

ולכן מקבל:

$$\frac{\partial L}{\partial w^{(o)}} = a^{(o-1)}\sigma'(z^{(o)})2(a^{(o)} - y)$$

עבור כל הדאטה שלנו, אנו לוקחים ממוצע משוקל:

$$\frac{\partial L}{\partial w^o} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial L_k}{\partial w^{(o)}}$$

הביטוי הזה הוא רק אחד מרכיבי הגרדיאנט אותו אנו רוצים לחשב (במודגש):

$$\nabla L = \left(\frac{\partial L}{\partial w^{(0)}}, \frac{\partial L}{\partial b^{(0)}}, \dots, \frac{\partial L}{\partial b^{(o)}}, \dots, \frac{\partial L}{\partial b^{(n)}} \right)^T$$

באופן דומה נוכל לרשום את הגרדיאנט של איבר bias:

$$\frac{\partial L_0}{\partial b^o} = \frac{\partial z^{(o)}}{\partial b^{(o)}} \cdot \frac{\partial a^{(o)}}{\partial z^{(o)}} \cdot \frac{\partial L_0}{\partial a^{(o)}}$$

רק האיבר הראשון במכפלה (הנגזרת של z לפि b משתנה, והיתר נשארים זהים):

$$z^{(o)} = w^{(o)}a^{(o-1)} + b^{(o)} \rightarrow \frac{\partial z^{(o)}}{\partial b^{(o)}} = 1$$

ולכן מקבל:

$$\frac{\partial L_0}{\partial b^{(o)}} = 1 \cdot \sigma'(z^{(o)}) \cdot 2(a^{(o)} - y)$$

וממוצע על פני כל הדאטה:

$$\frac{\partial L}{\partial b^{(o)}} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial L_k}{\partial b^{(o)}}$$

כל הפיתוח שעשינו מתייחס רק לנירון של הפלט, וחישבנו כיצד פונקציית המחריר מושפעת משינויים בקלטים של נירון זה. נרצה להציג את הביטויים גם עבור יתר השכבות ברשת. נתבונן למשל על שכבה אחת לפני אחורונה:

$$\frac{\partial L_0}{\partial w^{(o-1)}} = \frac{\partial z^{(o-1)}}{\partial w^{(o-1)}} \cdot \frac{\partial a^{(o-1)}}{\partial z^{(o-1)}} \cdot \frac{\partial L_0}{\partial a^{(o-1)}}$$

שני האיברים הראשונים ניתנים לחישוב באופן מפורש:

$$\frac{\partial z^{(o-1)}}{\partial w^{(o-1)}} = a^{(o-2)}$$

$$\frac{\partial a^{(o-1)}}{\partial z^{(o-1)}} = \sigma'(z^{(o-1)})$$

האיבר האחרון במכפלה ניתן לחישוב באמצעות כלל השרשרת:

$$\frac{\partial L_0}{\partial a^{(o-1)}} = \frac{\partial z^{(o)}}{\partial a^{(o-1)}} \cdot \frac{\partial a^{(o)}}{\partial z^{(o)}} \cdot \frac{\partial L_0}{\partial a^{(o)}} = w^{(o)} \cdot \sigma'(z^{(o)}) \cdot 2(a^{(o)} - y)$$

פרט לביטוי $\frac{\partial z^{(o)}}{\partial a^{(o-1)}}$, את היתר חישבנו בשלבים קודמים, וכיוון שביטוי זה שווה בדיק ל- $L^{(o)}$, נוכל לרשום:

$$\frac{\partial L_0}{\partial w^{(o-1)}} = a^{(o-2)} \cdot \sigma'(z^{(o-1)}) \cdot w^{(o)} \cdot \sigma'(z^{(o)}) \cdot 2(a^{(o)} - y)$$

אם ננסח זאת במילים – לצורך החישוב של $\frac{\partial L_0}{\partial a^{(o-1)}}$ הינו צריכים לדעת את $\frac{\partial L_0}{\partial w^{(o-1)}}$, ואף הוא נתון לנו על ידי החישובים שביצעו באיטרציה הקודמת.

עד כה התייחסנו לרשת נירונים בה בכל שכבה יש נירון אחד.icut גראטיב את הדיוון גם למקדים בהם יש שכבות אם יותר מניירון אחד. מלבד האינדקס העליון שיש לכל איבר (המייצג את השכבה), נוסף לכל איבר עוד משתנה (sub script) שמייצג את מספר הנירון באותה שכבה. הביטוי של L_0 יחשב דומה, אלא שכעת יש ללקחת בחשבון את כל הנירונים בrama האחורונה (נניח שיש n_o נירונים):

$$L_0 = \sum_{j=0}^{n_o-1} \left(a_j^{(o)} - y_j \right)^2$$

icut נסמן את המשקל בין $a_j^{(o)}$ ו- $a_k^{(o-1)}$ ב- $w_{jk}^{(L)}$. בהתאם, כל אקטיבציה תהיה מוגדרת כך:

$$a_j^{(o)} = \sigma \left(w_{j,0}^{(o)} a_0^{(o-1)} + \dots + w_{j,n_o-1}^{(o-1)} a_{n_o-1}^{(o-1)} + b_j^{(o)} \right) = \sigma \left(z_j^{(o)} \right)$$

נשים לב שהמשקלים מייצגים את כל האלומות בין הנירון $-j$ בrama $-o$ לבין כל הנירונים שברמה הקודמת – $(o-1)$:

$$z_j^{(o)} = \sum_{k=0}^{n_{o-1}-1} w_{jk}^{(o)} a_k^{(o-1)} + b_j^{(o)}$$

בשלב זה כלל השרשרת יראה כך:

$$\frac{\partial L_0}{\partial w_{jk}^{(o)}} = \frac{\partial z_j^{(o)}}{\partial w_{jk}^{(o)}} \cdot \frac{\partial a_j^{(o)}}{\partial z_j^{(o)}} \cdot \frac{\partial L_0}{\partial a_j^{(o)}}$$

כאשר:

$$\frac{\partial z_j^{(o)}}{\partial w_{jk}^{(o)}} = a_k^{(o-1)}, \quad \frac{\partial a_j^{(o)}}{\partial z_j^{(o)}} = \sigma'(z_j^{(o)}), \quad \frac{\partial L_0}{\partial a_j^{(o)}} = 2 \cdot (a_j^{(o)} - y_j)$$

לכן בסך הכל נקבל שעבור דוגמה בודדת, הנגזרת של פונקציית המחיר ביחס למשקלים ברמה o הינה:

$$\frac{\partial L_0}{\partial w_{jk}^{(o)}} = a_k^{(o-1)} \cdot \sigma'(z_j^{(o)}) \cdot 2(a_j^{(o)} - y_j)$$

וכאשר ממצאים את הנגזרות עבור n דוגמאות:

$$\frac{\partial L}{\partial w_{jk}^{(o)}} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{\partial L_i}{\partial w_{jk}^{(o)}}$$

ועבור $bias$:

$$\begin{aligned} \frac{\partial L_0}{\partial b_j^{(o)}} &= \frac{\partial z_j^{(o)}}{\partial b_j^{(o)}} \cdot \frac{\partial a_j^{(o)}}{\partial z_j^{(o)}} \cdot \frac{\partial L_0}{\partial a_j^{(o)}} = 1 \cdot \sigma'(z_j^{(o)}) \cdot 2(a_j^{(o)} - y_j) \\ \frac{\partial L}{\partial b_j^{(o)}} &= \frac{1}{n} \sum_{i=0}^{n-1} \frac{\partial L_i}{\partial b_j^{(o)}} \end{aligned}$$

זכור, אינו בחלק הקודם (בדיוון על רשות בסיסית) שלצורך החישוב של $\frac{\partial L_0}{\partial a^{(o-1)}}$ הינו צריכים לדעת את $\frac{\partial L_0}{\partial a^{(o-1)}}$, אך ביטוי זה היה נתון לנו מחישובים שביצענו באיטרציות קודומות. למעשה מה שהתקבל כאן הוא שהשינוי של פונקציית המחיר כפונקציה של משקלי הרשות תלויה בשינוי של פונקציית המחיר כפונקציה של האקטיבציות של השכבות הקודמות, ביטוי אשר חישבנו מפורשות בכל שלב. אם כן, יש לנו צורך מובהק בחישוב הקשר $\frac{\partial L_0}{\partial a_k^{(o-1)}}$ ולשם כך נרשום:

$$\frac{\partial C_0}{\partial a_k^{(o-1)}} = \sum_{j=0}^{n_{L-1}} \frac{\partial z_j^{(o)}}{\partial a_k^{(o-1)}} \cdot \frac{\partial a_j^{(o)}}{\partial z_j^{(o)}} \cdot \frac{\partial L_o}{\partial a_j^{(o)}}$$

שהרי בשונה מהמקרה בו יש רק קדוק אחד, התלות של L באקטיבציה של $a_k^{(o-1)}$ היא ביטוי של כל הנוירונים המחברים אליה בשכבה הבאה, ולא רק לאחד.

נסכם את הכל באלגוריתם:

לעדכן המשקל של השכבה $-l$:

$$\frac{\partial L_0}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \cdot \sigma'(z_j^{(l)}) \cdot \frac{\partial L_0}{\partial a_j^{(l)}}, \quad average \rightarrow \frac{\partial C}{\partial w_{jk}^{(l)}} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{\partial L_i}{\partial w_{jk}^{(l)}}$$

לעדכן bias של השכבה $-l$:

$$\frac{\partial L_0}{\partial b_j^{(l)}} = \sigma'(z_j^{(l)}) \frac{\partial L_0}{\partial a_j^{(l)}}, \quad average \rightarrow \frac{\partial L}{\partial b_j^{(l)}} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{\partial L_i}{\partial b_j^{(l)}}$$

זאת כאשר:

$$\frac{\partial L_0}{\partial a_j^{(l)}} = \begin{cases} \sum_{j=0}^{n_{l+1}-1} w_{jk}^{l+1} \cdot \sigma'(z_j^{l+1}) \cdot \frac{\partial L_0}{\partial a_j^{(l+1)}}, & l < L \\ 2(a_j^{(L)} - y_j), & l = L \end{cases}$$

הפלט בסופו של דבר הוא הווקטור שכניסותיו הם

$$\frac{\partial L}{\partial w_{jk}^{(l)}} = \frac{1}{n} \sum_{i=1}^{n-1} \frac{\partial L_i}{\partial w_{jk}^{(l)}}$$

$$\frac{\partial L}{\partial b_j^{(l)}} = \frac{1}{n} \sum_{i=1}^{n-1} \frac{\partial L_i}{\partial b_j^{(l)}}$$

4.3 Optimization

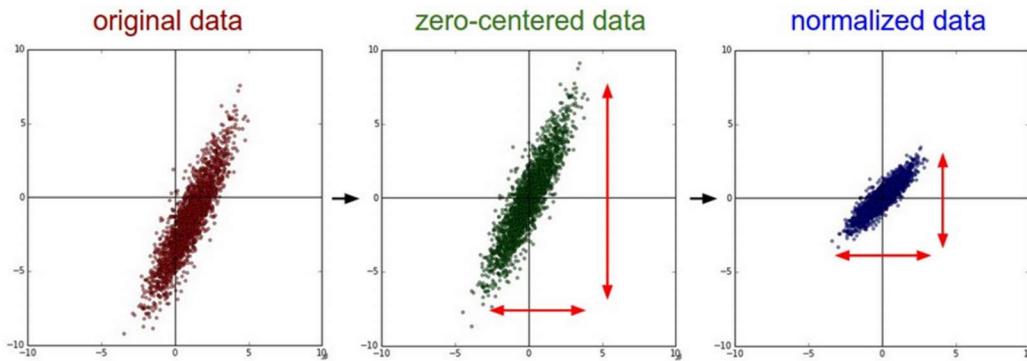
מציאת אופטימום למשקלים על פני כל העומק של הרשת היא בעיה לא קמורה, ולכן אין לה בהכרח מינימום גלובלי. לכן מלבד עדכון המשקלים בשיטת backpropagation יש לבצע אופטימיזציות נוספת על הרשת על מנת לשפר את הביצועים שלה.

4.3.1 Data Normalization

חלק מפונקציות הפעלה אינן ממורכבות סביב ה-0, ועבור ערכי גובהים הנ' קבועות בקרוב ולכן גרדיאנט בערכים אלו מתאפס, דבר שאינו אפשר לעדכן את המשקלים בשיטת GD. כדי להימנע מהגעה לתוךם ה"רווייה" בו הגרדיאנט מתאפס, ניתן לנורמל את הדadata כך שהיא בעל תוחלת 0 ושונות 1, ובכך הוא יהיה ממורכב סביב ה-0:

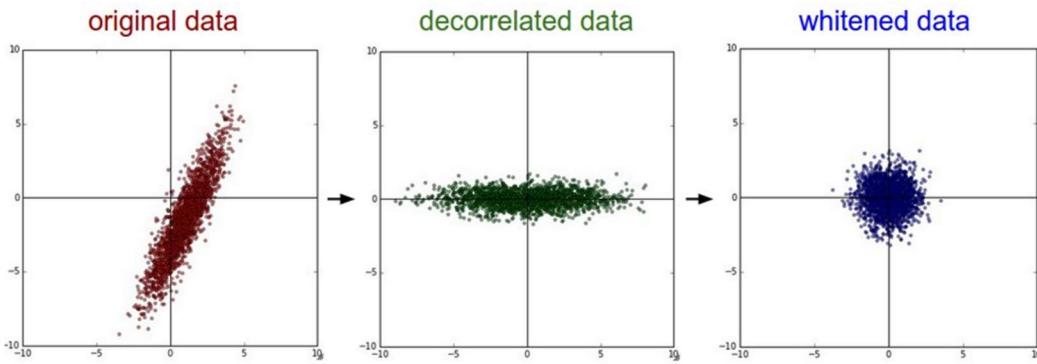
$$X_i = \frac{X_i - \mu_i}{\sigma_i}$$

ובאופן חזותי:



איור 4.11 נרמול דadata בשני שלבים – איפוס ההזולות (ירוק) ונרמול השונות ל-1 (כחול).

שלב זה הוא למעשה שלב pre-processing הנועד להכין את הדadata לפניה כניסה לרשת, כדי לשפר את אימונו הרשת. ישנו אופנים נוספים לנרמול את הדadata – ללקסן את מטריצת Covariance של הדadata או להפוך אותה למטריצת היחידה:



איור 4.12 דרכי נספנות לנורמל את הדטה – ללקסן את מטריצת ה covariance (ירוק) או להפוך אותה למטריצת היחידה (כחול).

4.3.2 Weight Initialization

ענין נוספת שיכל להשפיע על האימון ונitin להתייחס אליו עוד בשלב ה pre-processing הוא אתחול המשקלים. 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, להगביר אותן מהתפלגות מסויימת שתאפשר אימון טוב של הרשת.

אפשרות אחרת לאתחול היא להגביר עבור כל משקל ערך קטן מהתפלגות נורמלית עם שונות קטנה – $(\alpha, 0)(N)$, כאשר $\alpha = 0.01 \text{ or } 0.1$. אתחול באופן זה עובד טוב לרשותות קטנות יחסית, אך ברשותות עם הרבה שכבות אתחול בערכים קטנים גורם לאיפוס הגרדיאנט מהר מאד. כדי להתמודד עם בעיה זו, ניתן לבחור $\alpha = 1$, אך זה יכול לגרום להתבדרות הגרדיאנט. שיטה נוספת יותר נקראת Xavier Initialization, הלוקחת בחשבון את הגודל של השכבות – האתחול יבוצע בעזרת התפלגות נורמלית, אך השונות לא תהיה מספר ללא משמעות, אלא תהיה תלולה במספר השכבות – $\frac{1}{\sqrt{n}} = \alpha$. שיטה זו טובה גם לרשותות עם הרבה שכבות, אך היא בעייתית במקרים בו פונקציית הפעולה הינה U , כיוון שהאתחול מניח שפונקציית הפעולה ממורכצת סיבוב 0 (כמו למשל \tanh). כדי לאפשר גמישות גם מבחינת פונקציית הפעולה, ניתן לבחור $\sqrt{\frac{2}{n}} = \alpha$, ואז האתחול יתאים גם ל- $ReLU$.

אפשרות נוספת לאתחול הפרמטרים היא להגביר מהתפלגות אחידה, כאשר באופן דומה ל- Xavier-Initialization, גם כאן הגבולות יהיו תלויים בגודל השכבות – $[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$.

4.3.3 Batch Normalization

כאשר מביצעים normalization Data, למעשה דואגים לכך שכניסה לרשת הדטה יהיה מנורמל סביב ה-0. באופן זה נמנעים מהגעה למצב בו יש ערכים גבוהים בעומק הרשת, הגורמים להתאפסות או להתבדרות של הגרדיאנט. בפועל, הנרמול הזה לא תמיד מספיק טוב עבור כל השכבות, ואחרי כמה שכבות של הכפלה במשקלים ומעבר בפונקציות הפעלה הרבה פעמים מתקיים ערכיהם גבוהים. באופן דומה ל- $ReLU$ normalization הפונקציית המתבצעת לפני האימון, ניתן תוך כדי האימון לבצע Batch normalization שודאג לנרמול הערכים שנכנסים לנירונים בשכבות החבויות. התהליך נעשה בשלושה שלבים:

- עבור כל נירון בעל פונקציית הפעלה לא לינארית, מחשבים את התוחלת והשונות של כל הערכים היוצאים ממנו.
- מנורמלים את כל היציאות – מחסרים מכל יציאה את התוחלת ומחלקים את התוצאה בשונות (בתוספת אפסילון, כדי להימנע מחלוקת ב-0).
- הנורמל יכול לגרום לאיבוד מידע, שכן מביצעים לתוצאה המנורמלת scale and shift – הגדילה ושינוי קינה המידה. התיקון מתבצע בעזרת פרמטרים נלמדים.

עבור שכבות גדולות חישוב התוחלת והשונות יקר כיוון שלנירון יש הרבה יציאות, אך לוקחים רק חלק מהיציאות – Mini Batch: $\mathcal{B} = \{x_1 \dots x_m\}$

באופן פורמלי ניתן לנסח את ה- $Batch$ Normalizing transform (Mini) כך:

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i, \sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$$

כאשר β, γ הם פרמטרים נלמדים (עבור כל ניירון יש פרמטרים שונים).

בשלב המבחן, השונות והתוחלת שבעזרתם מבצעים את הנרמול אינם נלקחים מהיציאות של הניירונים, אלא לוקחים ממוצע של כמה מה-Mini Batch האחרונים.

יש כמה יתרונות לשימוש ב-normalization Batch: האימון נעשה מהר יותר, יש פחות רגשות לאותול של המשקלים, אפשר שימוש ב-learning rate גדול יותר (מוני מגראדיאנט להטפס), אפשר שימוש במגוון פונקציות הפעלה (גם כאשר אין ממורכבות סביב 0) ומספק באופן חלקי גם רגולריזציה (שונות נמוכה בפועל).

4.3.4 Mini Batch

במקרים רבים הדadata-סט גדול, ולחשב את הגראדיאנט עבור כל הדadata צריך הרבה חישוב. בכל צעד של קידום ניתן לחשב את הגראדיאנט עבור חלק מהdataset, וביצע את הקידום לפי הכיוון של הגראדיאנט המתתקבל. למשל, ניתן לבחור באופן אקראי נקודה אחת ולחשב עליה את הגראדיאנט. בחירה כזו נקראת (SGD), Stochastic Gradient Descent. כיוון שבכל צעד יש בחירה אקראיית של נקודה. בחירה אקראיית של נקודה בודדת יכולה לגרום לשונות גדולות ככל שהחישוב מתמקד, ולכן בדרך כלל מבצעים mini-batch learning – חישוב הגראדיאנט על חלק מהdataset. באופן זה גם יש הפחיתה של כמות החישובים, וגם אין שונות גבוהה. אם מבצעים את החישוב בשיטה זו יש לדאוג שהdataset מעורב כדי שהמשקלים אכן יתעדכו בצורה נכונה, ובנוסף שה-mini-batch יהיה מספיק גדול כך שהיא בו יציג לכל הדadata. כל מעבר על פני כל הדadata-סט נקרא Epoch (אם הדadata הוא בגודל N, והגודל של כל batch הוא S, אז כל Epoch הוא S/N איטרציות).

אמנם כל צעד הוא קירוב לגראדיאנט, אך החישוב מאד מהיר ביחס לגראדיאנט המדויק, וזה יתרון משמעותי שיש לשיטה זו על פני learning batch. בנוסף, המשקלים שמתאימים קרובים מאד לאלו שהו מתקבלים באמצעות batch learning 3.8, כפי שפורסם באירוב, batch learning

4.3.5 Gradient Descent Optimization Algorithms

בשיטת GD, עדכון המשקלים בכל צעד הוא: $w_i = w_{i+1} - \epsilon \frac{\partial L}{\partial w}$, כאשר ϵ הוא פרמטר שנקרא (lr), Learning Rate, והוא קבוע עד כמה יש לשנות המשקל בכיוון הגראדיאנט. בגין דבוקט רגרסיה, אופטימיזציית רשת ניירונים היא לרוב בעיה שאינה קמורה, שכן לא מובטחת התכנסות למינימום הכלובלי. משום כך, אם בכל צעד הולכים יותר מדי לכיוון הגראדיאנט השילילי, ניתן להתכנס לנקודות אוקפי או למינימום לוקאלי שהוא אינן בהכרח המינימום הכלובלי. מצד שני אם מתקדים מעט מדי לכיוון הגראדיאנט, המשקל בקושי מתעדכן. פרמטר ה-zeta נועד להתגבר על בעיות אלו, שכן צריך שלא יהיה גודל מדי (אחרת תהיה התבדורות של המשקלים או התכנסות למינימום לוקאלי) ולא יהיה קטן מדי (אחרת לא תהיה התקדמות או שהיא תהיה מאד איטית). כיוון שאין ערך אבסולוטי שמתאים לכל הבעיה, יש מגוון שיטות המנסות למצוא את העדכון האופטימלי בכל צעד. יש שיטות משתמשות בפרמטר משתנה – lr – adaptive, ויש שיטות שימושיפות פרמטרים אחרים לביטוי של העדכון.

Momentum

ישנם מצבים בהם יש כל מיני פיתולים בדרך לנקודת מינימום. במצב זה, בכל צעד הגראדיאנט יפנה לכיוון אחר, וההתכנסותلنקודת מינימום תהיה איטית. הדבר דומה לנחל שוזרים לים, אך הוא לא זורם ישן אלא יש לו הרבה פיתולים. כדי להאיץ את ההתכנסות במקורה זה, ניתן לנסות לבדוק את הכיוון הכללי של הגראדיאנט על סמך כמה צעדים, ולהוציא התקדמות גם לכיוון זהה. שיטה זו נקראת מומנטום, כיוון שהיא מחפש את המומנטום הכללי של הגראדיאנט. 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 בטווח $[0.9, 0.99]$. 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 יLER ויקטן, וכך תהיה השפעה לצעדים שכבר היו לפני הרבה עדכון. תחת הנחה שהגראדיאנט זהה לכל הפרמטרים, ניתן לפתח נוסחה סגורה לרקורסיה:

$$m_{i+1} = \mu m_i - \epsilon \frac{\partial L(w)}{\partial w} = \mu^2 m_{i-1} - \mu \epsilon \frac{\partial L(w)}{\partial w} - \epsilon \frac{\partial L(w)}{\partial w} = \dots = -\epsilon \frac{\partial L}{\partial w} (1 + \mu + \mu^2)$$

הביטוי שמתקיים הוא סדרה הנדסית מתכנסת, ובoser הכל מתקיים הביטוי:

$$w_{i+1} = w_i - \frac{\epsilon}{1 - \mu} \frac{\partial L}{\partial w}$$

היעילות של המומנטום תלויה בבעיה – לעיתים היא מאייצה את התכנסות ולפעמים כמעט ולא لها השפעה, אך היא לא יכולה להזיק.

וריאציה של שיטת המומנטום נקראת Nesterov Momentum. בשיטה זו לא מחשבים את הגראדיאנט 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 עבור בעיות קמורות, ככלומר היא מצליצה לתכנס יותר טוב מאשר המומנטום הרגיל, אך היא איטית יותר.

learning rate decay

באימון רשותות עמוקות בדרך כלל כדאי להקטין את μ עם הזמן. הסיבה לכך היא שככל שמתתקדים לכיוון המינימום, יש צורך בצעדים יותר קטנים כדי להצליח לתכנס אליו ולא לחוץ מסביבו מצד לצד. עם זאת, קשה לקבוע כיצד לבדוק להקטין את μ : הקטנה מהירה שלו תימנע הגעה לאזור של המינימום, והקטנה איטית שלו לא תעצור 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 שכאר גוף הלמידה של ה-validation יש להקטין את ה- μ .
- ב. דעיכה אקספוננציאלית של ה- μ : $\mu = \mu_0 e^{-kt}$, כאשר k , μ_0 הם היפר-פרמטרים, ו- t יכול להיות צעד או epoch.
- ג. דעיכה לפי $1/t$: $\mu = \frac{\mu_0}{1+k}$, כאשר k , μ_0 הם היפר-פרמטרים, ו- t הינו צעד של עדכון.

Adagrad and RMSprop

בעוד השיטה הקיימת מעדכנת את μ בצורה קבועה מראש, ניתן לשנות אותו גם באופן מסתגל לפי ההתקדמות בכיוון הגראדיאנט. בכל צעד ניתן לבדוק עד כמה גדול היה השינויים בצעדים הקודמים, ובהתאם לכך אפשר לקחת צעד מתאים, מתוך מגמה להקטין אותו ככל שמתתקדים לכיוון המינימום. באופן פורמלי, אלגוריתם Adagrad נראה 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_0 הוא מספר קטן הנועד למנוע חילוקה ב-0. כיוון ש- α_i הולך וגדל, הביטוי $\frac{\epsilon}{\sqrt{\alpha_i + \epsilon_0}}$ הוא ביחס ישיר לקצב ההתקדמות בכיוון הגרדיאנט. בכר מרוויחים דעיכה של - α_i , בקצב המשנה לפי ההתקדמות. באופן יחסי, הדעיכה של - α_i מהירה, כיוון שהסכום $\sum_{j=1}^i \left(\frac{\partial L}{\partial w_j}\right)^2$ גדל במהלך המהלך. 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 הגרדיאנט של כל הצעדים הקודמים באופן שווה, מבצעים moving average, וכך שבעבר יותר צעדים מצטרים עד לצעד הנוכחי, כך תהיה לו פחות השפעה על דעיכת - α_i :

$$w_{i+1} = w_i - \epsilon_i \frac{\partial L}{\partial w}, \quad \epsilon_i = \frac{\epsilon}{\sqrt{\alpha_i + \epsilon_0}}, \quad \alpha_i = \beta \alpha_{i-1} + (1 - \beta) \left(\frac{\partial L}{\partial w}\right)^2$$

Adam

ניתן לשלב בין הרעיון של מומנטום לבין adaptive learning rate:

$$\begin{aligned} \alpha_i &= \beta_1 \alpha_{i-1} + (1 - \beta_1) \left(\frac{\partial L}{\partial w}\right)^2, \quad m_i = \beta_2 m_{i-1} + (1 - \beta_2) \frac{\partial L}{\partial w} \\ \hat{\alpha}_i &= \frac{\alpha_i}{1 - \beta_1^i} \hat{m}_i = \frac{m_i}{1 - \beta_2^i} \\ w_{i+1} &= w_i - \frac{\epsilon}{\sqrt{\hat{\alpha}_i + \epsilon_0}} \hat{m}_i \end{aligned}$$

מספרים טיפוסיים: ($\epsilon = 10^{-2}$ or 5^{-4}) $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\alpha_0 = 1$. האלגוריתם למעשה גם מוסיף התקדמות בכיוון המומנטום (הכיוון הכללי של הגרדיאנט), וגם מביא לדעיכה אדפטטיבית של - α_i . זה האלגוריתם הכי פופולרי בראשות עמדות, אך הוא לא מושלם ויש לו שתי בעיות עיקריות: האימון הריאוני לא יציב, כיון שבתחלת האימון יש מעט מקודות לחישוב הממוצע עבור m_i . בנוסף, המודל המתקיים גוטה ל- overfitting בגין מומנטום.

יש הרבה וריאציות חדשות על בסיס Adam שנדרדו להתגבר על בעיות אלו. ניתן למשל להתחילה לאמן בקצב נמוך, וכאשר המודל מתגבר על בעיית התיצבות הראשונית, להגבר את הקצב (warm-up). במקביל, ניתן להתחילה עם Adam ולהחליף ל-SGD כאשר קритריון מסוים מתקיים. כך ניתן לנצל את ההתקנסות המהירה של Adam בתחלת האימון, ואת יכולת ההמליה של SGD.

4.4 Generalization

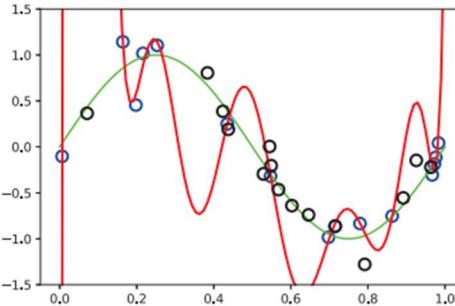
כל מודל שנבנה נסمر על>Data קיים, מתוך מגמה שהמודל יתאים גם לדאטה חדשה. لكن יש חשיבות גדולה שהמודל ידע להצליח כמה יותר טוב, על מנת שהוא יצורה טובה לא רק לדאטה המקוריים אלא גם לדאטה חדש. במקרה אחר, יש לוודא שהמודל לא מתאים את הפרמטרים שלו רק לדוגמאות שהוא רואה, אלא שינסה להבין מתר הדוגמאות מה החקיקות הכללית, שמתאיימה גם לדוגמאות אחרות.

4.4.1 Regularization

כפי שהסביר בפרק 3.1.3, מודל יכול לסבול מהטיה לשני כיוונים – Overfitting ו-Underfitting. Overfitting הוא מצב בו ניתנת הערצת יתר לכל נקודה בסט האימון, מה שגורר מודל מסדר גבוה בעל שונות גבוהה. במצב זה המודל מתאים רק לסט האימון, אך הוא לא מצליח להצליח גם נקודות חדשות. Underfitting הוא המצב ההפוך – מודל שלא מצליח למצאו קי מגמה המכיל מספיק מידע על הדוגמאות הנתונות, ויש לו רעש חזק.

ברשת נוירונים, ככל שמספר הפרמטרים גדול, כך השגיאה של-training test קטנה. לגבי-h-test השגיאה הולכת ויורדת עד נקודה מסוימת, ומשם היא גדלה בחזרה. בהתאם השגיאה יורדת כיוון שמליצחים לבנות מודל יותר מדויק וממנעים מ-underfitting, אך בנקודת מסוימת יש יותר מדי פרמטרים והם נהנים מותאמים יותר מדי לסט האימון. על מנת שירץ למצוא את היחס הנכון בין מספר הפרמטרים (סדר המודל) לבין גודל הדאטה. overfitting. על מנת שירץ למצוא את היחס הנכון בין מספר הפרמטרים (סדר המודל) לבין גודל הדאטה. כיוון שאין אפשר לזרות Overfitting בעזרת ה-training loss בלבד, שחרי-h-Loss קטן ככל שיש יותר פרמטרים, ניתן לחלק את הדאטה לשני חלקים – training and validation. בשלב ראשון בונים מודל בהינתן ה-training set ולאחר מכן בוחנים את המודל על ה-validation set – אם המודל לא מתאים ל-validation set, אז מודל המודל

מתאים רק לדוגמאות שהוא ראה והוא נותן להם הערכת יתר. ככל שהגראף של ה-validation accuracy קרוב יותר ל-training accuracy, כך יש פחתות overfitting.



איור 4.13 בדיקת overfitting בערת validation set. הנקודות הכתולות שייכות ל-training set והשchorות שייכות ל-validation set. המודל האדור ממתאים רק לנוקודות הכתולות, אך אפשר לומר שהוא נוטה ל-overfitting. המודל הירוק לעומת זאת ממתאים גם לנוקודות השchorות, אותן הוא לא ראה בשלב האימון, ככלור הוא הצליח להכפיל טוב גם לדוגמאות חדשות.

האפשרות הכי פשוטה להימנע מ-overfitting היא פשוט להוריד פרמטרים, כלומר להקטין את גודל הרשת. בנוסח ניתן לבצע Early stopping – לחשב בכל Epoch את גրף Epoch Loss של ה-hLoss, וכאשר הוא מתחילה לעלות להפסיק את האימון. שיטות אלה פשוטות מאוד לשימוש, אך ישנן שיטות אחרות שמספקות ביצועים יותר טובים, ונבחן אותן接下來。

4.4.2 Weight Decay

בדומה לרגוליזציה של linear regression, גם ברשת נוירונים ניתן להוסיף איבר ריבועי לפונקציית המחיר, 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 למצער את פונקציית המחיר, שכן נשאף לכך שהabitioי הריבועי יהיה כמה שיורט קטן. בתוספת האיבר עדכון של המשקלים יהיה:

$$w_{i+1} = w_i - \epsilon \left(\frac{\partial L}{\partial w} + \lambda w \right) = (1 - \epsilon \lambda) w - \epsilon \frac{\partial L}{\partial w}$$

הביטוי הזה דומה מאד ל-GD רגיל, כאשר נוסף איבר שאלג'. אם $\lambda < 0$, אז ללא קשר לגודל'יאנט המשקל יורד בכל צעך, וזה נקרא "Weight decay".

ניתן לבצע רגוליזציה עם איבר לא ריבועי, מה שמכונה L1 Regularization:

$$Cost(w; x, y) = L(w; x, y) + \lambda \sum_i |w_i|$$

ואז העדכון יהיה:

$$w_{i+1} = w_i - \epsilon \left(\frac{\partial L}{\partial w} + \lambda \cdot sign(w) \right)$$

בעוד L2 regularization מושך יחיד וניסיה להקטין אותו, L1 regularization "מעונייש" אם סכום המשקלים בערך מוחלט גדול, מה שיגרום לחלק מהמשקלים להתאפס ולדילול מספר הפרמטרים של הרשת.

4.4.3 Model Ensembles and Drop Out

עבור דאטה קיימים לבנות מספר מודלים, ואז כשבאים לבדוק דאטה חדשה בודקים אותו על כל המודלים ולוקחים את המוצע. סט המודלים נקרא ensemble. ניתן לבנות מודלים שונים במספר דרכים:

א. לאמן רשת עם אתחולים שונים למשקלים.

ב. לאמת מספר רשות על חלקים שונים של הדאטא.

ג. לאמן רשות במספר ארכיטקטורות.

יצירת ensemble בדרכים אלה יכולה לעזור בהכללה, אך יקר ליצור את ה-ensemble ולפעמים קשה לשלב בין מודלים שונים.

יש דרך נוספת ליצור ensemble – בוצע Dropout, כלומר למחוק באופן אקראי נוירון אחד או יותר. אם יש רשות מסוימת ומוחקים את אחד הנוירונים – למעשה מקבלים רשות אחרת, ובפועל אפשר לקבל ensemble בעזרת רשות אחת בלבד פעמיים מוחקים מנוירון אחד או יותר. היתרון של יצירת ensemble בדרך זו הוא שהרטשות חולקת את אותן פרמטרים ולבסוף מקבלים רשות אחת מלאה עם כל הנוירונים והמשקלים. בפועל עברו כל דוגימה מגרלים רשות (מוחקים כל נוירון בהסתברות $= 0.5$) וכך לומדים במקביל הרבה רשותות שונות עם אותן פרמטרים. באופן זהה כל נוירון מוכרכ להיות יותר שימושית בלי אפשרות להסתמך על נוירונים אחרים שיעשו את הלמידה, כיוון שלא תמיד הם קיימים. אמנם כל ריצה יחידה יכולה להיות בעלת שונות גבוהה אך המשקלים מביא לשונות נמוכה.

בשלב המבחן, לא מפעילים את ה-Dropout אלא לוקחים את כל הנוירונים, כאשר מחלקים את כל המשקלים בחצי. הסיבה לכך היא שניתן להניח שבשלב האימון חצי מהפעמים המשקל היה 0 כיוון שהנוירון הקשור אליו נמחק, ובחצי מהפעמים היה משקל שנלמד. ניתן גם לקחת הסתברות אחרת למוחיקת נוירונים, למשל $= 0.25$, ועוד שימושים את כל הרשותות השונות לשילוח בהסתברות המתאימה. החיסרונו של שיטה זו הוא שלוקה לה זמן התוכנה.

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 יצירת וירציות של הדוגמאות המקוריות. שיטה זו נקראת **Data Augmentation**, והרעיון הוא לבצע עייפות קטן לכל דוגמא כך שהיא עדין נשמר על המשמעות המקורית שלה, אך תהיה מספיק שונה מהמקור כדי להיות דוגמא נוספת משמעותית בסט האימון. בדומין של תМОנות האוגמנטציות הנפוצות הן:

- סיבוב תמונה בזווית מסוימת (rotate), הנבחרת מהתפלגות איחידה מהתחום $[0, 2\pi]$.
- הוספת רעש לכל פיקסל, כאשר הרעש משתנה מפיקסל לפיקסל, והוא קטן מ- $|e|$.
- שינוי הגודל (rescaling) של התמונה בפקטור מסוים – בדרך כלל הפקטור שייר לתחום $\left[\frac{1}{1.6}, 1.6\right]$.
- 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 בגוני 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 הקשרים והפרמטרים גדלה, באופן צזה שבתי מעשי למחזק את הרשת. מלבד בעיות הגודל, בפועל לא תמיד יש צורך בכל הקשרים, כיון שלא תמיד יש קשר בין כל איברי הכניסה. למשל, עבור תמונה המזונת לרשת, שימוש רבותקשר בין פיקסלים רחוקים בתמונה אינו שימושי, שכן אין חשיבות לחבר את הכניסה לכל הנוירונים בשכבה הראשונה ולקשר בין כל שתי שכבות סמוכות באופן מלא. כדי להימנע מבעיות אלו, לרוב יהיה כדאי להשתמש בשכבות קונבולוציה, שאינן מושכות בין כל 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

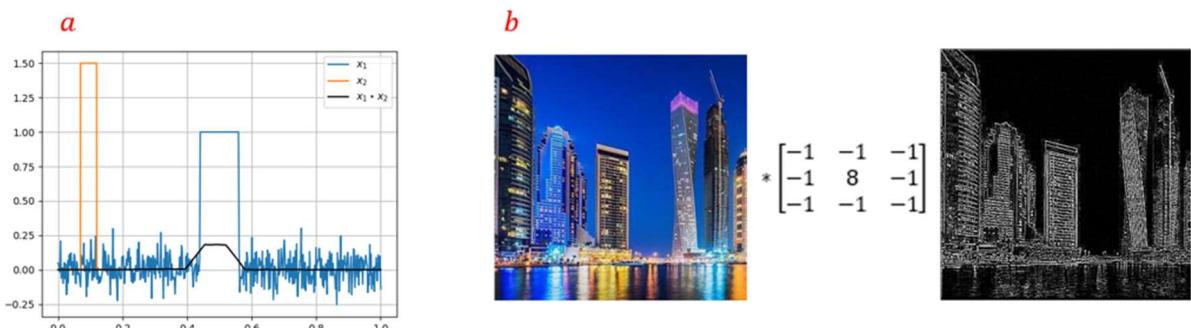
5.1.1 From Fully-Connected Layers to Convolutions

האלמנוט הבסיסי ביותר ברשות קונבולוציה הינו שכבת קונבולוציה לינארית על פני דאטה בכדי לקבל ייצוג אחר ופשטוט יותר שלו. לרוב, שכבת קונבולוציה מבצעת פעולה קרוס-קורלציה בין וקטור המשקלים לבין x מסוים (וקטור הכניסה או וקטור היוצא משכבה חבויה). וקטור המשקלים נקרא גרעין הקונבולוציה (filter) או מסנן (convolution kernel) או מסנן (convolution kernel) או מסנן (convolution kernel).

$$y[n] = \sum_{m=1}^{K-1} x[n-m]w[m]$$

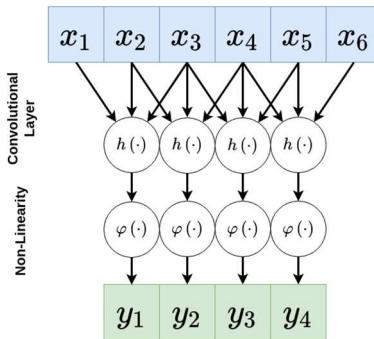
כאשר $x \in \mathbb{R}^n$ הוא וקטור הכניסה ואילו $w \in \mathbb{R}^K$ הוא וקטור המשקלים אשר נלמדים במהלך האימון. וקטור המשקלים w זהה לכל הكنيסות בשכבה וכן מספר הפרמטרים הנלמדים לעומת שכבת FC הינו קטן בהרבה – שכבת FC מכילה $N_{inputs} \times N_{outputs}$ משקלים ואילו שכבת קונבולוציה מכילה K משקלים בלבד (לרוב מתקיים $K \ll N_{inputs} \times N_{outputs}$).

מלבד הקטנות כמות המשקלים, השימוש בגרעין קונבולוציה מסיע לזרחי דפוסים ולמציאות מאפיינים. יכולות אלו מבעות מאופי פעולה הקונבולוציה, הבודקת חפיפה 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 (שחור) בודקת את החפיפה בין הסיגナル לבין הגרעין, וניתן לראות שנקס $\text{sum } x = x_1 + x_2$ יש איזור עם הרבה חפיפה. b) קונבולוציה דו ממדית למציאת קווי מתחאר של בתוך תמונה.

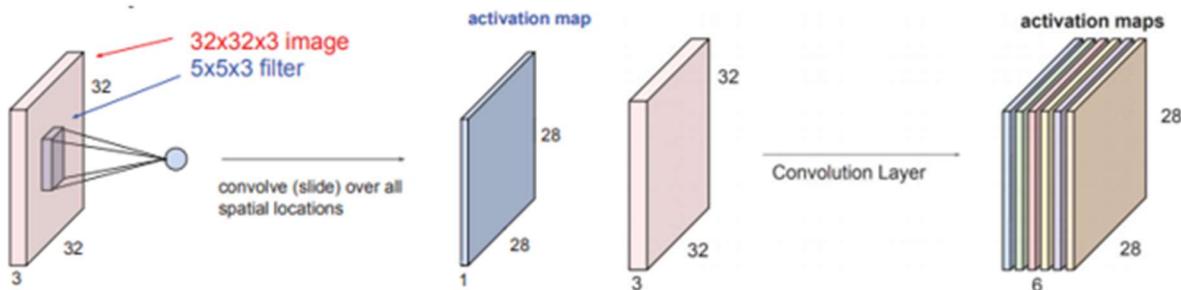
המוצא של שכבת הקונבולוציה עובר בפונקציית הפעלה לא לינארית (בדרכ כל \tanh או ReLU), והוא מכונה מפת הפעלה או מפת מאפיינים (feature map). הקונבולוציה יחד עם האקטיבציה נראה כ:



איור 5.2 דатаה x עובר דרך שכבה קונבולוציה ולאחריה פונקציית הפעלה, ובמזאת מתקבלת מפת אקטיבציה y .

לרוב בכל שכבה קונבולוציה יהיה כמה מסכנים, אשר כל אחד מהם אמור ללמידה מאפיין אחר בתמונה. ככל שהרשף הולכת ועמוקה, כך המאפיינים בתמונה אמרורים להיות מובחנים באופן יותר אחד מהשני, וכך המסכנים בשכבות העומקות 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 העומקות יותר יזהו אלמנטים מורכבים 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, כolumnר 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 המסן לבין אחד דו ממדי אחר בתמונה.



איור 5.3 מסן $\in \mathbb{R}^{5 \times 5 \times 3}$ 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 מספר מסכנים וליצור מפת אקטיבציה עם מספר שכבות – עברו שישה מסכנים הממד של המפה הינו $\in \mathbb{R}^{28 \times 28 \times 6}$ y (ימין).

5.1.2 Padding, Stride and Dilation

כמו ברשת FC, גם ברשת קונבולוציה יש היפר-פרמטרים הנקבעים מראש וקובעים את אופן פעולות הרשת. ישנו שני פרמטרים של שכבה קונבולוציה – גודל המסן ומספר ערוצי הקלט וכן שלושה פרמטרים עיקריים נוספים הקובעים את אופן פעולות הקונבולוציה:

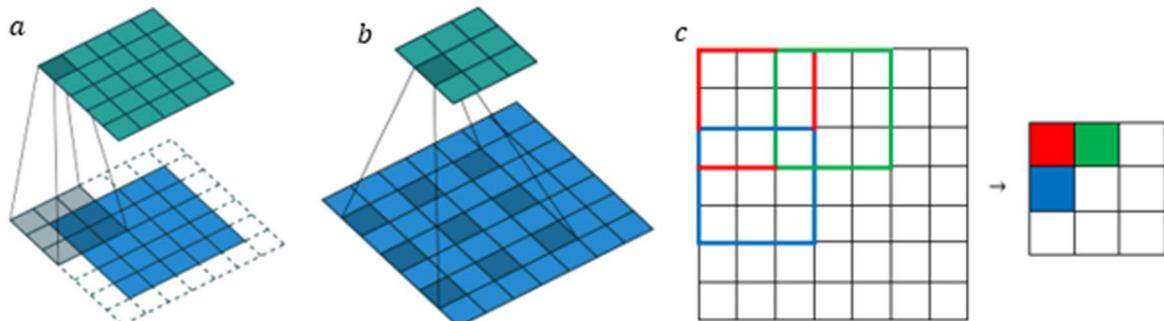
ריפורד (Padding): פעולות הקונבולוציה המוגדרת באמצעות המסן הינה פעולה מרחבית, כלומר, המסן פועל על מספר איברים בכל פעולה. בנוסף, נשים לב כי פעולה הקונבולוציה לא מוגדרת על איברי הקצוות لكن לא נוכל להפעיל את המסן במקומות אלו. באירור 5.2 ניתן לראות כיצד פעולה על תמונה במד של 32×32 מקטינה את ממד הפלט ל- 28×28 , דבר הנבע מכך שהקונבולוציה לא מוגדרת על הפיקסלם בקצוות התמונה וכן לא מופעלת עליהם. אם רוצים לבצע את הקונבולוציה גם על הקצוות, ניתן לרדוף את שולי הקלט (באפסים או שכפול של ערכי הקצה). עבור מסן 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 אזורים 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 המסן על כל האזוריים

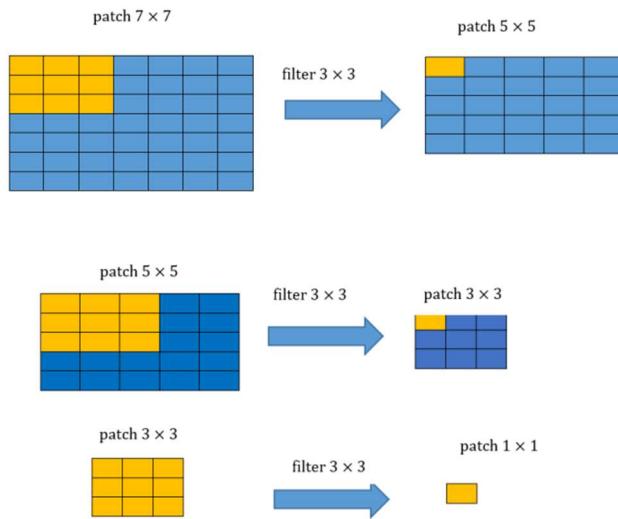
האפשרים ברשות, אלא ניתן לבצע דילוגים, כך שלאחר כל חישוב קונבולוציה יבוצע דילוג בגודל הצעד לפני הקונבולוציה הבאה. גודל צעד טיפוסי הינו $s = 2$.

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 לשימוש לב שערבי ההיפר-פרמטרים (padding, dilation and stride) וכן גודל הגרעין נדרשם להיות מספרים טבעיים אשר מקיימים את נוסחת גודל שכבת הפלט (O) הנ"ל, כך שגם O הינו מספר טבעי.



איור 5.4 (a) ריפוד באפסים על מנת ביצוע קונבולוציה גם על הקצוות של הדאטה. (b) התוצאות ($2 = d$): ביצוע הקונבולוציה וטור השמטת איברים סמוכים מתוך הנהנה שנראה הם דומים. (c) הצעת המשן בצעד של $2 = s$.

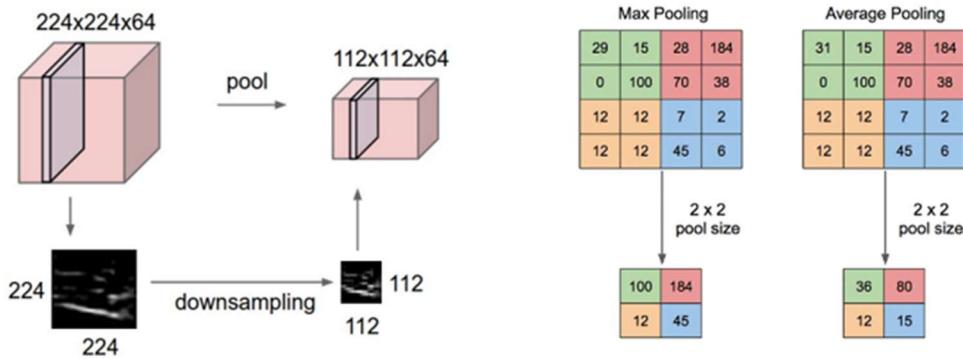
תמרק (Receptive field) של איבר ברשת מוגדר להיות כל התחום בכניסה אשר משפיע על אותו איבר לאורך השכבות.



איור 5.5 של ערך מסוים במושג של שלוש שכבות קונבולוציה רצופות עם משן 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 הרחבה (dilation) כפי שתואר לעיל. שיטה אחרת לניצול עבודה זו היא לבצע Pooling – אחרי כל ביצוע קונבולוציה, דגימתה ערך ייחיד מאזור 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).



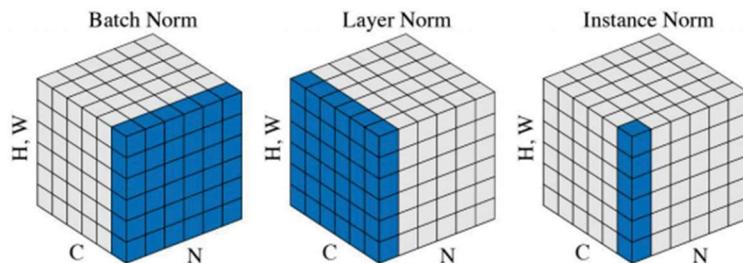
איור 5.6 הקטנת הממד של הדadata בעזרת Pooling (שמאל), והמחשה מספרית של ביצוע max/average pooling בגדול של 2×2 .

5.1.4 Training

ככל, תהליך 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 הגרדיאנטים של כל האזורים השונים. בפועל, הגרדיאנט בכל צעד יהיה הסכום של הגרדיאנטים על פני כל הדadata, ועבור המקרה הכללי בו יש N אזורים שונים עליהם מופעל המסנן הגרדיאנט יהיה:

$$\frac{\partial L}{\partial w_k} = \sum_{i=1}^N \frac{\partial L}{\partial w_k(i)}$$

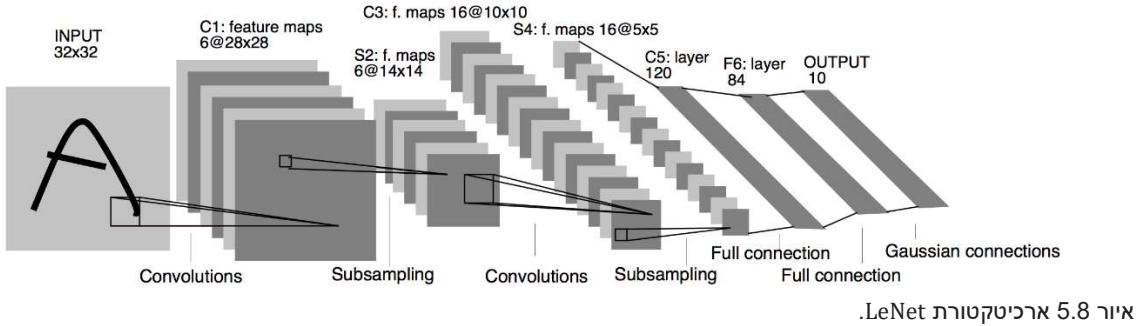
בדומה ל-FC, גם ב-CNN ניתן לבצע נורמליזציה (Batch Normalization, Mini-Batch Normalization) על מנת לאפשר רשת CNN ל训וד על סט של וקטורים מסוימים (לשם הנוחות נתייחס לווקטורים של הדadata כתמונה). אפשרות פשוטה לנפוצה היא לנורמלizar סט של תמונות (Batch Norm), כלומר לנקוט את כל הפיקסלים בסט של תמונות ונורמלם כל מסנן בפני עצמו על פני כמה תמונות (Batch Norm). אפשרות נוספת היא לנקוט חלק מהמידע של סט תמונות, אך לנורמל אותו ביחס לאותו מידע על פני מסננים אחרים (Layer Norm). יש וריאציות של הנורמלים האלה, כמו למשל Layer Norm, Instance Norm, הלוקח מסנן אחד ותמונה אחת ומורמל את הפיקסלים של אותה תמונה.



איור 5.7 נורמול שכבות של רשת קומבולוציה.

5.1.5 Convolutional Neural Networks (LeNet)

בעזרת שרשור של שכבות וחיבור כל האלמנטים הש依יכים לקומבולוציה ניתן לבנות רשת שלמה עבו מגוון משלימות 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 להסתכל על הרשת הכוללת כשי שלבים – בשלב הראשון מבצעים קומבולוציה עם מסננים שונים, שכל אחד מהם נועד לזרות מאפיין אחר, ובשלב השני מחברים חזרה את כל המידע שנансף על ידי חיבור כל הננוירונים באמצעות FC. לראשונה השתמשו בארכיטקטורה זו בשנת 1998, בראשת הנקראת LeNet (על שם Yann LeCun), ומוצגת באיור 5.8. רשת זו השיגה דיוק של 98.9% בזיהוי ספרות, כאשר המבנה שלה הוא שתי שכבות של קומבולוציה ושלוש שכבות FC, כאשר לאחר כל אחת משכבות הקומבולוציה מבצעים pooling.



איור 5.8 ארכיטקטורת LeNet.

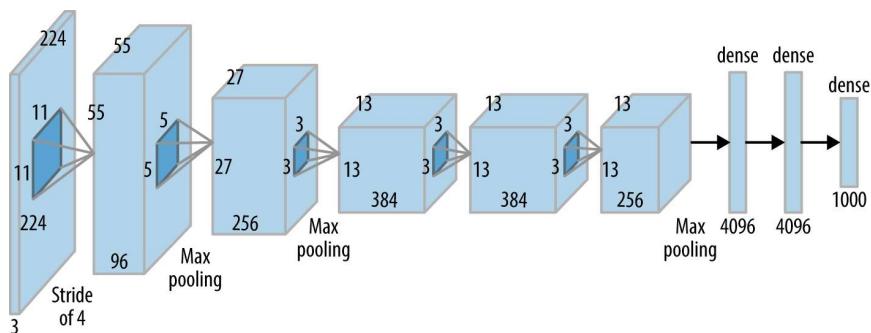
5.2 CNN Architectures

בשנים שלאחר העיסוק ברשתות נוירוניים عمוקות די זנח, עקב חוסר המשאבים לבצע חישובים רבים בעיליות ובמהירות. בשנת 2012 רשות בשם AlexNet המבוססת על שכבות קונבולוציה ניצחה בתחרות ImageNet (תחרות ליזיו תמונות), כאשר היא הציגה שיפור משמעותית מהותוצה הći טוביה בשנה שלפני. יחד עם התפתחות יכולות החישוב, העיסוק ברשתות عمוקות חזר להיות מרכזי ופותחו הרבה מאוד ארכיטקטורות מתקדמות.

5.2.1 AlexNet

רשת AlexNet שאהה את ההשראה למבנה שלה מארכיטקטורת LeNet, כאשר היכולת שלה להתמודד עם משימות יותר מורכבות מאשר LeNet נובעת מכך שנהיי דאטא-סיטים גדולים מאוד שנitin לאם עליהם את הרשות, ובנוסף כבר היה קיים GPU שבזרכתו ניתן לבצע חישובים מורכבים. הארכיטקטורה של הרשות מורכבת שכבות קונבולוציה ושלוש שכבות FC, כאשר לאחר שתי השכבות הראשונות של הקונבולוציה מתבצע pooling ו-normalization. ה-input הוא מממד $3 \times 224 \times 224$, ומופעלים עליו 96 מסננים בגודל 11×11 , עם גודל צעד $s = 4$ ולא ריפוד באפסים. لكن המוצא של הקונבולוציה הינו מממד 55×55 . לאחר מכן מתבצע max-pooling שמחזית את שני הממדים הראשונים, ומתקבלת שכבה בממד 27×27 . בשכבת הקונבולוציה השנייה יש 256 מסננים בגודל 5×5 עם גודל צעד $s = 2$ וריפוד באפסים $= k$, لكن במאז הממד הוא 27×27 , ואחרי max-pooling שכבה בממד 13×13 . לאחר מכן יש עוד 2 שכבות של קונבולוציה עם מסננים בממד 3×3 , גודל צעד $s = 1$ וריפוד $= k$, ואז שכבת הקונבולוציה אחרונה עם 256 מסננים בממד 3×3 , עם $1 = k$. במאז הקונבולוציות יש עוד max-pooling, ואז שלוש שכבות FC, כאשר המוצא של השכבה האחרון הוא וקטור באורך 1000, המציג 1000 קטגוריות שונות שיש בDATA-SET. ImageNet

פונקציית האקטיבציה של הרשת הינה ReLU (בשונה מ- ReLU שהשתמשה ב- tanh), וההיפר פרמטרים הם: $\text{learning rate} = 1e-2$, $\text{SGD+momentum} = 0.9$, $\text{batch size} = 128$, $\text{Dropout} = 0.5$ בערך 60 מיליון.

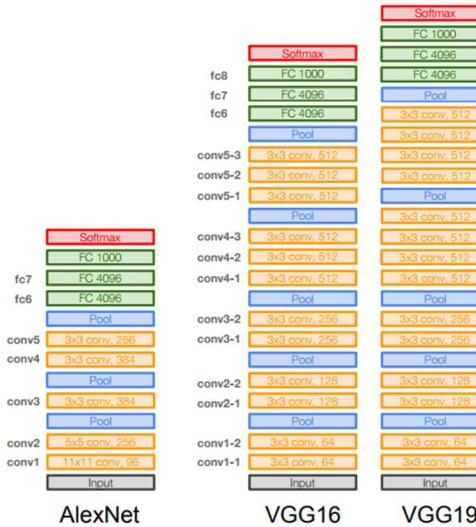


איור 5.9 ארכיטקטורת AlexNet.

שנה לאחר הפרסום של רשת AlexNet, פותחה רשת דומה בשם ZFNet, הבנויה באותה ארכיטקטורה עם הבדלים קטנים בהיפר-פרמטרים ובמספר הפילטרים: השכבה הראשונה של הקונבולוציה הפהה מ- 11×11 ל- 5×5 , 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 אינם נובע מסיבה מסוימת אלא מניסוי וטעיה – במסו תוצאות רבות ומתוקן נבחרה זו בעלת הביצועים הטובים ביותר. לאחר שהרשתות מבוססות קונבולוציה הוכחו את כוחן, השלב הבא היה לבנות רשתות عمוקות, 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 בשלוש שכבות של 3×3 ולקבל את אותן תמר (receptive field), כאשר מרווחים חסכו משמעותי במספר הfilטרים הנלמדים. לפילטר בגודל d הפעול על c ערוצי קלט ופלט יש $c^2 d^2$ פילטרים נלמדים, לכן 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.

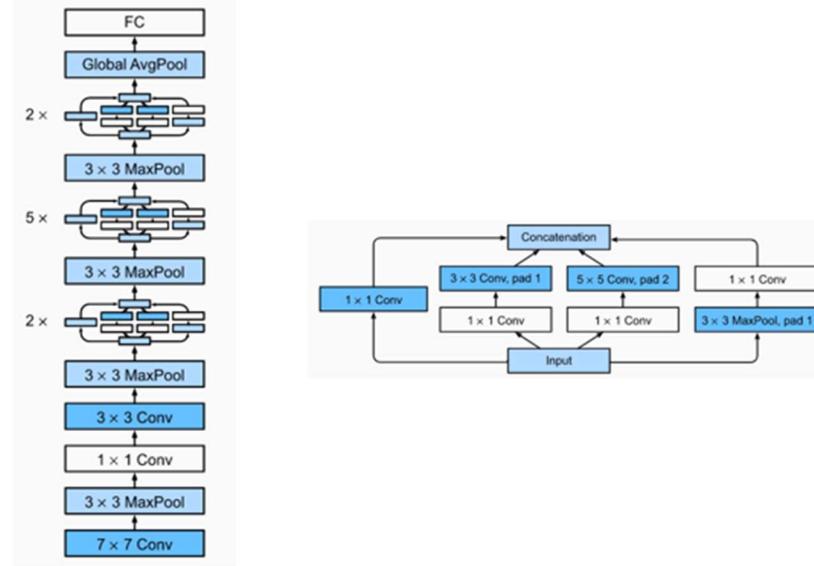


איור 5.10 ארכיטקטורת VGG (ימין) ביחס לארQUITקטורת AlexNet (שמאל).

5.2.3 GoogleNet

המודלים הקודמים היו יקרים חישובית עקב מספר הfilטרים הגדל. כדי להציגם להגעה לאוטם ביצועים עם אותו עומק אבל עם הרבה פחות פילטרים, בוצעת מפתחים מגול 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 שויםvr קר 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 3×3 עם padding 1 max pooling עם 3 stride 2, ולאחריה קונבולוציה 1×1 . לבסוף, הפלטים של ארבעת הענפים משורשרים יחד ומהווים את פלט הבלוק.

המבנה הזה שקול למספר רשותות במקביל, כאשר היתרונות של המבנה זהה הוא כפול: כמה פילטרים נמוכה ביחס לרשותות קודמות וחישובים יחסית מהירים כיון שהם נעשים במקביל. ניתן לחבר שכבות קונבולוציה רגילים עם בלוקים כאלה, ולקיים רשת עמוקה. נעשו הרבה ניסויים כדי למצוא את היחס הנכון בין הרכיבים והמודדים בכל שכבה המביאים לביצועים אופטימליים.



איור 5.11 Inception Block 5.11 (ימין), וארכיטקטורת GoogleNet מלאה (שמאל).

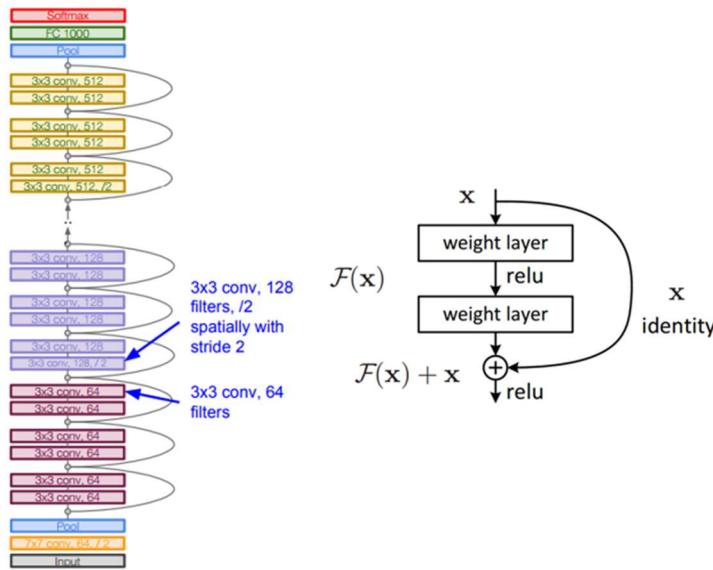
5.2.4 Residual Networks (ResNet)

לאחר שראו שככל שהרשת عمוקה יותר כך היא מושגה תוצאות טובות יותר, ניסו לבנות רשתות עם מאות שכבות, אך הן השיגו תוצאות פחות טובות מהרשתות הקודמות שהיו בעלות סדר גודל של 20 שכבות. הבעה המרכזית של הרשתות העמוקות נבעה מכך שללאור מספר שכבות מסוים התקבל ייצוג מספק טוב, ולכן השכבות היו צרכיות לא לשנות את הקטל אלא להנברר את היצוג כמו שהוא. בשיביל לבצע זאת המשקלים בשכבות אלו צריכים להיות 1. הסתבר לשכבות קשה ללמידה את פונקציית הזרה והן מעשנה פגעו בתוצאה. אתגר נוסף ברשתות עמוקות נבע מהקשה לבצע אופטימיזציה כמו שצריך למשקלים בשכבות עמוקות.

ניתן לנסח את הבעה 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 תוסיף מידע, או לכל הפחות לא תפגע במידע הקיים, בנו רשת חדשה בעזרת Residual Blocks – יצירת בלוקים של שכבות קובולוציה, כאשר ב绷וסף למעבר של המידע בתוך הבלוק, מחברים גם בין הכניטה למצאה שלו. כעת אם בלוק מבצע פונקציה מסוימת $(x)^F$, אז הימצא יינו $x + (x)^F$. באופן זהה כל בלוק ממוקד בלמוד משהו שונה ממה שנלמד עד עכשווי, ואם אין מה להוסיף – הפונקציה $(x)^F$ 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 sotion .weight decay=1e-5 ו batch size=256 .validation error מתישר, ומחולק ב-10 בכל פעם שהה-validation error יתקרב ל-0.001.

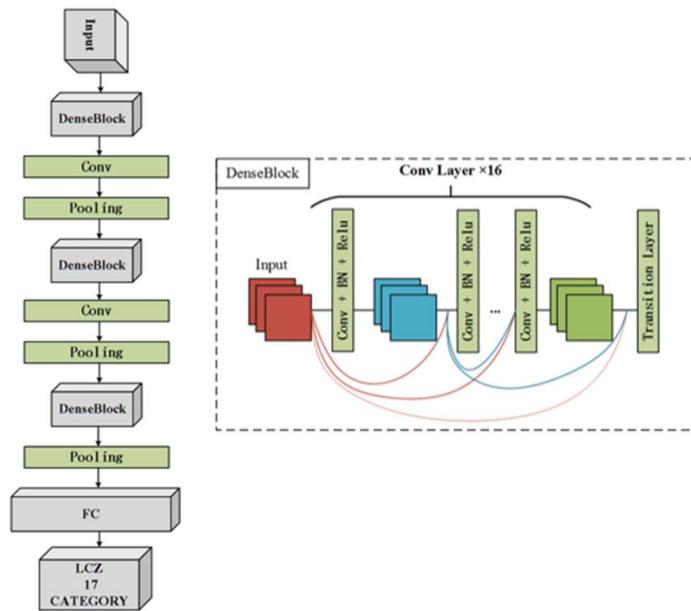
רשתות מתקדמות יותר שילבו את גישת ה-inception יחד עם ResNet על מנת לשלב בין היתרונות של שתי השיטות.



איור 12 Residual Block 5.12 (ימין), וארכיטקטורת ResNet מלאה (שמאל).

5.2.5 Densely Connected Networks (DenseNet)

ניתן להרחיב את הרעיון של Residual Block כך שלא רק מחברים 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 והמודל לא נעשה מורכב מדי.

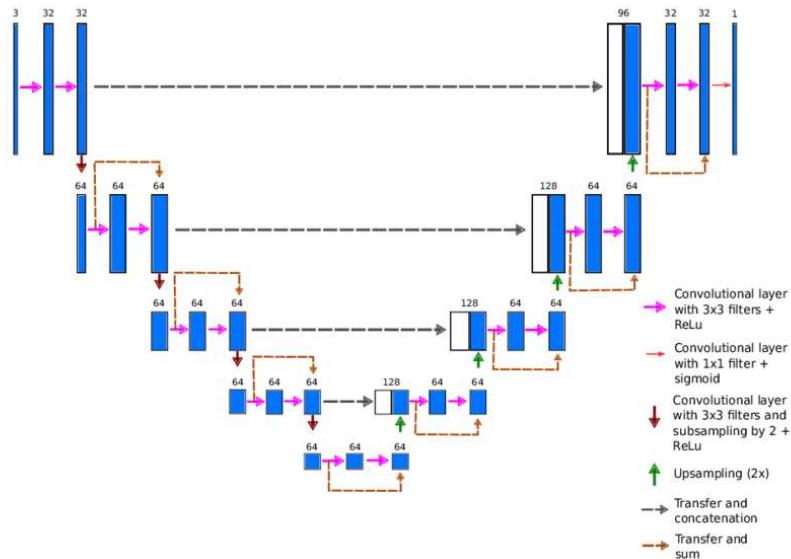


איור 13 Dense Block 5.13 (ימין), וארכיטקטורת DenseNet מלאה (שמאל).

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 פייהם לקבוע מה יש בתמונה, אלא צריך גם לשחרר 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).

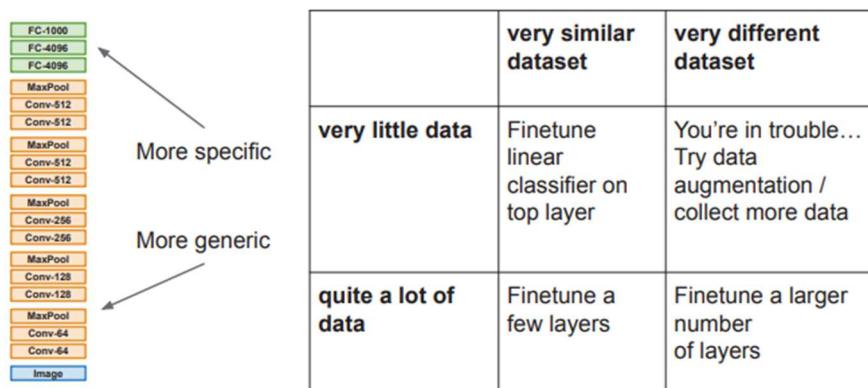
באיור, בחלק הראשון יש טופולוגיה רגילה של רשת קונבולוציה, המבוצעת בעזרת שכבות קונבולוציה וביצוע pooling. השוני בין השלב זהה לבין רשת קונבולוציה קלאסית הוא החיבור שיש בין כל שלב בתהילר לבין חלקים בהמשך התהילר. לאחר המעבר בצדואר הבקבוק יש למעשה שחזור של התמונה עם הסגמנטציה. השחזור נעשה בעזרת sampling על הווקטור שהתקבל במצוא צדוואר הבקבוק יחד עם המידע שנשאר מהחלק הראשון של התהילר. פונקציית המבחן המשמשים 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 ברשומות הקיימות שאומנו כבר ולהתאים אותן לביעות אחרות. גישה זו נקראת **Transfer Learning**, וההגיון מאחוריו טוון שעבור שימוש כל סוג דאטה השכבות הראשונות למודדות אותו דבר (ziehi שפות, קווים וצורות כלליות, מאפיינים כלליים 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, וכך מאמנים את השכבות החדשות על הדאטה החדשה כך שהיינה מוכנות לדאטה הספציפי של המשימה החדשה. ככל שיש יותר דאטה חדש ניתן להוסיף יותר שכבות ולקלד 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. 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 מסוון באיברים השונים יוצרים סדרה שיש לסדר האיברים חשיבות, כמו למשל טקסט, גלי קול, רצף DNA ועוד. כמובן שדאטא מהסוג הזה 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 לכך 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 לחתם 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 ערך העבר. כאשר נכנס וקטור x , הוא מוכפל במשקל w_{xh} ונכנס לרכיב זיכרון h_t , אשר h_t הוא פונקציה של h_{t-1} :

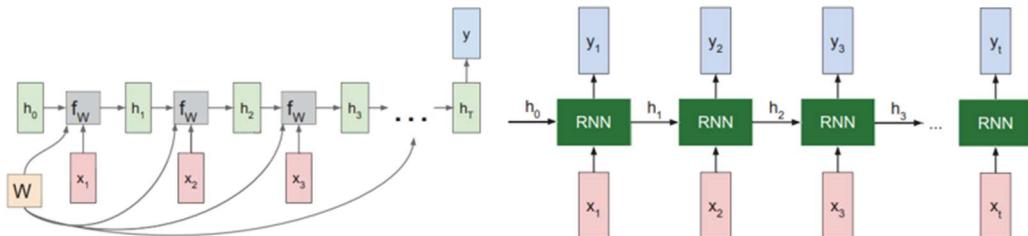
$$h_t = f(h_{t-1}, x_t)$$

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_{hh}, w_{hy}, w_{hx} 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:



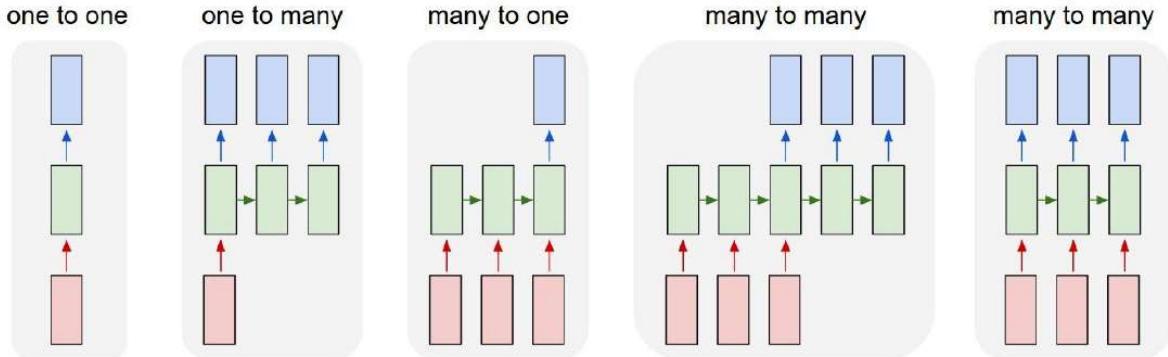
איור 6.1 ארכיטקטורות RNN בסיסיות: Many to One (מימין) – One to many (משמאל) (Many to Many). על כל חץ יש משקל מתאים עליו מתבצע הלמידה.

כמובן שניתן גם לשרשראן שכבות ח比亚ות ולקבל רשת עמוקה, כאשר פלט של שכבה מסוימת הופך להיות הקלט של השכבה הבאה. ישנם מודלים שונים של RNN, המתאיםים לביעויות 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 דאטה (x_i, y_i) – עבור המשמעות cross entropy ועבור בעיות רגסצייה נשתמש בקריטריון MSE. האימון יבוצע בעזרת GD, אך לא ניתן להשתמש ב-*backward propagation* 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-*backward propagation through time* (BPTT) – מסתכלים על הרשות הנפרשת קרשת אחת גדולת, מחשבים את הגרדיאנט עבור כל משקל, ואז סוכמים או ממצאים את כל הגרדיאנטים. אם הדאטה בכניסה הוא בגודל a , ככלומר יש a דוגמאות בזמן, אז יש a רכיבי זיכרון, ו- $a - 1$ – a משקלים w_{hh} . لكن הגרדיאנט המשוקל יהיה:

$$L(\theta) = \frac{1}{n} \sum_i L(\hat{y}_i, y_i, \theta)$$

כאשר הפונקציה $L(\hat{y}_i, y_i, \theta)$ תותאם למשימה – עבור משימות סיווג נשתמש ב-*cross entropy* ועבור בעיות רגסצייה נשתמש בקריטריון MSE. האימון יבוצע בעזרת GD, אך לא ניתן להשתמש ב-*backward propagation* 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-*backward propagation through time* (BPTT) – מסתכלים על הרשות הנפרשת קרשת אחת גדולת, מחשבים את הגרדיאנט עבור כל משקל, ואז סוכמים או ממצאים את כל הגרדיאנטים. אם הדאטה בכניסה הוא בגודל a , ככלומר 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, $w_{hh} = (t)$ והשניי בזמן יהיה רק לאחר ביצוע ה-BPTT ויהיה 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 מתקיימים $1 > ||w_{hh}||$ אז הגרדיאנט יתבדר, ואם $1 < ||w_{hh}||$ הגרדיאנט יתאפס. לעומת זאת, של התבדורות או התאפסות הגרדיאנט, יכולה להופיע גם ברשותות אחרות, אבל בגלל המבנה של RNN והLINEARITY של ה-BPTT ברשותות רקוריוביית זה קורה כמעט תמיד.

עבור הבעיה של התבדורות הגרדיאנט ניתן לבצע clipping – אם הגרדיאנט גדול מקבוע מסוים, מנורמלים אותו:

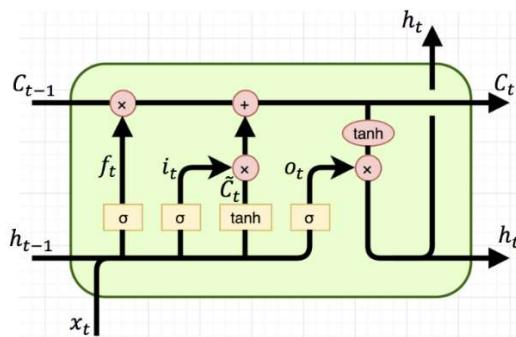
$$\text{if } \|g\| > c, \text{ then } g = \frac{cg}{\|g\|}$$

הבעיה של התאפסות הגראדיינט אמנים 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 אטטת בעקבות טווח ארוך, כמו משפט ארוך או תופעות שימושיות לאט. בגלל הבעיה הזאת לא משתמשים ב-RNN הקלסי (שנקרא גם Vanilla RNN), אלא ממצאים עליון שיפורים, כפי שיאסביר 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 , h_{t-1} , ובעזרתן מחשבים את המוצא על ידי שימוש בפונקציה $(x_t, h_{t-1}) \cdot f_w$. 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 ומוסמנת 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 זיכרון של הטווח 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 ממבצעים פעולה בין x_t לבין h_{t-1} אלא הם נשאים בנפרד ואת כל הפעולות עושים על כל איבר בנפרד). **שער הראשון** בזאת פועל על x_t 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 בזכרון. **שער השני** הוא שער זיכרון והוא אחראי על כמה יש לזכור את המידע החדש לטווח ארוך. אם לדוגמה אכן יש במשפט מוסים נושא חדש, אז השער יחליט שיש לזכור את המידע הזה. אם לעומת זאת המידע החדש הוא תיאור שלRALONOTI להמשך אז אין טעם לזכור אותו. **שער השלישי** הוא שער שכחה שהוא אחראי על כמה מהמידע רלוונטי לדאטת הנוכחית x_t , כלומר מהי הפלט של התא בהינתן מידע בעבר. 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 להזכיר לתא 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 c_t שאחריו על זיכרון לטווח 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 לכינסה (i_t) ולהוציא רק חלק מהמידע המשווקל הקיים (o_t). באופן פורמלי ניתן לנסח את פעולות התא כ:

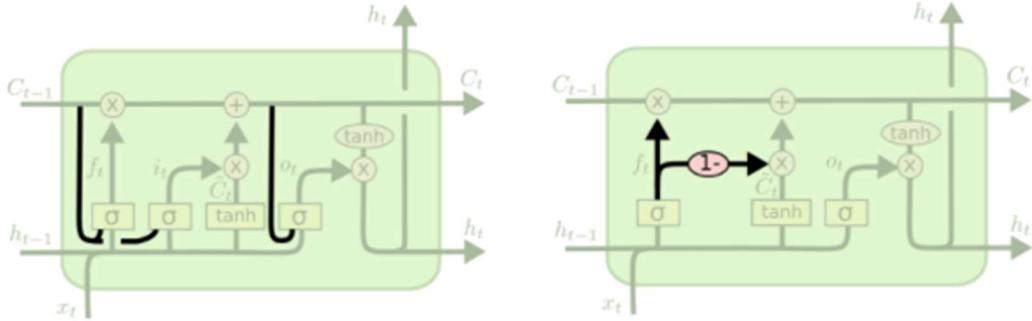
$$\begin{pmatrix} i \\ f \\ o \\ \tilde{c} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} = \begin{pmatrix} \sigma(w_i \cdot [x_t, h_{t-1}] + b_i) \\ \sigma(w_f \cdot [x_t, h_{t-1}] + b_f) \\ \sigma(w_o \cdot [x_t, h_{t-1}] + b_o) \\ \tanh(w_{\tilde{c}} \cdot [x_t, h_{t-1}] + b_{\tilde{c}}) \end{pmatrix}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, h_t = o_t \odot \tanh(c_t)$$

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 להתבונן ב- c_{t-1} באופן ישיר. יש ארכיטקטורות שמחברות את c_{t-1} לכל השערים, ויש ארכיטקטורות שמחברות אותו רק לחלק המשערם. חיבור כל השערים ל- c_{t-1} משנה מבנה את משוואות השערם. במקום $(b + w) \cdot [c_{t-1}, x_t, h_{t-1}]$ המשווה החדש יהיה $c_t = f_t \Theta c_{t-1} + (1 - f_t) \Theta \tilde{c}_t$.

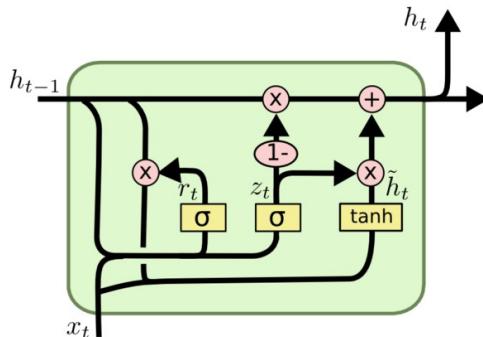
וריאציה אחרת של LSTM מתחדשת בין שער השכחה f_t לבין שער הזיכרון i_t , והחלה עד כמה יש למחוק מידע מהזיכרון מתקבלת יחד עם החלטה כמה מידע חדש יש לכתוב. שינוי זה ישפייע על memory cell, כאשר במקום $c_t = f_t \Theta c_{t-1} + i_t \Theta \tilde{c}_t$, משווהות העדכון תהיה:



איור 6.4 וריאציות של LSTM – LSTM – (ימין) ו-peephole connections (שמאל).

6.2.2 Gated Recurrent Units (GRU)

ישנה ארכיטקטורה נוספת של תא זיכרון הנkirאת (GRU), והוא 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 מtabססים רק על הקלט והמצוא של התא הקודם. כדי לאפשר זיכרון הן לטווח קצר, והן לטווח קצר, 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 [0, 1] $\in t$ כי כיוון שהוא תוצאה של סיגמואיד. מנתנו על \tilde{h}_t ביחס לרכיב זיכרון פשוט של $r_t \cdot h_{t-1}$. כאשר r_t קרוב ל-1 מתקובל הביטוי: Vanilla RNN

$$\tilde{h}_t = \tanh(w \cdot [x_t, r_t \Theta h_{t-1}]) \approx \tanh(w[x_t, h_{t-1}]) = \tanh(w_{hx}x_t + w_{hh}h_{t-1})$$

המשמעות היא שעבור $1 \rightarrow r_t$ מתקבל רכיב הזיכרון הקלאסי, השומר על זיכרון לטוח קצר. אם לעומת זאת $0 \rightarrow r_t$, אז מתתקבל $\tilde{h}_t = \tanh(w_{xh}x_t + w_{hh}h_{t-1})$, ולמעשה הזיכרון של הטווח הקצר מתאפס (reset).

לאחר החישוב של \tilde{h}_t מחשבים את המוצא של המצב החבוי בעזרת z_t , שגם הוא מקבל ערכים בין 0 ל-1:

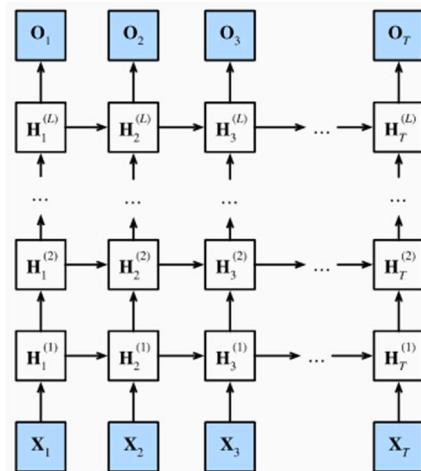
$$h_t = (1 - z_t)\Theta h_{t-1} + z_t\Theta \tilde{h}_t$$

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 מהתוצאות הקודם כמו שהוא ולוקחים את ה- \tilde{h}_t .Candidate hidden state הוא למעשה שקולול של המצב הקודם עם איבר הקלט הנוכחי. עבור כל ערך אחר של z_t , מקבלים שקולול 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 שער העדכון הקרוב 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, שנitinן לשרשר אותם יחד ולקבל שכבה שיכולה לנתח. ניתן להרחיב את המודל הפשוט לרשות בעל מספר שכבות עומוקות.



איור 6.6 ארכיטקטורת Deep RNN

נתאר את הרשות 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 לרשות 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 -הינה 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 . באמצעות 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 $w_{xh}^{(\ell)} \in \mathbb{R}^{h \times n}$, $w_{hh}^{(\ell)} \in \mathbb{R}^{h \times h}$, $b_h^{(\ell)} \in \mathbb{R}^{1 \times h}$, $w_{hh}^{(\ell)}$ הם הפרמטרים של השכבה החבוייה ה- ℓ . הפלט o_t תלוי באופן ישיר רק בשכבה ה- L , והוא מחושב על ידי:

$$o_t = H_t^{(L)} w_{hq} + b_q^{(L)}$$

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

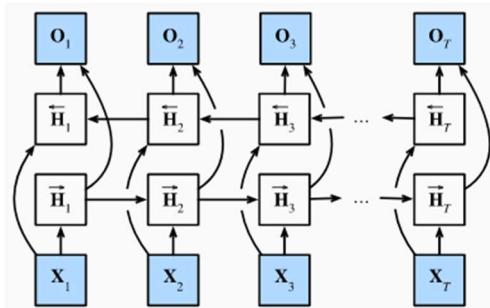
כל הרכיבים והרטות שנידונו עד כה עוסקו בסדרות סיבתיות, כלומר, סדרות בהן כל איבר מושפע מקודמי או אף לא מallow הבאים אחריו. למשל, ערך מניה ביום מסוים קשור לערכה ביום הקודמים, אך הערך שלא ביום המחרת (שכלל עוד לא ידוע) לא מושפע צורה על ערכה ביום הנוכחי. דוגמא נוספת – התפתחות של גל בזמן תוליה בערכי הקודמים של הגל אף אינה מושפעת ממצבי הגל בעבר. זהו אמונם המצב היותר 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 שימושוני על איזו 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, והוא למעשה מבצעת ניתוח של שני מצבים שונים הכוונים שלה במקביל. באופן זה כל מצב חבוי נקבע בו זמן על ידי הנתונים של שני מצבים חבויים אחרים – זה שלפניו וזה לאחריו.



איור 6.6 ארכיטקטורת RNN Bidirectional.

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 $w_{xh}^{(b)} \in \mathbb{R}^{d \times h}, w_{hh}^{(b)} \in \mathbb{R}^{h \times h}, b_h^{(b)} \in \mathbb{R}^{1 \times h}$ $w_{xh}^{(f)} \in \mathbb{R}^{d \times h}, w_{hh}^{(f)} \in \mathbb{R}^{h \times h}, b_h^{(f)} \in \mathbb{R}^{1 \times h}$ הם הפרמטרים של המודל. לאחר החישוב של \vec{H}_t ו- \bar{H}_t משרשים אותם יחד ומקבלים 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 דיסקרימינטיביים, קרי הם מאמנים לבצע פעולות על בסיס נתונים, אך לא יכולים ליצור פיסות מידע או דוגמאות חדשות בעצמן. ברגעוד אליהם קיימים מודלים גנרטיביים, المسؤولים לצור פיסות חדשות על בסיס הדוגמאות שמלמדו. באופן פורמלי, בהינתן אוסף דוגמאות $\mathcal{X} \in \mathbb{R}^{n \times d}$ וטס' $\mathcal{Y} \in \mathbb{R}^d$, מודל דיסקרימינטיבי מאמין לשערך את ההסתברות $(x|y)Pr$. מודל גנרטיבי לעומת זאת לומד את ההסתברות $(y|x)Pr$ (או את $(x|z)Pr$ במקרה שהתגים אינן נתונות), כאשר y , x הן צמד נתונים של דוגמא-label, מתוכן ניתן ליצור דוגמאות חדשות.

ישנם שני סוגי עיקרים של מודלים גנרטיביים: סוג אחד של מודלים מאמן למצואו באופן מפורש את פונקציית הפילוג של הדטה הנתון, ובעזרת הפילוג ליצר דוגמאות חדשות (על ידי דגימה מההתפלגות שמלמדה). סוג שני של מודלים אינם עוסקים בשערוך 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)

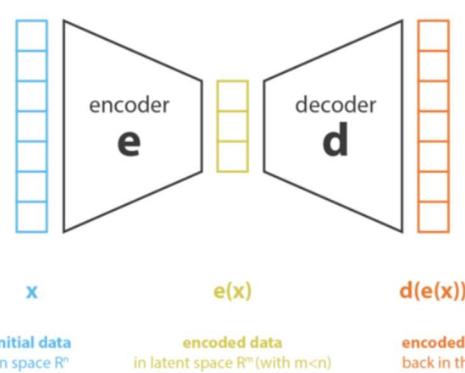
המודל הראשון הינו VAE, וכדי להבין כיצד ניתן בעזרתו ליצור מידע חדש, יש להסביר קודם כיצד מודדים מודדים ומהו החסרונות שלהם. Autoencoder הינו רשת המאומנת לבצע הורדת ממד לדטה, תוך איבוד מינימלי של מידע. כדי להבין את המבנה ואופיו הפעולה שלו, נקדים מעט על הורדת ממד באופן כללי.

7.1.1 Dimensionality Reduction

במקרים רבים, הדטה אותו רוצים לנתח הוא בעל ממד גבוה, למשל גובה, כלמר, לכדי שהיא 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 גבוה 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 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 ליצג את הדטה בממד נמוך יותר, בו באים לידי ביטוי רק המאפיינים המשמעותיים של הדטה.

היצוג של הדטה בממד נמוך נקרא **היצוג הלטנטי** (חבי) או הקוד הלטנטי, ואומרו, יותר קל לבנות מודלים למשימות שונות על סמך היצוג הלטנטי של הדטה מאשר לעובד עם הדטה המקורי. כדי לקבל **יצוג לטנטי איקוטי**, ניתן לאמן אותו באמצעות מנגנון הנקרא **decoder**, הבוחן את יכולת השחזור של הדטה מהייצוג הלטנטי שלו. ככל שניתן לשחרר בצורה מדויקת יותר את הדטה מהייצוג הלטנטי, ככלmore אובדן המידע בתהליך הוא קטן, כך הקוד הלטנטי אכן מייצג בצורה אמינה את הדטה המקורי.

תהליך האימון הוא דו שלבי: פיסת מידע המוצג על ידי קטטור המאפיינים $\mathcal{Z} \in \mathbb{R}^m$ ($m < n$) encoder, שמטרתו להפיך מהדטה את היצוג הלטנטי שלו $\mathcal{Z} = e(x)$, כאשר $n > m$. לאחר מכן התוצאה מוכננת ל-**decoder** בצד השני לשחרר את הדטה המקורי, ולבסוף מתקבל קטטור $\mathcal{Z} = d(e(x))$. 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 לשחרר אותו במלואו בפועל. באופן אינטואיטיבי, אם אנו מצליחים לשחרר את הדטה המקורי מהייצוג שלו בממך הנמוך בדיקות טוב מספיק, נראה שהייצוג הלטנטי הצליח להפיך את המאפיינים העיקריים של הדטה המקורי.



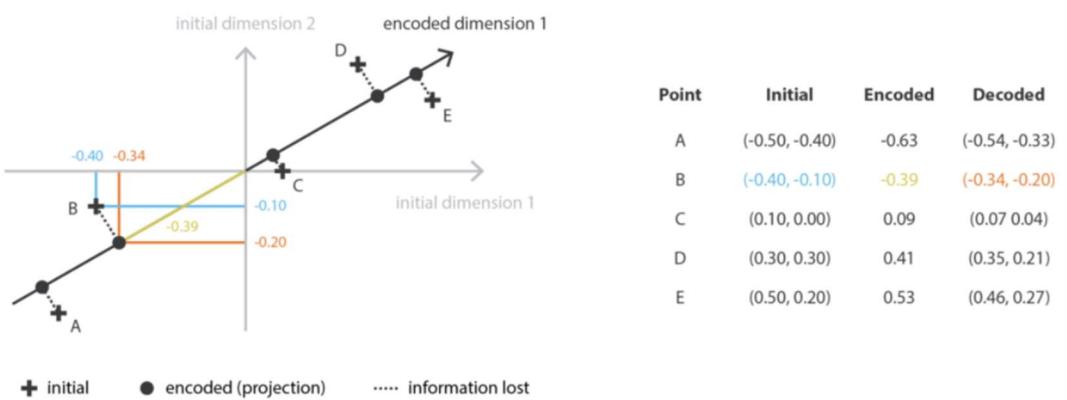
איור 7.1 ארכיטקטורת encoder-**i** decoder.

כאמור, המטרה העיקרית של השיטות להורדת ממד הינה לקבל ייצוג לטנטו איקוטי עד כמה שניתן. הדריך לעשوت זאת היא לאמן את זוג encoder-decoder השומרים על מקסימום מידע בעת הקידוד, ומבייאים למינימום את שגיאות שחזור בעת הפענוח. אם נסמן בהתאמה E ו-D את כל הזוגות של encoder-decoder האפשריים, ניתן לנסח את בעיית הורדת הממד באופן הבא:

$$(e^*, d^*) = \arg \min_{(e,d) \in E \times D} \epsilon(x, d(e(x)))$$

если $((x,e,d)) \in \epsilon$ то есть разница между данными и моделью минимальна.

Principal Components Analysis היא שיטה שמאפשרה להסתכל עליה בצורה הזו (PCA). בשיטה זו מטילים (בצורה לינארית) נתונים מממד n לממד $m < n$, כך שהמאפיינים של הייצוג הlatentiy של הדגימות המקוריות יהיו אורתוגונליים. תהליך זה נקרא גם feature decorrelation, והמטרה שלו היא לסייע את המרחק האוקלידי בין הדאטאות המקוריות לדאטאות המשוחזר, בצורה לינארית גם כן, מהויצוג החדש במרחב ה- m -ממדי.

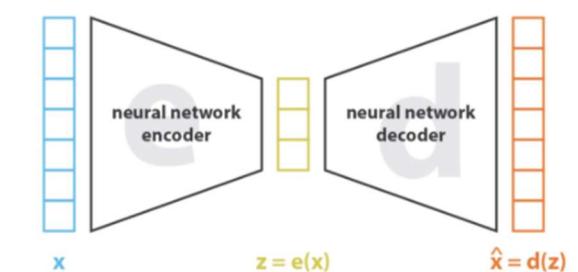


איור 7.2 דוגמא להורדת ממד בשיטת PCA.

במנוחים של encoder-decoder, ניתן להראות כי אלגוריתם PCA מփש את הזוג של encoder-decoder שמקיימים שני תנאים: א. ה-encoder מבצע טרנספורמציה ליניארית על הדטה כך שהמאפיינים החדשניים (בממד נmor) של הדטה יהיו אורתוגונליים. ב. ה-decoder הליניארי המתאים יביא לשגיאה מינימלית במנוחים של מרחק אוקלידי בין הדטה המקורי לבין זה המשוחזר מה-Y'וצג החדש. ניתן להוכיח שהה-encoder האופטימלי מכיל את הוקטוריהם העצמיים של covariace-הdesign, של מטריצת-hencoder, וה-h-decoder הוא השולוף של-h-design.

7.1.2 Autoencoders (AE)

ניתן לחקות את המבנה של encoder-decoder המתואר בפרק הקודם ולהשתמש בראשת נוירונים עבור בניית היצוג החדש ועבור השחזור. מבנה זה נקרא Autoencoder:



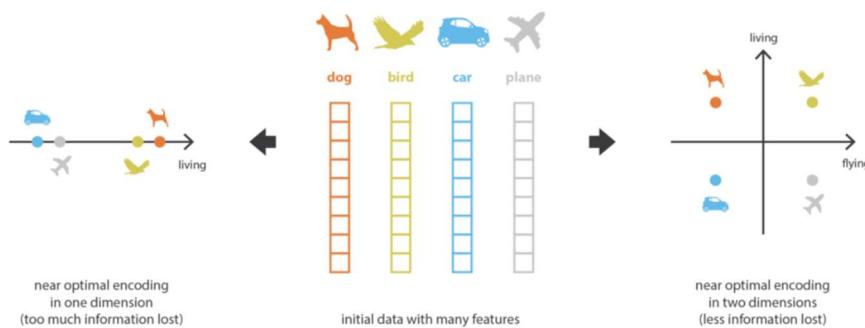
$$\text{loss} = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \|\mathbf{x} - \mathbf{d}(\mathbf{z})\|^2 = \|\mathbf{x} - \mathbf{d}(\mathbf{e}(\mathbf{x}))\|^2$$

איור 7.3 Autoencoder – שימוש ברשתות נירוניים עבור הורדת הממד והשחזור.

בapon זהה, הארכיטקטורה יוצרת לדatta צוואר בקבוק מידע (information bottleneck), שمبرטיח שرك המאפיינים החשובים של הדatta, שבאמצעותם ניתן לשחזר אותה בדיק טוב, ימשכו לייצוג במרחב הlatent. במקורה הפשטן בו בכל רשת יש רק שכבה חניה אחת והיא לא משתמשת בפונקציות הפעלה (activation functions) לא לינאריות, ניתן לירות כי ה-*autoencoder* יחשוף טרנספורמציה לינארית של הדatta, שבאמצעותו ניתן לשחזרו

באופן לינארי גם כן. בדומה ל-PCA, גם רשות כזו 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 שלהבדיל מ-PCA המאפיינים החדשניים (לאחר הורדת ממד) עשויים לצאת לא אורתוגונליים (קורלציה שונה מ-0).

כעת נניח שהרשאות של *encoder* וה-*decoder* הן רשותות עמוקות ומשתמשות בפונקציות הפעלה לא לינאריות. במקרה זה, ככל שהארקיטקטורה של הרשותות מורכבת יותר, כך רשות *encoder*-*decoder* יתירה 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 נוירונים), ניתן להפחית ממד של כל דאטה לחד-ממד ללא כל איבוד מידע. עם זאת, הפקחת ממד דרסטית שכזו עלולה לגרום לדאטה המשוחזר לאבד את המבנה שלה. לכן יש חשיבות גדולה בבחירה מספר הממדים של המרחב הלטני, כך שמצד אחד אכן יבוצעיפוי של מאפיינים פחות משמעותיים ומצד שני המידע עדין יהיה בעל שימושים שונים. 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 נוירונים מורכבת מספיק מאפשרת לבנות ייצוג של כל הדוגמאות על קו ישר, כך שיכל שפרט מסוים נמצא יותר ימיןה, כך הוא יותר "חי". באופן זה אמן מתקיים 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. כמובן שה הפרדה זו היא הרבה יותר פשוטה מאשר הסתכבות על כל הפרמטרים (-הפיקסלים) של הדוגמאות. דוגמא זו מראה את החשיבות שיש בבחירה הממדים של ה-encoder.

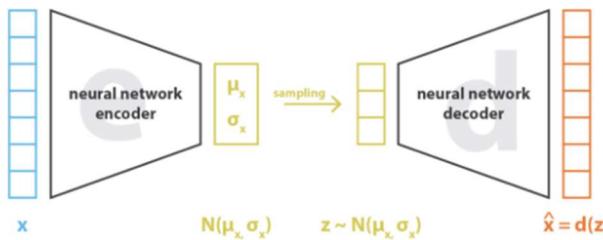
7.1.3 Variational AutoEncoders (VAE)

ניתן לקחת את AE-VAE ולהפוך אותו למודל גנרטיבי, כלומר מודל שמסוגל לייצר בעצמו דוגמאות חדשות שאכן מתפלגות כמו הפולוג של הדטה המקורי. אם מדובר בדומין של תמונות למשל, אז נרצה שהמודול יהיה מסוגל לייצר תמונות שנראות אוטנטיות ביחס לדאטה עליו אומן. AE מאמן לייצג את הדטה בממד נמוך, שולקח בחשבון את המאפיינים העיקריים, ולאחר מכן משחזר את התוצאה לממד המקורי. אולם, מגננן זה אינו אפשרי להשפיע על האופן בו הדטה מייצגת במרחב הלטני. אם יוגרך וקטור כלשהו מהמרחב הלטני – קרוב לוודאי שהוא ייצוג שדומה לדאטה המקורי. אם היינו מכנים אותו ל-*decoder*, סביר להניח שהחזרה לא תהיה דומה בכלל לדאטה 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 שולקח דאטה ורק בונה לו ייצוג מממד נמוך, VAE קובע התפלגות פרוירית למרחב הלטני z – למשל התפלגות נורמלית עם תוחלת 0 ומטריצת covariance Σ . בהינתן התפלגות זו, רשות *encoder*-*decoder* מאומנת לקבל דאטה x ולהוציא פרמטרים של התפלגות פואטריאורית $x|z$, מתחזק מטריה למזרע כמה שניתן את המרחק בין התפלגות z ו- $x|z$. לאחר מכן דוגמים וקטורים מההתפלגות הפואטריאורית $x|z$ (הנתונה על ידי הפרמטרים המוחושבים ב-*encoder*), ומעבירים אותם דרך *decoder* כדי לייצר פרמטרים של התפלגות $x|z$. חשוב להבהיר שאם הדטה המקורי הוא תמונה המורכבת מảoוסף של פיקסלים, אז במושג יתקבל $x|z$ לכל פיקסל $x|z$ בנפרד ומההתפלגות הזו 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 $|z$ טובה, קרי קרובה להתפלגות המקורית של x , ניתן בעזרתה גם ליצור דוגמאות חדשות, ובuczם מתקבל מודל גנרטיבי.

כאמור, h-encoder מנסה לייצג את הדאטה המקורי באמצעות התפלגות במדן נmor יותר, למשל התפלגות נורמלית – עם תוחלת ומטריצת covariance: $\text{covariance} = N(\mu_x, \sigma_x) = (x|z)p \sim z$. חשוב לשים לב להבדל בתפקיד של h-decoder – בעוד שב-VAE הוא מעד לתהילך האימון בלבד ובפועל מה שחייב זה הייצוג הלטנטי, ב-h-VAE החשוב לא פחות מאשר הייצוג הלטנטי, כיוון 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 h-encoder. VAE מאמין רשות שמקבלת דוגמאות מסט האימון, ומנסה להפיק מהן פרמטרים של התפלגות $|z$ הקוראים כמה שnitן לפרמטרים של התפלגות הפרויריות z , שכאמור נקבעה מראש. מההתפלגות הנלמדת זו דוגמים וקטורים לטנטים חדשים ומעבירים אותם ל-h-decoder. h-decoder מבצע את הפעולה ההפוכה – לוקח וקיטור שנדגם מהמרחב הלטנטי $|z$, ומיציר באמצעותו דוגמא חדשה שאמורה להיות דומה לדאטה המקורי. תהילך האימון היה צזה שימזעր את השגיאה של שני חלקים – VAE – גם $|z|$ שבmoza היה כמו שייתר קרוב לא-המקור, וגם התפלגות $|z$ תהיה כמה שייתר קרובה להתפלגות הפרוירית z .

ונתאר באופן פורמלי את בעיית האופטימיזציה של VAE – מנסה לפטור. נסמן את הווקטורים של המרחב הלטנטי $-z$, את הפרמטרים של h-decoder θ , ואת הפרמטרים של h-encoder λ . כדי למצאו את הפרמטרים האופטימליים של שתי הרשות, נרצה להביא למקסימום את $L(\theta; \lambda)$, כלומר למצאו את הנראות המרבית של סט האימון תחת θ . כיוון שפונקציית $\log p$ מונוטונית, נוכל לקחת את לוג ההסתברות:

$$L(\theta) = \log p(x; \theta)$$

אם נביא למקסימום את הביטוי הזה, נקבל את θ האופטימלי. כיוון שלא ניתן לחשב במפורש את $(\theta; x; \lambda)$, יש להשתמש בקיוב. נניח והפלט של h-encoder הוא בעל התפלגות $(\lambda; |z|)$ (מה ההסתברות לקבל את z בהינתן x בכנסה), וננסה לייצג את התפלגות זה בעזרת רשות נוירונים עם סט פרמטרים λ . כתע ניתן לחלק ולהכפיל את $L(\theta; \lambda)$:

$$\log p(x; \theta) = \log \sum_z p(x, z; \theta) = \log \sum_z q(z; \lambda) \frac{p(x, z; \theta)}{q(z; \lambda)} \geq \sum_z q(z; \lambda) \log \frac{p(x, z_i; \theta)}{q(z; \lambda)}$$

כאשר אי השווין האחרון נובע מאי-שוויון ינסן, והביטוי שמייןראי השוויון נקרא Evidence Lower Bound (ELBO). ניתן להוכיח שההפרש בין ELBO(θ, λ) לבין הערך שלפני הקירוב הוא המרחק בין שתי התפלגות $(z|x, \theta, \lambda)$, שנקרא Kullback-Leibler divergence ומסומן ב- $\mathcal{D}_{KL}(q(z|x, \theta, \lambda) \| p(x, z_i; \theta))$

$$\log p(x; \theta) = ELBO(\theta, \lambda) + \mathcal{D}_{KL}(q(z; \lambda) \| p(z|x; \theta))$$

אם שתי התפלגות זהות, אז מרחק \mathcal{D}_{KL} ביניהן הוא 0 ומתקיים שוויון: $\log p(x; \theta) = ELBO(\theta, \lambda)$. כזכור, אנחנו מחפשים למציאת פונקציית המחר $\log p(x; \theta)$, וכעת בעזרת הקירוב ניתן לרשום:

$$L(\theta) = \log p(x; \theta) \geq ELBO(\theta, \lambda)$$

$$\rightarrow \theta_{ML} = \arg \max_{\theta} L(\theta) = \arg \max_{\theta} \max_{\lambda} ELBO(\theta, \lambda)$$

כתע ניתן בעזרת שיטת Gradient Descent (GD) למציאת האופטימום של הביטוי, וממנו להפיק את הפרמטרים האופטימליים של h-encoder ושל h-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:

$$= \mathbb{E}_{q(z|x; \lambda)} \log N(x; \mu_\theta(z), \sigma_\theta(z)) - \mathcal{D}_{KL}(N(\mu_\lambda(x), \sigma_\lambda(x)) \| N(0, I))$$

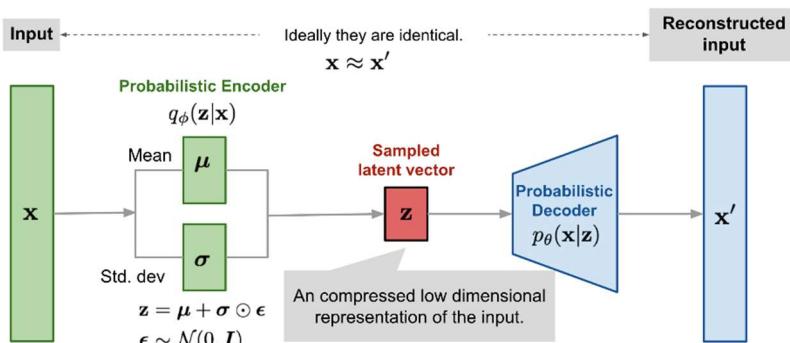
כדי לחשב את התוחלת ניתן פשוט לדגום דוגמאות מההתפלגות $(x|z)$ ולקבל:

$$\mathbb{E}_{q(z|x; \lambda)} \log N(x; \mu_\theta(z), \sigma_\theta(z)) \approx \log N(x; \mu_\theta(z), \sigma_\theta(z))$$

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 דוגמת גם למצער הפרש בין הכניסה של VAE לבין המוצא שלו, וגם למצער המרחק בין ההתפלגות הפרוירית של z לבין ההתפלגות $x|z$ שבמוצא encoder-VAE.



$$\begin{aligned} x_t &\rightarrow \mu_\lambda(x_t), \Sigma_\lambda(x_t) \rightarrow z_t \sim \mathcal{N}(\mu_\lambda(x_t), \Sigma_\lambda(x_t)) \rightarrow \mu_\theta(z_t), \Sigma_\theta(z_t) \\ ELBO &= \sum_t \log \mathcal{N}(x_t; \mu_\theta(z_t), \Sigma_\theta(z_t)) - \mathcal{D}_{KL}(\mathcal{N}(\mu_\lambda(x_t), \Sigma_\lambda(x_t)) \| \mathcal{N}(0, I)) \end{aligned}$$

איור 7.6 תהליכי הלמידה של VAE

כאשר נתון אוסף דוגמאות X , ניתן להעביר כל דוגמא x כ-encoder ולקבל עבורה את σ_x . לאחר מכן דוגמים נקבעו לテンטי z מההתפלגות עם פרמטרים אלו, מעבירים אותם כ-decoder ומקבלים את σ_z . לאחר התהילר ניתן להציב את הפרמטרים המתקיים ב-ELBO ולחשב את ערך פונקציית המחר. ניתן לשימוש לבשה-ELBO מרכיב שני איברים – האיבר הראשון משער את הדמיון בין הדוגמא שבסכינסה לבין התפלגות שמתבקשת במצב, והאיבר השני מבצע רגולרייזציה לתפלגות הפרIORיות במרחב הlatent. הרגולרייזציה גורמת לכך שההתפלגות במרחב הlatent $x|z$ תהיה קרובה עד כמה שנitinן לתפלגות הפרIORיות z . אם התפלגות במרחב הlatent קרובה לתפלגות הפרIORיות, אז ניתן באמצעות decoder ליצור דוגמאות חדשות, ובמובן זה ה-VAE הוא מודל גנרטיבי.

הדגימה של z מההתפלגות במרחב הlatent יוצרת קושי בחישוב הגדריאנט של ה-ELBO,回购 Reparameterization trick – דוגמים z_0 מההתפלגות נורמלית סטנדרטית, ואז כדי לקבל את ערך הדגימה של z משתמשים בפרמטרים של ה-encoder: $(x) = \sigma_x + \sigma_z z_0$. בגישה זו כל התהילר יהיה דטרמיניסטי – מגרילים z_0 מראש ואז רק נשאר לחשב באופן סכמטי את התפשטות הערך ברשות.

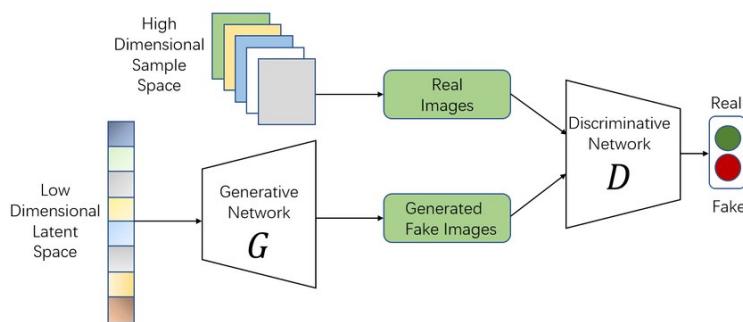
7.2 Generative Adversarial Networks (GANs)

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 הממקסימים את הנראות המירבית של סט האימון), אלא יוצרים>Data באופן אחר. הרעיון הוא לאמן שתי רשותות במקביל – רשות אחת שלומדת לייצר דוגמאות, ורשת שנייה שלומדת להבחן בין דוגמא אמיתית מסט האימון לבין תמונה סינטטית שנוצרה על ידי הרשות הראשונה. הרשות הראשונה מאומנת לייצר דוגמאות שייגרם לרשות השנייה לחושב 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>Data האמיתי.

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 שיזכרת>Data (generator) ורשת שמכריעה האם>Data האם הدادה ה-*real* סינטטי או אמיתי (discriminator), כאשר האימון נעשה על שתי הרשותות יחד. ה-*discriminator* מקבל כקלט ה-*real* את הדוגמאות האמיתיות והן את הפלט של ה-*generator*, כדי ללמד להבחן בין>Data אמיתית לבין>Data סינטטי. ה-*generator* מייצר דוגמאות ומתקבל פידבק מה-*discriminator* וכך לומד לייצר דוגמאות שנראות אמיתיות. נסמן את ה-*generator* ב-*G*, ואת ה-*discriminator* ב-*D*, ונקבל את הסכמה הבאה:



איור 7.7 ארכיטקטורת GAN.

ה-*discriminator* D הוא למעשה מסוג שהפלט שלו הוא הסתברות שהקלט הינו דוגמא אמיתית, ונסמן ב- (x) את ההסתברות הזו. כדי לאמן את ה-*discriminator* נרצה להשיג שני דברים: א. למקסם את (x) עבור x מסט האימון, כלומר, לטעות כמה שפחות בזיהוי>Data אמיתי. ב. למזער את (x) עבור>Data אמיתית, 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 ה-*generator* מועוני לגרום ל-*discriminator* להוציא ערכים כמה שיותר גבוהים עבור>Data הסינטטי שהוא מייצר. בשwil לאמן ייחד את שני חלקים המודול, נבנה פונקציית מחיר 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-*discriminator* יחשב 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 דатаה 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, המטריה 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- D_{ϕ_d}). האימון מתבצע במספר מסויים של Epochs, כאשר כאמור מאמנים לסירוגין את G ו- D . בפועל דוגמאים mini-batch בגודל m מסת 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-*discriminator*, 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 המטריה של ה-*generator*. כיון שבהתחלת הדגימות המיוצרות על ידי ה-*generator* לא דומות לחולוטן לאלו מסת האימון, ה-*discriminator* מזהה אותן בקלות כمزיפות. כתוצאה לכך הביטוי $(z) D(G(z))$ מקבל ערכים מאד קרובים ל-0, ומילא גם הביטוי $(1 - D(G(z)))$ קרוב 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 $\mathbb{E}_{z \sim Noise} \log \left(1 - D(G(z)) \right)$ מփשים מינימום לביטוי $(D(G(z)) - \mathbb{E}_{z \sim Noise} \log(D(G(z)))$. 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 ה-*discriminator* עבור generator נתון, ובוורו לחשב את הערך של פונקציית המחיר. לשם הנוחות נסמן את התפלגות הדטה האמיתית ב- r_g , ואת התפלגות הדטה הסינטטי המיוצר 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 המקירה בו-h-GAN מצליח לייצר דוגמאות שנראות אמיתיות לחלווטין, כלומר $p_g(x) = p_r(x)$, אז מתקיים $D(x) = \frac{1}{2}$. הסתברות זו משמעותה שהיא discriminator לא יודע להחליט לגבי הקבלת המתקבל, והוא קובל שהסתברות שהקלט אמיתי זהה לזה שהוא הקלט 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 המתקבל 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:

$$\mathcal{D}_{JS} = \frac{1}{2} \mathcal{D}_{KL}(P||M) + \frac{1}{2} \mathcal{D}_{KL}(Q||M), M = \frac{1}{2}(P + Q)$$

קייםנו שבעור D אופטימלי, פונקציית המחיר שווה למרחק \mathcal{D}_{JS} בין p_r לבין p_g עד כדי קבוע, ובאופן מפורש:

$$V(G, D_{opt}) = \mathcal{D}_{JS}(p_r, p_g) - \log 4$$

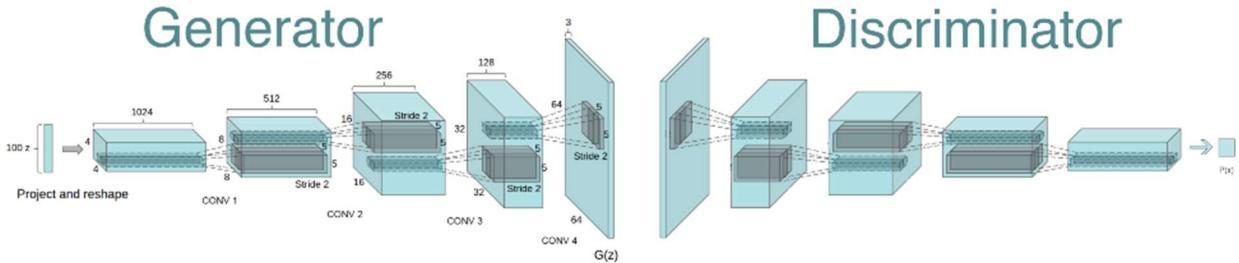
כאשר G אופטימלי ומתקיים $(p_r(x) = p_g(x))$, אז המרחק בין התפלגיות שווה 0, כלומר $\mathcal{D}_{JS}(p_r, p_g) = 0$, ולכן מתקבל:

$$V(G_{opt}, D_{opt}) = -\log 4$$

יש משמעות גדולה לביטוי שהתקבל – ככל שנצליח למזרע יותר את ($\mathcal{D}_{JS}(p_r, p_g)$, כך נצליח לקבלGAN יותר טוב).

7.2.2 Deep Convolutional GAN (DCGAN)

כפי שהוסבר בפרק 5, רשותות קונבולוציה יעילות יותר בהדמיין של תמונות מאשר FC. לכן היה טבעי לחקת 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 רשת קונבולוציה שעובד סיווג ביןארו אם התמונה אמיתי או סינטטי. DCGAN הומצא ב-2015 ומאז פותחו רשותות שמייצרות תמונות יותר איכותיות הן מבחינת הרזולוציה והן מבחינת הדמיין שלון 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 אונטנית. למשל, עבור אוסף תמונות המציגות את הספורות 0-9, ונרצה שה-GAN יוצר תמונה של ספרה מסוימת. במקרה אחד, במקרה מסוים של הוכנת ה- z , ה-GAN מקבל תנאי נוסף נסוסף 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 , המוגדר מההתפלגות מסוימת (בדרכו כלל התפלגות גאומטרית סטנדרטית, אך זה לא מוכחה). ישן גישות נוספת ליצור דאטה חדש, כמו למשל ייצור תמונה חדשה על בסיס קוווי מתאר כליליים שלה. סט האימון במקורה זה בניית מזגות של תמונות והסקציות שלהן.

שיטה Pix2Pix משתמשת בארכיטקטורה של GAN אך במקומם לדגם את וקטור z מההתפלגות ממשהו, Pix2Pix מקבלת סקיצה של תמונה בתור קלט, וה-generator לומד להפוך את הסקיצה לתמונה אמיתי. הארכיטקטורה של ה-discriminator נשארת ללא שינוי ביחס למאה שתואר קודם (פרט להתאמתה לבינה הקלט), אך ה-discriminator מקבל זוג תמונות – אחת הסקיצה ואת התמונה (פעם כמשתנה – במקום לקבלת תמונה ולבצע עליה סיווג ביןאר), הוא מקבל זוג תמונות – את הסקיצה ואת התמונה (פעם כטמונה אימון המתאימה לסקיצה S ופעם זאת שמיוצרת על ידי ה-generator על בסיס S). על ה-discriminator לקבוע האם התמונה היא אכן תמונה אמיתי 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 כך שה-discriminator יאמין שהן אמיתיות, וגם לסייע את המרחק בין התמונה שנוצרת לבין תמונה אמיתיות השיכת לסקיצה.

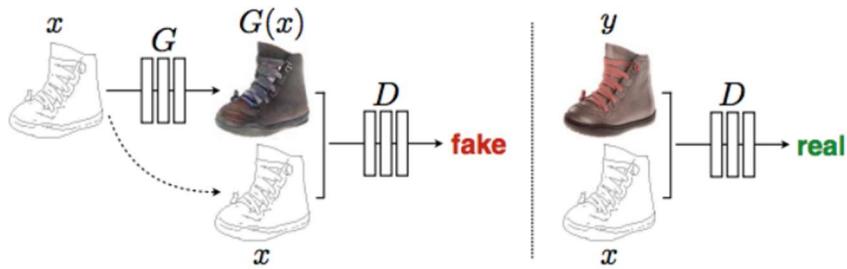
כעת נסמן תמונה אמיתיות השיכת לסקיצה ב- y , ורשום את פונקציית המחר כשי חלקים נפרדים – cross entropy ומודרך אוקלידי L_1 בין תמונת המקור לבין הפלט:

$$V(D, G) = \min_G \max_D \mathbb{E}_{x,y} (\log D(x,y) + \log (1 - D(x,G(x))))$$

$$\mathcal{L}_{L1}(G) = \min_{\theta_g} \mathbb{E}_{x,y} \|G(x) - y\|_1$$

$$\mathcal{L}(G, D) = V(D, G) + \lambda \mathcal{L}_{L1}(G)$$

ניתן להסתכל על pix2pix המפה התמונה למתמונה (image-to-image translation). נציין שבמקרה זה הקלט והפלט של pix2pix שייכים לתחומים (domains) שונים (סקיצה ותמונה רגילה).

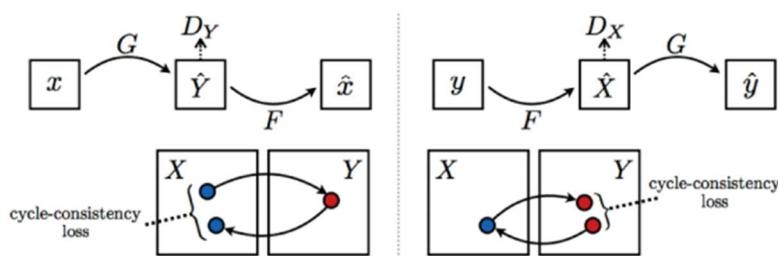


.Image-to-Image Translation - Pix2Pix

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 דedata מתחומים שונים. הארכיטקטורה עבורה המשימה הזאת מורכבת משני generators – בהתחלה מכנים דוגמא מהדומין הראשי x ל- G , G -generator. המוצא של G הוא דוגמא מהדומין השני y , והפלט נכנס ל- D_y discriminator השמנסה לשחרר את המקור x . המוצא של G הוא אכן לא רק ל- F אלא גם ל- D_y discriminator שנוועד ליזוחות האם התמונה שהתקבלה הינה אמיתית או לא (בעבור הדומין של y). ניתן לבצע את התהיליך זהה באופן דו-אי עבור y – מכנים אט y ל- F על מנת לקבל את x ואת המוצא ל- D_x discriminator בכדי לבצע סיוג בין-ארוי ו- G -על מנת לנסוט לשחרר את המקור. ה- G -generator השני F נדרש לשפר את תהליכי הלמידה – לאחר ש- x הופך ל- y , F , ניתן לקבל חזרה את x אם נعتبر את y דרך F מתוך ציפוייה לקבל $x \approx (G(y))$. התהיליך של השוואות הכנסה למושג *cycle consistency*, והוא מוסיף עוד איבר לפונקציית המבחן, שמטרתו לסייע עד כמה שניית את המהלך בין התמונה המקורית לתמונה המשוחזרת:

$$V(D_x, D_y, G, F) = \mathcal{L}_{GAN}(G, D_y, x, y) + \mathcal{L}_{GAN}(F, D_x, x, y) \\ + \lambda (\mathbb{E}_x \|F(G(x)) - x\|_1 + \mathbb{E}_y \|G(F(y)) - y\|_1)$$

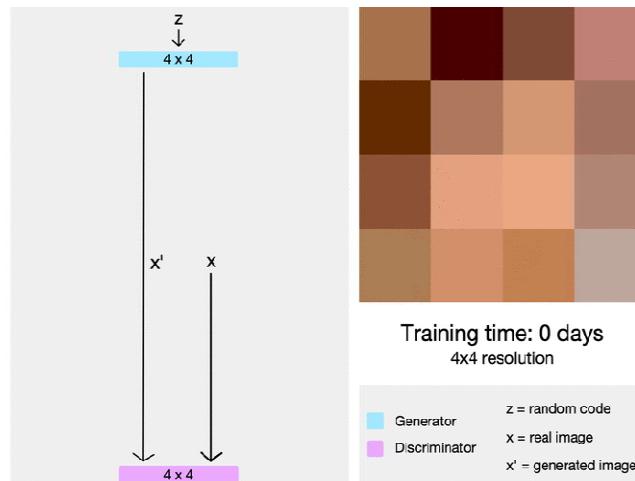


.CycleGAN

7.2.6 Progressively Growing GAN (ProGAN)

כאמור לעיל, עבור דומיין של תמונות, הגיוני להשמש ברשומות קונבולוצייה עבור יצירת תמונות חדשות, וזה הרעיון הבסיסי שמאחורי DCGAN. למרות יכולת המרשימה של DCGAN ביצירה של תמונה באיכות גבוהה, יכולת זאת מוגבלת לתמונות בגודל מסוים. ככל שהרזולוציה של התמונה גבוהה יותר, כך יותר קל להבחן אם התמונה זו אמיתית או נזרה על ידי רשת גנטית. בעוד ש-DCGAN מצליח ליצור תמונות שנראות אונטניות יותר, כמו למשל רזולוציה של 64×64 , 32×32 , 16×16 , 8×8 , 4×4 , והוא היה הראשון שפץ את מחסום הרזולוציה והצליח ליצור תמונות אינטלקטואליות מאוד (במאמר המקורי של ProGAN – עד רזולוציה של 1024×1024 בלי שהוא ניתן להבחן כמעט אליה סינטטיות. אולם עוד לפני ProGAN היו GANs שהצליחו ליצור תמונה בעלת רזולוציה גבוהה מהתמונה אחרת ברזולוציה גבוהה (4×4), אך זו ממשאה אחרת, מכיוון שבשבילה צריך רק ללמוד לשנות תכונות של תמונות קלות, ולא לייצר תמונה חדשה לגמרי מאפס.

הרעיון העיקרי מאחורי ProGAN, שהוצע ב-2017 על ידי חוקרים מחברת Nvidia, הינו לייצר תמונות ברזולוציה הולכת וגדרה בצורה הדרגתית. כלומר, במקרים לנסוט לאמן את כל השכבות של ה- G -generator בבת אחת, כפי שנעשה בכל ה-GANs לפניו כן, ניתן לאמן אותו לייצר תמונות ברזולוציה משתנה – בהתחלה הוא מתאמן לייצר תמונות ברזולוציות מאוד נמוכה (4×4), לאחר מכן המשיכו לייצר תמונות ברזולוציה 8×8 , אחר כן 16×16 וכך הלאה עד יצירה של תמונה ברזולוציה של 1024×1024 .



איור 7.11 ארכיטקטורת ProGAN

כדי לאמן GAN לייצר תמונות בגודל 4×4 , התמונות מסת האימון הוקטנו לגודל זה (down-sampling). אחרי שה-GAN לומד לייצר תמונות בגודל 4×4 , מוסיפים 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 משקלים שאומנו קודם לכן, אך לא "מקפיאים" אותו, ככלומר הם מעודכנים גם כן תוך כדי אימון הרשת בשבייל לייצור תמונה ברוחולציה כפולה. הגדלה הדרגתית של הרוחולציה מאלצת את הרשותות להתמקד תחילתה בפרטים ("הגים") של התמונה (דפוסים בתמונה מוטשטש מואוד). לאחר מכן הרשת "לומדת" לבצע sampling-קָפָה (להכפיל את הרוחולציה) של התמונות המוטשטשות האלה. תהליך זה משפר את יכולת התמונה הסופית כיון שבאופן זה הסבירות שהרשות תלמד דפוסים שגאים קטנה משמעותית.

7.2.7 StyleGAN

StyleGAN, שיצא בשלהי שנת 2018, מציע גרסה משודרגת של ProGAN, עם דגש על רשת-hs (generator). מחבריו המאמר שמו לב כי הינו הפטונציאלי של שכבות ProGAN המייצרות תמונה בצורה הדרגתית נובע מיכולתו לשנות בתכונות (מאפיינים) ויזואליות שונות של התמונה, אם משתמשים בהן כראוי. ככל שהשכבה והרוחולציה נمواה יותר, כך התכונות שהיא משפיעה עליה גסות יותר.

למעשה,_styleGAN הינו GAN הראשון שניתן יכולת לשנות באמצעות מאפיינים ויזואליים (אומנם לא בצורה מלאה) של התמונה הנוצרת. מחבריו StyleGAN חילקו את התכונות הויזואליות של תמונה ל-3 סוגים:

- **גס:** משפיע על תנוחה, סגנון שיער כליל, צורת פנים וכו'.
- **אמצעיות:** משפיעה על תוכני עדינים יותר, סגנון שיער, עיניים פקוחות/עצומות ועוד.
- **רוחולציה דקה:** משפיעה על צבע (עיניים/שיער/עור) ועל שאר תכונות המיקרו של תמונה.

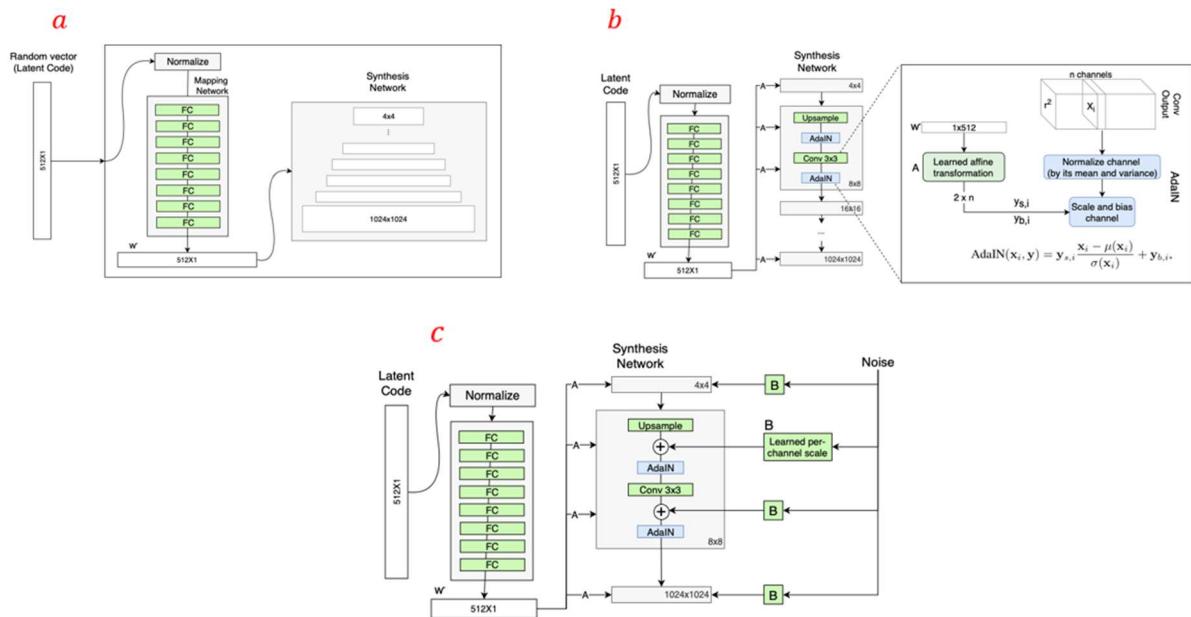
כדי להעניק ל-StyleGAN את יכולות האל, נדרש מספר שינויים ביחס לארכיטקטורה של ProGAN (נתאר רק את שלושת השינויים החשובים ביותר כאן):

- **הוספת רשת מיפוי:** מטרת רשת המיפוי היא קידוד וקטור הקלט לווקטור ביןים w (הנקרא וקטור סגנון) אשר האיברים השונים שלו שולטים בתכונות ויזואליות שונות של התמונה הנוצרת. זהו תהליך לא טריוויאלי מכיוון שהיכולת של הרשת לשנות בתכונות ויזואליות באמצעות וקטור הקלט הינה מוגבלת. הסיבה לכך טמונה בעובדה שווקטור הקלט נאלץ "עלוקוב אחר צפיפות ההסתברות של סט האימון" שגורם לתופעה הנקראת (FE) feature entanglement (-ערוב 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 הרשות 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 ותקבל תמונה של אישה.

רשת המיפוי שהנוספה לארכיטקטורה הופכת את וקטור הקלט לווקטור ביןים w שאינו צריך לעקוב אחר התפלגות של סט האימון, וכך יש פחות ערבוב המאפיינים. במקרים אחרים, רשת זו מאפשרת את יכולות לשנות באמצעות מאפיינים ויזואליים של התמונה הנוצרת באמצעות שניי רכיביו של וקטור w . רשת המיפוי מורכבת ממשמונה שכבות FC וגודל הפלט שלה זהה לגודל הקלט.

- **החלפת BN ב-AdaIN:** רשותות הקונבולוציה של ה-generator, שנעודו לייצור תמונות ברזולוציות 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-BN, במנגנון AdaIN סטנדרטי פרמטרים אלו נלמדים כמו המשקלים האחרים ולא תלויים במצבם של שכבה כלשהי.

- **ויתור על אקריאות של וקטור קלט:** ב-StyleGAN וקטור הקלט אינו וקטור המוגדר מהתפלגות גאוסית אלא וקטור דטרמיניסטי עם רכיבים נלמדים. וקטורי הרעש מתווספים ישירות לפלייטים של ערכיו קונבולוציה ברשתות ה-generator כאשר העוצמה שלהם נŁתן לכל ערך בנפרד. שימוש בוקטור קלט דטרמיניסטי במקומם בוקטור אקריאי מקל כל הנראה על הפרדת המאפיינים על ידי רשת המיפוי (ויתר קל לעשوت זאת על וקטור קבוע מאשר להתאים את משקליו רשת המיפוי לווקטורי כניסה אקריאים).



איור 7.12 השינויים העיקריים בארכיטקטורת StyleGAN. (a) שימוש ב-AdaIN. (b) שימוש ב-AdaIN בקלט דטרמיניסטי. (c) הוספה רשת מיפוי.

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 7.13 תמונות שיוצרו באמצעות StyleGAN.

7.2.8 Wasserstein GAN

הנחה היסודית ברוב המודלים האנרגטיביים, ובפרט ב-GANs, היא שהדעתה הרוב ממד' (למשל תמונה) "חיה" במשתח מממד נמוך בתוכו. אפשר להסכל על משטח בתור הכללה של ת-מרחב וקטורי מממד נמוך הנפרש על ידי ת-קבוצה של וקטורי בסיס של מרחב וקטורי מממד גובה יותר. גם המשטח נוצר מהת-קבוצה של וקטורי הבסיס של "מרחבי האם", אך ההבדל בין ת-מרחב וקטורי מתבבא בכך למשטח עשוי להיות צורה מאוד מורכבתיחסית לתת-מרחב וקטורי. משתמש מכך שניתן ליצור דעתה רוב ממד' על ידי טרנספורמציה של וקטורי מרחב בעל מממד נמוך (וקטור לטנטני). למשל, ניתן באמצעות רשת נירונים ליצור תמונה בגודל $k \times 3 > 64 \times 64$ פיקסלים מוקטור באורך 100 בלבד. זאת ועוד, שאם התפלגות התמונות של הרשות האנרגטיבית וגם ההתפלגות של הדעתה האמיתית נמצאים ב"משטח בעל מממד נמוך" בתוקן מרחב בעל מממד גובה של הדעתה המקורי. באופן פורמלי יותר, משטח זה נקרא ירעה (manifold), וההשערה שתוארה מעלה מהווה הנחת יסוד בתחום הנקרא למידת יריעות (manifold learning). מכיוון שמדובר במשטחים בעלי מממד נמוך בתוקן מרחב בעל מממד גובה, קיימות סבירות גבואה שלא יהיא שום חיתוך בין המשטח בו "חיה" הדעתה האמיתית לבין זה של הדעתה הסינטטי (לכל הפחות בתחלת תהליכי האימון של GAN), ויתרה מכך, המרחק בין משטחים אלה עשוי להיות די גדול. מכך נובע שה-discriminator D עשוי ללמוד להבחין בין הדעתה האמיתית לסינטטי בקלות, בעוד שבמרחב מממד גובה יש מרחק גדול בין ירעה אמיתית לבין הירעה של הדעתה הסינטטי. בנוסף, D כנראה יתנו לדוגמאות סינטטיות ציונים (score) ממש קרובים לאפס כי אכן קל למצוא "משטח הפרדה" בין שתי היריעות – זה של הדוגמאות האמיתיות וזה של הסינטטיות, כיוון שהם נתונים להיות רוחקים מאוד אחד מהשני.

ሩק זה מסיע להבין מדוע הפער שיש בין generator וה-discriminator מבחן אווי הלמידה מהווע בעיה. כאמור, ה-generator מעדק את המשקלים שלו על סמך הציונים שהוא מקבל מה-discriminator (דרך פונקציית discriminator של ב-GAN). אבל אם ה-generator מוציא ציונים מאד נמוכים (עקב מרחק גדול בין היריעות שתואר לעלה) לדוגמאות המיצירות על ידי ה-generator, ה-generator פשוט לא יכול לשפר את איכות התמונהות שהוא מייצר. במיללים, D "פושט הרבה יותר מדי טוב יחסית ל-G". אתגר זה בא לידי ביטוי גם במצבה של פונקציית המבחן, שלא מאפשרת "הברחה עיליה של ידי" מה-discriminator-generator ל-discriminator.

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 המחיר. השיטות הבולטות הן:

- התאמת פיצרים (feature matching)
 - .minibatch discrimination
 - .virtual batch normalization
 - מיצוע היסטורי

כפי שהסביר, הבעיה של המרחק בין היריעות משתקפת במבנה של פונקציית המחר, וכךון שכר, ניתן לנוסות ולפטור את הבעיה מהשורש על ידי שימוש בפונקציית מחיר יותר מתאימה. לשם כך ראשית נסמן את התפלגות הדטה האמיתית ב- r_{-k} , ואת התפלגות הדטה הסינטטי המיצר על ידי ה-generator ב- p_{-g} . לעיל הראיינו שפונקציית המחר Jensen-Shannon divergence – \mathcal{D}_{JS} – המציגת את המרחק בין התפלגיות על ידי

ניתן להוכיח כי מרחק D_{JS} בין התפלגיות p_g , p_r לא רגיש לשינויים ב- p_g כאשר המשטחים שביהם "חיים" p_r ו- p_g רחוקים אחד מהשני. לעומת זאת, מרחק D_{JS} כמעט ולא ישתנה אחרי עדכון המשקלים של ה-generator, וממילא לא ישקוף את המרחק המעודכן בין שתי התפלגיות p_r ו- p_g . זו למעשה הבעיה המהותית ביותר עם פונקציית המחריר המקורית של ה-generator, שעדכון המשקלים לא משפיע כמעט על D_{JS} , בעוד שטראס התפלגיות p_r ו- p_g רחוקות כמעט מרבשות.

באותו זמן, הוצג מושג Wasserstein GAN המשמש בפונקציית מחיר אחרת, בה עדכון המשקלים generator בין התפלגיות p ו- q . פונקציית המחר החדשה מבוססת על מרחק הנקריא (EM), המהווה מקרה פרטי של מרחק וורשטיין המשומן ב- \mathcal{W}_p . מרחק וורשטיין 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 - M) עם פונקציות שליליות (marginal) השווות ל- μ ו- ν בהתאם. תחת סימן האינטגרל יש את המרחק האוקלידי מסדר p בין הנקריא. מרחק EM הינו מקרה פרטי של מרחק וורשטיין, כאשר $p = 1$, ובאופן מפורש:

$$EM = W_1(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \int_{M \times M} d(x, y) d\gamma(x, y)$$

הגדרה זו נראהת מאוד מסובכת וננסת לתת עברורה אינטואיציה, ולהבין מדוע עבור $p = 1$, מרחק וורשטיין נקרא EM. לשם הפשטות נניח שהמרחב M הינו חד ממדי, כלומר x , ועליו עשר 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 נראה ש为了让 x להיות משקל של 0.3_{kg} בנקודת $x = 4$, ישנו שפה שתהינה מפוזרות באופן הבא: בנקודת $x = 4$ יהיה משקל של 0.3_{kg} , בנקודת $x = 5$ יהיה משקל של 0.5_{kg} , ושאר המשקלות (0.2_{kg}) יהיו בנקודת $x = 8$.

כמובן שיש הרבה דרכים לבצע את היזמת המשקלות, ונרצה למצוא את הדרך היעילה ביותר. לשם כך נגידיר 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 היזמת המשקלות באופן הבא:

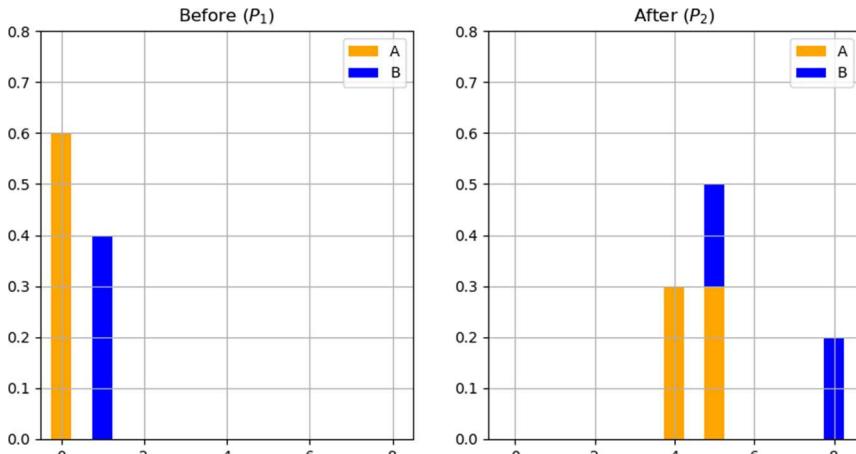
$$\text{מועברים מ-0} = x = 4 - 0 = 4, \text{ כאשר המאמץ הנדרש לכך הינו } 0.3_{kg}.$$

$$\text{מועברים מ-0} = x = 5 - 0 = 5, \text{ כאשר המאמץ הנדרש לכך הינו } 0.3_{kg}.$$

$$\text{מועברים מ-1} = x = 5 - 1 = 4, \text{ כאשר המאמץ הנדרש לכך הינו } 0.2_{kg}.$$

$$\text{מועברים מ-1} = x = 8 - 1 = 7, \text{ כאשר המאמץ הנדרש לכך הינו } 0.2_{kg}.$$

$$\text{סך המאמץ המינימלי שווה במקרה זה ל: } 4 + 5 + 4 = 13.$$



איור 7.14 העברת משקלות באופן אופטימלי. P_1 מייצג את המצב ההתחלתי, ו- P_2 הינו המצב לאחר היזמת המשקלות.

כעת, במקום להסתכל על משקלים, נתיחס להתפלגיות 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 התפלגות \mathcal{D}_g , שקופה לדוגמא של הזרת המשקלות. מרחק EM בין שתי התפלגות \mathcal{D}_g ו- p , מוגדר להיות ה"מאז" המינימלי הנדרש בשבייל להעביר את המסמה ההסתברותית מ- p ל- \mathcal{D}_g , או במילים אחרות – מרחק EM מגדר מהי כמות ה"עבודה" (מאז) המינימלית הנדרשת בשבייל להפוך p ל- \mathcal{D}_g . אם נחזור לדוגמא של המשקלות, נוכל להבין מדוע \mathcal{D}_g עבור $1 = p$ נקרא מרחק Earth Mover – מרחק בין שתי התפלגות שקול לכמה מאז נדרש להעביר כמות אדמה במשקל מסוים כדי לעבור מחלקה מסוימת של אדמה לחולקה אחרת. באופן יותר פורמלי – מידת ההסתברות על מרחב המכפלה בנוסחה של מרחק EM מתארת את האופן שבו אנחנו מעבירים את המסמה ההסתברותית (משקל מסוים של אדמה), כאשר הביטוי (y, x) מציין כמה מסה הסתברותית מועברת מנוקודה x לנוקודה y .

לאחר שהסביר מהו מרחק ורטשטיין \mathcal{D}_W ומהו מרחק EM, ניתן להבין כיצד אפשר להשתמש במקרים אלו עבור פונקציית מחיר של GAN. נציין כי מרחק \mathcal{D}_W בין מידות ההסתברות מתחשב בתכונות של הקבוצות עליהם מידות אלו מוגדרות בצורה מפורשת, על ידי המתחשבים בקבוצות אלו. תכוונה זו היא למעשה בדיקת מה שzier בשבייל למדוד את המרחק בין התפלגות האמיתית של דאטה p לבין התפלגות של הדטה הסינטטי \mathcal{D}_g . מרחק EM ייעד לשערך בצורה טוביה את המרחק בין היריעות שבנה "חוות" שתי התפלגות, ככלומר אם מזינים את הירעה של הדטה הסינטטי, נוכל לדעת בערך מרחק EM עד כמה השתנה המרחק בין היריעות. נציין שזה לא קורה כאשר משתמשים בפונקציית המחיר המקורי הננדדת באמצעות \mathcal{D}_{JS} . לעומת זאת, בערך פונקציית המחיר החדשה המבוססת על מרחק EM, ניתן לדעת עד כמה עדכון המשקלים מקרוב או מרחק את \mathcal{D}_g מ- p .

באופן תיאורתי זהמצוין, אך עדין זה לא מספיק, כיון שצריך למצוא דרך לחשב את \mathcal{D}_W , או לכל הפחות את המקהלה הפרטני שלו עבור $1 = p$, ככלומר את מרחק EM. במקור מרחק זה מוגדר כבעית אופטימיזציה של מידות הסתברות על מרחב המכפלה, וצריך למצוא דרך להשתמש בו כפונקציית מחיר. בשיביל לבצע זאת, ניתן להשתמש בצורה דואלית של \mathcal{D}_W עבור $1 = p$ – שיווין RK (Rubinstein-Kantorovich) לערך נטען לחשב את $1 = p$ – שיווין RK (Rubinstein-Kantorovich) לערך נטען לחשב את המרחק בין התפלגותים 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). כתת נניח ש-(x, f) הינה פונקציית K-לייפשיץ רציפה המתארת discriminator בעל סט הפרמטרים w . ה-z-generator מחשב באופן מקרוב את המרחק בין התפלגותים באופן הבא:

$$L(p(r), p(g)) = \max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))]$$

פונקציית מחיר זו מודדת את המרחק \mathcal{D}_W בין התפלגות \mathcal{D}_g ו- p , וככל שפונקציה זו תקבל 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 לממוד פונקציית K-לייפשיץ רציפה המודדת את \mathcal{D}_W בין התפלגותים p_g ו- p_r . ה-z-generator L לעומת זאת מאמין למזרע את (p_g, p_r) (L כאשר רק האיבר השני שתלי ב- θ), וככל שפונקציית המחיר הולכת וקטנה, כך g מתקרוב יותר ל- p_r .

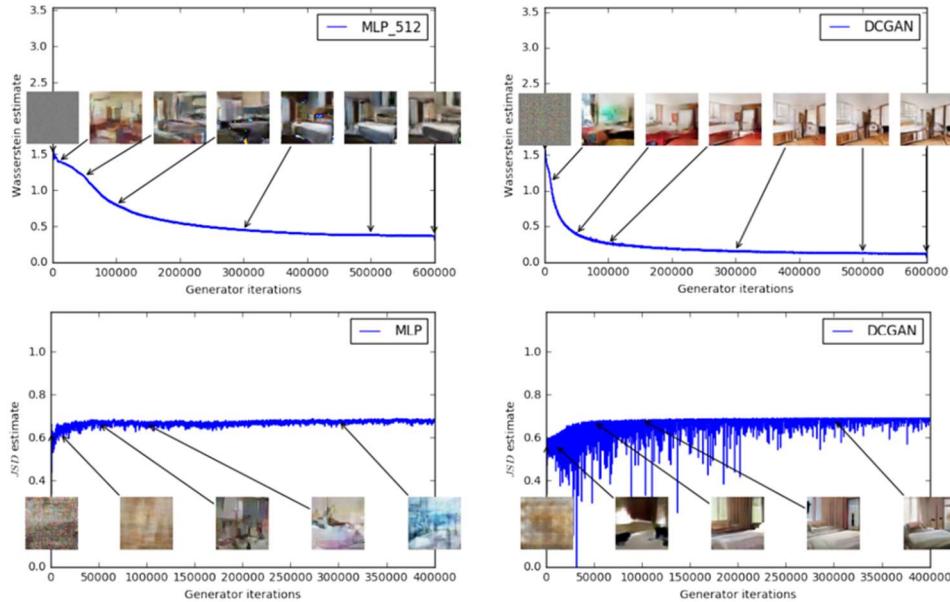
כאמור, תנאי הכרחי לשימוש במרקח זה בפונקציית המחיר הינו שהפונקציה תהיה K-לייפשיץ רציפה. מסתבר שקיים תנאי זה אינו ממשה קלה כל. כדי להבטיח את קיומו, המאמר המוקורי הצע לבעץ קטימה של משקלי ה-discriminator לטוויה סופי מסוים, נניח $[0.01, 0.01]$. ניתן להראות כי קטימה זו מבטיחה את ש- \mathcal{D}_W ה-לייפשיץ רציפה. אולם, כמו שכותבי המאמר מודים בעצמם, ביצוע קטימה בכדי לדאוג לקיום תנאי לייפשיץ יכול לגורום לבעיות 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, ההתכנסות עלולה להיות מאוד איטית. נציין שיש עוד מספר דרכים לכפות על f להיות לייפשיץ-רציפה למשל gradient penalty.

הإيمان של Wasserstein GAN דומה לאימן של ה-GAN המקורי, למעט שני הבדלים עיקריים:

- א. קיצוץ טווח המשקלים על מנת לשמור על רציפות-לייפשיץ.
- ב. פונקציית מחיר המסתמכת על \mathcal{D}_W במקום על \mathcal{D}_{JS} .

תהליכי הלמידה מתבצע באופן הבא – לאחר כל עדכון משקלים של ה-z-generator (gradient ascent) (באמצעות discriminator), מוצאים את טווח המשקלים. לאחר מכן מבצעים עדכון רגיל של משקלי ה-z-generator (gradient descent).gradient descent

Wasserstein GAN מצליח לגרום 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 היבט באמצעות גרפים הבוחנים את הייחס בין \mathcal{D}_W לבין \mathcal{D}_{JS} .



איור 7.15 שערור מרחק W בין \mathcal{D}_g ל- \mathcal{D}_D כפונקציה של מספר האיטרציות (בגרפים העליונים), לעומת שערור מרחק JS בין \mathcal{D}_g ל- \mathcal{D}_D כפונקציה של מספר האיטרציות (בגרפים התחתונים).

ניתן לראות בבירור כי ככל ששיעור התמונה שה-generator מיציר עולה, כך \mathcal{D}_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.

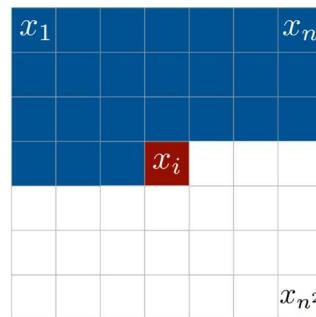
נוקודה נוספת שיכולה להסביר את ההצלחה הייחסית של השימוש ב- \mathcal{D}_D נובעת מכך שמטריקת JS חלשה יחסית למטריקת JS , וננסה להבהיר נוקודה זו.

באופן אידיאלי, כאשר אנו מאמנים מודל הינו רצים להיות בטוחים שאם ננגב בצורה נאותה ובכל צעד נעדקן המודל בדיק על פי הוראות הגרדיינט, נסימם את האימון בנקודה כמעט ללא אופטימיליזציה. אולם בפועל זה לא תמיד כך, כיון שישנן בעיות שעבורן מטריקות מסוימות יגיעו לנקודת קומפקט זיהו ואחרות לא. ניקח לדוגמא שני אנשים שעומדים על סף תחום ורוצחים להגיע עמוק. האחד מodd את הגובה ומתקדם על פי, ולכן הוא יגיע למיטה בكلות ייחסית. الآخر מתענין במיקומו על ציר צפוני דרום, ולכן הוא עשוי להגיע רק בשיטות 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 מהמטריקה השנייה. העובדה ש- W_{JS} חלש יותר מ- JS_W בעצם אומרת שיתכן ויש בעיות שעבורן תתקבל תוצאה אופטימלית עבור W_{JS} אך לא עבור JS_W .

7.3 Auto-Regressive Generative Models

משפחה נוספת של מודלים גנרטיביים נקראת Auto-Regressive Generative Models, ובודומה ל-VAE גם מודלים אלו מוצאים התפלגות מפוששת 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 המדוייקת 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 מכפלה הастברויות 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 הастברות של DATA מסוימת לייצג תמונה אמיתית, אך נרצה למקסם את הביטוי הזה כדי לקבל מודל שמייצג תמונות שנראות אוטנטיות עד כמה שניתן.

7.3.1 PixelRNN

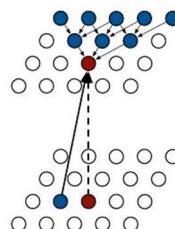
אפשרות אחת לחשב את (x) היא להשתמש ברכיבי זיכרון כמו LSTM עבור כל פיקסל. באופן טבעי היינו רצים לחבר כל פיקסל לשכנים שלו:

$$\text{Hidden State } (i,j) = f(\text{Hidden State } (i-1, j), \text{Hidden State } (i, j-1))$$

הבעיה בחישוב זה היא הזמן שהוקח לבצע אותו. כיוון שכל פיקסל דרוש לדעת את הפיקסל שלפניו – לא ניתן לבצעAIMON מקבילי לרכיבי-h-TM. כדי להתגבר על בעיה זו הוצעו כמה שיטות שונות לאפשר חישוב מקבילי.

Row LSTM

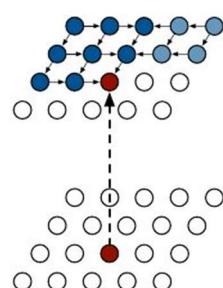
במקרה להשתמש במצב החבוי של הפיקסל הקודם, ניתן להשתמש רק בשורה שמעל הפיקסל אותו רצים לחשב. שורה זו עצמה מחושבת לפניה על ידי השורה שמעליה, ובכך למעשה לכל פיקסל יש receptive field של מושולש. בשיטה זו ניתן לחשב באופן מקבילי כל שורה בנפרד, אך יש לכך מחיר של איבוד הקשר בין פיקסלים באותה שורה (loss context).



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 בעזרת 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.

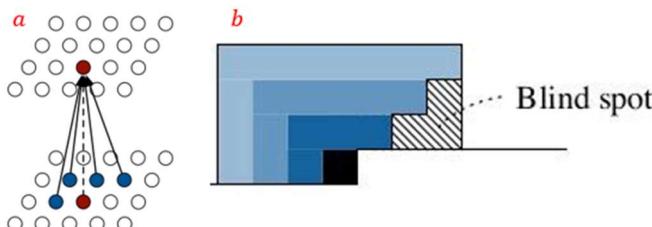


איור 7.17 Diagonal BLSTM – כל פיקסל מחושב על ידי $3 \geq k$ פיקסלים בשורה שמעליו.

כדי לשפר את השיטות שימושות ברכיבי זיכרון ניתן להוסיף עוד שכבות, כמו למשל residual blocks שעזרים להאייש את התחכחות - Masked convolutions כדי להפריד את התלות של העורכים השונים של כל פיקסל.

7.3.2 PixelCNN

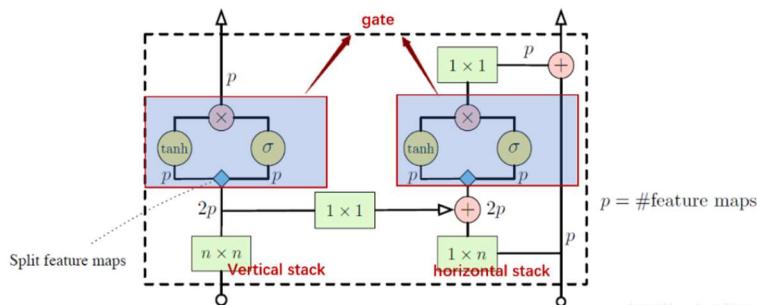
היחסון העיקרי של PixelRNN נובע מהaimon האיטי שלו. מקום רכיבי זיכרון ניתן להשתמש ברשף קונבולוציה, ובכך להאיץ את תהליך הלמידה ולהגדיל את ה-*receptive field*. גם בשיטה זו מתחילה מהפיקסל הפנימי, רק-cutת הלמידה היא לא בעזרת רכיבי זיכרון אלא באמצעות שכבות קונבולוציה. היתרונות של שיטה זו על פני PixelRNN מתבטאת בקיצור משמעותי של תהליכי האימון, אך התוצאות פחות טובות. יחסי נסוף בשיטה זו נובע מהמבנה של המסננים ו-*receptive field* – כל פיקסל מtabסס 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-*field* *receptive*.



איור 7.18 receptive field של PixelCNN (a) והיסטרוגרף של PixelCNN (b).

7.3.3 Gated PixelCNN

בכדי להתגבר על בעיות אלו – ביצועים לא מספיק טובים והתעלמות מפיקסלים יחסית קרובים שאינם ב-field – נעשה שימוש ברכיב זיכרון הדומה ל-LSTM, המשלב את רשותות הקונבולוציה בתוך RNN.



איור 7.19 שכבה של Gated PixelCNN

כל ריבוי זיכרון בניי משני חלקים – horizontal stack and vertical stack – כאשר כל אחד מהם הוא למעשה שכבת konvolוציה. ה-stack vertical stack בניי מזכירן של כל השורות שהו עד כה בתמונה, וה-stack horizontal stack הוא מסנן יחיד על הקטל הנוכח. ה-stack horizontal stack עובר דרך שער של אקטיבציות לא לינאריות ובנוסף מתחבר ל-stack vertical על הקטל הנוכחי. פנוי כל כניסה של stack לתוך שער, אשר גם החיבור בינוים עובר דרך שער של אקטיבציות לא לינאריות. לפניו כל כניסה של stack לתוכה שער,

$$\gamma = \tanh(w_f * x) \odot \sigma(w_a * x)$$

7.3.4 PixelCnn++

שיפור אחר של CNN PixelOpen הוצע על ידי AIOpen. והוא מביאס על מספר מודלים קיימים:

- SoftMax שקובעת את צבע הפיקסל המרכזי הרובה צ'িקרן, כיוון שיש הרבה צבעים אפשריים. בנוסח, היא גורמת לארדייאנט להתאפס מהר. כדי להתגבר על כך ניתן לבצע דיסקרטיזציה לצבעים, ולאפשר טווח צבעים קטן יותר. באופן זהה קל יותר לקבוע את ערכו של כל פיקסל, ובנוסח תהליך האימון יותר יעיל.
 - במקום לבצע בכל פיקסל את ההתניה על כל צבע בנפרד (כפי שהראינו בפתחה), ניתן לבצע את ההתניה 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 לשומר 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

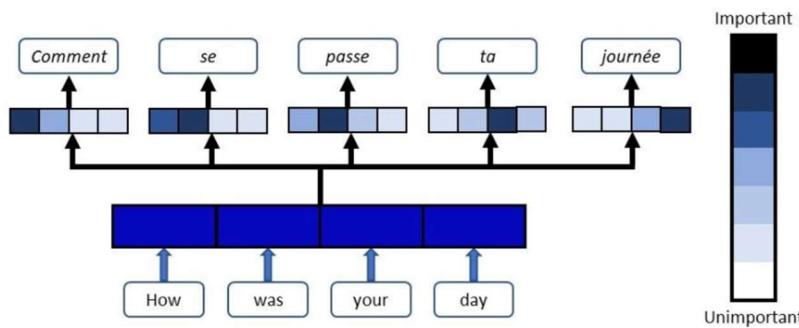
8. Attention Mechanism

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. ברשותה 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



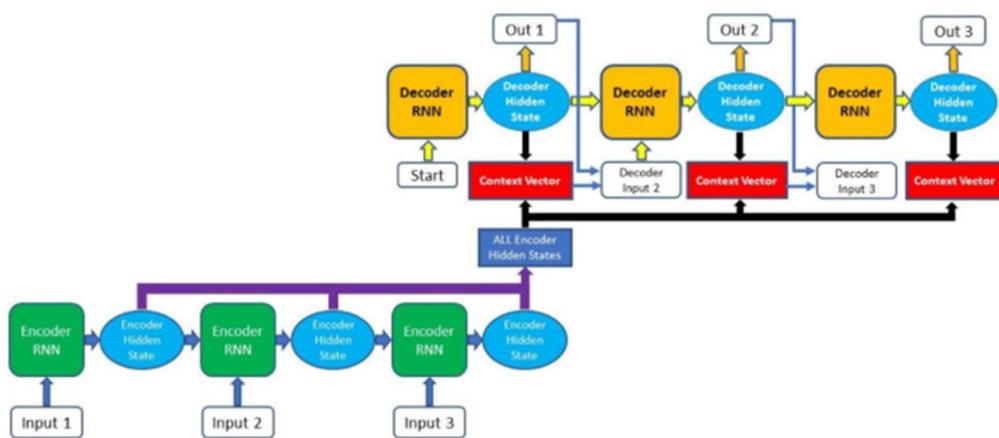
איור 8.1 מנגנון attention – נתינת משקל לכל אחת ממלילות הקלט בהתאם לכל אחת ממלילות הפלט.

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 במנגנון זה תוך קבלת ביצועים מעולים.

בחלק זה יוצגו היחסות המשלבות בין רשות NN ל-attention ולאחר מכן יוסבר על ה-transformer שמשתמש ב-self-attention וב-position encoding self-attention.

8.1.2 Bahdanau Attention and Luong Attention

.Dzmitry Bahdanau על שם המציגו שלה – Bahdanau Attention הגישה הראשונה שהוצעה נקראה



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 משקלן 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 לפלט, 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 score של האיבר הקודם ב-decoder, 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-decoder וה-encoder,alignment score יתקבל על ידי:

$$\text{alignment score} = w_{\text{alignment}} \times \tanh(w_d H_d + w_e H_e)$$

כאשר w_d , w_e הם המשקלים הנלמדים של ה-encoder וה-decoder בהתאמה. את התוצאה מعتبرים דרך SoftMax(H_e) ומתקבלים את ה-context vector:

$$\text{context vector} = H_e \times \text{SoftMax}(\text{alignment score})$$

הווקטור המתתקבל מכיל משקל של כל אחד מאיברי הפלט בהתאם לאיבר הפלט הנוכחי. את התוצאה כאמור מחברים לפלט 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-attention score.

8.2 Transformer

לאחר שמנגנון ה-attention התחיל לצבור תאוצה, הומצאה ארכיטקטורת המבוססת על חיבור 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 חישוב 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 $\mathbb{R}^d \in x$, מחשבים וקטור מממד $1 \times d$ באופן הבא:

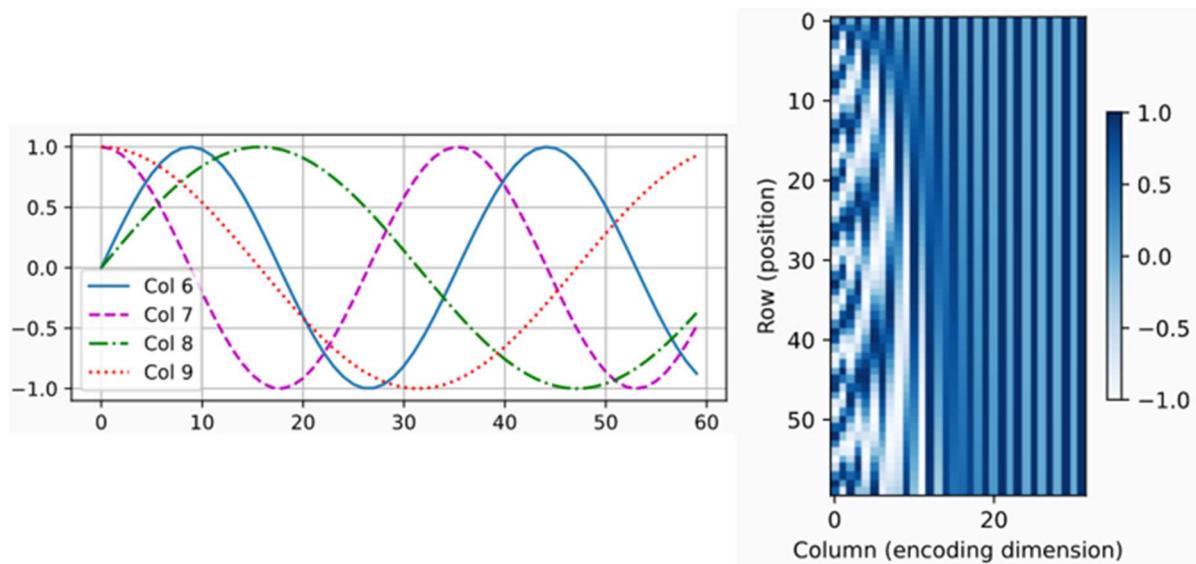
$$p_t(i) = \begin{cases} \sin(\omega_k t), & i \text{ is even} \\ \cos(\omega_k t), & i \text{ is odd} \end{cases}, \omega_k = \frac{1}{10000^{\frac{2k}{d}}} \rightarrow p_t = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \vdots \\ \sin\left(\omega_{\frac{d}{2}} t\right) \\ \cos\left(\omega_{\frac{d}{2}} t\right) \end{bmatrix}_{d \times 1}$$

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 אותו 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 הכח נמוכה, ואילו ה-LSB משתנה בתדרות הכח 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 \vec{u} – הוא מכיל הרבה פונקציות טריגונומטריות בעלות תדרות הולכת וקטנה, ולפי התדרות שמתואספת 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 (שמאל). 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 לאיל האיברים בסדרה. איברים דומים (קרובים) בסדרה יקבלו ערכי דמיון גבוהים, ואילו איברים שונים (רחוקים) בסדרה ינתנו ערכים נמוכים (ב- BERT זה יכול להיות מילים שסביר שיפויו בסמכות, 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, וכך ניתן לסכם את מכפלת כל המוקדים באיברים המקוריים, ולקבל ייצוג חדש לאיבר המקורי.

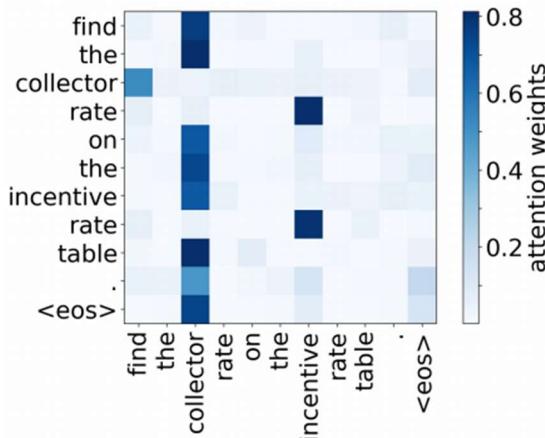
המכיל גם קשר בין האיבר הנוכחי לאיברים דומים בסדרה. במיילים אחרים, ניתן להסתכל על וקטור המכיל את הקשרים של איבר מסוים בסדרה כדי החדש המשקף את הקשרו עם שאר איברי הסדרה.

באופן פורמלי, בשביל לחשב את self-attention מטריצות של מקדים עבור סדרת הכנסה. המטריצות 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 $x_i \cdot Q = q_i$, ומכללים 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 ההכפלה זו עבור כל סדרת הקלט יוצר מטריצה חדשה בה כל שורה מייצגת את הקשר בין איבר מסוים לשאר איברי הסדרה. ההכפלה זו היא בעצם $K \cdot Q$, אשר 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 SoftMax. באופן זהה מקבלים מטריצה של מספרים בטוחווים $[0, 1]$, המייצגים כאמור את הקשר בין כל שני איברים בסדרה המקורי. נסמן כל איבר במטריצה ב- w_{ij} , ונוכל לקבל אותו יישור על ידי הנוסחה:

$$w_{ij} = \text{SoftMax}\left(\frac{q_i \cdot k_j}{\sqrt{d_k}}\right) = \frac{\exp\left(\frac{q_i^T k_j}{\sqrt{d_k}}\right)}{\sum_{s=1}^n \exp\left(\frac{q_i^T k_s}{\sqrt{d_k}}\right)}$$

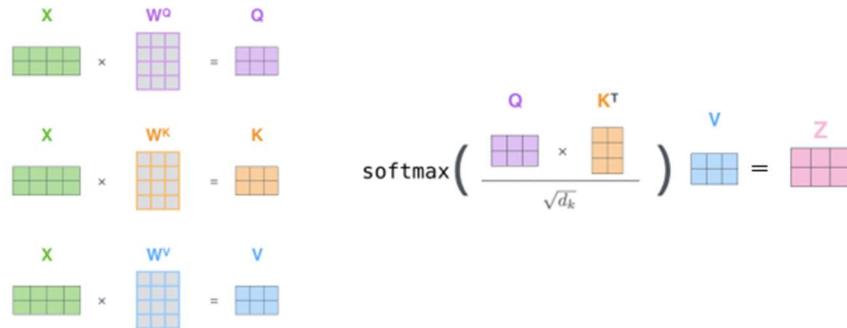


איור 8.5 מטריצת משקלים של המשפט "Find the collector rate on the incentive rate table". כל קשר בין שתי מיללים חזק יותר-כך המשקל ביניהם גבוה יותר. כמו כן שיש גם שימוש לסדר – המשקל בין "collector" ל-"Find" שונה מאשר בין "collector" ל-"Find".

כעת בעזרה משקלים אלו בונים ייצוג חדש לסדרה המקורי, על ידי הכפלתם בוקטור v :

$$z_i = \sum_{j=1}^n w_{ij} v_j = \frac{\sum_{j=1}^n \exp(q_i^T k_j)}{\sum_{s=1}^n \exp(q_i^T k_s)} v_j$$

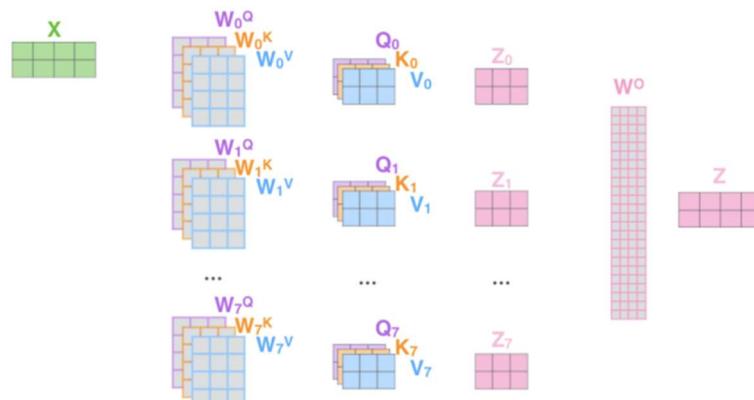
סדרה המתקבלת z היא למעשה ייצוג חדש של סדרה המקורי, כאשר כל איבר z_i מייצג איבר בסדרה המקורי יחד עם מידע על הקשרים ביןו לבין שאר איברי הסדרה. את סדרה המתקבלת ניתן להעביר ב-decoder המכיל שכבות נוספות, ובכך לבצע כל מיני משימות, כפי שיואר בהמשך.



איור 8.6 ביצוע Self-attention – ייצור מטריצות Query, Key, Value (שמאל) וחישוב ה-score (ימין).

8.2.3 Multi Head Attention

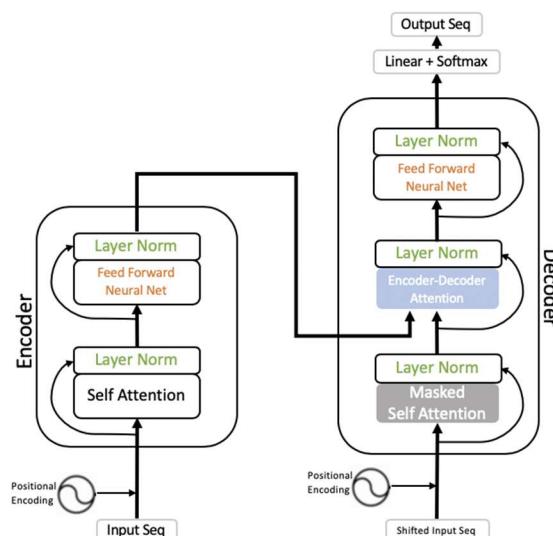
ניתן להשתמש במנגנון self-attention מספר פעמים במקביל. כל פעם מקבלים שלוש מטריצות (Q_r, K_r, V_r), ובעזרתهما מחשבים את הייצוגים החדשניים של איברי הסדרה (attention score). כל מנגנון זה נקרא attention head, וסביר במקביל של כמה attention heads נקרא Multi-head attention. באופן זה לכל איבר כניסה x_i יש כמה "יצוגים שונים" Z_{ir} , אותן ניתן להכפיל במטריצת משקלים W^o 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 מנגוני 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 סדרה מסוימת x (לרוב אחרי שבעריה embedding מסוים) ומבצע עלייה positional encoding.layer 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 שכבה fully connected ומשם יצא הפלט לכוון ה-decoder.

ה-decoder בניו בדומה מודול דומה, עם שני הבדלים העיקריים: הקלט שלו הוא איברי הפלט שעדיין לא הגיעוencoder self-attention שכבה של decoder. שכבה זו מקבלת את כל איברי הפלט שעדיין לא הגיעו, ומטרת 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: המטריצות מה-encoder וה-decoder. השכבה שילוב של decoderencoder self-attention את המכפלה $K \cdot Q$, נלקחות מה-encoder. כתע כשמבאים דמיון בין איברים של decoder סדרת הפלט (ביצוגם שלם לאחר ה-decoder) לבין איברי סדרת הקלט (ביצוגם שלם לאחר שכבת masked-Q). בשלב זה דומה מאוד למטריצת המוקורי, רק שהיא ייצוגים שהתקבלו לא נעזרים ברכיבי זיכרון. כאמור, המכפלה $K \cdot Q$ מייצרת מטריצת משקלים שכל איבר בה אומר מה היחס בין איבר בסדרה המקורית לבין איבר בסדרת הפלט. את המטריצה הזאת מכפילים ב-V, וקטור מסויים שהוא ייצוג חדש של איבר הפלט הבא. וקטור זה עובר בשכבות FC ו-SoftMax, וכך מקבל 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 המכפלה $K \cdot Q$). באופן זה ניתן למצוא אזוריים דומים ושונים בתמונה, כאשר דמיון ושווא זה לאו דוקא 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. בשלב זה למעשה אפשר לבצע זיהוי של אובייקטים, בלי לדעת מה הם אותם אובייקטים. בשביל לבצע סיוג כל אובייקט שゾהה, מעבירים את הייצוג החדש של התמונה ב-decoder, וכך ה-Query שמנכנים זה כל מיני ידיילים אפשריים, ומוחפשים מבין כל ה-Query את הפלט של decoder שמצליח ליצור תמונה שהכי דומה ל-Query.

אם למשל יש תמונה גדולה ויש אזכור מסוים בו יש חתול, אז ה-encoder מוצא איפה החתול בתמונה, וה-decoder משווה את האזכור הזה לכל מיני חיות אפשריות. כל Query שלא היה חתול, המכפלה $K \cdot Q$ תהיה קרובה ל-0, וה-decoder יזהה שה-Query הנוכחה לא תואם לאובייקט שゾהה. אך כאשר ה-Query היה חתול, אז כמובן ש- $K \cdot Q$ אחד לשני, המכפלה $K \cdot Q$ תביא לכך שהייצוג החדש $\sum_{j=1}^n z_j w_{ij}$ קיימת 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 במגוון שימושות, והוא הינו הרשאה להמון השימושים הנשענים על attention בלבד. מלבד הרשאה 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 ללקחת transformer שאותן על משימה מסוימת, ולהתאים אותו לשימוש חדשה שדומה למשימה המקורית. בפועל לא כל היחסים משתמשים בכל ה-transformer, אלא בהתאם לשימושם 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 self-attention. היאלקח משפט ולהוציא משפט בשפה אחרת, וזה נעשה בעזרת ייצוג המשפט המקורי באמצעות Encoder-Decoder Attention לשפה אחרת.

Bidirectional Encoder Representations from Transformers (BERT) – מודל שפה מבוסס encoder. מודל שפה פונקציית המקבל קלט טקסט ומחזירה את ההתפלגות למילה הבאה על פי כל המילים במילון. השימוש הכי מוכר ואינטואיטיבי 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, למשה מקרים ייצוגים חדשים שלהם יחד עם הקשרים בין המילים השונים. לכן ה-transformer encoder ב-BERTencoder מביא מילון קונטקט שלם יותר. בנוסף, אם מאמנים אותו בצורה מתאימה. המפתחים של BERTencoder מביא מילון משפטים בשני כיוונים – גם מההתחלת לסופו וגם מהסוף להתחלה, וכך הייצוגים שנלמדו קיבלו קונטקט שלם יותר. בנוסף, הם אימנו את המודל על משפטים בהם כל פעם באופן רנדומלי עושים למלילים מסוימות, ומטרת המודל הוא לחזות את המילים החסרות.

מהי המילה הבאה באמצעות decoder בלבד. מכניסים משפט קצרוע באמצעותו במאצע, ולבוחן את ההתאמנה שלהן למשפט הנתון, והמילה שהכי מתאימה נבחרת להיות המילה הבאה. המשפט הקטוע הוא למעשה ה-**Query**, והוא-Key

למעשה ה-**Query**, וה-**Key** שנכנסו הוא כל פעם מילה אחרת במילון, וכך בעזרת attention בוחנים איזה מילים יתאים בצורה הטובה ביותר ל-**Key** 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

9.1 Object Detection

9.1.1 Introduction to Object Detection

זהוי אובייקטים היא משימה מאוד נפוצה בעולם של ראייה ממוחשבת, ויש לה המונ "ישומים מגוונים". ניקח למשל מכונית אוטונומית, שבכל רגע צריכה לזהות את האובייקטים שסביבה ולקבל תמונה מצב עדכנית על המתרחש, או לחולופן מצלמה של טלפון נייד ש יודעת לזהות פנים של בנאדם בצד' לבצע עליהם פוקוס או לתזק רעש רקע, ועוד המונ "ישומים נרחבים בכל מיני תחומים".

בשלב ראשון יש להגדיר באופן מדויק את המשימה – מה הכוונה לזהות אובייקט בתמונה? יש כמה רמות שונות של זהוי. המשימה הקלואית של סיווג (classification) מניחה שיש אובייקט יחיד בתמונה, והמטרה היא לסווג אותו בצורה נכונה, כלומר לקבוע מה-class שלו. משימה יותר מתקדמת היא לא רק לומר איזה אובייקט נמצא בתמונה, אלא גם לומר איפה בדיקו הוא נמצא. גם פה יש שתי רמות – ניתן לסמן את האזור בו הוא נמצא בערך מלבד (bbox) שמקיף את השולדים שלו, ונitin לסווג כל פיקסל בפני עצמו האם הוא שייך לאובייקט או לרקע. זהוי מושג הראשון נכנס תחת התחום של **Object Detection** ואילו זהוי סמנטי של פיקסלים נכנס תחת התחום של **Segmentation**.

ניתן להכליל את משימת זהוי גם ל-**multiple object**. כמובן, יש מספר לא ידוע של אובייקטים בתמונה, והמטרה היא למצוא היכן הם נמצאים ולסווג כל אחד מהם ל-class המתאים. בעצם משימה זו מורכבת משתי תתי משימות – מציאת המיקום של האובייקט (Localization/Regression) וסיווג האובייקט ל-class הנכון (Classification). נשים לב שעבור משימת Object Detection, ההכללה מאובייקט יחיד למספר אובייקטים הינה ישירה – על המודל לספק מספר bounding boxes ועבור כל אחד מהם להתאים class. במשימת Segmentation לעומת זאת, ההכללה למספר אובייקטים יכולה להיעשות בשתי דרכי: 1) Semantic Segmentation – שיר כל פיקסל ל-class מסוים ללא הבחנה בין פיקסליהם השונים לאובייקטים שונים בעלי אותו class. במקרה זה אם יהיו שני כלבים בתמונה, אז במעט הסגמנטציה נראה הרבה פיקסליהם המשוכרים ל-class של כלב, אך לא יוכל להבדין בין הכלבים השונים, ואפילו המשטיכים ל-class ספציפי אך שייכים לאובייקטים נפרדים.

בפרק זה נדון במשימת **Object Detection** ובפרק הבא ב-**Segmentation**. בשבייל להגדיר את המשימה ולקבוע מה נחשבת הצלחה, ראשית יש להגדיר מה נחשב השטח בו נמצא האובייקט, ובהתאם מה אנו מזמנים מהמודל שיספק לנו כפלט. בתחום של **Object Detection**, המוסכמה היא לסמן אובייקט באמצעות מלבן, שנקרא **bounding box** (או בקיצור **bbox**), כאשר קצנות האובייקט משקפים **bbox**. נניח יש תמונה ובה בנאדם, אז ה-**bbox** יהיה מלבן המכיל את כל הגוף שלו, והמלבן ישיק לקצנות הגוף – החלק העליון של המלון ישיק לנקודה הגבוהה ביותר של הראש, החלק התיכון ישיק לכפות הרגליים, ובאופן דומה בשני הצדדים. אם יש כמה אובייקטים, אז לכל אובייקט ישיר **bbox** בנפרד, כפי שownbelow לראות באior הבא:

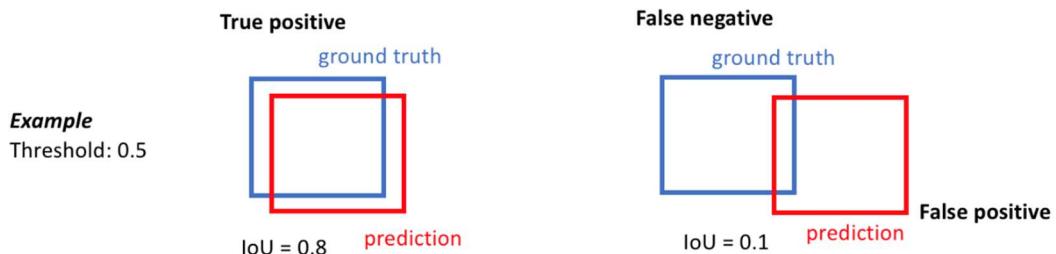


איור 9.1 תמונה ובה מספר אובייקטים, כאשר כל אובייקט מסומן באמצעות bbox – מלבן המקיף אותו ומשיק לקצנות שלו מכל כיוון.

לאחר שהגדכנו בצורה מדויקת מה נחשב השטח בו נמצא האובייקט – נוכל לנקחת תמונה ולסמן בה את האובייקטים בהתאם למוסכמה זו, כאשר נהוג לנחות כל אובייקט מתייג-ground truth. אם נאמן גלי (detector) לפי המוסכמה זו, אז נרצה שהפלט שלו יהיהiosoף של דטקטציות (detections) – מלבדים מסביב לאובייקטים ייחד עם סיווג תואם

לכל אחד מהאובייקטים. כאשר נרים את הגלאי המאוון על תמונה מסוימת, נוכל להשוות בין כל האובייקטים שתויגו (groundtruths), לבין הדלקציות של המודל. מתקבל לבצע את ההשוואה בעזרת מדד שנקרא Intersection Over Union, או בקיצור IoU. מדד זה בוחן עבור כל אובייקט את היחס של גודל החפיפה בין groundtruth לבין הדלקציה התואמת עבורו. אם החפיפה גדולה מסויימת, אז נוכל לומר שהגלאי דיביה בהצלחה שישפה אובייקט. סוף זה נקרא IoU threshold, וערך טיפוסי עבורו הינו 50%, כלומר חפיפה של מעל 50% בין מיקומו האמיתי של האובייקט לבין הדלקציה של המודל נחשבת כמציאות נכונה של האובייקט. אם גם הסיווג של אותו אובייקט על פי המודל זהה ל-class האמתי (בהתאם למה שתויג מראש), אז יש פה התאמה מלאה ואוטומטית אובייקט הוא בעצם true positive – דוגמה חיובית שזוהתה באופן נכון.

אם יש אובייקט שאין אף דלקציה שחופפת לאזור בו הוא נמצא, אז אותו אובייקט הינו false negative (false negative). אובייקט יהיה false negative גם במקרה בהם יש חפיפה אך היא אינה עובייה את הסף שנקבע, או לחילופין יש חפיפה מספיקה אך הסיווג שגוי. במקרים אלו מעריכים את המקרים האחרונים יש גם false positive – הדלקציה אינה מתאימה לאף דלקציה groundtruth, ולכן היא למעשה דיביה אובייקט שבאמת לא קיים. הגדרות אלו מאפשרות לשימושם במשימות Object Detection במטריקות מקובלות כמו precision ו-recall באופן דומה לשימוש הקלאסי classification, כפי שיאסביר בהמשך בהמשך.



איור 9.2 (IoU) – מדד חפיפה בין groundtruth לבין דלקציה. עבור threshold=0.5, דלקציה false negative נינה עובייה את סף החפיפה הינה ובונוסף הינו groundtruth- false negative.

לאחר שהגדכנו בצורה מדויקת ומפורטת מהי משימת דיביה אובייקטים נסbir בקצרה על הגישות השונות בתחום והפתחותן לאורך השנים. אמנים בשנים האחרונות כל המודלים הינם מבוססי געון Deep Learning, אך כموון שימוש דיביה אובייקטים היזהה קיימת עוד לפניה התפתחות הענף והו אלגוריתמים קלאסיים לצורך כך, כמו למשל אלגוריתם Scale-Invariant Feature Transform (SIFT) – Localization and Classification – פופולרי מאוד בעשור הראשון של המאה. לרוב, גם האלגוריתמים הקלאסיים וגם אלו מבוססי רשות נוירונים פועלם על פי אותו רעיון, שהוינו פשוט ואינטואיטיבי. כיוון המשימות דיביה אובייקטים כוללה משני חלקים – Classification – נרצה שהמודול שלנו גם הוא יפעל בשני חלקים – בשלב הרាឌון נמצא אזור עניין בתמונה, כלומר אזורים החשודים ככליה שיש בהם אובייקטים ROI – Region Of Interest), ולאחר מכן נבצע שני דברים במקביל – ננסה לחזות את ה-area המדויק של אובייקט הנמצא באותו אזור, ובונוסף ננסה לסwoג אותו לפחות מסויים.

למרות שגישה זו פשוטה יחסית, יש בה שתי בעיות מובנות. ראשית כל – יתרונו אזורים שהמודול יחשוד שיש בהם אובייקט אך בפועל באמת אין בהם אובייקט אלא הם מכילים רק רקע. שנית – הרבה אלגוריתמים מתוכננים באופן כזה שהקלט שלהם הוא בגודל קבוע וידוע מראש, והם גם מוצאים פלט בגודל קבוע. כיוון שמספר האובייקטים הוא משתנה, פלט בגודל קבוע יכול להיות בעייתי. למעשה יש פתרון פשוט לשתי הבעיה – נדרש מהמודול לצרף לכל אחת מהדלקציות מספר הסתברותי בין 0-1 שמשמעותו היא "עד כמה אני בטוח שבאמת יש פה אובייקט". במקביל לכך, על המשתמש להחליט על סף מסויים (confidence threshold), כך שרק אובייקטים בעלי הסתברות גבוהה יוצאים זה יחשבו כדיבויים של המודול, ומכל יתר הדיבויים פשוט נטעלם. באופן זה נוכל לקבוע מראש מהמודול יוציא מספר קבוע של דיבויים, למשל 100, אך נתעלם מכל אלה שלא עברו את הסף. מודל טוב ייתן הסתברות גבוהה רק לאזורים בהם באמת יש אובייקטים, ואילו כל יתר האזורים החשודים יהיו בעלי הסתברות נמוכה. רוקן דע להתעלם מאזור עניין שנמצא בחשודים שלו ולתת להם הסתברות נמוכה. (בהמשך יסביר על אלגוריתם NMS שגם מתקשר לעניין של מספר הדלקציות).

כאמור לעיל, בשנים האחרונות כל הגישות המובילות בתחום זה הן מבוססות Deep Learning, כאשר הן פועלות על פי אותו עיקנון – חילוץ ROI בשלב הרាឌון, ולאחר מכן ביצוע גרסיה ל-bbox של האובייקט יחד עם סיווג שלו לכלי מסוים. באופן כללי ניתן לחלק את הגלאים לשולש קבוצות: class specific. בפועל כל אחד יתבצע על-

גלאי דו-שלבי (Two-stage detector):

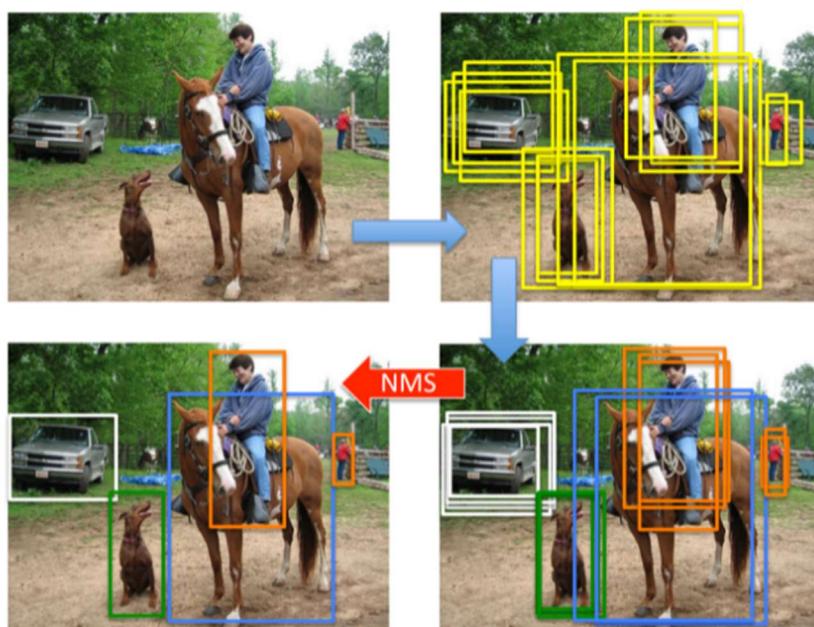
גישה אחת, הנקראת two-stage detection, akan פועלת לפי שני השלבים שתיארנו קודם: בהתחלה מופעל אלגוריתם ליז'וי ROI, כאשר אלגוריתם זה הוא רשות נוירונים שמחלצת פיצרים בתמונה נתונה ומצהה בה אזור עניין, ולאחר מכן אזורים אלו נכנסים לרשות המבצעת רגסיה ל-*bbox* וסיווג ל-class. המוצא של הרשות השנייה בניי בצורה של head detection two, כאשר לרוב ראש הרגסיה מנשה למודד בתהליך האימון (ולוחות ב-inference) ארבעה פרמטרים – את מרכז האובייקט (x, y) והאורך והרוחב שלו (w, h).

גישה זו היא בעצם השיטה הci אינטואטיבית, ומשפחת הגאלים הראשונה ואלו הci מפורסמת הפעלת לפי עיקנון זה של two-stage היא two-stage Family R-CNN. משפחה זו הינה סדרה של גאלים שפותחו בשנים 2014-2017, כאשר כל גלאי מתבסס על קודמו ומשפר אותו, וכך גם ייחד בניים בצורה של two-stage. עם הזמן התפתחו עוד גאלים הפעילים באופן דומה, כמו למשל SSP ו-FPN שנדרן בהם בהרחבה בהמשך הפסק.

גלאי חד-שלבי (One-stage detector):

גישה נוספת, דומה בקונספט אך שונה בדרך הביצוע, נקראת one-stage detection. גם בגישה זו הרעיון הוא למצאו אזור עניין ועבורם לבצע רגסיה של *bbox* וסיווג ל-class, אך הפעם אזור העניין אינם נקבעים באמצעות אלגוריתם כלשהו, אלא הם קובעים עצמם מראש. אנו למשה אמורים מראש למודל איפה בדיקן לחפש אובייקטים, כאשר יש anchors אמורים מסומנים מראש נקראים anchors. בדרך כלל קובעים מראש הרבה anchors, כאשר יש את אותם אזורים מסומנים מראש נקראים anchors. הרעיון הוא שנקבע מראש anchors ובצורות שונות, על מנת שנוכל לזהות אובייקטים במגוון גדלים וצורות. למשל – נחלק את התמונה לגדלים בגודלים שונים וכל מלבן יהיה anchor, אך נרצה להוסיף גם anchors מבנים רוחביים בצדדים של זהות מכניות, וכן נרצה anchors מבנים אופקיים בצדדים של זהות אנשים עומדים, וכך הלאה. העיקנון הוא שנקבע מראש הרובה anchors כך שכל אזור בתמונה יcosa במספר anchors בגדלים שונים, כדי לאפשר זיהוי של אובייקטים בעלי אופי שונה. גלאי מפורסם שעבד לפי העיקנון של one-stage הוא You Only Look Once – YOLO, והוא – הוא עבר על התמונה רק פעם אחת (ולא כמו ב-two-stage), שם יש צורך לעבור פעמיים על התמונה – פעם אחת לחילוץ ROI ופעם שנייה לרגסיה וסיווג). גאלים נוספים הפעילים לפי גישה זו הינם SSD ו-RetinaNet, שגם בהם נדרן בהרחבה בהמשך.

בשונה מגלאי דו-שלבי המחלץ בעצמו את אזור העניין, גלאי המשמש ב-one-stage קובע מראש את מספר אזורים העניין. עובדה זאת יכולה לגרום לכך שההיא כמה אזור עניין שונים המכילים את אותו אובייקט. אם למשל חילקנו את התמונה לגדיד של 10×10 ויש אובייקט שתופס רבעת התמונה, אז יהיו כ-25 anchors שיכילו בתוכם את האובייקט. כדי לאחד את כל היזוריהם של anchors הahaha מקובל להשתמש באlgorigthム K-Loss שנקרא Non Maximum Suppression, או בקיצור NMS. הרעיון של האלגוריתם הוא לעבור על כל הדטכציות, ואם יש דטכציות השויות אותה קלאס ויש בינהן חפיפה מסוימת, אז הן יתאחדו לכדי דטכציה אחת. השימוש באlgorigthムים זה נפוץ גם בغالים דו-שלבי – אמנם הגאלים מחלצים בלבד את אזור העניין, אך ישם גאלים בהם מספר אזור העניין קבוע מראש, ולכן יתרוכנו הרבה דטכציות של אותו אובייקט, ויש צורך לאחד אותן.



איור 9.3 Non-Maximum-Suppression (NMS) – איחוד דטקטיות השויות לאוות אובייקט. בשלב הראשון מוחפשים אובייקטי עניין, לאחר מכן מסוגים כל אחד מהם, ולבסוף מאחדים זיהויים השווים לאוות class ויש בהם חפיפה גבוהה.

גליי מבוסס טרנספורמר (Transformer-based detector)

כניסתם של הטרנספורמרים לעולם ה-vision לא דילגה על DETR. Object Detection (DEtection Transformers) שהראה כיצד ניתן להשתמש במנגנון של self-attention בשביל למצוא אובייקטים בתמונה ולאחר מכן להשתמש במנגנון של attention בשבייל לסוגו אוטם, כפי שיפורט בהרבה בהמשך הפרק. מאז ייצאו הרבה מאמרם המבוססים על הרעיון זהה בו-DET, כאשר מאמרם אלו מנסים לשפר את התוצאות תוך תוקן התיחסות למספר בעיות מסוימות הקשורות בשאלות כאשר משתמשים בטרנספורמרים במשימות של ראייה ממוחשבת. בפרט, יש מספר אתגרים בסיסיים: א. הסיבוכיות של טרנספורמר היא ריבועית בגודל הקלט הוא משפט, או אפילו פסקה, זה לא מהוות בעיה, אך כאשר מדובר בתמונות ברזולוציה גבוהה (למשל 4K), אז שימוש בטרנספורמרים נהיה לא עיל, ועבור ממשימות זמן אמת אפילו בלתי אפשרי. ב. טרנספורמרים תוכנוו במקור לקלט חד ממד', כמו למשל משפט שהוא סדרה של מילים. בשבייל להשתמש בהם עבור תמונות, יש להפוך את התמונות מייצוג דו-ממדי לייצוג חד ממד'. מעבר זה יכול לאיבוד קשרים מרחביים הקיימים בתמונה. ג. מנגנון ה-attention יכול לזהות תבניות מרחביות, אך הוא לא מתייחס לממד הצלבים, שכמוכן יש לו שימושות בעיבוד התמונה.

כל אחת מהגישות יש יתרונות וחסרונות וכמוכן שיש כל מיני עבודות שמנסחות לשבל בין היתרונות של הגישות השונות. באופן פשוט ניתן לומר כייחס לשתי הגישות הראשונות, שהן מקיימות trade-off בין דיוק לבין מהירות – גליי חד-שלבי הוא כמוכן יותר מהיר, אך עם זאת הוא פחות מדויק. עבור מכשורי קיצה שרים בזמן ואין להם כוח חישוב רב, כמו למשל פלאפונים, הרכאים העיקריים של המשתמש הם מהירות חישוב וחיסכון בסוללה, וכן נראה שתהיה עדיפות לגליי חד-שלבי. אם לעומת זאת המשתמש צריך דיוק גבוה ויש לו משאבים חישוב חזקים, כמו למשל רופא שמתעסך בעונח בדיקות רפואיות, אז תהיה לו עדיפות לגליי דו-שלבי או לגליי מבוסס טרנספורמר, המספקים תוצאות מעט יותר טובות.

כאמור לעיל, כל הгалאים בשנים האחרונות הם מבוססי רשותות עבר זיהוי IsItOR, רשותות רגסיה ויזואג, רשותות של טרנספורמרים ועוד. בשבייל לאמן את הרשותות האלה יש צורך בדעתה, וכמוכן שצורך שהוא היה מתויג. בשונה מתיווג תמונות למשימות סיווג, שם כל שנדרש זה רק לתת label לכל תמונה, תיוג של תמונות עבור משימות Object Detection היא משימה מורכבת פי כמה – גם יש כמה אובייקטים בתמונה, וגם צריך לסמן לכל אחד מהם *bbox* בזורה מדויקת. עקב המורכבות של תיוג זהה, התפתחו אלגוריתמים של semi-supervised, המשלבים למידה מבוססת דedataה מתיויג יחד עם למידה שנייה דורשת תיוג.

נזכיר בקצרה רעיון אחד של semi-supervised learning עבור משימת זיהוי אובייקטים, המאפשר לאמן מודל בלי הרבה>Dataה מתיויג. הרעיון הוא לאמן את המודל בשני שלבים – בשלב הראשון משתמשים בדעתה המתויג ומאמנים מודל באחת הדריכים שראינו עד כה. לאחר מכן מושרים את הлокלייזציה באופן הבא – מרכיבים אלגוריתם *clustering* שמודיע לזהות אובייקטים (הוא בעצם עושה סוג של *label* למלט של המודול המקורי שאיemann). כיוון שהאלגוריתם השני מאומן אובייקטים), ומושווים בין הפלט של המודול המקורי שאיemann. רק לבצע לוקלייזציה והוא אינו נדרש לבחין בין אובייקטים שונים, ניתן להשתמש עבורי בהרבה>Dataה, ולכן הנחה סבירה היא שהוא מזהה אובייקטים בזורה טוביה. אם כן, ניתן לבצע אימון שנייה לעשוות *tune* fine לлокלייזציה של המודול הראשון כך שה-*bbox* שהוא מספק יהיה דומה עד כמה שניתן לפולט של האלגוריתם השני.

בדומה לכך, ישנו אלגוריתמים של self-supervised שאינם מצריכים בכלל>Dataה מתיויג. אלגוריתמים אלה לומדים לזהות אובייקטים ולהבחין בין אובייקטים שונים, אך הם אינם יכולים לדעת מה ה-label של כל קבוצה, כיוון שאין להם שום>Dataה מתיויג. זה יכול להיות שימושי למשל עבור רופאים המסתמכים על תוצאות של צילומים רפואיים – האלגוריתם יזהה את האובייקטים ויסוג כל אובייקט לקבוצה מסוימת, וזה הרופא יסתכל על התוצאות וידע שתת את ה-label המתאים לכל אחת מהקבוצות.

9.1.3 You Only Look Once (YOLO)

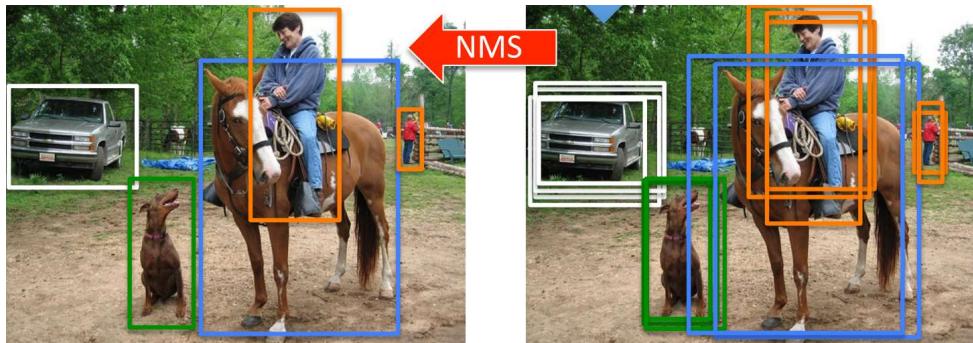
עד שנת 2016 כל הгалאים (detectors) המבוססים על למידה عمוקה (כמו למשל R-CNN family) היו גליים דו-שלביים – בשלב הראשון יוצר אלפי הצעות (proposals) למסגרות מלכניות החשודות כמכילות אובייקטים, ואלו הוזנו בזה אחר זה לשבל השני אשר דיק את המסגרות וביצע סיווג לאובייקטים המוכלים בהן.

ארQUITktורת YOLO, הנגדית מראשי התיבות של ONLY LOOK ONCE, הוצאה על ידי ג'וזף רדמן ב-2016, והיתה הגלאי הראשון שモרכיב משלב יחיד, ובו הרשת מנבאת את המסגרות וגם מסוגת את האובייקטים שבתוכן בביטחון. בנוסף, ארQUITktורת YOLO מתאפיינת במיועט פרמטרים וכמות פעולות אРИתמטיות. היא אמנם משלמת

על המבנה הרזה והאלגנטית שלה בדיק נמוך יותר, אך המהירות הגבוהה (שנובעת בעיקר מהיעדר האילוץ לעבד מסגרת יחידה בשלב השני בכל מעבר של הלולאה) הפכה את גישת השלב היחיד לאטרקטיבית מאוד, במילוי אחד למעבדים קטנים כדוגמת מכשירי mobile. בעקבות עבודה זו פותחו גלים רבים על בסיס שלב יחיד, כולל גרסאות מתקדמות יותר של YOLO (הगרסה המתקדמת ביותר כיום היא 5).

NMS (Non-Maximum Suppression)

כמעט כל אלגוריתם של זיהוי אובייקטים מייצר מספר רב של מסגרות חדשות, כאשר רובן מיותרות ויש צורך לדלול את מספן. הסיבה ליצירת מספר גדול של מסגרות נובע מ敞开 פועלות הgalios. בזכות התכונות הלוקאליות של פועלות הקונבולוציה, מפת הפיצרים במקומות הgalios ניתנת לתיאור כמטריצת משbezנות אשר כל משbezצת שකולה לריבוע של הרבה פיקסלים בתמונה המקורית. רוב הgalios פעילים בשיטת עוגנים (anchors), כאשר כל משbezצת במקומות הgalios מנבאת מספר קבוע של מסגרות שעשוות להכיל אובייקט (למשל, ב-YOLOv2 המספר הוא 5, ובגרסאות המתקדמות יותר המספר הוא 3). השיטה זו יוצרת לפחות מוגנות שרק מעתות מהן הן משמעותיות. בנוסף – יש ריבוי של מסגרות דומות בסביבת כל אובייקט (למשל – YOLOv3 – מנבאת יותר מ-7000 מסגרות לכל תמונה). אחת הדריכים הפופולריות לסנן את אלף המסגרות ולהשאר רק את המשמעותיות נקראת NMS. בשיטה זו מתבצעת השוואה בין זוגות של קופסאות מואתיה המחלקה (למשל – חתול), ובמידה שיש ביןיהם חפיפה גבוהה – מוחקים את המסגרת בעלי התוודאות הנמוכה הייתר ונשארים רק עם המסגרת בעלי רמת התוודאות הגבוהה. שיטה זו בזבנית בחישוב (סיבוכיות פרופורציונלית לריבוע מספר המסגרות) ואני חלק מהמודול המתאים, אך עם זאת הינה אינטואיטיבית יחסית לימוש, ומשום כך נמצאת בשימוש נפוץ בגלאים, כולל ב-YOLO.

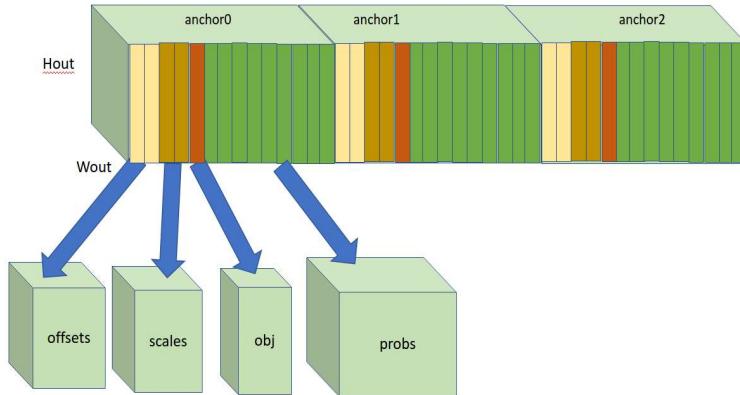


איור 9.1 אלגוריתם NMS.

YOLO Head

כמו רוב הгалאים, YOLO הינו מבוסס-עוגנים (anchor-based), שהין תיבות מלכניות קבועות ושונות זו מזו בצורתן. לכל עוגן מוקצה מקטע של פיצרים במתפת המוצא של הרשת וכל הניבויים במקטע זהה מקודדים כסתויות (offsets) ביחס לממד העוגן. כפי שניתן לראות באירור 9.2, הפיצרים של כל תא מרחבי במתפת המוצא מחולקים למקטעים על פי העוגנים (שלושה עוגנים במרקחה זהה).

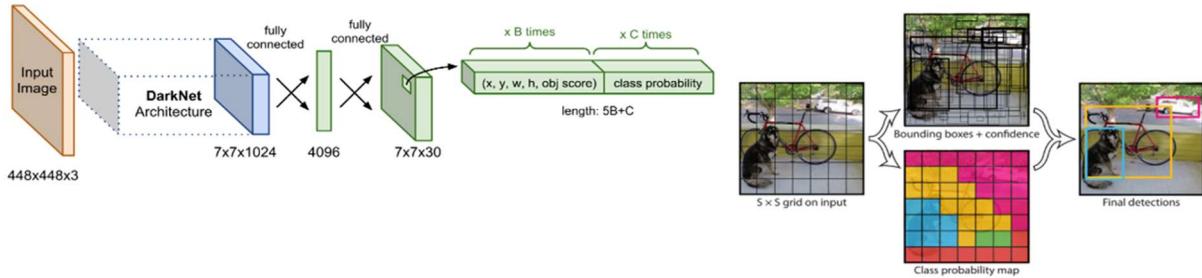
ניבו הוא מסגרת שמרכזתה נמצא בנקודה כלשהי בשטח התא (השקלול לריבוע של מספר פיקסלים בתמונה המקורית). ההסתה המדוייקת של מרכז המסגרת ביחס לתא ניתנת על ידי שני הפיצרים הראשונים ברכף.elog הממדים של הקופסה (ביחס לממד העוגן) ניתן באמצעות שני הפיצרים הבאים ברכף. הפיצר החמישי לומד את מידת ה-objectness, כפי שהסבירה לעיל. שאר הפיצרים ברכף של העוגן הנ"ל הם הסתברויות המותnenות לכל מחלקה (אם אוסף הנתונים מכל 80 מחלקות, יהיו 80 פיצרים כאלה). על מנת לקבל רמת ודאות סופית, יש להכפיל את מדד ה-objectness במדד הסתברויות המותנה לכל מחלקה.



איור 9.2 ראש YOLO.

YOLOv1

מודלי YOLO מבוססים על גרסאות Backbone הנקראות Darknet ומשמשות לעיבוד פיצ'רים מתוך התמונה, ומרחשת detection המקבל את הפיצ'רים האלה ומתאים ליצור מהם ניבויים למסגרות סביב אובייקטים. המודל מחלק את התמונה לרשת בעלת S משבצות, אשר כל משבצת מנابت N מסגרות של אובייקטים בשיטת העוגנים, כאמור לעיל. כל יוביי כולל את מספר ערכיהם, הסטת הקואורדינטות u, v , של מרכז המסגרת ביחס למשבצת, הגובה והרוחב של המסגרת, ורמת ה-objectness, כפי שהווסבירה לעיל. בנוסף, כל מסגרת מבצעת גם סיווג, כולם מנबאת את רמת הוודאות של השתייכות האובייקט לכל אחת מהמחלקות האפשריות. החידוש באlgorigthms נועד בעובדה שחייבי המסגרות וסיווג לאובייקטים נעשו במקביל, ולא באופן דו-שלבי. הרעיון הוא להתייחס לסוג האובייקט בעוד פיצ'ר שהרשת מנסה לחזות בנוסף למקומם וגודלה של המסגרת.



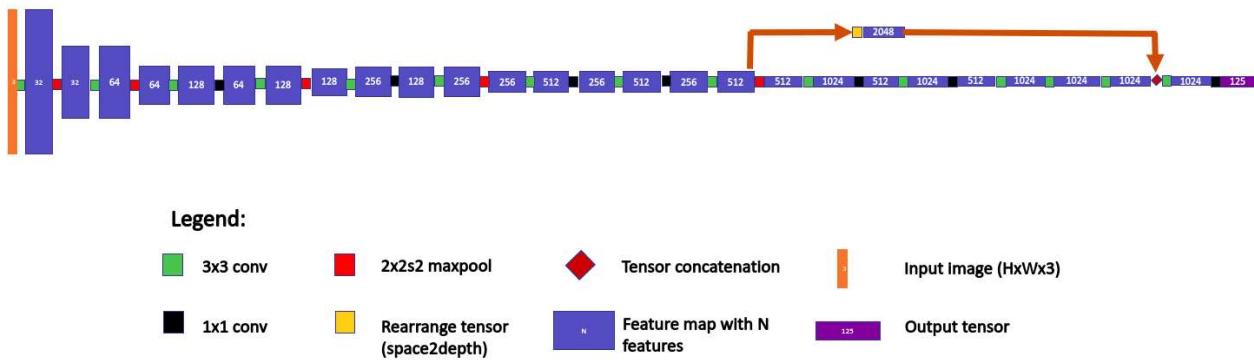
איור 9.3 ארכיטקטורת YOLO.

הגרסה הראשונה של ארכיטקטורת YOLO כללה שכבה fully connected שהוסרה בגרסאות הבאות. בנוסף, פונקציית LOSS וודר התכוונות של המסגרות בموقع הרשת השטני, אבל הרעיון נותר זהה.

YOLOv2

גרסה זו משתמשת ברשת Backbone הנקראת Darknet19 ובנה 19 קוןволוציות. המודל כולל 22 קוןולוציות (מלבד 5 שכבות MAXPOOL המקבילות על מנת למצוא את הפיצ'רים), ועוד מסלול עוקף בסמוך לסוף הרשת המחזק את יכולת העיבוד. הכותב של המאמר המקורי, רדמן, נהג לפתח גם גרסאות "Tiny" לכל מודל. הוויריאנט Tiny YOLOv2 מכיל רשת Backbone קצרה יותר ולא מסלול עוקף ומבנהו לינארי ופשטוט מואוד. ביצועיו אמורים נಮוכים יותר אך הוא מהיר מאוד (207 תמונות לשניה לעומת 67 של מודל2 YOLO, על מעבד TitanX).

מעניין לציין ש-2 YOLOv2 הוא המודל הראשון שעובד על תמונות בממדים משתנים, תהליך המשפר את דיק המודל.

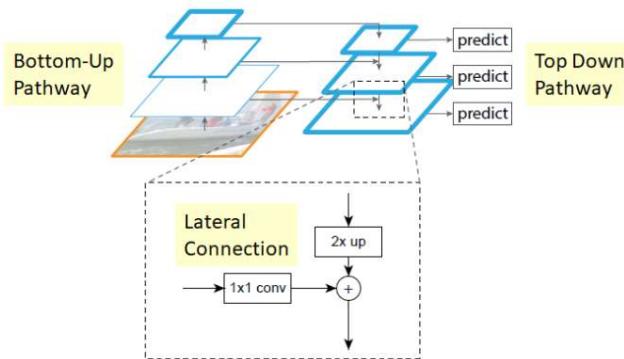


איור 9.4 ארכיטקטורת YOLOv2.

YOLOv3

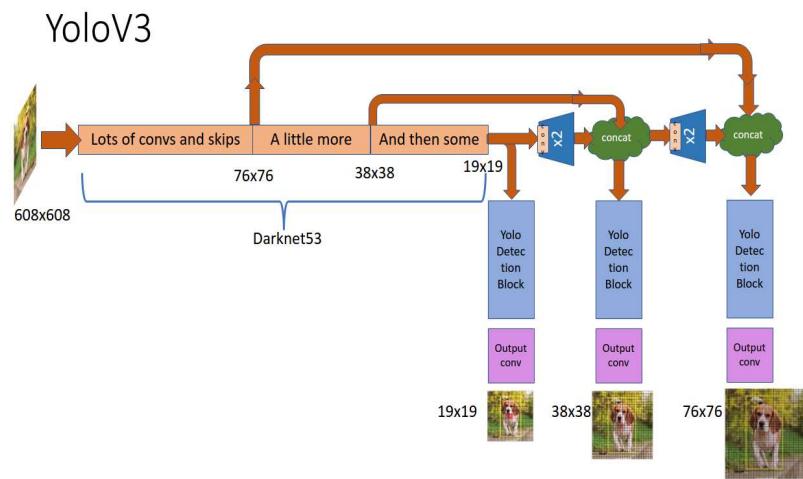
כל דור נוסף של YOLO הציג חידושים ארכיטקטוניים שהגדרו את מרכיבות החישוב וגודל המודול ושיפורו את ביצועיו. גרסה מס' 3 מבוססת על רשת Backbone גדולה בהרבה שנקראת Darknet53 ובה, כפי שעולה ממנה, 53 קונבולוציות. כמו כן הרשת מכילה צואර של ארכיטקטורת Feature Pyramid Network (FPN) בעלת שלושה ראשיים גילוי, כאשר כל אחד מהם הוא בעל רזולוציה שונה (38×19 , 38×19 , 76×76). ארכיטקטורת FPN היא תוספה בקצה Backbone, אשר מגדילה חזרה באופן הדרוגי את מפת הפיצרים. תוספה זו נועדה ליצור מסלול Top Down במקביל למסלול Feed Forward ברשת Backbone, שבינוי מעשה בקורסית Up, Bottom, בו ככל שמתקדמיים בשכבות Backbone-המצביע עיבוד יתבצע ברמה גבוהה יותר והרזולוציה המרחבית של מפת הפיצרים הולכת וקטנה. בעזרת השימוש ברשת FPN המודול לומד לנצל את המיטב משני עולמות: הוא משתמש במידע שטמונו במפת הפיצרים הגדולה, שהוא אמן פחות מעובדת אך בעלת פיצ'רים ברזולוציה מרחבית גבוהה, ובנוסף הוא מנצל גם את המידע ממפת הפיצרים הקטנה, שהוא אמן בעלת פיצ'רים ברזולוציה מרחבית נמוכה, אך עם זאת הוא מעובדת יותר.

לאחר כל הגדלה של מפת הפיצרים מתבצע חיבור בין התוצאה בין מפת פיצ'רים קדומה יותר במדדים זהים (מתוך Backbone). זאת בדומה לחיבורים העקיפים ברשת ResNet המשמשים להתקנסות האימון. השכבות השונות שלרשת FPN מאפשרות לגלאי למצוא מיקום מדויק יותר של האובייקט ברזולוציות השונות, מה שמעניק לרשת זו את היכולת להבחן באובייקטים קטנים בתמונה גדולה.



איור 9.5 ארכיטקטורת FPN (FPN), המשלבת מסלול top down לאחר ה-*up* bottom.

לרשות המודול של YOLOv3 יש מספר עופי detection, אשר כל אחד מהם פועל על מפת פיצ'רים ברזולוציה שונה ובאופן טבעי מתמחה בגילוי אובייקטים בגודל שונה (הענף בעל הרזולוציה הנמוכה מתמחה בגילוי אובייקטים גדולים, והענף בעל הרזולוציה הגבוהה מתמחה בגילוי אובייקטים קטנים).



איור 9.6 ארכיטקטורת 3vYOLO.

YOLOv4

רשת YOLOv4 היא בעלת ראש זהה לזו של שתי הגרסאות הקודמות, אך ה-*Backbone* שונה. הוא נקרא CSPDarknet53 כאשר הוא קיצ'ר של CSPNet. רשת זו מפצלת מפות פיזרים לטובות קונבולוציה בחלקים ואיחודה מחדש. פיזול זה אפשרי, כמפורט במאמר המקורי, חלהlus טוב יותר של Leaky ReLU (ולא Mish) בחלק האימון. בנוסף, נעשה ברשת זו שימוש בפונקציית אקטיבציה הנקראת Mish (ולא Leaky ReLU) כמו בגרסאות הקודמות).

אנדרטת מענינית – החל מגרסה זו התוצאות אינן של היוצר המקורי ג'וזף דמדון, שהחליט לפרוש מחקר ראייה מהחשכתם בגלל שיקולים אטיים של שימושים צבאים או שימושים הפוגעים בפרטיות.

YOLOv5

רשת YOLOv5 מוסיפה עוד שכליים על רשת Backbone-*53*, על פניו זו של הדור הקודם, ומציגת אופרטור חדש המארגן פיקסלים סמוכים בתמונה בממד הפייצרים. אופרטור זה דואג לכך שהכינסה לרשת היא לא עמוק (3 פיצרים מקובל (RGB), אלא 12 פיצרים, תוך הקטנת הממד המקורי). באופן זה הרשת מתאמת לעיבוד תמונות ברחולוציה גבוהה, אף לזראות אובייקטים גדולים בקהלות רבה יותר, שכן שדה התמך (receptive field) של הקונבולוציות מכל מידע משתח תמונה גדול יותר.

9.1.5 Spatial Pyramid Pooling (SPP-net)

Spatial Pyramid Pooling (SPP) הינה שכבת pooling ברשת נירונים, שמטירה להסיר את האילוץ של רשתות קונבולוציה, הדורש שכבת הכניסה לרשת תכיל תמונה בגודל קבוע (כמו למשל רשת VGG, המתקבל רק תמונות בגודל 224×224).

כיום קיימים מכשירי צילום רבים – החל מצלמות ניידות, מקצועיות, מצלמות אבטחה ואף מצלמות בטלפונים סלולריים ורוחניים. מצלמות שונות עשויות להוציא צפלי תמונות בגודלים שונים, מגוון סיבות (למשל איות התמונה או מטרת המכשיר). אם נרצה לבצע סיווג באותה רשת נירונים 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). פעולות אלה עלולות לפגוע בזיהוי עקב שניי היחס בתמונה (מתיחה/כיווץ), החסירה של פרטים או שילוב של השניים. מוטיבציה זו היא שהובילה לשימוש בטכניקת SPP.

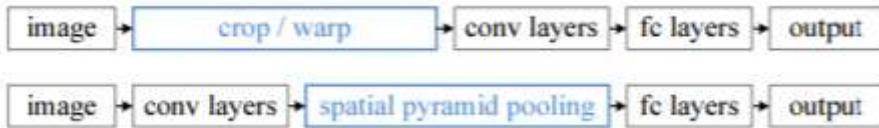
ניתן לראות דוגמא לשינוי גודל באמצעות מתיחה ולחיתוך באירור הבא:



איור 9.7: מימין - שינוי פרופורציה. משמאלי – חיתוך.

למעשה, הדרישה לתמונה קולט בגודל קבוע בכניסה לרשותת אלה אינה הכרחית, שכן שכבות הקונבולוציה יכולות לחוץ מאפייני קולט (feature maps) בכל גודל. לעומת זאת, שכבות ה-FC בעומק הרשות הן השכבות שדורשות בכניסה להן קולט בגודל קבוע.

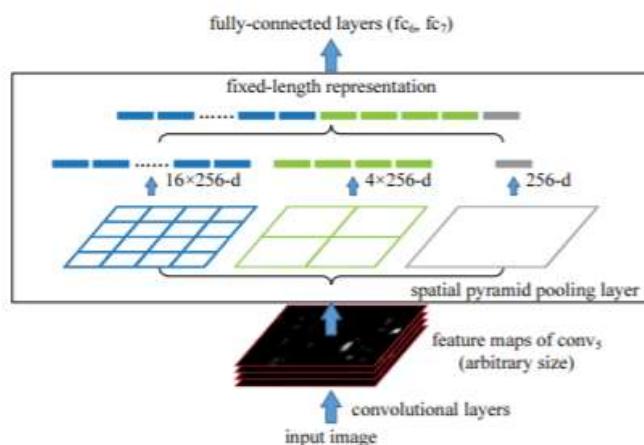
כעת, לאחר שהובירה המוטיבציה לייצור רשת זו, ניתן להבין את אופן פועלתה. על בסיס שכבת ה-SPP מתבססת רשת SPP-Net. החידוש במבנה הרשת, הוא שבמקום שכבה לרשota (לפניהם שכבות הקונבולוציה) תבוצע התאמת תמונה הקולט לגודל הנדרש ברשות. ההתקאה המבוצעה בשכבת SPP תבוצע לאחר מציאת המאפיינים בשכבות הקונבולוציה, אך לפני שכבת ה-FC, כמוואר באיר הבא:



איור 9.8 – מבנה של רשת CNN קלאסית (למעלה) לעומת מבנה של רשת SPP-Net (למטה).

הreasון מאחורי שכבת SPP הוא חלוקה של הפלט של שכבות הקונבולוציה, ביצוע max-pooling בכל חלק ורשור של התוצאות לווקטור שגודלו אחד. בambilם אחרים, עבור כל ה-*maps* המתקבלים לאחר שכבות הקונבולוציה, מיצרים שלושה וקטורים לכל :

- וקטור בגודל 1 המתקיים 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. מתקיים וקטור בגודל 4.
 - חלוקה של ה-*maps* ל-16 תת-חלקים (4×4) וביצוע max-pooling בכל חלק מתוכם. מתקיים וקטור בגודל 16.
 - שרשור כל הווקטוריים יחד לווקטור בעל 21 ערכיהם.
 - בסיום התהילה, תתקבל שכבה זו "מחליפה" את שכבת ה-*pooling* הממוקמת לאחר שכבת הקונבולוציה המשכבות הקונבולוציה. שכבה זו מחליפה את שכבת ה-*FC* האחורונה ותוצרה הוא הקולט לשכבת ה-*FC*.
- ניתן לראות את מבנה הרשת באיר הבא, בו התקבלו *features* 256:



איור 9.9: ארכיטקטורת SPP-Net.

9.2 Segmentation

אחד האתגרים הכי משמעותיים בעולם הראייה הממוחשבת הוא זהה אובייקטים בתמונה והבנת המתרחש בה. אחת הטכניקות הקלasicיות להתמודדות עם משימה זו הינה ביצוע סגמנטציה, כלומר, התאמת *label* לכל פיקסל בתמונה. בתחילת הסגמנטציה מבצעים חלוקה/בידול בין עצמים שונים בתמונה המצוולמת באמצעות סיווג ברמת הפיקסל, כלומר כל פיקסל בתמונה יסוויג וישיך למחלקה מסוימת.

ישנם שימושים מגוונים באלגוריתמים של סגמנטציה – הפרדה של עצמים מסוימים מהרקע שמאחוריהם, מציאת קשרים בין עצמים ועוד. לדוגמה, תוכנות של שיחות וUIDה, כמו zoom, skype, teams, ועוד, מאפשרות בחירת רקע אחדים שונים עבור המשתמש, כאשר מלבד הרקע הנבחר רק הגוף של המשתמש מוצג בוידאו. הפרדה גוף האדם מהרקע והטמעת רקע אחר מתבצעת באמצעות אלגוריתמים של סגמנטציה. דוגמא נוספת – ניתן להזמין לתמונה אדם, כלב, וביניהם רצעה, ומכך ניתן להסיק שתוכן התמונה הוא אדם מחזיק כלב בעזרת רצעה. במקרה זה, הסגמנטציה נועדה למצוא קשר בין עצמים ולהבין את המתרחש.

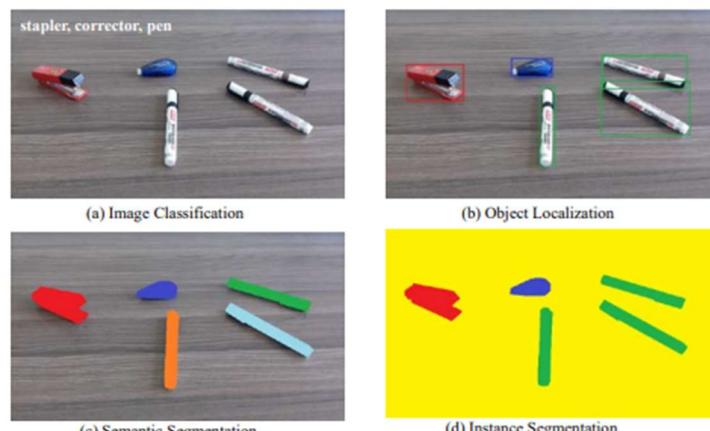
9.2.1 Semantic Segmentation vs. Instance Segmentation

קיימים שני סוגי עיקריים של סגמנטציה:

Semantic segmentation (חלוקת סמנטית) – חלוקה של כל פיקסל בתמונה למחלקה אליה העצם אותו הוא מייצג שייר. למשל, פיקסל יכול להיות משיר לכלי רכב, בן אדם, מבנה וכו'.

Instance segmentation (חלוקת מופע) – חלוקה של פיקסל בתמונה למופע של אותה מחלקה אליה העצם אותו הוא מייצג שייר. במקרה זה, בתמונה בה מופיעים מספר כלי רכב, תבוצע חלוקה של כל פיקסל לאיזה כלי רכב אותו פיקסל מייצג – מכונית 1, מכונית 2, אופנוע 1, משאית 1 וכו'.

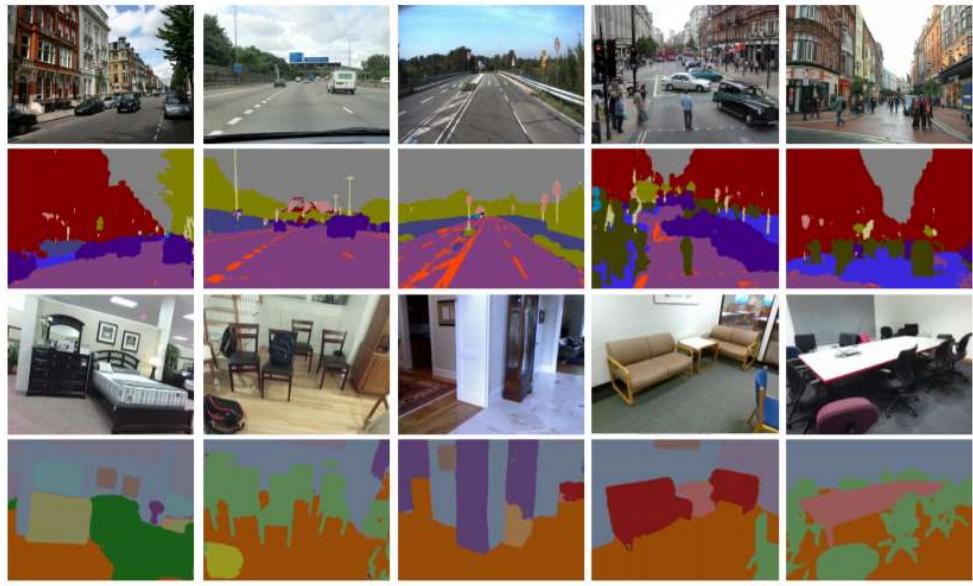
ההבדל העיקרי בין שני סוגי האו ברמת עומק המידע של פיקסל – המידע עשוי לסוג את הפיקסל למחלקה כלשהי, או לעצם ספציפי בתמונה. עומק המידע משליך גם על עלות המידע.חלוקת הסמנטית מבוצעת ישירות, בעוד שהחלוקת המופע דרושת בנווסף ביצוע של זיהוי אובייקטים כדי לסוג מופעים שונים של המחלקות.



איור 9.7 שימושות שונות בתחום של Computer Vision. ניתן להבחין בהבדל שבין Semantic segmentation (ההתאמה כל פיקסל למחלקה מסוימת) לבין Object localization (התאמת כל פיקסל למופע של מחלקה מסוימת).

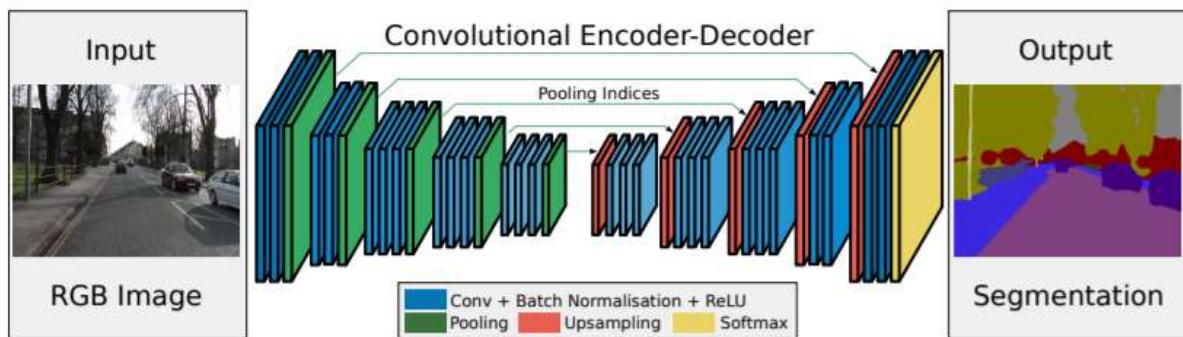
9.2.2 SegNet neural network

רשת SegNet הינה רשת קומבולוציה عمוקה, שמטרתה לבצע חלוקה סמנטית (Semantic segmentation) לתמונה הקלט. בתחילת הרשת פותחה להבנה של תכונות חז' (למשל כביש עם מכוניות ובצדדים בתים והולכי רגל) ותכונות פנים (למשל חדר עם מיטה וכייסאות). הרשת נבנתה מtower מטריה להיות ייעילה בהיבטי זיכרון וזמן חישוב, תוך שמירה על דיקט מעשי.



איור 9.7 סיווג סמנטי בעזרת רשת SegNet עבור תמונות פנים וחוץ.

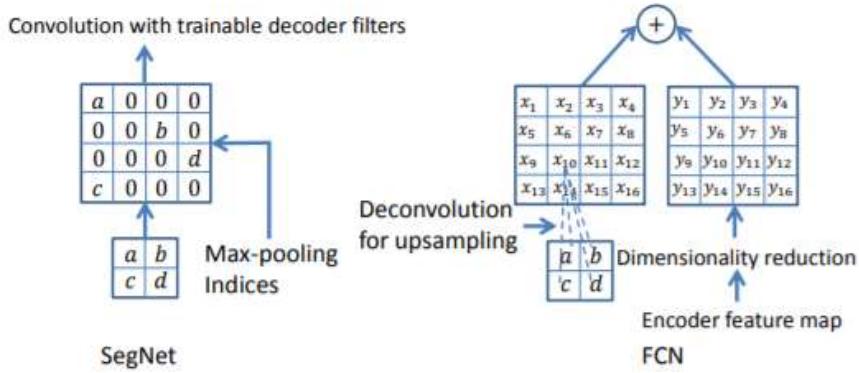
רשת בנויה כארQUITקטורת מקודד-מפענה (encoder-decoder). המקודד מורכב מ-13 השכבות הראשונות של רשת VGG16, ומטרתו לחלץ את מפת המאפיינים (feature maps). לכל שכבה קידוד יש שכבת פיענוח תואמת ומכאן שם רשת המפענה מכילה 13 שכבות. מטרת המפענה היא לבצע sampling-קָפָה וליצור תמונות מאפיינים בגודל המקורי. את מוצא המפענה מעבירים דרך מסווג SoftMax, המתאים את המחלקה בעלת ההסתברות הגבוהה ביותר לכל פיקסל בנפרד.



איור 9.8 ארכיטקטורת רשת SegNet

המקודד מורכב משכבות קוונטולוציה, 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.

החידוש ברשת זו הוא אופן הפעולה של המפענה. בשונה מרשתות אחרות, בהן תהליכי sampling-קָפָה גורר ביצוע חישובים עבור הפיענוח, כמוואר באIOR הבא, הרעיון ברשת זו הוא שמירת מיקומי ערכי המקסימום הנבחרים מכל רבייה. רק הערכים שנבחרו כמקסימום יושלמו בתהליך ושאר הערכים יתאפסו.



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. למחרת 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 שכבה SoftMax המבצעת סיווג ברמת הפיקסל. מוצא הרשת, שהוא גם מוצא שכבה SoftMax, הינו מטריצת הסתברויות, כאשר עבור כל פיקסל יש וקטור באורך K, כאשר K הוא מספר המחלקות לסיווג. כמובן שהסיווג מתבצע בהתאם לגובהה ביותר המתאימה לכל פיקסל.

אימון הרשת בוצע על בסיס מידע קטע ייחודי של 600 תמונות דרך צבעוניות בגודל 480×360 , שנלקחו מבסיס המידע CamVid road scene dataset. סט האימון הכיל 367 תמונות וסיט הבדיקה הכליל את 233 התמונות הנותרות. המטריה הייתה להזות בתמונות אלה 11 מחלקות (דרך, בניין, מכונית, הולכי רגל וכדומה). כל תמונה עברה נרמול מקומי לערך הניגודיות של תמונה הקלט לפני הכניסה לרשת. האימון בוצע בשיטת SGD, עם קצב למידה קבוע שערכו 0.1 ומומנטום שערכו 0.9. האימון נמשך עד שהגיית האימון התכנסה. לפני כל epoch סט האימון עורב וחולק לmini-batch של 12 תמונות. פונקציית המבחן הינה cross-entropy.

לעתים, נדרש לבצע איזון-מחלקות (class balancing). מונח זה מתייחס לכך שבו קיימים שונים גדולים בין כמות הפיקסלים המשווים לכל מחלקה, למשל כאשר קיימת הטיה מסוימת - סכינה שברובו מכילה בניינים / דרכים. במקרה זה, יבוצע משקל מוחודש לפונקציית השגיאה, באמצעות תהליך "איזון התדריות החזיניות" (median frequency balancing). התהליך מושך מחדש את המחלקות בפונקציית המבחן, באופן יחסית לחזין של תדריות הופעות המחלקות בכל סט האימון, תוך חלוקה בתדריות הופעת המחלקה:

$$\alpha_c = \frac{\text{median freq}}{\text{freq}(c)}$$

משקל זה משנה את היחסים בפונקציית המבחן כך שהתרומה של כל המחלקות לפונקציית המבחן תהיה שווה. לכן, הוא מעניק למחלקות הגדולות יותר משקל נמוך יותר ולמחלקות הקטנות משקל גבוה יותר.

9.2.3 Atrous Convolutions (Dilated Convolutions)

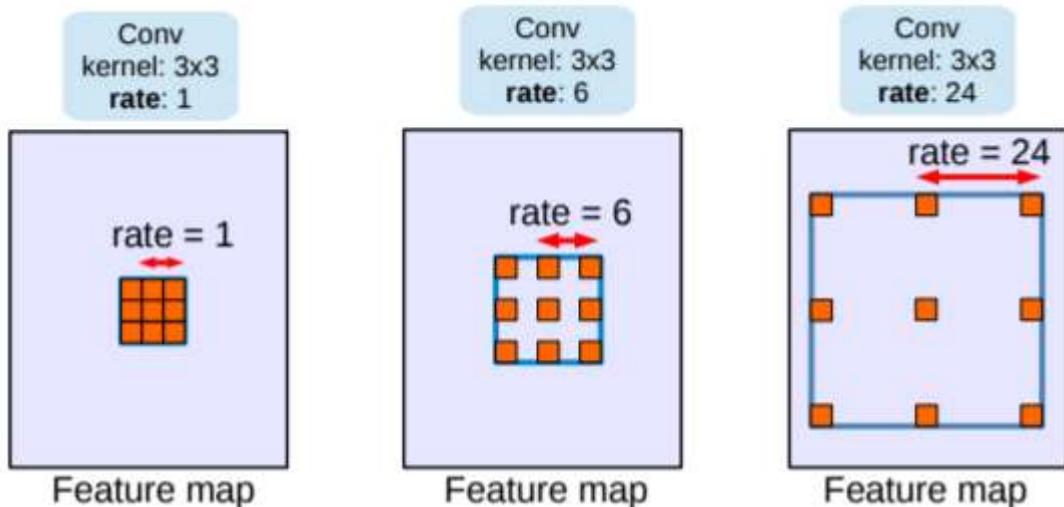
המונח Atrous מזכיר בשפה הצרפתית – "à trous", שמשמעותו "עם חורים". לכן, ניתן לתרגם את המונח Atrous convolution – כ-קונבולוציה מחוררת, ובמשמעות מעט יותר מתאימה – קונבולוציה מרוחקת (או בשם אחר – Dilated convolution – – קונבולוציה מרווחבת).

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, ואחר שנו ישר dilation של $1 > r$, מתקבלת קונבולוציה מרוחקת בפקטור r . יתרונה של קונבולוציה נועז בכך שהוא אותו קernel ועבור אותה כמה 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 קernel בגודל 3×3 ופרמטר התרחבות $1, 6, 24 = r$. בהתאם לכל פרמטר מתקבל field-of-view בגודל שונה – $3, 16, 49 \times 49 \times 49$ בהתאם.

9.2.4 Atrous Spatial Pyramidal Pooling

9.2.5 Deeplab V3+

נוקח רגע אחד אחרה בעט שנגדיր את שלוש הביעות המרכזיות של שימוש ברשתות קונבולוציה עמוקות (deep convolutional neural networks :Semantic segmentation networks

1. **מצומצום ברזולוציית הפיצרים** - קיימות שתי סיבות מרכזיות לתופעה זאת. הסיבה הראשונה היא שאימון של Semantic segmentation מורכב ויקר בהרבה מאשר קלסיפיקציה לתמונה. לכן, נועד לפתח המשמש ב-transfer learning מאשר ברשת שאומנה זה מכבר ובעל ידע קודם. יש לכך יתרונות וחסרונות, היתרון העיקרי הוא ההפחיתה המשמעותית בעליוויות. לעומת זאת, אחד החסרונות הוא שרוב הרשתות אומנו על dataset שמכיל תמונות אובייקטים רבים, אך התמונות בו בעלות רזולוציה נמוכה, מה שעלול להוביל לרזולוציית פיצרים נמוכה.

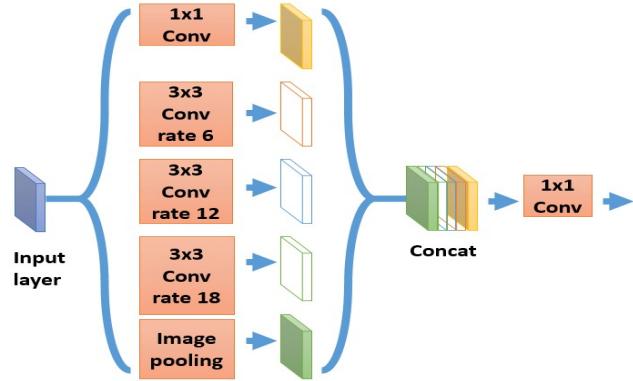
הסיבה השנייה היא שכאשר אנחנו משתמשים ברשתות קונבולוציה עמוקות אנחנו מקבלים לבנייה לאובייקט שמסתתר לנו בתמונה. על ידי זיהוק האובייקט מתוך התמונה הגדולה נוצרת באופן אינהרנטי רידעה ברזולוציית הפיצרים.

2. **הימצאות של אובייקטים בגודלים שונים** – נוקח לדוגמא Tamuna בה מופיעה מכונית גדולה מאוד ומוכנית קטנה. במקיריים מסווג זה, הרשת תנסה לשער את שני האובייקטים הללו לאותה קטגוריה.

3. **הפחיתה בדיקת המיקומי בಗל השונות של רשתות קונבולוציה עמוקות** - במקרה אחר, רשתות קונבולוציה עושות עבודה מעולמת בלזחות דפוסים, אך יש פה סוג של trade off - הרשת תזהה את הדפוס הכללי אבל "תשכח" איפה בדיקת מופיע האובייקט בתמונה.

כותבי המאמר של Deeplab V3+ תיארו שיטות להטמודדות עם הביעות שתיארנו. הם השתמשו בשתי ארכיטקטורות מרכזיות, חיבורו אותן ועם מספר שיפורים יצרו רשת חזקה עם תוצאות יפות על ה-datasets המרכזיים.

נדון כעת בשתי השיטות המרכזיות שהופיעו במאמר –



איור 1 – (ASPP) Atrous Spatial Pyramid Pooling -

השיטה הראשונה היא **קונволוציה מרוחקת** (Atrous convolution) כפי שנכתב לעיל, מדובר בכל מודול חזק. כל זה מאפשר לנו למכוד מידע קונטסטואלי עשיר ביותר ביותר ע"י איגוד של פיצ'רים בגודלים שונים. כאשר אנו עושים שימוש במסיבים אנו מקבלים ראייה מרחבית משתנה ולא קבועה, איגוד התמונות לתמונה אחת מאפשר לרשות כוח אדיר של הבניה. באמצעות זאת, הרשות מסוגלת ליצור קשרים בין אובייקטים שונים ואף בין אותם אובייקט שמופיע בגודלים שונים בתמונה.

האינטרואיציה שעומדת מאחורי ASPP היא שכאשר אנו עושים קונволוציה רגילה אנחנו בוחרים להתרცח בחלק מאוד ספציפי (בגודל קבוע) בתמונה. באופן זה נקבל את הפיצ'רים מאותו החלק אבל ההקשר הסביבתי יהיה מוגבל. זאת לעומת השימוש ב- ASPP אשר מקנה הקשר סביבתי שונה וגדל יותר (בגלל הפרמטר של המרווח המשותה). כמובן, כאשר מגדים את התמונות לתמונה אחת אנחנו מוחזקים את הבניה הגלובלית שיש לרשות על התמונה. זאת לאור העובדה שכבר לא מדובר בהסתכלות על תמונה אחת אלא על הרבה זוויות שונות שעוזרת לרשות להבין קשרים והקשרים בצורה יותר ברורה. אין לנו 2 שלפנינו -

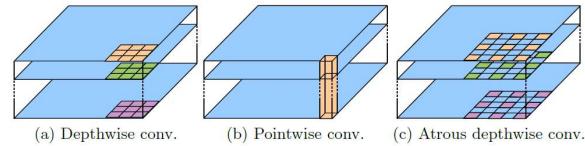


איור 2 – דוגמא לשימוש בשיטה

אם נרצה שהרשת "תסתכל" רק על הכלים המופיעים בRiboux הצהוב היא תחשב שואלי מדובר במכונית. אך ברגע שהרשת "תתפое" את המים מסביב ואת התמונה הרחבה יותר -- היא תקבל הבנה عمוקה יותר בין שמדובר בסירה ולא במכונית. דבר דומה יתרחש כאשר יהיה מופיע בתמונה של אותו אובייקט בגודלים שונים: האיגוד של התמונות עם המרווחים השונים עוזר לרשות לקשר בין אותם האובייקט.

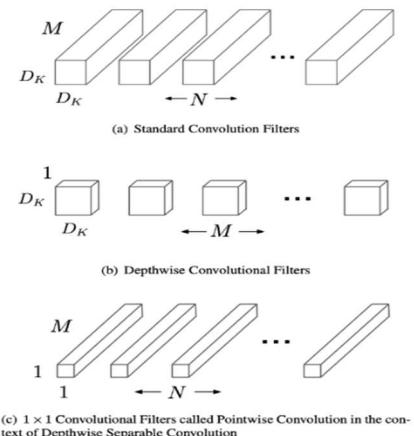
מבחינה טכנית, הפעולות שאנו מבצעים ב- ASPP על התמונה מורכבות מ- 4 קונволוציות מרוחקות. בכל אחת מהן אנו מקבלים תמונה באותו הגודל, אבל בעלת ראייה מרחבית שונה. לאחר כל קונволוציה נבצע גרמול ונפעיל פונקציית אקטיבציה (Relu).

במהלך לביצוע הקונולוציות, נבצע פעולה pooling על התמונה. מימושים שונים למאמר השתמשו ב- Adaptive average pooling על מנת שנוכל לחזור לגודל התמונה ולבצע שרשור בין כל התמונות.



Ajor 3 - Atrous Separable Convolution

בhbet של עליות חישוב, כתבי המאמר משתמשים בטcnיקת **Atrous Separable Convolution** שמטרתה להוריד בצורה ניכרת את מספר הפרמטרים והчисובים שהרשף צריך לבצע.



Ajor 4 - Separable convolution

- **Depthwise convolution** מכילה בתוכה שתי טכניקות עיקריות – **Pointwise convolution**.

סביר אותנו בקצרה: ב-**depthwise conv** אנחנו מבצעים את הפעולות על מרחב מימד התמונה (ציר ה x ו-y) וב-**pointwise conv** הפעולות מתבצעות על מרחב העזרץ.

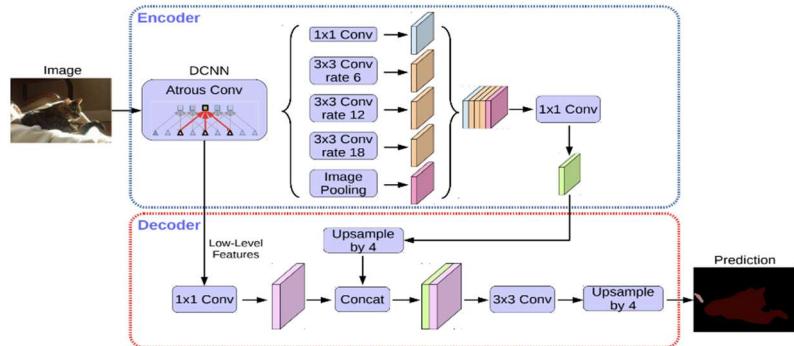
邏輯ית, נניח שה- **input feature maps** $F \in D_F \times D_F \times M$ ושה- **output feature maps** $G \in D_F \times D_F \times N$. במידה והיינו משתמשים בקונולוציה רגילה, ה- **convolution kernel**, הוא $D_K \times D_K \times M \times N$. וכאן עלות החישוב תהיה $D_F \times D_F \times D_K \times M \times N$.

אבל אם נבחר להשתמש ב- **separable convolution**, ראיית נבצע conv **depthwise**, בה פילטר הקונולוציה הוא מימד $M \times D_K \times D_K \in K$. כלומר, במצב זה אנחנו פועלם רק על מימד התמונה ולא על מימד העזרץ. לכן הפעולות תהיה $D_F \times D_F \times D_K \times D_K \times M$. לאחר מכן conv **pointwise** בה אנחנו פועלם בימייד העזרץ. פילטר הקונולוציה במקורה זה יהיה בגודל $1 \times 1 \times M \in K$ ולכן עלות החישוב תהיה $N \times M \times N$. אם נחבר את שניהם עלות החישוב תהיה:

$D_F \times D_F \times D_K \times M + D_F \times D_F \times M \times N$, ניתן לראות שבניגוד לكونולוציה הרגילה, במצב זה נוצר חיסכון משמעותי בעליות החישוב ובמספר הפרמטרים.

כאשר נשתמש בקונולוציה מרוחקת נקבל חיסכון.

ארQUITטורות מקודד מפענח (encoder decoder)



אייר 5 - ?

نبיט על ארכיטקטורת המקודד ונראה שכותבי המאמר למעשה העתיקו את המקודד שבו השתמשו בראשת Deeplab V3 המקודד מחולק ל-3 חלקים:

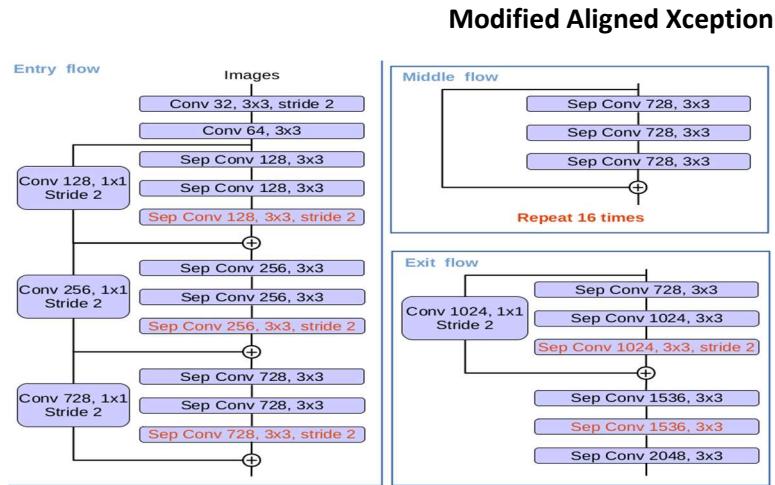
1. ראש שמתארת לחץ פיצרים מהתמונה - במאמר משתמשים בורסיה של Xception Aligned (נרחיב עליה בהמשך). בנוסף לכך ראש ה- backbone מוחזירה גם low level feature שאותו היא מקבלת לאחר ה- entry flow.
2. לאחר מכן אנו מבצעים ASPP על התמונה.
3. לבסוף ביצוע קונבולוציה 1X1 על מנת לעבור למספר העורצים הרצוי.

לעומת זאת המפענה עבר שינוי דרמטי מכיוון שכותבי המאמר ראו שהשחזר של התמונה לגודל המקורי לוקה בחסר ופוגע ביכולת הדיווק של הראש, שלוש הטכניקות שבן הם השתמשו:

1. שימוש ב- low level feature לשחזרו.
2. שימוש ב- upsamples בהדרגה.
3. הפעלת קונבולוציה 3X3 על מנת לקבל אובייקטים חדשים וברורים יותר.

מבחן טכנית הסכמה של המפענה מתבצעת כך:

- מתקבלת תמונה מהמקודד וعليה ניתן לבצע upsampling פי 4 (לדוגמה עם התמונה היא בגודל 32X32 היא עצשי תהיה בגודל 128X128).
- לאחר מכן, יש לבצע קונבולוציה 1X1 על הקולט שהתקבל מה- low level feature על מנת להקטין את גודל מימד העроз.
- לאחר מכן, יש לבצע שרשור של התמונה עם ה- low level feature.
- יש לבצע קונבולוציה 3X3 כדי לחזק את האובייקטים שבתמונה נוספת, מספר העורצים שבתמונה ישנה למספר הקטגוריות שיש לנו ב- dataset.
- לבסוף יש לבצע upsampling נוסף לתמונה בגודל פי 4 על מנת לחזור לגודל המקורי של התמונה.



איור 6 - Modified Aligned Xception

כותבי המאמר בחרו להשתמש בראשת זו כ- backbone ידועה בביטויים טוביים ומהירים במשמעות הקוליסיפיקציה ועם עלויות מואוד נמוכות. כמו כן, Aligned Xception הוכיחה את עצמה בביטויים טוביים במשמעותה - object detection. כותבי המאמר ביצעו מספר שינויים קלים בראשת שיטאותו ספציפית למשימה שלו.

- במקום max pooling אנו מבצעים convolution stride 2 עם atrous separation convolution.
- לאחר כל ביצוע של conv 3X3 אנו מבצע normalization ואקטיבציה.

ביצועים

א- Miou 89 עם PASCAL VOC 2012 Test Set state of the art חדש גם ב- Deeplab V3+
ב- Miou 82.1 עם Cityscapes dataset

9.3 Face Recognition and Pose Estimation

9.3.1 Face Recognition

אחד מהישומים החשובים בראייה מוחשבות הינו זיהוי פנים, כאשר ניתן לחלק משימה זו לשלושה שלבים:

- .1 Detection – מציאת הפרצופים בתמונה.
- .2 Embedding – מיפוי כל פרצוף למרחב חדש, בו המאפיינים שאינם קשורים לתיאור הפנים (למשל: זווית, מיקום, תארורה ועוד) אינם מופיעים על הייזוג.
- .3 Searching – חיפוש במAGER של תמונות למציאת תמונת פנים הקרובה לתמונה הפנים שהוחלזה מהתמונה המקורי.

גישה פשוטה, כמו למשל בניית מסגר המכיל מספר יציאות כמספר הפנים אותן רוצים לזהות, הינה בעייתית מעתה סיבות עיקריות: ראשית יש צורך באלפי דוגמאות לכל אדם (שלא ניתן בהכרח להשיג). כמו כן, נדרש ללמד את

המערכת חדשה בכל פעם שורצים להויספ מישוה חדש. כדי להתגבר על בעיות אלו מבצעים "למידת מטריקה" (metric learning) בה מזקקים מאפיינים של פנים ויצרים וקטור יחסית קצר, למשל באורך 128, המכיל את האלמנטים המרכזיים בתמונה הפנים.Cutout נפרט את שלושת השלבים:

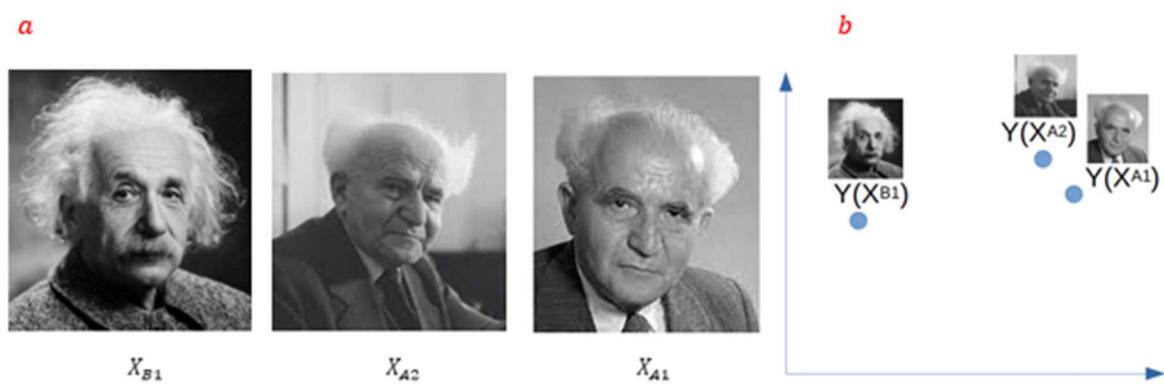
1. מזיאת פנים.

כדי למצוא פרצופים בתמונה ניתן להשתמש ברטשות הממציאות detection, כפי שתואר בפרק 9.1. שיטה מקובלת למשימה זו הינה *YOLO*, המבוססת על חלוקת התמונה למשבצות, כאשר עבור כל משבצת בוחנים האם יש בה אובייקט מסוים, מהו אותו אובייקט, ומה ה-*bounding box* שלו.

2. תיאור פנימ.

כאמור, המשימה בתיאור פנים נעשית בעזרת metric learning, כאשר הרעיון הוא לחקק פנים לוקטור שאינו מושפע ממאפיינים שלא שייכים באופן מהותי לפנים הספציפיות האלה, כגון זווית צילום, רמת תאורה וכו'. בכך לעשות זאת יש לבנות רשת המקבלת פנים של בנאדם ומחזירה וקטור, כאשר הדרישה היא שעבור שתי תמונות של אותו אדם יתקבלו וקטורים מאד דומים, ועבור פרצופים של אנשים שונים יתקבלו וקטורים שונים. למעשה, פונקציית loss-
תתקבל בכל פעם minibatch, ותעניש בהתאם לקרבה בין וקטורים של אנשים שונים וריבוק בין וקטורים של אותו אדם.

icut nich shish lnu kllt X , hamekil aosoF Fretzofim. Kl' iSh yosomn baot achrt - A, B, C , v'tmzonot shonot shel adam yosomno ul ydi otot v'mespar, kr shl'mshl X_{A1} zohi htmonha hrashona shel adam A b'sut hkllt X , v'cmobon sh- $X_{A2} X_{A1}$ v- X_{A2} han sheti tmzonot shel adam. Baofen grfi, b'do-mmd nitn ltaher zat kr (b'po'el hokktorim hmizgim pnim ihui b'mmd gboha yotra):



איור 9.10 a) דוגמאות מסת הפרצופים X. b) איך נרצה שהדעתה ימופת למדוד חדש Y.

כאמור, נרצה לבנות פונקציית loss שמעודדת קירבה בין X_{A1} ו- X_{A2} , וריחוק בין X_{B1} ו- X_B . פונקציית loss מורכבת משני איברים, המודדים מרחק אוקלידי בין וקטורים שונים:

$$L = \sum_X \|Y(X^{Ai}) - Y(X^{Aj})\| - \|Y(X^{Ai}) - Y(X^{Bj})\|$$

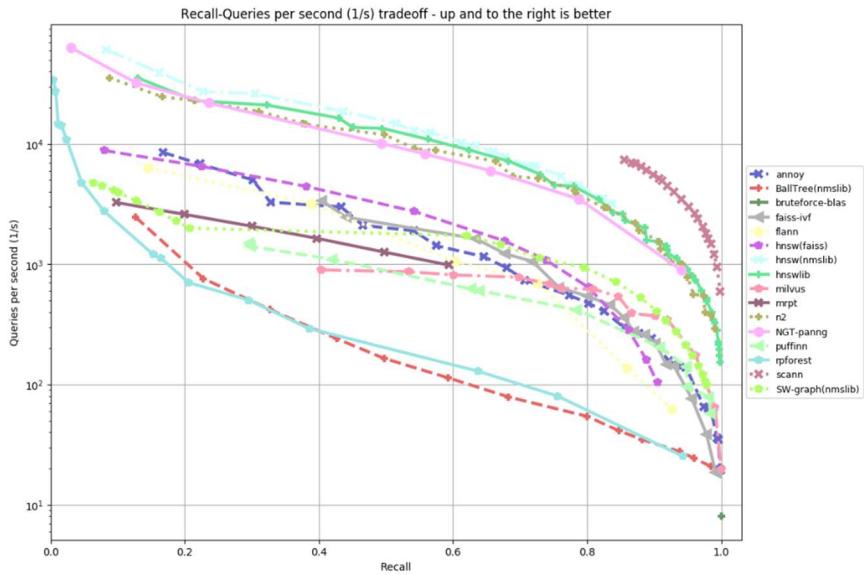
כאשר האיבר הראשוני ינסה להביא למיניהם וקטוריים של אותו אדם, והאיבר השני ינסה להביא למיניהם וקטוריים של פרצופים שאינם שייכים לאותו אדם. כיוון שנרצה להימנע מתקבל ערכאים שליליים, נוסיף פונקציית מקרים. בנוסף, ניתן 'להרחיק' תוצאות של פרצופים שונים על ידי הוספה קבוע k , כך שהפרש בין המרחק של פרצופים של אנשים שונים לבין המרחק של פרצופים של אותו איש יהיה לפחות k :

$$L = \sum_y \max(\|Y(X^{Ai}) - Y(X^{Aj})\| - \|Y(X^{Ai}) - Y(X^{Bj})\| + k, 0)$$

loss צה נקרא triplet loss, כיוון שיש לו שלושה איברי קלט – שתי תמונות 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 לווקטוריים דומים עד כמה שניתן, ואילו פרצופים של אנשים שונים יקבלו וקטוריים רחוקים זה

3 מזיאת האדם

בשלב הקודם, בו הבצע האימון, יצרנו למשה מאגר של פרצופים במרחב חדש. כעת כשייגע פרצוף חדש, כל שנוטר זה למפות אותו למרחב החדש, ולחפש למרחב זה את הווקטור הקרוב ביותר לוקטור המציג את הפנים החדשות. בכך לעשות זאת ניתן להשתמש בשיטות קלאסיות של machine learning, כמו למשל חיפוש שכן קרוב (כפי שהואסביר 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 הדיוק). בעזרת השיטה המובילה כרגע (SCANN) ניתן להגיע לכמה מאות חיפושים שלמים בשניה (הchiposh ב-100 ממדיים מתוך מאגר של 10000 דוגמות).



איור 9.11 השוואת ביצועים של שיטות חיפוש שונות. עבור פרצוף נתון, מחפשים עבורי וקטור תואם במדד החדש המכיל יצוג וקטורי של הפרצופים הידועים. בכל שיטה יש טריד-אוף בין מהירות החיפוש לבין הדיוק.

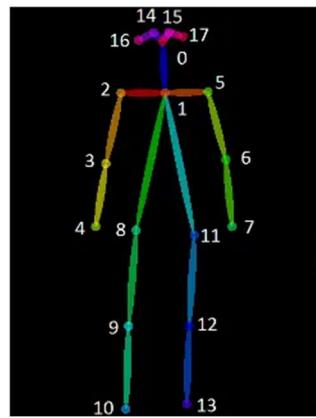
מלבד זיהוי וסיווג פנים, יש גם שיטות למציאת אלמנטים של פנים היכולות אף, עיניים וכו'. אחת השיטות המקובלות משתמשת בשערור הצורה של פנים אנושיות, וניסיון למציאו את איברי הפנים לפי הצורה הסטנדרטית. בשיטה זו ראשית מבצעים יישור של הפנים והתאמת לסקרה אנושית (על פי מרחוק בין האיברים השונים בפנים), ולאחר מכן מטילים 68 נקודות עניין מרכזיות על התמונה המיישרת, מתוך ניסיון להתאים בין הצורה הידועה לבין התמונה המבוקשת.



איור 9.12 זיהוי אזוריים בפנים של אדם על ידי התאמת פנים לסקירה אנושית והשווואה לבניה של פנים המכיל 68 נקודות מרכזיות.

9.3.2 Pose Estimation

ישום פופולרי נוסף של אלגוריתמים השייכים לראייה ממוחשבת הינו קביעת תנוחה של אדם – האם עומד או יושב, מה התנוחה שלו, באיזה זווית האיברים נמצאים וכו'. ניתן להשתמש בניתוח התנוחה עבור מגוון תחומים – ספורט, פיזיותרפיה, משחקים שונים ועוד. לרוב, תנוחה מיוצגת על ידי המיקומים של חלק גוף עיקריים כגון ראש, כתפיים, מרפקים וכו'. ישנו כמה סטנדרטים מקובלים, למשל COCO, posenet, COCO人体pose ועוד. ב-COCO התנוחה מיוצגת באמצעות מערכת של 17 נקודות (בדו-מידה):



איור 9.13 מיפוי תנוחה ל-17 נקודות מרכזיות.

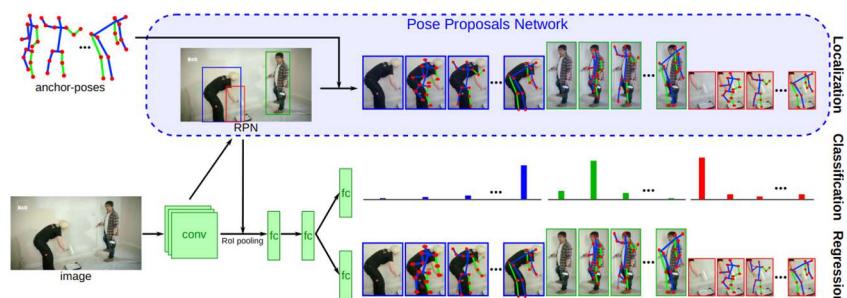
שאר הפורטטים דומים; מוסיפים עוד מידע (למשל מיקום הפה), משתמשים בתלת ממד במקום בדו ממד, משתמשים בסידור אחר וכך. כאשר רוצים לאסוף נתונים על מנח גוף שונים, ניתן לשים חישנים על אותן נקודות מרכזיות וככה לקבל מידע על מנה הגוף לאורך זמן.

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 שמתארים את המיקומים של 17 הנקודות בפורטט COCO בדו-ממד). במקרה זה ניתן לעשות למידה סטנדרטית לחולstein של בעיית גרגסיה בעלת 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 הגוף מוצאת את הפרטים ולאחר מכן מכלילה 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 באזורי שונה בתמונה, כיון שאין צורך לשירות איברים לאנשים.

רשות פופולרית לקביעת תנוחה נקראת *pose estimation*, המורכבת למעשה משתי תת-רשתות ופועלת 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 הוקטוריהם השונים מצביעים כלפיו איבר הגוף הבא בתווך (למשל זרוע מצביעה לيد), וכן ניתן לשירות איברים שונים אחד לשני, ואת כל ייחד לגוף מסוים.

רשות פופולרית אחרת, הפועלת בגישה bottom-up, 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 מתקבל ציון המציג את טיב השערור של העוגן והנוחה של האדם מתבצע clustering לכל העוגנים, ככלומר כל עוגן מקבל ציון המציג את השערור הסופי בעזרת מיצוע של הרובה העוגנים. הנמצא בתוכו. השלב השלישי מפשט את העוגנים ומשקל את טיב השערור הסופי בעזרת מיצוע של הרובה העוגנים. שלושת השלבים משתמשים ברשת קוונבולוציה משותפת, כמפורט באירא.



איור 9.14 Localization-Classification-Regression 9.14

9.4 Few-Shot Learning

יכולת הצלחתם של אלגוריתמי למידה עמוקה נשלמת על כמות ואיכות הדטה לאימון. עבור משימת סיווג תמונות (Image Classification), נדרש שעבור 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 לגרום הטיה בתוצאות הסיווג לטובת הקטגוריות להן יש יותר דוגמאות. פרק זה עוסק בשיטות כיצד ניתן להתמודד עם מצבים בהם הדטה אינם מאוזן.

9.4.1 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 הרשת. באופן פורמלי, קיימות קטגוריות הנקראות קטגוריות בסיס (base classes), עבורן יש כמות גדולה של דוגמאות, ובנוסף ישן קטגוריות חדשות (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 החדשות הקיימות סך הכל. כל בעיה מוגדרת על ידי "k-shot n-way learning" (k-shot learning), ולמשל "5-way one-shot learning". מבחן בו יש חמישה קטגוריות חדשות, ומכל אחת מהן יש רק דוגמא אחת לאימון הרשת. בכלל, בקטגוריות הבסיס תהייה כמות גדולה של דוגמאות. למשל בסט התמונות האופני לעיבוט藻, mini-ImageNet, יש 600 דוגמאות לכל קטגוריה בסיס ולרוב 1-5 דוגמאות עבור הקטגוריות החדשניות.

האתגר בלמידה ממיינט דוגמאות נובע מה הצורך להכין לרשת כמות ידע קטנה נוספת הנרחבת הקאים, תוך הימנענות מ-overfitting כתוצאה מכמות הפרמטרים הגדולה של הרשת לעומת כמות המועיטה של הדטה. לכן, גישה נאיבית כמו אימון מחדש של רשת על מעט דוגמאות נוספת יכולה ליצור הטיה בתוצאות.

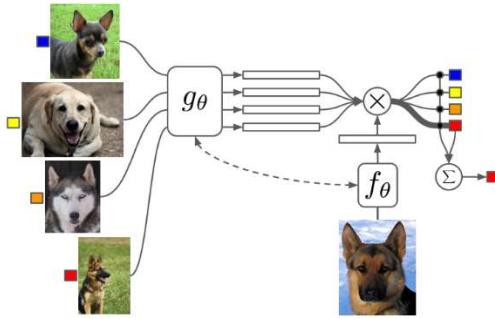
יש לציין כי בכל בעית הלמידה ממיינט דוגמאות עבור האימון, אף שהשאיפה היא להצליח באופן זהה בזיהוי כל הקטגוריות בשלב המבחן, בו לא יהיה חוסר איזון. لكن בעיות אלו לוונטיות לשימושים רבים כמו: זיהוי חיות נדירות באופן זהה לחיות יותר נפוצות, מערכת זיהוי טילים שצריכה להתמודד גם עם אינויים נדירים יותר (ניתן לחשב למשל על פצצת אוטום), מערכות זיהוי פנים שצריכות לעבוד טוב עבור כל אדם ללא תלות בדатаה שהיא קיימת באימון הרשת.

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 ריעונות השאבים ממספר גישות יחיד אך נשענים על האלגוריתמים המשמעותיים מהעבר. לבסוף, נציג את התחום של Zero-Shot Learning, כלומר יכולת למידה של קטgorיה חדשה כאשר לא קיימת אף דוגמא שלא לאימון.

9.4.2 Metric Learning

שיטות להתמודדות עם למידה ממיינט דוגמאות הנוקטות בגישת למידת מטריקה, שואפות לייצג את הדוגמאות כווקטורים של מאפיינים מרוחב רב-ממד', כך שניתן היה למצוא 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 הווקטורי של דוגמאות מאותה הקטגוריה (intra-class similarity).

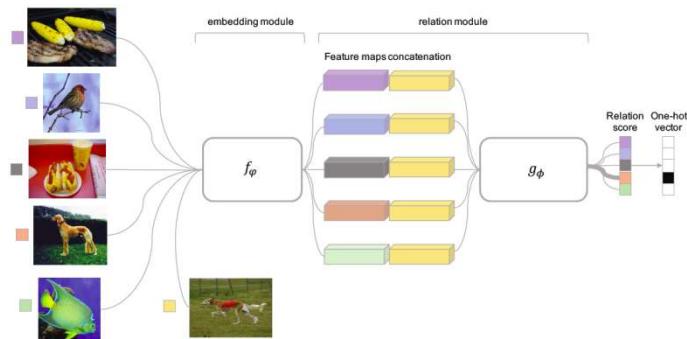
התקדמות משמעותית של שיטות אלו הוצאה במאמר Matching Networks for One Shot Learning בשנת 2016. שיטה זו משתמשת בזיכרון שהגישה אליו נעשית באמצעות מגנן Attention, על מנת לחשב את ההסתברות של דוגמא להיות שייכת לכל קטגוריה, בדומה לשיטות השכן הקרוב (Nearest Neighbors).



איור 9.15 אילוסטרציה של שיטת Matching Networks

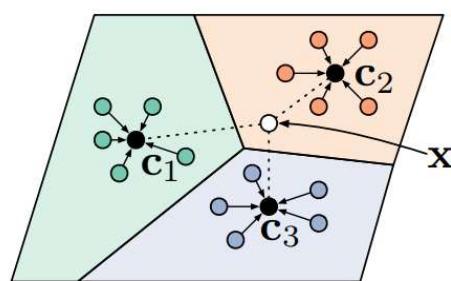
ההידוש המשמעותי בשיטת האימון המבוצעת באפיוזות (Episodes). בשיטה זו האימון מכיל כמות של שימושים, כאשר כל שימוש היא למעשה מבחן של הדאטה שבו יש קטגוריות מסוימות שהן חדשות ואחרות מהן קטגוריות בסיס. על ידי דוגמאות רבות ויצירת שימושים מלאכותיים כאלה, בהן בכל פעם נלקחות קטגוריות אחרות ליצג את החדשניות, מתבצע אימון המתאים לבעה של מיעוט דוגמאות. שיטת אימון באפיוזות הפכה לנפוצה ביותר בלמידה ממיעוט דוגמאות, גם בגין הatrixה שונראת 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 נכונים למודל המשערך מודד דמיון בעזרתו ניתן לסואג את דוגמת המבחן.



איור 9.16 Learning to Compare: Relation Network for Few-Shot Learning

שיטה נוספת המשמשת נוספת הנקוטה בגין למידת מטריקה נקראת Prototypical Networks. בגין 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 מסווג 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 שהזכרנו על הקטגוריות החדשניות. יש להזכיר כי כל השיטות מגינות לאחיזה דיק נמכרים משמעותית מאחיזה הדיק המדוחה במרקם של איזון בין כמה דוגמאות 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.

9.4.3 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 הרענון 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. לצורך, כל משימה באימון היא אפיוזה שבה קטגוריות מסוימות נבחרות רנדומלית לדמות את הקטגוריות החדשוויות. בכל משימה צזו גלמים פרמטרים של המודל האגנוטטי כך שעದכון בכיוון הגראדיינט יוביל להתאמאה לשימה החדש. הכותבים מצינו שמנקודות מבט של ממערכות דינמיות, ניתן להתבונן על תהליך הלמידה שלהם כenza שמנקנסם את רגישות פונקציית המחיר של משימות חדשות ביחס לפרמטרים. כאשר הריגשות גבוהה, שניוי פרמטרים קתנים יכולים להוביל לשיפור משמעותו במחיר של המשימה. מתמטית, פרמטרי המודל, המיצגים על ידי θ , משתנים עבור כל משימה i להיות θ'_i , כאשר:

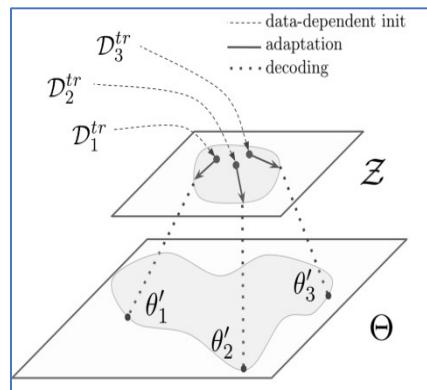
$$\theta'_i = \theta - \alpha \nabla_{\theta} L_{T_i}(f_{\theta})$$

עבור פונקציית מחיר L והיפרפרמטר α . כאשר מבצעים מטא-למידה לעדכון הפרמטרים, מחשבים למעשה SGD:

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{T_i \sim p(T)} (L_{T_i}(f_{\theta}))$$

כאשר המשימות דוגמאות מトー (T) k - β הוא גודל הצעד של המטא-למידה.

שיטה מעניינת נוספת בשם (LEO) Latent Embedding Classification מימושת את הגישה של מטא-למידה במרחב יציג לטנסי בעל ממדים נמוכים, ולאחר מכן עבר בחרזה למרחב המאפיינים הרב ממד.



איור 9.19 שיטת (LEO) Latent Embedding Classification

השימוש במרחב מאפיינים בעל ממדים נמוכים המשמרים את המאפיינים החשובים לייצוג הקטגוריות, שיפר 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 ביצוע דיק של שיטות מטא-למידה על קטגוריות חדשות עבור mini-ImageNet.

9.4.4 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 תאורה וקונטרסט, שינוי סקללה, שינוי צוויות, ואף הוספת רעש רנדומלי. כל אלו הראו שיפורים ביכולות הרשות ללמידה קטגוריות שהו במצב חוסר איזון. דרך נוספת היא שימוש ברטיביות (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 התיאוגים 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 למכונת "להבין" את התוכן הטקסטואלי, ואת הニアנסים והקשרים של השפה. עלם ה-NLP מורכב ממספר רב של תמי משנהות, כגון ניתוח סנטימנט של טקסט (Sentiment analysis), תמצאות/סיכום אוטומטי של טקסט (Text summarization), מציאת תשובה עבור שאלה מסוימת (Question answering) ועוד. ועוד משימות רבות אחרות. פיתוח כלים המסייעים להתחמוץ עם משימות אלה יאפשר לנו (בין היתר) לphet אפליקציות שיעזרו לנו ביום יום כגון עוזרות קוליות, מערכות תרגום, מערכות אוטומטיות לבדיקת דקדוק ועוד. כמה דוגמאות לאפליקציות כאלה כולנו מכירים הן Siri, העוזרת הקולית של אפל, הרשומה 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 ויזוגי מילים: נראה כיצד מייצרים מודל שפה מדatta טקסטואלי (שנקרא גם "Corpus", וכיצד מייצגים את הטקסט כך שהיא 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 ההסתברות של כל רצף אפשרי להופיע. מודל כזה אפשר 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 המורכב מכל הסדרות האפשריות 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 המילה Take , כפולה ההסתברות של המילה corpus להופיע אחרי הציגוף Take a big .

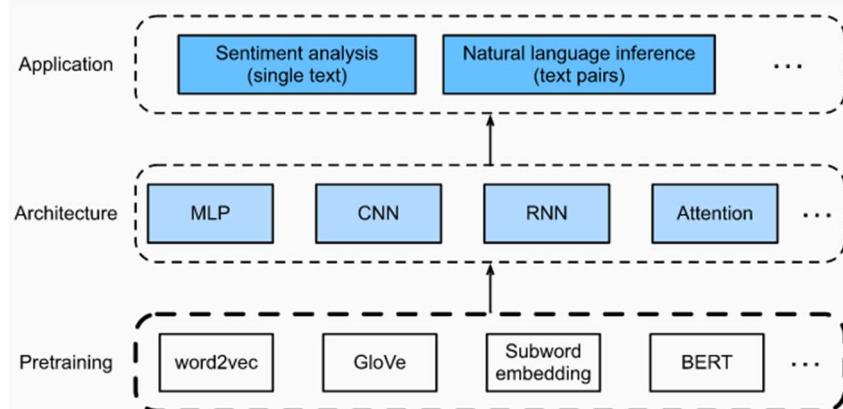
באופן הפשטני ביותר נוכל לשערר את ההסתברויות האלו באופן הבא: ניקח corpus מספק גדול ועшир בעל N מילים שונות, כמו למשל כל ערבי וקיידי, ופושט נספר את כל המילים והציגופים שמופיעים בו. ההסתברות של כל מילה תהיה אחוֹז הפעמים שהיא מופיעה בטקסט, וההסתברות המותנית תחוושב באופן דומה – על ידי מנתית מספר המופיעים 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 שניציר 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, נctrkr קודם כל לייצג את הטקסט בצורה כלשהיא. בשיטה שהוזגה לעיל, כל מילה (Token) מוצגת באמצעות צירוף אותיות. כך למשל המילה הראשונה במשפט

שראינו מרכיבת מצירוף של האותיות e, T. כפי שיפורט בהמשך, נוכל גם לדבר על ייצוג למילים עצמןvr כר שכל חידה אוטומטית תהיה מילה וצירוף של היחידות האלה ירכיבו ייצוג של משפט. באופן סכמטי, ניתן לתאר את מושגית עיבוד השפה מקצה למקצה באופן הבא:



איור 11.1 תהlixir פיתוח אלגוריטם של מודל שפה: א. מייצגים את הטקסט בזורה כלשהיא (ניתן כמובן לקחת ייצוג קיימש שנבנה על בסיס דאטסהט אחר). ב. מאמנים מודל שמקבל כתובע את הטקסט אותו יציגו בדרך כלשהיא בשלב הראשוני, והמודל מוציא output מסויים. למודל זה יכולות להיות ארכיטקטורות שונות. ג. באמצעות המודל המאומן ניתן לבצע שימושים קצה שונים.

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 שימוש.

10.1.1 Basic Language Models

מודל השפה הראשון אותו נציג הינו N-Grams, שהוא מודל סטטיסטי המניח שהסתברות למילה הבאה תליה אר-ורק ב-1 – N המילים שקדמו לה בסדרה. הנחה זאת נקראת "הנחה מרקוב" (Markov assumption), ובאופן כללי יותר, מודלי מרקוב (או שרשרת מרקוב במרקלה הדיסקרט) מסדר n הם מודלי הסתברות המניחים שנייתן לחזות הסתברות של אירוע עתידי T בהתבסס על האירועים שהתרחשו ב- n נקודות הזמן שקדמו למועד T מבלי להתחשב באירועי עבר רוחקים מדי.

מודל-h-N-Gram הפשט ביוטר נקרא unigram. במודל זה אנחנו חוזים את המילה הבאה לפי התדריות של המילה עצמה – w_n מבלי להתחשב بما שקדם לה. כמובן שיחסיו כזה הינו בעייתי, מכיוון שהמילה הבאה **חייבת להיות תליה במילים שקדמו לה**, וכך מילום כמו the שפיעות באופן תדיר בטקסט שאין בהכרח שפיעות על ההקשר. לכן, נסתכל על מודל קצת יותר מורכב 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 משפטיים וננתן אותם בשיטת bigram:

- I know you
- I am happy
- I do not know Jonathan

נכיה ונרצה לבדוק את ההסתברות שהמילה Jonathan היא המילה הבאה אחרי הסדרה do I. ראשית נגדיר את המילון, המכיל את כל המילים האפשריות 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} קיימים בטקסט, וכך נוכל לפשט את המשוואה האחרונות ורשום במקומה:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

אם כך, ההסתברות לכך שהמילה $know$ 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 במיללים שלא נתקלו בהן בטקסט האימון וכן לא נוכל להגיד את ההסתברות עבור *-N-grams* המכילים מיללים 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 ונסקרו 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 ש-*n-grams* מסוימים מופיעים מועט של פעמיים (או אפילו לא מופיעים בכלל). אם למשל יש צירוף ש-*n-grams* רק פעם אחת ולעומתו יש צירוף שנמצא 1000 פעמים, אז הוספה הקבוע \mathcal{K} משמעותה שעכשיו שונח 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 הוספה 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 שניתנו לבחור כל ערך \mathcal{K} שרצים ואותו להוסיף למיניה. באופן זה כן ניתן לשנות בrama מסויימת עד כמה להגדיל את ההסתברויות לקומבינציות שאינן מופיעות בסט האימון על חשבון הקומבינציות השכיחות (כתלות-ב- \mathcal{K}). בחירה למשל של $\mathcal{K} = 0.5$ מאפשרת לפחות את העיונות יכול להתקבל ממשני הספרה.

Backoff and Interpolation

שיטת החלוקת בסעיף הקודם נותרה לקביעת הסתברות של מאורעות המקבילים הסתרות 0, וישן גישות נוספת לשונות מענה למוגבלות האלה. נניח ואנחנו משתמשים במודל trigram, מודל המניח שהסתברות למילה הבאה תלולה בשתי המילים שבאותם לפנייה. אם נבצע את השערוך של כל שלישיה לפי הגדרה (=ספירת המופיעים שלהם בטקסט), כל שלישית מילים שאינה מופיעה כראץ בטקסט תקבל הסתרות 0, ובuczם תקיים את המשוואה:

$$P(w_n | w_{n-1}, w_{n-2}) = 0$$

בכדי להימנע מלחת הסתרות 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 ולשערר אותם מחדש באמצעות מודל bigram. לאחר מכן את כל המילים שעדיין נותרו עם הסתרות 0 (כלומר, צמדי מילים שאין מופיעים בטקסט) ונשערר אותם באמצעות unigram. באופן זה מקווים להגיע לפחות במספר המילים בעלי הסתרות 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 הסתרות 0 ולשערר אותם במודל הנמצא בדרגה אחת מתחת (למשל – לשערר בעזרת bigram במקומם trigram), המודל הראשון מתיחס לקומבינציה כלשהי של המילים – למשל קומבינציה ליניארית). בשיטה זו גם מקרים שאינם בעלי הסתרות 0 נעזרים ב-unigram, bigram and trigram בוגר unigram and bigram.

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 סינטקטיים (=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 חיית מחמד שאנשים מגדלים בדירותם. מודל המבוסס על יציג טקסטואלי של השפה לא יכול לחזור להבנה סמנטית שלה, ולכן נרצה שהייצוג שלנו יהיה מספיק עשיר ויכיל הן מאפיינים תחביריים והן מאפיינים סמנטיים של השפה.

בפועל, מוכן למפות את הייצוג הטקסטואלי לייצוג נומירי בצורה פשוטה בעזרת One-Hot vectors – מערכת בגודל המילון שלנו, המיצג כל מילה במיון בעזרת 1 באיבר המתאים במערך. לדוגמה נתון המילון הבא:

Index	Word
0	Dog
1	Cat
2	Lion

מוכן 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 פשטי זה הינו בעייתי מכיוון שייהיה קשה ללמידה ממונו מאפיינים סמנטיים. כדי להבין את הסיבה לכך ראשית יש להגדיר 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_1, w_2, \dots, w_N , שניים בעלי אותו הממד N . המכפלה הפנימית (dot product) בין הווקטורים אלה מוגדרת באופן הבא:

$$w_1 \cdot w_2 = v^T w_2 = \sum_{i=1}^N v_i w_i$$

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: $[1, 0, 0] \rightarrow [1, 0, 0]$. לכן, לכן המכפלה הפנימית במרחב זה תהיה:

$$[0, 1, 0] \cdot [1, 0, 0] = 0 \cdot 0 + 0 \cdot 1 + 1 \cdot 0 = 0$$

תוצאה זו מדגימה את הביעיותו בייצוג פשוטי זה, מכיוון שה邏ים רוצים שווקטוריהם אלה כן יהיו דומים במובן מסוים, ולא שהתוצאה 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 שמליה אחרת נמצאת אליה באותו הקשר. שיטה זאת נקראת Bag-of-Words, ולפניהם שוכן להדגים אותה, נctrar להגדיר מהו הקשר של מילה. באופן פשוט הקשר של מילה זה המשפט בו היא מופיעה, אך ניתן גם לנקח רק חלון של מספר מילים מתוך המשפט (לרוב אורך החלון קטן מאורך המשפט). לדוגמה, נניח 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.

מילים:

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. במשמעותם של המילים לפני ואחרי המילה שנרצה לייצג. חלון זה מסומן בטקסט בסוגרים מרובעות בצלב 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 נוספת הנובעות מפשטות פתרון זה:

1. מילים פחות משמעותיות כמו is, the, a, to, in ועוד שמן לא בהכרח מושיפות מידע משמעותי לייצוג.
2. מילים המופיעות בטקסט לעיתים רחוקות יהיו בעלות ייצוג וקטורי מאד דיליל, מה ש慷慨יל 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 מהmillion. אולם, פתרון זה אינו יעיל, כיון שהתדרות של מילים משתנה בין תחומים/דומיניים שהם נלקח הטקסט. לכן פותחה שיטת ייצוג הנקראת TF-IDF, ומטרתה להפחית את רוש זה באופן אוטומטי.

בשיטת TF-IDF, מגדירים פונקציה ניקוד אחרת על מנת שמיילאים ייחיד 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 מופיעעה בקונטקסט (=בחלון) של המילה המיוצגת w (על התוצאה מפעלים \log בשבייל rescaling של ערכי גובהים מאוד).

עד כה השיטה אינה שונה במאובטת ממספרה כמו שראינו בwords-of-Bag, 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 מופיעעה בקונטקסטים אחרים, בעוד N מייצג את כמות המילים במילון. נשים לב שגם מילה מסוימת, למשל the, מופיעעה בכל הקונטקסטים של כל המילים במילון, אך הביטוי בתוך הלוג יהיה 1 וכן df יהיה 0. לעומת זאת, אם מילה מסוימת מופיעעה אך ורק בקונטקסט יחיד, אז הערך שהוא בתוך הלוג יהיה N .

לבסוף, TF-IDF מוגדר באופן הבא:

$$TF - IDF = tf \cdot idf$$

מדד משוקל זה מצליח עבור ייצוג של מילה מסוימת לתת משקל גדול למילים אחרות הנמצאות אליה בקונטקסט באופן תדירים 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 בונפרד – $\frac{P(x,y)}{P(x)P(y)}$. 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 y, x ביחד שווה לכפל ההסתברויות לראות כל אחת לחוד, נקבל שערך הביטוי הוא $\log(1)$ כלומר 0. לעומת זאת, אם הסיכוי לראות את המילים האלו 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 שהעריכים החלילם 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 וקטוריים מאוד דילילים, ובנוסף לכך גודל הווקטורים תלוי בכמות המילים שיש לנו במילון, מה שיוצר וקטוריים גדולים שמכבידים על החישובים במשימות השפה השונות. למשל – ראיינו קודם שנדעתן ליציג מילה באמצעות וקטור שכלו אפסים למעט תא אחד עם הערך 1 במקומות ייחודיים למילה. עבור שפה עם אלפי מילים ואף יותר מכך, כל וקטור המיציג מילה הוא באורך עצום, ועם זאת הוא מאוד 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 (supervised learning) "דונה" למדידה מונחית (Self-Supervised learning) רגילה, אך התוצאות אינם נתונים אלא נוצרים באופן הדומה ללא מתואג. באופן זה ניתן לאמן מודלים עם כמות גדולה של דאטה לא מותיג בצורה יעה ולא צריך בתיאוג (שעלול להיות מודע לך). בהקשר זה, אלגוריתם Word2Vec משתמש ב-SGNS – Skip-Grams-With-Negative-Sampling באופן של Self-supervision בעקבות סיווג שמרתחה לחזות מה ההסתברות של מילים שונות להיות בקונטקסט של מילה נתונה. בסוף הוא להגדיר בעיות סיווג שמרתחה לחזות מה ההסתברות של מילים שונות להיות בקונטקסט של מילה נתונה. בסופו האימון לוקחים את המושגים שנוצרו בעקבות תהליכי אימון הראשית, והם יהיו הייצוג של המילה. בכך ניתן זאת לעומק, נבחן טקסט פשוט וחסית בעזרת אלגוריתם Word2Vec. נניח ונ天真 המשפט הבא כחלק מהטקסט 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 שלמו דוגמאות חיוביות של המילים שנמצאות בקונטקסט עם tales. בנוסף, כדי למנוע התנונות של כל הייצוגים לוקטור בודד, נדרש "להראות" למודל איך נראות דוגמאות שליליות ולכך השתמש בשיטה הנקראת negative sampling. בשיטה זו דוגמים מהמלון בהסתברות פרופורציונלית לתזדיות המילה (עם תיקון קטן שנשתן 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 דאטה מתואג עברו משימת הסיווג, ונוכל להשתמש בתיוגים אלה בשבל לאמן את המודל. הכוונה במשימת סיווג בהקשר זה מעט שונה מסיווג במובן הפשוט של המילה: מטרת המודל היא שבהינתן מילה *sh* (במקרה שלנו), נרצה שהייצוג של מילים המופיעות באותו קונטקסט עם כל מילה – אלו שופיעו עם

(tales באותו חלון) יהיה קרוב לייצוג של tales בעוד שמלים שאינן מופיעות באותו קונטקסט יהיו בעלות ייצוג "שונה" (mbhinit פנימית או similarity). נניח שהחरנו 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 KonText_c- e_c . השאייפה היא שהמכפלה הפנימית של וקטורי הייצוג של המילים בكونטקסט של w יהיה גבוהה יחסית, בעוד שהמכפלה הפנימית של וקטורי הייצוג של מילים שאינן מופיעות באותו קונטקסט יהיה נמוך. כאמור לעיל, Cosine similarity (המטריקה המגדירה דמיון בין וקטורים) היא בעצם מכפלת פנימית של הווקטורים (=מכפלת dot עם נורמל). כעת נוכל להציג בעית logistic regression באופן הבא:

$$p(+|w, c) = \sigma(e_w, e_c) = \frac{1}{1 + e^{-e_w \cdot e_c}}$$

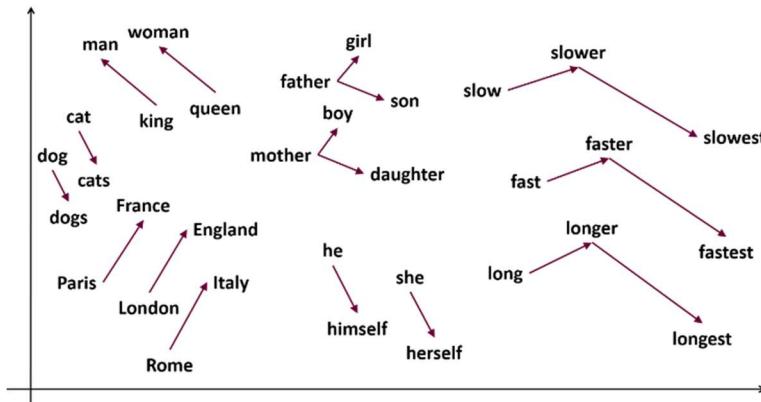
$$p(+|w, c) = 1 - p(+|w, c)$$

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 מכפלה מינימיציה לפונקציית מטרה זו נגרום למכפלה הפנימית בין הייצוג של מילה לבין מילת הקונטקסט להיות גבוהה ובו בזמן למכפלה הפנימית בין וקטורים שאינם בكونטקסט להיות נמוכה. כך הייצוג של המילה tales ("דומה") (= קרוב במשמעותו של המילה "דומה") יוצג של המילים בكونטקסט של המילים בكونטקסט. את התהיליך המינימיציה 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 המילים Queen, Woman. דוגמה נוספת – הווקטור בין שם של ארצות הברית לביירה שלו מקביל ובעל אורכו דומה לווקטור שבין ארצות הברית ועיר הבירה המתאימה. כמו כן ניתן ללמוד מכך על קשרים סמנטיים, כמו למשל שהיחס בין King, Man זהה לזה של Queen, Woman.

10.1.3 Contextual Embeddings

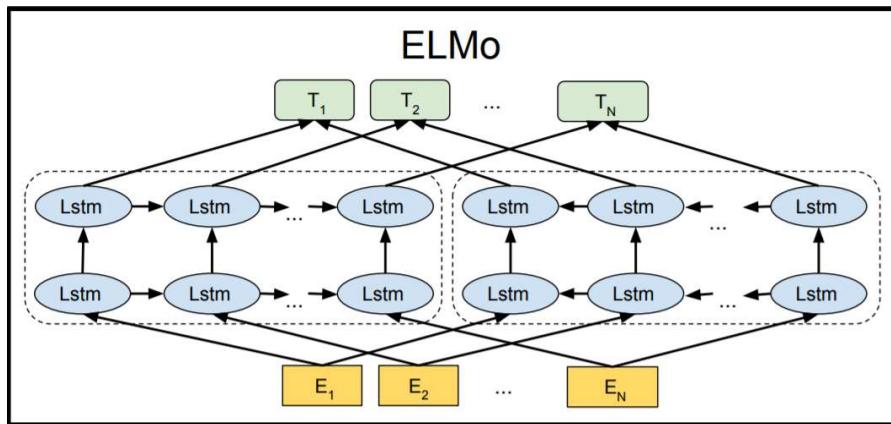
מנגנון בנייה ייצוגי מילים (embedding) שראינו עד כה למדו ייצוג סטטי עבור כל מילה. אך דבר זה יכול להיות בעיתוי מכיוון ששפה טבעית היא דינמית ותלויה בקשר, ולאותה מילים יכולה להיות ככמה פירושים. בשיביל להבין את הבעייתיות נסתכל על המשפטים הבאים:

1. We need to **book** the flight as soon as possible
2. I read the **book** already

למילה **book** יש כפלי משמעויות במשפטים האלו. 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 מגנוניword2vec כמו את המילה באותו וקטור יציג עבור שני המפעעים. לכן נרצה לפתח מנגנון embedding עבור המילים, שבאמצעותו וקטור המיצג של המילה יהיה תלוי בהקשר בו היא מופיעה.

Embeddings from Language Models (ELMo)

אחת 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 ללקח את ייצוג של מילה, להוסיף לו מידע נוסף מההקשר של המילה במשפט ולקבל ייצוג חדש התלי גם בהקשר שלו. בניסוח אחר ניתן לומר Sh-ELMo הינה פונקציה המתקבלת משפט שבו כל מילה מיוצגת בדרך כלל לשאה (למשל – Word2Vec), ומוסיפה 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 ה- t (c_t, h_t), וקלט נוסף של האיבר הנוכחי בסדרה, שבמקרה שלנו זה המילה הנוכחיית (x_t). המוצא של ה-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 מודול שפה וחוזים את המילה הבאה בהינתן המילים הקודמות. אך בשונה מאלגוריתמים אחרים, ELMo משתמש בארכיטקטורה דו-כיוונית, כך שבתהליך האימון משלבת משימת שפה נוספת המנסה לחזות את המילה הקודמת בהינתן הסוף של המשפט. הארכיטקטורה של ELMo בנויה ממספר שכבות של LSTM שמורכבות זו על גבי זו, ולפי כתבי המאמר השכבות התחרתנות 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 משימות 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 עצמן 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 צויל (Fine tune) עבור משימה ספציפית. כך לדוגמא נוכל להתאים את הייצוג של המילים לשימת סיווג 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 והואoperates on it to produce a vector representation of the text**.

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"

כלומר, עת מי שירצה לבצע משימה שפה כבר לא יתבסס רק על ייצוג סטטי של המילים אלא הוא יסתמך על מודל שפה מאומן שיעד לנקח את ייצוג התחלתי של מילים ולהפוך אותם ליצוגים קונטקטואליים. ס.ELMo ועוד מאמריהם 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)

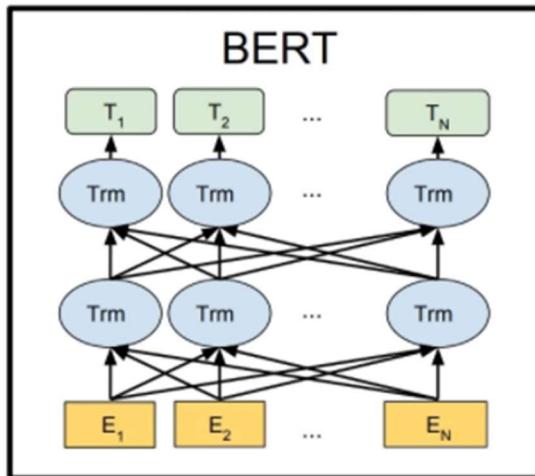
כאמור, הרעיון של ELMo הוא לייצר contextual embedding, ובכך לקבל מודל שפה המאפשר לנקח ייצוג וקטורי של מילים ולהעшир אותו במידע על הקשרן של כל מילה בטקסט. למרות ש-ELMo משתמש בשני היכיוונים של המשפט (חוזה את המשך המשפט מתחילה ואת תחילת המשפט מסוף) הוא אינו לומד משנה היכיוונים בתהילר אחד, אלא צריך לחלק את הלמידה לשני חלקיים שונים. בנוסף, כפי שהסביר בהקדמה לפרק 8, כאשר מתעניינים עם סדרות ארוכות של מילים, וקיטור הייצוג של כל איבר נהיה בעייתי כיוון שהוא מוגבל ביכולת שלו להכיל קשרים בין מספר רב של איברים. במילים אחרות, כאשר רוצים להוציא לפוקטור הייצוג של מילה מסוימת קשור למילים רחוקות, אנו מלאכים את הייצוג "לזכרו" מידע רב, אך הייצוג והוקטור של המילים ב-ELMo אינם מצליחים לעשות זאת בaczורה מסוימת טובה. לכן, על אף הצלחת גישה זו במשימות שונות, היא עדין התקשה במשימות בהן גדרות יכולת לנתח טקסטים ארוכים (כמו למשל משימה של summarization). בנוסף לכך, האלגוריתם יחסית איטי, כיוון 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-attention (–묘сор בחרחה בפרק 8). אחד השימושים הראשונים במנגנון ה-attention עבור משימת עיבוד שפה היה בטרנספורמים, ובפרט בארכיטקטורת רשת הנקראט BERT, המבוססת על ה-encoder של הטרנספורמר המקורי. שימוש זה הראה פריצת דרך בתחום, וכךים ברוח המוחלט של המחבר והפיתוח בתחום ה-NLP משתמשים בארכיטקטורת רשת מבוססת attention. למעשה, BERT מציע שיטה לבניית ייצוג קונטקטואלי של מילים הבא להתמודד עם החולשות הקיימות ב-ELMo. נתאר בקצרה את העקרונות של מנגנון ה-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-*on-to-on*, ניתן לבנות ייצוג של מילה שתלו בקשרים שלה עם מילים הנמצאות רחוק ממנה בקטע טקסט, ככלור ההקשרים המתקבלים בין המילים יכולם להיות מיזגים בצורה טובה גם עבור רצפים ארוכים ומילים שאינן נמצאות בסמיכות יחסית (שכאמור זה היה אחד החסרונות הגודלים של ELMo). בנוסף, מנגנון זה מיותר את הצורך לעבור מילה אחר מילה בקטע טקסט לצורך בניית ייצוג המילים שבו. במקרה מעבר זה, ה-encoder מקבל הקלט את כל קטע הטקסט כמקשה אחת, מה שעשו 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 בודדת 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 ווקטור ייחודי לכל מיקום במשפט ובסיום מבצעים חיבור של הווקטור שנוצר מהקלט והווקטור שנוצר מהמיקום.

הפותחים של BERT ימכו 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 מייצגת 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 במשמעות מסוים ולהוסיף לו מידע על היחס בין המילים השונות שבtekst. תהליכי יצירת המודל היה אמם יקר, אך ניתן ללקחת אותו ויחסית בקלות לכילו אותו ואך להוסיף שכבות בקצת עבורה שימושות שפה שונות.

GPT: Generative Pre-trained Transformer

עם הכניסה של מנגנון attention וטרנספורמרים לעולם ה-NLP, הוצעו יותר ויותר מודלי שפה מבוססי attention. לצורך ההמחשה ניתן לציין שבשנים הבזוקות שעברו 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 מהתחלה של מילה בודדת. אם נרצה לבדוק, המודלים הללו לא תמיד משתמשים במילים מיוחדות, למשל מילויים ואפיו או תווים נקראים טוקנים או אסימונים. דבר זה יכול לעזור לנו בהכללה ולהקטיין את הסיסי לטוקן 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, כאשר בשונה מ-BERT ה-GPT משתמש רק ב-decoder (מנגן ה-self-attention) שמקודד את הפיצרים, והפלט שלו הינו הטוקן 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 בכך לקוד פיצרים מהם נסיק את הטוקן הבא. התהילר זהה מתרבע באמצעות רכיב הנקרה Masked Self attention. בשונה ממנגן self-attention רגיל שמקודד כל טוקן בעזרת הקונטקסט של כל שאר הטקסט, GPT צריך לקודד כל טוקן רק בעזרת הטוקנים שקדמו לו, כיוון שבשלב זה המידע היחיד שהקיים זה הטוקנים שנוצרו עד כה (וכמובן שאין גישה לטוקנים שעדיין לא נוצרו). כאשר מקודדים את הייצוג עבור טוקן מסוים, רכיב Masked Self attention מופיע כל וקטור של טוקן שבא אחריו, כך שהמודול לא יכול ללמידה ייצוג התלו依 מילים שבאות לאחר הטוקן המוצע, אלא עליו להפוך את המירב מהטוקנים הקודמים לו.

כיוון ש-GPT פועל בצורה של auto-regressive, ניתן לאחר האימון ליצור טקסט באמצעותו – ניתן למודל התחלת קיצור של טקסט, ובבקש ממנו ליצור את המילים הבאות. כך בכל שלב ניתן לו לקרוא את הטקסט הראשון ואת הטוקנים שייצר 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 הינה נפוצה למדידת "עוצמה" של מודל שפה הינה perplexity, שזהו מושג הלקווח מהתורת האינפורמציה והוא מודד כמה טוב מודל השפה חוצה את השפה ב-Corpus שאותו ניסינו למודל.

לפנינו שנסביר את המושג באופן פורמלי. ניתן אינטואיציה למה אנו מצפים לקבל מהמטריקה שנבחרה. נניח וננו מבצעים את הפעולה הבאה: ראשית לוקחים משפט שלם וחותכים ממנו את התחלה, ואז לוקחים את אותה התחלה ומכניםים כיקלט למודל שפה ומקשים 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 האמיתי. באופן יותר כללי ניתן למשפט טקסט המכיל כמה משפטים כרצוננו, להכניס חלקים ממנו למודל השפה, ולהשווות את הפלט המתkeletal לטקסט המקורי. כיוון שמודל שפה הינו סטטורי, השוואת הפלט לשפט המקורי בלבד לאינה מספקת, כיון שהיא אינה משקפת בצורה מספיק טוביה את מידת הצלחה שלה. אם למשל במשפט המקורי ה"יתה כתובה המילה "לבנה" ואילו המודל חזה את המילה "רוח", השוואת שתי המילים כשלעצמם מראה לכואורה שהמודל שגה לחוץין, אך בפועל אנו יודעים שמלים אלו נרדפות ולכן הפלט של המודל במקורה 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 בটקסט זה ובודק **מה ההסתברות שמודל השפה ינביא את המילה הבאה בטקסט בהינתן כל המילים שלפניהם**. 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 מנביא 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:

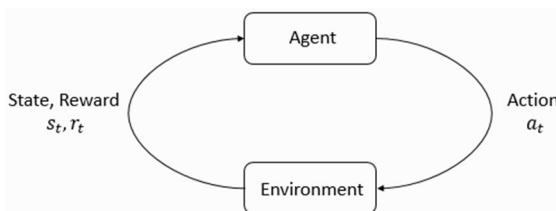
<https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html>

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>Data{} {x = S, כאשר x \in \mathbb{R}^d}, אליו אוסף של אובייקטים (למשל נקודות במרחב, אוסף של תמונות וכו'), ו- y \in \mathbb{R}^n הוא אוסף של labels. לכל אובייקט x \in \mathbb{R}^d יש label מתאים 1 \in y.
2. אלגוריתמים של למידה לא מונחת עבורם הדטה x הוא אוסף של אובייקטים ללא labels, ומנסים למצוא כלים מסוימים על>Data{} {x}. למשל – חולקה לאשכולות, הורדת מד ווуд).

למידה מבוססת חזוקים הינה פרדיגמה נוספת תחת התחום של למידת מכונה, כאשר במקרה זה הלמידה לא מסתמכת על>Data{} {x}, אלא על חקירה של הסביבה ומיציאת המדיניות/הסטרטגיה הטובה ביותר לפועלה. שמו סוכן שנמצא בסביבה שאינה מוכרת, וועלן לבצע צעדים כך שהתגמול המתקבל אותו הוא יקבל יהיה מקסימלי. בלמידה מבוססת חזוקים, ביכולת לפרט הפעולות של למידת מכונה, הסביבה לא ידועה מבעוד מועד. הסוכן נמצא באירועים ודים וains יודע בשום שלב מה הצעד הנכון 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 ליצור אסטרטגיה כך שככל מני מצבים לא ידועים הסוכן יבחר בפעולות שבאupon מctrber'יו היכי'יעילות עבורה. 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 בעדרת התגמול, כאשר נרצה שהסוכן יבצע פעולות המזקחות אותו בתגמול חיובי (-חזוק) וימנע מפעולות עבורה הוא מקבל תגמול שלילי, ובמצטרב הוא ימקסם 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 לנסוח באופן פורמלי את התיאור המתמטי של חלקו' הבעיה השוניים.

11.1 Introduction to RL

בפרק זה נגדיר באופן פורמלי תהליך מרקוב, שעוזר לנו לתאר בעיות של למידה מבוססת חזוקים, ונראה כיצד ניתן למצוא אופטימום לבעיות אלו בהינתן מודל וכל הפרמטרים שלו. לאחר מכן בקשרו במספר שיטות המנסות למצוא אסטרטגיה אופטימלית עבור תהליך 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 עבורה רוצים למצוא אופטימום.

11.1.1 Markov Decision Process (MDP) and RL

המודל המתמטי העיקרי העיקרי עליו מבוסים האלגוריתמים השונים של RL הינו תהליך החלטה מרקובי, כלומר תהליך שבו המעברים בין המצבים מקיימים את תכונת מרקוב, לפיה ההסתגלות של מצב מסוים תלויה רק במצב הקודם לו:

$$P(s_{t+1} = j | s_t = i, s_1, \dots, s_t) = P(s_{t+1} = j | s_t = i)$$

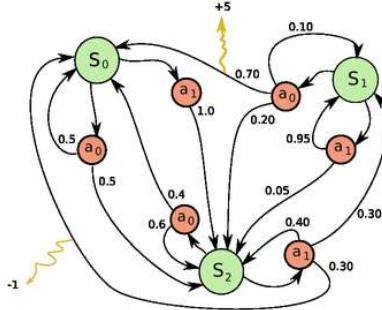
תהליך קבלת החלטות מרקובי מתואר על ידי סט הפרמטרים $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}\}$:

- \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} – הביטוי: $[0, 1] \rightarrow \mathcal{S} | s, a \rightarrow T(s, a) = s'$ הינו פונקציית מעבר, המחשבת את ההסתברות לעבור בזמן t במצב s_t למצב s_{t+1} על ידי הפעולה a : $P(s_{t+1} = s' | s_t = s, a_t = a) = T(s, a) = s'$. ביטוי זה מעשה מייצג את המודול – מה ההסתברות שבחירה הפעולה a במצב s תביא את הסוכן למצב s' .
- \mathcal{R} – הביטוי: $\mathbb{R} \rightarrow (s, a) \rightarrow R(s, a)$ הינו פונקציה הנוגנת לתגמול/רווח לכל פעולה a הגורמת למעבר ממצב s למצב s' , כאשר בדרך כלל $[0, 1] \in \mathcal{R}_a$. לעיתים מסמנים 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 בכך 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 $\{s_0, a_0, r_0, \dots, s_t, a_t, r_t, s_{t+1}\}$, כאשר המצב ההתחלתי מוגדר מהתפלגות כלשהיא $(\cdot | s_0)$, 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 מركזוי. הינה שלושה ממצבים – $\{s_0, s_1, s_2\}$, 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 ממצבים 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 המציג** $\sum_{t=0}^{\infty} \mathbb{E}[R(s_t, a_t) | \pi]$. כיוון שלא תמיד אפשר לחשב באופן ישיר את האסטרטגיה האופטימלית, ניתן להגדיר ערך החזרה (Return) המבטא סכום של תגמולים, ומנסה למקסם את התוחלת שלו $\mathbb{E}[Return | \mathcal{S}, \mathcal{A}]$. ערך ההחזרה הינו נפוץ ונקרא Return (0,1), והוא מוגדר באופן הבא: עבור פרמטר $\gamma \in [0, 1]$, ה- γ -return הסכום הבא:

$$Return = \sum_{t=1}^T \gamma^{t-1} r_t$$

אם $\gamma = 1$, אז מתעניינים רק בתגמול המיידי, וככל ש- γ גדול כך יותר נתונים יותר משמעות 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:

$$\mathcal{V}^\pi(s) = \mathbb{E}[Return | s_0 = s]$$

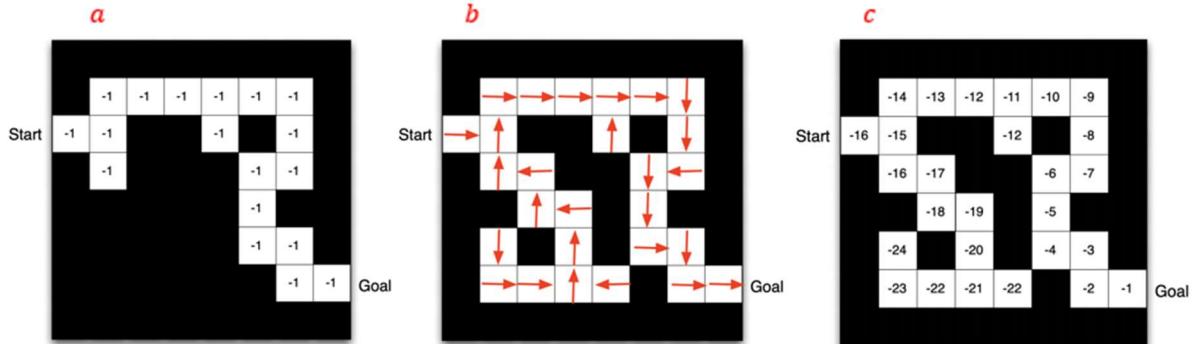
בעזרת ביטוי זה ניתן לחשב את **האסטרטגיה האופטימלית**, כאשר ניתן לנוקוט בגישה ישירה ובגישה עקיפה. הגישה הישירה מנסה למצאו בכל מצב מה פעולה הכי כדאי. בהתאם לכך, חישוב האסטרטגיה האופטימלית יעשה באופן הבא:

$$\pi(s) = \arg \max_a \sum_{s'} p_a(s, s') (\mathcal{R}_a(s, s') + \gamma \mathcal{V}^\pi(s'))$$

לעתים החישוב השיר 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 הכוונים, כל פעולה גוררת תגמול של -1, והסתברויות המעבר קבועות לפני הצבעות של המשבצות (או אפשר למלכט המשבצות השחורות). (b) מדיניות – החלטה בכל מצבizia צעד לבצע. (c) Value של כל משבצת.

לסיכום, ניתן לומר שכל התחומים של RL מבוססי על שלוש אבני יסוד:

- מודל: האופן בו אנו מתארים את מרחב המצבים והפעולות. המודל יכול להיות נתון או שנוצרך לשערק אותו, והוא מורכב מהסתברויות מעבר בין מצבים ותגמול עבור כל צעד:
$$\mathcal{P}_{ss'}^a = p_\pi(s, s') | s_t = s, a_t = a$$

$$\mathcal{R}_{ss'}^a = \mathcal{R}_\pi(s, s') = \mathbb{E}[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s']$$
- 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, Value function 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 שהוא 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 ערך כל מצב את הפעולות האפשריות.

ונכל לסמן ב- $V^*(s)$ 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 .

עתנו נتون מודל מרקובי עם כל הפרמטרים שלו – אוסף המצבים והפעולות, הסתברויות המעבר והתגמול עבור כל פעולה, ומעוניינים למצוא דרך פעולה אופטימלית עבור מודל זה. ניתן לעשות זאת בשתי דרכים עיקריות – מציאת האסטרטגיה (s^*, \mathcal{V}^*) 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^*, \mathcal{V}^*) משתמשת ב-Bellman equation, המבוססת על תכונות דינמי. נפתח את הביטוי של (s^*, \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:

$$= \mathbb{E}_\pi \left[r_{t+1} + \gamma \cdot \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s \right]$$

עתנו נשתמש 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:

$$V = [V_1, V_n]^T, R = [r_1, \dots, r_n]^T$$

$$T = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

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 שمبرטיך שייהיה פתרון למשווהה, ופתרון זה הוא אף ייחיד עבור \mathcal{V}^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 $(3^a)^T$, 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 B , 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 $(\cdot)^\pi$ 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, כיון 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: $\|u - v\|_\infty = \max_{s \in S} |u(s) - v(s)|$. 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 זהה נקודת השבת היחידה ומילא הגענו להתכנסות. בפועל, השתמש 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:

$$V_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, V_1 = \begin{pmatrix} 0.1 \\ 1.3 \\ 3.4 \\ 1.9 \\ 0.4 \end{pmatrix}, V_2 = \begin{pmatrix} 0.6 \\ 2.6 \\ 6.1 \\ 3.7 \\ 1.2 \end{pmatrix}, \dots, V_{10} = \begin{pmatrix} 7.6 \\ 10.8 \\ 18.2 \\ 16.0 \\ 9.8 \end{pmatrix}, \dots, V_{50} = \begin{pmatrix} 14.5 \\ 17.1 \\ 26.4 \\ 26.8 \\ 18.4 \end{pmatrix}, V^\pi = \begin{pmatrix} 14.7 \\ 17.9 \\ 26.6 \\ 27.1 \\ 18.7 \end{pmatrix}$$

ניתן לשים לב שאחרי 50 איטרציות המתקובל בצורה האיטרטיבית קרוב מאוד לפתרון המתקובל בצורה האנליטית.

Policy Iteration (PI)

חישוב π -Value function מאפשר לחשב את ערכו של $(s)^\pi$ עבור כל s , אך הוא אינו מבטיח שנגיעה 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 $(s)^\pi Q$ 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 ולאחר מכן לחזור לפעול בהתאם לאסטרטגיה $(s|a)\pi$ הנגזרת מחישוב -Value function . למעשה, ניתן לחפש את כל הפעולות עבורן כדי לבצע פעולה מסוימת עבורו התגמול יהיה גבוה יותר מאשר האסטרטגיה של $(s|\pi)\mathcal{V}$. באופן פורמלי יותר, נרצה להגדיר אסטרטגיה דטרמיניסטית, עבורה בהסתברות 1 ננקוט בכל מצב s בפעולה הכי כדאי a :

$$\pi'(a|s) = 1 \text{ for } a = \arg \max_{a'} \mathcal{Q}^\pi(s, a')$$

נשים לב שרעיון זה הוא בעצם להשתמש באסטרטגיה גרידית – בכל מצב לננקוט בפעולה הכי משלימה בטוחה של צעד ייחיד, ואז להמשיך עם האסטרטגיה הנתונה. השאלה העולה היא כמובן – מדוע זה בהכרח נכון? כמובן, האם הרעיון שאומר שלא משנה באיזה מצב אנו נמצאים, הבחירה של הפעולה האופטימלית בהכרח טוביל לקבלת אסטרטגיה יותר טובה מאשר האסטרטגיה הנוכחיית? בכך להוכיח זאתணסח זאת ממשפט:

בاهינתן 2 אסטרטגיות π' , כאשר π' דטרמיניסטיבית, אז כאשר $(s|\pi')\mathcal{Q}^\pi(s, a) > \mathcal{V}^\pi(s)$ בהכרח לכל s יתקיים: $(s|\pi')\mathcal{V}^\pi(s) > \mathcal{V}^\pi(s)$. ראשית נפתח לפיה הדרישה:

$$\mathcal{V}^\pi(s) < \mathcal{Q}^\pi(s, \pi'(s)) = \mathbb{E}_\pi[r_{t+1} + \gamma \cdot \mathcal{V}^\pi(s_{t+1}) | s_t = s, a_t = \pi'(s)]$$

כיוון שהאסטרטגיה הינה דטרמיניסטיבית, הפעולה הנבחרת אינה רנדומלית ביחס ל- π' , ולכן נוכל לרשום:

$$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot \mathcal{V}^\pi(s_{t+1}) | s_t = s]$$

כעת לפיה אותו אי שוויון שבנהנחו 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 שוויה 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 $(s|\pi)\mathcal{V}$.

כעת יש בידינו שתי טכניקות שאנו יודעים לבצע:

(E) – בהינתן אסטרטגיה מסוימת נוכל לפתור את משוואות בלמן ולקבל את $(s|\pi)\mathcal{V}$.

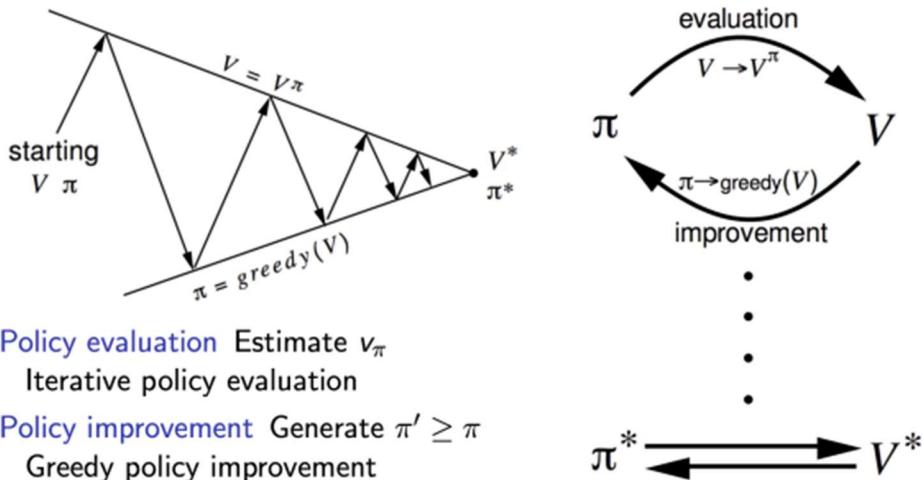
(I) – בהינתן הערך של Value function , נוכל לשפר אותה באמצעות בחירה גרידית של פעולה.

בעזרת טכניקות אלו ניתן להתחיל מאסטרטגיה רנדומלית, ואז לבצע איטרציות המורכבות משתי הטכניקות האלה באופן הבא:

$$\pi_0 \xrightarrow{E} \mathcal{V}^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} \mathcal{V}^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots$$

תהליך זה נקרא Policy iteration – בכל צעד בו יש לנו אסטרטגיה נפתרה עבורו משווהות בלמן ובכך נחשב את Value function שלה, ולאחר מכן נשפר את האסטרטגיה באמצעות $\text{policy improvement}$, שכך מוביל לבצע בחירה גרידית שבתווח הקצר יותר מאשר -Value function שחיישבנו. ניתן להוכיח שאחרי מספר סופי של איטרציות האסטרטגיה תתכנס לנקודת שבת (fixed point).

$$\pi(s) = \arg \max_a \mathcal{Q}^\pi(s, a) = \pi'(s)$$



איור 11.4 – ביצוע איטרציות של Policy improvement ו-Policy evaluation על מנת למצוא בכל שלב את ה-value function ולשפר אותו באמצעות בחירה גreedית.

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 יצאונו אך במקומם התוחלת שהייתה לנו בהתחלה, כתע' יש *max*. נרצה להראות שהפתרון של משוואות אלה הוא *the value function* 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. בשபיל להוכיח 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 בנקר 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 אותה 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 להתכנסות:

$$\mathcal{V}_{k+1} = \max_a \sum_{s'} p_\pi(s, s') (\mathcal{R}_\pi(s, s') + \gamma \cdot \mathcal{V}_k(s'))$$

נשים לב שבשיטת זו אין לנו מידע לגבי האסטרטגיה אלא רק חישובו את ה-value function, אך ממנה ניתן לגוזר את \mathcal{Q} ואז לבחור באסטרטגיה גרידית, שהינה בקרה זה גם אופטימלית:

$$\pi(s) = \arg \max_a \mathcal{Q}^\pi(s, a)$$

ניתן להראות כי בשיטה זו ההתקנסות מהירה יותר ודרשות 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 והסביבה באופן שלם 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 ההחזקה **בהינתן** מודל מركזוי. השתמשנו בשתי הנחות עיקריות על מנת להתמודד עם הבעיה:

1. Tabular MDP – הנחנו שהסביבה סופית ולא גדולה מדי, כך שנוכל ליצג אותה בזכרון ולפתרו אותה.
2. Known environment – הנחנו שהמודול ידוע לנו, כלומר נתונה לנו מטריצת המעברים שקובעת מה הסיכוי לעבור מצב s' במצב s כשלוקטים בעולה a (סימנו את זה בתור $P_{ss'}^a$, ובנוסף נתון לנו מה ה-reward המתקבל עבור כל action a סימנו את זה בתור $R_{ss'}^a$).

בעזרת שתי הנחות פיתחנו את המשוואות בלבד, כאשר הינו לנו שני צמדים של משוואות. משוואות בלבד מעבור אסטרטגיה נתונה כתובות באופן הבא:

$$\mathcal{V}^\pi(s) = \sum_{a, s'} \pi(a|s) P_{ss'}^a (\mathcal{R}_{ss'}^a + \gamma \cdot \mathcal{V}^\pi(s'))$$

$$\mathcal{Q}^\pi(s, a) = \sum_{s'} P_{ss'}^a \left(\mathcal{R}_{ss'}^a + \gamma \sum_{a'} \mathcal{Q}^\pi(s', a') \right)$$

ובנוסף פיתחנו את המשוואות עבור הפתרון האופטימלי:

$$\mathcal{V}^*(s) = \max_a \sum_{s'} P_{ss'}^a (\mathcal{R}_{ss'}^a + \gamma \cdot \mathcal{V}^*(s'))$$

$$\mathcal{Q}^*(s, a) = \sum_{s'} P_{ss'}^a \left(\mathcal{R}_{ss'}^a + \gamma \max_a \mathcal{Q}^*(s', a') \right)$$

הראינו שתי דרכי להגעה לפתרון האופטימלי:

1. Policy improvement – Policy evaluation ו-לאחריו Policy iteration.

2. Value iteration – פתרון משוואות בלבד באמצעות יישיר בעזרת איטרציות על ה-value function.

כאמור, דרכי פתרון אלו מניחים שהמודול ידוע, ובנוסף שמדובר במצבים אינו גדול מדי וכי יכול להיות מוצג בזכרון. האתגר האמייתי מתחילה בנקודה בה לפחות אחת מהנחות אלה אינה תקפה, ולמעשה פה מתחילה התפקיד של RL. עיקר ההתקנות של אלגוריתמים אלו יהיה למצאו באופן יעיל את האסטרטגיה האופטימלית כאשר לא נתונים הפרמטרים של המודול, ואז צריך 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 המודול של המשחק או להשתמש במודול

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 בחורה 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 ג'ו. במקרים אלו המודל הוא המשחק והחוקים שלו, והתוכנה משתמשת בידע זהה בכך כדי לבחון את כל הפעולות והתגובה 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 מצבה כמה אתגרים, כאשר העיקרי ביניהם הוא הסכנה שהסטרטגיה הנלמדת תהיה טוביה רק עבור המקרים אותם ראה הסוקן, אך לא תאים למקרים חדשים שיבואו. אלגוריתמים שמחפשים באופן ישיר את האסטרטגיה האופטימלית אמנים 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 $(a_t | s_t) \pi$ 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:

- A. Policy Optimization – ניסוח האסטרטגיה כבעית אופטימיזציה של מציאת סט הפרמטרים θ הממקסם את $\pi(a | s)$. פתרון בעיה זו יכול להיעשות באופן ישיר על ידי שיטת Gradient Ascent עבור פונקציית המחיר $\mathbb{E}[R(\tau)] = \pi(\theta)$, או בעזרת קירוב פונקציה זו ומיציאת מקסימום עבורה.
- B. Q-learning – שערוך $Q(s, a)$ על ידי $Q_\theta(s, a)$. מיציאת המשערך האופטימלי יכולה להתבצע על ידי חיפוש θ שيسפק את השערוך הטוב ביותר ביחס לשניון למצואו, או על ידי מיציאת 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 – לעומת זאת הוא לרוב אלגוריתם off-policy, כלומר בכל פעולה ניתן להשתמש בכל המידע שנცבר עד כה. היתרון של שיטות האופטימיזציה נובע מכך שהן מנסות למצואו ישיר את האסטרטגיה הטובה ביותר ביחס, בעוד שאלגוריתם Q-learning רק משעריך 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 Zarot לחילופין, וישנם אלגוריתמים שמנוסים לשלב בין הרעיון ו לנצל את החזקוות והיתרונות שיש לכל גישה.

Model-based learning

גם בגישה זו יש שתי קטגוריות מרכזיות של אלגוריתמים:

- A. Model-based RL with a learned model – אלגוריתמים המנסים ללמידה אין את המודל עצמו והן את ה-Value function או את האסטרטגיה π .
- B. 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 הוואדות לא קיים, ולכן ניתן להתמקד בביצועים אסימפטומטיים. במקרים בהם המודל אינו ידוע, הדגש העיקרי הוא על למידת המודל.

11.2 Model Free Prediction

לאחר שסקרנו בפרק המבוא את הבסיס המתמטי של בעיות RL והציגו את משויות בלמן ופתרון, בפרקם הבאים נציג גישות שונות להתמודדות עם בעיות RL עבורי פתרונות אלה אינן מספיקים – או מפניהם גודל יותר מזה שניתן לפתור באמצעות משויות בלמן. בפרק זה נציג שתי שיטות הבאות להתמודד עם מקרים בהם המודל אינו ידוע (לומר האלגוריתם הינו *Model-Free*), והדרך שלהן להתמודד עם אתגר זה היא **לשער את האסטרטגיה האופטימלית** בדרכים אחרות שאינן מצריכות את ידיעת המודל.

11.2.1 Monte-Carlo (MC) Policy Evaluation

האלגוריתם הראשון אותו נציג הינו *Monte Carlo*, והוא מציע דרך לשערת את *Value function* בלי לדעת את המודל. ראשית נסביר בקצרה מהו אלגוריתם *Monte Carlo* ואז נראה כיצד ניתן לישם אותו בעיות RL.

נניח ונרצה לשערת תוחלת של פונקציית התפלגות כלשהיא – $\mathbb{E}_p[f(x)]$. התוחלת יכולה להיות סכום או אינטגרל שקשה מאד לחשב. ניתן לשערת את התוחלת על ידי דוגמאות רנדומליות מההתפלגות וחישוב הממוצע של הדוגמאות:

$$x_1, \dots, x_n \sim p(x)$$

$$\mathbb{E}_p[f(x)] \approx \frac{1}{n} \sum_i f(x_i)$$

לפי חוק המספרים הגדולים הממוצע הדוגמאות מתכנס לתוחלת. משערך זה הינו חסר הטיה, ובנוסף השונות שלה קטנה ביחס לנארית למוגנות הדוגמאות:

$$\begin{aligned} \mathbb{E}\left[\frac{1}{n} \sum_i f(x_i)\right] &= \mathbb{E}[f(x)] \\ Var\left[\frac{1}{n} \sum_i f(x_i)\right] &= \frac{Var[f(x)]}{n} \end{aligned}$$

כאמור לעיל, ה-*Value function* הינה תוחלת עבור אסטרטגיה נתונה π , כאשר מתחילה במצב s :

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right]$$

אםنم התוחלת היא על סכום אינסופי, אך עבור מקרים רבים נוכל להניח שהሪיצה היא סופית (Episodic MDP). בפועל נבצע את הפעולה הבאה: עבור אסטרטגיה נתונה π , ניציר מנתה ריצה של T צעדים, ואז ניקח ממוצע ונסתכל על כל ה-rewards שמתקבים בריצחה זו החל ממנה. באופן זהה קיבלנו value עבור הערך של אותו מצב. הריצה על אותה פעולה שוב ושוב תיציר ריצות שונות ומילא ערכים שונים למצב מסוים, ומיצוע על פני הערכים ייתן לנו שערך לערך האמתי של אותו מצב. נעיר בהערכת אגב שכיוון שמצבים יכולים לחזור על עצם, נוצרת בעיה שבריצעה כזו הממצבים אינם בלתי תלויים. בכך לhattגבר על כך, אם מצב חוזר על עצמו יותר מפעם אחת אפשרים לדוגם רק את המופיע לראשונה של אותו מצב ולא את יתר המופיעים (ישן עוד דרכים להתגבר על כך, אך זהה הדרך פשוטה ביותר). באופן פורמלי, נניח שיש לנו ריצה של T צעדים:

$$S_1, A_1, R_1, \dots, S_{T-1}, A_{T-1}, R_{T-1}, S_T$$

از דוגמה אחת מתוך התפלגות ($\sum_{k=0}^{\infty} r_{t+k+1} | s_t = S_t$) תראה באופן הבא:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{T-1} R_T = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

למשל, הדוגמה G_1 תהיה מורכבת מכל ה-rewards שהגיעו לאחר הצעד הראשון: $R_1 + \gamma R_2 + \gamma^2 R_3 + \dots + \gamma^{T-1} R_T$. הסכום הזה ייתן לנו value עבור אותו מצב ממנו התחלנו (S_1), ועל ידי שערך אותו מצב שוב ושוב ביחס לריצות שונות נוכל לקבל ערכים שונים, ולאחר מכן למציע אותם בכספי לשערך את *Value function* האמתי של אותו מצב S_1 .

באופן פורמלי, העדכן של מצב לאחר כל דוגמה נראה כך:

$$\#sample\ of\ s_t: N(S_t) = N(S_t) + 1$$

$$update\ the\ value\ of\ s_t: \mathcal{V}(s_t) = \mathcal{V}(s_t) + \frac{1}{N(S_t)}(G_t - \mathcal{V}(s_t))$$

ולאחר הרבה דגימות השערוך מתקבל על ידי התוחלת שלהן:

$$\mathcal{V}(s) = \mathbb{E}_{\pi}[G_t | s_t = s]$$

באופן סכמתי ניתן לתאר את האלגוריתם באופן הבא:

First-visit MC prediction, for estimating $V \approx v_{\pi}$

Initialize:

$\pi \leftarrow$ policy to be evaluated
 $V \leftarrow$ an arbitrary state-value function
 $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

Generate an episode using π
For each state s appearing in the episode:
 $G \leftarrow$ the return that follows the first occurrence of s
Append G to $Returns(s)$
 $V(s) \leftarrow$ average($Returns(s)$)

איור 11.5 אלגוריתם MC עבור שערוך ה-*mciton* Value function בהינתן Policy. עבור כל מצב מתחילה רשימה ריקה, ולאחר מכן מייצרים המון ריצות שונות. עבור כל ריצה, עוברים על כל הממצבים ובודקים מה ה-*G*-שליהם, ומוסיפים אותו לרשימה של אותו מצב. לבסוף מחשבים את ה-value של כל מצב על ידי מיצוע הרשימה (ערכי *G* השונים) של אותו מצב.

יש מספר יתרונות לשיטה זו: היא מספקת משערוך חסר הטיה עבור ה-*mciton*, מבטיחה התכנסות אחריו מספיק איטרציות, ובנוסף ניתן לשערך באמצעותה את השגיאה. אפשר גם באוטה דרך גמ את $(a, s) \in \mathcal{Q}^{\pi}$ אך זה יהיה יותר רועש ודרושים יותר דגימות.מן הצד השני יש גם חסרון לשיטה זו: ראשית, היא מתאימה רק ל-*MDP Episodic* ולא לריצות אינסופיות. עם בעיה זו ניתן להתמודד בקהלות כיוון שעבור בעיה אינסופית ניתן לקחת ריצה סופית ולהחסם את השגיאה באמצעותה. שנית, ההתכנסות יכולה להיות מאוד איטית, ובנוסף השונות יחסית גבוהה.

11.2.2 Temporal Difference (TD) – Bootstrapping

במקום להסתכל על ריצה שלמה, ניתן אחריו כל צעד לעדכן את ה-value. ממשוואת בלמן ניתן להראות שהביטוי $\gamma \mathcal{V}^{\pi}(S_{t+1}) + R_{t+1}$ הינו משערוך חסר הטיה עבור $(S_t)^{\pi} \mathcal{V}$. א' אפשר להשתמש במשערוך זה כמו שהוא, כיון שאנו חסם לא יודעים את ה-*mciton*, ומילא הביטוי $(S_{t+1})^{\pi} \mathcal{V}$ לא ידוע. בשביל בכל זאת לשערוך את $(S)^{\pi} \mathcal{V}$ באמצעותו אותו משערוך, ניתן להחליף את $(S_{t+1})^{\pi} \mathcal{V}$ ב- $(S_{t+1}) \mathcal{V}$, ולבצע את השערוך באופן הבא:

$$\mathcal{V}^{\pi}(S_t) = R_{t+1} + \gamma \mathcal{V}(S_{t+1})$$

הרעין מאחוריו השימוש הזה הוא להיעזר במידע שיש לנו מ- R_t . נניח ונניח שערך כלשהו עבור $(S_t)^{\pi} \mathcal{V}$ וניחס זה יהיה גרווע. אפשר מעט לשפר את הניחוש באמצעות ניחוש $(S_{t+1}) \mathcal{V}$ ושימוש ב-reward R_{t+1} שהתתקבל עבור אותו מצב S_t , שבעצם מספק מידע כלשהו על מצב זה. שערוך זה עדיף על ניחוש מוחלט כיוון שהאלמנט של הניחוש מקבל משקל נמוך יותר עקב המכפלה ב- γ , ויש יותר משקל ל- R_{t+1} מאשר ל- $(S_t)^{\pi} \mathcal{V}$. צירkus לב שאנו מנסים לשערוך את ה-*mciton* מתוך הערכאים שלה עצמאה. מתחילה את כל הערכאים במספרים כלשהם (למשל – וקטור של 0), ואז עוברים צעד צעד וمعدכנים את הניחושים בעזרת התוגמלים. אינטואטיבית זה נראה מעט שונה, אך מסתבר שפתרון זה הוא אחד הכלים החזקים בבעיות RL. באופן פורמלי האלגוריתם נראה כך:

Tabular TD(0) for estimating v_π

```

Input: the policy  $\pi$  to be evaluated
Initialize  $V(s)$  arbitrarily (e.g.,  $V(s) = 0$ , for all  $s \in \mathcal{S}^+$ )
Repeat (for each episode):
    Initialize  $S$ 
    Repeat (for each step of episode):
         $A \leftarrow$  action given by  $\pi$  for  $S$ 
        Take action  $A$ , observe  $R, S'$ 
         $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$ 
         $S \leftarrow S'$ 
    until  $S$  is terminal

```

איור 11.6 אלגוריתם (TD) Temporal Difference עברו שערוך ה-Value function בהינתן Policy. מוחשים ערך עבר כל מצב ואז באופן איטרטיבי משפרים את הניחושים בעזרת שערוך התלוי-ב-reward ובערך המצב הבא.

מגדירים את השגיאה של ה-TD כהפרש שבין הביצוע עבר ערך המצב בין השערור שלו:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

בשונה משערור MC, השערור בשיטת TD הוא בעל הטיה, אך השונות קטנה יותר. בנוסף, כיוון שבכל צעד מבצעים שיפור לערך של מצב, תהליך השערור יותר מהיר מאשר ב-MC. הרבה פעמים מוסיפים פרמטר α לשערור (כפי ש谟פייע באירור 11.6):

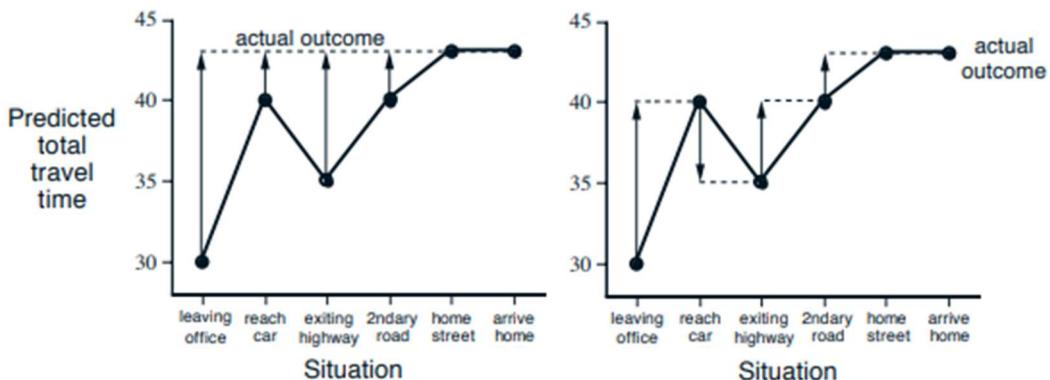
$$V(S_t) = V(S_t) + \alpha \cdot [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

עבור ערכי α מתאימים, המשערק מתכנס לתוצאות האמיתית.

ניתן דוגמה שתמחיש את שיטת MC ושיטת TD ואת היחס ביניהם: הנה יצא מהמשרד שלו ונסע לבתו, ובדרך הוא ניסה לשערק את זמן ההגעה שלו וקיבל את הזמנים הבאים:

מצב	כמה זמן עבר	שערן הזמן הנוכחי הכללי	שערן הזמן הנוכחי הכלול
יציאה מהמשרד	0	30	30
הליכה למכוונית תחת גשם	5	35	40
הגעה לכביש מהיר	20	15	35
נסעה מאחוריו משאית	30	10	40
הגעה לרחוב של הבית	40	3	43
הגעה הביתה	43	0	43

נשרטט את שני המשערכים:



איור 11.7 שעורך MC (משמאל) וشعורך TD (מימין) ביחס לתצפית הנתונה.

כיוון שהשיטה MC מספקת ערך לאחר ריצה שלמה, אז ניקח את זמן ההגעה בפועל של הנהג וניתן את הערך זהה לכל מצב היבניים. שיטת TD לעומת זאת מעדכנת את הערך בכל מצב בהתאם למצב הבא. ניתן לראות שהשינוי בשערוך TD קטן מזו שהתקבלה בשערוך MC.

ניתן להסתכל על שיטת TD כבעית רגסיה "динמית":

עבור כל צעד נרצה ש- $(S_t) \mathcal{V}$ יהיה שווה למשוואות בלמן, כלומר נרצה לשערק את $(S_t) \mathcal{V}$ כך שיתקיים:

$$\mathcal{V}(S_t) = \mathbb{E}_\pi[R_{t+1} + \gamma \mathcal{V}(S_{t+1}) | s_t = S_t]$$

עבור בעיה זו נוכל להגדיר פונקציית מחיר (Loss) מותאמת:

$$L = \frac{1}{2} (\mathbb{E}_\pi[R_{t+1} + \gamma \mathcal{V}(S_{t+1}) | s_t = S_t] - \mathcal{V}(S_t))^2$$

את הפונקציה הזו ניתן למצער על ידי דגימות סטטיסטיות של $(S_{t+1}) \mathcal{V} R_{t+1} + \gamma \mathcal{V}(S_{t+1})$. נשים לב לדבר חשוב – המטרה שלנו היא לשערק את ההוּא באמצעות העתיד ולא להיפך, כיוון שהעתיד הוא בעל יותר מידע – הוא ראה ומצב חדש. הבדיקה זו משפיעה על איך שאנו רוצים שפונקציית המחיר תתנהג – אנחנו רוצים ש- $(S_t) \mathcal{V}$ יתקרב לערך של $s_t = S_t | s_t = S_t = \mathbb{E}_\pi[R_{t+1} + \gamma \mathcal{V}(S_{t+1})]$ ולא להיפך. בדוגמה של הנגשה שימושית את זמן הנסעה – הוא רוצה לעדכן את השערוק של כל מצב בהתאם למצב הבא. אם למשל הרערכה שלו בזמן t הינה 30 דקות ואז הוא מגיע לפיקק וمعدכן את הערכה ל-35 דקות, אז הוא ירצה לתקן את השערוק הקודם ($(S_t) \mathcal{V}$) כך שהיא דומה לשערוק הנוכחי ($(S_{t+1}) \mathcal{V} + \gamma \mathcal{V}(S_{t+1})$), ולא לעדכן את השערוק הנוכחי כך שהיא דומה לקודם. בכך לדאוג לכך, נתיחס לעתיד כקבוע ולא נחשב עבורי גודיאנט (למאות ש- \mathcal{V} מופיע בו). באופן פורמלי נוכל לנסח זאת כך:

$$T(S_t) = \mathbb{E}_\pi[R_{t+1} + \gamma \mathcal{V}(S_{t+1}) | s_t = S_t]$$

$$L = \frac{1}{2} (T(S_t) - \mathcal{V}(S_t))^2$$

ואז הגדרינאנט יהיה:

$$\frac{\partial L}{\partial \mathcal{V}} = T(S_t) - \mathcal{V}(S_t)$$

בהתאם לכך, הדגימה $(S_{t+1}) \mathcal{V} R_{t+1} + \gamma \mathcal{V}(S_{t+1})$ משתנה בכל צעד (אמנם קיבענו אותה בכל צעד היחיד, אך היא עדין תלוי ב- $(S_t) \mathcal{V}$). במובן זהה בעית הרגסיה שהגדרנו הינה "динמית", כיוון שהיא משתנה בכל צעד.

11.3.2 TD(λ)

נסזה לבדוק את הקשר בין שתי השיטות שראינו. שערוך TD מבצע דגימה של ריצה ואז מעדכן את הערך של כל מצב בהתאם למצב הבא בלבד:

$$\mathcal{V}(S_t) = \mathcal{V}(S_t) + \alpha \cdot [R_{t+1} + \gamma \mathcal{V}(S_{t+1}) - \mathcal{V}(S_t)]$$

בגלל ההתייחסות למצב אחד בכל פעם השונות של המשערק נמוכה, אך יש הטיה.

שיטת MC לעומת זאת דוגמת ריצה וمعدכנת את הערך של כל מצב בהתאם לכל המצביעים שבאים לאחר מכן:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

$$\mathcal{V}(s_t) = \mathcal{V}(s_t) + \frac{1}{N(S_t)} (G_t - \mathcal{V}(s_t))$$

משערק זה הינו חסר הטיה, אך עם זאת השונות שלו גבוהה.

נראה כיצד ניתן לחבר בין שני המשערכים ולמצוא שיטה שתיה אופטימלית מבחינת היחס שבין ההטיה לשונות. ניתן להציג את שני המשערכים לנוסחה כללית באופן הבא:

$$\mathcal{V}(s_t) = \mathcal{V}(s_t) + \alpha \left(G_t^{(n)} - \mathcal{V}(s_t) \right)$$

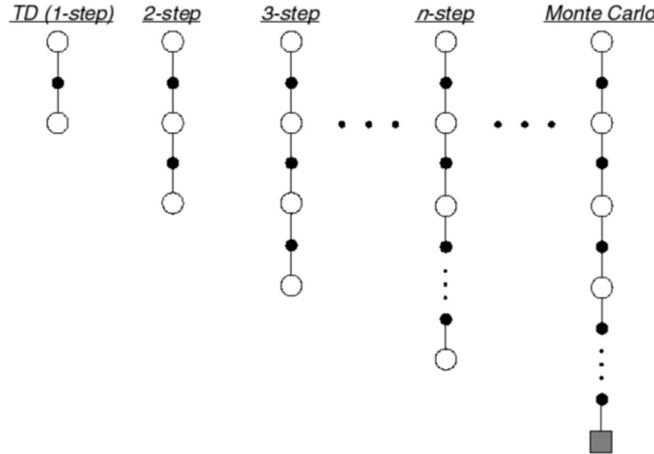
כאשר $G_t^{(n)}$ הוא הסכום של n ה-rewards הבאים:

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n \mathcal{V}(s_{t+n}) = \sum_{k=0}^{n-t-1} \gamma^k R_{t+k+1}$$

כעת נוכל להגיד:

$$TD(n) \rightarrow \mathcal{V}(s_t) = \mathcal{V}(s_t) + \alpha \left(G_t^{(n)} - \mathcal{V}(s_t) \right)$$

ולפי סימון זה נוכל לשים לב שערך MC הוא למשעה $(\infty \rightarrow n)$ וairo משערך TD שווה ל- $(1 - \gamma)$.



איור 11.8 משערך MC ($n = 1$). משערך MC הינו המקירה הפרטיאן ($TD(n = 1)$).

אם $\gamma < 1$ נקבל משערך "ממוצע" בין MC לבין TD. ניתן להציג גרסה יותר טובת למשערך זה תחת ההנחה שככל שמאורעות סמוכים אחד לשני קריך יש להם יותר השפעה. בדומה ל- λ -.discount factor שMOVED את השפעה של תגמול כל שהוא יותר רחוק מהמצב הנוכחי, כך גם ניתן לכל מאורע עתידי משקל הולך וקטן. נdag שכל המשקלים יסתכם ל-1 ונקבל את השערוך הבא, הידוע גם בכינוי $TD(\lambda)$:

$$G_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

$$\mathcal{V}(s_t) = \mathcal{V}(s_t) + \alpha \left(G_t^\lambda - \mathcal{V}(s_t) \right)$$

שערוך זה הוא בעל שונות גודלה יותר מאשר TD, כיוון שהוא מתחשב ביותר צעדים, אך ההטיה שלו קטנה יותר. כאמור, הוא מנסה למצער בינהו שני המשערכים שראינו ולאזן בין השונות להטיה.

נסכם בקצרה את מה שראינו בפרק זה. התיכון למקרים בהם המודל אינו ידוע ואנו רוצים לשערך אותו, והציגנו שלוש גישות המנויות לשערך את המודל באמצעות דוגמאות סטטיסטיות: MC, TD (שהזהו מקירה פרטיאן (n)) וגישה המנסה לשלב בין השתיים – $TD(\lambda)$.

לסיטים נעיר ששערוך האסטרטגיה כשלעצמה אינו מספיק, כיוון שהשערוך אמן מספק אסטרטגיה מסוימת עבור הבעה, אך היא אינה בהכרח אופטימלית. בפרק הקודם ראינו כיצד בהינתן המודל ניתן לשפר אותו ולמצוא את האסטרטגיה האופטימלית. יכולנו לעשות זאת כיון שידענו את המודל במלואה, מה שאיפשר לבצע בכל צעד מהלך חמדני ובכך להתכנס לבסוף לאסטרטגיה האופטימלית. במקרים בהם אנו רק משערכים את האסטרטגיה ללא ידיעת המודל, אין לנו בהכרח מידע מספיק טוב עבור כל המצבים. מצבים ופעולות שלא נועדו כלל (או נועדו רק בפעם נדירה), השערוך עברום יכול להיות די גרוע, ואז השיפור באמצעות בחירה גרידית לא בהכרח יכול להביא את האסטרטגיה להיות אופטימלית.

11.3 Model Free Control

בפרק הקודם ראינו כיצד ניתן לשערך את המודל באמצעות דוגמאות סטטיסטיות. שיטות השערוך אפשרו לנו לקבל מידע על המודל, אך הן אינן התיכון לשאלת האם הוא אופטימלי. בפרק הראשון ראינו כיצד ניתן לקחת מודל או

אסטרטגיה ולהביא אותם לאופטימליות, אך אי אפשר להשתמש בשיטה זו עבור מודל משוער. הסיבה לכך נעוצה שמודל זה לא בהכרח ראה את כל הזוגות האפשרים של המוצבים והפעולות, וממילא הוא לא יכול לשפר את הבחירה שלו על סמך אסטרטגיה גרידית. ניקח לדוגמה מקרה בו עבור מצב מסוים האסטרטגיה הינה דטרמיניסטית והפעולה שנבחרת הינה תמיד ללקת מעלה. במקרה זה אין לנו שום מידע על יתר הפעולות האפשרות במצב זה וכן המודל שלנו לא שלם, ומילא לא יוכל להשתמש ב-*Policy improvement* המבוסס על כך שיש לנו מידע על כל המוצבים והפעולות.

ב כדי להתמודד עם בעיה זו נהיה חייבים לדאוג לכך שנבחר בכל המוצבים. כਮון שנוכל לנתקט בגישה פשטנית של אסטרטגיה אקראית, שאחרי מספיק גדול של פעולות קרוב לוודאי שנבחר בכל המוצבים. אסטרטגיה זו אמונה טוביה לבדיקת מצבים ופעולות חדשים, אך כמובן שהיא מלהיות אופטימלית, שכן נרצה להשתמש באסטרטגיה שמצד אחד מנסה להיות אופטימלית ומצד שני שבה ממד של אקרניות המביא לכך שנבחר גם במצבים שלא הינו מගיעים אליהם לפי האסטרטגיה הנוכחית. בחירת פעולה בהתאם לастטטגיה נקראת **Exploration** ואילו בחירה של מצבים חדשים נקראת **Exploration**, והמטרה שלנו תהיה לאזן בין השניים תחת הדרישות הבאות:

1. ביקור בכל הזוגות של המוצבים והפעולות אינסוף פעמים.
2. ככל שמספר הצעדים גדל, כך האסטרטגיה מתכנסת לאסטרטגיה הגרידית:

$$\lim_{k \rightarrow \infty} \pi_k(a|s) = 1 \left[a = \arg \max_{a'} Q(s, a') \right]$$

מצד אחד נרצה לבקר בכל המוצבים ומצד שני נרצה שכך שנעשה יותר צעדים ככה נשפר את האסטרטגיה שלנו. אסטרטגיה המקייםת דרישות אלה נקראת **(GLIE)**, Greedy in the Limit of Infinite Exploration (GLIE). ונינתן להוכחה שקיים דרישות אלה מביא את האסטרטגיה להיות אופטימלית. דוגמה לאסטרטגיה פשוטה העונה על הדרישות הינה $\epsilon - greedy$ – בחירה של האסטרטגיה האופטימלית בהסתברות ϵ – $(1 - \epsilon)$ ובבחירה מצב אחר בהסתברות ϵ . עבור ϵ שהוא קטן עד ל-0 בקצב שאין מהיר מדי, אסטרטגיה זו הינה GLIE.

עליל הרינו כיצד בהינתן מודל ניתן לשפר את האסטרטגיה (Policy improvement) על ידי כך שבחורנו באופן גריד'י פעולות. כתעת נרצה להראות שבאופן דומה מתקיים אותו רעיון גם עבור $\epsilon - greedy$ ו. ר. t. Q^{π_i} , כלומר שאם השתרמשנו בשיטה זו והגענו ל-*Value function*, אז ניתן בכך הבא לשפר את ה-*Value function* על ידי אסטרטגיה ϵ -גרידית. אסטרטגיה זו בוחרת באופן גריד'י את הפעולה האופטימלית לפי האסטרטגיה הנתונה, אך nonetheless לכל יתר הפעולות הסתברות ϵ הגדולה או שווה להסתברות שהיא תהיה לפעולה זו בשימוש באסטרטגיה ϵ -גרידית. ננסח את המשפט באופן פורמלי:

$$\text{If Policy } \pi_i \text{ has } \forall s, a: \pi_i(a|s) \geq \frac{\epsilon}{|A|} \text{ and } \pi_{i+1} \text{ is } \epsilon - \text{greedy w.r.t } Q^{\pi_i}, \text{ then } V^{\pi_{i+1}} \geq V^{\pi_i}$$

הוכחה: נסתכל על המקרה בו נוקטים צעד אחד גריד'י לפי π_{i+1} ואז ממשיכים לפי האסטרטגיה הקודמת π_i :

$$\sum_a \pi_{i+1}(a|s) Q^{\pi_i}(s, a) = \sum_a \frac{\epsilon}{|A|} Q^{\pi_i}(s, a) + (1 - \epsilon) \max_a Q^{\pi_i}(s, a)$$

נרשום את $(\epsilon - 1)$ בצורה אחרת: במקומות ϵ נרשום $\sum_a \frac{\epsilon}{|A|}$, ובמקומות 1 נרשום $\sum_a (1 - \epsilon)$ (הסכום אכן שווה ל-1 כי סוכנים את כל האפשרויות עבור התפלגות נתונה). נקבל:

$$= \sum_a \frac{\epsilon}{|A|} Q^{\pi_i}(s, a) + \max_a Q^{\pi_i}(s, a) \sum_a \left(\pi_i(a|s) + \frac{\epsilon}{|A|} \right)$$

cutet נחליף את הביטוי $Q^{\pi_i}(s, a) \leq \max_a Q^{\pi_i}(s, a)$. כיוון שמתקיים $Q^{\pi_i}(s, a) \leq \max_a Q^{\pi_i}(s, a)$. נקבל:

$$\geq \sum_a \frac{\epsilon}{|A|} Q^{\pi_i}(s, a) + Q^{\pi_i}(s, a) \sum_a \left(\pi_i(a|s) - \frac{\epsilon}{|A|} \right)$$

cutet יש שני איברים זהים שמקטמו, ונשאר עם:

$$= \sum_a \pi_i(a|s) Q^{\pi_i}(s, a)$$

ובסך הכל קיבלנו:

$$\sum_a \pi_{i+1}(a|s) Q^{\pi_i}(s, a) \geq \sum_a \pi_i(a|s) Q^{\pi_i}(s, a)$$

הביטוי שהתקבל מסמל את value function π כאשר עוקבים אחר האסטרטגיה π . יצא מכך שאם עושים צעד אחד לפי האסטרטגיה π ואז ממשיכים לפי π , זה בהכרח יותר טוב מאשר גם את הצעד הראשון לעשות לפי π .Cut בדומה להוכחה שהראינו לעיל, ניתן להוכיח שמתוקים $(s) \leq V^{\pi_{i+1}}(s) \leq V^{\pi_i}(s)$ וביצוע השיפור שוב ושוב יביא את האסטרטגיה להתכנס לזו האופטימלית. הבעה בשיטה זו היא חוסר הייעולות שבה, כיון שהיא דורשת המון דגימות והמון איטרציות. עקב כך שיטה זו לא פרקטית, ובמקרה נציג Cut שיטות אחרות המאפשרות לשפר את האסטרטגיה עבור מודל משוער. פורמלית, שיטות אלה נקבעו תחת קטגוריה הנקראת Model Free Control – שליטה (ושיפור) אסטרטגיה שאינה מتبוססת על מודל ידוע מראש.

11.3.1 SARSA – On-Policy TD control

בפרק הקודם ראיינו כיצד ניתן באמצעות שערור TD לשפר את value function, כאשר העדכון בכל צעד מתקיים באופן הבא:

$$V(S_t) = V(S_t) + \alpha \cdot [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Cut אנחנו מתעניינים באסטרטגיה ולא רק ב-value function, לכן במקום לעדכן את $V(S_t)$ נעדכן את פונקציית ה-state-action $Q(S_t, A_t)$. באופן זה נקבל טבלה בגודל $|S| \times |A| \times |Q|$: המכילה מידע על כל הזוגות האפשריים של (S, A) , ועבור כל זוג יש ערך מסוים (טבלה זו נקראת table – Q). בהתחלה הערכים בטבלה לא ישקוו את הערכים האמיתיים, אך עם התקדמות הלמידה הطالה תשפר ועלוי אף תכנס לאופטימליות. העדכון מתבצע באופן הבא:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \cdot [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

העדכון בכל צעד מtbצע על סמך המצביעים והפעולות של שתי יחידות זמן: $\{S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}\}$ ולכן האלגוריתם נקרא SARSA. אלגוריתם זה הינו On-Policy Learning, כלומר, העדכון בכל צעד גואה על סמך מידע מהגע מהאסטרטגיה הידועה באותו זמן: בוחרים לבצע פעולה A_t במצב S_t (בהתאם לאסטרטגיה), ואז מערכנים אותה על סמך התגמול R_{t+1} שהתקבל בעקבות הפעולה A_t . באופן סכמטי איטרציה אחת של האלגוריתם מתוארת באופן הבא:

```

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$ 
Repeat (for each episode):
    Initialize  $S$ 
    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Repeat (for each step of episode):
        Take action  $A$ , observe  $R, S'$ 
        Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
         $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$ 
         $S \leftarrow S'; A \leftarrow A'$ ;
    until  $S$  is terminal

```

איור 11.7 אלגוריתם SARSA. שערור policy-on-the-fly של $(s, a) \rightarrow Q(s, a)$ בעזרת דגימות מאסטרטגיה ϵ -גרידית ביחס ל- Q הנוכחי.

בכל מצב הפעולה שנבחרת הינה ϵ -גרידית ביחס ל- $Q(s, a)$ הנקחי. כלומר: אם במצב S_t יש לנקיוט לפי האסטרטגיה את המצב $\bar{A}_t = \bar{A}$, אז האסטרטגיה ה- ϵ -גרידית ביחס לכך הינה:

$$A_t = \begin{cases} \bar{A} & w.p. 1 - \epsilon \\ A \neq \bar{A} & w.p. \epsilon \end{cases}$$

ניתן להוכיח שאלגוריתם SARSA מביא את האסטרטגיה לאופטימליות תחת שני תנאים:
א. שהאסטרטגיה תהיה GLIE.

ב. שיתקיים תנאי Robbins-Monroe עבור α (תנאי זה דואג לכך שנגיעה בהכרח לכל המצביעים ומצד שני $0 \rightarrow \alpha_i$):

$$\sum_{i=0}^{\infty} \alpha_i = \infty, \sum_{i=0}^{\infty} \alpha_i^2 < \infty$$

לגישה זו, הפעלת בגישה *on-policy learning* יש מספר חסרונות:

1. המטרה היא ללמד את האסטרטגיה האופטימלית אבל בפועל *the-off*-exploration הוא ביחס לאסטרטגיה הנוכחית בכל מצב.
2. לא ניתן להשתמש בצדדים יישנים, כיון שהם מתייחסים לאסטרטגיה שכבר לא רלוונטית.
3. לא ניתן להשתמש במידע שmagיע מבחוץ.

11.3.2 *Q*-Learning

ניתן להפוך את אלגוריתם SARSA להיות *policy-off*, והאלגוריתם המתkeletal, שנקרא *Q* – Learning, הוא אחד האלגוריתמים השימושיים בתחום של RL. נתובן בפונקציית העדכון של SARSA:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \cdot [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

האלמנט המרכזי באסטרטגיה הינו (S_{t+1}, A_{t+1}, Q) , כיון שבו אנחנו נוקטים בפועל A_{t+1} בהתאם לאסטרטגיה. במקרה לבחור בפועל זה, ניתן להיות גריד'י ולקחת את הפעולה בעלת הערך הכי גדול בצד הקרוב, ועל פיה לעדכן את ה-*Q-value*:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \cdot \left[R_{t+1} + \gamma \max_A Q(S_{t+1}, A) - Q(S_t, A_t) \right]$$

cutת האסטרטגיה אינה משפיעה על פונקציית העדכון, וממיילא היא *off-policy*.

```

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$ 
Repeat (for each episode):
    Initialize  $S$ 
    Repeat (for each step of episode):
        Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
        Take action  $A$ , observe  $R, S'$ 
         $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$ 
         $S \leftarrow S'$ ;
    until  $S$  is terminal

```

איור 11.8 אלגוריתם *Q* – Learning. שעריך $Q(s, a)$ של Q בעזרת דגימות מאסטרטגיה ϵ -גרידית ועדכן ה-*value* – Q בהתאם לפועל בעלת הערך הכי גדול בצד הקרוב.

האסטרטגיה לא משפיעה על עדכון ה-*value* – Q , אך כן יש לה השפעה על **כמויות** הפעמים שנבקר בכל מצב. בנויגוד לאלגוריתם SARSA בו דרשו שהאסטרטגיה תהיה GLIE, GLIE – Learning מדרישה הינה רק שנבקר אינסוף פעמים בכל מצב. הבדל זה הוא משמעותי, כיון ש-GLIE מושך שהאסטרטגיה תתכנס לאסטרטגיה הגרידית (כלומר, הדרישה היא ϵ -ילך). דרישת זו מקשה על הלמידה כיון שצדדים גריד'ים מבאים מעט מידע חדש. הסרת הדרישת על ה- ϵ -אפשרות exploration בצדורה יותר חופשית, והלמידה נעשית בצדורה הרבה יותר מהירה. עם זאת, יותר מדי exploration זה גם לא טוב, כיון שນבקרים בהרבה מצבים לא רלוונטיים מספר רב של פעמים.

נתובן על האלגוריתמים שראינו עד כה ונשווה בין הפתרונותות שהיו מבוססים על ידיעת המודל לבין פתרונות משוערכים:

Full Backup (Dynamic Programming)	Sample Backup (TD)
Iterative Policy Evaluation: $\mathcal{V}(S) = \mathbb{E}[\mathcal{R} + \gamma \cdot \mathcal{V}(S') S]$	TD Learning: $\mathcal{V}(S) = \mathcal{V}(S) + \alpha \cdot [R_{t+1} + \gamma \mathcal{V}(S') - \mathcal{V}(S)]$
Q-Policy Iteration: $Q(S, A) = \mathbb{E}[\mathcal{R} + \gamma \cdot Q(S', A') S, A]$	SARSA: $Q(S, A) = Q(S, A) + \alpha \cdot [R + \gamma Q(S', A') - Q(S, A)]$
Q-Value Iteration:	Q-Learning:

$\mathcal{Q}(S, A) = \mathbb{E} \left[\mathcal{R} + \gamma \cdot \max_A \mathcal{Q}(S, A) S, A \right]$	$\mathcal{Q}(S, A) = \mathcal{Q}(S, A) + \alpha \cdot \left[\mathcal{R} + \gamma \max_A \mathcal{Q}(S', A) - \mathcal{Q}(S, A) \right]$
--	--

פתרונות המשוערים מצלחים להתמודד עם בעיותBINONIOT, אך הצורך בדגימות יכול להיות בעית משתי סיבות:
א. האלגוריתמים נדרשים להמן דגימות בשביל להצלחה. ב. חסרן נוסף בגין שאלות שאלן model-based קשור ל-exploration עבור בעיות של העולם האמיתי, לא תמיד אפשר לראות דוגמאות שליליות כדי ללמידה לא טובות. רכב אוטומטי למשל, אם רוצה שהוא יוכל לא לנסוע ברמזור אדום, אי אפשר לאמן אותו על ידי זה שהיתן לו יד חופשית לבצע exploration וכך הוא יכול למציבים בהם הוא יישב באור אדום ויקבל על כך תגמול שלילי. לעומת זאת, בעיות הבעיות בסימולציה הן יותר מתאימות לאלגוריתמים שהציגו המבוססים על דגימות ושערור, כיוון שניתן בזורה אפשרות זולה לבצע המן דגימות, ובנוסף אין בהן בעיות בטיחות בביוזה exploration-h-exploration.

מלבד בעיות הדגימות והשערור, האתגר העיקרי של גישות אלה נעוץ ביכולת ההכללה של המודלים הנלמדים. התוצאה של SARSA-and-Learning – \mathcal{Q} הינה כאמור טבלה של value – \mathcal{Q} , ובטבלה זו אין שום קשר בין הערכים השונים בטבלה. כל איבר בטבלה עומד בפני עצמו, ואי אפשר ללמידה ממנו על שאר האיברים. אם למשל הענו לUMBRELLA החדש שעוד לא ראיינו אך הרבה ממציבים דומים לו, לא יוכל ללמידה מהם שום דבר לגבי המצב החדש. אתגר זה משליך גם על גודל הבעיות אותן ניתן לפתור – אם אי אפשר להכליל ממציב אחד לממציב אחר, מミלא זה מגביל מאוד את גודל הבעיה שהיא ניתן להתמודד בעזרת אלגוריתמים אלו. במקרים בהם מרחיב הממציבים הוא רציף, אך מרחב הממציבים הוא אינסופי ועוד בכלל לא ניתן להשתמש בಗישות אלו.

11.3.3 Function Approximation

כמו, אלגוריתמים שמנסים לשערר את \mathcal{Q} אינם ישים בעיות גדולות בעיקר בגלל חוסר יכולת שלהם להכליל ממציב אחד לממציב אחר. גם אם יש בידינו אפשרות לשמר טבלאות ענקיות של value – \mathcal{Q} , לא יוכל לשפר את הטבלה ולערך בה ערכים אופטימליים, כיוון שלא ניתן להשליך ממציבים בהם ביצרנו על ממציבים חדשים. בכך להתמודד עם בעיה זו, נרצה להחליף את \mathcal{Q} בפונקציה שמחזירה ערך עבור כל זוג של state-action. במקום לחשב טבלה ענקית $|A| \times |S|$: \mathcal{Q} , נרצה למצוא פונקציה עם סט פרמטרים θ כך שיתקיים:

$$\mathcal{Q}_\theta(S, A) \approx \mathcal{Q}(S, A)$$

היתרון של שימוש בפונקציה שמנסה לשערר את הערכים הינו כפול: א. לא צריך לשמר טבלה בגודל של מרחב הממציבים. ב. כן ניתן ללמידה ממציבים שיש לנו עליהם ידע על ממציבים חדשים. בכך למצא פרמטרים שיישערו את המודל בזורה איקוותית יש לפתור בעיות אופטימיזציה, כפי שנגיד בהמשך, אך הלמידה יכולה להיות מאוד לא יציבה. ראשית, כפי שראינו ב-TD, המטרה אותה רוצים לשערר $(S_t) \mathcal{Q}$ או $(S_t, A_t) \mathcal{Q}$ משתנה בכל צעד. בנוסף, בוגד לאלגוריתמים הקודמיים, כתע יש קשר בין הממציבים, ולכן אם אנחנו מושנים ערך מסוים, זה גם ישפיע על ערכים אחרים, מה שמקשה מאוד על יציבות הלמידה.

הדוגמה פשוטה ביותר לפונקציה כזו הינה מודל לינארי מהצורה:

$$\mathcal{Q}(S, A) = w^T \phi(S, A)$$

מודל זה מניח שיש סט של פרמטרים המקיימים קשר לינארי בין ה-state-action ומפתח פיצרים קלשיה $\phi(S, A)$ לבין הערך שלהם $\mathcal{Q}(S, A)$. באופן פשוטו ביותר, בשביל למצוא את הפונקציה הזו, נרצה למצוור כמה שיטות את המרחק שבין $(S)^\pi \mathcal{Q}$ לבין $(S)^\pi \mathcal{V}_\theta$, ולשם כך נבנה את פונקציית המטרה הבאה:

$$L(\theta) = \frac{1}{2} \mathbb{E}_\theta \left[(\mathcal{Q}(S, A) - \mathcal{V}_\theta(S))^2 \right]$$

כמובן שחישוב זה אינו אפשרי משום שאנו לא יודעים מה הוא $(S)^\pi \mathcal{Q}$ – זה בדיקת הביטוי אותנו רוצים לשערר! בנוסף, חישוב התוחלת יכול להיות מאד מסובך, בטח במקרה בו אנו לא יודעים את כל הערכים האמיתיים של $(S)^\pi \mathcal{Q}$. בשביל להתגבר על בעיות אלו נשים לב כיצד כן ניתן לשפר את $(S)^\pi \mathcal{Q}$ – אם נגזר את פונקציית המטרה ונבצע צעד בכיוון הגרדיאנט, נקטין את ערכיה, וכך נרצה להיות מסוגלים לחשב את $\Delta \theta$. נבצע זאת בדומה בעזרת דגימות סטטיסטיות, כפי שכבר ראינו בפרק קודם, כלומר נרצה לדגום ממציבים מהסטרטgia בשביל לחשב את הביטוי הבא:

$$\Delta \theta = (\mathcal{V}_\theta(S_t) - \mathcal{V}_\theta(S_t)) \nabla_\theta \mathcal{Q}_\theta(S_t)$$

עם ביטוי זה עדין לא ניתן להתקדם כיון ש- $(S)^\pi$ לא ידוע, אך גם אותו ניתן לשער, למשל בעזרת MC או TD. בעזרת שיטת MC ניתן להריץ episode שלם, לחשב את התגמול המczטבר ולשים אותו במקום $(S)^\pi$, ובעזרת TD ניתן להריץ צעד אחד ולהחליפ את $(S)^\pi$ בתגמול המיידי והשערו של המצב הבא:

$$MC: \Delta\theta = (G_t - \mathcal{V}_\theta(S_t)) \nabla_\theta \mathcal{V}_\theta(S_t)$$

$$TD: (R_{t+1} + \gamma \mathcal{V}_\theta(S_{t+1}) - \mathcal{V}_\theta(S_t)) \nabla_\theta \mathcal{V}_\theta(S_t)$$

יש לשים לב שביביטוי האחרון יש שני איברים שתלויים ב- θ : $\mathcal{V}_\theta(S_{t+1})$ ו- $(S)^\pi$, אך נגזר רק את הביטוי הראשוני כיון שנרצה לשנות אותו כך שייתקרב לביטוי השני ולא להיפך. עם זאת, בנגוד למה שראינו לעיל בוגען-LTD, שם שינוי של $(S)^\pi$ לא השפיע על $(S)^\pi$ כיון שהעריכים בטבלה הוי בלתי תלויים אחד בשני, כאן כן יש השפעה בין שני ערכיהם אלו, מה שמקשה על היציבות של הלמידה.

בסיום של דבר, אנו יכולים לחשב את הביטוי Δ ובעזרתו לנסות לאפותם הפרמטרים באיזה שיטה אופטימיזציה סטנדרטית (למשל – stochastic gradient descent).

בדומה לשערוך שהצענו עבור MC ו-TD, ניתן גם לבצע control ולשערך את ה-table – \mathcal{Q} :

$$SARSA: \Delta\theta = (R_{t+1} + \gamma \mathcal{Q}_\theta(S_{t+1}, A_{t+1}) - \mathcal{Q}_\theta(S_t, A_t)) \nabla_\theta \mathcal{Q}_\theta(S_t, A_t)$$

$$Q - Learning: \Delta\theta = \left(R_{t+1} + \gamma \max_A \mathcal{Q}_\theta(S_{t+1}, A_t) - \mathcal{Q}_\theta(S_t, A_t) \right) \nabla_\theta \mathcal{Q}_\theta(S_t, A_t)$$

וגם כאן, בGESTRAה נרצה להתייחס ל- $\mathcal{Q}_\theta(S_{t+1}, A_{t+1})$ כקבועים ולגזר רק את $(S, A, \mathcal{Q}_\theta, \text{כיוון})$ שנרצה לשנות את $(S, A, \mathcal{Q}_\theta)$ כך שייתקרב בערךו לשאר הביטויים המבוססים על מידע נוסף.

יש כמה אתגרים שיכולים להקשוט מאוד על שיטות אלה להביא את המודל להתקנסות:

A. ראשית, הדרישה שאומරת לנו רצוי לשנות את $\mathcal{Q}_\theta(S_t, A_t)$ ביחס למידע העתידי הינה בעייתית, כיון שכעת כל הביטויים תלויים באותו פרמטר θ .

B. באלגוריתם שהוא off-policy אנו דוגמים מاستרטגיה ואפ' פועלים לפי הדגימות האלה, אך אנו לא מנסים להביא לאופטימום דזוקא את האסטרטגיה הזה (וממילא היא לא בהכרח האופטימלית עבורה). כאשר ניסינו ללמידה tabular Q-table, זה היה יכול לגורם לכך ששכיחות הממצאים שנדגומים אינה תואמת את השכיחות שלהם לפי האסטרטגיה האמיתית, מה שיכל להשפיע על קצב הלמידה, אך זה לא יגרום לשגיאות באסטרטגיה הנלמדת. כתע' כאשר יש קשר בין הממצאים על ידי הפרמטר θ , אז דגימה לא לפי השכיחות האמיתית יכולה להטעות את הלמידה לכיוונים שגויים.

אתגרים אלו מופיעים על יכולת המודל להתקנס לאסטרטגיה האופטימלית. בעוד שעבור יכולנו להוכיח התקנסות, עבור functional-approximation הנה פונקציית ההבאה:

Algorithm	Tabular	Linear	Non-Linear
MC	Y	Y	Y
SARSA	Y	Y	N
Q-Learning	Y	N	N

אחת השיטות פורצות הדרך שיתה שימוש ברשתות נירונים בתורת functional-approximation, ובשם הנפוץ שלה – Deep Learning. השימוש ברשת נירונית בפני עצמה לא הייתה מספקה כיון שכאמור הלמידה היא מאוד לא יציבה ויש להכין מנגנונים נוספים שיעזרו למודל להשתפר. נציג שתי טכניקות שהוותמעו בתחום הלמידה:

1. כאשר דוגמים כל מיני מצבים, יש ביניהם קורלציה יחסית גבוהה, מה שמנע מהשונות של הגדרי-אנט לקטון. כדי להתגבר על כך, במקומות לעדכן את \mathcal{Q} בכל שלב, שומרים את הרבייעיות $(S_t, A_t, R_{t+1}, S_{t+1})$ של N המצבים האחרונים שראינו (למשל: $N = 1,000$), ואז מעדכנים לפי k מצבים (למשל: $k = 10$) אקרים מתחום N המצבים הקודמים שיש לנו בזיכרון (לאחר שהזיכרונו מתמלא, אך כל מצב חדש שmag'ע דורס את המצב הכי ישן בזיכרון). בצורה זו

השונות תקען כי הדוגמאות שנעדכן לפיהן יהיו בעלות קורלציה נמוכה הרבה יותר. המחיר של טכניקה זו הוא השימוש בהרבה זיכרון, אבל יש לה השפעה משמעותית על הביצועים.

2. הזכירנו לעיל שגם רוצים לשנות את $(S_t, A_t) \mathcal{Q}$ ביחס למידע העתידי, אך זה יכול להיות בעיתוי, כיוון שכעת כל הביטויים תלויים באותו פרמטר θ . במקרה אחרות – אנו רוצים לקרב את האסטרטגיה למטרה מסוימת, אך שתיהן תלויות באותו פרמטר. במקרה לייצב את הרגשותה, ניתן לשנות את המטרה באופן הרבה פחות תדר. באופן פורמלי, נשמר סט פרמטרים $\hat{\theta}$ ונשנה אותו כל מספר קבוע של צעדים לסט הפרמטרים החדש, ככלمر – כל k צעדים נעדכו את $\hat{\theta}$ להיות שווה $-l_t$. יחד עם הפרמטרים החדשם, נשנה מעט את הביטוי אותו נרצה להביא לאופטימום:

$$\mathbb{E}_{S_t, R_{t+1}, A_t, S_{t+1}} \left[R_{t+1} + \gamma \max_A \mathcal{Q}_{\hat{\theta}}(S_{t+1}, A) - \mathcal{Q}_{\theta}(S_t, A_t) \right]$$

11.3.4 Policy-Based RL

יש כמה מגבלות וחסכנות בשיטות שראינו עד כה. שיטות אלו מתמקדות בלמידת \mathcal{Q} – \mathcal{Q} , בין אם על ידי עדכון ישיר של ערכי הטבלה ובין אם על ידי למידת הפונקציה $(S, A) \mathcal{Q}$ התלויה בפרמטר θ . ראיינו שבבעיות גדולות הייצוג והלמידה יכולים להיות קשים.אתגר זה מתעצם ואנו נהיה בלתי אפשרי עבור בעיות רציפות, בהן מרחב המצבים הוא כביכול אינסופי (למשל – תנועה של רובוט). חסרון נוסף שיש בגישות שראינו נועז במטרה של הלמידה – אנו מנסים ללמידה את $(S, A) \mathcal{Q}$ אך בפועל מה שמעניין אותנו זה $\max_A \mathcal{Q}_{\theta}(S, A)$, מה שיכל להביא לבזוז משאבים עבור למידה של דברים לא הכרחיים. למשל – אם אנחנו צריכים להחליט בין להשקיע כסף בשוק ההון לבין השקעה בתרמיה פירמידה, אז אנחנו רק צריכים להחליט איפה איזו אופציה לנו, אך אנו לא נדרשים לדעת מה בדיקן כוללת כל אופציה. החלטה בין האופציות היא פשוטה, ואילו ללמידה מה ואיך להשקיע בשוק ההון זה דבר מורכב. לכן במקומות ללמידה את כל המערכת, נוכל פשוט ללמידה מה הפעולות הנדרשות עבורנו, וגישה ישירה זו יכולה לחסוך במשאבים.

כאמור, במקומות ללמידה את \mathcal{Q} , ננסה ללמידה באופן ישיר את האסטרטגיה האופטימלית – $(s|a) \pi$. באופן מעט יותר פורמלי, נגדיר את פונקציית המטרה:

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} \right]$$

מציאת האסטרטגיה האופטימלית שקופה למציאת המקסימום של פונקציית המטרה $(\theta) \mathcal{J}$.

נרצה להשתמש באסטרטגיה סטוכסית מכמה סיבות. ראשית, הרבה פעמים יותר קל למציאו אופטימום עבור בעיה רציפה מאשר בעיה דיסקרטית, ואסטרטגיה סטוכסית יכולה להפוך בעיה דיסקרטית לרציפה. שנית, יהיו מקרים בהם נתוניינו דווקא באסטרטגיה הסטוכסית ולא בזו הדטרמיניסטית. בנוסף, הסטוכסיות מאפשרת exploration, וכפי שראינו זה מופיע הכרחי בשלégung לכל המצבים. במבט יותר רחב, אסטרטגיה סטוכסית היא הכללה של אסטרטגיה דטרמיניסטית – אם כל הנסיבות של המודל הינם 0 או 1. במובן זהה, השימוש באסטרטגיה סטוכסית לא ממעט מה יכולת של אסטרטגיה דטרמיניסטית אלא הוא רק מכך אותה.

באופן הכי פשוט, עבור בעיה בה מרחיב בפעולות הינו דיסקרטי, ניתן להפוך אסטרטגיה שמספקת ערך לכל פעולה $\phi(a; x, \theta)$ להתפלגות על ידי SoftMax:

$$\pi_\theta(a|s) = \frac{\exp(\phi(a; x, \theta))}{\sum_{a'} \exp(\phi(a'; x, \theta))}$$

אם מרחיב הפעולות הוא רציף, ניתן להשתמש בගאוסיאן:

$$\pi_\theta(a|s) \sim \mathcal{N}(\mu(x; \theta), \Sigma(x; \theta))$$

ואם רוצים לאפשר יותר חופש פעולה, ניתן להשתמש ב-*Gaussian mixture model*. הפונקציה שבאמצעותה רוצים לייצג את האסטרטגיה יכולה להיות כרzonנו – פונקציה לינארית, רשת ניירונים וכו'.

רוב האלגוריתמים שעושים אופטימיזציה $-(\theta) \mathcal{J}$ מבוססי גרדיאנט, אך ישנים גם אלגוריתמים אחרים, כמו למשל אלגוריתמים גנטיים, אך עליהם לא נדבר. הדרך הסטנדרטית לבצע אופטימיזציה היא להשתמש ב-*gradient descent/ascent* ו-*stochastic gradient descent/ascent*, והמקדש לנו יהיה בשאלת

כיצד לשערך את הגרדיינט ופחות נתעסק בשיטות האופטימיזציה השונות. האלגוריתם הבסיסי מבוסס על עדכון מהצורה הבאה:

$$\theta_{t+1} = \theta_t + \alpha \nabla f(\theta)$$

נניח שיש לנו MDP episodic, ומהאסטרטגיה הנתונה נוכל לדגום N מסלולים $\pi \sim \pi_N, \dots, \tau_1, \tau$, כאשר כל מסלול הינו $G_i = \sum_{t=0}^{T(i)} \gamma^t R_{t+1}^{(i)}$. שערוך מונטה-קרלו הרגיל ישער את המטרה באופן הבא:

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} \right] \approx \frac{1}{N} \sum_i G_i$$

הבעיה היא שהאפקט של הפרמטר θ על המסלול G_i היא לא ישירה – אנחנו דוגמים מסלול מאסטרטגיה π התלויה בפרמטר θ , ואז מחשבים את התגמול של המסלול וממנו מרכיבים את G . لكن אי אפשר פשוט לגזר את G_i לפי θ , שריי הקשר ביניהם הוא עקיף ותליי ב- $f(\theta)$. כדי להתמודד עם בעיה זו משתמש ב-trick log-derivative (לעתים הenthal'יך זהה נקרא Reinforce). נגדיר:

$$g(\theta) = \mathbb{E}_{\pi_\theta}[f(x)]$$

ואז הנגזרת הינה:

$$\nabla_\theta g = \mathbb{E}_{\pi_\theta}[f(x) \nabla \log(\pi_\theta(x))]$$

הוכחה:

$$\nabla_\theta g = \nabla_\theta \mathbb{E}_{\pi_\theta}[f(x)] = \nabla_\theta \int \pi_\theta(x) f(x) dx$$

נחליף את הסדר של הגרדיינט והאינטגרל:

$$= \int \nabla_\theta \pi_\theta(x) f(x) dx$$

cutet בשימוש בקשר הבא (המתוקים עברו כל פונקציה): $\frac{\nabla_\theta h(x)}{h(\theta)}$, ונקבל:

$$= \int \nabla_\theta \log(\pi_\theta(x)) \pi_\theta(x) f(x) dx$$

זהו בדיק שווה לתוחלת הבהאה:

$$= \mathbb{E}_{\pi_\theta}[f(x) \nabla \log(\pi_\theta(x))]$$

cutet, במקום לחשב את הגרדיינט של G_i , נוכל לחשב את התוחלת $\mathbb{E}_{\pi_\theta}[f(x) \nabla \log(\pi_\theta(x))]$ על ידי דגימות ושערור מונטה קרלו, רק cutet הנגזרת אותה אנו צריכים לחשב אינה תלולה בפרמטר θ ולכן ניתן לחשב אותה בצורה ישירה. בצורה מפורשת, פונקציית המטרה שלמו הינה:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[G(\tau)], G(\tau) = \sum_{t=0}^{T(\tau)} \gamma^t R_{t+1}^{(\tau)}$$

ושימוש ב-trick log-derivative יביא אותנו לביטוי:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[G(\tau) \nabla \log(\pi_\theta(\tau))]$$

נשים לב שהאסטרטגיה $(\tau) \pi$ מרכיבת משני אלמנטים – הפעולה שאנו בוחר לבצע, והמצב אליו אני מגיע יחד עם התגמול המתkeletal מצעד זה. החלק הראשון הוא לגמר בשליטה שלנו ואנו יודעים אותו, אך החלק השני הוא דינמי ותלו依 בסטוכסטיות של המערכת, ולכן לא ידוע ולא ניתן לבצע את הגזירה. אך אם נפתח את הביטוי שקיבלנו נראה שכל החלקים התלויים בדינמיות של המערכת אינם תלויים ב- θ ויתאפסו בגזירה. ניתן לרשום את האסטרטגיה כמכפלת ההסתברויות של כל הצעדים באופן הבא:

$$\pi_\theta(\tau_i) = P(S_0^i) \cdot \pi_\theta(A_0^i | S_0^i) \cdot P(S_1^i, R_1^i | S_0^i, A_0^i) \cdots \pi_\theta(A_{T^i-1}^i | S_{T^i-1}^i) \cdot P(S_{T^i}^i, R_{T^i}^i | S_{T^i-1}^i, A_{T^i-1}^i)$$

cut-off אם נוציא לוג, המכפלה תהפוך לסכום:

$$\log(\pi_\theta(\tau_i)) = \log(P(S_0^i)) + \sum_{t=0}^{T^i-1} \log(\pi_\theta(A_t^i | S_t^i)) + \sum_{t=0}^{T^i-1} \log(P(R_{t+1}^i, S_{t+1}^i | S_t^i, A_t^i))$$

הביטוי האמצעי מבטא את בחירות הצעדים של המשתמש, והוא **היחיד** שתלי' בפרמטר θ . שני הביטויים הנוספים מבטאים את הדינמיות של המערכת שאין לנו מידע כלפייהם, והם אינם תלויים ב- θ , لكن בנגזרת לפי θ הם מתאפסים. עובדה זהה בעצם מספקת יתרון נוסף לשימוש בטרייק של הלוג – מלבד האפשרות לחשב את הנגזרת באופן ישיר, הלוג גם מפריד בין הצעדים של המשתמש לבין הדינמיות הלא ידועה של המערכת. لكن בסך הכל נקבל:

$$\nabla_\theta \log(\pi_\theta(\tau_i)) = \sum_{t=0}^{T^i-1} \nabla_\theta \log(\pi_\theta(A_t^i | S_t^i))$$

cut-off נציב את זה בגרדיאנט של פונקציית המטרה ונשתמש בדגימות מונטה קרלו:

$$\begin{aligned} \nabla J(\theta) &= \mathbb{E}_{\tau \sim \pi_\theta}[G(\tau) \nabla \log(\pi_\theta(\tau))] \approx \frac{1}{N} \sum_i G_i \nabla_\theta \log(\pi_\theta(\tau_i)) \\ &= \frac{1}{N} \sum_i \left(\sum_{t=0}^{T^i-1} \gamma^t R_{t+1}^i \right) \left(\sum_{t=0}^{T^i-1} \nabla_\theta \log(\pi_\theta(A_t^i | S_t^i)) \right) \end{aligned}$$

חישוב הגראדיאנט בצורה זו נקרא בשם **policy gradient**. cut-off משחישבו את הגראדיאנט, נוכל להשתמש בשיטות אופטימיזציה סטנדרטיות על מנת למצוא את θ האופטימלי. בקרה נוכל לסכם את כל התהליך ה-**Reinforce** בשלושה שלבים:

1. Run the policy and sample $\{\tau^i\}$ from $\pi_\theta(A_t | S_t)$.
2. Estimate gradients: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \left(\sum_{t=0}^{T^i-1} \gamma^t R_{t+1}^i \right) \left(\sum_{t=0}^{T^i-1} \nabla_\theta \log(\pi_\theta(A_t^i | S_t^i)) \right)$.
3. Improve the policy using gradient descent/ascent: $\theta \leftarrow \theta_\alpha \nabla_\theta J(\theta)$.

נתבונן על הביטוי של הנגזרת שקיבלו וננסה לנתח אותו. נניח שיש מומחה שיוודע לייצר trajectories (imitation learning) – במקורה כזה נרצה לקחת את מה שהוא מייצר ולגרום לכך שהאסטרטגיה שלנו תיציר בהסתברות יותר גבוהה דוגמאות כמו שקיבלו מומחה. במקרים אחרים, כאשר נתונים לנו מסלולים מוצלחים, נרצה לגרום לאסטרטגיה שלנו "לחקות" את אותם מסלולים. הדרך הסטנדרטית לבצע זאת מושימה היא באמצעות maximum likelihood – לקחת את המסלולים האלה ולסוכם אותם, ואז לחפש את המקסימום באמצעות גזירה של הביטוי המתקובל. באופן שקול ניתן גם לחפש מקסימום עבור הסכום של **לוג** המסלולים, ובכך הכל נרצה לחשב את הביטוי:

$$\sum_{t=0}^{T^i-1} \nabla_\theta \log(\pi_\theta(\tau_i))$$

זה בדיקת הביטוי השני בגראדיאנט שראינו קודם. בנוסף גמ' את הגורם הראשון שמשמעותו כל מסלול בהתאם ל-reward, כך שמסלולים בעלי reward גבוה יקבלו משקל גבוה וailleurs מסלולים נמוך. יקבלו משקל נמוך. יצא מכך, שהgraadiאנט מנסה להביא לכך שמסלולים "מוצלחים" יופיעו בהסתברות יותר גבוהה.

השעורך של graadiאנט באמצעות הביטוי שראינו הוא אמן חסר הטיטה, אך יש לו שנות גבוהה. בנוסף, אם כל הדגימות הם בעלי reward חיובי (או כלן בעלי reward שלילי), אז יהיה קשה לבדוקין איזה צעדים יותר טובים ומה לא כדאי לעשות. נראה שתי דרכי לשיפור היציבות של השימוש בgraadiאנט:

בנסיבות (casualty) – כיוון שהצעד a_t שהתרחש בצעד t לא יכול להשפיע על כל הצעדים שקרו לפני כן, ניתן להחלף את הביטוי $\sum_{t=0}^{T^i-1} \gamma^{t-1} R_{t+1}^i$ בביטוי שמתחשב רק בצעדים שקרו החל מצעד t , כלומר: $\{a_0, a_1, \dots, a_{t-1}\}$ – ביטוי זה נקרא גם **Reward to go**. הخلافה זו מוריידה את השונות כיוון שהיא גורמת לכך שככל

צעד יהיה תלוי בפחות איברים אקראיים. הביטוי שהתعلמנו ממנו הוא $\sum_{k=0}^{t-1} \gamma^k \mathcal{R}_{k+1}^i$, וניתן לראות שכן אין לו השפעה על העתיד.

Baseline – ניתן להראות שהוסף של קבוע ל-reward משפיע על התוצאה של הגרדיאנט. קבוע מסוים יכול לגרום לכך שנרצה לשפר צעדים מסוימים ולהימנע מצעדים אחרים, ואילו קבוע אחר יכול לגרום לכך שנרצה לשפר דו-קאה צעדים אחרים. קבוע זה הוא למעשה פרמטר נוסף שנייה לשלוט בו, ועל ידי בחירה מושכלת שלו ניתן להפחית את השונות. באופן פורמלי, את הביטוי $\sum_{t=0}^{T^i-1} \gamma^t \mathcal{R}_{t+1}^i$ נחליף בביטוי זהה עם תוספת קבוע:

$$\sum_{t=0}^{T^i-1} \gamma^t \mathcal{R}_{t+1}^i - b$$

השאלה היא כיצד לבחור את הקבוע b בצורה טובה? באופן פשוט ניתן לבחור את הקבוע בהתאם הממוצע של ה-rewards. בחירה כזו תביא לכך שהיא שמתוחת לממוצע נרצה להוריד את ההסתברות שלו ומה שמעל הממוצע נרצה לחזק אותו. בחירה כזו akan מזרידה את השונות, אך היא איננה אופטימלית. ניתן לחשב את הקבוע האופטימלי בצורה מדויקת. הנגזרת של פונקציית המטרה יחד עם האיבר הנוסף הינה:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [(G(\tau) - b) \nabla \log(\pi_\theta(\tau))]$$

הממוצע אינו תלוי ב- b , لكن נוכל לרשום את השונות כך:

$$\begin{aligned} Var &= \mathbb{E}_{\tau \sim \pi_\theta} [(G(\tau) - b)^2 \nabla \log(\pi_\theta(\tau))^2] + C \\ &= \mathbb{E}_{\tau \sim \pi_\theta} [G(\tau)^2 \nabla \log(\pi_\theta(\tau))^2] + \mathbb{E}_{\tau \sim \pi_\theta} [b^2 \nabla \log(\pi_\theta(\tau))^2] - 2\mathbb{E}_{\tau \sim \pi_\theta} [G(\tau) \cdot b \nabla \log(\pi_\theta(\tau))^2] + C \end{aligned}$$

כעת נגזר את השונות לפי b :

$$\begin{aligned} \frac{dVar}{db} &= 2\mathbb{E}_{\tau \sim \pi_\theta} [b \nabla \log(\pi_\theta(\tau))^2] - 2\mathbb{E}_{\tau \sim \pi_\theta} [G(\tau) \cdot b \nabla \log(\pi_\theta(\tau))^2] = 0 \\ &\rightarrow b_{opt} = \frac{\mathbb{E}_{\tau \sim \pi_\theta} [G(\tau) \nabla \log(\pi_\theta(\tau))^2]}{\mathbb{E}_{\tau \sim \pi_\theta} [b \nabla \log(\pi_\theta(\tau))^2]} \end{aligned}$$

הביטוי המתתקבל הוא אכן הממוצע של reward, אך מוכפל במשקל התלוי בנגזרת של לוג האסטרטגיה. השערור של הביטוי המדויק יכול להיות מסויך ורועש, ולכן קיימות שיטות פשוטות יותר**ב-*reward***.

אלגוריתם policy gradient הינו, כאמור האסטרטגיה שבודرتה אנו דוגמים מסלולים הינה אותה אסטרטגיה שאנו מנסים להביא לאופטימום. כאמור לעיל, באלגוריתמים שהם policy-on, אי אפשר ל"מחזר" דגימות, כלומר בכל נקודת זמן ניתן להשתמש רק בדגימות שנלקחו מהאסטרטגיה הנתונה באותה נקודת. עברו מצלבים בהם הדגימות יקרות, נרצה להשתמש בהן דגימות עבור נקודות זמן שונות, וכך להפוך את האלגוריתם להיות-off.政策. בכך לעשויות זאת שلتakin את השגיאה שנוצרת מכך שננקודות זמן מסוימת אנו משתמשים בדגימות שנלקחו מאסטרטגיה שכבר השתנה וכעת התרפלוות של האסטרטגיה שונה. במקרה אחר – יש לנו דגימות שנלקחו מתוחלת מסוימת, ובעזרתן אנו רוצים לשערר תוחלת אחרת. זו בעיה מתוחום הסטטיסטיקה, והפתרון שלו נקרא **Importance sampling**

נניח ואני רוצים לשערר את $\mathbb{E}_p[f(x)]$, אך אנו לא יכולים לדוגם מהתפלגות p אלא רק מהתפלגות q , כאשר ההנחה היחידה שלנו היא שאם $0 < p < q$. באמצעות מעבר מתמטי ד' i פשוט ניתן לייצג את התוחלת של p באמצעות התוחלת של q :

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx = \int f(x) \frac{p(x)}{q(x)}q(x)dx = \mathbb{E}_q \left[f(x) \frac{p(x)}{q(x)} \right]$$

כעת נדגים q -ים נושא את התוחלת באמצעות מונטה קרלו:

$$\mathbb{E}_p[f(x)] = \mathbb{E}_q \left[f(x) \frac{p(x)}{q(x)} \right] \approx \frac{1}{N} \sum_i f(x_i) \frac{p(x_i)}{q(x_i)}$$

אם ההתפלגיות $(x, q(x))$ הן יחסית דומות, אז התוצאה המתקבלת משערת בצורה יחסית טובה את התוחלת הרציה. אם זה לא המצב, השונות תהיה גדולה והטואה תהיה לא יציבה. מכל מקום, נוכל להשתמש בReLU זה עבור שערוך הגראינט:

- For trajectory probability the dynamics does not depend on θ so

$$\frac{\pi_{\theta'}}{\pi_{\theta}} = \frac{\prod \pi_{\theta}(a_t | s_t)}{\prod \pi_{\theta'}(a_t | s_t)}$$

- Can now Compute $\nabla_{\theta'} J(\theta')$ with samples from π_{θ}

- $\sum_{i=1}^N \left(\sum_{t=0}^{T^{(i)}-1} \gamma^t r_{t+1}^{(i)} \right) \left(\sum_{t=0}^{T^{(i)}-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \right) \frac{\prod \pi_{\theta}(a_t | s_t)}{\prod \pi_{\theta'}(a_t | s_t)}$

- Can improve with causality

- $\sum_{i=1}^N \sum_{t=0}^{T^{(i)}-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) \frac{\prod_{k=0}^t \pi_{\theta}(a_k | s_k)}{\prod_{k=0}^t \pi_{\theta'}(a_k | s_k)} \left(\sum_{k=t}^{T^{(i)}-1} \gamma^k r_{k+1}^{(i)} \right)$

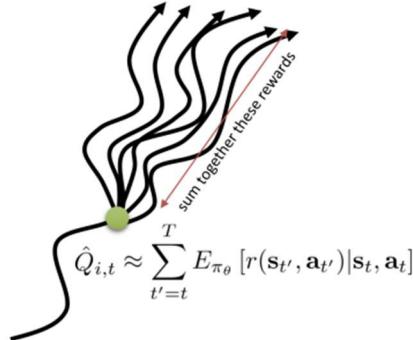
כאמור, שימוש בReLU זה יכול להיות טוב אם ההתפלגיות θ' יחסית דומות. אם הן רחוקות, התוצאה המתקבלת לא בהכרח מהוות שערוך טוב, ולכן הלמידה לא תהיה מספקת טוביה. לסיום – אפשר להפוך את האלגוריתם של אופון ישיר (Policy gradient) בamusance off-policy sampling לשובן שוב בביטוי policy gradient

11.3.5 Actor Critic

בפרקים הקודמים דיברנו על שתי גישות כדי לשערך את האסטרטגיה בלי לדעת המודל. גישה אחת התמקדה בשערוך ה-Action-Value function (SARSA/Q-Learning) – Action-Value function (Q-Learning). בעוד הגישה השניה ניסתה לשערוך את האסטרטגיה באופן ישיר (Policy gradient). בפרק זה נציג אלגוריתם המנסה לשלב את שתי הגישות יחד. נתבונן שוב בביטוי של הגראינט, כאשר בהתייחסות ל-rewards נתווסף רק ל-go reward:

$$\frac{1}{N} \sum_i \left(\sum_{k=t}^{T^i-1} \gamma^k \mathcal{R}_{k+1}^i \right) \left(\sum_{t=0}^{T^i-1} \nabla_{\theta} \log (\pi_{\theta}(A_t^i | S_t^i)) \right)$$

ניתן לבדוק כי הביטוי $\sum_{k=t}^{T^i-1} \gamma^k \mathcal{R}_{k+1}^i$ הוא למעשה דגימה של $\mathcal{Q}^{\pi_0}(s_t, a_t)$, כלומר עבור נקודת זמן, ניתן לדגם מסלול באמצעות ה-Action-Value function ולקבל סכום של rewards. החיסרון בדגימה זו נעוץ בכך שיש לה שנות גבואה והיא יכולה להיות מאוד רועשת. כדי שניתן לראות באיזו, תכננו הרבה מסלולים שונים היוצאים מאותה נקודת, וכן הביטוי של ה-*reward to go* הוא מאוד רועש ביחס ל-



איור 11.9 מסלולים שונים אפשריים היוצאים מאותה נקודת הביטוי/התוחלת של הביטוי אמנים מורד את השונות ל-0, אך כל דוגמה בפני עצמה היא בעלת שונות גבוהה.

הבחנה זו של הקשר בין reward to go לבין $Q^{\pi_0}(s_t, a_t)$ מעלה את השאלה האם אפשר להחליף את בין הביטוי הרועש $\sum_{k=1}^{T-1} \gamma^k \mathcal{R}_{k+1}^i$ לבין הפונקציה עצמה $(s_t, a_t) Q^{\pi_0}$. בשביל לבצע החלפה זו, יש להוכיח שהיא אכן חוקית ושמורה על תוצאה נכונה, וכך לעשות זאת באמצעות המתארה כך שתופיע בצורה יותר נוחה. כזכור, פונקציית המתארה אותה אנו מעוניינים להביא לאופטימום הינה התוחלת של rewards-discount (מכפלים ב-factor

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} \right]$$

המשמעות של התוחלת היא לבצע אינטגרל על כל צמד של state-action (s, a) , כפול המשקל של אותו צמד. כיוון שצמד יכול לחזור על עצמו בזמנים שונים (כלומר בשתי נקודות זמן t_1, t_2 המערכת יכולה להיות באותו מצב ולבצע את אותו פעולה), אך למעשה באינטגרל ישנים ביטויים שחוורים על עצמם. כדי להימנע מכך, ניתן לחשב כל ביטוי באינטגרל פעם אחת, ולהכפיל אותו במסקל המתאים למספר הביקורים באותו צמד action-state. עם זאת, עדין צריך לזכור שה貌עים של אותו צמד נבדלים זה מזה ב-discount factor, כיוון שהמשקל של reward בזמן t_1 שונה מהמשקל שלו בזמן t_2 , ויש לקחת את ההבדל הזה בחשבון. כתע נתבונן על המפורש של האינטגרל המתkeletal, כאשר ראשית נחליף את סדר התוחלת והסכימה:

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} \right] = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi_\theta} [\mathcal{R}_{t+1}]$$

התוחלת של כל rewards מרכיבת מיצוע שלם על פניו: 1) ההסתברות להתחיל במצב s_0 כפול 2) ההסתברות להגיע במצב s בזמן t בהינתן שהתחלנו מ- s_0 כפול 3) הסיכוי לבצע a במצב s שהתקבל בזמן t . באופן פורמלי הביטוי המתkeletal הינו

$$\begin{aligned} &= \sum_{t=0}^{\infty} \gamma^t \int_S \int_S \int_A p(s_0) p_t^\pi(s|s_0) \pi(a|s) \mathcal{R}(s, a) ds_0 ds da \\ &= \int_S \left(\sum_{t=0}^{\infty} \gamma^t \int_S p(s_0) p_t^\pi(s|s_0) ds_0 \right) \int_A \pi(a|s) \mathcal{R}(s, a) ds da \end{aligned}$$

נסמן את הביטוי שבסוגרים $C(s)^\pi$, והמשמעות שלו היא מה הסיכוי להגיע ב- t צעדים ממצב s_0 למצב s , כאשר עברו כל t יש להכפיל ב-discount factor. ביטוי זה נקרא Discount state visitation measure, ולמעשה הוא כולל בתוכו את כל המופעים של צמד (s, a) , ובכך במקומם לעשות אינטגרל על אותו צמד (a) . מספר פעמים, עושים זאת עבור כל נקודות הזמן השונות ומכפילים ב-discount factor את המתאים. באופן מפורש:

$$= \int_S \rho^\pi(s) \int_A \pi(a|s) \mathcal{R}(s, a) ds da,$$

$$\rho^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \int_s p(s_0) p_t^\pi(s|s_0) ds_0$$

יש לשים לב שהביטוי הכלול הוא לא בצורה הפשטota של תוחלת – אינטגרל של הסתברות כפول פונקציית צפיפות, אך הוא עדין משקף תוחלת ולכן ניתן לשער אותו באמצעות מונטה-קרלו. כתע נתבון על הגדיאנט של פונקציית המטרה. אנחנו רוצים לגזר את הביטוי הבא לפ' θ :

$$\mathcal{J}(\theta) = \int_S \rho^\pi(s) \int_A \pi(a|s) \mathcal{R}(s, a) ds da$$

כל האיברים תלויים ב- θ אך ניתן להראות (שколоויות 11-10 במצגת 7) שהנגזרת תליה רק ב- $(s|a)\pi$, כלומר:

$$\nabla_\theta \mathcal{J} = \int_S \rho^\pi(s) \int_A \nabla_\theta (\pi(a|s)) \mathcal{Q}^{\pi_\theta}(s, a) ds da$$

כעת ניתן להיעזר ב-log-derivative trick ולקבל:

$$\nabla_\theta \mathcal{J} = \int_S \rho^\pi(s) \int_A \pi(a|s) \nabla_\pi \log(\pi(a|s)) \mathcal{Q}^{\pi_\theta}(s, a) ds da$$

ואז לשער את הביטוי באמצעות דגימות מונטה-קרלו:

$$\nabla_\theta \mathcal{J} \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T^{(i)}-1} \nabla_\theta \log \pi_\theta \left(a_t^{(i)} | s_t^{(i)} \right) \mathcal{Q}^{\pi_\theta} \left(s_t^{(i)}, a_t^{(i)} \right)$$

הביטוי המתקיים טוב מאוד מבחינת זה שהוא מצליח להפחית את השונות, אך הבעיה בו נועצה בכך שהאיבר $\mathcal{Q}^{\pi_\theta}(s_t^{(i)}, a_t^{(i)})$ אינו ידוע! אלגוריתם **Actor Critic** בא להתמודד עם בעיה זו, והרעיון הוא לנסוט לשער את \mathcal{Q}^{π_θ} באמצעות \mathcal{Q}^w במקביל לשערו של הגדיאנט. כאמור, בתהליך הלמידה אנו מנסים לשער שני דברים במקביל:
 π_θ – הגורם המחליט איזה צעד לנוקוט בכל מצב.
 \mathcal{Q}^w – הגורם המבצע אבולוציה של \mathcal{Q} על מנת לשפר את הבחירה של ה-Actor.

הurret אגב: לעיל ראיינו שהשימוש ב- \mathcal{Q} – \mathcal{Q} יכול להיות בעיתי בבעיות רציפות בגלל שמרחב הממצבים הוא אינסופי ומאוד קשה לפתור את בעיית האופטימיזציה של בחירת action s בערך הביטוי $\pi_\theta(s, a)$. CAN בעיה זאת איננה, כיון שאנו נאדרשים לבחור את ה-action \mathcal{Q} באמצעות \mathcal{Q} אלא רק לשפר באמצעותו את האסטרטגיה (או במילים אחרות – אנחנו לא צריכים לפתור בעית אופטימיזציה על \mathcal{Q} עבור מרחב ממצבים רציף).

האלגוריתם פועל בהתאם לשלבים הבאים. ראשית נתחל את w, θ . כתע נבצע T איטרציות באופן הבא:

- נדגום מצב התחלתי s_0 מתוך מרחב הממצבים S ופועלה מתוך האסטרטגיה $(s_0 | \cdot) \sim \pi_\theta(a|s_0)$.
- כתע כל עוד לא הגיענו למצב הסופי:
- נחשב את המצב החדש s_{new} ואת התגמול המתkeletal r .
- בנויסף, נדגום פעולה חדשה $(s_{new} | new) \sim \pi_\theta(a | new)$.
- נחשב את ה-TD error:

$$\delta = r + \gamma \mathcal{Q}^w(new s, new a) - \mathcal{Q}^w(s, a)$$

- נעדכן את הפרמטרים:

$$\begin{aligned} \theta &\leftarrow \theta + \alpha \mathcal{Q}^w(s, a) \nabla_\theta \log \pi_\theta(a|s) \\ w &\leftarrow w + \beta \delta \nabla_w \mathcal{Q}^w(s, a) \end{aligned}$$

- נעדכן את המצב והפעולה:

$$a = new a, s = new s$$

פואדו קוד עבור האלגוריתם:

```

Initialize parameters  $s, \theta, w$  and learning rates  $\alpha_\theta, \alpha_w$ ; sample  $a \sim \pi_\theta(a|s)$ .
for  $t = 1 \dots T$ : do
    Sample reward  $r_t \sim R(s, a)$  and next state  $s' \sim P(s'|s, a)$ 
    Then sample the next action  $a' \sim \pi_\theta(a'|s')$ 
    Update the policy parameters:  $\theta \leftarrow \theta + \alpha_\theta Q_w(s, a) \nabla_\theta \log \pi_\theta(a|s)$ ; Compute
    the correction (TD error) for action-value at time t:
         $\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s, a)$ 
    and use it to update the parameters of Q function:
         $w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s, a)$ 
        Move to  $a \leftarrow a'$  and  $s \leftarrow s'$ 
end for

```

.איור 11.10 אלגוריתם reward-to-go Policy gradient במקומ הביטוי של ה- Q .

חשוב להזכיר שהעדכן של הגרדיינט מתבצע בדומה ל-SARSA ולא כמו Learning Q – Learning Policy gradient. כאמור לעיל, ההבדל ביןיהם הוא באופי האלגוריתם – SARSA הוא אלגוריתם policy-on, כלומר הוא מסה לאפטם את האסטרטגיה ממנו הוא דוגם את המסלולים, בעוד Q מנסה ללמוד את האסטרטגיה האופטימלית בעלי תוצאות בדוגמאות אותן הוא רואה. אלגוריתם Actor Critic רוצה לשפר את ה- Q , כלומר המטרת היא לשפר את האסטרטגיה הנתונה, וכך יש לבצע מידה שהיא On policy.

בדומה ל- Q , גם Learning Policy gradient ניתן להוסיף בדומה על מנת לשפר את תהליכי הלמידה. בנגדוד ל- Q הוא התוספת היתה של קבוע, כאן נהוג להוסיף את $(s)^\pi$, שהוא תלוי ב- s . אם ה- Q baseline- s רק במצב s , אז זה בסדר כפי שנויכח מיד (שוף 16), אבל אם הוא תלוי גם ב-action a , אז זה מוסיף bias וહלמיה תגעה. הביטוי: $(s)^\pi = Q^\pi(s, a) - A^\pi(s, a)$ נקרא A^π advantage function, והוא הינה לבירה שלו C -baseline. היא פשוטה – אם הערך של $(s)^\pi$ חיובי, אז זה סימן לכך שהבחירה בצד a היא טובה, וכן נרצה לחזק אותה. אם לעומת זאת הערך של $(s)^\pi$ שלילי, זה אומר שהפעולה לא טובה ומילא נרצה להימנע منها. במקרה אחר: $(s)^\pi$ הינו הערך של פעולה ספציפית עבור מצב s , וכך נוכל לדעת האם נרצה做强 את הפעולה זו או לא.

לאחר שראינו שניין להשתמש ב- Q על מנת לשפר Actor Critic ויש היגיון להשתמש ב- $(s)^\pi$ בטור ה- Q baseline, נראה כיצד ניתן ליישם זאת בפועל. אם מרחיב המצבים הוא DISKRTI, ניתן ללמידה את $(s)^\pi$ בשיטות הסטנדרטיות שראינו, אך לחשב באופן מדויק את $(s)^\pi$ בעזרת הקשר $(s)^\pi = \sum_a \pi(a|s) Q^\pi(s, a)$. במצבים מסוימים ניתן להשתמש באותו רעיון גם עבור מרחיב מצבים רציף – נניח $(s)^\pi$ הוא לינארי ביחס למרחב הפעולות, כלומר מתקיים: $\phi(s)^\pi = \phi(s)^\pi$, והאסטרטגיה היא גאומטרית: $(s)^\pi = \mu(s) + \sigma^2(s)$. אז $(s)^\pi$ הוא התוחלת של $(s)^\pi$ על פני כל הפעולות: $(s)^\pi = \mathbb{E}_{a \sim \pi_\theta} [Q^\pi(s, a)]$. בעזרת הנוסחה הסוגורה זו ניתן להשתמש בשערור $(s)^\pi$ ולחשב את $(s)^\pi$ ואז להשתמש בו כ-base.

כאשר לא ניתן להשתמש ב- $(s)^\pi$ על מנת לחשב את $(s)^\pi$, ניתן לנסות לשערר את שני הביטויים בנפרד, אך זה יכול להיות קשה. לחילופין, ניתן לבנות מודל (למשל רשת נירונים) שתנסה לשערר במקביל את שני הביטויים, כלומר הפלט שלה יהיה גם $(s)^\pi$ וגם $(s)^\pi$. גישה אחרת מציעה להשתמש במשוואות בלבד. לפי משוואות אלו מתקיים בקירוב $(s_t, a_t)^\pi = r_{t+1} + \gamma Q^\pi(s_{t+1})$, ולפי זה ה- A^π advantage function יהיה:

$$A(s_t, a_t) = r_{t+1} + \gamma \mathcal{V}(s_{t+1}) - \mathcal{V}(s_t)$$

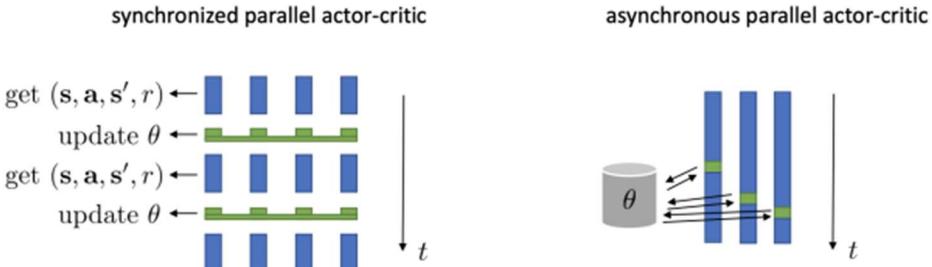
את הביטוי הזה נכניס לגרדיינט ובכך נשערר את האסטרטגיה. נשים לב שבביטוי זה תלוי במצב בלבד ולא בפועל, מה שמקל על הלמידה, כיוון שייתר קל ללמידה את $(s)^\pi$ השיר למרחב בגודל G מאשר את $(s, a)^\pi$ השיר למרחב בגודל $A \times G$. המשמעות הפרקטית של הביטוי היא שמסתכלים על שני מצבים, ובוחנים את הפרש הערכים שלהם ועוד reward. אם הערך חיובי – אז נרצה לחזק את המעבר מהמצב הראשון למצב השני, ואם הערך שלילי, אז נרצה למנוע צזה מעבר.

חסר – שkopiyat 19.

אחד הביעות שיש גם ב- Q וגם ב- C Learning – Q – הוא הקורלציה שיש בין הדגימות – עבור מסלול שנדגם, יש קשר בין המצבים השונים, ולכן השונות עדין גבוהה. לעיל רأינו שבאלגוריתם Learning – Q ניתן להתגבר על

כך בעזרת שמירה של N מצבים ועדכו לפִי k מצבים אקראיים מתוך אלה שנשמרו (למשל: $N = 1000, k = 10$). ניתן ניתן להשתמש בפתרון זה, שהרי הלמידה היא policy-onwards ואפשר להשתמש במצבים מודגימות קודמות, כיוון שהן נדגמו מאסטרטגיה שכרגע היא לא רלוונטי. ניתן בכל אופן להשתמש במצבים שנדרגו מה-epochs והאחרונים, תחת הנחה שהאסטרטגיה לא השתנה הרבה (ובנוסף ניתן לתקן באמצעות *Importance sampling*).

פתרון יותר מוצלח הוא לאמן מספר סוכנים במקביל ולהשתמש במידע שmagiu מכוון, כאשר היתרון הגדול של שיטה זו הוא שהדגםות של הסוכנים הושנים הן חסורת קורלציה. שיטה זו הציגה תוצאות מעולות ובזמן הרובה יותר מאשר המשיטות הקודמות שראינו. יש שתי אפשרויות להריץ במקביל – ריצה סינכרונית (הסוכנים רצים במקביל, ולאחר מכן מסיימים epoch מחשבים את הגרדיינט וمعدכנים האסטרטגיה) וריצה אסינכרונית (כל פעם שסוכן מסיים epoch הוא מחשב גרדיאנט וمعدכן האסטרטגיה). הריצה הסינכרונית יכולה להיות קצרה יותר איטית כיוון שבכל עדכון יש לחכות לכך שכל הסוכנים יסיימו. מצד שני הריצה האסינכרונית יכולה להיות קצרה יותר מידיית כיוון שיתכוון מצבים שלසוכנים שונים תהיה אסטרטגיה מעט שונה (אם הם עדכנו את האסטרטגיה שלהם בזמןים שונים ובאמצעות היה עדכון של הגרדיינט). עם זאת ניתן להניח שההבדלים יחסית לא גדולים, כיוון שהסוכנים מעדכנים את האסטרטגיה שלהם בערך באותו זמן.



איור 11.11 אימון מספר סוכנים במקביל בשיטת Actor-Critic. ניתן לבצע את העדכון בצורה סינכרונית – כאשר כל הסוכנים מסיימים epoch מחשבים את הגרדיינט וمعدכנים את האסטרטגיה, או בצורה אסינכרונית – כל סוכן שמסיים epoch מעדכן את האסטרטגיה בהתאם.

לסיום, שיטת *Actor-Critic* מנשה לשלב עקרונות מתוך \mathcal{Q} – \mathcal{Q} יחד עם *Policy gradient Learning* הבסיסי הוא להחליף את הביטוי של *reward-to-go Policy gradient* $(s/a)(\mathcal{Q})/\mathcal{Q}$, שהוא הרובה פחות רועש. את \mathcal{Q} ניתן לשער במקביל לגרדיינט עצמו, אך יש לעשות זאת באמצעות *policy-on*. אמן השנות יורדת אך עדין יש הטיה, כיוון שיש צורך לשער גם את \mathcal{Q} . בדומה להוספה *Baseline*, גם כאן ניתן להוסף את *policy gradient*.

11.4 Model Based Control

בפרק הקודם דיברנו על אלגוריתמים שלא מתעסקים במודל של הבעיה, אלא מנסים ללמידה ישירות מתוך הניסיון את הערך של כל מצב/מצב-פעולה $(s/a)(\mathcal{Q})/\mathcal{Q}$ או את האסטרטגיה האופטימלית עבור הסוכן. בפרק זה נדבר על אלגוריתמים שמנסים ללמידה את האסטרטגיה מתוך המודל או הדינמיקה של הבעיה (בין אם ידועים ובין אם הם נלמדת).

יש מספר יתרונות ללמידה מתוך המודל ולא מתוך ניסיון בלבד. ראשית, הרובה פעמים הלמידה של מודל יכול להיות שימושית פשוטה יותר מאשר למידה של $(s/a)(\mathcal{Q})/\mathcal{Q}$, מה שהופך את הלמידה להרבה יותר חסכונית. ניקח לדוגמה מבוקש פשוט – הבנה של הדינמיקה שלו היא הרובה יותר פשוטה מאשר למידה של הערך של כל מצב. בנוסף, למודל אפשררת *Self-exploration* – נניח ואנו רוצים לאמן סוכן שמבצע נחיתה אוטונומית, אם הוא לא ידע את הדינמיקה של הסביבה אלא ינסה ללמוד רק על בסיס הניסיון, הוא יהיה חייב לבצע פעולות שליליות (למשל לדרכו אנשים ולעשות תאונות) על מנת לקבל משוב שלילי, וכך ללמידה שיש להימנע מפעולות אלה. אם לעומת זאת אנו יודעים את המודל והסביבה, ניתן לייצר סימולציה ובה לבחון את המצבים השונים וכן לאמן את הסוכן.

עם זאת, ישנו גם חסרונו של למידה של המודל. ראשית, הרובה פעמים יש פערים בין המודל לבין העולם האמיתי. נניח ואייינו סוכן של רכב אוטונומי בסביבה יירטואלית, המ עבר לעולם האמיתי איינו טריוייאלי כי הסימולציה פשוטה לא מספיק דומה לעולם האמיתי (למשל – רגולציה תמורה הקולט של הסוכן בסימולציה הרובה יותר גבוהה מזו של מצלמה אמיתית המותקנת על הרכב). בנוסף, למודל מסויפה מקור נוסף של שגיאה – נניח ומנסים לשער מודל ועל בסיס המודל רוצים ללמידה אסטרטגיה, אך יש פה שני שלבים של למידה וכל אחד מהם יכול לגרום לשגיאה.

11.4.1 Known Model – Dyna algorithm

נתחיל מהמקרה הייתר פשוט, בו המודל ידוע לנו מנסים באמצעותו ללמידה אופטימלית. בפרק המבוא ראיינו כיצד ניתן להשתמש ב-*Policy iteration* עבור MDP Tabular בעלת מרחב מצבים לא גדול על מנת ללמידה את האסטרטגיה האופטימלית, ובפרק זה נרצה להתמקד במרקטים בהם פתרון זה לא מספיק טוב.

הרעין הפשטוט ביותר בו ניתן להשתמש הוא לייצר סימולציה, כלומר ללקחת את המודל הידוע, ולאמן בעזרתה סוקן כאלו שהוא נמצא בעולם האמיתי. למידה של האסטרטגיה או של (s, Q) באופן הזה לחוב תהיה זולה משמעותית מאשר למידה שלהם ללא עזרת המודל, כיון שנטען הרבה הרכה פחות איטרציות והשימוש במודל הידוע מסיע להכזין את הלמידה. עם זאת, הבעיה היא שיש פער בין הסימולציה לבין העולם האמיתי, מה שיוצר אתגר כפוף: א. הסימולציה לא מגיעה לרמת דיק מספיק טוביה בתיאור העולם האמיתי ולכן הלמידה לא מספיק אינטואטיבית. ב. נניח ויש לנו אפשרות ללמוד על העולם האמיתי, אך גם אם יש שגיאה – היא לחוב תהיה קטנה מאוד. בסימולציה לעומת זאת, שגיאה כזו יכולה להציג וולגרום לסוקן לשגיאות גדולות.

בכדי להתגבר על אתגרים אלו ניתן **לשלב** בין Model free ו-Model Based, וכעת נציג אלגוריתם בשם **Dyna** שעושה לבדוק את השילוב הזה. אלגוריתם זה משלב ידע משני מקורות – DATA אמתי וDATA המגיע מסימולציה, כאשר הוא פועל באופן איטרטיבי לפי השלבים הבאים:

- ראשית מקבלים DATA אמתי, ומשפרים באמצעותו **שני** דברים – מודל שעליו מושתת הסימולציה וסוקן שמאומן בכלים של model free.
- לאחר מכן מייצרים DATA מהסימולציה, ומשתמשים בו בשבייל לשפר את הסוקן.

מבצעים את השלבים האלה שוב ושוב וככה משפרים במקביל גם את הסוקן (שלומד על בסיס DATA מהסימולציה). באופן הזה אנחנו משפרים את הסימולציה כל הזמן, ועוד גם אם יש בעיה בסימולציה והוא לא זהה בבדיקה למודל של העולם האמיתי, אנחנו לא מתעלמים ממנו אלא מנסים לתקן אותה. בוגיגוד לשימולציה פשוטה שלא משתמשת בDATA מהעולם האמיתי אלא רק מtaboo בסיס המודל הידוע, אכן אנו לא זונחים את DATA האמיתי אלא עושים לו אוגמנטציות בעזרת הסימולציה (שבעצמה גם הולכת ומשתפרת). נניח מקרה פשוט של מודל דטרמיניסטי ובחירה של אלגוריתם Learning – Q לאימון הסוקן, האלגוריתם יפעל לפי לשלבים הבאים (כמפורט שניתן להרחב בזורה טריויאלית לכל מודל ולכל אלגוריתם Model-Free):

```

Initialize  $Q(s, a)$  and  $Model(s, a)$  for all  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$ 
Do forever:
    (a)  $S \leftarrow$  current (nonterminal) state
    (b)  $A \leftarrow \varepsilon\text{-greedy}(S, Q)$ 
    (c) Execute action  $A$ ; observe resultant reward,  $R$ , and state,  $S'$ 
    (d)  $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$ 
    (e)  $Model(S, A) \leftarrow R, S'$  (assuming deterministic environment)
    (f) Repeat  $n$  times:
         $S \leftarrow$  random previously observed state
         $A \leftarrow$  random action previously taken in  $S$ 
         $R, S' \leftarrow Model(S, A)$ 
         $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$ 

```

איור 11.12 אלגוריתם Dyna. שילוב של מידע מסימולציה המבוססת על מודל ידוע יחד עם סוקן שלומד בעזרת אלגוריתם Model-Free.

מלבד העובדה שתוך כדי הלמידה אנו משפרים את המודל, גם DATA שמיוצר הוא יותר רלוונטי ובעל ערך ללמידה – לא סתם מוגלים מצבים אקראיים, אלא המצבים שהסוקן רואה קשורים לאסטרטגיה שנלמדה על בסיס DATA מהעולם האמיתי. באופן הזה יש שימוש כפוף ו"ממצאה" יותר של DATA מהעולם האמיתי – הוא גם מסיע לבנות את המודל וגם מסיע לסוקן להשתפר.

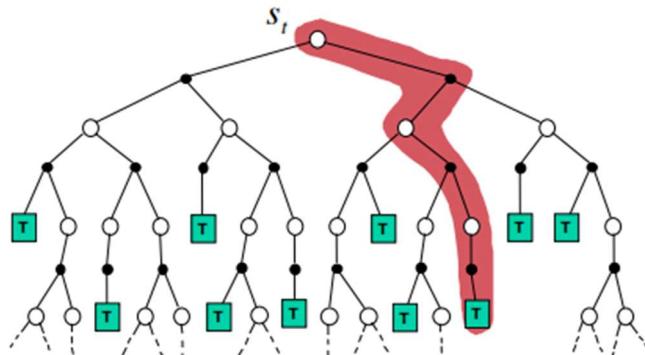
గרפ' דוגמה – שקיופיות 12,11. שני מודל למודל לא נכון – שקיופית 13 (מודל קבוע לא היה מצליח להשתפר לאחר השינוי).

11.4.2 Known Model – Tree Search

כאמור, אלגוריתם Dyna נעזר בידיעת המודל על מנת לאמן סוקן בעזרת אלגוריתם של Model-Free. יוצא מכך שאמנם הסוקן איינו תלוי במודל, אך עדין האסטרטגיה אותה אנו מנסים ללמידה היא גלובלית, כלומר מנסה לדעת מה לעשות בכל סיטואציה אפשרית. עבור משימות מאוד מורכבות, כמו למשל משחק שחמט, זה עדין לא מספיק, כיון שאסטרטגיה כללית עבור כל המצבים אינה אפשרית ללמידה עקב רבוי המצבים. בכדי להתמודד עם בעיות מורכבות

ניתן להפוך את הפתרון להיות **локאלי**, כלומר להסתכל על המצב הנוכחי בלבד ולנסות ללמוד בעזרת הסימולציות מה הפעולה שתביא את התוצאה הכי טובה.

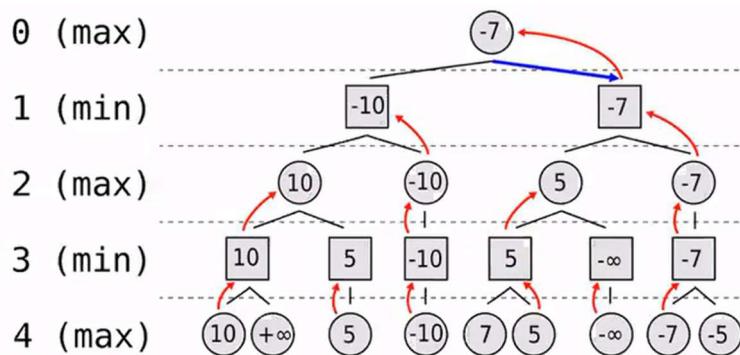
הweeney של הלמידה הлокאלית – חיפוש הפעולה הכי טובה עבור מצב נתון – קשור לתחום רחב שנקרא Forward Search (ງער שבעה זו קשורה להרבה תחומים ולא רק ל-RL, ופה נדון בעיקר במה שקשר לחיפוש באמצעות אלגוריתם של RL). תחום זה בא להתמודד עם מצבים בהם הבעיה הכללית היא מאוד מורכבת, אך אנו מנסים לפתור תת-בעיה המסתכלת רק על המצב הנוכחי ומה שמסתעף ממנו. ניתן לדמות את הסביבה לעץ בו כל צומת הוא מצב והוא מתפצל למצבים נוספים לפי הפעולות השונות האפשריות באותו מצב. בעזרת הסימולציה נבדוק כל מיני הסתעויות שונות מאותו מצב, ונוכל לבחון מי מבינה הטובה ביותר.



איור 11.13. חיפוש הפעולה הכי טובה ביחס למצב נתון, כל צומת בעץ מציג מצב, והסתעויות מתפתחות בהתאם לפועלות האפשריות בכל מצב, כאשר מצב ט�ימי (terminal state) מיוצג על ידי T. סימולטור המבוסס על מודל ידוע יכול לעוזר לבחון את הסתעויות השונות.

כאשר חלק מהבעיה יש גם ירייב שאסטרטגיה שלו אינה ידועה, יש כמה דרכיים איך להתייחס להתנחות שלו, כאשר הנפוצות שביןן הן **לייחס את אותה אסטרטגיה כמו של הסוכן** (Self-play) או **לייחס אליו את האסטרטגיה האופטימלית** (Minimax), כאשר יש להגדיר במדויק מהי אותה אסטרטגיה אופטימלית. אם נניח שמדובר במצבים של משחק סכום-אפס (בهم התגמול של שני המשתתפים הוא קבוע), אסטרטגיה אופטימלית מניחה שהיריב ינקוט בפעולות שימקסמו את התגמול שלו וימזערו עד כמה שניתן את התגמול של הסוכן. במקרה זה, הסוכן ירצה לבחור בפעולה שתמקסם את התגמול שלו ב-worst case, כלומר ביריב בחר את הפעולות הטובות ביותר עבורו.

הבעיה שעולה מיד היא שבפועל אי אפשר להתחיל במצב מסוים ולבודק את כל הסתעויות עבור כל עומק אפשרי של העץ. לכן יש להגדיר עומק חיפוש מקסימלי, ואז כל מצב שהגענו אליו שנמצא באותו עומק יקבל ערך לפי נוסחה מסוימת מראש (למשל – בשחמט ניתן לחתם במצב לפי שווים של כל הסוכן הנמצאים באותו רגע על הלוח פחות שווים של כל היריב). נתבונן על דוגמה של עץ Minimax בעומק 4 בצדדי להבהיר את המצב אותו אנו מנסים ללמוד:



איור 11.14 עץ Minimax בעומק 4. העיגולים מסמנים את המצבים בהם תור היריב, והריבועים מסמנים את המצבים בהם תור הסוכן. החיצים האדומים מסמנים את הפעולות האופטימליות עבור היריב, ואנו מניחים שהוא יבחר בהן.

הסוכן מתחילה במצב בשורה 0, ובעזרת סימולציה בודק את כל הסתעויות עד עומק 4. בשלב ראשון הסוכן משער באמצעות הסימולציה רק את הערכיהם של הצמתים **בעומק המקסימלי**. אם זה מצב טרמינלי (כלומר המשחק הסטיים), הערך בעומק זה יהיה הערך שהתקבל בסימולציה, ואם המצב אינו טרמינלי אז הערך נקבע לפי היריסטייה קבועה מראש. לאחר מכן הסוכן הולך אחריה וממשקל את כל הצמתים בעומק הקודם תחת הנחת ה-worst case – ככלומר מניחים שהוא יבחר את הפעולות הגראעה עבורו הסוכן (מסומן בחיצים האדומים). ככה למשל הצמת

השמאלי בעומק 3 קיבל את הערך 10, כי אנו מניחים שבצומת זה הסוקן יבחר ללכת שמאלה. בדומה, הצומת הימני בעומק 3 קיבל את הערך 7 – כי אנו מניחים שהוא יבחר ללכת שמאלה באוטו צומת. כך נתונים ערכים לכל השורה הזאת, כאשר הכלל הוא שכל צומת מקבל את הערך **המינימלי** מבין כל אפשרות הפיזול שלו. כתע עוברים לתת ערכים לשורה בעומק 2, כאשר כאן הכלל הוא הפק – כיון שהטור הוא של הסוקן, נניח שהוא יבחר בפעולה הכי טובה, ולכן כל צומת מקבל את הערך המינימלי מבין אפשרות האפשרים שלו. למשל הצומת השמאלי בעומק 2 קיבל את הערך 10, כיון שהוא הערך של ההתפצלות הכי טובה שהסוקן יכול לבחור. באופן זה עולים למעלה וממלאים את כל ערכי העץ עד השורש, עד שמקבלים ערך עבור המצב הנוכחי בו אנו נמצאים.

כמה העורות חשובות:

1. תacen והיריב יעשה טעויות ולא יבחר את הצעדים הכי טובים עבורי, אז בפועל נקבל ערך אחר למצב בו אנו נמצאים, אבל השיטה הזהירה ביותר היא להניח שהיריב יהיה אופטימי, ועל בסיס זה לחת ערכים לצמתים ולמצב הנוכחי. לאחר שביצענו את הפעולה הכי טובה בהתאם להערכת הנוכחית, היריב בתורו בוחר צעד ומתיקל מצב חדש, ואז הסוקן שוב יבצע סימולציות על מנת לחשב את כל ההסתעפויות מאותה נקודה ולחת ערך למצב החדש שהתקבל.
2. כיון שהעץ גדל אקספוננציאלית, העומקים אותם ניתן לחשב הם לא גדולים, ולכן יש חשיבות להירוטטיקה שקובעת את הערכים עבור הצמתים שבעומק המקסימלי שלא קיבלו ערך בסימולציה עצמה (כלומר מצבים שאיןם טרמינליים).
3. במשחקים בהם יש גם גורם של מזל (כמו לمثال הטלת קוביות), ניתן להוסיף שכבת של chance nodes בין כל שתי שורות בעז, כאשר החישוב ביחס לשכבות החדשנות הינו בעזרה התוחלת של התוצאה שלהן. עץ זה נקרא **.Expectiminimax Tree**

אלגוריתם חיפוש זה עובד טוב עבור מגוון בעיות, אך הוא עדין נתקל בבעיות עבור אתגרים בעלי מרחב מצבים גדול מאוד, כמו למשל המשחק Go. במקרים אלה מספר הפיזולים גדול מדי בשביל שהיה בר חישוב, וממילא לא ניתן לבחון את כל ההסתעפויות עבור עומק סטנדרטי, מה שלא מאפשר לחשב בצורה איקוית את ערכי הצמתים. לכן, במקומות לבדוק את כל ההסתעפויות, ניתן להיעזר בסימולציה ולבוחן רק חלק מהן, ועבור כל סימולציה נפרק עד לעומק המקסימלי כדי שההתוצאה תהיה אמיטית ולא תלויות הירוטטיקה. כמובן שבחירת ההסתעפויות לא נעשית באופן אקראי, אלא היא מבוססת על מדיניות נלמדת של שני היריבים. באופן זה בהתחלה הבעיה היא קלה כיון שליריב אין אסטרטגיה טובה, אך ככל שאנו משתפר כך גם היריב משתפר והבעיה הולכת ונעשה קשה, וממילא ניתן להשתפר עוד ועוד. אלגוריתם חיפוש זה נקרא **Monte-Carlo Search**, וניתן לנסה אותו באופן פורמלי כך:

נניח ונתונה אסטרטגיה ההתחלתית π שאינה מספיק טובה. נתנו מצב s_t , ורוצים לדעת מה הערך של כל פעולה a אפשרית. נדגום K מסלולים עבור כל פעולה a אפשרית, ניעזר בסימולציה כדי לבצע את המסלול עד הסוף, וnochesh את תוחלת הרוחות של כל הסימולציות. באופן זה, עבור המצב s_t הערך של פעולה a יהיה:

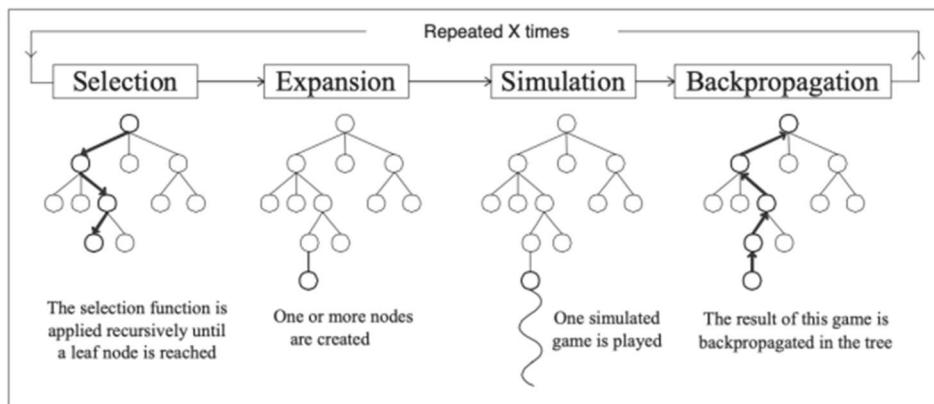
$$Q^\pi(s_t, a) = \frac{1}{K} \sum_k \sum_i \gamma^i R_{t+i+1}^k$$

לאחר שיחסבנו את הערך של כל הפעולות, נבחר את הפעולה בעלת הערך הכי גבוה:

$$\max_a Q^\pi(s_t, a)$$

יש להעיר שבדי שאלוגריהם יעבד, צריך שהאסטרטגיה ההתחלתית π תהיה ברמה סבירה, כיון שם היא גראעה, השיפור יהיה מאד איטי ולא מספיק.

ניתן לחקת את הרעיון הזה ולהרחיב אותו לחיפוש על עץ כפי שתיארנו קודם. כאמור לעיל, האתגר המרכזי בחיפוש על עץ נעוץ בכך שהוא גדל אקספוננציאלית ולא ניתן לחשב באופן יעיל בעיות עם מספר מצבים גדול מאוד. בצד' להתגבר על כך ניתן להשתמש באלגוריתם (MCTS) **Monte-Carlo Tree Search**, שהוכיח את עצמו כמצוין חזק ואמיל בעיות גדולות מאוד. אלגוריתם זה למעשה משלב בין שני חלקים שככל אחד מהם בפני עצמו לא מספיק טוב – מצד אחד יש לנו אסטרטגיה ההתחלתית, אך היאBINONOTY ולא מספיק איקוית, ומצד שני יש לנו סימולציה המבוססת על ידיעת המודל, אך היא אינה יכולה לבדוק את כל האפשרויות. הרעיון של MCTS הוא לחקת אסטרטגיהBINONOTY ולשפר אותה באמצעות בדיקה מושכלת של העתיד – הרכהה של העץ לא נעשית באמצעות בחינת כל האפשרויות, אלא באמצעות בחירה של כמה אפשרויות שנראות סבירות ביחס לאסטרטגיה הנוכחיית. האלגוריתם ניתן לתיאור באמצעות ארבעה שלבים, שוחזרים על עצם באופן איטרטיבי a פעמים עבור כל פעולה a :



איור 11.14 אלגוריתם Monte-Carlo Tree Search: 1. בחירת הצומת הבא אותו רוצים לבחון. 2. הרחבת העץ מאותו צומת. 3. הרצת סימולציה מאותו צומת עד הסוף (=הגעה למשחק טרמינלי). 4. נתינת ערך לכל צומת מהצומת הנבחר ואחוריה עד השורש.

כאמור, האלגוריתם מורכב מארבעה שלבים:

1. בחירה (Selection) – ראשית יש לבחור את המצביע s עבורו אנו רוצים לבדוק מה הפעולה הכי טובה במשחק זה. לשם כך נבצע סדרה של צעדים עד שנגיע למצביע s_t , כאשר את הצעדים ניתן לבחור על בסיס score שלהם ואפשר גם לשלב רנדומליות עבור exploration כפי שנראה בהמשך.
2. הרחבה (Expansion) – מושגנו לאוטו מצב s_t ורחיב אותו בצוות נסוף על ידי בחירה של אחת מהפעולות האפשריות. את הפעולה ניתן לבחור באופן רנדומלי או על בסיס האסטרטגיה הנתונה (שכאמרור היא רוחקה מאופטימליות).
3. הרצת סימולציה (Simulation) – כעת נרייצ' סימולציה מהמצביע החדש עד הסוף ונבדוק כיצד הסטיים המשחק בהתאם לתוצאות המשחק ניתן לאוטו מצב חדש ערך.
4. נתינת ערכאים לצומתים (Backpropagation) – נחלחן אחריה את הערך שהתקבל ונעדכן את כל הצומתים שנבחנו. אם למשל המשחק הסטיים בניצחון, אז נעדכן את ערך הצומת החדש (שהרchieב את העץ) ל-1/1 (=ניצחון אחד מתוך משחק אחד). בהתאם נעדכן את כל יתר המצביעים שעברנו בהם – אם למשל השורש מהם יצאנו היה בעל יחס של 3/5, אז נעדכן אותו ל-6/4.