

Contents

1. Introduction	6
1.1 What is Machine Learning?	6
1.1.1 The Basic Concept.....	6
1.1.2 Data, Tasks and Learning.....	7
1.2 Applied Math	8
1.2.1 Linear Algebra.....	8
1.2.2 Calculus.....	13
1.2.3 Probability	15
1. References.....	25
2. Machine Learning.....	26
2.1 Supervised Learning Algorithms.....	26
2.1.1 Support Vector Machines (SVM).....	26
2.1.2 Naive Bayes	29
2.1.3 K-Nearest Neighbors (K-NN)	31
2.1.4 Quadratic\Linear Discriminant Analysis (QDA\LDA)	33
2.1.5 Decision Trees.....	35
2.2 Unsupervised Learning Algorithms.....	49
2.2.1 K-means.....	49
2.2.2 Mixture Models.....	50
2.2.3 Expectation–maximization (EM)	52
2.2.4 Hierarchical Clustering.....	54
2.2.5 Local Outlier Factor (LOF).....	55
2.3 Dimensionally Reduction.....	57
2.3.1 Principal Components Analysis (PCA)	58
2.3.2 t-distributed Stochastic Neighbors Embedding (t-SNE).....	62
2.4 Ensemble Learning.....	65
2.4.1 Introduction to Ensemble Learning.....	65
2.4.2 Bootstrap aggregating (Bagging).....	65
2.4.3 Boosting.....	67
2. References.....	70
3. Linear Neural Networks	71
3.1 Linear Regression	71
3.1.1 The Basic Concept.....	71
3.1.2 Gradient Descent	73

3.1.3 Regularization and Cross Validation.....	73
3.1.4 Linear Regression as Classifier	74
3.2 Softmax Regression.....	76
3.2.1 Logistic Regression.....	76
3.2.2 Cross Entropy and Gradient descent.....	77
3.2.3 Optimization	78
3.2.4 SoftMax Regression – Multi Class Logistic Regression	79
3.2.5 SoftMax Regression as Neural Network.....	80
3. References.....	81
4. Deep Neural Networks	82
4.1 Multilayer Perceptron (MLP).....	82
4.1.1 From a Single Neuron to Deep Neural Network.....	82
4.1.2 Activation Function	83
4.1.3 Xor	85
4.2 Computational Graphs and propagation.....	86
4.2.1 Computational Graphs.....	86
4.2.2 Forward and Backward propagation.....	86
4.3 Optimization	88
4.3.1 Data Normalization	88
4.3.2 Weight Initialization	88
4.3.3 Batch Normalization.....	89
4.3.4 Mini Batch	89
4.3.5 Gradient Descent Optimization Algorithms	90
4.4 Generalization.....	92
4.4.1 Regularization.....	92
4.4.2 Weight Decay	93
4.4.3 Model Ensembles and Drop Out.....	93
4.4.4 Data Augmentation.....	94
4. References.....	95
5. Convolutional Neural Networks (CNNs).....	96
5.1 Convolutional Layers.....	96
5.1.1 From Fully-Connected Layers to Convolutions	96
5.1.2 Padding, Stride and Dilatation.....	97
5.1.3 Pooling.....	98
5.1.4 Training.....	99
5.1.5 Convolutional Neural Networks (LeNet).....	99

5.2 CNN Architectures	100
5.2.1 AlexNet.....	100
5.2.2 VGG	101
5.2.3 GoogleNet.....	101
5.2.4 Residual Networks (ResNet).....	102
5.2.5 Densely Connected Networks (DenseNet).....	103
5.2.6 U-Net.....	103
5.2.7 Transfer Learning	104
5. References.....	105
6. Recurrent Neural Networks	106
6.1 Sequence Models	106
6.1.1 Vanilla Recurrent Neural Networks	106
6.1.2 Learning Parameters.....	107
6.2 RNN Architectures.....	108
6.2.1 Long Short-Term Memory (LSTM)	108
6.2.2 Gated Recurrent Units (GRU).....	109
6.2.3 Deep RNN	110
6.2.4 Bidirectional RNN	111
6.2.5 Sequence to Sequence Learning.....	111
6. References.....	111
7. Deep Generative Models.....	113
7.1 Variational AutoEncoder (VAE)	113
7.1.1 Dimensionality Reduction	113
7.1.2 Autoencoders (AE)	114
7.1.3 Variational AutoEncoders (VAE)	115
7.2 Generative Adversarial Networks (GANs)	118
7.2.1 Generator and Discriminator	118
7.2.2 Deep Convolutional GAN (DCGAN).....	121
7.2.3 Conditional GAN (cGAN)	121
7.2.4 Pix2Pix	121
7.2.5 CycleGAN	122
7.2.6 Progressively Growing GAN (ProGAN).....	122
7.2.7 StyleGAN	123
7.2.8 Wasserstein GAN	125
7.3 Auto-Regressive Generative Models	128
7.3.1 PixelRNN	129

7.3.2 PixelCNN	130
7.3.3 Gated PixelCNN.....	130
7.3.4 PixelCnn++	130
7. References.....	132
8. Attention Mechanism.....	133
8.1 Sequence to Sequence Learning and Attention.....	133
8.1.1 Attention in Seq2Seq Models	133
8.1.2 Bahdanau Attention and Luong Attention	133
8.2 Transformer.....	134
8.2.1 Positional Encoding.....	134
8.2.2 Self-Attention Layer.....	135
8.2.3 Multi Head Attention.....	137
8.2.4 Transformer End to End.....	137
8.2.5 Transformer Applications.....	138
9. Computer Vision.....	141
9.1 Object Detection.....	141
9.1.2 You Only Look Once (YOLO).....	141
9.1.4 Spatial Pyramid Pooling (SPP-net)	144
9.2 Segmentation.....	146
9.2.1 Semantic Segmentation vs. Instance Segmentation.....	146
9.2.2 SegNet neural network.....	146
9.2.3 Atrous Convolutions (Dilated Convolutions)	148
9.3 Face Recognition and Pose Estimation.....	149
9.3.1 Face Recognition.....	149
9.3.2 Pose Estimation.....	151
9.4 Few-Shot Learning.....	153
9.4.1 The Problem.....	153
9.4.2 Metric Learning	153
9.4.3 Meta-Learning (Learning-to-Learn).....	155
9.4.4 Data Augmentation.....	156
9. References.....	157
10. Natural Language Processing.....	158
10.1 Language Models and Word Representation.....	158
10.1.1 Basic Language Models.....	159
10.1.2 Word representation (Vectors) and Word Embeddings	161
10.1.3 Contextual Embeddings	165

10. References.....	170
11. Reinforcement Learning (RL).....	171
11.1 Introduction to RL.....	171
11.1.1 Markov Decision Process (MDP) and RL	171
11.1.2 Bellman Equation	173
11.1.3 Learning Algorithms.....	178

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 חישיבה אנושי. בתחום רחב זה יש רמות שונות של בינה מלאכותית – יש מערכות שמסГОלות ללמידה דפוסי התנהגות ולהתאים את עצמן לשינויים, ואילו יש מערכות שאמנים מנגנון חישיבה אנושי אך הן לא מתוחכמת מעבר لما שתכננו אותן בהתחלה. שואב רובוטי הידע לחשב את גודל החדר ואת מסלול הנקוי האופטימלי פועל לפי פרוצדורה ידועה מראש, ואין בו תחכים מעבר לתוכנות הראשוני שלו. לעומת זאת תוכנה הידעית לסון רושים באופן מסתגל, או להמליץ על שירים בגין מזיקה בהתאם לسانון של המשתמש, משתמשת לבינה מלאכותית ברמה גבוהה יותר, כיוון שהן לומדות עם הזמן דברים חדשים.

המונה בינה מלאכותית מתייחס בדרך כלל למערכת שמקהה התנהגות אנושית, אך היא שגרתית, לא לומדת חדשה, ועשויה את אותו הדבר כל הזמן. מערכת זו יכולה להיות משוכלת ולהשכיל דברים מסוימים ואף להסיק מסקנות על דוגמאות חדשות שהיא מעולם לא ראתה, אך תמיד עברו אותה הפלט (Output). היה אוטו הפלט (Output).

נראה לדוגמא מערכת סטרימינג של סרטים, למשל Netflix. חלק משיפור המערכת והגדלת זמני הצפייה, ניתן לבנות מנגנון המלצות הבניי על היסטוריית השימוש של לקוחות של המערכת –izia סרטים הם אוהבים,izia ז'אנרים ומות'. כישיש מעט צופים ומעט סרטים, ניתן לעשות זאת באופן יידי – 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 "הגשרים של מחוז מדיסון".

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

נתובן על דוגמא נוספת – מערכת לניטוט רכב. ניתן להגדיר כלל פשוט בו אם משתמש יצא מטל אביב ורוצה להגיע לפתח תקווה, אז האפליקציה תיקח אותו דרך מסלול ספציפי שנבחר מראש. מסלול זה לא מתחשב בpermטרים קרייטיים כמו מה השעה, האם יש פקקים או חסימות ועוד. כמות הpermטרים שיש להתייחס אליהם איננה ניתנת לטיפול על ידי מערכת כללים ידועה מראש, וגם הפונקציונליות המתאפשרה היא מוגבלת מאוד – למשל לא ניתן לחזות מה תהיה שעת ההגעה וכדומה.

Machine Learning (ML)

למידת מכונה הוא תחום בתחום של בינה מלאכותית, הבא להתמודד עם שני האתגרים שתוארו קודם – יכולת לתוכנת מערכות על בסיס מסוות של נתונים וpermטרים, וחיזוי דברים חדשים כתלות בpermטרים רבים שיכולים להשנותם עם הזמן. מנגנוני ML מנהלים כמהיות אידיות של DATA ומנסחות לחשב את מושך הנסעה המשוער. נניח והיא חצתה 20 דקות נסעה. המערכת תנתח את כל אותם הפקטורים ותנסה לחשב את משך הנסעה המשוער. אם מדובר באפליקציית ניוט, אם בסופו של דבר הנסעה ארוכה 30 דקות, האלגוריתם ינסה להבין פקטור השתנה במהלך הדרך ומדווח הווא נכשל בחיזוי (למשל: הcabesh באנרבעה נתיבים, אבל במקטע מסוים הוא מוצטמצם לאחד וזה מייצר עיכוב, וזה עיכוב קבוע ברוב שעותם הימה ולא פקק אקראי). בהינתן מספיק מקרים כאלה, האלגוריתם "מבנה" שהוא טועה, והוא פשוט יתקן את עצמו ויכניס למערך החישובים גם פקטור של מספר נתיבים ויריד אויל את המשקל של הtempoורה בחוץ. וככה באופן חזרתי האלגוריתם שוב ושוב מקבל קלט, מוציא פלט ובודק את התוצאה הסופית. לאחר מכן הוא בודק היכן הוא טעה, משנה את עצמו, מתקן את המשקל שהוא נותן לפקטורים שונים ומשתכל מנסעה לנסעה.

במערכות אלה הקלט נשאר לכורה קבוע, אבל הפלט משתנה – עבור זמן יציאה שונים, האלגוריתם יעריך זמן נסעה שונים, כתלות במוגן הpermטרים הרלוונטיים.

מערכות ML משמשות את כל רשות הpermטום הגדוילות. כל אחת מנסה בדרכה שלא לחזות למשל,izia משתמש שהקליק על המודעה צפי שיבצע רכישה. הפלטpermוטות השונות מנסחות לזיהות כוונה (Intent) על ידי למידה מניסין. בהתחלה אין פשוט ניחשו על פי כמה פקטורים שהזינו להם על ידי בני אדם. נניח, גוגל החלטהשמי צופה

בஸרטוני יוטיוב של **Intent** הוא ב-**Intent** גבויה של רכישה. בהמשך הדריך, בהנחה והמשתמש מבצע רכישה כלשהו, האלגוריתם מקבל "נקודה טוביה". אם הוא לא קנה, האלגוריתם מקבל "נקודה רעה". ככל שהוא מקבל יותר נקודות טובות ורעות, האלגוריתם יודע לשפר את עצמו, לתת משקל גדול יותר לפרמטרים טובים ולהזניח פרמטרים פחות משמעותיים. אבל רגע, מי אמר למערכת להסתכל בכלל בסרטוני **Intent**?

האמת שהיא שאף אחד. מישחו, בנאדים, אמר למערכת להזיהות את כל הסרטונים שימוש צופה בהם ביזיובי, להזיהות מתוך הסרטון, האודיו, תיאור הסרטון ומילוי המפתח וכו' – איזה סוג סרטון זה. ייתכן שאחרי מיליארדי צפויות הסרטונים, האלגוריתם מתחליל למצוא קשר בין סוג מסוים של סרטונים לבין פעולות כמו רכישה באתר. באופן היזה, גוגל מזינה את האלגוריתם בכל הפעולות שהמשתמש מבצע. המידע שהוא קורא, המיקומות שהוא מסתובב בהם, התמונות שהוא מעלה לען, ההודעות שהוא שולח, כל מידע שיש אליו גישה. הכל נשפר לטור מאגר הנתונים העצום בו מוסה גוגל לבנות פרופילים ולמצוא קשר בין הסיסמי שלו לרכוש או כל פעולה אחרת שהוא לה זיהות.

המכונה המופלאה זו לומדת כל הזמן דברים חדשים ומנסה כל הזמן למצוא הקשרים, לחזות 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 בייבי שארק, אז היא תשתחם בו גם אם זה לא נשמע הגיוני.

חשוב לשים לב לעניין המטרה – המטרה היא לא המצאה של האלגוריתם. הוא לא קם בבודק 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 – משתמש יבצע רכישה, והכל מתנקז לזה בסוף, כי גוגל בראש ובראשונה היא מערכת פרטום. אגב, גם ההגדרה של מסלול "אופטימי" היא מעשה ידי אדם. המכונה לא יודעת מה זה אופטימי, זו רק מילה. אז צריך לעזור לה ולהגיד לה שאופטימי זה מינימום זמן, מעט עצירות, כמה שפחות רמזורים וכו'. לסיכום, המטרה מאופיינת על ידי האדם ולא על ידי המכונה. המכונה רק חותרת למטרה שהוגדר לה.

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

כך או כך, המערכת הסובך זהה הקרי **ML** בני מנגנונים שונים המiomנים בניתוח טקסט, אלגוריתמים אחרים המתמקדים בעיבוד אודיו, כללה המנתחים היסטוריים גלישה או זיהוי מותק דף **Web** בו אתם צופים ועוד. عشرות מאות מנגנונים כאלה משתמשים ורצים ובונים את המפה השלמה. ככה רוב רשות הפרסום הגדלות עובדות. ככל שהמכונה של גוגל/פייסבוק תהיה חכמה יותר, ככה היא תדע להציג את המודעה המתאימה למשתמש הנוכחי, בזמן הנוכחי ועל **Device** המתאים.

1.1.2 Data, Tasks and Learning

כאמור, המטרה הבסיסית של למידת מכונה היא יכולת להכליל מהתוך הניסיון, ולבצע משימות באופן מדויק ככל הניתן על דата חדש שעדיין לא נצפה, על בסיס צבירת ניסיון מדאות קיימים. באופן כללי ניתן לדבר על שלושה סוגים של למידה:

למידה מונחתית (**supervised learning**) – הדטה הקיימינו אוסף של דוגמאות, וכל דוגמא יש תווית (**label**). מטרת האלגוריתמים במרקחה זה היא לסואג דוגמאות חדשות שלא נצפו בתהller הלמידה. באופן פורמלי,
 $\text{labels} \in \mathbb{R}^{n \times d}$, יש אוסף y , ומ Chapman את האלגוריתם שמבצע את המיפוי $Y \rightarrow X: g$ בצורה הטובה ביותר, כלומר בהינתן דוגמא חדשה $x \in \mathbb{R}^n$, המכונה היא למצוא עבורה את ה- y הנוכחי. המיפוי נמדד ביחס לפונקציות מחיר, כפי שיסביר בהמשך בוגר לתהller הלמידה.

למידה לא מונחתית (**unsupervised learning**) – הדטה הקיימינו אוסף של דוגמאות במרחב, בלי שננתן עליון מידע כלשהו המבחן בינהן. במרקחה זה, בדרך כלל האלגוריתמים יחפשו מודל המסביר את התפלגות הנקודות – למשל חלוקה לקבוצות שונות וצדומה.

למידה באמצעות חיזוקים (**reinforcement learning**) – הדטה בו נעזרים אינו מצוי בתחום התוכנית אלא נאוסף עם הזמן. ישנו סוכנים הנמצאים בסביבה מסוימת ומעבירים מידע למשתמש, והוא בתורו למד אסטרטגייה בה הסוכנים ינקטו בפעולות הטובים עבורם.

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

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

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 בעיקר 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:

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

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

דוגמאות:

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

חיבור וקטוריים:

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

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 לעיל).

שים לב: ניתן לחסר וליחס 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 וקטור שורה, נותן וקטור עמודה ולהפך.

- **מטריצת יחידה:**

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

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

ובאופן מפורש:

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

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 $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathbb{I}_2$

- **מטריצת 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.

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

$$\text{קטgor} \vec{b}, \text{ הינו וקטgor عمודה, שאייבורו הם האיבר החופשי.}$$

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

מכפלה פנימית, נורמה, אורטוגונליות

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

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

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

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

A. מכפלה סקלרית על מרחב הווקטורים הגיאומטריים:

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

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

כאשר \vec{u} הינו הצמוד המרוכב של 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:

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

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

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

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

דוגמה:

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

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

אורותוגונליות

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

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

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

אורותוגונליות היא הכללה של תכונת הניצבות המוכרת מגאומטריה. בגאומטריה, שני ישרים במשור האוקלידי ניצבים זה זה אם 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 אי-שוויון קושי שוורץ מתקיים: $\| y \| \cdot \| x \| \leq \langle y, x \rangle$, ולכן ימ"נ תמיד קטן או שווה בערכו המוחלט ל-1. כיוון שכן, תמיד ניתן לחשב זוית בין שני וקטורים באמצעות מכפלה פנימית.

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

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

- אם $\mathbf{v} \perp \mathbf{u}$, אז $\mathbf{u} \perp \mathbf{v}$.
- אם $\mathbf{v} \perp \mathbf{u}$, אז לכל סקלר λ גם $\mathbf{u} \perp \lambda\mathbf{v}$.
- אם $\mathbf{v} \perp \mathbf{u}$ וגם $\mathbf{v} \perp \mathbf{w}$, אז $\mathbf{u} \perp (\mathbf{v} + \mathbf{w})$.
- 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 $\mathbf{v} \in \mathbb{F}^n$ נקרא ערך עצמי של A אם $\mathbf{v} \neq 0$ ונראה הווקטור העצמי המתאים אם מתקיים:

$$A \cdot \mathbf{v} = \lambda \cdot \mathbf{v}$$

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

אם נסמן $[\mathbf{v}_1, \dots, \mathbf{v}_n] = V$, אז מתקיים:

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

כאשר (Λ) הוא ערכי האלכסון של המטריצה A .

פירוק לערכים סינגולריים

ניתן לפרק מטריצה $M \in \mathbb{R}^{m \times n}$ למינימלית 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, $-I^n \Sigma I^n \in \mathbb{C}^{n \times n}$ היא מטריצה אוניטרית מרוכבת (או ממשית). פירוק זה נקרא פירוק לערכים סינגולריים (Singular value decomposition - SVD).

ערכים האלכסון של Σ – 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 $\{0,1\}$). בדרך כלל הסימון מופיע כר: $Y \rightarrow X$, כאשר X ו- Y הינם התחום והטווח בהתאם.

דוגמא: $\mathbb{R}^+ \rightarrow \mathbb{R}^2$: הינה פונקציה, הולוקחת וקטורים גיאומטריים דו-ממדים, ומחזירה מספר ממשי אי שלילי. הפונקציה עצמה \cdot 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 $0 \neq n$ מתקיים: $\frac{d(x^n)}{dx} = nx^{n-1}$.
- חיבור או חיסור פונקציות: $\frac{d(f(x)+g(x))}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$.
- מכפלת שתי פונקציות: $\frac{d(f(x) \cdot g(x))}{dx} = \frac{f(x)dg(x)}{dx} + \frac{g(x)df(x)}{dx}$.
- כלל שרשרת: $\frac{df(g(x))}{dx} = \frac{df}{dg} \frac{dg}{dx}$.

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

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

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

נקודות אקסטרום

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

לדוגמה: $x^3 = f(x)$. נגזרת הפונקציה הינה $3x^2 = f'(x)$ והוא 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 כר שבמקרה זה ידיעת אחד אינה משפיעה על הסיכוי של השני.

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

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

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

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

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

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

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

נוסחת ההסתברות השלמה וחוק ב'י'

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

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

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

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

משפט הכללה 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

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

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

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

פונקציה זו מקיימת שלוש אקסiomות:

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

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

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

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

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

פונקציית ההתפלגות מקיימת את התכונות הבאות:

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

תכונות ופרמטרים עבור משתנה מקרי

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

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

תכונות:

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

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

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

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

תכונות:

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

שונות משותפת – ממד ליחס אפשרי בין שני משתנים מקרים:

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

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

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

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

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

פונקציה יוצרת מומנטים (התמרת לפילס של פונקציית הצפיפות):

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

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

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

המומנט הראשון הוא התוחלת והמומנט השני הוא השונות.

התפלגות מיוחדות (בדיד)

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

התפלגות ברנולי: ($X \sim Ber(p)$)

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

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

התפלגות בינומית: ($X \sim B(n, p)$)

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

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

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

התפלגות גיאומטרית: ($X \sim G(p)$)

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

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

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

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

התפלגות אחדית: ($X \sim U[a, b]$)

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

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

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

יש לשים לב שכאן ההתפלגות נמצדת ליחידת זמן.

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

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

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

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

חוitzrim 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 מיוחדת (רציף)

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

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

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

התפלגות אחידה: $X \sim U(a, b)$

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

פונקציית הצפיפות: $F(t) = \frac{t-a}{b-a}$. פונקציית ההתפלגות המצטברת: $f(x) = \frac{1}{b-a}$.

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

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

התפלגות נורמלית היא ההתפלגות חסובה מאוד כיון שהיא מופיעה בהמוני מקרים. פונקציית הצפיפות של ההתפלגות הנורמלית נראה כmo פעמן, כאשר לעקומה קוראים גם עקומת גאות. ההתפלגות הנורמליות 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:

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

המומנטים מקיימים:

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

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

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

אי שיוניים

מרקוב

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

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

צ'בישוב

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

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

צ'רנוフ

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

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

ינט'

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

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

התפלגות דו ממדית

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

תכונות:

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

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

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

אם y, x בלתי תלויים אז מתקיים:

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

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

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

באופן שקול מתקיים:

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

התפלגות שולית:

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

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

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

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

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

ստטיטיסטיקה היסקית

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

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

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

דוגמאות לאמדים בלתי מוטים:

אומד בלתי מוטה לתוחלת – ממוצע חשבוני:

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

אומד בלתי מוטה לשונות:

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

הוכחה:

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

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

אומד נראות מרבית – (MLE)

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

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

זהוי פונקציה hn של הדגימות והן של הפרמטר.

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

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

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

מכoon *-log* הינה מונוטונית, למקסם את L שקול למקסם את $(\log(L))$, 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 בהתפלגות פואסונית:

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

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

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

מוסבר למצוא להזיה מקסIMUM, لكن נוציא לה:

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

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

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 נקבל:

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

ובודד את הפרמטר אותו מנוטים לאמוד:

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

וקיבלנו אומד עבור הפרמטר λ , וכך נקבע סט התוצאות, פשוט נציב אותו, ונמצא מפורשות את הערך של האומד. זה בעצם תהליכי מציאת-*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, וכך 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):

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

מציא לוג:

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

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

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

cut בשביל לאמוד את התוחלת יש לגזור לפי μ , וכך לאמוד את השונות יש לגזור לפי σ^2 :

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

וכשנשווה ל-0 נקבל:

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

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

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

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 a -ממדי, ומנסה למצוא משור המפריד בצורה טובה כמה שניתן בין דוגמאות האימון השויות לקטגוריות השונות.

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

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

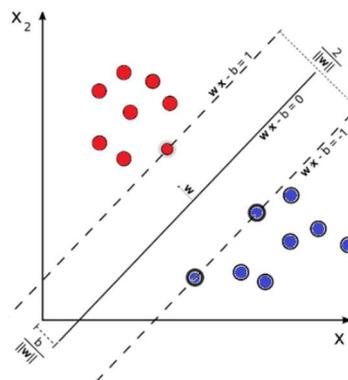
$$w^* = \operatorname{argmin}_w \left(\frac{1}{2} \|w\|^2 \right), \text{ s.t. } y_i (x_i \cdot w + b) \geq 1 \quad \forall i$$

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

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

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

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

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

Soft-Margin (soft SVM)

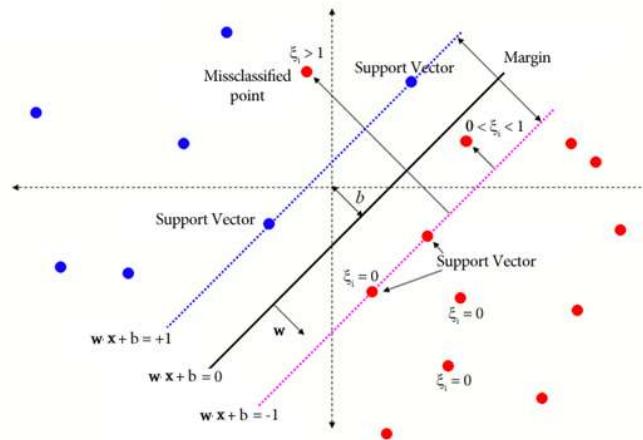
הפרדה מושלמת באמצעות מישור לנארו לעתים קרובות אינה אפשרית, ולכן ניתן להרחיב את המודל כך שיאפשר לנקודות מסוימות לא להיות בצד המותאים להן. הרחבה זו, היוצרת "הפרדה רכה", מאפשרת לטפל בעיות שבהן אין הפרדה לנארו בין הקבוצות, כמו למשל שיש נקודות חריגות. משמעות ההרחבה היא שכל וקטור ממוקם לפחות אחד מהאלצים, אך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

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

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

לשם קבלת אינטואיציה, נשים לב לתפקיד המשתנים:

אם $\xi_i = 0$, מתקיים התנאי שנדרש בהפרדה קשיחה, כלומר הנקודה x_i גם נמצאת בצד הנכון של המפריד וגם מתקיימת הדרישה 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 שאחראי על "ענישה" של דוגמאות שאינן בצד הנכון של המפריד. ערך C גבוה פירושו העדפת הסיווג הנכון על פני שלויים רחבים, ואילו C נמוך מעדייף הכללה (שלויים רחבים), גם במקרים של דוגמאות האימון הספציפיות אין 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案 אשר 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 מסוג לינארי. לשם כך, משתמשים בטריק קרNEL שמקבל כקלט וקטורים למרחב המקורי ומחזיר את המכפלה הפנימית (dot product) של הווקטוריים למרחב החדש (נקרא גם מרחב התכונות – feature space):

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

דוגמאות של פונקציות קרNEL נפוצות:
קרNEL לינארי:

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

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

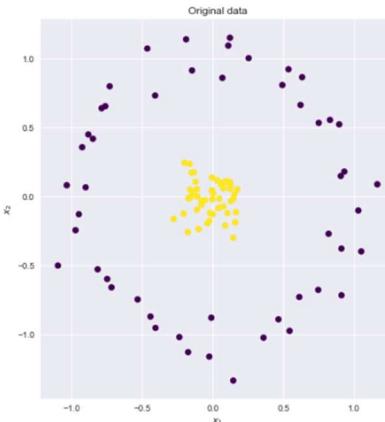
$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + c)^d$$

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 $c = 0$, הקרNEL 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 בתנויי האימון.

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



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

- נמפה את התכונות המקוריות למרחב הגובה יותר (מייפוי תכונות).
- نبצע SVM לינארי למרחב החדש.
- נמצא את קבועות המשקלות התואמות את מישור גבול ההחלה.
- 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 $K(x_1, x_2) = \varphi(x_1)^T \varphi(x_2)$. אם נוכל למצוא פונקציית קרナル המקבילה למפת התכונות שלעיל, נוכל להשתמש בפונקציה ייחד עם SVM לינארי וכך לבצע את החישובים בעילוות.

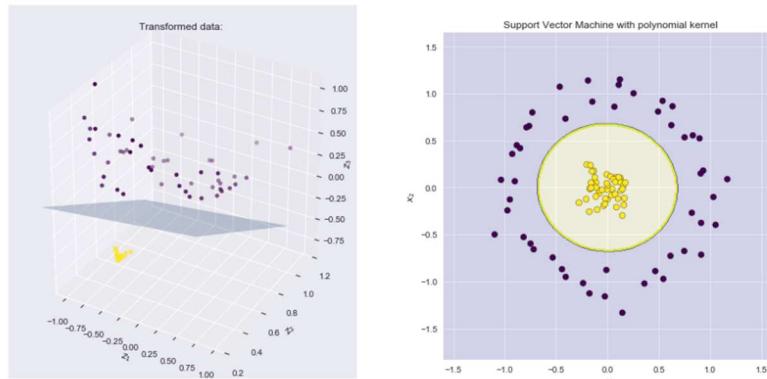
מתברר שמרחב התכונות שלעיל תואם את השימוש בקרナル פולינומי ידוע: $(x^T x')^d = (x' x)^d = K(x, x')$. נבחר $d=2$. נסמן $(x_1, x_2)^T = \alpha$ ונקבל:

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

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

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

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



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

2.1.2 Naive Bayes

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

נניח שיש מודל המקובל וקטור מאפיינים בינהירים {cab, ראש, משתעל, חום 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 סוגי אפשרים, 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 $(y_k | x)$. הסתפלות הפרIORיות $p(y_k | x)$ ידועה, ובנוסף ידועות הסתפלות המותנות של המאפיינים בהנחה 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 תליי ב- k , ולכן מספיק למצוא את y_k שעבורו המונה מקסימלי. לפי כלל השרשרת מתקיים:

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

בביטוי זה כל האיברים ידועים, ולכן קל שנוצר זה רק להציב את הנתונים ולקבל את y_k עבורו ביטוי זה הכי 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 היה 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 הסתפלות מסוימת עבור המדגם, ולהחשב את הפרמטרים של הסתפלות שיטות שונות (למשל בשערת נראות 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 הסתפלות, ואומרו הם משוערים בשערת MLE או שיטת שערוך אחרת. אם הסתפלות היא לא נורמלית, ניתן להשתמש באלגוריתם [Kernel density estimation](#) עבור שערוך הסתפלות. גישה אחרת להערכתם עם מאפיינים היכולים לקבל ערכים רציפים היא לבצע דיסקרטיזציה לערכים אותם המאפיינים יכולים לקבל.

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

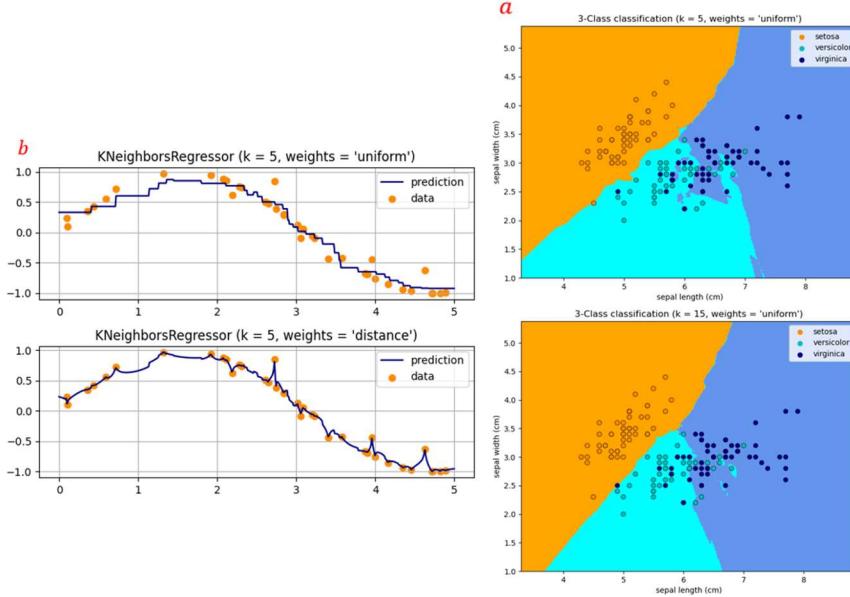
$$\arg \min_k p(y_k) \prod_{i=1}^n \frac{1}{p(x_i|y_k)}$$

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 ובנוסף ידוע ה-label של כל אחת מהן. אלגוריתם זה מתאים הן לבעיות סיווג (שיוך נקודה חדשה למחלקה מסוימת) והן לבעיות רגרסיה (נתינת ערך מסוין לנקודה חדשה). האלגוריתם הינו מודל חסר פרמטרים, והוא מבצע סיווג לנוגדים בעזרת הcrcutta הרוב. עבור כל נקודה במדגם, המודל בוחן את ה-label של K nearest הנקודות הקרובות אליו ביותר, ומסווג את הנקודה לפי ה-label שקיבל את מרבית 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 ה-label 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 מהנקודה אותה רצים לחשב, כך שכל ששן מסויים קרוב יותר לנקודה אותה רצים לחשב כך הוא יותר ישפיע עליה, ביחס של הופכי המראק בין השן לבין הנקודה (distance).



איור 2.4 (a) סיווג בעזרת אלגוריתם N-NN-K: מסוגים את המראק לאזוריים בהתאם ל-K השכנים הקרובים ביותר, כך שאם TABOA נקודה חדשה היא תהיה מסוגת בהתאם לצבע של האזור שלאה, הנקבע כאמור לפי השכנים הקרובים ביותר. ניתן לראות שישandel הבדל בין ערכי K שונים, וככל ש-K יותר גבוהה ככל האזוריים יותר חלקים יש פוחות מובלעת. (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 מהנקודה 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 אובייקטיבים של המחלקה, כמוör כל נקודה 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 לסwoog את התלמידים למספר קבוצות. אם יעשה זאת לפי צבע שער, עיניים, לדוגמא, סביר להניח שלא יתקשה בקשר; אם לעומת זאת הוא ינסה לסwoog 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 קלתת 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 שצורך להיקבע מראש. היפר פרמטר זה קובע את מספר הנקודות 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 לנכוני האימון, מה שmobivill לדיווק גבוה נתונים האימון, ודיווק נמוך נתונים המבחן. מן העבר השני, כאשר K גבוהה מדי, למשל K = 100, נוצר מצב ההופך – מודל שמתחשב יותר מדי בDATAה ולא מצליח למצא הכללה נכונה לסיווג. מומלץ לבחור K איזוגי לבגלאוון הפעולה של האלגוריתם – הכרעת

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

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

השיטה הנפוצה ביותר למדידת מרחק בין משתנים רציפים היא מרחק אוקלי – עבור שתי נקודות במרחב, המרחק ביניהם יחושב לפי הנוסחה: $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} = d$. במידה ומדובר במשתנים בדידים, כגון טקסט, ניתן להשתמש במטריקות אחרות כגון מרחק המיניג, המודד את מספר השינויים הדרושים לכך להפוך מחרוזת אחת למחרוזת 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 מספרן הגדל, מחלוקת זו צפואה להיות נפוצה הרבה יותר בקרב K השכנים של כל תצפית. הדבר עשוי להוביל לתוכאות מוטטות, 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 שבהתנאי 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- $\{\Sigma_k, \mu_k, \dots, \Sigma_1, \mu_1\}$, 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:

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

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

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

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

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

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

נוסף:

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

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

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

ונקבל:

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

וזהו משטח הפרדה ריבועי.

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

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

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

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

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

ומסווג המתקבל הינו:

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

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

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:

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

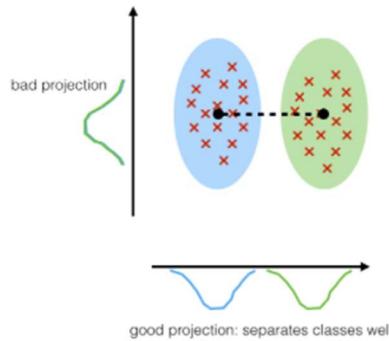
$$b = \frac{1}{2} (\mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1) + \log \frac{p(y = 1)}{p(y = 0)}$$

והסיגו הימנו:

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

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- שתביא לסייע 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

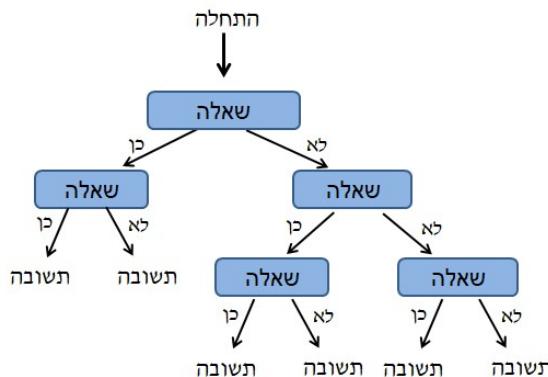
1. הקדמה

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 וכללי הסיגו לקבוצות השונות יכולים להציגו בצורה עצ, אלגוריתם זה נקרא "עדי החלטה".

הגישות השונות בעדי החלטה פשוטות וឥטיות 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 עצים כך יכולת הפרשנות של המודל נהיית מורכבת יותר ופחות אינטואיטיבית לצופה שאין מעורב במבנה המודל (למשל גורם עסקי בארגון שאנו מונחים 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)
- (עומק) – מספר ה-nodes במסלול הארוך ביותר בעץ.
- Depth

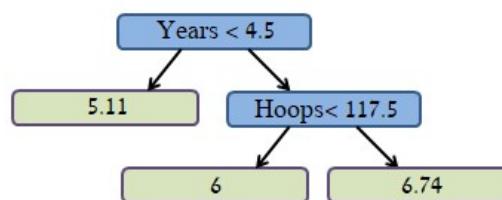
יסודות של עצי החלטה

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 ללא 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 שנים ($years < 4.5$), 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 תכיפות חדשה

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

בסופו של דבר העץ מחלק את השחקנים לשלווש קבוצות:

- שחוקנים ששחיקו 4 עונות או פחות:

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

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

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

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

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

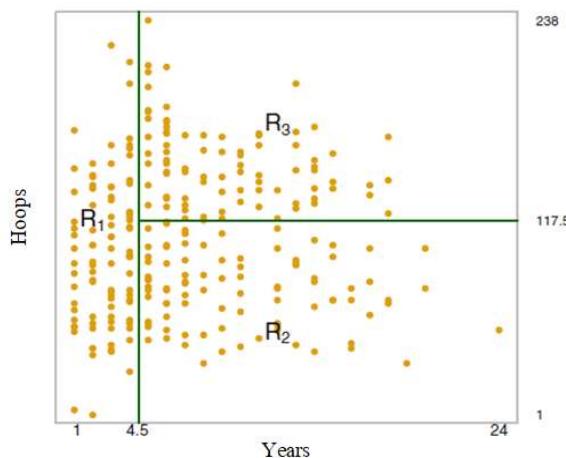
בתואם לדוגמה של העץ, הקבוצות R_1, R_2, R_3 מכונות בשם nodes terminal, או "עלים" של העץ.

אפשר לפרש את עץ הרגרסיה שבדוגמה הצורה נוספת: $Years$ הוא הפרמטר המרכזי ביותר בקביעה מה יהיה גובה השכר, וחקוקנים עם פחות שנות ניסיון ירוויחו פחות משחקנים מנוסים יותר. עבור שחוקן יחסית חדש (עד 5 שנים בלבד) הפרמטר של כמות הסליםאותה הוא קלע בעונה הקודמת מתברר כפחות משפע על השכר. כמובן, 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 קלווע בשנה הקודמת כן משפעה על השכר, וחקוקנים שקבעו הרבה סלים בשנה הקודמת כנראה ירוויחו יותר כסף מאשר שחוקנים עם אותן ניסיון אך קלווע 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 ייעלה, 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 ההחלטה 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 תצפית שנופלת באזורי 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 תנועה, ומתר ניסיון לעקוף את הפקק נבחרת הפניה הראשונה שנראית פחות פוקואה, מבלי להתחשב בפקק שעשו לבוא בהמשך. כਮובן שפניה זו לא בהכרח טוביל לדרך יותר מהירה, ויתכן שבראייה יותר ארוכת טווח דזוקא היה מוטב להימנע מפניה זו כיון שהיא מובילת לפיקק גדול יותר בהמשך.

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

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

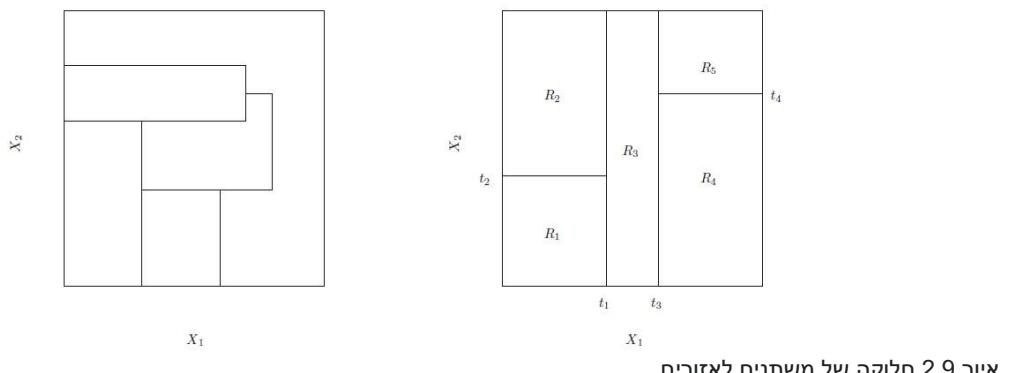
$$\begin{aligned} R_1(j, s) &= \{X | X_j < s\} \\ R_2(j, s) &= \{X | X_j \geq s\} \end{aligned}$$

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

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

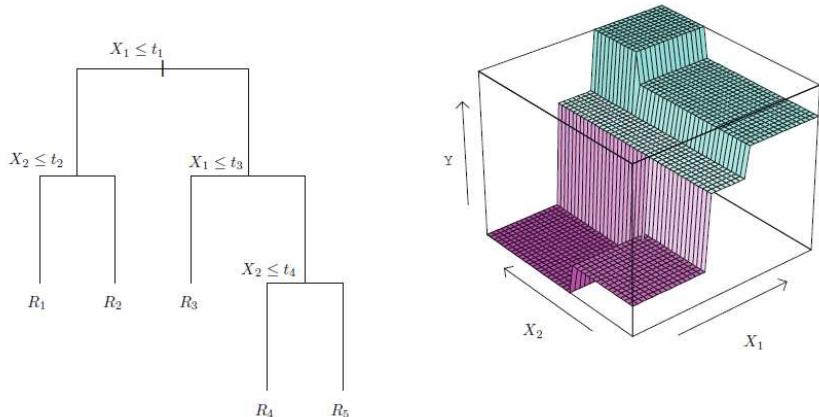
מציאת ערכי s ו- j שמבאים למינימום את המשוואה יכולה להתבצע די בקלות באמצעות הפרמטרים לא גודלה מייד, אך זה תהליך שעלול להיות די סבוך כשייש הרבה פרמטרים.

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



איור 2.9 חלוקה של משתנים לאזורים.

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, ומצד ימין ישנה פרספקטיבה רחבה כיצד חיזויים מתבצעים באמצעות העץ.

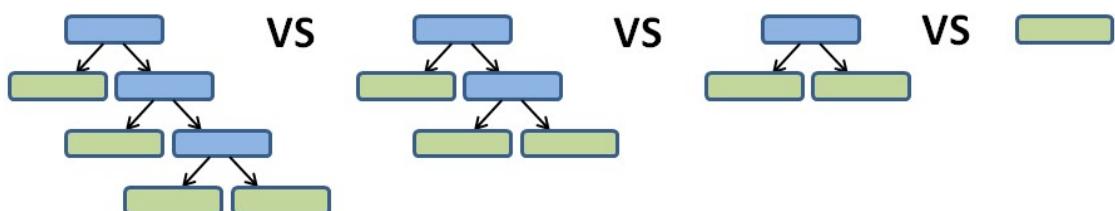
3. גיזום (pruning):

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" ומתאים יתר על המידה לנתוני 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 נראה יוביל לשונות 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 נעוץ בכך שהוא קצרת-رأיה. יתרון מכך בו יש פיצול בעל תרומה קטנה יותר והוא יכול להוביל לפיצול אחר בעל תרומה גדולה RSS-ב-RSS בהמשך. שיטה זו מדגלת על אותו פיצול בעל ערך קטן, מארח והוא לא עובר 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 ציפויות. כמובן שצעד זה יגרום לירידה בביטויים על פני סט האימון, אך השαιפה היא שזה ישתלם בבדיקה השגיאה בסט המבחן.

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

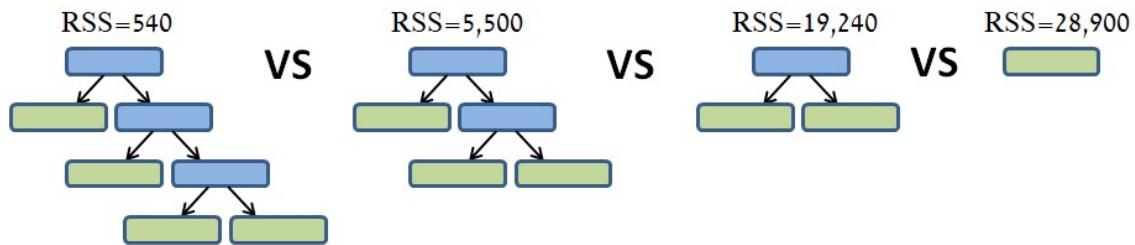


איור 11.2 חיפוש תת עץ אופטימי ביחס ל- T_0 . מתחילה מ- T_0 (העץ השמאלי) וכל פעם מורידים פיצול יחיד עד למגיעים ל-root.

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 רגסティ גודל 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 מועד להקטין את השגיאה באמצעות הקטנת ה-RSS, ואילו גיזום עשויה את ההיפך – הוא מועד 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. בין הרצון להימנע עד כמה שניתן מ-overfitting. המוחובר αT מכונה Tree Complexity Penalty, ואומרו הוא מועד "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 הריביב αT בנוסחה שווה ל-0. במקרה זה ה-Tree score יהיה לפחות למדוד RSS:

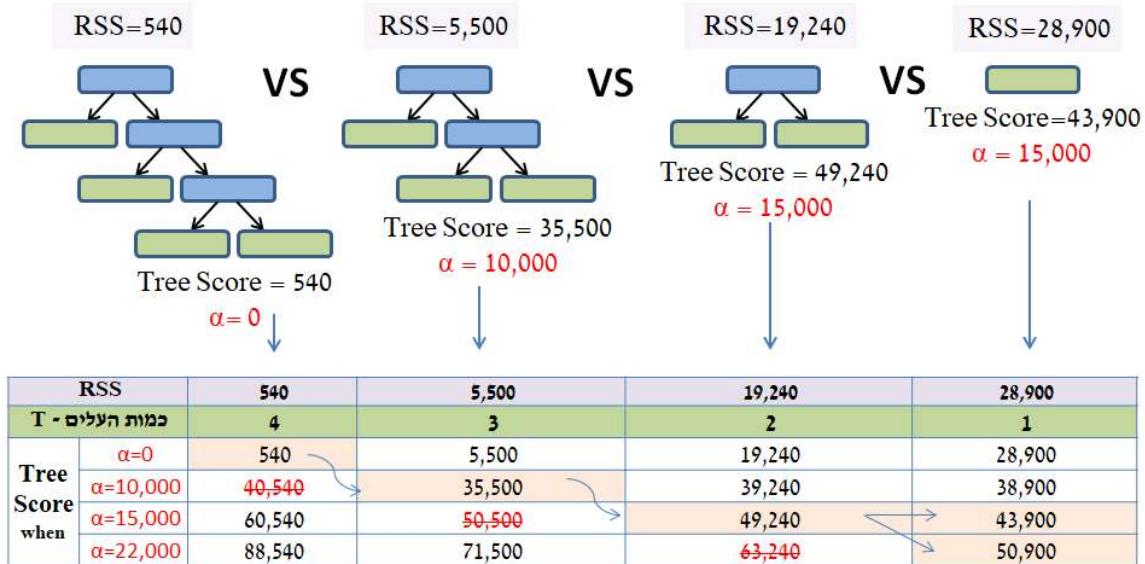
$$\text{Tree score} = \text{RSS} + 0 \cdot T = \text{RSS}$$

滿ילא שאר שלבי החישוב מיותרים, כי אנחנו כבר ידועים 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 אכן גבוה יותר מאשר 35,500 וכן 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 ערך של α קיים subtree מתאים $T_0 \subset T$ כך שה-tree score המתקבל יהיה קטן ככל האפשר.

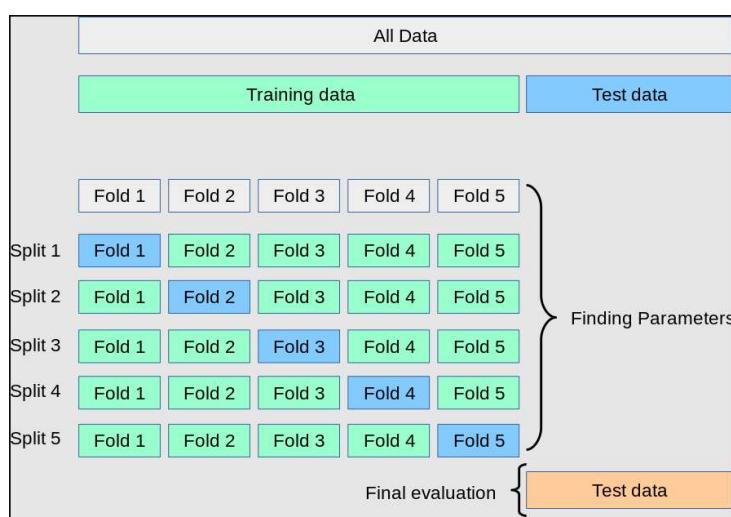


איור 2.13 חישוב ערך α מותאם לכל עץ משנה כך שיתקבל Tree score קטן ככל האפשר.

icut, נבחר ב-base score העץ הנמוך ביותר – ככלומר 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:

- ערכי α חשובים, כיוון שלכל α עשוי להתקבל עץ אחר בעל ציון מינימלי. עד לשלב זה העץ נבנה ביחס לכל הדאטה (בלי חלוקה ל-Test ו-Train), אך המטריה היא לבחון מהו ערך ה- α שייתן את התוצאה האופטימלית בעזרת העץ הגזום עבור דאטה חדש.

ט. כדי לבחור את ערך ה- α האופטימלי ניתן להשתמש ב-cross-validation. לשם כך, יש לקחת את הדאטה המלא (ממנו נבנה העץ הראשון – T_0), ולחילק אותו באופן הבא:

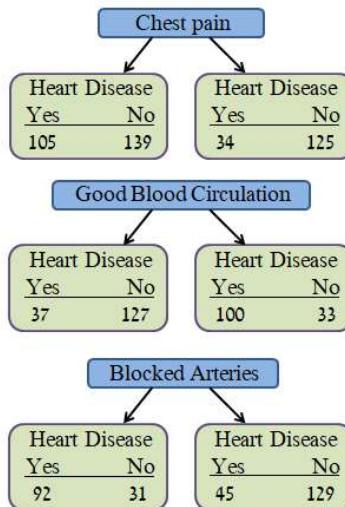


. $K_{fold} = 5$ ב cross-validation 2.14

- 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 ערכיו 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'א 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 ערכיו α שנקבעו כבר מראש בסעיף ז'. לאחר K האיטרציות יש לבדוק מיהו העץ עם RSS המינימלי מבין כל האיטרציות, ומהו ערך α של העץ 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.
 - ב כדי 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 ייתן 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. קר 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.

א. ממד Gini index :

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

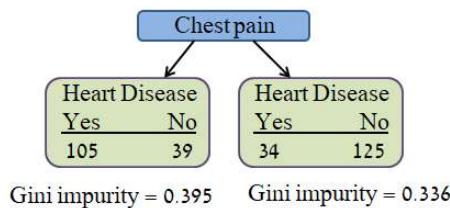
$$Gini = 1 - \sum_{j=1}^J p_j^2$$

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:

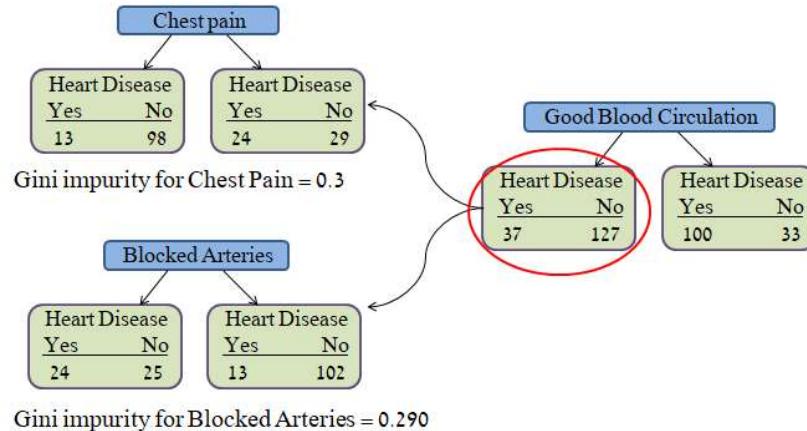
$$\text{Gini impurity for chest pain} = \left(\frac{144}{144 + 159} \right) \times 0.395 + \left(\frac{159}{144 + 159} \right) \times 0.336 = 0.364$$

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

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

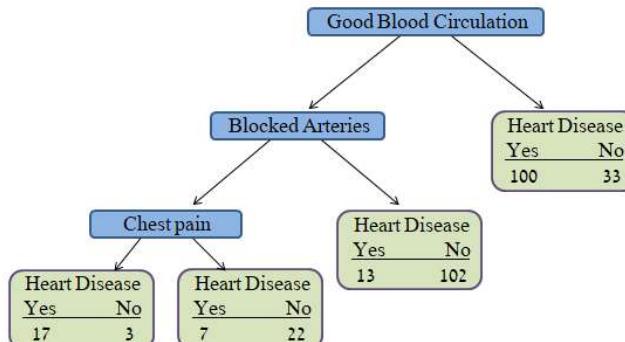
$$\text{Gini impurity for blocked arteries} = 0.381$$

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:



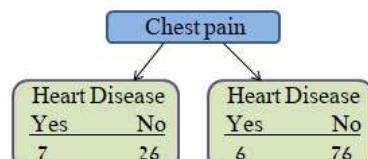
איור 2.17 חישוב מדד Gini עבור שאר הפרמטרים לאחר קביעת ה-root.

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



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

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



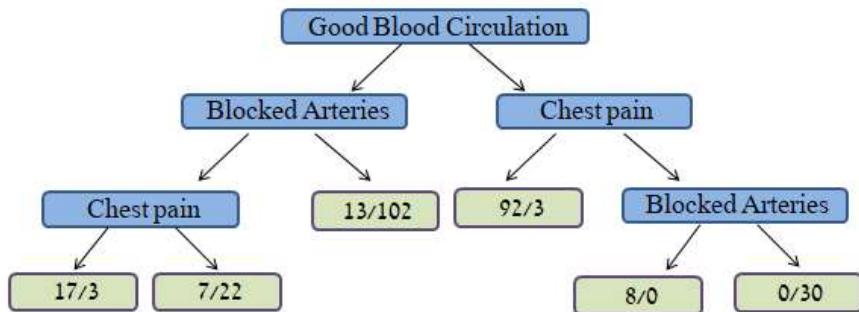
$$\text{Gini impurity for Chest Pain} = 0.29$$

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

מדד InG שהתקבל הינו 0.29, בעוד שבבלי הפיצול המדרד של העלה היה 0.2, וכך במקרה זהה עדיף להשאיר אותו כפי שהוא לפני ניסיון הפיצול.icut באופן דומה נבנה גם את הענף הימני של העץ:

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

עż טיפוסי לאחר סיום התהילה נראה כך:

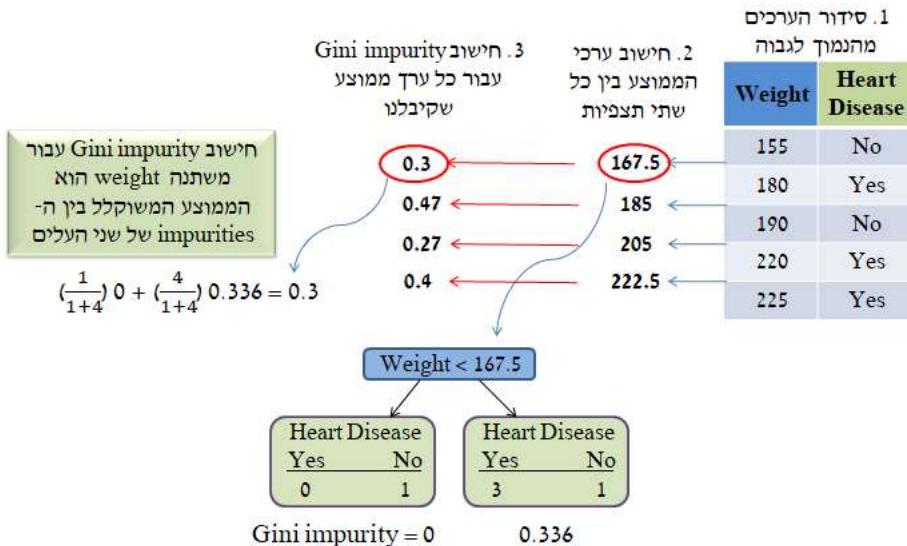


איור 2.20 חישוב מדד Gini עבור פיצול לפי הפרמטר Chest pain ביחס לעלה 13/102.

מדד Gini שהתקבל הינו 0.29, בעוד שבלי הפיצול המדד של העלה היה

התהילך שתואר מתאים למצבים בהם הפרמטרים מתפצלים באופן בינהי, כמו למשל הפיצול של כל פרמטר נקבע על ידי שאלה שעליה יש תשובה של כן או לא. במקרים בהם ישנו משתנים רציפים, הפיצול דרוש כמו שלבים מקדים:

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



איור 2.21 פיצול פרמטרים רציפים לפי מדד Gini.

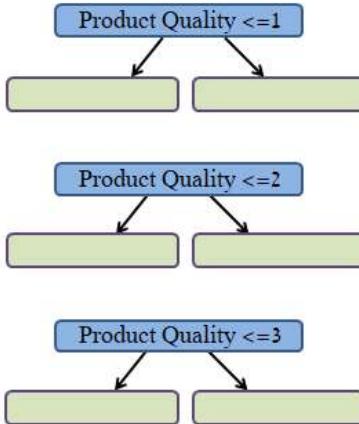
אנו מקבלים את ה-Gini הנמוך ביותר מתי שאנחנו קובעים threshold של 205.

אך זה ה-Gini ונשנה המשנה weight ליתר המשתנים.

עד עכשיו על איך מחלקים משתנה רציף ואיך מחלקים משתנה בינהרי (שאלות כ/לא). כתת נדבר איך מחלקים משתנים קטגוריאליים. למשל: משתנה מודרג שמקבל ערכים מ-1 עד 4. למשל: טיב מוצר 1-4. או משתנה שמקביל מספר קטגוריות. למשל: צבע מועדף (מקיל ערכים: אדום, ירוק, כחול וכו').

משתנה מודרג מאד דומה למשתנה רציף, חוץ מזה שהוא אナンון ארים לחשב בו את היצוא עבור כל חלוקה אפשרית.

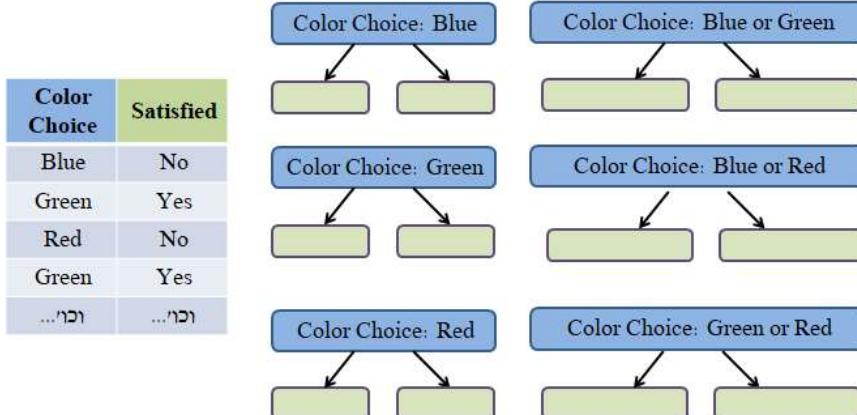
Product Quality	Satisfied
1	No
1	Yes
3	No
1	Yes
... וכך...	... וכך...



שימוש לב: אנחנו לא צריכים לחשב את ה- impurity score ל- $\text{prod. quality} \leq 4$ כי השוואת כל ערך ב�לוב בעצם את כלם

איור 2.22 פיצול פרמטרים קטגוריאליים לפי מדד Gini.

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



איור 2.23 פיצול פרמטרים קטגוריאליים לפי מדד Gini.

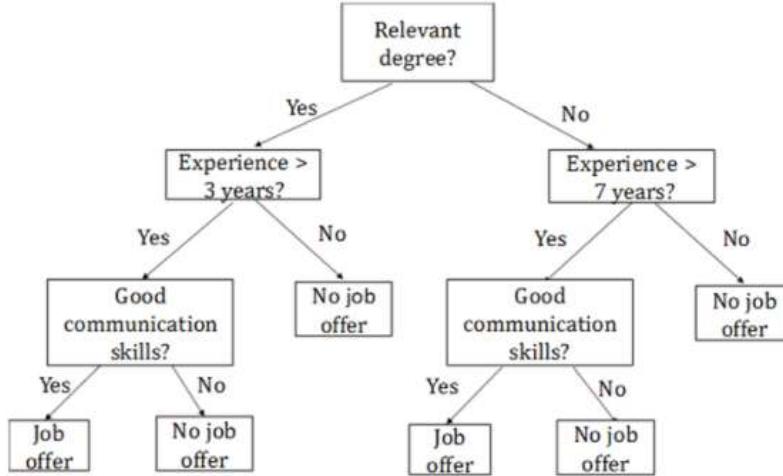
ב. מדד Entropy & Information Gain

מדד נוסף לפיצול צמתי העז מtabased על האנתרופיה של העלים, בעזרתה ניתן לבחון את ה-information gain המקיים מכל פיצול. אנטרופיה באה לממדוד את השגיאה של התפלגות המשתנה הנבחן מול משתנה המטרה. נניח 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, הפיצול האופטימלי נבחר על ידי המשתנה 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 המשתנה המוסבר, מדד האנתרופיה יהיה 1 (זהה הערך המקיים שמדד אנטרופיה יכול לקבל).

לעיל פירטנו שלב אחריו שלב את חישוב מדד Gini ל- Use-Case. במקרה אחרה פשוטה יותר הנוגעת לקבלת מועד לתפקיד מסוים, כאשר מטרת העז 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 ומוגדרים 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 הוודאות שלו לקבל את המשרה. אם אי הוודאות נמדדת באמצעות מדד ה-*Entropy*, Entropy, הרי שהרווח מהמידע הוא:

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

הן עבור מועמדים בעלי תואר והן עבור כלה ללא תואר, המשטנה 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 ההצלחות של התוצאות. במקרה זה פונקציית המבחן תהיה:

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

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

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

סיכום

עż החלטה (Decision Tree) הינו אלגוריתם 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'ני (Gini). כאשר ערכו של משתנה מסוים נחזה, אי הוודאות נמדדת באמצעות RSS. חשיבותו של מאפיין הינו הרוח מההמיעץ הצפוי שלו (Expected Information Gain). הרוח מההמיעץ הצפוי נמדד על ידי הירידה באז הוודאות הצפואה אשרתרחש כאשר יתקבל מידע אודות המאפיין.

במקרה של חלוקת **משתנה קטgorיאלי**, המידע המתkeletal הינו על פי רוב 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 "מרקזום 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 לפחות חיראות 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 תוצאות דורשים יותר זמן הרצה לאימון המודל, ומשכך מדובר באלגוריתם "יקר" במשאבים.
- האלגוריתם של עż ההחלטה אינו מספק ליישום רגסיה ולניבוי ערכים רציפים.
- עż ההחלטה נוטה לעתים תוצאות לנוטות ל-Overfitting.

2.2 Unsupervised Learning Algorithms

2.2.1 K-means

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

סנטרואיד הוא מונח מתחום 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 המזעיר את סכום ריבוע המרחק של כל הנקודות באשכול. מدد זה נקרא WCSS, והוא מدد משמעותי ביותר בקרוב לאלגוריתמים שմבצעים חלוקה לאשכולות, K-means. הסיבה להזקה במשווה היא שאנו רוצים להגבר את ההשפעה של המרחק, מעין "עונש" לתוצאות רחוקות מהמרכז.

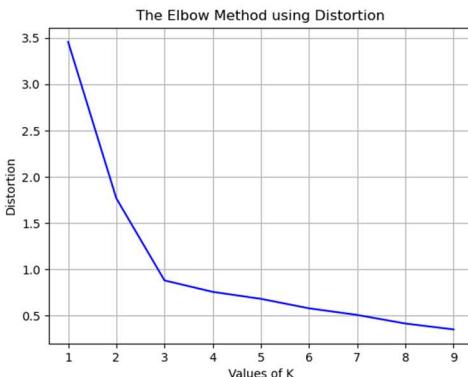
מדד WCSS הוא אחד הדריכים המקבילים ביוטר להעיר את תוצאות החלוקת לאשכולות ב-K-means. היתרונו של ממד זה הוא האפשרות לראות את מידת ההצלחה של המודל, כמובן לקבל מספר ממש שמאמת את הצלחת המודל. מנגד, WCSS הוא מסוףר ללא תחום מסוים והוא דרוש פרשנות, כיון שהערך והמשמעות שלו משתנים ממודל למודל. ערך מסוים יכול להיחשב 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 ממד WCSS ובין מספר האשכולות הרצוי: ככל שמספר האשכולות גדול יותר, כך ה-WCSS יקטן. הדבר מתיישב עם הריגיון – פיזור סentrואידים רבים (כלומר, חילקה ליותר אשכולות) על פני הנתונים יוביל לכך שבhartech 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.

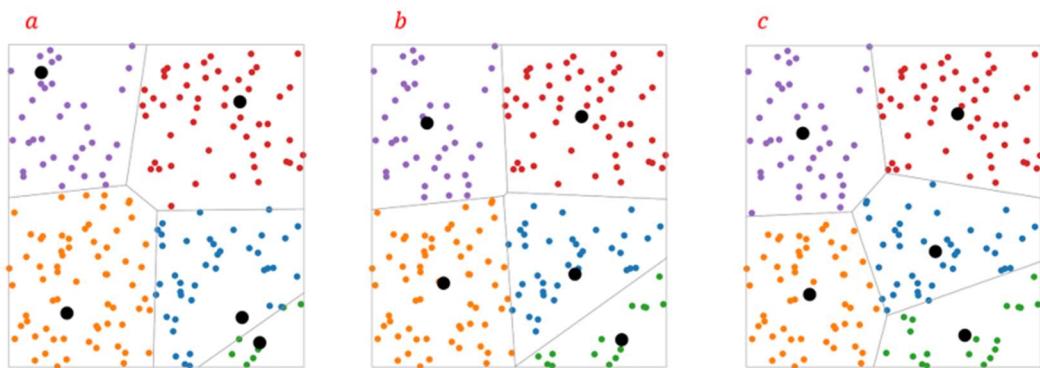
לכן מצד אחד, נרצה לבחור K גדול שמיוצר את ה-WCSS; מצד שני, הסיבה שהשתמשנו ב-K-means מלכתחילה היא בכך לפשט את הנתונים למספר סביר של אשכולות, כזה שיאפשר לנו לעורך אנליזה נוחה. שיטת המפרק (Elbow method) היא טכנית המשמשת לפתור סוגיה זו. הרעיון הוא לבחור את ה- K הקטן ביותר שמננו השיפור במדד ה-WCSS הוא מטען 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. בדוגמה זו ניתן לראות שבמעבר של $K = 2$ - 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 $K = 3$ היא הינה בחרה 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 משוככת לסנטראיד הקרוב אליה ביותר.
- ג. **עדכן:** הסנטראיד מוחזק שכ-הWCSS של המודול יمواזר.
- ד. חזרה על שלבים ב, ג עד אשר הסנטראידים לא דומים לאחר העדכון, כלומר יש התכנסות.



איור 2.7 (a) אתחול 6 סנטראידים באופן רנדומלי. (b) שיר כל נקודה לסנטראיד הקרוב ביותר אליו, ועדכן הסנטראידים לפי מודד-WCSS. (c) 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 ה- 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 המינימום הראשון 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 למקם 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 להסתפלגות 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 להסתכל 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:

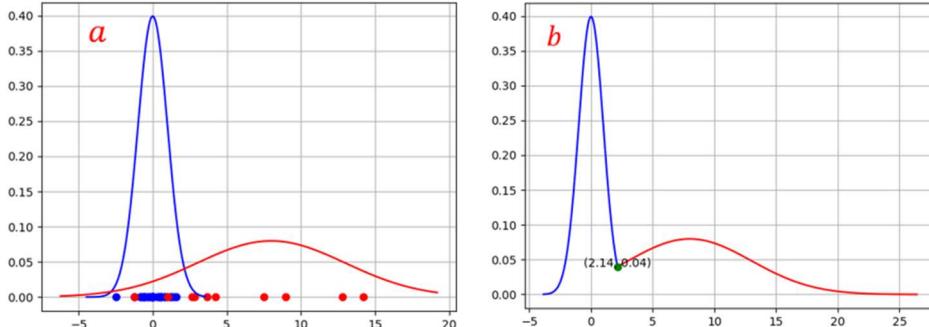
$$p(y = i|x) = \frac{p(y = i) \cdot p(x|y = i)}{p(x)}$$

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:

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

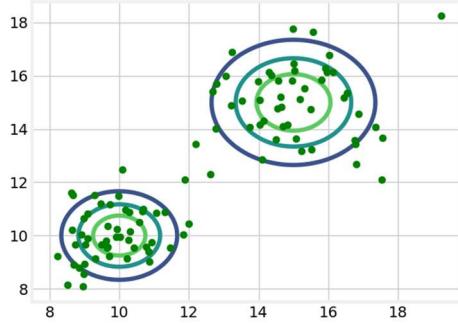


איור 2.8 (a) תערובת של שני גאוסיאנים במדוד אחד: בשלב ראשון מחלקים את הנקודות לשני אשכולות ומתייחסים לכל אשכול התפלגות מסוימת. במקרה זה אשכול אחד (מוסמן בכחול) הותאם להתפלגות $\mathcal{N}(0, 1)$, ואשכול אחד (מוסמן באדום) הותאם להתפלגות $\mathcal{N}(8.5, 1)$. (b) נקודה חדשה x תסוג לאשכול הכחול אם $2.14 < x < 2.14 + 2.14 \cdot 0.04 > 2.14 + 0.04 = 2.18$. 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 מה气colonות, יש לבדוק את ערך ההתפלגות בנקודה החדש. ההתפלגות שבעזרה ההסתברות (x) היא הגדולה ביותר, היא זאת שאליה תהיה משוייכת הנקודה. ההתפלגות יכולות להיות בחד ממד, אך הן יכולות להיות גם בממד יותר גבוה. למשל אם מסתכלים על מישור, ניתן להתאים לכל אשכול ההתפלגות נורמלית דו-ממדית. במקרה ה- n -ממד, ההתפלגות נורמלית $(\Sigma, \mu) \sim \mathcal{N}(\Sigma, \mu)$ היא בעלת הצפיפות:

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

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



איור 2.9 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 הפשוט:

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

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

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

קייבלו למשה את הפוסטוריור y_i (האשכול אליו רוצים לשיר את x_i), אך הוא לא נתן אלא הוא חבו. כדי לחשב את המבוקש ננחש ערך התחלתי θ - 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 תוחאמ לאות מסוימים y_i . עבור כל הנקודות y_i נחשב תוחלת ובעזרתה נגדר את הפונקציה $(Q(\theta, \theta_0), Q(\theta_0))$, 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 להתקנות.

עבור Mixture of Gaussians נוכל לחשב באופן מפורש את הביטויים:

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

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

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

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

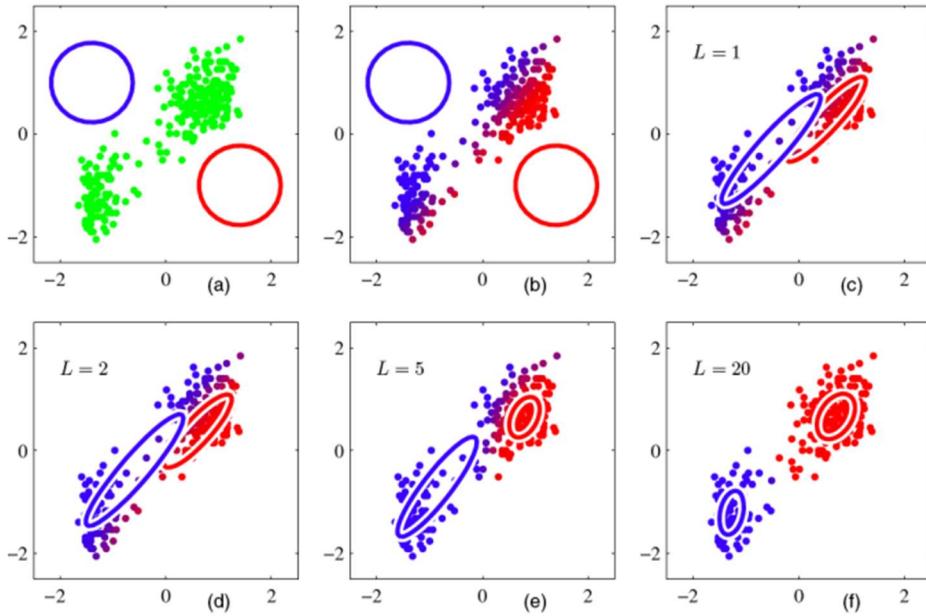
וכעת ניתן לגזר ולמצוא אופטימום:

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

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

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

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 ההטפלוגיות, ובכל איטרציה יש שיפור כך שההטפלוגיות מייצגות בצורה יותר טובה את הדadata המקורי.

ונכון שהאלגוריתם משתמש בכל איטרציה, ככלمر שעבור כל (θ_0, θ) מתקיים:

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

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

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

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 $H(\theta, \theta_0) \geq H(\theta_0, \theta_0)$, 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

האשכולות הקרובים ביותר זה לזה ואותם לאשכל 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: $\min\{d(a, b) : a \in A, b \in B\}$.

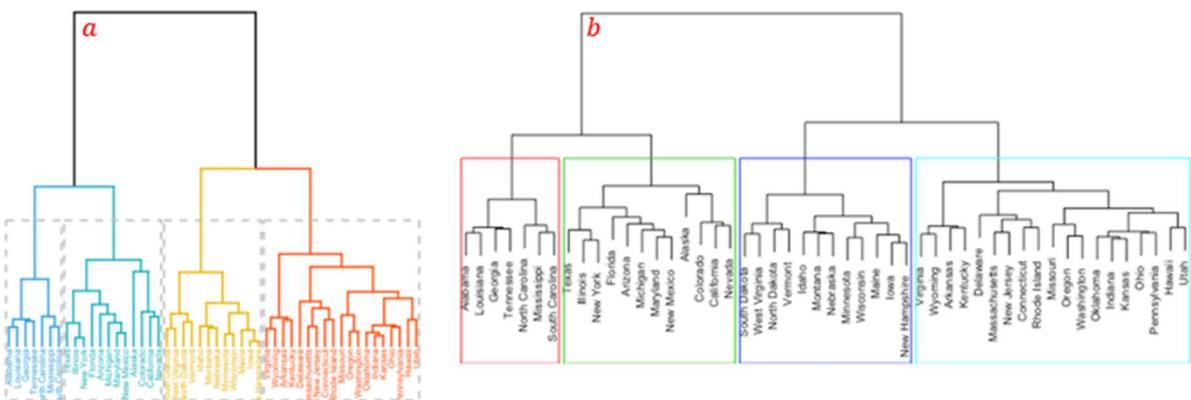
Unweighted average linkage clustering (UPGMA): $\frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$.

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 אשכל עצמאי ובכל צעד של האלגוריתם מסpter האשכולות קטן באחד. במקרים אחרים, האלגוריתם בונה את האשכולות ממצב שבו אין למעשה חלוקה לאשכולות למשך שבועות נסקרים נציגים אשכולים וגדים.

בשיטה זו מבצעים פעולה הפוכה – מסתכלים על כל הנקודות האשכל אחד, ואז בכל שלב מבצעים חלוקה של אחד האשכולות לפי כלל חלוקה שנקבע מראש, עד שmaguiim 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 האשכל בעל השונות היכי גדולה ומחלקים אותו לשניים. שיטה זו מכונה "top-down" ("down-top") – היא שתחילה יש אשכל יחיד ובכל צעד של האלגוריתם מתווסף עוד אשכל.

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



איור 2.11 תצוגה של Hierarchical Clustering (a) – התחילה האשכל יחיד ופיזול עד שמגעים למספר האשכולות המקורי (במקרה זה 4). (b) – בהתחילה כל נקודה הינה אשכל, ובכל צעד מחברים שני אשכולות עד שמגעים 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 $N_k(x_i)$.Cutנqdior את $N_k(x_i)$ של כל נקודה כמרחק שלה מהשכן הרחוק ביותר מבין השכנים ב- $N_k(x_i)$. אם למשל $k = 3$, אז $N_k(x_i)$ הוא סט המכיל את שלושת השכנים הקרובים ביותר ל- x_i , וה-distance k-between הוא המרחק השליishi היכי קרוב. חישוב המרחק בין שני שכנים נתון לבחירה – זה יכול להיות למשל מרחק אוקלידי, מרחק מנהטן ועוד. עבור בחירה של מרחק אוקלידי, ניתן להסתכל על k -distance – מנגנון של מעגל המינימלי המכיל את כל נקודות השיכנות ל- $N_k(x_i)$ הוא k -distance.

לאחר חישוב השיכנות ל- $N_k(x_i)$ 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.

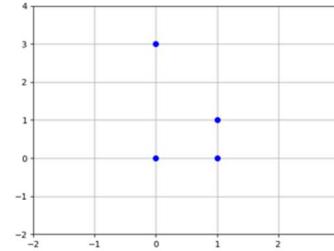
בשלב האחרון בוחנים עבור כל נקודה x את היחס בין ה-LRD שלה וה-LRD של $N_k(x)$. היחס הזה הוא ה-LOF, והוא מחושב באופן הבא:

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

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 מסיכון 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:

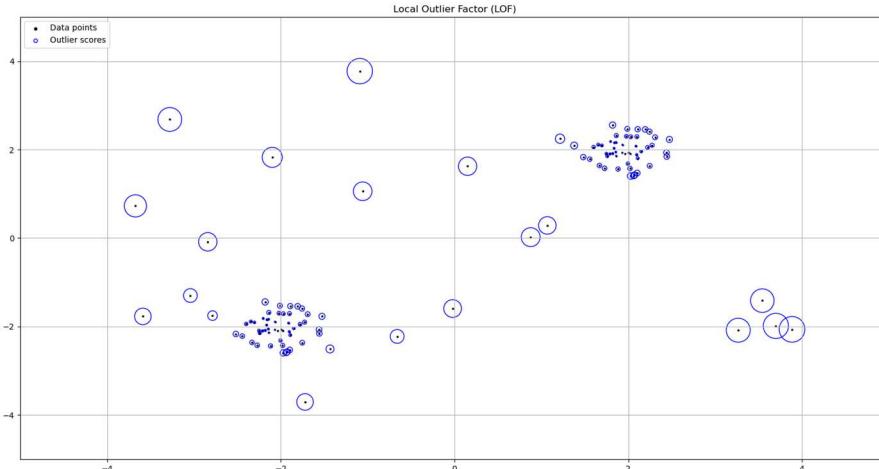
$$LOF_2(A) = \frac{LRD_2(B) + LRD_2(C)}{k} \times \frac{1}{LRD_2(A)} = 0.87$$

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

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

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

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), כקהה הנקודה יותר רוחקה משאכלו של נקודות.

יש שני אתגרים מרכזים בשימוש באלאגוריתם 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 שמייצגת אתגר זה. k גדול לעומת זאת יתגבר על תוצאות פרשנות לתוצאות המתקבלות, ולהחליט על סף מסוים ש-LOF, שהחול ממנו נקודה מסווגת כחריגה. LOF קtan מ-1 הוא בוודאי לא outlier אשר עבר ערכי LOF גדולים מ-1 אין כלל חד משמעי עבור איזה ערך הנקודה היא outlier ועבור איזה ערך היא לא. כדי להתמודד עם אתגרים אלו הוצעו הרחבות לשיטה המקורית, כמו למשתמש בשיטות סטטיסטיות שונות המורידות את התלות בבחירה הערך k (LoOP – Local Outlier Probability), או שיטות סטטיסטיות העוזרות לחתת פרשנות לערכים המתקבלים (Interpreting and Unifying Outlier Scores).

2.3 Dimensionally Reduction

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

דוגמה למערכת מרובת ממדים יכולה להיות מדידת רמות חלבונים (proteins) של גנים (genes) המבוטאים בתא ח', כאשר כל ממד, או אפילו (פיצ'), מתאים לגן אחר. 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 גבואה (אלפי תאים ואלפי גנים בכל תא) מأتגרת את המחקר – הן מבחינותഴיה המאפיינים, או רמות הגנים המבוטאים, הרלוונטיים והמשמעותיים ביותר, והן מבחינות ניסיון למדל את הדאטה בצורה כמה שיוצר פשטוטה. במחקר משנת 2007 נלקחו 105 דגימות של תא סרטן שד, כאשר לכל דגימה (או דוגמא) נמדדו רמות התבניות של 27,648 גנים שונים. כMOVON שלבנה את המידע בצורה הגלומינית זו משימה בלתי אפשרית, ויש הכרח לבצע עליו מניפולציה כלשהיא כדי שהיא אפשר לעבוד איתו.

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

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

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

2.3.1 Principal Components Analysis (PCA)

כפי שהזכר לעיל, ניתוח גורמים ראשיים מבוסס על טרנספורמציה לינארית של המאפיינים המקוריים. הגורם הראשי הראשון (first principal component, PCA₁) הינו הצירוף לינארי של המאפיינים הנתונים בעל השונות הגדולה ביותר. הגורם הראשי השני (second principle component, PCA₂) הוא גם צירוף לינארי של המאפיינים הנתונים, השונות שלו היא השניה הגדולה ביותר, ובנוסף דורשים ש-PCA₂ יהיה אורתוגונלי ל-PCA₁: PCA₂ \perp PCA₁. הגורם השלישי, הוא צירוף לינארי בעל השונות השלישית הגדולה ביותר, ומאנגן לשני הגורמים הראשונים – PCA₂ \perp PCA₃ ו-PCA₃ \perp PCA₁, וכן הלאה כך שהגורם הראשי מסדר i , הוא בעל השונות ה- i -ית הגדולה ביותר בתחום תחת אילוץ של גורמים מאונכים – $j < i \Rightarrow PCA_j \perp PCA_i$. הורדת הממד מתבצעת על ידי לקיחת מספר גורמים ראשיים ראשוניים, והזנתה הקטנים ביותר.

לאחר שאפינו את הגורמים הראשיים בהם אנו מעוניינים, עולה השאלה כיצד ניתן לבצע טרנספורמציה לינארית שבuzzرتה ניתן למצוא את הגורמים הראשיים הללו. נניח שבידינו DATA $\mathbb{R}^{M \times N}$ ב-

$$\text{יד, } \vec{X} = \begin{bmatrix} \vec{X}_1 \\ \vdots \\ \vec{X}_M \end{bmatrix} \in \mathbb{R}^{M \times N}, \text{ כאשר כל וקטור שורה, } \{\vec{X}_m\}_{m=1}^M, \text{ הינו נתוני המדידות של } M \text{ תאים שונים, כאשר עבור כל תא נמדד רמות ביוטי של } N \text{ גנים שונים. נסמן את מטריצת המאפיינים על}$$

המאפיינים השונים בדוגמא מספר m , ובהתאמה, וקטור עמודה $\{1, \dots, N\} \in \mathbb{R}^n$, (שימו לב לשינוי סימון, אינדקס עלון עבור וקטורי עמודה), הינו נתוני המדידות של מאפיין מסוים על כל הדוגמאות. נניח שסכום המדידות עבור כל מאפיין הוא אפס, זאת אומרת שכל מאפיין א-מי-מתקיי:

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

מכיוון שכל עמודה של המטריצה מסמלת ערכים של מאפיין מסוים במדידות שונות, סכום כל עמודה במטריצה \hat{X} הוא אפס. עתה, נרצה לבצע הטלה (טרנספורמציה) 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 $K \times K$ $\hat{W} \in \mathbb{R}^{N \times K}$, כך ש- $\hat{W} \in \mathbb{R}^{M \times K}$. הטלה 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 לנתח באופן פורמלי על ידי שלושה תנאים:

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

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

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

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

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

זאת אומרת:

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

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

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

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

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

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

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

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

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

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

כאשר \widehat{W}^2 הינו הווקטור העצמי המתאים לערך העצמי המקסימלי של \tilde{X} , ובפועל הוא הערך העצמי השני בגודלו. עבור מטריצה $\widehat{X} = \widehat{X}^T$.

(בчисוב השתמשנו בעובדה כי $\widehat{W}^1 \perp \widehat{W}^2$). באופן כללי, כדי למצוא את \widehat{W}^k והערך העצמי המתאים לו λ_k , נגדיר מטריצה חדשה \tilde{X} באופן הבא:

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

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

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

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

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

העצמיים. הוקטורים העצמיים בהגדרתם הינם אורתוגונליים זה לזה, וכיון $\hat{W}^k = \left\| \hat{W} \right\|_F^k$ לכל k , הם בעצם אורתוגונרמליים.

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

$$\hat{X}^m = \hat{X}^m - \text{mean}_n(\hat{X}^m) \quad \text{א. "מרכז" את הנתונים כך שהממוצע עבר כל מאפיין הוא אפס:}$$

$$\hat{S} = (\hat{X})^T \cdot \hat{S} \quad \text{ב. מצא את מטריצת השונות המשותפת } \hat{X}^T \cdot \hat{S}.$$

$$\hat{S} = \hat{W}^T \cdot \hat{L} \cdot \hat{W} \quad \text{ג. מצא את } \hat{W}^T \cdot \hat{L} \cdot \hat{W}.$$

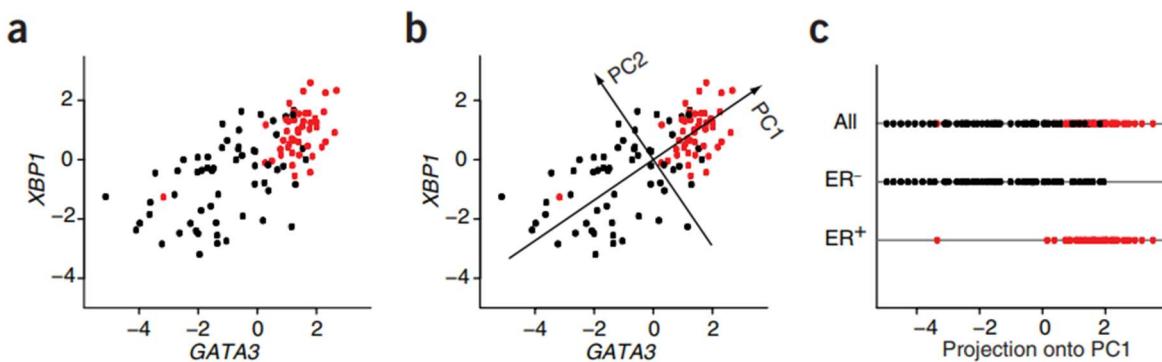
$$\hat{X} = \hat{W} \cdot \hat{L} \cdot \hat{X}^m \quad \text{ד. חשב } \hat{W} \cdot \hat{L} \cdot \hat{X}^m.$$

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

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

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

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



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

נבייא בנוסף דוגמא חשובה מפורטת. נניח וננתן המערכת הדו-ממדי הבאה:

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

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

מערך זה כבר ממורכז, כלומר מתקיים עבור המאפיין הראשון:

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

ועבור המאפיין השני:

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

נחשב את מטריצת השונות המשותפת:

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

על מנת ללקסן מטריצה זו, נפתחו:

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

כאשר לפולינום אופיני זה שתי פתרונות: $\lambda_1 \approx 0.04$, $\lambda_2 \approx 1.02$, $\lambda_1 < \lambda_2$, התוצאות המובאות בחלק זה מקובלות וכאן הסימון \approx .

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

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

כך ש:

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

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

כך שמטריצת המשקלים נתונה על ידי:

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

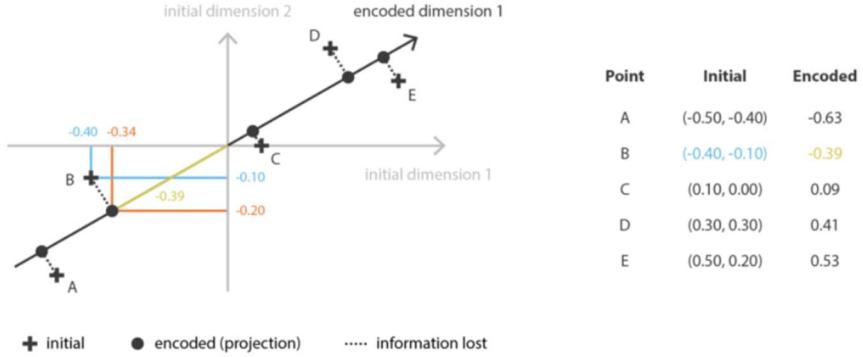
הטלה המדידות למערכת המאפיינים החדשה נתונה על ידי:

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

לכן, המדידות של הגורם הראשי הראשון, נתונת על ידי

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

נראה זאת באופן גרפי:



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

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

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

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

הערך העצמי המקסימלי מקיים:

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

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

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

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

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

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

שהנקודה $\|\vec{X}_{m_1} - \vec{X}_{m_2}\|$ שכנה של \vec{X}_{m_1} גדולה. לעומת זאת כאשר הנקודות רחוקות זו מזו, כלומר, ככלمر השהנטברות 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 שהמדד של \vec{Y}_m הינו נמוך (2 או 3 מדדים). בנוסף, נדרש שנקודות דומות ("שכנות") במרחב \mathcal{X} , ישארו שכנות לאחר המיפוי למרחב \mathcal{Y} . מתרbaru שפונקציית ההנטברות המותנית, המתאימה לתיאור דמיון בין נקודות שכנות למרחב החדש \mathcal{Y} , הינה התפלגות \mathcal{t} , הנקראת גם התפלגות סטודנט עם דרגת חופש אחת (נדון ברענון לבחור בפונקציות הסטברות אלו בהמשך). כך, נכתמת את הדמיון בין m_1 לבין m_2 , על ידי ההנטברות המשותפת Q_{m_1, m_2} המוגדרת באופן הבא:

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

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

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

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

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

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

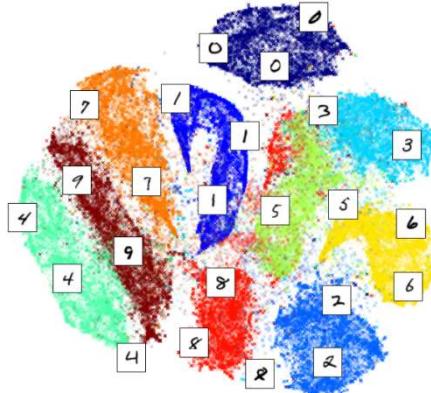
чисוב המינימום באופן אנלטי לא תמיד אפשרי או לא תמיד יעיל, ולכן מקובל להשתמש בשיטת gradient descent שaining שיטה איטרטיבית למציאת המינימום של פונקציה (פירוט על שיטה זו ווריאציות שונות שלה מופיע בחלק 4.3.5). עבור הורדת הממד, חישוב המינימום בעזרת שיטה זו יעשה באופן הבא:

- א. אתחול: * נתון $X \in \mathbb{R}^{M \times N}$.
- * פרמטר לפונקציית הדמיון: בחירת השונות σ^2 .
- * בחירת פרמטרים לאופטימיזציה: קצב הלמידה η , מומנטום $\alpha(t)$.
- ב. חשב את P_{m_1, m_2} .
- ג. אתחל את המיפוי ($s \hat{I}_M \sim N(0, s \hat{I}_M)$) [ז"א בחר את הערכים ההתחלתיים לפני התפלגות גאוסיאנית עם ממוצע 0 וסטיית תקן s (s נבחר להיות קטן, נניח $s = 10^{-4}$). \hat{I}_M מטריצת יחידה].
- ד. עברו איטרציה:

 - * חשב את Q_{m_1, m_2} .
 - * חשב את הגרדיינט של פונקציית המחיר $\frac{\delta C}{\delta \vec{y}}$.
 - * ערך: $\vec{y}^{(t)} = \vec{y}^{(t-1)} + \eta \frac{\delta C}{\delta \vec{y}} + \alpha(t)[\vec{y}^{(t-1)} - \vec{y}^{(t-2)}]$.

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

במאמר המקורי שהציג את השיטה הובאה דוגמא של שימוש באלגוריתם עבור הטלה של הספרות 0 עד 9, המיצגות על ידי תמונה במד 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 מופתעת לאחור אחר, כיוון שבפועל נוצרו עשרה אשכולות שונים, המובחנים בצורה ברורה אחד מהשני. בייצוג הדו-ממדי אין משמעות לציריהם, כיוון שבאלגוריתם זה יש חשיבות רק למרחק היחסיבי בין הנקודות.



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

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

א. סימטריה:

fonkciyah ddimion gaoosianit bin shati nukodot bermachb χ hinya fonkciyah simetria, kolomer $P_{m_1, m_2} = P_{m_2, m_1}$. Aolom nitn lehagidir gam fonkciyah ddimion a-simetria, mabiosset ul htaplagot motanat (bemachom htaplagot meshutaf). fonkciyah motanat natuna ul id:

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

cr sh:

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

ב. בחירת Fonkciyah ddimion bermachom Fonkciyah t:

באלגוריתם שתואר, Fonkciyah ddimion bin shati nukodot bermachb t -y natuna ul id htaplagot t . Nitn lehagidir gam fonkciyah achra, lmasl at Fonkciyah ddimion gaoosianit ubor shati midot bermachb t . Shiyta zo nkrat SNE, u hagradiant shel fonkciyah mchir bmkraha zha natuna ul id:

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

g. Aolom, Fonkciyah ddimion gaoosianit bermachb t ycolah lagrom lcr shnkodot la maod krovot bermachb χ , ymopo lnkodot krovot bermachb t , cion shagaoosian butzim gorom latraktor (mshicha) ychisit chzq bin shati nukodot, gam bmkrim bhem hnukodot ainin maod krovot. Luvomat zat, casher fonkciyah ddimion hinya htaplagot stodont t , shiinya htaplagot um znb cbd yoter, shati nukodot shainin maod krovot ymopo bezora roaya lermachb t cr shainin "nmeskot" ao matkrodot zo lzo. Shiyta achra, nkrat SNE-UNI, mtsia lahshemtsh htaplagot achida, ark gam la chsrn domha l-SNE, casher shati nukodot la maod domot zo lzo, ain "dzhohot" achita at

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

לשיטת t-SNE יש שלוש מגבלות עיקריות

- א. הורדת ממד: השיטה משמשת לויזואליזציה של מידע מממד גבוה בדו-ממד או תלת-ממד. אולם, באופן עקרוני, יתכן ונרצה להוריד את הממד לא לשם הצגתו, אלא לצרכים אחרים, כאשר הממד החדש הינו גדול מ-3. יתכן ובממד גבוה פונקציית התפלגות סטודנט t עם דרגת חופש אחת, אשר לה משקל גבוה יחסית במרקחים גבוהים, לא תשמיר את המבנה של המידע המקורי. לכן, כאשר נרצה להוריד לממד גבוה מ-3, פונקציית התפלגות t עם יותר מדרגת חופשichert מתאמיות יותר.
- ב. קלילת הממדיות: t-SNE מבוססת על מאפיינים מקומיים בין נקודות. השיטה, המבוססת על טריצית מרחק אוקלידי, וכר מנicha לנאריות מקומיות על גבי הירעה המתמטית בה מתקיימות הנקודות. אולם, במרחב נתונים בו הממד הפנימי גבוה, שיטת t-SNE עלולה להיכשל כיוון שהנחה הלינארית לא מתקיימת. למרות שישנן מספר שיטות למצער תופעה זו, עדין, בהגדלה, כאשר הממד הפנימי גבוה, לא ניתן להוריד ממד כרך שמבנה המידע ישמר באופן מלא.
- ג. פונקציית מחיר לא קמורה: הרבה שיטות למידה מבוססות על פונקציית הפסד קמורה, כרך שתיאורטיבית מציאות אופטימיזציה (יחידה) לפונקציה זו אפשרית תמיד. אולם, בשיטת t-SNE, פונקציית המחיר אינה קמורה, והפתרון המתתקבל על ידי האופטימיזציה משתנה בהתאם לפרמטרים הנבחרים.

2.4 Ensemble Learning

2.4.1 Introduction to Ensemble Learning

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

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

בפועל, המודלים במציאות ינעו לאורך חיי קצאות: מודלים עם שונות גבוהה והטיה נמוכה, ומודלים עם שונות נמוכה והטיה גבוהה. הזרחי של המיקום שלנו לאורך ציר זה קרייטי, כיוון שהוא מאפשר לנו לבחור את דרך ההתמודדות הטובה ביותר. יש מספר משפחות של t-SNE models, ושני העקרורים שבהם נקראים "לומדים". Bagging and Boosting. כאשר ניתקל במודלים עם שונות גבוהה, ככל מרובה מודל הסובל מ-Overfitting, לרוב נרצה להשתמש באנסמבל 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 שפועלים ביחד, על-מנת להציגו לתוצאה משופרת. כאמור, אלגוריתמים מסווג bagging נועד להגדיל את יציבות המודל והעלאת הדיק שלו, זאת תוך הורדת השונות והימנענות מ-overfitting. Bagging מרכיב מספר רב של אלגוריתמים, המכונים "לומדים" (Weak learners), כאשר כל אחד מהם מבצע למידה ותחזית על חלק מן הנתונים, מתוך מטרה להציגו לתוצאה איקונית. אלגוריתם bagging הינו שיטה נפוצה ו פשוטה יחסית לשיפור ביצועים, אם כי הוא עשוי להיות קרה מבחינה חשיבותית.

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

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

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

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

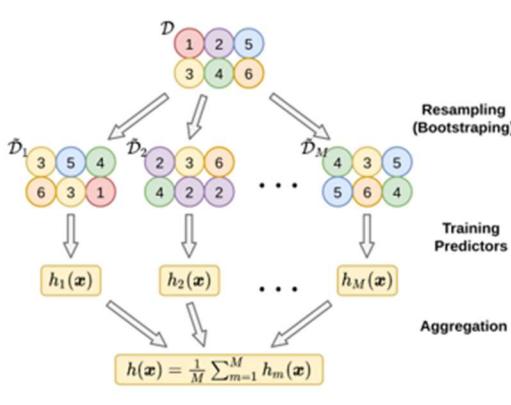
באופן פורמלי, עבור DATAה $\mathcal{D} \in \mathbb{R}^{n \times d}$, ניצור M קבוצות חדשות באותו גודל של הדadataה המקורי – עבור כל קבוצה $m \in \mathcal{X}_m \in \mathbb{R}^{n \times d}$, ועבור כל קבוצה m נבנה מודל $(x)_m$. עבור עיות סיוג ההחלטה תתקבל על פי הצבעת הרוב:

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

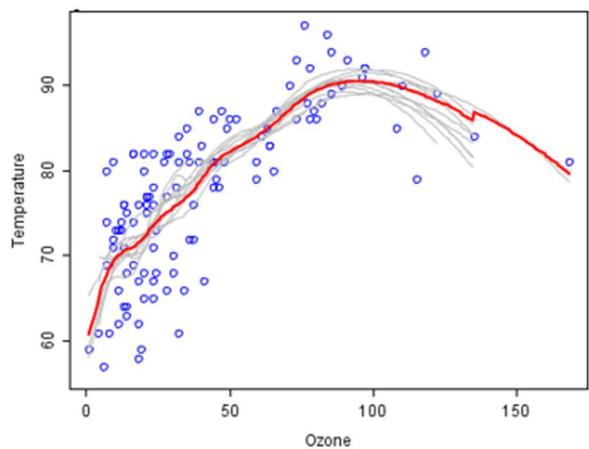
ועבור עיות גרסיה ההחלטה תבוצע בעזרת מיצוע כל המודלים:

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

a



b



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

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

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

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

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

ובסך הכל השונות הסופיות הינה בקירוב 2^{5d} , וביטוי זה לרוב קטן מאשר השונות של מודל המבוסס על הדאטה המקורי ללא שימוש ב-ensembles. ניתן לשים לב שכך שהקורסיב בין הקבוצות קטנה, כך השונות של המודל המשוקל גם כן קטנה יותר.

מודל נפוץ מאוד מסוג bagging הוא Random Forest. אלגוריתם זה משלב בין עצי החלטה לבני הרעועין הבסיסי של bagging, כאשר הוא מפצל את הנתונים ואת המשתנים לעצי החלטה רבים, וכל אחד מהם מקבל חלק מסוים מן השלים. העצים הם בעלי שנות גובהה, כלומר – כל אחד מהם הוא overfittingoso בפני עצמו, אך עם זאת הקורולציה ביניהם נמוכה, מה שמקל על הורדת השונות והימנעות מ-overfitting. לבסוף, השקלול של כל המודלים ביחיד מיציל לייצר מודל בעל שנות נמוכה, וモוביל לתוצאות טובות.

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

בנעה מלאכותית יש חשיבות רבה ליכולת הפרשנות של המודל; למשל, יידרש הסבר פחות טכני של תוצאות המודל למ啓בי החלטות או לצרכנים. הם עשויים לא לקבל כלל החלטות של מודל שיראה כ"קופסה שחורה". יש Kosher רב לתשתת פרשנות להחלטות של מודלים מבוסס bagging, והדבר מקשה על השימוש בו. מעבר לכך, bagging עשוי להיות יקר מבחינה חישובית. עקב לכך, הוא שימושי מאוד במקרים בהן שיפור זעיר עשוי להוביל להצלחה, אך לרוב תינוק עדיפות למודלים פשוטים יותר של ensembles.

2.4.3 Boosting

כאמור, המושג **boosting** מתייחס למשפחת אלגוריתמים המשתמשים באוסף של מודלים "חלשים" על מנת ליצור מודל אחד "חזק", כאשר מודלים אלו מתמקדים בניסיון להפחית את ההטיה שיש למודל. מבחינה אינטואיטיבית, מודל חלש הוא כזה שתוצאותיו מעט טובות יותר מNICOSH אקראי בעוד שאחד חזק מתקרב לביצועים אופטימליים. ביגוד לטכניקות ensemble אחרות שפועלות במקביל, העקרון המנחה כאן הוא לרשר את המודלים באופן כזה שכל מודל שמתווסף יטפל בשגיאות שקודםיו פספסו. היפוי נוצע בכר-sh-boosting מוכיח כי למידה חלשה בהכרח מובילה על קיום של שיטת למידה חזקה, לרבות, מודלים מבוססי boosting מתמקדים בעוויות סיווג ביןארו.

נמחיש את הרעיון של boosting בעזרת דוגמא: נניח שיש לנו אוסף נתונים X , המחולק באופן אקראי לשולש קבוצות שוות (כל אחת מכילה שליש מהנתונים) $-x_3, x_2, x_1$.icut בונים מודל לצורך סיווג ביןארי, המסומן ב- $-h$. נמצא כי h מתאים בצורה טוביה רק לקבוצות x_2, x_1 אך מסוויג בצורה לא טוביה את פריטי הקבוצה x_3 . כיוון ש- x_3 מכיל שליש מסך הנתונים, שגיאת הסיווג גדולה $-c_1$ הוא מודל חלש, ונרצה לשפר אותו. בצד לעשות זאת ניקח רק חלק מהנתונים, $X \in X'$, ונdag לכך ש- X' יכול הרבה יותר מאשר X .icut נבנה מודל נוסף (x_2) על בסיס X' , מתוך כוונה שמודל זה יתמקדש גם בקבוצת x_3 ויסוויג את אי-ביריה בצורה טוביה.icut נניח שמודל זה אכן מסוויג בצורה נאותה את אי-בירי x_3 , אך הפעם המודל שוגה בצורה קשה בסיווג אי-בירי x_2 .uck השגיאה בסיווג x_2 המודל השני גם הוא מודל חלש, אךicut נבנה שני מודלים חלשים שהחולשה בכל אחד נובעת מקבוצת אי-בירים אחד של אוסף הנתונים המקורי X . אם נמצא דרך הולמת לחבר את שני המסוויגים, יוכל ליצור מודל בעל פוטנציאל להצלחה ולסוויג את X כמו שצרים.

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

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

$:t = 1, \dots, T \text{ עברו } .2$

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

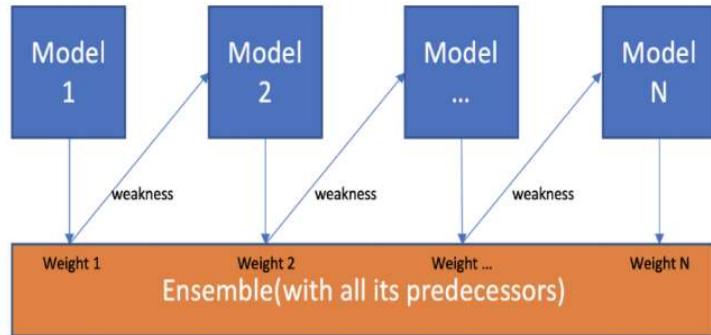
. $\epsilon_t = P_{x \sim D_t}(c_t(x) \neq f(x))$ חישוב שגיאת המסואג:

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

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

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

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



איור 2.17 – סכמת כללית של boosting. המודלים (במקרה זה מוחבר בעץ החלטה רדו-ארץ זה תקף לכל מודל חלש) מוחברים 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 שהקימות טכניקות נוספות נוספות, היא בין הפופולריות ביותר בתחום (אם כי יש לה מספק לא מבוטל של וריאנטים). העצמה הכלומה בטכניקה זו נובעת מכך שגם בהינתן מספר מאפיינים רב, האלגוריתם מצליח להיפגע פחות מ"קלחת המדדיות" ולשמור על יכולות ניבוי טובות, בניגוד לאלגוריתמים אחרים של סיוג, כמו למשל SVM או אפילו רשתות נירונים.

זכור, תחת ההנחה שהקיים אלגוריתם למודל חלש (x, c), המטרה היא למצאו דרך להפוך אותו למודל חזק (x, C). באופן אינטואיטיבי היה ניתן לחשב שאפשר פשטוט לאמן מספר מודלים על תת קבוצות של הדאטה המקורי (עם אפשרות לחיפויות בין תת-קבוצות), להשתמש ב-vote-majority, ובכך לשרשר את ההיפותזות של כל המודלים לפולט אחד. גישה זו מבוסנת נאיבית ופשטנית, ואינה לוקחת בחשבון מקרה בו מרבית המודלים 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 להוויף תחוכם לרעיון זה, כך שבכל שלב ינתן יותר דגש איברים בדאטה שהמודלים הקודמים שגו בסיווג שלהם, ובכך בכל שלב בו מאמנים מודל נוסףணף תהיה הצלחה יותר גדולה מאשר המודל הקודם. חלק זה הינו החלק האדפטיבי (Adaptive) באלגוריתם, על שמו נקרא האלגוריתם Adaboost.

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

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

בנייה מסווג אופטימלי (x, c_t) c_t ביחס לאוסף הנתונים המשווקל.

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

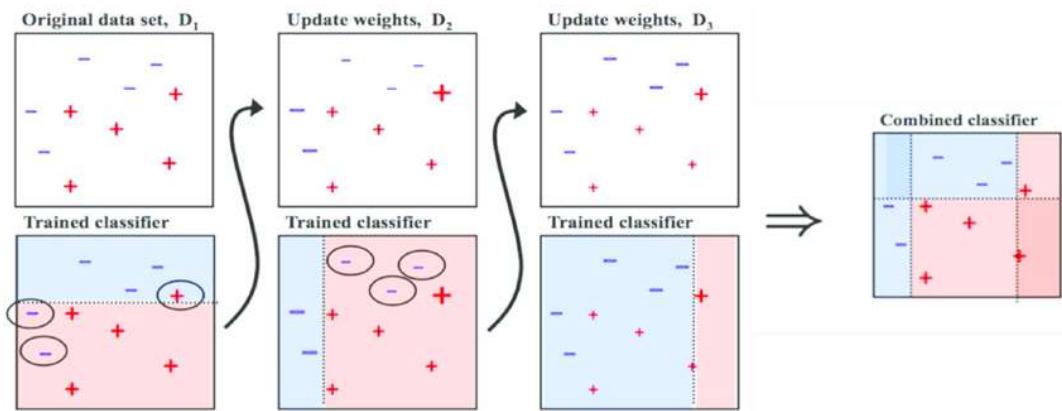
חישוב משקל עבור מסווג זה: $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

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 המשׂׂוג המשווקל, שהוא קומבינציה לינארית של המסווגים החלשים:

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



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

2. References

SVM:

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

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

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

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

Naïve Bayes:

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

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

K-NN:

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

EM:

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

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

Hierarchical Clustering:

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

LOF:

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

PCA:

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

3. Linear Neural Networks

פרק זה עוסק בבעיות רגראסיה – כיצד ניתן בעזרת אוסף דוגמאות נתון לבנות מודל המסוגל לספק מידע על נקודות חדשות שיגיעו ויחסר עליהן מידע. המודלים שיוצגו בפרק זה מתיחסים 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 לוגיסטיות (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 ממוצעת (MSE) – הממחשת 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 הלינאר הוא $b + ax = \hat{y}$. עבור המקירה זהה, פונקציית המחיר תהיה:

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

ובכדי למצוא אופטימום יש לגזר ולהשווות ל-0:

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

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

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

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

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

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

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

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

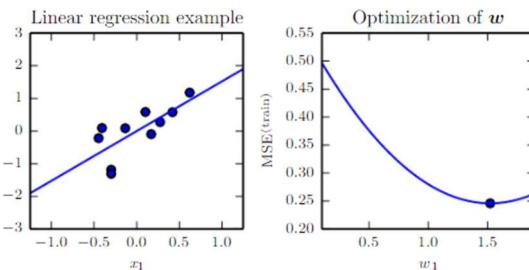
ובה ניתן אוסף דוגמאות:

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

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

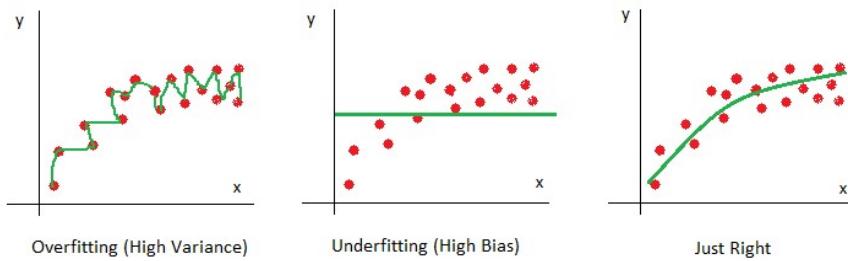
הרבה פעמים מציאת המינימום של פונקציית המחיר היא משימה קשה. דרך מקובלת להתמודד עם חישוב הפרמטר האופטימלי היא שיטת gradient descent (GD). בשיטה זו מתחילה מינוחש מסוים עבור הפרמטרים, וכל פעם מבצעים צעד לכיוון הגראדינט השילוי. הגראדינט הוא הנגזרת של הפונקציה, והוא מגדיר את הכוון שערך הפונקציה עולה בו בצורה מקסימלית. אם לוקחים את הכוון השילוי של הגראדינט, בעצם הולכים לכיוון בו יש את הירידה הכי גדולה, וכך כל פעם ל הגיעו למינימום 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 (העדרון מתבצע באופן סימולטני עבור כל θ_j):

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

קידום זה יבוצע שוב ושוב עד התכנסות לערך מסוים. כיוון שהבעיה קמורה מובטח שתהיה התכנסות למינימום, אך היא יכולה להיות איטית עקב לכך עדכונים גדולים או קטנים מדי. פרמטר ה- ϵ , learning rate, מוגדר את קצב ההתכנסות, אך רצוי לבחור פרמטר לא קטן מדי כדי לא להאט את ההתכנסות ולא גדול מדי כדי למנוע ההתכנסות.

3.1.3 Regularization and Cross Validation

אחד האתגרים המרכזיים של בעיית הרגרסיה (ושאר בעיות הלמידה) הוא לפתח מודל שייהיה מוצלח לא רק עבור אוסף הדוגמאות הידוע (אימון), אלא שייהיה מספיק טוב גם עבור דוגמאות חדשות ולא מוכחות (קבוצת מבחן). כל מודל יכול לסביר מהטיה לשוני כיוונים – Overfitting ו-Underfitting. Underfitting – מודל מציב בו יתר על נקודה בסט האימון, מה שגורר מודל מסדר גבוה בעל שונות גדולה. במצב זה המודל מתאים רק לסט האימון, אך הוא לא מצליח להסביר גם נקודות חדשות. Overfitting – מודל שלא מצליח למצואן מוגמה המכיל מספיק מידע על הדוגמאות הנוכחיות, 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 בצורה מספיק טובה את המידע (אמצע). מצב מאוזן – מודל בעל שגיאה מינימלית, המתאר בצורה טובה את המידע, ובנוסף נמנע משגיאת יתר עבור דוגמאות חדשות (ימין).

בכדי להימנע מהטויות אלו, יש לבצע regularization – regularization היא שיטת אילוץ המונע מהמודל להיות מוטה באופן הפגע בתוצאות. לאחר הוספת האילוץ, פונקציית המחיר תהיה בצורה:

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

יש מספר דרכים לבצע את ה-regularization:

Ridge Regression / L2 Regularization

דרך אחת לבצע את ה-regularization היא להוסיף איבר נוסף למתייחס לריבוע הפרמטרים:

$$L(\theta) = \text{MSE}_{\text{train}} + \lambda w^T w = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (y_i - w^T x_i)^2 + \lambda \|w\|^2$$

cutet האופטימום של הביטוי הינו:

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

הוספת האילוץ גורמת לכך שבנוסף לחיזוי מדויק של משתנה המטריה, המודל מנסה למצער את ריבוע הפרמטרים, וכך נסות 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 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 מנתנית משקל יתר לפרמטרים וגם ניסיון לאפס פרמטרים, ובכך לקבל מודל פשוט ככל הנניתן. פונקציית המחייר במקרה זה תהיה מהצורה:

$$L(\theta) = \text{MSE}_{\text{train}} + \lambda_1 \|w\|^2 + \lambda_2 |w|$$

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*: חלוקת n הדוגמאות לשיט האימון ל- k קבוצות, אימון כל תתי הקבוצות בלבד אחת, ואז בדיקת הפרמטרים שהתקבלו בשלב 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 שלמעה בכל איטרציה יש דוגמא אחת בלבד שלא נכללה בסט האימון ועליה מתבצעת הבדיקה של הפרמטרים שנלמדו.

3.1.4 Linear Regression as Classifier

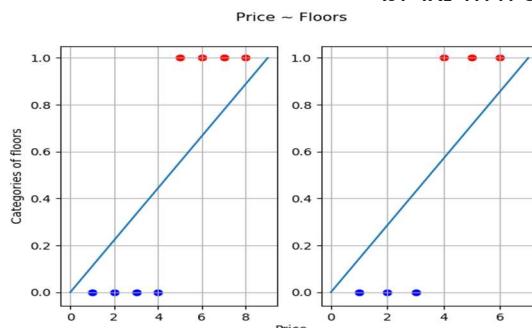
משימת סיווג מוגדרת באופן הבא: בהינתן סט פרמטרים מסוימים $\{x_1, \dots, x_n\}$ X השיר לתצפית מסוימת, יש לסתור אותו לאחת מתוך m קטגוריות אפשריות: $\{1, \dots, m\} \in y$. לדוגמה: נתונה תמונה בעלת n פיקסלים המייצגת חייה, ויש לקבוע איזו חייה היא, כאשר הבחירה נעשית מתוך הווקטור y . לרוב הווקטור y מורכב ממספרים 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}^l , $\{0,1\}$, כלומר כל נקודה במרחב ממופה לאחד משני ערכי אפסריים: קבועים ערך סף $T = 0.5$, ועבור נקודה חדשה x_n בזדוקים מה היחס בין הביטוי $b + w^T x_n < 0.5$ לבין ערך הסף. אם הנקודה החדשה מקיימת: $b + w^T x_n < 0.5$ אז הנקודה החדשה תתויג בקטgorיה 1. אחרת, הנקודה החדשה תתויג בקטgorיה 0. באופן פורמלי:

$$y = \text{sign}(w^T x_{\text{new}} + b - 0.5) = \begin{cases} 1 & w^T x_{\text{new}} + b > 0.5 \\ 0 & w^T x_{\text{new}} + b < 0.5 \end{cases}$$

הבחירה בערך הסף $T = 0.5$ נובעת מכך שיש שתי קטגוריות $\{0,1\}$, וערך הסף נקבע להיות נקודת המיצע ביניהן. בדוגמה: נתונים z בתים ועבור כל אחד מהם ידוע מה מחירו והאם יש בו קומה אחת או שתיים. כעת רוצים לבדוק את היחס בין המחיר למספר הקומות ולקבוע עבור מחיר בית נתון מה מספר הקומות שלו. במקרה זה יש 2 קטגוריות: $\{1 \text{ floor}, 2 \text{ floors}\} \in \{0,1\} = \{1 \text{ floor}, 2 \text{ floors}\}$, ויש להיעזר במידע על z הבטים בכך לבנות מודל מסווג. הדרך לעשות זאת היא לבצע רגסיה לינארית, ואז כשבוחנים מחיר של בית, יש לבדוק אם $b + w^T x \cdot \text{price} < 0.5$ או קטן ממנו, כאשר (w, b) הם הפרמטרים של הרגסיה הלינארית.



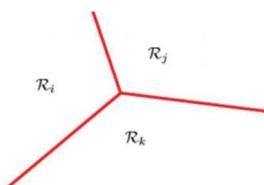
איור 3.4 רגסיה לינארית מסווג בינהר: מיפוי הנקודות בהתאם למיקומן ביחס לקו ההפרדה של הרגסיה הלינארית. בדוגמה הימנית ערך הסף מתקיים עבור $x_T = 3.5$, כלומר $3.5 + b = 0.5 \cdot 3.5 + b = 0.5$. בדוגמה השاملית ערך הסף מתקיים עבור $x_T = 4.5$. עבור כל בית חדש, בהינתן מחיר ניתן ישותו לאחת משתי הקטגוריות, בהתאם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

עבור m נקודות ידועות – $(x_1, y_1), \dots, (x_n, y_n), y_i \in \{0,1\}$, פונקציית המחיר הינה:

$$L(\theta) = \sum_{i=1}^m 1_{\{y_i \neq \text{sign}(w^T x_i + b + 0.5)\}}$$

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 כעת m קטגוריות: $\{m, \dots, 1, y_i\}$. 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 קוים 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 הקטgorיות. 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 הקטgorי, 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:

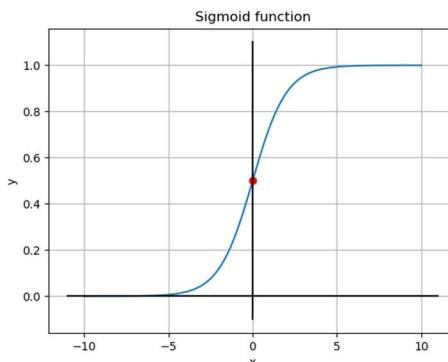
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

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

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

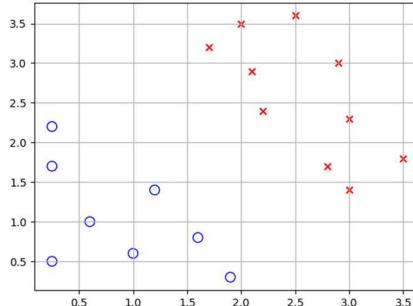
$$p(y = 0|x; \theta) = 1 - \sigma(w^T x + b) = 1 - \frac{1}{1 + e^{-(w^T x + b)}} = \frac{e^{-(w^T x + b)}}{1 + e^{-(w^T x + b)}}$$

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 הנקודה 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 הנקודה x_n להיות משוכנת לקטגוריה 1 קטנה מחצי. כתוב עולה השאלה מתי $w^T x + b = 0$, והתשובה היא שזה תלוי בכך 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 שנ נתונים 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 x 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 $w^T x + b = 0$, ולכן אם תהיה נקודה חדשה שางם מקיימת $w^T x_n + b = 0$, המשמעות היא שנקודה זו נמצאת בבדיקה על קו ההפרדה. נקודה כזו תקבל הסתברות של 50% להיות משוכנת לכל אחת מהקטגוריות. ככל שהנקודה החדשה תתרחק מקו ההפרדה, כך הביטוי $w^T x + b$ יתרחק מה-0, וכך גם $w^T x + b$ יתקרב לאחד מערכי הקצה 0 או 1, והמשמעות היא כמפורט.

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 לקטgorיה בעלת ההסתברות הגבוהה ביותר. במקרה הבינארי וקטור ההסתברויות הינו $[p(y=1|x), p(y=0|x)] = [\hat{y}, 1-\hat{y}]$, והקטgorיה של \hat{y} תהיה שזהו ה- \hat{y} בעל ההסתברות הגדולה ביותר.

3.2.2 Cross Entropy and Gradient descent

בכדי למצוא את הפרמטרים $(b, w) = \theta$ האופטימליים בהינתן n דוגמאות, ניתן להחליף את קרייטריון השגיאה הריבועית הממוצעת בקריטריון אחר למצער פונקציית המחיר – Cross entropy. קרייטריון זה אומר שיש לחייב למינימום 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:

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

זכור, $\sigma(z) = p(y=1|x; \theta)$, ולכן שלחשב גם את הנגזרת עבור $(z) = 1 - \sigma(z)$:

$$\frac{\partial (1 - \sigma(z))}{\partial z} = -\sigma(z)(1 - \sigma(z))$$

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 $\log p(y=1|z) = 1 - \sigma(z) = y - \sigma(z)$ הינה $\frac{\partial}{\partial z} \log p(y_i|z) = y_i - \sigma(z)$. לכן אם $y \in \{0,1\}$, אז ניתן לרשום בקיצור: $\sigma(z) = y_i - \sigma(z)$. במקרה של רגرسיה לוגיסטיבית, 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 לרוגרסיה לינארית, גם כאן חישוב הערך האופטימלי של $\hat{\theta}$ יהיה איטרטיבי בשיטת

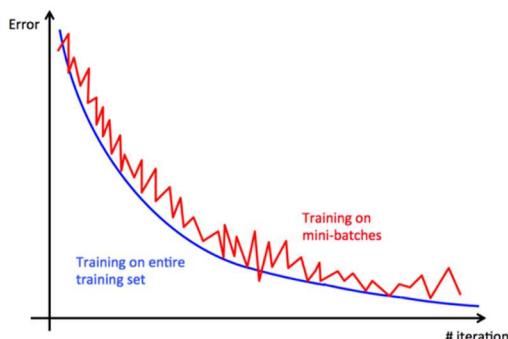
$$\hat{\theta}_{j+1} = \hat{\theta}_j - \epsilon \cdot \frac{\partial}{\partial \theta_j} L(\theta)$$

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

במקרים רבים הدادה סט הוא גדול, ולחשב את הגרדיינט עבור כל הدادה צריכה הרבה חישוב. בכל צעד של קידום ניתן לחשב את הגרדיינט עבור חלק מהدادה, וביצוע את הקידום לפי הכוון של הגרדיינט הנוכחי. למשל ניתן לבחור באופן אקראי נקודה אחת ולחשב עליה את הגרדיינט. בחירה זו נקראת Stochastic Gradient Descent (SGD), כיוון שבכל צעד יש בחירה אקראיית של נקודה. חישוב בשיטת SGD יכול לגרום לשונות גדולות מתקדם, ולכן עדיף 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 מתקבל 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 להרחיב 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 $L(\theta)$:

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

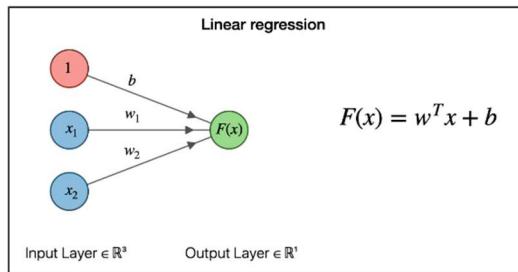
כעת ניתן לחשב את θ האופטימלי בשיטת gradient descent

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

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 טוביה עבור בעיות מורכבות. יש מגוון בעיות בהן על מנת לבנות מודל המסוגל להפריד בין קטגוריות שונות, יש צורך במנגנון הפרדה לא לינארית.

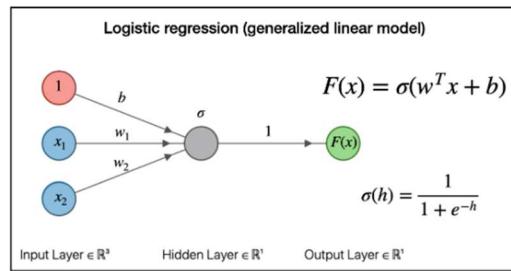
דרך מקובלת לבניית מודלים לא לינאריים היא שימוש ברשות נוירונים עמוקות, ובכדי להבין את הקונספט של זה היטב, ראשית יש ליזג את המודלים הלינאריים שכבה של נוירונים, כאשר המודל הזה שקול לחולוטן לכל מה שהוזג עד כה. בעיית Linear regression לוחצת סט של מאפיינים ומכפילה כל אחד מהם במשקל, ולאחר מכן סוכמת את כל האלמנטים (בצירוף bias) לכדי משתנה יחיד הקובל מה הקטgorיה של סט זה. ניתן ליזג את המודל על ידי התיאור הגרפי הבא:



איור 3.9 ייזוג 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 ייזוג רגסיה לוגיסטיבית כרשת נוירונים עם שכבה אחת.

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

בפרק הבא ייזג 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.

3. References

<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/>

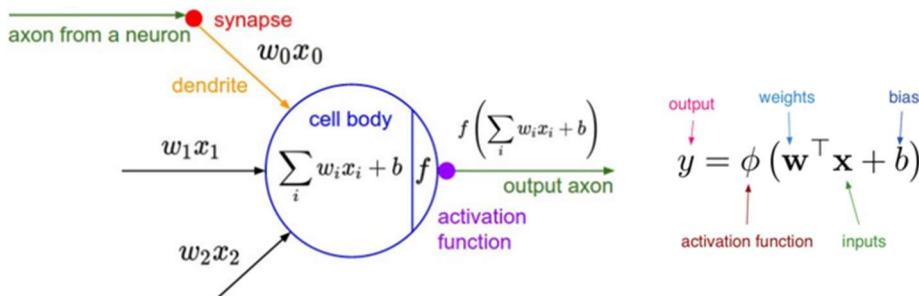
4. Deep Neural Networks

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 הדטה 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:

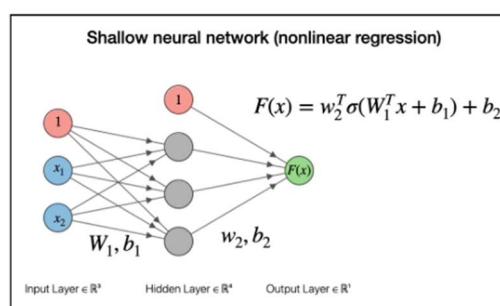


איור 4.1 יציג של נירון מלאכותי, המתקבל קלות, סוכם אותו וublisher את התוצאה בפונקציית הפעלה.

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

במקרה בו הנירונים המוחברים ל-*שעוטו* אינם מהווים את המוצא אלא הם מוכפלים במשקלים ומחברים לשכבה נוספת נירונית, אז השכבה המוחברת ל-*שעוטו* 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 הכוונה למצוא אינו לינארי, וזה הינו שיס למודל זה. נתבונן במקרה של שכבה חבוייה וnochesh את הקשר בין הכוונה למצוא: נסמן את המשקלים בין הכוונה לבין השכבה החבוייה ב- b_1, w_1 ואת המשקלים בין השכבה החבוייה לבין המוצא ב- b_2, w_2 , וכן את השכבה החבוייה מתיקל הביטוי: $y = f_1(w_1^T x + b_1) + b_2$. ביטוי זה עובר בפונקציית הפעלה נוספת ומתקבל המוצא:

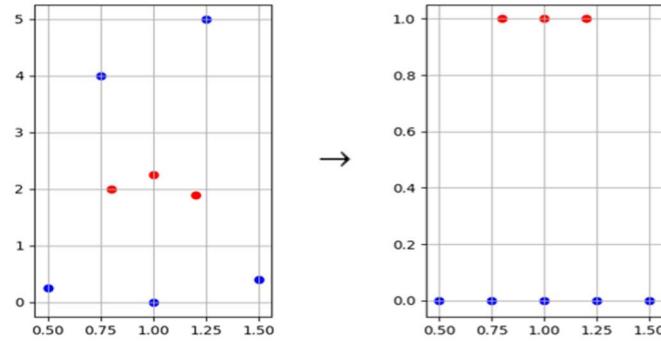
$$\hat{y} = f_2(w_2 \cdot f_1(w_1^T x + b_1) + b_2)$$



איור 4.2 רשת נירונים בעלת שכבה חבוייה אחת.

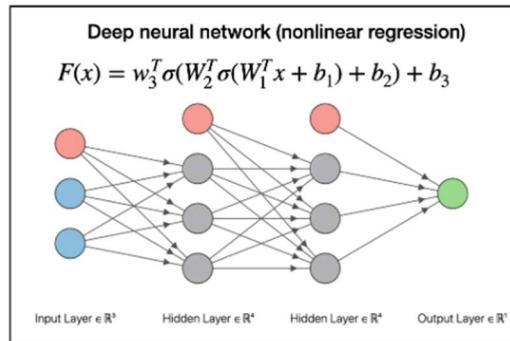
חשוב להזכיר שמטרת הרשת היא לבצע פעולות לא לינאריות על ה-*שעוטו* כך שהיא יסודר באופן חדש הניתן להפרדה לינארית. למעשה מבדילים את הפרדה הלינארית הנעשה באמצעות הרוגסיה, אלא מבצעים פניה שלב מקדים של העתקה לא לינארית. תהילך זה נקרא למדידת "יצוגים" (representation learning), כאשר בכל שכבה

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.

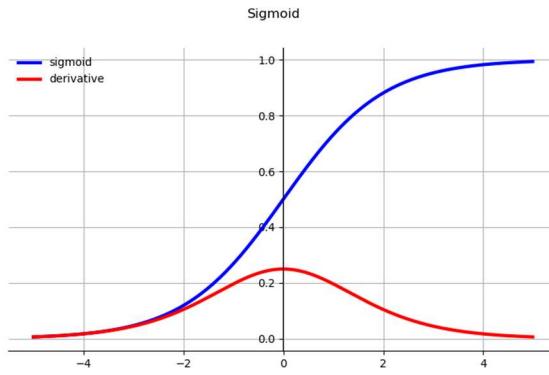
4.1.2 Activation Function

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

Sigmoid

פונקציית הסיגמודיאד 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.

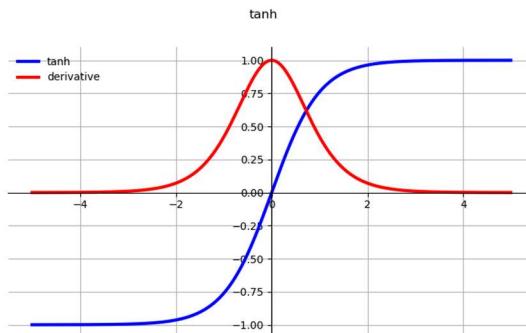
יש לפונקציה זו שלושה חסרונות:

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

tanh

פונקציית טנגנס היפרבולי הינה מהצורה:

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



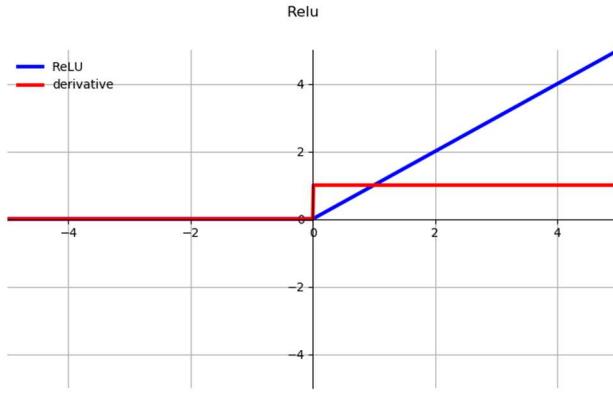
איור 4.6 פונקציית טנגנס היפרבולי והנגזרת שלה.

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ישה 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. לבסוף, בפונקציה זו הגרדיינט לא מתאפשר בערכים גבוהים. יתרון נוסף שיש לפונקציה זו – היא מתכנסת יותר מהפונקציות הקודמות (αx). 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 LU ו-PReLU:

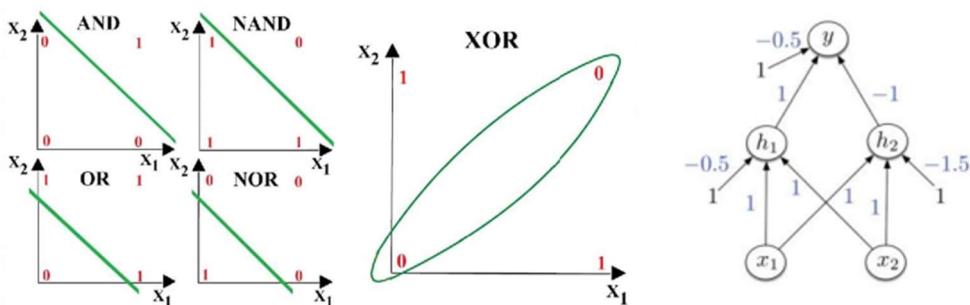
$$PReLU(z) = \max(\alpha z, z), ELU(z) = \begin{cases} z & z > 0 \\ \alpha(e^z - 1) & z \leq 0 \end{cases}$$

בפונקציית PReLU, המקירה הפרטி בו $\alpha = 0.1$ נקרא Leaky ReLU. בפונקציית ELU, הפרמטר α הוא פרמטר נלמד.

ישנן עוד פונקציות, אך אלה הן העיקריות, כאשר לרוב מקובל להשתמש בReLU ובורותיות שלו.

4.1.3 Xor

אחת הדוגמאות הידועות ביותר שאין ניתנות להפרדה לינארית היא בעיית ה-XOR. יש שתי כניסה - x_1, x_2 והמוצא הוא 0 אם הכניסות שוות ו-1 אם הן שונות. פונקציה זו מיפה שתי כניסה ליציאה, כאשר יש שתי קטגוריות 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 ליצור מודל פשוט לאופרטור XOR.

בדוגמא המובאת באյור הכניסות עוברות דרך שכבה חבויה אחת בעלי שתי נוירונים - h_1, h_2 , המקבלים בנוסף גם bias. פונקציית הפעלה של נוירונים אלו היא פונקציית הסימן, ונitin לכתוב את המוצא של שכבה זו כך:

$$h_1 = sign(x_1 + x_2 - 0.5), h_2 = sign(x_1 + x_2 - 1.5)$$

לאחר השכבה החבויה הנוירונים מחוברים למוצא, שגם לו יש bias, והסכום של הכניסות והbias עוביים במסווג:

$$y = sign(h_1 - h_2 - 0.5) = \begin{cases} 1 & if h_1 - h_2 - 0.5 > 0 \\ 0 & if h_1 - h_2 - 0.5 < 0 \end{cases}$$

נבחן את המשמעות של הנוירונים: הנירון h_1 יהיה 0 אם שתי הכניסות שוות 0, אחרת הוא יהיה שווה 1. הנירון 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.

התליך האימון מתבצע בשני שלבים – ראשית מכנים דוגמא ידועה לתחילת הרשות ו"疎傳" (Forward propagation) (Forward propagation), כלומר, מחשבים את השינוי שהיא עוברת כאשר היא מוכפלת במסקלים וועוררת בנוירונים החבויים. לאחר שמגיעים למקום, משווים את מה שהתקבל למה שאמור להיות במצבו לפי מה שידוע על דוגמא זו, ועוד מבצעים פעוף לאחר מכן (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 (L), ועוד מבצעים עדכון בשיטת $\frac{\partial L}{\partial w_i} \epsilon - w_{i+1} = w_i$. כיון שהרשות יכולה להכיל מיליאדיים של מסקלים, יש למצאו דרך יעילה לחישוב הגרדיאנט עבור כל מסקל.

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

4.2.2 Forward and Backward propagation

באופן פורמלי, עבור N מסקלים התהליך מנוטה כך:

Forward pass:

For i in 1 ... N:

Compute w_i as function of $w_0 \dots w_{i-1}$

Backward pass:

$$\overline{w_N} = 1$$

For i in N - 1 ... 1:

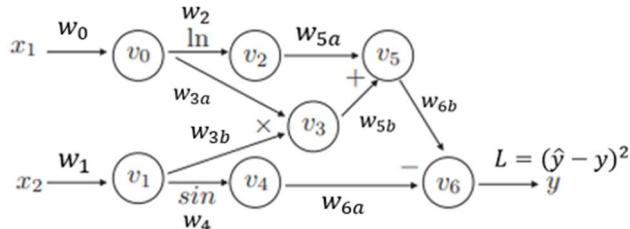
$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial w_N} \cdot \frac{\partial w}{\partial w_{N-1}} \dots \frac{\partial w_{i+1}}{\partial w_i}$$

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

בשלב הראשון מחשבים כל צומת על סמך הצמתים 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 הממשקלים בשיטת backpropagation הוכיח את עצמו, למרות שהמשקלים לא בהכרח הגיעו לאופטימום שלהם.

4.3 Optimization

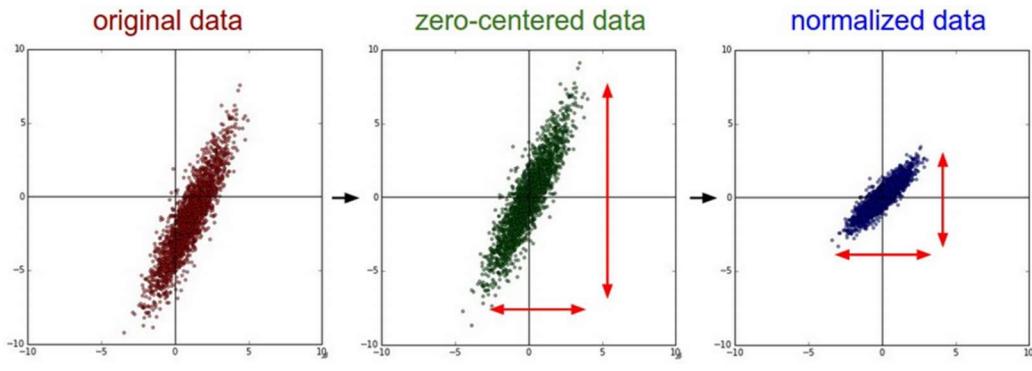
מציאת אופטימום למשקלים על פני כל העומק של הרשת היא בעיה לא קמורה, ולכן אין לה בהכרח מינימום גלובלי. לכן מלבד עדכון הממשקלים בשיטת backpropagation יש לבצע אופטימיזציות נוספת על הרשת על מנת לשפר את הביצועים שלה.

4.3.1 Data Normalization

חלק מפונקציות הפעולה אין ממורכבות סביר-0, ועבור ערכים גבוהים הן קבועות בקריב גראדיינט בערכיהם אלו מטאפס, דבר שאינו אפשר לעדכן את המשקלים בשיטת GD. כדי להימנע מהגעה לתחום ה"רויה" בו הגראדיינט מטאפס, ניתן לנורמל את הדadata כך שהיא בעל תוחלת 0 ושונות 1, ובכך הוא יהיה ממורכב סביר-0:

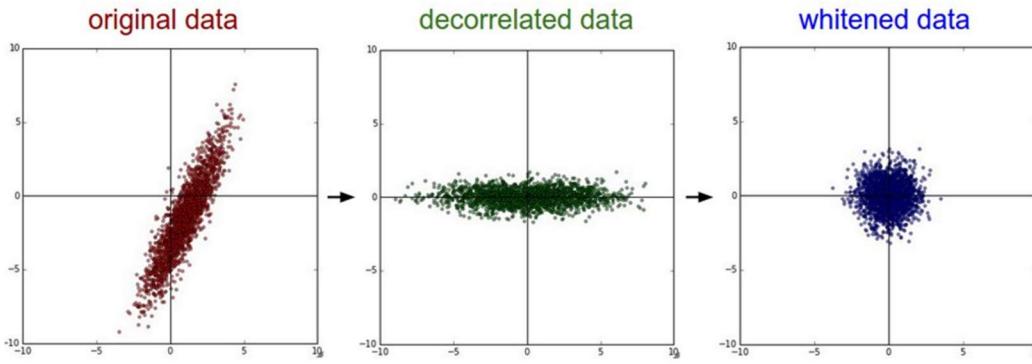
$$X_i = \frac{X_i - \mu_i}{\sigma_i}$$

ובאופן חזותי:



איור 4.10 נרמול>Data בשלבים – איפוס התוחלת (ירוק) ונרמול השונות -1 (כחול).

שלב זה הוא למעשה שלב pre-processing הנועד להcin את הדadata לפני כניסה לרשת, כדי לשפר את האימון הרשת. ישנו אופנים נוספים לנרמול את הדadata – ללקסן את מטריצת covariance של הדadata או להפוך אותה למטריצת היחידה:



איור 4.11 דרכים נוספים לנרמל את הדadata – ללקסן את מטריצת covariance (ירוק) או להפוך אותה למטריצת היחידה (כחול).

4.3.2 Weight Initialization

ענין נוסף שיכול להשפיע על האימון וניתן להתייחס אליו עוד בשלב ה-pre-processing הוא אתחול המשקלים. אם כל המשקלים מאותחלים ב-0, אז המוצא וכל הגראדיינטים יהיו גם כן 0, ולא יבוצע עדכון למשקלים. לכן יש לבחור את המשקלים ההתחלתיים בצורה מושכלת, למשל, להגריל אותם מהתפלגות מסוימת שתאפשר אימון טוב של הרשת.

אפשרות אחת להגריל היא עבור כל משקל ערך קטן מהתפלגות נורמלית עם שונות קטנה – $(\alpha, 0, N)$, כאשר $\alpha = 0.01$ or 0.1 .

בערכיים קטנים גורם לאיפוס הגרדייאנט מהר מדי. כדי להתמודד עם בעיה זו, ניתן לבחור $\alpha = 1$, אך זה יכול לגרום להתקדרות הגרדייאנט. שיטה נוספת יותר נקראת Xavier Initialization, הלוקחת בחשבון את הגודל של השכבות – האתחול יבוצע באמצעות התפלגות נורמלית, אך השונות לא תהיה מספר ללא משמעות, אלא תהיה תלולה במספר השכבות – $\frac{1}{\sqrt{n}}$. שיטה זו טובה גם לרשות עם הרבה שכבות, אך היא בעייתית במקרים בו פונקציית הפעלה הינה ReLU, כיוון שהאתחול מניח שפונקציית הפעלה ממורכזת סביב 0 (כמו למשל \tanh). כדי לאפשר גמישות גם מבחינת פונקציית הפעלה, ניתן לבחור $\alpha = \sqrt{\frac{2}{n}}$, ואז האתחול יתאים גם ל-ReLU.

Xavier-Initialization היא להಗיל מהתפלגות איחודית, כאשר באופן דומה ל- ReLU גם גבולות יהיו תלויים בגודל השכבות – $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right] U$.

4.3.3 Batch Normalization

כאשר מביצעים normalization, למשה דוגמים לכך שבכינסה לרשת הדטה היה מנורמל סביב ה-0. באופן זה נמנעים מהגעה לUMB משב בו יש ערכים גבוהים בעומק הרשת, הגרומים להתאפסות או להתקדרות של הגרדייאנט. בפועל, הנרמול הזה לא תמיד מספיק טוב עבור כל השכבות, ואחרי כמה שכבות של הכפלה במשקלים ומעבר בפונקציות הפעלה הרבה פעמים מתקבלים ערכים גבוהים. באופן דומה ל- ReLU normalization המבוצע לפני האימון, ניתן תוך כדי האימון לבצע normalization שודרג לנרמול הערכים שנכנסים לנירונים בשכבות החבויות. התהליך נעשה בשלושה שלבים:

- א. עברו כל ניירון בעל פונקציית הפעלה לא לינארית, מחשבים את התוחלת והשונות של כל הערכים היוצאים ממנו.
 - ב. מנורמלים את כל היציאות – מחסרים מכל יציאה את התוחלת ומחלקים את התוצאה בשונות (בתוספת אפסיון, כדי להימנע מחילוקה ב-0).
 - ג. הנרמול יכול לגורם לאיבוד מידע, לכן מבצעים לתוצאה המנורמלת scale and shift – השזה ושינוי קנה המידה. התיקון מ被执行 בעזרת פרמטרים נלמדים.
- עבור שכבות גדולות חישוב התוחלת והשונות יקר כיוון שלמיירון יש הרבה יציאות, אך לוקחים רק חלק מהיציאות – Mini Batch: $\mathcal{B} = \{x_1 \dots m\}$.

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

$$\begin{aligned}\mu_{\mathcal{B}} &= \frac{1}{m} \sum_{i=1}^m x_i, \quad \sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \\ \hat{x}_i &= \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \\ y_i &= \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)\end{aligned}$$

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

בשלב המבחן, השונות והתוחלת שבעזרתם מביצעים את הנרמול אין נלקחים מהיציאות של הנירונים, אלא לוקחים ממוצע של כמה מה- Mini Batch האחרונים.

יש כמה יתרונות לשימוש ב- BN : האימון נעשה מהר יותר, יש פחות ריגשות לאתחול של המשקלים, אפשר שימוש ב-rate learning גדול יותר (מוני מהגרדייאנט להתקדר או להתאפס), אפשר שימוש במוגן פונקציות הפעלה (אם אלה שאין ממורכזות סביב 0) ומספק באופן חלקי גם רגולריזציה (שונות נמוכה במקרה).

4.3.4 Mini Batch

במקרים רבים הדטה-ט גודל, ולחשב את הגרדייאנט עבור כל הדטה נדרש הרבה חישוב. בכל צעד של קידום ניתן לחשב את הגרדייאנט עבור חלק מהדטה, ובוצעו את הקידום לפי הכוון של הגרדייאנט המתפרק. למשל, ניתן לבחור באופן אקראי נקודה אחת ולחשב עליה את הגרדייאנט. 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 מתќדם, ולכן כל ממצאים mini-batch learning – חישוב הגרדיאנט על חלק מהדטה. באופן זה גם יש הפחטה של כמות החישובים, וגם אין שנות גבואה. אם ממצאים את החישוב בשיטה זו יש לדאוג שהדטה מעורבת כדי שהמשקלים אכן יתעדכו בצורה נכונה, ובנוסף שה-mini-batch יהיה מספיק גדול כך שהיא בו יציג לכל הדטה. כל מעבר על פני כל הדטה-ט נקרא Epoch (אם הדטה הוא בגודל N, והגודל של כל-mini-batch הוא S, אז כל Epoch הוא S/N איטרציות).

אמנם כל צעד הוא קירוב לגרדיאנט, אך החישוב מאד מהיר ביחס לגרדיאנט המדויק, וזה יתרון משמעותי שיש לשיטה זו על פני batch learning. בנוסף, המשקלים שמתќבלים קרובים מאד לאלו שהיו מתקבלים באמצעות 3.8. batch learning

4.3.5 Gradient Descent Optimization Algorithms

בשיטת GD, עדכון המשקלים בכל צעד הוא: $w_{i+1} = w_i - \epsilon \frac{\partial L}{\partial w}$, כאשר ϵ הוא פרמטר שנקרא (lr) Learning Rate והוא קבוע עד כמה יש לשנות המשקל בכיוון הגרדיאנט. בגין דמיון רגסיה, אופטימיצית רשת תרונים היא לרוב בעיה שאינה קמורה, שכן לא מוצעת התכנסות למינימום הגלובלי. משום כך, אם בכלל צעד הולכים יותר מדי לכיוון הגרדיאנט השילי, ניתן להתכנס לנקודות אוכף או למינימום לוקאלי שהוא אינו בהכרח המינימום הגלובלי. מצד שני אם מתќדים מעט מדי לכיוון הגרדיאנט, המשקל בקצבו מתקדם. פרמטר ה- $-\epsilon$ נדרש להתגבר על בעיות אלו, שכן צריך שהוא לא יהיה גדול מדי (אחרת תהיה התובדות של המשקלים או התכנסות למינימום לוקאלי) ולאחר מכן קטן מידי (אחרת לא תהיה התקדמות או שהיא תהיה מאוד איטית). כיוון שאין ערך אבסולוטי שמתאים לכל הבעיה, יש מגוון שיטות המנסות למצוא את העדכון האופטימלי בכל צעד. יש שיטות משתמשות בפרמטר משתנה – lr, ויש שיטות שימושיפות פרמטרים אחרים לביטוי של העדכון.

Momentum

ישנו מצבים בהם יש כל מיני פיתולים בדרך לנקודות מינימום. במקרה זה, בכל צעד הגרדיאנט יפנה לכיוון אחר, וההתכנסות לנקודות מינימום תהיה איטית. הדבר דומה לנחל שזורם לים, אך הוא לא זורם ישר אלא יש לו הרבה פיתולים. כדי להאיץ את ההתכנסות במקרה זה, ניתן לנסות לבדוק את הכוון הכללי של הגרדיאנט על סמך כמה צעדים, ולהוסיף התקדמות גם לכיוון זהה. שיטה זו נקראת מומנטום, כיון שהיא מחפש את המומנטום הכללי של הגרדיאנט. החישוב של המומנטום מתבצע בהתאם רקורסיבית:

$$m_{i+1} = \mu m_i - \epsilon \frac{\partial L}{\partial w}$$

ואז העדכון הינו:

$$w_{i+1} = w_i + m_{i+1}$$

הפרמטר μ הינו פרמטר דעיכה עם ערך טיפוסי 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^n ילך ויקטן, וכך תהיה פחות השפעה לצעדים שכבר היו לפני הרבה עדכנים. תחת הנחה שהgradient זהה לכל הפרמטרים, ניתן לפתח נוסחה סגורה לרכיבי:

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

הביטוי שמתќבל הוא סדרה הנדסית מתכנסת, ובສוף הכל מתќבל הביטוי:

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

היעילות של המומנטום תלויות בבעיה – לעיתים היא מאייצה את ההתכנסות ולפעמים כמעט ואין לה השפעה, אך היא לא יכולה להזיז.

וריאציה של שיטת המומנטום נקראת Nesterov Momentum. בשיטה זו לא מחשבים את הגרדיינט על הצעד הקודם, אלא על המומנטום הקודם:

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

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

שיטה זו עובדת טוב יותר מאשר בUIKitות קmorות, כלומר היא מצליחה להתכנס יותר טוב מאשר המומנטום הרגיל, אך היא איטית יותר.

learning decay

באימון רשותות עמוקות בדרך כלל כדאי להקטין את ה- ϵ עם הזמן. הסיבה לכך היא שיכל שמתקדמים לכיוון המינימום, יש צורך בצעדים יותר קטנים כדי להצליח להתכנס אליו ולא לחוץ מסביבו מצד לצד. עם זאת, קשה לקבוע כיצד לבדוק להקטין את ה- ϵ : הקטנה מהירה שלו תימנע הגעה לאזור של המינימום, והקטנה איטית שלו לא תעזר להתכנס למינימום כאשר מגעים לאזור שלו. ישנו שלושה סוגים נפוצים של שינוי הפרמטר:

א. שינוי הפרמטר בכל כמה Epochs. מספרים טיפוסיים הם הקטנה בחצי כל 5 epochs או חלוקה ב-10 כל 20 epochs. באופן כללי ניתן לומר שכאשר גרפ' הלמידה של ה-validation משתפר, יש להקטין את ה- ϵ .

ב. דעיכה אקספוננציאלית של ה- ϵ : $\epsilon_0 e^{-kt}$, כאשר k הם היפר-פרמטרים, ו- t יכול להיות צעד או epoch.

ג. דעיכה לפי $1/t$: $\epsilon = \frac{\epsilon_0}{1+kt}$, כאשר k הם היפר-פרמטרים, ו- t הינו צעד של עדכון.

Adagrad and RMSprop

בעוד השיטה הקודמת מעדכנת את ה- ϵ בצורה קבועה מראש, ניתן לשנות אותו גם באופן מסתגל לפי ההתקדמות בכיוון הגרדיינט. בכל צעד ניתן לבדוק עד כמה גדול היה השינוי בצעדים הקודמים, ובהתאם לכך אפשר לחתה ϵ מתאים, מתוך מגמה להקטין אותו ככל שמתקדמים לכיוון המינימום. באופן פורמלי, אלגוריתם Adagrad מוגדר כך:

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

כאשר ϵ הוא מספר קטן הנועד למנוע חלקה ב-0. כיוון ש- α הולך וגדל, הביטוי $\frac{\epsilon}{\sqrt{\alpha_i + \epsilon_0}}$ הולך וקטן, וקצב הדעיכה הוא ביחס ישיר לקצב ההתקדמות בכיוון הגרדיינט. בכך מרווחים דעיכה של ה- ϵ , בקצב המשתנה לפי ההתקדמות. באופן ייחודי, הדעיכה של ה- ϵ מחרירה, כיוון שהסכום $\sum_{j=1}^i \left(\frac{\partial L}{\partial w_j} \right)^2$ גדל במהירות. כדי להאט את קצב הדעיכה יש שיטות בהן נתונים יותר משקל לצעדים האחרונים ופחות לצעדים שכבר עברו זמן. השיטה הפופולרית נקראת RMSprop, ובשיטה זו במקומם לסכום את ריבוע הגרדיינט של כל הצעדים הקודמים באופן שווה, מבצעים moving average, וככל שעברו יותר צעדים מוצאים עד צעד הנוכחי, כך תהיה לו פחות השפעה על דעיכת ה- ϵ :

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

Adam

ניתן לשלב בין הרעיון של מומנטום לבין adaptive learning rate:

$$\alpha_i = \beta_1 \alpha_{i-1} + (1 - \beta_1) \left(\frac{\partial L}{\partial w} \right)^2, m_i = \beta_2 m_{i-1} + (1 - \beta_2) \frac{\partial L}{\partial w}$$

$$\hat{\alpha}_i = \frac{\alpha_i}{1 - \beta_1^i} \hat{m}_i = \frac{m_i}{1 - \beta_2^i}$$

$$w_{i+1} = w_i - \frac{\epsilon}{\sqrt{\hat{a}_i + \epsilon}} \hat{m}_i$$

מספרים טיפוסיים: 5^{-4} or $\epsilon = 10^{-2}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$. האלגוריתם למעשה גם מוסיף התקדמות בכיוון המומנטום (הכיוון הכללי של הגרדיאנט), וגם מביא לדעיכה אדפטיבית של ה- β -ים. זה האלגוריתם הכי פופולרי ברטשות עמוקות, אך הוא לא מושלם ויש לו שתי בעיות עיקריות: האימון הראשון לא יציב, כיון שבתחלת האימון יש מעט נקודות לחישוב הממוצע עבור t . בנוסף, המודל המתkeletal נוטה ל-overfitting עם מומנטום.

יש הרבה וריאציות חדשות על בסיס Adam שנעדו להתגבר על בעיות אלו. ניתן למשל להתחילה לאמן בקצב נמוך, וכאשר המודל מתגבר על בעיית ההתייצבות הראשונית, להגבר את הקצב (ups-warm). במקביל, ניתן להתחילה עם Adam ולהחליף ל-SGD כאשר קритריונים מסוימים מתקיימים. כך ניתן לנצל את התכונות המהירה של Adam בתחלת האימון, ואת יכולת ההכללה של SGD.

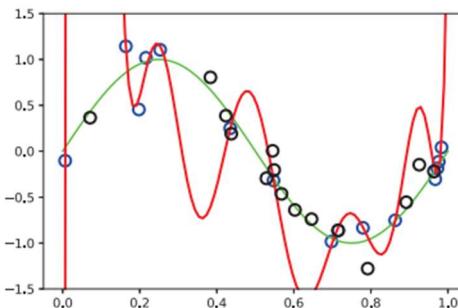
4.4 Generalization

כל מודל שנבנהנסמך על דатаה קיימ, מ透ת מגמה שהמודל יתאים גם לדאטה חדש. לכן יש חשיבות גדולה שהמודל ידע להקליל כמה שיותר טוב, על מנת שהוא יתאים בצורה טובה לא רק לדאטה המקוריים אלא גם לדאטה חדש. במקרה אחד, יש לוודא שהמודל לא מצליח את הפרמטרים שלו רק לדוגמאות שהוא רואה, אלא שינסה להבין מ透ת הדוגמאות מה החקיקות הכללית, שמתאיימה גם לדוגמאות אחרות.

4.4.1 Regularization

כפי שהסביר בפרק 3.1.3, מודל יכול לסבול מהטייה לשני ציוונים – Overfitting ו-underfitting. מובל בונתת הערכת יתר לכל נקודה בסט האימון, מה שגורר מודל מסדר גבוה בעל שונות גדולות. במצב זה המודל מתאים רק לסט האימון, אך הוא לא מצליח להקליל גם נקודות חדשות. Underfitting הוא מצב ההפוך – מודל שלא מצליח למצאו קן מגמה המכיל מספיק מידע על הדוגמאות הנתונות, ויש לו רושץ.

ברשת נוירונים, ככל שמספר הפרמטרים גדול, כך השגיאה של ה-test training קטנה. לגבי ה-test training קטנה. עד נקודה מסוימת, ושם היא גדלה בחזרה. בהתחלה השגיאה יודדת כיון שמצליחים לבנות מודל יותר מדויק ונמנעים מ-underfitting, אך בנקודה מסוימת יש יותר מדי פרמטרים והם נהנים מותאמים יותר מידי לסט האימון. overfitting. למעשה צרכי למצוא את היחס הנכון בין מספר הפרמטרים (סדר המודל) לבין גודל הדאטה. כיון שאין אפשר לזהות Overfitting בלבד, שחרי ה-Loss-Ktan ככל שיש יותר פרמטרים, ניתן לחלק את הדאטה לשני חלקים – training and validation. בשלב ראשון בונים מודל בהינתן ה-training, ולאחר מכן בוחנים את המודל על ה-validation – אם המודל לא מתאים ל-validation – סימן שיש overfitting. ככל המודל מתאים רק לדוגמאות שהוא רואה והוא נתן להם הערכת יתר. ככל שהגרף של ה-validation accuracy קרוב יותר ל-training accuracy, כך יש פחות overfitting.



איור 4.12 בדיקת overfitting set validation. הנקודות הכהולות שייכות ל-training והשחורות שייכות ל-validation. המודל האדום מתאים רק לנקודות הכהולות, אך אפשר לומר שהוא נוטה ל-overfitting. המודל הירוק לעומת זאת מתאים גם לנקודות השחורות, אותן הוא לא ראה בשלב האימון, ככל שהוא הצליח להקליל טוב גם לדוגמאות חדשות.

האפשרויות היכי פשוטה להימנע מ-overfitting היא פשוט להוריד פרמטרים, ככלומר להקטין את גודל הרשת. בנוסף ניתן לבצע Early stopping – לחשב בכל Epoch את גרפ Epoch של ה-Loss-Shall ה-validation, וכאשר הוא מתחילה לעלות להפסיק את האימון. שיטות אלה פשוטות מאוד לישום, אך ישן שיטות אחרות שמספקות ביצועים יותר טובים, ובוחן אותן כעת.

4.4.2 Weight Decay

בדומה לרגולריזציה של linear regression, גם ברשת נוירונים ניתן להוסיף איבר ריבועי לפונקציית המבחן, מה שמכונה L2 Regularization:

$$Cost(w; x, y) = L(w; x, y) + \frac{\lambda}{2} \|w\|^2$$

ההוספה של הביטוי האחורי דואגת לכך שהמשקל לא יהיה גדול מדי, שהרי רצים למזער את פונקציית המבחן, שכן לאף לכך שהביטוי הריבועי יהיה כמו שייתר קטן. בתוספת האיבר ערךן של המשקלים יהיה:

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

הביטוי הזה דומה מאוד ל-GD רגיל, כאשר נוסף איבר של λ . אם $\lambda < 0$, אז ללא קשר לגראדיינט המשקל יורד בכל צעד, וזה נקרא "Weight decay".

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

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

ואז העדכון יהיה:

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

בעוד L2 regularization מושך יחיד ונiska להקטין אותו, ניתן לבנות דוגמאות אחרות למודלים להתקפס ולדילול מספר הפרמטרים של הרשת.

4.4.3 Model Ensembles and Drop Out

עבור דadata קיימן ניתן לבנות מספר מודלים, ואז כשבאים לבחון דатаה חדש בודקים אותו על כל המודלים ולוקחים את המוצע. סט המודלים נקרא ensemble. ניתן לבנות מודלים שונים במספר דרכים:

א. לאמן רשות עם אתחולים שונים למשקלים.

ב. לאמת מספר רשותות על חלקים שונים של הדטה.

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 ליצורensemble – באמצעות Dropout. ככלומר למחוק באופן אקראי נוירון אחד או יותר. אם יש רשות מסוימת ומוחקים את אחד הנוירונים – למעשה מתקבלים רשות אחרת, ובפועל אפשר לקבלensemble בעזרת רשות אחת בלבד פעמיים מוחקים ממנו נוירון אחד או יותר. היתרון של יצירת ensemble בדרך זו הוא שההרשעות חולקות את אותן פרמטרים ולבסוף מקיבלים רשות אחת מלאה עם כל הנוירונים והמשקלים. בפועל עבור כל דוגימה מגරילים רשות (מווחקים כל נוירון בהסתברות $0.5 = k$) וכך לומדים במקביל הרבה רשותות שונות עם אותן פרמטרים. באופן זה כל נוירון מוכחה להיות יותר משמעותי בלי אפשרות להסתתר על נוירונים אחרים שיעשו את הלמידה, כיוון שלא תמיד הם קיימים. אמנם כל ריצה יחידה יכולה להיות בעלת שונות גבוהה אך המוצע של המשקלים מביא לשונות נמוכה.

בשלב המבחן, לא מפעילים את dropout, אלא לוקחים את כל הנוירונים, כאשר מחלקים את כל המשקלים בחצי. הסיבה לכך היא שניתן להניח שבשלב האימון חצי מהפעמים המשקל היה 0 כיוון שהנוירון הקשור אליו נמחק, ובחצי מהפעמים היה משקל שנלמד. ניתן גם לקחת הסתברות אחרת למחיקת נוירונים, למשל $0.25 = k$, ואז כמשמעותם את כל הרשותות השונות יש לחלק בהסתברות המתאימה. החיסרונו של שיטה זו הוא שלווח לה זמן לה恬נו.

4.4.4 Data Augmentation

שיטה אחרת להימנע מ-overfitting היא להגדיל את סט האימון, וכך המודל שנוצר יתאים ליותר דוגמאות. ניתן לעשות זאת על ידי ייצרת וריאציות של הדוגמאות המקוריות. שיטה זו נקראת **Data Augmentation**, והרעיון הוא לבצע עיבوت קטן לכל דוגמא כך שהיא עדין נשמר על המשמעות המקוריות שלה, אך תהיה מסווגת שונה מהמקורה כדי להיות דוגמא נוספת משמעותית בסט האימון. בדומין של תמונה האוגמנטציות הנפוצות הן:

- סיבוב תמונה בزاوية מסוימת (rotate), הנבחרת מהתפלגות אחידה מהתחום $[0, 2\pi]$.
- הוספת רעש לכל פיקסל, כאשר הרעש משתנה מפיקסל לפיקסל, והוא קטן מ- ϵ .
- שינוי הגודל (rescaling) של התמונה בפקטור מסוים – בדרך כלל הפקטור שייר לתוחם $\left[\frac{1}{1.6}, 1.6\right]$.
- שיקוף התמונה (flip).
- מתיחה ומריצה של התמונה (shearing and stretching).

4. 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).

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

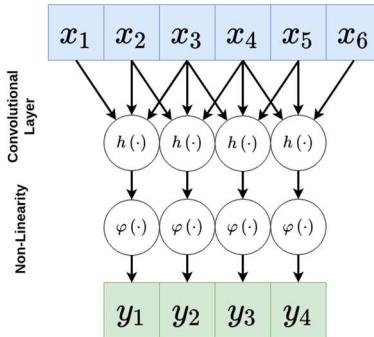
כאשר $x \in \mathbb{R}^n$ הוא וקטור הכניסה ואילו $w \in \mathbb{R}^K$ הוא וקטור המשקלים אשר נלמדים במהלך האימון. וקטור המשקלים w זהה לכל הכנימות בשכבה וכן מספר הפרמטרים הנלמדים לעומת שכבת FC הינו קטן בהרבה – שכבת FC מכילה $N_{inputs} \times N_{outputs}$ משקלים ואילו שכבת קונבולוציה מכילה K משקלים בלבד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

מלבד הקטנות�数ות המשקלים, השימוש בגרעין קונבולוציה מסיע לזרחי דפוסים ולמציאות מאפיינים. יכולות אלו מבעות מאופי פעולה הקונבולוציה, הבודקת חיפויה בין חלקים מוקטור הכניסה לבין גרעין הקונבולוציה. הקונבולוציה יכולה למצוא מאפיינים בסיגナル, ושניהם גרעיני קונבולוציה שיכולים לבצע אוסף פעולות שימושיות, כמו למשל החלקה, נגזרת ועוד. אם מתייחסים על תמונה הרבה גרעינים שונים, ניתן למצוא בה כל מיני מאפיינים – למשל אם הגרעין הוא בצורה של עין או אף, אז הוא מסוגל למצוא את האזוריים בתמונה המקוריים הדומים לעין או אף.



איור 5.1 a) קונבולוציה חד ממדית בין שתי פונקציות: x_1 הינו מלבן בגובה 1 עם רעש קטן (כחול), x_2 הינו גרעין קונבולוציה מלבני שרעף כל השר (כתום). פעולה הקונבולוציה (שחור) בודקת את החיפויה בין הסיגナル לבין הגרעין, וניתן לראות שסקיב $= x$ שרעף על 0.5 ± 0.1 יש אזכור עם הרבה חיפויה. b) קונבולוציה דו ממדית למציאת קווי מתחאר של בתוך תמונה.

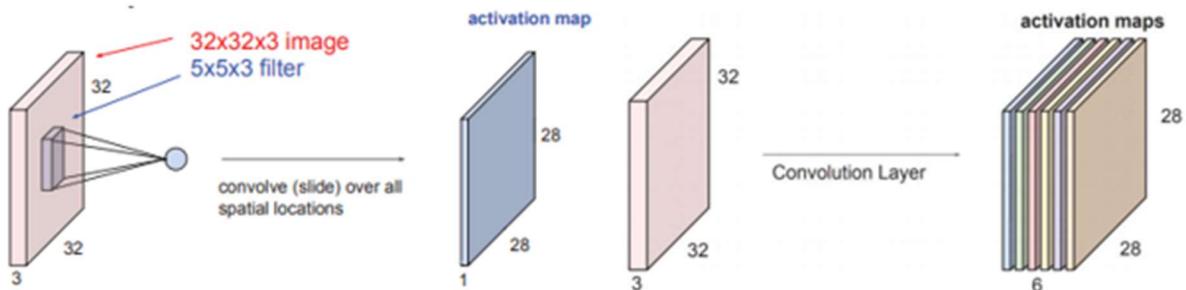
המוצא של שכבת הקונבולוציה עובר בפונקציית הפעלה לא לינארית (בדרכ כל \tanh או ReLU), והוא מכונה מפת הפעלה או מפת מאפיינים (feature map). הקונבולוציה יחד עם האקטיבציה נראה כ:



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

לרוב בכל שכבה קונבולוציה יהיה כמה מסננים, אשר כל אחד מהם אמור ללמידה מאפיין אחר בתמונה. ככל שהרשף הולכת ועמוקה, כך המאפיינים בתמונה אמרורים להיות מובחנים באופן יותר אחד מהשני, וכך המסננים בשכבות העומקות אמרורים להבדיל בין דברים מסוימים יותר. למשל, פעמים רבות ניתן לבדוק אם המסננים בשכבות הראשונות יזהו את שפות האלמנטים שבתמונה או בצורות אבסטרקטיות, ואילו מסננים 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, מסננים 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 מסננים וליצור מפת אקטיבציה עם מספר שכבות – עברו שישה מסננים 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:

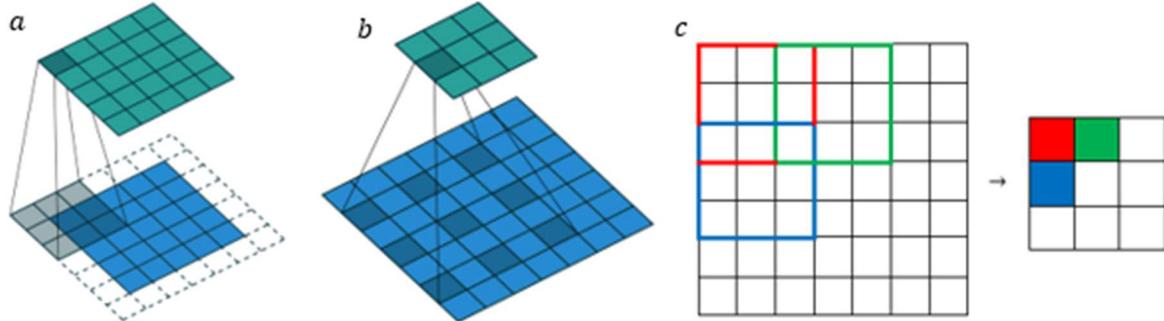
ריפורד (Padding): פעולות הקונבולוציה המוגדרת באמצעות המסן הינה פעולה מרחבית, כלומר, המסן פועל על מספר איברים בכל פעולה. בנוסף, נשים לב כי פעולה הקונבולוציה לא מוגדרת על איברי הקצוות لكن לא נוכל להפעיל את המסן במקומות אלו. באירור 5.2 ניתן לראות כיצד פעולה על תמונה במד של 32×32 מקטינה את ממד הפלט ל- 28×28 , 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

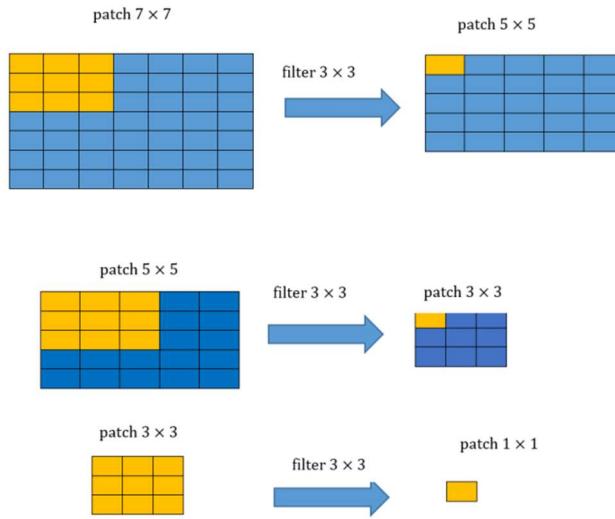
האפשרים ברשות, אלא ניתן לבצע דילוגים, כך שלאחר כל חישוב קונבולוציה יבוצע דילוג בגודל הצעד לפני הקונבולוציה הבאה. גודל צעד טיפוסי הינו $s = 2$.

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) התרחבות ($2 = d$): ביצוע הקונבולוציה וטור השמטה איברים סמוכים מתוך הנהנה שכנראה הם דומים. (c) הצעת המשן בצעד של $2 = s$.

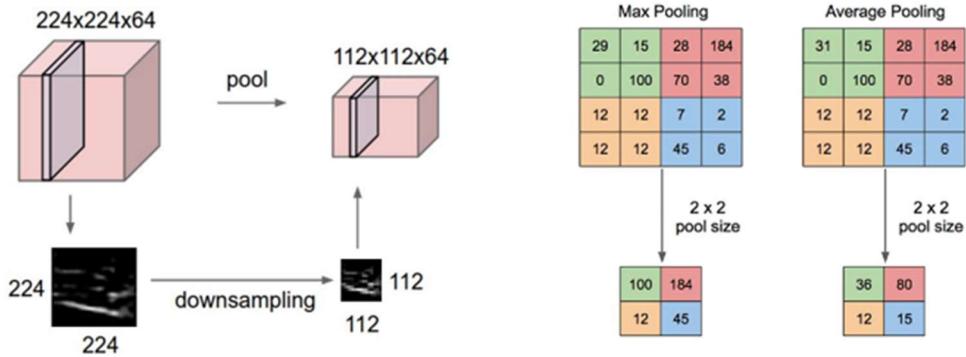
תמרק (Receptive field) של איבר ברשת מוגדר להיות כל התחום בכניסה אשר משפיע על אותו איבר לאורך השכבות.



איור 5.5 של ערך מסוים בmozo 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 עובדה זו ב כדי להוריד את מספר החישובים הדרושים בעזרת דילוגים (Strides) או הרחבה (dilation) כפי שתואר לעיל. שיטה אחרת לניצול עובדה 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).



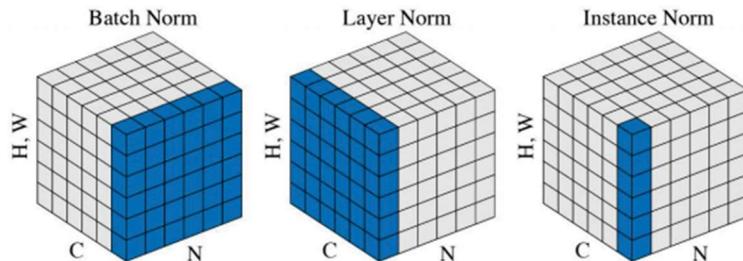
איור 5.6 הקטנת הממד של הדadata בעזרת Pooling (שמאל), והמחשה מספרית של ביצוע max/average pooling בגודל של 2×2 .

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

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

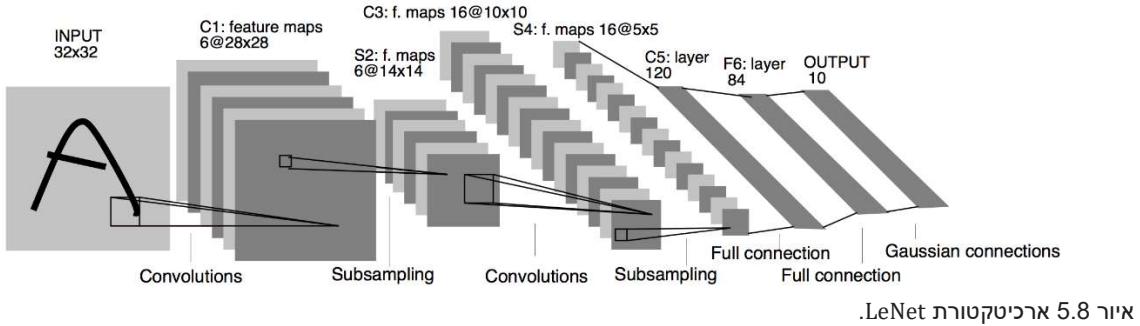
בדומה ל-FC, גם ב-CNN ניתן לבצע normalization של הנרמול על סט של וקטורים מסוימים (לשם הנוחות נתיחס לווקטורים של הדadata כתמונות). אפשרות פשוטה ונפוצה היא לנормל כל מסנן בפני עצמו על פניו כמה תמונות (Batch Norm), ככלומר לקחת את כל הפיקסלים בסט של תמונות ולנормל בתוחלת ובשונות שלהם. אפשרות נוספת היא לקחת חלק מהמידע של סט תמונות, אך לנормל אותו ביחס לאותו מידע על פניו מסננים אחרים (Layer Norm). יש וריאציות של הנרמולים האלה, כמו למשל Layer Norm, הלוקח מסנן אחד ותמונה אחת ומnormל את הפיקסלים של אותה תמונה.



איור 5.7 נרמול שכבות של רשת קומבולוציה.

5.1.5 Convolutional Neural Networks (LeNet)

בעזרת שרשור של שכבות וחיבור כל האלמנטים הש依יכים לקומבולוציה ניתן לבנות רשת שלמה עבו מגוון משימות 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 חוזרת את כל המידע שנאנסף על ידי חיבור כל הנויונים באמצעות FC. לראשונה השתמשו בארכיטקטורה זו בשנת 1998, בראשת הנקרatte LeNet (על שם Yann LeCun), ומוצגת באיור 5.8. רשת זו השיגה דיוק של 98.9% בזיהוי ספרות, כאשר המבנה שלה הוא שתי שכבות של קומבולוציה ושלוש שכבות FC, כאשר לאחר כל אחת משכבות הקומבולוציה מבצעים pooling.



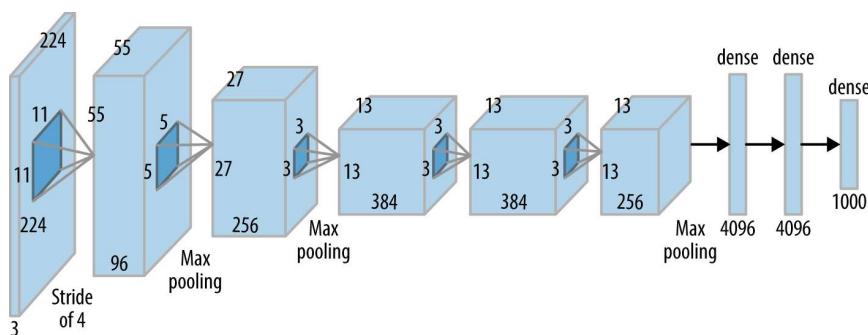
5.2 CNN Architectures

בשנים שלאחר העיסוק ברשתות נוירוניים عمוקות די-נץ, 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 מהותואה 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 שהוא דאטא-סיטים גדולים מאוד שנitin לאם עליהם את הרשות, ובנוסף כבר היה קיים GPU שביצרתו ניתן לבצע חישובים מורכבים. הארכיטקטורה 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 $s = 4$ וללא ריפוד באפסים. لكن המוצא של הקונבולוציה הינו מממד 55×55 . לאחר מכן מתיבצע 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 $= k$, אך במאז הממד הוא $27 \times 27 \times 256$, ואחרי max-pooling שכבה בממד 13×13 . לאחר מכן יש עוד 2 שכבות של קונבולוציה עם מסננים בממד $3 \times 344 \times 3$, 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 של כהן המוצא של השכבה האחרון הוא וקטור באורך 1000, המציג 1000 קטגוריות שונות שיש בדטה-סיט ImageNet.

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

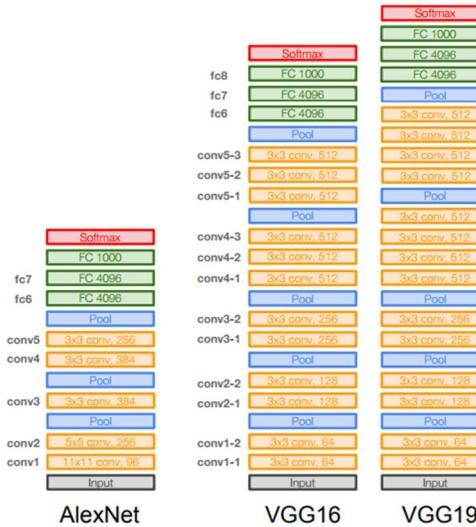


איור 5.9 ארכיטקטורת AlexNet.

שנה לאחר הפירוט 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 מסויים.

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 אותו תمر (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, لكن לפילטר של 7×7 יש $49c^2$ פילטרים נלמדים ואילו לשולש שכבות של 3×3 יש $c^2 \cdot 3^2 = 27c^2$ פילטרים נלמדים – חיסכון של 45%. הרשת המקורית שפיתחו נקראה VGG16 והיא מכילה 138 מיליון פילטרים, ויש לה וריאציה המוסיפה עוד שתי שכבות קונבולוציה ומוכנה VGG19.

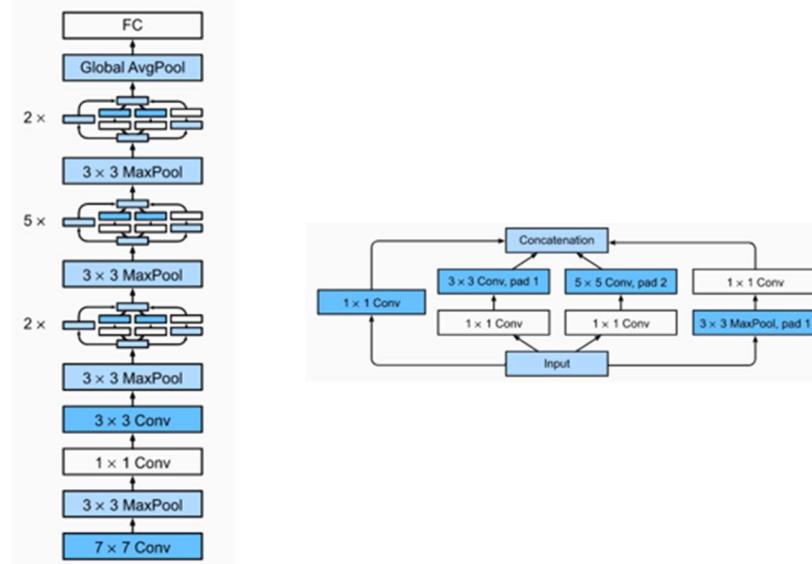


איור 5.10 ארכיטקטורת VGG (ימין) ביחס לארכיטקטורת AlexNet (שמאל).

5.2.3 GoogleNet

המודלים הקודמים היו יקרים חישובית עקב מספר הfilטרים הגדל. כדי להציגם להגעה לאותם ביצועים עם אותו עומק אבל עם הרבה פחות פילטרים, בוצעת מפתחים מוגול הציגו קונספט Snorkel module 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 שוים כרך שנייתן 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 היתרון של המבנה זהה הוא כפול: כמה פילטרים נמוכה ביחס לרשותות קודמות וחישובים יחסית מהירים ייוון שהם נעשים במקביל. ניתן לחבר שכבות קונבולוציה רגילים עם בלוקים כאלה, ולקיים רשת עמוקה. נעשו הרבה ניסויים כדי למצוא את היחס הנכון בין הרכיבים והמדריכים בכל שכבה המביאים לביצועים אופטימליים.



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

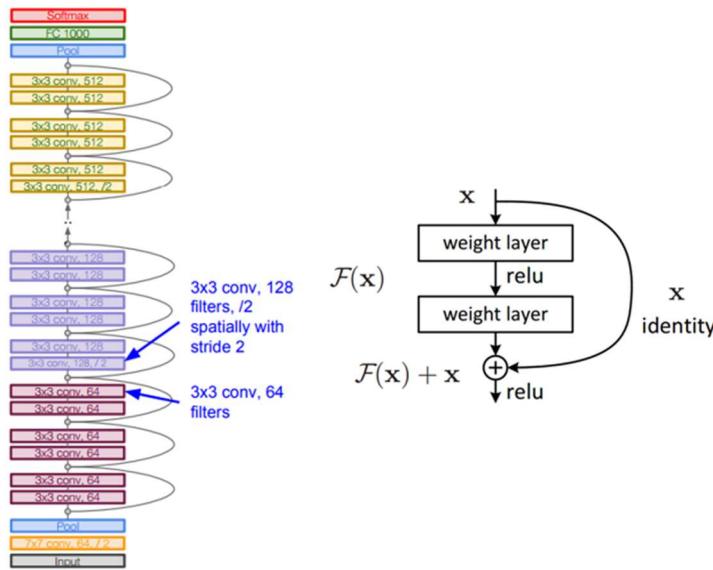
5.2.4 Residual Networks (ResNet)

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. הבעיה המרכזית של הרשתות העמוקות נבעה מכך שללאור מספר שכבות מסוים התקבל ייצוג מספק טוב, ולכן השכבות היו צרכות לא לשנות את הקולט אלא להנברר את היצוג כמו שהוא. בשיביל לבצע זאת המשקלים בשכבות אלו צרכים לפחות 1. הסתבר לשכבות קשה ללמידה את פונקציית ההזאות והן מעשנה פגעו בתוצאה. אתגר נוסף ברשתות עמוקות נבע מהקשה לבצע אופטימיזציה כמו שצורך למשקלים בשכבות עמוקות.

ניתן לנסח את הבעיה המרכזית באופן מעט שונה – בהינתן רשת עם N שכבות, יש טעם להוסיף שכבה נוספת רק אם היא תוסיף מידע שלא קיים עד עכשווי. כדי להבטיח ששכבת תוסיף מיידע, או לכל הפחות לא תפגע במידע קיימים, בנו רשת חדשה עזרת Residual Blocks – יצירת בלוקים של שכבות קוונבולוציה, 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 משזה 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 קוונבולוציה. בין כל שלשה יש הכפלה Batch normalization והורדנה של הממד פ' 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 מתיישר,

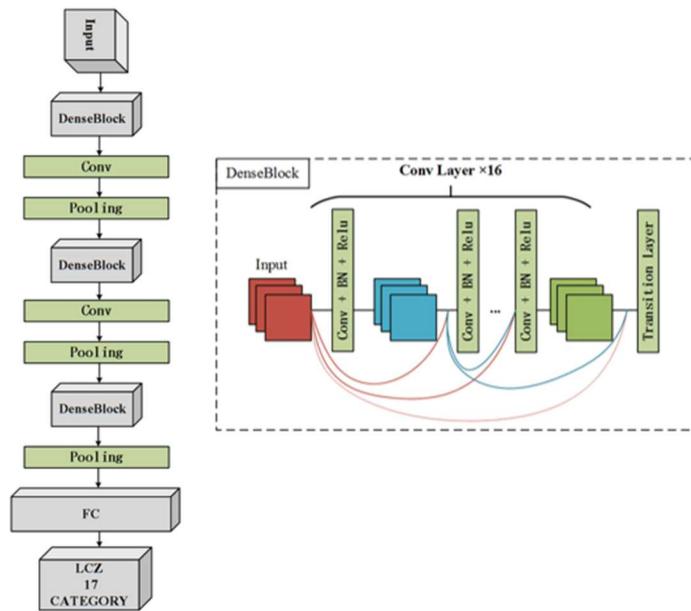
רשתות מתקדמות יותר שילבו את גישת ה-inception יחד עם ResNet על מנת לשלב בין היתרונות של שתי השיטות.



איור 12 Residual Block 5.12 (ימין), וארQUITקטורת 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 למצאה 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.

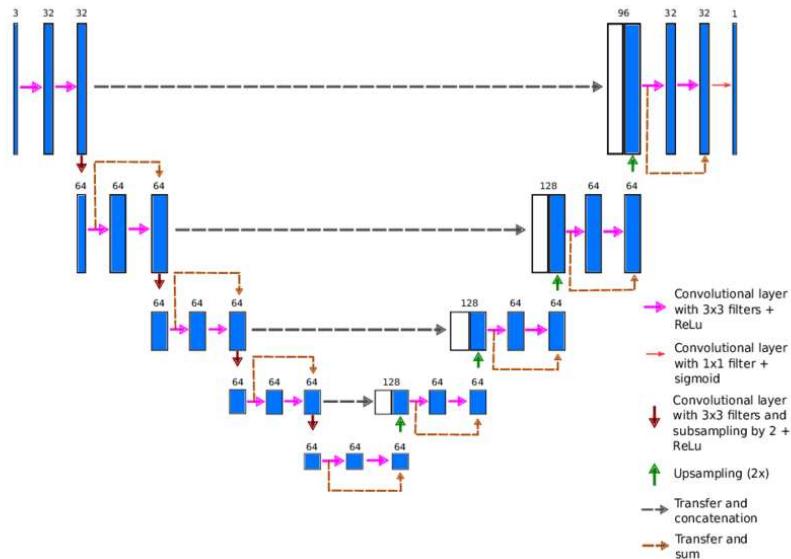


איור 13 Dense Block 5.13 (ימין), וארQUITקטורת DenseNet מלאה (שמאל).

5.2.6 U-Net

ברשותות קונבולוציה המיפויות לסיווג, בסוף התהילה מתקובל וקטורי של הסתברויות, כאשר כל איבר הוא הסתברות של label מסוים. במשימת סגמנטציה זה בעייתי, כיוון שצריך בסוף התהילה לא רק ללמד את המאפיינים שבתמונה ועל פייהם לקבוע מה יש בתמונה, אלא צריך גם לשחרר את מיקומי הפיקסלים והתוצאות 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).

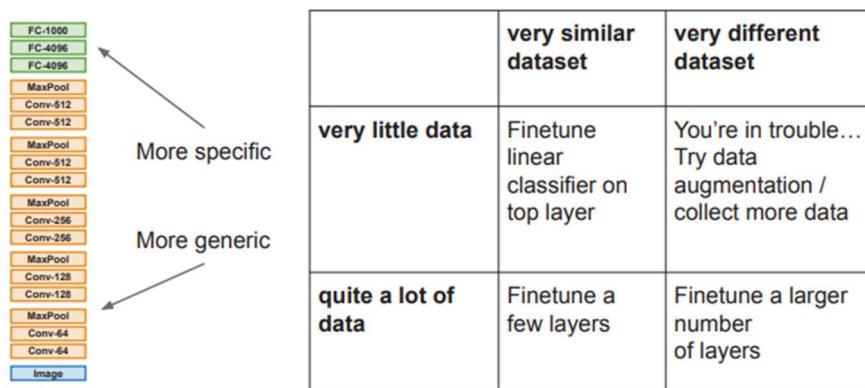
באיור, בחלק הראשון יש טופולוגיה רגילה של רשת קונבולוציה, המבוצעת בעזרת שכבות קונבולוציה וביצוע pooling. השוני בין השלב זהה לבין רשת קונבולוציה קלאסית הוא החיבור שיש בין כל שלב בתהילר לבין חלקים בהמשך התהילר. לאחר המעבר בצדואר הבקבוק יש למעשה שחזור של התמונה עם הסגמנטציה. השחזור נעשה בעזרת sampling על 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 ברשיות 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 שהיא מוכוונת 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 לאמן מספר 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 אטגרם אלו יש לבנות ארכיטקטורה שמקבלת סדרה של וקטורים וממציאות וקטור יחיד, כאשר הווקטור היחיד מקודד קשרים על הדאטא המקורי שנכנס אליו. את וקטור המוצא ניתן להעביר בשכבה FC או בכל מסוג אחר, תלוי באופן המשימה.

6.1 Sequence Models

6.1.1 Vanilla Recurrent Neural Networks

רשתות רקורסיביות הן הכללה של רשתות נוירונים עמוקות, כאשר הן מכילות משקלות המאפשרות להן לחת משמעות לסדר של איברי הכניסה. ניתן להסתכל על משקלות אלה כרכיב זיכרון פיני, כאשר כל איבר שוכנו משקל בלבד ביחס לפונקציה קבועה בתוספת רכיב משתנה שתלו 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} :

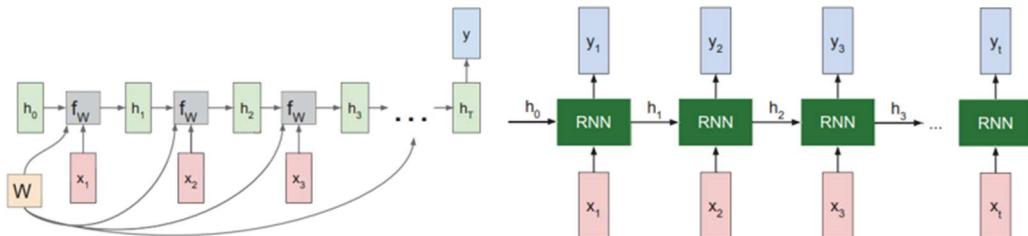
$$h_t = f(h_{t-1}, x_t)$$

מלבד המשקלים הפעילים על וקטור הכניסה, יש גם משקלים שפועלים על המשקלות הפנימיות (רכיב הזיכרון) – w_{hh} , ומשלימים 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:



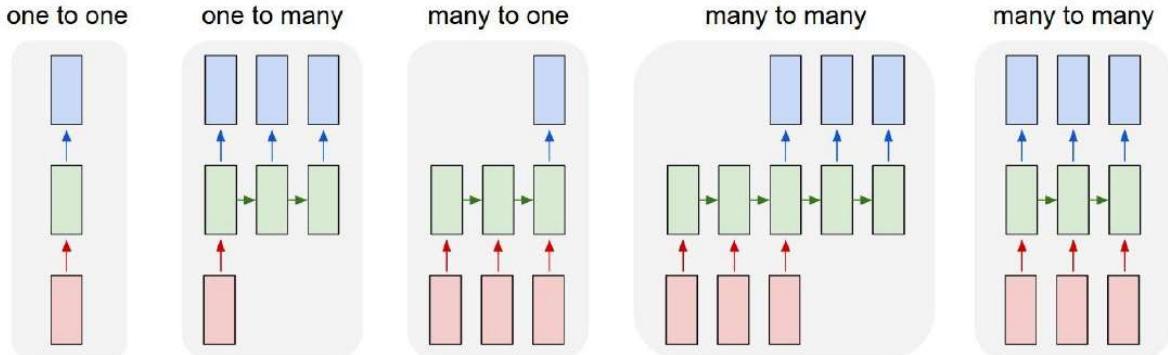
איור 6.1 ארכיטקטורות RNN בסיסיות: Many to One (מימין) – One to many (משמאל) (Many to Many). על כל חץ יש משקל מתאים עליו מתבצע הלמידה.

כמובן שניתן גם לשרשרא שכבות ח比亚ות ולקבל רשת עמוקה, כאשר פלט של שכבה מסוימת הופך להיות הקלט של השכבה הבאה. ישנם מודלים שונים של RNN, המתאים לבעיות 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) הגדיר את פונקציית המחר:

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

כאשר הפונקציה $L(\hat{y}_i, y_i, \theta)$ תותאם למשימה – עבור משימות סיווג cross entropy ועבור בעיות רגראטיות נשתמש בקריטריון MSE. האימון יבוצע בעזרת GD, אך לא ניתן להשתמש ב-backpropagation כיוון שכל משקל מופיע מספר פעמים – למשל w_{hx} פועל על כל הכניסות ו- w_{hh} פועל על כל רכיבי הזיכרון. כדי לבצע את עדכון המשקלים משתמשים ב-(BPTT) backpropagation through time. אם הדאטה בוכנינה גדולה, מחשבים את הגרדיאנט עבור כל משקל, ואז סוכמים או ממצאים את כל הגרדיאנטים. אם הדאטה בוכנינה הוא בגודל a , כלומר יש a דוגמאות בזמן, אז יש a רכיבי זיכרון, ו- $a - 1$ – a משקלים w_{hh} . لكن הגרדיאנט המשוקל יהיה:

$$\frac{\partial L}{\partial w_{hh}} = \sum_{n=1}^a \frac{\partial L}{\partial w_{hh}(t)} \quad \text{or} \quad \frac{\partial L}{\partial w_{hh}} = \frac{1}{a} \sum_{n=1}^a \frac{\partial L}{\partial w_{hh}(t)}$$

כיוון שהמשקלים זהים לאורך כל הרשות, $w_{hh} = (t)$ והשני בזמן יהיה רק לאחר ביצוע ה-BPTT יהיה רלוונטי רק לוקטור הבא.

הצורה הפשוטה של ה-BPTT יוצרת בעיה עם הגרדיאנט. נניח שרכיב הזיכרון מיוצג באמצעות הפונקציה הבאה:

$$h_t = f(z_t) = f(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$$

לפי כלל השרשרת:

$$\frac{\partial h_n}{\partial x_1} = \frac{\partial h_n}{\partial h_{n-1}} \times \frac{\partial h_{n-1}}{\partial h_{n-2}} \times \dots \times \frac{\partial h_2}{\partial h_1} \times \frac{\partial h_1}{\partial x_1}$$

כיוון ש- w_{hh} קבוע ביחס בזמן עבור קטור כניסה יחיד, מתקבל:

$$\frac{\partial h_t}{\partial h_{t-1}} = f'(z_t) \cdot w_{hh}$$

אם נציב זאת בכלל השרשרת, נקבל שעבור חישוב הנגזרת $\frac{\partial h_n}{\partial x_1}$ מכפילים 1 – a פעמים ב- w_{hh} . לכן אם 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 רקוריוביות זה קורה כמעט תמיד.

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

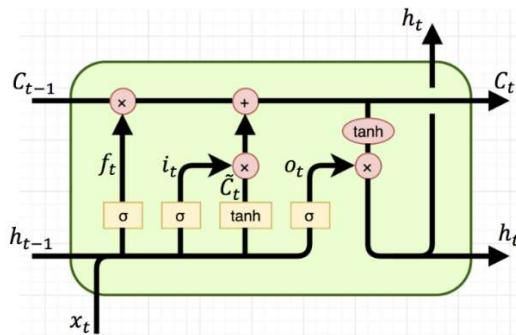
$$\text{if } \|g\| > c, \text{ then } g = \frac{cg}{\|g\|}$$

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 במידע. ב-RNN הפשט לריכב הזכרון יש שתי כניסה – x_t ו- h_{t-1} , ובעזרתן מחשבים את המוצא על ידי שימוש בפונקציה $(x_t, h_{t-1}) \cdot f_w$. 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 $f_t = \sigma(x_t \cdot w_f + b_f)$ הוא 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** $i_t = \sigma(x_t \cdot w_i + b_i)$ הוא שער זיכרון והוא אחראי על כמה יש לזכור את המידע החדש לטווח ארוך. אם לדוגמה אכן יש במשפט מוסים חדש 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** $\tilde{c}_t = \tanh(x_t \cdot w_c + b_c)$ הוא שער מוצא והוא אחראי על כמה מהמידע RALONOTI לדעתה הנוכחי c_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** $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ (לפערמים מסוימים באות g) שאחראי על השאלה כמה מהזיכרון להזכיר לתא הבא. שער זה משקל את המידע המתkeletal יחד עם c_t שאומר עד כמה יש לזכור להמשך את המידע החדש.

באופן זהה מקבלים הוא את c_t שאחראי על זיכרון לטווח קצר $Vanilla$ RNN, והן את c_t שאחראי על זיכרון של כל העבר. ארכיטקטורת הריכב מאפשרת להתייחס לאלמנטים נוספיםים הקשורים לזכרון – ניתן לשכוח חלקים לא רלוונטיים של התא הקודם (f_t), להתייחס באופן סלקטיבי לכינסה (i_t) ולהוציא רק חלק מהמידע המשוקל הקיים (o_t). באופן פורמלי ניתן לנסח את פעולות התא כ:

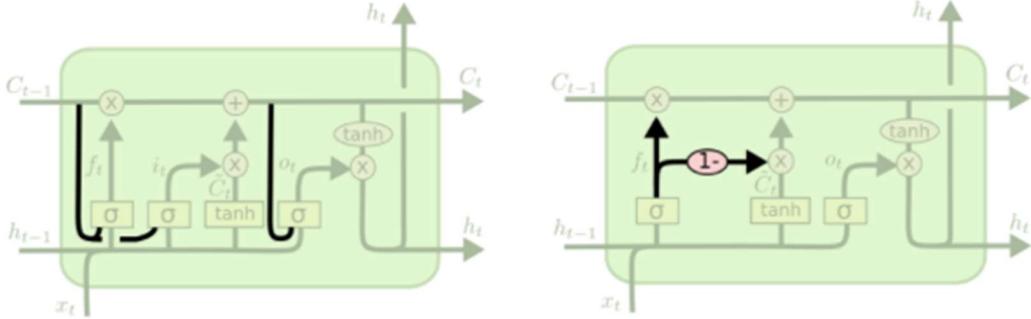
$$\begin{pmatrix} i \\ f \\ o \\ \tilde{c} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} = \begin{pmatrix} \sigma(w_i \cdot [x_t, h_{t-1}] + b_i) \\ \sigma(w_f \cdot [x_t, h_{t-1}] + b_f) \\ \sigma(w_o \cdot [x_t, h_{t-1}]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

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

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 למוצא h_t אלא גם לשאר השערים. חיבור כזה נקרא peephole, כיוון שהוא מאפשר לשערים 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 השערם $\Theta c_{t-1} \cdot w$ משנה מבנה את משוואות השערם. במקום $(b + b) \cdot w$ המשווה החדש יהיה $(b + \Theta c_{t-1} \cdot w) \cdot w$.

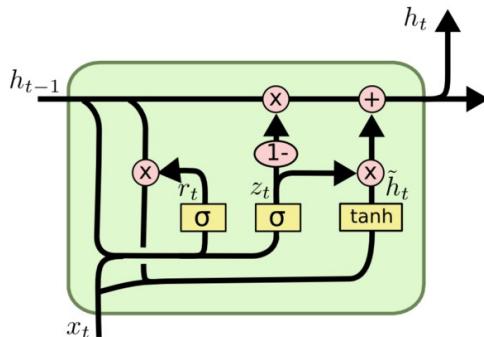
וריאציה אחרת של LSTM מתחדשת בין שער השכחה f_t לבין שער הזיכרון i_t , והחלה עד כמה יש למחוק מידע מהזיכרון מתקבלת יחד עם החלטה כמה מידע חדש יש לכתוב. שינוי זה ישפייע על memory cell, כאשר במקום $c_t = f_t \Theta c_{t-1} + (1 - f_t) \Theta \tilde{c}_t$, משווהות העדכון תהיה: $c_t = f_t \Theta c_{t-1} + i_t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



איור 6.4 וריאציות של LSTM – LSTM – (ימין) ו- i -pesphole 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 והמוצא 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 $\in [0, 1]$ כי Ci כיוון שהוא תוצאה של סיגמויד. כתעת נתבונן על \tilde{h}_t ביחס לרכיב זיכרון פשוט של Vanilla RNN: $h_t = f_W(w_{hh}h_{t-1} + w_{xh}x_t)$

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

המשמעות היא שעבור $1 \rightarrow r_t$ 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:

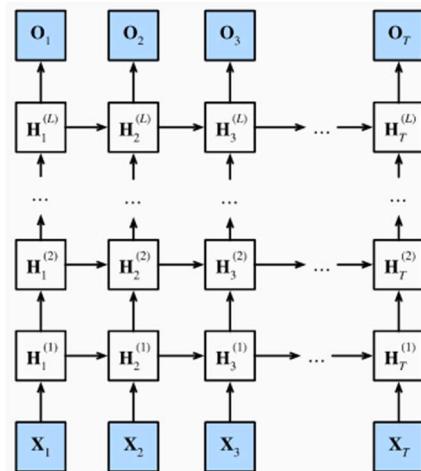
$$h_t = (1 - z_t)\Theta h_{t-1} + z_t\Theta \tilde{h}_t$$

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 שעור העדכון קרוב ל-1 כל הזמן, אז בעצם מעבירים את המצב החבוי כמו שהוא, ולמעשה הזיכרון נשמר לאורק זמן. בנוסף, אין בעיה של התבדרות הגראדיינט, כיוון שאין שאמם השינוי בין תא לתא 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 -ה� 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 החבואה 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

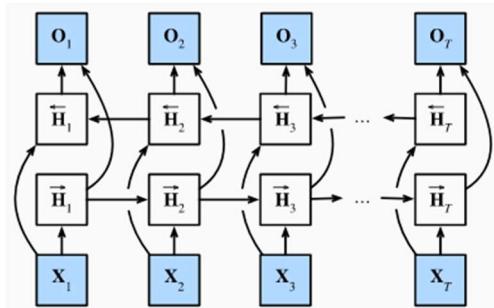
כל הרכיבים והרטות שנידונו עד כה עוסקו בסדרות סיבתיות, כלומר, סדרות בהן כל איבר מושפע מקודמי או אף לא מallow הבאים אחריו. למשל, ערך מניה ביום מסוים קשור לערכה ביום הקודמים, אך הערך שלא ביום המחרת (שכלל עוד לא ידוע) לא מושפע צורה על ערכה ביום הנוכחי. דוגמא נוספת – התפתחות של גל בזמן תלויה בערכי הקודמים של הגל אך 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 שימושי 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 מושרים אותם יחד ומתקבלים את $H_t \in \mathbb{R}^{n \times 2h}$, ובעזרתו מחשבים את המוצא:

$$o_t = H_t w_{hq} + b_q$$

כאשר $w_{hq} \in \mathbb{R}^{2h \times q}, b_q \in \mathbb{R}^{1 \times q}$ הם הפרמטרים של שכבת הפלט.

6.2.5 Sequence to Sequence Learning

ב

<https://buomssoo-kim.github.io/blog/tags/#attention-mechanism>

6. References

Vanilla:

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

LSTM, GRU:

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Deep RNN, Bidirectional RNN:

http://d2l.ai/chapter_recurrent-modern/index.html

7. Deep Generative Models

המודלים שהוצגו בפרקם הקודמים הינם מודלים דיסקרימינטיביים, קרי הם מאמנים לבצע פעולות על בסיס נתונים, אך לא יכולים ליצור פיסות מידע או דוגמאות חדשות בעצמו. 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.

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 אינו עוסק 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)

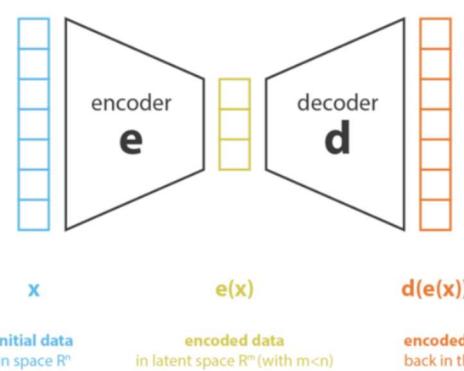
המודל הראשון הינו VAE, וכדי להבין כיצד ניתן בעזרתו לייצר מידע חדש, יש להסביר קודם כיצד מודדים אותו עובדים ומה החסרונות שלהם. Autoencoder הינו רשת המאומנת לבצע הורדת מידע לדטה, תוך איבוד מינימלי של מידע. כדי להבין את המבנה ואופיו הפעולה שלו, נקדים מעט על הורדת מידע באופן כללי.

7.1.1 Dimensionality Reduction

במקרים רבים, הדטה אותו רוצים לנתח הוא בעל ממד גובה, למשל, כל דגימה 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 יתר מאשר הגיל הממוצע של העובדים. דוגמא נוספת – במקרים רבות חיזוי גיל של אדם על פי תכונת הפנים שלו, לא כל הפיקסלים בתמונה הפנים יהיו בעלי אותה חשיבות לצורך החיזוי. כיוון שקשה לנתח דטה מממד גובה ולבנות מודלים עבור דטה כזו, במקרים רבים מנסים להוריד את הממד של הדטה תוך איבוד מידע מינימלי עד כמה שניתן. בהתאם לכך, מינימום מינימלי עבור ממד יותר נורא, כאשר היצוג 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 הלטני** (latent representation) או הקוד הלטני (latent code). ואומרו, יותר קל לבנות מודלים למשימות שונות על סמך היצוג הלטני של הדטה מאשר לעבוד עם הדטה המקורי. כדי לקבל ייצוג לטני איותי, ניתן לאמן אותו באמצעות מנגנון הנקרא **decoder**, הבוחן את יכולת השחזור של הדטה מהיצוג הלטני שלו. ככל שניתן לשחרר בצורה מדויקת יותר את הדטה מהיצוג הלטני, ככל מוגן שגורן המידע בתהיליך הוא קטן יותר, כך הקוד הלטני אכן מייצג בצורה אמינה את הדטה המקורי.

טהיליך האימון הוא דו שלבי: פיסת מידע המוצג על ידי קטור המאפיינים $\mathcal{X} \in \mathbb{R}^n$ עובר דרך encoder, שמטרתו להפיך מהדטה את היצוג הלטני שלו $\mathcal{Z} \in \mathbb{R}^m$, כאשר $n > m$. לאחר מכן התוצאה מוכנעת ל-**decoder** בצד השני לשחרר את הדטה המקורי, ולבסוף מתקובל וקטור $\mathcal{Z} \in \mathbb{R}^m$ (decoder input) $\mathcal{Z}(x) = d(e(x))$. אם מתקיים השוויון $d(e(x)) = x$ אז למעשה לא אבד שום מידע בתהיליך, 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 מהיצוג שלו בממך הנמוך בדיקות טוב מספיק, נראה שהיצוג הלטני הצליח להפיך את המאפיינים העיקריים של הדטה המקורי.



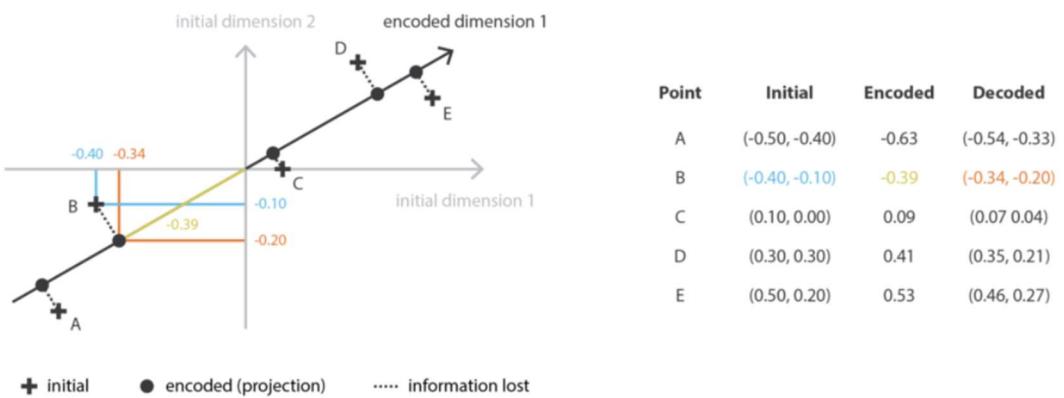
איור 7.1 ארכיטקטורת encoder-*i*-decoder.

כאמור, המטרה העיקרית של השיטות להורדת ממד הינה לקבל ייצוג לטנסי אינטואיטיבי עד כמה שניתן. הדרך לעשות זאת היא לאמן את זוג encoder-decoder השומרים על מקסימום מידע בעת הקידוד, וمبאים למינימום את שגיאת שחזור בעת הפענוח. אם נסמן בהתאם E ו-D את כל הזוגות של encoder-decoder האפשריים, ניתן לנסח את בעיית הורדת הממד באופן הבא:

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

כאשר $\epsilon(x, d(e(x)))$ הוא שגיאת השחזור שבין הדטה המקורי לבין הדטה המשוחזר.

אחד השיטות השימושיות להורדת ממד שאפשר להסתמך עליה בצורה זו היא Principal Components Analysis (PCA). בשיטה זו מטילים (בצורה לינארית) דאטה מממד n לממד $m < n$, כך שהמאפיינים של הייצוג הלטנטי של הדוגמאות המקוריות יהיו אורתוגונליים. תהליך זה נקרא גם feature decorrelation, והמטרה שלו היא למצער את המרחק האוקלידי בין הדטה המקורי לדאטה המשוחזר, בצורה לינארית גם כן, מהויצוג החדש במרחב ה- m -ממדי.

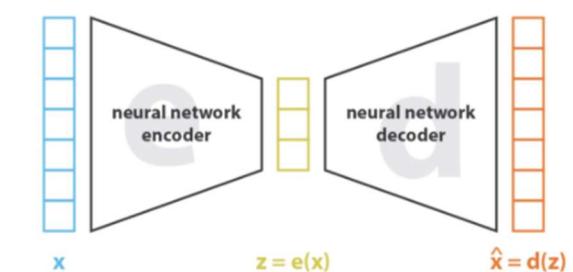


איור 7.2 דוגמא להורדת ממד בשיטת PCA.

במנוחים של encoder-decoder, ניתן להראות כי אלגוריתם PCA מ Chapman את הזוג של שמי'מים שני תנאים: א.encoder מבצע טרנספורמציה לינארית על הדטה כך שהמאפיינים החדשים (בממד נמוך) של הדטה יהיו אורתוגונליים. ב. decoder הלינארי המתאים יביא לשגיאה מינימלית במרחק אוקלידי בין הדטה המקורי לבין זה המשוחזר מהויצוג החדש. ניתן להוכיח שה-encoder האופטימלי מכיל את הווקטורים העצמיים של מטריצת covariance של מטריצת $\text{cov}(x)$, וה-decoder הוא השולוף של ה-encoder.

7.1.2 Autoencoders (AE)

ניתן לקחת את המבנה של ה-AE המtauור בפרק הקודם ולהשתמש ברשת נירונים עבור בניהת הייצוג החדש ועבור השחזור. מבנה זה נקרא Autoencoder:



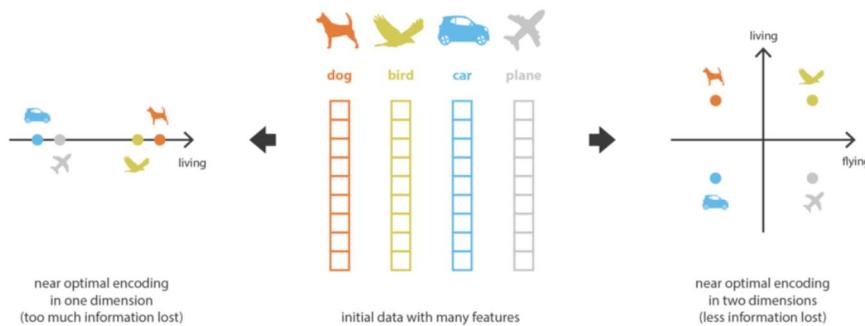
$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$

איור 7.3 – שימוש ברשתות נירונים עבור הורדת הממד והשחזור.

באופן זהה, הארכיטקטורה יוצרת לדטה צוואר בקבוק מידע (information bottleneck), שمبرטיח שרק המאפיינים החשובים של הדטה, שבאמצעותם ניתן לשחזר אותה בדיק טוב, יושמשו לייצוג במרחב הלטנטי. במקרה הפشو בו בכל רשת יש רק שכבה חבויה אחת והוא לא משתמש בפונקציות הפעלה (activation functions) לא לינאריות, ניתן לראות כי ה-AE יחשוף טרנספורמציה לינארית של הדטה, שבאמצעותו ניתן לשחזרו

באופן לינארי גם כן. בדומה ל-PCA, גם רשות 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 ה-encoder וה-decoder הן רשותות עמוקות ומשתמשות בפונקציות הפעלה לא לינאריות. במקרה זה, ככל שהארקיטקטורה של הרשותות מורכבת יותר, כך רשות ה-encoder יכולת להוריד יותר ממדים תוך יכולת באמצעות ה-decoder לבצע שחזור ללא כל איבוד מידע. באופן תיאורטי, אם ל-encoder ול-decoder יש מספיק דרגות חופש (למשל מספיק שכבות ברשת 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 downstream. 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 שمبחר ביןיהם 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 מתקובל ייצוג חד-ממדי, אבל הוא גורם לאיבוד המבנה הסמנטי של הדוגמאות ולא באמנת נתן 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.

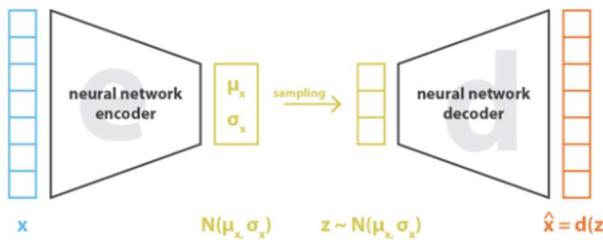
7.1.3 Variational AutoEncoders (VAE)

ניתן לקחת את AE ולהפוך אותו למודל גנרטיבי, כלומר מודל שמסוגל לייצר בעצמו דוגמאות חדשות שאכן מתפלגות כמו הפולוג של הדאטה המקורי. אם מדובר בדומין של תמונות למשל, אז נרצה שהמודול יהיה מסוגל לייצר תמונות שנראות אוטנטיות ביחס לדאטה עליו אומן. AE מאמן לייצג את הדאטה בממד נמוך, שולקח בחשבון את המאפיינים העיקריים, ולאחר מכן משחזר את התוצאה לממד המקורי. אולם, מגננן זה אינו אפשרי להשפיע על האופן בו הדאטה מייצג במרחב הלטני. אם יוגרך וקטור כלשהו מהמרחב הלטני – קרוב לוודאי שהוא ייצוג שדומה לדאטא המקורי. אם היינו מכנים אותו ל-decoder, סביר להניח שהחזרה לא תהיה דומה בכלל לדאטא המקורי. למשל אם AE אומן על אוסף של תמונות של כלבים ודוגמים באקריא וקטור מהמרחב הלטני שלו, הסיכוי לקבל תמונה כלב כלשהו לאחר השחזור של ה-decoder הינו אפסי.

כדי להתמודד עם בעיה זו, ניתן להשתמש ב-(VAE) Variational AutoEncoder. בשונה מ-AE שולקח דאטא 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 ה-encoder מאומנת לקבל דאטא x ולהוציא פרמטרים של התפלגות פואטריאורית $x|z$, מתחזק מטריה למזער כמה שניתן את המרחק בין התפלגות z ו- $x|z$. לאחר מכן וקטורים מההתפלגות הפואטריאורית $x|z$ (הנתונה על ידי הפרמטרים המוחשבים ב-encoder), ומעבירים אותם דרך decoder כדי לייצר פרמטרים של התפלגות $x|z$. חשוב להבהיר שאם הדאטה המקורי הוא תמונה המורכבת מאוסף של פיקסלים, אז בmozaica 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, כיון שהוא שמש ליצירת דאטה חדש לאחר תהילך האימון, או במקרים אחרות, הוא הופך את המערכת למודל גנרטיבי.



איור 7.5 ארכיטקטורה של VAE.

לאחר שהציג המבנה הכללי של VAE, ניתן לתאר את תהילך האימון, ולשם כך נפריד בשלב זה בין שני החלקים של h-VAE. מאמן 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 כזה שימזעր את השגיאה של שני חלקים – VAE – גם $|z|$ שבmoza היה כמו שיוצר קרוב ל- x המקורי, וגם התפלגות $|z$ תהיה כמה שיוצר 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 למציאת הנראות המרבית של סט האימון תחת θ . כיון 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. כיון 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 זו 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 שמיין לאי 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- $\mathcal{D}_{KL}(q(z|x, \theta, \lambda) \| p(z|x, \theta))$

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

אם שתי התפלגות זהות, אז 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 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

$$= \sum_z q(z|x; \lambda) (\log p(x|z; \theta) + \log p(z; \theta)) - \sum_z q(z|x; \lambda) \log q(z|x; \lambda)$$

$$= \sum_z q(z|x; \lambda) \log p(x|z; \theta) - \sum_z q(z|x; \lambda) (\log q(z|x; \lambda) - \log p(z; \theta))$$

$$= \sum_z q(z|x; \lambda) \log p(x|z; \theta) - \sum_z q(z|x; \lambda) \frac{\log q(z|x; \lambda)}{\log p(z; \theta)}$$

הביטוי השני לפי הגדרה שווה ל- $-\mathcal{D}_{KL}(q(z|x; \lambda) \| p(z; \theta))$, שכן מתקובל:

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

הביטוי הראשון הוא בדיקת התוחלת של $\log p(x|z; \theta)$. תחת ההנחה ש- z מתפלג נורמלית, ניתן לרשום:

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

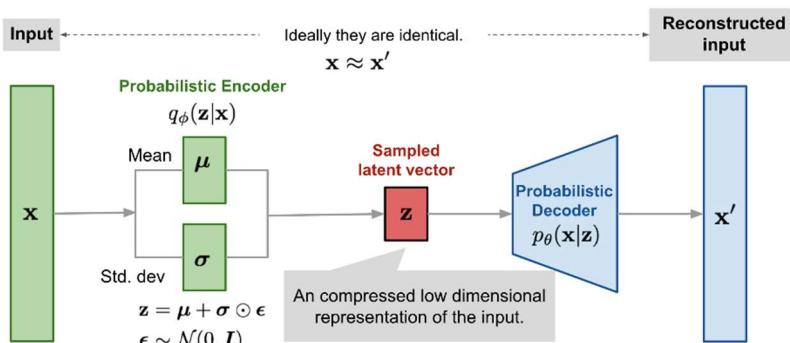
כדי לחשב את התוחלת ניתן פשוט לדוגמאות מההתפלגות $(x|z)$ ולקבל:

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

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 תלולה רק ב- θ , אך באופן שיפתחנו אותה היא למעשה דואגת גם למצער הפרש בין הכניסה של VAE לבין המוצא שלו, וגם למצער המרחק בין ההתפלגות הפרוירית של z לבין ההתפלגות $x|z$ שבמוצאה encoder-VAE.



$$x_t \rightarrow \mu_\lambda(x_t), \Sigma_\lambda(x_t) \rightarrow z_t \sim \mathcal{N}(\mu_\lambda(x_t), \Sigma_\lambda(x_t)) \rightarrow \mu_\theta(z_t), \Sigma_\theta(z_t)$$

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

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 – מוגבלים z_0 מראש ואז רק נשאר לחשב באופן סכמטי את התפשטות הערך ברשות.

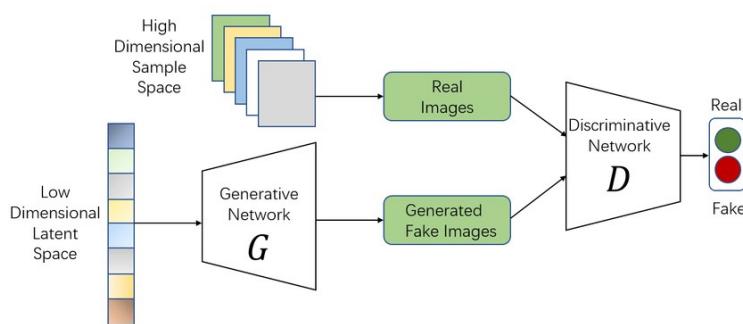
7.2 Generative Adversarial Networks (GANs)

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, ורשות 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 לרשota 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) ורשות שמכריעה האם>Data ה-*real* ה-*fake* סינטטי או אמיתי (discriminator), כאשר האימון נעשה על שתי הרשותות יחד. ה-*discriminator* מקבל כקלט *real* או *fake* *sample* ובודק אם הוא אמת או לא. ה-*generator* מיציר דוגמאות ומתקבל פידבק מה-*discriminator* לומד לייצר דוגמאות שיראו 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- $D(x)$ את ההסתברות הזו. כדי לאמן את ה-*discriminator* נרצה להשיג שני דברים: א. למקסם את $D(x)$ עבור x מסט האימון, כלומר, לטעות כמה שפחות בזיהוי DATA אמית. ב. למזער את $D(x)$ עבור DATA סינטטי, כלומר, להזות כמה שיותר דוגמאות סינטטיות שייצרו על ידי ה-*generator*. באופן דומה נרצה לאמן את ה-*generator* כך שהדוגמאות שהוא מייצר תהינה כמה שיותר דומות לדוגמאות אמיתיות, כלומר generator-*discriminator* מעוניין לגרום ל-*discriminator* להוציא ערכים כמה שיותר גבוהים עבור הDATA 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, המטרה היא לאמן את G בעזרת D (במצבו הנוכחי), כדי שהיא מסוגל ליצור דוגמאות הנראות אוטנטיות. האימון של ה-*generator* נעשה באמצעות Gradient Descent (מצער פונקציית המחיר ביחס ל- θ_g), והאימון של ה-*discriminator* נעשה באמצעות Gradient Ascent (מקסום פונקציית המחיר ביחס ל- D_{ϕ_d}). האימון מתבצע במספר מוסים של Epochs, כאשר כאמור מאמנים לסירוגין את G ו- D . בפועל דוגמים mini-batch בגודל m מסט האימון (x_m, z_1, \dots, z_m) ו- m דוגמאות של רעש (x_1, \dots, z_1), ומכוונים את הקלט ל- G . הגרדיאנט של פונקציית המחיר לפि הפרמטרים של ה-*generator* במהלך האימון מחושב באופן הבא:

$$\nabla_{\theta} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\theta} \sum_{i=1}^m \log \left(1 - D_{\phi}(G_{\theta}(z_i)) \right)$$

וכאשר מאמנים את ה-*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 המטרה של ה-*generator*. כיון שבהתחלת הדגימות המיוצרות על ידי ה-*generator* לא דומות לחולוטן לאלו מסט 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 מtabצע 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:

$$f(D(x)) = p_r(x) \log D(x) + p_g(x) \log(1 - D(x))$$

נגזר 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 $\frac{1}{2} = D(x)$). 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:

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

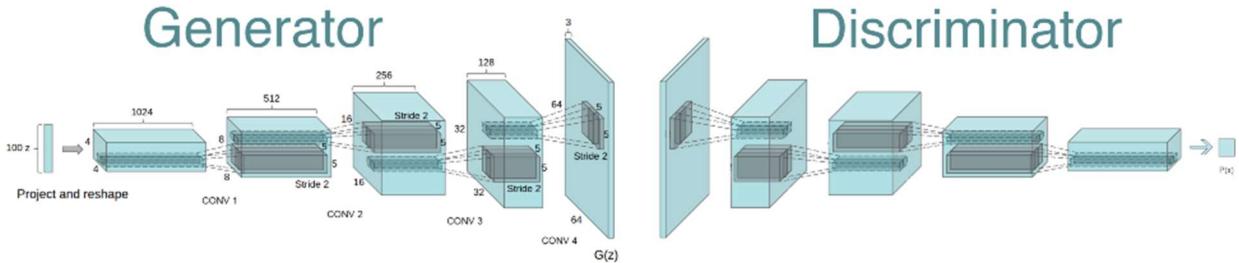
כאשר G אופטימלי ומתקיים ($p_g(x) = p_r(x)$, כלומר המרחק בין התפלגיות שווה 0, כלומר $\mathcal{D}_{JS}(p_r, p_g) = 0$), אך מתקבל:

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

יש משמעות גדולה לביטוי שהתקבל – ככל שנצליח למיצער 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 ביןארい 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 אוטנטית. למשל, עבור אוסף תמונות המציגות את הספורות 0-9, ונרצה שה-GAN ייצור תמונה של ספרה מסוימת. במקרה אחד, במקרה מסוים של הוכנסת 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 שלGAN רגיל למעט העבודה שהבאים הופכים להיות מותנים:

$$\mathcal{L}_c(D, G) = \min_G \max_D \mathbb{E}_{x \sim Data} \log D(x|y) + \mathbb{E}_{z \sim Noise} \log (1 - D(G(z|y)))$$

7.2.4 Pix2Pix

כפי שראינו, ה-GAN הקלאסי שתוואר לעיל מסוגל ליצור דוגמאות חדשות מוקטור אקראי z , המוגדר מההתפלגות מסוימת (בדרכו כלל התפלגות גאוסית סטנדרטית, אך זה לא מוכחה). ישן גישות נוספת ליצור דטה חדש, כמו למשל ייצור תמונה חדשה על בסיס קוווי מתאר כליליים שלה. סט האימון במקורה זה בניית מזגאות של תמונות והסקציות שלהן.

שיטה Pix2Pix משתמשת בארכיטקטורה של GAN אך במקומם לדגם את וקטור z מההתפלגות ממשה, 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 להתקאה 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 כטammone מסט האימון המתאימה לסקיצה S ופעם זאת שמיוצרת על ידי ה-generator על בסיס S). על ה-discriminator לקבע האם התמונה היא אכן Tamona אמיתית של הסקיצה או Tamona סינטטית. ו/orientation זו של ה-GAN משנה גם את פונקציית המחר – כתעת ה-generator צריך ללמוד שני דברים – גם ליצור Tamona טובות כרשה-discriminator יאמין שהן אמיתיות, וגם לסייע את המרחק בין Tamona שנוצרת לבין Tamona אמיתיות השיכת לסקיצה.

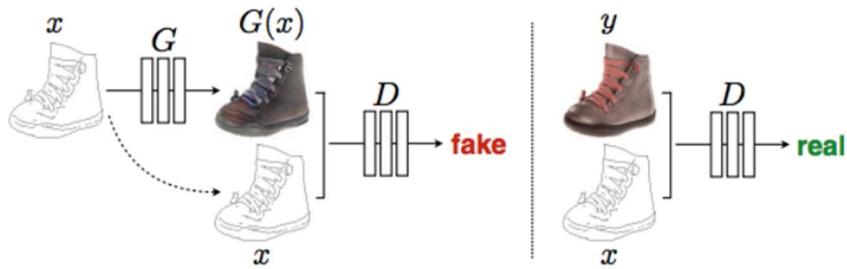
כעת נסמן Tamona אמיתיות השיכת לסקיצה ב- y , ונרשום את פונקציית המחר כשי חלקיים נפרדים – cross entropy ומודרך אוקלידי L_1 בין Tamona המקורי לבין הפלט:

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

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

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

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

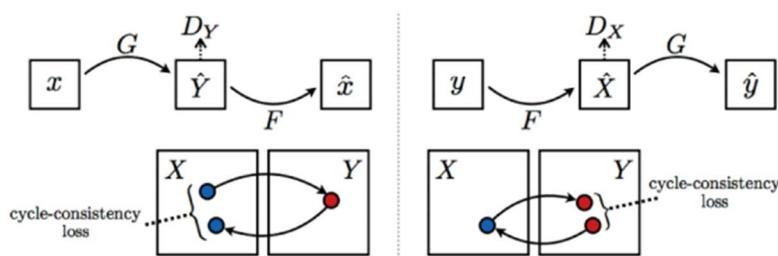


.Image-to-Image Translation - Pix2Pix

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 הראשוני 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 שמנס את התמונה שהתקבלת הינה אמיתית או לא (בעבור הדומין של y). ניתן לבצע את התהליכי זהה באופן דו-אי עבור y – מכנים את 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 , ניתן לקבל חזרה את x אם נwrócić את y דרך F מתוך ציפוייה לקבל $x \approx (G(y))$. התהליכי של השוואת הכנסה למוצא נקרא cycle consistency, והוא מוסיף עוד איבר לפונקציית המבחן, שמטרתו לסייע עד כמה שניתן את המרחק בין התמונה המקורית לתמונה המשוחזרת:

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

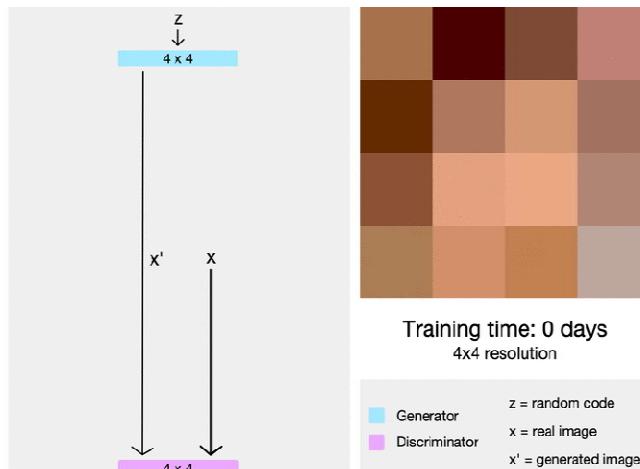


.CycleGAN

7.2.6 Progressively Growing GAN (ProGAN)

כאמור לעיל, עבור דומיין של תמונות, הגיוני להשמש ברשומות קונבולוצייה עבור יצירת תמונות חדשות, וזה הרעיון הבסיסי שמאחורי DCGAN. למרות היכולת המרשימה של DCGAN ביצירה של תמונה באיכות גבוהה, יכולת זאת מוגבלת לתמונות בגודל מסוים. ככל שהיא רחלוצייה של תמונה גבוהה יותר, כך יותר קל להבחן אם תמונה זו אמיתית או נזרה על ידי רשת גברטיבית. בעוד ש-DCGAN מצליח ליצור תמונות שנראות אונטניות יותר, כמו למשל רחלוצייה של 64×64 , 32×32 , 16×16 , 8×8 , 4×4 , 2×2 , 1×1 , והוא היה הראשון שפץ את מחסום הרחלוצייה והצליח ליצור תמונות איזומטריות מואוד (במאמר המקורי של ProGAN – עד רחלוצייה של 1024×1024) בלי שהוא ניתן להבחן שתמונות אלה סינטטיות. אולם עוד לפני ProGAN היו GANs שהצליחו ליצור תמונה בעלת רחלוצייה גבוהה מתמונה אחרת ברחלוצייה גבוהה (az2pix), אך זו ממשאה אחרת, מכיוון שבשבילה צריך רק למודד לשנות תכונות של תמונות קלות, ולא לייצר תמונה חדשה לגמרי מופיע.

הרעיון העיקרי מאחורי ProGAN, שהוצע ב-2017 על ידי חוקרים מחברת Nvidia, הינו לייצר תמונות ברחלוצייה 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 ברחלוצייה משתנה – בהתחלה הוא מתאמן לייצר תמונות ברחלוציות מאוד נמוכה (4×4), לאחר מכן המשיכו לייצר תמונות ברחלוצייה 8×8 , אחר כן 16×16 , וכן הלאה עד יצירה של תמונה ברחלוצייה של 1024×1024 .



איור 7.11 ארכיטקטורת ProGAN.

כדי לאמן GAN ליצור תמונות בגודל 4×4 , התמונות מוסט האימון הוקטנו לגודל זה (down-sampling). אחרי שה-GAN לומד ליצור תמונות בגודל 4×4 , מושגים לו עוד שכבה המאפשרת להכפיל את גודל התמונות המיצירות, קרי ליצור תמונות בגודל 8×8 . יש לציין שהאימון של הרשת עם השכבה הנוספת מתחילה עם המשקלים שאומנו קודם לכן, אך לא "מקפאים" אותם, ככלומר הם מעודכנים גם כן תוך כדי אימון הרשת בשביל ליצירת תמונה ברזולוציה כפולה. הגדלה הדרגתית של הרזולוציה מאלצת את הרשתות להתמקדש תחילה בפרטים ("הגס") של התמונה (דפוסים בתמונה מוטשטשת מאוד). לאחר מכן הרשת "לומדת" לבצע "down-sampling" (להכפיל את הרזולוציה) של התמונות המוטשטשות האלה. תהליך זה משפר את איכות התמונה הסופית כיון שבאופן זה הסבירות שהרשת תלמד דפוסים שגויים קטנה משמעותית.

7.2.7 StyleGAN

StyleGAN, שיצא בשליחי בשנת 2018, מציע גרסה משודרגת של ProGAN generator. עם דגש על רשת ה-*latent* (generator) המאמר שלו לב Ci היתרונות הפוטנציאלי של שכבות ProGAN המייצרות תמונה בצורה הדרגתית נובע מיכולתו לשלוט בתוכנות (מאפיינים) ויזואליות שונות של התמונה, אם משתמשים בהן כראוי. ככל שהשכבה והרזולוציה נמוכה יותר, כך התוכנות שהיא משפיעה עליה גסות יותר.

למעשה,_styleGAN הינו GAN הראשון שנותן יכולת לשולוט במאפיינים ויזואליים (אומנם לא בצורה מלאה) של התמונה הנוצרת. מחברי StyleGAN חילקו את התוכנות הוויזואליות של תמונה ל-3 סוגים:

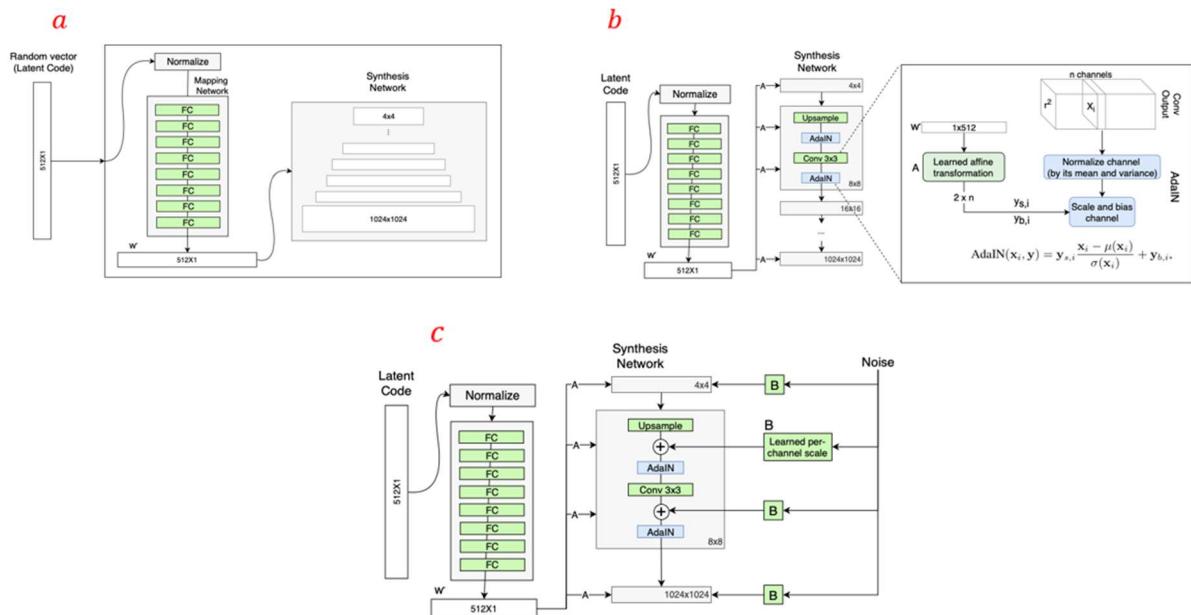
- **גס:** משפיע על תנוחה, סגנון שיער כללי, צורת פנים וכו'.
- **אמצעית:** משפיע על תווי פנים עדים יותר, סגנון שיער, עיניים פקוחות/עצומות ועוד.
- **רזולוציה דקה:** משפיעה על צבע (עיניים/שיער/עור) ועל שאר תכונות המיקרו של תמונה.

כדי להעניק ל-*GAN* את היכולות האלו, נדרש מספר שינויים ביחס לארכיטקטורה של ProGAN (נתאר רק את שלושת השינויים החשובים ביותר כאן):

- **הוספת רשת מיפוי:** מטרת רשת המיפוי היא קידוד וקטור הקלט לווקטור ביןיהם w (הנקרא וקטור סגןון) אשר האיברים השונים שלו שולטים בתכונות ויזואליות שונות של התמונה הנוצרת. זהו תהליך לא טריוויאלי מכיוון שהיכולת של הרשת לשולוט בתכונות ויזואליות באמצעות וקטור הקלט הינה מוגבלת. הסיבה לכך טמונה בעובדה שווקטור הקלט נאלץ "לעקוב אחר ציפיות ההסתברות של סט האימון" שגורם לתופעה הנקראית (FE - feature entanglement) (ערובה מאפיינים). FE בין תכונות צבע השיער והמגדר יכול להופיע אם למשל בסט האימון יש מגמה כללית של גברים עם שיער קצר ונשים בעלות שיער ארוך. במקרה זה הרשת תלמד שגברים יכולים להיות בעלי שיער קצר בלבד ולהיפך אצל נשים. כתוצאה לכך, אם "נשחק" עם רכיבי וקטור הקלט כדי ליצר תמונה של גבר בעל שיער ארוך, בסופו של דבר מגדרו ישתנה גם כן וכן תמונה של אישה.
- רשת המיפוי שהתווסף לארכיטקטורה הופכת את וקטור הקלט לווקטור ביןיהם w שאינו צריך לעקוב אחר התפלגות של סט האימון, וכך יש פחות ערבוב המאפיינים. במקרים אחרים, רשת זו מאפשרת את היכולת לשולוט במאפיינים ויזואליים של התמונה הנוצרת באמצעות שני רכיביו של וקטור w . רשת המיפוי מורכבת משמונה שכבות FC וגודל הפלט שלה זהה לגודל הקלט.

- **החלפת BN ב-AdaIN:** רשותות הקונבולוציה של ה-generator, שנוועדו 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 גאומטרית אלא וקטור דטרמיניסטי עם רכיבים נלמדים. וקטורי הרעש מתווספים ישירות לפליטים של ערכיו קונבולוציה ברשתות ה-generator כאשר העוצמה שלהם נŁmdת לכל ערך בנפרד. שימוש בוקטור קלט דטרמיניסטי במקומ וקטור אקריאי מקל כל הנראה על הפרדת המאפיינים על ידי רשת המיפוי (ויתר קל לעשوت זאת על וקטור קבוע מאשר להתאים את משקליו רשת המיפוי לווקטור 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 יכולה לשולט 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 מתבआ בכך שלמשטח עשוי להיות צורה מאוד מורכבתיחסית לתת-מרחב וקטורי. משתמש מכך שניתן ליצור דעתה רב מממד' על ידי טרנספורמציה של וקטור מורכב בעל מממד נמוך (וקטור לטנטני). למשל, ניתן באמצעות רשת נירונים ליצור תמונה בגודל $12k > 3 \times 64 \times 64$ פיקסלים מוקטור באורך 100 בלבד. זאת ועוד, שאם התפלגות התמונות של הרשות האנרגטיבית וגם ההתפלגות של הדעתה האמיתית נמצאים ב"משטח בעל ממך 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 ממך 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 הצינויים שהוא מקבל מה-discriminator (דרך פונקציית discriminator של ב-GAN). אבל אם ה-generator מוציא צינויים מאדנו נומוכים (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 שהוסבר, הבעיה של המרחק בין היריעות משתקפת במבנה של פונקציית המחר, וכךון שכר, ניתן לנסות ולפטור את הבעיה מהשורש על ידי שימוש בפונקציית מחיר יותר מתאימה. לשם כך ראשית נסמן את התפלגות הדטה האמיתית $b_{-r, k}$, ואת התפלגות הדטה הסינטטי המיצר על ידי ה-generator $b_{-g, k}$. 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 $\geq p \in [1, \infty]$ שתי מידות הסתברות x, y 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, n) עם פונקציות שליליות (marginal) השווות ל- μ, 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 k ישר, 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 $x = 4$, יהיה נדרש 0.3_{kg} בנקודת $x = 5$, 0.3_{kg} בנקודת $x = 8$, 0.2_{kg} בנקודת $x = 0$, ושאר המשקלות (0.2_{kg}) יהיו בנקודת $x = 8$.

כמובן שיש הרבה דרכים לבצע את היזמת המשקלות, ונרצה למצאו את הדרך היעילה ביותר. לשם כך נגידיר באמצעות מכפלה של משקל במרחב M 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:

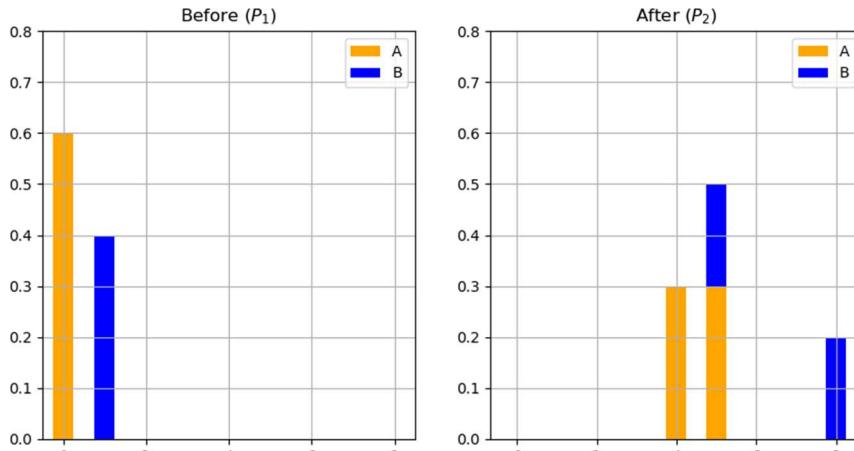
$$\text{מועברים מ-0} = x = 4 - 0 = 4, \text{ כאשר המאמץ הנדרש לכך הינו } 0.3_{kg}.$$

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

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

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

$$\text{סך המאמץ המינימלי שווה במקרה זה ל: } 4 + 5 + 4 + 7 = 16.$$



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 לתפלגיות p_1, p_2 , המוגדרות באופן הבא:

$$p_1(x) = \begin{cases} 0.6, & x = 0 \\ 0.4, & x = 1 \\ 0, & \text{else} \end{cases}, \quad p_2(x) = \begin{cases} 0.3, & x = 4 \\ 0.5, & x = 5 \\ 0.2, & x = 8 \\ 0, & \text{else} \end{cases}$$

השאלה כיצד ניתן להעביר מסה הסתברותית מ- p כרך שתתקבל 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 $1 = k$ 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 המכפלת 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 שהסבירנו מהו מרחק ורשותן \mathcal{D} ומהו מרחק EM, ניתן להבין כיצד אפשר להשתמש במקרים אלו עבור פונקציית מחיר של GAN. נציין כי מרחק \mathcal{D} בין מידות ההסתברות מתחשב בתכונות של הקבוצות עליהם מידות אלו מוגדרות בצורה מפורשת, על ידי המתחשב במרקםם בין קבוצות אלו. תכוונה זו היא למעשה בדיקת מה שציבור 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 J_S . 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 זה מצויין, אך למעשה זה לא מספיק, כיון שצריך למצוא דרך לחשב את \mathcal{D}_g , 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 המכפלת, וצריך למצוא דרך להשתמש בו כפונקציית מחיר. בשבייל לבצע זאת, ניתן להשתמש בצורה דואלית של \mathcal{D}_W עבור $1 = p$ – שיווין RK (Rubinstein-Kantorovich) לערך \mathcal{D}_W באופן הבא:

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L} E_{x \sim p} [f(x)] - E_{x \sim p_g} [f(x)]$$

כאשר (x, f) הינה פונקציית K-לייפשיץ רציפה (כלומר, פונקציה רציפה עם קצב השתנות החסום על ידי K). כתוב נניח ש-(x, f) הינה פונקציית K-לייפשיץ המטא-ריצפה discriminator בעל סט הפרמטרים w . ה- z -generator מחשב באופן מקרוב את המרחק בין התפלגותים באופן הבא:

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

פונקציית מחיר זו מודדת את המרחק \mathcal{D}_W בין התפלגות \mathcal{D}_g ו- p , וככל שפונקציה זו תקבל ערכיים 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- g_θ), וככל שפונקציית המחיר הולכת וקטנה, כך g מתקרוב יותר ל- p_r .

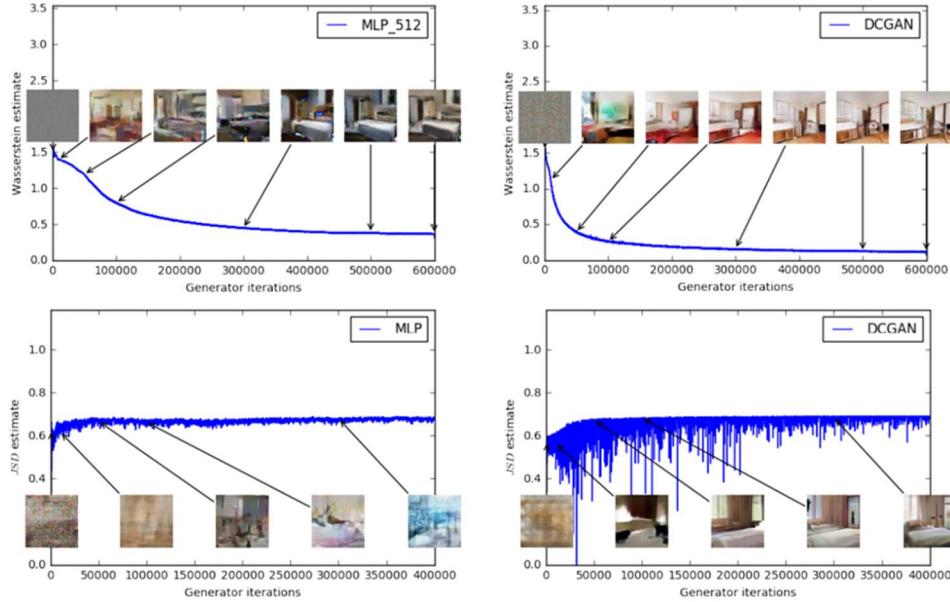
כאמור, תנאי הכרחי לשימוש במרקם זה בפונקציית המחיר הינו שהפונקציה תהיה K-לייפשיץ רציפה. מסתבר שקיים תנאי זה אינו ממשה קלה כל. כדי להבטיח את קיומו, המאמר המוקורי הצעיר לבצע קטימה של משקלי ה-discriminator לטוויה סופי מסוים, נניח $[0.01, 0.01]$. ניתן להראות כי קטימה זו מבטיחה את ש- \mathcal{D}_w ה-לייפשיץ רציפה. אולם, כמו שכותבי המאמר מודים בעצמם, ביצוע קטימה בכדי לדאוג לקיום תנאי לייפשיץ יכול לගרום לביעות אחרות. למעשה, כאשר חלון הקטימה של המשקלים צר מדי, הגדריאנטים של Wasserstein GAN עלולים להתאפס, מה שיאט את תהליכי הלמידה. מצד שני, כאשר חלון זה רחב מדי, 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 ריציפות-לייפשיץ.
- ב. פונקציית מחיר המסתמכת על \mathcal{D}_W במקום על J_S .

תהליכי הלמידה מתבצע באופן הבא – לאחר כל עדכון משקלים של ה- z -generator (gradient ascent,gradient descent) 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 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 J_S .



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.

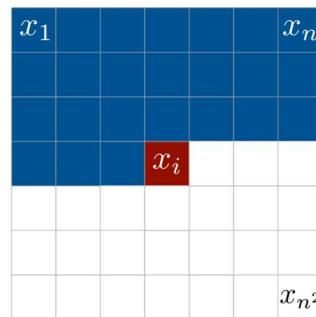
נקודה נוספת שיכולה להסביר את ההצלחה הייחסית של השימוש ב- \mathcal{D}_D נובעת מכך שמטריקת JS חלשה יחסית למטריקת SJ , וננסה להבהיר נקודה זו.

באופן אידיאלי, כאשר אנו מאמנים מודל הינו רצים להיות בטוחים שאם ננרג בצורה נאותה ובכל צעד נעדק 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 זיה 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 לאחר 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. העבודה ש- \mathcal{D}_{JS} בבעצם אומרת שיתכן ויש בעיות שעבורן תתקבל תוצאה אופטימלית עבור W אך לא עבור JS .

7.3 Auto-Regressive Generative Models

משפחה נוספת של מודלים גנרטיביים נקראת VAE, ובדומה לו Auto-Regressive Generative Models. מודלים אלו מוצאים התפלגות מפורשת של מרחב מסוים ובעזרת התפלגות זו מייצרים דאטה חדש. עם זאת, בעוד VAE מוצא קירוב להתפלגות של המרחב הלטני, 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

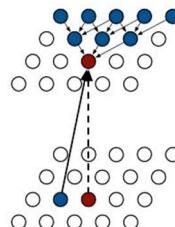
אפשרות אחת לחשב את (x) היא להשתמש ברכיבי זיכרון כמו LSTM עבור כל פיקסל. באופן טבעי היינו רצים לחבר כל פיקסל לשכנים שלו:

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

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

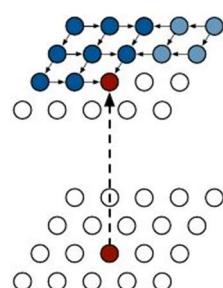
במקרה להשתמש במצב החבוי של הפיקסל הקודם, ניתן להשתמש רק בשורה שמעל הפיקסל אותו רצים לחשב. שורה זו עצמה מחושבת לפניה על ידי השורה שמעליה, ובכך למעשה לכל פיקסל יש receptive field של מושלש. בשיטה זו ניתן לחשב באופן מקבילי כל שורה בנפרד, אך יש לכך מחיר של איבוד הקשר בין פיקסלים באותה שורה (loss context).



איור 16 Row LSTM 7.16 – כל פיקסל מחושב על ידי $\geq k$ פיקסלים בשורה שמעליה.

Diagonal BiLSTM

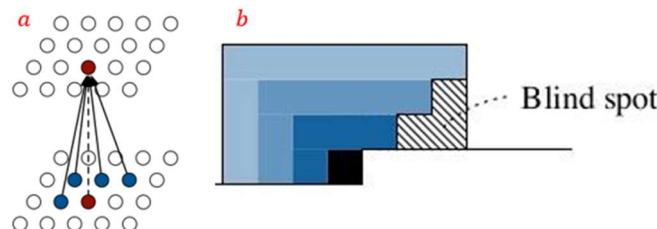
כדי לאפשר גם חישוב מקבילי וגם שמירה על קשר עם כל הפיקסלים, ניתן להשתמש ברכיבי זיכרון דו כיווניים. בכל שלב מחשבים את רכיבי הזיכרון משני הצדדים, וכך כל פיקסל מחושב גם באמצעות הפיקסל שלידו וגם על ידי זה שמעליו. באופן זה receptive field גודל יותר ואין loss context, אך החישוב יותר איטי מהשיטה הקודמת, כיוון שהשורות לא מחושבות בפעם אחת אלא כל פעם שני פיקסלים.



איור 7.17 Diagonal BLSTM – כל פיקסל מחושב על ידי $3 \geq k$ פיקסלים בשורה שמעליו. כדי לשפר את השיטות המשתמשות ברכיבי זיכרון ניתן להוסיף עוד שכבות, כמו למשל Residual blocks כדי להפריד את התלות של העורכים השונים של כל פיקסל.

7.3.2 PixelCNN

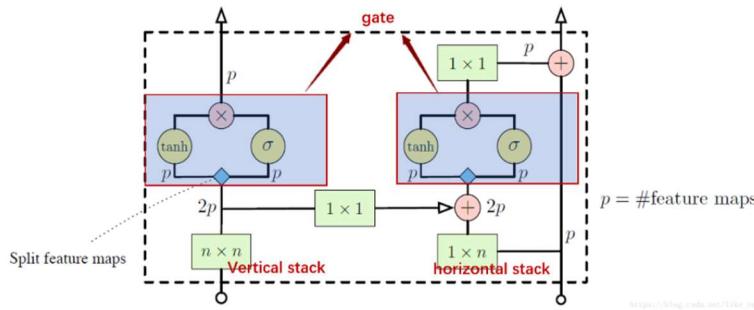
הчисIRON העיקרי של PixelRNN נובע מהאימון האיטי שלו. במקום רכיבי זיכרון ניתן להשתמש ברשת קונבולוציה, ובכך להאיץ את תהליכי הלמידה ולהגדיל את ה-receptive field. גם בשיטה זו מתחילה מפהיקסל הפינתי, רק כעת הלמידה היא לא בעזרת רכיבי זיכרון אלא באמצעות שכבות קונבולוציה. היתרון של שיטה זו על פני PixelRNN מटבṭא בקיצור משמעותי של תהליכי האימון, אך התוצאות פחות טובות. חיסרון נוסף בשיטה זו נובע מהמבנה של המנסנים ו-h-field – כל פיקסל מtabסס על שלושה פיקסלים שמעליו, והם בתורם כל אחד תלוי בשלושה receptive field. מבנה זה מנתק את התלות בין פיקסלים קרובים יחסית אך אינם ב-field.



איור 7.18 receptive field (a) של PixelRNN. (b) של PixelCNN – ניתוק בין פיקסלים יחסית קרוביים.

7.3.3 Gated PixelCNN

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



$$y = \tanh(w_f * x) \odot \sigma(w_g * x)$$

7.3.4 PixelCNN++

שיפור אחר של PixelCNN הוצע על ידי OpenAI, והוא מבוסס על מספר מודיפיקציות:

- שכבת SoftMax שקובעת את צבע הפיקסל צורכת הרבה זיכרון, כיוון שיש הרבה צבעים אפשריים. בנוסף, היא גורמת לארדיינט להתפס מהר. כדי להתגבר על כך ניתן לבצע דיסקרטיזציה לצבעים, ולאחר מכן קטע יותר. באופן זה קל יותר לקבוע את ערכו של כל פיקסל, ובנוסף תהליכי האימון יותר יעיל.
- במקרה לבצע בכל פיקסל את ההתניתה על כל צבע בנפרד (כפי שהראינו בפתחה), ניתן לבצע את ההתניתה על כל הצבעים יחד.
- אחד האתגרים של PixelCNN הוא יכולת המוגבלת למצוא תלויות בין פיקסלים רחוקים. כדי להתגבר על כך ניתן לבצע sampling down, ובכך להפחית את מספר הפיקסלים בכל מסנן, מה שמאפשר לשמור את הקשרים בין פיקסלים בשורות רחוקות.

- בדומה ל-Net-U, ניתן לבצע חיבורים בעזרת Residual blocks ולשמור על יציבות במהלך הלמידה.
- שימוש ב-Dropout לצורף רגולרייזציה והימנעות מ-fitting.

7. References

VAE:

<https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>

<https://jaan.io/what-is-variational-autoencoder-vae-tutorial/>

<https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html>

GANs:

<https://arxiv.org/abs/1406.2661>

<https://arxiv.org/pdf/1511.06434.pdf>

<https://phillipi.github.io/pix2pix/>

<https://junyanz.github.io/CycleGAN/>

<https://arxiv.org/abs/1710.10196>

<https://arxiv.org/abs/1812.04948>

<https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431>

<https://arxiv.org/abs/1701.07875>

AR models:

<https://arxiv.org/abs/1601.06759>

<https://arxiv.org/abs/1606.05328>

<https://arxiv.org/pdf/1701.05517.pdf>

<https://towardsdatascience.com/auto-regressive-generative-models-pixelrnn-pixelcnn-32d192911173>

https://wiki.math.uwaterloo.ca/statwiki/index.php?title=STAT946F17/Conditional_Image_Generation_with_PixelCNN_Decoders#Gated_PixelCNN

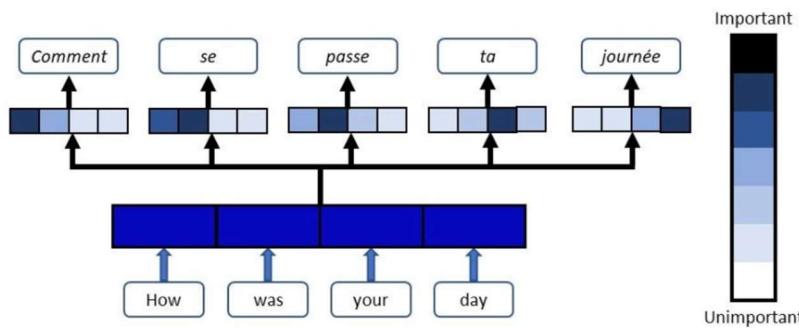
8. Attention Mechanism

8.1 Sequence to Sequence Learning and Attention

8.1.1 Attention in Seq2Seq Models

ניתוח סדרות בהן יש קשר בין האיברים יכול להיעשות בעזרת רשות עם רכיבי זיכרון, כפי שהואר בפרק 6. ברשותה אלו הסדרה הנכנסת לרשות עוברת דרך היצר וקטור בגודל ידוע מושך המציג את הסדרה המקורי, תוך התחשבות בסדר של איברי הסדרה ובקשר ביניהם. לאחר מכן וקטור זה עובר ב-decoder שיכל לפענח את המידע שיש בווקטור ולהציג אותו בצורה אחרת. למשל בתרגום משפה לשפה – מודל של seq2seq מקודד משפט בשפה אחת לווקטור מסוים ולאחר מכן מפענח את הווקטור לשפה השנייה.

הדרך המקובלת ליצור את הווקטור ולפענח 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 – במקומ 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 במוחא קשורה לכל אחת מהמלילים במשפט המקורי. באופן זה כל איבר בסדרת הפלט משקל כל אחד מאיברי סדרת הפלט. מנגנון זה נקרא .attention



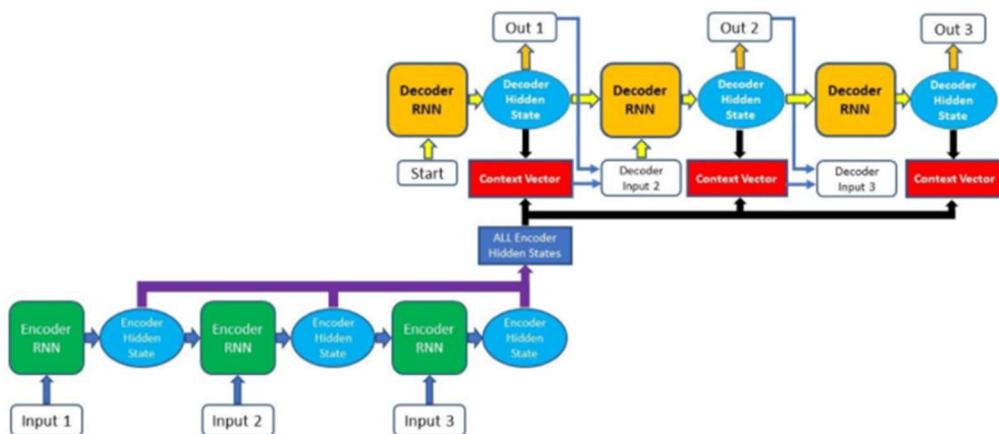
איור 8.1 מנגנון attention – נתינת משקל לכל אחת ממלילות הקלט בהתאם לכל אחת ממלילות הפלט.

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 RNN לבין 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 נקראת



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 אותם יחד עם הממצבים החבויים של ה-encoder. עבור כל אחד מאיברי סדרת הפלט מחשבים alignment score בין המצב החבוי של רכיב הזיכרון הקודם בסדרת הפלט לבין כל הממצבים החבויים של ה-encoder, וכך יוצרים context vector שבעזרתו מחשבים את הפלט עבור האיבר הנוכחי ב-decoder. ביצוע הפעולה זו הוא הלב של מנגנון ה-attention, כיוון שהוא קשור בין הפלט 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-encoder וה-decoder,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, ה-decoder והחיבור ביניהם. את התוצאה מעתיריים 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 שיש 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-decoder.

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:

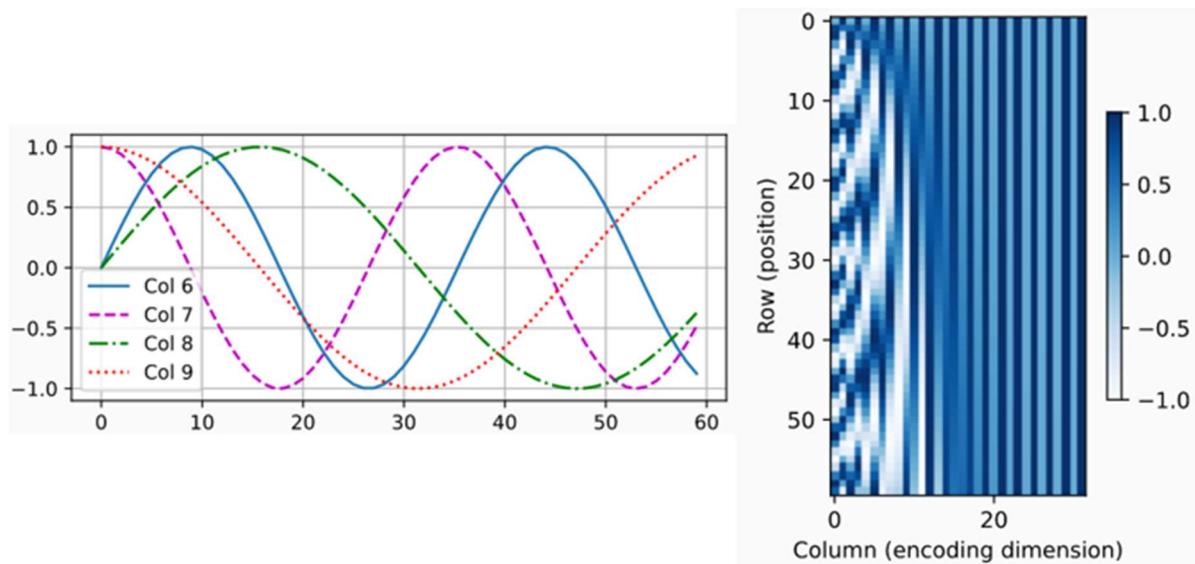
$$p_t(i) = \begin{cases} \sin(\omega_k t), & i \text{ is even} \\ \cos(\omega_k t), & i \text{ is odd} \end{cases}, \omega_k = \frac{1}{10000^{\frac{2k}{d}}} \rightarrow p_t = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \sin(\omega_2 t) \\ \cos(\omega_2 t) \\ \vdots \\ \sin\left(\omega_{\frac{d}{2}}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

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 אותם בצורה בינארית, נוכל לראות שככל שהיבט יש משקל גדול יותר, כך הוא משתנה בתדריות נמוכה יותר, ולמעשה תדריות שינוי הביט היא אינדיקציה למיקום שלו.

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

איור 8.3 ייצוג בינארי של מספרים. ה-MSB משתנה בתדרות הכח נמוכה, ואילו ה-LSB משתנה בתדרות הכח גבוהה.

כיוון שמתusalem במספרים שאינם בהכרח שלמים, הייצוג הבינארי של מספרים שלמים הוא יחסית פשוט, ולכן רקחת 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 שמתווסףת לכל איבר בסדרה המקורית ניתן לקבל אינדיקציה על מיקומו.



איור 8.4 Positional encoding. דוגמא למספר פונקציות בעלות תדרות הולכת וקטנה, בהתאם לאיבר אותו הן מייצגות (שמאלו). המuschא לקצב השני של כל פונקציה בהתאם למיקום של האיבר אותו היא מייצגת – מען גרסה רציפה לקצב 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, ולה חשב 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 בסמכיות, ובתמונה זה יכול להיות פיקסליהם דומים). דמיון בין איברים נמדד על פי הקשר שיש ביניהם, והוא מחושב באמצעות מכפלה פנימית בין וקטוריו ייצוג של האיברים. כל מכפלה פנימית בין שני איברים נותנת מוקדם שהוא מספר ממשי, וצר ניתן לסכם את מכפלת כל המוקדים באיברים המקוריים, ולקבל ייצוג חדש לאיבר המקורי.

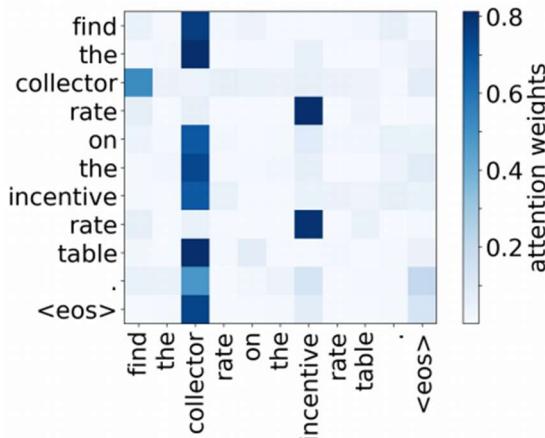
המכיל גם קשר בין האיבר הנוכחי לאיברים דומים בסדרה. במיילים אחרים, ניתן להסתכל על וקטור המכיל את הקשרים של איבר מסוים בסדרה כדי החדש המשקף את הקשרו עם שאר איברי הסדרה.

באופן פורמלי, בשביל לחשב את self-attention המטריצות 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 המקורית x יוצר שורה בכל אחת מהמטריצות. כאשר לוקחים את השורה $x_i \cdot Q = q_i$, ומכפילים אותה בכל אחת מהשורות במטריצה K , מקבלים וקטור חדש, שכל איבר j בוקטור אומר עד כמה יש קשר בין האיברים i, j , בסדרה המקורית. ביצוע ההכפלה זו עברו 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 ה-embedding כדי לשמר על יציבות הגרדיאנט, ולאחר מכן מנורמלים על ידי SoftMax. באופן זהה מקבלים מטריצה של מספרים בטוחווות $[0, 1]$, המייצגים כאמור את הקשר בין כל שני איברים בסדרה המקורית. נסמן כל איבר במטריצה ב- w_{ij} , ונוכל לקבל אותו יישור על ידי הנוסחה:

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

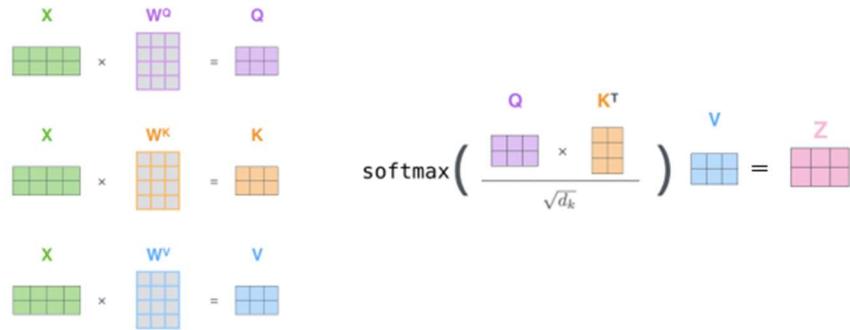


איור 8.5 מטריצת משקלים של המשפט "Find the collector rate on the incentive rate table". כל קשר בין שתי מילים חזק יותר-כך המשקל ביניהם גבוה יותר. כמו כן שיש גם שימוש לסדר – המשקל בין "collector" ל-"Find" שונה מאשר בין "collector" ל-"Find".

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

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

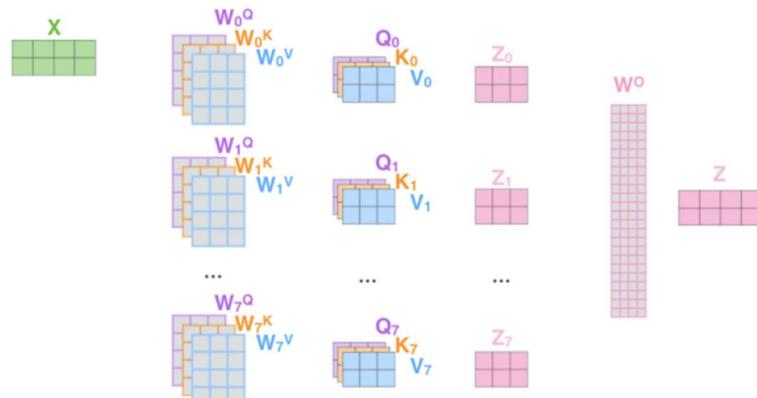
סדרה המתקבלת z היא למעשה ייצוג חדש של הסדרה המקורית, כאשר כל איבר z_i מייצג איבר בסדרה המקורית יחד עם מידע על הקשרים ביןו לבין שאר איברי הסדרה. את הסדרה המתקבלת ניתן להעביר ב-*decoder* המכיל שכבות נוספות, ובכך לבצע כל מיני משימות, כפי שיואר בהמשך.



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

8.2.3 Multi Head Attention

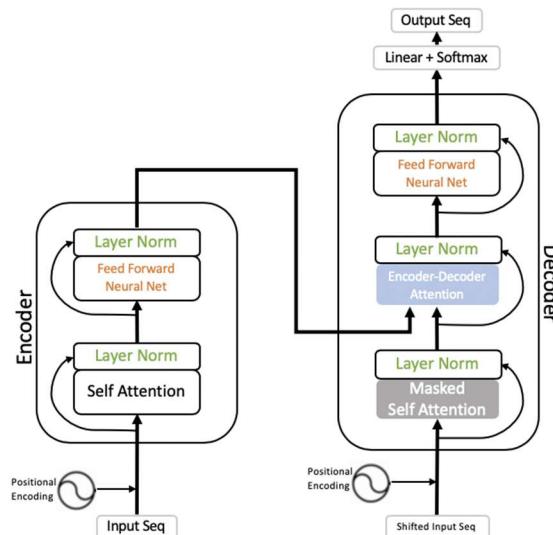
ניתן להשתמש במנגנון self-attention מספר פעמים במקביל. כל פעם מקבלים שלוש מטריצות (Q_r, K_r, V_r), ובעזרתهما מחשבים את הייצוגים החדשניים של איברי הסדרה (attention score). כל מנגנון זה נקרא attention head, וסביר במקביל של כמה attention heads נקרא Multi-head attention. באופן זה לכל איבר כניסה x_i יש כמה ייצוגים שונים Z_i , אותן ניתן להכפיל במטריצה משקלים W ולקבל את הייצוג המשוקל של אותו איבר באמצעות attention heads רבים.



איור 8.7 Self-attention with 8 heads

8.2.4 Transformer End to End

בעזרת 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 באIOR ה-transformer מרכיב משני חלקים – encoder-ו-decoder.encoder – encoder – positional encoding מסויים ומצבע עלייה embedding מסויים. לאחר מכן הסדרה עוברת דרך residual block של self-attention, ומתקבלת התוצאה $x + \text{attention}(x)$. על תוצאה זו מבצעים layer normalization (כפי שהואר בפרק 5.1.4). לאחר מכן יש residual block נסוף, המכיל שכבת fully connected normalization (כפי יוצג בהמשך).encoder-encoder

decoder בניו בוצרה מאוד דומה, עם שני הבדלים העיקריים: הקלט שלו הוא איברי הפלט שהו עד כה, ובנוסף יש בתחילת ה-decoder שכבה של Masked self-attention. שכבה זו מקבלת את כל איברי הפלט שהו עד כה, ומטרת ה-decoder היא ללמידה עצמאו 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 שבהו מטריצות **Query**, **Key** ו-**Value** מתקבלות מה-encoder ו-**decoder**. השכבה שילוב של encoder וה-decoder מקבלת מטריצות **Query** ו-**Key** מה-encoder ו-**Value** מה-**decoder**. מטריצת **Query** מוחפשים דמיון בין איברים של סדרה אחת לבין איברים של סדרת הפלט (ביצוגם שלham לאחר שכבת masked). מטריצת **Key** מוחפשים דמיון בין איברי סדרת הפלט (ביצוגם שלham לאחר שכבת masked). מטריצת **Value** מוחפשים דמיון בין איברי סדרת הפלט (ביצוגם שלham לאחר שכבת masked). מטריצת **Query** מוחפשים דמיון בין איבר אחד לבין איברים של סדרת הפלט (ביצוגם שלham לאחר שכבת masked). מטריצת **Key** מוחפשים דמיון בין איבר אחד לבין איברים של סדרת הפלט (ביצוגם שלham לאחר שכבת masked). מטריצת **Value** מוחפשים דמיון בין איבר אחד לבין איברים של סדרת הפלט (ביצוגם שלham לאחר שכבת masked).

נich דוגמא ממאמר שנקרא [DETR](#) המראה כיצד ניתן להשתמש transformer בשביל **ZhI** אובייקטים בתמונה. בשלב הראשון לוקחים כל פיקסל בתמונה ומשווים אותו לשאר הפיקסלים (זהו בעצם המכפלה $K \cdot Q$). באופן זה ניתן למצוא אזורים דומים ושוניים בתמונה, כאשר דמיון ושוני זה לאו דווקא פיקסלים עם ערכים קרובים, אלא זה יכול להיות למשל שני אזורים שונים בפנים של אדם. לאחר מכן מיצרים ייצוג חדש לתמונה, באמצעות המשקלים והכפלתם **V**. בשלב זה למעשה אפשר לבצע **ZhI** של אובייקטים, בלי לדעת מה הם אוטם אובייקטים. בשביל לבצע ייצוג לכל אובייקט שゾהה, מעבירים את הייצוג החדש של התמונה ב-**decoder**, כאשר ה-[Query](#) שמכניסים זה כל מיני ליבלים אפשריים, ומוחפשים מבין כל ה-[Query](#) את הפלט של ה-**decoder** שמצילח לייצר תמונה שהći דומה ל-

אם למשל יש תמונה גדולה ויש איזור מסוים בו יש חתול, אז ה-encoder מוצא איפר החתום בתמונה, וה-decoder משווה את האזור הזה לכל מיני חיוט אפשרויות. כל **Query** שלאי יהיה חתול, המכפלה $K \cdot Q$ תהיה קרובה ל-0, וה-decoder יזהה שה-**Query** הנוכחי לא מתאים שזזהה. אך כאשר ה-**Query** יהיה חתול, אז כיוון ש- $K \cdot Q = \sum_{j=1}^n w_{ij} z_i$ דומימ אחד לשני, המכפלה $K \cdot Q$ תביא לכך שה"יצוג החדש $\sum_{j=1}^n w_{ij} z_j$ כן יהיה דומה לחתום. יציג זה עבור בשכבה FC, ולאחר מכן ה-SoftMax יסוציא את התמונה החזו כחתול.

8.2.5 Transformer Applications

transformer ה-*state-of-the-art* בוצעו במאזן מושגים, והוא הינו השרה להמן "শומים הנשנים על attention בלבד. מלבד הרמה הגבוהה של הביצועים, תהליכי האימון של transformer מרשחות קוגנולוציה או רשותות רקורסיביות. כמו במקרים אחרים, גם עם transfer transformer ניתן לבצע learning, כלומר ללקחת transformer שאומן על משימה מסוימת, ולהתאים אותו למשימה חדשה שדומה למשימה המקורית. בפועל לא כל היישומים משתמשים בכל ה-*transformer*, אלא בהתאם למשימה לוקחים חלקים מסוימים שלו ובודים בעזרתם מודל עבור משימה מסוימת. נזכיר מספר דוגמאות:

Machine Translation – תרגום משפטים בין שפות שונות הוא שימוש טריזואלי של ה-transformer המלה. המשימה היא ללקחת משפט ולהוציא משפט בשפה אחרת, וזה נעשה באמצעות ייצוג המשפט המקורי באמצעות self-attention ולאחר מכן המרתו באמצעות Encoder-Decoder Attention לשפה אחרת.

במהלך כל פעם באופן רנדומלי עושים masking למלילים מסוימות, ומטרת המודול הוא לחזות את המילים החסרות. בהם כל שפה הוא פונקציה המקבלת כקלט טקסט ומחזירה את התפלגות למילה הבאה על פי כל המיללים במלון. השימוש הינו מוכר ואינטואיטיבי של מודל שפה הוא השלה אוטומטי, שמציעה את המילה או המיללים היכי סבירות בהינתן מה שהמשתמש הקליד עד כה. כאשר מבצעים self-attention על משפטים, למעשה מקיבלים ייצוגים חדשים שליהם יחד עם ההקשרים בין המיללים השונים. لكن ה-encoder transformer יכול ליצור מודל שפה, אם מאמנים אותו בצורה מתאימה. המפתחים של BERT בנו encoder מקבל כל מיני משפטים בשני כיוונים – גם מההתחלת לסוף ומהסוף להתחלה, וכן הייצוגים שנלמדו קיבלו קונטקט שלם יותר. בנוסף, הם אימנו את המודול על משפטים

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*, והוא כל פעם מילה אחרת במילון, וכך בעזרתו *attention* בוחנים איזה *Query* מתאים 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

להכnis Region Based CNNs (R-CNN Family)

להכnis מטריקות

https://github.com/taldatech/ee046746-computer-vision/blob/spring20/ee046746_tut_05_deep_semantic_segmentation.ipynb

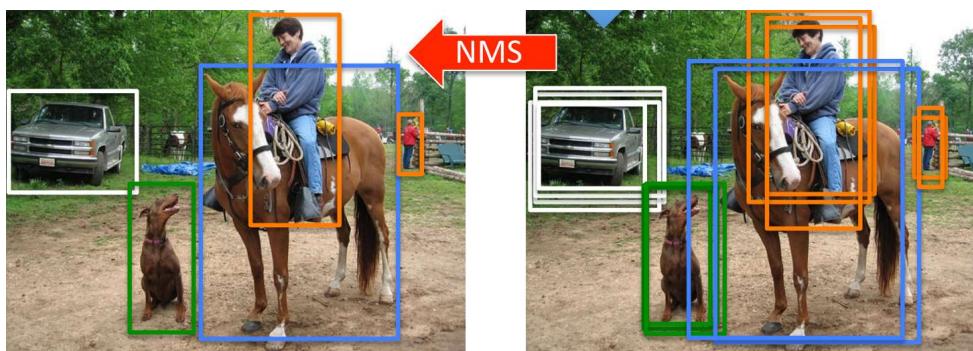
9.1 Object Detection

9.1.2 You Only Look Once (YOLO)

עד שנות 2016 כל הgalais (detectors) המבוססים על מידע عمוקה (כמו למשל R-CNN family) היו גלאים דו-שלביים – השלב הראשון יצר אלף ההצעות (proposals) למסגרות מלכניות החשודות כמכילות אובייקטים, ואלו הוזנו בזה אחר זה לשלב השני אשר דיבק את המסגרות וביצע סיווג לאובייקטים המוכלים בהן. ארQUITוות YOLO, הנגזרת מראשי התיבות של ONLY YOU ONLY LOOK ONCE, הוצאה על ידי ג'וזף רדמן ב-2016, והיתה הgalai הראשון שモרכב משלב יחיד, ובו הרשת מנבאת את המסגרות וגם מסוגגת את האובייקטים שבתוכן בביטחון. בנויסף, ארQUITוות YOLO מתאפיינת במיועט פרמטרים וכמעט פועלות אРИטמטיות. היא אמנת מושלמת על המבנה הרזה והאלגנטית שלה בDIOK נמוך יותר, אך מהירות הגבואה (שנובעת בעיקר מהיעדר האילוץ לעבד מסגרת יחידה בשלב השני בכל מעבר של הלולאה) הפכה את השלב היחיד לארתקטיבית מאוד, במיוחד למבקרים קטנים כדוגמת מכשירי mobile. בעקבות עובודה זו פותחו גלמים רבים על בסיס שלב יחיד, כולל גרסאות מתקדמות יותר של YOLO (הגרסה המתקדמה ביותר כיוון היא 5).

NMS (Non-Maximum Suppression)

כמעט כל אלגוריתם של זיהוי אובייקטים מייצר מספר רב של מסגרות חשודות, כאשר רובן מיותרות ויש צורך לדלול את מספן. הסיבה לכך היא שמספר גדול של מסגרות נובע מיפוי פועלות הgalais. בזכות התכונות הлокליות של פועלות הקונבולוציה, מפת הפיצרים במויאת הgalai ניתנת לתיאור כמטריצת משבצות כאשר כל משבצת שකולה לריבוע של הרבה פיקסלים בתמונה המקורית. רוב הgalais פועלם בשיטת עוגנים (anchors), כאשר כל משבצת במויאת הgalai מנבאת מספר קבוע של מסגרות שעשויה להכיל אובייקט (למשל, YOLOv2 המספר הוא 5, ובגרסאות המתקדמות יותר המספר הוא 3). השיטה זו יוצרת אלפי מסגרות שרק מועטות מהן הן שימושיות. בנויסף – יש ריבוי של מסגרות דומות בסביבת כל אובייקט (למשל – YOLOv3 מנבאת יותר מ-7000 מסגרות לכל תמונה). אחת הדרכים הפופולריות לסנן את אלפי המסגרות ולהשאר רק את המשמעותיות נקראת NMS. בשיטה זו מטבחת השוואה בין זוגות של קופסאות מאותה המחלקה (למשל – חתול), ובמידה שיש ביןיהם חפיפה גבוהה – מוחקים את המסגרת בעלת הווהות הנמוכה הייתר ונשארים רק עם המסגרת בעלת רמת הווהות הגבוהה. שיטה זו נזקנית בחישוב (סיבוכיות פרופורציונלית לריבוע מספר המסגרות) ואני חלק מהמודל המתאים, אך עם זאת הינה אינטואיטיבית יחסית למימוש, ומשום כך נמצאת בשימוש נפוץ בgalais, כולל ב-YOLO.



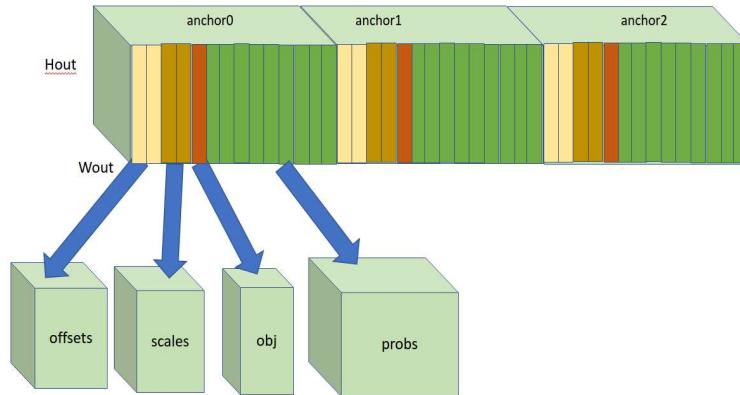
איור 9.1 אלגוריתם NMS.

YOLO Head

כמו רוב הgalais, YOLO הינו מבוסס-עוגנים (anchor-based), שהין תיבות מלכניות קבועות ושונות זו מזו בצורתן. לכל עוגן מוקצה מקטע של פיצרים במפת המזיא של הרשת וכל הניבויים במקטע זהה מקודדים סטיות (offsets)

ביחס לממד הугון. כפי שניתן לראות באירור 9.2, הפיצרים של כל תא מרחבי בModelProperty המוצא מוחולקים למקטעים על פי העוגנים (שלושה עוגנים במרקחה זהה).

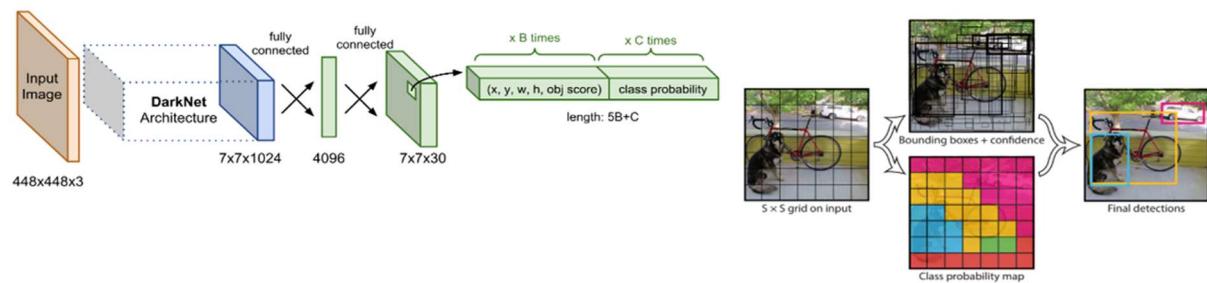
ניבוי הוא מסגרת שמרכזה נמצא בנקודה כלשהי בשטח התא (השקלן לריבוע של מספר פיקסלים בתמונה המקורית). ההסתה המדוקיקת של מרכז המסגרת ביחס לתא ניתנת על ידי שני הפיצרים הראשונים ברכף. לוג הממדים של הקופסה (ביחס לממד הугון) ניתן באמצעות שני הפיצרים הבאים ברכף. הפיצר החמישי לומד את מידת ה-objectness, כפי שהסבירה לעיל. שאר הפיצרים ברכף של הугון הנו הם הסתברויות המותנות לכל מחלקה (אם אוסף הנתונים מכיל 80 מחלקות, יהיו 80 פיצרים כאלה). על מנת לקבל רמת ודאות סופית, יש להכפיל את מדד ה-objectness במדד הסתברות המותנה לכל מחלקה.



איור 9.2 ראש YOLO.

YOLOv1

מודל YOLO מבוססיים על גרסאות Backbone הנקראות Darknet ומשמשות לעיבוד פיצרים מתוך התמונה, ורואה detection המקבל את הפיצרים האלה ומתאמן לייצר מהם ניבויים למסגרות סביב אובייקטים. המודל מחלק את התמונה לרשת בעלת $S \times S$ משבצות, כאשר כל משבצת מנבאת N מסגרות של אובייקטים בשיטת העוגנים, כאמור לעיל. כל ניבוי כולל את מספר ערכיהם: הסטת הקואורדינטות u, v של מרכז המסגרת ביחס למשבצת, הגובה והרוחב של המסגרת, ורמת ה-objectness, כפי שהסבירה לעיל. בנוסף, כל מסגרת מבצעת גם סיוג, כולם מנבאות את רמת הוודאות של השתייכות האובייקט לכל אחת מהמחלקות האפשריות. החידוש באלגוריתם נעוץ בעובדה שחייבי המסגרות ויסווגו לאובייקטים שנעשה במקביל, ולא באופן דו-שלבי. הרעיון הוא להתייחס לסוג האובייקט כדי פיצר שהרשת מנסה לחזות בונוס למקומם וגודלה של המסגרת.



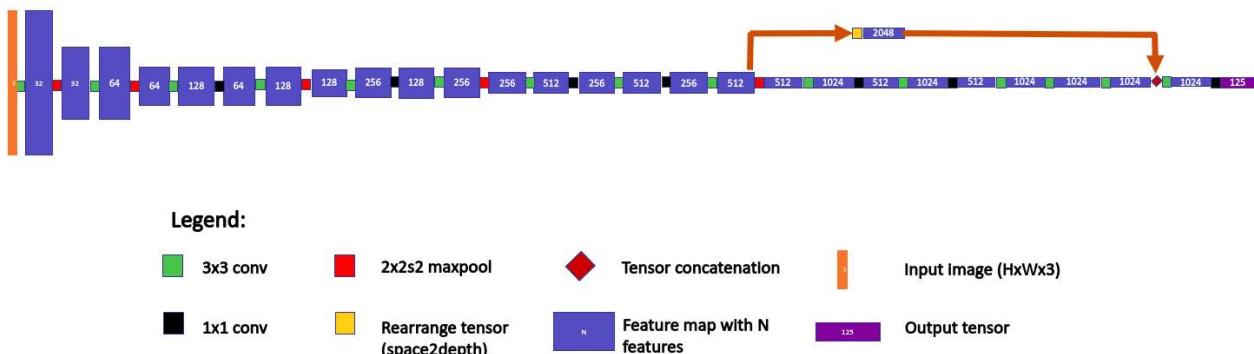
איור 9.3 ארכיטקטורת YOLO.

הgresה הראשונה של ארכיטקטורת YOLO כללה שכבה fully connected שהוסירה בגרסאות הבאות. בנוסף, פונקציית LOSS וסדר התוכנות של המסגרות במצוא הרשת השתנו, אבל הרעיון נותר זהה.

YOLOv2

gresה זו משתמשת ברשת Backbone הנקראת Darknet19 ובها 19 קונבולוציות. המודל כולל כולל 22 קונבולוציות (מלבד 5 שכבות ה-LPOOL המקבילות על מנת למצוא את הפיצרים), ועוד מסלול עוקף בסופו לסתור לרשת המחזק את יכולת העיבוד. הכותב של המאמר המקורי, דודמו, נהג לפתח גם גרסאות "Tiny" לכל מודל. הוויאנט

נמצא יוטר אך הוא מהיר מאוד (207 תמונות לשנייה לעומת 67 של מודול YOLOv2, על מעבד TitanX.). מעניין לציין ש-YOLOv2 הוא המודול הראשון שאומן על תמונות במדדים משתנים, תהיליך המשפר את דיק המודול.



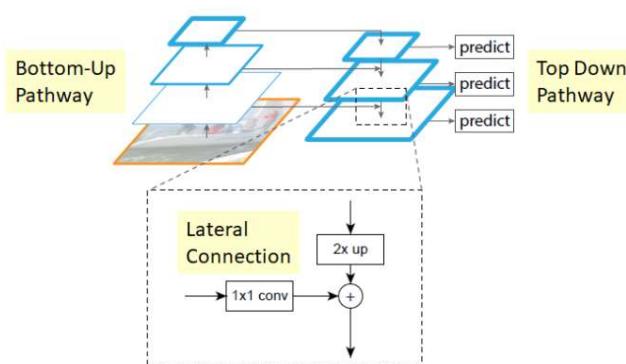
איור 9.4 ארכיטקטורת YOLOv2.

YOLOv3

כל דור נוסף של YOLO הציג חידושים ארכיטקטוניים שהגדילו את מורכבות החישוב וגודל המודל ושיפרו את ביצועיו. גרסה מספר 3 מבוססת על רשת Backbone גדולה בהרבה שנקראת Darknet53 ובה, כפי שעהלה משמה, 53 קונבולוציות. כמו כן הרשת מכילה צוואר של ארכיטקטורת Feature Pyramid Network (FPN) בעלת שלושה ראש גילי, כאשר כל אחד מהם הוא בעל רוחולזיה שונה (38×38 , 19×19 , 76×76).

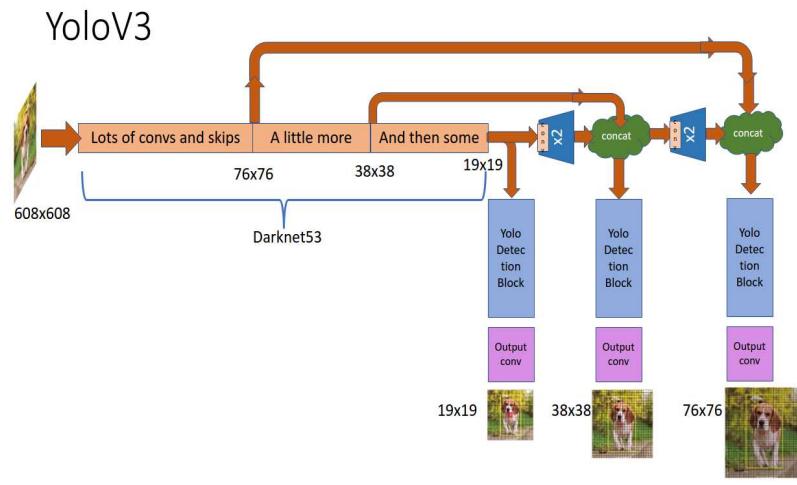
ארQUITקטורת FPN היא תוספת בקצה Backbone, אשר מגדילה חזקה באופן הדרמטי את מפת הפיצרים. תוספת זו נועדה ליצור מסלול Down מבקבי למסלול Up ברשת Backbone-FF. שבועה למשה בצורת Up, Bottom, בו ככל שמתקדמיים בשכבות Backbone-U מבצע עיבוד ברמה גבוהה יותר והרוחולזיה המרחבית של מפת הפיצרים הולכת וקטנה. בעזרתה השימוש ברשת FPN המודול יוכל לנצל את המיטב משני עולמות: הוא משתמש במידע שטחוני במפת הפיצרים הגדולה, שהוא אמן פחות מעובדת אך בעלת פיצרים ברוחולזיה מרחבית גבוהה, ובנוסף הוא מנצל גם את המידע ממפת הפיצרים הקטנה, שהוא אמן בעלת פיצרים ברוחולזיה מרחבית נמוכה, אך עם זאת היא מעובדת יותר.

לאחר כל הגדלה של מפת הפיצרים מtbody'ר בין התוצאה לבין מפת פיצרים קדומה יותר במדדים זהים (מתוך Backbone-FF). זאת בדומה לחיבורם העקיפים ברשת ResNet המסייעים להתקנות האימון. השכבות השונות של FPN מאפשרות לגלאי למצוא מיקום מדויק יותר של האובייקט ברוחולזיות השונות, מה שמעניק לרשת זו את יכולת להבחן באובייקטים קטנים בתמונה גדולה.



איור 9.5 ארכיטקטורת FPN (FPN), המשלבת מסלול down top לאחר ה-*up* bottom.

לרأس המודול של YOLOv3 יש מספר ענפי detection, אשר כל אחד מהם פועל על מפת פיצרים ברוחולזיה שונה ובאופן טבעי מתמחה בגלי אובייקטים בגודל שונה (הענף בעל הרוחולזיה הגבוהה מתמחה בגלי אובייקטים קטנים).



איור 9.6 ארכיטקטורת YoloV3.

YOLOv4

רשת YOLOv4 היא בעלת ראש זהה של שתי הגרסאות הקודמות, אך ה-Backbone שונה ומורכב יותר. הוא נקרא CSPDarknet53 כאשר Cross-Stage Partial Network מופיע כקיזור של CSPNet. רשת זו מפצלת מפות פיצרים לטובות קונבולוציה בחלקים ואיחוד מחדש. פיצול זה אפשרי, כמו במאמר המקורי, חלחול טוב יותר של הגרדיאנטים בשלב האימון. בנוסף, נעשה ברשת זו שימוש בפונקציית אקטיבציה הנקראת Mish (ולא ReLU) כמו בגרסאות הקודמות).

נקודטה מעניינת – החל מגרסת זו התוצאות אינן של היוצר המקורי ג'וזף רדמן, שהחליט לפרש מחקר ראייה ממוחשבת בגליל שיקולים אתים של שימושים צבאים או שימושים הפוגעים בפרטיות.

YOLOv5

רשת YOLOv5 מוסיפה עוד שכליים על רשת-h-Backbone על פני זו של הדור הקודם, ומציגת אופרטור חדש המארגן פיקסלים סטטיסטיים בתמונה בממד הפיצרים. אופרטור זה דואג לכך שהכניסה לרשת היא לא עמוק 3 פיצרים מכוקבל (RGB), אלא 12 פיצרים, תוך הקטנת הממד המרחב. באופן זה הרשת מתאמת לעיבוד תמונות ברזולוציה גבוהה, ואף לזרחות אובייקטים גדולים בклות רבה יותר, שכן שדה התמך (receptive field) של הקונבולוציות מכל מידע משתח תמונה גדול יותר.

9.1.4 Spatial Pyramid Pooling (SPP-net)

Spatial Pyramid Pooling (SPP) הינה שכבת pooling ברשת נוירונים, שמטרתה להסיר את האילוץ של רשות קונבולוציה, הדורש שכבת הכניסה לרשת תכיל תמונה בגודל קבוע (כמו למשל רשת VGG, המכבלת רק תמונות בגודל 224×224).

קיים קיימים מכשרי צילום רבים – החל מצלמות ניידות, מצלמות אבטחה ואף מצלמות טלפונים סלולריים ורחפנים. מצלמות שונות עשוות להוציא כפלט תמונות בגודלים שונים, מגוון סיבוט (למשל איקות התמונה או מטרת המכשיר). אם נרצה לבצע סיווג באמצעות רשת נוירונים לכל אותן תמונות, נאלץ לבצע שלב נוסף בתחלת הדרך – התאמת התמונה בכניסה לרשת.

נשים לב כי על מנת להתאים תמונה כלשהי לגודל מסוים, לרוב יבוצע חיתוך (crop), או שינוי גודל (resize/unwrap). פעולות אלה עלולות לפגוע ביזיהו עקב שינוי היחס בתמונה (מתיחה/כיווץ), החסירה של פרטים או שילוב של השניים. מוטיבציה זו היא שהובילה לשימוש בטכניקת SPP.

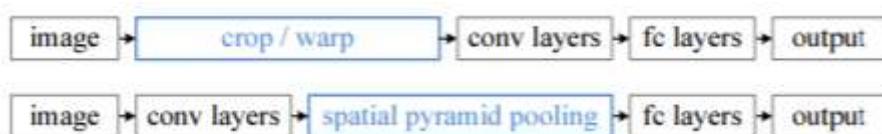
ניתן לראות דוגמא לשינוי גודל באמצעות מתיחה ולחיתוך באירור הבא:



איור 9.7: מימין - שינוי פרופורציה. משמאל – חיתוך.

למעשה, הדרישה לתרומות קלט בגודל קבוע בכניסה לרשותת אלה אינה הכרחית, שכן שכבות הקונבולוציה יכולות לחלץ מאפייני קלט (feature maps) בכל גודל. לעומת זאת, שכבות ה-FC בעומק הרשות הן השכבות שדורשות בכניסה להן קלט בגודל קבוע.

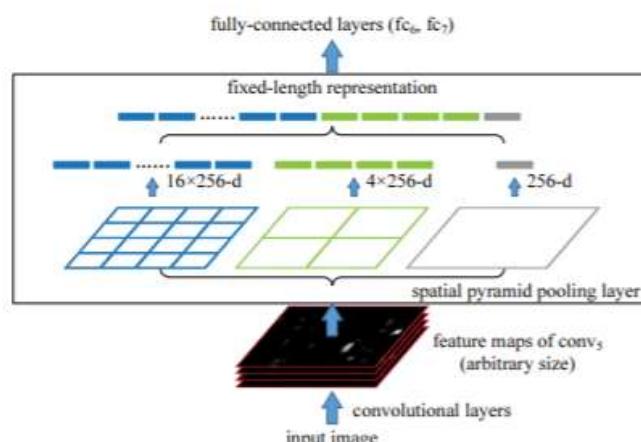
כעת, לאחר שהובירה המוטיבציה לייצרת רשת זו, ניתן להבין את אופן פועלתה. על בסיס שכבת ה-SPP מתבססת רשת רשות CNN. החידוש במבנה הרשות, הוא שבמקום שכבות בכניסה לרשות (לפניהם שכבות הקונבולוציה) תבצע התאמת תמונות ה קלט לגודל הנדרש ברשות. ההתקדמות המבוצעת בשכבת SPP תתבצע לאחר מציאת המאפיינים בשכבות הקונבולוציה, אך לפני שכבת ה-FC, כמוואר באיר הבא:



איור 9.8 – מבנה של רשת CNN קלאסית (למעלה) לעומת מבנה של רשת SPP-Net (למטה).

הרעין מאחורי שכבת SPP הוא חלוקה של הפלט של שכבות הקונבולוציה, ביצוע max-pooling בכל חלק ורשור של התוצאות לווקטור שגודלו אחד. במילים אחרות, עבור כל ה-features המתקבלים לאחר שכבות הקונבולוציה, מייצרים שלושה וקטורים לכל feature:

- וקטור בגודל 1 המתקבל באמצעות ביצוע max-pooling על כל הערכים באותו ה-feature.
 - חלוקה של ה-feature ל-4 תת-חלקים (2×2) וביצוע max-pooling בכל חלק מתוכם. מתקבל וקטור בגודל 4.
 - חלוקה של ה-features ל-16 תת-חלקים (4×4) וביצוע max-pooling בכל חלק מתוכם. מתקבל וקטור בגודל 16.
 - שרשור כל הווקטורים יחד לוקטור בעל 21 ערכים.
 - בסיום התהליך, מקבל שכבה ביצוג אחד בגודל 21, בהכפלה במספר ה-features שהתקבלו משכבות הקונבולוציה. שכבה זו "מחליפה" את שכבת ה-pooling הממוקמת לאחר שכבת הקונבולוציה האחרונה ותוצרה הוא ה קלט לשכבת ה-FC.
- ניתן לראות את מבנה הרשות באיר הבא, בו התקבלו features 256:



איור 9.9: ארכיטקטורת SPP-Net.

9.2 Segmentation

אחד האתגרים הכי משמעותיים בעולם הריאיה הממוחשבת הוא זיהוי אובייקטים בתמונה והבנת המתרחש בה. אחת הטכניקות הקלסיות להתחממות עם משמעותה זו הינה ביצוע סגמנטציה, כלומר, התאמת label לכל פיקסל בתמונה. בהתאם הסגמנטציה מבצעים חילוק/בידול בין עצמים שונים בתמונה המצלמת באמצעות סיווג ברמה הפיקסל, כלומר כל פיקסל בתמונה יסוג וישיר למחלקה מסוימת.

ישנם שימושים מגוונים באלגוריתמים של סגמנטציה – הפרדה של עצמים מסוימים מהרקע שמאחוריהם, מציאת קשרים בין עצמים ועוד. לדוגמה, תוכנות של שיחות וUID, zoom, skype, teams וכו', מאפשרות בחירת רקעים שונים עבור המשטח, כאשר מלבד הרקע הנבחר רק הגוף של המשטח מוצג בוידאו. הפרדת גוף האדם מהרקע והטמעת רקע אחר מtbodyות באמצעות אלגוריתמים של סגמנטציה. דוגמא נוספת – ניתן לזהות בתמונה אדם, כלב, וביניהם רצעה, ומכך ניתן להסיק שתוך התמונה הוא אדם מחזיק כלב בערתת רצעה. במקרה זה, הסגמנטציה מועדה למצאו קשר בין עצמים ולהבין את המתרחש.

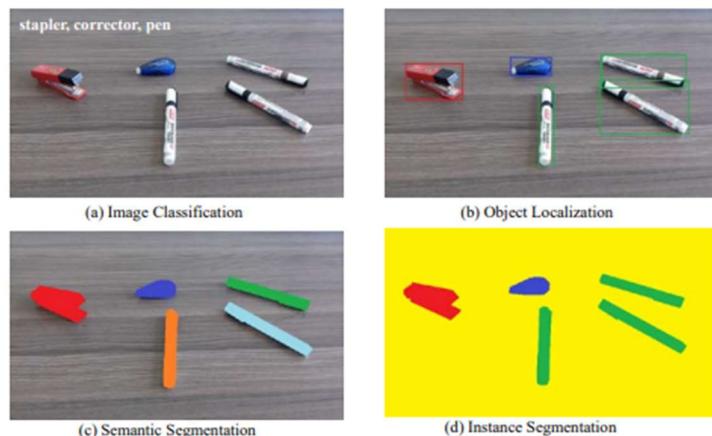
9.2.1 Semantic Segmentation vs. Instance Segmentation

קיימים שני סוגי עיקריים של סגמנטציה:

Semantic segmentation (חלוקת סמנטית) – חלוקה של כל פיקסל בתמונה למחלקה אליה העצם אותו הוא מייצג שיר. למשל, פיקסל יכול להיות משיר לכלי רכב, בן אדם, מבנה וכו'.

Instance segmentation (חלוקת מופע) – חלוקה של פיקסל בתמונה למופע של אותה מחלקה אליה העצם אותו הוא מייצג שיר. במקרה זה, בתמונה בה מופיעים מספר כלי רכב, תבצע חלוקה של כל פיקסל לאיזה כלי רכב אותו פיקסל מייצג – מכונית 1, מכונית 2, אופנוע 1, משאית 1 וכו'.

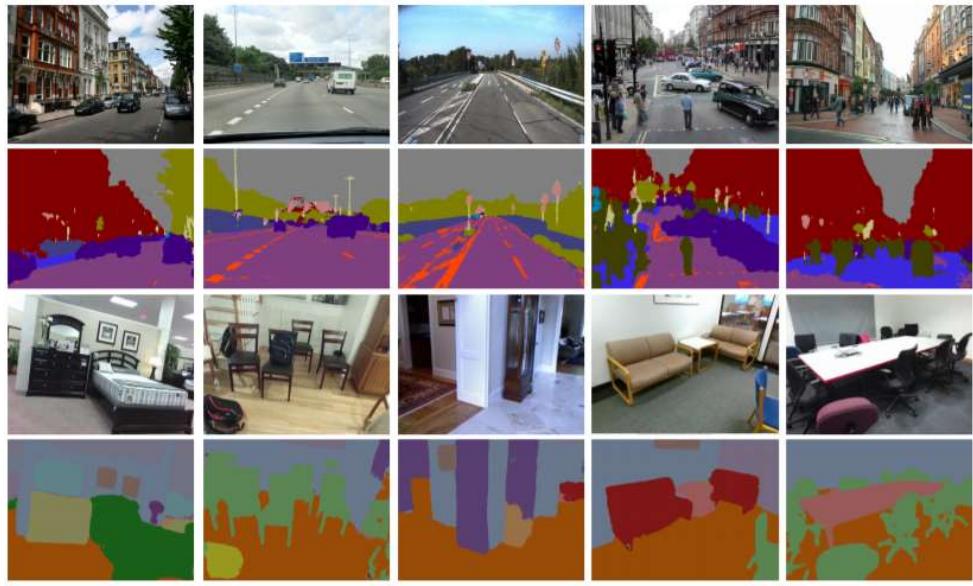
הבדל העיקרי בין שני סוגי אלו הוא ברמת עומק המידע של פיקסל – המידע עשוי לסווג את הפיקסל למחלקה כלשהי, או לעצם ספציפי בתמונה. עומק המידע משליך גם על עלות המידע. החלוקה הסמנטית מבצעת ישירות, בעוד שהחלוקת המופע דרושת בנוסף ביצוע של זיהוי אובייקטים כדי לסווג מופעים שונים של המחלקות.



איור 9.7 שימושות שונות תחת התחום של Computer Vision. ניתן להבחין בהבדל שבין Semantic segmentation (התאמת כל פיקסל למחלקה מסוימת) לבין Instance segmentation (התאמת כל פיקסל למופע של מחלקה מסוימת).

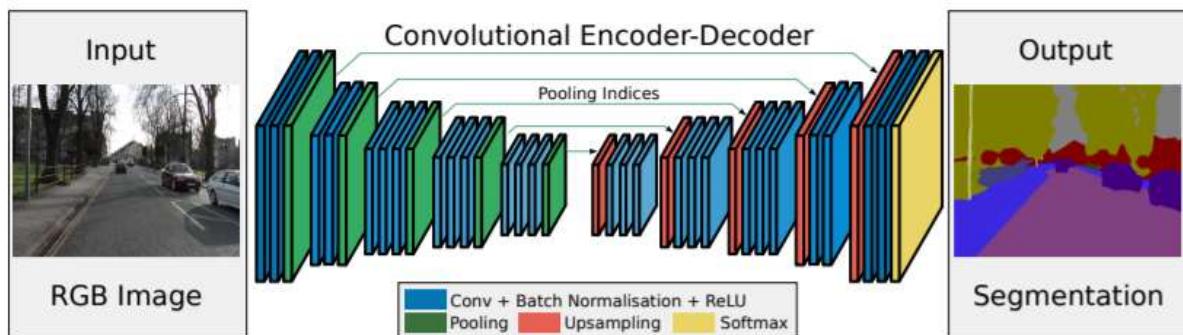
9.2.2 SegNet neural network

רשת SegNet הינה רשת קוונבולוציה عمוקה, שמטרתה לבצע חלוקה סמנטית (Semantic segmentation) לתמונה הקלט. בתחילת הרשת פותחה להבנה של תמונות חז' (למשל כביש עם מכוניות ובצדדים בתים והולכי רגל) ותמונות פנים (למשל חדר עם מיטה וכיסאות). הרשת נבנתה מtower מטריה להיות ייעילה בהיבטי זיכרון וזמן חישוב, תוך שמירה על דיוק מעשי.



איור 9.7 סיווג סמנטי בעזרת רשת SegNet עבור תמונות פנים וחוץ.

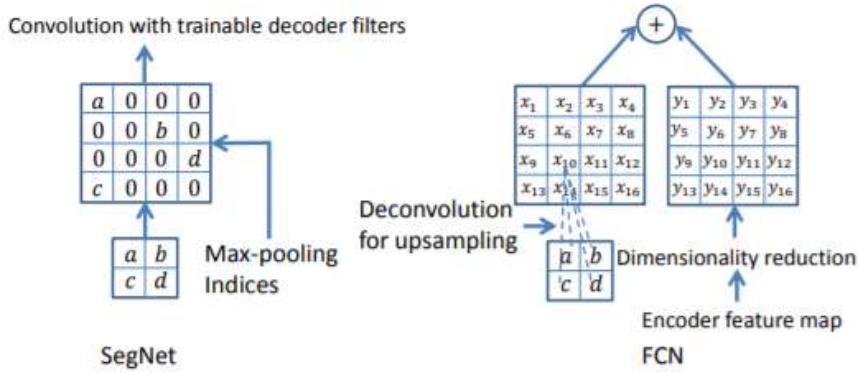
רשת בנויה כארQUITקטורת מקודד-מפענה (encoder-decoder). המקודד מורכב מ-13 השכבות הראשונות של רשת VGG16, ומטרתו לחלץ את מפת המאפיינים (feature maps). לכל שכבה קידוד יש שכבת פיענוח תואמת ומכאן שם רשת המפענה מכילה 13 שכבות. מטרת המפענה היא לבצע sampling-קָפָה וליצור תמונות מאפיינים בגודל המקורי. את מוצא המפענה 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 (activation function). לאחר שכבות אלו, ישנה שכבת max-pooling (subsampling), בעלת גודל חלון 2×2 ומרוחך (stride) בגודל 2.

החידוש ברשת זו הוא אופן הפעולה של המפענה. בשונה מרשתות אחרות, בה תהליך sampling-קָפָה גורר ביצוע חישובים עבור הפענוח, כמוואר באIOR הבא, הרעיון ברשת זו הוא שמירת מיקומי ערכי המקסימום הנבחרים מכל רבייה. רק הערכים שנבחרו כמקסימום יושחזו 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. למרות זאת, ביצוע הרשת מתאים לשימושים פרקטיים והפגעה בדיק קטנה מאוד.

בדומה למקודם, לאחר כל שכבה sampling-skip בפענוח, יופיעו שכבות קונבולוציה, שכבות נרמול ושכבות אקטיבציה מסוג ReLU. את מוצא המפענה מעבירים דרך שכבת SoftMax המבצעת סיווג ברמת הפיקסל. מוצא הרשת, שהוא גם מוצא שכבת SoftMax, הינו מטריצת הסתברויות, כאשר עבור כל פיקסל יש וקטור באורך K, כאשר K הוא מספר המחלקות לסיווג. כנובן 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 להזדהות 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 המחר'hינה' הינה:

$$\alpha_c = \frac{\text{median freq}}{\text{freq}(c)}$$

משמעותה של כל מחלוקת היא שיחסים בפונקציית המחיר כך שהתרומה של כל מחלוקת למחיר יהיה שווה. לכן, הוא מעניק למחלקות הגדלות יותר משקל נמוך יותר ולמחלקות הקטנות משקל גבוה יותר.

9.2.3 Atrous Convolutions (Dilated Convolutions)

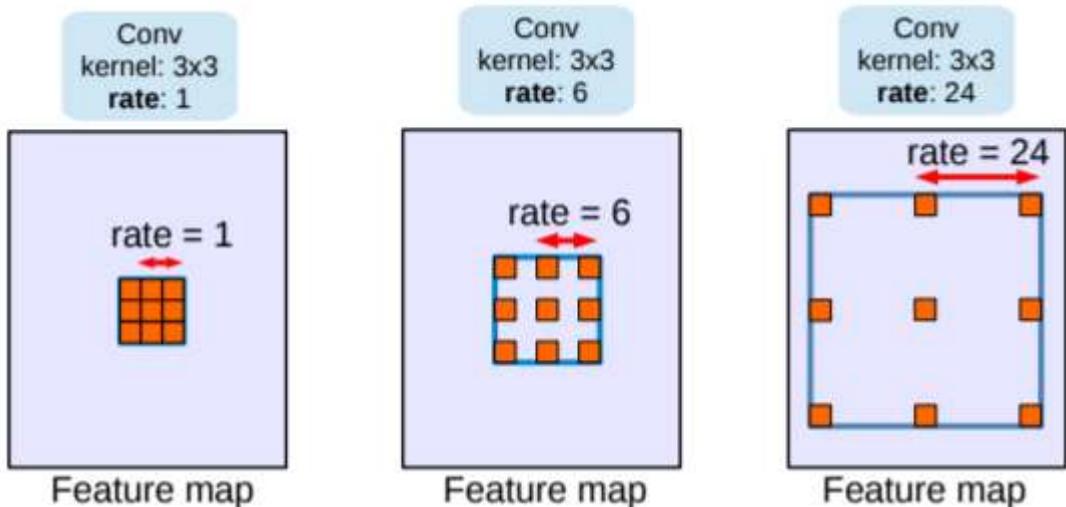
המושג Atrous מזכיר בשפה הצרפתית – "à trous", שמשמעותו "עם חורים". לכן, ניתן לתרגם את המונח Atrous convolution – "קונבולוציה מחוררת", ובמשמעותו מעט יותר מתאימה – קונבולוציה מרוחקת (או בשם אחר – Dilated convolution – – קונבולוציה מרווחת).

בטכניקת קונבולוציה זו, יש שימוש בפרמטר נוסף בנוסף למספרה הקונבולוציה – dilation rate. פרמטר זה מסמן את המרווח בין כל איבר בגרעין הקונבולוציה (הרחבת על פרמטר זה – בפרק 5.1.2). נוסחת הקונבולוציה עבור המקרה החד ממד' יחד עם פרמטר ההתרחבות r ניתנת לתיאור באופן הבא:

$$y[i] = \sum_{k=1}^K x[i + r \cdot k]w[k]$$

עבור $1 = r$ מתקבלת הקונבולוציה הרגילה, כאשר ישנו dilation של $1 > r$, מתקבלת קונבולוציה מרוחקת בפקטור r . יתרונה של קונבולוציה נעוץ בכך שuber אותו קרנל וüberו אותה כמות חישובים, מרחיבים את ה- (FoV) field of view של הקונבולוציה.

ניתן לראות את הרחבת field of view באירור הבא:



איור 10.9: קונבולוציה מרוחקת דו-ממדית, עם קרNEL בגודל 3×3 ופרמטר התחרחות $1, 6, 24 = r$. בהתאם לכל פרמטר מתקבל - field-of-view בגודל שונה – $3, 16, 49 \times 49 \times 3$ בהתאם.

9.3 Face Recognition and Pose Estimation

9.3.1 Face Recognition

אחד מהיישומים החשובים בראיה ממוחשבת הינו זיהוי פנים, כאשר ניתן לחלק משימה זו לשולש שלבים:

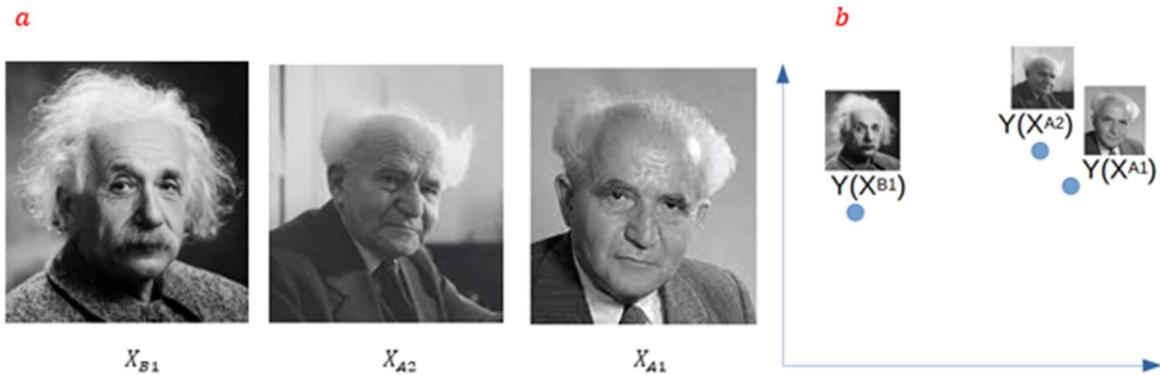
1. Detection – מציאת הרצופים בתמונה.
2. Embedding – מיפוי כל הרצוף למרחב חדש, בו המאפיינים שאינם קשורים לתיאור הפנים (למשל: זווית, מיקום, תארה וכדו) אינם משפיעים על הייצוג.
3. Searching – חיפוש במאגר של תמונות למיציאת תמונה פנים הקרובה לתמונה הפנים שהוצאה מהתמונה המקורית.

גישה פשוטנית, כמו למשל בניית מסגר המכיל מספר יציאות כמספר הפנים אותם רצים לזהות, הינה בעייתית מושתת סיבות עיקריות: ראשית יש צורך באלי דוגמאות לכל אדם (שלא ניתן בהכרח להציג). כמו כן, נוצרר למד את המערכת מחדש בכל פעם שהוא מישהו חדש. כדי להתגבר על בעיות אלו מוצעים "מידת מטריקה" (metric learning) בה מזקקים מאפיינים של פנים ויוצרים וקטור יחסית קצר, למשל באורך 128, המכיל את האלמנטים המרכזיים בתמונה הפנים. כתע נפרט את שלושת השלבים:

1. מציאת פנים.
 2. תיאור פנים.
 3. מציאת פנים.
- כדי למצוא הרצופים בתמונה ניתן להשתמש ברשות המבצעות detection, כפי שתואר בפרק 9.1. שיטה מקובלת למשימה זו הינה SIFT, המבוססת על חלוקת התמונה למשבצות, אשר עברו כל משבצת בוחנים האם יש בה אובייקט מסוים, מהו אותו אובייקט, ומה ה- bounding box שלו.

כאמור, המשימה בתיאור פנים נעשית בעזרת metric learning, כאשר הרעיון הוא לאקף פנים לוקטור שאינו מושפע ממאפיינים שלא שייכים באופן מהותי לפנים הספציפיות האלה, כגון זווית צילום, רמת תאורה ועוד. בכך לעשות זאת יש לבנות רשף המקבלת פנים של בנאים ומחזירה וקטור, כאשר הרדיפה היא שעבור שתי תמונות של אותו אדם יתקבלו וקטורים מאוד דומים, ועבור הרצופים של אנשים שונים יתקבלו וקטורים שונים. למעשה, פונקציית ה-loss תקבל בכל פעם minimization, ותעניש בהתאם לקרבה בין וקטורים של אנשים שונים וריחוק בין וקטורים של אותו אדם.

cutת נניח שיש לנו קלט X , המכיל אוסף פרצופים. כל איש יסומן באות אחרת – A,B,C. ותמונהות שונות של אותו אדם יסומנו על ידיאות ומספר, כך למשל X_{A1} זהה התמונה הראשונה של אדם A בסט הקלט X , ומכובן X_{A1}, X_{A2}, \dots הן שתי תמונהות של אותו אדם. באופן גרפי, בוד-ממד ניתן לתאר זאת כך (בפועל הוקטוריהם המיצגים פנימם יהיו בממד גובה יותר):



איור 9.10 (a) דוגמאות מסט הפרצופים X . (b) איר נרצה שהדעתה ימופה לממד חדש Y .

כאמור, נרצה לבנות פונקציית loss שמעודדת קירבה בין X_{A1}, X_{A2} , וריחוק בין X_{A1}, X_{B1} . פונקציית loss מרכיבת משני איברים, המודדים מרחק אוקלידי בין וקטורים שונים:

$$L = \sum_X \|Y(X^{Ai}) - Y(X^{Aj})\| - \|Y(X^{Ai}) - Y(X^{Bj})\|$$

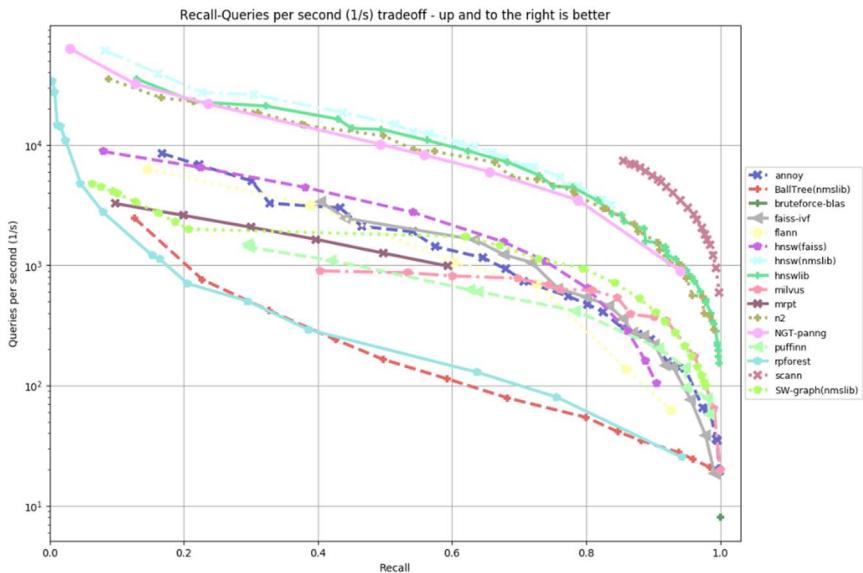
כאשר האיבר הראשון ינסה להביא למינימום וקטורים של אותו אדם, והאיבר השני ינסה להביא למינימום וקטורים של פרצופים שאינם שייכים לאותו אדם. כיוון שנרצה להימנע מקבלת ערכים שליליים, נוסיף פונקציית מקסימום. בנוסף, ניתן 'להרחיק' תוצאות של פרצופים שונים על ידי הוספה קבוע k , כך שהפרש בין המרחק של פרצופים של אנשים שונים לבין המרחק של פרצופים של אותו איש יהיה לפחות k :

$$L = \sum_X \max(\|Y(X^{Ai}) - Y(X^{Aj})\| - \|Y(X^{Ai}) - Y(X^{Bj})\| + k, 0)$$

loss זה נקרא triplet loss, כיוון שיש לו שלושה איברי קלט – שתי תמונהות של אותו אדם ואחת של מישחו אחר. כאמור, הפלט של הרשת הנלמדת צריך להיות וקטור המאפיין פנים של אדם, ומטרת הרשת היא למפות פרצופים שונים של אותו אדם לווקטורים דומים עד כמה שניתן, ואילו פרצופים של אנשים שונים יקבלו וקטורים רחוקים זה מזה.

3. מציאת האדם

בשלב הקודם, בו ה被执行 האימון, יצרנו למשה מאגר של פרצופים במרחב חדש. cutת כשיגיע פרצוף חדש, כל שנוטר זה למפות אותו למרחב החדש, ולחפש למרחב זה את הוקטור הקרוב ביותר לווקטור המיצג את הפנים החדש. בכך לעשות זאת ניתן להשתמש בשיטות קלאליסיות של machine learning, כמו למשל חיפוש שכן קרוב (כפי שהסביר בחלק 2.1.3). שיטות אלו יכולות להיות איטיות עבור מאגרים המכילים מילוני וקטורים, וישן שיטות חיפוש מהירות יותר (ובדרך כלל המהירות באה על חשבון הדיווק). בעזרת השיטה המובילה CarGAN (SCANN) ניתן להגיע לכמה מאות חיפושים שלמים בשניה (הchiposh ב-100 ממדים מתוך מאגר של 10000 דוגמות).



איור 9.11 השוואת ביצועים של שיטות חיפוש שונות. עבור פרט אחד נתון, מהפשים עבורי וקטורי תואם במדד החדש המכיל ייצוג וקטורי של הפרצופים הידועים. בכל שיטה יש טריד-אוף בין מהירות החיפוש לבין הדיק.

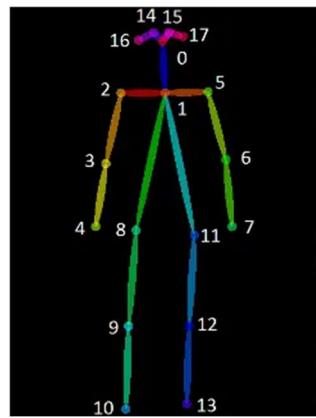
מלבד זיהוי וסיווג פנים, יש גם שיטות של מציאת אלמנטים של פנים הוכולות אף, עיניים וכו'. אחת השיטות המקובלות משתמשת בשערוך הצורה של פנים אנושיות, וניסיון למצוא את איברי הפנים לפי הצורה הסטנדרטית. בשיטה זו ראשית מבצעים יישור של הפנים והתאמאה לסקירה אנושית (על פי מרחק בין האיברים השונים בפנים), ולאחר מכן 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 למבנה 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 הוא עומד או יושב, מה התנוחה שלו, באיזה זווית האיברים נמצאים וכו'. ניתן להשתמש בניתוח התנוחה עבור מגוון תחומיים – ספורט, פיזיותרפיה, משחקים שונים ועוד. לרוב, תנוחה מיוצגת על ידי מיקומים של חלק גוף עיקריים כגון ראש, כתפיים, מרפקים וכו'. ישנו כמה סטנדרטים מקובלים, למשל COCO, posenet, COCO הנקודות מיוצגות בעזרת מערכת מערך של 17 נקודות (בדו-מיד):



איור 9.13 מיפוי תנוחה ל-17 נקודות מרכזיות.

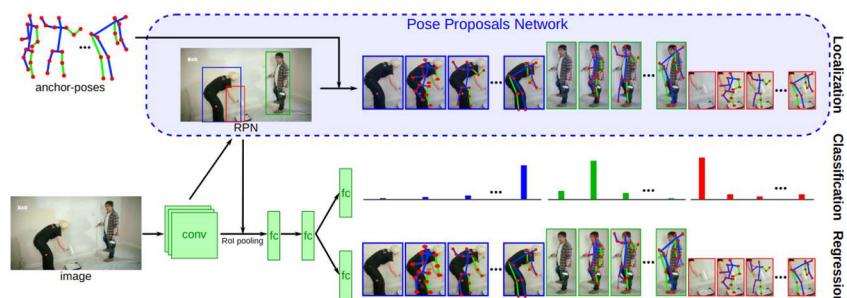
שאר הפורטטים דומים; מוסיפים עוד מידע (למשל מיקום הפה), משתמשים בתלת ממד במקום בדו ממד, משתמשים בסידור אחר וכך. כאשר רצים לאסוף נתונים על מנח גוף שונים, ניתן לשים חישנים על אותן נקודות מרכזיות וככה לקבל מידע על מנה הגוף לאורך זמן.

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 להזות באמצעות גרסיה (למשל, 34 מספרים שמתארים את המיקומים של 17 הנקודות בפורטט COCO בדו-ממד). במקרה זה ניתן לעשות למידה סטנדרטית לחולstein של בעיית גרסיה בעלי 34 יציאות.

ישנן מספר גישות כיצד להכילה את הרשות כך שתוכל לטפל גם במקרה הכללי בו יש יותר גוף אחד בתמונה. באופן נאיבי ניתן לבצע תהליכי מוקדים של מציאת כל האנשים בתמונה, ואז להפעיל על כל אחד מהם בנפרד את הרשות שמבצעת גרסיה, כפי שתואר לעיל. שיטה נוספת פועלת בכיוון הפוך – ראשית כל הרשות מוצאת את כל האיברים בתמונה נתונה, ולאחר מכן משיכת אותן לאנשים שונים. השיטה השנייה נקראת "מלמטה למעלה" (top-bottom), כלומר שקדם כל היא מוצאת את הפרטים ולאחר מכן מכלילה אותן. גישה זו עיליה ל蹶ה בו יש הרבה חפיפה בין האנשים בתמונה, כיון שאין לה צורך לבצע תהליכי מוקדים של הפרדת האנשים. השיטה הראשונה, הנקראת "מלמטה למטה" (bottom-top), תהיה פשוטה יותר עבור מקרים בהם אין חפיפה בין האנשים בתמונה וכל אחד מהם נמצא באזור שונה בתמונה, כיון שאין צורך לשירות איברים לאנשים.

רשות פופולרית לקביעת תנוחה נקראת *pose estimation*, המורכבת למעשה משתי תת-רשתות ופעולות בשיטת top-bottom. הרשות שאחראית על שייר חלק גוף לאדם מסוים, נקראת (PAF, part affinity fields) והרעיון שלו הוא לייצג כל איבר כשדה וקטורי. ביצוג זה 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, הפעולת בגישה down-top, נקראת LCR-NET, והוא מבוססת על רעיון של 'מיקום-סיווג-גרסיה' (Localization-Classification-Regression). בשלב הראשון יש תת-רשת המיצרת עוגנים עבור אנשים, אזורים בהם הרשות חושבת שנמצא באדם, ולאחר מכן הרשות משלבת את התוצאות שליהם. בשלב השני ככלומר, אזורים בהם הרשות חושבת שנמצא באדם, ככלומר כל עוגן מקבל ציון המיציג את טיב השערור של העוגן והתנוחה של האדם מתבצע clustering לכל העוגנים, ככלומר כל עוגן מקבל ציון המיציג את הטווח הסופי בעזרת מיצוע של הרובה עוגנים. הנמצא בתוכו. השלב השלישי מפשט את העוגנים ומשקל את טיב השערור הסופי בעזרת מיצוע של הרובה עוגנים. 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 נעה על כמות ואיכות הדата 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 הדата אינם מאוזן.

9.4.1 The Problem

התחום של למידה מORITY דוגמאות (Few-Shot Learning) נוצר על מנת להתמודד עם מצב של חוסר איזון קיצוני בין כמות הדוגמאות של כל קטgorיה לאימון הרשות. באופן פורמלי, קיימות קטגוריות הנקראות קטגוריות בסיס (base classes), עבורן יש כמות גדולה של דוגמאות, ובנוסף ישן קטגוריות 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 הרשות. בכלל, בקטגוריות הבסיס תהיינה 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 מORITY דוגמאות נובע מה הצורך להכין לרשות כמות ידועה נוספת הנרכבת הקיימ, תוך הימנענות מ-overfitting כתוצאה מכמות הפרמטרים הגדולה של הרשות לעומת כמות המועיטה של הדטה. לכן, גישה נאיבית כמו אימון מחדש של רשות על מעט דוגמאות נוספת יכולה ליצור הטיה בתוצאות.

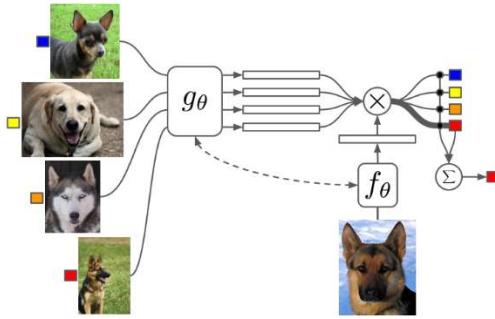
יש לציין כי בכל בעית הלמידה מORITY דוגמאות עבור האימון, אף שהשאיפה היא להצליח באופן זהה בזיהוי כל הקטגוריות בשלב המבחן, בו לא יהיה חוסר איזון. لكن בעיות אלו לוונטיות לשימושים רבים כמו: זיהוי חיות נדירות באופן זהה לחיות יותר נפוצות, מערכת זיהוי טילים שצריכה להתמודד גם עם אינויים נוספים יותר (ניתן לחשב למשל על פצת אוטום), מערכות זיהוי פנים שצריכות לעבוד טוב עבור כל אדם ללא תלות בדעתה שהיא קיימ באימון הרשות.

פרק זה נתאר את שלוש הגישות העיקריות לפתרון בעיות למידה מORITY דוגמאות. עבור כל גישה נציג את האלגוריתמים המשמעותיים ביותר שנקטו בגישה זו. לאחרונה, מפותחים יותר וייתר אלגוריתמי למידה מORITY דוגמאות שימושיים יחד ריעונות השאבים ממספר גישות יחיד אך נשענים על האלגוריתמים המשמעותיים מה עבר. 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 מORITY דוגמאות הנוקטות בגישת למידת מטריקה, שואפות לייצג את הדוגמאות כווקטורים של מאפיינים מרוחב רב-ממד', כך שניתן היה למצאו בקהלות את השיר הקטגוריה של דוגמא חדשה, גם אם היא תהיה מקטגוריה חדשה. שיטות אלו מבוססות על עיקנון הגדלת המרחק בין יצוגים וקטורים של דוגמאות מקטגוריות שונות (inter-class dissimilarity), בד בבד שמרה על מרחק קטן בין הייצוג הווקטורי של דוגמאות מאותה הקטגוריה (intra-class similarity).

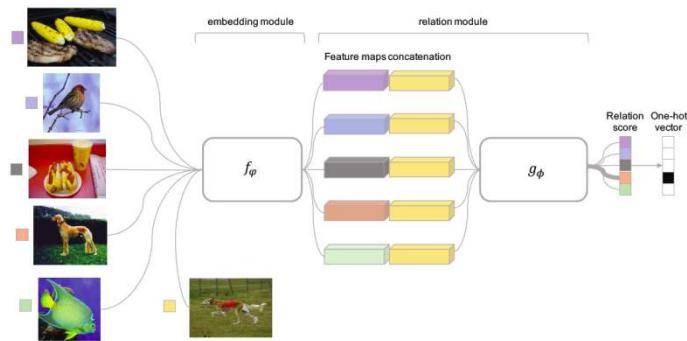
התקדמות משמעותית של שיטות אלו הוצאה במאמר Matching Networks for One Shot Learning בשנת 2016. שיטה זו משתמשת בזיכרון שהגישה אליו נעשית באמצעות מגנן Attention, על מנת לחשב את ההסתברות של דוגמא להיות שייכת לכל קטגוריה, בדומה לשיטות השכן הקרוב (Nearest Neighbors).



איור 9.15 אילוסטרציה של שיטת Matching Networks

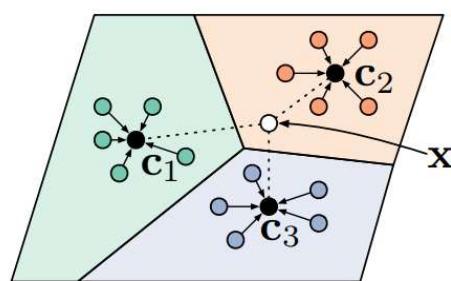
ההידוש המשמעותי בשיטת Matching Networks נועד בשיטת האימון המבוצעת באפיוזודות (Episodes). בשיטה זו האימון מכיל כמות של שימושים, כאשר כל שימוש היא למעשה מבחן של הדאטה שבו יש קטגוריות מסוימות שהן חדשות ואחרות מהן קטגוריות בסיס. על ידי דוגמאות רבות ויצירת שימושים מלאכותיים כאלה, בהן בכל פעם נלקחות קטגוריות אחרות ליצג את החדשניות, מתבצע אימון המתאים לבעה של מיעוט דוגמאות. שיטת אימון באפיוזודות הפכה לנפוצה ביותר בלמידה ממינית דוגמאות, גם בגין הatrixה ווגם בגישות שנראתה בהמשך.

שיטות רבות מتبוססות על הרעיון שמדובר זה. למשל שיטת Relation Network משרשרת וקטורי מאפיינים של דוגמת מבחן לבין כל דוגמא של קטגוריות האימון. אלו נכונים למודל המשערך ממד 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 המאפשרת נקודות אבטיפוס אופיינית המוחשבת על ידי המוצע של הדוגמאות בקטgorיה זו. בכך מחשבים מסווג לנארה המפריד בין הקטגוריות. בעת המבחן מסווג דוגמא חדשה על סמך מרחק אוקלידי מנקודות האבטיפוס.



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 מוגנות לאחיזה דיק נמכרים משמעותית מażוד הדיק המדוחים במקרים של איזון בין כמות הדוגמאות בקטגוריות השונות (לרוב מעל 90% דיק).

Method	5-way 1-Shot	5-way 5-Shot
Matching Networks	46.6%	60.0%
Prototypical Networks	49.42%	68.20%
Learning To Compare	50.44%	65.32%

איור 9.18 השוואת ביצועי דיוק של שיטות למידת מטריקה על קטגוריות חדשות עבור mini-ImageNet.

9.4.3 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 מצינים שמנקודת מבט של מערכות דינמיות, ניתן להתבונן על תהליך הלמידה שלהם כenza שמנקזם את ריגשות 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:

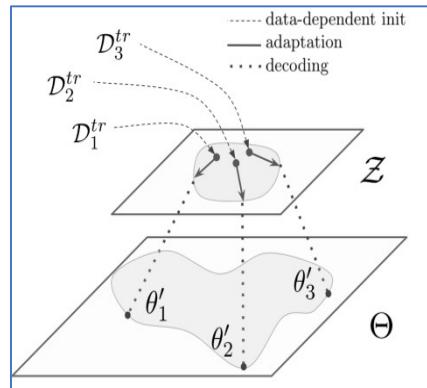
$$\theta'_i = \theta - \alpha \nabla_{\theta} L_{T_i}(f_{\theta})$$

עבור פונקציית מחיר L והפרמטר 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 לטנטי בעל ממדים נמוכים, ולאחר מכן עבר בחרזה למרחב המאפיינים הרוב ממד.



איור 9.19 שיטת (LEO) Latent Embedding Classification

השימוש במרחב מאפיינים בעל ממדים נמוכים המשמרים את המאפיינים החשובים לייצוג הקטגוריות, שיפור באופן ניכר את תוצאות הסיווג, כפי שניתן לראות בטבלה הבאה:

Method	5-way 1-Shot	5-way 5-Shot
MAML	48.7%	63.11%
LEO	61.76%	77.59%

איור 9.20 השוואת ביצוע דיק של שיטות מטא-למידה על קטגוריות חדשות über mini-ImageNet.

9.4.4 Data Augmentation

גישה שונה להטמודדות עם מיעוט דוגמאות נוקטת ביצירת דוגמאות כדי להימנע מהטיה. שיטות אוגמנטציה למשה יוצרות>Data New על סמך הדאטה המקורי. השיטות הפשוטות 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/סיקום אוטומטי של טקסט (Text summarization), מציאת תשובה עבור שאלה מסוימת (Question answering) ועוד. ועוד משימות רבות אחרות. פיתוח כלים המסייעים להtauוו 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 שוכלו מקרים הן Siri, העוזרת הקולית של אפל, הרשומה האוטומטית בתיבת החיפוש ב-Google ו-Grammarly, מערכת לתיקון תחבירי לטקסט באנגלית.

10.1 Language Models and Word Representation

בפרקים הקרובים נדונ 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 שייה איפשר 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 מאפשר לבנות בעזרת אימון על גבי דטה טקסטואלי, והוא נקרא "מודל שפה" (Language Model). לפני ביצוע האימון, יש לבצע מיפוי של הטקסט לייזוג מסויים (Word Representation), המאפשר למcona לנקות 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 שתنبي 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 אוף 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 המשווה הזה כך: ההסתברות לקבל את המשפט הנתון שווה להסתברות של המילה להופיע כפול ההסתברות של המילה a להופיע אחרי המילה big להופיע אחרי המילה Take , כפול ההסתברות של המילה corpus להופיע אחרי הציגוף Take a big .

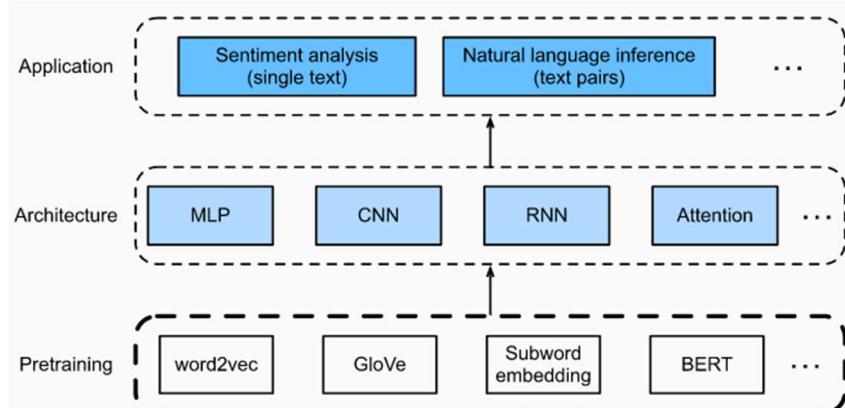
באופן הפטני ביותר נוכל לשערר את ההסתברויות האלו באופן הבא: ניקח corpus מספק גדול ועшир בעל N מילים שונות, כמו למשל כל ערבי וקיידי, ופושט נספר את כל המילים והציגופים שמופיעים בו. ההסתברות של כל מילה תהיה אחוֹד הפעמים שהיא מופיעה בטקסט, וההסתברות המותנית תיחס באופן דומה – על ידי מנת מספור המופיעים של צירוף 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 לנקות את מודל השפה זה או שניצר 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 שהוצגה לעיל, כל מילה (Token) מיצגת באמצעות צירוף אותיות. כך למשל המילה הראשונה במשפט

בapon סכמטי, ניתן לתאר את משימת עיבוד השפה מקופה לקרה באופן הבא:



11.1 תהליך פיתוח אלגוריתם של מודל שפה: א. מיצגים את הטקסט בצורה כלשהיא (ניתן כMOVIE לחתוך ייצוג קיימן שבנה על בסיס דאטאטס אחר). ב. מאמנים מודל שמקבל כ-*input* את הטקסט אותו יציגו בדרך כלל כלשהיא בשלב הראשון, והמודל מוציא *output* מסוים. למודל זהה יכולות להיות ארכיטקטורות שונות. ג. באמצעות המודל המאומן ניתן לבצע משימות קיצה שונות.

במהלך פרק זה נתמקד בשכבה התתaconה של התרשיים: נתאר מספר שיטות מרכזיות לייצוג טקסטים, ונראה כיצד ניתן לאמן מודלי שפה שונים היכולים לבצע כל מיני משימות.

10.1.1 Basic Language Models

מודל השפה הראשון אותו נציג הינו **N-Grams**, שהוא מודל סטטיסטי המניח שהסתברות למילה הבאה תלויה אף ב- N המילים שקדמו לה בסדרה. הנחה זאת נקראת "הנחה מרקוב" (Markov assumption), ובאופן כללי יותר, מודלי מרקוב (או שרשות מרקוב במקורה הדיסקרטי) מסדר n הם מודלי הסתברות המניחים شيئا'ן לחזות הסתברות של אירוע עתידי T בהתבסס על האירועים שהתרחשו ב- n נקודות הזמן שקדמו למועד T מבל' להתחשב באירועי עבר רוחניים מדי.

מודלigram N-gram היפשר ביותר נקרא n-gram. מודול זה אנהנו חוזים את המילה הבאה לפי התדריות של המילה עצמה ב-corpus מוביל להתחשב במה שקדם לה. כמוכן שחייב צזה הינו בעיתוי, מכיוון שהמילה הבאה **חייבת להיות תלויה במילים שקדמו לה**, וכך גם הינו מילים כמו the שמופיעות באופן תדיר בטקסט שאין בהכרח משמעות על ההקשר. לכן, נסתכל על מודול קצר יותר מרכיב המקרה bigram, המתיחס למילה האחרונה הקודמת למילה הנחוצה. במודול bigram אנו נזכיר את המשמעות 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 שלפנייה הן חסרות השפעה על התפקידות של המילה הנחזית (וממילא גם על המשך המשפט). באופן כללי, מודל N-grams מניח את המשווואה על הראב:

$$P(w_n | w_{n-1}, w_{n-2}, \dots, w_1) = P(w_n | w_{n-1}, w_{n-2}, \dots, w_{n-N}), N \leq n$$

:bigram ניקח לדוגמא מספר משפטים וננתח אותם בשיטת *tom*

- I know you
 - I am happy
 - I do not know Jonathan

נניח ונרצה לבדוק את ההסתברות שהמילה Jonathan היא המילה הבאה אחרי הסדרה I. ראשית נגידר 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, ואופן 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/כליל 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 שמאפיעים מספר מועט של פעמים (או אפילו לא מופיעים בכלל). אם למשל יש צירוף שמאפיע רק פעם אחת ולעומתו יש צירוף שנערכות 1000 פעמים, אז הוספה הקבוע \mathcal{K} משמעותה שעכשו נמינה את הצירוף הראשון ככזה המופיע $(\mathcal{K} + 1)$ פעמים וביחס לצירוף השני נתיחס ככזה שהופיע $(\mathcal{K} + 1000)$ פעמים. הוספה זו כמעט ולאינה משפיעה על הצירוף השני, אך היא יכולה להכפיל פי כמה את ההסתברות של הצירוף הראשון. כמובן שכאשר אנחנו מתעסקים עם הסתברויות נרצה לאחר הוספת הקבוע במונה לתקן גם את מקדם הנורמל. באופן פורמלי, החלקת לפסל מוגדרת באופן הבא:

$$P_{Laplace}(w_n|w_{n-1}) = P_{Add-\mathcal{K}}(w_n, w_{n-1}) = \frac{C(w_{n-1}w_n) + \mathcal{K}}{\sum_w C(w_{n-1}w_n) + \mathcal{K}} = \frac{C(w_{n-1}w_n) + \mathcal{K}}{C(w_{n-1}) + \mathcal{K} \cdot |V|}$$

כאשר $|V|$ הוא גודל המילון (כמות המילים המופיעות ב-*n-grams*). היתרון בשיטה זאת היא הפשטות שלה, אך עם זאת יש לה חסרון בולט הנובע מכך שהיא משנה באופן ממשוני 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 למכנה. באופן זה כן ניתן לשולוט 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 משינוי הספרה.

Backoff and Interpolation

שיטת החלוקת בסעיף הקודם נותרה קבועה להסתברות של מאורעות המקבילים הסתברות 0, וישן גישות נוספת לשונת מענה למוגבלות האלה. נניח ואנחנו משתמשים במודל trigram, מודל המניח שההסתברות 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 הסתברות 0, ובuczם תקיים את המשוואה:

$$P(w_n | w_{n-1}, w_{n-2}) = 0$$

בכדי להימנע מלחת הסתברות 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 הסתברות 0 (כלומר, צמדי מילים שאין מופיעים בטקסט) ונשערר אותם באמצעות unigram. באופן זה מקווים להגיע לפחות במספר המילים בעלי הסתברות 0 היא קטנה (עד אפסית), כדי ששיעור השימוש במודלים יותר פשוטים עשוי שימוש במוגן רחוב יותר של קומבינציות הקיימות בטקסט. שיטה דומה נקראת Interpolation, ובזה במקומ לחת את הרצפים בעלי הסתברות 0 ולשערר אותם במודל הנמצא בדרגה אחת מתחת (למשל – לשערר בעזרת bigram במקומ trigram), המודל הראשון מתיחס לקומבינציה כלשהי של המילים – למשל קומבינציה ליניארית). בשיטה זו גם מקרים שאינם בעלי הסתברות 0 נעצרים ב-unigram, bigram and trigram ב-gram 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 סינקטטיים (=תחביריים) של השפה, קרי אין המילה נכתבת. עם זאת, יציג זה חסר מאד, כיון שהוא יתקשה בלמידת יציג של מאפיינים סמנטיים. דוגמא להבנה – סמנטית של שפה היא ההבנה שכלב וחתול הן לא מילים בלבד, אך הן כן קשורות אחת לשנייה באופן כלשהו – שתיהן מייצגות 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, המייצג כל מילה במלון 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 שהיא 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, v , שניהם בעלי אותו הממד N . המכפלה הפנימית (dot product) בין הווקטורים אלה מוגדרת באופן הבא:

$$v \cdot w = v^T w = \sum_{i=1}^N v_i w_i$$

באמצעות הגדרה זו ננסה לאמוד את הדמיון בין הייצוג של המילים כלב וחתול במרחב הוקטורי שנוצר בעקבות ייצוג בעזרת One-Hot vectors. כאמור, הייצוג של המילה חתול במרחב זה הוא: $[0, 1, 0] \rightarrow [0, 1, 0]$, בעוד שהייצוג של המילה כלב באותו מרחב הינה: $[1, 0, 0] \rightarrow [1, 0, 0]$. 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, נכללות בספירה למرات שאין לה בהכרח מושיפות מידע משמעותי לייצוג.
2. מילים המופיעות בטקסט לעתים 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, מגדים פונקציה ניקודacha על בעלת שני רכיבים. במקרה להסתכל על חלון יחיד מסביב לכל מילה, ניתן להסתכל על כל המילים בחולנות הנוצרים מסביב למילה המייצגת, ולתת לה ניקוד לפי התדרות של אותה מילה. באופן פורמלי, tf מוגדר בצורה הבאה:

$$tf = \log(C(w, c) + 1)$$

משמעות הביטוי היא שאנו סופרים כמה המילה c מופיעה בקונטקסט (=בחלון) של המילה המיוצגת w (על התוצאה מפעלים \log בשבייל rescaling של ערכים גבוהים מאוד).

עד כה השיטה אינה שונה במהותה מספירה 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 מופיעה בקונטקסטים אחרים, בעוד N מייצג את כמות המילים במילון. נשים לב שגם מילה מסוימת, למשל the, מופיעה בכל הקונטקסטים של כל המילים במילון, אך הביטוי בתוך הלוג יהיה 1 וכן df יהיה 0. לעומת זאת, אם מילה מסוימת מופיעה אך ורק בקונטקסט אחד, אז הערך שהוא בתוך הלוג יהיה N .

לבסוף, TF-IDF מוגדר באופן הבא:

$$TF - IDF = tf \cdot idf$$

מדד משוקל זה מצליח עבור ייצוג של מילה מסוימת לתת משקל גדול למילים אחרות הנמצאות אליה בקונטקסט באופן תקין אך אין נמצאות בקונטקסט של מילים אחרות.

PPMI - Positive Pointwise Mutual Information

כעת נרצה לפתור את הבעיה השנייה – בעית הייצוג הדليل למילים שאינן תדרות בטקסט. שיטת PPMI מגדייה 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 x, y ביחד שווה לכפל ההסתברויות לראות כל אחת לחוד, נקבל שערך הביטוי הוא $\log(1)$ כלומר 0. לעומת זאת, אם הסיכוי לראות את המילים 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 שמכבים על החישובים במשימות השפה השונות. למשל – ראיינו קודם שנדעתן ליצג מילה באמצעות וקטור שכלו אפסים למעט תא אחד עם הערך 1 במקומות ייחודיים לכל מילה. עבור שפה עם אלפי מילים ואף יותר מכך, כל וקטורי המיצג 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 ישרה ולא נדרש בתיאוג (שעלול להיות מודע לך). בהקשר זה, אלגוריתם Word2Vec משתמש ב-SGNS – Skip-Grams-With-Negative-Sampling באופן של Self-supervision בעקבות סיווג שמרתרה לחזות מה ההסתברות של מילים שונות להיות בקונטקסט של מילה נתונה. בסוף הוא להגדיר בעיית סיווג שמרתרה 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 ונתנו 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 שמאפיינן עם

(tales באותו חלק) יהיה קרוב לייצוג של tales בעוד שמלים שאינן מופיעות באותו קונטקסט יהיו בעלות ייצוג "שונה" (mbhinit פנימית או similarity). נניח שהחומר שהייצוג של כל מילה יהיה וקטור בגודל 100. נתחיל את התהיליך כך שכל מילה מקבלת וקטור רנדומלי. נסמן את וקטור היצוג של המילה w - e_w ואת הווקטור של מילת Kontekst c - e_c . השאייפה היא שהמכפלה הפנימית של וקטורי היצוג של המילים בكونטקסט של w יהיה גבוהה יחסית, בעוד שהמכפלה הפנימית של וקטורי היצוג 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).Cut נוכל להגיד בעיית logistic regression באופן הבא:

$$p(+|w, c) = \sigma(e_w, e_c) = \frac{1}{1 + e^{-e_w \cdot e_c}}$$

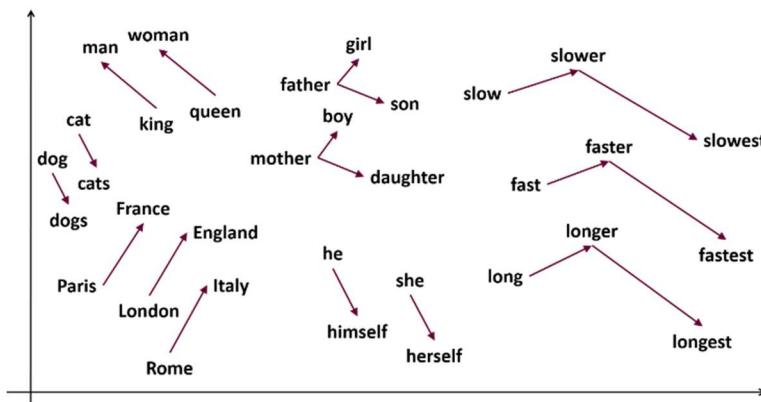
$$p(+|w, c) = 1 - p(+|w, c)$$

ובהתאם לכך פונקציית המטרה (Loss) תוגדר באופן הבא:

$$\begin{aligned} L &= -\log[p(+|w, c_{pos}) \prod_{i=1}^k p(-|w, c_{neg_i})] \\ &= -[\log(p(+|w, c_{pos})) + \log(\prod_{i=1}^k p(-|w, c_{neg_i}))] \\ &= -\left[\log(\sigma(e_w \cdot e_{c_{pos}})) + \sum_{i=1}^k \log(1 - \sigma(e_w \cdot e_{c_{neg_i}}))\right] \end{aligned}$$

נשים לב שאנו מניחים כי תלות בייצוג של הדוגמאות השליליות. בכך שנבצע מינימיזציה לפונקציית מטרה זו נוצרים למכפלה הפנימית בין היצוג של מילה לבין מילת הקונטקסט להיות גבוהה ובו בזמן למכפלה הפנימית בין וקטורים שאינם בكونטקסט להיות נמוכה. כך היצוג של המילה tales ("דומה") (= קרוב במונחים של similarity) לייצוג של המילים בكونטקסט שונה מיצוגן של המילים שאינם בكونטקסט. את תהליך המינימיזציה במשך האימון נוכל לבצע באמצעות stochastic gradient descent.

אחת התוצאות היפות והחשובות של שיטת Word2Vec ניתנת להמחשה על ידי פריסת וקטורי היצוג במרחב נמוך. בעזרת שיטות מתקדמות להורדת ממד (כפי שהוסבר בהרחבה בפרק 2), ניתן לצייר בדו-ממד או תלת ממד את וקטורי היצוג של המילים לאחר האימון.



איור 11.2 וקטורי היצוג 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 היצוג של המילים King, Man מקביל ובעל אורכו דומה לווקטור המחבר בין היצוג של Queen, Woman. דוגמה נוספת – הווקטור בין שם של ארצות הברית להיבירה של אותה ארצות הברית הוא קשור 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

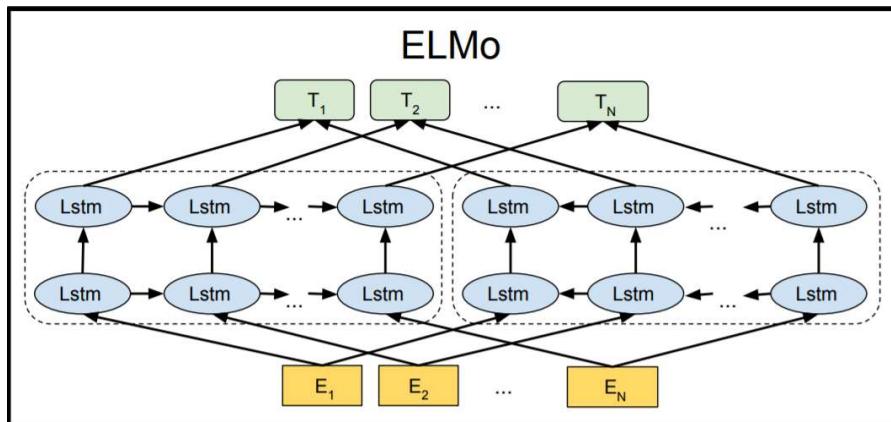
מנגנון בנייה ייצוגי מילים (embedding) שראינו עד כה למדו ייצוג סטטי עבור כל מילה. אך דבר זה יכול להיות בעיתוי מסוון לשפה טבעית היא דינמית ותלויה בקשר, ולאותה מילים יכולה להיות כמה פירושים. בשביב להבין את הבעיות נסתכל על המשפטים הבאים:

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 מנגנון embedding כמו word2vec יציג 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 שלה בתוך המשפט. הרעיון במודל זה הוא לקח 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- 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 המודל בעזרת מושימת מידול שפה וחוזים את המילה הבאה בהינתן המילים הקודמות. אך בשונה מאלגוריתמים אחרים, ELMo משתמש בארכיטקטורת דו-כיוונית, כך שבתהליך האימון משלבת משימת שפה נוספת המנסה לחזות את המילה הקודמת בהינתן הסוף של המשפט. הארכיטקטורת של ELMo בנויה ממספר שכבות של LSTM שמורכבות זו על גבי זו, ולפי כתבי המאמר השכבות 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 המודל ולהשתמש בו עבור מושימות אחרות. הכותבים מציעים לשרשר את הייצוג הווקטור 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 ציול (Fine tune) עבור מושימה ספציפית. כך לדוגמא נוכל להתאים את הייצוג של המילים למושימת סיווג של משפט לעומת מושימת תיאוג של ישויות במשפט.

פה חשוב להזכיר נקודת מרכזית – בסופו של דבר התוצר של ELMo הינו **מודל שפה הлокט ייצוג של טקסט והופך אותו ליצוג תלי הקשר**. שכבות ה-LSTM השונות מאמננות מודל שפה על מנת ליצור ייצוג חדש עבור המילים,

המתיחס גם להקשר. לאחר סיום האימון של מודל השפה (pre-training), ניתן לנקח אותו ולבצע transfer learning, כלומר להשתמש ביצוגים שהוא מפיק גם למשימות אחרות על ידי הוספת שכבות בקצה. לאחר פרסום המאמר, Sebastian Ruder (חוקר NLP מפורסם) טען כי:

"It is very likely that in a year's time NLP practitioners will download pretrained language models rather than pre-trained word embeddings"

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)

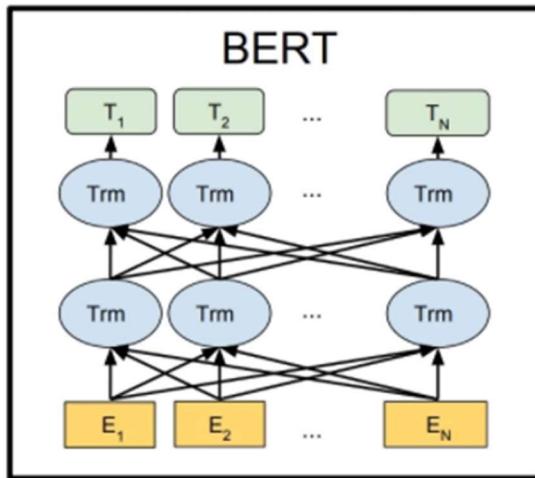
כאמור, הרעיון של ELMo הוא לייצר contextual embedding, ובכך לקבל מודל שפה המאפשר לנקח ייצוג וקטורי של מילים ולהעיר אותו במידע על הקשרן של כל מילה בטקסט. למרות ש-ELMo משתמש בשני היכיוונים של המשפט (חוזה את המשך המשפט מתחילה ואת תחילת המשפט מסוף) הוא אינו לומד משנה היכיוונים בתהילר אחד, אלא צריך לחלק את הלמידה לשני חלקיים שונים. בנוסף, כפי שהסביר בהקדמה לפרק 8, כאשר מתעניקים עם סדרות ארוכות של מילים, וקטור 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 הווקטורי של המילים ב-ELMo אינו מצליח 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 (–묘사 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 המתקברים בין המילים יכולם להיות מיזגים בצורה טובה גם עבר רצפים ארוכים ומילים שאינן נמצאות בסמיכות יחסית (שכאמור זה היה אחד החסרונות הגודלים של ELMo). בנוסף, מנגנון זה מיותר את הצורך לעבר 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 המוצג באופן מסוים ולהוסיף לו מידע על היחס בין המילים השונות שבtekst. תהליכי יצירת המודל היה אמם יקר, אך ניתן ללקחת אותו ויחסית בקלות לכילו אותו ואף 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 הבזוקדות 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 מיללים ואפיו או אוטו-טוקנים נקראו טוקנים או אסימונים. דבר זה יכול לעזור לנו בהכללה ולהקטיין את הסיסוי לטוקן שלא נמצא במילון (Out of Vocabulary).

הארQUITקטורה של GPT בונה מ-transformers מה שמאפשר לבנות ארכיטקטורה עמוקה שמתחשבת בקונטקט 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. התהליך זהה מתבצע באמצעות רכיב הנקרא Masked Self attention. בשונה ממנגן self-attention רגיל שמקודד כל טוקן בעזרת הקונטקט של כל שאר הטקסט, GPT צריך לקודד כל טוקן רק בעזרת הטוקנים שקדמו לו, כיוון שהשלב 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 מאפס 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 hei 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 מסוימת 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 בテקסט זה ובודק **מה ההסתברות שמודל השפה ינביא את המילה הבאה בטקסט בהינתן כל המילים שלפנייה**. ככל שהסתברות יותר גבוהה, כך המודל יותר מוצלח. באופן פורמלי, perplexity מוגדר באופן הבא:

$$PP(W) = P(w_1, w_2, \dots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}}$$

ככל שהמודל 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:

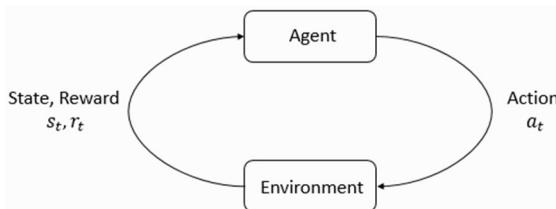
<https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html>

11. Reinforcement Learning (RL)

רוב האלגוריתמים של עולם הלמידה הינם מבוססי דата, כלומר, בהינתן מידע מסוים הם מנסים למצאו 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 = S, אליו אוסף של אובייקטים, הורדת ממד ועוד.}

למידה מבוססת חיזוקים הינה פרדיגמה נוספת תחת התחום של למידת מכונה, כאשר במקרה זה הלמידה לא מסתמכת על>Data{} {x = S}, אלא על חקירה של הסביבה ומיציאת המדיניות/הסטרטגיה הטובה ביותר לפועלה. שמו סוקן שנמצא בסביבה שאינה מוכרת, ועליו לבצע צעדים כך שהתגמול המתקבל אותו הוא יקבל יהיה מksamיל. בלמידה מבוססת חיזוקים, ביכולתו לפרט אחורות של למידת מכונה, הסביבה לא ידועה מבעוד מועד. הסוקן 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 מימי מצבים לא ידועים הסוקן יבחר בפעולות שבאupon מctrber' הינו הכיעילות עבורה. נתאר את תהליך הלמידה באופן גרפי:



איור 11.1 מודל של סוקן וסביבה.

בכל צעד הסוקן 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 שהסוקן יבצע פעולות המזקחות אותו בתגמול חיובי (-חיזוק) 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 חיזוקים עובדים רואית יש להגדיר את המושגים השונים, ובנוסף יש לנסח באופן פורמלי את התיאור המתמטי של חלקו הבעה השונים.

11.1 Introduction to RL

בפרק זה נגדיר באופן פורמלי תהליכי מרקוב, בעדרותם ניתן לתאר בעיות של למידה מבוססת חיזוקים, ונראה כיצד ניתן למצוא אופטימום לבעיות אלו בהינתן מודל וכל הפרמטרים שלו. לאחר מכן בקשרו במספר שיטות המנסות למצוא אסטרטגיה אופטימלית עבור תהליכי מרקוב כאשר לא כל הפרמטרים של המודל נתונים, ובפרק הבא נדבר על שיטות אלה בהרחבה. שיטות אלה הן למעשה להלך של למידה מבוססת חיזוקים, כיוון שהן מנוסות למצוא אסטרטגיה אופטימלית על בסיס תגמולים ללא ידיעת הפרמטרים של המודל המרковי עבורו רוצים למצוא אופטימום.

11.1.1 Markov Decision Process (MDP) and RL

המודל המתמטי העיקרי העיקרי עליו מבוסים האלגוריתמים השונים של RL הינו תהליכי מרקוב, כלומר תהליכי שבה המעברים בין המצבים מקיימים את תכונת מרקוב, לפיה ההסתגלות של מצב מסוים תלויה רק במצב הקודם לו:

$$P(s_{t+1} = j | s_t = i, s_1 = s_1, \dots, s_t) = P(s_{t+1} = j | s_t)$$

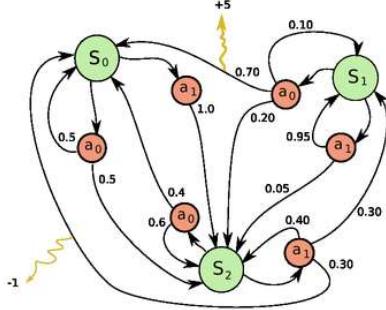
תהליכי קבלת החלטות מרקובי מתואר על ידי סט הפרמטרים $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}\}$:

- (\mathcal{S}) – מרחב המצבים של המערכת. המצב ההתחלתי מסומן ב- S_0 .
- (\mathcal{A}) – מרחב הפעולות. A_s הוא מרחב הפעולות האפשריות במצב S .
- (\mathcal{T}) – 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 הסוקן למצב s' .
- (\mathcal{R}) – הביטוי: $\mathbb{R} \rightarrow (s, a) \rightarrow \mathcal{R}_a$ הינו פונקציה הנונctaת תגמול/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 מركז. יונם 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.

סטרטגיה של סוכן, המסומנת ב- π , הינה בחירה של אוסף מהלכים. בעיות של למידה מבוססת חיזוקים, נרצה למצוא **סטרטגיה אופטימלית** (Optimal Policy) $\pi: S \rightarrow A$ הממקסמת את **התגמול המצטבר** $(\sum_{t=0}^{\infty} \mathbb{E}[R(s_t, \pi(s_t))])$. כיוון שלא תמיד אפשר לחשב באופן ישיר את האסטרטגיה האופטימלית, ניתן להגדיר ערך החזרה (Return) המבטא סכום של תגמולים, ומנסים למקסם את התוחלת שלו $\mathbb{E}[Return | \mathcal{S}, \mathcal{A}]$. ערך ההחזרה הינו נפוץ ונקרא $Return(1)$, והוא מוגדר באופן הבא: עבור פרמטר $\gamma \in (0, 1)$, ה- γ -return הינו הסכום הבא:

$$Return = \sum_{t=1}^T \gamma^t r_t$$

אם $\gamma = 0$, אז מתעניינים רק בתגמול המיידי, וככל ש- γ גדול כך יותר נתונים יותר משמעותם 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 נותנת לכל מצב ערך מסוים המשקף את תוחלת התגמול שנitin להישיג דרך מצב זה. באופן פורמלי, כאשר מתחילה במצב s , ה- γ -return מוגדר להיות:

$$\mathcal{V}^\pi(s) = \mathbb{E}[R(\tau) | s_0 = s]$$

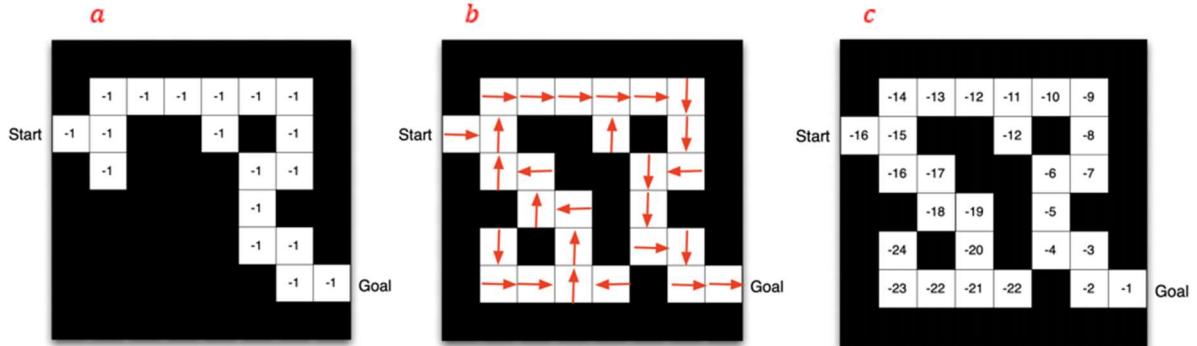
בעזרת ביטוי זה ניתן לחשב את **הסטרטגיה האופטימלית**, כאשר ניתן לנוקוט בגישה ישירה ובגישה עקיפה. הגישה הישירה מנסה למצאו בכל מצב מה פעולה הכי כדאי. בהתאם לכך, חישוב האסטרטגיה האופטימלית יעשה באופן הבא:

$$\pi(s) = \arg \max_a \sum_{s'} p_a(s, s') (\mathcal{R}_a(s, s') + \gamma \mathcal{V}(s'))$$

לעתים החישוב השיר מסובך, כיוון שהוא צריך ללקח בחשבון את כל הפעולות האפשרות, ולכן מסתכלים רק על ה- γ -return. לאחר שלכל מצב יש ערך מסוים, בכל מצב הסוכן יעבור למצב בעל הערך הכי גדול מבין כל הממצבים האפשריים אליו הם ניתן לעבור. חישוב הערך של כל מצב נעשה באופן הבא:

$$\mathcal{V}(s) = \sum_{s'} p_\pi(s, s') (\mathcal{R}_\pi(s, s') + \gamma V(s'))$$

ניתן לשים לב שבעוד הגישה הראשונה מתמקדת במציאת אסטרטגיה/מדיניות אופטימלית על בסיס הפעולות האפשריות בכל מצב, הגישה השנייה לא מסתכלת על הפעולות אלא על הערך של כל מצב, המשקף את תוחלת התגמול שnitן להשיג כאשר נמצאים במצב זה.



איור 11.3 (a) מודל: המצב של הסוקן הוא המשבצת בו הוא נמצא, הפעולות האפשריות הן ארבעת הכיוונים, כל פעולה גוררת תגמול של -1, והסתברויות המעבר קבועות לפני 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:
$$\mathcal{P}_{ss'}^a = p_\pi(s, s' | s_t = s, a_t = a)$$

$$\mathcal{R}_{ss'}^a = \mathbb{E}[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s']$$
- פונקציה המתארת את התוחלת של התגמולים העתידיים: Value function

$$V^\pi(s) = \mathbb{E}[R(\tau) | s_0 = s]$$

- מדיניות/אסטרטגיה (Policy) – בחירה (דטרמיניסטיבית או אקראית) של צעד בכל מצב נתון:

$$\pi(s|a)$$

11.1.2 Bellman Equation

לאחר שהגדכנו את המטרה של למידה מבוססת חיזוקים, ניתן לדבר על שיטות לחישוב אסטרטגיה אופטימלית. בפרק זה נתייחס למקרה הספציפי בו נתון מודל מركובי עם כל הפרמטרים שלו, כמו ראות המצבים, הפעולות והסתברויות המעבר ידועים. כאמור, Action-value function היא 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 זיהogaoptimal π^* . 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 אנלטי, 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 $O(n^3)$, ועבור \mathcal{A} גדול החישוב נהיה מאוד יקר ולא עיל. כדי לחשב את הפתרון באופן עיל, ניתן כאמור להשתמש בשיטות איטרטיביות. שיטות אלו מבוססות על אופרטור בלמן, המוגדר באופן הבא:

$$BO(V) = R^\pi + \gamma T^\pi \cdot V$$

ניתן להוכיח שאופרטור זה הינו העתקה מכווצת (contractive mapping), כלומר הוא מקיים את התנאי:

$$\forall x, y: \|f(x) - f(y)\| < \gamma \|x - y\| \text{ for } 0 < \gamma < 1$$

במילים: עבור שני וקטורים במרחב, אופרטור $(\cdot)^\pi$ 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: $\|(s)^\pi - u\|_\infty = \max_{s \in \mathcal{S}} |u(s) - v(s)|$. 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 $(x)^\pi = x$ וסדרה $(x_t)^\pi = x_{t+1}$ 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 המתќבל בצורה האיטרטיבית קרוב מאוד לפתרון המתќבל 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 לבחר בה ורק לאחר מכן לבצע בהתאם לאסטרטגיה $(s|a)\pi$ הנגזרת מחישוב ה-Value function. למעשה, ניתן לחפש את כל הפעולות עבורן כדי לבצע פעולה מסוימת עבורו התגמול יהיה גבוה יותר מאשר האסטרטגיה של $(s)\pi'$. באופן פורמלי יותר, נרצה להגדיר אסטרטגיה דטרמיניסטית, עבורו בהסתברות 1 ננקוט בכל מצב s בפעולה הכי כדאי a :

$$\pi'(s) = \arg \max_{a'} \mathcal{Q}^\pi(s, a')$$

נשים לב שרגע זה הוא בעצם להשתמש באסטרטגיה גרידית – בכל מצב ננקוט בפעולה הכי משלימה בטוחה של צעד אחד. השאלה היא כמובן – מדוע זה בהכרח נכון? לעומת, אם הרעיון של לבחור **בכל** צעד את a האופטימלי בהכרח טוביל לקבל אסטרטגיה אופטימלית עבור כל הזרים כולם יחד? בכך להוכיח זאתணסח זאת ממשפט:

בהינתן 2 אסטרטגיות π' , π , כאשר π' דטרמיניסטיבית, אז כאשר $\mathcal{Q}^\pi(s) > \mathcal{Q}^{\pi'}(s)$ בהכרח לכל s יתקיים: $\mathcal{V}^\pi(s) > \mathcal{V}^{\pi'}(s)$. ראשית נפתח לפני הגדרה:

$$\mathcal{V}^\pi(s) < \mathcal{Q}^\pi(s, \pi'(s)) = \mathbb{E}_\pi[r_{t+1} + \gamma \cdot \mathcal{V}^\pi(s_{t+1}) | s_t = s, a_t = \pi'(s)]$$

כיוון שהאסטרטגיה הינה דטרמיניסטיבית, הפעולה הנבחרת אינה רנדומלית ביחס ל- π' , ולכן נוכל לרשום:

$$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot \mathcal{V}^\pi(s_{t+1}) | s_t = s]$$

cutת לפני אותו אי שווין שבנה חוצה נוכל לבצע את אותו חישוב גם לצעד הבא s_{t+2} :

$$< \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot \mathcal{Q}^\pi(s_{t+1}, \pi'(s_{t+1})) | s_t = s]$$

זה שווה ל:

$$= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot \mathcal{V}^\pi(s_{t+2}) | s_t = s]$$

וכך הלאה, ולמעשה הוכחנו את הדריש – נקיית הפעולה הכי עילית בכל מצב תמיד תהיה יותר טובה מהפתרון של $(s)\pi'$. cutת יש בידינו שתי טכניקות שאנו יודעים לבצע:

Evaluation (E) – בהינתן אסטרטגיה מסוימת נוכל לפתור את משוואות בלמן ולקבל את $\mathcal{Q}^\pi(s, a)$ -ים.

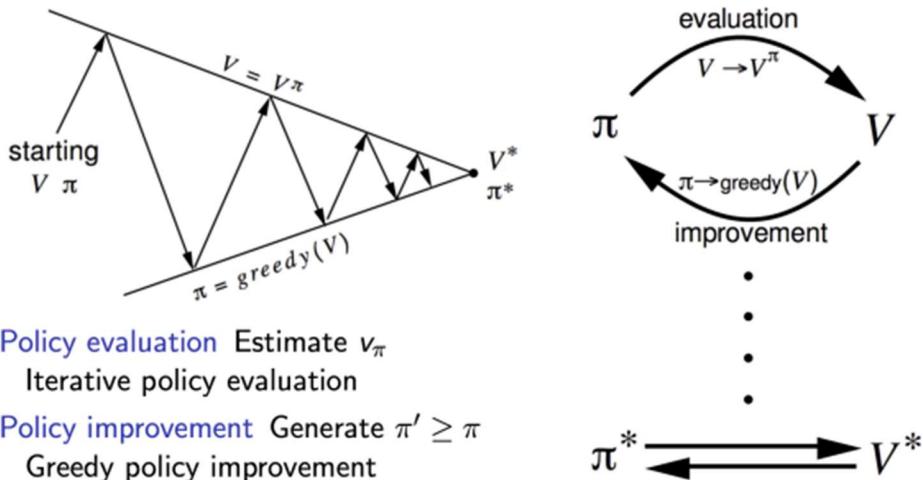
Improvement (I) – בהינתן הערך של value function של π , נוכל לבצע איטרציות המורכבות משתי הטכניקות האלה:

ניתן להתחילה אסטרטגיה רנדומלית, ואז לבצע איטרציות המורכבות משתי הטכניקות האלה באופן הבא:

$$\pi_0 \xrightarrow{E} \pi_1 \xrightarrow{I} \pi_2 \xrightarrow{E} \pi_3 \xrightarrow{I} \dots$$

תהליך זה נקרא Policy iteration – בכל צעד בו יש לנו אסטרטגיה נפתרו עבורה משוואות בלמן ובכך נחשב את ה- Value function שלה, ולאחר מכן נשפר את האסטרטגיה באמצעות policy improvement, שכמובן מבוצע בחירה גרידית שבתוך הקצר טובה יותר מאשר ה-Value function שחישבנו. ניתן להוכיח שאחרי מספר סופי של איטרציות האסטרטגיה תתכנס לנקודת שbat (fixed point), ואז הפעולה הבאה לפיה האסטרטגיה תהיה זהה לבחירה הgridית:

$$\pi(s) = \arg \max_a \mathcal{Q}^\pi(s, a) = \pi'(s)$$



איור 11.4 – ביצוע איטרציות של Policy improvement ו-Policy evaluation על מנת למצוא בכל שלב את ה-value function ולשפר אותו באמצעות בחירה גריידית.

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 בנך יש לו נקודת שבת יחידה. כיוון שהראינו שימוש ב-*policy iteration* מביא את האסטרטגיה לנקודת שבת מסוימת, נוכל לצרף לכך את העבודה שהאופרטור שהגדכנו הינו העתקה מכווצת וממילא קיבל שאותה נקודת שבת הינה יחידה, וממילא אופטימלית.

Value Iteration

הראנו שבעזרת שיטת *policy iteration* ניתן להגיע לאסטרטגיה אופטימלית, אך התהליך יכול להיות איטי. ניתן לנוקוט גם בגישה יותר ישירה ולנסות לחשב באופן ישיר את הפתרון של משוואות האופטימליות של בלמן (ופתרון הינו אופטימלי כיוון שהראינו שהפתרון הוא נקודת שבת יחידה). נתחיל עם פתרון רנדומלי V_0 ולאחר מכן נקבע איטרציות באופן הבא עד שנגיע להתקנסות:

$$\mathcal{V}_{k+1} = \max_a \sum_{s'} p_\pi(s, s') (\mathcal{R}_\pi(s, s') + \gamma \cdot \mathcal{V}_k(s'))$$

נשים לב שהשיטה זו אין לנו מידע לגבי האסטרטגיה אלא רק חישבנו את ה-Value function, אך ממנה ניתן לגוזר את Q ואז לבחור באסטרטגיה גרידית, שהינה במקורה זה גם אופטימלית:

$$\pi(s) = \arg \max_a Q^\pi(s, a)$$

ניתן להראות כי בשיטה זו ההתקנסות מהירה יותר 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 והסיבבה 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 ההצעה **בהינתן** מודל מרקובי. 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 כמפורט בפיעול a (סימנו את זה בתור $\mathcal{P}_{ss'}^a = p_\pi(s, s')$, ובנוסף נתון לנו מה ה-reward המתקבל עבור כל action (סימנו את זה בתור $\mathcal{R}_{ss'}^a = \mathcal{R}_\pi(s, s')$).

בעזרת שתי הנחות פיתחנו את המשוואות בלבד, כאשר הינו לנו שני צמדים של משוואות. משוואות בלבד עבור אסטרטגיה נתונה נקבעות באופן הבא:

$$\mathcal{V}^\pi(s) = \sum_{a, s'} \pi(a|s) \mathcal{P}_{ss'}^a (\mathcal{R}_{ss'}^a + \gamma \cdot \mathcal{V}^\pi(s'))$$

$$Q^\pi(s, a) = \sum_{s'} \mathcal{P}_{ss'}^a \left(\mathcal{R}_{ss'}^a + \gamma \sum_{a'} Q^\pi(s', a') \right)$$

ובנוסף פיתחנו את המשוואות עבור הפתרון האופטימלי:

$$\mathcal{V}^*(s) = \max_a \sum_{s'} \mathcal{P}_{ss'}^a (\mathcal{R}_{ss'}^a + \gamma \cdot \mathcal{V}^*(s'))$$

$$Q^*(s, a) = \sum_{s'} \mathcal{P}_{ss'}^a \left(\mathcal{R}_{ss'}^a + \gamma \max_a Q^*(s', a') \right)$$

הראינו שתי דרכי להגעה לפתרון האופטימלי:

1. Policy improvement המורכב מ-Policy evaluation ולאחריו Policy iteration.

2. Value iteration – פתרון משוואות בלבד באופן ישיר באמצעות איטרציות על ה-Value function.

כאמור, דרכי פתרון אלו מניחים שהמודול ידוע, ובנוסף שמדובר במצבים איננו גדול מדי וכי יכול להיות מוצג בזכרון. האתגר האמייתי מתחילה בנקודה בה לפחות אחת מהנחות אלה אינה תקיפה, ולמעשה פה מתחילה התפקיד של אלגוריתמי RL. עיקר ההתקנות של אלגוריתמים אלו יהיה למצאו באופן יעיל את האסטרטגיה האופטימלית כאשר לא נתונים הפרמטרים של המודול, וזאת לשערך אותם (Model-based learning) או למצוא דרך אחרת לחישוב האסטרטגיה האופטימלית ללא שימוש במודל (Model free learning). אם למשל יש משחק בין משתמש לבין המחשב, אלגוריתמים השייכים ל-Model based learning ינסו ללמידה את המודל של המשחק או להשתמש במודל

קיים, ובעזרת המודל הם ינסו לבחון כיצד יגיב המשתמש לכל תור שהמחשב יבחר. לעומת זאת אלגוריתמים מסווג Model free learning לא יתעניינו בכך, אלא ינסו ללמידה ישירות את האסטרטגיה הטובה ביותר עבור המחשב.

היתרון המשמעותי של אלגוריתמים המשתיכים על המודל של הבעיה (Model-based) נובע מהיכולת לתקן מספר צעדים קדימה, כאשר עברו כל 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 גו. 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 מציבה כמהאתגרים, כאשר העיקרי ביניהם הוא הסכנה שהסטרטגיה הנלמדת תהיה טוביה רק עבור המקרים אותם ראה הסוקן, אך לא תתאים 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 $Q_\theta(s, a)$. מציאת המשערך האופטימלי יכולה להתבצע על ידי חיפוש θ שيسפק את השערוך הטוב ביותר ביותר שנייתן למצוא, או על ידי מציאת הפעולה שתמקסם את המשערך:

$$a(s) = \arg \max_a Q_\theta(s, a)$$

השיטות המנסות למצוא אופטימום לאסטרטגיה הן לרוב policy-ho, כלומר כל פעולה נקבעת על בסיס האסטרטגיה המעודכנת לפי הפעולה הקודמת. Q-learning – 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 למצוא ישר את האסטרטגיה הטובה ביותר ביזה, בעוד שאלגוריתם Q-learning רק משערך את $Q(s, a)$, ולעתים השעריך לא מספיק ואז התוצאה המתבקשת אינה מספקת טובה. מצד שני, כאשר השעריך מצליח, הביצועים של Q-learning טובים יותר, כיון שהשימוש במידע על העבר מנוצל בצורה יעילה יותר מאשר אלגוריתמים המבצעים אופטימיזציה של האסטרטגיה. שתי הגישות האלה אינן זרות לחלווטין, וישנם אלגוריתמים שמנוסים לשלב בין הרעיוןות ולנצל 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 הווודאות לא קיים, ולכן ניתן להתמקד בביצועים אסימפטומטיים. במקרים בהם המודל אינו ידוע, הדגש העיקרי הוא על למידת המודל.