# שיעור 1

# שיעור 2

## Regularization

### **Dropout**

Dropout הוא טכניקת רגולריזציה בשימוש ברשתות נוירונים עמוקות שנועדה למנוע תופעת **אוברפיטינ.** הרעיון הבסיסי של Dropout הוא שבכל איטרציה של אימון, חלק מהנוירונים מוסרים באופן אקראי (מוגדר כ-"dropping"), כלומר מפסיקים לתפקד באותה איטרציה. הסיכוי להוריד נוירון נקבע על ידי היפר-פרמטר שנקרא **הסתברות Dropout**, כאשר ערך נפוץ הוא 0.5.

**למה Dropout מועיל?**

1. **מניעת הסתגלות משותפת של תכונות (Co-Adaptation)**:
   * כאשר כל נוירון יכול להסתמך על נוירונים ספציפיים אחרים, הרשת עשויה ללמוד דפוסים שלא יכללו תכונות כלליות יותר.
   * Dropout מאלץ את הרשת ללמוד **ייצוגים מיותרים ורדונדנטיים**, שכן אף נוירון אינו יכול להסתמך לחלוטין על נוירונים אחרים.
   * לדוגמה: אם נרצה לזהות "חתול" על סמך מאפיינים כמו: יש לו אוזניים, הוא פרוותי, יש לו זנב. Dropout יוודא שכל אחד מהמאפיינים הללו יכול לתרום בנפרד לזיהוי המושג "חתול", ולא רק השילוב המדויק שלהם.
2. **גישה של אימון מודלים רבים (Ensemble Models)**:
   * Dropout מדמה מצב שבו אנו מאמנים **מספר רב של מודלים קטנים**, שכל אחד מהם משתמש בתת-קבוצה אחרת של פרמטרים.
   * כל "מסיכה בינארית" (binary mask) שמגדירה אילו נוירונים פועלים היא למעשה מודל עצמאי. *התוצאה תהיה ממוצע של התתי מודלים.*
3. **שיתוף פרמטרים (Shared Parameters)**:
   * כיוון שכל המודלים משתמשים באותם פרמטרים של רשת האם, משיגים חסכון חישובי ושיפור כללי בביצועים.

**איך Dropout מיושם בפועל?**

* במהלך האימון, נוירונים מוסרים באקראי (או "מכובים") בהתאם להסתברות מסוימת.
* בשלב האימות (inference), לא משתמשים בDropout, אך הערכים המתקבלים מנורמלים כך שיתאימו למבנה הכולל של הרשת.

**יתרונות עיקריים:**

* מונע אוברפיטינג.
* יוצר רשת עמידה ומגוונת יותר – רובסטית.
* משפר את יכולת הכללה (generalization) של הרשת לדוגמאות חדשות.

**Test Time:**

בשלב הבדיקה (Test Time), יש להתמודד עם האקראיות שמוסיפה טכניקת הDropout במהלך האימון. Dropout מכבה נוירונים באופן אקראי בשלב האימון כדי למנוע הסתגלות משותפת של תכונות ולשפר את היכולת הכללית של הרשת. עם זאת, בשלב הבדיקה, חשוב להבטיח פעולה יציבה ודטרמיניסטית של המודל. הפתרון לכך הוא למצע את האפקט של האקראיות על ידי התאמת המשקלים: כל נוירון פעיל מוכפל בהסתברות Dropout (p) כדי להתחשב באפקט האקראיות שנוצר במהלך האימון.

גישה זו מאפשרת לרשת לפעול בצורה קבועה ודטרמיניסטית בזמן הבדיקה, מבלי לאבד את היתרונות של Dropout. במקום להריץ את Dropout בפועל, המשקלים מותאמים כבר בשלב האימון כך שיפצו על הנוירונים המכובים. כתוצאה מכך, המשקלים המחושבים מייצגים את התנאים שהיו במהלך האימון, והרשת נותרת יציבה. היתרון המרכזי הוא מניעת אקראיות שעלולה להשפיע על הביצועים ושמירה על אמינות המודל בעת יישום על נתונים חדשים.

**Inverted Dropout**

Inverted Dropout היא גישה נפוצה ומתקדמת לטכניקת Dropout ברשתות נוירונים, המיועדת למנוע אוברפיטינג ולשפר את יציבות המודל. בשיטה זו, ההתאמות הנדרשות לשימור ממוצע ההפעלות (activations) מתבצעות כבר בשלב האימון. כל נוירון מופעל בהסתברות , אך ערכי ההפעלה מוכפלים ב כדי להבטיח שממוצע ההפעלות נשאר עקבי. כלומר, אם , כל ערך מופעל מוכפל ב , מה שמאזן את האפקט של נוירונים מכובים (dropped-out) בשכבה.

בשלב הבדיקה, אין צורך בהתאמות נוספות למשקלים או לערכי ההפעלה, מכיוון שהממוצע כבר הותאם במהלך האימון. זה הופך את התהליך לפשוט ויעיל יותר, ומאפשר למודל לפעול בצורה יציבה ודטרמיניסטית. יתרון זה הופך את Inverted Dropout לנפוץ במימושים מודרניים, שכן הוא מפשט את הקוד ומשפר את הביצועים על נתונים חדשים, תוך שמירה על יכולת הכללה (generalization) מיטבית.

# שיעור 3

## Optimizers

### **הקדמה**

**Gradient Descent**

Gradient Descent הוא תהליך שבו מתבצעת אופטימיזציה של פונקציית עלות (Loss) על ידי חישוב גרדיאנט (שיפוע) ומעבר בכיוון ההפוך לשיפוע. המטרה היא למצוא את הערכים המינימליים של פונקציית העלות.

בשיטה זו, בכל איטרציה מחושבים הגרדיאנטים של פונקציית העלות ביחס למשקלים (W), ומתבצע עדכון על בסיס ערך זה.

**בעיות עיקריות ב-SGD**

* 1. **שינוי שיפוע בקצבים שונים לכל כיוון**
* אם פונקציית העלות משתנה מהר בכיוון אחד (תלול) ואיטי בכיוון אחר (שטוח), האלגוריתם יתקשה להתכנס למינימום. הוא ינוע באיטיות רבה בכיוון השטוח ויקפוץ הלוך ושוב בכיוון התלול.
* זו בעיה ידועה כאשר פונקציית העלות בעלת מספר גדול של משתנים עם **High Condition Number**, כלומר יחס גדול בין הערך העצמי הגבוה ביותר והנמוך ביותר של מטריצת ההסיאן.
  1. **Local MinimaוSaddle Points**
* Gradient Descent עשוי "להיתקע" בנקודות מינימום מקומיות (Local Minima) או בנקודות אוכף (Saddle Points) שבהן הגרדיאנט מתאפס, מה שעלול להאט מאוד את תהליך הלמידה.
* Saddle Points נפוצים במיוחד בבעיות במימדים גבוהים.
  1. **רעש בגרדיאנטים מMini Batches**
* ב SGD חישוב הגרדיאנטים נעשה לעיתים קרובות על קבוצות קטנות של נתונים (Mini Batches) כדי לחסוך בחישובים. עם זאת, חישובים אלו עלולים להיות "רועשים" (noisy), דבר שגורם לתנועה אקראית שאינה יציבה בתהליך האופטימיזציה.

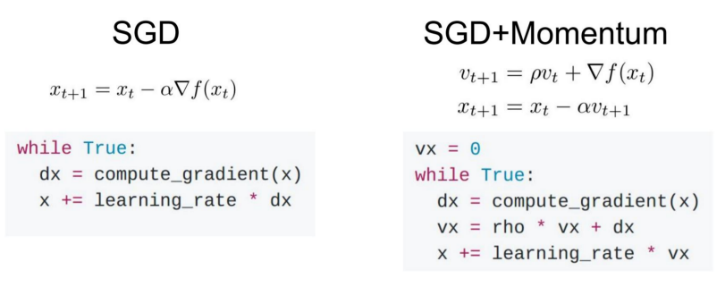
**פתרונות נפוצים**

* **שיטות אופטימיזציה מתקדמות** כמו Adam, RMSprop, ו-Adagrad מתמודדות עם בעיות אלו על ידי התאמת קצב הלמידה (Learning Rate) לכל פרמטר בנפרד, הפחתת רעשים ושיפור היציבות בתהליך הלמידה.
* שימוש בפונקציות סדירות ואלגוריתמים מתקדמים מאפשרים להתמודד עם תנאי Saddle Points ולשפר את היכולת למצוא את המינימום הגלובלי.

### **Momentum**

שיטת Momentum נועדה להוסיף מהירות (velocity) לתהליך האופטימיזציה על ידי התחשבות בתנועה הקודמת של הגרדיאנטים. הרעיון הוא לחבר את הגרדיאנט הנוכחי עם "זיכרון" של הגרדיאנטים הקודמים כך שהתנועה תהיה חלקה יותר ופחות מושפעת מרעש או מהמכשולים בדרך (כמו Saddle Points).

העדכון ב SGD + Momentum מתבצע כך:

כאשר ρ הוא פרמטר החיכוך (לרוב ρ=0.9 או (ρ=0.99שמתאר עד כמה זיכרון התנועה הקודמת נשמר.

עדכון הפרמטרים (משקלים)

כך שהמיקום החדש משלב את הגרדיאנט הנוכחי ואת כיוון המהירות המצטברת.

**יתרונות Momentum**

1. **שיפור יציבות**: שיטה זו מסייעת "להחליק" רעשים גרדיאנטיים (Gradient Noise) שנוצרים עקב חישוב על קבוצות נתונים קטנות (Mini Batches).
2. **מעבר מהיר דרך כיוונים שטוחים**: במקרים של כיוונים עם קצבי שינוי איטיים (Poor Conditioning), התוספת של המהירות מאפשרת התקדמות מהירה יותר.
3. **התגברות על Saddle Points**: במקום להיתקע בנקודות אוכף (Saddle Points), המהירות המצטברת עוזרת לדחוף את האופטימיזציה קדימה.
4. **הקטנת תנודות בכיוונים תלולים:**  Momentumמייצב את המסלול של תהליך האופטימיזציה כך שהתנועה לא תהיה קופצנית מדי.