- ערכים גבוהים או נמוכים מדי יגרמו לפונקציה לעבוד בתחום לא מתאים (גרדיאנט 0)

שיטות נורמליזציה:

- Min-Max: $v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$
- Z-Score: $v' = \frac{v - mean_A}{stand\_dev_A}$

אלגוריתמים לאופטימיזציות:

נרצה לעשות את הלמידה בצעדים קטנים יותר (ללא קפיצות גדולות). נשתמש בקצב למידה (LR).

- GD
- SGD
- Mini-Batch GD
- Momentum
- AdaGrad
- RMSprop
- Adam

GD:

מטרה: ביצוע אופטימיזציה. מזעור ה-Loss.

יש w וקטור משקולות שבעזרתו ניתן משקל לכל פיצ'ר.

תכונות:

- קירוב טוב לגרדיאנט
- צריך לעבור בכל איטרציה על כל הדוגמאות
- מתכנס בצורה איטית

$$w^t \leftarrow w^{t-1} - eta * \frac{\partial L}{\partial w^{t-1}} \qquad where\ w^t\ is\ the\ vector\ in\ iteration\ t$$

בכל עדכון מחשבים את הגרדיאנט על פי הממוצע על כל דוגמאות סט האימון.

ההתכנסות איטית (לאפוק).

SGD:

תכונות:

- נדגום רנדומלית דוגמא אחת בכל איטרציה
- לאחר מספיק איטרציות האלגוריתם יתכנס

אותה פונקציית עדכון.

GD VS SGD:

GD איטי יותר. SGD יותר "רועש" ופחות צפוי.

Mini-Batch GD:

דומה ל-SGD אך בכל איטרציה נדגום k דוגמאות אקראיות וב"ת.

Momentum:

ל-SGD קשה באיזורים בהם המשטח של הדוגמאות מתעקל בצורה תלולה יותר (עם שיפוע תלול יותר), במימד אחד יותר מאשר באחר. במקרים האלו SGD עובר במישור בתנודות גדולות אך עם התקדמות מאוד איטית לאופטימום המקומי. לכן נרצה להשתמש ב-momentum.

זוהי בעצם מתודה שהיא תוספת ל-SGD שעוזרת לכוון את וקטורי הגרדיאנטים לכיוון הנכון מהר יותר (התכנסות מהירה יותר). שיטה זו כמעט תמיד עובדת טוב יותר ומהר יותר מאשר SGD.

כלל עדכון:

$$v_t = \gamma v_{t-1} + eta * \frac{\partial L}{\partial \theta_i} \ , \qquad \theta_i = \theta_i - v_t$$

AdaGrad:

אלגוריתם לאופטימיזציה מבוססת גרדיאנטים שמתאים את קצב הלמידה (eta) לכל פרמטר.

- נפחית את קצב הלמידה של משקולות עם גרדיאנטים גבוהים.
- נעלה את קצב הלמידה של משקולות שמקבלות עדכונים קטנים או לא מתעדכנים בצורה עקבית.

כלל עדכון:

$$c_{t,i} = (\sum_{k=0}^{t} \frac{\partial L}{\partial \theta_{k,i}})^2 \ , \ \ \theta_{t+1,i} = \theta_{t,i} - \frac{\frac{\partial L}{\partial \theta_{t,i}}}{\sqrt{c+\epsilon}}$$

RMSprop:

במקרה של deep learning ל-AdaGrad יש מגרעה, השינויים בקצב למידה שלו לרוב הם מאוד אגרסיביים וגורמים לעצירת הלמידה מוקדם מדי. לכן במקרה הזה נשתמש בשיטה הזו.

כלל עדכון:

$$c_{t,i} = \gamma c_{t-1,i} + (1-\gamma)(\frac{\partial L}{\partial \theta_{t,i}})^2 \ , \ \ \theta_{t+1,i} = \theta_{t,i} - \frac{\frac{\partial L}{\partial \theta_{t,i}}}{\sqrt{c+\epsilon}}$$

ערכים טיפוסיים ל-γ הם 0.9, 0.99, 0.999.

שלא כמו AdaGrad עדכונים לא נהיים קטנים יותר מונוטונית.

Adam

בעקרון זה שילוב של momentum ו-RMSprop. מעשית זה האלגוריתם הדיפולטיבי בהרבה חבילות.

כלל עדכון:

$$m_{t,i} = \gamma_1 m_{t-1,i} + (1-\gamma_1)\frac{\partial L}{\partial \theta_{t,i}}$$

$$c_{t,i} = \gamma_2 v_{t-1,i} + (1-\gamma_2)(\frac{\partial L}{\partial \theta_{t,i}})^2$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{m_{t,i}}{\sqrt{v_{t,i}+\epsilon}}$$

ערכים מומלצים הם: $\gamma_1 = 0.9 , \gamma_2 = 0.99$

<u>למיד PAC (PAC learnability):</u>

נגדיר δ – הסיכוי שנעבוד על סט דוגמאות לא מייצג.

ε – אי דיוק הסיווג.

$$E_{x,y}[l(y, f_\theta(x))] > \varepsilon => classifier\ failed$$

$$E_{x,y}[l(y, f_\theta(x))] \leq \varepsilon => classifier\ approximatly\ correct$$

סט היפותיזות H נקרא למיד PAC ביחס לפונקציית $\bar{l}(x,y;h)$ אם קיים מס' מינימלי של דוגמאות אימון m שמוגרל בצורת ב"ת ומאותה התפלגות מהסתברות D ומתקיים:

$(x,y)\sim D,\ \ E\left[\bar{l}(x,y;h)\right] \leq \min_{h'\in H} E\left[\bar{l}(x,y;h')\right] + \epsilon$ בהסתברות $1-\delta$ על בחירת m הדוגמאות.

זה אומר שקיים סט היפותיזות שעבורו אם נקבל מספיק דוגמאות (m) אז נצליח למצוא סיווג טוב בשגיאה עד ϵ בהסתברות $1-\delta$.

<u>משפט:</u> סט היפותיזות סופי הוא למיד PAC.

$$P_S\left(\left|\frac{1}{m}\sum_{i=1}^{m}\bar{l}(x_i,y_i;h) - E\left[\bar{l}(x,y;h)\right]\right| < \varepsilon\right) > 1-\delta$$

במילים: דיוק גבוה של הלמידה ← ההפרש בין הממוצע על סט האימון לבין תוחלת הדוגמאות שלא ראינו הוא קטן.

<u>סוגי אלגוריתמים:</u>

- KNN
- K Means
- Perceptron
- bayesian network
- SVM
- Convolutional neural network (CNN)