יעל בן-ארי

רותם בן חמו

Stochastic Gradient descent

MNIST digit classification

<u>מטרת הפרויקט:</u>

בניית רשת נוירונים המסוגלת לזהות בסבירות גבוהה סיפרה הכתובה בכתב יד שמתקבלת כקלט.

<u>הקדמה והגדרות:</u>

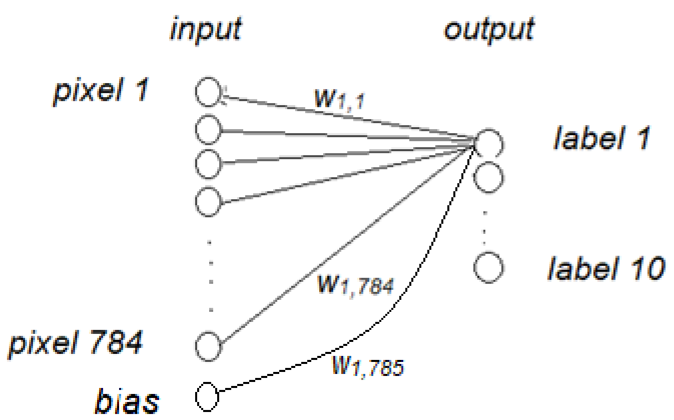
בכדי ליישם את המטרה נשתמש במאגר הנתונים MNIST, הכולל 60,000 תמונות מתויגות של ספרות בין 0 ל-9.

על מנת ללמוד את מאגר הנתונים באופן יעיל ונכון, נגדיר פונקציית מטרה המתארת את השגיאה של הרשת עבור מאגר הנתונים הנתון. נרצה לאמן את הרשת על ידי מציאת נקודת מינימום בשיטה איטרטיבית לפונקציית המטרה, כלומר מציאת מינימום שגיאה (או קירוב מספק למינימום השגיאה).

לשם כך נממש את האלגוריתם Stochastic Gradient Descent(SGD).

נגדיר:

- $X \epsilon R^{n*m}$ $set\ of\ m\ images, each\ with\ size\ n\ pixels.$
- $labeled\ data\ \{x_i, y_i\}_{i=1}^{m}, y_i \epsilon \{1 \dots l\}, x_i \epsilon R^n.$
- $F(W)\ is\ the\ objective\ function. W \in R^{l*n}.$
- $w_i \in R^n,\ i \epsilon \{1 \dots l\},$
  $is\ a\ vector\ of\ unknown\ weights\ that\ we\ will\ find$
  $by\ minimizing\ the\ objective\ function. i.e\ by\ calculating$
  $$argmin_{W \in R^{l*n}} F(W)$$
- $Indicator: I_{k=j}\ equals\ 1\ if\ k=j, otherwise\ 0.$
- $C_j = [I_{y1=j} \dots I_{ym=j}]^T, j \epsilon \{1 \dots l\}$
- $\alpha\ -\ learning\ rate$



שרטוט סכמתי של רשת הנוירונים במימוש שלנו:

<u>הגדרת פונקציית המטרה softmax:</u>

ישנן אפשרויות שונות לבחירת פונקציית המטרה, בבעיית הclassification שלנו נבחר במודל logistic regression.

פונקציית המטרה softmax מתאימה את מודל השגיאה של logistic regression לבעיה שלנו, המכילה 10 תוויות אפשריות לתמונה.

הפונקציה מתארת את השגיאה של הרשת עבור מאגר הנתונים הנלמד, על סמך W- המשקולות ברשת שלנו:

$$L_X(\{w_k\})_{k=1}^{10} = -\sum_k c_k^T * \log\left(\frac{\exp(X^T w_k)}{\sum_j \exp(X^T w_j)}\right)$$

אתחול המשקולות נעשה על ידי מתן ערכים רנדומליים בטווח [0,1).

<u>הגדרת הגרדיאנט של פונקציית המטרה:</u>

נגזור את פונקציית המטרה לפי וקטור המשקולות w_p. נקבל:

$$\nabla w_p L_X = X\left[\sum_j \frac{\exp(X^T w_p)}{\sum_j \exp(X^T w_j)} - c_p\right]$$

בנוסף, נגדיר וקטור *bias*: $b \in R^{10}$, כך שחישוב ערך השגיאה יכלול פרמטר חופשי. במימוש שלנו נוסיף פיקסל עם ערך 1 לכל תמונה שנלמד.

<u>אלגוריתם SGD</u>:

אלגוריתם SGD בוחר בכל איטרציה תת - קבוצה אקראית מתוך מאגר הנתונים $(X_s \subseteq X)$, ומחשב גרדיאנט עבור פונקציית המטרה המתארת את השגיאה של הקבוצה האקראית שנבחרה. גרדיאנט זה (∇L_{X_s}) מהווה הערכה לנגזרת של פונקציית המטרה המקורית, ומאפשר חישוב מהיר יותר של הנגזרות. מכיוון שאנו מחפשים נקודת מינימום, בכל איטרציה נעדכן את משקולות הרשת בכיוון ההפוך לגרדיאנט.

<u>הערה</u>: פונקציית המטרה שהצגנו היא סכום של השגיאות של כל התמונות במאגר הנתונים X. את שגיאה זו נחלק בגודל מאגר הנתונים, ולכן כאשר נתייחס לפונקציה המחשבת שגיאה עבור תת קבוצה אקראית X_s, נחלק את השגיאה בגודל תת הקבוצה ונקבל הערכה לשגיאה המקורית על סמך הקבוצה האקראית.

בסיום האלגוריתם אנו מבצעים שתי בדיקות:

1- בדיקת אחוזי הצלחה בפענוח של מאגר הנתונים שלמדנו, על מנת להעריך את הלמידה שביצענו.

2- בדיקת אחוזי הצלחה בפענוח של מאגר נתונים שלא למדנו.

נצפה לראות אחוזי הצלחה דומים וגבוהים בשתי הבדיקות.

<u>הלולאה הראשית במימוש שלנו באופן סכמטי:</u>

Load and shuffle MNIST training data X.

Add bias: add 1 pixel with value 1 to each image.

Initiate $W \in R^{10*785}$ with random values in range [0,1).

for k=1…maxIter:

  calculate loss: $L_X(W)$ ##to show improvement and convergence

  ~~S~~ ← list of B random batches

  for each s in ~~S~~:

    F'(W) ← $\frac{1}{|s|} \nabla L_{X_s}$

    W ← W - α * F'(W) ##α is the learning rate

classifyImages(training data X) ##checking success rate on the training data

classifyImages(testing data)##checking success rate on the testing data

<u>הערה</u>: מכיוון ש $w^T x$ לא חסום, חישובי האקספוננט עלולים להיות לא בטוחים. לכן הגדרנו

$$\eta = max_j\{x^T w_j\}$$

ובחישובי האקספוננט החסרנו ערך זה ע"מ להבטיח חישוב בטוח ולמנוע numerical overflow:

$$\frac{\exp(x^T w_k)}{\sum_j \exp\left(x^T w_j\right)} = \frac{\exp(x^T w_k - \eta)}{\sum_j \exp\left(x^T w_j - \eta\right)}$$

בנוסף, ערכי הפיקסלים המקוריים הם בטווח [0,255]. לאחר טעינת התמונות אנו מבצעים להם נרמול ומחלקים אותם ב255 לערכים בטווח [0,1].

<u>בדיקת הגרדיאנט:</u>

מימשנו את בדיקת הגרדיאנט המתוארת בקובץ NOA.pdf, על מנת לוודא את נכונות חישוב הגרדיאנט שלנו לפונקציית המטרה.

ראשית בחרנו וקטור רנדומלי d בגודל 785- כגודל תמונה, וערך ϵ_0 התחלתי רנדומלי.

השווינו במספר איטרציות בין הערכים המתקבלים מהמשוואות:

$$|f(x + \epsilon_i \mathrm{d}) - \mathrm{f(x)}| = O(\epsilon_i)$$

$$|f(x + \epsilon_i d) - f(x) - \epsilon_i d^T grad(x)| = O({\epsilon_i}^2),$$

כאשר באיטרציה ה-i: $\epsilon_i = (0.5)^i \epsilon_0$

קיבלנו כמצופה שהערכים יורדים בפקטור של חצי ורבע בהתאמה.

תוצאות:

נציג את תוצאות האלגוריתם עבור ריצה של 100 איטרציות, כאשר בכל איטרציה מבצעים B=30 שיפורים עבור 30 batches .

כיוון שגודל batch יחיד משפיע על איכות ההערכה של הגרדיאנט שאנו מחשבים לגרדיאנט פונקציית המטרה (המתחשבת בכל מאגר הנתונים), נציג תוצאות עבור גדלי batch שונים :

Batch Size	Training data success rate	Testing data success rate
1	77.7%	77.8%
5	83.9%	84.0%
10	90.0%	90.0%
20	90.4%	90.2%
50	91.3%	91.3%
100	91.5%	91.6%

כמובן שניתן לציין גם תוצאות ריצות ארוכות יותר וקומבינציות שונות של הפרמטרים השונים, אך בסיכום הבדיקות השונות מגיעות לאחוז הצלחה מקסימלי של כ-91% עבור הTesting data ועבור הTraining data.

מצורפים קבצי התכנית:

Preproccesing.py – טעינת הData, נרמול הפיקסלים והוספת פיקסל bias.

Main.py – מימוש הSGD.

gradientTest.py – מימוש בדיקת הגרדיאנט.

*הקוד נכתב ונבדק ב Anaconda, Inc :: Python 3.6.4.