שיפורים לאלגוריתם מורד הגרדיאנט

מומנטום

לאלגוריתם SGD פורסמו תוספות ושיפורים רבים, אשר הפופולרי שבהם הוא שימוש **במומנטום**. הרעיון מאחוריו פשוט: נדמיין את האלגוריתם כחלקיק הנע במרחב הפרמטרים במורד הר – זו פונקציית המחיר. עם תנועתו, החלקיק צובר מהירות אשר משפיעה על כיוון תנועתו בהמשך. על החלקיק מופעל כוח בכיוון השלילי של הגרדיאנט בנקודה הנוכחית, הכיוון התלול ביותר מטה, אך תנועתו של החלקיק אינה מושפעת רק מכוח זה, שכן הוא הגיע אל הנקודה הנוכחית עם מהירות מסויימת, זהו המומנטום הדוחף אותו להמשיך את התנועה בכיוון המקורי, שאינו בהכרח הכיוון התלול ביותר בנקודה הנוכחית.

בנוסחאות, תוספת זו באה לידי ביטוי בכך שאנו שומרים ממוצע רץ של הגרדיאנטים שחושבו באיטרציות הקודמות, זהו וקטור **המהירות**, v_t, לו אנו מוסיפים את הגרדיאנט החדש, ∇C_t, ובוקטור זה אנו משתמשים לעדכון ערכי הפרמטרים, בדומה ל-SGD הרגיל. נוסחת העדכון המתקבלת היא

$$v_t = \beta v_{t-1} + \nabla C_t$$

$$Parameters = Parameters - \alpha v_t$$

כאשר β הוא פרמטר המומנטום אשר לרוב נבחר בין 0 ל-1. הביטוי המפורש עבור v_t הוא

$$v_t = \sum_{k=0}^{t} \beta^{t-k} \nabla C_k .$$

מנוסחה זו אפשר להבין את משמעות הפרמטר β: הוא מבטא את המשקל שהגרדיאנטים הקודמים מקבלים בעת עדכון הפרמטרים. כאשר $\beta = 1$ מתקבל

$$v_t = \sum_{k=0}^{t} \nabla C_k$$

ומשמעות הדבר שכל הגרדיאנטים ההיסטוריים מקבלים משקל שווה בחישוב הצעד הבא של האלגוריתם. כאשר $\beta = 0$ משקלם של הגרדאינטים הקודמים מתאפס, והאלגוריתם זהה ל-SGD ללא מומנטום. לרוב נבחר ערך $0 < \beta < 1$, וכך גרדיאנטים חדשים יותר יקבלו משקל יתר בחישוב צעד העדכון הנוכחי.

יש לשים לב שבדומה לחלקיק הצובר מהירות בתנועתו במורד ההר, כך גם גודל הצעד ב-SGD עם מומנטום עשוי לגדול, ככל שהגרדיאנטים הקודמים נצברים בוקטור המהירות. ראו כי גרדיאנט אשר חושב באיטרציה t ממשיך להשפיע על התנועה גם באיטרציות העוקבות: באיטרציה הבאה יתווסף לוקטור המהירות הערך $\beta \nabla C_t$, באיטרציה העוקבת, $\beta^2 \nabla C_t$ וכיוצא בזה. לבסוף, השפעתו הכוללת של גרדיאנט זה על ערכי הפרמטרים, בהנחה שבוצעו T איטרציות נוספות, היא

$$-\alpha \left(\sum_{k=0}^{T} \beta^k \nabla C_t \right) = \left(-\alpha \sum_{k=0}^{T} \beta^k \right) \nabla C_t \approx -\frac{\alpha}{1-\beta} \nabla C_t .$$

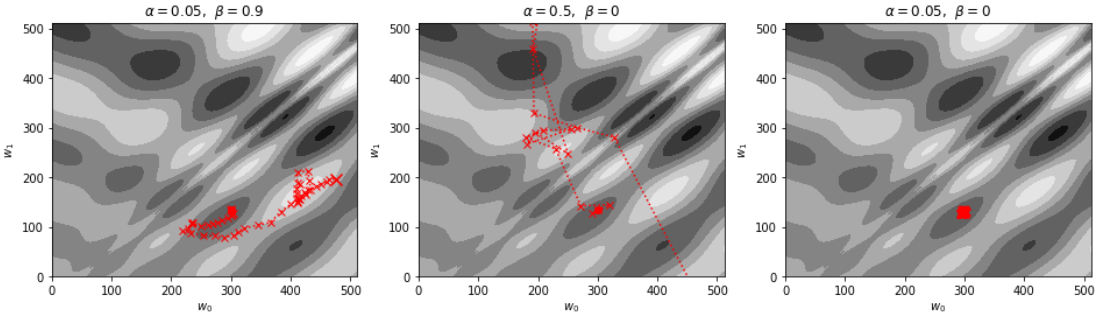
בהתאם נהוג לבחור את גודל הצעד ההתחלתי, או את מתזמן קצבי הלמידה במחשבה שגודל הצעד האפקטיבי יכול להגיע אף עד $\frac{\alpha}{1-\beta}$ ולא α.

מלבד האצת ההתכנסות, הבאה לידי ביטוי בצעדים גדולים יותר, יתרונות אפשריים נוספים של שימוש במומנטום הם הפחתת ההשפעה של רנדומליות הגרדיאנט המקרי על כיוון התנועה עם התקדמות האלגוריתם, וכן האפשרות לדלג מעל נקודות מינימום מקומי אם האלגוריתם מגיע אליהם במהירות חיובית.

בהיותו כה פופולרי, אלגוריתם המומנטום מובנה בתוך אובייקט האופטימיזציה הבסיסי של PyTorch. כל שיש לעשות הוא להעביר ערך מתאים עבור פרמטר המומנטום, כלהלן.

```
optimizer = torch.optim.SGD(model, lr=alpha, momentum=beta)
```

ראו באיור הבא דוגמה להשפעה של השימוש במומנטום על ריצת אלגוריתם מורד הגרדיאנט המקרי, כפי שהפעלנו אותו על פונקציית קרטון הביצים בפרק הקודם.



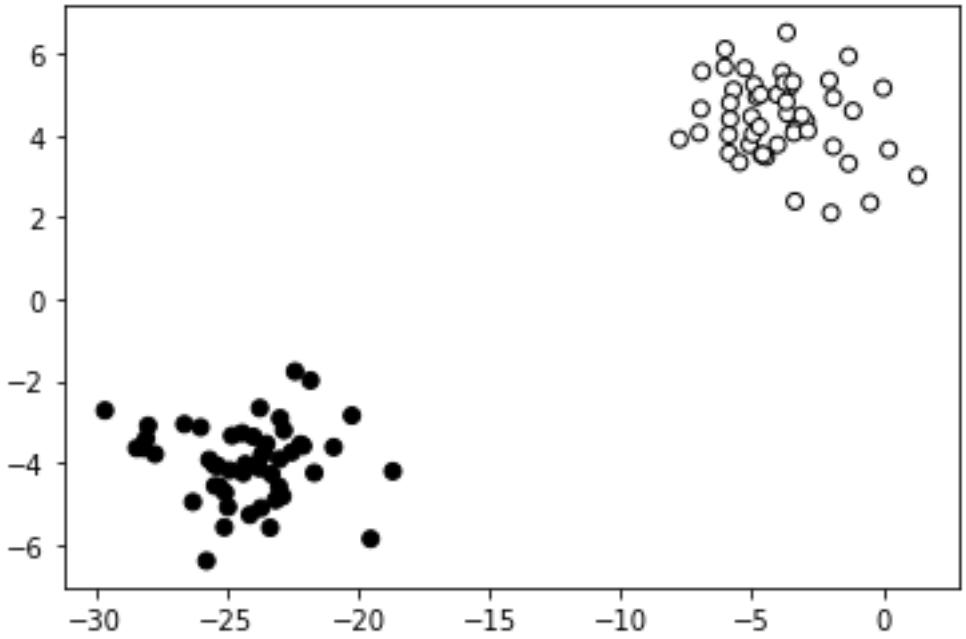
שימו לב לכך שבאיור הימני קצב הלמידה הוא קטן (ואין מומנטום) ובהתאם האלגוריתם אינו זז מנק' ההתחלה, בעוד שבאיור השמאלי קצב הלמידה ההתחלתי קטן, אך האלגוריתם צובר תאוצה וגודל הצעד גדל עד אשר לבסוף המומנטום מביא את האלגוריתם אל נקודת המינימום הקרובה. באיור האמצעי ניתן לראות ששימוש בקצב הלמידה הגבולי של המומנטום, $\frac{\alpha}{1-\beta}$, ללא המומנטום עצמו מוביל להתבדרות.

קצב למידה אדפטיבי

לעתים פונקציית המחיר תהיה תלויה בכל אחד מפרמטרי המודל בצורה שונה – שינוי קטן באחד ישפיע משמעותית על ערכה, בעוד ששינוי קטן באחר יהיה משמעותי פחות. מצב זה בו הפרמטרים בעלי סדרי גודל שונים יוצר קשיים באופטימיזציה ולצערנו ניתקל בו לא פעם. נוכל לדמות בנקל סיטואציה זו בעזרת שינוי קל במודל הנוירון היחיד לסיווג נקודות שחורות ולבנות. נדגום את הנקודות כפי שעשינו בעבר, אך לאחר זאת – את אחד המימדים נכפול בקבוע גדול.

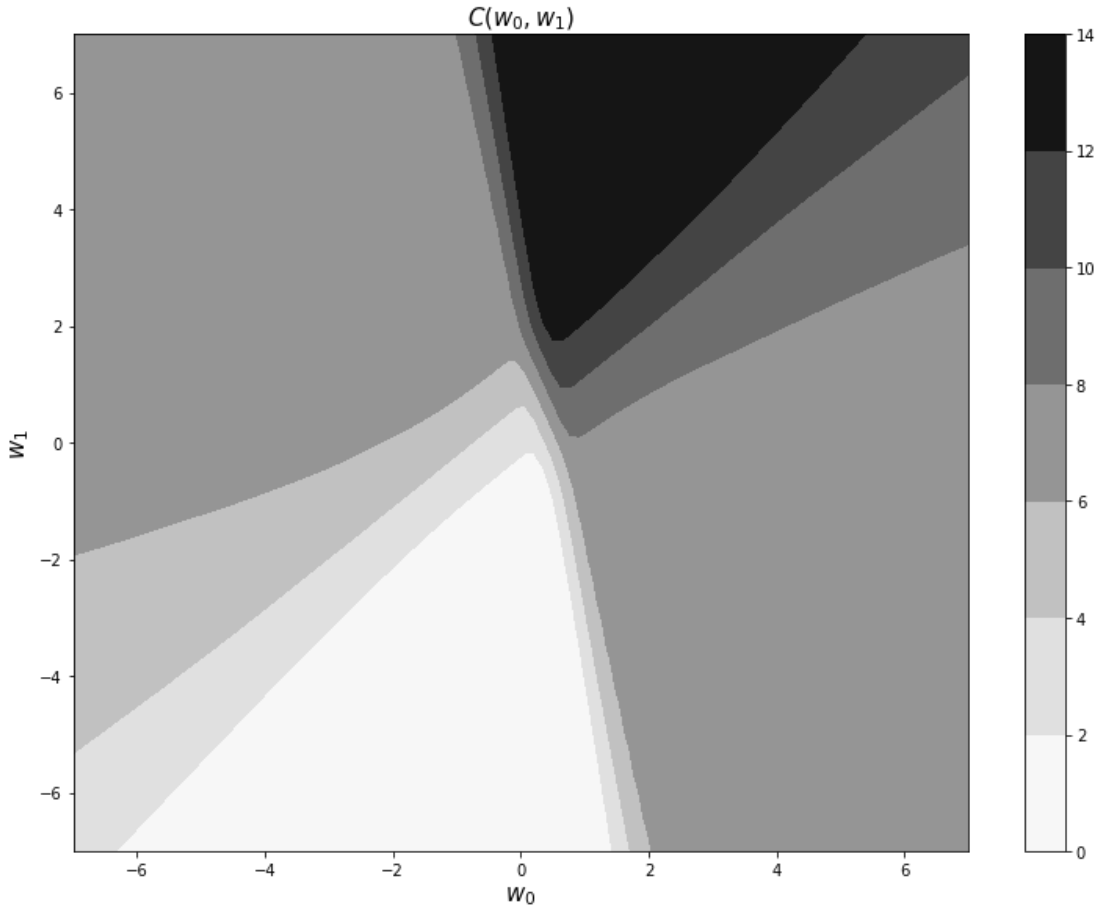
```
import sklearn.datasets as skds
X, Y = skds.make_blobs(n_samples=100,n_features=2,
                       centers=2,random_state=1)
X, Y = torch.tensor(X), torch.tensor(Y)
X[:,0]= 2.5*X[:,0]
plt.scatter(X[:, 0], X[:, 1],
            c=Y, cmap="Greys", edgecolor="black");
```

פלט:

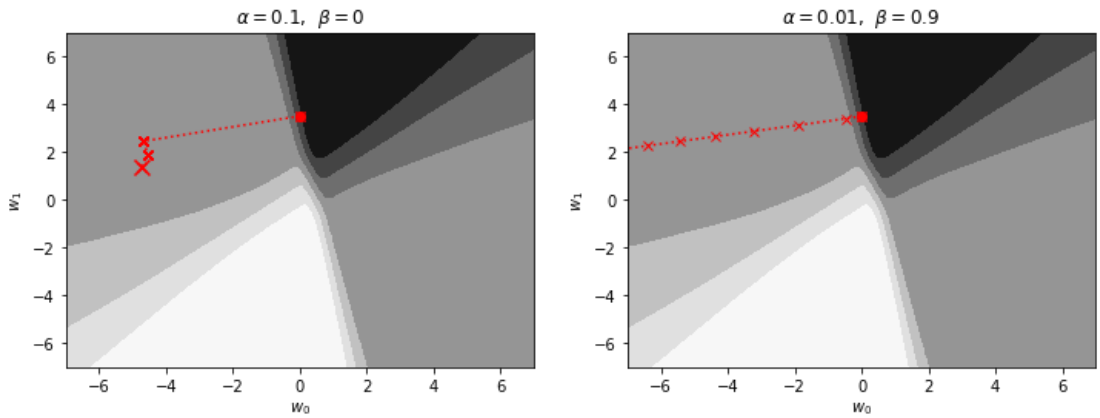


יש לשים לב כעת לשינוי בציר ה-X: השונות בו גדולה משמעותית מבעבר.

נזכיר שבמודל זה שלושה פרמטרים, w_0, w_1, b וכמו בדיון הקודם, נבחר את b להיות קבוע בערכו על 5 ונצייר את משטח המחיר התלוי בשני הפרמטרים האחרים.



מאיור זה ניכרת ההשפעה של כפל המימד הראשון בקבוע גדול על פונקציית המחיר – היא רגישה הרבה יותר לשינויים במשתנה w_0 מאשר ב- w_1. בהתאם, הנגזרת $\frac{\partial C}{w_0}$ תהיה גדולה משמעותית מאשר $\frac{\partial C}{w_1}$ וכיוון התנועה של אלגוריתם מורד הגרדיאנט, לכל גודל צעד יהיה כמעט אופקי. גם שימוש במומנטום אינו יעזור במקרה זה, כפי שניתן לראות באיור הבא.



אחד הפתרונות לבעיה זו הוא להשתמש בגדלי צעד שונים עבור הפרמטרים השונים: פרמטרים רבי השפעה יעודכנו בעדינות, בצעדים קטנים ופרמטרים פחות משמעותיים יעודכנו בצעדים גדולים יותר. רעיון זה עומד מאחורי האלגוריתמים Adagrad ו-RMSprop, ולבסוף בא לידי ביטוי באלגוריתם הפופולרי Adam, המשלב גודל **צעד אדפטיבי לכל פרמטר** יחד עם מומנטום. ב-Adam אנו שומרים בנוסף לוקטור המהירות גם וקטור סקאלה המתעדכן בצורה דומה, בו אנו משתמשים לנרמל את עוצמת התנועה עבור כל פרמטר בנפרד. נוסחת עדכון הפרמטרים היא אם כן,

$$v_t = \beta_1 v_{t-1} + (1-\beta_1)\nabla C_t$$

$$s_t = \beta_2 s_{t-1} + (1-\beta_2)\left(\nabla C_t\right)^2$$

$$Parameters = Parameters - \alpha \frac{v_t}{\sqrt{s_t}}$$

וכאן יש לשים לב שחישוב $\left(\nabla C_t\right)^2$ מתבצע על ידי העלאה בריבוע איבר-איבר של הוקטור ∇C_t.

התוצאה המתקבלת היא שפרמטרים בעלי נגזרות קטנות יעודכנו בקצב מהיר יותר. לאלגוריתם שלושה פרמטרים אשר יש לקבוע לפני ריצתו: β_1, המקביל במשמעותו לפרמטר המומנטום, β_2, אשר מהווה פרמטר הזכרון של וקטור הסקאלה ו- α, קצב הלמידה המוכר לנו. חשוב לתת את הדעת לכך שזהו קצב הלמידה האפקטיבי, בשונה מ-SGD עם מומנטום, שכן וקטור העדכון של הפרמטרים עובר נרמול בכל איטרציה, ולכן המהירות הנצברת "מקוזזת" עם הצטברות גורם הנרמול.

נוסחת העדכון הנ"ל שימושית להבנת תפקוד האלגוריתם, אך במציאות יש לה מספר חסרונות: ראשית, תיתכן חלוקה באפס בעת הנרמול, ולכן מוסיפים למכנה מספר קטן, ϵ. שנית, נהוג לבחור את הפרמטרים β_2, β_1 קרובים ל-1, ובהתאם באיטרציות הראשונות ערכי v_t ו- s_t יהיו קטנים מאוד. בעיה זו אנו פותרים על ידי נרמול הוקטורים בכל איטרציה, ונוסחת העדכון המתקבלת היא

$$\hat{v}_t = \frac{v_t}{1-(\beta_1)^t}, \quad \hat{s}_t = \frac{s_t}{1-(\beta_2)^t}$$

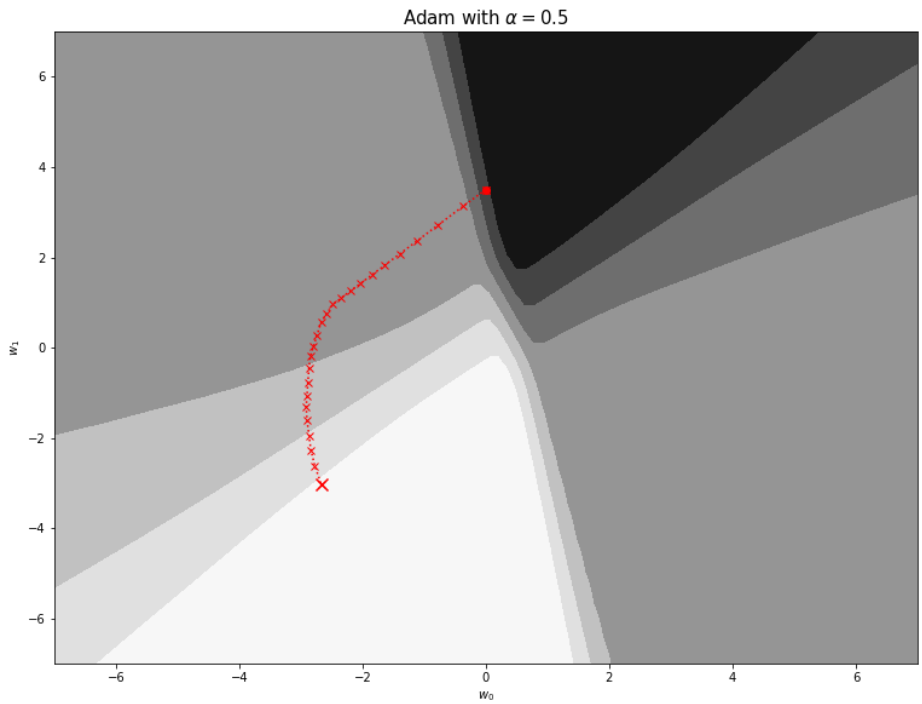
.

$$Parameters = Parameters - \alpha \frac{\hat{v}_t}{\sqrt{\hat{s}_t} + \epsilon}$$

השימוש ב-Adam ב-PyTorch פשוט. כל שיש לעשות הוא להחליף את אובייקט האופטימיזציה, כלהלן.

```
optimizer = torch.optim.Adam(model, lr=alpha)
```

תוצאת הפעלת Adam על הדוגמה האחרונה, מאויירת בהמשך, וניתן לראות בבירור את היתרון של השיטה על פני הקודמות.



נסיים פרק זה בסייג חשוב: ישנן מעט תוצאות תיאורטיות המבטיחות את התכנסות האלגוריתמים למינימום מקומי של פונקציית המחיר וכן עוד פחות תוצאות המאפשרות לנו לדעת מראש איזה אלגוריתם עדיף על פני אחר. המצב חמור עוד יותר בהקשר של רשתות נוירונים עמוקות, שכן התוצאות הקיימות אינן תקפות לגבי פונקציות המחיר שלהן. בהתאם, עיקר הידע בבחירת האלגוריתם הנכון מגיע דרך ניסוי וטעיה. בהמשך, לא פעם נפעל לפי כללי אצבע מקובלים אשר אינם מגובים בהוכחה תיאורטית אך קיים קונצנזוס מקצועי התומך באפקטיביות שלהם, למשל כאשר נבחר את הפרמטרים של Adam להיות $\beta_1 = 0.9$, $\beta_2 = 0.99$ ולו רק מפני שעשו זאת לפנינו בהצלחה לא מעטה.

שאלות לתרגול

1. הניחו שפונקציית המחיר היא ליניארית בפרמטרים, קרי $C(w_0,...,w_n) = \sum_{k=0}^{n} c_k w_k + b_c$, כאשר $(w_0,...,w_n)$ הם פרמטרי המודל, וכן שאלגוריתם מורד הגרדיאנט עם מומנטום מאותחל בנקודה $(w_0,...,w_n) = (0,...,0)$.
   א. מצאו ביטוי פשוט לערכי הפרמטרים לאחר האיטרציה ה-t.
   ב. חשבו את המרחק בין ערכי הפרמטרים בין שתי איטרציות.
   ג. האם תוכלו להסביר מדוע קצב הלמידה הגבולי של SGD עם מומנטום הוא (במקרים מסויימים) $\frac{\alpha}{1-\beta}$? **רמז:** השתמשו בנוסחת הסכום של טור הנדסי.
2. חזרו לדוגמה האחרונה בה הראינו ש-Adam עדיף על פני SGD (רגיל או עם מומנטום) ושנו את נקודת ההתחלה של האלגוריתמים, קצב הלמידה ואת פרמטר המומנטום. מצאו שילוב ערכים בהם ביצועיו של Adam פחותים.