בס"ד

ליז אהרוניאן 316584960
אורי בן-זקן 311492110
רז שינקמן 311130777

Deep Learning – Ass4

שאלות:

- **Which paper you chose the implement.**
- **Why you chose that particular one.**
- **What method was used in the paper?**
- **What was the result reported in the paper.**
- **Did your code manage to replicate this result?**
- **What was your performance on that dataset (how does your report compare to theirs)?**
    - **Report your accuracies on the train and test sets, and your learning curve graphs**
- **What was involved in replcating the result?**
    - **If you tried several hyperparameters, or several approaches, describe them, and what was the result of each.**
    - **If you didn't manage to replicate the result, describe your attempts in details.**
        - **"I tried really hard and couldn't make it to work" is not a proper answer.**
        - **"I tried A and B and C and D, and I got such as and such results from them, and then I tried E and ..." etc, is good.**
- **What worked straightforward out of the box? what didn't work?**
- **Are there any improvements to the algorithm you can think about?**

תשובות:

המאמר שבחרנו לממש הוא מאמר בשם " A Decomposable Attention Model for Natural Language Inference". כותבי המאמר הינם: Ankur P. Parikh, Oscar Tackstrom, Dipanjan Das וכן Jakob Uszkoreit. המאמר פורסם ביוני 2016.

בחרנו לממש דווקא אותו משום שראינו שכותבי המאמר הגיעו לביצועים טובים על הdev וה-train, שכן הם הגיעו לתוצאות accuracy של 86.3% ו-88.5% בהתאמה.

בנוסף, ראינו כי כותבי המאמר השתמשו בשיטות אימון המוכרות לנו מהשיעור:
שימוש ב- pretrained words embedding, שימוש ברשתות נוירונים, וכן attention weights.

על המאמר:

נתאר כעת בפירוט את פעולת המודל של כותבי המאמר, ולאחר מכן נתייחס לקוד שכתבנו ולפעולות שנקטנו כדי לנסות להגיע לאותם אחוזי הצלחה.

המשימה: NLI - natural language inference - בהינתן שני משפטים לדעת להגיד את הקשר ביניהם מתוך אחת האופציות הבאות:

- Contradiction.
- Neutral.
- Entailment.

בס"ד
ליז אהרוניאן 316584960
אורי בן-זקן 311492110
רז שינקמן 311130777

הכנות לפני האימון: כותבי המאמר השתמשו ב-pretrained word embedding בגודל 200.
כפי שלמדנו בהרצאה, word embedding הם וקטורים המייצגים כל מילה שנתקלנו בה בשלב האימון. השימוש בהם מסייע לנו להתחיל את האימון עם ייצוג וקטורי דומה עבור מילים דומות במשמעותן ומסייע לנו ללמוד בצורה מהירה ויעילה יותר.

כותבי המאמר בחרו למפות מילים שלא נתקלו בהם באימון (OOV – Out Of Vocabulary) ל100 random embeddings.

בנוסף, הם הסירו דוגמאות שמופו ל"-" (כלומר לא התקבלה החלטה ע"פ רוב לתיוג עבורן) וצמצמו ומיקדו את הdataset.

שלבי האימון:

עבור כל זוג משפטים a,b נסמן בai כל מילה במשפט a וכן בbi כל מילה במשפט b.

שלב 1 – attend:

תחילה נחשב attention weights באופן הבא:

$$e_{ij} := F'(\bar{a}_i, \bar{b}_j) := F(\bar{a}_i)^T F(\bar{b}_j)\,.$$

F הינה feed forward neural network עם פונקציית אקטיבציה RELU.

לאחר מכן מנרמלים את הattention weights:

$$\beta_i := \sum_{j=1}^{\ell_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_b}\exp(e_{ik})}\bar{b}_j\,,$$
$$\alpha_j := \sum_{i=1}^{\ell_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_a}\exp(e_{kj})}\bar{a}_i\,.$$

שלב 2 – compare:

משווים כל מילה עם הattention weight המנורמל המתאים לה, ונעזרים בפונ' G שגם היא feed forward network.

$$\mathbf{v}_{1,i} := G([\bar{a}_i, \beta_i]) \quad \forall i \in [1,\ldots,\ell_a]\,,$$
$$\mathbf{v}_{2,j} := G([\bar{b}_j, \alpha_j]) \quad \forall j \in [1,\ldots,\ell_b]\,.$$

נעיר שהקלט לפונ' G הוא שרשור של ai עם הבטא i המתאים. או bj עם שרשור של האלפא j המתאים.

שלב 3 – aggregate:

מבצעים סכימה של הוקטורים שקיבלנו בשלב הקודם (עבור a בנפרד, ועבור b בנפרד), באופן הבא:

$$\mathbf{v}_1 = \sum_{i=1}^{\ell_a}\mathbf{v}_{1,i} \quad , \quad \mathbf{v}_2 = \sum_{j=1}^{\ell_b}\mathbf{v}_{2,j}\,.$$

לסיום –

מכניסים את 1V וכן 2V לתוך H, פונקציית ההיפותזה, היא מחזירה y_hat שהינו וקטור הסתברויות שעליו ניתן לבצע argmax כדי לדעת מה התיוג המתאים.

בס"ד
ליז אהרוניאן 316584960
אורי בן-זקן 311492110
רז שינקמן 311130777

כותבי המאמר השתמשו בmulti-class cross-entropy loss.

<u>היפר פרמטרים שצוינו במאמר:</u>

4 – Batch size

0.2 – Dropout ratio

0.05 – Learning rate

200 – Hidden layer size

2 – Hidden layers number

השיטה של כותבי המאמר מתעלמת מסדר המילים בתוך המשפטים. השיטה שלהם הוכיחה, שלפחות בשביל המשימה הזו (NLI) השוואות זוגות מילים (כפי שעשינו בשלב 2), חזקה בהרבה בהשוואה לייצוג שלם של כל המשפט.

הצעות שיפור לאלגוריתם:

1. כפי שציינו, כותבי המאמר נעזרו בfeed forward neural networks בשלבים 1 (attend) ו-2 (compare). כותבי המאמר בחרו שעומק הרשתות הנ"ל הינו 2. ניתן להעמיק את הרשתות הנ"ל ולהוסיף להן מספר שכבות. אמנם רשת עמוקה יותר בד"כ תצטרך הרבה data כדי לעבוד טוב, אך נראה שסט הדוגמאות המתוייגות גדול ורחב מספיק.

2. ביצוע נרמול לword embedding – במסגרת החיפושים אחר שיפור האלגוריתם, מצאנו מאמר שמדבר על נרמול word embedding לטובת שיפור התוצאות. שם המאמר הינו: [Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation]. המאמר מתחיל ומדבר על חשיבות השימוש בword embedding בבעיות של תרגום למשל. Word embedding מספקות ייצוג חזק למילים המופיעות בtrain data. הembeddings מקודדות את המילים כך שיימצא דמיון בקידוד למילים הדומות במשמעותן ומשפרות את זמני האימון ועוזר להגיע לתוצאות טובות מהר יותר.
ביצענו בקוד שלנו נרמול על הglove שהינן ה-word embedding שאיתם עבדנו.

המודל שלנו:

עיבוד המידע:

בתחילה, פרסרנו כל שורה באופן בו לקחנו את sentence1 וכן את sentence2, והפרדנו בין המילים בעזרת split לפי רווח.
ראינו שאנו מגיעים לתוצאות פחות טובות, הבנו שהפרסור שלנו לא טוב מספיק, זאת משום שאנו מפצלים כל משפט למילים ביחד עם סימני פיסוק.
הפתרון שחשבנו עליו היה לפרסר כל משפט לפי העלים בparsing tree וכך היה לנו קל ליצור lists של המילים מכל משפט ללא סימני פיסוק שהופיעו במשפט (כגון נקודה בסוף כל משפט וסימני פיסוק באמצע).

היפר פרמטרים:

ביצענו ניסיונות לכייל את הפרמטרים השונים כגון: גודל השכבות הנסתרות, גודל הבאטצ', מספר וגודל שכבות של ה-feed forward neural networks, רמת הdropout, learn rate.

epochs – מספר הפעמים שמריצים את אלגוריתם האימון, חשוב מצד אחד שנלמד מספיק טוב ונעשה מספיק חזרות על סט האימון כדי שלא נגיע לunderfitting, אך מצד שני עלינו להימנע ממצב של overfitting אם נאמן את המודל "יותר מידיי". מניסיונות שונים שערכנו ראינו שהמודל שלנו מגיע לתוצאות טובות לאחר 10 epochs.

בס"ד
ליז אהרוניאן 316584960
אורי בן-זקן 311492110
רז שינקמן 311130777

Batch size – המודל רץ במשך המון זמן, לפיכך ניסינו להגדיל את גודל הבאצ' איתו עבדו כותבי המאמר (4) לגודל באצ' משמעותי יותר, כך שנעבור על יותר דוגמאות ונעדכן פחות פעמים את הפרמטרים.
לצערנו, הניסיון הסתיים בתוצאות פחות טובות על הdev, לבסוף לאחר ניסיונות שונים, בחרנו לעבוד עם באצ' בגודל 32.

Dropout ratio – זוהי טכניקה עבור רגולריזציה, כלומר שיטה למניעת overfitting ברשת נוירונים, ע"י מניעת מצב שהמודל שלנו מאומן יותר מידיי על הtrain data ואז לא יודע להתמודד כמו שצריך עם דוגמאות חדשות שמעולם לא ראה. שיטה זו למעשה משמיטה באופן רנדומלי מרשת הנוירונים תוך כדי האימון. זה מונע co-adaptation לסט האימון. מדד הdropout שלנו זהה לכותבי המאמר והינו 0.2.

Learning rate – 0.05, מקדם הלמידה, משפיעה על העוצמה שבה הפרמטרים מעודכנים.

Hidden layer size - 100 נוירונים בכל שכבה נסתרת, לעומת כותבי המאמר שבחרו להשתמש ב200 נוירונים בכל שכבה נסתרת. בחרנו במספר זה כדי לשפר את זמן הריצה ומשום שראינו שהוא לא פוגע בתוצאות.

Hidden layers number – מפאת משך זמן האימון הממושך לא באמת הצלחנו לבחון את דרך פעולתן של שכבות נוירונים עמוקות מהרגיל ולאחר כמה נסיונות בחרנו להישאר עם העומק שכותבי המאמר בחרו בו והוא 2.

Optimizer - בחנו אופטימיזרים שונים כגון Adam שהינו שילוב של Momentum וRMSprop שעליהם למדנו בקורס למידת מכונה.

לבסוף בחרנו בAdagrad, Adagrad הינו אלגוריתם עבור אופטימיזציה מבוססת גרדיאנטים שמתאימה את הלמידה עבור כל פרמטר. עבור משקולות שמקבלות נגזרות גבוהות, מקדם הלמידה שלהם יופחת, לעומת משקולות שמתעדכנים לעיתים רחוקות ואז מקדם הלמידה שלהם בכלל העדכון יעלה. נוסחת העדכון בשימוש באופטימיזר זה היא כזו:

AdaGrad Update

$$c_{t,i} = \left(\sum_{k=0}^{t} \frac{\partial L}{\partial \theta_{k,i}} \right)^2$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\frac{\partial L}{\partial \theta_{t,i}}}{\sqrt{c + \epsilon}}$$

בס"ד

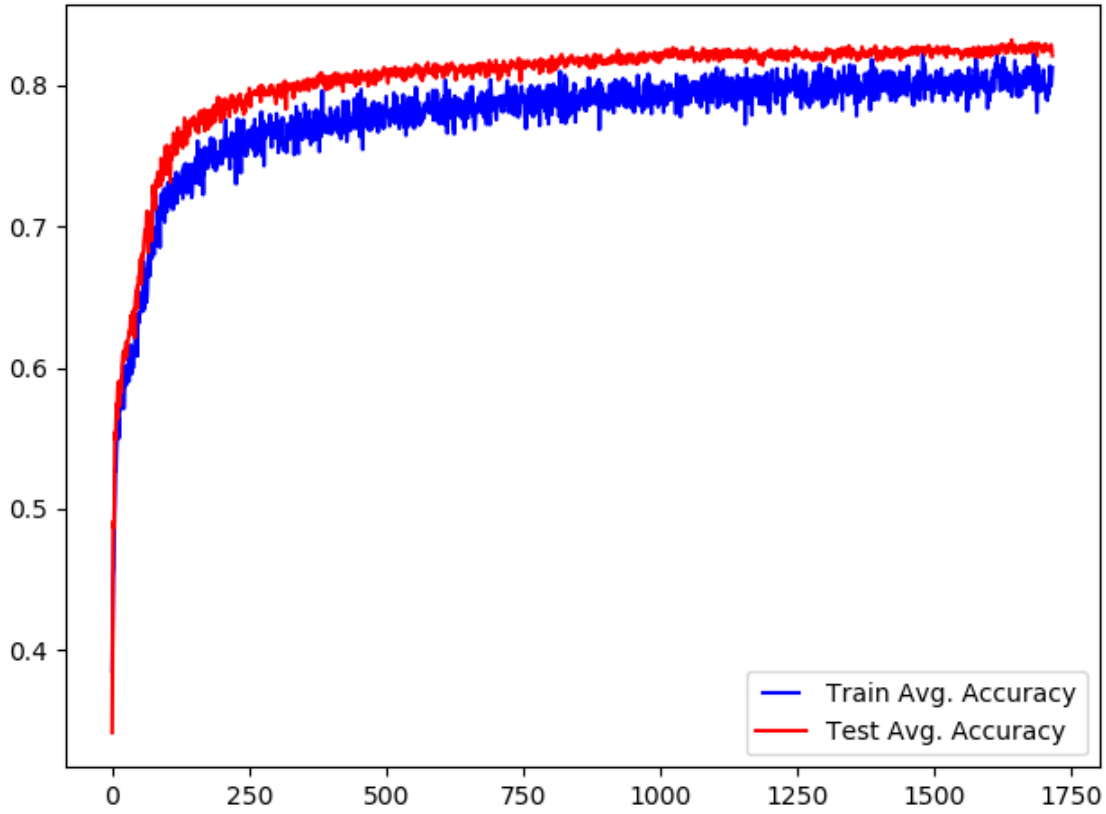
ליז אהרוניאן 316584960
אורי בן-זקן 311492110
רז שינקמן 311130777

תוצאות המודל שלנו:

תחילה, כדי להציג ולהסביר את התוצאות שהמודל שלנו הגיע אליהם, נציג את הגרפים המכילים מידע אודות הaccuracy והloss על הtrain והtest.
עבדנו כאמור בגודל באטצ' של 32, לאחר כל 100 פעמים שלמדנו עם באטצ' מתוך הtrain, אספנו נתונים אודות הtrain וכן הtest, כלומר מדדנו loss וaccuracy של המודל שלנו על הdatasets הנ"ל ושמרנו את התוצאות במילון הממפה בין מספר הפעמים שלקחנו 100 batch-ים מהמידע שבtrain, לloss\accuracy על הtest והdev כך שנוכל לשרטט את הגרפים ולהשוות בינהם.

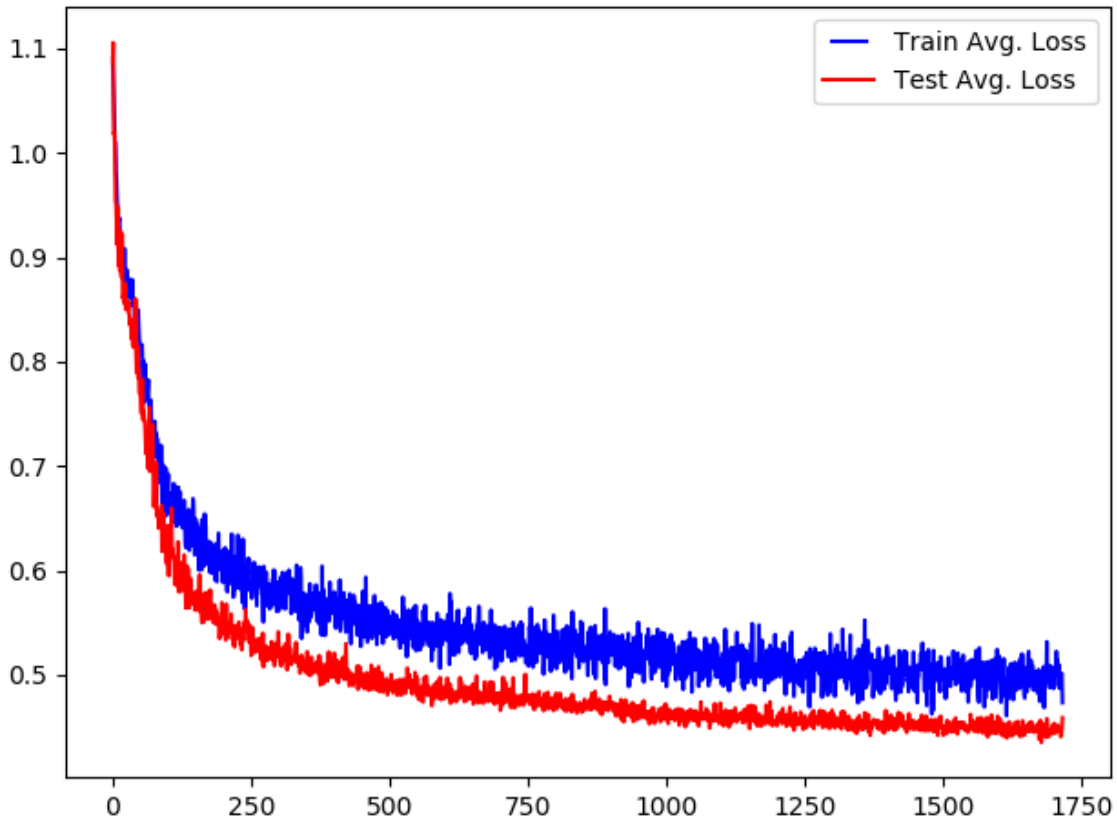
גרפים:

גרף 1 – תוצאות accuracy על הtrain והtest:



בס"ד
ליז אהרוניאן 316584960
אורי בן-זקן 311492110
רז שינקמן 311130777

גרף 2 – תוצאות loss על train וtest:



[ציר האיקס מייצג את מספר הbatch-ים חלקי 100].

כלומר, ניתן לראות שהמודל שלנו כמעט והצליח להגיע לתוצאות של כותבי המאמר.
נזכיר שכותבי המאמר הגיעו לביצועים טובים על הtest וה-train, שכן הם הגיעו לתוצאות accuracy של 86.3% ו-88.5% בהתאמה.

אנחנו הצלחנו להגיע לaccuracy של 79.9375% על הtrain, ושל 82.8481% על הtest.

כמו כן, באפוק האחרון (העשירי במספרו), הגענו לtrain loss: 0.5003 ולtest loss: 0.4425.

ניתן לראות מהגרפים שלא הגענו לoverfitting במודל שלנו, משום שהתוצאות על הtest אפילו יותר טובות מעל הtrain, הן במדד הaccuracy והן במדד הloss. בהתחשב בעובדה הזו, ניסינו להריץ את המודל שוב, עם מספר epochs גדול יותר, אך לא ראינו שיפור.
הגרפים משקפים שיש עלייה הדרגתית בaccuracy של הtest והtrain – דבר שמראה שהמודל לומד יפה ומתקדם בלמידה ככל שהמודל מתאמן על יותר ויותר batch-ים. כמו כן, ניתן לראות גם ירידה עקבית בloss, גם עבור הtrain וגם עבור הtest – שזה מעיד על כך שהמודל פחות ופחות טועה.

הערה לבודק -

מצורף נספח של תוצאות הריצה שערכנו על המודל. ראה קובץ results.pdf אם ברצונך לעיין בתוצאות הריצה.

תודה רבה,

ליז, רז ואורי