לאחר כמה ניסיונות קצרים שנכשלו מכל מיני סיבות הבנתי שאני חייב להבין איך הנגזרת עוברת ברשת לפי כל פונקציה ושלב ברשת.

לכן אני רוצה לעבור פונקציות שונות ולהגיע לחישובים של הפונקציה שלא חורגים למספרים גדולים ולהגיע לנגזרות פשוטות שישארו בטווח ערכים סביר ויהיו קלות לחישוב.

<u>פונקציות שימושיות והנגזרות שלהם:</u>

<u>SoftMax:</u>

$$f(x_i)=\frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}$$

החישוב שיכול לחרוג פה הוא e^x אם הx גדול מידי נקבל אין סוף ונקרוס :(

אפשר לפתור את זה ע"י הורדה של הx המקסימלי בדוגמה מכל הפיצ'רים בדוגמה לפני העלאה בחזקה

ואז בהעלאה בחזקה נקבל:

$$e^{x_i-max(x)}=\frac{e^{x_i}}{e^{max(x)}}$$

ואז לכל i :

$$f(x_i-max(x))=\frac{e^{x_i}/e^{max(x)}}{\sum_{j=1}^{d} e^{x_j}/e^{max(x)}}=\frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}=f(x_i)$$

כלומר תוצאת החישוב עדיין נכונה אבל לא נחרוג במהלך החישוב
כמובן לא חשבתי על זה לבד, זה המקור:
https://www.tutorialexample.com/implement-softmax-function-without-underflow-and-overflow-/deep-learning-tutorial

לכל מספר בתוצאה הוא הושפע באופן כלשהו ע"י כל המספרים בווקטור הנכנס ולכן בחישוב הנגזרות נצטרך לחשב נגזרת מכל מספר בתוצאה לכל המספרים במקור.
כלומר מטריצה מהצורה:

$\frac{\delta softmax(X)[1]}{\delta x_1}$	$\frac{\delta softmax(X)[1]}{\delta x_2}$	$\frac{\delta softmax(X)[1]}{\delta x_3}$	...	$\frac{\delta softmax(X)[1]}{\delta x_d}$
$\frac{\delta softmax(X)[2]}{\delta x_1}$	$\frac{\delta softmax(X)[2]}{\delta x_2}$	$\frac{\delta softmax(X)[2]}{\delta x_3}$	...	$\frac{\delta softmax(X)[2]}{\delta x_d}$
$\frac{\delta softmax(X)[3]}{\delta x_1}$	$\frac{\delta softmax(X)[3]}{\delta x_2}$	$\frac{\delta softmax(X)[3]}{\delta x_3}$	...	$\frac{\delta softmax(X)[3]}{\delta x_d}$
...	...	...		...
$\frac{\delta softmax(X)[d]}{\delta x_1}$	$\frac{\delta softmax(X)[d]}{\delta x_2}$	$\frac{\delta softmax(X)[d]}{\delta x_3}$	...	$\frac{\delta softmax(X)[d]}{\delta x_d}$

בסוף לכל x_i נסכום את השורה ה i וזו הנגזרת שתזרום בו – ההצדקה שלי לזה היא שתמיד אחרי softmax צריך לבוא crossEntrophyLoss שיגרום לזה שהנגזרות ברוב הפלטים של הsoftmax יהיו 0 מבלבד 1 שהוא התיוג הנכון ואז כל ה-x יושפעו מהאחד הזה.

צורת הנגזרת: $(d_{\text{in}},)$ ניתן להכפיל בלי בעיה בנגזרת שזרמה מסוף הרשת כי היא באותה צורה

נחלק את חישוב הנגזרת לשתי מקרים:

$j=i$:

$$\frac{\delta \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}}{\delta x_i}=\frac{\delta \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}}{\delta e^{x_i}}\cdot e^{x_i}=\frac{\delta \frac{e^{x_i}}{e^{x_i}+\sum_{j=1,\, j\neq i}^{d} e^{x_j}}}{\delta e^{x_i}}\cdot e^{x_i}=\ldots$$

$$\ldots=\frac{1\cdot \sum_{i\in\{1..d\}} e^{x}-1\cdot e^{x_i}}{\sum_{i=1}^{d} e^{x_j}{}^2}*e^{x_i}=\left(\frac{1}{\sum_{i=1}^{d} e^{x_j}}-\frac{e^{x_i}}{\sum_{i=1}^{d} e^{x_j}{}^2}\right)*e^{x_i}=f(x_i)-f^2(x_i)=f(x_i)\cdot(1-f(x_i))$$

$i\neq j$:

$$\frac{\delta \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}}{\delta x_j}=\frac{\delta \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}}{\delta e^{x_j}}\cdot e^{x_j}=\frac{-e^{x_i}}{\left(\sum_{j=1}^{d} e^{x_j}\right)^2}\cdot e^{x_j}=\frac{-f^2(x_i)\cdot e^{x_j}}{e^{x_i}}=-f^2(x_i)\cdot e^{x_j-x_i}$$

זה אומר שבכל חישוב נגזרת צריך tensor בצורה: $(\text{batch size}, d, d)$ …

<u>CrossEntrophyLoss:</u>

בשלב זה מעורבים שתי וקטורים:

1. X – תוצאה של softmax
2. Y – ווקטור תיוג בצורת onehot

ואז מחשבים:

$$f(x_i)=Avg(-y_i*\ln(x_i))$$

חישוב שהוא לא מסוכן מבחינת חריגה לכן נעבור ישר לנגזרת

$$\frac{\delta f(x_i)}{\delta x_i}=\frac{\delta Avg(-y_i*\ln(x_i))}{\delta -y_i*\ln(x_i)}\frac{\delta -y_i*\ln(x_i)}{\delta x_i}=\frac{-1}{d}\frac{\delta y_i*\ln(x_i)}{\delta \ln(x_i)}\frac{1}{x_i}=\frac{-y_i}{d\cdot x_i}$$

* d הוא כמות הפיצ'רים

צורת הנגזרת היא $(d_{\text{out}},)$ שזה מימד הפלט של SoftMax

ReLu:

$$f(x_i)=\begin{matrix} 0 & if & x_i \le 0 \\ x_i & if & x_i > 0 \end{matrix}$$

הנגזרת כאן ממש פשוטה:

$$f'(x_i)=\begin{matrix} 0 & if & x_i \le 0 \\ 1 & if & x_i > 0 \end{matrix}$$

ניתן להכפיל ישירות בנגזרת משכבת הפלט כי אין הבדל בצורה

לבסוף צריך נגזרת של שכבה לינארית כלשהי:
כאן חשוב לזכור שצריך נגזרת לw אבל גם חייב נגזרת לX כי זה יכול להיות פלט של שכבה קודמת

$$f(x_i)[k]=\sum_{j=1}^{d_{\text{in}}} x_{i,j} w_{j,k}$$

הנגזרת עבור w תהיה:

$$\frac{\delta f(x_i)[k]}{\delta w_{j,k}}=\frac{\delta \sum_{j=1}^{d_{\text{in}}} x_{i,j} w_{j,k}}{\delta w_{j,k}}=x_{i,j}$$

* k – האינדקס של הפלט שממנו הנגזרת מגיעה

* j – אינדקס הפיצ'ר בקלט

* i - אינדקס הדוגמה בקלט

זה אומר שבעצם הנגזרת של השכבה הלינארית היא שיכפול של x_i לאורך ציר הפלט והכפלה בנגזרת מהפלט

הנגזרת עבור x...

גם כאן יש מטריצה כי כל $x_{i,j}$ בקלט הוא משתתף בכל אחד מהפלטים.

כאן המטריצה מהצורה:

$\frac{\delta f(x_i)[1]}{\delta x_{i,1}}$	$\frac{\delta f(x_i)[2]}{\delta x_{i,1}}$	...	$\frac{\delta f(x_i)[d_{\text{out}}]}{\delta x_{i,1}}$
$\frac{\delta f(x_i)[1]}{\delta x_{i,2}}$	$\frac{\delta f(x_i)[2]}{\delta x_{i,2}}$	...	$\frac{\delta f(x_i)[d_{\text{out}}]}{\delta x_{i,2}}$
...	...	...	...
$\frac{\delta f(x_i)[1]}{\delta x_{i,d_{\text{in}}}}$	$\frac{\delta f(x_i)[2]}{\delta x_{i,d_{\text{in}}}}$	...	$\frac{\delta f(x_i)[d_{\text{out}}]}{\delta x_{i,d_{\text{in}}}}$

ואז הנגזרת תהיה סכום העמודות.

נחשב את איברי הטבלה:

נחשב נגזרת של $x_{i,m}$ ביחס לאיבר הk בפלט

$$\frac{\delta f(x_i)[k]}{\delta x_{i,m}} = \frac{\delta \sum_{j=1}^{d_{in}} x_{i,j} w_{j,k}}{\delta x_{i,m}} = w_{m,k}$$

יוצא שהנגזרת היא הכפלה של שורות W בנגזרת מהפלט ואז סכום העמודות שלו :)

כתיבת הרשת:

ננסה לכתוב מנגנון דינאמי לרשת כעץ:

כל צומת ברשת מחזיק מטריצה שעוברת ברשת קדימה והנגזרת עד אליו מסוף הרשת

לכל צומת יש מילון ילדים שממפה כל ילד לתפקיד שלו – לדוגמא בcrossEntrophyLoss יהיה ילד שהוא התוצאה של softmax וילד שהתפקיד שלו הוא onehot של התיוגי אמת.

כל צומת יכיל את הפונקצייה שתקבל מילון של ילדים ותפקידם ותחשב את המטריצה של הצומת.

בדרך חזרה הצומת יחשב לכל ילד את הנגזרת אליו ויעביר אותה

כלומר צומת גם צריך לדעת לקבל נגזרת מההורים שלו.

הרשת עבדה למימדים קטנים (לא טוב – אבל לפחות הצליחה ללמוד משהו)

ראיתי שהבעיה נובעת מהנגזרת של crossEntrophy. מקבלים מספרים נורא גדולים במימדים לא כל כך גדולים.

לפי המקור הזה: https://towardsdatascience.com/derivative-of-the-softmax-function-and-the-categorical-cross-entropy-loss-ffceefc081d1

כדאי לאחד את הנגזרת של SoftMax ושל CrossEntrophyLoss ואז מקבלים נגזרת פשוטה לחישוב שלא תחרוג המון.

ננסה לחשב להם נגזרת מאוחדת:

* x_i פיצ'ר ה-i של קלט של הsoftmax
* y_i - 0 או 1 אם המחלקה הi נכונה
* s_i פלט של הSoftMax למחלקה ה-i

$i \neq j$:

$$\frac{\delta CrossEntrophy(s_i, y_i)}{\delta x_j} = \frac{\delta - \sum_{i=1}^{C} y_i * \ln(s_i)}{\delta s_i} \frac{\delta s_i}{\delta x_j} = \frac{\delta \sum_{i=1}^{C} y_i * \ln(s_i)}{\delta \ln(s_i)} \frac{1}{s_i} s_i^2 * e^{x_j - x_i} = \sum_{i=1}^{C} (y_i) * s_i * e^{x_j - x_i} = s_i * e^{x_j - x_i}$$

$i = j$:

$$\frac{\delta CrossEntrophy(s_i, y_i)}{\delta x_i} = \frac{\delta - \sum_{i=1}^{C} y_i * \ln(s_i)}{\delta s_i} \frac{\delta s_i}{\delta x_i} = \frac{\delta \sum_{i=1}^{C} y_i * \ln(s_i)}{\delta \ln(s_i)} \frac{1}{s_i} s_i * (1 - s_i) = \sum_{i=1}^{C} (y_i)(1 - s_i) = 1 - s_i$$

נזכרתי ששכחתי להוסיף bias לרשת שלי.

שינוי הנגזרת של CrossEntrophyLoss לא כל כך עזר אז אחזור לקודם

יש שתי דרכים שאני יכול להתמודד עם הbias:

1. להוסיף אותו כnode לעץ הווקטורים ברשת

2. לעשות אותו אוטומטי כחלק מהשכבה הלינארית

ננסה את הראשון כי הוא לא דורש שיכתוב מוחלט של הקוד קודם.

<u>לאחר כמה ימים:</u>

מספר ניסיונות הבאים די נכשלו. כנראה שעקב סיבות אחרות מהנגזרות אבל לא הצלחתי למצוא אותם בהתחלה. לאחר מכן גיליתי שעשיתי shuffle רק לX אז הוספתי את הY גם אבל זה לא פתר כי מסתבר ש numpy.random.shuffle משנה את סדר הפיצ'רים בדוגמאות גם ואז התבנית במידע נאבדת

אבל לפני שהבנתי את זה החלטתי לשפר את הנגזרות. התחלתי מלעבור שוב על הנגזרות ולהחליף את התוכנה לדוגמה יחידה במקום batch.

לבסוף סיימתי עם השלבים הבאים:

<u>בשכבה הלינארית:</u>

הנגזרת הלינארית ביחס למשקולות היא עדיין X עצמו אבל הדרך חישוב יותר יעילה, ומסתבר שבחישוב הקודם בנגזרת לX העברתי נגזרות לא נכונות כי חתכתי את הbias בצד הלא נכון.

מה שאני עושה בנגזרת של W זה להכפיל מכפלה חיצונית של X בנגזרת שחלחלה מקצה הרשת, ואז מוסיף רגולריזציה ע"י הוספה של w*2

את שתי אילו תרגמתי למימד הbatch:

את X הופכים לעמודה של עמודות ואת הנגזרת שחלחלה הופכים לעמודה של שורות כך מכפלה סקלרית ביניהם תיצור מכפלה חיצונית בין כל דוגמה של X לנגזרת שחילחלה מקצה הרשת בדוגמה זו.

בW חותכים את הbias מהקצה שזה המקום הנכון ואז עושים מכפלה סקלרית עם הנגזרת שחלחלה כאשר הנגזרת בצורת עמודה ואז המשקולות כפול עמודה יצור שורה באורך הקלט של השכבה

<u>בשכבת הsoftmax:</u>

לפי המאמר הזה: https://towardsdatascience.com/derivative-of-the-softmax-function-and-the-categorical-cross-entropy-loss-ffceefc081d1

ניתן לפשט את הנגזרת של softmax ו- cross entrophy וכך להימנע מהמטריצה:

כש $i=j$:

$$\frac{\delta \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}}{\delta x_i} = f(x_i)\cdot(1-f(x_i))$$

כש $i \neq j$ (כאן הייתה לי טעות בחישוב שעכשיו ראיתי):

$$\frac{\delta \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}}{\delta x_j} = \frac{\delta \frac{e^{x_i}}{\sum_{j=1}^{d} e^{x_j}}}{\delta e^{x_j}} \cdot e^{x_j} = \frac{-1 * e^{x_i}}{(\sum_{j=1}^{d} e^{x_j})^2} \cdot e^{x_j} = \frac{-e^{x_i} * e^{x_j}}{(\sum_{j=1}^{d} e^{x_j}) * (\sum_{j=1}^{d} e^{x_j})} = -f(x_i) * f(x_j) = f(x_i)(0 - f(x_j))$$

ואז בכל מקרה ניתן לרשום את הביטוי כאותה פונקציה שבה מציבים 0 או 1 אם $i = j$ נסמן את זה כך:

$$\{i=j\} = \begin{matrix} 0 & if & i \neq j \\ 1 & if & i = j \end{matrix}$$

ואז הגזרת של s_i כלשהו היא וקטור של נגזרות לכל x_j שבו כל איבר j נראה כך:

$$s_i(\{i=j\} - s_j)$$

נחשב נגזרת של CrossEntropy ונשלב בה את הנגזרת של Softmax (נסמן ב- s_i את תוצאת הsoftmax
ונחשב נגזרת לקלט של הsoftmax x_i

$$f(x_i) = \sum_{i=1}^{d} (-y_i * \ln(s_i))$$

$$\frac{\delta f(x_i)}{\delta x_i} = \frac{\delta \sum_{i=1}^{d} (-y_i * \ln(s_i))}{\delta x_i} = -\sum_{i=1}^{d} \frac{\delta y_i * \ln(s_i)}{\delta x_i} = -\sum_{i=1}^{d} y_i * \frac{\delta \ln(s_i)}{\delta x_i} = -\sum_{i=1}^{d} \frac{y_i}{s_i} \frac{\delta s_i}{\delta x_i} = \ldots$$

$$\ldots = -\sum_{i=1}^{d} \frac{y_i}{s_i} s_i(\{i=j\} - s_j) = -\sum_{i=1}^{d} y_i(\{i=j\} - s_j) = \sum_{i=1}^{d} y_i * s_j - \sum_{i=1}^{d} y_i \{i=j\} = s_j * \sum_{i=1}^{d} (y_i) - y_j = s_j - y_j$$

ומצאנו נגזרת הרבה הרבה יותר פשוטה :)

מכאן מצאתי עוד כמה באגים. הוספתי שמירה של המשקולות הכי טובים על ולידציה ושמירה לקובץ

וזה הסתדר :)

מכאן התחלתי לנסות לאמן את הרשת עם פרמטרים שונים:

נסיון ראשון עקבתי אחרי 3blue1brown עם רשת של 784→16→16→10 אבל הלכתי על רשת של softmax עם cross entrophy עשיתי רגולריזציה של 1e-3 וגם קצב למידה זהה.

batch size נקבע להיות 64

התחלתי להריץ אותו כל פעם 10 אפוקים ואז לתעד דיוק:

נסיון - try1ofcse

מספר אפוק	הפסד על אימון	הפסד על ולידציה	טעות
10	0.60749	0.5675	0.17
20	0.42871	0.39922	0.1225
30	0.37887	0.35668	0.1112
40	0.35166	0.33458	0.1033
50	0.33335	0.32054	0.0964
60	0.32022	0.31068	0.0916

לוג של האימון יצורף כקובץ טקסט

כאן כבר החלתי לעצור ולנסות רשת שנייה עם MSE:

גם פה עקבתי אחרי אותו מבנה 784→16→16→10 עם אותו batch size וlr וגם alpha:

על אותו עיקרון כל פעם 10 אפוקים:

מספר אפוק	הפסד על אימון	הפסד על ולידציה	טעות
10	0.90718	0.90589	0.8865
20	0.89905	0.89887	0.8862
30	0.89789	0.89777	0.8852

פה החלטתי לעצור ולהעלות את קצב הלמידה ל 1e-2 אבל להשאיר את אותה רגולריזציה

נסיון - try1ofmse

מספר אפוק	הפסד על אימון	הפסד על ולידציה	טעות
10	0.8957	0.8953	0.8852
20	0.88796	0.88693	0.7171
30	0.85629	0.853	0.6972
40	0.79096	0.78515	0.6211
50	0.70943	0.70224	0.5
60	0.60269	0.59283	0.3862
70	0.51749	0.5099	0.3282
80	0.47247	0.46645	0.2988
90	0.44727	0.44209	0.2782
100	0.4313	0.42644	0.2554
110	0.41416	0.41889	0.2384
120	0.40716	0.40223	0.2237
130	0.39375	0.38855	0.2034
140	0.37607	0.3706	0.1786
150	0.35235	0.34685	0.1556

160	0.32722	0.32238	0.1367
170	0.30544	0.30188	0.1236
180	0.28948	0.28711	0.1174
190	0.27901	0.2775	0.1149

לוג יצורף בקובץ

ננסה שוב את MSE מהתחלה הפעם נשים רגולריזציה 0 וקצב אימון של 0.1 באפוקים הראשונים

נסיון - try2ofmse

מספר אפוק	הפסד על אימון	הפסד על ולידציה	טעות
10	0.20037	0.18921	0.1021
20	0.13238	0.13716	0.0809
30	0.10985	0.12258	0.0765
40	0.09698	0.11531	0.0745

כאן אקטין את קצב הלמידה פי 10 ואמשיך באימון על אותן משקולות

מהשורה השנייה אתחיל לדווח כל 20 אפוקים

מספר אפוק	הפסד על אימון	הפסד על ולידציה	טעות
50	0.09324	0.11484	0.0733
70	0.09127	0.11403	0.0727
90	0.08957	0.11327	0.0728
110	0.08784	0.11259	0.0725

המודל עדיין לומד אבל ממש לאט. אני עוצר פה והלוג יצורף גם לריצה הזו

בכל ניסיון רשום שם של התיקייה שמכילה קובץ בינארי ולוג אימון או פרמטרים של האימון

לאחר מכן הרצתי את שתי המודלים כמה פעמים בניסיון לשפר ולכל אחד הגעתי לנסיון הכי טוב שהצלחתי:

מודל CrossEntrophy – נמצא בתיקייה softmax_relu_94p:

מבחינת קצב למידה:

10 אפוקים עם 1e-2 , 10 אפוקים עם 1e-3 , 10 אפוקים עם 1e-4

רגולריזציה: 0

מבנה השכבות: [784,200,100,10]

הגיע לטעות של 0.062

הלוג מצורף

במודל MSE נמצא בתיקייה best_attempt_97p:

מבנה שכבות: [784,200,100,10]

קצב למידה: 300 אפוקים עם 1e-2, ואז 100 אפוקים עם 1e-3

alpha = 0

הגיע לטעות של 0.0273

* על מנת לטעון את הקבצים צריך לסמן בהערה את השורה של fit ואז להחליף את f_name לנתיב היחסי של הקובץ הבינארי. ואז להחליף את סוג הרשת בין MSEnet ל-SoftmaxNet בהתאם למודל