تمرین سری اول

یادگیری ژرف

امیرحسین محمدی

۹۹۲۰۱۰۸۱

مسئله ۱. Linear Regression (۱۵ نمره)

مجموعه دادگان $S = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ را در نظر بگیرید به گونه‌ای که $x \in \mathbb{R}^d$ و $y \in \mathbb{R}$ است. برای تخمین برچسب یک نمونه x از رابطه رگرسیون خطی به صورت $\hat{y} = \sum_{j=1}^{d} w_j x_j + b$ و برای آموزش مدل از تابع هزینه SSE به صورت $\mathcal{L}(w, b) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$ استفاده می‌کنیم. حال به سوالات زیر پاسخ دهید.

الف) با استفاده از گرادیان کاهشی رابطه به‌روز رسانی برای وزن‌ها ارائه دهید.

Assumptions:

$$S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$$

$$x \in \mathbb{R}^d$$

$$y \in \mathbb{R}$$

$$\hat{y} = \sum_{j=1}^{d} w_j x_j + b$$

$$\mathcal{L}(w, b) = \frac{1}{2n} \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 \Rightarrow \text{SSE}$$

Assumptions:

$$x = \begin{bmatrix} x_0^1 & \cdots & x_D^1 \\ \vdots & \ddots & \vdots \\ x_0^n & \cdots & x_D^n \end{bmatrix}_{N*D} \qquad w = \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_D \end{bmatrix}_{D*1} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{bmatrix}_{N*1}$$

We assume:

$$y = \sum_j w_j x_j + b$$

We assume, prediction y is from the target t

So: $L(y,t) = \frac{1}{2}(y-t)^2$

We have:

For vector W: $V(w_1, \dots, w_D, b) = \frac{1}{n}\sum_{i=1}^{n} L\left(y^{(i)}, t^{(i)}\right) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)}, t^{(i)}\right)^2 = \frac{1}{2n}\sum_{i=1}^{n}\left(\sum_j w_j x_j^{(i)} + b - t^{(i)}\right)^2$

Calculate $\frac{\partial V}{\partial w_j}$: $\frac{\partial V}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}\left(\sum_{j'} w_{j'} x_{j'}^{(i)} + b - t^{(i)}\right)$ Simplification $\frac{\partial V}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}\left(y^{(i)} - t^{(i)}\right)$

Calculate $\frac{\partial V}{\partial b}$: $\frac{\partial V}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j'} w_{j'} x_{j'}^{(i)} + b - t^{(i)}\right)$ Simplification $\frac{\partial V}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - t^{(i)}\right)$

$$\frac{\partial V}{\partial w} = \begin{pmatrix} \frac{\partial V}{\partial w_1} \\ \frac{\partial V}{\partial w_2} \\ . \\ . \\ . \\ \frac{\partial V}{\partial w_D} \end{pmatrix} \quad w = w - \alpha \frac{\partial v}{\partial w} \quad \text{OR} \quad w_j = w_j - \alpha \frac{\partial v}{\partial w_j}$$

Now for Vectorization:

$$x = \begin{bmatrix} x_0^1, & \cdots & , x_D^1 \\ \vdots & \ddots & \vdots \\ x_0^n, & \cdots & , x_D^n \end{bmatrix}_{N*D} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_D \end{bmatrix}_{D*1} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{bmatrix}_{N*1}$$

$$xw = \begin{bmatrix} x_0^1 w_0 + & \cdots & + x_D^1 w_D \\ \vdots & \ddots & \vdots \\ x_0^n w_0 + & \cdots & + x_D^n w_D \end{bmatrix}_{N*1} - \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{bmatrix}_{N*1} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_N \end{bmatrix}_{N*1} = \text{Error}$$

$$xw = \begin{bmatrix} x_0^1 w_0 + & \cdots & + x_D^1 w_D \\ \vdots & \ddots & \vdots \\ x_0^n w_0 + & \cdots & + x_D^n w_D \end{bmatrix}_{N*1} - \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{bmatrix}_{N*1} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_N \end{bmatrix}_{N*1} = \text{Error}$$

$$\begin{bmatrix} W_1' \\ W_2' \\ . \\ . \\ W_N' \end{bmatrix}_{N*1} = \begin{bmatrix} W_1 \\ W_2 \\ . \\ . \\ W_D \end{bmatrix}_{D*1} - \left(\alpha * \frac{1}{N} \begin{bmatrix} E_1 x_0^1 + & \cdots & + E_N w_D \\ \vdots & \ddots & \vdots \\ E_1 w_0 + & \cdots & + E_N w_D \end{bmatrix}_{D*1} \right)$$

$$\begin{bmatrix} W_1' \\ W_2' \\ . \\ . \\ W_N' \end{bmatrix}_{N*1} = \begin{bmatrix} W_1 \\ W_2 \\ . \\ . \\ W_N \end{bmatrix}_{N*1} - \left(\alpha * \frac{1}{N} \left[[E_1, \ldots, E_N] * \begin{bmatrix} x_0^1 \\ x_0^2 \\ . \\ . \\ x_0^N \end{bmatrix} \right]_{N*1} \right)$$

$$w' = w - \alpha * \frac{1}{m} \left[(X * w - y)' X_j \right]'$$

به صورت کلی داریم:

$$w' = w - \alpha * \frac{1}{m} \left[(X * w - y)' * X \right]'$$

Or

$$w' = w - \alpha * \frac{1}{m} \left[X' * (X * w - y) \right]$$

ب) حال رابطه محاسبه وزن‌ها را به صورت فرم بسته ارائه دهید.

Another way:

One way to compute the minimum of a function is to set the partial derivatives to zero. Recall from single variable calculus that (assuming a function is differentiable) the minimum x^*. of a function f has the property that the derivative $\frac{df}{dx}$ is zero at x = x^*. An analogous result holds in the multivariate case: if f is differentiable, then all of the partial derivatives $\frac{\partial f}{\partial xi}$ are zero at the minimum point.

❖ We assume weight w_0 acts as a bias:

$$\frac{\partial V}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}\left(\sum_{j'=1}^{D} w_{j'}x_{j'}^{(i)} - t^{(i)}\right) = 0$$

$$\frac{\partial V}{\partial w_j} = \frac{1}{n}\sum_{j'=1}^{D}\left(\sum_{i=1}^{n} x_j^{(i)}x_{j'}^{(i)}\right)w_{j'} - \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}t^{(i)} = 0$$

Now for Vectorization:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

روش اول

- $L(w,b) = \sum_{i=0}^{n}(y^i - w^T x^i)^2$ ⟵ Bias را به عنوان w_0 فرض می کنیم
- $L(w) = ||y - Xw||$
- $\overline{\overline{\nabla}}_w\, L(w) = -2X^T(y - Xw)$
- $\overline{\overline{\nabla}}_w\, L(w) = 0$
- $X^T X w = X^T y$
- $w = (X^T X)^{-1} X^T y$

روش دوم

- $L(w,b) = \sum_{i=0}^{n}(y^i - w^T x^i)^2$ ⟵ Bias را به عنوان w_0 فرض می کنیم

بازنویسی عبارت بالا به فرم دیگر

- $Xw = y$

جهت معکوس کردن X، ضرب طرفین در X^T تا ماتریس مربعی شود

- $X^T X w = X^T y$
- $w = (X^T X)^{-1} X^T y$

ج) با فرض اینکه گرادیان کاهشی بعد از m به‌روز رسانی به پاسخ بهینه می‌رسد، این دو روش را از نظر مرتبه زمانی با یک‌دیگر مقایسه کنید.

Closed form	Gradient Descent
$O(n^3)$	$O(kn^2)$

- مشکل معادله ی بسته به این شکل است که ما اگر بخواهیم وارون ماتریس را محاسبه کنیم به اندازه ی $o(n^3)$ پیچیدگی زمانی داریم بنابراین از مهمترین مشکلات استفاده از این رابطه ی می توان به سربار محاسباتی بالای آن اشاره کرد. می دانیم مرتبه ی زمانی ضرب دو ماتریس $O(n^3)$ و اگر n از یک حد معین مثلا 10000 باشند این زمان اجرا بسیار قابل ملاحظه بوده و به نوعی گلوگاه محسوب می شود و با توجه به اینکه عمده ی مسائل از حجم قابل توجهی ویژگی برخودار هستند بنابراین استفاده از رابطه ی مستقیم به عنوان کی مزین محسوب نمی شود. از جمله راه حل هایی که برای رفع این مشکل ارائه می شود کاهش ابعاد و ویژگی هاست که خود این مسئله می تواند و به کاهش دقت منجر شود. از دیگر راه حل های ارائه شده برای رفع این مشکل استفاده از گرادیان کاهشی برای محاسبه ی بهترین w برای مسئله ی مورد نظر است.
- از دیگر مشکلات این روش، می توان به معکوس ناپذیری X^TX اشاره کرد که دلیل آن افزونگی ویژگی ها (وابستگی خطی بین نها) و زیاد بودن ویژگی هاست. که از راه حل های آن می توان به استفاده از ماتریس شبه معکوس اشار کرد.

```
w= pinv(X.T @ X) @ X.T @ y
```

از دیگر راه حل ها برای رفع مشکل معکوس ناپذیری می توان به حذف برخی از ویژگی ها با استفاده از تنظیم اشاره کرد.

بررسی دقیق پیچِدگی زمانی معادله ی بسته

- ماتریس ورودی X همانطور که در فرضیات مسئله اشاره شده ابعاد N*D دارد و ترانهاده ماتریس X ابعاد D*N دارد بنابراین حاصل ضرب ماتریس X^TX دارای پیچیدگی زمانی $O(D^2N)$ است.

- محاسبه ی وارون ماتریس X^TX دارای پیچیدگی زمانی $O(N^3)$ است.

- محاسبه ی ضرب ماتریس X^Ty دارای پیچیدگی زمانی $O(DN)$ است.

- محاسبه ی ضرب دو ماتریس X^TX^{-1} و X^Ty دارای پیچیدگی زمانی $O(D^2)$ است.

بنابراین به صورت کلی داریم:

$$O(D^2N)+ O(N^3) +O(DN)+ O(D^2)$$

بررسی دقیق پیچیدگی زمانی روش گرادیان کاهشی

- محاسبه ی ضرب ماتریس $X^T y$ دارای پیچیدگی زمانی $O(DN)$ است.
- محاسبه ی ضرب ماتریس $\alpha X^T y$ دارای پیچیدگی زمانی $O(DN)$ است.
- محاسبه ی ضرب ماتریس X^TX دارای پیچیدگی زمانی $O(D^2N)$ است.
- محاسبه ی ضرب ماتریس αX^TX دارای پیچیدگی زمانی $O(D^2N)$ است.
- محاسبه ی ضرب ماتریس جمع ها و تفریق ها دارای پیچیدگی زمانی $O(D)$ است.
- همچنین ما k تکرار یا iteration داریم.

بنابراین به صورت کلی داریم:

$$O(kD^2)+O(D^2N) = \mathrm{O((k+N)D^2)}$$

همچنین لازم به ذکر است در صورت اپتیماز کردن الگوریتم و استفاده از روش هایی پیچیدگی زمانی گرادیان کاهشی به صورت زیرقابل تغییر است: $O(NDK)$

مسئله ۲. Activation Function (۱۰ نمره)

به سوالات زیر پاسخ دهید.

الف) ابتدا مشکل vanishing gradient را توضیح دهید. سپس توابع فعالسازی ReLU و sigmoid با بررسی مشتق از این نظر مقایسه کنید.

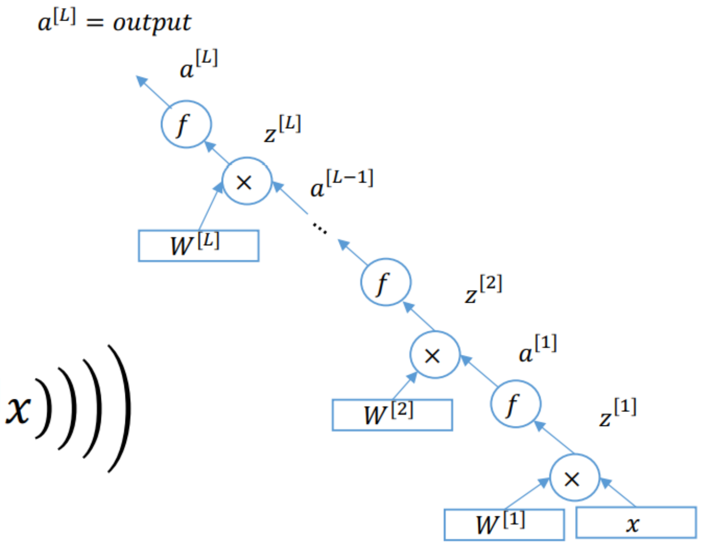
ب) ابتدا مقداردهی Xavier را توضیح دهید و سپس بررسی کنید که چگونه به مشکل محو شدن کمک می‌کند.

ج) فرایند آموزش یک شبکه عصبی با تابع فعال‌سازی sigmoid را در صورتی که مقداردهی اولیه وزن‌ها بزرگ است، بررسی کنید.

در شبکه های عصبی می دانیم که برای به روز رسانی وزن ها باید مشتق تابع loss را نسبت به ورودی اولیه محاسبه کنیم، بنابراین طبق طبق الگوریتم Back Propagation ما به یک زنجیره متوالی از ضرب مشتقات در یک دیگر رو به رو خواهیم بود. اط طرفی می دانیم که در شبکه های عمیق این مسئله به صورت شدید تری وجود دارد (چون تعداد لایه ها بیشتر است بنابراین برای محاسبه ی مشتق تابع loss نسبت به ورودی با یک زنجیره بزرگتری مواجه هستیم):

ساختار شبکه عصبی

$$
\begin{aligned}
Output &= a^{[L]} \\
&= f\left(z^{[L]}\right) \\
&= f\left(W^{[L]}a^{[L-1]}\right) \\
&= f\left(W^{[L]}f(W^{[L-1]}a^{[L-2]}\right) \\
&= f\left(W^{[L]}f\left(W^{[L-1]} \dots f\left(W^{[2]}f(W^{[1]}x)\right)\right)\right)
\end{aligned}
$$



$$
Loss(x) = E\left(f^{[L]}\left(W^{[L]}f^{[L-1]}\left(W^{[L-1]}f^{[L-2]}(\dots W^{[1]}x)\right)\right)\right)
$$

$$Loss(x) = E\left(f^{[L]}\left(W^{[L]}f^{[L-1]}\left(W^{[L-1]}f^{[L-2]}(\dots W^{[1]}x)\right)\right)\right)$$

$$\nabla_{f^{[l]}}Loss = \nabla_{f^{[L]}}Loss.\nabla f^{[L]}.W^{[L]}.\nabla f^{[L-1]}.W^{[L-1]}\dots\nabla f^{[l+1]}.W^{[l+1]}$$

همانطور که ما از رابطه ی بالا مشخص است ما ما حاصل ضرب یک سری گرادیان مواجه هستیم، بنابراین در اثر ضرب متوالی بردارهای وزن در بردارهای ویژه ای که مقدار بزرگ تر از یک دارند (در اثر لایه های زیاد و شبکه های عمیق) مقادیر وزن ها به صورت خیلی زیاد تغییر کند explosion gradient رخ می دهد. به همین ترتیب اثر ضرب متوالی بردارهای وزن در بردارهای ویژه ای که مقدار کوچکتر از یک دارند مقادیر وزن ها به صورت خیلی ناچیز تغییر می کند و Vanishing gradient رخ می دهد. همچنین می دانیم که حاصل توابع فعالیت Tanh و sigmoid و ReLU در شبکه های عصبی معمولی و بازگشتی مقداری کمتر از یک دارند. مثلا تابع سیگموید Sigmoid، مقادیر ورودی با مقیاس بزرگ را در یک بازه‌ی کوچک میان صفر و 1 قرار می‌دهند؛ بنابراین زمانی‌که یک تغییر بسیار بزرگ در مقدار ورودی تابع اتفاق می‌افتد، خروجی تابع تنها مقدار کمی تغییر می‌کند؛ این یعنی مقدار مشتق آن خیلی کوچک می‌شود.

در شبکه‌های کم‌عمق که تعداد لایه‌های آن‌ها کم است استفاده از توابع یاده شده و مشکل Vanishing Gradient خیلی اهمیت ندارد، اما زمانی‌که تعداد لایه‌های شبکه زیاد باشد، این مشکل به این می‌انجامد که مقدار گرادیان در حدی کوچک شود که امکان آموزش شبکه وجود نداشته باشد.

درواقع گرادیان شبکه با فرایند انتشار روبه‌عقب محاسبه می‌شود. در واقع در فرایند انتشار روبه‌عقب مقدار مشتق‌های شبکه با حرکت از سمت لایه‌ی خروجی به سمت لایه‌ی ورودی به دست می‌آید(درواقع یک مشتق زنجیره‌ای از لایه‌ی آخر به‌سمت لایه‌ی اول انجام می‌شود).

حال تصور کنیم که nلایه از تابعی مانند تابع سیگموید استفاده می‌کنند؛ این یعنی n مقدار مشتق کوچک در هم ضرب می‌شوند (مشتق زنجیره‌ای). این امر به این می‌انجامد که مقدار گرادیان در طول انتشار روبه‌عقب به‌صورت نمایی کاهش یابد.

مقدار گرادیان کوچک به‌معنای این است که وزن‌های لایه‌های اول به‌درستی به‌روزرسانی نمی‌شوند. از آنجا که لایه‌های اول در شناسایی عناصر اصلی داده ورودی خیلی اهمیت دارند، این موضوع درنهایت کاری می‌کند که شبکه‌ی خروجی قابل‌قبولی را ارائه نکند.

ب) ابتدا مقداردهی Xavier را توضیح دهید و سپس بررسی کنید که چگونه به مشکل محو شدن کمک می‌کند.

هدف Xavier Initialization این است که وزن ها را به گونه ای مقداردهی کند که واریانس خروجی توابع فعال ساز در هر لایه یکسان باشد. این راه حل مشکلاتی همچون Vanishing gradient و Exploding gradient کمک می کند.

- ❖ در واقع این مسئله از اینجا شکل می گیرد که اگر ما خروجی تابع فعال ساز یک تابع را در نظر بگیریم خواهیم داشت:

$$\mathrm{Var}\left(z_j^{(l)}\right) = \mathrm{Var}\left(\sum_{j=1}^{m_{l-1}} W_{jk}^{(l)} a_k^{(l-1)}\right)$$

$$= \sum_{j=1}^{m^{(l-1)}} \mathrm{Var}\left[W_{jk}^{(l)} a_k^{(l-1)}\right] = \sum_{i=1}^{m^{(l-1)}} \mathrm{Var}\left[W_{jk}^{(l)}\right] \mathrm{Var}\left[a_k^{(l-1)}\right]$$

$$= \sum_{j=1}^{m^{(l-1)}} \mathrm{Var}\left[W^{(l)}\right] \mathrm{Var}\left[a^{(l-1)}\right] = m^{(l-1)}\, \mathrm{Var}\left[W^{(l)}\right] \mathrm{Var}\left[a^{(l-1)}\right]$$

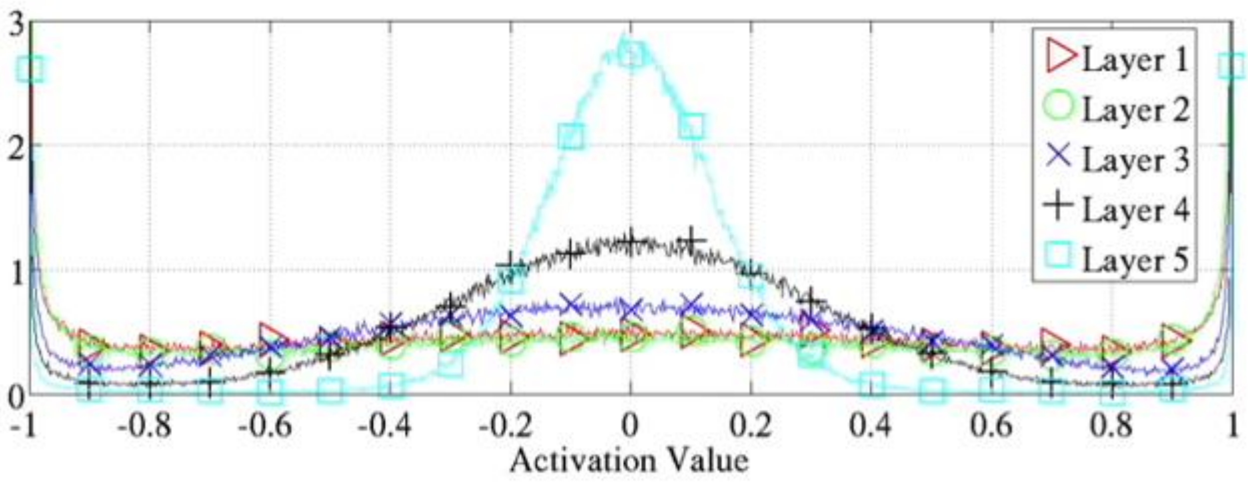
بنابراین روش Xavier به دنبال راهی برای جلو گیری از این اتفاق است.

همانطور که میبینیم وارایاس هر لایه به صورت خطی با تعداد ویزگی های موجود در حالت قبل ضرب می شود و هر چقدر این ضریب بزرگتر باشد وارایانس افزایش می یابد.

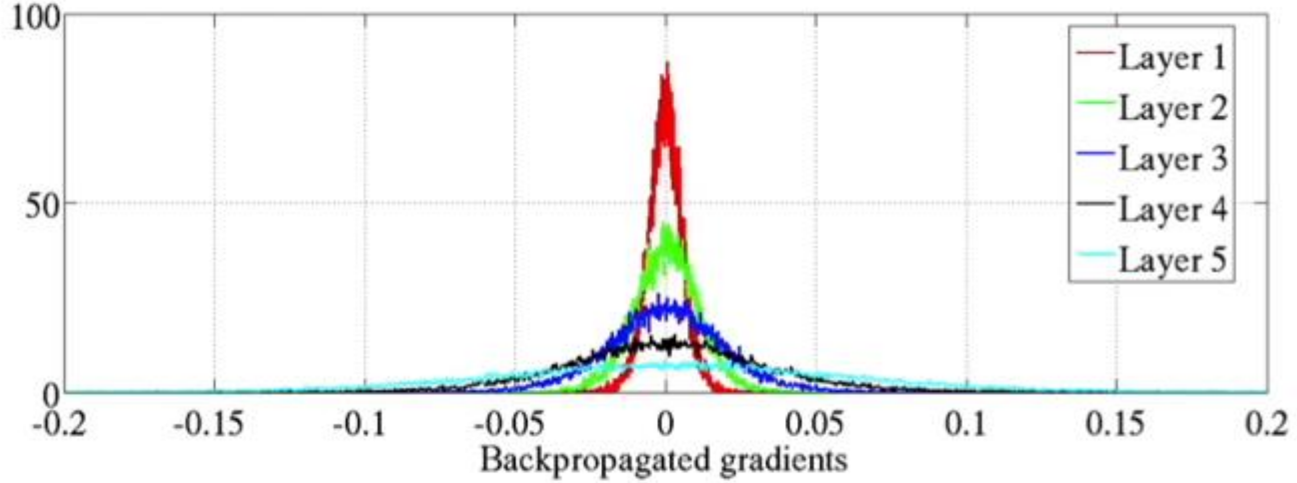
بنابراین بر اساس مقدار دهی Xavier ما در ابتدا بعد از اینکه مقدار وزنها بر اساس یه توزیع مقدار دهی کردیم (مثلا یکنواخت یا گوسی) سپس وزنهای هر لایه را اسکیل می کنیم به صورت زیر:

$$W^{(i)} = W^{(i)} * \frac{1}{\sqrt{n^{l-1}}}$$

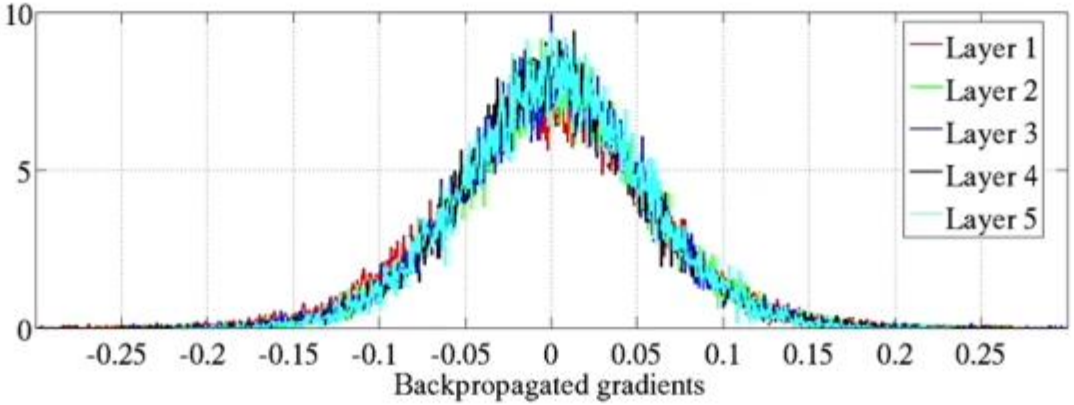
در واقع ما وزن هر لایه را با استفاده از معکوس ریشه ی دوم تعداد ویژگی های لایه ی قبل (برای لایه ی اول ویژگی های ورودها را در نظر می گیریم) اسکیل می کنیم. این ضریب به نوعی ازروند خطی که در اسلاید قبل به آن پرداخته شد جلوگیری می کند.



همانطور که در شکل مقابل مشاهده می کنیم از لایه ی یک تا ۵ به به دلیل ضرب تعداد ویژگی های لایه ی قبل در لایه فعلی (همانطور که در اسلاید قبل به آن پرداخته شد) توزیع از حالت یکنواخت یا گوسی در (حالت اولیه) به حول نقطه ی صفر متمرکز می شود (به تدریج در لایه ها بعدی تا لایه پنج در واقع اسکیل می شوند)می یابد.



بنابراین مسئله با توجه به مسئله ی پرداخته شده در اسلاید قبلی ما دچار مشکل vanishing در لایه های ابتدایی می شویم. در واقع به دلیل کوچک شدن توزیع (به خاطر مسئله اسکیل شدن یا همان واریانس) مقادیر گرادیان لایه ی پنچم همانور که در کشل مقابل میبینیم بسیار کوچک می شود و این گردیان تا لایه ی اول که مهم ترین بخش به نوعی برای آپدیت وزنها هست دچار محو شدگی می شود.

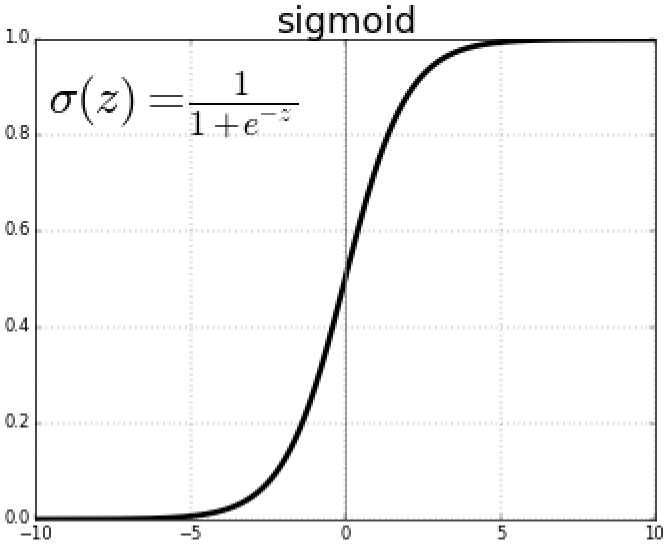


اما با استفاده از مقدارهی xavier پس از اینکه ما مقدار وزن ها را با یک تابع توزیع احتمال مقداردهی کردیم با استفاده از ضریبی که معرفی کردیم در واقع از این اسکیل شدن واریانس جلوگیری می کنیم و همانطور که در شکل مقابل می بینیم گردایان تا حد خوبی بین تمامی لایه های یکسان است و از مشکل vanishing جلوگیری می شود.

ج) فرایند آموزش یک شبکه عصبی با تابع فعال‌سازی sigmoid را در صورتی که مقداردهی اولیه وزن‌ها بزرگ است، بررسی کنید.

❖ فرض کنیم که رابطه ی زیر بر قرار است:

$$z = w_i act_i + b_i$$



در رباطه ی بالا w_i وزن های لایه ی بعدی است و act_i خروجی تابع فعال ساز لایه ی قبل است. زمانی که مقدار w_i بزرگ لست آنگاه z به نقاط صفر یا یک میل می کند. همانطور که میدانیم این قسمت ها در تابع سیگموید قسمت اشباع (saturation) است و مقدار گرادیان در این نقطه بسیار نزدیک به صفر است. بنابراین زمانی که میزان وزن ها بسیار بزرگ باشه عملا فرِآیند یادگیری و آپدیت وزن ها به صورت بسیار کند صورت می گیرد. (و همچنین می تواند مشکلاتی مثل Vanishing به وجود بیاورد)

مسئله ۳. Regularization & Optimization (۲۰ نمره)

به سوالات زیر پاسخ دهید.

الف) چرا منظم‌ساز L_2 معمولا روی Bias شبکه اعمال نمی‌شود؟ همچنین توضیح دهید چرا منظم‌ساز L_1 منجر به صفر شدن برخی از وزن‌ها می‌شود؟

ب) مسئله رگرسیون خطی بر روی n داده با تابع هزینه $\mathcal{L}(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - {x^{(i)}}^{T} w\right)^2$ را در نظر بگیرید. اثبات کنید اضافه کردن نویز از توزیع $\mathcal{N}(0, \sigma^2 I)$ به داده‌های ورودی معادل استفاده از منظم‌ساز L_2 در تابع هزینه است.

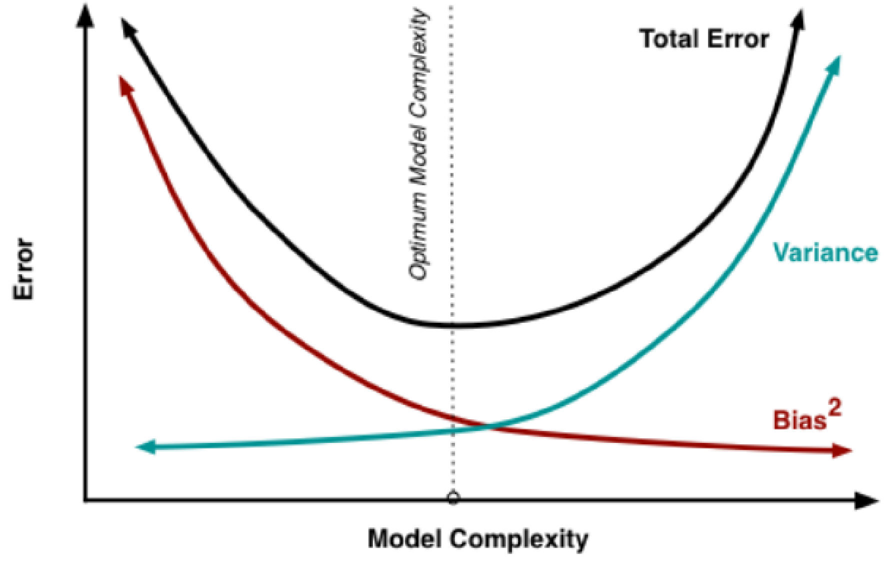
ج) توضیح دهید که Batch Normalization چگونه فرایند آموزش را سرعت می‌بخشد. همچنین توضیح دهید زمانی که سایز batch کوچک باشد، استفاده از Batch Normalization چه تاثیری در فرایند آموزش دارد.

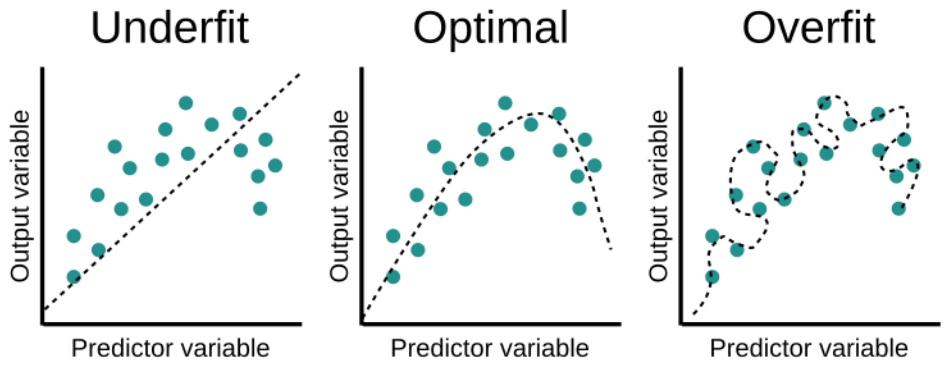
د) رابطه گشتاور اول بهینه‌ساز آدام را به صورت $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_t)$ در نظر بگیرید. نشان دهید چرا مقادیر m_t گرایش به صفر دارند و چرا مقدار $\hat{m}_t$ که به صورت $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ محاسبه می‌شود، با این مشکل روبرو نمی‌شود.

الف) چرا منظم‌ساز L_2 معمولا روی Bias شبکه اعمال نمی‌شود؟ همچنین توضیح دهید چرا منظم‌ساز L_1 منجر به صفر شدن برخی از وزن‌ها می‌شود؟

همانطور که می دانیم زمانی که مدل ما overfit می شود در واقع واریانس مدل ما افزایش یافته و بایاس کاهش یافته هست و زمانی که مدل ما underfit می شود در واقع ما در جهت افزایش بایاس پیش رفته ایم و در جهت کاهش واریانس (همانطور که در شکل مقابل می بینیم). زمانی که مدل ما overfit می شود ما با استفاده از جملات regularization می توانیم از وقوع overfit تا حد زیادی جلوگیری کنیم(به دلیل اینکه overfitting به دلیل افزایش واریانس و کاهش بایاس رخ می دهد) بنابراین ما باید واریانس را کاهش دهیم و برای این کار از منظم ساز مختلف مثل L1وL2 استفاده می کنیم. چون Bias هیچ تاثیر اساسی و چشم گیری در افزایش واریانس ندارد (تاثیر به سزایی در ایجاد واریانس ندارد) و پارامتر L2 Regularization صرفا واریانس را کاهش می دهد، برای همین معمولا از پارامتر L2 Regularization نسبت به Bias صرف نظر می کنیم.

$$J(\boldsymbol{w}) = \sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}^T\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\right)^2 + \lambda\boldsymbol{w}^T\boldsymbol{w}$$



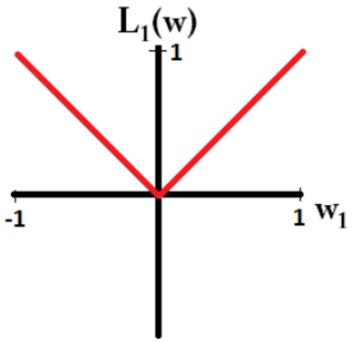


❖ بردار w مقابل را در نظر بگیریم:

$$(w_1, w_2, \dots, w_n)$$

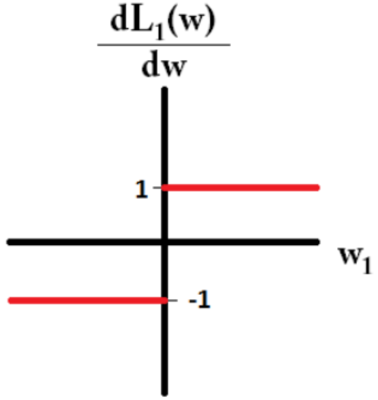
❖ برای منظم ساز L1 داریم:

$$L_1(w) = \Sigma_i |w_i|$$

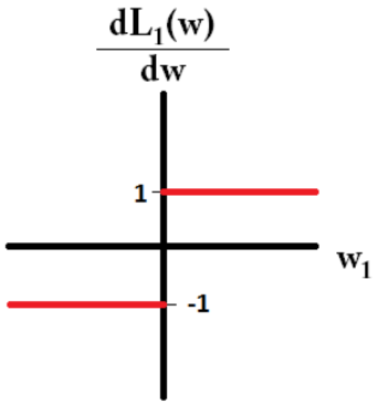


❖ زمانی که ما می خواهیم تابع هزینه را بهنیه کنیم باید از الگوریتم گرادیان کاهش استفاده کنیم همچنین می دانیم که با اضاقه کردن جمله منظم ساز به تابع هزینه، در واقع در عملیات gradient descent جمله ی منظم ساز هم شرکت می کند. برای گرادیان منظم ساز L1 داریم:

$$\frac{dL_1(w)}{dw} = sign(w), \text{ where } sign(w) = \left(\frac{w_1}{|w_1|}, \frac{w_2}{|w_2|}, \dots, \frac{w_m}{|w_m|}\right)$$

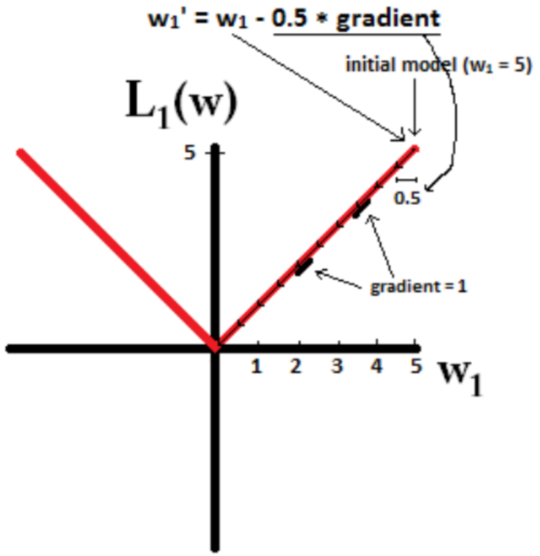


همانطور که از گرادیان منظم ساز L1 نسبت به w مشخص است (مطابق شکل مقابل) گرادیان در نقط مثبت یک است و در نقاط منفی منفی یک است و در نقطه ی W=0 صفر است و از آنجا که ما عکس جهت گرادیان حرکت می کنیم خواهیم داشت:



فرض کنیم که مقدار w1 در ابتدا τ باشد برای آپدیت وزنها با استفاده از گرادیان کاهش داریم:

$$w_1 = w_1 - \alpha * \frac{dL_1(w)}{dw} = \tau - \alpha$$



مشخص است که پس از چند تکرار بلافاصله مقدار w1 به صفر می رسد و این مسئله همانطور که مشخص است برای منظم ساز L1 به وضوح رخ می دهد و بسیاری از وزنها به صفر می کند یا نزدیک به صفر می شود.

ج) توضیح دهید که Batch Normalization چگونه فرایند آموزش را سرعت می‌بخشد. همچنین توضیح دهید زمانی که سایز batch کوچک باشد، استفاده از Batch Normalization چه تاثیری در فرایند آموزش دارد.

❖ یکی از مشکلاتی که همواره در فرآیند آموزش شبکه های عصبی مورد توجه قرار می گیرد وقوع مشکل vanishing یا exploding است و این مسئله به این خاطر است که اسکیل داده ها ممکن از نه تنها در ابتدا متفاوت باشد بلکه در اثر لایه های متوالی هم اسکیل داده ها و هم وزن ها متفاوت می شود. بنابراین روش Batch Normalization با نگه داشتن اسکیل داده ها در یک range مشخص از وقوع این مشکلات به صورت چشم گیری جلوگیری می کند و این عمر در روند یادگیری شبکه تسریع می بخشد.

❖ علاوه بر اینها Batch Normalization ورودی های هر یک تابع های فعال ساز را به گونه ای تغییر می دهد که از افتادن در نقاط اشباع شده تا حد زیادی جلوگیری شود و این مسئله خود به افزایش سرعت شبکه کمک می کند.

❖ به دلیل اینکه توزیع ویژگی ها در ابتدا ممکن است متفاوت باشد در واقع Batch Normalization توزیع را یکسان می کند و همچنین تابع خطای مورد استفاده ما سطح هموار تری به خود می گیرد و این مسئله از افتادن در نقاط بهینه محلی تا حد زیادی جلوگیری می کند و ما را به سرعت به نقطه ی بهینه نزدیک می کند و این امر در تسریع فرآیند آموزش تاثیر به سزایی دارد.

❖ مسئله ی دیگر internal covariate shift است که این مسئله خود به تسریع بخشی فرآیند آموزش شکه کمک می کند.

❖ Batch Normalization وابستگی گرادیان ها به مقیاس پارامترها یا مقادیر اولیه آنها را کاهش می دهد، که این مسئله به ما کمک می کند که بتوانیم از نرخ یادگیری (learning rate) بزرگتری استفاده کنیم بدون اینکه نگران واگرایی باشیم.

$$\mu_B = \frac{1}{K}\sum_{k=1}^{K} x_k \qquad \sigma_B^2 = \frac{1}{K}\sum_{k=1}^{K}(x_k - \mu_B)^2$$

$$\hat{x}_k = \frac{x_k - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$\hat{y}_k = \gamma \hat{x}_k + \beta = BN_{\gamma,\beta}(x_k)$$

$$\mu_B = \frac{1}{K}\sum_{k=1}^{K} x_k \qquad \sigma_B^2 = \frac{1}{K}\sum_{k=1}^{K}(x_k - \mu_B)^2$$

$$\hat{x}_k = \frac{x_k - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

همانطور که در رابطه ی batch normalization می بینیم برای نرمال کردن ویژگی ها ما به میانگین و واریانس آنها واسته هستیم. بنابراین وقتی اندازه ی batch کوچک می شود در واقع میانگین و واریانس بدست آمده از تعداد کمی از داده ها بدست آمده اند و قابل تعمیم نیستند بنابراین batch normalization نمی تواند نقش خود را به خوبی بازی کند و دقت نسبتا کم می شود.

ب) مسئله رگرسیون خطی بر روی n داده با تابع هزینه $\mathcal{L}(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - x^{(i)'} w\right)$ را در نظر بگیرید. اثبات کنید اضافه کردن نویز از توزیع $\mathcal{N}(0,\sigma^2 I)$ به داده‌های ورودی معادل استفاده از منظم‌ساز L_2 در تابع هزینه است.

❖ برای تابع لاس با استفاده از جمله ی منظم ساز داریم:

$$L = \underbrace{\sum_{n=1}^{N}\left(y^{(n)} - f_w\left(x^{(n)}\right)\right)^2}_{Loss\ function} + \lambda \underbrace{\sum_{i=1}^{k} w_i^2}_{L_2\ Regularization}$$

❖ multivariate Gaussian داریم:

$$N(x;\mu;\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$

بر اساس رابطه ی بیز داریم:

➢ $p(w|D) = \frac{p(D|W)p(w)}{p(D)}$ ⇒ ➢ $p(w|D) = p(D|w)p(w)$ ⇒ $(\prod_{n}^{N} N(y^{(n)}; f_w\left(x^{(n)}\right), \sigma_y^2))N(w; 0, \sigma^2 I)$

$$\left(\prod_{n}^{N} N\left(y^{(n)}; f_w\left(x^{(n)}\right), \sigma_y^2\right)\right) N(w; 0, \sigma^2 I)$$

می توانیم گوسی های چند بعدی را جدا سازی کنیم زیرا کوواریانس ما یک ماتریس همانی است.

$$\left(\prod_{n}^{N} N\left(y^{(n)}; f_w\left(x^{(n)}\right), \sigma_y^2\right)\right) \prod_{I=1}^{K} N(w; 0, \sigma^2)$$

منفی log احتمال بدست آمده از قاعده ی بیز را محاسبه می کنیم

$$-\log\big(p(w|D)\big) = -\sum_{n=1}^{N} \log\left(N\left(y^{(n)}; f_w\left(x^{(n)}\right), \sigma_y^2\right)\right) - \sum_{i=1}^{k} \log(N(w; 0, \sigma^2)) + C$$

ساده سازی

$$-\log\big(p(w|D)\big) = \frac{1}{2\sigma_y^2} \sum_{n=1}^{N} \left(y^{(n)} - f_w\left(x^{(n)}\right)\right)^2 + \frac{1}{2\sigma_y^2} \sum_{i=1}^{k} w_i^2 + C$$

$$L = \sum_{n=1}^{N} \left(y^{(n)} - f_w\left(x^{(n)}\right)\right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

د) رابطه گشتاور اول بهینه‌ساز آدام را به صورت $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_t)$ در نظر بگیرید. نشان دهید چرا مقادیر m_t گرایش به صفر دارند و چرا مقدار $\hat{m}_t$ که به صورت $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ محاسبه می‌شود، با این مشکل روبرو نمی‌شود.

برای بهینه ساز آدام داریم:

momentum

$$\begin{cases} m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_t) \\ v_t = \beta_2 v_{t-1} + (1-\beta_2)\nabla_\theta J(\theta_t)^2 \\ w_t = w_{t-1} - \eta * \dfrac{m'_t}{\sqrt{v'} + \varepsilon} \end{cases}$$

RMSProb

$$\Rightarrow$$

$$m'_t = \frac{m}{1-\beta_1^t}$$

$$v'_t = \frac{v}{1-\beta_2^t}$$

برای رابطه ی گشتاور اول داریم:

در ابتدا m را صفر در نظر میگیریم

- $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_t)$ $\Rightarrow$
- $m_0 = 0$ $\Rightarrow$
- $m_1 = \beta_1 m_0 + (1-\beta_1)\nabla_\theta J(\theta_t) = (1-\beta_1)\nabla_\theta J(\theta_t)$

$\Rightarrow$
- $m_2 = \beta_1 m_1 + (1-\beta_1)\nabla_\theta J(\theta_t) = \beta_1(1-\beta_1)\nabla_\theta J(\theta_t) + (1-\beta_1)\nabla_\theta J(\theta_t)$

$\Rightarrow$
- $m_3 = \beta_1 m_2 + (1-\beta_1)\nabla_\theta J(\theta_t) = \beta_1(\beta_1(1-\beta_1)\nabla_\theta J(\theta_t) + (1-\beta_1)\nabla_\theta J(\theta_t)) + (1-\beta_1)\nabla_\theta J(\theta_t)$ $\Rightarrow$

$\Rightarrow$
- $m_3 = \beta_1^2(1-\beta_1)\nabla_\theta J(\theta_t) + (1-\beta_1)\nabla_\theta J(\theta_t)) + (1-\beta_1)\nabla_\theta J(\theta_t)$ $\Rightarrow$
- $m_t = (1-\beta_1)\sum_{t=0}^{t}\beta_1^{t-1}\nabla_\theta J(\theta_t)$

$$E[m_t] = E[= (1-\beta_1)\sum_{t=0}^{t} \beta_1^{t-1}\,\nabla_\theta J(\theta_t)]$$

$$= E[\nabla_\theta J(\theta_t)](1-\beta_1)\sum_{t=0}^{t}\beta_1^{t-1} + c = E[\nabla_\theta J(\theta_t)](1-\beta_1^t) + c$$

- $m_t = (1-\beta_1)\sum_{t=0}^{t}\beta_1^{t-1}\,\nabla_\theta J(\theta_t)$

- به صورت معمول مقدار β را یک عدد بین صفر و یک در نظر می گیرند به صورت معمول ۰.۹ و ۰.۹۹ به همین دلیل همانطور که از رابطه ی مقابل مشخص است هر چقدر که آموزش می دهیم مقدار m_t به صفر میل می کند. در واقع β که مقدار صفر و یک دارد به توان می رسد و همچنان بین صفر و یک می ماند از طرفی مقدار $(1-\beta_1)$ بین یک و صفر است و این مسئله باعث می شود ضرب این دو عبارت همچنان بین یک و صفر باقی بماند.

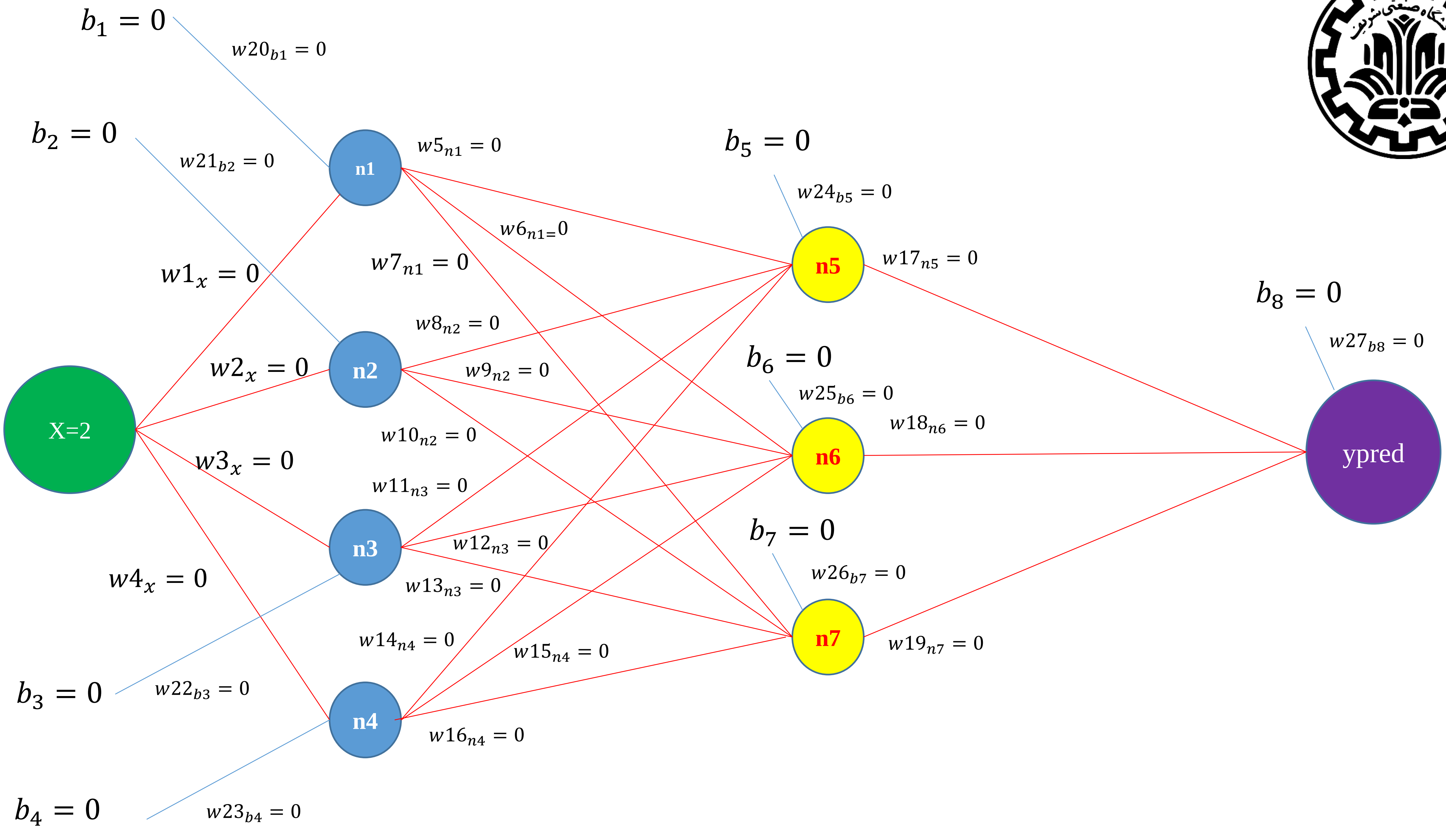
- برای رفع این مشکل ما از عبارت زیر استفاده می کنیم:

- $m_t' = \frac{m}{1-\beta_1^t}$

بر اساس این راهکار ما در گام های اول آموزش که مقدار t کوچک است عبارت مخرج یک عدد کوچک بین صفر و یک می شود اما بسیار نزدیک به صفر است و این مسئله باعث می شود که با تقسیم m بر یک عدد کوچک بین صفر عدد بزرگتری حاصل شود. در گام های بعدی رفته رفته چون ما به نقطه ی بهینه نزدیک می شود به روز رسانی های با شدت کمتری می خواهیم بنابراین به ازای t های بزرگ مخرج ما بزرگ می شود و به نزدیک یک میل می کند و m مقدار خودش باقی می ماند.

مسئله ۴. Backpropagation (۲۰ نمره)

الف) یک شبکه عصبی feedforward را با دولایه نهان با تابع فعال‌سازی sigmoid برای مسئله دسته‌بندی دودویی در نظر بگیرید. لایه اول نهان شامل ۴ نورون و لایه دوم شامل ۳ نورون است. ابعاد ورودی دلخواه در نظر گرفته می‌شود. در ابتدا تمامی وزن‌ها و بایاس شبکه صفر مقداردهی می‌شوند. به‌ازای یک ورودی، شبکه چه مقداری را خروجی دهد؟ بعد از یک بار به‌روز رسانی وزن‌ها با استفاده از SGD، بررسی کنید مقادیر وزن‌ها چگونه تغییر می‌کند.



$b_1 = 0$
$w20_{b1} = 0$
$b_2 = 0$
$w21_{b2} = 0$
$w1_x = 0$
$w2_x = 0$
$w3_x = 0$
$w4_x = 0$
$b_3 = 0$
$w22_{b3} = 0$
$b_4 = 0$
$w23_{b4} = 0$
X=2
n1
n2
n3
n4
$w5_{n1} = 0$
$w6_{n1=}0$
$w7_{n1} = 0$
$w8_{n2} = 0$
$w9_{n2} = 0$
$w10_{n2} = 0$
$w11_{n3} = 0$
$w12_{n3} = 0$
$w13_{n3} = 0$
$w14_{n4} = 0$
$w15_{n4} = 0$
$w16_{n4} = 0$
$b_5 = 0$
$w24_{b5} = 0$
$b_6 = 0$
$w25_{b6} = 0$
$b_7 = 0$
$w26_{b7} = 0$
n5
n6
n7
$w17_{n5} = 0$
$w18_{n6} = 0$
$w19_{n7} = 0$
$b_8 = 0$
$w27_{b8} = 0$
ypred

Forward path

Hidden layer 1 output:

- $n_1 = X * W1_x + b_1 * w20_{b1} = 2 * 0 + 0 * 0 = 0$
- $n_2 = X * W2_x + b_2 * w21_{b2} = 2 * 0 + 0 * 0 = 0$
- $n_3 = X * W3_x + b_3 * w22_{b3} = 2 * 0 + 0 * 0 = 0$
- $n_4 = X * W4_x + b_4 * w23_{b4} = 2 * 0 + 0 * 0 = 0$

- $out\ n_1 = \frac{1}{(1+e^{-n1})} = \frac{1}{(1+e^{-0})} = 0.5$
- $out\ n_3 = \frac{1}{(1+e^{-n3})} = \frac{1}{(1+e^{-0})} = 0.5$
- $out\ n_2 = \frac{1}{(1+e^{-n2})} = \frac{1}{(1+e^{-0})} = 0.5$
- $out\ n_4 = \frac{1}{(1+e^{-n4})} = \frac{1}{(1+e^{-0})} = 0.5$

Hidden layer 2 output:

- $n_5 = out\ n1 * W5_{n1} + out\ n2 \ * W8_{n2} + out\ n3 * W11_{n3} + out\ n4 * W14_{n4} + b_5 * w24_{b5} = 0.5 * 0 + 0.5 * 0 + 0.5 * 0 + 0.5 * 0 + 0 * 0 = 0$
- $n_6 = out\ n1 * W6_{n1} + out\ n2 * W9_{n2} + out\ n3 * W12_{n3} + out\ n4 * W15_{n4} + b_6 * w25_{b6} = 0.5 * 0 + 0.5 * 0 + 0.5 * 0 + 0.5 * 0 + 0 * 0 = 0$
- $n_7 = out\ n1 * W7_{n1} + out\ n2 * W10_{n2} + out\ n3 * W13_{n3} + out\ n4 * W16_{n4} + b_7 * w26_{b7} = 0.5 * 0 + 0.5 * 0 + 0.5 * 0 + 0.5 * 0 + 0 * 0 = 0$

- $out\ n_5 = \frac{1}{(1+e^{-n5})} = \frac{1}{(1+e^{-0})} = 0.5$
- $out\ n_6 = \frac{1}{(1+e^{-n6})} = \frac{1}{(1+e^{-0})} = 0.5$
- $out\ n_7 = \frac{1}{(1+e^{-n7})} = \frac{1}{(1+e^{-0})} = 0.5$

output:

- $ypred = out\ n5 * W17_{n5} + out\ n6 * W18_{n6} + out\ n7 * W19_{n7} + b_8 * w27_{b8} = 0.5 * 0 + 0.5 * 0 + 0.5 * 0 + 0.5 * 0 + 0 * 0 = 0$

- $outypred = \frac{1}{(1+e^{-ypred})} = \frac{1}{(1+e^{-0})} = 0.5$

$$J(\boldsymbol{w}) = -\sum_{i=1}^{n} \log p\left(y^{(i)} \middle| \boldsymbol{w}, \boldsymbol{x}^{(i)}\right)$$

$$= \sum_{i=1}^{n} -y^{(i)} \log\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right)\right) - (1 - y^{(i)}) \log\left(1 - f\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right)\right)$$

Assume: $y = 1$

$E = J(w) = -y * \log(out\ ypred) - 0 =$

$E = J(w) = -1 * \log(0.5) = -1 * -0.30102999566$=0.301

Backward path

Back propagation for hidden layer 2:

(Update $w17_{n5}, w18_{n6}, w19_{n7}, w27_{b8}$)

Step 1 ➡ $w17_{n5}$

❖ $\frac{\partial E}{\partial w17_{n5}} = \frac{\partial E}{\partial out\ ypred} * \frac{\partial out\ ypred}{\partial ypred} * \frac{\partial ypred}{\partial w17_{n5}}$

➢ $\frac{\partial E}{\partial out\ ypred} = -\frac{y}{out\ ypred} + \frac{(1-y)}{(1-out\ ypred)}$

➢ $\frac{\partial out\ ypred}{\partial ypred} = out\ ypred * (1 - out\ ypred)$

➢ $\frac{\partial E}{\partial out\ ypred} * \frac{\partial out\ ypred}{\partial ypred} = out\ ypred - y$

➢ $\frac{\partial ypred}{\partial w17_{n5}} = out\ n5 = 0.5$

$$\frac{\partial E}{\partial w17_{n5}} = (out\ ypred - y) * out\ n5 = -0.25$$

Backward path

Back propagation for hidden layer 2:

(Update $w17_{n5}, w18_{n6}, w19_{n7}, w27_{b8}$)

Step 2 ➡ $w18_{n6}$

❖ $$\frac{\partial E}{\partial w18_{n6}} = \frac{\partial E}{\partial out\ ypred} * \frac{\partial out\ ypred}{\partial ypred} * \frac{\partial ypred}{\partial w18_{n6}}$$

➢ $$\frac{\partial E}{\partial out\ ypred} = -\frac{y}{out\ ypred} + \frac{(1-y)}{(1-out\ ypred)}$$

➢ $$\frac{\partial out\ ypred}{\partial ypred} = out\ ypred * (1 - out\ ypred)$$

➢ $$\frac{\partial E}{\partial out\ ypred} * \frac{\partial out\ ypred}{\partial ypred} = out\ ypred - y$$

➢ $$\frac{\partial ypred}{\partial w18_{n6}} = out\ n6 = 0.5$$

➢ $$\frac{\partial E}{\partial w18_{n6}} = (out\ ypred - y) * out\ n5 = -0.25$$

Backward path

Back propagation for hidden layer 2:

(Update $w17_{n5}, w18_{n6}, w19_{n7}, w27_{b8}$)

Step 3 $\Rightarrow$ $w19_{n7}$

❖ $$\frac{\partial E}{\partial w19_{n7}} = \frac{\partial E}{\partial out\ ypred} * \frac{\partial out\ ypred}{\partial ypred} * \frac{\partial ypred}{\partial w19_{n7}}$$

➢ $$\frac{\partial E}{\partial out\ ypred} = -\frac{y}{out\ ypred} + \frac{(1-y)}{(1-out\ ypred)}$$

➢ $$\frac{\partial out\ ypred}{\partial ypred} = out\ ypred * (1 - out\ ypred)$$

➢ $$\frac{\partial E}{\partial out\ ypred} * \frac{\partial out\ ypred}{\partial ypred} = out\ ypred - y$$

➢ $$\frac{\partial ypred}{\partial w19_{n7}} = out\ n7$$

➢ $$\frac{\partial E}{\partial w19_{n7}} = (out\ ypred - y) * out\ n5 = -0.25$$

Backward path

Back propagation for hidden layer 2:

(Update $w17_{n5}, w18_{n6}, w19_{n7}, w27_{b8}$)

Step 4 ➡ $w27_{b8}$

❖ $$\frac{\partial E}{\partial w27_{b8}} = \frac{\partial E}{\partial out\ ypred} * \frac{\partial out\ ypred}{\partial ypred} * \frac{\partial ypred}{\partial w27_{b8}}$$

➢ $$\frac{\partial E}{\partial out\ ypred} = -\frac{y}{out\ ypred} + \frac{(1-y)}{(1-out\ ypred)}$$

➢ $$\frac{\partial out\ ypred}{\partial ypred} = out\ ypred * (1 - out\ ypred)$$

➢ $$\frac{\partial E}{\partial out\ ypred} * \frac{\partial out\ ypred}{\partial ypred} = out\ ypred - y$$

➢ $$\frac{\partial ypred}{\partial w27_{b8}} = b_8$$

➢ $$\frac{\partial E}{\partial w27_{b8}} = (out\ ypred - y) * b_8 = 0$$

Backward path

Back propagation for hidden layer 1:

(Update $w5_{n1}$,$w6_{n1}$, $w7_{n1}$, $w8_{n2}$, $w9_{n2}$, $w10_{n2}$, $w11_{n3}$, $w12_{n3}$, $w13_{n3}$, $w14_{n4}$, $w15_{n4}$, $w16_{n4}$, $w24_{b5}$, $w25_{b6}$, $w26_{b7}$)

Step 1 → $w5_{n1}$

- $$\frac{\partial E}{\partial w5_{n1}} = \frac{\partial E}{\partial out\ n5} * \frac{\partial out\ n5}{\partial\ n5} * \frac{\partial n5}{\partial w5_{n1}}$$

- $$\frac{\partial E}{\partial out\ n5} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n5} = 0$$

- $$\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$$

- $$\frac{\partial\ ypred}{\partial out\ n5} = W17_{n5} = 0$$

- $$\frac{\partial out\ n5}{\partial\ n5} = out\ n_5 * (1 - out\ n_5) = 0.25$$

- $$\frac{\partial n5}{\partial w5_{n1}} = out\ n1 = 0.5$$

→ $$\frac{\partial E}{\partial w5_{n1}} = 0$$

Backward path

Back propagation for hidden layer 1:

(Update $w5_{n1}, w6_{n1}, w7_{n1}, w8_{n2}, w9_{n2}, w10_{n2}, w11_{n3}, w12_{n3}, w13_{n3}, w14_{n4}, w15_{n4}, w16_{n4}, w24_{b5}, w25_{b6}, w26_{b7}$)

Step 2 $w6_{n1}$

- $\frac{\partial E}{w6_{n1}} = \frac{\partial E}{\partial out\ n6} * \frac{\partial out\ n6}{\partial\ n6} * \frac{\partial n6}{\partial w6_{n1}}$

- $\frac{\partial E}{\partial out\ n6} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n6} = 0$

- $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$

- $\frac{\partial\ ypred}{\partial out\ n6} = W18_{n6} = 0$

- $\frac{\partial out\ n6}{\partial\ n6} = out\ n_6 * (1 - out\ n_6) = 0.25$

- $\frac{\partial n6}{\partial w6_{n1}} = out\ n1 = 0.5$

$$\frac{\partial E}{w6_{n1}} = 0$$

Backward path

Back propagation for hidden layer 1:

(Update $w5_{n1}$,$w6_{n1}, w7_{n1}, w8_{n2}, w9_{n2}, w10_{n2}, w11_{n3}, w12_{n3}, w13_{n3}, w14_{n4}, w15_{n4}, w16_{n4}, w24_{b5}, w25_{b6}, w26_{b7}$)

Step 3 ➡ $w7_{n1}$

❖ $\frac{\partial E}{w7_{n1}} = \frac{\partial E}{\partial out\ n7} * \frac{\partial out\ n7}{\partial\ n7} * \frac{\partial n7}{\partial w7_{n1}}$

➢ $\frac{\partial E}{\partial out\ n7} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n7}$=0

➡ $\frac{\partial E}{w7_{n1}} = 0$

➢ $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$

➢ $\frac{\partial\ ypred}{\partial out\ n7} = W19_{n7} = 0$

➢ $\frac{\partial out\ n7}{\partial\ n7} = out\ n_7 * (1 - out\ n_7) = 0.25$

➢ $\frac{\partial n7}{\partial w7_{n1}} = out\ n1 = 0.5$

Backward path

Back propagation for hidden layer 1:

(Update $w5_{n1}$,$w6_{n1}$, $w7_{n1}$, $w8_{n2}$, $w9_{n2}$, $w10_{n2}$, $w11_{n3}$, $w12_{n3}$, $w13_{n3}$, $w14_{n4}$, $w15_{n4}$, $w16_{n4}$, $w24_{b5}$, $w25_{b6}$, $w26_{b7}$)

Step 4 → $w8_{n2}$

- $\frac{\partial E}{w8_{n2}} = \frac{\partial E}{\partial out\ n5} * \frac{\partial out\ n5}{\partial\ n5} * \frac{\partial n5}{\partial w8_{n2}}$

- $\frac{\partial E}{\partial out\ n5} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n5}$=0

→ $\frac{\partial E}{w8_{n2}} = 0$

- $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$

- $\frac{\partial\ ypred}{\partial out\ n5} = W17_{n5} = 0$

- $\frac{\partial out\ n5}{\partial\ n5} = out\ n_5 * (1 - out\ n_5) = 0.25$

- $\frac{\partial n5}{\partial w8_{n2}} = out\ n2 = 0.5$

Backward path

Back propagation for hidden layer 1:

(Update $w5_{n1}$,$w6_{n1}, w7_{n1}, w8_{n2}, w9_{n2}, w10_{n2}, w11_{n3}, w12_{n3}, w13_{n3}, w14_{n4}, w15_{n4}, w16_{n4}, w24_{b5}, w25_{b6}, w26_{b7}$)

As the same way:

$$\frac{\partial E}{w9_{n2}} = 0 \qquad \frac{\partial E}{w12_{n3}} = 0 \qquad \frac{\partial E}{w15_{n4}} = 0$$

$$\frac{\partial E}{w10_{n2}} = 0 \qquad \frac{\partial E}{w13_{n3}} = 0 \qquad \frac{\partial E}{w16_{n4}} = 0$$

$$\frac{\partial E}{w11_{n3}} = 0 \qquad \frac{\partial E}{w14_{n4}} = 0$$

Backward path

Back propagation for hidden layer 1:

(Update $w5_{n1}, w6_{n1}, w7_{n1}, w8_{n2}, w9_{n2}, w10_{n2}, w11_{n3}, w12_{n3}, w13_{n3}, w14_{n4}, w15_{n4}, w16_{n4}, w24_{b5}, w25_{b6}, w26_{b7}$)

Step 13 ➡ $w24_{b5}$

❖ $\frac{\partial E}{w24_{b5}} = \frac{\partial E}{\partial out\ n5} * \frac{\partial out\ n5}{\partial\ n5} * \frac{\partial n5}{\partial w24_{b5}}$

➢ $\frac{\partial E}{\partial out\ n5} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n5}$=0

➢ $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$

➢ $\frac{\partial\ ypred}{\partial out\ n5} = W17_{n5} = 0$

➢ $\frac{\partial out\ n5}{\partial\ n5} = out\ n_5 * (1 - out\ n_5) = 0.25$

➢ $\frac{\partial n5}{\partial w24_{b5}} = b_5 = 0$

➡ $\frac{\partial E}{w24_{b5}} = 0$

As the same way:

$$\begin{cases} \frac{\partial E}{w25_{b6}} = 0 \\ \frac{\partial E}{w26_{b7}} = 0 \end{cases}$$

Back propagation for hidden layer 0

$Update\ w1_x, w2_x, w3_x, w4_x, w20_{b1}, w21_{b2}, w22_{b3}, w23_{b4}$

Step 1 ⇨ $w1_x$

❖ $\frac{\partial E}{w1_x} = \frac{\partial E}{\partial out\ n1} * \frac{\partial out\ n1}{\partial\ n1} * \frac{\partial n1}{w1_x}$

➢ $\frac{\partial E}{\partial out\ n1} = \frac{\partial E}{\partial\ n_5} * \frac{\partial\ n5}{\partial out\ n_1} + \frac{\partial E}{\partial\ n_6} * \frac{\partial\ n6}{\partial out\ n_1} + \frac{\partial E}{\partial\ n_7} * \frac{\partial\ n7}{\partial out\ n_1} = 0$

- ➢ $\frac{\partial E}{\partial\ n_5} = \frac{\partial E}{\partial\ out\ n_5} * \frac{\partial\ out\ n_5}{\partial\ n_5} = 0$
- ➢ $\frac{\partial E}{\partial out\ n5} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n5}$=0
- ➢ $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$
- ➢ $\frac{\partial\ ypred}{\partial out\ n5} = W17_{n5} = 0$

- ➢ $\frac{\partial E}{\partial\ n6} = \frac{\partial E}{\partial\ out\ n_6} * \frac{\partial\ out\ n_6}{\partial\ n_6} = 0$
- ➢ $\frac{\partial E}{\partial out\ n6} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n5}$=0
- ➢ $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$
- ➢ $\frac{\partial\ ypred}{\partial out\ n6} = W18_{n6} = 0$

Backward path

- $\frac{\partial E}{\partial\, n_7} = \frac{\partial E}{\partial\ out\ n_7} * \frac{\partial\ out\ n_7}{\partial\ n_7} = 0$
- $\frac{\partial E}{\partial out\ n7} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n7}$=0
- $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$
- $\frac{\partial\ ypred}{\partial out\ n7} = W19_{n7} = 0$

- $\frac{\partial out\ n1}{\partial\ n1} = out\ n1 * (1 - out\ n1) = 0.25$
- $\frac{\partial n1}{w1_x} = x = 2$

- $\frac{\partial E}{w1_x} = \frac{\partial E}{\partial out\ n1} * \frac{\partial out\ n1}{\partial\ n1} * \frac{\partial n1}{w1_x} = 0 * 0.25 * 2$
- $\frac{\partial E}{w1_x} = 0$

Backward path

Back propagation for hidden layer 0:

$Update\ w2_x, w3_{x,} w4_x, w20_{b1}, w21_{b2}, w22_{b3}, w23_{b4}$

As the same way:

$$\frac{\partial E}{w2_x} = 0$$

$$\frac{\partial E}{w3_x} = 0$$

$$\frac{\partial E}{w4_x} = 0$$

Update all weights

❖ $w = w - lr * \frac{\partial E}{\partial w}$

$$\partial w17_{n5} = \partial w17_{n5} - lr * \frac{\partial E}{\partial w17_{n5}} = 0 - \mathrm{lr} * (-0.25)$$

$$\partial w18_{n6} = \partial w18_{n6} - lr * \frac{\partial E}{\partial w18_{n6}} = 0 - \mathrm{lr} * (-0.25)$$

$$\partial w19_{n7} = \partial w19_{n7} - lr * \frac{\partial E}{\partial w19_{n7}} = 0 - \mathrm{lr} * (-0.25)$$

همانطور که دیده شد چون در مقداردهی اولیه وزن همه وزنها صفر بوده است بنابراین آموزش به خوبی انجام نمی شود و گرادیان بسیاری از وزنها صفر شده است و بنابراین آپدیت نمی شوند.

$Update\ w1_x, w2_x, w3_x, w4_x, w20_{b1}, w21_{b2}, w22_{b3}, w23_{b4}$

Step 1 ➡ $w20_{b1}$

❖ $\frac{\partial E}{w20_{b1}} = \frac{\partial E}{\partial out\ n1} * \frac{\partial out\ n1}{\partial\ n1} * \frac{\partial n1}{w20_{b1}}$

➢ $\frac{\partial E}{\partial out\ n1} = \frac{\partial E}{\partial\ n_5} * \frac{\partial\ n5}{\partial out\ n_1} + \frac{\partial E}{\partial\ n_6} * \frac{\partial\ n6}{\partial out\ n_1} + \frac{\partial E}{\partial\ n_7} * \frac{\partial\ n7}{\partial out\ n_1} = 0$

➢ $\frac{\partial E}{\partial\ n_5} = \frac{\partial E}{\partial\ out\ n_5} * \frac{\partial\ out\ n_5}{\partial\ n_5} = 0$

➢ $\frac{\partial E}{\partial out\ n5} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n5} = 0$

➢ $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y$
$= -0.5$

➢ $\frac{\partial\ ypred}{\partial out\ n5} = W17_{n5} = 0$

➢ $\frac{\partial E}{\partial\ n6} = \frac{\partial E}{\partial\ out\ n_6} * \frac{\partial\ out\ n_6}{\partial\ n_6} = 0$

➢ $\frac{\partial E}{\partial out\ n6} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n5} = 0$

➢ $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$

➢ $\frac{\partial\ ypred}{\partial out\ n6} = W18_{n6} = 0$

Backward path

- $\frac{\partial E}{\partial\, n_7} = \frac{\partial E}{\partial\ out\ n_7} * \frac{\partial\ out\ n_7}{\partial\ n_7} = 0$
- $\frac{\partial E}{\partial out\ n7} = \frac{\partial E}{\partial\ ypred} * \frac{\partial\ ypred}{\partial out\ n7}$=0
- $\frac{\partial E}{\partial ypred} = \frac{\partial E}{\partial\ out\ ypred} * \frac{\partial\ out\ ypred}{\partial\ ypred} = out\ ypred - y = -0.5$
- $\frac{\partial\ ypred}{\partial out\ n7} = W19_{n7} = 0$

- $\frac{\partial out\ n1}{\partial\ n1} = out\ n1 * (1 - out\ n1) = 0.25$
- $\frac{\partial n1}{w20_{b1}} = b_1 = 0$

- $\frac{\partial E}{w20_{b1}} = \frac{\partial E}{\partial out\ n1} * \frac{\partial out\ n1}{\partial\ n1} * \frac{\partial n1}{w20_{b1}} = 0 * 0.25 * 0$
- $\frac{\partial E}{w20_{b1}} = 0$

Back propagation for hidden layer 0:

$Update\ w21_{b2}, w22_{b3}, w23_{b4}$

As the same way:

$$
\begin{cases}
\dfrac{\partial E}{w21_{b2}} = 0 \\[2ex]
\dfrac{\partial E}{w22_{b3}} = 0 \\[2ex]
\dfrac{\partial E}{w23_{b4}} = 0
\end{cases}
$$

$$w5_{n1} = w5_{n1} - \mathrm{lr} *$$

ب) شبکه زیر را در نظر بگیرید.

$$h = W^T x$$
$$u = W'^T h$$
$$\hat{y} = Softmax(u)$$
$$\mathcal{L}(W, W') = -y^T \log \hat{y}$$

که در آن $x \in \mathbb{R}^d$ و $h \in \mathbb{R}^H$ و $\hat{y} \in \mathbb{R}^d$ و y برچسب *one-hot* شده است. با در نظر گرفتن تابع هزینه برای یک نمونه ورودی، با استفاده از روش گرادیان کاهشی روابط به‌روز رسانی وزن‌ها را با نوشتن جزئیات مراحل بنویسید.

فرضیات مسئله

$$w = \begin{bmatrix} w_1^1, & \cdots & , w_H^1 \\ \vdots & \ddots & \vdots \\ w_1^D, & \cdots & w_H^D \end{bmatrix}_{D*H} \qquad x = \begin{bmatrix} x_1^i \\ . \\ . \\ . \\ x_D^i \end{bmatrix}_{D*1} \qquad W' = \begin{bmatrix} w, & \cdots & w \\ \vdots & \ddots & \vdots \\ w, & \cdots & w \end{bmatrix}_{H*D}$$

- با توجه به فرضیات مسئله داریم:

$$ypred = [0, \dots \quad 1 \quad , \dots 0]_{1*D}$$

$$y = [0, \dots \quad 1 \quad , \dots 0]_{1*D}$$

$$h = W^T x$$
$$u = W'^T h$$
$$\hat{y} = Softmax(u)$$
$$\mathcal{L}(W, W') = -y^T \log \hat{y}$$

$$xi \xrightarrow{w} h \xrightarrow{w'} u \longrightarrow ypred$$

- با توجه به فرضیات و برای یک نمونه داریم:

1

$$\mathcal{L}(W, W') = -y^T \log \hat{y}$$

2

$$ypred_k = \frac{\exp(u_k)}{\sum_j u_j} \qquad k = 1, \dots, n$$

1

$$\mathcal{L}(W,W') = -y^T \log \hat{y}$$

2

$$ypred_k = \frac{\exp(u_k)}{\sum_j u_j} \quad k = 1, \dots, n$$

$$\frac{\partial L}{\partial w'} = \frac{\partial L}{\partial u} * \frac{\partial u}{\partial w'}$$

به دلیل اینکه خروجی تابع سافت مکس به همه ورودهایش بستگی دارد به همین دلیل عناصر غیر قطری ماتریس jacobian غیر صفر می باشد و چون احتمال هستند همگی مثبت هستند بنابراین ما با استفاده از یک trick می توانیم به جای مشتق نسبی، نسبت به log خروجی ها مشتق نسبی بگیریم:

$$\begin{bmatrix} \partial ypred_1/\partial u_1, & \cdots & \partial ypred_1/\partial u_n, \\ \vdots & \ddots & \vdots \\ \partial ypred_n/\partial u_1, & \cdots & \partial ypred_n/\partial u_n, \end{bmatrix}_{D*H}$$

$$\frac{\partial}{\partial u_j}\log(ypred_i) = \frac{1}{s_i} * \frac{\partial ypred_i}{\partial u_j}$$

$$\frac{\partial ypred_i}{\partial u_j} = s_i * \frac{\partial}{\partial u_j}\log(ypred_i)$$

$$\log ypred_i = \log\left(\frac{\exp(u_i)}{\sum_{l=1}^{n}\exp(u_l)}\right) = u_i - \log(\sum_{l=1}^{n}\exp(u_l))$$

$$\frac{\partial}{\partial u_j}\log(ypred_i) = \frac{\partial u_i}{\partial u_j} - \frac{\partial}{\partial u_j} * \log(\sum_{l=1}^{n}\exp(u_l)$$

$$\frac{\partial}{\partial u_j}\log(ypred_i) = \frac{\partial u_i}{\partial u_j} - \frac{\partial}{\partial u_j} * \log(\sum_{l=1}^{n} \exp(u_l)) \Rightarrow \begin{cases} \frac{\partial u_i}{\partial u_j} = 1 & \text{اگر i==j باشد} \\ \frac{\partial u_i}{\partial u_j} = 0 & \text{اگر i!=j باشد} \end{cases}$$

$$\frac{\partial}{\partial u_j}\log(ypred_i) = 1\{i == j\} - \frac{1}{\sum_{l=1}^{n}\exp(u_l)} \left(\frac{\partial}{\partial u_j}\sum_{l=1}^{n}\exp(u_l)\right) \Rightarrow \frac{\partial}{\partial u_j}\sum_{l=1}^{n}\exp(u_l) = e^{u_j}$$

$$\frac{\partial}{\partial u_j}\log(ypred_i) = 1\{i == j\} - \frac{e^{u_j}}{\sum_{l=1}^{n}\exp(u_l)} = 1\{i == j\} - ypred_j$$

برای تابع هزینه cross entropy داریم:

$$\frac{\partial ypred_i}{\partial u_j} = ypred_i * \frac{\partial}{\partial u_j}\log(ypred_i) = ypred_i * (1\{i == j\} - ypred_j) \Rightarrow -\sum_{i=1}^{n} ytrue_i * \log(ypred_i)$$

$$-\sum_{i=1}^{n} ytrue_i * \log(ypred_i) \Rightarrow \frac{\partial L}{\partial u_i} = -\frac{\partial}{\partial u_i}\sum_{i=1}^{n} ytrue_i * \log(ypred_i) - \sum_{i=1}^{n} ytrue_i * \frac{\partial}{\partial u_i}\log(ypred_i)=$$

$$= -\sum_{i=1}^{n} \frac{ytrue_i}{ypred_i} * \frac{\partial ypred_i}{\partial u_i} \Rightarrow \frac{\partial L}{\partial u_i} = -\sum_{i=1}^{n} \frac{ytrue_i}{ypred_i} * ypred_i * \left(1\{i == j\} - ypred_j\right)$$

$$\frac{\partial L}{\partial u_i} = -\sum_{i=1}^{n} ytrue_i * \left(1\{i == j\} - ypred_j\right) \Rightarrow \frac{\partial L}{\partial u_i} = \sum_{i=1}^{n} ytrue_i * ypred_j - \sum_{i=1}^{n} ytrue_i * 1\{i == j\}$$

$\Rightarrow$ چون زمان هایی که i==j است مقدار ۱ می شود می توانیم بنویسیم $\Rightarrow$

$$\frac{\partial L}{\partial u_i} = \sum_{i=1}^{n} ytrue_i * ypred_j - ytrue_j$$

چون طبق فرضیات مسئله one-hot هستیم

$$\Rightarrow \frac{\partial L}{\partial u_i} = ypred_j * \sum_{i=1}^{n} ytrue_i - ytrue_j = ypred_j\text{-}ytrue_j \Rightarrow \frac{\partial L}{\partial u} = ypred - ytrue$$

❖ $\frac{\partial L}{\partial w'} = \frac{\partial L}{\partial u} * \frac{\partial u}{\partial w'}$

$$\frac{\partial L}{\partial u} = ypred - ytrue$$

$$u = {W'}^T h \Rightarrow \frac{\partial u}{\partial w'} = h$$

$$\Rightarrow \frac{\partial L}{\partial w'} = h * (ypred - ytrue)$$

❖ $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial h} * \frac{\partial h}{\partial w}$

$$\frac{\partial L}{\partial h} = \frac{\partial L}{\partial u} * \frac{\partial u}{\partial h} = w' * (ypred - ytrue)$$

$$\frac{\partial L}{\partial u} = \frac{\partial L}{\partial ypred} * \frac{\partial ypred}{\partial u} = ypred - ytrue$$

$$\Rightarrow \frac{\partial L}{\partial w} = w' * (ypred - ytrue) * x$$

$$w = w - lr * \frac{\partial L}{\partial w}$$

$$w' = w' - lr * \frac{\partial L}{\partial w'}$$

$$\Rightarrow$$

$$w = w - lr * (w' * (ypred - ytrue) * x)$$

$$w' = w' - lr * (h * (ypred - ytrue))$$