Assignment NO.6 Solutions

Deep learning | spring 1401 | Dr.Mohammadi

Teacher Assistant:

Fatemeh anvari

Hadiseh mahmoudi

Student name : **Amin Fathi**

Student id : **400722102**

Problem 1

روش به کار رفته در این قسمت در واقع در ابتدا میانگین بردار های گرادیان را محاسبه کرده و سپس آن را از بردار های گرادیان کم میکند تا میانگین به صفر برسد و به همین دلیل نیز به آن GC (gradient centralization) می گویند

$$\Phi_{GC}(\nabla_{\mathbf{w}_i}\mathcal{L}) = \nabla_{\mathbf{w}_i}\mathcal{L} - \mu_{\nabla_{\mathbf{w}_i}\mathcal{L}}$$

$$\text{where } \mu_{\nabla_{\mathbf{w}_i}\mathcal{L}} = \frac{1}{M}\sum_{j=1}^{M}\nabla_{\mathbf{W}_{i,j}}\mathcal{L}.$$

در واقع ما فقط نیاز داریم تنها میانگین را برای هر یک از ستون های ماتریس گرادیان وزن ها حساب کنیم و سپس داریه های هر ستون را از میانگین آن کم کنیم و سپس ماتریس حاصل را در آپدیت کردن وزن ها به کار ببریم ، لازم به ذکر است این روش قابلیت سازگاری با انواع اپتیمایزر ها را دارد و به طور مثال نحوه استفاده آن به همراه اپتیمایزر Adam را در شکل زیر مشاهده می کنید :

Algorithm 2 Adam with Gradient Centralization

Input: Weight vector $\mathbf{w}^0$, step size α, β_1, β_2, ϵ, $\mathbf{m}^0$,$\mathbf{v}^0$

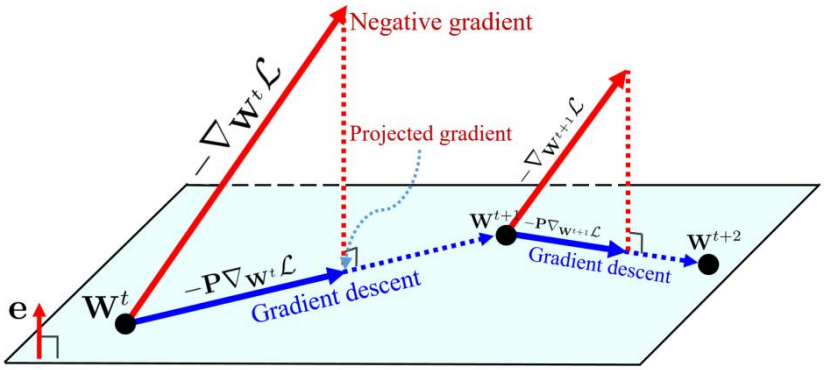
Training step:

1: **for** $t = 1, ...T$ **do**
2: $\mathbf{g}^t = \nabla_{\mathbf{w}^t}\mathcal{L}$
3: $\widehat{\mathbf{g}}^t = \Phi_{GC}(\mathbf{g}^t)$
4: $\mathbf{m}^t = \beta_1\mathbf{m}^{t-1} + (1-\beta_1)\widehat{\mathbf{g}}^t$
5: $\mathbf{v}^t = \beta_2\mathbf{v}^{t-1} + (1-\beta_2)\widehat{\mathbf{g}}^t \odot \widehat{\mathbf{g}}^t$
6: $\widehat{\mathbf{m}}^t = \mathbf{m}^t/(1-(\beta_1)^t)$
7: $\widehat{\mathbf{v}}^t = \mathbf{v}^t/(1-(\beta_2)^t)$
8: $\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha\frac{\widehat{\mathbf{m}}^t}{\sqrt{\widehat{\mathbf{v}}^t}+\epsilon}$
9: **end for**

که در واقع در خط شماره 3 گرادیان GC را حساب کرده و سپس از آن در ادامه پروسه استفاده کرده است .

در واقع ما با این کار به نوعی بردار گرادیان در فضایی n بعدی را به فضای hyperplane n-1 بعدی نگاشت می کنیم و از آن برای آپدیت کردن وزن ها استفاده می کنیم

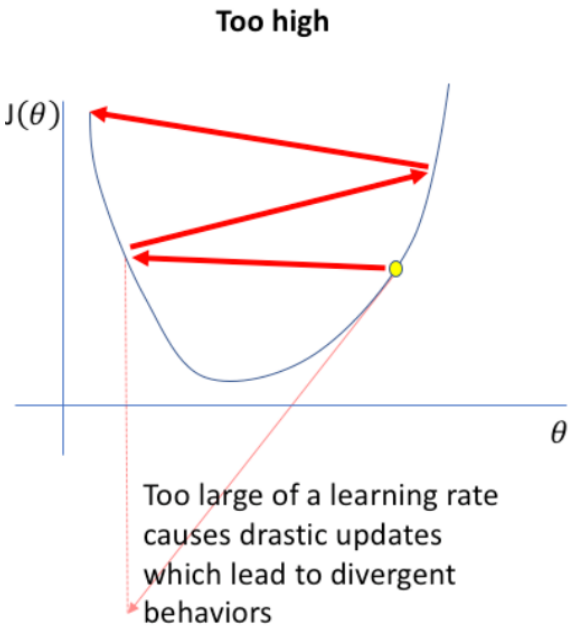
در شکل زیر نحوه آپدیت شدن وزن ها در t+1 و t+2 را میتوان مشاهده کرد .



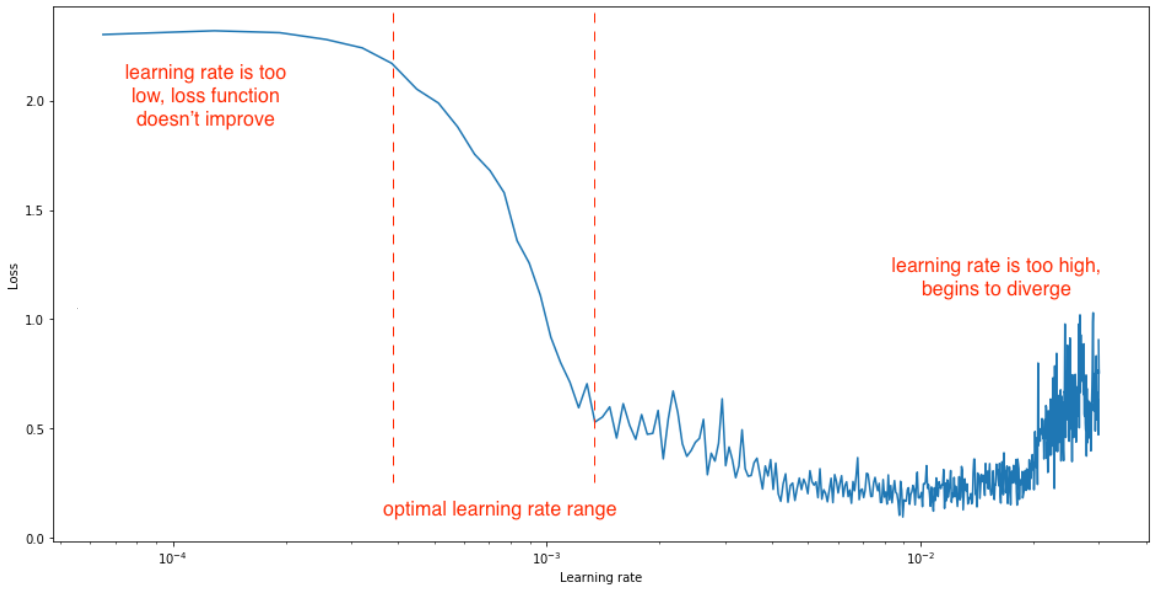
از طرف دیگر می توان نشان داد (در ضمیمه مقاله) که اگر میانگین وزن های اولیه شبکه نزدیک به صفر باشند بنابر این خروجی نسبت به تغییرات شدید بردار ویژگی های ورودی مقاوم است و این ویژگی با استفاده از GC مهیا می شود (چرا که قصد آن تبدیل میانگین وزن های هر ستون به ۰ است) و این عمل تعمیم شبکه را افزایش می دهد .

Problem 2

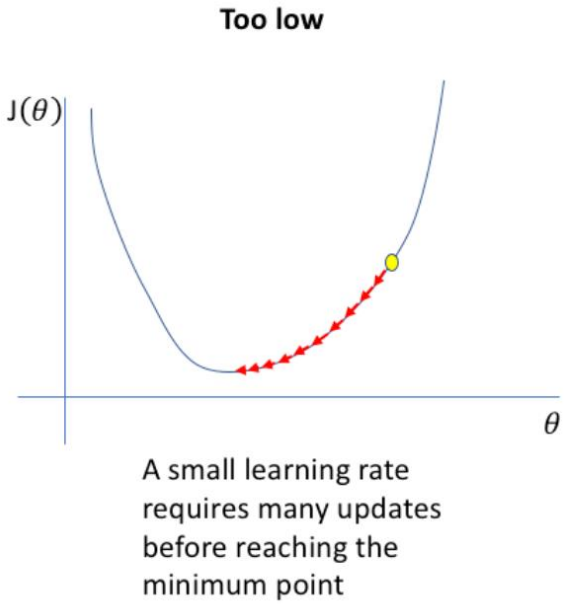
چنانچه نرخ یادگیری بزرگی انتخاب شود ما سریعتر به نقطه بهینه همگرا می شویم ولی در حوالی نقطه بهینه چنانچه نرخ آموزش همچنان بزرگ باقی بماند می تواند باعث جابه جایی های مکرر در اطراف نقطه بهینه شود ، راه تشخیص آن را میتوان در تفاوت در وزن های آپدیت شده دید ، چنانچه سیر به روز رسانی وزن ها به گونه ای باشد که به دفعات و به صورت یک در میان وزن آپدیت شده نسبت به وزن قبلی خود کاهش یا افزایش داشته باشد میتوان حدس زد که نرخ یادگیری بالا می باشد ، این مفهوم در شکل زیر نمایش داده شده است .



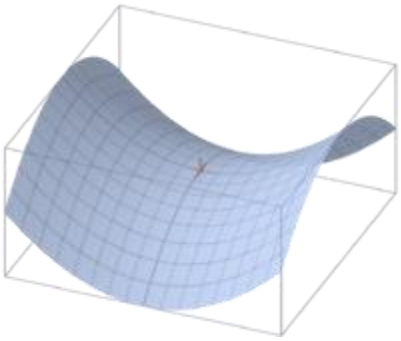
همچنین از روی loss به جا مانده هم میتوان به این قضیه پی برد ، چنانچه مقدار loss مدل به صورت نوسانی در حال افزایش و کاهش بود یا اصلاحا واگرا ، میتواند یکی از نشانه های نرخ یادگیری بسیار بالا باشد .



نرخ یادگیری پایین هم سرعت همگرایی بسیار پایینی دارد و میتواند باعث طولانی شدن روند همگرایی و یافتن نقطه بهینه شود



ج) نقطه زینی در واقع نقطه ای است با شیب صفر و سکون که البته اکسترمم موضعی نیست و به طور مثال در یک راستا به بالا و در یک راستای دیگر به پایین خم می شود ، همانند نقطه قرمز در شکل زیر :



در sgd چنانچه نقطه ابتدایی به صورت رندوم و شانسی انتخاب شود احتمال گیر افتادن در نقطه زینی تقریبا صفر است چرا که مساحتی که نقطه زینی به نسبت کل نقاط اشغال میکند تقریبا صفر است فلذا احتمال گیر کردن هم تقریبا صفر است اگر از mini batch استفاده کند . در مورد adam هم با توجه به اینکه از مشتق اول و دوم استفاده می شود لزوما صفر شدن مشتق اول موجب بروز اشکال نمی شود .

د) در شکل B از mini batch استفاده شده است چرا که تابع ضرر به صورت نوسان دار است و این به این خاطر است که در واقع در هر ایپاک داده های رندم و متفاوتی به شبکه به عنوان ورودی برای یادگیری می دهیم و این باعث نوسان در فرایند یادگیری و تابع ضرر می شود

منابع :

اسلاید های درس استاد

Intro to optimization in deep learning: Gradient Descent (paperspace.com)

Saddle point - Wikipedia

(2) What is a saddle point in machine learning? - Quora

Why doesn't gradient descent terminate on saddle point? - Cross Validated (stackexchange.com)

gradient descent - How can it be trapped in a saddle point? - Cross Validated (stackexchange.com)

Problem 3

الف) ابتدا خروجی z1 و z2 و در نتیجه h1 و h2 را حساب می کنیم :

$$z_1 = i_1 \times w_{11} + i_2 w_{21} + b_1$$
$$= (2 \times 1) + (-1 \times 0.5) + 0.5 = 2$$

$$z_2 = i_1 \times w_{12} + i_2 w_{22} + b_2 = (2 \times -0.5) + (-1 \times -1) - 0.5$$
$$= -0.5$$

$$h_1 = Relu(2) = 2$$
$$h_2 = Relu(-0.5) = 0$$

حال به سراغ محاسبه o1 و o2 می رویم :

$$O_1 = h_1 \times w_{31} + h_2 w_{41} + b_3 = 1 - 1 = 0$$
$$O_2 = h_1 \times w_{32} + h_2 w_{42} + b_4 = -2 + 0.5 = -1.5$$

ب) مقدار خطای MSE به شرح زیر است :

$$MSE = \frac{1}{n} \sum_{1}^{n} (Y_i - \hat{Y}_i)^2 = \frac{1}{2} \left[(1-0)^2 + (0.5 + 1.5)^2 \right]$$
$$= \frac{1}{2} \times 5 = 2.5$$

ج) برای update کردن مقدار w21 مشتق زنجیره ای را می نویسیم :

$$\frac{\partial E_{total}}{\partial w_{21}} = \left[\left(\frac{\partial E_1}{\partial o_1} \times \frac{\partial o_1}{\partial h_1}\right) + \left(\frac{\partial E_2}{\partial o_2} \times \frac{\partial o_2}{\partial h_1}\right)\right] \times \frac{\partial h_1}{\partial z_1} \times \frac{\partial z_1}{\partial w_{21}}$$

مشتق خطا نسبت به هر یک از ورودی ها طبق فرمولی زیر محاسبه می شود :

$$2 \times -1 \times \frac{1}{n} \times (y - \hat{y}_i) = 2 \times \frac{1}{n} \times \hat{y}_i - y_i$$

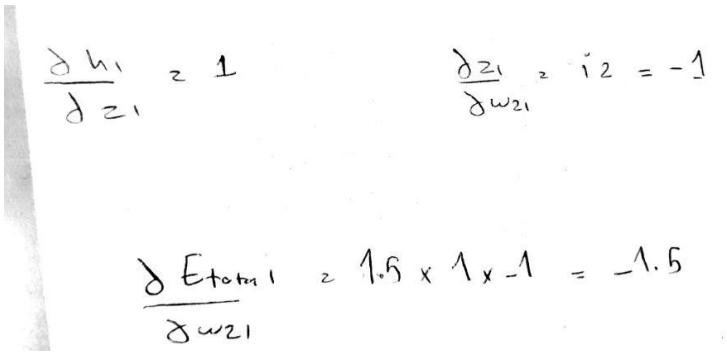
$$= \sum_{i=1}^{n=2} (\hat{y}_i - y_i)$$

$$\frac{\partial E_1}{\partial o_1} \times \frac{\partial o_1}{\partial h_1} = (o_1 - t_1) \times w_{31} = -1 \times 0.5 = -0.5$$

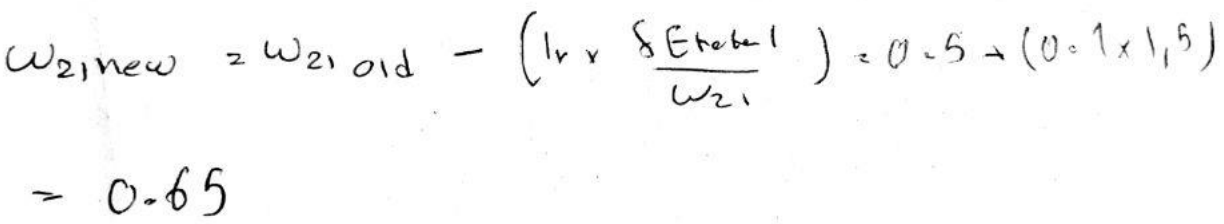
$$\frac{\partial E_2}{\partial o_2} \times \frac{\partial o_2}{\partial h_1} = (o_2 - t_2) \times w_{32} = -2 \times -1 = 2$$

مشتق خروجی تابع Relu چنانچه ورودی اش مثبت باشد برابر ۱ است و چنانچه ورودی اش منفی باشد برابر ۰ است .

حاصل مشتق زنجیره ای را حساب می کنیم که برابر 1.5- می باشد .



و حال وزن w21 را به روز رسانی می کنیم.



در مورد آپدیت کردن وزن w22 با توجه به اینکه در قائده مشتق زنجیره ایش از مشتق z2 نسبت به h2 هم استفاده می کنیم .

(که برابر ۰ است ، فلذا وزن w22 آپدیت نمی شود)

توضیحات ریاضی بالا را به صورت کد پیاده سازی کردم که در پیوست آمده است که نتایج را به شرح زیر مشاهده می کنید :

خروجی o1 و o2 در forward :

```
forward(input)
```

```
array([[ 0. , -1.5]])
```

خطای MSE برای خروجی هم به شرح زیر است :

```
MSE(forward(input))
```

```
2.5
```

همچنین خروجی لایه های اول بعد از یک بار backward به شرح زیر است :

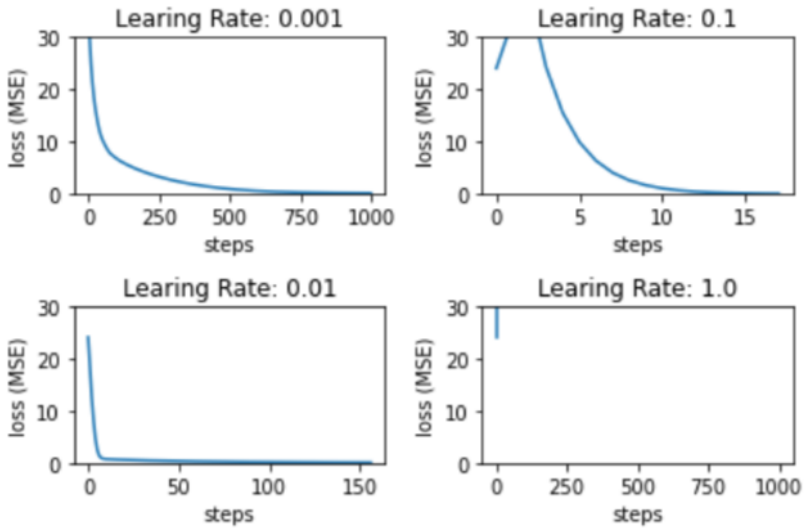
```
backward(forward(input))
```

```
array([[ 0.7 , -0.5 ],
       [ 0.65, -1.  ]])
```

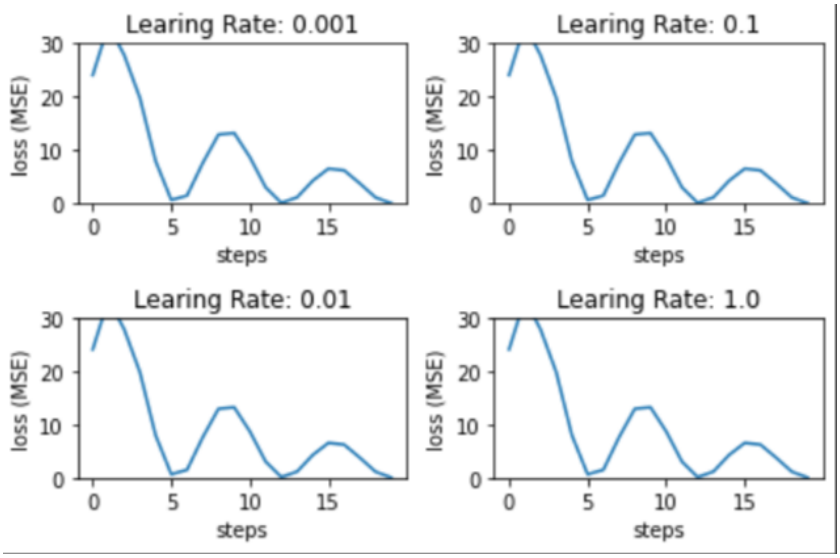
که درایه های ماتریس آن نظیر ماتریس زیر است :

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

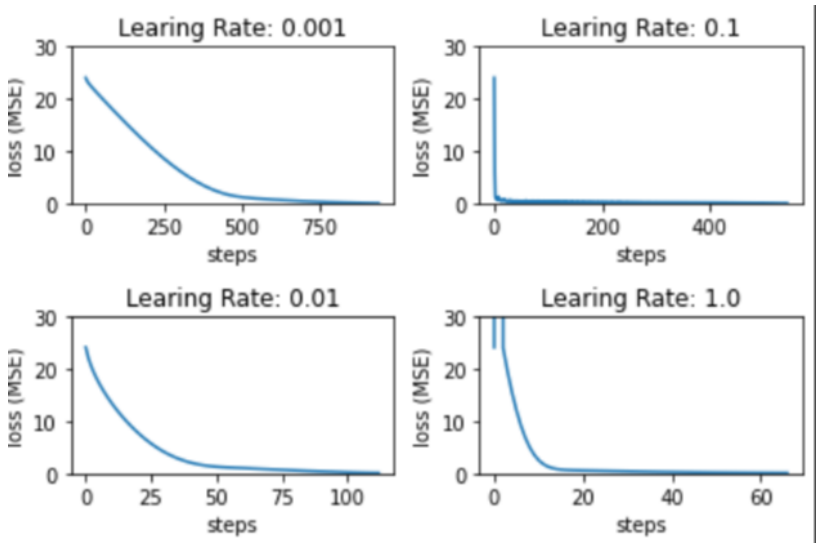
Problem4



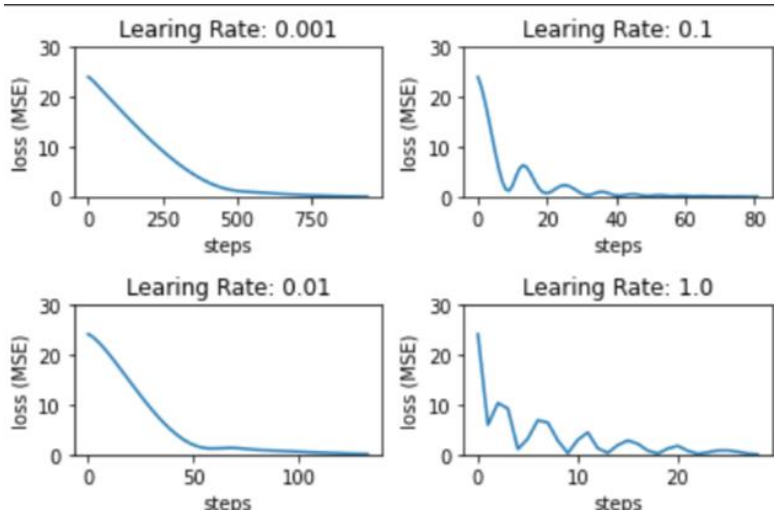
همانطور که مشاهده می شود در نرخ یادگیری 0.001 یادگیری به شدت کند بوده و تا قدم ۱۰۰۰ ام باید پیش رفته و این به معنای این است که نرخ یادگیری بسیار پایینی را انتخاب کرده ایم که دیر همگرا می شود ، در نرخ یادگیری 0.01 اوضا خوب است و نرخ یادگیری مناسبی است ، به نظر می رسد نرخ یادگیری 0.1 با batch های داده ی ما در ابتدا سر ناسازگاری دارد و توانایی یادگیری الگو بین دسته داده های اولیه را ندارد وبرای همین خطا در ابتدا افزایش یافته است و سپس کاهش یافته ، مشاهده می شود که در ایپاک و قدم کمتری نسبت به نرخ یادگیری 0.01 همگرا می شود و این نیز گزینه مناسبی است ؛ اما در مورد نرخ یادگیری 1 می توان مشاهده کرد که این نرخ یادگیری ، بسیار زیاد بوده و باعث واگرایی مدل می شود !



به نظر کد اشتباه می آید چون هر ۴ تصویر یک حالت را نشان می دهند .



از آنجا که RMSprop در پی نرمالیزه کردن گرادیان هاست و به میانگین گرادیان ها کار دارد ، در نیتجه می تواند بهتر از پس مسایل پیچیده تر بر بیاد ، چون در گرادیان بالا ،قدم ها را اهسته تر میکند تا انفجار گرادیان و واگرایی رخ ندهد (در نرخ یادگیری ۱ مشاهده میشود) و در و در گرادیان های پایین هم قدم ها را سریعتر می کند (نرخ یادگیری 0.001 که نسبته به sgd سریعتر است) نرخ یادگیری 0.01 هم بهترین نرخ یادگیری است و نرخ یادگیری 0.1 هم احتمالا میانگین گرادیان ها نزدیک صفر می باشد ، فلذا قدم ها در حوالی نقطه بهینه بسیار کوچک بوده و دیرتر از 1 و 0.01 همگرا می شود .



مشاهده می شود که با افزایش نرخ یادگیری همگرایی سریعتر رخ می دهد ، اما همچنان می شود فهمید که در تصویر ها که با توجه به این که از mini batch استفاده شده است در برخی قدم ها رفتار سینوسی داریم خصوصا در نرخ های یادگیری بالا .

منابع :

neural networks - Explanation of Spikes in training loss vs. iterations with Adam Optimizer - Cross Validated (stackexchange.com)