سید محمد عرفان موسوی منزه
401722199
تمرین سوم یادگیری عمیق

1

الف

بهینه ساز AdaBelief که در مقاله پیشنهاد شده، برای ایجاد تعادل بین هم گرایی سریع و توانایی تعمیم در بهینه سازی طراحی شده است. هدف AdaBelief دستیابی به هم گرایی سریع مانند روشهای اداپتیو و تعمیم خوب مانند روش های شتابی است.

ایده ای که این بهینه ساز استفاده می کند، ایجاد یک پیش بینی از میزان گرادیان آینده (طول گام به طور دقیق تر) و مقایسه گرادیان بدست آمده با آن است. اگر گرادیان بدست آمده با گرادیان پیش بینی شده بسیار متفاوت باشد، این گرادیان قابل اعتماد نخواهد بود و با ضریب کمی در جهتش حرکت می کنیم. (استپ کوچک تری بر میداریم). نحوه پیش بینی گرادیان بر اساس استفاده از Exponential moving average است. اگر گرادیان بدست آمده با پیش بینی نزدیک باشد گام بزرگتری بر میداریم.

مزایای AdaBelief این است که از سایر روش ها از نظر هم گرایی سریع، دقت بالا و پایداری در کارهای مختلف یادگیری عمیق مانند طبقه بندی تصویر و مدل سازی زبان بهتر عمل می کند. به طور خاص، AdaBelief به دقت قابل مقایسه با SGD در ImageNet دست می یابد و پایداری بالایی را نشان می دهد و کیفیت نمونه های تولید شده را در مقایسه با بهینه ساز Adam به خوبی تنظیم شده در آموزش GAN در Cifar10 بهبود می بخشد.

2

ب

Algorithm 1: Adam Optimizer

Initialize θ_0, $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$

While θ_t not converged

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

Bias Correction

$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$, $\widehat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t}$

Update

$\theta_t \leftarrow \prod_{\mathcal{F}, \sqrt{\widehat{v_t}}} \left(\theta_{t-1} - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} \right)$

Algorithm 2: AdaBelief Optimizer

Initialize θ_0, $m_0 \leftarrow 0$, $s_0 \leftarrow 0$, $t \leftarrow 0$

While θ_t not converged

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$s_t \leftarrow \beta_2 s_{t-1} + (1-\beta_2)(g_t - m_t)^2 + \epsilon$

Bias Correction

$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}$, $\widehat{s_t} \leftarrow \frac{s_t}{1-\beta_2^t}$

Update

$\theta_t \leftarrow \prod_{\mathcal{F}, \sqrt{\widehat{s_t}}} \left(\theta_{t-1} - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{s_t}} + \epsilon} \right)$

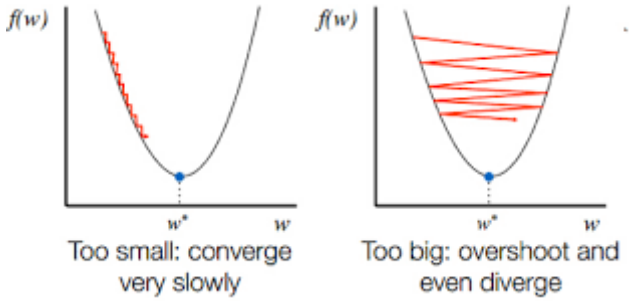
همانطور که در عکس مشخص شده است، این الگوریتم بسیار به آدام شبیه است به جز قسمت های آبی رنگ که در تصویر بالا مشخص شده است.

- g_t = gradian value at step t
- m_t = EMA of g_t
- v_t = EMA of g_t^2
- s_t = EMA of $(g_t - m_t)^2$

با در نظر گرفتن متغیر های بالا: در آدام ما بعد از عملیات Bias Correction مقادیر m را بر $\sqrt{v_t}$ تقسیم میکنیم اما در آداب لیف بر $\sqrt{s_t}$ تقسیم می کنیم.

اگر $\sqrt{s_t}$ مقدار بزرگی داشته باشد، به این معنی که تفاوت زیادی بین m_t و g_t وجود داشته است. در این صورت میزان آپدیت کوچک می شود. و بلعکس اگر مقدار کوچکی داشته باشد، استپ بزرگ تر می شود.

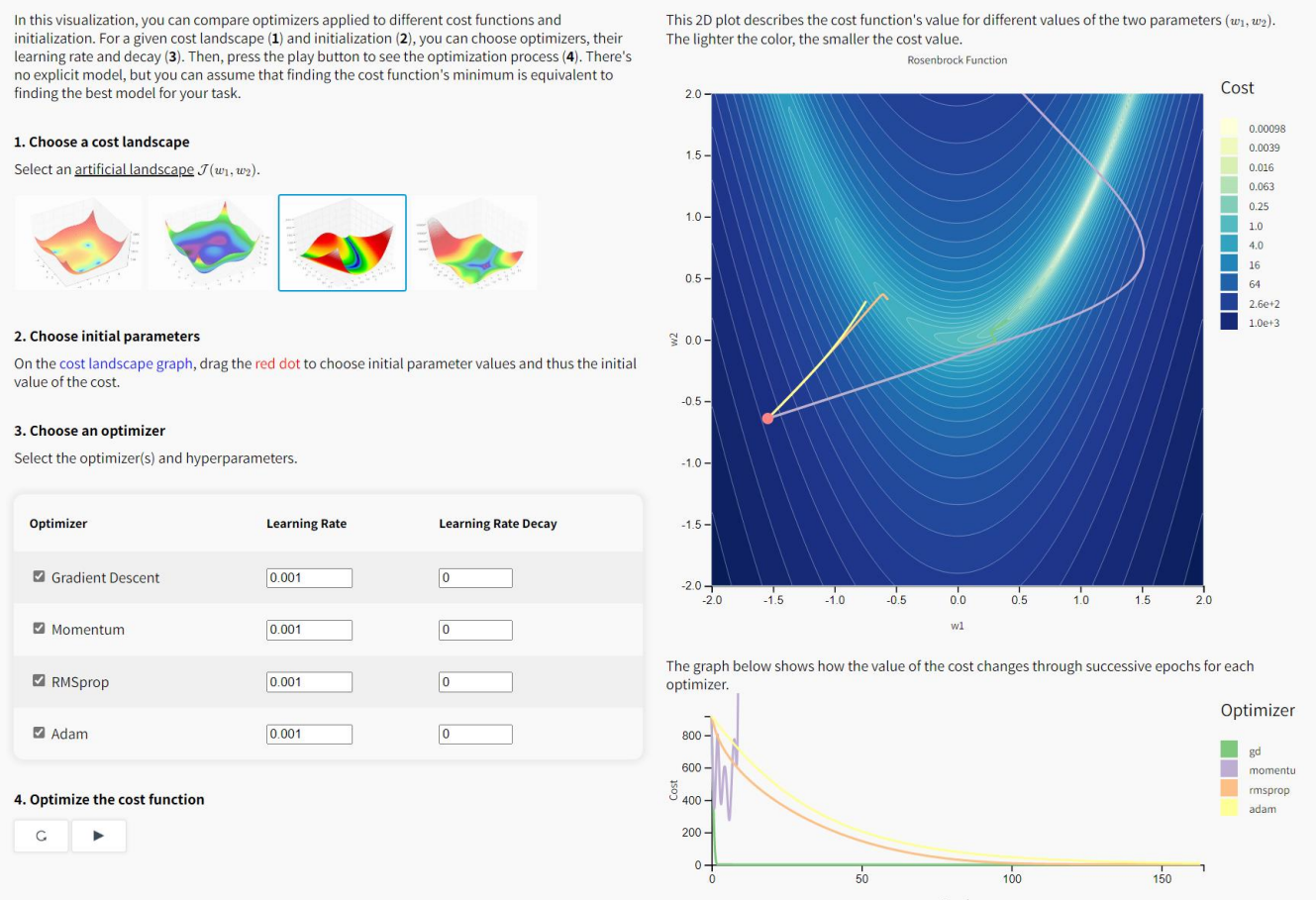
2
الف



با در نظر گرفتن نرخ یادگیری بزرگ، احتمال رخ دادن آورشوت پیش می آید. در این حالت الگوریتم ممکن است از نقطه مینیمم عبور کند و دوباره به نقاط با لاس بالا برگردد.
در تصویر بالا، این قضیه نشان داده شده است. در عکس سمت راست، این کانسپت نمایش داده شده است.
2
ب
برای رفع این مشکل میتوان از الگوریتم هایی مانند نمونه سوال قبل استفاده کرد. اینها اصطلاحا از adaptive learning rate استفاده می کنند که متناسب با فضای فعلی مسئله نرخ یادگیری تنظیم میشود.
راه حل دیگر استفاده از زمانبند های نرخ یادگیری است.
2
ت
نرخ یادگیری کوچک باعث کند شدن الگوریتم می شود. همچنین باعث گیر افتادن الگوریتم در مینیمم محلی می شود. نرخ یادگیری بزرگ باعث overshoot شدن می شود.

برای غلبه بر این مشکل راه حل های متفاوتی پیشنهاد شده است. به عنوان مثال میتوان از ممان استفاده کرد. در این حالت گرادیان مانند شتاب و ممان مانند سرعت عمل می کنند. داشتن سرعت در هر نقطه صرف نظر از مقدار شتاب (گرادیان) جلوی گیر افتادن در مینیمم محلی را میگیرد.
راه حل دیگر استفاده از زمانبند های نرخ آموزش است که به طور پیوسته نرخ آموزش را متناسب با گام آموزش تنظیم میکنند. همچنین استفاده از روش هایی مانند آدام و آرام اس پراب نیز مناسب است. این روش ها با درنظر کرفتن گرادیان سرعت حرکت در راستای های مختلف را تنظیم می کنند.

الف

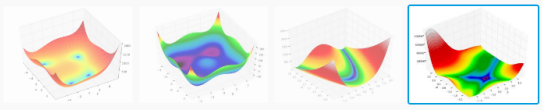


با توجه به convex بودن فضای مسیله، GD توانسته است به سرعت به مینیمم محلی برسد. GD با حرکت در شیب مسئله به سرعت به این نقطه رسیده است. اما Momentum نتوانسته است همگرا شود. دلیل آن داشتن سرعت است. اگر به تصویر نگاه کنیم میبینیم که ممان در ابتدا در مسیر رسیدن به بهینه اصلی حرکت می کرده است اما به خاطر داشتن سرعت از کنار آن رد شده و بعد کاملا در جهت دیگری حرکت کرده است. RMSو Adam عملکردی شبیه به هم داشته اند اما آدام مقداری کند تر از آر ام اس حرکت کرده است.

In this visualization, you can compare optimizers applied to different cost functions and initialization. For a given cost landscape (**1**) and initialization (**2**), you can choose optimizers, their learning rate and decay (**3**). Then, press the play button to see the optimization process (**4**). There's no explicit model, but you can assume that finding the cost function's minimum is equivalent to finding the best model for your task.

1. Choose a cost landscape

Select an artificial landscape $\mathcal{J}(w_1, w_2)$.



2. Choose initial parameters

On the cost landscape graph, drag the red dot to choose initial parameter values and thus the initial value of the cost.

3. Choose an optimizer

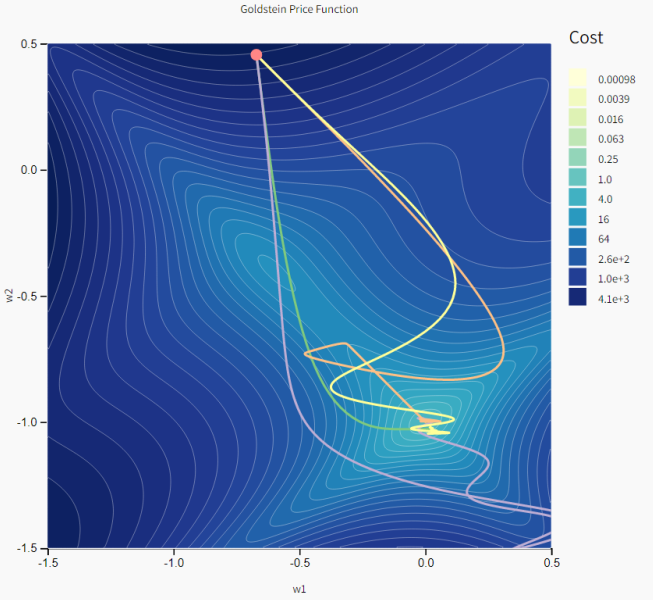
Select the optimizer(s) and hyperparameters.

Optimizer	Learning Rate	Learning Rate Decay
☑ Gradient Descent	0.000032	0
☑ Momentum	0.0000025	0.01
☑ RMSprop	0.5	0.1
☑ Adam	0.2	0.02

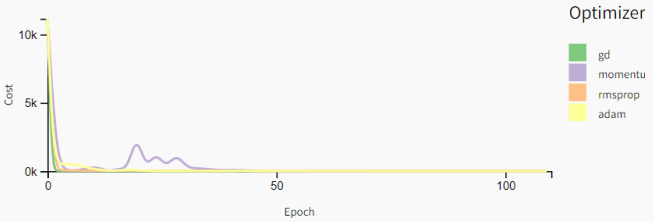
4. Optimize the cost function

↻ ▶

This 2D plot describes the cost function's value for different values of the two parameters (w_1, w_2). The lighter the color, the smaller the cost value.



The graph below shows how the value of the cost changes through successive epochs for each optimizer.

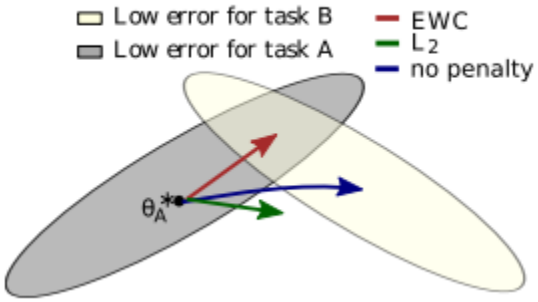


مقاله زیر مطالعه شد و راه حل پیشنهادی ازین مقاله است:

Overcoming catastrophic forgetting in neural networks - James Kirkpatrick et al. – 2017 - DeepMind

روش پیشنهادی مقاله با کاهش سرعت یادگیری وزن هایی که برای تسک های قبلی با اهمیت است، آن ها را حفظ می کند و در نتیجه باعث حفظ آن تسک می شود.

برای توضیح بیشتر، هنگامی که یک شبکه عصبی یاد می گیرد، پارامترهای عددی یا همون وزن ها را تنظیم می کند که نحوه پردازش اطلاعات را تعیین می کند. در روش بیان شده در مقاله، به طور انتخابی فرآیند یادگیری را برای وزن هایی که برای کارهای قدیمی مهم هستند، کند می کنیم. این بدان معناست که شبکه هنگام یادگیری یک کار جدید، این وزن ها را به شدت تغییر نمی دهد، بنابراین حتی پس از یادگیری کارهای جدید، همچنان می تواند وظیفه قبلی را به خوبی انجام دهد. این رویکرد از روش یادگیری انسان الهام گرفته شده است، جایی که ما فراموش نمی کنیم که چگونه کارهایی را که مدت ها پیش آموخته ایم انجام دهیم، حتی اگر مدتی است آنها را انجام نداده باشیم.



تصویر بالا اثر استفاده از روش پیشنهادی مقاله را نشان میدهد. در این حالت به کمک لاس جدید ما به نقطه ای رسیدیم که هم برای تسک اول و هم تسک دوم مناسب است. (نقطه اشتراک بهینه دو تسک)

6

الف

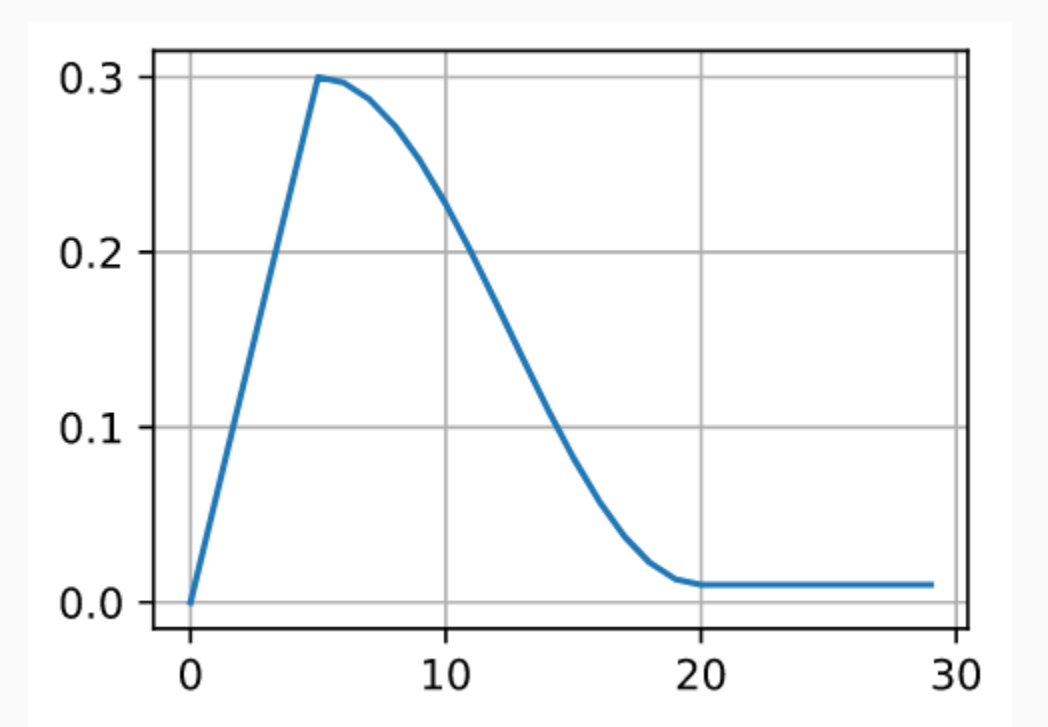
در یادگیری عمیق، گرم کردن به تکنیکی اطلاق می شود که در آن نرخ یادگیری یک شبکه عصبی به تدریج از مقدار کم اولیه به مقدار مطلوب در طول چند دوره اول آموزش افزایش می یابد. این تکنیک اغلب برای حل مشکل ناپدید شدن گرادیان ها استفاده می شود، که می تواند زمانی رخ دهد که گرادیان تابع لاس نسبت به وزن شبکه بسیار کوچک شود و یادگیری شبکه را دشوار کند. ایده پشت گرم کردن این است که به شبکه اجازه داده شود تا با شروع یادگیری، به آرامی با وزن‌های جدید سازگار شود، نه اینکه فوراً سعی کنیم به‌روزرسانی‌های بزرگی برای وزن‌ها انجام دهیم که می‌تواند باعث از بین رفتن گرادیان شود. با افزایش تدریجی نرخ یادگیری، شبکه می‌تواند به‌روزرسانی‌های کوچک و تدریجی وزنه‌ها را انجام دهد که می‌تواند به جلوگیری از ناپدید شدن گرادیان‌ها و بهبود پایداری کلی فرآیند آموزش کمک کند. استفاده از گرم کردن در یادگیری عمیق چندین مزیت دارد. اول، می تواند به بهبود نرخ همگرایی شبکه کمک کند و به آن امکان می دهد سریعتر و کارآمدتر یاد بگیرد. دوم، می‌تواند به جلوگیری از گیرکردن شبکه در بهینه های محلی در طول آموزش کمک کند، که می‌تواند زمانی رخ دهد که شیب‌ها خیلی کوچک شده و شبکه قادر به پیشرفت بیشتر نباشد. در نهایت، گرم کردن می تواند به بهبود دقت کلی شبکه کمک کند و به آن اجازه می دهد تا تنظیمات کوچکی را در وزن ها در مراحل اولیه تمرین انجام دهد، که می تواند تأثیر قابل توجهی بر عملکرد نهایی شبکه داشته باشد.

6

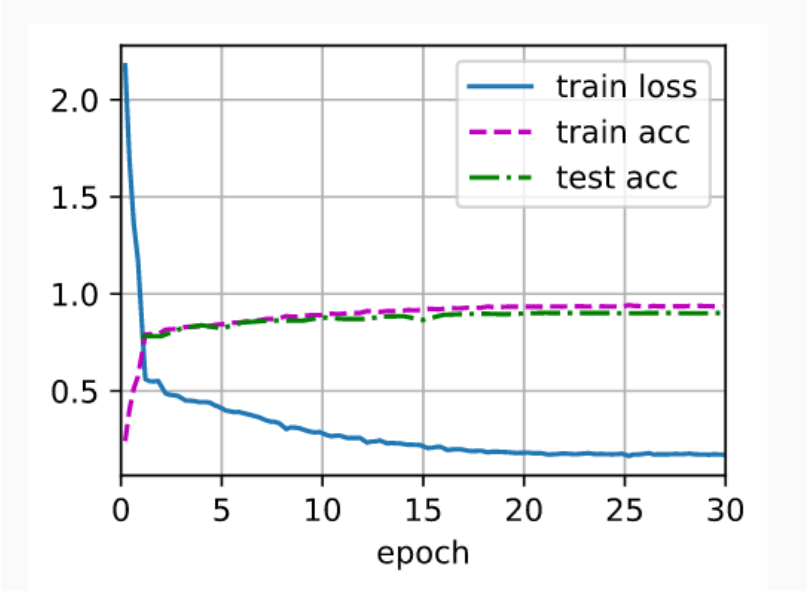
ب

گرم کردن و کاهش سرعت یادگیری هر دو تکنیک‌هایی هستند که در یادگیری عمیق برای تنظیم میزان یادگیری در طول تمرین استفاده می‌شوند، اما تفاوت‌های مهمی در نحوه کار دارند. گرم کردن تکنیکی است که به تدریج نرخ یادگیری شبکه عصبی را از مقدار اولیه کم به مقدار بالاتر در طول چند دوره اول آموزش افزایش می دهد. ایده پشت گرم کردن این است که شبکه را از به‌روزرسانی وزن‌های بزرگ در اوایل فرآیند تمرین، که می‌تواند باعث ناپایدار شدن گرادیان‌ها و منجر به عملکرد ضعیف شود، جلوگیری کند. با افزایش تدریجی نرخ یادگیری، گرم کردن به شبکه اجازه می‌دهد تا به‌روزرسانی‌های کوچک‌تر و پایدارتر وزنه‌ها را انجام دهد، که می‌تواند به بهبود پایداری و کارایی کلی فرآیند تمرین کمک کند. در مقابل، کاهش نرخ یادگیری تکنیکی است که نرخ یادگیری شبکه را در طول زمان با پیشرفت آموزش کاهش می دهد. ایده کاهش نرخ یادگیری این است که به شبکه اجازه می‌دهد در اوایل تمرین، زمانی که گرادیان‌ها بزرگ‌تر هستند، به‌روزرسانی‌های وزن بزرگ‌تر را انجام دهد، اما سپس با شروع همگرا شدن شبکه، سرعت یادگیری را کاهش داد تا از ایجاد به‌روزرسانی‌های وزن بزرگ که می‌تواند گرادیان‌ها را بی‌ثبات کند، جلوگیری کند. و منجر به عملکرد ضعیف شود. یکی از تفاوت های کلیدی بین کاهش سرعت گرم کردن و یادگیری تمرکز آنها بر مراحل مختلف فرآیند تمرین است. Warmup روی مراحل اولیه تمرین متمرکز است، جایی که شبکه هنوز در حال تنظیم با وزنه های جدید است و ممکن است مستعد بی ثباتی باشد. در مقابل، کاهش سرعت یادگیری بر مراحل بعدی آموزش متمرکز است، جایی که شبکه قبلاً به یک راه حل همگرا شده است و ممکن است از به روز رسانی های وزن کمتر برای تنظیم دقیق عملکرد خود بهره مند شود. تفاوت مهم دیگر بین این دو تکنیک تأثیر آنها بر پایداری کلی شبکه است. Warmup برای جلوگیری از ناپایدار شدن شبکه در اوایل تمرین با به روز رسانی وزن های کوچکتر طراحی شده است، در حالی که کاهش نرخ یادگیری برای جلوگیری از ناپایدار شدن شبکه در مراحل بعدی تمرین با ایجاد به روز رسانی های کوچکتر برای تنظیم دقیق عملکرد آن طراحی شده است.

در برخی موارد، مقدار دهی اولیه پارامتر ها برای تضمین یک راه حل بهینه کافی نیست. این مشکل به ویژه در طراحی شبکه های پیشرفته که ممکن است منجر به مشکلات بهینه سازی ناپایدار شوند، وجود دارد. می توان این مشکل را با انتخاب یک نرخ یادگیری کافی کوچک حل کرد تا در ابتدا از انحراف جلوگیری شود. با این حال، این کار به معنای پیشرفت کند است. برعکس، یک نرخ یادگیری بزرگ در ابتدا به انحراف منجر می شود.راه حلی نسبتاً ساده برای این تناقض استفاده از یک دوره گرم شدن است، در طی این دوره نرخ یادگیری به آرامی تا مقدار اولیه بیشینه افزایش پیدا کرده و سپس تا پایان فرآیند به کاهش می رسد. به طور معمول، یک افزایش خطی در طی دوره گرم شدن استفاده می شود که باعث تشکیل یک برنامه زمانی با فرم زیر می شود:



اینکار به بهبود همگرایی در استپ های ابتدایی شبکه می شود. به شکل زیر که نمایانگر همین نکته است توجه کنید:



استفاده از گرم کردن در هر زمان بندی امکان پذیر است که این یک مزیت بزرگ برای آن به حساب می آید. با تحلیل نمودار بالا، یک روند پایدار در آموزش را مشاهده می کنیم که این امر حاصل استفاده از گرم کردن خطی ابتدای آموزش است که باعث شده است شبکه در نقاط بهتری قرار بگیرد.

$$y = w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2 + b$$

batch 1 → sample 1 and 2

$$y = x_1^2 - x_2^2 - x_1 x_2 + 1$$

$\hat{y}_1 = 1 - 1 - (1)(-1) + 1 = 2$ $\quad mse(y, \hat{y}) = \frac{1}{2}((1-2)^2 + (13-5)^2) = \frac{1}{2}[64+64] = 64$

$\hat{y}_2 = 4 - 0 - 0 + 1 = 5$

$$\frac{\partial L}{\partial w_1} = \sum_{i=1}^{2}(-y_i + w_1 x_{1i}^2 + w_2 x_{2i}^2 + w_3 x_{1i} x_{2i} + b) x_{1i}^2$$

$$\frac{\partial L}{\partial w_2} = \sum_{i=1}^{2}(-y_i + w_1 x_{1i}^2 + w_2 x_{2i}^2 + w_3 x_{1i} x_{2i} + b) x_{2i}^2$$

$$\frac{\partial L}{\partial w_3} = \sum_{i=1}^{2}(-y_i + w_1 x_{1i}^2 + w_2 x_{2i}^2 + w_3 x_{1i} x_{2i} + b) x_{1i} x_{2i}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{2}(-y_i + w_1 x_{1i}^2 + w_2 x_{2i}^2 + w_3 x_{1i} x_{2i} + b)$$

$$\frac{\partial L}{\partial w_1} = -8 \times 1 - 8 \times 2 = -24$$

$$\frac{\partial L}{\partial w_2} = -8 \times (-1) - 8 \times 0 = 8$$

$$\frac{\partial L}{\partial w_3} = -8 \times 1 \times -1 - 8 \times 2 \times 0 = 8$$

$$\frac{\partial L}{\partial b} = -8 - 8 = -16$$

momentum

$V_0 = 0, 0, 0, 0$

$V_{t+1} = \beta V_t + \nabla f(x)$ $\quad \beta \rightarrow V_1 = 0.9(0, 0, 0, 0) + (-24, 8, 8, -16) = (-24, 8, 8, -16)$

$x_{t+1} = x_t - \alpha V_{t+1} \rightarrow x_1 = (w_1, w_2, w_3, b) = (1, -1, -1, 1) - 0.1(-24, 8, 8, -16) = (3.4, -1.8, -1.8, 2.6)$

batch 2 → 3, 4

$y = 3.4 x_1^2 - 1.8 x_2^2 - 1.8 x_1 x_2 + 2.6$

$\hat{y}_3 = 3.4 \times 0 - 1.8 \times 2^2 - 1.8 \times 0 \times 2 + 2.6 = 9.8$

$\hat{y}_4 = 3.4 \times (-1)^2 - 1.8 \times 1^2 - 1.8(-1) \times 1 + 2.6 = 6$

$mse(y, \hat{y}) = \frac{1}{2}((11-9.8)^2 + (4-6)^2) = 2.6$

$$\frac{\partial L}{\partial w_1} = -(11-9.8) \times 0 - (4-6) \times (-1)^2 = 2$$

$$\frac{\partial L}{\partial w_2} = -(11-9.8) \times 2^2 - (4-6) \times 1 = -2.8$$

$$\frac{\partial L}{\partial w_3} = -(11-9.8) \times 0 \times 2 - (4-6) \times -1 \times 1 = -2$$

$$\frac{\partial L}{\partial b} = -(11-9.8) - (4-6) = 0.8$$

$V_1 = (-24, 8, 8, -16) \rightarrow V_2 = 0.9(-24, 8, 8, -16)$

$V_2 = (-19.6, 4.4, 5.2, -13.6) + (2, -2.8, -2, 0.8)$

$(w_1, w_2, w_3, b) = (3.4, -1.8, -1.8, 2.6) - 0.1 V_2$

$= (5.36, -2.24, -2.32, 3.96)$