به نام خدا

مهرانه مقتدائی فر ۹۷۲۲۲۰۸۶

تاریخ: ۱۴۰۰/۱۱/۰۵

گزارش تمرین سری ۵ – OPTIMIZATION

درس: مبانی علوم داده

استاد درس: دکتر خردپیشه

سوالات تئوری

۱.

a) این loss در مسئله های classification بیشتر مورد استفاده قرار میگیرد. برای آنکه متوجه شویم که از کدام کلاس چند label درست پیشبینی شده است. استفاده از negative log likelihood به ما کمک میکند تا بتوانیم با minimize کردن این مقدار، احتمال آنکه چقدر از لیبل های واقعی، درست پیشبینی شده است بیشینه کنیم. همچنین چون محاسبات با استفاده از log صورت میگیرد راحت تر است. هر چه مقدار این Loss کمتر باشد یعنی به پیشبینی صحیح نزدیک تر هستیم.

b) این دو روش برای جلوگیری از overfit شدن مدل استفاده میشوند. به این ترتیب که در هردو یک جمله به loss function اضافه خواهد شد درواقع یک penalty term دارند که با اضافه شدن بر loss function میتواند برخی feature ها را انتخاب کند و برای دیتاست هایی که دارای feature های زیادی هستند کاربرد دارند.

در L1 regularization جمله ای که به تابع هزینه اضافه میشود، برابر است با مجموع قدرمطلق تمامی وزن های مدل که در یک مقدار ثابت λ ضرب میشود تا تاثیر این جمله را کنترل کند. این جمله در زیر مشخص است:

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

عملکرد این term بسیار به انتخاب λ وابسته است. به طوریکه اگر λ برابر با صفر باشد همان loss function قبلی را خواهیم داشت و اما اگر λ مقدار بزرگی اختیار کند، ضرایب به سمت صفر میروند و موجب under fit شدن میشود. با این روند و انتخاب λ مناسب ، مدل خلوت تر خواهد شد. بدین ترتیب که فیچر هایی با اهمیت کمتر، با استفاده از این تکنین به سمت صفر میل خواهند کرد و درواقع نوعی feature selection داریم.

در L2 regularization در جمله ای که به تابع هزینه اضافه میشود، برابر است با جمع مجذور تمامی وزن های مدل که باز هم در مقدار ثابت λ ضرب میشود. در اینجا اگر λ خیلی بزرگ باشد آنگاه باعث میشود که وزن زیادی به مدل اضافه شود و باز هم دچار Underfitting خواهیم شد.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

درواقع L2 وزن های مدل یادگیری را وا می دارد تا کوچک باقی بمانند و بر خلاف L1 آن ها را صفر نمی کند؛ در نتیجه استفاده از L2 منجر به ایجاد یک مدل خلوت نمی شود. یعنی موجب feature selection نمیشود اما میتوانیم جلوی overfitt شدن مدل را بگیریم.

تفاوت اصلی این دو تکنیک در آن جمله‌ای است که به مدل اضافه میکنیم.

همانطور که گفتیم L1 خاصیت feature selection دارد اما L2 اینطور نیست.

ممکن است براساس اثردهی زیرمجموعه های متفاوت از ویژگی ها، مدل با L1 چند یادگیری متفاوت ایجاد کند اما با L2 چنین نیست و تنها یک یادگیری دارد.

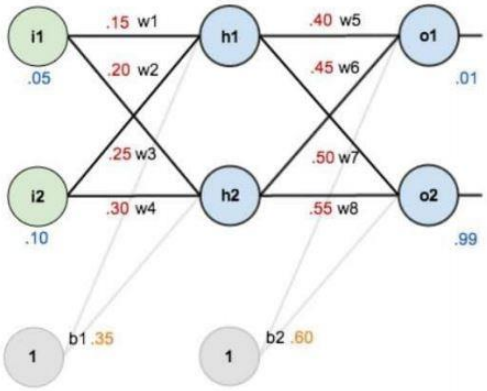
L1 در مواجهه با داده های outlier عملکرد خوبی دارد اما L2 چنین نیست، چرا که در نقاط outlier خطای پیش‌بینی مدل بسیار زیاد می شود و با داشتن جمله پنالتی L2 در آن تابع، وزن های مدل کوچکتر خواهد شد.

c) Momentum جمله‌ای است که با اضافه شدن به SGD باعث میشود تا بردار های گرادیان در جهت اصلی و درست، به سرعت حرکت کنند و منجر به همگرایی سریعتر مدل خواهد شد. درواقع یک نوع optimizer است که به عنوان یک ضریب اضافه میشود و به مدل کمک میکند تا سریعتر مقدار مورد نظر را پیدا کند.

۲.

a.

در مرحله feed-forward باید برای هر لایه محاسبه کنیم که چقدر از هر لایه به لایه بعدی منتقل میشود.



برای h1 داریم:

$$a_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 = 0.15 * 0.05 + 0.20 * 0.10 + 0.35 = 0.3775$$

اعمال sigmoid بر روی a_{h1}:

$$out_{h_1} = \frac{1}{1 + e^{-a_{h1}}} = 0.593269$$

برای *h2* داریم:

$$a_{h2} = 0.05 * 0.25 + 0.10 * 0.30 + 0.35 = 0.3925$$

اعمال sigmoid بر روی a_{h2}:

$$out_{h2} = \frac{1}{1 + e^{-a_{h2}}} = 0.596884$$

حال در مرحله بعد ورودی را دو نورون h2 , h1 در نظر میگریم تا خروجی o1 و o2 را محاسبه کنیم.

برای o1 داریم:

$$a_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 = 0.40 * 0.593269 + 0.45 * 0.596884 + 0.60 = 1.1059054$$

اعمال sigmoid بر روی a_{o1}:

$$out_{o1} = \frac{1}{1 + e^{-a_{o1}}} = 0.7513649$$

برای o2 داریم:

$$a_{o2} = 0.50 * 0.593269 + 0.55 * 0.596884 + 0.60 = 1.2249207$$

اعمال sigmoid بر روی a_{o2}:

$$out_{o2} = \frac{1}{1 + e^{-a_{o2}}} = 0.77292834$$

محاسبه خطا برای o1 و o2:

$$E_{o1} = 1/2(target_{o1} - out_{o1})^2 = 1/2(0.01 - 0.7513649)^2 = 0.2748109575$$

$$E_{o2} = \frac{1}{2}(0.99 - 0.77292834)^2 = 0.0235600528$$

به طور کلی خطا برابر است با:

$$E_{total} = E_{o1} + E_{o2} = 0.298371$$

.b

اعمال backpropagation برای آپدیت کردن وزن ها است. به طوریکه خروجی به target اصلی نزدیک تر شود و بدین ترتیب خطا شبکه کاهش یابد.

از لایه آخر این غملیات را شروع میکنیم:

با توجه به لایه آخر و خروجی ها و خطای بدست آمده، میتوانیم وزن های w5, w6, w7, w8 را آپدیت کنیم. برای شروع نیاز است تا مشتق جزئی خطا را نسبت به این وزن ها بدست آوریم و سپس با توجه به فرمول زیر وزن ها را آپدیت میکنیم:

$$w_{new} = w_{old} - \eta * \frac{\partial E_{total}}{\partial w_{old}}$$

در اینجا η همان learning rate است که در سوال گفته شده آن را 0.3 در نظر بگیریم.

برای وزن w5 داریم:

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial w5}$$

از آنجایی که:

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

پس داریم:

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.7513649) = 0.7413649$$

همچنین:

$$out_{o1} = \frac{1}{1 + e^{-a_{o1}}}$$

پس داریم:

$$\frac{\partial out_{o1}}{\partial a_{o1}} = out_{o1}(1 - out_{o1}) = 0.7513649\ (1 - 0.7513649) = 0.1868156$$

همچنین:

$$a_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2$$

پس داریم:

$$\frac{\partial a_{o1}}{\partial w5} = out_{h1} = 0.593269$$

حال خواهیم داشت:

$$\frac{\partial E_{total}}{\partial w5} = 0.7413649 * 0.1868156 * 0.593269 = 0.082166$$

پس وزن جدید w_5 برابر خواهد بود با:

$$w_5' = 0.40 - 0.3 * 0.082166 = 0.3753502$$

برای وزن های دیگر نیز همین عملیات را باید انجام دهیم و داریم:

$$\frac{\partial E_{total}}{\partial w6} = 0.7413649 * 0.1868156 * 0.596884 = 0.082667$$

$$w_6' = 0.45 - 0.3 * 0.082667 = 0.425199$$

برای w7 و w8 باید از out_{o2} استفاده کنیم:

$$\frac{\partial E_{total}}{\partial w7} = \frac{\partial E_{total}}{\partial out_{o2}} * \frac{\partial out_{o2}}{\partial a_{o2}} * \frac{\partial a_{o2}}{\partial w7}$$

$$\frac{\partial E_{total}}{\partial out_{o2}} = -(target_{o2} - out_{o2}) = -(0.99 - 0.77292834) = 0.21707166$$

$$out_{o2} = \frac{1}{1 + e^{-a_{o2}}}$$

$$\frac{\partial out_{o2}}{\partial a_{o2}} = out_{o2}\,(1 - out_{o2}) = 0.77292834\,(1 - 0.77292834) = 0.1755101$$

$$a_{o2} = w_7 * out_{h1} + w_8 * out_{h2} + b_2$$

$$\frac{\partial a_{o2}}{\partial w7} = out_{h1} = 0.593269$$

$$\frac{\partial E_{total}}{\partial w7} = 0.21707166 * 0.1755101 * 0.593269 = 0.02260252$$

$$w_7' = 0.50 + 0.3 * 0.02260252 = 0.50678$$

$$\frac{\partial E_{total}}{\partial w8} = 0.21707166 * 0.1755101 * 0.596884 = 0.022740247$$

$$w_8' = 0.55 + 0.3 * 0.022740247 = 0.55682$$

برای وزن های w1 تا w4 باید از لایه hidden استفاده کنیم.

برای w1 داریم:

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial w1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial a_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial a_{o1}} = 0.7413649 * 0.1868156 = 0.138498$$

$$\frac{\partial a_{o1}}{\partial out_{h1}} = w5 = 0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = 0.138498 * 0.40 = 0.055399$$

$$\frac{\partial E_{o2}}{\partial out_{h1}} = \frac{\partial E_{o2}}{\partial a_{o2}} * \frac{\partial a_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o2}}{\partial a_{o2}} = \frac{\partial E_{o2}}{\partial out_{o2}} * \frac{\partial out_{o2}}{\partial a_{o2}} = -0.21707166 * 0.1755101 = -0.038098$$

$$\frac{\partial a_{o2}}{\partial out_{h1}} = w7 = 0.50$$

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = 0.055399 - 0.019049 = 0.03635$$

$$\frac{\partial out_{h1}}{\partial a_{h1}} = out_{h1}\,(1 - out_{h1}) = 0.593269\,(1 - 0.593269) = 0.241300$$

$$\frac{\partial a_{h1}}{\partial w1} = i_1 = 0.05$$

$$\frac{\partial E_{total}}{\partial w1} = 0.0363 * 0.241300 * 0.05 = 0.000438$$

$$w_1' = 0.15 - 0.3 * 0.000438 = 0.14986$$

محاسبه برای w2:

$$\frac{\partial E_{total}}{\partial w2} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial w2}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = 0.055399 - 0.019049 = 0.03635$$

$$\frac{\partial out_{h1}}{\partial a_{h1}} = 0.241300$$

$$\frac{\partial a_{h1}}{\partial w2} = i_2 = 0.10$$

$$\frac{\partial E_{total}}{\partial w2} = 0.0363 * 0.241300 * 0.10 = 0.0008759$$

$$w_2' = 0.20 - 0.3 * 0.0008759 = 0.199737$$

برای w3:

$$\frac{\partial E_{total}}{\partial w3} = \frac{\partial E_{total}}{\partial out_{h2}} * \frac{\partial out_{h2}}{\partial a_{h2}} * \frac{\partial a_{h2}}{\partial w3}$$

$$\frac{\partial E_{total}}{\partial out_{h2}} = \frac{\partial E_{o1}}{\partial out_{h2}} + \frac{\partial E_{o2}}{\partial out_{h2}}$$

$$\frac{\partial E_{o1}}{\partial out_{h2}} = \frac{\partial E_{o1}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial out_{h2}}$$

$$\frac{\partial E_{o1}}{\partial a_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial a_{o1}} = 0.7413649 * 0.1868156 = 0.138498$$

$$\frac{\partial a_{o1}}{\partial out_{h2}} = w6 = 0.45$$

$$\frac{\partial E_{o1}}{\partial out_{h2}} = 0.138498 * 0.45 = 0.0623241$$

$$\frac{\partial E_{o2}}{\partial out_{h2}} = \frac{\partial E_{o2}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial out_{h2}}$$

$$\frac{\partial E_{o2}}{\partial a_{o1}} = \frac{\partial E_{o2}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial a_{o2}} = 0.7413649 * 0.1868156 = 0.138498$$

$$\frac{\partial a_{o1}}{\partial out_{h2}} = w8 = 0.55$$

$$\frac{\partial E_{o2}}{\partial out_{h2}} = 0.0761739$$

$$\frac{\partial E_{total}}{\partial out_{h2}} = 0.0623241 + 0.0761739 = 0.138498$$

$$\frac{\partial out_{h2}}{\partial a_{h2}} = out_{h2}\,(1 - out_{h2}) = 0.596884\,(1 - 0.596884) = 0.2406134$$

$$\frac{\partial a_{h2}}{\partial w3} = i_1 = 0.05$$

$$\frac{\partial E_{total}}{\partial w3} = 0.138498 * 0.2406134 * 0.05 = 0.0016662$$

$$w_3' = 0.25 - 0.3 * 0.0016662 = 0.249500$$

برای w4:

$$\frac{\partial E_{total}}{\partial w4} = \frac{\partial E_{total}}{\partial out_{h2}} * \frac{\partial out_{h2}}{\partial a_{h2}} * \frac{\partial a_{h2}}{\partial w4}$$

$$\frac{\partial E_{total}}{\partial out_{h2}} = 0.0623241 + 0.0761739 = 0.138498$$

$$\frac{\partial out_{h2}}{\partial a_{h2}} = 0.2406134$$

$$\frac{\partial a_{h2}}{\partial w3} = i_2 = 0.10$$

$$\frac{\partial E_{total}}{\partial w4} = 0.138498 * 0.2406134 * 0.10 = 0.0033324$$

$$w_4' = 0.30 - 0.3 * 0.0033324 = 0.2990002$$

۳.

a) این مورد کمک خواهد کرد ، زیرا با وزن های رندم بهتر که به وزن خوب نزدیک باشند سریع تر، جواب را می یابیم.

b) این مورد کمک می کند. زیرا به کمک بچ های کوچک تر، نسبت به حالت اولیه می تواند سریعتر جواب را پیدا کند.

c) این روش کمک میکند . استفاده از آدام به پیدا کردن جواب در زمان کمتر معروف است.

d) این روش کمک نخواهد کرد. زمانی که وزن های رندم به صورت بهتری انتخاب شوند قاعدتا جواب بهتری هم می گیریم و 0 قرار دادن آنها مناسب نیست.

e) این روش کمک می کند. پیدا کردن learning rate مناسب که نوعی هایپرپارامتر محسوب می شود مطمئنا تاثیر گذار است.

پیاده سازی

در این بخش به پیاده سازی الگوریتم های optimization پرداخته شده است.

الگوریتم Gradient Decent

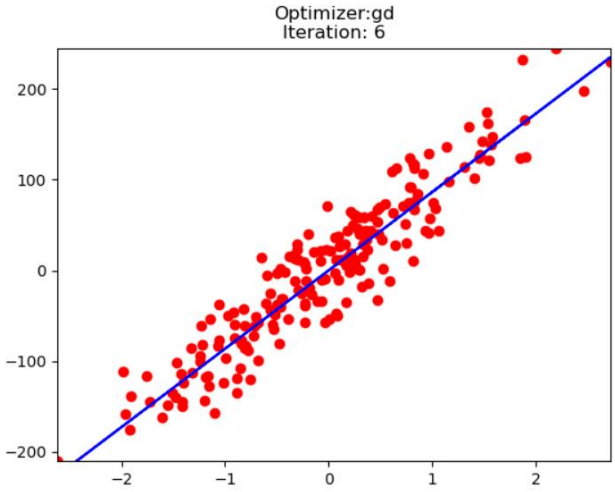
الگوریتم gradient decent را در تمرین قبل نیز پیاده سازی کرده بودیم. در اینجا کد داده شده شامل تابع compute gradient است که با استفاده از آن و فرمول زیر، الگوریتم gradient decent را پیاده سازی میکنیم:

$$w_{\text{new}} = w - \alpha \frac{\partial L}{\partial w}$$

آلفا یا همان نرخ یادگیری، در این الگوریتم به عنوان hyperparameter قرار داده میشود. نرخ یادگیری بیانگر سرعت (گام) بروزرسانی وزن‌ها است، وابسته به اینکه این عدد را چند در نظر بگیریم، نتایج متفاوتی از مدل خواهیم داشت.

با قرار داده alfa = 0.002 پس از 18 تکرار به مقدار optimum میرسد، در این تکرار مقدار loss برابر با 933 است.

اگر alfa = 0.003 باشد پس از 6 تکرار به همان مقدار loss میرسیم. از ابتدا نیز مقدار loss زیاد نیست.

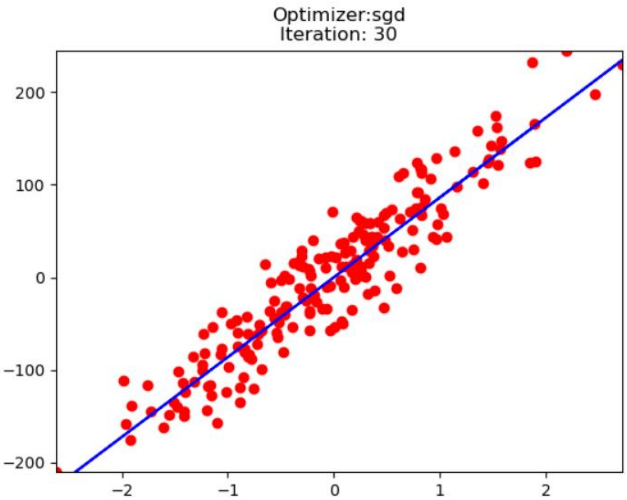


الگوریتم SGD

این الگوریتم، با توجه به الگوریتم GDعمل میکند اما تفاوت آن با gradient decent در آن است که، به صورت batch عمل میکند. یعنی به طول random تعدادی از داده ها را گرفته و روی آنها GD را اجرا میکند بعد به سراغ batch بعدی میرود و همین روند را تکرار میکند تا در نهایت به حالت Optimum برسد.

در این الگوریتم hyperparameter ها درواقع سایز batch و همان learning rate است.

با همان alpha = 0.003 و batch_size = 20 بعد از 30 تکرار به مقدار Loss برابر با 933 میرسیم:



در این تکرار ها، عملا از تکرار ۹ به بعد تغییر فاحشی در LOSS دیده نمیشود اما الگوریتم متوقف نخواهد شد و از آن به بعد کند میشود. این مشکل با استفاده از Momentum حل خواهد شد.

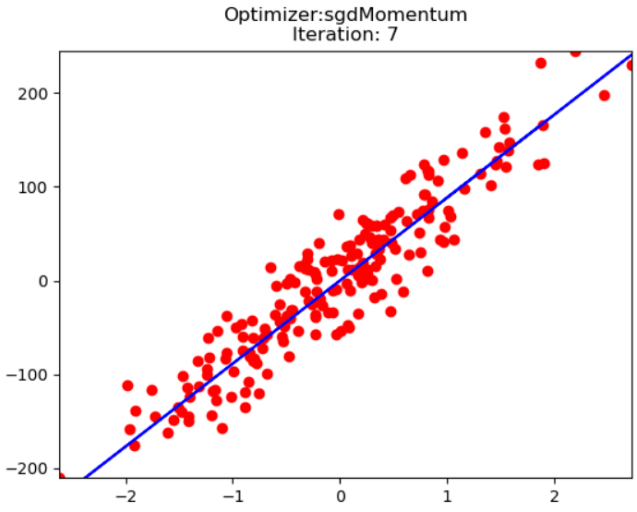
الگوریتم SGD with Momentum

این الگوریتم طبق فرمول زیر پیاده شده است:

$$w_{t+1} = w_t - \alpha m_t$$

این الگوریتم با حالت قبلی تنها در یک هایپرپارامتر تفاوت دارد. این هایپر پارامتر همان Momentum است که باعث افزایش سرعت مدل شده و اگر درست تعیین شود، از گیر کردن مدل در local optimum ها جلوگیری میکند درنتیجه باعث خواهد شد تا با برداشتن گام های بیشتر، شتاب مدل افزایش یابد و بدین ترتیب، به جواب optimum اصلی زودتر برسیم.

با قرار دادن momentum = 0.9 و بقیه هایپر پارامتر ها مانند قبل، در نتیجه مشاهده میشود که پس از 7 تکرار به جواب بهینه میرسد. همچنین در گیف بدست آمده مشاهده میشود که سرعت عملکرد مدل بسیار سریع است و پس از 7 تکرار متوقف خواهد شد.



اگر مقدار Momentum را کاهش دهیم، تعداد تکرار ها بیشتر خواهد شد.

الگوریتم AdaGrad

این الگوریتم با توجه به فرمول زیر پیاده سازی شده است:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

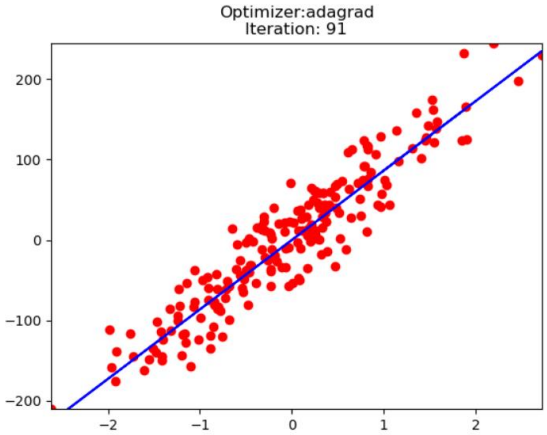
$$v_t = v_{t-1} + \left[\frac{\partial L}{\partial w_t}\right]^2$$

بردار v برابر است با که مجموع گرادیان های مجذور فعلی و گذشته است. تفاوت این الگوریتم با قبلی ها در آن است که learning rate را بر جذر بردار v تقسیم میکند.

با تعیین alpha = 40 و epsilon = 0.01 تعداد تکرار ها تا 1000 پیش رفت و الگوریتم متوقف نشد و بسیار کند به جواب رسید.

با زیادتر کردن alpha، برابر با 60 تعداد تکرار ها کمتر شد و با 580 تکرار به مقدار Loss بهینه رسید. اما باز هم در انتها کند عمل میکند.

اگر alpha = 80 باشد با 90 تکرار به جواب بهینه میرسد.



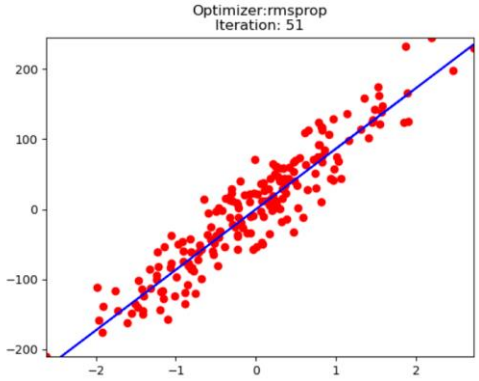
الگوریتم RMSProp:

این الگوریتم با استفاده از فرمول های زیر پیاده شده است:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}$$

$$v_t = \beta v_{t-1} + (1 - \beta)\left[\frac{\partial L}{\partial w_t}\right]^2$$

مانند AdaGrad است با این تفاوت که روش بدست آوردن بردار V آن متفاوت است. پارامتر جدید beta را نیز باید در آن تعیین کنیم. سرعت این الگوریتم از قبلی بیشتر است. با قرار دادن alpha = 50، beta = 0.1 و Epsilon = 0.01 پس از 54 تکرار به جواب بهینه میرسد.



الگوریتم Adam

این الگوریتم با استفاده از فرمول ها زیر پیاده شده است:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w_t}$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\partial L}{\partial w_t} \right]^2$$

تعیین پارامتر های beta1, beta2, alpha, epsilon حائز اهمیت است.

با قرار دادن alpha = 5، beta1 = 0.02، beta2 = 0.01 و epsilon = 0.01 پس از 17 تکرار به جواب بهینه میرسد.

اگر alpha را کم کنیم با تعداد تکرار های بیشتر و آهسته تر به جواب میرسد.