به نام خدا

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده کامپیوتر

های عصبی‌درس شبکه

تمرین شماره 1

استاد درس:

دکتر تشنه لب

اعضا گروه:

سعید صمیمی، مرجان محمدی، محمد علی اهرابی

مهرماه 1402

1-بررسی جزییات هر الگوریتم

روش SGD: یک روش گرادیان نزولی است و یک الگوریتم iterative است و با استفاده از شیب (گرادیان) سعی در کمینه‌کردن تابع (هدف) دارد و از طریق فرمول زیر وزن‌ها را به‌روز می‌کند.

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

روش Momentum: به‌جای اینکه همانند SGDاز گرادیان فعلی برای به‌روزرسانی وزن‌ها استفاده کند از روشی تحت عنوان میانگین متحرک نمایی (EMA) استفاده می‌کند که باتوجه‌به فرمول زیر، علاوه بر گرادیان همان مرحله از گرادیان مرحله قبلی نیز تأثیر می‌پذیرد. استفاده از این روش باعث می‌شود تا سرعت همگرایی بالاتر رود و همچنین می‌تواند در عدم گیرکردن در بهینه‌های محلی، جلوگیری از نوسانات و تغییرات زیاد پارامترهای شبکه نیز مؤثر باشد.

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

$$V_t = \beta V_{t-1} + (1-\beta)\frac{\partial L}{\partial w_t}$$

روش NAG: در این روش از میانگین متحرک نمایی استفاده می‌شود و مفهومی تحت عنوان گرادیان نگاشت شده استفاده می‌شود. (این مدل شبیه روش مومنتوم است.)

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

$$V_t = \beta V_{t-1} + (1-\beta)\frac{\partial L}{\partial w^*}$$

$$w^* = w_t - \alpha V_{t-1}$$

روش AdaGrad: در این روش بر روی نرخ یادگیری تمرکز شده است (از طریق تقسیم آن بر مجذور S_t) که S_t مجموع مربع گرادیان فعلی و مرحله قبلی است. از اپسیلون، باهدف جلوگیری از تقسیم بر صفر در رابطه استفاده می‌شود.

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

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

روش :RMSprop یکی دیگر از روش هایی است که از نرخ یادگیری تطبیقی استفاده می کند با این تفاوت نسبت به AdaGrad که به جای محاسبه مجموع گرادیان ها از میانگین متحرک نمایی برای محاسبه St استفاده می کند و در نهایت مقدار نرخ یادگیری را تقیسم بر مجزور Stبه اضافه مقدار کوچک اپسیلون می کند .

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

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

روش Adadelta: در این روش نیز همانند دو روش قبلی، بر بهبود نرخ یادگیری متمرکزشده است و در این روش به‌جای استفاده از نرخ یادگیری از مجذور D مجذور اضافه اپسیلون استفاده می‌شود که Dt همان میانگین متحرک نمایی دلتای وزن‌هاست (دلتای وزن‌ها تفاوت وزن فعلی از وزن قبلی است) نرخ یادگیری جدید، همانند روش‌های قبلی، بر مجذور St به‌اضافه اپسیلون تقسیم می‌شود.

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

$$D_t = \beta D_{t-1} + (1 - \beta)[\Delta w_t]^2$$

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

$$\Delta w_t = w_t - w_{t-1}$$

روش Adam: این روش، ترکیبی از روش‌های Momentum و RMSprop است که ابتدا مقدار V (میانگین متحرک نمایی گرادیان) و مقدار S (میانگین متحرک نمایی مربع گرادیان) را محاسبه کرده سپس با محاسبه $\hat{V}_t$ و $\hat{S}_t$ (که مقدارشان از طریق تقسیم S_t و V_t بر ضرایب کوچک‌تر از به دست می‌آید) رابطه گرادیان را تغییر داده و با تقسیم نرخ یادگیری بر مجذور $\hat{S}_t$ و ضرب مقدار به‌دست‌آمده در $\hat{V}_t$ وزن‌ها را آپدیت می‌کند.

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t + \epsilon}} \cdot \hat{V}_t$$

$$\hat{V}_t = \frac{V_t}{1 - \beta_1^t}$$

$$\hat{S}_t = \frac{S_t}{1 - \beta_2^t}$$

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w_t}$$

$$S_t = \beta_2 S_{t-1} + (1 - \beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2$$

روش AdaMax: یک روش تطبیقی بر پایه Adam است که در آن $\hat{V}_t$ همانند روش Adam محاسبه شده ولی در محاسبه S_t از نرم (Norm) بین‌هایت استفاده می‌شود که همان مقدار بیشینه بین ضریب S_t قبلی یا قدرمطلق گرادیان را ذخیره می‌کند و در رابطه به‌روزرسانی وزن‌ها نرخ یادگیری را بر S_t تقسیم و در $\hat{V}_t$ ضرب می‌کند.

$$w_{t+1} = w_t - \frac{\alpha}{S_t} \cdot \hat{V}_t$$

$$\hat{V}_t = \frac{V_t}{1 - \beta_1^t}$$

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w_t}$$

$$S_t = \max(\beta_2 S_t, \left|\frac{\partial L}{\partial w_t}\right|)$$

روش Nadam: مخفف Nesterov و Adam است، در این روش $\hat{V}_t$ و $\hat{S}_t$ همانند روش Adam محاسبه شده ولی رابطه به‌روزرسانی وزن‌ها متفاوت است و در محاسبه از $\hat{V}_t$ مرحله فعلی استفاده می‌کند درحالی‌که در روش Adam از $\hat{V}$ مرحله قبلی استفاده می‌شد.

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w_t} \qquad V_t = \beta_1 V_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w_t}$$

$$S_t = \beta_2 S_{t-1} + (1 - \beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2 S_t = \beta_2 S_{t-1} + (1 - \beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2$$

روش AMSGrad: یکی دیگر از روش های مبتنی بر Adam است که $\hat{S}_t$ را از طریق پیداکردن مقدار بیشترین S_t بین حالات قبلی و فعلی محاسبه کرده و در رابطه به‌روزرسانی وزن‌ها بر خلاف Adam از V_t به‌جای $\hat{V}_t$ استفاده می‌کند.

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t + \epsilon}} \cdot V_t$$

$$\hat{S}_t = \max(\hat{S}_{t-1}, S_t)$$

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w_t}$$

$$S_t = \beta_2 S_{t-1} + (1 - \beta_2)\left[\frac{\partial L}{\partial w_t}\right]^2$$

2-ایجاد داده ، ذخیره سازی و کد های مربوطه

باتوجه‌به کدهای داده شده، ابتدا هم برای sample1 و هم برای sample2 به‌صورت رندوم دادگانی تولید و در فایل ضمیمه با پسوند CSV ذخیره‌سازی شد که در اجرای کدها برای تمامی الگوریتم‌ها فراخوانی و استفاده شده است.

نام داده های مربوط به قسمت اول:

NN_HW1(data1).csv

و نام داده های مربوط به قسمت دوم:

NN_HW1(data2).csv

میباشد.

کدهای مربوط به sample1 به نام:

NN_HW1_1_Group2

و کدهای مربوط به sample2 به نام:

NN_HW1_2_Group2

و کدهای مربوط به مقاله به نام:

NN_HW1_3_Group2

ذخیره‌سازی شده‌اند.

3-بررسی MSE ها برای $ax + b$

ابتدا به بررسی خطای میانگین مربعات خطا برای کد مربوط به خط ($ax + b$) می‌پردازیم.

مقادیر MSE که دارای یک مجموعه پارامتر هستند در طی اجراهای مختلف مقادیر مختلف ولی نزدیک به هم دارند و نتایج گزارش شده در جدول زیر نیز از این قاعده مستثنی نیستند. یکی از دلایل تفاوت در MSE ها، کم‌بودن داده‌ها و ماهیت تصادفی الگوریتم‌ها است. البته که ما برای همهٔ الگوریتم‌ها از یک مجموعه‌داده ثابت استفاده کرده‌ایم.

باتوجه‌به نتایج جدول مشخص است که اکثر روش‌ها در بهترین حالت به MSE حدود 3.82 رسیده‌اند (ستون 3) و باید توجه داشت در شرایطی که پارامترها مقادیر نامناسبی دارند نتایج نیز دارای MSE بالاتری است و با کمی تغییر پارامترها می‌توان به نتایج بهتری رسید به‌عنوان‌مثال با کوچک‌تر کردن نرخ یادگیری و بیشتر کردن تعداد ایپاک ها در بسیاری از الگوریتم ها نتیجه بهتری به دست آمده است(ستون5).

Optimization Algo.	پارامترهای حالت عادی	MSE برای حالت عادی	پارمتر های کمترین MSE	کمترین MSE
SGD	Learning rate = 0.01 Epoch = 10000	4.11	Learning rate = 0.001 Epoch = 100000	3.83
Momentum	Learning rate = 0.01 Epoch = 10000 Beta = 0.9	3.88	Learning rate = 0.001 Epoch = 100000 Beta = 0.9	3.82
NAG	Learning rate = 0.01 Epoch = 10000 Beta = 0.9	3.95	Learning rate = 0.001 Epoch = 100000 Beta = 0.9	3.82
AdaGrad	Learning rate = 0.01 Epoch = 10000 Epsilon = 10^-7	19.67	Learning rate = 0.1 Epoch = 100000 Epsilon = 10^-7	3.82
RMSprop	Learning rate = 0.01 Epoch = 10000	3.87	Learning rate = 0.001 Epoch = 100000	3.82

	Beta = 0.9 Epsilon = 10^-7		Beta = 0.9 Epsilon = 10^-7	
Adadelta	Learning rate = 0.01 Epoch = 10000 Beta = 0.9 Epsilon = 10^-7	3.83	Epoch = 100000 Beta = 0.99 Epsilon = 10^-7	3.83
Adam	Learning rate = 0.01 Epoch = 10000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	3.83	Learning rate = 0.001 Epoch = 100000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	3.82
Adamax	Learning rate = 0.01 Epoch = 10000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.07	Learning rate = 0.001 Epoch = 100000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	3.82
Nadam	Learning rate = 0.01 Epoch = 10000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	3.86	Learning rate = 0.0001 Epoch = 100000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	3.82
AMSGrad	Learning rate = 0.01 Epoch = 10000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	3.84	Learning rate = 0.001 Epoch = 100000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	3.82

4-بررسی MSE ها برای $ax^2 + bx + c$

همانند قسمت قبل ابتدا به بررسی خطای میانگین مربعات خطا برای کد مربوط به خط ($ax^2 + bx + c$)می پردازیم.

همانند قسمت بالا یکی از دلایل تفاوت در MSE ها، کم بودن داده ها و ماهیت تصادفی الگوریتم ها می باشد که به دلیل درجه 2 بودن خط ، این تفاوت نسبت به حال قبل ، مقداری بیشتر شده است. در این قسمت نیز برای تمامی الگوریتم ها از داده یکسان استفاده شده است.

با توجه به نتایج جدول مشخص است که اکثر روش ها در بهترین حالت به MSE در محدوده 4.79 تا 5.33 (ستون 3) رسیده اند و همانند بالا با تغییرات در تعداد تکرار ها و میزان نرخ آموزش نتایج بهتر

شده اند(ستون 5) که البته قابلیت مقایسه با یکدیگر را ندارند.بدیهی است با توجه به ماهیت الگوریتم ها در هر بار اجرای کد ها نتایج در مقایسه با نتایج قبل مقداری متفاوت هستند.

Optimization Algo.	پارامترهای حالت عادی	MSE برای حالت عادی	پارمتر های کمترین MSE	کمترین MSE
SGD	Learning rate = 0.001 Epoch = 100000	4.88	earning rate = 0.0001 Epoch = 100000	4.79
Momentum	Learning rate = 0.001 Epoch = 100000 Beta = 0.9	5.71	Learning rate = 0.0001 Epoch = 200000 Beta = 0.9	4.82
NAG	Learning rate = 0.001 Epoch = 100000 Beta = 0.9	5.08	Learning rate = 0.001 Epoch = 200000 Beta = 0.9	4.83
AdaGrad	Learning rate = 0.001 Epoch = 100000 Epsilon = 10^-7	327.289	Learning rate = 0.01 Epoch = 400000 Epsilon = 10^-7	4.78
RMSprop	Learning rate = 0.001 Epoch = 100000 Beta = 0.9 Epsilon = 10^-7	498.082	Learning rate = 0.01 Epoch = 100000 Beta = 0.9 Epsilon = 10^-7	195.182
Adadelta	Learning rate = 0.001 Epoch = 100000 Beta = 0.9 Epsilon = 10^-7	4.79	Learning rate = 0.01 Epoch = 100000 Beta = 0.99 Epsilon = 10^-7	4.78
Adam	Learning rate = 0.001 Epoch = 100000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.78	Learning rate = 0.001 Epoch = 200000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.78
Adamax	Learning rate = 0.001 Epoch = 100000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.78	Learning rate = 0.001 Epoch = 400000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.78
Nadam	Learning rate = 0.001 Epoch = 100000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.78	Learning rate = 0.001 Epoch = 400000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.78
AMSGrad	Learning rate = 0.001 Epoch = 100000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.78	Learning rate = 0.001 Epoch = 400000 Beta1 = 0.9 Beta2 = 0.999 Epsilon = 10^-7	4.78

5-مقاله

بررسی روش ارائه شده مقاله ADAM ACCELERATED(Aadam)

این مقاله روشی را در جهت بهبود روش Adam ارائه کرده است. ایده اصلی پشت AAdam سرعت بخشیدن به پیشرفت در امتداد ابعادی است که گرادیان به طور مداوم در یک جهت قرار دارد. رابطه روش پیشنهادی به صورت زیر است

$$\begin{aligned} \theta_{n+1} &= \theta_n - (\eta \frac{\beta_1}{\sqrt{\hat{v_n} + \epsilon}} \hat{m}_n + d) \\ d &= \Delta\theta_{n-1} * sign(\nabla_{\theta_n} J(\theta)) * (1 - \beta_1) \\ \hat{m}_n, \hat{v_n} &= \text{Adam Parameters} \\ \Delta\theta_{n-1} &= \text{Last update step.} \end{aligned}$$

اختالف این روش با Adam در افزودن ضریب β_1در صورت کسر وزن دهی به نرخ آموزش و همچین افزودن پارامتر d است.دیگر پارامترها،همانند پارامترهای Adam هستند.

نوآوری این روش در افزودن پارامتر d است.این پارامتر شامل مقدار تغییر وزن در دوره قبل $(\Delta\theta_{n-1})$ و همچنین شیب تغیرات گرادیان در مرحله فعلی $(sign(\nabla_{\theta_n} J(\theta))$است.

نتیجه مقایسه Adam و AAdam

برای هر دو حالت،از داده های یکسان و پیکربندی ثابتی استفاده شده است تا مقایسه به‌صورت عادلانه صورت گیرد.

η (ax+b) = 0.01

η (ax2+bx+c) = 0.001

epochs(ax+b) = 10000

epochs(ax2+bx+c) = 100000

ε = 10−8

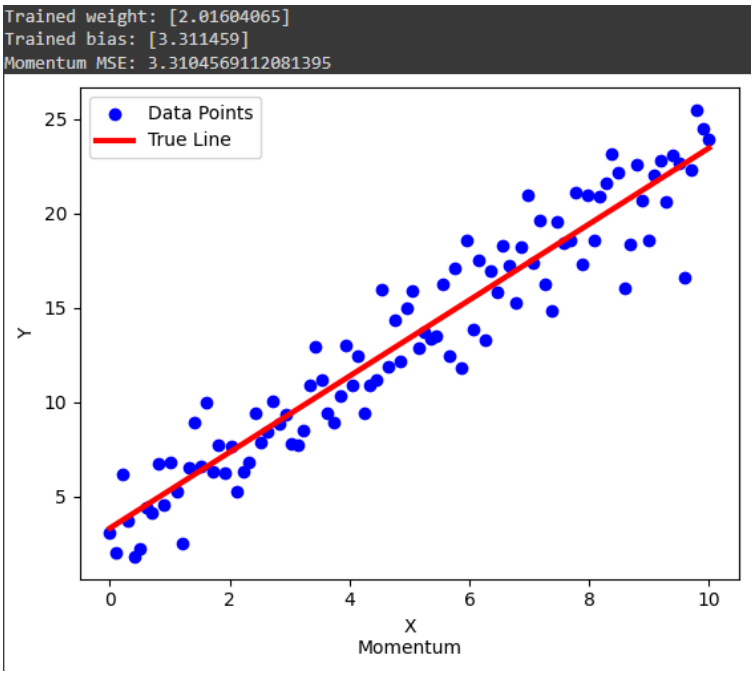
β1 = 0.9

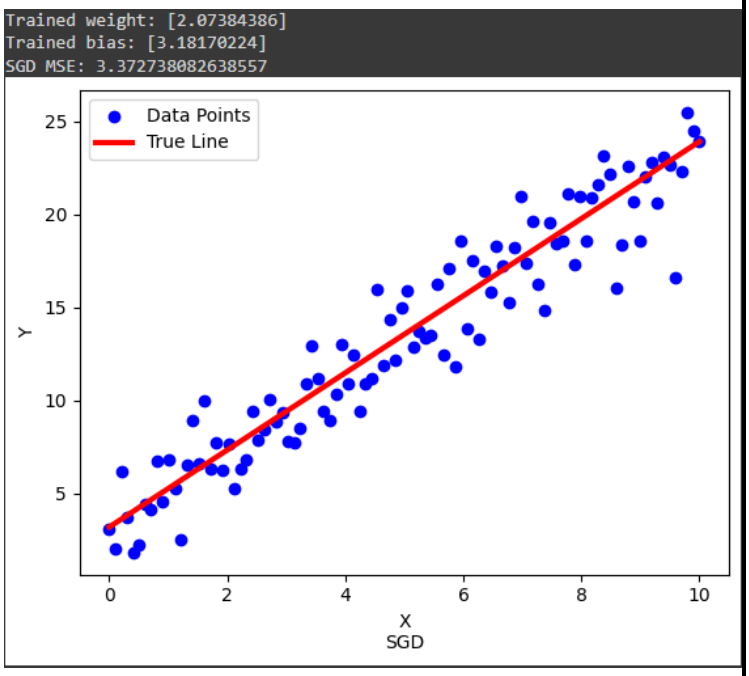
β2 = 0.999

	ax+b(error)	ax2+bx+c(error)
Adam	3.643564445839073	4.394243755772239
AAdam	3.308915538304003	4.784867308372225

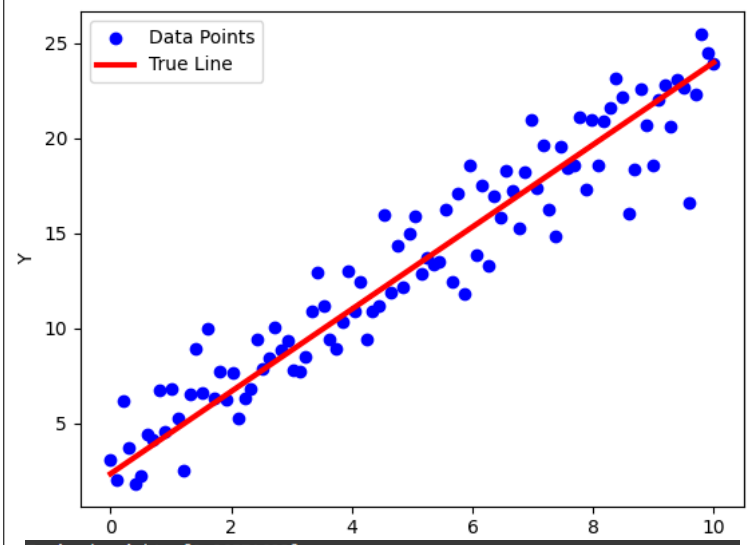
نتایج تقریباً مساوی هستند؛ اما ازآنجاکه داده‌های ما ساده هستند، شاید نتوان تأثیر روش ارائه شده را به‌خوبی دید و اختلاف آن را با روش Adam بررسی کرد. روش ارائه شده در مجموع روش خوبی است و نتایج خوبی را ارائه کرده و دقت مدل را خراب نکرده است.

6-تصاویر نتایج بدست آمده

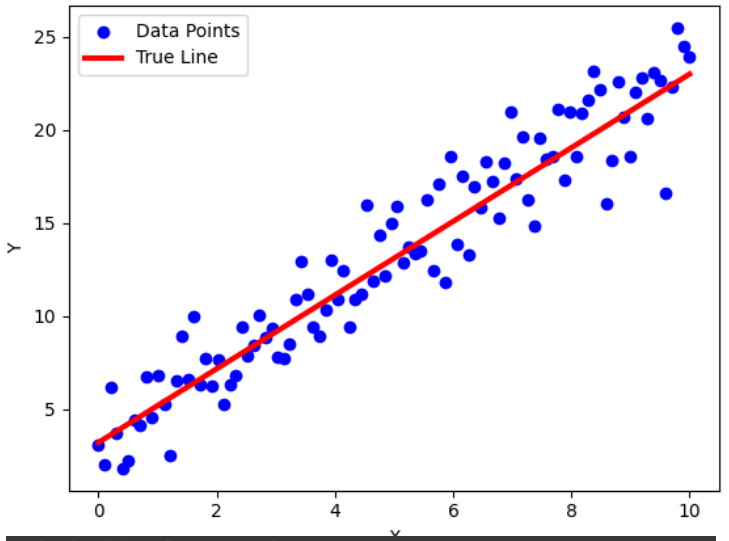




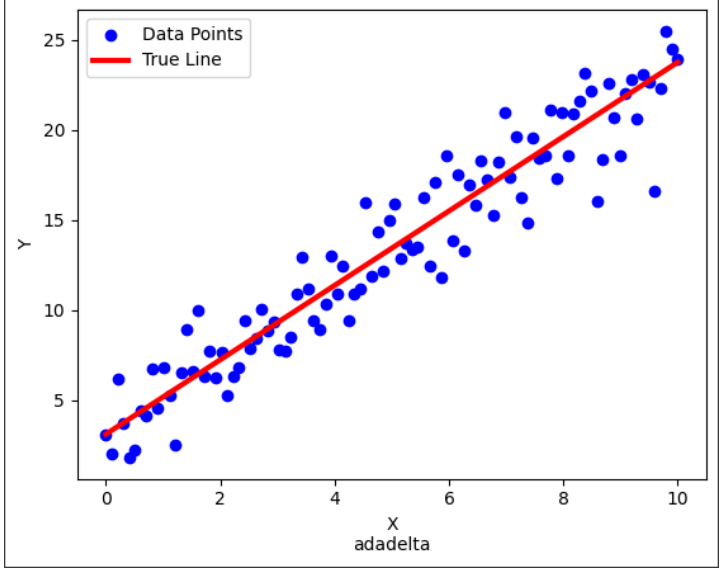
```
Trained weight: [2.16605267]
Trained bias: [2.33433031]
adaGrad MSE: 3.5131791764898295
```



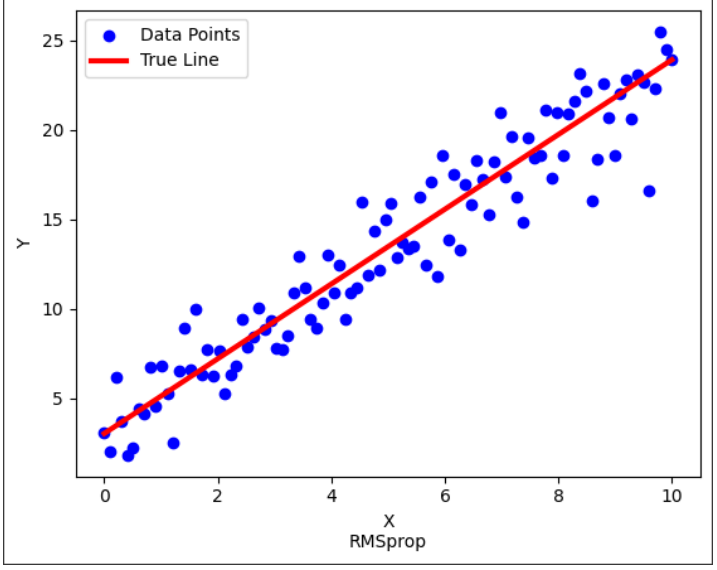

```
Trained weight: [1.98033174]
Trained bias: [3.20043936]
NAG MSE: 3.3813038427736473
```



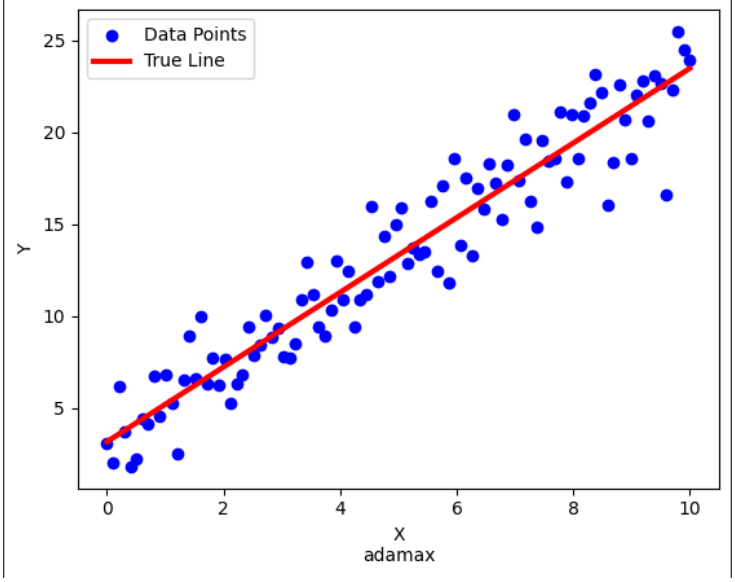

```
Trained weight: [2.06571693]
Trained bias: [3.10543498]
adadelta MSE: 3.3313593542096918
```



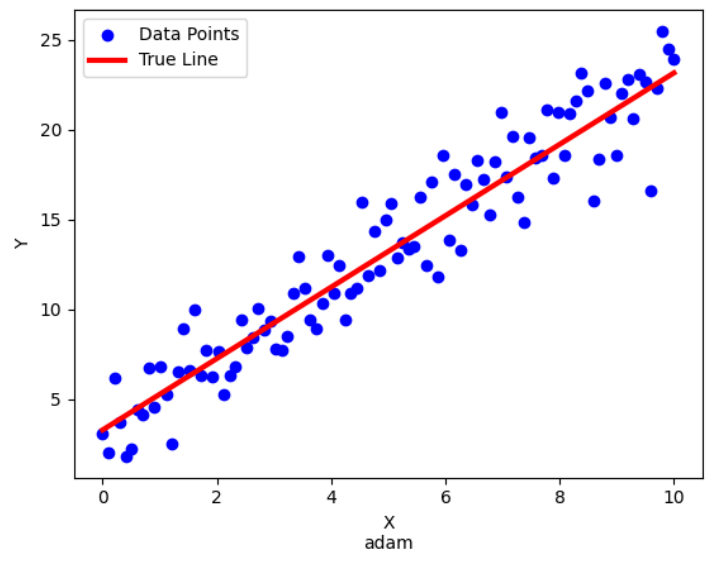

```
Trained weight: [2.09009675]
Trained bias: [3.02319319]
RMSprop MSE: 3.362853168494329
```



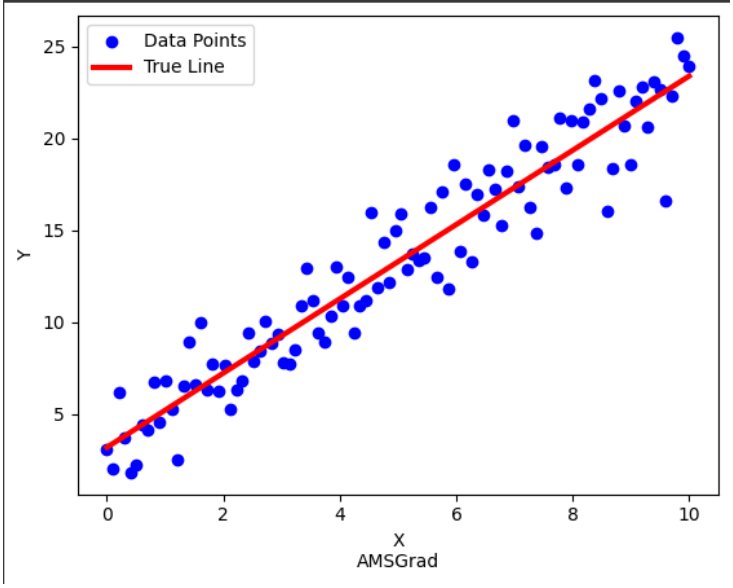

```
Trained weight: [2.03252041]
Trained bias: [3.15847875]
adamax MSE: 3.3089001716519193
```



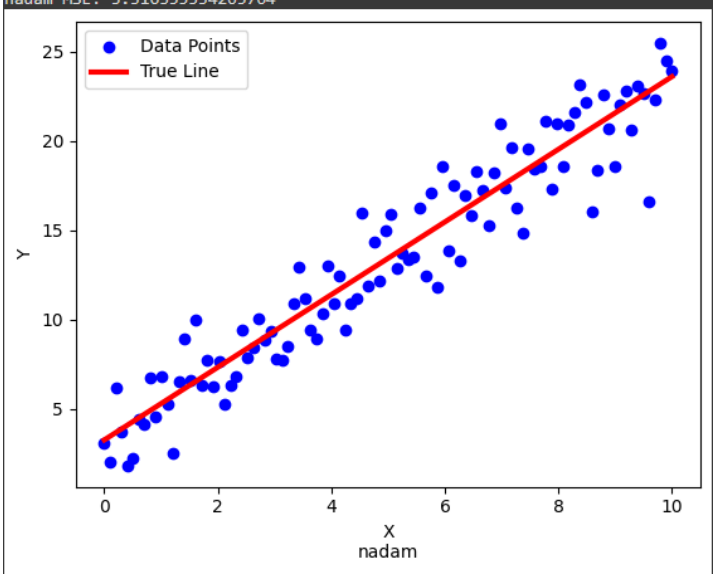

```
Trained weight: [1.98880192]
Trained bias: [3.27661229]
adam MSE: 3.3326778962784327
```



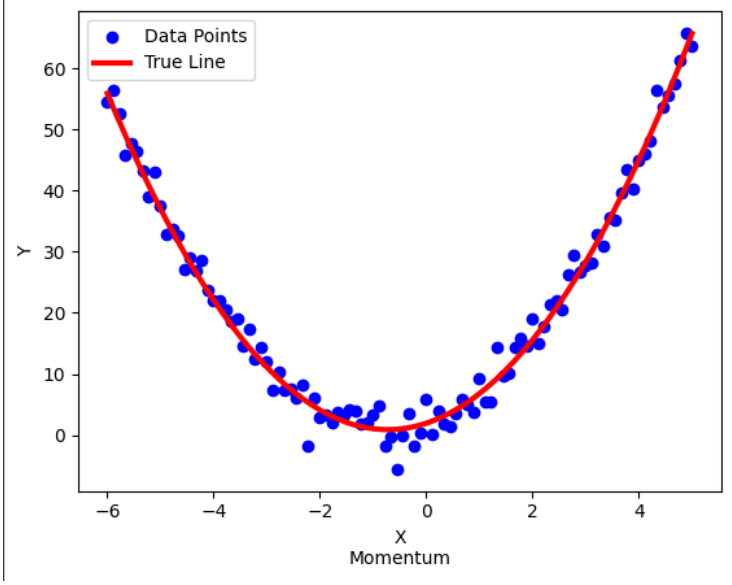
```
Trained weight: [2.02168548]
Trained bias: [3.17992141]
AMSGrad MSE: 3.310678703351966
```



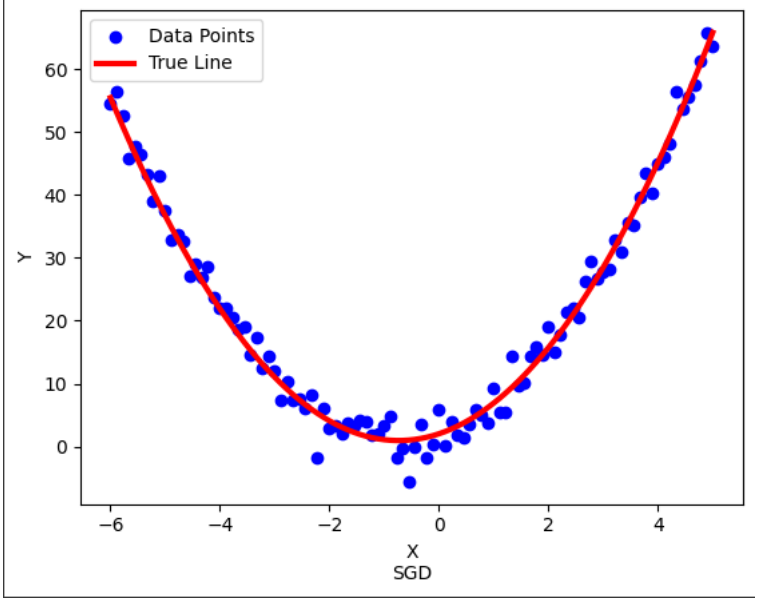

```
Trained weight: [2.0356881]
Trained bias: [3.25033822]
nadam MSE: 3.316355534265764
```



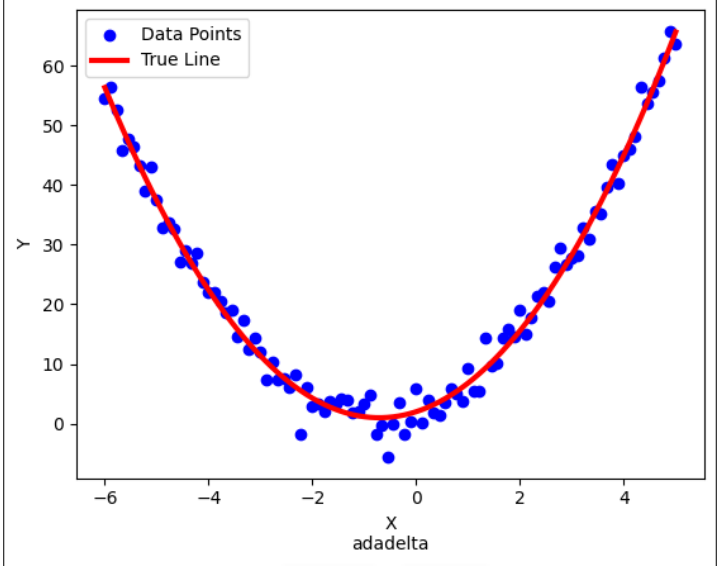

```
Trained weight: [1.97813329] [2.86958224] [1.92829838]
Momentum MSE: 4.819259197092259
```



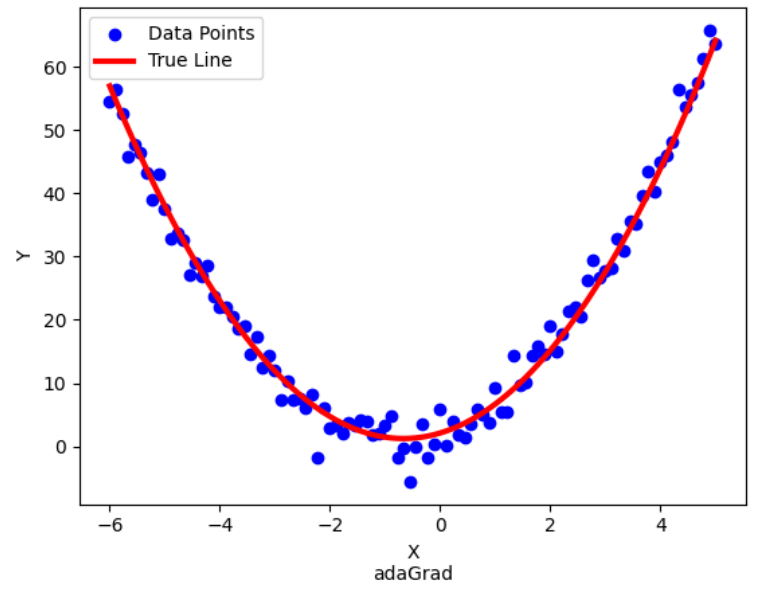

```
Trained weight: [1.97083588] [2.91361421] [2.01533301]
SGD MSE: 4.8869788564077545
```



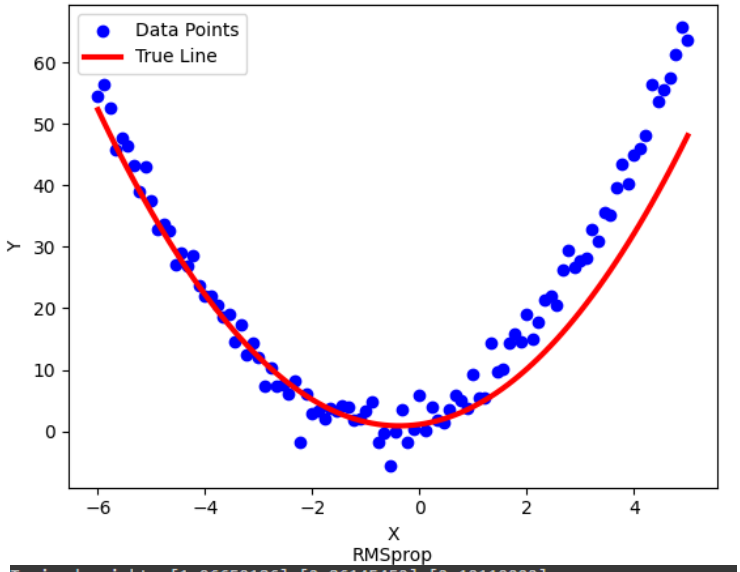

```
Trained weight: [1.98198816] [2.82925962] [1.98772111]
adadelta MSE: 4.792359831323971
```



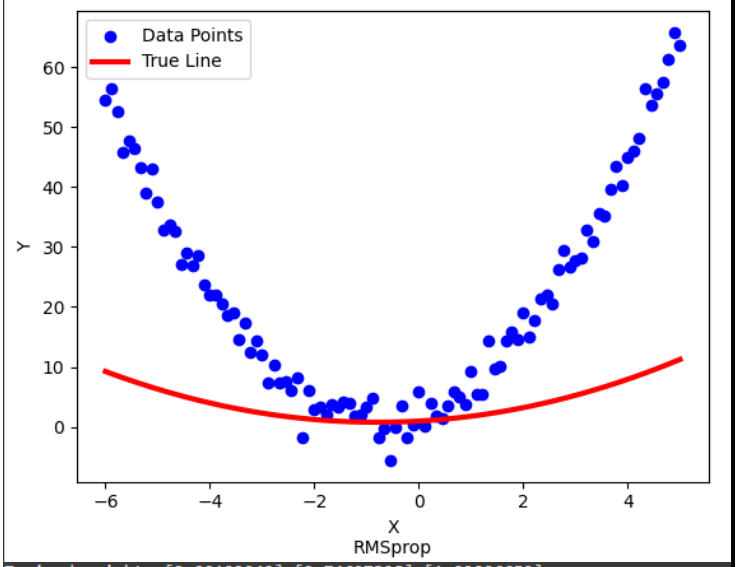

```
Trained weight: [1.9607132] [2.61616364] [2.0968034]
adaGrad MSE: 5.333458612550122
```



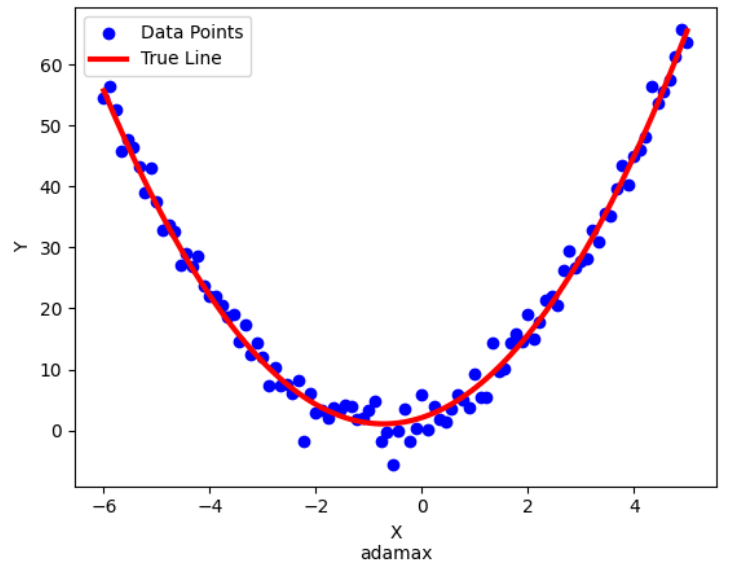
```
Trained weight: [1.63068405] [1.24490919] [1.10704988]
RMSprop MSE: 46.58863673868725
```



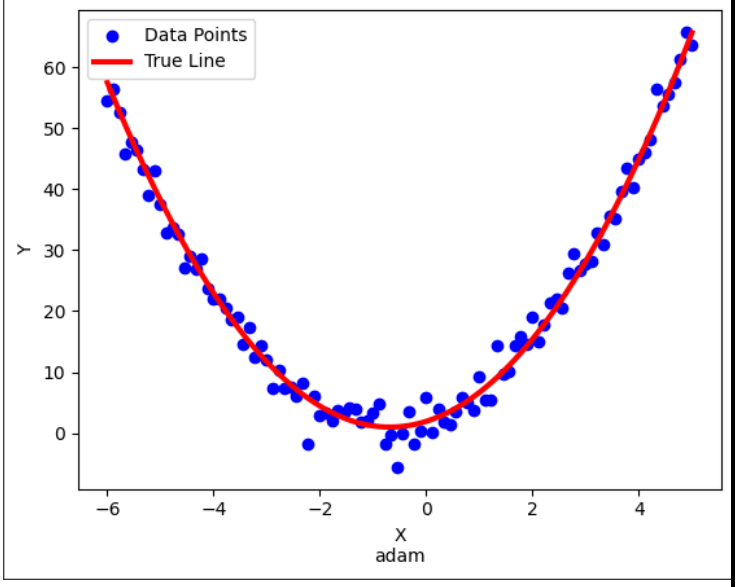

```
Trained weight: [0.31241714] [0.49493631] [0.95651036]
RMSprop MSE: 553.5854734760562
```



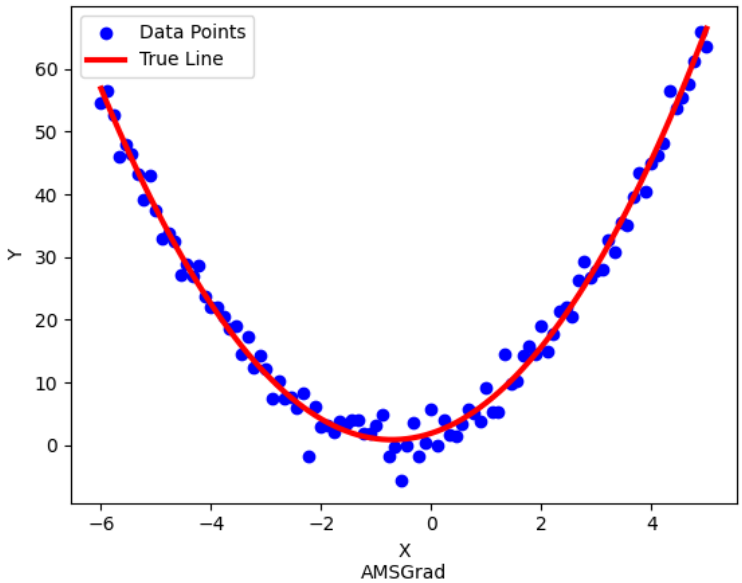

```
Trained weight: [1.96650186] [2.86145452] [2.10119002]
adam MSE: 4.8372677145427465
```



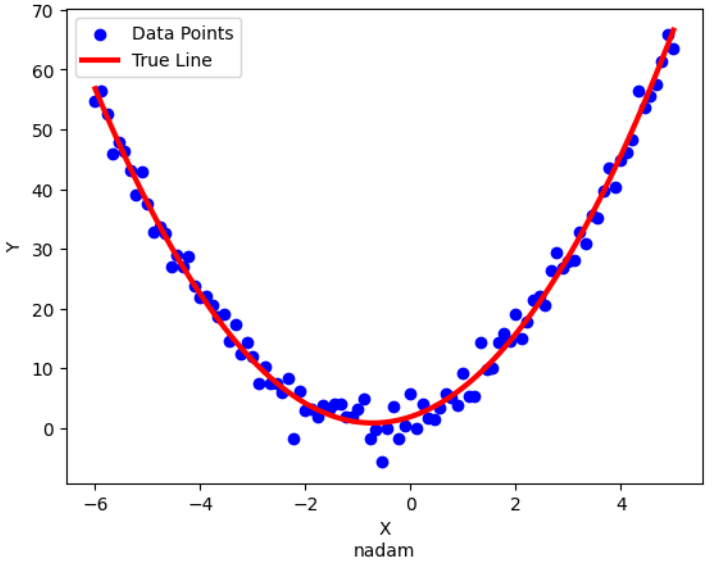

```
Trained weight: [2.00192942] [2.74627398] [1.92006659]
adam MSE: 4.986197432057369
```




```
Trained weight: [2.00504867] [2.87371965] [1.90282248]
AMSGrad MSE: 4.813263533241096
```




```
Trained weight: [2.00774868] [2.89812619] [1.89774496]
nadam MSE: 4.826744452439769
```



```
Dataset 1 - Trained w: [2.01996525] Trained b: [3.20913427] Aadam_error: 3.308915538304003
Dataset 2 - Trained w: [1.98844283] Trained b: [2.87695614] Trained c: [1.94335806] Aadam_error: 4.784867308372225
```