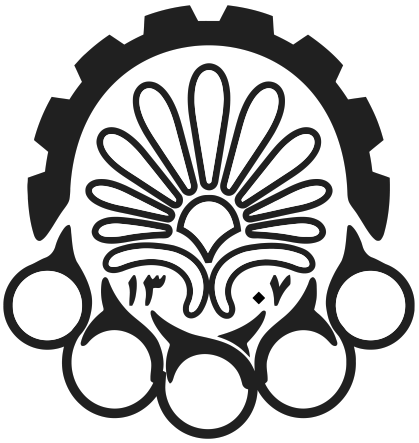
شبکه‌های عصبی و یادگیری عمیق
دکتر صفابخش



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

تمرین پنجم
شبکه‌های RNN

۱۱ خرداد ۱۴۰۳

شبکه‌های عصبی و یادگیری عمیق

تمرین پنجم

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

دانشکده مهندسی کامپیوتر

سوال اول - نظری

به سوالات زیر بصورت خلاصه و برای هر یک حداکثر در سه بند پاسخ دهید:

۱. به‌طور کلی بهینه‌سازها[۱] (نظیر ADAM) به دنبال یافتن وزن‌های شبکه‌های عصبی هستند بطوریکه توابع هزینه[۲] کمینه شود. مشتق‌پذیر بودن توابع یاد شده چه تاثیری در بهینه‌ساز دارد؟ اگر مشتق‌پذیر نباشد، چه رویکردهایی برای بهینه‌سازی آن وجود دارد؟ یک مورد را به دلخواه توضیح دهید.

پاسخ

بهینه‌سازهایی مانند ADAM از گرادیان توابع هزینه برای به‌روزرسانی وزن‌های شبکه استفاده می‌کنند. مشتق‌پذیر بودن این توابع به معنای وجود گرادیان است که به بهینه‌سازها کمک می‌کنند جهت حرکت به سمت مینیمم سراسری را پیدا کنند. بدون مشتق‌پذیری، تعیین دقیق جهت و میزان تغییر وزن‌ها دشوار می‌شود.
در صورتی که تابع هزینه مشتق‌پذیر نباشد، روش‌های دیگری نظیر الگوریتم‌های مبتنی بر مشتقات تقریبی یا تکنیک‌های بهینه‌سازی بدون مشتق مانند الگوریتم ژنتیک یا بهینه‌سازی ازدحام ذرات (Particle Swarm Optimization) مورد استفاده قرار می‌گیرند.
بهینه‌سازی ازدحام ذرات (PSO) یک روش الهام گرفته از طبیعت است که بدون نیاز به مشتق تابع کار می‌کند. این الگوریتم با استفاده از حرکت ذرات در فضای جستجو و به‌روزرسانی موقعیت‌های آنها بر اساس بهترین موقعیت‌های خود و همسایگانشان، به سمت بهینه و پیدا کردن مینیمم سسراسری حرکت می‌کند.

*

References

[1] I. Goodfellow, Y. Bengio & A. Courville, (2016). Deep Learning. MIT Press (Link)

[2] Rios, Luis Miguel, and Nikolaos V. Sahinidis. ”Derivative-free optimization: a review of algorithms and comparison of software implementations.” Journal of Global Optimization 56.3 (2013): 1247-1293.

۲. محدب[۳] بودن توابع به چه معناست و چرا مطلوب است که در بهینه‌سازی، توابع هزینه محدب باشد؟ اگر محدب نباشد، چگونه می‌توان آن را بهینه نمود؟

[۱] Optimizer
[۲] Loss Functions
[۳] Convex

پاسخ

یک تابع محدب است اگر خط واصل بین هر دو نقطه از نمودار آن تابع، همیشه بالای نمودار تابع قرار گیرد. این ویژگی باعث می‌شود که هر مینیمم محلی، مینیمم سراسری نیز باشد، که جستجو برای یافتن نقطه بهینه را آسان می‌کند.
توابع محدب از این جهت برای ما مفید هستند چون تضمین می‌کنند که بهینه‌سازها می‌توانند به راحتی و با اطمینان به نقطه بهینه سراسری برسند، بدون اینکه در مینیمم‌های محلی گیر کنند. این ویژگی فرآیند بهینه‌سازی را کارآمدتر و قابل اعتمادتر می‌سازد.
در صورت محدب نبودن توابع هزینه، می‌توان از تکنیک‌هایی نظیر الگوریتم‌های تصادفی (Stochastic Algorithms)، چندین شروع تصادفی (Multiple Random Starts)، و روش‌های بهینه‌سازی مبتنی بر شبیه‌سازی (Simulated Annealing) برای جستجوی بهینه سراسری استفاده کرد.

*

References

[1] S. Boyd & L. Vandenberghe, (2004). Convex Optimization. Cambridge University Press. (Link)

[2] J. Nocedal, & S. j. Wright, (2006). Numerical Optimization. Springer. (Chapters on non-convex optimization)

۳. الگوریتم بهینه‌سازی نیوتن را مطالعه کرده و آن را با نزول در راستای گرادیان[۴] مقایسه کنید. در چه نوع مسائلی استفاده از الگوریتم نیوتن ارجحیت دارد؟

پاسخ

الگوریتم نیوتون، از مشتق دوم تابع هزینه (hessian) برای بهبود به‌روزرسانی وزن‌ها استفاده می‌کند. بروزرسانی وزن‌ها با استفاده از فرمول زیر انجام می‌شود که H همان ماتریس hessian است.

$$\theta_{\text{new}} = \theta_{\text{old}} - H^{-1}\nabla L(\theta_{\text{old}})$$

نزول گرادیان فقط از مشتق مرتبه اول استفاده می‌کند و به‌روزرسانی وزن‌ها را با توجه به جهت و میزان مشتق انجام می‌دهد. الگوریتم نیوتن به دلیل استفاده از اطلاعات مشتق مرتبه دوم می‌تواند به سرعت به نقطه بهینه نزدیک شود، اما محاسبه و بدست آوردن وارون ماتریس hessian هزینه‌بر است.
الگوریتم نیوتن برای مسائلی با تعداد پارامترهای کم و توابع ساده، که محاسبه و وارون‌سازی ماتریس hessian را دشوار نکند، مناسب‌تر است. این الگوریتم در مسائلی که به دقت بالاتر و همگرایی سریع‌تر نیاز داریم، ارجحیت دارد.

[۴] Gradient Descent

پاسخ

*

References

[1] I. Goodfellow, Y. Bengio & A. Courville, (2016). Deep Learning. MIT Press, (Chapter on Optimization) (Link)

[2] J. Nocedal, & J. S. Wright, (2006). Numerical Optimization. Springer. (Chapters on second-order methods)

۴. ضمن مطالعه کلی الگوریتم AdaGrad، بیان کنید که چگونه می‌توان از آن برای بهینه ساختن نرخ یادگیری[۵] بهره گرفت.

فرض کنید مسئله‌ی دسته بندی دودویی بحرانی بودن/نبودن شرایط یک کارگاه صنعتی بر اساس اطلاعاتی محیطی آن را در اختیار دارید که داده‌های دما، رطوبت، فشار و ذرات معلق بر اساس سنسورهای نصب شده در هر یک ثانیه ارسال می‌گردد. شما بایستی با در نظر گرفتن دنباله‌ای از داده‌های ارسالی بتوانید تشخیص دهید که شرایط بحرانی است یا خیر.

پاسخ

AdaGrad (Adaptive Gradient) الگوریتم بهینه‌سازی‌ای است که نرخ یادگیری را به صورت داینامیک و متناسب با تاریخچه گرادیان تنظیم می‌کند. این الگوریتم با تقسیم نرخ یادگیری اولیه بر مجموع ریشه مربع گرادیان‌های قبلی، نرخ یادگیری را برای هر پارامتر به صورت جداگانه تنظیم می‌کند.
در AdaGrad، هر پارامتر نرخ یادگیری خاص خود را دارد که با توجه به میزان نوسانات آن پارامتر تنظیم می‌شود. این کار به الگوریتم اجازه می‌دهد تا در مسیرهای با گرادیان زیاد نرخ یادگیری را کاهش دهد و در مسیرهای با گرادیان کم آن را افزایش دهد، که منجر به بهینه‌سازی دقیق‌تر و جلوگیری از نوسانات شدید می‌شود.

*

References

[1] J. Duchi, E. Hazan, & Y. Singer, (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(Jul), 2121-2159.

[2] S. Ruder, (2016). An overview of gradient descent optimization algorithms. (Link)

۵. یک شبکه‌ی بازخوردی Elman که با دولایه‌ی مخفی که به ترتیب سه و دو نورون تعبیه شده است، طراحی نمایید و تعداد وزن‌های مورد نیاز برای یادگیری در این شبکه را با بیان علت محاسبه نموده و ابعاد تمامی بردارهای (Vectors & Tensors) مشاهده شده در شبکه (ورودی‌ها/میانی‌ها/خروجی‌ها) را با محاسبات و استدلال نمایش دهید. انتظار می‌رود که شما بتوانید سیر تغییرات ابعاد بردارها و چگونگی آن را نشان دهید؛ مثلا شکل بردار ورودی برای یک دسته (batch) چگونه تعیین می‌شود و تا رسیدن به خروجی شکل آن چرا و چگونه تغییر پیدا کرده است و با چه وزن‌هایی متاثر شده است.

[۵] Learning Rate

پاسخ

طبق صورت مسئله، فرضیات و نوتیشن‌های زیر را درنظر می‌گیریم:

(آ) ورودی $x(t)$ با ابعاد n_x است

(ب) لایه مخفی اول ($h_1(t)$) دارای ۳ نورون با ابعاد ۳

(ج) لایه مخفی دوم ($h_2(t)$) دارای ۲ نورون مخفی با ابعاد ۲

(د) خروجی $y(t)$ با ابعاد n_y

در مرحله اول تعداد وزن‌های لایه‌های مختلف را با بیان جزئیات محاسبه می‌کنیم:

(آ) **لایه ورودی به لایه مخفی اول:**

- وزن‌های بین ورودی و لایه مخفی اول: $\mathbf{W}_{xh1}$ با ابعاد $3 \times n_x$
- وزن‌های بازگشتی از خروجی لایه مخفی اول به خودش: $\mathbf{W}_{h1h1}$ با ابعاد 3×3
- بایاس‌های لایه مخفی اول: $\mathbf{b}_{h1}$ با ابعاد ۳

$$\text{تعداد وزن‌های لایه اول} = 3 \times n_x + 3 \times 3 + 3$$

(ب) **لایه مخفی اول به لایه مخفی دوم:**

- وزن‌های بین لایه مخفی اول و دوم: $\mathbf{W}_{h1h2}$ با ابعاد 2×3
- وزن‌های بازگشتی از خروجی لایه مخفی دوم به خودش: $\mathbf{W}_{h2h2}$ با ابعاد 2×2.
- بایاس‌های لایه مخفی دوم: $\mathbf{b}_{h2}$ با ابعاد ۲

$$\text{تعداد وزن‌های لایه دوم} = 2 \times 3 + 2 \times 2 + 2 = 12$$

(ج) **لایه مخفی دوم به لایه خروجی:**

- وزن‌های بین لایه مخفی دوم و خروجی: $\mathbf{W}_{h2y}$ با ابعاد $n_y \times 2$
- بایاس‌های لایه خروجی: $\mathbf{b}_y$ با ابعاد n_y

$$\text{تعداد وزن‌های لایه خروجی} = n_y \times 2 + n_y$$

در ادامه ابعاد بردارها در شبکه را محاسبه می‌کنیم:

(آ) **ورودی:**

- بردار ورودی $\mathbf{x}(t)$ با ابعاد n_x

پاسخ

(آ) **لایه مخفی اول:**

- ورودی به لایه مخفی اول:

$$\mathbf{h}_1(t) = \sigma(\mathbf{W}_{xh1}\mathbf{x}(t) + \mathbf{W}_{h1h1}\mathbf{h}_1(t-1) + \mathbf{b}_{h1})$$

- ابعاد $\mathbf{h}_1(t)$: ۳

(ب) **لایه مخفی دوم:**

- ورودی به لایه مخفی دوم:

$$\mathbf{h}_2(t) = \sigma(\mathbf{W}_{h1h2}\mathbf{h}_1(t) + \mathbf{W}_{h2h2}\mathbf{h}_2(t-1) + \mathbf{b}_{h2})$$

- ابعاد $\mathbf{h}_2(t)$: ۲.

(ج) **خروجی:**

- خروجی:

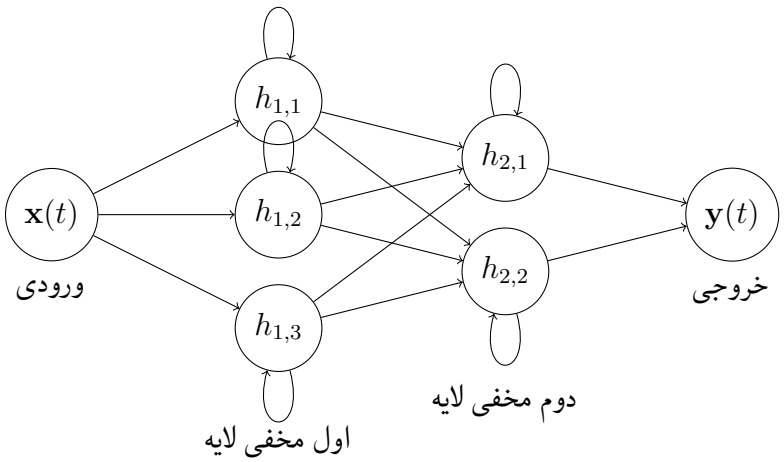
$$\mathbf{y}(t) = \mathbf{W}_{h2y}\mathbf{h}_2(t) + \mathbf{b}_y$$

- ابعاد $\mathbf{y}(t)$: n_y.

درنهایت با ترکیب همه وزن‌ها و بایاس‌ها تعداد کل وزن‌های شبکه به‌صورت زیر می‌شود:

$$\text{تعداد کل وزن‌ها} = (3 \times n_x + 3 \times 3 + 3) + (2 \times 3 + 2 \times 2 + 2) + (n_y \times 2 + n_y)$$
$$= (3n_x + 9 + 3) + (6 + 4 + 2) + (2n_y + n_y)$$
$$= 3n_x + 3n_y + 24$$

درنهایت دیاگرام شبکه طراحی شده به‌صورت زیر است:



شکل ۱: دیاگرام شبکه Elman با دو لایه مخفی

سوال دوم - عملی

آیا تا کنون به روند عملکرد بهینه‌سازها فکر کرده‌اید؟ آیا می‌توان آن را یک شبکه‌ی بازرخدادی در نظر گرفت؟ در این پروژه هدف طراحی و پیاده‌سازی یک بهینه‌ساز می‌باشد. برای درک بهتر، توضیحات ریاضیاتی زیر داده می‌شود.
روند یادگیری پارامترهای (θ) یک شبکه عمیق (f) با الگوریتم های مرسوم نزول در راستای گرادیان (نظیر SGD) را می‌توان به ازای ورودی‌های آموزشی x بصورت رابطه‌ی (۱) در نظر گرفت:

$$\theta_{i+1} = \theta_i - \alpha \nabla f(x; \theta_i) \quad (۱)$$

حال اگر فرض شود که به جای نرخ یادگیری ثابت α از یک تابع (شبکه‌ی عمیق) نظیر g با پارامترهای قابل یادگیری ϕ استفاده کنیم، می‌توان رابطه‌ی (۱) را بصورت رابطه‌ی (۲) بازنویسی نمود:

$$\theta_{i+1} = \theta_i + g(\nabla f(x; \theta_i); \phi) \quad (۲)$$

در نهایت می‌توان پارامتر θ_i را نیز به عنوان یک ورودی دیگر به g در نظر گرفت و رابطه‌ی (۲) را نیز بصورت زیر بازنویسی کرد:

$$\theta_{i+1} = g(\nabla f(x; \theta_i), \theta_i; \phi) \quad (۳)$$

حال می‌توان نتیجه گرفت که اگر تابع g را یک شبکه‌ی بازرخدادی (نظیر LSTM یا GRU) در نظر گرفت، امکان ارئه‌ی یک بهینه‌ساز وجود دارد، که کل فرایند یاد شده را می‌توان با دو حلقه (بیرونی[۶] و درونی[۷]) انجام داد که معماری کلی آن در ادامه ضمیمه شده است. پیمایش یکبار حلقه‌ی بیرونی معادل است با یک تکرار (Epoch) برای آموزش شبکه‌ی g و پیمایش یکبار حلقه درونی معادل است با تولید یک داده آموزشی برای شبکه‌ی g.
در این سوال هدف طراحی و پیاده سازی یک بهینه‌ساز بر اساس توضیحات فوق می‌باشد:

۱. مجموعه داده اول را به عنوان مجموعه آموزشی در نظر بگیرید، آن را نمایش داده و پس درهم سازی به ۵۰ زیر مجموعه تقسیم کنید بطوریکه در هر مجموعه داده از هر کلاس به تعداد برابر نمونه موجود باشد و انتخاب نمونه‌ها نیز با احتمال یکنواخت صورت گرفته باشد. حال، یک شبکه‌ی MLP دلخواه طراحی نمایید و آن را f بنامید که $f_1, f_2, \ldots$ شبکه‌های MLP با معماری یکسان هستند و صرفا مقادیردهی اولیه‌ی آن‌ها هر بار متفاوت است.

پاسخ

مجموعه داده ورودی (`dataset_1.csv`) متشکل از نقاط `x`، `y` و `label` ای است که جدا کننده دو کلاس داده‌ها است. ابعاد این دیتاست $(100000, 3)$ است.

[۶] Outer loop
[۷] Inner loop

پاسخ

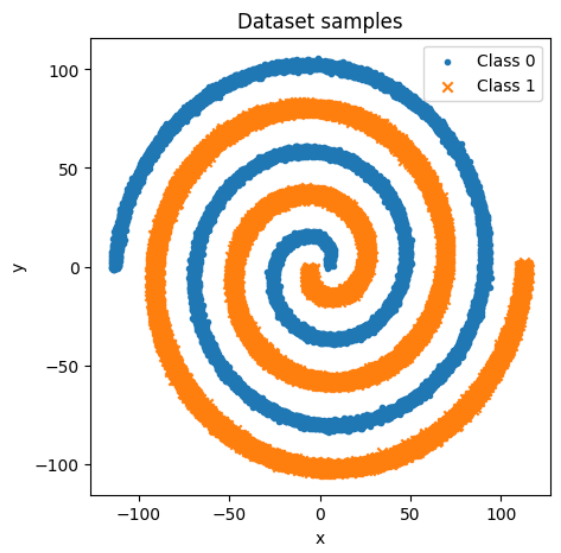
shape of data frame is: (100000, 3)

	x	y	label
45089	73.46281	62.60531	0.0
81101	-64.70864	-37.63372	0.0
9474	-69.99582	-4.58385	0.0
42881	-20.40946	102.57979	0.0
45989	-40.61369	16.67236	1.0
...	...	...	...
92339	-18.14189	13.13609	0.0
36902	46.58511	-43.49732	1.0
13738	40.37781	63.62238	1.0
37846	67.03522	22.81052	1.0
18939	34.80919	68.97051	1.0

100000 rows × 3 columns

شکل ۲: خروجی دیتاست `dataset_1.csv`

رسم گرافیکی دیتاست ورودی به‌صورت زیر است:



شکل ۳: دیتاست `dataset_1.csv`

سپس طبق صورت مسئله، دیتاست ورودی را به ۵۰ زیردیتاست تقسیم می‌کنیم. سایز زیردیتاست های تولید شده $(2000, 3)$ است. در ادامه خروجی این تقسیم دیتاست آورده شده است:

پاسخ

```
Subset 1:
tensor([[ 90.3425, -13.8065,   0.0000],
        [ 12.2387,  99.9308,   0.0000],
        [-62.3302, -34.1428,   0.0000],
        ...,
        [ 55.9775, -35.9741,   1.0000],
        [-73.2319, -62.9040,   1.0000],
        [ 16.9411, -58.0383,   1.0000]], device='cuda:0')
------------------------------------------------
Subset 2:
tensor([[ 47.4137,  85.5787,   0.0000],
        [ 22.5193,  98.6952,   0.0000],
        [ 46.2943,  -0.9892,   0.0000],
        ...,
        [-34.0279, -94.8296,   1.0000],
        [-10.1396,  78.8019,   1.0000],
        [100.5447, -46.0529,   1.0000]], device='cuda:0')
```

شکل ۴: زیردیتاست های تولید شده

در ادامه شبکه ای MLP با مشخصات زیر تعریف شده است:

- تعداد نرون های ورودی: ۲
- تعداد لایه‌های مخفی: ۲
- تعداد نرون‌های لایه مخفی اول: ۱۲۸
- تعداد نرون‌های لایه مخفی دوم: ۶۴
- تعداد نرون‌های خروجی: ۱
- تابع فعالساز: ReLU

۲. یک شبکه‌ی بازرخدادی مبتنی بر GRU طراحی نمایید و آن را g بنامید که وظیفه‌ی آن بهینه‌سازی وزن‌های قابل یادگیری معماری f برای هدف مورد نظر می‌باشد. با استفاده از ۵۰ زیر مجموعه‌ی ایجاد شده در قسمت قبل، شبکه‌ی g را آموزش دهید. روند پیاده‌سازی آموزش، معماری طراحی شده و سایر جزئیات مورد نظر را در گزارش درج نمایید. توجه داشته باشید به ازای هر حلقه‌ی درونی (در هر حلقه‌ی بیرونی)، یک مقداردهی کاملا جدید برای شبکه‌ی f صورت می‌گیرد. در هر زیر مجموعه نسبت آموزش به آزمون ۲:۸ است.

پاسخ

در این قسمت، بهینه‌سازی مبتنی بر شبکه بازرخدادی GRU با مشخصات زیر تعریف می‌کنیم:

- سایز ورودی: ۲
- سایز لایه مخفی: ۱۲۸
- سایز خروجی: ۱
- نرخ یادگیری: ۰/۰۰۱

سپس شبکه g را با زیرداده های بدست آورده آموزش می‌دهیم و به میانگین دقت و خطای زیر می‌رسیم:

```
Average Accuracy: 49.99
Average Error: 2.2704
```

۳. مجموعه داده‌ی دوم را نیز بارگذاری کرده و آن را نمایش دهید و تفاوت‌های آن را با مجموعه داده‌ی اول بیان کنید. حال آن را به ۳۰ زیر مجموعه همانند توضیحات قسمت ۱ تقسیم کنید. در نهایت، به ازای هر مجموعه داده، یک شبکه با معماری f در نظر گرفته و با شبکه‌ی g آن را بهینه‌سازی کنید. میانگین دقت و خطا را گزارش نمایید.

پاسخ

در این قسمت مجموعه داده دوم (`dataset_2.csv`) را می‌خوانیم و آن را نمایش می‌دهیم. این دو دیتاست در ظاهر شبیه‌به هم هستند اما ابعاد آنها با هم متفاوت است.
ابعاد دیتاست `dataset_2.csv`، $(60000, 3)$ است.

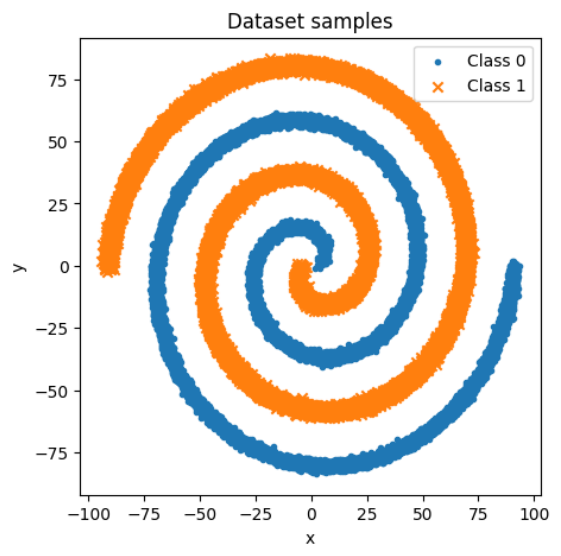
shape of data frame is: (60000, 3)

	x	y	label
32992	-6.83735	15.59275	0.0
54004	-14.62318	-29.97327	0.0
10529	7.97737	32.53706	1.0
31980	-17.03517	-77.61633	0.0
23985	48.94415	59.16045	1.0
...	...	...	...
10881	25.57032	47.04603	0.0
14668	51.60917	-37.17503	1.0
32864	-17.30422	-76.30360	0.0
58328	-84.93697	24.73012	1.0
18994	-24.61062	-7.75692	0.0

60000 rows × 3 columns

شکل ۵: خروجی دیتاست `dataset_2.csv`

دیتاست `dataset_2.csv` به‌صورت زیر رسم می‌شود:



شکل ۶: دیتاست `dataset_2.csv`

پاسخ

در ادامه مشابه با قسمت قبل، دیتاست را می‌بایست به تعدادی زیردیتاست تقسیم کنیم. در این قسمت دیتاست ورودی را به ۳۰ زیردیتاست تقسیم می‌کنیم. ابعاد هر زیردیتاست مشابه با قسمت قبل، $(2000, 3)$ می‌شود.
در ادامه مشابه با قسمت قبل، سعی می‌کنیم با استفاده از بهینه‌ساز g وزن‌های شبکه f را این‌بار با استفاده از دیتاست `dataset_2.csv` بهینه‌سازی کنیم.
در این قسمت نیز، از همان تابع g تعریف شده در قسمت قبل استفاده شده است. پس از انجام آموزش، مقدار میانگین خطا و دقت به‌صورت زیر بدست می‌آید:

```
Average Accuracy for dataset_2: 49.96
Average Error for dataset_2: 1.8790
```

۴. **(اختیاری)** با مطالعه و تحقیق روشی ارائه دهید تا بتوان عملکرد بهینه‌ساز g را بصورت کمی و کیفی ارزیابی نموده و همچنین بتوان آن را با بهینه‌ساز ADAM مقایسه نمود.

پاسخ

برای ارزیابی بهینه‌ساز g میانگین دقت و خطا یکی از معیارهای کمی مناسب برای ارزیابی است. همچنین می‌توان همانند سایر بهینه‌ساز ها معیار های کمی دیگری مانند F1-score، AUC-ROC و ... را به کد اضافه نمود و بر اساس آنها نیز عملکرد بهینه‌ساز را ارزیابی کرد.
همچنین برای مقایسه خروجی‌های بهینه‌سازی که ما نوشته ایم با بهینه‌سازی مانند Adam می‌توان شبکه را دوبار با همین دیتاست‌ها یک بار با بهینه‌ساز g و بار دیگر با بهینه‌ساز Adam آموزش داد و سپس خروجی‌های آنها که شامل میانگین دقت، خطا، امتیاز F1 و ... است مقایسه کنیم.
برای مقایسه کیفی نیز می‌توان از نمایش لصری توزیع داده‌های پیش‌بینی شده توسط هر دو بهینه‌ساز استفاده نمود.
همچنین معیاری دیگر برای برای مقایسه، بررسی پایداری بهینه‌ساز نوشته شده توسط ما و بهینه‌ساز Adam نسبت به تغییر پارامترهای ورودی شبکه و نرخ یادگیری است. مطلوب است بهینه‌ساز در برابر تغییرات پارامترهای ورودی شبکه نظیر نرخ یادگیری، تعداد Epoch و ... کمترین میزان تغییر و نوسان را داشته باشد.