تمرین سری اول – درس یادگیری ژرف

فاطمه براتی ۴۰۲۳۲۴۵۵۰۳

۱- الگوریتم Perceptron را که در کلاس دیدیم، برای طبقه‌بندی داده‌های زیر اعمال کنید (تا زمان همگرا شدن به‌روزرسانی بردار وزن را ادامه دهید).

$$(4,3,3) \in N \quad (2,-2,3) \in P \quad (1,0,-3) \in P \quad (4,2,2) \in N$$

همانطور که می‌بینید چهار داده داریم که دو داده متعلق به کلاس منفی (N) و دو داده متعلق به کلاس مثبت (P) هستند. فرض کنید بردار وزن اولیه در الگوریتم Perceptron برابر $(w_0, w_1, w_2, w_3) = (1,0,0,0)$ باشد، که w_0 بیانگر بایاس است.

الف- مراحل الگوریتم و مقدار بردار وزن را در هر مرحله از بروزرسانی الگوریتم به طور کامل بنویسید.

۱-

A: (4,3,3) ∈ N B: (2,−2,3) ∈ P
C: (4,2,2) ∈ N D: (1,0,−3) ∈ P

$(w_0, w_1, w_2, w_3) = (1, 0, 0, 0)$ ($w_0 \to b$)

الف)

Net Input:
$$y_{in} = w \cdot x = w_0 + \sum_{i=1}^{3} w_i x_i$$

Update: (α = learning rate = 1)
$$\Delta w_i = \alpha t x_i \quad i = 1,2,3$$
$$\Delta b = \alpha t$$
$$w_{i_{new}} = w_{i_{old}} + \Delta w_i$$
$$b_{new} = b_{old} + \Delta b$$

output:
$$y = f(y_{in}) = \begin{cases} 1 & \text{if } w \cdot x > 0 \\ 0 & \text{if } w \cdot x = 0 \\ -1 & \text{if } w \cdot x < 0 \end{cases}$$

از آن جا که y و t متفاوت اند، پس w ها باید update شوند.

epoch 1

Inputs x_1	x_2	x_3	Target (t)	y_{in}	y	Weight changes Δb	Δw_1	Δw_2	Δw_3	Weights b	w_1	w_2	w_3
4	3	3	−1	1	1	−1	−4	−3	−3	0	−4	−3	−3
2	−2	3	1	−11	−1	1	2	−2	3	1	−2	−5	0
4	2	2	−1	−17	−1	0	0	0	0	1	−2	−5	0
1	0	−3	1	−1	−1	1	1	0	−3	2	−1	−5	−3

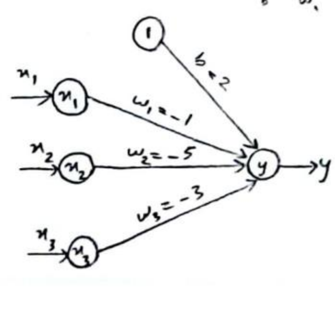
چون در این مرحله w ها update شدند، پس یک iteration دیگر نیز خواهیم داشت.

epoch 2

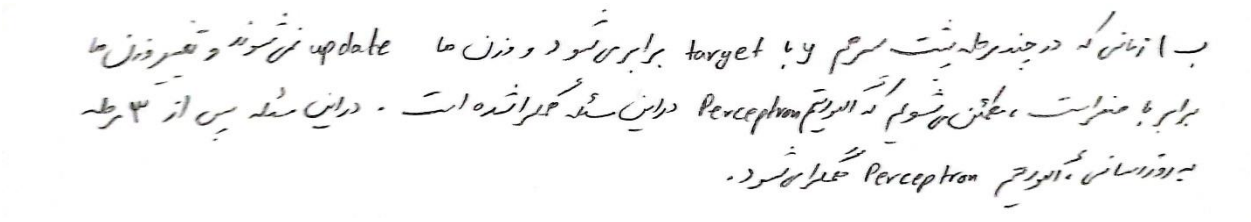
Inputs x_1	x_2	x_3	Target (t)	y_{in}	y	Weight changes Δb	Δw_1	Δw_2	Δw_3	Weights b	w_1	w_2	w_3
4	3	3	−1	−26	−1	0	0	0	0	2	−1	−5	−3
2	−2	3	1	1	1	0	0	0	0	2	−1	−5	−3
4	2	2	−1	−18	−1	0	0	0	0	2	−1	−5	−3
1	0	−3	1	10	1	0	0	0	0	2	−1	−5	−3

چون در این مرحله w ها نیازی به update نداشتند و y با t مطابقت داشت، پس نیازی به ادامه دادن iteration ها نیست.

final weights = (2, −1, −5, −3) (b, w_1, w_2, w_3)



ب- چه موقع مطمئن می‌شوید که الگوریتم Perceptron در این مسئله همگرا شده است؟ پس از چند مرحله بروزرسانی این اتفاق رخ داده است؟



ج- کد پایتون الگوریتم Perceptron بالا را در پایتون بنویسید. برای پیاده سازی این الگوریتم تنها به محاسبات ساده نیاز دارید و می‌توانید برای راحتی بیشتر از Numpy یا PyTorch نیز استفاده کنید.

کد نوشته شده به نام Perceptron Learning Algorithm.py در پوشه code موجود است.

۲- یکی از الگوریتم‌های بهینه‌سازی مورد بررسی در کلاس الگوریتم بهینه‌سازی Momentum بود.

الف- با توجه به مباحث کلاس و نیز با جستجو در اینترنت دلیل برتری این روش نسبت به الگوریتم پایه Gradient descent را توضیح دهید.

Momentum اغلب بهتر از Gradient Descent در نظر گرفته می شود زیرا می‌تواند برخی از محدودیت های GD را رفع کند.

علل برتری Momentum:

۱. همگرایی سریع تر در موارد خاص:

- Momentum به تسریع همگرایی از طریق مناطق flat (مسطح) یا با انحنای کم کمک می‌کند. هنگامی که گرادیان نویز دارد یا تابع هدف دارای نواحی flat است، Momentum به فرآیند بهینه‌سازی اجازه می‌دهد تا به طور موثرتری از این مناطق عبور کند.

- در مقابل، GD می‌تواند در مناطق flat گیر کند یا به دلیل شیب های نویز نوسان کند.

۲. غلبه بر مینیمم های محلی:

- Momentum الگوریتم بهینه‌سازی را قادر می‌سازد تا به طور مؤثرتری از مینیمم‌های محلی فرار کند. با جمع آوری گرادیان های گذشته، اینرسی ایجاد می‌کند و می‌تواند از مینیمم های محلی کم عمق فرار کند.

- GD، بدون Momentum، ممکن است به مینیمم محلی کمتر از حد مطلوب همگرا شود.

۳. هموارسازی مسیر بهینه سازی:

- Momentum با در نظر گرفتن گرادیان های تاریخی مسیر بهینه سازی را هموار می‌کند. نوسانات را کم می‌کند و مسیری پایدارتر به سمت مینیمم فراهم می‌کند.

- GD می‌تواند رفتار نامنظم از خود نشان دهد، به خصوص زمانی که گرادیان ها دارای نویز هستند.

۴. تجسم:

- تصور کنید توپی را از یک تپه می‌غلتانید: Momentum به توپ اجازه می‌دهد تا هنگام غلتیدن در سراشیبی سرعت بگیرد و احتمال گیر کردن آن در یک دره کم عمق را کاهش دهد.

- GD، بدون Momentum، مانند یک توپ که مستقیماً بر اساس گرادیان می‌غلتد رفتار می‌کند، که می‌تواند به مسیرهای غیربهینه منجر شود.

۵. اجرای عملی:

- در عمل، بسیاری از الگوریتم‌های بهینه‌سازی مانند Adam و RMSProp را برای بهبود همگرایی ترکیب می‌کنند.

- محققان Momentum را در انجام وظایف مختلف یادگیری ماشین موثر می‌دانند.

ب- تفاوت روش Nesterov Momentum با روش Momentum چیست و کدامیک نسبت به دیگری برتری دارد؟ در این قسمت نیز با توجه به مباحث کلاس و جستجو در اینترنت پاسخ دهید.

۱. Momentum:

- الگوریتم:

- در GD مبتنی بر Momentum، از به‌روزرسانی وزن قبلی m و گرادیان فعلی g برای به‌روزرسانی پارامترها p استفاده می‌کنیم.

- قانون به روز رسانی این است: $v = \beta m - \eta g$ ، که در آن β یک ثابت است، η نرخ یادگیری است، و v نشان دهنده سرعت است.

- مقدار پارامتر جدید به صورت زیر بدست می آید: $pnew = p + v$.

- رفتار:

- Momentum شیب های گذشته را جمع می‌کند و به مسیرهای بهینه سازی هموارتر منجر می‌شود.

- می‌تواند به طور موثرتری از مینیمم های محلی فرار کند.

- به دلیل Momentum انباشته شده سریعتر حرکت می‌کند.

- تجسم:

- تصور کنید توپی را از یک تپه می‌غلتانید: Momentum به توپ اجازه می‌دهد تا هنگام غلتیدن در سراشیبی سرعت بگیرد و احتمال گیر کردن آن در یک دره کم عمق را کاهش دهد.

۲. Nesterov Momentum نستروف (NAG):

- الگوریتم:

- به جای محاسبه گرادیان برای پارامترهای فعلی W ، Momentum نستروف گرادیان ها را برای (W - β * Vt-۱) محاسبه می‌کند، که در آن Vt-۱ سرعت قبلی است.

- قانون به روز رسانی این است: pnew = p + βv - ηg.

- رفتار:

- Nesterov Momentum رویکردی «gamble-correct» دارد:

- ابتدا در جهت سرعت (مثل قمار) قدم می گذارد.

- سپس بردار سرعت را بر اساس مکان جدید تصحیح می‌کند.

- هدف آن این است که به جای دنبال کردن سرعت به مقدار واقعی برآورد شده نزدیکتر شود.

- به بهبود سرعت همگرایی SGD کمک می‌کند.

به طور خلاصه، Nesterov Momentum جهت به روز رسانی را با در نظر گرفتن گرادیان در موقعیت "آینده" تنظیم می‌کند که منجر به رفتار همگرایی بهتر می شود. در حالی که هر دو روش گرادیان نزولی را بهبود می‌دهند، Nesterov Momentum مزایای بیشتری از نظر سرعت همگرایی و فرار از مینیمم‌های محلی ارائه می‌کند.

۳- یک شبکه CNN دارای سه لایه کانولوشنی، سه لایه fully connected، و یک لایه max pooling است. همچنین یک لایه فعالسازی بعد از هر لایه کانولوشنی و هر لایه fully connected قرار گرفته است. ورودی شبکه تصاویر **رنگی** با ابعاد ۲۰۰*۲۰۰ است. در مورد این شبکه داریم:

ترتیب لایه ها به این صورت است:

Conv1 >> Activation1 >> Conv2 >> Activation2 >> Max-pooling >> Conv3 >> Activation3 >> FC1 >> Activation4 >> FC2 >> Activation5>> FC3>>Activation6

که Conv بیانگر لایه کانولوشنی و FC بیانگر لایه Fully connected است.

لایه های کانولوشنی اول تا سوم به ترتیب دارای 10، 4، و 20 فیلتر هستند. تمام فیلترها را با ابعاد ۳*۳ فرض کنید.

بدیهی است عدد ۳ بیانگر عرض و ارتفاع فیلتر است و عمق فیلتر به عمق داده ورودی به لایه مرتبط است.

padding برای لایه های کانولوشنی اول و دوم برابر ۱ و برای لایه کانولوشنی سوم برابر صفر است.

Stride برای لایه کانولوشنی اول برابر ۲ و برای لایه های کانولوشنی دوم و سوم برابر ۱ است.

شبکه فقط دارای یک لایه max pooling است و فیلتر های این لایه ۲*۲ هستند و stride آن نیز برابر ۲ است.

تابع فعالسازی همه لایه ها sigmoid است.

تعداد نورون های لایه های FC1، FC2، و FC3 را به ترتیب برابر ۲۰۰، ۱۰۰، و ۱ فرض کنید.

به موارد زیر پاسخ دهید:

ابعاد داده سه بعدی در خروجی تمام لایه‌ها را بدست آورید. یعنی بیان کنید در خروجی هر لایه، داده با چه ابعادی داریم؟ (به ازای یک تصویر رنگی ۲۰۰*۲۰۰ به عنوان ورودی شبکه)

اگر فرض کنیم هر فیلتر و نیز هر نورون علاوه بر وزن‌هایش یک مقدار بایاس نیز دارد، تعداد کل پارامترهای این شبکه (تعداد کل وزن ها و بایاس ها) را بدست آورید.

# of output pixels = $\frac{n+2p-f}{s}+1$

Input

200x200x3

Conv Layer 1
10 filters
filter size = 3
stride = 2
padding = 1

100x100x10

parameters = 10 x (3x3x3+1) = 280

Conv Layer 2
4 filters
filter size = 3
stride = 1
padding = 1

100x100x4

parameters = 4 x (3x3x10+1) = 364

Conv Layer 3
20 filters
filter size = 3
stride = 1
padding = 0

98x98x20

parameters = 20 x (3x3x4+1) = 1820

Max Pooling Layer
1 filter
filter size = 2
stride = 2
No additional parameters
(only downsampling)

49x49x20

FC Layer 1
Neurons = 200

200

parameters = 49x49x20x200 + 200 = 968200

FC Layer 2
Neurons = 100

100

parameters = 200x100 + 100 = 20100

FC Layer 3
Neurons = 1

1

parameters = 100x1 + 1 = 101

Total parameters in the entire network:

280 + 364 + 1820 + 968200 + 20100 + 101 = 990865