Subject: 8101983358 امیرمهدی افشاری پور

Year. Month. Date. ( )

1) الف: توابع هزینه: توابعی هستند که بایی Jacobian شان در شبکه عصبی نسبت به وزنهای شبکه minimize شود. بعبارت دیگر، اگر تابع هزینه را با $f(\vec{w}, \vec{x})$ نشان دهیم:

$$\vec{w}^* = \arg\min_{\vec{w}} f(\vec{w}, \vec{x})$$

($\vec{w}^*$ همان بردار وزن بهینه برای شبکه عصبی است.)

همچنین تابع هزینه نقش مهمی در update کردن وزنها دارد:

$$w_{ij}{}_{new} = w_{ij}{}_{old} \pm \eta \frac{\partial f(\vec{w}, \vec{x})}{\partial w_{ij}} \quad \text{(learning weights)}$$

بنابراین تعریف درست تابع هزینه تأثیر مهمی در training شبکه عصبی دارد. بعنوان مثال، تابع هزینه مورد استفاده در الگوریتم شبکه های عصبی بصورت زیر است:

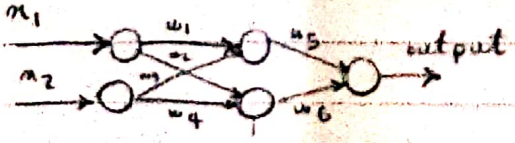
$$f(\vec{w}, \vec{x}) = \sum_{i \in train} (\text{sigmoid}(z_i) - \text{actual}_i)^2$$

(z_i خروجی آخرین لایه شبکه عصبی برای sample x_i است.)

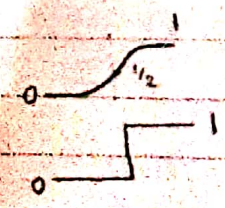
- توابع فعالساز (activator): نمایش متار $w^T x$ برای لایه بعد نام دارد. این متار به یک تابع فعالساز میرود که دامنه‌ای $[0, 1]$ است، سپس وارد لایه $i+1$ میشود.

توابع فعالساز به ما این قابلیت را میدهند که لایه‌های غیرخطی بین y و x را تضمین دهیم.

اگر این توابع نباشند، شبکه عصبی ما رگرسیون خطی خواهد بود:



$$\begin{aligned} \text{output} &= w_5(w_2 x_2 + w_1 x_1) + \\ &\quad w_6(w_3 x_1 + w_4 x_2) \\ &= \alpha x_1 + \beta x_2 \end{aligned}$$

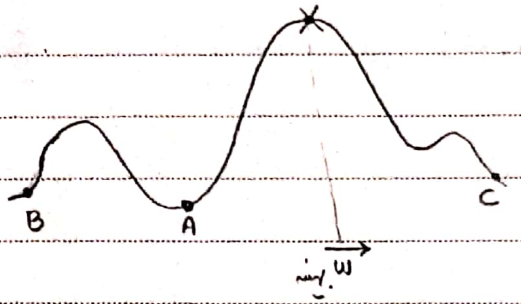


اگر توابع فعالساز پیوسته باشند، احتمال تعلق به کلاس را مشخص می‌کند.

" " ناپیوسته " " خود کلاس را مشخص می‌کند.

PAPCO

۱) ب: فرض کنید توزیع $\vec{w}$ بصورت روبرو باشد:



هدف دست یافتن به global optima از طریق gradient descent است. یعنی:

$$\vec{w}_{new} = \vec{w}_{old} - \eta \frac{\partial err}{\partial \vec{w}}$$

حال اینکه ما $\vec{w}_0$ را کجا انتخاب کنیم (مثلا میان A, B, C) تاثیرهای زیر را دارد:

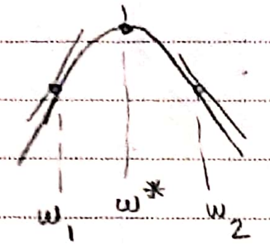
۱) اگر $\vec{w}_0$ نزدیک اکسترمم مطلق باشد، بسرعت به w بهینه می‌رسیم و در زمان training کمی انجام میگردد.

اما در حالت کلی چنانچه η مقدار مناسبی داشته باشد بالاخره به $\vec{w}$ بهینه می‌رسیم و تفاوتی در نتیجه نهایی در آنها وجود نخواهد داشت.

۱) ج: forward propagation عبارت است از ترکیب خطی بردار $\vec{w}$ و ضرب نمودن ها در هر لایه و عبور دادن از تابع activator. پارامترها update نمی‌شود و از لایه input به output می‌رسیم.

back propagation عبارت است از update کردن وزن‌ها تنها با استفاده از gradient descent و از لایه آخر به اول باز می‌گردیم.

۱) د: چنانچه η مقدار بسیار کمی داشته باشد، رسیدن به global optima (یا همان train) بسیار زمانبر خواهد بود، ممکن است در local optima گیر بیفتیم و به $\vec{w}$ بهینه نرسیم. اگر مقدار η زیادی هم داشته باشد ممکن است حول global optima نوسان کنیم.



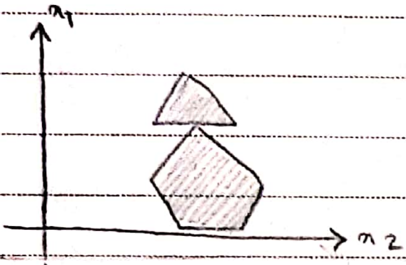
همانطور که دیده میشود، طبق فرمول $w_{new} = w_{old} - \eta \frac{\partial err}{w_{old}}$

بصورت مداوم حول w^* نوسان می‌کنیم و علامت گرادیان

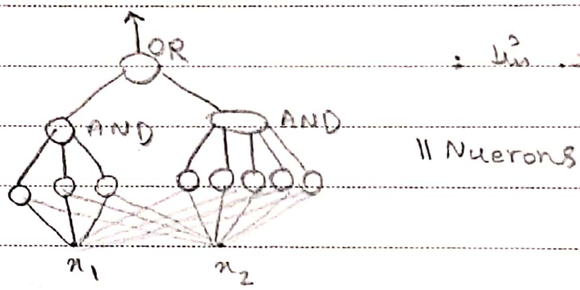
بین + و − مدام عوض میشود.

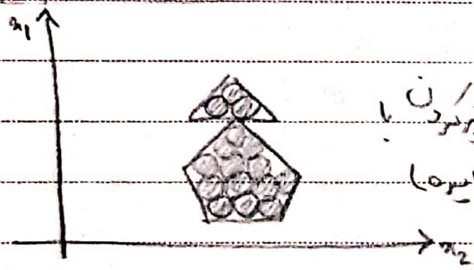
۵) با داشتن لایه‌های بیشتر، امکان ساختن مرزهای تقسیم پیچیده با تعداد نورون کم فراهم میشود.

در حالیکه با یک لایه سازی همان مرز تقسیم با تعداد لایه کم مرتب میشود width (عرض) شبکه بسیار زیاد میشود. مثلا:

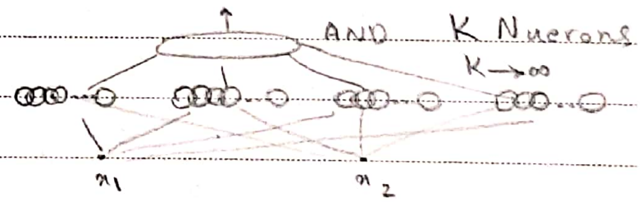


(راه ۱):





(راه ۲): پُر کردن با دایره

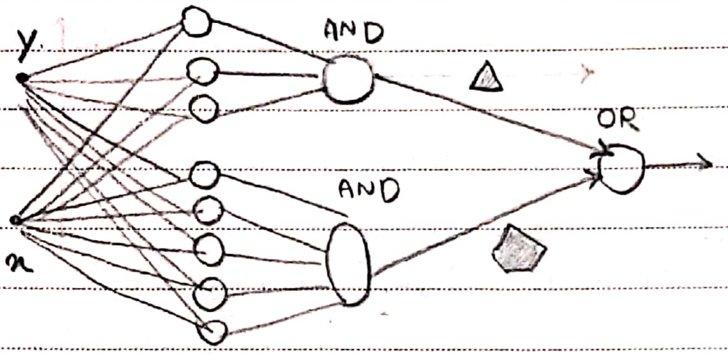


دقت شود هر دایره با یک لایه هم قرار داد، تعداد بسیار زیاد نورون بدست می‌آید.

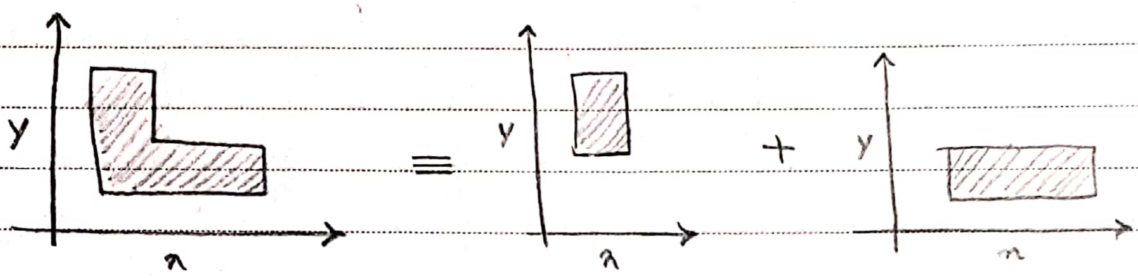
۶) overfitting بزرگترین مشکل train کردن شبکه‌های عمیق است. از آنجا که ما سعی می‌کنیم

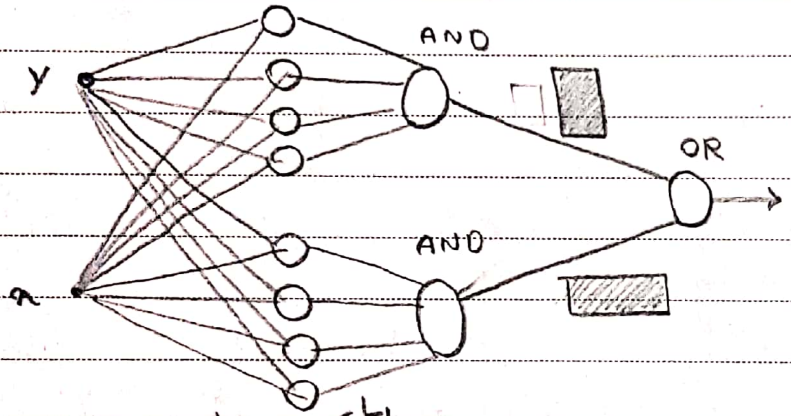
تابع هزینه $\frac{1}{2}\sum_i [sigmoid(z_{out}) - actual_i]^2$ را minimize کنیم، یا بعبارت دیگر به global optima برسیم، شبکه روی داده‌های train به دقت بسیار بالایی می‌رسد که این از علائم overfitting است. از جمله راه‌حل‌هایی که وجود دارد اینست که اجازه دهیم در مرحله train شبکه overfit کند و وزنها بهینه شوند، اما در مرحله test تعدادی از وزنها را صفر کنیم، یا تعدادی از نورون‌ها را حذف کنیم.

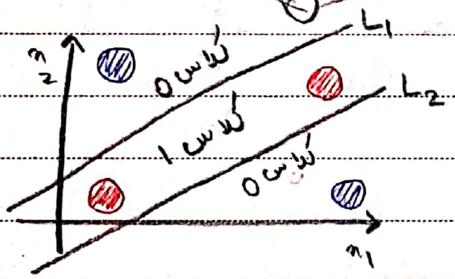
۲) الف: شبکه از ۲ لایه hidden تشکیل شده که به ترتیب ۸ و ۲ نورون دارند



۲) ب:

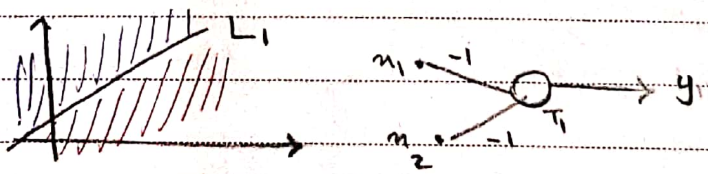






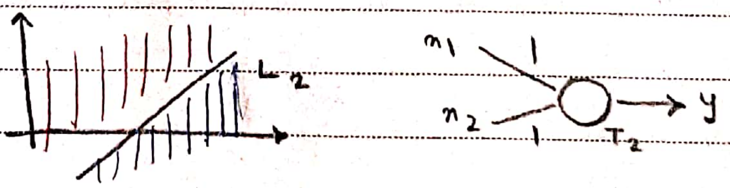
۲) ج: هر کدام از خطوط L_1 و L_2 را میتوان با یک perceptron نشان داد؛ به عبارت دیگر:

L_1:



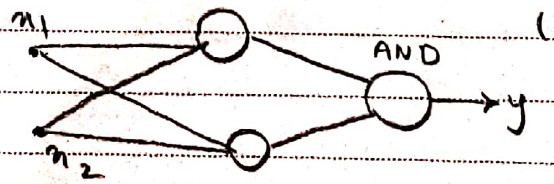
$$y = \begin{cases} 1 & -x_1 - x_2 > T_1 \\ 0 & ow \end{cases}$$

L_2:



$$y = \begin{cases} 1 & x_1 + x_2 > T_2 \\ 0 & ow \end{cases}$$

⟹



حاصل AND y_1 و y_2 میشود ناحیه کلاس قرمز (۱)

و باقی منطقه ناحیه کلاس آبی (۰)

Subject,
Year. Month. Date. ( )

4) الف: طبق تعریف تابع relu :

$$relu(z) = \begin{cases} z & \text{if } z>0 \\ 0 & \text{o.w.} \end{cases}$$

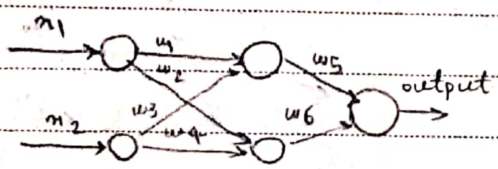
این تابع حاصل $\sum x_i w_i$ در هر لایه را بعنوان ورودی گرفته و تنها یک شرط روی آن می گذارد و هیچ عملیات دیگری انجام نمی دهد. از طرفی مشتق این تابع نیز بسیار آسان است و هنگام update کردن وزنها با محاسبات کمتری داریم :

$$\frac{d}{dz} relu(z) = \begin{cases} 1 & \text{if } z>0 \\ 0 & \text{if } z<0 \end{cases}$$

$$\frac{d}{dz} tanh(z) = 1 - (tanh(z))^2$$

$$\frac{d}{dz} sigmoid(z) = sigmoid(z)\,(1 - sigmoid(z))$$

برای مثال داریم :



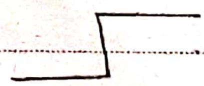
$$\frac{\partial error}{\partial w_5} = \frac{\partial error}{\partial relu(w_5 z_1 + w_6 z_2)} \times \frac{\partial relu(w_5 z_1 + w_6 z_2)}{\partial (w_5 z_1 + w_6 z_2)} \times \frac{\partial (w_5 z_1 + w_6 z_2)}{\partial w_5}$$

z_1

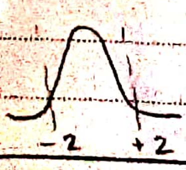
← 1 یا 0 (ساده تر از مشتق های sigmoid و tanh)

4) ب : همان شبکه عصبی بخش الف را در نظر بگیرید. چنانچه حاصل $|w_5 z_1 + w_6 z_2| > 2$ ، آنگاه $\frac{\partial tanh(w_5 z_1 + w_6 z_2)}{\partial (w_5 z_1 + w_6 z_2)}$ صفر میشود (برای sigmoid هم همینطور است). در نتیجه وزن ها w_5 ، w_6 update نمی شوند (همان مقدار قبلی را میگیرند). به این مشکل vanishing gradient گویند. دلیل این مشکل شکل $\frac{d}{dz} tanh(z)$ و $\frac{d}{dz} sigmoid(z)$ است، اما تابع relu [illegible] $\sum w_i x_i$ مقادیر بسیار بالایی هم بگیرند در آن نقاط هم مشتق مقدار دارد و فرآیند update وزن ها انجام می شود.

$relu'(z)$:



$sigmoid'(z)$ & $tanh'(z)$:



PAPCO

Subject:
Year. Month. Date. ( )

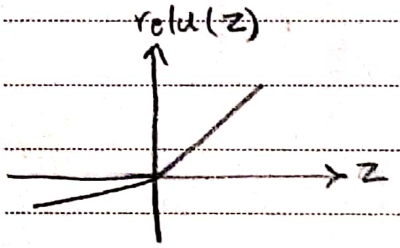
۴) ج: برای classification از sigmoid استفاده می‌کنیم زیرا bounded است

(بین [0,1]) . در cost function داریم: $Error(z) = \frac{1}{2} \sum_i [f(z_i) - output_i]^2$

در مسئله کلاس بندی چون output یا 0 است یا 1 ، منطقی است که از relu که

مقادیر بسیار بیشتری از 1 هم می‌تواند بگیرد استفاده نکنیم. (relu برای لایه آخر regression

مناسب است.)



۴) د: تابع

$$leaky\_relu(z) = \begin{cases} z & \text{if } z > 0 \\ az & \text{o.w.} \end{cases}$$

استفاده از leaky_relu مزایای زیر را دارد:

۱) از آنجایی که خروجی $relu(\sum_i w_i x_i)$ زمانیکه $\sum_i w_i x_i < 0$ ، صفر است، برخی نورون‌ها

فعال نمی‌شوند و در نتیجه برخی وزنها تا آخر صفر می‌مانند، update نمی‌شوند. در نتیجه بهتر است

از leaky_relu استفاده کنیم که اطلاعات بیشتری در مورد sampleهای با مقادیر منفی داشته باشیم

۲) در regression که خروجی مطلوب نمونه ام ($output_i$) می‌تواند منفی هم باشد استفاده از

leaky relu بهتر است.

۵) الف: برای متغیر استخدام داریم:

$$4+,\ 6- \Rightarrow H(\text{استخدام}) = -\frac{4}{10}\log(\frac{4}{10}) - \frac{6}{10}\log(\frac{6}{10}) = 0.292$$

ب) برای متغیر تحصیلات تکمیلی داریم:

کارشناسی: 3+ 1- : $H(\text{کارشناسی}) = -\frac{3}{4}\log(\frac{3}{4}) - \frac{1}{4}\log(\frac{1}{4}) = +0.244$

کارشناسی ارشد: 1+ 3- : $H(\text{ارشد}) = -\frac{1}{4}\log(\frac{1}{4}) - \frac{3}{4}\log(\frac{3}{4}) = +0.244$

دکتری: 2- : $H(\text{دکتری}) = -0\log(0) - 1\log(1) = 0$

$\Rightarrow H(\text{تحصیلات تکمیلی}) = [0.244 \times 4 + 0.244 \times 4 + 0]/10 = 0.195$

$\Rightarrow \text{Info\_Gain}(\text{تحصیلات تکمیلی}) = 0.292 - 0.195 = 0.097$

متغیر سابقه کاری:

سال1: 3- 1+ : $H(\text{سال}1) = -\frac{3}{4}\log(\frac{3}{4}) - \frac{1}{4}\log(\frac{1}{4}) = 0.244$

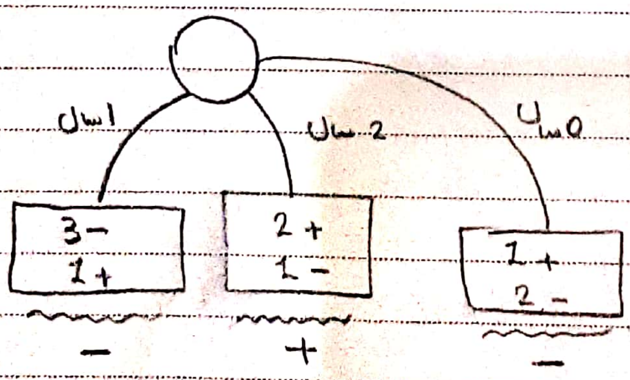
سال2: 2+ 1- : $H(\text{سال}2) = -\frac{2}{3}\log(\frac{2}{3}) - \frac{1}{3}\log(\frac{1}{3}) = 0.276$

سال0: 1+ 2- : $H(\text{سال}0) = -\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3}) = 0.276$

$\Rightarrow H(\text{سابقه کاری}) = [4 \times 0.244 + 3 \times 0.276 + 3 \times 0.276]/10 = 0.2632$

$\Rightarrow \text{Info\_Gain}(\text{سابقه کاری}) = 0.292 - 0.263 = 0.029$

ج):



act / pred	-	+
-	5	2
+	1	2

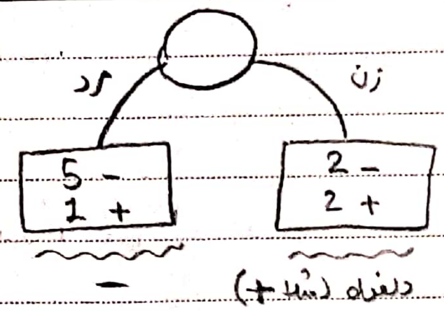
$\text{accuracy} = \frac{3+2+2}{10} = 0.7$

$\text{precision} = \frac{2}{3}$

$\text{recall} = \frac{2}{4}$

$\text{specificity} = \frac{5}{5+1} = \frac{5}{6}$

د):



pred \ act	-	+
-	5	1
+	2	2

$accuracy = \frac{7}{10}$ $recall = \frac{2}{3}$

$precision = \frac{2}{4}$ $specificity = \frac{5}{7}$

3) الف: تمام تصاویر کلاس اول یک حفره را در بر می‌گیرند. به عبارت دیگر تعدادی پیکسل مشکی تعدادی پیکسل سفید را محاصره می‌کنند.

3) ب: تصویر اول از کلاس اول (چپ بالا):

0	0	1	0
0	1	1	1
0	1	0	1
0	1	1	1

×

1	1	1
1	0	1
1	1	1

=

0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	1
0	0	0	1	0	1
0	1	1	1	1	1

$\sum w_i a_i + bias$ →

1	3
2	6

2 + 6
maxpooling → 6

تصویر دوم از کلاس اول (راست بالا):

1	1	1	0
1	0	1	0
1	0	1	0
1	1	1	0

×

1	1	1
1	0	1
1	1	1

=

1	1	1	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	0	0
1	1	1	1	1	0

$\sum w_i x_i + bias$ ⟹

5	4
5	4

max pooling ⟹ 5

تصویر اول از کلاس دوم (پایین چپ):

0	0	0	0
0	1	0	1
1	1	1	0
0	1	0	0

×

1	1	1
1	0	1
1	1	1

=

0	0	0	0	0	0
0	0	0	1	0	1
1	1	1	1	1	0
0	1	0	1	0	1
1	0	1	1	0	0
0	1	0	1	0	0

$\sum w_i x_i + bias$ ⟹

1	2
2	2

max pooling ⟹ 2

تصویر دوم از کلاس دوم (پایین راست):

0	1	0	0
0	1	0	0
0	1	1	0
0	1	0	0

×

1	1	1
1	0	1
1	1	1

=

0	1	0	1	0	0
0	0	0	1	0	0
0	1	1	1	0	0
0	1	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	0

$\sum w_i x_i + bias$ ⟹

1	1
1	1

max pooling ⟹ 1

ج) فیلتری که دنبالش بودیم در هر تصویر اعمال شد. یک حفره سینی بود که توسط تعداد پیکسل قرمز محاصره شده بود (feature کلاس 1). حال با convolve و max pooling کردن شباهت هر عکس با فیلتر (اینکه آیا بخشی از عکس شبیه این فیلتر هست؟) ارزیابی میشود. همانطور که دیده میشود، این شباهت برای عکسهای کلاس اول زیاد است، بنابراین با قرار دادن یک treshold میتوان عکسهای کلاس اول و دوم را از هم جدا کرد.