
بسم الله الرحمن الرحيم

فهرست مطالب

1. مزایا و انتظارات از شبکه عصبی
2. مسائل مناسب برای یادگیری شبکه های عصبی
3. Perceptron
4. یادگیری یک پرسپترون
5. توانائی پرسپترون
6. آموزش پرسپترون
7. قانون دلتا Delta Rule
8. مشکلات روش Gradient descent

مزایا و انتظارات از شبکه عصبی

۱) قابلیت یادگیری

– شبکه عصبی سیستم کاملا پیچیده و غیرخطی

- قابلیت یادگیری یعنی توانایی تنظیم پارامترهای شبکه (وزنهای سیناپتیکی) در مسیر زمان که محیط شبکه تغییر می کند و شبکه شرایط جدید را تجربه می کند، با این هدف که شبکه بتواند با آموزش مختصر برای شرایط جدید نیز کارآمد باشد.

۲) پراکندگی اطلاعات

– پردازش اطلاعات به صورت متن

– هر نرون متاثر از فعالیت سایر نرونها (هر وزن مربوط به همه ورودیها)

– اگر بخشی از نرونهای شبکه حذف شوند یا عملکرد غلط داشته باشند, باز هم احتمال رسیدن به پاسخ صحیح وجود دارد.

مزایا و انتظارات از شبکه عصبی

۳) قابلیت تعمیم

- آموزش شبکه با مثالهای اولیه
- ارائه خروجی مناسب در مقابل ورودیهای آموزش داده نشده
- شبکه تابع را یاد می گیرد، الگوریتم را می آموزد و یا رابطه تحلیلی مناسبی برای نقاطی در فضا به دست می آورد.

۴) پردازش موازی

- پاسخ همزمان سلولهای قرار گرفته در یک تراز به ورودیهای آن تراز
- افزایش سرعت پردازش
- توزیع وظیفه پردازش بین پردازنده های کوچکتر مستقل

۵) مقاوم بودن

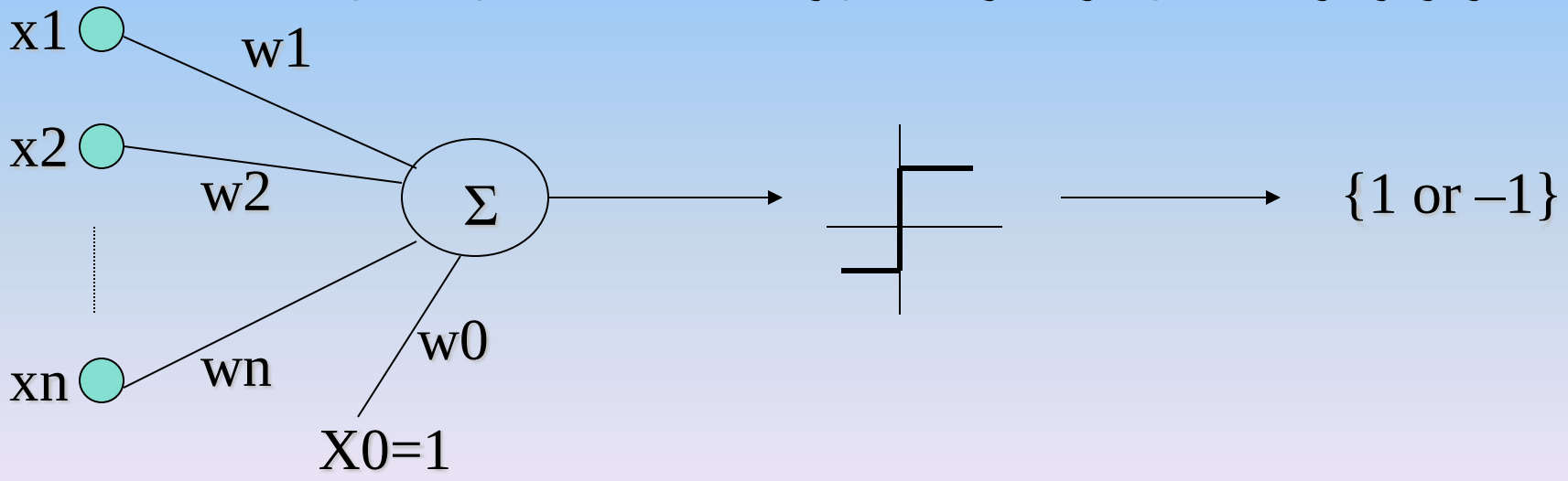
- رفتار مستقل هر سلول
- تصحیح خطاهای محلی در یک روند همکاری
- افزایش قابلیت مقاوم بودن (تحمل پذیری خطا) در سیستم

مسائل مناسب برای یادگیری شبکه های عصبی

- خطا در داده های آموزشی وجود داشته باشد. مثل مسائلی که داده های آموزشی دارای نویز حاصل از دادهای سنسورها نظیر دوربین و میکروفن ها هستند.
- مواردی که نمونه ها توسط مقادیر زیادی زوج ویژگی-مقدار نشان داده شده باشند. نظیر داده های حاصل از یک دوربین ویدئوئی.
- تابع هدف دارای مقادیر پیوسته باشد.
- زمان کافی برای یادگیری وجود داشته باشد. این روش در مقایسه با روشهای دیگر نظیر درخت تصمیم نیاز به زمان بیشتری برای یادگیری دارد.

Perceptron

- نوعی از شبکه عصبی برمبنای یک واحد محاسباتی به نام پرسپترون ساخته میشود. یک پرسپترون برداری از ورودیهای با مقادیر حقیقی را گرفته و یک ترکیب خطی از این ورودیها را محاسبه میکند. اگر حاصل از یک مقدار آستانه بیشتر بود خروجی پرسپترون بر اساس تابع فعالیت زیربرابر با 1 و در غیر اینصورت معادل 1- خواهد بود.

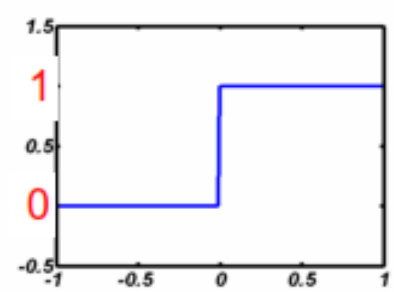


برخی توابع تحریک مرسوم نرون مصنوعی

تابع غیر خطی

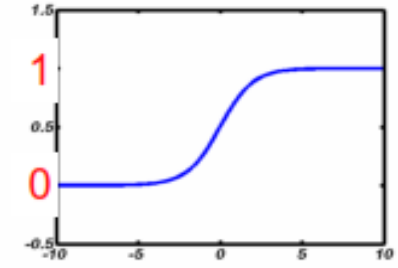
- سخت (پله‌ای)

$$y = \begin{cases} 1 & Y \geq 0 \\ 0 & Y < 0 \end{cases}$$



- نرم تک قطبی (سیگموئید)

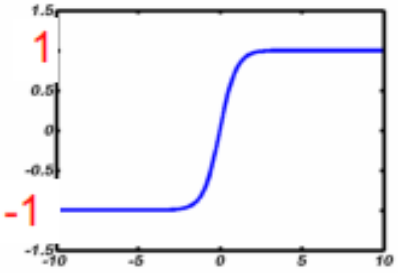
$$f(Y) = \frac{1}{1+e^{-Y}}$$



- نرم دوقطبی

$$f(Y) = \frac{1-e^{-Y}}{1+e^{-Y}} = \tanh(\frac{Y}{2})$$

$$\tanh(Y) = \frac{e^{Y}-e^{-Y}}{e^{Y}+e^{-Y}} = \frac{1-e^{-2Y}}{1+e^{-2Y}}$$



یادگیری یک پرسپترون

- خروجی پرسپترون توسط رابطه زیر مشخص میشود:

$$O(x_1, x_2, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

- که برای سادگی آن را می توان بصورت زیر نشان داد:

$$O(\mathbf{X}) = \text{sgn}(\mathbf{WX}) \text{ where}$$

$$\text{Sgn}(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

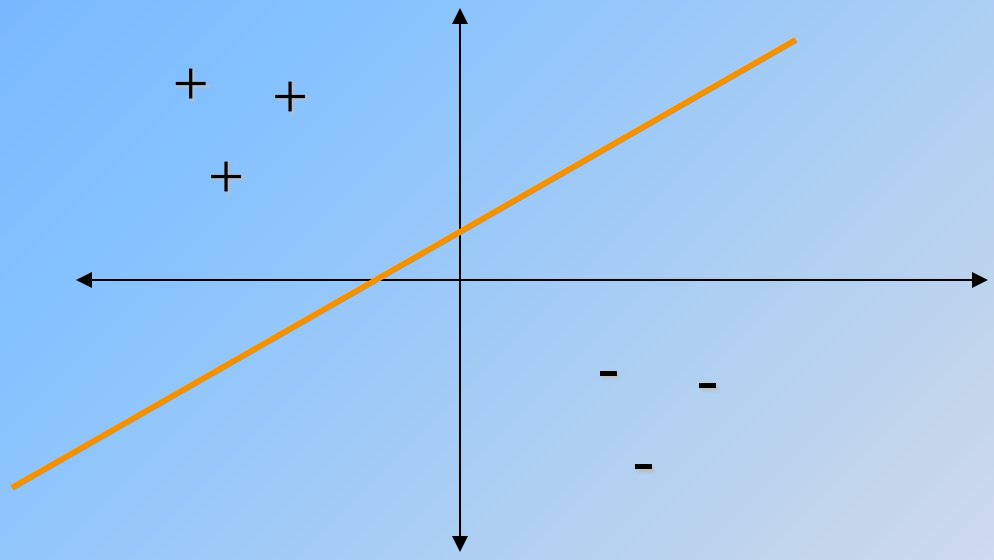
یادگیری پرسپترون عبارت است از:

پیدا کردن مقادیردرستی برای W

بنابراین منظور از یادگیری پرسپترون عبارت است ازمجموعه تمام مقادیر حقیقی ممکن برای بردارهای وزن.

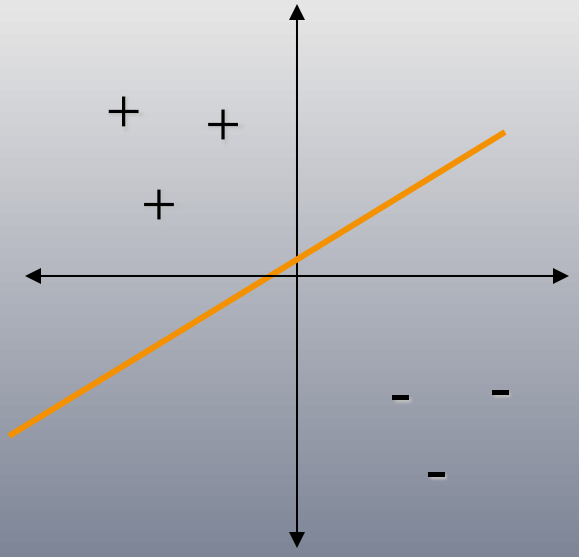
توانائی پرسپترون

- پریسپترون را میتوان بصورت یک سطح تصمیم hyperplane در فضای n بعدی نمونه ها در نظر گرفت. پرسپترون برای نمونه های یک طرف صفحه مقدار 1 و برای مقادیر طرف دیگر مقدار 1- بوجود میاورد.

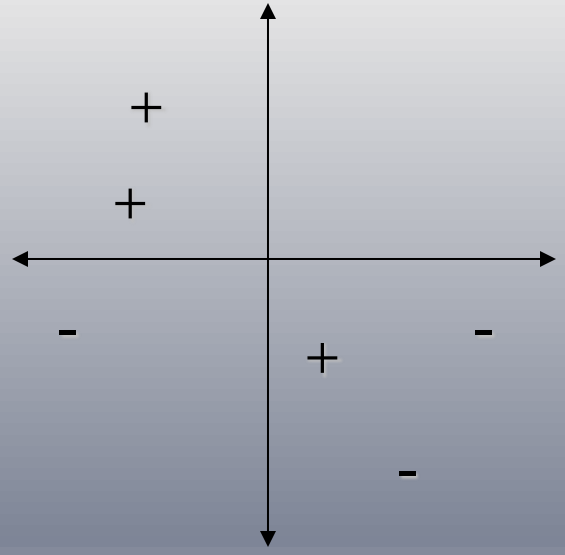


توابعی که پرسپترون قادر به یادگیری آنها می باشد

- یک پرسپترون فقط قادر است مثالهائی را یاد بگیرد که بصورت خطی جداپذیر باشند. اینگونه مثالها مواردی هستند که به طور کامل توسط یک hyperplane قابل جدا سازی میباشند.



Linearly separable

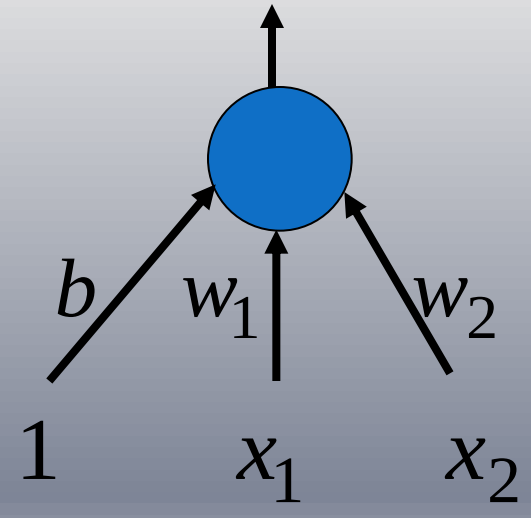


Non-linearly separable

افزودن بایاس

$$\hat{y} = b + \sum_i x_i w_i$$

- افزودن بایاس موجب میشود تا استفاده از شبکه پرسپترون با سهولت بیشتری انجام شود.
- برای اینکه برای یادگیری بایاس نیازی به استفاده از قانون دیگری نداشته باشیم بایاس را به صورت یک ورودی با مقدار ثابت 1 در نظر گرفته و وزن W_0 را به آن اختصاص میدهیم.



$$\hat{y} = w_0 + \sum_{i=1} x_i w_i$$

آموزش پرسپترون

- چگونه وزنهای یک پرسپترون آموزش داده شوند به نحوی که پرسپترون برای مثالهای آموزشی مقادیر صحیح را ایجاد نماید؟
- دو راه مختلف:
  - **قانون پرسپترون**
  - **قانون دلتا**

آموزش پرسپترون

الگوریتم یادگیری پرسپترون

1. مقادیری تصادفی به وزنها نسبت میدهیم.
2. پرسپترون را به تک تک مثالهای آموزشی اعمال می کنیم. اگر مثال غلط ارزیابی شود مقادیر وزنهای پرسپترون را تصحیح می کنیم.
3. آیا تمامی مثالهای آموزشی درست ارزیابی میشوند:
   - بله ← پایان الگوریتم
   - خیر ←به مرحله 2 برمیگردیم

قانون پرسپترون

- برای یک مثال آموزشی $X = (x_1, x_2, ..., x_n)$ در هر مرحله وزنها بر اساس قانون پرسپترون به صورت زیر تغییر میکند:

$$w_i = w_i + \Delta w_i$$

که در آن

$$\Delta w_i = \eta (t - o) x_i$$

t: target output

o: output generated by the perceptron

η: constant called the learning rate (e.g., 0.1)

اثبات شده است که برای یک مجموعه مثال جداپذیرخطی این روش همگرا شده و پرسپترون قادر به جدا سازی صحیح مثالها خواهد شد.

قانون دلتا Delta Rule

این الگوریتم که بعضاً میانگین مربع خطا نیز نامیده می شود وقتی استفاده می‌گردد که سیگنال خطا مینیمم شود که به واقع اختلاف بین خروجی واقعی شبکه و خروجی مورد انتظار (مطلوب) مینیمم باشد. در این روش سیگنال خطا جهت اصلاح وزن ها و بایاس‌های نرون ها به عقب منتشر می‌شود. الگوریتم پس انتشار خطا معمول ترین روش پیاده سازی از قانون یادگیری دلتا است که دست کم در ۷۵٪ از کاربردهای شبکه های عصبی مصنوعی استفاده می‌گردد.

قانون دلتا Delta Rule

- وقتی که مثالها به صورت خطی جداپذیر نباشند قانون پرسپترون همگرا نخواهد شد. برای غلبه بر این مشکل از قانون دلتا استفاده میشود .
- ایده اصلی این قانون استفاده از Gradient descent برای جستجو در فضای فرضیه وزنهای ممکن می باشد. این قانون پایه روش Back propagation است که برای آموزش شبکه با چندین نرون به هم متصل به کار میرود.
- همچنین این روش پایه ای برای انواع الگوریتمهای یادگیری است که باید فضای فرضیه ای شامل فرضیه های مختلف پیوسته را جستجو کنند.

قانون دلتا Delta Rule

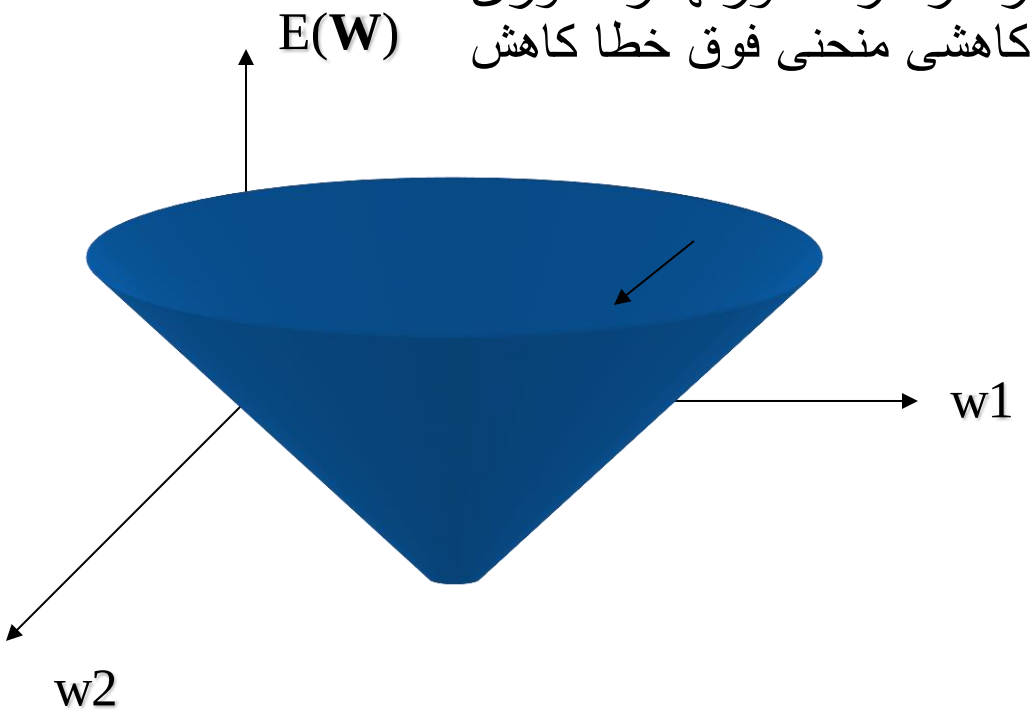
- برای درک بهتر این روش آن را به یک پرسپترون فاقد حد آستانه اعمال می کنیم.
- خطای آموزش

$$E = \frac{1}{2} \sum_i (t_i - o_i)^2$$

- که این مجموع برای تمام مثالهای آموزشی انجام می شود.

الگوریتم Gradient descent

- با توجه به نحوه تعریف E سطح خطا بصورت یک سهمی خواهد بود. ما بدنبال وزنهائی هستیم که حداقل خطا را داشته باشند . الگوریتم Gradient descent در فضای وزنها بدنبال برداری می گردد که خطا را حداقل کند. این الگوریتم از یک مقدار دلخواه برای بردار وزن شروع کرده و در هر مرحله وزنها را طوری تغییر می دهد که در جهت شیب کاهشی منحنی فوق خطا کاهش داده شود.



به دست آوردن قانون gradient descent

- ایده اصلی : گرادیان همواره در جهت افزایش شیب E عمل میکند.
- گرادیان E نسبت به بردار وزن w بصورت زیر تعریف می شود:

$$\nabla E(W) = [\, E'/w_0,\ E'/w_1,\ \ldots,\ E'/w_n \,]$$

- که در آن $\nabla E(W)$ یک بردار و E' مشتق جزئی نسبت به هر وزن میباشد.

Delta Rule قانون دلتا

- برای یک مثال آموزشی $X = (x_1 , x_2, ..., x_n)$ در هر مرحله وزنها بر اساس قانون دلتا بصورت زیر تغییر میکند:

$$w_i = w_i + \Delta w_i$$

Where $\Delta w_i = -\eta \ E'(\mathbf{W}) / w_i$

η: learning rate (e.g., 0.1, $0 < \eta <= 1$)

محاسبه گرادیان

- با مشتق گیری جزئی از رابطه خطا میتوان به سادگی گرادیان را محاسبه نمود:

$$if \quad o_j = \sum w_{ji} x_i$$

$$E(w_{ji}) = 1/2\ (t_j - o_j)^2$$

$$E(w_{ji}) = 1/2\ (t_j^2 - 2\ t_j\ o_j + o_j^2)$$

$$E'(w_{ji}) = \partial E / \partial w_{ji} = \partial E / \partial o_j \,.\, \partial o_j / \partial w_{ji}$$

$$= (o_j - t_j) \,.\, x_i = -(t_j - o_j) \,.\, x_i$$

محاسبه گرادیان

The minus sign is eliminated to enable us to move the weight in the negative direction of the gradient to minimize error. Therefore:

$$\Delta w_i = \eta \Sigma_i (t_i - O_i) x_i$$

خلاصه یادگیری قانون دلتا

الگوریتم یادگیری با استفاده از قانون دلتا بصورت زیر میباشد.

1. به وزنها مقدار تصادفی نسبت دهید
2. تا رسیدن به شرایط توقف مراحل زیر را ادامه دهید
   - هر وزن ∇w_i را با مقدار صفر عدد دهی اولیه کنید.
   - سپس برای هر مثال :وزن ∇w_i را بصورت زیر تغییر دهید:

$$\nabla w_i = \eta (t_i - O_i) x_i$$

مقدار w_i را بصورت زیر تغییر دهید:

$$w_i = w_i + \nabla w_i$$

تا خطا بسیار کوچک شود

مشکلات روش Gradient descent

1. ممکن است همگرا شدن به یک مقدار مینیمم زمان زیادی لازم داشته باشد.
2. اگر در سطح خطا چندین مینیمم محلی وجود داشته باشد تضمینی وجود ندارد که الگوریتم مینیمم مطلق را پیدا بکند.

در ضمن این روش وقتی قابل استفاده است که:

- فضای فرضیه دارای فرضیه های پارامتریک پیوسته باشد.
- رابطه خطا قابل مشتق گیری باشد.