شبکه‌های عصبی مصنوعی

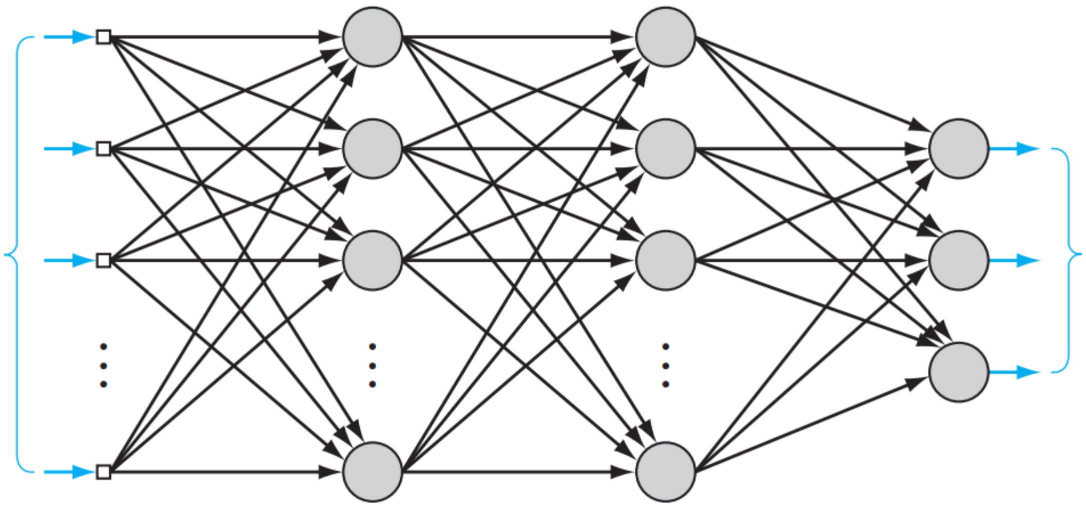
جلسه هفتم:

پرسپترون چند لایه (۲)

(Multi-Layer Perceptron = MLP)

محمد فرخی (farrokhi AT iust.ac.ir)

پرسپترون چندلایه (MLP)



– این الگوریتم که در واقع فرم تعمیم یافته الگوریتم کمترین میانگین مربعات (LMS) است، از دو فاز متمایز از هم تشکیل می‌شود:

۱– فاز پیش‌گذر (Forward pass):

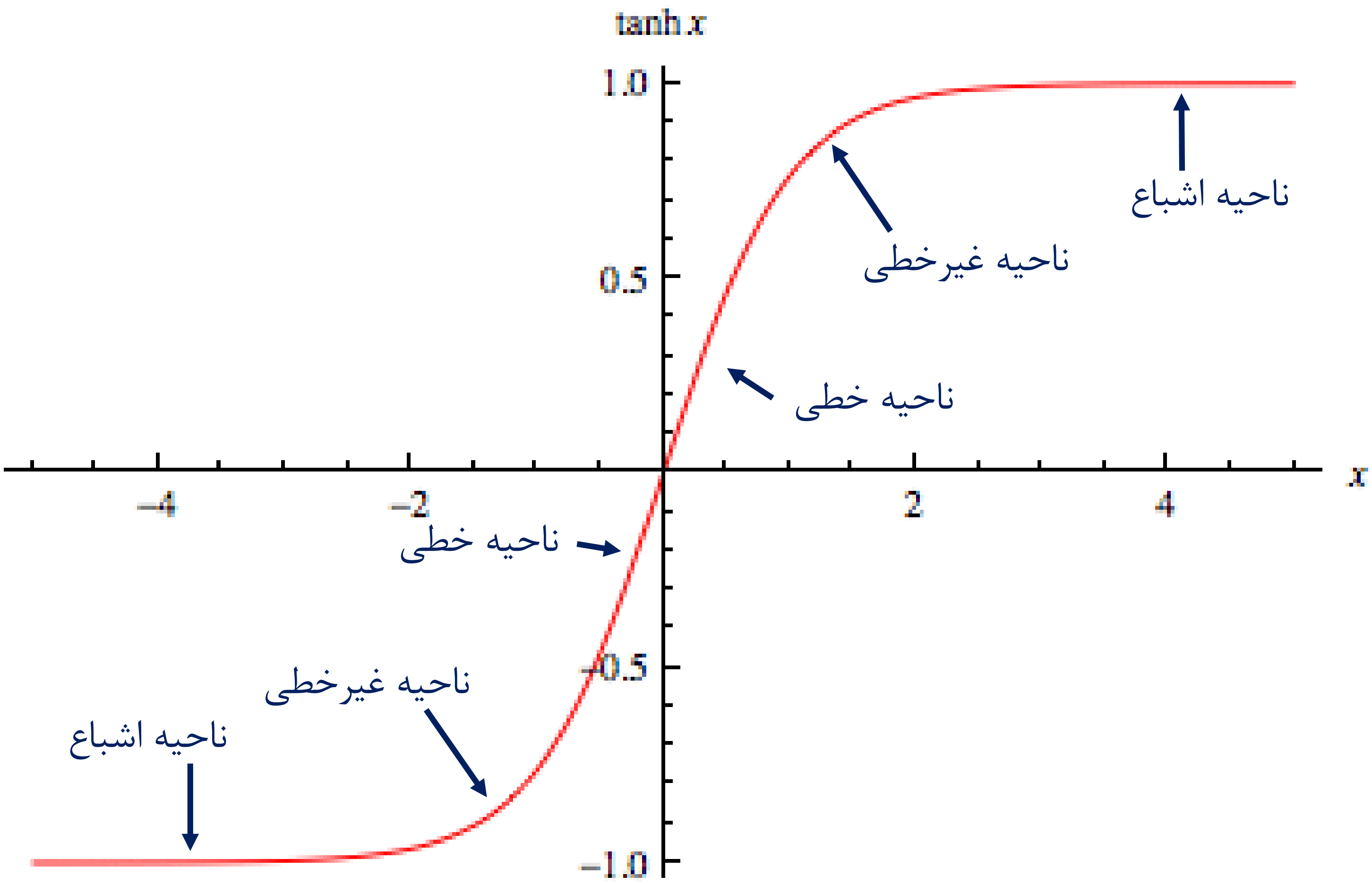
که در آن بردار سیگنال‌ها به لایه ورودی اعمال‌شده و اثر آن در شبکه، لایه–به–لایه محاسبه شده تا در نهایت پاسخ شبکه به‌دست‌آید. در این مرحله، وزن‌های شبکه ثابت است.

۲– فاز پس‌گذر (Backward pass):

که در آن تفاضل بین پاسخ شبکه و پاسخ دلخواه به‌عنوان خطای شبکه محاسبه‌شده و این سیگنال خطا از خروجی به ورودی، به‌صورت لایه–به–لایه انتشار می‌یابد. دراین حالت وزن‌های شبکه تنظیم شده تا پاسخ شبکه به پاسخ دلخواه نزدیک‌شود.

پرسپترون چندلایه (MLP)

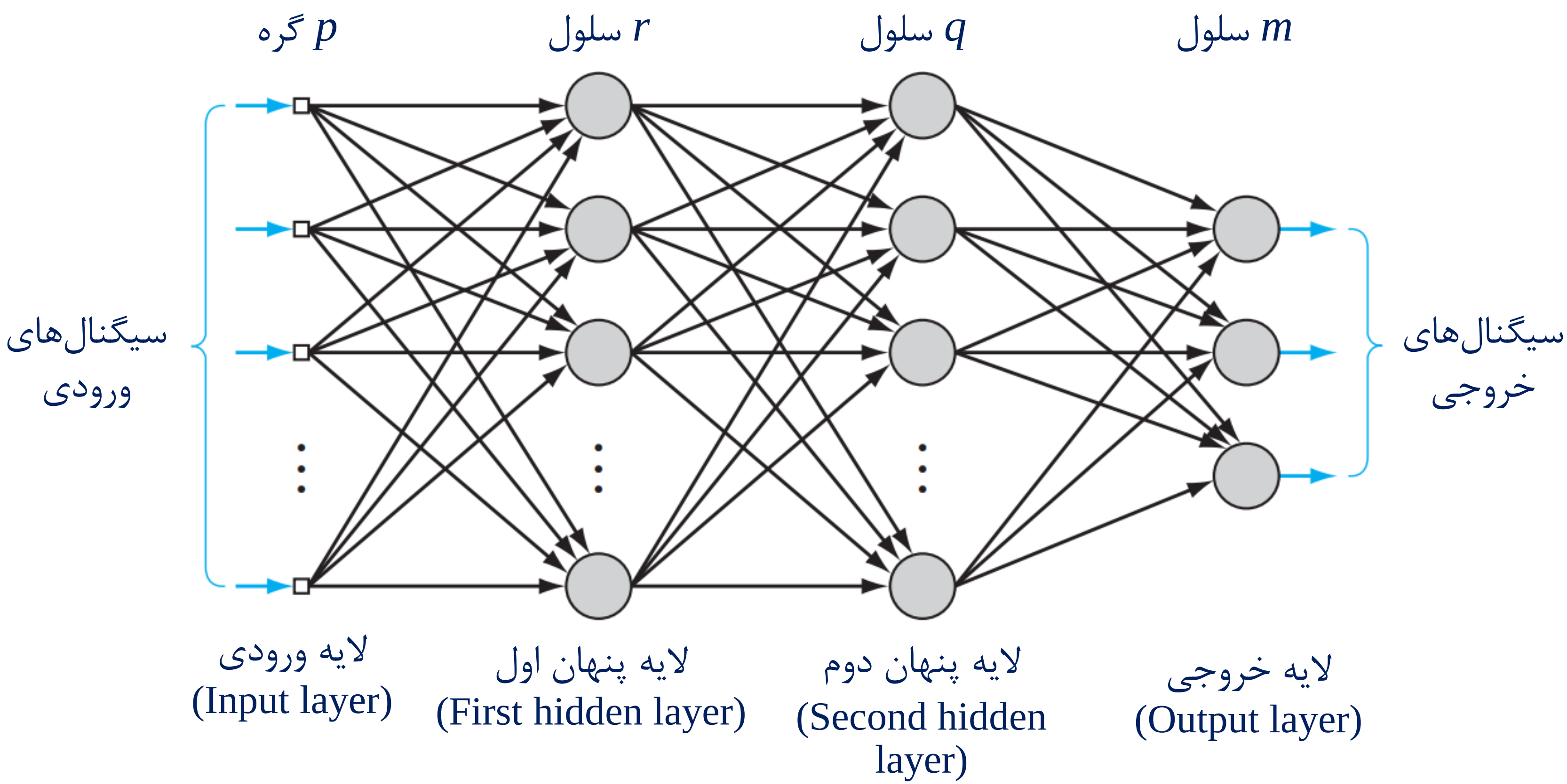
نواحی مختلف توابع S شکل



پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

– این الگوریتم را برای شبکه زیر با دو لایه پنهان اجرا می‌کنیم. اگرچه برای هر تعداد لایه پنهان توسعه‌پذیر است.



پرسپترون چندلایه (MLP)

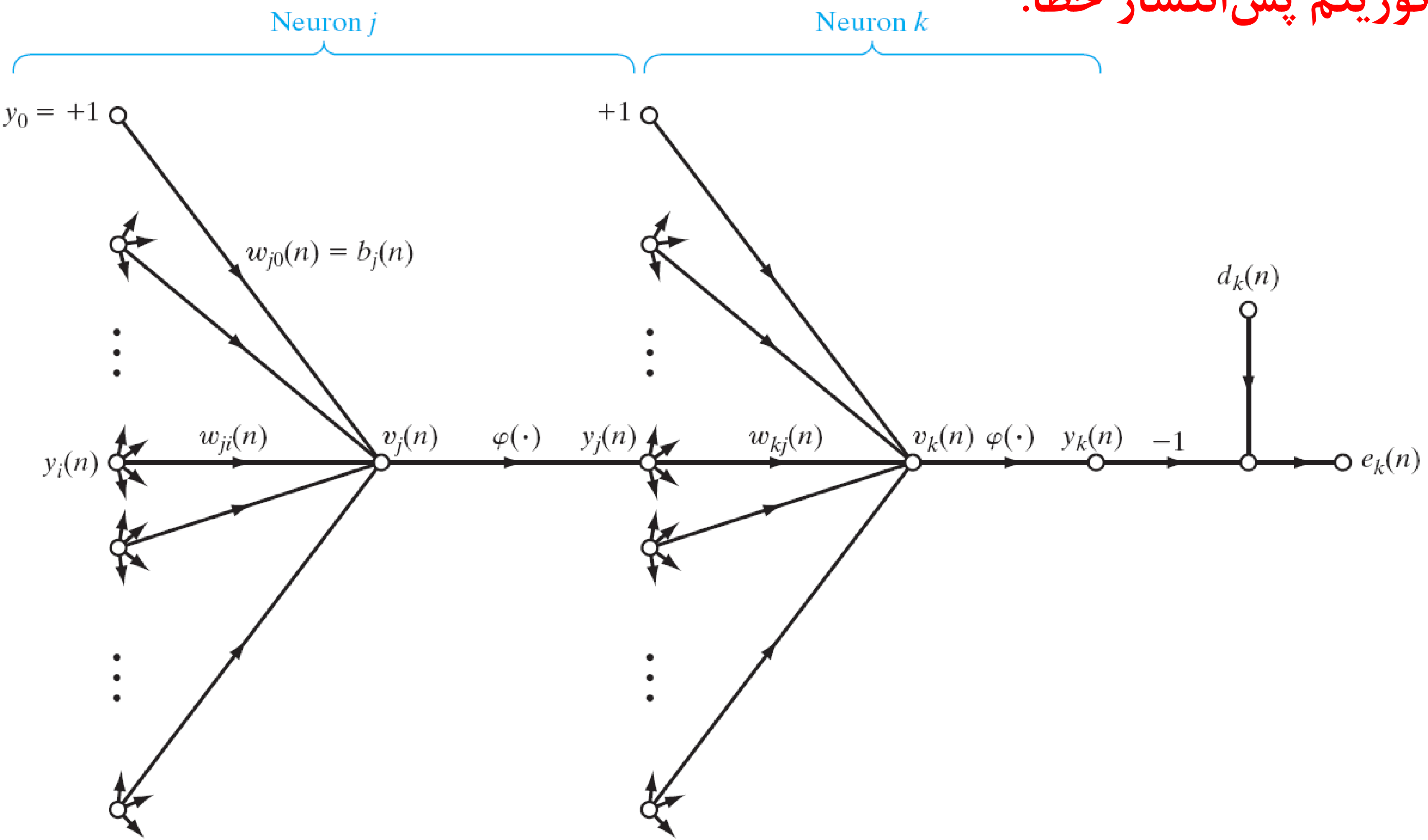
الگوریتم پس‌انتشار خطا:

ابتدا، نمادهای زیر را تعریف می‌کنیم:

- n مرحله تکرار nام است (اعمال الگوی آموزش nام به شبکه)
- $\mathcal{E}(n)$ جمع لحظه‌ای مربعات خطا و $\mathcal{E}_{av}$ میانگین مقادیر $\mathcal{E}(n)$ برای تمام مراحل n
- اندیس‌های i، j و k برای سلول‌های مختلف در شبکه
  سلول j در سمت راست سلول i و سلول k در سمت راست سلول j
- $e_k(n)$، $d_k(n)$ و $y_k(n)$ به‌ترتیب خطای سیگنال خروجی، خروجی دلخواه و خروجی واقعی سلول kام در لایه خروجی
- $w_{ji}(n)$ وزن اتصالی بین سلول i و سلول j در مرحله n
- $w_{j0}(n)$ مقدار آستانه سلول j در مرحله n (متصل به ورودی ثابت $y_0 = +1$)
- $v_j(n)$ جمع خطی سیگنال‌های ورودی به سلول j در مرحله n
- $\varphi_j(v_j(n))$ تابع فعال‌سازی (تابع غیرخطی) بین ورودی و خروجی سلول j در مرحله n
- $x_i(n)$ ورودی iام به شبکه درمرحله n
- η ضریب آموزش

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

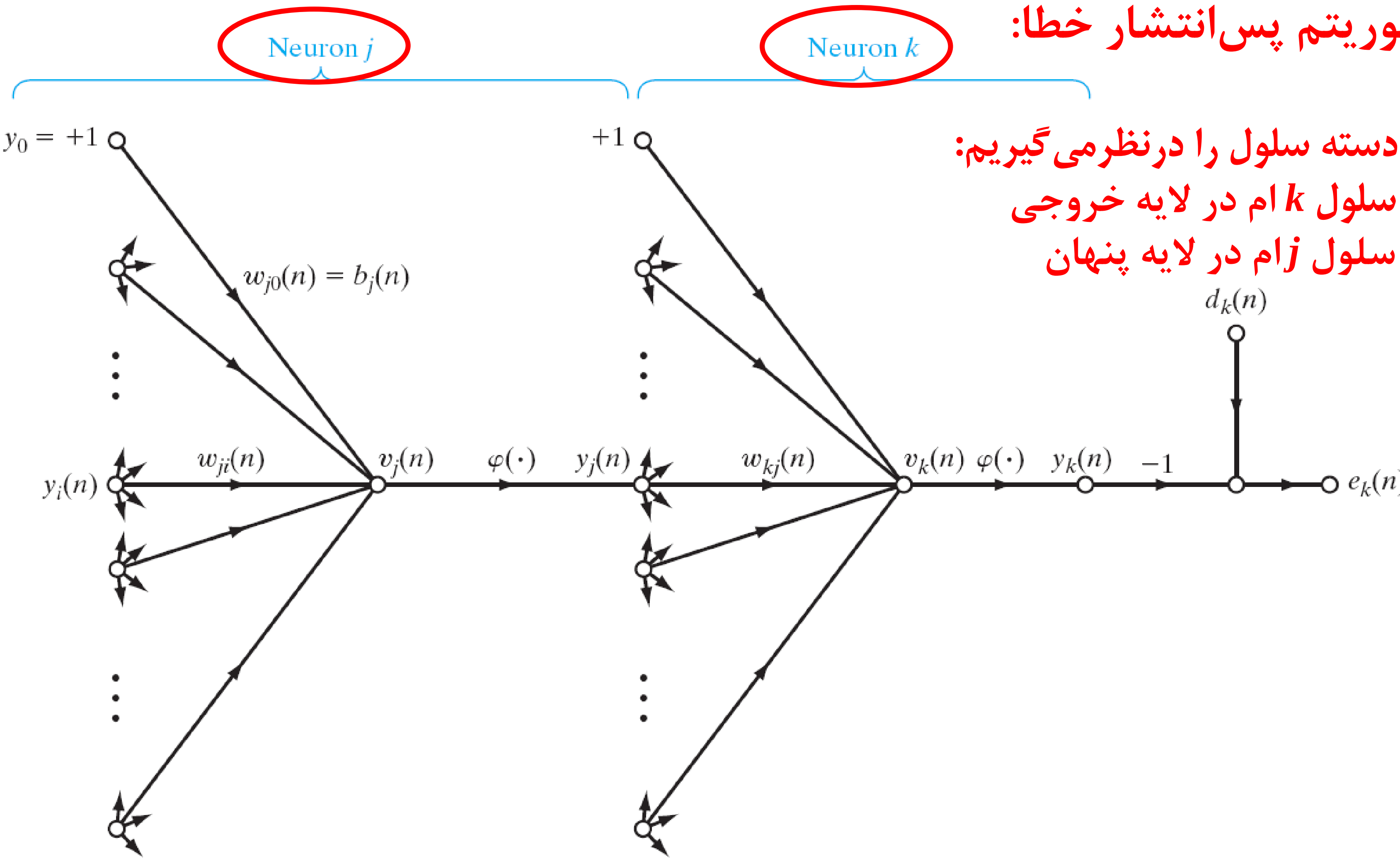


پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

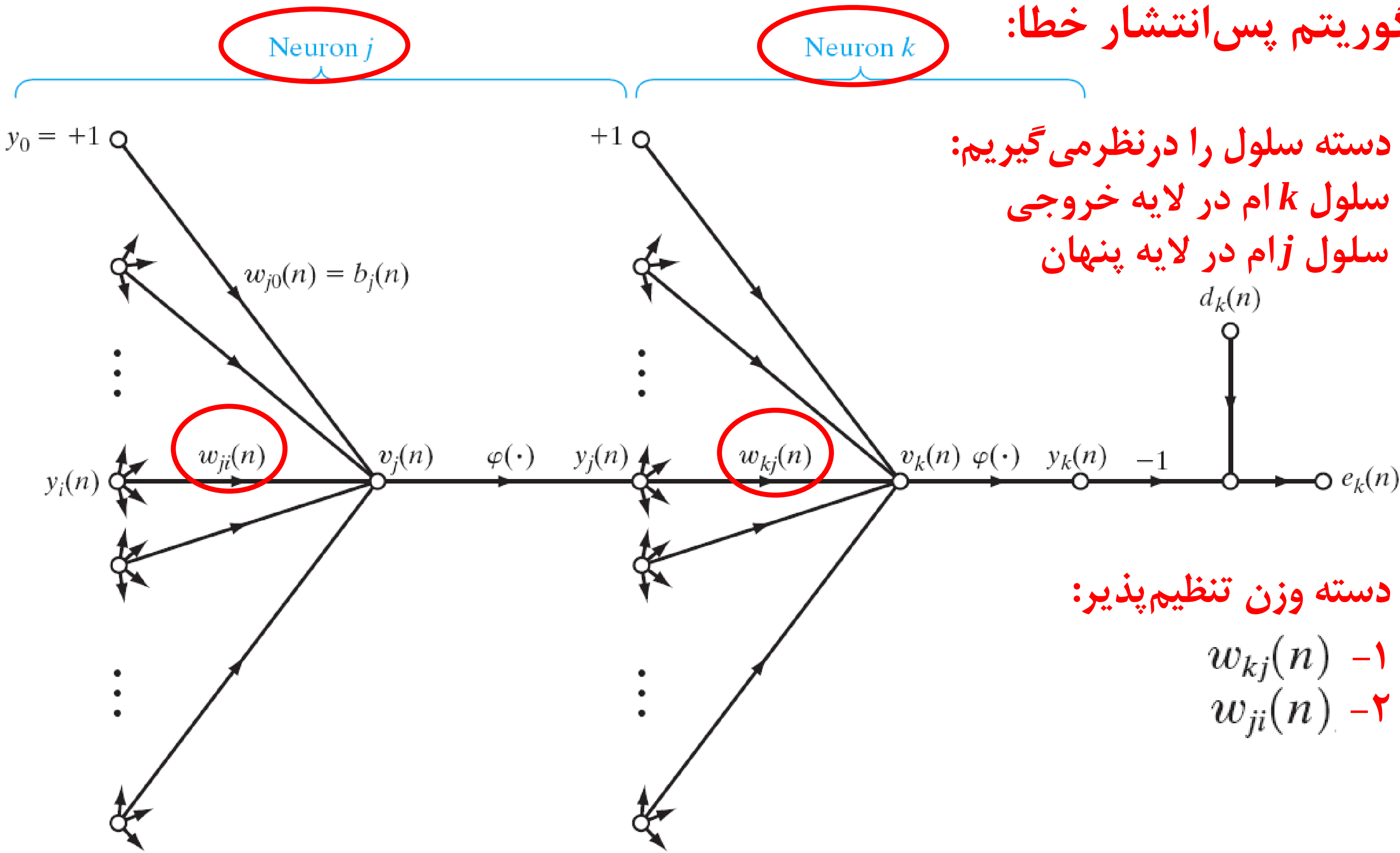
دو دسته سلول را درنظرمی‌گیریم:

- **سلول k ام در لایه خروجی**
- **سلول j ام در لایه پنهان**



پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



دو دسته سلول را درنظرمی‌گیریم:

- سلول ***k*** ام در لایه خروجی
- سلول ***j*** ام در لایه پنهان

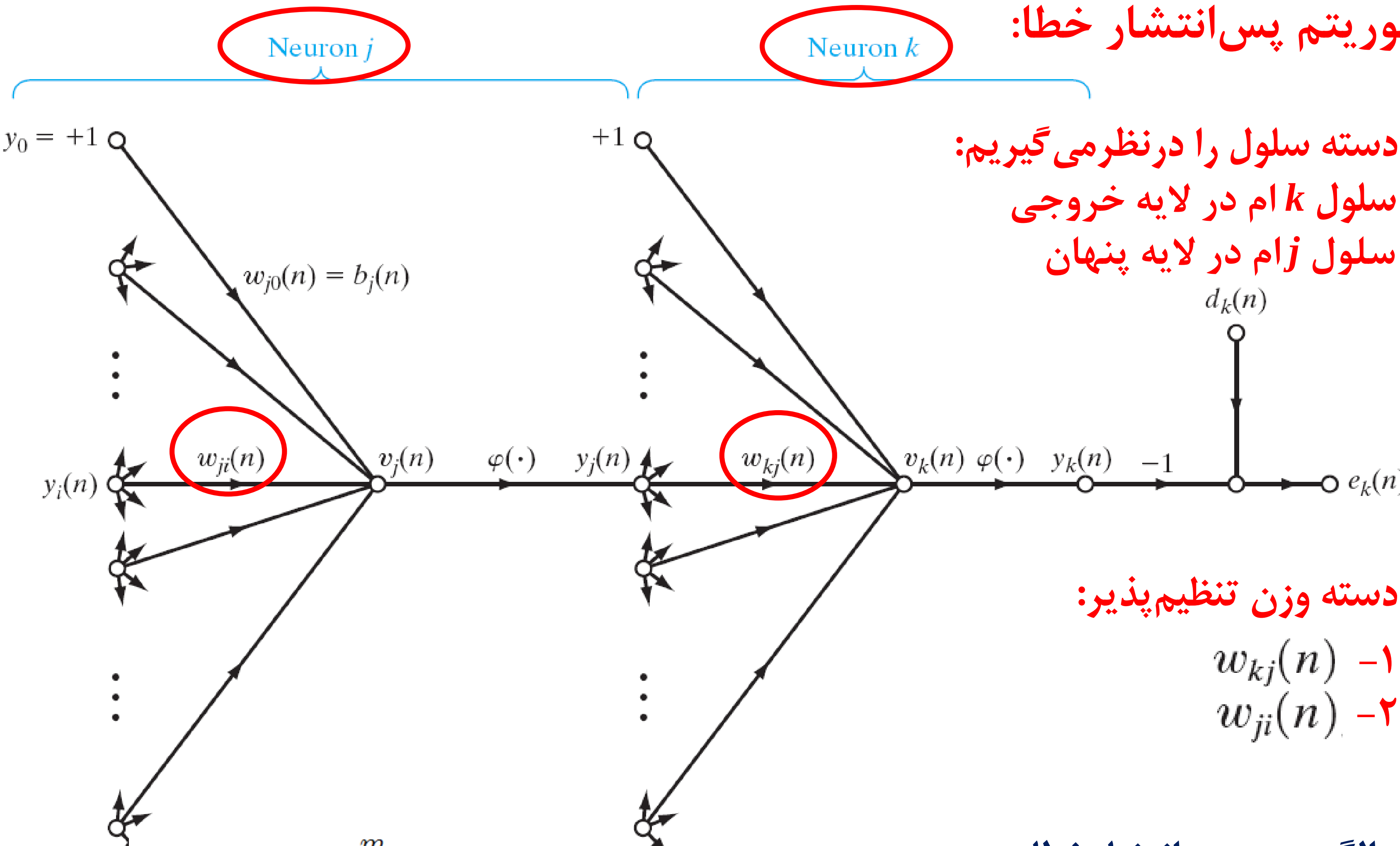
دو دسته وزن تنظیم‌پذیر:

۱- $w_{kj}(n)$

۲- $w_{ji}(n)$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



دو دسته سلول را درنظرمی‌گیریم:

- سلول k ام در لایه خروجی
- سلول j ام در لایه پنهان

دو دسته وزن تنظیم‌پذیر:

۱- $w_{kj}(n)$

۲- $w_{ji}(n)$

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n)$$

هدفِ الگوریتم پس‌انتشارخطا:

تنظیم وزن‌ها برای کاهش «مجموع مربعات خطای لحظه‌ای»

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

برای کمینه‌کردن این خطا، همانند روش LMS از آن نسبت به وزن مورد نظر مشتق‌گرفته تا تغییرات در وزن‌ها به‌دست آید.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

برای کمینه‌کردن این خطا، همانند روش LMS از آن نسبت به وزن مورد نظر مشتق‌گرفته تا تغییرات در وزن‌ها به‌دست آید.

۱- وزن‌های $w_{kj}(n)$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

برای کمینه‌کردن این خطا، همانند روش LMS از آن نسبت به وزن مورد نظر مشتق‌گرفته تا تغییرات در وزن‌ها به‌دست آید.

۱- وزن‌های $w_{kj}(n)$

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

برای کمینه‌کردن این خطا، همانند روش LMS از آن نسبت به وزن مورد نظر مشتق‌گرفته تا تغییرات در وزن‌ها به‌دست آید.

۱- وزن‌های $w_{kj}(n)$

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum\nolimits_{j=0}^{q} w_{kj}(n) y_j(n)$$

با استفاده از قاعده مشتق زنجیره‌ای

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = \frac{\partial E(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial w_{kj}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

برای کمینه‌کردن این خطا، همانند روش LMS از آن نسبت به وزن مورد نظر مشتق‌گرفته تا تغییرات در وزن‌ها به‌دست آید.

۱- وزن‌های $w_{kj}(n)$

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

با استفاده از قاعده مشتق زنجیره‌ای

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = \frac{\partial E(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial w_{kj}(n)}$$

$$\Downarrow$$

$$e_k(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

برای کمینه‌کردن این خطا، همانند روش LMS از آن نسبت به وزن مورد نظر مشتق‌گرفته تا تغییرات در وزن‌ها به‌دست آید.

۱- وزن‌های $w_{kj}(n)$

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum\nolimits_{j=0}^{q} w_{kj}(n) y_j(n)$$

با استفاده از قاعده مشتق زنجیره‌ای

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = \underset{e_k(n)}{\underset{\Downarrow}{\frac{\partial E(n)}{\partial e_k(n)}}} \; \underset{-1}{\underset{\Downarrow}{\frac{\partial e_k(n)}{\partial y_k(n)}}} \; \frac{\partial y_k(n)}{\partial v_k(n)} \; \frac{\partial v_k(n)}{\partial w_{kj}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

برای کمینه‌کردن این خطا، همانند روش LMS از آن نسبت به وزن مورد نظر مشتق‌گرفته تا تغییرات در وزن‌ها به‌دست آید.

۱- وزن‌های $w_{kj}(n)$

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

با استفاده از قاعده مشتق زنجیره‌ای

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = \underbrace{\frac{\partial E(n)}{\partial e_k(n)}}_{\Downarrow\; e_k(n)} \underbrace{\frac{\partial e_k(n)}{\partial y_k(n)}}_{\Downarrow\; -1} \underbrace{\frac{\partial y_k(n)}{\partial v_k(n)}}_{\Downarrow\; \varphi_k'(v_k(n))} \frac{\partial v_k(n)}{\partial w_{kj}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n) \qquad e_k(n) = d_k(n) - y_k(n)$$

برای کمینه‌کردن این خطا، همانند روش LMS از آن نسبت به وزن مورد نظر مشتق‌گرفته تا تغییرات در وزن‌ها به‌دست آید.

۱- وزن‌های $w_{kj}(n)$

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

با استفاده از قاعده مشتق زنجیره‌ای

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = \underbrace{\frac{\partial E(n)}{\partial e_k(n)}}_{e_k(n)} \underbrace{\frac{\partial e_k(n)}{\partial y_k(n)}}_{-1} \underbrace{\frac{\partial y_k(n)}{\partial v_k(n)}}_{\varphi_k'(v_k(n))} \underbrace{\frac{\partial v_k(n)}{\partial w_{kj}(n)}}_{y_j(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi_k'(v_k(n))y_j(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi_k'(v_k(n))y_j(n)$$

قاعده گرادیان نزولی:

$$\Delta w_{kj}(n) = -\eta \frac{\partial E(n)}{\partial w_{kj}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi_k'(v_k(n))y_j(n)$$

قاعده گرادیان نزولی:

$$\Delta w_{kj}(n) = -\eta \frac{\partial E(n)}{\partial w_{kj}(n)} \quad \Rightarrow \quad \Delta w_{kj}(n) = \eta \; e_k(n)\varphi_k'(v_k(n)) \; y_j(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi_k'(v_k(n))y_j(n)$$

قاعده گرادیان نزولی:

$$\Delta w_{kj}(n) = -\eta \frac{\partial E(n)}{\partial w_{kj}(n)} \quad \Rightarrow \quad \Delta w_{kj}(n) = \eta \underbrace{e_k(n)\varphi_k'(v_k(n))}_{\delta_k(n)} y_j(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi_k'(v_k(n))y_j(n)$$

قاعده گرادیان نزولی:

$$\Delta w_{kj}(n) = -\eta\frac{\partial E(n)}{\partial w_{kj}(n)} \quad \Rightarrow \quad \Delta w_{kj}(n) = \eta\ \underbrace{e_k(n)\varphi_k'(v_k(n))}_{\delta_k(n)}\ y_j(n)$$

$$\Delta w_{kj}(n) = \eta\ \delta_k(n)y_j(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- وزن‌های $w_{ji}(n)$.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- وزن‌های $w_{ji}(n)$،

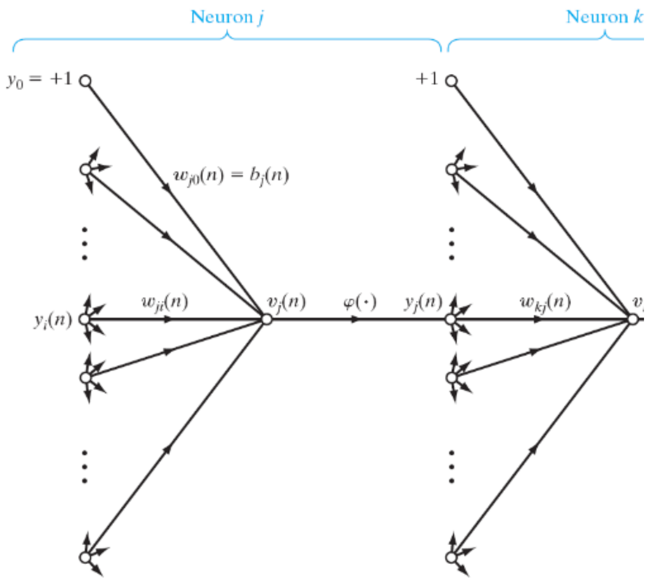
$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- وزن‌های $w_{ji}(n)$،



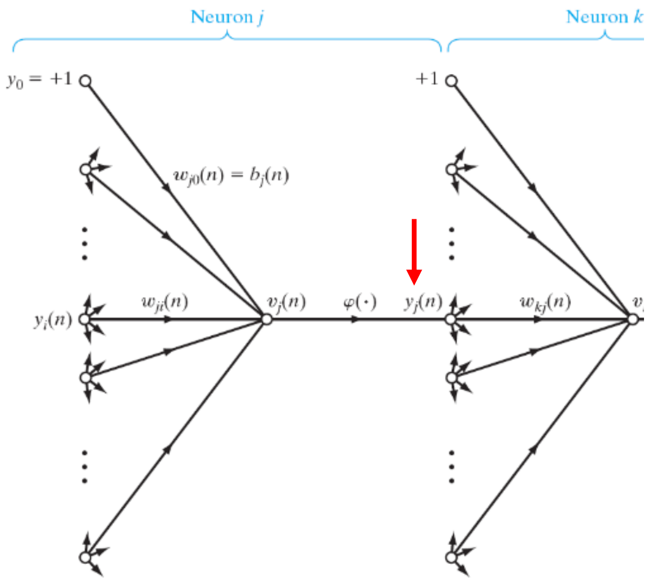
$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- وزن‌های $w_{ji}(n)$،



$$y_k(n) = \varphi_k(v_k(n))$$

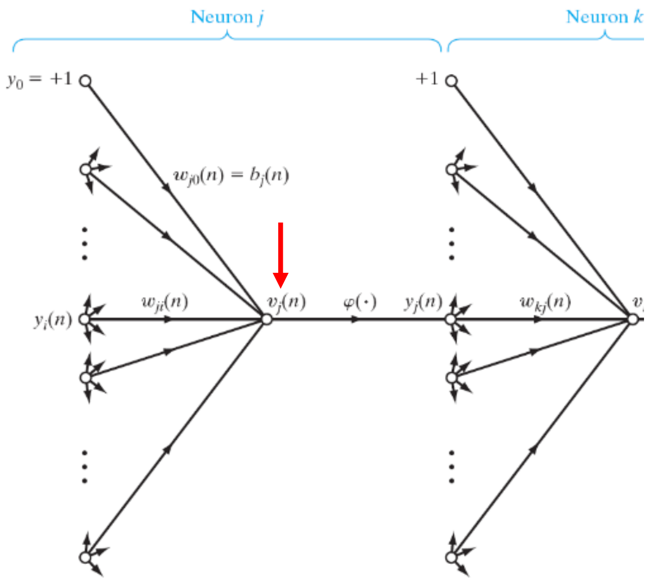
$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

$$y_j(n) = \varphi_j(v_j(n))$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- وزن‌های $w_{ji}(n)$،



$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

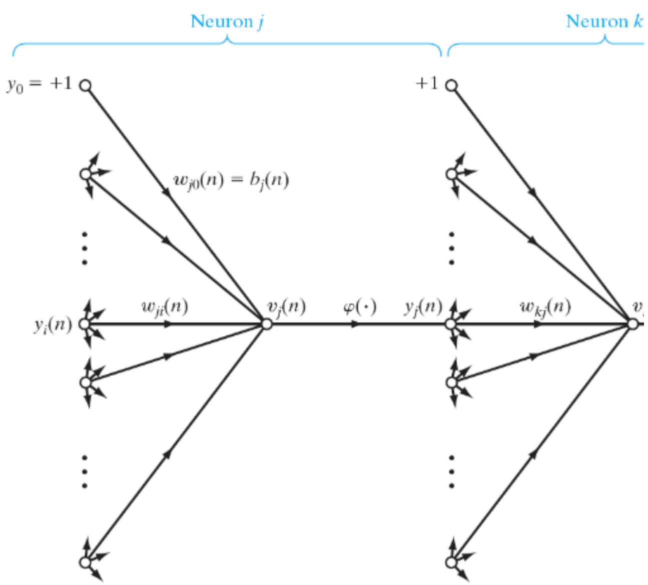
$$y_j(n) = \varphi_j(v_j(n))$$

$$v_j(n) = \sum_{i=0}^{r} w_{ji}(n) y_i(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- وزن‌های $w_{ji}(n)$،



$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

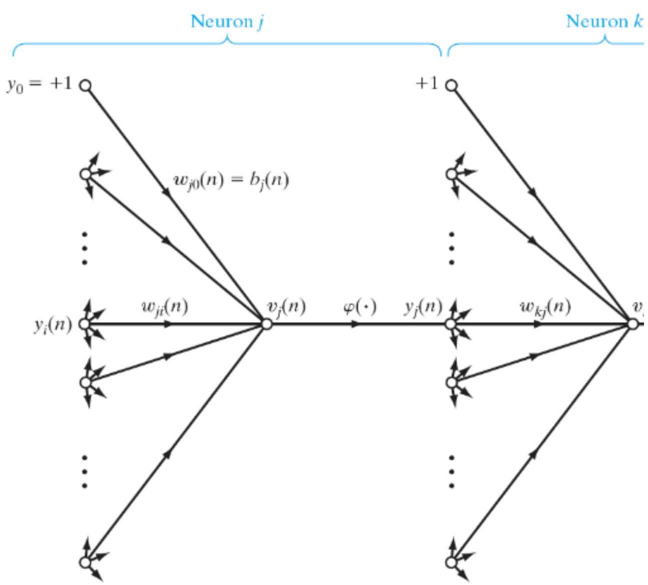
$$y_j(n) = \varphi_j(v_j(n))$$

$$v_j(n) = \sum_{i=0}^{r} w_{ji}(n) y_i(n)$$

$$y_k(n) = \varphi_k\left\{\sum_{j=0}^{q} w_{kj}(n)\, \varphi_j\left[\sum_{i=0}^{r} w_{ji}(n) y_i(n)\right]\right\}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



۲- وزن‌های $w_{ji}(n)$،

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

$$y_j(n) = \varphi_j(v_j(n))$$

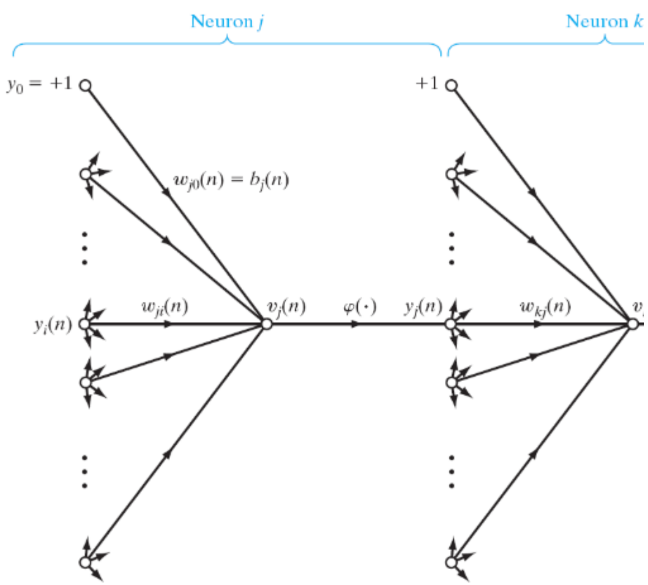
$$v_j(n) = \sum_{i=0}^{r} w_{ji}(n) y_i(n)$$

$$y_k(n) = \varphi_k\left\{\sum_{j=0}^{q} w_{kj}(n)\, \varphi_j\left[\sum_{i=0}^{r} w_{ji}(n) y_i(n)\right]\right\}$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



۲- وزن‌های $w_{ji}(n)$،

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

$$y_j(n) = \varphi_j(v_j(n))$$

$$v_j(n) = \sum_{i=0}^{r} w_{ji}(n) y_i(n)$$

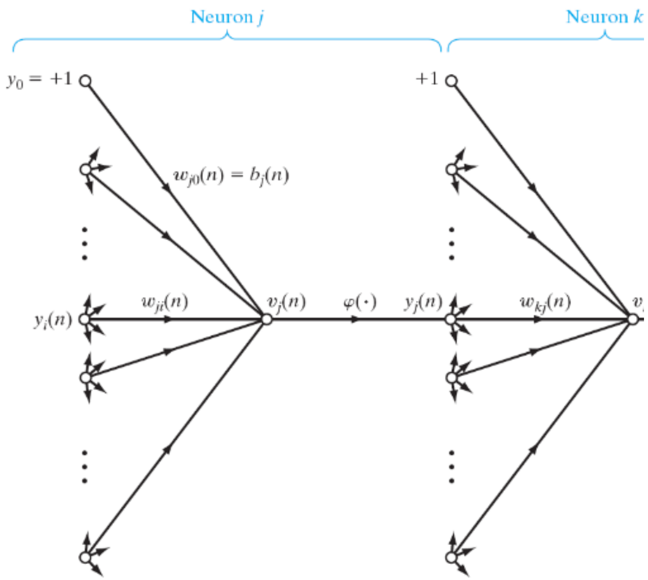
$$y_k(n) = \varphi_k\left\{\sum_{j=0}^{q} w_{kj}(n)\, \varphi_j\left[\sum_{i=0}^{r} w_{ji}(n) y_i(n)\right]\right\}$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \underbrace{\frac{\partial E(n)}{\partial e_k(n)}}_{e_k(n)} \underbrace{\frac{\partial e_k(n)}{\partial y_k(n)}}_{-1} \underbrace{\frac{\partial y_k(n)}{\partial v_k(n)}}_{\varphi_k'(v_k(n))} \underbrace{\frac{\partial v_k(n)}{\partial y_j(n)}}_{y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- وزن‌های $w_{ji}(n)$،



$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

$$y_j(n) = \varphi_j(v_j(n))$$

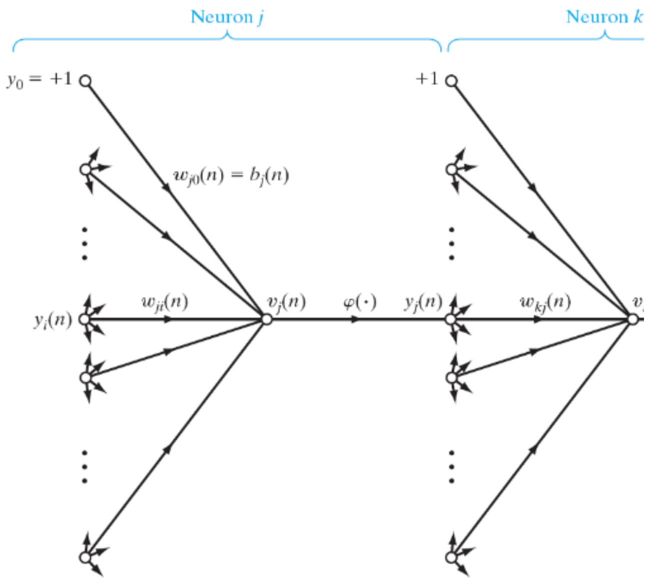
$$v_j(n) = \sum_{i=0}^{r} w_{ji}(n) y_i(n)$$

$$y_k(n) = \varphi_k\left\{\sum_{j=0}^{q} w_{kj}(n)\, \varphi_j\left[\sum_{i=0}^{r} w_{ji}(n) y_i(n)\right]\right\}$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \underbrace{\frac{\partial E(n)}{\partial e_k(n)}}_{\Downarrow\; e_k(n)} \underbrace{\frac{\partial e_k(n)}{\partial y_k(n)}}_{\Downarrow\; -1} \underbrace{\frac{\partial y_k(n)}{\partial v_k(n)}}_{\Downarrow\; \varphi'_k(v_k(n))} \underbrace{\frac{\partial v_k(n)}{\partial y_j(n)}}_{\Downarrow\; y_j(n)} \underbrace{\frac{\partial y_j(n)}{\partial v_j(n)}}_{\Downarrow\; \varphi'_j(v_j(n))} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



۲- وزن‌های $w_{ji}(n)$،

$$y_k(n) = \varphi_k(v_k(n))$$

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

$$y_j(n) = \varphi_j(v_j(n))$$

$$v_j(n) = \sum_{i=0}^{r} w_{ji}(n) y_i(n)$$

$$y_k(n) = \varphi_k\left\{\sum_{j=0}^{q} w_{kj}(n)\, \varphi_j\left[\sum_{i=0}^{r} w_{ji}(n) y_i(n)\right]\right\}$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \underbrace{\frac{\partial E(n)}{\partial e_k(n)}}_{\Downarrow\; e_k(n)} \underbrace{\frac{\partial e_k(n)}{\partial y_k(n)}}_{\Downarrow\; -1} \underbrace{\frac{\partial y_k(n)}{\partial v_k(n)}}_{\Downarrow\; \varphi_k'(v_k(n))} \underbrace{\frac{\partial v_k(n)}{\partial y_j(n)}}_{\Downarrow\; y_j(n)} \underbrace{\frac{\partial y_j(n)}{\partial v_j(n)}}_{\Downarrow\; \varphi_j'(v_j(n))} \underbrace{\frac{\partial v_j(n)}{\partial w_{ji}(n)}}_{\Downarrow\; y_i(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\sum_{k=1}^{m} e_k(n)\ \varphi'_k(v_k(n))\ w_{kj}(n)\ \varphi'_j(v_j(n))\ y_i(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\sum_{k=1}^{m} \underbrace{e_k(n)\ \varphi'_k(v_k(n))}_{\delta_k(n)}\ w_{kj}(n)\ \varphi'_j(v_j(n))\ y_i(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\sum_{k=1}^{m} \underbrace{e_k(n)\ \varphi'_k(v_k(n))}_{\delta_k(n)}\ w_{kj}(n)\ \varphi'_j(v_j(n))\ y_i(n)$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\sum_{k=1}^{m} \delta_k(n)\ w_{kj}(n)\ \varphi'_j(v_j(n))\ y_i(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\sum_{k=1}^{m} \underbrace{e_k(n)\ \varphi_k'(v_k(n))}_{\delta_k(n)}\ w_{kj}(n)\ \varphi_j'(v_j(n))\ y_i(n)$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\underbrace{\sum_{k=1}^{m} \delta_k(n)\ w_{kj}(n)\ \varphi_j'(v_j(n))}_{\delta_j(n)}\ y_i(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\sum_{k=1}^{m} \underbrace{e_k(n)\ \varphi_k'(v_k(n))}_{\delta_k(n)}\ w_{kj}(n)\ \varphi_j'(v_j(n))\ y_i(n)$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \underbrace{-\sum_{k=1}^{m} \delta_k(n)\ w_{kj}(n)\ \varphi_j'(v_j(n))}_{\delta_j(n)}\ y_i(n)$$

قاعده گرادیان نزولی:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\sum_{k=1}^{m} \underbrace{e_k(n)\ \varphi_k'(v_k(n))}_{\delta_k(n)}\ w_{kj}(n)\ \varphi_j'(v_j(n))\ y_i(n)$$

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\underbrace{\sum_{k=1}^{m} \delta_k(n)\ w_{kj}(n)\ \varphi_j'(v_j(n))}_{\delta_j(n)}\ y_i(n)$$

قاعده گرادیان نزولی:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \Rightarrow \Delta w_{ji}(n) = \eta\ \delta_j(n) y_i(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\Delta w_{kj}(n) = \eta \, \delta_k(n) y_j(n)$$

$$\Delta w_{ji}(n) = \eta \, \delta_j(n) y_i(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\left.\begin{aligned}\Delta w_{kj}(n) &= \eta\ \delta_k(n) y_j(n) \\ \Delta w_{ji}(n) &= \eta\ \delta_j(n) y_i(n)\end{aligned}\right\}$$

قاعده دلتا

(Delta rule)

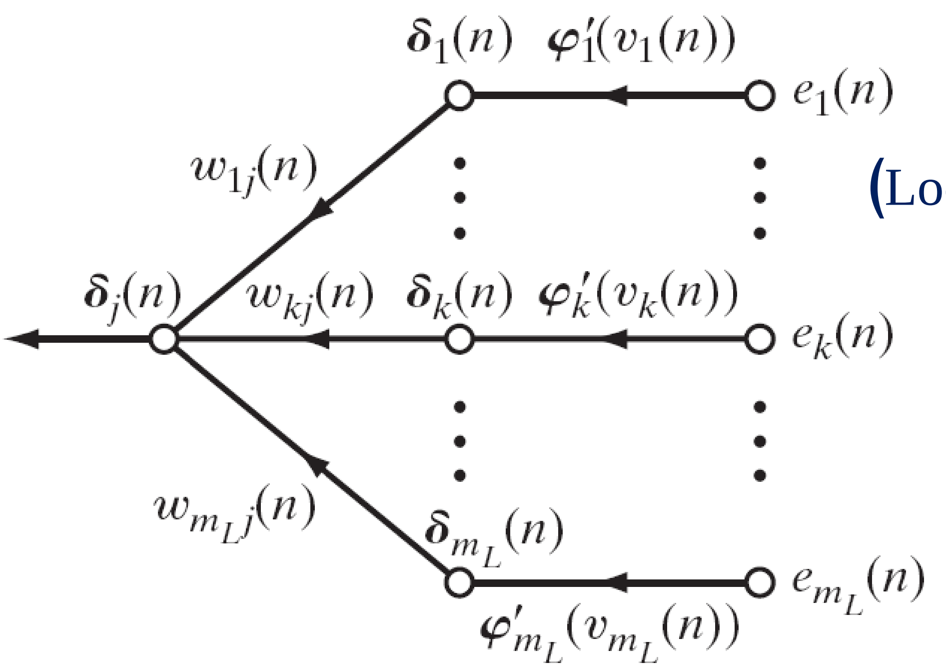
پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

$$\Delta w_{kj}(n) = \eta\, \delta_k(n) y_j(n)$$

$$\Delta w_{ji}(n) = \eta\, \delta_j(n) y_i(n)$$

قاعده دلتا
(Delta rule)



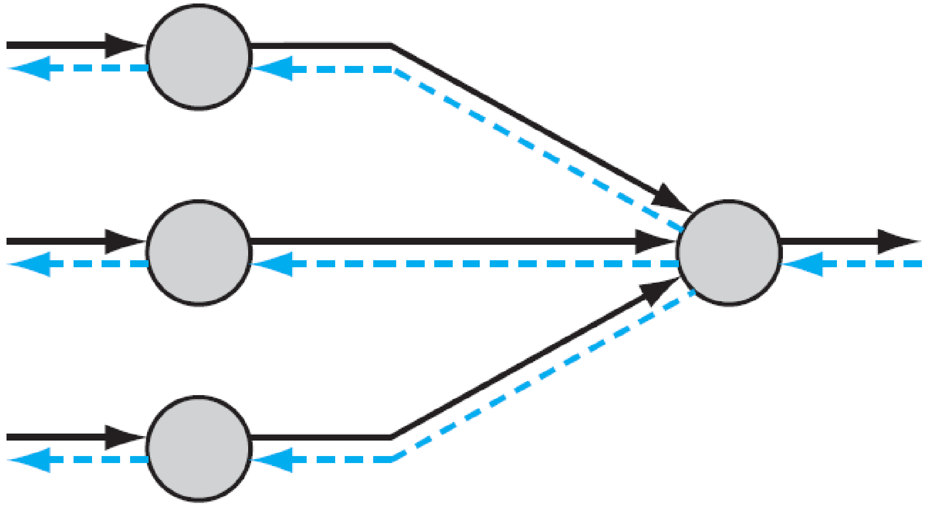
به دلتاها گرادیان‌های محلی (Local gradient) نیز می‌گویند زیرا برای هر سلولی جداگانه محاسبه می‌شود.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

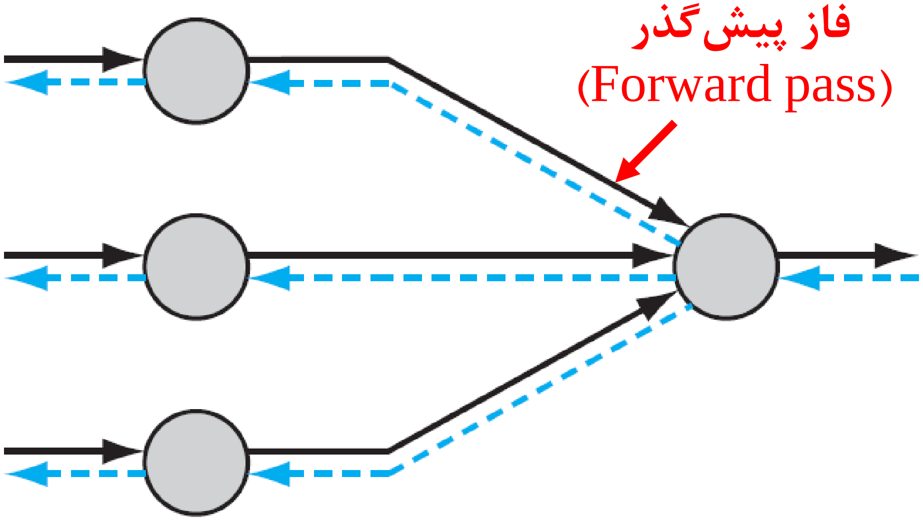
پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



پرسپترون چندلایه (MLP)

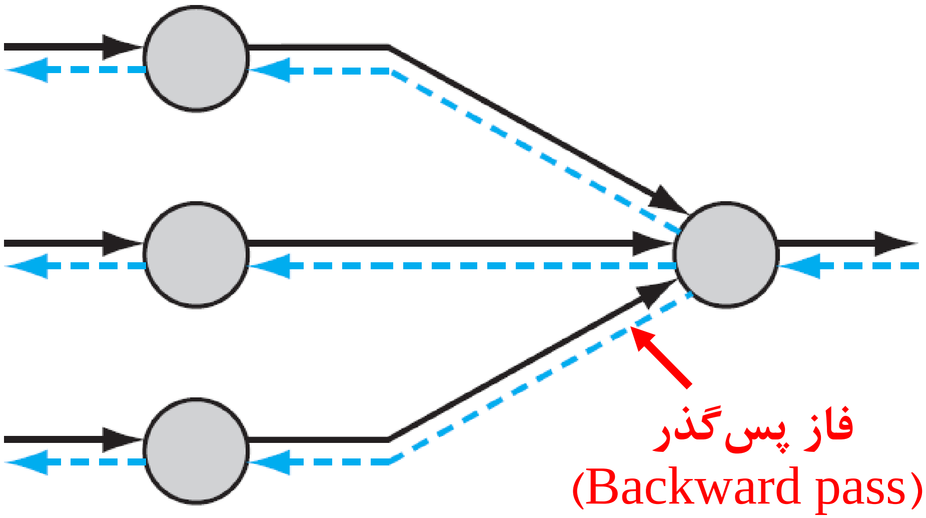
الگوریتم پس‌انتشار خطا:



$$
\begin{cases}
v_j(n) = \sum_{i=0}^{r} w_{ji}(n) y_i(n) & \forall j = 1, \ldots, q \\
y_j(n) = \varphi_j(v_j(n)) & \forall j = 1, \ldots, q \\
v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n) & \forall k = 1, \ldots, m \\
y_k(n) = \varphi_k(v_k(n)) & \forall k = 1, \ldots, m \\
e_k(n) = d_k(n) - y_k(n) & \forall k = 1, \ldots, m
\end{cases}
$$

پرسپترون چندلایه (MLP)

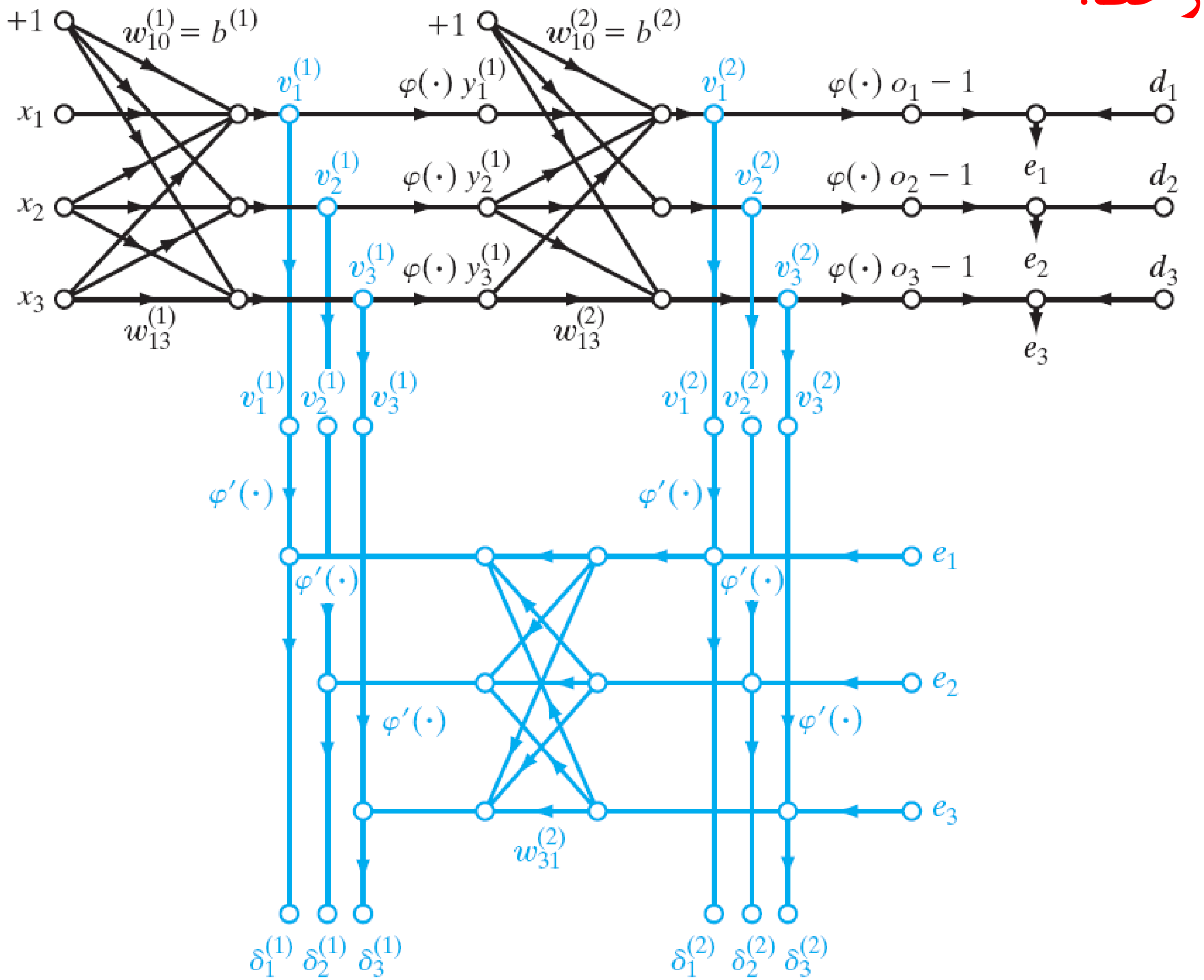
الگوریتم پس‌انتشار خطا:



$$\begin{cases} \delta_k(n) = \varphi_k'(v_k(n))e_k(n) & \forall k = 1, \ldots, m \\ \Delta w_{kj}(n) = \eta\ \delta_k(n) y_j(n) & \begin{cases} \forall k = 1, \ldots, m \\ \forall j = 0, \ldots, q \end{cases} \\ \delta_j(n) = \varphi_j'(v_j(n)) \sum_{k=1}^{m} w_{kj}(n)\delta_k(n) & \forall j = 1, \ldots, q \\ \Delta w_{ji}(n) = \eta\ \delta_j(n) y_i(n) & \begin{cases} \forall j = 1, \ldots, q \\ \forall i = 0, \ldots, r \end{cases} \end{cases}$$

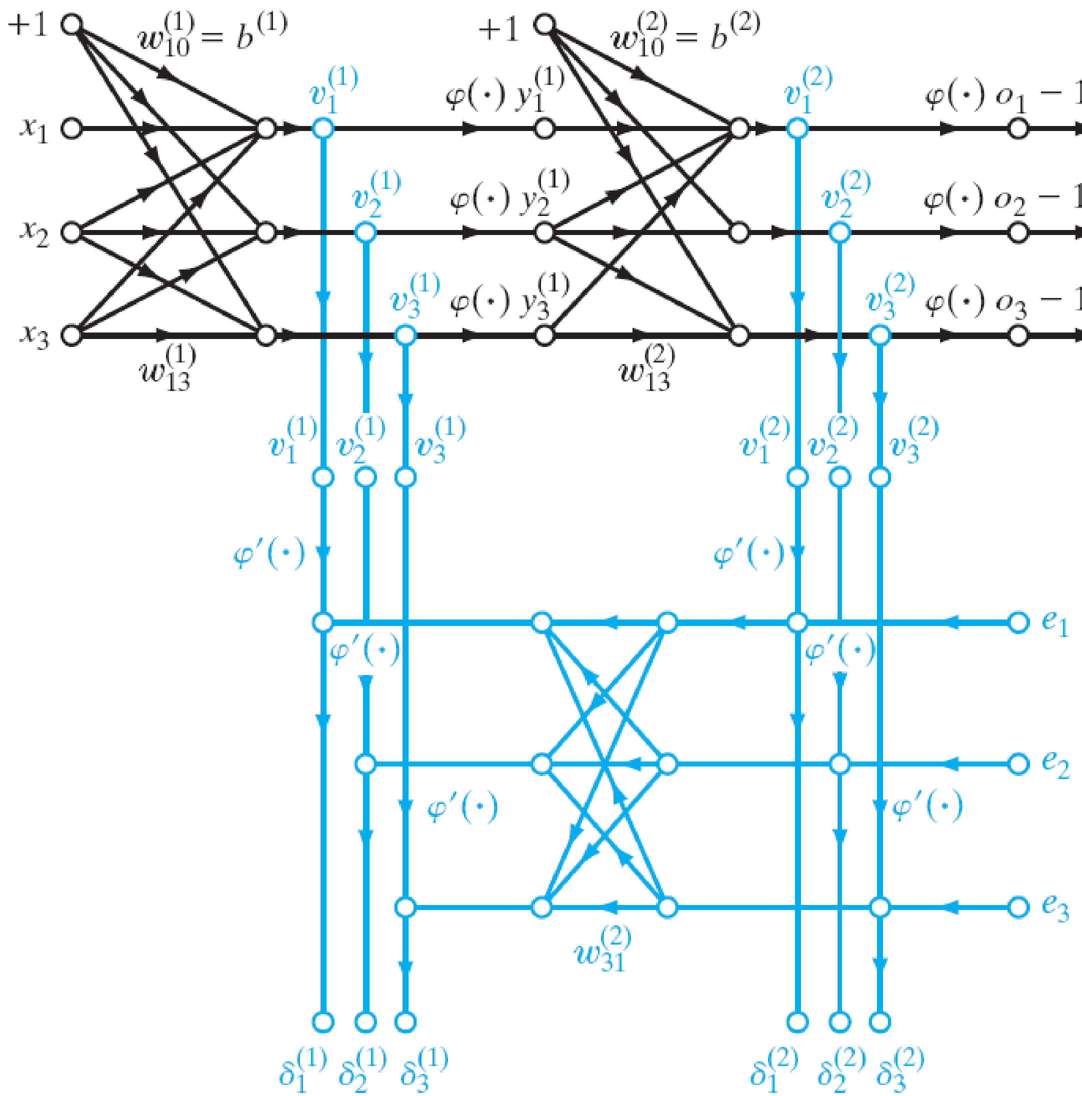
پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

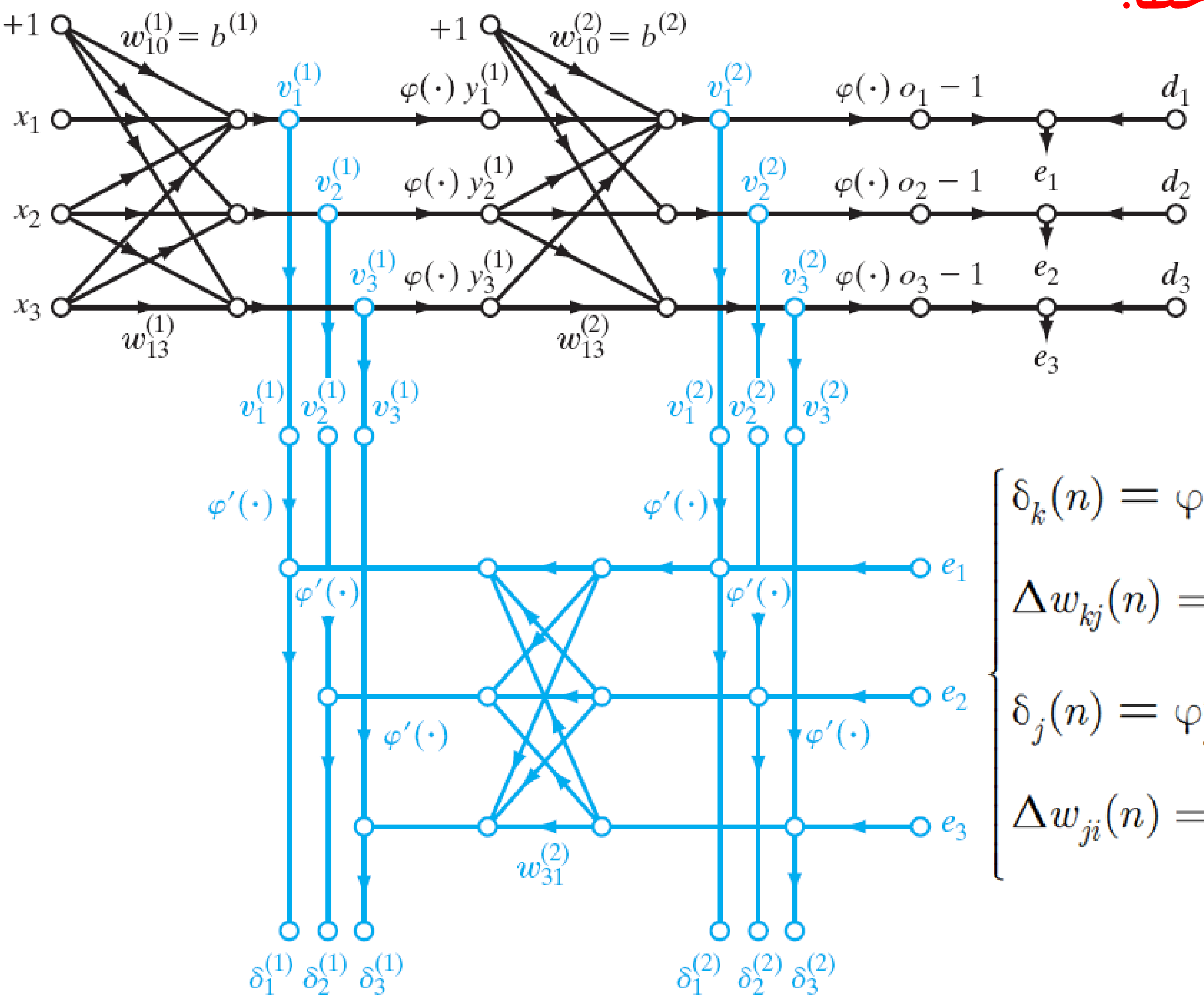


فاز پیش‌گذر
(Forward pass)

$$
\begin{cases}
v_j(n) = \sum_{i=0}^{r} w_{ji}(n) y_i(n) \\
y_j(n) = \varphi_j(v_j(n)) \\
v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n) \\
y_k(n) = \varphi_k(v_k(n)) \\
e_k(n) = d_k(n) - y_k(n)
\end{cases}
$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



فاز پس‌گذر

(Backward pass)

$$\begin{cases} \delta_k(n) = \varphi_k'(v_k(n))e_k(n) \\ \Delta w_{kj}(n) = \eta\ \delta_k(n)y_j(n) \\ \delta_j(n) = \varphi_j'(v_j(n))\sum_{k=1}^{m} w_{kj}(n)\delta_k(n) \\ \Delta w_{ji}(n) = \eta\ \delta_j(n)y_i(n) \end{cases}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

الگوریتم ارایه شده، براساس الگو-به-الگو انجام می‌شود که به آن روش الگویی (Pattern mode) **یا روش آموزش برخط** (Online learning) **می‌گویند.**

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

الگوریتم ارایه شده، براساس الگو-به-الگو انجام می‌شود که به آن روش الگویی (Pattern mode) یا روش آموزش برخط (Online learning) می‌گویند.

مجموعه داده‌ها

$$\mathcal{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

الگوریتم ارایه شده، براساس الگو-به-الگو انجام می‌شود که به آن روش الگویی (Pattern mode) **یا روش آموزش برخط** (Online learning) **می‌گویند.**

مجموعه داده‌ها

$$\mathscr{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N}$$

MLP

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

الگوریتم ارایه شده، براساس الگو–به–الگو انجام می‌شود که به آن روش الگویی (Pattern mode) یا روش آموزش برخط (Online learning) می‌گویند.

مجموعه داده‌ها

$$\mathcal{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N} \Rightarrow \{\mathbf{x}(1), \mathbf{d}(1)\} \Rightarrow \boxed{\text{MLP}}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

الگوریتم ارایه شده، براساس الگو-به-الگو انجام می‌شود که به آن روش الگویی (Pattern mode) یا روش آموزش برخط (Online learning) می‌گویند.

مجموعه داده‌ها

$$\mathscr{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N} \Rightarrow \{\mathbf{x}(2), \mathbf{d}(2)\} \Rightarrow \boxed{\text{MLP}}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

الگوریتم ارایه شده، براساس الگو-به-الگو انجام می‌شود که به آن روش الگویی (Pattern mode) یا روش آموزش برخط (Online learning) می‌گویند.

مجموعه داده‌ها

$$\mathscr{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N} \Rightarrow \{\mathbf{x}(N), \mathbf{d}(N)\} \Rightarrow \boxed{\text{MLP}}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

الگوریتم ارایه شده، براساس الگو-به-الگو انجام می‌شود که به آن روش الگویی (Pattern mode) **یا روش آموزش برخط** (Online learning) **می‌گویند.**

مجموعه داده‌ها

$$\mathscr{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N} \Rightarrow \{\mathbf{x}(N), \mathbf{d}(N)\} \Rightarrow \boxed{\text{MLP}}$$

در واقع، در این‌جا از جمع خطای لحظه‌ای برای تنظیم وزن‌ها استفاده می‌شود

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

الگوریتم ارایه شده، براساس الگو–به–الگو انجام می‌شود که به آن روش الگویی (Pattern mode) یا روش آموزش برخط (Online learning) می‌گویند.

مجموعه داده‌ها

$$\mathscr{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N} \Rightarrow \{\mathbf{x}(N), \mathbf{d}(N)\} \Rightarrow \boxed{\text{MLP}}$$

در واقع، در این‌جا از جمع خطای لحظه‌ای برای تنظیم وزن‌ها استفاده می‌شود

$$E(n) = \frac{1}{2}\sum_{k=1}^{m} e_k^2(n)$$

میانگین خطا برای یک دوره (Epoch)

$$E_{\text{ave}} = \frac{1}{N}\sum_{n=1}^{N} E(n) = \frac{1}{2N}\sum_{n=1}^{N}\sum_{k=1}^{m} e_k^2(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

در مقابل آموزش برخط، روش دسته‌ای (Batch mode) یا آموزش برون‌خط (Offline learning) ارایه شده است.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

در مقابل آموزش برخط، روش دسته‌ای (Batch mode) یا آموزش برون‌خط (Offline learning) ارایه شده است.

مجموعه داده ها

$$\mathscr{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N} \Rightarrow \left\{ \begin{array}{c} \mathbf{x}(1), \mathbf{d}(1) \\ \mathbf{x}(2), \mathbf{d}(2) \\ \vdots \\ \mathbf{x}(N), \mathbf{d}(N) \end{array} \right\} \Rightarrow$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

در مقابل آموزش برخط، روش دسته‌ای (Batch mode) یا آموزش برون‌خط (Offline learning) ارایه شده است.

مجموعه داده ها

$$\mathcal{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N} \Rightarrow \left\{ \begin{array}{c} \mathbf{x}(1), \mathbf{d}(1) \\ \mathbf{x}(2), \mathbf{d}(2) \\ \vdots \\ \mathbf{x}(N), \mathbf{d}(N) \end{array} \right\} \Rightarrow \boxed{\text{MLP}}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

در مقابل آموزش برخط، روش دسته‌ای (Batch mode) یا آموزش برون‌خط (Offline learning) ارایه شده است.

مجموعه داده ها

$$\mathscr{T} = \{\mathbf{x}(n), \mathbf{d}(n)\}_{n=1}^{N} \Rightarrow \left\{ \begin{array}{c} \mathbf{x}(1), \mathbf{d}(1) \\ \mathbf{x}(2), \mathbf{d}(2) \\ \vdots \\ \mathbf{x}(N), \mathbf{d}(N) \end{array} \right\} \Rightarrow \boxed{\text{MLP}}$$

در واقع در این روش، پس از اعمال تمامی داده‌ها و محاسبه خطای میانگین، وزن‌ها به‌روز می‌شوند.

$$E_{\text{ave}} = \frac{1}{N} \sum_{n=1}^{N} E(n) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{k=1}^{m} e_k^2(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

درنتیجه، به‌روز رسانی وزن‌ها دراین دو روش

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

درنتیجه، به‌روز رسانی وزن‌ها دراین دو روش

$$\Delta \hat{w}_{ji} = \frac{1}{N}\sum_{n=1}^{N}\Delta w_{ji}(n)$$

$$= -\frac{1}{N}\eta\sum_{n=1}^{N}\frac{\partial E(n)}{\partial w_{ji}(n)}$$

آموزش برخط

$$= -\frac{1}{N}\eta\sum_{n=1}^{N}e_k(n)\frac{\partial e_k(n)}{\partial w_{ji}(n)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

درنتیجه، به‌روز رسانی وزن‌ها دراین دو روش

آموزش برخط

$$\begin{aligned}\Delta\hat{w}_{ji} &= \frac{1}{N}\sum_{n=1}^{N}\Delta w_{ji}(n)\\ &= -\frac{1}{N}\eta\sum_{n=1}^{N}\frac{\partial E(n)}{\partial w_{ji}(n)}\\ &= -\frac{1}{N}\eta\sum_{n=1}^{N}e_k(n)\frac{\partial e_k(n)}{\partial w_{ji}(n)}\end{aligned}$$

آموزش برون‌خط

$$\begin{aligned}\Delta w_{ji} &= -\eta\,\frac{\partial E_{\text{ave}}}{\partial w_{ji}}\\ &= -\frac{1}{N}\eta\sum_{n=1}^{N}e_k(n)\frac{\partial e_k(n)}{\partial w_{ji}}\end{aligned}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

مزایا و معایب دو روش:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

مزایا و معایب دو روش:

- **آموزش برخط (یا الگویی):**

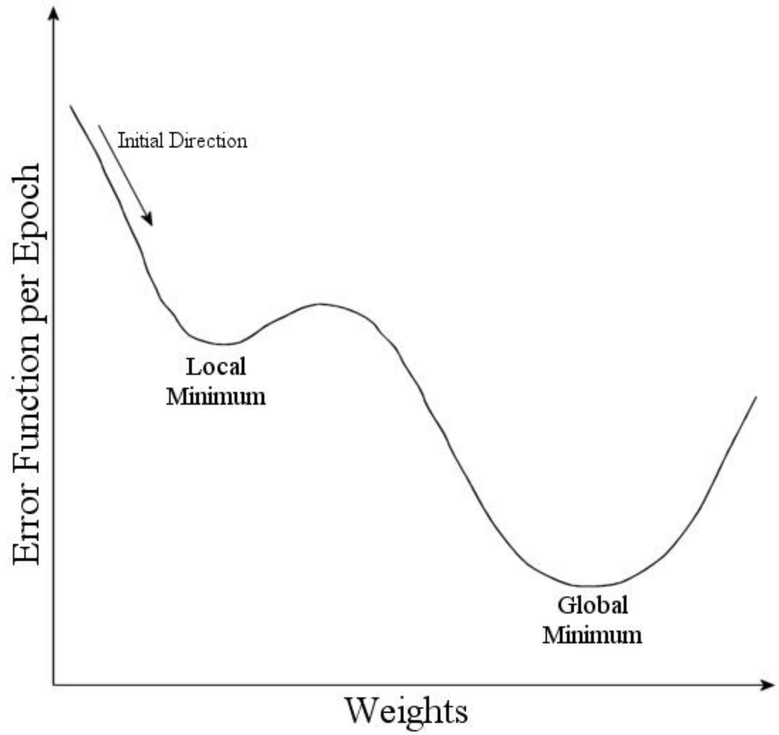
پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

مزایا و معایب دو روش:

– آموزش برخط (یا الگویی):

- امکان خارج‌شدن از کمینه‌های محلی با اعمال نمونه‌ها به‌صورت اتفاقی



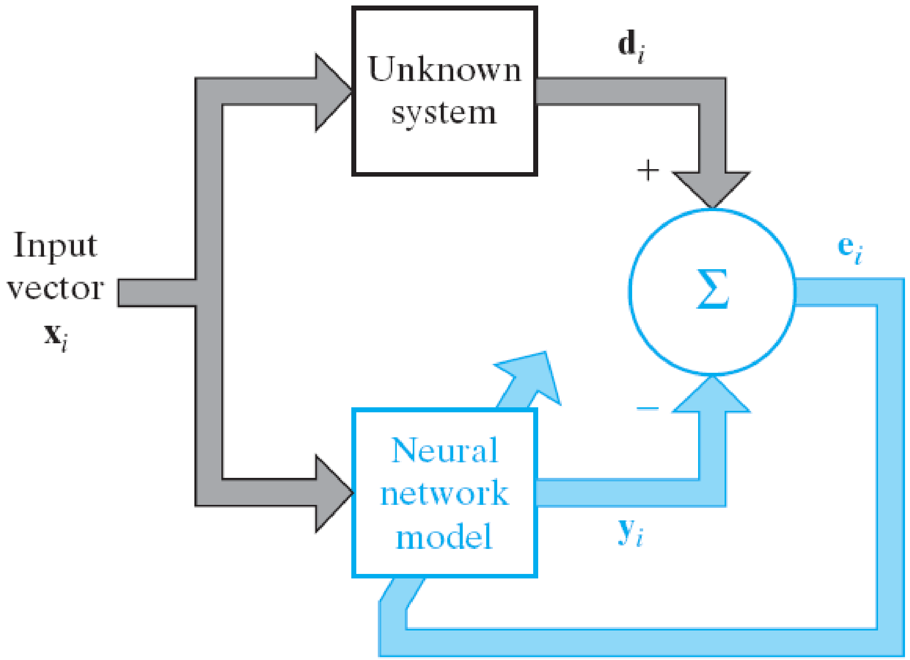
پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

مزایا و معایب دو روش:

– **آموزش برخط (یا الگویی):**

- **امکان خارج‌شدن از کمینه‌های محلی با اعمال نمونه‌ها به‌صورت اتفاقی**
- **قابلیت تطبیق با تغییر پارامترهای سیستم (فرآیند)**



پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

مزایا و معایب دو روش:

- آموزش برخط (یا الگویی):
  - امکان خارج‌شدن از کمینه‌های محلی با اعمال نمونه‌ها به‌صورت اتفاقی
  - قابلیت تطبیق با تغییر پارامترهای سیستم (فرآیند)

- آموزش برون‌خط (یا دسته‌ای)

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

مزایا و معایب دو روش:

- آموزش برخط (یا الگویی):
  - امکان خارج‌شدن از کمینه‌های محلی با اعمال نمونه‌ها به‌صورت اتفاقی
  - قابلیت تطبیق با تغییر پارامترهای سیستم (فرآیند)

- آموزش برون‌خط (یا دسته‌ای)
  - دقت بهتر در تخمین تغییرات وزن‌ها.
    یعنی، تخمین بهتر از مشتق تابع هزینه نسبت به وزن‌ها

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

مزایا و معایب دو روش:

- آموزش برخط (یا الگویی):
  - امکان خارج‌شدن از کمینه‌های محلی با اعمال نمونه‌ها به‌صورت اتفاقی
  - قابلیت تطبیق با تغییر پارامترهای سیستم (فرآیند)

- آموزش برون‌خط (یا دسته‌ای)
  - دقت بهتر در تخمین تغییرات وزن‌ها.
    یعنی، تخمین بهتر از مشتق تابع هزینه نسبت به وزن‌ها
  - در نتیجه، همگرایی سریعتر (همانند Steepest descent)

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

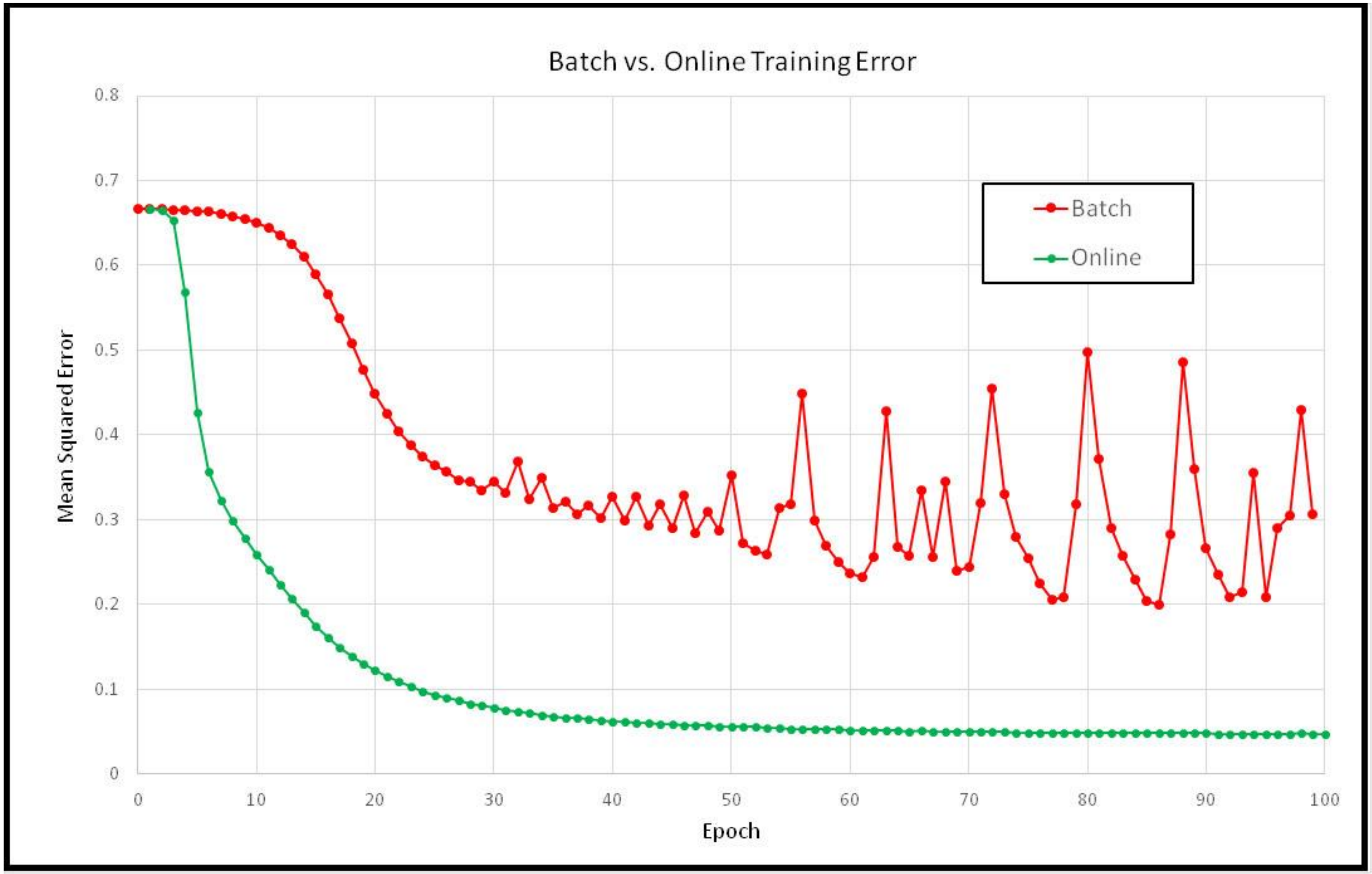
مزایا و معایب دو روش:

- **آموزش برخط (یا الگویی):**
  - **امکان خارج‌شدن از کمینه‌های محلی با اعمال نمونه‌ها به‌صورت اتفاقی**
  - **قابلیت تطبیق با تغییر پارامترهای سیستم (فرآیند)**

- **آموزش برون‌خط (یا دسته‌ای)**
  - **دقت بهتر در تخمین تغییرات وزن‌ها.**
    یعنی، تخمین بهتر از مشتق تابع هزینه نسبت به وزن‌ها
  - **در نتیجه، همگرایی سریعتر (همانند Steepest descent)**
  - **نیاز به حافظه زیادتر، مخصوصا برای داده‌های خیلی بزرگ**

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:



پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

انتخاب تابع غیرخطی سلول‌ها:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

انتخاب تابع غیرخطی سلول‌ها:

– نکته بسیار مهم: نیاز به محاسبه مشتق توابع

$$\delta_k(n) = \varphi_k'(v_k(n))e_k(n)$$

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_{k=1}^{m} w_{kj}(n)\delta_k(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

انتخاب تابع غیرخطی سلول‌ها:

– نکته بسیار مهم: نیاز به محاسبه مشتق توابع

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\delta_j(n) = \varphi'_j(v_j(n))\sum_{k=1}^{m} w_{kj}(n)\delta_k(n)$$

بنابراین:

آ) تابع باید مشتق‌پذیر باشد.

ب) محاسبه مشتق باید ساده باشد.

پ) خروجی تابع باید کراندار باشد.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

انتخاب تابع غیرخطی سلول‌ها:

– نکته بسیار مهم: نیاز به محاسبه مشتق توابع

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\delta_j(n) = \varphi'_j(v_j(n))\sum_{k=1}^{m} w_{kj}(n)\delta_k(n)$$

بنابراین:

آ) تابع باید مشتق‌پذیر باشد.

ب) محاسبه مشتق باید ساده باشد.

پ) خروجی تابع باید کراندار باشد.

تمامی توابع sشکل (Sigmoid) ویژگی‌های آ و پ را دارند.

ولی فقط توابع لجستیکی و تانژانت هیپربولیک ویژگی بسیار مهم ب را دارند.

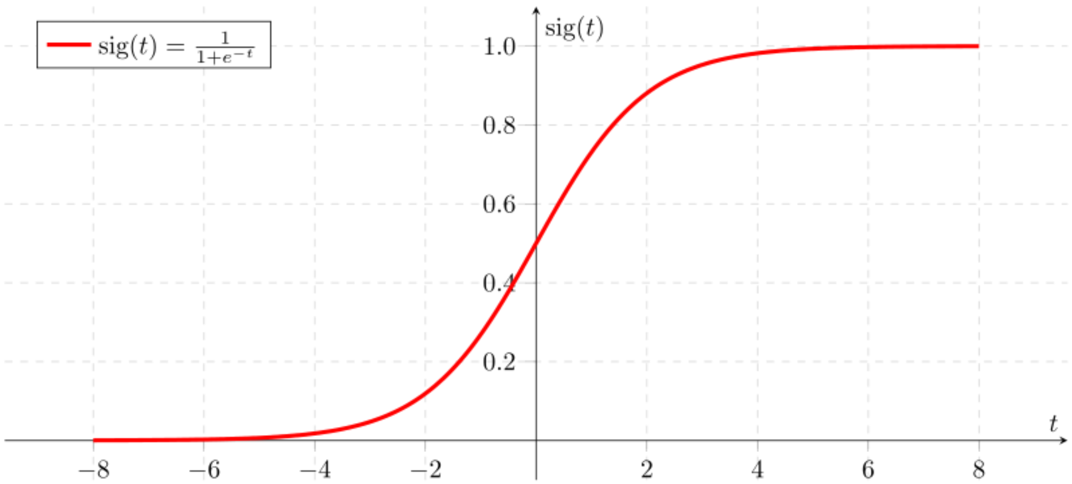
پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

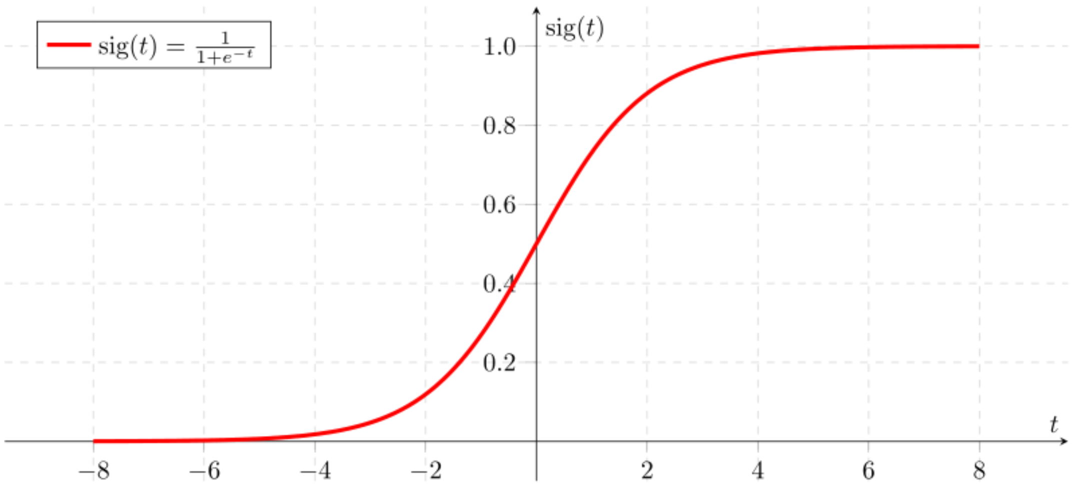
۱- تابع لجستیکی



پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی

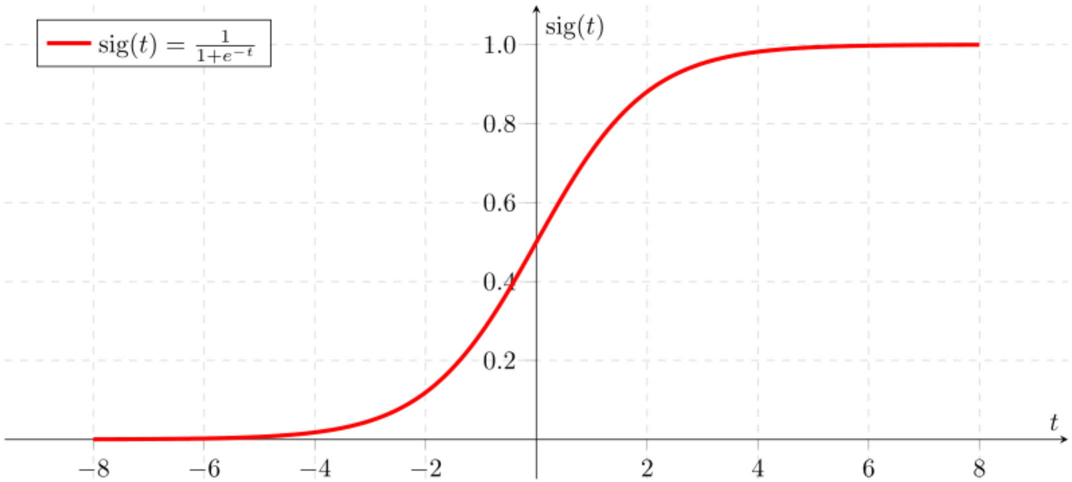


$$y_j(n) = \varphi_j(v_j(n)) = \frac{1}{1+e^{-(av_j(n))}}, \; -\infty < v_j(n) < \infty, \; 0 < y_j(n) < 1$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی



$$y_j(n) = \varphi_j(v_j(n)) = \frac{1}{1+e^{-(av_j(n))}}, \; -\infty < v_j(n) < \infty, \; 0 < y_j(n) < 1$$

$$\varphi_j'(v_j(n)) = ay_j(n)[1 - y_j(n)]$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی

محاسبه دلتاها:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi_k'(v_k(n))e_k(n)$$

$$\varphi_k'(v_k(n)) = a y_k(n)[1 - y_k(n)]$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\varphi'_k(v_k(n)) = ay_k(n)[1 - y_k(n)]$$

$$\delta_k(n) = ay_k(n)[1 - y_k(n)][d_k(n) - y_k(n)] \quad \forall k = 1,\ldots,m$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\varphi'_k(v_k(n)) = ay_k(n)[1 - y_k(n)]$$

$$\delta_k(n) = ay_k(n)[1 - y_k(n)][d_k(n) - y_k(n)] \quad \forall k = 1,\ldots,m$$

ب) سلول‌های لایه پنهان:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\varphi'_k(v_k(n)) = ay_k(n)[1 - y_k(n)]$$

$$\delta_k(n) = ay_k(n)[1 - y_k(n)][d_k(n) - y_k(n)] \;\; \forall k = 1,\ldots,m$$

ب) سلول‌های لایه پنهان:

$$\delta_j(n) = \varphi'_j(v_j(n))\sum_{k=1}^{m} w_{kj}(n)\delta_k(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۱- تابع لجستیکی

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\varphi'_k(v_k(n)) = ay_k(n)[1 - y_k(n)]$$

$$\delta_k(n) = ay_k(n)[1 - y_k(n)][d_k(n) - y_k(n)] \quad \forall k = 1,\ldots,m$$

ب) سلول‌های لایه پنهان:

$$\delta_j(n) = \varphi'_j(v_j(n))\sum_{k=1}^{m} w_{kj}(n)\delta_k(n)$$

$$\delta_j(n) = ay_j(n)[1 - y_j(n)]\sum_{k=1}^{m} w_{kj}(n)\delta_k(n) \quad \forall j = 0,\ldots,q$$

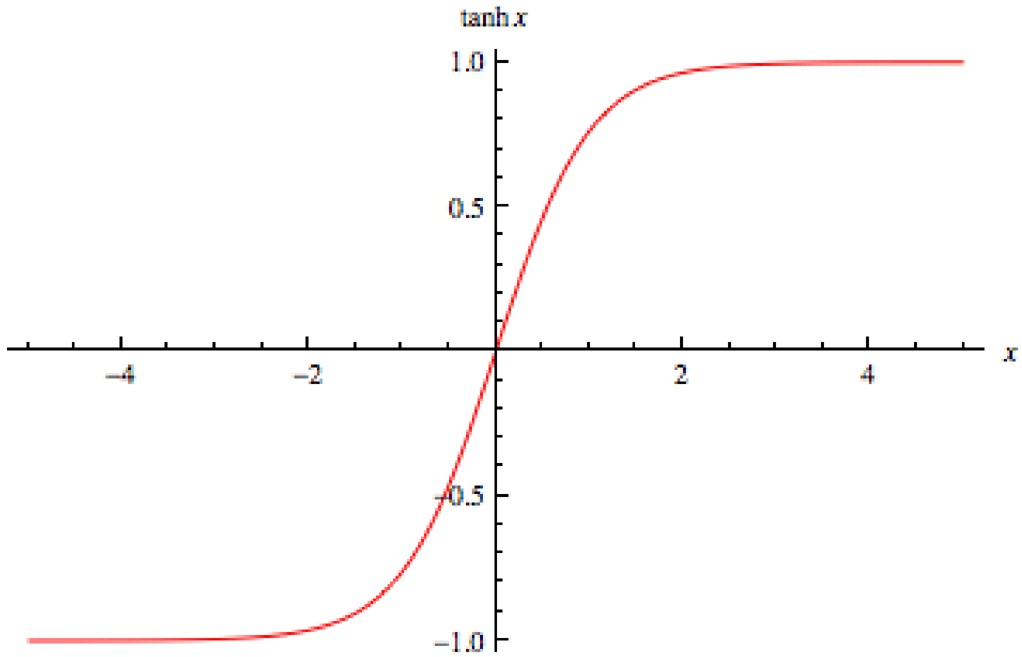
پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

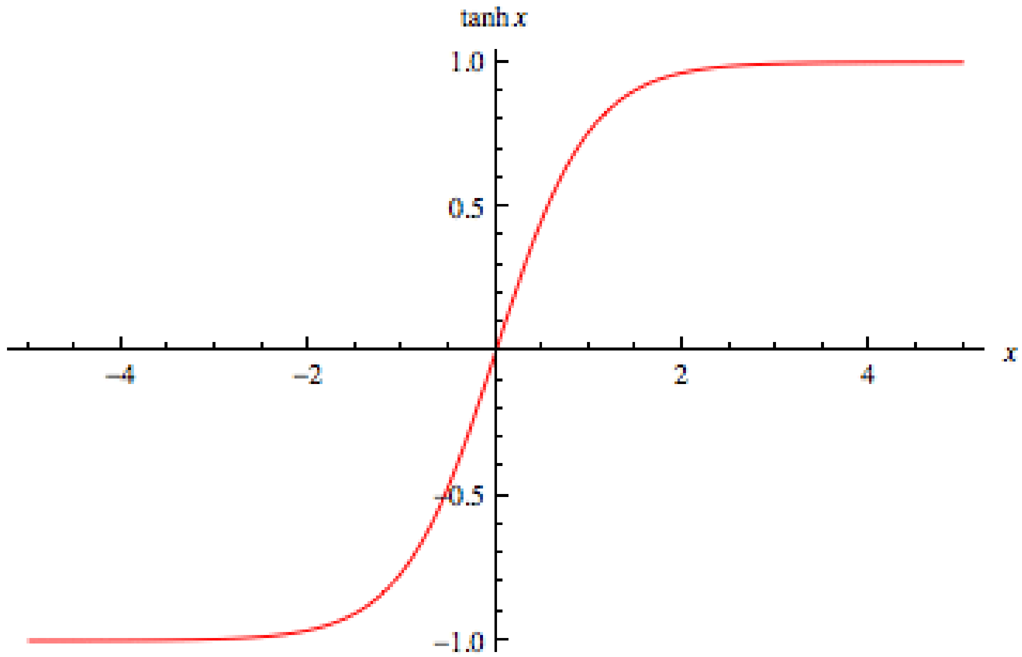
۲- تابع تانژانت هیپربولیک



پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک

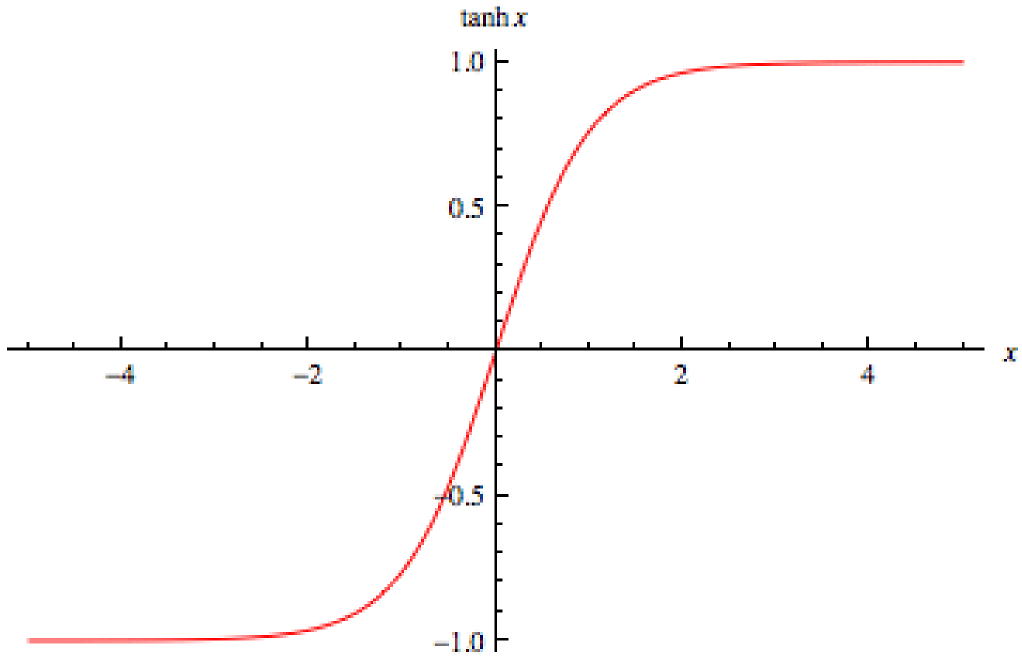


$$y_j(n) = \varphi_j(v_j(n)) = \frac{1 - e^{-(av_j(n))}}{1 + e^{-(av_j(n))}}, \ -\infty < v_j(n) < \infty, \ -1 < y_j(n) < 1$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک



$$y_j(n) = \varphi_j(v_j(n)) = \frac{1-e^{-(av_j(n))}}{1+e^{-(av_j(n))}}, \; -\infty < v_j(n) < \infty, \; -1 < y_j(n) < 1$$

$$\varphi_j'(v_j(n)) = a[1-y_j^2(n)]$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک

محاسبه دلتاها:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\varphi'_k(v_k(n)) = a[1 - y_k^2(n)]$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\varphi'_k(v_k(n)) = a[1 - y_k^2(n)]$$

$$\delta_k(n) = a[1 - y_k^2(n)][d_k(n) - y_k(n)] \;\; \forall k = 1, \ldots, m$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi_k'(v_k(n))e_k(n)$$

$$\varphi_k'(v_k(n)) = a[1 - y_k^2(n)]$$

$$\delta_k(n) = a[1 - y_k^2(n)][d_k(n) - y_k(n)] \quad \forall k = 1, \ldots, m$$

ب) سلول‌های لایه پنهان:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi'_k(v_k(n))e_k(n)$$

$$\varphi'_k(v_k(n)) = a[1 - y_k^2(n)]$$

$$\delta_k(n) = a[1 - y_k^2(n)][d_k(n) - y_k(n)] \quad \forall k = 1, \ldots, m$$

ب) سلول‌های لایه پنهان:

$$\delta_j(n) = \varphi'_j(v_j(n)) \sum_{k=1}^{m} w_{kj}(n)\delta_k(n)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

۲- تابع تانژانت هیپربولیک

محاسبه دلتاها:

آ) سلول‌های لایه خروجی:

$$\delta_k(n) = \varphi_k'(v_k(n))e_k(n)$$

$$\varphi_k'(v_k(n)) = a[1 - y_k^2(n)]$$

$$\delta_k(n) = a[1 - y_k^2(n)][d_k(n) - y_k(n)] \quad \forall k = 1, \ldots, m$$

ب) سلول‌های لایه پنهان:

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_{k=1}^{m} w_{kj}(n)\delta_k(n)$$

$$\delta_j(n) = a[1 - y_j^2(n)]\sum_{k=1}^{m} w_{kj}(n)\delta_k(n) \quad \forall j = 0, \ldots, q$$

پرسپترون چندلایه (MLP)

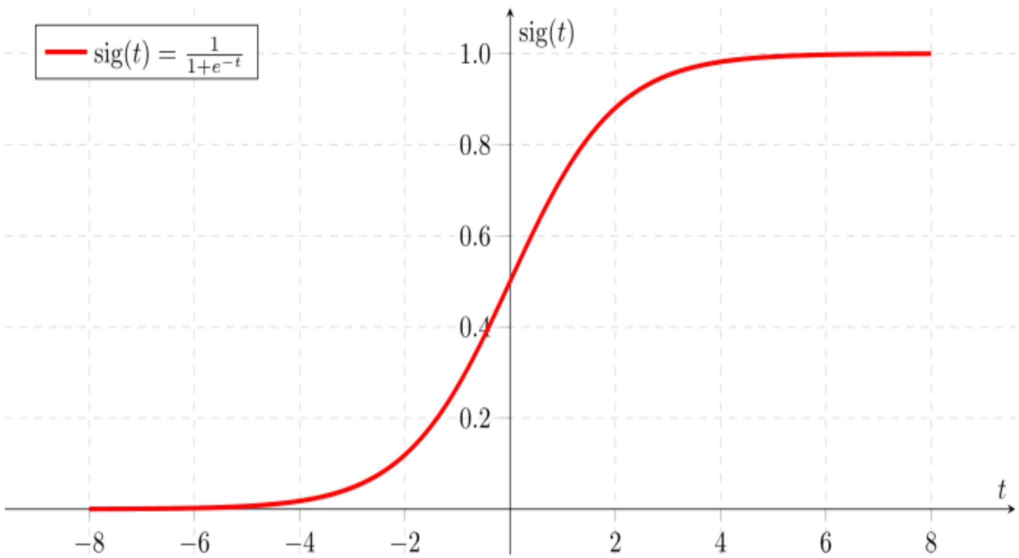
الگوریتم پس‌انتشار خطا:

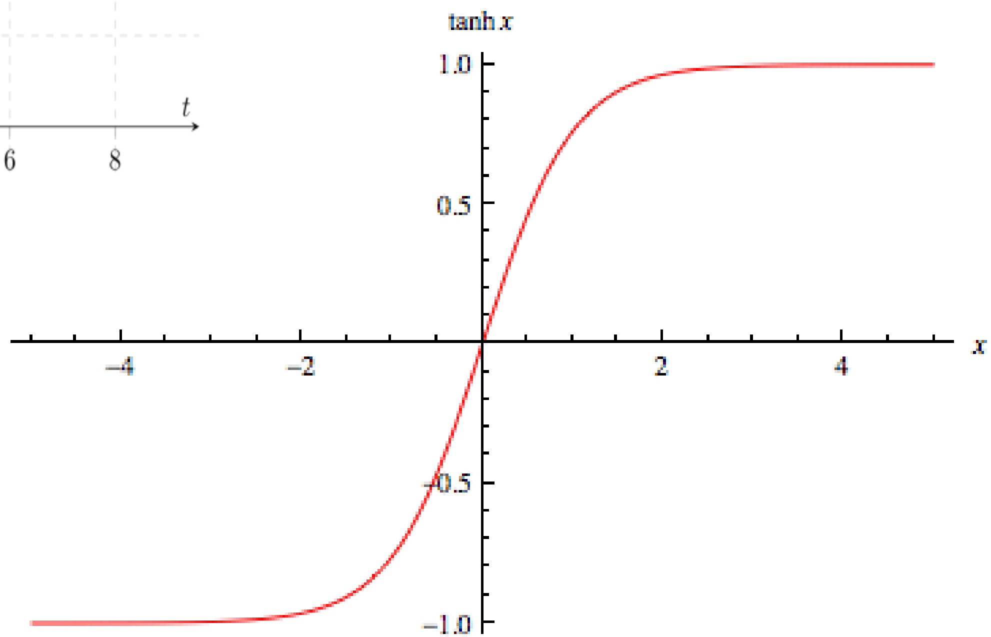
مقایسه دو تابع:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

مقایسه دو تابع:





پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

<u>ضریب آموزش</u>

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

<u>ضریب آموزش</u>

۱- ضریب آموزش کوچک:
- تضمین همگرای الگوریتم
- تغییرات اندک در وزن‌ها، یعنی تکرارهای خیلی زیاد

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

<u>ضریب آموزش</u>

۱- ضریب آموزش کوچک:
- تضمین همگرای الگوریتم
- تغییرات اندک در وزن‌ها، یعنی تکرارهای خیلی زیاد

۲- ضریب آموزش بزرگ:
- تغییرات زیاد در وزن‌ها، یعنی تکرارهای کم
- امکان واگرای الگوریتم

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

۱- ضریب آموزش کوچک:
- تضمین همگرای الگوریتم
- تغییرات اندک در وزن‌ها، یعنی تکرارهای خیلی زیاد

۲- ضریب آموزش بزرگ:
- تغییرات زیاد در وزن‌ها، یعنی تکرارهای کم
- امکان واگرای الگوریتم

یکی از راه‌های افزایش سرعت همگرایی و در عین حال تضمین همگرایی الگوریتم، اضافه‌کردن ضریب ممنتم است.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

۱- ضریب آموزش کوچک:
- تضمین همگرای الگوریتم
- تغییرات اندک در وزن‌ها، یعنی تکرارهای خیلی زیاد

۲- ضریب آموزش بزرگ:
- تغییرات زیاد در وزن‌ها، یعنی تکرارهای کم
- امکان واگرای الگوریتم

یکی از راه‌های افزایش سرعت همگرایی و در عین حال تضمین همگرایی الگوریتم، اضافه‌کردن ضریب ممنتم است.

$$\Delta w_{ji}(n) = \eta\, \delta_j(n) y_i(n) + \alpha\, \Delta w_{ji}(n-1)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

۱- ضریب آموزش کوچک:
- تضمین همگرای الگوریتم
- تغییرات اندک در وزن‌ها، یعنی تکرارهای خیلی زیاد

۲- ضریب آموزش بزرگ:
- تغییرات زیاد در وزن‌ها، یعنی تکرارهای کم
- امکان واگرای الگوریتم

یکی از راه‌های افزایش سرعت همگرایی و در عین حال تضمین همگرایی الگوریتم، اضافه‌کردن ضریب ممنتم است.

$$\Delta w_{ji}(n) = \eta\, \delta_j(n) y_i(n) + \alpha\, \Delta w_{ji}(n-1)$$

از قبل داشتیم

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

۱- ضریب آموزش کوچک:
- تضمین همگرای الگوریتم
- تغییرات اندک در وزن‌ها، یعنی تکرارهای خیلی زیاد

۲- ضریب آموزش بزرگ:
- تغییرات زیاد در وزن‌ها، یعنی تکرارهای کم
- امکان واگرای الگوریتم

یکی از راه‌های افزایش سرعت همگرایی و در عین حال تضمین همگرایی الگوریتم، اضافه‌کردن ضریب ممنتم است.

$$\Delta w_{ji}(n) = \eta\, \underbrace{\delta_j(n) y_i(n)}_{\text{از قبل داشتیم}} + \underbrace{\alpha\, \Delta w_{ji}(n-1)}_{\text{جمله ممنتم}}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

<u>ضریب آموزش</u>

۱- ضریب آموزش کوچک:
- تضمین همگرای الگوریتم
- تغییرات اندک در وزن‌ها، یعنی تکرارهای خیلی زیاد

۲- ضریب آموزش بزرگ:
- تغییرات زیاد در وزن‌ها، یعنی تکرارهای کم
- امکان واگرای الگوریتم

یکی از راه‌های افزایش سرعت همگرایی و در عین حال تضمین همگرایی الگوریتم، اضافه‌کردن ضریب ممنتم است.

قاعده عمومی دلتا

$$\Delta w_{ji}(n) = \eta\, \underbrace{\delta_j(n) y_i(n)}_{\text{از قبل داشتیم}} + \underbrace{\alpha\, \Delta w_{ji}(n-1)}_{\text{جمله ممنتم}}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

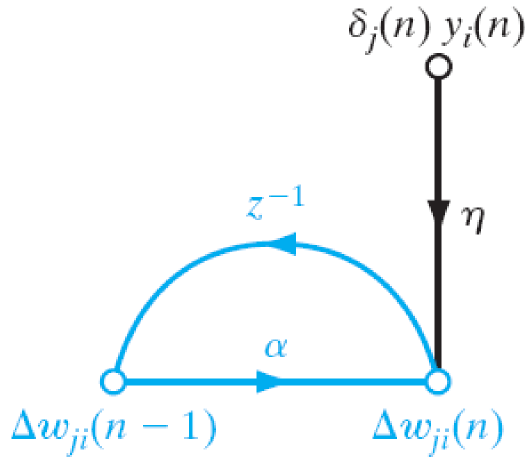
$$\Delta w_{ji}(n) = \eta\, \delta_j(n) y_i(n) + \alpha\, \Delta w_{ji}(n-1)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = \eta\, \delta_j(n) y_i(n) + \alpha\, \Delta w_{ji}(n-1)$$



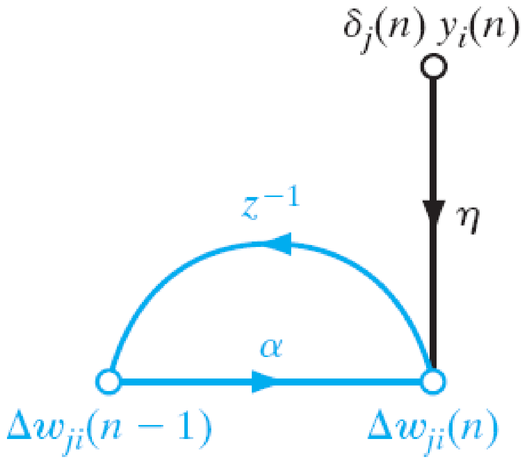
نمودار گذر سیگنال این معادله

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = \eta\, \delta_j(n) y_i(n) + \alpha\, \Delta w_{ji}(n-1)$$



نمودار گذر سیگنال این معادله

با حل معادله قاعده عمومی دلتا، می‌توان آن را به‌صورت سری زمانی نیز نشان‌داد:

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t}\, \delta_j(t) y_i(t)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

<u>ضریب آموزش</u>

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t)$$

قبلا داشتیم:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\delta_j(t)\, y_i(t)$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t)$$

قبلا داشتیم:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -\delta_j(t)\, y_i(t)$$

بنابراین

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

نتیجه گیری در مورد این سری زمانی:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

نتیجه گیری در مورد این سری زمانی:

۱- برای همگرایی، باید $0 \leq |\alpha| < 1$. مقدار ضریب ممنتم مثبت درنظرگرفته می‌شود.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

نتیجه گیری در مورد این سری زمانی:

۱- برای همگرایی، باید $0 \leq |\alpha| < 1$. مقدار ضریب ممنتم مثبت درنظرگرفته می‌شود.

۲- توجه به علامت مشتق پاره‌ای در مراحل تکرار پیاپی:

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

نتیجه گیری در مورد این سری زمانی:

۱- برای همگرایی، باید $0 \leq |\alpha| < 1$. مقدار ضریب ممنتم مثبت درنظرگرفته می‌شود.

۲- توجه به علامت مشتق پاره‌ای در مراحل تکرار پیاپی:

آ- همواره یکسان: مقدار تغییرات در وزن در تمامی مراحل (از صفر تا n) با هم جمع‌شده و در نتیجه به همگرایی الگوریتم تسریع می‌بخشد.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

نتیجه گیری در مورد این سری زمانی:

۱- برای همگرایی، باید $0 \leq |\alpha| < 1$. مقدار ضریب ممنتم مثبت درنظرگرفته می‌شود.

۲- توجه به علامت مشتق پاره‌ای در مراحل تکرار پیاپی:

آ- همواره یکسان: مقدار تغییرات در وزن در تمامی مراحل (از صفر تا n) با هم جمع‌شده و در نتیجه به همگرایی الگوریتم تسریع می‌بخشد.

ب- متفاوت: یعنی این که وزن‌ها حول نقطه بهینه حالت نوسانی دارد. یعنی امکان واگرایی الگوریتم.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

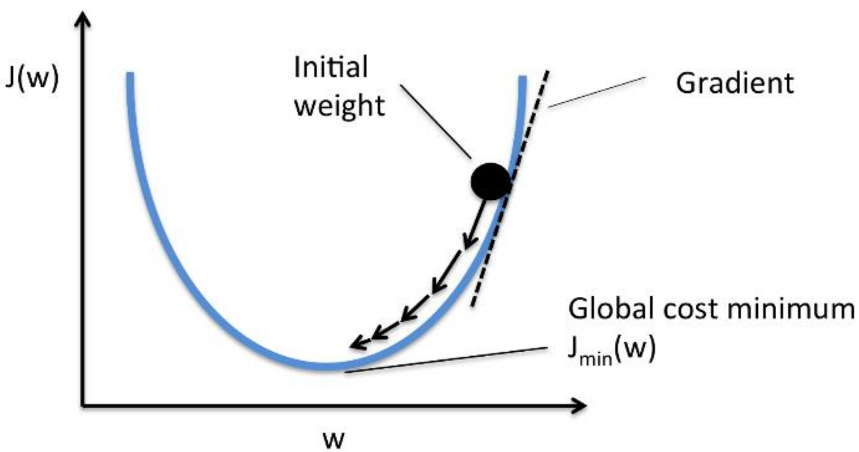
نتیجه گیری در مورد این سری زمانی:

۱- برای همگرایی، باید $0 \le |\alpha| < 1$. مقدار ضریب ممنتم مثبت درنظرگرفته می‌شود.

۲- توجه به علامت مشتق پاره‌ای در

آ- همواره یکسان: مقدار تغییرا
جمع‌شده و در نتیجه به همگر

ب- متفاوت: یعنی این که وزن‌ها
واگرایی الگوریتم.



پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

<u>ضریب آموزش</u>

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)}$$

نتیجه گیری در مورد این سری زمانی:

۱- برای همگرایی، باید $0 \le |\alpha| < 1$. مقدار ضریب ممنتم مثبت درنظرگرفته می‌شود.

۲- توجه به علامت مشتق پاره‌ای در مراحل تکرار پیاپی:

آ- همواره یکسان: مقدار تغییرات در وزن در تمامی مراحل (از صفر تا n) با هم جمع‌شده و در نتیجه به همگرایی الگوریتم تسریع می‌بخشد.

ب- متفاوت: یعنی این که وزن‌ها حول نقطه بهینه حالت نوسانی دارد. یعنی امکان واگرایی الگوریتم.

در این حالت، اضافه‌کردن ضریب ممنتم باعث تغییر علامت در جملات و در نتیجه کاهش تغییرات در وزن می‌شود. اثر پایداری برروی الگوریتم پس‌انتشار خطا خواهدداشت.

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

<u>ضریب آموزش</u>

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

<u>ضریب آموزش</u>

انتخاب مقادیر مناسب

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

انتخاب مقادیر مناسب

$$1)\ \begin{cases} \alpha \to 1 \\ \eta \to 0 \end{cases}$$

پرسپترون چندلایه (MLP)

الگوریتم پس‌انتشار خطا:

ضریب آموزش

انتخاب مقادیر مناسب

$$1)\ \begin{cases} \alpha \to 1 \\ \eta \to 0 \end{cases}$$

$$2)\ \begin{cases} \alpha \to 0 \\ \eta \to 1 \end{cases}$$