آموزش شبکه‌های عصبی مصنوعی

سید ناصر رضوی www.snrazavi.ir

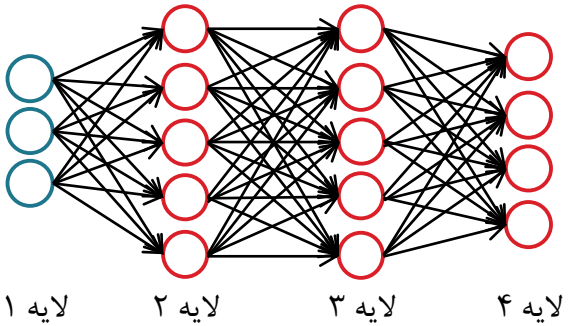
۱۳۹۶

شبکه‌های عصبی (کلاس‌بندی)

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$$

تعداد کل لایه‌ها در شبکه عصبی $= L$

تعداد واحدها (بدون در نظر گرفتن بایاس) در لایه l $= s_l$



کلاس‌بندی دودویی

$y \in \{0,1\}$

۱ واحد خروجی

کلاس‌بندی چندکلاسی (k کلاس)

$y \in \mathbb{R}^K$

$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$ $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$ $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

پیاده رو موتورسیکلت خودرو کامیون

k واحد خروجی

تابع هزینه

- رگرسیون لجستیک.

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

- شبکه عصبی.

$h_\Theta(x) \in \mathbb{R}^K \qquad \left(h_\Theta(x)\right)_i = i^{th}\ output$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right) \log\left(1 - \left(h_\theta\left(x^{(i)}\right)\right)_k\right)\right]$$

$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} \left(\Theta_{ji}^{(l)}\right)^2$$

الگوریتم پس‌انتشار خطا

محاسبه گرادیان

- تابع هزینه.

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right)\log\left(1 - \left(h_\theta\left(x^{(i)}\right)\right)_k\right)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\Theta_{ji}^{(l)}\right)^2$$

- هدف.

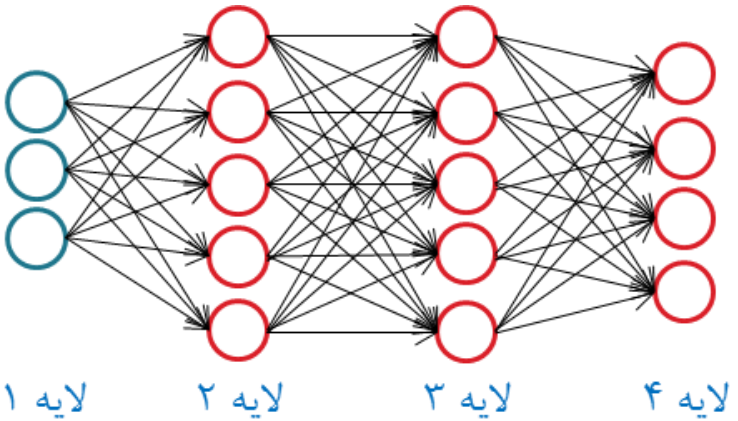
$$\min_{\Theta} J(\Theta)$$

- کمیت‌هایی که باید محاسبه شوند:

$$J(\Theta) \qquad \frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$$

محاسبه گرادیان

با داشتن یک نمونه آموزشی به صورت (x,y)

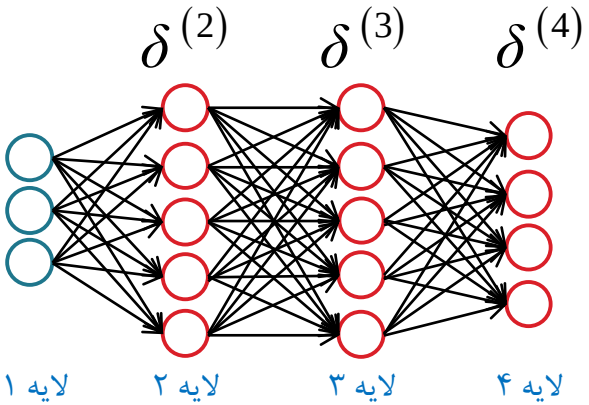


انتشار پیش رو

$$a^{(1)} = x$$
$$z^{(2)} = \Theta^{(1)} a^{(1)}$$
$$a^{(2)} = g\left(z^{(2)}\right) \quad \left(\text{add } a_0^{(2)}\right)$$
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$a^{(3)} = g\left(z^{(3)}\right) \quad \left(\text{add } a_0^{(3)}\right)$$
$$z^{(4)} = \Theta^{(3)} a^{(3)}$$
$$a^{(4)} = h_\Theta(x) = g\left(z^{(4)}\right)$$

محاسبه گرادیان: الگوریتم پس‌انتشار خطا

تعریف. $\delta_j^{(l)}$ = خطای گره j در لایه l



محاسبه خطا برای واحدهای خروجی $(l = 4)$

$$\delta^{(4)} = \left(y - a^{(4)}\right) .\times\ g'\left(z^{(4)}\right)$$

محاسبه خطا برای واحدهای مخفی $(l = 2,3)$

$$\delta^{(3)} = \left(\Theta^{(3)}\right)^T \delta^{(4)} .\times\ g'\left(z^{(3)}\right)$$

$$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .\times\ g'\left(z^{(2)}\right)$$

$$g'\left(z^{(3)}\right) = a^{(3)} .\times \left(1 - a^{(3)}\right)$$

$$g'\left(z^{(2)}\right) = a^{(2)} .\times \left(1 - a^{(2)}\right)$$

الگوریتم پس‌انتشار خطا

مجموعه آموزشی: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ (for all $l, i. j$).

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2,3, \dots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

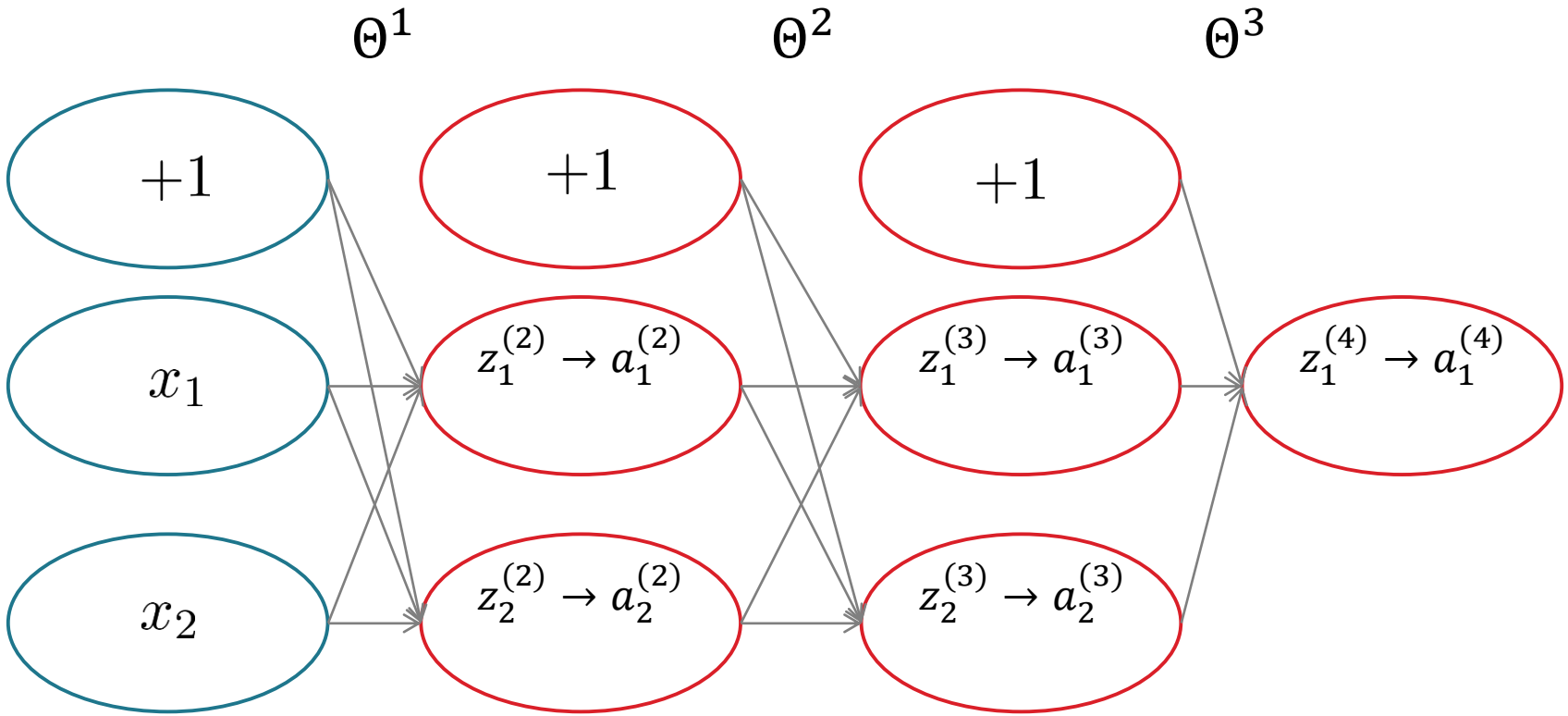
$D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)}$ if $j \neq 0$

$D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)}$ if $j = 0$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

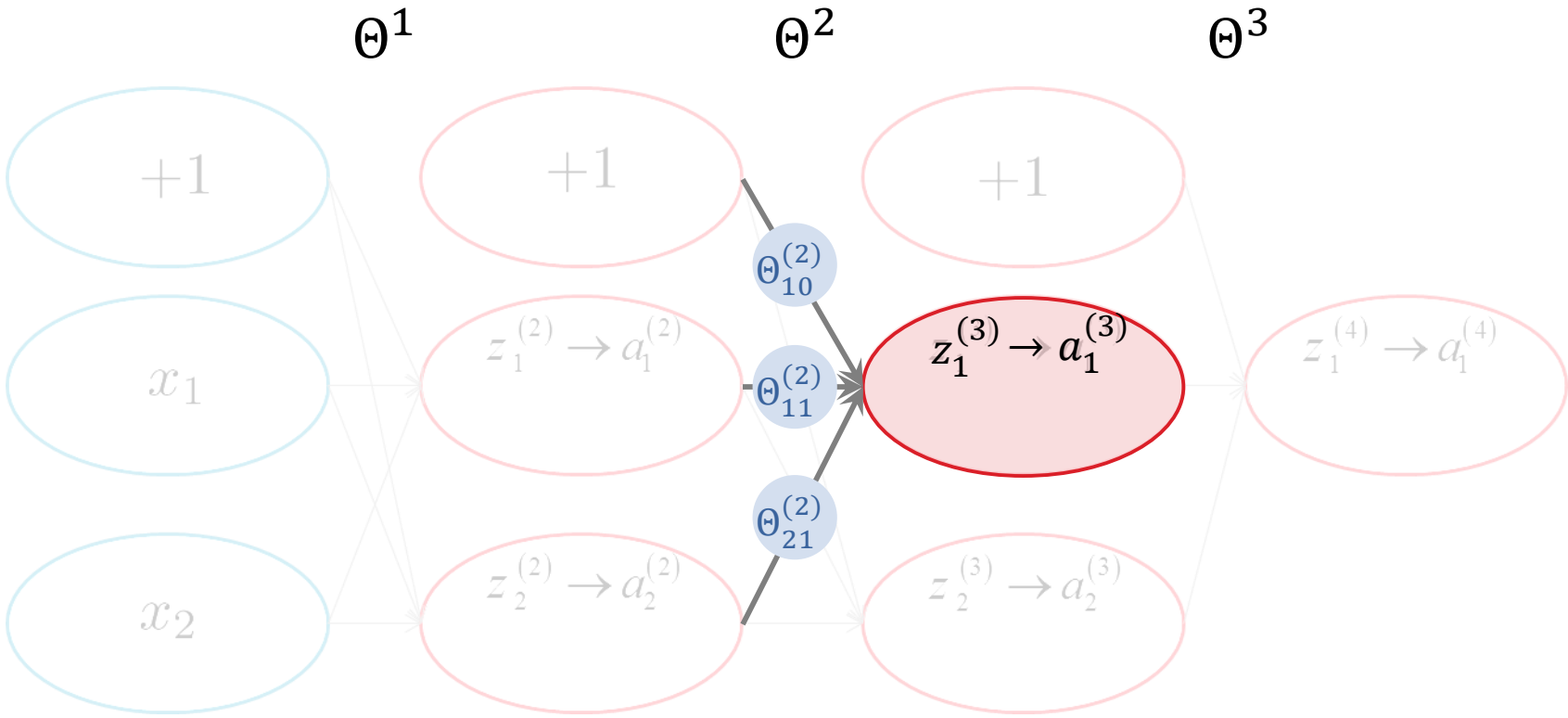
الگوریتم پس انتشار خطا: معنای شهودی

انتشار رو به جلو



$$\left(x^{(i)}, y^{(i)}\right)$$

انتشار رو به جلو



محاسبه مجموع وزن‌دار ورودی‌ها

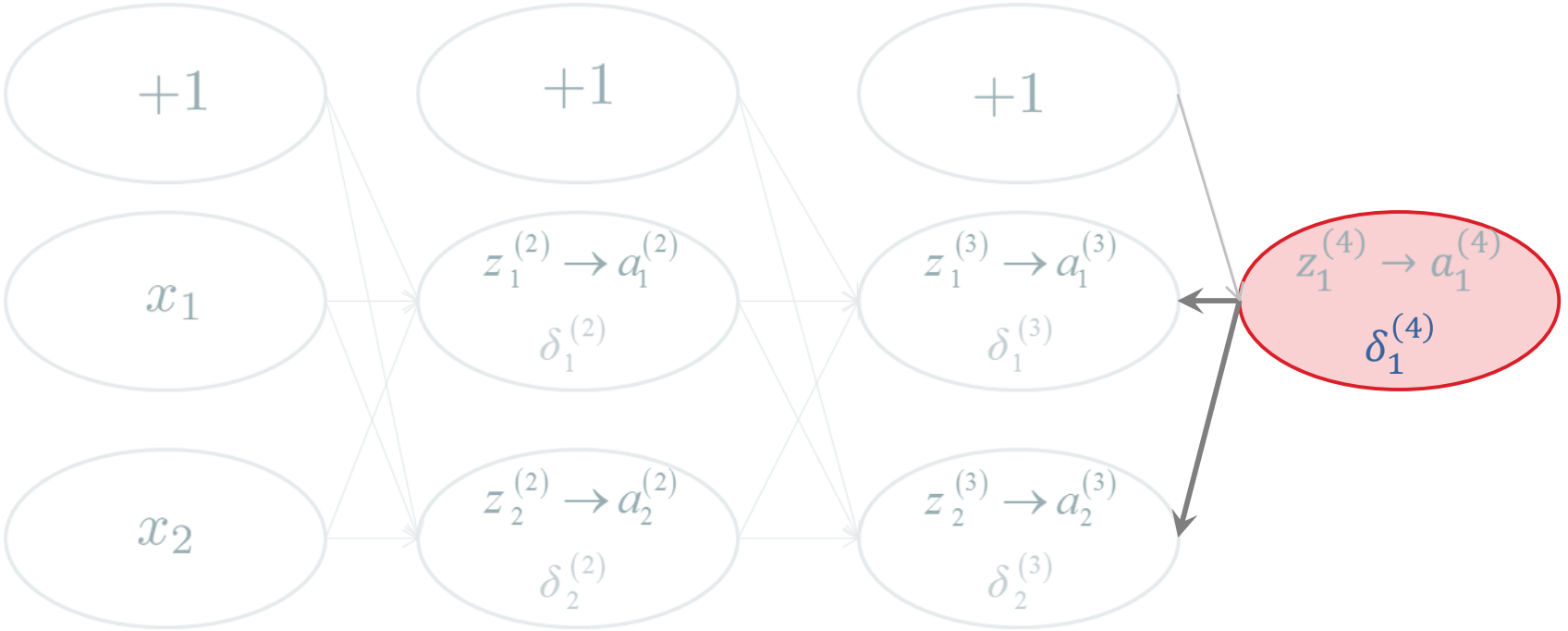
$$z_1^{(3)} = \Theta_{10}^{(2)} \times a_0^{(2)} + \Theta_{11}^{(2)} \times a_1^{(2)} + \Theta_{12}^{(2)} \times a_2^{(2)}$$

اعمال یک تابع غیرخطی

$$a_1^{(3)} = g\left(z_1^{(3)}\right)$$

پس‌انتشار خطا

$$(x^{(i)}, y^{(i)})$$

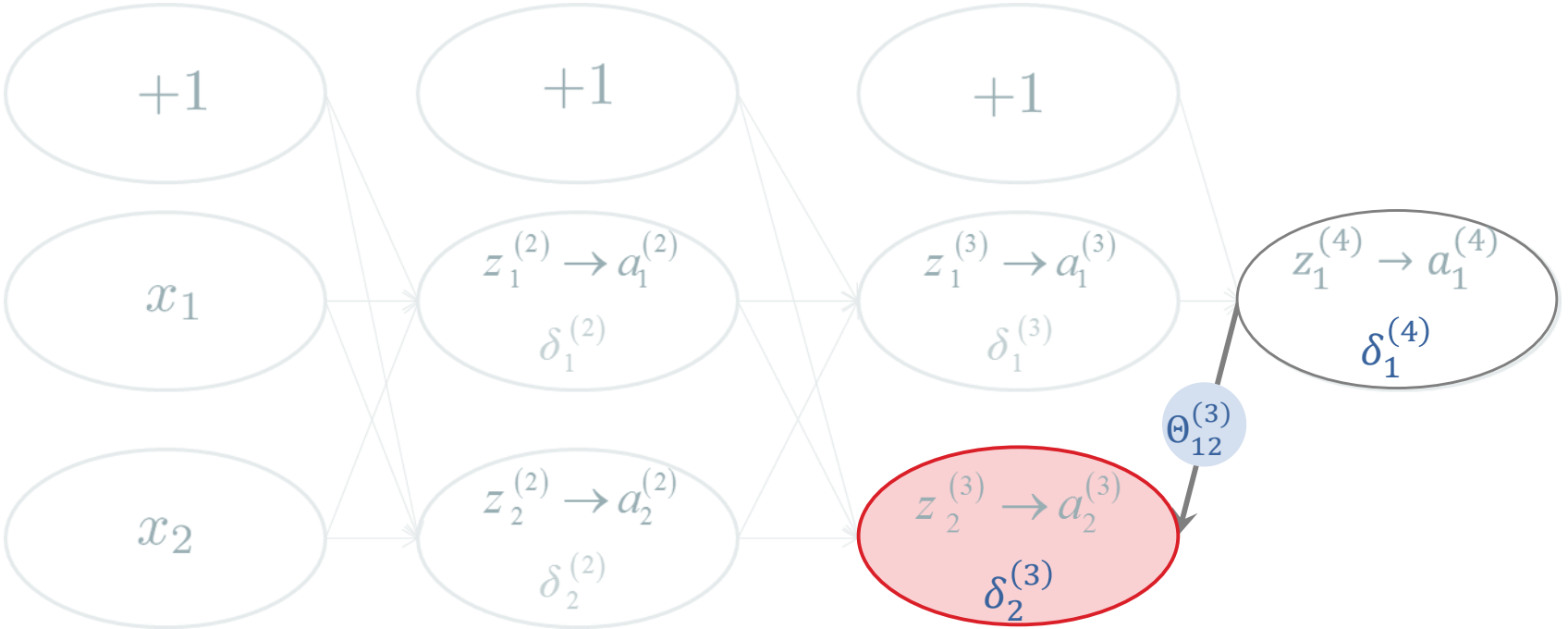


سیگنال خطا در نورون خروجی

$$\delta_1^{(4)} = \left(y^{(i)} - a_1^{(4)}\right) \times g'(z_1^4)$$

پس‌انتشار خطا

$$(x^{(i)}, y^{(i)})$$

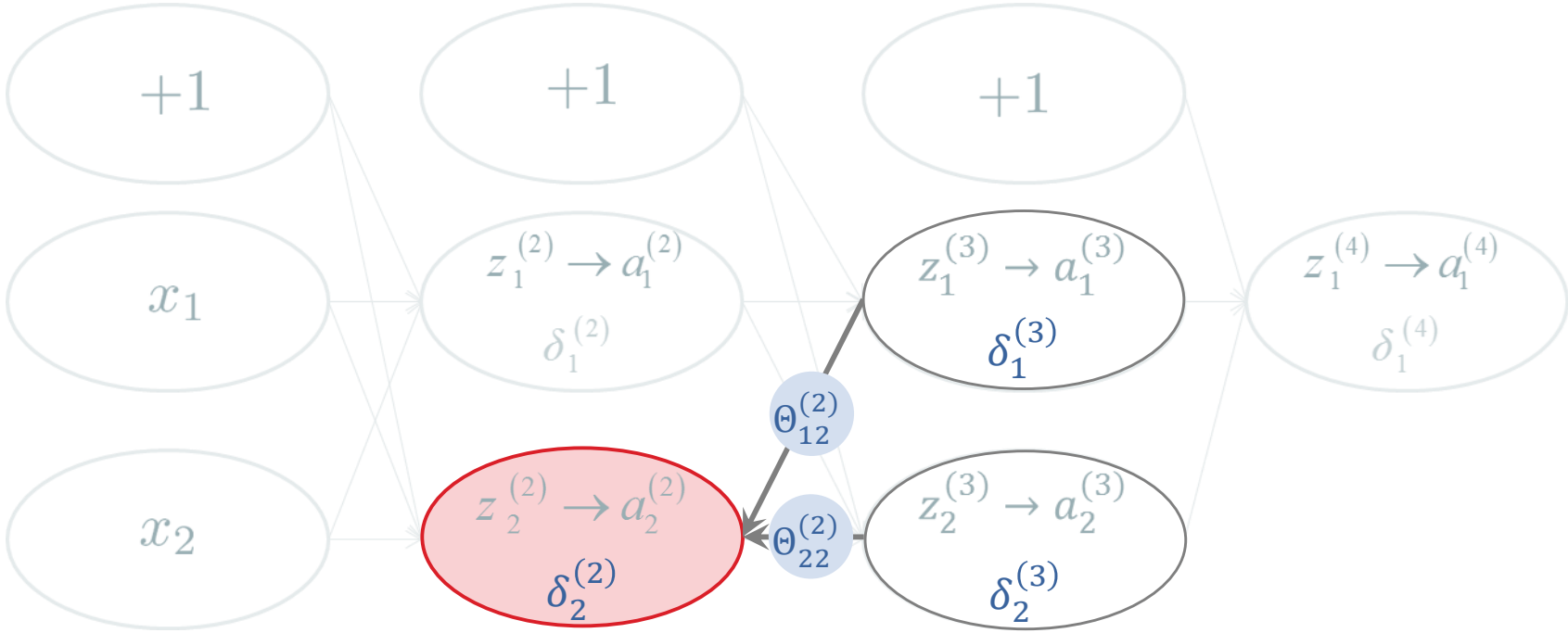


$$\delta^{(3)} = \left(\left(\Theta^{(3)}\right)^T * \delta^{(4)}\right).\times\, g'\left(z^{(3)}\right)$$

$$\delta_2^{(3)} = \Theta_{12}^{(3)} \times \delta_1^{(4)} \times g'\left(z_2^{(3)}\right)$$

پس‌انتشار خطا

$$(x^{(i)}, y^{(i)})$$



$$\delta^{(2)} = \left(\left(\Theta^{(2)}\right)^T * \delta^{(3)}\right) .\times\, g'\left(z^{(2)}\right)$$

$$\delta_2^{(2)} = \left(\Theta_{12}^{(2)} \times \delta_1^{(3)} + \Theta_{12}^{(2)} \times \delta_2^{(3)}\right) \times g'\left(z_2^{(2)}\right)$$

نکاتی در مورد پیاده‌سازی BP

بهینه‌سازی پیشرفته


```
function [jVal, gradient] = costFunction(theta)
                            ∈ ℝ^(n+1)        ∈ ℝ^(n+1)
...

optTheta = fminunc(@costFunction, initialTheta, options);
```

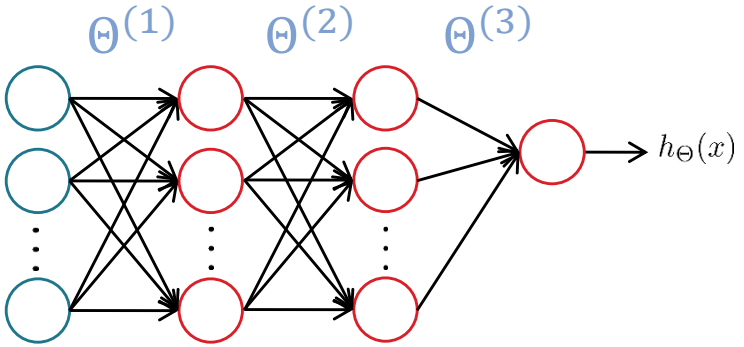
- پارامترها در یک شبکه عصبی ۴ لایه [۲ لایه مخفی]:
  - ماتریس‌های وزن (`Theta1, Theta2, Theta3`)
  - ماتریس‌های تغییر وزن (`D1, D2, D3`)
- برای استفاده از روش‌های بهینه‌سازی پیشرفته، باید هر سه ماتریس به یک بردار تبدیل شوند.

مثال

$s_1 = 10 \quad s_2 = 10 \quad s_3 = 10 \quad s_4 = 1$

$\Theta^{(1)} \in \mathbb{R}^{10\times 11} \quad \Theta^{(2)} \in \mathbb{R}^{10\times 11} \quad \Theta^{(3)} \in \mathbb{R}^{1\times 11}$

$D^{(1)} \in \mathbb{R}^{10\times 11} \quad D^{(2)} \in \mathbb{R}^{10\times 11} \quad D^{(3)} \in \mathbb{R}^{1\times 11}$




```
thetaVec = [ Theta1(:); Theta2(:); Theta3(:) ];
DVec     = [ D1(:)    ; D2(:)    ; D3(:)     ];
```

تبدیل ماتریس‌ها به بردار

```
Theta1 = reshape(thetaVec(  1:110), 10, 11);
Theta2 = reshape(thetaVec(111:220), 10, 11);
Theta3 = reshape(thetaVec(221:231),  1, 11);
```

تبدیل بردارها به ماتریس

الگوریتم یادگیری

- با داشتن پارامترهای اولیه $\Theta^{(1)}$، $\Theta^{(2)}$، $\Theta^{(3)}$
  - این ماتریس‌ها را به بردار **initialTheta** تبدیل کنید تا بتوانید آن را به عنوان آرگومان به تابع زیر ارسال کنید:

```
fminunc(@costFunction, initialTheta, options)
```

- سپس، تابع هزینه را به صورت زیر بنویسید:

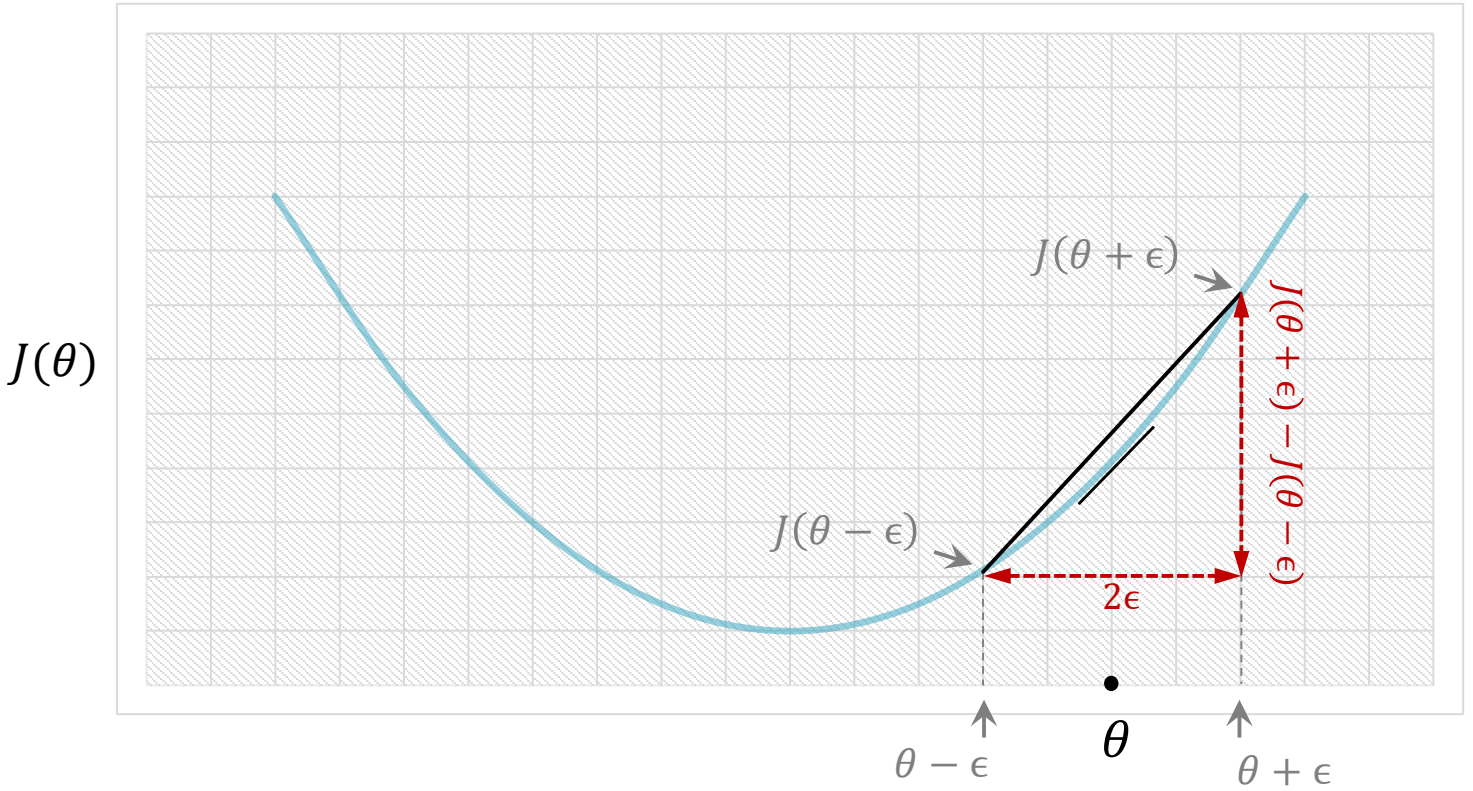
```
function [jVal, gradientVec] = costFunction(thetaVec)
```

  - از روی بردار **thetaVec** ماتریس‌های $\Theta^{(1)}$، $\Theta^{(2)}$، $\Theta^{(3)}$ را بازیابی کنید.
  - با استفاده از انتشار رو به جلو $J(\Theta)$ و با پس انتشار خطا ماتریس‌های $D^{(1)}$، $D^{(2)}$، $D^{(3)}$ را محاسبه کنید.
  - ماتریس‌های $D^{(1)}$، $D^{(2)}$ و $D^{(3)}$ را به بردار **gradientVec** تبدیل کنید.

بررسی گرادیان

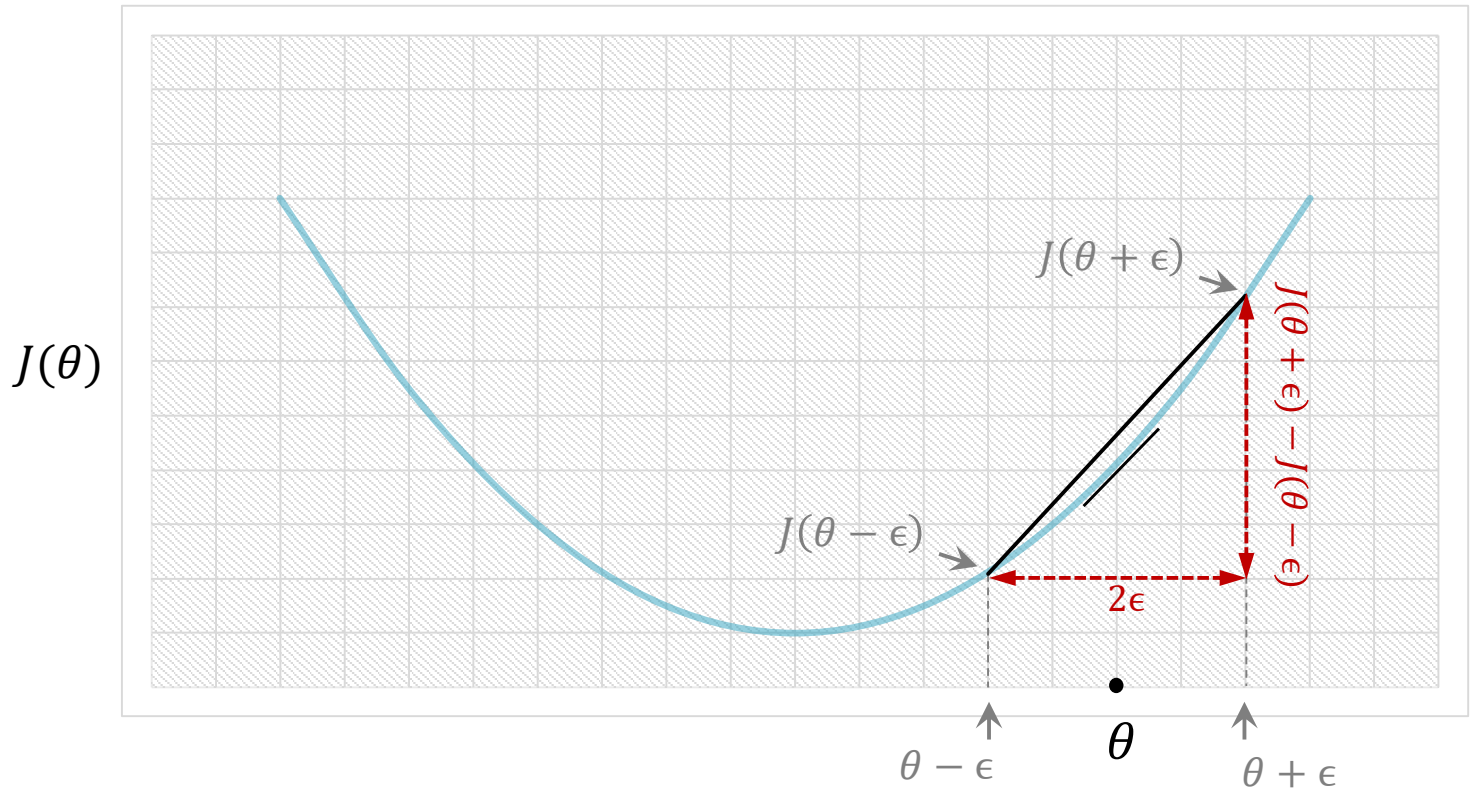
چگونه می‌توان مطمئن شد الگوریتم پس‌انتشار خطا را به درستی پیاده‌سازی نموده‌ایم؟

تخمین عددی گرادیان‌ها (تابع هزینه تک متغیره)



$$\frac{d}{d\theta}J(\theta)=\frac{J(\theta+\epsilon)-J(\theta-\epsilon)}{2\epsilon}$$

تخمین عددی گرادیان‌ها (تابع هزینه تک متغیره)




```
gradApprox = (J(theta + EPSILON) - J(theta - EPSLION)) / (2 * EPSILON);
```

تخمین گرادیان‌ها به صورت عددی (تابع هزینه چند متغیره)

$\theta \in \mathbb{R}^n \qquad \theta = \theta_1, \theta_2, \theta_3, \dots, \theta_n \qquad (\theta \leftarrow \Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)})$

$$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta_1 + \epsilon, \theta_2, \theta_3, \dots, \theta_n) - J(\theta_1 - \epsilon, \theta_2, \theta_3, \dots, \theta_n)}{2\epsilon}$$

$$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta_1, \theta_2 + \epsilon, \theta_3, \dots, \theta_n) - J(\theta_1, \theta_2 - \epsilon, \theta_3, \dots, \theta_n)}{2\epsilon}$$

$\vdots$

$$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta_1, \theta_2, \theta_3, \dots, \theta_n + \epsilon) - J(\theta_1, \theta_2, \theta_3, \dots, \theta_n - \epsilon)}{2\epsilon}$$

پیاده‌سازی: محاسبه عددی گرادیان


```
for i = 1:n,
   thetaPlus     = theta;
   thetaPlus(i)  = thetaPlus(i) + EPSILON;
   thetaMinus    = theta;
   thetaMinus(i) = thetaMinus(i) - EPSILON;
   gradApprox(i) = (J(thetaPlus) - J(thetaMinus)) / (2 * EPSILON);
end;
```

مطمئن شوید: `DVec ≈ gradApprox`

نکات پیاده‌سازی

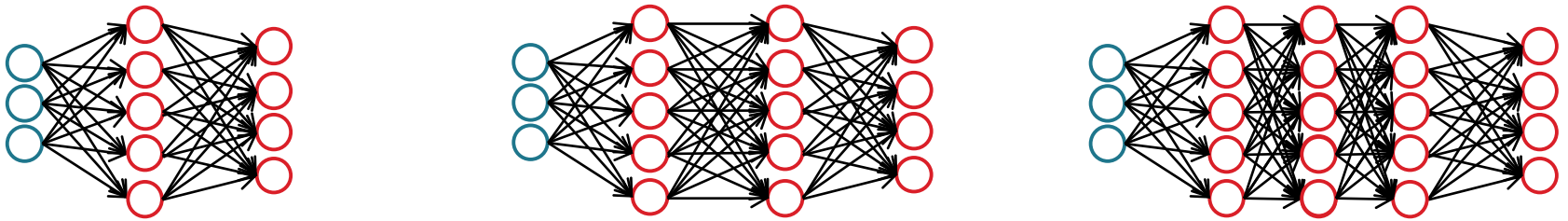
- الگوریتم پس‌انتشار خطا را به منظور محاسبه بردار **DVec** پیاده‌سازی کنید.
- تابع محاسبه عددی گرادیان را برای محاسبه **gradApprox** پیاده‌سازی کنید.
- اطمینان حاصل کنید این دو بردار شامل مقادیر مشابهی هستند.
- تابع بررسی گرادیان را غیر فعال کنید.
- از الگوریتم پس‌انتشار خطا به منظور آموزش شبکه عصبی استفاده کنید.

- قبل از شروع فرآیند آموزش در شبکه عصبی، مطمئن شوید تابع بررسی گرادیان را غیر فعال نموده‌اید، در غیر این صورت برنامه شما بسیار آهسته اجرا خواهد شد.

جمع‌بندی

آموزش یک شبکه عصبی

- انتخاب یک معماری برای شبکه (الگوی اتصالات میان نورون‌ها)



- تعیین تعداد لایه‌ها و تعداد نورون‌ها در هر لایه.
  - تعداد واحدهای ورودی: برابر با تعداد ویژگی‌ها
  - تعداد واحدهای خروجی: برابر با تعداد کلاس‌ها
  - تعداد لایه‌های مخفی: معمولاً برابر با یک است، اما اگر بیش از یک لایه مخفی وجود داشته باشد بهتر است تعداد نورون‌ها در این لایه‌های مخفی با یکدیگر برابر باشد.

پیاده‌سازی یک شبکه عصبی (۱)

- مقداردهی تصادفی به وزن‌ها
- پیاده‌سازی مرحله انتشار رو به جلو به منظور محاسبه خروجی شبکه به ازای هر ورودی مانند $x^{(i)}$
- پیاده‌سازی یک تابع به منظور محاسبه مقدار تابع هزینه $J(\theta)$
- پیاده‌سازی مرحله پس‌انتشار خطا به منظور محاسبه مشتق‌های جزئی

```
for i = 1 : m {
     perform forward propagation and backpropagation using example (x^(i), y^(i))
     (Get activations a^(l) and delta terms δ^(l) for l = 2,…, L)
     compute Δ^(l) = Δ^(l) + δ^(l) (a^(l))^T
}
compute partial derivatives of  J(Θ) considering regularization term
```

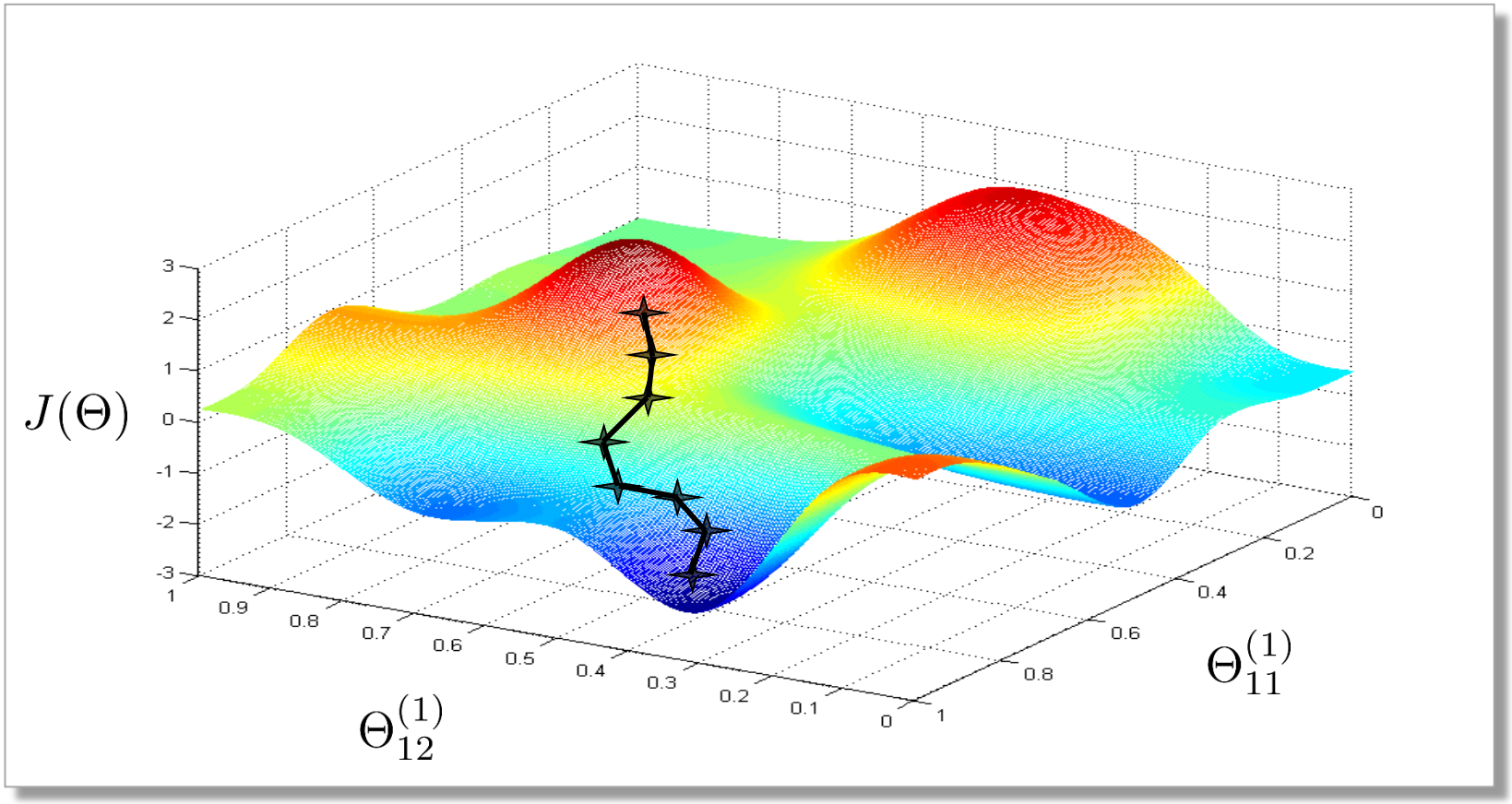
پیاده‌سازی یک شبکه عصبی (۲)

- بررسی گرادیان.
  - پیاده‌سازی یک تابع به منظور محاسبه عددی مقادیر گرادیان‌ها و مقایسه این مقادیر با مقادیر محاسبه شده به وسیله الگوریتم پس‌انتشار خطا
  - غیرفعال نمودن تابع بررسی گرادیان‌ها

- بهینه‌سازی.
  - استفاده از روش گرادیان کاهشی یا یکی از روش‌های بهینه‌سازی پیشرفته به همراه الگوریتم پس‌انتشار خطا به منظور سعی در کمینه‌سازی تابع هزینه $J(\Theta)$ به عنوان تابعی از پارامترهای Θ

- با توجه به «غیر کوژ» بودن تابع هزینه، گرادیان کاهشی یا هر یک از روش‌های بهینه‌سازی پیشرفته ممکن است در بهینه محلی گیر کنند.

یادآوری: گرادیان کاهشی



مثال: رانندگی خودمختار

رانندگی خودمختار