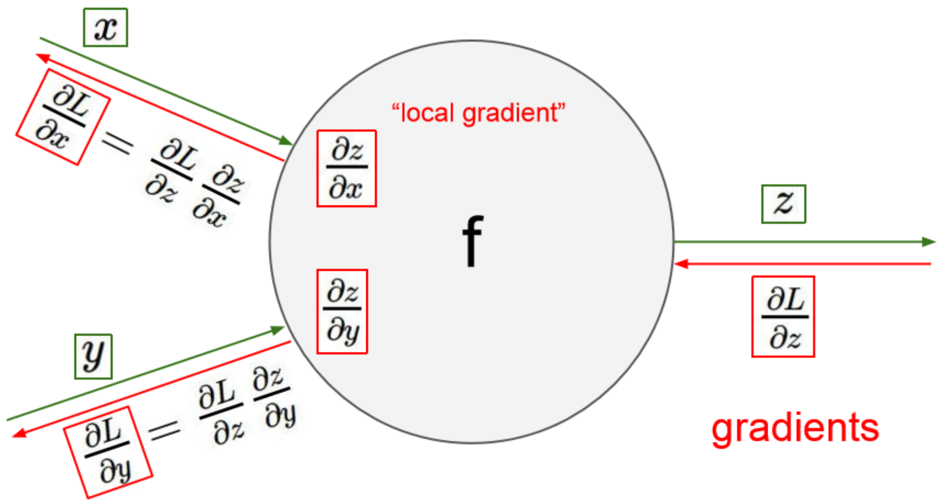
Backpropagation

`



参考资料

1. [ufldl resources]
2. [Machine Learning-Neural Networks:Learning-Backpropagation]

Forward and reverse mode

- The chain rule:

$$y = f(g(h(x))) = f(g(h(w_0))) = f(g(w_1)) = f(w_2) = w_3$$

$$\frac{dy}{dx} = \frac{dy}{dw_2}\frac{dw_2}{dw_1}\frac{dw_1}{dx}$$

- Forward accumulation(mode)

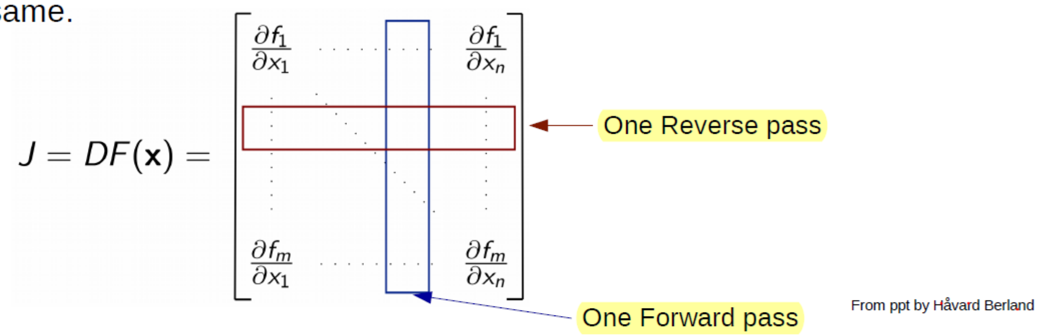
$$\frac{dw_i}{dx} = \frac{dw_i}{dw_{i-1}}\frac{dw_{i-1}}{dx} \qquad i = 0, 1, 2, 3$$

- Reverse accumulation

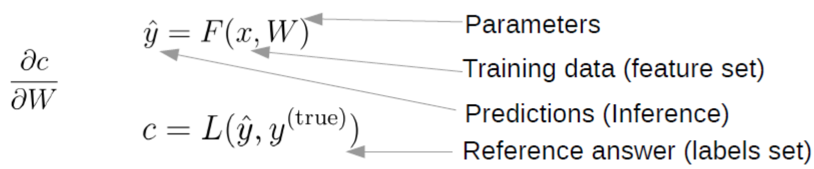
$$\frac{dy}{dw_i} = \frac{dy}{dw_{i+1}}\frac{dw_{i+1}}{dw_i} \qquad \boxed{i = 3, 2, 1, 0}$$

> Forward, 从里向外计算: $\frac{dw_0}{dx}, \frac{dw_1}{dx}, \frac{dw_2}{dx}, \frac{dw_3}{dx}$
>
> Reverse, 从外向里计算: $\frac{dy}{dw_3}, \frac{dy}{dw_2}, \frac{dy}{dw_1}, \frac{dy}{dw_0}$

- F: R^n → R^m
- Computational cost of one forward or reverse pass are roughly the same.



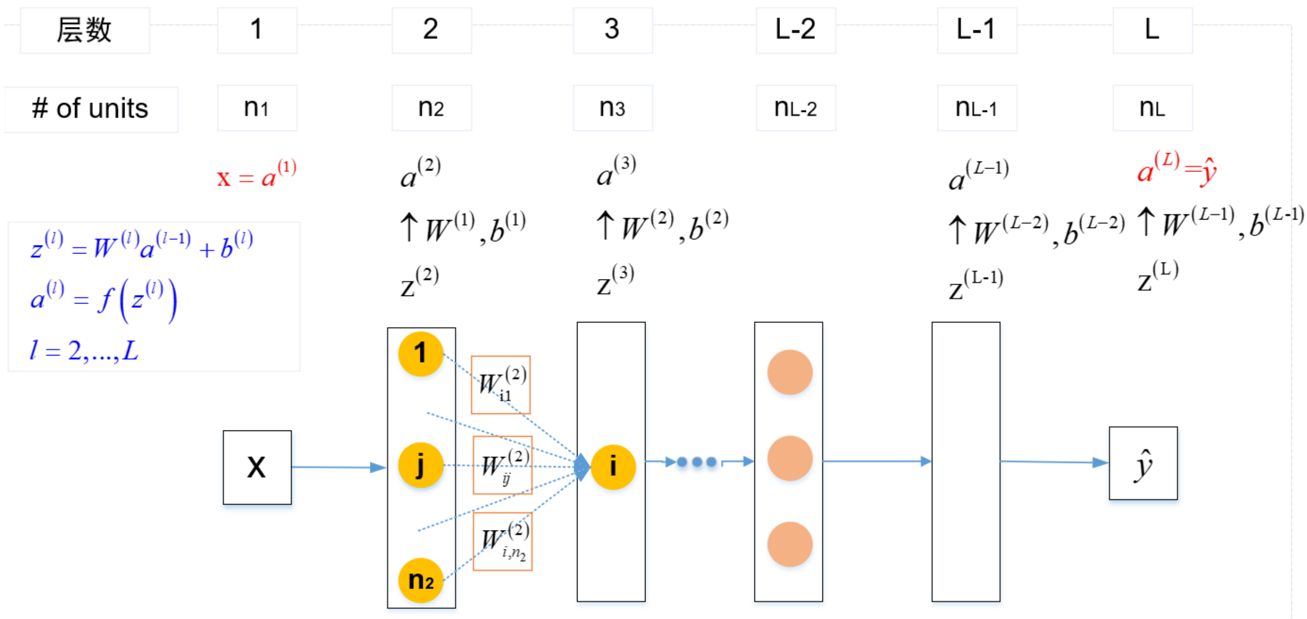
- Deep Learning: Backpropagation – specialized reverse mode



> $DF(x) = \frac{dF(x)}{dx}$

BP

model



Gradient descent

给定样本$\{(x^{(i)}, y^{(i)})\}_{i=1}^m$, 损失函数 为

$J(W, b) = J(W^{(1)}, \dots, W^{(L-1)}, b^{(1)}, \dots, b^{(L-1)}; \{(x^{(i)}, y^{(i)})\}_{i=1}^m)$,
$= \frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)})$

参数更新的迭代公式为($l = 1, 2, \dots, L - 1. i = 1, 2, \dots, n_{l+1}. j = 1, 2, \dots, n_l$):

$$
\begin{aligned}
W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \\
b_i^{(l)} &= b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b)
\end{aligned}
$$

其中:

$$
\begin{aligned}
\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x^{(i)}, y^{(i)}) \\
\frac{\partial}{\partial b_i^{(l)}} J(W, b) &= \frac{1}{m} \sum_{i=1}^m \frac{\partial}{\partial b_i^{(l)}} J(W, b; x^{(i)}, y^{(i)})
\end{aligned}
$$

为了更新参数，需要计算每一对样本点的损失函数关于参数的导数。记 $(x, y) = (x^{(i)}, y^{(i)})$, 上中的一项可改写为

$$
\begin{aligned}
\frac{\partial J(W, b; x, y)}{\partial W_{ij}^{(l)}} &= \frac{\partial J(W, b; x, y)}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}} &= \delta_i^{(l+1)} a_j^{(l)} \\
\frac{\partial J(W, b; x, y)}{\partial b_i^{(l)}} &= \frac{\partial J(W, b; x, y)}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial b_i^{(l)}} &= \delta_i^{(l+1)}.
\end{aligned}
$$

其中$\delta_i^{(l+1)}$ 在BP的推导中处于核心位置，可以理解为 “error" of node i in layer l [1]. 计算过程如下, reverse mode:

- $l = L$

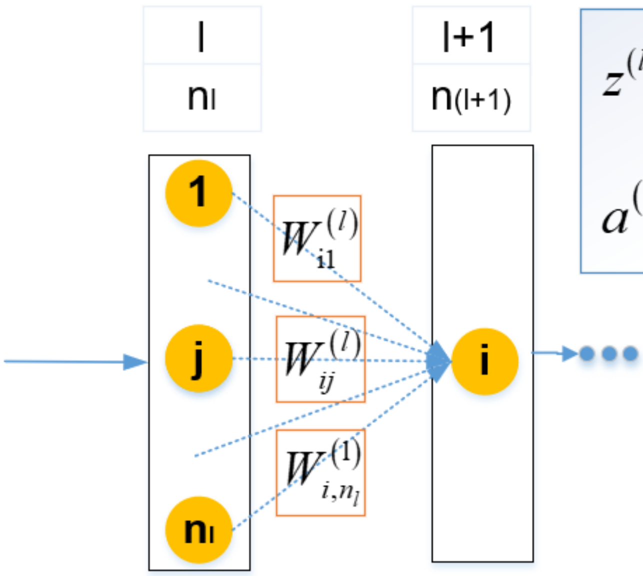
$$\begin{aligned}
\delta_i^{(L)} &= \frac{\partial J(W,b;x,y)}{\partial z_i^{(L)}} = \frac{\partial}{\partial z_i^{(L)}} \frac{1}{2}\|y-\hat{y}\|_2^2 \quad \leftarrow \hat{y} = a^{(L)} = f(z^{(L)}) \\
&= \frac{\partial}{\partial z_i^{(L)}} \frac{1}{2} \sum_{j=1}^{n_L} \left(y_j - a_j^{(L)}\right)^2 \\
&= -(y_i - a_i^{(L)}) \cdot f'(z_i^{(L)})
\end{aligned}$$

> Mark: UFLDL中是这样计算的，但是在Andrew Ng的课程中 和书《Python Machine Learining》[2] 用的是$\delta_i^{(L)} = a_i^{(L)} - y_i$. 可以把$f'(z_i^{(L)})$理解为一个scale，所以有没有都可以

- $l = L-1, L-2, \ldots, 2$

$$\begin{aligned}
\delta_i^{(l)} &= \frac{\partial J(W,b;x,y)}{\partial z_i^{(l)}} \\
&= \sum_{j=1}^{n_{l+1}} \frac{\partial J(W,b;x,y)}{\partial z_j^{(l+1)}} \frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}} \quad \leftarrow \frac{\partial J(W,b;x,y)}{\partial z_i^{(l+1)}} = \delta_i^{(l+1)} \\
&= \sum_{j=1}^{n_{l+1}} \left[\delta_i^{(l+1)} \frac{\partial z_j^{(l+1)}}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}}\right] \\
&= \left[\sum_{j=1}^{n_{l+1}} \delta_i^{(l+1)} W_{ji}^{(l)}\right] f'(z_i^{(l)})
\end{aligned}$$

其中 z,a,W,f 之间的关系为：



$$z^{(l+1)} = W^{(l)}_{n_{l+1}\times n_l} a^{(l)} + b^{(l)}$$

$$a^{(l)} = f\left(z^{(l)}\right)$$

1. Machine leraning course, Andrew Ng ↩
2. 书中测试的例子有$f'(z_i^{(L)})$，收敛速度更慢。↩