第四讲 反向传播

《深度学习》

南开大学 人工智能学院

算梯度

基于梯度的优化

- 基于梯度的优化算法（即一阶算法）是深度学习中最常用的训练算法
  - 二阶矩阵昂贵的计算和存储开销，使得二阶算法在深度学习中并不常用
  - 一阶优化算法的不足是求解高精度解需要较长时间，但机器学习一般不需要求解到高精度解
    - 训练太长时间可能会导致过拟合，甚至需要提前终止
    - 训练误差和测试误差有时并不一致，没必要把训练误差降得很低
      - 交叉熵 v.s. 错误率
    - 机器学习需要在较短时间求解一个合适精度的解，这是一阶优化算法的优势
- 基于梯度的优化算法的核心开销是计算梯度
- 反向传播：深度神经网络中计算梯度的高效方法
  - 注意：反向传播不是网络训练算法，仅是计算梯度的方法，SGD、Adam是网络训练算法

4.1 导数基础回顾

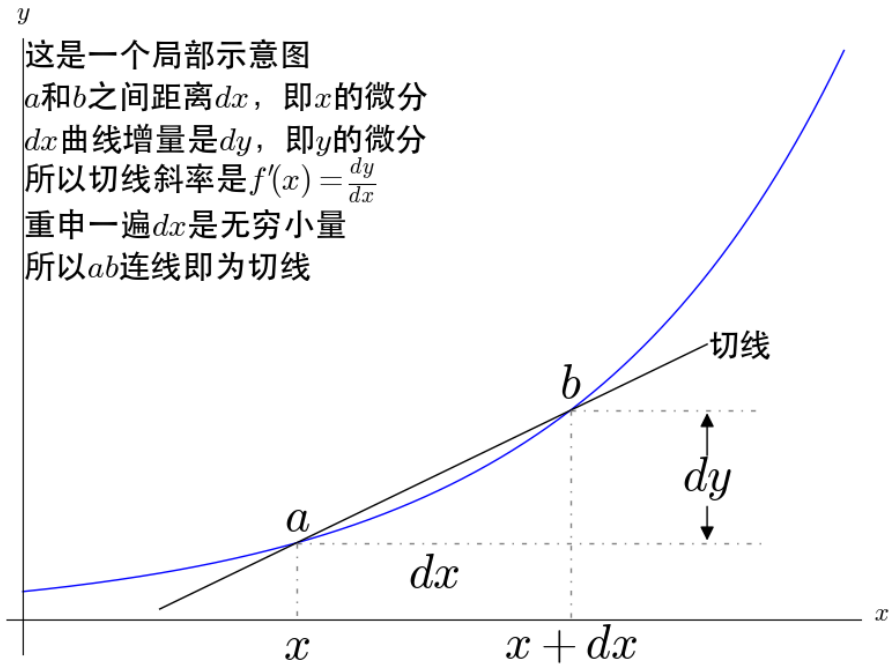
- 当 x和 $y=f(x)$ 均为标量时，y关于x的导数，记为

  $f'(x)$或 $\frac{\partial y}{\partial x}$ 或$\frac{\partial f}{\partial x}$，定义为

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

  表示 $f(x)$在点 x 处的斜率，它表明当 x 变化ε时，y的变化比例

$$f(x+\varepsilon)\approx f(x)+\varepsilon f'(x)$$



- 运算法则
  - 常数相乘　　$[Cf(x)]'=Cf'(x)$
  - 加法　　$[f(x)+g(x)]'=f'(x)+g'(x)$
  - 乘法　　$[f(x)g(x)]'=f(x)g'(x)+f'(x)g(x)$

4.1 导数基础回顾

- 将导数由标量扩展到向量
  - x为列向量，$y=f(x)$为标量

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \frac{\partial y}{\partial x}=\left[\frac{\partial y}{\partial x_1},\frac{\partial y}{\partial x_2},\cdots,\frac{\partial y}{\partial x_n}\right]$$

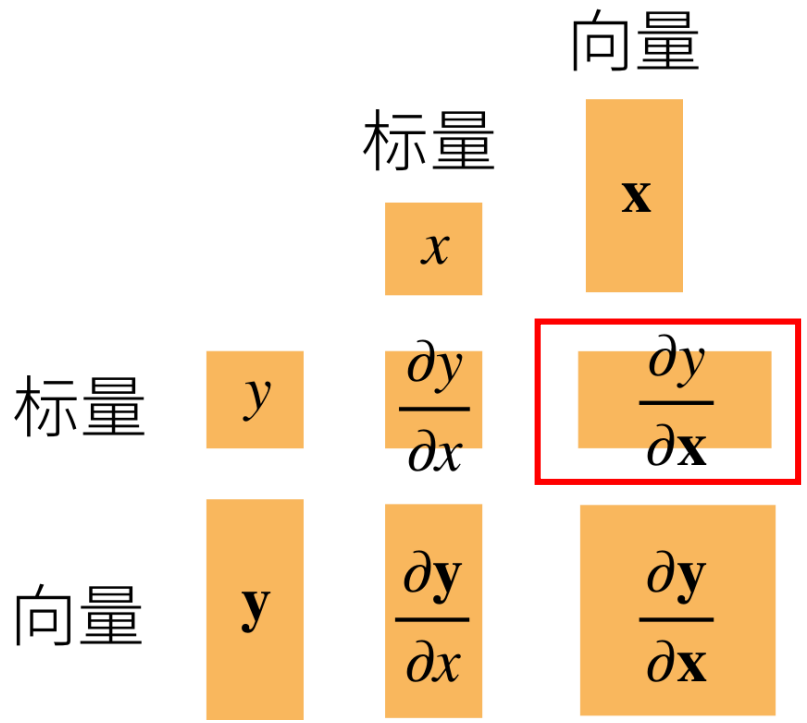
$\frac{\partial y}{\partial x}$ 为行向量，其中

$$\frac{\partial y}{\partial x_i}=\lim_{h\to 0}\frac{f(x_1,\cdots,x_{i-1},x_i+h,x_{i+1},\cdots,x_n)-f(x)}{h}$$

  - **梯度** $\nabla_x f$ 为列向量，与 x 保持一致

$$\nabla_x f=\begin{bmatrix} \nabla_{x_1} f \\ \nabla_{x_2} f \\ \vdots \\ \nabla_{x_n} f \end{bmatrix}=\begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad \nabla_x f=\left(\frac{\partial y}{\partial x}\right)^T$$

  - 有文献将 $\frac{\partial f}{\partial x}$ 写成列向量，根据上下文而定

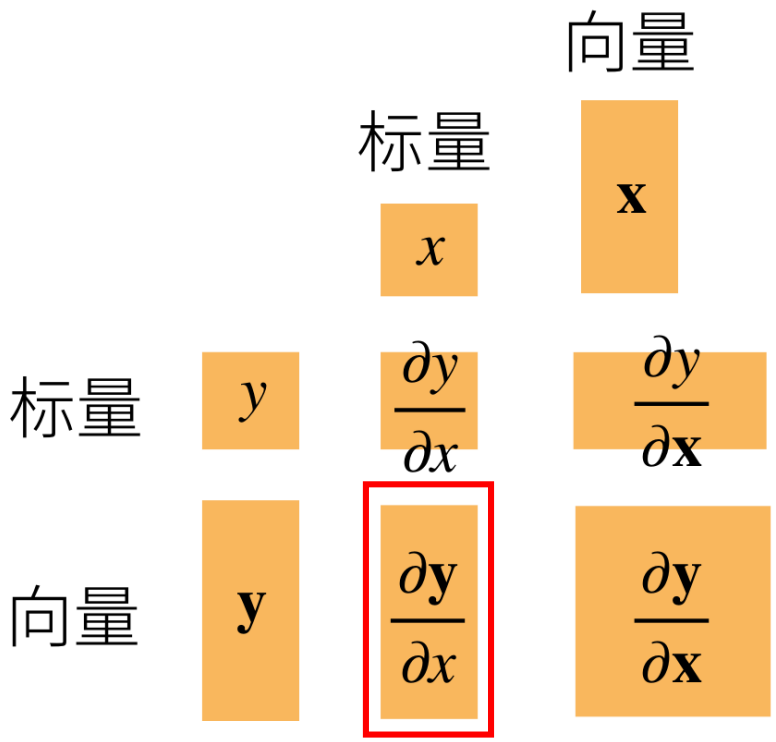


4.1 导数基础回顾

- 将导数由标量扩展到向量
  - x 为标量，y 为列向量

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_m}{\partial x} \end{bmatrix}$$

则 $\frac{\partial y}{\partial x}$ 为列向量

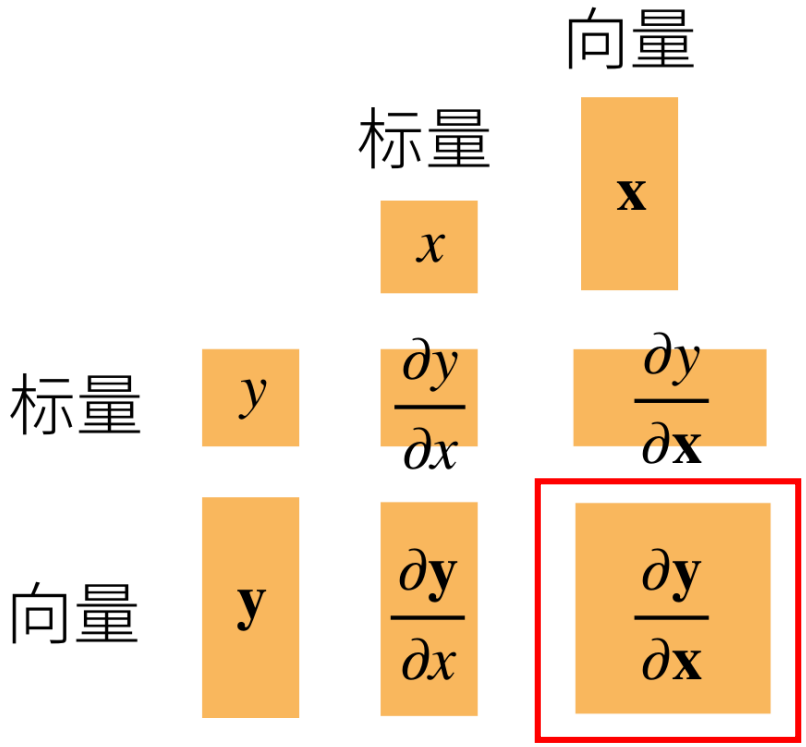


4.1 导数基础回顾

- 将导数由标量扩展到向量
  - x为列向量,y 为列向量

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

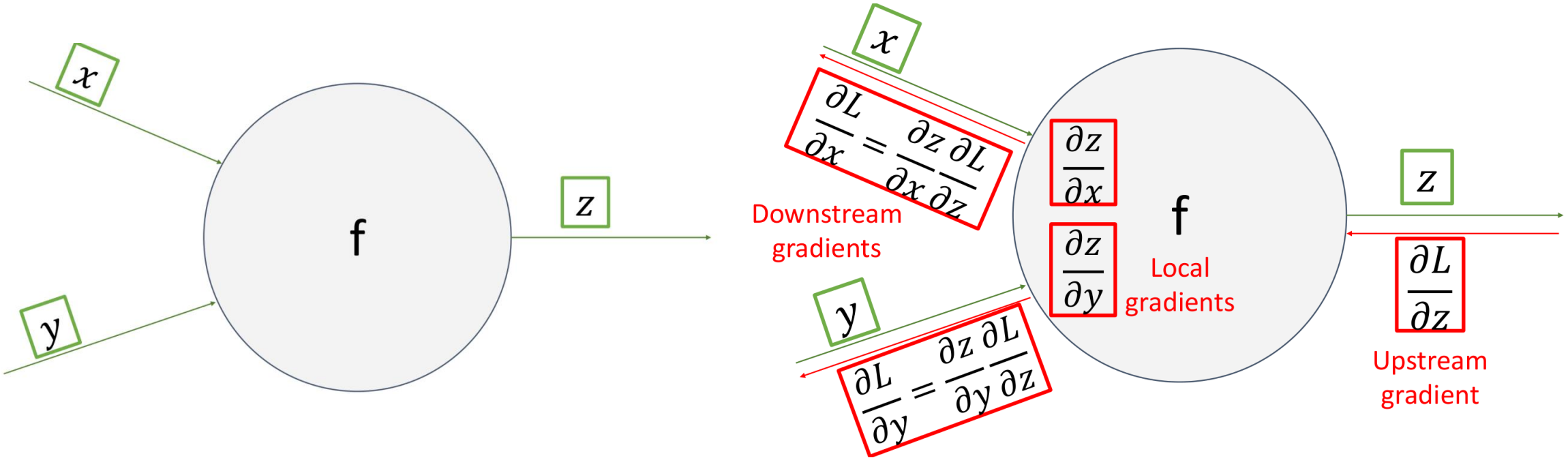
$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$



则 $\frac{\partial y}{\partial x}$ 为矩阵，矩阵的行数为y的元素个数，矩阵的列数为x的元素个数；该矩阵又称为**Jacobian矩阵**

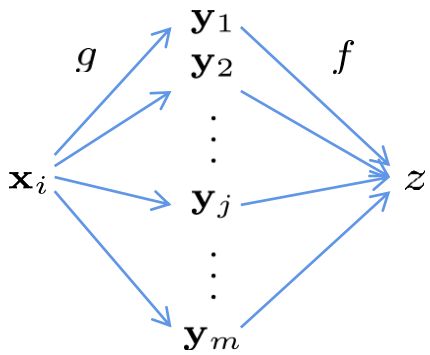
4.2 链式法则

- Loss $= L(z), \quad z = f(x, y)$ ，由链式法则有 $\dfrac{\partial L}{\partial x} = \dfrac{\partial L}{\partial z}\dfrac{\partial z}{\partial x}, \quad \dfrac{\partial L}{\partial y} = \dfrac{\partial L}{\partial z}\dfrac{\partial z}{\partial y}$
  - **链式法则**就是将复合函数的导数分解为各个组成函数的导数的乘积



downstream gradient = local gradient × upstream gradient

4.2* 链式法则



- 考虑 $x \in \mathbb{R}^n, \quad y \in \mathbb{R}^m$
- 令 $f : \mathbb{R}^m \to \mathbb{R}, \quad g : \mathbb{R}^n \to \mathbb{R}^m$
- 假设 $y = g(x), \quad z = f(y)$，则有

$$\frac{\partial z}{\partial x_i} = \sum_{j=1}^{m} \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

所有的 y_j 都受 x_i 影响

- 写成矩阵形式

$$\frac{\partial z}{\partial x} = \left[\frac{\partial z}{\partial x_1}, \cdots, \frac{\partial z}{\partial x_n} \right] \longleftarrow 1 \times n$$

$$= \left[\sum_{j=1}^{m} \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_1}, \cdots, \sum_{j=1}^{m} \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_n} \right]$$

$$= \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

$1 \times m$ $\qquad$ $m \times n$

带入验证即可

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial z}{\partial y} = \left[\frac{\partial z}{\partial y_1}, \frac{\partial z}{\partial y_2}, \cdots \frac{\partial z}{\partial y_m} \right]$$

通过定义 $\frac{\partial z}{\partial x}$、$\frac{\partial z}{\partial y}$、$\frac{\partial y}{\partial x}$ 的形状，使得 $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$ 是良定义的

4.3 标量形式的反向传播

- 考虑流经输入层、隐藏层、输出层的信息都是标量的简化网络

$$
\begin{aligned}
h_1 &= \sigma\left(w_1 x + b_1\right) \\
h_2 &= \sigma\left(w_2 h_1 + b_2\right) \\
&\cdots \\
h_k &= \sigma\left(w_k h_{k-1} + b_k\right) \\
&\cdots \\
y = h_L &= \sigma\left(w_L h_{L-1} + b_L\right)
\end{aligned}
$$

如何计算 $\frac{\partial f}{\partial w_l}$ 和 $\frac{\partial f}{\partial b_l}$？

其中 $l = 1, 2, \cdots, L$

- 记训练过程中的目标函数为$f(w, b)$
- 为了简化描述，本讲考虑只有一个样本的情形
  - 在随机梯度下降中，每次迭代随机选择一个小批量样本，计算每个样本的梯度，然后取平均

4.3 标量形式的反向传播

- **前向传播**指的是：从输入层到输出层按顺序计算和存储神经网络中每层的结果

$$
\begin{aligned}
h_1 &= \sigma\left(w_1 x + b_1\right) \\
h_2 &= \sigma\left(w_2 h_1 + b_2\right) \\
&\cdots \\
h_k &= \sigma\left(w_k h_{k-1} + b_k\right) \\
&\cdots \\
y = h_L &= \sigma\left(w_L h_{L-1} + b_L\right)
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
z_1 &= w_1 x + b_1, \quad h_1 = \sigma(z_1) \\
z_2 &= w_2 h_1 + b_2, \quad h_2 = \sigma(z_2) \\
&\cdots \\
\boxed{z_k &= w_k h_{k-1} + b_k, \quad h_k = \sigma(z_k)} \\
z_{k+1} &= w_{k+1} h_k + b_{k+1}, \quad h_{k+1} = \sigma(z_{k+1}) \\
&\cdots \\
z_L &= w_L h_{L-1} + b_L, \quad y = h_L = \sigma(z_L)
\end{aligned}
$$

- 在前向传播中，w_k只通过z_k作用于最终的f，z_k又通过h_k作用于最终的f，由链式法则有

$$
\frac{\partial f}{\partial w_k} = \frac{\partial f}{\partial h_k} \frac{\partial h_k}{\partial z_k} \frac{\partial z_k}{\partial w_k} = \frac{\partial f}{\partial h_k} \sigma'(z_k) h_{k-1}
$$

- b_k同样只通过z_k作用于最终的f，z_k又通过h_k作用于最终的f，由链式法则有

$$
\frac{\partial f}{\partial b_k} = \frac{\partial f}{\partial h_k} \frac{\partial h_k}{\partial z_k} \frac{\partial z_k}{\partial b_k} = \frac{\partial f}{\partial h_k} \sigma'(z_k)
$$

- 现在只需要计算 $\dfrac{\partial f}{\partial h_k}$

4.3 标量形式的反向传播

- **前向传播**指的是：按顺序（从输入层到输出层）计算和存储神经网络中每层的结果

$$
\begin{aligned}
h_1 &= \sigma\left(w_1 x + b_1\right) \\
h_2 &= \sigma\left(w_2 h_1 + b_2\right) \\
&\cdots \\
h_k &= \sigma\left(w_k h_{k-1} + b_k\right) \\
&\cdots \\
y = h_L &= \sigma\left(w_L h_{L-1} + b_L\right)
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
z_1 &= w_1 x + b_1, \quad h_1 = \sigma(z_1) \\
z_2 &= w_2 h_1 + b_2, \quad h_2 = \sigma(z_2) \\
&\cdots \\
z_k &= w_k h_{k-1} + b_k, \quad h_k = \sigma(z_k) \\
z_{k+1} &= w_{k+1} h_k + b_{k+1}, \quad h_{k+1} = \sigma(z_{k+1}) \\
&\cdots \\
z_L &= w_L h_{L-1} + b_L, \quad y = h_L = \sigma(z_L)
\end{aligned}
$$

- 在前向传播中，h_k只通过 z_{k+1} 作用于最终的 f，z_{k+1}只通过 h_{k+1} 作用于最终的 f，由链式法则有

$$
\frac{\partial f}{\partial h_k} = \frac{\partial f}{\partial h_{k+1}} \frac{\partial h_{k+1}}{\partial z_{k+1}} \frac{\partial z_{k+1}}{\partial h_k} = \frac{\partial f}{\partial h_{k+1}} \sigma'(z_{k+1}) w_{k+1}
$$

- 这样就把 $\frac{\partial f}{\partial h_k}$ 的计算转换为递归计算，递归初始时有 $\frac{\partial f}{\partial h_L} = \frac{\partial f}{\partial y}$

4.3 标量形式的反向传播

- 标量形式的反向传播算法

可根据损失函数的具体形式直接计算

前向传播，存储每一个 h_k 值和 z_k 值

初始化：$\dfrac{\partial f}{\partial h_L} = \dfrac{\partial f}{\partial y}$

for $k = L, L-1, \cdots, 1$

$$\frac{\partial f}{\partial w_k} = \frac{\partial f}{\partial h_k}\sigma'(z_k)h_{k-1}$$

$$\frac{\partial f}{\partial b_k} = \frac{\partial f}{\partial h_k}\sigma'(z_k)$$

$$\frac{\partial f}{\partial h_{k-1}} = \frac{\partial f}{\partial h_k}\sigma'(z_k)w_k$$

end for

对所有 $l = 1, 2, \cdots, L$，返回 $\dfrac{\partial f}{\partial w_l}$ 和 $\dfrac{\partial f}{\partial b_l}$

$k = L$

$$\frac{\partial f}{\partial w_L} = \frac{\partial f}{\partial h_L}\sigma'(z_L)h_{L-1}$$

$$\frac{\partial f}{\partial b_L} = \frac{\partial f}{\partial h_L}\sigma'(z_L)$$

$$\frac{\partial f}{\partial h_{L-1}} = \frac{\partial f}{\partial h_L}\sigma'(z_L)w_L$$

$k = L-1$

$$\frac{\partial f}{\partial w_{L-1}} = \frac{\partial f}{\partial h_{L-1}}\sigma'(z_{L-1})h_{L-2}$$

$$\frac{\partial f}{\partial b_{L-1}} = \frac{\partial f}{\partial h_{L-1}}\sigma'(z_{L-1})$$

$$\frac{\partial f}{\partial h_{L-2}} = \frac{\partial f}{\partial h_{L-1}}\sigma'(z_{L-1})w_{L-1}$$

不指

4.3 标量形式的反向传播

- 标量形式的反向传播算法

可根据损失函数的具体形式直接计算

前向传播，存储每一个 h_k 值和 z_k 值

初始化：$\dfrac{\partial f}{\partial h_L} = \dfrac{\partial f}{\partial y}$

for $k = L, L-1, \cdots, 1$

$$\frac{\partial f}{\partial w_k} = \frac{\partial f}{\partial h_k}\sigma'(z_k)h_{k-1}$$

$$\frac{\partial f}{\partial b_k} = \frac{\partial f}{\partial h_k}\sigma'(z_k)$$

$$\frac{\partial f}{\partial h_{k-1}} = \frac{\partial f}{\partial h_k}\sigma'(z_k)w_k$$

end for

对所有 $l = 1, 2, \cdots, L$，返回 $\dfrac{\partial f}{\partial w_l}$ 和 $\dfrac{\partial f}{\partial b_l}$

- **反向传播**：根据微积分中的链式规则，按相反的顺序从输出层到输入层遍历网络，计算神经网络损失函数关于参数的梯度
- 激活值：在深度学习中，h_k 称为激活值

4.3 标量形式的反向传播

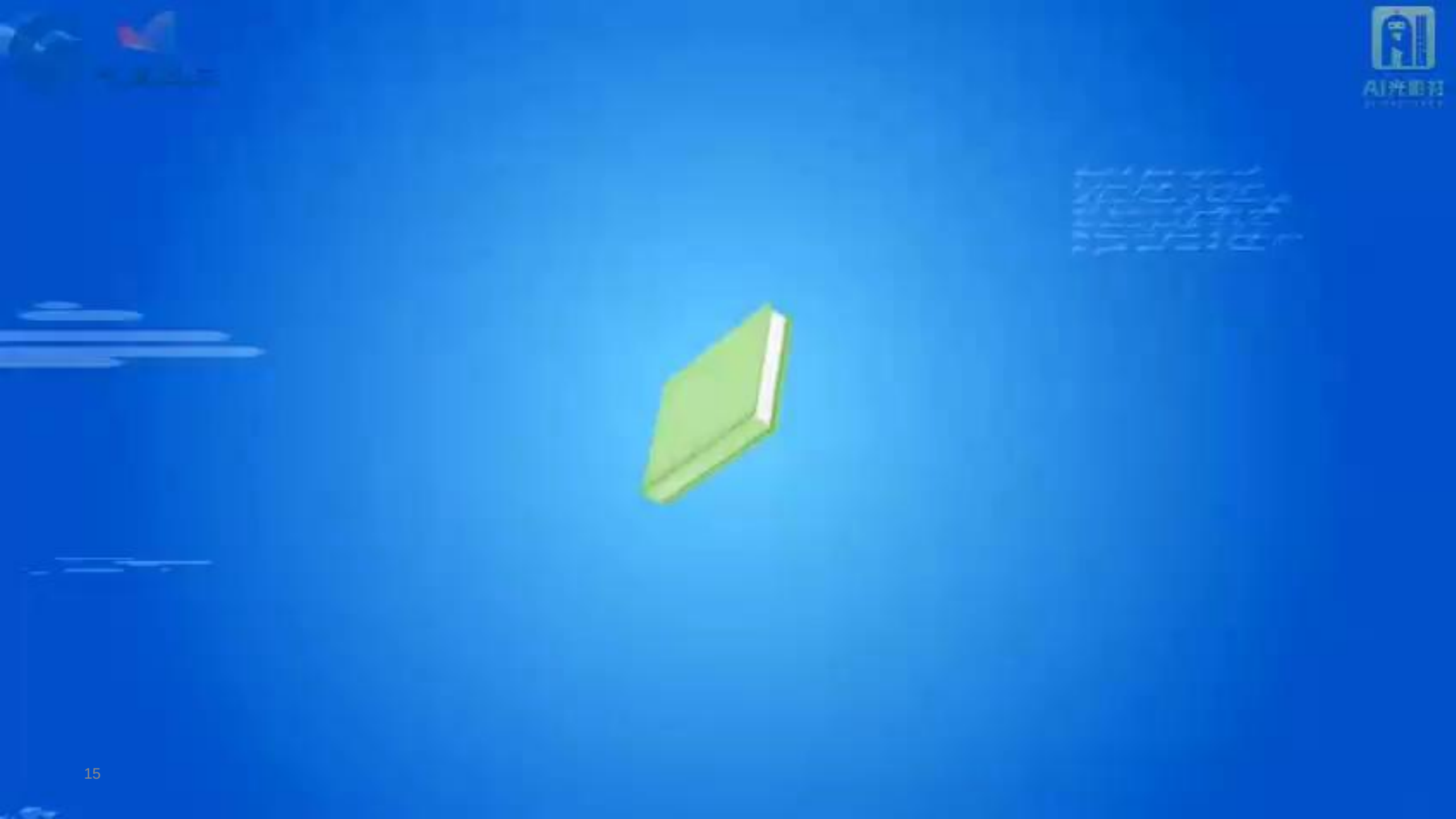
- PyTorch实现

API: Tensor.backward(gradient=None, retain_graph=None, create_graph=False, inputs=None)

参数：一般使用默认设置即可，细节参考 PyTorch 文档。

示例：

```
x = torch.randn(5)
y = torch.randn(3)
w = torch.randn(5, 3, requires_grad=True)
b = torch.randn(3, requires_grad=True)
z = torch.matmul(x, w)+b
# z = w^T x + b
loss = nn.MSELoss()
output = loss(z, y)
# 反向传播，计算梯度
output.backward()
# 输出 output 关于 w 的梯度
print(w.grad)
# 输出 output 关于 b 的梯度
print(b.grad)
```



4.4 梯度消失和梯度爆炸

概念
引入问题
解决方法

- 将递归形式的梯度计算写成连乘形式

$$\frac{\partial f}{\partial h_{k-1}} = \frac{\partial f}{\partial h_k}\sigma'(z_k)w_k$$

$$= \frac{\partial f}{\partial h_{k+1}}\sigma'(z_{k+1})w_{k+1}\sigma'(z_k)w_k$$

$$= \frac{\partial f}{\partial h_{k+2}}\sigma'(z_{k+2})w_{k+2}\sigma'(z_{k+1})w_{k+1}\sigma'(z_k)w_k$$

$$= \frac{\partial f}{\partial h_L}\sigma'(z_L)w_L \cdots \sigma'(z_{k+1})w_{k+1}\sigma'(z_k)w_k$$

$$= \frac{\partial f}{\partial h_L}\Pi_{t=k}^{L}\sigma'(z_t)w_t$$

$$\frac{\partial f}{\partial h^k} = \frac{\partial f}{\partial h^{k+1}}\sigma'(z^{k+1})w^{k+1}$$

$$\frac{\partial f}{\partial h_{k+1}} = \frac{\partial f}{\partial h_{k+2}}\sigma'(z_{k+2})w_{k+2}$$

$$\frac{\partial f}{\partial w_k} = \frac{\partial f}{\partial h_k}\sigma'(z_k)h_{k-1}$$

$$\frac{\partial f}{\partial b_k} = \frac{\partial f}{\partial h_k}\sigma'(z_k)$$

4.4 梯度消失和梯度爆炸

- 将递归形式的梯度计算写成连乘形式

$$\frac{\partial f}{\partial h_{k-1}} = \frac{\partial f}{\partial h_L} \Pi_{t=k}^{L} \sigma'(z_t) w_t$$

$$\frac{\partial f}{\partial w_k} = \frac{\partial f}{\partial h_k} \sigma'(z_k) h_{k-1}$$

$$\frac{\partial f}{\partial b_k} = \frac{\partial f}{\partial h_k} \sigma'(z_k)$$

- **梯度消失**：在深层网络中，梯度需要通过多个网络层进行反向传播。根据链式法则，梯度在传播过程中会不断相乘，当层数较多时，梯度值可能会以指数形式衰减并趋近于零，导致梯度消失

- 如果$\sigma'(z_t)$和w_t对所有t都比较小，那么累乘之后的$\frac{\partial f}{\partial h_{k-1}}$在$k$较小时可能会趋于0，这称为梯度消失
  - 使用sigmoid做激活函数时，sigmoid对大部分输入的导数都接近于0，即使最大值也只有1/4，因此$\sigma'(z_t)$很小，故sigmoid会导致梯度消失
- 梯度消失使得优化算法训练缓慢
  - 每一次迭代前进很小
- 改善梯度消失的方法
  - 改用ReLU做激活函数：当$x > 0$时，$\mathrm{ReLU}(x)' = 1$，从而减缓梯度消失问题
  - BN：对每一层做标准化操作，使每一层的输出稳定在合理范围内，从而即使是对于sigmoid，将其输入标椎化在0附近，其导数不会接近0
  - ResNet（见第5讲）中的旁支连接

4.4 梯度消失和梯度爆炸

- 将递归形式的梯度计算写成连乘形式

$$\frac{\partial f}{\partial h_{k-1}} = \frac{\partial f}{\partial h_L} \Pi_{t=k}^{L} \sigma'(z_t) w_t$$

$$\frac{\partial f}{\partial w_k} = \frac{\partial f}{\partial h_k} \sigma'(z_k) h_{k-1}$$

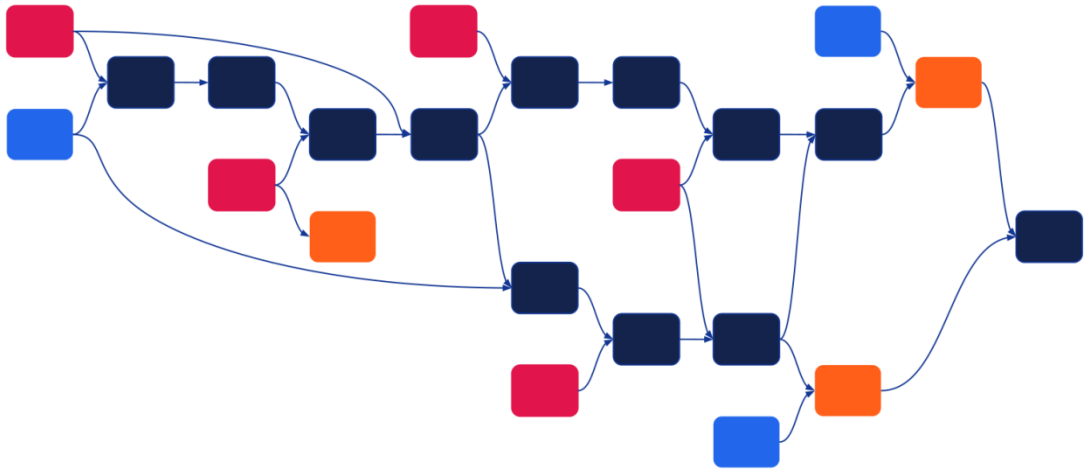
$$\frac{\partial f}{\partial b_k} = \frac{\partial f}{\partial h_k} \sigma'(z_k)$$

- **梯度爆炸**：深层网络中的梯度在传播过程中也可能因链式法则的连乘效应而迅速增长，甚至呈指数级增长，导致网络参数更新过大，网络不稳定

- 如果$\sigma'(z_t)$和w_t对所有t都比较大，那么累乘之后的$\frac{\partial f}{\partial h_{k-1}}$在$k$较小时可能会趋于无穷大，这称为梯度爆炸
- 梯度爆炸使得优化算法不稳定
- 解决梯度爆炸的方法
  - ℓ_2正则化 / 权重衰减：避免w_t极端大
  - 梯度截断（见第7讲）
  - BN层

4.5 计算图

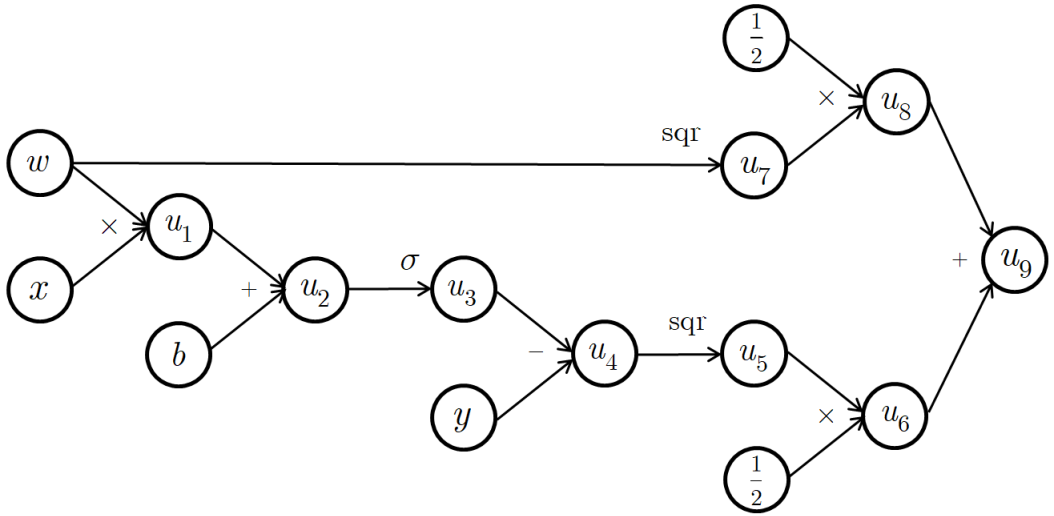
- 一般网络的BP算法涉及大量的矩阵微分，十分繁琐
- 如果更改网络结构，需要重新推导
- 针对任意网络结构的自动微分方法？
  - 解决办法：计算图
  - 计算图使用“图”这种数据结构建模前向传播过程，将变量（包括输入数据、网络参数、中间计算结果等）表示为图中的节点，变量之间的计算关系表示为图中的边，并基于计算图进行反向传播



4.6.1 前向传播—计算图构建

- 考虑 $f(w,b)=\frac{1}{2}\left(\sigma(wx+b)-y\right)^2+\frac{1}{2}w^2$
  - 只有一个隐藏层的网络
- 计算图构建过程
  - w和b是函数的参数，x和y可以认为是输入数据，他们都是输入变量，**每个输入变量一个节点，使用圆圈表示节点**
  - w和x相乘，得到中间变量，记为u_1，并且分别画一条从w和x到u_1的有向边，在u_1旁边使用标记×表示相乘
  - ……
  - 把$f(w,b)$的计算分解为一系列加减乘除平方等最小的计算单元:

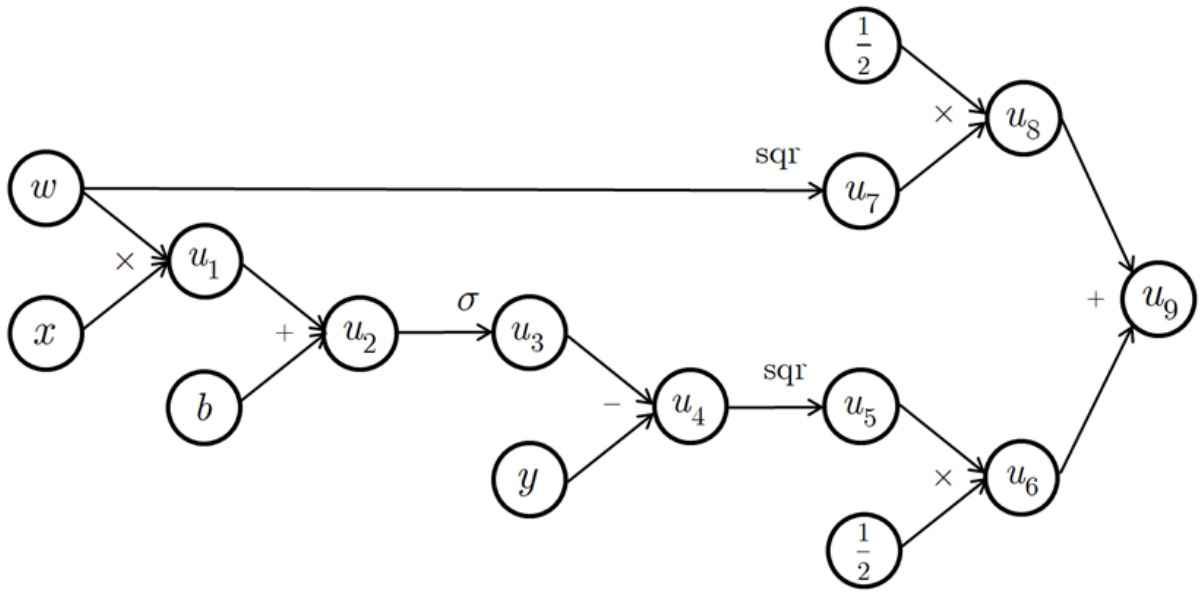
$$u_1=wx,\quad u_2=u_1+b,\quad u_3=\sigma(u_2),\quad u_4=u_3-y,$$
$$u_5=u_4^2,\quad u_6=\frac{1}{2}u_5,\quad u_7=w^2,\quad u_8=\frac{1}{2}u_7,\quad u_9=u_6+u_8$$



4.6.1 前向传播—计算图构建

- 在构造计算图时，需要做一些额外的工作
  - 为了方便反向遍历计算图，在基于前向传播构造计算图时，记录每一个节点的创建顺序，一般把输入变量放在前面，中间计算变量放在中间，结果变量放在最后。对于前面例子，节点顺序为

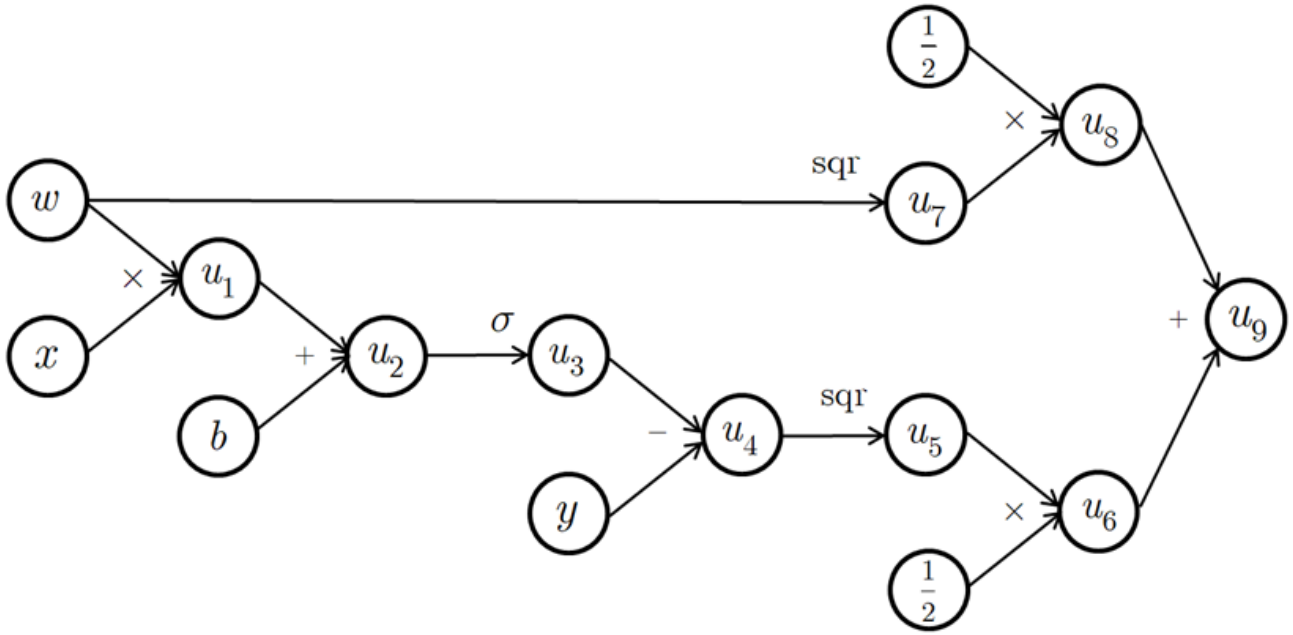
$$w, x, b, y, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9$$



4.6.1 前向传播—计算图构建

- 在构造计算图时，需要做一些额外的工作
  - 为了方便通过链式法则计算梯度，对每一个节点u使用集合$S(u)$记录它的所有子节点（有向边指向的下游节点）

$$
\begin{aligned}
S(w) &= \{u_1, u_7\} \\
S(x) &= u_1 \\
S(u_1) &= u_2 \\
S(b) &= u_2 \\
S(u_2) &= u_3 \\
S(u_3) &= u_4 \\
S(y) &= u_4 \\
S(u_4) &= u_5 \\
S(u_5) &= u_6 \\
S(u_6) &= u_9 \\
S(u_7) &= u_8 \\
S(u_8) &= u_9
\end{aligned}
$$

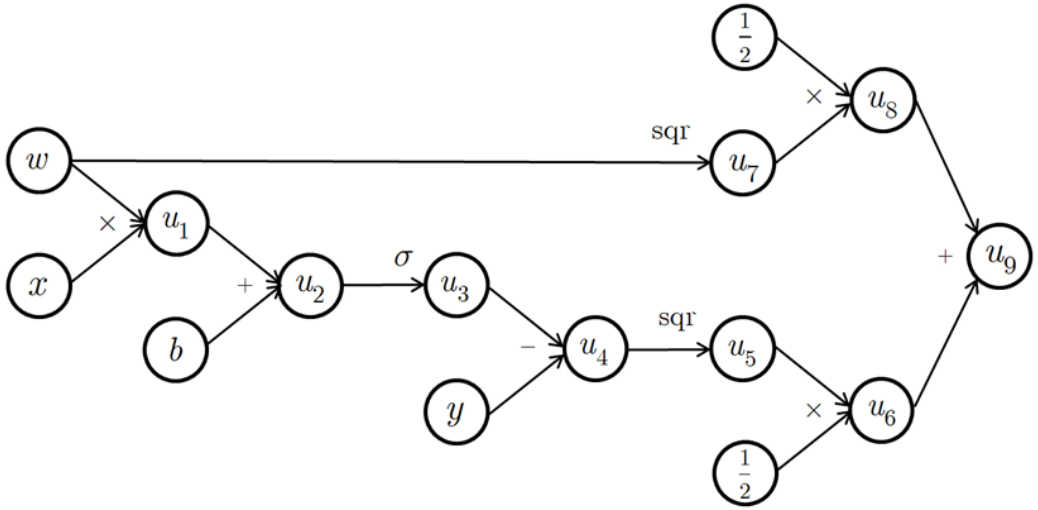


4.6.1 前向传播—计算图构建

- 在构造计算图时，需要做一些额外的工作
  - 对每个计算单元，编写输出值对输入值的求导法则。对于上面例子，求导如下：

$$u_1 = wx, \quad u_2 = u_1 + b, \quad u_3 = \sigma(u_2), \quad u_4 = u_3 - y,$$
$$u_5 = u_4^2, \quad u_6 = \frac{1}{2}u_5, \quad u_7 = w^2, \quad u_8 = \frac{1}{2}u_7, \quad u_9 = u_6 + u_8$$

$$\frac{\partial u_1}{\partial w} = x, \quad \frac{\partial u_2}{\partial u_1} = 1, \quad \frac{\partial u_2}{\partial b} = 1, \quad \frac{\partial u_3}{\partial u_2} = \sigma'(u_2), \quad \frac{\partial u_4}{\partial u_3} = 1, \quad \frac{\partial u_5}{\partial u_4} = 2u_4,$$
$$\frac{\partial u_6}{\partial u_5} = \frac{1}{2}, \quad \frac{\partial u_7}{\partial w} = 2w, \quad \frac{\partial u_8}{\partial u_7} = \frac{1}{2}, \quad \frac{\partial u_9}{\partial u_6} = 1, \quad \frac{\partial u_9}{\partial u_8} = 1,$$

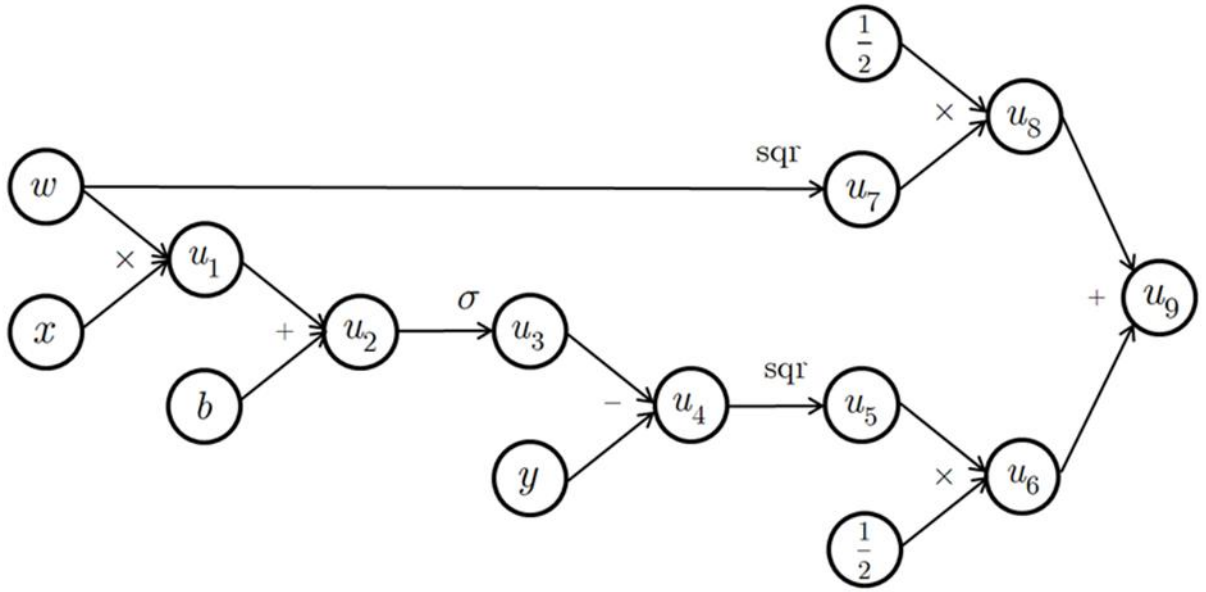


由于计算图的边表示基本的计算单元，因此只需要在程序中编写这些基本计算单元的求导法则，这也是为什么我们要把前向过程拆解到最小的计算单元

4.6.2 反向传播—计算图使用

- 使用计算图进行反向传播，计算 $\frac{\partial f}{\partial w}$ 和 $\frac{\partial f}{\partial b}$
  - 基于构建计算图时记录的节点创建顺序反向遍历计算图，使用链式法则计算 f关于每个节点的导数

$$w, x, b, y, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9$$



4.6.2 反向传播—计算图使用

- 使用计算图进行反向传播，计算 $\frac{\partial f}{\partial w}$ 和 $\frac{\partial f}{\partial b}$
  - 基于构建计算图时记录的节点创建顺序反向遍历计算图，使用链式法则计算 f 关于每个节点的导数

$$w, x, b, y, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9$$

  - f 关于 u_9 的导数为1，根据链式法则，f 关于 u_8 的导数 $\frac{\partial f}{\partial u_8} = \frac{\partial f}{\partial u_9}\frac{\partial u_9}{\partial u_8}$，其中 $\frac{\partial f}{\partial u_9}$ 我们刚刚计算过，而 $\frac{\partial u_9}{\partial u_8}$ 是基本计算单元 $u_9 = u_6 + u_8$ 中的输出关于输入的导数，可以直接带入编写的求导法则计算
  - 类似地，我们可以沿着与前向计算相反的方向逐步计算 f 关于每个节点的导数

$$\frac{\partial f}{\partial u_9} = 1, \quad \frac{\partial f}{\partial u_8} = \frac{\partial f}{\partial u_9}\frac{\partial u_9}{\partial u_8} = 1, \quad \frac{\partial f}{\partial u_7} = \frac{\partial f}{\partial u_8}\frac{\partial u_8}{\partial u_7} = \frac{1}{2}, \quad \frac{\partial f}{\partial u_6} = \frac{\partial f}{\partial u_9}\frac{\partial u_9}{\partial u_6} = 1,$$

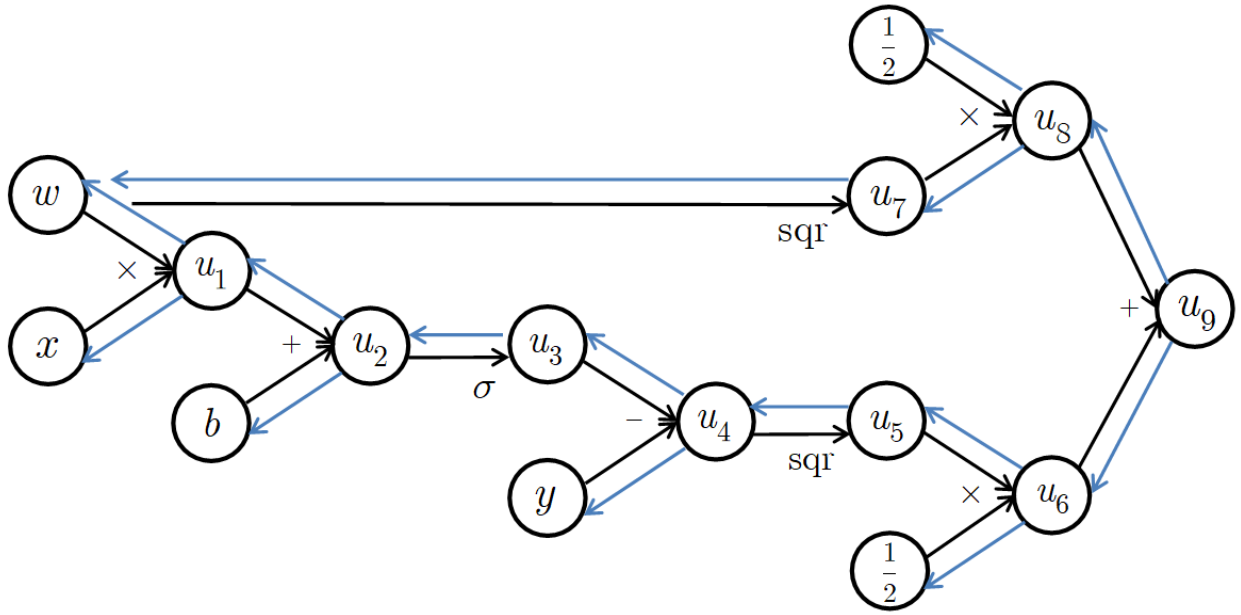
$$\frac{\partial f}{\partial u_5} = \frac{\partial f}{\partial u_6}\frac{\partial u_6}{\partial u_5} = \frac{1}{2}, \quad \frac{\partial f}{\partial u_4} = \frac{\partial f}{\partial u_5}\frac{\partial u_5}{\partial u_4} = u_4 = u_3 - y = \cdots = \sigma(wx+b) - y,$$

$$\frac{\partial f}{\partial u_3} = \frac{\partial f}{\partial u_4}\frac{\partial u_4}{\partial u_3} = \sigma(wx+b) - y, \quad \frac{\partial f}{\partial u_2} = \frac{\partial f}{\partial u_3}\frac{\partial u_3}{\partial u_2} = \frac{\partial f}{\partial u_3}\sigma'(u_2) = \sigma'(wx+b)\big(\sigma(wx+b) - y\big),$$

$$\frac{\partial f}{\partial u_1} = \frac{\partial f}{\partial u_2}\frac{\partial u_2}{\partial u_1} = \sigma'(wx+b)\big(\sigma(wx+b) - y\big), \quad \frac{\partial f}{\partial b} = \frac{\partial f}{\partial u_2}\frac{\partial u_2}{\partial b} = \sigma'(wx+b)\big(\sigma(wx+b) - y\big),$$

$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial u_1}\frac{\partial u_1}{\partial w} + \frac{\partial f}{\partial u_7}\frac{\partial u_7}{\partial w} = x\sigma'(wx+b)\big(\sigma(wx+b) - y\big) + w.$$

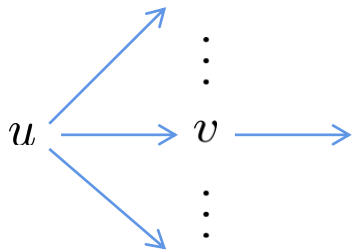
w 有两个子节点 u_1 和 u_7，故 $\frac{\partial f}{\partial w} = \frac{\partial f}{\partial u_1}\frac{\partial u_1}{\partial w} + \frac{\partial f}{\partial u_7}\frac{\partial u_7}{\partial w}$



4.6.2 反向传播—计算图使用

- 根据链式法则，有

$$\frac{\partial f}{\partial u} = \sum_{v \in S(u)} \frac{\partial f}{\partial v} \frac{\partial v}{\partial u}$$



- 对每个节点u，在构建计算图时，u的子节点总是在u之后构建，因此在记录的节点构造顺序里，u总是排在它的子节点的前面，从而使得在反向传播时，总是先遍历完u的所有子节点再遍历u

- 在计算$\frac{\partial f}{\partial u}$时，所有需要的$\frac{\partial f}{\partial v}$都已经计算完毕，而$\frac{\partial v}{\partial u}$可以使用已编写的求导法则直接计算，因此反向传播总是可以执行下去

4.6.3 基于计算图的反向传播

算法：基于计算图的反向传播

输入：基于计算图构建过程记录的节点创建顺序：$u_1, u_2, \cdots, u_n$

输出：grad_table，记录 f 关于每个节点的梯度，其中 $\frac{\partial f}{\partial u_n} = 1$

for $k = n-1, \cdots, 1$

对当前节点 u_k，读取其子节点集合 $\mathrm{S}(u_k)$

grad = 0;

for each $v \in \mathrm{S}(u_k)$

从 grad_table 读取 $\frac{\partial f}{\partial v}$

grad = grad + $\frac{\partial f}{\partial v}\frac{\partial v}{\partial u_k}$ # 链式法则 $\frac{\partial f}{\partial u} = \sum_{v \in S(u)} \frac{\partial f}{\partial v}\frac{\partial v}{\partial u}$

end

将 grad 作为 $\frac{\partial f}{\partial u_k}$ 记录在 grad_table 中 # 以后使用时直接读取，无须重复计算

end for

4.6.4* 全连接网络反向传播—向量形式

- 研究生教学内容，本科生拓展内容
- 前向传播

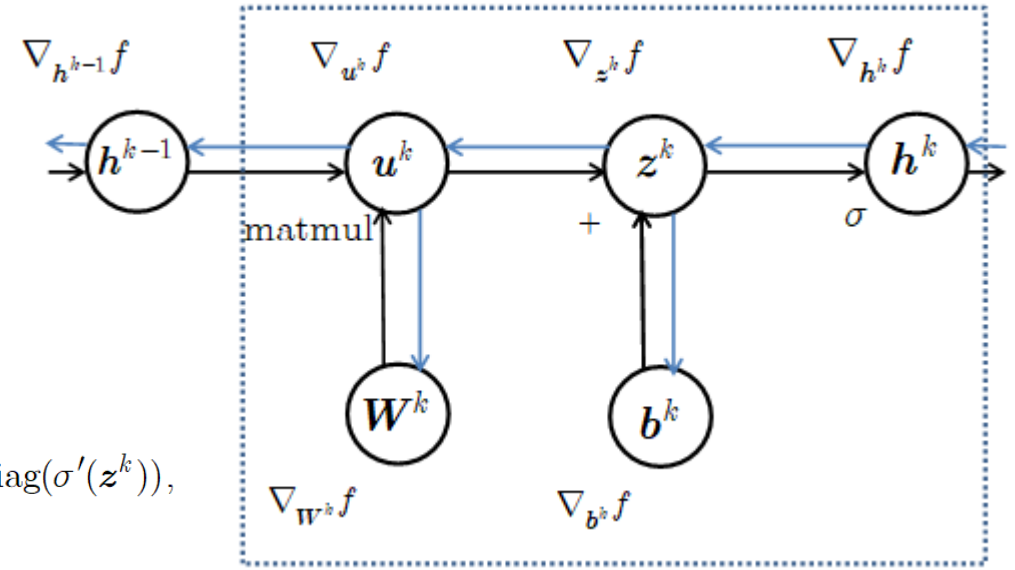
$$
\begin{aligned}
&\boldsymbol{h}^0 = \boldsymbol{x} \\
&\boldsymbol{z}^k = \boldsymbol{W}^k \boldsymbol{h}^{k-1} + \boldsymbol{b}^k, \quad \boldsymbol{h}^k = \sigma(\boldsymbol{z}^k), \quad k = 1, \cdots, L \\
&l = f(\boldsymbol{h}^L)
\end{aligned}
$$

为了区分网络层数和向量及矩阵的元素索引，把网络层数索引放到右上角，向量及矩阵的元素索引放到右下角

4.6.4* 全连接网络反向传播—向量形式

- 前向传播：

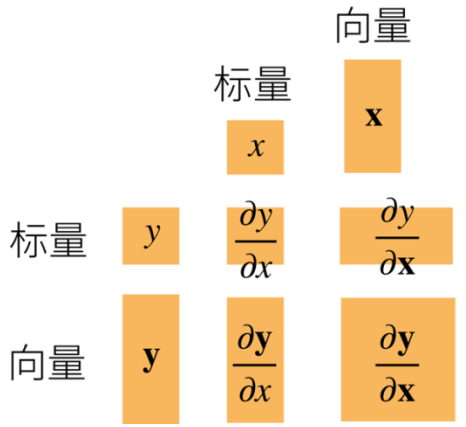
$$\boldsymbol{u}^k = \boldsymbol{W}^k \boldsymbol{h}^{k-1}, \quad \boldsymbol{z}^k = \boldsymbol{u}^k + \boldsymbol{b}^k, \quad \boldsymbol{h}^k = \sigma(\boldsymbol{z}^k)$$



- 对每个计算单元，输出值对每个输入的导数：

$$\frac{\partial \boldsymbol{u}^k}{\partial \boldsymbol{h}^{k-1}} = \boldsymbol{W}^k, \quad \frac{\partial \boldsymbol{u}_i^k}{\partial \boldsymbol{W}_i^k} = \left(\boldsymbol{h}^{k-1}\right)^T, \quad \frac{\partial \boldsymbol{z}^k}{\partial \boldsymbol{u}^k} = \boldsymbol{I}, \quad \frac{\partial \boldsymbol{z}^k}{\partial \boldsymbol{b}^k} = \boldsymbol{I}, \quad \frac{\partial \boldsymbol{h}^k}{\partial \boldsymbol{z}^k} = \mathrm{diag}(\sigma'(\boldsymbol{z}^k)),$$

其中W_i^k表示W的第i行元素构成的列向量，$\mathrm{diag}(x)$表示对角线元素为向量x的对角矩阵



4.6.4* 全连接网络反向传播—向量形式

- 回顾：引理 1：假设 $\boldsymbol{x} \in \mathbb{R}^n$，$\boldsymbol{y} \in \mathbb{R}^m$，$g: \mathbb{R}^n \to \mathbb{R}^m$，$f: \mathbb{R}^m \to \mathbb{R}$，$\boldsymbol{y} = g(\boldsymbol{x})$，$z = f(\boldsymbol{y})$，则有

$$\underbrace{\nabla_{\boldsymbol{x}} z}_{\mathbb{R}^n} = \underbrace{\left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^T}_{\mathbb{R}^{n\times m}} \underbrace{\nabla_{\boldsymbol{y}} z}_{\mathbb{R}^m}$$

（即 $\dfrac{\partial z}{\partial x} = \dfrac{\partial z}{\partial y}\dfrac{\partial y}{\partial x}$）

- 不妨设 $\boldsymbol{h}^{k-1} \in \mathbb{R}^n$，$\boldsymbol{u}^k, \boldsymbol{b}^k, \boldsymbol{z}^k, \boldsymbol{h}^k \in \mathbb{R}^m$，$\boldsymbol{W}^k \in \mathbb{R}^{m\times n}$

$$\nabla_{\boldsymbol{z}^k} l = \left(\frac{\partial \boldsymbol{h}^k}{\partial \boldsymbol{z}^k}\right)^T \nabla_{\boldsymbol{h}^k} l = \nabla_{\boldsymbol{h}^k} l \odot \sigma'(\boldsymbol{z}^k),$$

$$\nabla_{\boldsymbol{u}^k} l = \left(\frac{\partial \boldsymbol{z}^k}{\partial \boldsymbol{u}^k}\right)^T \nabla_{\boldsymbol{z}^k} l = \nabla_{\boldsymbol{h}^k} l \odot \sigma'(\boldsymbol{z}^k),$$

$$\nabla_{\boldsymbol{b}^k} l = \left(\frac{\partial \boldsymbol{z}^k}{\partial \boldsymbol{b}^k}\right)^T \nabla_{\boldsymbol{z}^k} l = \nabla_{\boldsymbol{h}^k} l \odot \sigma'(\boldsymbol{z}^k),$$

$$\underbrace{\nabla_{\boldsymbol{h}^{k-1}} l}_{\mathbb{R}^n} = \left(\frac{\partial \boldsymbol{u}^k}{\partial \boldsymbol{h}^{k-1}}\right)^T \nabla_{\boldsymbol{u}^k} l = \underbrace{\left(\boldsymbol{W}_k\right)^T}_{\mathbb{R}^{n\times m}} \underbrace{\nabla_{\boldsymbol{u}^k} l}_{\mathbb{R}^m} = (\boldsymbol{W}_k)^T \left(\nabla_{\boldsymbol{h}^k} l \odot \sigma'(\boldsymbol{z}^k)\right),$$

$$\nabla_{\boldsymbol{W}_i^k} l = \left(\frac{\partial \boldsymbol{u}_i^k}{\partial \boldsymbol{W}_i^k}\right)^T \nabla_{\boldsymbol{u}_i^k} l = \boldsymbol{h}^{k-1} \nabla_{\boldsymbol{u}_i^k} l \Rightarrow \nabla_{(\boldsymbol{W}^k)^T} l = \left(\boldsymbol{h}^{k-1}\nabla_{\boldsymbol{u}_1^k} l \quad \cdots \quad \boldsymbol{h}^{k-1}\nabla_{\boldsymbol{u}_m^k} l\right) = \boldsymbol{h}^{k-1}\left(\nabla_{\boldsymbol{u}^k} l\right)^T$$

$$\Rightarrow \underbrace{\nabla_{\boldsymbol{W}^k} l}_{\mathbb{R}^{m\times n}} = \underbrace{\left(\nabla_{\boldsymbol{u}^k} l\right)}_{\mathbb{R}^m} \underbrace{\left(\boldsymbol{h}^{k-1}\right)^T}_{\mathbb{R}^{1\times n}} = \left(\nabla_{\boldsymbol{h}^k} l \odot \sigma'(\boldsymbol{z}^k)\right)\left(\boldsymbol{h}^{k-1}\right)^T,$$

$$\boldsymbol{u}^k = \boldsymbol{W}^k \boldsymbol{h}^{k-1}, \qquad \boldsymbol{z}^k = \boldsymbol{u}^k + \boldsymbol{b}^k, \qquad \boldsymbol{h}^k = \sigma(\boldsymbol{z}^k)$$

$$\frac{\partial \boldsymbol{u}^k}{\partial \boldsymbol{h}^{k-1}} = \boldsymbol{W}^k, \qquad \frac{\partial \boldsymbol{u}_i^k}{\partial \boldsymbol{W}_i^k} = \left(\boldsymbol{h}^{k-1}\right)^T, \qquad \frac{\partial \boldsymbol{z}^k}{\partial \boldsymbol{u}^k} = \boldsymbol{I},$$

$$\frac{\partial \boldsymbol{z}^k}{\partial \boldsymbol{b}^k} = \boldsymbol{I}, \qquad \frac{\partial \boldsymbol{h}^k}{\partial \boldsymbol{z}^k} = \operatorname{diag}(\sigma'(\boldsymbol{z}^k)),$$

其中 $\odot$ 表示 Hadamard 乘积 $\begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \odot \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ \vdots \\ a_m b_m \end{bmatrix}$

4.6.4* 全连接网络反向传播—向量形式

算法：向量形式的反向传播

初始化：$\nabla_{\boldsymbol{h}^L} l = \nabla_{\boldsymbol{h}^L} f$

for $k = L, L-1, \cdots, 2, 1$

$$\nabla_{\boldsymbol{W}^k} l = \left(\nabla_{\boldsymbol{h}^k} l \odot \sigma'(\boldsymbol{z}^k)\right)\left(\boldsymbol{h}^{k-1}\right)^T$$

$$\nabla_{\boldsymbol{b}^k} l = \nabla_{\boldsymbol{h}^k} l \odot \sigma'(\boldsymbol{z}^k)$$

$$\nabla_{\boldsymbol{h}^{k-1}} l = (\boldsymbol{W}_k)^T \left(\nabla_{\boldsymbol{h}^k} l \odot \sigma'(\boldsymbol{z}^k)\right)$$

end for

4.6.4* 全连接网络反向传播—向量形式

- 交叉熵损失函数求导：$\nabla_h f$
  - 将交叉熵损失重写如下：

$$\begin{aligned}\ell(y, \hat{y}) &= -\sum_{j=1}^{q} y_j \log \frac{\exp(h_j)}{\sum_{i=1}^{q} \exp(h_i)} \\ &= -\sum_{j=1}^{q} y_j \left(h_j - \log \left(\sum_{i=1}^{q} \exp(h_i) \right) \right) \\ &= \log \left(\sum_{i=1}^{q} \exp(h_i) \right) \sum_{j=1}^{q} y_j - \sum_{j=1}^{q} y_j h_j \\ &= \log \left(\sum_{i=1}^{q} \exp(h_i) \right) - \sum_{j=1}^{q} y_j h_j\end{aligned}$$

  - 对 h 求导

$$\begin{aligned}\nabla_{h_j} \ell &= \frac{\exp(h_j)}{\sum_{i=1}^{q} \exp(h_i)} - y_j = \text{softmax}(h)_j - y_j \\ \nabla_h \ell &= \text{softmax}(h) - y\end{aligned}$$

- 回顾：二次损失函数求导：

$$\ell(y, h) = \frac{1}{2}(h - y)^2, \quad \nabla_h \ell = h - y$$

- 二者在形式上类似

总结

- 介绍前向全连接网络的反向传播算法
- 介绍了针对任意网络结构的计算图上的BP算法