微积分

微积分

- 导数、微分、积分
- 链式求导法则
- 反向传播算法

导数

◆ 导数的概念

导数是数学中的一个基本概念，主要用于研究函数随变量变化的快慢，即变化率。给出一个函数 f(x)，如果我们想要计算在某一点 x = a 处的导数，我们会考虑函数在 x 接近 a 时的变化情况。
更正式地说，函数 f(x) 在点 x = a 处的导数定义为：

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

如果这个极限存在，我们就说函数在点 a 是可导的

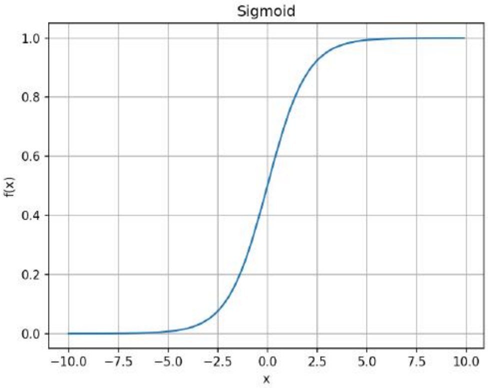
导数

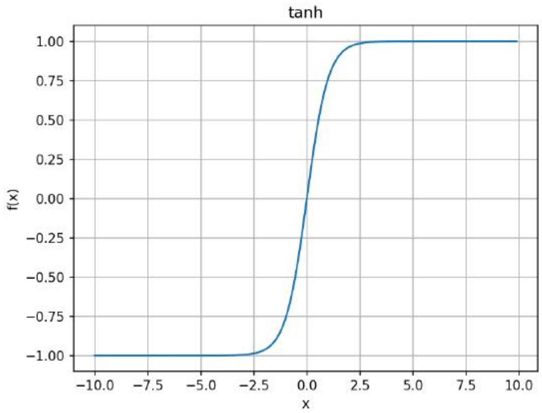
◆ 常见导数

初等函数	导数
x^n	$n \cdot x^{n-1}$
e^x	e^x
a^x	$a^x \ln(a)$
$\ln(x)$	$\frac{1}{x}$
$\log_a(x)$	$\frac{1}{x \ln(a)}$
$\sin(x)$	$\cos(x)$
$\cos(x)$	$-\sin(x)$
$\tan(x)$	$\sec^2(x)$
$\arcsin(x)$	$\frac{1}{\sqrt{1-x^2}}$
$\arccos(x)$	$-\frac{1}{\sqrt{1-x^2}}$
$\arctan(x)$	$\frac{1}{1+x^2}$

激活函数求导

◆ 常见激活函数



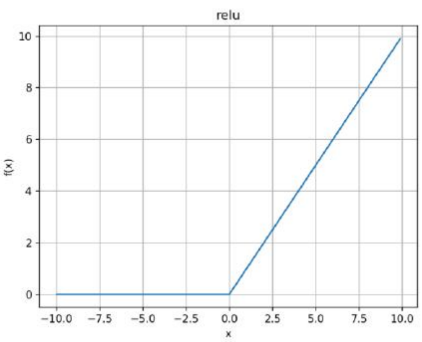


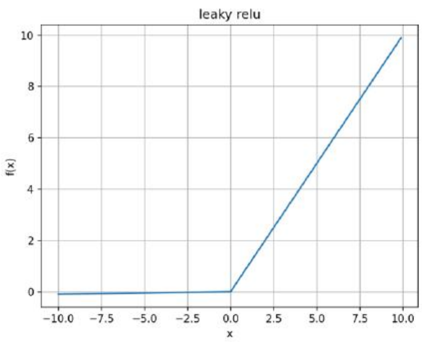
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

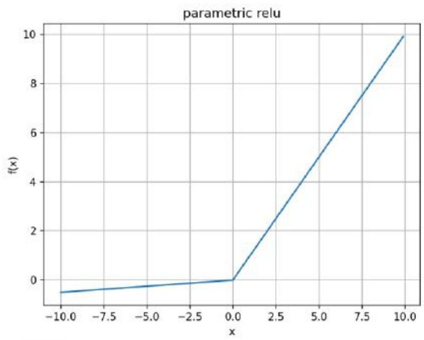
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

激活函数求导

◆ 常见激活函数







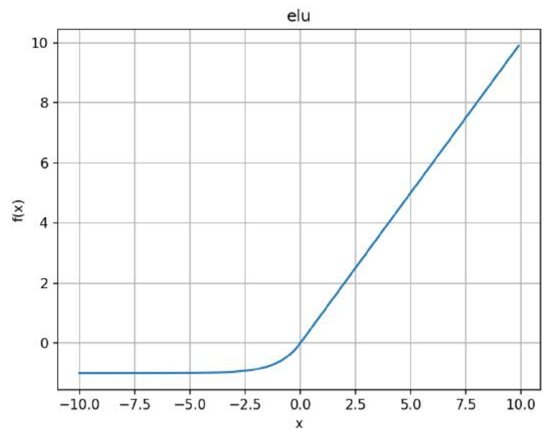
$$\mathrm{ReLU}(x) = \max(0, x)$$

$$\mathrm{LeakyReLU}(x) = \max(0.01x, x)$$

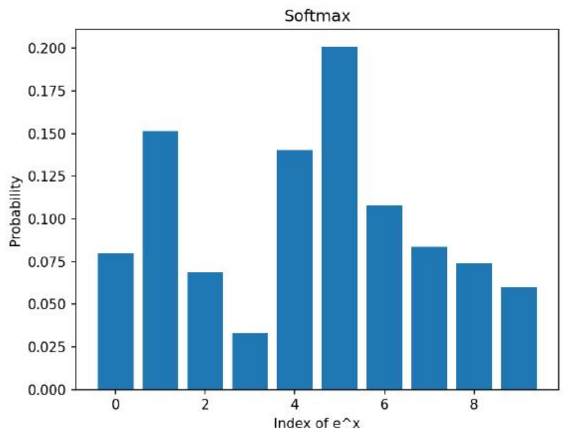
$$\mathrm{PReLU}(x) = \max(\alpha x, x)$$

激活函数求导

◆ 常见激活函数



$$\mathrm{ELU}(x)=\begin{cases}x, & \text{if } x>0\\ \alpha(e^x-1), & \text{if } x\le 0\end{cases}$$



$$\mathrm{Softmax}(x_i)=\frac{e^{x_i}}{\sum_j e^{x_j}}$$

激活函数求导

◆ 常见激活函数的导数

激活函数	表达式	导数	特点
Sigmoid	$\sigma(x)=\frac{1}{1+e^{-x}}$	$\sigma'(x)=\sigma(x)(1-\sigma(x))$	将输入压缩到(0,1)区间，用于二分类问题，在深层网络中容易产生梯度消失
Tanh	$\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$	$\tanh'(x)=1-\tanh^2(x)$	将输入压缩到(-1,1)区间，比sigmoid有更大的输出范围，也可能导致梯度消失
ReLU	$\mathrm{ReLU}(x)=\max(0,x)$	$\mathrm{ReLU}'(x)=\begin{cases}1, & \text{if } x>0\\ 0, & \text{if } x\le 0\end{cases}$	在正区间内保持梯度不变，解决了梯度消失问题，但在负区间梯度为0
Leaky ReLU	$\mathrm{LeakyReLU}(x)=\max(0.01x,x)$	$\mathrm{LeakyReLU}'(x)=\begin{cases}1, & \text{if } x>0\\ 0.01, & \text{if } x\le 0\end{cases}$	解决了ReLU在负区间梯度为0的问题，允许小梯度存在
Parametric ReLU	$\mathrm{PReLU}(x)=\max(\alpha x,x)$	$\mathrm{PReLU}'(x)=\begin{cases}1, & \text{if } x>0\\ \alpha, & \text{if } x\le 0\end{cases}$	ReLU的泛化形式，其中α是一个可学习的参数
ELU	$\mathrm{ELU}(x)=\begin{cases}x, & \text{if } x>0\\ \alpha(e^x-1), & \text{if } x\le 0\end{cases}$	$\mathrm{ELU}'(x)=\begin{cases}1, & \text{if } x>0\\ \mathrm{ELU}(x)+\alpha, & \text{if } x\le 0\end{cases}$	试图结合sigmoid和ReLU的优点，输出对负值有小幅度的响应

案例实践

◆ 使用Python绘制出上述激活函数

微分

◆ 微分的概念

函数 f(x) 在点 x 的微分，通常表示为 df 或 df(x) ，在几何上对应于函数图像在 x 点的切线上， x 的一个小变化 dx 所引起的函数值 f(x) 的变化。这个变化称为函数 f 的微分，数学上可以写作

$$df(x) = f'(x)dx ,$$

其中 f'(x) 是函数 f 在 x 处的导数。

微分

◆ 加法法则

如果有两个函数 u(x) 和 v(x)， 它们的和为 f(x) = u(x) + v(x)， 那么 f(x) 的导数是这两个函数导数的和：

$$\frac{d}{dx}[u(x)+v(x)]=\frac{d}{dx}u(x)+\frac{d}{dx}v(x)$$

微分

◆ 乘法法则

如果有两个函数 u(x) 和 v(x)， 它们的乘积为 f(x) = u(x) * v(x)， 那么 f(x) 的导数是：

$$\frac{d}{dx}[u(x)\cdot v(x)]=u(x)\frac{d}{dx}v(x)+v(x)\frac{d}{dx}u(x)$$

微分

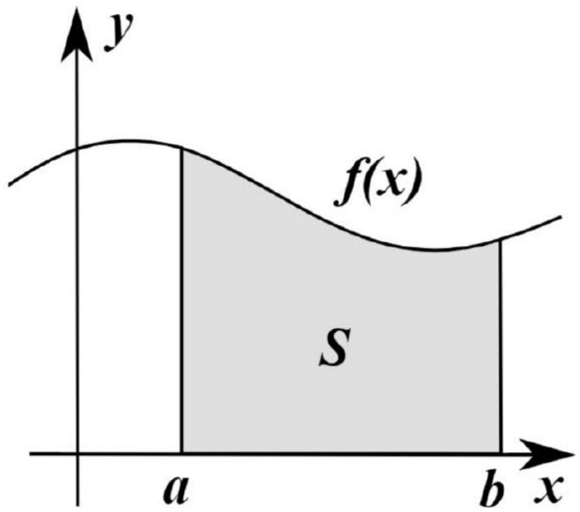
◆ 除法法则

如果有两个函数 u(x) 和 v(x)，它们的商为 $f(x)=\frac{u(x)}{v(x)}$，那么 f(x) 的导数是：

$$\frac{d}{dx}\left[\frac{u(x)}{v(x)}\right]=\frac{v(x)\frac{d}{dx}u(x)-u(x)\frac{d}{dx}v(x)}{[v(x)]^2}$$

积分

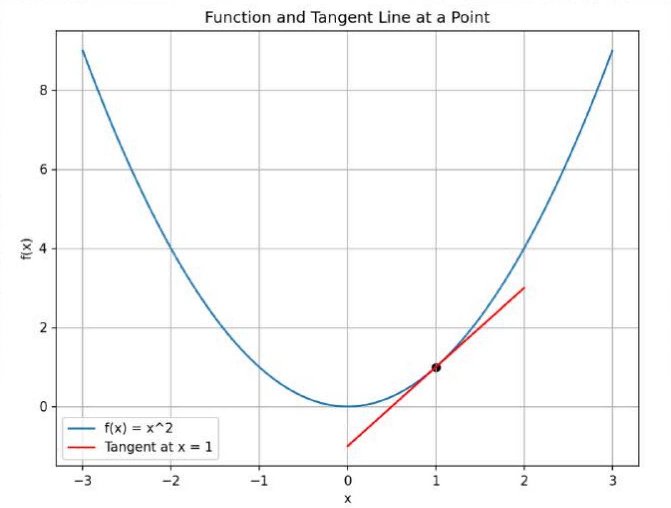
◆ 积分示意图



$$\int_a^b f(x)\,dx$$

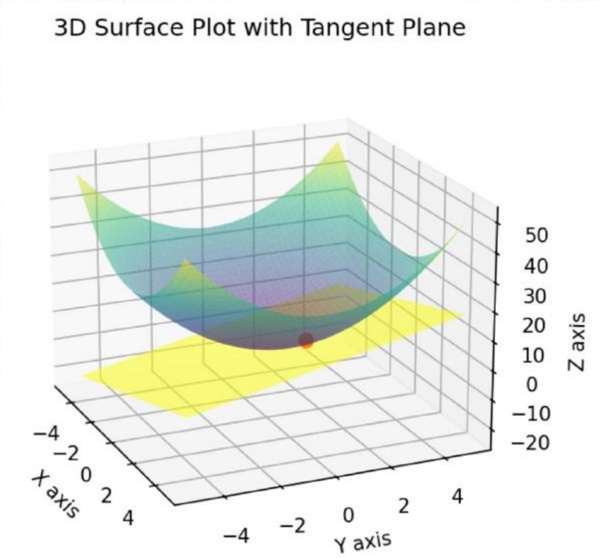
案例展示

◆ 展示二维图像切线



案例展示

◆ 展示三维函数切面



链式求导法则

◆ 链式求导法则

设有两个函数 y = f(u) 和 u = g(x)，它们都在相应的点可导。如果我们定义一个复合函数 y = f(g(x))，那么复合函数 y 关于 x 的导数是：

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

这里，$\frac{dy}{du}$ 是外函数 f(u) 关于其变量 u 的导数，而 $\frac{du}{dx}$ 是内函数 g(x) 关于 x 的导数。

链式求导法则

◆ 链式求导法则

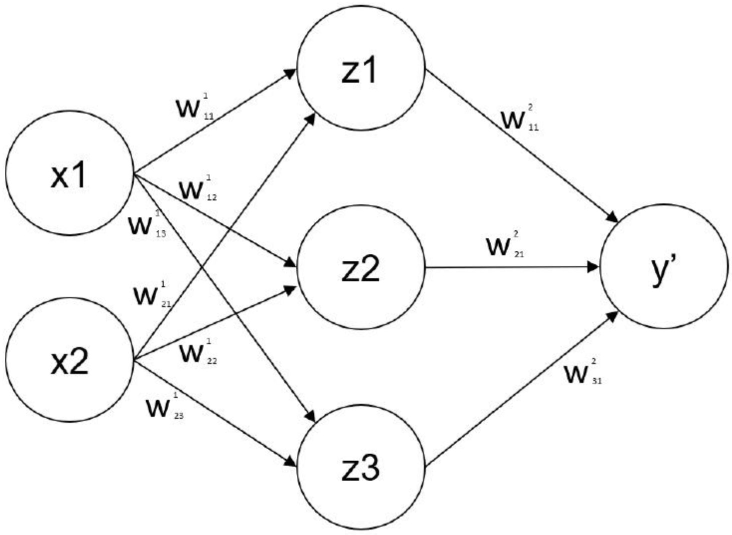
假设有复合函数 $h(x) = (3x+2)^2$ ，对他进行求导

令 $f(u) = u^2$ ， $g(x) = 3x + 2$

1. 首先求 $g(x)$ 的导数：$g'(x) = 3$。
2. 然后求 $f(u)$ 对 u 的导数：$f'(u) = 2u$。
3. 将 $g(x)$ 代入 $f'(u)$ 得到 $f'(g(x)) = 2(3x+2)$。
4. 最后，将 $f'(g(x))$ 和 $g'(x)$ 相乘得到 $h'(x)$：$h'(x) = 2(3x+2) \cdot 3$。

反向传播算法

◆ 求反向传播



假设：

第一层参数全部初始化为0.5
第二层参数全部初始化为1

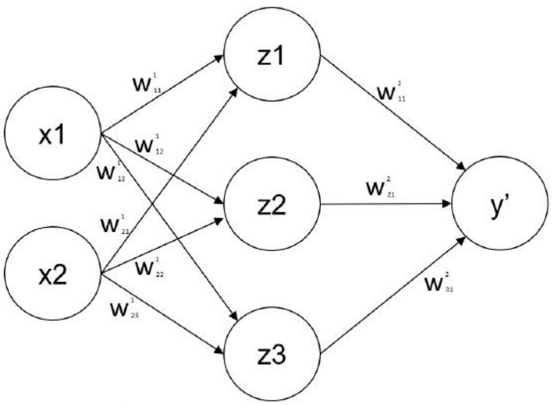
输入x1=40,x2=80,目标输出y=60

学习率为1e-5

反向传播算法

◆ 前向传播

第一层参数全部初始化为0.5
第二层参数全部初始化为1
输入x1=40,x2=80,目标y=60
学习率为1e-5



$$z_1 = w_{11}^1 x_1 + w_{21}^1 x_2 = 0.5\times40 + 0.5\times80 = 60$$

$$z_2 = w_{12}^1 x_1 + w_{22}^1 x_2 = 0.5\times40 + 0.5\times80 = 60$$

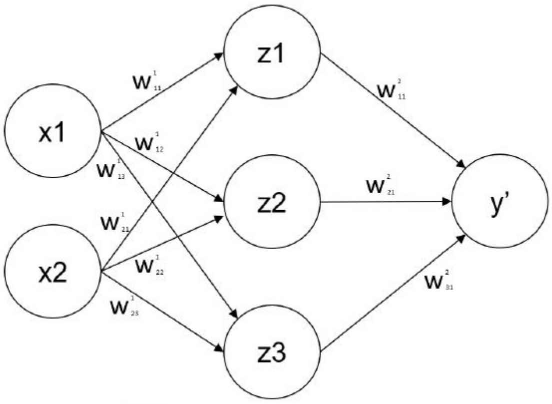
$$z_3 = w_{13}^1 x_1 + w_{23}^1 x_2 = 0.5\times40 + 0.5\times80 = 60$$

$$y' = w_{11}^2 z_1 + w_{21}^2 z_2 + w_{31}^2 z_3 = 1\times60 + 1\times60 + 1\times60 = 180$$

反向传播算法

◆ 计算误差

第一层参数全部初始化为0.5
第二层参数全部初始化为1
输入x1=40,x2=80,目标y=60
学习率为1e-5



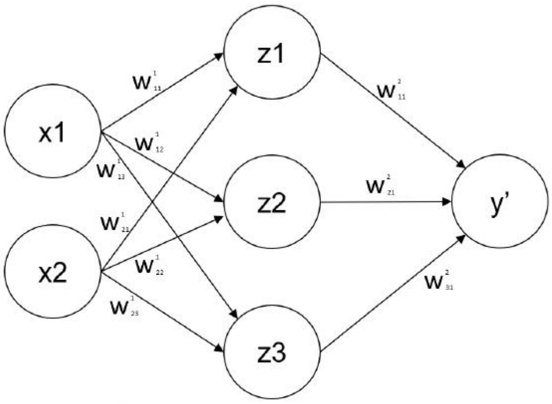
$$
\begin{aligned}
y' &= W_{11}^{2} Z_1 + W_{21}^{2} Z_2 + W_{31}^{2} Z_3 \\
&= 1\times 60 + 1\times 60 + 1\times 60 \\
&= 180
\end{aligned}
$$

$$
\begin{aligned}
Loss &= \frac{1}{2}(y-y')^2 \\
&= \frac{1}{2}(60-180)^2 \\
&= 7200
\end{aligned}
$$

反向传播算法

◆ 反向传播：计算损失函数的梯度

第一层参数全部初始化为0.5
第二层参数全部初始化为1
输入x1=40,x2=80,目标y=60
学习率为1e-5



$$
\begin{aligned}
\frac{\partial L}{\partial y'} &= \frac{1}{2}\cdot 2\cdot (y-y')\cdot(-1) \\
&= y'-y \\
&= 180-60 \\
&= 120
\end{aligned}
$$

反向传播算法

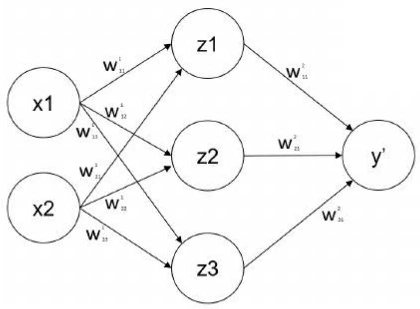
◆ 反向传播：计算第2层的梯度

第一层参数全部初始化为0.5
第二层参数全部初始化为1
输入x1=40,x2=80,目标y=60
学习率为1e-5

$$\frac{\partial L}{\partial W_{11}^2}=\frac{\partial L}{\partial y'}\cdot\frac{\partial y'}{\partial W_{11}^2}=120\times Z_1=120\times 60=7200$$

$$\frac{\partial L}{\partial W_{21}^2}=\frac{\partial L}{\partial y'}\frac{\partial y'}{\partial W_{21}^2}=120\times Z_2=120\times 60=7200$$

$$\frac{\partial L}{\partial W_{31}^2}=\frac{\partial L}{\partial y'}\cdot\frac{\partial y}{\partial W_{31}}=120\times Z_3=120\times 60=7200$$



反向传播算法

◆ 反向传播：计算第1层的梯度

第一层参数全部初始化为0.5
第二层参数全部初始化为1
输入x1=40,x2=80,目标y=60
学习率为1e-5

$$\frac{\partial L}{\partial W_{11}^1}=\frac{\partial L}{\partial y'}\frac{\partial y'}{\partial Z_1}\frac{\partial Z_1}{\partial W_{11}^1}=120\times W_{11}^2\times X_1=120\times 1\times 40=4800$$

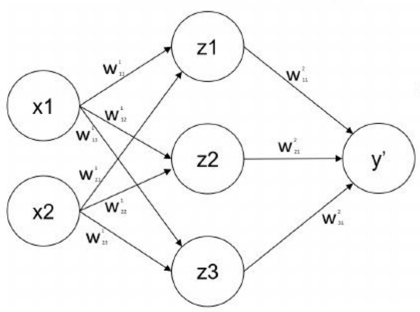
$$\frac{\partial L}{\partial W_{12}^1}=\frac{\partial L}{\partial y'}\frac{\partial y'}{\partial Z_2}\frac{\partial Z_2}{\partial W_{12}^1}=120\times W_{21}^2\times X_1=120\times 1\times 40=4800$$

$$\frac{\partial L}{\partial W_{13}^1}=\frac{\partial L}{\partial y'}\frac{\partial y'}{\partial Z_3}\frac{\partial Z_3}{\partial W_{13}^1}=120\times W_{31}^2\times X_1=120\times 1\times 40=4800$$

$$\frac{\partial L}{\partial W_{21}^1}=\frac{\partial L}{\partial y'}\frac{\partial y'}{\partial Z_1}\frac{\partial Z_1}{\partial W_{21}^1}=120\times W_{11}^2\times X_2=120\times 1\times 80=9600$$

$$\frac{\partial L}{\partial W_{22}^1}=\frac{\partial L}{\partial y'}\frac{\partial y'}{\partial Z_2}\frac{\partial Z_2}{\partial W_{22}^1}=120\times W_{21}^2\times X_2=120\times 1\times 80=9600$$

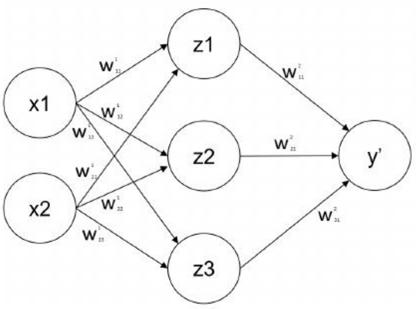
$$\frac{\partial L}{\partial W_{23}^1}=\frac{\partial L}{\partial y'}\frac{\partial y'}{\partial Z_3}\frac{\partial Z_3}{\partial W_{23}^1}=120\times W_{31}^2\times X_2=120\times 1\times 80=9600$$



反向传播算法

◆ 梯度更新

第一层参数全部初始化为0.5
第二层参数全部初始化为1
输入x1=40,x2=80,目标y=60
学习率为1e-5



$$W = W - \eta \frac{\partial L}{\partial W}$$

$$W_{11}^2 = W_{11}^2 - \eta \frac{\partial L}{\partial W_{11}^2} = 1 - 10^{-5} \times 7200 = 1 - 0.072 = 0.928$$

$$W_{21}^2 = W_{21}^2 - \eta \frac{\partial L}{\partial W_{21}^2} = 1 - 10^{-5} \times 7200 = 1 - 0.072 = 0.928$$

$$W_{31}^2 = W_{31}^2 - \eta \frac{\partial L}{\partial W_{31}^2} = 1 - 10^{-5} \times 7200 = 0.928$$

$$W_{11}^1 = W_{11}^1 - \eta \frac{\partial L}{\partial W_{11}^1} = 0.5 - 10^{-5} \times 4800 = 0.5 - 0.048 = 0.452$$

$$W_{12}^1 = W_{12}^1 - \eta \frac{\partial L}{\partial W_{12}^1} = 0.5 - 10^{-5} \times 4800 = 0.5 - 0.048 = 0.452$$

$$W_{13}^1 = W_{13}^1 - \eta \frac{\partial L}{\partial W_{13}^1} = 0.5 - 10^{-5} \times 4800 = 0.5 - 0.048 = 0.452$$

$$W_{21}^1 = W_{21}^1 - \eta \frac{\partial L}{\partial W_{21}^1} = 0.5 - 10^{-5} \times 9600 = 0.5 - 0.096 = 0.404$$

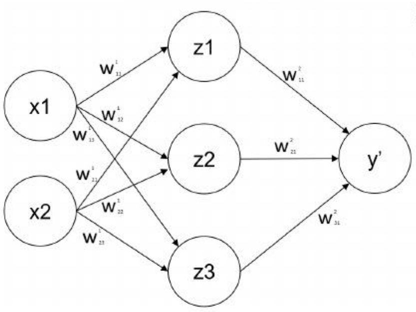
$$W_{22}^1 = W_{22}^1 - \eta \frac{\partial L}{\partial W_{22}^1} = 0.5 - 10^{-5} \times 9600 = 0.5 - 0.096 = 0.404$$

$$W_{23}^1 = W_{23}^1 - \eta \frac{\partial L}{\partial W_{23}^1} = 0.5 - 10^{-5} \times 9600 = 0.5 - 0.096 = 0.404$$

反向传播算法

◆ 重新进行前向传播

第一层参数全部初始化为0.5
第二层参数全部初始化为1
输入x1=40,x2=80,目标y=60
学习率为1e-5



$$Z_1 = W_{11}^1 x_1 + W_{21}^1 x_2 = 0.452 \times 40 + 0.404 \times 80 = 18.08 + 32.32 = 50.4$$

$$Z_2 = W_{12}^1 x_1 + W_{22}^1 x_2 = 0.452 \times 40 + 0.404 \times 80 = 50.4$$

$$Z_3 = W_{13}^1 x_1 + W_{23}^1 x_2 = 0.452 \times 40 + 0.404 \times 80 = 50.4$$

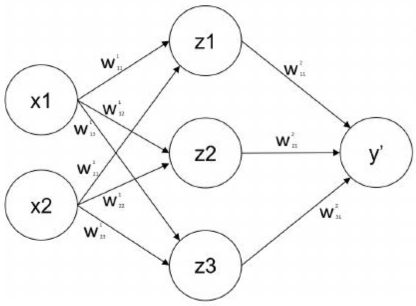
$$y' = W_{11}^2 Z_1 + W_{21}^2 Z_2 + W_{31}^2 Z_3 = 0.928 \times 50.4 \times 3 = 140.3136$$

反向传播算法

◆ 重新计算误差

第一层参数全部初始化为0.5
第二层参数全部初始化为1
输入x1=40,x2=80,目标y=60
学习率为1e-5

$$
\begin{aligned}
Loss &= \frac{1}{2}(y-y')^2 \\
&= \frac{1}{2}(60-140.3156)^2 \\
&\approx 3225.157 < 7200\text{(更新前)}
\end{aligned}
$$

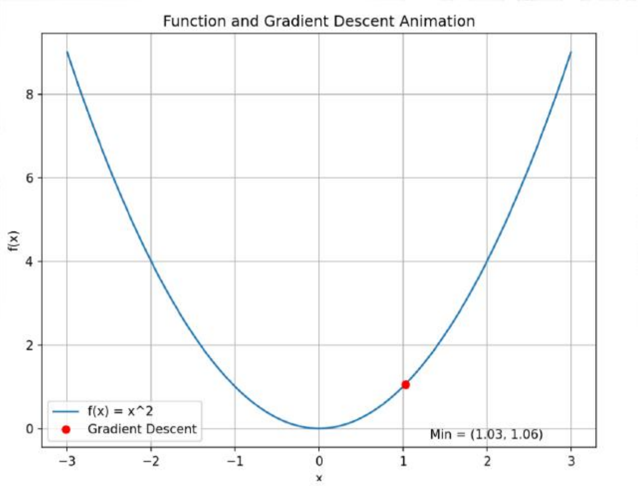


案例展示

代码实战：使用代码完成上述的神经网络反向传播

案例展示

代码实战：制作梯度下降求最小值的动画



案例展示

代码实战：实现三维平面的梯度下降