Deep Learning I

2. 预备知识

WU Xiaokun 吴晓堃

xkun.wu [at] gmail

2021/02/28

数据操作

N维数组

也称为**张量 tensor**

- `Numpy`的`ndarray`：仅支持CPU计算
- `PyTorch`和`TensorFlow`中`Tensor`：支持自动微分

N维数组

也称为**张量 tensor**

- `Numpy`的`ndarray`：仅支持CPU计算
- `PyTorch`和`TensorFlow`中`Tensor`：支持自动微分

难点：

- 理解维度、形状与索引
- 理解张量的构造
    - 增减维度
    - 拼接

几个常见特例

0-d（标量）

```
1.0
```

- 类别

1-d（向量）

```
[1.0, 2.7, 3.4]
```

- 特征向量

2-d（矩阵）

```
[
    [1.0, 2.7,
3.4]
    [5.0, 0.2,
4.6]
    [4.3, 8.5,
0.2]
]
```

- 样本-特征矩阵

几个应用特例

3-d

```
[[[0.1, 2.7,
3.4]
 [5.0, 0.2, 4.6]
 [4.3, 8.5,
0.2]]
 [[3.2, 5.7,
3.4]
 [5.4, 6.2, 3.2]
 [4.1, 3.5,
```

- RGB图片（通道x宽x高）

4-d

```
[[[[. . .
 . . .
 . . .]]]]
```

- 批量RGB图片（批量x通道x宽x高）

5-d

```
[[[[[. . .
 . . .
 . . .]]]]]
```

- 批量视频（批量x时间x通道x宽x高）

实验： N维数组

数据预处理

pandas软件包

Pandas (Python Data Analysis Library)

- Python 的**核心数据分析支持库**，提供了快速、灵活、明确的数据结构，旨在简单、直观地处理关系型、标记型数据。

实验：pandas软件包

线性代数

几何意义：向量

两种解释

- 位置：坐标系上，与原点的相对距离
  - 标准正交基的**线性组合**
- 方向和长度

几何意义：向量

两种解释

- 位置：坐标系上，与原点的相对距离
  - 标准正交基的**线性组合**
- 方向和长度

向量加法：连接起点、终点

几何意义：向量

两种解释

- 位置：坐标系上，与原点的相对距离
  - 标准正交基的**线性组合**
- 方向和长度

向量加法：连接起点、终点

向量内积：度量**相似度**

- 投影后相乘：$a \cdot b = |a|(|b| \cos\theta)$

矩阵乘法

行列维数需要一致

矩阵乘法

行列维数需要一致

注意：Python中*默认是**按元素乘法**

- 数学上的Hadamard积：$\odot$

几何意义：矩阵

矩阵乘法：**空间线性变形**

- 矩阵乘以矩阵可以看成矩阵乘以多个列向量
  - 特例：标准正交基，即坐标系

几何意义：矩阵

矩阵乘法：**空间线性变形**

- 矩阵乘以矩阵可以看成矩阵乘以多个列向量
  - 特例：标准正交基，即坐标系
- 仿射变换

$$\begin{bmatrix} m_{00} & m_{01} & t_x \\ m_{10} & m_{11} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

矩阵的特征向量

不被此矩阵改变方向

$$Ax = \lambda x$$

实验：线性代数

Halftime

多元微分

标量导数

一元微分（标量对标量）

- 基本函数：

y	a	x^n	$\exp(x)$	$\log(x)$	$\sin(x)$
$\frac{dy}{dx}$	0	nx^{n-1}	$\exp(x)$	$\frac{1}{x}$	$\cos(x)$

- 复合函数：

y	$u+v$	uv	$y=f(u), u=g(x)$
$\frac{dy}{dx}$	$\frac{du}{dx}+\frac{dv}{dx}$	$\frac{du}{dx}v+\frac{dv}{dx}u$	$\frac{dy}{du}\frac{du}{dx}$

梯度

多元微分（**标量对向量**）

$$df = \sum_i \frac{\partial f}{\partial x_i} dx_i = \frac{\partial f}{\partial x} d\mathbf{x}$$

- 全微分公式；梯度向量与微分向量的内积

梯度

多元微分（**标量对向量**）

$$df = \sum_i \frac{\partial f}{\partial x_i} dx_i = \frac{\partial f}{\partial x} d\mathbf{x}$$

- 全微分公式；梯度向量与微分向量的内积

- 微分：$d\mathbf{x} = [dx_1, dx_2, .., dx_n]^T$
- （偏）导数：$\frac{\partial}{\partial \mathbf{x}} = [\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, .., \frac{\partial}{\partial x_n}]$
  - “梯度横着走”

雅可比矩阵 Jacobian matrix

多元微分（向量对向量）

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

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

特例

多元微分（**向量对标量**）

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

- $\frac{\partial y}{\partial \mathbf{x}}$是行向量，$\frac{\partial \mathbf{y}}{\partial x}$是列向量
  - 只是其中一种约定方式

规律

	$x:(1,)$	$\mathbf{x}:(n,1)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$

- x逆序排
- 先排y，再排x

规律

	$x:(1,)$	$\mathbf{x}:(n,1)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$

- x逆序排
- 先排y，再排x

聪明的数学家非常善于提取、总结客观规律

- 微分学被系统地搭建起来
- 高度抽象的概念需要大量的**标记符**来概括描述
  - 于是本来没有的**问题被创造出来**
- 更聪明的物理学家想出了简化标记

拓展到矩阵

	$x:(1,)$	$\mathbf{x}:(n,1)$	$\mathbf{X}:(n,k)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$	$\frac{\partial y}{\partial \mathbf{X}}:(k,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}:(m,k,n)$
$\mathbf{Y}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial x}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}:(m,l,n)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}:(m,l,k,n)$

拓展到矩阵

	$x:(1,)$	$\mathbf{x}:(n,1)$	$\mathbf{X}:(n,k)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$	$\frac{\partial y}{\partial \mathbf{X}}:(k,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}:(m,k,n)$
$\mathbf{Y}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial x}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}:(m,l,n)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}:(m,l,k,n)$

思考：矩阵是二维张量，拓展到N维张量?

实验：微分计算

自动微分

链式法则

$$y = f(u), u = g(x) \Rightarrow \frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

链式法则

$$y = f(u), u = g(x) \Rightarrow \frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

手算举例：

$$z = (\langle \mathbf{x}, \mathbf{w} \rangle - y)^2, \frac{\partial z}{\partial \mathbf{w}} = ?$$

$$\begin{aligned} a &= \langle \mathbf{x}, \mathbf{w} \rangle \\ b &= a - y \\ z &= b^2 \end{aligned}$$

$$\begin{aligned} \frac{\partial z}{\partial \mathbf{w}} &= \frac{\partial z}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial \mathbf{w}} \\ &= 2b \cdot 1 \cdot \mathbf{x}^T \\ &= 2(\langle \mathbf{x}, \mathbf{w} \rangle - y)\mathbf{x}^T \end{aligned}$$

链式法则

$$y = f(u), u = g(x) \Rightarrow \frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$

手算举例:

$$z = (\langle \mathbf{x}, \mathbf{w} \rangle - y)^2, \frac{\partial z}{\partial \mathbf{w}} = ?$$

$$\begin{aligned} a &= \langle \mathbf{x}, \mathbf{w} \rangle \\ b &= a - y \\ z &= b^2 \end{aligned}$$

$$\begin{aligned} \frac{\partial z}{\partial \mathbf{w}} &= \frac{\partial z}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial \mathbf{w}} \\ &= 2b \cdot 1 \cdot \mathbf{x}^T \\ &= 2(\langle \mathbf{x}, \mathbf{w} \rangle - y)\mathbf{x}^T \end{aligned}$$

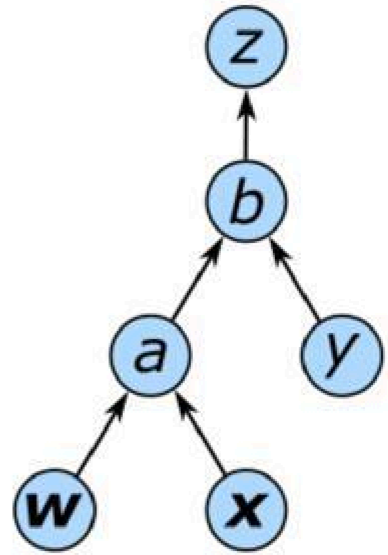
- 繁琐、易错；现在一般使用**自动微分**软件包

自动求导

构建计算图：将计算分解成算子的有向无环图

$$z = (\langle \mathbf{x}, \mathbf{w} \rangle - y)^2, \frac{\partial z}{\partial \mathbf{w}} = ?$$

$$\begin{aligned} a &= \langle \mathbf{x}, \mathbf{w} \rangle \\ b &= a - y \\ z &= b^2 \end{aligned}$$



自动求导的两种模式

链式法则：

$$\frac{\partial y}{\partial x}=\frac{\partial y}{\partial u_n}\frac{\partial u_n}{\partial u_{n-1}}\cdots\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial x}$$

- **前向积累 forward accumulation**：

$$\frac{\partial y}{\partial x}=\frac{\partial y}{\partial u_n}\left(\frac{\partial u_n}{\partial u_{n-1}}\left(\cdots\left(\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial x}\right)\right)\right)$$

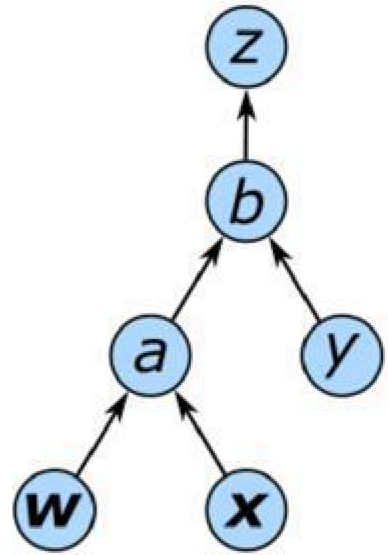
- **反向传递 backward propagation**：

$$\frac{\partial y}{\partial x}=\left(\left(\left(\frac{\partial y}{\partial u_n}\frac{\partial u_n}{\partial u_{n-1}}\right)\cdots\right)\frac{\partial u_2}{\partial u_1}\right)\frac{\partial u_1}{\partial x}$$

前向积累

前向：执行计算图，并存储中间结果

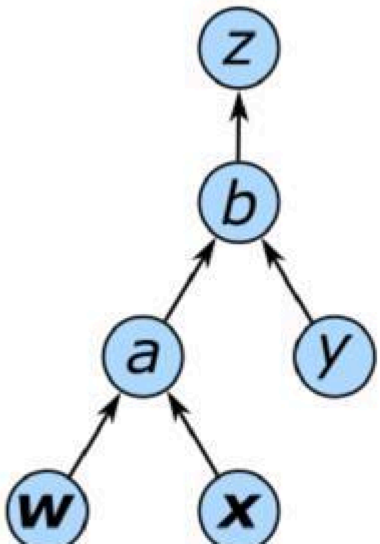
$$
\begin{aligned}
a &= \langle \mathbf{x}, \mathbf{w} \rangle \\
b &= a - y \\
z &= b^2
\end{aligned}
$$



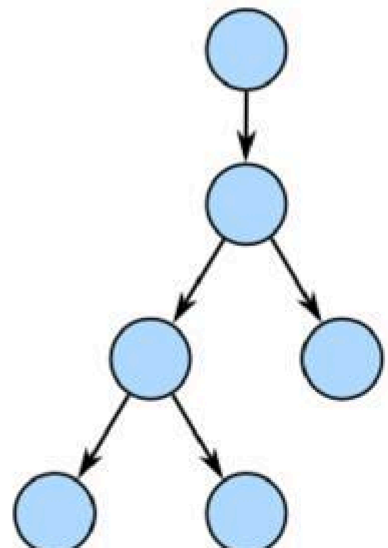
反向传递 I

反向：**计算梯度值**，剪除不需要的枝，避免重复计算。

$$z = b^2$$



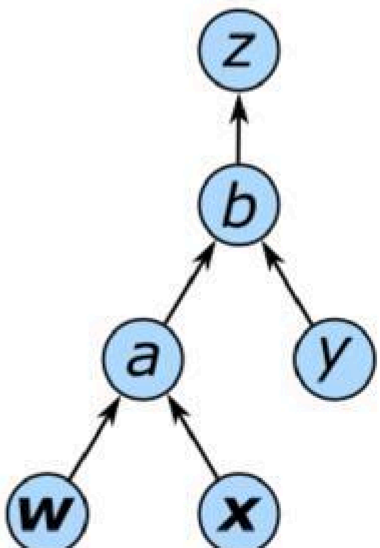
$$\frac{\partial z}{\partial b} = 2b$$



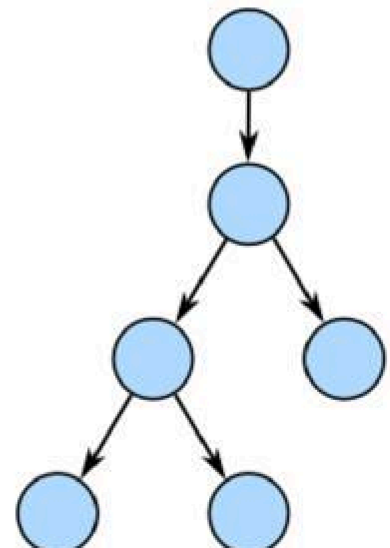
反向传递 II

反向：**计算梯度值**，剪除不需要的枝，避免重复计算。

$$b = a - y$$



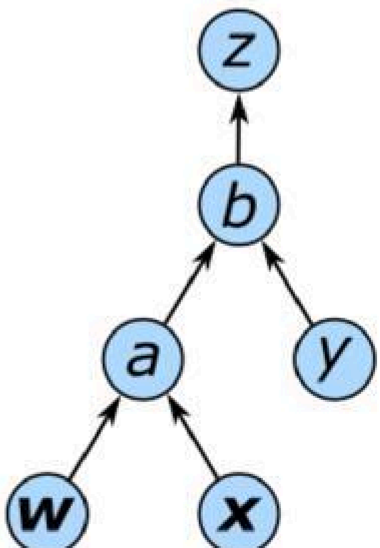
$$\frac{\partial b}{\partial a} = 1$$

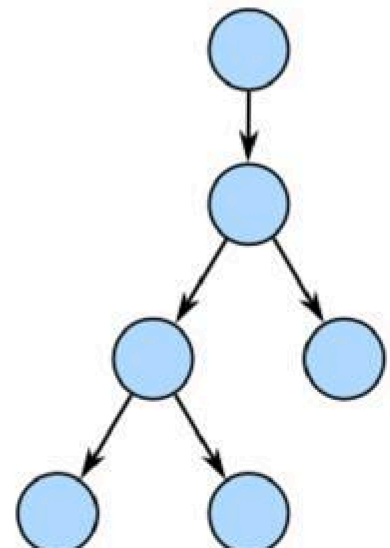


反向传递 III

反向：**计算梯度值，剪除不需要的枝，避免重复计算。**

$$a = \langle \mathbf{x}, \mathbf{w} \rangle \qquad \frac{\partial a}{\partial \mathbf{w}} = \mathbf{x}^T$$



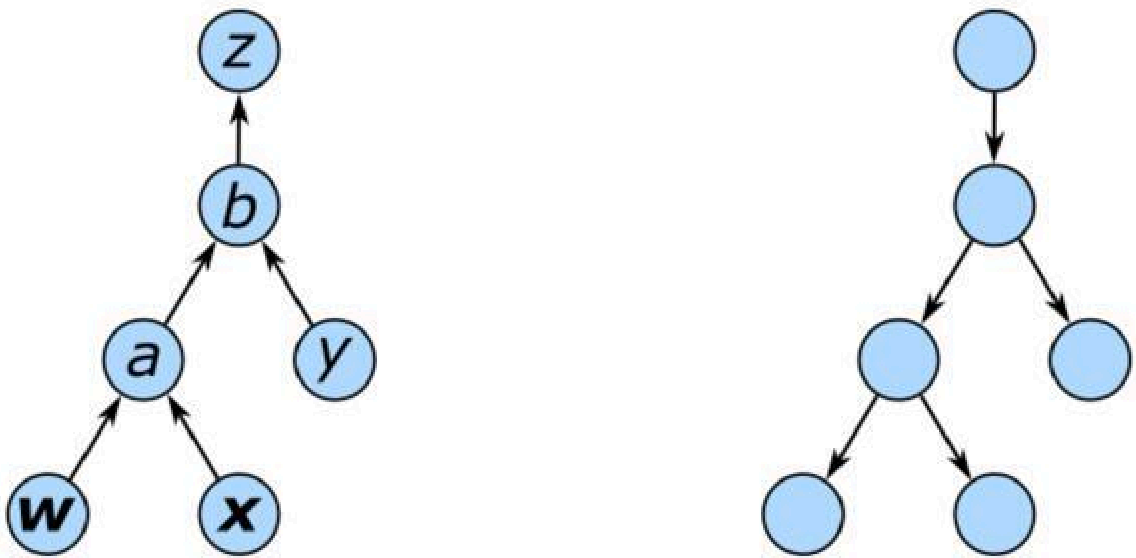


- 类比剪枝：剪除y对应的分支

为什么可以避免重复计算?

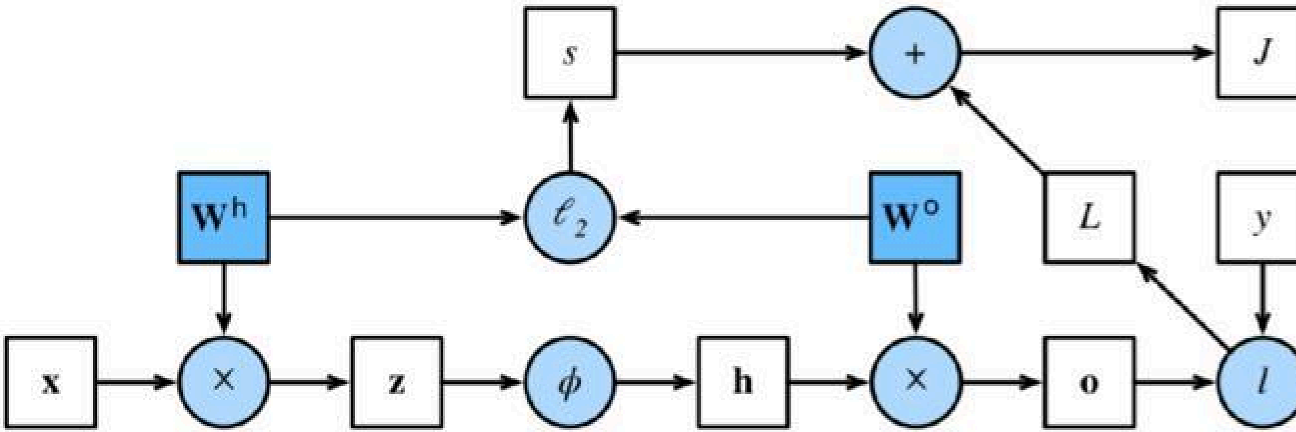
反向：**计算梯度值**，剪除不需要的枝，避免重复计算。

- 计算$\frac{\partial z}{\partial b}$需要$b$，进而需要$a$



- 类比：动态规划中的查表

实例：MLP 计算图



$$\mathbf{h} = \sigma(\mathbf{W}_h \mathbf{x} + \mathbf{b}_h)$$
$$\mathbf{o} = \mathbf{W}_o \mathbf{h} + \mathbf{b}_o$$
$$s = \frac{\lambda}{2}\left(\|\mathbf{W}_h\|_F^2 + \|\mathbf{W}_o\|_F^2\right)$$

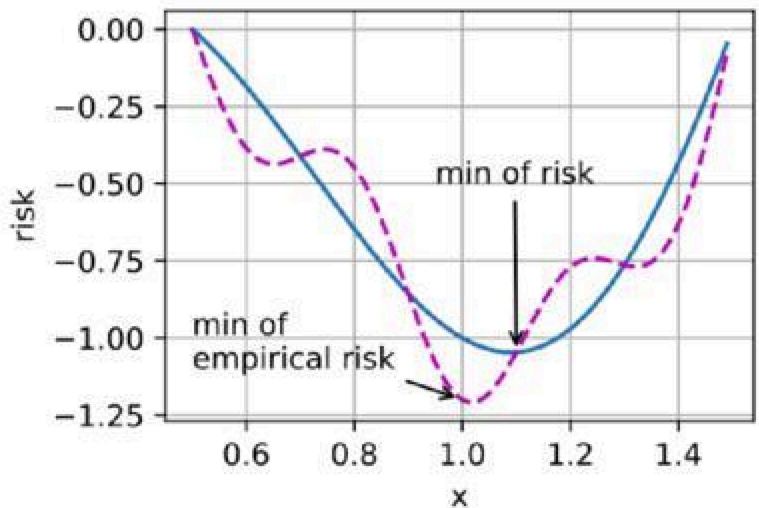
$$\mathbf{z} = \mathbf{W}_h \mathbf{x}$$
$$\mathbf{h} = \phi(\mathbf{z})$$
$$\mathbf{o} = \mathbf{W}_o \mathbf{h}$$
$$\mathcal{L} = l(O, y)$$

实验：自动微分

优化

优化、泛化

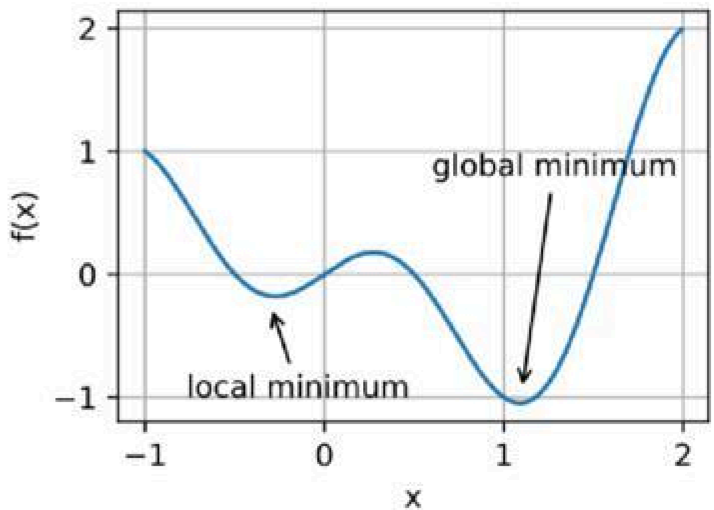
经验风险是训练数据集的平均损失；而**风险**则是整个数据群的预期损失。



局部最小、全局最小

深度学习模型的目标函数通常有许多**局部最优解**。

$$f(x) = x \cdot \cos(\pi x), -1.0 \leq x \leq 2.0$$



梯度下降

Taylor展开：函数的**一阶近似**

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

梯度下降

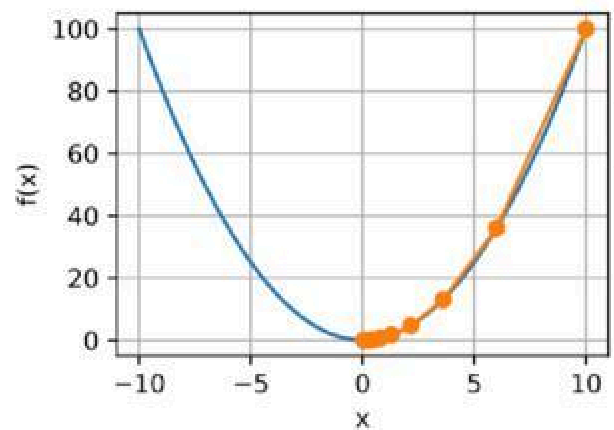
Taylor展开：函数的**一阶近似**

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

梯度：函数值**增长最快**的方向

- 向梯度的**反方向**走一小步

- $f(x - f'(x)\eta) \lessapprox f(x)$
- $x \leftarrow x - f'(x)\eta$



梯度下降

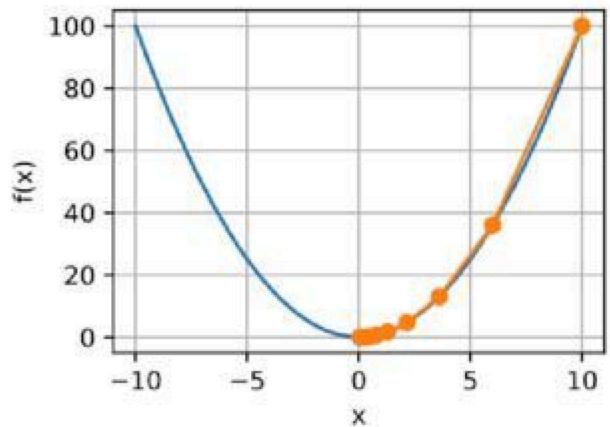
Taylor展开：函数的**一阶近似**

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

梯度：函数值**增长最快**的方向

- 向梯度的**反方向**走一小步

- $f(x - f'(x)\eta) \lessapprox f(x)$
- $x \leftarrow x - f'(x)\eta$

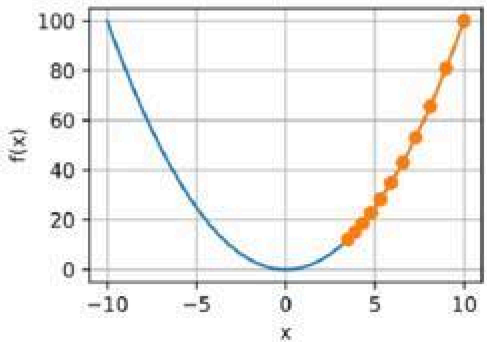


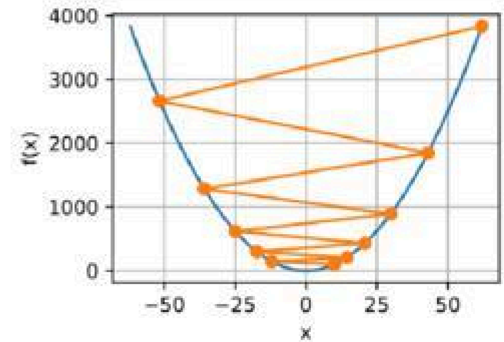
牛顿法：**二阶展开**，需要能够计算二阶导数（Hessian）

学习率

学习率 learning rate： η

- 太小：收敛慢
- 太大：发散

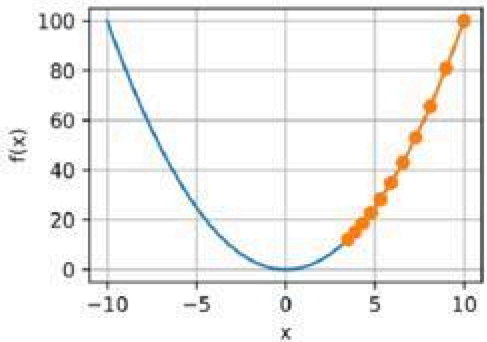


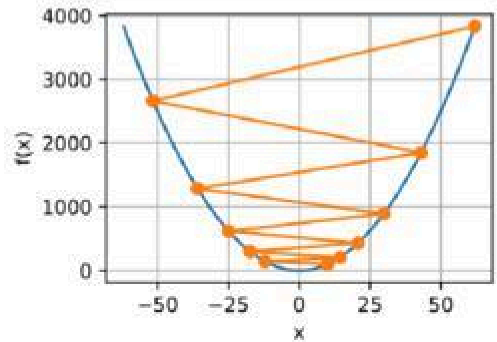


学习率

学习率 learning rate： η

- 太小：收敛慢
- 太大：发散



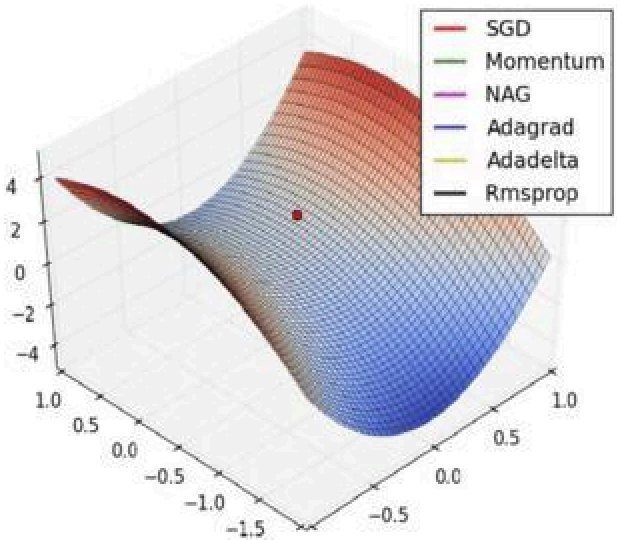


注意：地形平坦的区域步长自然变小

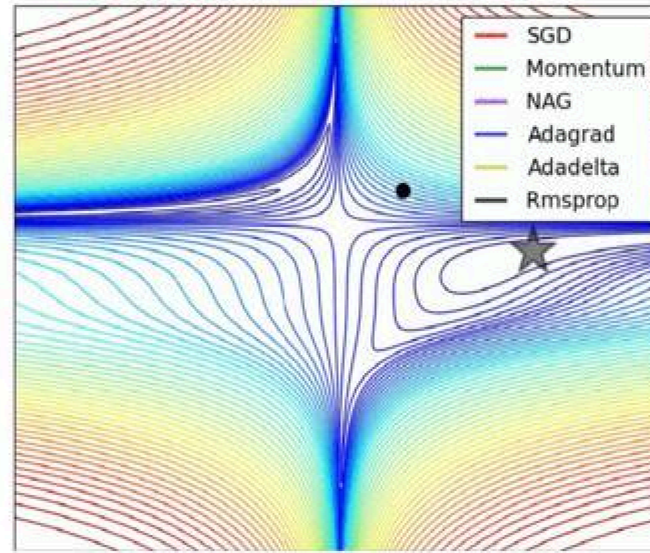
- $x \leftarrow x - f'(x)\eta$

带动量的梯度下降

摆脱鞍点：



摆脱多个极小值：



Review

本章内容

张量，张量运算。多元微分，自动求导。链式求导：前向积累与反向传播。基于梯度的优化。实验：二维仿射变换。实验：基本激活函数。实验：ReLU合成法构造一般函数。

重点：张量、张量运算、三种基本激活函数、深度学习层间运算的一般形式；

难点：张量运算的几何解释、反向传播算法、基于梯度的优化、ReLU合成法、一致逼近理论。

≡

学习目标

- 理解张量的概念；
- 掌握张量运算，并理解其几何解释；
- 理解基于梯度的优化及随机梯度下降算法；
- 理解反向传播算法的理论基础：链式求导法；
- 掌握三种基本激活函数：ReLU，Sigmoid，Tanh；
- 了解ReLU合成法构造连续函数的方法。

问题

(*) 列举5种基本的二维仿射变换，并应用于图片变形。

(*) 简述深度学习的几何解释。

(*) 用ReLU合成法拟合分段线性函数$(-10,-10),(0,0),(5,-5),(10,20)$。

(*) 假设$\mathbf{Y}$是(a,b,c,d)维张量，$\mathbf{X}$是(h,i,j,k,l)维张量，那么$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$的维度是多少?

画出$z=(\langle \mathbf{x},\mathbf{w}\rangle - y)^2$的计算图，并使用前向积累和反向传递计算$\frac{\partial z}{\partial \mathbf{w}}$。

简述梯度下降法的算法流程。

≡

概念

张量

数据的**容器**。0D、1D、2D张量又分别成为标量、向量、矩阵。

张量运算

数据不同表示之间的**变换函数**。

多元微分

雅可比矩阵 Jacobian matrix（向量对向量）

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

矩阵微分：

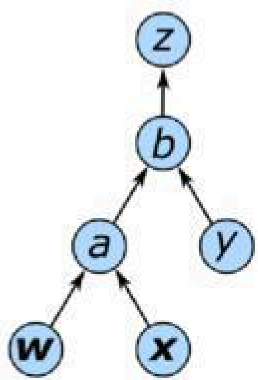
	$x:(1,)$	$\mathbf{x}:(n,1)$	$\mathbf{X}:(n,k)$
$y:(1,)$	$\frac{\partial y}{\partial x}:(1,)$	$\frac{\partial y}{\partial \mathbf{x}}:(1,n)$	$\frac{\partial y}{\partial \mathbf{X}}:(k,n)$
$\mathbf{y}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial x}:(m,1)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}:(m,n)$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}:(m,k,n)$
$\mathbf{Y}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial x}:(m,l)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}:(m,l,n)$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}:(m,l,k,n)$

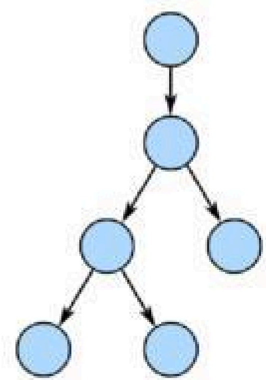
反向传播算法

链式求导

将链式法则应用于神经网络梯度值的计算，得到的算法叫作**反向传播算法**。

- 计算梯度值，剪除不需要的枝，避免重复计算。





梯度下降法

优化目标

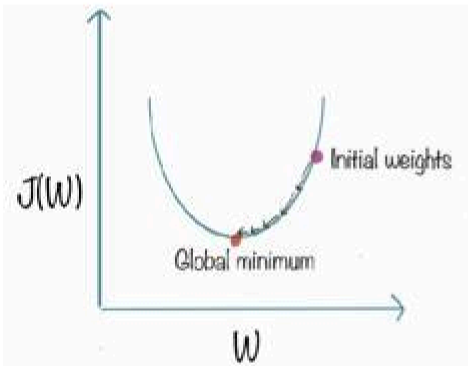
Objective: $\arg\min_{W,b} J$,

with: $J = \|\bar{y} - y\|, \bar{y} = \sum relu(\mathbf{W} * x + \mathbf{b})$

梯度下降更新

$W_1 = W_0 - \nabla J * s$

- 步长（学习率）要选取合适；
- 动量解决收敛速度、局部极值。



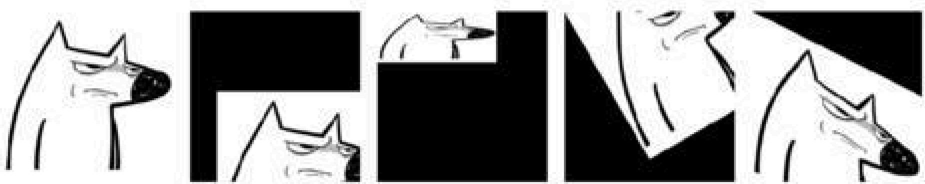
二维仿射变换

二维仿射变换的一般形式

$$\underline{GT} = \mathbf{W} * \underline{input} + \mathbf{b}$$

基本二维仿射变换矩阵

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & e_x & 0 \\ e_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



深度学习的层间运算

深度学习层间运算的一般形式

$$\underline{output} = activate(\underline{GT})$$

$$\underline{GT} = \mathbf{W} * \underline{input} + \mathbf{b}$$

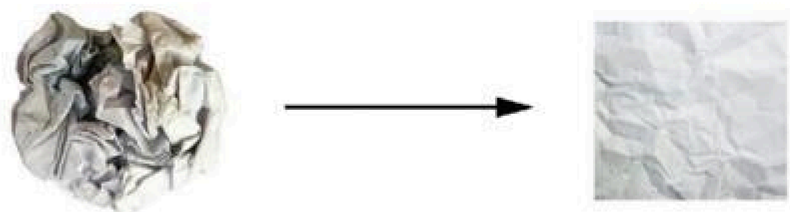
如何想象高维空间?

首先研究低维空间，归纳出规律，然后**将规律泛化到高维**。

深度学习的几何解释

深度学习可以解释为高维空间中非常复杂的几何变换

- 一切数据都是张量，即**几何空间中的点**。
- 模型的每一层对数据点做一个**几何变换**；而模型本身是这些变换的合成。
- 注意：几何变换必须可微。



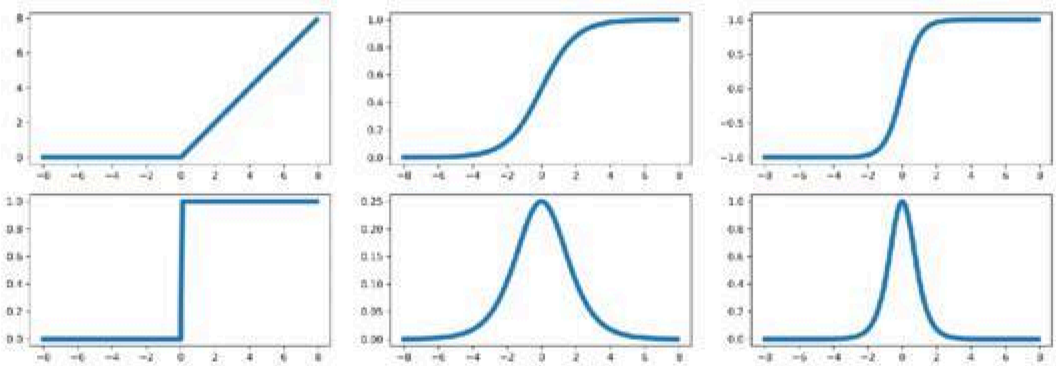
模型的可探索空间（假设空间）需要足够大

只要模型的参数足够多，就能捕捉到原始数据中所有的映射关系。想象“Ω路径”。

激活函数

三种基本激活函数

ReLU, sigmoid, tanh.



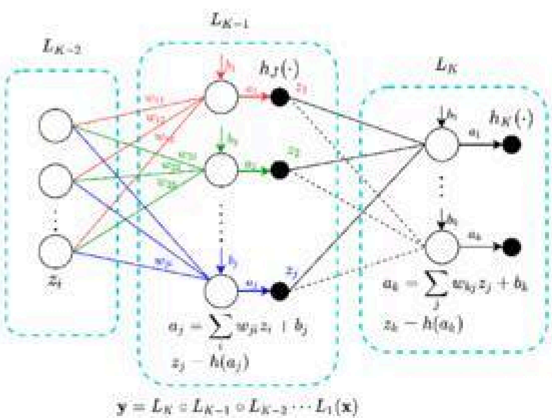
激活函数的作用

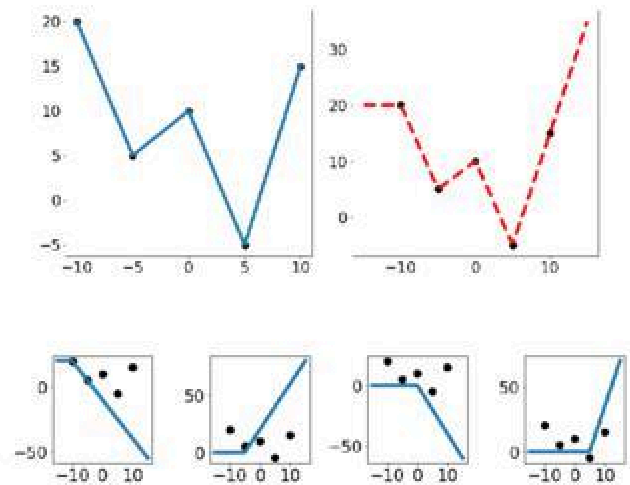
提供非线性。

ReLU合成法

ReLU 合成的一般形式

$$\sum relu(\mathbf{W} * \underline{input} + \mathbf{b})$$





一致逼近原理

Universal approximation theorem（非常有用的废话）

In approximation theory, both *shallow* and *deep* networks are known to **approximate any continuous functions** at an **exponential cost**.

构造方法：

- 构造线性函数；
- 构造分段线性函数；
- 构造离散化的任意函数。