Machine Learning I

1. 预备知识

WU Xiaokun 吴晓堃

xkun.wu [at] gmail

Contents

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**
  - 提示：想像成叠盘子
- 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,
6.2]]]
```

- 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

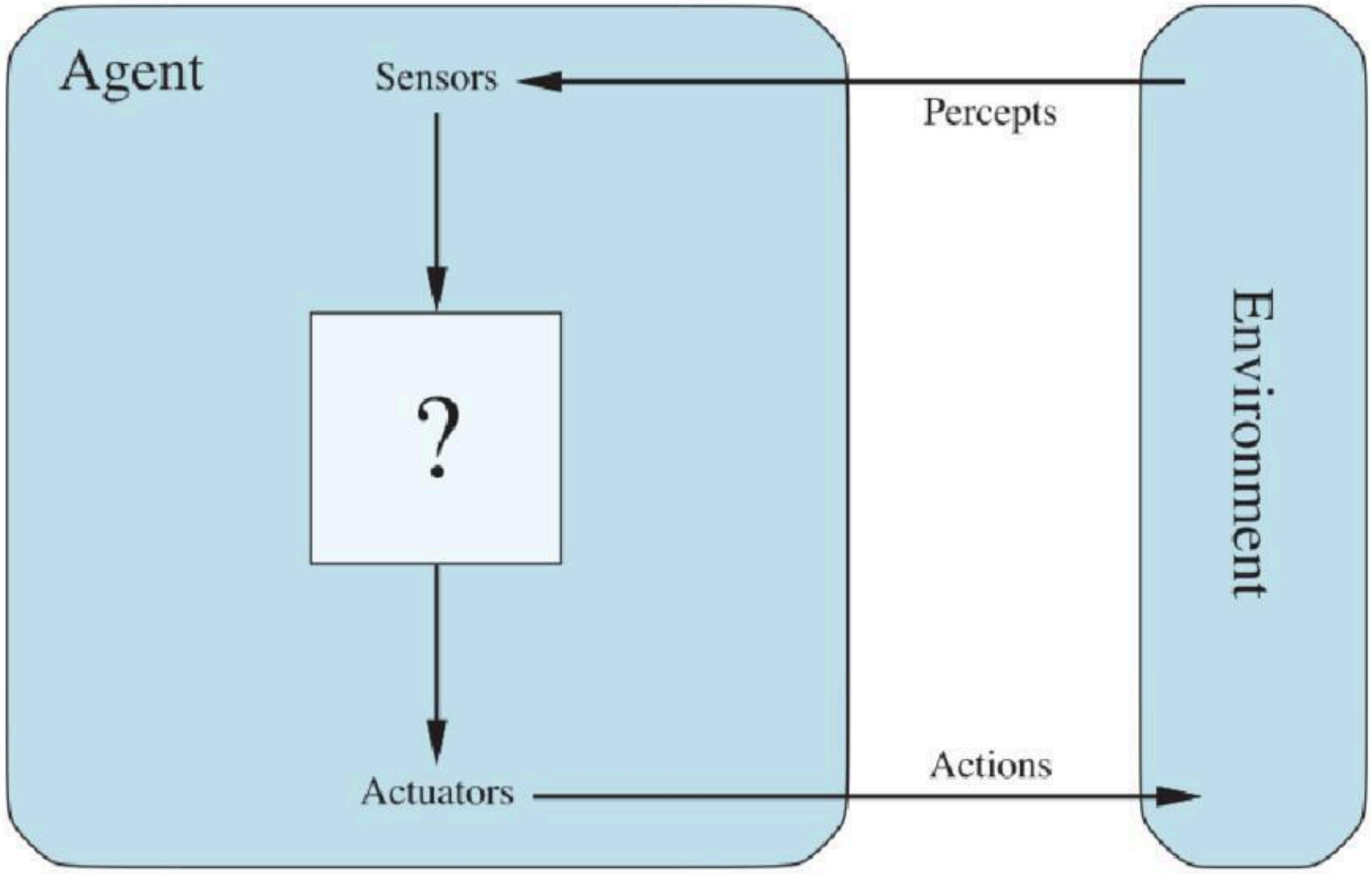
Pandas (Python Data Analysis Library)

- 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

学习：基本形式

回顾：智能体与环境



学习、机器学习

学习：智能体在观察世界获得经验后提高性能

学习、机器学习

学习：智能体在观察世界获得经验后提高性能

当智能体是计算机时，称为**机器学习**

- 计算机观察数据样本，并从数据中构造**模型**
- 模型既是对世界的**假设**，又是解决问题的软件

为什么需要机器去学习?

经典编程模式中，人是思考并解决问题的主体

- 难道程序员不能胜任实际工作吗?

为什么需要机器去学习?

经典编程模式中，人是思考并解决问题的主体

- 难道程序员不能胜任实际工作吗?

两个主要原因：

- 设计者很难预料到所有未知情况
- 有时设计者确实不知道怎么解决问题

归纳、演绎

从具体的观察实例中总结出规律的方法叫做**归纳法**

- 观察信号灯；太阳照常升起

归纳、演绎

从具体的观察实例中总结出规律的方法叫做**归纳法**

- 观察信号灯；太阳照常升起

注意：归纳结论可能会出错（不符合客观真理）

- 回顾：基于逻辑推演的**演绎法**必然得到正确结论，只要前提条件是正确的

输出：分类、回归

回顾：智能体建模基本形式

- 数据：数值型、结构型、关系型
- 模型：（一阶）逻辑、概率

输出：分类、回归

回顾：智能体建模基本形式

- 数据：数值型、结构型、关系型
- 模型：（一阶）逻辑、概率

按照输出数据格式：

- 输出有限数值：**分类**问题
- 输出连续数值：**回归**问题

对输入的反馈

监督学习：学习将输入映射到输出的函数

- 智能体能观察到：输入、输出数据对，其中输出通常称为**标签**
  - 与提供数据的智能体是“师生关系”
- 例如：行人检测，自动驾驶的制动

对输入的反馈

监督学习：学习将输入映射到输出的函数

- 智能体能观察到：输入、输出数据对，其中输出通常称为**标签**
  - 与提供数据的智能体是“师生关系”
- 例如：行人检测，自动驾驶的制动

无监督学习：学习数据的内在模式

- 智能体能观察到：混杂在一起的数据（杂烩）
  - 智能体无需对数据进行反馈
- **聚类**问题：相似物体归为一类

对输入的反馈

监督学习：学习将输入映射到输出的函数

- 智能体能观察到：输入、输出数据对，其中输出通常称为**标签**
  - 与提供数据的智能体是“师生关系”
- 例如：行人检测，自动驾驶的制动

无监督学习：学习数据的内在模式

- 智能体能观察到：混杂在一起的数据（杂烩）
  - 智能体无需对数据进行反馈
- **聚类**问题：相似物体归为一类

强化学习：学习应对外界刺激的行为

- 智能体能观察到：强化剂，如奖励、惩罚
- 例如象棋：确定哪一步是导致赢输的关键手

(*)监督学习概念

监督学习：数学定义

给定由 N 个输入、输出数据对组成的**训练（数据）集**

- $(x_1, y_1), (x_2, y_2), .., (x_N, y_N)$
- 每对数据由未知函数生成：$y = f(x)$

监督学习：数学定义

给定由 N 个输入、输出数据对组成的**训练（数据）集**

- $(x_1, y_1), (x_2, y_2), .., (x_N, y_N)$
- 每对数据由未知函数生成：$y = f(x)$

任务目标：找到真实函数 f 的**最佳近似函数** h

- h 称为**假设函数**：字面意即智能体对所在世界的假设

假设、假设空间

假设空间 $\mathcal{H}$：对当前问题考虑的所有可能假设

- 例如：三次多项式，所有Python程序
- 统计学术语，有时也称“**模型类**”，“**函数类**”

假设、假设空间

假设空间 $\mathcal{H}$：对当前问题考虑的所有可能假设

- 例如：三次多项式，所有Python程序
- 统计学术语，有时也称“**模型类**”，“**函数类**”

假设空间决定是否能找到最优解，如何选取?

- 有数据生成过程的**先验知识**? 直接建模
- 否则：**探索性数据分析**（如统计检测、可视化）获取对数据的感觉
- 或者：逐个测试，评测优劣

一致性假设、最佳拟合、泛化

一致性假设：对每个训练集里的数据点都预测正确，即$h(x_i) = y_i$

- 貌似很理想，但很难实现，且一般不是最优（从预测角度思考：为什么？）

最佳拟合：通常我们只需要 $h(x_i)$ 与 y_i 足够接近（思考：如何度量？）

一致性假设、最佳拟合、泛化

一致性假设：对每个训练集里的数据点都预测正确，即$h(x_i) = y_i$

- 貌似很理想，但很难实现，且一般不是最优（从预测角度思考：为什么？）

最佳拟合：通常我们只需要 $h(x_i)$ 与 y_i 足够接近（思考：如何度量？）

对假设的评测取决于其对**未知数据**的预测能力

- 训练集上的度量不是决定性因素：已知正确答案，无需判断
- 但未知数据意味着无法度量正确性?

一致性假设、最佳拟合、泛化

一致性假设：对每个训练集里的数据点都预测正确，即$h(x_i) = y_i$

- 貌似很理想，但很难实现，且一般不是最优（从预测角度思考：为什么？）

最佳拟合：通常我们只需要 $h(x_i)$ 与 y_i 足够接近（思考：如何度量？）

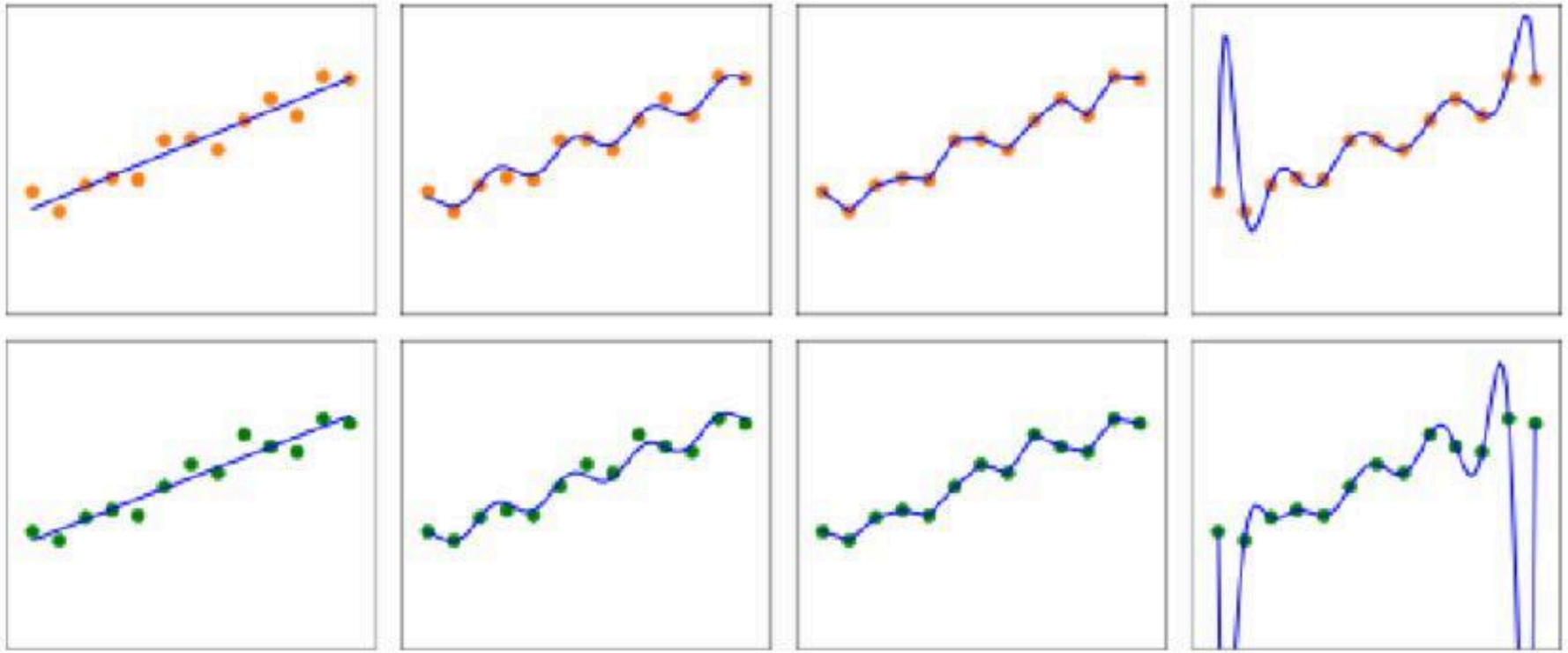
对假设的评测取决于其对**未知数据**的预测能力

- 训练集上的度量不是决定性因素：已知正确答案，无需判断
- 但未知数据意味着无法度量正确性?

模拟未知数据：专门划拨一组数据对做评测，称为**测试（数据）集**

- 假设的**泛化**能力强：在测试集上预测结果好

数据拟合实例：同分布采样两组13点



线性：$h(x) = w_1 x + w_0$

正弦：$h(x) = w_1 x + \sin(w_0 x)$

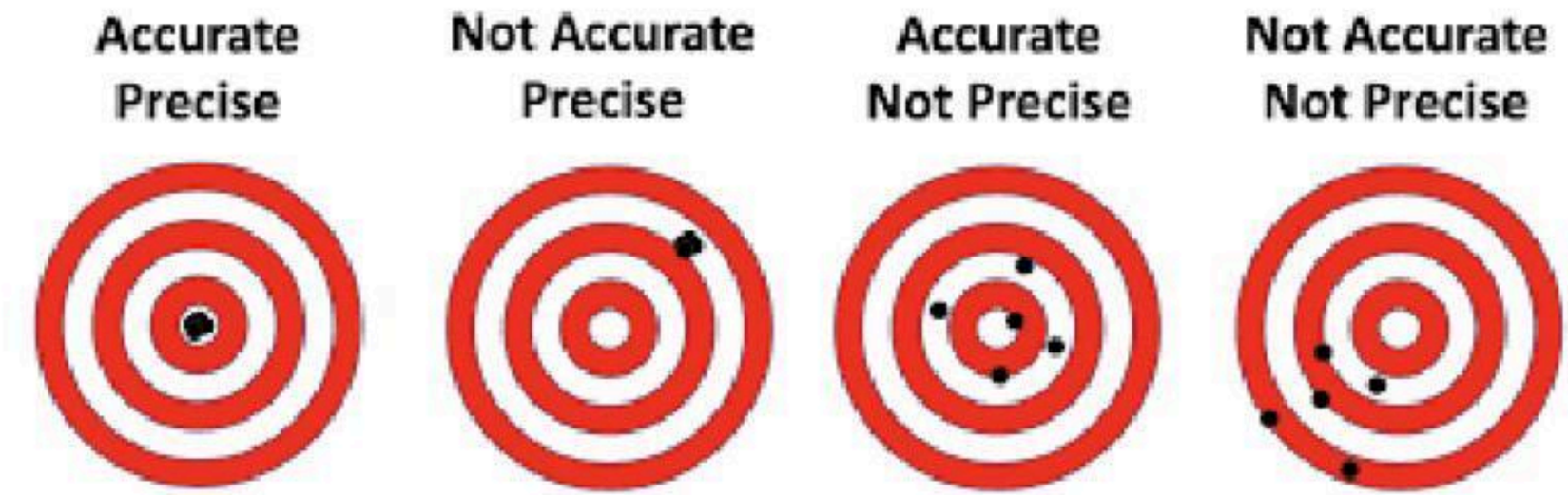
分段线性

12次多项式：$h(x) = \sum_{i=0}^{12} w_i x^i$

注意：由于随机性和噪音，同分布的两组采样数据在概率上不可能完全一样

偏差、方差

可以从两个统计角度对假设空间进行分析：偏差、方差

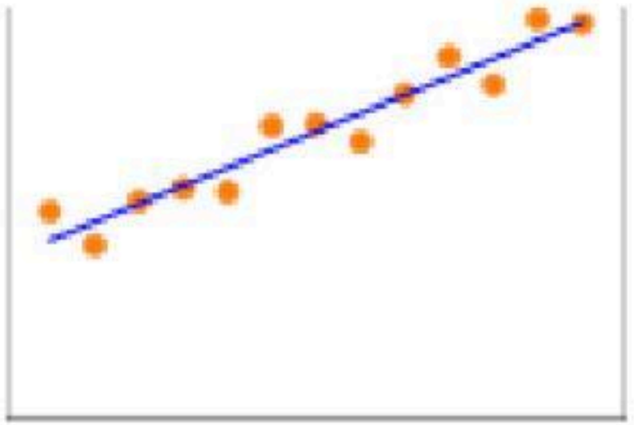


- **偏差**：偏离预期值的程度
- **方差**：数值扰动引起的变化程度

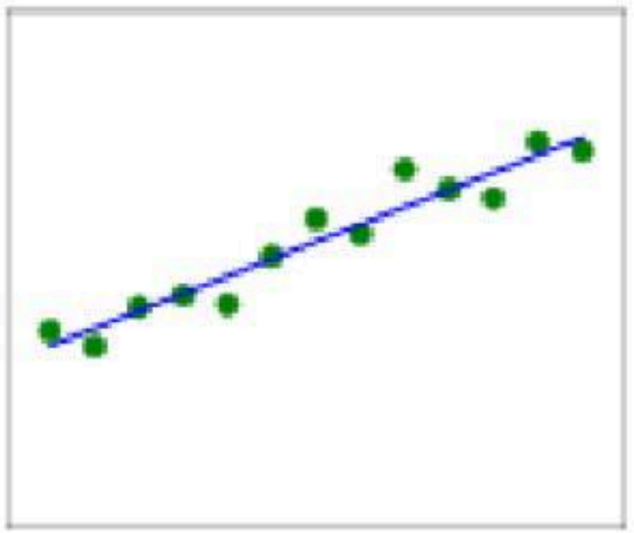
偏差、欠拟合

偏差一般源自**假设空间本身的局限性**

- 线性模型的偏差较高：斜率固定
- 分段线性的偏差较低：完全数据驱动



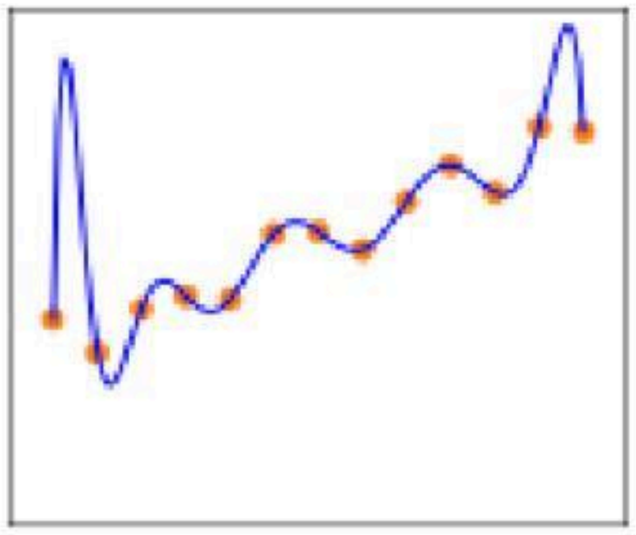
欠拟合：假设不能发现数据中某种内在规律



方差、过拟合

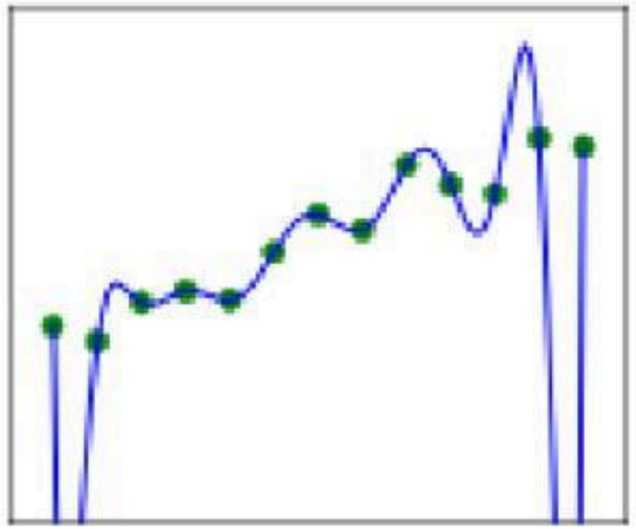
方差通常由于**假设过于复杂**

- 12次多项式必定可以插值13个数据点
- 注意：两组数据间只有轻微的扰动
  - 两个假设差异巨大：至少一个没能逼近真实分布



过拟合：对假设的优化过度强调训练集上的误差

- 极易导致弱泛化性，即对未知数据的预测不准



权衡偏差、方差

那种建模思路更好?

- 低偏差：模型复杂，数据拟合误差低，泛化弱
- 低方差：模型简单，数据拟合误差高，泛化强

权衡偏差、方差

那种建模思路更好?

- 低偏差：模型复杂，数据拟合误差低，泛化弱
- 低方差：模型简单，数据拟合误差高，泛化强

> *“the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.” – Albert Einstein*

权衡偏差、方差

那种建模思路更好?

- 低偏差：模型复杂，数据拟合误差低，泛化弱
- 低方差：模型简单，数据拟合误差高，泛化强

> *“the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.” – Albert Einstein*

Ockham’s razor 奥卡姆剃刀原则

“plurality [of entities] should not be posited without necessity”

如何定义“简单”?

参数个数并非很好的度量

- 线性模型只有两个参数；深度神经网络非常复杂，但仍然能很好地泛化

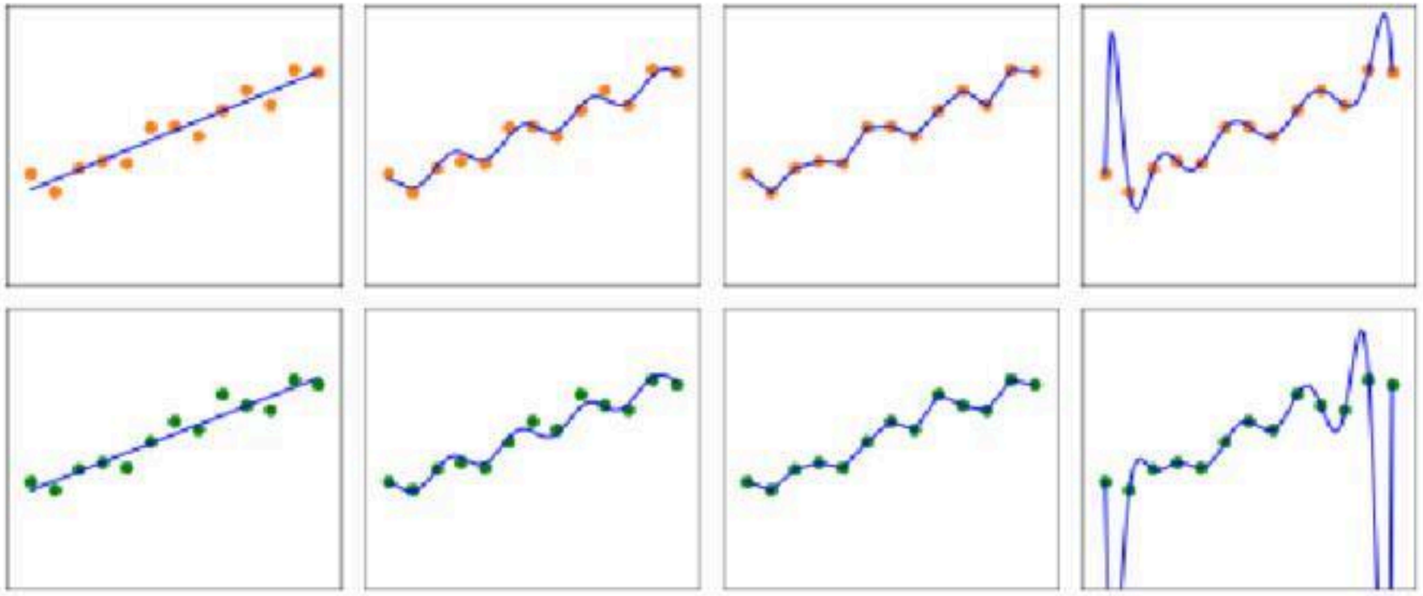
如何定义“简单”?

参数个数并非很好的度量

- 线性模型只有两个参数；深度神经网络非常复杂，但仍然能很好地泛化

数据拟合实例：哪个假设更好？如果没有足够的**先验知识**，无法确定

- 假设数据采自逐日跟踪的网络数据，依赖时间的循环模式就很合理
- 如果确定数据没有循环模式，数值的扰动很可能来自噪音



如何编码先验知识?

实践分析中，很难对一个假设做出“非对即错”的判断

- 通常需要计算其正确的**可能性** $P(h|data)$

如何编码先验知识?

实践分析中，很难对一个假设做出“非对即错”的判断

- 通常需要计算其正确的**可能性** $P(h|data)$

目标：选出给定数据的条件下，最**可能的一个假设**

$$h^* = \arg\max_{h \in \mathcal{H}} P(h|data) = \arg\max_{h \in \mathcal{H}} P(data|h)P(h)$$

如何编码先验知识?

实践分析中，很难对一个假设做出“非对即错”的判断

- 通常需要计算其正确的**可能性** $P(h|data)$

目标：选出给定数据的条件下，最**可能的一个假设**

$$h^* = \arg\max_{h \in \mathcal{H}} P(h|data) = \arg\max_{h \in \mathcal{H}} P(data|h)P(h)$$

- 偏好简单模型：给低次多项式较高的先验概率 $P(h)$
- 允许相对罕见的假设，但降低其先验概率

假设空间是否可以“无所不包”?

极端情况：$\mathcal{H}$ 选成所有计算机程序的集合，或所有图灵机

- 可表示性：理论上就可以解决任何可计算问题
- 计算复杂度：在无限大假设空间中找到最佳假设的可能性几乎为零

假设空间是否可以“无所不包”?

极端情况：$\mathcal{H}$ 选成所有计算机程序的集合，或所有图灵机

- 可表示性：理论上就可以解决任何可计算问题
- 计算复杂度：在无限大假设空间中找到最佳假设的可能性几乎为零

假设空间对问题进行**约束**，是简化计算的有效手段

- 相对简单的假设空间仍然是更佳选择
- 相对简单的假设更易于应用
  - 相反，图灵机甚至不能保证停机
  - 即使是深度神经网络：表示并不简单，但一定可以在一定步内完成计算

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**.

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**.

就像C语言：能够表达**任何可计算的程序**

- 问题是：如何把这个程序符合规范地写出来?
- 指数级复杂度：NP问题

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：⊙

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

(*)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）

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

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

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

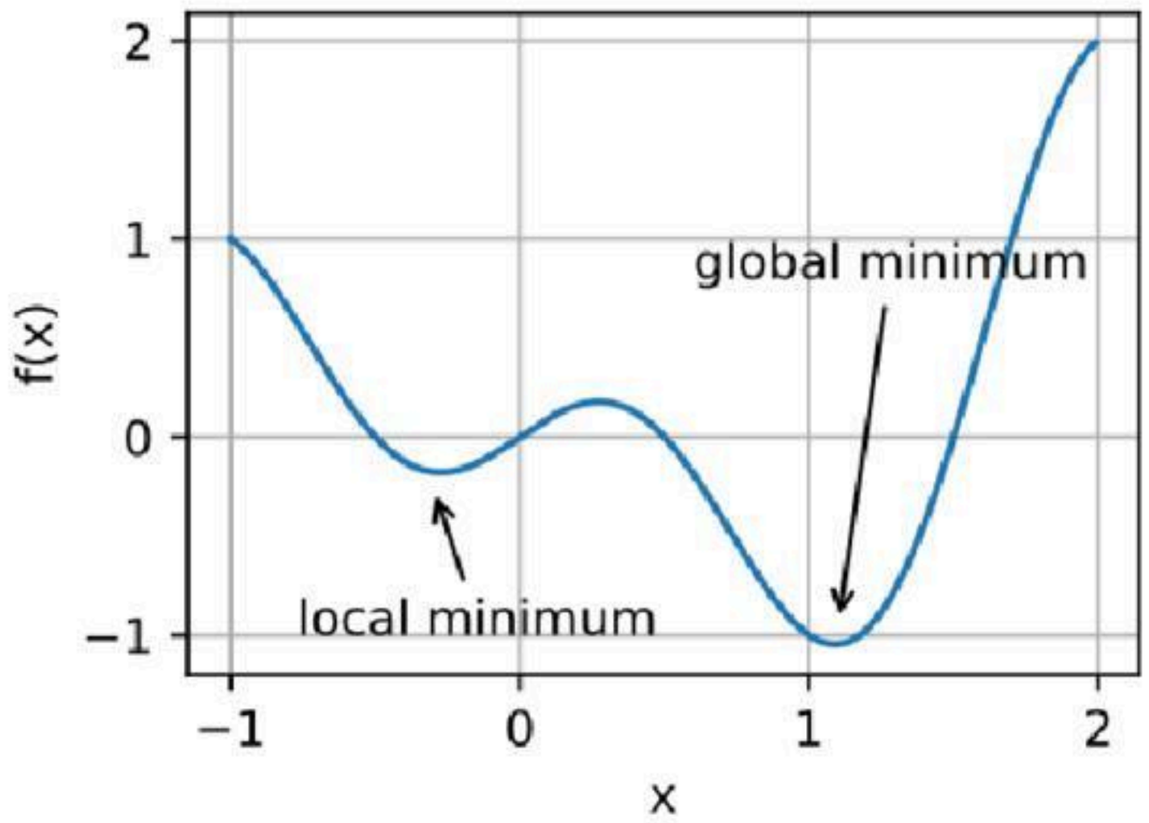
优化

优化、泛化

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

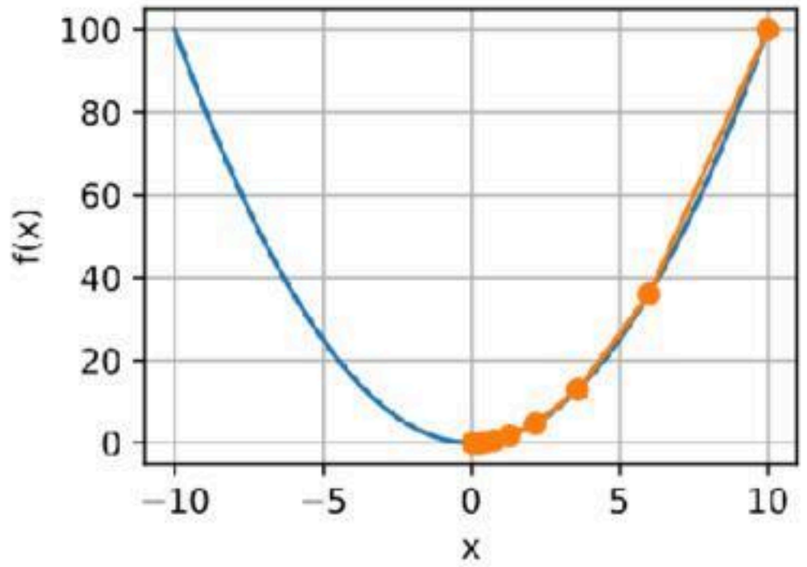
Taylor展开：函数的**一阶近似**

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

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

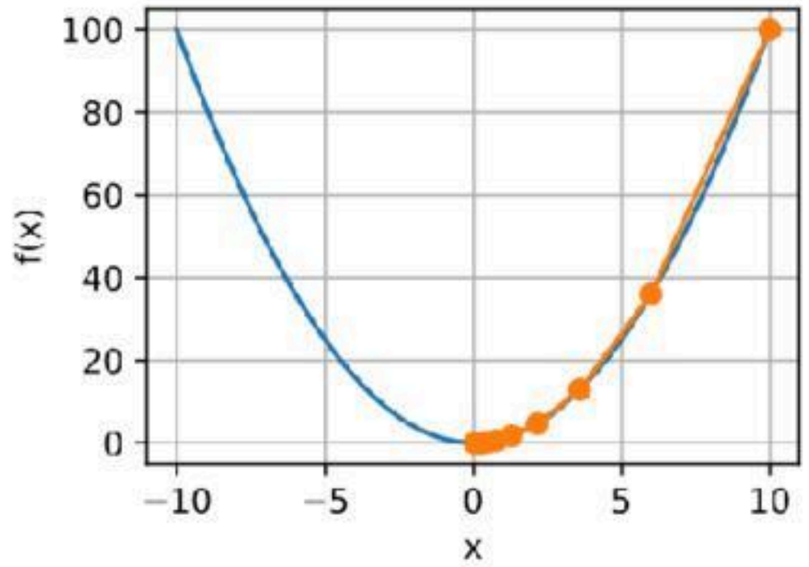
Taylor展开：函数的**一阶近似**

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

梯度：函数值**增长最快**的方向

- 向梯度的**反方向**走一小步

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

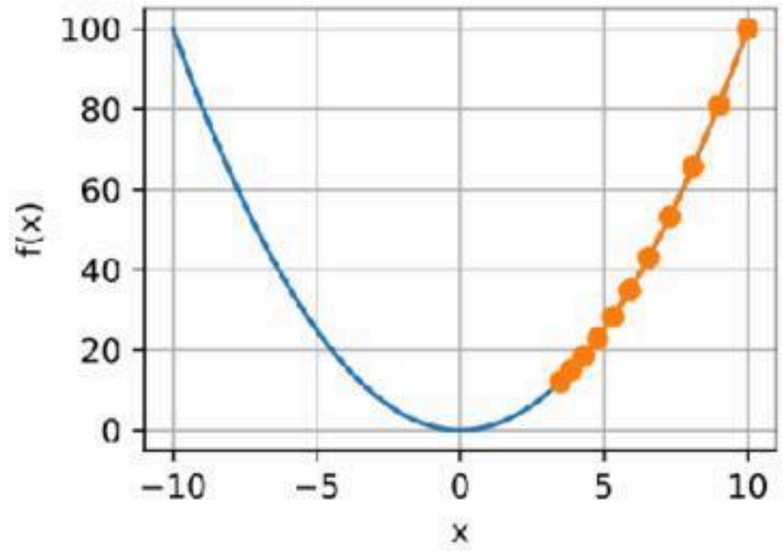


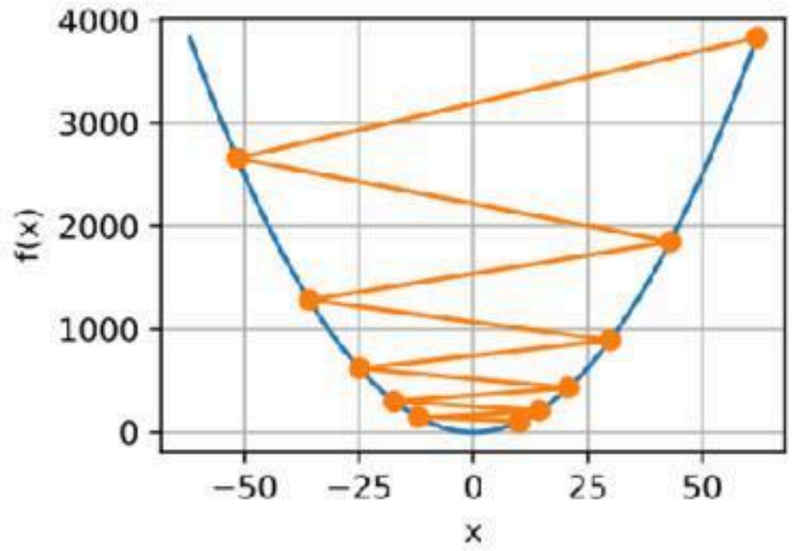
牛顿法：**二阶展开**，需要能够计算二阶导数（Hessian）

学习率

学习率 learning rate：η

- 太小：收敛慢
- 太大：发散

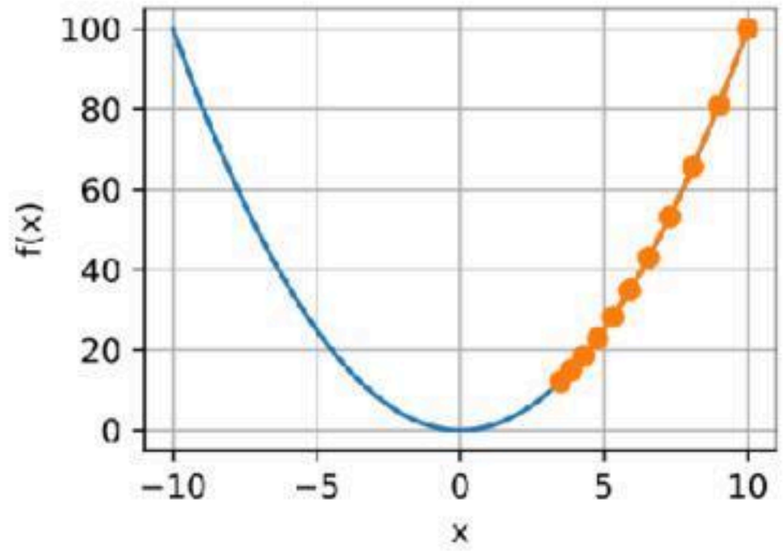


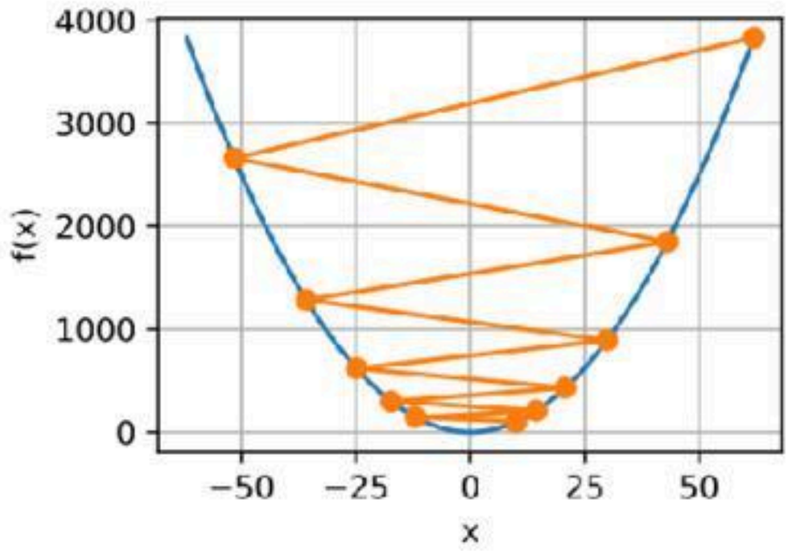


学习率

学习率 learning rate：η

- 太小：收敛慢
- 太大：发散



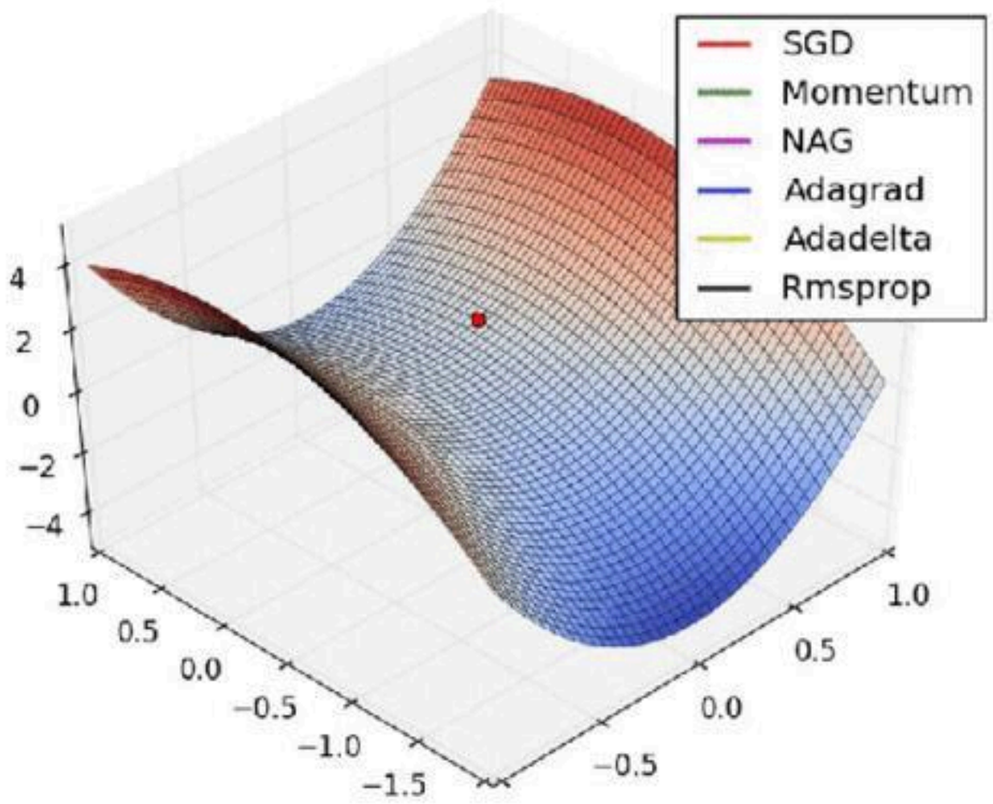


注意：地形平坦的区域步长自然变小

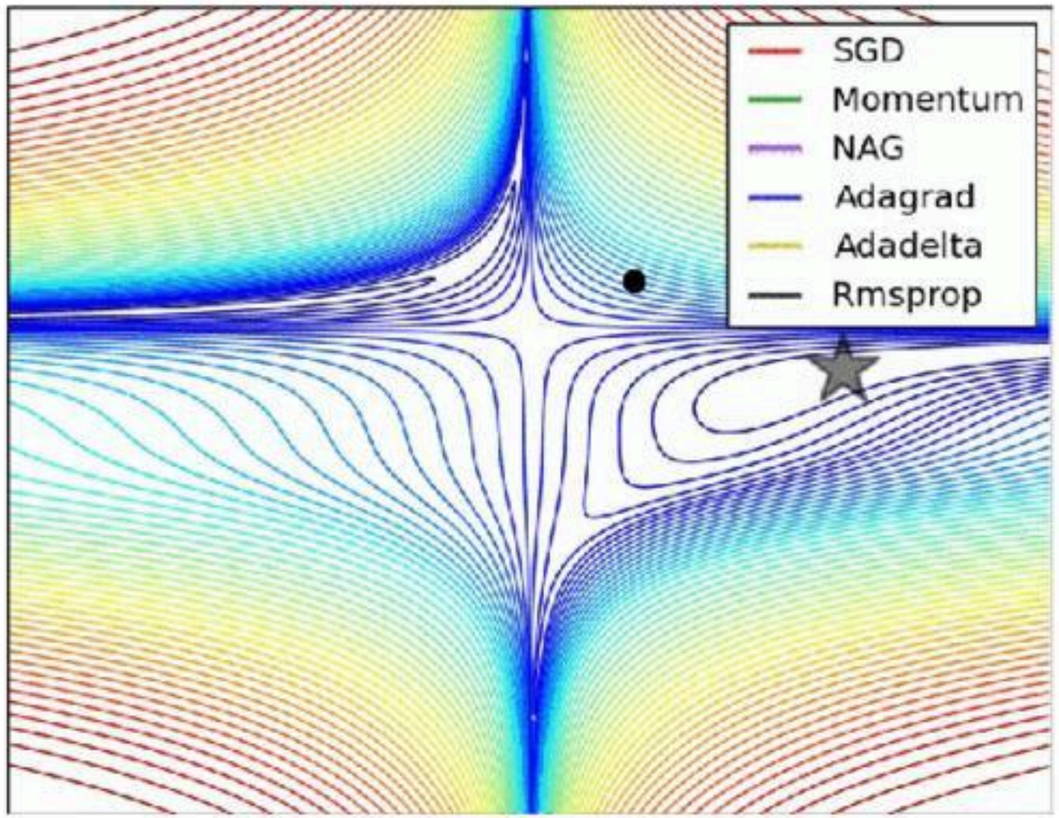
- $x \leftarrow x - f'(x)\eta$

带动量的梯度下降

摆脱鞍点：



摆脱多个极小值：



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：

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

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

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

链式法则

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

手算举例：

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

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

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

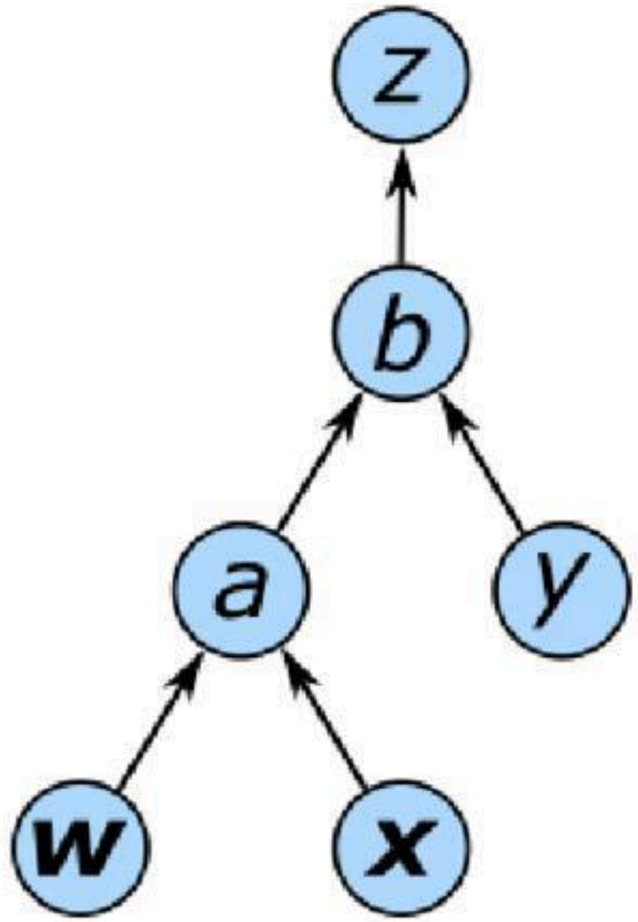
- 繁琐、易错；现在一般使用**自动微分**软件包

自动求导

首选构建**计算图**：将计算分解成算子的有向无环图

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

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



自动求导的两种模式

链式法则：乘法满足结合律，故求导顺序可以根据需要指定

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

- 前向求导：求导顺序自底向上

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

- **反向传递 backward propagation**：求导顺序自顶向下

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

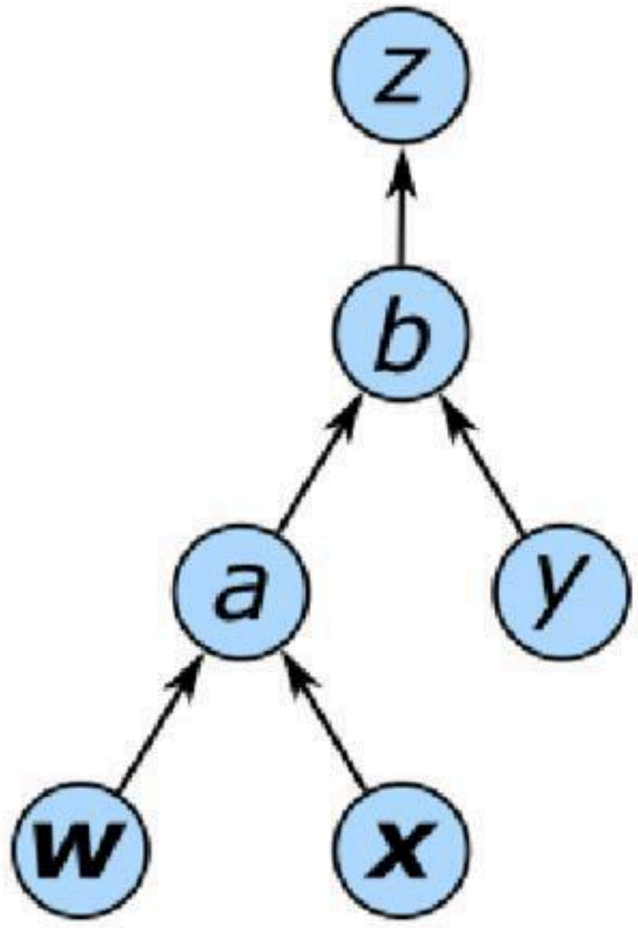
前向积累

反向传递算法分为两个主要步骤：

1. **前向积累 forward accumulation**：执行计算图
   - 直接目的：获取目标函数值
   - 隐含好处：存储中间结果

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

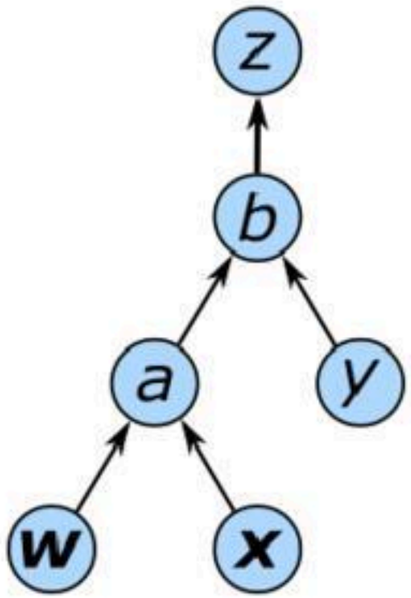
2. 反向传递：计算梯度值



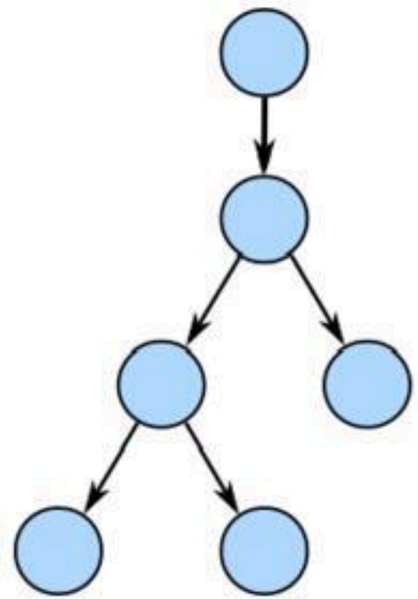
反向传递 I

反向：**计算梯度值。**

$$z = b^2$$



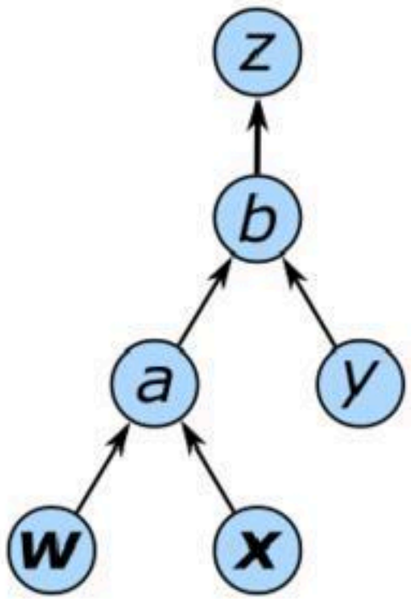
$$\frac{\partial z}{\partial b} = 2b \approx \frac{\Delta z}{\Delta b}$$



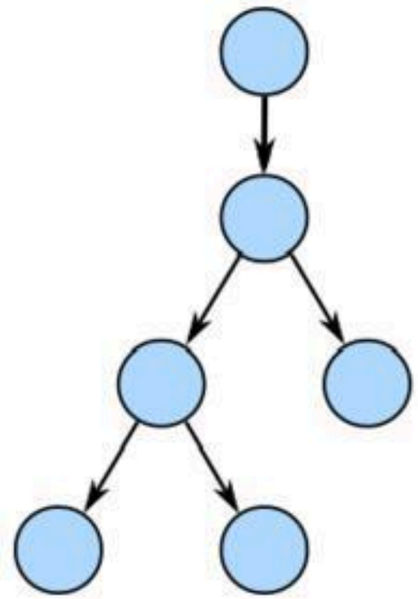
反向传递 II

反向：**计算梯度值。**

$$b = a - y$$



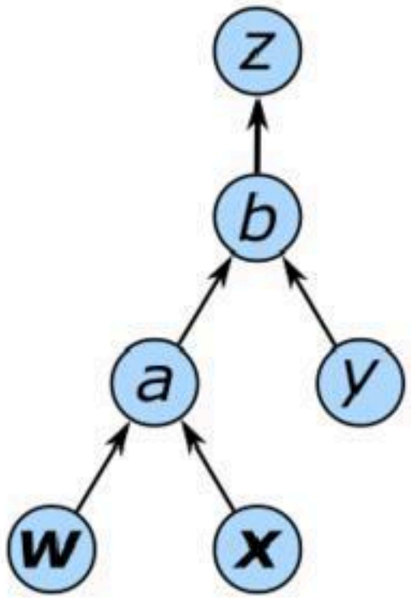
$$\frac{\partial b}{\partial a} = 1$$



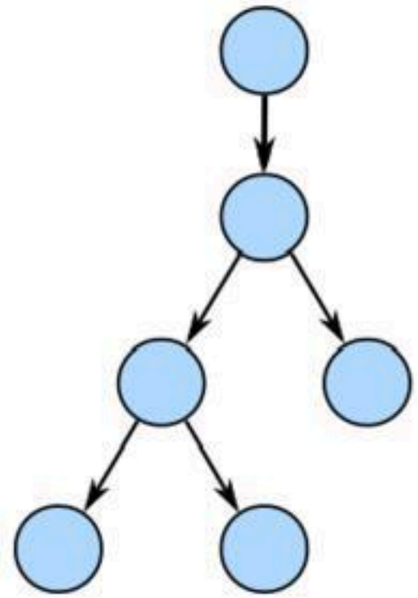
反向传递 III

反向：**计算梯度值**，剪除不需要的枝。

$$a = \langle \mathbf{x}, \mathbf{w} \rangle$$



$$\frac{\partial a}{\partial \mathbf{w}} = \mathbf{x}^T \approx \frac{\Delta a}{\Delta \mathbf{x}}$$

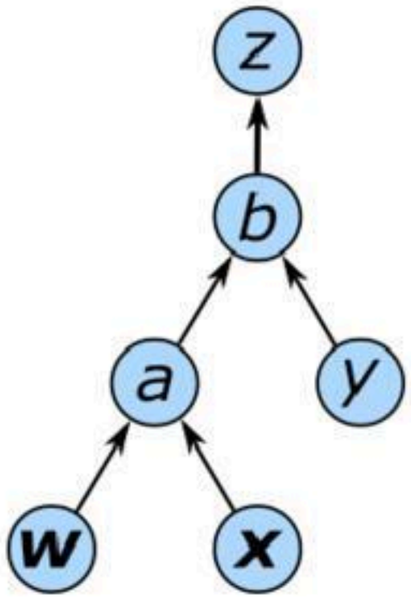


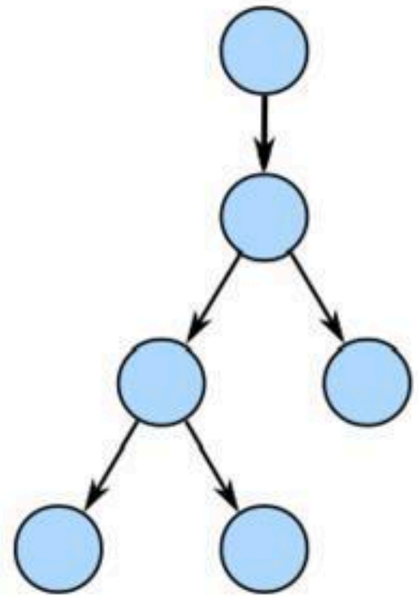
- 类比剪枝：剪除y对应的分支

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。

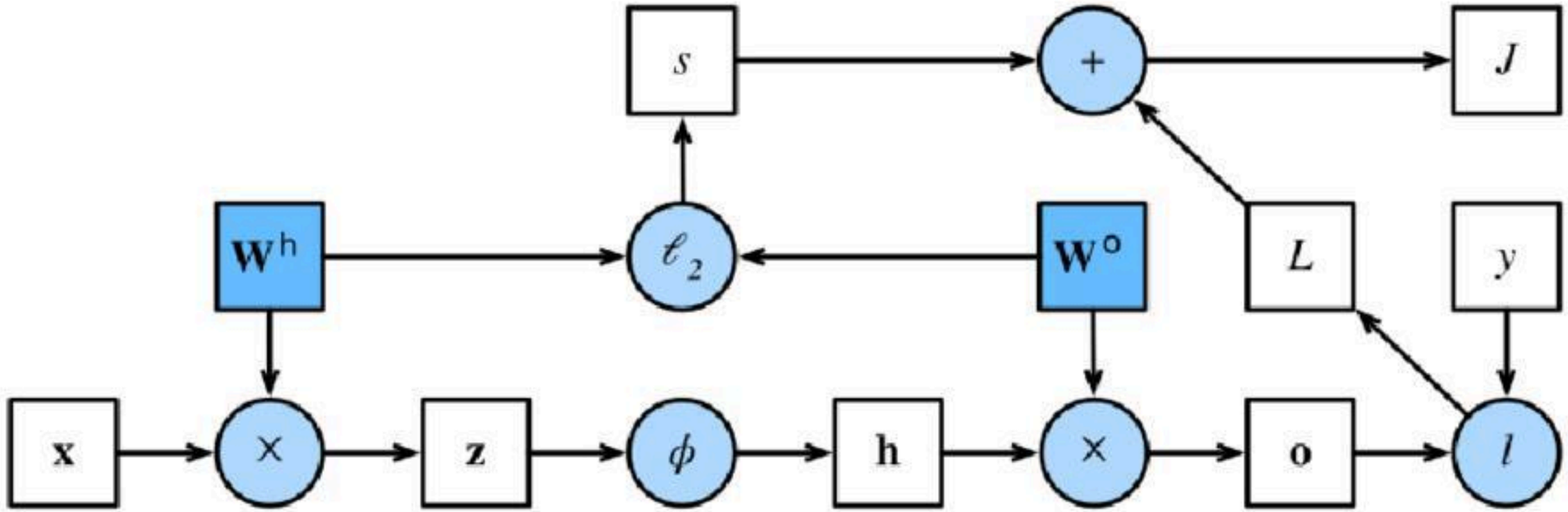
- 计算$\frac{\partial z}{\partial b}$需要$b$，进而需要$a$：已在前向累积中获取





- 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

Review

本章内容

1. 监督学习概念
2. 实验环境配置
3. 基础编程训练

重点：监督学习概念；实验环境配置；张量、张量运算。

难点：张量运算的几何解释；测试简单示例。

学习目标

1. 掌握机器学习的基本形式
2. 掌握监督学习相关概念
3. 掌握实验环境配置方法
4. 理解张量的概念
5. 掌握张量运算，并理解其几何解释

问题

1. (*)列举5种基本的二维仿射变换，并应用于图片变形。
2. (*)简述深度学习的几何解释。
3. 简述梯度下降法的算法流程。

深度学习工作站的简单配置

安装Anaconda/Miniconda

打开命令行终端Anaconda Powershell Prompt，并输入以下代码：

```
conda create -n dl pytorch torchvision torchaudio
        cudatoolkit=11.3 -c pytorch
```

MNIST 手写数字分类：简单全连接

典型的工作流程

1. 定义训练数据：输入张量和目标张量。
2. 定义层组成的网络（或模型），将输入映射到目标。
3. 配置学习过程：选择损失函数、优化器和需要监控的指标。
4. 调用优化函数在训练数据上进行迭代。

实际应用训练好的模型进行预测

- 调整输入数据的格式：预测输入要与训练输入格式一致。

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**。

二维仿射变换

二维仿射变换的一般形式

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

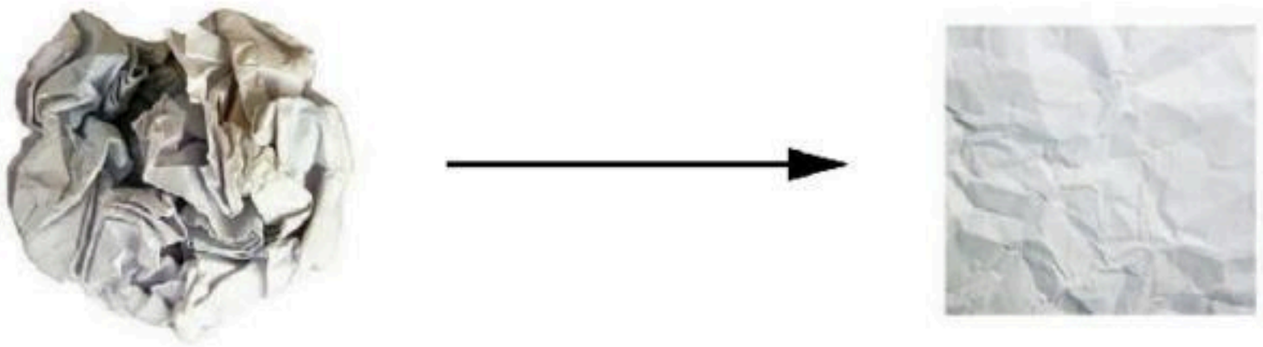
基本二维仿射变换矩阵

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

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”。

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。