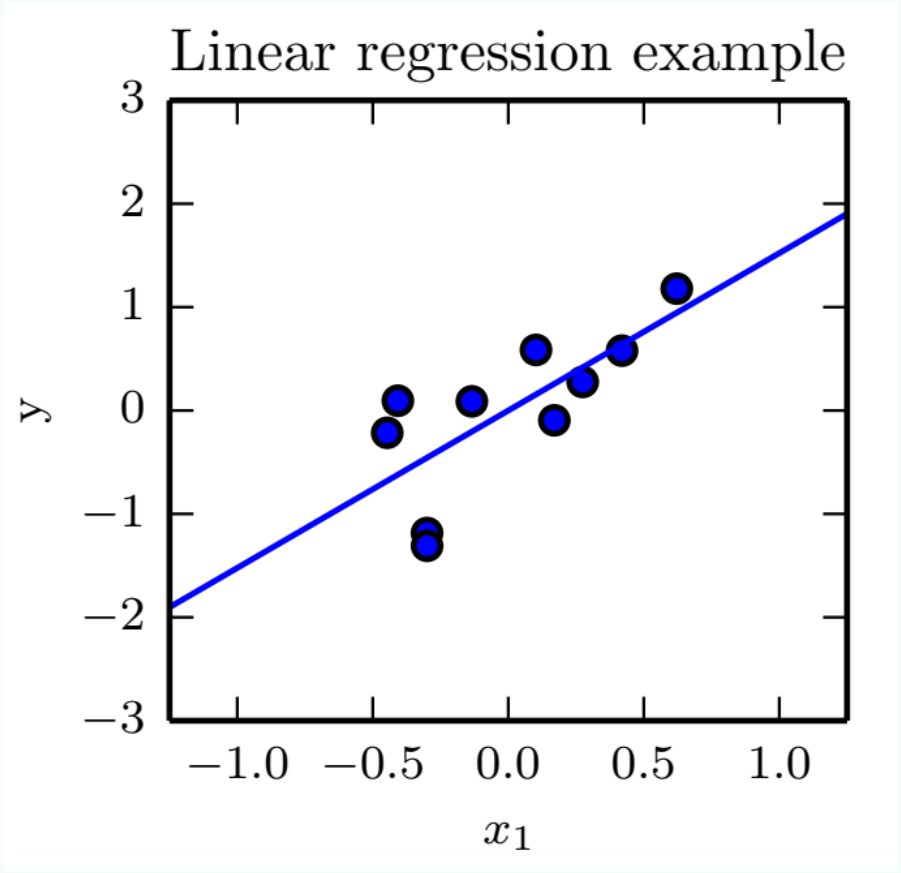
绪论

清华大学计算机系 陈键飞

《大模型计算》课程团队

函数拟合

- 给定训练数据集 $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
- x: D_X维向量，y: D_Y维向量
- 找到函数$f: \mathbb{R}^{D_X} \rightarrow \mathbb{R}^{D_Y}$
- 最小化拟合误差$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, f(x_i))$
- 函数包含某些未知参数w
- 训练(training)：从数据集$\mathcal{D}$中学习得到模型f（参数w）
- 推理(inference)：根据模型f为新的x预测$y = f(x)$



例：线性回归

- 房产估价模型
- 特征x1：房屋面积
- 特征x2：与市中心距离
- 标签y：房产价格

序号	区域	房屋面积（x1）	离市中心距离（x2）	房屋总价（Y）
1	西城-金融街附近	60	2	900
2	海淀-中关村	75	8	750
3	顺义-中央别墅区	220	25	1100
4	昌平-天通苑	80	20	400

例：线性回归

- 房产估价模型
- 特征x1：房屋面积
- 特征x2：与市中心距离
- 标签y：房产价格

序号	区域	房屋面积（x1）	离市中心距离（x2）	房屋总价(Y)	预测
1	西城-金融街附近	60	2	900	880
2	海淀-中关村	70	8	700	825
3	顺义-中央别墅区	220	25	1100	1095
4	昌平-天通苑	80	20	400	680

- 模型 $y = w1\ x1 + w2\ x2 + c$

每平米单价　　每公里惩罚　　基础房价

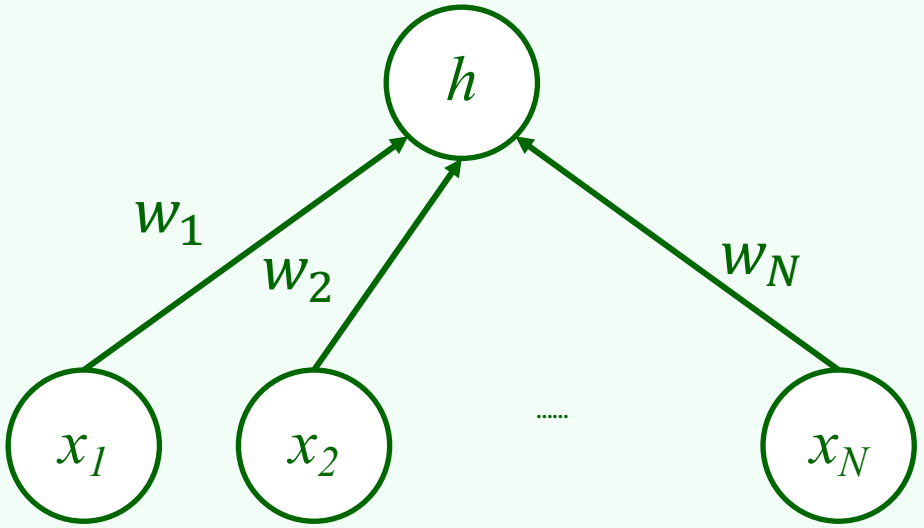
- $w1 = 3.5$　　$w2 = -15$　　$c = 700$

深度神经网络

清华大学计算机系 陈键飞

《大模型计算》课程团队

神经元



$$h = g\left(\sum_j w_j x_j + c\right)$$

向量形式

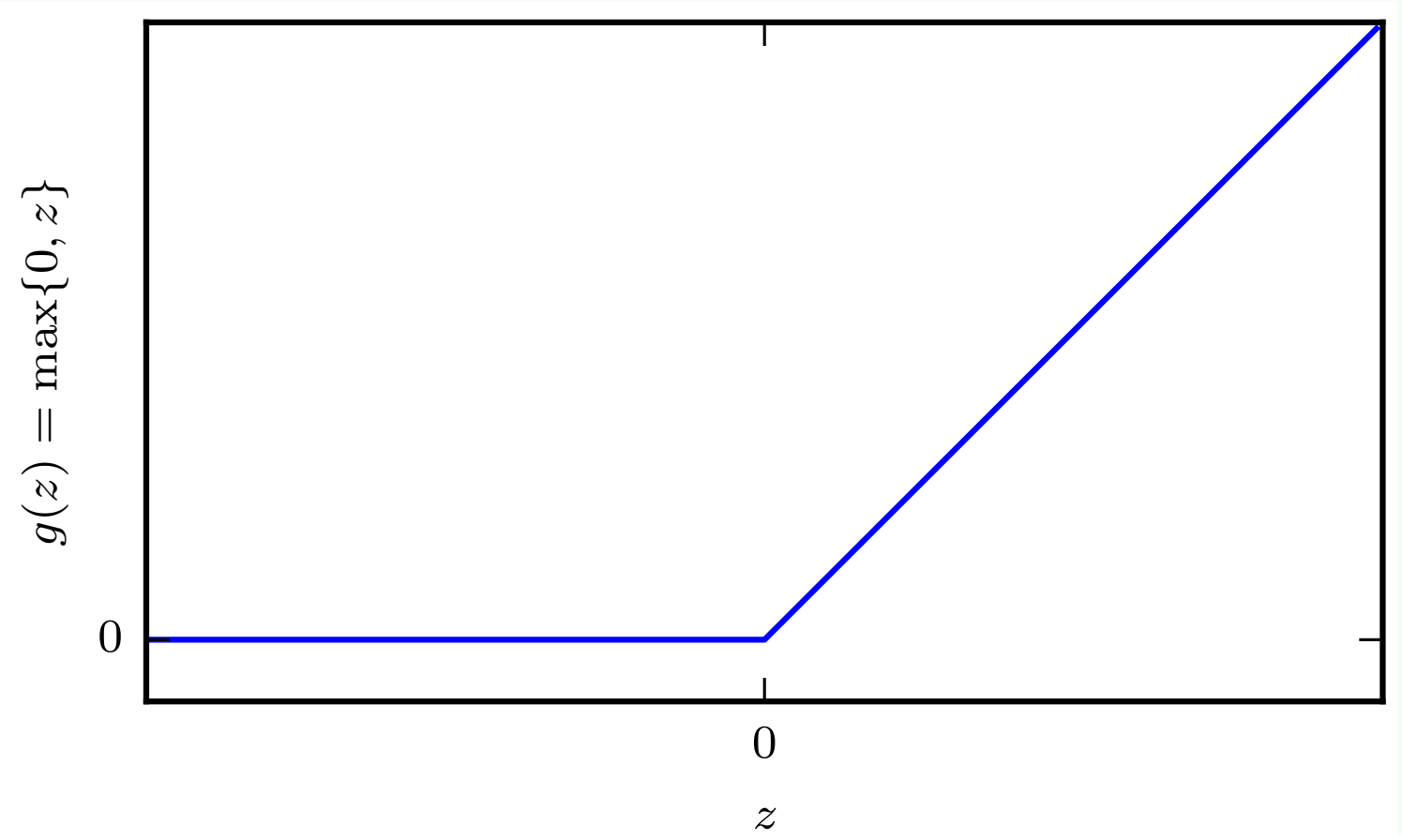
$$h = g(\boldsymbol{w}^\top \boldsymbol{x} + c)$$

权重 weight　　**激活/输入** activation　　**偏置** bias

单个神经元可以完成线性回归

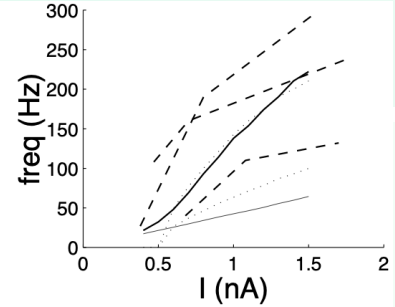
整流线性单元Rectifier Linear Unit (ReLU)

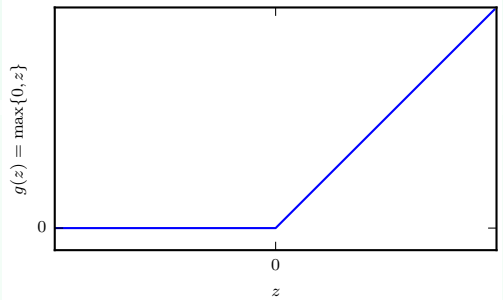
$$g(z) = \max\{0, z\}$$

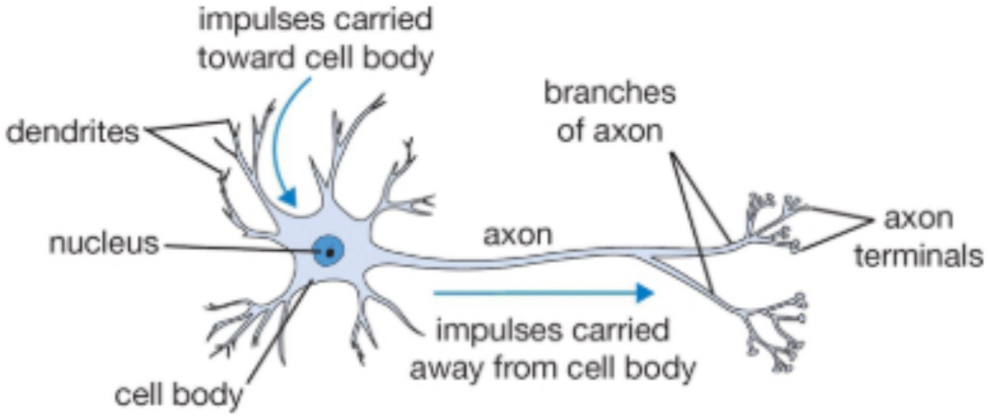


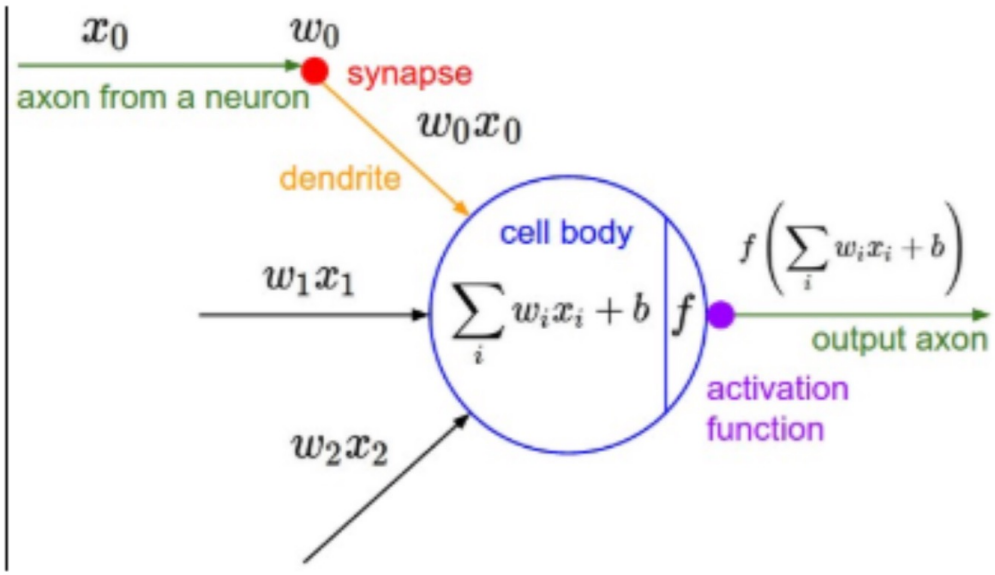
生物神经元

Troyer T W, Miller K D. Integrate-and-fire neurons matched to physiological fI curves yield high input sensitivity and wide dynamic range[M]//Computational Neuroscience: Trends in Research, 1997. Boston, MA: Springer US, 1997: 197-201.









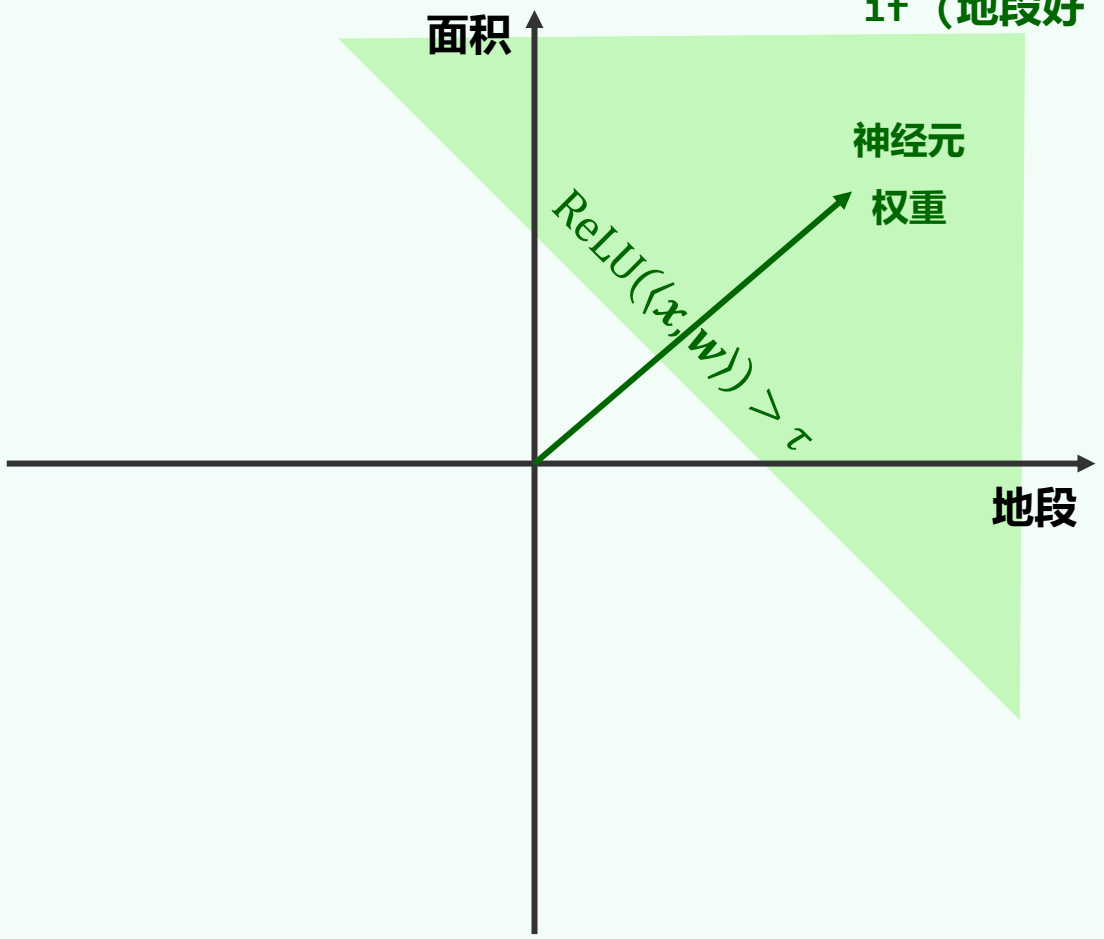
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

生物神经元 **人工神经元**

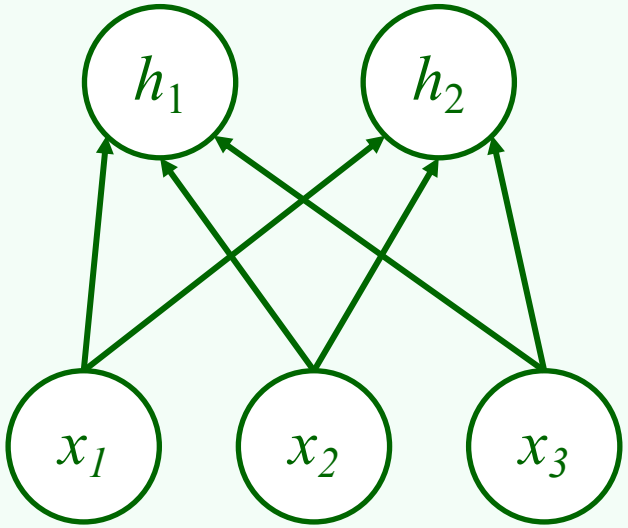
神经元 = 组合特征捕捉器

捕捉“地段好**且**面积大”概念

if（地段好 and 面积大）…



全连接层（线性层）



$$h_i = g\left(\sum_j w_{ij}x_j + c_i\right)$$

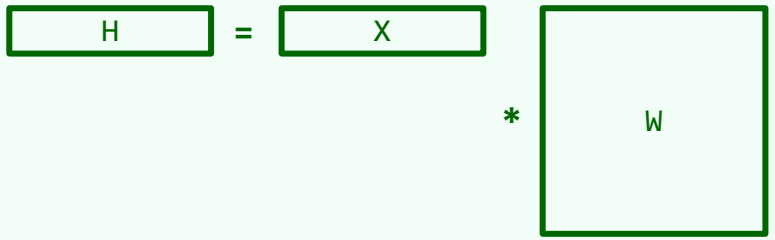
矩阵形式（输入为向量）

$$\boldsymbol{h} = g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{c})$$

矩阵形式（输入为矩阵）

$$\boldsymbol{H} = g\left(\boldsymbol{X}\boldsymbol{W}^{\mathrm{T}} + \boldsymbol{c}\right)$$

N*C　N*D　D*C　1*C

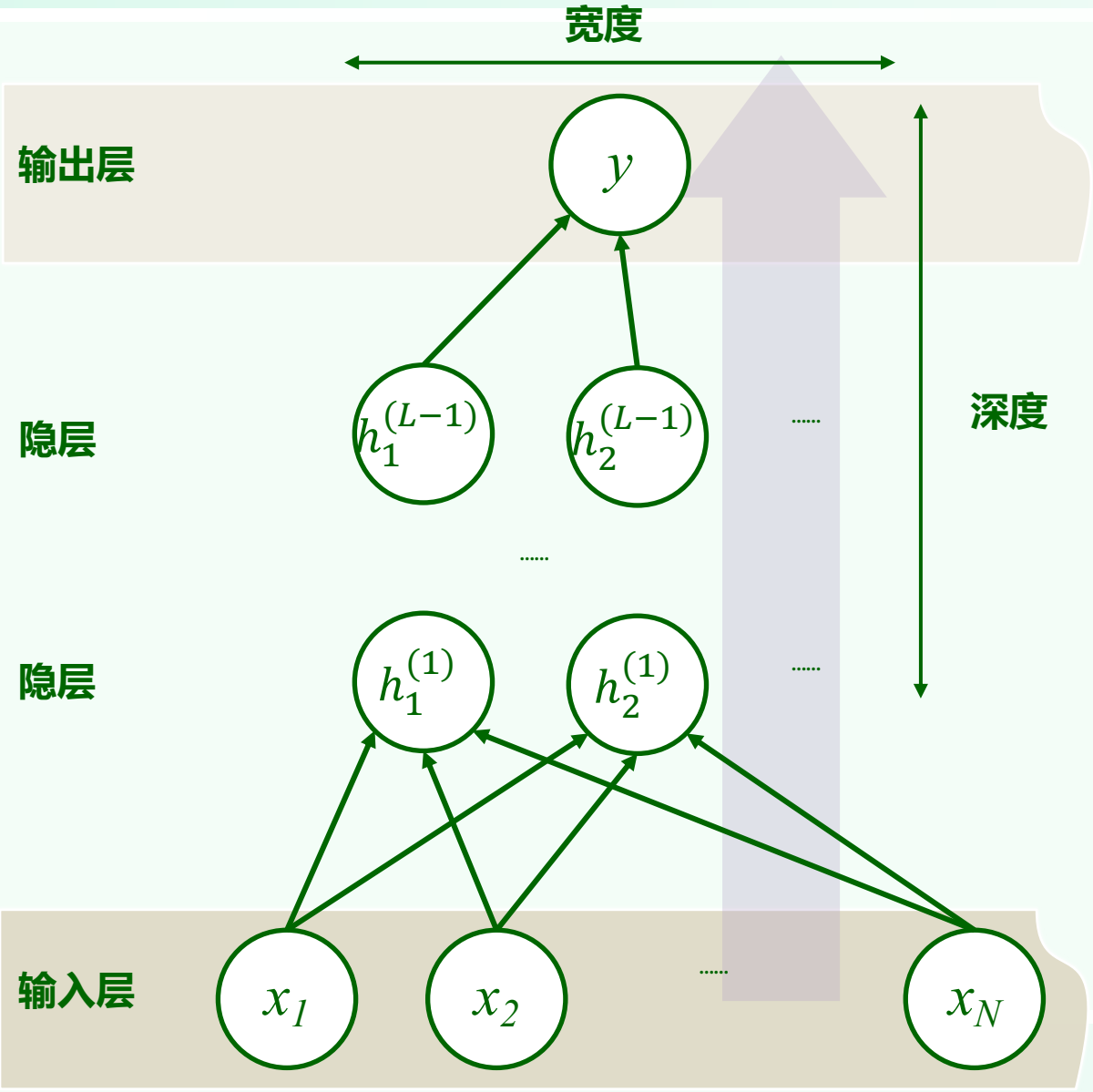


N：数据

C：输出维度

D：输入维度

神经网络




```
h1 = linear(x, w1)
h2 = linear(h1, w2)
        …
y  = linear(hL-1, wL)
```

$$f(x; W)$$

数据　　模型

全连接神经网络 (Fully-connected network)

多层感知机 (Multi-layer perceptron, MLP)

前馈网络 (Feed-forward network, FFN)

例：线性回归

- 房产估价模型
- 特征x1：房屋面积
- 特征x2：与市中心距离
- 标签y：房产价格

序号	区域	房屋面积（x1）	离市中心距离（x2）	房屋总价(Y)	预测
1	西城-金融街附近	60	2	900	880
2	海淀-中关村	70	8	700	825
3	顺义-中央别墅区	220	25	1100	1095
4	昌平-天通苑	80	20	400	680

- 模型 $y = w1\ x1 + w2\ x2 + c$

每平米单价　　每公里惩罚　　基础房价

- $w1 = 3.5$　　$w2 = -15$　　$c = 700$

例：线性回归

- 房产估价模型
- 特征x1：房屋面积
- 特征x2：与市中心距离
- 标签y：房产价格

序号	区域	房屋面积（x1）	离市中心距离（x2）	房屋总价(Y)	预测
1	西城-金融街附近	60	2	900	880
2	海淀-中关村	70	8	700	825
3	顺义-中央别墅区	220	25	1100	1095
4	昌平-天通苑	80	20	400	680
5	西城-金融街附近	200	2	4000	1370

- 模型 $y = w1\ x1 + w2\ x2 + c$

每平米单价　　每公里惩罚　　基础房价

- $w1 = 3.5 \quad w2 = -15 \quad c = 700$

例：线性回归

- 房产估价模型
- 特征x1：房屋面积
- 特征x2：与市中心距离
- 标签y：房产价格

序号	区域	房屋面积（x1）	离市中心距离（x2）	单价	房屋总价(Y)	预测
1	西城-金融街附近	60	2	15	900	880
2	海淀-中关村	70	8	10	700	825
3	顺义-中央别墅区	220	25	5	1100	1095
4	昌平-天通苑	80	20	5	400	680

单价受地段影响很大！

- 模型 $y = w1\ x1 + w2\ x2 + c$

每平米单价　　每公里惩罚　　基础房价

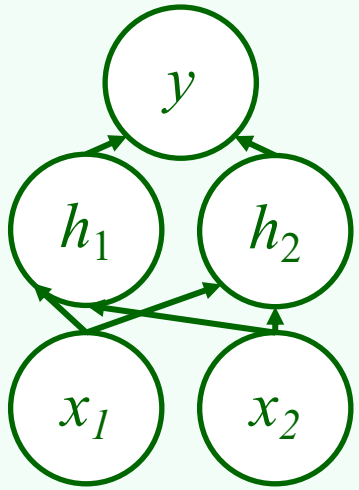
- $w1 = 3.5$　　$w2 = -15$　　$c = 700$

例：线性回归

- 房产估价模型
- 特征x1：房屋面积
- 特征x2：与市中心距离
- 标签y：房产价格

序号	区域	房屋面积（x1）	离市中心距离（x2）	单价	房屋总价(Y)	h1	h2	预测
1	西城-金融街附近	60	2	15	900	300	596	896
2	海淀-中关村	70	8	10	700	350	344	694
3	顺义-中央别墅区	220	25	5	1100	1100	0	1100
4	昌平-天通苑	80	20	5	400	400	0	680

- 基础房价　　h1 = 5 * x1
- 核心城区溢价　　h2 = ReLU(12 * x1 – 62 * x2)　有指向性的筛选出部分数据
- 　　y = h1 + h2



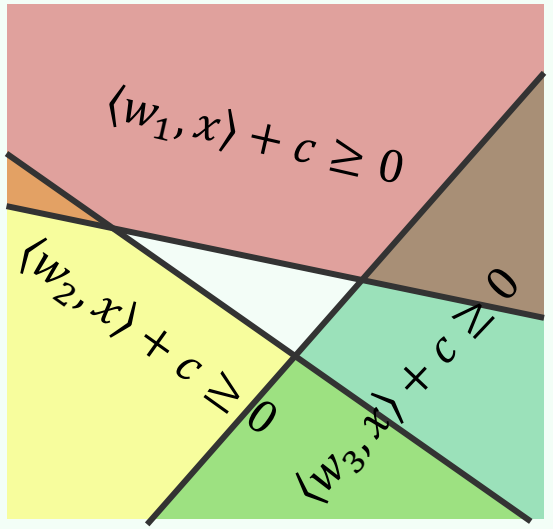
例：线性回归


y = 5 * x1
y = 17 * x1 - 62 * x2
两者取较大
海淀区
德胜街道
北二环
太阳宫桥
四元桥
将府公园
北京朝阳站
京密引水渠
定慧桥
东城区
朝阳区
西城区
东四环
朝阳路
远通桥
石景山区
复兴路
北京站
广渠门桥
南沙窝桥
北京西站
衙门口桥
东五环
北京园博园
岳各庄桥
丽泽桥
菜户营桥
北京南站
分钟寺桥
园博园桥
丰台区
豆各庄(地区

万有近似定理（Universal Approximation Theorem）

给定任意数据分布$p(x)$及连续函数$f(x)$，存在一族单隐层神经网络，当网络宽度趋于无穷时，对(x, f(x))的拟合误差趋于0。

```
if input is x1 then output y1
if input is x2 then output y2
if input is x3 then output y3
……
```



Cybenko G. Approximation by superpositions of a sigmoidal function[J]. Mathematics of control, signals and systems, 1989, 2(4): 303-314.

PyTorch

清华大学计算机系 陈键飞

《大模型计算》课程团队

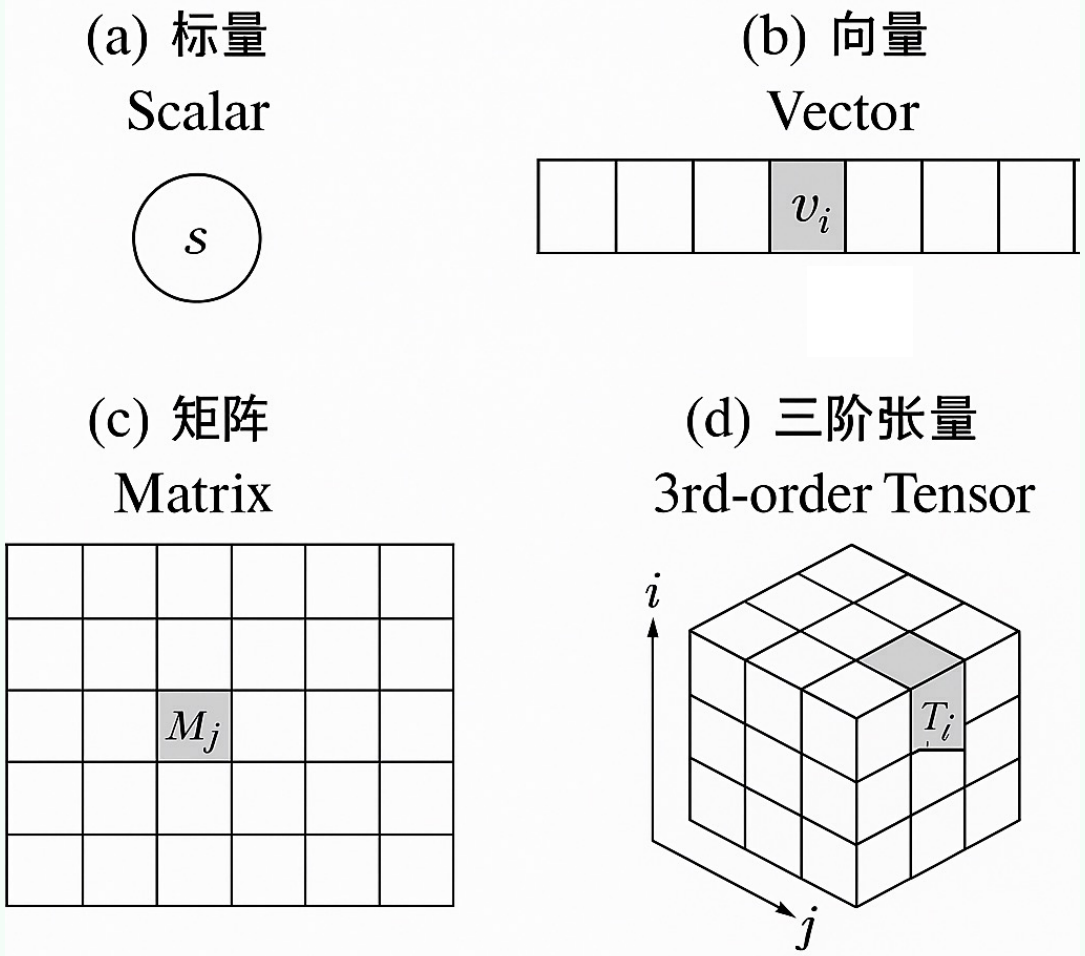
张量

- 数值组成的网格
- 形状Shape [D_1, D_2, …, D_R]

R阶张量

R个轴 (axis)

- 标量 []
- 3维向量 [3]
- 5*3矩阵 [5, 3]
- Transformer隐层 [128, 512, 768]

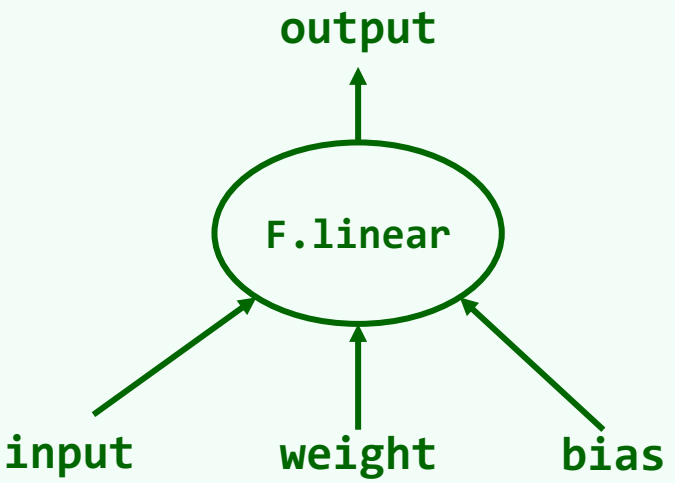


算子

- 多张量输入，多张量输出的函数

$$y1,y2,\ldots,ym = f(x1,x2,\ldots,xn)$$

- 逐元素加法 `torch.add(x, y)`
- 矩阵乘法 `torch.dot(x, y)`
- 转置 `x.t()`
- 奇异值分解 `U, s, V = torch.svd(X)`
- 全连接层 `h = F.linear(input, weight, bias)`



模块

❖ 神经网络模型的抽象

- ➢ 参数　　Parameter
- ➢ 前向传播 forward

```
class Linear:
    def __init__(self, in_features: int, out_features: int, bias: bool = True) :
        self.weight = Parameter(torch.empty((out_features, in_features)))
        if bias:
            self.bias = Parameter(torch.empty(out_features,))
        self.reset_parameters()                                    # 初始化参数

    def forward(self, input: Tensor):
        return F.linear(input, self.weight, self.bias)             # input @ self.weight.t() + self.bias
```

统计机器学习

清华大学计算机系 陈键飞

《大模型计算》课程团队

离散概率分布

- **V面骰子**🎲 $Y \sim \text{Categorical}(\boldsymbol{p})$，**概率**$P(Y = v) = p_v$
- **观测到了N次骰子投掷的结果**$\mathcal{D} = y_1, \ldots, y_N$，**其中**$y_i \in \{1, \ldots, V\}$
- **问题：如何估计骰子自身的参数**$\boldsymbol{p}$**?**

- **极大似然估计：给定参数**$\boldsymbol{p}$，**出现结果**$\mathcal{D}$**的概率为**

$$P(\mathcal{D}; \boldsymbol{p}) = \prod_{i=1}^{N} p_{y_i} = \prod_{v=1}^{V} p_v^{s_v}$$

离散概率分布

- V面骰子 $Y \sim \text{Categorical}(\boldsymbol{p})$，概率 $P(Y = v) = p_v$
- 观测到了N次骰子投掷的结果 $\mathcal{D} = y_1, \ldots, y_N$，其中 $y_i \in \{1, \ldots, V\}$
- 问题：如何估计骰子自身的参数 $\boldsymbol{p}$?

- 极大似然估计：给定参数 $\boldsymbol{p}$，出现结果 $\mathcal{D}$ 的概率为

$$P(\mathcal{D}; \boldsymbol{p}) = \prod_{i=1}^{N} p_{y_i} = \prod_{v=1}^{V} p_v^{s_v}$$

- 负对数似然度（negative log-likelihood, NLL）

$$-\log P(\mathcal{D}; \boldsymbol{p}) = -\sum_v s_v \log p_v$$

- 独热（one-hot）编码

整数表示	坐标向量表示
$y = 1$	$\boldsymbol{y} = (1,0,0,0)$
$y = 2$	$\boldsymbol{y} = (0,1,0,0)$
$y = 3$	$\boldsymbol{y} = (0,0,1,0)$
$y = 4$	$\boldsymbol{y} = (0,0,0,1)$

- 充分统计量

各面计数

$$\boldsymbol{s} = \prod_{i=1}^{N} \boldsymbol{y}_i$$

极大似然估计

❖ **极大似然估计：从所有可能的参数p中找出现结果$\mathcal{D}$的概率最大者**

$$\min_{\boldsymbol{p}} -\log P(\mathcal{D};\boldsymbol{p}) = -\sum_v s_v \log p_v$$
$$\text{s.t.} \sum_{v=1}^{V} p_v = 1$$

❖ **拉格朗日乘子**

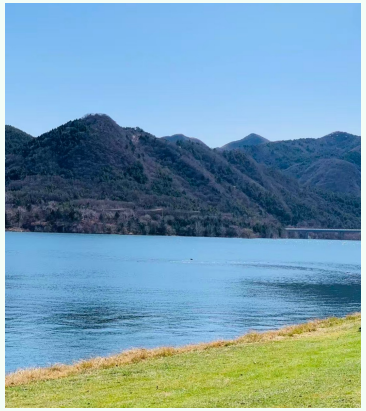
$$\mathcal{L} = -\sum_v s_v \log p_v + \lambda\left(\sum_v p_v - 1\right)$$

❖ **求解**

$$\mathcal{L}'_{p_v} = \frac{s_v}{p_v} + \lambda = 0 \quad \Rightarrow \quad p_v = -\frac{s_v}{\lambda}$$
$$\sum_{v=1}^{V} p_v = 1 \quad \Rightarrow \quad -\sum_v \frac{s_v}{\lambda} = 1 \quad \Rightarrow \quad \lambda = -\sum_v s_v$$
$$\Rightarrow p_v = \frac{s_v}{\sum_w s_w}$$

分类问题

- 问题：骰子的概率取决于条件x，要估计的是条件概率$P(y|x)$
- 例：天气预测



$P(y = \text{晴}\,|x) = 0.9$
$P(y = \text{雨}\,|x) = 0.1$

$P(y = \text{多云}\,|x) = 0.7$

$P(y = \text{雨}\,|x) = 0.9$

- 挑战：x和y之间关系可能很复杂
- 思路：利用神经网络建模

分类问题

- **模型：** $P(\boldsymbol{y}|\boldsymbol{x})\sim\text{Categorical}(\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}))$
- $\boldsymbol{f}$：从x（D维）到p（V维）
- 模型参数由概率向量p变成网络权重w

- **负对数似然度** $-\log P(\mathcal{D};\boldsymbol{w}) = -\sum_v s_v \log \boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})_v$
- 等等……哪里不太对?

分类问题

- **模型：** $P(\boldsymbol{y}|\boldsymbol{x})\sim\text{Categorical}(\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}))$
- $\boldsymbol{f}$：从x（D维）到p（V维）
- 模型参数由概率向量p变成网络权重w

- **负对数似然度** $-\log P(\mathcal{D};\boldsymbol{w}) = -\sum_v s_v \log \boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})_v$
- 骰子🎲的概率p应该是归一化的，但神经网络的输出无法保证这点……
- 解决方法：手工归一化

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}) = \text{softmax}\big(\underbrace{\boldsymbol{g}(\boldsymbol{x};\boldsymbol{w})}_{\text{神经网络}}\big)$$

概率 ⇦

probability

logits

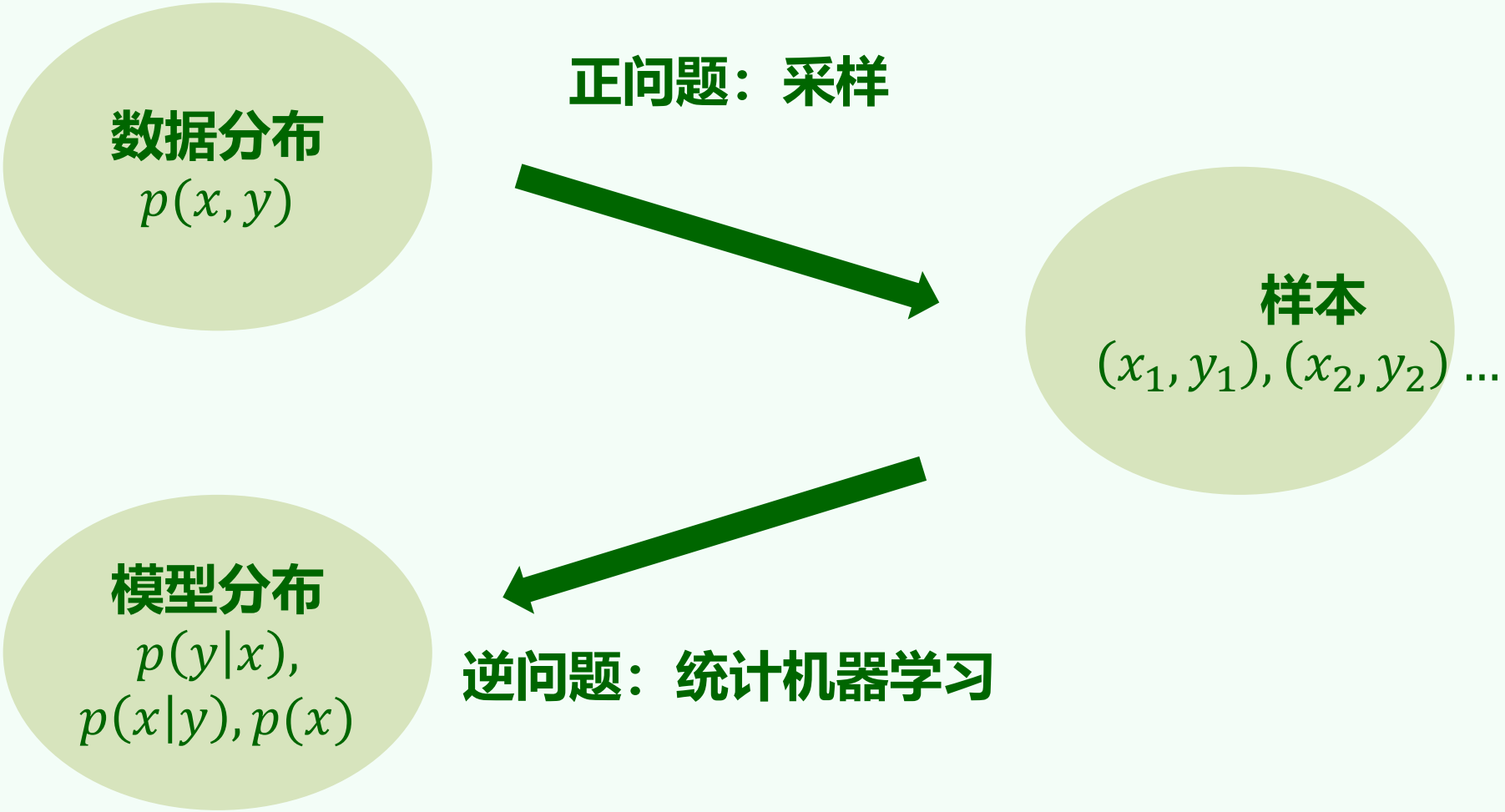
$$\text{softmax}(\mathbf{g})_v = \exp(g_v)/\sum_w \exp(g_w)$$

交叉熵损失

- **模型：** $P(\boldsymbol{y}|\boldsymbol{x})\sim\text{Categorical}(\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}))$
- $\boldsymbol{f}$：从x（D维）到p（V维）
- 模型参数由概率向量p变成网络权重w

- 负对数似然度 $-\log P(\mathcal{D};\boldsymbol{w})=-\sum_v s_v \log \boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})_v=-\sum_v s_v g_v(x;w)+N\log\sum_w \exp(g_w(x;w))$
- 无解析最小值，需要用数值方法（AdamW）迭代求解……
- 模型有什么用?
- 采样：　　给定x，产生 $y\sim p(y|x)$
- 概率计算：给定x和y，计算 $p(y|x)$

小结



大语言模型

清华大学计算机系 陈键飞

《大模型计算》课程团队

语言模型

- 学习概率分布$P(\text{文档})$
- P是 文档$\mapsto[0,1]$ 的映射，给每篇可能的文档一个概率值

我的家乡有一条清澈的小河，鱼儿在水中嬉戏，河边柳树轻轻摇曳。我和小伙伴常在这里玩耍，快乐无比。我爱我的家乡！	我滴家多有一条清彻见低的小盒，鱼仔在水中自游自在地嘻戏，河边柳树轻摇业。我和小伙半常在这里晚要，快乐比无。我哎我家多	荇蓒餈美蔻篪訖葽菡萎葉莘葅莖蒿薶荳蕢菻莊䌡摹蔦蓮造淺斜蓓許莧蒲蔆祥葅葯膺蕢聃縈蓌移稍蕉蕚犁葙蕭梅蔾蓶蔌葜蔬冀薴蘄酤蔔葺登
P=1e-100	P=1e-120	P=1e-500

- 如何把文档转换成模型能理解的形式？（最好是向量）

分词（tokenize）

❖ 如何把文档表示成向量?

It’s a good day today.

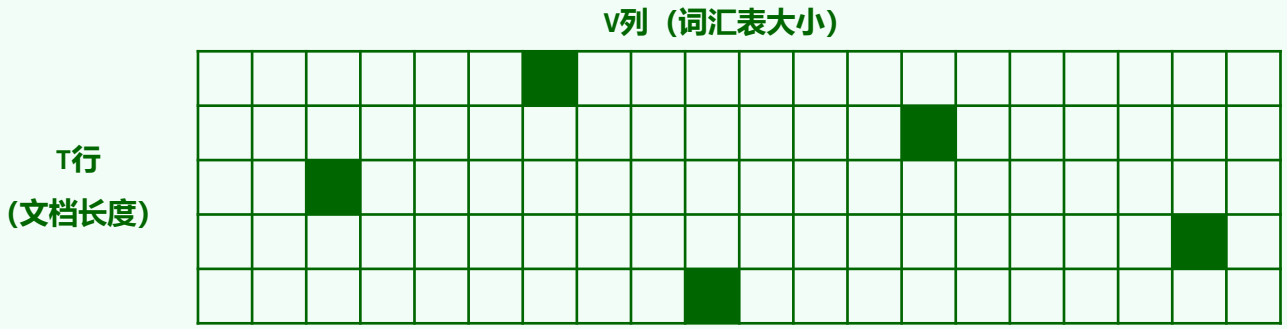
文字转整数

(15000, 1, 10000, 3000, 50000, 80000)

Token

词元

整数转独热



注：该过程并不精确，实际分词的算法比此处复杂。

❖ 词汇表

Id	词
1	a
2	abandon
3000	day
10000	good
15000	It’s
50000	today
80000	.
100000	<EOS>

序列建模

- 如何建模长度可变，且最长可能可达128K，甚至1M的词元序列？
- 长度可变：设计特殊词元<EOS>

- 数据变为定长的词元序列$(x_1, \dots, x_T)$
- 直接利用统计的手段建模$P(x_1, \dots, x_T)$需要存储V^T个参数……

- $V = 10^6, T = 8192$，则$V^T \approx 10^{50000}$

自回归模型（autoregressive model）

- **思路：每次只预测一个词**
- $p(x_1, \ldots, x_T) = p(x_1)p(x_2|x_1) \ldots p(x_T|x_1, \ldots, x_{T-1})$
- **把T维分布的建模问题转换为T个1维分布的建模问题**

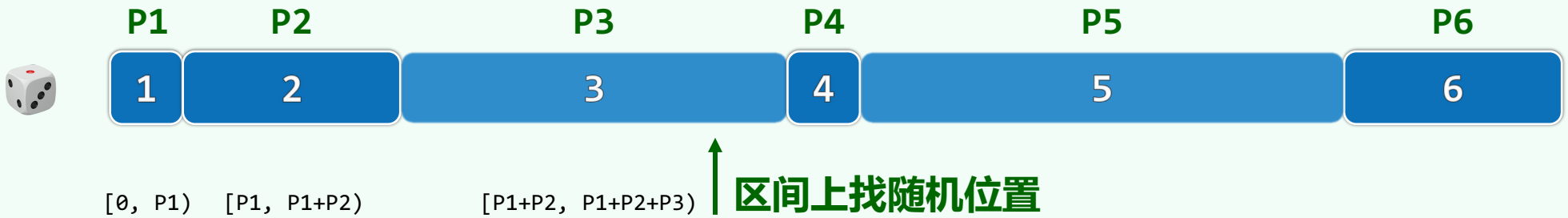
- $p(x_i|x_{<i}) = \mathrm{softmax}\big(g(x_{<i})\big)$

It’s a good day _____

- P(x5 | “It’s a good day”) = …
- 采样

Id	词	概率
1	a	1e-7
2	abandon	1e-7
3000	day	0.01
10000	good	0.0001
15000	It’s	1e-7
50000	today	1e-7
80000	.	1e-7
100000	<EOS>	1e-7

小知识：从离散分布中采样




```
Input: probability p

s = cumsum(p)                  # s_i = ∑_{j≤i} p_j

pos = rand() * s[-1]           # [0, ∑_{j=1}^{N} p_j]

i = upper_bound(s, pos)        # min_i s.t. s_i > pos
```