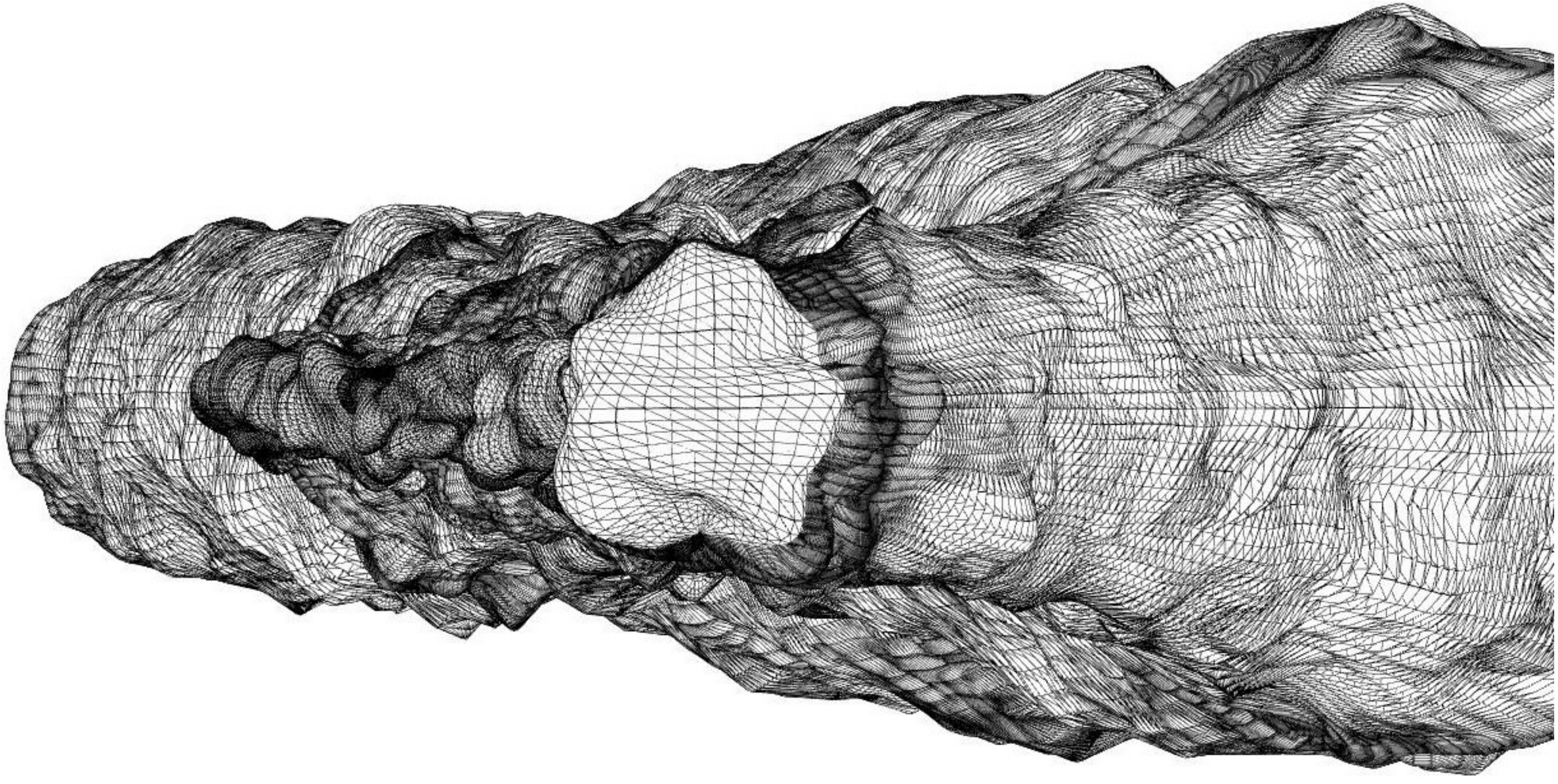
机器学习介绍

Xin Tao



日程

I. **直觉类算法部分：**

支持向量机（SVM）

II. **应用讨论**

III. **数学类讲解部分：**

逻辑回归

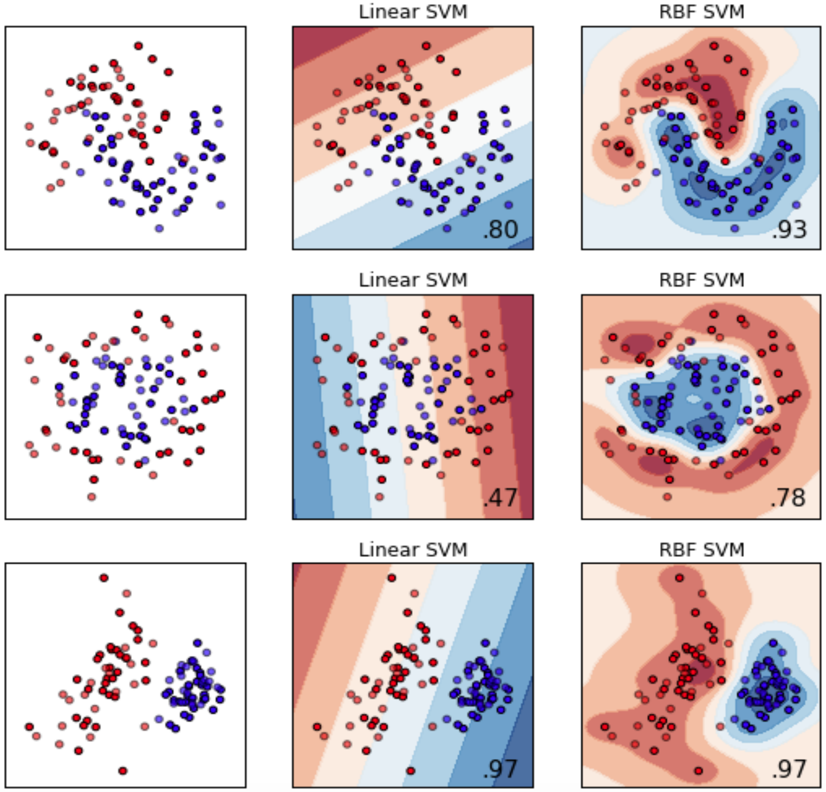
神经网络入门

支持向量机（SVM）

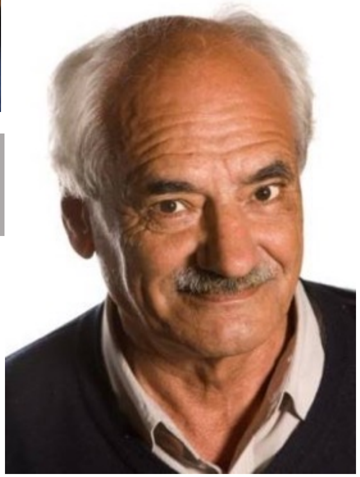
软件里面的所有问题都可以通过加一个中间层来解决。

Support Vector Machines

- 极为有效的分类算法，神经网络出现前最为强悍的分类算法。



弗拉基米尔 · 万普尼克

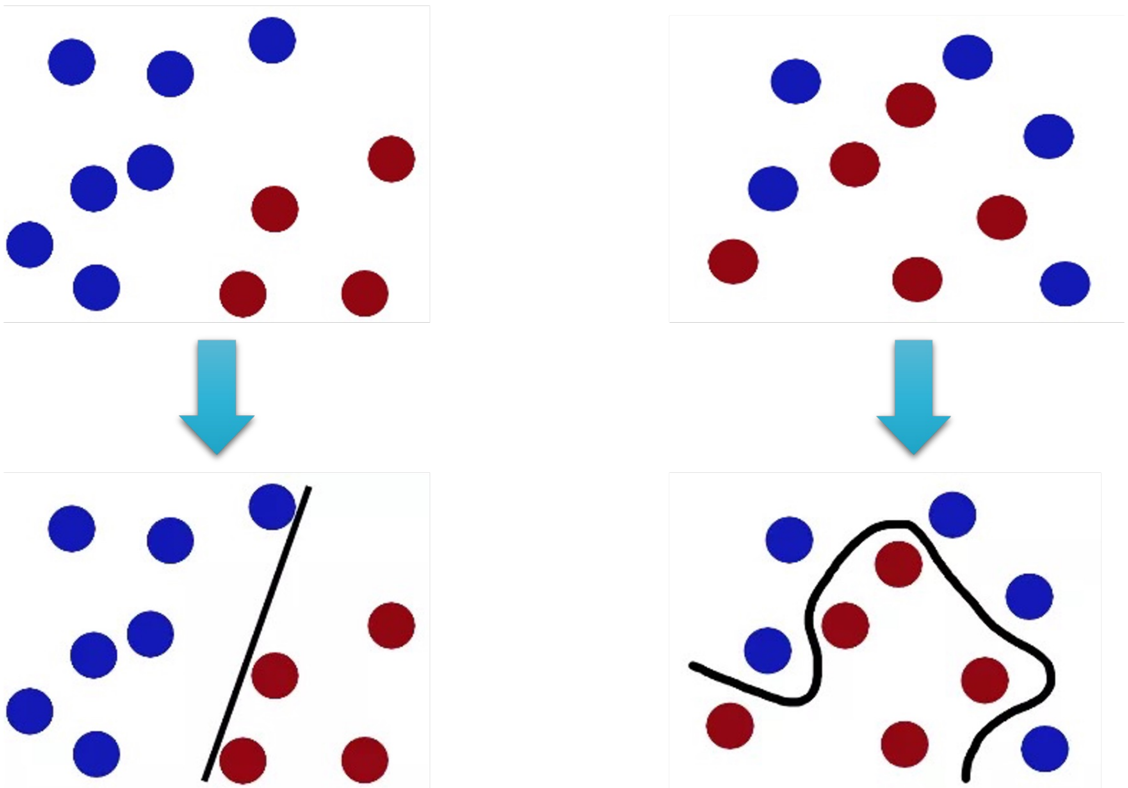


亚历克塞 · 泽范兰杰斯

支持向量机（SVM）

人以类聚，物以群分 *Support Vector Machines*

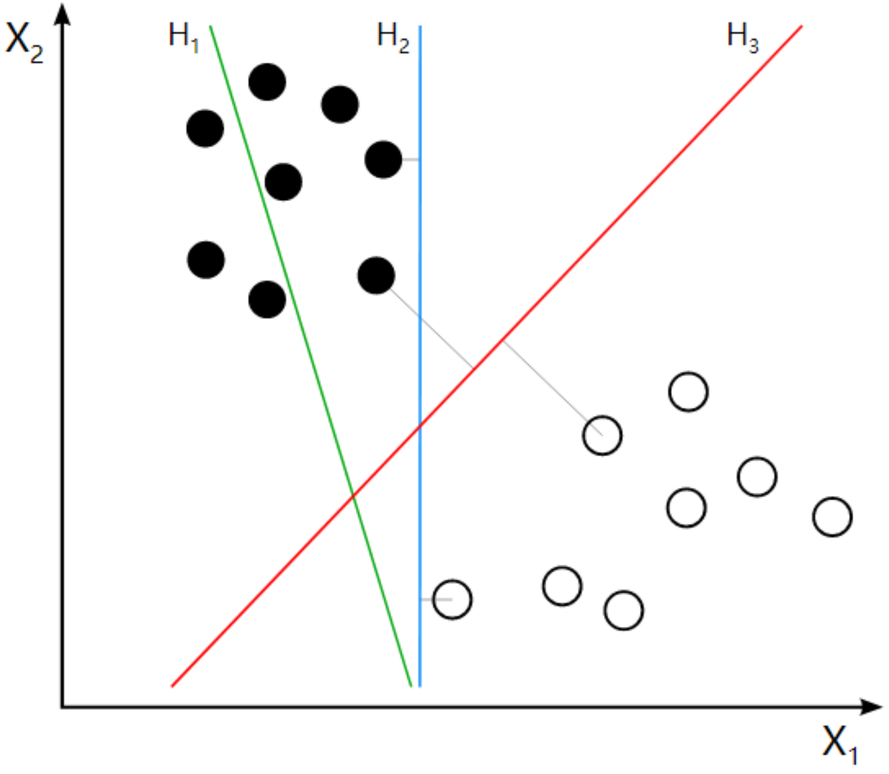
数据、信息的复杂程度决定了分类算法的复杂程度

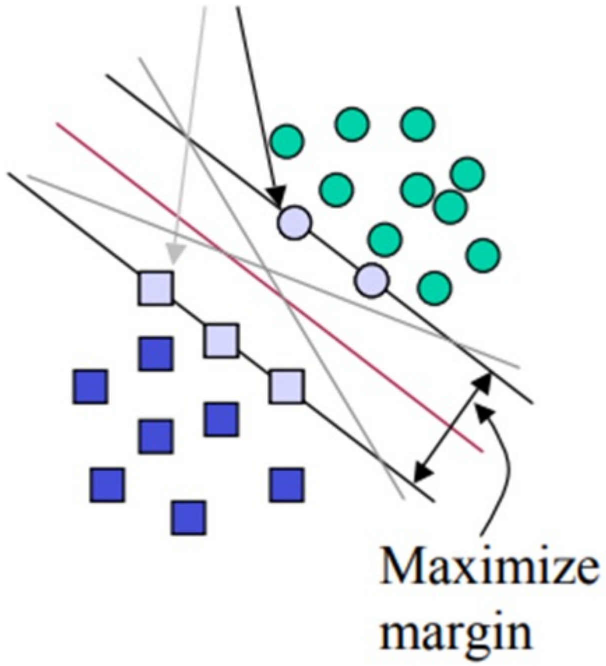


支持向量机（SVM）

分类算法并不只是要解决样本数据的分类，更重要的是在实际中解决问题 *Support Vector Machines*

支持点、边际、最大安全边际

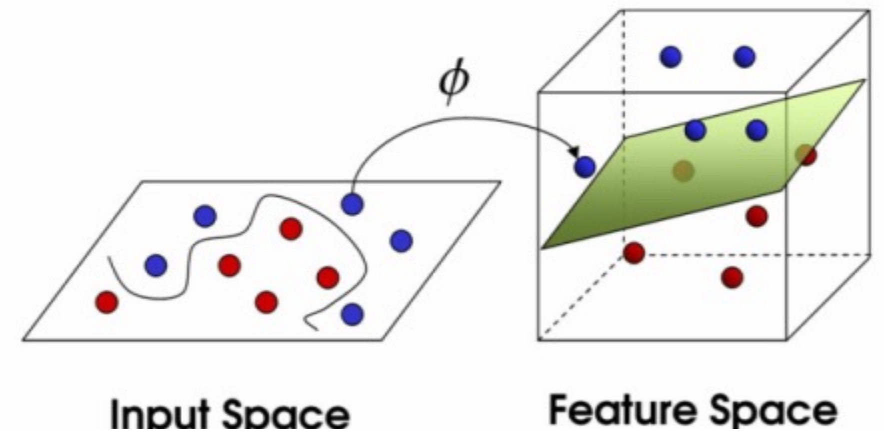


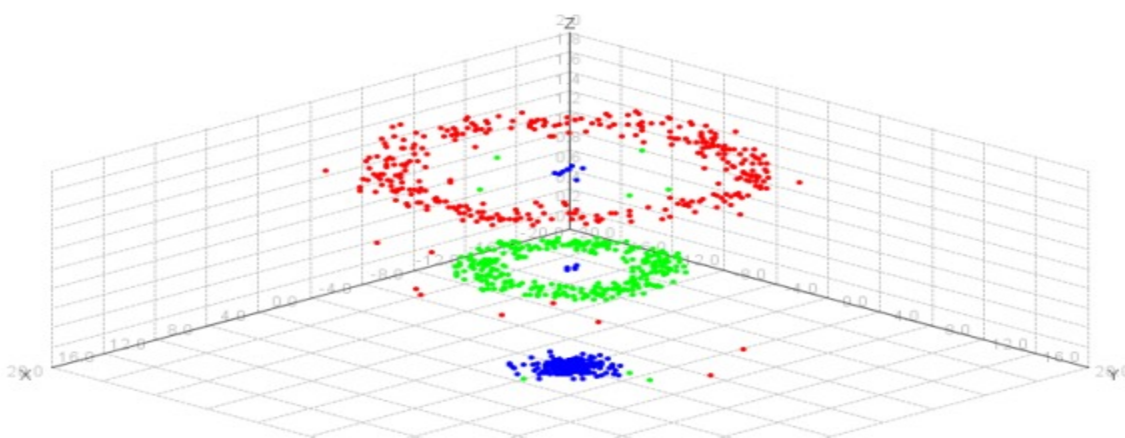


支持向量机（SVM）

将复杂的问题简单化以及将简单的问题复杂化 *Support Vector Machines*

核方法





支持向量机（SVM）

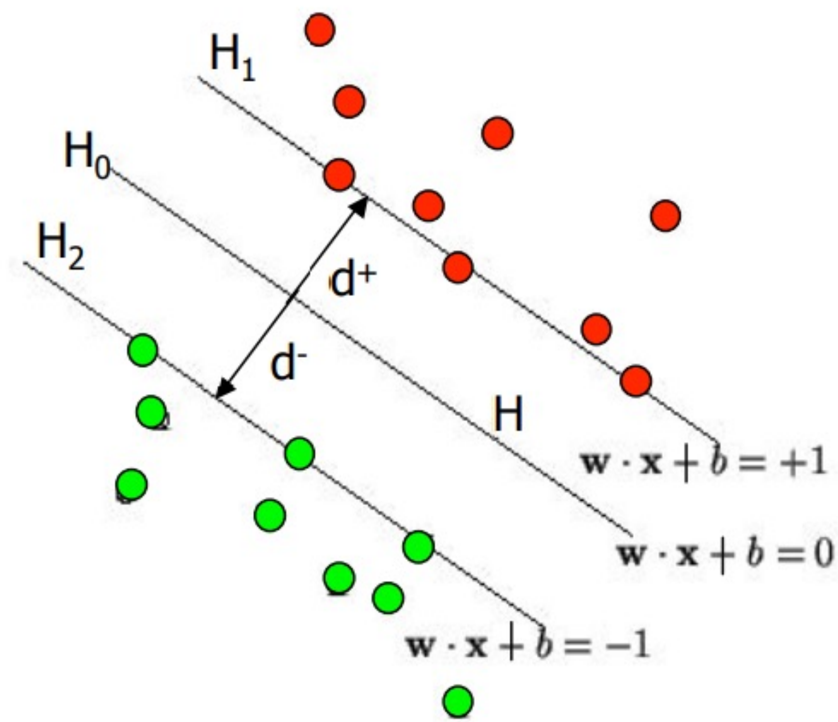
数学简易原理与过度拟合 *Support Vector Machines*

$$H_1: w \bullet x_i + b = +1$$
$$H_2: w \bullet x_i + b = -1$$

$$\mathbf{w}^T\mathbf{x} + b = 0$$

拉格朗日法

损失方程 $$min: \frac{1}{2}||\omega||^2$$
$$s.t. \quad y_i(wx_i + b) \geq 1$$



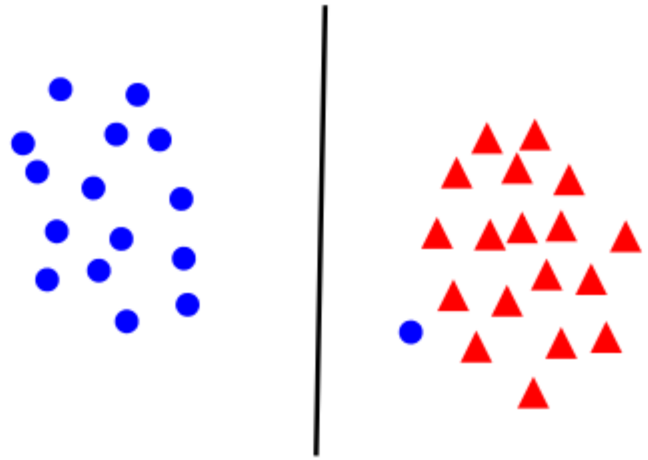
支持向量机（SVM）

至刚易折 Support Vector Machines

过度拟合：

- Hard Margin SVM：SVM原始形态；不允许错误，易过度拟合。
- Soft Margin SVM：基于Hard Margin SVM；允许错误，能对抗噪音的影响，相对不易过度拟合。

软分界SVM



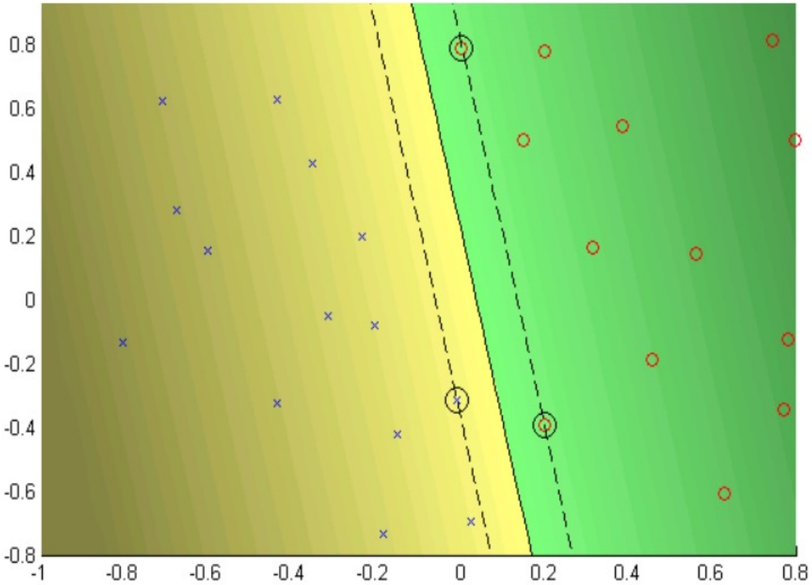
$$\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

$$s.t.\quad y_i\left(\mathbf{w}^\top\mathbf{x}_i + b\right) \geq 1 - \xi_i \text{ for } i = 1 \ldots N$$

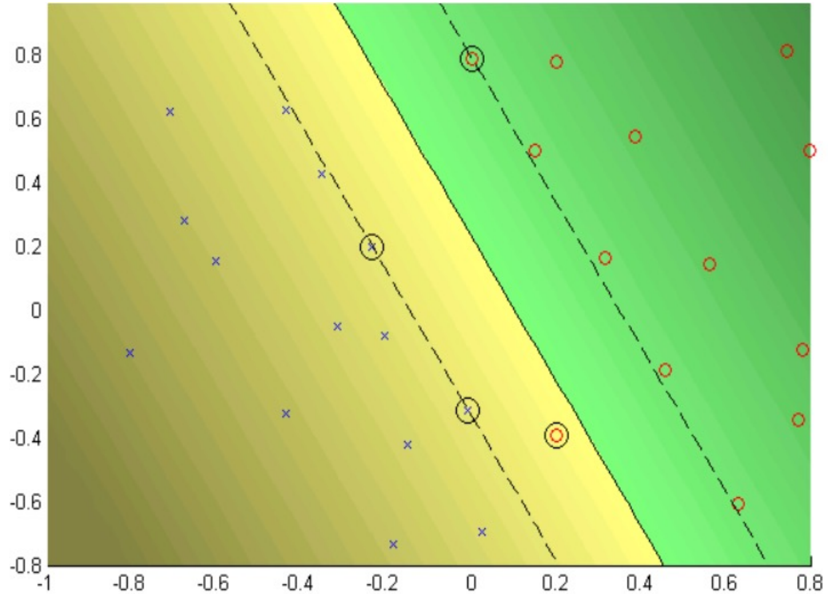
支持向量机（SVM）

鱼和熊掌不可兼得

Support Vector Machines



硬分界SVM：不允许错误



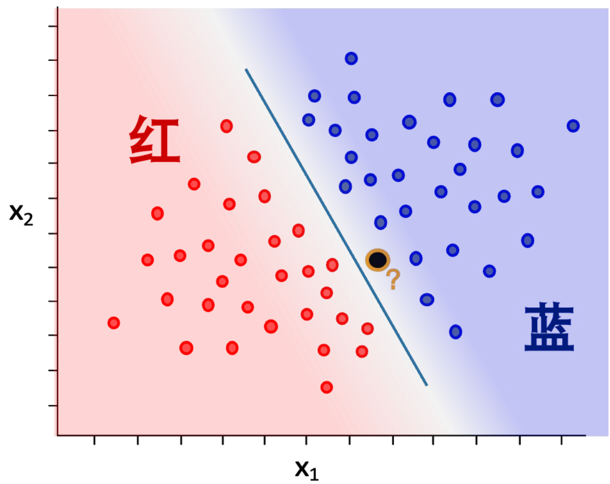
软分界SVM：允许错误

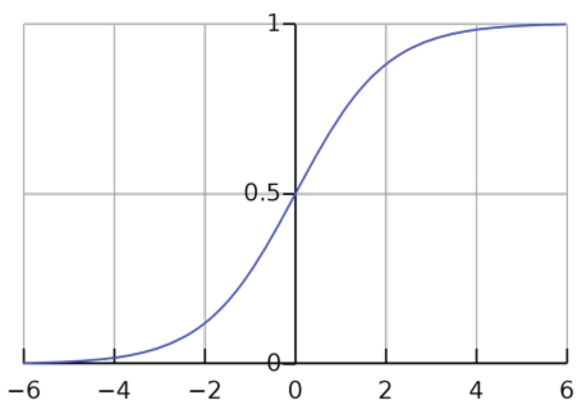
查准 VS 查全

应用讨论

逻辑回归

逻辑回归是一种分类算法，复合了两层算法





sigmoid方程

- 逻辑回归是极为常见的分类算法，尤其在信用分析和反欺诈领域。
- **目的**：**判断新样本**的标签（例：新顾客是否骗贷）。
- **方式**：用**线 (2维)\超平面（3维或以上）**将样本**空间分割**成不同标签归属部分。
- 逻辑回归方程**表达式**为：

$$f(x_1, x_2) = sig\left(l\left(x_1, x_2\right)\right) = \frac{1}{1 - e^{-(ax_1 - x_2 + b)}}$$

- 实际上由两个部分复合组成，一部分为**sigmoid方程** sig(z)，值域为(0,1), **模拟期望概率**：

$$sig\,(z) = \frac{1}{1 - e^{-z}}$$

- **sig(z)的机制**：当输入z超过0后，sig(z)的值 (期望概率) 迅速从近似0上升到近似1。

逻辑回归

输入坐标、转化成位置标量、再转化成概率。

- 另一部分是线性表达式 $l(x_1,x_2)$，表达式为：

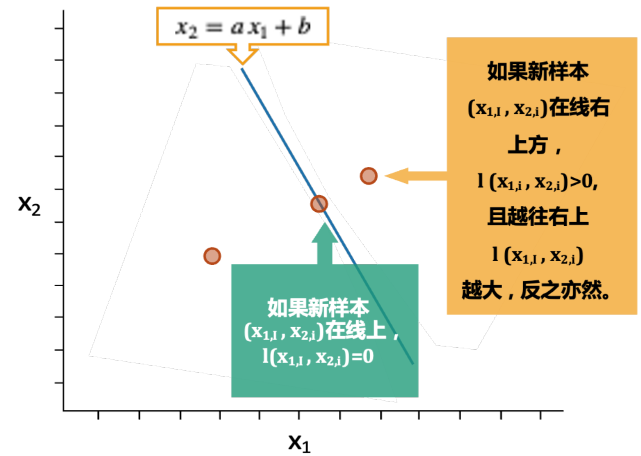
$$l\left(x_1, x_2\right) = a\,x_1 - x_2 + b$$

- $l(x_1,x_2)$表达式包括了 线性方程 $x_2=ax_1+b$, 但本身不代表一条直线。作用为判断样本位置在线右上（输出正值）或左下（输出负值），具体见右上图。

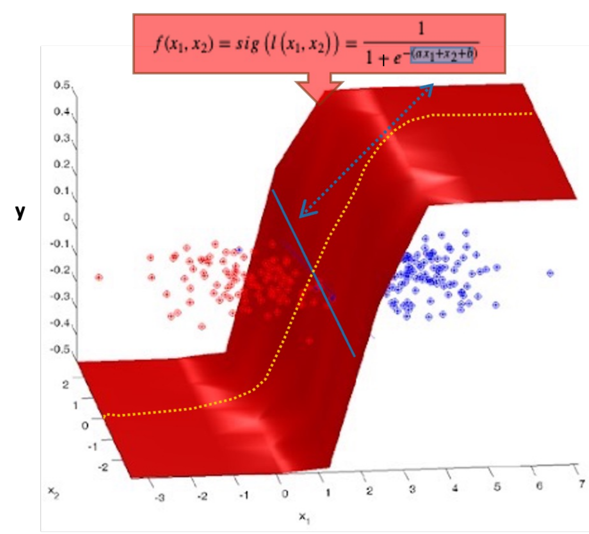
- **因此，逻辑回归的整体工作机制可以看成：第一步 $l(x_1,x_2)$ 将样本坐标转化为表达其相对直线 $x_2= ax_1+b$位置的数值（正或者负）；第二步 sig(z) 将该数值转化为概率。**

- 例子：若直线 $x_2= ax_1+b$ 右边为蓝色，且样本 $(x_{1,1},x_{2,1})$在直线右上 $l(x_{1,1},x_{2,1})$将输出一个正值，sig($l(x_{1,1},x_{2,1})$)将输出一个0.5至1.0之间的值，例如 0.9，可以理解为**“样本为蓝色的概率为90%”**。

$$p(y_i = blue) = sig\left(l\left(x_1,i, x_2,i\right)\right) = \frac{1}{1+e^{-(ax_1,i-x_2,i+b)}}$$



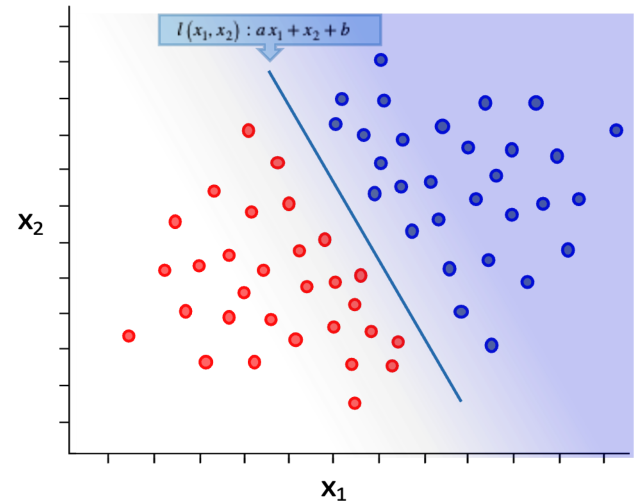
l(x1,x2)方程



逻辑回归训练：损失方程

逻辑回归使用负对数似然损失方程

- **并不是任意一条直线都可以正确分开样本数据。**
- 训练的主要目的寻找正确的线，或者说正确的$l(x_1,x_2)$参数。
- 在此之前，需要设计计**正确的损失方程**，以评价训练效果。
- 假设样本数据 $x_{1,i},x_{2,i}$ 真实标签为 y_i（0或1），模型估计预估样本数据标签为1的概率为$\mu_i=sig(\,l(x_{1,i},x_{2,i}))$，直线参数为 w (即a,b等等)



- 让负对数似然NLL损失方程为：

$$NLL(w) = -\sum_{i=1}^{N} log\left[\mu_i^{\delta_{y_i,1}} \times (1-\mu_i)^{\delta_{y_i,0}}\right]^{*} = -\sum_{i=1}^{N}\left[y_i log \mu_i + (1-y_i) log(1-\mu_i)\right]$$

模型个例判断效果

- ☐ 正确样本判断1：真实标签y_1=1,模型判断μ_1=0.9，模型个例判断部分约等于0, NLL损失不变。
- ☐ 正确样本判断2：真实标签y_2=0,模型判断μ_2=0.1，模型个例判断部分约等于0，NLL损失不变。
- ☐ 错误样本判断3：真实标签y_3=1,模型判断μ_3=0.05，模型个例判断部分约等于-1，NLL损失上升。
- ☐ 错误样本判断4：真实标签y_4=0,模型判断μ_4=0.95，，模型个例判断部分约等于-1，NLL损失上升。

* $\delta_{a,b}$为克罗内克方程，当a=b时为1，否则为0

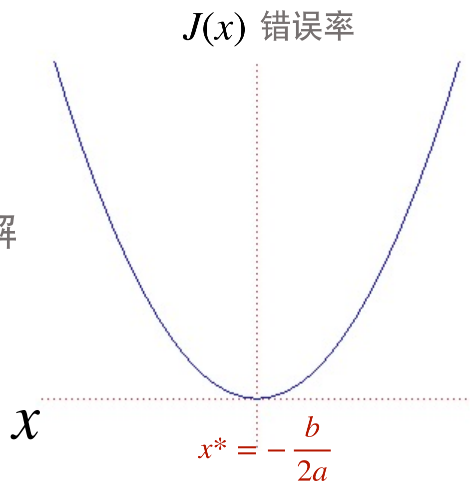
机器学习核心思想1: 迭代优化

回顾：梯度下降法

先开始、再一步一步来。

简单问题(一元二次方程极小值)

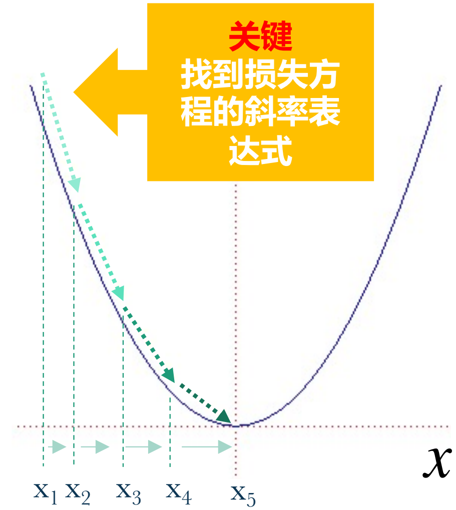
公式解



机器学习训练设计的核心问题:

1. 怎么设立一种机制让模型、参数一次比一次更准确，最快接近目标？

2. 怎么利用机器算力实现快速迭代？



机器学习:

梯度下降法

Gradient descent

利用斜率信息一步一步接近极值

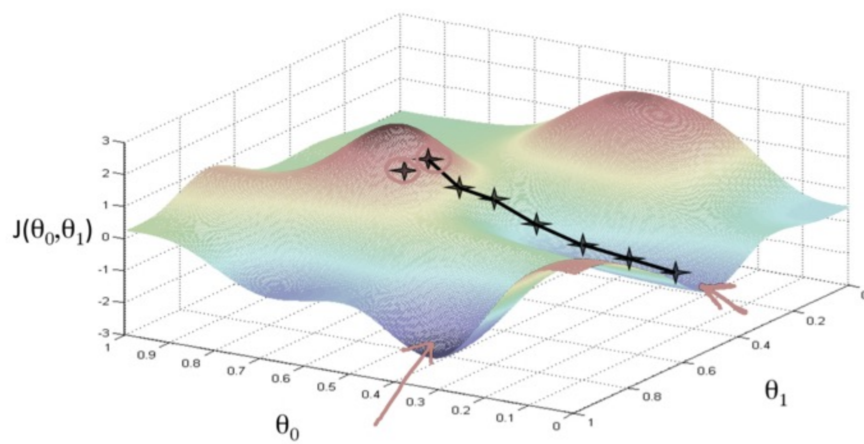
复杂问题

公式解:

可能不存在

机器学习:

梯度下降法逼近可能找到最优或次优解



逻辑回归训练：梯度下降优化

梯度下降需要对损失函数求微分

- 通过调整线性参数w，最小化NLL即可找到最适合的分割线。
- 逻辑回归NLL表达式不适合求解析解，使用 **梯度下降法** 寻找最优w。
- μ_i为sigmoid方程，sigmoid方程导数有如下性质：

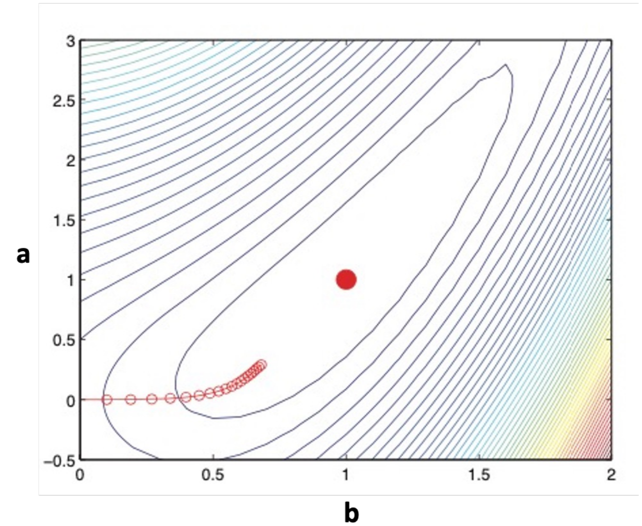
$$\frac{\mu(z)}{dz} = \mu(z)\big(1-\mu(z)\big)$$

- 因此NLL导数可以简化为：

$$g_{NLL} = \frac{dNLL(w)}{dw} = -\sum_{i=1}^{N}\left[\frac{y_i}{\mu_i}\frac{d\mu_i}{dw} - \frac{1-y_i}{1-\mu_i}\frac{d\mu_i}{dw}\right] = -\sum_{i=1}^{N}\left[\frac{y_i-\mu_i}{\mu_i(1-\mu_i)}\right]\frac{d\mu_i}{dw} = -\sum_{i=1}^{N}(y_i-\mu_i)\frac{d\mu_i}{dw}$$

$$= \sum_{i=1}^{N} x_i(\mu_i - y_i) = X(\mu - Y)$$

向量表达式、非常适合计算机计算



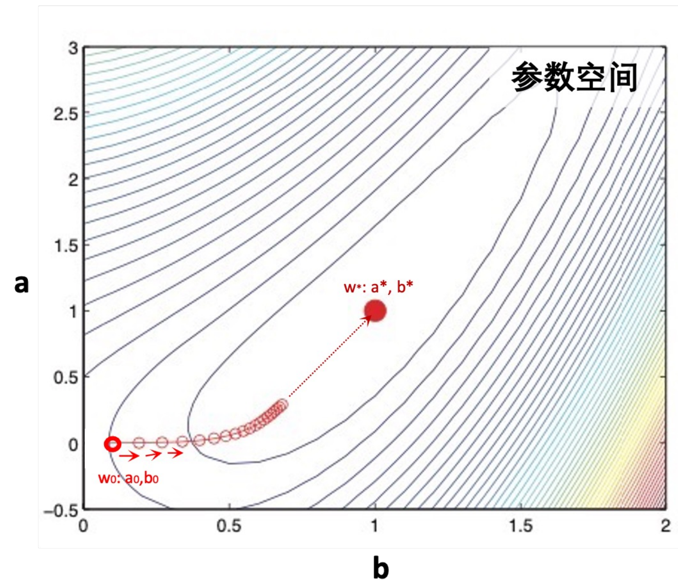
逻辑回归训练：梯度下降优化

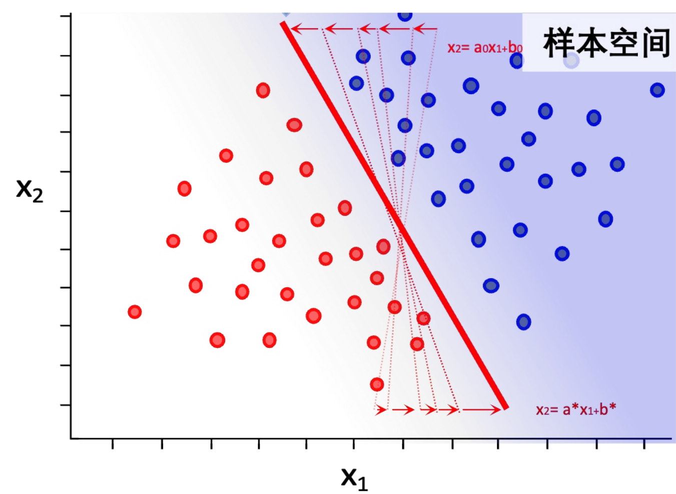
CyberFin

算法计算过程总结

梯度下降过程：

1. 随机设定初始w_0
2. 设定学习速率L
3. $w_1 = w_0 - L * g_{\mathrm{NLL}}(w_0)$
4. $w_{\mathrm{i+1}} = w_{\mathrm{i}} - L * g_{\mathrm{NLL}}(w_{\mathrm{i}})$
5. 重复第4步，直至NLL(w_i)-NLL(w_{i-1}) 接近于某个近似于零的预定阈值。
6. 最终模型参数为w*
7. **最终模型 f(x)=sig(l(x|w*)), 输入样本坐标数据x，即可得到x标签为1的预期概率。**

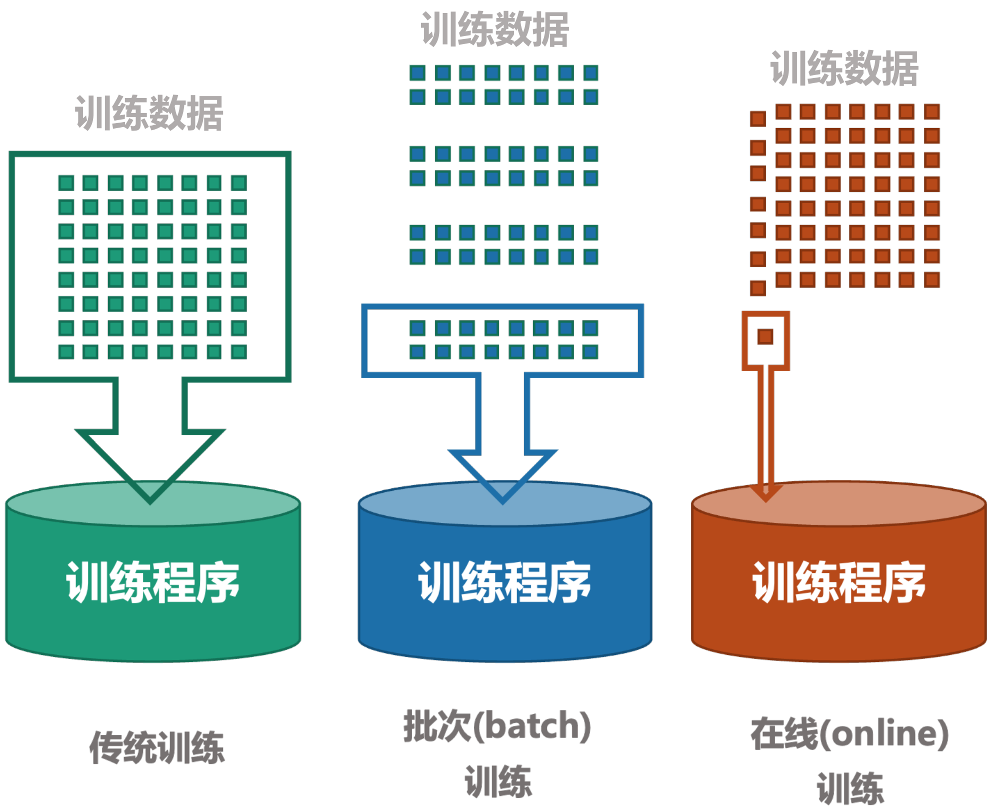




** 优化方法的深入讨论，包括随机梯度下降和牛顿法，请见MLAPP 8.3.2-8.3.7*

逻辑回归训练：训练方式

训练数据可以用不同方式喂给训练程序

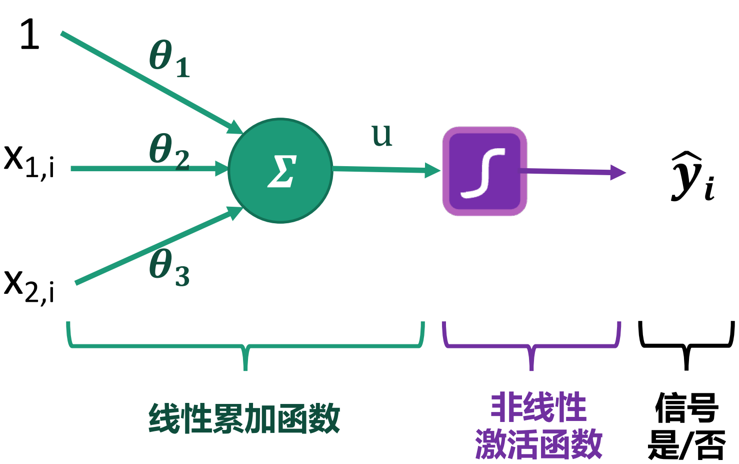


训练方式的选择由数据产生方式、程序处理能力、其它要求等综合因素考量决定。

神经网络入门：神经元

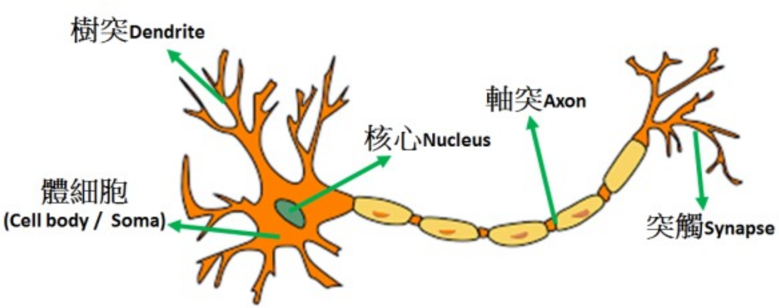
逻辑回归可以看成数字化的一种神经元

逻辑回归模型的另一种表达方式：



根据输入和线性系数输出1或0的信号

神经元：



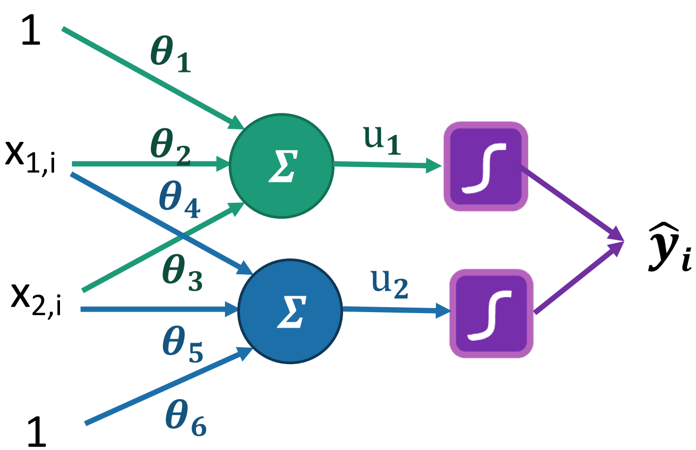
根据输入刺激强度决定是否激活发出电/化学信号

神经网络入门：神经网络

CyberFin

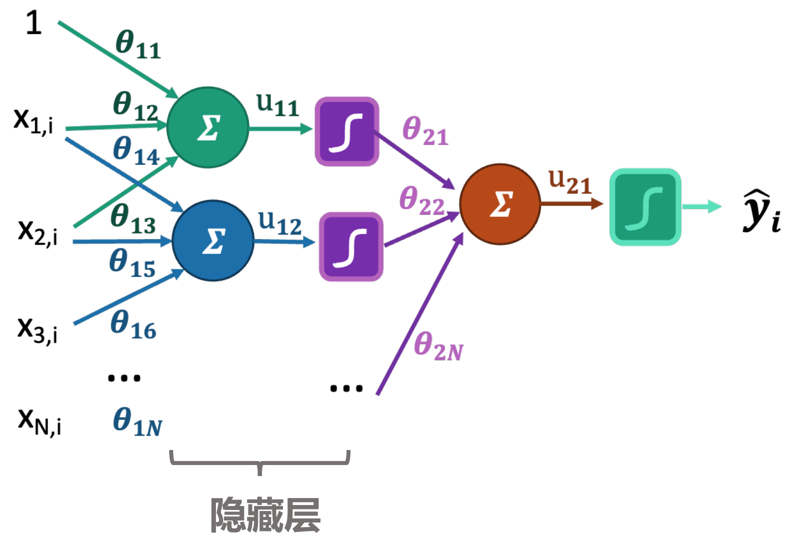
神经元的联合可以产生不同高级机器学习算法

更为复杂的情况：多标签逻辑回归



根据输入和线性系数输出 (1,1)，(1,0)，(0,1)，(0,0)的信号

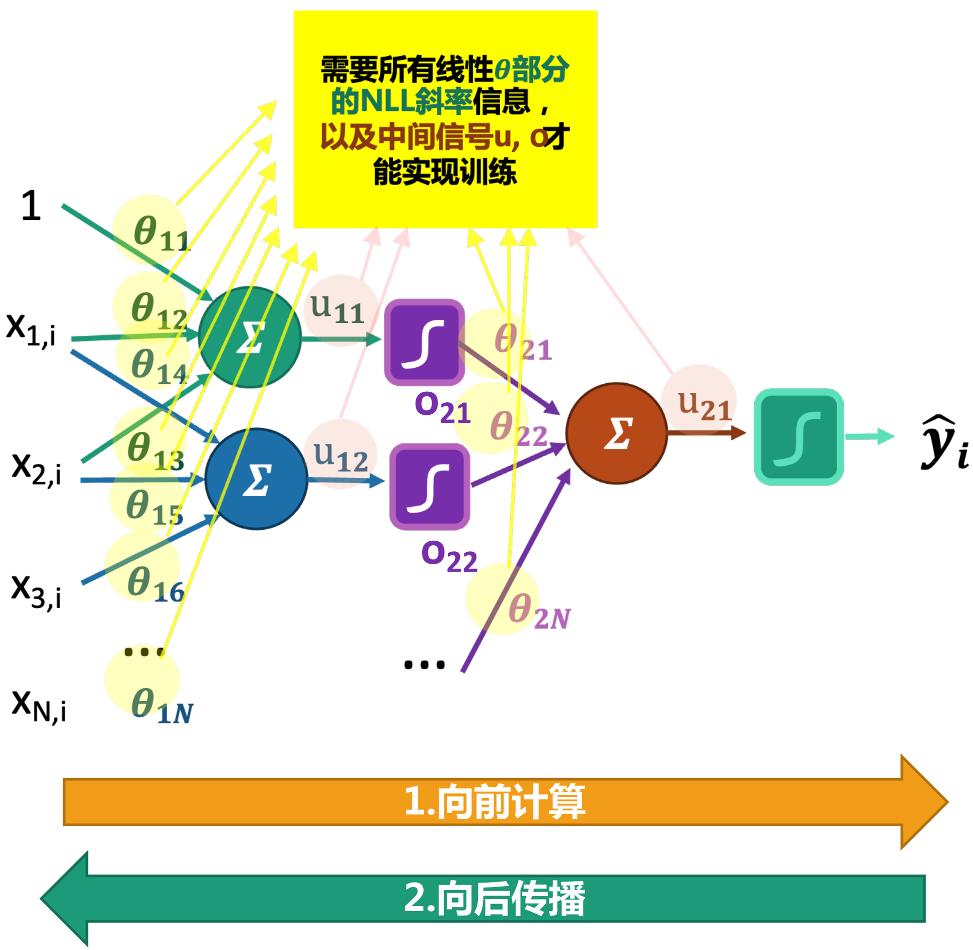
多层次神经网络（2层）：



高纬度输入，非线性决策边界，输出1或0的信号（神经元越多、层数越多决策边界可以越复杂）。

神经网络入门：训练

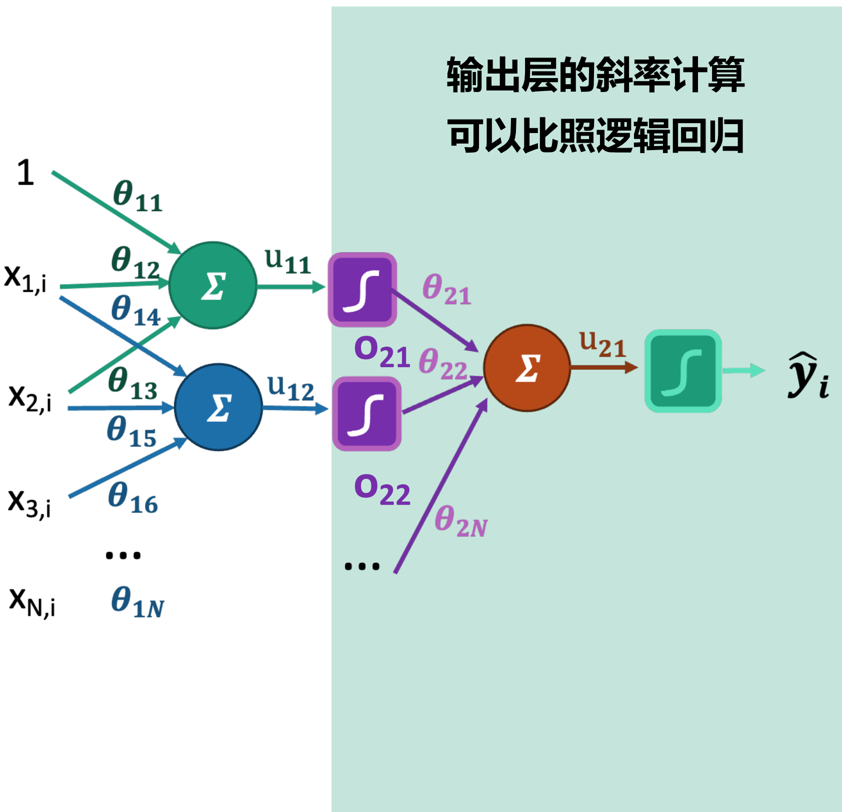
训练神经网络需要向前计算和向后传播



- **神经网络NLL和逻辑回归没有本质不同，也主要依靠梯度下降训练。**
- 每轮训练前：
  - ◻ 我们知道各个θ的初始值。
  - ◻ 我们知道真实标签**y**的值。
  - ◻ 未知中间值**u**、**o**的值。
  - ◻ 未知模型标签估计$\hat{y}$的值。
- 为了进行训练调整参数我们需要上述所有的值。
- 为此我们每次需要进行两轮计算：
  - ① **向前计算**：输入x计算出**u**、**o**、$\hat{y}$的值。
  - ② **向后传播**：反向计算出θ的斜率信息和需要调整量。

神经网络入门：向后传播（输出层）

向前计算比较直接，不做过多解释。训练需要两个层面的斜率信息g_1和g_2，从最后一层开始。



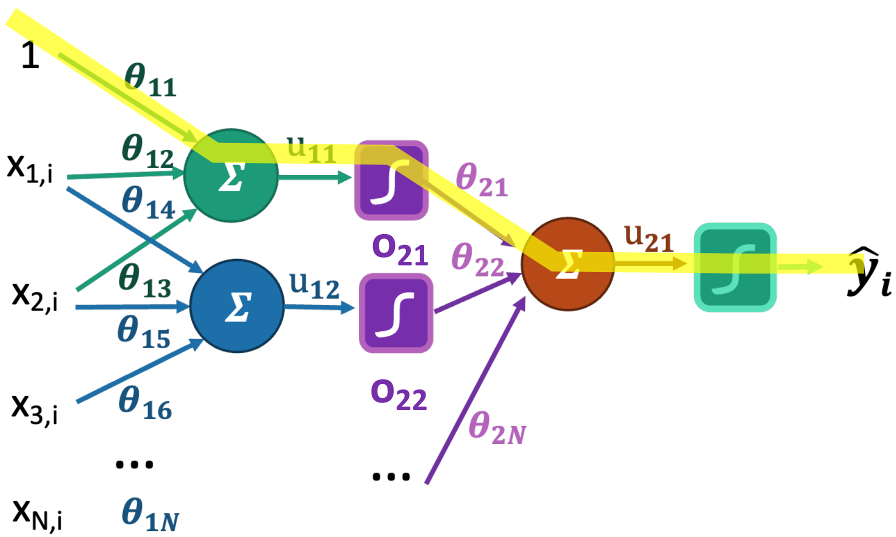
输出层即$\hat{y}_i$

$$g_{NLL} = \frac{dNLL(w)}{dw} = \sum_{i=1}^{N} x_i(\mu_i - y_i) = X(\mu - Y)$$

输出层即o_{2i}

即 $$g_2 = \frac{dNLL}{d\theta_2} = \sum_{i=1}^{N} o_i(\hat{y}_i - y_i)$$

神经网络入门：向后传播（隐藏层）



隐藏层的斜率主要利用求导链式法则求出，并利用向前计算步骤计算出的中间量。

- 线性层的导数就是参数本身
- 激活层导数形式取决于激活函数本身，sigmoid仅仅是其中一种。
- **利用斜率信息即可迭代式调整参数。**

$$g_{11} = \frac{dNLL}{d\theta_{11}} = \frac{dNLL}{d\theta_{21}}\frac{d\theta_{21}}{d\theta_{11}} = -\sum_{i=1}^{N}(y_i-\hat{y}_i)\frac{d\hat{y}_{21}}{d\theta_{21}}^{*}\frac{d\theta_{21}}{d\theta_{11}} = -\sum_{i=1}^{N}(y_i-\hat{y}_i)\underbrace{\frac{d\hat{y}_{21}}{du_{21}}}_{\text{Sigmoid 方程求导}}\overbrace{\frac{du_{21}}{do_{21}}}^{\theta_2}\underbrace{\frac{do_{21}}{d\theta_{11}}}_{\text{逻辑回归求导}}$$

= … (具体推导省略) …

* $g_{NLL} = \frac{dNLL(w)}{dw} = -\sum_{i=1}^{N}\left[\frac{y_i}{\mu_i}\frac{d\mu_i}{dw} - \frac{1-y_i}{1-\mu_i}\frac{d\mu_i}{dw}\right] = -\sum_{i=1}^{N}\left[\frac{y_i-\mu_i}{\mu_i(1-\mu_i)}\right]\frac{d\mu_i}{dw} = -\sum_{i=1}^{N}(y_i-\mu_i)\frac{d\mu_i}{dw}$