《神经网络与深度学习》

各损失函数的理论基础

马千里

计算机科学与工程学院

上次课，我们知道：

- 对逻辑回归，正确的损失函数应该是交叉熵损失函数
- 那么……
- 这又是为什么？为什么说逻辑回归用交叉熵损失函数是正确的？合理性在哪里？是必然的吗？

一切从极大似然(Maximum Likelihood)开始说起:各损失函数的理论基础

- 什么是极大似然?
- 它是如何导致误差平方和（Sum Squared Error, SSE)是线性回归的损失函数?
- 它是如何导致交叉熵是逻辑回归的损失函数?
- 它是如何导致交叉熵是多分类模型的损失函数?
- 总结

内容

- *什么是极大似然?*
- 它是如何导致误差平方和（Sum Squared Error, SSE)是线性回归的损失函数?
- 它是如何导致交叉熵是逻辑回归的损失函数?
- 它是如何导致交叉熵是多分类模型的损失函数
- 总结

极大似然

- 主要思想:
- 给定数据D，模型中哪些参数W最有可能产生这种数据D?
- 也就是说，我们希望参数W是代表最大化的P(W | D).
- 这是很困难的一个问题.
- 幸好我们有贝叶斯定理:

$$P(W \mid D) = \frac{P(D \mid W)P(W)}{P(D)}$$

极大似然

- 贝叶斯定理:

$$P(W \mid D) = \frac{P(D \mid W)P(W)}{P(D)} = \frac{\text{Likelihood*Prior}}{\text{normalizing constant}}$$

- 注意到:
  - 数据的概率P(D)是标准化常量.
  - 对P(W)，我们没有理由，先验地假设一些W比另一些好，所以我们假设，先验是平坦的，即所有W的概率都是相等的.
  - 因此对 P(W)，这个先验,可以看做常数.

极大似然

- 因此，对于一个均匀分布的先验(*uniform prior*):

$$P(W \mid D) = \frac{P(D \mid W)P(W)}{P(D)} = \frac{\text{Likelihood*Prior}}{\text{normalizing constant}}$$

$$\max_W P(W \mid D) = \max_W \frac{P(D \mid W)P(W)}{P(D)} = \max_W P(D \mid W)$$

- 由此得名：*Maximum Likelihood.*
- 具体来说，我们希望模型（神经网络参数W）能尽可能地使观察到的数据成为可能.
- 如何<u>对数据分布建模</u>就成为了关键。

极大似然

- 举例: 高斯分布(Gaussian distributions)

$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- 假设数据点都是独立同分布的（independently identically distributed , iid), 则数据的似然函数:

$$L = \prod_{n=1}^{N} p(x^n) = \frac{1}{\sqrt{2\pi\sigma^2}^N} \prod_{p=1}^{N} e^{-\frac{(x^n-\mu)^2}{2\sigma^2}}$$

极大似然

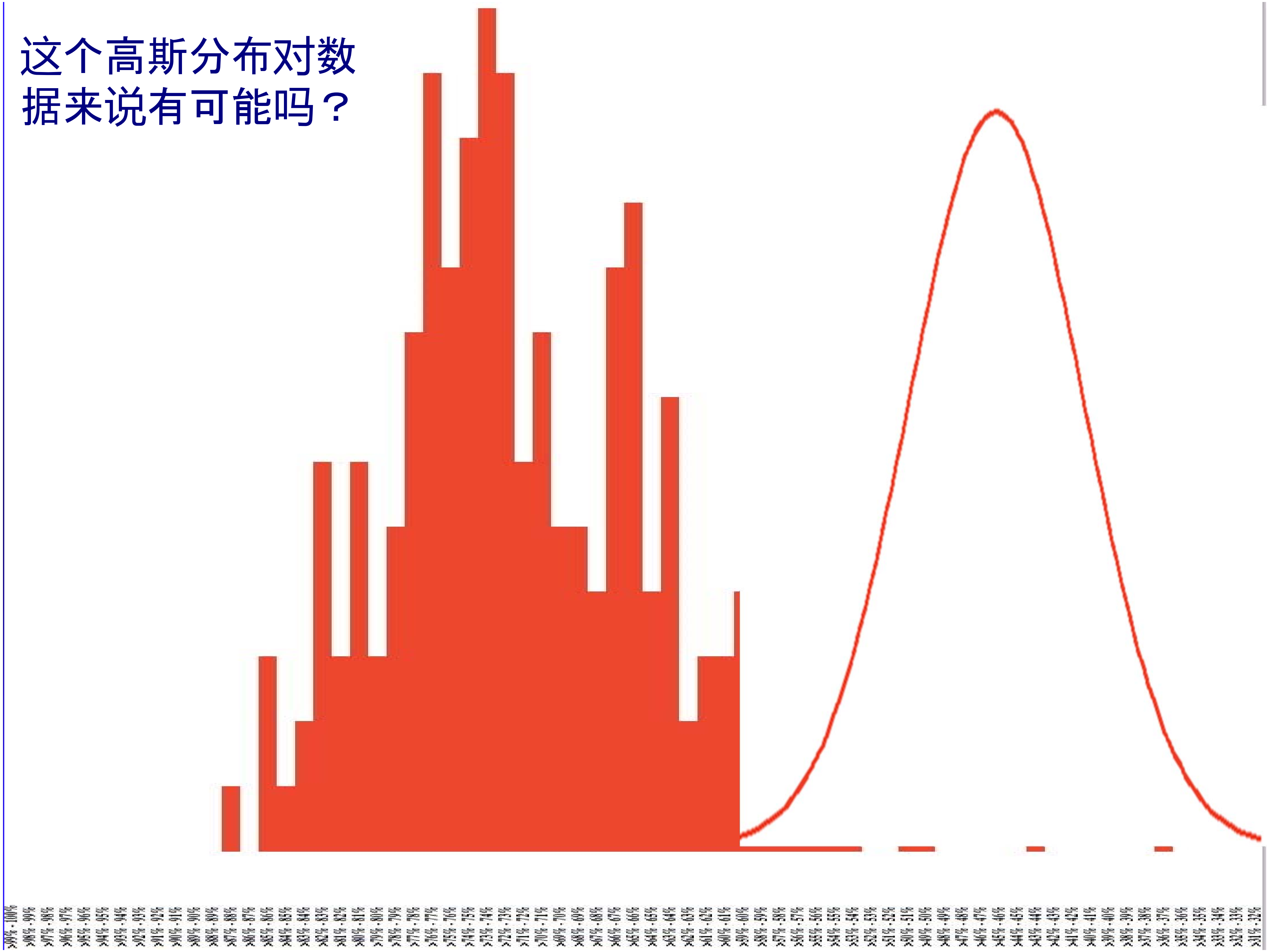
- 最终，极大似然要求我们选择 μ 和 σ ，使得:

$$(\mu,\sigma)=\arg\max_{\mu,\sigma} L$$
$$=\arg\max_{\mu,\sigma}\prod_{n=1}^{N}p(x^n)=\arg\max_{\mu,\sigma}\frac{1}{\sqrt{2\pi\sigma^2}^{N}}\prod_{p=1}^{N}e^{-\frac{(x^n-\mu)^2}{2\sigma^2}}$$

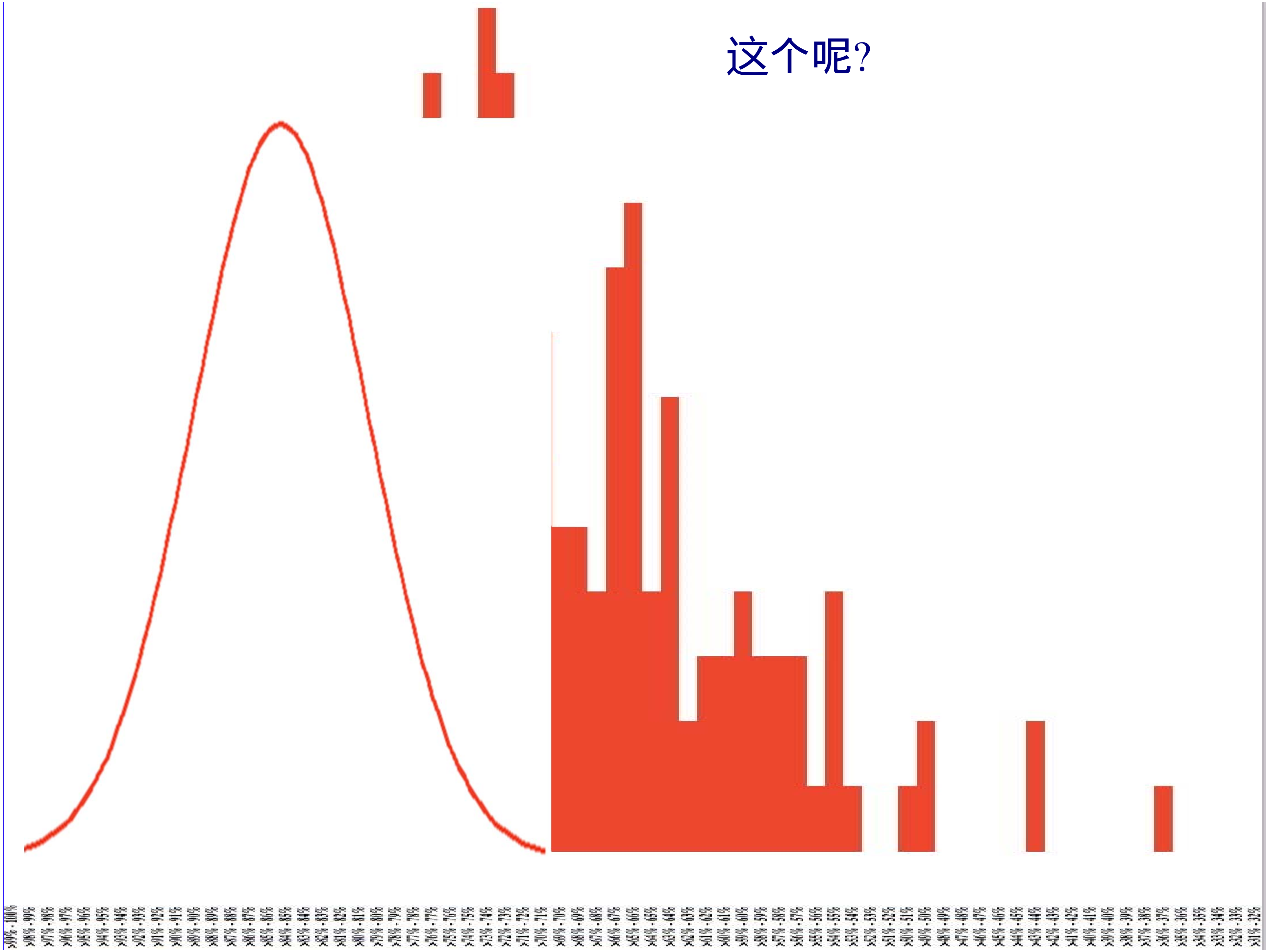
- 选择参数，最大化似然函数，也就是说，我们应该选择最大化生成这些数据可能性的参数。

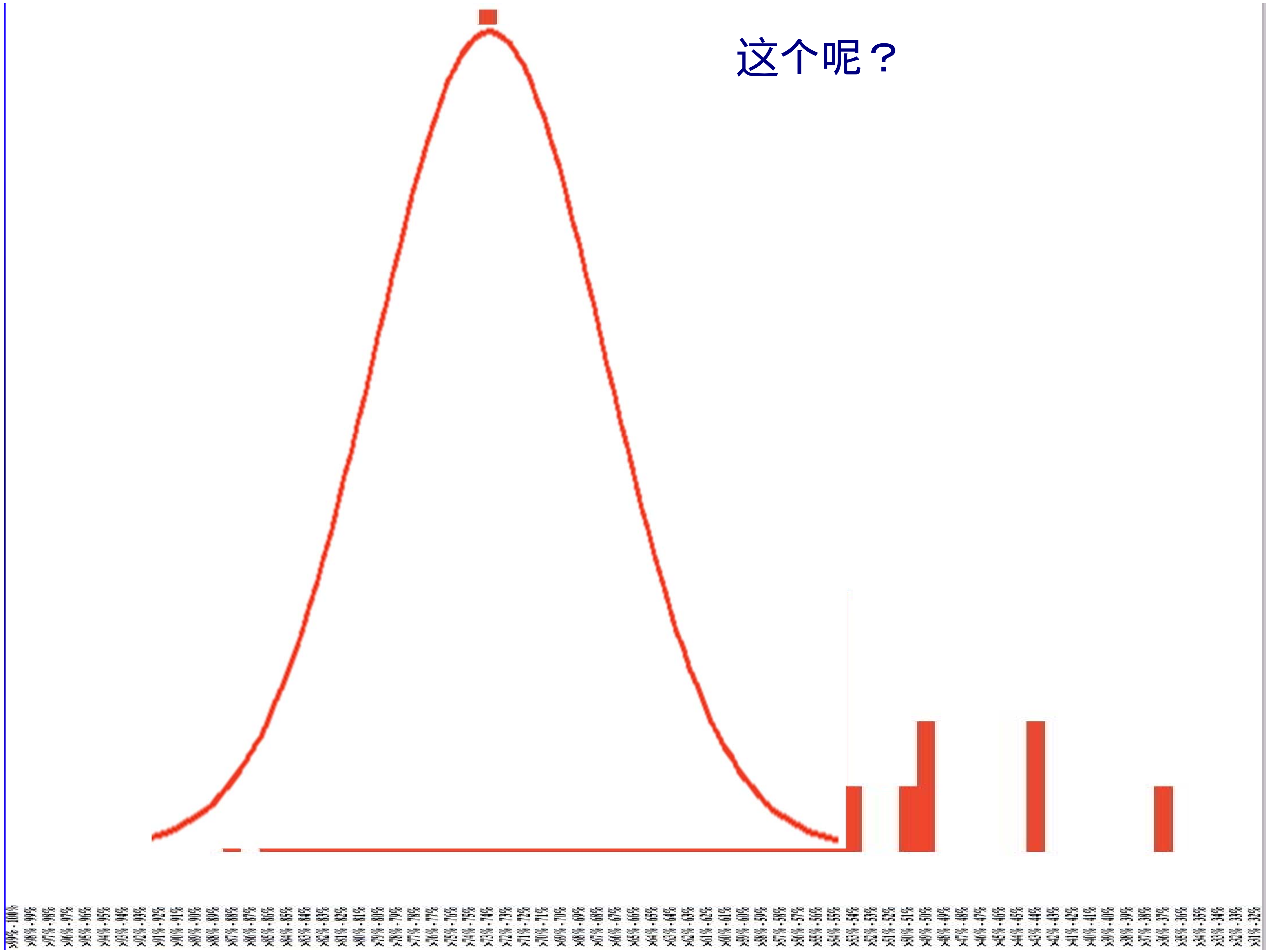
- 那么，如何做到？

这个高斯分布对数据来说有可能吗？



这个呢?




这个呢？

如何找到极大似然的参数?

- 通常，最小化负对数似然更容易，比如，

$$(\mu,\sigma) = \arg\max_{\mu,\sigma} \ln \prod_{n=1}^{N} p(x^n)$$

$$= \arg\min_{\mu,\sigma} \; -\ln \prod_{n=1}^{N} p(x^n)$$

$$= \arg\min_{\mu,\sigma} \; -\sum_{n=1}^{N} \ln p(x^n)$$

$$= \arg\min_{\mu,\sigma} \; -\sum_{n=1}^{N} \frac{-(x^n-\mu)^2}{2\sigma^2} - N \ln \sqrt{2\pi\sigma^2}$$

- 最后一行结果代表如果我们假设是高斯分布

如何找到极大似然的参数?

- 术语名称:

$$L = \prod_{n=1}^{N} p(x^n)$$ "Likelihood" “似然函数”

$$\ln\ L = \ln \prod_{n=1}^{N} p(x^n)$$ "Log Likelihood" “对数似然函数”

$$-\ln L = -\ln \prod_{n=1}^{N} p(x^n)$$ "Error" “负对数似然函数”

- 最后的负对数似然也称作误差.

内容

- 什么是极大似然?
- ***它是如何导致误差平方和（Sum Squared Error, SSE)是线性回归的损失函数?***
- 它是如何导致交叉熵是逻辑回归的损失函数?
- 它是如何导致交叉熵是多分类模型的损失函数?
- 总结

建模“输入-输出”数据

- 在之前的内容中，我们只是假设一维数据。
- 在神经网络(和回归)的应用中，我们有输入和输出数据，则似然函数变为:

$$L = \prod_{n=1}^{N} p(x^n, t^n) = \prod_{n=1}^{N} \underbrace{p(t^n}_{\text{目标(输出)数据}} \mid \underbrace{x^n}_{\text{输入数据}}) p(x^n)$$

- 同样取负对数，得到:

$$-\ln L = -\sum_{n=1}^{N} \left(\ln p(t^n \mid x^n) + \ln p(x^n) \right)$$

建模“输入-输出”数据

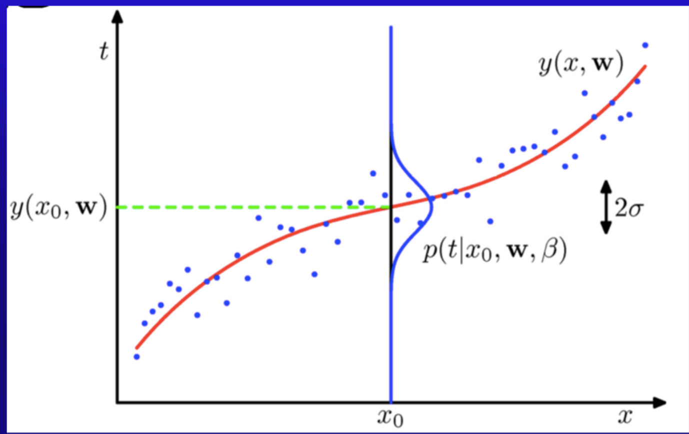
$$-\ln L = -\sum_{n=1}^{N}\left(\ln p(t^n \mid x^n) + \ln p(x^n)\right)$$

- 由于是对从x到t的映射建模，求解关于参数的上式最小化时，第二项与参数无关，所以我们可以不考虑它。

极大似然与线性回归

- 主要思想: 假设目标数据服从高斯分布
- 换句话说:

$$p(t^n \mid x^n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t^n - h(x^n))^2}{2\sigma^2}}$$



- 在这里,我们假设有一个潜在的确定性函数h(红线),和一些满足0均值的高斯噪声ε:

$$t^n = h(x^n) + \varepsilon \quad 或 \quad \varepsilon = t^n - h(x^n)$$

极大似然与线性回归

因此,在所有数据上的似然函数为:

$$L = \prod_{n=1}^{N} p(t^n \mid x^n) = \frac{1}{\sqrt{2\pi\sigma^2}^N} \prod_{p=1}^{N} e^{-\frac{(t^n - y(x^n;w))^2}{2\sigma^2}}$$

这里，我们用y(x^n;w)代替刚才的确定性函数h(x^n) (这种写法强调了我们的模型（神经网络）是由权重w来进行参数化的.)

则负对数似然或误差就写成:

$$E = -\ln \frac{1}{\sqrt{2\pi\sigma^2}^N} \prod_{p=1}^{N} e^{-\frac{(t^n - y(x^n;w))^2}{2\sigma^2}}$$

极大似然与线性回归

负对数似然或误差就写成:

$$E = -\ln \frac{1}{\sqrt{2\pi\sigma^2}^N} \prod_{p=1}^{N} e^{-\frac{(t^n - y(x^n;w))^2}{2\sigma^2}}$$

$$= \frac{1}{2\sigma^2} \sum_{n=1}^{N} (t^n - y(x^n;w))^2 + \ln\left(\sqrt{2\pi\sigma^2}^N\right)$$

由于第二项和权重w无关,可以去掉,另外$1/2\sigma^2$也对最小化没影响,所以我们得到:

$$\sum_{n=1}^{N} (t^n - y(x^n;w))^2$$

误差平方和

极大似然与线性回归

小结一下:

在线性回归中，如果

1) 假设目标数据满足高斯分布，并且

2) 通过最小化数据的负对数似然来最大化数据的似然（概率），

那么我们发现实际上需要最小化误差平方和SSE!

内容

- 什么是极大似然?
- 它是如何导致误差平方和（Sum Squared Error, SSE)是线性回归的损失函数?
- ***它是如何导致交叉熵是逻辑回归的损失函数?***
- 它是如何导致交叉熵是多分类模型的损失函数?
- 总结

极大似然与逻辑回归

- 逻辑回归就是对0/1二分类问题建模
- 在这里,我们希望模型（神经网络）能够生成输出为类别1的概率，即:

$$y(x^n)=P(C_1|x^n) = P(t^n=1|x^n)$$

令$t^n = 1$代表类别1, $t^n = 0$ 代表类别2.

- 现在数据的似然函数是什么?

$$\mathrm{L} = \prod_{n=1}^{N} p(t^n \mid x^n) = \prod_{n=1}^{N} (y^n)^{t^n} (1-y^n)^{(1-t^n)}$$

对0/1二分类建模就像对抛硬币建模

- 同样，模型（神经网络）是对数据的概率分布建模.
- 线性回归中是假设数据服从正态分布; 这里，假设数据服从伯努利分布（*Bernoulli distribution*）(和抛硬币类似!).
- 所以:
- $\mathrm{L} = \prod_{n=1}^{N} p(t^n \mid x^n) = \prod_{n=1}^{N} (y^n)^{t^n} (1-y^n)^{(1-t^n)}$

对0/1二分类建模就像对抛硬币建模

则：

$$-\ln L = -\ln \prod_{n=1}^{N} (y^n)^{t^n} (1-y^n)^{(1-t^n)}$$

$$= -\sum_{n=1}^{N} \ln (y^n)^{t^n} (1-y^n)^{(1-t^n)}$$

$$= -\sum_{n=1}^{N} t^n \ln(y^n) + (1-t^n) \ln(1-y^n)$$

交叉熵出现!

极大似然与逻辑回归

小结一下:

在二分类问题（也就是逻辑回归）中，如果

1) 假设目标数据满足伯努利分布(Bernoulli distribution), 并且

2) 通过最小化数据的负对数似然来最大化数据的似然（概率），

那么我们发现实际上是需要最小化交叉熵损失（Cross Entropy Error）！

内容

- 什么是极大似然?
- 它是如何导致误差平方和（Sum Squared Error, SSE)是线性回归的损失函数?
- 它是如何导致交叉熵是逻辑回归的损失函数?
- ***它是如何导致交叉熵是多分类模型的损失函数?***
- 总结

多分类模型（Softmax）

- 当类别数多于两类，我们需要多个输出.
- 假设我们有 c 个输出, 每个对应于一类.
- 我们要求概率: $P(C_k|x^n) = y_k(x^n)$, 它代表输入数据属于第k 类的输出概率.
- 令 $t_k^n = 1$ 表示第n个样本来自于第k类, 否则就为0 (one-hot 编码)
- *如何写出似然函数?*

多分类模型(Softmax)

- 其中某个样本属于某类的概率为: $p(t^n \mid x^n) = \prod_{k=1}^{c} (y_k^n)^{t_k^n}$
- 例如,假设第n个样本属于总共4类中的第3类,则:

$$p(t^n \mid x^n) = \prod_{k=1}^{4} (y_k^n)^{t_k^n}$$

$$= (y_1^n)^0 (y_2^n)^0 (y_3^n)^1 (y_4^n)^0$$

$$= y_3^n$$

- 因此总的似然函数写为:

$$L = \prod_{n=1}^{N} p(t^n \mid x^n) = \prod_{n=1}^{N} \prod_{k=1}^{c} (y_k^n)^{t_k^n}$$

多分类模型(Softmax)

- 因此总的似然函数写为:

$$L=\prod_{n=1}^{N} p(t^n \mid x^n) = \prod_{n=1}^{N}\prod_{k=1}^{c}(y_k^n)^{t_k^n}$$

负对数似然写为:

$$-\ln L = -\ln\prod_{n=1}^{N}\prod_{k=1}^{c}(y_k^n)^{t_k^n} = -\sum_{n=1}^{N}\sum_{k=1}^{c} t_k^n \ln y_k^n$$

交叉熵再现!

极大似然与多分类(c类)

小结一下:

在多分类（c类）中(Softmax属于其中), 如果

1) 假设目标数据满足Multinoulli分布（很多写为多项式分布，不太准确，其实课本上有澄清），

2) 通过最小化数据的负对数似然来最大化数据的似然,

那么我们发现实际上是需要最小化交叉熵损失（Cross Entropy Error）！

总结

- ***极大似然**是一种*“元目标（meta-objective）函数”：
  - 总体而言，我们是想调整模型参数来最大化观测数据出现的似然（可能性）.
  - 损失函数的具体形式会随着待建模数据的分布类型变化而变化.
- 数据的高斯分布假设导致了要使用SSE
- 数据的伯努利分布假设导致了要使用cross entropy
- 数据的Multinoulli分布假设导致了要使用cross entropy.