Neural Network
Loss Function

损失函数

```
model.compile(loss='mean_squared_error', optimizer='sgd', metrics=['mae', 'acc'])
```

大纲

- **入门**
  - **定义, 特性, 训练过程**
- 常用的损失函数
  - 回归
  - 分类
- 正则化Regularization
- 实例

定义

在深度学习中, 损失函数是用来衡量一组参数的质量的函数, 衡量的方式是比较网络输出和真实输出的差异.

命名

损失函数loss function

=

价值函数cost function

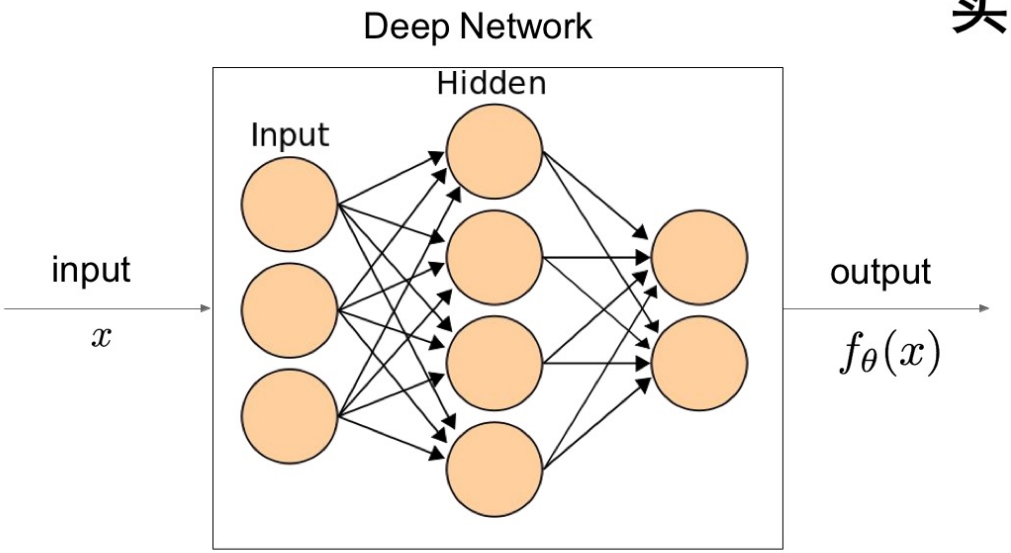
=

目标函数objective function

=**误差函数error** function

损失函数(1)

衡量网络输出和真实值的差异



labels (ground truth)

input

$$\mathcal{L}(\theta) = distance(f_\theta(x), y)$$

error

parameters (weights, biases)

损失函数 (2)

- 损失函数并不使用测试数据(test/validation data)来衡量网络的性能.
- 损失函数用来指导训练过程, 使得网络的参数向损失降低的方向改变.

训练过程

随机梯度下降法Stochastic gradient descent

- 试图找到一组参数使得损失函数的值越小越好.
- 调整参数的大小和方向取决于损失函数相对于参数的偏导数:

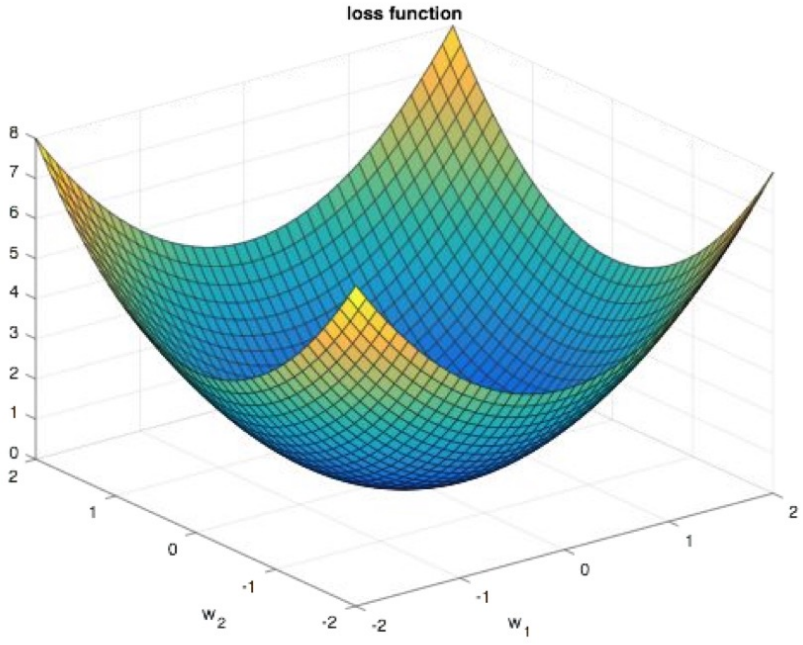
$$\frac{\partial \mathcal{L}}{\partial w} \qquad \frac{\partial \mathcal{L}}{\partial b}$$

特性 (1)

- 最小值(0): 当网络的输出和真实输出一致
- 当输出和真实输出越不一致时值越大

特性 (2)

- 理想情况 → convex 凸函数
- 实际上 → not convex非凸函数
  - 需要根据输出的变化而平滑的变化
    - 需要可导(SGD优化)
    - 需要容易求导



前提

- 为了使得误差向后传递**backpropagation**工作:
  - 损失函数为每一个独立训练样本的损失的均值

empirical risk

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_i$$

  - 损失函数为网络输出的函数

大纲

- 入门
  - 定义, 特性, 训练过程
- **常用的损失函数**
  - **回归**
  - 分类
- 正则化Regularization
- 实例

常用的损失函数(1)

- 不同的任务类型需要不同的损失函数:
  - 回归Regression: 网络输出一个连续的数值
    - 例如: 预测一栋房屋的价值
    - 损失函数:绝对值误差, 平方差

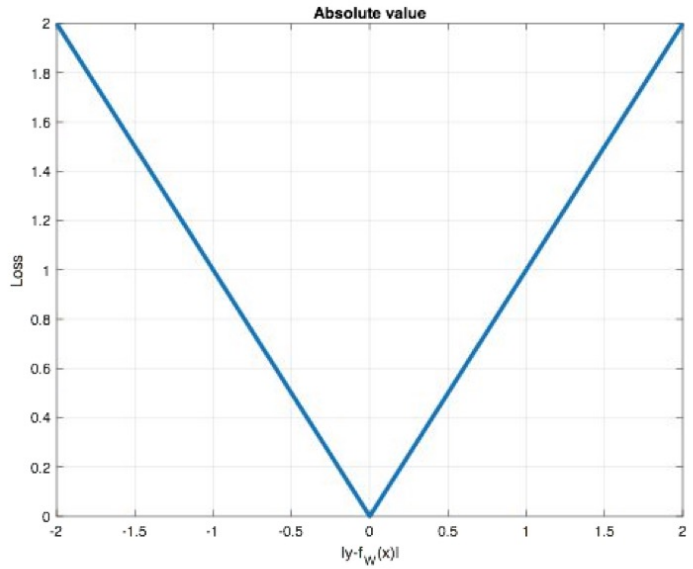
常用的损失函数(2)

- 不同的任务类型需要不同的损失函数:
  - 分类Classification: 网络的输出为一个类别, 从预定义的一组类别中的一个
    - 实例: 判断邮件是否是垃圾邮件.
    - 损失函数: hinge loss, Cross-entropy loss

绝对误差函数Absolute value, L1-norm

- 非常质感的损失函数
  - 得到的解会比较稀疏 sparser
    - 在高纬任务中表现比较好
    - 预测速度快
  - 对outliers不敏感

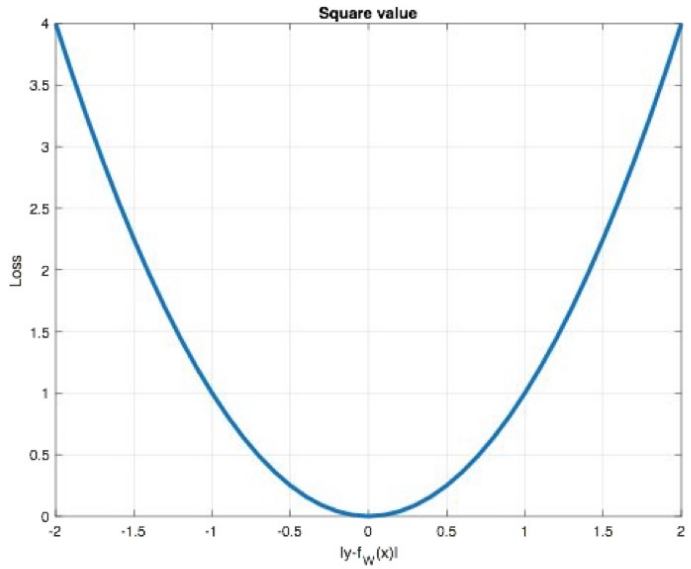
$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}|y_i - f_\theta(x_i)|$$



方差函数Square error, Euclidean loss, L2-norm

- 常用的损失函数
  - 比绝对误差函数得到的结果更精准
  - 对大的误差输出更敏感
  - 对outliers很敏感

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2$$



大纲

- 入门
  - 定义, 特性, 训练过程
- **常用的损失函数**
  - 回归
  - **分类**
- 正则化Regularization
- 实例

分类 (1)

将输入分为固定的几个类别

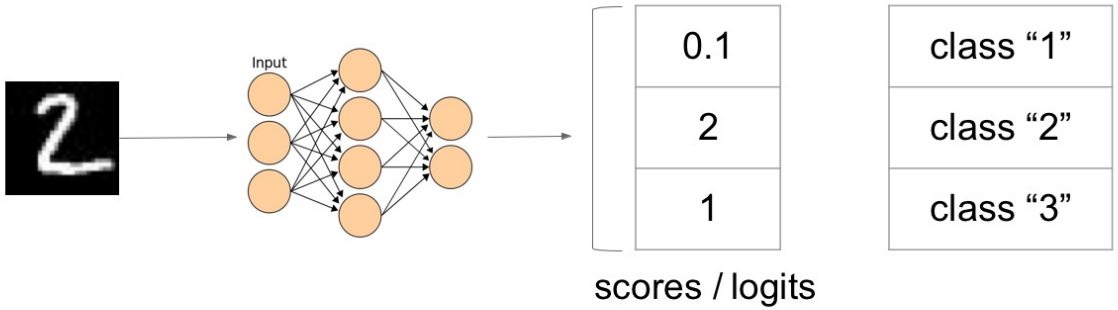
class “1”

class “2”

class “3”

分类 (2)

- 我们期望得到的最终结果是每一个输入对应一个输出类别
  - 网络的输出包含了对每一个类别的预测值
    - 如果有K个类别, 网络的输出为K个值

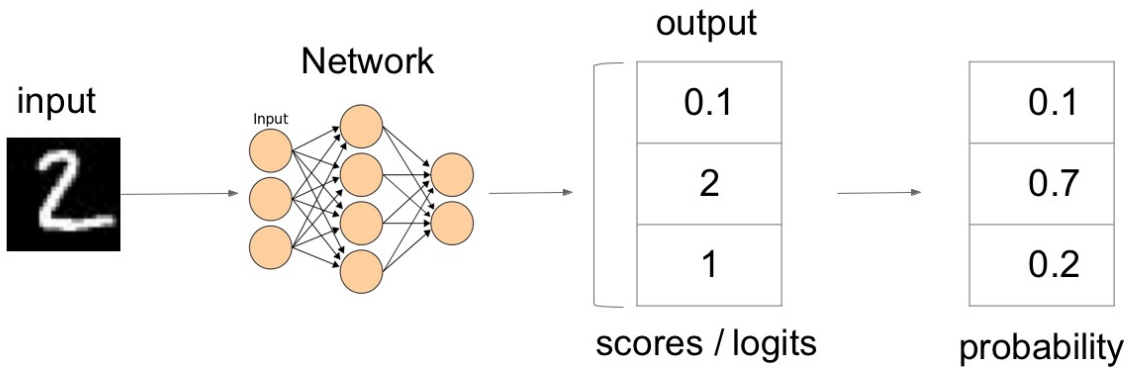


分类 (3)

- 如何设计损失函数?
  - 将真实的唯一输出编码为一个向量 → 独热编码 One-hot encoding
  - 非概率的解释 → **hinge loss**
  - 概率解释: 将输出转换为概率函数→ Softmax

Softmax (1)

- 将输出转换为概率
  - 概率的值范围为 0.0 到 1.0
  - 属于所有类别的概率之和为1.0

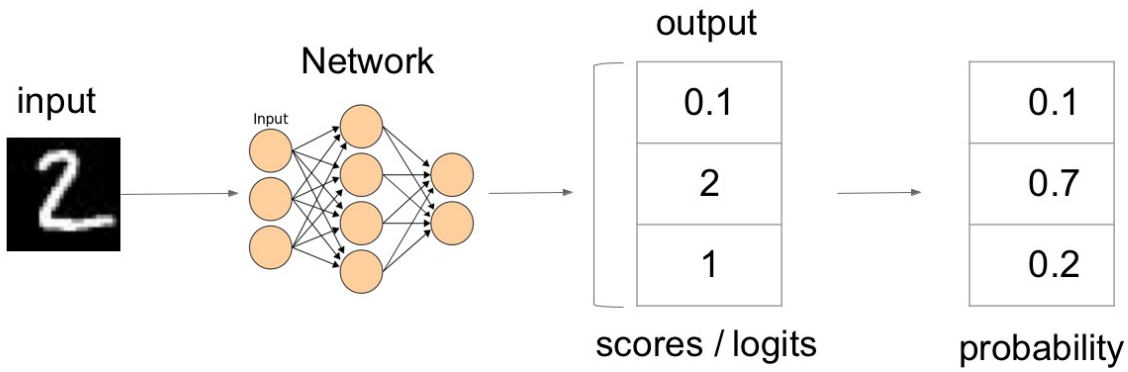


Softmax (2)

- Softmax function

scores (logits)

$$S(l_i) = \frac{e^{l_i}}{\sum_k e^{l_k}}$$

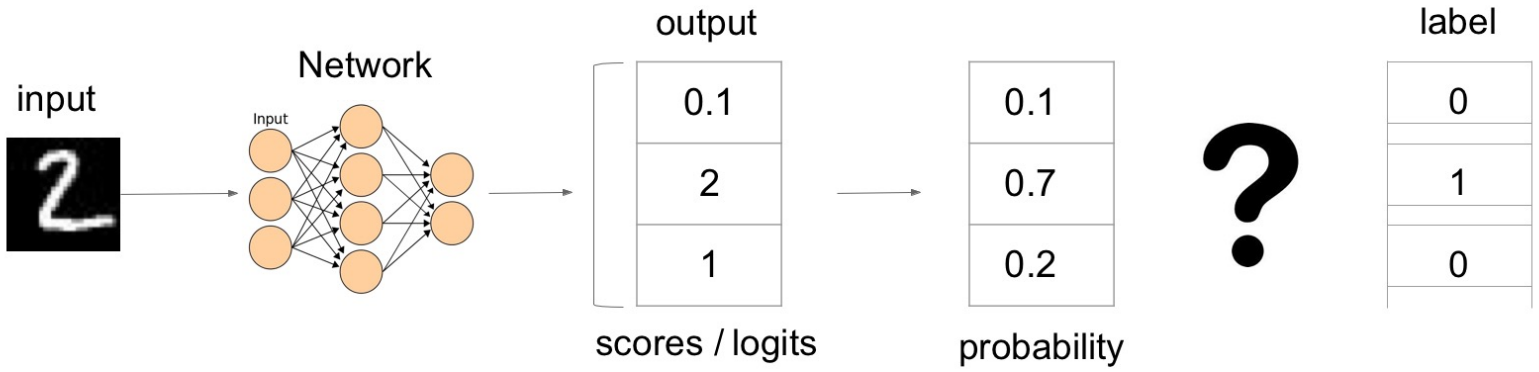


独热编码One-hot encoding

- 将一个类别标签转换为一个向量 (每一个元素的值为1 或者 0)
  - 向量的元素个数为总的类别的数量K
  - 1 对于正确的类别

class “1”	class “2”	class “3”
1	0	0
0	1	0
0	0	1

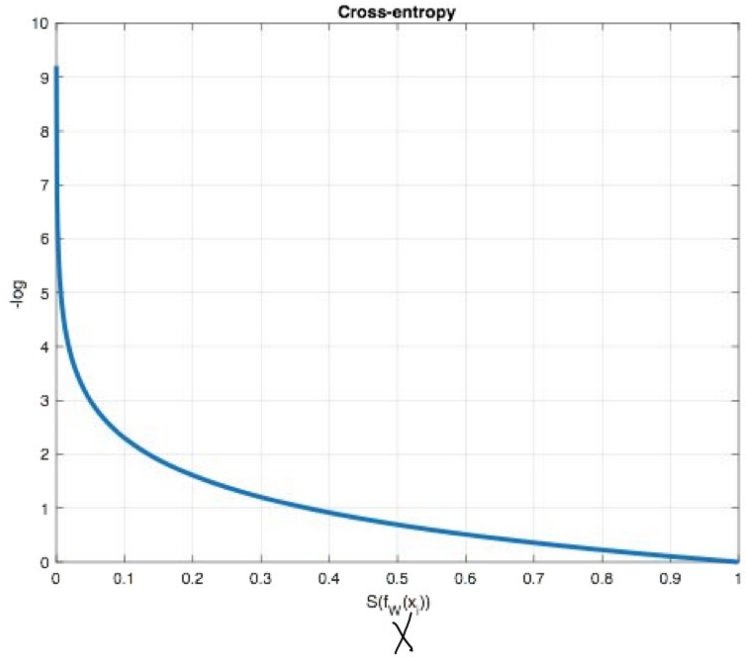
Cross-entropy loss (1)



$$\mathcal{L}_i = -\sum_k y_k \log(S(l_k))$$

Cross-entropy loss (2)

$$\mathcal{L}_i = -\sum_k y_k \log(S(l_k))$$



Cross-entropy loss (3)

- 对应n个输入

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_i$$

labels (one-hot)

$$\mathcal{L} = -\sum_{i=1}^{n}\mathbf{y}_i \log(S(f_\theta(\mathbf{x}_i)))$$

Softmax

Cross-entropy loss (4)

- 一般而言cross-entropy loss比方差函数 square error loss在分类的情形下要好:
  - 方差函数对误差的输出惩罚非常大
  - 如果使用Softmax激活函数, 加上方差函数作为损失函数, 梯度包含$\hat{y}\,(1-\hat{y})$, 当输出接近0.0 或者 1.0 的时候, 梯度值非常小, 网络的训练会比较慢.

Gradient of square error loss (4)

prediction $\hat{y}_i = \sigma(Wx_i + b)$

loss $J = \frac{1}{2}(y_i - \hat{y}_i)^2$

$$\frac{dJ}{dW} = (y_i - \hat{y}_i)\sigma'(Wx_i + b)x_i$$

$$\sigma'(Wx_i + b) = \sigma(Wx_i + b)(1 - \sigma(Wx_i + b))$$

$$\frac{dJ}{dW} = (y_i - \hat{y}_i)\hat{y}_i(1 - \hat{y}_i)x_i$$

Gradient of cross-entropy loss (4)

$$\widehat{y_i} = \sigma(x_i) = \sigma(Wx_i + b)$$

$$J = \sum_{i=1}^{n} \left[-y_i \log\big(\sigma(x_i)\big) - (1 - y_i)\log(1 - \sigma(x_i)) \right]$$

$$\frac{dJ}{dW} = \sum_{i=1}^{n} \left[-y_i \log\big(\sigma(x_i)\big) - (1 - y_i)\log(1 - \sigma(x_i)) \right]$$

$$\frac{dJ}{dW} = \sum_{i=1}^{n} \left[-y_i \frac{\sigma(x_i)\big(1 - \sigma(x_i)\big)x_i}{\sigma(x_i)} + (1 - y_i) \frac{\sigma(x_i)\big(1 - \sigma(x_i)\big)x_i}{1 - \sigma(x_i)} \right]$$

$$\frac{dJ}{dW} = \sum_{i=1}^{n} \left[-y_i\big(1 - \sigma(x_i)\big)x_i + (1 - y_i)\sigma(x_i)x_i \right]$$

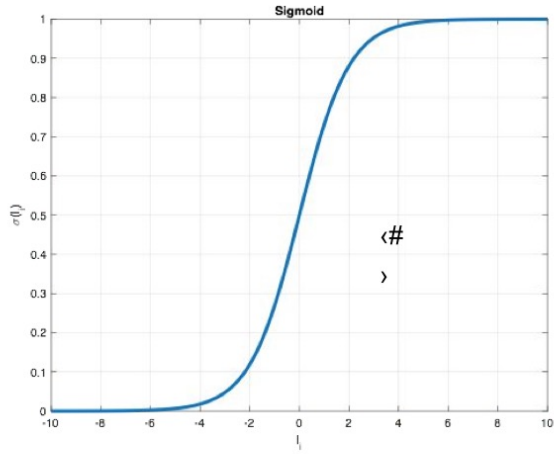
$$\frac{dJ}{dW} = \sum_{i=1}^{n} \left[-y_i x_i + y_i \sigma(x_i)x_i + \sigma(x_i)x_i - y_i \sigma(x_i)x_i \right]$$

$$\frac{dJ}{dW} = \sum_{i=1}^{n} \left[\left(\sigma(x_i) - y_i\right)x_i \right] = \sum_{i=1}^{n} (\widehat{y_i} - y_i)x_i$$

Multi-label 分类 (1) , not Multi-class 分类

- 输出属于多个类别中的一个或者多个类
  - 例如一副包含猫咪的图像可以同时属于“猫”， “哺乳动物”， “宠物”
- 对每一个输出独立使用Sigmoid激活函数, 不使用 softmax

$$\sigma(l_i) = \frac{1}{1 + e^{-l_i}}$$



Multi-label 分类 (2) , not Multi-class 分类

- Cross-entropy loss for multi-label classification:

$$\mathcal{L}_i = -\sum_k y_k \log(\sigma(l_i)) + (1 - y_k) \log(1 - \sigma(l_i))$$

- Cross-entropy loss for multi-class classification:

$$\mathcal{L}_i = -\sum_k y_k \log(S(l_k))$$

大纲

- 入门
  - 定义, 特性, 训练过程
- 常用的损失函数
  - 回归
  - 分类
- **正则化Regularization**
- 实例

正则化Regularization

- 避免网络过拟合 **prevent overfitting**
  - L2-regularization (weight decay):

regularization parameter

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2} W^2$$

  - L1-regularization:

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2} |W|$$

大纲

- 入门
  - 定义, 特性, 训练过程
- 常用的损失函数
  - 回归
  - 分类
- 正则化Regularization
- **实例**

实例

$$\mathcal{L} = -\sum_{i=1}^{n} \mathbf{y}_i \log(S(f_\theta(\mathbf{x}_i)))$$

网络输出 (Softmax)	真实输出	是否正确
0.3, 0.3, 0.4	0 0 1	是
0.3, 0.4, 0.3	0 1 0	是
0.1, 0.2, 0.7	1 0 0	否

Classification accuracy = 2/3
cross-entropy loss = 1.38
square loss = 0.81

网络输出 (Softmax)	真实输出	是否正确
0.1, 0.2, 0.7	0 0 1	是
0.1, 0.7, 0.2	0 1 0	是
0.3, 0.4, 0.3	1 0 0	否

Classification accuracy = 2/3
cross-entropy loss = 0.64
square loss = 0.34

网络输出 (Softmax)	真实输出	是否正确
0.3, 0.3, 0.4	0 0 1	是
0.3, 0.4, 0.3	0 1 0	是
0.1, 0.2, 0.7	1 0 0	否

正确率: 2/3=0.67

第一个sample的cross-entropy-loss:

-((ln(0.3)*0) + (ln(0.3)*0) + (ln(0.4)*1)) = -ln(0.4)

第一个sample的square-error-loss:

(0.3 - 0)^2 + (0.3 - 0)^2 + (0.4 - 1)^2 = 0.09 + 0.09 + 0.36 = 0.54

平均cross-entropy-loss: -(ln(0.4) + ln(0.4) + ln(0.1)) / 3 = 1.38

平均square-error-loss: (0.54 + 0.54 + 1.34) / 3 = 0.81

网络输出 (Softmax)	真实输出	是否正确
0.1, 0.2, 0.7	0 0 1	是
0.1, 0.7, 0.2	0 1 0	是
0.3, 0.4, 0.3	1 0 0	否

正确率: 2/3=0.67

平均cross-entropy-loss: -(ln(0.7) + ln(0.7) + ln(0.3)) / 3 = 0.64

平均square-error-loss: (0.14 + 0.14 + 0.74) / 3 = 0.34

答疑时间

如何证明Logistic Regression的损失函数具有全局最优解

- 证明思路:
  - 证明Logistic Regression的loss function是Convex的

证明函数Convex的方法一

$f : X \to \mathbf{R}$

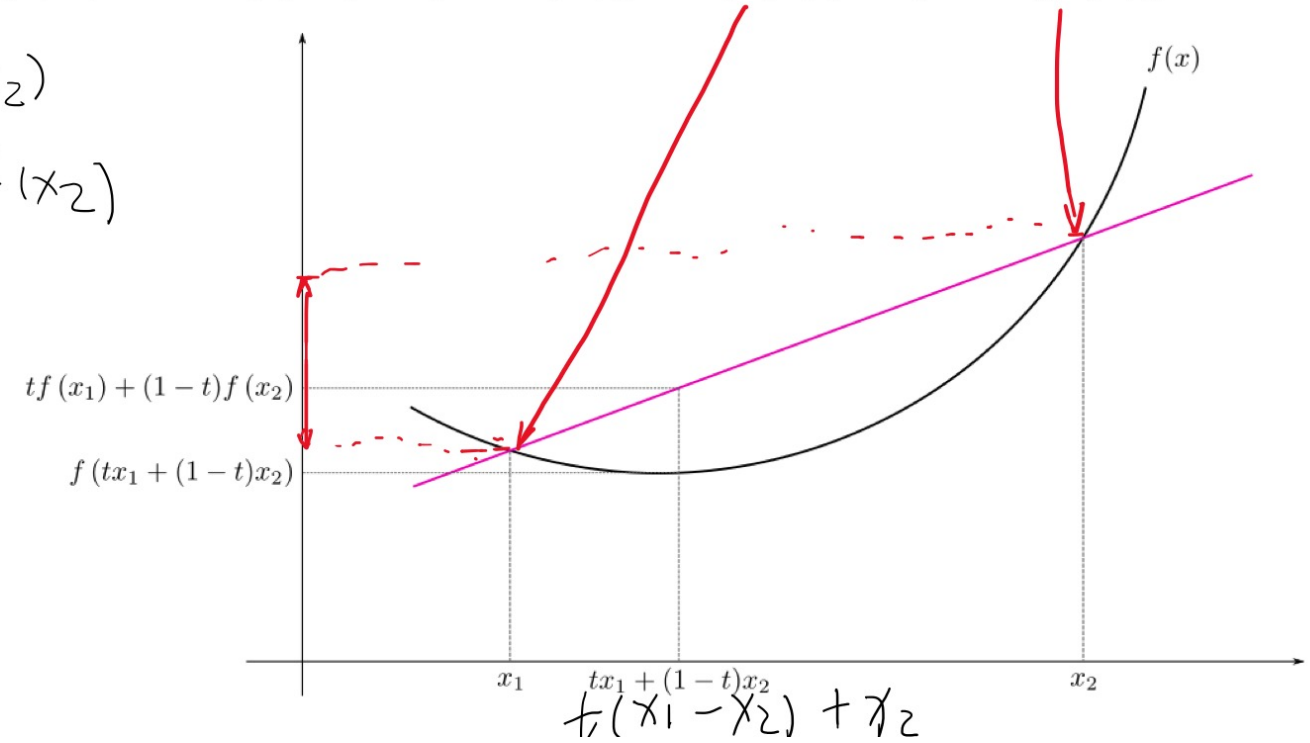
$\forall x_1, x_2 \in X, \forall t \in [0,1]:$ $f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$

$tf(x_1) + (1-t)f(x_2)$

$t[f(x_1) - f(x_2)] + f(x_2)$

斜率 $f(x_1) - f(x_2)$

截距 $f(x_2)$



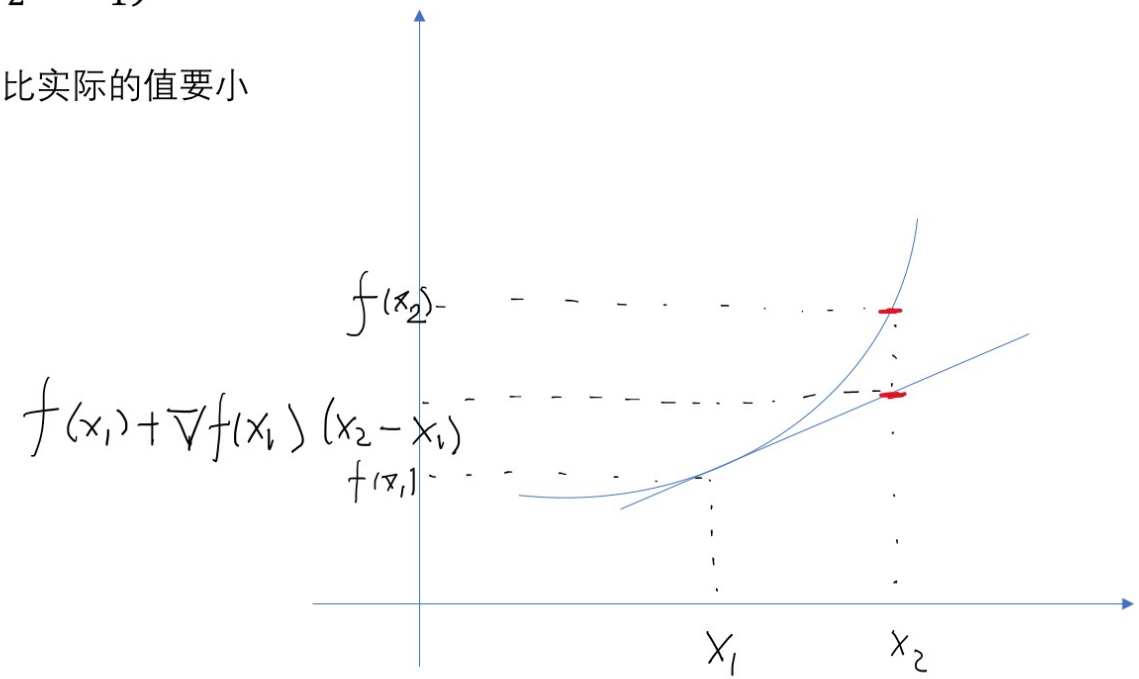
$t(x_1 - x_2) + x_2$

证明函数Convex的方法二

$f: X \to \mathbf{R}$ 如果 f一阶 可倒

$$f(x_2) \geq f(x_1) + \nabla f(x_1)(x_2 - x_1)$$

直觉上讲, 就是f(x)的一阶泰勒系数比实际的值要小



证明函数Convex的方法三

$f : X \to \mathbf{R}$　　如果 f 二阶可倒　　如果f(x)的hessian矩阵(二阶偏导矩阵)是正定矩阵, 即:

$\forall z$:　　$z^T \nabla^2 f(x) z \geq 0$

$f(x_1, x_2, \ldots, x_n)$, 如果 f 所有的二阶偏导数都存在，那么 f 的黑塞矩阵的第 ij–项即：

$$H(f)_{ij}(x) = D_i D_j f(x)$$

其中$x = (x_1, x_2, \ldots, x_n)$，即

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\, \partial x_n} \\ \frac{\partial^2 f}{\partial x_2\, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\, \partial x_1} & \frac{\partial^2 f}{\partial x_n\, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Hessian Matrix举例

$$f(x,y) = x^2 + 3y^2$$

$$\frac{\partial f}{\partial x} = 2x \qquad \frac{\partial^2 f}{\partial x^2} = 2 \qquad \frac{\partial^2 f}{\partial x \partial y} = 0$$

$$\frac{\partial f}{\partial y} = 6y \qquad \frac{\partial^2 f}{\partial y \partial x} = 0 \qquad \frac{\partial^2 f}{\partial y^2} = 6$$

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix}$$

证明函数Convex的方法四

$let\ f(x), g(x)\ are\ 2\ convex\ functions, then\ any\ linear\ positive\ combination\ of\ these\ 2\ functions\ are\ convex$

$\lambda_1 f(x) + \lambda_2 g(x)\ \ convex\ when\ \lambda_1 \geq 0\ and\ \lambda_2 \geq 0$

证明Logistic Regression的loss function是Convex function

$$J(\theta) = \sum_{i=1}^{n} y^i [-\log(h_\theta(x^i))] + (1-y^i)[-\log(1-h_\theta(x^i))]$$

$$-\log(h_\theta(x)) = -\log \frac{1}{1+e^{-\theta^T x}}$$

$$= \log(1+e^{-\theta^T x})$$

$$\nabla_\theta \left[-\log\left(h_\theta(x)\right) \right] = \nabla_\theta \left[\log\left(1 + e^{-\theta^T x}\right) \right]$$

(EQ-1)

$$= \frac{1}{1 + e^{-\theta^T x}} \cdot e^{-\theta^T x} \cdot -x$$

$$= \left(\frac{-e^{\theta^T x}}{1 + e^{-\theta^T x}} \right) x$$

$$= \frac{1 - (1 + e^{-\theta^T x})}{1 + e^{-\theta^T x}} x$$

$$\nabla_\theta \left[-\log(h_\theta(x)) \right] = (h_\theta(x) - 1)\, x$$

hessian:

$$\nabla_\theta^2 \left[-\log(h_\theta(x)) \right] = \nabla_\theta \left(\nabla_\theta \left[-\log(h_\theta(x)) \right] \right)^T$$

$$= \nabla_\theta \left((h_\theta(x) - 1)\, x \right)$$

$$= x \nabla_\theta (h_\theta(x)) = x \cdot h_\theta(x) (1 - h_\theta(x))\, x^T$$

hessian matrix is positive semi-definite

$z^T \nabla_\theta^2 (-\log(h_\theta(x)))\, z$ FQ-2

$= z^T \left[h_\theta(x)(1 - h_\theta(x)) x^T \right] z$

$= h_\theta(x)(1 - h_\theta(x))\, z^T x x^T z$

$= h_\theta(x)(1 - h_\theta(x))(z^T x)^2 \geq 0$

证明Logistic Regression的loss function是Convex function

$$J(\theta) = \sum_{i=1}^{n} y^i [-\log(h_\theta(x^i))] + (1-y^i)[-\log(1-h_\theta(x^i))]$$

$$-\log(1-h_\theta(x)) = -\log\left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

$$= -\log\left(\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}\right) = -\log e^{-\theta^T x} + \log(1+e^{-\theta^T x})$$

$$= \theta^T x + \log(1+e^{-\theta^T x})$$

$$\nabla_\theta \left[-\log(1 - h_\theta(x)) \right] = \nabla_\theta \left[\theta^T x + \log(1 + e^{-\theta^T x}) \right]$$

$$= x + \nabla_\theta \left[\log(1 + e^{-\theta^T x}) \right]$$

(EQ - 1)

hessian:

$$\nabla_\theta^2 \left[-\log\left(1 - h_\theta(x)\right) \right] = \nabla_\theta \left(\nabla_\theta \left[-\log\left(1 - h_\theta(x)\right) \right] \right)$$

$$= \nabla_\theta \left(x + \nabla_\theta \left[\log\left(1 + e^{-\theta^T x}\right) \right] \right)$$

$$= \nabla_\theta^2 - \left[\log\left(\frac{1}{1 + e^{-\theta^T x}} \right) \right]$$

$$= \nabla_\theta^2 \left[-\log\left(h_\theta(x)\right) \right] \quad (\text{EQ} - 2)$$

proved:

$-\log(h_\theta(x^i))$ and

$-\log(1-h_\theta(x^i))$ both

convex

$$J(\theta) = \sum_{i=1}^{n} y^i \left[-\log\left(h_\theta(x^i)\right) \right] +$$

$$\left(1 - y^i\right) \left[-\log\left(1 - h_\theta(x^i)\right) \right])$$

convex (4-th rule)

Why Loss of Neural Network Non Convex

https://papers.nips.cc/paper/1028-exponentially-many-local-minima-for-single-neurons.pdf

We show that for a single neuron with the logistic function as the transfer function the number of local minima of the error function based on the square loss can grow exponentially in the dimension.

Why local minima close to global minima in large Neural Networks

- **The Loss Surface of Deep and Wide Neural Networks:** http://proceedings.mlr.press/v70/nguyen17a/nguyen17a-supp.pdf
- We show that this is (almost) true, in fact almost all local minima are globally optimal, for a fully connected network with squared loss and analytic activation function given that the number of hidden units of one layer of the network is larger than the number of training points and the network structure from this layer on is pyramidal.

- **The Loss Surfaces of Multilayer Networks**
- https://arxiv.org/abs/1412.0233
- We show that for large-size decoupled networks the lowest critical values of the random loss function form a layered structure and they are located in a well-defined band lower-bounded by the global minimum. The number of local minima outside that band diminishes exponentially with the size of the network.