

Contents

1	1. logistic regression as a neural network	1
1.1	1.1. binary classification	1
1.2	1.2. logistic regression	2
1.3	1.3. logistic regression cost function	2
1.4	1.4. gradient descent	2
1.5	1.5. derivatives	2
1.6	1.6. more derivative examples	2
1.7	1.7. computation graph	3
1.8	1.8. derivatives with a computation graph	3
1.9	1.9. logistic regression gradient descent	3
1.10	1.10. gradient descent on m examples	3
2	2. python & vectorization	6
2.1	2.1. vectorization	6
2.2	2.2. more examples of vectorization	6
2.3	2.3. vectorizing logistic regression	8
2.4	2.4. vectorizing logistic regression's gradient output	8
2.5	2.5. broadcasting in python	8
2.6	2.6. a note on python/numpy vectors	8
2.7	2.7. quick tour of jupyter/ipython notebooks	10
2.8	2.8. explanation of logistic regression cost function	10
2.9	2.9. programming assignments	10

contents

- 1. logistic regression as a neural network
- 1.1. binary classification
- 1.2. logistic regression
- 1.3. logistic regression cost function
- 1.4. gradient descent
- 1.5. derivatives
- 1.6. more derivative examples
- 1.7. computation graph
- 1.8. derivatives with a computation graph
- 1.9. logistic regression gradient descent
- 1.10. gradient descent on m examples
- 2. python & vectorization
- 2.1. vectorization
- 2.2. more examples of vectorization
- 2.3. vectorizing logistic regression
- 2.4. vectorizing logistic regression's gradient output
- 2.5. broadcasting in python
- 2.6. a note on python/numpy vectors
- 2.7. quick tour of jupyter/ipython notebooks
- 2.8. explanation of logistic regression cost function
- 2.9. programming assignments

1 1. logistic regression as a neural network

1.1 1.1. binary classification

维度为 (64, 64, 3) 的图片 ==> img vector: $x =$ 维度为 $(64 \times 64 \times 3 = 12288, 1)$ 的列向量。($n_x = 12288$)

$$(x, y), x \in R^{n_x}, y \in \{0, 1\}$$

$m = m_{train}$ 个训练样本: $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$, m_{test} 个测试样本。

X 表示一个 $n_x * m$ 的训练样本矩阵, 在 python 里就是 $X.shape=(n_x, m)$ Y 表示一个 $1 * m$ 的向量, 在 python 里是 $Y.shape=(1, m)$

1.2 1.2. logistic regression

given x , want $\hat{y} = P(y = 1|x), x \in R^{n_x}$

params: $w \in R^{n_x}, b \in R$

output: $\hat{y} = \sigma(w^T x + b), \sigma(z) = \frac{1}{1+e^{-z}}$

1.3 1.3. logistic regression cost function

$(x^{(i)}, y^{(i)})$ 表示第 i 个样本。

Loss(error) function 只针对一条训练样本:

- square error 的 loss function:

$$L(\hat{y}, y) = 1/2 * (\hat{y} - y)^2$$

- logistic regression 的 loss function:

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

if $y = 1, L(\hat{y}, y) = -\log \hat{y}$, want $\log \hat{y}$ large, want \hat{y} large

if $y = 0, L(\hat{y}, y) = -\log(1 - \hat{y})$, want \hat{y} small

Cost function 针对全体训练样本:

$$J(w, b) = 1/m \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = -1/m \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

1.4 1.4. gradient descent

lr 的 $J(w, b)$ 是一个凸函数, 所以有全局最优。因为有全局最优, 所以 lr 的初始化一般是 0, 不用随机。梯度下降: 不断重复 $w = w - \alpha \frac{dJ(w)}{dw}$ 直到收敛。后续, 用 dw 来指代 $\frac{dJ(w)}{dw}$ 。梯度下降的公式:

$$w = w - \alpha dw$$

$$b = b - \alpha db$$

1.5 1.5. derivatives

derivative = slope, 就是 $dy/dx = \Delta y / \Delta x$

1.6 1.6. more derivative examples

$$f(a) = \log_e(a) = \ln(a), df(a)/da = \frac{1}{a}$$

1.7 1.7. computation graph

正向计算图算出输出，算每个参数的梯度就反向算。

Computation Graph

$$J(a, b, c) = 3(a + bc) = 3(5 + 3 \cdot 2) = 33$$

\underbrace{a}_{u}
 \underbrace{bc}_{v}
 $\underbrace{J}_{\text{J}}$

$$u = bc$$

$$v = a + u$$

$$J = 3v$$

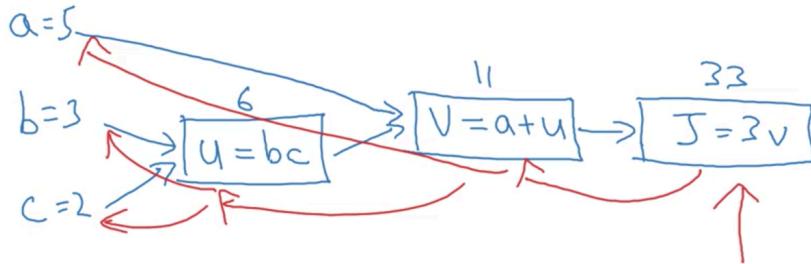


Figure 1: computation graph

1.8 1.8. derivatives with a computation graph

$$J = 3v, v = a + u, u = bc$$

$$\frac{dJ}{dv} = 3, \frac{dv}{da} = 1$$

$$\text{so, } \frac{dJ}{da} = \frac{dJ}{dv} \frac{dv}{da} = 3 * 1 = 3$$

$$\text{if } b = 2, \text{then } \frac{dJ}{dc} = \frac{dJ}{dv} \frac{dv}{du} \frac{du}{dc} = 3 * 1 * b = 3 * 1 * 2 = 6$$

写代码时，将 $\frac{dFinalOutputVar}{dvar}$ 记为 $dvar$ (最后输出对这个变量的偏导)

1.9 1.9. logistic regression gradient descent

$$\begin{aligned}
 z &= w^T x + b \\
 \hat{y} &= a = \sigma(z) \\
 L(a, y) &= -(y \log(a) + (1 - y) \log(1 - a))
 \end{aligned}$$

1.10 1.10. gradient descent on m examples

首先，根据 J 的公式，可以知道 $dJ/dw1$ 其实就是对每个样本的 $dw1$ 求和，然后/m。

每一次迭代，遍历 m 个样本，算出 $J/dw1/dw2/db$ ，然后用这些梯度去更新一次 $w1/w2/b$ 。

但这的 for loop 太多了。。所以我们需要 vectorization!

Computing derivatives

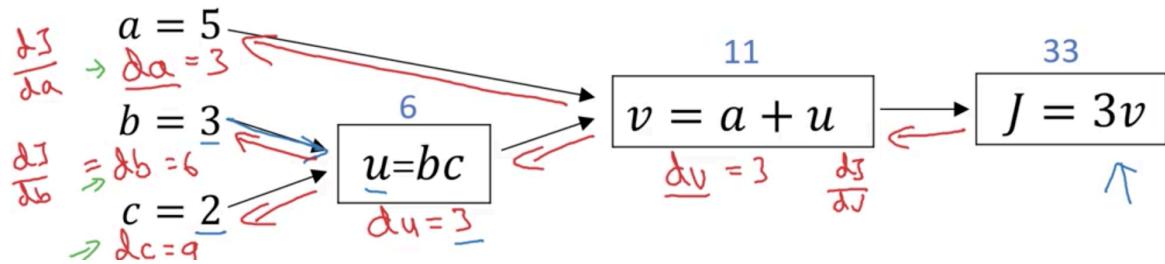


Figure 2: derivatives in computation graph

Logistic regression derivatives

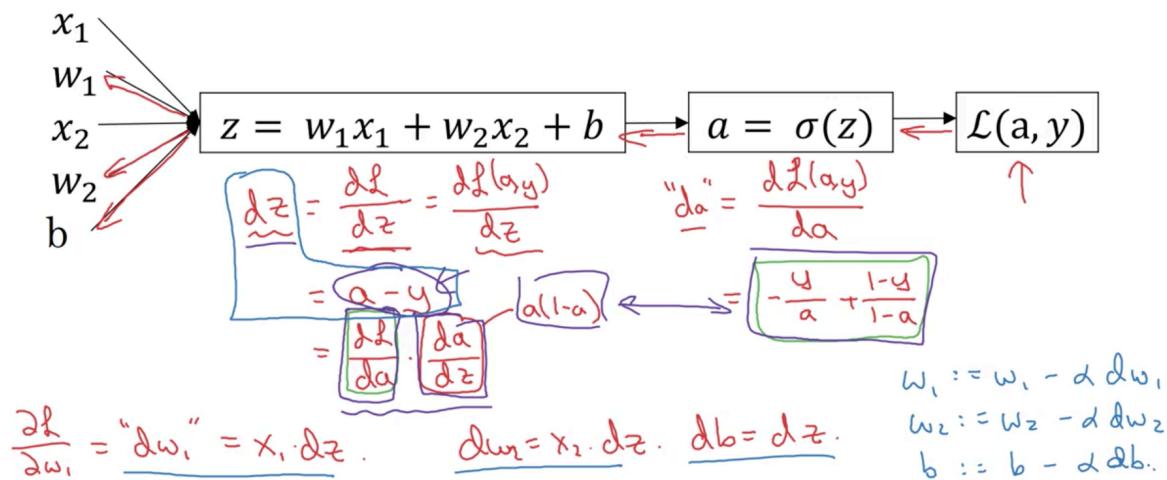


Figure 3: derivatives in computation graph in lr

Logistic regression on m examples

$$\underline{J(w, b)} = \frac{1}{m} \sum_{i=1}^m \underline{\ell(a^{(i)}, y^{(i)})}$$

$$\rightarrow \underline{a^{(i)}} = \underline{\hat{y}^{(i)}} = \sigma(\underline{z}^{(i)}) = \sigma(\underline{w}^\top \underline{x}^{(i)} + b)$$

$$\underline{d w_1^{(i)}}, \underline{d w_2^{(i)}}, \underline{d b^{(i)}}$$

$$\underline{\frac{\partial}{\partial w_1} J(w, b)} = \frac{1}{m} \sum_{i=1}^m \underbrace{\frac{\partial}{\partial w_1} \ell(a^{(i)}, y^{(i)})}_{d w_1^{(i)}} - (x^{(i)}, y^{(i)})$$

Figure 4: gradient_descent_lr_m_examples_djdw

Logistic regression on m examples

$$J = 0; \underline{d w_1} = 0; \underline{d w_2} = 0; \underline{d b} = 0$$

$$\rightarrow \text{For } i = 1 \text{ to } m$$

$$\underline{z}^{(i)} = \underline{w}^\top \underline{x}^{(i)} + b$$

$$\underline{a}^{(i)} = \sigma(\underline{z}^{(i)})$$

$$J += -[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log (1-a^{(i)})]$$

$$\underline{d z}^{(i)} = \underline{a}^{(i)} - \underline{y}^{(i)}$$

$$\left[\begin{array}{l} \underline{d w_1} += \underline{x}_1^{(i)} \underline{d z}^{(i)} \\ \underline{d w_2} += \underline{x}_2^{(i)} \underline{d z}^{(i)} \\ \underline{d b} += \underline{d z}^{(i)} \end{array} \right] \quad n=2$$

$$J /= m \leftarrow$$

$$\underline{d w_1} /= m; \underline{d w_2} /= m; \underline{d b} /= m. \leftarrow$$

$$d w_1 = \frac{\partial J}{\partial w_1}$$

$$w_1 := w_1 - \alpha \underline{d w_1}$$

$$w_2 := w_2 - \alpha \underline{d w_2}$$

$$b := b - \alpha \underline{d b}$$

Vectorization

Figure 5: gradient_descent_lr_m_examples

2 2. python & vectorization

2.1 2.1. vectorization

What is vectorization?

$$z = \underline{w^T x + b}$$

$$\omega = \begin{bmatrix} : \\ : \\ : \end{bmatrix} \quad x = \begin{bmatrix} : \\ : \\ : \end{bmatrix} \quad w \in \mathbb{R}^{n_x} \quad x \in \mathbb{R}^{n_x}$$

Non-vectorized:

$$z = 0$$

$$\text{for } i \text{ in range}(n-x):$$

$$z += \underline{\omega[i] * x[i]}$$

$$z += b$$

Vectorized

$$z = \underbrace{\text{np.dot}(\omega, x)}_{w^T x} + b$$

\Rightarrow GPU CPU } SIMD - single instruction multiple data.

Figure 6: gradient_descent_lr_m_examples

对于两个 100w 维的向量进行点乘，vectorization(1.5ms) 比 for loop(470ms+) 快

2.2 2.2. more examples of vectorization

Logistic regression derivatives

$$J = 0, \quad \boxed{dw_1 = 0, dw_2 = 0}, \quad db = 0 \quad dw = \text{np.zeros}((n_x, 1))$$

\rightarrow for $i = 1$ to n :

$$z^{(i)} = w^T x^{(i)} + b$$

$$a^{(i)} = \sigma(z^{(i)})$$

$$J += -[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

$$\frac{\partial J}{\partial w_j} = \frac{\partial J}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial w_j}$$

$$\frac{\partial z^{(i)}}{\partial w_j} = x_j^{(i)}$$

$$\frac{\partial J}{\partial w_j} = \sum_{i=1}^m x_j^{(i)} a^{(i)} (1 - a^{(i)})$$

$$dw_1 += x_1^{(i)} dz^{(i)}$$

$$dw_2 += x_2^{(i)} dz^{(i)}$$

$$db += dz^{(i)}$$

$$J = J/m, \quad \boxed{dw_1 = dw_1/m, dw_2 = dw_2/m}, \quad db = db/m$$

$$dw /= m$$

如上图，将 n_x 维的 dw 变为一个 np.array 即可干掉内层的 for loop。

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$$u = \mathbf{A}v$$

矩阵 \mathbf{A} 向量 v

$$u_i = \sum_j A_{ij} v_j$$

$$u = np.zeros((n, 1))$$

```

for i ...      ←
    for j ...   ←
        u[i] += A[i][j] * v[j]
    
```

$u = np.dot(A, v)$

Figure 7: gradient_descent_lr_m_examples

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \rightarrow u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_n} \end{bmatrix}$$

$$\rightarrow u = np.zeros((n, 1))$$

$$\boxed{\text{for } i \text{ in range}(n):} \leftarrow$$

$$\rightarrow u[i] = \mathbf{math.exp}(v[i])$$

$\text{import numpy as np}$
 $u = np.exp(v) \leftarrow$
 \nearrow
 $\text{np.log}(v)$
 $\text{np.abs}(u)$
 $\text{np.maximum}(v, 0)$
 $v**2$
 v/v

Figure 8: gradient_descent_lr_m_examples

2.3 2.3. vectorizing logistic regression

可见，整个求 Z 的过程可以变成一句话，而求 A 时，需要封装一个基于 numpy 的 sigmoid 函数。

Vectorizing Logistic Regression

$$\begin{aligned} \rightarrow z^{(1)} &= w^T x^{(1)} + b \\ \rightarrow a^{(1)} &= \sigma(z^{(1)}) \end{aligned}$$

$$z^{(2)} = w^T x^{(2)} + b$$

$$a^{(2)} = \sigma(z^{(2)})$$

$$z^{(3)} = w^T x^{(3)} + b$$

$$a^{(3)} = \sigma(z^{(3)})$$

$$X = \begin{bmatrix} | & | & | & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & | & | \end{bmatrix} \quad \frac{(n_{x,m})}{R^{n_x \times m}}$$

$$w^T \begin{bmatrix} | & | & | & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & | & | \end{bmatrix}$$

$$\underline{\underline{Z}} = \begin{bmatrix} z^{(1)} & z^{(2)} & \dots & z^{(m)} \end{bmatrix} = w^T X + [b \ b \ \dots \ b] \quad \begin{bmatrix} w^T x^{(1)} + b \\ \vdots \\ w^T x^{(m)} + b \end{bmatrix}$$

$$\rightarrow Z = np.dot(w.T, X) + b \quad (1,1) \quad R$$

$$A = [a^{(1)} \ a^{(2)} \ \dots \ a^{(m)}] = \sigma(Z)$$

"Broadcasting"

Figure 9: gradient_descent_lr_m_examples

2.4 2.4. vectorizing logistic regression's gradient output

2.5 2.5. broadcasting in python

```
A = ndarray([[1,2,3,4],[2,3,4,5],[3,4,5,6]]) # 3*4
calc = A.sum(axis=0) # A 的每列求和, 得到 1*4
calc2 = A.sum(axis=1) # A 的每行求和, 得到 3*1
A/calc.reshape(1,4) # 得到一个 3*4 的矩阵, 就是 broadcasting。其实等价于 A/calc, 但为了保险, 可以调用 reshape(1,4) 来确保
```

小结:

2.6 2.6. a note on python/numpy vectors

numpy 的 broadcasting 文档

其实就两个准则:

- they are equal, or
- one of them is 1

```
a=np.random.randn(5) # a.shape=(5,) 是一个 vector(rank 1 array), 不是一个矩阵, 所以 a.T 还是 (5,), np.dot(a,a.T)=
```

```
b=np.random.randn(5,1) # a.shape=(5,1), a.T.shape=(1,5), np.dot(a,a.T) 是一个 5*5 的, np.dot(a.T,a) 是一个
```

可以加一句:

Vectorizing Logistic Regression

$$\begin{aligned}
 dz^{(1)} &= a^{(1)} - y^{(1)} & dz^{(2)} &= a^{(2)} - y^{(2)} & \dots \\
 dz &= [dz^{(1)} \ dz^{(2)} \ \dots \ dz^{(m)}] \leftarrow \\
 A &= [a^{(1)} \ \dots \ a^{(m)}], \quad Y = [y^{(1)} \ \dots \ y^{(m)}] \\
 \Rightarrow dz &= A - Y = [a^{(1)} - y^{(1)} \ a^{(2)} - y^{(2)} \ \dots] \\
 \rightarrow dw &= 0 \\
 dw &+ = \underbrace{x^{(1)} dz^{(1)}}_{\text{for } i} \\
 dw &+ = \underbrace{x^{(2)} dz^{(2)}}_{\text{for } i} \\
 &\vdots \\
 dw &+ = \underbrace{x^{(m)} dz^{(m)}}_{\text{for } i} \\
 dw &/= m \\
 db &= 0 \\
 db &+ = dz^{(1)} \\
 db &+ = dz^{(2)} \\
 &\vdots \\
 db &+ = dz^{(m)} \\
 db &/= m
 \end{aligned}$$

$$\begin{aligned}
 db &= \frac{1}{m} \sum_{i=1}^m dz^{(i)} \\
 &= \frac{1}{m} \text{np.sum}(dz) \\
 dw &= \frac{1}{m} X dz^\top \\
 &= \frac{1}{m} \left[\underbrace{x^{(1)} \ \dots \ x^{(m)}}_1 \right] \left[\begin{array}{c} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{array} \right] \\
 &= \frac{1}{m} \left[\underbrace{x^{(1)} dz^{(1)}}_{n \times 1} + \dots + \underbrace{x^{(m)} dz^{(m)}}_{n \times 1} \right]
 \end{aligned}$$

Figure 10: gradient_descent_lr_m_examples

Implementing Logistic Regression

$$\begin{aligned}
 J &= 0, \ dw_1 = 0, \ dw_2 = 0, \ db = 0 \\
 \text{for } i &= 1 \text{ to } m: \\
 z^{(i)} &= w^T x^{(i)} + b \leftarrow \\
 a^{(i)} &= \sigma(z^{(i)}) \leftarrow \\
 J &+= -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})] \\
 dz^{(i)} &= a^{(i)} - y^{(i)} \leftarrow \\
 \left[\begin{array}{l} dw_1 += x_1^{(i)} dz^{(i)} \\ dw_2 += x_2^{(i)} dz^{(i)} \\ db += dz^{(i)} \end{array} \right] & \left[\begin{array}{l} dw_t = X \times dz^{(i)} \\ \text{for } i \end{array} \right] \\
 J &= J/m, \ dw_1 = dw_1/m, \ dw_2 = dw_2/m \\
 db &= db/m
 \end{aligned}$$

$$\begin{aligned}
 \text{for } iter &\in \text{range}(1000): \leftarrow \\
 z &= w^T X + b \\
 &= np.dot(w.T, X) + b \\
 A &= \sigma(z) \\
 dz &= A - Y \\
 dw &= \frac{1}{m} X dz^\top \\
 db &= \frac{1}{m} \text{np.sum}(dz) \\
 w &:= w - \alpha dw \\
 b &:= b - \alpha db
 \end{aligned}$$

Figure 11: gradient_descent_lr_m_examples

General Principle

(m, n) <u>matrix</u>	\pm	$(1, n)$	$\rightsquigarrow (m, n)$
	\times	$(m, 1)$	$\rightsquigarrow (m, n)$

$(m, 1)$ $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$	$+$	R	$= \begin{bmatrix} 101 \\ 102 \\ 103 \end{bmatrix}$
$[1 \ 2 \ 3]$	$+$	100	$= [101 \ 102 \ 103]$

Figure 12: broadcasting

```
assert(a.shape == (5, 1))
## 如果不小心搞了个 rank 1 array, 也可以手动 a.reshape((5, 1))=a.reshape(5, 1)
```

2.7 quick tour of jupyter/ipython notebooks

2.8 explanation of logistic regression cost function

单个样本的 loss function, log 越大, loss 越小:

如果是 iid (独立同分布), 那么, m 个样本的 cost function, 其实就叫对数似然。对他求极大似然估计, 其实就是对 m 个样本求每个 cost function 的 min:

2.9 programming assignments

squeeze

```
np.squeeze(a, axis=None)
## 删掉维数是一的部分, axis 可以是 Int/int 数组, 表示只去掉指定下标的部分, 如果该部分维数不是 1, 会报错
x = np.array([[[0], [1], [2]]])
x.shape=(1,3,1)
np.squeeze(x)=array([0,1,2]) # shape=(3,)
np.squeeze(x, axis=(2,))=array([[0, 1, 2]]) # shape=(1,3)
```

把一个 shape 是 (a,b,c,d) 的 array 转成一个 type 是 (bcd,a) 的 array:

```
X_flatten = X.reshape(X.shape[0], -1).T
```

图片的预处理: + Figure out the dimensions and shapes of the problem (m_train, m_test, num_px, ...) + Reshape the datasets such that each example is now a vector of size (num_px * num_px * 3, 1) + “Standardize” the data: 对图片而言, 所有元素除以 255 就可以了

Logistic regression cost function

→ If $y = 1$: $p(y|x) = \hat{y}$

→ If $y = 0$: $p(y|x) = 1 - \hat{y}$

$p(y|x) = \hat{y}^y (1-\hat{y})^{(1-y)}$

If $y=1$: $p(y|x) = \hat{y}^1 (1-\hat{y})^0 = \hat{y}$

If $y=0$: $p(y|x) = \hat{y}^0 (1-\hat{y})^{(1-y)} = 1 \times (1-\hat{y}) = 1-\hat{y}$

$\uparrow \log p(y|x) = \log \hat{y}^y (1-\hat{y})^{(1-y)} = y \log \hat{y} + (1-y) \log (1-\hat{y})$

$= -\frac{1}{m} \sum_{i=1}^m \ell(\hat{y}_i, y_i)$

Figure 13: lr-loss

Cost on m examples

$$\log p(\text{labels in training set}) = \log \prod_{i=1}^m p(y^{(i)}|x^{(i)}) \leftarrow$$

$$\log p(\dots) = \sum_{i=1}^m \underbrace{\log p(y^{(i)}|x^{(i)})}_{-\mathcal{L}(\hat{y}^{(i)}, y^{(i)})}$$

$$= -\sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

maximum likelihood estimator ↗

极大似然估计:
max log p
即 min L(\hat{y}, y)

Cost: $J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Figure 14: lr-cost-m-examples