Machine Learning

by Sunny

Oct 29th

Week 1

Linear Regression with One Variable

一. Model and Cost Function

1. Cost Function

$h_\theta(x) = \theta_0 + \theta_1 x$

说明. θ_0, θ_1. $h_\theta(x)$ is close to y.

minimize $\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

θ_0, θ_1

m: training examples.

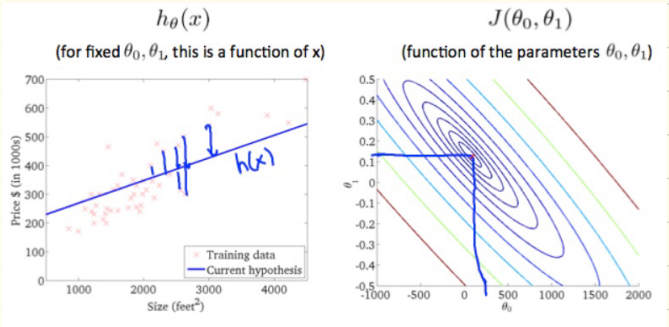
Cost Function

$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

Square Cost Function

goal: minimize $J(\theta_0, \theta_1)$.

2. contour plots.



Week 1

Linear Regression with One Variable

二. Parameter Learning

1. Gradient Descent

outline:

① start with some θ_0, θ_1

② keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ convergence 收敛.

⇒ tips.

$a := b$ assignment eg. $a := a+1$ ✓

$a = b$ truth assertion. eg. $a = a+1$ ✗

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

α: learning rate 下降幅度.

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

⇒ $\frac{\partial}{\partial \theta_j}$ 梯度 (微积分)

α { too small: gradient descent slow
too big: may overshoot the minimum

越接近最小值. 缩小 α.

走到 local minimum

走到的 minimum. 保留 local optimum θ_0, θ_1.

Week 1

Linear Regression with One Variable

2. Gradient Descent For Linear Regression

Gradient descent algorithm

repeat until convergence {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (for $j = 0$ and $j = 1$)

}

$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

$\Rightarrow \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$

统计算

θ_0 $j=0$: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$

θ_1 $j=1$: $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$

Gradient descent algorithm

repeat until convergence {

$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$

$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$

}

凸函数.
Convex function

∴ 基于原始代价函数

"This method looks at every example in the entire training set, and is called

batch gradient descent"

Week 2
Linear Regression with Multiple Variables

一、Multivariate Linear Regression

1. Multiple Features

Notation:

- n = number of features
- $x^{(i)}$ = input (features) of i^{th} training example.
- $x_j^{(i)}$ = value of feature j in i^{th} training example.

上标 superscript
下标 subscript

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

为了便 $x_0 = 1$

$$h_\theta(x) = [\theta_0\ \theta_1\ \cdots\ \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

transpose 转置

(特征向量)

2. Gradient Descent for Multiple Features

Cost function

$\vec{\theta}$ (n+1维向量)

$$J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

变形

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(\sum_{j=0}^{j=n} \theta_j x_j^{(i)} - y^{(i)} \right)^2$$

gradient descent

repeat until convergence : {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \quad \text{for } j := 0 \cdots n$$

}

Week 2
Linear Regression with Multiple Variables

3. Gradient Descent in Practise

<1> Feature Scaling & Mean Normalization

目的 $-1 \le x_{(i)} \le 1$ or $0.5 \le x_{(i)} \le 0.5$

方法 $x_i := \frac{x_i - \mu_i}{s_i}$

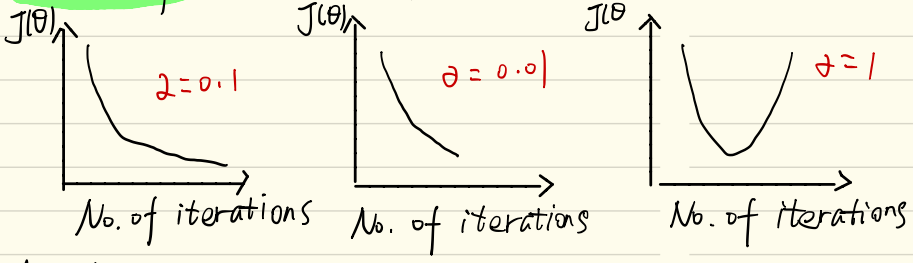
μ_i: $\overline{x}$ s_i: x_i 极差

特征缩放与均值归一化

give or take a few 多一点或少一点

<2> Learning Rate

An example. $\alpha = 0.01, 0.1, 1$.



Method

- try many α

triple

... 0.001 ^ 0.003 0.01 ^ 0.1 ^ 1 ...

- Automatic convergence test

if θ decrease less than E ✓ (difficult)

To summarize:

If α is too small: slow convergence.

If α is too large: may not decrease on every iteration and thus may not converge.

Week 2
Linear Regression with Multiple Variables

4. Features and Polynomial Regression

(aware of) combine features into one.

举例 注意形状

$$\begin{cases} h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 & \text{(quadratic function)} \\ h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 & \text{(cubic function)} \\ h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1} & \text{(square root function)} \end{cases}$$

Feature Scaling is important !

二. Computing Parameters Analytically

1. Normal Equation

gradient descent 和 normal equation → ~~feature scaling~~

本质都是一种求最优 θ 的方法.

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \quad m \times (n+1)$$

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \quad m\text{-dimensional vector}$$

$$\theta = (X^T X)^{-1} X^T y$$

pinv(X'*X)*X'*y

Comparision of the two method

Week 2
Linear Regression with Multiple Variables

G-D	N-E
need to choose α	~~α~~
need many iterations	~~iterate~~
$O(kn^2)$	$O(n^3)$
work well when $n\uparrow$	slow when $n\uparrow$

↓

generally $n > 10,000$

*2. Normal Equation Noninvertibility (不可逆)

normal equation

$$\theta = (X^TX)^{-1}X^Ty$$

若 X^TX 不可逆，

则 Octave 的可计算 pinv(X' * X) * X' * y

Reasons that cause N-E singular

What if X^TX is non-invertible?

- Redundant features (linearly dependent).
  E.g. x_1 = size in feet^2
  ~~x_2 = size in m^2~~
  $x_1 = (3.28)^2 x_2$
  $1_m = 3.28$ feet
  → $m = 10$ ←
  → $n = 100$ ←
  $\theta \in \mathbb{R}^{101}$
- Too many features (e.g. $m \le n$).
  - Delete some features, or use regularization.
    ↓ later

Week 3

Logistic Regression

一. Classification and Representation

1. Classification (Logistic)

$y \in \{0, 1\}$ 0: "Negative Class" 1: "Positive Class"

discrete 离散的

⇒ tips

0: absence of something that we want to see
eg. malignant tumor

1: presence ... , eg. benign tumor

2. Hypothesis Representation

Logistic Regression Model

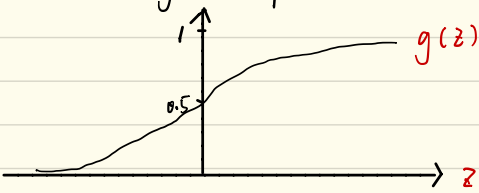
want $0 \leq h_\theta(x) \leq 1$

$h_\theta(x) = g(\theta^T x) \rightarrow g(z) = \frac{1}{1+e^{-z}}$ $z: \theta^T x$

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

called { Sigmoid function / Logistic function

$h_\theta(x)$: the probability when output is 1.



$$h_\theta(x) = P(y=1 \mid x;\theta) = 1 - P(y=0 \mid x;\theta)$$

$$P(y=1 \mid x;\theta) + P(y=0 \mid x;\theta) = 1$$

Week 3

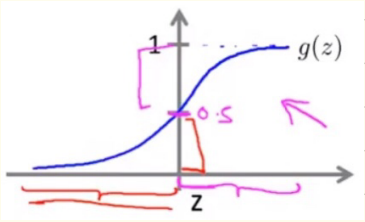
Logistic Regression

3. Decision Boundary

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

Suppose predict "y=1" if $h_\theta(x) \geq 0.5$

predict "y=0" if $h_\theta(x) < 0.5$



Desicion Boundary , to separate the area where y=0 and where y=1.

由 θ 决定，和数据集无关。

$h_\theta(x) \geq 0.5 \rightarrow y=1$

$h_\theta(x) < 0.5 \rightarrow y=0$

Remember

$$z=0, e^0=1 \Rightarrow g(z)=1/2$$
$$z \to \infty, e^{-\infty} \to 0 \Rightarrow g(z)=1$$
$$z \to -\infty, e^{\infty} \to \infty \Rightarrow g(z)=0$$

二. Logistic Regression Model

1. Cost Function

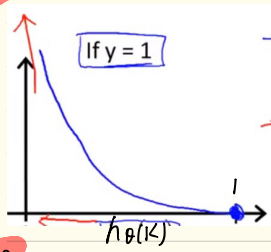
Week 3

Logistic Regression

Logistic regression cost function

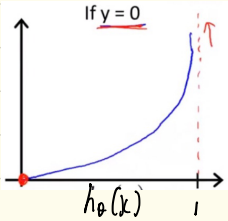
$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & y = 1 \\ -\log(1 - h_\theta(x)), & y = 0 \end{cases}$$

$y=1$



- Cost = 0 if $y = 1, h_\theta(x) = 1$
  - But as $h_\theta(x) \to 0$, $Cost \to \infty$
- Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

$y=0$



cost = 0 if $y=0, h_\theta(x) = 0$
But as $h_\theta(x) \to 1$
$\text{Cost} \to \infty$
if $h_\theta(x) = 1$, (predict $P(y=0|x;\theta) = 1$)
but $y=0 \Rightarrow$ a large cost

To summarize

$$\text{Cost}(h_\theta(x), y) = 0 \text{ if } h_\theta(x) = y$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$

This cost function guarantees that $J(\theta)$ is convex for Logistic regression.

2. Simplified Cost Function and Gradient Descent
   L-R cost function

Week 3

Logistic Regression

Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

y = 0 or 1 always

$$\Rightarrow \text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

Gradient Descent

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

repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$= \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left[(h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}\right]$$

Simultaneously update

}

同线性规划

→ 向量实现方式

$$\theta := \theta - \alpha/m \, X^T (g(X\theta) - \vec{y})$$

3. Advanced Optimization

Optimization algorithms:
- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:
- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:
- More complex

→ 调用内置函数即可，算法过程见后面.

eg.

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[options, functionVal, exitFlag] = fminunc(
    @costFunction, initialTheta, options);
```

Week 3

Logistic Regression

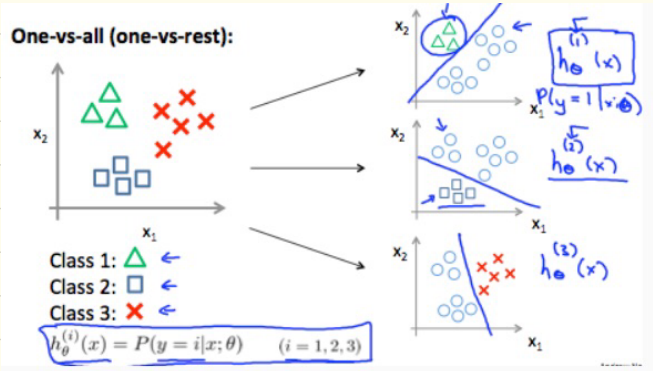
theta = $\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ theta(1) theta(2) theta(n+1) 返回值

```
function [jVal, gradient] = costFunction(theta)
    jVal = [code to compute J(θ)];
    gradient(1) = [code to compute ∂/∂θ0 J(θ)];
    gradient(2) = [code to compute ∂/∂θ1 J(θ)];
    ⋮
    gradient(n+1) = [code to compute ∂/∂θn J(θ) ];
```

三. Multiclass Classification

1. Multiclass Classfication : one vs all

one vs rest



$y \in \{0, 1, \cdots n\}$ ∴ $h_\theta^{(0)}x = P(y=0|x;\theta)$

∴ $h_\theta^{(n)}x = P(y=n|x;\theta)$

prediction = $\max_i (h_\theta^{(i)}(x))$

即，对 x 预测 y 值，取 $h_\theta^{(i)}(x)$ 最大者

Week 3

Logistic Regression

四. Solving the Problem of Overfitting

1. The Problem of Overfitting

underfit ✓ overfit

function is too simple, too few features uses

overfitting (caused by a complicated function)

work well for training sets, but predict bad.

Address overfitting

Options:

1. Reduce number of features.
   - → — Manually select which features to keep.
   - → — Model selection algorithm (later in course).
2. Regularization.
   - → — Keep all the features, but reduce magnitude/values of parameters θ_j.
   - — Works well when we have a lot of features, each of which contributes a bit to predicting y.

2. Cost Function

若得选上图中的overfitting. 则需 $\theta_3 \to 0$, $\theta_4 \to 0$

则 cost function

→ $\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$ 使得向2th图发展

$\Rightarrow \theta_3 \approx 0$, $\theta_4 \approx 0$

当不知道需从哪处减小 θ_i 时.

校正后的 cost function 如下页

Week 3

Logistic Regression

$$min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

若 λ too large → underfit
($\theta_1 \cdots \theta_n \downarrow, \approx 0$)
则 $h_\theta(x) = \theta_0$

λ: regularization parameter

3. Regularized Linear Regression

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

① Gradient Descent

repeat {

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \cdot \theta_j \right]$$

}

⇓ 比1稍小

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

② Normal Equation

$$\theta = (X^T X + \lambda L)^{-1} X^T \vec{y}$$

where

$$L = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix}$$

$(n+1) \times (n+1)$ dimension

note: 若 $m < n$, $X^T X$ 不可逆.
但 $+\lambda L$, 一定可逆

4. Regularized Logistic Regression

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Week 3

Logistic Regression

Gradient descent

Repeat {

$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$ separatedly

$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$

$(j = 1, 2, 3, \ldots, n)$

$\theta_1 \ldots \theta_n$

}

$\frac{\partial}{\partial \theta_j} J(\theta)$

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

Week 4

Neural Networks : Representation

一. Motivations

1. Non-linear Hypotheses

too many features

2. Neurons and the Brain

Neural Network origins:

Algorithms that try to mimic the brain.

二. Neural Networks

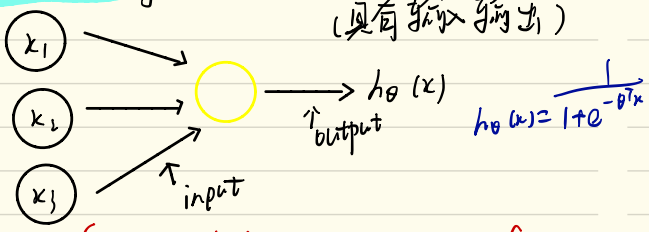
1. Model Representation I

① Neurons in the brain

{ 树突/细胞体接收刺激.
{ 轴突. 传出. (传递信号)

② Neural Network

model : Logistic unit



(具有输入输出)

x_0 : bias unit

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

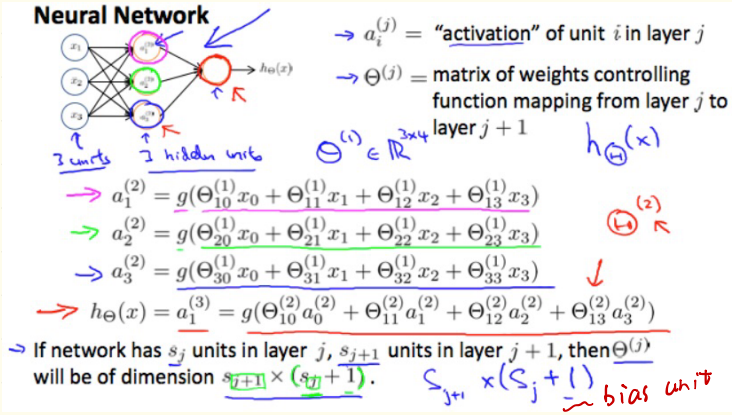
Sigmoid (logistic) activation function.

$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$ = "weight"

Neural Network

Week 4

Neural Networks : Representation



subscript 都對、的 layer

2. Model Representation II

Vectorized implementation

由上式 $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$

$z^{(2)} = \Theta^{(1)} x \Rightarrow a^{(2)} = g(z^{(2)})$

Setting $x = a^{(1)}$

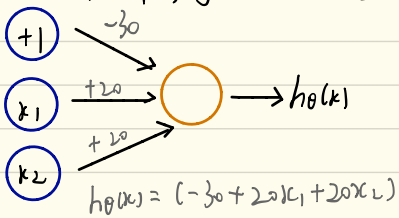
$\Rightarrow z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$ $a^{(j)} = g(z^{(j)})$

三. Applications

1. Examples and Intuitions I

Simple example : AND

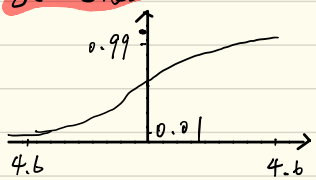
$x_1, x_2 \in \{0, 1\}$, $y = x_1$ AND x_2



Week 4

Neural Networks : Representation

g(z) 函数



x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

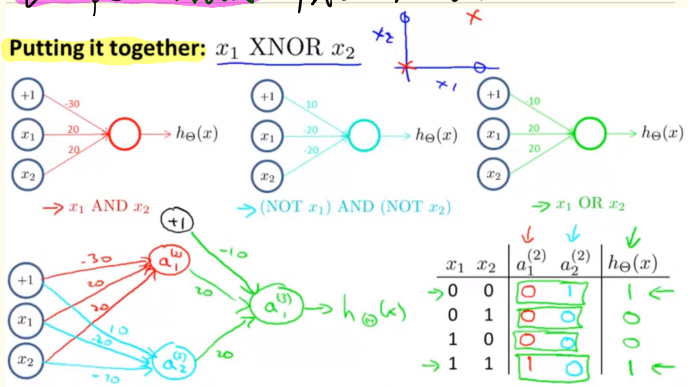
化简逻辑表达式. 取特值 0, 1 观察

Q: 将 $\Theta = [-10 \ 20 \ 20]$, 预测? (x_1 or x_2)

2. Examples and Intuition II

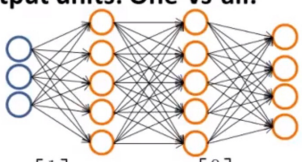
Example : XNOR 异或为 0. 反之为 1

Putting it together: x_1 XNOR x_2



3. Multiclass Classification

Multiple output units: One-vs-all.



$h_\Theta(x) \in \mathbb{R}^4$

Want $h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

表示可能类别

类别数

↑

输出层单元 n 个 unit

Week 4

Neural Networks : Representation

课后题

Which of the following statements are true? Check all that apply.

- [ ] A two layer (one input layer, one output layer; no hidden layer) neural network can represent the XOR function.

常进行查询可选事

- [x] Any logical function over binary-valued (0 or 1) inputs x_1 and x_2 can be (approximately) represented using some neural network.

- [ ] Suppose you have a multi-class classification problem with three classes, trained with a 3 layer network. Let $a_1^{(3)} = (h_\Theta(x))_1$ be the activation of the first output unit, and similarly $a_2^{(3)} = (h_\Theta(x))_2$ and $a_3^{(3)} = (h_\Theta(x))_3$. Then for any input x, it must be the case that $a_1^{(3)} + a_2^{(3)} + a_3^{(3)} = 1$.

由前一层计算所得可最终得到

- [x] The activation values of the hidden units in a neural network, with the sigmoid activation function applied at every layer, are always in the range (0, 1).

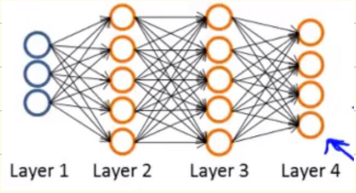
Week 5

Neural Networks : Learning

一、Cost Function and Backpropagation

1. Cost Function

Neural Network (Classification)



定义：

L：总层数

s_l：第 l 层的units数

K：输出units/classes数

K分类模型是K个0-1分类，则有K分类代价函数　$h_\Theta(x)_k$[输出第k个...]

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log\left((h_\Theta(x^{(i)}))_k\right) + (1-y_k^{(i)})\log\left(1-(h_\Theta(x^{(i)}))_k\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$$

注：Θ矩阵行数为当前层node个数，列数为下一层node个数

Note

- the double sum simply adds up the logistic regression costs calculated for each cell in the output layer
- the triple sum simply adds up the squares of all the individual Θs in the entire network.
- the i in the triple sum does **not** refer to training example i

2. Backpropagation

Gradient computation

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)}\log h_\theta(x^{(i)})_k + (1-y_k^{(i)})\log(1-h_\theta(x^{(i)})_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_j^{(l)})^2$$

cost function

$\min_\Theta(J(\Theta))$　need code to compute $J(\Theta)$, $\frac{\partial}{\partial\Theta_{i,j}^{(l)}}J(\Theta)$

Week 5

Neural Networks : Learning

Backpropagation algorithm

$\delta_j^{(l)}$ = "error" of node j in layer l

for total L layer. $\delta_j^{(L)} = a_j^{(L)} - y_j$

$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$ → $a^{(3)} .* (1-a^{(3)})$

$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$ → $a^{(2)} .* (1-a^{(2)})$

⇒ $\delta^{(l)} = (\Theta^{(l)})^T \delta^{(l+1)} .* a^{(l)} .* (1-a^{(l)})$

算法步骤

Training set $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j). (used to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$)

For $i = 1$ to m ← $(x^{(i)}, y^{(i)})$

- Set $a^{(1)} = x^{(i)}$
- Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \dots, L$ (向前)
- Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$ (就是 $\delta^{(1)}$)
- $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ ← $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

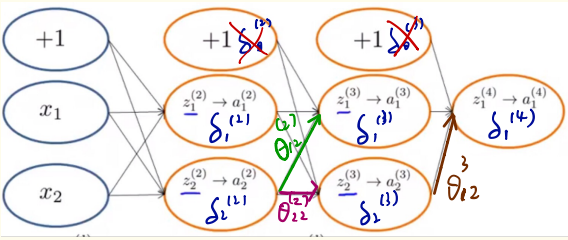
$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0$

$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$

$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$

Δ：δ的大写

3. Backpropagation Intuition.



$\delta_1^{(4)} = y^{(i)} - a_1^{(4)}$ (for example i)

有 $\delta_2^{(2)} = \Theta_{12}^{(2)} \delta_1^{(3)} + \Theta_{22}^{(2)} \delta_2^{(3)}$

$\delta_2^{(3)} = \Theta_{12}^{3} \delta_1^{(4)}$

⋮

Week 5

Neural Networks : Learning

对K=1的分类问题, for example t

$cost(t) = y^{(t)} \log(h_\Theta(x^{(t)})) + (1-y^{(t)}) \log(1-h_\Theta(x^{(t)}))$

$\Rightarrow \delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} cost(t)$? 不太懂

二. Backpropagation in Practice

1. Implementation Note: Unrolling Parameters

Unroll into a long vector

eg. thetaVector = [Theta1(:); Theta2(:); Theta3(:)]

get back: Theta1 = reshape(thetaVector(1:110), 10, 11)

取1到110列 变成 10×11 矩阵

To summarize:

Learning Algorithm

- Have initial parameters $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$.
- Unroll to get `initialTheta` to pass to
- `fminunc(@costFunction, initialTheta, options)`

```
function [jval, gradientVec] = costFunction(thetaVec)
    From thetaVec, get Θ(1), Θ(2), Θ(3).
    Use forward prop/back prop to compute D(1), D(2), D(3) and J(Θ).
    Unroll D(1), D(2), D(3) to get gradientVec.
```

2. Gradient Checking

检验subtle bugs

以近似方法估计梯度, 与backpop计算出的对比.

$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}$ = gradApprox

单侧双侧都可算

算法过程

```
for i = 1:n,
  thetaPlus = theta;
  thetaPlus(i) = thetaPlus(i) + EPSILON;
  thetaMinus = theta;
  thetaMinus(i) = thetaMinus(i) - EPSILON;
  gradApprox(i) = (J(thetaPlus) - J(thetaMinus))
                  /(2*EPSILON);
end;
```

$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_i + \varepsilon \\ \vdots \\ \theta_n \end{bmatrix}$

check gradApprox ≈ Dvec

↑from backpop

Week 5

Neural Networks : Learning

Implementation Note:

- Implement backprop to compute `DVec` (unrolled $D^{(1)}, D^{(2)}, D^{(3)}$)
- Implement numerical gradient check to compute `gradApprox`.
- Make sure they give similar values.
- Turn off gradient checking. Using backprop code for learning.

Important:

- Be sure to disable your gradient checking code before training your classifier. If you run numerical gradient computation on every iteration of gradient descent (or in the inner loop of `costFunction(...)`) your code will be very slow.

3. Random Initialization

Random initialization: Symmetry breaking

Initialize each $\Theta_{ij}^{(l)}$ to a random value in $[-\epsilon, \epsilon]$ (i.e. $-\epsilon \le \Theta_{ij}^{(l)} \le \epsilon$)

E.g.

```
Theta1 = rand(10,11)*(2*INIT_EPSILON)
         - INIT_EPSILON;

Theta2 = rand(1,11)*(2*INIT_EPSILON)
         - INIT_EPSILON;
```

Why zeros(n, 1)

△ After each update, parameters corresponding to inputs going into each of hidden unit are identical.

4. Putting It Together

Training a neural network (a network archit)

① No. of input units : Dimension of features $x^{(i)}$

② No. of output units : Number of classes

③ Reasonable default : 1 hidden layer

or >1 hidden && have same no. of hidden units in every layer.

Week 5

Neural Networks : Learning

Training a Neural Network

1. Randomly initialize the weights
2. Implement forward propagation to get $h_\Theta(x^{(i)})$ for any $x^{(i)}$
3. Implement the cost function
4. Implement backpropagation to compute partial derivatives
5. Use gradient checking to confirm that your backpropagation works. Then disable gradient checking.
6. Use gradient descent or a built-in optimization function to minimize the cost function with the weights in theta.

Ⅲ. Application of Neural Networks

1. Autonomous Driving Example