Coursera – Machine Learning (taught Andrew Ng) – Notes by Xiaofan Wang

Name: Xiaofan Wang

Chapter 1.

一. What's Machine Learning

1. Two definitions

Old: The field of study that gives computers the ability to learn without being explicitly programmed.

"计算机无须进行明确编程即可学习"的领域

Modern: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if it is performance at tasks in T, as measured by P, improves with experience E."

如果某计算机程序在T任务中的性能(由P衡量)随着经验E的提高而提高，则说它是从经验E中学习有关某类任务T和性能度量P的.

Example: Playing Checkers 玩跳棋

E = experience of playing many games

P = probability that program will win next game.

T = task of playing checkers.

2. ML Algorithm

Supervised learning & Unsupervised Learning.

二. Supervised Learning.

已知: Dataset & correct Output

探求: Relationship between "input" and "output"

Regression: predict continuous "output" (values)

Classification: predict discrete "output" (categories)

三. Unsupervised learning.

Derive structure from data, don't necessarily know the effect of variables. No feedback on prediction results.

Derive structure by clustering data based on relationships among variables in the data.

基于变量之间的关系对数据进行聚类来获得此结构

不必知道变量的影响. 没有基于预测结果的反馈.

Example:

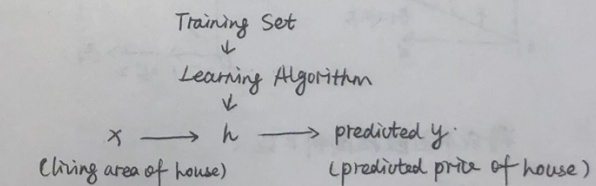
Clustering: Take a collection of 1000--+ different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables (e.g. lifespan. location. roles--)

收集许多不同基因. 并找到一种方法将基因组自动分组. (相似. 相关)

Non-clustering: find structure in a chaotic environment. (e.g. identify individual voices and music from a mesh of sounds at cocktail party).

四. Model and Cost Function.

1. Model Representation.

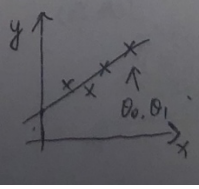


2. Cost Function

Measure the accuracy of our hypothesis function.

This takes an average difference of all the results of the hypothesis with inputs from x's and the actual output y's

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



Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for training examples (x, y)

Hypothesis: $h_\theta(x)=\theta_0+\theta_1 x$ / $h_\theta(x)=\theta_1 x$

Parameters: θ_0, θ_1 / θ_1

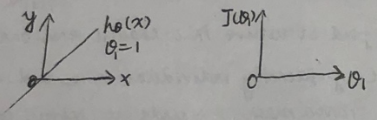
Cost Function: $J(\theta_0,\theta_1)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$

Goal: $\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$ / $\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

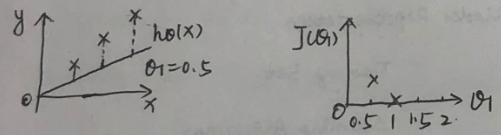
[redacted]

1. 训练数据集散布在 xy 平面上. 尝试画条直线 ($h_\theta(x)$) 穿过这些分散的数据点.
2. Best possible: Average squared vertical distances of the scatter points from the line will be the least.

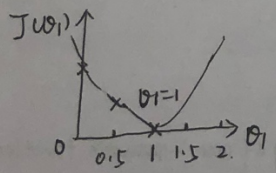
最佳: 散射点与该线的平均垂直距离最小. $J(\theta_0,\theta_1)=0$



3. $\theta_1=1$ 时, 每个数据点的斜率均为 1.
$\theta_1=0$ 时, 从拟合到数据点的垂直距离增加了.



将成本函数提高到 0.58:



最小量减小成本函数.

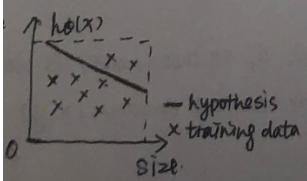
$\theta_1=1$ 是 global minimum.

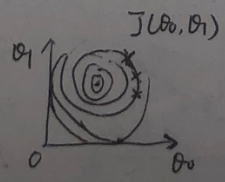
4. Contour Plot

等高线:

同一条线上的所有点具有恒定值.

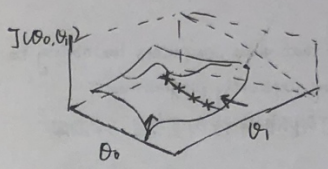
A contour line of a two variable function has a constant value at all points of the same line.





五. Parameter Learning.

1. 衡量假设函数与数据的拟合程度: 估计假设函数中的参数 (梯度下降)



成本函数位于凹坑最底部时, 其值最小.

采用成本函数的导数 (函数的切线). 切线斜率是该点的导数. 它将为我们提供一个方向. 沿下降最陡的方向逐步降低成本函数. (由 para α 学习率决定)

Para α: 步长.

$$\theta_j=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)$$

$$\text{temp}0=\theta_0-\alpha\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)$$

$$\text{temp}1=\theta_1-\alpha\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1)$$

$$\theta_0=\text{temp}0$$

$$\theta_1=\text{temp}1$$

① 许 α is too small, gradient descent can be slow.

② 许 α is too large, gradient descent can overshoot the minimum. It may fail to converge. or even diverge. 收敛. 发散.

2. 线性回归的梯度下降

$$\frac{\partial}{\partial\theta_j}J(\theta)=\frac{\partial}{\partial\theta_j}\cdot\frac{1}{2}(h_\theta(x)-y)^2$$

$$=\frac{1}{2}\cdot 2(h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}(h_\theta(x)-y)$$

$$=(h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i-y\right)$$

$$=(h_\theta(x)-y)\,x_j$$

椭圆是二次函数的轮廓. 图中的 x 标记了梯度下降收敛到最小值时经历的 θ 的连续值.

Name: Xiaofan Wang

Chapter 2. Linear Regression with Multiple Variables

一. Multiple Features.

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

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

二. Gradient Descent for Multiple Variables.

1. The gradient descent equation itself is generally the same form. repeat it for "n" features.

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

$$\theta_1 = \theta_1 - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 = \theta_2 - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

$$\vdots$$

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

2.
- hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$
- Parameters: $\theta_1, \theta_2 \cdots \theta_n$. ($\vec{\theta}$)
- cost function: $J(\theta_0 \cdots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

三. Gradient Descent (Feature Scaling) I.

1. Feature Scale: Make sure features are on a similar scale.

Get every feature into approximately a $-1 \le x_i \le 1$ range.

2. Mean Normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean.

$$x_i = \frac{x_i - \mu_i}{s_i}$$

四. Gradient Descent II (Learning Rate).

1. Debugging gradient descent

Make a plot with number of iterations on the x-axis.

Now plot the cost function, $J(\theta)$ over the number of iterations of gradient descent. If $J(\theta)$ ever increases, ~~they~~ then you probably need to decrease α.

调试梯度下降：绘制一个在x轴上有迭代次数的图.

现在在梯度下降的迭代次数上绘制成本函数 $J(\theta)$.

如果 $J(\theta)$ ↑. 则 ↓ α.

2. Automatic convergence test.

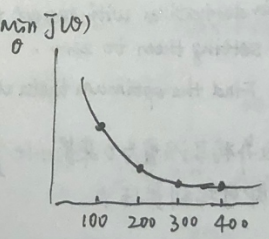
Declare convergence if $J(\theta)$ decreases by less than E in on iteration, where E is small value such as 10^{-3}.

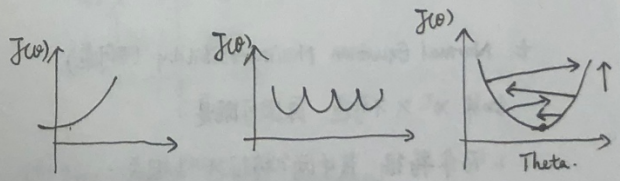
However, in practice it's difficult to choose this threshold value.

自动收敛测试：如果在一次迭代中 $J(\theta)$ ↓ < E.

则声明收敛. 其中E是一些小值. 例如 10^{-3}.

但实际上，很难选择此阈值.





- For sufficiently small α, $J(\theta)$ should decrease on every iteration. 如果α太小，收敛慢.
- But if α is too small, gradient descent can be slow to converge.

如果α太大，不会每次迭代都减小. 可能不会收敛

五. Features & Polynomial Regression.

$H_\theta(x) = \theta_0 + \theta_1 x_1$

$H_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$.

$H_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$

$H_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$

六. Normal Equation.

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

Gradient Descent.	VS.	Normal Equation.
Need to choose alpha.		No Need to choose alpha.
Need many iterations.		No need to iterate.
$O(kn^2)$.		$O(n^3)$, calculate $X^T \cdot X$
work well when n is large.		Slow if n is very large.

1. Gradient Descent : Give one way of minimizing.

2. Normal Equation : minimize J by explicitly taking its ~~devia~~ derivatives with respect to θ_j's and setting them to zero.
=> Find the optimum theta without iteration.

针对 θ_j 取导数并将其设置为 0 来最小化 J.
使得无需迭代即可找到最佳 θ.

七. Normal Equation Noninvertibility (不可逆)

如果 $X^T \cdot X$ 不可逆. 原因可能是：

1. 冗余特征：其中两个特征线性相关.

2. 特征过多（删除某些功能用正则化）.

Name: Xiaofan Wang

Chapter 3. Classification.

一. Classification and Representation.

1. Classification.

(1) $y \in \{0, 1\}$. 0: "Negative Class" (e.g. benign tumor)

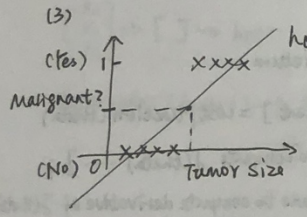
1: "Positive Class" (e.g. malignant tumor)

(2) Example: Email (Spam / Not Spam)

Online Transactions: Fraudulent (Yes / No)

Tumor: ~~Mag~~ Malignant / Benign.

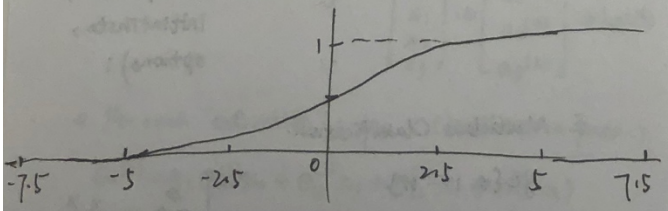
(3)



2. Hypothesis's Representation. 假设表示.

Logistic Function (Sigmoid Function)

$$\begin{cases} h_\theta(x) = g(\theta^T \cdot x) & 0 \le h_\theta(x) \le 1 \\ z = \theta^T \cdot x \\ g(z) = \frac{1}{1+e^{-z}} \end{cases}$$



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

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

3. Decision Boundary. 决策边界.

(1) Logistic Regression: $h_\theta(x) = g(\theta^T \cdot x)$

$$g(z) = \frac{1}{1+e^{-z}}$$

$$\Downarrow$$

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

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

$$\begin{cases} z=0, & g(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^0} = \frac{1}{2} = 0.5. \\ z=\infty, & g(z) = \frac{1}{1+\frac{1}{e^\infty}} = 1 \\ z=-\infty, & g(z) = \frac{1}{1+e^\infty} = 0. \end{cases}$$

$g(z) \ge 0.5$. When $z \ge 0$

$h_\theta(x) = g(\theta^T \cdot x) \ge 0.5$ when $\theta^T \cdot x \ge 0$

$$\Downarrow$$

$\theta^T \cdot x \ge 0 \rightarrow y=1$

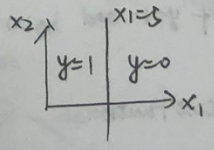
$\theta^T \cdot x < 0 \rightarrow y=0$

(2) ~~Desi~~ Decision Boundary.

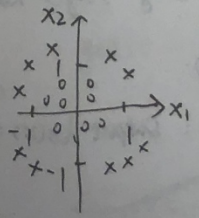
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

By: $\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$, $h_\theta(x) = g(5 - x_1)$

$5 - x_1 \ge 0$. $x_1 \le 5$.



(3) Non-linear Decision Boundary.



$z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$

$h_\theta(x) = g(\theta_0 + \theta_1 x_1^2 + \theta_2 x^2 \cdots)$

O: +
X: -

二. Logistic Regression Model

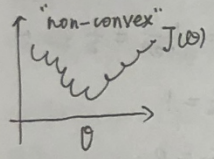
1. Training Set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots (x^{(m)}, y^{(m)})\}$

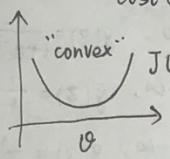
$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

2. cost Function

(1) Logistic Regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$

$\text{cost}(h_\theta(x^{(i)}) - y^{(i)})$



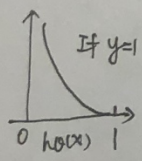


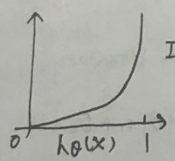
(2) cost func for LR: ($y=0/1$ always)

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

$\text{cost}(h_\theta(x), y) = -\log(h_\theta(x))$ If $y=1$.

$\text{cost}(h_\theta(x), y) = -\log(1-h_\theta(x))$ If $y=0$.





$\text{cost}(h_\theta(x), y) = 0$. If $h_\theta(x) = y$.

$\text{cost}(h_\theta(x), y) \to \infty$ If $y=0$. and ~~Rest~~ $h_\theta(x) \to 1$

$\text{cost}(h_\theta(x), y) \to \infty$. If $y=1$. and $h_\theta(x) \to 0$.

3. ~~cost~~ Logistic Regression Cost Function.

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

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

To fit parametres θ: $\min_\theta J(\theta)$.

To make a prediction given new x: Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

4. Gradient Descent.

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

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

A vectorized implementation: $\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$

5. Advanced Algorithm 高级优化

"共轭梯度" / "BFGS" / "L-BFGS" ⇒ 是更复杂、更快速的优化θ的方法. 可用于梯度下降. (不需自己编写复杂算法, 较调一些库)

(1) Func: $J(\theta)$. $\frac{\partial}{\partial \theta_j} J(\theta)$

Write a single func to return:

```
>> function [jVal, gradient] = costFunction(theta)
>>     jVal = [...code to compute J(theta)...];
>>     gradient = [...code to compute derivative of J(theta)...];
>> end.
```

(2) optimization Algorithm: fminunc(), optimset()

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

三. Multiclass Classification.

$y \in \{0, 1 \cdots n\}$

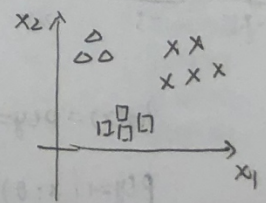
$h_\theta^{(0)}(x) = P(y=0 \mid x; \theta)$

$h_\theta^{(1)}(x) = P(y=1 \mid x; \theta)$

$\vdots$

$h_\theta^{(i)}(x) = P(y=i \mid x; \theta)$

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



Name: Xiaofan Wang

Chapter 4. Neural Networks

一. Model Representation. (I)

1. Neurons are basically computational units that take inputs as electrical inputs that are channeled to outputs.
(dendrites 树突) (axons 轴突)

神经元是计算单元，将输入(树突)作为输入.
并且输出到输出(轴突)的计算单元.

2. In the model, input features: $x_1, x_2 \cdots x_n$
output is the result of hypothesis function: $\frac{1}{1+e^{-\theta^T x}}$.
(activation function) (θ: weights)

$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow [\ ] \rightarrow h_\theta(x)$

Input nodes (layer 1) → another node (layer 2) → hypothesis funtion (output layer)

Intermediate layers: hidden layers. ($a_0^2 \cdots a_n^2$ activation units)
(Between input & output layer)

$\begin{cases} a_i^{(j)} = \text{"activation" of unit } i \text{ in layer } j \\ \Theta^{(j)} = \text{matrix of weights controlling function mapping from layer } j \text{ to layer } j+1 \end{cases}$

3. If having on hidden layer: $\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \rightarrow h_\theta(x)$

4. For each "activation" nodes: (3×4 matrix para-)

$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$

$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$

$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$

$h_\theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$

$\Theta^{(j)}$ dimension: $S_{j+1} \times (S_j + 1)$

二. Model Representation (II. replace parameter with "z")

$a_1^{(2)} = g(z_1^{(2)})$

$a_2^{(2)} = g(z_2^{(2)})$

$a_3^{(2)} = g(z_3^{(2)})$

1. For layer $j=2$. node k:

$z_k^{(2)} = \Theta_{k,0}^{(1)} x_0 + \Theta_{k,1}^{(1)} x_1 + \cdots + \Theta_{k,n}^{(1)} x_n$

2. The vector representation of x, z^j:

$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix} \Rightarrow z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$

$z^{(j+1)} = \Theta^{(j)} a^{(j)}$

3. Final result: $h_\theta(x) = a^{(j+1)} = g(z^{(j+1)})$

三. Examples and Intuitions (I)

1. AND.

$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow [\ g(z^{(2)})\ ] \rightarrow h_\theta(x)$

$\Theta^{(1)} = [-30 \ \ 20 \ \ 20]$.

$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$

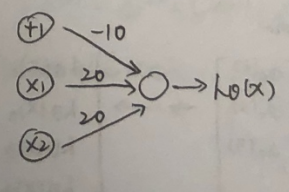
$x_1 = 0$ and $x_2 = 0$. $g(-30) \approx 0$

$x_1 = 0$ and $x_2 = 1$ then $g(-10) \approx 0$.

$x_1 = 1$ and $x_2 = 0$. then $g(-10) \approx 0$

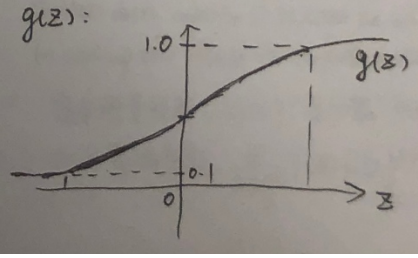
$x_1 = 1$ and $x_2 = 1$ then $g(10) \approx 1$

2. OR.



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

$g(z)$:



Ⅲ. Examples and Intuitions (Ⅱ)

1. AND: $\Theta^{(1)} = [-30 \;\; 20 \;\; 20]$.
   NOR: $\Theta^{(1)} = [10 \;\; -20 \;\; -20]$
   OR: $\Theta^{(1)} = [-10 \;\; 20 \;\; 20]$.

2. combine these to "XNOR":

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} \rightarrow [a^{(3)}] \rightarrow h_\Theta(x)$$

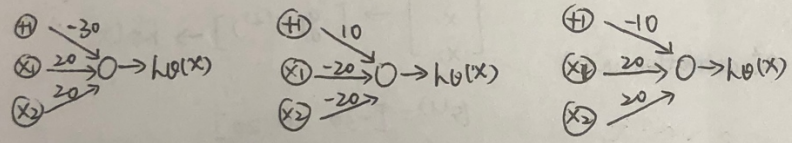
3. $\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \\ 10 & -20 & -20 \end{bmatrix}$. $\quad \Theta^{(2)} = [-10 \;\; 20 \;\; 20]$

4. $a^{(2)} = g(\Theta^{(1)} x)$

   $a^{(3)} = g(\Theta^{(2)} a^{(2)})$

   $h_\Theta(x) = a^{(3)}$

5. put it together: x_1 XNOR x_2



x_1 AND x_2. (NOT x_1) AND (NOT x_2) x_1 OR x_2.

Ⅳ. Multiclass Classification.

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

Final hypothesis function:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(3)} \\ a_1^{(3)} \\ a_2^{(3)} \\ \vdots \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} h_\Theta(x)_1 \\ h_\Theta(x)_2 \\ h_\Theta(x)_3 \\ h_\Theta(x)_4 \end{bmatrix}$$

Name: Xiaofan Wang

Chapter 5. Neural Networks: Learning.

一. Cost Function. (generalization of Logistic regression)

1. define variables:

L: total number of layers in the network 层数

S_l: number of units in layer l.

k: number of output units

2. For regularized logistic regression:

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

For Neural Networks, it's more complicated:

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

① 添加 j 一些数值求和，该求和遍历输出节点的数量.

② 看总多少个 Θ matrix. { 列数 = 当前层的节点数. / 行数 = 下一层的节点数.

3. Notes.

① Double sum 总 adds up the logistic regression costs calculated for each cell in the output layer.

② Triple sum adds up the squares of all the individual Θs in the entire network.

③ the i in the triple sum does not refer to training example.

二. Backpropagation Algorithm. (minimize cost function)

Min $J(\Theta) \rightarrow \frac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta)$

1. Training Set $\{(x^{(1)}, y^{(1)}), \cdots (x^{(m)}, y^{(m)})\}$

Set $\Delta_{i,j}^{(l)}=0$ for all (l,i,j)

For training example $t=1$ to m: Set $a^{(1)}=x^{(t)}$

2. perform forward propagation to compute $a^{(l)}$ for $l=2,3\cdots L$

Gradient Computation:

$a^{(1)}=x$

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

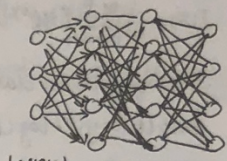
$a^{(2)}=g(z^{(2)})$ (add $a_0^{(2)}$)

$z^{(3)}=\Theta^{(2)}a^{(2)}$

$a^{(3)}=g(z^{(3)})$ (add $a_0^{(3)}$)

$z^{(4)}=\Theta^{(3)}a^{(3)}$

$a^{(4)}=h_\Theta(x)=g(z^{(4)})$



Layer 1 L_2 L_3 L_4

$a^{(1)}$ $a^{(2)}$ $a^{(3)}$ $a^{(4)}$

3. Using $y^{(t)}$, compute $\delta^{(L)}=a^{(L)}-y^{(t)}$

$a^{(L)}$: vector output of activation units for last year.

To get the $\delta^{(L)}$ before last layer, back from right to left

4. Compute $\delta^{(L-1)}, \delta^{(L-2)} \cdots \delta^{(2)}$

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

$(\Theta^{(l)})^T\delta^{(l+1)}$: 下一层增量 * Θ Matrix = 第一层增量值

$a^{(l)}.*(1-a^{(l)})$: $g'(z^{(l)})=a^{(l)}.*(1-a^{(l)})$ activation function.

5. $\begin{cases}\Delta_{i,j}^{(l)}=\Delta_{i,j}^{(l)}+a_j^{(l)}\delta_i^{(l+1)} \\ \Delta^{(l)}=\Delta^{(l)}+\delta^{(l+1)}(a^{(l)})^T\end{cases}$

Update New Matrix: $\begin{cases}D_{i,j}^{(l)}=\frac{1}{m}(\Delta_{i,j}^{(l)}+\lambda\Theta_{i,j}^{(l)}), \text{ if } j\neq 0 \\ D_{i,j}^{(l)}=\frac{1}{m}\Delta_{i,j}^{(l)}, \text{ if } j=0.\end{cases}$

Capital-delta matrix D is used as an "accumulator" to add up our values → compute partial ~~delta~~ derivative.

大本增量矩阵D被用于"累加器"来求和.

以计算偏导数 $\frac{\partial}{\partial\Theta_{i,j}^{(l)}}J(\Theta)=D_{i,j}^{(l)}$

三. Backpropagation Intuition.

1. Cost Function for NN:

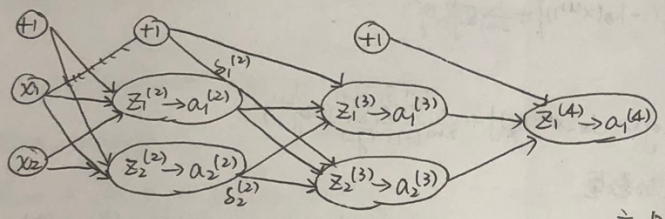
$$J(\Theta) = \frac{-1}{m}\sum_{t=1}^{m}\sum_{k=1}^{K}\left[y_k^{(t)}\log\left(h_\Theta(x^{(t)})\right)_k + (1-y_k^{(t)})\log\left(1-h_\Theta(x^{(t)})_k\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{j,i}^{(l)}\right)^2$$

If non-multiclass classification ($k=1$) and disregard regularization:

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

$$\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}}cost(t)$$

2. Forward ~~pro~~ propagation:



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

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

四. Backpropagation Implementation: Unrolling Parameters.

1. Use Optimizing function "fminunc()":

fminunc(@costFunction, initialTheta, options)

function [jval, gradientVec] = costFunction(thetaVec)

2. From thetaVec: $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$

Use forward prop/back prop: $D^{(1)}, D^{(2)}, D^{(3)}, J(\Theta)$.

↓ gradientVec

3. Matrix: $\begin{bmatrix}\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}\ldots \\ D^{(1)}, D^{(2)}, D^{(3)}\ldots\end{bmatrix}$

put into long Vector. $\begin{bmatrix}thetaVector = [Theta1(:); Theta2(:); Theta3(:);] \\ deltaVector = [D1(:); D2(:); D3(:);]\end{bmatrix}$

4. If the dimension of Theta1 (10×11), Theta2 (10×11), Theta3 (1×11).

Get Back original matrixs

$$\begin{cases}Theta1 = reshape(thetaVector(1:110), 10, 11) \\ Theta2 = reshape(thetaVector(111:220), 10, 11) \\ Theta3 = reshape(thetaVector(221:231), 1, 11)\end{cases}$$

五. Gradient Checking.

1. Approximate the derivative of Cost Function:

$$\frac{\partial}{\partial\Theta}J(\Theta) \approx \frac{J(\Theta+\varepsilon) - J(\Theta-\varepsilon)}{2\varepsilon}.$$

2. With multiple theta matrix:

$$\frac{\partial}{\partial\Theta_j}J(\Theta) \approx \frac{J(\Theta_1\cdots\Theta_j+\varepsilon\cdots\Theta_n) - J(\Theta_1\cdots\Theta_j-\varepsilon\cdots\Theta_n)}{2\varepsilon}.$$

六. Random Initialization. (symmetry breaking).

Initial each $\Theta_{ij}^{(l)}$ to a random value in $[-\varepsilon, \varepsilon]$

$(-\varepsilon \le \Theta_{ij}^{(l)} \le \varepsilon)$.

```
Theta1 = rand(10,11) * (2*INIT_EPSILON) - INIT_EPSILON
Theta2 = rand(1,11) * (2*INIT_EPSILON) - INIT_EPSILON
```

七. Pick a network architecture.

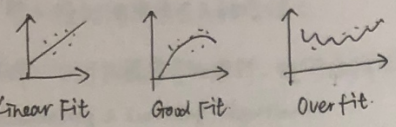
- Number of input units = dimension of features $x^{(i)}$
- Number of output units = number of classes.
- Number of hidden units per layer = usually more the better.
- Defaults: 1 hidden layer. (If >1, same number of units in every hidden layer).

八. Training a Neural Network.

1. Randomly initialize the weights.
2. Implement forward ~~implement~~ propagation → $h_\Theta(x^{(i)})$
3. Implement Cost Function.
4. Implement backpropagation → partial derivatives
5. Gradient Checking. → Then disable it.
6. Gradient descent / optimization function to minimize the cost function. with the weights in theta.

[2]. Solving the Problem of Overfitting

1. The problem of over-fitting.



(1) Underfitting - hypothesis func h maps poorly to the trend of the data. (Too simple func. too few features)

(2) over fitting: not generalize well to predict new data.
(Too complicated func, ~~to~~ create unnecessary curves and angles unrelated to data)

(3) To address overfitting:

① reduce the number of features:
- Manually select which features to keep.
- use a model selection algorithm

② Regularization:
- keep all features, but reduce the magnitude of θ_j
- Regularization works well when we have lots of slightly useful features.

2. cost Function.

$\theta_0 + \theta_1 \cdot x + \theta_2 \cdot x^2 + \theta_3 \cdot x^3 + \theta_4 \cdot x^4$

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

regularize: $\lambda \sum_{j=1}^{n} \theta_j^2$

(when λ too large → under fitting)

3. Regularized Linear Regression.

(1) Gradient Descent.

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

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

$j \in \{1, 2 \cdots n\}$.

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

(2) Normal Equation.

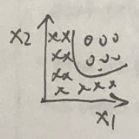
$$\theta = (X^T \cdot X + \lambda \cdot L)^{-1} \cdot X^T \cdot y$$

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

如果 $m < n$, 则 $X^T \cdot X$ 是不可逆的.

但当添加 $\lambda \cdot L$ 时, $X^T \cdot X + \lambda L$ 变为可逆的.

4. Regularized Logistic Regression.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 \cdots)$$

Cost Function:

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

regularize this equation by adding a term:

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

Name: Xiaofan Wang

Chapter 6. Apply Machine Learning.

-些人可以高效运用这些学习算法.

-些人将时间浪费在无效尝试上.

本课：对于机器学习的设计，如何选择正确的道路.

-. Evaluating a Learning Algorithm.

1. 若 hypothesis makes unacceptably large errors in its prediction.

- Get more training examples
- Try smaller sets of features.
- Try getting additional features.
- Try adding polynomial features ($x_1^2, x_2^2, x_1 \cdot x_2, \cdots$)
- Try decreasing λ
- Try increasing λ.

⇓

Machine Learning Diagnose.

2. 假 hypothesis evaluation.

A hypothesis may have a low error for training examples but still be inaccurate (because of overfitting).

Typically { 70% training set. / 30% test set.

⇓

New procedure using two sets:

① Learn θ, minimize $J_{train}(\theta)$ using training set.

② compute the test set error $J_{test}(\theta)$

3. Test Error.

① For linear regression: $J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)} - y_{test}^{(i)})^2$

② For classification: (Misclassification Error).

$$err(h_\theta(x), y) = \begin{cases} 1 & (\text{if } h_\theta(x) \geq 0.5 \text{ and } y=0 \\ & \text{or } h_\theta(x) < 0.5 \text{ and } y=1) \\ 0 & \end{cases}$$

$$\text{Test Error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(h_\theta(x_{test}^{(i)}), y_{test}^{(i)})$$

This gives the proportion of test data that was misclassified.

4. Model Selection. & Train. Validation. Test Sets.

① Overfitting problem.

② Model Selection.

d = degree of polynomial.

先选择第一个模型，然后求训练误差的最小值.
可得一个参数向量 θ.

再选第二个模型，过程同理，得到另一个参数向量 θ.

③ Break down the dataset into three sets:

Training set: 60%. $(x^{(1)}, y^{(1)}) \cdots (x^{(m)}, y^{(m)})$

Cross Validation set: 20%. $(x_{cv}^{(1)}, y_{cv}^{(1)}) \cdots (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

Test Set: 20%. $(x_{test}^{(1)}, y_{test}^{(1)}) \cdots (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

④ Train/ Validation/ Test error

Train error: $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Cross Validation error: $J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$

Test error: $J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$

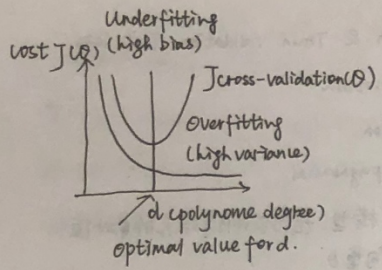
- Optimize the parameters in θ using the training set for each polynomial degree. 对每个多项式的训练集优化参数 θ
- Find the polynomial degree d with the least error using the cross validation set. ~~(基于 $J_{cv}(\theta)$ 来找出最小误差.~~ 使用交叉验证集找到误差最小的对应 d
- Estimate the generalization error using the test set with $J_{test}(\theta)$ (d = θ from polynomial with lower error).

使用带 $J_{test}(\theta)$ 的测试集估计泛化误差.

=. Bias & Variance.

1. Diagnosing Bias & Variance

当学习算法不理想时，{ 偏差较大.（欠拟合） / 方差较大.（过拟合） / ~~欠拟合. 过拟合.~~



High Bias (underfitting): $J_{train}(\theta)$, $J_{cv}(\theta)$ will be high.

$J_{train}(\theta) \approx J_{cv}(\theta)$.

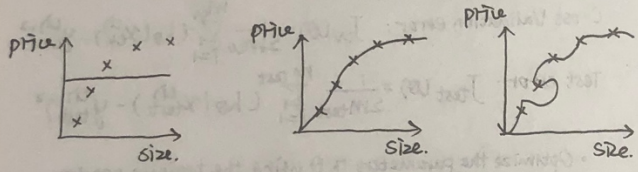
High variance (overfitting): $J_{train}(\theta)$ will be low.

$J_{cv}(\theta)$ will be much greater than $J_{train}(\theta)$

2. Regularization and Bias/Variance.

① Linear Regression: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_4 x^4$

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



Large λ.

High Bias (underfit)

$\lambda = 10000$. $\theta_1 \approx 0$.

$\theta_2 \approx 0$

$h_\theta(x) \approx \theta_0$

Intermediate λ.

Just Right.

Small λ.

High Variance (overfit)

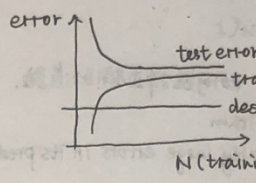
$\lambda = 0$.

② Choose the model and the regularization term λ.

- Create a list of lambdas (λ).
- Create a set of models with different degrees or any other variants.
- Iterate through λ(s) and for each λ go through all models to learn some θ.
- Compute the cross validation error using the learned θ (with λ) on the $J_{cv}(\theta)$ without regularization or $\lambda = 0$.
- Select best combo that produces the lowest error on CV test.
- Using the best combo θ and λ, apply it on $J_{test}(\theta)$ to see if it has a good generalization of the problem.

3. Learning Curves.

① Experiencing High Bias:

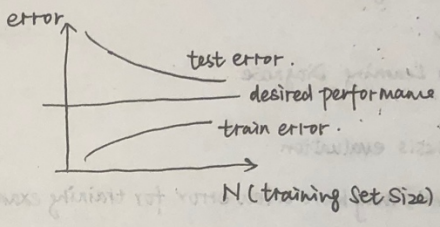


Low training set size: $J_{train}(\theta)$ to be low.

$J_{cv}(\theta)$ to be high.

Large training set size: $J_{train}(\theta) \approx J_{cv}(\theta)$.

② Experiencing High Variance:



Low training set size: $J_{train}(\theta)$ will be low.

$J_{cv}(\theta)$ will be high.

Large training set size: $J_{train}(\theta) < J_{cv}(\theta)$.

4. Diagnose.

① Getting more traing examples: Fix high variance.

② Trying smaller sets of features: Fix high variance.

③ Adding features: Fix high Bias.

④ Adding polynomial features: Fix high Bias.

⑤ Decrease λ: Fix high Bias.

⑥ Increase λ: Fix high Variance.

5. Diagnose Neural Network.

① Fewer parameters → underfitting (computationally cheaper)

② More parameters → overfitting (computationally expensive)

(Hidden Layers ~ Cross Validation Set)

6. Model Complexity Effects

① Low-order polynomials (low model complexity)
have high bias and low variance. The model fits poorly consistently.

② Higher-order polynomials (high model complexity)
fit training data extremely well and the test data extremely poorly. These have low bias on the training data, but very high variance.

③ In reality, choose a model somewhere in between.

Name: Xiaofan Wang

Chapter 7 Support Vector Machines

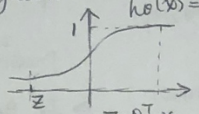
1. Large Margin Classification.

1. Optimization Objective.

① SVM: 在学习非线性的方程时，提供一更清晰和强大的方式

② Alternative view of logistic regression:

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



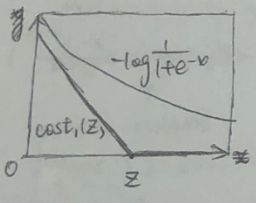
If $y=1$, we want $h_\theta(x)\approx 1$. $\theta^T x \geq 0$

If $y=0$, we want $h_\theta(x)\approx 0$. $\theta^T x \leq 0$

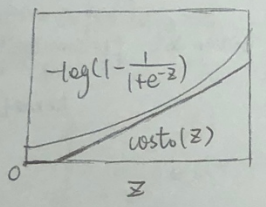
cost of example: $-y\log h_\theta(x) + (1-y)\log(1-h_\theta(x))$

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

If $y=1$, $(\theta^T x \geq 0)$



If $y=0$ $(\theta^T x \leq 0)$



③ Support Vector Machine

logistic regression: $\min_\theta \frac{1}{m}\left[\sum_{i=1}^m y^{(i)}(-\log h_\theta(x^{(i)})) + (1-y^{(i)})(-\log(1-h_\theta(x^{(i)})))\right] + \frac{\lambda}{2m}\sum_{j=1}^m \theta_j^2$

(A = the bracketed term; B = the regularization term)

$\text{cost}_1(\theta^T x^{(i)})$ $\qquad$ $\text{cost}_0(\theta^T x^{(i)})$

support vector machine: $\min_\theta \left(\frac{1}{m}\right)\sum_{i=1}^m y^{(i)}\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T x^{(i)}) + \frac{\left(\frac{\lambda}{2m}\right)}{\times}\sum_{j=1}^n \theta_j^2$

常量，可以去掉

例 $\min_u (u-5)^2+1 \Rightarrow u=5$

$\min_u 10(u-5)^2+10 \Rightarrow u=5$

logistic regression: $A+\lambda B$

SVM: $CA+B$ (C=大时，两式意义相同，0同)

(用参数决定，是侧重A优化，or B优化)

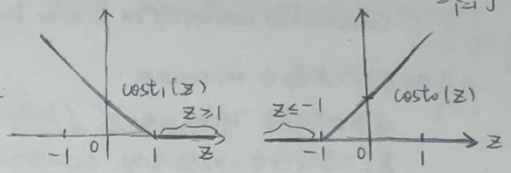
Hypothesis:

h_θ: 1 if $\theta^T x \geq 0$; 0 otherwise.

$\Rightarrow$ SVM: $\min_\theta C\sum_{i=1}^m [y^{(i)}\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T x^{(i)})] + \frac{1}{2}\sum_{i=1}^n \theta_j^2$

2. Large Margin Intuition

SVM: $\min_\theta C\sum_{i=1}^m [y^{(i)}\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T x^{(i)})] + \frac{1}{2}\sum_{i=1}^n \theta_j^2$



If $y=1$, we want $\theta^T x \geq 1$ (not just ≥ 0).

If $y=0$, we want $\theta^T x \leq -1$ (not just <0)

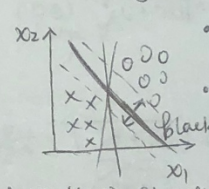
① SVM Decision Boundary:

Whenever $y^{(i)}=1$: $\theta^T x^{(i)} \geq 1$

Whenever $y^{(i)}=0$: $\theta^T x^{(i)} \leq -1$

$\min_\theta C\times 0 + \frac{1}{2}\sum_{j=1}^n \theta_j^2$

② Linearly Separable case



Large Margin Classifier

- 这些决策边界看起来不常用选择
- black: 更稳健，在分离正负样本更好
- 决策界和训练样本之间有更大的最短距离 (margin) ⇒ 具有鲁棒性的原因

↓

用一个最大间距来分离样本

(大间距分类器)

③ Large Margin Classifier in presence of outliers.

⇒ C very large. :

Outlier would have impact on decision boundary

⇒ 正则化系数 C not too large :

决策边界将会 "ignore outliers". (better decision boundary)

甚至不是线性可分的状况时, SVM也可将它们恰当分开.

3. The mathematics behind large margin classification. (Optional)

① vector inner Product.

$\vec{u}=\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ $\vec{v}=\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$.

$u^T \cdot v = [u_1\ u_2]\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = u_1v_1 + u_2v_2$.

$(u^T \cdot v = p \cdot \|u\|,\ p \in \mathbb{R})$

p = length of projection of v onto u.

$\|u\|$ = length of vector $u = \sqrt{u_1^2+u_2^2}$. $(\in \mathbb{R})$.

② SVM Decision Boundary.

$\min_\theta \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}(\theta_1^2+\theta_2^2) = \frac{1}{2}\left(\sqrt{\theta_1^2+\theta_2^2}\right)^2 = \frac{1}{2}\|\theta\|^2$ ($\|\theta\|$)

$\theta^T \cdot x^{(i)} = p^{(i)}\|\theta\| = \theta_1 \cdot x_1^{(i)} + \theta_2 \cdot x_2^{(i)}$

$(u^T \cdot v)$

s.t. $\begin{cases} p^{(i)} \cdot \|\theta\| \geq 1 & \text{if } y^{(i)}=1 \\ p^{(i)} \cdot \|\theta\| \leq -1 & \text{if } y^{(i)}=1 \end{cases}$

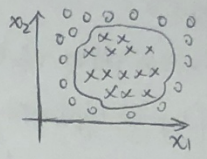
where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplication: $\theta_0=0$

= kernels. (核函数)

1. kernels I.

① Non-linear Decison Boundary

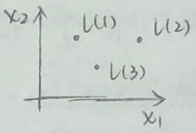


Predict $y=1$.

⇒ If $\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_1x_2+\theta_4x_1^2+\theta_5x_2^2\cdots \geq 0$

$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0+\theta_1x_1\cdots \geq 0) \\ 0 & \text{otherwise.} \end{cases}$

② Kernel.



Given x, compute new feature depending on proximity to landmarks $l^{(1)}$, $l^{(2)}$, $l^{(3)}$.

Given x: $f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2}\right)$

↑ kernel ↑ Gaussian kernels

$k(x, l^{(1)})$

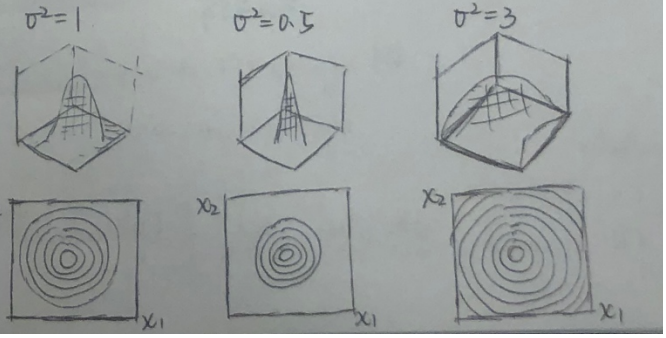
$= \exp\left(-\frac{\sum_{j=1}^{n}(x_j-y_j^{(1)})^2}{2\sigma^2}\right)$

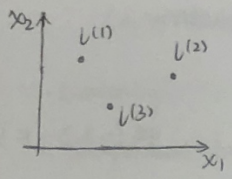
$\begin{cases} \text{If } x \approx l^{(1)}: & f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1 \\ \text{If } x \text{ far from } l^{(1)}: & f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0 \end{cases}$

Example:

$l^{(1)}=\begin{bmatrix} 3 \\ 5 \end{bmatrix}$, $f_1 = \exp\left(-\frac{\|x-l^{(1)}\|^2}{2\sigma^2}\right)$

σ^2 大时, 特征变量的值减小速度会变慢.





Predict "1" when

$\theta_0+\theta_1 f_1+\theta_2 f_2+\theta_3 f_3\geq 0$

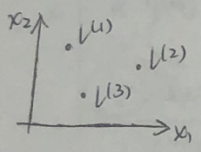
(x 与特征值)

$\theta_0=-0.5,\ \theta_1=1,\ \theta_2=1,\ \theta_3=0$

$\begin{cases} f_1\approx 1,\ f_2\approx 0,\ f_3\approx 0 & -0.5+1=0.5\geq 0 \\ f_1,\ f_2,\ f_3\approx 0 & -0.5<0 \end{cases}$

2. Kernels II

① Choose the landmarks.



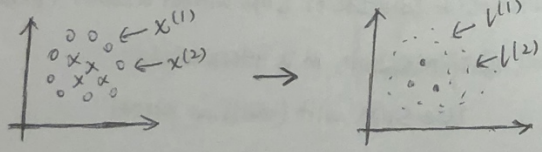
Given x:

$f_i=\text{similarity}(x, l^{(i)})$

$=\exp\left(-\frac{\|x-l^{(i)}\|^2}{2\sigma^2}\right)$

Predict $y=1$ if $\theta_0+\theta_1 f_1+\theta_2 f_2+\theta_3 f_3\geq 0$.

Where to get $l^{(1)}, l^{(2)}, l^{(3)}$ ---?



将训练样本 $x^{(1)}$--- 作为标记点 $l^{(1)}, l^{(2)}$--- $l^{(m)}$.

② SVM with Kernels

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)})$ --- $(x^{(m)}, y^{(m)})$

Choose $l^{(1)}=x^{(1)},\ l^{(2)}=x^{(2)}$ --- $l^{(m)}=x^{(m)}$

Given example x:

$f_1=\text{similarity}(x, l^{(1)})$

$f_2=\text{similarity}(x, l^{(2)})$

$f=\begin{bmatrix} f_0 \\ f_1 \\ \vdots \end{bmatrix}$ $f_0=1$

For training example $(x^{(i)}, y^{(i)})$:

$x^{(i)} \rightarrow \begin{cases} f_1^{(i)}=\text{similarity}(x^{(i)}, l^{(1)}) \\ f_2^{(i)}=\text{similarity}(x^{(i)}, l^{(2)}) \\ \cdots \end{cases}$

$f^{(i)}=\begin{bmatrix} f_0^{(i)} \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$ $x^{(i)}\in\mathbb{R}^{n+1}$

描述样本的特征向量.

Hypothesis: Given x, compute features $f\in\mathbb{R}^{m+1}$

Predict "$y=1$" if $\theta^T f\geq 0$.

Training: $\min_\theta C\sum_{i=1}^{m} y^{(i)}\,\text{cost}_1(\theta^T f^{(i)})+(1-y^{(i)})\cdot \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2}\cdot\sum_{j=1}^{n}\theta_j^2$.

③ SVM parameters:

$C(=\frac{1}{\lambda})\Rightarrow$ Large C: Lower bias, High variance. (λ 小)

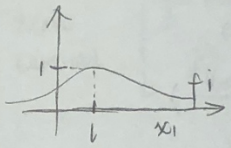
Small C: Higher bias, low variance (λ 大)

$\sigma^2\Rightarrow$ Large σ^2: Features f_i vary more smoothly.

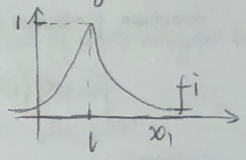
Higher bias, lower variance.

Small σ^2: Features f_i vary less smoothly

Lower bias, higher variance.



Large σ^2



Small σ^2.

Ⅲ. Using an SVM

1. Use SVM software package (e.g. liblinear, libsvm --) to solve for parameters θ.

- Need to specify: { Choice of parameter C. / Choice of kernel (similarity function).

  Eg. No kernel ("linear kernel")
  Predict "y=1" if $\theta^T x \geq 0$

- Gaussian kernel: $f_i = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$, where $l^{(i)} = x^{(i)}$

  Need to choose σ^2.

2. kernel (similarity) functions:

function f = kernel (x_1, x_2)

$$f = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

return.

(Note: Do perform feature scaling before using the Gaussian kernel).

$\|x - l\|^2 \Rightarrow v = x - l$

$$\|v\|^2 = v_1^2 + v_2^2 + \cdots + v_n^2$$
$$= (x_1 - l_1)^2 + (x_2 - l_2)^2 + \cdots (x_n - l_n)^2$$

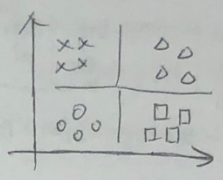
3. Other choices of kernel.

Note: Not all similarity functions "similarity (x, l)" make valid kernels.
(Need to satisfy technical condition called "Mercer's Theorem" to make sure SVM packages' optimizations run correctly, and do not diverge).

① Many off-the-shelf kernels available:

- Polynomial kernel: $k(x, l) = (x^T \cdot l)^2$ $(x^T \cdot l)^3$. $(x^T l + 1)^3$. $(x^T l + 5)^4 \Rightarrow (x^T l + \text{constant})^{\text{degree}}$
- More esoteric: String kernel. Chi-square kernel. Histogram intersection kernel.

4. Multi-class Classification.



$y \in \{1, 2, 3 \cdots K\}$

① Many SVM packages already have built-in multi-class classification functionality.

② Otherwise, use one-vs-all method.
(Train K SVMs, one to distinguish $y=i$ from the rest, for $i = 1, 2 \cdots K$.) get $\theta^{(1)} \cdots \theta^{(K)}$.
Pick class i with largest $(\theta^{(i)})^T \cdot x$

5. Logistic Regression VS. SVMs.

n = number of features ($x \in \mathbb{R}^{n+1}$)
m = number of training examples.

① If n is large, (relative to m):
Use LR or SVM without a kernel. (linear).

② If n is small, m is intermediate:
Use SVM with Gaussian kernel.

③ If n is small, m is large:
Create/Add more features, then use logistic regression or SVM without a kernel.

[Neural Network likely to work well for most of these ~~things~~ settings, but maybe slower to train].

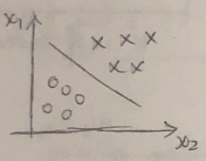
Name: Xiaofan Wang

Ch 8 Unsupervised Learning

一. Clustering

1. Unsupervised learning. (从未标记的数据中学习)

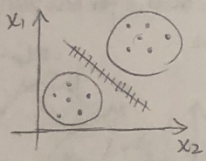
Supervised Learning (with labeled Data)



Training Set:

$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \cdots (x^{(m)}, y^{(m)})\}$

Unsupervised Learning. (With unlabeled Data)



Training Set:

$\{x^{(1)}, x^{(2)} \cdots x^{(m)}\}$

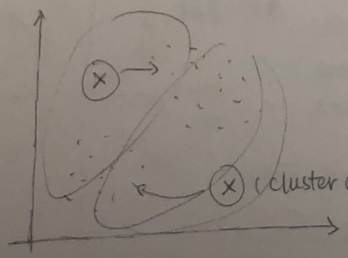
$\Rightarrow$ Clustering Algorithm

Applications of Clustering:
- Market Segmentation.
- Social Network Analysis.
- Organize Computing Clusters.
- Astronomical Data Analysis.

2. k-means Algorithm (most popular Clustering Algorithm)

⇓

帮助找到紧密的子集成簇.



k-means Algorithm:

① Input: { k (number of clusters); Training Set $\{x^{(1)}, x^{(2)} \cdots x^{(m)}\}$ }

$x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention).

② Randomly initialize K cluster centroids

$\mu_1, \mu_2, \mu_3 \cdots \mu_K \in \mathbb{R}^n$.

~~Rand~~ Repeat {

Cluster Assignment Step: for i=1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

Move Centroid: for k=1 to K

μ_k := average of points assigned to cluster k. }

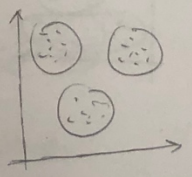
$x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$

↓

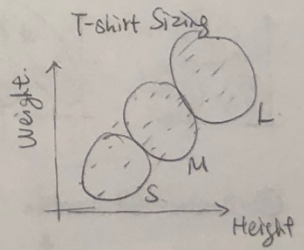
$c^{(1)}=2, c^{(5)}=2, c^{(6)}=2, c^{(10)}=2$

$\mu_2 = \frac{1}{4}[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}] \in \mathbb{R}^n$.

③ K-means for non-separated clusters



Separated Clusters (K=3)



Non-Separated Clusters

例: 市场细分

↓

不同的顾客群体

不同的需求.

(1) 簇分配: 将所有小点分配到两个不同的聚类中心中.

(2) 移动聚类中心: 迭代下去. 聚类中心不再变. 聚集点也不变. $\Rightarrow$ (k-means 收敛)

3. Clustering: Optimization Objective.

$c^{(i)}$ = index of cluster (1, 2, ... K) to which example $x^{(i)}$ is currently assigned.

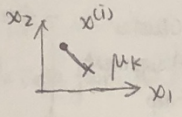
μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned.

Optimization Objective:

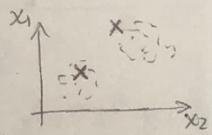
$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \cdot \sum_{i=1}^{m} \| x^{(i)} - \mu_{c^{(i)}} \|^2$$

$$\min_{c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K}$$



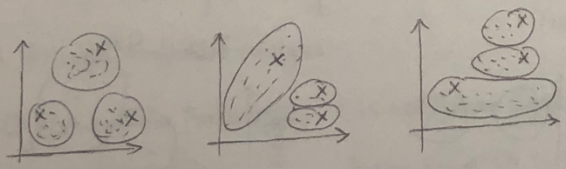
4. Clustering: ~~Bound~~ Random Initialization.

① Random Initialization.



- Should have $K < m$
- Randomly pick K training examples.
- Set $\mu_1, \dots, \mu_K$ equal to these K examples

② Local Optima.



③ Random Initialization:

For i = 1 to 100 {

Randomly initialize K-means.

Run K-means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$.

Compute cost function $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

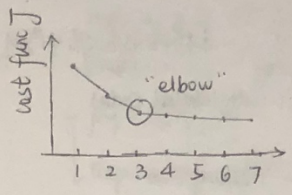
}

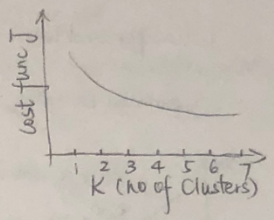
Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$

(K=2-10. 尝试多次不同初始化. 较大的时 → min J.)

5. Choosing the Number of Clusters

① Elbow Method. (肘部法则)

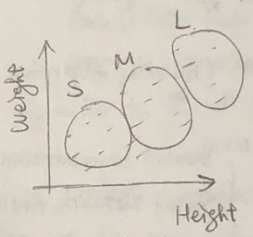


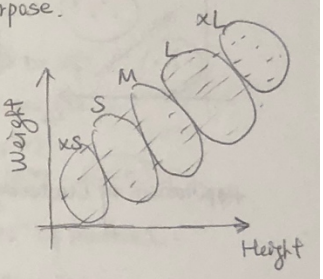


K=3. 之后下降非常慢.

② For later / downstream purpose.

Evaluate K-means based on a metric for how well it performs for that later purpose.



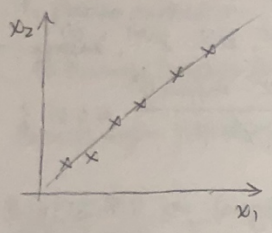


T-shirt Sizing. K=3 or K=5. ⇒ later purpose

二. Dimensionality Reduction (减少内存，算法提速)

1. Data Compression.

Reduce memory / disk needed to store data.
Speed up learning algorithm.



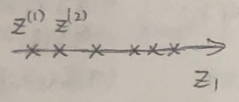
Reduce data from 2D to 1D

$x^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$

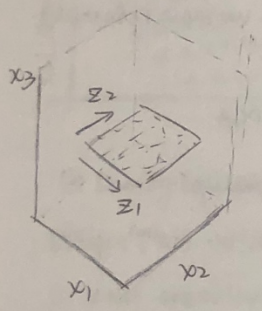
$x^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$.

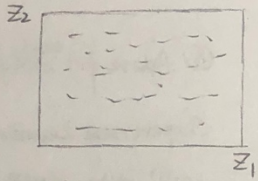
$\vdots$

$x^{(m)} \dashrightarrow z^{(m)}$



Reduce Data from 3D to 2D.





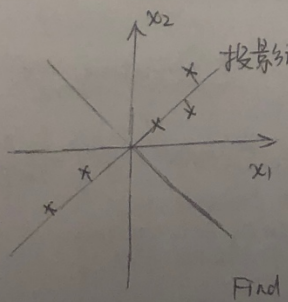
$z^{(i)} \in \mathbb{R}^2$

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ $z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ z_2^{(i)} \end{bmatrix}$

2. Data Visualization. (将高维数据可视化为二维、三维~)

3. Principal Component Analysis.

① PCA problem Formulation. (is not linear regression $x \rightarrow y$. $x_1 \cdots x_m$)



Reduce from 2D to 1D:

Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project ~~the~~ the data so as to minimize the projection error.

Reduce from nD to KD:

Find K vectors $u^{(1)}, u^{(2)} \cdots u^{(k)}$ onto which to project the data, so as to minimize the projection error. (寻找一组向量，K个方向，K维的平面，K个方向来定义平面中点的位置.)

② PCA Algorithm.

Data preprocessing:

Training Set: $x^{(1)}, x^{(2)} \cdots x^{(m)}$

Preprocessing: (feature scaling / mean normalization)

$\mu_j = \frac{1}{m} \cdot \sum_{i=1}^{m} x_j^{(i)}$

Replace each $x_j^{(i)}$ with $x_j - \mu_j$

If different features on different scales
(x_1 = size of house, x_2 = number of bedrooms),
scale features to have comparable range of values.

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

PCA Algorithm:

Reduce data from nD to KD

Compute "covariance matrix":

$\Rightarrow Sigma(\Sigma) = \frac{1}{m} \cdot \sum_{i=1}^{m} (x^{(i)})(x^{(i)})^T$ $(n \times 1)$ $(1 \times n)$ → $n \times n$

Compute "eigenvectors" of matrix Σ:

$\Rightarrow$ [U, S, V] = svd(Sigma); → $n \times n$ matrix.

$\Rightarrow$ Ureduce = U(:, 1:k);

$\Rightarrow z = Ureduce' * x;$

svd: singular value decomposition
eig(Sigma). 奇异值分解.

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

K.

$x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^k$

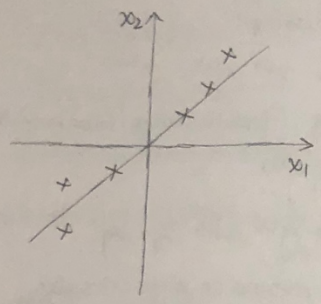
$$z = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}^T \cdot x$$

$(z \in \mathbb{R}^k)$ $(n \times 1)$

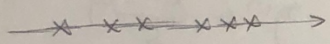
$n \times K$.

Ureduce

③ Reconstruction from compressed representation.



$z = U_{reduce}^T x$



$z \in \mathbb{R} \rightarrow x \in \mathbb{R}^2$

$x_{approx}^{(i)} = U_{reduce} \cdot z^{(i)}$ (z映射回原始高维数据的近似值)

$(\in \mathbb{R}^n)$ $n \times k$ $k \times 1$

$n \times 1$

④ Choosing the number of Principal Components (k).
(主成分数量)

Average squared projection error: $\frac{1}{m} \cdot \sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|^2$

Total variation in the data: $\frac{1}{m} \cdot \sum_{i=1}^{m} \| x^{(i)} \|^2$

Typically, choose k to be smallest value so that:

$$\frac{\frac{1}{m} \cdot \sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|^2}{\frac{1}{m} \cdot \sum_{i=1}^{m} \| x^{(i)} \|^2} \leq 0.01 \quad (1\%)$$

0.05 5%

0.10 10%

99% of variance is retained.

95%

90%

许多特征高度相关

Choosing k (number of Principal components):

Algorithm:

Try PCA with k=1. 2 ...

Compute U_{reduce}, $z^{(1)}$, $z^{(2)}$ .. $z^{(m)}$, $x_{approx}^{(1)} \cdots x_{approx}^{(m)}$

Check if: $\frac{\frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|^2}{\frac{1}{m} \cdot \sum_{i=1}^{m} \| x^{(i)} \|^2} \leq 0.01$?

$[U, S, V] = svd(Sigma)$ ⇒ 只需运行此步

Pick smallest value of k: 可得S对角线上的值 改变k值 (不需从头运行)

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}} \geq 0.99$$

(99% of variance retained)

⑤ Adive for applying PCA.

Supervised Learning Speedup:

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots (x^{(m)}, y^{(m)})$

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)} \cdots x^{(m)} \in \mathbb{R}^{10000}$

$x \rightarrow z$: $h_\theta(z) = \frac{1}{1+e^{-\theta^T x}}$ ↓PCA $\rightarrow z^{(1)}, z^{(2)} \cdots z^{(m)} \in \mathbb{R}^{1000}$

New training set: $(z^{(1)}, y^{(1)}) (z^{(2)}, y^{(2)}) \cdots$

Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

⑥ Bad use of PCA: To prevent overfitting. (x)
(更容易舍弃有价值的信息).

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

⑦ PCA shouldn't be: design of ML system.

⑧ Application of PCA.

Data Compression: reduce memory & speedup algorithm

Data Visualization: 2D. 3D.

Name: Xiaofan Wang

Chapter 9 Problem Motivation (Anomaly Detection)

一. Density Estimation.

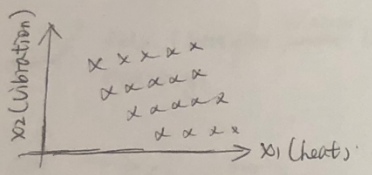
1. Problem Motivation.

① Anomaly Detection Example.

Aircraft engine features: x_1 = ~~heated~~ generated
x_2 = vibration intensity

Dataset: $\{x^{(1)}, x^{(2)}, \dots x^{(m)}\}$

New Engine: x_{test}



② Density Estimation

Dataset $\{x^{(1)}, x^{(2)}, \dots x^{(m)}\}$

Is x_{test} anomalous? Model $p(x)$ $\begin{cases} p(x_{test}) < \varepsilon \text{ (Anomaly)} \\ p(x_{test}) \geq \varepsilon \text{ (OK)} \end{cases}$

③ Example.

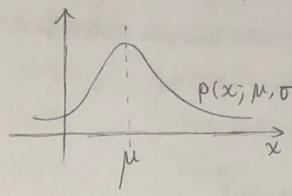
Fraud Detection: $\begin{cases} x^{(i)} = \text{features of users' } i \text{ activities} \\ \text{Model } p(x) \text{ from data} \\ \text{Identify unusual users by checking } p(x) < \varepsilon \end{cases}$

Manufacturing.

Monitoring computers in a data center: $\begin{cases} x^{(i)} = \text{features of machine } i \\ x_1 = \text{memory use,} \\ x_2 = \text{number of disk } \sim \\ x_3 = \text{CPU load} \\ x_4 = \text{CPU load / network traffic} \end{cases}$

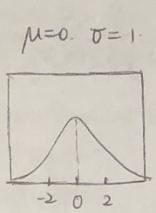
二. Gaussian Distribution

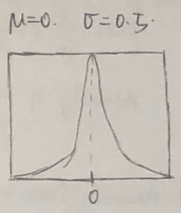
1. Say $x \in R$. If x is a distributed Gaussian with mean μ, variance σ^2. $x \sim N(\mu, \sigma^2)$

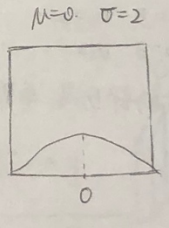


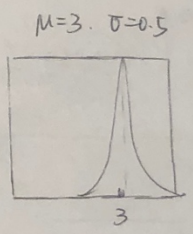
$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\cdot\sigma} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

2. Gaussian Distribution Example.







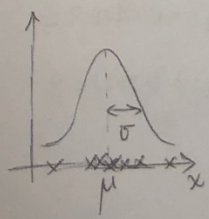


3. Parameter Estimation.

Dataset $\{x^{(1)}, x^{(2)}, \dots x^{(m)}\}$ $x^{(i)} \in R$.

假设每个样本服正态/高斯分布. $x^{(i)} \sim N(\mu, \sigma^2)$

参数估计：给定数据集. 希望可以找到能估算 μ 和 σ 的值.



其中 μ 对应分布函数的中心.

标准差 σ 控制高斯分布的宽度.

↓

很好地拟合了数据

分布在中心区域的概率较大.

$$\mu = \frac{1}{m}\cdot\sum_{i=1}^{m} x^{(i)}, \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)^2$$

三. Algorithm. (应用高斯分布开发异常检测算法)

1. Anomaly Detection Algorithm

① Choose features x_i that you think might be indicative of anomalous examples. (描述数据相关属性特征)

② Fit parameters $\mu_1 \cdots \mu_n$. $\sigma_1^2 \cdots \sigma_n^2$.

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

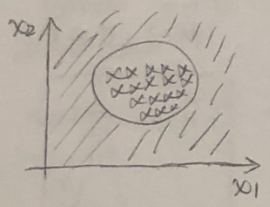
$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

③ Given new example x, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}\right)$$

Anomaly if $p(x) < \varepsilon$.

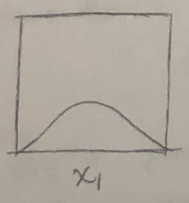
2. Anomaly Detection Example.



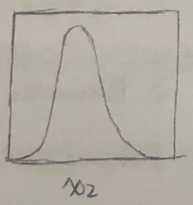
$\mu_1 = 5$. $\sigma_1 = 2$.

$\mu_2 = 3$. $\sigma_2 = 1$.

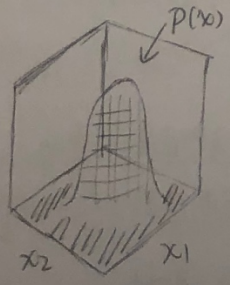
阴影部分是"异常样本"



$p(x_1; \mu_1, \sigma_1^2)$



$p(x_2; \mu_2, \sigma_2^2)$



$\varepsilon = 0.02$

$p(x_{test}^{(1)}) = 0.0426 \geq \varepsilon$

$p(x_{test}^{(2)}) = 0.0021 < \varepsilon$