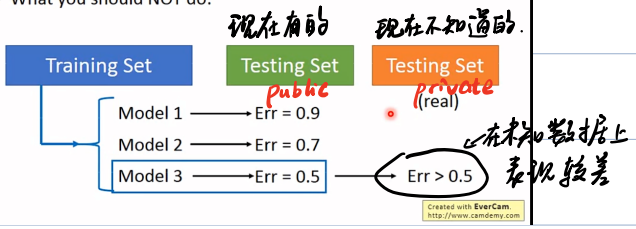
Model Selection

Model Selection

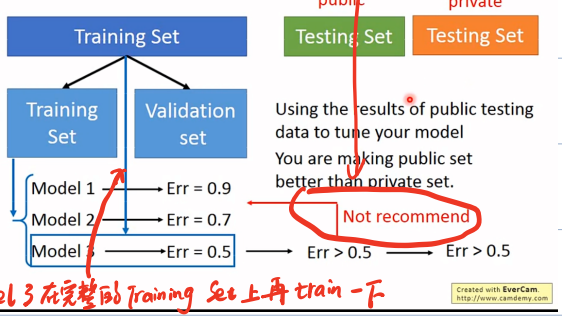
- There is usually a trade-off between bias and variance.
- Select a model that balances two kinds of error to minimize total error
- What you should NOT do:



对于在真实数据(private Testing Set)上反应结果较差的方法：在Training Set中分出Validation set，相当于把从现有数据(public Testing Set)中得到最好函数这一步提前到Validation set来做. 然后把该函数在public Set上测试，从而在public set上得到的Err可以和在private Set上得到的Err更接近更真实(相当于把所有步骤提前一步，使public当private)

Cross Validation

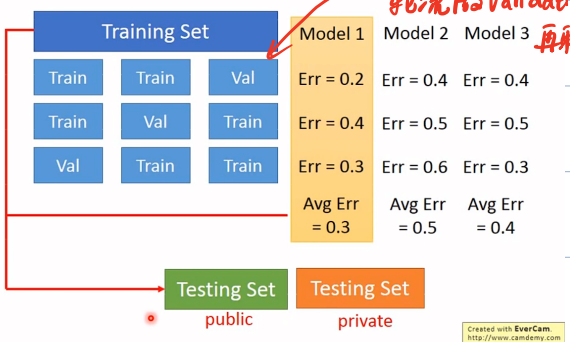
不打乱抽样的. 要刚提前这一步就失去了意义.



担心选出来的Validation set中混有不好的数据 how解决?

N-fold Cross Validation

将Training set分成3组，轮流做Validation set. 再取平均



Gradient Descent

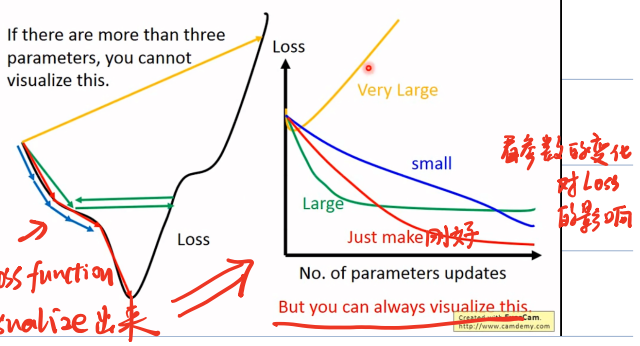
Review...

注意点：Tip1：小心地调learning rate.

Learning Rate

$$\theta^i = \theta^{i-1} - \eta \nabla C(\theta^{i-1})$$

Set the learning rate η carefully



调 Learning rates 麻烦

⇒ Adaptive Learning Rates. (自动调)

Adaptive Learning Rates

- Popular & Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate
  - E.g. 1/t decay: $\eta^t = \eta/\sqrt{t+1}$ (t为迭代次数. 随t减小)
- Learning rate cannot be one-size-fits-all
  - Giving different parameters different learning rates 给不同参数不同η.

Adagrad

$$\eta^t = \frac{\eta}{\sqrt{t+1}} \qquad g^t = \frac{\partial C(\theta^t)}{\partial w}$$

- Divide the learning rate of each parameter by the ***root mean square of its previous derivatives***

Vanilla Gradient descent

$$w^{t+1} \leftarrow w^t - \eta^t g^t$$

w is one parameters

Adagrad 对其中某一参数

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t$$

σ^t: ***root mean square*** of the previous derivatives of parameter w 方均根

与w有关 → Parameter dependent

Adagrad

σ^t: ***root mean square*** of the previous derivatives of parameter w

$$w^1 \leftarrow w^0 - \frac{\eta^0}{\sigma^0} g^0 \qquad \sigma^0 = \sqrt{(g^0)^2}$$

$$w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\frac{1}{2}[(g^0)^2 + (g^1)^2]}$$

$$w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\frac{1}{3}[(g^0)^2 + (g^1)^2 + (g^2)^2]}$$

$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2}$$

Adagrad

- Divide the learning rate of each parameter by the ***root mean square of its previous derivatives***

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t$$

$$\eta^t = \frac{\eta}{\sqrt{t+1}}$$ 1/t decay

$$\sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2}$$

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

方根，不用求平均

Why $\begin{cases} w^{t+1} \leftarrow w^t - \gamma g^t, & g^t \text{越大步伐越大} \\ w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t, & g^t \text{越大，步伐越小？} \end{cases}$

解释1：

Intuitive Reason

$$\eta^t = \frac{\eta}{\sqrt{t+1}} \quad g^t = \frac{\partial C(\theta^t)}{\partial w}$$

- How surprise it is 反差

特别大

g^0	g^1	g^2	g^3	g^4	.....
0.001	0.001	0.003	0.002	0.1	.....
g^0	g^1	g^2	g^3	g^4	.....
10.8	20.9	31.7	12.1	0.1	.....

特别小

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

造成反差的效果

看和过去相比反差多大，以此判定其相对大小

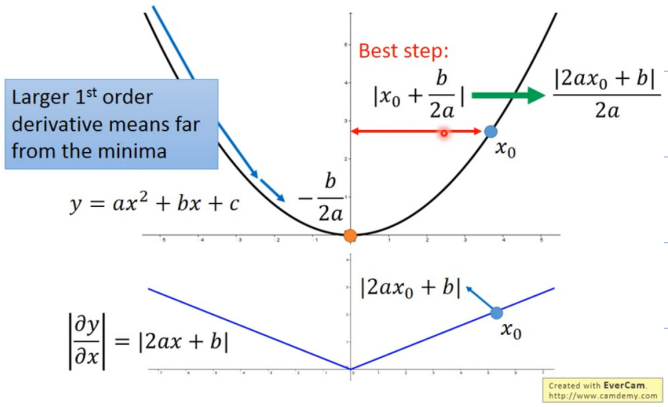
解释2：

普对单一的参数而言，用 $w^{t+1} \leftarrow w^t - \gamma g^t$ 确实合理

因为一般 gradient 越大离最低点越远，是合理的

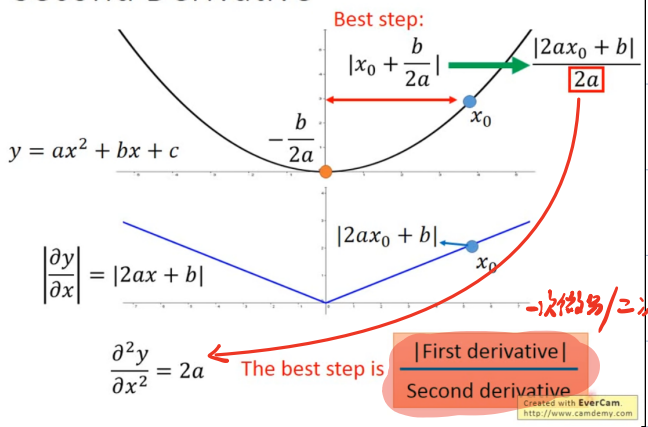
但对比2个参数，

Larger gradient, larger steps?



不一定 gradient 越大的离最低点越远，跨参数比较，如a和c点，c的gradient比a大，但实际a离最低点更远，故在多参数时gradient是一个相对值，找最好的 step

Second Derivative



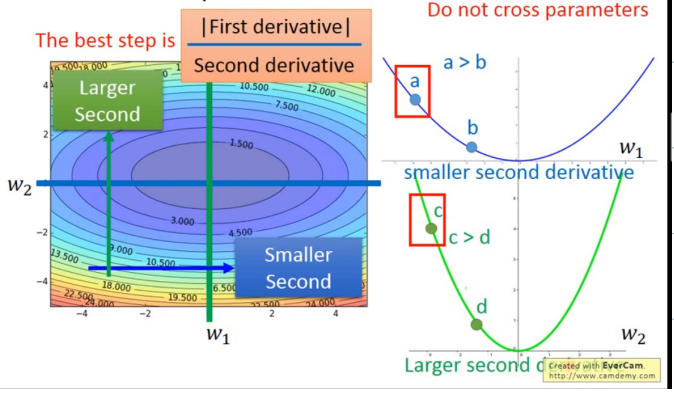
一次微分/二次微分

Comparison between different parameters

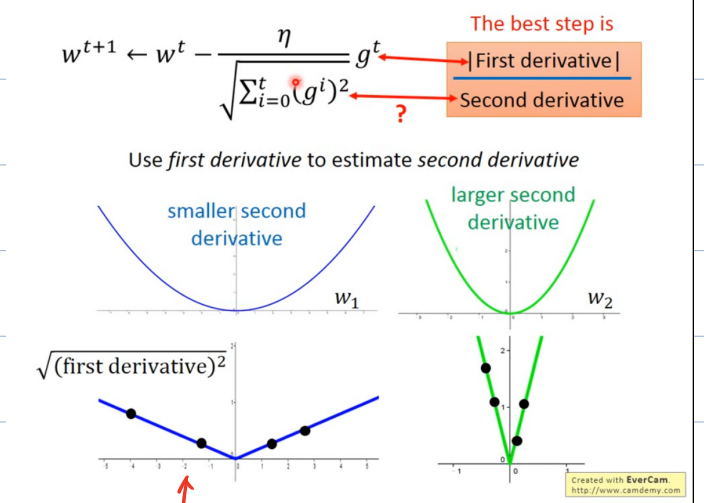
Larger 1st order derivative means far from the minima

Do not cross parameters

The best step is |First derivative| / Second derivative



这种 一次微分/二次微分 与 Adagrad 关系？



通过取平方来反映二次微分的大小

Tip 2: Stochastic (随机) Gradient Descent

让 Training 更快

Stochastic Gradient Descent

$$L = \sum_n \left(\hat{y}^n - \left(b + \sum w_i x_i^n \right) \right)^2$$

Loss is the summation over all training examples

◆ ***Gradient Descent*** $\theta^i = \theta^{i-1} - \eta \nabla L(\theta^{i-1})$

◆ ***Stochastic Gradient Descent***

Pick an example x^n ← 随机选一点算Loss

$$L^n = \left(\hat{y}^n - \left(b + \sum w_i x_i^n \right) \right)^2 \qquad \theta^i = \theta^{i-1} - \eta \nabla L^n(\theta^{i-1})$$

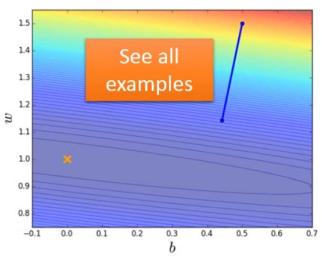
Loss for only one example

使用 Stochastic Gradient Descent 若有 m 个参数，就要将参数 θ update m 次

Stochastic Gradient Descent

Gradient Descent

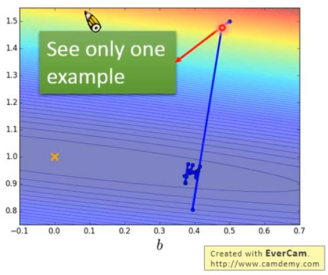
Update after seeing all examples



Stochastic Gradient Descent

Update for each example

If there are 20 examples, 20 times faster.

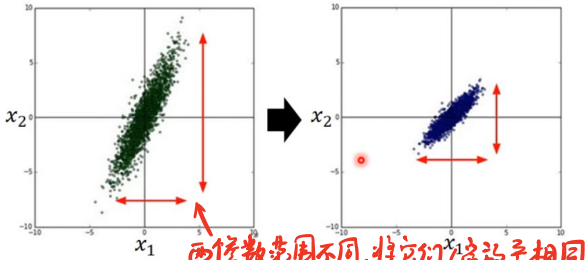


Tip 3: Feature Scaling. 特征放缩

Feature Scaling

Source of figure: http://cs231n.github.io/neural-networks-2/

$$y = b + w_1 x_1 + w_2 x_2$$



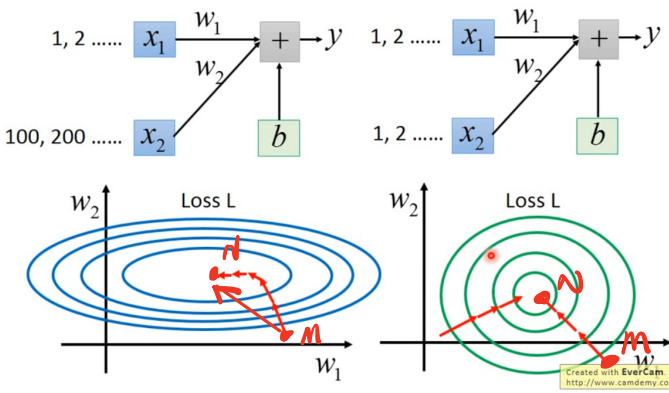
两个数范围不同，特征们缩放至相同

Make different features have the same scaling

Why 这样做？

Feature Scaling

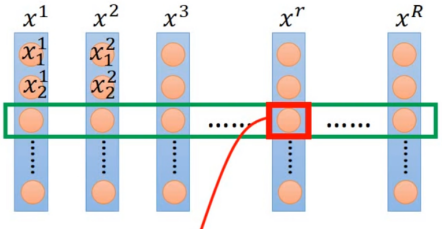
$$y = b + w_1 x_1 + w_2 x_2$$



若用 Adagrad，w_1 和 w_2 对 Loss 的影响很不一样。若 Loss 曲线像左边一样为椭圆，则从 M→N 是沿着等高线走，而非从最高点指向最低点。但 rescaling 成圆形后就会指向 N 点

How 做 Feature Scaling.

Feature Scaling



For each dimension i:

mean: m_i

standard deviation: σ_i ← 标准差

$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

The means of all dimensions are 0, and the variances are all 1

关于 Gradient Descent 的数学原理.

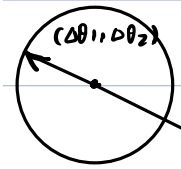
用对多变量函数的泰勒展开，在 (a,b) 点处 $\theta_1 = a$, $\theta_2 = b$.

$$L(\theta) = L(a,b) + \frac{\partial L(a,b)}{\partial \theta_1}(\theta_1 - a) + \frac{\partial L(a,b)}{\partial \theta_2}(\theta_2 - b)$$

$L(a,b)$ ↑ s　　$\frac{\partial L(a,b)}{\partial \theta_1}$ ↑ u　　$\frac{\partial L(a,b)}{\partial \theta_2}$ ↑ v

考虑在 a,b 附近一个小圆范围内 $L(\theta)$ 的最小值.

L(θ)最小值同时可满足 (u,v) 与 $(\theta_1-a, \theta_2-b)=(\Delta\theta_1, \Delta\theta_2)$ 在 $\Delta\theta_1^2+\Delta\theta_2^2 \le d^2$ 范围内的最小值，显然相反且 $\Delta\theta_1^2+\Delta\theta_2^2=d^2$ 时最小，则 $(\Delta\theta_1, \Delta\theta_2)$ 可表示为 $\begin{bmatrix}\Delta\theta_1\\ \Delta\theta_2\end{bmatrix}=-\eta\begin{bmatrix}u\\ v\end{bmatrix}$



即 $\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}=\begin{bmatrix}a\\ b\end{bmatrix}-\eta\begin{bmatrix}u\\ v\end{bmatrix}$

gradient descent ⟶ $=\begin{bmatrix}a\\ b\end{bmatrix}-\eta\begin{bmatrix}\frac{\partial L(a,b)}{\partial\theta_1}\\ \frac{\partial L(a,b)}{\partial\theta_2}\end{bmatrix}$

但让该值精确的前提是泰勒展开是精确的，就要求画的圈足够小，d够小 ∴ 理论上 Learning rate 要足够小才行。

泰勒也可展开到更高阶，此时 Learning rate 可以大一点，但要求更多的会更麻烦。

Classification

Example Application

POKÉMON TYPE SYMBOLS

NORMAL	FIRE	WATER	ELECTRIC	GRASS	ICE
FIGHTING	POISON	GROUND	FLYING	PSYCHIC	BUG
ROCK	GHOST	DRAGON	DARK	STEEL	FAIRY

$f($  $)=$  　$f($  $)=$  　$f($  $)=$ 

Created with EverCam. http://www.camdemy.com

⇒要把宝可梦作输入，将其数值化

将其各个特征数值化后组成向量，一向量代表一只

Example Application

pokemon games (*NOT* pokemon cards or Pokemon Go)

- **Total**: sum of all stats that come after this, a general guide to how strong a pokemon is 320
- **HP**: hit points, or health, defines how much damage a pokemon can withstand before fainting 35
- **Attack**: the base modifier for normal attacks (eg. Scratch, Punch) 55
- **Defense**: the base damage resistance against normal attacks 40
- **SP Atk**: special attack, the base modifier for special attacks (e.g. fire blast, bubble beam) 50
- **SP Def**: the base damage resistance against special attacks 50
- **Speed**: determines which pokemon attacks first each round 90

Can we predict the "type" of pokemon based on the information?

Created with EverCam. http://www.camdemy.com

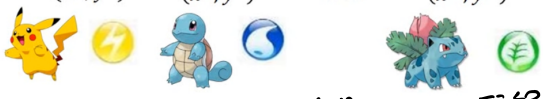
⇒七个分量的向量

如何分类？

How to do Classification

- Training data for Classification

$(x^1, \hat{y}^1)$　$(x^2, \hat{y}^2)$　……　$(x^N, \hat{y}^N)$



Classification as Regression? 当成Regression硬解

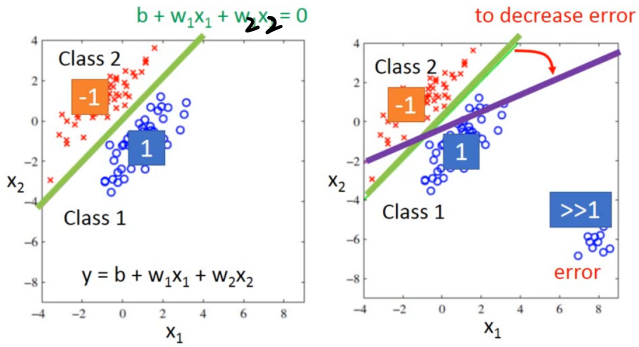
Binary classification as example

Training: Class 1 means the target is 1; Class 2 means the target is -1

Testing: closer to 1 → class 1; closer to -1 → class 2

Created with EverCam. http://www.camdemy.com

当Regression解遇到的问题



Penalize to the examples that are "too correct" … (Bishop, P186)

- Multiple class: Class 1 means the target is 1; Class 2 means the target is 2; Class 3 means the target is 3 …… problematic

Created with EverCam. http://www.camdemy.com

问题1：对Classification而言，绿线作为分界是最好的。但对Regression来说，由于有些"过正确"的值(>>1)，为使loss更小，会选择紫色线，即两者对于model好坏的定义不相同。

问题2：若有多个类别，Regression会认为相邻的两个class之间有某种关系，但可能实际上这两class间并无关系。

Ideal Alternatives

- Function (Model): $f(x)$

$x \rightarrow$ $g(x) > 0$ Output = class 1; $else$ Output = class 2

- Loss function:

$$L(f) = \sum_n \delta(f(x^n) \neq \hat{y}^n)$$

The number of times f get incorrect results on training data.

- Find the best function:
  - Example: Perceptron, SVM — Not Today

以上这种方法，在f(x)中找一个g(x)，用g(x)>0来判定Binary Classification. Loss的意义是将y^n代入f(x)与正确答案$\hat{y}$比较，不一样就返回1，一样返回0，但现在无法通过微分等方法求f.

另一Solution.

从机率问题引入

Two Boxes

Box 1: $P(B_1) = 2/3$; $P(Blue|B_1) = 4/5$; $P(Green|B_1) = 1/5$

Box 2: $P(B_2) = 1/3$; $P(Blue|B_2) = 2/5$; $P(Green|B_2) = 3/5$

从Box 1中抽一个球的机率

from one of the boxes

Where does it come from? 从Box 1中抽到蓝球的机率

$$P(B_1 \mid Blue) = \frac{P(Blue|B_1)P(B_1)}{P(Blue|B_1)P(B_1) + P(Blue|B_2)P(B_2)}$$

⇒把盒子换成类别.

Two Classes

Estimating the Probabilities From training data

Class 1: $P(C_1)$; $P(x|C_1)$

Class 2: $P(C_2)$; $P(x|C_2)$

Given an x, which class does it belong to 是属于class 1的机率

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

Generative Model $P(x) = P(x|C_1)P(C_1) + P(x|C_2)P(C_2)$ 这的就sample到不的机率

$P(C_1)$, $P(C_2)$, $P(x|C_1)$, $P(x|C_2)$ 要通过Training Data算出来

Prior

$P(C_1)$, $P(C_2)$ 叫 Prior

Class 1: $P(C_1)$ — Water

Class 2: $P(C_2)$ — Normal

Water and Normal type with ID < 400 for training, rest for testing (Binary情况)

Training: 79 Water, 61 Normal

$P(C_1) = 79 / (79 + 61) = 0.56$

$P(C_2) = 61 / (79 + 61) = 0.44$

接下来考虑某只宝可梦在某类型中被取出的概率

Probability from Class

$P(x|C_1) = ?$ $P($  $|Water) = ?$

Each Pokémon is represented as a vector by its attribute. → feature

Water Type

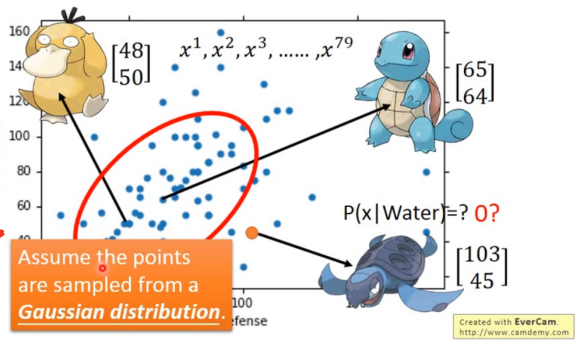
图中海龟并无法在79个里找到，但确实是水系，所以也不能说在水系中选到它概率为0/79

⇒每个宝可梦都是用features的向量表示.

Probability from Class - Feature

- Considering **Defense** and **SP Defense**

先将数值化后可看成水系还有79个，79个是从Gaussian distribution中sample出来的 Water Type



Gaussian Distribution (高斯分布/正态分布)

https://blog.slinuxer.com/tag/pca

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

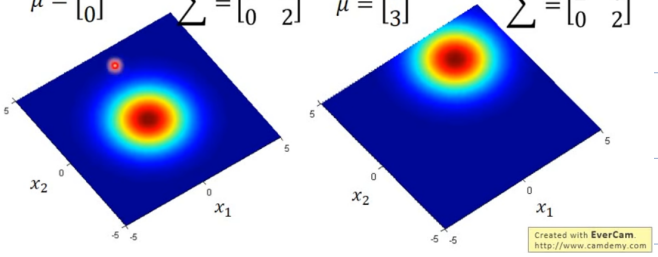
Input: vector x, output: probability of sampling x

The shape of the function determines by **mean μ** and **covariance matrix Σ** ← 代表分布最高点

协方差

μ 代表分布的平均程度.

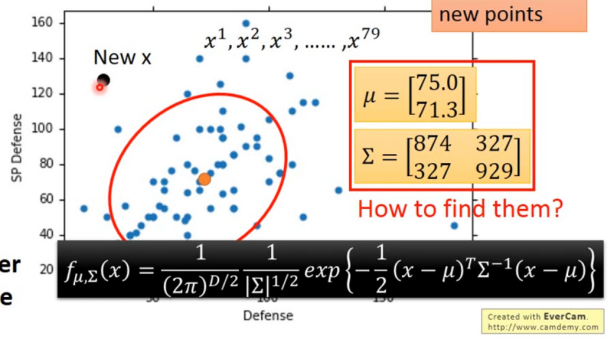
$\mu = \begin{bmatrix}0\\0\end{bmatrix}$ $\sum = \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$ $\mu = \begin{bmatrix}2\\3\end{bmatrix}$ $\sum = \begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$



Probability from Class

Assume the points are sampled from a Gaussian distribution

Find the Gaussian distribution behind them ➡ Probability for new points



算出 μ, Σ 后代入 $f_{\mu,\Sigma}(x)$ 即可得 x 在正态分布中被 sample 出来的概率.

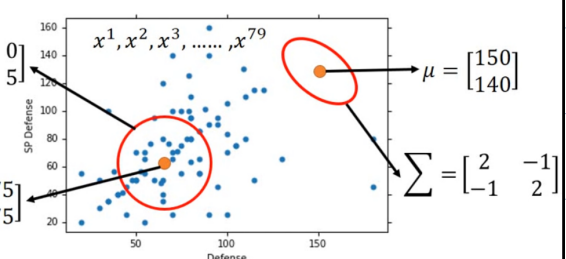
How 找 μ, Σ? ⇒ Maximum Likelihood.

每个 Gaussian distribution 都可能 sample 出这 79 个点. 但每个分布可以 sample 出这 79 个点的概率是不一样的.

Maximum Likelihood

$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$

$\sum = \begin{bmatrix}5 & 0\\0 & 5\end{bmatrix}$ $\mu = \begin{bmatrix}150\\140\end{bmatrix}$ $\sum = \begin{bmatrix}2 & -1\\-1 & 2\end{bmatrix}$ $\mu = \begin{bmatrix}75\\75\end{bmatrix}$



The Gaussian with any mean μ and covariance matrix Σ can generate these points. ➡ Different Likelihood

Likelihood of a Gaussian with mean μ and covariance matrix Σ

= the probability of the Gaussian samples $x^1, x^2, x^3, \ldots\ldots, x^{79}$

$$L(\mu,\Sigma) = f_{\mu,\Sigma}(x^1) f_{\mu,\Sigma}(x^2) f_{\mu,\Sigma}(x^3) \ldots\ldots f_{\mu,\Sigma}(x^{79})$$

sample 出这 79 点的可能性大小, 等于单独选出各点概率之积.

Maximum Likelihood

We have the "Water" type Pokémons: $x^1, x^2, x^3, \ldots\ldots, x^{79}$

We assume $x^1, x^2, x^3, \ldots\ldots, x^{79}$ generate from the Gaussian (μ^*, Σ^*) with the ***maximum likelihood***

$$L(\mu,\Sigma) = f_{\mu,\Sigma}(x^1) f_{\mu,\Sigma}(x^2) f_{\mu,\Sigma}(x^3) \ldots\ldots f_{\mu,\Sigma}(x^{79})$$

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

$$\mu^*, \Sigma^* = arg \max_{\mu,\Sigma} L(\mu,\Sigma)$$

$$\mu^* = \frac{1}{79}\sum_{n=1}^{79} x^n \qquad \Sigma^* = \frac{1}{79}\sum_{n=1}^{79}(x^n-\mu^*)(x^n-\mu^*)^T$$

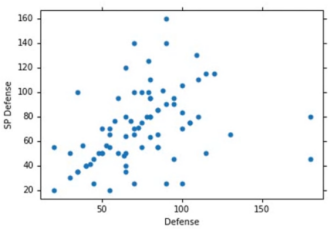
average

算出来就是平均值.

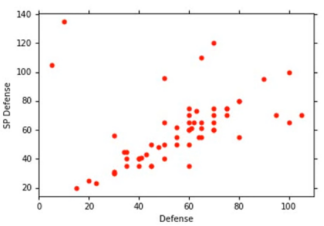
回到实际问题.

Maximum Likelihood

Class 1: Water



Class 2: Normal



$\mu^1 = \begin{bmatrix}75.0\\71.3\end{bmatrix}$ $\Sigma^1 = \begin{bmatrix}874 & 327\\327 & 929\end{bmatrix}$ $\mu^2 = \begin{bmatrix}55.6\\59.8\end{bmatrix}$ $\Sigma^2 = \begin{bmatrix}847 & 422\\422 & 685\end{bmatrix}$

Now we can do classification ☺

$f_{\mu^1,\Sigma^1}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma^1|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu^1)^T(\Sigma^1)^{-1}(x-\mu^1)\right\}$

$\mu^1 = \begin{bmatrix}75.0\\71.3\end{bmatrix}$ $\Sigma^1 = \begin{bmatrix}874 & 327\\327 & 929\end{bmatrix}$

P(C1) = 79 / (79 + 61) =0.56

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

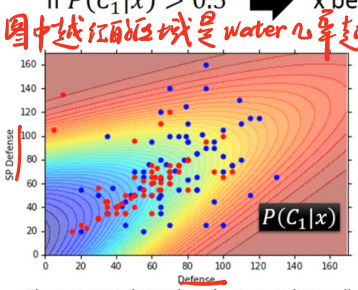
$f_{\mu^2,\Sigma^2}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma^2|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu^2)^T(\Sigma^2)^{-1}(x-\mu^2)\right\}$

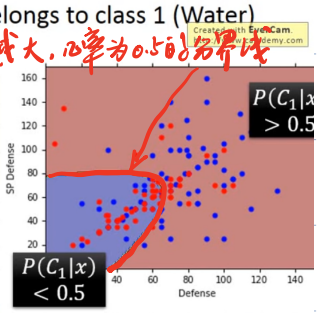
$\mu^2 = \begin{bmatrix}55.6\\59.8\end{bmatrix}$ $\Sigma^2 = \begin{bmatrix}847 & 422\\422 & 685\end{bmatrix}$

P(C2) = 61 / (79 + 61) =0.44

If $P(C_1|x) > 0.5$ ➡ x belongs to class 1 (Water)

图中越红面积域是 water 心率越大, 心率为 0.5 的为界线





Blue points: C_1 (Water), Red points: C_2 (Normal)

How's the results?

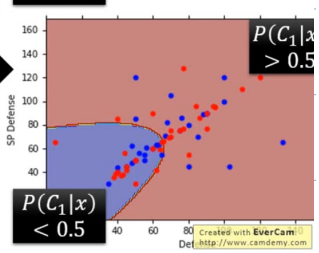
Testing data: 47% accuracy ➡

All: total, hp, att, sp att, de, sp de, speed (7 features)

μ^1, μ^2: 7-dim vector

Σ^1, Σ^2: 7 x 7 matrices

54% accuracy ... ☹

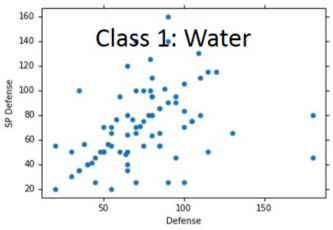


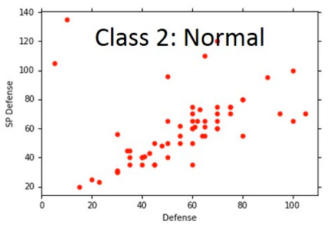
发现就算是用了7个features结果还是不好.

⇒修改模型

上面的都非常用模型.

Modifying Model





$\mu^1 = \begin{bmatrix} 75.0 \\ 71.3 \end{bmatrix}$ $\Sigma^1 = \begin{bmatrix} 874 & 327 \\ 327 & 929 \end{bmatrix}$ $\mu^2 = \begin{bmatrix} 55.6 \\ 59.8 \end{bmatrix}$ $\Sigma^2 = \begin{bmatrix} 847 & 422 \\ 422 & 685 \end{bmatrix}$

The same Σ

Less parameters

上面不同的Gaussian distribution用了不同的Σ,这会造成参数过多,从而容易over fitting.

⇒强行令Σ为同一个.

Modifying Model

Ref: Bishop, chapter 4.2.2

- Maximum likelihood

"Water" type Pokémons: $x^1, x^2, x^3, \ldots\ldots, x^{79}$

"Normal" type Pokémons: $x^{80}, x^{81}, x^{82}, \ldots\ldots, x^{140}$

μ^1 Σ μ^2

Find μ^1, μ^2, Σ maximizing the likelihood $L(\mu^1, \mu^2, \Sigma)$

$$L(\mu^1, \mu^2, \Sigma) = f_{\mu^1,\Sigma}(x^1) f_{\mu^1,\Sigma}(x^2) \cdots f_{\mu^1,\Sigma}(x^{79}) \times f_{\mu^2,\Sigma}(x^{80}) f_{\mu^2,\Sigma}(x^{81}) \cdots f_{\mu^2,\Sigma}(x^{140})$$

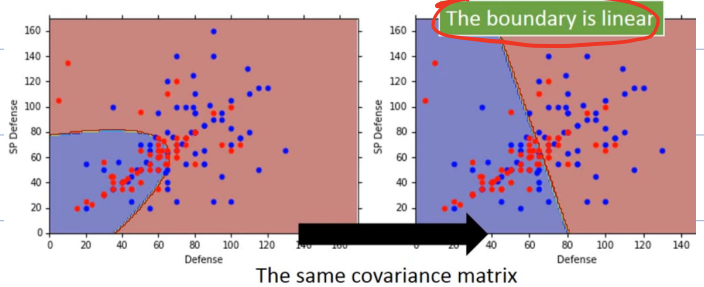
μ^1 and μ^2 is the same (仍是各自的mean.)

$\Sigma = \frac{79}{140}\Sigma^1 + \frac{61}{140}\Sigma^2$

改后的结果

Modifying Model

同一个Σ导致boundary成变为线性的.



The same covariance matrix

All: total, hp, att, sp att, de, sp de, speed

54% accuracy → 73% accuracy

回顾模型.

也是ML的3个steps.

Three Steps

- Function Set (Model):

$x \rightarrow$ $P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$

If $P(C_1|x) > 0.5$, output: class 1

Otherwise, output: class 2

- Goodness of a function:
  - The mean μ and covariance Σ that maximizing the likelihood (the probability of generating data)
- Find the best function: easy

⇒Why用Gaussian distribution?

用其他distribution也行.

简单:参数少:Bias大,Variance小.

复杂,参数多:Bias小,Variance大

Probability Distribution

- You can always use the distribution you like ☺

正常情况下认为各参数间有关系,关系反映在Σ中.

$P(x|C_1) = P(x_1|C_1)\ P(x_2|C_1)\ \cdots\cdots\ P(x_k|C_1)\ \cdots\cdots$

$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ \vdots \\ x_K \end{bmatrix}$

1-D Gaussian

For binary features, you may assume they are from Bernoulli distributions.

即假设各参数间无关.此时多维Gaussian变成多个一维相乘,即Σ变成对角阵

If you assume all the dimensions are independent, then you are using *Naive Bayes Classifier*.

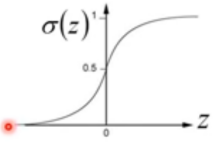
Posterior Probability

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

$$= \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} = \frac{1}{1 + exp(-z)} = \sigma(z)$$

Sigmoid function

$z = ln\frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$



Z长什么样？

经变形后：

$$z = ln\frac{|\Sigma^2|^{1/2}}{|\Sigma^1|^{1/2}} - \frac{1}{2}x^T(\Sigma^1)^{-1}x + (\mu^1)^T(\Sigma^1)^{-1}x - \frac{1}{2}(\mu^1)^T(\Sigma^1)^{-1}\mu^1 + \frac{1}{2}x^T(\Sigma^2)^{-1}x - (\mu^2)^T(\Sigma^2)^{-1}x + \frac{1}{2}(\mu^2)^T(\Sigma^2)^{-1}\mu^2 + ln\frac{N_1}{N_2}$$

Created with EverCam

以上是当$\Sigma_1 \neq \Sigma_2$时，若$\Sigma_1 = \Sigma_2$可化简。

$$P(C_1|x) = \sigma(z)$$

$$z = ln\frac{|\Sigma^2|^{1/2}}{|\Sigma^1|^{1/2}} - \frac{1}{2}x^T(\Sigma^1)^{-1}x + (\mu^1)^T(\Sigma^1)^{-1}x - \frac{1}{2}(\mu^1)^T(\Sigma^1)^{-1}\mu^1 + \frac{1}{2}x^T(\Sigma^2)^{-1}x - (\mu^2)^T(\Sigma^2)^{-1}x + \frac{1}{2}(\mu^2)^T(\Sigma^2)^{-1}\mu^2 + ln\frac{N_1}{N_2}$$

$$\Sigma_1 = \Sigma_2 = \Sigma$$

$$z = \underbrace{(\mu^1 - \mu^2)^T\Sigma^{-1}}_{w^T}x \underbrace{- \frac{1}{2}(\mu^1)^T(\Sigma^1)^{-1}\mu^1 + \frac{1}{2}(\mu^2)^T(\Sigma^2)^{-1}\mu^2 + ln\frac{N_1}{N_2}}_{b}$$

$$P(C_1|x) = \sigma(w \cdot x + b)$$

一但w和b是对一个C而言，如此处w和b是对C_1

Created with EverCam http://www.camdemy.com

Logistic Regression

Step 1: Function Set

We want to find $P_{w,b}(C_1|x)$

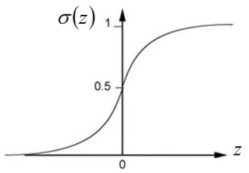
If $P_{w,b}(C_1|x) \geq 0.5$, output C_1

Otherwise, output C_2

$$P_{w,b}(C_1|x) = \sigma(z)$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$



Function set:

$$f_{w,b}(x) = P_{w,b}(C_1|x)$$

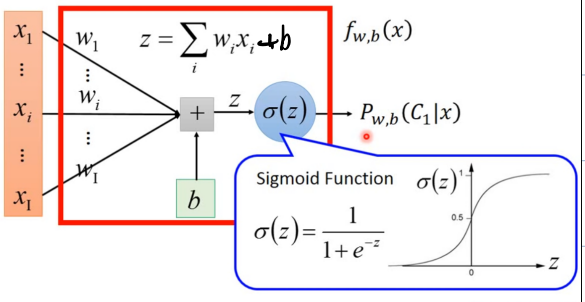
Including all different w and b

Created with EverCam http://www.camdemy.com

回顾上节课，function set是与w和b有关的。

将模型图形化：

Step 1: Function Set



Created with EverCam http://www.camdemy.com

上节课是通过找最好的Σ和μ来找到最好的function，但由于最后function可以由w和b来表示，所以可以直接找最好的w和b.

Step 2: Goodness of a Function

Training Data

x^1	x^2	x^3	……	x^N
C_1	C_1	C_2		C_1

Assume the data is generated based on $f_{w,b}(x) = P_{w,b}(C_1|x)$

Given a set of w and b, what is its probability of generating the data?

判断一组w和b可以产生这组Training Data的概率

$$L(w,b) = f_{w,b}(x^1)f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right)\cdots f_{w,b}(x^N)$$

产生C2的概率

The most likely w^* and b^* is the one with the largest $L(w,b)$.

$$w^*, b^* = arg\max_{w,b} L(w,b)$$

Created with EverCam http://www.camdemy.com

x^1	x^2	x^3	……
C_1	C_1	C_2	

x^1	x^2	x^3	……
$\hat{y}^1 = 1$	$\hat{y}^2 = 1$	$\hat{y}^3 = 0$	……

和regression一样有真实值分，只不过此处设为0和1.

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$L(w,b) = f_{w,b}(x^1)f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right)\cdots$$

进行数学上的变换

$$w^*, b^* = arg\max_{w,b} L(w,b) = w^*, b^* = arg\min_{w,b} -lnL(w,b)$$

$$-lnL(w,b)$$

由于用了ln，可拆成加法.

$$= -lnf_{w,b}(x^1) \Rightarrow -[\hat{y}^1 lnf(x^1) + (1-\hat{y}^1)ln(1-f(x^1))]$$

$$-lnf_{w,b}(x^2) \Rightarrow -[\hat{y}^2 lnf(x^2) + (1-\hat{y}^2)ln(1-f(x^2))]$$

$$-ln\left(1 - f_{w,b}(x^3)\right) \Rightarrow -[\hat{y}^3 lnf(x^3) + (1-\hat{y}^3)ln(1-f(x^3))]$$

$$\vdots$$

数学转化

Created with EverCam http://www.camdemy.com

⇒通过上述数学转化将要minimize的对象写成一个function.

Step 2: Goodness of a Function

$$L(w,b) = f_{w,b}(x^1)f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right)\cdots f_{w,b}(x^N)$$

$$-lnL(w,b) = lnf_{w,b}(x^1) + lnf_{w,b}(x^2) + ln\left(1 - f_{w,b}(x^3)\right)\cdots$$

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$= \sum_n -\left[\hat{y}^n lnf_{w,b}(x^n) + (1-\hat{y}^n)ln\left(1 - f_{w,b}(x^n)\right)\right]$$

Cross entropy between two Bernoulli distribution

交叉熵

伯努利分布，即两点分布

Distribution p:		Distribution q:
$p(x=1) = \hat{y}^n$	cross entropy	$q(x=1) = f(x^n)$
$p(x=0) = 1 - \hat{y}^n$		$q(x=0) = 1 - f(x^n)$

$$H(p,q) = -\sum_x p(x)ln(q(x))$$

Created with EverCam http://www.camdemy.com

算Distribution p和q的cross entropy，代表的是算出它们之间多接近

	Logistic Regression	Linear Regression
Step 1:	$f_{w,b}(x)=\sigma\left(\sum_i w_i x_i+b\right)$ Output: between 0 and 1	$f_{w,b}(x)=\sum_i w_i x_i+b$ Output: any value
Step 2:	Training data: $(x^n,\hat{y}^n)$ $\hat{y}^n$: 1 for class 1, 0 for class 2 $L(f)=\sum_n C(f(x^n),\hat{y}^n)$	Training data: $(x^n,\hat{y}^n)$ $\hat{y}^n$: a real number $L(f)=\frac{1}{2}\sum_n (f(x^n)-\hat{y}^n)^2$

将Linear model代进Sigmoid函数

Cross entropy:

$$C(f(x^n),\hat{y}^n)=-[\hat{y}^n \ln f(x^n)+(1-\hat{y}^n)\ln(1-f(x^n))]$$

我希望把$f(x^n)$和$\hat{y}^n$看成2个伯努力分布，这两个分布越接近其结果越好

Created with EverCam. http://www.camdemy.com

进行 Gradient descent.

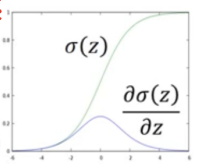
Step 3: Find the best function

$$-\frac{\partial \ln L(w,b)}{\partial w_i}=\sum_n -\left[\hat{y}^n \frac{\partial \ln f_{w,b}(x^n)}{\partial w_i}+(1-\hat{y}^n)\frac{\partial \ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$$\frac{\partial \ln f_{w,b}(x)}{\partial w_i}=\frac{\partial \ln f_{w,b}(x)}{\partial z}\frac{\partial z}{\partial w_i} \qquad \frac{\partial z}{\partial w_i}=x_i$$

复合函数求导

$$\frac{\partial \ln \sigma(z)}{\partial z}=\frac{1}{\sigma(z)}\frac{\partial \sigma(z)}{\partial z}=\frac{1}{\sigma(z)}\sigma(z)(1-\sigma(z))$$



$$f_{w,b}(x)=\sigma(z)=1/1+exp(-z) \qquad z=w\cdot x+b=\sum_i w_i x_i+b$$

Created with EverCam. http://www.camdemy.com

Step 3: Find the best function

$$-\frac{\partial \ln L(w,b)}{\partial w_i}=\sum_n -\left[\hat{y}^n \frac{\partial \ln f_{w,b}(x^n)}{\partial w_i}+(1-\hat{y}^n)\frac{\partial \ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$$\frac{\partial \ln\left(1-f_{w,b}(x)\right)}{\partial w_i}=\frac{\partial \ln\left(1-f_{w,b}(x)\right)}{\partial z}\frac{\partial z}{\partial w_i} \qquad \frac{\partial z}{\partial w_i}=x_i$$

$$\frac{\partial \ln(1-\sigma(z))}{\partial z}=-\frac{1}{1-\sigma(z)}\frac{\partial \sigma(z)}{\partial z}=-\frac{1}{1-\sigma(z)}\sigma(z)(1-\sigma(z))$$

$$f_{w,b}(x)=\sigma(z)=1/1+exp(-z) \qquad z=w\cdot x+b=\sum_i w_i x_i+b$$

Created with EverCam. http://www.camdemy.com

Step 3: Find the best function

$$-\frac{\partial \ln L(w,b)}{\partial w_i}=\sum_n -\left[\hat{y}^n \frac{\partial \ln f_{w,b}(x^n)}{\partial w_i}+(1-\hat{y}^n)\frac{\partial \ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$$=\sum_n -\left[\hat{y}^n\left(1-f_{w,b}(x^n)\right)x_i^n-(1-\hat{y}^n)f_{w,b}(x^n)x_i^n\right]$$

$$=\sum_n -\left[\hat{y}^n-\hat{y}^n f_{w,b}(x^n)-f_{w,b}(x^n)+\hat{y}^n f_{w,b}(x^n)\right]x_i^n$$

$$=\sum_n -\left(\hat{y}^n-f_{w,b}(x^n)\right)x_i^n$$

Larger difference, larger update

$$w_i \leftarrow w_i-\eta\sum_n -\left(\hat{y}^n-f_{w,b}(x^n)\right)x_i^n$$

Created with EverCam. http://www.camdemy.com

Logistic Regression和Linear Regression用Gradient descent来update参数的式子是一样的

均为 $w_i \leftarrow w_i-\eta\sum_n -(\hat{y}^n-f_{w,b}(x^n))x_i^n$.

回到前面一个问题

在step2中，两种Regression都是要去找使得$f(x^n)$和$\hat{y}^n$最小的w和b，那为什么Logistic Regression不能像Linear Regression一样用Square Error来判断，而是要用Cross entropy？

Logistic Regression + Square Error

Step 1: $f_{w,b}(x)=\sigma\left(\sum_i w_i x_i+b\right)$

Step 2: Training data: $(x^n,\hat{y}^n)$, $\hat{y}^n$: 1 for class 1, 0 for class 2

$$L(f)=\frac{1}{2}\sum_n (f_{w,b}(x^n)-\hat{y}^n)^2$$

Step 3:

$$\frac{\partial (f_{w,b}(x)-\hat{y})^2}{\partial w_i}=2(f_{w,b}(x)-\hat{y})\frac{\partial f_{w,b}(x)}{\partial z}\frac{\partial z}{\partial w_i}$$

$$=2(f_{w,b}(x)-\hat{y})f_{w,b}(x)\left(1-f_{w,b}(x)\right)x_i$$

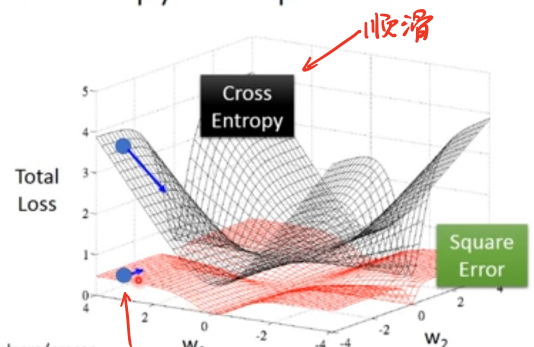
$\hat{y}^n=1$ 以class 1为例

If $f_{w,b}(x^n)=1$ (close to target) $\Rightarrow \partial L/\partial w_i=0$ ✓

If $f_{w,b}(x^n)=0$ (far from target) $\Rightarrow \partial L/\partial w_i=0$ ✗

Created with EverCam. http://www.camdemy.com

Cross Entropy v.s. Square Error



顺滑

http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

在为Target远的地方反而走得慢

Created with EverCam. http://www.camdemy.com

关于Discriminative model和Generative model

上述课上是先用Generative model：即先假定一个分布(高斯)，然后再去找Σ, μ^1, μ^2，同时发现所得就是只和2个参数有关，即w和b。不过有可能w和b并非只有通过Generative model才可以

化出，而是本来$P(C_1|x)$就是由2个参数得来，只是通过特殊的Generative model揭露出这一事实。这种认为$P(C_1|x)$只与w和b有关而与怎么分布无关的称为Discriminative model。前面Logistic regression就是用这种model。

Discriminative v.s. Generative

$$P(C_1|x) = \sigma(w \cdot x + b)$$

directly find **w** and b

Find $\mu^1, \mu^2, \Sigma^{-1}$

$$w^T = (\mu^1 - \mu^2)^T \Sigma^{-1}$$

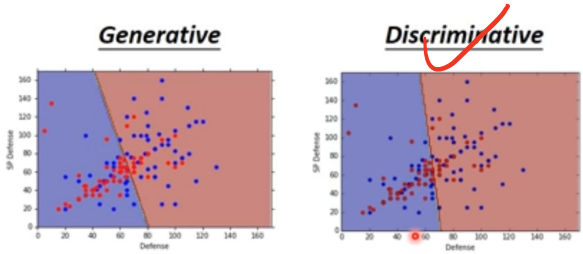
$$b = -\frac{1}{2}(\mu^1)^T(\Sigma^1)^{-1}\mu^1 + \frac{1}{2}(\mu^2)^T(\Sigma^2)^{-1}\mu^2 + \ln\frac{N_1}{N_2}$$

Will we obtain the same set of w and b?

The same model (function set), but different function is selected by the same training data.

以上两者选出的w和b不是同一个，哪个更好？

Generative v.s. Discriminative



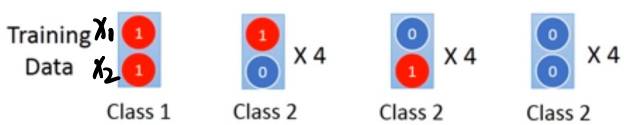
All: total, hp, att, sp att, de, sp de, speed

73% accuracy　　79% accuracy

解释：

Generative v.s. Discriminative

- Example



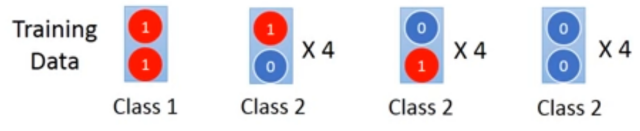
Testing Data　Class 1 Class 2?

How about Naïve Bayes?　认为各几率之间独立。

$$P(x|C_i) = P(x_1|C_i)P(x_2|C_i)$$

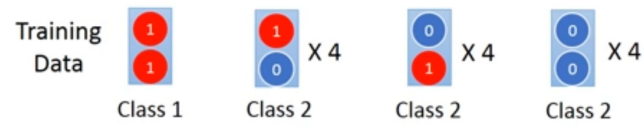
Generative v.s. Discriminative

- Example



$$P(C_1) = \frac{1}{13} \quad P(x_1 = 1|C_1) = 1 \quad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \quad P(x_1 = 1|C_2) = \frac{1}{3} \quad P(x_2 = 1|C_2) = \frac{1}{3}$$



Testing Data

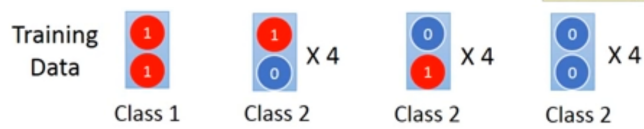
$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

<0.5

1×1　$\frac{1}{13}$　$\frac{1}{3}\times\frac{1}{3}$　$\frac{12}{13}$

$$P(C_1) = \frac{1}{13} \quad P(x_1 = 1|C_1) = 1 \quad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \quad P(x_1 = 1|C_2) = \frac{1}{3} \quad P(x_2 = 1|C_2) = \frac{1}{3}$$



Testing Data

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

<0.5

1×1　$\frac{1}{13}$　$\frac{1}{3}\times\frac{1}{3}$　$\frac{12}{13}$

$$P(C_1) = \frac{1}{13} \quad P(x_1 = 1|C_1) = 1 \quad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \quad P(x_1 = 1|C_2) = \frac{1}{3} \quad P(x_2 = 1|C_2) = \frac{1}{3}$$

错误原因：Generative model通过分布假设，脑补"模型"，它可能会脑补出Class 2中也有出现$x_1=1, x_2=1$的情况，所以把$x_1=1, x_2=1$归为class 2。

⇒ Discriminative model不一定求这结果比Generative model好。由于Discriminative model是看着data说话的，Generative model由于有自己的假设所以没有那么吃data的。

所以一般 data 少时 Generative 结果好. (或有些 data 有误时)

data 多时 Discriminative 结果好.

对3类及以上的 Classification.

Multi-class Classification (3 classes as example)

[Bishop, P209-210]

$C_1: w^1, b_1 \quad z_1 = w^1 \cdot x + b_1$

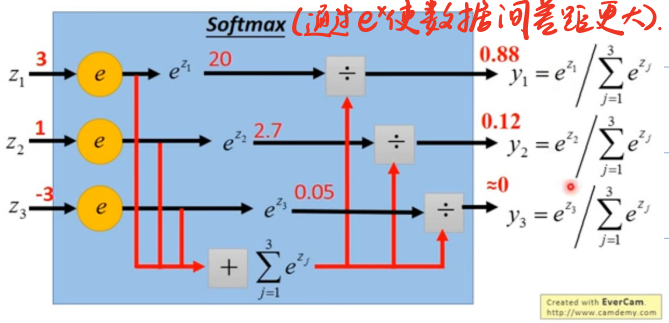
$C_2: w^2, b_2 \quad z_2 = w^2 \cdot x + b_2$

$C_3: w^3, b_3 \quad z_3 = w^3 \cdot x + b_3$

Probability:

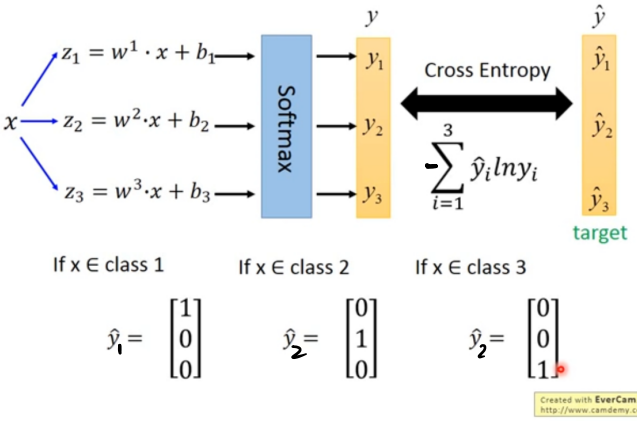
- $1 > y_i > 0$
- $\sum_i y_i = 1$

$y_i = P(C_i \mid x)$



可以得到通过 Softmax function 得到 y_i. 并把 y_i 当作 $P(C_i|x)$. (在前面. 3会通过 sigmoid function, 这里相当于换了一个 function). (使用 Softmax function 可以被证明. 若3个 class 都符合高斯分布且 share the same Σ, 即可证明)

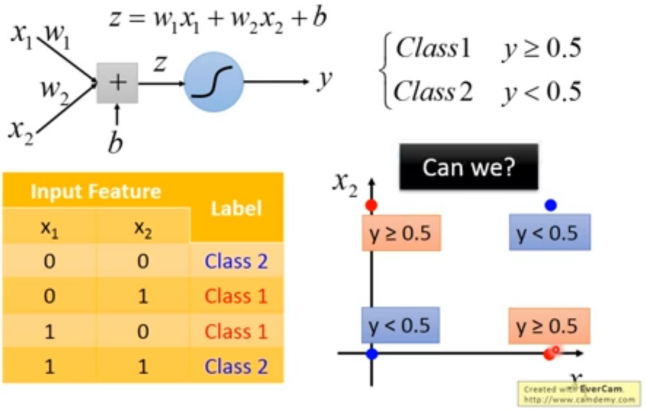
Multi-class Classification (3 classes as example)



上图不能给 $\hat{y_1}=1$. $\hat{y_2}=2$. $\hat{y_3}=3$. 若这样设则默认 class 1和2更有关. class 2和3更有关; 而 class 1和3更无关.

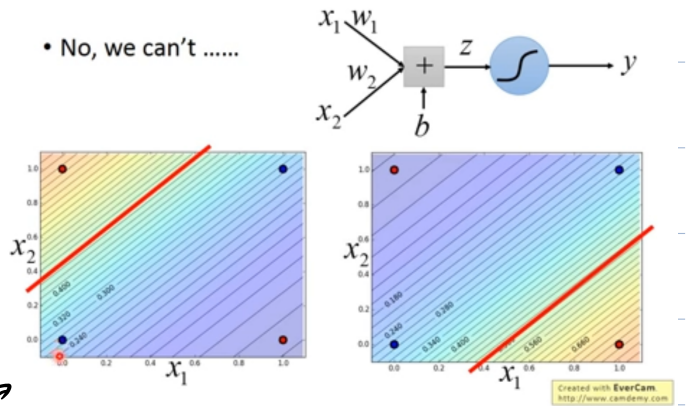
Logistic Regression 有非常强的限制性.

Limitation of Logistic Regression



Limitation of Logistic Regression

- No, we can't .....



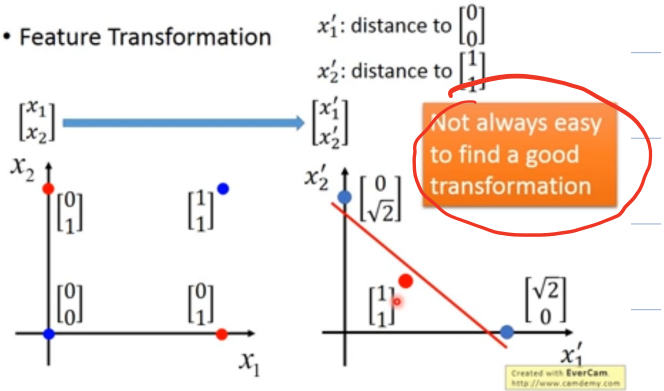
如上. Logistic Regression 始终是线性的. 它只能在 x_1, x_2 平面上画一条直线, 要是出现上面那种无法放在直线同一边的情况就做不了.

⇒解决方法: Feature Transformation.

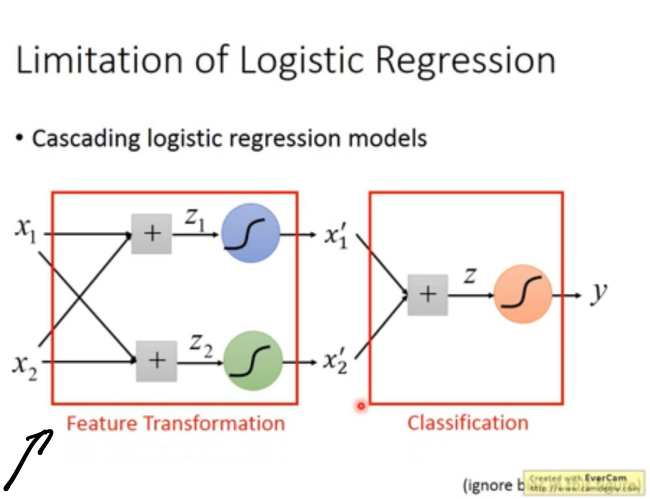
(即坐标转换. 让其变成可被一条线分割的)

Limitation of Logistic Regression

- Feature Transformation



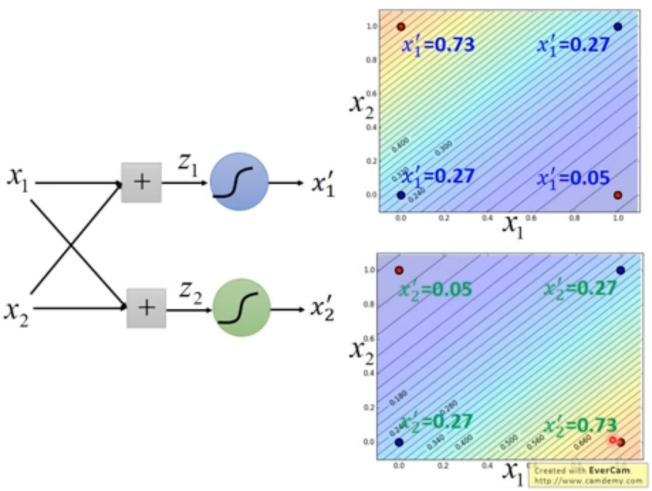
尝试让机器自己找一个 transformation.

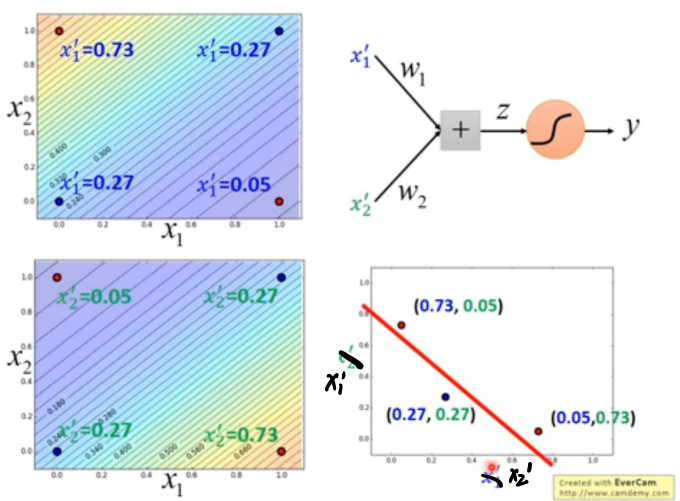


即多次通过 logistic regression models 来 transformation

特征们都乘不同 weights 后通过 sigmoid 来得到新生特征.

实例





像上面一样用多个 Logistic Regression 层叠解决问题叫 Neural Network (类神经网络)