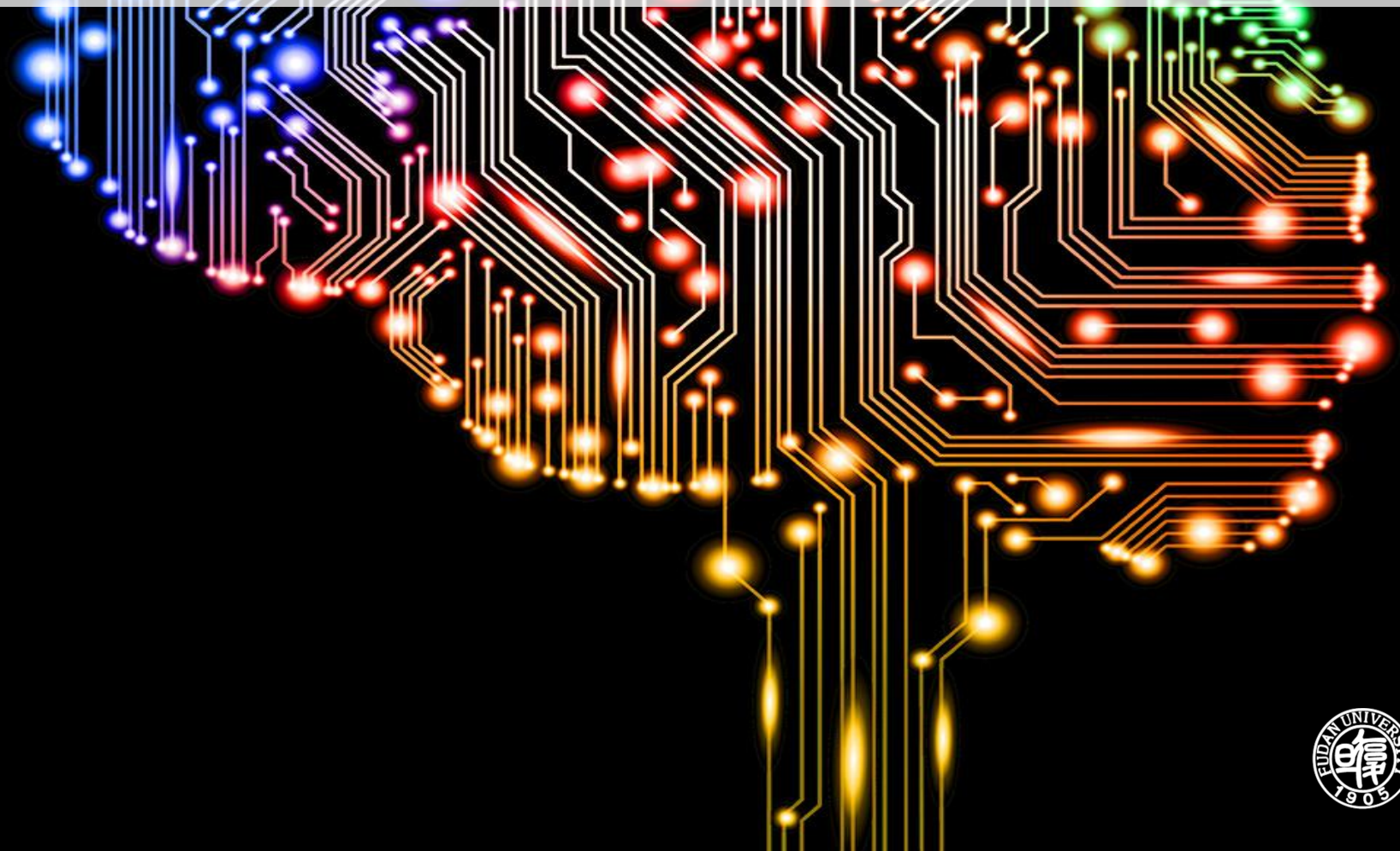
数据驱动的人工智能（2b）机器学习基础

Data Driven Artificial Intelligence

邬学宁 SAP硅谷创新中心

2017 / 02



目录

日程

Multi-Variable Linear Regression

Machine Learning System Design

Logistic / Softmax Regression

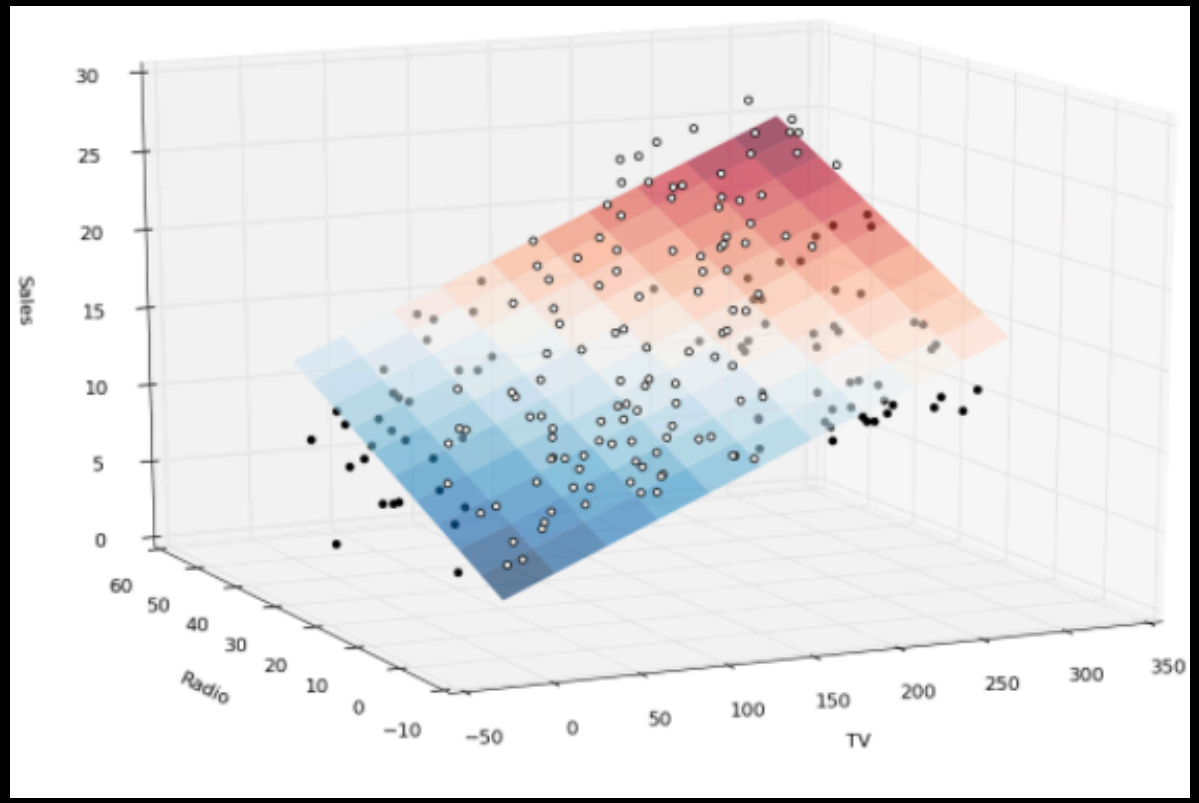
Normal Equation

Cross-entropy

Perceptron

Feeding Forward MLP

Multivariate Linear Regression



Model:

$Sales = \theta_0 + \theta_1 * Radio + \theta_2 * TV$

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

Source: http://nbviewer.ipython.org/urls/s3.amazonaws.com/datarobotblog/notebooks/multiple_regression_in_python.ipynb

Multivariate Linear Regression

Model Representation

$$H_\theta(x) = \theta_0 + \theta_1 x_1 + \bullet\bullet\bullet + \theta_n x_n$$
$$= \sum_{i=0}^{n} \theta_i x_i \;=\; \theta^T X \qquad (x_0 = 1)$$

Multivariate Linear Regression

Multiple features

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Credit: Andrew Ng

Notation:

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

Multivariate Linear Regression

Multiple features（多变量线性回归）

Model:

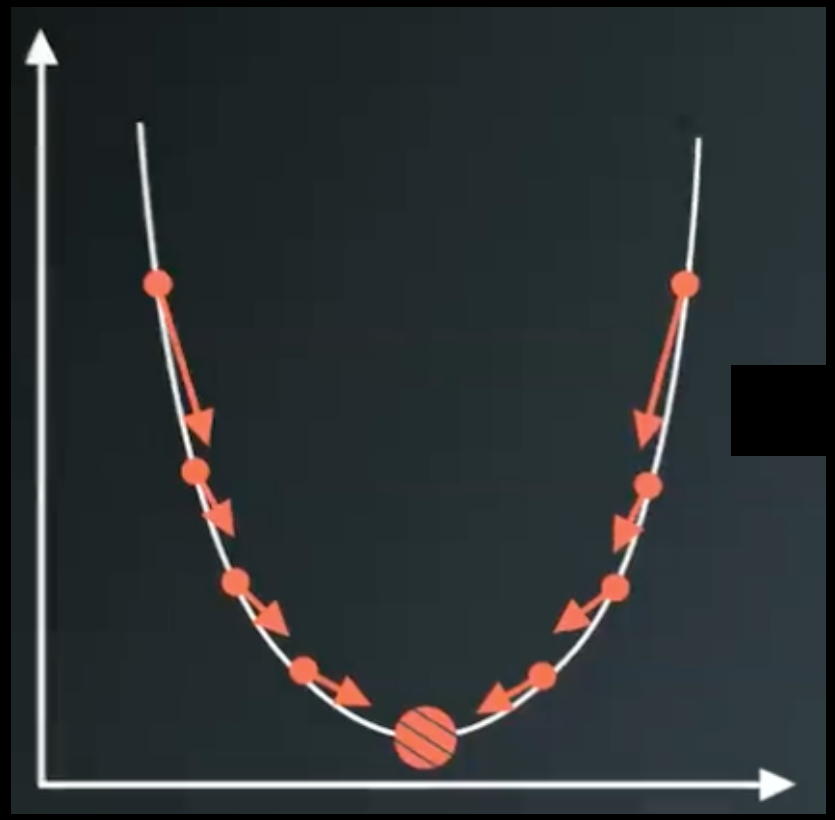
$$h_\theta(\mathrm{x}) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \qquad (x_0 = 1)$$
$$= \theta^T x$$

Parameters: $\theta_0 \ , \theta_1 \ , \cdots \ , \theta_n$

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

Gradient Descent step: $\theta := \theta - \alpha \nabla_\theta \mathcal{J}$

The Direction of Gradient



Source: Udacity

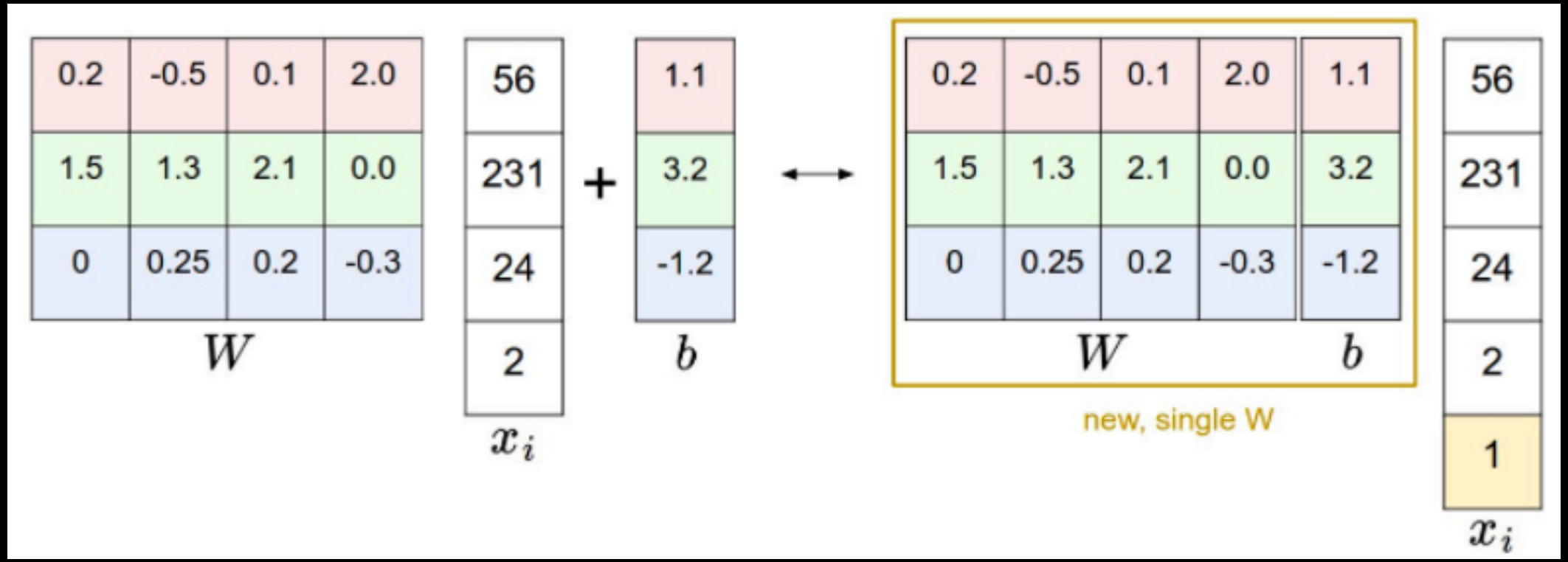
$$\theta_{i=}\theta_i + \Delta\,\theta_i$$

$$\Delta\,\theta_i \propto -\frac{\partial \mathcal{J}}{\partial \theta_i}$$

$$\Delta\,\theta_i = -\alpha\frac{\partial \mathcal{J}}{\partial \theta_i}$$

Multivariate Linear Regression

The Bias Trick



Multivariate Linear Regression

Gradient Descent（梯度下降）

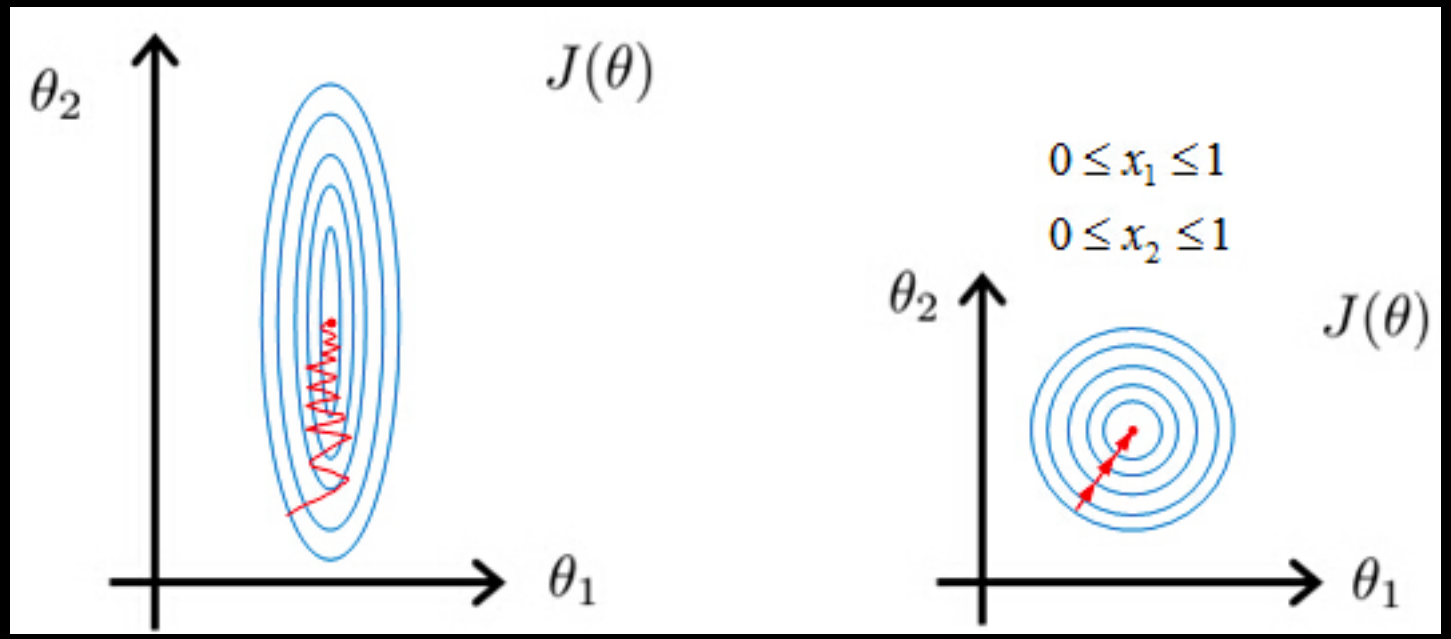
♻迭代至收敛，每 Iteration / epoch 内使用矢量表达避免循环

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

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

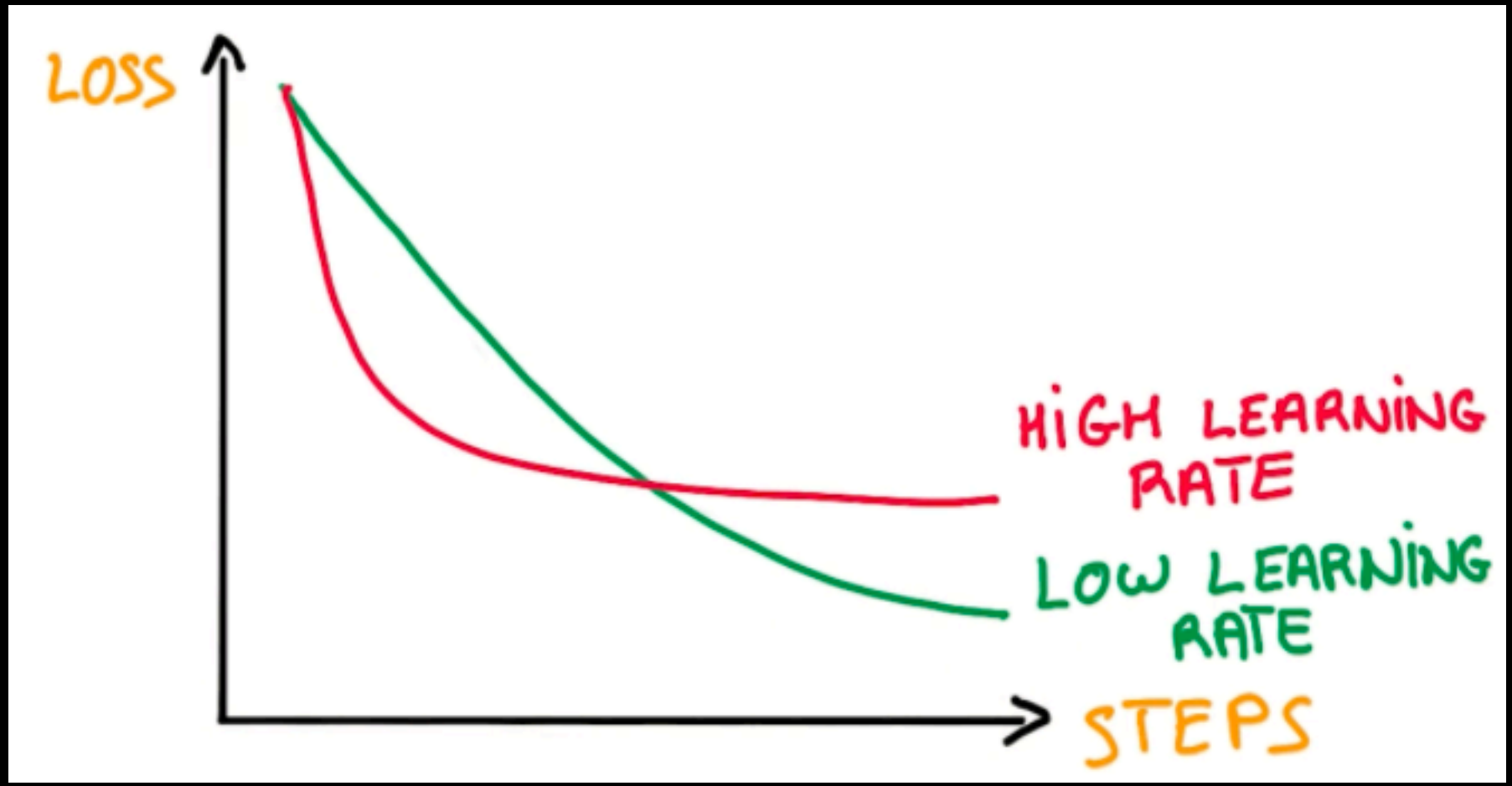
- 程序运行期间，需要监控代价函数，确保其不但不断下降，而且还不至于太慢
- 合理选择α学习率
- Declare $\mathcal{J}(\boldsymbol{\theta})$ is convergence if decreases by less than 10^{-3} in one iteration

Feature Scaling



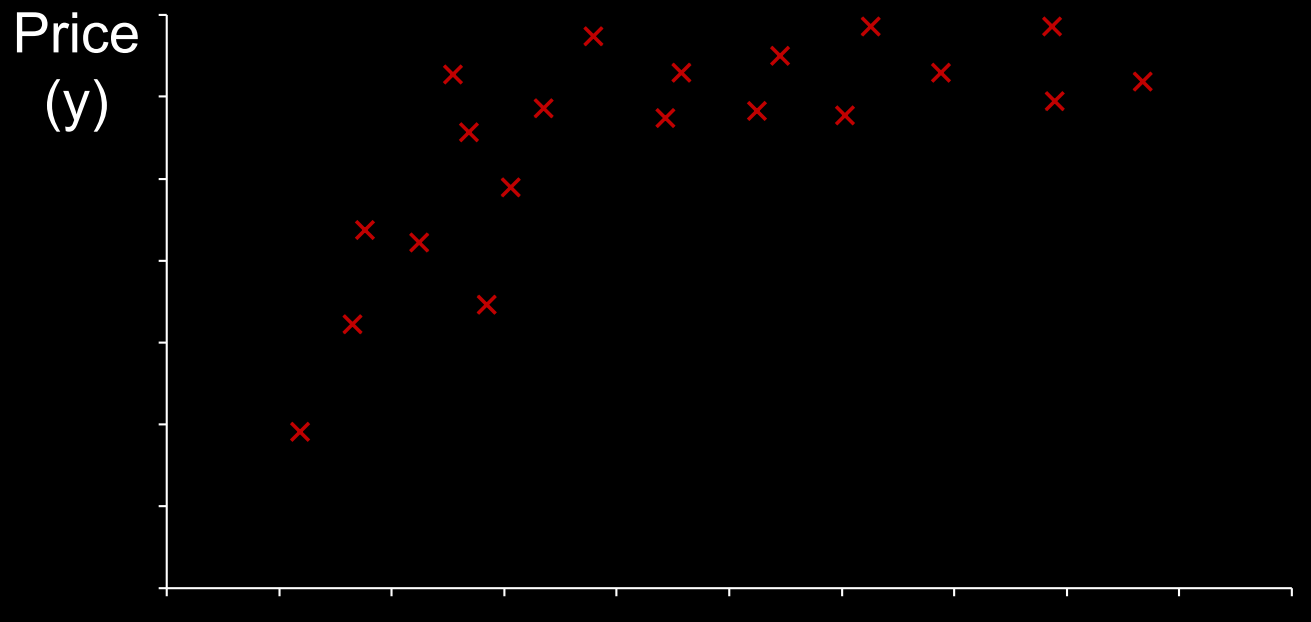
公式：$x' := \dfrac{x - x_{min}}{x_{max} - x_{min}}$ 或 $x' := \dfrac{x - \bar{x}}{\sigma(x)}$

Learning Rate & Cost



Source: Udacity

Polynomial Regression (多项式回归)



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_2 x^2$$

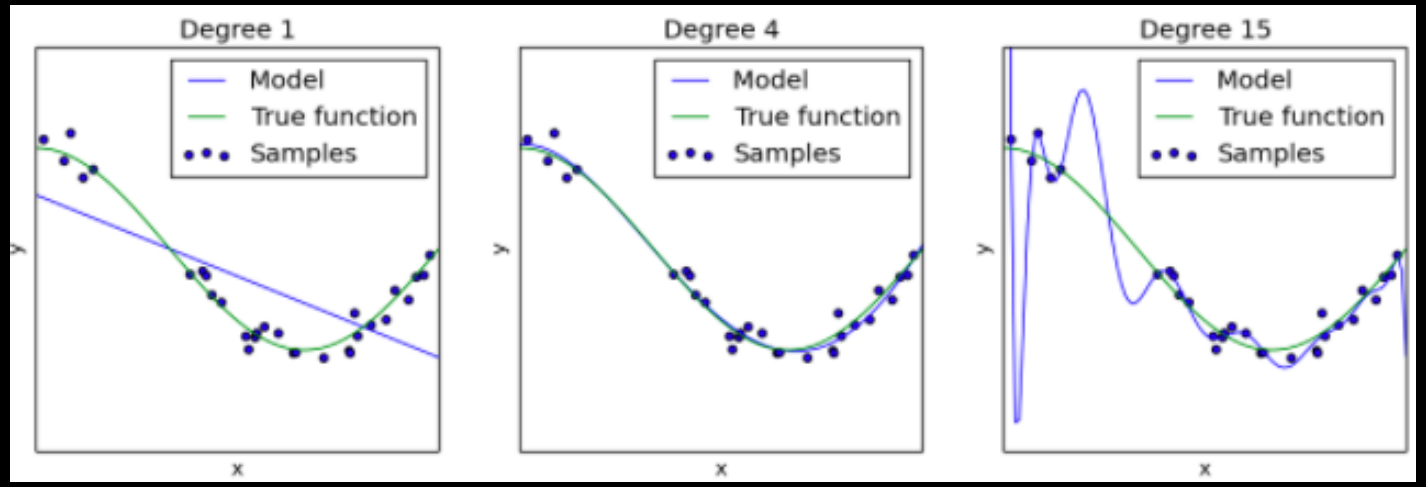
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_2 \sqrt{x}$$

$$h_\theta(\mathrm{x}) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \qquad (x_0 = 1)$$

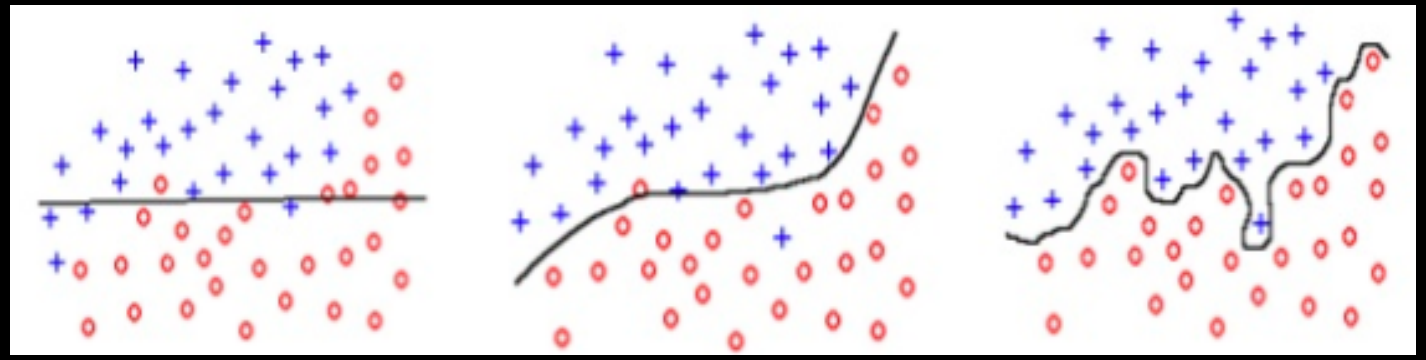
$$= \theta_0 + \theta_1 (size) + \theta_2 (size^2) + \theta_2 (size^3)$$

Feature scaling变得很重要，另外，我们也可以对不同的特征进行组合，例如$x_1 x_2$

欠拟合 vs 过拟合(Underfit vs Overfit)



Underfit Goodrfit Overfit



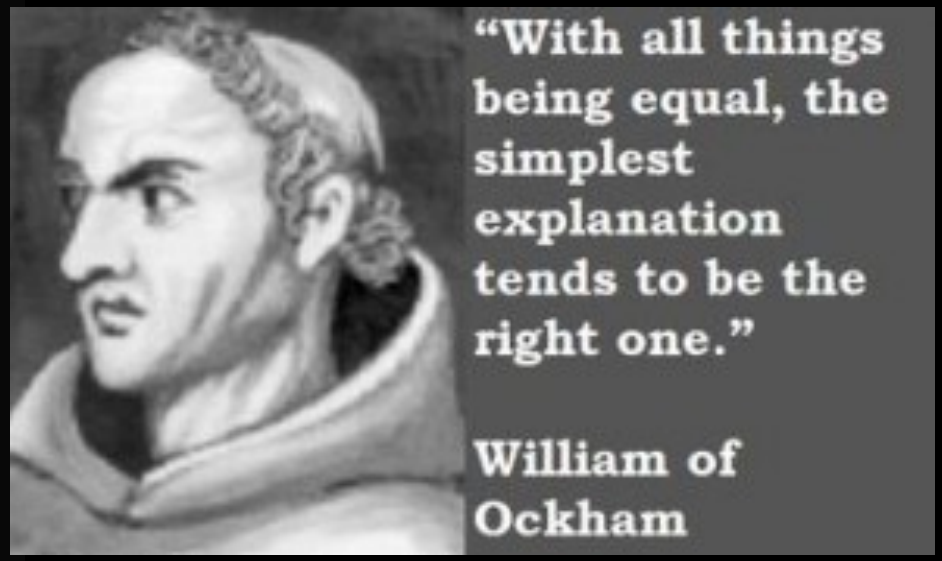
Underfit:
模型过于简单，不足以表达数据的复杂性

http://scikit-learn.org/0.15/auto_examples/plot_underfitting_overfitting.html

Underfit:
模型过于复杂，把“噪音”也学习了，奥卡姆剃刀(Occam razor)

Credit: Vojtech Franc

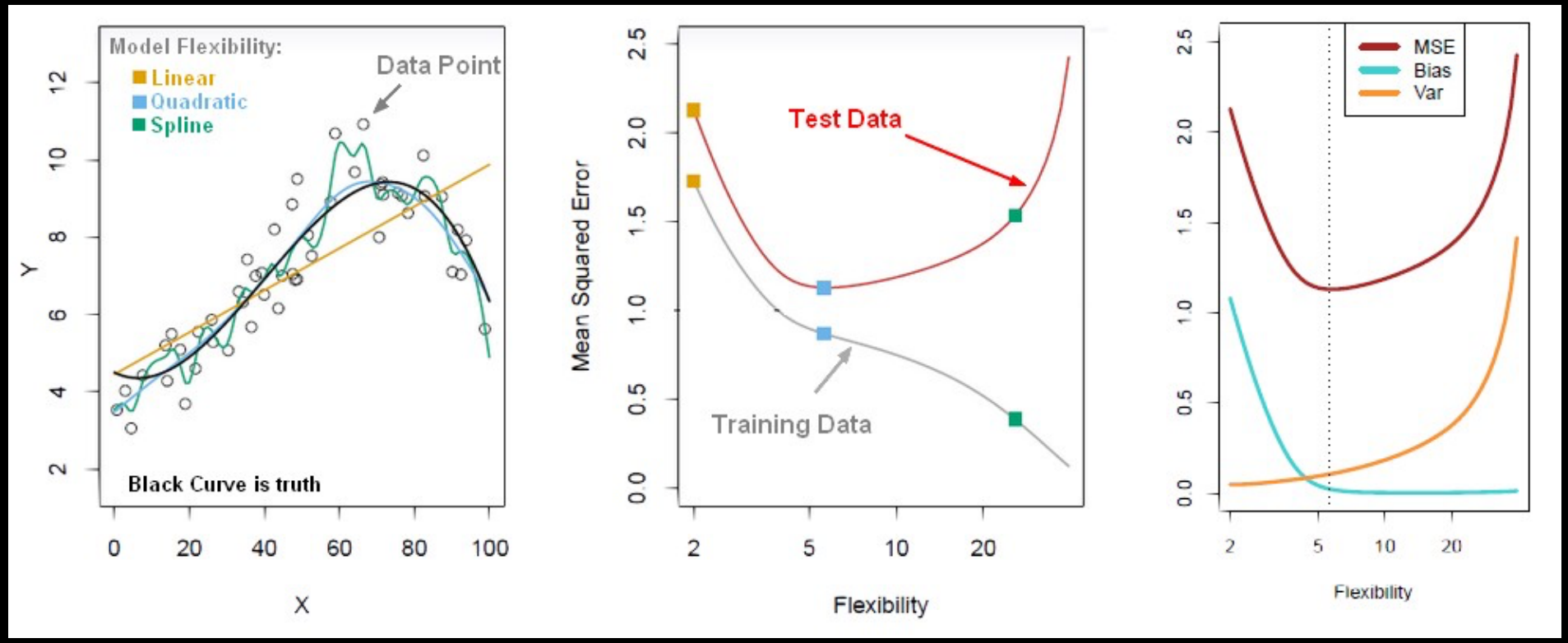
奥卡姆剃刀(Occam razor)



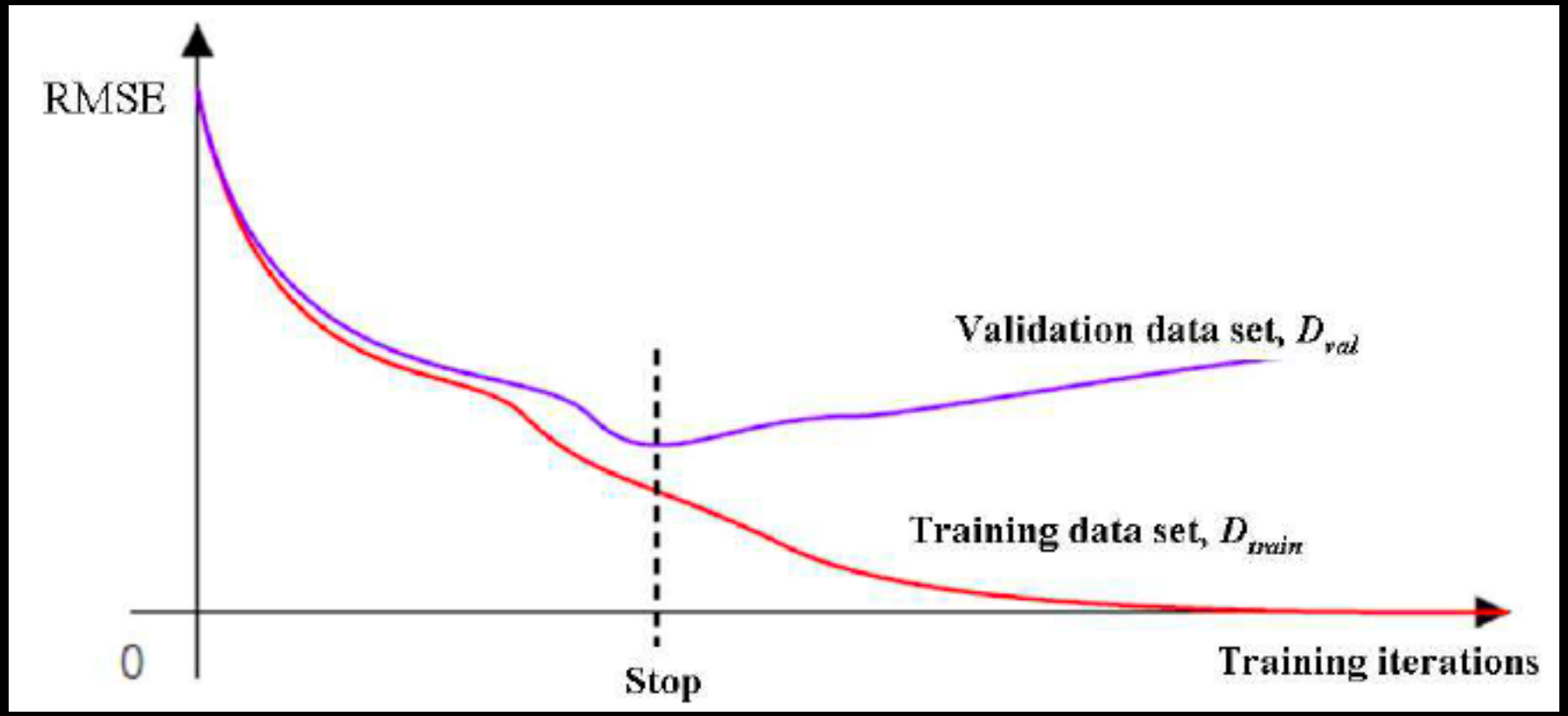
Tom M. Mitchell say's…Occam got this idea during shaving

Wikipedia say's... The term razor refers to the act of shaving away unnecessary assumptions to get the simplest explanation.

欠拟合 vs 过拟合(Underfit vs Overfit)



Early Stopping



https://www.researchgate.net/figure/223790695_fig2_Figure-2-Early-stopping-the-ANN-training-to-avoid-overfitting

L2 Regularization

$$\mathcal{J}' = \mathcal{J} + \frac{1}{2}\lambda\|\theta\|_2^2$$

New Loss　Loss

$$\mathcal{J}(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right)$$

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

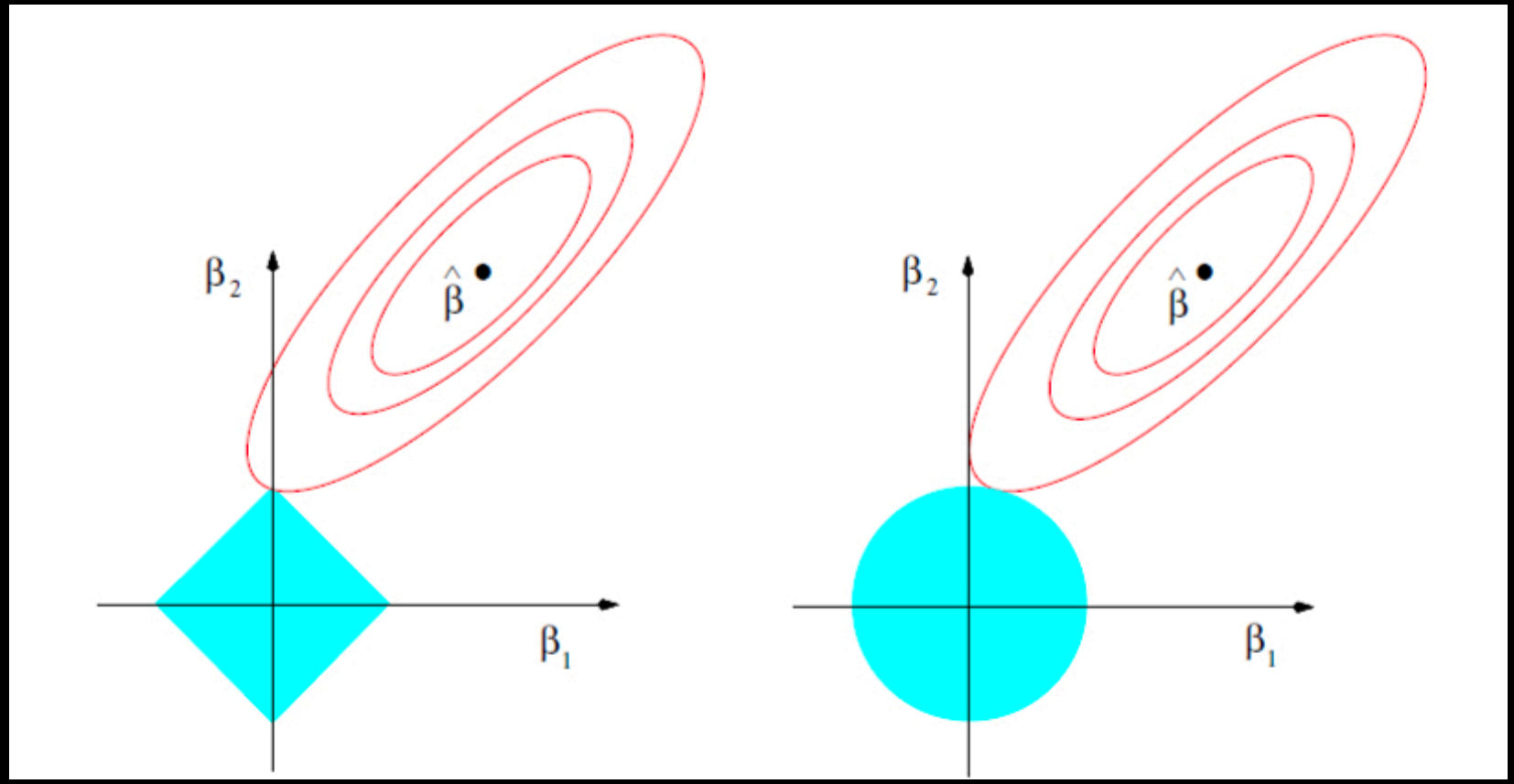
$$= \theta_j - \alpha\frac{1}{m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)} + \lambda\theta_j\right)$$

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

由于α通常很小，而m一般较大，所以$(\alpha\frac{\lambda}{m})$一般是个很小的正数。

L1 & L2 Regularization

- 变量收缩
- 进行变量筛选（线性相关变量）



Lasso

$|\beta_1|+|\beta_2| < t$

ElasticNet

Ridge

$\beta_1^2 + \beta_2^2 = t^2$

- 求导方便
- 不能进行变量筛选
- 很多小的权重

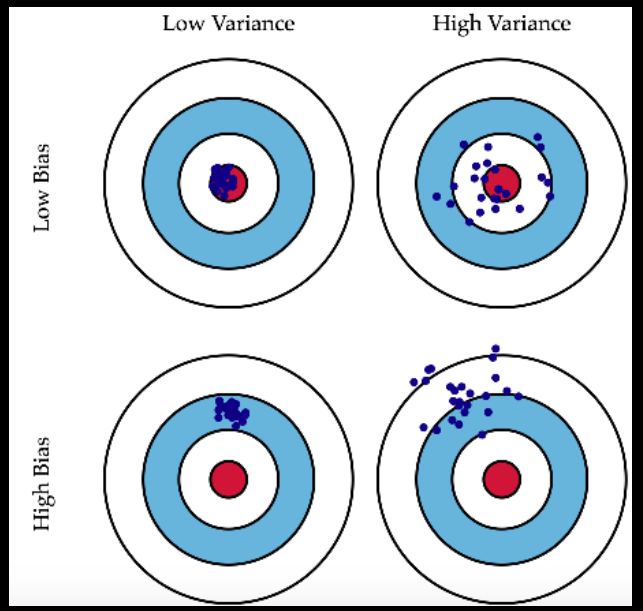
The Elements of statistical learning

High Bias vs High Variance

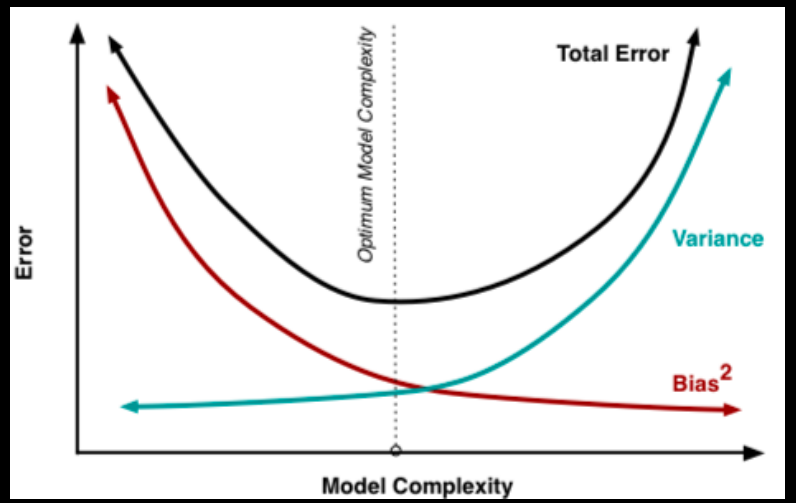
Bias: 模型的期望（平均）预测值与实际值*的差异。假设我们可以多次重复建模过程，每次基于新的数据，进行新的建模（$\hat{f}(x)$），每次建模都是对理想模型$f(x)$的一次估计($Y = f(x) + \varepsilon$).

Bias = $E[\hat{y}] - f(x)$, 被用来测量一类估计模型$\hat{f}(x)$整体的预测值与理想模型$f(x)$实际值的偏差。

Variance = $E[(\hat{y} - E[\hat{y}])^2]$ 模型预测值与实际值的差异的变化（同样，假设我们多次进行建模）



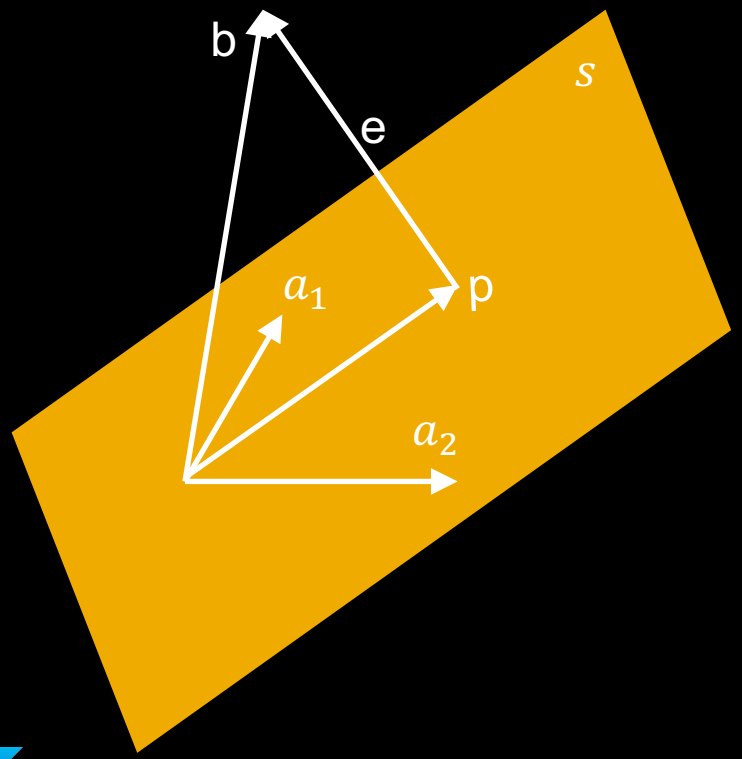
$$
\begin{aligned}
\text{MSE} &= E[(f(x) - \hat{y})^2] \\
&= E[(f(x) - E[\hat{y}] + E[\hat{y}] - \hat{y})^2] \\
&= E[(f(x) - E[\hat{y}] + E[\hat{y}] - \hat{y})(f(x) - E[\hat{y}] + E[\hat{y}] - \hat{y})] \\
&= E[f(x)^2 - f(x)E[\hat{y}] + f(x)E[\hat{y}] - f(x)\hat{y} \\
&\quad - E[\hat{y}]f(x) + E[\hat{y}]^2 - E[\hat{y}]^2 + E[\hat{y}]\hat{y} \\
&\quad + E[\hat{y}]f(x) - E[\hat{y}]^2 + E[\hat{y}]^2 - E[\hat{y}]\hat{y} \\
&\quad - \hat{y}f(x) + \hat{y}E[\hat{y}] - \hat{y}E[\hat{y}] + \hat{y}^2] \\
&= E[\hat{y}^2 - 2E[\hat{y}]\hat{y} + E[\hat{y}]^2] \\
&\quad + E[E[\hat{y}]^2 - 2E[\hat{y}]f(x) + f(x)^2] \\
&\quad + E[f(x)E[\hat{y}] - f(x)\hat{y} - E[\hat{y}]^2 + E[\hat{y}]\hat{y}] \\
&\quad + E[\hat{y}]f(x) - E[\hat{y}]^2 - \hat{y}f(x) + \hat{y}E[\hat{y}]] \\
&= \text{Variance} + \text{Bias}^2 \\
&\quad f(x)E[\hat{y}] - f(x)E[\hat{y}] + E[\hat{y}]^2 - E[\hat{y}]^2 \\
&\quad + f(x)E[\hat{y}] - f(x)E[\hat{y}] + E[\hat{y}]^2 - E[\hat{y}]^2 \\
&= \text{Variance} + \text{Bias}^2
\end{aligned}
$$



*实际上应该是理想模型的预测值，但是理想模型不可能知道，所以以实际值代表
http://scott.fortmann-roe.com/docs/BiasVariance.html

偏题：向量投影

向量b在由基向量a_1, a_2所张成(span)的向量空间（平面）s之外，b在s上的投影p可以用a_1, a_2的线性组合来表示，即



$$p = a_1x_1 + a_2x_2$$

以矩阵的形式表达：

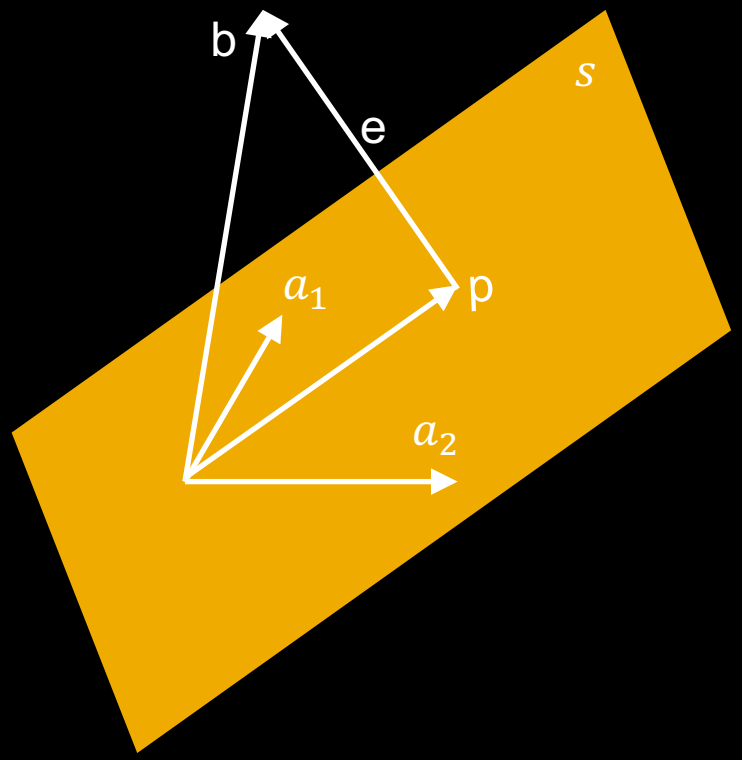
$$p = \mathrm{A}x$$

其中：$\mathrm{A} = [a_1 \quad a_2], \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$

$\because e = b - p$ 且 $e \perp a_1, e \perp a_2$

$$\therefore \begin{cases} a_1^T \cdot e = 0 \\ a_2^T \cdot e = 0 \end{cases} \Rightarrow \begin{cases} a_1^T \cdot (b - \mathrm{A}x) = 0 \\ a_2^T \cdot (b - \mathrm{A}x) = 0 \end{cases}.$$

偏题：向量投影



$$\begin{cases} a_1^T \cdot e = 0 \\ a_2^T \cdot e = 0 \end{cases} \Rightarrow \begin{cases} a_1^T \cdot (b - \mathrm{A}x) = 0 \\ a_2^T \cdot (b - \mathrm{A}x) = 0 \end{cases}.$$

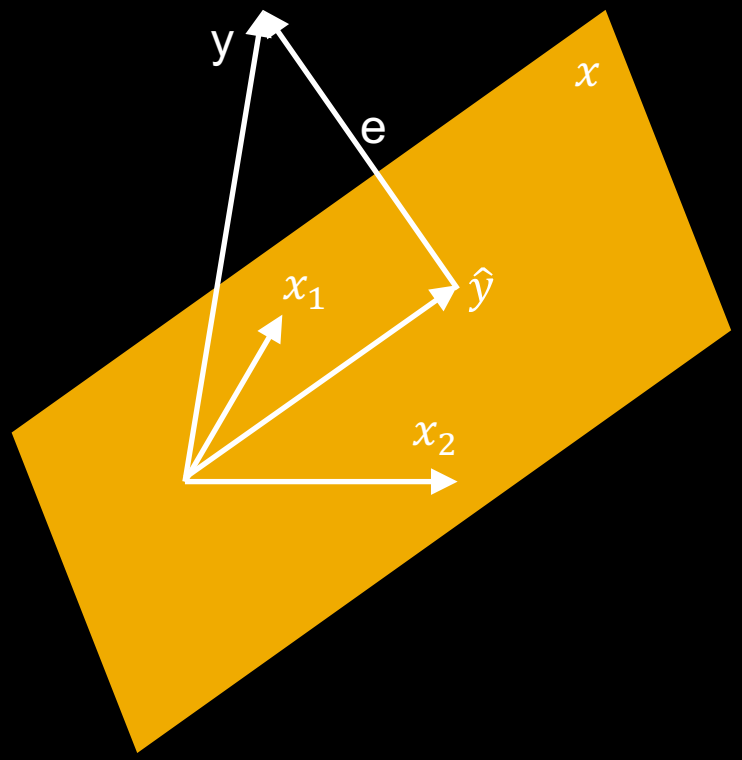
即：$\begin{bmatrix} a_1^T \\ a_2^T \end{bmatrix} (b - \mathrm{A}x) = \vec{0}$

以矩阵形式表达：$A^T\,(b - \mathrm{A}x) = \vec{0}$

整理：$A^T Ax = A^T b$

$$x = (A^T A)^{-1} A^T b$$

类似：Normal Equation



y在x所张成的线性空间以外，几乎不存在θ能满足：$x\theta = \mathrm{y}$，但是，能找到θ，使得$x\theta = \hat{y}$，即y在平面的投影上（平面上最接近y 的点）：

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

Normal Equation & Design Matrix

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

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad Y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

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

Pseudo Inverse:

Theta = np.linalg.pinv($(X^T X)^{-1} X^T y$). When $X^T X$ is not invertible

Regularization (Normal Equation)

如果 m<n , $X^T X$ 不可逆。然而，如果$\lambda > 0$

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

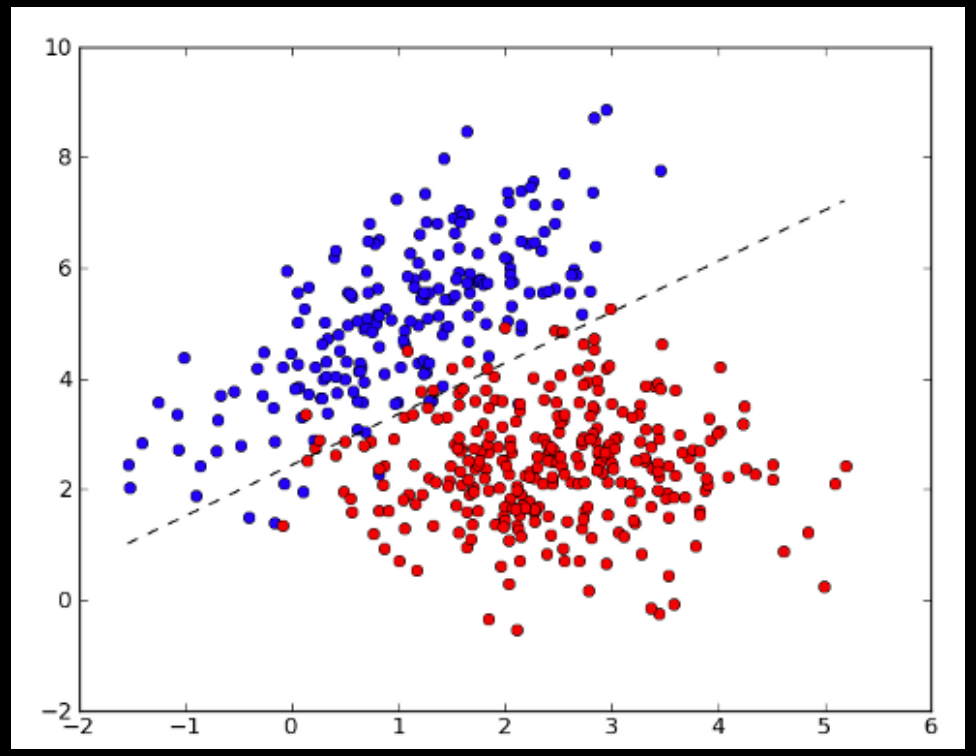
其中： L= $\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$

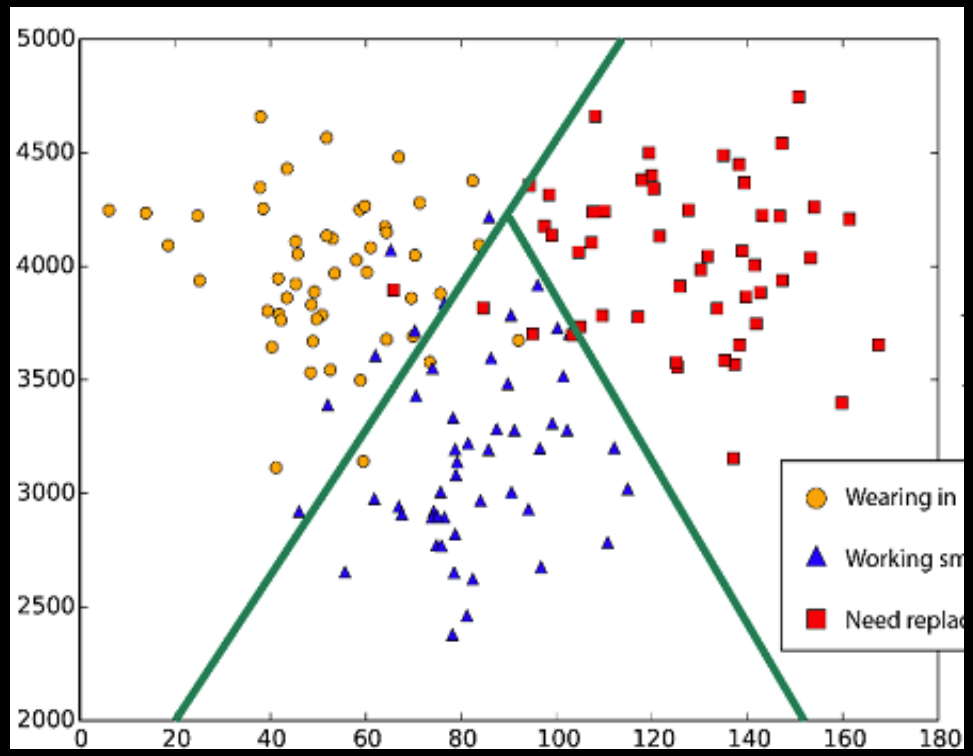
e.g. if n=2, then L= $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Gradient Descent vs Normal Equation

Gradient Descent	Normal Equation
需要选择 α	无需选择 α
需要迭代很多Iterations	无需迭代Iterations
无需计算逆矩阵$(X^T X)^{-1}$	需要计算逆矩阵$(X^T X)^{-1}$
当n很大时，也能很好的工作	当n很大时，计算比较费时

Logistic ~~Regression~~ (Binary / Multiclass)





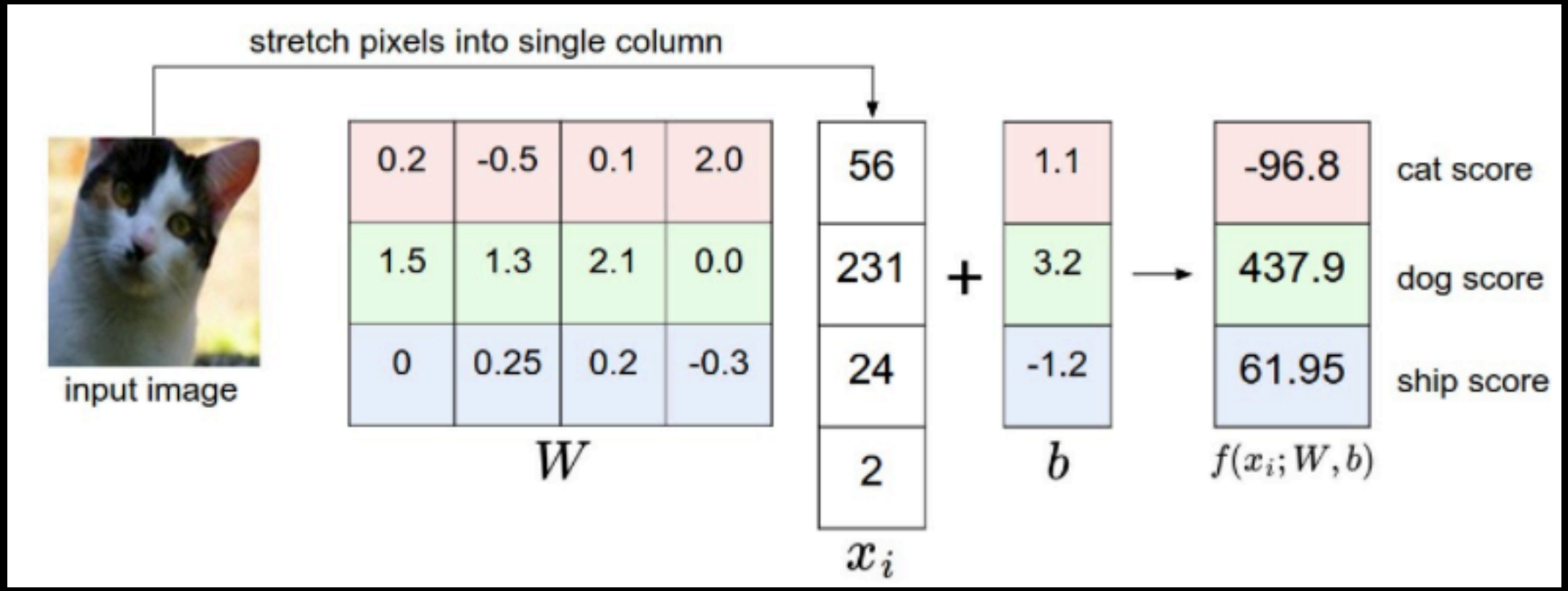
Logistic Regression : one-hot encoding

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Logistic Regression : Linear Classifier

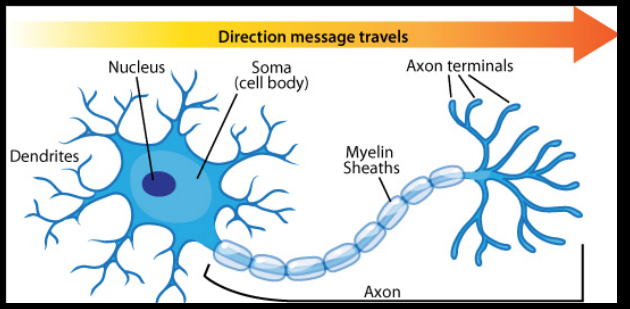


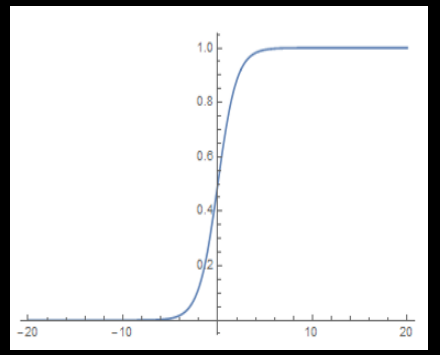
Source: CS231n

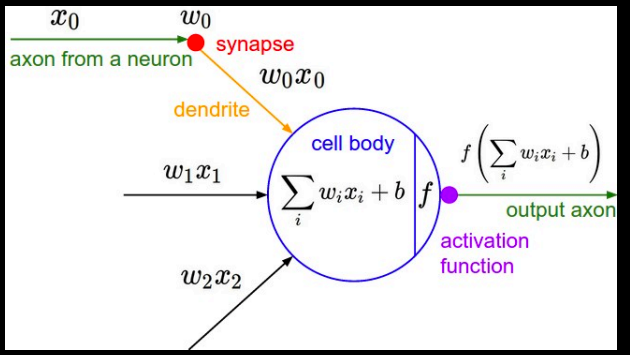
An example of mapping an image to class scores. For the sake of visualization, we assume the image only has 4 pixels (4 monochrome pixels, we are not considering color channels in this example for brevity), and that we have 3 classes (red (cat), green (dog), blue (ship) class). (Clarification: in particular, the colors here simply indicate 3 classes and are not related to the RGB channels.) We stretch the image pixels into a column and perform matrix multiplication to get the scores for each class. Note that this particular set of weights W is not good at all: the weights assign our cat image a very low cat score. In particular, this set of weights seems convinced that it's looking at a dog.

Binary Logistic Regression Model

- $Y \in \{0, 1\}$
- 我们希望 $0 \leq h_\theta(\mathrm{x}) \leq 1$，以表示分类0/1的概率
- 线性回归：$h_\theta(\mathrm{x}) = \theta^T \cdot x$
- 逻辑回归：$h_\theta(\mathrm{x}) = g(\theta^T \cdot x)$.　　$Sigmoid$函数：$g(z) = \frac{1}{1+e^{-z}}$
- $h_\theta(\mathrm{x}) = \frac{1}{1+e^{-\theta^T x}}$
- NN最常用的3个激活函数之一
- 从神经科学看，中央区类似神经元的兴奋态，两侧区类似抑制态。







Sigmoid Function Derivative

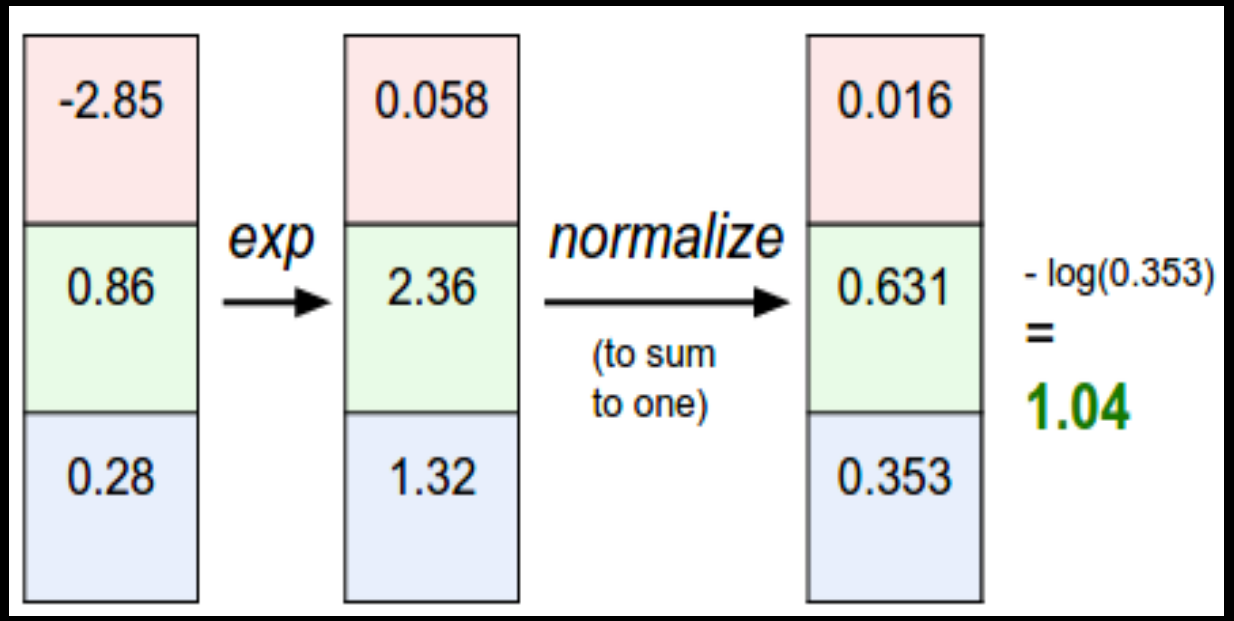
- $g(z) = \frac{1}{1+e^{-z}}$

- $g'(z) = \frac{d}{dz}\frac{1}{1+e^{-z}} = \frac{1}{(1+e^{-z})^2}(e^{-z})$

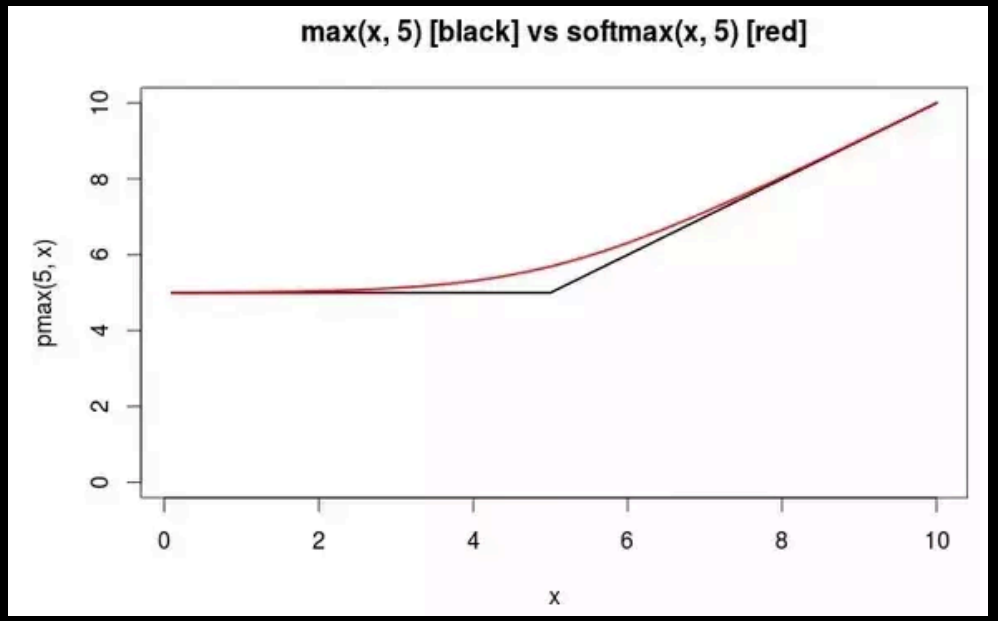
$$= \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= g(z)\big(1 - g(z)\big).$$

Logistic Regression : Softmax Function

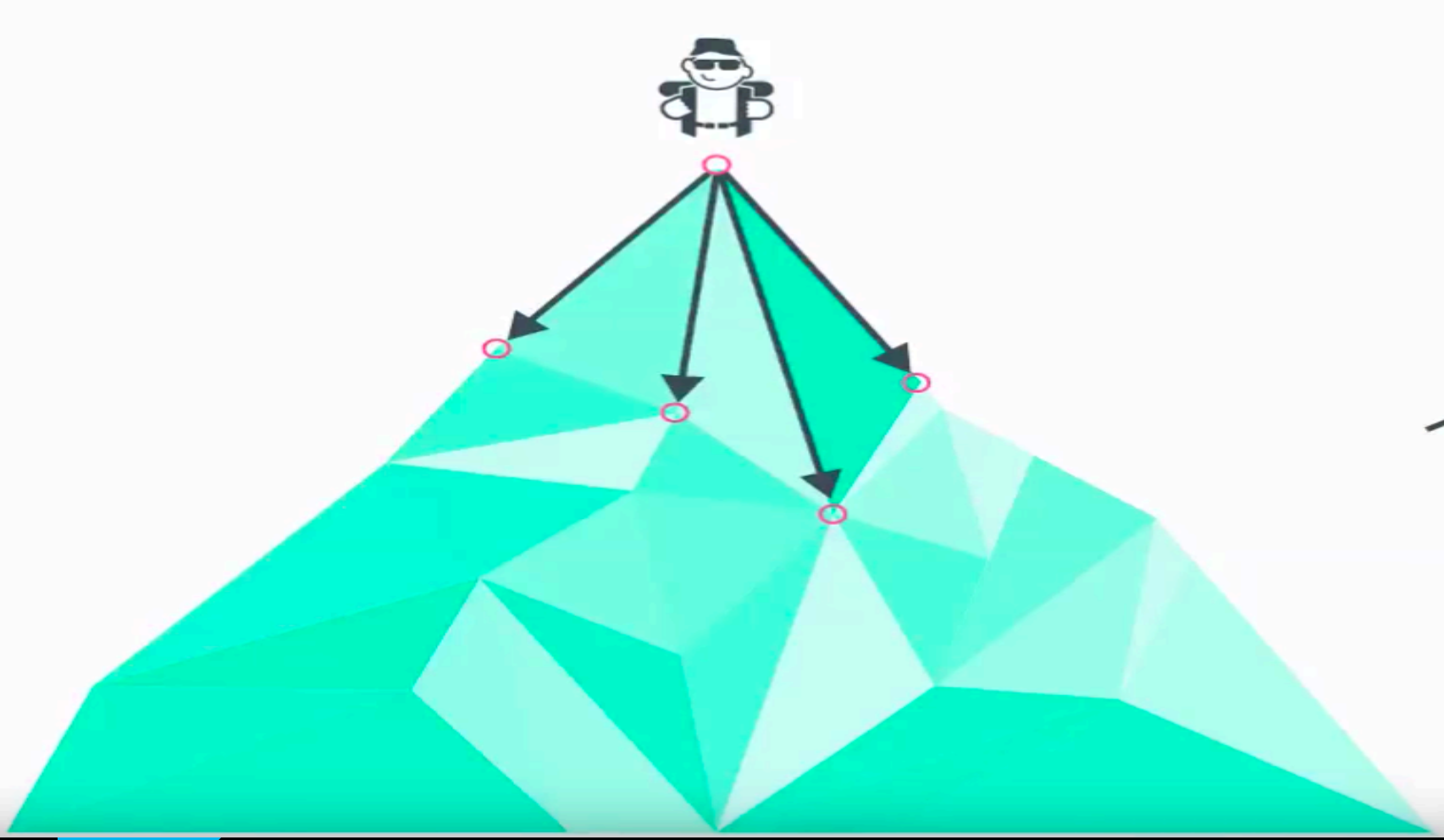


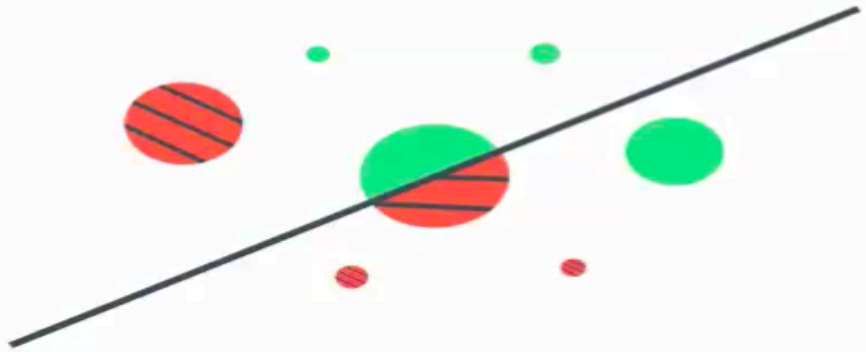
Source: CS231n



Hardmax v.s. Softmax

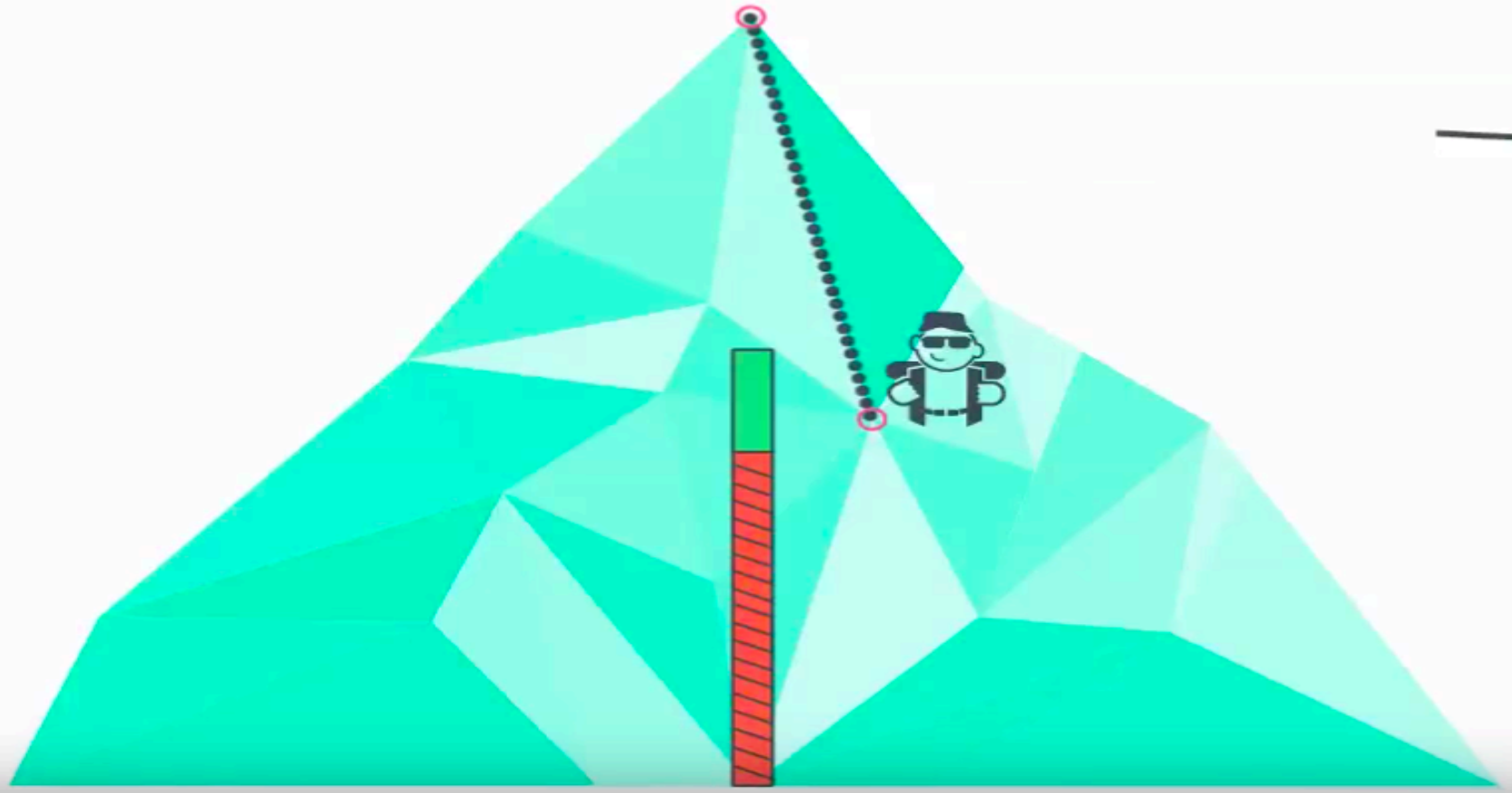
Logistic Regression Gradient Descent





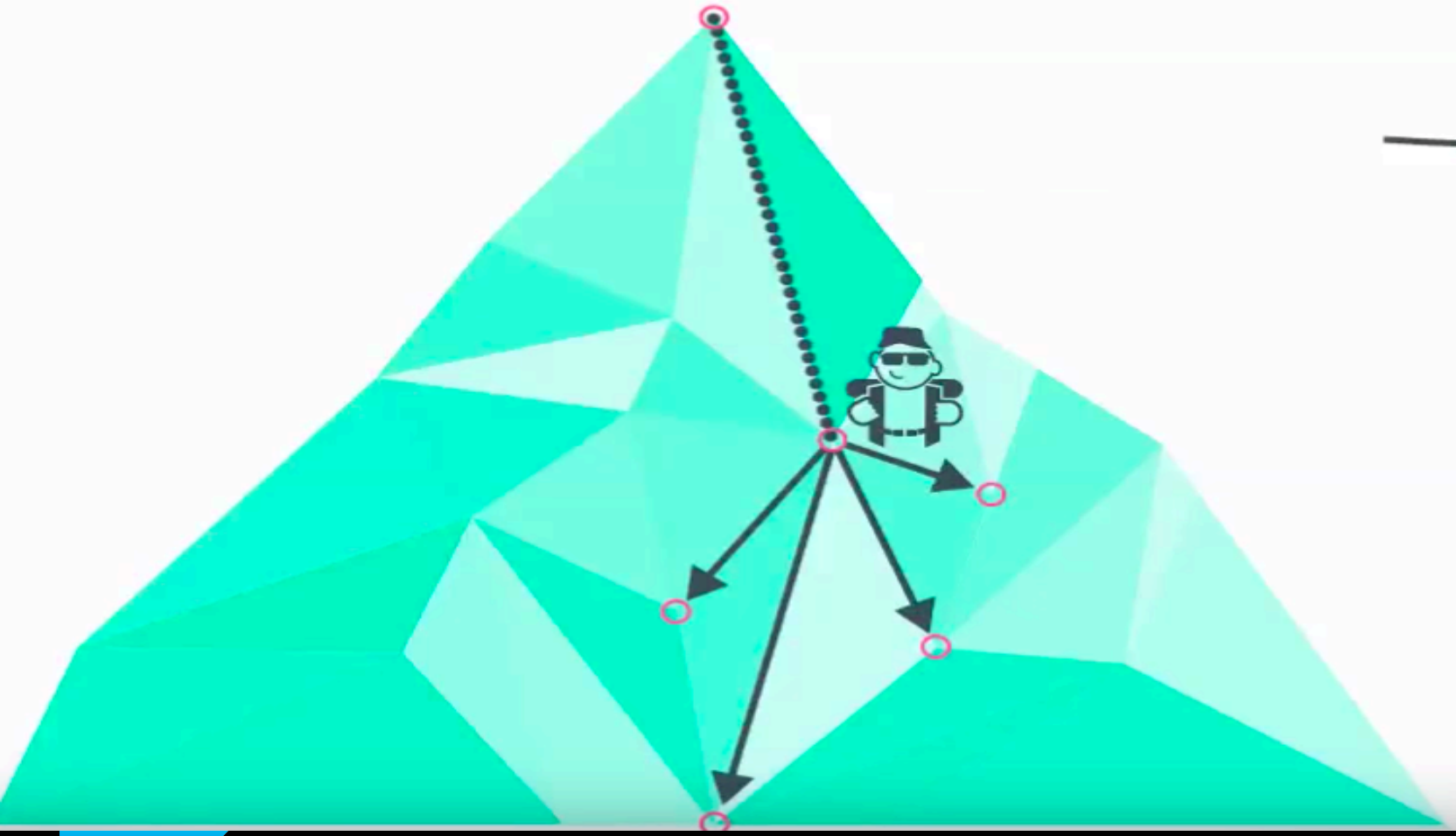
Source: Udacity

Logistic Regression Gradient Descent



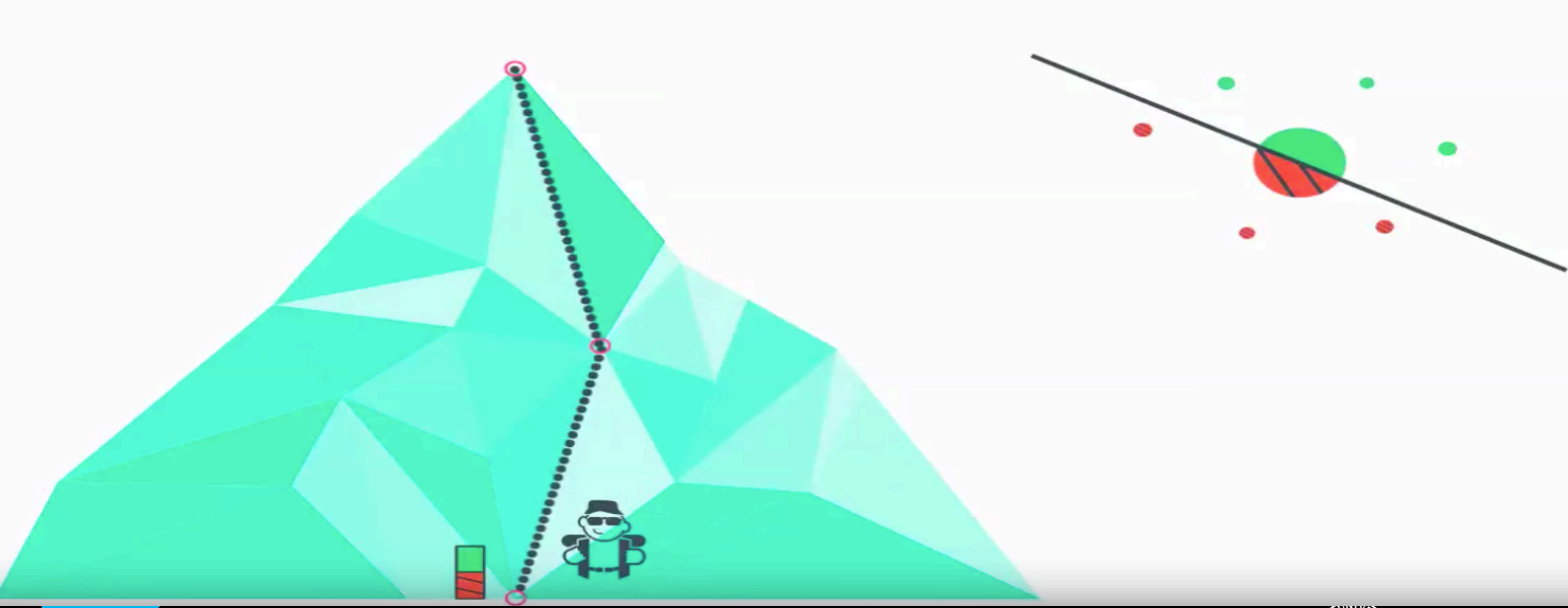
Source: Udacity

Logistic Regression Gradient Descent



Source: Udacity

Logistic Regression Gradient Descent



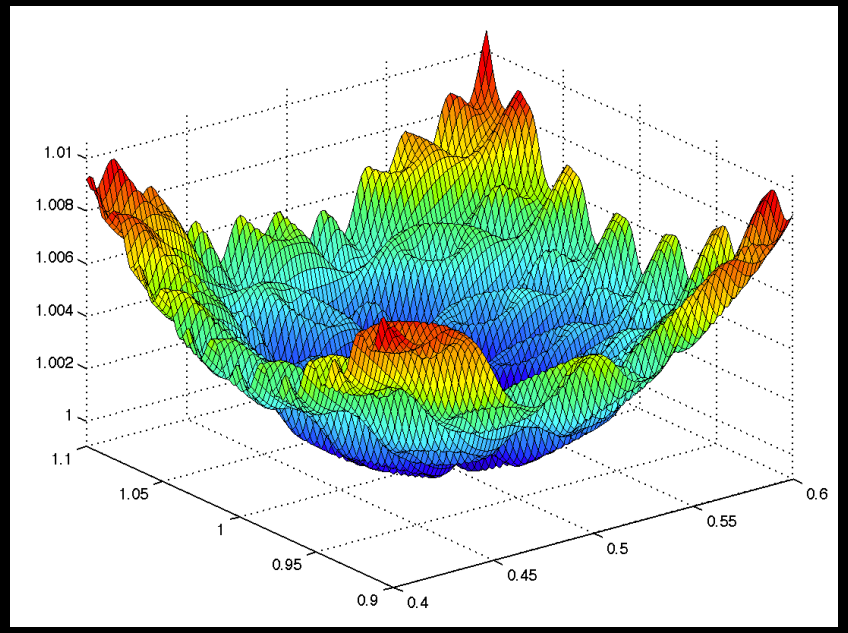
Source: Udacity

Logistic Regression Cost Function

MSE is non-convex

- 对于*Sigmoid 函数，以MSE作为代价函数会导致 non-convex*

$$\text{Cost} = \frac{1}{2}\,(h(\theta) - y)^2$$

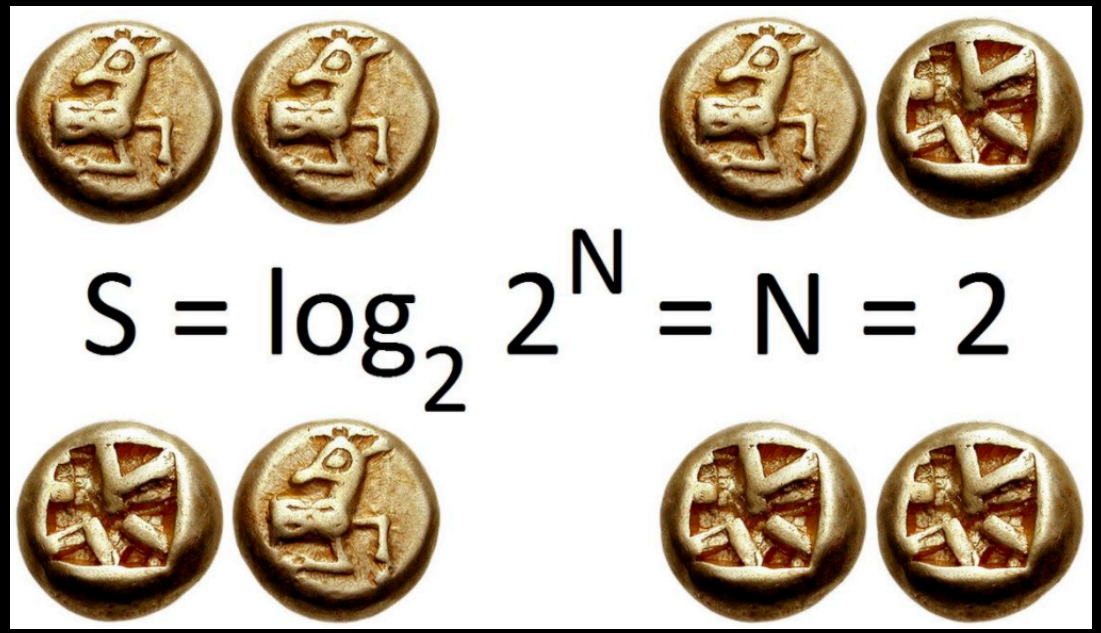


$$h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Source: Udacity

热力学与信息论中的熵（Entropy）

混乱程度与不群定性的度量



Entropy is expected value of information contained in each message.

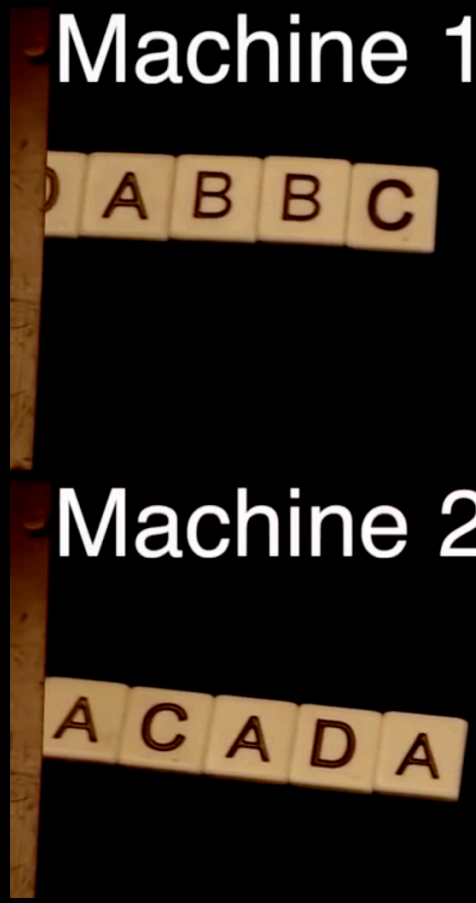
信息论中，系统被建模为发送者、渠道和接受者，发送者生成消息，并通过渠道传输，渠道可能以某种发生对消息进行了改变，例如，混入噪声，接收方试图推理出的消息原始内容。

信息是可能的消息或事件概率分布的负对数。每个事件的信息量所形成的随机变量的期望值就是熵，单位是Bit。

$$S = k \log W$$

K=1.3806505(24) × 10^-23 J/K ；W is number of microstates

香农熵（Shannon Entropy）



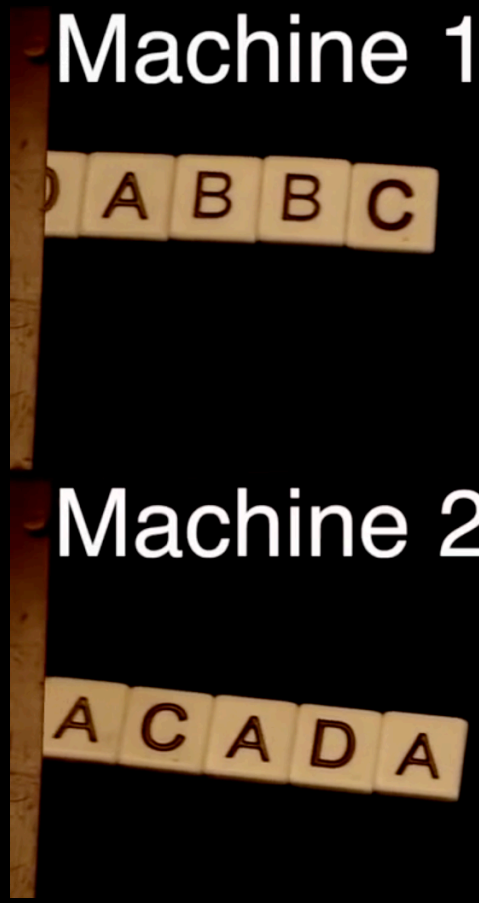
P(A) = 0.25
P(B) = 0.25
P(C) = 0.25
P(D) = 0.25

哪台机器产生更多的信息？

P(A) = 0.50
P(B) = 0.125
P(C) = 0.125
P(D) = 0.25

Source: KhanAcademy.org

香农聪明的换了一个问法



P(A) = 0.25
P(B) = 0.25
P(C) = 0.25
P(D) = 0.25

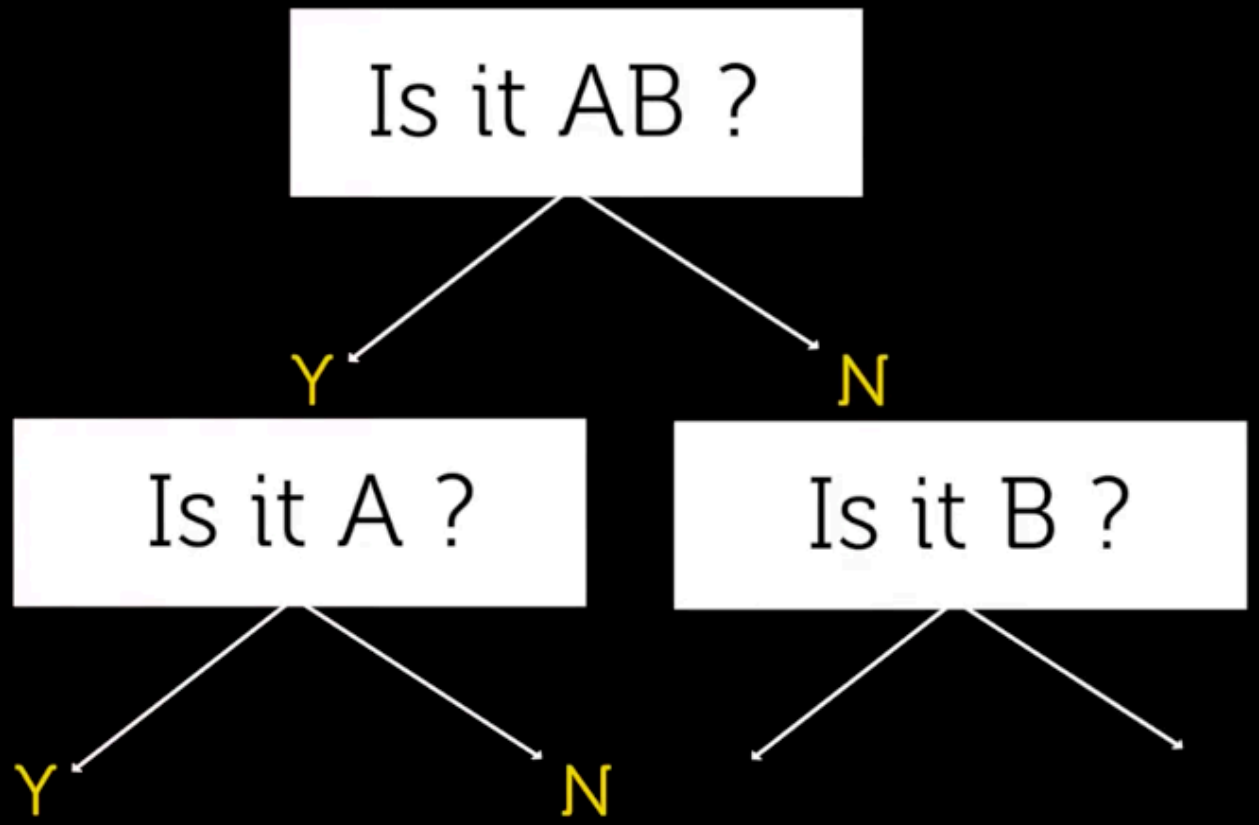
对于每台机器而言，你要问的最少的Yes / No问题是几个？

P(A) = 0.50
P(B) = 0.125
P(C) = 0.125
P(D) = 0.25

Source: KhanAcademy.org

第一台机器

?

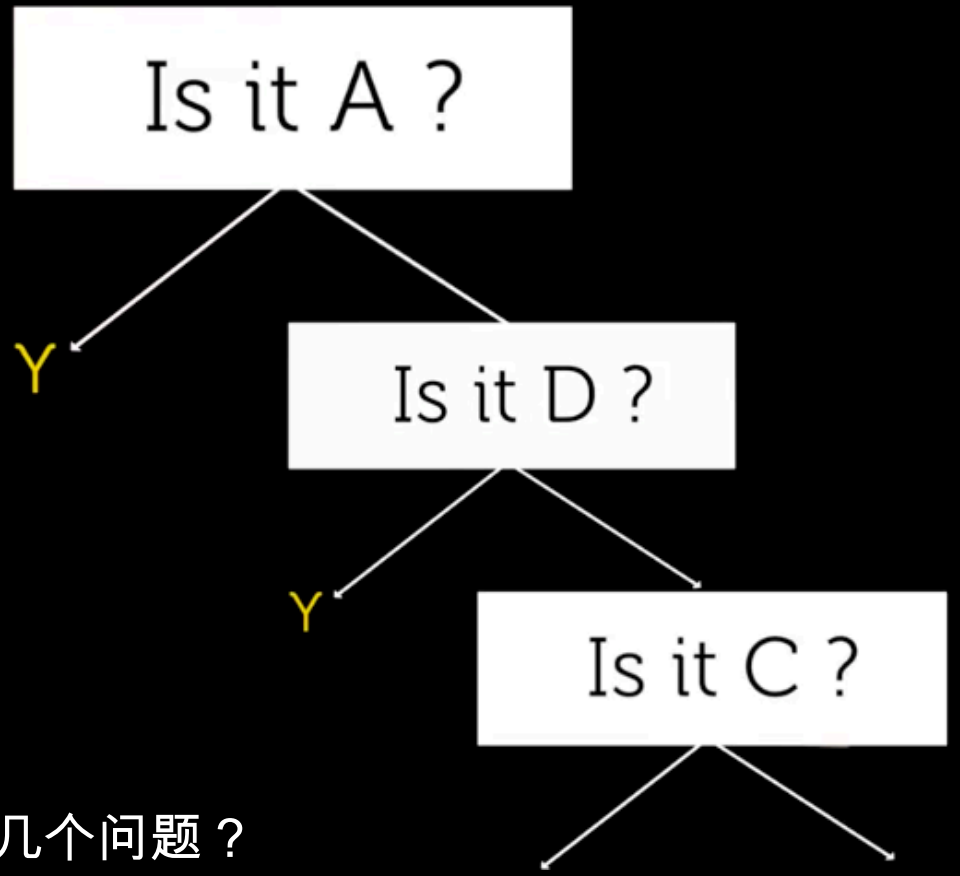


需要问两个问题

Source: KhanAcademy.org

第二台机器呢？

P(A) = 0.50
P(B) = 0.125
P(C) = 0.125
P(D) = 0.25



为了确定一个字符，平均需要问几个问题？

E[问题数量]=$P_A \times 1 + P_D \times 2 + P_B \times 3 + P_C \times 3 = \mathbf{1.75}$

Source: KhanAcademy.org

$$H = \sum_{i=1}^{n} P_i \times \#\text{问题}$$

第二台机器呢？

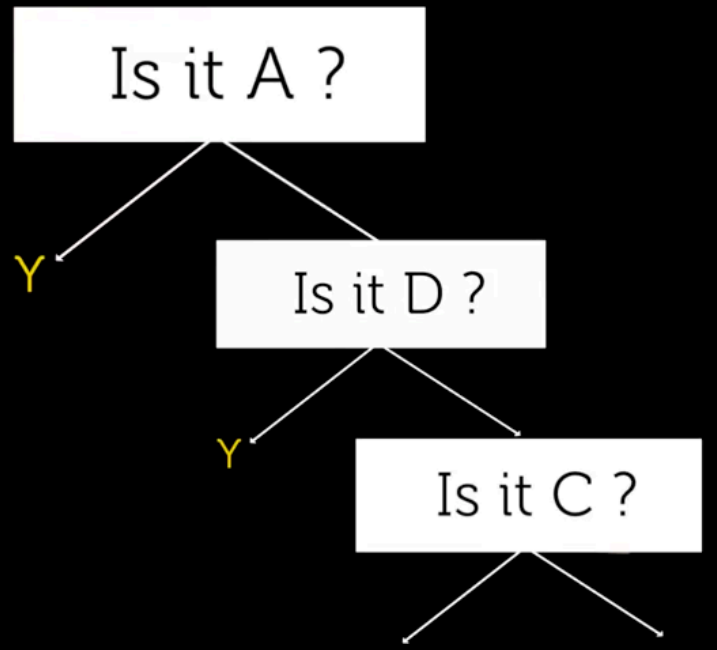
E[问题数量]=$P_A \times 1 + P_D \times 2 + P_B \times 3 + P_C \times 3 = \mathbf{1.75}$

$$\mathrm{H} = \sum_{i=1}^{n} P_i \times \#\text{问题}$$

#问题由*节点在树中的位置决定，即由概率决定*

$$\#\text{问题} = log_2\left(\frac{1}{P_i}\right)$$

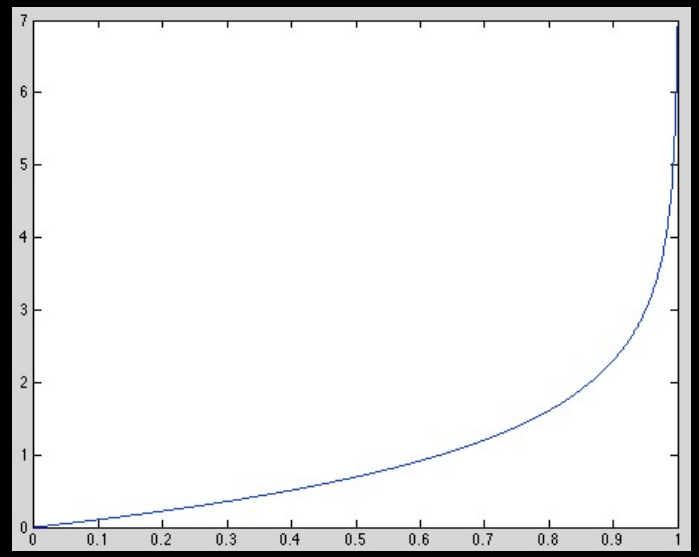
$$\mathrm{H} = \sum_{i=1}^{n} P_i \times log_2\left(\frac{1}{P_i}\right) = -\sum_{i=1}^{n} P_i \times log_2(P_i)$$



Source: KhanAcademy.org

Binary Logistic Regression Cost Function

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



可以将上式合并，因为：当y=1时，第二项 $(1 - y)log(1 - h_\theta(x)) = 0$，*不必考虑，同样的，当y=0时，第一项也等于0，不必考虑。*

$$Cost(h_\theta(x), y) = -ylog(h_\theta(x)) - (1 - y)log(1 - h_\theta(x))$$

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

Binary Logistic Regression Gradient Descent

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

我们从一个数据样本开始，进行随机梯度下降：

$$\frac{\partial}{\partial\theta_j}\mathcal{J}(\theta) = \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j} g(\theta^T x)$$

$$= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial\theta_j}\theta^T x$$

$$= \left(y\left(1-g(\theta^T x)\right) - (1-y)g(\theta^T x)\right)x_j$$

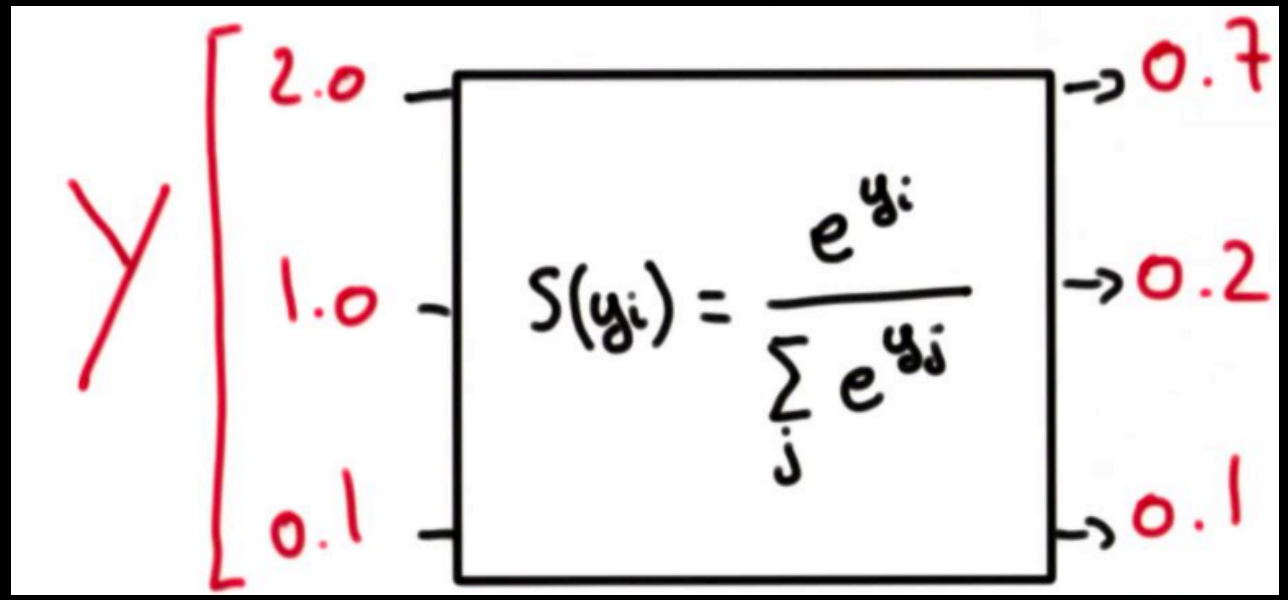
$$= (y - h_\theta(x))\, x_j$$

$$\theta_j := \theta_j + \alpha\,(y - h_\theta(x^{(i)}))x_j^{(i)}$$

注意：梯度下降公式虽然形式上与线性回归完全相同，但是
$h_\theta(x^{(i)})$ *已变为Sigmoid 函数！！*

Logistic Regression / Softmax Function

对于多分类的情况：



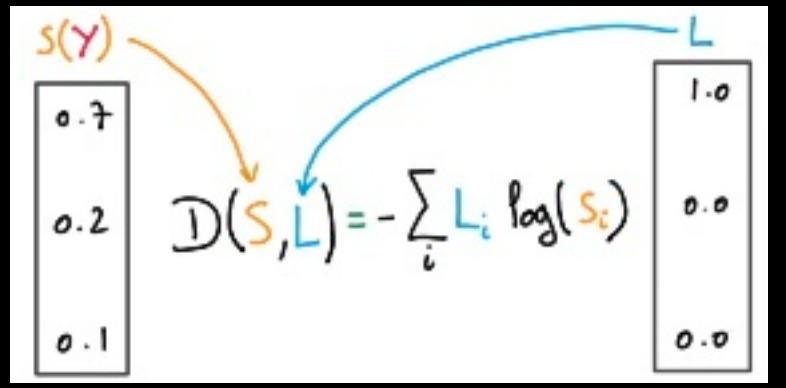
y_i	e^{y_i}	$e^{y_i} / \sum_i e^{y_j}$
2	7.389056099	0.66
1	2.718281828	0.24
0.1	1.105170918	0.10
$\sum$	11.21250885	1

Source: Udacity

Cost Function of Softmax

Cross-entropy （CE,交叉熵）V.S. MSE（均方差）

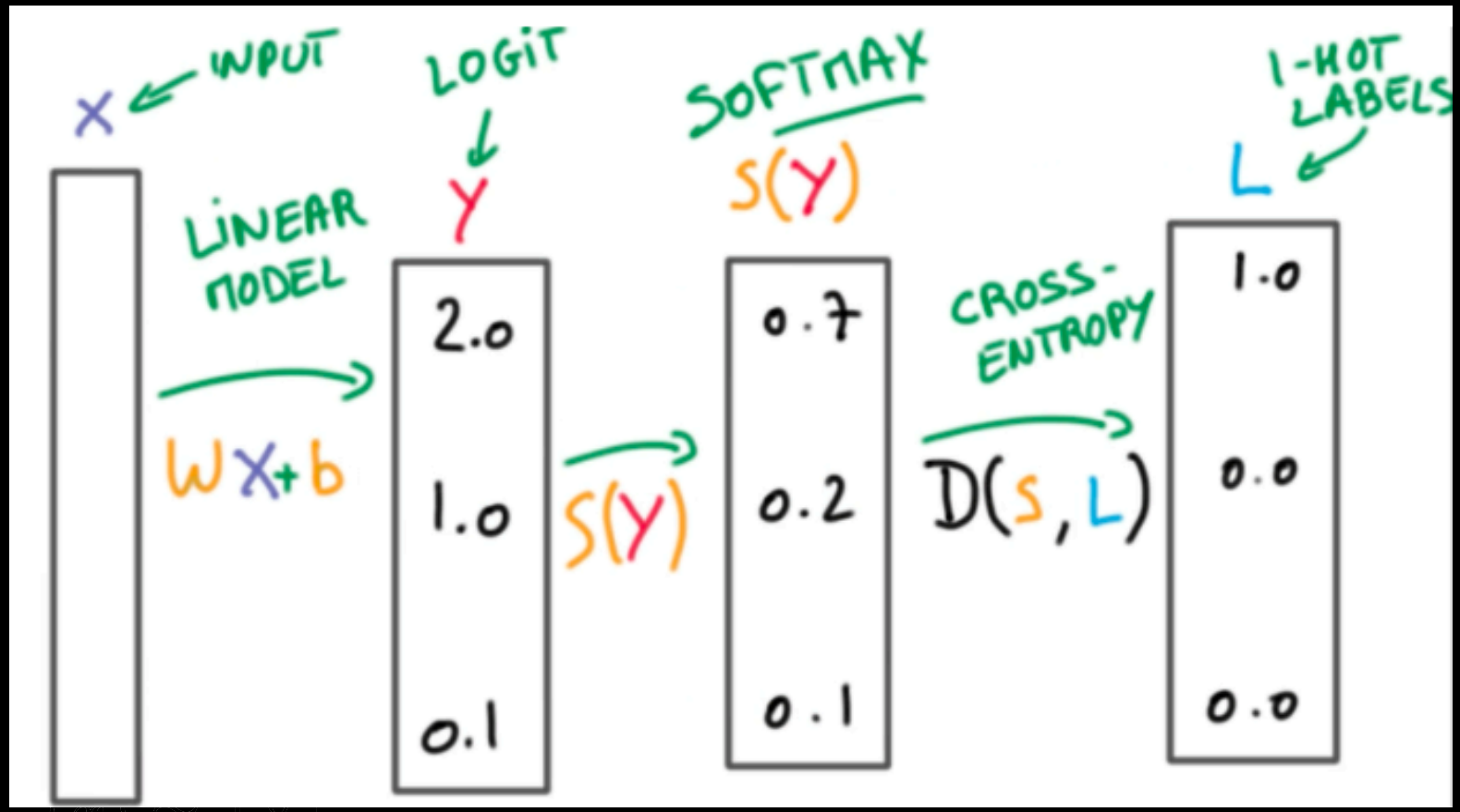
- MSE *更适合目标变量是连续值，实际值在预测值两边呈Gaussian分布的情况。*
- 如何度量分类模型的Cost？即如何度量预测的可能性(Likelihood, *S(Y)*)与One-Hot Vector标签(L)的距离？
- 问题再进一步：两个模型，预测出的Likelihood分别是0.51与0.99，都作出了正确的分类，预测标签都是“1”，两个模型一样好吗？
- 关键在于如何度量*S(Y)*与Label之间的距离。



Source: Udacity

- 对于分类问题，目标变量是离散值，对于错误分类的情况，要给予特别的惩罚，CE更合适。

Multinomial Logistic Classification



Source: Udacity

$$D(S(WX+b), L)$$

Thank you!

Contact information:

邬学宁（i025497)
Chief Data Scientist, SAP Silicon Valley Innovation Center
Address: No. 1001, Chenghui Road, Shanghai, 201023
Phone number: +8621-6108 5287
Email: x.wu@sap.com