Logistic Regression as a Neural Network

$$\hat{y} = \sigma\left(w^T x + b\right), \text{ where } \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\text{Given } \left\{\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}, \text{ want } \hat{y}^{(i)} \approx y^{(i)}$$

Loss(error) function:

Don't use this, non-convex

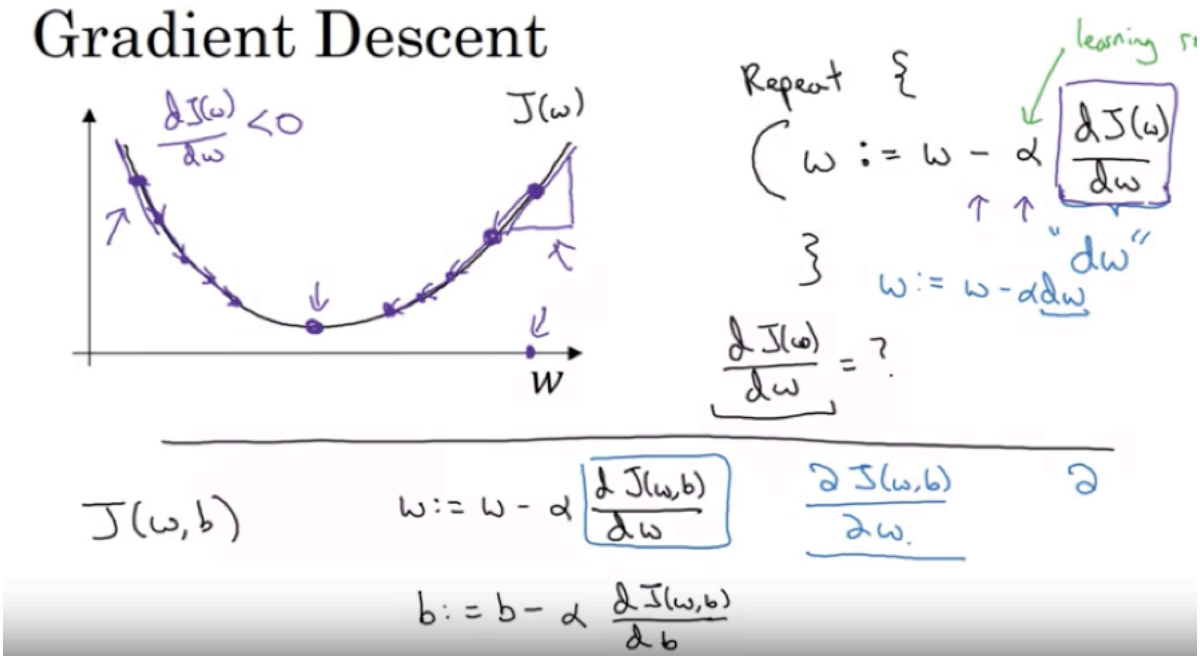
$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

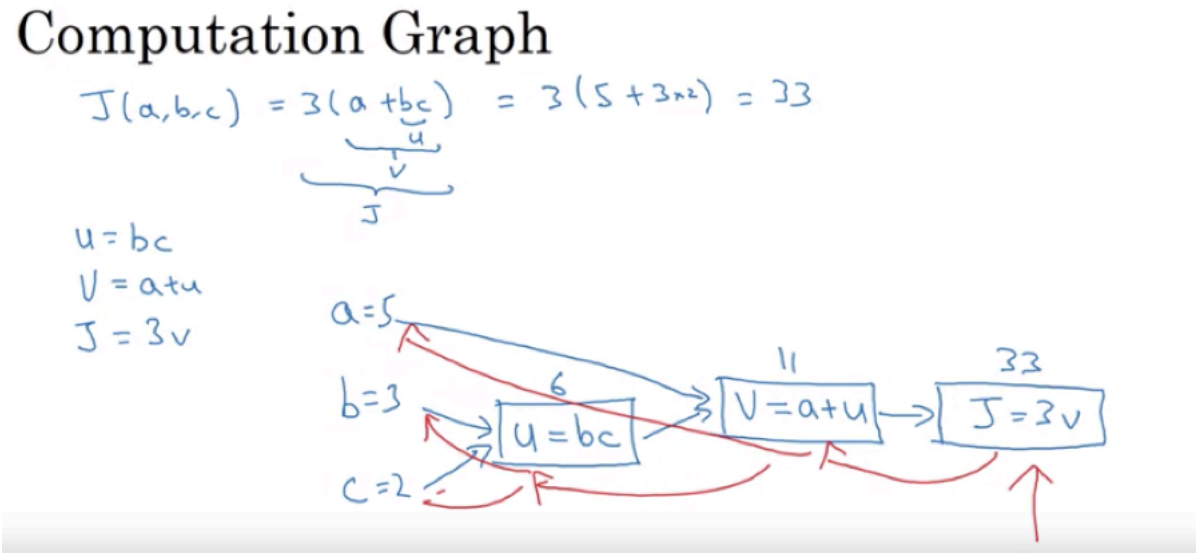
The loss function computes the error for a single training example; the cost function is the average of the loss functions of the entire training set.

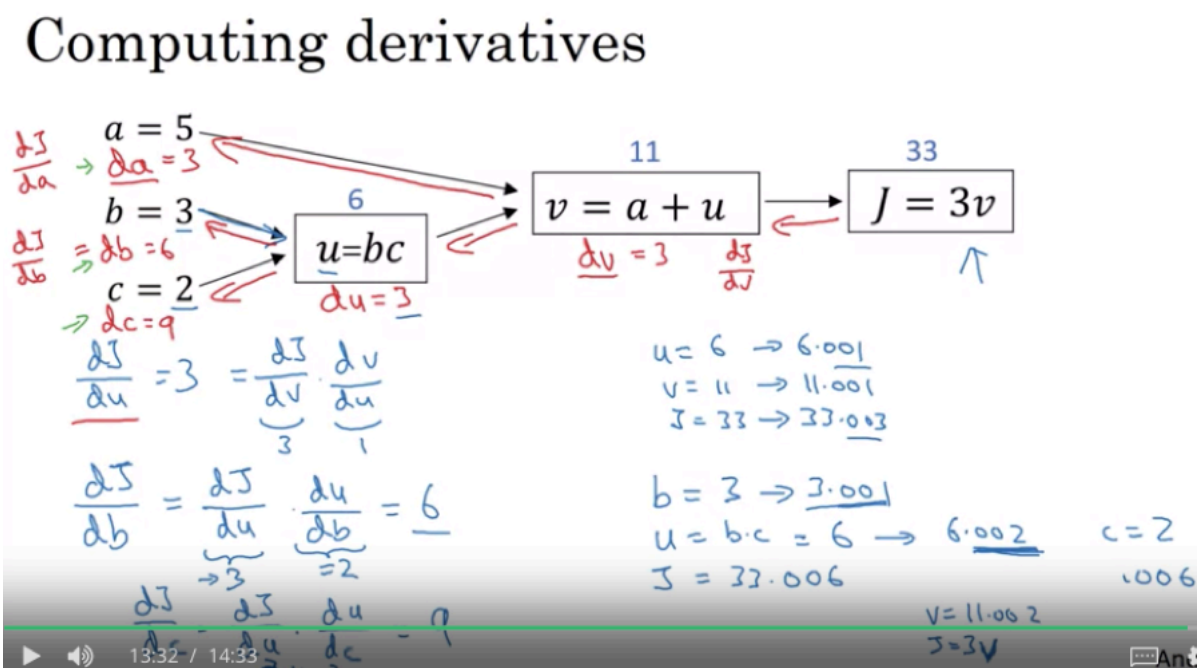
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

$$\text{Want to find } w, b \text{ that minimize } J(w,b)$$

a

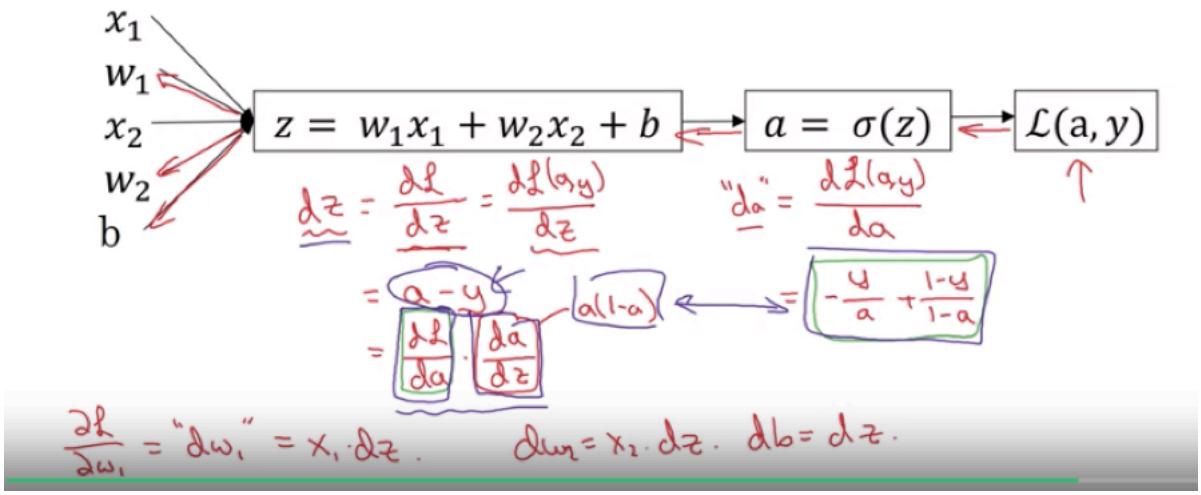






Gradient Desent

Logistic regression derivatives



Logistic regression on m examples

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

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

$(x^{(i)}, y^{(i)})$

$dw_1^{(i)}$, $dw_2^{(i)}$, $db^{(i)}$

$\frac{\partial}{\partial w_1} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial w_1} \mathcal{L}(a^{(i)}, y^{(i)})$

$dw_1^{(i)}$ — $(x^{(i)}, y^{(i)})$

Step 1 : $\frac{dL}{da}$

$L = -(y \times \log(a) + (1-y) \times \log(1-a))$

$\frac{dL}{da} = -y \times \frac{1}{a} - (1-y) \times \frac{1}{1-a} \times -1$

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}\left(1+\mathrm{e}^{-x}\right)^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}
$$

In the previous video, Andrew refers to $dz = a(1-a)$

Note that Andrew is using "dz" as a shorthand to refer to $\frac{da}{dz} = a(1-a)$.

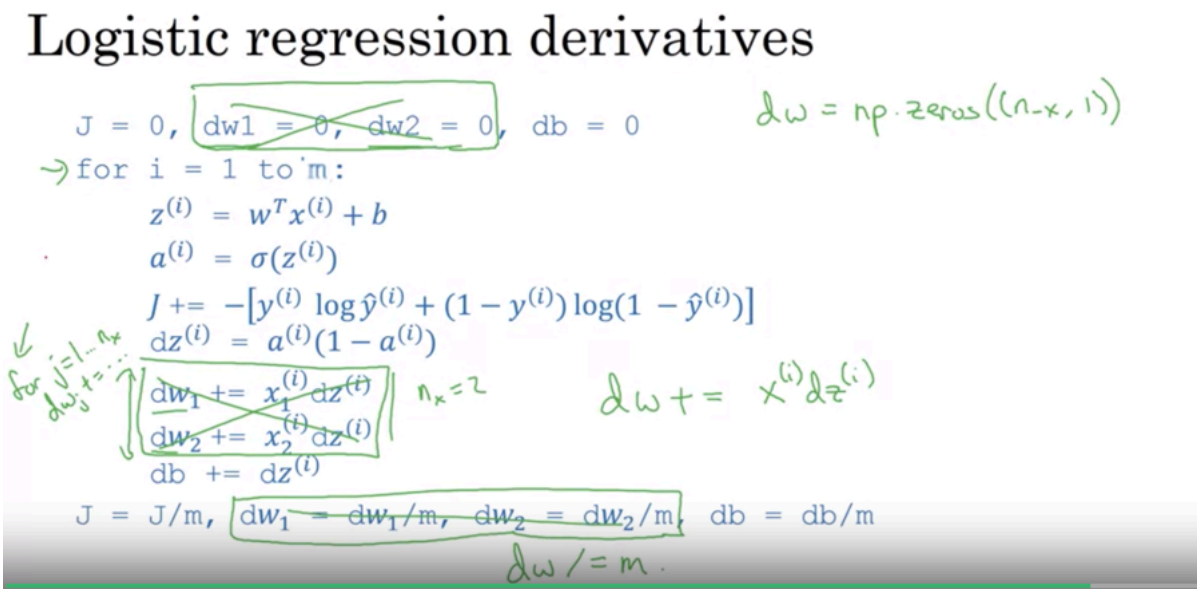
To clarify, earlier in this week's videos, Andrew used the name "dz" to refer to a different derivative: $\frac{dL}{dz} = a - y$.

Recall that the relationship between $\frac{dL}{dz}$ and $\frac{da}{dz}$ is:

$\frac{dL}{dz} = \frac{dL}{da} \times \frac{da}{dz}$

$\frac{dL}{dz} = \frac{a-y}{a(1-a)} \times a(1-a) = a - y$

Vectorization



Vectorizing Logistic Regression

$dz^{(1)} = a^{(1)} - y^{(1)}$ $dz^{(2)} = a^{(2)} - y^{(2)}$ ....

$dZ = [dz^{(1)}\ dz^{(2)} \dots dz^{(m)}]$ ←

$1 \times m$

$A = [a^{(1)} \dots a^{(m)}]$. $Y = [y^{(1)} \dots y^{(m)}]$

$\rightarrow dZ = A - Y = [a^{(1)} - y^{(1)}\ \ a^{(2)} - y^{(2)} \dots]$

$\rightarrow dw = 0$

$dw += x^{(1)} dz^{(1)}$

$dw += x^{(2)} dz^{(2)}$

$\vdots$

$dw /= m$

$db = 0$

$db += dz^{(1)}$

$db += dz^{(2)}$

$\vdots$

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

$db /= m.$

$$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)}$$

$$= \frac{1}{m} \text{np.sum}(dZ)$$

$$dw = \frac{1}{m} X dZ^T$$

$$= \frac{1}{m} \begin{bmatrix} x^{(1)} & \dots & x^{(m)} \\ 1 & & 1 \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$

$$= \frac{1}{m} \left[x^{(1)} dz^{(1)} + \dots + x^{(m)} dz^{(m)} \right]$$

$n \times 1$

Implementing Logistic Regression

$J = 0,\ dw_1 = 0,\ dw_2 = 0,\ db = 0$

for i = 1 to m:

$z^{(i)} = w^T x^{(i)} + b$ ←

$a^{(i)} = \sigma(z^{(i)})$ ←

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

$dz^{(i)} = a^{(i)} - y^{(i)}$ ←

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

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

$dw += x^{(i)} * dz^{(i)}$

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

$J = J/m,\ dw_1 = dw_1/m,\ dw_2 = dw_2/m$

$db = db/m$

$Z = w^T X + b$

$= \text{np.dot}(w.T, X) + b$

$A = \sigma(Z)$

$dZ = A - Y$

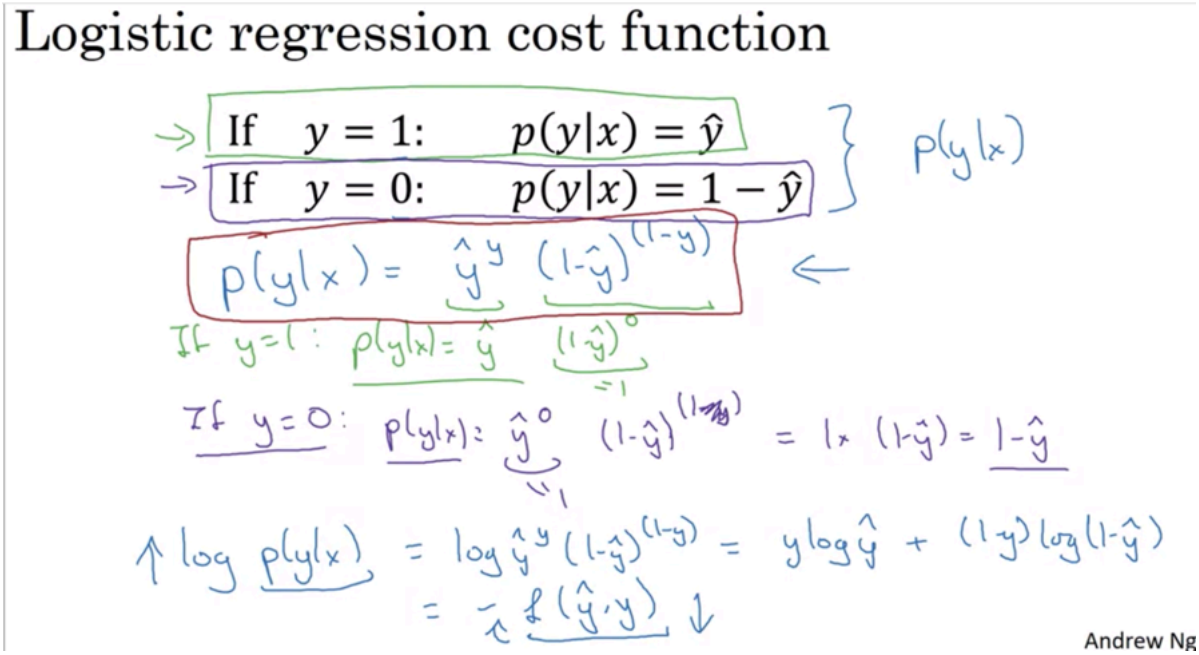
$dw = \frac{1}{m} X dZ^T$

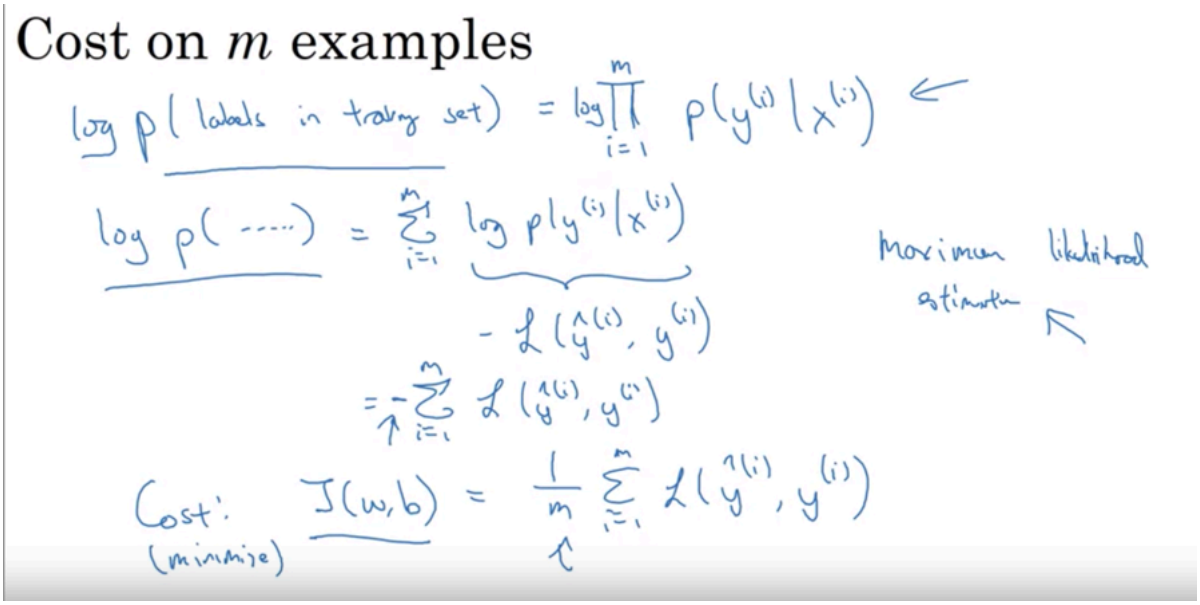
$db = \frac{1}{m} \text{np.sum}(dZ)$

$w := w - \alpha\, dw$

$b := b - \alpha\, db$

Andrew





Code

For convenience, you should now reshape images of shape (num_px, num_px, 3) in a numpy-array of shape (num_px $*$ num_px $*$ 3, 1). After this, our training (and test) dataset is a numpy-array where each column represents a flattened image. There should be m_train (respectively m_test) columns.

Exercise: Reshape the training and test data sets so that images of size (num_px, num_px, 3) are flattened into single vectors of shape (num_px $*$ num_px $*$ 3, 1).

A trick when you want to flatten a matrix X of shape (a,b,c,d) to a matrix X_flatten of shape (b∗c∗d, a) is to use:

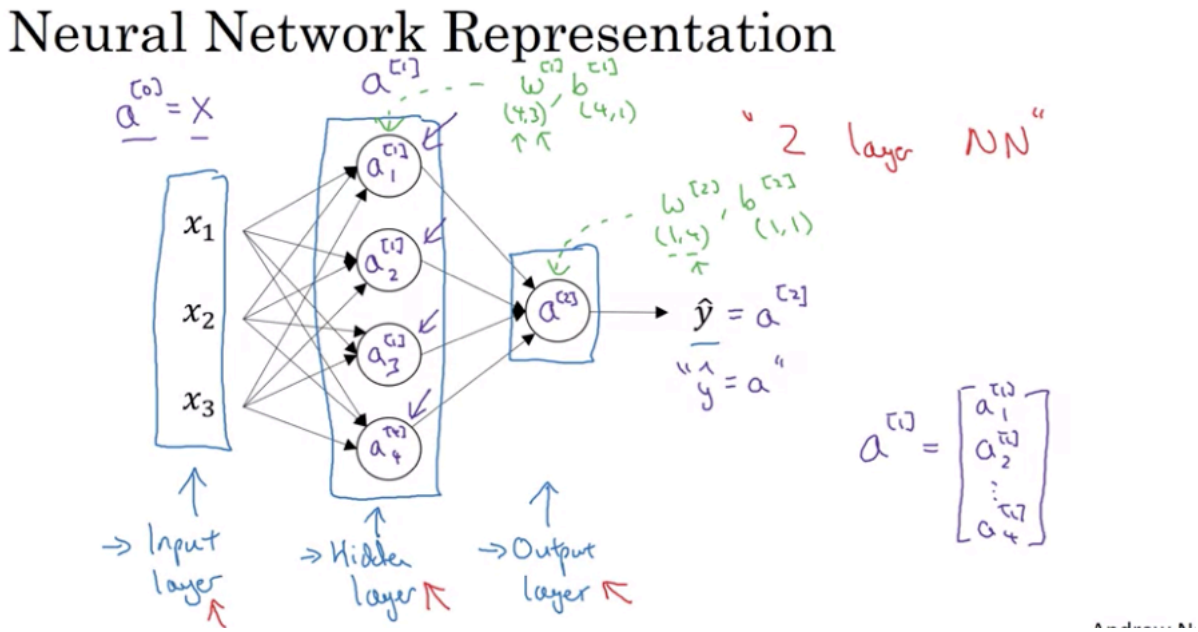
```
X_flatten = X.reshape(X.shape[0], -1).T      # X.T is the
transpose of X
```

To represent color images, the red, green and blue channels (RGB) must be specified for each pixel, and so the pixel value is actually a vector of three numbers ranging from 0 to 255.

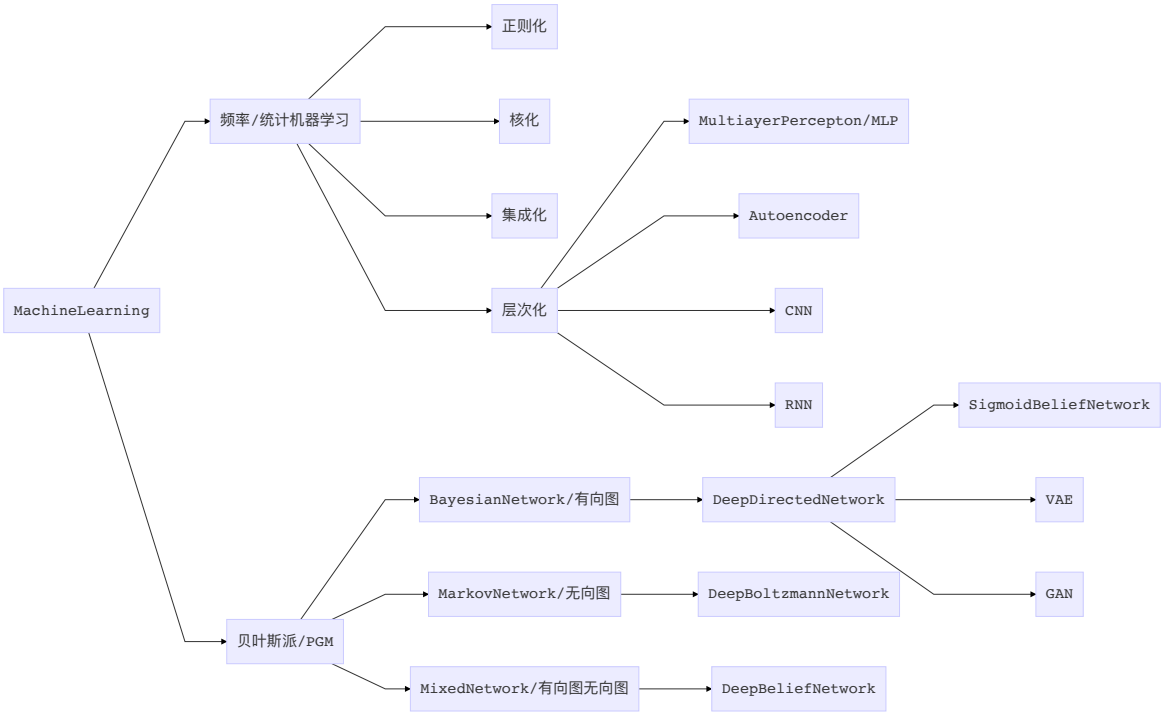
One common preprocessing step in machine learning is to center and standardize your dataset, meaning that you substract the mean of the whole numpy array from each example, and then divide each example by the standard deviation of the whole numpy array. But for picture datasets, it is simpler and more convenient and works almost as well to just divide every row of the dataset by 255 (the maximum value of a pixel channel).

Let's standardize our dataset.

Neural Network



基础



统计机器学习

- 正则化：Loss Function + regularizer
- 核化: Kernel SVM
- 集成化: AdaBoost, RandomForest
- 层次化: Neural Network/Deep Neural Network
    - MLP
    - Autoencoder
    - CNN
    - RNN

贝叶斯派

- BayesianNetwork ⇒ Deep Directed Network
    - Sigmoid Belief Network
    - Variational Autoencoder(VAE)

    - GAN
- Markov Network ⇒ Deep Boltzmann Network
- Mixed Netowork ⇒ Deep Belief Network

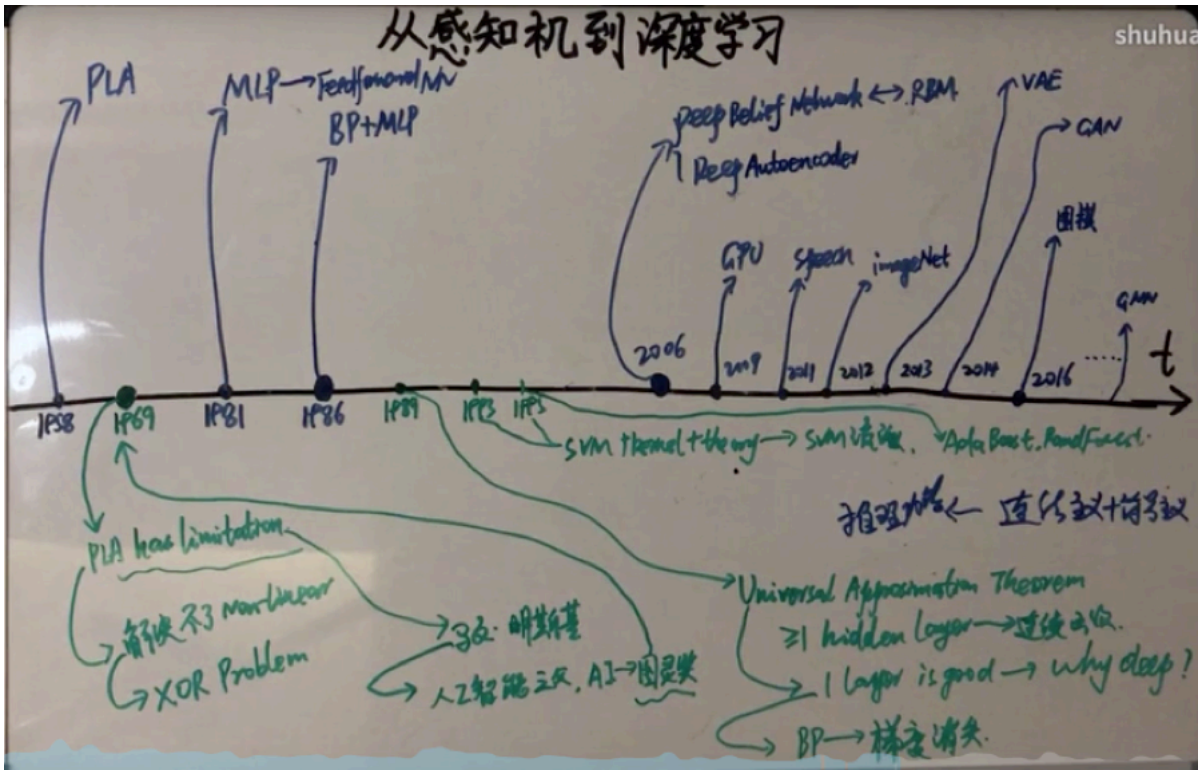
> Deep Generative Model

狭义的DeepLearning：Deep Neural Network

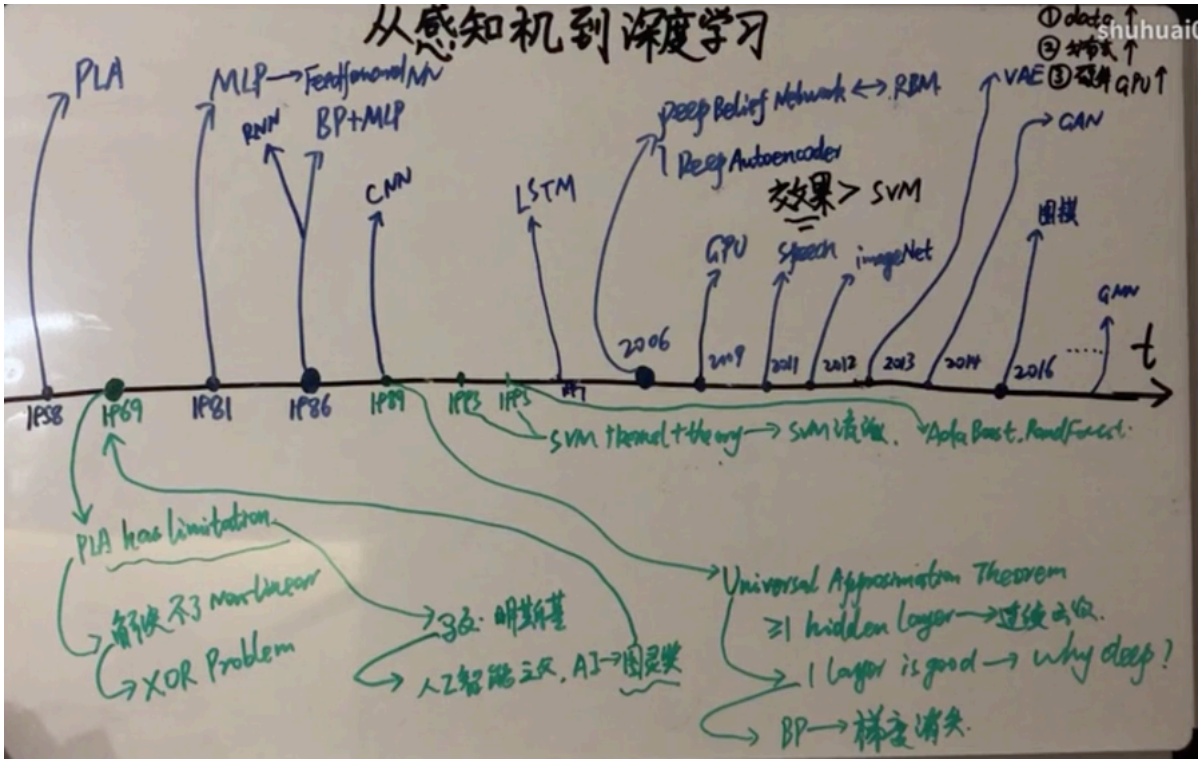
其实应该包括：

1. Deep Neural Network
2. Deep Generative Model（当层次非常多，推断非常困难）

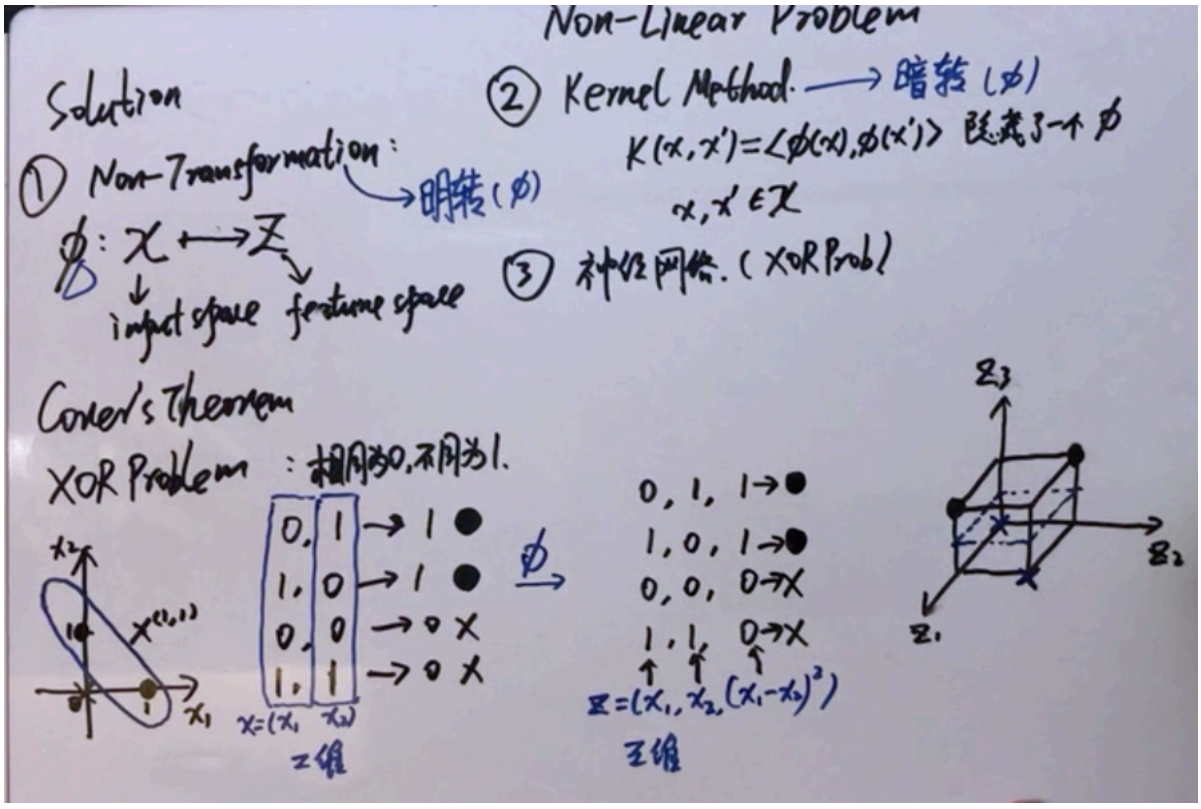
时间线



1. 深度学习的理论在2006年已经成型，直到现在，理论并没有根本性突破。
2. 为什么take off
    1. data
    2. 分布式
    3. 硬件 GPU
    4. 效果



Non-Linear Problem



Neural Network

$x, x' \in \mathcal{X}$

③ 神经网络. (XOR Prob)

XOR ⊕ 复合运算

OR ∨ AND ∧ NOT ¬ 基本运算

XOR → 相同为0, 不同为1

$x_1 \oplus x_2 = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$

x_1 XOR x_2 = {1,0 ① / 0,1 ②

= ① ∨ ②

= $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$

OR: 包含

1, 0 → 1

0, 1 → 1

1, 1 → 1

AND 多者

1, 1 → 1

NOT

0 → 1

$x, x' \in \mathcal{X}$

③ 神经网络. (XOR Prob)

XOR ⊕ 复合运算

OR ∨ AND ∧ NOT ¬ 基本运算

XOR → 相同为0, 不同为1

$x_1 \oplus x_2 = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$

PLA

有向无环图

神经网络

① ∧

② ∧

③ ∨

∨

1, 0

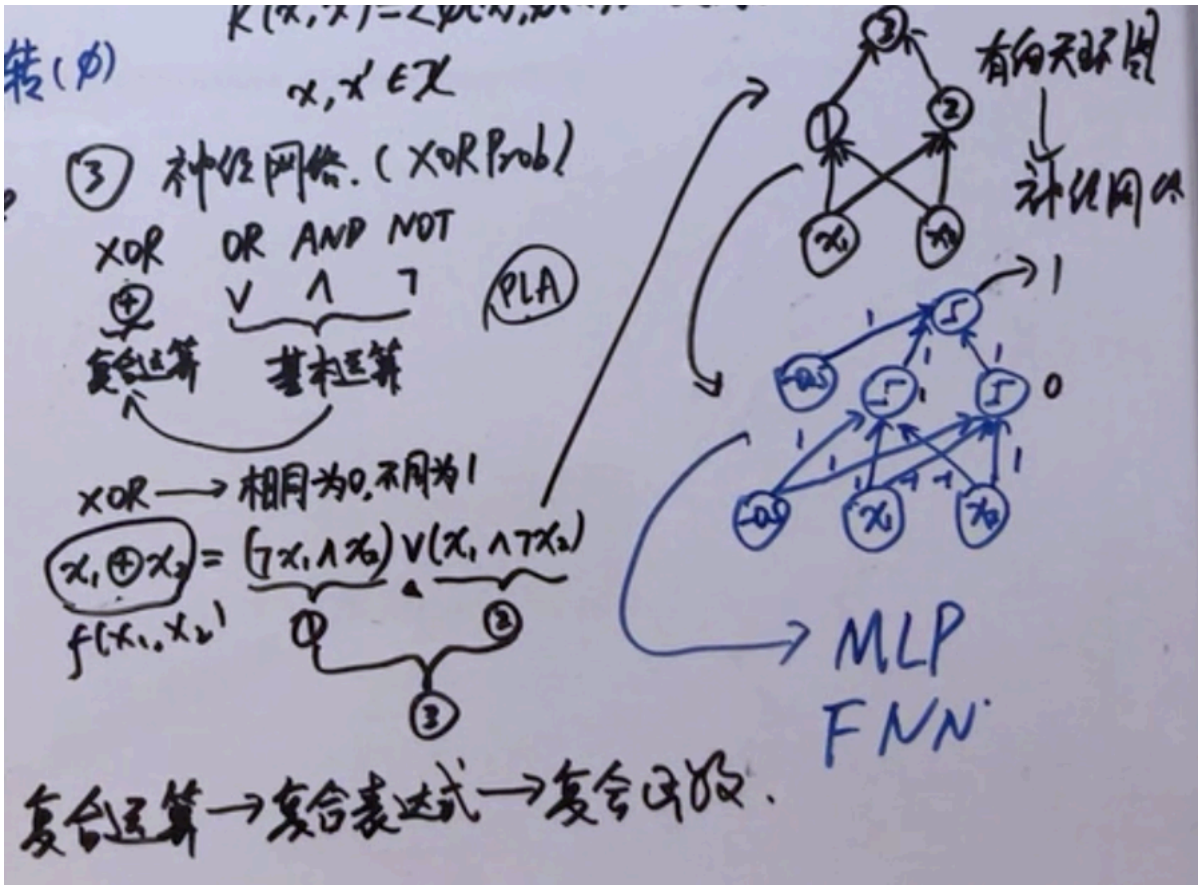
0, 1

1, 1

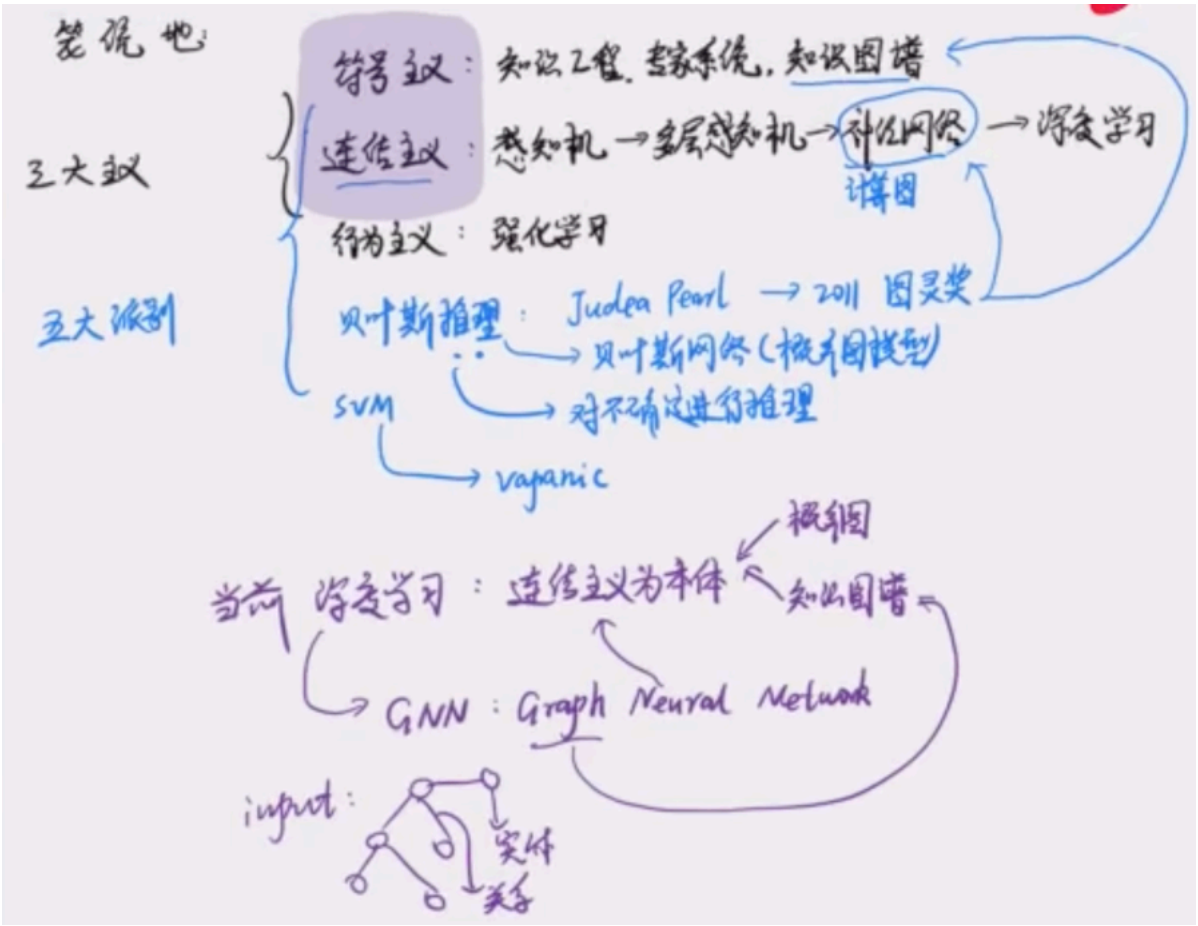
0 → 0

复合运算 → 复合表达式

符合运算-》复合表达式-〉复合函数



人工智能 两大阵营 三大主义


符号主义：
连接主义：
行为主义：
三大主义
五大派别
贝叶斯推理：
SVM
vapanic
当前 深度学习：连接主义为本体
概率图
知识图谱
GNN：Graph Neural Network
input：
实体
关系