神经网络第五章

2022年7月14日 星期四 13:52

1. Train / Dev / Test sets

训练，验证，测试

Best output → 无偏估计.

(Train : Test = 7:3)

(Train : Dev : Test = 6:2:2)

Big data sample : (98% : 1% : 1%)

或 (99% : 0.5% : 0.5%)

2. Bias / Variance.

high bias → 欠拟合

high variance → 过拟合 . . .

避免 high bias：增加隐藏层个数，神经元个数
训练时间↑，换用更复杂的 NN

避免 high variance：增加训练样本数据，或进行
正则化 Regularization.

(L2 regulation)

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}||w||_2^2$$

$$||w||_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w.$$

(L1 regulation)

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}||w||_1$$

$$||w_1|| = \sum_{j=1}^{n_x} |w_j|$$

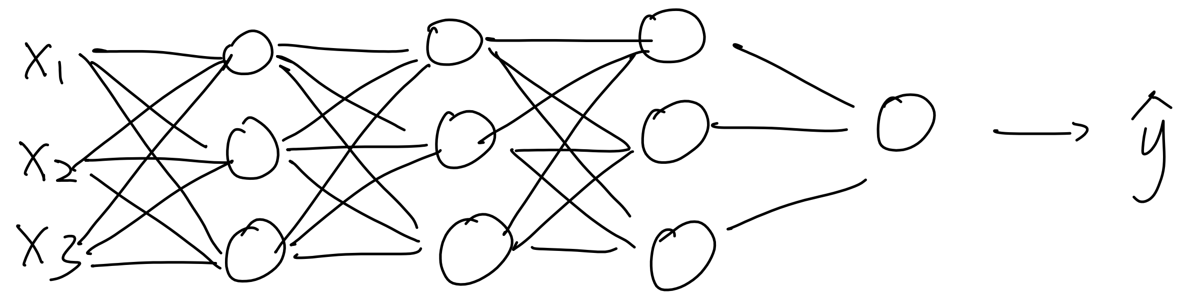
加入正则化项后.

$$dw^{[l]} = dw^{[l]}_{before} + \frac{\lambda}{m} w^{[l]}$$

$$w^{[l]} := w^{[l]} - \alpha dw^{[l]}$$

L_2 regularization → weight decay.

$$w^{[l]} := w^{[l]} - \alpha dw^{[l]}$$
$$= w^{[l]} - \alpha (dw^{[l]}_{before} + \frac{\lambda}{m} w^{[l]})$$
$$= (1 - \alpha\frac{\lambda}{m}) w^{[l]} - \alpha dw^{[l]}_{before}$$



dropout regularization :

训练过程中，对于每层神经元，按照一定概率将其从神经网络中丢弃
起到简化模型的效果，来避免过拟合

```
dl = np.random.randn(al.shape[0], al.shape[1]) < keep_prob
al = np.multiply(al, dl)
al /= keep-prob.
```

(设定 keep-prob = 0.8)

Other regularization methods

① 对已有样本进行处理 → 更多样本

“猫”图片进行水平翻转，垂直翻转，
任意角度旋转，缩放，扩大 . . . . . .

(不需要增加额外成本，却能防止过拟合 . . .)

② 增加一些 noise

③ 防止过拟合，提前 “early stopping”

Normalizing inputs.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)})^2$$

$$X := \frac{X-\mu}{\sigma^2}$$ (对数据归一化)

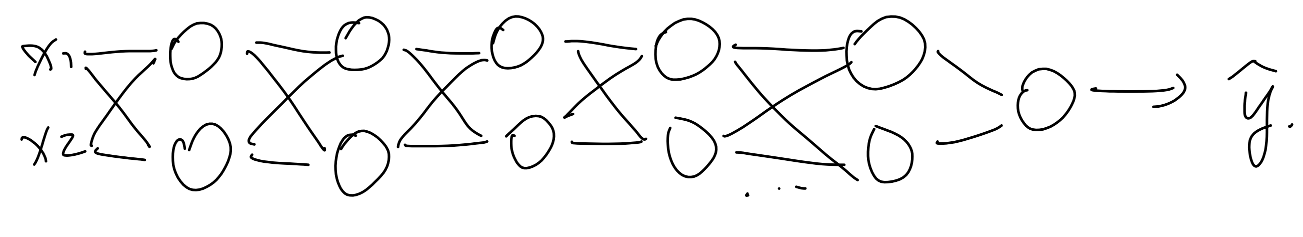
$x_1 \in [1, 1000]$, $x_2 \in [0, 1]$

让输入归一化同样的尺度上 . . .

由于 w_1, w_2 数值差异很大，只能选择很小的学习因子
避免了振荡

Vanishing and Exploding gradients

eg:



$$\hat{y} = w^{[L]} w^{[L-1]} w^{[L-2]} \cdots w^{[3]} w^{[2]} w^{[1]} X$$

L 非常大时，让当先过大或过小

Weight Initialization for Deep Networks

◦ 改善梯度爆炸问题方法.

$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n$

$a = g(z)$

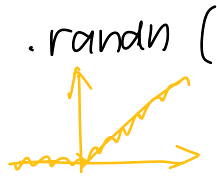
(让 w 与 n 有关，n 越大，X 越小)

w[l] = np.random.randn(n[l], n[l-1]) * np.sqrt(1/n[l-1])

(激活函数此时为 $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

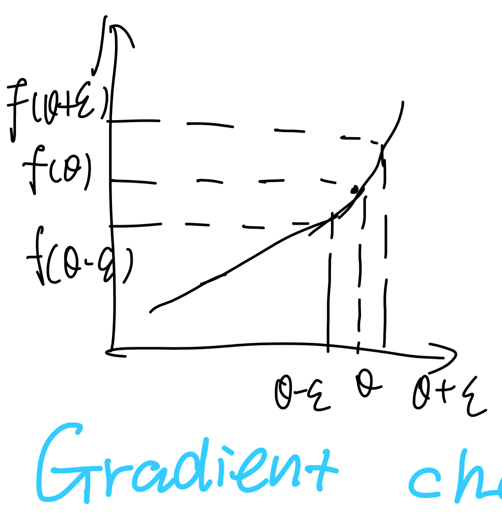
w[l] = np.random.randn(n[l], n[l-1]) * np.sqrt(2/n[l-1])

(激 . . . 为 ReLU



w[l] = np.random.randn(n[l], n[l-1]) * np.sqrt(2/n[l-1] * n[l])

Numerical approximation of gradients



$$g(\theta) = \frac{f(\theta+\varepsilon) - f(\theta-\varepsilon)}{2\varepsilon}$$

其中 ε 足够小.

Gradient checking

$$d\theta_{approx}[i] = \frac{J(\theta_1, \theta_2 \ldots \theta_i + \varepsilon, \ldots) - J(\theta_1, \theta_2, \ldots, \theta_i - \varepsilon, \ldots)}{2\varepsilon}$$

$$\frac{||d\theta_{approx} - d\theta||_2}{||d\theta_{approx}||_2 + ||d\theta||_2}$$

故两距离越小，
则反向梯度计算，则无 bugs

Grd checking Implementation

① 不要整个过程都梯度，仅作为 debug

② 不要忽略正则化项

③ 梯度检查时关闭 dropout，完成后打开 dropout

④ 开始检查，一段时间再检查