

网易云课堂

《神经网络和深度学习》

deeplearning.AI.

1-3. 监督学习 Supervised

“回归” 连续 regression

“分类” 离散 $[0,1]$ classification (Alpha Go)

structure learning 分类

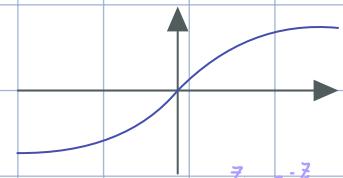
1-4. 非监督学习 Unsupervised

“聚类” 无标签 Octane

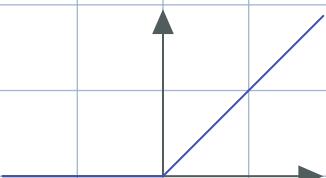
《激活函数》



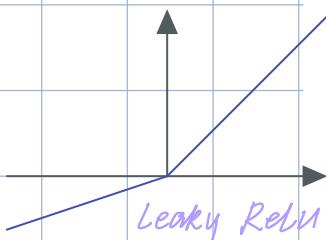
$$\text{Sigmoid. } a = \frac{1}{1+e^{-z}} \quad \text{tanh: } a = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 2\text{tanh}(z)$$



$$\text{tanh: } a = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 2\text{tanh}(z)$$



$$\text{ReLU. } a = \max(0, z)$$



$$\text{Leaky ReLU}$$

rectified linear unit 线性单元 $a = \max(0, \alpha z, z)$

Cross-Entropy 交叉熵损失

$$\sum p = 1 \quad \sum q = 1, \quad d = -\sum p \log q,$$

Logistic 损失只是交叉熵的特殊情况。

《Logistic》

$$w_1 * x_1 \\ w_2 * w_2 \\ \vdots \\ w_n * w_n \\ b$$

$$\sum = \sum w_n x_n + b \rightarrow o = \sigma(z) \rightarrow L(a, y)$$

$$\sigma = \frac{1}{1+e^{-z}}$$

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

$$\frac{dL}{dz} = \frac{\partial L}{\partial z} = \frac{dL}{da} \cdot \frac{da}{dz} \rightarrow a' = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1+e^{-z}-1}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}(1 - \frac{1}{1+e^{-z}}) = a(1-a) = a(1-a)$$

$$\therefore dL = a - y$$

$$dL = \frac{\partial L}{\partial w_n} = x_n \cdot dz, \quad db = 1 \cdot dz$$

$$w_n = w_n - \alpha \cdot dL \\ b = b - \alpha \cdot db$$

《神经网络》

※ 如果均采用线性激活函数，结果只会是线性组合，依旧是线性。

[输出层可被使用]

$$W^{[L]} : (n^{[L]}, n^{[L-1]})$$

$$Z^{[L]} = W^{[L]} \cdot a^{[L-1]} + b^{[L]}$$

$n^{[L]}$ $n^{[L]}$ $(n^{[L]}, n^{[L-1]}) \times (n^{[L-1]}, 1) \rightarrow (n^{[L]}, 1)$

$$b^{[L]} : (n^{[L]}, 1)$$

Backward for layer L.

> Input da^[L]

> Output da^[L-1] dW^[L] db^[L]

$$dZ^{[L]} = da^{[L]} * g^{[L]}'(Z^{[L]}) \quad \leftarrow a^{[L]} = g^{[L]}(Z^{[L]})$$

$$> dW^{[L]} = dZ^{[L]} * a^{[L-1]} \quad \leftarrow Z^{[L]} = W^{[L]} \cdot a^{[L-1]} + b^{[L]}$$

$$> db^{[L]} = dZ^{[L]}$$

$$> da^{[L-1]} = W^{[L]T} * dZ^{[L]}$$

$$\Downarrow dZ^{[L-1]} = W^{[L]T} * da^{[L]} * g^{[L-1]}'(Z^{[L-1]})$$

⋮

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = g^{[1]}(Z^{[1]})$$

$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(Z^{[2]})$$

⋮

$$A^{[L]} = g^{[L]}(Z^{[L]}) = \hat{Y}$$

$$dZ^{[L]} = A^{[L]} - Y$$

$$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L]T}$$

$$db^{[L]} = \frac{1}{m} np.sum(dZ^{[L]}, axis=1, keepdims=True)$$

$$dZ^{[L-1]} = dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]})$$

$$\vdots$$

$$dZ^{[1]} = dW^{[L]T} dZ^{[2]} g'^{[1]}(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[1]T}$$

$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis=1, keepdims=True)$$

Hyperparameters:

α learning rate

iterations 所有数据训练完需要几次

epoch 所有数据训练次数

batch size - 一次训练的数据量.

hidden layer

hidden units $n^{[l]}$..

activity.

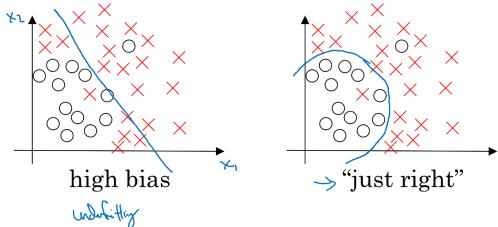
改善深层神经网络>

train / dev / test SET 训练, 验证, 测试
<10%

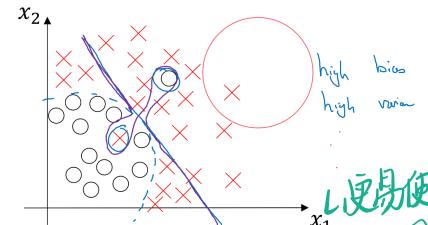
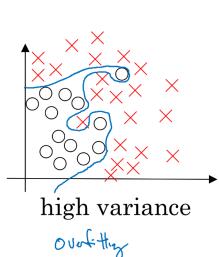
>高偏差 \rightarrow 欠拟合 bias 训练测试都差. \rightarrow Bigger Network

高方差 \rightarrow 过拟合 variance 训练集效果好, 测试差. More Data

Bias and Variance



High bias and high variance



惩罚项 $L_1 \|w\|_1$ 有效去除无关特征
 $L_2 \|w\|_2$ 限制绝对值.

> 正则化 Regularization 防止过拟合

$$\text{① L2 } J(W^{[l]}, b^{[l]}, \dots, W^{[L]}, b^{[L]}) = \frac{1}{m} \sum_{i=1}^n L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^L \sum_{j=1}^{n^{[l]}} \|W^{[l]}_{j,:}\|_F^2$$

"Frobenius Norm" $\sqrt{\text{norm2} \sum_{i=1}^n \sum_{j=1}^{n^{[l]}} (W^{[l]}_{j,:})^2}$

↓

$$dW^{[l]} = (\text{from back}) + \frac{\lambda}{m} W^{[l]}$$

$$W^{[l]} = W^{[l]} - \alpha dW^{[l]} = (\underbrace{1 - \frac{\lambda}{m\alpha}}_{<\downarrow \text{ weight decay}}) W^{[l]} - \alpha (\text{backprop})$$

正则项会减少权重矩阵的复杂度. $\uparrow W^{[l]} \approx 0$, 即减少了部分神经元.

误差减小, 防止过拟合

↑ 洋

$\downarrow z = w \cdot x + b$, 由于 a 是函数线性部分, 每层都趋线性.
 $a = g(z)$ i.e. \tanh \uparrow
整体拟合 Linear.

② Dropout

随机去除某神经元，防止过拟合

Layer d3.shape = a3.shape < keep-prob 0.8

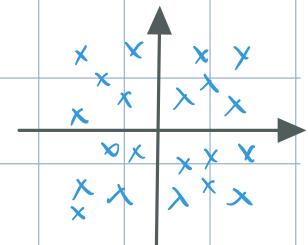
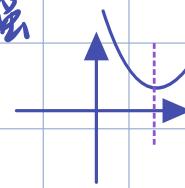
$$a_3 = a_3 * d_3$$

$$\underline{a_3 / \text{keep-prob}} \Rightarrow z^{[4]} = w^{[4]} \cdot a^{[3]} + b^{[4]}$$

Inverted dropout 反向随机失活 $\uparrow \rightarrow 0.2$, 使 $a^{[3]}$ 期望不变.

其他方法：getting more data 数据增强

early stopping 早停止.



> 标准化输入 使代价函数 J 优化更快

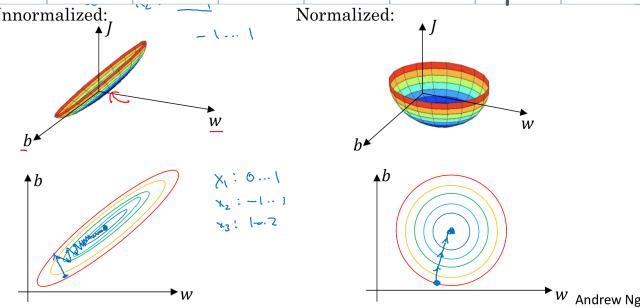
$$x = x - \mu \quad \text{标准化输入特征.}$$

$$x = x / \sigma^2 \quad \text{测试集和训练集统一 } \mu, \sigma.$$

→ 每个特征的分布都成为 $\sigma=1, \mu=0$.

> 梯度消失 / 爆炸.

- 权重指数级增长 / 下降



> 权重矩阵初始化.

$$W^{[L]} = np.random.randn(\text{shape}) * \sqrt{\frac{2}{n^{[L-1]}}} \quad g^{[L]}(z) = \text{ReLU}(z)$$
$$\dots * \sqrt{\frac{1}{n^{[L-1]}}} / \quad g^{[L]}(z) = \tanh(z)$$

或 $\sqrt{\frac{2}{n^{[L-1]} + n^{[L]}}}$

> 梯度检验 "双边误差" Grad check

$$g'(\theta) = \frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2\epsilon} \quad O(\epsilon^2). \quad \text{实际 check}$$
$$\frac{\|d\theta_{\text{approx}} - d\theta\|_2}{\|d\theta_{\text{approx}}\|_2 + \|d\theta\|_2}$$

② 包括正则项，但不能与 dropout 同时使用

• 优化算法 Optimization

> Mini-batch 梯度下降

1 epoch = 遍历一次所有数据 = iter * batch

mini-batch size = max \Rightarrow batch gradient descent $m < 2000$

mini-batch size = 1 \Rightarrow stochastic gradient descent 随机梯度下降

in practice. Iter max $2^n \Rightarrow 64, 128, 256 \dots [CPU/GPU memory]$

2 指数加权平均.

$$V_t = \beta V_{t-1} + (1-\beta) D_t, \quad V_t \approx \frac{1}{\beta} \text{ average.} \quad ? \Rightarrow (1-\beta)^{-1} = \frac{1}{\beta} \approx 10 \text{ 步长}$$

相当于平均了过去 10 次数据

3 偏差修正.

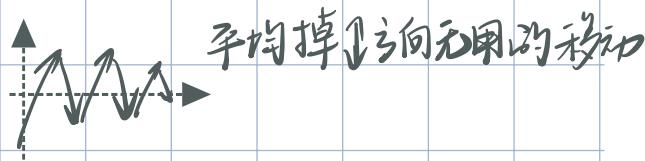
$$V_t = \bar{V}_t / (1-\beta)$$

2 > 动态梯度下降法 momentum

$$V_{dw} = \beta_1 V_{dw} + (1-\beta_1) dw, \quad w = w - \alpha V_{dw}$$

$$V_{db} = \beta_1 V_{db} + (1-\beta_1) db, \quad b = b - \alpha V_{db}$$

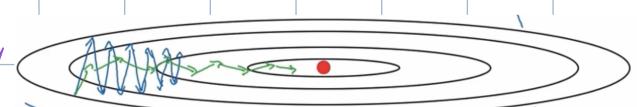
常用 $\beta = 0.9$ 不常用偏差修正.



3 > RMSprop root mean square 均方根.

$$S_{dw} = \beta_2 S_{dw} + (1-\beta_2) dw^2, \quad w = w - \alpha \frac{dw}{\sqrt{S_{dw}}}$$

$$S_{db} = \beta_2 S_{db} + (1-\beta_2) db^2, \quad b = b - \alpha \frac{db}{\sqrt{S_{db}}} \rightarrow$$



↓ 以设置较大的学习率.

2, 3, 均为了消除下降中无用的摆动

4 > Adam adaptive moment

$$V_{dw}^{corr} = V_{dw} / (1-\beta_1^t), \quad V_{db}^{corr} = V_{db} / (1-\beta_1^t)$$

$$S_{dw}^{corr} = S_{dw} (1-\beta_2^t), \quad S_{db}^{corr} = S_{db} (1-\beta_2^t)$$

$$w := w - \alpha \frac{V_{dw}^{corr}}{\sqrt{S_{dw}^{corr} + \epsilon}}, \quad b := b - \alpha \frac{V_{db}^{corr}}{\sqrt{S_{db}^{corr} + \epsilon}}$$

$$\beta_1 = 0.9$$

$$\beta_2 = 0.999$$

$$\epsilon = 1 \times 10^{-8}$$

> 学习率衰减

若 α 恒定，在训练后期不易收敛。

$$\alpha = \frac{1}{1 + \text{decay-num} * \text{epoch-num}} \alpha_0$$

其他： $\alpha = 0.95^{\text{epoch-num}} \cdot \alpha_0$ - exponential decay.

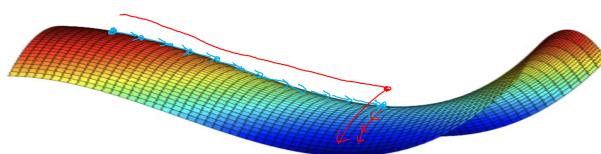
$$\alpha = \frac{K}{\sqrt{\text{epoch-num}}} \cdot \alpha_0, \text{ or } \alpha = \frac{K}{\sqrt{t}} \alpha_0$$



> 局部最优化

鞍点、 $\cap \cup$

Problem of plateaus



- Unlikely to get stuck in a bad local optima
- Plateaus can make learning slow

• 超参数调优

> 合适范围(标注)

$n^{[i]}$, # layers L 线性平均

$\alpha = 10^r, r \in [-4, 0]$ } 指数平均

$1 - \beta = 10^r, r \in [3, -1]$ }

> 超参数搜索

2) babysitting one model

training many models in parallel

> 标准化网络的激活

输入 \rightarrow 正则隐层输入 Batch Norm. (1) 标准化 (standardization)

(1) 标准化输入. $M = \frac{1}{m} \sum x^{(i)}$, $\sigma^2 = \frac{1}{m} \sum (x^{(i)} - M)^2$, $x^{(i)} = \frac{x^{(i)} - M}{\sigma}$

(2) Batch norm \leq mini-batch, $m = \text{size(minibatch)}$

$M = \frac{1}{m} \sum z^{(i)}$; $\sigma^2 = \frac{1}{m} \sum (z^{(i)} - M)$ 在激活前先 Norm

$$Z_{norm}^{(l)} = \frac{Z^{(l)} - \mu}{\sqrt{\sigma^2 + \epsilon}} \rightarrow \text{防止 } \sigma=0, \text{ 若每层网络都 } (0,1) \text{ 正态分布, 会导致学习不训特征.}$$

$$\tilde{Z}^{(l)} = \gamma Z_{norm}^{(l)} + \beta \quad \text{平移缩放参数, } \gamma, \beta \quad (\text{学习参数})$$

参数: $W^{(l)}, \beta^{(l)}, \gamma^{(l)}$.

\Downarrow $(w^{(l)}, \beta^{(l)}, \gamma^{(l)})$

$$Z^{(l)} = W^{(l)} a^{(l-1)}$$

$$\tilde{Z}^{(l)} = \gamma^{(l)} Z_{norm}^{(l)} + \beta^{(l)}$$

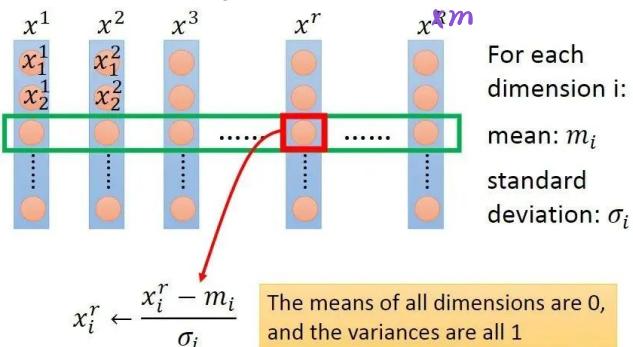
\Downarrow $b^{(l)}$ 常数均值消去

$$W^{(l)} = W^{(l)} - \alpha dW^{(l)}$$

$$\beta^{(l)} = \beta^{(l)} - \alpha d\beta^{(l)}$$

$$\gamma^{(l)} = \gamma^{(l)} - \alpha d\gamma^{(l)}$$

momentum, RMSprop, Adam, ...



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \dots m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = BN_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

不同batch间使用“指数加权平均”，来重新计算M.O.

Sum: ① 让模型的各个子模块(各层网络)更加相对独立
加快训练速度和稳定性, 放弃学习率.

解决: 1. covariate shift 协变量偏移. \Rightarrow 泛化性, 训练速度
输入的数据分布会变化, 发生在训练中或测试与训练集间.

2. Internal Covariate shift

② 网络层间的数据分布变化, 为下一层网络的学习困难.
也起到了正则化的作用(简化模型)

使每层的输入都在一定范围内, 不会严重饱和, 落在线性区域, 减少复杂度.
(类似加噪, 可代替 dropout).

> Softmax 回归 \Leftrightarrow hard max 单图单标签.

最后一层为 softmax layer.

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

$$z^{[L]} = w^{[L]} a^{[L-1]} + b^{[L]}, n^{[L]} = C. \text{ 分类数量.}$$

$$t = e^{(z^{[L]})}, a^{[L]} = e^{(z^{[L]})} / \sum_{j=1}^C t_j, a_i^{[L]} = t_i / \sum_{j=1}^C t_j, a = \hat{y}.$$

单层 softmax 可以作 C 个类别 的线性分类

损失函数.

$$L(\hat{y}, y) = -\sum_{j=1}^C y_j \log \hat{y}_j \quad \text{e.g. } -y_2 \log \hat{y}_2 = -\log \hat{y}_2, \quad y_2 \uparrow, L \downarrow.$$

〈机器学习策略〉

- > 反变化: 改变系统特征方向的性能
- > 单一数字评估指标 提高效率

Sing number evaluation metric + Dev Set.

e.g. 分类: Precision 查准率 Recall 查全率 (Accuracy)

$$F_1 \text{ Score} = \frac{2}{P+R} \text{ "Harmonic Mean"}$$

- > 满足与优化指标

Satisficing. & Optimizing.

N metric: 1 优化尽可能好

N-1 满足: 符合条件即可

- > Dataset Distribution

Dev: 评估不同方法, 改进和选择, 检测不同模型的区别

Test: 评估最终的成本偏差, 高置信度评价

Dev & Test 必需满足同一分布 i.e. 数据随机分布在验证集和测试集上.

传统 10⁵: Train: Test = 7:3; Train: Dev: Test = 6:2:2

大数据 10⁶: Train: Dev: Test = 98:1:1

* 可以合并 Dev & Test, 当不需要高置信度评估时.

正文化:

1 dim: 定义评价指标和标注. Dev 达到目标.

2 dim: 优化系统提高指标评分 达到目标. e.g. 修改损失函数.

实际应用与 Dev/Test 不符, 改变指标和 Dev Set.

> 与人类比较.

ML 精度可以超过人类, 最终趋近于 贝叶斯最优误差.

> 可避免偏差、改善模型.

↓ 可用人类水平估计

1% Human-level error (Bayes)

↑ Available bias 偏差 ↗ 更大的模型(网格)
10% Training error ↗ 优化算法. momentum, RMSprop, Adam.
15% Dev error ↗ 其他架构/超参数搜索.

↑ Variance 差 ↗ 更大的数据集
20% Test error ↗ 正则化, L2, dropout, 数据增强
其他架构/超参数搜索.

↑ Dev overfitting 过拟合. 更多 Dev 数据

Test error

> 超越人类

> 错误分析

Image	Dog	Great Cats	Blurry	Instagram	Comments
1	✓			✓	Pitbull
2			✓	✓	
3		✓	✓		Rainy day at Zoo
:	:	:	:	:	
% of total	8%	43%	61%	12%	

train set: 评估机误差有鲁棒性, 但系统误差无.

dev/test set: 可忽略较少标签错误的影响.

> 不匹配数据

Dev/Test 的数据较少，无法直接训练。(目标数据少) 用户实际数据

Train 数据用其他类似数据代替。 网站爬虫数据

即 T 与 D/V 数据分布不同，模型优化分析调整。

增加 Training-Dev set，分布与 T 一致。

⋮

Training set error

↑ Variance.

⇒ Training-Dev set error

↓ Data mismatch 数据不匹配

Dev error.

⋮

> 解决

加随机噪声，若只选择所有可能的部分来模拟，易造成过拟合

• 多类数据同时学习

> 迁移学习 [平行]

Transfer 数据量大的问题 → 数据量小的问题 相同输入。

↓ 模型已有低层次特征。

方法：去掉已有模型的输出层，并在其后添加 n 层 CNN 网络，重新训练。

> 多任务学习 [并行]

Multi-task 一个网络解决多个问题，e.g. 检测多种物体在一张图上

是够大，不会降低性能

Set one 是一圈给一种标签。

多任务物间的特征可共用

$$\text{该图: } y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}_{4 \times m} \quad \text{Loss} = \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^4 l(y_j^{(i)}, \hat{y}_j^{(i)})$$

$\downarrow \text{logistic}$

$$-y_j^{(i)} \log \hat{y}_j^{(i)} - (1-y_j^{(i)}) \log (1-\hat{y}_j^{(i)})$$

> 端到端的 DL

End-to-End -一步到位, 因为从头到尾

复杂网络 + 大量数据

$$d \begin{bmatrix} h \\ \vdots \\ h \end{bmatrix}^T \stackrel{d}{\rightarrow} d \begin{bmatrix} \vdots \end{bmatrix}$$