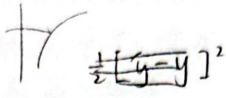
Course1: Toy Example

Deep learning Basics & Predictive models

如何选择损失函数/优化器/神经网络/正则化方法
loss optimizer neural network regularization

如何通过数据增强/生成数据来训练更强大的模型
augment generate

$\lambda=1$



概念：

1. Supervised learning 监督学习：通过使用带标签的历史数据训练模型，使其能够对未知新数据进行预测的机器学习方法。

2. Classification 分类：监督学习的核心任务之一，旨在根据输入特征将其预测为两个或多个离散的类别标签。

3. Transfer Learning 迁移学习：一种机器学习方法，它将从一个任务（源任务）中学到的知识，应用到另一个不同但相关的任务（目标任务）中，从而减少对大量新标注数据的依赖并加快模型训练。

4. Unsupervised Learning 无监督学习：一种机器学习方法，它在使用无标签数据进行训练时，自主发现数据内部隐藏的结构、分布或模式（如聚类或降维），而无需人为提供正确答案。

5. Generative Model：通过从训练数据中学习隐含的概率分布，能够生成与原始数据相似的全新样本（如图像、文本或音频）

6. Generalization 泛化：模型在训练集上学习后，能够对未曾见过的全新数据做出准确预测的能力。

7. Linear Models 线性模型：假设输入特征与输出之间存在线性关系，通过特征的加权和进行预测。形如：$y(x,w) = w_0 + w_1x + w_2x^2 + \cdots + w_Mx^M = \sum_{j=0}^{M} w_jx^j$，$x$为特征，$w_j$为模型参数（权重），$y$为预测输出，对$w_j$而言是线性。

~~8. Model Complexity~~

8. Overfitting 过拟合：模型过度学习训练数据中的噪声和细节，导致其在训练集上表现极好，但在未见过的测试集上表现很差。

在deep learning中参数数量n远远大于数据量

$\frac{1}{2}\sum(\hat{y}-y)^2$

MSE

$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,w)-t_n\}^2$

预测值 真实

1. 可导且平滑
2. 若假设 $\varepsilon \sim N(0,\beta)$，最大似然的结果等价于最小化MSE.
3. 因为误差被平方，较大的误差会被显著放大.

$L(\theta) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y^{(i)}-h_\theta(x^{(i)}))^2}{2\sigma^2}\right)$

$\ell(\theta) = n\log\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2}\sum(y^{(i)}-h_\theta(x^{(i)}))^2$

$|\hat{y}-y|$ $\frac{1}{N}\sum|\hat{y}-y|$ MSE

MAE

$E(w) = \frac{1}{n}\sum_{n=1}^{N}|y(x_n,w)-t_n|$

1. 与原始数据单位相同
2. 对异常异常值鲁棒 ok
3. 在0处不可导

$\sqrt{\frac{1}{N}\sum(\hat{y}-y)^2}$

RMS

$E_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\{y(x_n,w)-t_n\}^2}$

误差的均方根，单位与原始数据一致.

M越大，各个分量w_j的绝对值也会增大 → 拟合复杂.

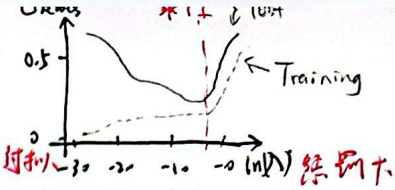
模型拟合数据点的方式是最小化误差方程.

增加训练数据可以减小过拟合.

9. Weight Decay 权重衰减：通过在损失函数中增加权重 w 的平方和惩罚项，来限制模型参数过大，从而有效抑制过拟合的常用正则化技术。

$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,w) - t_n\}^2 + \frac{\lambda}{2}\|w\|^2$，$\lambda \in (e^{-5}, e^{-3})$，$\ln\lambda \in (-5,-3)$ 正常范围

$\|w\|^2 = w^Tw = w_0^2 + w_1^2 + \cdots + w_M^2$ 减小 w_i 的绝对值大小。



惩罚小，过拟合　　惩罚大，欠拟合

10. Validation set 验证集：用于评估模型在训练过程中的泛化能力并调整超参数（如学习率、正则化强度）的数据子集，它不参与梯度更新，但会影响模型的选择与配置。

11. Test set 测试集：在模型训练和所有调参工作完全结束后，用来对最终一次性评估模型泛化能力的独立数据。

入，M 需要通过验证集确定，然后可以将验证集加入训练集

Why need test and validation sets? If limited data?

验证集用于调参，测试集用于评估，并且前期不能给模型接触。

数据少可以采用交叉验证。训练集分为 K 份，循环 K 次，每次用不同一份作为验证集。

Course 2: Linear Classification

~~Basic~~ Linear function: $y(x,w) = w_0 + w_1\phi_1(x) + \cdots + w_{M-1}\phi_M(x) = w_0 + \sum_{i=1}^{M-1} w_i\phi_i(x)$

↑ learnable parameters 可学习参数　　因子/基函数

1. Feature extraction 特征提取：将原始数据（如文本、图像、信号）通过变换或映射，转换为更具代表性，信息更紧缩的数值特征向量的过程。eg. PCA

真实　预测

$t_i = y_i(x_i,w) + \varepsilon_i$，$\varepsilon \sim N(0,\sigma^2) \rightarrow t_i \sim N(y_i(x_i,w), \sigma^2)$

$f(t_i|x_i,w) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(t_i - y_i(x_i,w))^2}{2\sigma^2}\right)$

$L(w) = \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(t_i - y_i(x_i,w))^2}{2\sigma^2}\right)$

$\varepsilon \sim N(0,\sigma^2)$ → "最大似然" ⇔ "最小化MSE"

$l(w) = \ln L(w) = \sum_{i=1}^{n}\ln\left[\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(t_i - y_i(x_i,w))^2}{2\sigma^2}\right)\right]$

$= n\ln\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(t_i - y_i(x_i,w))^2$

$= -n\ln(\sqrt{2\pi}\sigma) - \frac{1}{\sigma^2}E_0(w)$

$E_0 = \frac{1}{2}\sum_{n=1}^{N}(t_n - w^T\phi(x))^2$ 均方差

~~$\varepsilon \sim$ 均值为0的拉普拉斯分布~~

$\varepsilon \sim$ 均值为0的拉普拉斯分布 → "最大似然" ⇔ "最小化MAE"

尺度参数

$p(x|\mu,b) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$ 拉普拉斯分布

$f(t_i|x_i,w) = \frac{1}{2b}\exp\left(-\frac{|t_i - y_i(x_i,w)|}{b}\right)$

$L(w) = \prod_{i=1}^{n}\frac{1}{2b}\exp\left(-\frac{|t_i - y_i(x_i,w)|}{b}\right)$

$l(w) = -n\ln(2b) - \frac{1}{b}\sum_{i=1}^{n}|t_i - y_i(x_i,w)|$

$= -n\ln(2b) - \frac{n}{b}E_0(w)$　　$E_0(w) = \frac{1}{n}\sum_{i=1}^{N}|t_n - w^T\phi(x)|^2$

y_i 为真实值

$\hat{y}_i$ 为预测值

2. One-hot encoding 独热编码：将离散的类别特征转换为稀疏二进制向量的方法：为每个类别创建一个独立的维度，在样本所属类别的维度上置1，其余维度置0.

Cat 1 {1,0,0}

Chicken 2 → {0,1,0}

Dog 3 {0,0,1}

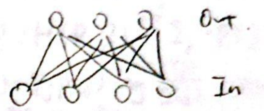
{1,2,3} {(1,0,0),(0,1,0),(0,0,1)}

X Y

$$\text{softmax}(x_i)=\frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

3. full-connected layer 全连接层：一种神经网络层，其中每个输入神经元与每个输出神经元之间均有权重连接，通过对输入向量进行线性变换（$y=Wx+b$）来学习全局特征间的组合关系.

（W：Weights，b：bias）



则 W 为 3×4

$$\underset{3\times4}{W}\,\underset{4\times1}{x}+\underset{3\times1}{b}$$

$P(y=i)\propto \exp O_i \xrightarrow{\text{权一化}} \hat{y}=\text{softmax}(O)$，$\hat{y}_i=\frac{\exp(O_i)}{\sum_j\exp(O_j)}$，和为1. ← 概率特性

$$\text{Softmax}(x_i)=\frac{\exp(x_i)}{\sum_j\exp(x_j)}$$

$\underset{j}{\arg\max}\,\hat{y}_j=\underset{j}{\arg\max}\,O_j$ ← 单调性

向量化：对每个样本计算 $O=Wx+b$ 太慢

↓

$X\in\mathbb{R}^{n\times d}$，$n$ 为样本数，d 为维度. $\underset{n\times q}{O}=\underset{n\times d}{X}\underset{d\times q}{W}+\underset{n\times q}{b}$ ← 会通过广播扩展到与 XW 相同维度

$\hat{Y}=\text{softmax}(O)$. 得到每个样本在 q 个类别上的概率分布.

4. Cross-entropy loss 交叉熵损失：用于量化模型预测的概率分布 $\hat{y}$ 与真实标签分布 y（通常为独热编码）之间的差异，其公式为 $-\sum_i y_i\log\hat{y}_i$；分类任务.

在分类任务中，它等价于最大化预测正确类别的对数似然，且当预测完全准确时损失为0. $l(y,\hat{y})=-\sum_{i=1}^{q}y_i\log\hat{y}_i\in[0,+\infty)$

正确类别的负对数概率

模型输出 $\hat{y}$ 表概率分布，真实标签 y 为独热编码

$l(y,\hat{y})=-\log\hat{y}_k$.

表达类别间相似度时用软标签.

$\hat{y}=\text{softmax}(O)$，where $\hat{y}_j=\frac{\exp(O_j)}{\sum_j\exp(O_j)}$，$l(y,\hat{y})=-\sum_{j=1}^{q}y_j\log\hat{y}_j$

$$\begin{aligned}
l(y,\hat{y})&=-\sum_{j=1}^{q}y_j\log\frac{\exp(O_j)}{\sum_{k=1}^{q}\exp(O_k)}\\
&=\sum_{j=1}^{q}y_j\log\left(\sum_{k=1}^{q}\exp(O_k)\right)-\sum_{j=1}^{q}y_jO_j\\
&=\log\left(\sum_{k=1}^{q}\exp(O_k)\right)-\sum_{j=1}^{q}y_jO_j
\end{aligned}$$

$$\begin{aligned}
\frac{\partial l(y,\hat{y})}{\partial O_j}&=\frac{\partial}{\partial O_j}\left(\log\left(\sum_{k=1}^{q}\exp(O_k)\right)\right)-\frac{\partial}{\partial O_j}\left(\sum_{j=1}^{q}y_jO_j\right)=\frac{1}{\sum_{k=1}^{q}\exp(O_k)}\frac{\partial}{\partial O_j}\left(\sum_{k=1}^{q}\exp(O_k)\right)-y_j\\
&=\frac{\exp(O_j)}{\sum_{k=1}^{q}\exp(O_k)}-y_j=\text{softmax}(O_j)-y_j=\hat{y}_j-y_j
\end{aligned}$$

交叉熵梯度 = 模型预测概率 − 真实

eg. $q=3$, $O=[2,1,0.1]$, $y_1=1, y_2=0, y_3=0$.

~~则~~ Softmax 输出: $\hat{y}=[0.659, 0.242, 0.099]$.

~~则~~ 交叉熵损失: $l=-\log(0.659)\approx 0.417$.

梯度 $\frac{\partial l}{\partial O}=[0.659-1, 0.242-0, 0.099-0]=[-0.341, 0.242, 0.099]$

$O\leftarrow O-\eta\nabla l$. 梯度相反方向. $\uparrow_2$, $\downarrow_1$, $\downarrow_{0.1}$

$\Sigma -P(j)\log Q(j)$ $\quad XW+b$ $\quad q\ \log q$

Entropy. CE loss and KL divergence.

P为真实分布, Q为模型预测分布 $\quad$ ~~$\Sigma P(j)\log P(j)$~~ $\quad$ $H(P,Q)$ $H[P]$ $\quad \Sigma -P(j)\log P(j)$ $\quad q\cdot\frac{1}{q}\log\frac{1}{q}$ $\quad H[P]=\Sigma -P(j)\log P(j)$

均匀分布 $[0,\log q]$ Entropy 熵(衡量一个概率分布的不确定性, 分布越集中熵越小): $H[P]=\sum_j -P(j)\log P(j)$

$[0,+\infty)$ Cross-entropy 交叉熵(衡量两个分布P和Q的差异, 越小差异越小): $H(P,Q)\overset{def}{=}\Sigma_j -P(j)\log Q(j)$

$[0,+\infty)$ KL divergence KL散度(衡量从分布Q到分布P的差异): $D_{KL}(P\|Q)=\Sigma_j P(j)\log\frac{P(j)}{Q(j)}$ 不对称.

$D_{KL}(P\|Q)=\Sigma P(j)\log\frac{P(j)}{Q(j)}$ $\quad \Sigma -P(j)\log Q(j)$ $\quad \Sigma P(j)\log\frac{P(j)}{Q(j)}$

$H(P,Q)=H[P]+D_{KL}(P\|Q)$ 字母顺序

① 当P固定时, 最小化交叉熵等价于最小化KL散度. $\Sigma P(j)\log\frac{P(j)}{Q(j)}$

② 若P分布不为独热, 则 $H[P]>0$. 即使 $P=Q$, ~~则~~ $H(P,Q)=H[P]$ 也 >0.

③ 只有当P分布为独热, $H[P]=0$. $H(P,Q)=D_{KL}(P\|Q)$. 完全正确预测才会使 $H(P,Q)=0$.

CE Loss在分类任务中优于MSE的根本原因: $\frac{\partial L_{mse}}{\partial z}=(\hat{y}-y)\hat{y}(1-\hat{y})$. 其中 $\hat{y}\cdot(1-\hat{y})$ 在 $\hat{y}$ 接近0或1时该项趋近于0. 梯度小. 而 $\frac{\partial L_{CE}}{\partial o}=\hat{y}-y$ 直接反映预测误差, ~~对错~~ 在[0,1]上对错误值更新更快.

Course 3: Multi-Layer Perceptron

线性模型中, 参数对结果的方向的影响是单调的

单调不足够 $\xrightarrow{\text{体现}}$ 温度对于健康 $\quad$ 固定的基函数不足够 $\xrightarrow{\text{体现}}$ 不同图像

Multilayer Perceptron 多层感知机: 一种由输入层, 一个或多个隐藏层和输出层构成的前馈神经网络, 通过激活函数引入非线性变换, 能学习复杂的函数映射关系.

$H=XW^{(1)}+b^{(1)}$. $O=HW^{(2)}+b^{(2)}$

$O=(XW^{(1)}+b^{(1)})W^{(2)}+b^{(2)}=XW^{(1)}W^{(2)}+b^{(1)}W^{(2)}+b^{(2)}=XW+b$ 等价于单层

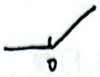
根据通用近似定理, 对于具有线性输出层和至少一个使用"挤压"性质的激活函数的隐藏层组成的前馈神经网络, 只要其隐藏层神经元的数量足够, 它可以以任意的精度来近似任何从一个定义在实数空间中的有界闭集函数.

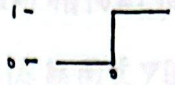
三种激活函数：

$H = \sigma(XW^{(1)} + b^{(1)})$

$O = HW^{(2)} + b^{(2)}$

① ReLU　$\max(x, 0)$





$\mathrm{ReLU}(x) = \max(x, 0)$　$\in [0, +\infty)$　梯度 $\in$ ~~$[0, +\infty)$~~ $[0, 1]$

简单、梯度、无梯度消失（但有死神经元），中间层常用

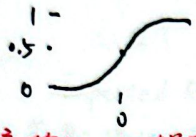
某些导元始终输出0

$\frac{1}{1+e^{-x}}$　$[1+\exp(-x)]^{-1}$

② Sigmoid（在二分类中当 softmax 用）

$\mathrm{sigmoid}(x) = \frac{1}{1+\exp(-x)}$　$\in [0, 1]$　梯度 $= \frac{\exp(-x)}{(1+\exp(-x))^2}$　$\in [0, \frac{1}{4}]$

$+ \frac{\exp(-x)}{(1+\exp(-x))^2}$



$x=0$ 时，最大 $= \frac{1}{(1+1)^2} = \frac{1}{4}$

也可写为：$\mathrm{sigmoid}(x)(1-\mathrm{sigmoid}(x))$

平滑、梯元导输出（但梯度消失），二分类常用

$\max(x, 0)$

$\frac{1}{1+e^{-x}}$

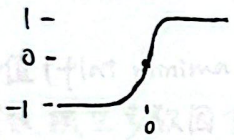
$\frac{1-e^{-2x}}{1+e^{-2x}}$

③ Tanh

$\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}$　$\in [-1, 1]$　梯度 $= \frac{4e^{-2x}}{(1+e^{-2x})^2}$　$\in [0, 1]$

$x=0$ 时，最大 $= \frac{4}{(1+1)^2} = 1$

"1"刚好，再大会发生梯度爆炸

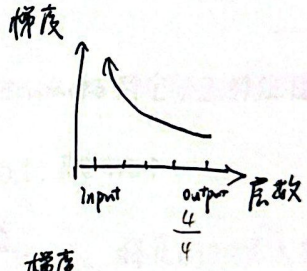


对称，范围广（但优化仍难），回归问题常用

梯度爆炸的表现：损失值变为 NaN　更新时间↑

原因：网络过深，激活函数不当，学习率过高　密相承

解决方法：梯度裁剪、残差连接，降低学习率。

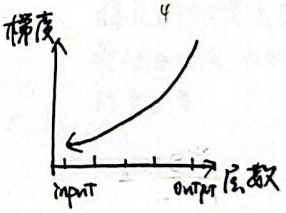


$\frac{4}{4}$

梯度消失的表现：损失值下降缓慢，权重几乎不更新

原因：网络太深，激活函数不当

解决方法：ReLU，残差连接。



一般而言，靠近输出层梯度越大，靠近输入层梯度越小。

$(1-e^{-2x})$

$2e^{-2x} + 2e^{-4x} + 2e^{-2x} - 2e^{-4x}$

隐藏层 → 增加非线性表达（隐藏层通过多层非线性变换，使得神经网络能够拟合任意复杂的函数）

激活函数 → 单加层仍为线性组合（如果没有激活函数，无论网络有多少层，最终的输出仍然是输入的线性组合）

Sigmoid 缺点 → 梯度消失，$(\mathrm{sigmoid})'_{max} = 0.25 < 1$

$\frac{4e^{-2x}}{(1+e^{-2x})^2}$

$\frac{1}{2}$

如果所有权重初始化为0？神经元会学习到相同的特征，导致网络失去表达能力。

为网络为什么更深而不是更宽？更深的网络能用更少的参数指数级地表达更复杂的函数，参数效率更高；且深层网络更容易捕获层次化的特征，泛化能力也更强。

Hinton的AlexNet使用ReLU来加速训练。

~~Course 4：Optimization~~

$$\theta^* = \arg\min_\theta \sum_{(x,y)\in D} L(f_\theta(x), y)$$

（损失；$\arg\min_\theta$：梯度下降；D：数据集；f_θ：神经网络）

Expected Risk　　Empirical Risk

真实分布 $P(z)$ 计算**期望风险** $R(f)$，训练集 D_n 上计算**经验风险** $\hat{R}(f, D_n)$。

$R(f) \neq \hat{R}(f, D_n)$，两者之间的差距称为泛化误差。（过拟合）

训练集 D_n 与真实数据分布 $P(z)$ 独立同分布 $\Rightarrow E(\hat{R}(f, D_n)) = R(f)$

$$E[\hat{R}(f, D_n)] = E\left[\frac{1}{n}\sum_{i=1}^{n} L(f, z_i)\right] = \frac{1}{n}\sum_{i=1}^{n} E[L(f, z_i)] = E[L(f, z)] = R(f)$$

平坦的最小值（flat minima of empirical risk）通常对应更好的泛化能力。
这意味着即使模型参数因为量化误差、随机初始化或训练集变动而发生微小偏移，损失值也不会有大大变化。

可能停止优化 ⇒ 认为优化收敛：梯度趋0 $\nabla f \approx 0$
- ~~全局~~ ~~最小~~ Global minimum（Hessian矩阵全为正特征值 ≥ 0）
- Local minimum（Hessian矩阵全为正特征值 ≥ 0）
- Saddle point 损失函数在某些方向↓，其它方向↑（Hessian矩阵同时有正和负的特征值）

可能减慢收敛

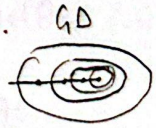
梯度的方向是函数值上升最快的方向，大小是该方向的变化率。

$$\theta_{t+1} = \theta_t + \eta g_t = \theta_t - \eta \nabla L(\theta_t)$$

Gradient Decent 梯度下降 GD

一种迭代算法，通过沿与梯度相反的方向移动来最小化损失函数

$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$。$\theta$ 为参数，η 为学习率，$\nabla L(\theta_t)$ 为损失函数 L 的梯度。



优：① 对于凸函数能收敛到全局最小；② 实现简单。
缺：① 计算成本高，每次迭代需使用整个数据集；② 对大数据集，训练很慢；
③ 对于非凸函数，会卡在局部最小。

$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; x_i, y_i)$　　随机初始化 θ。

Stochastic Gradient Decent 随机梯度下降 SGD

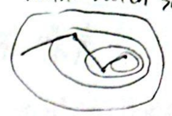
GD的一种变体，使用单个随机样本的梯度来更新参数。

$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t; x_i, y_i)$

优：① 效率高，每次迭代只用一个样本；
② 波动性有助于逃离局部最小。
缺：① 更新不稳定，导致收敛慢；② 可能无法达到精确最小值

Mini-batch SGD 小批量SGD

$\theta_{t+1}=\theta_t-\eta\nabla L(\theta_t; X_b, Y_b)$



GD和SGD之间的折衷，每次迭代使用小批量样本。

$\theta_{t+1}=\theta_t-\eta\nabla L(\theta_t; X_b, Y_b)$ (X_b, Y_b)是样本的一个小批量

优：①平衡了效率与稳定性；②可利用矩阵计算(GPU)加速；③很适合深度学习&大数据集.

缺：①需调整批量大小(eg. 32, 64, 256)；②仍容易陷入局部最小.

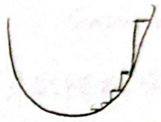
Momentum的主要优势是加速SGD在相关方向上的收敛，而不是减小方差（Adam等自适应方法的特点）

Batch Size & Learning Rate

批量越大，梯度方差越小&噪声越小，用大学习率

批量越小，梯度方差越大&噪声越大，用小学习率.

(学习率调度) Learning Rate Schedule 可以动态调整学习率.



改进SGD
- ① 学习率：学习率衰减(过大会发散，过小太慢；故前期大，后期小)
  - RMSprop 自适应学习率
  - 3. Cyclical learning rates
- ② 梯度方向：Momentum
  - NAG
  - Gradient clipping

(RMSprop + Momentum — Adam)

1998+24=2022

加了动量会更快

SGD: $\Delta\theta_t=-\alpha g_t$ 只看当前步

Cyclical Learning Rates 循环学习率，帮助模型跳出坏区域.

$\Delta\theta_t=-\alpha g_t$

Momentum Method 动量法：$\Delta\theta_t=\rho\Delta\theta_{t-1}-\alpha g_t=-\alpha\sum_{\tau=1}^{t}\rho^{t-\tau}g_\tau$

α为学习率，g_t为当前步的梯度，$\Delta\theta_t$为第t步的参数更新量，ρ为动量因子，控制历史梯度的衰减速度.

0.9/0.99 历史情况的重要

$\Delta\theta_t=\rho\Delta\theta_{t-1}-\alpha g_t$

Nesterov accelerate gradient (NAG)：$\Delta\theta_t=\rho\Delta\theta_{t-1}-\alpha g_t(\theta_{t-1}+\rho\Delta\theta_{t-1})$ 在这个点计算梯度

$\rho\Delta\theta_{t-1}$为历史更新方向，$(\theta_{t-1}+\rho\Delta\theta_{t-1})$为向前看一步的位置，$\alpha$为学习率.

优：“预见”能力，比动量法震荡更小，收敛更快.

$\Delta\theta_t=\rho\Delta\theta_{t-1}-\alpha g_t(\theta_{t-1}+\beta\theta_{t-1})$

Adam的本质是Momentum(动量)+RMSprop(自适应学习率)

$\Delta\theta_t=\rho\Delta\theta_{t-1}-\alpha g_t(\theta_{t-1}+\rho\Delta\theta_{t-1})$

Gradient Clipping
- by value: $g_t=\max(\min(g_t,b),a)$
- by norm: $g_t=\frac{b}{\|g_t\|}g_t$

$\Delta\theta_t=\rho\Delta\theta_{t-1}-\alpha g_t$

Parameter Initialization

如果初始参数全为0会导致对称权重 → 随机初始化($N(0,\varepsilon^2)$或$U[-\varepsilon,\varepsilon]$).

数据归一化使得梯度从指向最小值方向不直接 → 指向最小值方向直接.

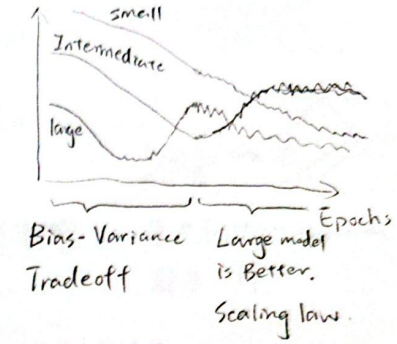
同时减小震荡，可使用更大学习率，对学习率敏感度更低.

Course 5: Regularization

过拟合原因：① 训练数据不足；② 参数过多(过参数化)；③ 数据存在噪声

double decent，且后期下降时震荡大 → overfitting 不是单调的

大模型(过参数化)通常需要更少的 epoch 就会开始过拟合



① 偏差的平方 $bias^2 = (\bar{g}(x) - f(x))^2$：$(\bar{g}(x) - f(x))^2$

平均模型 $\bar{g}$ 与真实目标函数 f 之间的差距。衡量模型集的表达能力。 表达能力越高，偏差越小

② 方差 $Var = E_D[(g^{(D)}(x) - \bar{g}(x))^2]$：$E_D[(g^{(D)}(x) - \bar{g}(x))^2]$

不同数据集训练出的模型 $g^{(D)}$ 与平均模型 $\bar{g}$ 之间的差异。衡量模型对数据扰动的敏感度。

小模型集：偏差大，方差小。 大模型集：偏差小，方差大。 ← 模型开始记住训练集的随机噪声

总误差 = $bias^2$ + variance + 数据本身的噪声

↑ 训练集上，模型开始拟合真实规律

数据数据是能确保有较好的泛化能力：$N \geq 10 \cdot$ 模型参数量

1、Regularization 正则化：通过对模型复杂度施加惩罚(如限制参数大小或稀疏性)来抑制过拟合、提升泛化能力的技术。

正则化 {
① 在优化过程中添加约束条件：L1/L2 正则化，数据增强
② 干扰优化过程：Weight Decay 权重衰减，SGD，Early stopping 早停
}

与 L2 区别：在自适应优化器中，L2 正则化效果会扭曲，Weight decay 正则化力度与梯度大小解耦

2. Data augmentation 数据增强：对现有训练数据进行微小的随机变换(如旋转、裁剪、翻转)生成更多变体，以低成本扩充数据集，从而缓解过拟合。不能改变 label

提高泛化能力的方法：$\|w\|_1 = \sum|w_i|$　$\|w\|_2^2 = \sum w_i^2$

① L1/L2 正则化：$\theta^* = \arg\min_\theta \frac{1}{N}\sum_{n=1}^{N} L(y^{(n)}, f(x^{(n)}, \theta)) + \lambda \ell_p(\theta)$，$p \in \{1, 2\}$，$\lambda$ 是正则化强度

λ 越大强度越大

L1: $\|w\|_1 = \sum_{j=1}^{m}|w_j|$，L2: $\|w\|_2^2 = \sum_{j=1}^{m} w_j^2$

稀疏化，用于特征选择(公司裁员，裁低工资)　削弱所有权重但仍保留，对更大的权重影响更大也叫权重衰减(公司降公工资，先降高薪资高的)

② Weight Decay 权重衰减：在每次迭代时引入一个衰减系数 w.

$$\theta_t \leftarrow (1-w)\theta_{t-1} - \alpha g_t$$

在标准 SGD 中，权重衰减与 L2 正则有相同效果

在 Adam 中，因为自适应学习率，二者不同，应用 Weight Decay.

$(L(\theta) + \frac{\lambda}{2}\|\theta\|^2)'$

$= g_t + \lambda\theta_{t-1}$

$\theta_t = \theta_{t-1} - \alpha(g_t + \lambda\theta_{t-1})$

$(1-\alpha\lambda)\theta_{t-1} - \alpha g_t$

1.001, 0.01, 0.1

L2: Loss function = $L(\theta) + \frac{\lambda}{2}\|\theta\|_2^2$

$g = g_t + \lambda\theta_{t-1}$

$\theta_t \leftarrow \theta_{t-1} - \alpha(g_t + \lambda\theta_{t-1})$

$\leftarrow (1-\alpha\lambda)\theta_{t-1} - \alpha g_t$

③ Early Stopping 早停

早停法通过在验证集误差不再下降时提前终止训练，防止模型在训练集上过度拟合，从而提升泛化能力.

④ Dropout

对于神经网络层 $y=f(Wx+b)$，引入函数 $d(\cdot)$，使得 $y=f(Wd(x)+b)$.

$$d(x)=\begin{cases} m\odot x & \text{训练阶段} \\ px & \text{测试阶段} \end{cases}$$

$y=f(Wx+b)$

$y=f(Wd(x)+b)$

每次训练的iteration都不一样

其中 $m\in\{0,1\}^d$ 是 dropout 掩码，以概率p按伯努利分布随机生成. P为"1"的概率，也就是保留的概率.

有效的原因

①集成学习：每次dropout，相当于采样一个子网络；n个神经元的网络可以采样出 2^n 个子网络.

②贝叶斯学习：$\mathbb{E}_{q(\theta)}[y]=\int_q f(x,\theta)q(\theta)d\theta \approx \frac{1}{M}\sum_{m=1}^{M} f(x,\theta_m)$ 经验采样的近似

（$f(x,\theta_m)$：第m次dropout后的网络）

⑤ Data augmentation

⑥ Label Smoothing 标签平滑

将硬标签(如one-hot中的1)替换为软标签($1\rightarrow 1-\varepsilon$，其余类别均分 ε)来防止模型过度自信，从而提升泛化能力.

$$y=[0,\cdots,0,1,0,\cdots,0]^T \rightarrow \tilde{y}=\left[\frac{\varepsilon}{K-1},\cdots,\frac{\varepsilon}{K-1},1-\varepsilon,\frac{\varepsilon}{K-1},\cdots,\frac{\varepsilon}{K-1}\right]^T$$

⑦ Batch Normalization (批归一化)

逐层归一化 —目标→
- ①更好的尺度不变性
- ②减少内部协变量偏移 (BN：对每个中间层进行标准化)
- ③更平滑的优化地形

方法举例 →
- Batch Normalization 批归一化
- Layer — 层归一化
- Weight 权重
- Local Response

BN：1. 在批次维度上进行归一化.
2. 依赖批次大小.
3. 常用于CNN
4. 稳定训练过程并降低CNN对初始参数的敏感性.

LN：1. 在单个样本维度进行归一化.
2. 不依赖于批量大小
3. 常用于RNNs (循环神经网络)，Transformer. 或批量大小变化的任务.
4. 在序列模型，批次统计不可靠时稳健.