深蓝学院
shenlanxueyuan.com

深度学习理论与实践

第四课：优化算法与参数调节

主讲人 郑元春

中科院大数据挖掘与知识管理重点实验室
中科院虚拟经济与数据科学研究中心

模型与风险

偏差与方差

评价指标

网络优化

代码讲解

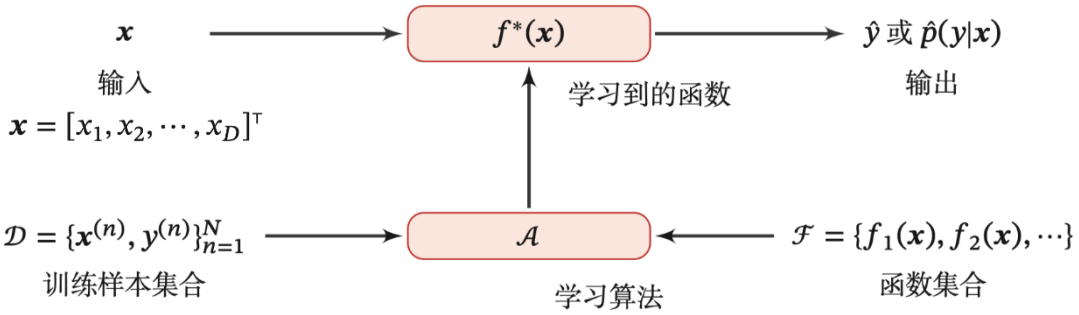
1. 模型与风险

什么是模型?

输入空间$\mathcal{X}$和输出空间$\mathcal{Y}$构成了一个样本空间。

对于样本空间中的样本$(\boldsymbol{x}, y)\in\mathcal{X}\times\mathcal{Y}$，假定$\boldsymbol{x}$和$y$之间的关系可以通过一个未知的真实映射函数$y=g(\boldsymbol{x})$或真实条件概率分布$Pr(y|\boldsymbol{x})$来描述。则机器学习的目标是找到一个模型来近似真实映射函数 $g(\boldsymbol{x})$ 或真实条件概率分布$Pr(y|\boldsymbol{x})$。

我们把从样本的特征$\boldsymbol{x}$映射到标签y的函数称为模型，或者说模型就是找到(训练)一个函数，这个函数能够从$\boldsymbol{x}$映射到y。



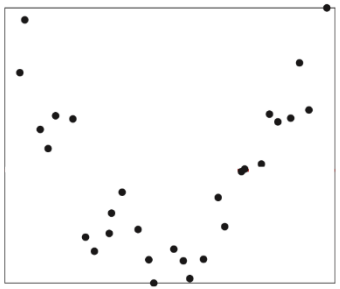
1. 模型与风险

什么是模型?

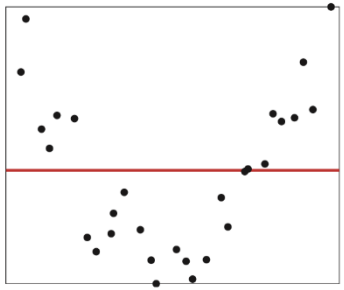
输入空间$\mathcal{X}$和输出空间$\mathcal{Y}$构成了一个样本空间。

对于样本空间中的样本$(x, y)\in\mathcal{X}\times\mathcal{Y}$，假定$x$和$y$之间的关系可以通过一个未知的真实映射函数$y=g(x)$或真实条件概率分布$Pr(y|x)$来描述。则机器学习的目标是找到一个模型来近似真实映射函数 $g(x)$ 或真实条件概率分布$Pr(y|x)$。

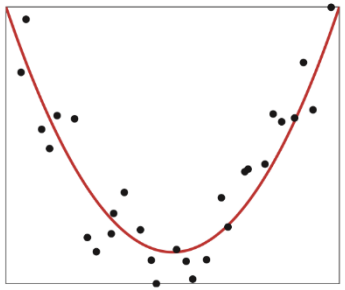
我们把从样本的特征x映射到标签y的函数称为模型，或者说模型就是找到(训练)一个函数，这个函数能够从x映射到y。



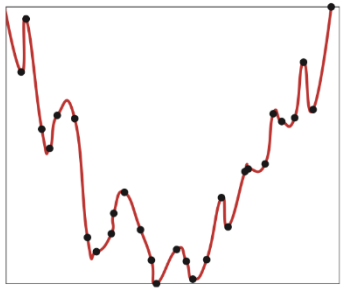
(a) 观察数据



(b) 拟合函数1



(c) 拟合函数2



(d) 拟合函数3

1. 模型与风险

模型的评价

对于相同的观察数据，存在着多个可以满足数据的模型(例如上一页中的图c和d)。从直观上来讲，我们会觉得图c的拟合函数2比较好，那么怎么定量的来衡量模型的好坏呢?

理想状态下，一个好的模型$f(\boldsymbol{x}, \theta^*)$应该在所有$(\boldsymbol{x}, y)$的可能取值上都与真实映射函数$y = g(\boldsymbol{x})$一致，或者是与真实条件概率分布$Pr(y|\boldsymbol{x})$一致，即：

$$|f(\boldsymbol{x}, \theta^*) - y| < \epsilon, \quad \forall(\boldsymbol{x}, y) \in x \times y$$

$$|f_y(\boldsymbol{x}, \theta^*) - p_r(y \mid \boldsymbol{x})| < \epsilon, \quad \forall(\boldsymbol{x}, y) \in x \times y$$

无法得到

期望风险(Expected Risk)：评估当前模型（也就是映射函数）在真实数据分布下的预测损失Loss的期望。前提是已知真实数据分布下的误差，那也就是说模型的真实误差。

在无法获得真实数据的时候，那么我们怎么来衡量模型的好坏呢?

1. 模型与风险

模型的评价

期望风险(Expected Risk)：评估当前模型（也就是映射函数）在真实数据分布下的预测损失Loss的期望。前提是已知真实数据分布下的误差，那也就是说模型的**真实误差**。

$$\mathcal{R}(\theta)^{exp} = \mathbb{E}_{(x,y)\sim p_r(x,y)}[\mathcal{L}(y, f(x;\theta))]$$

我们所能得到的观察数据是真实数据的一个真子集，因此可以利用模型在观察数据上的误差来近似反应模型在真实数据上的拟合能力，将这个误差称为**经验误差。**将在所有观察数据中得到的平均误差称为**经验风险(Empirical Risk)。**

$$\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$$

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(y^{(n)}, f\left(\boldsymbol{x}^{(n)};\theta\right)\right)$$

当数据集$|\mathcal{D}|$的大小趋向于无穷大的时候，经验风险就趋向于期望风险。

1. 模型与风险

模型的评价

期望风险(Expected Risk)：评估当前模型（也就是映射函数）在真实数据分布下的预测损失Loss的期望。前提是已知真实数据分布下的误差，那也就是说模型的**真实误差**。

$$\mathcal{R}(\theta)^{exp} = \mathbb{E}_{(x,y)\sim p_r(x,y)}[\mathcal{L}(y, f(x;\theta))]$$

经验风险最小化(Empirical Risk Minimization)就是找到一组参数θ^*使得经验风险最小：

$$\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$$

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(y^{(n)}, f\left(\boldsymbol{x}^{(n)};\theta\right)\right)$$

$$\theta^* = \arg\min_{\theta}\mathcal{R}_{\mathcal{D}}^{emp}(\theta)$$

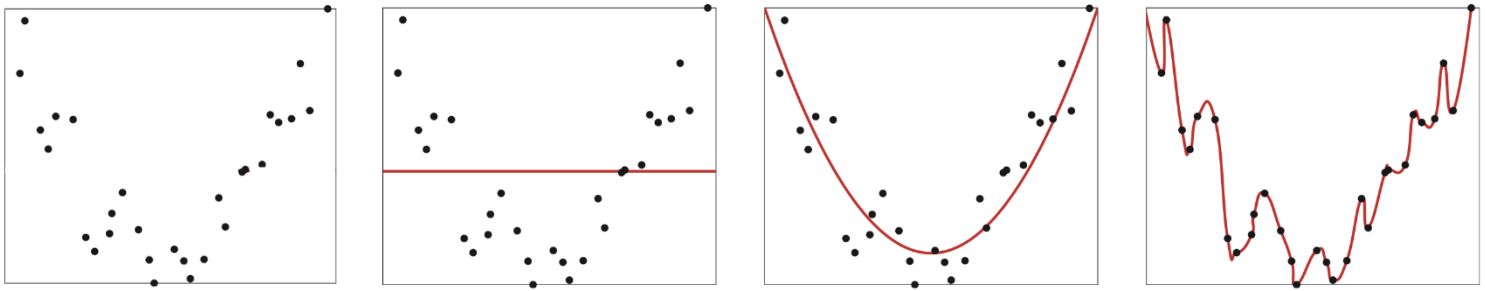
1. 模型与风险

模型的评价

经验风险最小化(Empirical Risk Minimization)就是找到一组参数θ^*使得经验风险最小：

$$\theta^* = \arg\min_{\theta} \mathcal{R}_{\mathcal{D}}^{emp}(\theta)$$

$L(y, f(x;\theta^1) = 1000$ $L(y, f(x;\theta^2) = 10$ $L(y, f(x;\theta^3) = 0$



(a)观察数据 (b)欠拟合 (c)较好拟合 (d)过拟合

1. 模型与风险

模型的评价

经验风险最小化就是找到一组参数θ^*使得经验风险最小：$\theta^* = \arg\min_{\theta} \mathcal{R}_{\mathcal{D}}^{emp}(\theta)$

过拟合是因为模型在训练数据集上拟合能力太强，反而对于新的测试数据表现不佳。根据**奥卡姆剃刀原则**需要限制模型的能力(模型的参数量)。

奥卡姆剃刀原则：如无必要，勿增实体。简单的模型泛化能力更好，如果有两个性能相近的模型，我们应该选择更简单的模型)。

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(b)欠拟合

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

(c)较好拟合

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \cdots)$

(d)过拟合

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}\left(y^{(n)}, f\left(\boldsymbol{x}^{(n)}; \theta\right)\right) + 1000\theta_5^2 + 1000\theta_6^2 + \cdots 1000\theta_n^2$$

那么该怎么限制经验风险，使模型具备较好的泛化能力？

1. 模型与风险

模型的评价

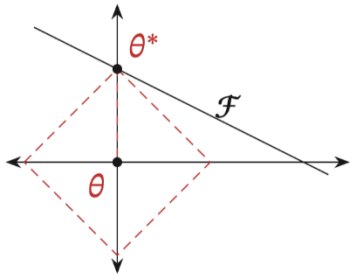
经验风险最小化就是找到一组参数θ^*使得经验风险最小：$\theta^* = \arg\min_{\theta} \mathcal{R}_{\mathcal{D}}^{emp}(\theta)$

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(y^{(n)}, f\left(\boldsymbol{x}^{(n)};\theta\right)\right) + \lambda\sum_{j=1}^{M}\theta_j^2$$

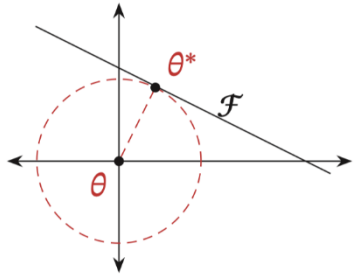
平衡经验风险与正则化项

结构风险= 经验风险 + 正则化

L1正则化：$L^1(\theta) = ||\theta||_1 = \sum|\theta_i|$



L2正则化：$L^2(\theta) = ||\theta||_2^2 = \sum\theta_i^2$



L1更加适合特征选择，L2更加适合防止过拟合

1. 模型与风险

模型的评价

经验风险最小化就是找到一组参数θ^*使得经验风险最小：$\theta^* = \arg\min_{\theta} \mathcal{R}_{\mathcal{D}}^{emp}(\theta)$

结构风险最小化(Structure Risk Minimization)就是找到一组参数θ^*使得结构风险最小：

$$\theta^* = \arg\min_{\theta} \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(y^{(n)}, f\left(\boldsymbol{x}^{(n)};\theta\right)\right) + \lambda\ell_p(\theta)$$

给定一个训练集，机器学习的目标是从假设空间中找到一个泛化误差较低的“理想”模型，以便更好地对未知的样本进行预测，特别是没有在训练集中出现的样本。因此，我们可以将机器学习看作一个从有限、高维、有噪声的数据上得到更一般性规律的泛化问题。

1. 模型与风险

模型的评价

用风险来评价模型的好坏，用最小化风险作为目标来指导模型的学习(参数的更新)。

理想情况

真实数据分布

$$|f(\boldsymbol{x},\theta^*)-y|<\epsilon,\quad \forall(\boldsymbol{x},y)\in x\times y$$

$$|f_y(\boldsymbol{x},\theta^*)-p_r(y\mid\boldsymbol{x})|<\epsilon,\quad \forall(\boldsymbol{x},y)\in X\times y$$

现实情况

只有观察数据

$$\mathcal{D}=\left\{\left(\boldsymbol{x}^{(n)},y^{(n)}\right)\right\}_{n=1}^{N}$$

现实情况

只有观察数据+泛化能力

$$\mathcal{D}=\left\{\left(\boldsymbol{x}^{(n)},y^{(n)}\right)\right\}_{n=1}^{N}$$

期望风险 --- 实际化 --- 经验风险 --- 实际化+正则化 --- 结构风险

$$\mathcal{R}(\theta)^{exp}=\mathbb{E}_{(x,y)\sim p_r(x,y)}[\mathcal{L}(y,f(x;\theta))]$$

$$\mathcal{R}_{\mathcal{D}}^{emp}(\theta)=\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(y^{(n)},f\left(\boldsymbol{x}^{(n)};\theta\right)\right)$$

$$\theta^*=\arg\min_{\theta}\mathcal{R}_{\mathcal{D}}^{emp}(\theta)$$

$$\mathcal{R}_{\mathcal{D}}^{stru}(\theta)=\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(y^{(n)},f\left(\boldsymbol{x}^{(n)};\theta\right)\right)+\lambda\ell_p(\theta)$$

$$\theta^*=\arg\min_{\theta}\mathcal{R}_{\mathcal{D}}^{stru}(\theta)$$

1. 模型与风险

损失函数

损失函数$\mathcal{L}(y, f(x;\theta)) \to \mathcal{R}$是一个非负实数函数，用来量化模型预测和真实标签之间的差异。

损失函数类型	表达式	适用范围
0-1损失函数 (0-1 Loss Function)	$\mathcal{L}(y, f(x;\theta)) = \begin{cases} 0 & \text{if } y = f(x;\theta) \\ 1 & \text{if } y \neq f(x;\theta) \end{cases}$	虽然0-1损失函数能够客观地评价模型的好坏，但其缺点是数学性质不好，即不连续且导数为0，难以优化。 因此经常用连续可微的损失函数替代。
平方损失函数 (Quadratic Loss Function)	$\mathcal{L}(y, f(x;\theta)) = \frac{1}{2}(y - f(x;\theta))^2$	经常用在预测标签y为实数值的任务中，一般不适用于分类问题。
交叉熵损失函数 (Cross-Entropy Loss Function)	$\mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x};\theta)) = -\boldsymbol{y}^{\top} \log f(\boldsymbol{x};\theta) = -\sum_{c=1}^{C} y_c \log f_c(\boldsymbol{x};\theta)$	一般用于分类问题，衡量两个概率分布的差异。
Hinge损失函数 (Hinge Loss Function)	$\mathcal{L}(y, f(x;\theta)) = \max(0, 1 - yf(x;\theta))$	通常被用于最大间隔算法，专用于二分类问题。

模型与风险

偏差与方差

评价指标

网络优化

代码讲解

2. 偏差与方差

模型平衡

为了避免过拟合，我们经常会在**模型的拟合能力和复杂度之间进行权衡。**

拟合能力强的模型一般复杂度会比较高，容易导致过拟合。相反，如果限制模型的复杂度，降低其拟合能力，又可能会导致欠拟合。因此，**如何在模型的拟合能力和复杂度之间取得一个较好的平衡，对一个机器学习算法来讲十分重要。**

偏差-方差分解(Bias-Variance Decomposition)为我们提供了一个很好的分析和指导工具。

回归问题使用平方损失函数，模型$f(\boldsymbol{x})$的期望误差(expected error)

样本真实分布

$$\mathcal{R}(f) = \mathbb{E}_{(x,y)\sim p_r(x,y)}\left[(y - f(x))^2\right]$$

最优模型

样本真实条件分布

$$f^*(x) = \mathbb{E}_{y\sim p_r(y|x)}[y]$$

最优模型$f^*(\boldsymbol{x})$的损失

$$\epsilon = \mathbb{E}_{(x,y)\sim p_r(x,y)}\left[(y - f^*(x))^2\right]$$

损失ϵ通常由样本分布以及噪声引起的，无法通过优化模型来减少

2. 偏差与方差

模型平衡

为了避免过拟合，我们经常会在**模型的拟合能力和复杂度之间进行权衡。**

拟合能力强的模型一般复杂度会比较高，容易导致过拟合。相反，如果限制模型的复杂度，降低其拟合能力，又可能会导致欠拟合。因此，**如何在模型的拟合能力和复杂度之间取得一个较好的平衡，对一个机器学习算法来讲十分重要。**

偏差-方差分解(Bias-Variance Decomposition)为我们提供了一个很好的分析和指导工具。

回归问题使用平方损失函数，模型$f(\boldsymbol{x})$的期望误差

$$\mathcal{R}(f) = \mathbb{E}_{(x,y)\sim p_r(x,y)}\left[(y - f(x))^2\right]$$

期望误差分解

$$\begin{aligned}\mathcal{R}(f) &= \mathbb{E}_{(x,y)\sim p_r(x,y)}\left[(y - f^*(\boldsymbol{x}) + f^*(\boldsymbol{x}) - f(\boldsymbol{x}))^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_r(\boldsymbol{x})}\left[(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2\right] + \epsilon\end{aligned}$$

当前模型和最优模型之间的差距，是机器学习算法可以优化的真实目标

2. 偏差与方差

模型平衡

在实际训练一个模型$f(\boldsymbol{x})$时，训练集$\mathcal{D}$是从真实分布$Pr(\boldsymbol{x}, y)$上独立同分布地采样出来的有限样本集合。不同的训练集会得到不同的模型。

令$f_{\mathcal{D}}(\boldsymbol{x})$表示在训练集$\mathcal{D}$上学习到的模型，一个机器学习算法（包括模型以及优化算法）的能力可以用不同训练集上的模型的平均性能来评价。

对于单个样本$\boldsymbol{x}$，不同训练集$\mathcal{D}$得到模型$f_{\mathcal{D}}(\boldsymbol{x})$与最优模型 $f^*(\boldsymbol{x})$的期望误差为：

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[\left(f_{\mathcal{D}}(\boldsymbol{x})-f^*(\boldsymbol{x})\right)^2\right] &= \mathbb{E}_{\mathcal{D}}\left[\left(f_{\mathcal{D}}(\boldsymbol{x})-\mathbb{E}_{\mathcal{D}}\left[f_{\mathcal{D}}(\boldsymbol{x})\right]+\mathbb{E}_{\mathcal{D}}\left[f_{\mathcal{D}}(\boldsymbol{x})\right]-f^*(\boldsymbol{x})\right)^2\right] \\ &= \underbrace{\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathcal{D}}(\boldsymbol{x})\right]-f^*(\boldsymbol{x})\right)^2}_{(\text{bias. }\mathbf{x})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(f_{\mathcal{D}}(\boldsymbol{x})-\mathbb{E}_{\mathcal{D}}\left[f_{\mathcal{D}}(\boldsymbol{x})\right]\right)^2\right]}_{\text{variance.x}}\end{aligned}$$

2. 偏差与方差

模型平衡

在实际训练一个模型$f(\boldsymbol{x})$时，训练集$\mathcal{D}$是从真实分布$Pr(y|\boldsymbol{x})$上独立同分布地采样出来的有限样本集合。不同的训练集会得到不同的模型。

令$f_{\mathcal{D}}(\boldsymbol{x})$表示在训练集$\mathcal{D}$上学习到的模型，一个机器学习算法（包括模型以及优化算法）的能力可以用不同训练集上的模型的平均性能来评价。

用 $\mathbb{E}_{\mathcal{D}}\left[\left(f_{\mathcal{D}}(\boldsymbol{x})-f^{*}(\boldsymbol{x})\right)^{2}\right]$ 代替 $\mathcal{R}(f)$ 中的 $\left(f(\boldsymbol{x})-f^{*}(\boldsymbol{x})\right)^{2}$，那么期望误差可以进一步写为：

$$
\begin{aligned}
\mathcal{R}(f) &= \mathbb{E}_{\boldsymbol{x}\sim p_r(\boldsymbol{x})}\left[\left(f(\boldsymbol{x})-f^{*}(\boldsymbol{x})\right)^{2}\right]+\epsilon \\
&= (\text{ bias })^{2}+\text{ variance }+\epsilon
\end{aligned}
$$

$$
\begin{aligned}
(\text{ bias })^{2} &= \mathbb{E}_{x}\left[\left(\mathbb{E}_{\mathcal{D}}\left[f_{\mathcal{D}}(\boldsymbol{x})\right]-f^{*}(\boldsymbol{x})\right)^{2}\right] \\
\text{variance } &= \mathbb{E}_{x}\left[\mathbb{E}_{\mathcal{D}}\left[\left(f_{\mathcal{D}}(\boldsymbol{x})-\mathbb{E}_{\mathcal{D}}\left[f_{\mathcal{D}}(\boldsymbol{x})\right]\right)^{2}\right]\right]
\end{aligned}
$$

2. 偏差与方差

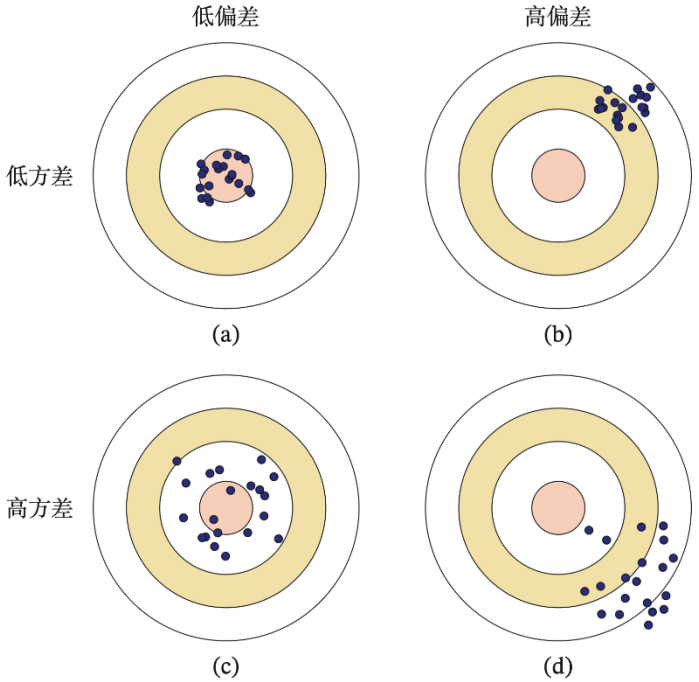
模型平衡

(a)图为一种理想情况，方差和偏差都比较低；

(b)图为高偏差低方差的情况，表示模型的泛化能力很好，但拟合能力不足；

(c)图为低偏差高方差的情况，表示模型的拟合能力很好，但泛化能力比较差，当训练数据比较少时会导致过拟合；

(d)图为高偏差高方差的情况，是一种最差的情况。



机器学习模型的四种偏差和方差组合情况

每个图的中心点为最优模型 $f^*(\boldsymbol{x})$；

每个点代表在不同训练集上训练得到的模型 $f(\boldsymbol{x})$ 的表现情况。

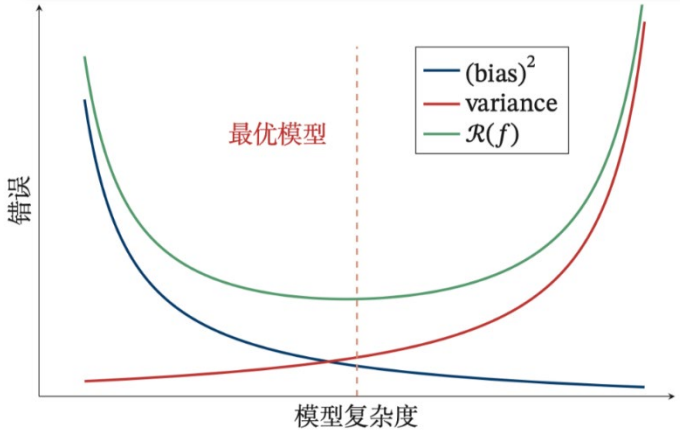
2. 偏差与方差

模型平衡

方差一般会随着训练样本的增加而减少。当样本比较多时，方差比较少，这时可以选择能力强的模型来减少偏差。然而，在很多机器学习任务上，**训练集往往都比较有限，最优的偏差和最优的方差就无法兼顾。**

随着模型复杂度的增加，模型的拟合能力变强，偏差减少而方差增大，从而导致过拟合。以结构风险最小化为例，我们可以调整正则化系数 λ 来控制模型的复杂度。

最优模型并不一定是偏差曲线和方差曲线的交点



模型的期望误差、偏差和方差随复杂度的变化情况

- 当 λ 变大时，模型复杂度会降低，可以有效地减少方差，避免过拟合，但偏差会上升。
- 当 λ 过大时，总的期望误差反而会上升。因此，一个好的正则化系数 λ 需要在偏差和方差之间取得比较好的平衡。

模型与风险

偏差与方差

评价指标

网络优化

代码讲解

3. 评价指标

数据集划分

在机器学习中选择不同的模型或在深度学习中选择不同的网络构建，对最终的结果影响很大，那么在某个数据集D上把模型训练出来之后，该怎么对模型进行验证呢？(也就是说，怎么知道训练出来的模型好不好)

- 把数据集D全部作为训练集，然后用这个训练集训练模型，用训练集验证模型。

  (选择训练集误差最小的模型)

- 把数据集D随机分为训练集和测试集，训练集训练模型，测试集验证模型。

  (选择测试集误差最小的模型)

- 把数据集D分为训练集、验证集和测试集，训练集训练模型，验证集验证模型，根据情况不断调整模型，选择出最好的模型。再用训练集和验证集训练出一个最终的模型，最后用测试集评估最终的模型。

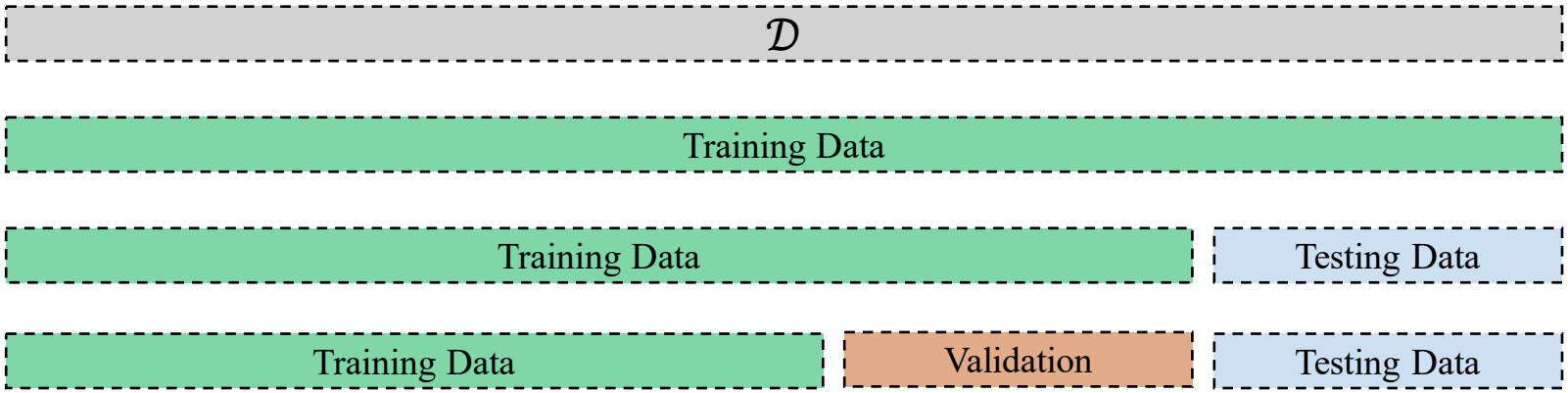
- 交叉验证把原始数据集平均分为K组不重复的子集，每次选择K-1组子集作为训练集，剩下一组子集作为验证集。

  (K次实验得到K个模型，将K个模型在各自验证集上的错误率的平均作为分类器的评价)

3. 评价指标

数据集划分

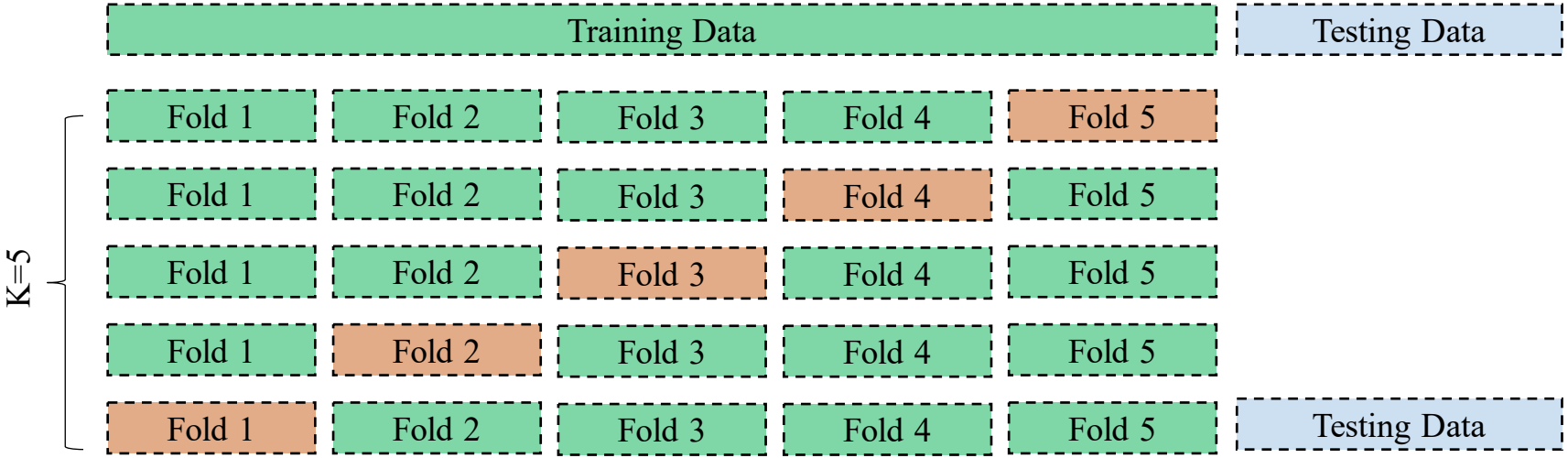
在机器学习中选择不同的模型或在深度学习中选择不同的网络构建，对最终的结果影响很大，那么在某个数据集$\mathcal{D}$上把模型训练出来之后，该怎么对模型进行验证呢？(也就是说，怎么知道训练出来的模型好不好)



3. 评价指标

数据集划分

在机器学习中选择不同的模型或在深度学习中选择不同的网络构建，对最终的结果影响很大，那么在某个数据集$\mathcal{D}$上把模型训练出来之后，该怎么对模型进行验证呢？(也就是说，怎么知道训练出来的模型好不好)



3. 评价指标

数据集划分

在机器学习中选择不同的模型或在深度学习中选择不同的网络构建，对最终的结果影响很大，那么在某个数据集$\mathcal{D}$上把模型训练出来之后，该怎么对模型进行验证呢？(也就是说，怎么知道训练出来的模型好不好)

- **训练集：**用于训练模型；
- **测试集：**用于评估模型的最终能力，在整个训练过程中并没有用这一部分数据；
- **验证集：**用于调整和选择模型。

传统的机器学习模型：倾向于使用K折交叉来完成模型选择与超参数的选取；

神经网络模型：由于采用梯度下降方法，会随机从数据集$\mathcal{D}$中划分训练集和测试集，并使用不同batch size的数据集合进行训练，类似第二种数据集划分方法。

3. 评价指标

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有**准确率、精确率、召回率和F值**等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x ; \theta^{*})$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy)	错误率 (Error Rate)	精确率 (Precision)	召回率 (Recall)	F值 (F Measure)	宏平均 (Macro Average)	微平均 (Micro Average)

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy) | **错误率 (Error Rate)** | **精确率 (Precision)** | **召回率 (Recall)** | **F值 (F Measure)** | **宏平均 (Macro Average)** | **微平均 (Micro Average)**

$$\mathcal{A}=\frac{1}{N} \sum_{n=1}^{N} I\left(y^{(n)}=\hat{y}^{(n)}\right)$$

$$y=[0, \quad 1, \quad 1, \quad 1, \quad 0, \quad 1, \quad 0, \quad 1, \quad 1, \quad 0]$$
$$\hat{y}=[1, \quad 1, \quad 1, \quad 1, \quad 1, \quad 0, \quad 0, \quad 0, \quad 1, \quad 0]$$

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy)	错误率 (Error Rate)	精确率 (Precision)	召回率 (Recall)	F值 (F Measure)	宏平均 (Macro Average)	微平均 (Micro Average)

$$\begin{aligned}\epsilon &= 1-\mathcal{A} \\ &= \frac{1}{N}\sum_{n=1}^{N} I\left(y^{(n)} \neq \hat{y}^{(n)}\right)\end{aligned}$$

$$\begin{aligned} y &= [0, \quad 1, \quad 1, \quad 1, \quad 0, \quad 1, \quad 0, \quad 1, \quad 1, \quad 0] \\ \hat{y} &= [1, \quad 1, \quad 1, \quad 1, \quad 1, \quad 0, \quad 0, \quad 0, \quad 1, \quad 0] \end{aligned}$$

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy)	错误率 (Error Rate)	精确率 (Precision)	召回率 (Recall)	F值 (F Measure)	宏平均 (Macro Average)	微平均 (Micro Average)

准确率和错误率是在所有类别上整体的性能平均，由于数据类别的分布不平衡性，例如正负比例为9:1，那么模型只需要将所有样本全部分类为正例，也能获得90%的准确率。

如果希望对每个类别都进行性能的评估，就需要计算精确率和召回率。

3. 评价指标

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为$\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy) | **错误率 (Error Rate)** | **精确率 (Precision)** | **召回率 (Recall)** | **F值 (F Measure)** | **宏平均 (Macro Average)** | **微平均 (Micro Average)**

以类别c为例，模型在测试集上的分类结构可以分为以下四种情况：

真正例(True Positive, TP) $TP_c=\sum_{n=1}^{N} I\left(y^{(n)}=\hat{y}^{(n)}=c\right)$

假正例(False Positive, FP) $FP_c=\sum_{n=1}^{N} I\left(y^{(n)} \neq c \quad \& \quad \hat{y}^{(n)}=c\right)$

假负例(False Negative, FN) $FN_c=\sum_{n=1}^{N} I\left(y^{(n)}=c \quad \& \quad \hat{y}^{(n)} \neq c\right)$

真负例(True Negative, TN) $TN_c=\sum_{n=1}^{N} I\left(y^{(n)} \neq c \quad \& \quad \hat{y}^{(n)} \neq c\right)$

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy) | **错误率 (Error Rate)** | **精确率 (Precision)** | **召回率 (Recall)** | **F值 (F Measure)** | **宏平均 (Macro Average)** | **微平均 (Micro Average)**

这四种情况可以用混淆矩阵来表示：

		预测类别	
		$\hat{y}=c$	$\hat{y}\neq c$
真实类别	$y=c$	TP_c	FN_c
	$y\neq c$	FP_c	TN_c

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)},y^{(1)}\right),\cdots,\left(\boldsymbol{x}^{(N)},y^{(N)}\right)\right\}$真实的标签记为 $y^{(n)}\in\{1,\cdots,C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为$\left\{\hat{y}^{(1)},\cdots,\hat{y}^{(N)}\right\}$。

准确率 (Accuracy) | **错误率 (Error Rate)** | **精确率 (Precision)** | **召回率 (Recall)** | **F值 (F Measure)** | **宏平均 (Macro Average)** | **微平均 (Micro Average)**

精确率(也叫精度或是查准率)，类别c的精确率是所有预测为c的样本中预测正确的比例：

$$p_c=\frac{TP_c}{TP_c+FP_c}$$

		预测类别	
		$\hat{y}=c$	$\hat{y}\neq c$
真实类别	$y=c$	TP_c	FN_c
	$y\neq c$	FP_c	TN_c

3. 评价指标

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy) | **错误率 (Error Rate)** | **精确率 (Precision)** | **召回率 (Recall)** | **F值 (F Measure)** | **宏平均 (Macro Average)** | **微平均 (Micro Average)**

召回率(也叫查全率)，是所有真实标签为c的样本中预测正确的比例：

$$r_c=\frac{TP_c}{TP_c+FN_c}$$

		预测类别	
		$\hat{y}=c$	$\hat{y}\neq c$
真实类别	$y=c$	TP_c	FN_c
	$y\neq c$	FP_c	TN_c

3. 评价指标

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)},y^{(1)}\right),\cdots,\left(\boldsymbol{x}^{(N)},y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)}\in\{1,\cdots,C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)},\cdots,\hat{y}^{(N)}\right\}$。

准确率 (Accuracy) | **错误率 (Error Rate)** | **精确率 (Precision)** | **召回率 (Recall)** | **F值 (F Measure)** | **宏平均 (Macro Average)** | **微平均 (Micro Average)**

F值是一个综合指标，为精确率和召回率的调和平均：

$$f_c=\frac{(1+\beta^2)\times p_c\times r_c}{\beta^2\times p_c+r_c}$$

$$F_1=f_c(\beta=1)$$

		预测类别	
		$\hat{y}=c$	$\hat{y}\neq c$
真实类别	$y=c$	TP_c	FN_c
	$y\neq c$	FP_c	TN_c

3. 评价指标

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy) | **错误率 (Error Rate)** | **精确率 (Precision)** | **召回率 (Recall)** | **F值 (F Measure)** | **宏平均 (Macro Average)** | **微平均 (Micro Average)**

为了计算分类算法在所有类别上的总体精确率、召回率和F_1值，经常使用两种平均方法，分别称为宏平均和微平均。

3. 评价指标

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$ 真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$ 对测试集中的每一个样本进行预测，预测结果为 $\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy)	错误率 (Error Rate)	精确率 (Precision)	召回率 (Recall)	F值 (F Measure)	**宏平均 (Macro Average)**	微平均 (Micro Average)

宏平均是每一类的性能指标的算术平均值：

$$p_{\text{macro}}=\frac{1}{C} \sum_{c=1}^{C} p_c \quad r_{\text{macro}}=\frac{1}{C} \sum_{c=1}^{C} r_c \quad F_{1 \text{macro}}=\frac{2 \times p_{\text{macro}} \times r_{\text{macro}}}{p_{\text{macro}}+r_{\text{macro}}}.$$

3. 评价指标

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x ; \theta^{*})$ 对测试集中的每一个样本进行预测，预测结果为$\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy)	错误率 (Error Rate)	精确率 (Precision)	召回率 (Recall)	F值 (F Measure)	宏平均 (Macro Average)	微平均 (Micro Average)

微平均是每一个样本的性能指标的算术平均值。对于单个样本而言，它的精确率和召回率是相同的(要么都是1，要么都是0)。因此精确率的微平均和召回率的微平均是相同的。同理，F1 值的微平均指标是相同的。

当不同类别的样本数量不均衡时，使用宏平均更合理些。宏平均会更关注小类别上的评价指标。

3. 评价指标

评价指标

已经确定了数据集的划分方法。那么在测试集上是用什么指标来反映模型的预测/分类能力呢？对于分类问题，常见的评价标准有准确率、精确率、召回率和F值等。

给定**测试集** $\mathcal{J}=\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots,\left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$真实的标签记为 $y^{(n)} \in\{1, \cdots, C\}$,最终选择的模型 $f(x;\theta^*)$对测试集中的每一个样本进行预测，预测结果为$\left\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\right\}$。

准确率 (Accuracy)	错误率 (Error Rate)	精确率 (Precision)	召回率 (Recall)	F值 (F Measure)	宏平均 (Macro Average)	微平均 (Micro Average)

在实际应用中，我们也可以通过调整分类模型的阈值来进行更全面的评价，比如AUC(Area Under Curve)、ROC(Receiver Operating Characteristic)曲线、PR(Precision-Recall)曲线等。此外，很多任务还有自己专门的评价方式，比如TopN准确率。

《PR,ROC,AUC的拓展材料》

模型与风险

偏差与方差

评价指标

网络优化

代码讲解

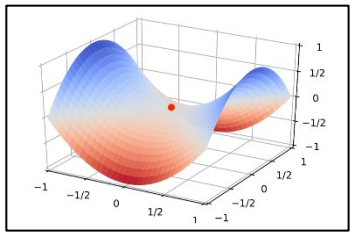
4. 网络优化

网络优化的难点

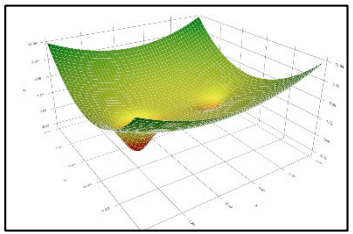
1. 网络结构的多样性，很难找到一种通用的优化方法，不同优化方法在不同网络结构上的表现也有比较大的差异。

2. 低维空间的非凸优化问题主要是存在一些局部最优点。基于梯度下降的优化方法会陷入局部最优点，因此在低维空间中非凸优化的主要难点是如何选择初始化参数和逃离局部最优点。

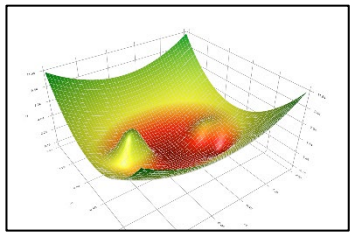
3. 深度神经网络的参数非常多，其参数学习是高维空间中的非凸优化问题，其挑战和在低维空间中的非凸优化问题有所不同。



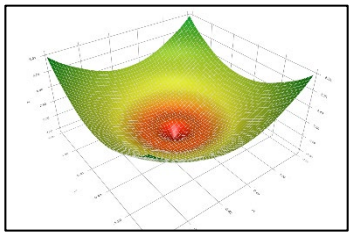
鞍点



局部极小值



局部山区



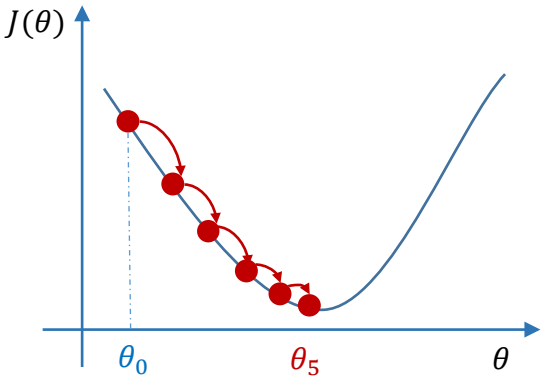
平台区域

4. 网络优化

梯度下降算法

梯度下降算法是最常用的优化算法之一，也是迄今为止优化神经网络最常用的方法之一。每一个先进的深度学习库都包含了各种梯度下降算法的实现。然而，这些算法经常被用作黑箱优化器，因为它们的优点和缺点很难得到实际的解释。

梯度下降是一种通过在目标函数梯度$\Delta_\theta J(\theta)$的反方向上更新参数来最小化目标函数$J(\theta)$的方法。实际使用中，通过一阶导数来寻找下降方向，并通过迭代的方式来更新参数。



梯度下降动画演示

思考：为什么参数沿着目标函数的一阶导数负方向更新，目标函数值就减小呢？

参考视频：梯度下降基础

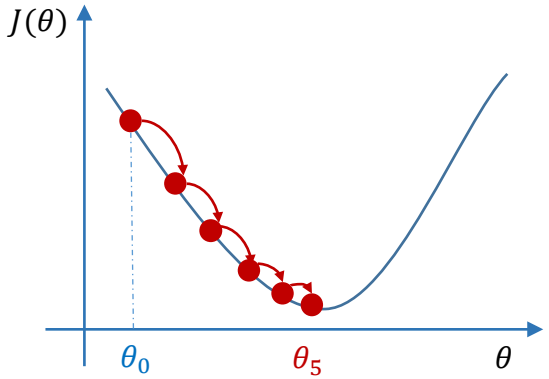
4. 网络优化

梯度下降算法

梯度下降是一种通过在目标函数梯度$\Delta_\theta J(\theta)$的反方向上更新参数来最小化目标函数$J(\theta)$的方法。实际使用中，通过一阶导数来寻找下降方向，并通过迭代的方式来更新参数。

依据每次更新参数时使用的数据量大小，权衡时间和精度之后，有3种不同的梯度下降算法实现：

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-Batch Gradient Descent



梯度下降动画演示

4. 网络优化

1. Batch Gradient Descent

批量梯度下降(Batch Gradient Descent)又叫Vanilla Gradient Descent，每次训练更新模型时采用的是整个训练集合的所有样本点。

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

```
for i in range(n_epochs):
    params_grad = evaluate_gradient(loss_function, data, params)
    params = params - lr * params_grad
```

优点：

- 每次更新朝着正确的方向进行
- 保证收敛于极值点，凸函数收敛于全局极值点

缺点：

- 学习时间太长
- 消耗大量内存

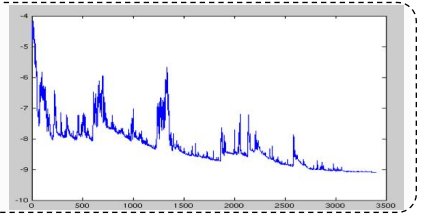
4. 网络优化

2. Stochastic Gradient Descent

随机梯度下降(Stochastic Gradient Descent)是对每个训练样本$\{x^{(i)}, y^{(i)}\}$进行参数更新。因为批量梯度下降在每次参数更新前重新计算类似样本的梯度，因此会对大数据集会有一些冗余的计算。而随机梯度下降由于每次只选取一个样本，所以会消除这种冗余。

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; \{x^{(i)}, y^{(i)}\})$$

```
for i in range(n_epochs):
    np.random.shuffle(data)
    for example in data:
        params_grad = evaluate_gradient(loss_function, example, params)
        params = params - lr * params_grad
```



优点：

- 运行速度快，允许在线更新模型
- 会跳出局部极小值点到另一个更好的局部极小值点

缺点：

- 每次更新可能并不按照正确的方向进行
- 以较大的方差频繁地更新，目标函数剧烈波动

4. 网络优化

3. Mini-Batch Gradient Descent

小批量梯度下降(Mini-Batch Stochastic Gradient Descent) 是在每次更新模型时使用的样本量做了权衡，即每次用多个样本组成的数据集来计算梯度并更新。

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i;i+n)}, y^{(i;i+n)})$$

```
for i in range(n_epochs):
    np.random.shuffle(data)
    for batch in get_batches(data, batch_size=50):
        params_grad = evaluate_gradient(loss_function, batch, params) / batch_size
        params = params - lr * params_grad
```

优点：

- 降低了参数更新的方差，获得更加稳定的收敛
- 利用深度学习库中常用的高度优化的矩阵优化方法

缺点：

- 学习率的选择困难
- 会陷入无限次的局部极小值，或鞍点

4. 网络优化

名词解释

在深度学习中，总是提到三个不同的概念，都是用于对训练数据的分割。对应着求梯度的步骤，这里统一进行名词的说明。

Epoch	使用训练集全部数据对模型进行一次完整的训练，称为“一代训练”。
Batch	使用训练集中一小部分样本利用MBGD更新一次参数，这部分样本被称为“一批数据”。
Iteration	使用一个Batch的数据对模型进行一次参数更新的过程，称为“一次训练”。

实际上，前面说到的3种不同梯度下降算法的根本区别就在于Batch-size的不同。

梯度下降方法	训练集大小	Batch-size	Batches数目
BGD	N	N	1
SGD	N	1	N
Mini-Batch	N	B	N/B + 1

4. 网络优化

Mini-Batch Gradient Descent

令 $f(\boldsymbol{x};\ \theta)$ 表示一个深度神经网络，θ为网络参数，在使用小批量梯度下降进行优化时，每次选取K个训练样本 $\mathcal{S}t$ = {($\boldsymbol{x}(k)$, $\boldsymbol{y}(k)$)}(k=1...K)。第 t 次迭代时损失函数关于参数 θ 的偏导数为：

$$\mathfrak{g}_t(\theta) = \frac{1}{K} \sum_{(x,y)\in \S_t} \frac{\partial \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x};\theta))}{\partial \theta}$$

第t次更新的梯度g_t用来更新模型参数：

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

$$g_t \triangleq \mathfrak{g}_t\left(\theta_{t-1}\right)$$

每次迭代时参数更新的差值$\Delta\theta_t$定义为：

$$\Delta\theta_t \triangleq \theta_t - \theta_{t-1}$$

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

1. 批量大小选择

一般而言，批量大小不影响随机梯度的期望，但是会影响随机梯度的方差。批量大小越大，随机梯度的方差越小，引入的噪声也越小，训练也越稳定，因此可以设置较大的学习率。而批量大小较小时，需要设置较小的学习率，否则模型会不收敛。

线性缩放规则(Linear Scaling Rule)：当批量大小增加m倍时，学习率也增加m倍。线性缩放规则往往在批量大小比较小时适用，当批量大小非常大时，线性缩放会使得训练不稳定。

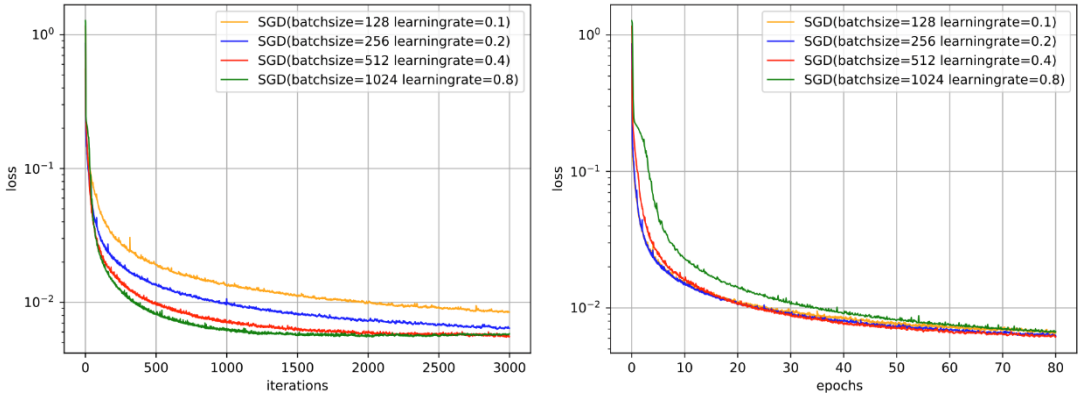
4. 网络优化

Mini-Batch Gradient Descent

1. 批量大小选择

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计



- 左边图为按迭代的损失变化，批量大小越大，下降效果越明显，下降曲线越平滑。
- 右边图为按回合的损失变化，批量大小越小，适当小的批量大小会导致更快的收敛。

此外，批量大小和模型的泛化能力也有一定的关系：批量越大，越有可能收敛到尖锐最小值；批量越小，越有可能收敛到平坦最小值。

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

从经验上看，学习率在刚开始的时候需要设置大一点保证收敛的速度，在收敛到最优点附近时要小一点避免来回震荡。令学习率的衰减方式设置为按迭代次数进行衰减，假设初始化学习率为α_0，在第t次迭代时学习率为α_t。

分段常数衰减
(Piecewise Constant Decay)

逆时衰减
(Inverse Time Decay)

指数衰减
(Exponential Decay)

自然指数衰减
(Natural Exponential Decay)

余弦衰减
(Cosine Decay)

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

从经验上看，学习率在刚开始的时候需要设置大一点保证收敛的速度，在收敛到最优点附近时要小一点避免来回震荡。令学习率的衰减方式设置为按迭代次数进行衰减，假设初始化学习率为α_0，在第t次迭代时学习率为α_t。

分段常数衰减
(Piecewise Constant Decay)

逆时衰减
(Inverse Time Decay)

指数衰减
(Exponential Decay)

自然指数衰减
(Natural Exponential Decay)

余弦衰减
(Cosine Decay)

每经过$T_1, T_2, \cdots, T_m$次迭代将学习率衰减为原来的$\beta_1, \beta_2, \cdots, \beta_m$，其中$T_m$和$\beta_m < 1$为根据经验设置的超参数。分段常数衰减也称为阶梯衰减(Step Decay) 。

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

从经验上看，学习率在刚开始的时候需要设置大一点保证收敛的速度，在收敛到最优点附近时要小一点避免来回震荡。令学习率的衰减方式设置为按迭代次数进行衰减，假设初始化学习率为α_0，在第t次迭代时学习率为α_t。

分段常数衰减 (Piecewise Constant Decay)　**逆时衰减** (Inverse Time Decay)　**指数衰减** (Exponential Decay)　**自然指数衰减** (Natural Exponential Decay)　**余弦衰减** (Cosine Decay)

$$\alpha_t = \alpha_0 \frac{1}{1 + \beta \times t}$$

衰减率

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

从经验上看，学习率在刚开始的时候需要设置大一点保证收敛的速度，在收敛到最优点附近时要小一点避免来回震荡。令学习率的衰减方式设置为按迭代次数进行衰减，假设初始化学习率为α_0，在第t次迭代时学习率为α_t。

分段常数衰减 (Piecewise Constant Decay)　**逆时衰减** (Inverse Time Decay)　**指数衰减** (Exponential Decay)　**自然指数衰减** (Natural Exponential Decay)　**余弦衰减** (Cosine Decay)

$$\alpha_t = \alpha_0 \beta^t$$

β<1衰减率

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

从经验上看，学习率在刚开始的时候需要设置大一点保证收敛的速度，在收敛到最优点附近时要小一点避免来回震荡。令学习率的衰减方式设置为按迭代次数进行衰减，假设初始化学习率为α_0，在第t次迭代时学习率为α_t。

分段常数衰减 (Piecewise Constant Decay)　**逆时衰减** (Inverse Time Decay)　**指数衰减** (Exponential Decay)　**自然指数衰减** (Natural Exponential Decay)　**余弦衰减** (Cosine Decay)

$$\alpha_t = \alpha_0 \exp(-\beta \times t)$$

β衰减率

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

从经验上看，学习率在刚开始的时候需要设置大一点保证收敛的速度，在收敛到最优点附近时要小一点避免来回震荡。令学习率的衰减方式设置为按迭代次数进行衰减，假设初始化学习率为α_0，在第t次迭代时学习率为α_t。

分段常数衰减 (Piecewise Constant Decay)　**逆时衰减** (Inverse Time Decay)　**指数衰减** (Exponential Decay)　**自然指数衰减** (Natural Exponential Decay)　**余弦衰减** (Cosine Decay)

$$\alpha_t = \frac{1}{2}\alpha_0 \left(1 + \cos\left(\frac{t\pi}{T}\right)\right)$$

T：总的迭代次数

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

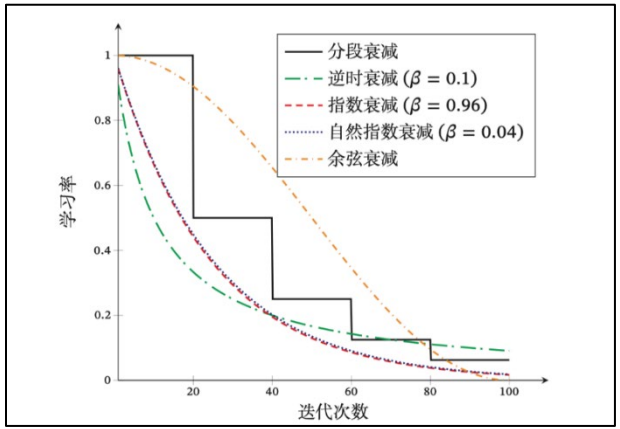
分段常数衰减
(Piecewise Constant Decay)

逆时衰减
(Inverse Time Decay)

指数衰减
(Exponential Decay)

自然指数衰减
(Natural Exponential Decay)

余弦衰减
(Cosine Decay)



- 学习率随着迭代次数逐渐降低；
- 分段衰减方法是阶梯状的，需要设置多个超参数，其他衰减方法是函数式的。

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

训练刚开始的时候模型的参数是随机设置的，因此在使用Mini-Batch梯度下降法时，将学习率设置较大的情况下梯度也往往比较大，使得训练不稳定。为了提高训练的稳定性，可以在最初几轮的时候，采用比较小的学习率，等梯度下降到一定程度之后再恢复到初始的学习率，这种方法称为学习率预热。

$$\alpha'_t = \frac{1}{T'}\alpha_0 \qquad (1 < t < T')$$

总的预热迭代次数

当预热过程结束，再选择一种学习率衰减方法来逐渐降低学习率。

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

为了使得梯度下降法能够逃离鞍点或尖锐最小值，一种经验性的方式是在训练过程中周期性地增大学习率。当参数处于尖锐最小值附近时，增大学习率有助于逃离尖锐最小值；当参数处于平坦最小值附近时，增大学习率依然有可能在该平坦最小值的吸引域(Basin of Attraction)内。

因此，周期性地增大学习率虽然可能短期内损害优化过程，使得网络收敛的稳定性变差，但从长期来看有助于找到更好的局部最优解。

循环学习率
(Cyclic Learning Rate)

带热重启的随机梯度下降
(Stochastic Gradient Decent With Warm Restarts)

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

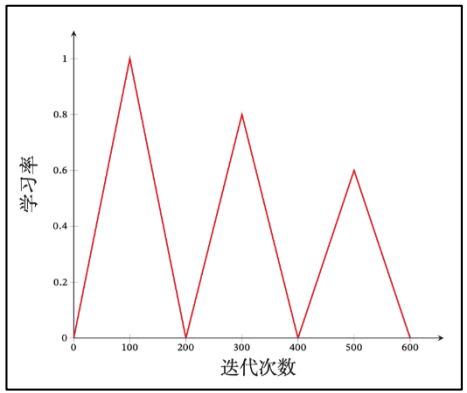
为了使得梯度下降法能够逃离鞍点或尖锐最小值，一种经验性的方式是在训练过程中周期性地增大学习率。

让学习率在一个区间内周期性缩放来调整学习率，称为三角循环学习率(Triangular Cyclic Learning Rate)。假设每个循环周期的长度相等都为 $2\Delta T$ ，其中前 ΔT 步为学习率线性增大阶段，后 ΔT 步为学习率线性缩小阶段。在第t次迭代时，其所在的循环周期数 m 为 $\left\lfloor 1+\frac{t}{2\Delta T} \right\rfloor$，则第$t$次迭代的学习率为：

$$\alpha_t = \alpha_{\min}^m + (\alpha_{\max}^m - \alpha_{\min}^m)\,(\max(0, 1-b))$$

第m个周期中学习率的上界与下界，随着m的增加而逐渐降低

$$b = \left|\frac{t}{\Delta T} - 2m + 1\right| \quad (b \in [0,1])$$



4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

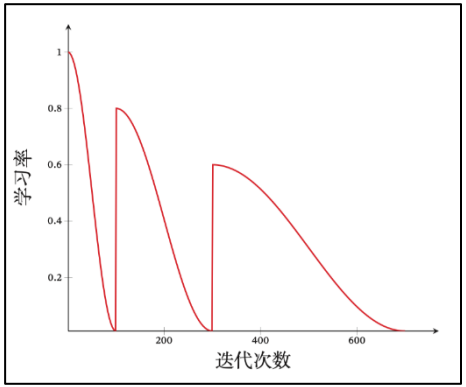
为了使得梯度下降法能够逃离鞍点或尖锐最小值，一种经验性的方式是在训练过程中周期性地增大学习率。

学习率每隔一定周期后初始化为某个预先设定值，然后逐渐衰减。每次重启模型后模型参数不是从头开始优化，而是从重启前的参数基础上继续优化。假设在梯度下降过程中重启 M 次，第 m 次重启在上次重启开始第T_m个回合后进行，T_m称为重启周期。在第 m 次重启之前，采用余弦衰减来降低学习率。第 t 次迭代的学习率为：

$$\alpha_t = \alpha_{\min}^m + \frac{1}{2}(\alpha_{\max}^m - \alpha_{\min}^m)\left(1 + \cos\left(\frac{T_{cur}}{T_m}\pi\right)\right)$$

第m个周期中学习率的上界与下界，随着m的增加而逐渐降低

$$T_m = T_{m-1} \times k$$



4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

2. 学习率调整

学习率过大，模型不会收敛；过小的话，收敛速度太慢。

对学习率的调整方法包括学习率衰减、学习率预热、周期性学习率调整、自适应调整学习率方法。

在标准的梯度下降法中，每个参数在每次迭代时都使用相同的学习率。由于每个参数的维度上收敛速度都不相同，因此根据不同参数的收敛情况分别设置学习率。

AdaGrad算法　　**RMSprop算法**　　**AdaDelta算法**

4. 网络优化

Mini-Batch Gradient Descent

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

2. 学习率调整

AdaGrad算法　RMSprop算法　AdaDelta算法

AdaGrad(Adaptive Gradient Algorithm)借鉴$\ell 2$正则化的思想，每次迭代时自适应地调整每个参数的学习率。在第t次迭代时，先计算每个参数梯度平方的累计值：

$$G_t = \sum_{\tau=1}^{t} \mathbf{g}_\tau \odot \mathbf{g}_\tau$$

$\mathbb{R}^{|\theta|}$ 是第 τ 次迭代时的梯度

AdaGrad算法的参数更新差值为：　$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t + \epsilon}} \odot g_t$

缺点：经过一定次数迭代依然没有找到最优点时，由于学习率已经非常小，很难再继续找到最优点。

4. 网络优化

Mini-Batch Gradient Descent

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

2. 学习率调整

AdaGrad算法　RMSprop算法　AdaDelta算法

RMSprop在有些情况下避免 AdaGrad 算法中学习率不断单调下降以至于过早衰减的缺点。

首先计算每次迭代梯度 g_t 平方的指数衰减移动平均:

$$G_t = \beta G_{t-1} + (1-\beta) g_t \odot g_t$$

$$= (1-\beta)\sum_{\tau=1}^{t} \beta^{t-\tau} g_\tau \odot g_\tau$$

衰减率，一般取0.9

RMSprop算法的参数更新差值为：

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t + \epsilon}} \odot g_t$$

4. 网络优化

Mini-Batch Gradient Descent

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

2. 学习率调整

AdaGrad算法　RMSprop算法　**AdaDelta算法**

AdaDelta也是AdaGrad算法的一个改进。和RMSprop算法类似AdaDelta算法通过梯度平方的指数衰减移动平均来调整学习率。此外，AdaDelta算法还引入了每次参数更新差值 $\Delta\theta$ 的平方的指数衰减权移动平均。第 t 次迭代时，参数更新差值 $\Delta\theta$ 的平方的指数衰减权移动平均为：

$$\Delta X_{t-1}^2 = \beta_1 \Delta X_{t-2}^2 + (1-\beta_1)\,\Delta\theta_{t-1} \odot \Delta\theta_{t-1}$$

RMSprop算法的参数更新差值为：

$$\Delta\theta_t = -\frac{\sqrt{\Delta X_{t-1}^2 + \epsilon}}{\sqrt{G_t + \epsilon}} \odot g_t$$

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

3. 梯度估计修正

在Mini-Batch梯度下降算法中，每次选取的样本数量比较小，损失会呈现震荡的方式下降(每次迭代时梯度的估计值和整个训练集上的最优梯度并不一致)。一种有效的方式是通过使用最近一段时间内的平均梯度来代替当前时刻的随机梯度来作为参数更新的方向，从而提高优化速度。

Momentum算法 **Nesterov加速梯度法** **Adam算法** **梯度截断法**

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

3. 梯度估计修正

Momentum算法　Nesterov加速梯度法　Adam算法　梯度截断法

Momentum法是利用之前积累的的动量来替代真正的梯度，每次迭代的梯度可以看作是加速度。

在第 t 次迭代时，计算负梯度的“加权移动平均”作为参数的更新方向。

动量因子　学习率

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \alpha g_t = -\alpha\sum_{\tau=1}^{t}\rho^{t-\tau}g_\tau$$

这样，每个参数的实际更新差值取决于最近一段时间内梯度的加权平均值。当某个参数在最近一段时间内的梯度方向不一致时，其真实的参数更新幅度变小；相反，当在最近一段时间内的梯度方向都一致时，其真实的参数更新幅度变大，起到加速作用。

4. 网络优化

Mini-Batch Gradient Descent

3. 梯度估计修正

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Momentum算法　**Nesterov加速梯度法**　Adam算法　梯度截断法

Nesterov加速梯度法是一种对动量法的改进。在Momentum方法中，实际的参数更新方向$\Delta\theta_t$为上一步的参数更新方向 $\Delta\theta_{t-1}$ 和当前梯度的反方向$-\boldsymbol{g}_t$的叠加。

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \alpha g_t \Rightarrow \begin{aligned} \hat{\theta} &= \theta_{t-1} + \rho\Delta\theta_{t-1} \\ \theta_t &= \hat{\theta} - \alpha g_t \end{aligned}$$

g_t：损失函数在点θ_{t-1}上的梯度

上式第二步更新中，更合理的更新方向应该是损失函数在$\hat{\theta}$处的梯度，合并后的更新方向为：

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

$g_t(\theta_{t-1} + \rho\Delta\theta_{t-1})$：损失函数在点$\hat{\theta}$上的偏导数

4. 网络优化

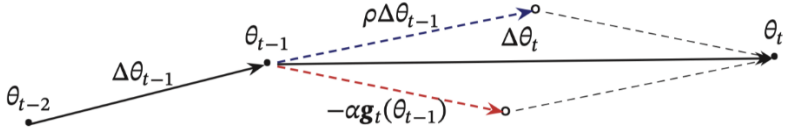
Mini-Batch Gradient Descent

3. 梯度估计修正

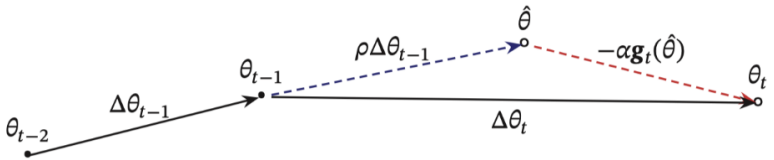
影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Momentum算法 **Nesterov加速梯度法** **Adam算法** **梯度截断法**



（a）动量法



（b）Nesterov加速梯度法

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

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \alpha \mathbf{g}_t\left(\theta_{t-1} + \rho\Delta\theta_{t-1}\right)$$

4. 网络优化

影响小批量梯度下降法的主要因素有：
- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

3. 梯度估计修正

Momentum算法　Nesterov加速梯度法　**Adam算法**　梯度截断法

Adam算法可以看作Momentum算法和RMSprop算法的结合，不但使用动量作为参数更新方向，而且可以自适应调整学习率。一方面计算梯度$\mathbf{g}_t$的指数加权平均，另一方面计算梯度平方$\mathbf{g}_t^2$的指数加权平均。

两个移动平均的衰减率
（$\beta_1 = 0.9$，　$\beta_2 = 0.99$）

$$M_t = \beta_1 M_{t-1} + (1-\beta_1)\,\mathbf{g}_t$$

动量法类似

$$G_t = \beta_2 G_{t-1} + (1-\beta_2)\,\mathbf{g}_t \odot \mathbf{g}_t$$

RMSprop算法类似

对偏差进行修正：$\hat{M}_t = \dfrac{M_t}{1-\beta_1^t}$　$\hat{G}_t = \dfrac{G_t}{1-\beta_2^t}$

参数更新差值为：$\Delta\theta_t = -\dfrac{\alpha}{\sqrt{\hat{G}_t + \epsilon}}\hat{M}_t$

4. 网络优化

影响小批量梯度下降法的主要因素有：

- 批量大小
- 学习率
- 梯度估计

Mini-Batch Gradient Descent

3. 梯度估计修正

Momentum算法　Nesterov加速梯度法　Adam算法　**梯度截断法**

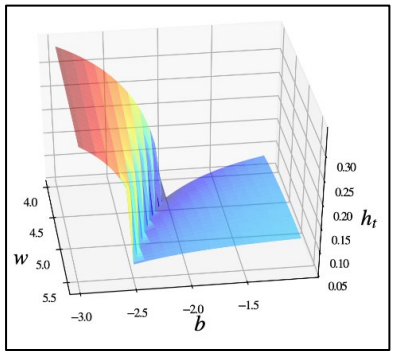
在深度神经网络或循环神经网络中，除了梯度消失之外，梯度爆炸也是影响学习效率的主要因素。在基于梯度下降的优化过程中，如果梯度突然增大，用大的梯度更新参数反而会导致其远离最优点。为了避免这种情况，当梯度的模大于一定阈值时，就对梯度进行截断。

- **按值截断**：限定在区间[a, b]内

$$\mathbf{g_t} = \max(\min(\mathbf{g_t}, b), a)$$

- **按模截断**：对于模大于b的部分限定

$$\mathbf{g_t} = \frac{b}{||\mathbf{g_t}||}\mathbf{g_t}$$



4. 网络优化

总结

类别		优化算法
学习率调整	固定衰减学习率	分段常数衰减、逆时衰减、(自然)指数衰减、余弦衰减
	周期性学习率	循环学习率、SGDR
	自适应学习率	AdaGrad、RMSprop、AdaDelta
梯度估计修正		动量法、Nesterov加速梯度、梯度截断
综合方法		Adam≈动量法+RMSprop

$$\Delta\theta_t = -\frac{\alpha_t}{\sqrt{G_t + \epsilon}} M_t$$

$$G_t = \psi\left(\mathbf{g}_1, \cdots, \mathbf{g}_t\right)$$

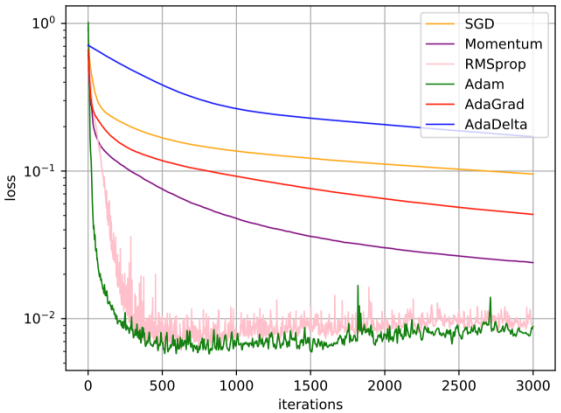
$$M_t = \phi\left(\mathbf{g}_1, \cdots, \mathbf{g}_t\right)$$

$\mathbf{g_t}$ 第t步的梯度

α_t 第t步的学习率，可衰减

$\psi(\cdot)$ 学习率缩放函数，取1或历史梯度的模的移动平均

$\phi(\cdot)$ 优化后的参数更新方向，取当前梯度或历史梯度的移动平均



课后作业

必做

网络优化中，对Epoch、Batch和Iteration的学习中，假设某个训练任务中有10万个样本作为训练数据，1万个样本作为测试数据。

现在选择Batch_Size=100对模型进行训练，总共迭代3万次，请计算回合数，每个回合的数据划分情况。

如果Batch_Size设置为128，结果是多少?