

# 第3章： 深度学习入门

周炜星 谢文杰

华东理工大学金融学系

2023年秋

# 纲要

- 1 深度学习简介
- 2 深度神经网络
- 3 深度卷积神经网络
- 4 深度循环神经网络
- 5 图神经网络
- 6 深度神经网络训练
- 7 应用实践

# 纲要

- 1 深度学习简介
- 2 深度神经网络
- 3 深度卷积神经网络
- 4 深度循环神经网络
- 5 图神经网络
- 6 深度神经网络训练
- 7 应用实践



# 深度学习与人工智能

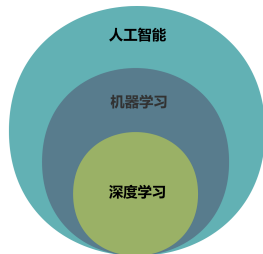


图 1: 深度学习、机器学习和人工智能的关系简图

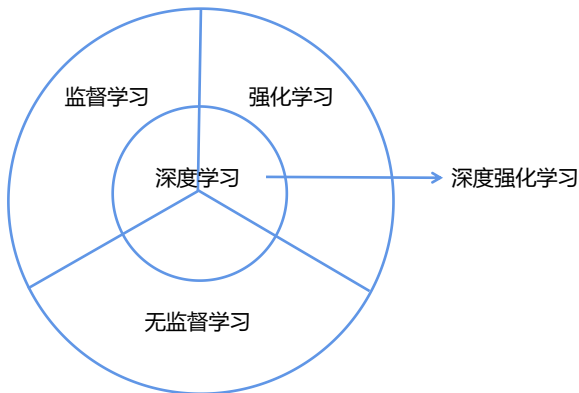
图1给出了深度学习、机器学习与人工智能的关系简图。机器学习是人工智能一项关键技术，深度学习是机器学习技术的重要组成部分。

# 深度学习与机器学习

- 人工智能融合了众多学科的理论和技术，此次人工智能浪潮的核心技术是机器学习，更准确地说是深度学习。
- 深度学习是一类技术集合，不仅仅局限于深度神经网络，但深度神经网络使用最为广泛。
- 机器学习可以分为监督学习、无监督学习和强化学习。三类机器学习范式都可以与深度学习技术融合，衍生出更为强大的机器学习算法，如“深度强化学习”。
- 深度强化学习融合了深度学习和强化学习的范式。

# 深度学习与机器学习

深度学习与机器学习三类学习范式：监督学习、无监督学习和强化学习的关系如图2所示。



# 深度学习与机器学习差异

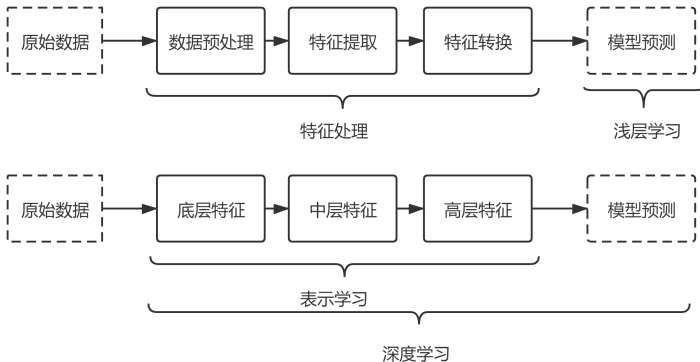


图 3: 深度学习与经典机器学习



# 深度学习与机器学习

- 经典机器学习预测模型做了浅层学习（Shallow Learning），可以选择线性回归模型（Linear Regression Model）、决策树（Decision Tree）、随机森林（Random Forest）、支持向量机（Support Vector Machine）等方法。
- 经典深度学习模型，包括深度神经网络（Deep Neural Networks）、卷积神经网络（Convolutional Neural Networks）、循环神经网络（Recurrent Neural Networks）、图神经网络（Graph Neural Networks）等。

# 深度学习与表示学习

- 表示学习（Representation Learning）是深度学习的核心概念。
- 一般来说，机器学习模型中有效的特征信息可以称作表示（Representation）。表示学习方法是应用优化算法自动地学习有效的特征信息，从而提高机器学习模型性能。
- 什么是有效的表示？这取决于机器学习模型解决的问题。

## 一个简单的例子

机器学习模型基于个人信息对个体性别进行判断，关于身高的特征表示意义不大，信息含量较低；关于头发长度的特征信息比较重要，对个体性别判断更有价值，有利于模型做出准确的性别判断。表示学习方法基于学习目标自动进行特征提取和特征表示。

# 深度学习的黑盒属性

- 深度神经网络进行表示学习的过程属于端到端学习（End-to-End Learning）。
- 应用端到端学习过程不需要人工特征提取，直接通过任务目标和输入数据之间的关系构建模型、优化参数，极大地方便了模型的训练和部署。
- 端到端学习也带来了另一个问题，即模型的可解释性欠缺，这也是深度学习模型广为诟病的“黑盒问题”。
- 模型训练难度大。
- 模型可解释性困难。
- 模型安全可靠有限。

# 纲要

- 1 深度学习简介
- 2 深度神经网络
- 3 深度卷积神经网络
- 4 深度循环神经网络
- 5 图神经网络
- 6 深度神经网络训练
- 7 应用实践

# 深度神经网络构建

- 深度神经网络（Deep Neural Networks）是很多深度学习模型和系统的标准模块，是智能系统中核心部件。
- 深度神经网络提升智能模型的性能得益于人工神经网络对任意复杂非线性函数的逼近能力。

严格来说，示例图4并非一个深度神经网络，因为它只有一个隐含层。深度神经网络的“深度”是指叠加的神经网络层数之多。

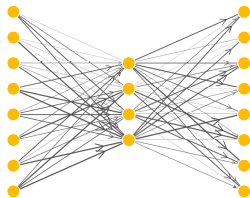


图 4: 神经网络示意图

# 深度神经网络构建

## 一个简单的例子

图4中的神经网络输入参数为 $x$ ，是一个列向量，大小为 $n_x = 8$ 。中间隐藏层有 $n_h = 4$ 个神经元，每个隐藏层神经元都计算神经元状态数值 $h$ ，并以 $n_x = 8$ 个输入神经元为输入变量：

$$h = \sigma(W_1 \cdot x + b_1), \quad (1)$$

$W_1$  为输入层和中间隐藏层之间的参数矩阵，大小为 $n_h \times n_x$ ，偏置项  $b_1$  的大小为 $n_h$ ， $\sigma$ 为非线性激活函数，具体形式为

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \quad (2)$$

# 深度神经网络构建

## 一个简单的例子

一般深度神经网络模型有多层隐藏层，在数学形式上表示为嵌套多个线性和非线性函数，神经网络输出值  $y$  可表示成

$$y = W_2 \cdot h + b_2, \quad (3)$$

其中  $W_2$  大小为  $n_y \times n_h$ ，偏置项  $b_2$  大小为  $n_y$ 。所以：

$$y = W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2. \quad (4)$$

如果神经网络模型叠加新的隐藏层，可以表示成

$$y = W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2) + b_3. \quad (5)$$

# 深度神经网络构建

深度神经网络模型不管神经元叠加了多少层、函数嵌套了多少次，都可以简化成一个输入向量 $x$ 到输出 $y$ 的复杂非线性函数

$y = f_{W_1, W_2, W_3, b_1, b_2, b_3}(x)$ 。我们基于数据样本

$\{(x_k, y_k)\}_{k=1,2,3,\dots,N}$ ，采用监督学习方法训练模型参数，可以构建损失函数作为机器学习优化的目标函数：

$$\mathcal{L}(W_1, W_2, W_3, b_1, b_2, b_3) = \frac{1}{N} \sum_{k=1}^N (f_{W_1, W_2, W_3, b_1, b_2, b_3}(x_k) - y_k)^2. \quad (6)$$

机器学习优化算法完成监督学习后，可以获得模型

$f_{W_1, W_2, W_3, b_1, b_2, b_3}$ 的参数 $W_1$ 、 $W_2$ 、 $W_3$ 、 $b_1$ 、 $b_2$ 、 $b_3$ ，进而进行模型预测和生成等应用。



# 深度神经网络实例：线性可分实例

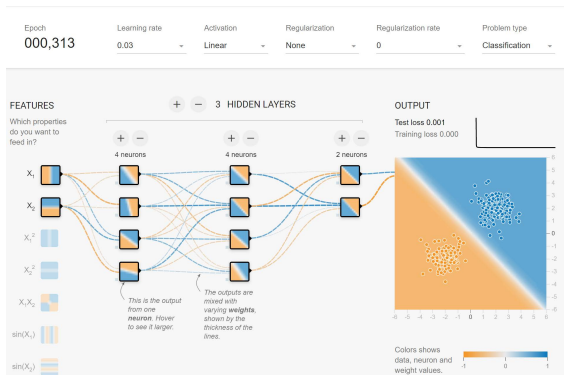


图 5: 分类问题示例: <https://playground.tensorflow.org>

# 深度神经网络实例：线性不可分实例

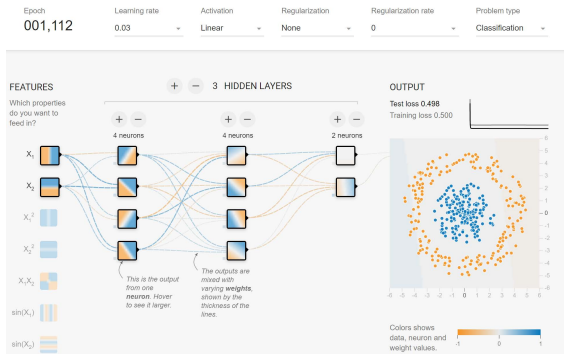


图 6: 分类问题示例: <https://playground.tensorflow.org>

# 深度神经网络实例：线性不可分实例

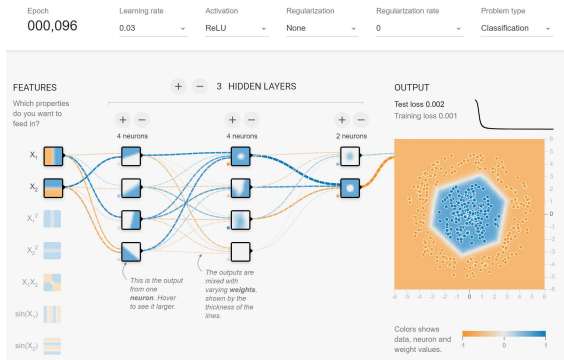


图 7: 分类问题示例: <https://playground.tensorflow.org>

# 深度神经网络实例：线性不可分实例

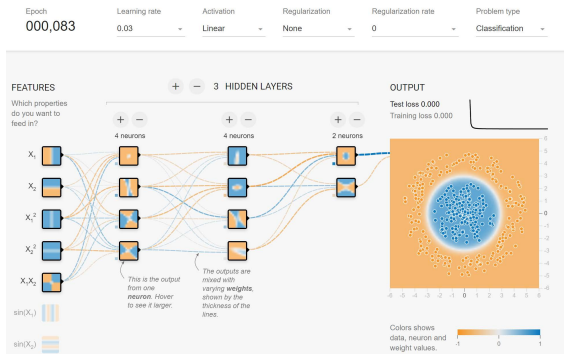


图 8: 分类问题示例: <https://playground.tensorflow.org>

# 纲要

- 1 深度学习简介
- 2 深度神经网络
- 3 深度卷积神经网络
- 4 深度循环神经网络
- 5 图神经网络
- 6 深度神经网络训练
- 7 应用实践

# 深度卷积神经网络

相较于基础深度神经网络（一般指深度前馈神经网络或全连接神经网络），深度卷积神经网络（Convolutional Neural Networks, CNN）具有三个重要的特征：

- 局部连接性：卷积核决定了卷积神经网络的局部连接性质
- 权值共享：大大减少深度神经网络模型的参数
- 下采样：池化操作

卷积神经网络有一个非常重要的组成部分叫做卷积核，可以看作一个算子。卷积核作用在数据上，提取数据特征，在叠加不同层后，抽取不同层次的特征，可达到智能决策的目的。

# 深度卷积神经网络实例

深度卷积神经网络模型结构如图9所示<sup>1</sup>。

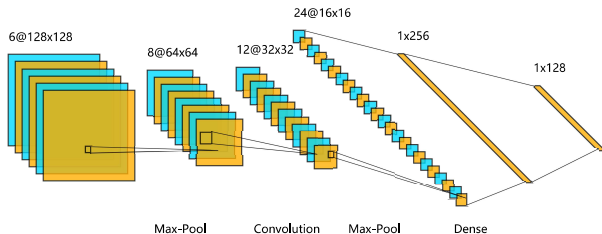


图 9: 卷积神经网络结构示意图

<sup>1</sup><http://alexlenail.me/NN-SVG/index.html>

# 深度卷积神经网络实例

图10给出了一个三维立体的且更加清晰的卷积神经网络结构图<sup>2</sup>。

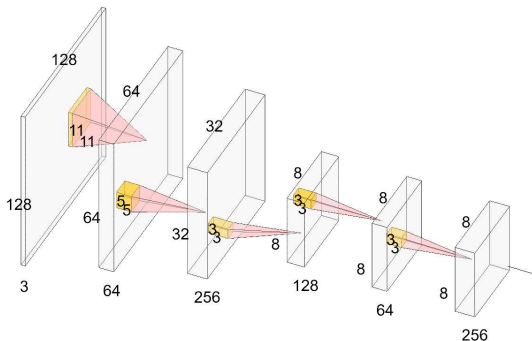


图 10: 三维立体卷积神经网络结构示意图



# 深度卷积神经网络

深度卷积神经网络结构可以非常复杂，叠加层数可达上千层，但是不管多复杂的卷积神经网络结构，都可以看做是构建了一个非线性函数映射关系：

$$y = f_{\text{CNN}}(x), \quad (7)$$

其中 $x$ 为卷积神经网络模型的输入层数据， $y$ 为输出层数据。经典的深度神经网络模型都涉及到非常深厚和庞杂的理论和技術细节，需要深入研究并参考更多的专业书籍和资料。

# 纲要

- 1 深度学习简介
- 2 深度神经网络
- 3 深度卷积神经网络
- 4 深度循环神经网络**
- 5 图神经网络
- 6 深度神经网络训练
- 7 应用实践

# 深度循环神经网络

当深度学习模型的输入数据 $x$ 具有时间序列特性时，我们可以采用深度循环神经网络（Recurrent Neural Networks, RNN）进行建模分析。循环神经网络模型的输入数据形式为 $\{x_0, x_1, \dots, x_T\}$ 。我们将简单介绍RNN模型中最著名的长短期记忆（Long Short-Term Memory, LSTM）模型架构，如图11所示：

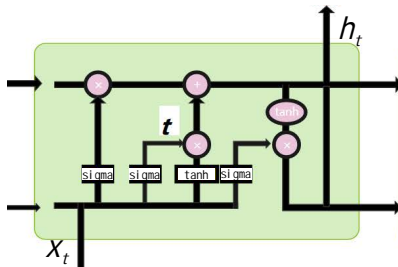


图 11: 深度循环神经网络模型LSTM的单元结构

# 深度循环神经网络

深度循环神经网络模型中的 $t$ 表示时间刻度，模型输入量 $i_t$ 为：

$$i_t = \sigma(W_x^i x_t + W_h^i h_{t-1} + b^i), \quad (8)$$

其中， $x_t$ 为输入变量， $W_x^i$ 为输入变量 $x_t$ 对应的权重系数矩阵， $h_{t-1}$ 为隐含变量， $W_h^i$ 为隐含变量 $h_{t-1}$ 对应的权重系数矩阵， $b^i$ 为输入量的偏置项。

遗忘门 $f_t$ 计算公式如下：

$$f_t = \sigma(W_x^f x_t + W_h^f h_{t-1} + b^f), \quad (9)$$

其中 $x_t$ 为输入变量， $W_x^f$ 为输入变量 $x_t$ 对应的权重系数矩阵， $W_h^f$ 为隐含变量 $h_{t-1}$ 对应的权重系数矩阵， $b^f$ 为遗忘量的偏置项。

# 深度循环神经网络

记忆门 $g_t$ 定义为:

$$g_t = \tanh(W_x^g x_t + W_h^g h_{t-1} + b^g), \quad (10)$$

其中 $W_x^g$ 为输入变量 $x_t$ 对应的权重系数矩阵， $W_h^g$ 为隐含变量 $h_{t-1}$ 对应的权重系数矩阵， $b^g$ 为输出量的偏置项， $\tanh(x)$ 为非线性激活函数，具体形式为

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (11)$$

输出量 $o_t$ 定义为:

$$o_t = \sigma(W_x^o x_t + W_h^o h_{t-1} + b^o) \quad (12)$$

其中 $W_x^o$ 为输入变量 $x_t$ 对应的权重系数矩阵， $W_h^o$ 为隐含变量 $h_{t-1}$ 对应的权重系数矩阵， $b^o$ 为输出量的偏置项。

# 深度循环神经网络

单元中的状态量 $\mathbf{c}_t$ 定义为:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t, \quad (13)$$

其中 $\odot$  表示元素相乘的操作符。单元中的状态量 $\mathbf{c}_t$ 由上一时刻单元中的状态变量 $\mathbf{c}_{t-1}$ 通过遗忘门后，叠加输入量 $\mathbf{i}_t$ 和记忆门 $\mathbf{g}_t$ 构成。

循环神经网络模型中隐含变量 $\mathbf{h}_t$ 的更新公式为:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (14)$$

在循环神经网络模型中， $\mathbf{W} = \{\mathbf{W}_x^i, \mathbf{W}_x^f, \mathbf{W}_x^g, \mathbf{W}_x^o, \mathbf{W}_h^i, \mathbf{W}_h^f, \mathbf{W}_h^g, \mathbf{W}_h^o\}$  和  $\mathbf{b} = \{\mathbf{b}^i, \mathbf{b}^f, \mathbf{b}^g, \mathbf{b}^o\}$  表示可学习的权重系数和偏置量。 $\mathbf{h}_0, \mathbf{c}_0$  初始化为0。

# 深度循环神经网络

循环神经网络模型通过循环迭代计算输出量 $\mathbf{o}_t$ 、状态量 $\mathbf{c}_t$ 、隐含变量 $\mathbf{h}_t$ 等，最后得到RNN模型的循环结构，如图12所示。

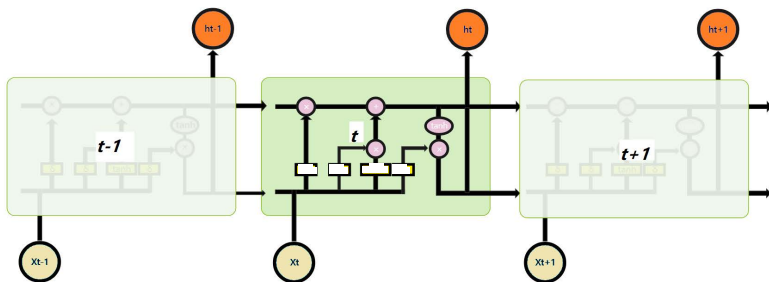


图 12: 深度循环神经网络模型LSTM结构示意图

# 纲要

- 1 深度学习简介
- 2 深度神经网络
- 3 深度卷积神经网络
- 4 深度循环神经网络
- 5 图神经网络**
- 6 深度神经网络训练
- 7 应用实践



# 深度图神经网络

- 深度神经网络、卷积神经网络和循环神经网络分别对向量形式数据、矩阵或空间结构数据、时序结构数据有着较好的适配性。
- 但是，深度卷积神经网络模型并非只能处理图片等类似结构的数据，如用卷积神经网络模型处理音频数据时，只需将卷积核设计成一维向量即可。
- 深度学习模型DNN、CNN、RNN所处理的大多数数据是欧式空间中的标量、向量、矩阵或张量数据。
- 现实世界中的很多关系型数据更加复杂，更加难以处理，如网络数据和图数据。
- **深度图神经网络（Graph Neural Networks, GNN）**模型是近年来发展较快的领域。图神经网络模型以网络数据为输入，通过深度学习算法能够学到网络节点、连边和整个网络的特征表示，进而进行智能决策。

# 纲要

- 1 深度学习简介
- 2 深度神经网络
- 3 深度卷积神经网络
- 4 深度循环神经网络
- 5 图神经网络
- 6 深度神经网络训练**
- 7 应用实践

# 模型训练挑战

- 深度学习模型在自然语言处理、计算机视觉等领域取得了非凡成就，得益于深度神经网络的深度（层数）和广度（神经元数量），深度神经网络模型具有强大的表示学习能力。
- 神经网络模型的深度和广度规模增加了优化模型参数的难度，使得模型训练异常困难。
- 神经网络模型已经发展了几十年，一直以来模型训练过程中梯度消失和梯度爆炸等问题都困扰着研究人员，也限制了深度学习模型的发展和应用。
- 一般而言，深度神经网络模型的参数优化问题是一个**超高维、非线性、非凸优化问题**。

# 模型训练挑战

图13给出了在二维空间中寻找最小值的优化问题示例。

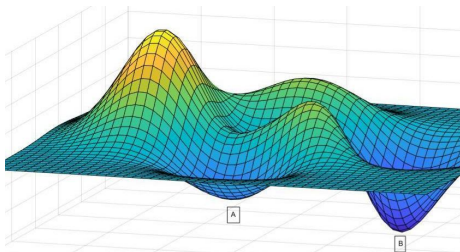


图 13: 优化问题示例

关于超高维、非线性、非凸函数优化问题，我们将简单介绍一些常用的算法和技巧，包括数据预处理、参数初始化、学习率调整、超参数优化、神经网络正则化和优化算法等。

# 数据预处理

深度学习模型处理的数据多源、高维、异构，输入变量可能不在一个尺度上。在数据预处理过程中，我们通过**最小值最大值归一化（Min-Max Normalization）**，将不同变量都缩放到0到1之间或者-1到1之间以减少变量间的尺度差异，具体计算如下所示：

$$\hat{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \quad (15)$$

其中， $x_{\min}$  和  $x_{\max}$  分别表示数据的最小值和最大值。数据进行最小值最大值归一化时，我们需要分析数据分布情况，如果数据中存在远大于平均值的离群点， $x_{\max}$  数值较大，那么在大部分情况下分母将远远大于分子，使得大部分数据将变换到0附近，不利于模型的特征提取和表示学习。

# 标准化 (Standard Normalization)

数据标准化是普遍使用的数据预处理方式，先计算数据均值：

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i, \quad (16)$$

以及标准差：

$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2, \quad (17)$$

则数据标准化计算公式为：

$$\hat{x}_i = \frac{x_i - \mu}{\sigma}, \quad (18)$$

当数据满足正态分布时，进行标准化后，将会有约99%的数据在-3到3之间。

# 白化 (Whitening)

白噪声是不同频率都有相同功率的随机信号。“白噪声”名字来源于白光，白光包含了光谱中所有的颜色。白噪声信号的平均值为0，且各个分量之间互不相关。数据零均值化操作简单，只需要减去均值

$$\hat{x}_i = x_i - \frac{1}{N} \sum_{i=1}^N x_i, \quad (19)$$

数据白化转换有多种选择，如Cholesky白化（Cholesky 分解）和PCA白化（PCA分解），其中主成分分析白化（PCA白化）比较常见。主成分分析得到的成分之间具有独立性，满足各分量之间互不相关的特性。

# 参数初始化

深度学习模型的训练过程是一个参数更新、参数优化的过程。深度学习模型初始化参数不同，可能得到不同的最优解，而且大部分是局部最优解。

- **预训练初始化**是指模型参数已经在大规模数据集上进行了训练，将训练好的模型参数作为初始化参数。
- 深度学习模型训练中广泛采用**随机初始化参数**，重复多次随机初始化参数并训练模型参数，验证模型的鲁棒性，获得更有效的深度学习模型。
- 对于深度学习模型中的一些特殊参数，我们可以基于经验知识选定**固定的数值进行初始化**，如深度神经网络中的偏置项（Bias），一般可初始化为0。



# 学习率调整

学习率是决定参数更新步长的关键超参数。一般而言，更新步长在开始阶段可以尽量大一些，加大对参数空间的探索，使得模型能够尽可能的收敛到全局最优解。在参数更新的后期阶段，参数更新步长应小一些，使参数更好地收敛到局部最优值。

我们设定初始的学习率为 $\alpha_0$ ，第 $t$ 步更新时学习率为 $\alpha_t$ 。在深度学习中，最简单的学习率调整策略是，随着更新时间逐步衰减学习率，在实际应用中常见的学习率衰减策略有多种选择。比如，学习率 $\alpha_t$ 随着时间衰减可以表示为：

$$\alpha_t = \alpha_0 \frac{1}{1 + \beta \times t}, \quad (20)$$

其中， $\beta$ 为衰减率。

# 学习率调整

学习率 $\alpha_t$ 随着时间呈指数衰减，可以表示为：

$$\alpha_t = \alpha_0 e^{-\beta \times t}, \quad (21)$$

其中， $\beta$ 为衰减率。

学习率 $\alpha_t$ 随着时间余弦衰减，可以表示为：

$$\alpha_t = \frac{1}{2} \alpha_0 \left( 1 + \cos \left( \frac{t\pi}{T} \right) \right), \quad (22)$$

其中， $T$ 为迭代总次数。

# 学习率调整

在学习率衰减函数中，当迭代步数特别大时，学习率会非常之小，因此后期参数更新速度会很慢。因此，我们可以设定初始学习率 $\alpha_0$ 和最小的学习率 $\alpha_{\min}$ ，同时设定学习率在初始的 $T_0$ 步以内并不进行衰减。具体公式如下：

$$\alpha_t = \begin{cases} \alpha_0, & t \leq T_0 \\ \alpha_0 \frac{1}{1+\beta \times t}, & t > T_0 \text{ and } \alpha_0 \frac{1}{1+\beta \times t} > \alpha_{\min} \\ \alpha_{\min}, & \alpha_0 \frac{1}{1+\beta \times t} < \alpha_{\min} \end{cases} \quad (23)$$

或者，在迭代总次数的后10%的迭代中采用最小学习率 $\alpha_{\min}$ ：

$$\alpha_t = \begin{cases} \alpha_0, & t \leq T_0 \\ \alpha_0 \frac{1}{1+\beta \times t}, & T_0 < t \leq 0.9T \\ \alpha_{\min}, & t > 0.9T \end{cases} \quad (24)$$

# 梯度优化算法

在机器学习基础中，我们已经介绍了随机梯度下降算法：

表 3.1 常用梯度优化算法汇总

伪代码编号	算法名称	核心公式
1	随机梯度下降算法	$\theta_k = \theta_{k-1} - \alpha \nabla_{\theta} \mathcal{L}$
2	动量随机梯度下降算法	$g_k = \beta g_{k-1} - \alpha \nabla_{\theta} \mathcal{L}$ $\theta_k = \theta_{k-1} + g_k$
3	Nestrov 动量随机梯度下降算法	$\nabla_{\theta} \mathcal{L}(\theta_{k-1} + \beta g_{k-1})$ $g_k = \beta g_{k-1} - \alpha \nabla_{\theta} \mathcal{L}(\theta_{k-1} + \beta g_{k-1})$ $\theta_k = \theta_{k-1} + g_k$
4	自适应梯度下降算法	$t_k = t_{k-1} + (\nabla_{\theta} \mathcal{L}(\theta_{k-1}))^2$ $\theta_k = \theta_{k-1} - \frac{\alpha}{\sqrt{t_k} + \epsilon} \nabla_{\theta} \mathcal{L}(\theta_{k-1})$
5	RMSprop 梯度下降算法	$t_k = \gamma t_{k-1} + (1 - \gamma) (\nabla_{\theta} \mathcal{L}(\theta_{k-1}))^2$ $\theta_k = \theta_{k-1} - \frac{\alpha}{\sqrt{t_k} + \epsilon} \nabla_{\theta} \mathcal{L}(\theta_{k-1})$
6	Adadelta 梯度下降算法	$t_k = \gamma t_{k-1} + (1 - \gamma) (\nabla_{\theta} \mathcal{L}(\theta_{k-1}))^2$ $g_k = -\frac{\sqrt{\Delta_{k-1} + \epsilon}}{\sqrt{t_k} + \epsilon} \nabla_{\theta} \mathcal{L}(\theta_{k-1})$ $\theta_k = \theta_{k-1} + g_k$ $\Delta_k = \gamma \Delta_{k-1} + (1 - \gamma) g_k^2$

# 超参数优化

深度学习模型的参数可以分成两类：可学习参数和超参数（Hyperparameter）。可学习参数是指通过优化算法更新和优化的参数，如神经网络权重参数等；超参数是指人工设定的模型参数，如学习率等。

- 深度学习模型的超参数有很多种，包括了网络层数、神经元数量、激活函数类型、优化算法类型、小批量样本数量、正则化系数等。
- 超参数优化（Hyperparameter Optimization）直接影响了优化算法找到的可学习参数的质量。超参数优化可以使用网格搜索、随机搜索和神经网络架构搜索等方法。

# 网格搜索

一般而言，超参数空间中不同超参数组合与模型最终损失函数值具有非线性关系，而网格搜索就是一种直接遍历超参数空间的方法（穷举法）。在学习率衰减的情况下，在模型训练前，超参数学习率 $\alpha$ 和衰减率 $\beta$ 需要提前设定。如下表所示：

	$\alpha_1$	$\alpha_2$	$\dots$	$\alpha_9$	$\alpha_{10}$
$\beta_1$	$(1, 10^{-1})$	$(1, 10^{-2})$	$\dots$	$(1, 10^{-9})$	$(1, 10^{-10})$
$\beta_2$	$(2, 10^{-1})$	$(2, 10^{-2})$	$\dots$	$(2, 10^{-9})$	$(2, 10^{-10})$
$\beta_3$	$(3, 10^{-1})$	$(3, 10^{-2})$	$\dots$	$(3, 10^{-9})$	$(3, 10^{-10})$
$\beta_4$	$(4, 10^{-1})$	$(4, 10^{-2})$	$\dots$	$(4, 10^{-9})$	$(4, 10^{-10})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\beta_9$	$(9, 10^{-1})$	$(9, 10^{-2})$	$\dots$	$(9, 10^{-9})$	$(9, 10^{-10})$
$\beta_{10}$	$(10, 10^{-1})$	$(10, 10^{-2})$	$\dots$	$(10, 10^{-9})$	$(10, 10^{-10})$

其中，每个网格元素对应一组超参数组合，因此称之为网格搜

# 随机搜索

随机搜索不需要按照网络搜索进行穷举，而是设定迭代总次数，从超参数空间中随机抽样超参数组合。模型训练总次数可控，随机搜索效率较高。

超参数空间可以看做是深度学习模型优化的更高层次的解空间。给定超参数 $\theta_{\text{Hyper}}$ ，最优超参数同样也可以表示成最小化损失函数 $\mathcal{L}(\theta_{\text{Hyper}})$ 的最优解：

$$\theta_{\text{Hyper}}^* = \arg \min_{\theta_{\text{Hyper}}} \left( \min_{\theta} \mathcal{L}(\theta_{\text{Hyper}}, \theta) \right), \quad (25)$$

超参数调优是更高层次的函数优化问题。为了提高效率，我们可以采用贝叶斯优化和动态资源分配等方法进行超参数调优，如神经架构搜索（Neural Architecture Search, NAS）。

# 正则化技术

在深度学习模型的优化过程中，巨量参数的估计问题容易造成过拟合。因此，我们需要很多网络正则化技术来训练深度神经网络模型参数：

- L1正则
- L2正则
- 权重衰减
- 提前停止
- 数据增强等

深度学习是一门综合性技术，来自不同领域的专家学者共同构建了一个庞大的学习算法平台，提供了大量高效的学习算法。

TensorFlow和PyTorch类型的计算平台是入门深度学习模型实现的较优选择。深度学习技术使用者可以更多地关注如何应用和实现深度学习技术。



# 纲要

- 1 深度学习简介
- 2 深度神经网络
- 3 深度卷积神经网络
- 4 深度循环神经网络
- 5 图神经网络
- 6 深度神经网络训练
- 7 应用实践**

# 深度学习平台TensorFlow安装

TensorFlow最初由谷歌机器学习研究团队的研究人员和工程师开发，用于机器学习和深度神经网络研究。TensorFlow系统具有足够的通用性和稳定性，落地工业应用较多，可以应用于非常广泛的领域。


- Python和Anaconda
- TensorFlow基本框架<sup>3</sup>
- TensorBoard
- Scikit-learn <sup>4</sup>
- Keras<sup>5</sup>

<sup>3</sup>An Open Source Machine Learning Framework for Everyone.<https://github.com/tensorflow/tensorflow>

<sup>4</sup>scikit-learn, <https://scikit-learn.org/>

<sup>5</sup>[https://keras-cn.readthedocs.io/en/latest/getting\\_started/sequential\\_model/](https://keras-cn.readthedocs.io/en/latest/getting_started/sequential_model/)

# 深度学习平台PyTorch安装

 PyTorch

[Get Started](#) [Ecosystem](#) [Mobile](#) [Blog](#) [Tutorials](#) [Docs](#) [Resources](#) [GitHub](#) [Q](#)

[Start Locally](#) [PyTorch 2.0](#) [Start via Cloud Partners](#) [Previous PyTorch Versions](#) [Mobile](#)

Shorecous

Prerequisites

- Supported Windows Distributions
- Python
- Package Manager

Installation

- Anaconda
- pip

Verification

Building from source

Prerequisites

## START LOCALLY

Select your preferences and run the Install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also **install previous versions of PyTorch**. Note that LibTorch is only available for C++.

PyTorch Build	Stable (2.0.1)	Preview (Nightly)
Your OS	Linux	Mac
Package	Conda	Pip
Language	Python	C++ / Java
Compute Platform	CUDA 11.7	CUDA 11.8
		ROCm 5.4.2
		CPU
Run this Command:	<pre>pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117</pre>	

# 掌握的问题

- 1 简要阐述深度学习、机器学习、人工智能的关系。
- 2 深度学习与经典机器学习的主要差别是什么？
- 3 构建一个深度前馈神经网络，并在MNIST上测试模型性能。
- 4 构建一个深度卷积神经网络，并在MNIST上测试模型性能。
- 5 构建一个深度循环神经网络，并在MNIST上测试模型性能。
- 6 熟悉TensorFlow参数初始化函数应用。
- 7 熟悉TensorFlow梯度优化算法应用。
- 8 熟悉TensorFlow正则化技术应用。