第2章：人工智能与机器学习

周炜星 谢文杰

华东理工大学金融学系

2023年秋

纲要

纲要

人工智能简介

- 人工智能（Artificial Intelligence， AI）定义很多。人工智能先驱马文·明斯基（Marvin Lee Minsky）认为：“人工智能就是研究‘让机器来完成那些如果由人来做需要智能的事情’的科学。”
- 围棋对弈被认为是人类智能的堡垒，一直是人工智能技术期望攻克的难题。AlphaGo作为第一个战胜围棋世界冠军（李世石，2016年）的人工智能程序，开启了又一轮人工智能浪潮。
- 人工智能在教育、医疗、机械、商业等领域都有着广泛运用。
- 人类从“互联网+”和“物联网+”走向了“人工智能+”。

人工智能+农业

农业是社会的基石，是人类赖以生存的根基。随着自然环境的恶化，土壤、气候、水资源等问题对农业生产造成了不利影响。

- 人工智能育种
- 农业土壤管理
- 农业灌溉和水资源管理
- 肥料使用、天气预测
- 计算机视觉识别农作物、检测杂草
- 识别害虫、智能农业机械制造、自动化收割等

人工智能+教育

人工智能在教育行业也有着较多应用，如智能化的教育评价、智能化的学习资源推荐、智能化的教学过程管理等。

对学生而言：

- 人工智能技术可实现个性化学习辅助和自适应学习
- 对学生在学习中出现的难点和薄弱点能够自动识别和智能发现
- 为学生制定个性化的、极具针对性的学习方案

对教师而言：

- 构建智能题库
- 优化整合教学资源并自动管理教学过程
- 教师可以将更多的精力放在学生的身心健康和思想品质上，更好地实现价值塑造和能力培养

人工智能+工业

人工智能在现代工业中的应用可谓是由来已久。从工业革命到信息革命，工业自动化是不变的主题，自动化领域一直是人工智能发展的另一通道，也是一个持续发展的研究领域。自动化控制、最优化控制等都是人工智能领域的重要技术和方法。

- 分析技术（Analytics Technology）
- 大数据技术（Big Data Technology）
- 云或网络技术（Cloud or Cyber Technology）
- 专业领域知识（Domain Knowledge）
- 证据（Evidence）

证据是指在工业应用中，收集工业数据与关联证据，改进人工智能模型，迭代更新，与时俱进，更好地完成工业任务。

人工智能+金融

随着人工智能相关技术的发展，自然语言处理、图像识别、深度强化学习算法和智能推荐算法等人工智能技术为银行金融业提供了大量的智能机器人。

- 产品推广、客户呼叫、客户服务、智能选股、智能投顾等领域。
- 在信用卡领域中，大量人工智能系统应用于业务推荐、客户信誉评级、智能催收等场景。
- 互联网金融、大数据金融、金融科技、科技金融等都是人工智能技术应用的重要领域。

人工智能前沿研究成果

世界各地顶级研究机构和知名大学科研人员设计了大量人工智能和机器学习算法，在医疗、教育、工业、农业、金融等领域得到了广泛应用。

- AlphaGo
- AlphaStar
- AlphaFold
- GPT
- ChatGPT

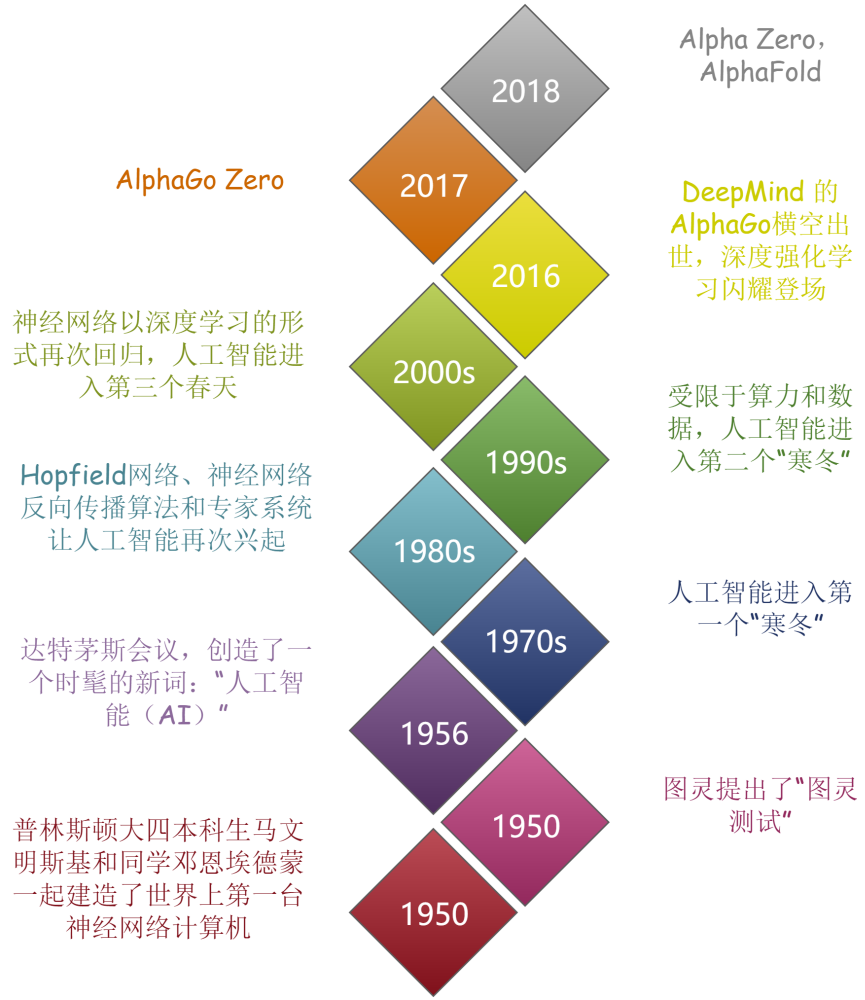
人工智能前沿研究

人工智能技术蓬勃发展，也伴随着不同的质疑。模型和方法的问题越来越受到各个领域的专家学者的重视。

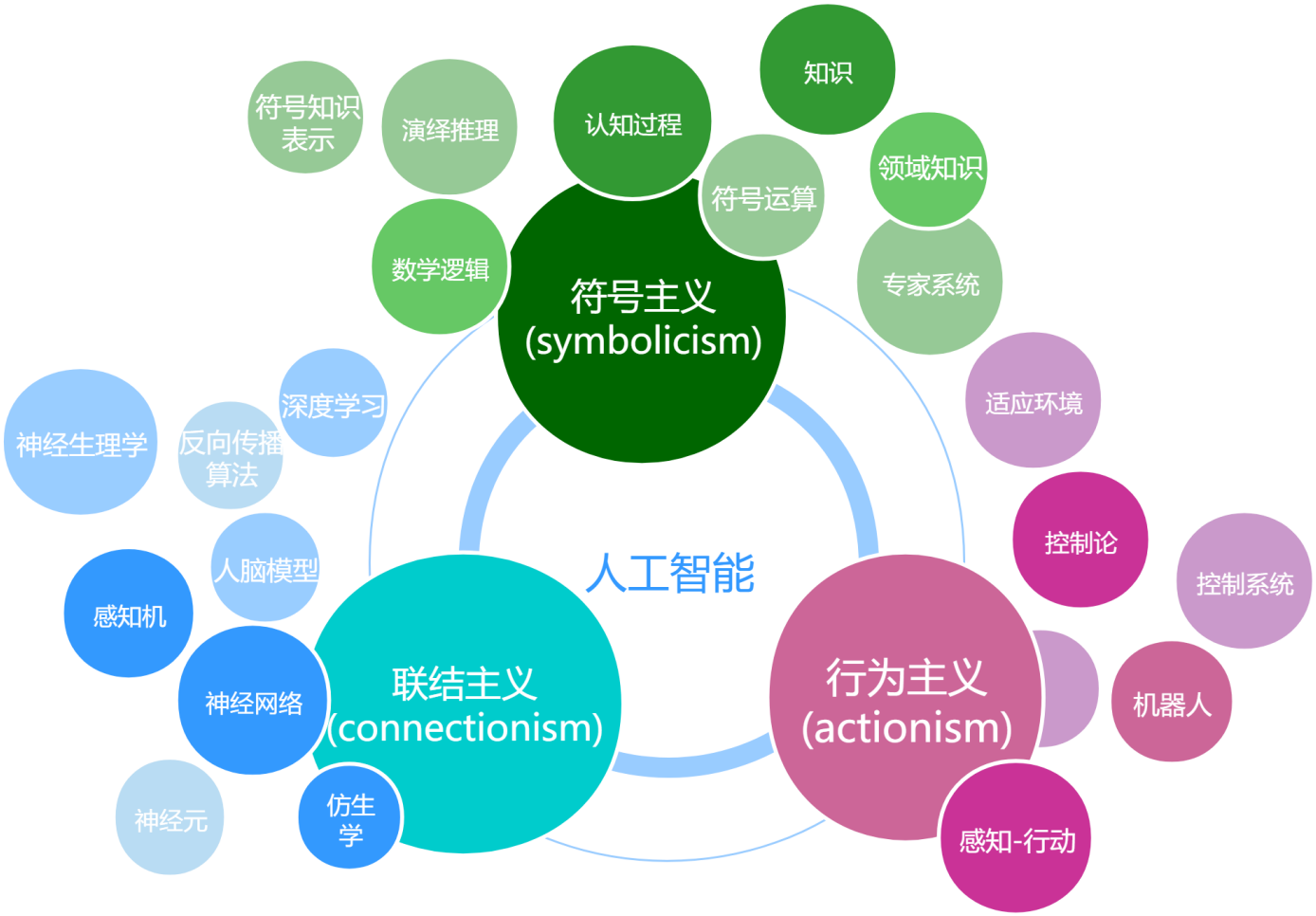
- 可靠性
- 可解释性
- 安全性
- 伦理问题

纲要

人工智能简史


2018
Alpha Zero，AlphaFold
2017
AlphaGo Zero
2016
DeepMind 的AlphaGo横空出世，深度强化学习闪耀登场
2000s
神经网络以深度学习的形式再次回归，人工智能进入第三个春天
1990s
受限于算力和数据，人工智能进入第二个"寒冬"
1980s
Hopfield网络、神经网络反向传播算法和专家系统让人工智能再次兴起
1970s
人工智能进入第一个"寒冬"
1956
达特茅斯会议，创造了一个时髦的新词："人工智能（AI）"
1950
图灵提出了"图灵测试"
1950
普林斯顿大四本科生马文明斯基和同学邓恩埃德蒙一起建造了世界上第一台神经网络计算机

人工智能流派


人工智能
符号主义
(symbolicism)
符号知识表示
演绎推理
认知过程
知识
符号运算
领域知识
数学逻辑
专家系统
联结主义
(connectionism)
神经生理学
深度学习
反向传播算法
人脑模型
感知机
神经网络
神经元
仿生学
行为主义
(actionism)
适应环境
控制论
控制系统
机器人
感知-行动

符号主义学派

- 符号主义学派认为人工智能源于数理逻辑，将人类认知和思维过程抽象成符号运算系统，认知过程就是在符号表示上的数学运算。
- 符号主义学派的代表人物有赫伯特·西蒙（Herbert Simon）、纽厄尔（Newell）和尼尔逊（Nilsson）等，其中赫伯特·西蒙是世界上唯一一位同时获得过图灵奖和诺贝尔经济学奖的科学家。
- 中国智能科学研究的开拓者和领军人、首届国家最高科学技术奖获得者吴文俊院士提出了初等几何和微分几何定理机器证明的理论和方法，吴文俊开创的数学机械化在国际上被誉为“吴方法”。由中国人工智能学会（Chinese Association for Artificial Intelligence，CAAI）发起、经科学技术部核准设立的“吴文俊人工智能科学技术奖”是中国历史上第一次以“人工智能”命名的奖项。

联结主义学派

- 联结主义学派的代表性人物是生理学家麦卡洛克（McCulloch）和数理逻辑学家皮茨（Pitts）。
- 皮茨等人提出的以感知机（Perceptron）为基础的脑模型是现代人工智能的基础，基于仿生学的思想模拟大脑神经元以及神经元之间的联结。
- 鲁梅尔哈特（Rumelhart）等人提出反向传播算法（BP）和近年来提出残差网络模型（ResNet）等。

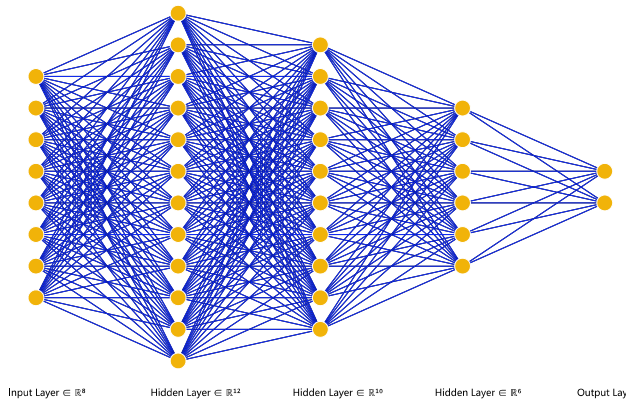


图 3: 神经网络模型

行为主义学派

- 20世纪末，行为主义引起许多人的兴趣。行为主义思想在20世纪50年代已经成熟，主要得益于控制论的发展。
- 控制论代表人物维纳（Norbert Wiener）在该领域做出了巨大贡献。
- 基于“感知—行动”的智能行为模拟方法，智能体能够适应环境变化，并基于环境状态做出智能决策。
- 深度强化学习方法与行为主义学派有着极强的联系，在思想层面有异曲同工之妙。
- 行为主义学派关注智能体在复杂动态环境中的最优化策略，获得最优回报。

纲要

人工智能基础

人工智能学科是众多理论和技术的结合体，横跨了多个学科领域，交叉融合，整合优化，互相促进，共同发展。

- 计算机
- 数学
- 物理学
- 心理学
- 认知科学
- 哲学

人工智能基础

图 4: 人工智能部分基础理论和技术

纲要

机器学习分类

机器学习方法分类有很多种，按照学习任务类型可以将机器学习分成三类：

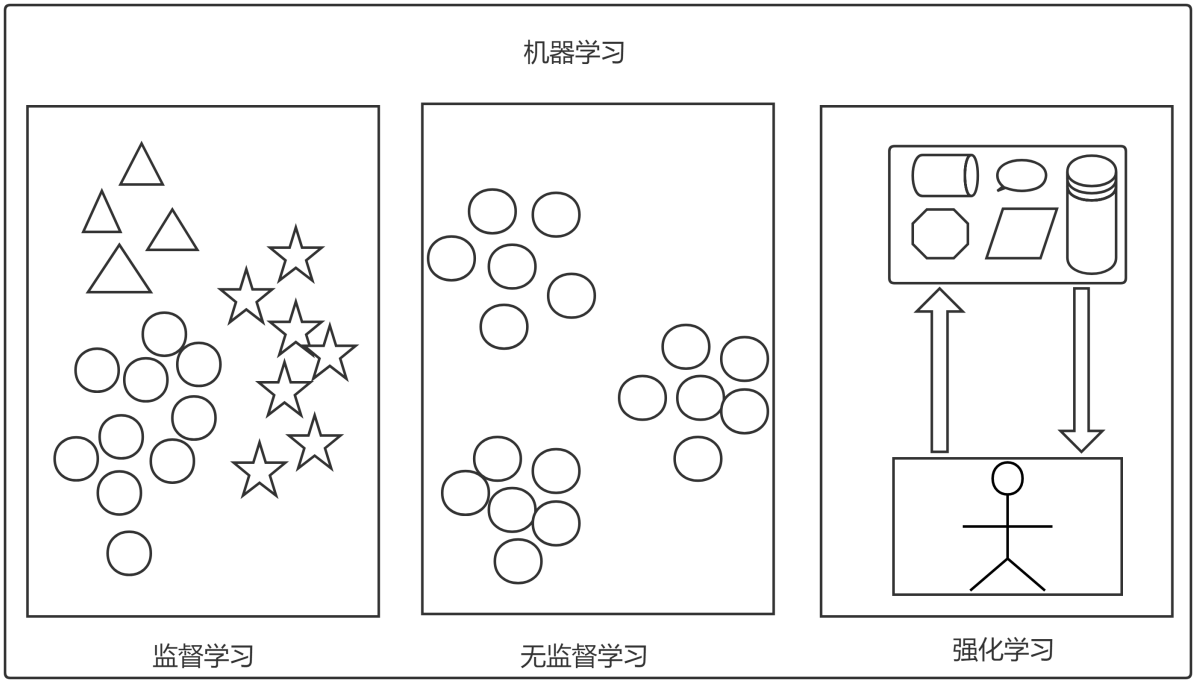


图 5: 机器学习分类示意图

机器学习模型

机器学习算法学到的是模型参数或者函数参数，机器学习模型可以看作是函数模型。我们用数学语言描述机器学习的过程，就是建模或学习一个函数映射的过程：

$$y = f_w(x), \tag{1}$$

其中f是模型，也是函数，w是模型参数，机器学习的目标就是从给定的样本数据(x, y)中学习到模型参数w。当然，如此简单描述并不严格，我们为了更容易理解机器学习过程而做了必要的简化。如在无监督学习中，样本数据(x, y)中的y未知，只有样本特征数据x。

监督学习

监督学习是人工智能和机器学习技术落地运用最广泛的学习算法。一般来说，监督学习的数据形式比较规整，如$\{(x_k, y_k)\}_{k=1,2,3,\ldots,N}$所示，其中$N$表示样本数量。监督学习是构建一个函数映射，将样本特征数据x映射到标签数据y。常见的监督学习任务可分为回归和分类。我们为了衡量监督学习算法的效果，构建基于均方误差（Mean Square Error）的目标函数：

$$\mathcal{L}(w) = \frac{1}{N}\sum_{k=1}^{N}\left(f_w(x_k) - y_k\right)^2. \tag{2}$$

该公式度量了标记数据y_k与模型预测$f_w(x_k)$之间的差异。均方误差目标函数并非监督学习中唯一的目标函数形式，均方误差一般适用于回归问题。

监督学习

一般来说，机器学习算法需要函数模型$y = f_w(x)$能够在训练集合上很好地拟合样本数据(x, y)，即函数模型的预测值$f_w(x_k)$与真实值y_k差距越小越好，目标函数$\mathcal{L}(w)$越小越好。我们确定好目标函数之后，通过优化算法最小化目标函数$\mathcal{L}(w)$，可以得到模型参数：

$$\hat{w} = \arg\min_w \mathcal{L}\left(w \middle| (x_k, y_k)_{k=1,2,3,\ldots,N}\right). \tag{3}$$

机器学习模型完成监督学习后，我们可以在验证集和测试集上对模型f_w进行验证和测试，验证和测试后的模型可以进行模型预测和模型生成等应用。

无监督学习

一般来说，无监督学习的样本数据如$\{x_k\}_{k=1,2,3,\ldots,N}$所示，其中$N$表示样本数量。经典多元统计学中有很多无监督学习的例子，如K均值聚类、系统聚类（层次聚类）、主成分分析（Principal Component Analysis，PCA）等。我们将以自编码器（Auto-Encoder）为例进行简单说明。为了衡量无监督学习算法的效果，同样构建一个基于均方误差的目标函数：

$$\mathcal{L}(w_1, w_2) = \frac{1}{N}\sum_{k=1}^{N}\left[f_{w_2}(f_{w_1}(x_k)) - x_k\right]^2. \tag{4}$$

我们为了模型简化，可以设定：

$$f_w(x) = f_{w_2}(f_{w_1}(x)), \tag{5}$$

其中$w = (w_1, w_2)$。

无监督学习

无监督学习虽然没有监督信号，即标记数据y_k，但在自编码器模型中样本x_k将自身x_k作为监督学习信号。自编码器模型学习一个恒等映射$f_w(x)$，即模型f_w将x映射到自身，满足

$$x = f_w(x), \qquad (6)$$

因此f_w叫作自编码器。自编码器模型结构的简单示例如图6所示。

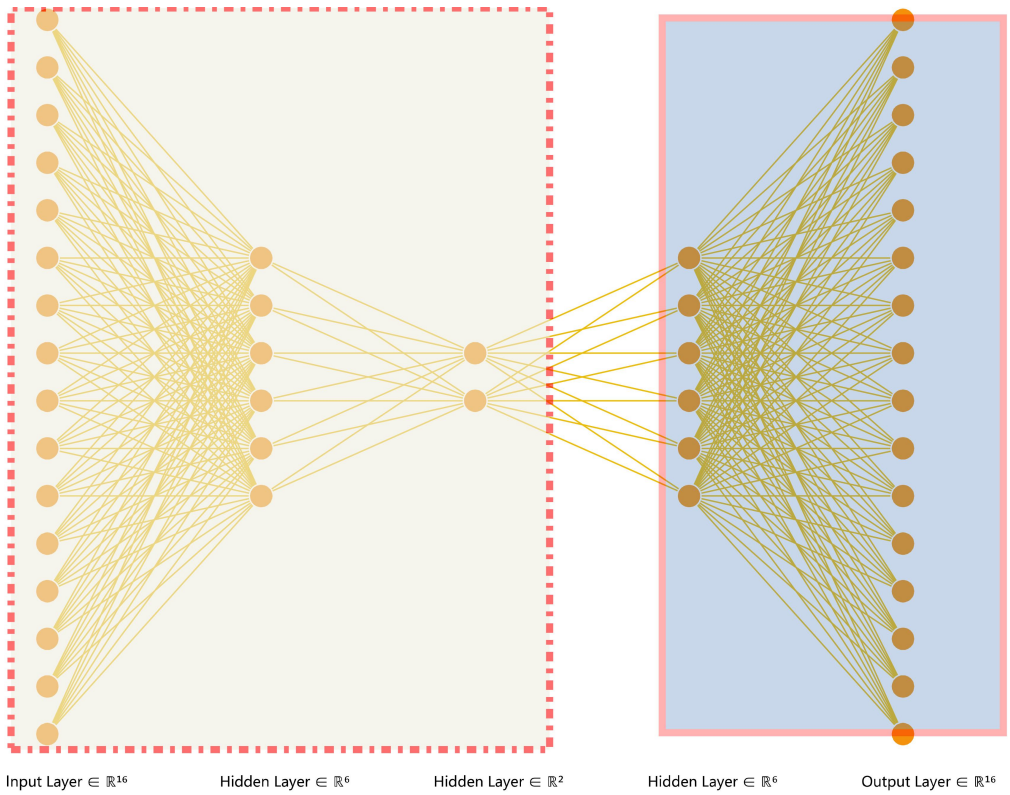


图 6: 自编码器模型结构示意图

无监督学习

同样，目标函数可以改写成：

$$\mathcal{L}(w) = \frac{1}{N}\sum_{k=1}^{N}\left(f_w(x_k) - x_k\right)^2. \tag{7}$$

确定好目标函数之后，我们通过优化方法最小化目标函数，可以得到模型参数：

$$\hat{w} = \arg\min_{w} \mathcal{L}\left(w|(x_k)_{k=1,2,3,\ldots,N}\right). \tag{8}$$

自编码器模型完成了无监督学习后，获得了两个子模型f_{w_1}和f_{w_2}，其中模型f_{w_1}可以看做是表示学习模型，对应的函数值$y_k = f_{w_1}(x_k)$可以是原始数据x_k的特征表示。模型f_{w_2}可以看做是生成模型，对应的函数值$x = f_{w_2}(y)$可以作为基于给定数据y生成的新样本数据。

自编码器的应用示例

自编码器还有很多其他应用，比如降噪。从全局视角来看，自编码器模型只是学习到了一个恒等映射，即

$$x = f_w(x) = f_{w_2}(f_{w_1}(x)). \tag{9}$$

如果我们给原始数据加上噪声ϵ，得到$x+\epsilon$，那么将其重新代入模型中后构建目标函数：

$$\mathcal{L}(w) = \frac{1}{N}\sum_{k=1}^{N}\left[f_w(x_k+\epsilon) - x_k\right]^2. \tag{10}$$

运用优化算法最小化目标函数，即可得到具有降噪功能的模型f_w或$f_{w_2}(f_{w_1}(x))$。自编码器模型f_w将具有噪声的数据$x+\epsilon$还原成了x，去掉了噪声ϵ。

强化学习

- 强化学习模型不同于监督学习和无监督学习，强化学习模型主要解决序贯决策问题。策略函数将环境状态空间映射到动作空间。
- 序贯决策问题是指目标函数值需要通过一系列关联的动作来确定，不是简单的通过一次或者互不关联的行为确定。
- 与监督学习和无监督学习类似，强化学习算法也是学习一个映射函数，即策略函数。
- 强化学习过程更加贴合人类学习过程，智能体通过与环境的交互来迭代优化策略函数。

深度强化学习与经典机器学习异同

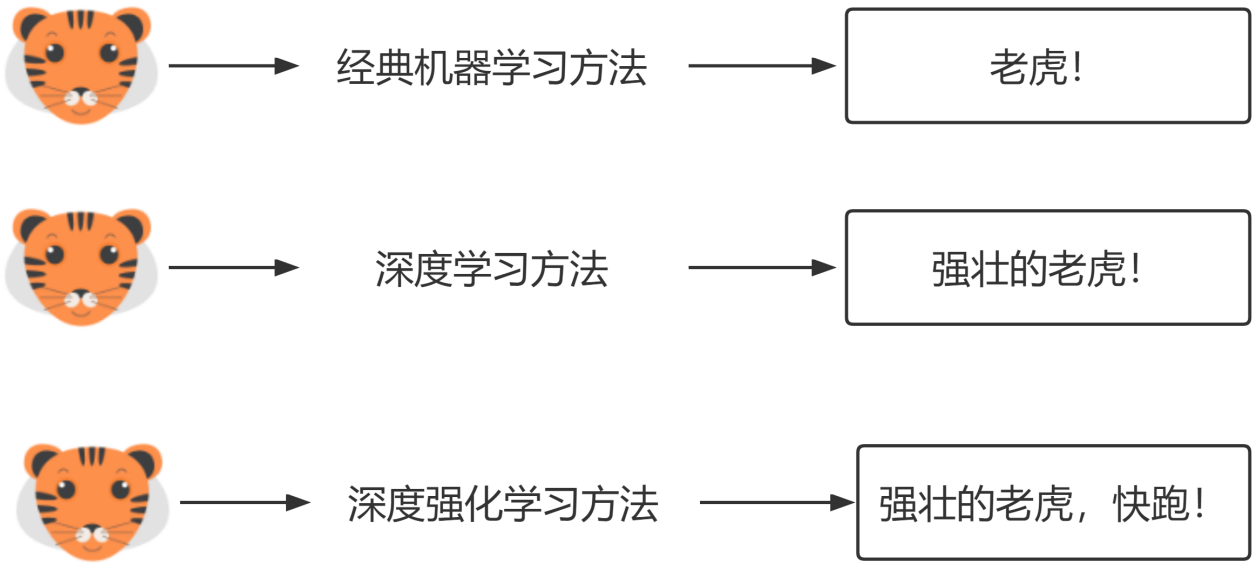


图 7: 深度强化学习与经典机器学习异同

纲要

激活函数

机器学习模型可理解成函数映射，一般不是简单函数或常见的线性函数，如多元线性回归模型。机器学习模型一般为非线性函数模型，或者是包含了非线性函数的嵌套函数等，其中激活函数是机器学习模型的重要组成部分。

- sigmoid函数（S型函数，也称为S型生长曲线函数）
- tanh函数（双曲正切函数，hyperbolic tangent function）
- ReLU（函数整流线性单元，Rectified Linear Unit）
- softmax函数等

激活函数应用示例

图6中神经网络模型的输入参数为x，x是一个列向量，大小为$n_x = 16$。神经网络模型下一层隐藏层有$n_h = 6$个神经元，隐藏层中每个神经元都基于$n_x = 16$个输入参数计算数值：

$$h = \sigma(W \cdot x + b), \tag{11}$$

其中W 为输入层和隐藏层之间的参数矩阵，大小为$n_h \times n_x$，偏置项 b 的大小为n_h，σ为非线性激活函数。

sigmoid函数

激活函数是机器学习中常见操作算子，常见的激活函数是sigmoid函数，数学函数表示如下

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}. \tag{12}$$

$\text{sigmoid}(x)$函数输出值介于0到1之间，既可以表示概率，也可以做分类标识，具有非常多的优良性质。神经网络模型计算梯度时，$\text{sigmoid}(x)$函数的导数为

$$\text{sigmoid}'(x) = \left(1 - \frac{1}{1+e^{-x}}\right)\frac{1}{1+e^{-x}} = (1-\text{sigmoid}(x))\,\text{sigmoid}(x). \tag{13}$$

因此，$\text{sigmoid}(x)$函数的导数仍然是$\text{sigmoid}(x)$函数的函数，在最优化过程中梯度计算较为简便。

tanh函数

激活函数$\tanh(x)$（双曲正切函数）为非线性函数，具体形式为

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{14}$$

$\tanh(x)$函数在机器学习模型中也经常使用，$\tanh(x)$函数输出值介于-1到1之间。同样，$\tanh(x)$函数具有较多优良性质，适合很多机器学习模型。

整流线性单元函数

在深度学习模型中，整流线性单元ReLU函数备受青睐，定义如下：

$$f(z) = \begin{cases} 0, & z \leq 0 \\ z, & z > 0 \end{cases} . \tag{15}$$

ReLU函数的优势显而易见，极其简单，却有着深刻含义。相较于$\mathrm{sigmoid}(x)$函数和$\tanh(x)$函数，ReLU函数的非线性特征并不明显，只是一个分段函数。但是在一些机器学习任务中ReLU函数表现出了较好的性能，如图像识别任务等。ReLU函数的导数同样极其简单：

$$f'(z) = \begin{cases} 0, & z < 0 \\ 1, & z > 0 \end{cases} . \tag{16}$$

但ReLU函数在0点处不可导。

Softmax函数

深度神经网络模型最后一层（输出层）一般不使用上述激活函数，模型输出层经常使用softmax函数进行归一化，使得神经网络模型输出一个概率分布向量，具体形式如下

$$f(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}}, \tag{17}$$

其中，N为输出层神经元数量，i为输出层神经元编号。神经网络输出层为一个概率分布向量，适合在分类任务中使用。

损失函数

均方误差损失：

一般而言，我们用神经网络模型的输出值和目标值之间的差异来构造损失函数，损失越小，模型的输出值和目标值之间的差异越小，说明模型参数越好。监督学习模型的损失函数可以表示为：

$$\mathcal{L}(w) = \frac{1}{N}\sum_{k=1}^{N}\left(f_w(x_k) - y_k\right)^2, \tag{18}$$

其中y_k为目标值，样本x_k为神经网络模型输入，w为神经网络模型参数，模型预测值为$f_w(x_k)$，而损失函数则为目标值y_k和预测值$f_w(x_k)$之间的误差平方和，最后进行平均。损失函数（18）称作均方误差。

平均绝对误差损失（Mean Absolute Error， MAE）

平均绝对误差（Mean Absolute Error, MAE）损失函数：

$$\mathcal{L}(w) = \frac{1}{N}\sum_{k=1}^{N}|f_w(x_k) - y_k|\,. \tag{19}$$

均方误差和平均绝对误差损失函数都可以用来作为目标函数，优化模型参数更新。均方误差具有较好的性质，如可导。均方误差损失函数方便计算梯度下降所需要的偏导数，而平均绝对误差损失函数在一些数据点不可导。

交叉熵损失（Cross Entropy Loss， CEL）

在机器学习模型训练集中有N个样本的情况下，K分类问题的交叉熵公式定义为：

$$H(P,Q) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} P(x_{ik}) \log Q(x_{ik}). \tag{20}$$

在机器学习模型中，一般分类问题将交叉熵公式作为模型损失函数，即目标函数。机器学习模型对估计分布函数$Q(x)$进行模型化和参数化，如建模成深度神经网络，参数为w，则K分类问题的交叉熵损失函数可写作：

$$\mathcal{L}(w) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} P(x_{ik}) \log Q_w(x_{ik}). \tag{21}$$

优化损失函数参数

我们通过最优化方法，找到最优参数w使得损失函数最小：

$$\hat{w} = \arg\min_{w} \mathcal{L}(w). \tag{22}$$

机器学习模型完成模型训练后，分类模型$Q_w(x)$可以估计新样本x属于K个分类的概率，并完成新样本分类任务。

如何优化目标函数？

优化算法

机器学习模型确定损失函数后，损失函数包含了机器学习模型的参数θ，参数估计过程就是一个典型的优化问题，最小化目标函数或损失函数$\mathcal{L}(\theta)$，估计最优参数θ^*，使得$\mathcal{L}(\theta^*)$最小，图8给出了简化版的模型参数优化过程示意图。

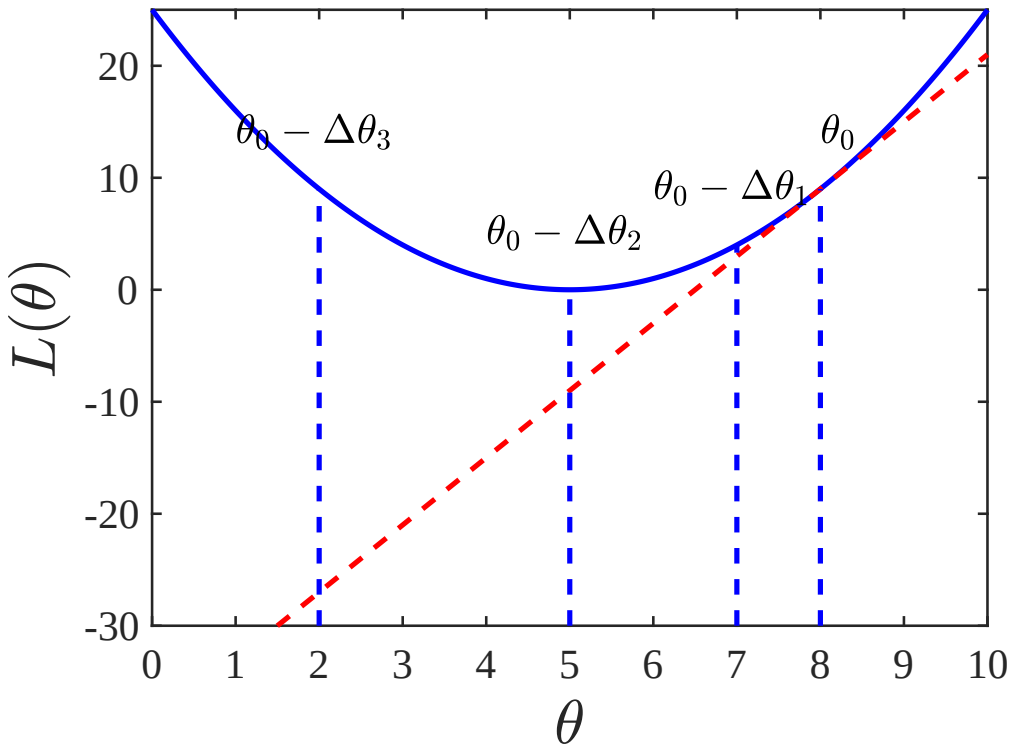


图 8: 机器学习模型参数优化过程示意图

梯度

在机器学习模型优化过程中，梯度是一个非常重要的概念。若函数$f(x,y,z)$为三元函数，则函数在点(x,y,z)处的梯度为向量

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right), \tag{23}$$

其中偏导数在点(x,y,z)处都可计算，梯度∇f 为目标函数f在此处变化最快的方向。图8示例为一维空间，梯度方向即为导数方向。目标函数在随机初始值θ_0附近，按照导数方向增加参数θ可以使得目标函数越来越大。损失函数优化过程中，我们需要损失函数越小越好，因此参数θ的变化方向应该为导数或梯度的反方向，即目标函数或损失函数可以运用梯度下降算法更新参数。

随机梯度下降（Stochastic Gradient Descent，SGD）

在机器学习或深度学习模型中，随机梯度下降（Stochastic Gradient Descent，SGD）算法是最常用的参数更新算法，伪代码如下所示：

Algorithm 1: 随机梯度下降算法伪代码

Input: 损失函数 $\mathcal{L}$，机器学习模型参数 $\boldsymbol{\theta}$，学习率 α，最大训练次数 S

Output: 最优参数 $\boldsymbol{\theta}^*$

初始化模型参数 $\boldsymbol{\theta}$

for $k = 0, 1, 2, 3, \cdots, S$ **do**

  随机抽样小批量样本计算损失函数 $\mathcal{L}$

  通过反向传播算法计算目标函数梯度 $\nabla_{\boldsymbol{\theta}}\mathcal{L}$

  梯度下降方法更新参数 $\boldsymbol{\theta}$：$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}$

返回最优参数 $\boldsymbol{\theta}^* = \boldsymbol{\theta}_k$

动量随机梯度下降（Momentum SGD）算法

借鉴物理系统中物体运动的动量概念，科研人员改进随机梯度下降（SGD）算法，设计了包含动量的随机梯度下降算法，即动量随机梯度下降算法，其伪代码如下所示：

Algorithm 2: 动量随机梯度下降算法伪代码

Input: 损失函数 $\mathcal{L}$，模型参数 $\boldsymbol{\theta}$，学习率 α，超参数 β，最大训练次数 S

Output: 最优模型参数 $\boldsymbol{\theta}^*$

初始化 $g_0 = 0$

for $k = 1, 2, 3, \cdots, S$ **do**

  随机抽样小批量样本计算损失函数 $\mathcal{L}$

  通过反向传播算法计算梯度 $\nabla_{\boldsymbol{\theta}} \mathcal{L}$

  结合动量思想，更新梯度时叠加上一次的梯度方向：$g_k = \beta g_{k-1} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}$

  更新参数 $\boldsymbol{\theta}$：$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + g_k$

返回最优参数 $\boldsymbol{\theta}^*$

Nesterov动量随机梯度下降（Nesterov Momentum SGD）算法

Nesterov动量随机梯度下降算法对动量随机梯度下降算法进行了改进，在计算梯度时可以使用提前位置的梯度进行更新，Nesterov动量随机梯度下降算法的伪代码如下所示：

Algorithm 3: Nesterov 动量随机梯度下降算法伪代码

Input: 损失函数 $\mathcal{L}$，模型参数 $\boldsymbol{\theta}$，学习率 α，超参数 β，最大训练次数 S

Output: 最优模型参数 $\boldsymbol{\theta}^*$

初始化 $g_0 = 0$

for $k = 1, 2, 3, \cdots, S$ **do**

  随机抽样小批量样本计算损失函数 $\mathcal{L}$

  根据上一次迭代梯度计算临时模型参数 $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}_{k-1} + \beta g_{k-1}$

  通过反向传播算法和临时参数计算梯度 $\boldsymbol{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1} + \beta g_{k-1})$

  结合动量思想更新梯度，考虑上一次更新的方向：$g_k = \beta g_{k-1} - \alpha \boldsymbol{\nabla}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1} + \beta g_{k-1})$

  更新参数 $\boldsymbol{\theta}$：$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + g_k$

返回最优参数 $\boldsymbol{\theta}^*$

自适应梯度下降（Adagrad）算法

随机梯度下降算法中，梯度决定了参数更新的方向，超参数学习率α决定了参数更新的步长大小。在地势平坦的地方，梯度较小，可以加大移动步长；在地势比较陡峭的地方，梯度较大，可以减小移动步长。自适应梯度下降（Adagrad）算法伪代码如下所示：

Algorithm 4: 自适应梯度下降算法伪代码

Input: 损失函数 $\mathcal{L}$，模型参数 $\boldsymbol{\theta}$，学习率 α，最大训练次数 S，非常小的常数 ϵ

Output: 最优模型参数 $\boldsymbol{\theta}^*$

初始化 $t_0 = 0$

for $k = 1, 2, 3, \cdots, S$ **do**

  抽样小批量样本计算损失函数 $\mathcal{L}$

  通过反向传播算法计算梯度 $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})$

  更新参数 t：$t_k = t_{k-1} + \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})\right)^2$

  更新参数 $\boldsymbol{\theta}$：$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \dfrac{\alpha}{\sqrt{t_k + \epsilon}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})$

返回最优参数 $\boldsymbol{\theta}^*$

RMSprop梯度下降（RMSprop）算法

Hinton等人为了避免自适应梯度下降算法中梯度越来越小的问题，提出了RMSprop梯度下降算法，改进了Adagrad算法。RMSprop梯度下降算法伪代码如下所示：

Algorithm 5: RMSprop 梯度下降算法伪代码

Input: 损失函数 $\mathcal{L}$，模型参数 $\boldsymbol{\theta}$，学习率 α，超参数 γ，最大训练次数 S

Output: 最优模型参数 $\boldsymbol{\theta}^*$

初始化 $t_0 = 0$

for $k = 1, 2, 3, \cdots, S$ **do**

  抽样小批量样本计算损失函数 $\mathcal{L}$

  通过反向传播算法计算梯度 $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})$

  更新参数 t：$t_k = \gamma t_{k-1} + (1-\gamma)\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})\right)^2$

  更新参数 $\boldsymbol{\theta}$：$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \dfrac{\alpha}{\sqrt{t_k + \epsilon}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})$

返回最优参数 $\boldsymbol{\theta}^*$

Adadelta梯度下降算法

Adadelta算法为了进一步解决超参数学习率动态调整的问题，继续进行了改进，不需要设定初始化学习率α，具体算法伪代码如下所示：

Algorithm 6: Adadelta 梯度下降算法伪代码

Input: 损失函数 $\mathcal{L}$，模型参数 $\boldsymbol{\theta}$，超参数 γ，最大训练次数 S

Output: 最优模型参数 $\boldsymbol{\theta}^*$

初始化 $g_0 = 0$

初始化 $t_0 = 0$

初始化 $\Delta_0 = 0$

for $k = 1, 2, 3, \cdots, S$ **do**

  抽样小批量样本计算损失函数 $\mathcal{L}$

  通过反向传播算法计算梯度 $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})$

  更新参数 t：$t_k = \gamma t_{k-1} + (1-\gamma)\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})\right)^2$

  更新参数 g：$g_k = -\dfrac{\sqrt{\Delta_{k-1} + \epsilon}}{\sqrt{t_k + \epsilon}} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_{k-1})$

  更新参数 $\boldsymbol{\theta}$：$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + g_k$

  更新参数 Δ：$\Delta_k = \gamma \Delta_{k-1} + (1-\gamma) g_k^2$

返回最优参数 $\boldsymbol{\theta}^*$

Adam梯度下降算法

Adam 梯度下降算法融合了诸多算法的精髓，成为了机器学习领域常用的优化算法，具体算法伪代码如下所示：

Algorithm 7: Adam 梯度下降算法伪代码

Input: 损失函数 $\mathcal{L}$，模型参数 $\boldsymbol{\theta}$，学习率 α，超参数 β_1，β_2，最大训练次数 S

Output: 最优模型参数 $\boldsymbol{\theta}^*$

初始化 $g_0 = 0$

初始化 $t_0 = 0$

初始化 $\Delta_0 = 0$

令 $t = 0$

for $k = 1, 2, 3, \cdots, S$ **do**

- $t = t + 1$
- 抽样小批量样本计算损失函数 $\mathcal{L}$
- 通过反向传播算法计算梯度 $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_{k-1})$
- 更新 g：$g_k = \boldsymbol{\nabla}_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_{k-1})$
- 更新 m：$m_k = \beta_1 m_{k-1} + (1-\beta_1) g_k$
- 更新 v：$v_k = \beta_2 v_{k-1} + (1-\beta_2) g_k^2$
- 更新 $\hat{m}$：$\hat{m}_k = \dfrac{m_k}{1-\beta_1^t}$
- 更新 $\hat{v}$：$\hat{v}_k = \dfrac{v_k}{1-\beta_2^t}$
- 更新参数 $\boldsymbol{\theta}$：$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \dfrac{\alpha}{\sqrt{\hat{v}_k + \epsilon}} \hat{m}_k$

返回最优参数 $\boldsymbol{\theta}^*$

纲要

基于机器学习的复杂网络分析方法

近年来，基于机器学习的复杂网络分析方法迅速发展，网络嵌入或图嵌入方法得到了大量研究者关注：

- 网络嵌入（Network Embedding，NE）
- 图嵌入（Graph Embedding，GE）
- 网络表示学习（Network Representation Learning, NRL）

算法的基本分类：

- 基于矩阵分解
- 基于随机游走

高阶邻近保留嵌入算法

高阶邻近保留嵌入算法（High-Order Proximity preserved Embedding，HOPE）是能够保留有向图的不对称传递性的网络嵌入算法。

高阶邻近性源自不对称传递性，采用机器学习优化算法最小化损失函数：

$$\min \|\mathcal{S} - \mathcal{U}^s \mathcal{U}^{t\top}\|^2 \tag{24}$$

$\mathcal{S}$是可视图网络高阶相似性测度指标值，$\mathcal{U}^s$和$\mathcal{U}^t$是每个网络节点嵌入向量组成的嵌入向量矩阵，高阶邻近保留嵌入算法与传统的奇异值分解（Singular Value Decomposition，SVD）方法类似。

高阶邻近保留嵌入算法

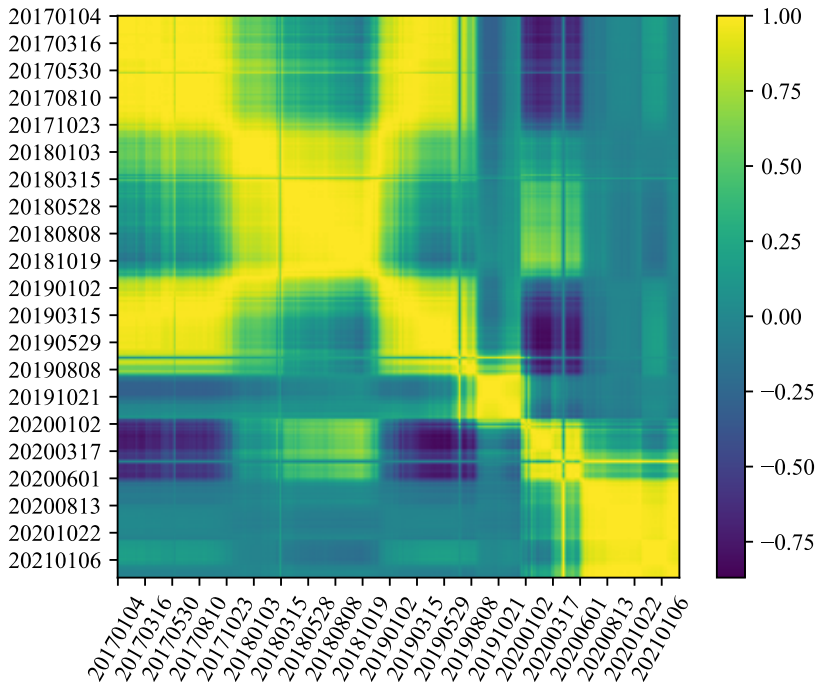


图 9: 恒生指数时间序列通过可视图算法转成网络后，网络嵌入算法（HOPE）对可视图网络进行表示学习。图中热度图为表示学习的特征向量的相似性矩阵。

掌握的问题

1. 什么是人工智能？
2. 举例人工智能技术在复杂金融系统中的应用。
3. 简要阐述人工智能的历史。
4. 简要阐述人工智能的三个主要学派。
5. 人工智能的基础理论和技术有哪些？
6. 一般机器学习可分成哪三类学习范式？
7. 强化学习、监督学习和无监督学习三者的联系和区别是？
8. 机器学习中有哪些常用的激活函数？
9. 机器学习中有哪些常用的损失函数？
10. 机器学习中有哪些常用的优化算法？