计算智能

计算智能是信息科学、生命科学、认知科学等不同学科相互交叉的产物. 它主要借鉴仿生学的思想, 基于人们对生物体智能机理的认识, 采用数值计算的方法去模拟和实现人类的智能. 计算智能的主要研究领域包括: 神经计算、进化计算、模糊计算、免疫计算、DNA 计算和人工生命等.

计算智能 (CI) 到目前还没有一个统一的的定义, 使用较多的是美国科学家贝慈德克 (J. C. Bezdek) 从计算智能系统角度给出定义:

贝慈德克的定义

如果一个系统仅处理低层的数值数据, 含有模式识别部件, 没有使用人工智能意义上的知识, 且具有计算适应性、计算容错力、接近人的计算速度和近似于人的误差率这 4 个特性, 则它是计算智能的.

从学科范畴看, 计算智能是在神经网络 (Neural Networks, NN)、进化计算 (Evolutionary Computation, EC) 及模糊系统 (Fuzzy System, FS) 这 3 个领域发展相对成熟的基础上形成的一个统一的学科概念.

神经计算

神经网络是一种对人类智能的结构模拟方法, 它是通过对大量人工神经元的广泛并行互联, 构造人工神经网络系统去模拟生物神经系统的智能机理的.

进化计算是一种对人类智能的演化模拟方法, 它是通过对生物遗传和演化过程的认识, 用进化算法去模拟人类智能的进化规律的.

模糊计算

模糊计算是一种对人类智能的逻辑模拟方法, 它是通过对人类处理模糊现象的认知能力的认识, 用模糊逻辑去模拟人类的智能行为.

计算智能的特点 (从贝慈德克对计算智能的定义和上述计算智能学科范畴分析)

- 第一, 计算智能是借鉴仿生学的思想, 基于生物神经系统的结构、进化和认知对自然智能进行模拟的.

计算智能的特点 (从贝慈德克对计算智能的定义和上述计算智能学科范畴分析)

- 第二, 计算智能是一种以模型 (计算模型、数学模型) 为基础, 以分布和并行计算为特征的自然智能模拟方法.

计算智能与人工智能的关系

目前

一种观点认为计算智能是人工智能的一个子集;
另一种观点认为计算智能和人工智能是不同的范畴.

第一种观点的代表人物是贝慈德克. 他把智能 (Intelligence, I) 和神经网络 (Neural Network, NN) 都分为计算的 (Computational, C)、人工的 (Artificial, A) 和生物的 (Biological,B) 3 个层次, 并以模式识别 (PR) 为例, 给出了下图所示的智能的层次结构.

在图1中, 底层是计算智能 (CI), 它通过数值计算来实现, 其基础是 CNN；中间层是人工智能 (AI), 它通过人造的符号系统实现, 其基础是 ANN；顶层是生物智能 (BI), 它通过生物神经系统来实现, 其基础是 BNN.

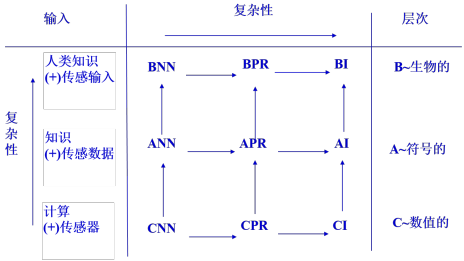


图 1: 贝慈德克的智能的 3 个层次

计算智能的产生与发展

按照贝慈德克的观点

CNN 是指按生物激励模型构造的 NN, ANN 是指 CNN+ 知识, BNN 是指人脑, 即 ANN 包含了 CNN, BNN 又包含了 ANN. 对智能也一样, 贝慈德克认为 AI 包含了 CI, BI 又包含了 AI, 即计算智能是人工智能的一个子集.

第二种观点——大多数学者所持有的观点

其代表人物是艾伯哈特 (R. C. Eberhart). 他们认为: 虽然人工智能与计算智能之间有重合, 但计算智能是一个全新的学科领域, 无论是生物智能还是机器智能, 计算智能都是其最核心的部分, 而人工智能则是外层.

事实上, CI 和传统的 AI 只是智能的两个不同层次, 各自都有自身的优势和局限性, 相互之间只应该互补, 而不能取代.

大量实践证明, 只有把 AI 和 CI 很好地结合起来, 才能更好地模拟人类智能, 才是智能科学技术发展的正确方向.

1992 年, 贝慈德克在《Approximate Reasoning》学报上首次提出了“计算智能”的概念.《Neural network-based approximate reasoning: principles and implementation》, IJC [**Nie207179208934320**].

WCCI’94

1994 年 6 月底到 7 月初, IEEE 在美国佛罗里达州的奥兰多市召开了首届国际计算智能大会 (简称 WCCI’94). 会议第一次将神经网络、进化计算和模糊系统这三个领域合并在一起, 形成了“计算智能”这个统一的学科范畴.

WCCI’98

在此之后, WCCI 大会就成了 IEEE 的一个系列性学术会议, 每 4 年举办一次. 1998 年 5 月, 在美国阿拉斯加州的安克雷奇市又召开了第 2 届计算智能国际会议 WCCI’98.

WCCI’02

2002 年 5 月, 在美国夏威夷州首府火奴鲁鲁市又召开了第 3 届计算智能国际会议 WCCI’02. IEEE 出版了计算智能有关的刊物.

目前, 计算智能的发展得到了国内外众多的学术组织和研究机构的高度重视, 并已成为智能科学技术一个重要的研究领域.

神经网络的综合基础第二版

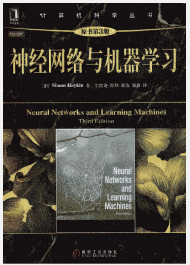
经典参考教材-2001 第二版

《神经网络的综合基础》, 第二版, 国际知名大学原版教材——信息科学学科与电气工程学科系列, Simon Haykin (S. Haykin 是一位物理、数学、信号处理和通信的科学家, 编著的著作有 50 多本), 2001, 清华大学出版社/培生教育出版集团, 1-836. Simon Haykin

神经网络的综合基础的第三版

《神经网络与机器学习》(Neural Networks and Learning Machines), Simon Haykin, 申富饶 / 徐烨 / 郑俊 / 晁静译, 机械工业出版社, 2011-3, 572, ISBN: 9787111324133

《随机信号处理》, New Directions in Statistical Signal Processing-From Systems to Brains. Simon Haykin, Jose C. Principe, Terrence J. Sejnowski, and John McWhirter, MIT press, 2006



Haykin-ECE 772 Graduate《Neural Networks and Learning Machines》

Statistical learning theory, including VC, regularization, and Bayesian theories. Algorithms for multilayer perceptrons, kernel-based learning machines, self-organizing maps, principal components analysis, and blind source separation. Sequential state estimation algorithms, including extended Kalman filter, unscented Kalman filter, and particle filters; applications to learning machines.

Haykin-ECE 775 Graduate《Cognitive Dynamic Systems》

Cognition. Neural information processing. Spectrum sensing. Bayesian filtering for state estimation. Cognitive dynamic programming for control. Cognitive radar. Cognitive radio Self-organizing systems.

神经计算及算法学习

神经计算

神经计算或叫神经网络, 是计算智能的重要基础和核心, 也是计算智能乃至智能科学技术的一个重要研究领域.

神经网络是受生物神经元启发构建的计算系统. 神经网络由许多独立的单元组成, 每个单元接收来自上一层单元的输入, 并将输出发送到下个单元（「单元」不一定是单独的物理存在；它们可以被认为是计算机程序的不同组成部分）. 单元的输出通常通过取输入的加权和通过某种简单的非线性转型, 神经网络的关键特性是基于经验修改与单元之间的链接比较相关权重.

常见误解——StuartRussell

「神经网络是一种新型计算机」

在实践中, 几乎所有的神经网络都运行在普通的计算机架构上. 一些公司正在设计专用机器, 它们有时会被称作是「神经计算机」, 可以有效地运行神经网络, 但目前为止, 这类机器无法提供足够的优势, 值得花费大量时间去开发.

「神经网络像大脑一样工作」

事实上, 生物神经元的工作方式比神经网络复杂得多, 自然界存在很多种不同的神经元, 神经元的连接可以随时间进行改变, 大脑中也存在其他的机制, 可以影响动物的行为.

神经计算基础

生物神经系统是人工神经网络的基础. 人工神经网络是对人脑神经系统的简化、抽象和模拟, 具有人脑功能的许多基本特征. 它由细胞体 (Soma)、轴突 (Axon) 和树突 (Dendrite) 三个主要部分组成.

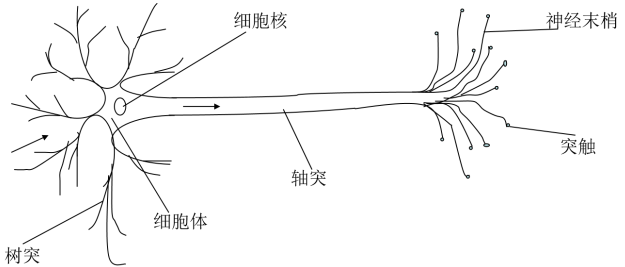


图 2: 突触的结构

细胞膜的外面是许多向外延伸出的纤维.

轴突

轴突是由细胞体向外延伸出的所有纤维中最长的一条分枝, 用来向外传递神经元产生的输出电信号.

- 每个神经元都有一条轴突, 其最大长度可达 1m 以上. 在轴突的末端形成了许多很细的分枝, 这些分支叫神经末梢.

- 每一条神经末梢可以与其它神经元形成功能性接触, 该接触部位称为突触. 所谓功能性接触, 是指非永久性的接触, 这正是神经元之间传递信息的奥秘之处.

- 树突是指由细胞体向外延伸的除轴突以外的其它所有分支. 树突的长度一般较短, 但数量很多, 它是神经元的输入端, 用于接受从其它神经元的突触传来的信号.

生物神经元的功能

根据神经生理学的研究, 生物神经元的 2 个主要功能是: 神经元的兴奋与抑制, 神经元内神经冲动的传导.

神经元的抑制与兴奋

抑制/兴奋状态

抑制是指神经元在没有产生冲动时的工作状态. 兴奋状态是指神经元产生冲动时的工作状态.

电神经机理

通常情况下, 神经元膜电位约为-70 毫伏, 膜内为负, 膜外为正, 处于抑制状态. 当神经元受到外部刺激时, 其膜电位随之发生变化, 即膜内电位上升、膜外电位下降, 当膜内外的电位差大于阈值电位 (约 +40 毫伏) 时, 神经元产生冲动而进入兴奋状态.

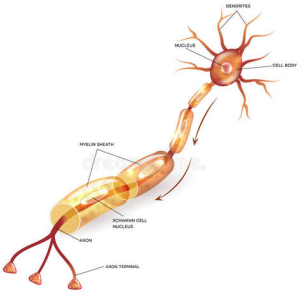
神经元每次冲动的持续时间大约 1 毫秒左右, 在此期间, 即使刺激强度再增加也不会引起冲动强度的增加. 神经元每次冲动结束后, 都会重新回到抑制状态. 如果神经元受到的刺激作用不能使细胞膜内外的电位差大于阈值电位, 则神经元不会产生冲动, 将仍处于抑制状态.

神经元内神经冲动的传导

神经冲动在神经元内的传导是一种电传导过程, 神经冲动沿神经纤维传导的速度却在 3.2-320km/s 之间, 且其传导速度与纤维的粗细、髓鞘的有无有一定关系.

人脑神经系统的联结机制

一般来说, 有髓鞘的纤维的传导速度较快, 而无髓鞘的纤维的传导速度较慢.



人脑神经系统的联结规模

人类大脑中由 860–1012 亿个神经元所组成, 其中每个神经元大约有 3×104 个突触. 小脑中的每个神经元大约有 105 个突触, 并且每个突触都可以与别的神经元的一个树突相连. 人脑神经系统就是由这些巨量的生物神经元经广泛并行互连所形成的一个高度并行性、非常复杂的神经网络系统.

人脑神经系统的分布功能

人脑神经系的记忆和处理功能是有机的结合在一起的, 每个神经元既具有存储功能, 同时又具有处理能力. 从结构上看, 人脑神经系统又是一种分布式系统.

人工神经网络的互连结构

协同工作、相互影响

人们通过对脑损坏病人所做的神经生理学研究, 没有发现大脑中的哪一部分可以决定其余所有各部分的活动, 也没有发现在大脑中存在有用于驱动和管理整个智能处理过程的任何中央控制部分. 人类大脑的各个部分是协同工作、相互影响的. 在大脑中, 不仅知识的存储是分散的, 其控制和决策也是分散的.

工神经网络 ANN

人工神经网络是由大量的人工神经元经广泛互联所形成的一种人工网络系统, 用以模拟人类神经系统的结构和功能.

MP 神经元模型

人工神经元是对生物神经元的抽象与模拟, 图7是一个 MP 神经元模型, 它由细胞体 (Soma)、轴突 (Axon) 和树突 (Dendrite) 三个主要部分组成.

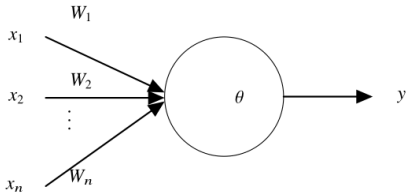


图 4: MP 神经元模型

常用的人工神经元模型——根据不同功能函数划分出的模型:

- 阈值型 (Threshold)

这种模型的神经元没有内部状态, 作用函数 f 是一个阶跃函数, 他表示激活值 σ 和输出之间的关系.

- 分段线性强饱和型 (Linear Saturation)

这种模型又称为伪线性, 其输入/输出之间在一定范围内满足线性关系, 一直延续到输出为最大值 1 为止. 但当达到最大值后, 输出就不再增.

激活函数 1

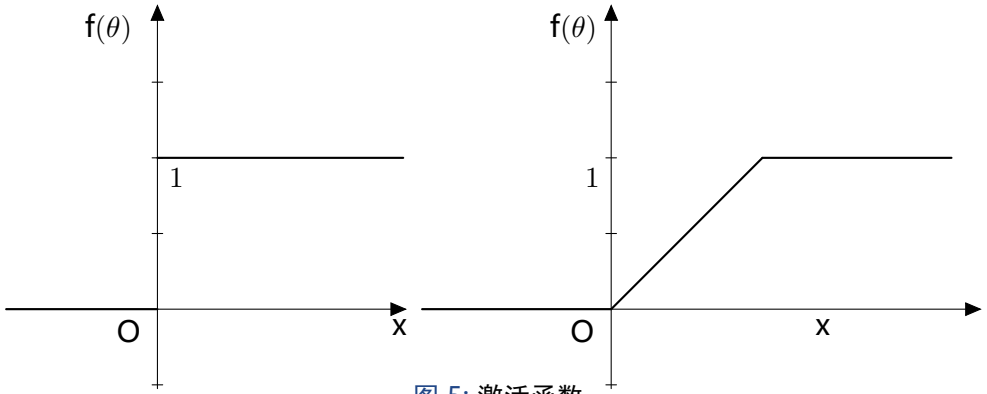


图 5: 激活函数

- S 型 (Sigmoid)

这是一种连续的神经元模型, 其输入输出特性常用指数、对数或双曲正切等 S 型函数表示. 它反映的是神经元的饱和特性.

- 子阈累积型 (Subthreshold Summation)

也是一个非线性函数, 当产生的激活值超过阈值 T 值时, 该神经元被激活产生反响. 在线性范围内, 系统的反响是线性的.

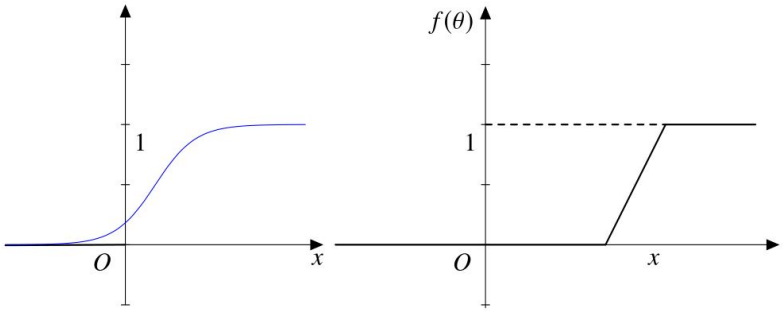


图 6: S(Sigmoid) 型和子阈累积型

互连结构

人工神经网络的互连结构 (或称拓扑结构) 是指单个神经元之间的连接模式, 它是构造神经网络的基础, 也是神经网络诱发偏差的主要来源.

从互连结构的角度看

仅含输入层和输出层, 且只有输出层的神经元是可计算节点. 除拥有输入和输出层外, 还至少含有一个、或更多个隐含层的前向网络.

$$y_j = f\left(\sum_{i=1}^{n} w_{ij}x_i - \theta_j\right), j = 1, 2, \cdots, m \tag{1}$$

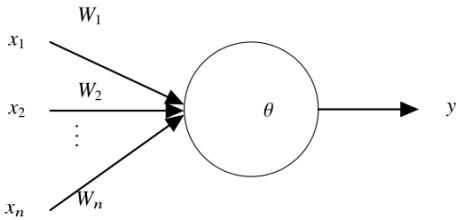


图 7: 激活函数

前馈网络

前馈网络

指不拥有隐含层的反馈网络, 包括单层前馈网络和多层前馈网络.

单层前馈网络

单层前馈网络是指那种只拥有单层计算节点的前向网络. 它仅含有输入层和输出层, 且只有输出层的神经元是可计算节点, 如图 8 所示

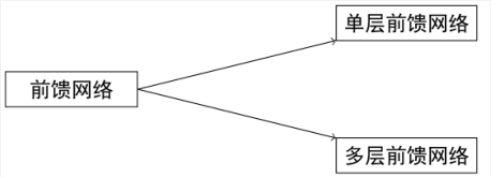


图 8: 单层前馈网络结构

其中, 输入向量为 $X = (x_1, x_2, \cdots, x_n)$；输出向量为 $Y = (y_1, y_2, \cdots, y_m)$；输入层各个输入到相应神经元的连接权值分别是 $w_{ij}, i = 1, 2, .., n, j = 1, 2, .., m.$

若假设各神经元的阈值分别是 $\theta_j, j = 1, 2, \cdots, m$, 则各神经元的输出 $y_j, j = 1, 2, \cdots, m$ 分别为:

$$y_j = f\left(\sum_{i=1}^{n} w_{ij} x_i - \theta_j\right), \quad j = 1, 2, \ldots, m \tag{2}$$

其中, 由所有连接权值 w_{ij} 构成的连接权值矩阵 W 为:

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots w_{1m} \\ w_{21} & w_{22} & \cdots w_{2m} \\ \vdots & \vdots & \vdots \\ w_{nl} & w_{n2} & \cdots w_{nm} \end{pmatrix} \tag{3}$$

在实际应用中, 该矩阵是通过大量的训练示例学习而形成的.

多层前馈网络

多层前馈网络是指那种除拥有输入、输出层外, 还至少含有一个、或更多个隐含层的前馈网络.

隐含层

隐含层是指由那些既不属于输入层又不属于输出层的神经元所构成的处理层, 也被称为中间层. 隐含层的作用是通过对输入层信号的加权处理, 将其转移成更能被输出层接受的形式.

多层前馈网络结构

多层前馈网络的输入层的输出向量是第一隐含层的输入信号, 而第一隐含层的输出则是第二隐含层的输入信号, 以此类推, 直到输出层. 多层前馈网络的典型代表是 BP 网络.

反馈网络

反馈网络是指允许采用反馈联结方式所形成的神经网络 (图9). 所谓反馈联结方式是指一个神经元的输出可以被反馈至同层或前层的神经元.

单层前馈网络结构

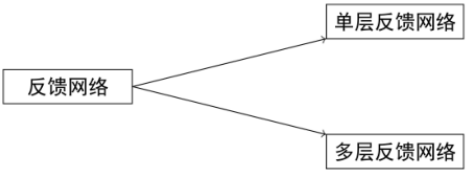


图 9: 单层前馈网络结构

反馈网络和前向网络不同

前向网络

属于非循环连接模式, 它的每个神经元的输入都没有包含该神经元先前的输出, 因此不具有“短期记忆”的性质.

反馈网络则不同, 它的每个神经元的输入都有可能包含有该神经元先前输出的反馈信息, 即一个神经元的输出是由该神经元当前的输入和先前的输出这两者来决定的, 这就有点类似于人类的短期记忆的性质.

反馈网络的典型例子是后面将要介绍的 Hopfield 网络.

人工神经网络的模型

网络模型

是指对网络结构、联结权值和学习能力的总括.

常用的网络模型

- 传统的感知机模型,
- 具有误差反向传播功能的反向传播网络模型,
- 采用多变量插值的径向基函数网络模型,
- 建立在统计学习理论基础上的支撑向量机网络模型.

感知机 (Perceptron) 模型

- 采用反馈联接方式的反馈网络模型,

- 基于模拟退火算法的随机网络模型.

感知器是美国学者罗森勃拉特 (Rosenblatt) 于 1957 年为研究大脑的存储、学习和认知过程而提出的一类具有自学习能力的神经网络模型, 其拓扑结构是一种分层前向网络.

单层感知器

单层感知器是一种只具有单层可调节连接权值神经元的前向网络, 这些神经元构成了单层感知器的输出层, 是感知器的可计算节点.

在单层感知器中, 每个可计算节点都是一个线性阈值神经元. 当输入信息的加权和大于或等于阈值时, 输出为 1, 否则输出为 0 或 -1.

单层感知器的输出层的每个神经元都只有一个输出, 且该输出仅与本神经元的输入及联接权值有关, 而与其他神经元无关.

若假设各神经元的阈值分别是 $\theta_j, j = 1, 2, \cdots, m$, 则各神经元的输出 $y_j, j = 1, 2, \cdots, m$ 分别为

$$y_j = f\left(\sum_{i=1}^{n} w_{ij} x_i - \theta_j\right) \quad j = 1, 2, \ldots, m \tag{4}$$

其中, 由所有连接权值 w_{ji} 构成的连接权值矩阵 W 为:

实际应用中, 该矩阵是通过大量的训练示例学习而形成的如下的权重矩阵:

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix} \tag{5}$$

输入向量为 $X = (x_1, x_2, \cdots, x_n)$；输出向量为 $Y = (y_1, y_2, \cdots, y_m)$；
输入层各个输入到相应神经元的连接权值分别是 $w_{ij}, i = 1, 2, \cdots, n, j = 1, 2, \cdots, m$.
感知器是为了对外部输入进行分类.

罗森勃拉特已经证明, 如果外部输入是线性可分的 (指存在一个超平面可以将它们分开), 则单层感知器一定能够把它划分为两类. 其判别超平面由如下判别式确定:

$$\sum_{i=1}^{n} w_{ij} x_i - \theta_j = 0 \quad j = 1, 2, \ldots, m \tag{6}$$

单个感知器实现逻辑运算的问题

单层感知器可以很好地实现“与”、“或”、“非”运算, 但却不能解决“异或”问题.

Example

“与”运算 ($x_1 \wedge x_2$)

表 1: 与运算

输入	输入	输出	超平面	阈值条件
x_1	x_2	$x_1 \wedge x_2$	$w_1 \times x_1 + w_2 \times x_2 - \theta = 0$	$\theta = 0$
0	0	0	$w_1 \times 0 + w_2 \times 0 - \theta < 0$	$\theta > 0$
0	1	0	$w_1 \times 0 + w_2 \times 1 - \theta < 0$	$\theta > w_2$
1	0	0	$w_1 \times 1 + w_2 \times 0 - \theta < 0$	$\theta > w_1$
1	1	1	$w_1 \times 0 + w_2 \times 1 - \theta \geq 0$	$\theta \leq w_1 + w_2$

可以证明此表有解.

例如取 $w_1 = 1$, $w_2 = 1$, $\theta = 1.5$, 其分类结果如图 **??** 所示, 其中, 输出为 1 的用实心圆表示, 输出为 0 的用空心圆表示. 后面约定相同.

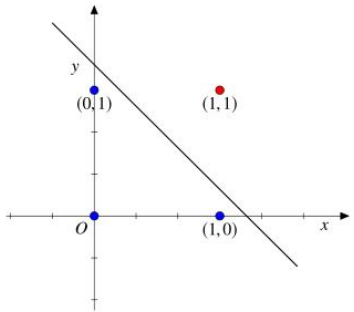


图 10: "与"问题

“非”运算

Example

“非”运算 ($\neg x_1$)

例如取 $w_1 = -1$, $\theta = -0.5$, 其分类结果如右图所示.

表 2: 初始种群情况表

输入	输出	超平面	阈值条件
x_1	$\neg x_1$	$w_1 \times x_1 - \theta = 0$	$\theta = 0$
0	1	$w_1 \times 0 - \theta \geq 0$	$\theta \geq 0$
1	0	$w_1 \times 1 - \theta < 0$	$\theta < -w_1$

Example

“异或”运算 (x_1 XOR x_2)

表 3: 初始种群情况表

输入	输出	超平面	阈值条件	
x_1	x_2	x_1 XOR x_2	$w_1 * x_1 + w_2 * x_2 - \theta = 0$	$\theta = 0$
0	0	0	$w_1 \times 0 + w_2 \times 0 - \theta < 0$	$\theta > 0$
0	1	1	$w_1 \times 0 + w_2 \times 1 - \theta \geq 0$	$\theta \leq w_2$
1	0	1	$w_1 \times 1 + w_2 \times 0 - \theta \geq 0$	$\theta \leq w_1$
1	1	0	$w_1 \times 1 + w_2 \times 1 - \theta < 0$	$\theta > w_1 + w_2$

此表无解, 即无法找到满足条件的 w_1, w_2 和 θ, 如图 11 所示. 因为异或问题是一个非线性可分问题, 需要用多层感知器来解决.

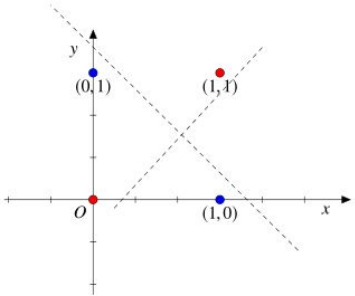
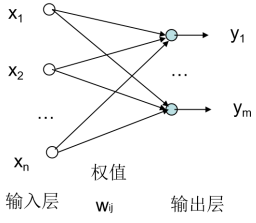


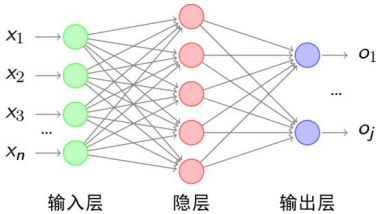
图 11: 异或问题的可分问题

多层感知器

多层感知器是通过在单层感知器的输入、输出层之间加入一层或多层处理单元所构成的.

其拓扑结构与图12所示的多层前向网络相似, 差别也在于其计算节点的连接权值是可变的.





多层感知器的输入与输出之间是一种高度非线性的映射关系, 如图12所示的多层前向网络, 若采用多层感知器模型, 则该网络就是一个从 n 维欧氏空间到 m 维欧氏空间的非线性映射.

多层感知器可以实现非线性可分问题的分类.

Example

对“异或”运算, 用图13所示的多层感知器即可解决.

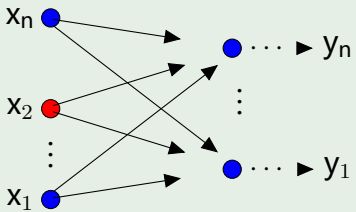


图 13: 多层感知网络

在图13中, 隐层神经元 x_{11} 所确定的直线方程为

$$1 \times x_1 + 1 \times x_2 - 0.5 = 0, \tag{7}$$

它可以识别一个半平面.

隐层神经元 x_{12} 所确定的直线方程为

$$1 \times x_1 + 1 \times x_2 - 1.5 = 0, \tag{8}$$

它也可以识别一个半平面.

输出层神经元所确定的直线方程为

$$1 \times x_{11} + 1 \times x_{12} - 1.5 = 0. \tag{9}$$

它相当于对隐层神经元 x_{11} 和 x_{12} 的输出作“逻辑与”运算, 因此可识别由隐层已识别的两个半平面的交集所构成的一个凸多边形, 如图14所示.

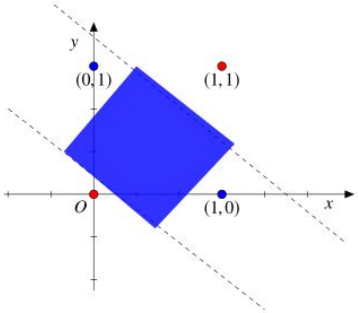


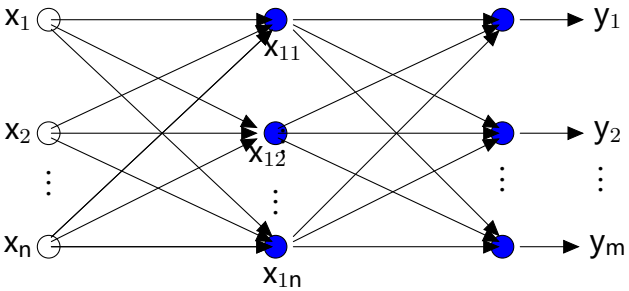
图 14: 可分区域

误差反向传播 (BP) 模型

误差反向传播 (Error Back Propagation) 网络

通常简称为 BP(Back Propagation) 网络, 是由美国加州大学的鲁梅尔哈特和麦克莱兰在研究并行分布式信息处理方法, 探索人类认知微结构的过程中, 于 1985 年提出的一种网络模型.

BP 网络的网络拓扑结构是多层前向网络, 如图15所示.



输入层 权可调 隐层 权可调 输出层

图 15: 多层 BP 网络的结构

在 BP 网络中, 同层节点之间不存在相互连接, 层与层之间多采用全互连方式, 且各层的连接权值可调.

BP 网络实现了明斯基的多层网络的设想, 是当今神经网络模型中使用最广泛的一种.

BP 网络的特点

- 第一, BP 网络的每个处理单元均为非线性输入/输出关系, 其作用函数通常采用的是可微的 Sigmoid 函数, 如:

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{10}$$

- 第二, BP 网络的学习过程是由工作信号的正向传播和误差信号的反向传播组成的. 所谓正向传播, 是指输入模式经隐层到输出层, 最后形成输出模式；所谓误差反向传播, 是指从输出层开始逐层将误差传到输入层, 并修改各层联接权值, 使误差信号为最小的过程.

反馈网络 (Hopfield) 模型

Hopfield 网络是由美国加州工学院物理学家霍普菲尔特 1982 年提出来的一种单层全互连的对称反馈网络模型. 它可分为离散 Hopfield 网络和连续 Hopfield 网络, 限于篇幅, 本书重点讨论离散 Hopfield 网络.

离散 Hopfield 网络是在非线性动力学的基础上由若干基本神经元构成的一种单层全互连网络, 其任意神经元之间均有连接, 并且是一种对称连接结构. 一个典型的离散 Hopfidld 网络结构如图16所示.

一个典型的离散 Hopfidld 网络结构如图16所示.

离散 Hopfield 网络模型是一个离散时间系统, 每个神经元只有 0 和 1(或-1 和 1) 两种状态, 任意神经元 i 和 j 之间的连接权值为 w_{ij}.

由于神经元之间为对称连接, 且神经元自身无连接, 因此有

$$w_{ij} = \begin{cases} w_{ji} & i \neq j \\ 0 & i = j \end{cases} \tag{11}$$

由该连接权值所构成的连接矩阵是一个对角为零的对称矩阵.

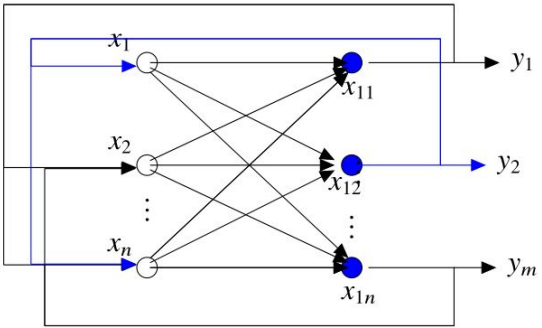


图 16: 离散 Hopfield 网络

最优剪枝极值学习机

在 Hopfidld 网络中, 虽然神经元自身无连接, 但由于每个神经元都与其他神经元相连, 即每个神经元的输出都将通过突触连接权值传递给别的神经元, 同时每个神经元又都接受其他神经元传来的信息.
这 Hopfidld 网络中的每个神经元来说, 其输出经过其他神经元后又有可能反馈给自己, 因此 Hopfidld 网络是一种反馈神经网络.

最优剪枝极值学习机

（OP-ELM）如图17, 图17显示了 OP-ELM 的三个步骤. 提出了 (OP-ELM 工具箱找到.

如 [**Miche2008OP**, **Miche2008A**, **Miche2010OP**] 中的描述, 对于 OP-ELM, 当给网络添加一个纯随机噪声变量（添加的噪声变量没有在图上显示）时, 拟合会变得松散和扩散. 实际上, ELM 的并没为这种情况设计. 本文的 OP-ELM 提出了一种三个步骤的方法来解决这个问题, 简言之, 为了解决含噪声的问题, 需要给 ELM 加一个含修剪策略的封装器, 之后增加了两个步骤.

首先是通过最小角度回归对神经元进行排序 (LARS [**efronhaste2004-32498**]);

实际上, 实现了 LARS 算法的 MRSR 算法也适用于多输出情况, 根据它们在输出值大小对它们进行排序.

然后, 用一个漏掉一个的准则来确定在最终的 OP-ELM 模型结构中需要保留多少个被分类的神经元.

LARS algorithm 的其实现来自 Lasso 方法, 对于 OP-ELM, 它提供了对隐藏神经元的排序方法,
这是由于 OP-ELM 算法设计总的神经元和输出之间的关系是线性的.

为了优化参数, (LOO) 方法通常是一种代价高昂的方法, 在整个数据集（只除去一个样本）上训练模型, 并在此样本上对数据集的所有样本重复求值.

在 OP-ELM 结构中, 隐藏层和输出层的结点是线性关系, LOO 误差有一个封闭的矩阵形式, 由 Allen 的预测平方和 (PRESS) 给出 (关于压 LOO 误差的计算细节见第 4 节). 这种闭合形式允许快速计算均方误差, 从而计算输出权重 **b**, 使得 OP-ELM 在计算上仍然有效, 并且比原始 ELM 对不相关/相关变量的鲁棒性更强, 因此, OP-ELM 可以被看作是一个“正则化”ELM.

通过使用一个 LARS 方法, 这里是对回归问题的 L_1 约束. 同时, 回归问题使用 L_1 和 L_2 (也包括 L_1 和 L_2 的联合形式) 惩罚项.

同时, 由于在压力公式中执行的矩阵运算的性质 (这些计算见第 4 节), 决定保留的神经元的最终数量（通过 LOO criteria）显示出潜在的不稳定性（数值）.

本文提出的解决方案是在 PRESS 公式的计算中使用正则化.

回顾用于执行正则化的最著名算法, 回归问题使用 L_1 和 L_2 (也包括 L_1 和 L_2 的联合形式) 惩罚项.

第 4 节中提出的方法结合了 OP-ELM 中 L_1 和 L_2 的约束, 使网络正规化.

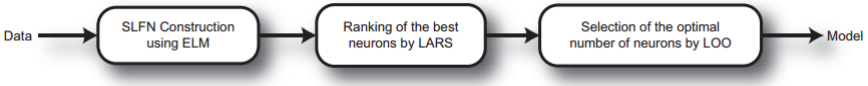


图 17: OP-ELM 的三部分框架结构

Tikhonov 正则化

Tikhonov 正则化

以 Andrey Tikhonov 的名字命名, 是不适定问题最常用的正则化方法之一.

在统计学上, 这种方法被称为岭回归, 在有多个独立发现的情况下, 它也被称为 **Tikhonov – Miller method**、**Phillips – Twomey method**, 约束线性反演方法和线性正则化索引方法. 它与针对非线性回归最小二乘问题的 Levenberg-Marquardt 算法有关.

最小二乘线性回归

假设对于已知的矩阵 **A** 和向量 **b**, 我们希望找到向量 **x**, 使得下式成立

$$\mathbf{Ax} = \mathbf{b}. \tag{12}$$

然而, 如果没有 **x** 满足等式, 或者有多个 x 满足等式, 则解决方案不是唯一的, 该问题称为不适定问题. 在这种情况下, 普通最小二乘估计会导致方程组的超定 (过拟合), 或者更常见的是欠定 (欠拟合).

大多数现实世界的现象都具有向前方向的低通滤波器的效果, 其中 A 将 **x** 映射到 **b**. 因此, 在求解逆问题时, 逆映射用作高通滤波器, 具有放大噪声的不良倾向 (特征值/奇异值在逆映射中最大, 而在正向映射中最小).

此外, 普通的最小二乘法隐式地将重构版本 **x** 的每个元素 (在 A 的空空间中) 置空, 不允许将模型用作 **x** 的前置.

最小二乘法

寻求最小化残差平方和, 其紧凑格式可写为

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2. \tag{13}$$

其中 $\|\cdot\|$ 是欧几里德范数.

为了得到具有期望性质的特定解, 对于适定的 Tikhonov 矩阵 Γ, 可在最小化问题中包括正则化项：

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \|\Gamma\mathbf{x}\|^2. \tag{14}$$

L_2 正则化方法

许多情况下, 该矩阵被选为恒等矩阵 $(\Gamma = \alpha \mathrm{I})$, 优先选择具有较小范数的解; 这是熟知的 L_2 正则化方法 [**Ng2004-32636**].

在其他情况下, 则可以使用 **lowpass operators**(例如, 差分运算符或 **weighted Fourier operator**) 来强制平滑.

这种正则化改进了问题的解, 从而使直接求得数值解成为可能. 由 $\hat{\mathrm{x}}$ 表示的显式解由以下公式给出:

$$\hat{\mathrm{x}} = (\mathbf{A}^{\mathsf{T}}\mathbf{A} + \Gamma^{\mathsf{T}}\Gamma)^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{b}. \tag{15}$$

正则化的效果可以通过矩阵 Γ 的尺度变化. 对于 $\Gamma = 0$, 如果存在 $(A^TA)^{-1}$, 则问题退化为未正则化的最小二乘解.

除了线性回归之外, L_2 正则化还用于许多上下文中, 例如使用 logistic 回归或支持向量机分类 [**Fan2008LIBLINEAR**] 以及矩阵因式分解 [**Guan2012Online**].

Tikhonov 正则化是在许多不同的上下文中独立发明的. 从 Andrey Tikhonov 和 David L. Phillips 的工作中, 它在积分方程中的应用广为人知. 有些作者使用术语 “Tikhonov-Phillips 正则化”. Arthur E.Hoerl 和 Manus Foster 采用统计方法, 阐述了有限维情况, 并将此方法解释为一个 **Wiener-Kolmogorov**（**Kriging**）滤波器. 在 Hoerl 之后, 它在统计文献中被称为岭回归.

对于 **x** 的一般多元正态分布和数据误差, 可以应用变量转换来减少上述情况. 同样地, 可以寻找 **x** 以最小化

$$\|\mathbf{Ax} - \mathbf{b}\|_{\mathbf{P}}^2 + \|\mathbf{x} - \mathbf{x}_0\|_{\mathbf{Q}}^2 \tag{16}$$

其中, 我们使用 $\|\mathbf{x}\|\,\mathbf{Q}^2$ 表示加权范数 $\mathbf{x}^{\mathrm{T}}\mathbf{Qx}$（与 Mahalanobis 距离比较）.

贝叶斯方法将 **P** 解释为 **b** 的逆协方差矩阵, $\mathbf{x}_0$ 是 x 的期望值, 而 **Q** 是 **x** 的逆协方差矩阵. 然后, 将 Tikhonov 矩阵作为矩阵 $\mathbf{Q} = \Gamma^{\mathrm{T}}\Gamma$（例如, **Cholesky factorization**）的因子分解给出, 并将其视为一个 **whitening filter**.

这个广义问题有一个最优解 x, 可以用公式显式地求解

$$\mathbf{x}^* = (\mathbf{A}^\mathsf{T}\mathbf{P}\mathbf{A} + \mathbf{Q})^{-1}(\mathbf{A}^\mathsf{T}\mathbf{P}\mathbf{b} + \mathbf{Q}\mathbf{x}_0). \tag{17}$$

等价表示

$$\mathbf{x}^* = \mathbf{x}_0 + (\mathbf{A}^\mathsf{T}\mathbf{P}\mathbf{A} + \mathbf{Q})^{-1}(\mathbf{A}^\mathsf{T}\mathbf{P}(\mathbf{b} - \mathbf{A}\mathbf{x}_0)). \tag{18}$$

TROP-ELM: 基于 LARS 和 Tikhonov 正则化的双正则化 ELM[**MichevanHeeswijk2011-30641**], 2011. 图 18 显示了建议的正则化 OP-ELM（TROP-ELM).

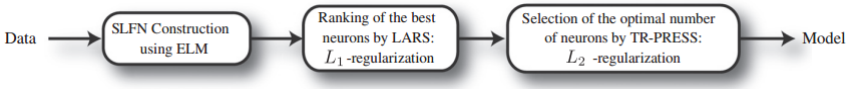


图 18: TROP-ELM.

Allen’ s PRESS(prediction sum of squares)

原始 PRESS 公式

OP-ELM 中使用的原始 PRESS 公式是由 Allen 在 [**MADavid1972**] 中提出. 原始的 PRESS 公式可以表示为

$$\mathrm{MSE}^{\text{TR-PRESS}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \mathbf{x}_i(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}^T y}{1 - \mathbf{x}_i(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i^T}\right)^2, \tag{19}$$

从公式可以看出, 每个观测值都是用其他 $n-1$ 观测值“预测”的, 残差最终被平方和相加. 算法通过矩阵运算, 有效地实现了该公式.

算法 1

1: 计算矩阵的逆 $\mathbf{C} = (\mathbf{X}^{\mathsf{T}}\mathrm{X})^{-1}$
2: 计算 $\mathbf{P} = \mathbf{XC}$;
3: 计算伪逆 $\mathbf{w} = \mathbf{CX}^{\mathsf{T}}\mathbf{y}$;
4: 计算 PRES 的分子 $\mathrm{D} = \mathbf{1} - \mathrm{diag}(\mathbf{PX}^{\mathsf{T}})$;
5: 计算 PRESS 的误差 $\mathrm{e} = \frac{\mathrm{y} - \mathbf{Xw}}{\mathbf{D}}$;
6: 优化误差 $\mathrm{MSE}^{\text{TR-PRESS}} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}}^{2}$.

方法的主要缺点

在于在计算中使用伪逆 (**Moore–Penrose sense** 中), 如果数据集 **X** 不是全秩, 则可能导致数值不稳定性. 不幸的是, 现实世界中的数据集经常出现这种情况. 下面的方法对原始的计算提出了两个改进: 正则化和快速矩阵计算.

Tikhonov 正则化的 PRESS(TR-PRESS)

在 [**golub1979 generalized**] 中, Golub 等人注意到: 使用奇异值分解（SVD）方法计算 PRESS 的统计结果比传统的伪逆方法更好.

在同一篇论文中, 提出了 Allen’ s PRESS 的一个推广, 即广义交叉验证（GCV）方法, 它在技术上优于原始的方法, 因为它可以处理数据定义非常糟糕的情况, 例如, 如果除对角线项外, 所有的 X 项都是 0.

实际上, 从实验来看, 虽然 GCV 具有无可替代的优越性, 和原始的 PRESS 和 Tikhonov 正则化的 PRESS 相比, 它会导致计算时间陡增.

矩阵形式

算法 1 以给出了用于确定 $\mathrm{MSE}^{\text{TR-PRESS}}$ 的计算步骤

$$\mathrm{MSE}^{\text{TR-PRESS}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \mathbf{x}_i(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{x}^T y}{1 - \mathbf{x}_i(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{x}_i^T}\right)^2 \tag{20}$$

这是公式（18）的正规化结果. 符号 $A \circ B$ 表示矩阵 $\mathbf{A}$ 和 $\mathbf{B}$ (Schur 积) 逐元素相乘. 在算法 2 的步骤 4 中使用它, 因为它比标准矩阵乘积快.

用 SVD 分解 $\mathbf{X}$: $\mathbf{X} = \mathbf{USV}^{\mathsf{T}}$;
计算后面将要用到的乘积: $\mathrm{A} = \mathbf{XV}$ and $\mathrm{B} = \mathbf{U}^{\mathsf{T}}\mathbf{y}$;
重复如下步骤:
对 $\mathbf{X}$ 用 SVD 分解, 按如下方式计算矩阵 $\mathbf{C}$:
计算 $\mathbf{P}$: $\mathbf{P} = \mathbf{CB}$;
计算 $\mathbf{D}$: $\mathbf{D} = 1 - \mathrm{diag}(\mathbf{CU}^{\mathsf{T}})$;
计算 $\mathrm{e} = \frac{\mathrm{y}-\mathbf{P}}{\mathbf{D}}$ 和 $\mathrm{MSE}^{\text{TR-PRESS}} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \epsilon_{\mathrm{i}}^{2}$.
直到 λ 收敛.
关联 $\mathrm{MSE}^{\text{TR-PRESS}}$ 和 λ 的最优值.

注解

在全局上, 该算法使用 **X** 的 SVD 来避免计算问题, 并在 SVD 计算伪逆时引入 Tikhonov 正则化参数. 由于在优化 λ 之前预先计算了效用矩阵 (**A**、**B** 和 **C**), 所以此特定实现碰巧运行得非常快.

在实际应用中, 该算法中 λ 的优化是通过一种 **Nelder**–**Mead** 最小化方法来实现的, 这种方法恰好在这个问题上收敛得非常快 (在 Matlab 中是 fminsearch 函数). 通过使用这个修改版的 PRESS, OP-ELM 对回归权重 (隐藏层和输出层之间的回归) 有 L_2 的惩罚, 对于这些权重, 神经元已经使用 L_1 的惩罚进行了排序. 图 18 是图的修改版本. 图 17 说明了 **TROP-ELM** 方法.

使用新的学习范式来训练 RVFL 网络

训练集说明

在这个新的训练集中, $\mathbf{x}_i \in X$ 是原始功能, 一般来说, 特权功能 $\mathbf{x}_i \in \tilde{x}$ 属于特权功能空间 $\tilde{x}$, 它不同于原始功能空间 x.

RVFL 的模型结构如此简单, 为什么 RVFL 能很好地完成大多数任务? Giryes 等人 [**Giryes2016-7439822**]. 对这个开放问题给出一个可能的理论解释.

Giryes 等人揭示了学习算法训练的本质 [**Giryes2016-7439822**]. 一般来说, 不同类别之间的样本角度大于同一类别内的样本角度 [**LiorWolf2003**]. 因此, 训练在学习算法中的作用是“不同类别的点之间的角度比同一类别的点之间的角度受到的惩罚更大”[**Giryes2016-7439822**].

RVFL 回顾

在这个大数据时代, 由于模型结构高度复杂, 学习过程中很难对所有参数进行快速调整, 这就需要极高的计算成本.

为了解决这个问题, 随机化是一些学习算法的理想选择, 从而降低了计算成本. 一个好的随机初始化允许学习算法是一个通用的训练前 [**Glorot2015**, **Zhang2015**].

RVFL 回顾

RVFL 使用混合策略来训练整个网络.
在 RVFL 中, 输入层和增强节点之间的随机初始参数处理具有可分辨角度的良好输入样本, 旋转输出权重进一步处理剩余数据.

RVFL 网络

是一种经典的单层前向神经网络, 其结构如图19所示.

RVFL 随机初始化输入层和增强节点之间的所有权重和偏差, 然后这些参数在训练阶段不需要调整.

RVFL 的网络结构

图19中红色实线代表的输出权重可以通过 Moore-Penrose 伪逆 ([**paopillips1995-6471**, **IgelnikPao1995-6470**]) 或岭回归 (引用 Bishop2012-6469) 来计算.

输入层和输出层之间的直接连接是防止 RVFL 网络过度拟合的一种有效而简单的正则化技术.

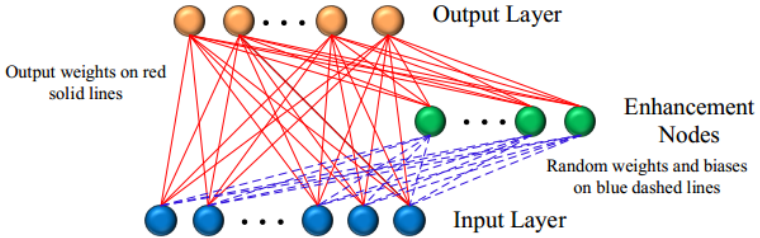


图 19: RVFL 的网络结构

RVFL+

RVFL+

是一种用于所有二元、多类别分类和回归问题的统一学习算法, 并且 (19) 中的输出函数可以直接应用于所有三种情况.

1. 二元分类: 测试样品的预测标签由 $\hat{y} = \text{sign}(f_{\text{test}}(z))$.
2. 多类分类: 我们采用一对多 (One vs All) 策略来确定多类分类中的预测标签. 设 $f_{\text{test}}^{k}(z)$ 为第 k 个输出节点的输出函数. 测试样品的预期标签由

$$\hat{y} = \arg \max_{k \in 1, \cdots, m} f_{\text{test}}^{k}(z). \tag{21}$$

1 回归: 预测值等于 RVFL 的输出函数 f——test（z）

$$\hat{y} = f_{test}(z). \tag{22}$$

RVFL+

按照 [**VAPNIK2009544**] 中的 SVM+ 公式, 我们可以将 RVFL+ 写成

$$\min_{\mathbf{w},\tilde{\mathbf{w}}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{\gamma}{2}\|\tilde{\mathbf{w}}\|_2^2 + C\sum_{i=1}^{N}\zeta_i(\tilde{\mathbf{w}}, \tilde{h}(\tilde{\mathbf{x}}_i))$$
$$\text{s.t. } \mathbf{h}(\tilde{x}_i)\mathbf{w} = \mathbf{y}_i - \zeta_i(\tilde{\mathbf{w}}; \tilde{h}(\tilde{\mathbf{x}}_i)), \forall \leq i \leq N. \tag{23}$$

其中 γ 是正则化系数. 与 $\mathbf{h}$（$\mathbf{x}_i$） 类似, $\tilde{\mathbf{h}}$（$\tilde{\mathbf{x}}_r i$） 也是与特权功能 $\tilde{\mathbf{x}}i$ 相对应的增强层输出向量, 可以用相同的方式计算. ζ_i（$\tilde{\mathbf{w}}, \tilde{h}$（$\tilde{\mathbf{x}}_i$）） 是特权功能空间中的更正函数（或可宽延函数）, 而 $\tilde{\mathbf{w}}$ 是更正函数的输出权重向量.

$$\zeta_i(\tilde{\mathbf{w}}, \tilde{h}(\tilde{\mathbf{x}}_i)) = \tilde{h}(\tilde{\mathbf{x}}_i)\mathbf{w}. \tag{24}$$

将(24)代入(23), 得到 RVFL+ 表达形式.

$$\min_{\mathbf{w},\tilde{\mathbf{w}}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{\gamma}{2}\|\tilde{\mathbf{w}}\|_2^2 + C\sum_{i=1}^{N}\tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}}$$
$$\text{s.t. } \mathbf{h}(\tilde{x}_i)\mathbf{w} = \mathbf{y}_i - \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}}, \forall \leq i \leq N. \quad (25)$$

从(25), 我们注意到 RVFL+ 最小化了 w 和 $\tilde{w}$ 上的目标函数. 因此, 训练阶段 RVFL+ 的分离超平面不仅取决于原始特征, 还取决于优先考虑的信息. 此外, 与 SVM+ 的原始模型相比 [**VAPNIK2009544**], RVFL+ 的修正函数是正的或负的. 换句话说, RVFL+ 不考虑约束组 ζ_i $(\mathbf{h}, b, \phi(\mathbf{x}i)) \geq 0$ $(i = 1, \cdots, N)$.

对于二值分类, RVFL+ 中的约束数目至少比 SVM+ 少 N 个, 这使得 RVFL+ 的优化约束要比 SVM+ 的小得多.

为了解决(25)中的优化问题, 构造如下的 Lagrangian 函数 $L(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})$

$$\min_{\mathbf{w},\tilde{\mathbf{w}},\boldsymbol{\lambda}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{\gamma}{2}\|\tilde{\mathbf{w}}\|_2^2 + C\sum_{i=1}^{N} \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}} - \sum_{i=1}^{N} \boldsymbol{\lambda}_i(\mathbf{h}(\tilde{\mathbf{x}}_i)\mathbf{w} - \mathbf{y}_i + \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}}), \tag{26}$$

其中 $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1, \cdots, \boldsymbol{\lambda}_N]^T$ 是拉格朗日乘子.

使用 KKT 条件来计算 Lagrangian 函数 $L(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})$ 关于变量 $\mathbf{w}, \tilde{\mathbf{w}}$ 和 $\boldsymbol{\lambda}$ 的鞍点.

$$\frac{\partial L(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \mathbf{H}^{\mathsf{T}}\boldsymbol{\lambda}, \tag{27}$$

$$\frac{\partial L(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})}{\partial \tilde{\mathbf{w}}} = 0 \Rightarrow \tilde{\mathbf{w}} = \frac{1}{\gamma}(\tilde{\mathbf{H}}^{\mathsf{T}}\boldsymbol{\lambda} - \tilde{\mathbf{H}}^{\mathsf{T}}\mathbf{C}\mathbf{1}), \tag{28}$$

$$\frac{\partial L(\mathbf{w}, \tilde{\mathbf{w}}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}_i} = 0 \Rightarrow \mathbf{h}(\mathbf{x}_i)\mathbf{w} - \mathbf{y}_i + \tilde{\mathbf{h}}(\mathbf{x}_i)\tilde{\mathbf{w}} = 0, 1 \leq i \leq N. \tag{29}$$

其中 $\mathbf{1} \in \mathbb{R}^{N \times m}$ 是单位矩阵. $\tilde{\mathbf{H}}$ 也是来自增强节点的连接输出矩阵, 它对应于 RVFL 的特性.

将(27)和(28)代入(29), 得到

$$\mathbf{H}\mathbf{H}^{\mathsf{T}}\boldsymbol{\lambda} + \frac{1}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^{\mathsf{T}}(\boldsymbol{\lambda} - \mathbf{C}\mathbf{1}) = \mathbf{Y}. \tag{30}$$

可以进一步将(30)重写为

$$\left(\mathbf{H}\mathbf{H}^{\mathsf{T}} + \frac{1}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^{\mathsf{T}}\right)\boldsymbol{\lambda} = \mathbf{Y} - \frac{\mathbf{C}\mathbf{1}}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^{\mathsf{T}}. \tag{31}$$

再由(27)和(31), 得到 RVFL+As 的最后闭式解

$$\mathbf{w} = \mathbf{H}^{\mathsf{T}}\left(\mathbf{H}\mathbf{H}^{\mathsf{T}} + \frac{1}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^{\mathsf{T}}\right)^{-1}\left(\mathbf{Y} - \frac{\mathbf{C}\mathbf{1}}{\gamma}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^{\mathsf{T}}\right). \tag{32}$$

根据岭回归方法 [**Bishop2012-6469**], 为了避免奇异性并保证 RVFL+ 的稳定性, 我们还附加了一个 $\frac{\mathbf{I}}{C}$ 项. 因此, 我们可以得到 RVFL+As 的最终闭式解

$$\mathbf{w} = \mathbf{H}^{\mathsf{T}} \left(\mathbf{H}\mathbf{H}^{\mathsf{T}} + \frac{1}{\gamma} \tilde{\mathbf{H}} \tilde{\mathbf{H}}^{\mathsf{T}} + \frac{\mathbf{I}}{C} \right)^{-1} \left(\mathbf{Y} - \frac{C\mathbf{1}}{\gamma} \tilde{\mathbf{H}} \tilde{\mathbf{H}}^{\mathsf{T}} \right). \tag{33}$$

RVFL+ 输出函数定义为

$$f(x) = \mathbf{h}(\mathbf{x})\mathbf{w} = \mathbf{h}(\mathbf{x})\mathbf{H}^{\mathsf{T}} \left(\mathbf{H}\mathbf{H}^{\mathsf{T}} + \frac{1}{\gamma} \tilde{\mathbf{H}} \tilde{\mathbf{H}}^{\mathsf{T}} + \frac{\mathbf{I}}{C} \right)^{-1} \left(\mathbf{Y} - \frac{C\mathbf{1}}{\gamma} \tilde{\mathbf{H}} \tilde{\mathbf{H}}^{\mathsf{T}} \right). \tag{34}$$

此外, 在测试阶段, 当使用测试数据 $\mathbf{z}$ 而不是训练数据 $\mathbf{x}$ 时, 我们可以直接获得输出函数 $f_{test}(\mathbf{z}) = \mathbf{h}(\mathbf{z}) - \mathbf{w}$.

张量结构的 RVFL

本节将随机向量函数链接网络 (RVFL) 推广到了张量结构, 增强节点使用了区间二型模糊集来扩展非线性激活函数, 主要目的是用二型模糊集来捕捉输入输出关系中蕴含的高阶信息.

两种随机向量函数链接网络

- 区间二型随机向量函数链接网络和随机向量函数链接网络类似, 不同之处在于其输出增强节点的输出是不确定加权方法的解模糊结果.
- 张量的二型随机向量函数链接网络也使用了不确定加权方法, 同时还使用了下隶属函数值和上隶属函数值, 由此形成了一个 4 阶张量.

- 基于这个结构, 基于 Einstein 积的偶数阶张量的 Moore-Penrose 逆被用来求解张量方程.
- 此外, 权重向量由一个平衡因子综合得到, 它是 RVFL 网络结果和张量方程结果的加权平均.
- 最后, 分别在三个非线性测试函数、非线性系统识别问题和四个回归问题上验证了给出的算法.

高斯区间二型模糊集有两种定义形式, 不确定均值或者是不确定方差. 本节使用的区间二型模糊集上隶属和下隶属的均值相同.
图 20 给出了均值相同、方差不同的区间二型模糊集.

隶属函数的定义

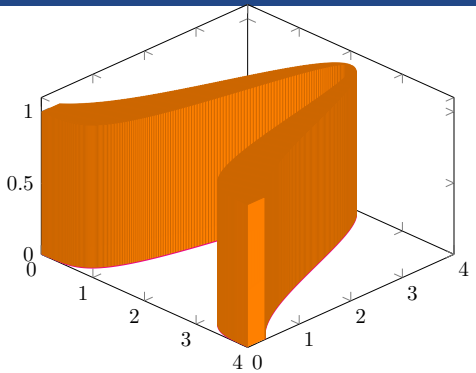


图 20: 区间二型模糊集

$$\bar{\mu}_i(x_i) = \exp\left(-\frac{(x_i - m_i)^2}{\bar{\sigma}_i^2}\right), \tag{35}$$

$$\underline{\mu}_i(x_i) = \exp\left(-\frac{(x_i - m_i)^2}{\underline{\sigma}_i^2}\right), \tag{36}$$

其中 m_i, $\bar{\sigma}_i$ 和 $\underline{\sigma}_i, (1 \leq i \leq L)$ 分别是二型模糊集的下隶属函数和上隶属函数的均值和方差. 下隶属函数和上隶属函数之间的隶属度是 1 的二型模糊集就是区间二型模糊集.

图 21 给出了基于张量积的区间二型随机向量函数链接网络 (TT2-RVFL), 其中 $\tilde{g}_i(\cdot)\,(i = 1, 2, \cdots, L)$ 是区间二型模糊激活函数. 不同于 RVFL, TT2-RVFL 使用区间二型模糊集, 扩展了 RVFL 的增强节点部分, 因此, 区间二型模糊集的输出可以看成是原来两个激活函数的输出结果. RVFL 网络中的乘法需要修正, 对于高维结构的

TT2-RVFL 的增强型节点使用了张量的 Einstein 积运算

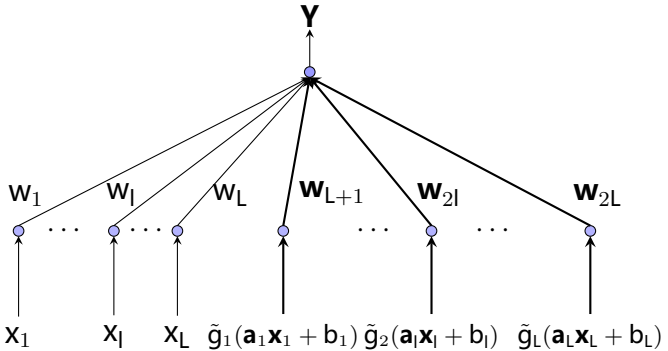


图 21: TT2-RVFL 的结构

TT2-RVFL 的输出

输出模型如图 21:

$$\mathbf{Y} = \mathcal{A} *_{\mathrm{N}} \mathcal{X} + \mathbf{H}\boldsymbol{\Omega}, \tag{37}$$

其中 $\mathcal{A} *_{\mathrm{N}} \mathcal{X}$ 为增强节点的张量乘积, $\mathbf{H}\boldsymbol{\Omega}$ 为线性部分的输出, $\mathcal{X}$ 为增强节点部分的权重矩阵；类似于 RVFL, $\mathbf{H}$ 是输入数据矩阵, 单列输出的数据需要复制一次, 以便与张量和矩阵的运算结果相容, Ω 是线性部分的权矩阵, 且 $\mathcal{A}$ 是一个张量, 它是由叠加区间二型模糊集的映射结果得到的张量. 对于 TT2-RVFL, $\mathbf{Y}$, $\mathcal{X}$ 和 Ω 需要特别考虑, 以便和后件形成的张量方程相容. 下一节将介绍由三个变量组成的张量结构.

TT2-RVFL

激活函数

RVFL 常使用 Radbas ($y = \exp(-s^2)$), Sine ($y = \sin(s)$), Sigmoid ($y = \frac{1}{1+e^{-s}}$) 和 Tribas ($y = \max(1-|s|, 0)$) 这几类激活函数, 其中 s 和 y 分别表示输入和输出变量.

对于 TT2-RVFL, 增强节点使用了区间二型模糊集. RVFL 的 Radbas 激活函数被推广成区间二型模糊集, 扩展后的 RVFL 使用了区间二型模糊集, 扩展后的激活函数简记为 IT2Radbas, 它是 (35) 和 (36) 的变种, 记为 $\tilde{g}_i(\cdot)$.

IT2Radbas

$$\bar{g}_i(x_i) = \exp(-k_1 s^2), \tag{38}$$

$$\underline{g}_i(x_i) = \exp(-k_2 s^2), \tag{39}$$

其中 $s = x_i - m_i, k_1 = \frac{1}{\bar{\sigma}_i^2}, k_2 = \frac{1}{\underline{\sigma}_i^2}\ (i = 1, 2, \cdots, L)$.

对于测试集 $\{D_t\}_{t=1}^{N}$, 其中 $D_t = (\mathbf{x}_t, y_t)$, $\mathbf{x}_t = (x_{t1}, x_{t2}, \cdots, x_{tN}) \in \mathbb{R}^N$, $y_t \in \mathbb{R}$ 且 $\mathbf{y} = [y_1, \cdots, y_N]^T$. 下隶属函数矩阵 $\underline{\Phi} \in \mathbb{R}^{N\times 2\times L\times 1}$ 可用下式构建

$$\underline{\Phi}_{:,:,1,1} = \begin{bmatrix} \underline{g}_1(\mathbf{a}_{11}\mathbf{x}_1 + b_{11}) & \underline{g}_1(\mathbf{a}_{12}\mathbf{x}_1 + b_{12}) \\ \vdots & \vdots \\ \underline{g}_1(\mathbf{a}_{11}\mathbf{x}_N + b_{11}) & \underline{g}_1(\mathbf{a}_{12}\mathbf{x}_N + b_{12}) \end{bmatrix},$$

$$\vdots$$

$$\underline{\Phi}_{:,:,L,1} = \begin{bmatrix} \underline{g}_L(\mathbf{a}_{L1}\mathbf{x}_1 + b_{L1}) & \underline{g}_L(\mathbf{a}_{L2}\mathbf{x}_1 + b_{L2}) \\ \vdots & \vdots \\ \underline{g}_L(\mathbf{a}_{L1}\mathbf{x}_N + b_{L1}) & \underline{g}_L(\mathbf{a}_{L2}\mathbf{x}_N + b_{L2}) \end{bmatrix},$$

其中 b_{il} 和 $\mathbf{a}_{il} = [w_{i1}, w_{i2}, \cdots, w_{iK}]$ $(i = 1, 2, \cdots, L; l = 1, 2)$ 是随机生成的偏置和输入权重.

下隶属函数矩阵使用下隶属函数来逼近输入 $\mathbf{x}_t$ 和期望输出 y_t 的关系. 上隶属函数矩阵 $\bar{\Phi} \in \mathbb{R}^{N\times 2\times L\times 1}$ 可以用类似的方式建立, 具有如下形式

$$\bar{\Phi}_{:,:,1,2} = \begin{bmatrix} \bar{g}_1(\mathbf{a}_{11}\mathbf{x}_1 + b_{11}) & \bar{g}_1(\mathbf{a}_{12}\mathbf{x}_1 + b_{12}) \\ \vdots & \vdots \\ \bar{g}_1(\mathbf{a}_{11}\mathbf{x}_N + b_{11}) & \bar{g}_1(\mathbf{a}_{12}\mathbf{x}_N + b_{12}) \end{bmatrix},$$

$$\vdots$$

$$\bar{\Phi}_{:,:,L,2} = \begin{bmatrix} \bar{g}_L(\mathbf{a}_{L1}\mathbf{x}_1 + b_{L1}) & \bar{g}_L(\mathbf{a}_{L2}\mathbf{x}_1 + b_{L2}) \\ \vdots & \vdots \\ \bar{g}_L(\mathbf{a}_{L1}\mathbf{x}_N + b_{L1}) & \bar{g}_L(\mathbf{a}_{L2}\mathbf{x}_N + b_{L2}) \end{bmatrix},$$

其中 $\mathbf{a}_{ij}$ $(i = 1, 2\cdots, L; j = 1, 2)$ 是构建张量所用的随机生成的权重向量.

不确定加权方法 [**RunklerCoupland2018-6356**] 被用于区间二型模糊激活函数的解模糊化计算, 它代表了隶属函数隶属度对于整体解模糊化结果的影响程度, 映射结果由下式计算得到

$$\mathsf{u}_{\mathrm{UW}}(\mathsf{x}) = \frac{1}{2}(\underline{\mathsf{u}}(\mathsf{x}) + \bar{\mathsf{u}}(\mathsf{x})) \cdot (1 + \underline{\mathsf{u}}(\mathsf{x}) - \bar{\mathsf{u}}(\mathsf{x}))^{\gamma}, \tag{40}$$

其中 $\gamma > 0$ 用于调整由 $(1+\underline{\mathsf{u}}(\mathsf{x})-\bar{\mathsf{u}}(\mathsf{x}))^{\gamma}$ 给出解模糊化结果. 不确定加权方法推广了简单的平均下隶属度和上隶属度方法, 清晰输出结果由下式给出

$$(1 + \underline{\mathsf{u}}(\mathsf{x}) - \bar{\mathsf{u}}(\mathsf{x}))^{\gamma} = \begin{cases} 0, & \underline{\mathsf{u}}(\mathsf{x}) = 0, \bar{\mathsf{u}}(\mathsf{x}) = 1, \\ 1, & \underline{\mathsf{u}}(\mathsf{x}) = \bar{\mathsf{u}}(\mathsf{x}). \end{cases} \tag{41}$$

方程 (41) 显示出: 当 $\gamma = 1$ 时, 解模糊结构呈线性, 对于所有的 $\gamma < 1$ 线性增加, 且对于所有 $\gamma > 1$, 线性将会减少.

当 (41) 被用于解模糊化, 只用区间二型模糊集拓展随机向量函数链接网络, 也即, 只是扩展了原网络的增强节点为区间二型模糊集, 这种类型的网络记为 IT2-RVFL.

最后得到的 4 阶张量 $\in \mathbb{R}^{N\times 2\times L\times 2}$ 是由前面提到的张量 $\underline{\Phi}_{:,:,:,1}$ 和 $\bar{\Phi}_{:,:,:,2}$ 构建出来的. 接下来, 张量 用 $\mathcal{A}$ 来记, 这是由于经典的张量方程中经常使用 $\mathcal{A}$ 表示张量方程的系数.

增强节点的权重矩阵

基于张量的乘积运算, 得到的增强节点的权重矩阵维数是 $\mathrm{L} \times 2$, 将 TT2-RVFL 的增强节点部分的权重矩阵记为

$$\mathcal{X} = \begin{bmatrix} \mathbf{w}_{\mathrm{L}+1\,1} & \mathbf{w}_{\mathrm{L}+1\,2} \\ \mathbf{w}_{\mathrm{L}+2\,1} & \mathbf{w}_{\mathrm{L}+2\,2} \\ \vdots & \vdots \\ \mathbf{w}_{\mathrm{L}+\mathrm{L}\,1} & \mathbf{w}_{\mathrm{L}+\mathrm{L}\,2} \end{bmatrix}. \tag{42}$$

TT2-RVFL 的输出

$$\mathbf{Y} = \alpha \mathcal{A} *_{\mathsf{N}} \mathcal{X} + (1-\alpha)\mathbf{H}\boldsymbol{\Omega}, 0 \leq \alpha \leq 1, \tag{43}$$

其中 α 是 TT2-RVFL 的平衡因子, $\mathcal{A}$ 是区间二型模糊激活函数的映射结果, $\mathcal{X}$ 是权重矩阵, $\mathbf{Y} = [\mathbf{y}\,\mathbf{y}]$, 从输入到增强节点的权重向量 $\mathbf{a}_{\mathsf{i}}\,(\mathsf{i} = 1, 2, \cdots, \mathsf{L})$ 是随机生成的, 使得 $\underline{\Phi}$, $\bar{\Phi}$ 不饱和; $\boldsymbol{\Omega} = [\boldsymbol{\omega}\,\boldsymbol{\omega}]$ 是待求的输入权重矩阵, $\mathbf{H}$ 是由输入样本构建的输入矩阵.

当下三角隶属函数和上三角隶属函数一样时, TT2-RVFL 退化为 RVFL, 也即, 激活函数是一型模糊集. 当不适用直接连接方式时, TT2-RVFL 将会退化为张量型极限学习机 [**HuangZhao2018NCAA-5838**]. 当同时使用 RVFL 和张量结构时, TT2-RVFL 是这两种结构的混合结果.

和 RVFL 相比, 基于张量的二型 RVFL 模型 (43) 做了三方面的扩展: 1) 增强节点, 使用区间二型模糊激活函数来得到非线性映射结果, 增强节点部分由张量和张量乘法表示, 且使用了张量方程的求解算法. 2) 为了和增强节点的运算相容, TT2-RVFL 的线性部分需要复制权重向量一次, 权重矩阵 $\boldsymbol{\Omega}$ 等于 $[\boldsymbol{\omega}\,\boldsymbol{\omega}]$, 其中 $\boldsymbol{\omega}$ 为权重列向量. 3) 对于输出 $\mathbf{Y}$, 和以前的输出向量 $\mathbf{y}$ 不同, TT2-RVFL 需要重复该向量一次, 以便和张量运算相容.

TT2-RVFL 中的张量逆

张量方程 $\mathcal{A} *_{\mathsf{N}} \mathcal{X} = \mathcal{Y}$ 的张量运算被用于求解 TT2-RVFL 网络的权重向量, 该运算非常适合计算 TT2-RVFL 增强节点部分的权重向量. 求解方程使用的偶数阶张量的 Moore-Penrose (M-P) 逆在算法 4 中给出, 在得到 M-P 逆的基础上, 可以很容易算出权重矩阵.

方程 $\mathcal{A} *_{\mathsf{N}} \mathcal{X} = \mathcal{Y}$ 的解也是如下张量方程的解

$$\mathcal{A}^{\mathsf{T}} *_{\mathsf{N}} \mathcal{A} *_{\mathsf{N}} \mathcal{X} = \mathcal{A}^{\mathsf{T}} *_{\mathsf{N}} \mathcal{Y}, \tag{44}$$

其中张量 $\mathcal{A} \in \mathbb{R}^{\mathsf{I}_1 \times \cdots \times \mathsf{I}_{\mathsf{N}} \times \mathsf{J}_1 \times \cdots \times \mathsf{J}_{\mathsf{N}}}$, $\mathcal{X} \in \mathbb{R}^{\mathsf{J}_1 \times \cdots \times \mathsf{J}_{\mathsf{N}} \times \mathsf{K}_1 \times \cdots \times \mathsf{K}_{\mathsf{M}}}$ 且 $\mathcal{Y} \in \mathbb{R}^{\mathsf{I}_1 \times \cdots \times \mathsf{I}_{\mathsf{N}} \times \mathsf{K}_1 \times \cdots \times \mathsf{K}_{\mathsf{M}}}$, 不论方程是否相容, 都可以应用表 4 中的算法来求解方程 $\mathcal{A} *_{\mathsf{N}} \mathcal{X} = \mathcal{Y}$ [**HuangZhao2019-9577**].

表 4: 算法 1: 偶数阶张量 $\mathcal{A}$ 的 M-P 逆 [**HuangZhao2019-9577**]

输入:	张量 $\mathcal{A}$, $I_{1\cdots N}$ 和指标 $J_{1\cdots N}$ N 为偶数, $I=\prod_{i=1}^{N} I_i$, $J=\prod_{i=1}^{N} J_i$ 变换张量 $\mathcal{A}\in\mathbb{R}^{I_{1\cdots N}\times J_{1\cdots N}}$ 为矩阵 $A\in\mathbb{R}^{I\times J}$ 对矩阵 A 做奇异值分解, A 可以分解为 $A=USV^*$, 其中 V^* 是矩阵 V 的共轭转置 分别将矩阵 U, S 和 V^* 转换为相应阶次的张量 $\mathcal{U},\mathcal{S}$ 和 $\mathcal{V}^*$
输出:	通过 $\mathcal{A}^+=\mathcal{V}*_N\mathcal{S}^+*_N\mathcal{U}^*$ 计算张量 $\mathcal{A}$ 的 M-P 逆

求解 TT2-RVFL 的张量逆算法

为了得到 TT2-RVFL 的解, 模型 (43) 的最小范目标函数定义为

$$\min_{\mathbf{\Omega},\mathcal{X}} \mathcal{E}_1(\mathcal{X}) + \mathcal{E}_2(\Omega), \tag{45}$$

其中 $\mathcal{E}_1(\mathcal{X}) = \|\mathbf{Y} - \mathcal{A}\mathcal{X}\|_{\mathsf{F}}^2 + \|\mathcal{X}\|_{\mathsf{F}}^2$, $\mathcal{E}_2(\Omega) = \|\mathbf{y} - \mathbf{H}\mathbf{\Omega}\|_2^2 + \|\mathbf{\Omega}\|_2^2$.

模型 (43) 的极小化问题可以划分成两个子问题来求解, 由下面的定理给出：极小化模型的解 1 对于极小化模型 (43), 令 Ω^* 是 $\mathbf{y} = \mathbf{H\Omega}$ 的解集, $\mathcal{X}^*$ 是 $\mathcal{Y} = \mathcal{A} *_{\mathrm{N}} \mathcal{X}$ 的解. 对于模型 (45) 的两个子问题 $\mathcal{E}_1(\mathcal{X})$ 和 $\mathcal{E}_2(\mathcal{X})$, $\mathcal{Y}$ 退化为一个矩阵, 即 $\mathcal{Y} = [\mathbf{y}\,\mathbf{y}]$, 且 $\mathbf{y}$ 是训练结果列向量. 式 (43) 的充分条件就是寻找如下子问题的解

$$\min_{\mathcal{X}} \mathcal{E}_1(\mathcal{X}) \text{ 和 } \min_{\mathbf{\Omega}} \mathcal{E}_2(\Omega), \tag{46}$$

则方程的解 $\begin{bmatrix}\mathcal{X} & \mathbf{\Omega}\end{bmatrix}^{\mathsf{T}}$ 满足下式

$$\mathbf{Y} = \mathcal{A} *_{\mathrm{N}} \mathcal{X} + \mathbf{H\Omega} = \begin{bmatrix}\mathcal{A} & \mathbf{H}\end{bmatrix} * \begin{bmatrix}\mathcal{X} \\ \mathbf{\Omega}\end{bmatrix}, \begin{bmatrix}\mathcal{X} \\ \mathbf{\Omega}\end{bmatrix} = \begin{bmatrix}\alpha\mathcal{X}^* \\ (1-\alpha)\mathbf{\Omega}^*\end{bmatrix}, \tag{47}$$

其中 α 是 TT2-RVFL ($0 \le \alpha \le 1$) 的平衡因子, $*_{\mathrm{N}}$ 是张量 $\mathcal{A}$ 和 $\mathcal{X}$ 的 Einstein 乘积.

$\mathbf{H\Omega} = \mathbf{Y}$ 的解是 Ω^*, $\mathbf{H\Omega}^* = \mathbf{Y}$ 成立.

$\mathcal{X}^*$ 是 $\mathcal{A} *_{\mathsf{N}} \mathcal{X} = \mathcal{Y}$ 的解, 则 $\mathcal{A} *_{\mathsf{N}} \mathcal{X}^* = \mathcal{Y}$.

给定平衡因子 α, $\mathcal{A} *_{\mathsf{N}} \alpha\mathcal{X}^* + \mathbf{H}(1-\alpha)\mathbf{\Omega}^* = \alpha\mathbf{Y} + (1-\alpha)\mathbf{Y} = \mathbf{Y}$ 成立. 对于 $\mathbf{Y} \equiv \mathcal{Y}$ 的情况, 张量 $\mathcal{Y}$ 退化为矩阵 $\mathbf{Y}$, 且 $\mathbf{Y} \in \mathbb{R}^{\mathsf{L}\times 2}$.

对于 TT2-RVFL, 权重向量是 RVFL 的输出结果和张量方程解结果的综合, RVFL 的权重向量记为 $\boldsymbol{\omega} = (\mathbf{H}^{\mathsf{T}}\mathbf{H} + \frac{\mathbf{I}}{\mathsf{C}})^{-1}\mathbf{H}^{\mathsf{T}}\mathbf{Y}$, 其中 $\mathbf{I}$ 是单位矩阵, 且 C 是平衡参数.

仿真实验

提出的算法使用了 Frobenius 范数来度量训练误差和测试误差, 所有结果都由 1000 次运行结果求得. 为了说明 IT2-RVFl 和 TT2-RVFL 的学习能力, 隐藏节点数选为 $L = [30, 35, 40]$. IT2-RVFL 和 TT2-RVFL 使用了均值一样, 方差不确定这一形式, 下隶属函数从区间 $[0.5, 0.9]$ 随机选取, 上隶属函数通过给下隶属函数值加 0.1 得到; 则 k_1, k_2 可由关系式 $k_1 = \frac{1}{\bar{\sigma}_i^2}, k_2 = \frac{1}{\underline{\sigma}_i^2}\ (i = 1, 2, \cdots, L)$ 求得.

Example

单输入 Sinc 函数的逼近问题如下

$$y = \begin{cases} 1, & x = 0 \\ \frac{\sin x}{x}, & x \neq 0 \end{cases} \quad x \in [-10, 10]. \tag{48}$$

训练误差和测试误差

表 5 给出了例 5.5 在不同 L 下的训练误差和测试误差. 对于 IT2-RVFL, 当 $L = 30, 35$, 它的训练误差要小于 TT2-RVFL 的训练误差. 然而, TT2-RVFL 的测试误差要小于 IT2-RVFL 的测试误差, 这意味着 TT2-RVFL 中的平衡因子对网络的学习能力有一定的影响.

表 5: 例 5.5 中不同 L 下的训练和测试误差

L	算法	训练误差		预测误差	
		均值	方差	均值	方差
30	IT2-RVFL	2.53e-02	4.51e-05	4.82e-02	7.22e-04
	TT2-RVFL	**2.36e-04**	**1.71e-05**	**2.32e-04**	**1.70e-05**
35	IT2-RVFL	2.34e-02	9.15e-05	4.66e-02	3.17e-04
	TT2-RVFL	**1.57e-02**	**7.44e-05**	**1.54e-02**	**7.61e-05**
40	IT2-RVFL	**2.18e-02**	**1.38e-04**	4.57e-02	7.92e-04
	TT2-RVFL	3.86e-02	4.25e-04	**3.76e-02**	**4.25e-04**

例 5.6: 含噪的非线性函数具有如下形式

$$\mathrm{y} = \frac{0.9\mathrm{x}}{1.2\mathrm{x}^3 + \mathrm{x} + 0.3} + \eta, \mathrm{x} \in [0, 1], \tag{49}$$

其中 η 是均匀分布的白噪声, η 的幅值设置为 0.2.

例 5.6 用来测试给出算法的野值拒绝能力. 对于 IT2RVFL 和 TT2-RVFL, 平衡参数设置为 $\mathrm{C} = 2^{10}$, $\gamma = 1$. 表 6 给出了例 5.6 在不同 L 值下的训练误差和测试误差. 结果表明, 当 $\mathrm{L} = 30, 35, 40$ 时, TT2-RVFL 性能要好于 IT2-RVFL, 其平均训练误差和测试误差要小于 IT2-RVFL 的结果.

表 6: 例 5.6 中不同 L 下的训练误差和预测误差

L	算法	训练误差		预测误差	
		均值	方差	均值	方差
30	IT2-RVFL	2.85e-01	**7.30e-06**	4.13e-01	5.58e-03
	TT2-RVFL	**2.07e-01**	8.43e-04	**3.07e-01**	**2.19e-03**
35	IT2-RVFL	2.85e-01	**2.94e-06**	4.15e-01	2.45e-03
	TT2-RVFL	**2.07e-01**	1.75e-03	**3.06e-01**	**1.96e-03**
40	IT2-RVFL	2.85e-01	**5.14e-06**	4.11e-01	5.10e-03
	TT2-RVFL	**2.06e-01**	4.54e-04	**3.04e-01**	**1.13e-03**

例 5.7——双输入单输出 Sinc 函数的逼近问题.

$$z = \left| \frac{\sin x \cdot \sin y}{xy} \right|, (x, y) \in [-\pi, \pi]^2. \tag{50}$$

采用等距采样方式, 41×41 训练数据用于训练, 30×30 等距采样结果用来检验模型的泛化能力. 对于 IT2-RVFL 和 TT2-RVFL, 平衡因子设置为 $C = 2^{10}$, $\gamma = 1$. 表 7 给出了例 5.7 在不同 L 下的训练误差和测试误差.

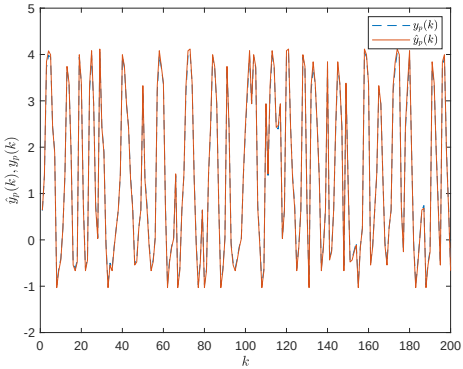


图 22: TT2-RVFL 对非线性系统识别问题的预测结果

TT2-RVFL 的结果要好于 IT2-RVFL 的结果, 训练误差和测试误差都要小于 IT2-RVFL 的结果.

表 7: 例 5.7 中不同 L 下的训练误差和预测误差

L	算法	训练误差		预测误差	
		均值	方差	均值	方差
30	IT2-RVFL	4.65e-02	**3.09e-06**	7.06e-02	**1.43e-09**
	TT2-RVFL	**4.36e-02**	1.30e-05	**6.77e-02**	6.05e-06
35	IT2-RVFL	4.63e-02	**8.69e-06**	7.06e-02	**2.40e-08**
	TT2-RVFL	**4.07e-02**	3.38e-05	**6.73e-02**	9.21e-06
40	IT2-RVFL	**4.63e-02**	**2.95e-06**	7.06e-02	**1.56e-09**
	TT2-RVFL	**3.98e-02**	4.72e-05	**6.75e-02**	1.09e-05

非线性系统

$$y_p(k) = \frac{y_p(k-1)y_p(k-2)(y_p(k-1)+2.5)}{1+(y_p(k-1))^2+(y_p(k-2))^2} + u(k-1). \tag{51}$$

系统的平衡态为 (0,0), 输入 $u(k) \in \{-2, 2\}$, 操作区间规定为 $[-2, 2]$. 均匀分布的随机变量从 $\{-2, 2\}$ 中产生, 测试使用的输入 $u(k) = \sin(2\pi k/25)$. 通过选取 $[y_p(k-1), y_p(k-2), u(k-1)]$ 作为输入变量, $y_p(k)$ 作为输出变量. 系统可以写为

$$\hat{y}_p(k) = \hat{f}(y_p(k-1), y_p(k-2), u(k-1)). \tag{52}$$

随机选用了 800 组数据, 600 组用于训练, 200 组用于测试.

对于 IT2RVFL 和 TT2-RVFL, 平衡参数设置为 C $= 2^{10}$. 对于仿真, TT2-RVFL 的预测结果见图 22.

训练误差均值的 Frobenius 范数是 1.87e-01, 测试误差均值的 Frobenius 范数是 3.94e-01.

结果显示, TT2-RVFL 能够得到满意的精度.

四个回归数据集 (Auto-Mpg

- Auto-MPG 数据集有 392 个样本, 每一个样本有 8 个属性. Bank 数据集有 8192 个样本, 每一个样本有 8 个属性.
- Diabetes 数据集有 768 个样本, 每一个样本有 4 个属性. Triazines 数据集有 186 个样本, 每一个样本有 5 个属性.
- Auto-MPG 和 Bank, 使用了 mapminmax 方法来计算映射到区间 $[-1, 1]$ 的数据.
- 对于 Diabetes 和 Triazines, 则直接使用原来的数据样本. 对于 IT2-RVFL 和 TT2-RVFL, 平衡参数设置为 C $= 2^{10}$, $\gamma = 1$.

回归问题

表 8: 数据集 Auto-Mpg, Bank, Diabetes 和 Triazines 上的训练和测试误差

数据集	算法	训练误差		预测误差	
		均值	方差	均值	方差
Auto-Mpg	ELM	4.3511e-01	2.4125e+0	4.3423e-01	2.5502e+0
	RELM	1.3514e+0	1.6125e+0	1.3132e+0	1.6101e+0
	RVFL	9.9152e-01	6.3012e-03	9.2356e-01	5.1511e-03
	IT2-RVFL	**2.2161e-02**	**1.5915e-04**	4.5358e-02	**1.6692e-04**
	TT2-RVFL	4.4055e-02	8.1940e-04	**4.3847e-02**	8.1024e-04
Bank	ELM	3.2400e-02	5.8710e-02	3.2103e-02	5.8425e-02
	RELM	3.3214e-02	5.8321e-02	**3.0936e-02**	5.8631e-02
	RVFL	1.1784e-01	6.3033e-03	5.0163e-02	1.3696e-04
	IT2-RVFL	**2.9035e-02**	**3.4437e-05**	7.0701e-02	9.9726e-05
	TT2-RVFL	4.3582e-02	1.1435e-04	5.1382e-02	**8.3401e-05**
Diabetes	ELM	1.5199e-01	1.5535e-01	1.5724e-01	1.3706e-01
	RELM	1.5046e-01	1.3448e-01	1.5741e-01	1.4002e-01
	RVFL	3.3931e-01	6.2734e-02	**8.6827e-02**	1.0573e-02
	IT2-RVFL	1.4838e-01	2.6181e-05	1.5938e-01	4.3915e-04
	TT2-RVFL	**1.4548e-01**	**2.1532e-06**	1.4522e-01	**8.8627e-05**
Triazines	ELM	9.8322e-02	2.4095e-01	1.0947e-01	1.6613e-01
	RELM	9.9402e-02	2.1026e-01	1.0739e-01	1.6718e-01
	RVFL	3.0201e-01	2.8114e-01	**2.2935e-02**	1.3741e-02
	IT2-RVFL	**8.2873e-02**	**6.5574e-05**	4.0105e-01	3.9270e-03
	TT2-RVFL	1.0360e-01	3.1127e-04	2.6182e-01	**1.0300e-03**

RVFL 的代码可以从 Matlab 的 File Exchange 下载, ELM 和 RELM 的代码可以从主页下载. 表 8 给出了 ELM、RELM、RVFL、IT2-RVFL 和 TT2-RVFL 算法的训练误差和测试误差.

- 对于 Auto-MPG, 扩展后的 RVFL 的训练误差和测试误差都得到了很好的结果.
- IT2-RVFL 在 Bank 上的训练误差最小, RELM 在 Bank 上的测试误差最小, 这说明正则化项在 ELM 的训练过程中是发挥作用的.
- TT2-RVFL 在 Diabetes 上的训练误差最小, 它的测试误差要大于 RVFL, 均值的误差大约是 0.0584.
- IT2-RVFL 在 Triazines 的训练误差最小, 而 RVFL 的测试误差则总体较小.

对于四个数据集的回归结果, 可以得出 IT2-RVFL 和 TT2-RVFL 是可以作为回归问题的求解器使用的.

如表 6 的结果所示, TT2-RVFL 更适合于数据含噪的情形.

Example

利用表 8的四个数据集和神经网络工具线或者神经网络 APP, 编写一个表 8的实现代码, 并对结果进行分析比较.

进化计算

进化计算

进化计算 (Evolutionary Computation,EC) 是在达尔文 (Darwin) 的进化论和孟德尔 (Mendel) 的遗传变异理论的基础上产生的一种在基因和种群层次上模拟自然界生物进化过程与机制的问题求解技术.

进化计算的四大分支

- 遗传算法 (Genetic Algorithm, GA)
- 进化策略 (Evolutionary Strategy, ES)
- 进化规划 (Evolutionary Programming, EP)
- 遗传规划 (Genetic Programming,GP)

进化计算

是一种模拟自然界生物进化过程与机制进行问题求解的自组织、自适应的随机搜索技术. 它以达尔文进化论的“物竞天择、适者生存”作为算法的进化规则, 并结合孟德尔的遗传变异理论, 将生物进化过程中的四种方式:

- 繁殖 (Reproduction),
- 变异 (Mutation),
- 竞争 (Competition),
- 选择 (Selection) 引入到了算法中.

(2) 进化计算的生物学基础

自然界生物进化过程是进化计算的生物学基础, 它主要包括遗传 (Heredity)、变异 (Mutation) 和进化 (Evolution) 理论.

① 遗传理论

遗传是指父代 (或亲代) 利用遗传基因将自身的基因信息传递给下一代 (或子代), 使子代能够继承其父代的特征或性状的这种生命现象. 正是由于遗传的作用, 自然界才能有稳定的物种.

染色体

在自然界, 构成生物基本结构与功能的单位是细胞 (Cell). 细胞中含有一种包含着所有遗传信息的复杂而又微小的丝状化合物, 人们称其为染色体 (Chromosome).

基因

在染色体中, 遗传信息由基因 (Gene) 所组成, 基因决定着生物的性状, 是遗传的基本单位.

染色体的形状

是一种双螺旋结构, 构成染色体的主要物质叫做 □ 脱氧核糖核酸 (DNA), 每个基因都在 DNA 长链中占有一定的位置. 一个细胞中的所有染色体所携带的遗传信息的全体称为一个基因组 (Genome). 细胞在分裂过程中, 其遗传物质 DNA 通过复制转移到新生细胞中, 从而实现了生物的遗传功能.

② 变异理论

变异是指子代和父代之间, 以及子代的各个不同个体之间产生差异的现象.
变异是生物进化过程中发生的一种随机现象, 是一种不可逆过程, 在生物多样性方面具有不可替代的作用.

引起变异的主要原因有以下两种

- 杂交, 是指有性生殖生物在繁殖下一代时两个同源染色体之间的交配重组, 即两个染色体在某一相同处的 DNA 被切断后再进行交配重组, 形成两个新的染色体.
- 复制差错, 是指在细胞复制过程中因 DNA 上某些基因结构的随机改变而产生出新的染色体.

③ 进化论

进化是指在生物延续生存过程中, 逐渐适应其生存环境, 使得其品质不断得到改良的这种生命现象. 遗传和变异是生物进化的两种基本现象, 优胜劣汰、适者生存是生物进化的基本规律.

达尔文的自然选择学说认为: 在生物进化中, 一种基因有可能发生变异而产生出另一种新的生物基因. 这种新基因将依据其与生存环境的适应性而决定其增殖能力.

一般情况下, 适应性强的基因会不断增多, 而适应性差的基因则会逐渐减少.

通过这种自然选择, 物种将逐渐向适应于生存环境的方向进化, 甚至会演变成为另一个新的物种, 而那些不适应环境的物种将会逐渐被淘汰.

进化计算的产生与发展

自 20 世纪 50 年代以来, 其发展过程大致可分为三个阶段.

① 萌芽阶段

这一阶段是从 20 世纪 50 年代后期到 70 年代中期. 20 世纪 50 年代后期, 一些生物学家在研究如何用计算机模拟生物遗传系统中, 产生了遗传算法的基本思想, 并于 1962 年由美国密执安 (Michigan) 大学霍兰德 (Holland) 提出. 1965 年德国数学家雷切伯格 (Rechenberg) 等人提出了一种只有单个个体参与进化, 并且仅有变异这一种进化操作的进化策略.

① 萌芽阶段

同年, 美国学者弗格尔 (Fogel) 提出了一种具有多个个体和仅有变异一种进化操作的进化规划.
1969 年美国密执安 (Michigan) 大学的霍兰德 (Holland) 提出了系统本身和外部环境相互协调的遗传算法. 至此, 进化计算的三大分支基本形成.

② 成长阶段

这一阶段是从 20 世纪 70 年代中期到 80 年代后期. 1975 年, 霍兰德出版专著《自然和人工系统的适应性 (Adaptation in Natural and Artificial System)》, 全面介绍了遗传算法.

同年, 德国学者施韦费尔 (Schwefel) 在其博士论文中提出了一种由多个个体组成的群体参与进化的, 并且包括了变异和重组这两种进化操作的进化策略.

1989 年, 霍兰德的学生戈尔德伯格 (Goldberg) 博士出版专著《遗传算法—-搜索、优化及机器学习 (Genetic Algorithm—-in Search Optimization and Machine Learning)》, 使遗传算法得到了普及与推广.

③ 发展阶段

这一阶段是从 20 世纪 90 年代至今. 1989 年, 美国斯坦福 (Stanford) 大学的科扎 (Koza) 提出了遗传规划的新概念, 并于 1992 年出版了专著《遗传规划——应用自然选择法则的计算机程序设计 (Genetic Programming:on the Programming of Computer by Means of Natural Selection)》, 该书全面介绍了遗传规划的基本原理及应用实例, 标志着遗传规划作为计算智能的一个分支已基本形成.

进入 20 世纪 90 年代以来, 进化计算得到了众多研究机构和学者的高度重视, 新的研究成果不断出现、应用领域不断扩大. 目前, 进化计算已成为人工智能领域的又一个新的研究热点.

进化计算的基本结构

进化计算尽管有多个重要分支, 并且不同分支的编码方案、选择策略和进化操作也有可能不同, 但它们却有着共同的进化框架. 若假设 P 为种群 (Population, 或称为群体), t 为进化代数, P(t) 为第 t 代种群, 则进化计算的基本结构可粗略描述如下:

确定编码形式并生成搜索空间; 初始化各个进化参数, 并设置进化代数 $t = 0$; 初始化种群 $P(0)$; 对初始种群进行评价 (即适应度计算); while(不满足终止条件)do $t = t + 1$; 利用选择操作从 $P(t-1)$ 代中选出 $P(t)$ 代群体; 对 $P(t)$ 代种群执行进化操作; 对执行完进化操作后的种群进行评价 (即适应度计算);

上述基本结构包含了生物进化中所必需的选择操作、进化操作和适应度评价等过程.

遗传算法的基本思想是从初始种群出发, 采用优胜劣汰、适者生存的自然法则选择个体, 并通过杂交、变异来产生新一代种群, 如此逐代进化, 直到满足目标为止. 遗传算法所涉及到的基本概念主要有以下几个:

- 种群 (Population): 种群是指用遗传算法求解问题时, 初始给定的多个解的集合. 遗传算法的求解过程是从这个子集开始的.
- 个体 (Individual): 个体是指种群中的单个元素, 它通常由一个用于描述其基本遗传结构的数据结构来表示. 例如, 可以用 0、1 组成的长度为 l 的串来表示个体.

- 染色体 (Chromos): 染色体是指对个体进行编码后所得到的编码串. 染色体中的每 1 位称为基因, 染色体上由若干个基因构成的一个有效信息段称为基因组.
- 适应度 (Fitness) 函数: 适应度函数是一种用来对种群中各个个体的环境适应性进行度量的函数. 其函数值是遗传算法实现优胜劣汰的主要依据.
- 遗传操作 (Genetic Operator): 遗传操作是指作用于种群而产生新的种群的操作. 标准的遗传操作包括以下 3 种基本形式:
- 选择 (Selection)
- 交叉 (Crosssover)
- 变异 (Mutation)

遗传算法

遗传算法的主要组成

染色体编码
初始种群设定
适应度函数设定
遗传操作设计等几大部分所组成.

算法主要步骤

(1) 选择编码策略, 将问题搜索空间中每个可能的点用相应的编码策略表示出来, 即形成染色体;
(2) 定义遗传策略, 包括种群规模 N, 交叉、变异方法, 以及选择概率 P_r、交叉概率 P_c、变异概率 P_m 等遗传参数;
(3) 令 $t = 0$, 随机选择 N 个染色体初始化种群 $P(0)$;
(4) 定义适应度函数 $f(f > 0)$;
(5) 计算 $P(t)$ 中每个染色体的适应值; (6) $t = t + 1$;
(7) 运用选择算子, 从 $P(t-1)$ 中得到 $P(t)$;
(8) 对 $P(t)$ 中的每个染色体, 按概率 P_c 参与交叉;
(9) 对染色体中的基因, 以概率 P_m 参与变异运算; (10) 判断群体性能是否满足预先设定的终止标准, 若不满足则返回 (5).

算法流程

如图23所示.

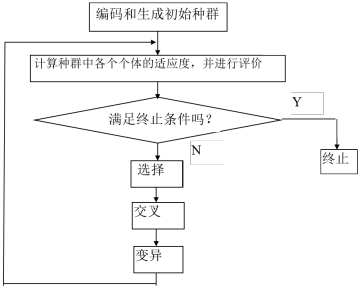


图 23: 基本遗传算法的算法流程图

遗传编码

有霍兰德二进制码、格雷码 (Gray Code)、实数编码和字符编码等.

(1) 二进制编码 (Binary encoding)

二进制编码是将原问题的结构变换为染色体的位串结构. 在二进制编码中, 首先要确定二进制字符串的长度 l, 该长度与变量的定义域和所求问题的计算精度有关.

假设变量 x $\in [5, 10]$, 要求的计算精度为 10^{-5}, 则需要将 [5, 10] 至少分为 600000 个等长小区间, 每个小区间用一个二进制串表示.

串长的要求

串长至少等于 20, 原因是:

$$524288 = 2^{19} < 600000 < 2^{20} = 1048576$$

这样, 对应于区间 $[5, 10]$ 内满足精度要求的每个值 x, 都可用一个 20 位编码的二进制串 $< b_{19}, b_{18}, \cdots, b_0 >$ 来表示.

二进制编码存在的主要缺点是汉明 (Hamming) 悬崖.

Example

7 和 8 的二进制数分别为 0111 和 1000, 当算法从 7 改进到 8 时, 就必须改变所有的位.

(2) 格雷编码 (Gray encoding)

格雷编码是对二进制编码进行变换后所得到的一种编码方法. 这种编码方法要求两个连续整数的编码之间只能有一个码位不同, 其余码位都是完全相同的. 它有效地解决了汉明悬崖问题, 其基本原理如下:

设有二进制串 $b_1, b_2, \cdots, b_n$, 对应的格雷串为 $a_1, a_2, \cdots, a_n$, 则从二进制编码到格雷编码的变换为:

$$a_i = \begin{cases} b_1, & i = 1 \\ b_{i-1} \oplus b_i & i > 1 \end{cases} \tag{53}$$

其中, $\oplus$ 表示模 2 加法. 而从一个格雷串到二进制串的变换为:

$$b_i = \sum_{j=1}^{i} a_i (\mathrm{mod} 2) \tag{54}$$

Example

十进制数 7 和 8 的二进制编码分别为 0111 和 1000, 而其格雷编码分别为 0100 和 1100.

(3) 实数编码 (Real encoding

实数编码是将每个个体的染色体都用某一范围的一个实数 (浮点数) 来表示, 其编码长度等于该问题变量的个数.

这种编码方法是将问题的解空间映射到实数空间上, 然后在实数空间上进行遗传操作. 由于实数编码使用的是变量的真实值, 因此这种编码方法也叫做真值编码方法.

实数编码适应于那种多维、高精度要求的连续函数优化问题.
适应度函数是一个用于对个体的适应性进行度量的函数. 通常, 一个个体的适应度值越大, 它被遗传到下一代种群中的概率也就越大.

(1) 常用的适应度函数

在遗传算法中, 有许多计算适应度的方法, 其中最常用的适应度函数有以下两种:

① 原始适应度函数

它是直接将待求解问题的目标函数 $f(x)$ 定义为遗传算法的适应度函数. 例如, 在求解极值问题

$$\max_{x\in[a,b]} f(x) \tag{55}$$

时, $f(x)$ 即为 x 的原始适应度函数.

采用原始适应度函数的优点是能够直接反映出待求解问题的最初求解目标, 其缺点是有可能出现适应度值为负的情况.

② 标准适应度函数

在遗传算法中, 一般要求适应度函数非负, 并其适应度值越大越好.

② 适应度函数的某种变换

这就往往需要对原始适应函数进行某种变换, 将其转换为标准的度量方式, 以满足进化操作的要求, 这样所得到的适应度函数被称为标准适应度函数 $f_{Normal}(x)$.

极小化问题

对极小化问题, 其标准适应度函数可定义为

$$f(x) = \begin{cases} f_{max}(x) - f(x) & f(x) < f_{max}(x) \\ 0 & f(x) \geq f_{max}(x) \end{cases} \tag{56}$$

其中, $f_{max}(x)$ 是原始适应函数 $f(x)$ 的一个上界. 如果 $f_{max}(x)$ 未知, 则可用当前代或到目前为止各演化代中的 $f(x)$ 的最大值来代替. 可见, $f_{max}(x)$ 是会随着进化代数的增加而不断变化的.

极大化问题

对极大化问题, 其标准适应度函数可定义为:

$$f(x) = \begin{cases} f(x) - f_{min}(x) & f(x) > f_{min}(x) \\ 0 & f(x) \leq f_{min}(x) \end{cases} \tag{57}$$

其中, $f_{min}(x)$ 是原始适应函数 $f(x)$ 的一个下界. 如果 $f_{min}(x)$ 未知, 则可用当前代或到目前为止各演化代中的 $f(x)$ 的最小值来代替.

(2) 适应度函数的加速变换

在某些情况下, 适应度函数在极值附近的变化可能会非常小, 以至于不同个体的适应值非常接近, 使得难以区分出哪个染色体更占优势. 对此, 最好能定义新的适应度函数, 使该适应度函数既与问题的目标函数具有相同的变化趋势, 又有更快的变化速度.

① 线性加速——适应度函数的定义

$$\mathrm{f}'(\mathrm{x}) = \alpha \mathrm{f}(\mathrm{x}) + \beta, \tag{58}$$

其中, $\mathrm{f}(\mathrm{x})$ 是加速转换前的适应度函数; $\mathrm{f}'(\mathrm{x})$ 是加速转换后的适应度函数; α 和 β 是转换系数.

② 非线性加速

- 幂函数变换方法

$$f'(x) = f(x)k \tag{59}$$

- 指数变换方法

$$f'(x) = \exp^{-\beta f(x)} \tag{60}$$

遗传算法中的基本遗传操作

包括选择、交叉和变异 3 种, 而每种操作又包括多种不同的方法, 下面分别对它们进行介绍.

• 选择操作

选择 (Selection) 操作是指根据选择概率按某种策略从当前种群中挑选出一定数目的个体, 使它们能够有更多的机会被遗传到下一代中.

选择策略可分为比例选择、排序选择和竞技选择三种类型

• 比例选择: 比例选择方法 (Proportional Model) 的基本思想是: 各个个体被选中的概率与其适应度大小成正比.

常用的比例选择策略

包括: (1) 轮盘赌选择. (2) 繁殖池选择.

• 轮盘赌选择

轮盘赌选择法又被称为转盘赌选择法或轮盘选择法. 在这种方法中, 个体被选中的概率取决于该个体的相对适应度. 而相对适应度的定义为:

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f\left(x_j\right)}, \tag{61}$$

其中, $P(x_i)$ 是个体 x_i 的相对适应度, 即个体 x_i 被选中的概率; $f(x_i)$ 是个体 x_i 的原始适应度; $\sum_{j=1}^{N} f\left(x_j\right)$ 是种群的累加适应度.

轮盘赌选择算法的基本思想是: 根据每个个体的选择概率 $P(x_i)$ 将一个圆盘分成 N 个扇区, 其中第 i 个扇区的中心角为:

$$2\pi \frac{f(x_i)}{\sum_{j=1}^{N} f_i\left(x_j\right)} = 2\pi p\left(x_i\right) \tag{62}$$

并再设立一个固定指针. 当进行选择时, 可以假想转动圆盘, 若圆盘静止时指针指向第 i 个扇区, 则选择个体 i. 其物理意义如图 5-19 所示.

从统计角度看, 个体的适应度值越大, 其对应的扇区的面积越大, 被选中的可能性也越大. 这种方法有点类似于发放奖品使用的轮盘, 并带有某种赌博的意思, 因此亦被称为轮盘赌选择.

(2) 交叉操作

交叉 (Crossover) 操作是指按照某种方式对选择的父代个体的染色体的部分基因进行交配重组, 从而形成新的个体.

交配重组是自然界中生物遗传进化的一个主要环节, 也是遗传算法中产生新的个体的最主要方法. 根据个体编码方法的不同, 遗传算法中的交叉操作可分为二进制交叉和实值交叉两种类型.

① 二进制交叉

二进制交叉 (Binary Valued Crossover) 是指二进制编码情况下所采用的交叉操作, 它主要包括单点交叉、两点交叉、多点交叉和均匀交叉等方法.

单点交叉也称简单交叉, 它是先在两个父代个体的编码串中随机设定一个交叉点, 然后对这两个父代个体交叉点前面或后面部分的基因进行交换, 并生成子代中的两个新的个体. 假设两个父代的个体串分别是:

$$X = x_1 x_2 \cdots x_k\, x_{k+1} \cdots x_{n-1}\, x_n \tag{63}$$

$$Y = y_1 y_2 \cdots y_k\, y_{k+1} \cdots y_{n-1}\, y_n \tag{64}$$

随机选择第 k 位为交叉点, 若采用对交叉点后面的基因进行交换的方法, 但点交叉是将 X 中的 x_{k+1} 到 x_n 部分与 Y 中的 y_{k+1} 到 y_n 部分进行交叉, 交叉后生成的两个新的个体是:

$$X' = x_1 x_2 \cdots x_k y_{k+1} \cdots x_{n-1} y_n \tag{65}$$

$$Y' = y_1 y_2 \cdots y_k x_{k+1} \cdots y_{n-1} x_n \tag{66}$$

Example

设有两个父代的个体串 $A = 001101$ 和 $B = 110010$, 若随机交叉点为 4, 则交叉后生成的两个新的个体是:

$$A' = 001110 \tag{67}$$

$$B' = 110001 \tag{68}$$

两点交叉

两点交叉是指先在两个父代个体的编码串中随机设定两个交叉点, 然后再按这两个交叉点进行部分基因交换, 生成子代中的两个新的个体. 假设两个父代的个体串分别是:

$$X = x_1\, x_2 \cdots x_i \cdots x_j \cdots x_n \tag{69}$$

$$Y = y_1\, y_2 \cdots y_i \cdots y_j \cdots y_n \tag{70}$$

随机设定第 i, j 位为两个交叉点 (其中 $i < j < n$), 两点交叉是将 X 中的 x_{i+1} 到 x_j 部分与 Y 中的 y_{i+1} 到 y_j 部分进行交换, 交叉后生成的两个新的个体是:

$$X' = x_1\, x_2 \cdots x_i\, y_{i+1} \cdots y_j\, x_{j+1} \cdots x_n \tag{71}$$

$$Y' = y_1\, y_2 \cdots y_i\, x_{i+1} \cdots x_j\, y_{j+1} \cdots y_n \tag{72}$$

Example

设有两个父代的个体串 $A = 001101$ 和 B=110010, 若随机交叉点为 3 和 5, 则交叉后的两个新的个体是:

$$A' = 001011 \tag{73}$$

$$B' = 110100 \tag{74}$$

多点交叉

多点交叉是指先随机生成多个交叉点, 然后再按这些交叉点分段地进行部分基因交换, 生成子代中的两个新的个体.

假设交叉点个数为 m, 则可将个体串划分为 $m+1$ 个分段, 其划分方法是:

- 当 m 为偶数时, 对全部交叉点依次进行两两配对, 构成 $m/2$ 个交叉段.
- 当 m 为奇数时, 对前 $(m-1)$ 个交叉点依次进行两两配对, 构成 $(m-1)/2$ 个交叉段, 而第 m 个交叉点则按单点交叉方法构成一个交叉段.

m = 3 为例

假设两个父代的个体串分别是

$$X = x_1x_2 \cdots x_i \cdots x_j \cdots x_k \cdots x_n$$

和

$$Y = y_1y_2 \cdots y_i \cdots y_j \cdots y_k \cdots y_n$$

随机设定第 i, j, k 位为三个交叉点 (其中 $i < j < k < n$), 则将构成两个交叉段. 交叉后生成的两个新的个体是:

$$X' = x_1x_2 \cdots x_iy_{i+1} \cdots y_jx_{j+1} \cdots x_ky_{k+1} \cdots y_n \quad (75)$$

$$Y' = y_1y_2 \cdots y_ix_{i+1} \cdots x_jy_{j+1} \cdots y_kx_{k+1} \cdots x_n \quad (76)$$

Example

设有两个父代的个体串 $A = 001101$ 和 $B = 110010$, 若随机交叉点为 1、3 和 5, 则交叉后的两个新的个体是:

$$A' = 010100 \tag{77}$$

$$B' = 101011 \tag{78}$$

均匀交叉

均匀交叉 (Uniform Crossover) 是先随机生成一个与父串具有相同长度, 并被称为交叉模版 (或交叉掩码) 的二进制串, 然后再利用该模版对两个父串进行交叉, 即将模版中 1 对应的位进行交换, 而 0 对应的位不交换, 依此生成子代中的两个新的个体.

事实上, 这种方法对父串中的每一位都是以相同的概率随机进行交叉的.

⑥ 实值交叉

实值交叉是在实数编码情况下所采用的交叉操作, 主要包括离散交叉和算术交叉, 下面主要讨论离散交叉 (部分离散交叉和整体离散交叉) .

部分离散交叉是先在两个父代个体的编码向量中随机选择一部分分量, ¤ 换, 生成子代中的两个新的个体.

整体交叉则是对两个父代个体的编码向量中的所有分量, 都以 1/2 的概率进行交换, 从而生成子代中的两个新的个体.

以部分离散交叉为例, 假设两个父代个体的 n 维实向量分别是 $X = x_1x_2\cdots x_i\cdots x_k\cdots x_n$ 和 $Y = y_1y_2\cdots y_i\cdots y_k\cdots y_n$, 若随机选择对第 k 个分量以后的所有分量进行交换, 则生成的两个新的个体向量是:

$$X' = x_1x_2\cdots x_ky_{k+1}\cdots y_n \tag{79}$$

$$Y' = y_1y_2\cdots y_kx_{k+1}\cdots x_n \tag{80}$$

Example

设有两个父代个体向量 $\mathrm{A} = 20\ 16\ 19\ 32\ 18\ 26$ 和 $\mathrm{B} = 36\ 25\ 38\ 12\ 21\ 30$, 若随机选择对第 3 个分量以后的所有分量进行交叉, 则交叉后两个新的个体向量是:

$$\mathrm{A}' = 20\ 16\ 19\ 12\ 21\ 30 \tag{81}$$

$$\mathrm{B}' = 36\ 25\ 38\ 32\ 18\ 26 \tag{82}$$

(3) 变异操作

变异 (Mutation) 是指对选中个体的染色体中的某些基因进行变动, 以形成新的个体. 变异也是生物遗传和自然进化中的一种基本现象, 它可增强种群的多样性. 遗传算法中的变异操作增加了算法的局部随机搜索能力, 从而可以维持种群的多样性.

根据个体编码方式的不同, 变异操作可分为二进制变异和实值变异两种类型.

二进制变异

当个体的染色体采用二进制编码表示时, 其变异操作应采用二进制变异方法. 该变异方法是先随机地产生一个变异位, 然后将该变异位置上的基因值由“0”变为“1”, 或由“1”变为“0”, 产生一个新的个体.

Example

设变异前的个体为 $A = 001101$, 若随机产生的变异位置是 2, 则该个体的第 2 位由“0”变为“1”. 变异后的新的个体是 $A' = 011101$.

实值变异

当个体的染色体采用实数编码表示时, 其变异操作应采用实值变异方法. 该方法是用另外一个在规定范围内的随机实数去替换原变异位置上的基因值, 产生一个新的个体. 最常用的实值变异操作有:

基于位置的变异方法

该方法是先随机地产生两个变异位置, 然后将第二个变异位置上的基因移动到第一个变异位置的前面.

Example

设选中的个体向量 $C = 20\,16\,19\,12\,21\,30$, 若随机产生的两个变异位置分别是 2 和 4, 则变异后的新的个体向量是:

$$C' = 20\,12\,16\,19\,21\,30. \tag{83}$$

基于次序的变异

该方法是先随机地产生两个变异位置, 然后交换这两个变异位置上的基因.

Example

设选中的个体向量 $\mathrm{D} = 20\,12\,16\,19\,21\,30$, 若随机产生的两个变异位置分别是 2 和 4, 则变异后的新的个体向量是:

$$\mathrm{D}' = 20\,19\,16\,12\,21\,30 \tag{84}$$

Example

用遗传算法求函数 $f(x) = x_2$ 的最大值, 其中 $x \in [0, 31], x \in \mathbb{Z}$ 间的整数.

这个问题本身比较简单, 其最大值很显然是在 $x = 31$ 处. 但作为一个例子, 它有着较好的示范性和可理解性. 按照遗传算法, 其求解过程如下:

(1) 编码

由于 x 的定义域是区间 $[0, 31]$ 上的整数, 由 5 位二进制数即可全部表示. 因此, 可采用二进制编码方法, 其编码串的长度为 5.

Example

用二进制串 00000 来表示, x $= 011111$ 表示 x $= 31$ 等. 其中的 0 和 1 为基因值.

(2) 生成初始种群

若假设给定的种群规模 N $= 4$, 则可用 4 个随机生成的长度为 5 的二进制串作为初始种群. 再假设随机生成的初始种群 (即第 0 代种群) 为:

$$s_{01} = 01101 \qquad s_{02} = 11001$$
$$s_{03} = 01000 \qquad s_{04} = 10010$$

(3) 计算适应度

要计算个体的适应度, 首先应该定义适应度函数. 由于本例是求 f(x) 的最大值, 因此可直接用 f(x) 来作为适应度函数. 即:

$$f(s) = f(x), \tag{85}$$

其中的二进制串 s 对应着变量 x 的值.

根据此函数, 初始种群中各个个体的适应值及其所占比例如表9所示.

表 9: 初始种群情况表

编号	个体串 (染色体)	x	适应值	百分比%	累计百分比%	选中次数
S_{01}	01101	13	169	14.44	14.44	1
S_{02}	11001	25	625	52.88	67.18	2
S_{03}	01000	8	64	5.41	72.59	0
S_{04}	10010	18	324	27.41	100	1

可以看出, 在 4 个个体中 S_{03} 的适应值最小, 是当前最佳个体.

(4) 选择操作

假设采用轮盘赌方式选择个体, 且依次生成的 4 个随机数 (相当于轮盘上指针所指的数) 为 0.85、0.32、0.12 和 0.46, 经选择后得到的新的种群为:

表 10: 初始种群情况表

S01=10010
S02=11001
S03=01101
S04=11001

其中, 染色体 11001 在种群中出现了 2 次, 而原染色体 01000 则因适应值太小而被淘汰.

(5) 交叉

假设交叉概率 π 为 50%, 则种群中只有 1/2 的染色体参与交叉. 若规定种群中的染色体按顺序两两配对交叉, 且有 S_{01} 与 S_{02} 交叉, S_{03} 与 S_{04} 不交叉, 则交叉情况如表 **??** 所示.

表 11: 初始种群的交叉情况表

编号	个体串 (染色体)	交叉对象	交叉位	子代	适应值
S_{01}	10010	S02	3	10001	289
S_{02}	11001	S01	3	11010	676
S_{03}	01101	S04	N	01101	169
S_{04}	11001	S03	N	11001	625

经交叉后的新种群

表 12: 种群情况表

S_{01}=10001
S_{02}=11010
S_{03}=01101
S_{04}=11001

(6) 变异

变异概率 P_m 一般都很小, 假设本次循环中没有发生变异, 则变异前的种群即为进化后所得到的第 1 代种群. 即:

表 13: 种群情况表

S_{11}=10001
S_{12}=11010
S_{13}=01101
S_{14}=11001

然后, 对第 1 代种群重复上述 (4)-(6) 的操作.

对第 1 代种群, 同样重复上述 (4)-(6) 的操作. 其选择情况如表14所示.

表 14: 第 1 代种群的选择情况表

编号	个体串 (染色体)	x	适应值	百分比%	累计百分比%	选中次数
S_{11}	10001	27	289	16.43	16.437	1
S_{12}	11010	26	676	38.43	54.86	2
S_{13}	01101	13	169	9.61	64.47	0
S_{14}	11001	25	625	35.53	100	1

若假设按轮盘赌选择时依次生成的 4 个随机数为 0.14、0.51、0.24 和 0.82, 经选择后得到的新的种群为:

表 15: 种群情况表

S11=10001
S_{12}=11010
S_{13}=11010
S_{14}=11001

可以看出, 染色体 11010 被选择了 2 次, 而原染色体 01101 则因适应值太小而被淘汰.

对第 1 代种群, 其交叉情况如表16所示.

表 16: 第 1 代种群的交叉情况表

编号	个体串 (染色体)	交叉对象	交叉位	子代	适应值
S_{11}	10001	S_{12}	3	10010	324
S_{12}	11010	S_{11}	3	11001	625
S_{13}	11010	S_{14}	2	11001	625
S_{14}	11001	S_{13}	2	11010	675

可见, 经杂交后得到的新的种群为:

表 17: 种群情况表

S_{11}=10010
S_{12}=11001
S_{13}=11001
S_{14}=11010

可以看出, 第 3 位基因均为 0, 已经不可能通过交配达到最优解. 这种过早陷入局部最优解的现象称为早熟. 为解决这一问题, 需要采用变异操作.

对第 1 代种群, 其变异情况如表18所示.

表 18: 第 1 代种群的变异情况表

编号	个体串 (染色体)	是否变异	变异位	子代	适应值
S_{11}	10010	N		10010	324
S_{12}	11001	N		11001	625
S_{13}	11001	N		11001	625
S_{14}	11010	Y		11110	900

它是通过对 S_{14} 的第 3 位的变异来实现的. 变异后所得到的第 2 代种群为:

表 19: 种群情况表

S_{21}=10010
S_{22}=11001
S_{23}=11001
S_{24}=11110

接着, 再对第 2 代种群同样重复上述 (4)-(6) 的操作:

对第 2 代种群, 同样重复上述 (4)-(6) 的操作. 其选择情况如表20所示.

表 20: 第 2 代种群的选择情况表

编号	个体串 (染色体)	x	适应值	百分比%	累计百分比%	选中次数
S_{21}	10010	18	324	23.92	23.92	1
S_{22}	11001	25	625	22.12	46.04	1
S_{23}	11001	25	625	22.12	68.16	1
S_{24}	11110	30	900	31.84	100	1

对第 2 代种群, 其交叉情况如表21所示.

表 21: 第 2 代种群的选择情况表

编号	个体串 (染色体)	交叉对象	交叉位	子代	适应值
S_{21}	11001	S22	3	11010	676
S_{22}	10010	S21	3	10001	289
S_{23}	11001	S24	4	11000	576
S_{24}	11110	S23	4	11111	961

这时, 函数的最大值已经出现, 其对应的染色体为 11111, 经解码后可知问题的最优解是在点 $x = 31$ 处. 求解过程结束.

若假设按轮盘赌选择时依次生成的 4 个随机数为 0.42、0.15、0.59 和 0.91, 经选择后得到的新的种群为:

表 22: 种群情况表

S_{21}=11001
S_{22}=10010
S_{23}=11001
S_{24}=11110

图 24: 种群的二进制编码

美国加州大学扎德 (Zadeh) 教授于 1965 年提出的模糊集合与模糊逻辑理论是模糊计算的数学基础. 它主要用来处理现实世界中因模糊而引起的不确定性. 目前, 模糊理论已经在推理、控制、决策等方面得到了非常广泛的应用. 本节主要讨论模糊理论的基础, 对于模糊推理将放到下一章讨论.

20 世纪 70 年代中期, 中国引进模糊数学. 80 年代在理论研究和应用研究方面都非常活跃并取得了大量研究成果, 主要代表人物有著名数学家汪培庄、张文修、蒲保明、刘应明、王国俊等教授. 1988 年, 汪培庄教授及其指导的几名博士生研制成功一台模糊推理机——分立元件样机, 它的推理速度为 1500 万次/秒; 这表明中国在突破模糊信息处理难关方面迈出了重要的一步, 确实是一项了不起的研究成果. 2005 年, 中国科学院院士刘应明教授获国际模糊系统协会（IFSA）“Fuzzy Fellow”称号, 是作为首位非发达国家学者获此殊荣.

图 25: 汪培庄教授

中国著名学者周海中教授曾指出: “模糊数学的诞生, 是科学技术发展的必然结果, 更是现代数学发展的必然产物. 但就现状而言, 模糊数学的理论尚未成熟、体系还未形成, 对它也还存在不同看法和意见; 这些都有待日后完善和实践检验. ”

模糊集

设 U 是给定论域, 是把任意 $u \in U$ 映射为 $[0,1]$ 上某个实值的函数, 即

$$U \to [0,1] \tag{86}$$

$$u \to \mu_F(u) \tag{87}$$

则称 $\mu_F(x)$ 为定义在 U 上的一个隶属函数, 由 (对所有) 所构成的集合 F 称为 U 上的一个模糊集, 称为 u 对 F 的隶属度.

① 模糊集 F 完全是由隶属函数 μ_F 来刻画的, 把 U 中的每一个元素 u 都映射为 $[0,1]$ 上的一个值 $\mu_F(x)$.
② $\mu_F(u)$ 的值表示 u 隶属于 F 的程度, 其值越大, 表示 u 隶属于 F 的程度越高. 当 $\mu_F(u)$ 仅取 0 和 1 时, 模糊集 F 便退化为一个普通集合.

模糊集及其运算

Example

设论域 $\mathsf{U} = \{20, 30, 40, 50, 60\}$ 给出的是年龄, 请确定一个刻画模糊概念“年轻”的模糊集 F.

解: 由于模糊集是用其隶属函数来刻画的, 因此需要先求出描述模糊概念“青年”的隶属函数. 假设对论域 U 中的元素, 其隶属函数值分别为:

$$\begin{aligned} &\mu_{\mathsf{F}}(20) = 1, \mu_{\mathsf{F}}(30) = 0.8, \mu_{\mathsf{F}}(40) = 0.4 \\ &\mu_{\mathsf{F}}(50) = 0.1, \mu_{\mathsf{F}}(60) = 0 \end{aligned} \tag{88}$$

则可得到刻画模糊概念“年轻”的模糊集

$$\mathsf{F} = \{1, 0.8, 0.4, 0.1, 0\}. \tag{89}$$

(1) 离散且为有限论域的表示方法

设论域 $U = \{u_1, u_2, \ldots, u_n\}$ 为离散论域, 则其模糊集可表示为:

$$F = \{\mu_F(u_1), \mu_F(u_2), \ldots, \mu_F(u_n)\} \tag{90}$$

为了能够表示出论域中的元素与其隶属度之间的对应关系, 扎德引入了一种模糊集的表示方式: 先为论域中的每个元素都标上其隶属度, 然后再用“+”号把它们连接起来, 即

$$F = \mu_F(u_1)/u_1 + \mu_F(u_2)/u_2 + \cdots + \mu_F(u_n)/u_n. \tag{91}$$

另一种形式

$$F = \sum_{i=1}^{n} \mu_F(u_i)/u, \tag{92}$$

其中, $\mu_F(u_i)$ 为 u_i 对 F 的隶属度;

“$\mu_{\mathrm{F}}(\mathrm{u_i})|\mathrm{u_i}$”不是相除关系, 只是一个记号; “+”也不是算术意义上的加, 只是一个连接符号.

$$\begin{aligned}\mathrm{F} &= \{\mu_{\mathrm{F}}(\mathrm{u}_1)/\mathrm{u}_1, \mu_{\mathrm{F}}(\mathrm{u}_2)/\mathrm{u}_2, \cdots, \mu_{\mathrm{F}}(\mathrm{u}_n)/\mathrm{u}_n\} \\ \mathrm{F} &= \{(\mu_{\mathrm{F}}(\mathrm{u}_1), \mathrm{u}_1), (\mu_{\mathrm{F}}(\mathrm{u}_2), \mathrm{u}_2), \cdots, (\mu_{\mathrm{F}}(\mathrm{u}_n), \mathrm{u}_n)\}\end{aligned} \tag{93}$$

在这种表示方法中, 当某个 $\mathrm{u_i}$ 对 F 的隶属度 =0 时, 可省略不写. 例如, 前面例 5.15 的模糊集 F 可表示为:

$$\mathrm{F} = 1/20 + 0.8/30 + 0.6/40 + 0.2/50 \tag{94}$$

有时, 模糊集也可写成如下两种形式:

$$\begin{aligned} \mathsf{F} &= \{\mu_{\mathsf{F}}(\mathsf{u}_1)/\mathsf{u}_1, \mu_{\mathsf{F}}(\mathsf{u}_2)/\mathsf{u}_2, \cdots, \mu_{\mathsf{F}}(\mathsf{u}_n)/\mathsf{u}_n\} \\ \mathsf{F} &= \{(\mu_{\mathsf{F}}(\mathsf{u}_1), \mathsf{u}_1), (\mu_{\mathsf{F}}(\mathsf{u}_2), \mathsf{u}_2), \cdots, (\mu_{\mathsf{F}}(\mathsf{u}_n), \mathsf{u}_n)\} \end{aligned} \tag{95}$$

其中, 前一种称为单点形式, 后一种称为序偶形式.

(2) 连续论域的表示方法

如果论域是连续的, 则其模糊集可用一个实函数来表示.

年龄为论域

“年轻”与“年老”这两的模糊概念的隶属函数

$$\mu_{老} = \frac{(\mathrm{l})}{\xi}^{*}(\mathsf{u}) = \begin{cases} 0, & 0 \le \mathsf{u} \le 50 \\ \left[1 + \left(\frac{5}{\mathsf{u}-50}\right)^2\right]^{-1}, & 50 < \mathsf{u} \le 100 \end{cases} \tag{96}$$

$$\mu_{年轻}(\mathsf{u}) = \begin{cases} 1, & 0 \le \mathsf{u} \le 25 \\ \left[1 + \left(\frac{\mathsf{u}-25}{5}\right)^2\right]^{-1}, & 25 < 4 \le 100 \end{cases} \tag{97}$$

模糊集

设 F_i 是 $U_i(i = 1, 2, \cdots, n)$ 上的模糊集, 则称

$$F_1 \times F_2 \times \cdots \times F_n = \int_{u_1 \times u_2 \times \cdots \times u_n} \left(\mu_{F_2}(u_1) \wedge \mu_{F_2}(u_2) \wedge \cdots \wedge \mu_{F_n}(u_n)\right) / (u_1, u_2, \cdots, u_n \tag{107}$$

为 $F_1, F_2, \cdots, F_n$ 的笛卡尔乘积, 它是 $U_1 \times U_2 \times \cdots \times U_n$ 上的一个模糊集.

模糊关系

在 $\prod_{i=1}^{n} U_i$ 上的一个 n 元模糊关系 R 是指以 $\prod_{i=1}^{n} U_i$ 上为论域的一个模糊集，记为

$$R = \int_{U_1 \times U_2 \times \cdots \times U_n} \mu_R\left(u_1, u_2, \cdots, u_n\right) / \left(u_1, \cdots, u_n\right). \tag{108}$$

Example

设有一组学生 $U = u_1, u_2$ ={秦学, 郝玩}, 一些在计算机上的活动 $V = v_1, v_2, v_3$ ={编程, 上网, 玩游戏} 并设每个学生对各种活动的爱好程度分别为 $\mu_F\left(u_i, v_j\right)$ $i = 1, 2; j = 1, 2, 3$, 即

$$\mu_R(u_i, v_j) = \mu_R \begin{cases} (u_1, v_1), & (u_1, v_2), & (u_1, v_3) \\ (u_2, v_1), & (u_2, v_2), & (u_2, v_3) \end{cases} \tag{109}$$

则 $U \times V$ 上的模糊关系 R 为

$$R = \begin{bmatrix} 0.9 & 0.6 & 0 \\ 0.2 & 0.3 & 0.8 \end{bmatrix}. \tag{110}$$

模糊关系的合成

模糊关系

设 R_1 与 R_2 分别是 $U \times V$ 与 $U \times W$ 上的两个模糊关系, 则 R_1 与 R_2 的合成是从 U 到 W 的一个模糊关系, 记为 $R_1 \circ R_2$. 其隶属函数为

$$\mu_{R_1 R_2}(u, w) = \vee \left\{ \mu_{R_1}(u, v) \wedge \mu_{R_2}(v, w) \right\}, \tag{111}$$

其中, $\wedge$ 和 $\vee$ 分别表示取最小和取最大.

Example

设有以下两个模糊关系

$$\mathsf{R}_1 = \begin{bmatrix} 0.4 & 0.5 & 0.6 \\ 0.8 & 0.3 & 0.7 \end{bmatrix},$$

$$\mathsf{R}_2 = \begin{bmatrix} 0.7 & 0.9 \\ 0.2 & 0.8 \\ 0.5 & 0.3 \end{bmatrix},$$

则 R_1 与 R_2 的合成是

$$\mathsf{R} = \mathsf{R}_1 \circ \mathsf{R}_2 = \begin{bmatrix} 0.5 & 0.5 \\ 0.7 & 0.8 \end{bmatrix}. \tag{112}$$

模糊变换

其方法是把 R_1 的第 i 行元素分别与 R_2 的第 j 列的对应元素相比较, 两个数中取最小者, 然后再在所得的一组最小数中取最大的一个, 并以此数作为 $R_1 \circ R_2$ 的元素 $R(i, j)$.

模糊变换

设 $F = \{\mu_F(u_1), \mu_F(u_2), \cdots, \mu_F(u_n)\}$, 是论域 U 上的模糊集, R 是 $U \times V$ 上的模糊关系, 则 $F \circ R = G$ 称为模糊变换.

Example

则

$$\begin{aligned}\mathsf{G} = \mathsf{F} \circ \mathsf{R} = \{&1 \wedge 1 \vee 0.6 \wedge 0.5 \vee 0.2 \wedge 0, 1 \wedge 0.5 \vee 0.6 \wedge 1 \vee 0.2 \wedge 0.5,\\ &1 \wedge 0 \vee 0.6 \wedge 0.5 \vee 0.2 \wedge 1, 1 \wedge 0 \vee 0.6 \wedge 0 \vee 0.2 \wedge 0.5\}\\ = \{&1, 0.6, 0.5, 0.2\}.\end{aligned}$$

=

探索

用遗传算法求 $f(x) = x\sin(10x) + 1.0$ 的最大值, 其中 $x \in [-1, 2]$.

探索

设有论域 $U = \{u_1, u_2, u_3, u_4, u_5\}$, 并设 F, G 是 U 上的两个模糊集, 且有

$$F = 0.9/u_1 + 0.7/u_2 + 0.5/u_3 + 0.3/u_4$$
$$G = 0.6/u_3 + 0.8/u_4 + 1/u_5.$$

请分别计算 $F \cap G, F \cup G, \neg F$.

探索

设有如下两个模糊关系:

$$\mathsf{R}_1 = \begin{bmatrix} 0.3 & 0.7 & 0.2 \\ 1 & 0 & 0.4 \\ 0 & 0.5 & 1 \end{bmatrix}$$

$$\mathsf{R}_2 = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.9 & 0.1 \end{bmatrix}.$$

请写出 R_1 与 R_2 的合成 $\mathsf{R}_1 \circ \mathsf{R}_2$.

探索

基于剪枝技术的一字棋博弈系统

1. 实验目的: 理解和掌握博弈树的启发式搜索过程, 能够用某种程序语言建立一个简单的博弈系统.
2. 实验环境: 在微型计算机上, 任选一种编程语言.
3. 实验要求

(1) 规定棋盘大小为 5 行 5 列, 要求自行设计估价函数, 按极大极小搜索方法, 并采用 $\alpha - \beta$ 剪枝技术.

(2) 采用人机对弈方式, 一方走完一步后, 等待对方走步, 对弈过程每一时刻的棋局都在屏幕上显示出来.

(3) 提交完整的软件系统和相关文档, 包括源程序和可执行程序.

参考文献