人工智能与神经网络实验探究

目录

引言

Algol 语言的发明者 Alan J. Perlis 曾经在著名的人工智能入门读物《计算机程序的构造和解释》（SICP，*Structure and Interpreter of Computer Programs*）中对人工智能的本质和未来做出了伟大的预言。他说：“……人工智能的研究产生出许多重要的程序设计问题。……问题的洪水孵化出一种又一种新的语言。确实，在任何非常大的程序设计工作中，一条有用的组织原则就是通过发明新语言，去控制和隔离作业模块之间的信息流动。这些语言趋向于越来越不基本，逐渐逼近系统的边界，逼近我们作为人最经常与之交互的地方。”其实，所谓“我们作为人最经常与之交互的地方”指的就是“高级的”人机界面，也就是人与人交流使用的最经常的方式；所谓的“不基本”，是指抽象的层次更高，更加复杂。

本文将结合个人的人工智能实验，来探讨人工智能的真正效果和展望。

概述

人工智能作为一种新兴的领域，其发展和演化十分令人瞩目。维基百科上对于人工智能的定义是：智慧主体（Intelligent Agent）的研究与设计。当前主要的人工智能应用领域主要有机器视觉、指纹识别、人脸识别、视网膜识别、虹膜识别、掌纹识别、专家系统、自动规划等；主要算法类型有机器学习、专家系统等等。

本次实验我们只使用 BP（Back Propagation，反向传播）算法的 DNN（Deep Neural Network，深层神经网络）来进行实验。深层反向传播神经网络是一种结构清晰的神经网络，主要由多个层（Layer）组成，分为输入层（Input Layer）、输出层（Output Layer）和多个隐藏层（Hidden Layers）三个部分。每个层有多个神经元（Neuron），每个人工神经元有一个权值和，人工神经元之间有“连接”，所谓的反向传播和正向传播指的就是神经元之间的数值传播和计算。正向传播是指输入层的多个数据经过隐藏层的多次计算（具体规则是将每个输入的数据乘上相应权值后加一个常数，再根据 Sigmoid 或 Relu 或 Leaky-Relu 或 Tanh 函数来计算传播给下一层的数据），得到输出层的输出；反向传播是指训练时，将标准的答案和网络的输出按反向传播算法计算出残差，再将残差反向（即从输出层到输入层）传播，以更改每个神经元的权值等。

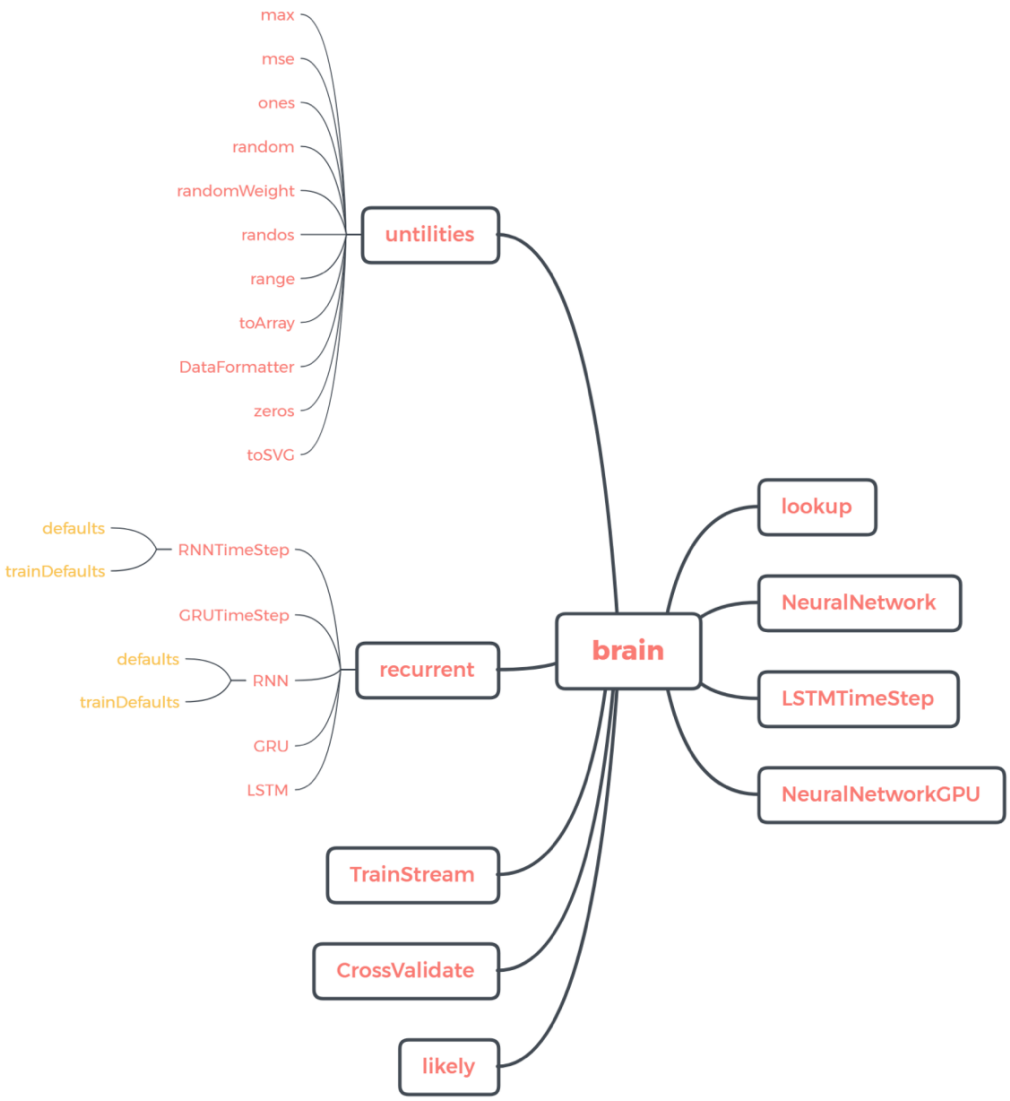
方法探讨

为切实体验人工智能程序的整个编写过程，我们采用一个开源的即开即用的 JavaScript 库 brain.js 来进行神经网络实验。JavaScript 是一种语法简便的泛类型解释性语言，其主流解释器 V8 引擎（V8 JavaScript Engine）是一种开源的高速 JavaScript 解释器，采用大量优化技术来提高解释速度。V8 引擎被用于 Google 的 Chrome 浏览器中解释 JavaScript 和本地解释器 Node.js。Node.js 可以方便地使用 JavaScript 进行本地实验，而 JavaScript 与 HTML 和 CSS 结合可以编写出可交

互式网站和 Web 应用，HTML 便捷的布局语法和 HTML DOM 使得本次实验能够方便地构建出图形界面。用 Node.js 实现后端 AI，HTML、CSS、JavaScript 编写前端代码，可以方便快捷地将这个神经网络库投入应用。

brain.js 是一套方便好用的神经网络算法库，内置了多种神经网络的算法，而且非常方便新手调用，几乎不需要学习任何相关的知识。

从 github.com 上克隆（clone）brain.js 项目后，阅读 README.MD 是必不可少的一步。brain.js 的基本构造如下：



JavaScript 库 brain.js 的神经网络类构造器（即 NeuralNetwork）采用在构造器中传入对象的方法进行训练，训练所用函数是(new NeuralNetwork()).train()其传入对象 config 是一个具有一些属性的对象。其主要属性见下表：

成员名	作用
iterations	迭代次数的最大值，大于 0 的值。
errorThresh	从训练数据处可接受的误差

log	值为 true，则当函数执行时使用 console.log 输出日志；值为函数则将日志传入函数的参数；值为 false 则不输出。
logPeriod	函数迭代间隔，大于 0 的值
learningRate	学习率，介于 0 和 1 之间
momentum	下一层的改变率，介于 0 和 1 之间
callback	回调函数
callbackPeriod	回调间隔
timeout	计算超时时间
hiddenLayers	隐藏层的数量列表
activation	激活函数，支持 sigmod、relu、leaky-relu 和 tanh

实验

我们先使用 Node.js 进行 brain.js 的初步试验。第一步，我们将使用一个三层的反向传播神经网络训练一个异或（XOR，eXclusive OR）逻辑门。第一步，我们先创建 NeuralNetwork 对象。

```
const brain=require('brain.js');
const network=new brain.NeuralNetwork({
    hiddenLayers: [5],
    activation: 'sigmoid'
});
```

通过以上操作，我们初始化了一个神经网络 network，可用于学习。现在，我们向其中传入测试数据：

```
network.train([{input: [0, 0], output: [0]},
               {input: [0, 1], output: [1]},
               {input: [1, 0], output: [1]},
               {input: [1, 1], output: [0]}]);
```

经过训练，网络返回了一些信息：

```
{ error: 0.004998823764627628, iterations: 5282 }
```

我们可以看到，本次学习的残差是 0.004998823764627628，共迭代了 5282 次。现在我们来查看训练的效果：

```
console.log(network.run([0,0]));
console.log(network.run([1, 0]));
console.log(network.run([0, 1]));
console.log(network.run([1, 1]));
```

网络给出了积极的反应：

```
> console.log(network.run([0, 0]));
Float32Array [ 0.04361853376030922 ]
```

```
undefined
> console.log(network.run([1, 0]));
Float32Array [ 0.9326930046081543 ]
undefined
> console.log(network.run([0, 1]));
Float32Array [ 0.936134934425354 ]
undefined
> console.log(network.run([1, 1]));
Float32Array [ 0.09677374362945557 ]
undefined
```

令我们震惊的是，这一结果基本达到了使用 CMOS 制作的逻辑门的水平。（按 5V 电压计算，高电平 4.65V，低电平 0.2V）

我们可以看出，网络很好地学习了 XOR 输入和输出之间的关系，此网络可以作为一种 XOR 逻辑门的实现投入应用。

为了体验神经网络的神奇，我们对网络 network 重新初始化，将 NAND 门的真值表传入网络，看看它能否适应不同的场景。现在，我们向其中重新传入测试数据：

```
network.train([{input: [0, 0], output: [1]},
               {input: [0, 1], output: [1]},
               {input: [1, 0], output: [1]},
               {input: [1, 1], output: [0]}]);
```

网络的运行取得了令人震惊的效果，输出数据低电平与高电平仍然清晰可见：

```
> console.log(network.run([0,1]));
Float32Array [ 0.9387826919555664 ]
undefined
> console.log(network.run([1,1]));
Float32Array [ 0.1103096678853035 ]
undefined
> console.log(network.run([0,1]));
Float32Array [ 0.9387826919555664 ]
undefined
> console.log(network.run([0,0]));
Float32Array [ 0.9995383620262146 ]
undefined
```

令人遗憾的是，该网络的低电平电压较高（按 5V 计算，达到约 0.55V）

实验感言

从刚才的实验中，我已经体会到了神经网络算法的神奇，它能够通过学习一些给定的数据，得到解决问题的算法。

发展展望

人工智能属于“朝阳产业”，其研究成果能够非常直接地为人类的生活提供各种便捷。但是，人工智能算法的正确性得不到保证，可靠性令人怀疑，所以提高人工智能算法的鲁棒性和普适性是未来的努力方向之一。

人工智能普及后的伦理问题也是一大亟待解决的问题。人工的智慧主体是否应该拥有与人等同的权利？人工智能普及后应怎样协调其与人之间的关系？人工智能算法出现错误应怎样干预？这些就留待未来的研究者思考吧。