机器学习

目录

1. 微积分

1.1. 微分学.

1.2. 偏导数与导数.

1.3. 积分学.

1.4. 曲线积分与曲面积分.

2. 级数学

2.1. 多元微积分的 taylor 展开.

3. 线性代数

线性代数主要研究两部分内容, 线性空间和线性变换. 从几何的概念出发, 线性空间也叫向量空间„ 向量和向量空间都是线性空间中的主要概念. 线性代数主要的概念是同构, 一般来讲, 所有的有限维空间都可以和 $\mathbb{R}^n$ 同构, 所有 n 维线性空间到自身的线性变换也可以构成一个线性空间, 并且可以和 $n \times n$ 矩阵组成的线性空间同构. 所有 n 维线性空间到 m 维线性变换也可以构成一个线性空间, 并且可以和 $n \times m$ 矩阵组成的线性空间同构.

3.1. **线性空间.**

3.1.1. 线性空间.

定义 3.1 (线性空间)**.**

定义 3.2 (向量)**.**

定义 3.3 (线性子空间)**.**

线性空间的基和维数
子空间
线性空间的同构
商空间
有范数的线性空间
有度量的线性空间
欧几里德空间
正交空间
辛空间
U 空间

3.2. **线性变换.**

定义 3.4 (线性变换)**.**

定义 3.5 (矩阵)**.**

矩阵的特征值与特征向量
矩阵的范数
矩阵的相抵与相似
矩阵的合同
线性变换的不变子空间

线性变换的 Jordan 标准型
线性变换的有理标准型
线性函数与对偶空间
U 变换
Hermite 变换
正规变换

3.3. 多线性代数.

3.3.1. 多线性代数.

定义 3.6 (多线性代数).

3.3.2. 张量.

定义 3.7 (张量).

例 3.1 (多元函数的任意解导数).

3.3.3. 外代数.

4. 机器学习

Langley（1996) 定义的机器学习是“机器学习是一门人工智能的科学，该领域的主要研究对象是人工智能，特别是如何在经验学习中改善具体算法的性能”。(Machine learning is a science of the artificial. The field's main objects of study are artifacts, specifically algorithms that improve their performance with experience.')

Tom Mitchell 的机器学习 (1997) 对信息论中的一些概念有详细的解释, 其中定义机器学习时提到，“机器学习是对能通过经验自动改进的计算机算法的研究”。(Machine Learning is the study of computer algorithms that improve automatically through experience.)

Alpaydin（2004）同时提出自己对机器学习的定义，“机器学习是用数据或以往的经验，以此优化计算机程序的性能标准。”(Machine learning is programming computers to optimize a performance criterion using example data or past experience.)

4.1. 十大经典机器学习算法.

4.1.1. *决策树*. 根据一些 feature（特征）进行分类，每个节点提一个问题，通过判断，将数据分为两类，再继续提问。这些问题是根据已有数据学习出来的，再投入新数据的时候，就可以根据这棵树上的问题，将数据划分到合适的叶子上。

4.1.2. *随机森林*. 在源数据中随机选取数据，组成几个子集：S 矩阵是源数据，有 1-N 条数据，A、B、C 是 feature，最后一列 C 是类别：由 S 随机生成 M 个子矩阵：这 M 个子集得到 M 个决策树：将新数据投入到这 M 个树中，得到 M 个分类结果，计数看预测成哪一类的数目最多，就将此类别作为最后的预测结果。

4.1.3. *逻辑回归*. 当预测目标是概率这样的，值域需要满足大于等于 0，小于等于 1 的，这个时候单纯的线性模型是做不到的，因为在定义域不在某个范围之内时，值域也超出了规定区间。那么怎么得到这样的模型呢?

这个模型需要满足两个条件“大于等于 0”，“小于等于 1”。大于等于 0 的模型可以选择绝对值，平方值，这里用指数函数，一定大于 0；小于等于 1 用除法，分子是自己，分母是自身加上 1，那一定是小于 1 的了。

(1)$p \geq 0$

$$p = exp(\beta_0 + \beta_1 age)$$

(2)$p \leq 1$

$$p = \frac{exp(\beta_0 + \beta_1 age)}{exp(\beta_0 + \beta_1 age) + 1}$$

于是就得到

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 age$$

4.1.4. *支持向量机*. 要将两类分开，想要得到一个超平面，最优的超平面是到两类的 margin 达到最大，margin 就是超平面与离它最近一点的距离

将这个超平面表示成一个线性方程，在线上方的一类，都大于等于 1，另一类小于等于 -1

$g(x) \geq 1, \forall x \in class1$
$g(x) \leq 1, \forall x \in class2$

4.1.5. *朴素贝叶斯*.

4.1.6. *K 近邻算法*. 给一个新的数据时，离它最近的 k 个点中，哪个类别多，这个数据就属于哪一类。

4.1.7. *K 均值算法*. 先要将一组数据，分为三类，粉色数值大，黄色数值小。最开始先初始化，这里面选了最简单的 3，2，1 作为各类的初始值。剩下的数据里，每个都与三个初始值计算距离，然后归类到离它最近的初始值所在类别。

分好类后，计算每一类的平均值，作为新一轮的中心点

几轮之后，分组不再变化了，就可以停止了：

4.1.8. *Adaboost.* Adaboost 是 Boosting 的方法之一。Boosting 就是把若干个分类效果并不好的分类器综合起来考虑，会得到一个效果比较好的分类器。

下图，左右两个决策树，单个看是效果不怎么好的，但是把同样的数据投入进去，把两个结果加起来考虑，就会增加可信度。

4.1.9. *神经网络*. Neural Networks 适合一个 input 可能落入至少两个类别里：NN 由若干层神经元，和它们之间的联系组成。第一层是 input 层，最后一层是 output 层。在 hidden 层和 output 层都有自己的 classifier。

input 输入到网络中，被激活，计算的分数被传递到下一层，激活后面的神经层，最后 output 层的节点上的分数代表属于各类的分数，下图例子得到分类结果为 class 1；同样的 input 被传输到不同的节点上，之所以会得到不同的结果是因为各自节点有不同的 weights 和 bias，这也就是 forward propagation。

4.1.10. 马尔可夫. Markov Chains 由 state（状态）和 transitions（转移）组成。例子，根据这一句话‘the quick brown fox jumps over the lazy dog’，要得到 markov chains。

步骤，先给每一个单词设定成一个状态，然后计算状态间转换的概率。

这是一句话计算出来的概率，当你用大量文本去做统计的时候，会得到更大的状态转移矩阵，例如 the 后面可以连接的单词，及相应的概率。

4.2. **神经网络模型.** 在深度学习十分火热的今天，不时会涌现出各种新型的人工神经网络，想要实时了解这些新型神经网络的架构还真是不容易。光是知道各式各样的神经网络模型缩写（如：DCIGN、BiLSTM、DCGAN……还有哪些？），就已经让人招架不住了。

因此，这里整理出一份清单来梳理所有这些架构。其中大部分是人工神经网络，也有一些完全不同的怪物。尽管所有这些架构都各不相同、功能独特，当我在画它们的节点图时……其中潜在的关系开始逐渐清晰起来。

把这些架构做成节点图，会存在一个问题：它无法展示神经网络架构内部的工作原理。举例来说，变分自编码机（VAE：variational autoencoders）看起来跟自编码机（AE：autoencoders）差不多，但它们的训练过程却大不相同。训练后的模型在使用场景上差别更大：VAE 是生成器，通过插入噪音数据来获取新样本；而 AE 仅仅是把他们所收到的任何信息作为输入，映射到“记忆中”最相似的训练样本上。

在介绍不同模型的神经元和神经细胞层之间的连接方式前，我们一步一步来，先来了解不同的神经元节点内部是如何工作的。

4.2.1. *神经元.* 对不同类型的神经元标记不同的颜色，可以更好地在各种网络架构之间进行区分。但是，这些神经元的工作方式却是大同小异。在下图的基本神经元结构后面，你会看到详细的讲解：

基本的人工神经网络神经元（basic neural network cell）相当简单，这种简单的类型可以在常规的前馈人工神经网络架构里面找到。这种神经元与其它神经元之间的连接具有权重，也就是说，它可以和前一层神经网络层中的所有神经元有连接。

每一个连接都有各自的权重，通常情况下是一些随机值（关于如何对人工神经网络的权重进行初始化是一个非常重要的话题，这将会直接影响到之后的训练过程，以及最终整个模型的性能）。这个权重可以是负值，正值，非常小，或者非常大，也可以是零。和这个神经元连接的所有神经元的值都会乘以各自对应的权重。然后，把这些值都求和。

在这个基础上，会额外加上一个 bias，它可以用来避免输出为零的情况，并且能够加速某些操作，这让解决某个问题所需要的神经元数量也有所减少。这个 bias 也是一个数字，有些时候是一个常量（经常是-1 或者 1），有些时候会有所变化。这个总和最终被输入到一个激活函数，这个激活函数的输出最终就成为这个神经元的输出。

4.2.2. *卷积神经元.* 和前馈神经元非常相似，除了它们只跟前一神经细胞层的部分神经元有连接。因为它们不是和某些神经元随机连接的，而是与特定范围内的神经元相连接，通常用来保存空间信息。这让它们对于那些拥有大量局部信息，比如图像数据、语音数据（但多数情况下是图像数据），会非常实用。

4.2.3. *解卷积神经元.* 恰好相反：它们是通过跟下一神经细胞层的连接来解码空间信息。这两种神经元都有很多副本，它们都是独立训练的；每个副本都有自己的权重，但连接方式却完全

相同。可以认为，这些副本是被放在了具备相同结构的不同的神经网络中。这两种神经元本质上都是一般意义上的神经元，但是，它们的使用方式却不同。

4.2.4. *池化神经元和插值神经元.* 经常和卷积神经元结合起来使用。它们不是真正意义上的神经元，只能进行一些简单的操作。

池化神经元接受到来自其它神经元的输出过后，决定哪些值可以通过，哪些值不能通过。在图像领域，可以理解成是把一个图像缩小了（在查看图片的时候，一般软件都有一个放大、缩小的功能；这里的图像缩小，就相当于软件上的缩小图像；也就是说我们能看到图像的内容更加少了；在这个池化的过程当中，图像的大小也会相应地减少）。这样，你就再也不能看到所有的像素了，池化函数会知道什么像素该保留，什么像素该舍弃。

插值神经元恰好是相反的操作：它们获取一些信息，然后映射出更多的信息。额外的信息都是按照某种方式制造出来的，这就好像在一张小分辨率的图片上面进行放大。插值神经元不仅仅是池化神经元的反向操作，而且，它们也是很常见，因为它们运行非常快，同时，实现起来也很简单。池化神经元和插值神经元之间的关系，就像卷积神经元和解卷积神经元之间的关系。

4.2.5. *均值神经元和标准方差神经元* (*Mean and standard deviation cells*) *(作为概率神经元它们总是成对的出现).* 是一类用来描述数据概率分布的神经元。均值就是所有值的平均值，而标准方差描述的是这些数据偏离（两个方向）均值有多远。比如：一个用于图像处理的概率神经元可以包含一些信息，比如：在某个特定的像素里面有多少红色。举个例来说，均值可能是 0.5，同时标准方差是 0.2。当要从这些概率神经元取样的时候，你可以把这些值输入到一个高斯随机数生成器，这样就会生成一些分布在 0.4 和 0.6 之间的值；值离 0.5 越远，对应生成的概率也就越小。它们一般和前一神经元层或者下一神经元层是全连接，而且，它们没有偏差 (bias)。

4.2.6. *循环神经元.* 不仅仅在神经细胞层之间有连接，而且在时间轴上也有相应的连接。每一个神经元内部都会保存它先前的值。它们跟一般的神经元一样更新，但是，具有额外的权重：与当前神经元之前值之间的权重，还有大多数情况下，与同一神经细胞层各个神经元之间的权重。当前值和存储的先前值之间权重的工作机制，与非永久性存储器（比如 RAM）的工作机制很相似，继承了两个性质：

第一，维持一个特定的状态；第二：如果不对其持续进行更新（输入），这个状态就会消失。

由于先前的值是通过激活函数得到的，而在每一次的更新时，都会把这个值和其它权重一起输入到激活函数，因此，信息会不断地流失。实际上，信息的保存率非常的低，以至于仅仅四次或者五次迭代更新过后，几乎之前所有的信息都会流失掉。