线性代数应该这样讲（一）

原创 夕小瑶 夕小瑶的卖萌屋 2017-05-01

前言

小夕为什么要讲线性代数呢?因为有人已经做了机器学习一段时间了，竟然认为矩阵就是用来存储数据的。小夕表示非常震惊。

而深刻透彻的理解核函数、PCA、LSI、谱聚类等以空间映射为理论核心的机器学习理论时，靠大学里教的那一套线性代数，很有可能是悲剧的（小夕所在的本科学校，数学专业全国top5，还是大牛的老师教的线性代数，然而小夕依然觉得学了假线性代数，心塞）

因此，在对国内高等教育不信任的基础上，小夕还是决定自己写文章讲点线代，以免让大家对以后小夕写的更高深的机器学习知识产生瓶颈式理解困难（虽然小夕的数学很渣很渣）。

但是注意，阅读本系列文章，**最好已经有线性代数的基础了**，比如最起码要知道矩阵看起来是什么样子，矩阵相乘是怎么乘的之类的。然后如果你虽然知道它们看起来的样子，却并不知道它们是什么意思，那么相信本系列文章会让您有所收获嗒。

注意！以下讲解线性代数的时候，小夕有时会建模到很多机器学习问题中，但是完全依赖线性代数解决机器学习问题的话，解决方案注定是极其简化的模型！它可以帮助你理解线性代数，但是其本身的工程价值很小！但是它还是后续很多高端算法的基础甚至原型！所以在本系列文章里不要太纠结工程上的可行性或者实用性啦。

为了提高大家的空间想象能力、思维抽象能力，小夕在这篇文章中尽量不画图（不要拆穿真实的原因是小夕比较懒\(//∇//)\）

好啦，啰嗦了好多，正式开始！

所谓矩阵

想象一下，你操纵着两个空间，一个是无边界的三维实数空间A，一个是无边界的二维实数空间B。然后你想建立一个桥梁，将这两个空间连接起来。这个桥梁可以将空间A的任意的点映射到空间B的某个点上。那么这个桥梁该如何表示呢？

没错！就是矩阵！**矩阵就是映射！**

举个最最简单的栗子。

空间A的任何一个点a一定是3个维度吧，设为a={a1,a2,a3}。空间B的任何一个点一定是2个维度吧，设为b={b1,b2}。那么如何将空间A的a点映射成空间B的某点b呢？

只需要建造一个叫做矩阵的桥梁，这个桥梁只需要满足维度是3*2就可以了，也就是3行2列。比如这个矩阵是$C=\begin{bmatrix} c11 & c12 \\ c21 & c22 \\ c31 & c32 \end{bmatrix}$。那么作为1*3维度的行向量a，乘以矩阵C后就成了1*2维度的行向量，也就是B空间的一个点（忘了矩阵乘法的见下文或者Google一下）。而不同的3*2矩阵就对应着不同的映射规则，会将A空间的同一个点a映射到B空间的不同位置。但是它们都是连通A与B的桥梁。

何为矩阵运算

有人学线代的时候问，矩阵乘法怎么这么麻烦！矩阵的逆运算就更加麻烦到爆了啊！然而他们有什么用呢？老师让我们学会做这些运算干什么？

这都是有原因的，只不过国内教材的悲剧就在于空讲知识不讲前因后果（莫非作者自己都不知道前因后果？）。跟着小夕一点点来~

下面是课本的讲解：

矩阵加法就是，对于一个矩阵$A=\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix}$和一个矩阵$B=\begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix}$，A+B就是$\begin{bmatrix} a11+b11 & a12+b12 \\ a21+b21 & a22+b22 \end{bmatrix}$

矩阵乘法就是，对于一个维度为i*j的矩阵A和一个维度为j*k的矩阵B，A*B就是将A中的每行分别点积B中的每列（点积就是将两个向量的每个位置的元素进行相乘后，再将每个位置的相乘结果累加的运算），因此得到一个维度为i*k的矩阵C。

矩阵的逆就是。。。手算太难了啊~假如我们已经算出来了矩阵A的逆是矩阵B，那么一定有A*B=I。其中I就是单位矩阵（对角线元素为1，其他元素为0的矩阵，更准确定义见下文）

然后是小夕的补充：

矩阵的加法就是映射规则的调整。矩阵A+矩阵B就可以看作，用B中各个元素的值去调整A这个映射规则！

矩阵的乘法是什么？就是代表着线性映射的叠加！比如，我们已知如何从X空间映射到Z空间了，也知道如何从Z空间映射到Y空间了。然而，恰好我们的机器学习样本就是在X空间！样本的类别恰好就在Y空间！怎么将样本映射成标签呢？

没错，X映射到Z（记为X->Z）的规则就是一个矩阵A，Z->Y就是一个矩阵B，所以若要X->Y，那么就是A*B的结果。这就是矩阵乘法。

矩阵逆运算是什么呢？就是映射的逆映射啊~比如在一个机器翻译问题中，源语言在X空间，目标语言在Y空间，我们通过训练知道了如何从X空间映射成Y空间了，就是通过训练得到的矩阵A。那么我们如何反向翻译呢？那就是求A的逆矩阵就可以了，完全无需再反向重新训练。

何为特征

它们为什么特殊

还有同学不知道我们为什么要单独的讨论一些特殊的矩阵，比如零矩阵，单位矩阵，对角矩阵等。

课本上学过了：

零矩阵就是所有元素都是0的矩阵，如$\begin{bmatrix}0 & 0\\0 & 0\\0 & 0\end{bmatrix}$；

单位矩阵就是对角元素都是1，其余元素都是0的方阵，如$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$；

对角矩阵就是**非**对角元素全为0的矩阵。如$\begin{bmatrix}3 & 0 & 0\\0 & -2 & 0\\0 & 0 & 8\end{bmatrix}$。

然后是小夕的补充：

零矩阵就是将任何空间映射成原点的矩阵。所以，某个空间的所有的点，一旦被吸入零矩阵的映射规则后，就像被吸进黑洞一样吸到原点。

注意，两个非零矩阵的线性映射叠加后（矩阵相乘后），完全可能变成零矩阵。也就是说变成一个黑洞，比如$\begin{bmatrix}3 & 0\\5 & 0\end{bmatrix}$乘以$\begin{bmatrix}0 & 0\\7 & 2\end{bmatrix}$后，就成了$\begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$，即一个黑洞（零矩阵）。

单位矩阵就是将空间映射成自己的矩阵！有同学疑惑，为什么单位矩阵不是所有元素都是1的矩阵呢？正是因为矩阵就是映射，单位矩阵的意义是将空间映射成自己，因此单位矩阵才是这个样子的。如果你让一个矩阵的所有的元素都是1，那么空间经过这个假单位矩阵的映射后，就不再是原来的空间了！（原来空间上的大部分点都不再位于它原来的位置了，也就是说被映射到了新的空间中，虽然新空间和旧空间的维数一样）

对角矩阵就是将原空间的各个坐标轴进行不同尺度的拉伸或挤压后生成新空间的矩阵！比如任何一个二维空间乘上一个对角矩阵$\begin{bmatrix}0.5 & 0\\0 & 2\end{bmatrix}$后，这个空间的第一个维度的坐标轴一定会缩短为原来的一半！第二个维度的坐标轴一定会膨胀成原来的两倍！所以比如这个二维空间中原来的点(2,2)经过这个对角矩阵的映射后，会变成(1,4)，看，是不是这样子！

为什么要分块？

还有分块矩阵，很多人也不理解有什么用。

课本上爱这样讲分块矩阵：

分块矩阵就是将一个大矩阵用横线和竖线分成好几块，这样就得到了好几个子矩阵。然后当大矩阵参与运算时，就可以让这些子矩阵分别参与运算。

小夕的补充：

分块矩阵对应的算法思想就是**分治**！一个很重要的应用就是**并行化矩阵运算，**这也是深度学习中离不开的底层运算。

设想，一个10000*10000维的超大矩阵再乘以一个10000*20000维的超大矩阵，如果让一个单核CPU去计算的话，那会累坏它的。我们知道，在深度学习任务中，由于参数非常多，数据维数也经常很高，因此大矩阵运算就是家常便饭。而我们还知道，深度学习任务喜欢用GPU去训练，为什么呢?一个很重要的原因就是GPU的“核”太多啦！经常几百几千，甚至几万核。因此，在深度学习中，通过将大矩阵分块，分成好多好多块，再将这些小块丢进GPU的成千上万的核里并行计算，那么这个大矩阵的运算瞬间就完成了！

如果没有大一你学到的矩阵分块这个姿势，那么让你写深度学习背后的科学计算包的话，你将一个大矩阵丢给CPU的一个核去计算那不得被上司骂死啦。

总结一下。矩阵的本质就是描述空间的映射，因此它就可以看作一个函数，接收一个空间作为输入，输出一个映射后的新空间。