知乎 首发于 计算主义

线性代数及其应用——嵌入向量

张学非

在深度学习广泛应用于自然语言处理和推荐系统的今天，人们会经常听说“嵌入向量”（Embedding Vector）。对于每一个实体（Item），比如物品、人、词等等，给它分配一个低维的向量（几十维或几百维）。这个向量称为这个实体的嵌入向量。为什么叫“嵌入”，后文会明了。

嵌入向量是实体的一种表示（Representation），也可以看做从实体中提取的一套特征。自然语言处理相关问题中的样本（句子，文章）是一堆词（或者 n-gram 之类 Token）。推荐系统的样本往往包含“人+物品”。样本中的词、人、物品等实体都可以先被转化成嵌入向量，再送交给网络的后续部分。

在这些领域的深度神经网络中，网络的前部往往先有一个“嵌入层”。最肤浅地看待嵌入层：它根据实体的 id 在一个大的嵌入向量表中检索出该实体的嵌入向量，输出给后续网络。训练时，嵌入向量表中的全体嵌入向量一并在反向传播+梯度下降过程中受到调整。训练完成后，所有实体的嵌入向量和整个网络的其他参数一样都固定下来。

嵌入向量只是网络参数的一部分。训练完成后，它们和网络其他参数一起构成了整个网络的“血肉”。网络往往是针对某一个具体问题而训练的，比如分类问题：“人”喜不喜欢这个“物”（是/否）。这其实是一个二分类问题。序列生成（比如翻译、自动回答）其实是一连串分类问题，也就是语言模型 —— 选择下一个词。

针对不同问题，嵌入层后连接不同结构的网络，计算并优化不同的损失（Loss）。那么嵌入向量当然也是朝着有利于这个具体问题的方向而被训练的。但是如果网络比较庞大，比较深，嵌入层就只是位于大网络前部的初始部分。可以认为网络的后部更多地承担了应对具体问题的责任，而嵌入层（嵌入向量）与具体问题隔得较远，更多地承担了比较通用的实体表示和特征提取的功能。特别是如果使用非监督学习，网络学习的不是某个具体的分类/回归目标，而是学习样本实体本身之间的关系（比如词与词之间的关系），那么学得的嵌入向量就更加与具体问题无关，而只关乎于现实中这些实体本身的某种性质和联系。

这样训练而得的实体嵌入向量，可以保存下来服务于其他任务。比如面对某个具体分类问题时，可以把之前非监督地，或者对某个庞大而通用的问题训练得到的嵌入向量直接拿来作为实体的表示，输入给（传统）模型或网络。如果把嵌入层视作网络的前部，这其实就是迁移学习。

这种实体的表示向量为什么叫“嵌入”？这种方法为什么 work？嵌入向量具体如何训练，也就是如何计算损失对嵌入向量的梯度，这就是本文要从数学角度说明的问题。

$$\mathbf{x}=\begin{pmatrix} x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}$$

这组数中的每一个称作该向量的分量： x_1 是 $\mathbf{x}$ 的第一分量， x_2 是 $\mathbf{x}$ 的第一分量 ..... 共有 n 个分量， $\mathbf{x}$ 是 n 维向量。但是这“一列数”并不是向量的本质，它只是这个向量的一个表示。向量（Vector）本是一个抽象概念。它存在于抽象的向量空间（Vector Space）中。

向量空间是向量的集合（可以有穷，可以无穷，甚至不不可列）记做： $\mathbf{v}\in\mathcal{V}$ 。我们用小写粗体字母表示向量，用花体字母表示向量空间。顺便提一嘴，我们用小写斜体字母表示标量（数），用大写粗体字母表示矩阵。向量空间中的两个向量之间可以进行加法。我们用 $+$ 表示这种向量加法。加法必须满足下面几条性质：

1. 交换律： $\mathbf{v}+\mathbf{w}=\mathbf{w}+\mathbf{v}$ ；
2. 结合律： $\mathbf{v}+(\mathbf{w}+\mathbf{u})=(\mathbf{v}+\mathbf{w})+\mathbf{u}$ ；
3. 存在“零向量” $\mathbf{0}$ ，满足： $\mathbf{0}+\mathbf{v}=\mathbf{v}$ ；
4. 对于每个向量 $\mathbf{v}$ 存在“反向量” $-\mathbf{v}$ ，满足： $\mathbf{v}+(-\mathbf{v})=\mathbf{0}$ 。

仅根据这几条性质，容易证明零向量以及每个向量的反向量都是唯一的。 $\mathbf{v}+(-\mathbf{w})$ 可以简单记为 $\mathbf{v}-\mathbf{w}$ 。向量“减法”就是加上反向量。除了向量之间的加法，数（标量）和向量之间可以进行乘法（数乘）。数乘必须满足：

1. 结合律： $a(b\mathbf{v})=(ab)\mathbf{v}$ ；
2. 数乘对向量加法的分配率： $a(\mathbf{v}+\mathbf{w})=a\mathbf{v}+b\mathbf{w}$ ；
3. 数乘对数加法的分配率： $(a+b)\mathbf{v}=a\mathbf{v}+b\mathbf{v}$ ；
4. 数 1 乘任何向量都等于该项量本身： $1\mathbf{v}=\mathbf{v}$ 。

用这几条性质可以证明数 0 乘任何向量都等于零向量 $\mathbf{0}$ （粗体，表示向量）。因为： $\mathbf{v}=1\mathbf{v}=(0+1)\mathbf{v}=0\mathbf{v}+\mathbf{v}$ ，等号两边都加上 $\mathbf{v}$ 的反向量，就得到： $0\mathbf{v}=\mathbf{0}$ 。配备了满足上述八条要求的加法和数乘的向量集合就是向量空间。如果一个向量 $\mathbf{u}$ 满足：

$$\mathbf{u}=a_1\mathbf{v}^1+a_2\mathbf{v}^2+\cdots+a_n\mathbf{v}^n=\sum_{i=1}^{n}a_i\mathbf{v}^i$$

则称向量 $\mathbf{u}$ 可以被这组向量 $\mathbf{v}^i\,(i=1,2,\ldots,n)$ 线性表出。注意，我们把系数的序号放在下标而不是上标，这在今后学习张量时有意义，现在可以不管。不论上标下标，就看做是序号即可。一组向量可以线性表出零向量（一定可以，最不济系数全为 0 即可）：

$$\mathbf{0}=a_1\mathbf{v}^1+a_2\mathbf{v}^2+\cdots+a_n\mathbf{v}^n=\sum_{i=1}^{n}a_i\mathbf{v}^i$$

如果满足上式的一组系数 $a_i\,(i=1,2,\ldots,n)$ 只能**全都为 0** ，非如此就不可能用这组向量线性表出零向量，则称这组向量是线性独立的（Linear Independent）。否则，如果可以找到一组不全为 0 的系数，用这组向量线性表出零向量，则称这组向量是线性相关的（Linear Dependent）。

如果一组线性独立的向量 $\mathbf{v}^i\,(i=1,2,\ldots,n)$ 能够线性表出向量空间中的所有向量，则称这组向量为该向量空间的一组基（Basis）。一组基中的向量称为基向量（Basis Vector）。一组基的基向量个数 n 称作该向量空间的维数。

）， 但是这另一组
ed）的了，是 n

间的维数。

选定一组基 $\mathbf{v}^i\,(i=1,2,\ldots,n)$ 。任意向量 $\mathbf{u}=\sum_{i=1}^{n}a_i\mathbf{v}^i$ 被这组基线性表出时候的系数 $a_i\,(i=1,2,\ldots,n)$ 是唯一的。因为假如存在另一组系数 $\mathbf{u}=\sum_{i=1}^{n}b_i\mathbf{v}^i$ ，则有：

$$\mathbf{0}=\mathbf{u}-\mathbf{u}=\sum_{i=1}^{n}(a_i-b_i)\,\mathbf{v}^i$$

因为基 $\mathbf{v}^i\,(i=1,2,\ldots,n)$ 是线性独立的，现在它们线性表出了零向量，那只可能那些系数都是 0 ，即 $a_i=b_i\,(i=1,2,\ldots,n)$ 。所以，任意向量被特定一组基线性表出的系数是唯一的。这组系数称为这个向量在这组基上的坐标（Coordinate）。在 n 维向量空间中可以用坐标表示一个向量。坐标就是 n 个数，而且有顺序。因为它们是向量在每个基向量上的系数，基中基向量的顺序就决定坐标的顺序。比如：

$$\mathbf{u}=\begin{pmatrix}a_1\\a_2\\a_3\\\vdots\\a_n\end{pmatrix}$$

这就是本文一开始的那个：“向量是一组有序数”的含义。但是向量 $\mathbf{u}$ 并不“固”在这一组坐标上。选择另外一组基 $\mathbf{u}$ 的坐标就变了。 $\mathbf{u}$ 是固定的，它的坐标随基不同而变。

现在我们来谈线性变换（Linear Transform）。有两个向量空间 $\mathcal{V}$ 和 $\mathcal{W}$ ，维数分别是 n 和 m 。一个线性变换 $\mathcal{T}$ 将 $\mathcal{V}$ 中的向量映射到 $\mathcal{W}$ 中的向量。比如：$\mathcal{T}(\mathbf{v})=\mathbf{u},\mathbf{v}\in\mathcal{V},\mathbf{u}\in\mathcal{W}$ ，而且要满足：

$$\mathcal{T}(a\mathbf{v}+b\mathbf{w})=a\mathcal{T}(\mathbf{v})+b\mathcal{T}(\mathbf{w})$$

这就是对线性变换的“线性”性要求。形象来说就是线性变换 $\mathcal{T}$ 可以“刺穿”数乘与加法。线性变换是从一个向量空间到另一个向量空间的映射，看上去好像很自由。但是“线性”性让它没那么自由了。选定一组基 $\mathbf{v}^i\,(i=1,2,\ldots,n)$ ，线性变换 $\mathcal{T}$ 在这组基上的行为就决定了它在所有向量上的表现。因为对于任意向量 $\mathbf{u}$ 来说：

$$\mathcal{T}(\mathbf{u})=\mathcal{T}\left(\sum_{i=1}^{n}a_i\mathbf{v}^i\right)=\sum_{i=1}^{n}a_i\mathcal{T}(\mathbf{v}^i)$$

$\mathcal{T}$ 将每一个基向量 $\mathbf{v}^i$ 映射成什么，就决定了它将任意向量 $\mathbf{u}$ 映射成什么。 $\mathcal{T}(\mathbf{u})$ 是 m 维（目标）向量空间 $\mathcal{W}$ 中的向量。给 $\mathcal{W}$ 选定一组基 $\mathbf{w}^j\,(j=1,2,\ldots,m)$ 。我们问：向量 $\mathcal{T}(\mathbf{u})$ 在这组基上的坐标是什么？先来看对于 $\mathcal{V}$ 的每一个基向量 $\mathbf{v}^i$ ， $\mathcal{T}(\mathbf{v}^i)$ 也都是 $\mathcal{W}$ 中的向量。它们在选定的那组 $\mathcal{W}$ 的基上的坐标是：

我们用 w 表示坐标。因为是 $\mathcal{T}\left(\mathbf{v}^i\right)$ 的坐标，下标先加一个 i ，又因为是在第 j 个基向量上的坐标，下标再加一个 j ，所以 w 就有了双重下标 $w_{j,i}$ 。现在看：

$$\mathcal{T}(\mathbf{u})=\sum_{i=1}^{n} a_i \mathcal{T}\left(\mathbf{v}^i\right)=\sum_{i=1}^{n} a_i\left(\sum_{j=1}^{m} w_{j,i} \mathbf{w}^j\right)=\sum_{j=1}^{m}\left(\sum_{i=1}^{n} a_i w_{j,i}\right) \mathbf{w}^j$$

上式表明， $\mathcal{T}(\mathbf{u})$ 在 $\mathcal{W}$ 空间的基 $\mathbf{w}^j(j=1,2,\ldots,m)$ 上的坐标是 $\sum_{i=1}^{n} w_{j,i} a_i \ (j=1,2,\ldots,m)$ 。这告诉了我们如何用 $\mathbf{u}$ 的坐标计算它在线性变换 $\mathcal{T}$ 下的“像” $\mathcal{T}(\mathbf{u})$ 的坐标（前提是在两个向量空间中选定各自的基）。现在我们把 $w_{j,i}\left(j=1,\ldots,m \text{ 和 } i=1,\ldots,n\right)$ 列成一个“阵列”：

$$\mathbf{T}=\begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix}$$

$\mathbf{T}$ 是一个 m 行 n 列的矩阵（Matrix）。它的第 j 行第 i 列元素 $w_{j,i}$ 是元空间 $\mathcal{V}$ 的第 i 个基向量 $\mathbf{v}^i$ 的像 $\mathcal{T}\left(\mathbf{v}^i\right)$ 在目标空间 $\mathcal{W}$ 的第 j 个基向量 $\mathbf{w}^j$ 上的坐标。线性变换 $\mathcal{T}$ 的所有行为都蕴含在矩阵 $\mathbf{T}$ 中。矩阵 $\mathbf{T}$ 是线性变换 $\mathcal{T}$ 在这两组基下的“表示”。正如“一列有序数”是向量在一组基下的表示一样。

于是我们可以定义矩阵与向量的乘法。我们要求：线性变换的矩阵乘以某向量（的表示）等于该向量在目标空间上的“像”（的表示）：

$$\mathbf{T}\mathbf{u}=\begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix}\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}=\begin{pmatrix} \sum_{i=1}^{n} w_{1,i} a_i \\ \sum_{i=1}^{n} w_{2,i} a_i \\ \vdots \\ \sum_{i=1}^{n} w_{n,i} a_i \end{pmatrix}$$

结论：（作为一列有序数的）向量是向量在一组特定基上的坐标（表示），矩阵是一个线性变换在两组特定基上的表示。我们回顾了基础线性代数知识，现在让我们回到嵌入向量的话题上来。

以下我们不再区分抽象的向量和“一列有序数”的向量，统称为向量。在你的问题中，有许多实体（词、人、物/商品），比如 10000 个。有什么好办法把它们表示为向量么？最开始你能采用的最好的办法就是 One-Hot 编码：将 10000 个实体排序，用 10000 维向量表示每一个实体，表示第 i 号实体的向量的第 i 个分量是 1 ，其余分量是 0 ：

$$\begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \end{pmatrix}$$

想假设实体与实体之间有任何关系。是的，你知道所谓“国王-女王与男-女”，但是你不可能手工（Handcraft）给 10000 种实体都人肉分配相互之间距离和角度都合适的向量。一开始，我们只能要求表示实体的向量中两两之间一致有相同的距离和夹角。那么只有在 10000 维空间中为每一个实体分配一个坐标轴上的单位向量，这正是 One-Hot 编码。

用 One-Hot 编码了向量，接下来就要把它送给模型了。我们头一件需要做的事就是降低维度——10000 维太高了。最简单的降低维度的方法就是用一个线性变换将 10000 维 One-Hot 向量映射到一个低维空间，比如100 维。以这个线性变换作为网络最头部的部分，接着再把得到的低维（100 维）向量输入给后面的其他组件。这样的线性变换就是：

$$\mathbf{h}_{100} = \mathbf{W}_{100\times 10000}\mathbf{o}_{10000}$$

$\mathbf{W}_{100\times 10000}$ 就是这个线性变换的矩阵，它的形状是 100×10000 。 $\mathbf{o}_{10000}$ 是 10000 维 One-Hot 向量，表示一个实体。 $\mathbf{h}_{100}$ 是变换得到的 100 维向量，是网络后续部分的输入。以下我们省略形状下标。根据矩阵与向量的乘法， $\mathbf{h}$ 的第 j 分量是：

$$h_j = \sum_{i=1}^{10000} w_{j,k}o_k$$

$w_{j,k}$ 是 $\mathbf{W}$ 的第 j 行第 k 列元素。 o_k 是 $\mathbf{o}$ 的第 k 分量。而 $\mathbf{o}$ 是 One-Hot 向量，对于第 i 号实体来说，只有 o_i 是 1 ，其余分量都是 0 。于是：

$$h_j = w_{j,i}$$

这说明 $\mathbf{h}$ 就是 $\mathbf{W}$ 的第 i 个**列**。也就是说，对第 i 号物品（的 One-Hot 向量）做线性变换，等价于取出该线性变换的矩阵的第 i 个列。这个列就是第 i 号实体的“嵌入向量”（Embedding Vector）。它把物品的高维表示（One-Hot）嵌入到了低维空间。线性变换就是“嵌入层”，它本质上就是线性变换层。这个线性变换的矩阵称作“嵌入矩阵”，它的所有元素也属于模型参数，在反向传播+梯度下降过程中一并训练。训练完成后，我们就得到了所有实体的嵌入向量（嵌入矩阵的所有列）。

回顾那个 i 从 1 加到 10000 的加式，那其中大部分是“乘 0 再累加”，算了个寂寞。但是 CPU/GPU 不管你乘的是零还是不是零，碳排放是一样的。于是当输入向量是 One-Hot 时可以这样实现嵌入层：就用物品的编号去索引嵌入矩阵的列就可以了。如果嵌入矩阵在存储中是按列存储的，那么一个偏移寻址就找到嵌入向量了。损失函数对嵌入矩阵各分量的偏导数的计算，本质上与损失函数对线性变换的矩阵的元素的计算一样。

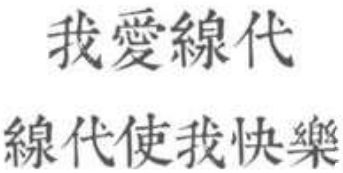
编辑于 2022-03-03 22:18

线性代数 嵌入 深度学习（Deep Learning）

知乎 首发于 **计算主义**

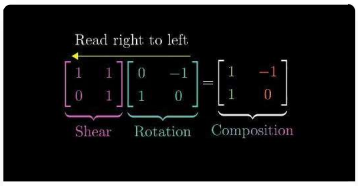
计算主义
"我不能建造者，我则没有真正理解"

推荐阅读



MIT线性代数课程精细笔记[第十一课]

忆臻 发表于机器学习算...



机器学习中常用的矩阵论知识

郑思座

14. MIT线性代数---正交向量与向量子空间

上一节回顾了一下之前学习的内容，这一节我们主要讨论四个子空间的性质以及正交向量的特点。如下图所示：在四个基本子空间中，对于秩为 r 的 m \times n 矩阵，其行空间（ dim C(A^T)=r ）...

August

还没有评论

写下你的评论...