深度学习
入门课程

袁振 [1]
山西大学

Contents

[1] 我的主页

1 从线性回归理解神经网络

深度学习是人工智能领域的一个分支，是一种以神经网络为架构的算法。现在，人们对深度学习非常狂热，因为认为它容易上手，觉得只需要写几行代码而没有像机器学习那样的数学公式推导。然而，这样的认识是肤浅的。如果你想对深度学习有更深层次的理解，想在该领域深耕，理论分析是不可避免的。

本课程是深度学习的入门课程，课程只需要一些微积分和线性代数的基础。虽然是入门课程，但并不意味着课程仅仅只是概念性的介绍。课程试图用最简单的语言对深度学习中的一些问题进行由浅入深的分析。当然，有数学推导的地方往往会有痛苦的存在。不过，借用Arthur Mattuck的一句话："推导数学公式，对于有些人来说就像在吃肉一样，而对于有些人来说就像在啃骨头一样，嚼下去，你会吃到肉的。"。

1.1 背景

如果我们想拥有一个智能医生，它可以通过对我们进行CT扫描，然后告诉我们身体的哪些部位有病无病。那么，我们如何利用人工智能技术来得到这样的智能医生呢？我们可以先收集很多患者和健康人的CT图像，然后把这些图像给电脑看，并训练它，告诉它哪些是病人和哪些是健康人的CT图像。我们通过这样的方式尝试让电脑找到诊断的标准。以上的过程我们称为训练阶段。该阶段包含了：数据、模型（电脑）、评判标准和训练四大要素。

当然，一个完整的过程还需要包含测试阶段。以上述为例，我们还需要从街上随机找一些志愿者（病人和健康人），然后让电脑对这些人进行诊断以验证其有效性。

本节课只讨论训练阶段，为了便于理解，我们以利用线性回归预测房价为例展开分析：如何构建模型、评判模型和训练模型。

1.2 数据

既然大家这么在乎房子，那就以预测房价为例吧。如果我们可以拿到全国n套房子的信息，即每套房子的属性和其对应价格：

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$$

其中$x^{(i)} \in R^p$是第i套房子的属性，例如，$x_1^{(i)}$ =城市、$x_2^{(i)}$ =地段、$x_3^{(i)}$ =面积等等，$y^{(i)} \in R$是房价。我们想利用这些现有的信息去训练一个房价预测模型。

1.3 模型

接下来，我们会构建一个房价预测模型，我们将以最简单的线性回归模型$h(x)$为例，因为神经网络可以看作是多个线性回归模型的堆叠：$f_{\text{神经网络}}(x) = \sigma(\ldots\sigma(h(x)))$。对于新开的楼盘，这个模型都可以通过房子的属性预测房价：

$$\hat{y} = h(x) = w^T\phi(x) + b$$

其中，$\phi(x)$是对x的一个变换，目的是使得预测更容易，例如，如果x中某些属性很重要，那么经过$\phi(x)$变换后，那些属性会变得更加突出，例如城市属性。本节课重点不在$\phi(x)$，不必太过注意。

1.4 评判标准

有了数据和模型，我们还需要建立一个评判标准来衡量这个模型预测结果的好坏，我们把评判标准定义为损失函数：

$$\begin{aligned} J(w) &= \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2n}\sum_{i=1}^{n}(w^T\phi(x^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2n}(\Phi w - Y)^T(\Phi w - Y) \end{aligned}$$

非常直观地，损失函数就是通过比较预测价格和实际价格的差别来判定模型的好坏。其中，$\Phi = \begin{bmatrix} - & \phi(x^{(1)}) & - \\ & \vdots & \\ - & \phi(x^{(n)}) & - \end{bmatrix}$，$Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$。

最后，我们只需要求解模型的参数w和b就可以了。求解的意思就是我们需要让模型预测的房价和实际房价越接近越好，这意味着我们需要把损失函数降到最低。我们把让模型损失函数为最低的模型参数称为最优解，用w_*和b_*表示。这样，对于一个新开的楼盘，我们便可以直接预测了：$\hat{y}^{(n+1)} = w_*^T\phi(x^{(n+1)}) + b_*$。

1.5 训练模型：直接求解法

注意到该模型是一个二次函数，我们可以直接求解。因为上式对于w来说是一个二次项，我们可以令$\frac{\partial J(w)}{\partial w} = 0$直接求解上式：

$$\frac{\partial J(w)}{\partial w} = \Phi^T(\Phi w - Y)$$
$$\Rightarrow \Phi^T\Phi w = \Phi^T Y$$
$$w = (\Phi^T\Phi)^{-1}\Phi^T Y$$

这里，我提供更加直观的一种解法。为了让$J(w)$最小，就是要让$\|\Phi w - Y\|$最小，而$\|\Phi w - Y\| \geq 0$，所以：

$$\Phi w - Y = 0$$
$$\Rightarrow w = \Phi^{-1}Y$$

上式成立的条件是Φ^{-1}存在且Y存在于Φ的列向量所构成的空间。可是上述条件都不成立呢？ 1. 如果Φ^{-1}不存在，而Y存在于Φ的列向量所构成的空间，我们仍然可以求解上式。 2. 如果Φ^{-1}不存在，且Y存在于Φ的列向量所构成的空间，我们会将Y垂直投影到Φ的列向量所构成的空间（因为这样距离$\|\Phi w - Y\|$为最近）于是：

$$\Phi w - Y = \Delta w$$
$$\Rightarrow \Phi^T\Phi w = \Phi^T Y + \Phi^T\Delta w$$
$$w = (\Phi^T\Phi)^{-1}\Phi^T Y$$

其中Δw是Φ的零空间，即$\Phi^T\Delta w = 0$。

1.6 训练模型：梯度下降法

但是，在神经网络$f_{\text{神经网络}}(x) = \sigma(\ldots\sigma(h(x)))$中，模型并非如此简单，模型参数$w, b$也并非如此少。因此，直接求解不切实际，而是采用一个搜索算法找到最优解。例如，给定一个初始值$w^{(0)}$，我们会根据一个规则来搜索w和b使得$J(w)$越来越小。

梯度下降算法就是最常用的规则，即：

$$w^{(k+1)} = w^{(k)} + \alpha\frac{\partial J(w)}{\partial w}$$

其中，α是步长，我们把$\nabla_w J(w) = \frac{\partial J(w)}{\partial w}$称为梯度。为什么是按梯度的方向进行搜索呢？因为梯度方向就是函数变化最快的方向。我们通过这样不断的搜索$w^{(0)} \to w^{(1)} \cdots \to w^{(k)}$，直到$w^{(k)}$无限接近$w_*$。

以函数

$$J(w) = w^2 - 2w + 1$$

为例，当给定初始$w^{(0)} = 4, \alpha = 0.1$时，此时损失为$J(4) = 9$。当我们沿着梯度的方向搜索$w$时，$\frac{\partial J(w)}{\partial w}|_{w=4} = 2w - 2 = 6$，则$w^{(1)} = 3.4$，此时损失为$J(3.6) = 6.76$。

1.6.1 梯度下降法更深刻的理解

我们把上式损失展开得：

$$J(w) = \frac{1}{2} w^T \Phi^T \Phi w - Y^T \Phi w + Y^T Y \qquad \text{(我们忽略}n\text{)}$$

我们令$A = \Phi^T \Phi, b = \Phi^T Y, c = Y^T Y$，则上式可以写为：

$$J(w) = \frac{1}{2} w^T A w - b^T w + c$$

注意到A是非常特殊的矩阵，A为对称矩阵，且为半正定矩阵，由此我们可以将A进行分解$A = QDQ^T$，其中Q为正交矩阵，D为对角矩阵，且$\min(D) \geq 0$（因为根据定义$x^T Ax \geq 0$）。

如果我们定义一个变换：

$$\bar{w} = T(w) = Q^T w$$

那么$J(w)$可以改写为：

$$J(\bar{w}) = \frac{1}{2} \bar{w}^T D \bar{w} - \bar{b}^T \bar{w} + c$$

此时：

$$\frac{\partial J(\bar{w})}{\partial \bar{w}} = D\bar{w} - \bar{b}$$

也就是说，我们可以单独地考虑$\bar{w}$中每个元素，分别地对它们进行梯度下降：

$$\bar{w}_j^{(k+1)} = \bar{w}_j^{(k)} - \alpha_j^{(k)} (d_j \bar{w}_j^{(k)} - \bar{b}_j)$$

其中，$d_j = D_{jj}$。我们还可以将上式改写为：

$$\bar{w}_j^{(k)} = \bar{w}_{*j} + (1 - \alpha d_j)^k (\bar{w}_j^{(0)} - \bar{w}_{*j})$$

由于A是半正定矩阵，即$d_j \geq 0$，我们分以下几种情况来讨论： 1. $d_j > 0$ 该情况下，$\bar{w}_j$一定有最优解，即$w_{*j} = -\frac{\bar{b}_j}{d_j}$ 2. $0 < \alpha d_j < 2$ 因为$|1 - \alpha d_j| < 1$，所以$\bar{w}_j^{(k+1)}$最终一定会收敛，即无限趋近于$\bar{w}_{*j}$。 3. $\alpha d_j = 2$ 因为$1 - \alpha d_j = -1$，所以，$\bar{w}_j^{(k+1)} = \pm \bar{w}_j^{(0)}$，故不会收敛。 4. $\alpha d_j > 2$ 因为$|1 - \alpha d_j| > 1$，所以$\bar{w}_j^{(k+1)}$会离$\bar{w}_{*j}$越来越远，故不会收敛。 5. $d_j = 0, \bar{b}_j \neq 0$ 由此我们可以得到$\bar{w}_j^{(k+1)} = \bar{w}_j^{(0)} - \alpha k \bar{b}_j$，即$\bar{w}_j^{(k+1)}$不会收敛。 6. $d_j = 0, \bar{b}_j = 0$ 此时，$\bar{w}_j^{(k+1)} = \bar{w}_j^{(0)}$

1.6.2 梯度下降法的收敛速度

我们通过探讨$d_j > 0$的情况来考察梯度下降法的收敛速度，我们可以取足够小的α来保证$0 < \alpha d_j < 2$。可是，我们究竟需要搜索多长时间才能到达$\bar{w}_{*j}$呢？

从上式我们看到，较大的d_j对应的$\bar{w}_j^{(k+1)}$，其收敛速度非常快。所以，梯度下降法的整体收敛速度取决于最慢的$\bar{w}_j$，也即对应具有最小的d_j。

对于矩阵A和其对应的正交分解QDQ^T，我们记$\delta_{min}, \delta_{max}$分别为$D$中最大和最小的非零元素，则矩阵$A$的条件数为：$\kappa = \frac{\delta_{max}}{\delta_{min}}$。如果我们的搜索步长为：$\alpha = \frac{1}{\delta_{max}}$，则：

$$
\begin{aligned}
\bar{w}^{(k+1)} &= \bar{w}^{(k)} - \alpha(D\bar{w}^{(k)} - \bar{b}) \\
\bar{w}^{(k+1)} - \bar{w}_* &= \bar{w}^{(k)} - \alpha(D\bar{w}^{(k)} - \bar{b}) - (\bar{w}_* - \alpha(D\bar{w}_* - \bar{b})) \\
&= (I - \alpha D)(\bar{w}^{(k)} - \bar{w}_*) \\
&= (I - \alpha D)^{(k+1)}(\bar{w}^{(0)} - \bar{w}_*) \\
\Rightarrow \|\bar{w}^{(k+1)} - \bar{w}_*\| &\leq \|(I - \alpha D)\|^{(k+1)} \|\bar{w}^{(0)} - \bar{w}_*\| \\
&\leq \|(I - \alpha\delta_{min})\|^{(k+1)} \|\bar{w}^{(0)} - \bar{w}_*\| \\
&\leq \exp(-\frac{k+1}{\kappa}) \|\bar{w}^{(0)} - \bar{w}_*\|
\end{aligned}
$$

所以，我们希望κ越小越好，即趋近于1。

1.7 正则化

回到损失函数，在实际过程中，我们常常会加入正则化项，即：

$$
\begin{aligned}
J(\bar{w}) &= \frac{1}{2}\bar{w}^T D\bar{w} - \bar{b}^T\bar{w} + c + \frac{\lambda}{2}\|\bar{w}\|^2 \\
&= \frac{1}{2}\bar{w}^T(D + \lambda)\bar{w} - \bar{b}^T\bar{w} + c
\end{aligned}
$$

其中$\lambda \geq 0$。若$\lambda > 0$那么$d_j > 0,\ \forall j$也即我们在讨论梯度下降收敛速度的情况。

1.8 总结

本节课通过一个回归模型介绍了深度学习的核心内容：构建模型、评判模型和训练模型。对于任何一个深度学习模型，我们都可以将其看作是线性回归模型对其进行分析。例如，多层感知器就是多层线性回归模型的叠加。

现在，大家觉得深度学习容易上手是因为人们现存大量的模型可供替换线性回归模型，例如将线性回归替换成卷积，那么就成了现在火热的卷积神经网

络。而实现这个过程仅仅只需要简单的几行代码。所以这样造成一个错觉，深度学习就是更换零部件的工作。

从本节课可以看出，我们的重点都在讨论训练模型过程的问题，也即优化模型，而这些往往被忽视。如果你想深度学习领域的专家，理解上述的各个过程都是不可或缺的。举个例子，通常，为了让模型训练的更快，我们常常会把数据进行归一化处理。每个人都知道归一化处理是一个必要过程，但是很少有人探究为什么要归一化处理。我们下一课会讲到这个内容。死记硬背只会让成为机器的操作工，无法去创造机器的新功能。只有知其然和知其所以然才会有创新。