深度学习

作　　者: 罗裕辉
时　　间: 2024年上

目录

一、概论

1.1 深度学习路线

如下：

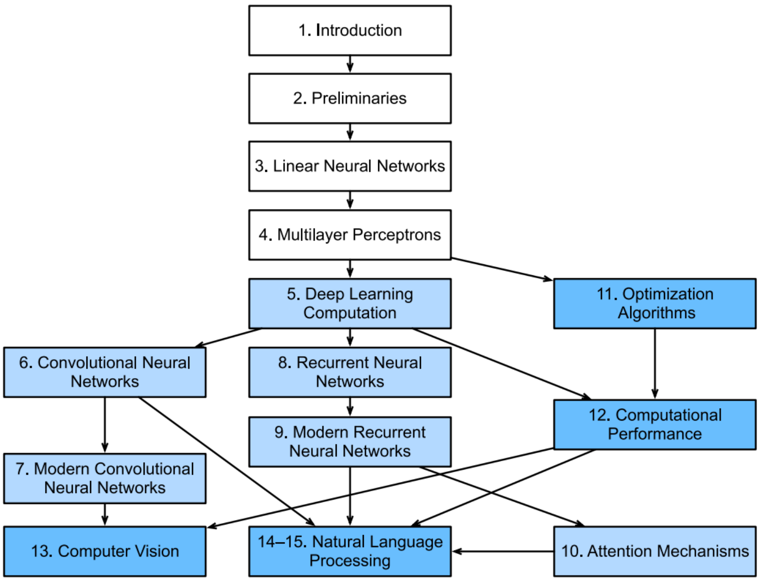

图 **1**

1.2 引言

传统的计算机程序是由软件开发人员从 first principles 开始编写的，这需要设计合适的业务逻辑，涉及应用程序和数据库的交互。

然而，对于某些问题，如下：

- 写一个程序预测明天的天气
- 写一个程序，接受一个问题，然后给出正确答案
- 写一个程序完成人脸识别。
- 写一个程序向用户推荐他们喜欢的产品。

这些问题的可能会随时间变化，内在关系也可能比较复杂，传统的程序不足以解决。这就需要 machine learning，它可以从经验中学习，性能表现不断提高，尤其是 deep learning，在 computer vision，natural language processing，speech recognition 等领域大放异彩。

1.3 机器学习的关键组件

机器学习的过程大概可以分为以下三步：

1. define a function set，or model family

2. define the loss function
3. pick the best funtion

机器学习中也有四个关键组件，包括 data、model、objective funtion、algorithm。

1.3.1 Data

通常来说，每个数据集由一个个样本组成，而样本由一组特征组成。

如果该样本的特征数量固定，我们可以说它是一个长度固定的向量。当然，很多情况下，样本的特征是变化的。

此外，数据的多少，也十分影响 deep learning 的训练效果，通常 deep learning 需要的数据是十分多的，否则未必比得上传统机器学习方法。

1.3.2 Models

深度学习主要关注功能强大的模型，这些模型由神经网络错综复杂地交织在一起，包含层层数据转换。

1.3.3 Objective Functions

我们会需要目标函数，以便评测我们 model 的好坏。并且通常来说，objective function 是越低越好的，所以它也叫 loss function(通常 squared error loss function)。

此外，由于在 training data 上表现好，不一定在 unseen data 上表现好，因此，我们通常还会将原始数据划分成 training dataset 和 test dataset

1.3.4 Optimization Algorithms

还需要有优化算法，它就是寻找到使得 loss function 最小的参数，通常是用 gradient descent 实现，也就是朝着使 loss 下降的方向更新参数。

1.4 机器学习的分类

1.4.1 supervised learning

监督学习就是 data 带标签。

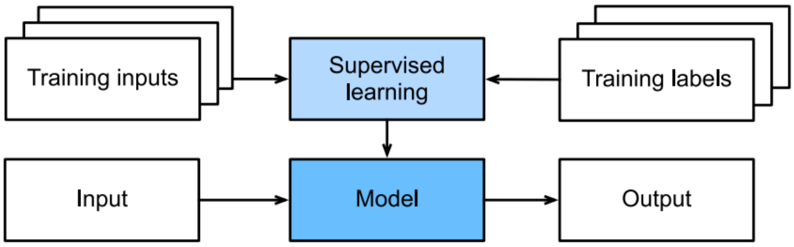

Fig. 1.3.1: Supervised learning.

图 2

监督学习包括以下几类：

- regression：针对 how many，loss function 通常是 squared error。
- classification: 针对 which one，并且通常是输出概率。loss function 通常是 cross-entropy。包括二分类、多分类和层次分类。
- tagging: tagging 就是贴标签，而且通常是多个标签，也就是 multi-label classification.
- search: 主要关注的是搜索结果的排序
- recommender systems：推荐系统关注对特定用户的个性化。
- sequence learning：输入序列和输出序列长度不固定，并且输入序列之间可能有关系。
- deep generative models. 能估计数据的密度。

1.4.2 semi-supervised learning

半监督学习就是标签数据远小于无标签数据。通常需要作出一些假设后进行训练。比如低密度分离假设、平滑假设、基于熵的正则化等。

1.4.3 unsupervised and self-supervised learning

无监督学习下数据不带标签，包括以下几类。

- clustering。聚类
- principal component analysis。分析数据的特性，主成分分析。
- generative adversarial networks. 生成对抗网络。

1.4.4 reinforcement learning

强化学习涉及智能体在一系列时间步骤中和环境交互。它会从环境中接收观察，选择一个动作，然后通过某种机制传输回环境，最后从环境中获得奖励，重复这个流程多次。强化学习目标是找到一个好的策略，能够选择好的动作。

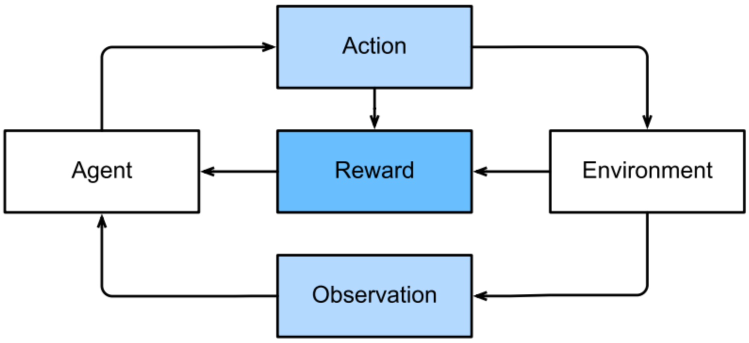

图 **3**

强化学习可以解决很多监督学习不能解决的问题。但是强化学习需要明确哪些动作可以导致好的 reward。

二、 预备知识

学习 deep learning 需要一些前置知识，包括线性代数、微积分、概率论与数理统计和信息论等。这里先复习部分，后面再慢慢补充完善。

2.1 Data manipulation 数据操作

首先我们需要知道如何存储和处理数据。通常我们使用的是 tensor，也就是它有点类似 numpy 的 ndarray(n-dimensional array)，但是 tensor 有几个有点，它可以用 GPU 加速以及支持自动微分等，deep learning 中我们经常使用它。

一个 tensor 就是 n 维数组，一维叫 vector，二维叫 matrix，更高维没有特殊的名字。

1. 0-d:0 维是一个标量，可以看作一个类比。
2. 1-d:1 维是一个向量，可以看作一个特征向量。
3. 2-d:2 维是一个矩阵，可以看作是一个特征矩阵，代表一个样本。
4. 3-d:3 维，例子是 RGB 图片 (宽度 x 高度 x 通道)。
5. 4-d:4 维，一个 RGB 图片批量 (批量大小 x 宽度 x 高度 x 通道)。
6. 5-d:5 维，一个视频批量 (批量大小 x 时间 x 宽度 x 高度 x 通道)。

- tensor 之间的二元运算会变成 tensor 每个元素之间的二元运算，并且即使 size 不匹配，也存在广播机制。
- tensor 可以用索引访问和修改多个 axis，如 X[0:2, :]。
- 为了减少内存开销以及多个位置使用同一个 tensor，我们会想要 in place 操作 tensor，可以使用切片，这样不会分配新的内存。

2.2 Data Preprocessing 数据预处理

实际问题中，我们首先要对原始数据进行预处理，将其转换成 tensor，常用的数据分析的包是 pandas。

2.3 Linear Algebra 线性代数

向量的范数可以理解为是向量的大小，包括 L2 范数和 L1 范数。

2.4 Calculus 微积分

2.4.1 梯度

参考下图 (注意标量只是代码它是一维，可能是多元的)，梯度实际上就是偏导，梯度指向值变化最大的方向，这里是分子布局法。

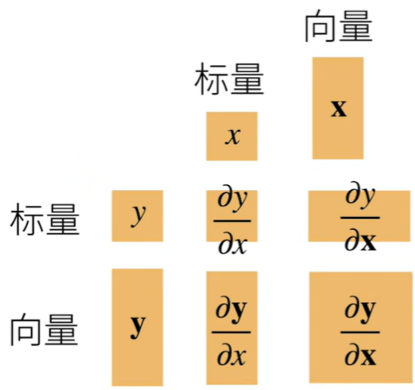

图 4

特别的，当 y 是一个向量，x 是一个向量，结果是一个矩阵

$$\partial \mathbf{y} / \partial \mathbf{x}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial \mathbf{x}} \\ \frac{\partial y_2}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial y_m}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1}, \frac{\partial y_1}{\partial x_2}, \ldots, \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1}, \frac{\partial y_2}{\partial x_2}, \ldots, \frac{\partial y_2}{\partial x_n} \\ \vdots \\ \frac{\partial y_m}{\partial x_1}, \frac{\partial y_m}{\partial x_2}, \ldots, \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

图 **5**

2.4.2 自动求导

自动求导计算一个函数在指定值上的导数，可以通过计算图理解。计算图将代码分解成操作子，将计算表示成一个无环图

$$z = \left(\langle \mathbf{x}, \mathbf{w} \rangle - y\right)^2$$

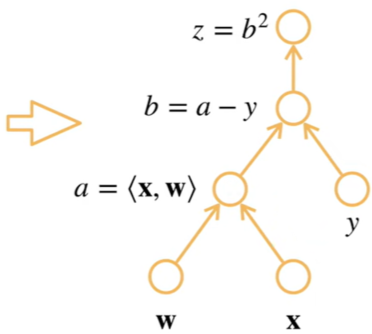

图 **6**

根据链式法则，计算梯度有两种方向：

自动求导的两种模式

- 链式法则：$\frac{\partial y}{\partial x}=\frac{\partial y}{\partial u_n}\frac{\partial u_n}{\partial u_{n-1}}\cdots\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial x}$
- 正向累积 $\frac{\partial y}{\partial x}=\frac{\partial y}{\partial u_n}\left(\frac{\partial u_n}{\partial u_{n-1}}\left(\cdots\left(\frac{\partial u_2}{\partial u_1}\frac{\partial u_1}{\partial x}\right)\right)\right)$
- 反向累积、又称反向传递

$$\frac{\partial y}{\partial x}=\left(\left(\left(\frac{\partial y}{\partial u_n}\frac{\partial u_n}{\partial u_{n-1}}\right)\cdots\right)\frac{\partial u_2}{\partial u_1}\right)\frac{\partial u_1}{\partial x}$$

图 **7**

反向累积

$z=\left(\langle\mathbf{x},\mathbf{w}\rangle-y\right)^2$

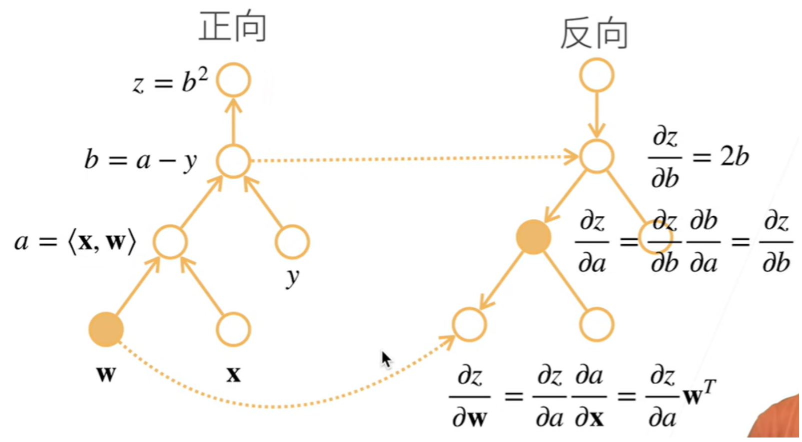

图 **8**

通常我们使用反向累积更新参数，虽然它需要计算图的中间结果，空间复杂度 O(n)，但是更新所有参数总体时间复杂度为 O(n)，牺牲空间换时间。

2.5 Probability and Mathematical Statistics 概率论与数理统计

2.6 Information theory 信息论

三、 线性回归

线性回归一般是深度学习遇到的第一类问题了，我们首先了解它，下面以购房为例进行介绍。

3.1 问题介绍

假设需要买一个房子，我们主要关心的是房子的价格，我们希望得到一个精准的房子预估价，而房子的价格与多种因素有关，比如卧室个数，卫生间个数和居住面积等，假设房子的价格与这些因素成线性关系，我们要训练一个模型，给定相关因素，能输出准确的预估价。

3.2 线性模型

3.3 模型建立

1. 输入：给定与房子有关的 n 个因素，即 $\mathbf{x} = [x_1, x_2, ..., x_n]^T$
2. 权重和偏差：n 个因素有 n 个权重，即 $\mathbf{w} = [w_1, w_2, ..., w_n]^T$，此外，还存在一个标量偏差 b。
3. 输出：模型输出是预估价，它是输入的加权和。$y = w_1x_1 + w_2x_2 + ... + w_nx_n + b$，即 $y = <\mathbf{w},\mathbf{x}> + b$。

事实上，线性模型可以看作是单层的神经网络 (有权重的层只有输入层)，它只有输入层和输出层，没有隐藏层。

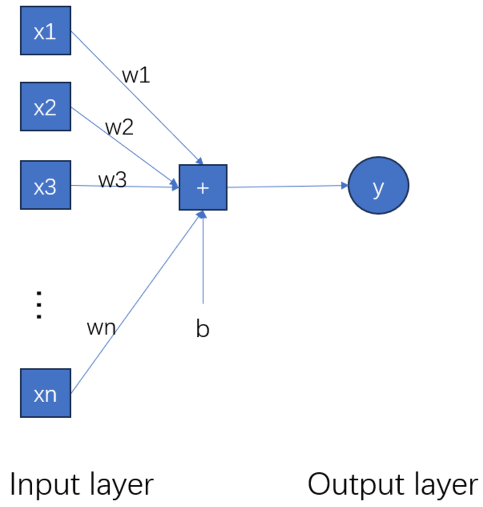

图 **9**

3.4 评估函数

我们的目标是根据已有的数据训练出模型，尽可能是模型的输出预估价 $\hat{y}$ 接近真实的购入价 y，可以定义损失函数 **(均方误差)**，n 为样本数：

$$l(y,\hat{y}) = \frac{1}{2n}\sum(y-\hat{y})^2$$

3.5 模型训练

我们首先需要收集过往的数据，它们包括各种因素以及最终的成交价，通常越多越好，假设有 n 个样本，即

$$\mathbf{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}]^T$$

$$\mathbf{y} = [y_1, y_2, ..., y_n]^T$$

其中，$\boldsymbol{X}$ 中的每个 $\boldsymbol{x_i}$ 经过转置后都是一个行向量，对应一个样本的不同因素，$\boldsymbol{y}$ 中同样 y_i 对应不同样本最后的成交价格。

我们训练的方法是利用训练数据最小化 loss function，并找到对应的参数 $\boldsymbol{w}$ 和 $\boldsymbol{b}$ 即：

$$\boldsymbol{w}*, \boldsymbol{b}* = arg\min_{w,b} l(y,\hat{y}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - <\boldsymbol{x_i},, \boldsymbol{w_i}> - b)^2$$

3.5.1 显式解——由凸函数性质直接给出最优解

由于我们的 loss function 是一个凸函数 (简单的说，就是两个点上的函数值连线的位置总是位于该函数图像的或之上)，凸函数的局部最优就是全局最优，根据数学原理，

我们知道当梯度为 0 时取最小值。

为了方便，将 b 纳入 w, 同时 X 加一列全 1，即 $\boldsymbol{X} \leftarrow [\boldsymbol{X}, \mathbf{1}], w \leftarrow \begin{bmatrix} w \\ b \end{bmatrix}$。

此时 $l = \frac{1}{2n}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||^2$，其中 || || 代表 L2 范数，于是

$$\frac{\partial l}{\partial \boldsymbol{w}} = -\frac{1}{n}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T\boldsymbol{X} = 0$$

$$\boldsymbol{w}* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

3.5.2 梯度下降

然而，通常来说深度学习的处理的问题不是凸问题，属于 NP-hard 问题，一般采用梯度下降法求解。

1. 选择一个参数的初始值 $\boldsymbol{w_0}$。
2. 选择一个学习率 η，**不能过大也不能过小**。
3. 重复迭代一定次数 $t = 1, 2, 3, ...$，沿着梯度方向的反方向更新参数。即：

$$\boldsymbol{w_t} = \boldsymbol{w_{t-1}} - \eta\frac{\partial l}{\partial \boldsymbol{w_{t-1}}}$$

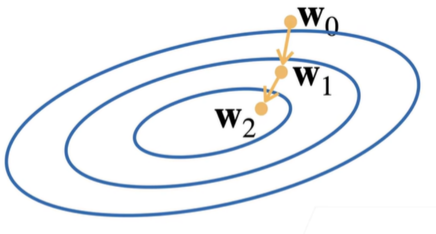

图 **10**

当然，考虑到训练数据一般很大，在整个训练集上计算梯度耗时太长，通常我们使用小批量梯度下降 (mini-batch)。

也就是随机从整个样本中采样 b 个样本，**不能过大也不能过小**，$I_b = i_1, i_2, ..., i_b$，计算这 b 个样本的损失，近似为整个样本的损失，即

$$l = \frac{1}{b}\sum_{I_b} l(y_i, \hat{y}_i) = \frac{1}{2b}\sum_{I_b}(y_i - <\boldsymbol{x_i}, , \boldsymbol{w_i}> -b)^2$$

3.6 代码实现

参考文献

[1]《动手学深度学习》