最后的退路：统计建模方法研讨

黎韬

中国人民大学

版本：0.0.1

更新：2021 年 8 月 19 日

ElegantLaTeX Program

感谢 ElegantLaTeX 提供的数学排版支持!

感谢我的女朋友，如果没有她，这本书的内容... 其实也不会变得更多的一些。但是有了她，不但本书的排版错误大大减少了，而且我也第一次感受到了幸福和快乐。

1 统计建模方法介绍

> There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown(Breiman, 2001).
>
> Leo Breiman

南京大学的周志华教授将机器学习定义为“致力于研究如何通过计算的手段，利用经验来改善系统自身的性能⋯⋯如果说计算机科学是研究关于算法的学问，那么类似的，机器学习是研究关于学习算法的学问 (周志华, 2016)。” Mitchell et al. (1997) 给了一个更形式化的定义：“假设用 P 来评估计算机程序在某任务类 T 上的性能，若一个程序通过利用经验 E 在 T 中任务上获得了性能 P 上改善，则我们就说关于 T 和 P，该程序对 E 进行了学习。”

根据这个定义，机器学习的范围很宽。她不仅包括了随机数据生成模型 (比如线性回归、逻辑回归、Cox 模型等)，还包括了最近热门的算法模型 (比如支持向量机、决策树、神经网络等)。Breiman (2001) 认为，随机数据生成模型 (下称 “数据模型”) 和算法模型在建模思路上是有区别的。数据模型会先假定数据生成过程，即目标变量是关于特征、噪声和误差的一个函数 ($Y = f(X, \epsilon, \beta)$，$f(\cdot)$形式给定)，比

如：

$$Y = X\beta + \epsilon,\ \epsilon \sim N(0, \sigma^2) \tag{1.1}$$

然后通过合适的参数估计方法去得到参数估计值及其标准误差。而算法建模不去假定数据生成过程，它直接认为目标变量是特征的函数：

$$Y = f(X),\ f(\cdot)\text{形式不给定} \tag{1.2}$$

我们可以通过一定的规则 (比如决策树、神经网络或支持向量机) 去学习得到 $f(\cdot)$。

1.1 人工智能、机器学习和深度学习

人工智能是指由人制造出来的机器所表现出来的智能。通常人工智能是指通过普通计算机程序来呈现人类智能的技术 (Wikipedia, 2021a)。深度学习是机器学习的分支，是一种以人工神经网络为架构，对数据进行表征学习的算法 (Wikipedia, 2021b)。

机器学习是介于人工智能和深度学习之间的概念。机器学习是一种自动化学习、不需要人工显式编程的人工智能。相对于那些需要显式编程的人工智能 (比如五子棋中指定不能让对手五子连结)，机器学习只要指定学习方法就可以让机器不断迭代进化，自动学习出规则。深度学习是机器学习的分支，它特指使用神经网络的方法 (如前馈神经网络 (NN)、卷积神经网络 (CNN)、循环神经网络 (RNN))，使

用海量的数据，针对特定的任务，为提高某项性能而学习。三者的关系总结如下：

- 人工智能
  - 机器学习
    - 算法模型
      - 深度学习
        - 卷积神经网络
        - 循环神经网络
        - 前馈神经网络
        - ⋮
      - 统计学习
        - 混合高斯模型
        - 贝叶斯网络
        - 支持向量机
        - ⋮
      - 符号学习
        - 随机森林
        - XGBoost
        - LightGBM
        - ⋮
    - 数据模型
      - 线性回归
      - **GLM** 回归
      - ARIMA
      - GARCH
      - ⋮
  - 规则编程
    - 五子棋程序
    - 魔塔游戏
    - ⋮

2 深度学习

2.1 循环神经网络

虽然前馈神经网络已经足够强大，但面临某些特定问题 (如图像识别与语音识别) 时，她也同样面临着参数爆炸的问题。为了解决这个问题，研究者们发明了循环神经网络 (Rumelhart et al., 1986)。正如卷积神经网络擅长处理超大型图片一样，循环神经网络擅长处理超长序列。

循环神经网络 (以下简称"RNN") 将权值共享的方法引入神经网络的建模当中。这种做法不仅大大降低了需要优化的参数量，使得参数校准更加精确，还赋予了神经网络接收不同长度的序列来训练的能力。以上两点优势是浅层的 (增大样本量就能解决)。更重要的是，因为权值共享，同一个词语对输出结果的影响，与其在句子中出现的位置无关。需要特别指出的是。RNN 并不能解决极性转移[1]问题。要想解决这个问题需要借助其它的辅助模型。

2.1.1 前向传导

假设一个样本为一个长句子，所有长句子最大长度为 τ(即最多有 τ 个单词数)，每个单词都使用维度为 d 的词向量去表示，那么所有样本的形状为 (None, d, τ)[2]。如果使用前馈神经网络去处理这个问题，假设有 N 个神经元，因为其只能接受一维的样本，所以需要将样本的形状展开，变成 $(\text{None}, d \times \tau)$。

[1]句子中的一些否定词、程度副词等的使用都可能会使得句子的极性发生偏转

[2]None 表示样本的数量不定

那么一个前馈神经网络的参数就为 $d \times \tau \times N + N$ 个[3]，前馈神经网络试图学习的函数 $g(\cdot)$ 可以表示为如下形式：

$$\boldsymbol{h} = g(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(\tau)}) \tag{2.1}$$

不同于前馈神经元，RNN 神经元可以直接接受形状为 (None, d, τ) 的样本。它的方法是在每一步 $t = 1, 2, ..., \tau$ 分别处理一个输入 $\boldsymbol{x}_t$ 和一个状态 $\boldsymbol{h}_{t-1}$

$$\begin{aligned} \boldsymbol{h}^{(t)} &= g(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(\tau)}) && (2.2) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}) && (2.3) \end{aligned}$$

通过这种变换，我们可以将学习 $g(\cdot)$ 的任务简化为学习 $f(\cdot)$，而 $f(\cdot)$ 中的参数 $\boldsymbol{\theta}$ 因为权值共享，其参数量已经大大减少了。

给定一个 $\boldsymbol{h}^{(0)}$，我们就可以计算出任意 $\boldsymbol{h}^{(t)}$，然后将 $\boldsymbol{h}^{(t)}$ 传入下一个输出层 (一般是全连接层) 中，就可以给出预测值。我们假设输出层中的神经元有 M 个。

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} && (2.4) \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) && (2.5) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} && (2.6) \\ \hat{\boldsymbol{y}}^{(t)} &= \text{softmax}(\boldsymbol{o}^{(t)}) && (2.7) \end{aligned}$$

[3] 其中有 N 个是偏置项

损失函数一般设置为交叉熵损失函数:

$$\begin{aligned} L\left(\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\right\}, \left\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\right\}\right) &= \sum_t L^{(t)} && (2.8) \\ &= -\sum_t \log p_{\text{model}}\left(\boldsymbol{y}^{(t)} \mid \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\right\}\right) && (2.9) \\ &= -\sum_{t=1}^{\tau} \sum_{i=1}^{M} y_i^{(t)} \log(\hat{y}_i^{(t)}) && (2.10) \end{aligned}$$

2.1.2 反向传播

RNN 中需要校准的参数有 $\boldsymbol{b}, \boldsymbol{W}, \boldsymbol{U}, \boldsymbol{c}, \boldsymbol{V}$:

$$\begin{aligned} \nabla_c L &= \sum_t (\nabla_{\boldsymbol{o}^{(t)}} L) \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} && (2.11) \\ \operatorname{vec}(\nabla_V L) &= \sum_t (\nabla_{\boldsymbol{o}^{(t)}} L) \frac{\partial \boldsymbol{o}^{(t)}}{\partial \operatorname{vec}(\boldsymbol{V})} && (2.12) \\ \nabla_b L &= \sum_t (\nabla_{h^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}} && (2.13) \\ \operatorname{vec}(\nabla_W L) &= \sum_t (\nabla_{h^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \operatorname{vec}(\boldsymbol{W})} && (2.14) \\ \operatorname{vec}(\nabla_U L) &= \sum_t (\nabla_{h^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \operatorname{vec}(\boldsymbol{U})} && (2.15) \end{aligned}$$

其中

$$(\nabla_{\boldsymbol{o}^{(t)}} L) = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o^{(t)}} = (\hat{y}^{(t)} - I_{\{y^{(t)}=\mathbf{1}\}})^T \tag{2.16}$$

注意 $\nabla_{\boldsymbol{o}^{(t)}} L$(以及之后的所有梯度) 都是行向量。推导出上述式子需要借助于:

1. $\partial L/\partial L^{(t)} = 1$
2. $L^{(t)} = \sum_i y_i^{(t)} \log(\hat{y}_i^{(t)}) = \log(\hat{y}_{i^*}^{(t)}),\ y_i^{(t)} = 1$ 如果 $i = i^*$
3. $y_{i^*}^{(t)} = \exp[o_{i^*}^{(t)}]/\sum_j \exp(o_j^{(t)})$
4. 矩阵求导使用分子布局的方式 (之后都使用该方式)

因为第 $\boldsymbol{h}^{(\tau)}$ 只会决定 $\boldsymbol{o}^{(\tau)}$，而不会决定 $\boldsymbol{h}^{(\tau+1)}$，所以:

$$\nabla_{\boldsymbol{h}^{(\tau)}} L = (\nabla_{o^{(\tau)}} L) \frac{\partial \boldsymbol{o}^{(\tau)}}{\partial \boldsymbol{h}^{(\tau)}} = (\nabla_{o^{(\tau)}} L) V \tag{2.17}$$

对于 $t = 1, 2, ..., (\tau - 1)$，我们知道 $\boldsymbol{h}^{(t)}$ 不仅可以决定 $\boldsymbol{o}^{(t)}$，还可以决定 $\boldsymbol{h}^{(t+1)}$，所以:

$$\nabla_{\boldsymbol{h}^{(t)}} L = (\nabla_{h^{(t+1)}} L) \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{a}^{(t+1)}} \frac{\partial \boldsymbol{a}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}} + (\nabla_{o^{(t)}} L) \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}} = (\nabla_{h^{(t+1)}} L) \mathrm{diag}[1 - (\boldsymbol{h}^{t+1})^2] W + (\nabla_{o^{(t)}} L) V \tag{2.18}$$

$$\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} = I_M \tag{2.19}$$

$$\frac{\partial \boldsymbol{o}^{(t)}}{\partial \mathrm{vec}(\boldsymbol{V})} = (\boldsymbol{h}^{(t)})^T \otimes I_M \tag{2.20}$$

$$\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}} = \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{a}^{(t)}} \frac{\partial \boldsymbol{a}^{(t)}}{\partial \boldsymbol{b}} = \mathrm{diag}[1 - (\boldsymbol{h}^{(t)})^2] I_N \tag{2.21}$$

$$\frac{\partial \boldsymbol{h}^{(t)}}{\partial \mathrm{vec}(\boldsymbol{W})} = \mathrm{diag}[1-(\boldsymbol{h}^{(t)})^2](\boldsymbol{h}^{(t-1)})^T \otimes I_N \tag{2.22}$$

$$\frac{\partial \boldsymbol{h}^{(t)}}{\partial \mathrm{vec}(\boldsymbol{U})} = \mathrm{diag}[1-(\boldsymbol{h}^{(t)})^2](\boldsymbol{x}^{(t)})^T \otimes I_N \tag{2.23}$$

参考文献

WIKIPEDIA, 2021a. 人工智能 — Wikipedia, The Free Encyclopedia[EB/OL]. https://zh.wikipedia.org/wiki/%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD.

WIKIPEDIA, 2021b. 深度学习 — Wikipedia, The Free Encyclopedia[EB/OL]. https://zh.wikipedia.org/wiki/%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0.

周志华, 2016. 机器学习 [M]. 北京: 清华大学出版社: 1.

BREIMAN L, 2001. Statistical modeling: The two cultures (with comments and a rejoinder by the author)[J]. Statistical science, 16(3): 199-231.

MITCHELL T M, et al., 1997. Machine learning[J].

RUMELHART D E, HINTON G E, WILLIAMS R J, 1986. Learning representations by back-propagating errors[J]. nature, 323(6088): 533-536.