模式识别与机器学习 **30250293-0 (2022 春)**

第 14 章 (深度学习 2) 作业

1 越时反向传播 (BPTT, Back Propagation Through Time)

考虑一个用最后时间步的输出做分类 logit 的文本二分类问题, 网络前向的过程为

$$\begin{aligned}
\boldsymbol{h}_0 &= \mathbf{0} \in \mathbb{R}^n \\
\boldsymbol{x}_t &\in \mathbb{R}^n,\ t = 1, 2, \ldots, l \\
\boldsymbol{h}_t &= \mathrm{ReLU}(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t) \in \mathbb{R}^n,\ t = 1, 2, \ldots, l \\
y_{\mathrm{pred}} &= \mathrm{Sigmoid}(\boldsymbol{V}\boldsymbol{h}_l + d) \in \mathbb{R},
\end{aligned}$$

使用交叉熵作为分类损失。考虑 batchsize=1 的 SGD 进行网络训练，且样本的标签为 y_{gt} 时，损失函数具体为

$$\mathcal{L}(y_{\mathrm{pred}}, y_{\mathrm{gt}}) = -(y_{\mathrm{gt}} \log(y_{\mathrm{pred}}) + (1 - y_{\mathrm{gt}}) \log(1 - y_{\mathrm{pred}}))$$

求

$$\frac{\partial \mathcal{L}}{\partial y_{\mathrm{pred}}}, \frac{\partial \mathcal{L}}{\partial d}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{V}}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}}$$

2 Attention 的修改

单头 self-attention 处理长度为 L，embedding 维度为 d 的序列 $X \in \mathbb{R}^{L \times d}$ 的过程为

$$\begin{aligned}
Q = XW^Q, K = XW^K, V = XW^V \\
X_o = \mathrm{Softmax}(\frac{QK^T}{\sqrt{d}})V
\end{aligned}$$

现在小明认为用两组线性变换分别产生 query, key 计算相似度的流程是多余的，所以他将两个训练参数 W^Q, W^K，约简成了同一组，即添加了约束 $W^Q = W^K$，其他流程不变。但却发现添加了这样约束的的模型性能不佳。现在你需要帮他分析原因。

2.1 引导题: 高维空间内的独立同分布的两个随机向量的夹角的分布

设 $\boldsymbol{x}, \boldsymbol{y} \overset{\text{i.i.d}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$。用抽样的方法求不同 n 下 (至少取 $n = 1, 2, 5, 10, 100, 1000$)，随机变量 $\theta = \arccos(\frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2})$ 的概率密度函数，说明分布随 n 的变化趋势。或直接证明

$$p(\theta) = \frac{\Gamma(n/2)}{\sqrt{\pi}\Gamma((n-1)/2)} \sin^{n-2}\theta$$

2.2 说明 $W^Q = W^K$ 的模型性能不佳的原因

在模型初始化时，通常将权重用高斯或类高斯分布进行随机初始化。所以我们不妨认为训练起始时，计算得到的 $\boldsymbol{q}_i, \boldsymbol{k}_i$ 都采样自同一高斯分布，且不妨认为特征维度 n 较大。

请你考虑在没有和有 $W^Q = W^K$ 的约束时，$\boldsymbol{q}_i, \boldsymbol{k}_j, (1 \leq i, j \leq l)$ 的内积结果，然后推导出 $\alpha_{i,j}$ 的取值，最后说明小明的改进无法得到好的效果的原因。

3 waimai_10k Mini 分类

waimai_10k 是从外卖平台上采集的每单外卖的用户文字评论和其评级星数的数据集。助教对数据集进行了一定的精简和预处理，预划分好了训练集和测试集。文字评论部分做好了停用词过滤和分词，评级星数也被简化成了差评好评 (0, 1) 两个类别，并且对应地提供了精简的词表和词向量。如果你对这些内容感兴趣，可以参看附录A

建议以下网络训练过程都使用学习率为 3e-5 的 Adam，word embedding 的维度为 300，将提供的训练集按 8:2 划分为训练集和验证集。与上次作业类似，你可以选择从模板代码中修改填空，也可以自行编写程序，请保证代码的复用性以减轻自己的工作量。此外，本题的任务是较为简单的短文本分类任务，几种方案的最终性能可能差距不大。

3.1 词向量平均线性分类器

加载提供的预训练词向量表，对于每个输入句子，将其所有词的词向量平均，作为句子的表征项链 $\boldsymbol{r}$，然后将 $\boldsymbol{r}$ 送入一个线性层，配合上恰当的激活函数 (Sigmoid 或 Softmax)，即可得到预测结果。本题中我们只训练这个线性层 (即冻结/不更新 Embedding 层)

报告训练过程中验证集上正确率的变化情况和最后在测试集上的正确率。

3.2 RNN

本题中，依然使用提供的预训练词向量表，仍然冻结 Embedding 层，只是将朴素的平均思想更换为可训练的单向单层 RNN。RNN 的隐层维度同样取为 300，重复以上实验，对比两种方案，

3.3 其他尝试

在以下的备选方案中选择你感兴趣的方案，重复实验，对比结果。(多做也不加分)

1. 在词向量平均和 RNN 方案中，使用随机可训练的 Embedding 层，使用预训练词向量但不冻结 Embedding 层
2. 使用多层、双向 RNN，至少报告 1 层双向，2 层单向、2 层双向三种情况。
3. 使用解决长期遗忘的方案，如将所有隐层平均，门控 RNN 等，至少报告用单向 RNN 所有时间步隐层做平均, 单向单层 GRU 最后一步输出做句向量, 单向单层 LSTM 最后一步输出做句向量三种情况。

A 关于文本数据预处理的杂谈

在本课程中，我们从课时和算法与课程关联度的角度考虑，不讲分词等文本预处理的方法。本节供对这部分内容有兴趣的同学参阅。

关于各种分词算法，感兴趣的同学可以参见相关教材和课程，这里可以给的推荐是 CMU CS 11-711 Advanced NLP 的相关章节[1]。分词、停用词过滤等文本预处理办法与所处理的语言、所面对的数据集、下游所使用的模型都有关系，比较完备地掌握这部分内容需要不少领域相关知识。

在编程接口方面，常见的深度学习框架基本都会内建一些文本预处理库。以 PyTorch 的文本支持库 torchtext 为例，进行说明。如果你是近些年才安装 PyTorch，你会发现较新的版本的官方默认安装命令都会连带安装 torchvision, torchaudio, 但却没有 torchtext。而自己手动安装的时候，如果你用的不是最新的 PyTorch stable 版本，你需要显式地指定和你 PyTorch 匹配的 torchtext 版本。大约在 0.9 版本之前 torchtext 管用的文本预处理接口是 torchtext.data.Feild 一站式地做分词，停用词过滤，转换为 token id 甚至转换词向量的工作；在近两年，0.9 到 0.11 版本发生接口的更替；在最新稳定的版本 0.12 正式弃用了这个接口，相关接口在 data, vocab, transforms 等子包。在搜索相关教程的时候注意识别版本。

此外还有一些三方库也可以完成这些功能。一些常用的分词预处理库比如 nltk, spacy, jieba。这些库的下载配置也不是那么直接，只用这些库的默认接口，也有它擅长/不擅长处理的语种的差异。这些都需要你比较详细地阅读相关文档，才能比较好地处理。

最后，课上已经提到过，attention based model 有较大的数据容量。反应到文本预处理上，就是它们不太会做停用词过滤，会用一些比较直接且细粒度的字词分割算法 (比如，Google 原版的英文 BERT 用的 WordPiece 方法，原版的中文 BERT 就是直接按字拆分)，与 RNN 上常用的分词算法和分词粒度都有较大差别。

[1]公开课件: http://www.phontron.com/class/anlp2021/schedule/wordseg-morphology.html, 视频 1：https://www.youtube.com/watch?v=WcyX07t_Me8 视频 2:https://www.bilibili.com/video/BV14L4y1x77G?p=12

[1]详见https://pypi.org/project/torchtext/