Naive Bayes Classifier 实验报告

作者： 陈熠豪

学号：2017011486

邮箱： chenyiha17@mails.tsinghua.edu.cn

代码实现

- 数据预处理：src/data_process.py
  - 将 index 文件转换为便于处理的数据表。

    数据表中共有两列：file、label，其中 file 为文件路径组合后的 6 位 id，label 为 0 或 1。

    转换好的数据表存为 label.csv 文件，便于后续直接载入使用。
  - 使用正则表达式，将邮件文件转换为结构化的 meta 数据和正文的单词集合。
    - meta 数据：邮件第一个分段（空行\n\n）前的数据为 meta 数据。先把跨行的 meta 数据合并，然后对每一个 meta 数据行进行切分，得到 key 和 value。最后得到结构化的 meta 数据结构体，每一个 key 对应一个 value 列表。将这个结构化的 meta 数据存为 json 文件（{id}.meta），便于后续载入使用。
    - 单词集合：邮件第一个分段后的数据为正文，对正文中的单词进行切分和清理，存为文本文件（{id}.txt），便于后续载入使用。
- 分类器：src/classifier.py
  - 数据划分：数据 shuffle 后，进行 n 折平均划分，依次选择一折作为测试集，其余作为训练集。实际实验时使用的是 5 折划分。
  - 训练：影响分类器效果的可能因素有以下几个
    - 算法：基于 Naive Bayes，我在具体的概率计算上考虑了一些修改，共有以下几个方法
      - naive_bayes_0：对于测试集中的某个单词 x_i，不考虑其在测试集中重复出现，即多个单词和 x_i 相同时，**条件概率只贡献一次**，而其贡献的概率 $P(x_i|y)$ 为“在 y 条件下 x_i **至少出现一次**的概率”
      - naive_bayes_1：对于测试集中的某个单词 x_i，考虑其在测试集中重复出现，即多个单词和 x_i 相同时，**条件概率贡献多次**，而其贡献的概率 $P(x_i|y)$ 为“在 y 条件下 x_i **至少出现一次**的概率”
      - naive_bayes_2：对于测试集中的某个单词 x_i，考虑其在测试集中重复出现 k 次，则**条件概率只贡献一次**，而贡献的概率 $P(x_i|y)$ 为“在 y 条件下 x_i **至少出现 k 次**的概率”
      - naive_bayes_3：对于测试集中的某个单词 x_i，考虑其在测试集中重复出现 k 次，则**条件概率只贡献一次**，而贡献的概率 $P(x_i|y)$ 为“在 y 条件下 x_i **恰好出现 k 次**的概

率”

- naive_bayes_4：对于测试集中的某个单词 x_i，不考虑其在测试集中重复出现，即多个单词和 x_i 相同时，**条件概率只贡献一次**，而其贡献的概率 $P(x_i|y)$ 为“在 y 条件下 x_i **平均出现的次数**”。此时已经不是在算概率了，而应考虑为权重计算

以上一些方法可能并不是很科学，但我认为有一定的探索价值，所以采用它们进行了对比实验。其中 naive_bayes_1 的方法对应于课程中给出的正确的 Naive Bayes 方法。需要注意的是，在计算概率时均使用对数加法以避免溢出。

- meta 数据：是否使用 meta 数据可能影响分类器性能，因此也分别训练，做了对比试验。

  不过 meta 数据最终还是作为计数，使用 naive bayes 方法计算概率的。实际上我觉得利用 meta 数据用 SVM 或 Decision Tree 可能会更好，但限于时间和精力，我没有进行实验。

- 训练比例：使用多少比例的训练数据可能影响分类器性能。分别以 0.05、0.5、1.0 的比例进行训练，做了对比实验

综合上述三个方面，所有的组合共有 30 种，我对这 30 种组合都进行了 5 折训练

- 测试和评价：在测试集中计算概率并进行分类，和 groundtruth 比对，并计算如下指标
  - accuracy = (tp + tn) / (tp + fp + fn + tn)
  - precision = tp / (tp + fp)
  - recall = tp / (tp + fn)
  - specificity = tn / (tn + fp)
  - prevalence = (tp + fn) / (tp + fp + fn + tn)
  - f1-score = 2 * precision * recall / (precision + recall)
  - train_time
  - test_time

  最终评价时，主要考虑 f1-score、precision、accuracy 和 train_time+test_time。在邮件分类问题下，precision 是一个重要的指标，因为垃圾邮件的误报代价更大。

- 分析和结果呈现：src/plot.py

  基于 f1-score 给出最好的 5 个组合，分析它们在各个指标下的结果，同时以结果最好的组合为 benchmark，给出该 benchmark 在某个方面改动的对比结果。绘制两个图像 comparison.png 和 best_5.png

实验结果分析

- 仓库中保存了一份完整的结果result/trec06p/result_03_27_20.csv，图像也是基于这份结果制成。复现该结果的方法请看 Readme
- 在所有的 30 个组合中，f1-score 排在前 5 的结果如下：

rank	model	train_ratio	use_meta
1	naive_bayes_2	1.0	false
2	naive_bayes_0	1.0	false
3	naive_bayes_4	1.0	false
4	naive_bayes_0	0.5	false
5	naive_bayes_2	0.5	false

它们具体的指标如下图所示。

可以看出它们都能很好地完成分类任务，f1-score 等指标都在 0.95 以上。

它们的每折训练时间与训练数据量有可见的线性相关性，因此训练过程是相对可控的。由于实现上的关系，分类器的概率计算是懒惰的，只在测试时计算，因此测试用时相比与训练用时会长很多。如果只是普通的 Naive Bayes，其实是可以先计算好概率的，但为了和其他模型保持代码上的统一，所以没有这么做。

从最好的五个组合中其实可以看出，naive_bayes_2 和 naive_bayes_0 的模型应该存在某些优势，在计算一个单词的概率贡献时，对其出现的次数做考虑，或许可以更准确地进行分类。

同时，更高的训练比例可以达到更好的结果，这一点毋庸置疑，从组合中也可以看出。

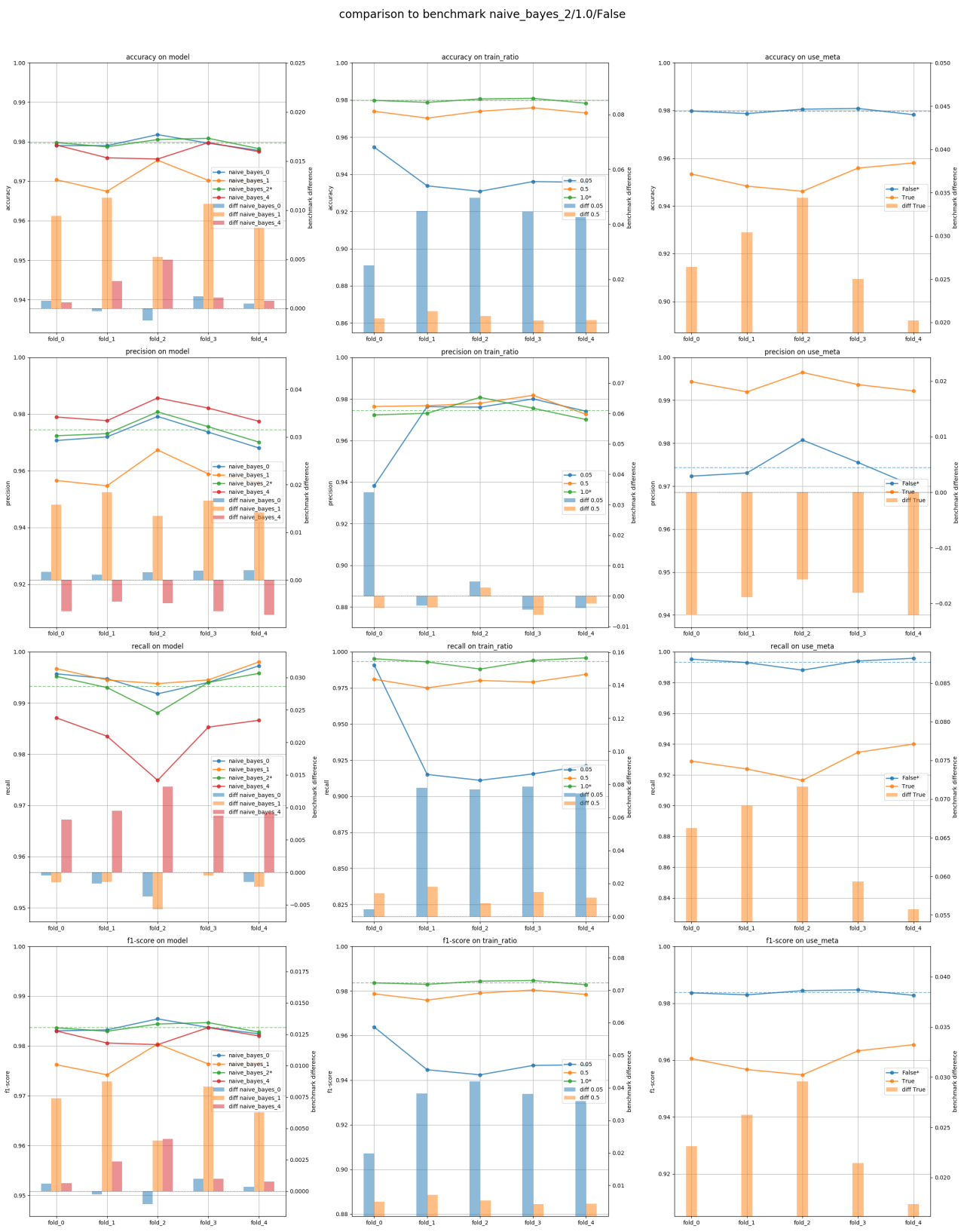
此外，在最好的 5 种组合中均未使用 meta 数据，我认为这可能和 meta 数据的处理方式有关，在我的实现下，meta 数据仅仅是根据 value 的个数进行计数，把计数向量当做“词频分布”来应用 Naive Bayes 的，这实际上比较别扭，也浪费了很多信息。其实可以用基于计数向量，用 SVM 或 Decision Tree 训练分类器，通过分类的 cluster 中不同 label 的占比来计算条件概率，从而和 Naive Bayes 结合在一起，或者单纯根据专家知识选择某些 meta 数据，对其信息进行筛选来做分类，可行的方案有很多，可作为本次实验的 future work。

meaures on best 5 combinations

在所有的组合中，f1-score 最高的组合为 (naive_bayes_2, 1.0, false)。以此为对照组（benchmark），分别改变 model、train_ratio 和 use_meta，进行了对比实验，结果如下图所示。其中折线图对应左侧的纵坐标，表示具体指标的大小，柱状图对应右侧的纵坐标，表示对照组相对于各实验组在指标上的差值（即 benchmark 的优势）。

由于 naive_bayes_3 的结果偏差较大，为了作图的美观，没有将该实验组的结果画出。naive_bayes_3 结果较差的原因和该算法的概率计算有关，因为是计算单词“恰好出现 k 次的概率”，考虑的过细，在这种规模的训练数据下可能无法正确地学习到分类能力。

从图中可以看出，使用 benchmark 的 model，相对于其他 model 可以有整体的提升，说明使用单词“至少出现 k 次”的计算方法来计算概率是有一定优势的。此外可以看出，较高的 train_ratio 可以在 accuracy、recall、f1-score 等指标上取得全面的优势，但在 precision 上，反倒有小幅度的下降，说明误报有所增多。最后，在是否使用 meta 数据上，可以看到，使用 meta 数据也是在 accuracy、recall、f1-score 上有提高，但在 precision 上有下降。从中可以得到的结论似乎是，随着数据信息量增加（train_ratio、use_meta），分类器的覆盖能力、泛化能力（recall）会增高，而正确做出预警的能力（precision）会有所下降。



讨论

- Issue 1：训练集的大小。前面已经叙述了训练集的选取方法，即通过 5 折交叉测试的方法得到完整的训练集，再通过 train_ratio 参数从中得到实际使用的训练集。在呈现结果时，使用 5 折交叉测试的平均 f1-score 作为排序指标，而在图像中则把各折的具体指标都画了。

- Issue 2：概率平滑处理。采用了实验指导 PPT 中提供的平滑方法，参数 alpha 设为 1，M 为 2。由于平滑处理主要是为了处理 0 概率的情况，参数选择实际对结果影响不大，所以没有对参数进行对比实验。
- Issue 3：特征选择。前面已经叙述了 meta 数据的处理方法，没有对具体的 feature 进行具体处理，是统一使用 key-value 对的计数向量作为 meta 数据的特征。
