SI 630：作业 1 - 分类

到期：见画布

1 导言

作业 1 将让你建立第一个 NLP 分类器：逻辑回归。Logistic Regres- sion 是一种简单但往往出奇有效的文本分类模型--你甚至可能在 sklearn 等常用的 Python 机器学习库中使用过 Logistic Regression！[1]在作业 1 中，您将*从头开始*构建 Logistic 回归，以了解这些模型的工作原理。你的实现将非常简单，但会让你更直观地了解机器学习模型是如何工作的，关键是深度学习模型是如何工作的。

作业 1 将让你在*两种情况下*实现 Logistic 回归。第一种，你将使用 `numpy` 实现逻辑回归的所有部分，包括随机梯度下降和更新步骤。在第二部分中，您将使用深度学习模型库 `pytorch` 实现相同的方法。在处理数值数据（如数字向量）时，pytorch 和 `numpy` 非常相似。[2]不过，`pytorch` 是为训练深度学习模型而设计的，可以自动完成大部分/全部更新步骤。我们让你使用不同的库实现两次逻辑回归，是为了让你更直观地了解 `pytorch` 在做什么（以及随机梯度下降是如何工作的！），同时也让你尽快掌握 `pytorch` 的使用方法，这在后面的作业中都会用到。你不需要先学会一种方法才能学会另一种方法；有些学生说 pytorch 部分的初始设置更简单，所以你可以选择更简单的方法。

由于 (1) 需要将概念和方程映射到 python 代码，(2) 需要学习如何使用两个新库，本作业的编程部分将具有一定的挑战性。实际上，逻辑回归的代码大约有 20-30 行（或更少），`numpy` 和 `pytorch` 模型共享一些用于简单 I/O 和设置数据的额外代码。最大的挑战在于如何将我们在课堂上讲到的数学概念落实到实际代码中，例如 "计算似然是什么意思？?"。要克服这一障碍，您需要挠头，还可能需要进行一些调试，但这都是值得的，因为

您将能够很好地理解较新算法的工作原理，甚至了解如何设计自己的算法！

作业中有方程式。即使你从高中起就没有上过数学课，也不应该害怕它们。方程的存在是为了提供精确的符号，只要有可能，我们都会尽量对它们进行更多的解释。如果您有需要，讲座材料和《语音与语言处理》教科书中也有很多说明，可以帮助您进一步解释这些内容。逻辑回归是一种*非常*常见的技术，因此您也可以在这里找到很好的解释。

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model。LogisticRegression.html

[2] https://rickwierenga.com/blog/machine%20learning/numpy-vs-pytorch-linalg.网页

如果需要，您可以上网查阅相关解释（例如，本视频中的可视化解释，可以让您了解偏置项的作用）。习惯使用方程式将有助于你今后的表达。如果您不确定某项内容的含义，请询问老师；从现在起到作业到期，我们有六个办公时间，另外还有一个活跃的 Piazza 讨论板。你并不是一个人在疑惑，有人可能会因为你的勇气而受益。

鉴于这些算法是 NLP 和许多其他领域的基本算法，您很可能会在网上找到逻辑回归的实现方法。虽然您可以查看它们（有时它们会提供很多信息！）、[3]您提交的所有作品都应是您自己的作品（参见第 9 节）。

最后，请记住这是一堂没有课业负担的课。如果您认为本作业的某些部分没有必要或过于费力，请告诉我们。我们很乐意提供更详细的解释，说明*为什么作业的*每一部分都是有用的。耗时过长的作业甚至可能是作业中的错误，我们将予以修正。

本作业的学习目标如下：

- 了解逻辑回归的工作原理、参数作用以及如何进行预测
- 如何使用稀疏数据结构
- 梯度下降和随机梯度下降在实践中如何发挥作用
- 了解如何使用 `numpy` 实现分类器，更广泛地说，了解如何使用新的 python 库
- 学习如何使用 `pytorch` 实现和训练（浅层）神经网络
- 提高软件开发和调试技能[4]

2 分类任务：居高临下

冒犯性语言是网络互动中的一个现实问题，NLP 开发了*许多*技术来识别仇恨言论等更明确形式的冒犯性语言。然而，有些形式的冒犯性语言是隐晦的。其中一种形式是居高临下，即一个人暗示另一个人在某种程度上低人一等。在作业 1 中，我们将使用来自 Perez-Almendros 等人（2022 年）的一个非常新的数据集，该数据集以针对弱势群体的居高临下标签为特征。按照现代 NLP 的标准，这个数据集的规模很小，只有 1 万个实例，但要在这个数据集上取得好成绩却很困难。该数据集也足够小，您可以轻松掌握，并快速训练您

的模型！

[3] 指导教师还想提醒大家，网上并非所有的实现都是正确的，也并非所有特定算法的实 现都是本作业所要求的。在前几个学期，有些学生在阅读了许多不同的算法描述/实现后，发现自己更加困惑，只有在自己尝试了这些材料后，才豁然开朗。如果您有任何问题，！

[4] Seriously！高级软件开发会让你发现各种奇怪的错误，这将使你的技术飞跃要弄清楚。请做好准备，但要知道，你在这里学到的技能将对你的职业生涯*大有*帮助。学习如何使用不熟悉的库调试代码并进行开发是本作业的一个关键目标！

3 作业设计说明

本作业分为三个部分。**第 1 部分**让您执行将文本转化为术语-文档矩阵的常用预处理步骤。您需要编写代码来读取数据、标记数据并将其转换为某种数字表示。第 **2 部分**让你使用 `numpy` 实现逻辑回归，包括随机梯度下降的所有细节。第 **3 部分**让你使用 `pytorch` 实现逻辑回归。在第 3 部分中，你还将使用开发数据做一些实验，看看调整某些超参数的效果如何。

符号说明：在整个赋值描述中，我们将把特征化文本数据称为 $x_1, \ldots, x_n \in X$，其中 x 是一个维度与特征数量相对应的向量，x_i 是该向量中第 i 个位置的值；X 是所有文档的矩阵集。刚开始时，每个特征将对应一个词的存在；不过，在后面的作业中，你可以自由尝试不同类型的特征，如表示两个连续词的大词，例如 "不好 "就是一个单一特征。我们将类别标签称为 $y_1, \ldots, y_n \in Y$，其中 y_i 是一个特定的类，Y 是所有类的集合。在我们的案例中，我们有 *一个二元*分类任务，所以实际上只有 y_1 和 y_2。

我们有时还会用 $exp(x_i)$ 来表示 e^{x_i}。当字体太小或使方程更难理解时，这种符号可以让我们避免使用超级脚本。

实现注意事项 除非另有说明，否则您的实现 *不应*使用任何现成的机器学习库或方法（例如，不使用 sklearn）。您将使用普通的 Python 以及 `numpy` 和 `pytorch` 等数值库来完成所有工作。

4 数据

作业 1 有三个相关文件（您可以在与该作业相关的 Kaggle 竞赛中找到它们）：

- `train.csv` 该文件包含用于训练分类器的评论、评论 ID 及其标签。标签列包含一个二进制标签，表示该消息是否居高临下（1 表示居高临下）；`tweet` 列包含 tweet 文本。
- `dev.csv` 该文件包含文本和标签，您将在部分作业中使用它们来评估您的系统。

- `test.csv` 该文件仅包含文本数据，您将使用训练有素的分类器进行预测，并将结果上传到 Kaggle。

作为本次作业的一部分，我们将使用 Kaggle InClass 报告对测试数据的预测。这项作业本身并不是 "排行榜竞赛"。相反，我们使用 Kaggle 是为了让你了解自己的预测结果与其他同学相比有多好。由于我们使用的是相同的数据，执行的是相同的算法，所以你的分数应该与其他同学比较接近。如果您决定参加可选部分，并进行一些特征

在工程设计方面，你们的分数可能会稍高一些，但没有人会因此而受到惩罚；我们知道大多数实施方案应该做得怎样。

我们为使用 PyTorch 的基本 Logistic 回归和 Logistic Re- gression 设置了 Kaggle 竞赛。您首先需要使用邀请链接注册，然后向比赛提交您的分数。

- **Numpy Kaggle InClass:** https://www.kaggle.com/t/e299a330a6c94091a14451818568a525
- **PyTorch Kaggle InClass:** https://www.kaggle.com/t/8febee6db6be4ee9a3137e1047ea2b08

5 第 1 部分：表示文本数据

在开始分类之前，我们需要将文本数据转化为分类器可以使用的数字表示。在课堂上，我们讨论过使用*词袋*表示法，即每个文档表示为长度为 |V|（所有文档中唯一单词的数量）的向量 x，我们计算每个单词在该文档中出现的次数，这样 x_i 就是*第 i* 列所代表的单词出现的次数。

第一部分分为两项任务：(1) 将文档标记为单词；(2) 将单词列表转化为向量。

5.1 任务 1.1：标记化

标记文本的方法有*很多很多*。在本作业中，我们将尝试使用两种方法，让您了解如何进行标记。标记化的方法很少有错，不过有些方法在标记质量和训练模型时的下游性能方面都会给你带来更好的结果。

- 编写一个名为 `tokenize` 的函数，该函数接收一个字符串并按空格进行标记，然后返回一个标记列表。在读取训练数据时应使用该函数，这样每个空格分隔的单词将被视为不同的特征（即不同的 x_i）。

请注意，由于标点符号的原因，我们的标记化工作可能做得不好。例如，"good "和 "good "被视为不同的特征，因为后者末尾有一个逗号，而 "Good "和 "good "也因为大写字母的不同而被视为不同的特征。我们是否希望这些都是不同的？此外，我们是否要将每个标记都列为特征？(提示：正则表达式能否帮助您过滤可能的特征？请注意，您必须自己

实现标记化；不能导入 NLTK/SpaCy 并封装它们的函数（尽管您可以从它们那里获得一些灵感）。

- 编写一个名为 `better tokenize` 的更好的标记化函数来解决这些问题。在你的报告中，描述你修复了哪类错误，包含了哪些功能，以及为什么做出这样的选择。

我们稍后在评估模型时会用到这些函数。

5.2 任务 1.2：构建术语文档矩阵

作为一个思想实验，考虑一下如果我们为训练数据创建一个术语-文档矩阵会发生什么。这个矩阵的大小为 $D \times V$。由于我们的词汇量可能会随着每篇新文档的增加而增大，因此这个矩阵将非常庞大，有可能达到数十或数百千兆字节，在个人电脑上根本放不下。然而，像 `sklearn` 这样的库却能正常工作，这是怎么回事呢？实际上，术语-文档矩阵*非常稀疏*；大多数文档并不包含大部分单词，因此我们实际上并不需要表示所有这些零！

在您的解决方案中，我们希望使用稀疏矩阵表示法。SciPy 库中有几种稀疏矩阵实现，我们可以使用，非常方便。[5]在构建术语-文档矩阵时，你需要创建其中一个矩阵。

创建矩阵时，您需要记录哪些维度对应哪些单词，这样当您看到一个新文档时，就知道如何创建一个新的向量，并为其单词填写正确的维度。

你应该在词汇 V 中包含多少单词？把所有的词都包括进去是很诱人的（毕竟这很容易）。不过，让我们来想想如果把所有词都包括进去会发生什么。首先，词频遵循齐普夫定律（Zipf's Law），这实际上意味着大多数词出现的频率非常低（大约 80% 的独特词只占我们看到的标记的 20% 以下）。因此，大多数独特词的出现频率都很低，很多词在我们的数据中只出现过一次。如果一个词出现的次数屈指可数，那么它对我们的分类就没有特别大的帮助，因为：(i) 这个词的出现并不能泛化到很多文档中；(ii) 包含这个词会增加我们需要学习的参数数量（可能对性能没有帮助）。因此，大多数文本分类系统都会规定词汇中包含该词的*最低词频*。

在您的解决方案中，您应该使用 2 的最小词频；每一个出现少于 2 次的词都*不会*包含在词汇表中（因此不会在 β 向量中得到一个维度）。欢迎您以后在实验中改变这个值，看看效果如何。

导入注意事项：在构建矩阵时，我们*最终*需要在 `numpy` 实现中考虑偏置项。通常，实践者只需在术语-文档矩阵中添加一列，其值始终为 1，然后乘以偏置系数。然而，`pytorch` 库会为我们处理偏置项（耶！），所以我们并不需要总是添加这一列。您需要设计矩阵构建代码，以便正确考虑偏置项。

一个提示是，在构建矩阵时，您可能会发现 python 中的 `set`、`defaultdict` 和 `Counter` 类非常有用。

第二个提示是尝试分解步骤并手动测试。如果您尝试只用两个文档来构建一个稀疏矩阵，结果会怎样？"猫吃东西 "和 "我喜欢猫"？稀疏矩阵会是什么样子？试着在代码中运行类似的简单案例，并验证输出结果是否符合您的预期--有时这比查看整个数据集更容易调试！

要获得满分，您需要编写能生成稀疏 SciPy 矩阵而非稠密矩阵的代码。

[5] https://cmdlinetips.com/2018/03/sparse-matrices-in-python-with-scipy/

6 第 2 部分：numpy 中的逻辑回归

在第 2 部分中，您将实施逻辑回归。

$$P(y = 1|x, \beta) = \frac{1}{1 + exp^{-\sum_{i=1}^{N} x_i \beta_i}}$$

请注意，我们的逻辑回归实施仅限于两个类别，我们将其表示为*二进制*，因此 y 要么是 0 要么是 1。[6]方便的是，您的训练数据已经为二元分类设置好了。

您的实现将是逻辑回归最简单的公式之一，您将使用随机梯度下降迭代找到更好的参数 β。作业设置的目的是让你了解如何计算梯度。

在任务 1 中，您将在不同的函数中实现逻辑回归分类器的每个步骤。作为其中的一部分，您还将编写代码来读入文本数据并进行标记化。这里的标记化指的是分离字符串中的单词。在下半部分中，您将重新审视标记化，询问是否能更好地确定哪些是单词。

- 执行一个名为 `sigmoid` 的函数，该函数实现了 sigmoid 函数 S

  $$S(X) = \frac{1}{1 + exp(-X)}$$

  .你的函数应该是*矢量化的*，这样它就能一次性计算整个数字矢量的 sigmoid。方便的是，`numpy` 通常会帮你做到这一点，因此如果你将一个数字乘以一个 `numpy` 数组，它就会乘以数组中的每一项（这同样适用于在数组上使用的函数（提示））。如果你不确定，请咨询我们！稍后你将需要使用 sigmoid 函数进行预测。

- 请注意，我们可以计算 likelihood，但由于我们将使用对数算术版本（因此称为 `log-likelihood`），以避免数值下溢，并且因为它的处理速度更快。请注意，我们**可以**计算整个训练数据的对数似然 ll，*即*

  $$ll = \sum_{i=1,\ldots,n} y_i \beta x^{\top}_i - log(1 + exp(\beta x_i))^{\top}$$

  其中，β 是所有系数的向量。要得出这个公式需要一些步骤，关于如何得出这个公式的细节将在讨论部分进行讨论。你可以跳过繁琐的数学推导，直接使用上面的公式来实现！不过，您需要实现*随机梯度下降*（SGD），即在计算从训练集中随机选

择（随机！）的*单个*项目的损失后更新权重。为获得满分，您的代码必须实现随机梯度下降，而*不是*梯度下降。

[6] 重要说明：SLP3 教科书详细介绍了多分类逻辑回归。这种多类设置更为复杂（虽然也更有用），因此我们不会在作业中实施。

- 在选择一定的 β 进行预测时，我们希望利用预测中的差值对数似然的梯度告诉我们哪个更新的方向（正或负）以及更新的幅度。编写一个函数 compute `gradient` 来计算梯度。请注意，如果我们使用的是梯度下降法，我们可以使用以下方法计算整个数据集的梯度

$$\nabla l = X^T (Y - \hat{Y})$$

请注意，Y 是一个二进制向量，包含我们的基本事实（即训练数据标签），$\hat{Y}$ 是的二进制向量。不过，我们将一次计算 *一个实例*的似然梯度，因此可以计算出如何更新 β *的*梯度为

$$\frac{\partial}{\partial \beta} ll(\beta) = (\sigma(\beta x_i) - y_i) x_i$$

这个等式会告诉我们二元预测结果 $\sigma(\beta x_i)$ 与基本事实（$y_i - \sigma(\beta x_i)$）的接近程度，然后使用损失来缩放当前实例 x *的*特征值$_i$，以确定如何更新 β。从本质上讲，这种更新会告诉我们如何更新 β，从而让它知道哪些特征的权重更高，以及如何对它们进行加权（或者反过来，忽略哪些特征！）。在进行学习时，我们将根据学习率 α *来调整*梯度，以确定每一步更新 β 的大小，如下所示

$$\beta_{new} = \beta_{old} - \alpha \delta \beta ll(\beta)$$

为了了解这种方法的原理，请想一想，如果我们对项目 i *的*预测结果 $\hat{y}_i$ 与地面实况 y_i 相同，梯度将等于多少；如果我们使用这种梯度来更新我们对 β_i 的权重，将会产生什么效果？

重要提示：SLP3 一书的第 5 章第 5.4 节还对逻辑回归和随机梯度下降进行了详细介绍，其中包括 5.4.3 更新步骤的详细示例。如果作业描述中有任何令人困惑的地方，请随时在 Piazza 上联系我们和/或查看 SLP3 资料！[7]

- 将所有内容整合在一起，编写一个逻辑回归函数，输入
  - 矩阵 X，其中每一行都是包含该实例特征的向量
  - 一个向量 Y，包含行的类
  - 学习率是一个参数，用于控制每一步改变 β 值的幅度

- `num_step` 停止前更新 β 的步数

您的函数应通过对 X 的每一行 i 进行预测（$\hat{y}_i$），在每一步迭代更新权重向量 β，然后使用这些预测来计算梯度。您还应该包括一个*偏置*系数。[8]

[7] 我们还想补充的是，尽管数学运算很花哨，但实际执行中大部分都是非常普通的 操作!

[8] 最简单的方法是在 X *中*添加一个值为 1 的额外特征（即列）；`numpy` 函数轴=1的 `np.ones` 和 `np.hstack` 将有所帮助。

请注意，使用矩阵运算可以让您的工作更轻松。例如，计算 $\hat{y}_i$ 时，将矩阵 X 的行乘以 β 向量。如果您不确定如何操作，不用担心！请在办公时间来找我们讨论！

- 编写一个预测函数，在给定新文本后，将其转换为一个向量（即类似于 X 中的一行），然后使用 β 向量预测类别并返回类别标签。
- 要了解模型的学习情况，首先在训练数据上训练模型 学习率=5e-5 (即一个很小的数字），并使用 `num steps = 1000`。**注意：**这意味着总共只对 1000 条信息进行训练；如果你的函数需要很长时间，你可能已经实现了完整的梯度下降，而这并不是要求的。在初始阶段，每 20 步绘制一次完整数据的对数似然。模型是否在某个时刻收敛（即对数似然是否保持稳定）？
- 现在你对超参数的工作原理有了一定的了解，可以训练模型直到收敛。这可能需要几个历元的时间，也可能在一个历元之前就很快收敛，这取决于你的学习速度（和数据集）。当每步之间的对数似然变化不大时，就可以停止训练。通常情况下，我们会使用一个很小的数字（例如 1e-5）来决定需要进行更多训练的变化程度，不过也欢迎您自行定义。
- 在训练数据上进行训练后，使用逻辑回归分类器在验证数据集上进行预测，并使用 F1 分数报告性能。
- 向 KaggleInClass 竞赛提交您的最佳模型对测试数据的预测，以便

`numpy` Logistic 回归

还有一些提示：

- 请确保您执行的不是完全梯度下降法，在这种方法中，您要计算所有项的梯度。随机梯度下降每次*只*使用*一个*实例。
- 如果您遇到内存和崩溃问题，请尝试确保您正在对*稀疏*矩阵进行乘法运算。稠密的术语-文档矩阵可能太大，内存装不下。

7 第 3 部分：使用 PyTorch 进行逻辑回归

现在，当你终于看到令人兴奋的 Py- Torch 部分时，你已经开始了漫长的深度学习之旅！ `PyTorch` 是一个在学术和工业领域都非常流行的深度学习框架，*也*是我们这次和今后所有 NLP 作业都要用到的库。有了这个框架，我们就能以简洁明了的方式构建具有灵活架构的深度学习模型。刚开始可能有点难度，但一旦掌握了方法，你就会惊奇地发现 PyTorch 是如此强大，只需几行代码就能构建复杂的模型。Parth 3 的目标是让你了解 PyTorch 如何训练模型的 "工作流程"（因为我们将在整个课程中使用这个框架），同时也为你的 `numpy` 实现提供一些参考比较点。

在第 3 部分中，您将使用 `PyTorch` 实现逻辑回归模型。您将比较你的 PyTorch 版本和你的 `numpy` 版本，看看其中一个是否比另一个性能更好。

如果你觉得在 numpy 部分卡住了，我们建议你先试试 pytorch 部分。

7.1 设置数据

PyTorch 可以使用张量。[9]张量是一个花哨的数学术语，表示具有任意维数的矩阵（例如，一个向量就是一个维数为 1 的张量！），这与 `numpy` 中的数组非常相似。*方便的是*，`pytorch` 还提供了稀疏张量能力、[10]我们需要用它来表示我们的术语文档矩阵张量（虽然它仍然是一个矩阵！）。

- 编写一个稀疏张量函数，获取您之前使用的术语-文档矩阵的稀疏 `numpy` (SciPy) 表示，并将其转换为稀疏张量对象。您将在后面的部分中使用张量格式的转换数据来训练模型。

7.2 构建逻辑回归神经网络

逻辑回归可被视为一种浅层神经网络（|V| 个输入神经元和 1 个输出神经元），因此我们将使用 PyTorch 构建神经网络的功能来帮助你开始本学期的学习。PyTorch 有一种特殊的方法来定义神经网络，这样我们就可以使用它非常方便的功能来训练它们。要创建一个新网络，你需要扩展 `nn.Module` 类、[11]该类是所有网络的基类。通过扩展该类，我们可以连接 PyTorch 的深度学习训练。特别是，该类将跟踪哪些参数在梯度下降过程中需要更新，如果这些参数被指定为该类的一个字段的话--这意味着我们不必像以前那样计算损失，*也*不必为随机梯度下降进行更新。

`nn.Module` 类定义了 `forward()` 函数，这是我们需要覆盖的最重要的函数。forward 函数从输入到输出（即从词包向量到预测方）。我们将指定输入，然后使用方法中的网络层产生输出。重要说明：由于神经网络是数学意义上**的**非线性函数，**PyTorch 已重载了函数运算符 () ，因此调用网络将调用 forward()** 。例如，代码 `model.forward(inputs)` 和 `model(inputs)` 是等价的！请注意，这一约定也适用于所有网络层；这些层接受输入并产生输出，因此我们可以像函数一样调用它们。

你可能还会猜到还有一个 backward() 函数，没错！这是一个从输出返回到输入的步骤，沿途更新权重。不过，我们在这里什么都不用做。这就是神奇之处：PyTorch 会为我们处理逆向更新，只要我们正确调用训练循环代码，它就会自动更新逻辑回归模型中的β 权重。

- 在本部分中，您将编写一个 `LogisticRegression` 类，它是 `nn.Module` 的扩展。想一想输入和输出是什么？需要训练的 权重形状是什么？逻辑回归有哪些层？

[9] https://en.wikipedia.org/wiki/Tensor
[10] https://pytorch.org/docs/stable/sparse.html
[11] 如果您以前没有看过类扩展，有很多很好的例子（如网上的这个例子；您的子类并不复杂，所以只要语法正确，不需要理解得太透彻。

7.3 设置培训

实施模型后，下一步就是训练网络。特别是，您需要定义损失函数和优化器，然后使用准备好的数据来训练模型。这些都是 pytorch 的特定对象，但我们已经看过这些概念的示例。*损失函数*告诉 pytorch 如何衡量我们的预测与正确输出的接近程度。在我们的设置中，我们有 *一个二元*预测任务，所以我们将使用二元交叉熵损失或 `BCELoss`[12].实际上，你已经在 `numpy` 中实现了这一点！Pytorch 还有很多其他损失函数，可以轻松切换到多分类（n *个*输出中的 1 个为真），甚至多标签分类（n *个*输出中的 k *个*为真）。

*优化器*对象告诉 pytorch 如何根据模型预测的错误程度更新模型参数。我们已经见过一个这样的优化器：随机梯度法！在我们的例子中，SGD 的更新规则对于更新我们的 β 参数来说非常简单--简单到我们可以很容易地用 `numpy` 写出来；然而，对于有多个层的大型网络来说，更新过程就变得非常复杂了--想想第 2 周讲座中的两层网络例子就知道了！PyTorch 的部分威力来自于优化器能够*自动*找出如何更新参数，而无需我们指定代码来更新。在这里，我们将使用 `SGD`[13]优化器。不过，还有很多其他优化器可以开始使用，你将在后面的作业中尝试使用一些优化器。所有优化器都需要知道它们要更新哪些参数，因此在实例化这些对象时，需要将模型参数作为参数传递进去，这就是为什么我们需要扩展 `nn.Module`，以便所有这些过程都能正常工作。

训练过程分为两步。首先，您将运行代码的*正向*传递，在这一过程中，您提供一些输入，然后模型做出预测。然后，您将启动该过程的*后向*部分，更新参数以做出更好的预测。这就是*反向传播*算法[14]根据网络中每个权重对最终预测的贡献来计算更新。损失函数对象将为我们计算损失和随后的梯度变化。然后，优化器将根据梯度更新参数。关于如何完成这一训练过程的教程有*很多很多*，因为无论训练网络的目的是什么，每个 pytorch 网络都会执行相同的一系列操作。pytorch 文档中有一个很好的视觉分类器训练过程示例，你还可以找到类似的步骤。

只要我们告诉 PyTorch，它 就会在多个步骤中自动跟踪梯度。当需要进行评估而不是训练时，我们需要告诉模型停止重新建立梯度。`pytorch` 模块类的 `train()` 和 `eval()` 函数可以

- 实例化 `BCELoss` 对象，将其用作损失函数

- 将 SGD 对象实例化为优化器。

[12] https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html#torch.nn。BCELoss

[13] https://pytorch.org/docs/stable/generated/torch.optim.SGD.html

14https://en.wikipedia.org/wiki/Backpropagation

- 编写你的训练循环，在指定的epoch次数内对数据集进行迭代，然后在epoch的每一步中随机采样一个实例（术语-文档张量中的行），并从 `pytorch LogisticRegression` 网络中获取预测结果。这个过程看起来与你在 `numpy` 中的核心训练循环非常相似，只是你要添加 `pytorch` 的特定位。使用损失函数测量损失，然后使用优化器执行反向传播步骤（使用 `backwards()` 和 `step()` 函数）。

- 编写一些代码，在训练过程中在开发数据上评估当前模型。您只需定期（而不是每一步）进行评估，预测代码将与测试数据相同（虽然没有分数）。请记住 `eval()` 调用！

7.4 培训与实验

现在我们已经定义了整个网络和训练过程，是时候开始训练了。由于在 `pytorch` 中训练的速度会更快一些，而且我们希望你能对代码进行一些探索，所以你将在 `pytorch` 中做一些实验，看看它们的效果。

对于大多数小型实验，你都要进行相同类型的操作：改变一个超参数，然后绘制模型在特定训练步骤中的表现。我们建议您尽量重构代码，以便于定期调用评估步骤来获取模型的损失及其在开发数据上的表现。对于绘图，我们强烈推荐使用 seaborn 进行绘图，尽管任何绘图都可以。虽然我们知道这些任务中有些看起来很重复，但它们的设计初衷是为了让您更直观地了解设计和超参数选择如何最终影响性能。您在这里看到的趋势可能会因未来的模型和数据而有所不同，但作为实践者，快速测试这些选择的影响的直觉和能力将在未来的家庭作业和其他作业中派上用场。

- 就像你对 `numpy` 代码所做的那样，对模型进行总共 1000 步的训练（即只向它展示 1000 个随机抽样的文档），并在每 20 步后报告损失。这应该可以验证损失在减少。

- 一旦您对模型的工作情况感到满意，请对模型进行至少 1 个历元的训练，并计算（并保存）(1) 每 50 步的损失和 (2) 模型在开发数据上的 F1 分数。

- 让我们看看加入正则化的效果如何。PyTorch 通过大多数优化器类（包括你使用的 `SGD`）的权重衰减参数来建立正则化。让我们看看将 L2 惩罚设置为 0（默认值）、0.001 和 0.1 会有什么效果。对于每个 L2 惩罚值，对模型进行总共 1 个 epoch 的训练，并在开发集上绘制每 50 步的损失和 F1 分数图，每个 L2 值用一条线（F1 vs. 损失图使用单独的图）。用几句话描述你所看到的：L2 对收敛速度和整体模型性能有什么影响?

- PyTorch 不仅仅有 SGD 优化器。回想一下，SGD 会根据梯度，利用学习率来调整步长。但如果我们

使用的不仅仅是梯度？例如，我们是否可以保留一点动力，让梯度保持同一方向？已经有很多新的优化器被提出，PyTorch 保留了比较成功的优化器的实现。一个更好的优化器的最大好处是可以帮助模型更快地学习。由于我们的一些大型模型可能需要数小时才能收敛，如果优化器能将训练时间缩短到一小时以内，这将在时间和环境方面带来巨大的好处。在这一步中，请将 `SGD` 优化器替换为 `RMSprop` 和 `AdamW` 这两种常见的优化器。对于每个优化器，对模型进行总共 1 个 epoch 的训练，并在开发集上绘制每 50 步的损失和 F1 分数图，每个优化器使用一行（使用单独的 F1 vs. 损失图）。用几句话描述你所看到的：优化器的选择对收敛速度和整体模型性能有什么影响?

- 我们有两种标记化方法。在实践中是否有一种方法表现更好？在基本设置（使用 SGD，无 L2 损失）下训练模型 1 个 epoch，每 50 步在开发集上绘制损失和 F1 分数图，每个优化器使用一行（使用单独的 F1 vs. 损失图）。用几句话描述您所看到的：标记化对整体模型性能有什么影响?

- 学习率对我们模型的收敛有什么影响？使用 SGD 的基本设置，将 `lr` 参数（即学习率）分别改为更大和更小的值，并持续 1 个 epoch，然后在开发集上每 50 步绘制损失和 F1 分数曲线，每个 `lr` 值绘制一条曲线。将三条曲线绘制在一起，并描述您观察到的情况：学习率对模型收敛速度有什么影响？欢迎（甚至鼓励！）尝试其他学习率。 如果你的模型收敛速度很快，你也可以减少本题的评估步骤数。

- 最后，使用您的最佳模型生成预测结果，并报告最终的 F1 分数。别忘了把预测结果上传到 Kaggle 上。请注意，这是一个单独的竞赛，因此您可以将您的分数与 `numpy` 版本进行比较。

7.5 可选部分 4

第 1 部分只使用了单字，即单词。然而，较长的单词序列（称为 *n-grams*）可以作为非常有用的特征。例如，"不令人反感 "和 "非常令人反感 "就是信息量非常大的特征，而单词本身可能无法为分类提供信息。然而，使用较长 *n-gram* 的缺点是，我们现在有了更多的

特征。例如，如果我们的训练数据中有 n 个单词，那么在最坏的情况下，我们可能会有 n 个^2 bigrams；仅仅 1000 个单词就会很快变成 1,000,000 个特征，这将会大大降低工作效率。因此，很多人根据频率或其他指标来阈值大词，以减少特征的数量。

在第 3 部分中，您将尝试添加大词组（两个单词）和三词组（三个单词），并测量它们对性能的影响。第 3 部分完全是可选的，是为那些想进一步了解功能工程方面的人准备的。

- 统计训练数据中的唯一词组、单词组、双词组和三词组的数量，并报告每个数字。

- 就我们可能观察到的数量而言，您所观察到的大字符和三字符数量是否接近最坏的情况？如果不是，为什么会这样？(提示：是否所有单词都同样常见？ 您还可以查看齐普夫定律）。
- 在开发数据中，训练数据中也出现的独特大词和小词的百分比是多少？
- 选择一个最小频率阈值，然后尝试更新解决方案，将其用作特征。我们建议创建一个新方法，封装 tokenize 方法并返回特征列表。

8 提交材料

请在截止日期前将以下内容上传到 Canvas：

1. PDF（首选）或 .docx 文件，其中包含您对上述问题的回答和绘图。**提交时请注明您在 Kaggle 上的用户名。**
2. 所有部分的代码。应作为单独文件上传，*而不是*放在 .zip 或其他压缩包中。

代码可以以独立文件（如 `.py` 文件）或 Jupyter 笔记本的形式提交。**我们保留运行您提交的任何代码的权利。无法运行或输出结果大相径庭的代码将被记为零分。**

9 学术诚信

除非作业中另有说明，否则所有提交的作业必须是您自己的原创作品。任何摘自他人作品的摘录、陈述或短语都必须清楚地标明出处，并提供适当的引文。任何违反大学学术和专业诚信政策的行为都可能导致严重的处罚，从不及格、不及格课程到 被开除出该专业。违反学术和职业诚信的行为将报告给学生事务处。影响作业或课程成绩的后果由任课教师决定；可能会施加额外的处罚。

在完成本作业时，不应使用 CoPilot 或 ChatGPT 等其他编程工具。

参考资料

Carla Perez-Almendros、Luis Espinosa-Anke、Steven Schockaert。2022.SemEval-2022任务4：谄媚和屈尊语言检测。第 16 届语义评估（SemEval-2022）国际研讨会论文集》，第 298-307 页，美国西雅图。As- sociation for Computational Linguistics.