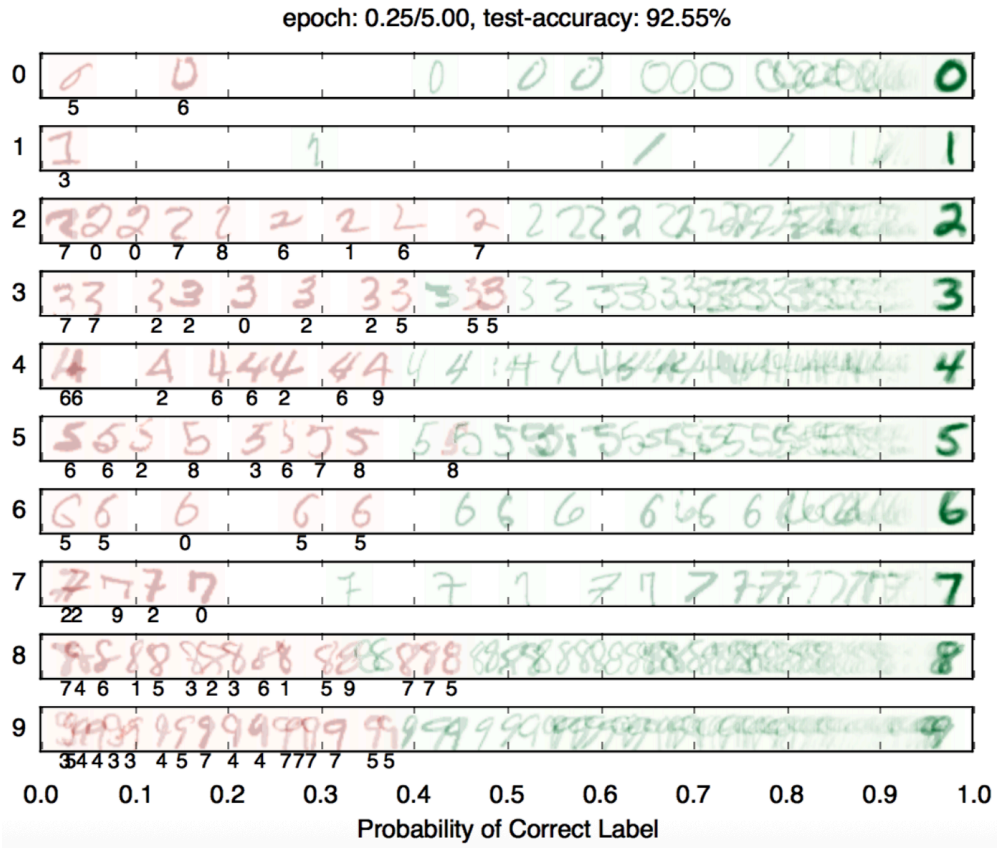
14 Machine Learning II



本节实验目标：练习机器学习分类

报告说明（重要）：本部分“机器学习”包含两周内容，本节课为第二周内容，请留意该文档目录中标有“报告应实现任务[分值]”的部分，请在实现后及时保存文件和运行结果截图，在课程报告中提交。

报告6 “Machine Learning” 提交说明

- 按照第13周和本周第14周的Question 1-4 的要求实现代码和获得运行结果截图
- **提交压缩包命名为”姓名_学号_报告序号.zip”** (如“彭振辉_2106666_报告6.zip”)
- 压缩包应**包含内容**：
  - **已实现的完整项目文件夹”Project_6_Machine_Learning_full“**
    - 其中 models.py 中有Question 1-4要求的函数实现
  - **一个doc或pdf说明文档，上面需要有：**
    - 开头一段说明 “**整体实现参考 + 2-3句简要体会（如教训、思路、拓展应用等）**”，如：-

    - “自行实现。挑战最大的是xxx内容，初始时报了什么错，通过什么方式解决，该部分的实现思路为xxx”
    - “xxx内容参考xxx同学/xxx网址。思考不出算法思路，探究后学习到了什么方法。”
- **要求实现的4个任务的成功运行截图，说明截图对应任务。**

Introduction

本堂课的代码文件包含：

你应该要编辑的文件:	
models.py	**适用于各种应用的感知器和神经网络模型**
你应该看但不要编辑的文件:	
nn.py	**神经网络迷你库**
你可以忽略的文件:	
autograder.py	Project autograder
backend.py	Backend code for various machine learning tasks
data	Datasets for digit classification and language identification
submission_autograder.py	Submission autograder (generates tokens for submission)

- Autograder会评判你的实现。请**不要** 更改代码中提供的任何函数或类的名称。
- **正确使用数据集**：你在此项目中的部分分数取决于你训练的模型在Autograder随附的测试集上的表现。我们不提供任何 API 供你直接访问测试集。

相关包的安装和nn.py的说明参考第13周的说明

Q1-2的实现也需要在本堂课的报告中提供

Q3-4为本节课作业要完成的代码

Question 3: Digit Classification 报告应实现任务[2分]

- **说明：该小题截图只需截 python autograder.py -q q3的运行结果，获得评分器给的全部9分，这道题才算得它占的2分。**

对于这个问题，你将训练一个网络来对 MNIST 数据集中的手写数字进行分类。

每个数字的大小为 28×28 像素，其值存储在一个 784 维的浮点数向量中。我们提供的每个输出都是一个 10 维向量，它除了对应于正确数字类别的位置的 1，在其它位置都为 0 （one-hot vector）。

在 `models.py` 中完成 `DigitClassificationModel` 类的实现。 `DigitClassificationModel.run()` 的返回值应该是包含分数的 *batch_size* × 10 大小的节点，其中分数越高表示数字属于特定类别 (0-9) 的概率越高。你应该使用 `nn.SoftmaxLoss` 作为你的损失。不要在网络的最后一层之后放置 ReLU 激活。

对于这个问题和 Q4，除了训练数据之外，还有验证数据和测试集。你可以使用 `dataset.get_validation_accuracy()` 计算模型在**验证集**上的准确度，这在决定是否停止训练时很有用。autograder将使用**测试集**。

要获得此问题的所有分数，你的模型应在测试集上达到至少 97% 的准确度。作为一个参考，我们的参考实现在经过大约 10 个 epoch 的训练后，能在验证数据上达到差不多 98% 的准确率。请注意，autograder对你在**测试集上的准确度(test accuracy)**进行评分，但你在实现过程中只能知道你在验证集上的准确度(validation accuracy) — 因此，就算你在验证集上达到了97%的阈值(threshold)，autograder仍然有可能因为你在测试集上达不到97%而不给分。因此，在验证准确度上设置稍高的停止阈值可能会有所帮助，例如 97.5% 或 98%。

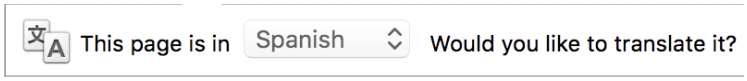
要测试你的实现，请运行：

```
python autograder.py -q q3
```

Question 4: Language Identification 报告应实现任务[1分]

- **说明：该小题截图只需截 python autograder.py -q q4的运行结果，获得评分器给的全部7分，这道题才算得它占的1分。**

语言识别(Language identification)的任务是在给定一段文本的情况下确定文本是用什么语言编写的。例如，你的浏览器可能能够检测你是否访问过外语页面并提供翻译给你。 这是 Chrome 的一个示例（它使用神经网络来实现此功能）：



在这个Question中，我们将构建一个较小的神经网络模型，一次识别一个单词的语言。 我们的数据集由五种语言的单词组成，如下表:

Word	Language
discussed	English
eternidad	Spanish
itseänne	Finnish
paleis	Dutch
mieszkać	Polish

不同的单词由不同数量的字母组成，因此我们的模型需要有一个可以处理可变长度输入的架构。跟前面的问题中单个输入 x 不一样，我们将把单词中的每个字符分别输入： $x_0, x_1, \ldots, x_{L-1}$ ，其中 L 是单词的长度。我们将首先应用一个网络 f_{initial}，它就像前面问题中的前馈网络一样。它接受其输入 x_0 并计算维度为 d 的输出向量 h_1： $h_1 = f_{\text{initial}}(x_0)$

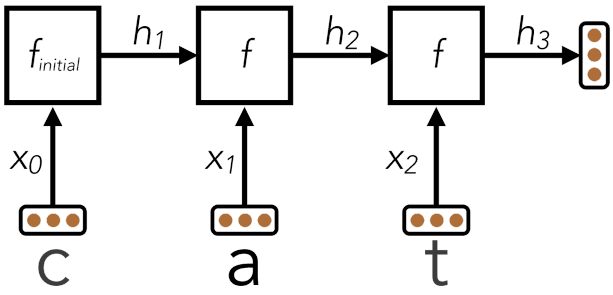
接下来，我们将上一步的输出与单词中的下一个字母相结合，生成关于单词前两个字母的向量。为此，我们将应用一个输入一个字母并输出一个隐藏状态的子网络，且它还依赖于先前的隐藏状态 h_1 。我们将此子网络表示为 f ： $h_2 = f(h_1, x_1)$

持续对输入单词中的所有字母进行这种操作， 其中每一步的隐藏状态总结了网络迄今为止处理的所有字母： $h_3 = f(h_2, x_2)$

$$\vdots$$

在这些计算中，函数 $f(\cdot,\cdot)$ 是同一块神经网络并使用相同的可训练参数； f_{initial}也将共享一些与$f(\cdot,\cdot)$相同的参数。这样处理不同长度的单词时使用的参数都是共享的。你可以在提供的输入 `xs` 上使用 `for` 循环来实现这一点，其中循环的每次迭代都会计算 f_{initial} 或 f。

上述技术称为循环神经网络 (RNN)。 RNN的示意图如下所示：



在这里，RNN 用于将单词“cat”编码为固定大小的向量 h_3。

在 RNN 处理完输入的完整长度后，它会将任意长度的输入单词*编码 (encode)* 为固定大小的向量 h_L，其中 L 是单词的长度。输入单词的向量编码现在可以输入到额外的输出层，以生成单词语言的分类分数。

批处理

尽管上述等式是针对单个单词的，但实际上你必须使用成批的单词来提高效率。为简单起见，我们在项目中的代码确保单个批次中的所有单词具有相同的长度。在批处理形式中，隐藏状态 h_i 被替换为维度 *batch_size* $\times d$ 的矩阵 H_i。

设计技巧

循环函数 $f(h, x)$ 的设计是这项任务的主要挑战。以下是一些提示：

- 从你选择的前馈架构 $f_{\text{initial}}(x)$ 开始，只要它至少具有一个非线性。
- 给定 $f_{\text{initial}}(x)$，你应该使用以下构造 $f(h, x)$ 的方法。 f_{initial} 的第一层将首先将向量 x_0 乘以某个权重矩阵 $\mathbf{W}$ 以产生 $z_0 = x_0 \cdot \mathbf{W}$ 。对于后续字母，你应该使用 `nn.Add` 操作将此计算替换为 $z_i = x_i\mathbf{W} + h_i\mathbf{W}_{\text{hidden}}$。换句话说，你应该用 `z = nn.Add(nn.Linear(x, W), nn.Linear(h, W_hidden))`。
- 隐藏层大小 d 应该足够大
- 从 f 的浅层网络开始，在使网络更深之前找出隐藏层大小和学习率的良好值。如果你立即开始使用深度网络，你将拥有成倍增加的超参数组合，并且任何单个超参数错误都会导致你的性能受到严重影响。

你的任务

完成 `LanguageIDModel` 类的实现。

要在这个问题上获得满分，你的架构应该能够在测试集上达到至少 81% 的准确度。

要测试你的实现，请运行：

```
python autograder.py -q q4
```

免责声明：此数据集是使用自动文本处理生成的。 它可能包含错误。 它也没有做敏感词过滤。 然而，尽管存在数据限制，我们的一个实现仍然可以正确分类超过 89% 的验证集。 作为参考，我们的一个实现需要 10-20 个 epoch 来训练。