Lab 2

目录

介绍

在这个项目中，你将使用numpy实现课上学到的各种算法，以帮助pacman识别具有不同正负分数的豆子。

代码库发生了大幅改变，请从教学网上下载lab2。

评估： 我们使用`autograder.py`对你的提交进行评分，但包含的测试用例与本地给出的样例文件有所不同。你可以在本地运行评分器对你的代码进行评测，但仅限于帮助调试代码，该分数不等于最终分数。

学术诚信： 我们会将你的代码与课堂上其他提交的代码进行逻辑查重。如果你拷贝了别人的代码，并做一些微小的修改，我们会很容易发现，请不要尝试。我们相信你们会独立完成作业。

相关文件

你需要修改的文件:

info.yaml 姓名学号

- 线性回归
  - answerLogisticRegression.py 线性回归模型
- 分类树
  - answerTree.py 树模型
  - answerRandomForest.py 随机森林模型
- 反向传播算法
  - autograd/BaseNode.py: 计算图上的结点
  - autograd/BaseGraph.py: 计算图类
  - answerMultiLayerPerceptron.py: 用反向传播实现MLP
- 给pacman添加视觉
  - MnistModel.py pacman agent的视觉模型
  - YourTraining.py 视觉模型的训练代码

你需要阅读的文件:

- mnist.py 载入MNIST数据集的接口
- debug_scalarfunction.py 用于debug 反向传播和评分的文件

- debug_neuralnetwork.py 用于debug 反向传播和评分的文件
- answerSoftmaxRegression.py：用反向传播实现SoftmaxRegression

你可以忽略的文件:

- `pacman.py` 运行Pacman游戏的主要文件。这个文件还描述了一个Pacman `GameState`类型。
- `game.py` 吃豆人世界运作的逻辑。这个文件描述了几个支持类型，如AgentState、Agent、Direction和Grid。
- `util.py` 实现算法时的可能会用到的数据结构。
- `graphicsDisplay.py` Pacman图像显示界面
- `graphicsUtils.py` 支持图像
- `textDisplay.py` Pacman的ASCII编码
- `ghostAgents.py` Ghosts程序
- `keyboardAgents.py` 键盘接口
- `layout.py` 导入地图的程序
- `autograder.py` 自动评分器

你需要修改并提交的文件:

所有需要修改的文件。你可以使用以下指令来打包答案文件。

```
python compressAnswers.py
```

关于数据集：

本项目使用MNIST数据集。我们按照1000:69000将其分成训练集和验证集(MNIST/train_data.npy, MNIST/valid_data.npy)，并通过对MNIST数据集进行数据增强得到测试集(MNIST/test_data.npy)，用于pacman游戏中。

`mnist.py` 提供了载入数据的统一接口。这一文件中你可能用到的变量有

```
num_feat = 784 # 每张图片是28*28的矩阵。在训练集和验证集中，图片已经展平成一个784维的向量。在测试集中没有展平。
num_class = 10 # MNIST的任务是识别0-9数字，即10分类。
trn_X, trn_Y # 训练集的图片与标签。每个标签是一个0-9的整数。
val_X, val_Y # 验证集的图片与标签。
test_X, test_Y # 测试集（子集）的图片与标签。
```

注意：

- q1，q2，q3的所有的模型都是在**训练集**上进行训练；相关文件（如`autograder.py`，`model*.py`）给出的最终准确率（acc）与评分都是在**验证集**上计算的，也请以**验证集**分数为依据进行调参。
- 对于q4，你可以自由对MNIST数据集合（trn_X, trn_Y, val_X, val_Y）进行划分，以进行模型的训练和调参（即，你甚至可以将所有验证集一起用于对模型的训练）；相关文件给出的准确率与评分是在已公开的**测试集子集**上计算得到的。

- 目前只公布了测试集的一个子集。该测试集子集的目的只是给大家一个关于最终测试数据形式的印象（即，大概进行了什么样的*数据增广*），以及大概的测试分数；**请不要使用该测试集子集进行训练或其他相关操作（如早停等）**。在计算q4的最终分数时，我们会在**完整测试集**上进行测试。

你可能用到的一些操作

- `copy.deepcopy`: 复制对象
- `numpy`库中的基本操作（官方文档）。请你特别注意与以下有关的概念/操作：array的indexing与slicing相关操作，np.unique(return_count=True)，random number随机数生成，transposing与reshaping，broadcast，。
- `A@B`: 对矩阵A，B进行矩阵乘法
- 注意以下操作的axis和keepdims参数：`np.sum`，`np.average`，`np.std`。
- 注意，`log(0)`，`/0`，`exp(很大的数字)`，都会导致`inf`和`nan`产生。
  - 前两者可以通过`log(x+EPS)`或`/(x+EPS)`避免（EPS是个很小的数，例如1e-6）。
  - 对于第三种情况，通常有如下解决方法：对于$\ln(exp(x1)+exp(x2))$，当$x1>x2$时，可以化为$x1+\ln(1+exp(x2-x1))$，来避免可能存在的数值溢出；类似地，当 $ln(1+e^x)$中的e^x很大时，可以改为等价地计算$x+ln(1+e^{-x})$。
- 本次作业的部分内容可能比较复杂，因此我们建议大家在遇到困难时灵活使用 debug 功能，方便检查各变量以及代码行为。
  - 以 Vscode 为例，在debug时，可以首先创建 .vscode/launch.json 并填入如下内容（默认配置）：

    ```
    {
    "version": "0.2.0",
    "configurations": [
      {
        "name": "Python Debugger: Current File",
        "type": "debugpy",
        "request": "launch",
        "program": "${file}",
        "console": "integratedTerminal",
      }
    ]
    ```

  - 在需要检查的代码处添加断点（对应行左边添加红点）；
  - 监视需要检查的变量（在 WATCH 处添加变量名，也可输入表达式）；
  - 运行 Python Debugger，代码将运行至断点处，方便观察代码运行情况及变量行为；
  - 根据情况添加新的断点，新的监视变量。选择运行至下一个断点，或进行单步调试等操作继续检查；
  - 图示教程可查看第一次上机辅导 ppt 中 vscode debug 相关部分；也可自由在网上搜索python debug的相关资料，提高你的debug能力。

新的pacman规则

- 与lab1相比，lab2的pacman中每个食物被赋予了一个数字；不同的数字对应不同的分数，且有正有负。
- pacman只允许走有限步。
- 只有当pacman在有限步内吃掉所有正分数的食物时获胜（此时会一次性+50分）；pacman移动时不扣分；pacman吃到食物时加上相应的分数。
- 你不需要自己写agent，只需要设计图片分类模型，以使得pacman能够辨认数字。

地图文件存放在layouts文件夹下，一个典型的地图文件如下

```
%%%%%%%%%%%
%87 32 2P0%
%65 6  1 4%
%4  10 993%
%%%%%%%%%%%
# -10 -10 -10 -10 -10 10 10 10 10 10 False
```

其中：

- P代表pacman的起始位置，%代表墙壁。数字代表食物的分数。根据数字的值，游戏会随机从数据集中抽取该值对应的图片。
- 最后一行从前到后依次代表着0-9数字对应的分数。最后一个True/False代表着是否轮换数字：例如在轮换后，地图左上角的数字8可能会变为2，2的分值会变为10，8的分值会变为-10。换言之，你不能让你的模型只对特定的类识别特别准。

所有用于评分的地图已经公布。测试地图中都没有鬼。如果你想实现有鬼的情况，可参见选做的q5（该题不计分，不提交）。

Question 1 (4 points): 逻辑回归

在这一部分，你需要根据第9节课上所教学的内容来实现逻辑回归：**构建一个线性模型，以在给定MNIST上的某个数字图片时，判断其是否为0（即，binary classification）；使用梯度下降法完成优化；进行适当的调参来达到较高的准确率。**

我们在`modelLogisticRegression.py`给出了模型的训练流程。你只需要修改`answerLogisticRegression.py`，完成线性模型的构建以及一次梯度下降过程的相关代码，以及超参数的调整。

代码完成后，运行

```
python modelLogisticRegression.py
```

以进行模型训练及模型保存。如果你发现计算结果中出现了`nan`或`inf`，可以参考`可能用到的一些操作`一节中的提示。

在完成debug与调参后，你可以运行

```
python pacman.py -p ReflexAgent -a model="LR" -l strict --maxstep 40
```

以查看你的模型在pacman中的运行效果。该例子使用了layouts/strict.lay作为地图，其中只有数字0是正分，且严格只有一条道路能够满足使得pacman吃完所有正分食物并获胜的条件。你可以在指令最后添加`-v`或者`--visualize`在地图中可视化模型对每个数字的预测结果（右下角红色数字）。

> **思考**：你可能会发现一个现象：虽然你的平均预测准确率很高，但在实际中pacman却表现得很差。这可能是由于类别不均衡（class imbalance）问题导致的：在MNIST中，90%的数据是非零，因此一个一直预测非零的分类器就可达到90%的准确率——但实际中它一个零也预测不出来。对于类别不均衡问题，准确率并不是个很好的指标，此时可以使用confusion matrix来更好地衡量模型性能。（然而，在本题目中你不需要解决类别不均衡问题。我们最终仍然将**以图片分类准确率来对q1进行评分**。）

最终，请你运行

```
python autograder.py --q q1
```

来对你的模型进行评分（分值范围：0-100）。注意，如果你使用默认的超参数，分数可能会很差。因此，你需要付出一定的努力来进行调参。本题的满分线是`valid acc = 0.98`.

> 注意，autograder可能会报错`python: command not found`，此时请修改autograder.py开头的PYTHON_PATH为你的python程序的绝对位置（Linux和苹果用户可以在shell中用`which python`, windows用户可以在powershell中用`gcm python`）

Question 2 (5 points): 树模型

在这一部分，你需要**构建一个分类树模型，以在给定MNIST上的某个数字图片时，判断其对应的数字值（即，10-way classification）；进行适当的调参来达到较高的准确率。**

我们在`modelTree.py`中给出了分类树模型的训练流程；你只需要修改`answerTree.py`，完成树模型构建的相关代码，并进行调参。

代码完成后，运行

```
python modelTree.py
```

以进行模型的训练及保存。

在完成debug与调参后，你可以运行

```
python pacman.py -p ReflexAgent -a model="Tree" -l accuracy --maxstep 40
```

来看你的模型在pacman中的运行效果。这个例子使用layouts/accuracy.lay作为地图，其中偶数数字是正分，奇数数字是负分。你可以在指令最后添加`-v`或者`--visualize`在地图中可视化预测结果。

在树模型的基础上，**你需要构建随机森林模型，来进一步提升模型的性能**。我们在`modelRandomForest.py`给出了随机森林模型的训练流程，你只需要修改`answerRandomForest.py`，完成随机森林模型的构建。

代码完成后，运行

```
python modelRandomForest.py
```

进行训练并保存模型。

> 如果你发现你所构建的随机森林模型性能很差，可以先把森林中树的数目设为1，以进行错误分析：此时，随机森林模型的性能应当与树模型的性能相当；如果此时随机森林模型的性能仍然与树的性能差距很大，建议你逐步删除森林比树多出的那些操作，找出性能差距的原因，并在找到问题后将这些操作加回去。

在完成debug与调参后，你可以运行

```
python pacman.py -p ReflexAgent -a model="Forest" -l accuracy --maxstep 40
```

来看你的模型在pacman中的运行效果。同样地，你可以在指令最后添加`-v`或者`--visualize`在地图中可视化预测结果。

最后，请你运行

```
python autograder.py --q q2
```

来对本部分的两个模型进行评分。本题的满分线是：树`valid acc 0.545`，森林`valid acc 0.71`。

Question 3 (6 points): 反向传播

在这一部分，请你**修改`autograd/BaseNode.py`与`autograd/BaseGraph.py`以完成计算图与相关计算节点的定义。**在`BaseNode.py`中，你需要补全一些计算结点的前向或反向传播结果（对于反向传播，输入为上游梯度，输出为下游梯度）；在`BaseGraph.py`中，你需要为一个任意给定的计算图实现前向传播和梯度的反向传播的基本逻辑。注意，本题中只需要考虑每个结点只有一个输入和一个输出的情况（即，结点连成直线的计算图；这里的“一个输入”可能是一个向量）。

你可以使用`debug_NeuralNetwork.py`与`debug_ScalarFunction.py`进行debug。它会通过有限元近似计算数值梯度，并将近似计算得到的数值梯度与你所编写的程序计算出的梯度打印出来。请确保你定义的计算图和计算结点通过反向传播得到的梯度和数值梯度大致相等，再进行后续模型的搭建。其中，`debug_ScalarFunction.py`的使用方式为：

```
python debug_ScalarFunction.py input/sf.relu.in
```

其中`sf.relu.in`是一个定义标量计算图的文件，可以换成`sf.sigmoid.in`, `sf.tanh.in`, `sf.mix.in`；若同时在命令中输入`--debug`，则会开启debug模式，输出各结点输入输出的矩阵形状（可选）。类似地，你也可以对神经网络计算图进行debug。

```
python debug_NeuralNetwork.py input/nn.BatchNorm.in
```

其中的`nn.BatchNorm.in`可以换成`nn.CELoss.in`, `nn.Dropout.in`, `nn.Linear.in`, `nn.multilayer.in`, `nn.NLLLoss.in`。同样，可选开启debug模式。

注意，上述debug程序只会检查数值梯度与反向传播梯度的接近程度。如果你的正反向传播同时错误，就会导致程序能通过debug，但是不能通过本题的autograder。

基于反向传播框架，你可以实现Softmax回归与多层感知机MLP，他们的训练流程分别在`modelSoftmaxRegression.py`与`modelMultiLayerPerceptron.py`。其中，Softmax回归已经在`answerSoftmaxRegression.py`中实现。**请你在`answerMultiLayerPerception.py`中使用上述计算图，来定义一个多层感知机(MLP)，以使其能在MNIST上进行10分类；同时进行适当的调参。**

代码完成后，运行

```
python modelSoftmaxRegression.py

python modelMultiLayerPerceptron.py
```

进行训练及模型保存。

在完成debug与调参后，你可以运行

```
python pacman.py -p ReflexAgent -a model="SR" -l accuracy --maxstep 40

python pacman.py -p ReflexAgent -a model="MLP" -l accuracy --maxstep 40
```

以查看你的模型在pacman中的运行效果。注意，即使你之前在验证集上获得极高的准确率(>90%)，有可能pacman仍然对测试集预测得不是很准，导致表现不好。你将会在下个问题中尝试解决它。具体可查看autograder.py。

最后，请你运行

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

来完成对本部分的评分。本部分的评分会查看你所实现的计算结点是否能通过有限元数值梯度的debug测试（占比80%），以及你实现的MLP在验证集上的准确率（占比20%）。本题的满分线是通过所有debug，MLP的valid acc达到0.87。

Question 4 (5 points): 给pacman添加视觉

在本问题中，你可**自由定义并训练任何模型**，使得pacman能较为准确地识别*测试集*中的数字。

具体而言：

- 你可以任意调整lr，wd，batch size或网络结构等，甚至可以实现卷积算子。
- 允许使用全部给到你的数据集。
- 允许模型集成。
- 允许数据增强。
- 你需要自己在`YourTraining.py`中编写模型训练的相关代码（可以参照之前`model*.py`相关文件中的代码），并且对训练好的模型参数进行保存（可以使用`pickle`库进行保存）。
- 你需要在`MnistModel.py`中创建你的模型类（其中，你需要首先对之前已保存的模型进行读取；可以参照该文件中其他模型类）；修改`MnistModel.py`中的`modeldict["Your"]`以设置你的模型。

但注意：

- 直接使用q1-q3中的模型，或暴力记住所有MNIST数字可能并不能取得很好的效果（因为我们会在模型没有见到过的测试集上进行模型测试）。
- 请将训练时间控制在600秒以内（我们的autograder会进行检查）。
- 最终测试时，我们会首先在一个没有test data的环境中运行`YourTraining.py`，然后在一个没有`YourTraining.py`的环境中测试Agent。由于时间限制，**请不要在测试时进行模型训练**，即不要在`MnistModel.py`中包含训练代码。

你可以通过执行

```
python pacman.py -p ReflexAgent -a model="Your" -l smallPKU
```

来看看具备"Your" model赋予的视觉能力的pacman在smallPKU地图上表现。如果你的模型一开始表现不是很好，也不要气馁，*这是因为这里使用的测试集都是经过数据增强（例如，轻微的旋转与平移等）的较难样本*。因此需要你进一步开动脑筋，努力提升你模型的泛化性能。

最后，请你运行

```
python autograder.py --q q4 --q4training
```

来对本部分进行评分。`--q4training`会在测试前首先调用`YourTraining.py`进行训练。如果已经训练完成，可以去掉这一选项。评分会分别在"largeAccuracy", "onepath", "VeryLargePKU"三个地图上收集多次pacman的得分来计算最终得分。达到本题的满分很难，但是达到70分左右比较容易。

> 再次注意，在最终测试时，我们会在没有testdata的环境中调用你的`YourTraining.py`文件对你的模型进行重新训练。因此，请确保你的模型训练代码`YourTraining.py`能够正常运行，且能够正常保存模型参数到指定路径；同时，你应当确保最终提交的`YourTraining.py`中不会调用MNIST test data以进行任何操作（如利用test data进行训练、根据test data进行早停、在中间阶段评估test data上的准确率等）。

Question 5 (0 points): 使用你自己的Agent

本题不提交，不计分。

本题允许使用你在lab1中创造的agent。可以将你的Agent加入MultiAgent.py中，并将原来的getAction改名称为getVisAction即可。

我们提供了一个有鬼的地图 ghostPKU，可运行如下命令测试你的Agent在有视力、对抗情况下的表现：

```
python pacman.py -p YourAgent -a model="Your" -l ghostPKU
```

Acknowledgement

Designed and implemented by Xiyuan Wang, Fanxu Meng and Muhan Zhang from the GraphPKU Lab. Thanks Hongze Li, Zichen Liu and Jiahuan Zhou for module contributions, testing and code comments. Thanks all professors and TAs of the 2023 Introduction to AI team for the valuable discussions. The project is based on UC Berkeley CS188 course project.