Training a Neural Network

요약 Summary

- 마감: 2022.05.30(월) 오후 11:59
- Part 1: 아주 간단한 1-layer neural network 의 training 을 위한 programming
  (모두 `model.py` 파일 내 `train_nn()` 함수 내)
  - Non-linear activation function(cross-entropy loss 의 경우 activation function 은 softmax 함수)의 계산을 포함한 forward pass 의 세부 과정을 구현하세요.
  - Cross-entropy loss function 를 계산하는 세부 과정을 구현하세요.
  - Network 의 parameter 에 대한 loss 함수의 gradient 를 계산하는 backpropagation 및 이를 통해 parameter 를 업데이트 하는 stochastic gradient descent 의 세부 과정을 구현하세요.
  - `model.py, main.py` 파일들의 주석 설명문을 자세히 읽고 작업하세요.
  - `hyperparameters.py` 파일에 명시된 learning rate, batch size 등 hyperparameter 값을 다양하게 조정해 가면서 다양한 실험을 진행하여 최적의 결과를 도출하세요.
- Part 2: PyTorch 를 이용하여 강의동영상에서 설명된 convolutional neural network 중 LeNet5 를 정의하고 이를 training 하기 위한 programming
  (모두 `model_pytorch.py` 파일 내)
  - 수업 시간의 설명을 바탕으로 자체적으로 PyTorch 의 사용 방법을 확인하면서 https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html 등 관련 자료를 조사하고 숙지하세요.
  - LeNet5 네트워크의 layer 변수를 정의하는 과정과 LeNet5 의 forward pass 의 세부 연산 과정을 각각 `class Model_pytorch` 의 `__init__(), forward()` 함수 내에 구현하세요.
  - `model_pytorch.py, main.py` 파일들의 주석 설명문을 자세히 읽고 작업하세요.
  - `hyperparameters.py` 파일에 명시된 learning rate, batch size 등 hyperparameter 값을 다양하게 조정해 가면서 다양한 실험을 진행하여 최적의 결과를 도출하세요.
- Part 3: 보고서
  - 1-layer neural network 의 training: `train_nn()` 함수의 각 세부 과정에 대한 관련 이론을 요약하여 구현된 코드와 관련지어 설명하세요.
  - Pytorch 기반 CNN 의 training: `class Model_pytorch` 의 `__init__(), forward()` 함수들의 각 세부 과정에 대한 관련 이론을 요약하여 구현된 코드와 관련지어 설명하세요.
  - 다양한 hyperparameter 에 대해 수행한 실험 결과들을 정리하여 제시하세요.
  - 기본 제공된 `Data` 폴더의 예시 영상들에 대해 도출된 결과를 제시하세요.

- 제출방법:
  - `model.py` 파일과 보고서를 하나의 파일로 압축하여 가상대학에 업로드 하세요.
  - 압축파일 파일 이름은 `hw4_본인학번.zip` 으로 하세요.
    (ex: `hw4_20181234.py`)
  - `model.py` 파일은 파일 이름을 `model_본인학번.py` 으로 변경하세요.
    (ex: `model_20181234.py`)
  - 보고서 파일는 report_본인학번.pdf 로 지정하세요. (ex: `report_20181234.pdf`)

개요 Overview

본 과제에서는 MNIST database 및 15 scene database(Lazebnik et al. 2006)에 대해 neural network 를 이용하는 image classification 기법의 학습 및 테스트를 수행합니다.

주어진 과제 Task 1: `train_nn()` 함수 내 1-layer neural network 의 training 과정 구현.

- Cross-entropy loss function 계산 과정을 포함한 forward pass 의 세부 과정
- Backpropagation 의 gradient 계산 및 이를 통해 parameter 를 update 하는 stochastic gradient descent 의 세부 과정

배경이론: 세부적인 각 과정의 이론은 동영상 강의 19,20 의 Training Neural Networks 강의에서 자세히 설명이 되었으며, 관련된 상세 이론은 아래와 같이 요약됩니다:

1-layer network 의 출력: $$l_j = \mathbf{w}_j \cdot \mathbf{x} + b_j = \sum_{i=1}^{784} w_{ij} x_i + b_j$$ → softmax 적용 후 $$p_j = \frac{e^{l_j}}{\sum_j e^{l_j}}$$

Cross entropy loss: $$L(w, b, x) = -\sum_{j=1}^{10} y_j \ln(p_j)$$

j = class index, y_j = 1 if true class, y_j = 0 if not true class

Network parameter 에 대한 gradient:

$$\frac{\delta L}{\delta w_{ij}} = \frac{\delta L}{\delta p_a} \frac{\delta p_a}{\delta l_j} \frac{\delta l_j}{w_{ij}} = \begin{cases} x_i(p_j - 1), a = j \\ x_i p_j, a \neq j \end{cases}$$

$$\frac{\delta L}{\delta b_j} = \frac{\delta L}{\delta p_a} \frac{\delta p_a}{\delta l_j} \frac{\delta l_j}{b_j} = \begin{cases} (p_j - 1), a = j \\ p_j, a \neq j \end{cases}$$

여기에서 설명된 내용을 바탕으로 forward pass 의 neural network 출력, softmax 함수, cross entropy loss 함수, gradient 등의 계산 수식을 `Numpy` 함수를 이용하여 구현하면 됩니다.

*__주의사항__: cross entropy loss 를 계산하는 코드를 구현할 때 log 함수에 0 이 대입될 경우 무한대를 의미하는 inf 값이 도출될 수 있으므로, 이런 문제를 예방하기 위해 최대한 수식을 잘 재정리하여 log 함수에 0 이 대입되는 경우를 예방하고, 혹시 지나치게 수치가 작은 값이 대입되어 numerical error 로 인한 inf 가 발생하지 않도록 입력 값을 제한(clipping)하도록 해야 할 것임.

활용가능한 함수들: `Python, Numpy` 의 기본 함수들 외 기타 함수 사용 필요하지 않음

주어진 과제 Task 2: `class Model_pytorch` 내 PyTorch 를 이용한 LeNet5 모델 학습 과정 구현.

- `class Model_pytorch` 의 `__init__()` 함수 내 network layer 변수들을 생성하는 과정
- `class Model_pytorch` 의 `forward()` 함수 내 network forward pass 연산 과정

배경이론: PyTorch 는 TensorFlow 와 함께 deep learning 을 수월하게 하는 가장 대표적인 python package 입니다. 사용을 위해서는 우선 패키지를 여러분의 환경에 설치해야 하는데, anaconda 기반의 python 환경에 대해 아래의 명령어를 통해 설치할 수 있습니다. (이때 본 강의의 실습 01 강의동영상을 참조하여 가상환경 등 설정에 유의하세요.)

To install this package with conda run:

```
conda install -c pytorch pytorch
```

활용가능한 함수들: `Python, Numpy, PyTorch` 의 기본 함수들 사용 가능함

제공되는 초기 코드 세부사항 Starter code details

초기성능 *Initial performance* : `main.py` 를 scene recognition 용 data 에 수행하면 모든 test image 의 category 를 임의로 찍어서 randomly guess 출력을 도출합니다. 이러면 15 개 classes 에 대해 15 개 중 하나 정도가 맞게 되므로 출력 정확도가 1-layer network 기준 약 ~7% 정도 됩니다.

scenerec Data : 초기 코드는 각 category 마다 100 개씩의 영상에 대해 학습 및 테스트를 수행합니다. 즉 1500 개의 training 영상에 대해 학습한 후 1500 개의 test 영상에 대해 분류 결과를 도출합니다. 실제 연구 논문에서는 전체 data 를 임의의 training/test 집합으로 임의 분류하고 정확도를 측정하는 작업을 반복 수행해야 하겠지만 초기 코드에서는 디버깅이 수월하도록 이런 과정이 생략되었습니다.

실험하기 Running the code

과제 3 과 마찬가지로 이번 과제에서도 Jupyter Notebook 이 아니라 Visual Studio Code 를 기본 통합개발환경 Integrated Development Environment (IDE)로 활용하겠습니다. Visual Studio Code 의 설치 및 사용법은 추가적인 문서로 제공해 드리니 참고하세요.

`main.py` 를 실행하기 위해서는 2 개의 command line 입력인자 argument 를 입력해야 합니다. `-data` 는 인식 문제 및 DB 의 종류(`-data mnist` 또는 `-data scenerec`)를 가리키고, -net 은 network 종류 (`-net cnn` 또는 `-net one_layer`), `-mode` 는 classifier 방식(`-mode torch`, `-mode nn` 또는 `-mode nn+svm`)을 가리킵니다.

*주의사항: -net cnn→ -mode 를 torch 로, -net one_layer → -mode nn 또는 nn+svm 중에서 선택해야 합니다.

예시:

```
$ python main.py -data scenerec -net cnn -mode torch'
$ python main.py -data scenerec -net one_layer -mode nn'
```

더 상세한 내용은 `main.py` 를 참고하세요.

Visual studio code 를 통해 위의 입력인자를 입력하면서 `main.py` 를 실행시키기 위해서는 Launch Configuration 를 생성해야 합니다. 이를 위해서 `launch.json` 이라는 파일에 입력인자들을 아래와 같이 기입하면 됩니다:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Current File (Integrated Terminal)",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "args": ["-data", "scenerec", "-net", "cnn", "-mode",
"torch"],
            "console": "integratedTerminal"
        },
        ...
    ]
}
```

보고서 작성 요령 Write up

배경 이론을 바탕으로 여러분이 구현한 함수의 각 과정을 간단하게 설명하세요. 구현한 코드보다는 구현된 코드를 통해 신경망을 학습하는 과정에 대한 다양한 실험의 결과에 대한 분석과, 특히 성능의 개선으로 이어진 사항들을 중점적으로 정리하도록 하세요.
보고서의 형식은 제시된 양식을 따르고 제출 파일 형식은 pdf 파일로 제출하세요.

요구사항 Requirements / 평가기준 Rubric

기본 사항

- +45 pts: Implement process for training neural network. (`train_nn()`)
- +40 pts: Implement process for training neural network. (`class Model_pytorch 내 __init__(), forward()`)
- +15 pts: 보고서 Writeup with implementation and evaluation.
- -05*n pts: 본 과제의 설명 문서에 제시된 지시 사항을 제대로 따르지 않은 회수 n*5 만큼 감점

+α (추가 점수, 최대 10 점까지 부여):

- 최대 +4pts: Hyperparameter tuning 으로 최적 hyperparameter 도출한 경우 (Part1, Part2 각 2 점까지 부여)
- 최대 +5pts: Data augmentation 추가하여 인식 성능을 향상시킨 경우
- 최대 +5pts: Part2 의 CNN 모델을 다양하게 구현하여 인식 성능을 향상시킨 경우

디버깅 및 수행속도 Debugging and Computation Speed

버그는 코드 실행 시에 interpreter 가 알려주는 종류의 문제성 코드보다도, interpreter 상으로는 문제가 없지만 코드의 실행 결과에 오류가 있어서 그 오류로 인해 그 이후의 결과들도 오류가 누적되다 결국 전혀 엉뚱한 최종 결과가 도출되거나 코드가 돌다가 죽거나 crash 하는 문제가 발생하는 종류가 훨씬 빈번합니다. 어떤 통계로는 프로페셔널 개발자들의 코드에서도 1000 줄의 코드에 대해 평균적으로 15~50 개의 버그가 존재한다고 합니다(S. McConnell, “Code Complete: A Practical Handbook of Software Construction”). 따라서 디버깅은 코드 실행 중에 interpreter 가 알려주는 버그 외에도 중간 중간 결과가 정확하게 잘 계산되었는지도 확실하게 확인하면서 계산 결과에 문제가 없는지를 정확하게 확인하는 과정을 포함해야 한다는 점을 명심하기 바랍니다.

출처/참고자료 정보 Credits

[1] James Tompkin, “Project 4, CS1430 course description,” accessed 2021. https://browncsci1430.github.io/webpage/proj4_cnns/