인공지능 과제 (2019년 1학기)

<과제 2: MNIST 손글씨 숫자 인식을 위한 3Layer Neural Network 설계

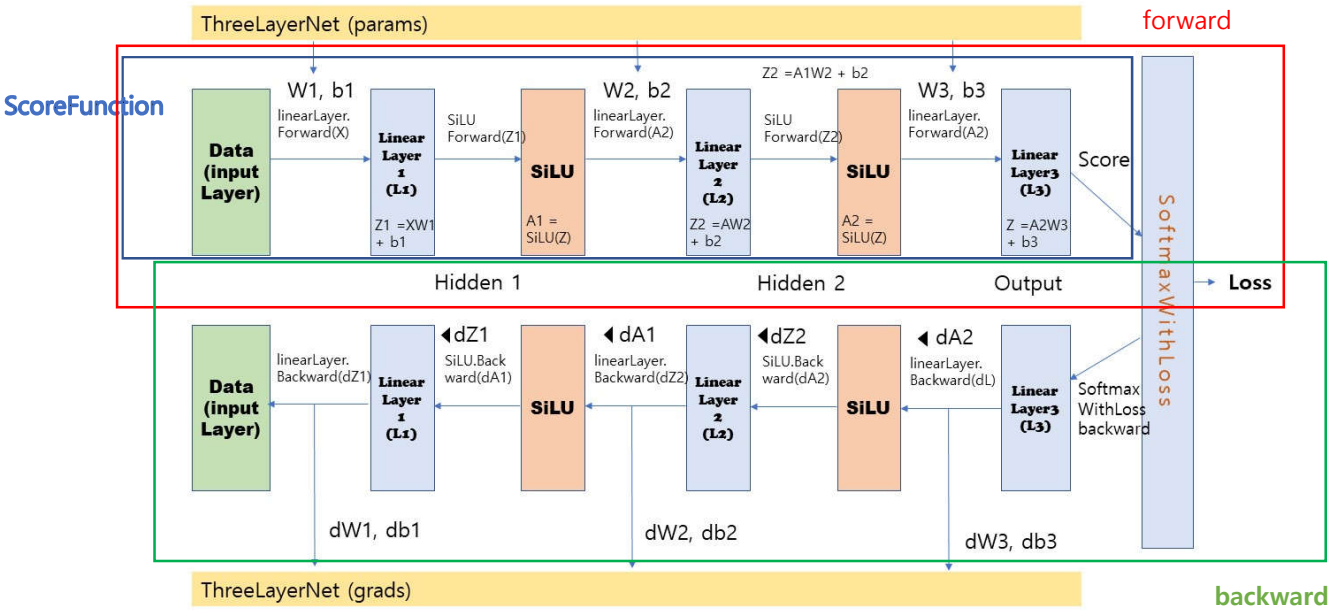
1. 개요

이 과제에서는 MNIST 데이터셋에 포함된 0부터 9까지 손글씨 이미지를 정확히 분류하기 위한 3-Layer Neural Network 를 디자인한다. Python으로 구현하고, 사용 가능한 라이브러리를 numpy, pillow로 제한한다.

2. MNIST 데이터셋

0부터 9까지의 손글씨 숫자 이미지로 구성된 이미지 집합으로, 본 과제에서는 10,000장(8000의 훈련 이미지와 2000의 테스트 이미지)으로 이루어져 있다. (10000, 785)의 형상의 minst.csv 데이터셋은 10000개의 벡터로 이루어져있다. 각 벡터는 1개의 레이블값(0~9) 및 28*28개의 회색조 픽셀값(0~255), 총 785(= 1+28*28)개의 원소로 구성된다.

3. 전체 디자인



4. 코드 (코드 65점 주석 25점)

4.1 train_test_split 함수

mnist.csv 데이터는 (10000, 785)의 형상을 가지고 있다. 이 것을 training에 사용할 train_data, train_label 과 test에 사용할 test_data, test_label 로 나누는 작업을 해야 한다. 현재 우리는 8000개의 이미지로 트레이닝을 하고 2000개의 이미지로 테스트를 하고 싶다. 요구사항에 맞추어 csv 데이터를 train_data (8000, 784), train_label(8000, 1) , test_data(2000, 784), test_label(2000, 1)로 나누어라.

4.2 one_hot_encoding 함수

본 과제는 이전 과제와 다르게 라벨값이 정수형으로 저장되어 있다. 그렇기에 정수형을 one_hot_label로 바꾸는 작업이 필요하다. input으로 train_label 혹은 test_label을 넣었을 때 거기에 맞는 one_hot_label 바꾸어주는 one_hot_encoding 함수를 구현하라.

4.3 Softmax 함수

본 과제는 Loss function으로 cross-entropy를 사용하는 뉴럴 네트워크이기에 Score 값을 확률값으로 바꾸어 주는 작업이 필요하다. 즉 데이터가 들어왔을 때 그 값을 확률값으로 바꾸어주는 softmax 함수를 구현하여라.

4.4 setParam_He 함수

본 과제에서는 activate function으로 relu와 유사한 SiLU를 사용할 것이다. 그렇기에 Weight 초깃값도 ReLU에 특화된 카이밍 히(kaiming He)의 Weight 초기값을 사용할 것이다. Neuronlist는 각 레이어의 뉴런의 개수가 들어온다. 즉 [input Layer neuron, hidden layer1 neuron, hidden layer2 neuron, output layer neuron]의 형식으로 들어온다. 여기서 W와 b 값을 레이어와 뉴런의 개수에 맞게 He의 방법으로 초기화해서 리턴 한다.

. 4.5 linearLayer 클래스

내적 연산을 통해 이전 레이어의 뉴런들에 Weight만큼의 가중치를 적용한 신호의 총합을 계산하는 forward 함수 및 미분값을 계산하는 backward 함수로 구성된다.

4.6 SiLU 클래스

Activate function으로 SiLU라는 activate 함수를 사용할 것이다. SiLU란 $A = x * \mathrm{sigmoid}(x)$로 나타나는 그래프로 ReLU와 그래프 형태가 유사하다. LinearLayer에서 들어온 값 z를 activation 하고 그 때의 Z값을 저장하는 forward 함수와 저장한 Z값으로 SiLU의 미분값을 구한 후 앞의 레이어에서 backward로 들어온 dActivation 값을 곱한 값 dZ를 출력하는 backward함수를 구현하여라

4.7 softmaxWithLoss 클래스

우리는 앞의 Layer들을 통해 구한 Score값을 가지고 Softmax와 CrossEntropy를 사용해 Loss 값을 구할 것이다. forward함수는 score를 softmax한 softmaxScore값과 one_hot_label 입력값을 저장하고 cross entropy에 기반한 loss값을 리턴하는 함수이다. backward함수는 loss에서 마지막 레이어의 바로 앞까지의 backpropagation한 미분 값을 리턴하는 함수이다.

4.8 ThreeLayerNet 클래스

위에 선언한 함수 혹은 클래스를 사용하여 ThreeLayerNet을 만드는 클래스이다. 먼저 setParam_He를 통해 self.params 라는 딕셔너리 객체에 W와 bias를 저장할 것이다. 그 후 self.layers 라는 OrderedDict() 객체에 linearLayer 객체와 SiLU객체를 저장할 것이다.

원래의 dictionary 객체는 key값과 value값을 저장하지만 list와 다르게 저장한 key의 순서대로 값을 리턴 할 수 없다. 그렇기에 OrderedDict()라는 입력한 데이터의 순서 또한 저장해주는 객체를 사용해 입력한 순서에 맞게 linearLayer 객체와 SiLU 객체를 불러와서 forward와 backward를 호출한다. 마지막으로 LastLayer에 SoftmaxWithLoss()객체를 저장한다.

4.8.1 scoreFunction(self, x)

scoreFunction은 저장한 레이어들에 데이터 x를 넣어 forward 시켜서 score를 얻는 함수이다. self.layers.values()는 layers에 저장된 value 값(self.layer['L1'], self.layers['SiLU1'] 등의 키 값에 대응하는 객체)를 불러오는 함수이다. 객체에 저장한 대로 불러오기에 각 layer 객체에 forward 함수

를 사용하면 score를 구할 수 있다.

4.8.2 forward(self, x, label)

위에 정의한 score함수를 사용해 self.LastLayer에 저장 되어있는 SoftmaxWithLoss()의 forward에 필요한 값을 넣어 Loss를 구하는 함수다.

4.8.3 accuracy(self, x, label)

입력한 데이터 X와 label을 통해 현재의 뉴럴네트워크의 정확도를 확인하는데 사용하는 함수이다.

4.8.4 backpropagation(self, x, label):

Chain-Rule에 입각한 dL/dW , dL/db 를 구하는 함수이다. SoftmaxWithLoss, SiLU, linearLayer에 각 클래스에 맞는 backward함수가 구현되어 있기에, forward에 했던 방식의 역순으로 backward를 구하면 된다. 다 구한 다음 각각의 dW, db값을 grads 라는 딕셔너리 객체에 저장한 후 그 값을 리턴한다.

4.8.5 gradientdescent(self, grads, learning_rate):

구한 dW, db 값이 저장된 grads를 사용해 learning_rate를 곱해 현재 뉴럴네트워크 객체의 W와 b를 업데이트하는 함수이다.

5. batchOptimization(dataset, ThreeLayerNet, learning_rate, epoch=1000)

dataset은 train_data와 one_hot_train, test_data와 one_hot_test 데이터를 모아둔 dictionary 객체이고, ThreeLayerNet 은 위의 선언한 클래스로 만든 객체이다. 한 epoch마다 8000개의 train_X를 한 번에 forward하고 backpropagation을 해서 Loss를 구하고, gradientdescent를 사용해 W와 b를 업데이트 한다. 10번마다 train_accuracy와 test_accuracy loss를 보여주고 각각의 train_acc_list, test_acc_list, Loss_list에 append 시킨다. 리턴값으론 ThreeLayerNet, train_acc_list, test_acc_list, Loss_list 를 리턴한다.

6.minibatch_Optimization(dataset, ThreeLayerNet, learning_rate, epoch=100, batch_size=1000)

Minibatch를 통한 forward, backpropagation, Loss를 구하는 함수이다. 먼저 train_X와 one_hot_train을 random하게 섞는다.(이 때 섞인 train_X와 one_hot_train간의 관계가 달라지면 안 된다. 즉 train_X[1] => train_X[5]로 갔으면 one_hot_train[1] =>one_hot_train[5]로 옮겨가야한다.) random 하게 섞는 것은 np.random.shuffle을 사용하면 편하다.

랜덤하게 셔플된 데이터를 batch단위로 나눠서 forward와 backpropagation, gradientdescent를 하고, 그렇게 랜덤하게 셔플된 모든 데이터가 forward와 backpropagation이 끝났을 때를 1 epoch이라 한다. 예를 들어 10개의 데이터를 2개의 batch 사이즈로 mini batch를 진행하면 0~2, 2~4, 4~6, 6~8, 8~10 순으로 forward와 backpropagation, gradientdescent를 진행한 것이 1 epoch이 된다.

1 epoch이 끝난 다음엔 다시 데이터를 랜덤하게 섞고 위의 방법을 반복한다. 5 epoch당 Loss, train_acc, test_acc을 출력하고 그 것을 각각 Loss_list, train_acc_list, test_acc_list에 append시킨다. 리턴값으론 ThreeLayerNet, train_acc_list, test_acc_list, Loss_list 를 리턴한다.

7. dropout_use_batchOptimization

Dropout을 사용한 Optimization이다. Dropout 알고리즘은 hidden Layer의 뉴런을 죽임으로써 레귤라이제이션을 하는 함수이다. kill_n_h1, kill_n_h2은 드롭아웃으로 뉴런을 죽이는 비율이다. 구현하는 방식은 마음대로 해도 된다. batchoOptimization과 똑같이 한 epoch마다 8000개의 train_X를 한 번에 forward하고 backpropagation을 해서 Loss를 구하고, gradientdescent를 사용해 W와 b를 업데이트 한다. 10번마다 train_accuracy와 test_accuracy loss를 보여주고 각각의 train_acc_list, test_acc_list, Loss_list에 append 시킨다. 리턴값으론 ThreeLayerNet, train_acc_list, test_acc_list, Loss_list 를 리턴한다.

4. 제출물 10점

(1) 보고서 내용

2차과제의 코드와 주석

Jupyter notebook file은 클래스넷으로 .py 파일은 리눅스로 제출

(2) 제출 마감 시간 및 장소

- ✓ 제출 마감 시간 : 5월 20일 월요일 오후 11시
- ✓ 장소 : jupyter파일은 py 파일로 변환 후 리눅스에 submit으로 제출. 자세한 제출 방법은 다음주 중으로 다시 공지. 개인 보고서와 jupyter notebook File은 클래스넷에 5월 20일 (월요일) 업로드 그리고 하드 카피는 5월 23일 목요일 수업시간 제출

5. 채점 및 감점 기준

(1) 만점 기준

- ✓ 코드가 정상적으로 잘 돌아가는가? (40점)
- ✓ Optimization 결과 후 나오는 Accuracy가 조교가 지정한 범위 안에 들어가는가(20점)
- ✓ 주석이 그 함수를 설명하는데 적절한가? (30점)
- ✓ 보고서를 제 때 제출했는가? (10점)

(2) 감점 사항

- ✓ 클래스넷 제출 기한(4월 16일 화요일 0시)을 넘길 시 0점
- ✓ 부정행위 발견 시 관련 학생 모두 F 학점 처리함
- ✓ 제출한 보고서의 코드와 제출한 코드가 다를 시 최대 50점 감점
- ✓ 하드카피와 제출한 보고서가 다를 시 최대 50점 감점
- ✓ 제공한 라이브러리가 아닌 다른 라이브러리를 사용했을 시 0점
- ✓ 코드가 작동안할 시 어떤 부분에서 작동이 안되는지 보고서에 정확히 기재. 보고서에 기재 없이 코드가 안 돌아갈 경우 0점

(3) 질문은 클래스넷 게시판과 평일 월, 금 오후 3시~5시 장소는 719-A호로 방문바랍니다. 또한 5월 17(금)~19일(일) kwon2019ai@gmail.com 으로도 질문을 받습니다.

참고자료 1. SiLU(Swish) A = x*sigmoid(x)