

**딥러닝 hw3 과제**

**과목명 딥러닝**

**담당교수 장익범교수님**

**제출일 20231113**

**전공 컴퓨터전자시스템**

**학번 201904458**

**이름 이준용**

**1. 직접 정의한 MLP 모델에 대해 자신의 언어로 설명하기.**

클래스에 정의된 MLP 모델은 MLP이미지 분류 작업을 위해 설계된 신경망입니다. 이는 컨벌루션 레이어와 완전 연결(선형) 레이어의 조합으로 구성됩니다. 각 구성 요소에 대한 분석은 다음과 같습니다.

* Convolutional Layers - Conv2d

self.conv1과 self.conv2는 컨볼루셔널 레이어 입니다.

이러한 레이어는 입력 이미지에서 특징을 추출하는 데 사용됩니다. 컨벌루션 레이어는 입력에 여러 필터를 적용하여 이미지 안에서 픽셀의 값이 갑자기 변하는 곳, 텍스처 등과 같은 spatial 특징을 잡아냅니다.

self.conv1은 3채널 이미지(RGB)를 입력으로 가져와서 6개의 특징 맵을 출력하는 동안 self.conv2가 6개의 특징 맵을 가져와서 16개를 출력합니다.

* Pooling Layer - MaxPool2d

self.pool은 max 풀링 계층입니다.

풀링 레이어는 입력 볼륨의 공간적 크기(높이 및 너비)를 줄여 계산을 보다 쉽게 ​​관리하고 매개변수 수를 줄이고 다음 layer로 넘겨줍니다.

* Fully Connected (Linear) Layers

self.fc1, self.fc2, 및 self.fc3은 Fully Connected (Linear) Layers입니다.

컨벌루션 및 풀링 레이어 연산을 통해 특징을 추출한 후 네트워크는 이러한 Fully Connected (Linear) Layers를 사용하여 추출된 특징을 기반으로 이미지를 분류합니다.

마지막 레이어인 self.fc3은 CIFAR-10 데이터세트의 10개 클래스에 해당하는 10개의 값을 출력합니다.

* Activation Functions - ReLU함수

모델은 각 컨벌루션 레이어와 처음 두 개의 선형 레이어 후에 ReLU(Rectified Linear Unit) 활성화 함수를 사용합니다.

ReLU는 모델에 비선형성을 도입하여 더 복잡한 패턴을 학습할 수 있습니다.

* Flattening 작업

컨벌루션 레이어의 출력을 완전 연결 레이어로 전달하기 전에 데이터가 평면화됩니다. (2D feature map에서 단일 출력 벡터로 변환)

* Forward pass

주어진 레이어의 가중치들로 다음 레이어의 노드 값을 편미분해서 저장해준다. 이런 편미분은 input레이어의 가중치부터 비용함수에 도달할 때까지 계속된다.

이 forward방법에서 입력 데이터는 다음 순서로 네트워크를 통해 흐릅니다.

1. conv1입력 데이터는 ReLU 활성화 함수를 거쳐 전달
2. 그런 다음 결과가 pool레이어를 통과
3. conv2에서도 동일한 시퀀스(Convolution -> ReLU -> Pooling)과정이 발생
4. 다음 출력이 Flattening 작업수행
5. Flatten된출력은 ReLU 활성화를 통해 두 개의 선형 레이어(fc1, fc2)를 통과
6. Fc2 layer 통과후에 fc3를 통과해서 최종 분류 스코어를 도출

이 MLP 모델은 10개 클래스의 60,000개의 32x32 컬러 이미지(클래스당 6,000개의 이미지)로 구성된 CIFAR-10 데이터 세트를 사용하여 학습되었습니다. 모델은 이러한 이미지를 10가지 카테고리(예: '비행기', '자동차', '새' 등) 중 하나로 분류하는 것을 목표로 합니다.

요약하면, MLP 모델은 2d 이미지 특징 추출을 위한 컨벌루션 레이어와 분류를 위한 완전 연결 레이어를 결합하여 이미지 분류 작업에 모델이 됩니다. ReLU 활성화 함수 및 풀링 레이어를 사용하면 계산 효율성을 관리하면서 데이터의 복잡한 패턴을 학습하는 데 도움이 됩니다.

**2. Hyperparameter를 바꾸며 도출된 성능 그래프를 첨부하고,**

**자신의 코드를 예시로 그래프의 결과 설명하기.**

텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

그림 1. Learning Rate: 0.001 Optimizer: Adagrad Batch Size: 4

optimizer = optim.Adagrad(model.parameters(), lr=0.001)

Loss가 잘 fitting되는 모습이고 AUC는 44%정도이다. 잘 학습되었지만 정확도는 절반 수준이다. 좋은 성능을 기대하기에는 어려움이 있을 것으로 예상된다.

텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

그림 2. Learning Rate: 0.01, momentum=0.9 Optimizer: SGD with Momentum 0.9 Batch Size: 32

optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

Overfitting 된 모습이다. 모멘텀을 사용했음에도 불구하고 훈련이 잘 되지 못하였다. 학습률과 배치 사이즈를 다시 조정해서 학습해보았지만 쉽게 overfitting을 해결하지 못했다.

텍스트, 도표, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

그림 3. Learning Rate: 0.0001 Optimizer: Adam Batch Size: 16

optimizer = optim.Adam(model.parameters(), lr=0.0001)

잘 학습된 모습이다. 살짝 overfitting된 느낌이 있지만 정확도에서는 65%수준에 해당하여 위의 adagrad와 SGD보다 훨씬 더 좋은 성능을 예상해 볼 수 있을 것이다.

**3. 코드에 사용된 Loss function과 optimizer에 대해 본인이 이해한대로 설명하고,**

**선택한 이유와 그 밖에 다른 종류에 대해 설명하고 어떤 상황에 사용하는지에 대해**

**설명하기. (최소 2종류 이상씩 조사하고, 코드로 예시를 들어도 됩니다.)**

* 본 모델에 사용한 손실 함수: 크로스 엔트로피 손실

코드에서 사용되는 손실 함수는 nn.CrossEntropyLossPyTorch에서와 같이 구현된 Cross-Entropy Loss입니다.

교차 엔트로피 손실은 분류 작업, 특히 여러 클래스의 경우 선호됩니다. 출력이 0과 1 사이의 확률 값인 분류 모델의 성능을 측정합니다. 두 확률 분포, 즉 실제 값과 예측 값 간의 차이를 계산합니다. 목표는 이 차이를 최소화하는 것입니다. 즉, 모델의 예측은 실제 라벨과 최대한 가까워야 합니다.

* 크로스 엔트로피 손실 선택한 이유

교차 엔트로피 손실은 이미지를 여러 클래스 중 하나로 분류해야 하는 이미지 분류(CIFAR-10의 경우)와 같은 작업에 특히 효과적입니다. 다중 클래스 문제에 적합하며 모델이 확률 분포를 출력할 때(예: 최종 레이어에서 softmax사용) 잘 작동합니다.

* 이외의 손실함수: MSE(평균 제곱 오류)

MSE 는 회귀 작업에 일반적입니다.

추정값과 실제값 사이의 평균 제곱 차이를 계산합니다.

* 이외의 손실함수: Binary Cross-Entropy

이진 분류 작업에 사용됩니다. 이는 두 클래스에 대한 교차 엔트로피 손실의 특별한 경우입니다.

* optimizer: adam

구성 중 하나에서 사용한 최적화 프로그램은 optim.AdamAdaptive Moment Estimation을 나타내는 Adam( )입니다.

Adam은 다른 두 가지 인기 있는 최적화 알고리즘인 Momentum(과거 경사를 고려하여 경사하강법 알고리즘을 가속화함)과 RMSprop(최근 경사의 크기를 사용하여 경사를 정규화함)의 아이디어를 결합합니다.

각 매개변수에 대한 적응형 학습률을 계산하여 더 빠른 수렴을 돕습니다.

* Adam 선택한 이유:

Adam은 희소 기울기 처리의 효율성과 이미지 분류 작업과 같은 다양한 문제 및 데이터 유형에 대한 적응성 때문에 종종 선택됩니다.

범용 최적화를 위한 좋은 기본 선택입니다.

대안:

* 이외의 optimizer: SGD(확률적 경사하강법)

한 번에 하나의 훈련 예제를 사용하여 매개변수를 업데이트합니다. 이는 학습 프로세스를 복잡하게 만들 수 있지만 로컬 최소값을 벗어나는 데 도움이 됩니다. SGD 는 데이터가 많고 문제가 상대적으로 간단할 때 좋은 선택입니다.

* 이외의 optimizer: Adagrad

학습 속도를 매개변수에 맞게 조정하여 자주 발생하지 않는 매개변수에 대해 더 큰 업데이트를 수행하고 빈번한 매개변수에 대해 더 작은 업데이트를 수행합니다. 희소 데이터에 유용합니다.

* 이외의 optimizer: RMSprop

급격하게 감소하는 학습 속도를 다루는 Adagrad의 확장입니다. 텍스트와 같은 희소 데이터를 처리할 때 자주 사용됩니다.

요약하자면, 손실 함수와 최적화 프로그램의 선택은 해결하려는 특정 문제와 데이터 특성에 따라 달라집니다. 교차 엔트로피 손실 및 Adam 선택은 프로젝트의 CIFAR-10 분류 작업에 적합하다고 판단했습니다.

**4. 학습된 결과에 대해 학습이 된 결과인지 서술하고, 학습된 결과라면 왜 학습이 되었다고 생각하는지 작성하고, 학습이 제대로 되지 않은 것 같다면 왜 그렇게 생각하는지 작성 후, 해당 결과로 제출한 이유에 대해서 설명. (예시. 학습이 진행되는 중에 epoch당 mini batch 값으로 도출된 값 첨부)**

* **Adagrad 잘 학습되었다고 판단**

Epoch당 mini batch 값이 잘 줄어드는 모습을 볼 수 있었다. 또한 그래프에서도 오차 값이 잘 감소하는 모습이고 정확도가 증가하는 모습이다. 정확도는 50% 이하이지만 fitting이 잘되었다고 판단하여 다음과 같은 결과값을 제출함

텍스트, 문서, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

* **SGD 잘 학습 안되었다고 판단**

텍스트, 종이, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

에폭과 배치사이즈, 학습률를 다시 조정해야될 것 같다. 에폭 4부터 오버피팅이 심하게 일어났다. Dropout를 사용하거나 은닉층을 더 줄인다거나(모델을 간소화) 해서 이전보다 더 좋은 학습을 기대해 볼 수 있을 것 같다.

이후로 여러 번 실험했지만 MLP에서의 이미지 학습에 있어서 SGD의 하이퍼 파라미터는 찾기 좀 힘들었다.

다른 옵티마이저를 사용할 것을 권장하는 의미에서 위와 같은 결과를 제출함.

* Adam 약간의 overfitting 느낌이 있지만 잘 학습된다고 판단

Epoch당 mini batch 값이 잘 줄어드는 모습을 볼 수 있었다. 또한 그래프에서도 오차 값이 잘 감소하는 모습이고 정확도가 증가하는 모습이다. 배치 사이즈와 학습률을 살짝 조정했다면 더 좋은 결과가 나오는 것을 기대해 볼 수 있겠다. 3개의 옵티마이저 중에 제일 높은 정확도를 보여주었기에 이 결과를 선택하였기 이와 같은 결과를 제출함.

텍스트, 폰트, 종이, 문서이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

**5. 과제를 진행하며 어려웠던 점 서술**

**(어려워던 부분이 없다면, 본인이 이해한 내용을 기반으로 3줄 요약해서 작성)**

평소에 많이 사용하는 옵티마이저인 SGD에서 하이퍼파라미터를 찾기가 굉장히 어려웠고, 오버피팅을 해결하기 위해 시간도 많이 걸렸다. conv, pooling, fully connected layer을 어떻게 설정해야 되는지 고민을 많이 했다.

**6. 과제를 진행하며 배운 점과 느낀 점 서술**

**(5번의 3줄 요약과 중복도 가능하며 중복된 내용이라도 작성하시길 바랍니다.**

**또한, 느낀 점은 반드시 다른 내용으로 작성해야 합니다.)**

Hw2에서 과제 했던 것을 베이스로 하여 배운 내용을 토대로 d2l에서 소개된 mlp 코드 보면서 DL\_MLP모델을 구현하였다. 그 과정에서 여러 MLP의 수행 과정 컨블루션 연산과 풀링 연산 과정을 알게 되었고, 다양한 손실함수와 옵티마이저의 동작과정을 알게 되었습니다. AI 관련해서 입문한지 이제 2개월 되어서 두번째 과제까지는 pytorch 문법이나 훈련 과정부분에서 헷갈렸지만 이번 과제인 MLP에 대해 자세하게 공부해보는 시간이 되었기에 다른 사람에게 설명할 수 있을 정도로 성장할 수 있었던 것 같습니다.