DeepLearning Assignment3 보고서

소프트웨어학부 20201815 강민석

Assignment 1

목적

한 타임스텝에서 RNN이 어떻게 hidden state를 갱신하는지 직접 구현하는 것이 목적이다.

내용

현재 입력과 이전 hidden state에 각각 가중치를 곱하고 편향을 더해 선형 결합을 만든 뒤,
tanh 활성 함수를 적용하여 다음 hidden state를 계산하였다.
이때, 이후 backward 계산을 위해 입력, 가중치, 출력 hidden state 등을 cache에 함께 저장하였다.

검증

제공된 고정 입력과 정답 hidden state와의 상대 오차를 비교했을 때, 약 1e-8 수준으로 forward 구현이 정확함을 확인하였다.

Assignment 2

목적

단일 타임스텝 RNN에 대한 역전파를 구현하여, 이후 시퀀스 전체에 대한 BPTT의 기반을 마련하는 것이 목표이다.

내용

다음 hidden state에 대한 손실의 gradient가 주어졌을 때,
tanh의 도함수(출력값을 이용하여 $1 - \tanh^2$ 형태)를 활용해 선형 결합에 대한 gradient를 구하고,
이를 다시 입력, 이전 hidden state, 두 종류 가중치, 편향에 대한 gradient로 분해하였다.

검증

입력, 이전 hidden, 가중치, 편향에 대해 수치 미분과 비교했을 때, 상대 오차가 모두 1e-8 수준 이하로 나타나 역전파 구현이 올바름을 확인하였다.

Assignment 3

목적
단일 스텝 RNN을 이용하여, 시퀀스 전체(T timesteps)에 대한 hidden state를 계산하는 함수를 구현하는 것이 목표이다.

내용
초기 hidden state에서 시작해 시간축을 따라 순차적으로 단일 스텝 forward를 호출하며, 각 timestep의 hidden state를 (배치, 시퀀스 길이, hidden 차원) 형태의 배열에 저장하였다.
또한, 각 timestep에서 나온 cache를 리스트 형태로 모아 backward에 사용할 수 있도록 구성하였다.

검증
과제에서 제공한 기대 hidden state 시퀀스와 비교했을 때 상대 오차가 약 1e-7 수준으로 측정되어 전체 시퀀스 forward가 정확함을 확인하였다.

Assignment 4

목적
RNN 전체에 대해 BPTT(Back-Propagation Through Time)를 수행하는 backward 함수를 구현하는 것이 목표이다.

내용
마지막 timestep에서 시작하여 역순으로 이동하면서,
각 timestep마다 "해당 timestep의 upstream gradient + 다음 timestep에서 전달된 hidden gradient"를 합산한 후 단일 스텝 backward를 호출하였다.
각 timestep에서 나온 입력 gradient를 모아 (배치, 시퀀스 길이, 입력 차원) 형태로 저장하고,
가중치와 편향 gradient는 전체 시퀀스에 걸쳐 누적하였으며, 마지막에 남은 hidden gradient를 초기 hidden state에 대한 gradient로 사용하였다.

검증
입력, 초기 hidden state, 두 가중치, 편향에 대해 수치 미분과 비교했을 때,
상대 오차가 대략 1e-6 이하로 나와 BPTT 구현이 정상적으로 동작함을 확인하였다.

Assignment 5

목적

정수 인덱스로 표현된 단어들을 학습 가능한 연속 벡터 공간으로 매핑하는 Word Embedding 레이어의 forward와 backward를 구현하는 것이 목표이다.

내용

- Forward에서는 (배치, 시퀀스 길이) 형태의 정수 인덱스를 임베딩 행렬의 행 인덱스로 사용하여,
  (배치, 시퀀스 길이, 임베딩 차원)의 단어 벡터 시퀀스를 생성하였다.
- Backward에서는 단어 인덱스에 대해 직접 미분할 수 없기 때문에, 임베딩 가중치에 대해서만 gradient를 계산하였다.
  같은 단어가 여러 위치에서 등장할 수 있으므로, 해당 단어 인덱스에 대해 gradient를 누적하는 방식으로 구현하였다.

검증

Forward 결과는 정답과 비교했을 때 오차가 1e-8 수준이었고,
임베딩 가중치 gradient는 수치 미분과 비교했을 때 오차가 1e-11 수준으로 매우 작게 나타났다.

Assignment 6

목적

지금까지 구현한 레이어들을 조합하여,
이미지 feature와 캡션을 입력받았을 때 손실과 모든 파라미터에 대한 gradient를 계산하는 CaptioningRNN의 학습 루프를 완성하는 것이 목표이다.

내용

1. 캡션을 입력용(captions_in)과 정답용(captions_out)으로 한 칸씩 시프트하여 분리하였다.
2. 이미지 feature를 선형 변환하여 초기 hidden state를 계산하였다.
3. 입력 캡션을 임베딩 벡터 시퀀스로 변환하였다.
4. cell_type이 'rnn'일 때는 RNN 시퀀스 함수를 이용해 hidden state 시퀀스를 계산하였다.
5. hidden state를 temporal affine 레이어를 통해 vocabulary score로 변환하였다.
6. temporal softmax loss와 <NULL> 마스크를 사용하여 최종 손실을 계산하였다.

7. backward에서는 위 과정을 역순으로 따라가며 vocabulary projection, RNN, 임베딩, 이미지 projection 순서로 gradient를 계산하였다.

검증

모든 파라미터를 고정 값으로 설정한 뒤 loss를 계산해 제공된 기대값과 비교한 결과 거의 일치했으며,
각 파라미터에 대한 gradient도 수치 미분과 비교했을 때 상대 오차가 약 1e-6 수준으로 나와 전체 모델의 forward/backward 구현이 올바른 것을 확인하였다.

Assignment 7

목적

1. 학습된 모델이 **테스트 시 이미지 feature만으로 캡션을 생성**할 수 있도록, 샘플링 함수를 구현한다.
2. 작은 데이터셋에 대해 모델을 과적합시켜 구현이 제대로 동작하는지 검증한다.

내용

- 샘플링 함수에서는 이미지 feature를 초기 hidden state로 변환하고,
  첫 입력으로 <START> 토큰을 사용한 뒤,
  매 timestep마다
    1. 이전 단어 임베딩
    2. 이전 hidden state를 입력으로 RNN 한 스텝을 수행하여 새로운 hidden state를 얻고
    3. vocabulary score의 argmax를 취해 다음 단어를 선택하는 **greedy decoding**을 수행하였다.
- 작은 COCO train subset(예: 50개 이미지)만 사용하여 CaptioningRNN을 학습했고, Adam optimizer와 적절한 learning rate, epoch 수를 설정하여 training loss가 0.1 이하까지 떨어지는 **강한 과적합**을 확인하였다.

결과 및 관찰

- train 데이터에 대해서는 생성된 캡션이 정답 캡션과 구조적으로 유사하거나 자연스러운 문장이 많이 나왔고,
- val 데이터에서는 여전히 어색한 문장이 자주 등장하여,

모델이 작은 데이터에 과도하게 특화되었음을 확인할 수 있었다.

이를 통해 구현한 RNN 기반 이미지 캡셔닝 모델이 학습·추론 모두 정상적으로 동작함을 확인하였다.