

Deep Learning Frequent Question & Answers

Q: yolo 알고리즘은 그리드별로 cnn 을 적용시킨다고 이해했는데, 그리드보다 사이즈가 큰 객체의 경우 어떻게 분류할 수 있는건가요? 또한 바운딩 박스가 그리드보다 큰 경우는 그리드 바깥 이미지의 정보를 사용하지 않고 어떻게 추론할 수 있는 건가요?

A: 그리드들은 전체 이미지를 정확히 partition 하지 않고 다소 overlapping 하는 형태로 구성하는 경우가 많습니다. 또한 convolution net 의 특성상 학습한 온전한 (전체) 객체 이미지 뿐만 아니라, 상당 부분이 존재하는 부분적 객체 이미지도 인식할 수 있습니다. 그러므로 그리드보다 사이즈가 큰 이미지나 바운딩 박스를 찾아낼 수도 있습니다. (혹시 이해가 안되면 convolution-pooling-마지막 regression 과정을 예를 들어 계산하면서 면밀히 따져 보면 이해할 수 있습니다.)

Q: Valid and Same Convolution 페이지

same 의 경우 패딩을 적용하여, input 과 output 의 size 를 같게 하는 것이라고 설명 해주셨는데, 이것에 Stride 도 고려되었는지 궁금합니다.

A: stride 를 고려한 것은 아닙니다. stride = 1 일때 same 이 의미가 있겠지요.

그리고, 이때 $n+2p-f+1 = n$ 가 되므로 $p = (f-1)/2$ 가 성립합니다. 따라서 f 가 홀수이면 p 가 자연수로 딱 떨어지므로 f 가 홀수인 filter 를 사용하는 것이 좋겠네요.

Q: 즉, Padding = Same, Stride = 2, Filter size = 3 이라고 한다면 식 " $\text{floor}((n+2p-f)/s+1) = n$ "에 의해서 $p = (n+1) / 2$ 로 계산되는 것이 맞나요?

A: 계산은 그렇게 될 수 있겠지만 (위 답변에 의해) 패딩이 n 값에 대해 정해지는 것은 현실적으로 의미가 없겠지요.

Q: LeNet5의 Parameter 수를 계산하는 부분에서 왜 입력 값의 채널 (이전 레이어의 출력 값)을 반영하지 않는지 궁금합니다. Summary of Notation 부분에서 Weight 개수를 계산할 때 (필터의 너비 * 필터의 높이 * 입력 값의 채널 수 * 필터의 개수) 로 계산하는 것으로 설명해주셨는데, LeNet5 의 Parameter 수 계산하는 예시에서는 (필터의 너비 * 필터의 높이 * 필터의 개수) 로 계산해 주셨습니다.

Summary of Notation 에서의 방식 처럼 계산을 해야할 것 같은데, 계산 방식에서 차이가 발생한 이유를 설명 해 주실 수 있을까요?

A: standard convolution 에서는 지적해 준 대로 Weight 개수를 계산할 때 (필터의 너비 * 필터의 높이 * 입력 값의 채널 수 * 필터의 개수) 로 계산하는 것이 맞습니다.

따라서, 208 --> 608, 416 --> 3216 이 되는 것이 이와 일관되는 설명입니다.

다만, LeNet5 의 Parameter 수를 계산하는 예시에서는 입력(이전층의) 각 채널에 대하여 평면적으로 convolution 을 수행하는 예(depthwise convolution 의 한 종류)로 든 것입니다. 즉, 이 경우 개별 convolution 을 위한 입력 채널 수는 그냥 1로 계산하는 것입니다.

실제로 convolution 은 standard convolution 외에도 depthwise convolution, pointwise convolution, grouped convolution 등의 다양한 방식이 있으며, 각각의 convolution filter 깊이(입력 채널 수)는 다양한 값을 가지게 됩니다.

Q: Neural Network Example (LeNet5) 페이지

지난번 질문 때 Parameter 수를 (필터의 너비 * 필터의 높이 * 입력 값의 채널 수 * 필터의 개수) 와 같이 계산하지 않은 이유가 Depthwise Convolution 라고 설명해주셨습니다.

하지만 보내주신 자료에 Depthwise Convolution 은 input 과 output 의 채널 수가 같아야 한다는 설명이 있어, 혼란스러운 상황입니다.

이 부분에 대해서 조금 더 자세히 설명 해 주실 수 있을까요?

A: Depthwise Convolution 은 각 채널 안에서만 필터를 적용시키므로 input, output 채널 수가 같은 것이 기본 방식이라고 하는 것 같은데, 예를 들어 10 개의 채널이 있다면 각 i 번째 채널의 convolution 은 i 번째 채널로 출력되는 것을 이해하면 되겠습니다.

Q: 혹자는 아래와 같이 수정되어야 한다고 제안하기도 하는데, 어떻게 이해하는 것이 좋을지 잘 모르겠습니다.

SL.No		Activation Shape	Activation Size	# Parameters
1.	Input Layer:	(32, 32, 3)	3072	0
2.	CONV1 (f=5, s=1)	(28, 28, 8)	6272	608
3.	POOL1	(14, 14, 8)	1568	0
4.	CONV2 (f=5, s=1)	(10, 10, 16)	1600	3216
5.	POOL2	(5, 5, 16)	400	0
6.	FC3	(120, 1)	120	48120
7.	FC4	(84, 1)	84	10164
8.	Softmax	(10, 1)	10	850

A: 이미 지난 번 메일에서 "standard convolution"에서는 지적해 준 대로 Weight 개수를 계산할 때 (필터의 너비 * 필터의 높이 * 입력 값의 채널 수 * 필터의 개수) 로 계산하는 것이 맞습니다. 따라서, 208 --> 608, 416 --> 3216 이 되는 것이 이와 일관되는 설명입니다."라고 답변한 바 있습니다. 제안해 준 수정 표는 이와 일치합니다. (그냥 복잡하게 생각할 필요없이 수정 표와 같이 이해해도 맞습니다.)

Q: 10_Convolutional Neural Networks 에 19pg 에 보면 # parameter 구하는 것에 첫번째 CONV1 에서 f=5 면 # parameter 가 $5 * 5 * \text{이전 filtersize} 3 * \text{filter 수 } 8 + 8 = 608$ 으로 계산되어야 하는거 같습니다.. 아래부분도 다 이전 필터사이즈를 무시하게 되는데 이전 필터사이즈도 고려해서 weight parameter 갯수를 구해야 하는게 아닌지 궁금합니다.(CONV2 = $5*5*8*16+16$ 개)

A: 그렇습니다. standard convolution에서는 지적해 준 대로 Weight 개수를 계산할 때 (필터의 너비 * 필터의 높이 * 입력 값의 채널 수 * 필터의 개수) 로 계산하는 것이 맞습니다. 따라서, 강의노트 표에서 208 --> 608, 416 --> 3216 이 되는 것이 이와 일관되는 설명입니다. (수정하여 이해하면 됩니다.)

다만, 강의노트의 parameter 수를 계산하는 예시(표)에서는 입력(이전층의) 각 채널에 대하여 평면적으로 convolution 을 수행하는 예(depthwise convolution 의 한 종류)로 든 것입니다. 즉, 이 경우 개별 convolution 을 위한 입력 채널 수는 그냥 1 로 계산하는 것입니다.

실제로 convolution 은 standard convolution 외에도 depthwise convolution, pointwise convolution, grouped convolution 등의 다양한 방식이 있으며, 각각의 convolution filter 깊이(입력 채널 수)는 다양한 값을 가질 수 있게 됩니다.

Q: Word Embedding 8 페이지에 skip-gram 에 quick 을 기준으로 window size = 1 일때 (quick, the), (quick, brown)이 나오는데 그 아래 그림에 quick 을 embedding 했을때 output softmax 에서 the 만 1 로 만드는데 the 를 1 로만드는 training 과 brown 을 1 로만드는 training 을 각각 하게되는건가요? 아니면 the 와 brown 을 동시에 확률을 높이는 방법으로 학습하는건가요? 라는 질문을 실습시간에 했었는데 따로 학습하게 된다는 답변을 받았습니다. 그러면 문장의 단어를 기준으로 정방향 단어를 예측하는 Weight 와 역방향의 단어를 예측하는 weight Matrix 가 각각 존재하게 되는건가요?

A: 워드 임베딩은 정방향 또는 역방향 단어 예측을 구분하여 학습하지 않습니다. 단순히 어떤 단어의 주변에 다른 단어가 (방향과 관계 없이) 나타날 분포를 학습하는 모델로 이해하면 됩니다. 결론적으로 가중치 행렬을 따로 가지지 않습니다.

Q: LSTM 에서는 RNN 에서의 hidden state 의 개념이 Cell 과 hidden state 2 개로 나뉘지는건가요?

A: 그렇게 볼 수 있습니다.

Q: 그리고 forget gate 가 있으면 결국 편미분값이 0 과 1 사이가 되어 계속 곱하다보면 여전히 vanishing gradient 문제가 생기지 않나요?

A: 그렇습니다. 모든 gate 는 출력값이 0 일 때, 연결이 단절되는 것이므로 그렇지 않을 때는 여전히 vanishing gradient 문제가 생길 수 있습니다. 다만, dependency 가 거의 없는 연결은 (거의) 단절시키고 반대인 경우는 주로 c 값을 통하여 바로(directly) 연결되는 역할을 수행하므로 RNN 모델의 성능(effectiveness)를 증대시켜주는 것이 중요한 기능입니다.

Q: Lec14 에서 12pg 에 Qtable memory 나타내는 부분에서 왜 지수함수 형태로 메모리가 필요한지 이해가 되지않습니다.. 2^n 이 아니라 n^2 아닌가요??

A: binary image 라고 가정하고 n 이 픽셀 개수라면 image(상태)의 전체 경우의 수는 2^n 이라는 것입니다. 따라서, Qtable 로 관리한다면 table (state) entry 가 2^n 이 되지요. 매우 비효율적. 그래서, Qtable 의 approximator 로서 딥러닝 모델을 쓰자는 것이지요.

Q: 이 부분은 교수님이 여러번 설명 하신거 같은데 여러번 들어봐도 잘 이해가 가지 않는데요, Double Q learning 에서 action selection 과 evaluation 이 다른 DQN 을 써야된다고 말씀하셨는데 일반적인 딥러닝 모델 학습할때도 같은 모델을 사용해서 학습하고 그 모델로 평가하게 되는데 왜 reinforcement learning 에서는 두개를 나눠서 하게 되나요? 말씀하신게 잘 이해가 안갑니다.. Minibatch 를 이용해서 저장된 state 와 action 을 불러오는 형태로 학습하면 선택된 action 에 대해서만 학습하는게 아니라 모든 action 과 state 에 대해 학습하는게 아닌가요?

A: reinforcement learning 에서는 학습데이터가 미리 주어지는 것이 아니라, 실제 모델을 환경에서 수행함으로써 얻어내고 있습니다. 그러므로, 하나의 DQN 이 선택한 action 을 이용하여 reward 를 받는 과정을 반복하면, 학습 데이터의 "sampling bias"가 생긴다는 것입니다. (Q 값이 크지 않은 action 의 사례는 sampling 되지 않게 된다. 실제 사람도 자신이 좋아하는 또는 잘하는 것만 사례로 학습해 보게 되어 이에 대한 경험만 많아지는 경향과 비슷.) 일반적인 딥러닝 모델은 (통계 sampling 기법적으로 보았을때) 최대한 사례가 고르게 분포하도록 (학습 모델과는 완전 독립적으로) 학습데이터를 수집하는 것이 중요하지요. 또한, 여기서 평가(evaluation)는 test 나 validation 을 의미하는 것이 아니므로, 대략적으로 이야기하자면 어떤 state 에 대한 action 을 선택하여 학습 데이터를 수집하는데 사용하는 DQN 모델과 그 학습 데이터를 실제로 학습하는 DQN 모델을 따로 두어 서로 번갈아 가면서 사용한다로 이해해도 되겠습니다.

Q: activation function 으로 non-linear function 을 쓰는 이유가 Cost function 에 비선형성을 주기 위함임은 이해하였고, identity function 을 쓴 예시도 이해하였습니다.

A: 그런데 ReLU 가 어떻게 비선형성을 부여하는지에 대해 의문점이 생겼습니다. 단순히 음수부분만 버리게 되어도 비선형성이 주어지는 이유가 궁금합니다.

비선형적 함수는 곡선(면)으로 출력이 생성된다는 것을 의미하는 것이 아니라. 아래와 같은 조건을 만족하는 함수 f (선형함수)가 아니라는 뜻입니다.

$$f(ax+by)=af(x)+bf(y)$$

따라서, 직선으로 끝없이 뻗어있는 함수가 아니라면 모두 비선형적이라고 할 수 있지요. 결론적으로 ReLU 는 비선형적이지만 경계면이 직선으로 생성되는 것은 선형함수(identity function)와 같습니다.(곡선은 여러개의 직선을 조합하여 근사(approximation)해 낼 수 있겠죠?ㅎㅎ) 다만, 미분이 음수에 대해서는 0, 양수에 대해서는 1 이므로 gradient

vanishing 문제를 해소할 수 있어(gradient 가 곱한다고 점점 작아지지 않으므로... 만약 음수에 대해서는 바로 0 이 되므로 그 전으로는 backpropagation 을 할 필요가 없게 되죠.) 쓰는 것입니다. 그리고 추가적으로 선형함수 보다 좀더 복잡한 경계면을 더 적은 뉴런과 층으로 근사하여 생성해 낼 수 있는 장점이 있기도 합니다. 혹시 궁금하시면 간단한 convex 또는 concave 함수를 identity function 을 (activation function 으로) 사용하는 뉴런의 조합으로 근사시키는 것과 ReLU 를 (activation function 으로) 사용하는 뉴런의 조합으로 근사시키는 것을 비교해 보시길 바랍니다.

설명이 좀 길었네요.ㅠ

Q: correlation 과 convolution 은 혼용해서 쓰는 건가요. 딥러닝에서 correlation 과 convolution 이 모두 순서 뒤바뀌지 않는것으로 생각해도 되나요?

A: CNN 에서 convolution 은 (bias 를 추가하여 쓸 수 있는 점을 제외하면) 신호처리에서의 (cross-)correlation 을 의미합니다. 물론 나도 처음 공부할 때, 이것 땀에 헛갈렸는데 역사는 반복되는군요.^^

Q: DQN 알고리즘 내에서 r_i 는 $R(s,a)$ 를 뜻하는 것인데 r_i 가 아닌 r_{i+1} 이어야 하지 않나요?

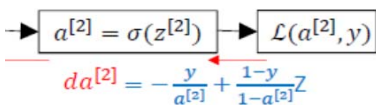
A: r_i 가 맞아 보이는데 어떤 부분이 헛갈리는지 모르겠습니다.ㅠ r_i 는 s_i 에서 a_i 를 수행했을때 받게되는 보상이므로 정의에 의하면 별 문제가 없는 표기입니다. (혹시 다음 상태에 도달해서 받게 되는 보상이라 r_{i+1} 이라고 생각하나요???)

Q: 위 질문에 대해 다시 한번 질문을 드리자면, 강의자료 6 페이지에 보면 $R(s,a)$ 에 대한 notation 을 r_{t+1} 이라고 표기한것을 봐서 오타가 아닌가 라고 생각했습니다.

그리고 6 페이지에 대한 수식을 이해 할때도 교수님이 말씀하신대로 (혹시 다음 상태에 도달해서 받게 되는 보상이라 r_{i+1} 이라고 생각하나요???) 이렇게 생각하고 저 notation 을 썼다고 이해했었습니다. 그런데 deep Q learning 에서는 notation 이 달라져서 헛갈려서 질문을 남겼습니다. 저기서는 $R(s,a)$ 에 대한 표시를 r_i 로 다시 정의한것인가요?

A: "3 번째 질문에 대해 다시 한번 질문을 드리자면, 강의자료 6 페이지에 보면 $R(s,a)$ 에 대한 notation 을 r_{t+1} 이라고 표기한것을 봐서 오타가 아닌가 라고 생각했습니다." <--
앗 그렇군요. 오타입니다.ㅠㅠ 헛갈렸겠네요.

" $R(s,a)$ 에 대한 표시를 r_i 로 다시 정의한것인가요?" <- **맞습니다.**

Q: 

위 식에서 Z 가 오타인지 의미가 있는지 궁금합니다.

A: 단순 typo 입니다. Z 는 단순 오타입니다. 찾아주어 고맙습니다.^^

Q: Adam optimizer 의 hyperparameter 를 정할 때 β_1 보다 β_2 를 크게 설정하는 이유가 있는지 궁금합니다.

A: 좋은 질문 입니다. 실험적으로(경험적으로) tuning 하여 얻게 된 값이며, 일반적으로 "gradient 의 variance($(dW)^2$, $(db)^2$) 항을 gradient momentum(dW , db) 보다 훨씬 천천히 움직이게 한다."는 것을 의미합니다. 즉, gradient 의 variance 보다 gradient 의 방향성에 보다 더 민감하게 움직이게 한다고 해석할 수도 있겠네요. (아래 내용 [11.10. Adam — Dive into Deep Learning 0.16.2 documentation \(d2l.ai\)](#) 참조.)

$$\begin{aligned} \mathbf{v}_t &\leftarrow \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1) \mathbf{g}_t, \\ \mathbf{s}_t &\leftarrow \beta_2 \mathbf{s}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2. \end{aligned} \tag{11.10.1}$$

Here β_1 and β_2 are nonnegative weighting parameters. Common choices for them are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. That is, the variance estimate moves *much more slowly* than the momentum term. Note that if we initialize $\mathbf{v}_0 = \mathbf{s}_0 = \mathbf{0}$ we have a significant amount of

Q: RNN 과 같은 회귀 모델은 어떻게 learning(학습)을 하는지 궁금하게 되어 메일을 작성하게 되었습니다. 기존의 FFNN 의 경우, 각 layer 마다 W matrix 를 update 해나가며 학습시키는 것으로 배웠습니다. 이는 각 layer 마다 W 가 독립적이기 때문에 쉽게 이해가 가능했습니다. 그러나 RNN의 경우, time step마다 input과 hidden state가 달라지는 반면 W 는 하나의 cell 로서 공유하기 때문에 update 를 어떻게 진행해야될지 감이 잘 오지 않습니다. 수업 내용에서도 W 에 대한 gradient 를 구하지 않고, hidden state 에 대한 radient 를 구하고 있는데, 이를 기존의 FFNN 처럼 해당 hidden state 를 update 하는 데에

사용하는 것 같지는 않습니다. FFNN 에서의 W 는 hyper parameter 이지만, hidden state 는 W 에 의해 결정되는 derived hyper parameter 같아서 hidden state 를 학습하는 것은 무언가 이상함을 느꼈습니다. 결과적으로, 저희가 구하는 C_t 에 대한 gradient 를 어디에 적용시켜서 학습하는 것인가요? initial hidden state 를 조정하는 것이 최종 학습의 목표일까요? 그렇다면 W 는 어떻게 조정이 되는지 궁금합니다.

$$\sum_{\tau=1}^{N_t} \sum_{t=1}^T \frac{\partial C_{\tau}}{\partial y_{\tau}} \frac{\partial y_{\tau}}{\partial h_t} \frac{\partial h_t}{\partial W}$$

아니면 혹시 위와 같은 식으로 dW 를 합하여 업데이트를 진행하는 걸까요?

A: computational graph 에서 path 가 여러개이면 미분 값을 모두 더하면 된다고 했으므로 제시해준 수식이 대략적이긴 하지만 기본 컨셉은 맞습니다. (recurrent weight matrix 를 한번 통과한 것 2 번 통과한 것, ..., i 번 통과한 것 path 가 모두 존재하므로 각 path 미분을 모두 더해주는 것으로 이해하면 좋겠네요.)

다시 말하면, 시간적으로 펼쳐서(unfold) 설명한 이유가 기존 feedforward network 와 같은 방식으로 미분을 보면 된다는 것을 보이기 위해서이기도 합니다. 다만, 이때 recurrent weight matrix W 는 공유하므로 각 시간단계에서 미분 계산된 값을 함께 더해주는 것으로 이해하면 되겠습니다.