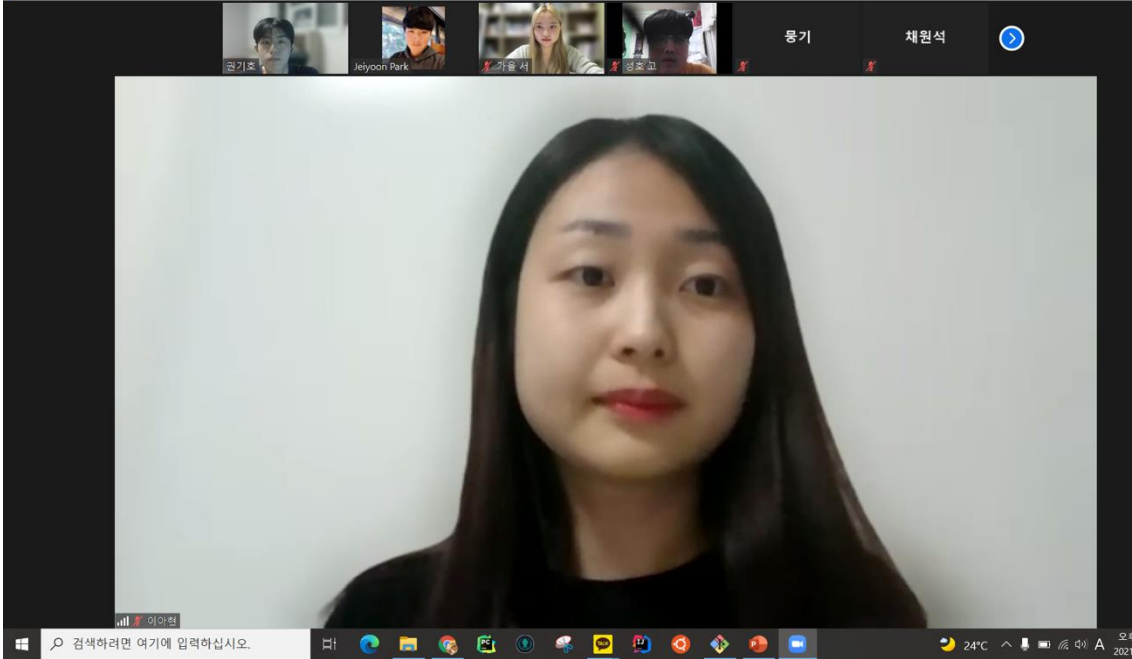
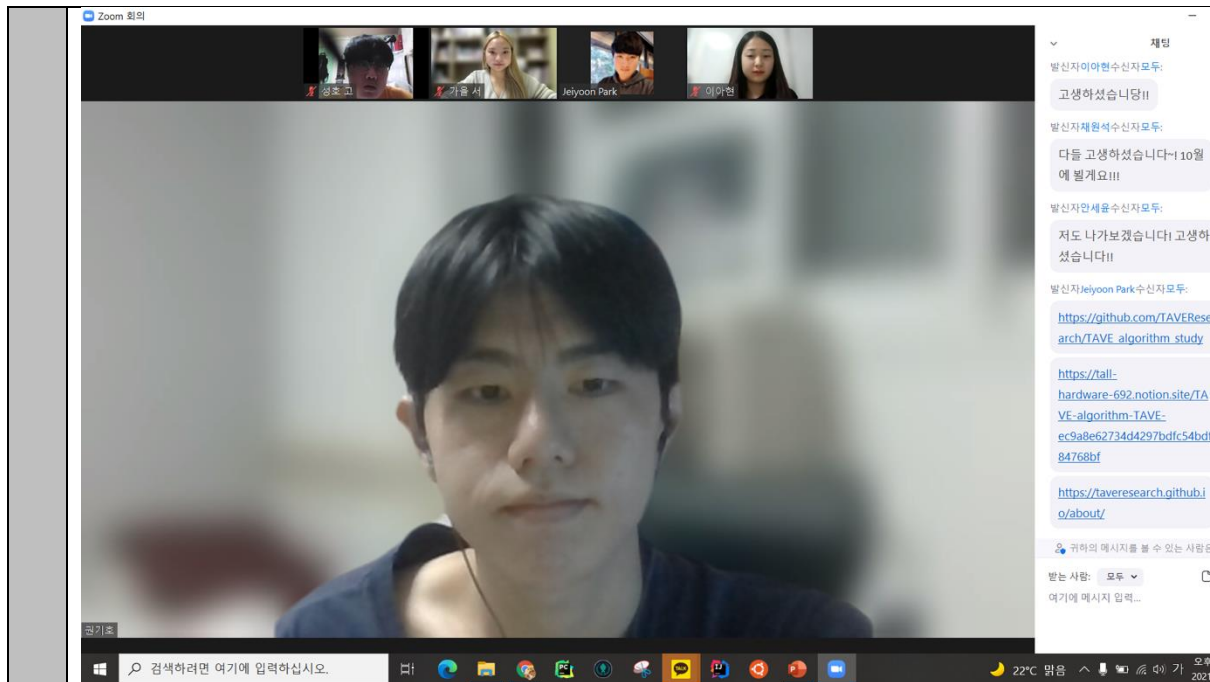


TAVE 서기

서기 내용			
서 기 일 자	21.09.17	서기	서가을
주 제	심층 신경망 훈련하기		
시 간	20:30~22:30	장소	Zoom 미팅
스 터 디 인 원	고성호, 권기호, 이아현, 서가을 : 시작		
			
	고성호, 권기호, 이아현, 서가을 : 종료		



내용

[목차]

Chapter 11. 심층 신경망 훈련하기

11.1 그래디언트 소실과 폭주 문제

11.2 사전훈련된 층 재사용하기

11.3 고속 옵티마이저

11.4 규제를 사용해 과대적합 피하기

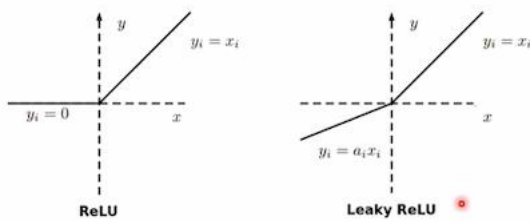
11.5 요약 및 실용적인 가이드라인

11.6 연습문제

11.1.1 글로럿과 He 초기화

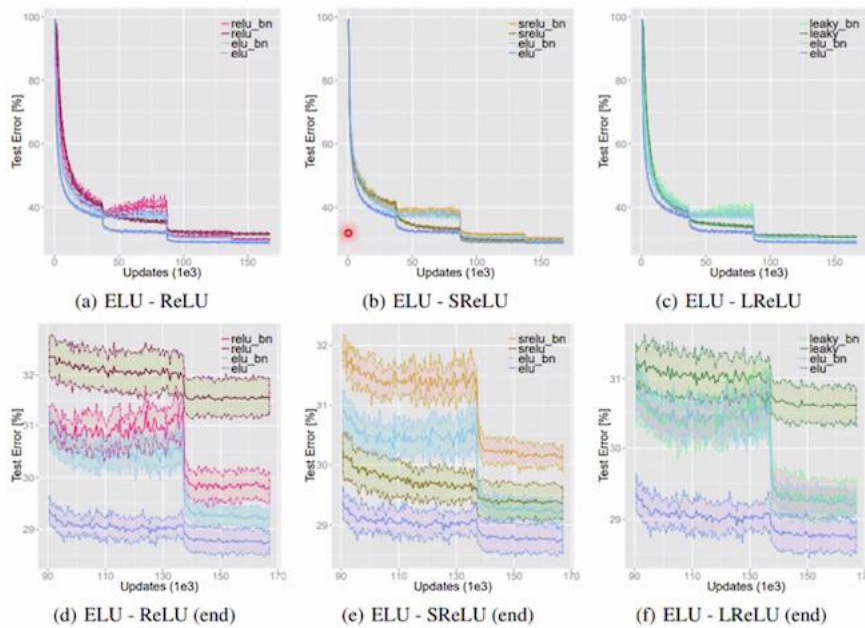
- Sigmoid를 사용하면 1보다 작은 수가 계속 곱해지므로 0으로 수렴해서 갈수록 업데이트가 안 되는 문제점 발생 : gradient 손실
- 글로럿: 여러 층의 기울기 분산 사이에 균형을 맞춰 특정 층의 부각 및 낙오를 방지
- 글로럿: S자의 활성화함수와는 좋은 성능, ReLU와는 좋지 않음.
- He: 다음 층의 뉴런 수 반영하지 않음.

11.1.2 수렴하지 않는 활성화 함수



$$f(x) = \max(0, x) \quad \text{LeakyReLU}_{\alpha}(x) = \max(\alpha x, x)$$

- $x < 0$ 의 경우 기울기가 0이 되며 죽은 ReLU가 됨 → Leaky ReLU
- ELU의 성능이 더 좋음



11.1.3. 배치 정규화

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

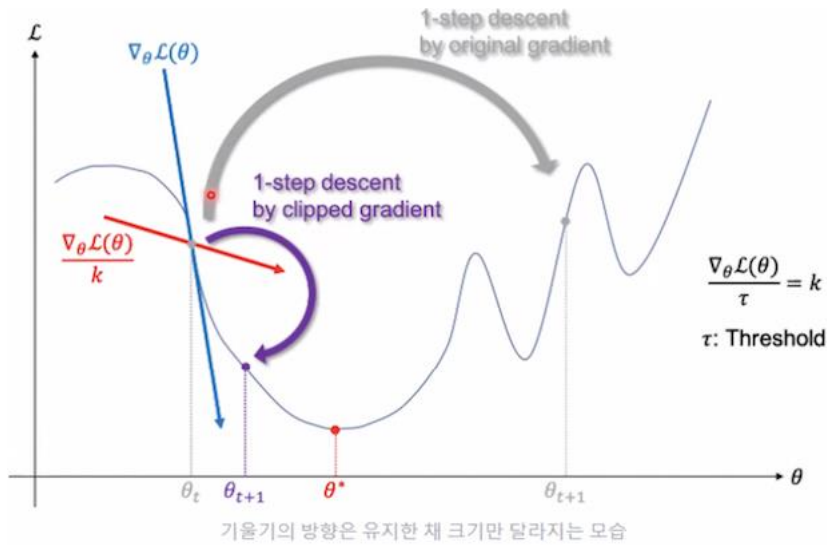
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

- 미니 배치의 평균, 분산 이용해 정규화 → scale 및 shift를 감마값, 베타값을 통해 실행
- 감마와 베타 값은 backpropagation 통해 학습

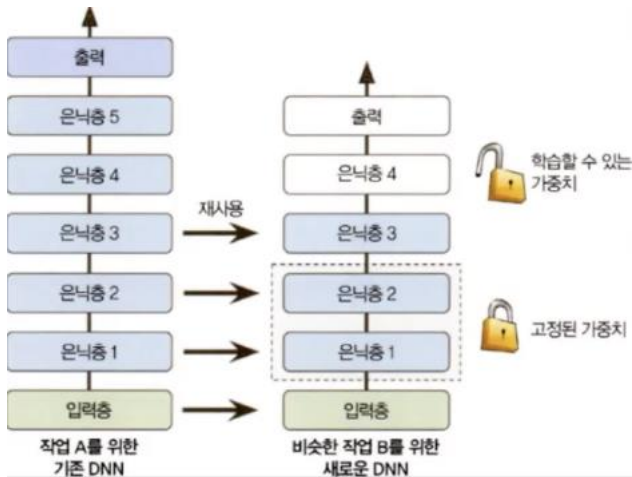
11.1.4 그레이디언트 클리핑



- 폭주문제 완화 (역전파 시 임계값을 넘지 못하게 gradient 잘라냄)
- 방향 유지, 간격 좁힘

11.2 사전훈련된 층 재사용하기

- 전이 학습의 필요성: 수많은 데이터의 요구와 학습 시간 단축



- 가중치 고정(동결) 후 출력층 추가

```
model_A = keras.models.load_model("my_model_A.h5")
model_B_on_A = keras.models.Sequential(model_A.layers[:-1])
model_B_on_A.add(keras.layers.Dense(1, activation="sigmoid"))
```

```
model_A_clone = keras.models.clone_model(model_A)
model_A_clone.set_weights(model_A.get_weights())
```

```
for layer in model_B_on_A.layers[:-1]:
    layer.trainable = False
```

→ 동결

```
model_B_on_A.compile(loss="binary_crossentropy",
                    optimizer=keras.optimizers.SGD(lr=1e-3),
                    metrics=["accuracy"])
```

```
history = model_B_on_A.fit(X_train_B, y_train_B, epochs=4,
                        validation_data=(X_valid_B, y_valid_B))
```

```
for layer in model_B_on_A.layers[:-1]:
    layer.trainable = True
```

```
model_B_on_A.compile(loss="binary_crossentropy",
                    optimizer=keras.optimizers.SGD(lr=1e-3),
                    metrics=["accuracy"])
```

```
history = model_B_on_A.fit(X_train_B, y_train_B, epochs=16,
                        validation_data=(X_valid_B, y_valid_B))
```

11.2.1 케라스를 이용한 전이 학습

- 99.25%의 정확도는 속임수(전이학습은 완전연결에서 불완전)
- 일반적 특성 감지하는 심층 합성곱 신경망에 더 잘 작동

11.2.2 비지도 사전훈련

- 레이블 되지 않은 데이터를 위해 (레이블하려면 시간과 돈이 더 많이 소요되므로 비지도 사용)
- 1. 레이블 없는 데이터를 비지도 학습 통해 모델 학습
- 2. Autoencoder나 GAN 판별자의 하위층 재사용 + 출력층 추가
- 3. 지도 학습 기법 사용해 레이블된 데이터를 위한 튜닝

11.2.3 보조 작업에서 사전훈련

- 얼굴 인식 시스템: 다른 사람인지 분별하는 첫 번째 레이어 재사용
- NLP: 텍스트 부족할 때 NLP 애플리케이션에서 코퍼스 통해 레이블된 데이터 자동 생성 후 모델 재사용

11.3 고속 옵티마이저

- SGD의 단점 (local minimum에 갇힘, parameter마다 learning rate 일정)

11.3.1 모멘텀 최적화

Momentum

$$\theta = \theta - V_t$$

$$V_t = \gamma V_{t-1} + \eta \nabla_{\theta} J(\theta)$$

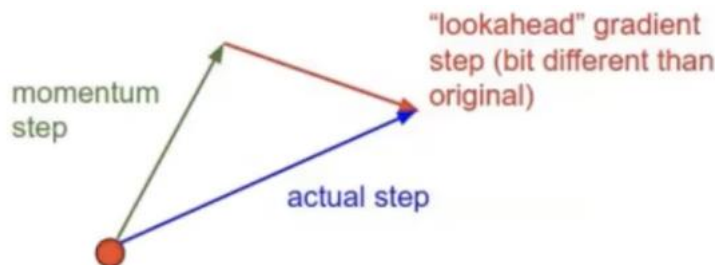
수가 계속
커지는 것을
막기 위해

이전 gradient의 합

gradient

- 모멘텀의 단점: global minimum에서 과도한 진동

11.3.2 네스테로프 가속 경사



- 진동을 감소시켜 수렴을 빨리 할 수 있음.

11.3.3 AdaGrad

- 급변한 parameter에는 작은 learning rate
- 완만한 parameter에는 큰 learning rate 적용
- 너무 빨리 느려지고 양수값이 계속 더해져서 수렴하지 못함.

11.3.4 RMSProp

- 가장 최근에 반복한 gradient만 누적
- AdaGrad의 공식에 감마를 곱함으로써 숫자 안정화

11.3.5 Adam과 Nadam

Adam 공식

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_{\theta} J(\theta)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_{\theta} J(\theta))^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \cdot 0.1$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \cdot 0.001$$

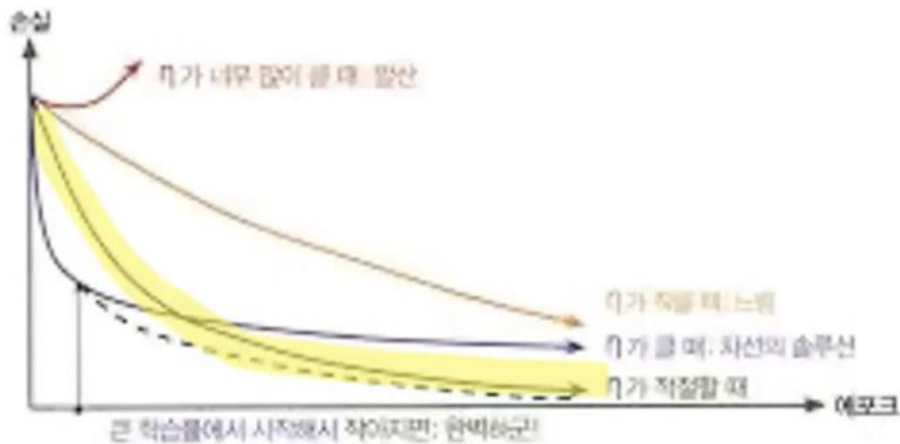
$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

두 숫자 간 차이 줄여 줌

- 학습률 parameter 튜닝할 필요 없음
- AdaMax – 최근 gradient에 비중 부여

- 업데이트 스케일 낮춰 안정적
- Nadam = Adam + NAG

11.3.6 학습률 스케줄링



- 거듭제곱 기반 스케줄링: 학습률은 스텝마다 감소, 처음에는 빠르게 감소하듯 점점 느리게 감소
- 지수 기반 스케줄링: 학습률이 스텝마다 10배씩 줄어듦
- 구간별 고정 스케줄링
- 성능 기반 스케줄링
- 1사이클 스케줄링

11.4 규제를 사용해 과대적합 피하기

- 오버피팅: 과도하게 학습이 되어 새로운 데이터의 예측력이 떨어짐

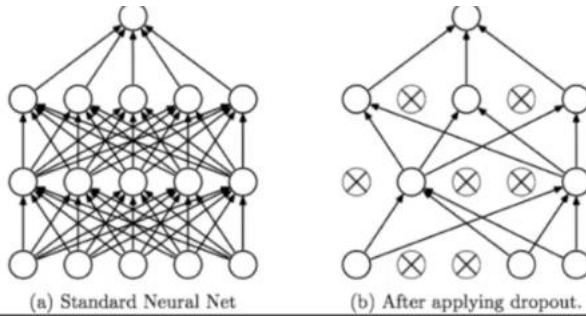
11.4.1 L1 규제와 L2규제

- Parameter 갱신은 cost function을 최소화하는 것을 목표로 함
- Weight에 제약을 줌으로써 overfitting 해결
- Norm? $L^x = \|p - q\|_x = (\sum |p_i - q_i|^x)^{\frac{1}{x}}$
- : 두 벡터 간의 크기, 길이
- L1규제는 규제의 크기가 커질수록 가중치가 0에 가까워지므로 underfitting될 가능성이 큼
- L2규제는 규제의 크기가 커져도 가중치가 0에 가까워지는 정도가 덜하기 때문에 더 선호됨.
- 케라스에서 표현한 L2 규제

```
1 layer=keras.layers.Dense(100, activation='elu', kernel_initializer='he_normal',
2 kernel_regularizer=keras.regularizers.l2(0.01))
```

11.4.2 드롭아웃

- 매 training step에서 뉴런에 p라는 확률을 부여해 제거하는 것



- Test 할 때는 모든 뉴런을 사용하므로 갑자기 많이 연결되는 뉴런의 환경을 보존하고자 보존확률(1-p)를 곱해줘야 함
- Co-adaptation 방지
- Ensemble 효과

11.4.3 몬테 카를로 드롭아웃

- Test 과정에서도 드롭아웃을 적용
- 여러개의 예측 결과를 통해 불확실성이 줄어든 예측 가능

```
1 import numpy as np
2 y_probas = np.stack([model_dropout(X_test, training=True)
3                       for sample in range(100)])
4 y_proba = y_probas.mean(axis=0)
```

11.4.4 맥스-노름 규제

- 손실함수에 규제항을 추가하지 않음
- 매 훈련 스텝이 끝날 때마다 가중치의 norm을 계산해 스케일 조정 시행

$$w \leftarrow w \frac{r}{\|w\|_2}$$

- R은 hyperparameter로서 조절 가능, 줄이면 overfitting 감소
- 케라스로 구현

```
1 keras.layers.Dense(100, activation='elu', kernel_initializer='he_normal',
2                     kernel_constraint=keras.constraints.max_norm(1.))
```

11.5 요약 및 실용적인 가이드라인

- 기본 DNN 설정

하이퍼파라미터	기본값
커널 초기화	He 초기화
활성화 함수	ELU
정규화	얕은 신경망 경우 없음, 깊은 신경망이라면 배치 정규화
규제	조기 종료 (필요하면 ℓ_2 규제 추가)
옵티마이저	모멘텀 최적화 (또는 RMSProp이나 Nadam)
학습률 스케줄	1사이클

- 실제로 모델을 돌리면서 hyperparameter가 어떤 결과를 내는지 직접 해보는 것이 더 중요.
- 완전 연결층 DNN을 위한 설정

하이퍼파라미터	기본값
커널 초기화	르쿰 초기화
활성화 함수	SELU 순환 신경망, 스킵연결 불가!
정규화	없음 (자기 정규화) SELU 활성화 함수가 평균 0 표준편차 1로 자동 정규화를 시켜 줌
규제	필요하다면 알파 드롭아웃 알파 드롭아웃 : Dropout인데 평균, 표준편차를 유지하는 드롭아웃
옵티마이저	모멘텀 최적화 (또는 RMSProp이나 Nadam)
학습률 스케줄	1사이클

- Selu – 자동으로 자기 정규화하기 때문에 따로 정규화가 필요 없고 완전연결에는 곤란하다.
- 이 또한 정해진 값을 참고만 하고 직접 경험하고 찾아보는 것이 더 중요
- 마스크드 언어 모델: NLP에서 최근에 출현.

01 시작을 어떻게 할지 모를 때 : 답지 참조 - 사전훈련된 신경망 일부 재사용 / 비지도 사전훈련 사용 / 보

02 희소모델 필요 시 (feature 개수 제한) : L1 규제

03 빠른 응답 필요 시 : Relu, LeakyRelu

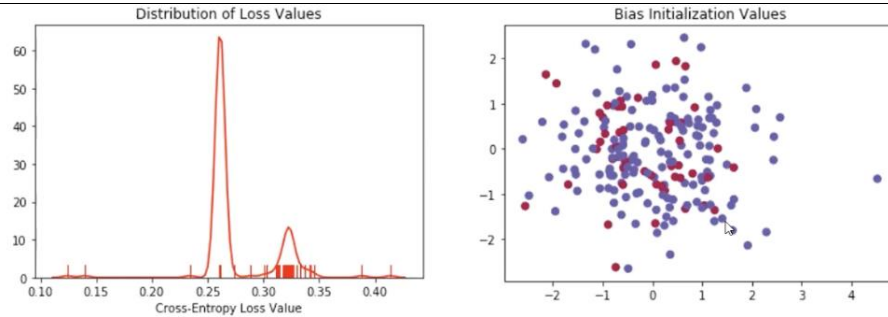
04 수십배 느려도 정확한 예측 필요 시 : MC Dropout

단, Input Feature는 정규화가 필요하다는 것!

- Relu나 leakyrelu는 일차함수이므로 간단하고 빠른 결과 가능
- 하이퍼파라미터 튜닝 (그리드 서치, 랜덤 탐색, Bayesian Optimizer)

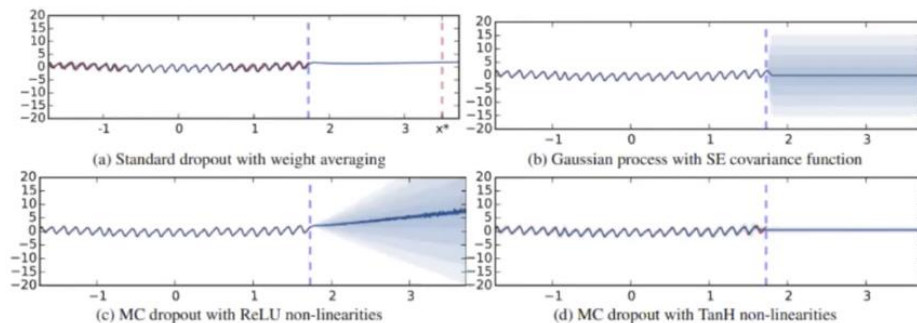
11.6 연습 문제

- 1번 문제: He초기화를 사용하여 무작위로 선택한 값이라도 모든 가중치를 같은 값을 초기화하면 안 된다.
- 여러 뉴런들이 같은 곳에서 출발하면 같은 값을 가지게 되므로 무작위 위치에서 출발해야 함. → 수렴 시간도 길고 성능 저하
- 2번 문제: 편향을 0으로 초기화해도 된다.
- 초기 연구에는 relu에 의한 죽은뉴런 방지 위해 0.1, 0.01 사용했지만 발전해서 거의 0으로 초기화함.



Weight는 안 됨!

- 5번 문제: SGD를 사용할 때 모멘텀 하이퍼파라미터를 너무 1에 가깝게 하면?
- SGD는 여러번 진동하게 되어 수렴이 오래 걸린다는 문제점이 있으므로 문제점이 발생했을 때 원인을 SGD의 사용으로 의심해보는 것이 합리적.
- 7번 문제: 드롭아웃이 훈련속도를 느리게 하지만 추론은 느리게 하지 않는다. 또 MC 드롭아웃은 시행마다 결과가 조금 다름
- 드롭아웃은 학습 시에만 적용하므로 inference 속도에 영향을 미치지 않음. Training에는 속도가 느려지지만 overfitting을 막아주는 장점이 있음



- 8번의 연습 과제는 직접 만들어보고 github에 올려둔 자료 참고하기 !

과
제
할
당

Chapter 12
12.1 안세윤
12.2 서가을
12.3 박제윤
12.4 이아현
12.5 허주희

특

스터디 발표 및 질의응답 시간 이후 전 시간에 합의한 대로 밑바닥부터 시작하는 딥러닝 1권을 공부하는 시간을 가졌습니다.

이 사 항	
비 고	없음