심층학습
04 심층 신경망 훈련

SW융합학부 양희경

주교재: 오렐리앙 제롱, 핸즈온 머신러닝(사이킷런과 텐서플로를 활용한 머신러닝, 딥러닝 실무), 한빛미디어, 2018.04

학기 내용

1. 심층학습 소개Deep learning
2. 신경망Neural network
3. 역전파Backpropagation
4. **심층 신경망 훈련**
5. 합성곱 신경망Convolutional neural network(CNN)
6. 오토인코더 Auto encoder(AE)
7. 적대적 생성 네트워크Generative adversarial network(GAN)
8. 순환 신경망Recurrent neural network(RNN)

내용

4.1 문제정의

4.2 그라디언트 소실과 폭주 문제 피하기

4.3 고속 옵티마이저

4.4 과대적합을 피하기위한 규제방법

4.5 미리훈련된 층 재사용

4.1 문제정의

- 수렴Convergence to global minimum
  - 그라디언트 소실과 폭주 문제Vanishing/Exploding gradient
  - 느린 수렴

- 과대적합Overfitting

4.1 문제정의

- 수렴Convergence to global minimum
  - 그라디언트 소실과 폭주 문제Vanishing/Exploding gradient
    - Sigmoid 활성화 함수 + 가중치 초기화 방법
      - 표준편차 1인 정규분포로 초기화 → 신경망의 출력 레이어로 갈 수록 분산이 계속 커져, 활성화 함수가 0이나 1로 수렴하게 됨
    - 4.2장

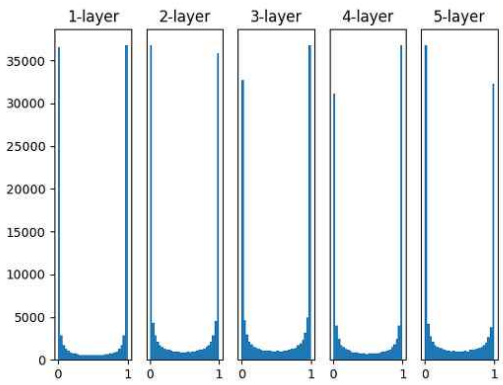
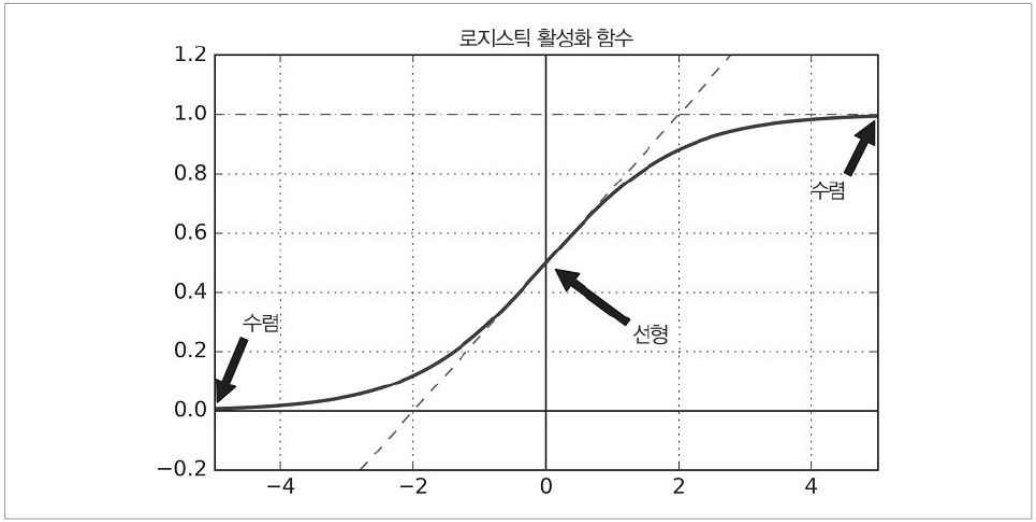


그림 11-1 로지스틱 활성화 함수의 수렴

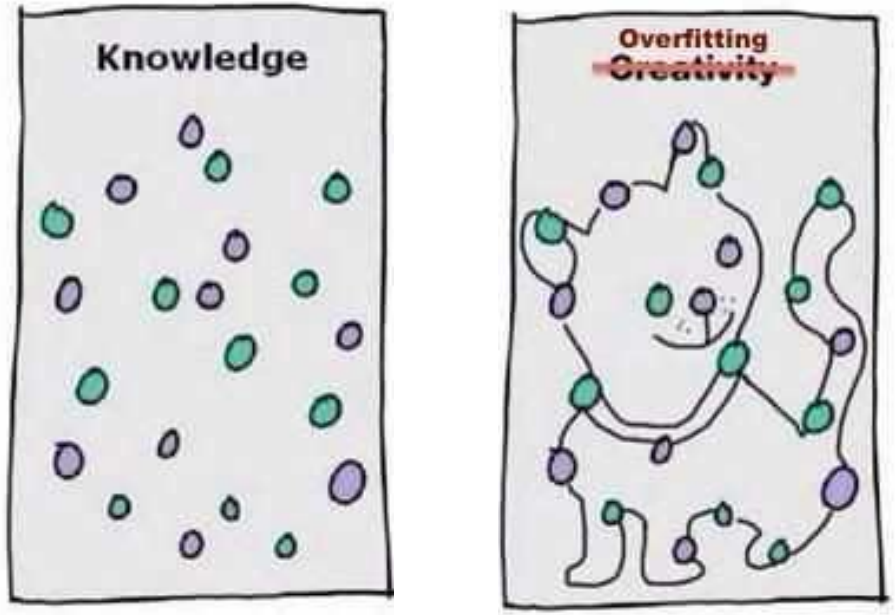


4.1 문제정의

- 수렴Convergence to global minimum
  - 느린 수렴
    - 4.3장

4.1 문제정의

- 과대적합/과소적합Overfitting/Underfitting
  - 과대적합 문제

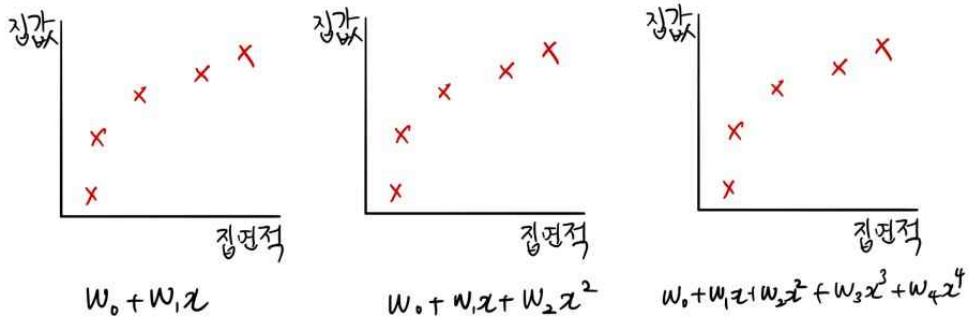


© Atul Aphale/Linkedin

4.1 문제정의

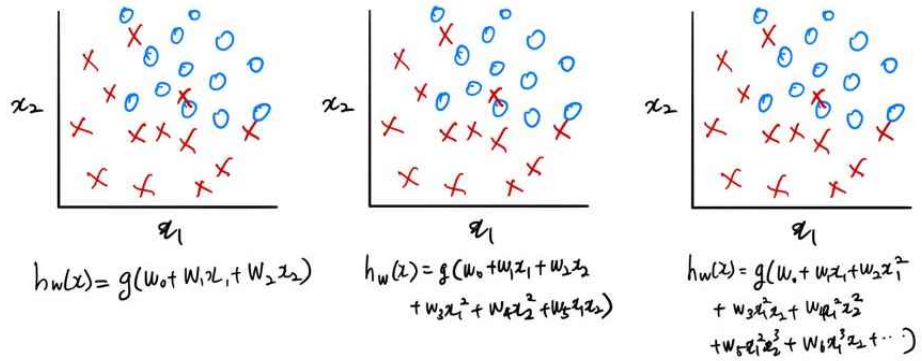
- 과대적합/과소적합Overfitting/Underfitting

회귀 문제



과소적합Underfit (High bias) | **알맞음** | **과대적합Overfit** (High variance)

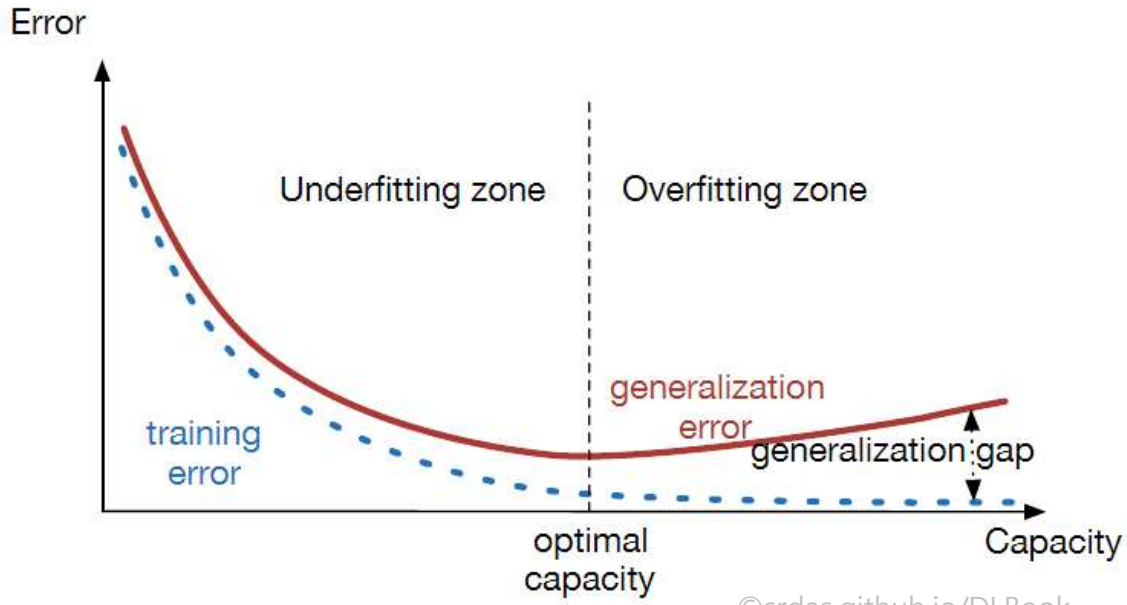
분류 문제



→ 규제/정규화Regularization 방법 (4.4장)

4.1 문제정의

- 과대적합/과소적합Overfitting/Underfitting
  - test error - train error = generalization gap
    - test error = generalization gap – train error
  - 목표는 test error의 최소화



내용

4.1 문제정의

4.2 그라디언트 소실과 폭주 문제 피하기

4.3 고속 옵티마이저

4.4 과대적합을 피하기위한 규제방법

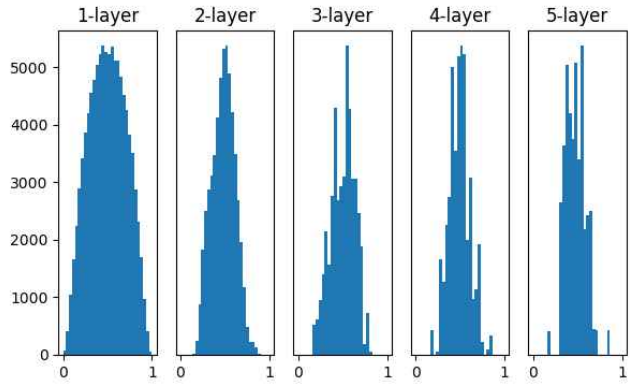
4.5 미리훈련된 층 재사용

4.2 그라디언트 소실과 폭주 문제 피하기

1) He 초기화
2) 활성화 함수
3) 데이터 정규화
4) 배치 정규화
5) 그라디언트 클리핑

4.2 그라디언트 소실과 폭주 문제 피하기

- 세이비어 초기화Xavier initialization
  - 세이비어 글로럿Xavier Glorot 과 죠슈아 벤지오Yoshua Bengio가 제안
  - 목적
    - 각 층의 출력에 대한 분산이 입력에 대한 분산과 같게 하자
    - 역방향에서 레이어를 통과하기 전과 후의 그라디언트 분산을 같게 하자
  - 초기화할 때 표준편차 $\frac{1}{\sqrt{n}}$ 인 정규분포 사용
  - Sigmoid 나 tanh 를 사용할 때 그라디언트가 잘 전달됨
  - 실전에서 잘 동작함을 입증함
    - 훈련 속도 높일 수 있었음
    - 현재 딥러닝의 성공을 견인한 기술로 평가



GLOROT, Xavier; BENGIO, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010. p. 249-256.

4.2 그라디언트 소실과 폭주 문제 피하기

1) He 초기화He initialization

- Kaiming He 가 제안
- 초기화할 때 표준편차 $\frac{2}{\sqrt{n}}$ 인 정규분포 사용
- ReLU 를 사용할 때 그라디언트가 잘 전달됨

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).

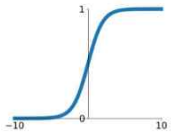
4.2 그라디언트 소실과 폭주 문제 피하기

2) 활성화 함수Activation function

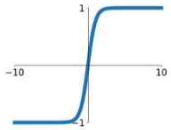
- 생물학적 뉴런의 방식과 비슷한 시그모이드 함수가 최선의 활성화 함수?
  - 글로럿과 벤지오의 2010년 이전까지 생각
  - 다른 활성화 함수가 심층 신경망에서 훨씬 잘 동작함이 밝혀짐

- ReLURectified linear unit
- LeakyReLU
- RReLURandomized leaky ReLU
- PReLUParametric leaky ReLU
- ELUExponential linear unit

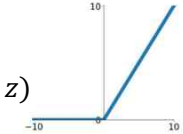
$$g(z) = \frac{1}{1+e^{-z}}$$



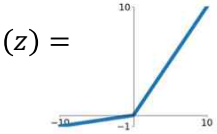
$$tanh\ (z) = \frac{1-e^{-z}}{1+e^{-z}}$$



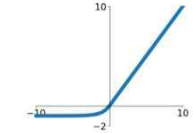
$$ReLU\ (z) = \max(0, z)$$



$$LeakyReL\quad U_\alpha(z) = \max(\alpha z, z)$$



$$ELU_\alpha(z) = \begin{matrix} \alpha(e^z - 1), when\ z < 0 \\ z, \qquad when\ z \geq 0 \end{matrix}$$



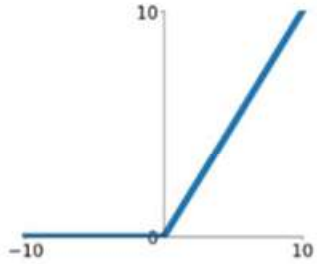
@towards data science

4.2 그라디언트 소실과 폭주 문제 피하기

2) 활성화 함수

- ReLU[Rectified linear unit]
  - ReLU(z)=max(0, z)
  - (+) 0 이상인 곳에서 그라디언트 존재
  - (+) 특정 양숫값에 수렴하지 않음
  - (-) 입력 값이 음수가 되면 그 다음부터 0 출력
    → 그라디언트 0 (죽은 ReLU[dying ReLU] 문제)

$$ReLU\ (z) = \max(0, z)$$



4.2 그라디언트 소실과 폭주 문제 피하기

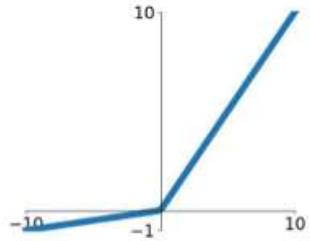
2) 활성화 함수 [Xu2015]

- LeakyReLU
  - $\text{LeakyReLU}_\alpha(z)=\max(\alpha z, z)$
  - α : 새는(leaky) 정도. 일반적으로 0.01
  - (+) z<0 이어도 작은 기울기를 갖게 되어, 혼수상태에 오래 있을 수 있지만 다시 깨어날 가능성을 얻게 됨
  - LeakyReLU 가 ReLU 보다 항상 높은 성능을 냄

- RReLU[Randomized leaky ReLU]
  - α 를 무작위로 선택하고, 테스트시에 평균을 사용
  - (+) 과대적합 위험 줄이는 역할

- PReLU[Parametric leaky ReLU]
  - α 가 훈련하는 동안 학습됨
  - (+) 대규모 이미지 데이터셋에서는 ReLU보다 크게 좋은 성능
  - (-) 소규모 데이터셋에서는 과대적합 위험

$LeakyReL\ \ U_\alpha(z) = \max(\alpha z, z)$



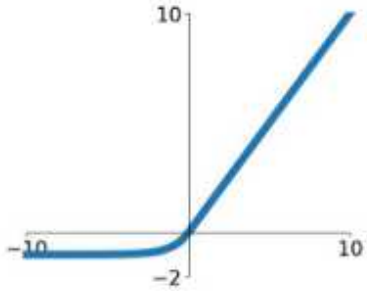
Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

4.2 그라디언트 소실과 폭주 문제 피하기

2) 활성화 함수

- ELU Exponential linear unit
  - 모든 ReLU 변종의 성능을 앞지름
  - $ELU_\alpha(z) = \begin{cases} \alpha(e^z - 1), when\ z < 0 \\ z, \quad when\ z \geq 0 \end{cases}$
  - (-) 지수 함수를 사용하기 때문에 ReLU 및 변종들보다 계산이 느림

$$ELU_\alpha(z) = \begin{matrix} \alpha(e^z - 1), when\ z < 0 \\ z, \quad when\ z \geq 0 \end{matrix}$$



Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

4.2 그라디언트 소실과 폭주 문제 피하기

- 일반적인 활성화 함수 선호도
  - ELU > LeakyReLU(변종들) > ReLU > tanh > sigmoid
  - 실행속도: LeakyReLU > ELU
  - 과대적합: RReLU
  - 큰 훈련 세트: PReLU

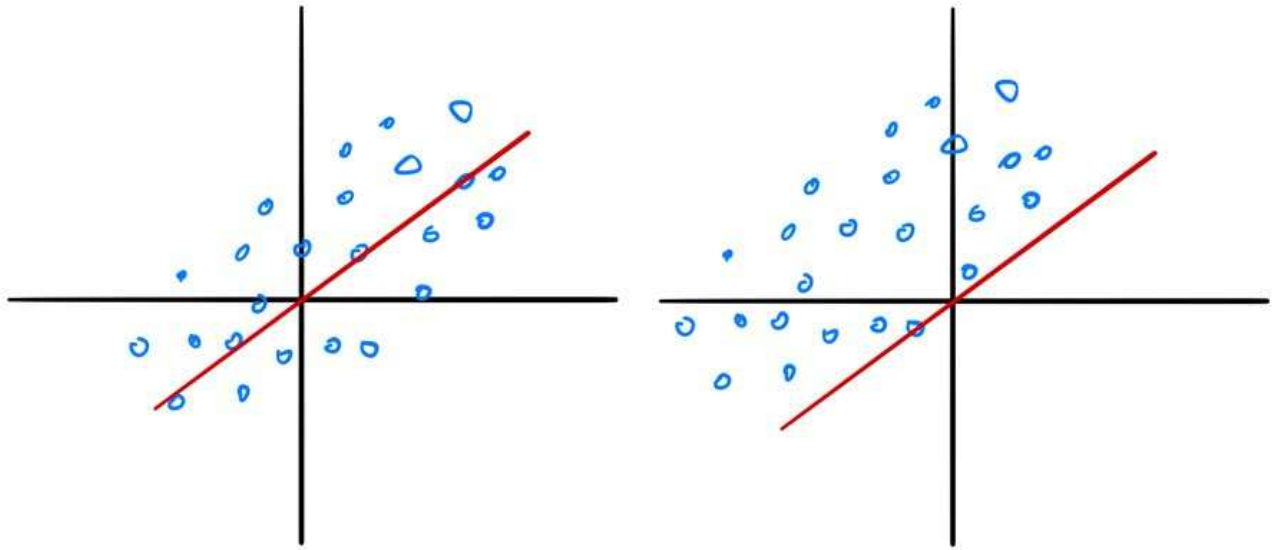
```
CLASS torch.nn.ReLU(inplace=False)
CLASS torch.nn.LeakyReLU(negative_slope=0.01, inplace=False)
CLASS torch.nn.PReLU(num_parameters=1, init=0.25)
CLASS torch.nn.RReLU(lower=0.125, upper=0.3333333333333333, inpla
CLASS torch.nn.ELU(alpha=1.0, inplace=False)
```

4.2 그라디언트 소실과 폭주 문제 피하기

3) 데이터 정규화Data normalization

– 필요성

- 학습 데이터와 테스트 데이터의 분포가 다를 수 있음



학습 데이터 테스트 데이터

4.2 그라디언트 소실과 폭주 문제 피하기

3) 데이터 정규화Data normalization

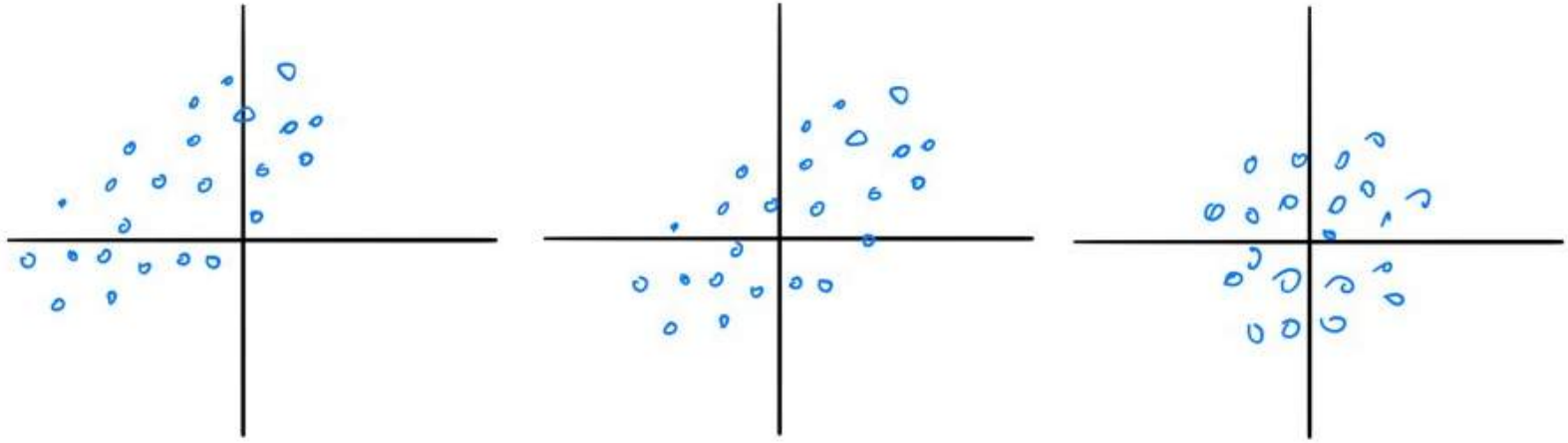
– 필요성

- 레나인지 아닌지 구분하는 데 명암은 상관 없음

4.2 그라디언트 소실과 폭주 문제 피하기

3) 데이터 정규화Data normalization

– 방법



Unnormalized $x' = x - \mu$ $x' = \frac{x - \mu}{\sigma}$

4.2 그라디언트 소실과 폭주 문제 피하기

3) 데이터 정규화Data normalization

– 방법

```
cifar10_train=dset.CIFAR10("../",train=True, transform=transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(mean=(0.5,0.5,0.5), std=(0.5,0.5,0.5)),

                         ])
                          , target_transform=None, download=False)
```

4.2 그라디언트 소실과 폭주 문제 피하기

4) 배치 정규화Batch normalization

- 각 레이어에 입력 값의 분포가 변화되는 문제 (Internal covariate shift)
  - 입력 레이어의 값이 정규화 되어있어도 레이어를 거치는 과정에서 또 다시 shift 가 일어남
- Activation 값들을 정규화 하자

$$
\begin{aligned}
&1. \quad \mu_B = \frac{1}{m_B}\sum_{i=1}^{m_B}\mathbf{x}^{(i)} \\
&2. \quad {\sigma_B}^2 = \frac{1}{m_B}\sum_{i=1}^{m_B}(\mathbf{x}^{(i)} - \mu_B)^2 \\
&3. \quad \hat{\mathbf{x}}^{(i)} = \frac{\mathbf{x}^{(i)} - \mu_B}{\sqrt{{\sigma_B}^2 + \epsilon}} \\
&4. \quad \mathbf{z}^{(i)} = \gamma\hat{\mathbf{x}}^{(i)} + \beta
\end{aligned}
$$

배치 정규화 알고리즘

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

4.2 그라디언트 소실과 폭주 문제 피하기

4) 배치 정규화Batch normalization

- 장점
  - Sigmoid 또는 tanh 같은 활성화 함수를 사용하더라도 그라디언트 소실 문제가 크게 감소됨
  - 네트워크가 초기 가중치에 훨씬 덜 민감해짐
  - ImageNet 분류 문제에서 사람의 능력을 뛰어 넘는 4.9% top-5 에러 달성
  - 과대적합에 대한 규제 역할(4.4장)

- 단점
  - 모델의 복잡도를 키움
  - 레이어마다 추가되는 계산 → 정방향 전파 속도 느려짐
  - 정방향 전파 속도(예측) 빨라야 하는 경우,
    ELU + He 초기화 만으로 얼마나 성능이 좋은지 체크할 필요

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

4.2 그라디언트 소실과 폭주 문제 피하기

5) 그라디언트 클리핑Gradient clipping

- 그라디언트 폭주 문제에서 사용되는 기법
- 역전파 시 임계값을 넘지 못하게 그라디언트를 단순히 자르자
- 순환 신경망에서 널리 사용됨(RNN, 8장)

내용

4.1 문제정의

4.2 그라디언트 소실과 폭주 문제 피하기

4.3 고속 옵티마이저

4.4 과대적합을 피하기위한 규제방법

4.5 미리훈련된 층 재사용

4.3 고속 옵티마이저

- Momentum
- Nesterov momentum
- AdaGrad
- RMSProp
- Adam

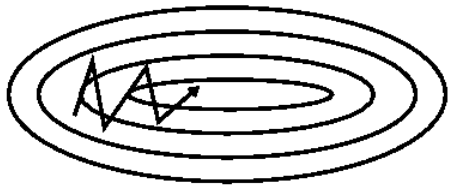
- Learning rate scheduling

4.3 고속 옵티마이저

- 모멘텀 최적화Momentum optimization
  - 경사 하강법
    - $\theta = \theta - \alpha \nabla_\theta J(\theta)$
    - 경사면을 따라 일정한 크기의 스텝으로 조금씩 내려감
  - 모멘텀 최적화
    - $\boldsymbol{m} = \beta \boldsymbol{m} + \alpha \nabla_\theta J(\theta)$

      $\theta = \theta - \boldsymbol{m}$
    - 경사면을 따라 처음에는 느리게 내려가지만, 빠르게 가속하자
    - 이전 그라디언트가 얼마였는 지를 고려
    - (+) 경사 하강법보다 빠르게 평편한 지역을 탈출하는 역할
    - (-) 튜닝할 하이퍼파라미터 늘어나지만, 보통 $\beta = 0.9$ 에서 잘 동작

SGD

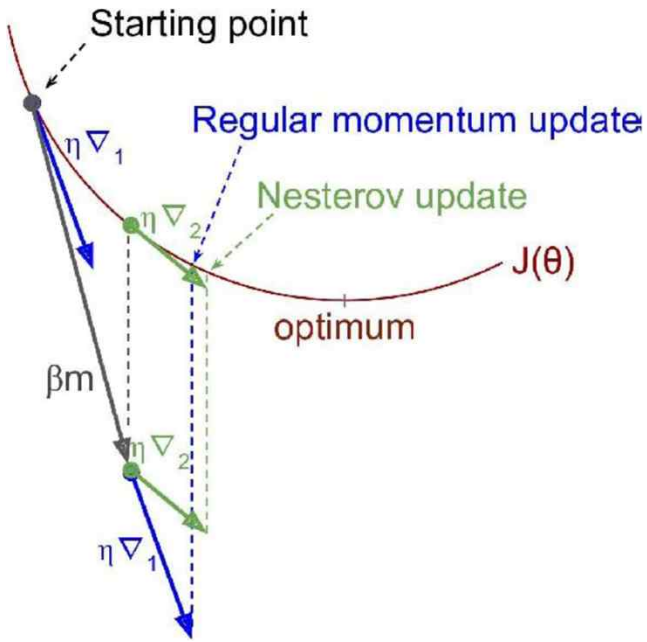


SGD with momentum

Polyak, Boris T. "Some methods of speeding up the convergence of iteration methods." *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964): 1-17.

4.3 고속 옵티마이저

- 네스테로프 모멘텀 최적화Nesterov momentum optimization
  - $\boldsymbol{m} = \beta \boldsymbol{m} + \alpha \nabla_\theta J(\theta - \beta \boldsymbol{m})$
    $\theta = \theta - \boldsymbol{m}$
  - 최적값에 조금 더 가까움



Nesterov, Yurii. "A method for unconstrained convex minimization problem with the rate of convergence O (1/k^ 2)." *Doklady an ussr*. Vol. 269. 1983.

4.3 고속 옵티마이저

- AdaGrad(Adaptive gradient algorithm)
  - 식

$$1. \quad \mathbf{s} \leftarrow \mathbf{s} + \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$$

$$2. \quad \theta \leftarrow \theta - \eta \, \nabla_\theta J(\theta) \oslash \sqrt{\mathbf{s} + \epsilon}$$

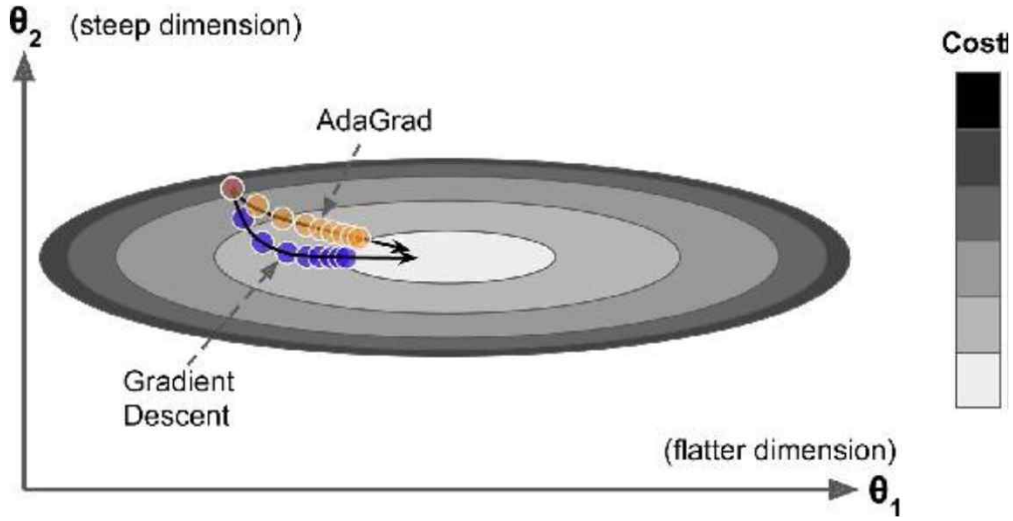
    - ⓧ: s_i에 대한 원소별 곱셈원소별 나눗셈
    - 각 원소별 그라디언트의 제곱을 누적
    - 그 값으로 현재 그라디언트 값을 나눔

  - 효과
    - i 번째 차원을 따라 비용함수가 가파르다면, s_i 는 점점 커짐
      → learning rate 가 큰 수로 나눠짐→ learning rate ↓
    - Learning rate 를 감소시키는데, 완만한 차원보다 가파른 차원에서 learning rate 가 더 작아지는 효과
      → 완만한 차원의 파라미터는 빨리 내려가고, 가파른 차원의 파라미터는 천천히 내려감

Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research* 12.Jul (2011): 2121-2159.

4.3 고속 옵티마이저

- AdaGrad(Adaptive gradient algorithm)
  - (+) 2차 방정식 문제에서는 잘 작동
  - (-) 심층 신경망을 훈련시킬 때 너무 일찍 멈추는 경향
    → 심층 신경망에서 사용하는 것 비추천



4.3 고속 옵티마이저

- RMSProp(Root mean square propagation)
  - 가장 최근에 계산된 그라디언트를 많이 고려하자.
  - 식

$$1. \quad \mathbf{s} \leftarrow \beta \mathbf{s} + (1-\beta)\nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$$

$$2. \quad \theta \leftarrow \theta - \eta\, \nabla_\theta J(\theta) \oslash \sqrt{\mathbf{s} + \epsilon}$$

    - 보통 $\beta = 0.9$
    - 누적된 그라디언트는 시간이 지날수록 덜 고려하겠음
  - 모멘텀, 네스테로프 최적화보다 더 빠르게 수렴
  - Adam 최적화 이전까지 가장 선호됨

Tijmen Tieleman & Geoffrey Hinton, Coursera neural network.
저자들이 해당 내용에 대한 논문을 쓰지 않았기 때문에, 종종 '강의6의 슬라이드 29' 라고 인용됨

4.3 고속 옵티마이저

- Adam(Adaptive moment estimation)
  - 모멘텀 최적화 + RMSProp
  - 식

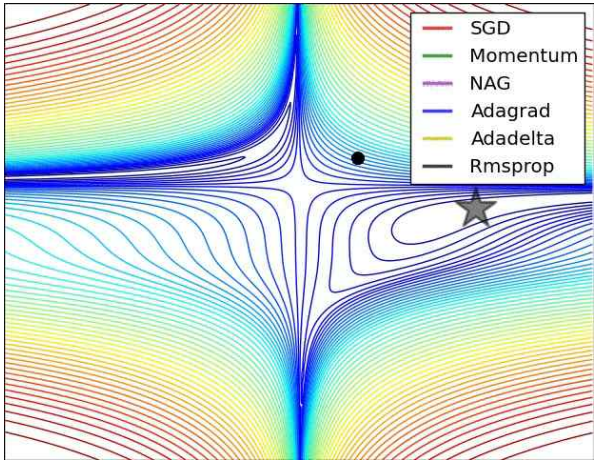
1. $\mathbf{m} \leftarrow \beta_1 \mathbf{m} - (1 - \beta_1) \nabla_\theta J(\theta)$
2. $\mathbf{x} \leftarrow \beta_2 \mathbf{x} + (1 - \beta_2) \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta)$
3. $\theta \leftarrow \theta + \eta \mathbf{m} \oslash \sqrt{\mathbf{x} + \epsilon}$

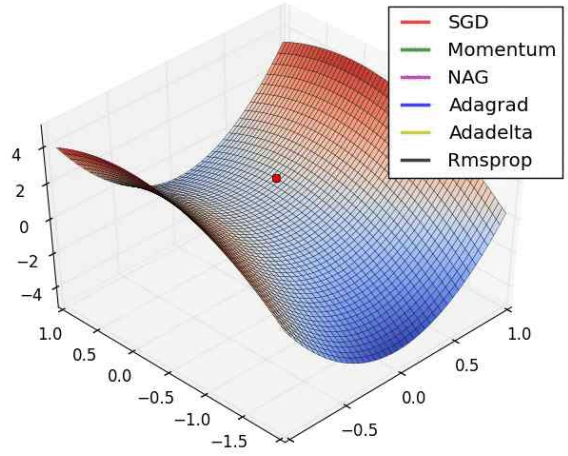
  - 보통 $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

[Kingma14] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

4.3 고속 옵티마이저

- 옵티마이저 비교
  - Adam 이전

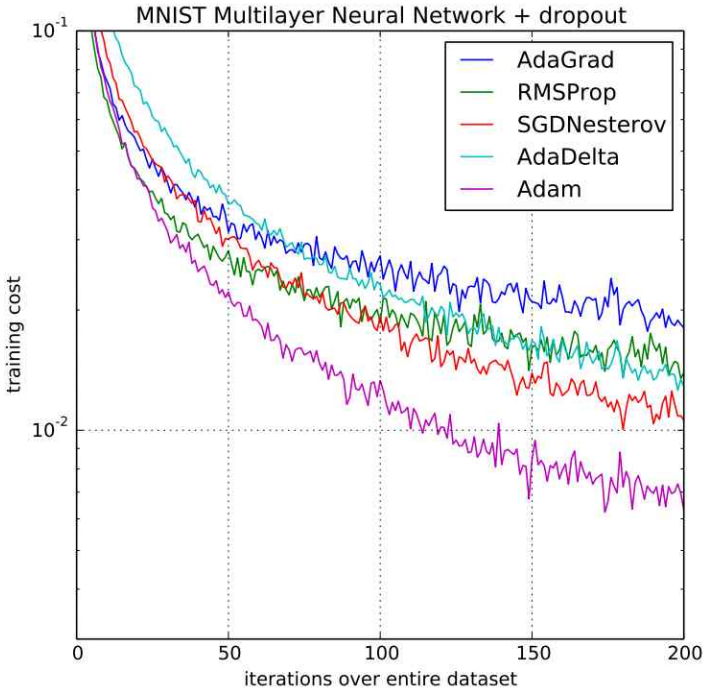


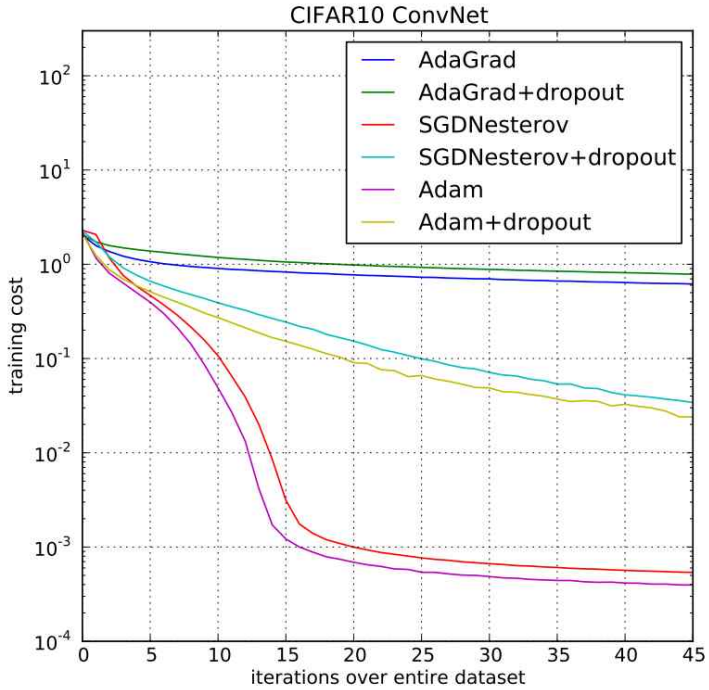


©Wikipedia

4.3 고속 옵티마이저

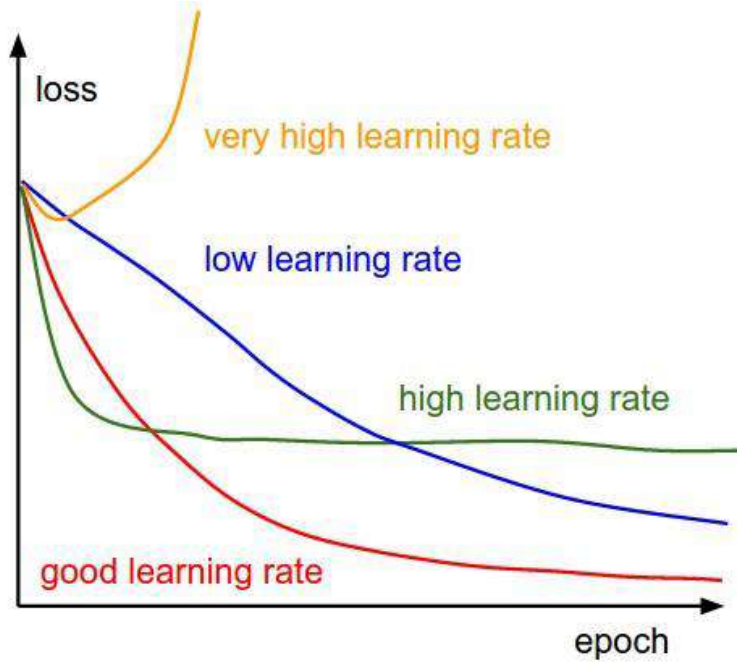
- 옵티마이저 비교
  - Adam [Kingma14]





4.3 고속 옵티마이저

- 학습률 스케줄링Learning rate scheduling(Learning rate decay)
  - 학습률의 중요성
    - 너무 작으면 매우 느리게 최적점에 수렴
    - 너무 크면 최적점 근처에서 요동(oscillation) 발생
  - 전략
    - 처음에는 크게 잡고 반복 횟수가 지남에 따라 학습률을 낮춰서 최적점에 수렴하도록 함



4.3 고속 옵티마이저

- 학습률 스케줄링Learning rate scheduling(Learning rate decay)
  - 미리 정의된 개별적인 고정 학습률
    - 해당 스텝마다(예: 0.1 → 50 epoch 후 → 0.001) 학습률 감소
    - 미리 정의된 배열의 epoch 마다 학습률 감소
  - 성능 기반 스케줄링
    - 해당 스텝마다 검증 오차를 측정하고, 오차가 줄어들지 않으면 학습률 감소
  - 지수 기반 스케줄링
    - 매 스텝 r 마다 $\eta(t) = \eta_0 10^{-\frac{t}{r}}$ 로 학습률 설정. r 스텝마다 1/10 씩 줄어듦

class `torch.optim.lr_scheduler.StepLR`(*optimizer, step_size, gamma=0.1, last_epoch=-1*)
class `torch.optim.lr_scheduler.MultiStepLR`(*optimizer, milestones, gamma=0.1, last_epoch=-1*)

class `torch.optim.lr_scheduler.ReduceLROnPlateau`(*optimizer, mode='min', factor=0.1, patience=10, verbose=False, threshold=0.0001, threshold_mode='rel', cooldown=0, min_lr=0, eps=1e-08*) [source]

class `torch.optim.lr_scheduler.ExponentialLR`(*optimizer, gamma, last_epoch=-1*) [source]

4.3 고속 옵티마이저

- 학습률 스케줄링Learning rate scheduling(Learning rate decay)
  - 선택 가이드[Senior13]
    - 모멘텀 최적화를 사용한 음성 인식용 심층 신경망 훈련시, 학습률 스케줄링 방법 비교함
    - 지수 기반 > 성능 기반 > 미리 정의된 개별적인 고정 학습률

Senior, A., Heigold, G., Ranzato, M. A., & Yang, K. (2013, May). An empirical study of learning rates in deep neural networks for speech recognition. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6724-6728). IEEE.

내용

4.1 문제정의

4.2 그라디언트 소실과 폭주 문제 피하기

4.3 고속 옵티마이저

4.4 과대적합을 피하기위한 규제방법

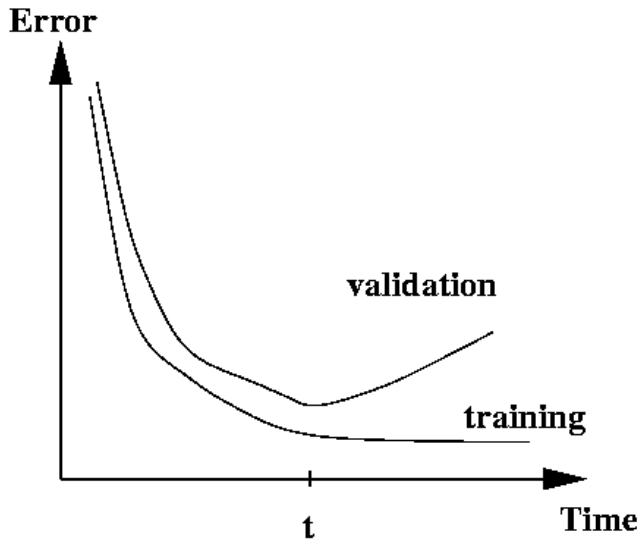
4.5 미리훈련된 층 재사용

4.4 과대적합을 피하기위한 규제방법Regularization

- 조기 종료
- 드랍아웃
- 데이터 증식
- L1과 I2 규제

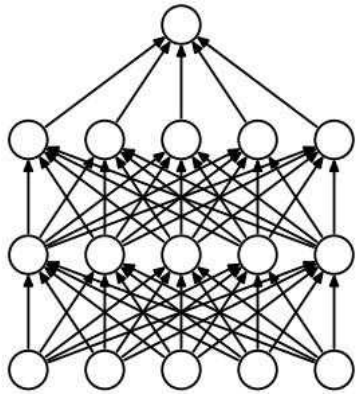
4.4 과대적합을 피하기위한 규제방법

- 조기 종료Early stopping
  - 검증 성능이 떨어지기 시작할 때 학습 중지
  - 구현 팁
    - 일정한 간격으로(예: 50 epoch) 검증 성능이 이전 최고 성능보다 좋을 경우 파라미터셋을 저장. 지정된 임계치(예: 2,000 epoch) 를 넘으면 중지 후, 마지막으로 저장된 파라미터 복원
    - 다른 규제 방법과 함께 사용

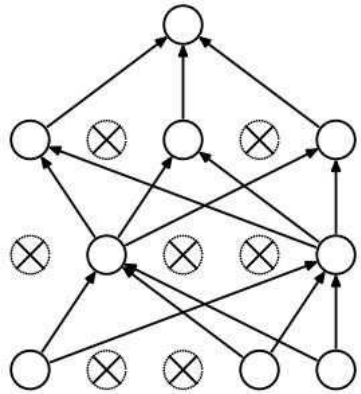


4.4 과대적합을 피하기위한 규제방법

- 드랍아웃Dropout
  - 가장 인기있는 규제 방법
  - 최고 성능의 신경망조차 정확도 1~2% 높임
    (95% → 97%) == 오차율 40% 줄어듦(5%→ 3%)
  - 방법
    - 매 훈련 스텝에서 각 유닛을 p 확률로 드랍아웃 시킴
      (이번 스텝에서 무시되더라도, 다음 스텝에서는 활성화될 수)
    - 해당 파라미터를 0으로 만드는것과 동일
    - 보통 dropout rate p=0.5
  - 주의: 훈련 마친 후에는
    드랍아웃 적용하면 안됨



(a) Standard Neural Net

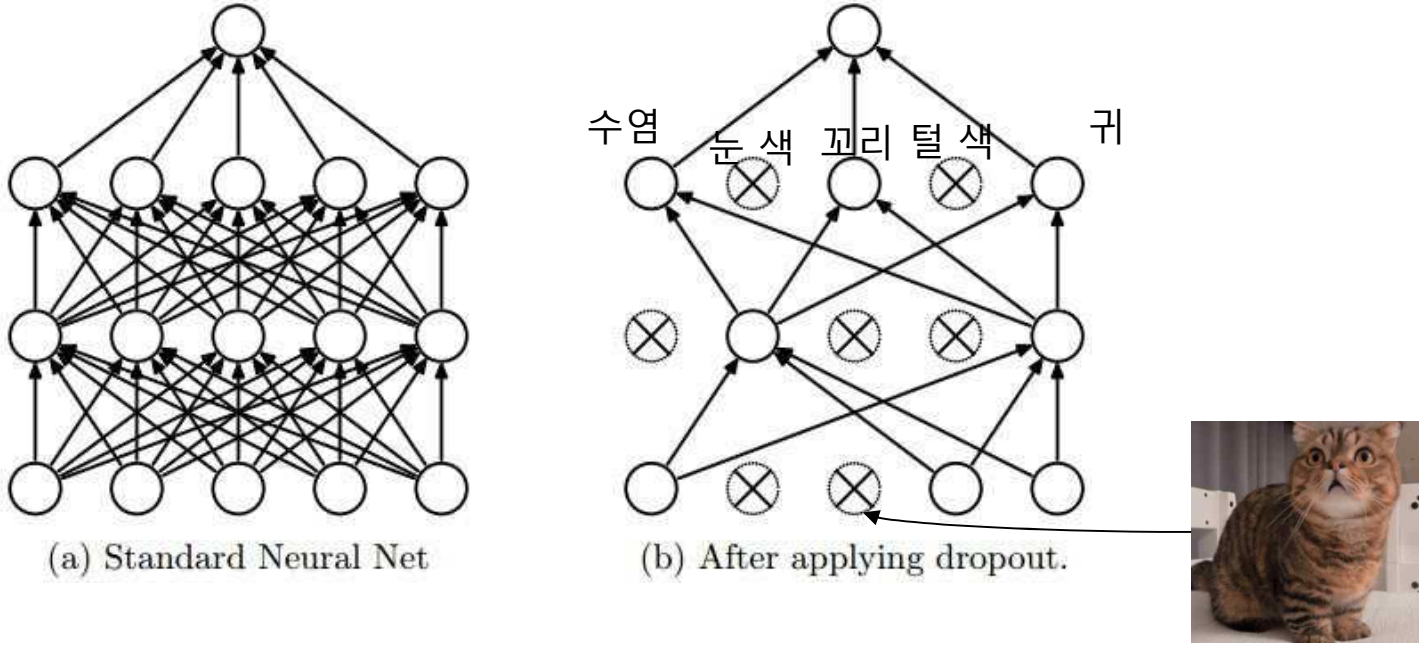


(b) After applying dropout.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

4.4 과대적합을 피하기위한 규제방법

- 드랍아웃Dropout
  - 역할 이해(예: 고양이 사진 인식)
    - 눈 색과 털 색은 드랍시킨 채 고양이를 인식하도록 함

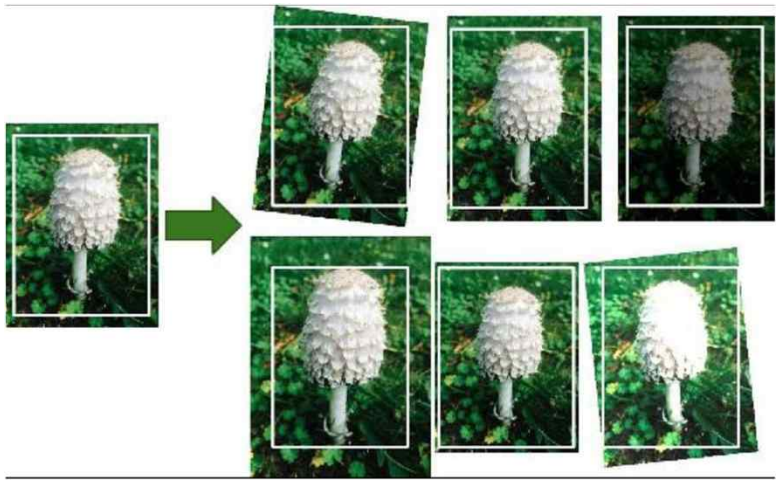


(a) Standard Neural Net

(b) After applying dropout.

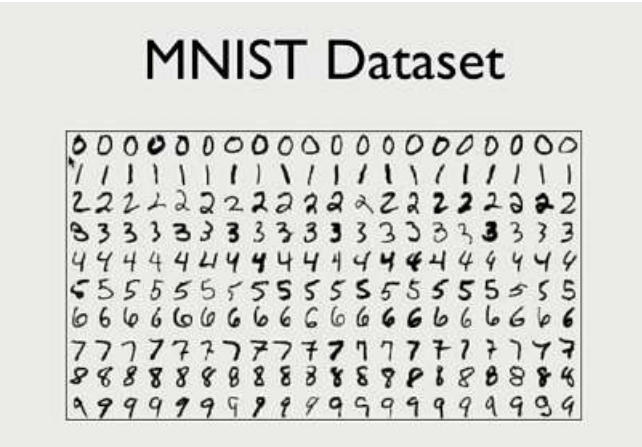
4.4 과대적합을 피하기위한 규제방법

- 데이터 증식Data augmentation
  - 기존 데이터로 새로운 데이터를 생성하여 인위적으로 훈련 세트 크기를 늘림
  - 방법
    - 밝기, 명암, 채도, 색조 제어
    - 위치 변경transpose, 이동shift, 회전rotation, 크기 조절resize, 뒤집기flip, 자르기crop 등
    - 보통 여러 방법을 조합하여 사용



4.4 과대적합을 피하기위한 규제방법

- 데이터 증식Data augmentation
  - 구현 팁
    - 매 epoch 마다 동적으로 훈련 예제를 생성
    - 작업과 데이터에 따라 적절한 데이터 증식 방법 선택해야
      - 예) MNIST, wearable device 의 데이터는 회전 비추천,
        의료 데이터는 뒤집기 비추천



4.4 과대적합을 피하기위한 규제방법

- 규제 방법 Regularization (기계학습 5장)
  - Capacity 를 넉넉히 주고 test error 를 줄이는 방법
  - 비용 함수에 regularization term 추가
    - Total loss = loss + regularization
    - 파라미터 값들이 작아짐

Linear regression with regularization

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

4.4 과대적합을 피하기위한 규제방법

- 규제 방법 Regularization (기계학습 5장)
  - L2 regularization: $\Sigma\theta^2$
  - L1 regularization: $\Sigma|\theta|$
  - Elastic net(L1+L2): $\Sigma(\beta\theta^2 + |\theta|)$

Linear regression with regularization

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

내용

4.1 문제정의

4.2 그라디언트 소실과 폭주 문제 피하기

4.3 고속 옵티마이저

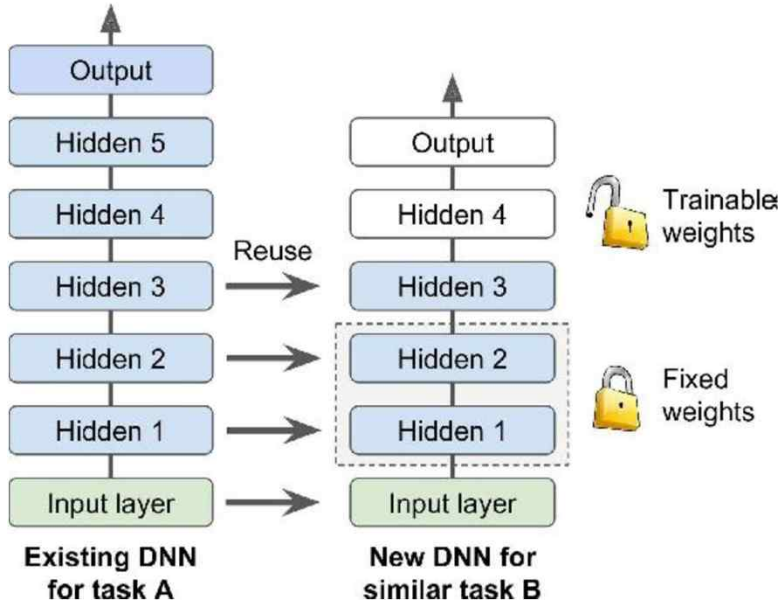
4.4 과대적합을 피하기위한 규제방법

4.5 미리훈련된 층 재사용

4.5 미리훈련된 층 재사용

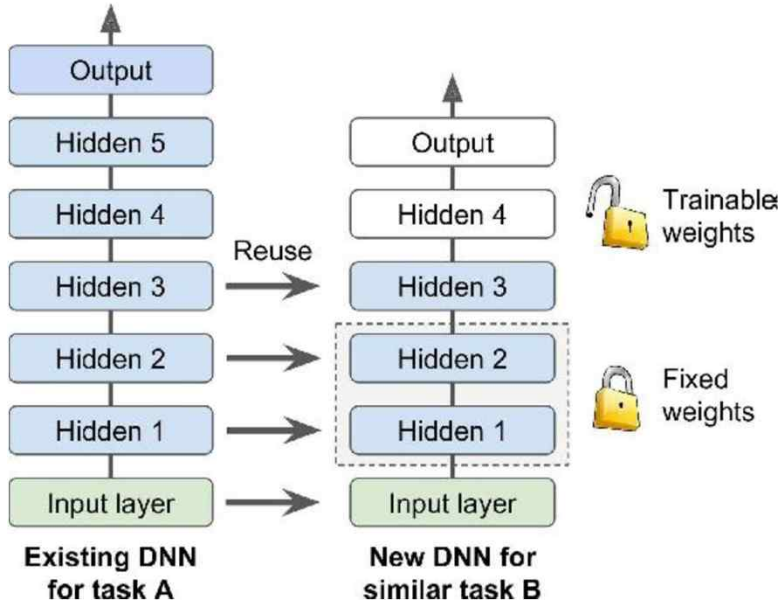
- 파이어니어의 팁pioneer's tip
  - 보통 큰 규모의 DNN을 처음부터 새로 훈련시키는 것 비추천

- 전이 학습Transfer learning
  - 풀고자 하는 작업과 비슷한 유형의 문제를 처리한 신경망이 있는 지 찾아보고, 그 신경망의 하위층을 재사용! (5장)
  - 훈련 속도 ↑, 훈련 데이터 적게 듦
  - 예) 동/식물, 자동차, 생필품 등
    100 개 카테고리 이미지
    분류하도록 훈련된 DNN이 있음
    → 자동차 종류 분류하는 DNN
    에서 재사용



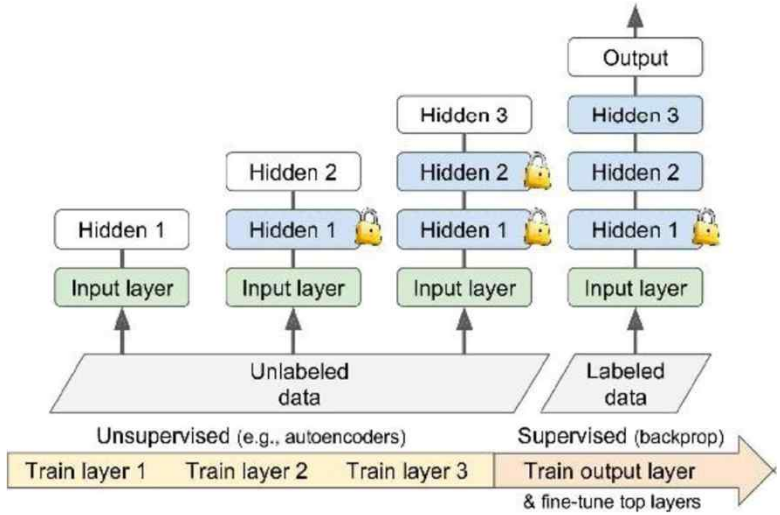
4.5 미리훈련된 층 재사용

- 전이 학습Transfer learning
  - 원래 문제의 이미지와 다른 크기의 이미지를 사용한다면, 원본 모델에 맞도록 크기를 조절하는 전처리 단계 필요
  - 일반적으로 전이학습은 입력 데이터가 유사한 저수준 특성을 가질 때 잘 작동
  - 보통 마지막 1~2 개 레이어를 붙여, 이 레이어만 학습
  - 모델 저장소model zoo



4.5 미리훈련된 층 재사용

- 비지도 사전훈련Unsupervised pretraining
  - 필요성: 레이블된 훈련 데이터가 적고 복잡한 문제 +
    비슷한 작업에 대해 훈련된 모델을 찾을 수 없을 때
    → (단순 해결책) 레이블된 훈련 데이터 더 많이 수집
    → 그것도 힘들 때
  - 제한된 볼츠만 머신Restricted Boltzmann Machines(RBM)(교재 부록 E),
    오토인코더Autoencoder(6장)



+실용적 가이드라인

- 기본 DNN 설정(교재 추천)
  - 초기화: He 초기화
  - 활성화 함수: ELU
  - 정규화: 배치 정규화
  - 규제: 드랍아웃
  - 옵티마이저: 네스테로프 가속 경사
  - 학습률 스케줄링: 없음

- 훈련에는 인내심 요구됨
  - GPU 머신 한대라면 며칠 또는 몇 달
  - 여러 GPU 머신과 여러 GPU 에 분산하는 방법 (교재 12장)

내용

4.1 문제정의

4.2 그라디언트 소실과 폭주 문제 피하기

4.3 고속 옵티마이저

4.4 과대적합을 피하기위한 규제방법

4.5 미리훈련된 층 재사용

학기 내용

1. 심층학습 소개Deep learning
2. 신경망Neural network
3. 역전파Backpropagation
4. 심층 신경망 훈련
5. **합성곱 신경망Convolutional neural network(CNN)**
6. 오토인코더 Auto encoder(AE)
7. 적대적 생성 네트워크Generative adversarial network(GAN)
8. 순환 신경망Recurrent neural network(RNN)

4.1 문제정의

- 과대적합/과소적합Overfitting/Underfitting
  - 규제방법Regularization: 과대적합 피하자