Machine Learning

Chapter 6

학습 관련 기술들

6. 학습 관련 기술들

6.1 매개변수 갱신

- **신경망 학습의 목적 : 최적화** optimization
  - 손실 함수의 값을 최대한 줄이는 매개변수의 최적값을 찾는 것
    - 매개변수 공간은 매우 넓고 복잡하다.
    - 다층 신경망에서는 매개변수의 수가 증가

      → 최적의 솔루션을 찾기 힘들다.
- **따라서 가중치 매개변수의 최적값을 탐색하는 다양한 최적화 방법이 존재한다.**
  - Stochastic Gradient Descent(SGD) / Batch Gradient Descent(BGD) / Mini-batch Gradient Descent
  - Momentum
  - Nesterov Accelerated Gradient
  - AdaGrad
  - AdaDelta
  - RMSprop
  - Adam
  - AdaMax
  - Nadam

6.1 매개변수 갱신

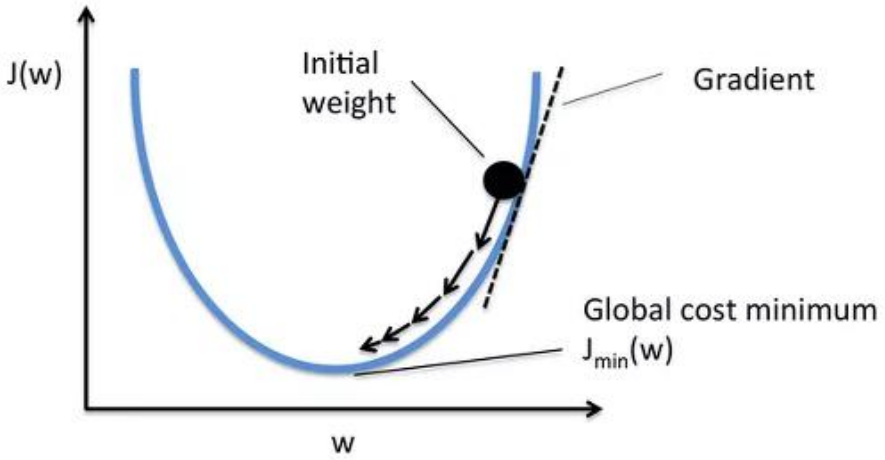
- **신경망 학습의 일반적 절차**

  1. 적절한 네트워크 선택
     - 구조(structure) : Single words / Bag of words / etc.
     - 비선형성(non-linearlity) 획득 방법 : ReLU / Sigmoid / Tanh / etc.
  2. 그라디언트 체크
  3. 학습 파라미터 초기화
  4. 학습 파라미터 최적화 : SGD / Adam / etc.
  5. 과적합 방지 : Dropout / Regularize / etc.

6.1 매개변수 갱신

- 확률적 경사 하강법(SGD, Stochastic Gradient Descent)

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

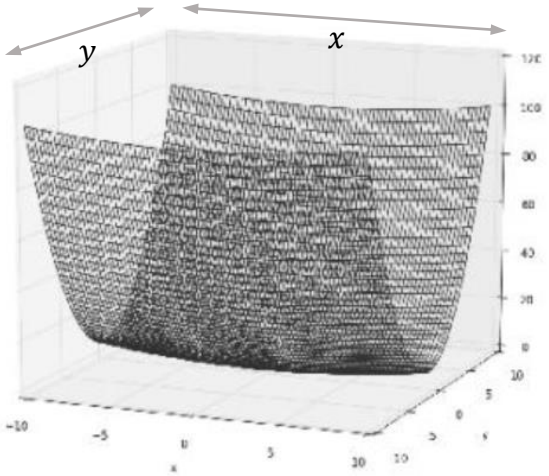


- $\mathbf{W}$: 갱신할 가중치 매개변수
- $\frac{\partial L}{\partial \mathbf{W}}$: $\mathbf{W}$에 대한 손실 함수의 기울기
- η : 학습률
  - 학습률은 실제로는 0.01이나 0.001과 같은 값을 미리 정하여 사용

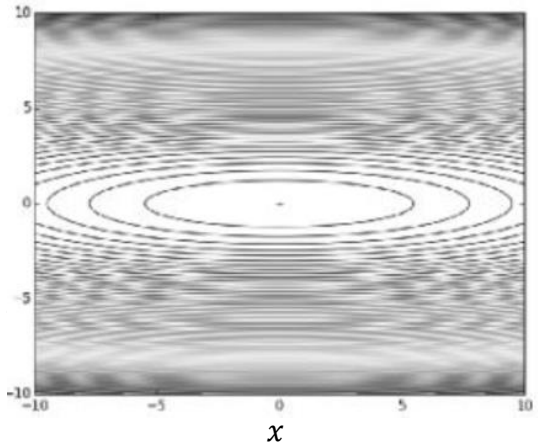
6.1 매개변수 갱신

- SGD의 단점
  - 다음 함수 $f(x,y) = \frac{1}{20}x^2 + y^2$ 의 최소값을 구하는 문제
    - y축 방향의 기울기는 크고 x축 방향의 기울기는 작다.
    - 최소값이 되는 장소는 $(x,y) = (0,0)$ 이지만
      기울기 대부분은 $(0,0)$ 방향을 가리키지 않는다.

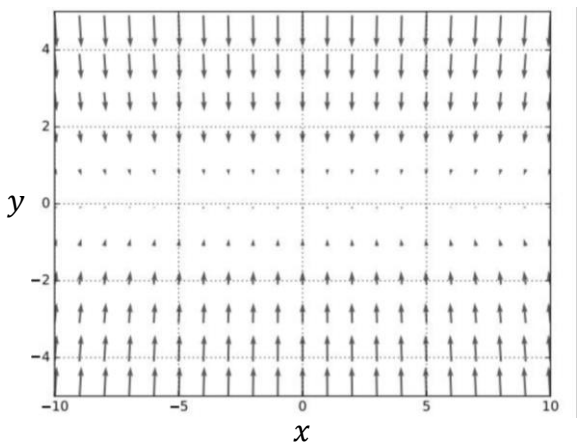
그래프



등고선

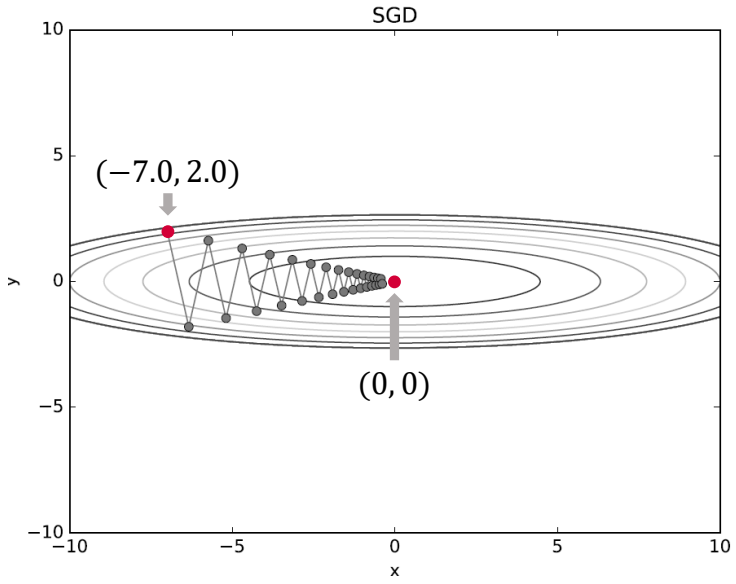


기울기



6.1 매개변수 갱신

- SGD의 단점
  - 임의의 초기값 $(x, y) = (-7.0, 2.0)$ 에서 최솟값 $(0, 0)$ 까지 지그재그로 비효율적인 이동을 한다.
  - SGD는 비등방성anisotropy 함수(방향에 따라 성질이 달라지는 함수)에서는 탐색 경로가 비효율적이다.



근본 원인 : 기울어진 방향이 본래의 최솟값과 다른 방향을 가리킴

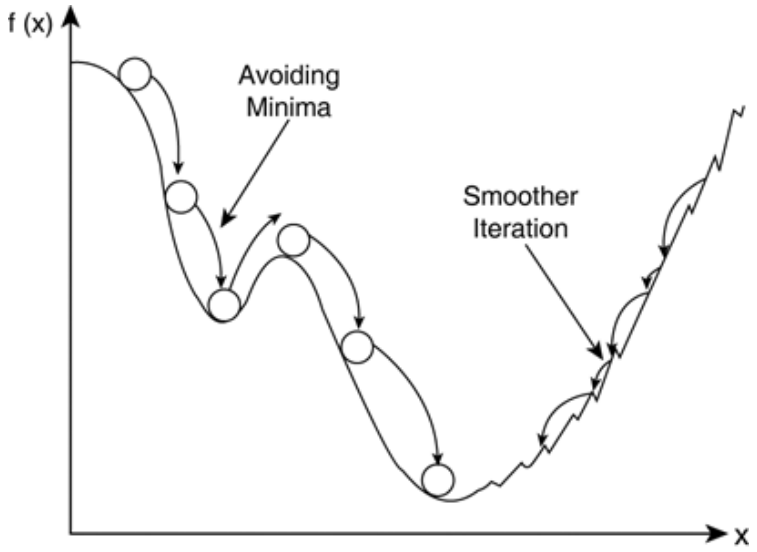
개선 : 모멘텀, AdaGrad, Adam

6.1 매개변수 갱신

- **모멘텀** momentum : '운동량'을 뜻하는 단어

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

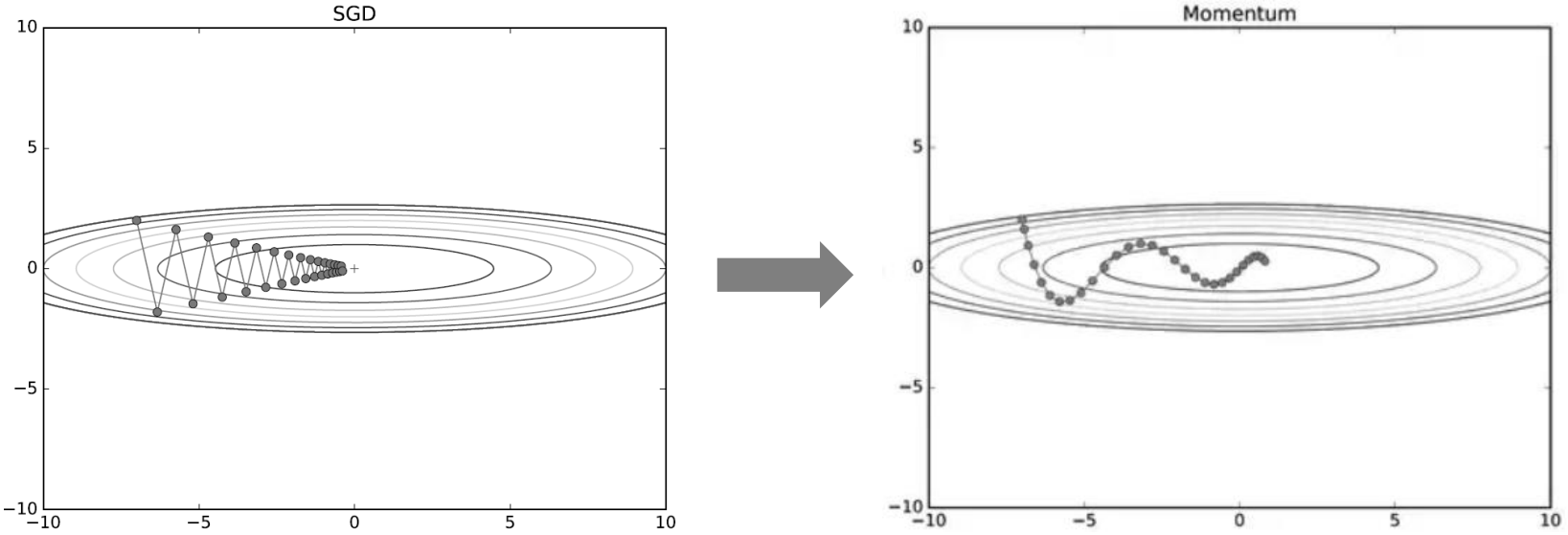
$$\mathbf{W} \leftarrow \mathbf{W} + \mathbf{v}$$



- $\mathbf{W}$: 갱신할 가중치 매개변수
- $\frac{\partial L}{\partial \mathbf{W}}$: $\mathbf{W}$에 대한 손실 함수의 기울기
- η : 학습률
- $\mathbf{v}$: 속도(velocity)
  - 기울기 방향으로 힘을 받아 물체가 가속된다.

6.1 매개변수 갱신

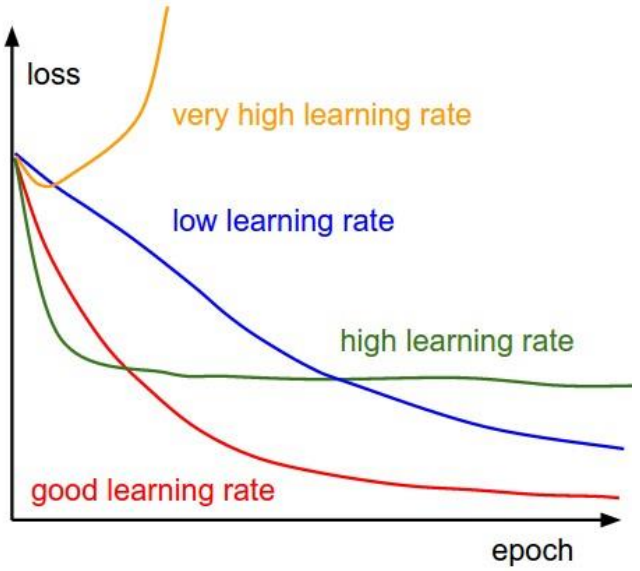
- 모멘텀을 사용한 최적화



- 지그재그 정도가 덜함을 알 수 있다.
- x축의 힘은 아주 작지만 방향은 변하지 않아서 한 방향으로 일정하게 가속되기 때문

6.1 매개변수 갱신

- AdaGrad
  - 신경망에서는 학습률(η)이 중요하다.
    - 너무 작으면 학습시간이 길어지고, 너무 크면 발산하여 학습이 제대로 이뤄지지 않음
  - **학습률 감소** learning rate decay
    - 학습을 진행하면서 학습률을 점차 줄여가는 방법
    - 처음에는 크게 학습하다가 조금씩 작게 학습
    - 가장 간단한 방법은 매개변수 전체의 학습률 값을 일괄적으로 낮추는 것
      → 이를 발전시킨 것이 AdaGrad
  - AdaGrad는 각각의 매개변수에 맞춤형 값을 만들어 준다.



6.1 매개변수 갱신

- AdaGrad
  - 개별 매개변수에 적응적으로 학습률을 조정하면서 학습을 진행

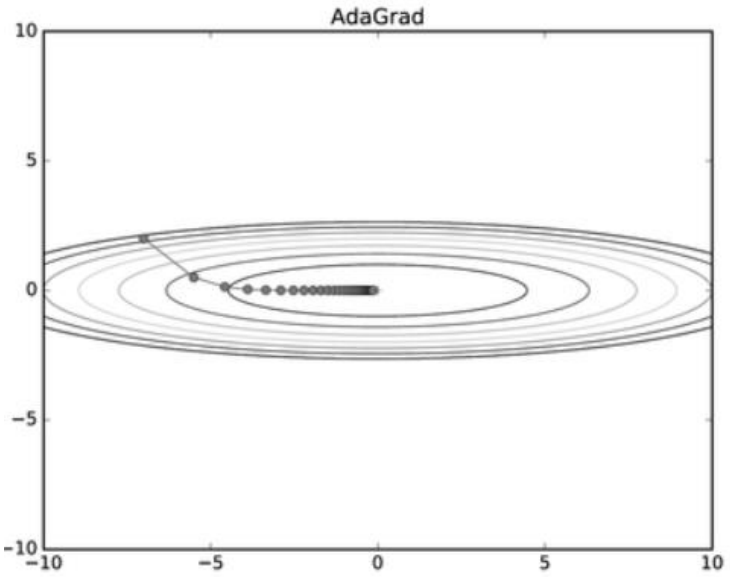
$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

- $\mathbf{W}$: 갱신할 가중치 매개변수
- $\frac{\partial L}{\partial \mathbf{W}}$: $\mathbf{W}$에 대한 손실 함수의 기울기
- η : 학습률
- $\mathbf{h}$: 기존 기울기를 제곱하여 계속 더해줌
  - 매개변수의 원소 중에서 많이 움직인(크게 갱신된) 원소는 학습률을 낮게 만들어줌
  - 학습률 감소가 매개변수의 원소마다 다르게 적용됨을 뜻한다.

6.1 매개변수 갱신

- AdaGrad
  - y축 방향은 기울기가 커서
    처음에는 크게 움직이지만,
    큰 움직임에 비례해 갱신 정도도
    큰 폭으로 작아지도록 조정된다.



AdaGrad는 과거의 기울기를 제곱하여 계속 더해가므로 학습을 진행할수록 갱신 강도가 약해진다. 실제로 무한히 계속 학습한다면 어느 순간 갱신량이 0이 되어 전혀 갱신되지 않게 되는데, 이 문제를 개선한 기법이 RMSProp이다.

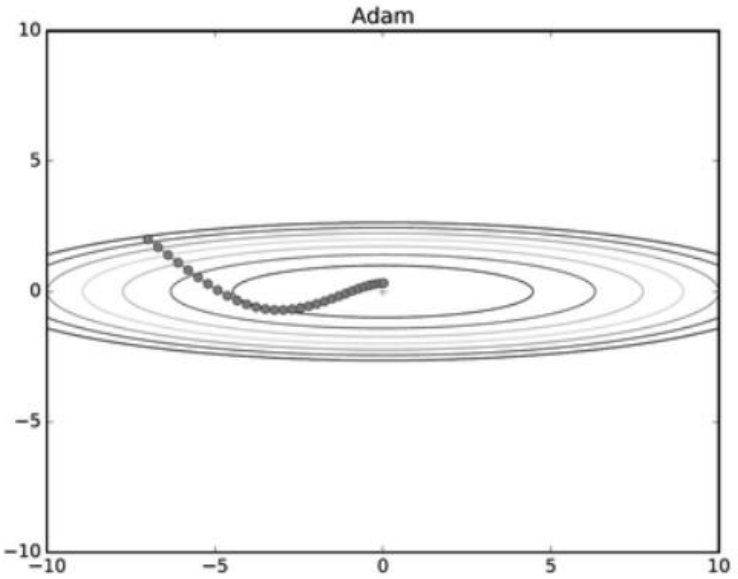
RMSProp은 과거의 모든 기울기를 균일하게 더해가는 것이 아니라, 먼 과거의 기울기는 서서히 잊고 새로운 기울기 정보를 크게 반영한다. 이를 지수이동평균이라 하여, 과거 기울기의 반영 규모를 기하급수적으로 감소시킨다.

6.1 매개변수 갱신

- Adam
  - 직관적으로는 모멘텀과 AdaGrad를 융합한 방법
    - Adaptive moment estimation(적응 모멘트 추정)

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

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



- m_t: momentum처럼 지금까지 계산해온 기울기의 지수평균을 저장
  * 그라디언트의 첫 번째 모멘트(평균)
- v_t : AdaGrad처럼 기울기 제곱값의 지수평균을 저장
  * 그라디언트의 두 번째 모멘트(집중되지 않은 분산)

6.1 매개변수 갱신

- Adam
  - 하이퍼파라미터의 '편향 보정'이 진행된다는 특징이 있다.
  - 편향 보정 bias correction
    - m과 v가 처음에 0으로 초기화
    - 학습의 초반부에는 m_t, v_t가 0에 가깝게 편향되어 있을 것이라고 판단하여 이를 unbiased하게 만들어 주는 작업

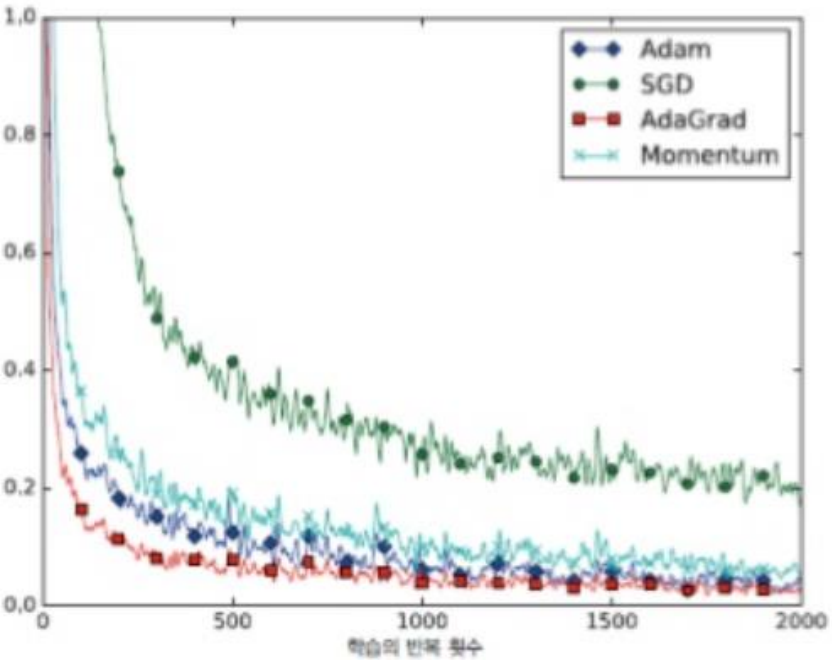
      * 감쇠율이 작은 경우(β_1과 β_2가 1에 가까워짐) 0으로 편향되어 있음
    - 편향 보정된 모멘트 추정치 $\hat{m}_t, \hat{v}_t$를 사용하여 갱신 규칙 생성

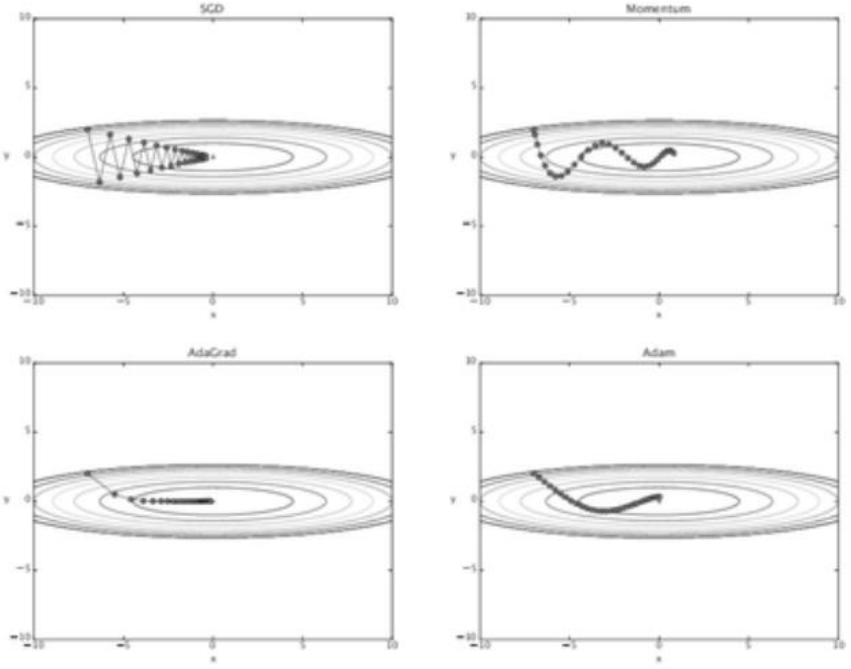
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad \Rightarrow \quad \theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t$$

* 보통 β_1로는 0.9, β_2로는 0.999, ϵ으로는 10^{-8} 정도의 값을 사용

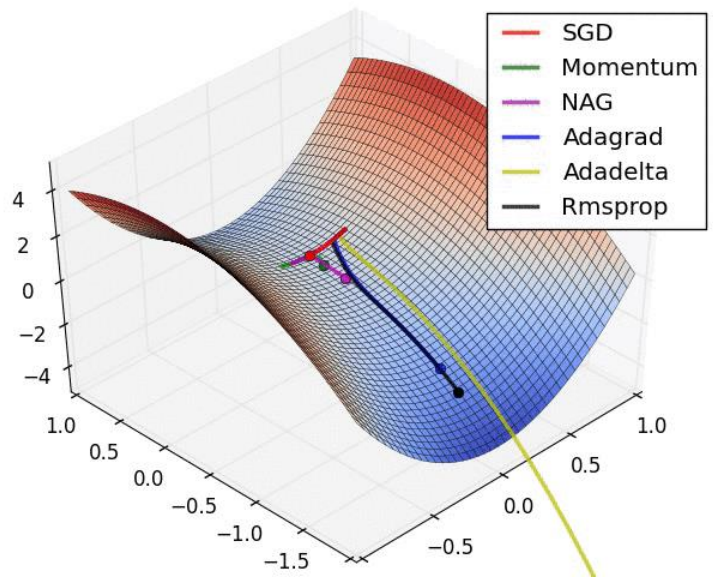
6.1 매개변수 갱신

- 어느 갱신 방법을 이용할 것인가?
  - 풀어야 할 문제에 따라 사용하는 기법이 달라진다.
  - 학습률이나 신경망의 구조(층 깊이 등) 등의 하이퍼파라미터를 어떻게 설정하느냐에 따라서도 달라진다.

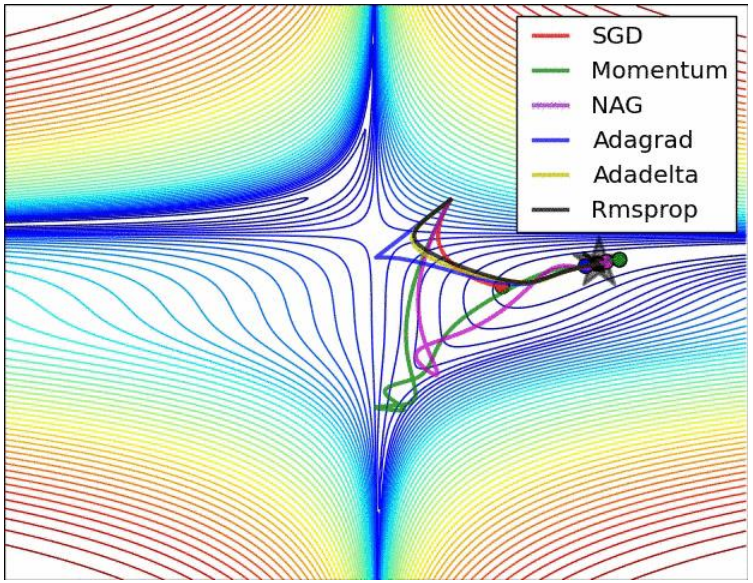




6.1 매개변수 갱신



(a) SGD optimization on loss surface contours



(b) SGD optimization on saddle point

* 손실 함수의 등고선 위에서 각 최적화 알고리즘들의 시간(iteration)에 따른 변화. 모멘텀-기반 방법론들의 "급가속" 행동들을 주목하라. 이게 최적화를 마치 언덕을 내려가는 공처럼 보이게 만든다.

* 목적함수에 안장점(saddle point)가 있을 때의 시각화. 안장점은 그라디언트가 0이지만 헤시안 행렬의 고유치(eigenvalue)에 양수/음수가 섞여있을 때 발생한다. SGD는 안장점에서 빠져나오는 데 매우 힘든 시간을 겪는다. 반대로, RMSprop같은 알고리즘들은 안장의 방향으로 매우 작은 그라디언트를 마주하게 되지만 분모-표준화 성질 덕분에 이 방향의 실질 학습속도를 높아질 수 있고 따라서 이 방향으로 빠져나올 수 있다.

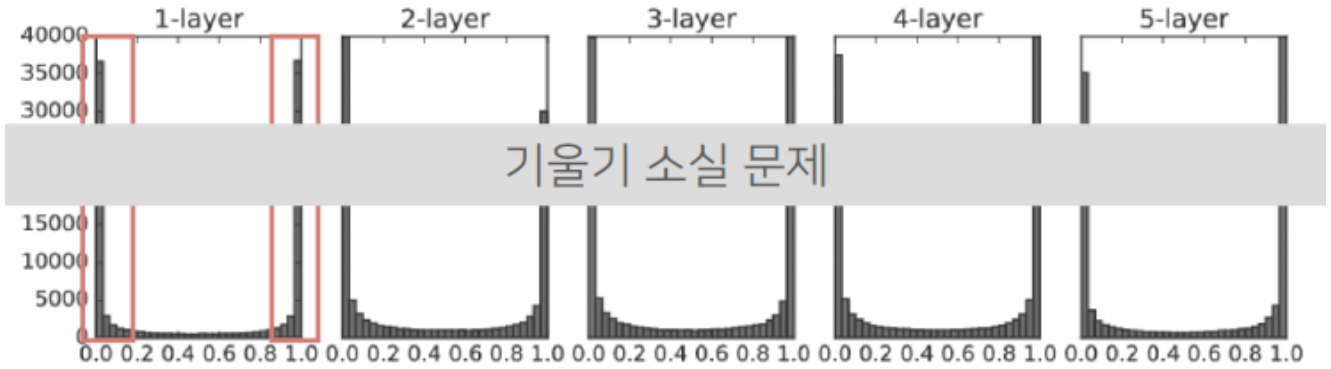
6.2 가중치의 초기값

- **가중치 감소** weight decay
  - 가중치 매개변수의 값이 작아지도록 학습하는 방법
    - 오버피팅이 일어나지 않게 한다.
- **가중치 초기값을 모두 0으로 설정(모두 균일한 값으로 설정)**
  - 학습이 올바르게 이루어지지 않는다.
    - 오차역전파의 모든 가중치 값이 똑같이 갱신되기 때문
    - 가중치들은 같은 값에서 시작하고 갱신해도 여전히 같은 값을 유지
  - 가중치가 고르게 되는 상황(가중치의 대칭적인 구조)을 막으려면 가중치의 초기값을 무작위로 설정해야 한다.

6.2 가중치의 초기값

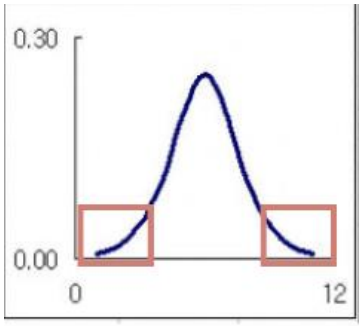
5층 신경망, 각 층의 뉴런 : 100개, 활성화 함수 : 시그모이드

- **표준편차를 1로 한 정규분포의 경우 활성화 값의 분포**

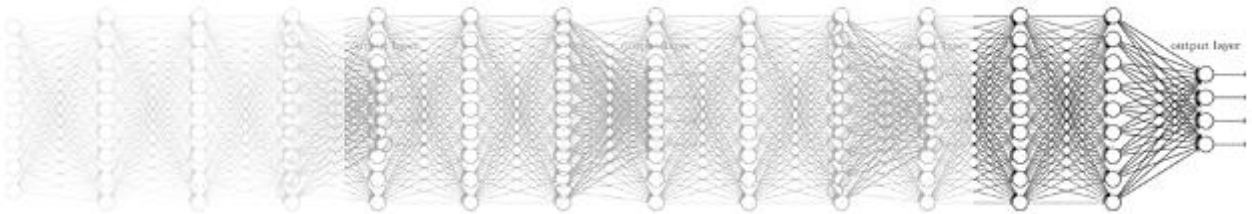


  - 활성화값이 0과 1에 치우쳐 분포
  - 출력이 0 또는 1에 가까울 때 미분값이 0으로 수렴
    - 역전파 시 기울기 값이 점점 작아지다가 사라짐

      → 기울기 소실 vanishing gradient



시그모이드의 편미분 함수



6.2 가중치의 초기값

- **표준편차를 0.01로 한 정규분포의 경우**
  - 0과 1로 치우치지 않아 기울기 소실 문제는 생기지 않음
  - 활성화 값들이 0.5 부근에서 치우침
    - 다수의 뉴런이 거의 같은 값을 출력, 여러 뉴런을 둔 의미가 없어짐

      → 표현력 관점에서 큰 문제

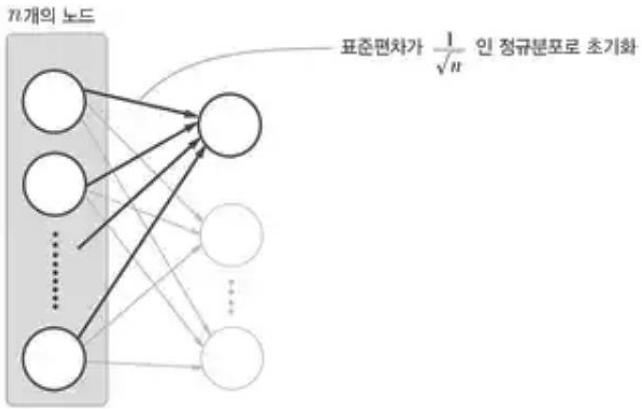
각 층의 활성화 값은 골고루 분포되어야 한다. 층과 층 사이에 적당하게 다양한 데이터가 흐르게 해야 신경망 학습이 효율적으로 이루어지기 때문이다.
반대로 치우친 데이터가 흐르면 기울기 소실이나 표현력 제한 문제에 빠져서 학습이 잘 이뤄지지 않는 경우가 생긴다.

6.2 가중치의 초기값

- Xavier 초기값
  - 일반적인 딥러닝 프레임워크에서 표준적으로 이용
  - 각 층의 활성화 값을 광범위하게 분포시킬 목적으로 가중치의 적절한 분포를 찾음

$$W \sim Uniform(n_{in}, n_{out})$$

$$Var(W) = \frac{1}{n_{in}}$$

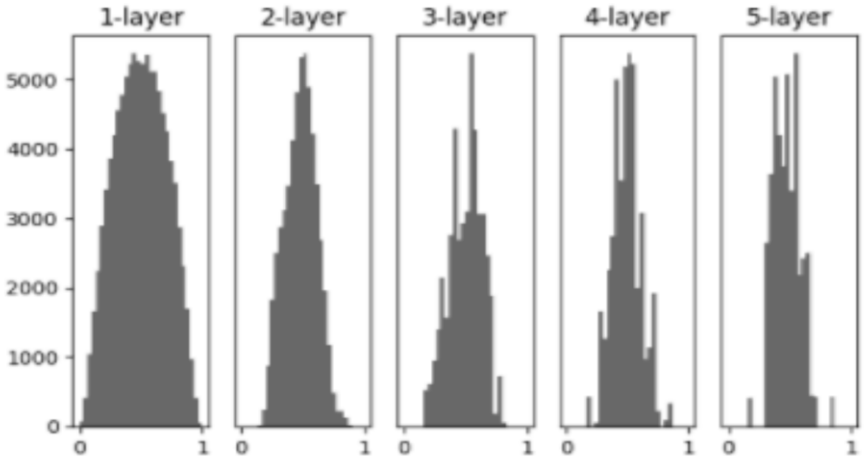


  - 앞 계층의 노드가 n개일 때 표준편차가 $\sqrt{\frac{1}{n}}$ 인 분포를 사용
    - 노드가 많을수록 대상노드의 초기값으로 설정하는 가중치가 좁게 퍼짐

* 논문은 앞 층의 입력 노드 수 외에 다음 층의 출력 노드 수도 고려한 설정값을 제안한다.(Glorot)
Caffe 등의 프레임워크는 앞 층의 노드만으로 계산하도록 단순화되었다.

6.2 가중치의 초기값

- Xavier 초기값
  - 가중치의 초기값으로 'Xavier'를 이용할 때 활성화값 분포
  - 각 층에 흐르는 데이터는 적당히 퍼져 있으므로, 시그모이드 함수의 표현력도 제한받지 않고 학습이 이뤄질 것으로 기대할 수 있음



> 활성화값의 분포를 보면 오른쪽으로 갈수록 형태가 약간 일그러진다.
> 이 일그러짐은 sigmoid 대신 tanh를 사용하면 말끔한 종 모양으로 분포된다.
>
> tanh 함수가 원점(0, 0)에서 대칭인 S 곡선인 반면, sigmoid 함수는 (0, 0.5)에서 대칭인 S 곡선이다. 활성화 함수용으로는 원점에서 대칭인 함수가 바람직하다고 알려져 있다.

6.2 가중치의 초기값

- **He 초기값** – ReLU 를 사용할 때
  - Xavier 초기값은 활성화 함수가 선형인 것을 전제로 한다.
    - sigmoid 함수와 tanh 함수는 좌우 대칭이라 중앙 부근이 선형인 함수이므로 Xavier 초기값이 적당하다.

$$W \sim Uniform(n_{in}, n_{out})$$

$$Var(W) = \frac{2}{n_{in}}$$

  - 앞 계층의 노드가 n개일 때 표준편차가 $\sqrt{\frac{2}{n}}$ 인 분포를 사용
  - ReLU는 음의 영역이 0이므로 음수 신호를 없앤다.
    - 따라서 활성화 값을 더 넓게 분포시키기 위해 2배의 계수가 필요하다.

- **Glorot 초기값**

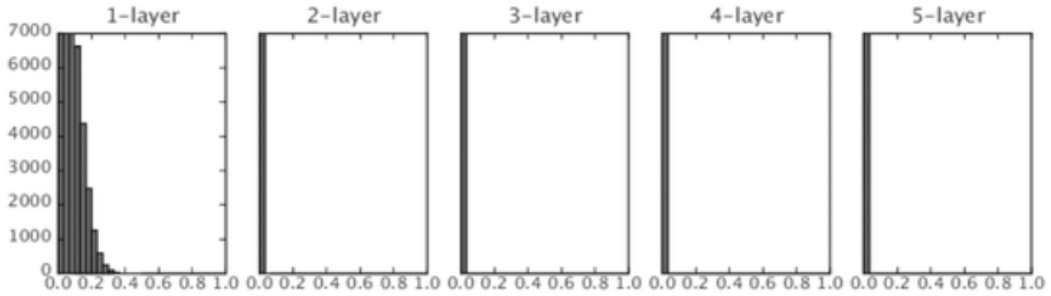
$$W \sim Uniform(n_{in}, n_{out})$$

$$Var(W) = \frac{2}{n_{in} + n_{out}}$$

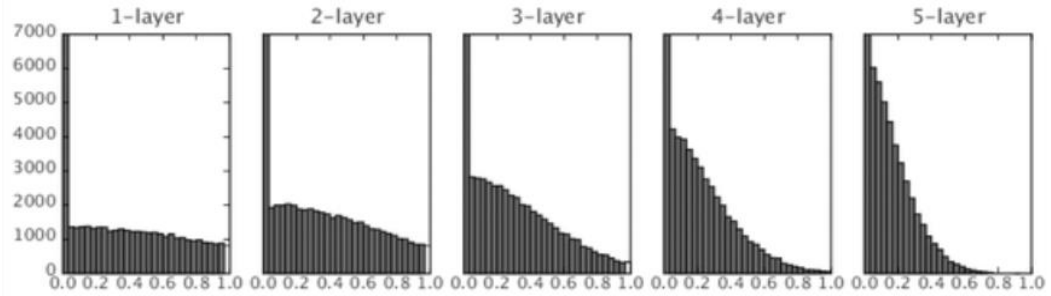
- **RBM** Restricted Boltzmann machine
  - DBN – pre-training(Fine tuning)

6.2 가중치의 초기값

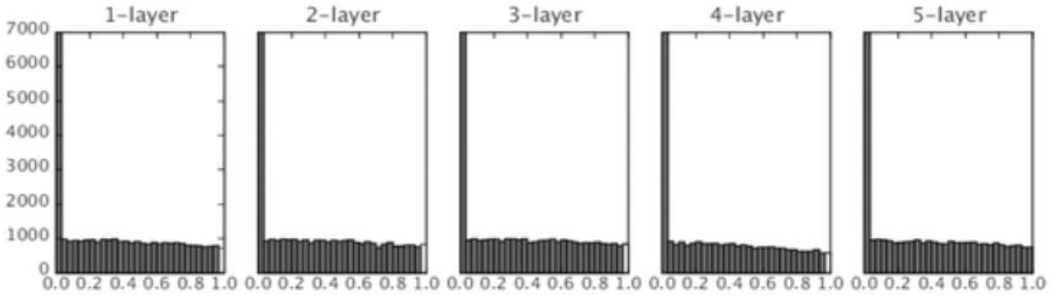
- 가중치 초기값에 따른 활성화값의 분포



표준편차가 0.01인 정규분포를 가중치 초깃값으로 사용한 경우



Xavier 초깃값을 사용한 경우



He 초깃값을 사용한 경우

* std=0.01
신경망에 아주 작은 데이터가 흐름
(0 근처로 밀집한 데이터)
→ 역전파 시 가중치의 기울기 역시 작아짐
→ 실제로 학습이 거의 이루어지지 않는다.
(가중치가 거의 갱신되지 않는다.)

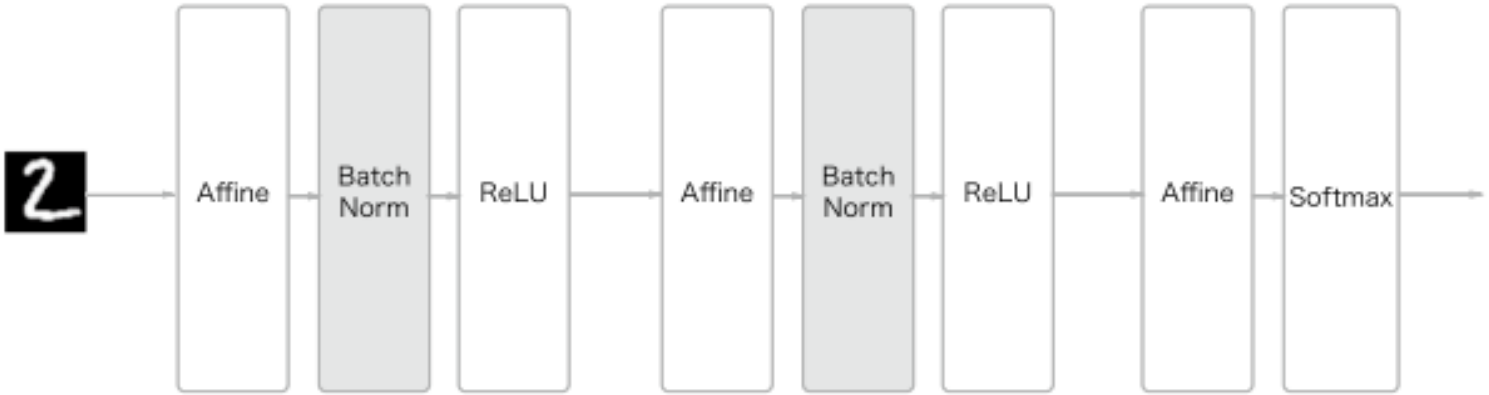
* Xavier
층이 깊어지면서 치우침이 조금씩 커짐
→ 활성화값의 치우침도 커짐
→ '기울기 소실' 문제를 일으킴

* He
층이 깊어져도 분포가 균일하게 유지
→ 역전파 때도 적절한 값이 나올 것

6.3 배치 정규화

배치 정규화 batch normalization

- 각 층의 활성화값이 적당히 분포되도록 강제로 조정하는 것
- 배치 정규화의 장점
  - 학습을 빨리 진행할 수 있다(학습속도 개선)
  - 초기값에 크게 의존하지 않는다
  - 오버피팅을 억제한다(드롭아웃 등의 필요성 감소)
- '배치 정규화batch norm 계층'을 신경망에 삽입



6.3 배치 정규화

- **배치 정규화 알고리즘**
  - 학습 시 미니배치를 단위로 정규화
    - 데이터 분포가 평균이 0, 분산이 1이 되도록 정규화

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

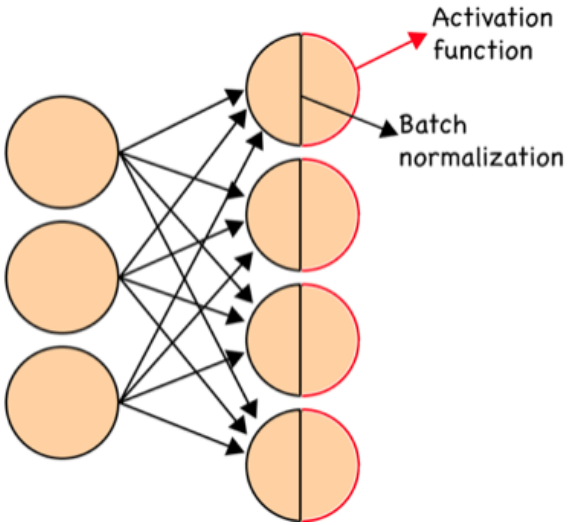
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$$

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

- m개 입력데이터의 집합인 미니배치 :
  $$\boldsymbol{B} = \{x_1, x_2, \ldots, x_m\}$$
- 평균 : $\boldsymbol{\mu_B}$ / 분산 : $\boldsymbol{\sigma_B^2}$
- ε : 작은 값 (ex. 10^{-7})
- 입력데이터를 평균 0, 분산 1인 데이터 $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m\}$ 으로 변환
- 정규화된 데이터에 고유한 확대$^{\text{scale}}(\gamma)$와 이동$^{\text{shift}}(\beta)$ 변환을 수행
  - $\gamma = 1, \beta = 0$ 부터 시작, 학습하면서 조정

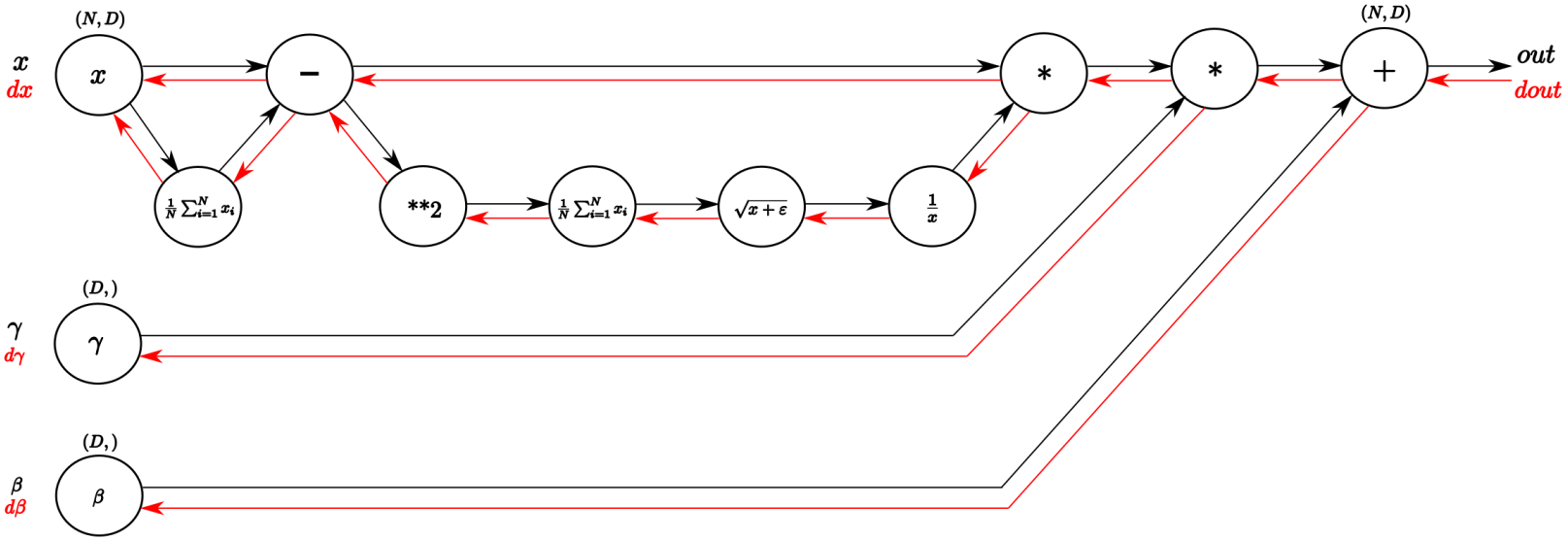
6.3 배치 정규화

- 배치 정규화 알고리즘
  - 이 처리를 활성화 함수의 앞(또는 뒤)에 삽입함으로써 데이터 분포가 덜 치우치게 할 수 있다.



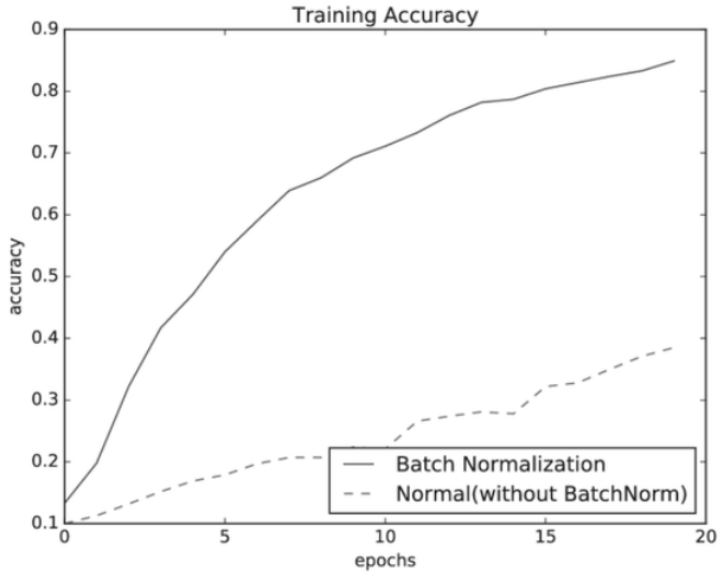
* Sergey Ioffe and Christian Szegedy(2015) : Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.
* Dmytro Mishkin and Jiri Matas(2015) : All you need is a good init.

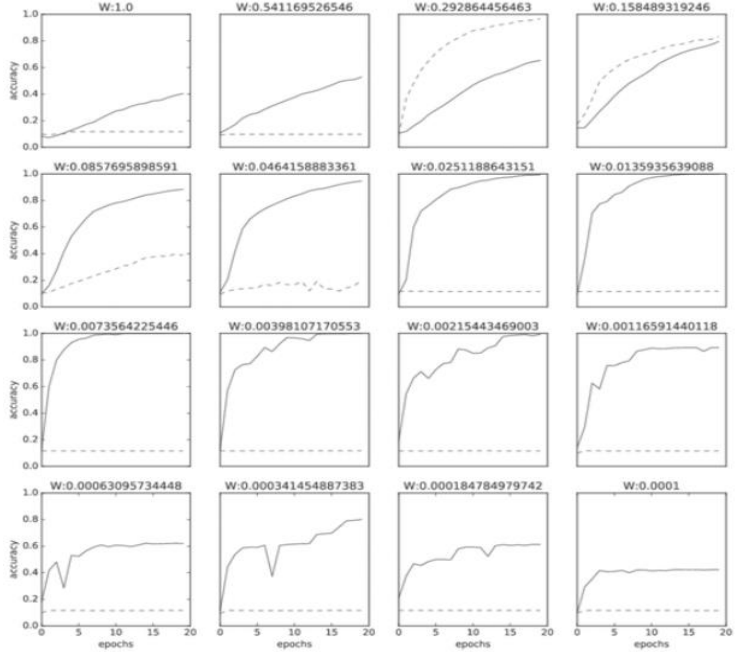
배치 정규화의 계산 그래프



6.3 배치 정규화

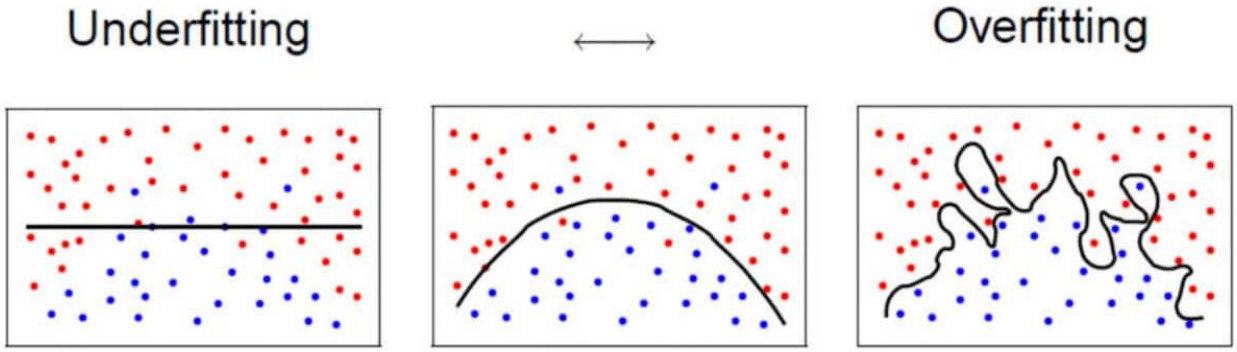
- 배치 정규화의 효과
  - 배치 정규화를 사용할 때의 학습 진도가 빠름
    - 배치 정규화를 이용하지 않는 경우 초기값이 잘 분포되어 있지 않으면 학습이 전혀 진행되지 않는 경우도 있다.
  - 배치 정규화를 사용하면 학습이 빨라지며, 가중치 초기값에 크게 의존하지 않아도 된다.





6.4 바른 학습을 위해

- **오버피팅** overfitting
  - 신경망이 훈련 데이터에만 지나치게 적응되어 그 외의 데이터에는 제대로 대응하지 못하는 상태
    - 기계학습은 범용 성능을 지향, 훈련 데이터 이외에 대해서도 바르게 식별해내는 모델이 바람직
- **오버피팅이 발생하는 경우**
  - 매개변수가 많고 표현력이 높은 모델
  - 훈련 데이터가 적음



6.4 바른 학습을 위해

- **오버피팅 방지를 위한 방법**
  - 모델 크기 줄이기
    - 층, 뉴런의 개수 등 학습해야 할 파라미터의 개수를 줄임
  - Early stopping
    - 학습을 일찍 중단
  - 가중치 감소
    - 학습 파라미터의 값이 크면 그에 상응하는 큰 패널티를 부과
      * L2 Regularization, L1 Regularization, L∞ Regularization
  - Dropout
    - 일부 뉴런을 꺼서 학습, 일종의 앙상블ensemble 효과를 냄
    - 학습 시 삭제할 뉴런을 무작위로 선택, 테스트 시 모든 뉴런을 사용

6.4 바른 학습을 위해

- **가중치 감소** weight decay
  - 오버피팅은 가중치 매개변수의 값이 커서 발생하는 경우가 많다.
    - 따라서 가중치가 클수록 그에 상응하는 큰 패널티를 부과하여 오버피팅을 억제한다.
- **L2 법칙에 따른 가중치 감소**
  - 신경망 학습의 목적 : 손실 함수의 값을 줄이는 것
    - 패널티 부과란 손실 함수의 결과에 $\frac{1}{2}\lambda\mathbf{W}^2$을 더하는 것 ($\mathbf{W}$; 가중치 벡터)
      * λ(람다)는 정규화의 세기를 조절하는 하이퍼파라미터, 클수록 패널티 커짐
      * $\frac{1}{2}$ 은 $\mathbf{W}$로 미분했을 때 $2\lambda\mathbf{W}$가 아닌 $\lambda\mathbf{W}$의 값을 갖도록 하기 위함
      * 오차역전파 시 계산한 기울기에 $\lambda\mathbf{W}$를 더하게 되고 이에 따라서 가중치 갱신은 $W \leftarrow W - \eta\left(\frac{\partial L}{\partial W} + \lambda W\right)$ 가 되므로 그만큼 가중치가 더 조절됨

6.4 바른 학습을 위해

- 가중치 감소 – 정규화 기법
  - L2 regularization $Loss_{new} = Loss_{old} + \alpha \sum_i w_i^2$
  - L1 regularization $Loss_{new} = Loss_{old} + \alpha \sum_i |w_i|$
  - Elastic net regularization $Loss_{new} = Loss_{old} + \alpha \sum_i w_i^2 + \beta \sum_i |w_i|$
  - Max norm constraint (L∞) $w = w \times min(1, c/||w||_2)$

그림 6-20 훈련 데이터(train)와 시험 데이터(test)의 에폭별 정확도 추이

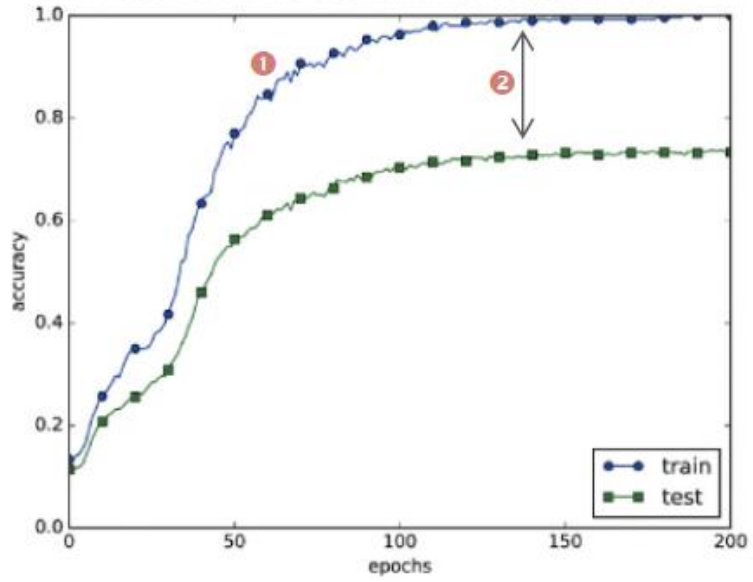
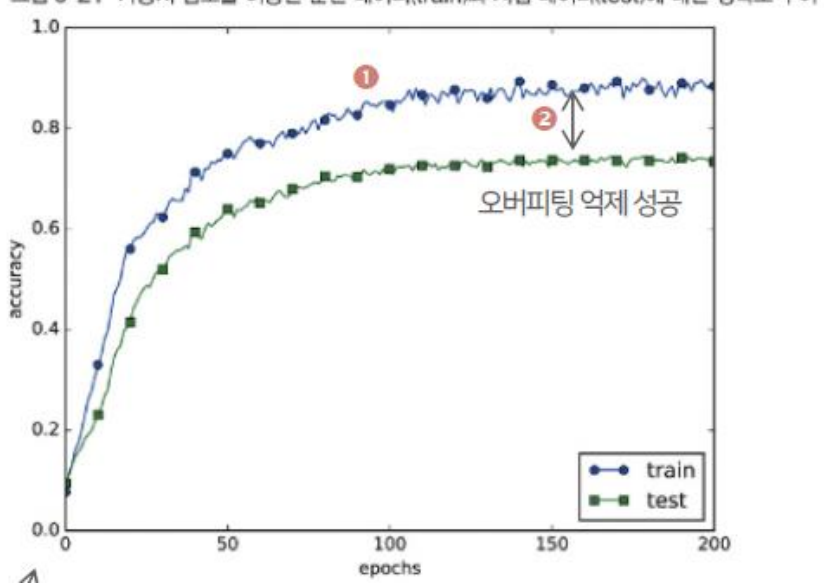


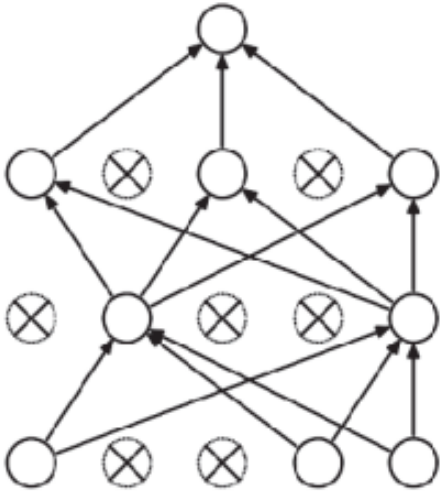
그림 6-21 가중치 감소를 이용한 훈련 데이터(train)와 시험 데이터(test)에 대한 정확도 추이



6.4 바른 학습을 위해

▪ 드롭아웃 dropout

- 신경망 모델이 복잡해지면 가중치 감소만으로는 대응이 어려움
- 은닉층의 뉴런을 임의로 삭제하면서 학습하는 방법
- 훈련 때는 삭제할 뉴런을 무작위로 선택하고, 시험 때는 모든 뉴런에 신호를 전달
- 시험 때는 각 뉴런의 출력에 훈련 때 삭제한 비율을 곱하여 출력



(b) 드롭아웃을 적용한 신경망

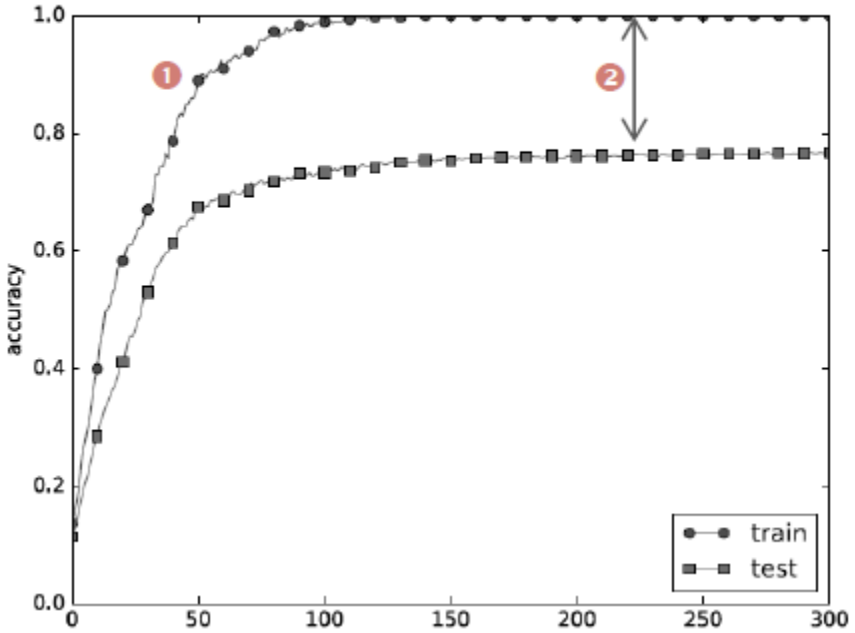
앙상블 학습ensemble learning 은 개별적으로 학습시킨 여러 모델의 출력을 평균 내어 추론하는 방식이다(투표 등 다른 방식도 사용). 앙상블 학습을 수행하면 신경망의 정확도가 몇 % 정도 개선된다는 것이 실험적으로 알려져 있다.

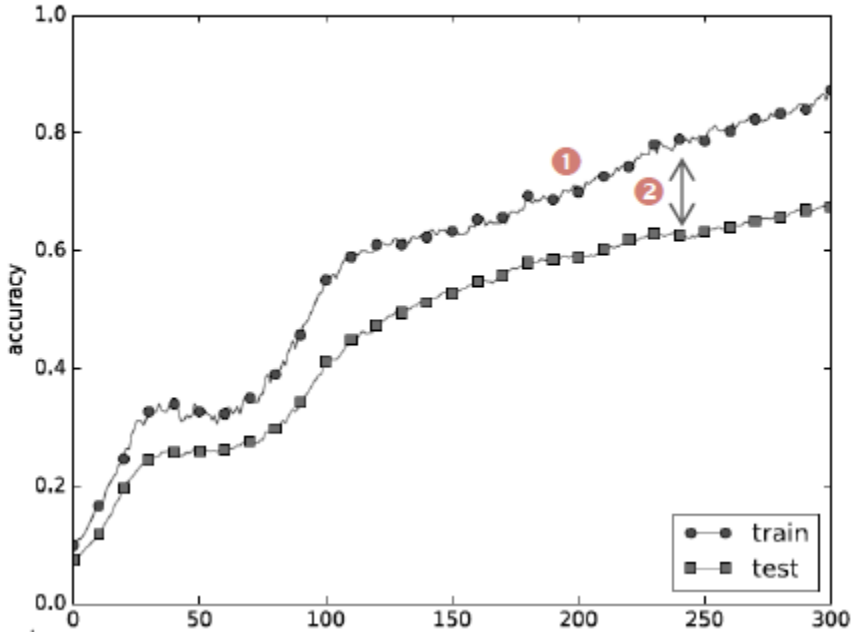
드롭아웃이 학습 때 뉴런을 무작위로 삭제하는 행위가 매번 다른 모델을 학습시키는 것으로 해석할 수 있기 때문에 앙상블 학습과 밀접하다. 그리고 추론 때는 뉴런의 출력에 삭제한 비율을 곱하여 앙상블 학습에서 여러 모델의 평균을 내는 것과 같은 효과를 얻는다.

6.4 바른 학습을 위해

- 드롭아웃 적용
  - 훈련 데이터와 시험 데이터에 대한 정확도 차이가 줄어듦
  - 훈련 데이터에 대한 정확도가 100%에 도달하지 않게 됨
    - 드롭아웃을 이용하면 표현력을 높이면서도 오버피팅을 억제함

그림6-23 왼쪽은 드롭아웃 없이, 오른쪽은 드롭아웃을 적용한 결과(dropout_rate = 0.15)





6.5 적절한 하이퍼파라미터 값 찾기

- **하이퍼파라미터** hyperparameter, 초모수
  - 뉴런 수, 배치 크기, 매개변수 갱신 시의 학습률과 가중치 감소 등
  - 하이퍼파라미터를 결정하기까지는 많은 시행착오를 겪음
- **검증 데이터** validation data
  - 하이퍼파라미터를 평가할 때는 시험데이터를 사용해서는 안됨
    - 하이퍼파라미터 값이 시험데이터에 오버피팅되기 때문
  - 하이퍼파라미터를 조정하기 위한 전용 확인 데이터가 필요
  - 훈련데이터를 20% 정도로 나누어 사용

> • 훈련 데이터 : 매개변수(가중치와 편향) 학습
> • 검증 데이터 : 하이퍼파라미터 성능 평가
> • 시험 데이터 : 신경망의 범용 성능 평가, 마지막에(이상적으로는 한 번만) 이용

6.5 적절한 하이퍼파라미터 값 찾기

- 하이퍼파라미터 최적화
  - '최적값'이 존재하는 범위를 조금씩 줄여간다는 것
  - 대략적인 범위를 설정하고 → 무작위로 하이퍼파라미터 값을 골라낸(샘플링) 후 → 그 값으로 정확도를 평가
  - 최적화 단계

    0단계 : 하이퍼파라미터 값의 범위를 설정

    1단계 : 설정된 범위에서 하이퍼파라미터의 값을 무작위로 추출

    2단계 : 1단계에서 샘플링한 값을 사용하여 학습하고,
    검증데이터로 정확도를 평가(단, 에폭은 작게 설정)

    3단계 : 1단계와 2단계를 특정 횟수 반복하며,
    그 정확도의 결과를 보고 하이퍼파라미터의 범위를 좁힌다.

> 하이퍼파라미터 최적화에서는 그리드 서치grid search 같은 규칙적인 탐색보다는 무작위로 샘플링해 탐색하는 편이 좋은 결과를 낸다고 알려져 있다.
> 이는 최종 정확도에 미치는 영향력이 하이퍼파라미터마다 다르기 때문이다.

6.5 적절한 하이퍼파라미터 값 찾기

- **베이즈 최적화** Bayesian optimization

  Practical Bayesian Optimization of Machine Learning Algorithms(NIPS 2012)

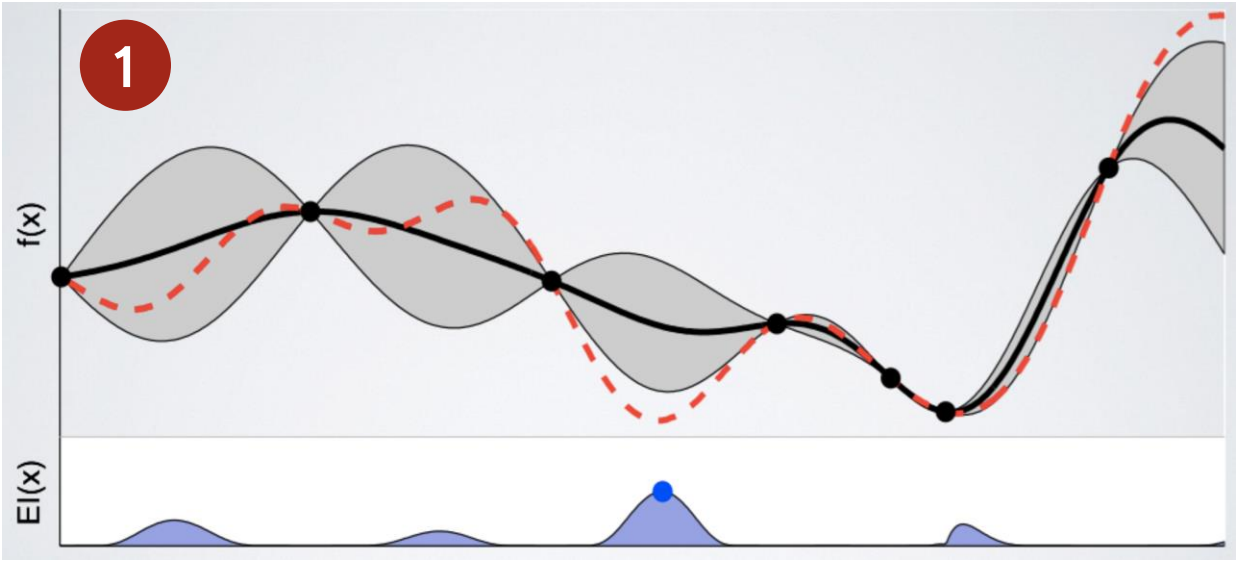
  - 알려지지 않은 'Black box' function을 최적화할 때 사용

$$x^* = \arg\min_{x \in X} f(x).$$

1. 지금까지 관측된 데이터들을 통해, 전체 $f(x)$를 어떤 방식을 사용해 추정한다.
2. $f(x)$를 더 정밀하게 예측하기 위해 다음으로 관측할 지점을 어떤 decision rule을 통해 선택한다.
3. 새로 관측한 지점을 데이터에 추가하고, 적절한 stopping criteria에 도달할 때까지 다시 1로 돌아가 반복한다.

6.5 적절한 하이퍼파라미터 값 찾기

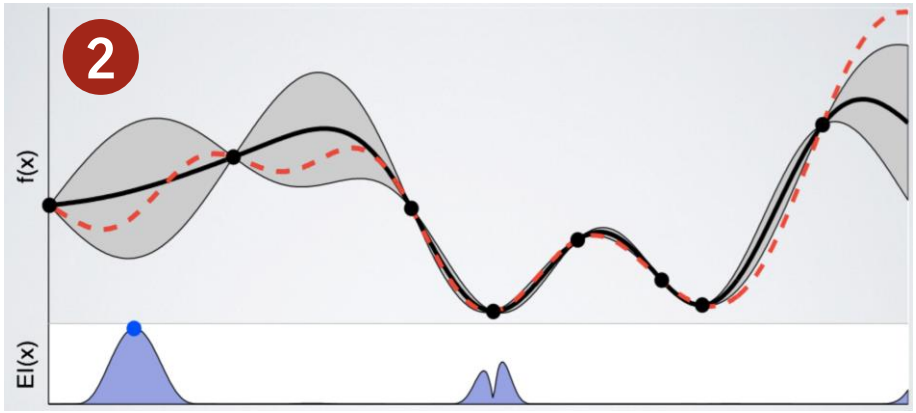
- **베이즈 최적화** Bayesian optimization
  - 빨간색 점선 : unknown black box function $f(x)$
  - 까만색 실선 : 지금까지 관측한 데이터를 바탕으로 예측한 $\hat{f}(x)$
  - 회색 영역 : $f(x)$가 존재할 confidence bound(신뢰경계)
  - $EI(x)$: acquistion function



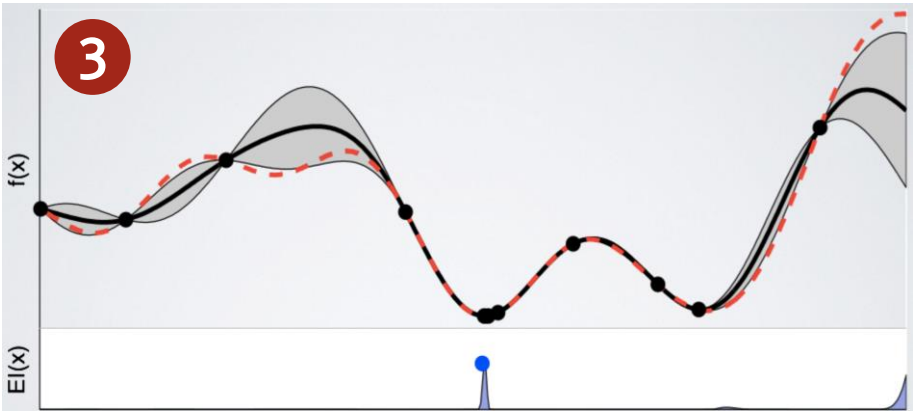
① 지금까지 관측한 데이터를 바탕으로, ($EI(x)$(acquisition function) 값이 제일 큰) 파란색 점이 찍힌 부분을 관측하는 것이 가장 좋다는 것을 알 수 있다.

6.5 적절한 하이퍼파라미터 값 찾기

- **베이즈 최적화** Bayesian optimization



② ①에서 $EI(x)$값이 제일 컸던 지점의 $f(x)$값을 관측하고 추정을 갱신한다.
함수의 불확실성을 의미하는 회색 영역이 크게 감소했음을 알 수 있다.
그러나 여전히 좌측과 우측에는 불확실한 부분이 꽤 큼을 알 수 있다.
다시 한 번 다음 관측할 지점을 $EI(x)$를 통해 고른다.



③ 계속 갱신을 진행한 결과, 추정 $\hat{f}(x)$와 실제 function $f(x)$가 거의 흡사해졌다.
이제 여태까지 관측한 지점 중 best point를 $\operatorname{argmin} f(x)$로 선택한다.

6.5 적절한 하이퍼파라미터 값 찾기

- **베이즈 최적화** Bayesian optimization
  - 실용적으로 쓰기는 어렵다.
    - Kernel/Acquisition function 등 모델에 따라 성능이 크게 바뀐다.
    - 베이즈 최적화 자체도 하이퍼파라미터가 있어서 이를 튜닝해야 한다.
    - 순차적 갱신을 해야 하기 때문에 병렬적인 처리가 되지 않는다.
  - 아래 개념들을 알고 있어야 한다.
    * Stochastic process(Random process)
    * Gaussian process(GP) & Kernel function of GP
    * Bayesian optimization & Acquisition function
    * Markov Chain Monte Carlo(MCMC)
  - 실험적 결과
    * Kernel function : Matern 5/2 kernel
    * Acquisition function : EI
    * GP hyperparameter : marginalized acquisition function

6.6 정리

이번 장에서 배운 내용

- 매개변수 갱신 방법에는 확률적 경사 하강법(SGD) 외에도 모멘텀, AdaGrad, Adam 등이 있다.
- 가중치 초기값을 정하는 방법은 올바른 학습을 하는 데 매우 중요하다.
- 가중치의 초기값으로는 'Xavier 초기값'과 'He 초기값'이 효과적이다.
- 배치 정규화를 이용하면 학습을 빠르게 진행할 수 있으며, 초기값에 영향을 덜 받게 된다.
- 오버피팅을 억제하는 정규화 기술로는 가중치 감소와 드롭아웃이 있다.
- 하이퍼파라미터 값 탐색은 최적 값이 존재할 법한 범위를 점차 좁히면서 하는 것이 효과적이다.