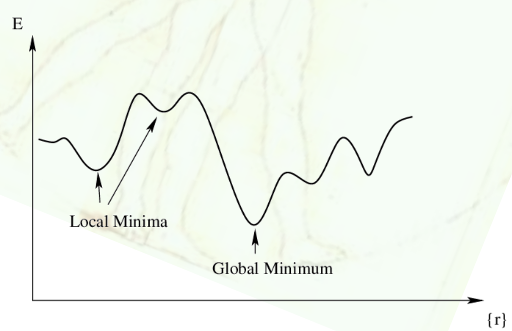
Training Neural Networks 2 (Optimizer)

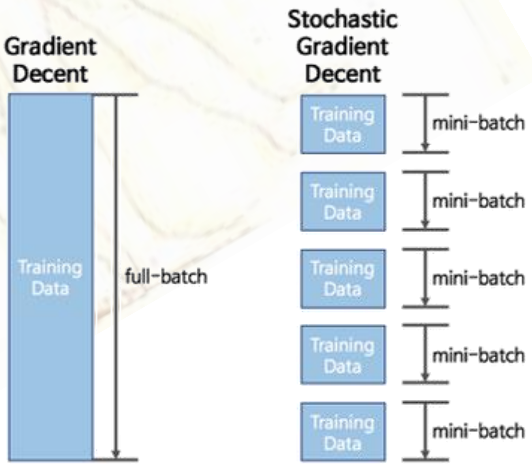
Pattern Recognition & Machine Learning Laboratory
Ji-Sang Hwang, July 21, 2021

Optimizer (1/6)

- 옵티마이저(Optimizer)란?
  - 딥러닝에서 모델을 학습시킨다는 것은 ‘가중치(Weight)’와 ‘편향(Bias)’ 같은 하이퍼 파라미터를 최적화 시키는 일
  - 최적화(Optimization)란 목적함수(Objective function)의 결과값을 찾는 과정
  - 최적화를 위해 손실 함수(Loss function)를 활용하여 가중치가 얼마나 잘 설정되어 있는지 확인함
- 경사하강법(Gradient descent)
  - 손실 함수의 기울기(Gradient)를 구하여 기울기의 절대값이 낮은 쪽으로 이동시켜 최적값(최솟값)에 이를 때까지 이를 반복하는 방법
  - **Batch gradient descent**
    - 스텝마다 전체 학습 데이터를 이용하여 연산을 진행
    - 연산양이 많아 학습 속도가 느림



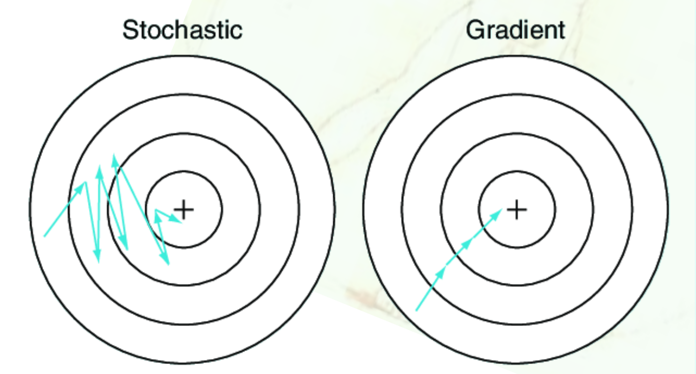
지역최솟값와 전역최솟값



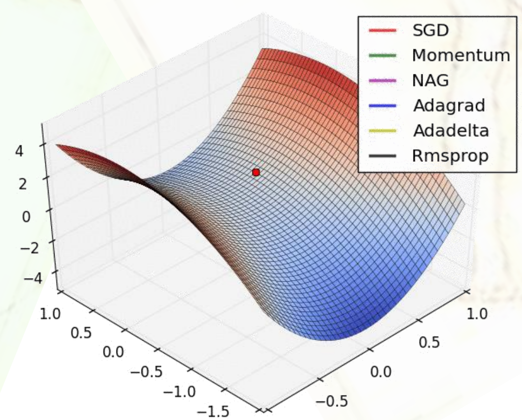
Full-batch와 Mini-batch

Optimizer (2/6)

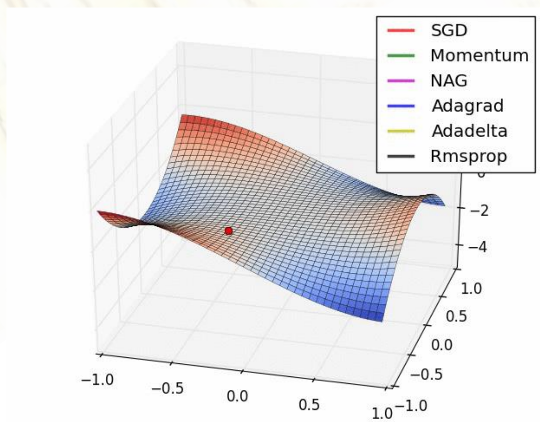
- 확률적 경사하강법(Stochastic gradient descent (SGD))
  - Batch gradient descent의 문제를 해결하기 위하여 등장
    - $\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\eta}\nabla\boldsymbol{J}(\boldsymbol{\theta})$
      - » $\boldsymbol{\theta}$: 모델의 파라미터 세트, $\boldsymbol{\eta}$: 학습률(Learning rate), $\boldsymbol{J}(\boldsymbol{\theta})$: 손실 함수
  - 전체 학습 데이터를 여러 개의 Mini-Batch로 나눠서 경사 학습
  - Mini-Batch를 어떻게 선택하는지에 따라 결과값이 달라지기 때문에 '확률적(Stochastic) 이다'라고 표현함
  - SGD의 문제점
    - 손실 함수의 최솟값을 찾는데 불안정하고 비효율적인 탐색경로를 가짐
    - 지역최솟값(Local minima), 안장점(Saddle point)에서 벗어나지 못함
    - 모든 파라미터에서 학습 보폭(Step size)이 같음



SGD와 GD 비교



지역최솟값에서의 SGD



안장점에서의 SGD

Optimizer (3/6)

SGD + Momentum

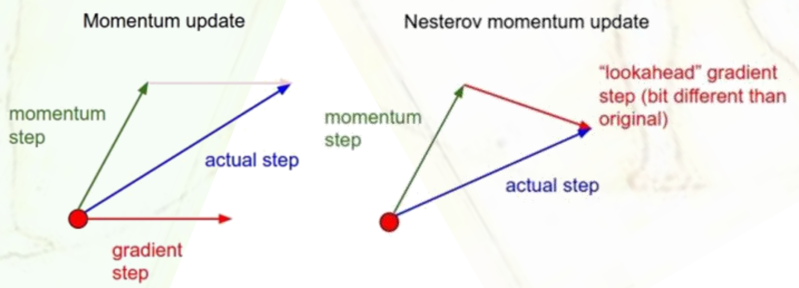
- 기존의 **SGD**에 속도**(Velocity)**라는 개념을 추가하여 관성**(Momentum)** 효과를 얻음
  - $\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v_t}$
  - $\boldsymbol{v_t} = \boldsymbol{\gamma v_{t-1}} + \boldsymbol{\eta} \nabla \boldsymbol{J(\theta)}$
    - » $\boldsymbol{\theta}$: 모델의 파라미터 세트, $\boldsymbol{\eta}$: 학습률, $\boldsymbol{J(\theta)}$: 손실 함수, $\boldsymbol{v_t}$: 속도
- $\boldsymbol{v_t}$를 통해 바로 멈추지 않고 **'Overshooting'**하여 지역최솟값과 안장점을 통과함
- **SGD** 대비 더 빠르고 부드럽게 **Convex** 최솟값을 찾을 수 있음
- 전역최솟값**(Global Minimum)**에서 바로 멈추거나 늦춰지지 않음

Nesterov Accelerated Gradient (NAG)

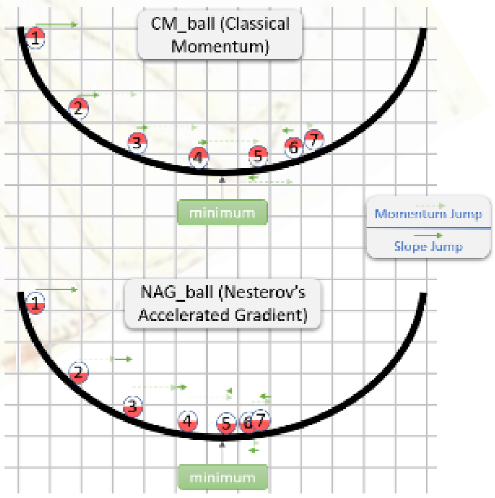
- **SGD+Momentum**을 변형시킨 옵티마이저
- **Momentum Step**이 진행된 위치에서 **Gradient Step**을 진행
- 전역최솟값에서 빠르게 수렴하는 효과를 얻음

```
vx = 0
while True:
    dx = compute_gradient(x)
    vx = rho * vx + dx
    x -= learning_rate * vx
```

SGD + Momentum 알고리즘



SGD+Momentum과 NAG



Convex 최솟값에서의 비교

Optimizer (4/6)

Adaptive Gradient (AdaGrad)

- 각각의 파라미터가 변한 만큼을 학습에 반영하기 위해 고안
  - $\theta_{t+1} = \theta - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla_\theta J(\theta_t)$
  - $G_t = G_{t-1} + (\nabla_\theta J(\theta_t))^2$
    - θ : 모델의 파라미터 세트, η : 학습률, $J(\theta)$: 손실 함수, G_t : 기울기의 제곱합
- 단계를 밟을수록 G_t 가 커지며 학습이 느려지는 현상 발생

Root Mean Squared Propagation (RMSProp)

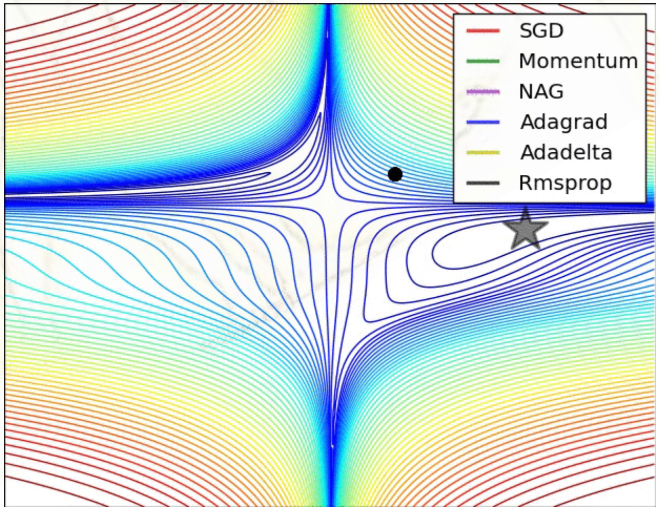
- 동일하게 G_t 를 사용하지만 누적 과정에서 **Decay**를 진행함
- G_t 가 단순 누적되어 무한대로 발산하는 것을 방지함

AdaDelta

- 파라미터를 변화시킬 때, 단위를 맞추기 위해 탄생
- 초기 학습률을 정의하지 않아도 됨

```
grad_squared = 0
while True:
   dx = compute_gradient(x)
   grad_squared += dx * dx
    x -= learning_rate * dx / (np.sqrt(grad_squared) + 1e-7)
```

AdaGrad 알고리즘



지금까지 알아본 옵티마이저

Optimizer (5/6)

➢ Adaptive Moment Estimation (Adam)

- **SGD+Momentum**의 v_t와 **RMSProp**의 G_t를 모두 사용한 옵티마이저
  - $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta J(\theta)$
  - $v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_\theta J(\theta))^2$
  - $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t \ (\hat{m}_t = \frac{m_t}{1-\beta_1^t},\ \hat{v}_t = \frac{v_t}{1-\beta_2^t})$
    - » β : Decay Constant, m_t : Momentum, v_t : Adaptive Learning Rate
- 불편추정치**(Unbiased estimate)**를 통하여 초기에 m_t 에서 이동하지 못하는 것과 v_t 에서 너무 큰 스텝을 밟는 것을 방지함
- **SGD+Momentum**처럼 최솟값에 **'Overshooting'**하는 모습과 **RMSPorp**처럼 최솟값을 향해 궤도를 조정하는 모습을 보임

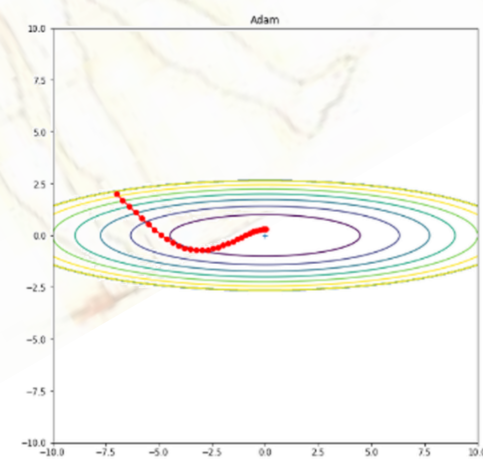
```
first_moment = 0
second_moment = 0
while True:
    dx = compute_gradient(x)
    first_moment = beta1 * first_moment + (1 – beta1) * dx
    second_moment = beta1 * second_moment + (1 – beta2) * dx * dx
    first_unbias = first_moment / (1 – beta1 ** t)
    second_unbias = second_moment / (1 – beta2 ** t)
    x -= learning_rate * first_unbias / (np.sqrt(second_unbias) + 1e-7))
```

Momentum

RMSProp

Bias Correction

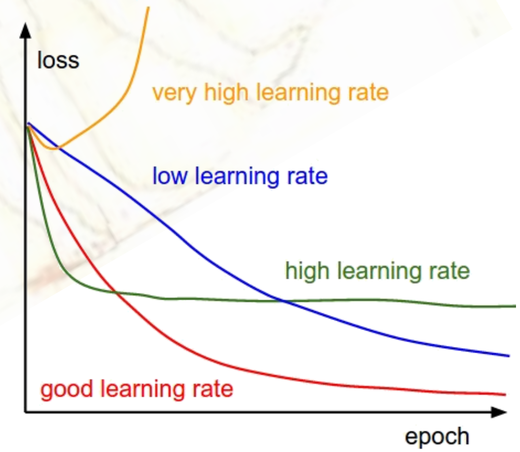
Adam 알고리즘



Adam 결과

Optimizer (6/6)

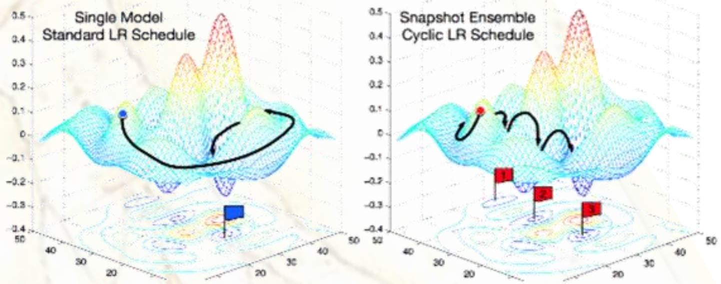
- **Second-Order Optimization**
  - 뉴턴의 방법**(Newton's method)**인 **2**차 테일러 근사**(quadratic approximation)**에 기반한 방법
  - **2**번 미분하여 이차곡선의 최솟값을 찾아서 이동함
    - 헤시안 행렬**(Hessian matrix)**을 이용하여 연산 진행
    - 행렬 연산 시 많은 메모리를 사용함
  - **Limited Memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS)**
    - 헤시안 행렬을 근사하여 연산에 필요한 메모리를 줄임
    - **Non-Convex** 문제를 해결하는데 적합하지 않음
    - **Full Batch**가 가능할 때 사용**(Stochastic Case**에 적합하지 않음**)**
- 학습률 조정**(Leanring rate decay)**
  - 모든 최적화 알고리즘은 학습률이 존재함
  - 학습을 진행하며 학습률 조정하여 학습을 개선하는 방법
    - **Exponential decay**
      - $\alpha = \alpha_0 e^{-kt}$
    - **1/t decay**
      - $\alpha = \frac{\alpha_0}{(1+kt)}$
        - α_0 : 기존 학습률, k : decay 상수, t : iteration number)



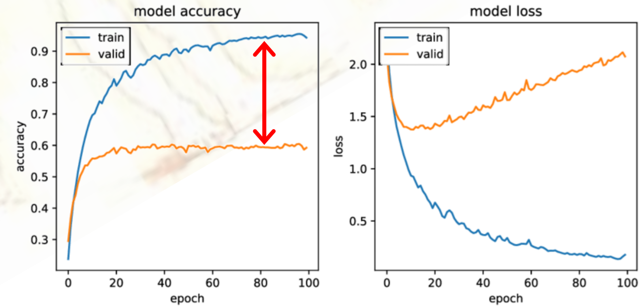
학습률

Beyond Training Error(1/2)

- 모델 앙상블(Model Ensembles)
  - 독립적으로 학습한 모델들의 결과값을 시험(**Test**) 단계에서 평균으로 이용
  - 모델의 개수와 종류가 증가할수록 효과가 증가함
  - 학습 도중 중간 모델들을 저장(**Snapshots**)하고 앙상블로 사용하여 학습률의 변동을 완화하기도 함
- 규제화(Regularization)
  - 손실 함수에 항 추가하기
    - 손실(**Loss**)을 줄이는데 기여하지 못하는 모수를 **0** 또는 **0**에 가까운 값으로 제한함
    - $J(\theta) + \boxed{\lambda R(W)}$
      - $J(\theta)$: 손실 함수
    - $R(W)$: 추가 항
      - **L2 Regularization (Lasso) :** $R(W) = \sum_k \sum_l W_{k,l}^2$
        - » 경사하강법에서 사용할 경우, 모든 가중치가 선형적으로 **Decay**하는 것을 의미
        - » **Neural Networks**와 어울리지 않음
      - **L1 Regularization (Ridge) :** $R(W) = \sum_k \sum_l |W_{k,l}|$
      - **Elastic Net(L1 + L2) :** $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + \ |W_{k,l}|$



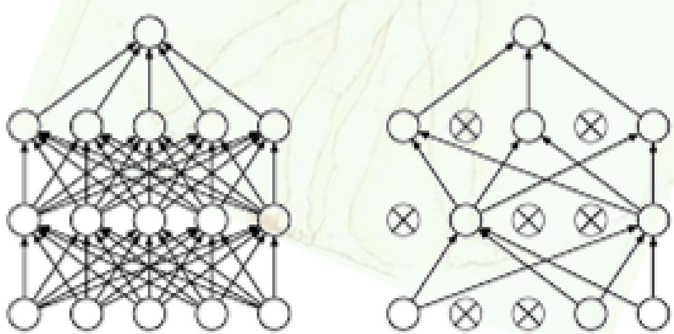
SGD와 앙상블 모델



모델의 정확도와 손실

Beyond Training Error(2/2)

- 드롭아웃(Dropout)
  - p의 확률로 임의의 뉴런의 **Activation**을 **0**으로 만듦
  - 드롭아웃의 장점
    - 변수간의 상호작용(**Co-adaptation**)하는 것을 막음으로써 과적합을 방지함
    - 랜덤으로 드롭아웃하여 단일 모델로 앙상블 효과를 가짐
- Data Augmentation
  - 학습 시 데이터를 무작위 변형시켜 이용함으로써 규제화 효과를 얻음
    - 좌우반전(**Horizontal Flips**), 자르기(**Crop**), 스케일링(**Scaling**), **Color Jittering** 등
- DropConnect
  - **Activation**이 아닌 가중치 행렬을 **0**으로 만듦
- Fractional Max Pooling
  - **pooling** 연산을 차례로 실행하는 것이 아닌, 임의의 지역에서 연산을 실행함



Fully Connected 네트워크와 드롭아웃

자르기의 예시

DropConnect