# Optimizers

August. 2018

LINK@KoreaTech
http://link.koreatech.ac.kr

# 매개변수 갱신

◈ 신경망 학습의 목적
- 손실 함수의 값을 가능한 낮추는 매개변수 찾기
- 즉 매개변수의 최적값 찾기
- 최적화 문제(Optimization Problem)
- 매개변수의 수가 많고, 매개변수 공간은 넓고 복잡함

◈ 최적의 매개변수를 찾는 방법
- 매개변수의 기울기(미분)이용
- 기울어진 방향으로 매개변수 값이 변하도록 반복적으로 갱신
- Stochastic Gradient Descent(SGD)

# Stochastic Gradient Descent

- "확률적 경사 하강법"
- 기울기(Gradient)를 이용하는 방법
  - $W \leftarrow W - \eta \frac{\delta L}{\delta W}$
    - $\eta$: Learning rate(학습률, step size). 0.01, 0.001
    - $W$: 갱신할 가중치 매개변수들
    - $L$: Loss Function (예측값과 실제 결과값의 차이를 정하는 함수)
  - Loss Function의 값을 최소화 하는 $W$ 값 찾기
  - 기울어진 방향으로 일정 step size만큼만 이동
  - `model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.sgd())`

(a) SGD without momentum (b) SGD with momentum

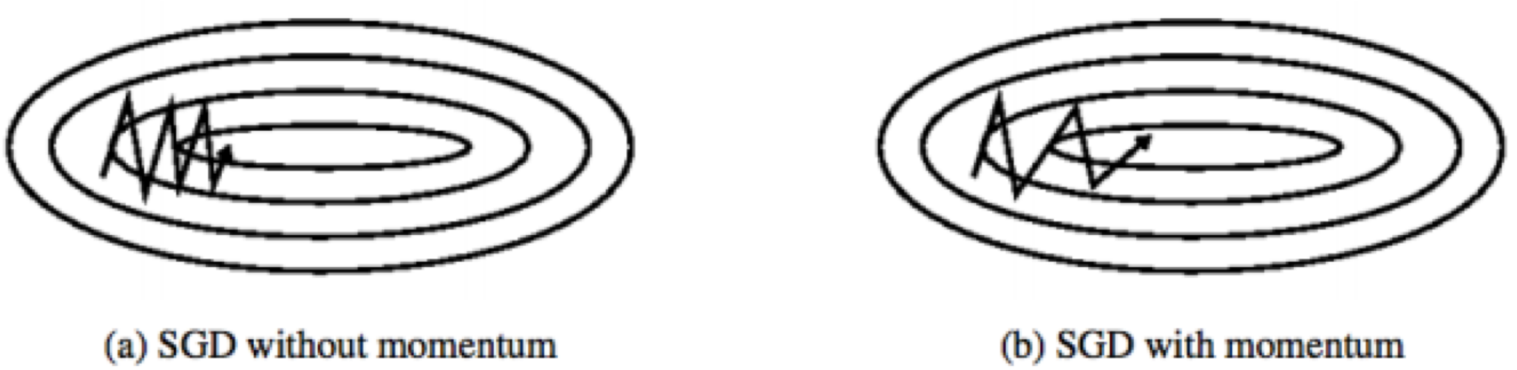
Figure 2: Source: Genevieve B. Orr

# Momentum

◈ Gradient Descent를 통해 이동하는 과정에 '관성'을 추가

- Momentum: 운동량
- 과거 이동했던 방식을 기억, 그 방향으로 일정 정도를 추가적으로 이동
- $v \leftarrow \alpha v - \eta \frac{\delta L}{\delta W}$

  $W \leftarrow W + v$
  - $\alpha$ : momentum term (지면 마찰, 공기 저항에 해당. 0.9 정도의 값을 사용)
  - $v$: 이동 벡터 (과거 얼마나 이동했는 지에 대한 이동 항)
- `optimizer=keras.optimizers.SGD(lr=0.01, momentum=0.9)`



(a) SGD without momentum

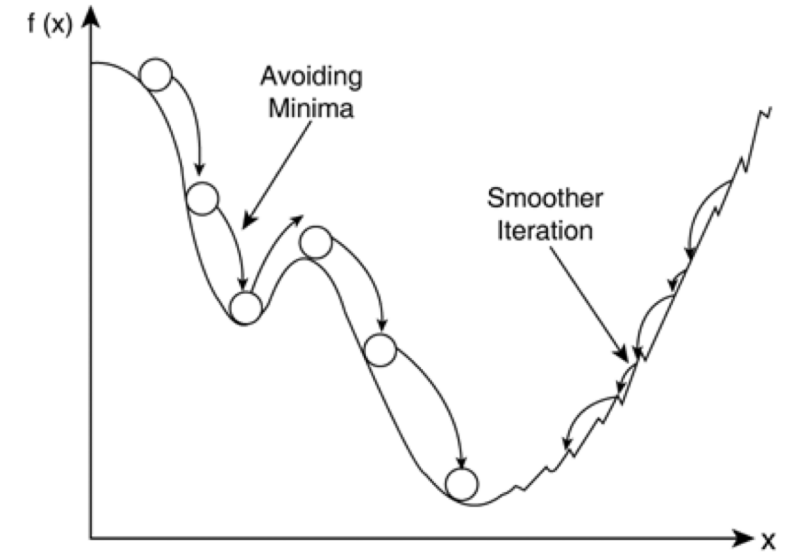
(b) SGD with momentum

Figure 2: Source: Genevieve B. Orr

# Momentum Optimizer

local minima를 빠져나오는 효과

- Local Minima에 빠지면 Gradient가 0이 되어 이동 불가




Avoiding Local Minima. Picture from http://www.yaldex.com.

# Adagrad

◈ 네트워크 변수들을 update할 때마다 step size를 다르게 설정

- "처음에는 크게 학습하다가 조금씩 작게 학습한다"
- "지금까지 많이 변화하지 않은 변수들은 step size를 크게 하고,
  지금까지 많이 변화했던 변수들은 step size를 작게 하자"
  - 많이 변화한 변수들은 optimum에 가까이 있을 확률이 높다.
  - 적게 변화한 변수들은 optimum에 더 가까이 다가가도록 보폭을 늘린다.

$$h \leftarrow h + \frac{\delta L}{\delta W} \odot \frac{\delta L}{\delta W}$$

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

  - $h$ 값으로 학습률 값을 조정
  - `Optimizer = keras.optimizers.Adagrad(lr=0.01, epsilon=1e-7)`
- 단점:
  - AdaGrad는 과거의 기울기를 제곱하여 계속 더함
  - 그래서 학습을 진행할수록 갱신 강도가 약해짐
  - 학습을 계속할수록 매개변수 갱신량이 0이 되어 갱신이 일어나지 않게 됨

# Adam

◈ AdaGrad와 Momentum 방식을 합친 Optimizer

- 2015년에 제안된 새로운 방법
- 직관적으로는 모멘텀과 AdaGrad를 융합한 방법
- 세 개의 하이퍼파라미터를 조정하게 되어있음
  - $\alpha$ : learning rate
  - $\beta_1$ : 일차 모멘텀용 계수
  - $\beta_2$ : 이차 모멘텀용 계수

- `Optimizer = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999)`

# 그 외 optimizer

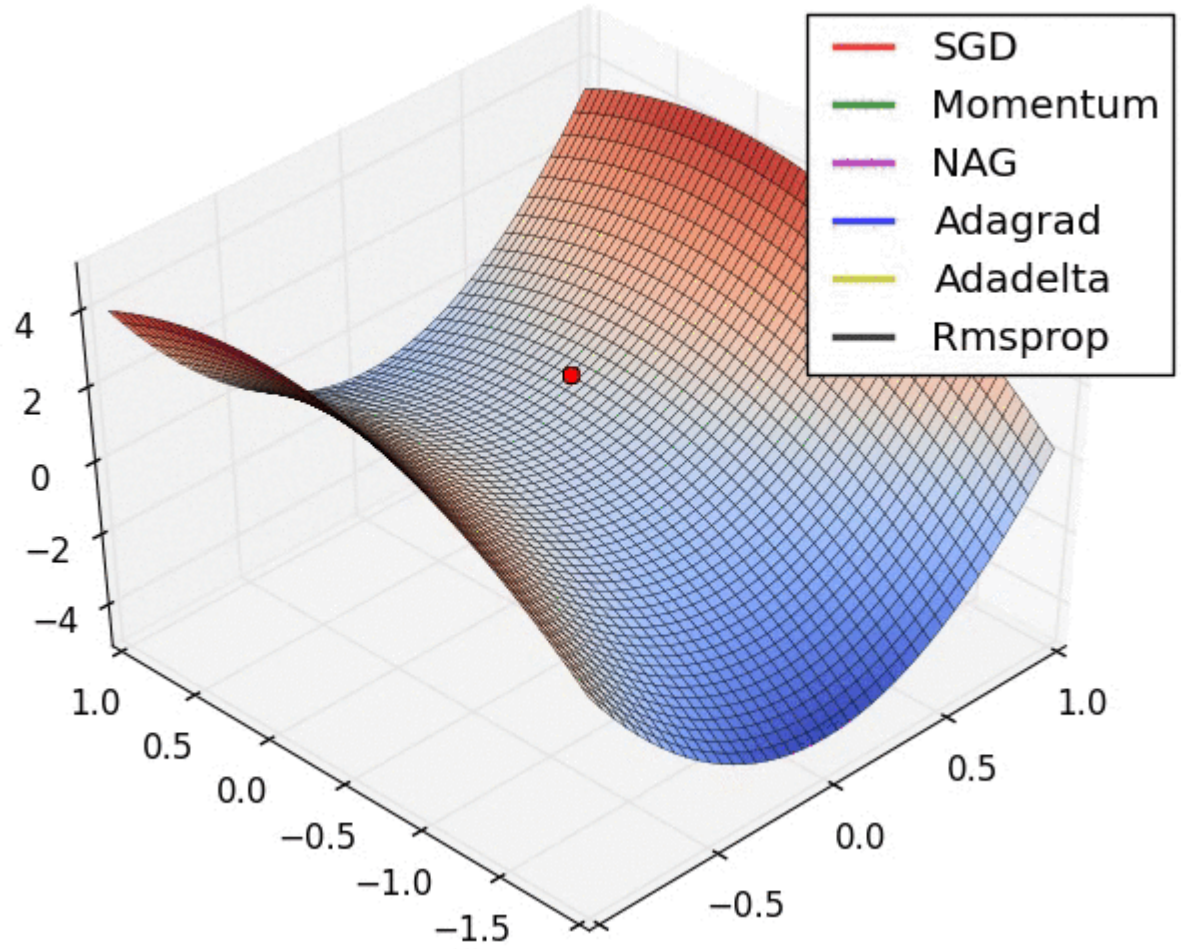
◈ SGD, Momentum, AdaGrad, Adam 이 외에
NAG, AdaDelta, RMSProp 등의 방법들도 있음

◈ 어느 Optimizer를 이용할 것인가?
- 풀어야 할 문제가 무엇이냐에 따라 달라짐
- 하이퍼파라미터를 어떻게 설정하느냐에 따라 결과가 달라짐
- 모든 문제에서 항상 뛰어난 기법은 없음
- SGD, Adam이 보편적으로 쓰임

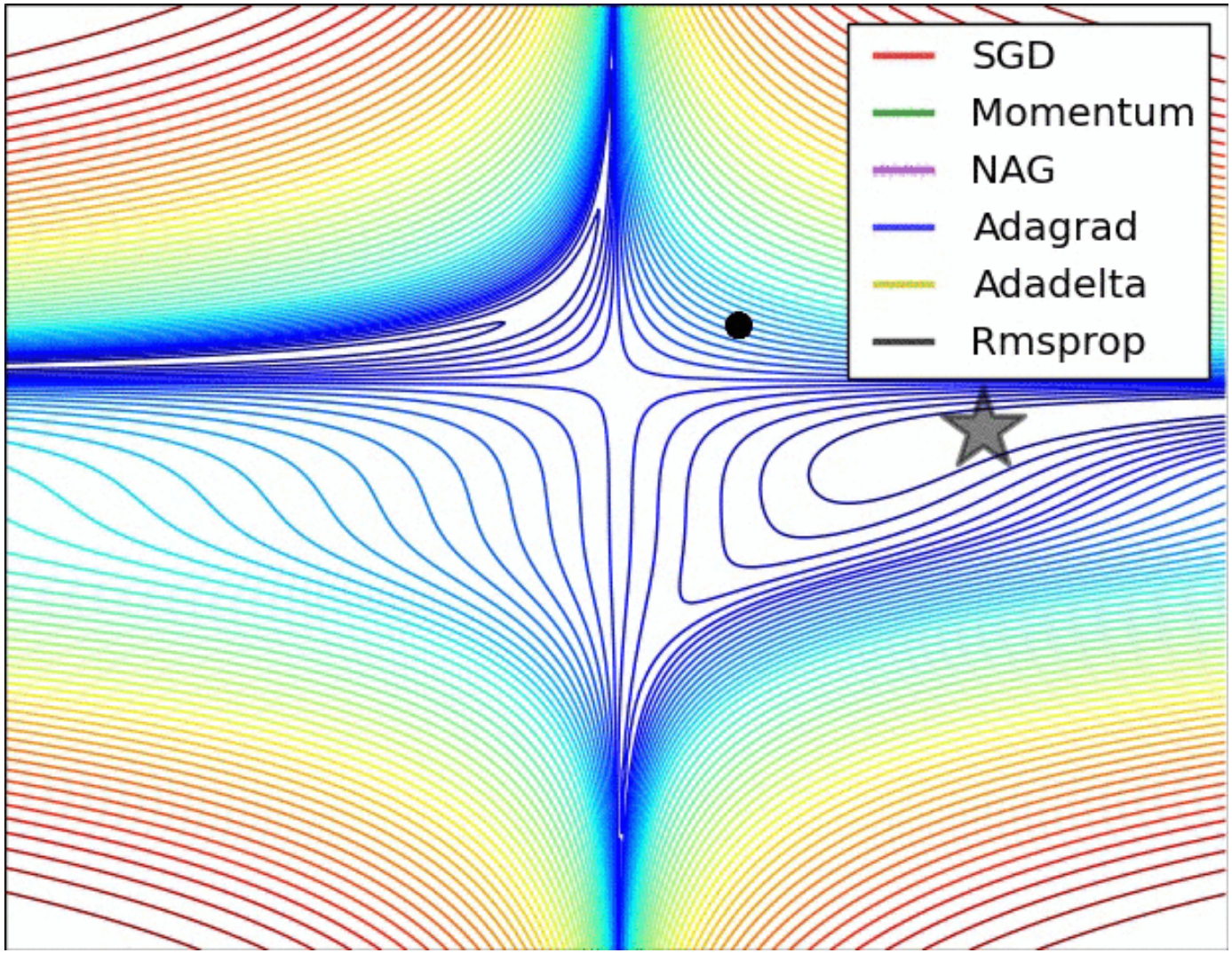
# 그 외 optimizer

◈ SGD, Momentum, AdaGrad, Adam 이 외에
NAG, AdaDelta, RMSProp 등의 방법들도 있음

# Optimizer 비교



# Optimizer 비교



# 가중치의 초기값

◈ 매개변수의 초기값을 어떻게 주느냐가 신경망 학습의 성패를 가름
◈ 가중치를 균일한 값으로 주는 것이 아이디어
◈ 표준편차 $\sigma$
  – 0 또는 0.01 → 사용하지 않기를 권장
  – Xavier 초기값 → S자 모양 곡선을 보이는 활성화 함수에 사용
    • tanh
    • sigmoid
  – He 초기값 → ReLU 함수에 사용

```
model.add(
        SimpleRNN(
                ….
                kernel_initializer=‘glorot_uniform’ or ‘he_init’
                ….
        )
)
```

# Thank you