# Training Neural Networks

Lecture 6 recap: Activation function 으로
ReLU "max(0, x)" 가 good default choice

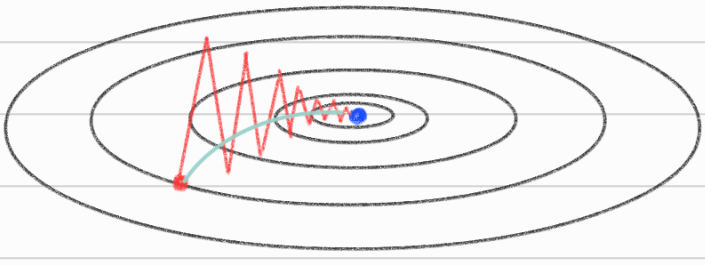
Weight initialization 때,
초기화 값이 너무 작으면 to zero (Vanish)
" 너무 크면 saturate, no learning
" 적당하면 nice distribution of activation 달성

Batch normalization {
input: $x: N \times D$
learnable params: $\gamma, \beta: D$
intermediates: $\mu, \sigma: D$, $\hat{x}: N \times D$
output: $y: N \times D$

Hyperparameter Search: Random > Grid

## Problems with SGD

① If loss changes quickly in one direction and slowly in another?



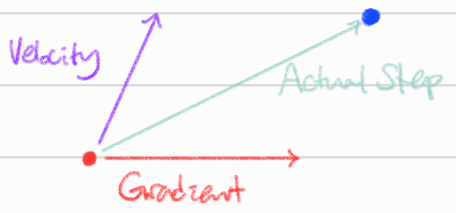
Very slow progress along shallow dimension, jitter along steep direction
dimension 클수록 더 문제

② Local minima or saddle point 

Saddle points much more common in high dimension
고차원에서 모든 dimension이 한 방향이기는 어려움

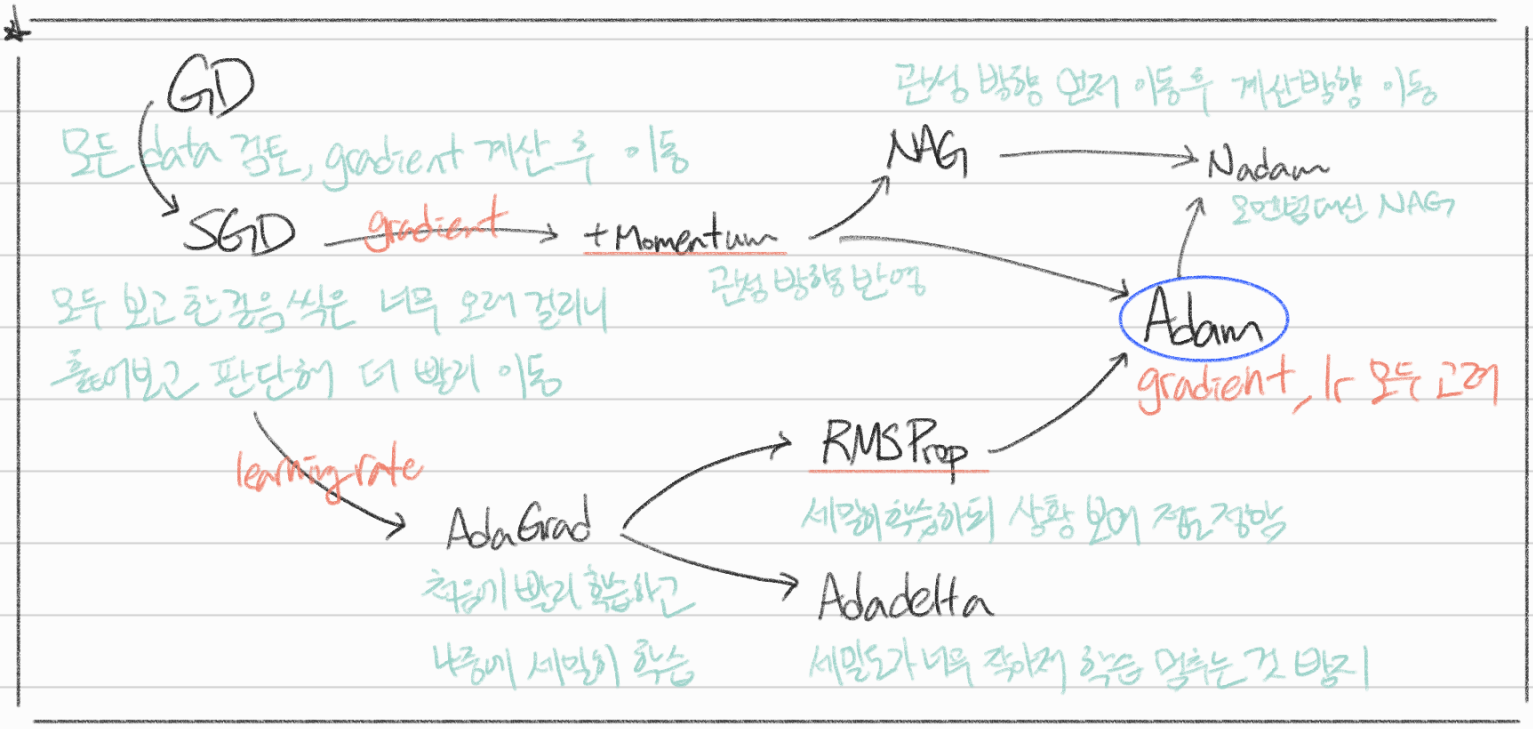
③ Gradients can be noisy since they are from minibatches

Momentum 으로 해결 SGD + Momentum (velocity vector 활용)




* Nestrov Momentum도 있음

AdaGrad → RMSProp




## Learning Rate Decay

- 처음에 학습률을 크게 설정하고 일정 epoch 마다 값을 감소시켜 최적값까지 더 빠르게 도달

Model Ensembles 앙상블 기법 통해 performance 향상 가능

BUT one model performance 향상하려면 "Regularization"

Dropout (for NN) 으로 ensemble 효과

<In Practice>

1. Drop in forward pass (Add some kind of randomness)
2. Scale at test time (Average out randomness)

## Data Augmentation

- Horizontal flips
- Random crops and scales
- Color jitter

Batch Normalization 만으로 충분할수도!

## Transfer Learning with CNNs

1. Train on Imagenet
2. Small Dataset
3. Bigger Dataset

Transfer learning with CNNs is pervasive