# Lab 6: Optimizers & Visualization tools

**<삼성 AI전문가 교육과정> 실습**

**서울대학교 바이오지능 연구실 (장병탁 교수)**
**최원석, 김윤성**
**2022.06.09**

Biointelligence Laboratory
Dept. of Computer Science and Engineering
Seoul National University

# Contents

- Optimizers
  - Optimizing methods
  - Weight initialization
- Visualization tools
  - tensorboard

SGD, AdaGrad, Adam, Xavier

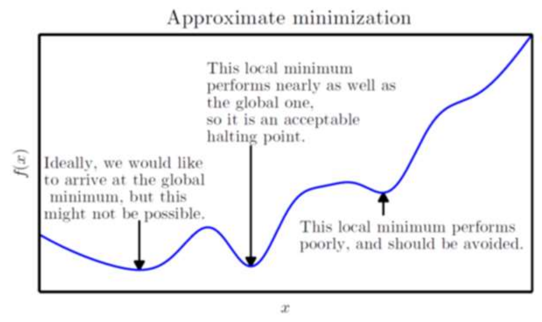
# Optimizers

# Optimization

- Optimization
  - 주어진 입력 X에 대해 함수 J(cost, loss, etc.)의 값을 최소로 만드는 weight들의 집합(w)를 찾는 과정

$$w_{opt} = argmin_w J(X; w)$$

  - 일반적으로 J로 형성된 함수는 w에 대해 매우 복잡하므로 수식 전개로 global optima(가장 최소의 값)을 찾는 것은 거의 불가능
  - Hill climbing algorithm의 방식으로 local optima을 찾음



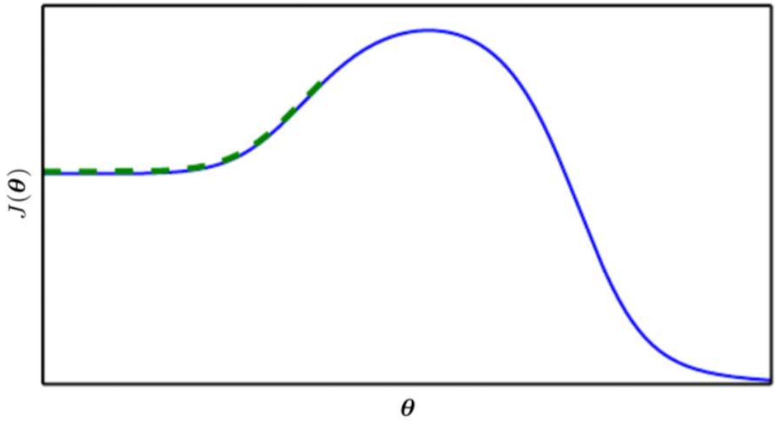
# Optimization 방식

- Optimization methods
  - Batch 방식 : 모든 데이터를 넣고 계산하여 한번 업데이트
  - Stochastic(online) 방식 : 하나의 데이터를 sampling하고, 한번 업데이트
  - Mini-batch 방식 : 전체에서 적절한 개수의 데이터를 sampling하고 한번 업데이트

  - 왜 stochastic하여야 하는가? : 데이터 편향 최소화

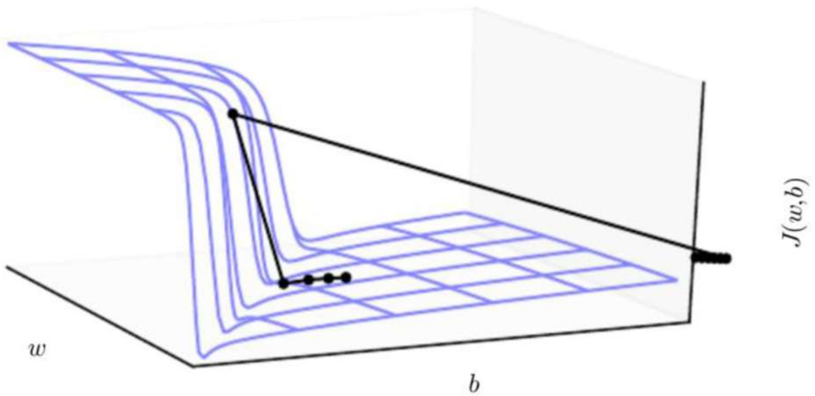
# Optimization 과정에서의 문제들

- 언덕 문제
  - 더 낮은 optima로 이동하여야 하는데 낮은 언덕으로 인해 업데이트 하지 못하는 경우



# Optimization 과정에서의 문제들

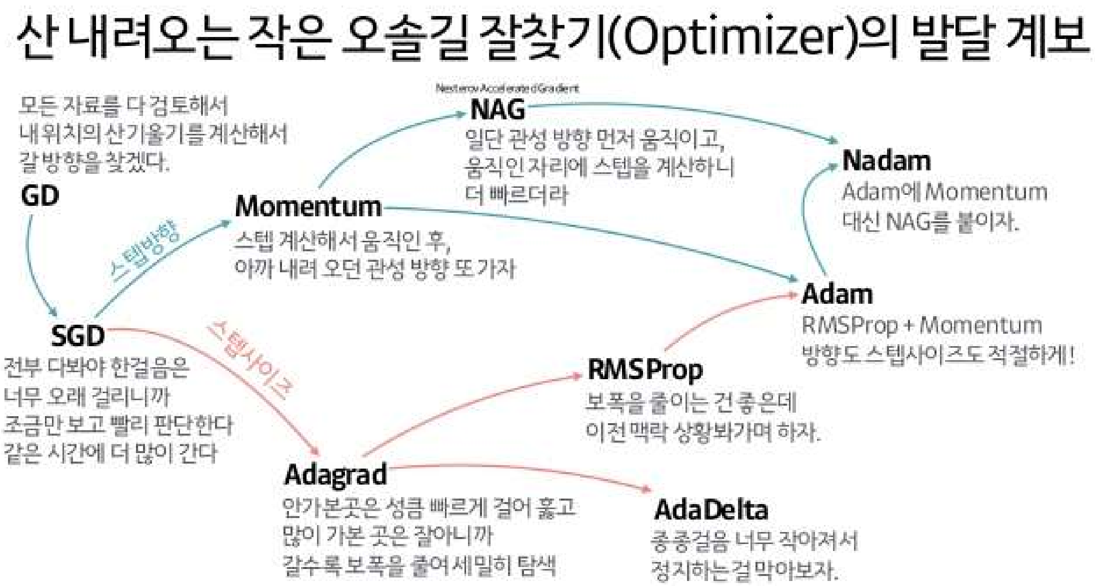
- 절벽으로 인한 gradient exploding 문제
  - 너무 급한 경사가 존재하는 경우, gradient 값이 너무 커져 지나치게 많이 이동하는 경우
  - Cf) gradient vanishment



Gradient Exploding

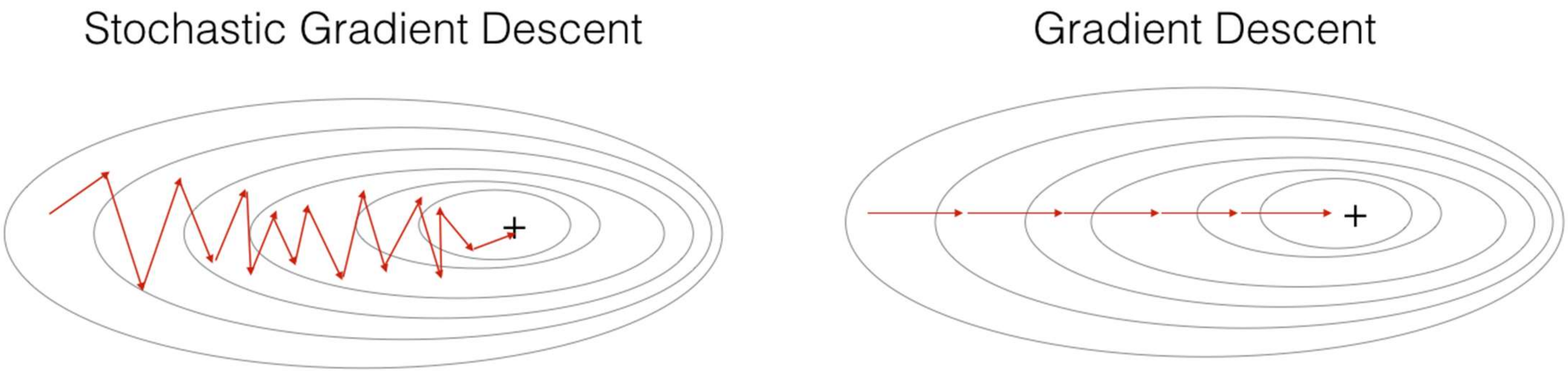
# Optimizer의 종류들

- GD(Gradient Descent) : 이론 시간에 다룸
- SGD(Stochastic Gradient Descent)
- Ada- (Adaptive-)
  - Adagrad
  - AdaDelta
  - Adam
- RMSProp



# SGD

- Stochastic Gradient Descent
  - Mini-batch Learning
  - Gradient descent와 과정은 동일하나 sampling으로 인해 더 좋은 성능
  - 화살표 하나 : 한번의 학습, SGD의 경우 1epoch이 수 회의 batch (GD는 1 batch)



# SGD

- Stochastic Gradient Descent
  - Mini-batch Learning
  - Gradient descent와 과정은 동일하나 sampling으로 인해 더 좋은 성능
  - 화살표 하나 : 한번의 학습, SGD의 경우 1epoch이 수 회의 batch (GD는 1 batch)

---

**Algorithm 8.1** Stochastic gradient descent (SGD) update at training iteration $k$

---

**Require:** Learning rate $\epsilon_k$.
**Require:** Initial parameter $\boldsymbol{\theta}$
  **while** stopping criterion not met **do**
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\underline{\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\hat{\boldsymbol{g}}}$
  **end while**

---

# AdaGrad

- Adaptive Gradient~
  - Rare한 정보(변수)에 대해서 더 많은 가중치를, common한 정보에 대해서 더 적은 가중치를 할당하여 gradient에 적용
  - 데이터에 대해서 Adaptive

**Algorithm 8.4** The AdaGrad algorithm

**Require:** Global learning rate $\epsilon$
**Require:** Initial parameter $\boldsymbol{\theta}$
**Require:** Small constant $\delta$, perhaps $10^{-7}$, for numerical stability
  Initialize gradient accumulation variable $\boldsymbol{r} = \boldsymbol{0}$
  **while** stopping criterion not met **do**
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$
    Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$.   (Division and square root applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  **end while**

# AdaDelta

- Ada+Delta
  - 다른 모델들이 first-order optimization(gradient)를 활용할 때, AdaDelta는 second-order까지 확인하여 optimization 진행

$$\Delta x \propto H^{-1} g \propto \frac{\frac{\partial f}{\partial x}}{\frac{\partial^2 f}{\partial x^2}} \propto \text{units of } x \qquad \Delta x = \frac{\frac{\partial f}{\partial x}}{\frac{\partial^2 f}{\partial x^2}} \Rightarrow \frac{1}{\frac{\partial^2 f}{\partial x^2}} = \frac{\Delta x}{\frac{\partial f}{\partial x}}$$

**Algorithm 1** Computing ADADELTA update at time $t$

**Require:** Decay rate $\rho$, Constant $\epsilon$
**Require:** Initial parameter $x_1$
1: Initialize accumulation variables $E[g^2]_0 = 0$, $E[\Delta x^2]_0 = 0$
2: **for** $t = 1 : T$ **do** %% Loop over # of updates
3: Compute Gradient: $g_t$
4: Accumulate Gradient: $\underline{E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2}$
5: Compute Update: $\Delta x_t = -\frac{\text{RMS}[\Delta x]_{t-1}}{\text{RMS}[g]_t} g_t$
6: Accumulate Updates: $\underline{E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho) \Delta x_t^2}$
7: Apply Update: $x_{t+1} = x_t + \Delta x_t$
8: **end for**

# RMSProp

- AdaGrad + exponential moving avg.
  - AdaGrad의 $r$가 무한히 커지는 것을 방지
  - Exponential moving average를 활용하여 convex한 부분에 대해 더 빠르게 학습을 진행
    - 초반 학습의 정도가 빠르다.

**Algorithm 8.5** The RMSProp algorithm

**Require:** Global learning rate $\epsilon$, decay rate $\rho$.
**Require:** Initial parameter $\boldsymbol{\theta}$
**Require:** Small constant $\delta$, usually $10^{-6}$, used to stabilize division by small numbers.
  Initialize accumulation variables $\boldsymbol{r} = 0$
  **while** stopping criterion not met **do**
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho) \boldsymbol{g} \odot \boldsymbol{g}$
    Compute parameter update: $\Delta \boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$. ($\frac{1}{\sqrt{\delta + \boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$
  **end while**

# Adam

- RMSProp + momentum
  - Momentum : 학습 방향의 관성(이전의 gradient가 반영)

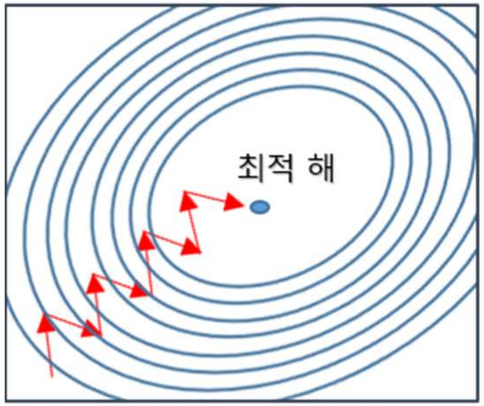
---

**Algorithm 8.7** The Adam algorithm

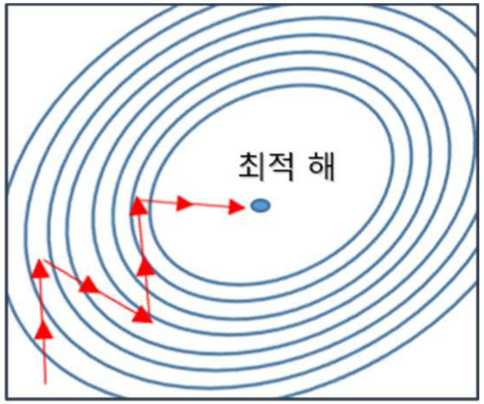
---

**Require:** Step size $\epsilon$ (Suggested default: 0.001)
**Require:** Exponential decay rates for moment estimates, $\rho_1$ and $\rho_2$ in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
**Require:** Small constant $\delta$ used for numerical stabilization. (Suggested default: $10^{-8}$)
**Require:** Initial parameters $\boldsymbol{\theta}$
  Initialize 1st and 2nd moment variables $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$
  Initialize time step $t = 0$
  **while** stopping criterion not met **do**
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    $t \leftarrow t + 1$
    Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1) \boldsymbol{g}$
    Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2) \boldsymbol{g} \odot \boldsymbol{g}$
    Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$
    Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$
    Compute update: $\Delta \boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ (operations applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$
  **end while**

---



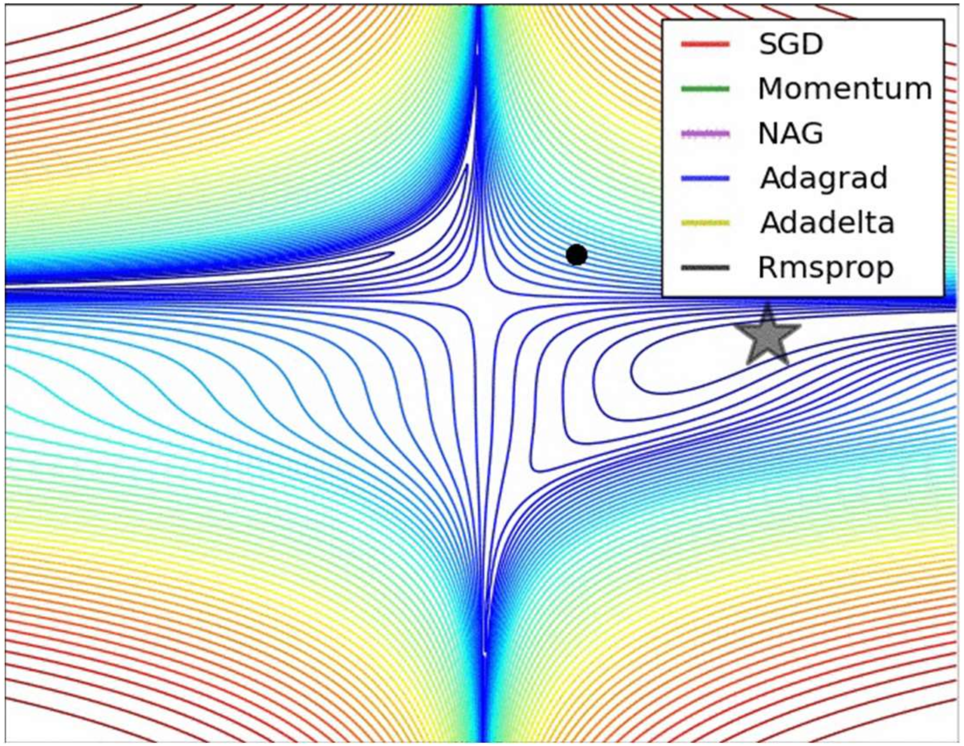

확률적 경사 하강법




모멘텀

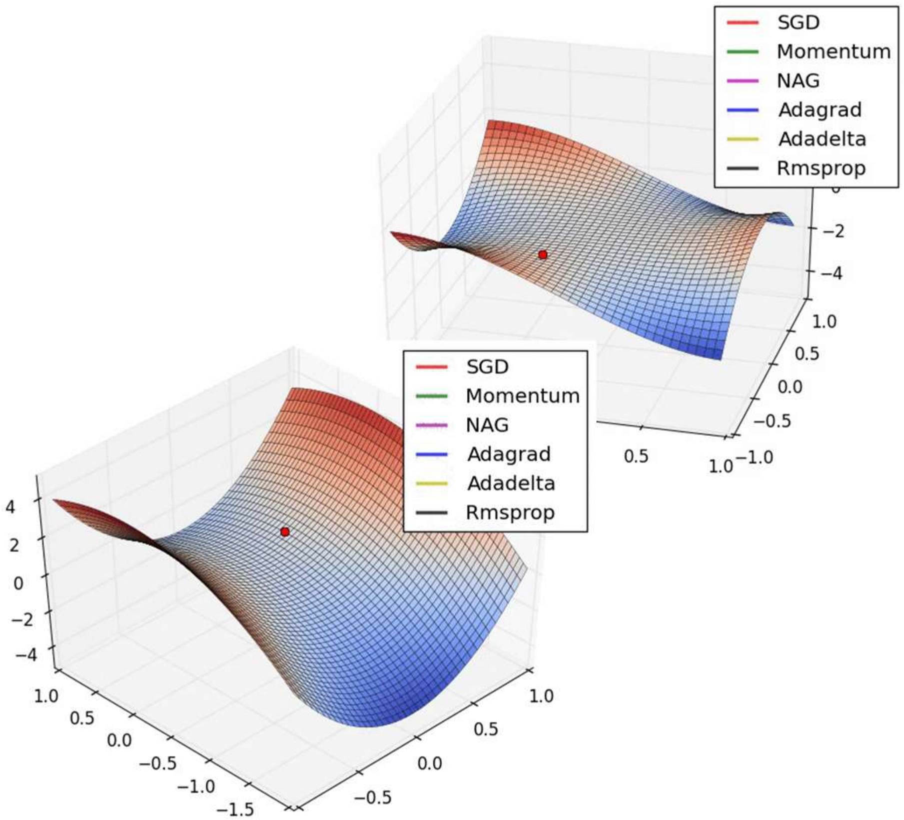
# Optimizer 문제별 성능 비교

- Convex한 경우
  - RMSprop 및 Adaptive한 optimizer 들이 높은 성능
  - Momentum만을 활용할 경우 학습이 잘 진행되지 않음



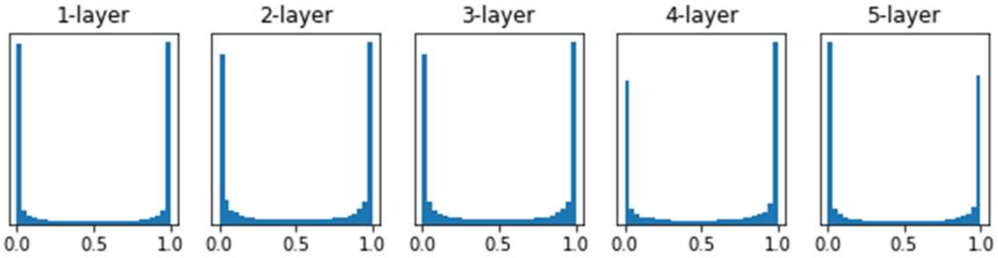
# Optimizer 문제별 성능 비교

- Non-Convex한 경우
  - Saddle-point problem
    - 생성된 지형이 안장점의 형태일때
    - E.g. GAN 등
  - SGD 및 momentum으로 학습시 매우 비효율적
  - RMSProp이나 Adam이 좋은 성능

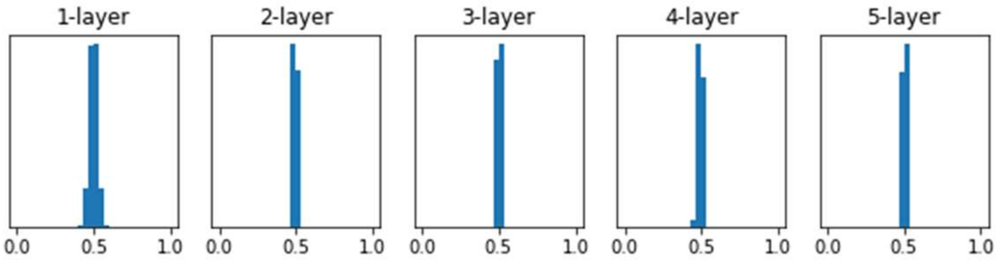


# Weight initialization

- Naïve initialization
  - 모두 0으로 초기화
    - 모든 가중치의 값이 똑같이 갱신됨
  - 정규분포로 랜덤하게 초기화
    - Activation - sigmoid
    - 표준편차가 1일 때



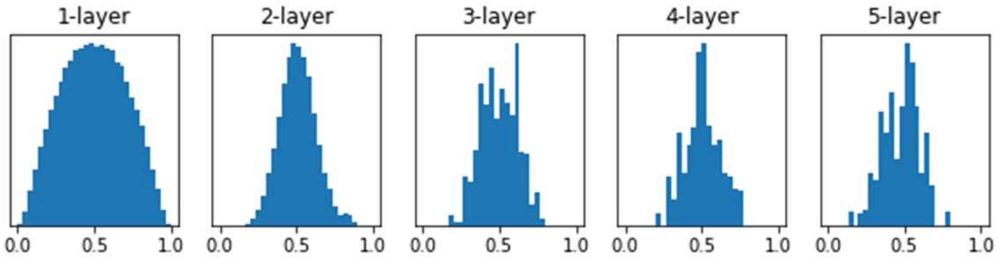

    - 표준편차가 0.01일 때



# Weight initialization

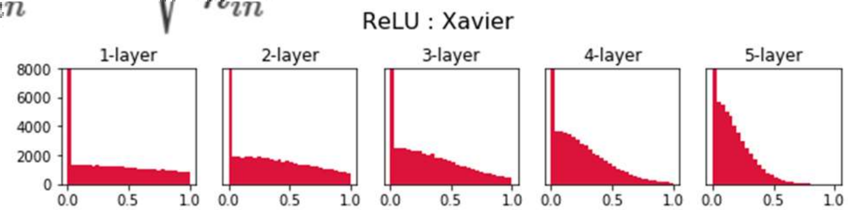
- Xavier Initialization (Xavier Glorot & Yoshua Bengio)
  - activation이 sigmoid일 때, 실험적으로 증명된 weight initialization 방식
    - m : input dim, n : output dim, U : uniform distribution

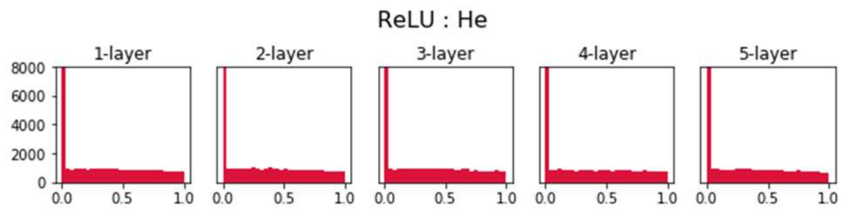
$$W_{i,j} \sim U\left(-\frac{6}{\sqrt{m+n}}, \frac{6}{\sqrt{m+n}}\right)$$




- He(Kaiming) Initialization (Kaiming He)
  - activation이 ReLu일 때, 실험적으로 증명된 weight initialization 방식

$$W \sim U\left(-\sqrt{\frac{6}{n_{in}}},\ +\sqrt{\frac{6}{n_{in}}}\right)$$

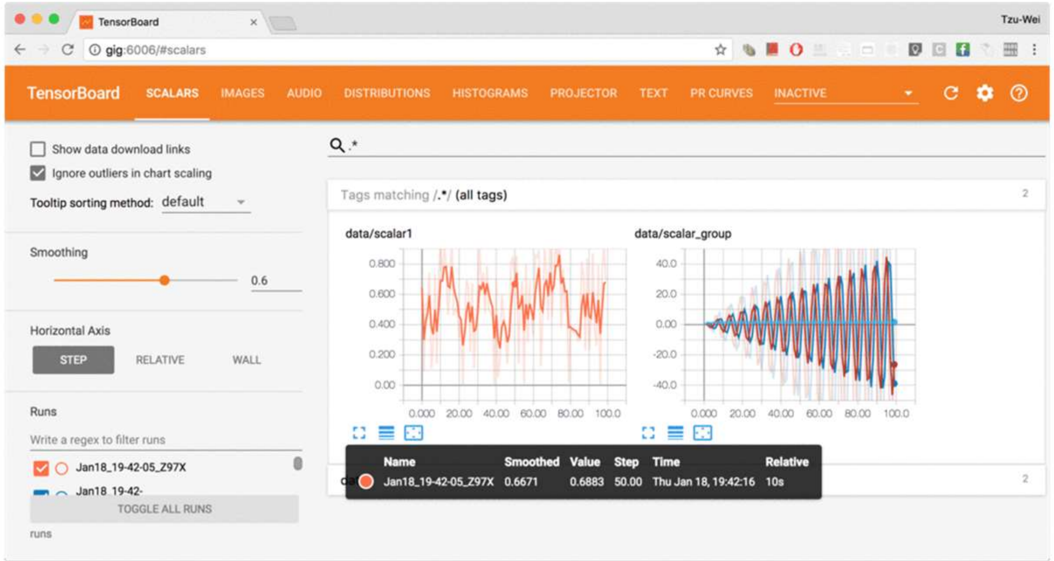


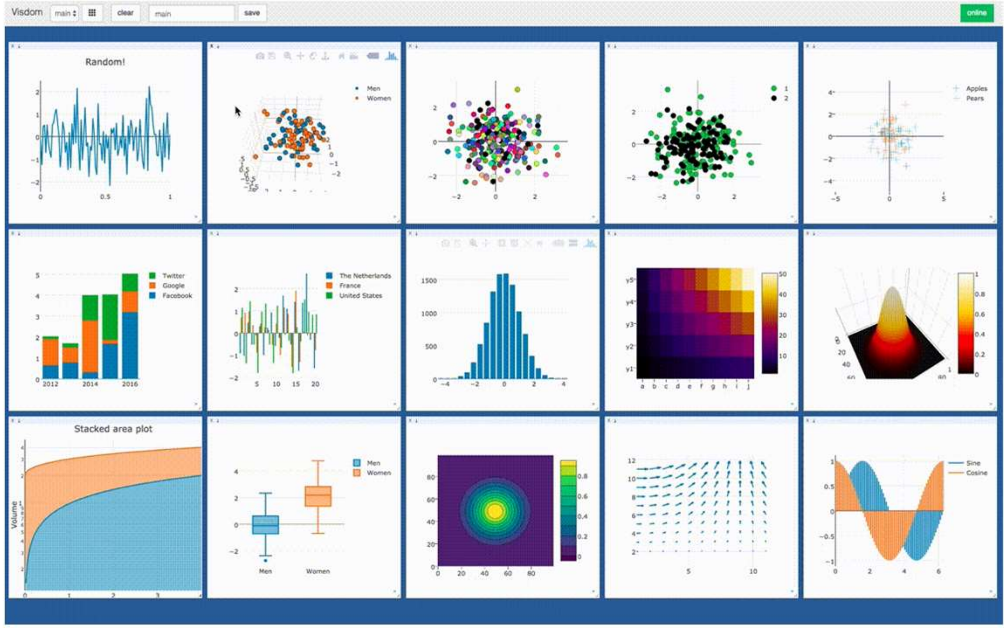
tensorboard, visdom

# Visualization tools

# Visualization tools




tensorboard




visdom

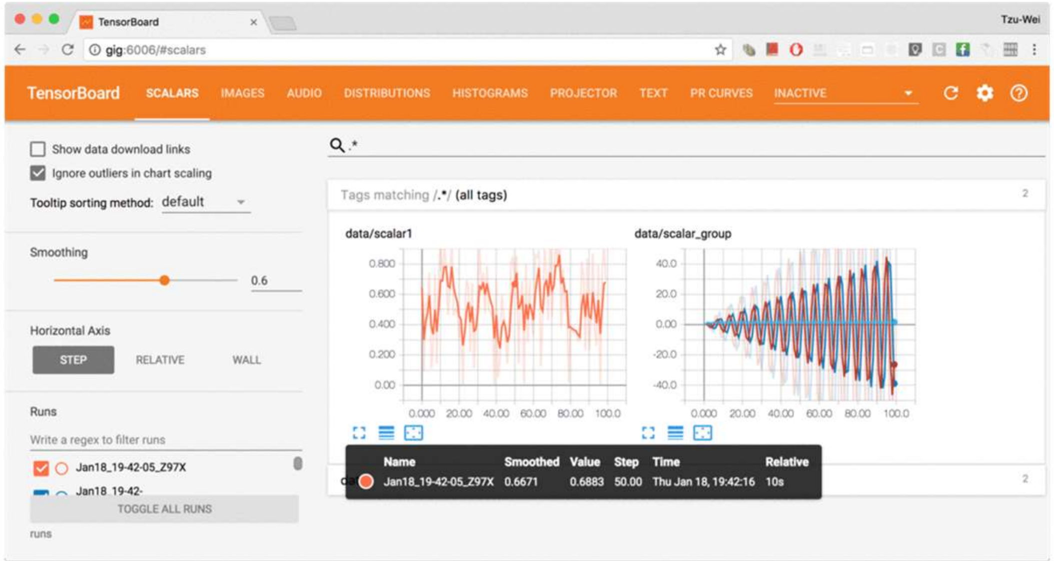
- The computations you'll use TensorFlow can be complex and confusing.
- To make it easier to understand, debug, and optimize TensorFlow programs, we've included a suite of visualization tools called TensorBoard.
- You can use TensorBoard to visualize your TensorFlow graph, plot quantitative metrics about the execution of your graph, and show additional data like images that pass through it.

# Visualization tools

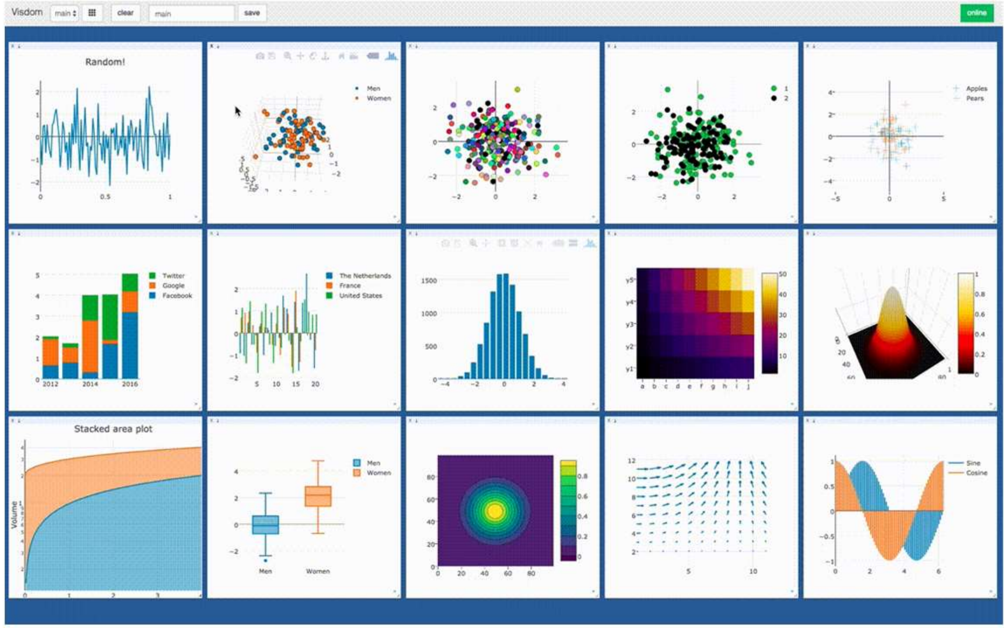
- Tensorboard
  - TensorFlow 개발을 위해 개발된 visualization tool
  - Pytorch에서는 tensorboardX라는 패키지를 통해 간접적으로 지원하다가
  - 1.1버전부터 공식적인 지원 (natively supported)
    - *pip install tensorboard*

- Visdom
  - (pytorch를 개발한) Facebook 개발팀에서 개발한 visualization tool
    - *pip install visdom*

# Visualization tools




tensorboard




visdom

- Tensorboard vs Visdom
  - Tensorboard 공식 지원 이후 대부분 tensorboard 사용을 선호하는 추세

# TensorBoard

- TensorBoard
  - 시각화 할 특정 event를 지정
  - 해당 event의 log를 SummaryWriter가 logfile(summary)에 저장
  - 해당 summary 을 읽어서 웹페이지 형태로 게시
    - In terminal - *tensorboard –log_dir=/path/to/root_log_dir*

- SummaryWriter
  - Logging할 event를 저장하는 class
  - 초기화
    - *from torch.utils.tensorboard import SummaryWriter*
    - *writer = SummaryWriter(log_dir)*

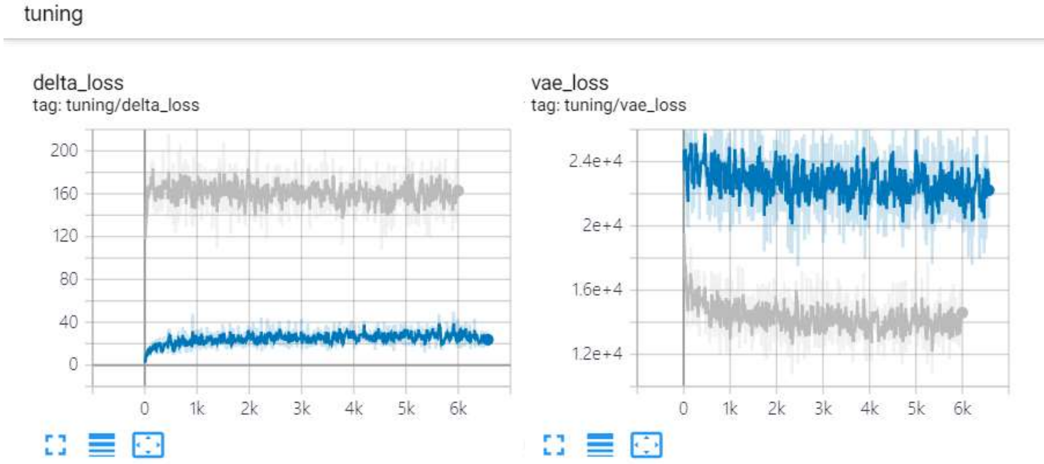
# TensorBoard

- SummaryWriter
  - Adding events
    - *writer.add_scalar(tag, scalar_value, global_step=None, walltime=None)*
      - step(혹은 시간) 에 따른 어떤 scalar값의 변화량을 표시
    - *writer.add_image(tag, img_tensor, global_step=None, walltime=None, dataformats='CHW')*
      - step(혹은 시간) 에 따른 이미지 데이터를 표시
    - *writer.add_scalars*
    - *writer.add_images*
    - *writer.add_figure*
    - *writer.add_histograms*

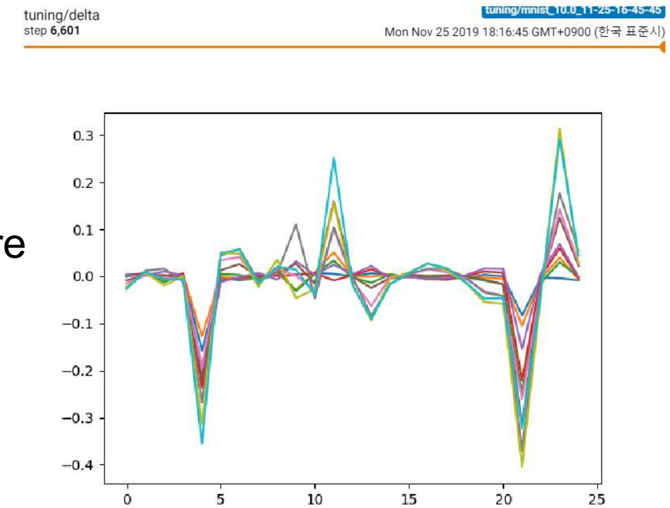
- https://pytorch.org/docs/stable/tensorboard.html?highlight=tensorboard

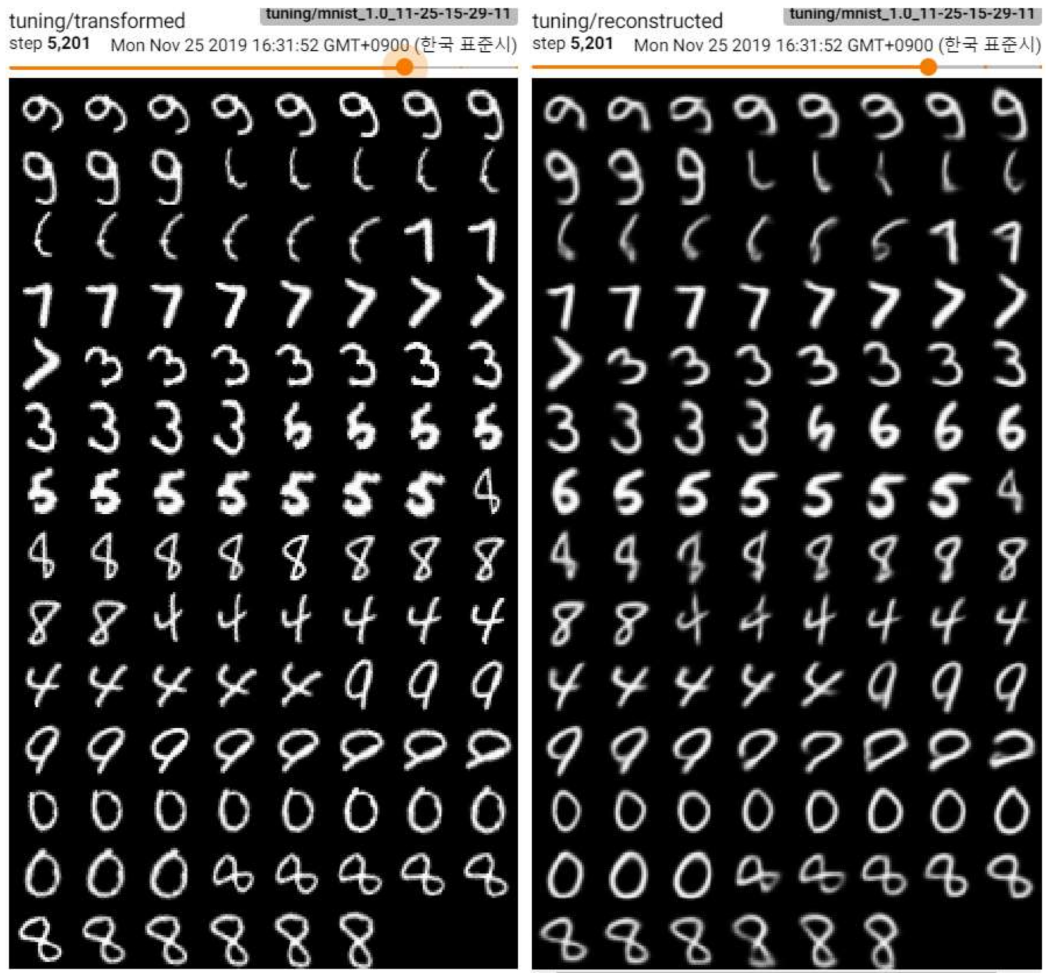
# TensorBoard

add_scalar




add_figure

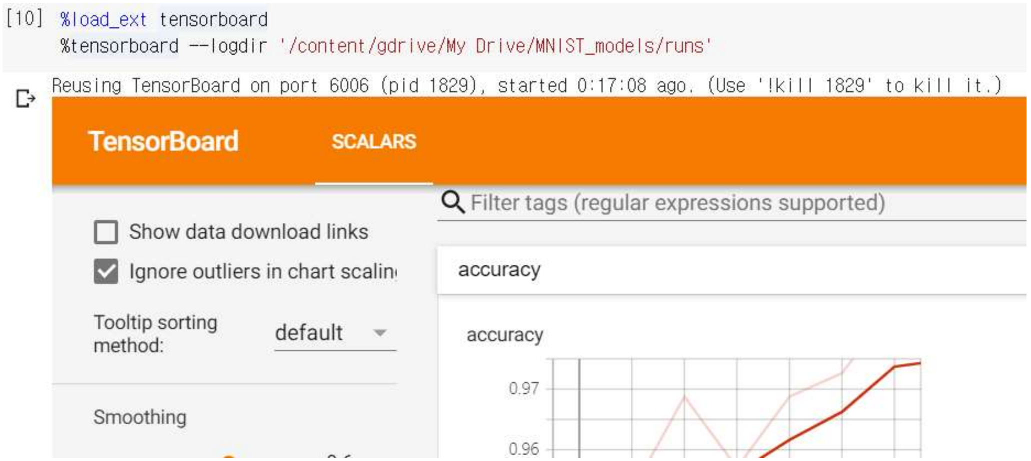



add_images

# Tensorboard on Colab

- 터미널에서 tensorboard 명령어 실행이 불가능




- Magic words
  - Colab cell 내부에 tensorboard 창 생성
- Using tensorboardcolab
  - *tb = TensorBoardColab()*
  - Tensorboard 링크를 자동 생성해서 제공
  - Colab 전용 라이브러리를 써야 하므로 추천하지 않음