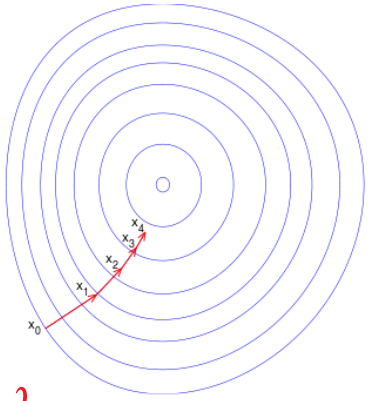
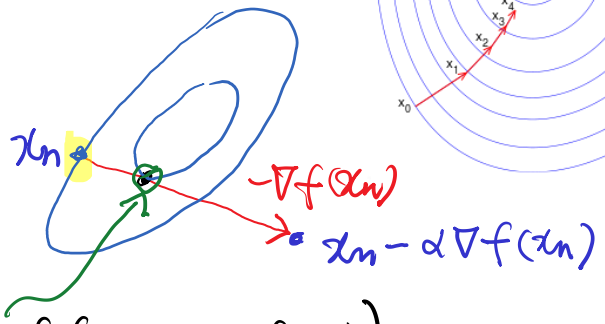
Optimization for ML

KAIST ISE
Hayong Shin

Gradient descent (GD)

- Local optimization problem
  - find local minimum point of $f(x)$ starting from x_0
  - GD : $x_{t+1} \leftarrow x_t - \alpha_t \nabla f(x_t)$
    - $\nabla f(x_t)$: steepest descent direction
      - TAQ. Is $\nabla f(x_t)$ the best direction?
  - α_t : step size
    - How to set α_t? → non-trivial





① Exact line search $\alpha_n = \underset{\alpha}{\operatorname{argmin}} f(x_n - \alpha \nabla f(x_n))$

② Backtracking line search (Armijo rule)

$\alpha = s$

while $f(x_n - \alpha \nabla f(x_n)) \geq f(x_n)$, $\alpha \leftarrow \alpha/2$

i.e. find the first $\alpha \in [s, \frac{s}{2}, \frac{s}{4}, \cdots]$ s.t. $f(x_n - \alpha \nabla f(x_n)) < f(x_n)$

③ 2nd order methods

Use $\nabla^2 f$ (Hessian) to determine α

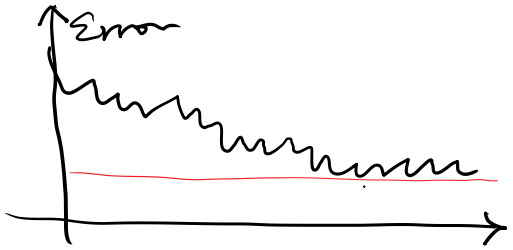
- $\nabla^2 f$ requires too much efforts to estimate
- approximate $\nabla^2 f$: BFGS, L-BFGS

Gradient descent : step size

④ constant $\alpha_n = \alpha$

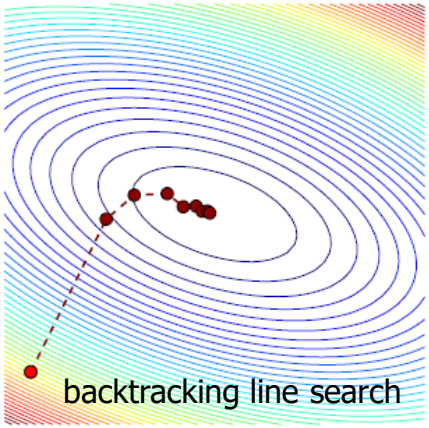
- bigger α : may diverge
- smaller α : slow

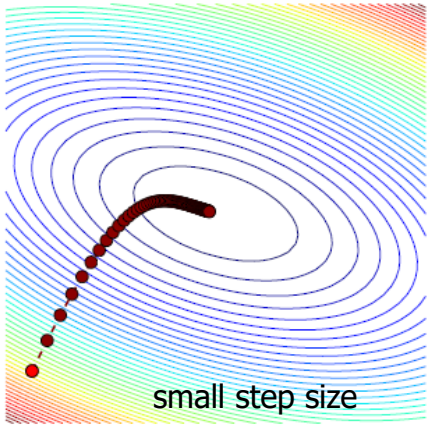
- may not converge in SGD

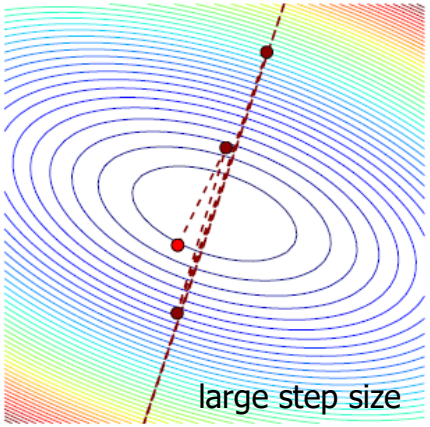


⑤ diminishing $\alpha_n \to 0$

- Essential for SGD
- Robbins – Monro condition $\sum_n \alpha_n = \infty$, $\sum_n \alpha_n^2 < \infty$







Stochastic Gradient

Local optimization problem : find x s.t. $\nabla f(x) = 0$

Let $g(x) = \nabla f(x) + \varepsilon$ $\quad E[\varepsilon] = 0$

If we are to find (local) minimum point of $f(x)$,

we can assume $f(x)$ is (locally) convex, $\Rightarrow \nabla f(x)$: increasing

$\Rightarrow$ $x_{n+1} = x_n - \alpha_n g(x_n)$,

where $g(x_n)$ is an unbiased estimator of $\nabla f(x_n)$ → stochastic gradient

$\sum_{n=1}^{\infty} \alpha_n \to \infty$ ① and $\sum_{n=1}^{\infty} \alpha_n^2 < \infty$ ②

* $f(x) = \sum_{i=1} f_i(x)$ · $f_i(x)$: loss function for i^{th} data, $i \in D = \{1, \dots, N\}$

$\nabla f(x) = \sum_{i \in D} \nabla f_i(x)$: (full) batch gradient

Let $M \subset D$ be a randomly chosen subset of size m ($|M| = m$)

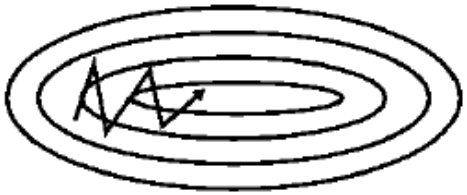
$g(x) = \sum_{i \in M} \nabla f_i(x)$ $\Rightarrow$ $E_M[g(x)] = \nabla f(x)$: unbiased

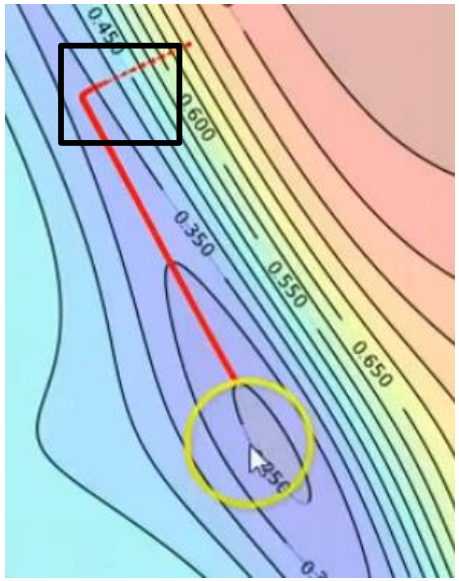
$\Rightarrow$ stochastic gradient using mini batch

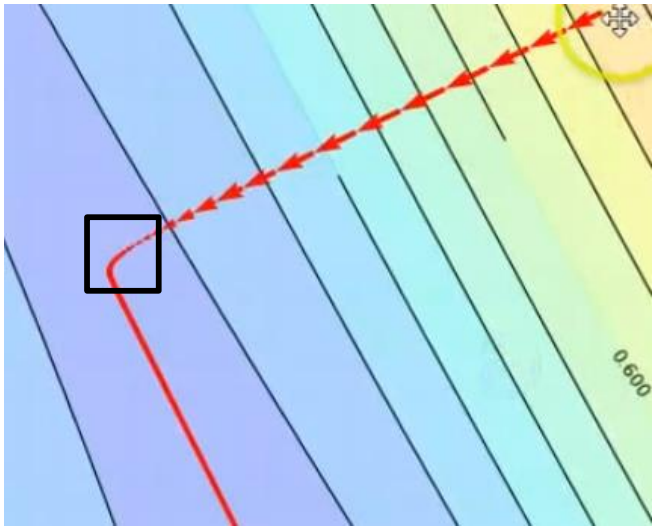
$\Rightarrow$ extream case : $m = 1$ Small m $\Rightarrow$ $Var[g(x)]$ gets bigger

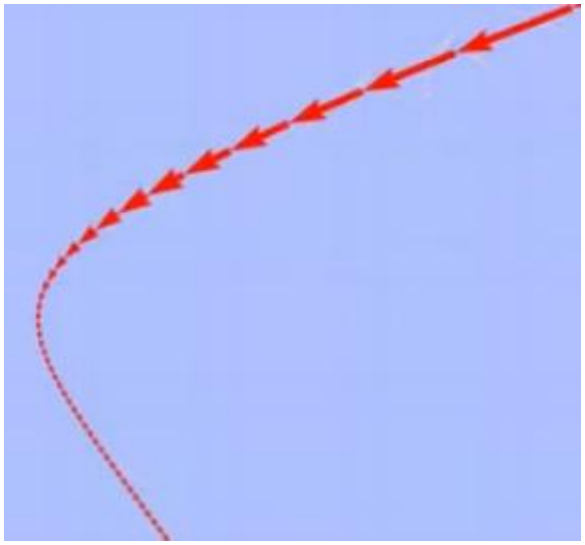
Enhancements in Optimization

- GD (Plain vanilla version)
  - $l(\boldsymbol{w})$: cost/loss/error function
  - $\boldsymbol{g}$: (an estimate of) $-\nabla l(\boldsymbol{w})$ (gradient at $\boldsymbol{w}$)
  - α : learning rate, step size
  - $\boldsymbol{v} = \alpha \boldsymbol{g}$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{v}$
  - very hard to set right step size

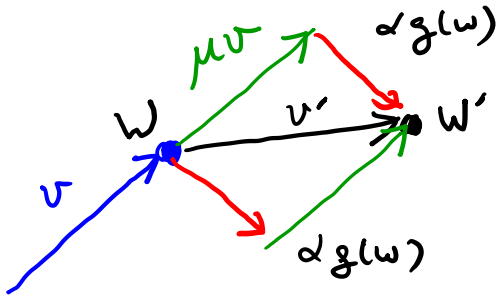




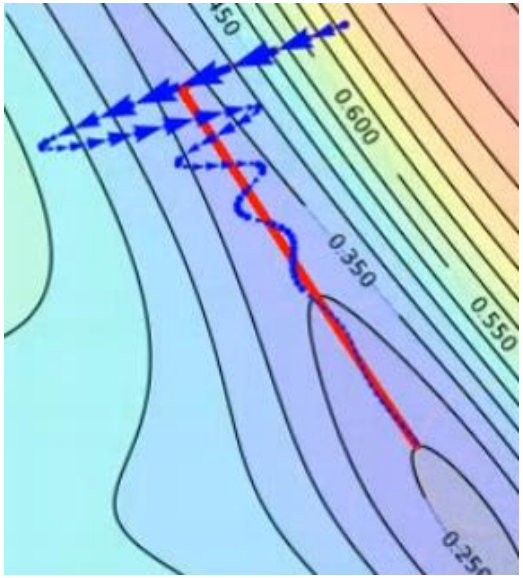


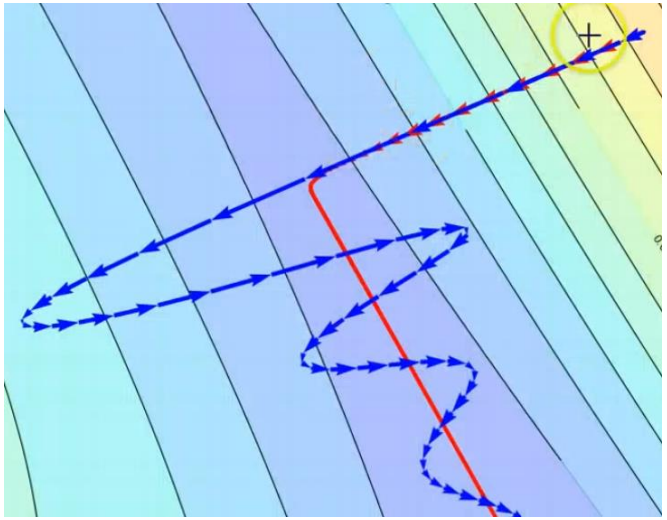


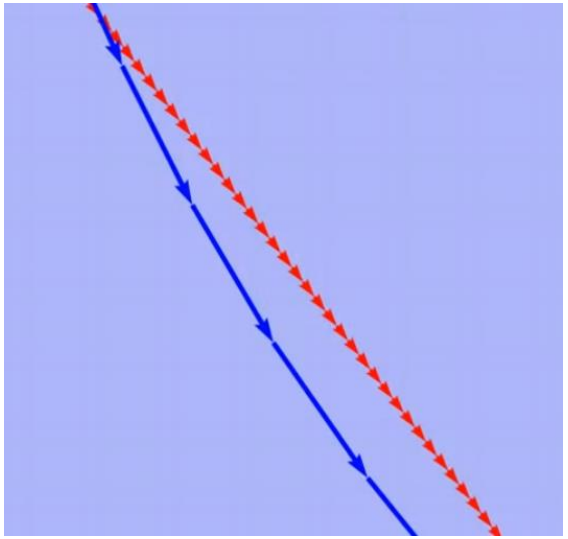
Enhancing GD : Momentum



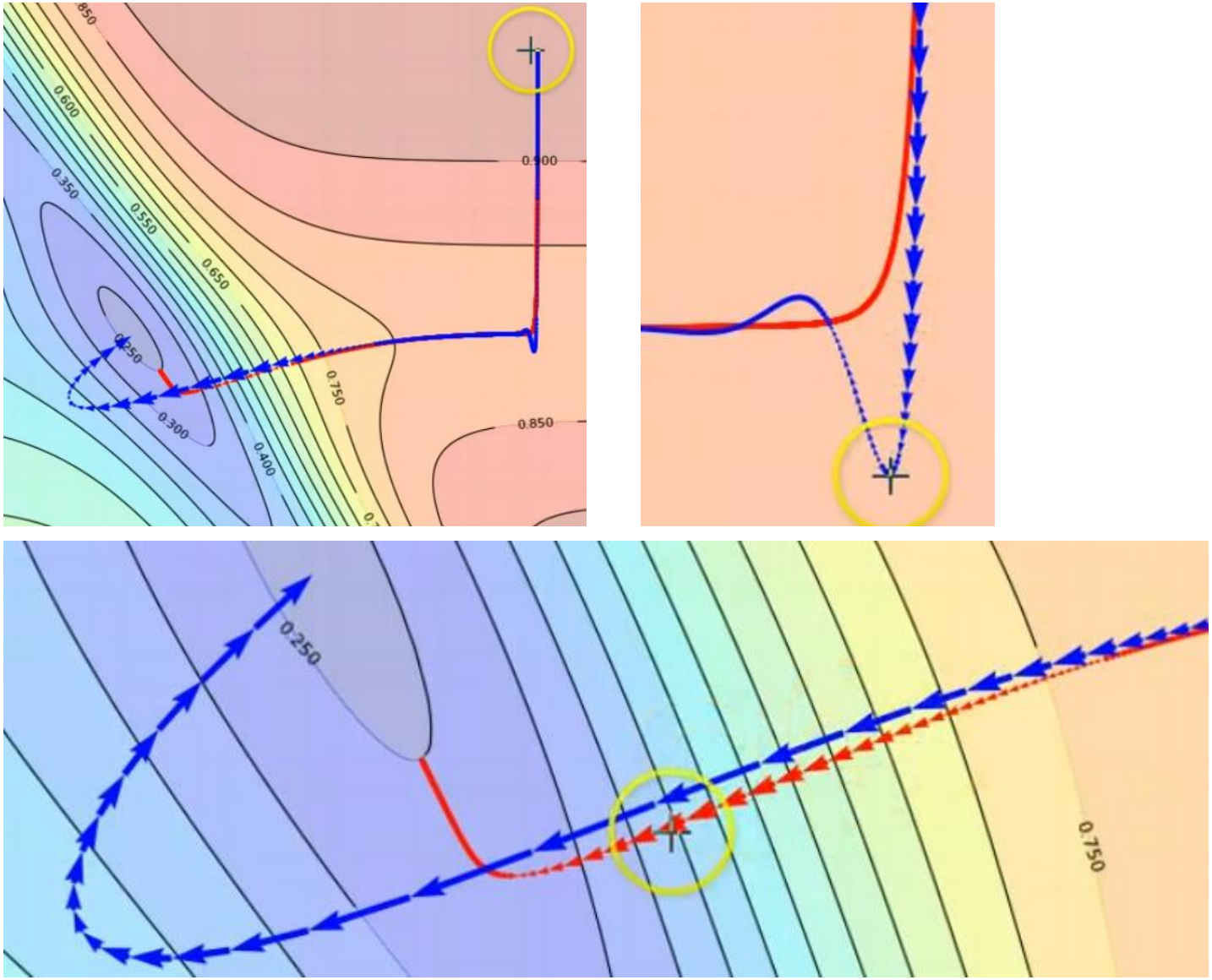
- Momentum : keep velocity of (heavy) ball
  - $\boldsymbol{v} \leftarrow \mu \boldsymbol{v} + \alpha \boldsymbol{g}$: velocity vector (initially 0)
    - μ : decaying factor, e.g. $\mu = 0.9$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{v}$
  - if $\boldsymbol{g}$ is constant, $\boldsymbol{w} = \alpha[\boldsymbol{g} + \mu \boldsymbol{g} + \mu^2 \boldsymbol{g} + \cdots] = \frac{\alpha}{1-\mu} \boldsymbol{g}$
    - if $\mu = 0.9$, the ball will move about 10 times faster





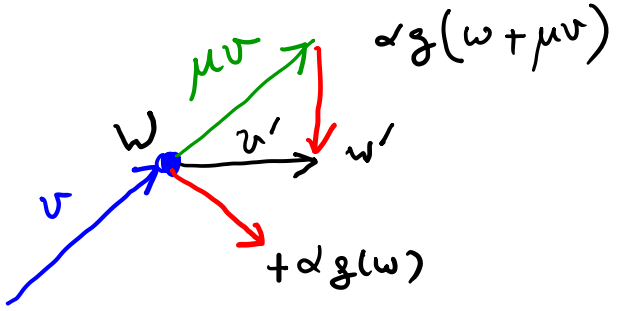


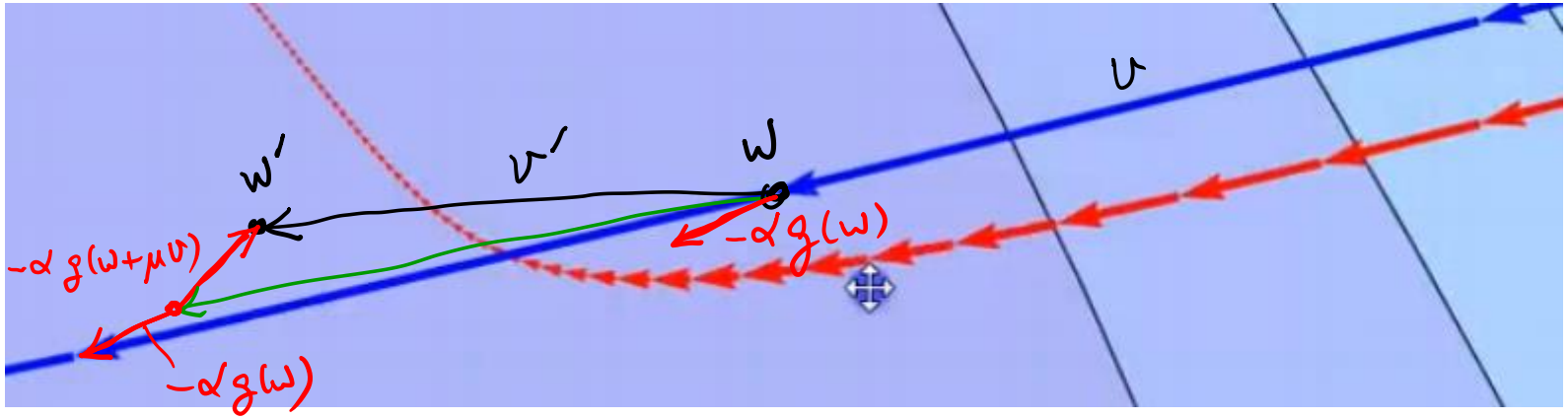
Enhancing GD : Momentum

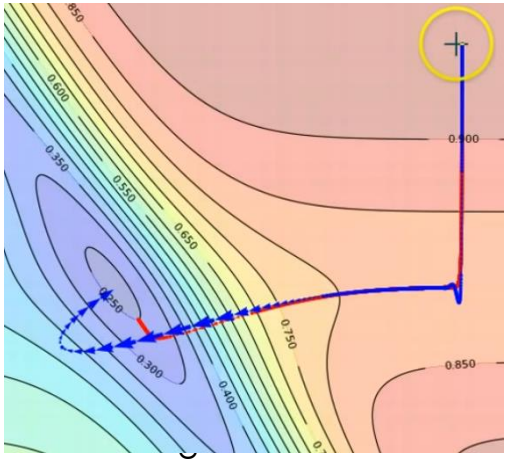


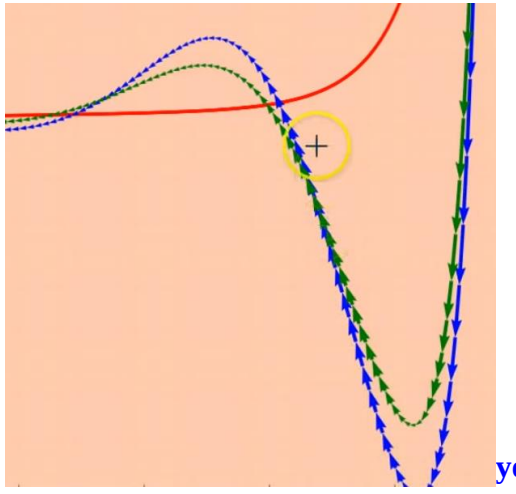
Nesterov Accelerated Gradient (Sutskever, 2013)

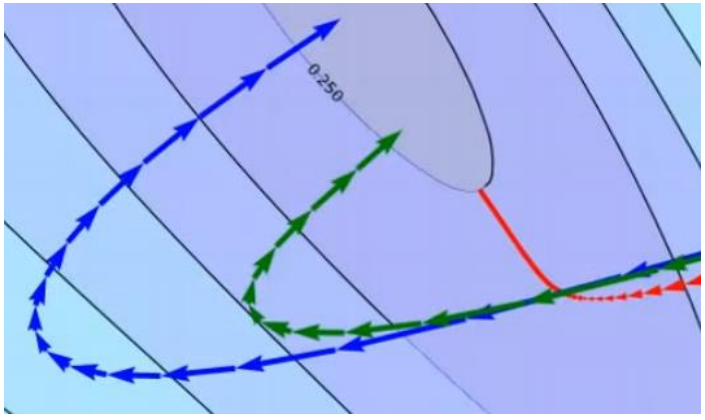
- NAG : Gamble, then correct
  - $\boldsymbol{g} = -\nabla l(\boldsymbol{w} + \mu \boldsymbol{v})$
  - $\boldsymbol{v} \leftarrow \mu \boldsymbol{v} + \alpha \boldsymbol{g}$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{v}$





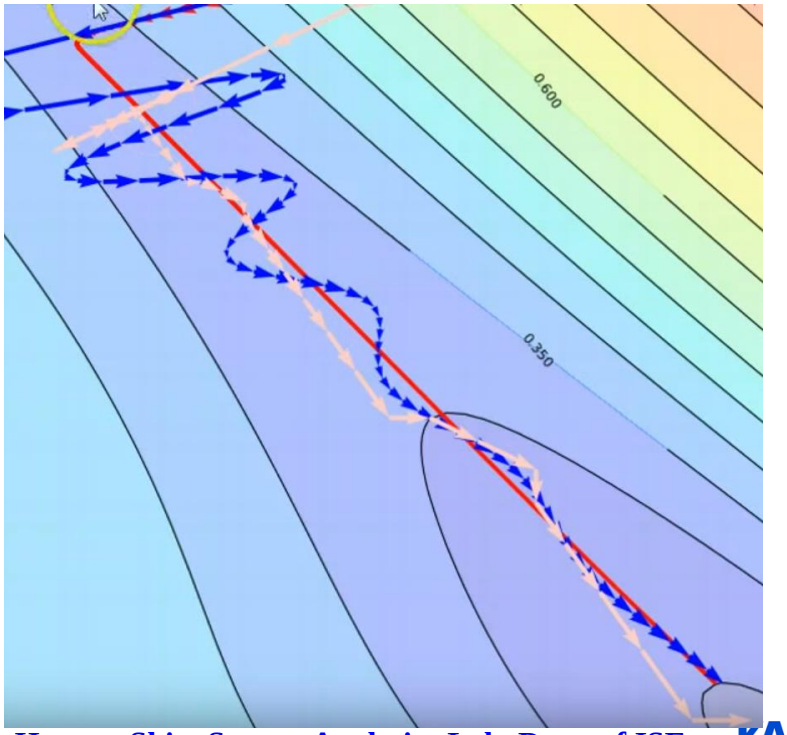






RProp (Resilient Propagation, Riedmiller & Braun, 1993)

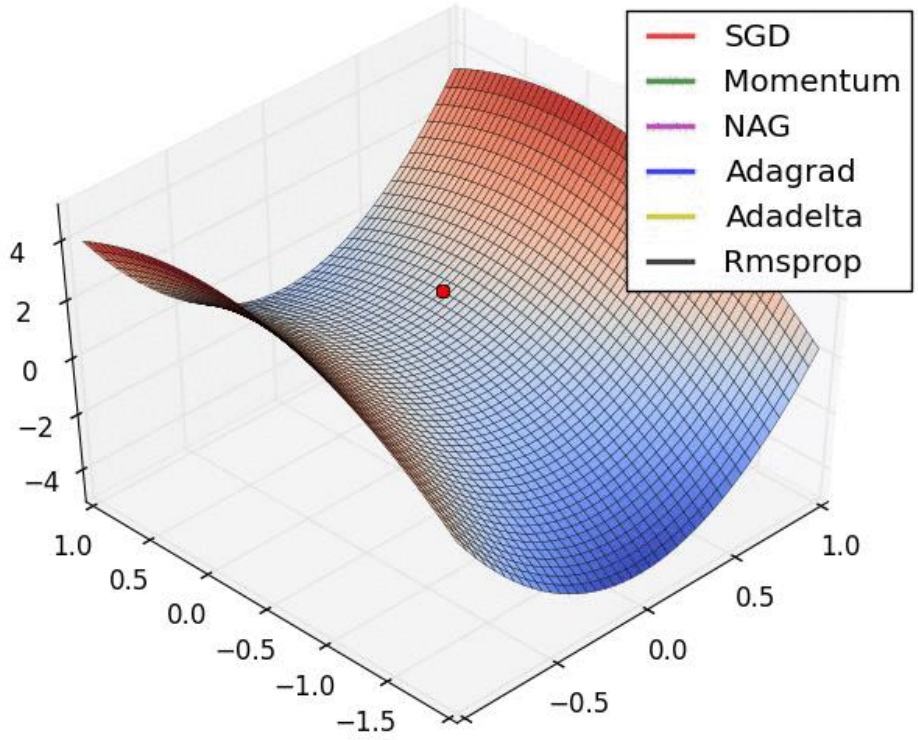
- Ignore gradient size, use only sign of gradient for each component
  - g_k : current gradient component k
  - $c_k = sign(g_k)$
  - p_k : previous sign, initially 0
- Keep step size α_k per component
  - if $p_k \cdot c_k > 0$ (same sign)
    - $\alpha_k \ *= 1.2$ (increase step size)
    - $\Delta w_k = \alpha_k \cdot c_k$
    - $p_k = c_k$
  - if $p_k \cdot c_k < 0$ (sign changes)
    - $\alpha_k \ /= 2$ (decrease step size)
    - $\Delta w_k = 0$
    - $p_k = 0$
  - else (sign changed)
    - $\Delta w_k = \alpha_k \cdot c_k$
    - $p_k = c_k$

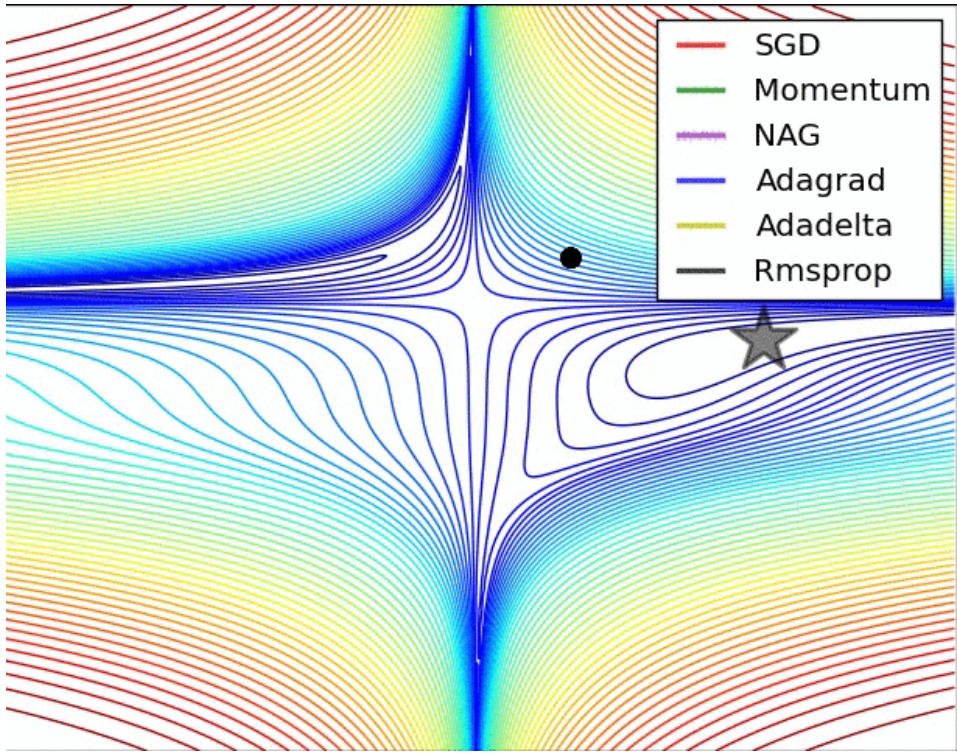


Other methods

- So far, we covered : GD, SGD, Momentum, NAG, RProp
- Recently, there are many other attempts on adaptive gradients
  - AdaGrad (Duchi et al, JMLR 2011)
  - AdaDelta (Zeiler, 2012)
  - RMSProp (Hinton, 2012)

$$\boldsymbol{p} = \sum \boldsymbol{g}^2 \,, \boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \frac{\boldsymbol{g}}{\sqrt{\boldsymbol{p}}}$$





ADAM (Kingma & Ba, 2015)

- ADAM : ADAptive Moment estimation
  - Combines benefits of AdaGrad (Duchi 2011), RMSProp (Hinton 2012)
  - Note element-wise operations below : $\boldsymbol{g}^2$, $\sqrt{\hat{\boldsymbol{p}}}$, and $\frac{\hat{\boldsymbol{v}}}{\sqrt{\hat{\boldsymbol{p}}}+\epsilon}$
    - Adaptive step size per component
  - Initialize : $\boldsymbol{v}=\boldsymbol{0}, \boldsymbol{p}=\boldsymbol{0}$, $\beta_1=0.9$, $\beta_2=0.999$, $\epsilon=0.000001$

$\boldsymbol{g}=-\nabla l(\boldsymbol{w})$: opposite of gradient

$\boldsymbol{v}\leftarrow\beta_1\boldsymbol{v}+(1-\beta_1)\boldsymbol{g}$: update velocity (1st moment) estimate

- $\boldsymbol{v}$: momentum + gradient ("signal")
- bigger for components with consistent sign, smaller for sign changing component

$\boldsymbol{p}\leftarrow\beta_2\boldsymbol{p}+(1-\beta_2)\boldsymbol{g}^2$: update 2nd moment estimate

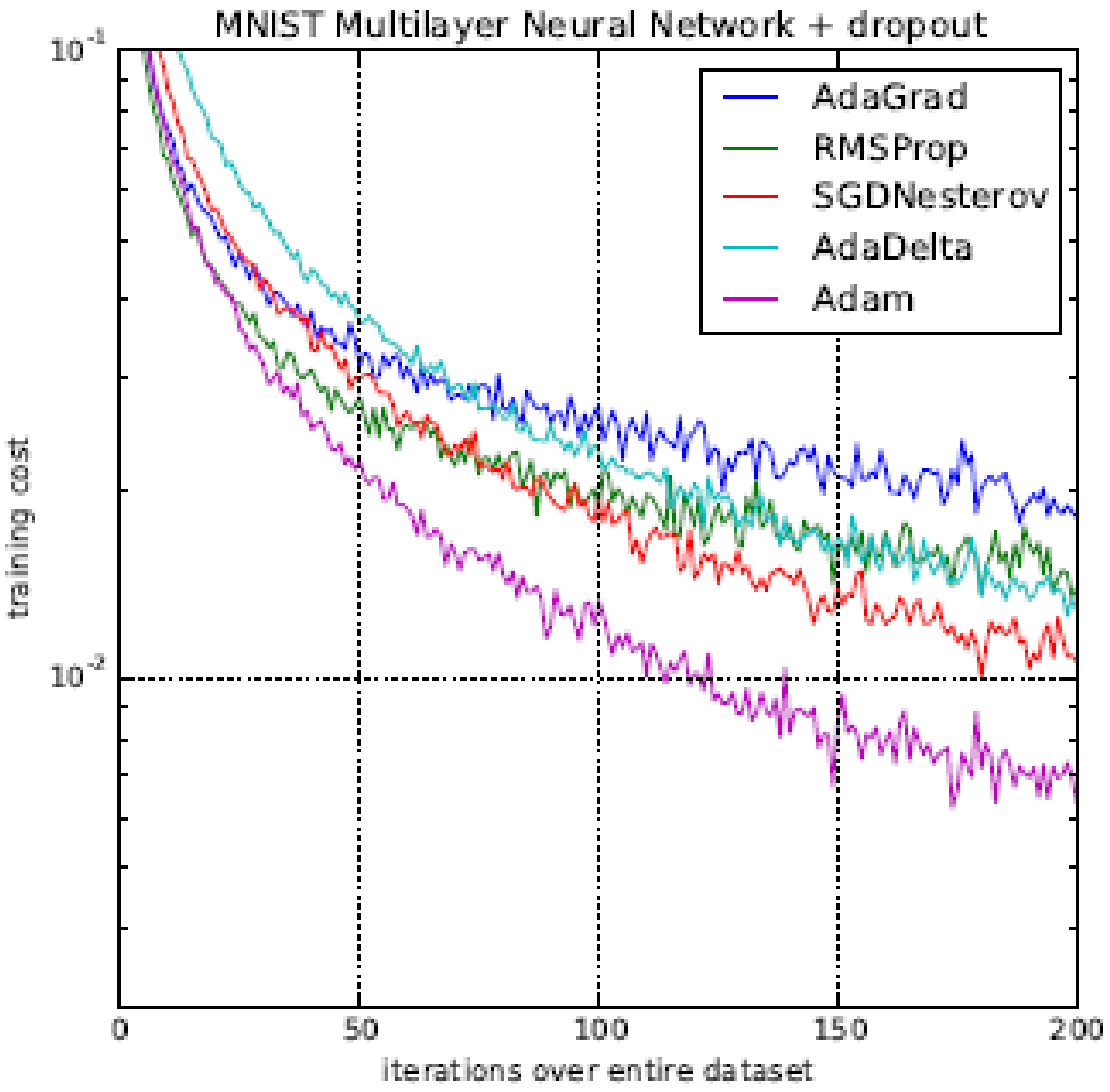
- $\boldsymbol{p}$: gradient variance ("noise")
- penalize components with bigger variance
- compensate scale difference between components

$\hat{\boldsymbol{v}}\leftarrow\frac{\boldsymbol{v}}{1-\beta_1^t}$, $\hat{\boldsymbol{p}}\leftarrow\frac{\boldsymbol{p}}{1-\beta_2^t}$: initial size bias correction

- $\boldsymbol{v}$ and $\boldsymbol{p}$ are small at the beginning
- $\beta_1^t=\beta_1\ \beta_1^{t-1}\rightarrow 0$ and $\beta_2^t=\beta_2\ \beta_2^{t-1}\rightarrow 0$

$\boldsymbol{w}\leftarrow\boldsymbol{w}+\alpha\frac{\hat{\boldsymbol{v}}}{\sqrt{\hat{\boldsymbol{p}}}+\epsilon}$: correct velocity with 2nd moment

Comparison


MNIST Multilayer Neural Network + dropout
AdaGrad
RMSProp
SGDNesterov
AdaDelta
Adam
training cost
iterations over entire dataset
0
50
100
150
200