Lesson 4: Optimization Algorithms

Introduction to Deep Learning

Sungbin Lim (SME)

Review

Previously we learned…

What is regularization?

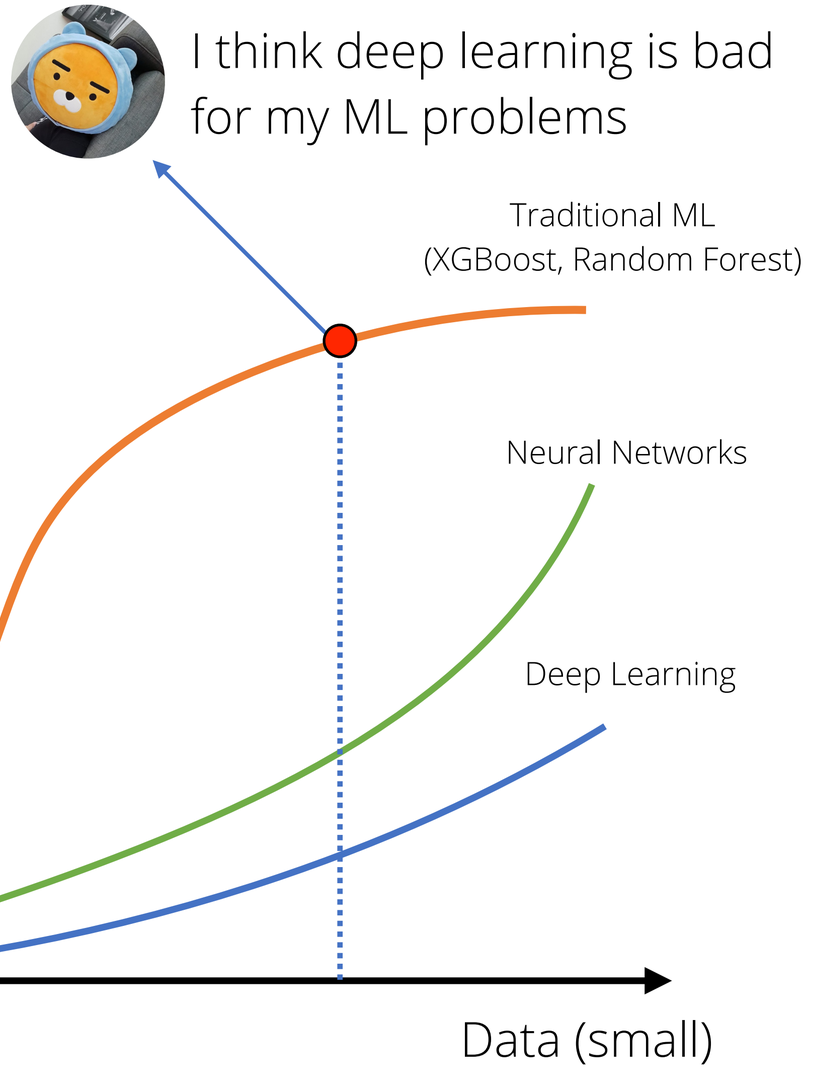
- In the view of statistical learning theory, regularization is the process of adding information in order to solve an overfitting
  - Sparsity
  - Uncertainty

Here the information means hypothesis restricting the parameter search space for ML models

- Regularization applies to objective function in ill-posed optimization problems
  - Function spaces (smoothness)
  - Constraint

Gather more fine data

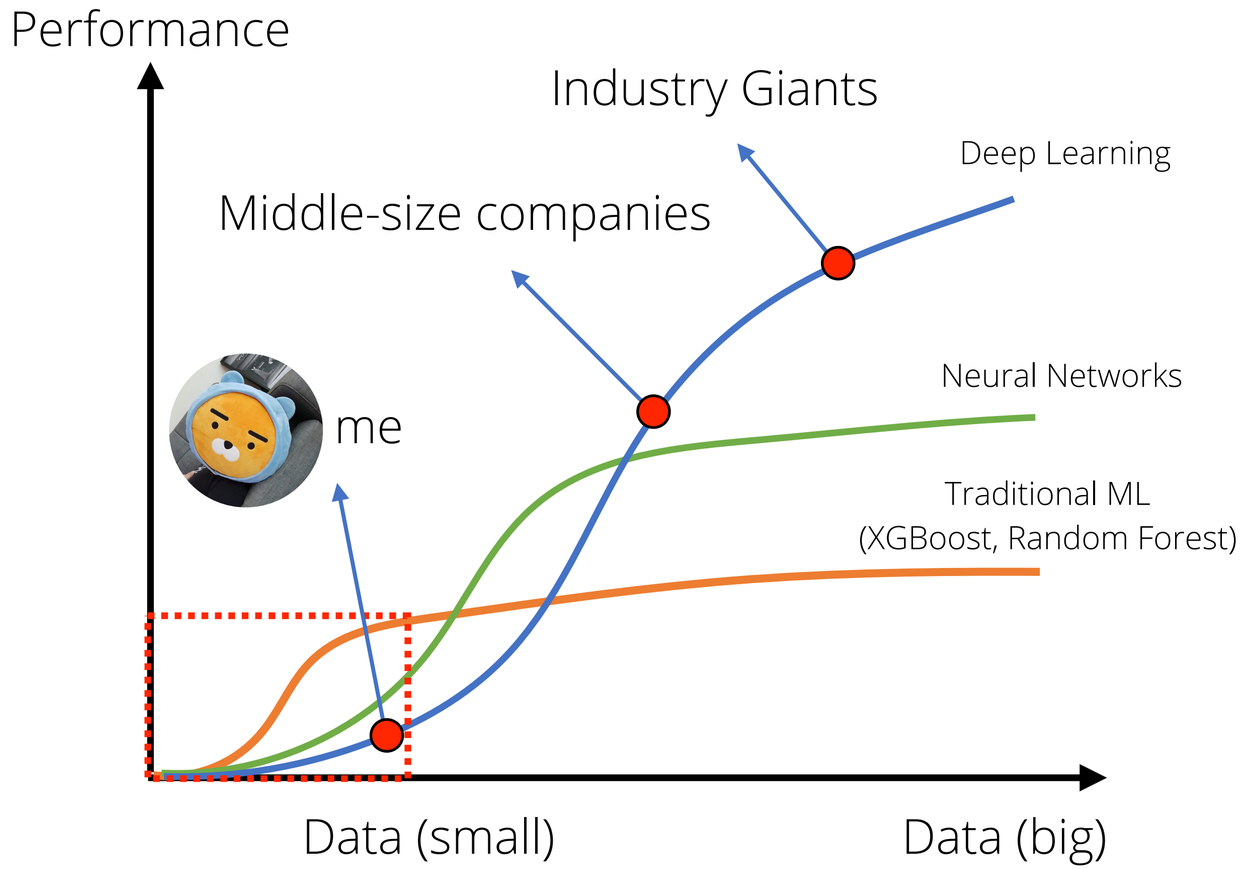
- Golden rule of deep learning
  - small data is not for DL



How Scale is Enabling Deep Learning, Andrew Ng, 2016

Gather more fine data

- Golden rule of deep learning
  - small data is not for DL
  - DL is good at big and high dimensional data
- How big?
  - You need at least hundreds of GPUs to train those data



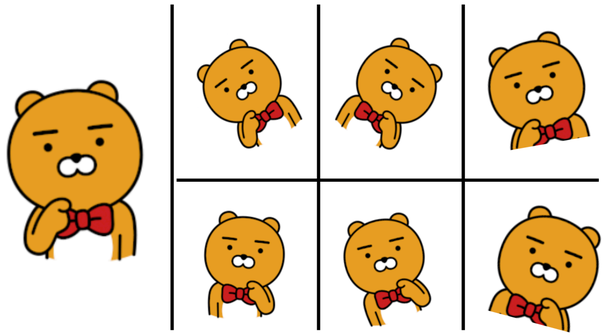
How Scale is Enabling Deep Learning, Andrew Ng, 2016

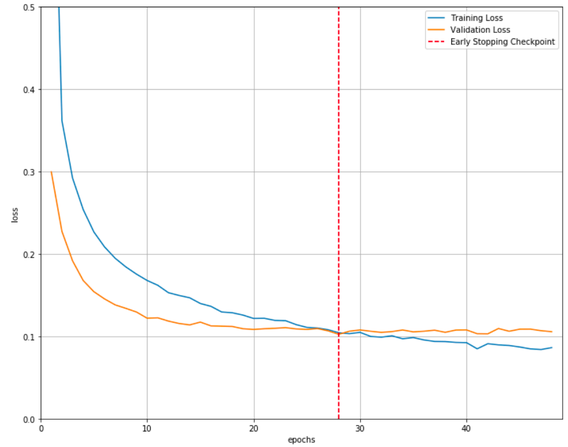
Regularization methods

- Data augmentation
- Weight decay
- Dropout
- Early stopping
- Label smoothing
- L_1, L_0 - regularization
- DropConnect
- Ensemble methods
- Adversarial training
- Double back-propagation

there are so many ways!

$$\min_{\theta} \left(L(f_\theta(x), y) + \lambda \|\theta\|_2^2 \right)$$





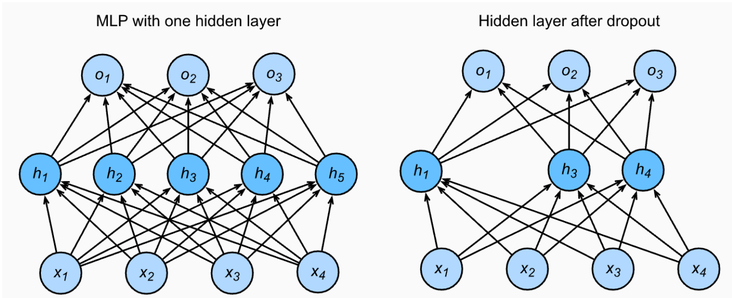


Fig. 4.6.1 MLP before and after dropout

$$\tilde{y} = (1 - \epsilon)y + \frac{\epsilon}{C}$$

Instability of gradients

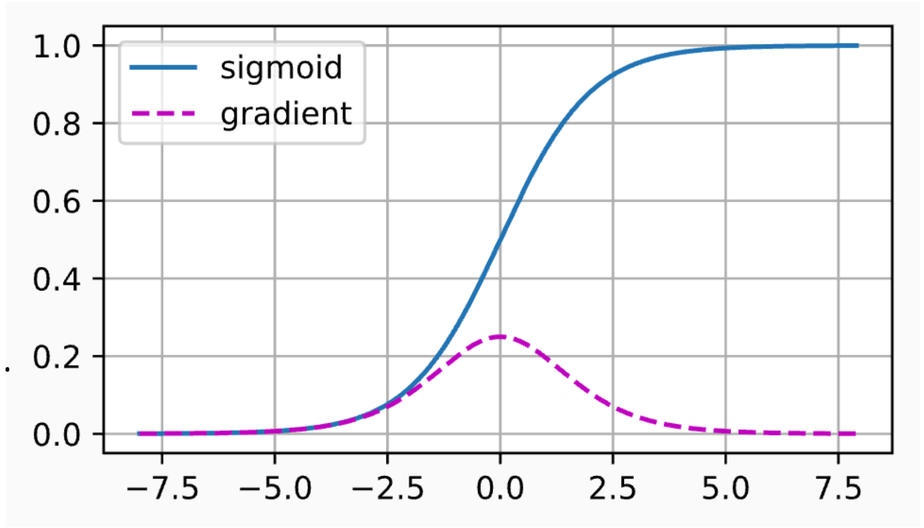
- In deep learning, the gradient for parameter update equals to the product of the gradient of hidden variables
- If we use sigmoid as an activation function in deep neural network architecture, then gradient vanishes

multi-layer

$$\mathbf{h}^{k+1} = f_k(\mathbf{h}^k) \quad y = f_L \circ \cdots \circ f_1(\mathbf{x})$$

$$\partial_\theta y = \partial_\theta \mathbf{h}^\ell \prod_{k=\ell+1}^{L} \partial_{\mathbf{h}^{k-1}} \mathbf{h}^k$$

So I said your are fired..



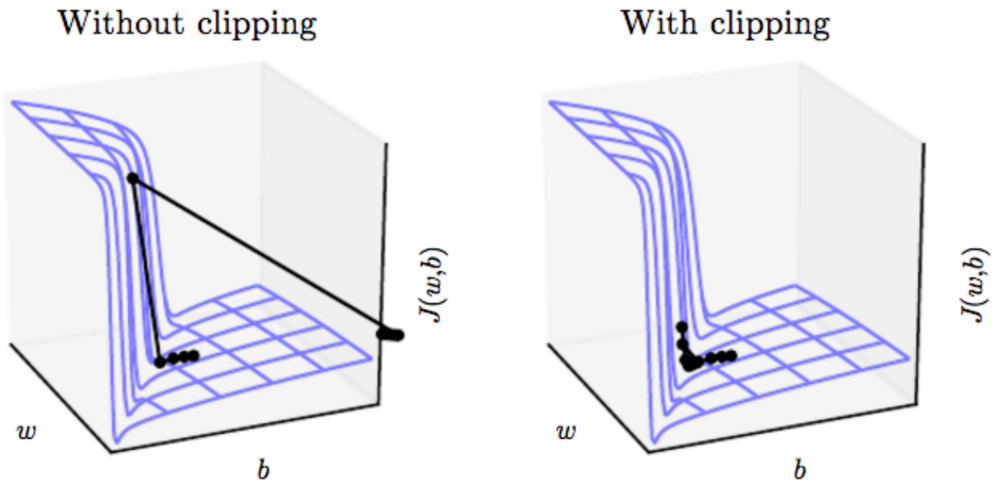
Instability of gradients

- In deep learning, the gradient for parameter update equals to the product of the gradient of hidden variables
- If we use sigmoid as an activation function in deep neural network architecture, then gradient vanishes
- On the other hand, gradient explosion can happen when the parameter norm is too huge → need gradient clipping!

multi-layer

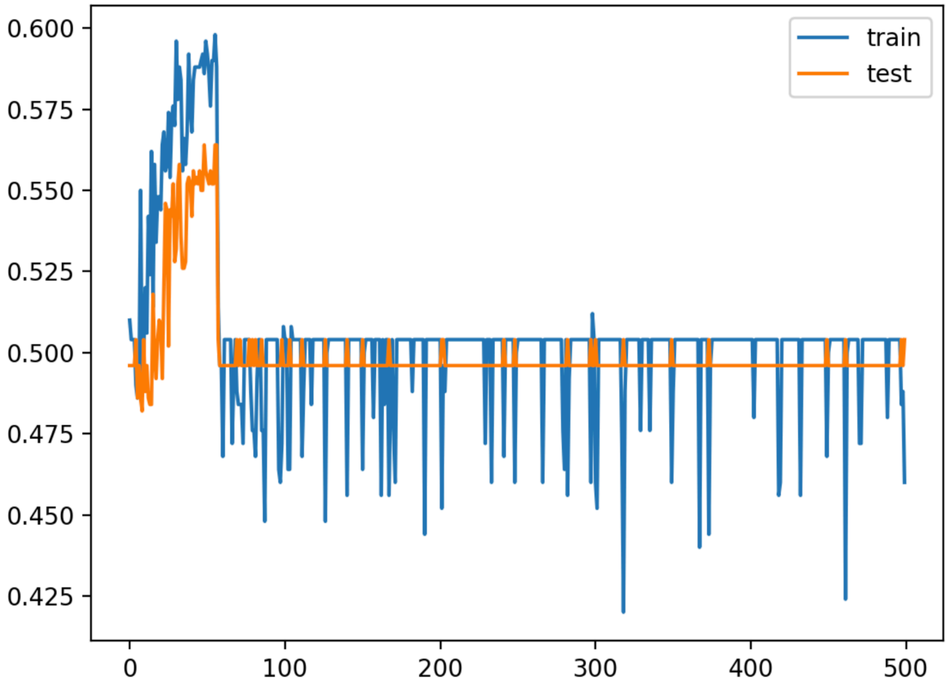
$$\mathbf{h}^{k+1} = f_k(\mathbf{h}^k) \quad y = f_L \circ \cdots \circ f_1(\mathbf{x})$$

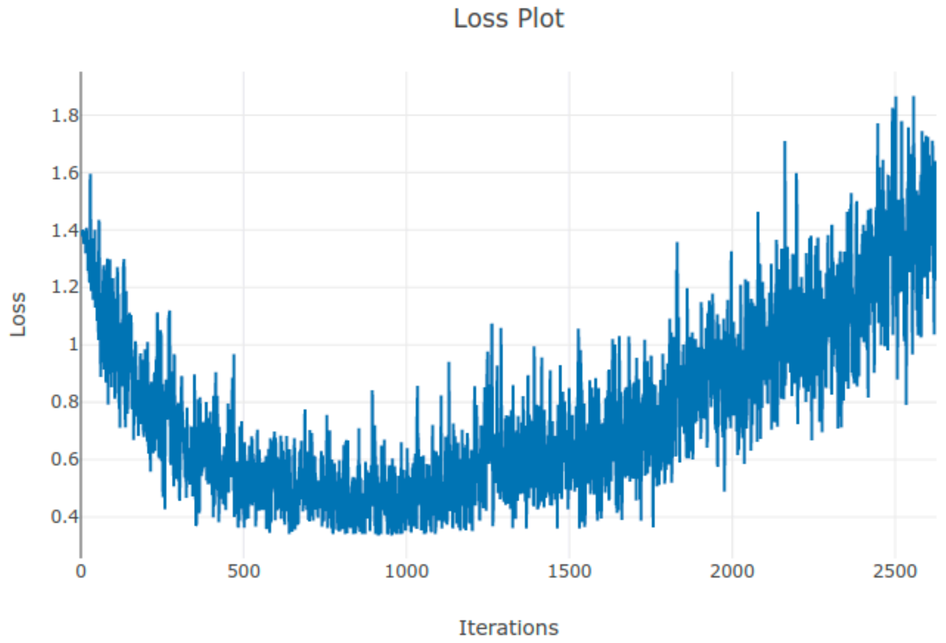
$$\partial_\theta y = \partial_\theta \mathbf{h}^\ell \prod_{k=\ell+1}^{L} \partial_{\mathbf{h}^{k-1}} \mathbf{h}^k$$



How can we detect those things?

- Method 1: observe the training loss or metric carefully





Both are typical examples of gradient explosion

Parameter initialization

- Quiz: if we set parameters equal at each layer, what will happen?
  - **Answer**: every parameter get the same update!
- We need to initialize neural net parameters breaking the symmetry of parameters on each layer
  - Xavier initialization: sigmoid, hyperbolic tangent
  - He initialization: ReLU
- Initialization can prevent the gradient vanishing problem

Optimization in Deep Learning

Heart of Deep Learning

Optimization & Deep Learning

- Optimization algorithms are important for deep learning
  - training can take hours, days, or even weeks
  - algorithms directly affect the model's efficiency

$$\underset{\theta}{\text{minimize}}\ L(f_\theta(\mathbf{x}), \mathbf{y}), \qquad (\mathbf{x}, \mathbf{y}) \in \mathcal{D}$$

data

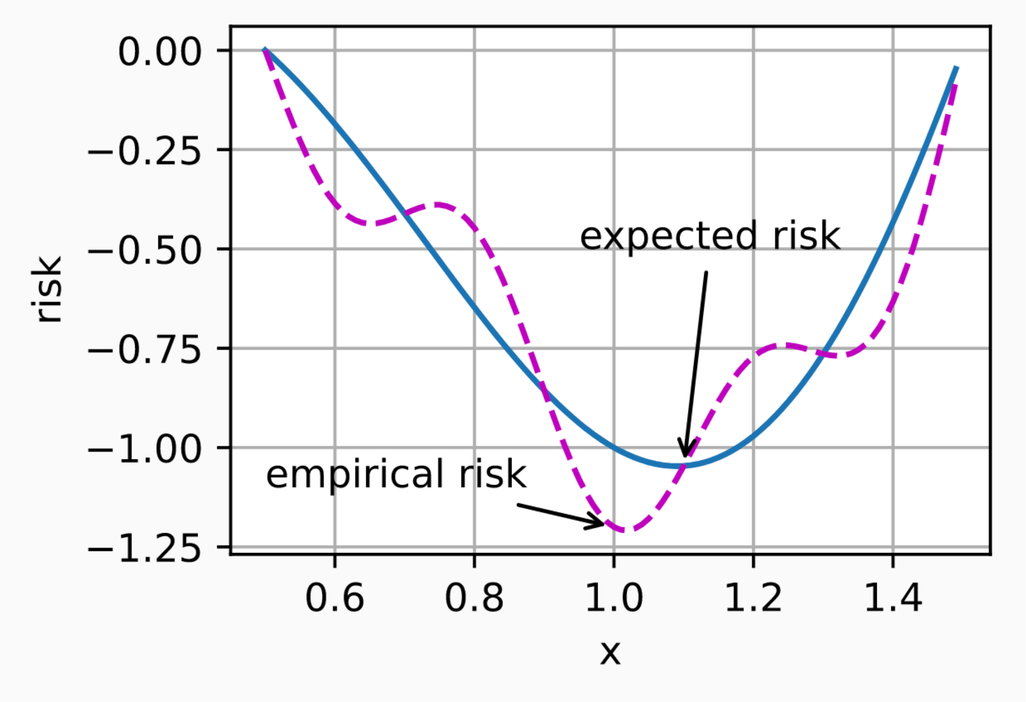
parameters

loss function

model (neural net)

Optimization ≠ Deep Learning

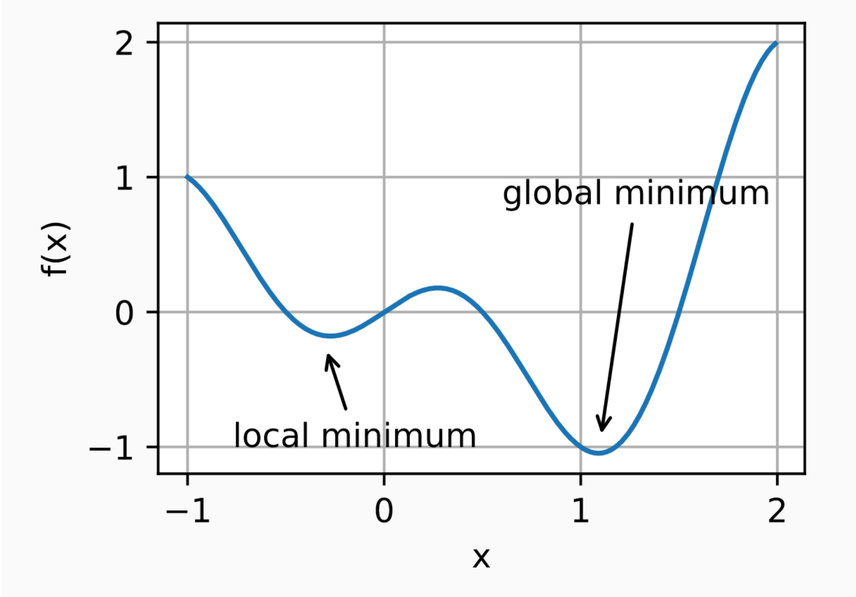
- Almost all optimization problems arising in DL are non-convex
- Optimization ≠ Deep Learning
  - goal of optimization is to reduce the training error
  - goal of statistical inference is to reduce the generalization error

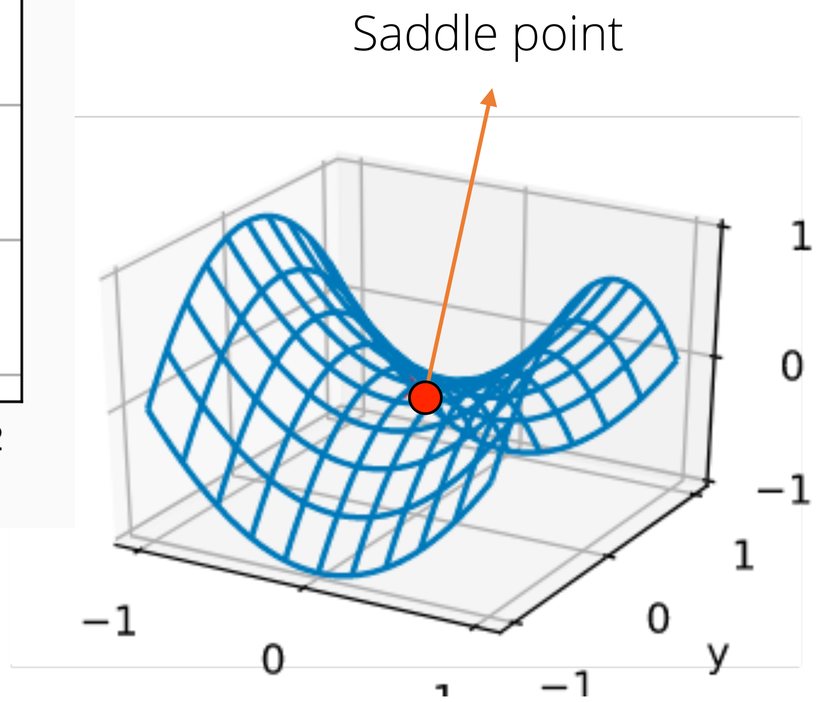


$$L_{\text{train}} \neq L_{\text{test}}$$

Optimization challenges in DL

- There are many challenges in deep learning optimizations
  - Local minima
  - Saddle points
  - Vanishing gradients
  - Gradient explosion

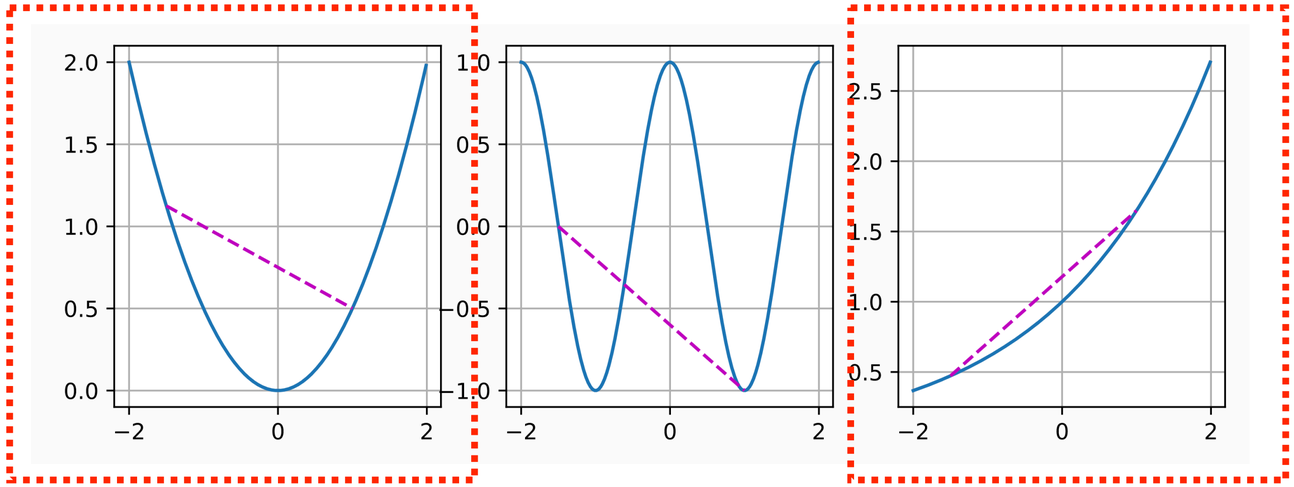




Convex optimization

DL is not convex optimization. But theories guide us to understand the algorithms deeply

- Suppose f is convex function: $\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\lambda x_1 + (1-\lambda) x_2)$



Convex optimization

- Suppose f is convex function: $\lambda f(x_1) + (1-\lambda) f(x_2) \geq f(\lambda x_1 + (1-\lambda) x_2)$

  Then f satisfies followings:
  - Jensen inequality
  - No local minima
  - Convex below-sets
  - Nonnegative-definite Hessian
    every eigenvalues of Hessian are nonnegative

$$\int_\Omega f \circ g(X) \mathrm{d}\mathbb{P} \geq f\left(\int_\Omega g(X) \mathrm{d}\mathbb{P}\right) = \mathbb{E}[g(X)]$$

$$\int P(x|y)P(y)dy = P(x)$$

here g is not necessarily convex!

$$E_{y\sim P(y)}[-\log P(x \mid y)] \geq -\log P(x).$$

Convex optimization

- If objective function is convex, then it allows us to handle constraints efficiently
- Solving a constrained optimization is difficult

$$\underset{\mathbf{x}}{\text{minimize}}\ f(\mathbf{x})$$

$$\text{subject to } c_i(\mathbf{x}) \leq 0 \text{ for all } i \in \{1, \ldots, N\}.$$

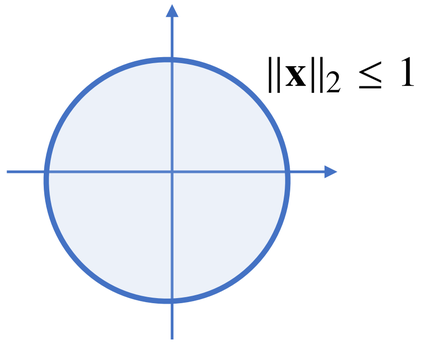
constraint functions

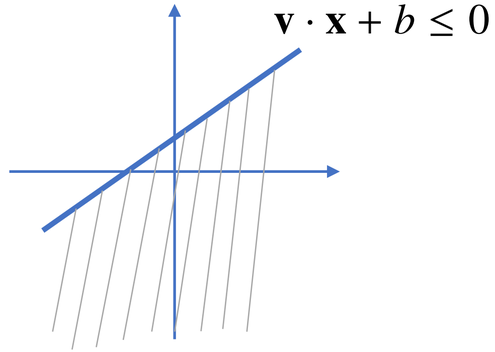
not exact, but this works for learning problems

$$L(\mathbf{x}, \alpha) = f(\mathbf{x}) + \sum_i \alpha_i c_i(\mathbf{x}) \text{ where } \alpha_i \geq 0.$$

ex) Ridge, Lasso, Elastic Net

$$c_1(\mathbf{x}) = \|\mathbf{x}\|_2 - 1 \quad c_2(\mathbf{x}) = \mathbf{v} \cdot \mathbf{x} + b$$





Convex optimization

- If objective function is convex, then it allows us to handle constraints efficiently
- Solving a constrained optimization is difficult

$$\underset{\mathbf{x}}{\text{minimize}}\ f(\mathbf{x})$$

$$\text{subject to } c_i(\mathbf{x}) \leq 0 \text{ for all } i \in \{1, \ldots, N\}.$$

constraint functions

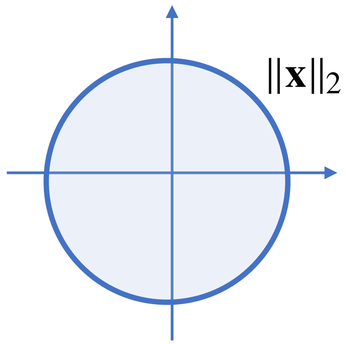
projection is another way for satisfying constraints

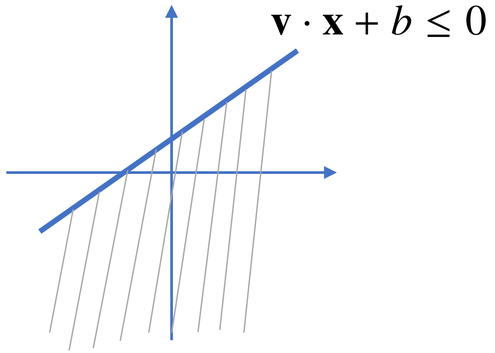
$$\text{Proj}_X(\mathbf{x}) = \underset{\mathbf{x}' \in X}{\text{argmin}} \|\mathbf{x} - \mathbf{x}'\|_2.$$

$$\mathbf{g} \leftarrow \mathbf{g} \cdot \min(1, c/\|\mathbf{g}\|).$$

ex) gradient clipping

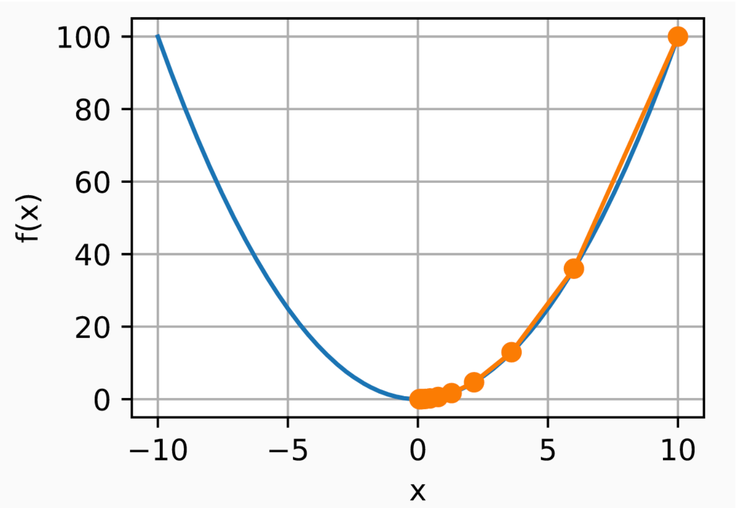
$$c_1(\mathbf{x}) = \|\mathbf{x}\|_2 - 1 \quad c_2(\mathbf{x}) = \mathbf{v} \cdot \mathbf{x} + b$$





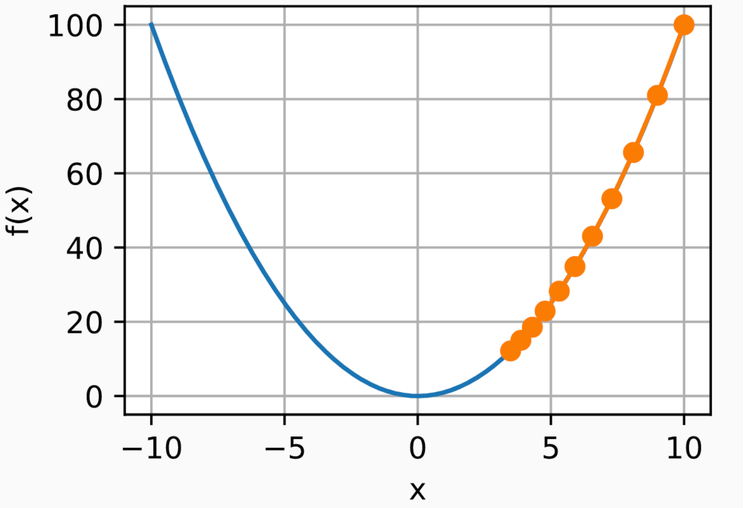
Gradient descent

objective function



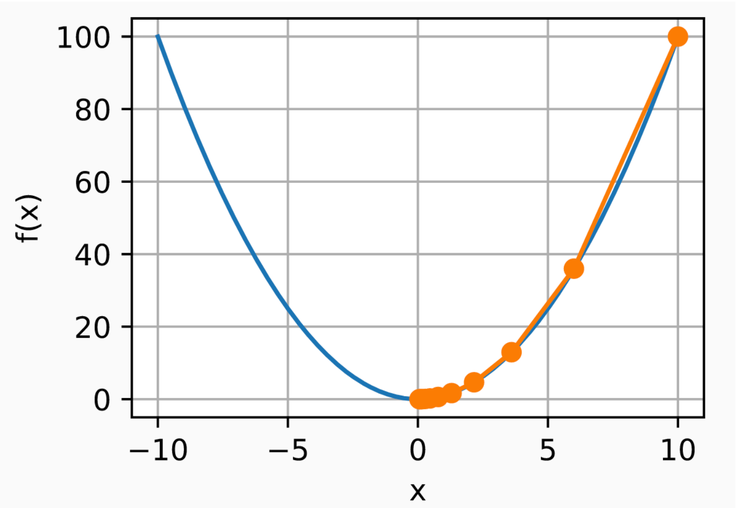
- If we use gradient descent iteratively the objective function might decline (why?)
- Learning rate is the key hyperparameter
  - if η is too small, then the algorithm updates the objective slowly

$$x \leftarrow x - \eta f'(x)$$



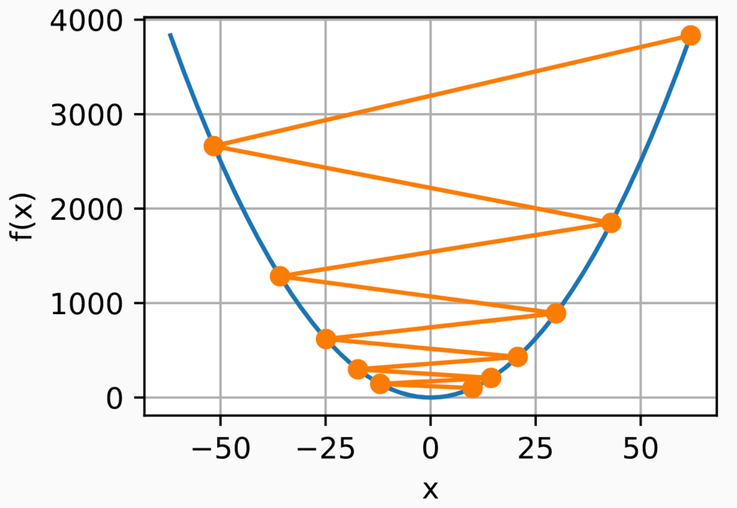
Gradient descent

objective function



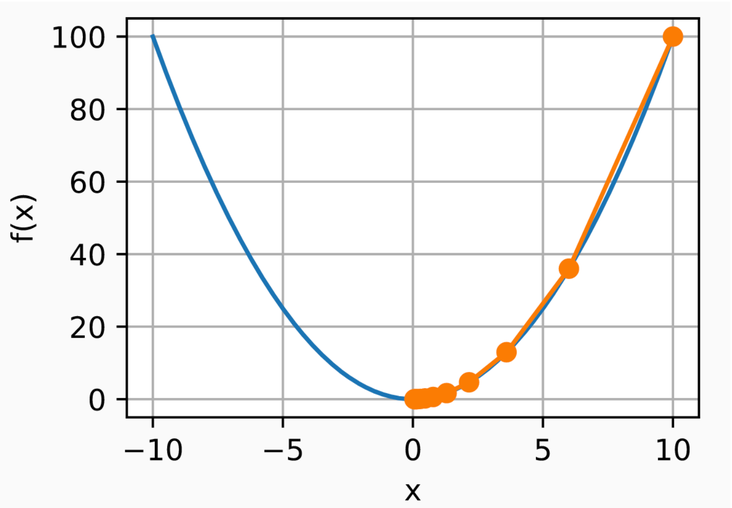
- If we use gradient descent iteratively the objective function might decline (why?)
- Learning rate is the key hyperparameter
  - if η is too small, then the algorithm updates the objective slowly
  - if η is too large, then the algorithm overshoots and diverges

$$x \leftarrow x - \eta f'(x)$$



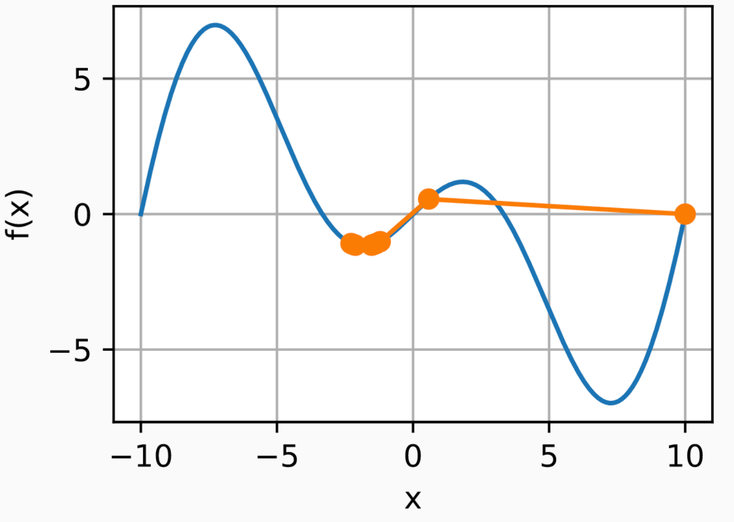
Gradient descent

objective function



- If we use gradient descent iteratively the objective function might decline (why?)
- Learning rate is the key hyperparameter
  - if η is too small, then the algorithm updates the objective slowly
  - if η is too large, then the algorithm overshoots and diverges
- What happens for non-convex objective?

$$x \leftarrow x - \eta f'(x)$$



Adaptive methods

- Getting the learning rate “just right” is tricky
  - we can use Newton’s method
  - or BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm
  - …or Limited-memory BFGS
- Preconditioning?
- Line search?

these are suggested already for deep learning
… but we use SGD more nowadays. Why?

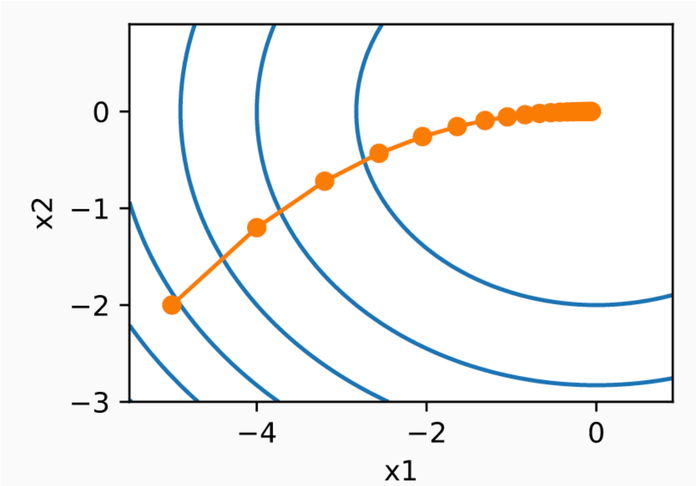
Stochastic Gradient Descent

- SGD reduces computational cost
  - on average SGD is good estimate of GD
  - shows more noisy trajectory than GD
- Dynamic learning rate

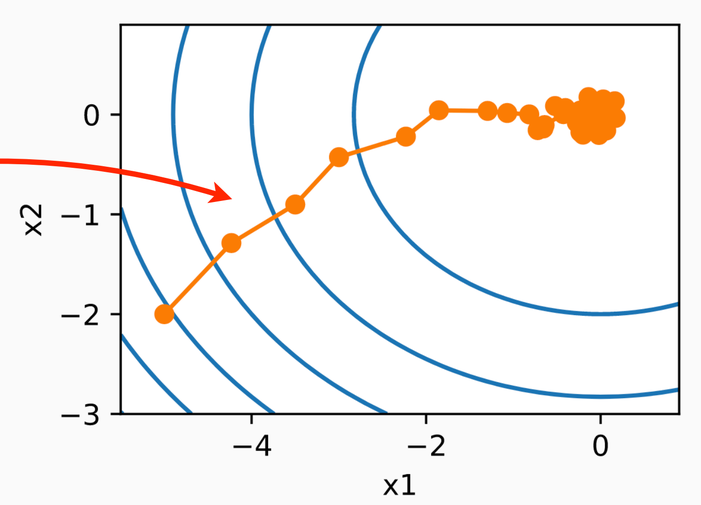
$$\eta(t) = \eta_i \text{ if } t_i \leq t \leq t_{i+1}$$
$$\eta(t) = \eta_0 \cdot e^{-\lambda t}$$
$$\eta(t) = \eta_0 \cdot (\beta t + 1)^{-\alpha}$$

$$\mathbb{E}_i \nabla f_i(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}).$$

hmm... can we trust this even for non-convex?



Gradient descent



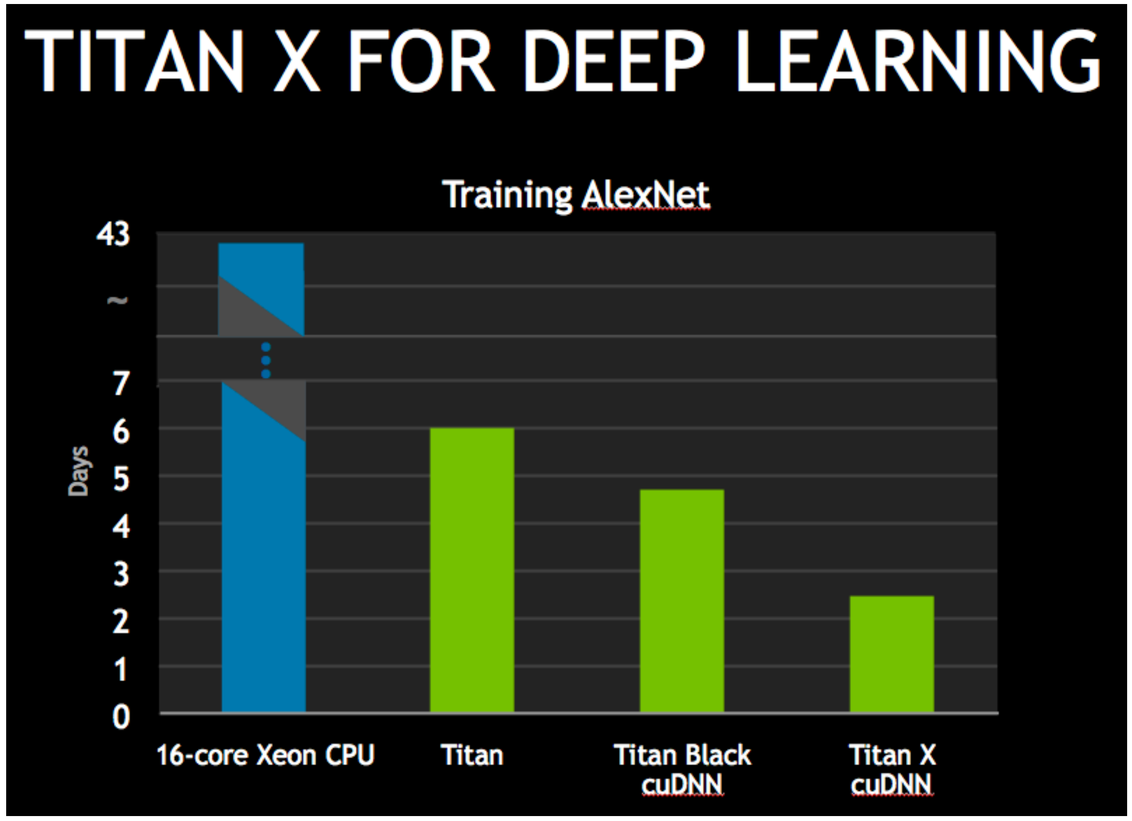
Stochastic gradient descent

Heart of Deep Learning

WARNING: Don't do mining using lab GPUs

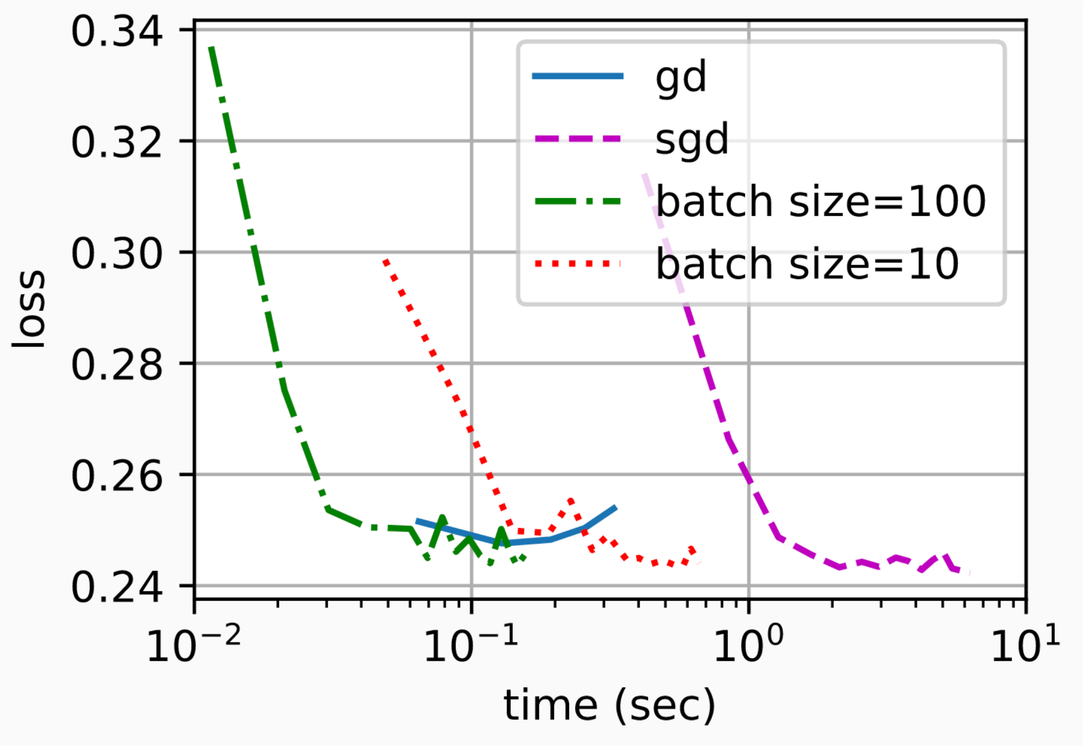
I can also earn some Bitcoins...

- What is the heart of deep learning?
  - parallelization
    - to multiple GPUs
    - to multiple servers



Heart of Deep Learning

- What is the heart of deep learning?
  - parallelization
    - to multiple GPUs
    - to multiple servers
- Use a hierarchy of CPU caches to supply the processor with data
- Mini-batch SGD reduces overhead when updating parameters

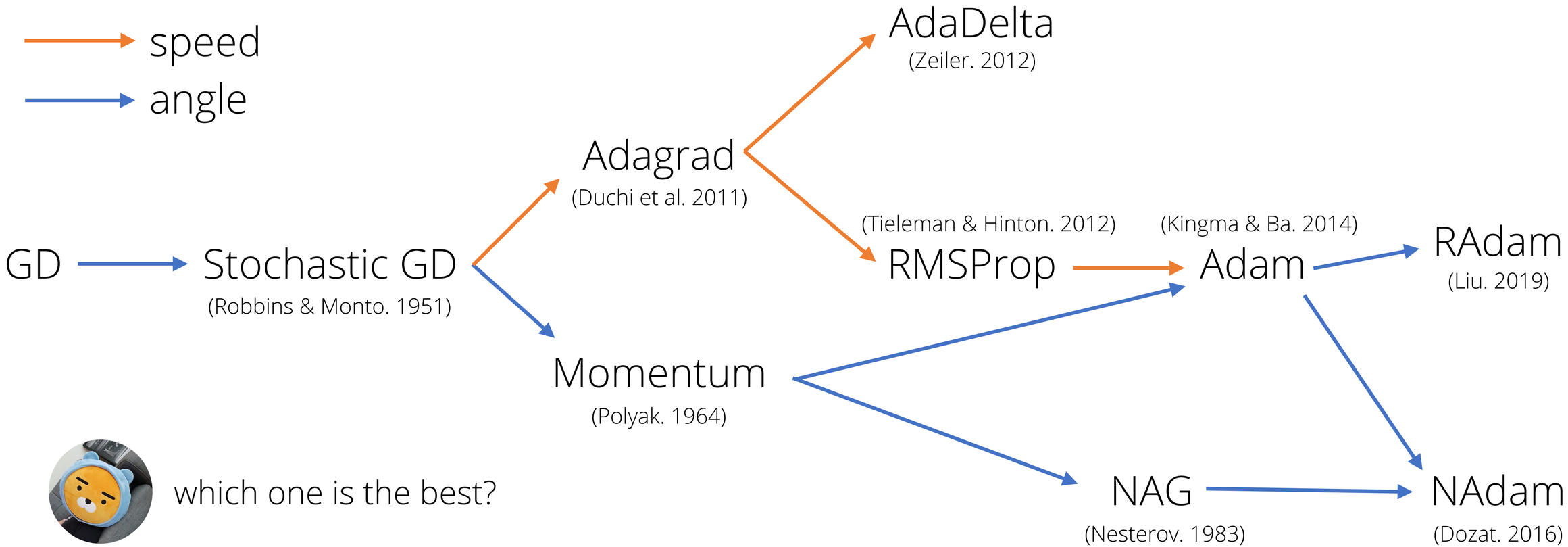


oops.. but the mini batch size is another hyperparameter!

Variants of SGD

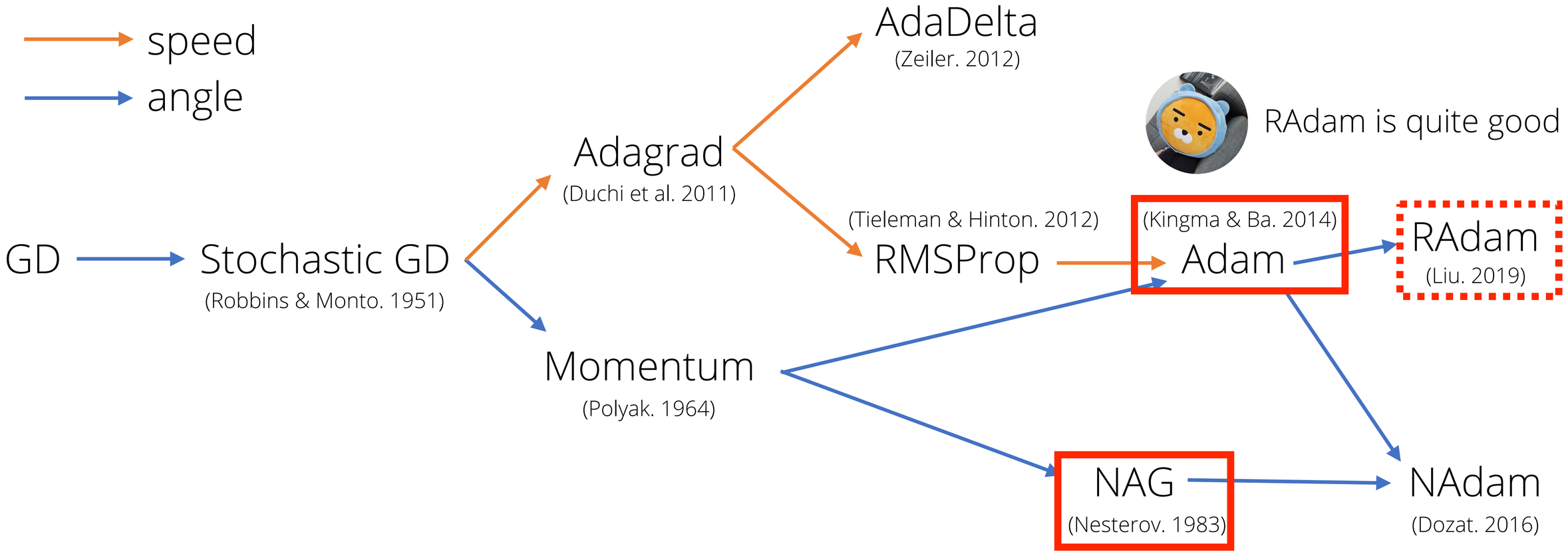
How to boost the training speed?

Variants of stochastic gradient descent



which one is the best?

Variants of stochastic gradient descent



Momentum

$$\mathbf{v}_t \leftarrow \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1},$$

gradient vector

$$\mathbf{g}_{ii} = \partial_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}_t)$$

$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \eta_t \mathbf{v}_t$$

- Replaces gradients with a leaky average over past gradients
  - $\mathbf{v}$ is called momentum and $\beta \in (0,1)$

Some methods of speeding up the convergence of iteration methods, Polyak, 1964

Momentum

$$\mathbf{v}_t \leftarrow \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1},$$

gradient vector

$$\mathbf{g}_{ii} = \partial_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}_t)$$

$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \eta_t \mathbf{v}_t$$

- Replaces gradients with a leaky average over past gradients
  - $\mathbf{v}$ is called momentum and $\beta \in (0,1)$
  - accumulates past gradients

$$\mathbf{v}_t = \beta^2 \mathbf{v}_{t-2} + \beta \mathbf{g}_{t-1,t-2} + \mathbf{g}_{t,t-1} = \cdots = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau,t-\tau-1}$$

Some methods of speeding up the convergence of iteration methods, Polyak, 1964

Momentum

$$\mathbf{v}_t \leftarrow \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1},$$

gradient vector

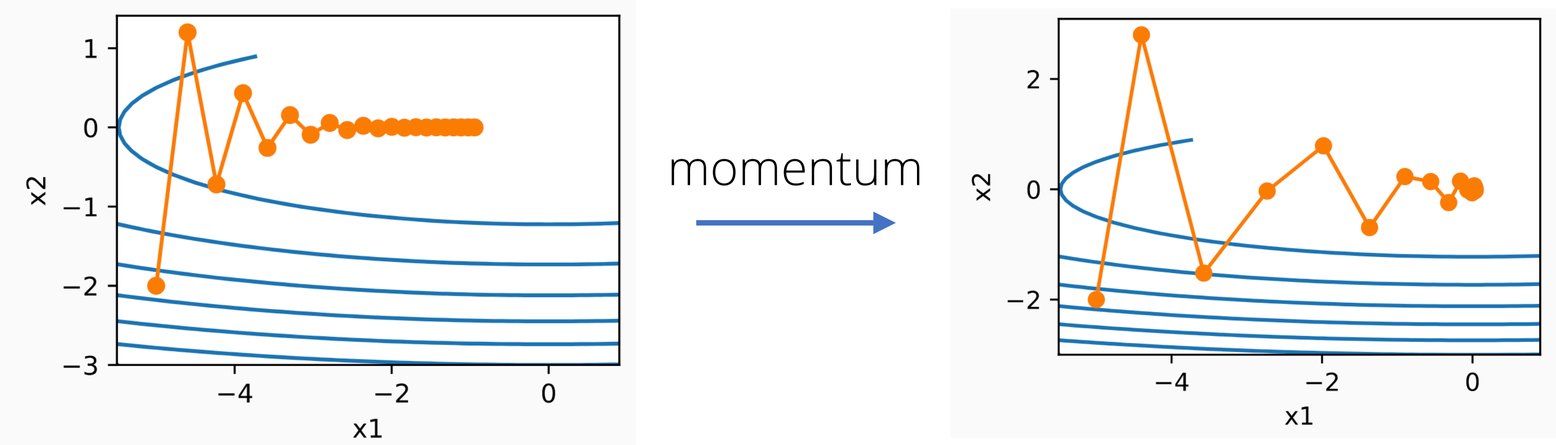
$$\mathbf{g}_{ii} = \partial_{\mathbf{w}} f(\mathbf{x}_i, \mathbf{w}_t)$$

$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \eta_t \mathbf{v}_t$$

- Replaces gradients with a leaky average over past gradients
  - $\mathbf{v}$ is called momentum and $\beta \in (0,1)$
  - accumulates past gradients

$$\mathbf{v}_t = \beta^2 \mathbf{v}_{t-2} + \beta \mathbf{g}_{t-1,t-2} + \mathbf{g}_{t,t-1} = \cdots = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau,t-\tau-1}$$

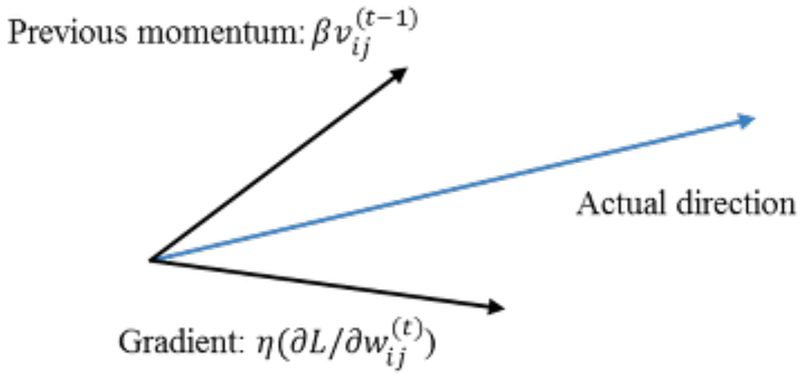
- Effective in cases of ill-conditioned optimization problems



Nesterov Accelerated Gradient (NAG)

- Variants of momentum method

$$\mathbf{v}_t \leftarrow \beta \mathbf{v}_{t-1} + \mathbf{g}_{t,t-1},$$
$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \eta_t \mathbf{v}_t$$



(a) Momentum update

On the importance of initialization and momentum in deep learning, Sutskever et al., ICML 2013

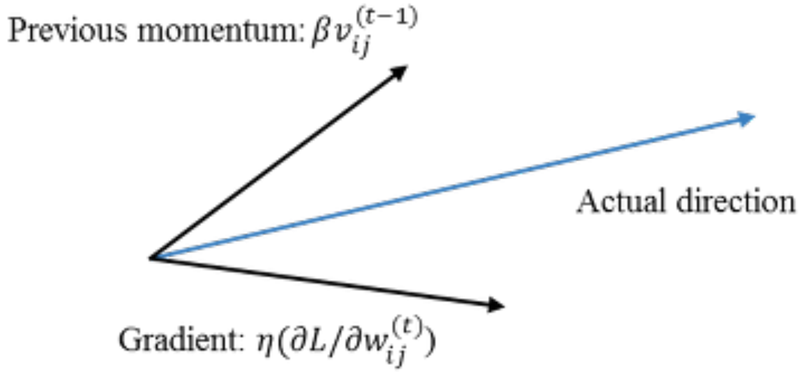
Nesterov Accelerated Gradient (NAG)

- Variants of momentum method
  - more efficient than momentum
  - reduce overshooting

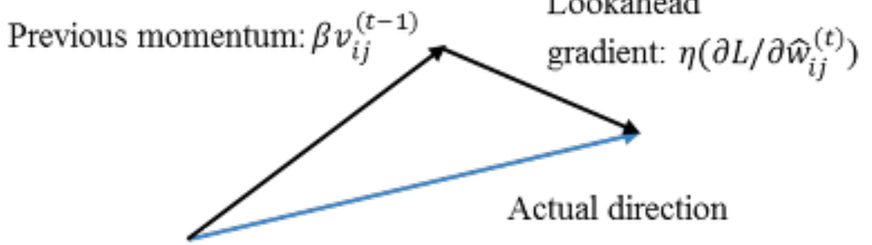
$$\mathbf{v}_t \leftarrow \beta \mathbf{v}_{t-1} - \partial_{\mathbf{w}} f(\mathbf{x}, \mathbf{w}_{t-1} - \beta \mathbf{v}_{t-1})$$

$$\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} - \eta_t \mathbf{v}_t$$

NAG observes the "lookahead" gradient



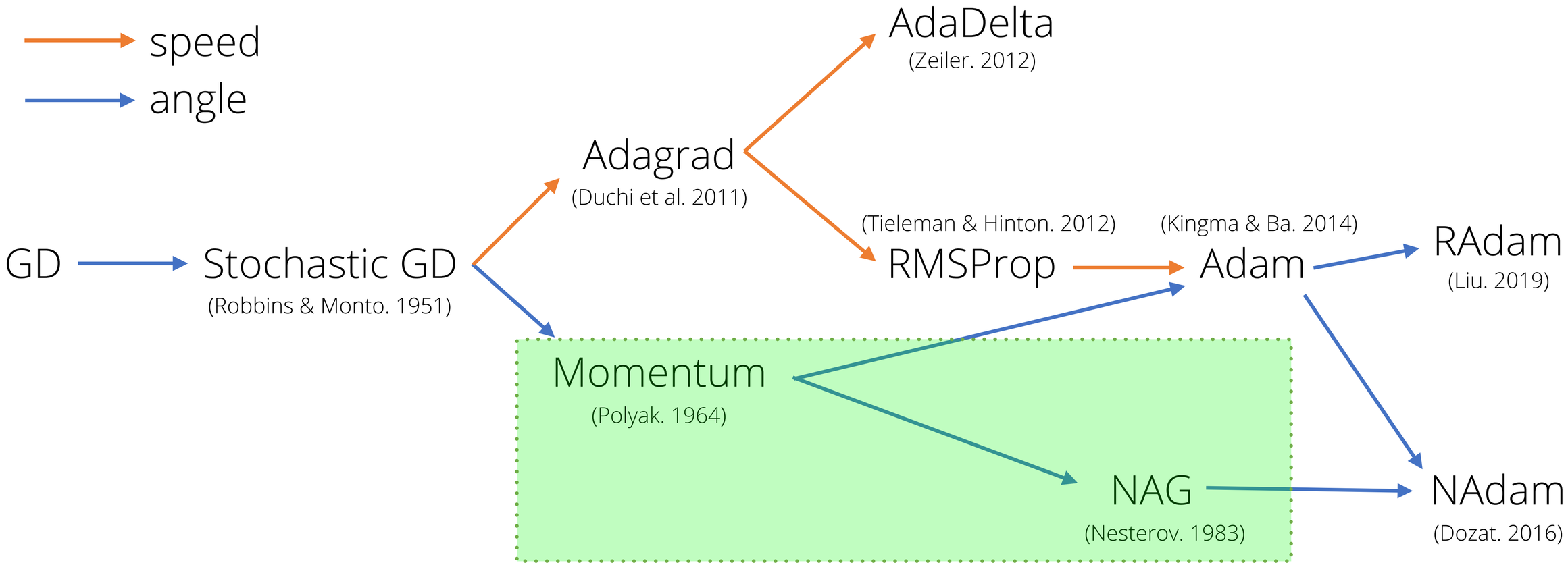
(a) Momentum update



(b) NAG momentum update

On the importance of initialization and momentum in deep learning, Sutskever et al., ICML 2013

Variants of stochastic gradient descent



Adagrad

- Adagrad decreases the learning rate dynamically
  - uses the magnitude of the gradient as a means of adjusting how quickly progress is achieved
  - particularly effective for sparse features
- Too aggressive in reducing learning rates
  - not good for DL

I don't use Adagrad

individual loss for features

variance of gradient

$$\mathbf{g}_t = \partial_{\mathbf{w}} l(y_t, f(\mathbf{x}_t, \mathbf{w})),$$
$$\mathbf{s}_t = \mathbf{s}_{t-1} + \mathbf{g}_t^2,$$
$$\mathbf{w}_t = \mathbf{w}_{t-1} - \frac{\eta}{\sqrt{\mathbf{s}_t + \epsilon}} \cdot \mathbf{g}_t.$$

Adaptive subgradient methods for online learning and stochastic optimization, Duchi et al., JMLR 2011

RMSProp

- Adagrad reduces the learning rate too fast $O(t^{-\frac{1}{2}})$
  - not good for non-convex optimization problems
- RMSProp uses leaky average of variance of gradients
  - adjust the coefficient-wise preconditioner
  - requires learning-rate scheduler

individual loss for features

variance of gradient with leak average

$$\mathbf{g}_t = \partial_{\mathbf{w}} l(y_t, f(\mathbf{x}_t, \mathbf{w})),$$

$$\mathbf{s}_t \leftarrow \gamma \mathbf{s}_{t-1} + (1-\gamma)\mathbf{g}_t^2,$$

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \frac{\eta}{\sqrt{\mathbf{s}_t + \epsilon}} \odot \mathbf{g}_t.$$

I don't use RMSProp too

Lecture 6.5: Divide the gradients by a running average of its recent magnitude, Tieleman & Hinton, 2012

AdaDelta

- Another variant of Adagrad (similar to RMSProp)
  - add $\Delta \mathbf{x}_t$ to store a leaky average of the second moment of the change of parameters
- No learning rate parameters
  - really?

$$\mathbf{s}_t = \rho \mathbf{s}_{t-1} + (1-\rho)\mathbf{g}_t^2,$$

$$\mathbf{g}_t' = \sqrt{\frac{\Delta \mathbf{x}_{t-1} + \epsilon}{\mathbf{s}_t + \epsilon}} \odot \mathbf{g}_t,$$

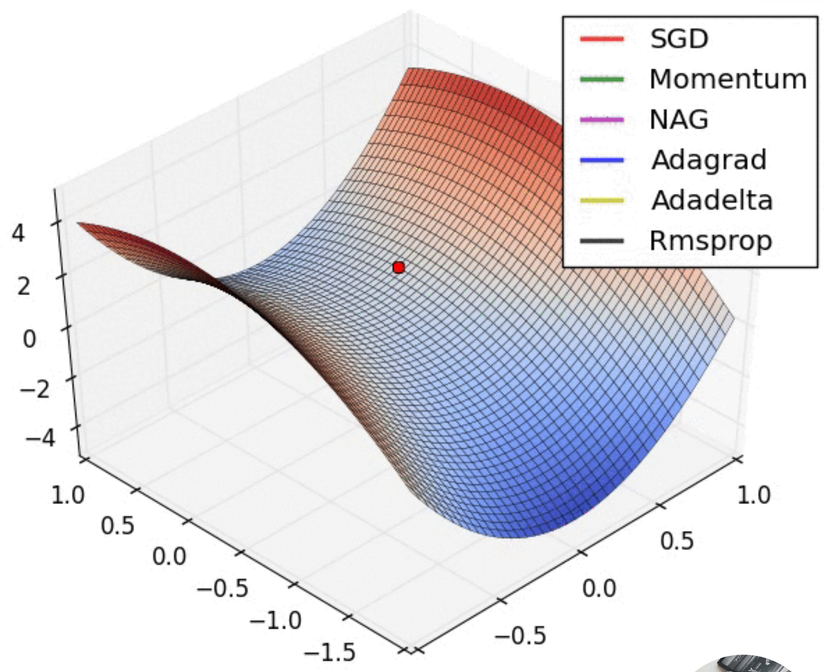
$$\mathbf{x}_t = \mathbf{x}_{t-1} - \mathbf{g}_t',$$

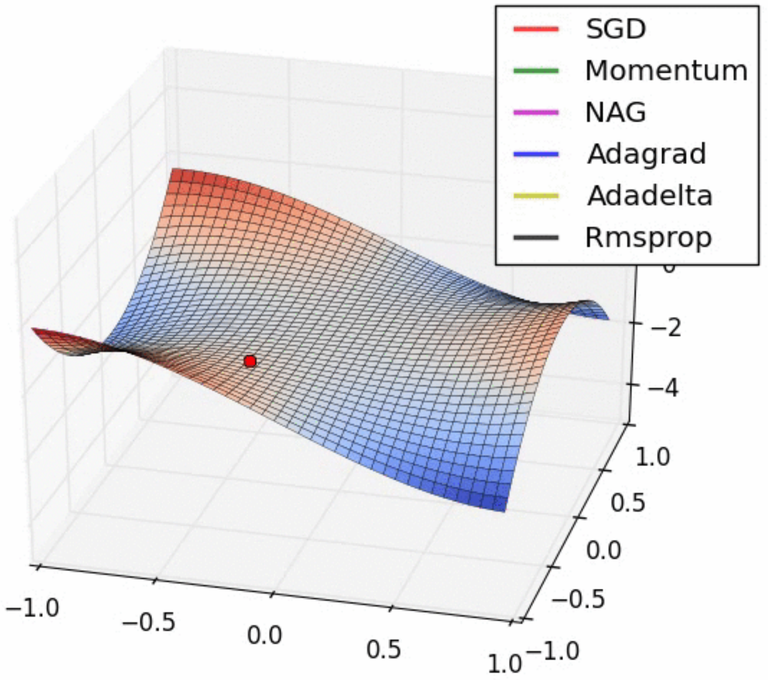
$$\Delta \mathbf{x}_t = \rho \Delta \mathbf{x}_{t-1} + (1-\rho)\mathbf{x}_t^2.$$

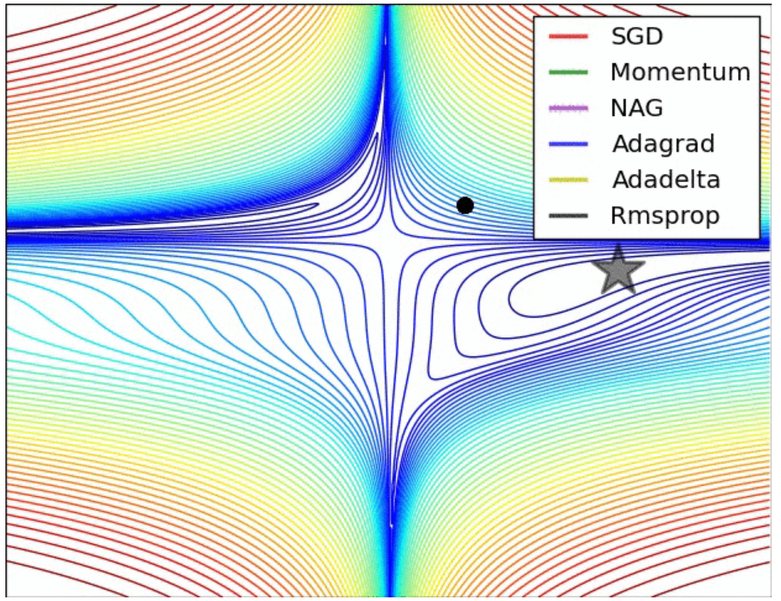
I don't use ...

AdaDelta: an adaptive learning rate method, M Zeiler, 2012

Comparisons btw variants







which one do you prefer? momentum? adaptive?

Simulation: https://ruder.io/optimizing-gradient-descent/

Adam

- **Ada**ptive **M**oment Estimation (Kingma and Ba, 2015, ICLR)
  - Adam combines every techniques into *efficient one*
  - robust & effective

Kingma is also famous for
Variational **A**uto**E**ncoder

Diederik P. Kingma

FOLLOW

Research Scientist, Google Brain
Verified email at google.com - Homepage
Machine Learning Bayesian Inference Deep Learning Neural Networks

TITLE	CITED BY	YEAR
Adam: A Method for Stochastic Optimization DP Kingma, J Ba Proceedings of the 3rd International Conference on Learning Representations …	41072	2014
Auto-Encoding Variational Bayes DP Kingma, M Welling Proceedings of the 2nd International Conference on Learning Representations …	8212	2013

Adam: A Method for Stochastic Optimization, Kingma & Ba, ICLR 2014

Adam

- Adam uses leaky averaging to obtain an estimate of both the momentum and also the second moment of the gradient
  - commonly, $\beta_1 = 0.9$ and $\beta_2 = 0.999$

$$\mathbf{v}_t \leftarrow \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1)\mathbf{g}_t,$$
$$\mathbf{s}_t \leftarrow \beta_2 \mathbf{s}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2.$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_1^t} \text{ and } \hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1 - \beta_2^t}.$$

$$\mathbf{g}_t' = \frac{\eta \hat{\mathbf{v}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon}.$$

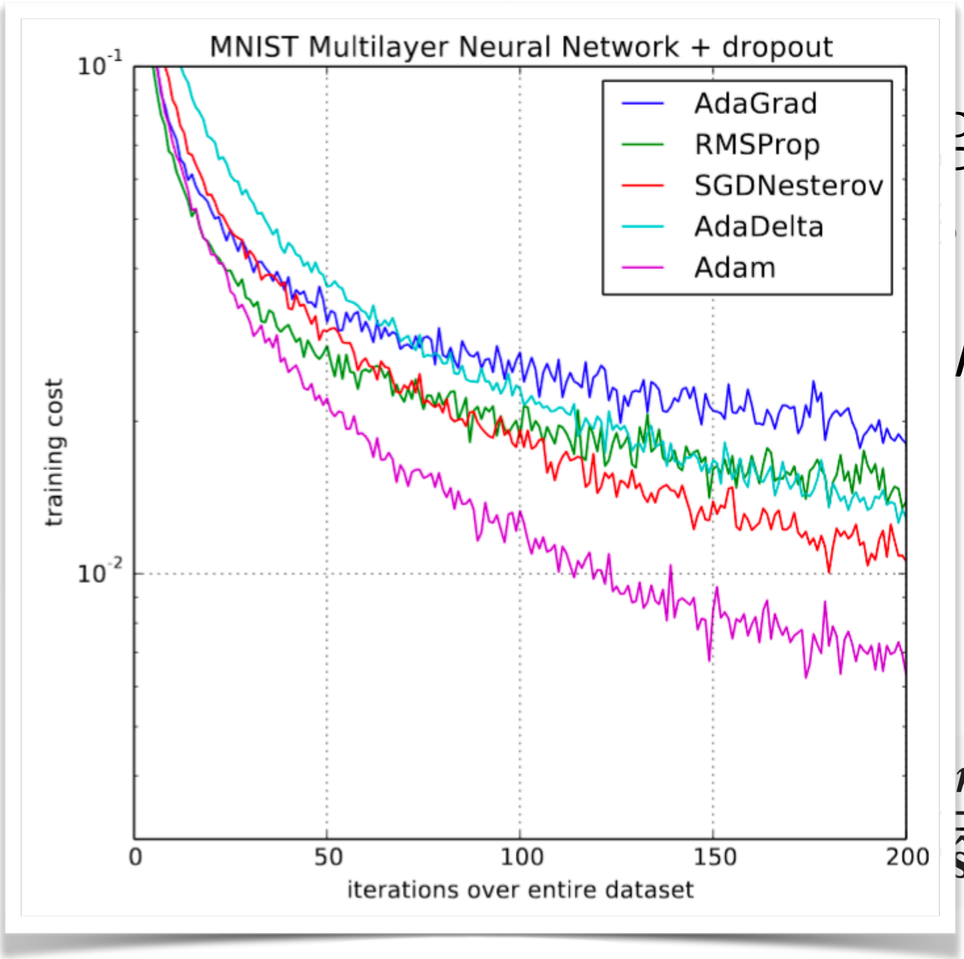
$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{g}_t'.$$

$$\sum_{i=0}^{t} \beta = \frac{1 - \beta^t}{1 - \beta}$$

this is called renormalization

Adam: A Method for Stochastic Optimization, Kingma & Ba, ICLR 2014

Adam



g to obtain an estimate of both the
econd moment of the gradient

$\beta_2 = 0.999$

$$\mathbf{v}_t \leftarrow \beta_1 \mathbf{v}_{t-1} + (1 - \beta_1)\mathbf{g}_t,$$
$$\mathbf{s}_t \leftarrow \beta_2 \mathbf{s}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2.$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_1^t} \text{ and } \hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1 - \beta_2^t}.$$

$$\frac{\eta \hat{\mathbf{v}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon}.$$

$$\sum_{i=0}^{t} \beta = \frac{1 - \beta^t}{1 - \beta}$$

this is called renormalization

Adam: A Method for Stochastic Optimization, Kingma & Ba, ICLR 2014

Adam

- Adam uses leaky averaging to obtain an estimate of both the momentum and also the second moment of the gradient
  - commonly, $\beta_1 = 0.9$ and $\beta_2 = 0.999$
  - Adam can fail to converge even in convex setting!

what?

$$\mathbf{v}_t \leftarrow \beta_1 \mathbf{v}_{t-1} + (1-\beta_1)\mathbf{g}_t,$$
$$\mathbf{s}_t \leftarrow \beta_2 \mathbf{s}_{t-1} + (1-\beta_2)\mathbf{g}_t^2.$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_1^t} \text{ and } \hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1-\beta_2^t}.$$

$$\mathbf{g}_t' = \frac{\eta \hat{\mathbf{v}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon}.$$

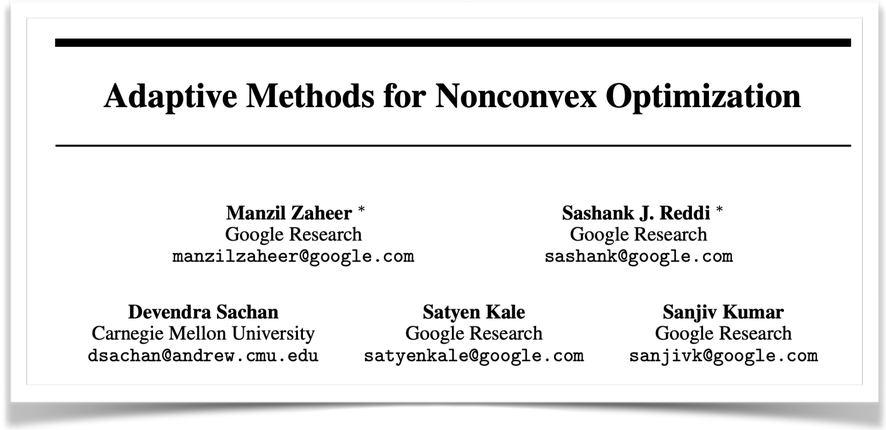
$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{g}_t'.$$

$$\sum_{i=0}^{t} \beta = \frac{1-\beta^t}{1-\beta}$$

Adam: A Method for Stochastic Optimization, Kingma & Ba, ICLR 2014

Yogi (Adam + refinement)

- Yogi uses leaky averaging to obtain an estimate of both the momentum and also the second moment of the gradient



Adaptive Methods for Nonconvex Optimization

Manzil Zaheer *
Google Research
manzilzaheer@google.com

Sashank J. Reddi *
Google Research
sashank@google.com

Devendra Sachan
Carnegie Mellon University
dsachan@andrew.cmu.edu

Satyen Kale
Google Research
satyenkale@google.com

Sanjiv Kumar
Google Research
sanjivk@google.com

$$\mathbf{v}_t \leftarrow \beta_1 \mathbf{v}_{t-1} + (1-\beta_1)\mathbf{g}_t,$$

$$\mathbf{s}_t \leftarrow \mathbf{s}_{t-1} - (1-\beta_2)\operatorname{sign}(\mathbf{s}_{t-1} - \mathbf{g}_t^2)\mathbf{g}_t^2$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_1^t} \text{ and } \hat{\mathbf{s}}_t = \frac{\mathbf{s}_t}{1-\beta_2^t}.$$

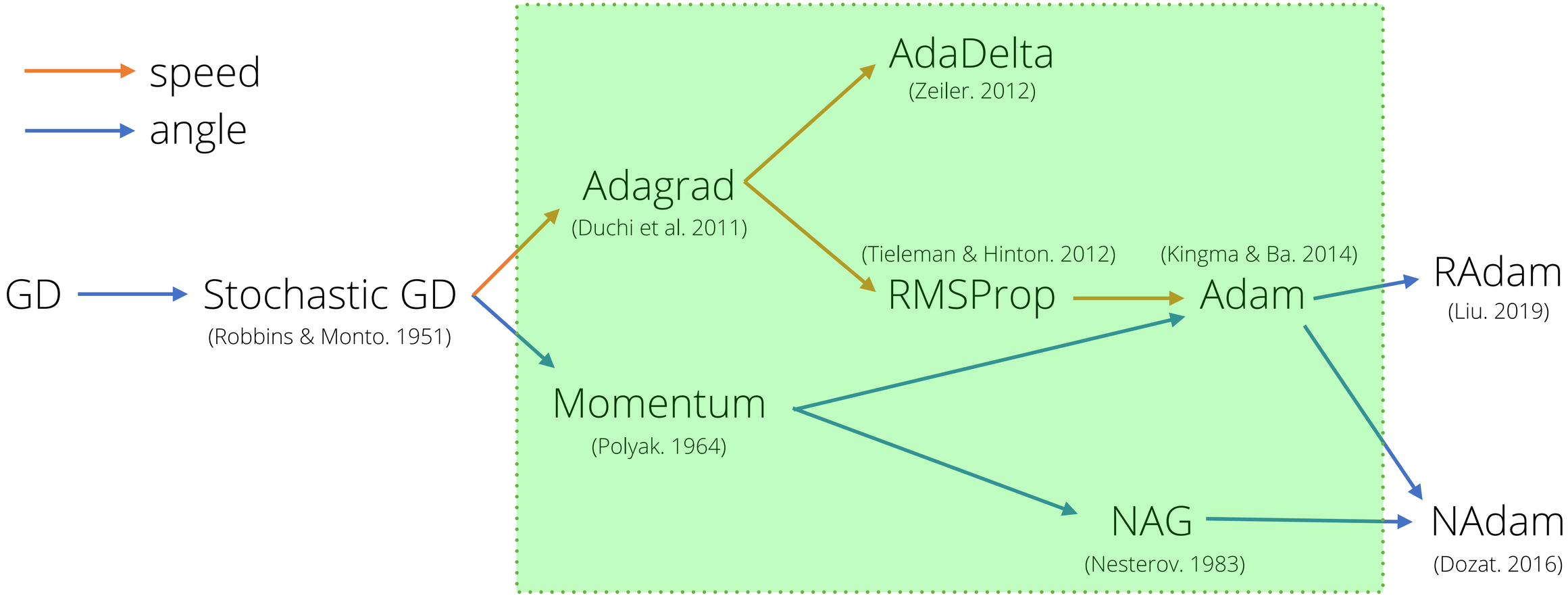
$$\mathbf{g}_t' = \frac{\eta \hat{\mathbf{v}}_t}{\sqrt{\hat{\mathbf{s}}_t} + \epsilon}.$$

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \mathbf{g}_t'.$$

$$\sum_{i=0}^{t} \beta = \frac{1-\beta^t}{1-\beta}$$

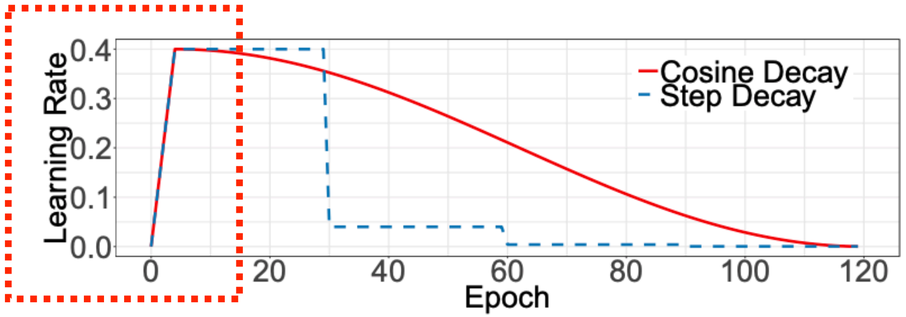
Adaptive methods for nonconvex optimization, Zaheer et al., NeurIPS 2018

Variants of stochastic gradient descent



RAdam

warmup is necessary
for Adam optimizers



(a) Learning Rate Schedule

- Rectified Adam
  - utilizes warmup heuristic for adaptive stochastic optimization

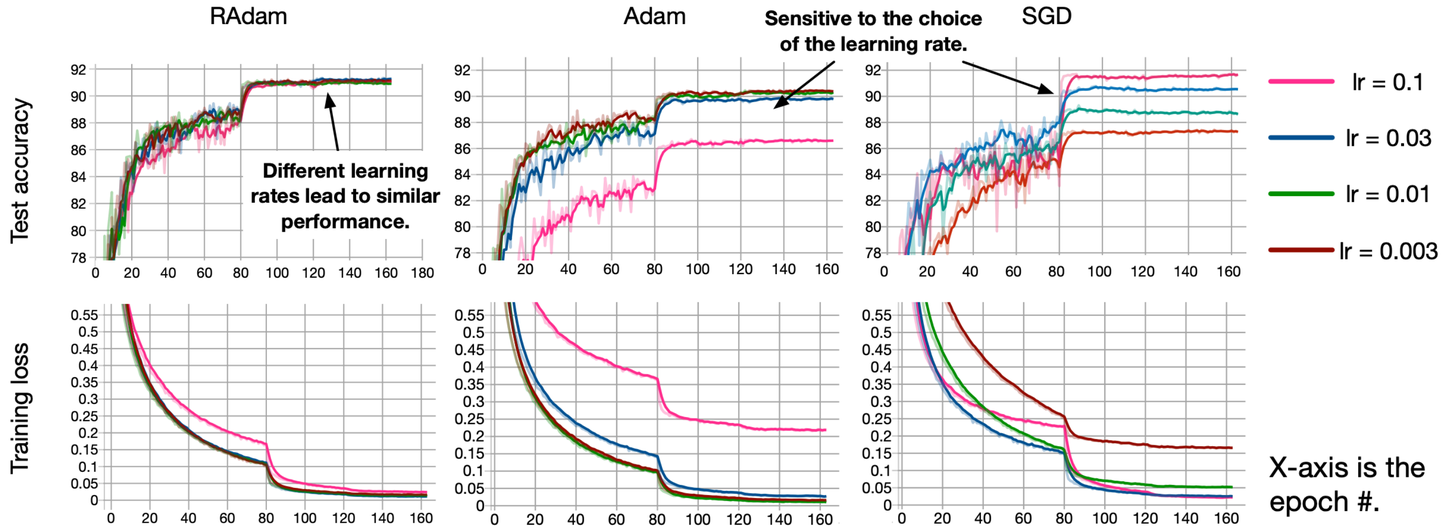


Figure 6: Performance of RAdam, Adam and SGD with different learning rates on CIFAR10.

ON THE VARIANCE OF THE ADAPTIVE LEARNING RATE AND BEYOND

Liyuan Liu *
University of Illinois, Urbana-Champaign
ll2@illinois

Haoming Jiang †
Georgia Tech
jianghm@gatech.edu

Pengcheng He, Weizhu Chen
Microsoft Dynamics 365 AI
{penhe,wzchen}@microsoft.com

Xiaodong Liu, Jianfeng Gao
Microsoft Research
{xiaodl,jfgao}@microsoft.com

Jiawei Han
University of Illinois, Urbana-Champaign
hanj@illinois

now we can escape from
learning rate search!
(Nope)

On the variance of the adaptive learning rate and beyond, Liu et al., ICLR 2020

Assignments

- Revise your proposal in detail (until 4/9) → Determine topics (until 4/23)
  - Explain your Data Science problems in detail
    - What is your goal? Why this is important problem?
    - How to evaluate your model empirically? What is your metric?
  - Describe [what is, how to gather, difficulty of] your data concretely
  - Read the issues in your GitHub and revise your proposal
- Read Textbooks
  - Optimization Algorithms (11.6 - 11.11)
- Implement ***RAdam*** optimizer for MNIST classification (use PyTorch)

Visit https://github.com/LiyuanLucasLiu/RAdam

Q & A