Introduction to Deep Learning (Optimization Algorithms)

WM CS
Zeyi (Tim) Tao
11/01/2019

Topics

SGD

Robbins , Monro : A Stochastic Approximation Method
https://projecteuclid.org › euclid.aoms
by H Robbins - 1951 - Cited by 6678 - Related articles
Let M(x) denote the expected value at level x of the response to a certain experiment. M(x) is assumed to be a monotone function of x but is unknown to the ...

SGDM

On the momentum term in gradient descent learning algorithms
https://www.sciencedirect.com › science › article › pii
by N Qian - 1999 - Cited by 855 - Related articles
The behavior of **gradient descent** near a local minimum is equivalent to a set of coupled and damped harmonic oscillators. Within a reasonable parameter range, the **momentum term** can improve the speed of convergence for most eigen components in the system by bringing them closer to critical damping.

Adaptive LR

AdaGrad

[PDF] Adaptive Subgradient Methods for Online Learning and ...
www.jmlr.org › papers › volume12
by J Duchi - 2011 - Cited by 5323 - Related articles
Before introducing our adaptive gradient algorithm, which we term **ADAGRAD**, we establish notation. Vectors and scalars are lower case italic letters, such as x ...

RMSprop

Neural Networks for Machine Learning

Lecture 6a
Overview of mini-batch gradient descent

Geoffrey Hinton

AdaDelta

ADADELTA: An Adaptive Learning Rate Method
https://arxiv.org › cs
by MD Zeiler - 2012 - Cited by 3495 - Related articles
Dec 22, 2012 - Abstract: We present a novel per-dimension learning rate method for gradient descent called **ADADELTA**. The method dynamically adapts over ...

Adam

Adam: A Method for Stochastic Optimization
https://arxiv.org › cs
by DP Kingma - 2014 - Cited by 33005 - Related articles
Some connections to related **algorithms**, on which **Adam** was inspired, are ... We also analyze the theoretical convergence properties of the **algorithm** and ...
Cite as: arXiv:1412.6980

ERM problem Statement

- **Given a training set of input-output pairs** $(X_1, d_1), (X_2, d_2), \cdots, (X_N, d_N)$

- **Minimize the following function (w.r.t W)**

Total Loss

Sum over all training instances

Output of net in response to input

Desired output (label)

$$Loss(W) = \frac{1}{N} \sum_{i=1}^{N} div(f(W_i; X), d_i) + \gamma(w)$$

Measurement functions

Regularizer

- **This is problem of function minimization**

Choice of div() functions
Mean Square Loss VS CrossEntropy

Prediction Mode: $f_w(x_i) = wx_i + b$

Square Loss: $$Loss_w = \frac{1}{N}\sum_{i}^{N}(d_i - f_x(x_i))^2$$

Suppose: $d_i = w_i x_i + b + n$ **where** $n \sim Nomal(0,1)$

$$E[d_i] = E[w_i x_i + b + n] = w_i x_i + b$$

$$Var[d_i] = Var[w_i x_i + b + n] = 1$$

Choice of div() functions
Mean Square Loss VS CrossEntropy

The probability of observing a single (x_i, d_i)

$$p(d_i \mid x_i) = e^{-\frac{(d_i - (wx_i + b))^2}{2}}$$

The Max Likelihood:

$$Like(d, x) = \prod_{i=1}^{N} e^{-\frac{(d_i - (wx_i + b))^2}{2}}$$

Choice of div() functions

Mean Square Loss VS CrossEntropy

$$Like(d,x) = \prod_{i=1}^{N} e^{-\frac{(d_i - (wx_i + b))^2}{2}}$$

$$l(d,x) = -\frac{1}{2}\sum_{i=1}^{N}(d_i - (wx_i + b))^2 \quad \text{(MAX)}$$

$$MSE(d,x) = \frac{1}{2}\sum_{i=1}^{N}(d_i - f_w(x_i))^2 \quad \text{(MIN)}$$

Choice of div() functions
Mean Square Loss VS CrossEntropy

Prediction Mode: $f_w(x_i) = \sigma(wx_i + b)$ **assume** $\sigma()$ **is softmax liked**

$$p(d_i = 1 | x_i) = f_w(x_i) \qquad p(d_i = 0 | x_i) = 1 - f_w(x_i)$$

$$p(d_i | x_i) = [f_w(x_i)]^{d_i}[1 - f_w(x_i)]^{(1-d_i)}$$

$$Like(d, x) = \prod_{i=1}^{N} [f_w(x_i)]^{d_i}[1 - f_w(x_i)]^{(1-d_i)}$$

Choice of div() functions
Mean Square Loss VS CrossEntropy

$$Like(d,x) = \prod_{i=1}^{N} [f_w(x_i)]^{d_i}[1 - f_w(x_i)]^{(1-d_i)}$$

$$like(d,x) = -\sum_{i=1}^{N} d_i \log(f_w(x_i)) + (1 - d_i)\log(1 - f_w(x_i))$$

(binary)

$$like(d,x) = -\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij} \log(f_w(x_i)_j) + (1 - d_{ij})\log(1 - f_w(x_i)_j)$$

(multi-class)

Choice of div() functions
Mean Square Loss VS CrossEntropy

$$f_w(x_i) = \sigma(wx_i + b) = \hat{d}_i$$

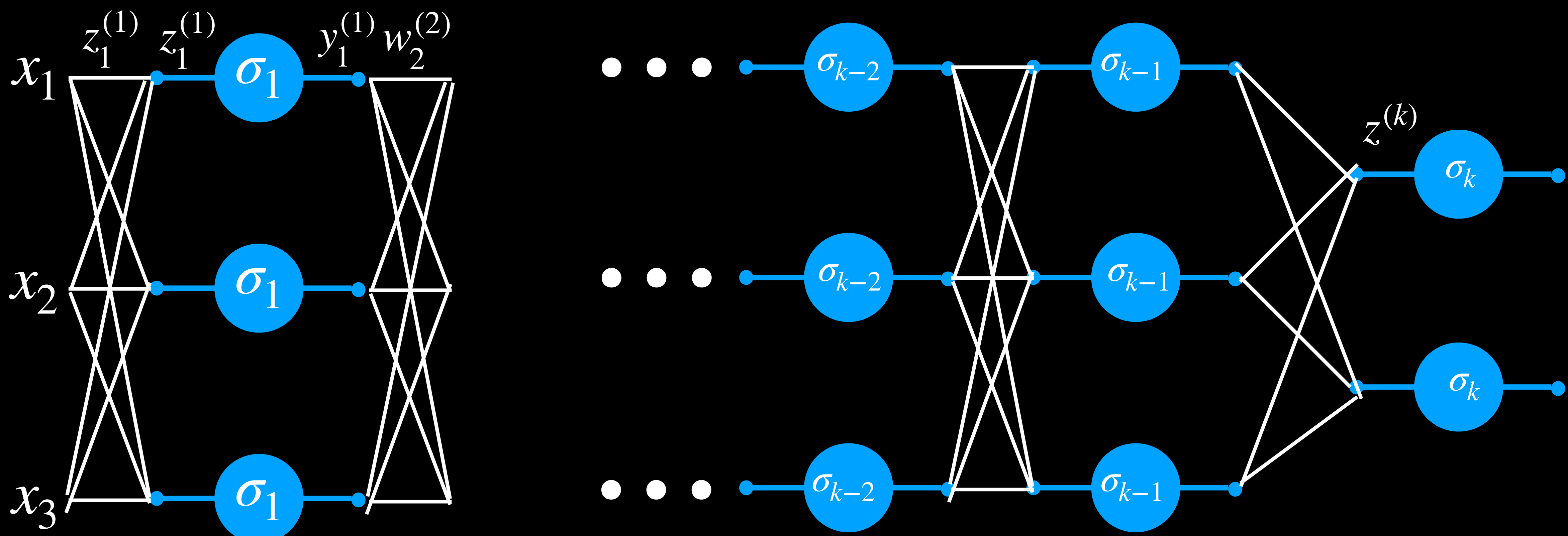
$$Loss = \frac{1}{2}\sum_{i=1}^{N}(\hat{d}_i - d_i)^2$$

$$Loss = -\sum_{i=1}^{N} d_i \log(\hat{d}_i) + (1 - d_i)\log(1 - \hat{d}_i)$$

$$\frac{dLoss}{dw} = \sum_{i=1}^{N}(\hat{d}_i - d_i)\sigma^{'}(wx_i + b)x_i$$

$$\frac{dLoss}{dw} = \sum_{i=1}^{N} x_i(\sigma(wx_i + b) - d_i)$$

The vanishing gradient problem

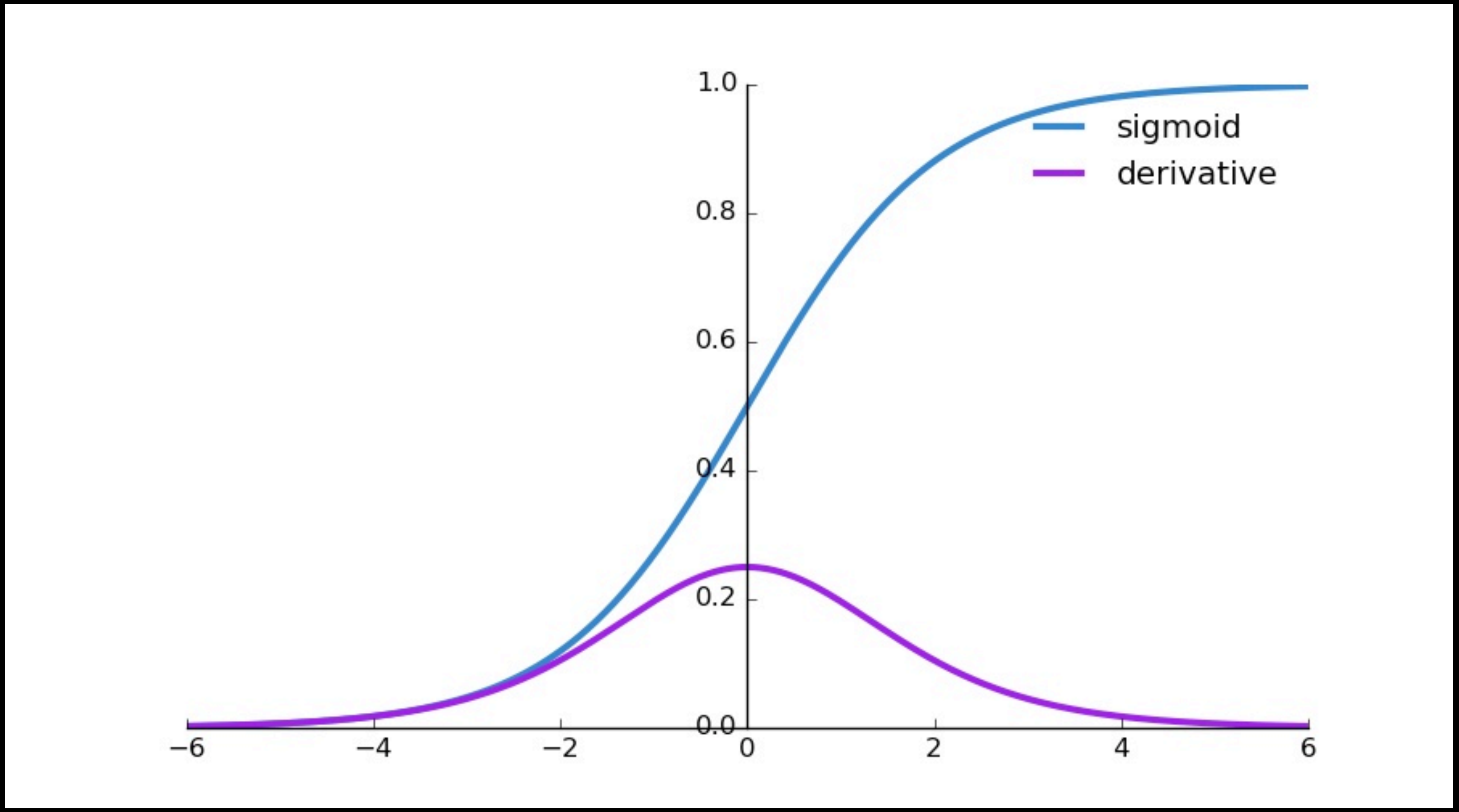


$$Y = \sigma_k(W_k\sigma_{k-1}(\cdots(\sigma_2(W_2\sigma_1(W_1x + b_1)\cdots))\cdots) + b_k)$$

Choice of div() functions
Mean Square Loss VS CrossEntropy

$$\frac{dLoss}{d_w} = \sum_{i=1}^{N} (\hat{d}_i - d_i)\sigma^{'}(wx_i + b)x_i$$

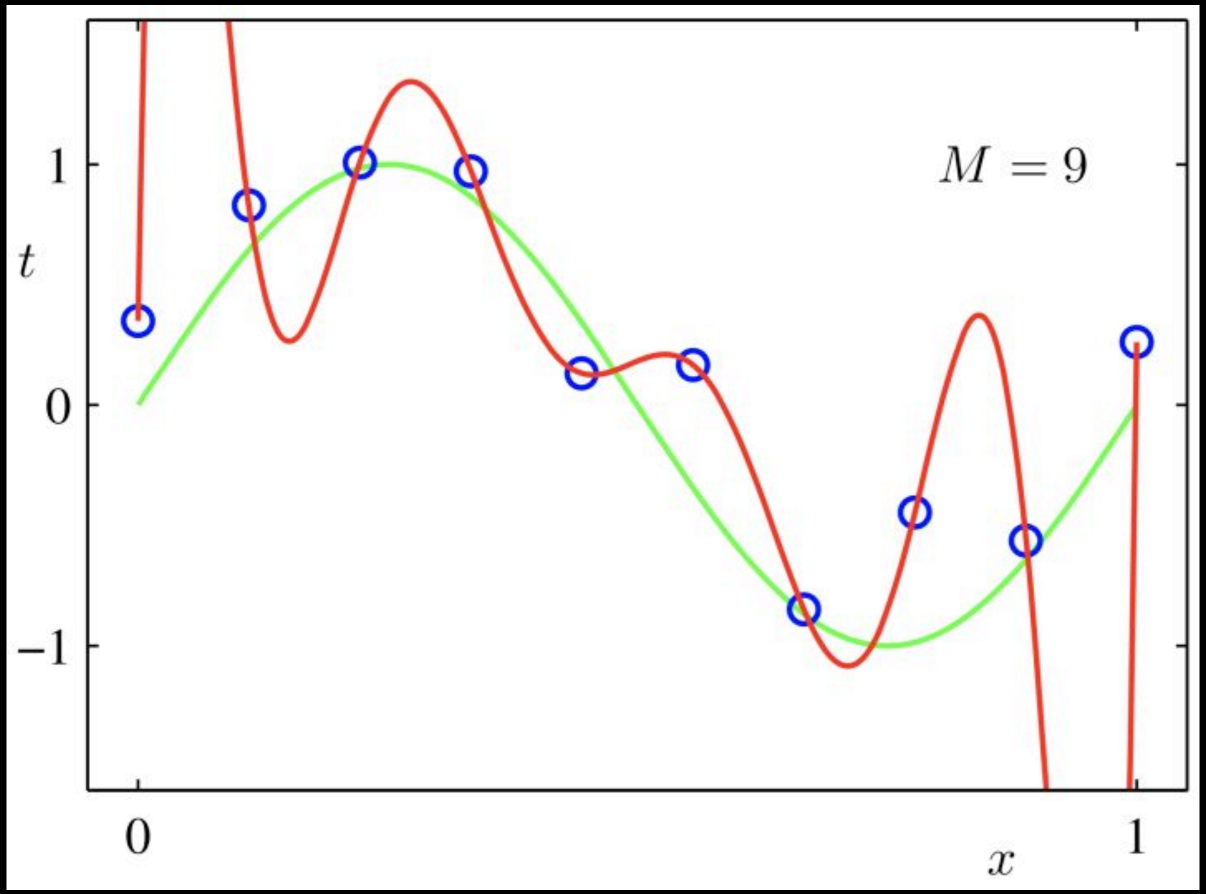
$$\frac{dLoss}{d_w} = \sum_{i=1}^{N} x_i(\sigma(wx_i + b) - d_i)$$



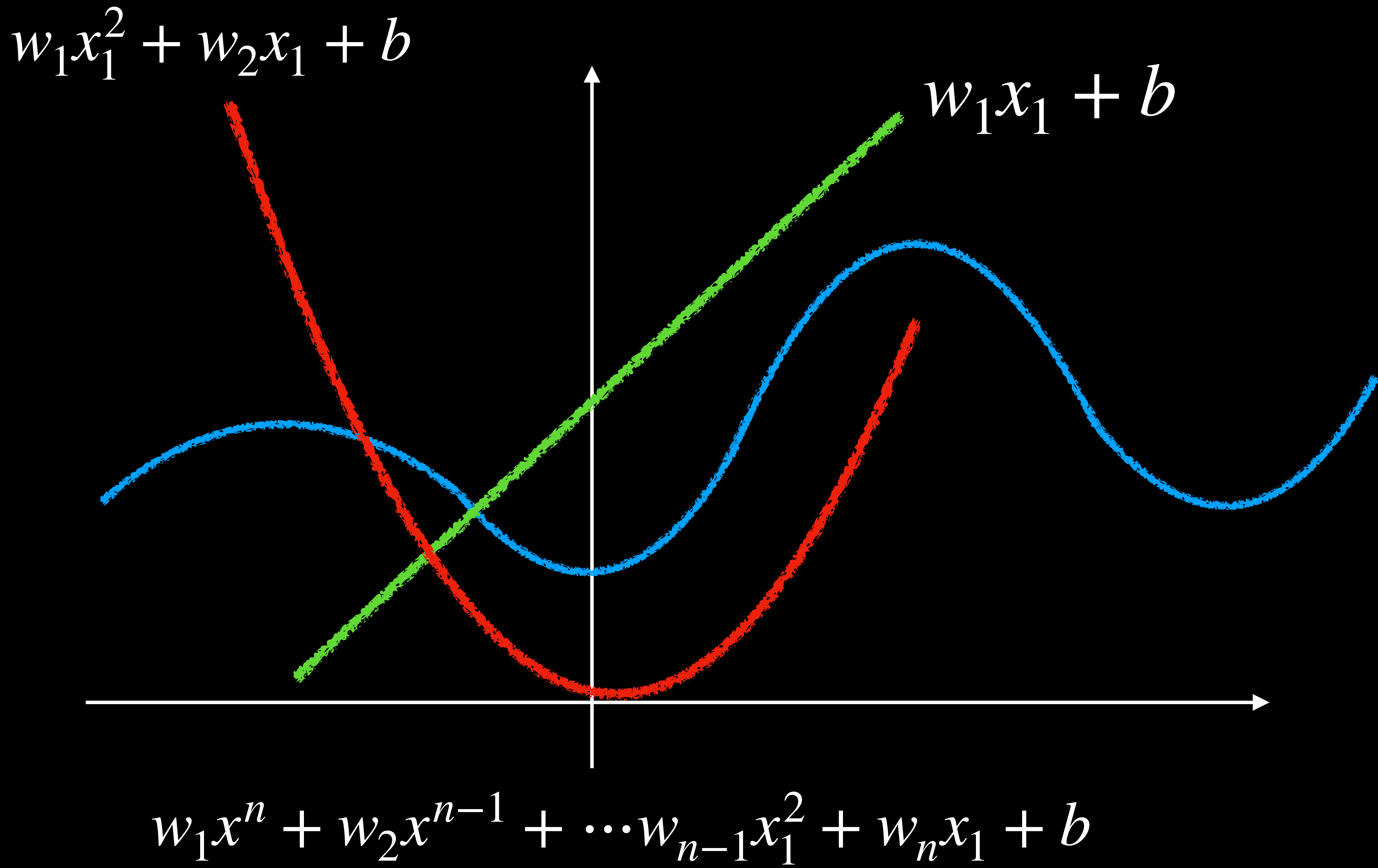
Regularization

$$Loss(W) = \frac{1}{N}\sum_{i=1}^{N} div(f(W_i; X), d_i) + \lambda\gamma(w)$$

$$\gamma(w) = \frac{1}{2}||w||^2 = \frac{1}{2}||w - 0||^2$$



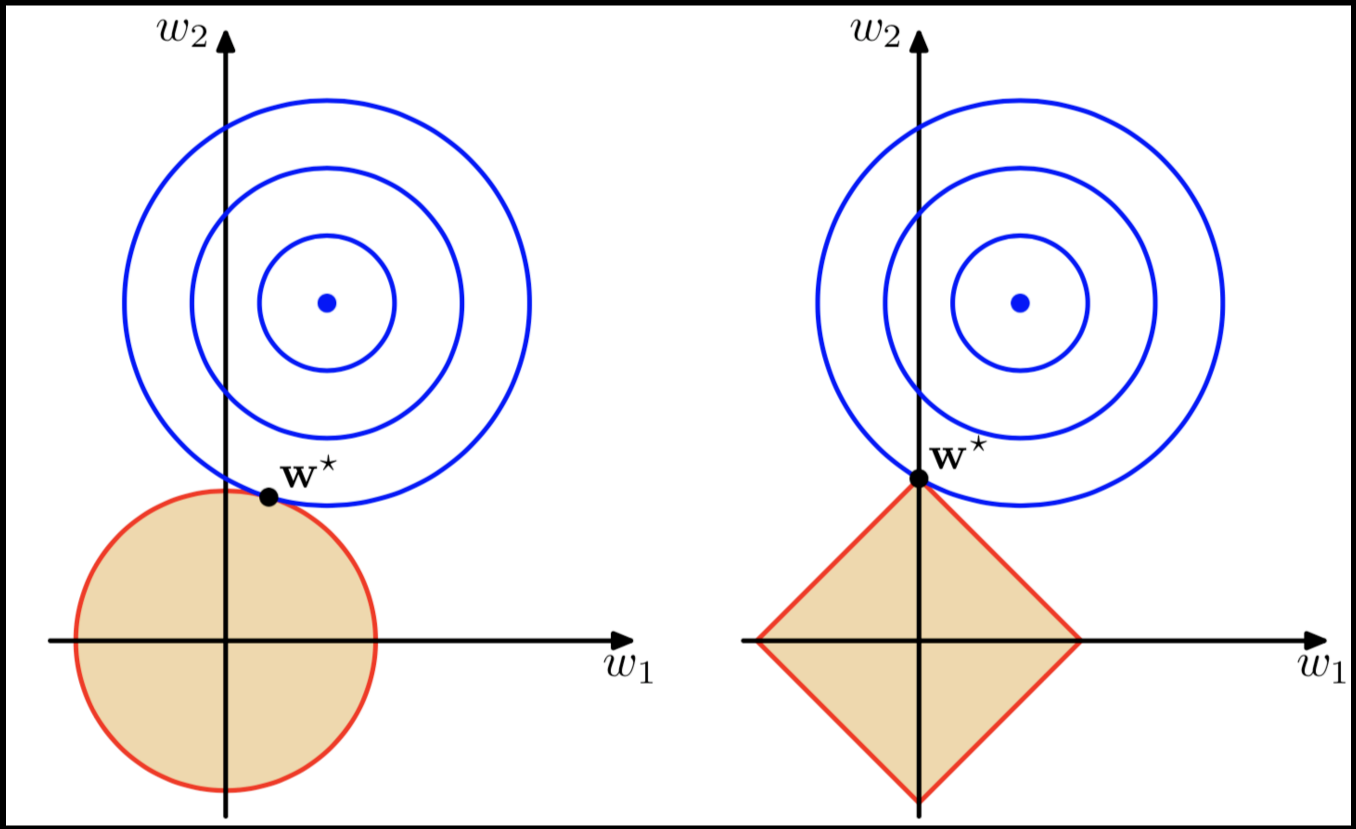
Regularization



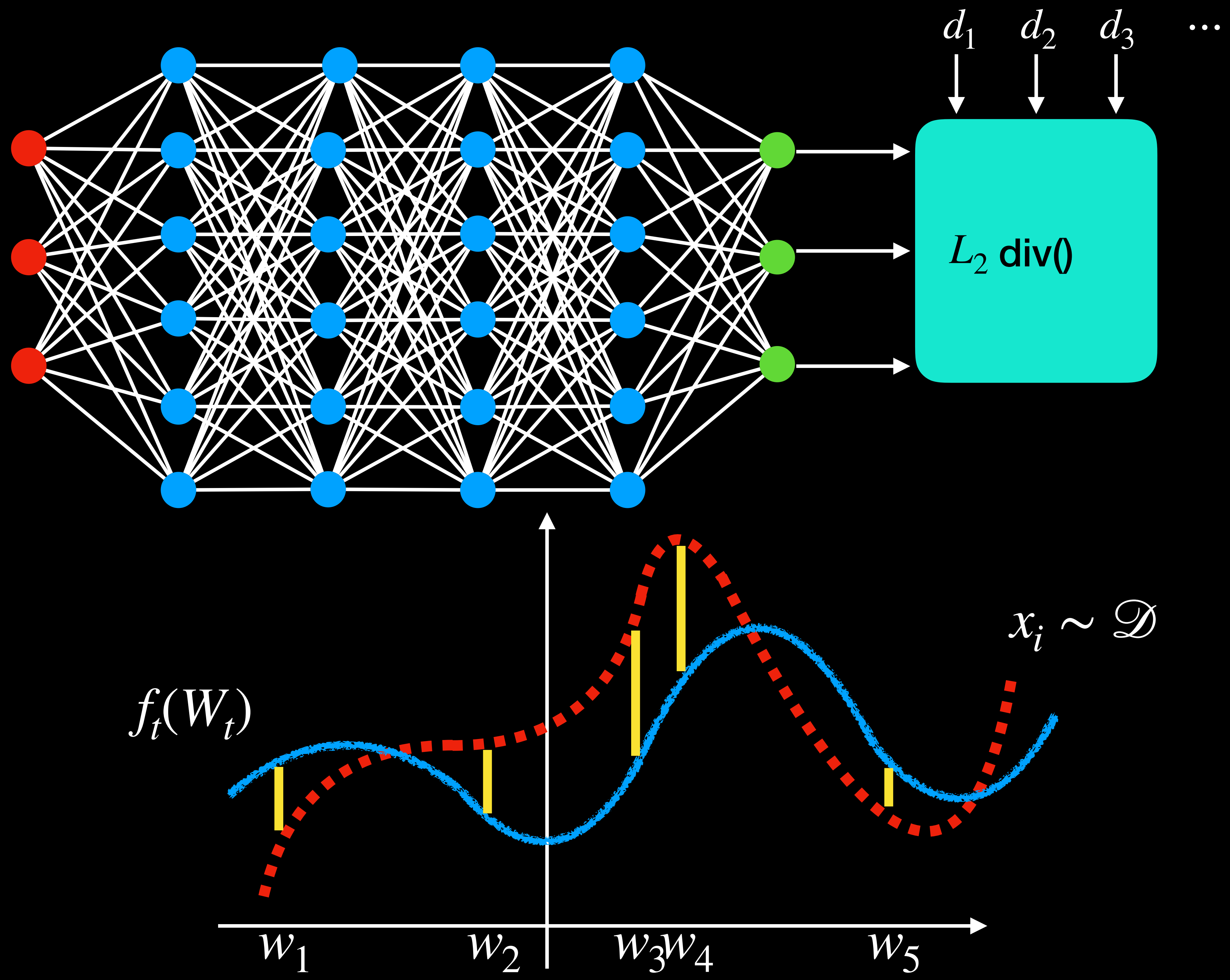
Regularization

$$min_w \frac{1}{N} \sum_{i=1}^{N} div(f(W_i; X), d_i) + \lambda \gamma(w)$$

$$min_w \frac{1}{N} \sum_{i=1}^{N} div(f(W_i; X), d_i) + 100w_n^2 + 100w_{n-1}^2 + \cdots$$



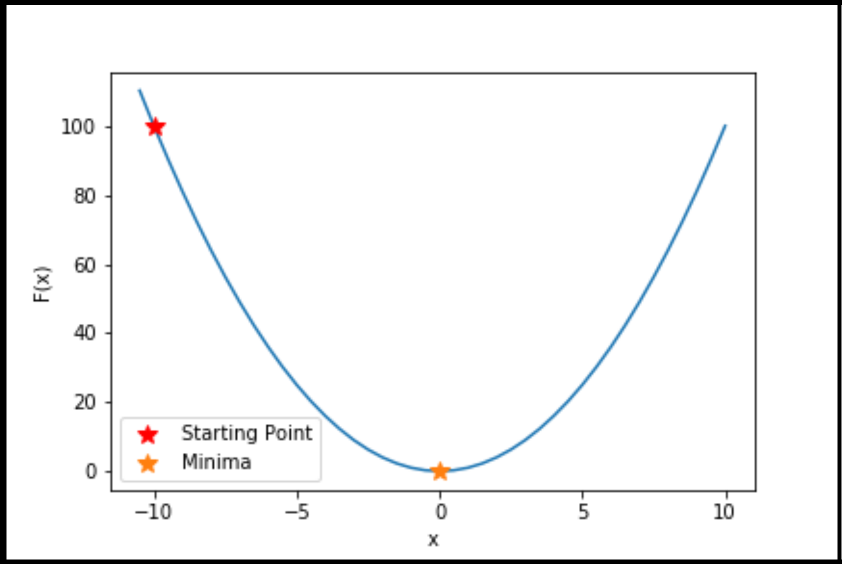
Deep neuron network

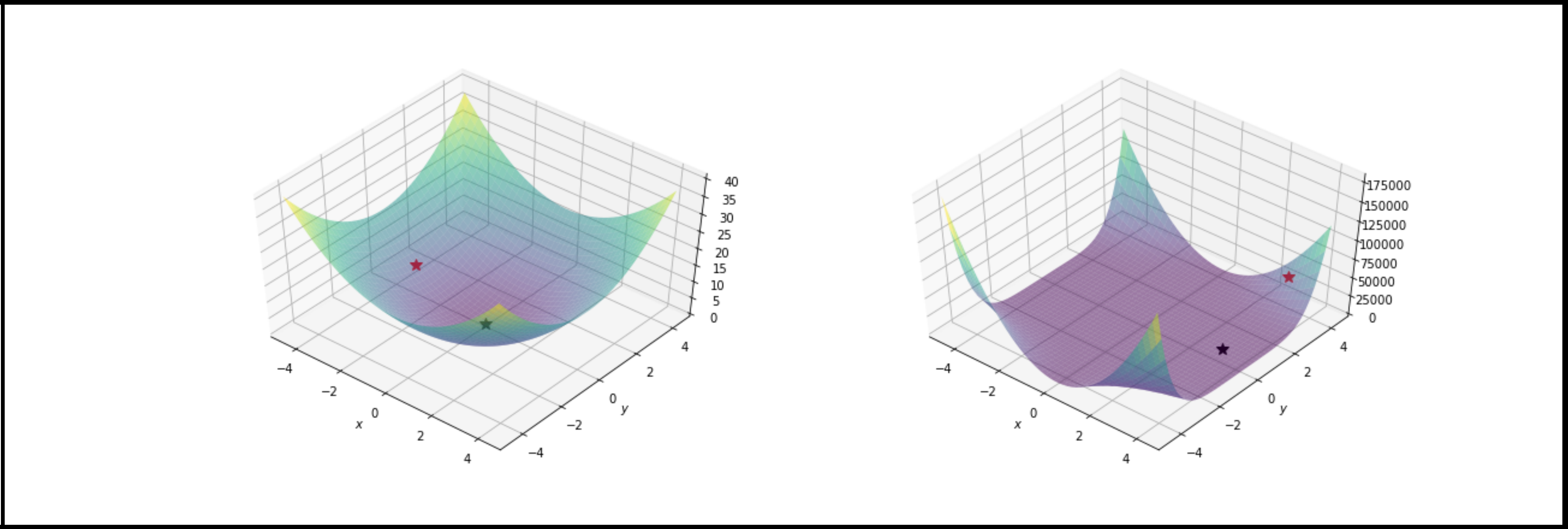


Unconstrained First-order Optimization

- For real-valued output vectors, the (scaled) L_2 divergence is popular

$$Div(\hat{d}, d) = \frac{1}{2}||\hat{d} - d||^2 = \frac{1}{2}\sum_i (\hat{d} - d_i)^2$$

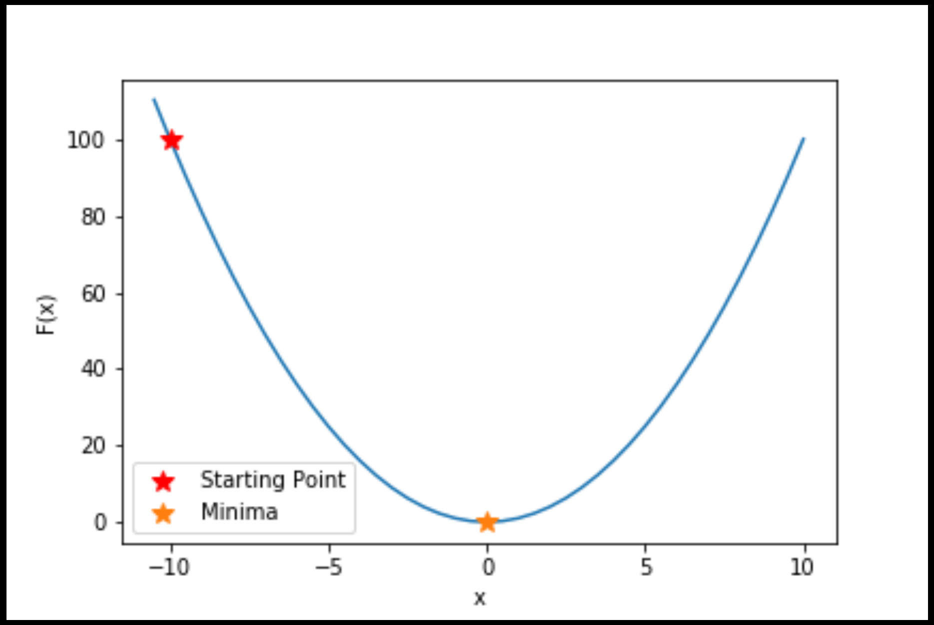




Unconstrained First-order Optimization

ERM: convex

Local optimal = Global optimal



First Order: Gradient Descent

$$f(w + \triangle w) = f(w) + f^{'}(w) \triangle w$$ **Linear approximation**

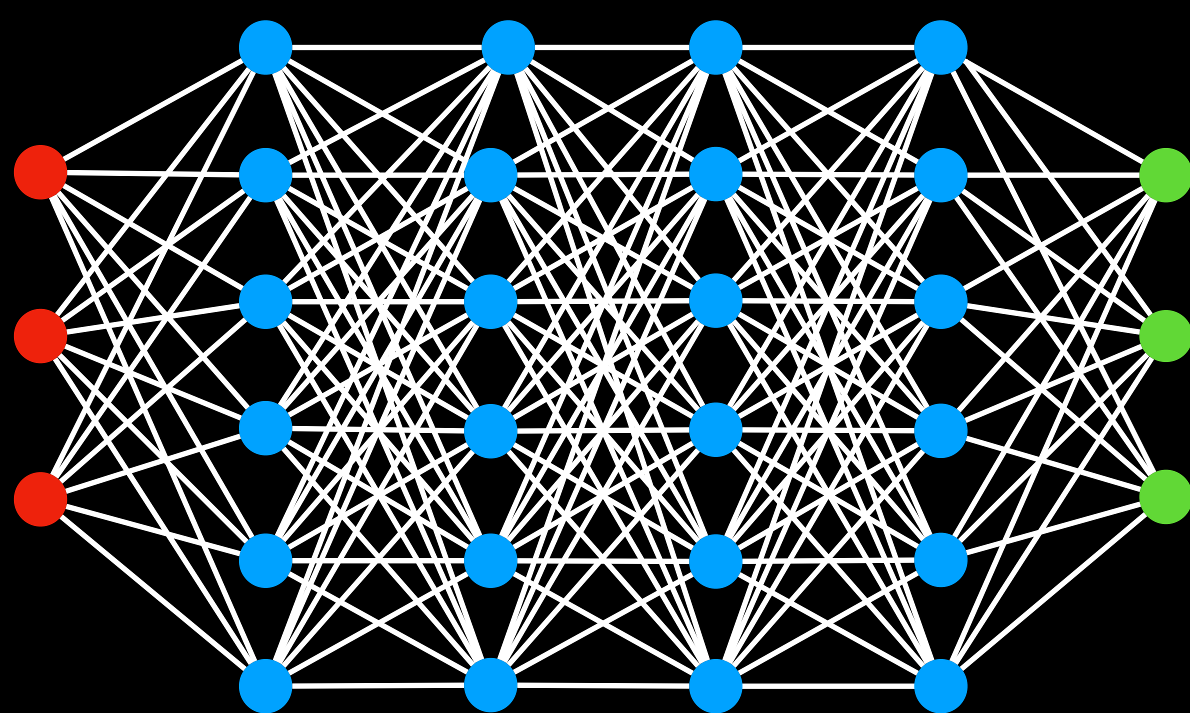
Second Order: Newton Method

$$f(w + \triangle w) = f(w) + f^{'}(w) \triangle w + \frac{1}{2} f^{''}(w)(\triangle x)^2$$

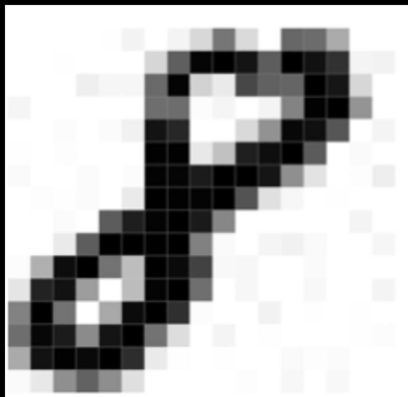
Gradient descent variants

Batch gradient descent

$$w_t = w_{t-1} - \eta^k \nabla_w f(W; X = \{(x_1, d_1), (x_2, d_2), \cdots\})$$



MNIST: 60k x 28 x28



=


```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 12, 0, 11, 39, 137, 37, 0, 152, 147, 84, 0, 0, 0, 0,
0, 1, 0, 0, 0, 41, 160, 250, 255, 235, 162, 255, 238, 206, 11, 13, 0, 0, 0, 0, 16, 9, 9, 150, 251, 45, 21, 184, 159, 154, 2
55, 233, 40, 0, 0, 10, 0, 0, 0, 0, 0, 145, 146, 3, 10, 0, 11, 124, 253, 255, 107, 0, 0, 0, 0, 3, 0, 4, 15, 236, 216, 0, 0,
38, 109, 247, 240, 169, 0, 11, 0, 1, 0, 2, 0, 0, 0, 253, 253, 23, 62, 224, 241, 255, 164, 0, 5, 0, 0, 6, 0, 0, 4, 0, 3, 252
, 250, 228, 255, 255, 234, 112, 28, 0, 2, 17, 0, 0, 2, 1, 4, 0, 21, 255, 253, 251, 255, 172, 31, 8, 0, 1, 0, 0, 0, 0, 0, 4,
 0, 163, 225, 251, 255, 229, 120, 0, 0, 0, 0, 0, 11, 0, 0, 0, 0, 21, 162, 255, 255, 254, 255, 126, 6, 0, 10, 14, 6, 0, 0, 9
, 0, 3, 79, 242, 255, 141, 66, 255, 245, 189, 7, 8, 0, 0, 5, 0, 0, 0, 0, 26, 221, 237, 98, 0, 67, 251, 255, 144, 0, 8, 0, 0
, 7, 0, 0, 11, 0, 125, 255, 141, 0, 87, 244, 255, 208, 3, 0, 0, 13, 0, 1, 0, 1, 0, 0, 145, 248, 228, 116, 235, 255, 141, 34
, 0, 11, 0, 1, 0, 0, 0, 1, 3, 0, 85, 237, 253, 246, 255, 210, 21, 1, 0, 1, 0, 0, 6, 2, 4, 0, 0, 0, 6, 23, 112, 157, 114, 32
, 0, 0, 0, 0, 2, 0, 8, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Gradient descent variants

Batch gradient descent

$$w_t = w_{t-1} - \eta^k \nabla_w f(W; X = \{(x_1, d_1), (x_2, d_2), \cdots\})$$

Stochastic gradient descent

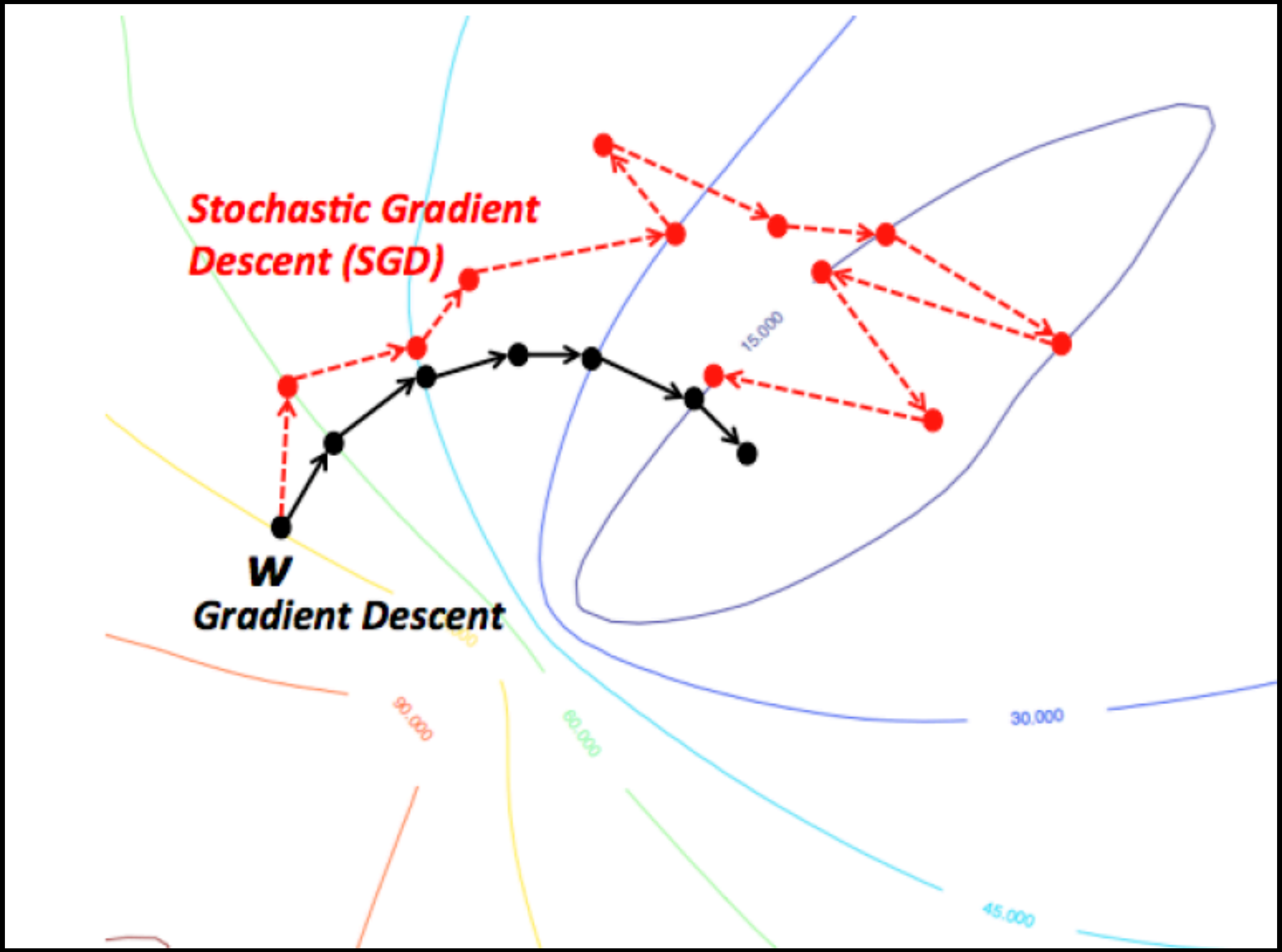
$$w_t = w_{t-1} - \eta^k \nabla_w f(W; (x_i, d_i))$$

Mini-batch gradient descent

$$w_t = w_{t-1} - \eta^k \nabla_w f(W; X = \{(x_i, d_i) \mid i = 1,2,\cdots b\})$$

Shuffle

Batch Gradient descent VS SGD



Gradient descent variants

Batch gradient descent

- Pro
- Less oscillations and noise
- Vectorization
- Stable

- Con
- Local optimal
- Not memory friendly

Stochastic gradient descent

- Pro
- Computationally fast
- Fast convergence(large dataset)
- Fit into memory

- Con
- Noisy
- Longer training time
- No vectorization

Mini-batch gradient descent

Mini-batch gradient descent (Vanilla SGD)

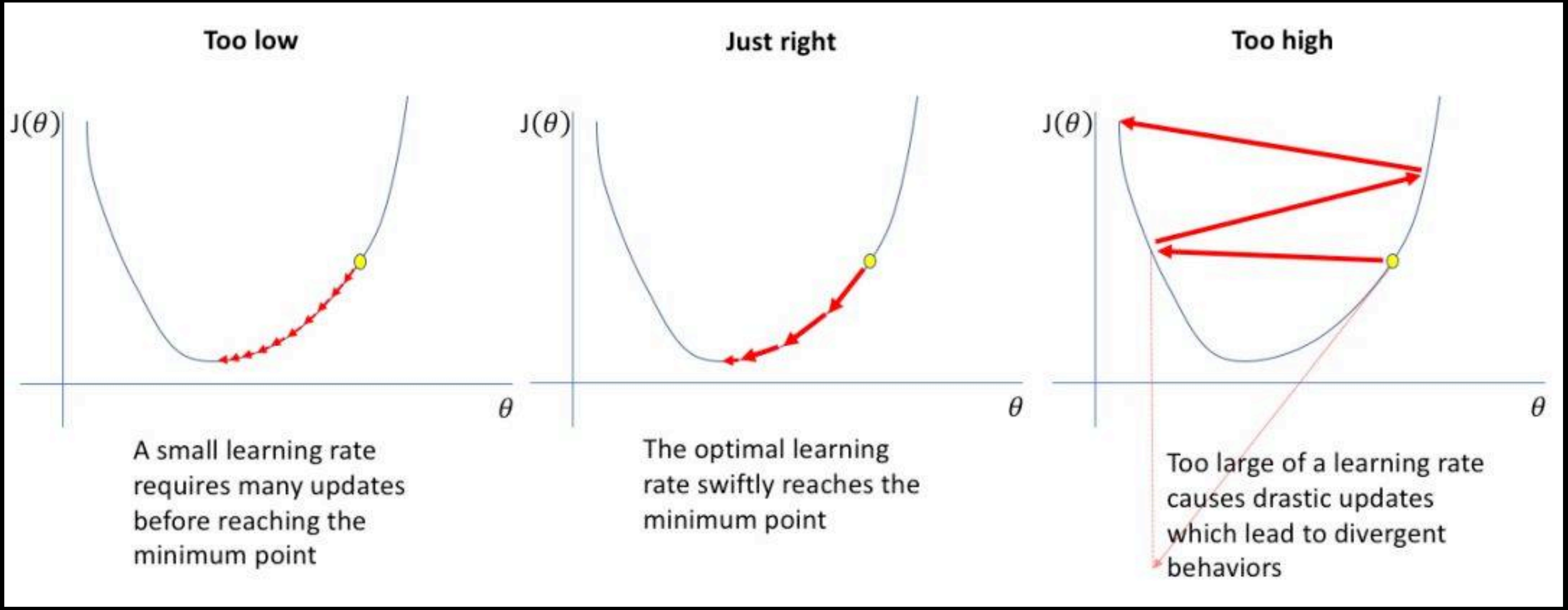
$$w_t = w_{t-1} - \eta^k \nabla_w f(W; (x_i, d_i))$$

$$w_t = w_{t-1} - \eta_k g_t$$

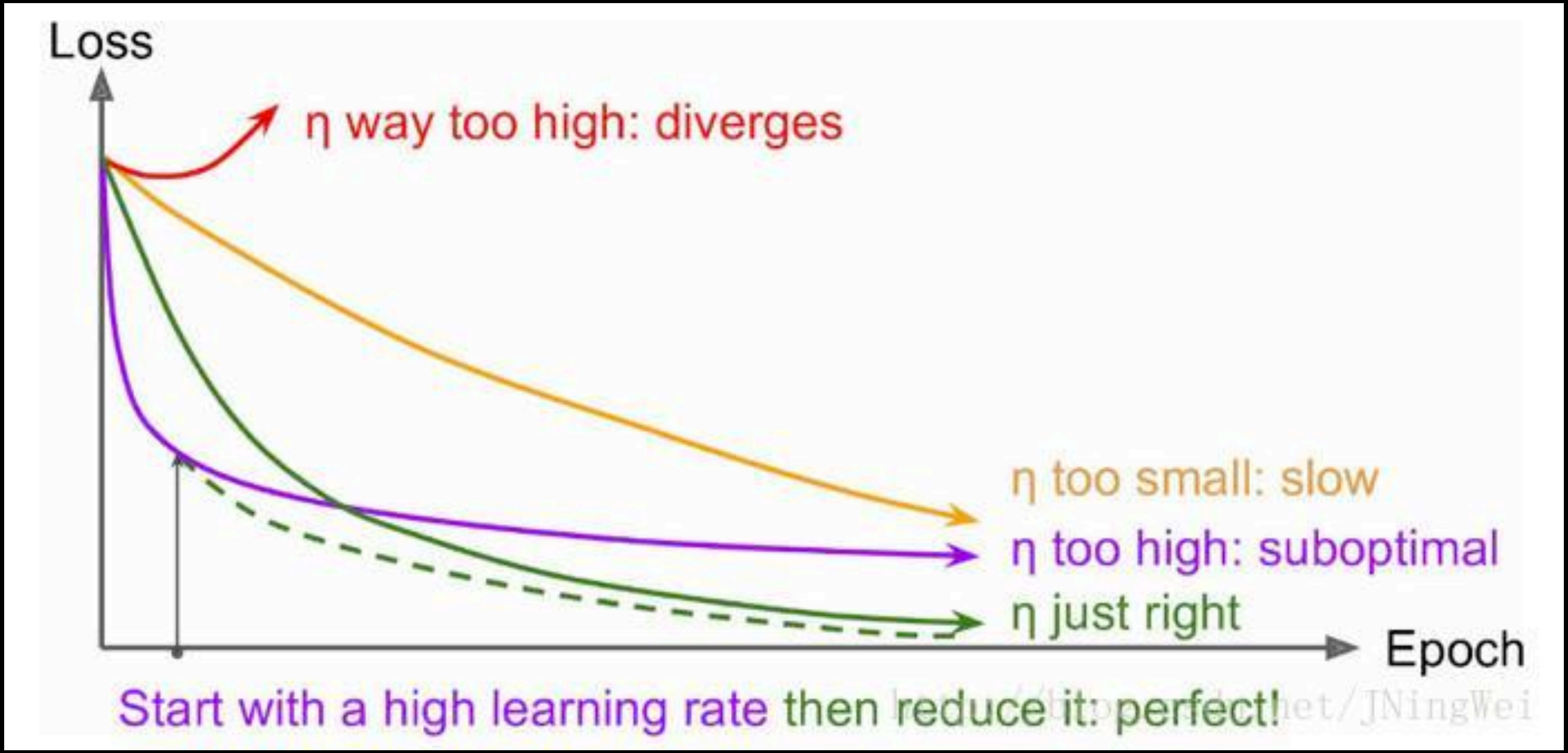
gradient
(direction)

Learning Rate
(Step Size)

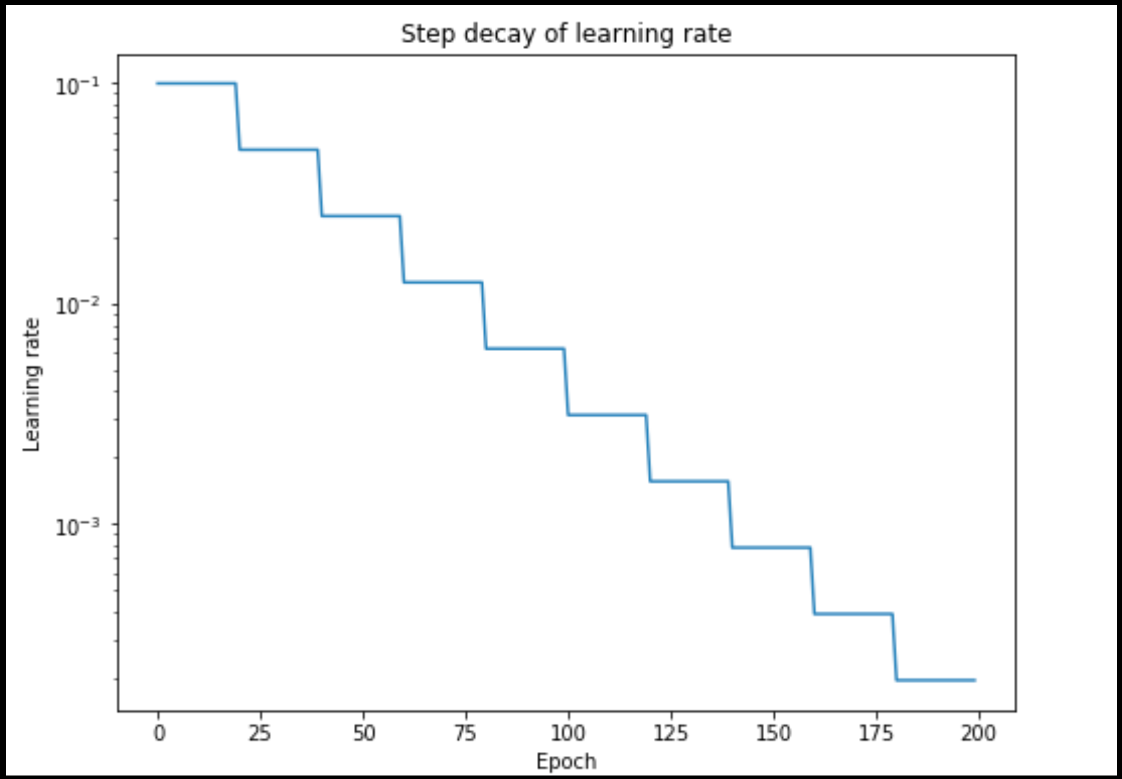
Learning Rate I



Learning Rate II



Learning Rate III: Strategies



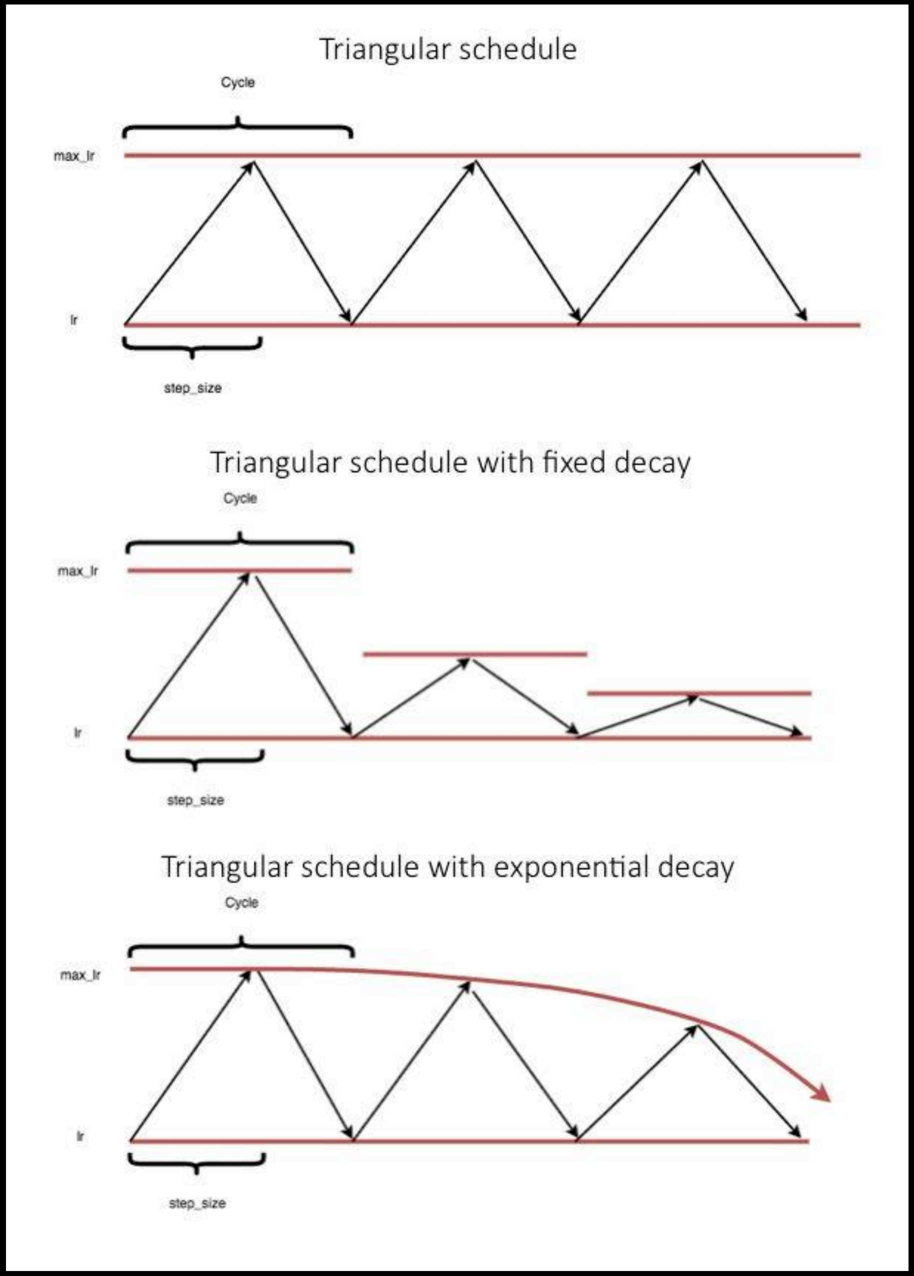
Step decay

CLASS `torch.optim.lr_scheduler.StepLR(optimizer, step_size, gamma=0.1, last_epoch=-1)` [SOURCE]

Sets the learning rate of each parameter group to the initial lr decayed by gamma every step_size epochs. When last_epoch=-1, sets initial lr as lr.

Parameters

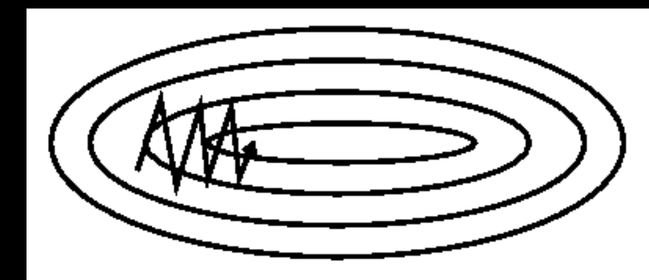
- **optimizer** (*Optimizer*) – Wrapped optimizer.
- **step_size** (*int*) – Period of learning rate decay.
- **gamma** (*float*) – Multiplicative factor of learning rate decay. Default: 0.1.
- **last_epoch** (*int*) – The index of last epoch. Default: -1.

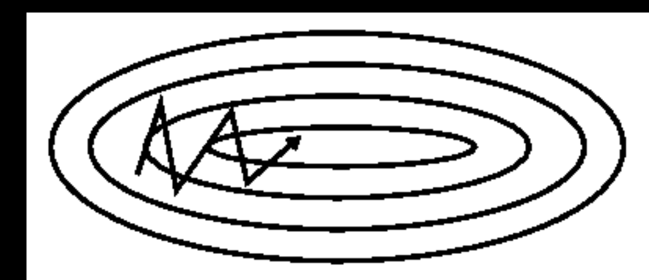


Cyclical Learning Rate

SGD to SGD with Momentum(SGDM)

On the momentum term in gradient descent learning algorithms
https://www.sciencedirect.com › science › article › pii
by N Qian - 1999 - Cited by 855 - Related articles
The behavior of **gradient descent** near a local minimum is equivalent to a set of coupled and damped harmonic oscillators. Within a reasonable parameter range, the **momentum term** can improve the speed of convergence for most eigen components in the system by bringing them closer to critical damping.





$$v_t = \beta v_{t-1} + \eta g_t$$
$$w_t = w_{t-1} - v_t$$

Pro:
- **Accelerate SGD**
- **Overcome local minima**
- **Dampen oscillations (begin)**

Con:
- **oscillations (end)**

The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions.

SGD to Adaptive Subgradient Methods (AdaGrad)

[PDF] Adaptive Subgradient Methods for Online Learning and ...
www.jmlr.org › papers › volume12 ▾
by J Duchi - 2011 - Cited by 5323 - Related articles
Before introducing our adaptive gradient algorithm, which we term **ADAGRAD**, we establish notation. Vectors and scalars are lower case italic letters, such as x ...

$$g_{t,i} = \nabla_{w_{t,i}} f$$

$$G_{t,i} = \sum_{i=1}^{t} g_{t,i}^2$$

$$w_{t+1,i} = w_{t,i} - \frac{\eta_0}{\sqrt{G_{t,i} + \epsilon}} \odot g_{t,i}$$

Pro:

- **Suit for dealing with sparse data**

Con:

- **Monotonically decreasing LR**

Performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features.

AdaGrad to AdaDelta

ADADELTA: An Adaptive Learning Rate Method
https://arxiv.org › cs
by MD Zeiler - 2012 - Cited by 3495 - Related articles
Dec 22, 2012 - Abstract: We present a novel per-dimension learning rate method for gradient descent called **ADADELTA**. The method dynamically adapts over ...

$$g_{t,i} = \nabla_{w_{t,i}} f$$

$$v_{t,i} = \beta v_{t-1,i} + (1-\beta) g_{t,i}^2$$

$$w_{t+1,i} = w_{t,i} - \frac{\eta_0}{\sqrt{v_{t,i} + \epsilon}} \odot g_{t,i}$$

Pro:

- **Suit for dealing with sparse data**
- **Using a sliding window**

Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size

AdaGrad to Adam

Adam: A Method for Stochastic Optimization
https://arxiv.org › cs ▾
by DP Kingma - 2014 - Cited by 33005 - Related articles
Some connections to related **algorithms**, on which **Adam** was inspired, are ... We also analyze the theoretical convergence properties of the **algorithm** and ...
Cite as: arXiv:1412.6980

$$g_{t,i} = \nabla_{w_{t,i}} f$$

$$m_{t,i} = \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}$$

$$v_{t,i} = \beta_2 v_{t-1,i} + (1 - \beta_2) g_{t,i}^2$$

$$w_{t+1,i} = w_{t,i} - \frac{\eta_0}{\sqrt{v_{t,i} + \epsilon}} \odot m_{t,i}$$

Pro:
- **Super fast (current)**
- **Exponential Moving exponential (EMA)**

Con:
- **Suboptimal solution**
- **EMA diminishes the changes of gradients**

Adam to ?

[PDF] on the convergence of adam and beyond - OpenReview
https://openreview.net › pdf ▾
by SJ Reddi - 2018 - Cited by 425 - Related articles
ON THE CONVERGENCE OF **ADAM AND BEYOND**. Sashank J. Reddi, Satyen Kale & Sanjiv Kumar. Google New York. New York, NY 10011, USA. {sashank ...
You've visited this page many times. Last visit: 10/21/19

The Marginal Value of Adaptive Gradient Methods in Machine ...
https://papers.nips.cc › paper › 7003-the-marginal-value-of-adaptive-gradi... ▾
by AC Wilson - 2017 - Cited by 288 - Related articles
The Marginal Value of Adaptive Gradient Methods in Machine Learning. Part of: Advances in Neural Information Processing Systems 30 (NIPS 2017).
You've visited this page 3 times. Last visit: 11/9/19

Improving Generalization Performance by Switching from Adam to SGD

Nitish Shirish Keskar, Richard Socher

(Submitted on 20 Dec 2017)

Decoupled Weight Decay Regularization

Ilya Loshchilov, Frank Hutter

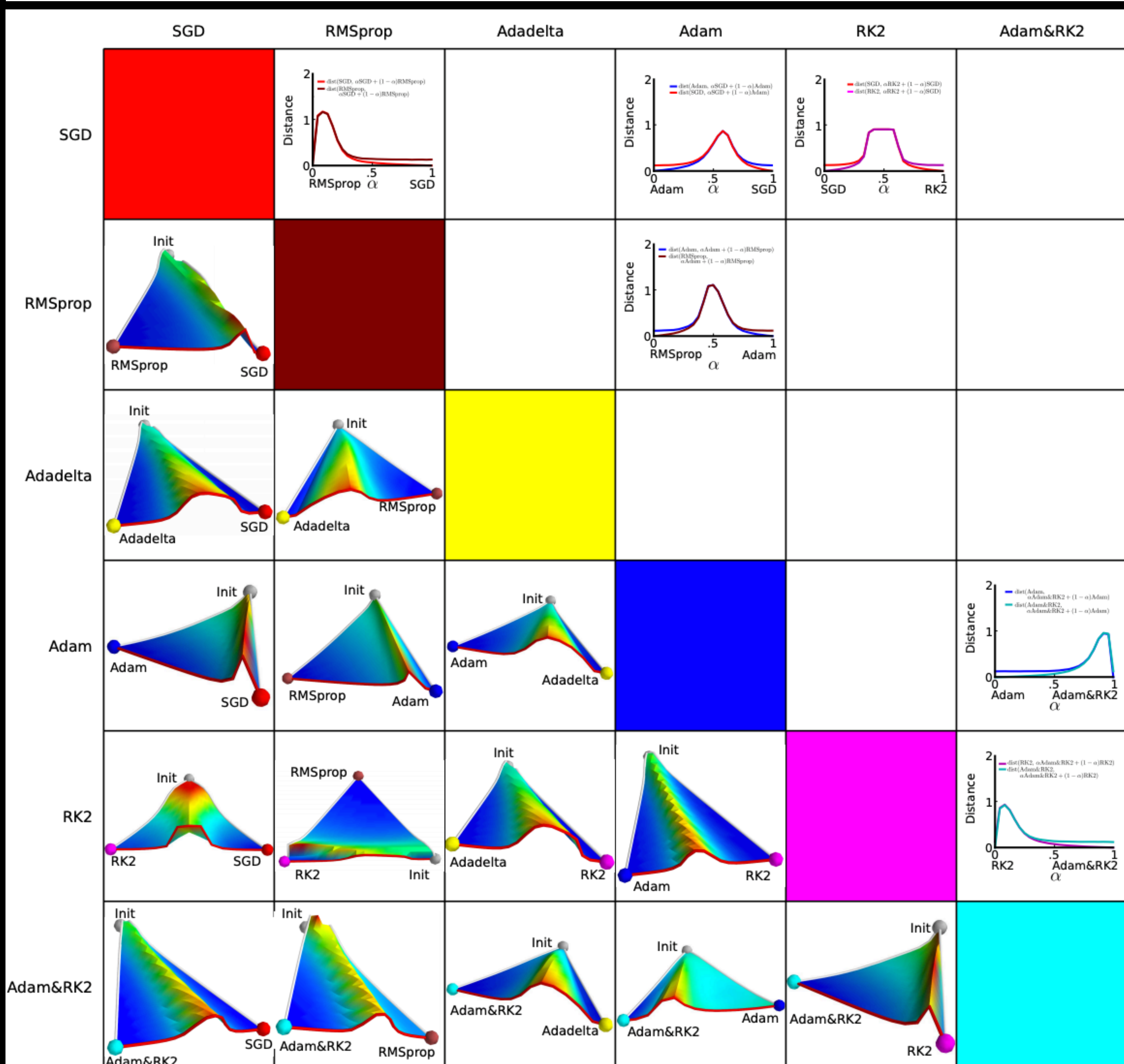
(Submitted on 14 Nov 2017 (v1), last revised 4 Jan 2019 (this version, v3))

Which to use?

An empirical analysis of the optimization of deep network loss surfaces

Daniel Jiwoong Im, Michael Tao, Kristin Branson

(Submitted on 13 Dec 2016 (v1), last revised 7 Dec 2017 (this version, v4))



- SGD + SGDM
- Familiar with
- Knowing your data
- Test on small batch
- Adam + SGD
- Shuffle
- Choosing prop LR