Important Techniques in Neural Network Training

Gradient Clipping

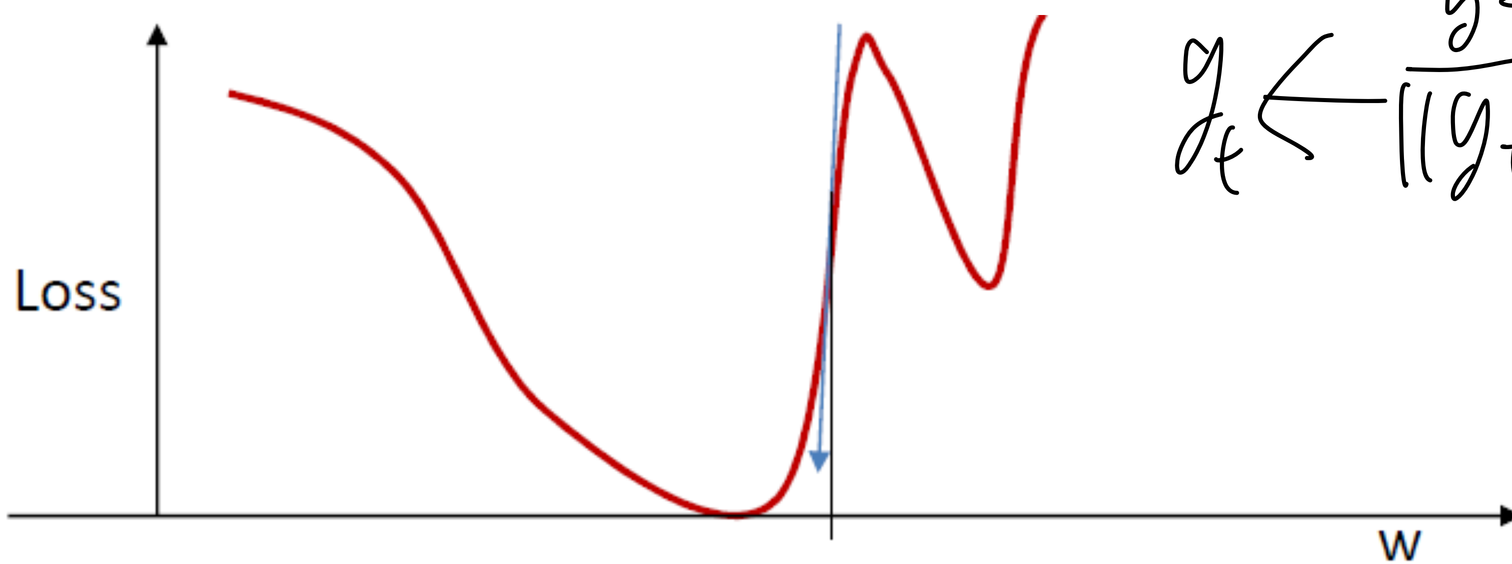
$X_t = X_t - \eta \nabla f(x_t)$

$g_t = \nabla f(X_t)$

- The loss can occasionally lead to a steep descent
- This result in immediate instability
- If gradient norm bigger than a threshold, set the gradient to the threshold.

if $\|g_t\|_2 > \text{threshold}$

$g_t \leftarrow \dfrac{g_t}{\|g_t\|_2} \cdot \text{threshold}$



Batch Normalization (Ioffe & Szegedy, '14)

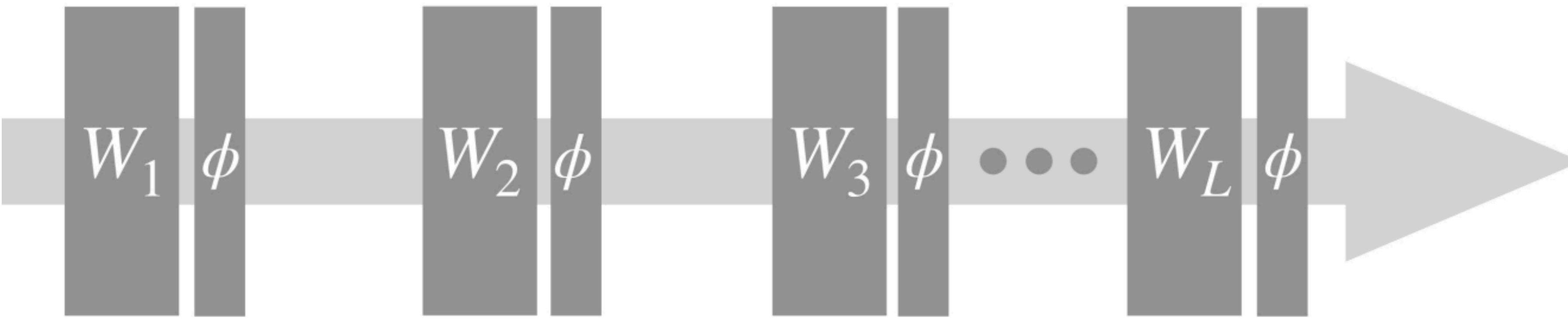
$x \to$ demean, divide by std

- Normalizing/whitening (mean = 0, variance = 1) the inputs is generally useful in machine learning.
  - Could normalization be useful at the level of hidden layers?
  - **Internal covariate shift**: the calculations of the neural networks change the distribution in hidden layers even if the inputs are normalized

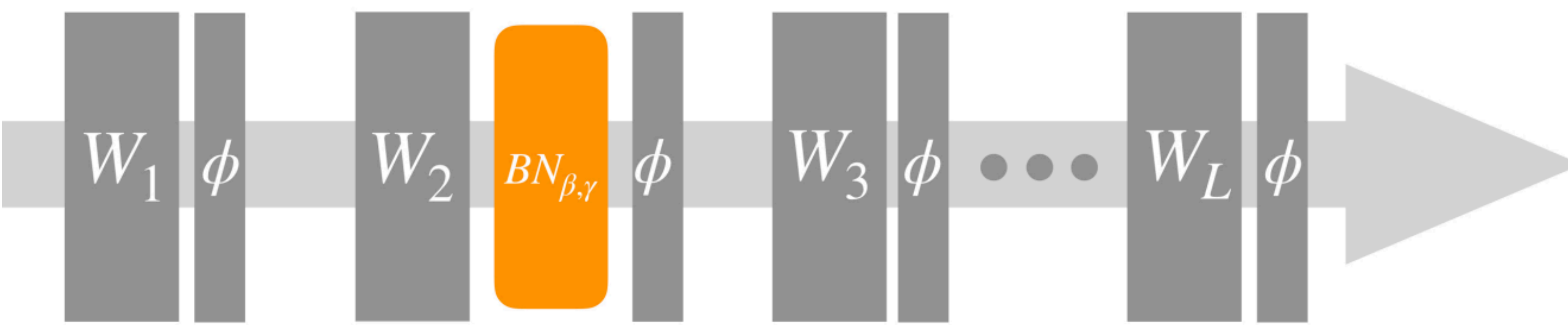
- **Batch normalization** is an attempt to do that:
  - Each unit's **pre-activation** is normalized (mean subtraction, std division)
  - During training, mean and std is computed for each minibatch (can be backproped!

Batch Normalization (Ioffe & Szegedy, ’14)

Standard Network



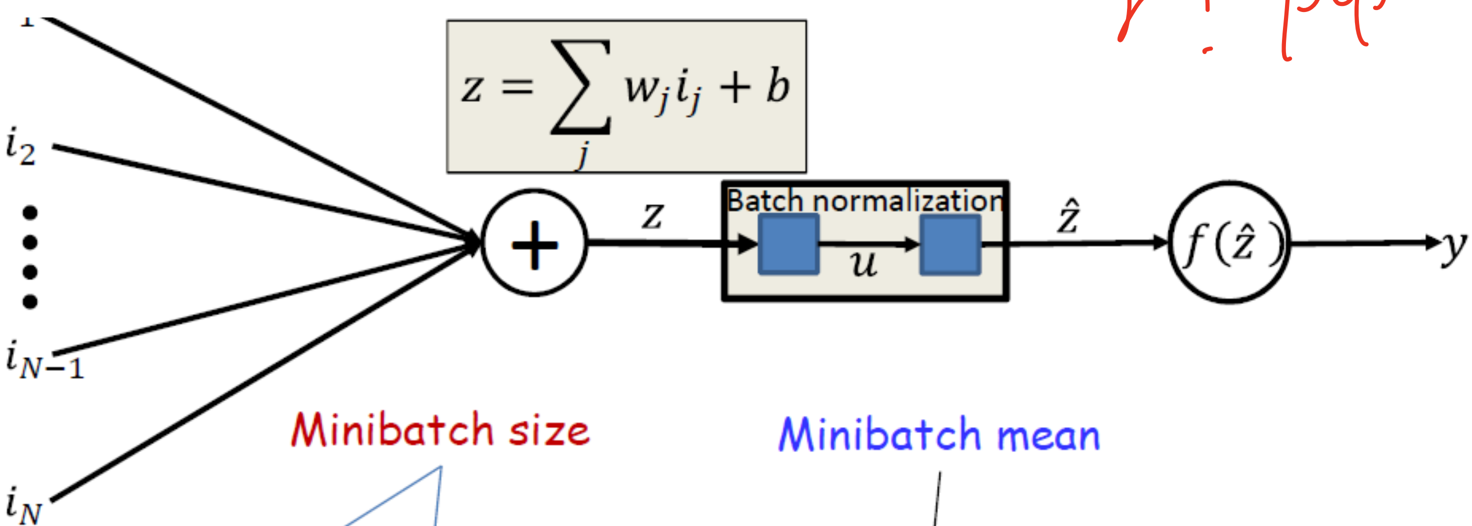
Adding a BatchNorm layer (between weights and activation function)



Batch Normalization (Ioffe & Szegedy, '14)

β: population mean

γ: population std



Minibatch size

Minibatch mean

Minibatch standard deviation

$$\mu_B = \frac{1}{B}\sum_{i=1}^{B} z_i$$

$$\sigma_B^2 = \frac{1}{B}\sum_{i=1}^{B} (z_i - \mu_B)^2$$

$$u_i = \frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$\hat{z}_i = \gamma u_i + \beta$$

$z_1, \ldots, z_B$

B: batch size

ϵ: to ensure denominator > 0

Batch Normalization (Ioffe & Szegedy, ’14)

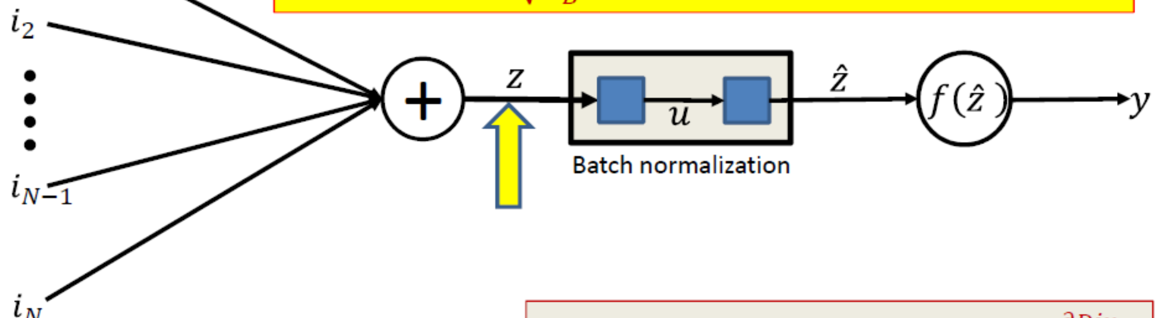
- BatchNorm at training time
  - Standard backprop performed for each single training data
  - Now backprop is performed over entire batch.

β, γ

$$\frac{\partial Div}{\partial \sigma_B^2} = \frac{-1}{2}(\sigma_B^2+\epsilon)^{-3/2}\sum_{i=1}^{B}\frac{\partial Div}{\partial u_i}$$

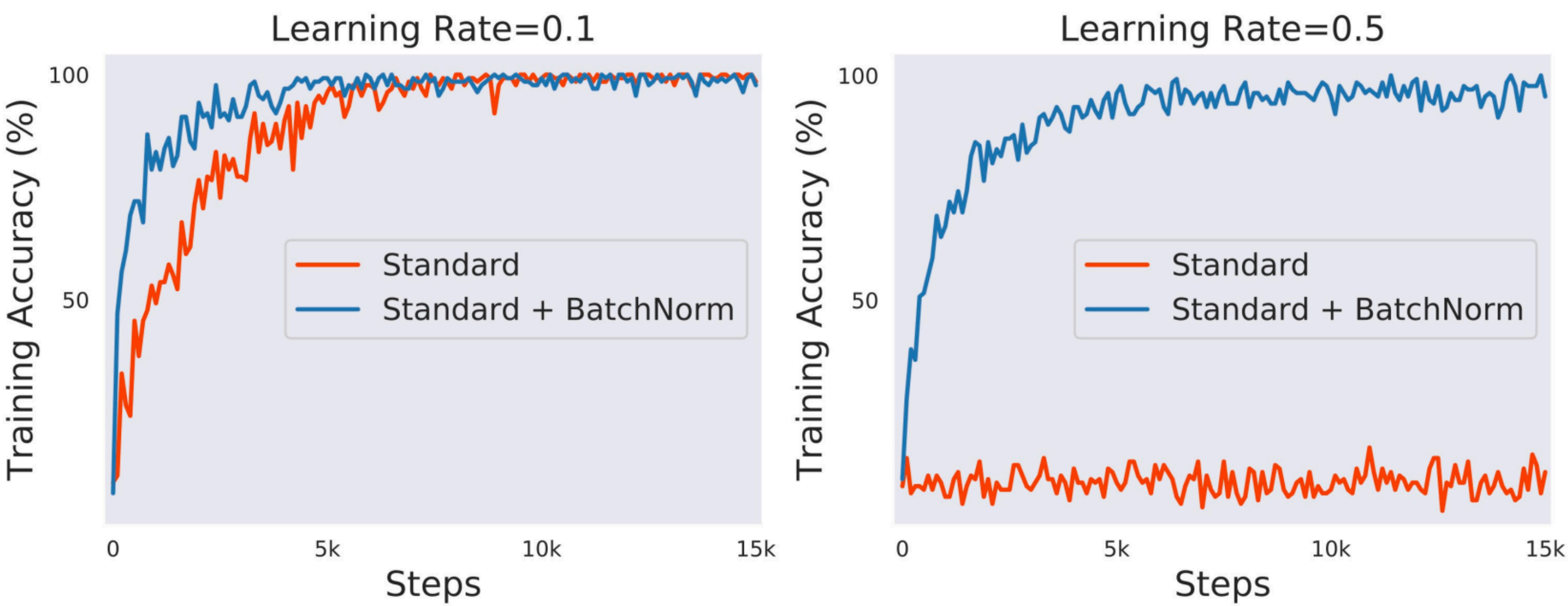
$$\frac{\partial Div}{\partial \mu_B} = \frac{-1}{\sqrt{\sigma_B^2+\epsilon}}\sum_{i=1}^{B}\frac{\partial Div}{\partial u_i}$$

$$\frac{\partial Div}{\partial z_i} = \frac{\partial Div}{\partial u_i}\cdot\frac{1}{\sqrt{\sigma_B^2+\epsilon}} + \frac{\partial Div}{\partial \sigma_B^2}\cdot\frac{2(z_i-\mu_B)}{B} + \frac{\partial Div}{\partial \mu_B}\cdot\frac{1}{B}$$



The rest of backprop continues from $\frac{\partial Div}{\partial z_i}$

Batch Normalization (Ioffe & Szegedy, ’14)



What is BatchNorm actually doing?

- May not due to covariate shift (Santurkar et al. ‘18):
  - Inject non-zero mean, non-standard covariance Gaussian noise after BN layer: removes the whitening effect
  - Still performs well.

- Only training β, γ with random convolution kernels gives non-trivial performance (Frankle et al. ’20)

- BN can use exponentially increasing learning rate! (Li & Arora ’19)

constant / $\frac{1}{\sqrt{t}}$

More normalizations

- Layer normalization (Ba, Kiros, Hinton, ’16)
  - Batch-independent
  - Suitable for RNN, MLP
- Weight normalization (Salimans, Kingma, ’16)
  - Suitable for meta-learning (higher order gradients are needed)
- Instant normalization (Ulyanov, Vedaldi, Lempitsky, ’16)
  - Batch-independent, suitable for generation tasks
- Group normalization (Wu & He, ‘18)
  - Batch-independent, improve BatchNorm for small batch size

NP-hard

Non-convex Optimization Landscape

Gradient descent finds global minima

min $L(\theta)$

⊠ overparameterized

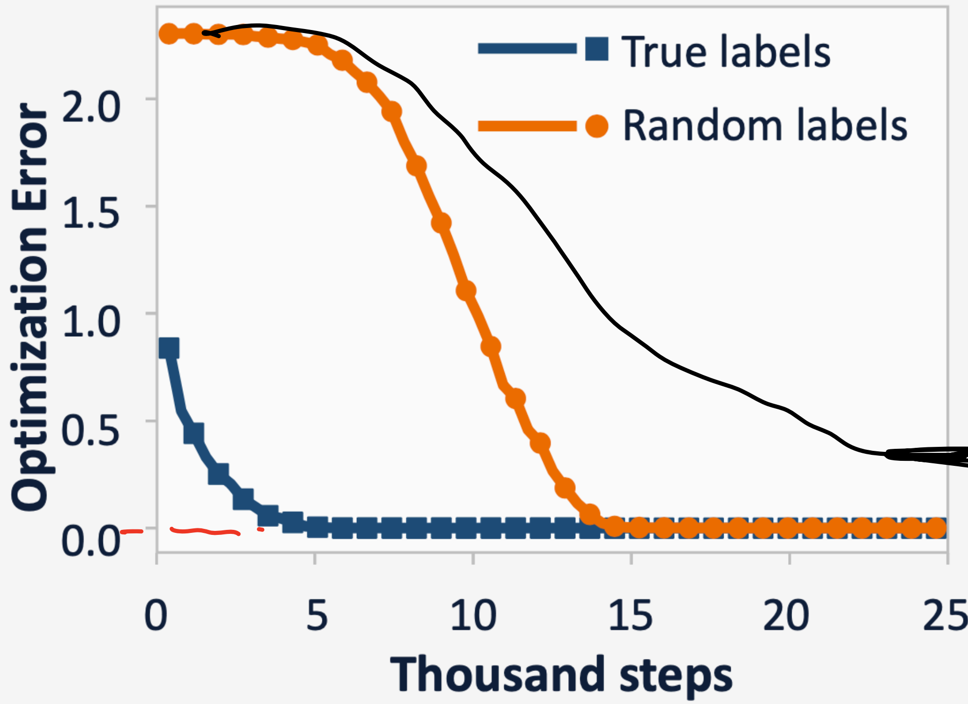
NN

$\dim(\theta) \gg n$

n: # of data

Practice: gradient descent

$$\theta(t+1) \leftarrow \theta(t) - \eta \frac{\partial L(\theta(t))}{\partial \theta(t)}$$



Optimization error **-> 0** for both ***true labels*** and ***random labels*** !

Zhang Bengio Hardt Recht Vinyals 2017

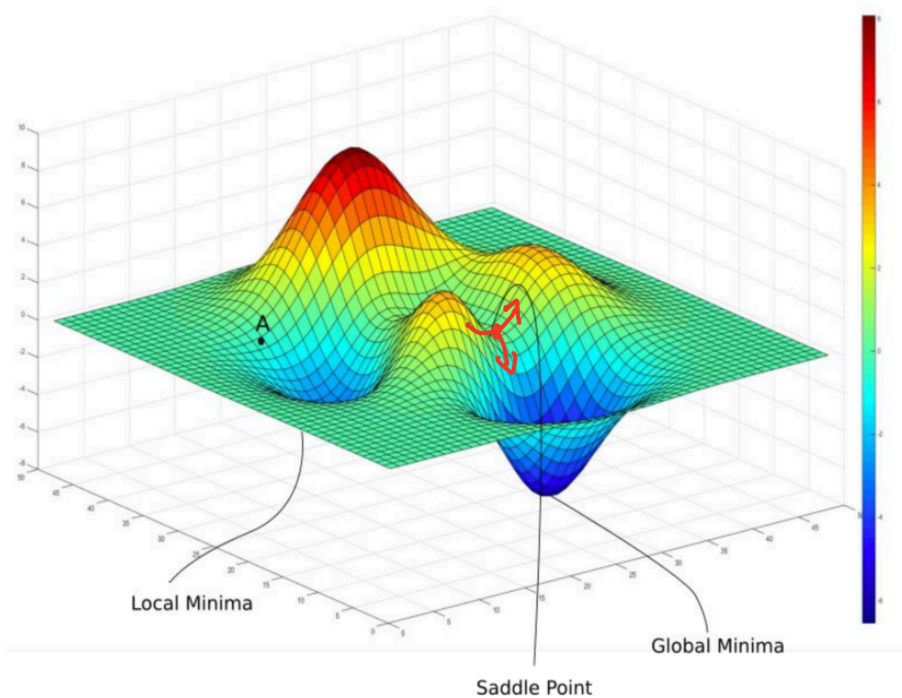
Understanding DL Requires Rethinking Generalization

Types of stationary points

$f(x)$

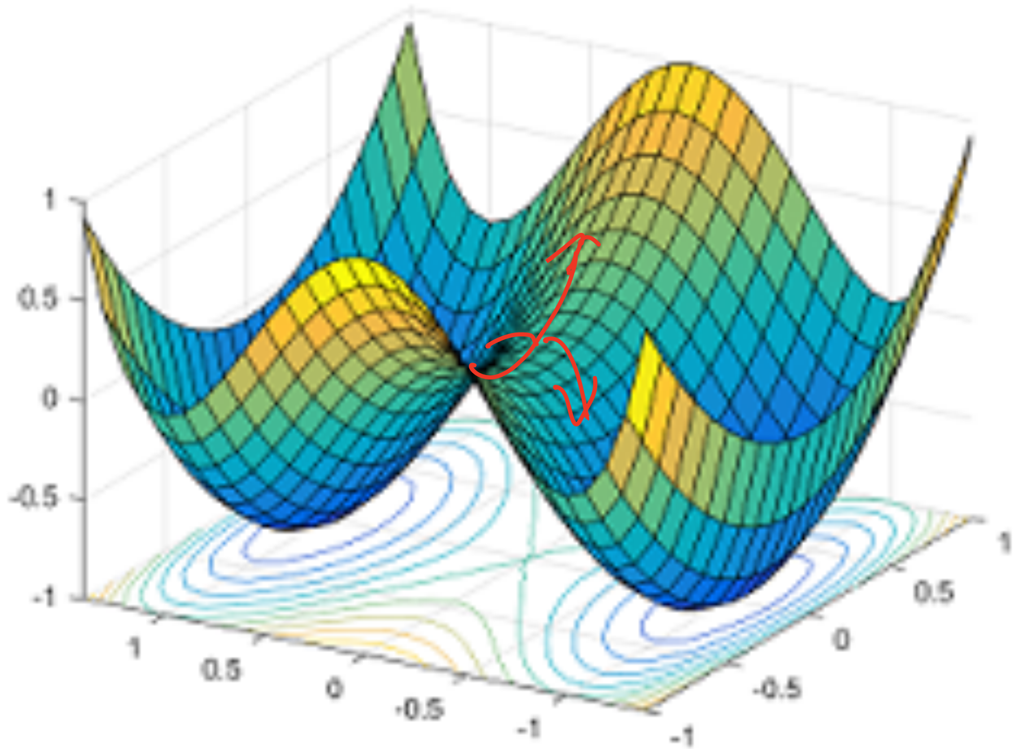
$\|\nabla f(x)\| \leq \varepsilon, \; O(\frac{1}{\varepsilon})$

- Stationary points: $x : \nabla f(x) = 0$,
- Global minimum:
  $x : f(x) \leq f(x') \forall x' \in \mathbb{R}^d$
- Local minimum:
  $x : f(x) \leq f(x') \forall x' : \|x - x'\| \leq \epsilon$
- Local maximum:
  $x : f(x) \geq f(x') \forall x' : \|x - x'\| \leq \epsilon$
- Saddle points: stationary points that are not a local min/max



$\nabla^2 f(x)$

Landscape Analysis



- All local minima are global!
- Gradient descent can escape saddle points.

Strict Saddle Points (Ge et al. ’15, Sun et al. ’15)

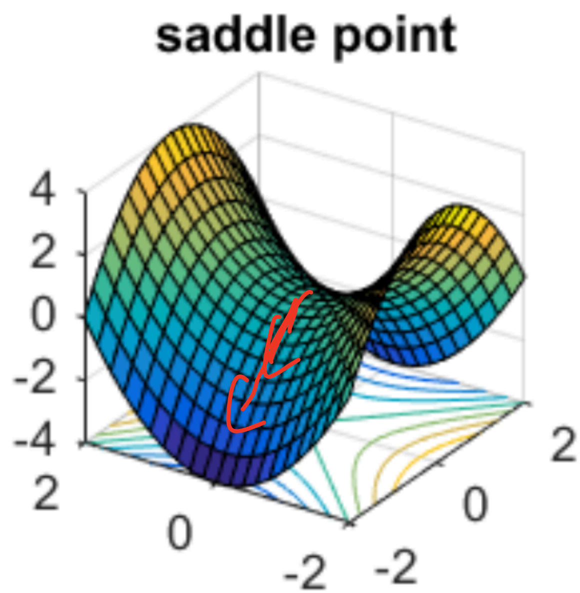
$f(x) = \frac{1}{2} x^T A x$

$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

if $x = 0$, $Ax = 0$

$\nabla^2 f(x) = A$

$\lambda_{min}(\nabla^2 f(x)) = -1$



$x = \begin{pmatrix} x^1 \\ x^2 \end{pmatrix}$

x

- Strict saddle point: a saddle point and $\lambda_{\min}(\nabla^2 f(x)) < 0$

$x_{t+1} = x_t - \eta A x_t$

$$x^2_{t+1} = x_t^2 + \eta x_t^2 = (1+\eta) x_t^2 = (1+\eta)^{t+1} x_0^2$$

if $x_0^2 \neq 0$

$|x^2_{t+1}| = (1+\eta)^{t+1} |x_0^2|$

Escaping Strict Saddle Points

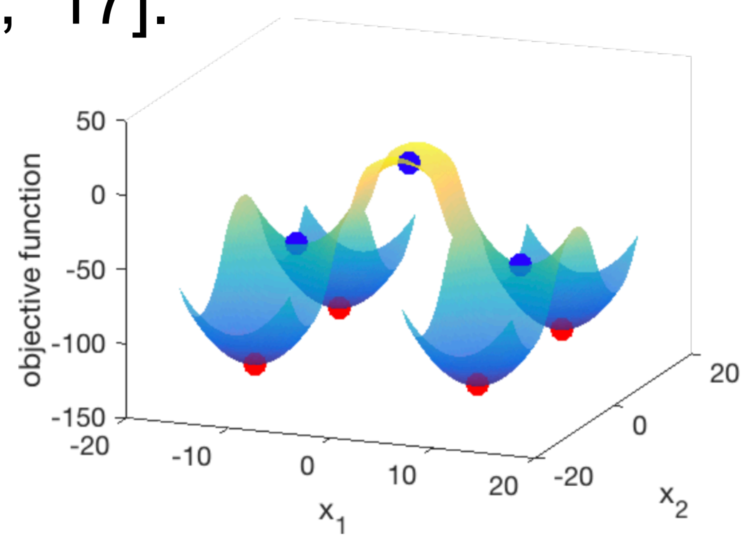
- Noise-injected gradient descent can escape strict saddle points in polynomial time [Ge et al., '15, Jin et al., '17].

$x_{t+1} = x_t - \eta \nabla f(x_t) + \eta \xi, \; \xi \sim \mathcal{N}(0, I)$

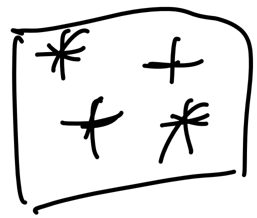
- Randomly initialized gradient descent can escape all strict saddle points asymptotically [Lee et al., '15]. X noise
  - Stable manifold theorem.

- Randomly initialized gradient descent can take exponential time to escape strict saddle points [Du et al., '17].

If 1) all local minima are global, and 2) are saddle points are strict, then noise-injected (stochastic) gradient descent finds a global minimum in polynomial time



What problems satisfy these two conditions

- Matrix factorization $\min_{U,V} \|UV^T - X\|_F^2$
- Matrix sensing $y_i = \langle A_i, X \rangle$, X: low-rank, $\min_{U,V} \sum_{i=1}^{n} (\langle A_i, UV^T \rangle - y_i)^2$
- Matrix completion 
- Tensor factorization
- Two-layer neural network with quadratic activation

$$f(x) = \sum_{j=1}^{m} \langle w_j, x \rangle^2$$

What about neural networks?

- Linear networks (neural networks with linear activations functions): all local minima are global, but there exists saddle points that are not strict [Kawaguchi '16].

$f(x) = W_{H+1} W_H \cdots W_1 x \Leftrightarrow f(x) = w^T x$

$\min_w \sum_{i=1}^n (w^T x_i - y_i)^2$

$\min_{W_{H+1}, \dots, W_1} \sum_{i=1}^n (f(x_i) - y_i)^2$

- Non-linear neural networks with:
  - Virtually any non-linearity, ReLU, sigmoid
  - Even with Gaussian inputs, $X \sim N(0, I)$
  - Labels are generated by a neural network of the same architecture, $y = NN(x)$ $\{(x_i, y_i)\}_{i=1}^n$, $\exists W, a$ $y_i = a^T \sigma(W x_i)$

There are many bad local minima [Safran-Shamir '18, Yun-Sra-Jadbaie '19].

$\exists x$, local but not global

Trajectory-based Analysis for Non-convex optimization

Gradient Flow: a Kernel Point of View

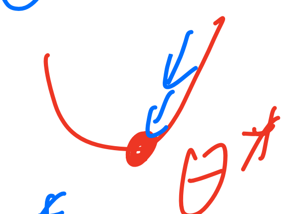
- $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\theta, x_i), y_i)$

$$\frac{\partial L(\theta)}{\partial \theta} = \frac{1}{n}\sum_{i=1}^{n} \ell'(f(\theta, x_i), y_i)\cdot \frac{\partial f(\theta, x_i)}{\partial \theta}$$

gradient flow: $$\frac{d\theta(t)}{dt} = -\frac{\partial L(\theta)}{\partial \theta}$$

if $L(\theta)$ is strongly convex, $\exists$ unique θ^*

$\theta(t) \longrightarrow \theta^*$



for NN, $\underline{\dim(\theta) > n}$, $\underline{\nexists\ \theta^*}$

$\Rightarrow$ study $\underline{f(\theta_t, x_i)}$ for each i $\longrightarrow y_i$ as $t \to \infty$

Gradient Flow: a Kernel Point of View

$$u_i(t) = f(\theta_t, x_i) \qquad u(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_n(t) \end{pmatrix} \in \mathbb{R}^n$$

$$\frac{d u_i(t)}{dt} = \left\langle \frac{\partial u_i(t)}{\partial \theta_t}, \frac{d\theta_t}{dt} \right\rangle$$

$$= \left\langle \frac{\partial u_i(t)}{\partial \theta_t}, -\frac{1}{n}\sum_{j=1}^{n} l'(u_j(t), y_j) \cdot \frac{\partial u_j(t)}{\partial \theta_t} \right\rangle$$

$$= -\frac{1}{n}\left[l'(u_1(t), y_1), \cdots, l'(u_n(t), y_n)\right]^T$$

$$\left[\left\langle \frac{\partial u_i(t)}{\partial \theta_t}, \frac{\partial u_1(t)}{\partial \theta_t}\right\rangle, \cdots, \left\langle \frac{\partial u_i(t)}{\partial \theta_t}, \frac{\partial u_n(t)}{\partial \theta_t}\right\rangle\right.$$

$l'(u(t), y) \in \mathbb{R}^n$

$[l'(u(t), y)]_i = l'(u_i(t), y)$

$H(t) \in \mathbb{R}^{n \times n}$

$$[H(t)]_{ij} = \left\langle \frac{\partial u_i(t)}{\partial \theta_t}, \frac{\partial u_j(t)}{\partial \theta_t} \right\rangle,$$

$$\boxed{\frac{du(t)}{dt} = -\frac{1}{n} H(t)\, l'(u(t), y)}$$

Gradient Flow: a Kernel Point of View

If ℓ quadratic, $\ell(u(t), y) = \frac{1}{2}\|u(t) - y\|_2^2$

$$\ell'(u(t), y) = u(t) - y$$

$$\frac{du(t)}{dt} = -\frac{1}{n} H(t)\left(u(t) - y\right)$$

If $H(t)$ is always P.D.

$\lambda_{\min}(H(t)) > 0$

$\Longrightarrow$ $u(t) \to y$