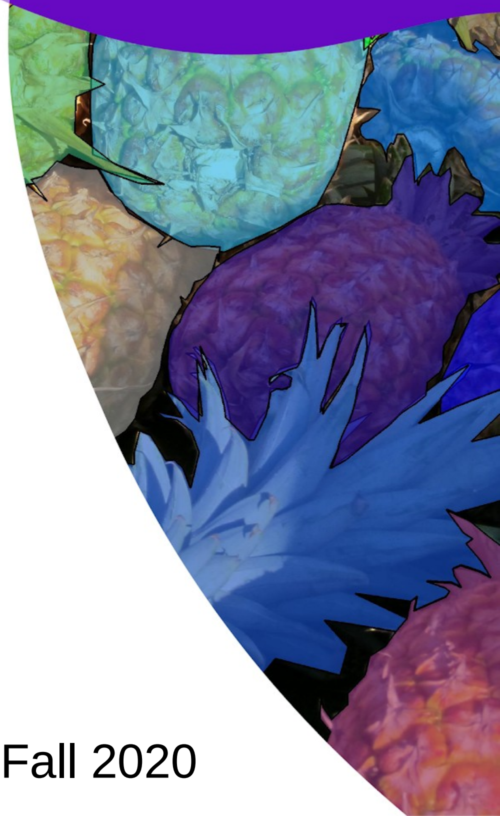
Optimization

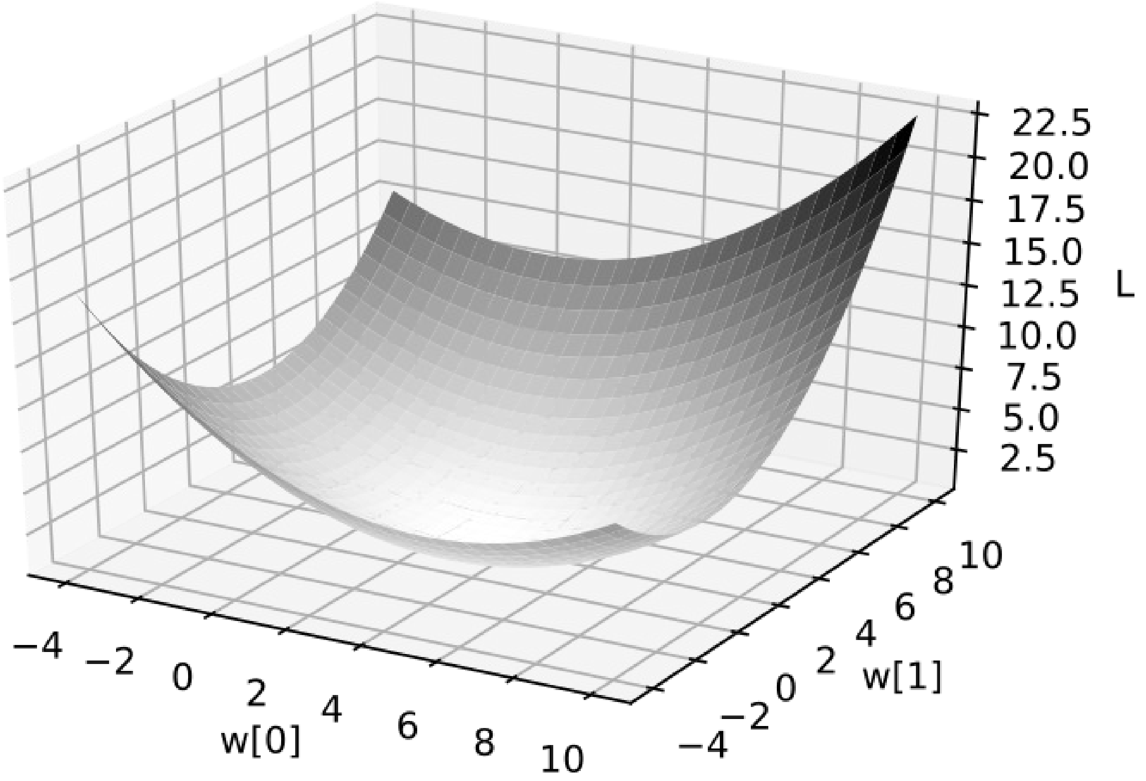
Yann LeCun
NYU - Courant Institute & Center for Data Science
Facebook AI Research

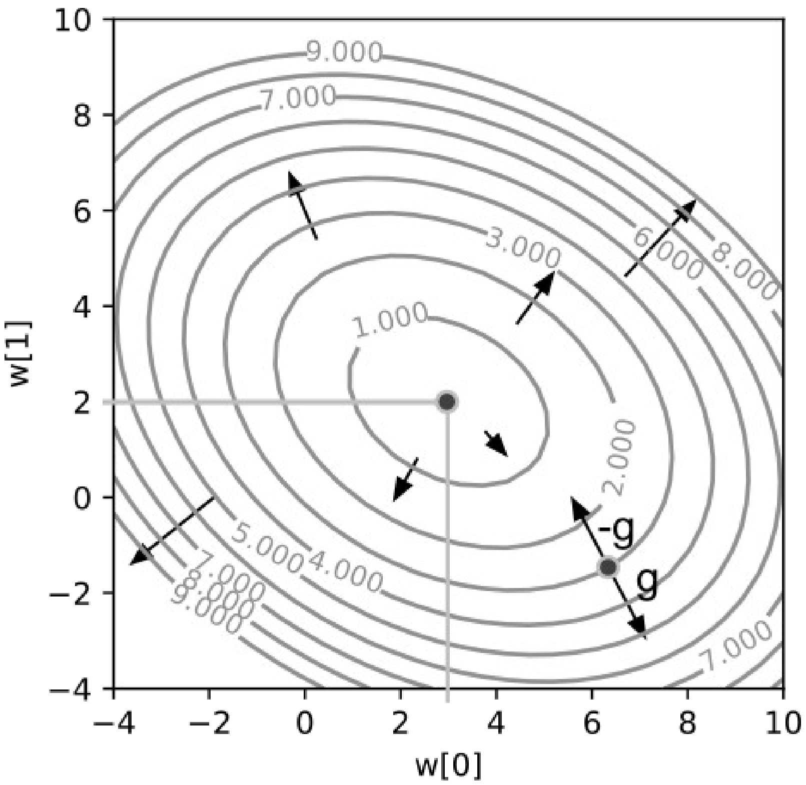


Deep Learning, NYU Fall 2020

Gradient descent

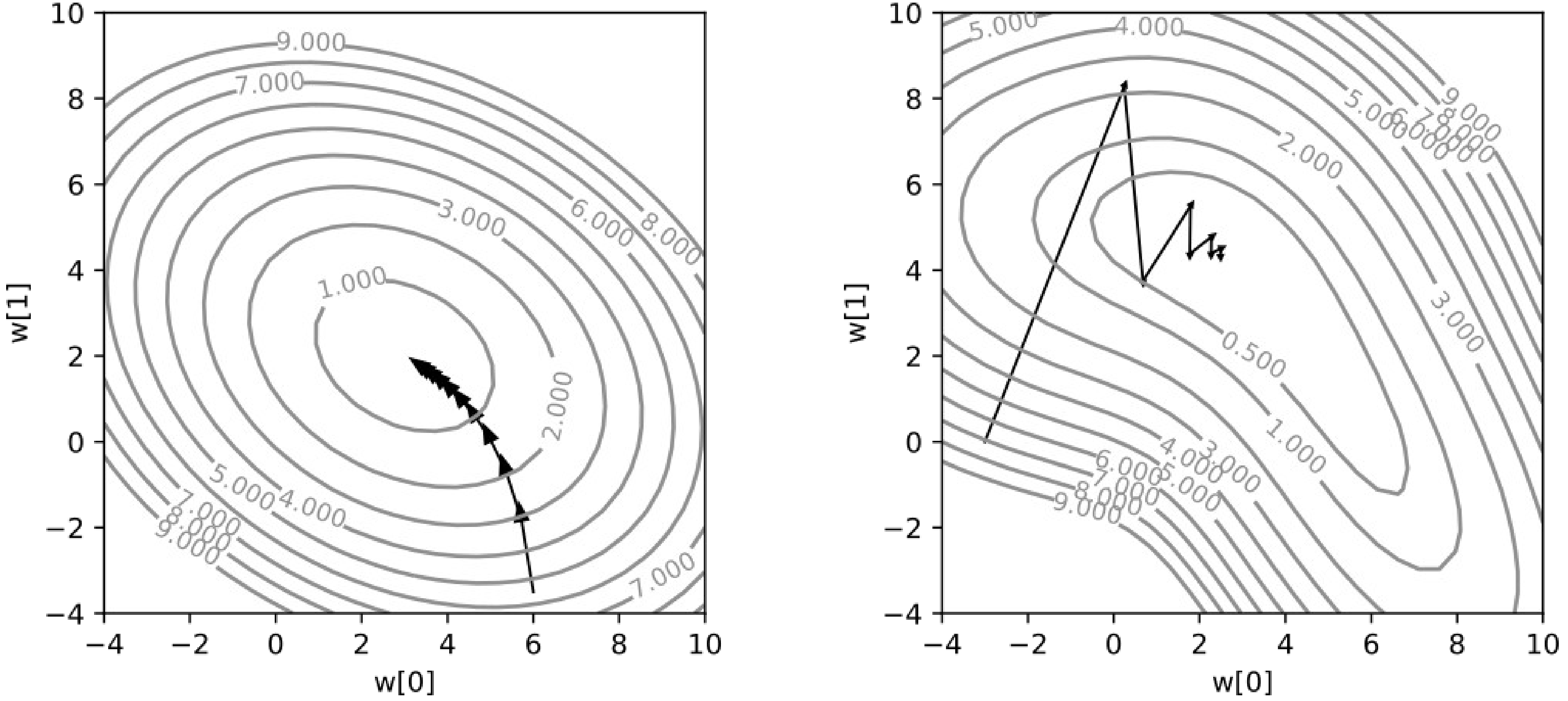
- **Convex, quadratic case.**





Gradient descent

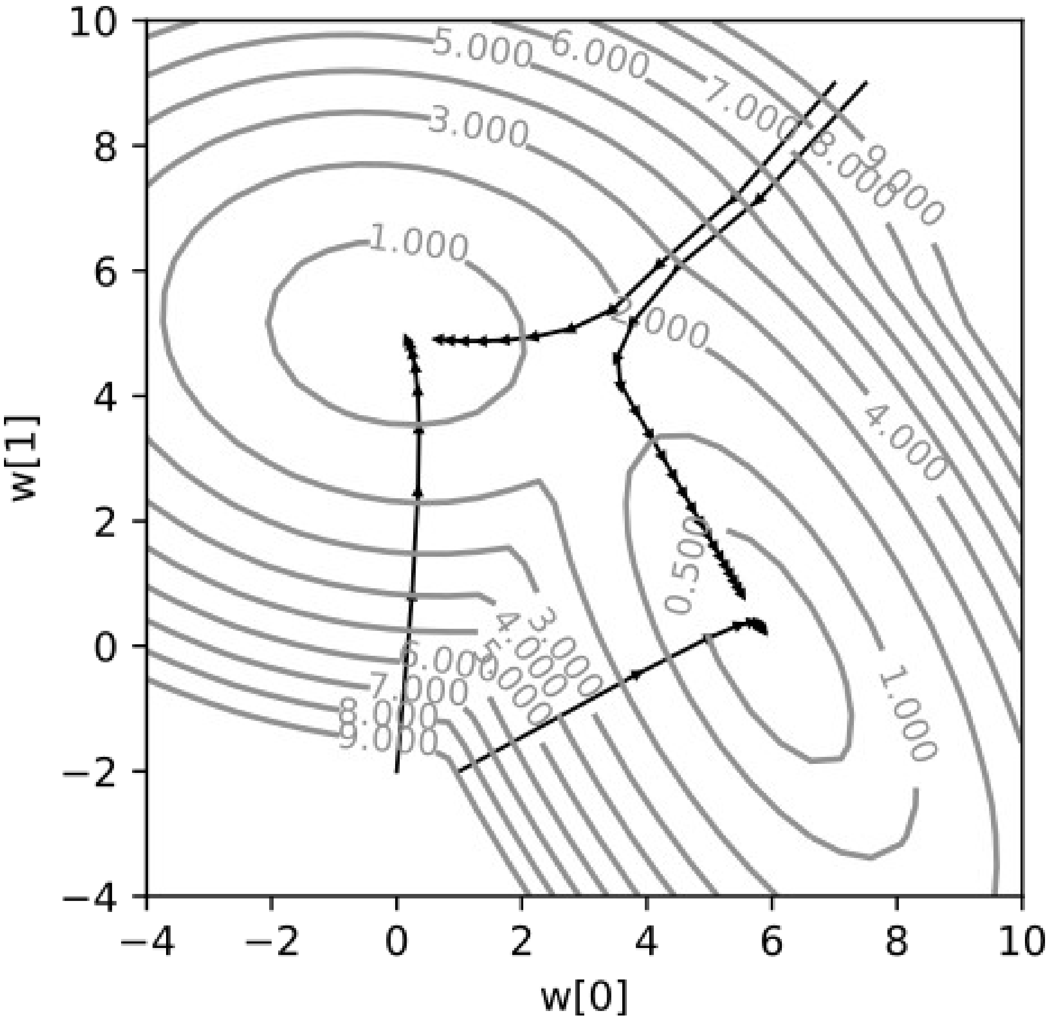
- **Small learning rate, Large learning rate**



Gradient descent

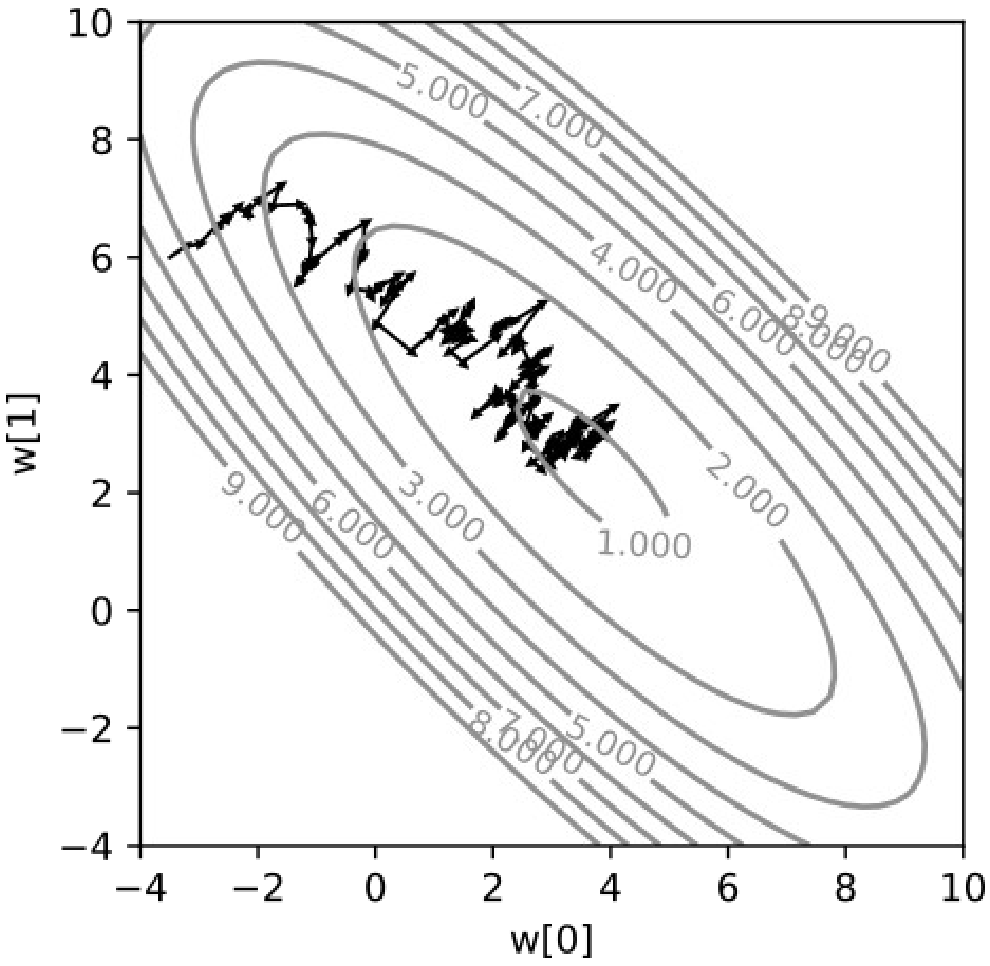
- **Non convex objective**

- **This is not not as much of a problem as you might think for neural nets**
  - Because of the high dimension
  - More on this later



Stochastic Gradient {Descent,Optimization}

- **For when the objective is an average of many similar terms**
- **Erratic but fast convergence**
- **Exploits the redundancy in the data**
- **Difficult to prove the convergence theoretically**
  - Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. Siam Review, 60(2), 223-311.



Y. LeCun

The Convergence of Gradient Descent

$$\omega \leftarrow \omega - \eta \frac{\partial E}{\partial \omega}$$

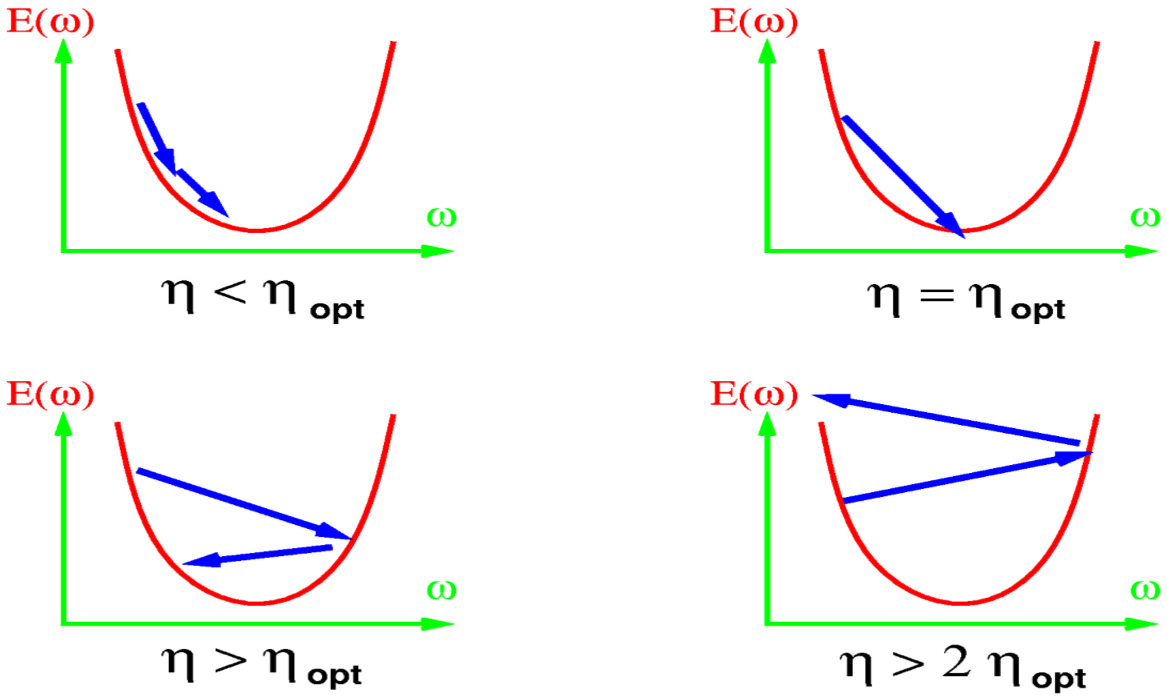
gradient of objective function

weight vector

learning rate

- Batch Gradient
- There is an optimal learning rate
- Equal to inverse 2nd

$$\boxed{\eta_{opt} = \left(\frac{\partial^2 E}{\partial \omega^2}\right)^{-1}}$$



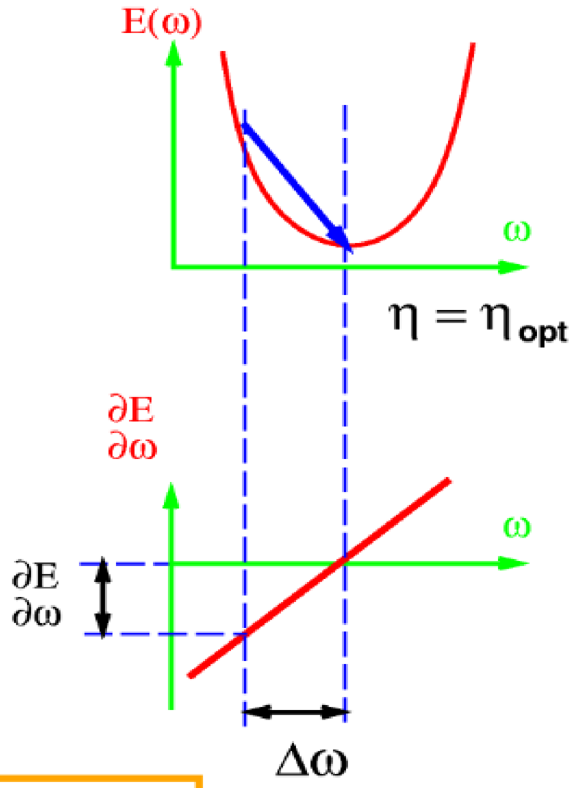
Optimal learning rate in 1D

Weight change:

$$\Delta\omega = \eta \frac{\partial E}{\partial \omega}$$

Assuming E is quadratic:

$$\frac{\partial^2 E}{\partial \omega^2} \Delta\omega = \frac{\partial E}{\partial \omega}$$



Optimal Learning Rate

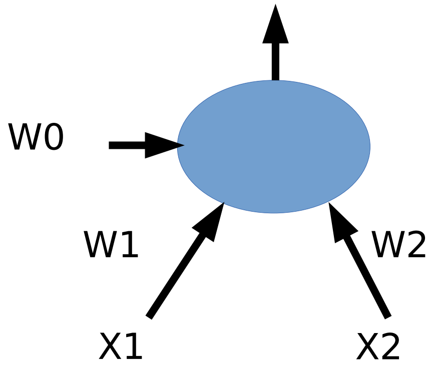
$$\boxed{\eta_{opt} = \left(\frac{\partial^2 E}{\partial \omega^2}\right)^{-1}}$$

Maximum Learning Rate

$$\boxed{\eta_{max} = 2\,\eta_{opt}}$$

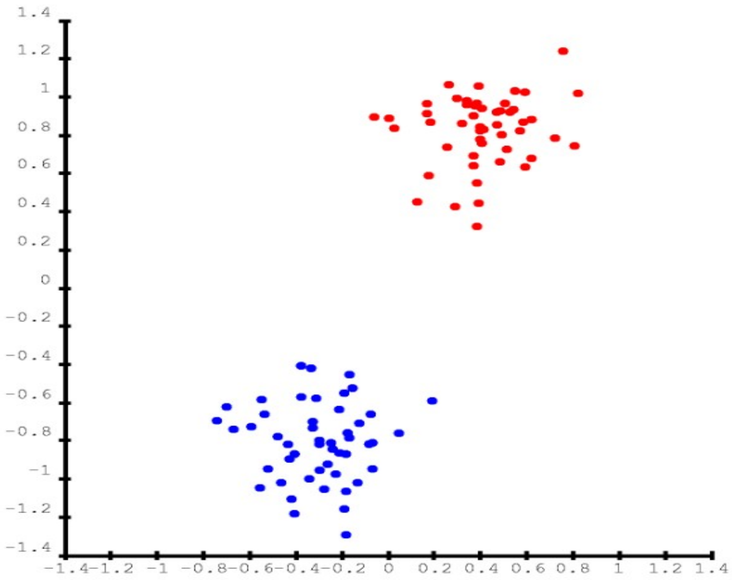
Let's Look at a single linear unit

- **Single unit, 2 inputs**



- **Quadratic loss**
  - $E(W) = 1/p \sum_p (Y - W \bullet X_p)^2$

- **Dataset: classification: Y=-1 for blue, +1 for red.**



- **Hessian is covariance matrix of input vectors**
  - $H = 1/p \sum X_p X_p^T$

- **To avoid ill conditioning: normalize the inputs**
  - Zero mean
  - Unit variance for all variable

Convergence is Slow When Hessian has Different Eigenvalues

Batch Gradient, small learning rate

Learning rate:

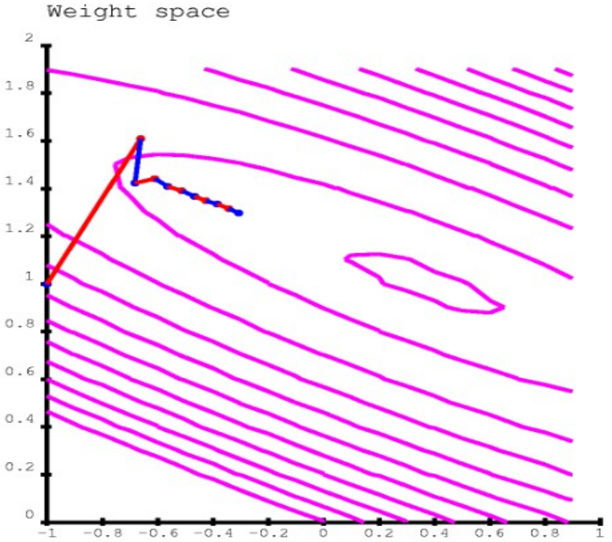
$\eta = 1.5$

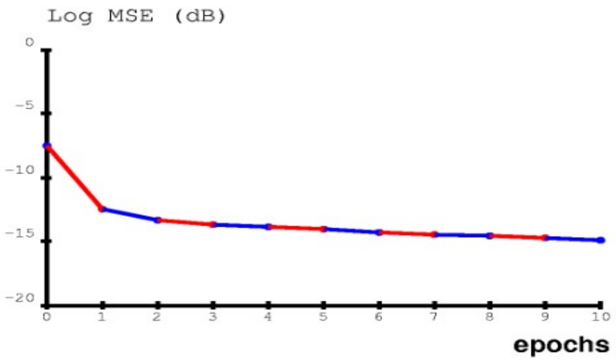
Hessian largest eigenvalue:

$\lambda_{max} = 0.84$

Maximum admissible Learning rate:

$\eta_{max} = 2.38$





Batch Gradient, large learning rate

Learning rate:

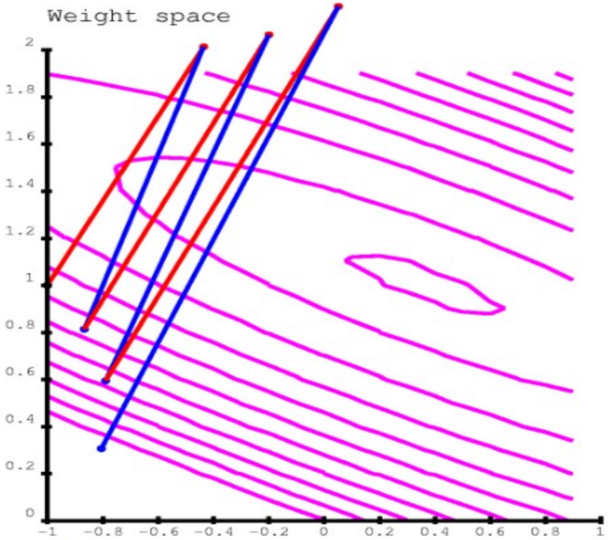
$\eta = 2.5$

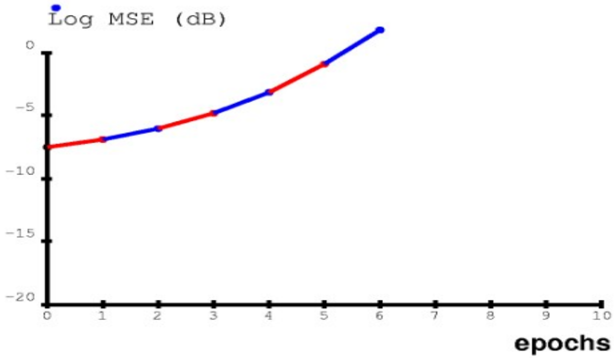
Hessian largest eigenvalue:

$\lambda_{max} = 0.84$

Maximum admissible Learning rate:

$\eta_{max} = 2.38$





Convergence is Slow When Hessian has Different Eigenvalues

- **Batch Gradient, small learning rate**

Learning rate:

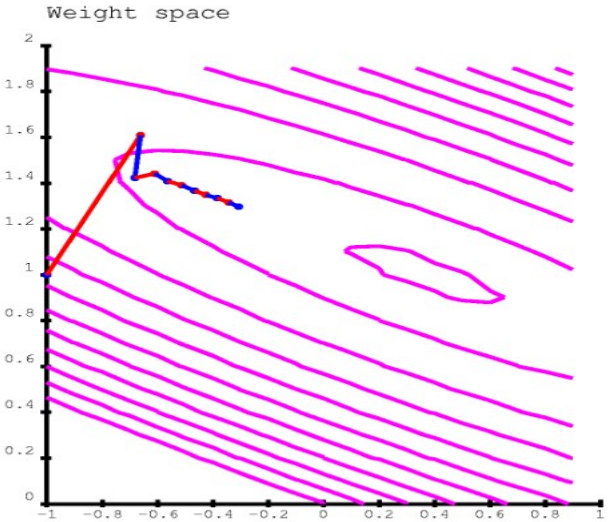
$\eta = 1.5$

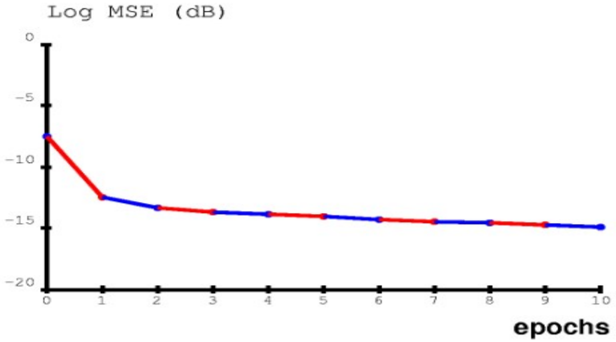
Hessian largest eigenvalue:

$\lambda_{max} = 0.84$

Maximum admissible Learning rate:

$\eta_{max} = 2.38$





- **Stochastic Gradient: Much Faster**
- **But fluctuates near the minimum**

Learning rate:

$\eta = 0.2$

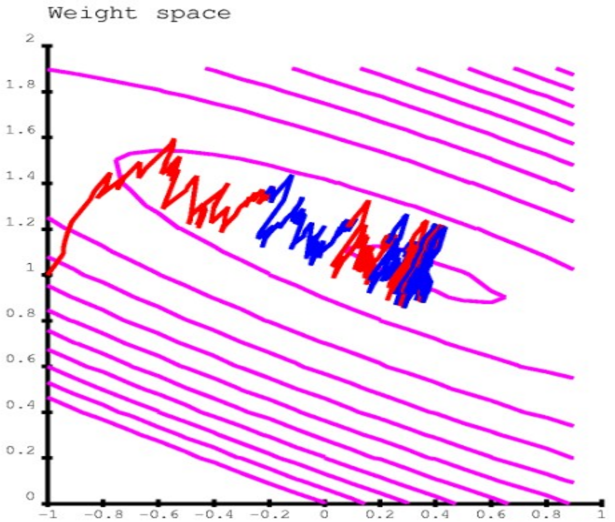
(equivalent to a batch learning rate of 20)

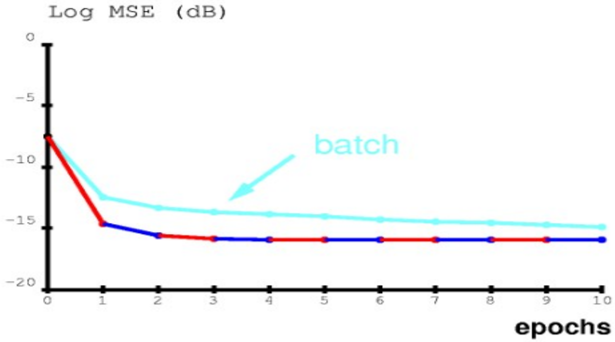
Hessian largest eigenvalue:

$\lambda_{max} = 0.84$

Maximum admissible Learning rate (for batch):

$\eta_{max} = 2.38$

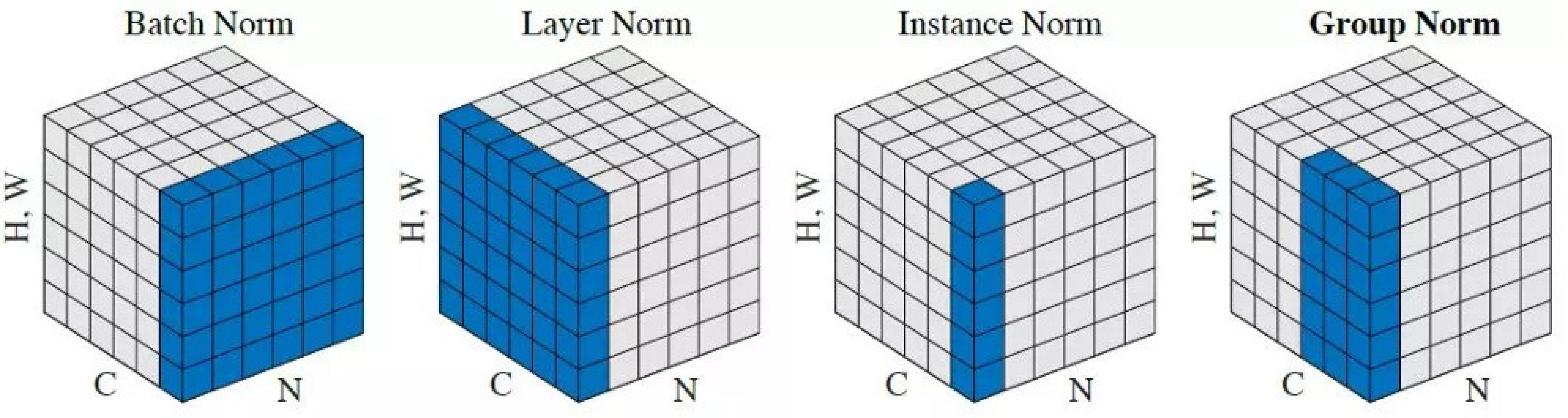




Convergence of GD

- **Convergence speed depends on conditioning**
- **Conditioning: ratio of largest to smallest non-zero eigenvalue of the Hessian**
- **How to condition?**
  - Center all the variables that enter a weight
  - Normalize the variance of all variables that enter a weight

Normalization tricks



- N=batch, C=channels, H,W space
- Batch norm: N,H,W
- Layer norm: C,H,W
- Instance norm: H,W
- Group norm: N, C subset

Normalization tricks

- **N=batch, C=channels, H,W space**
- **Batch norm: N,H,W**
- **Layer norm: C,H,W**
- **Instance norm: H,W**
- **Group norm: N, C subset**

- **Before non-linearity**
  - Needs scale and shift
- **After non-linearity**
  - Scale and shift may hurt

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

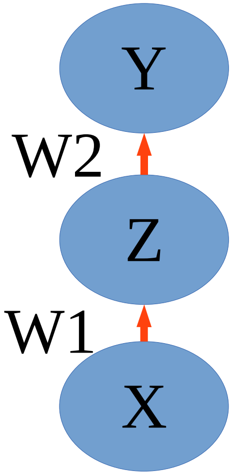
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

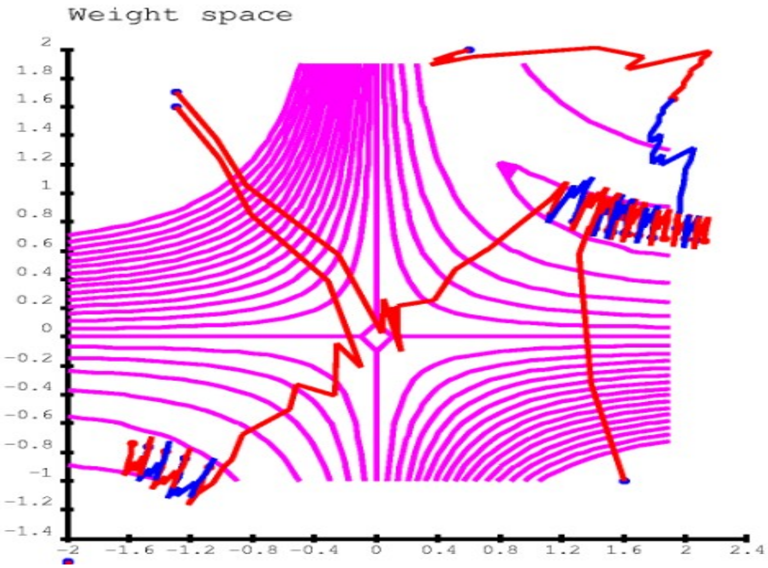
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

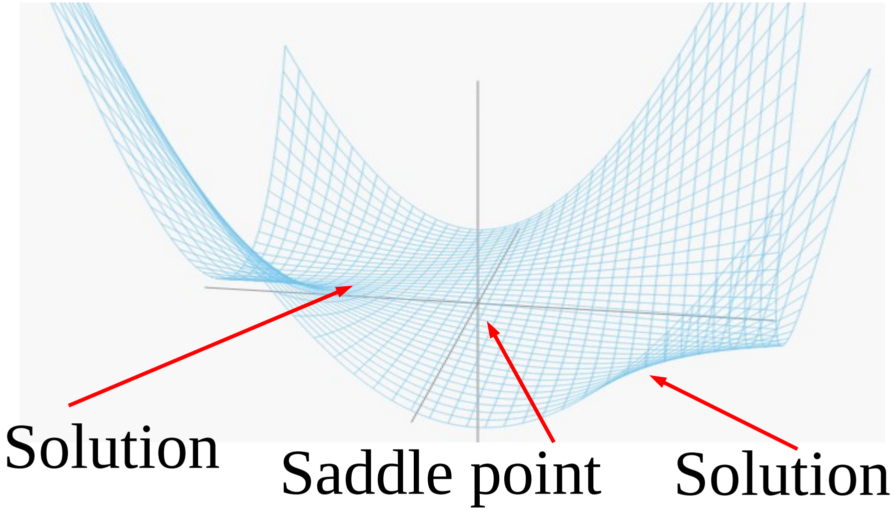
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Multilayer Nets Have Non-Convex Objective Functions

- **1-1-1 network**
  - Y = W1*W2*X
- **trained to compute the identity function with quadratic loss**
  - Single sample X=1, Y=1 L(W) = (1-W1*W2)^2
- **Solution: W2 = 1/W2 hyperbola.**



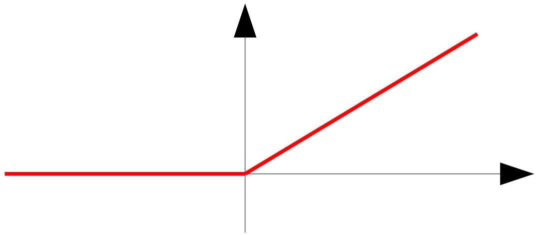


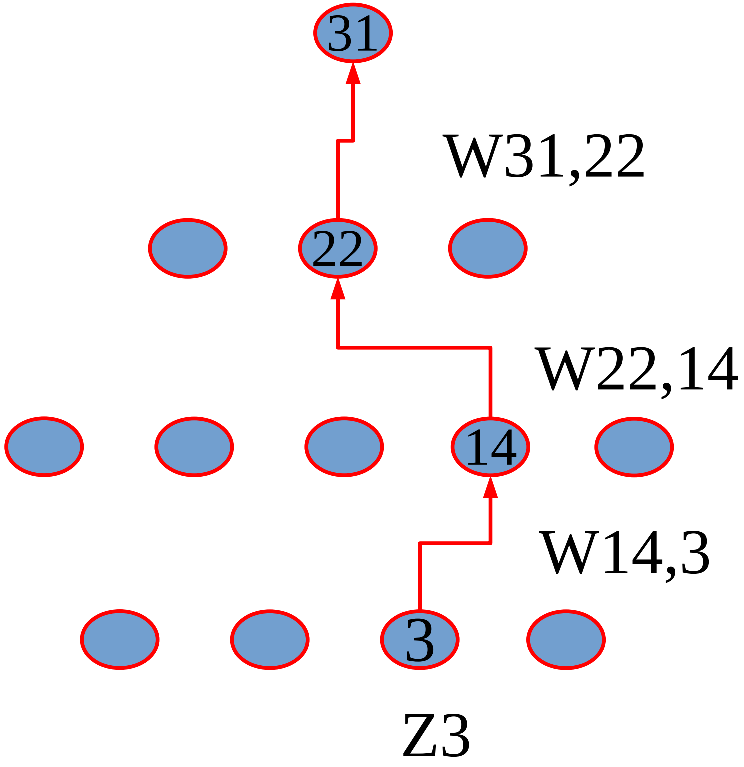


Deep Nets with ReLUs and Max Pooling

- **Stack of linear transforms interspersed with Max operators**
- **Point-wise ReLUs:**

$$ReLU(x) = max(x, 0)$$





- **Max Pooling**
  - "switches" from one layer to the next
- **Input-output function**
  - Sum over active paths
  - Product of all weights along the path
  - Solutions are hyperbolas
- **Objective function is full of saddle points**