Lecture 6. Neural Networks : Optimization (part1)

CMU 11-785   Introduction to Deep Learning

TAVE Research DL001
Changdae Oh

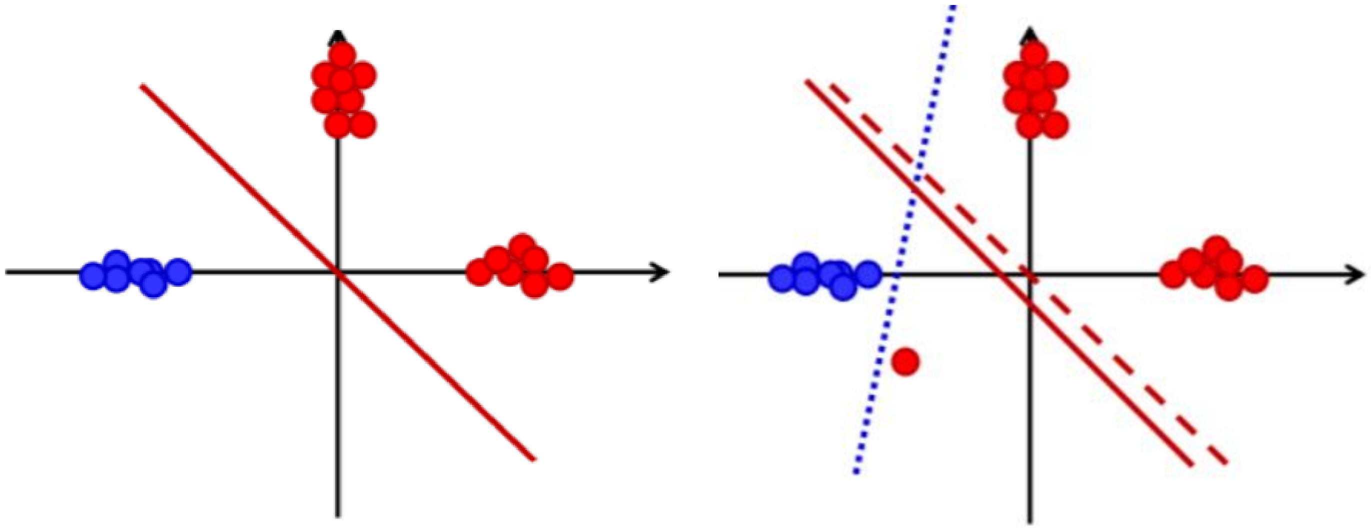
2021. 01. 31

Topics

- Convergence issues in backpropagation
- Second order methods
- Learning rate control

Convergence issues in backpropagation

Solution of backprop vs perceptron



low bias, high variance

vs

low variance, potential cost of bias

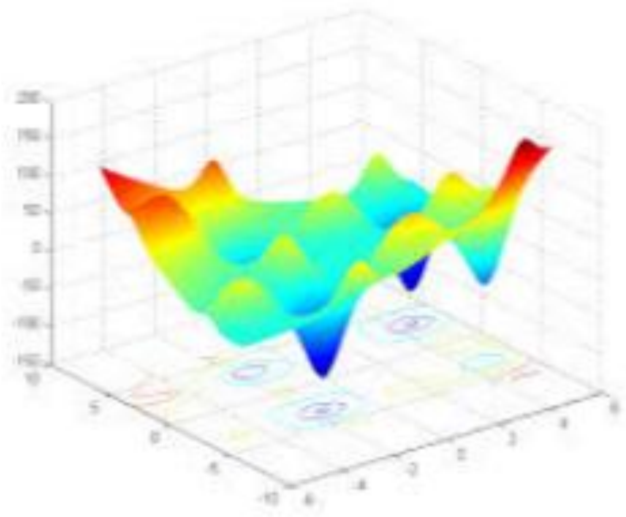
- Perceptron finds the perfect linear separator (if separable)
- Backprop will often not find a separating solution

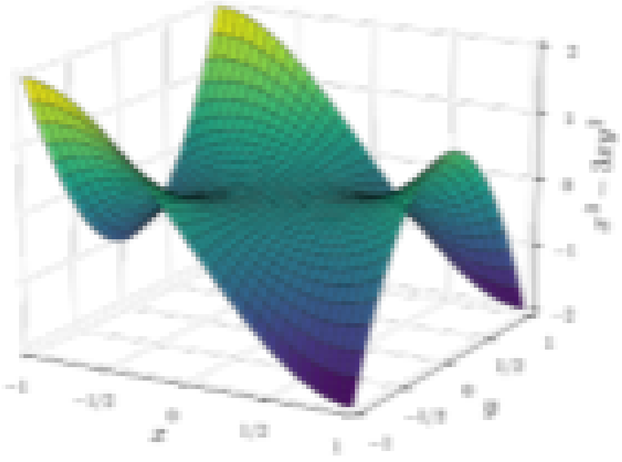
  (because the separating solution is not a feasible optimum for the loss fucntion)

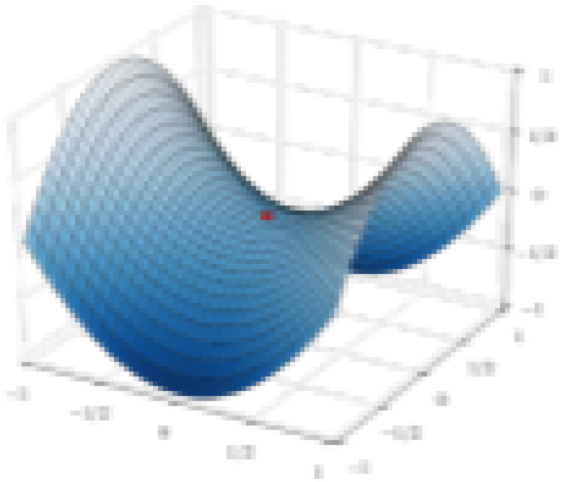
➡ **Lower variance** than an optimal classifier for the training data

Convergence issues in backpropagation

The Loss surface







- Saddle point
- Local minimum

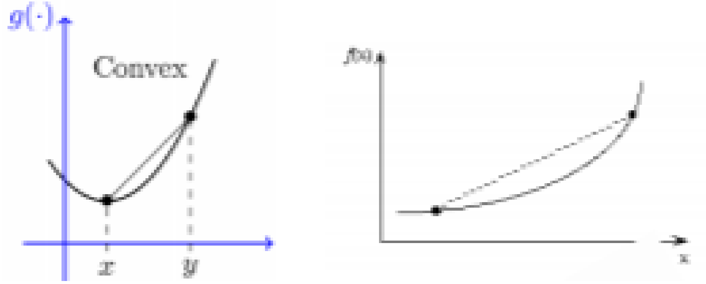
Popular Hypothesis :

- In large networks, saddle points are far more common than local minima
- Most local minima are equivalent and close to global minimum

Fact) Gradient descent algorithms often get stuck in saddle points

Second order methods

Definition of convex

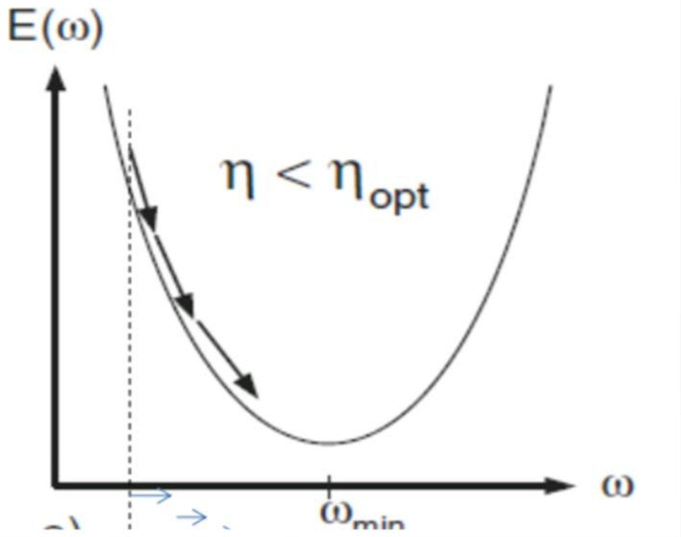


Convex optimization

Gradient descent with fixed step size eta to estimate scalar parameter w

$$Minimize\ E = \frac{1}{2}aw^2 + bw + c$$

$$w^{(k+1)} = w^{(k)} - \eta \frac{dE\left(w^{(k)}\right)}{dw}$$



How can we determine the optimal step size ?

Newton' s method

$$E(w) = E\left(w^{(k)}\right) + E'\left(w^{(k)}\right)\left(w - w^{(k)}\right) + \frac{1}{2}E''\left(w^{(k)}\right)\left(w - w^{(k)}\right)^2$$

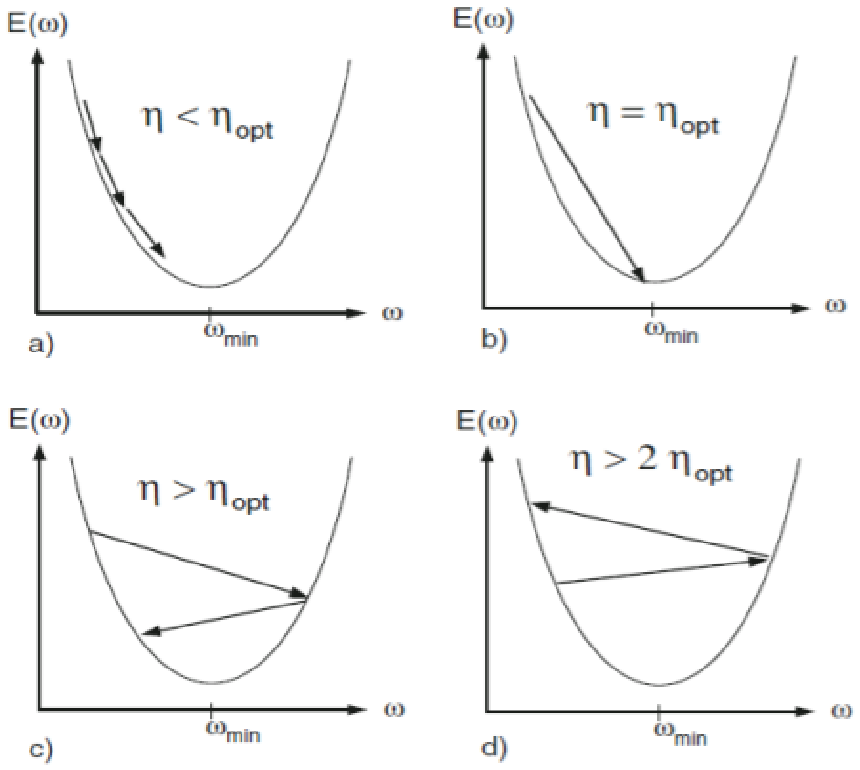
$$w_{min} = w^{(k)} - E''\left(w^{(k)}\right)^{-1} E'\left(w^{(k)}\right)$$

$$\boxed{\eta_{opt} = E''\left(w^{(k)}\right)^{-1} = a^{-1}}$$

Second order methods

Convex optimization

non-optimal step size



Generalize

Any differentiable convex objective function can be approx. as

$$E \approx E\left(\mathrm{w}^{(k)}\right) + \left(w - \mathrm{w}^{(k)}\right)\frac{dE\left(\mathrm{w}^{(k)}\right)}{dw} + \frac{1}{2}\left(w - \mathrm{w}^{(k)}\right)^2 \frac{d^2E\left(\mathrm{w}^{(k)}\right)}{dw^2} + \cdots$$

$$\boxed{\eta_{opt} = \left(\frac{d^2E\left(\mathrm{w}^{(k)}\right)}{dw^2}\right)^{-1}}$$

Second order methods

For Functions of multivariate inputs

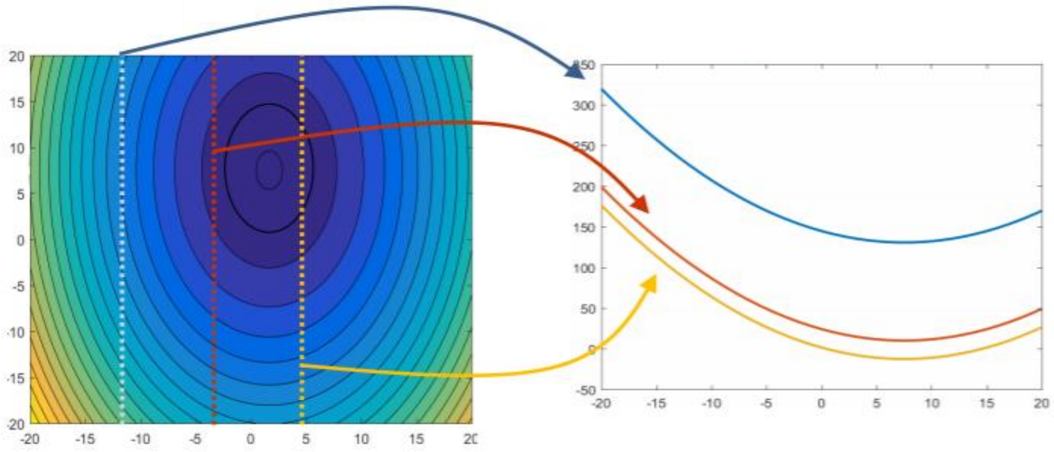
Simple quadratic convex function

$$E = \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} + \mathbf{w}^T\mathbf{b} + c$$

(E : convex -> A : positive definite)

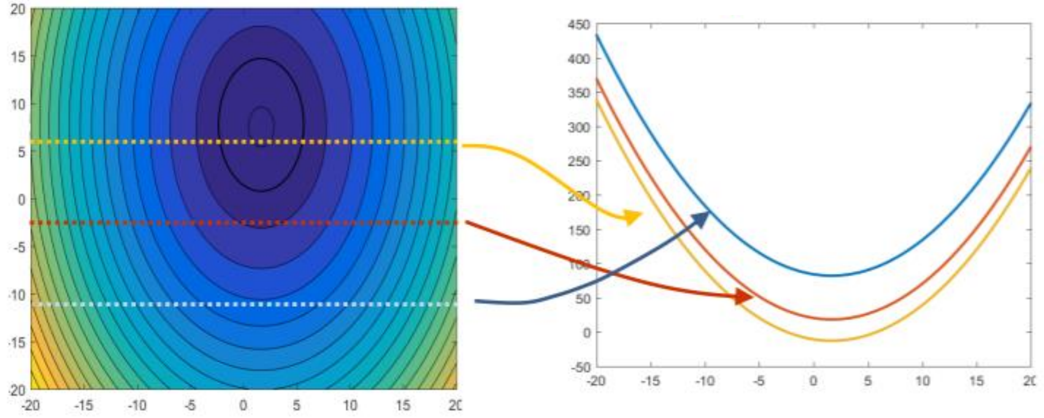
When A is diagonal

$$\Rightarrow \quad E = \frac{1}{2}\sum_i \left(a_{ii}w_i^2 + b_i w_i\right) + c$$



$$E = \frac{1}{2}a_{11}w_1^2 + b_1 w_1 + c + C(\neg w_1)$$

$$\eta_{1,opt} = a_{11}^{-1}$$



$$E = \frac{1}{2}a_{22}w_2^2 + b_2 w_2 + c + C(\neg w_2)$$

$$\eta_{2,opt} = a_{22}^{-1}$$

- The optimum of each coordinates is not affected by the other coordinates
- **Optimal learning rate is different for the different coordinates**

Second order methods

For Functions of multivariate inputs

$$\eta < 2 \min_i \eta_{i,opt}$$

$$\eta_{i,opt} \leq \eta < 2\eta_{i,opt}$$

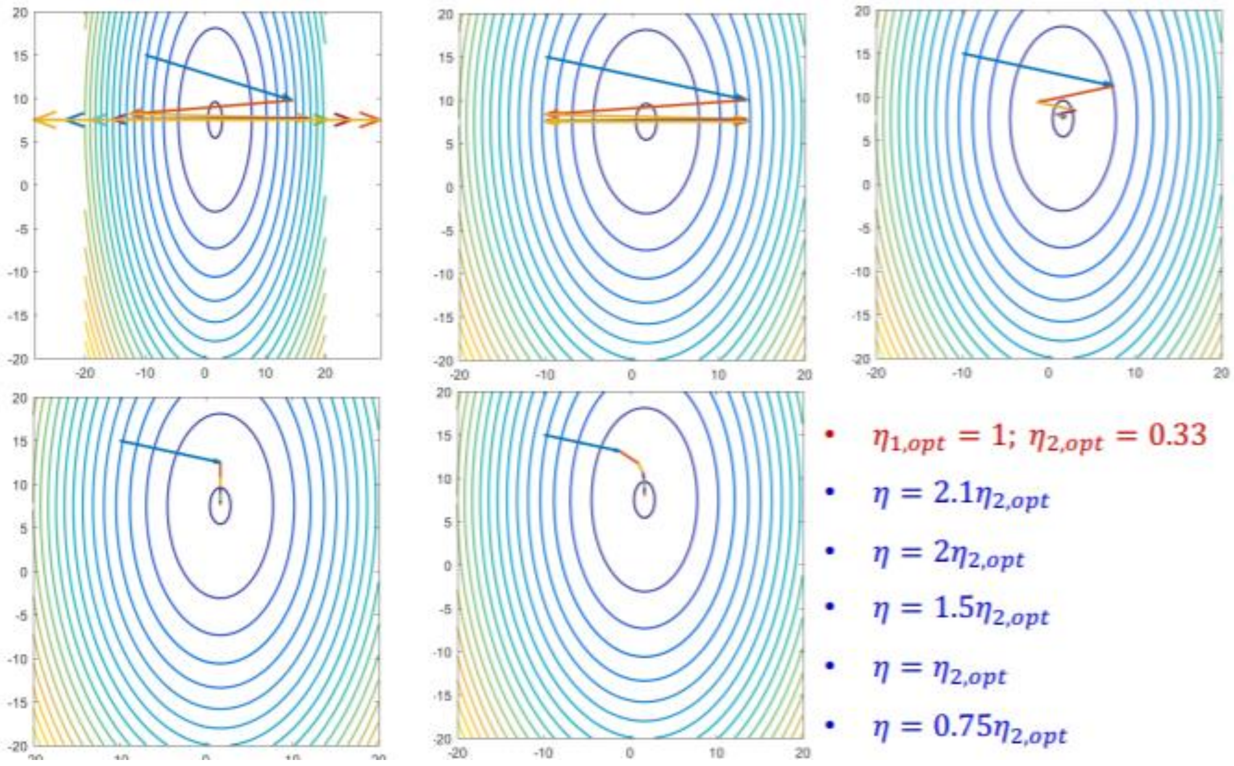
BUT

conventional vector Update rules
for gradient descent

$$w_i^{(k+1)} = w_i^{(k)} - \eta \frac{\partial E\left(w_i^{(k)}\right)}{\partial w}$$

1. Fixed learning rate

2. The same learning rate is applied to all components



- $\eta_{1,opt} = 1;\ \eta_{2,opt} = 0.33$
- $\eta = 2.1\eta_{2,opt}$
- $\eta = 2\eta_{2,opt}$
- $\eta = 1.5\eta_{2,opt}$
- $\eta = \eta_{2,opt}$
- $\eta = 0.75\eta_{2,opt}$

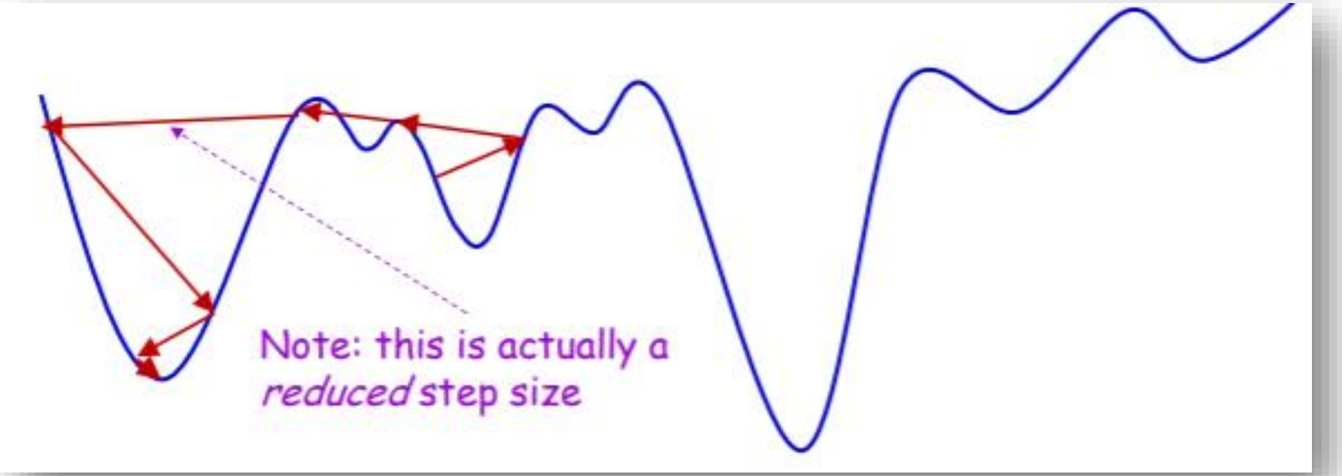
Slow & Oscillate ..

Many problems arise because of
requiring a fixed step size across all dimension

Learning rate control

Decaying learning rate

$\eta > 2\eta_{opt}$



- Start with a large learning rate
- Gradually reduce it with iterations

– Converge faster

– Help escape local optima

< Scheduling strategy >

– Linear decay: $\eta_k = \frac{\eta_0}{k+1}$

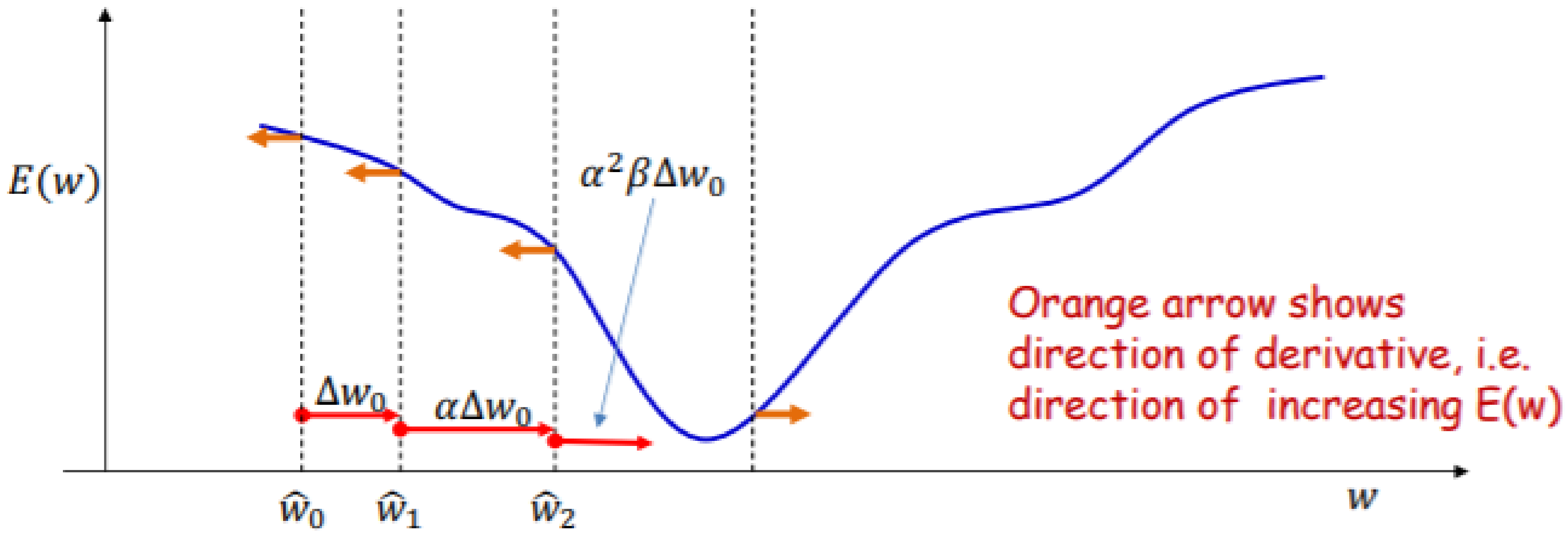
– Quadratic decay: $\eta_k = \frac{\eta_0}{(k+1)^2}$

– Exponential decay: $\eta_k = \eta_0 e^{-\beta k}$, where $\beta > 0$

Learning rate control

Rprop

- Simple first order algorithm but efficient
- Applied independently to each component
- minimal assumptions about the loss function (no convexity assumption)



- Same sign -> increasing learning rate
- different sign -> decreasing learning rate

Learning rate control

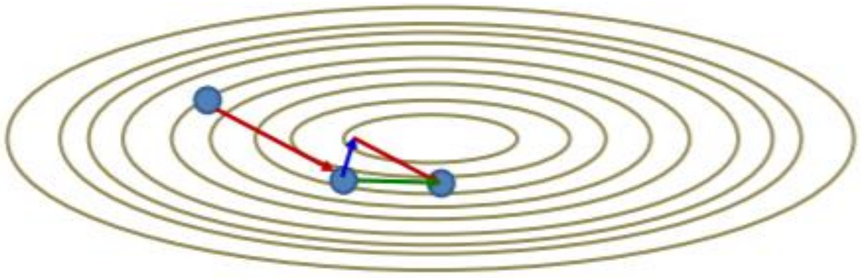
Momentum & Nesterov

Proposal :

- Emphasize steps in directions that converge smoothly
- Shrink steps in directions that bounce around

➡ Maintain a running average of all past steps

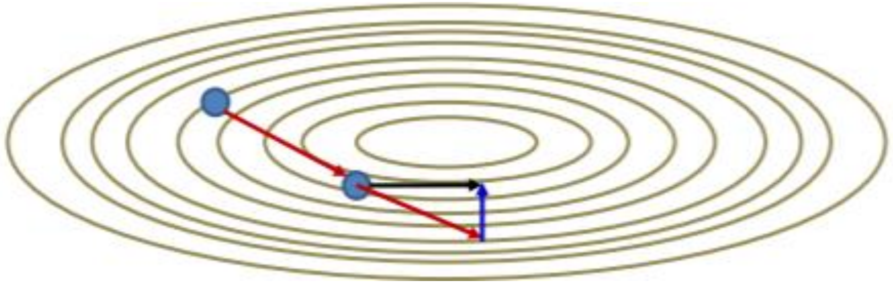
Momentum



$$\Delta W^{(k)} = \beta \Delta W^{(k-1)} - \eta \nabla_W Loss\left(W^{(k-1)}\right)^T$$

$$W^{(k)} = W^{(k-1)} + \Delta W^{(k)}$$

Nesterov



$$\Delta W^{(k)} = \beta \Delta W^{(k-1)} - \eta \nabla_W Loss\left(W^{(k-1)} + \beta \Delta W^{(k-1)}\right)^T$$

$$W^{(k)} = W^{(k-1)} + \Delta W^{(k)}$$

Summary

Gradient descent can miss obvious answers

- And this may be a *good* thing

Vanilla gradient descent may be too slow or unstable due to the differences between the dimensions

Second order methods can normalize the variation across dimensions, but are complex

Adaptive or decaying learning rates can improve convergence

Methods that decouple the dimensions can improve convergence

Momentum methods which emphasize directions of steady improvement are demonstrably superior to other methods