Gradient Descent

Sargur Srihari
srihari@cedar.buffalo.edu

Topics

- Simple Gradient Descent/Ascent
- Difficulties with Simple Gradient Descent
- Line Search
- Brent's Method
- Conjugate Gradient Descent
- Weight vectors to minimize error
- Stochastic Gradient Descent

Gradient-Based Optimization

- Most ML algorithms involve optimization
- Minimize/maximize a function $f(\boldsymbol{x})$ by altering $\boldsymbol{x}$
  - Usually stated a minimization
  - Maximization accomplished by minimizing $-f(\boldsymbol{x})$
- $f(\boldsymbol{x})$ referred to as objective or criterion
  - In minimization also referred to as loss function cost, or error
  - Example is linear least squares $\boxed{f(x) = \frac{1}{2} || Ax - b ||^2}$
  - Denote optimum value by $\boldsymbol{x}^* = \operatorname{argmin} f(\boldsymbol{x})$

Calculus in Optimization

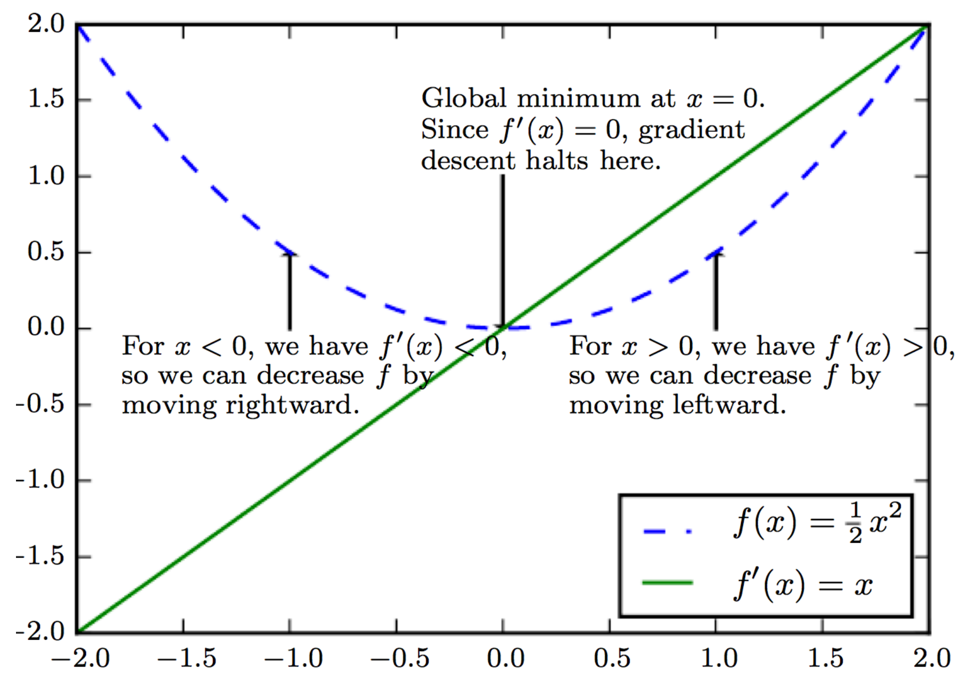
- Consider function $y=f(x)$, x, y real nos.
  - Derivative of function denoted: $f'(x)$ or as dy/dx
    - Derivative $f'(x)$ gives the slope of $f(x)$ at point x
    - It specifies how to scale a small change in input to obtain a corresponding change in the output:

$$f(x+\varepsilon) \approx f(x) + \varepsilon f'(x)$$

  - It tells how you make a small change in input to make a small improvement in y
  - We know that $f(x - \varepsilon \operatorname{sign}(f'(x)))$ is less than $f(x)$ for small ε. Thus we can reduce $f(x)$ by moving x in small steps with opposite sign of derivative
    - This technique is called *gradient descent* (Cauchy 1847)

Gradient Descent Illustrated

- Given function is $f(x)=½\ x^2$ which has a bowl shape with global minimum at $x=0$
  - Since $f'(x)=x$
    - For $x>0$, $f(x)$ increases with x and $f'(x)>0$
    - For $x<0$, $f(x)$ decreases with x and $f'(x)<0$
- Use $f'(x)$ to follow function downhill
  - Reduce $f(x)$ by going in direction opposite sign of derivative $f'(x)$

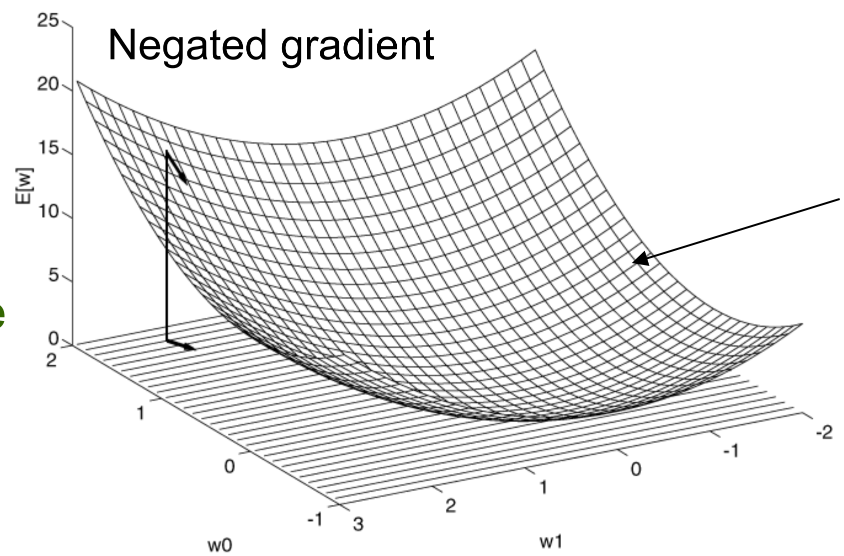


Minimizing with multiple inputs

- We often minimize functions with multiple inputs: $f: \mathbb{R}^n \rightarrow \mathbb{R}$
- For minimization to make sense there must still be only one (scalar) output

Application in ML: Minimize Error

Direction in w_0-w_1 plane producing steepest descent



Error surface shows desirability of every weight vector, a parabola with a single global minimum

- Gradient descent search determines a weight vector w that minimizes $E(\mathrm{w})$ by
  - Starting with an arbitrary initial weight vector
  - Repeatedly modifying it in small steps
  - At each step, weight vector is modified in the direction that produces the steepest descent along the error surface

Definition of Gradient Vector

- The *Gradient (derivative)* of E with respect to each component of the vector w

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$$

- Notice $\nabla E[\vec{w}]$ is a vector of partial derivatives
- Specifies the direction that produces steepest increase in E
- Negative of this vector specifies direction of steepest decrease

Directional Derivative

- Directional derivative in direction u (a unit vector) is the slope of function f in direction u
  - This evaluates to $\boxed{\mathrm{u}^T \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right)}$
- To minimize f find direction in which f decreases the fastest
  - Do this using $\boxed{\min_{\mathrm{u},\mathrm{u}^{\mathrm{T}}\mathrm{u}=1} \mathrm{u}^T \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right) = \min_{\mathrm{u},\mathrm{u}^{\mathrm{T}}\mathrm{u}=1} \left|\left|\mathrm{u}\right|\right|_2 \left|\left|\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right)\right|\right|_2 \cos\theta}$
    - where θ is angle between u and the gradient
    - Substitute $||\mathrm{u}||_2=1$ and ignore factors that not depend on u this simplifies to $\min_u \cos\theta$
    - This is minimized when u points in direction opposite to gradient
    - In other words, the *gradient points directly uphill, and the negative gradient points directly downhill*

Gradient Descent Rule

$$\boxed{\vec{w} \leftarrow \vec{w} + \overrightarrow{\Delta w}}$$

– where

$$\overrightarrow{\Delta w} = -\eta \nabla E[\vec{w}]$$

– η is a positive constant called the learning rate

- Determines step size of gradient descent search

- Component Form of Gradient Descent
  – Can also be written as

$$w_i \leftarrow w_i + \Delta w_i$$

- where

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Method of Gradient Descent

- The gradient points directly uphill, and the negative gradient points directly downhill
- Thus we can decrease f by moving in the direction of the negative gradient
  - This is known as the method of steepest descent or gradient descent
- Steepest descent proposes a new point

$$\boldsymbol{x}' = \boldsymbol{x} - \eta \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right)$$

  - where ε is the learning rate, a positive scalar. Set to a small constant.

Simple Gradient Descent

Procedure Gradient-Descent (

$\boldsymbol{\theta}^1$ //Initial starting point

f //Function to be minimized

δ //Convergence threshold

)

$t \leftarrow 1$

do

$\quad \boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta \nabla f\left(\boldsymbol{\theta}^t\right)$

$\quad t \leftarrow t+1$

while $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t-1}\| > \delta$

return $\left(\boldsymbol{\theta}^t\right)$

Intuition

Taylor's expansion of function $f(\theta)$ in the neighborhood of θ^t is $f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}^t) + (\boldsymbol{\theta} - \boldsymbol{\theta}^t)^T \nabla f(\boldsymbol{\theta}^t)$

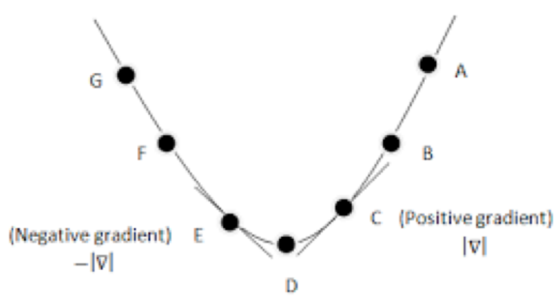
Let $\theta = \theta^{t+1} = \theta^t + h$, thus $f(\boldsymbol{\theta}^{t+1}) \approx f(\boldsymbol{\theta}^t) + h \, \nabla f(\boldsymbol{\theta}^t)$

Derivative of $f(\theta^{t+1})$ wrt h is $\nabla f(\boldsymbol{\theta}^t)$

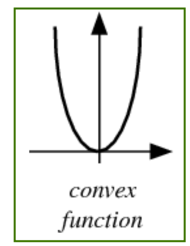
At $h = \nabla f(\boldsymbol{\theta}^t)$ a maximum occurs (since h^2 is positive) and at $h = -\nabla f(\boldsymbol{\theta}^t)$ a minimum occurs.

Alternatively,

The slope $\nabla f(\boldsymbol{\theta}^t)$ points to the direction of steepest ascent. If we take a step η in the opposite direction we decrease the value of f



One-dimensional example



Let $f(\theta) = \theta^2$

This function has minimum at $\theta = 0$ which we want to determine using gradient descent

We have $f'(\theta) = 2\theta$

For gradient descent, we update by $-f'(\theta)$

If $\theta^t > 0$ then $\theta^{t+1} < \theta^t$

If $\theta^t < 0$ then $f'(\theta^t) = 2\theta^t$ is negative, thus $\theta^{t+1} > \theta^t$

Ex: Gradient Descent on Least Squares

- Criterion to minimize $\boxed{f(\mathrm{x}) = \frac{1}{2} \| A\mathrm{x} - \mathrm{b} \|^2}$
  - Least squares regression $\boxed{E_D(\mathrm{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{ t_n - \mathrm{w}^T \phi(x_n) \right\}^2}$
- The gradient is

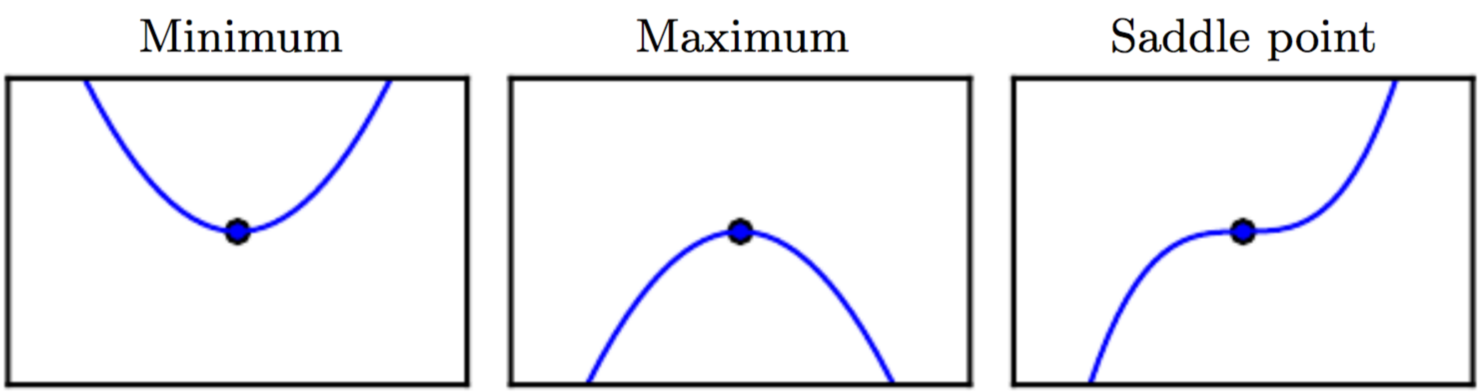
$$\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right) = A^T \left(A\boldsymbol{x} - b\right) = A^T A\boldsymbol{x} - A^T \boldsymbol{b}$$

- Gradient Descent algorithm is
  1. Set step size ε, tolerance δ to small, positive nos.
  2. *while* $\| A^T A\boldsymbol{x} - A^T \boldsymbol{b} \|_2 > \delta$ do

     $$\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta\left(A^T A\boldsymbol{x} - A^T \boldsymbol{b}\right)$$
  3. *end while*

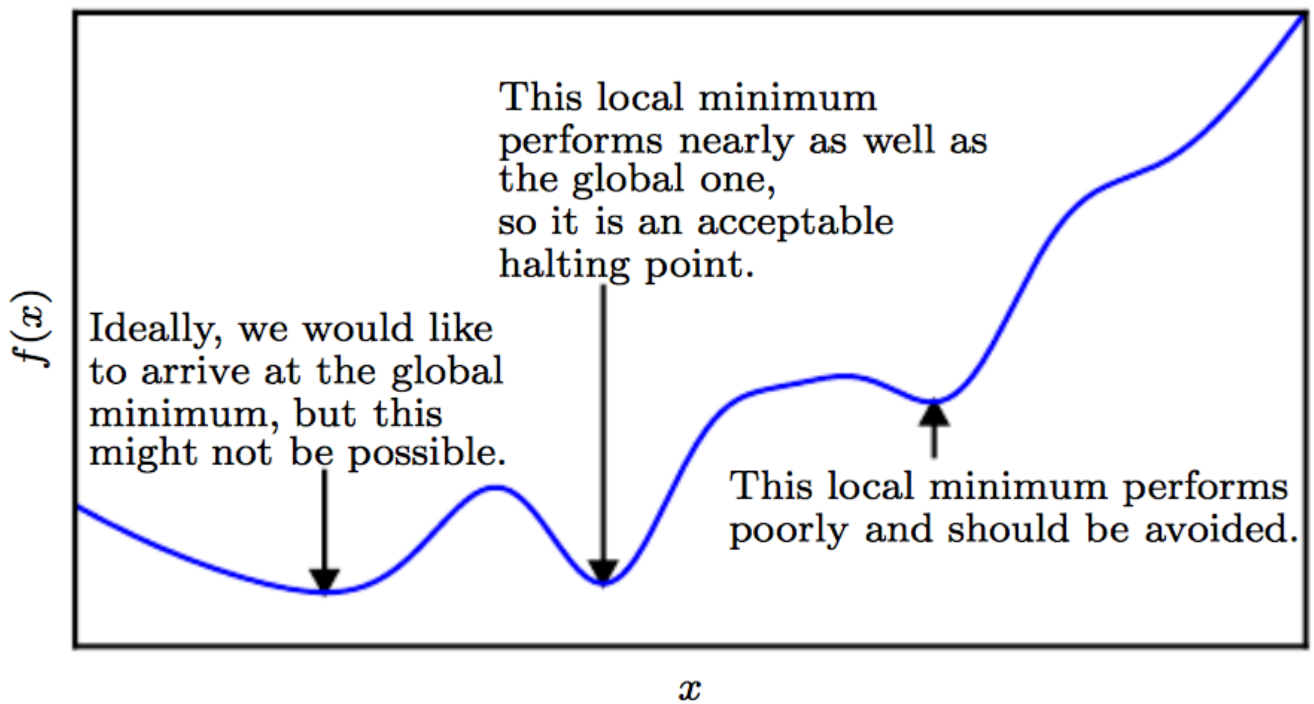
Stationary points, Local Optima

- When $f'(x)=0$ derivative provides no information about direction of move
- Points where $f'(x)=0$ are known as *stationary* or critical points
  - Local minimum/maximum: a point where $f(x)$ lower/ higher than all its neighbors
  - Saddle Points: neither maxima nor minima



Presence of multiple minima

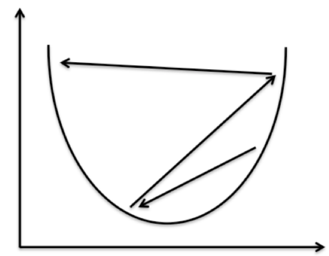
- Optimization algorithms may fail to find global minimum
- Generally accept such solutions



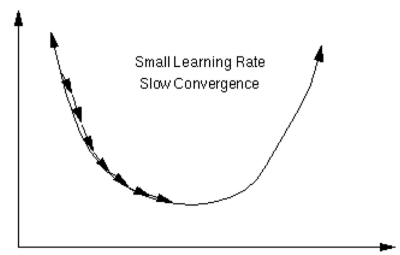
Difficulties with Simple Gradient Descent

- Performance depends on choice of learning rate η

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^{t} - \eta \nabla f\left(\boldsymbol{\theta}^{t}\right)$$



- Large learning rate
  - "Overshoots"



- Small learning rate
  - Extremely slow
- Solution
  - Start with large η and settle on optimal value
  - Need a schedule for shrinking η

Convergence of Steepest Descent

- Steepest descent converges when every element of the gradient is zero
  - In practice, very close to zero
- We may be able to avoid iterative algorithm and jump to the critical point by solving the equation $\boxed{\nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right)=0}$ for x

Choosing η : Line Search

- We can choose η in several different ways
- Popular approach: set η to a small constant
- Another approach is called *line search*:
- Evaluate $\boxed{f(\boldsymbol{x}-\eta\nabla_{\boldsymbol{x}}f(\boldsymbol{x}))}$ for several values of η and choose the one that results in smallest objective function value

Line Search (with ascent)

- In Gradient Ascent, we increase f by moving in the direction of the gradient

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^{t} + \eta \nabla f\left(\boldsymbol{\theta}^{t}\right)$$

- Intuition of Line Search: Adaptive choice of step size η at each step
  - Choose direction to ascend and continue in direction until we start to descend
  - Define "line" in direction of gradient

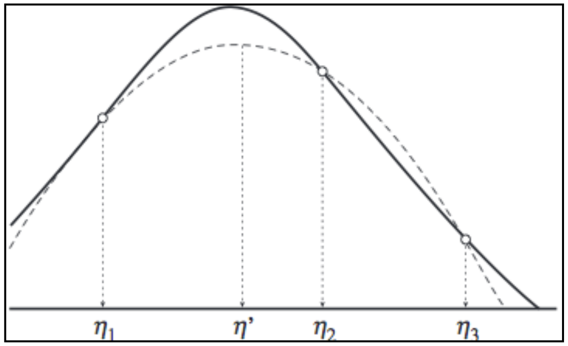
$$g\left(\eta\right) = \vec{\boldsymbol{\theta}}^{t} + \eta \nabla f\left(\boldsymbol{\theta}^{t}\right)$$

    - Note that this is a linear function of η and hence it is a "line"

Line Search: determining η

- Given $g(\eta) = \vec{\boldsymbol{\theta}}^t + \eta \nabla f\left(\boldsymbol{\theta}^t\right)$
- Find three points $\eta_1 < \eta_2 < \eta_3$ so that $f(g(\eta_2))$ is larger than at both $f(g(\eta_1))$ and $f(g(\eta_3))$
  – We say that $\eta_1 < \eta_2 < \eta_3$ *bracket* a maximum
- If we find an η' so that we can find a new tighter bracket $\eta_1 < \eta' < \eta_2$
  - To find η' use binary search

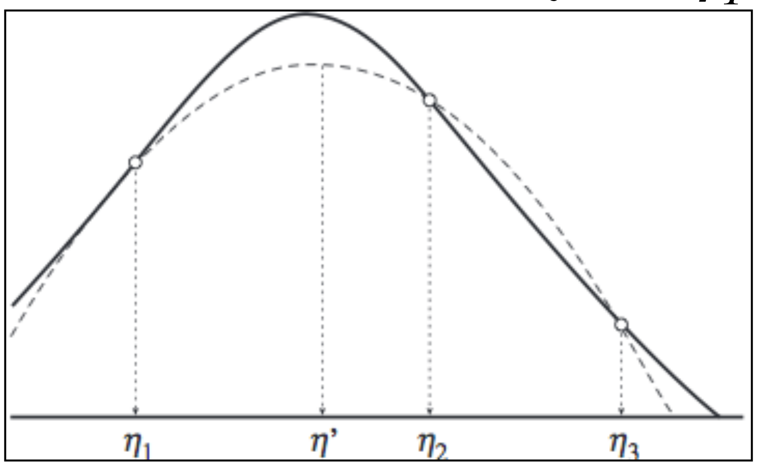
    Choose $\eta' = (\eta_1 + \eta_3)/2$



- Method ensures that new bracket is half of the old one

Brent's Method

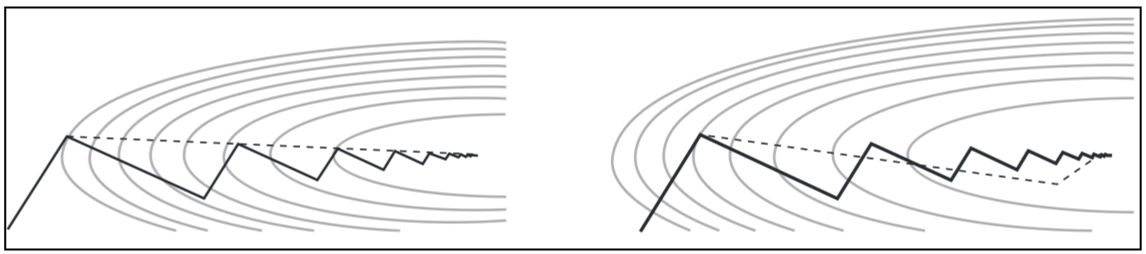
- Faster than line search
- Use quadratic function
  - Based on values of f at η_1, η_2, η_3



- Solid line is $f(g(\eta))$
- Three points bracket its maximum
- Dashed line shows quadratic fit

Conjugate Gradient Descent

- In simple gradient ascent:
  - two consecutive gradients are orthogonal:
  - $\nabla f(\boldsymbol{\theta}^{t+1})$ is orthogonal to $\nabla f(\boldsymbol{\theta}^{t})$
  - Progress is zig-zag: progress to maximum slows down



Quadratic: $f(x,y)=-(x^2+10y^2)$ Exponential: $f(x,y)=\exp[-(x^2+10y^2)]$

- Solution:
  - Remember past directions and bias new direction $\boldsymbol{h}^t$ as combination of current $\boldsymbol{g}^t$ and past $\boldsymbol{h}^{t-1}$
  - Faster convergence than simple gradient ascent

Sequential (On-line) Learning

- Maximum likelihood solution is

$$w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T t$$

- It is a batch technique
  - Processing entire training set in one go
- It is computationally expensive for large data sets
    - Due to huge N x M Design matrix Φ
- Solution is to use a sequential algorithm where samples are presented one at a time
- Called Stochastic gradient descent

Stochastic Gradient Descent

- Error function $\boxed{E_D(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\, t_n - \mathrm{w}^T\varphi(\mathrm{x}_n)\,\}^2}$ sums over data
  - Denoting $E_D(\mathrm{w})=\Sigma_n E_n$ update parameter vector w using

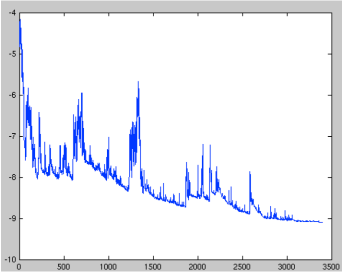
$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla E_n$$

    - Substituting for the derivative

$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} + \eta(t_n - \mathrm{w}^{(\tau)}\phi(\mathrm{x}_n))\phi(\mathrm{x}_n)$$

    - w is initialized to some starting vector $\mathrm{w}^{(0)}$
    - η chosen with care to ensure convergence
- Known as *Least Mean Squares* Algorithm

SGD Performance



Flucutations in objective as gradient steps are taken in mini batches

Most practitioners use SGD for DL

Consists of showing the input vector for a few examples, computing the outputs and the errors, Computing the *average gradient* for those examples, and adjusting the weights accordingly.

Process repeated for many small sets of examples from the training set until the average of the objective function stops decreasing.

Called stochastic because each small set of examples gives a noisy estimate of the average gradient over all examples.

Usually finds a good set of weights quickly compared to elaborate optimization techniques.

After training, performance is measured on a different test set: tests generalization ability of the machine — its ability to produce sensible answers on new inputs never seen during training.