Fundamentals of Machine Learning

OPTIMIZATION

Amit K Roy-Chowdhury

Acknowledgments: Adapted from slides at https://probml.github.io/pml-book/teaching1.html by Prof. Saw Shier Nee

Outline

- Convex Function
- Gradient Descent
- Newton's Method
- Stochastic Gradient Descent
- Constrained Optimization

Optimization

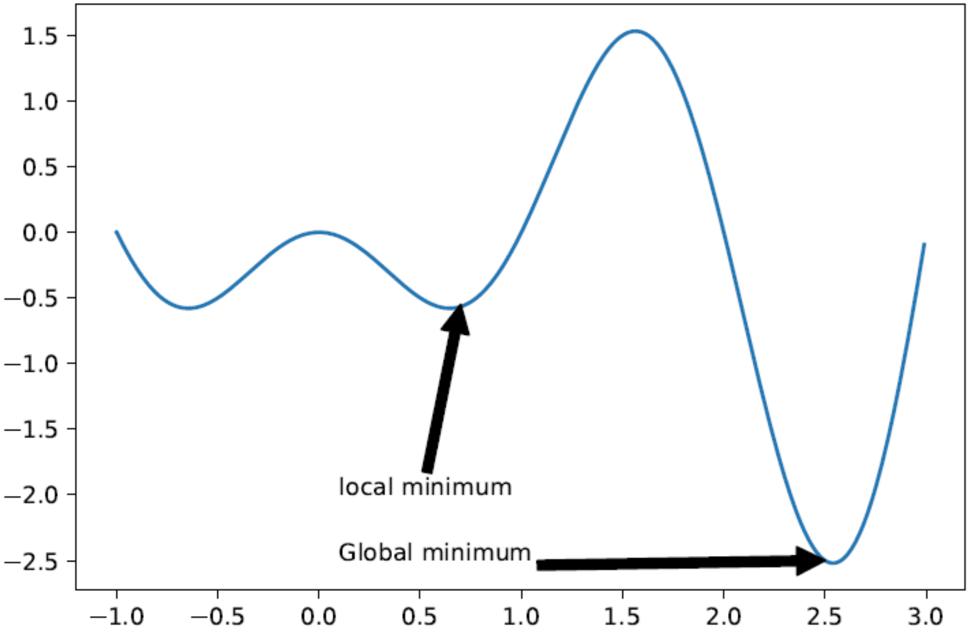
The core problem in machine learning is parameter estimation (aka model fitting).

This requires solving an optimization problem, where we try to find the values for a set of variables, $\theta \in \Theta$, that minimize a scalar-valued loss function or cost function, $\mathcal{L}(\theta)$.

$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \, \mathcal{L}(\boldsymbol{\theta})$$

Global / Local Optimization

$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{\theta})$$



Definitions: Gradient, Hessian, Jacobian

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}.$$

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

$$\mathbf{H}(f(\mathbf{x})) = \mathbf{J}(\nabla f(\mathbf{x})).$$

$\mathbf{f} : \mathbf{R}^n \to \mathbf{R}^m$

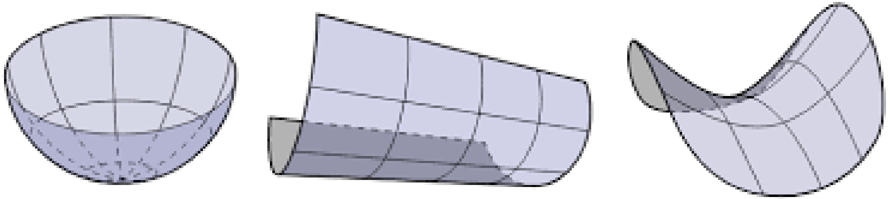
$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Global / Local minima

Let $\boldsymbol{g}(\boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta})$ be the gradient vector,

$\mathbf{H}(\boldsymbol{\theta}) = \nabla^2 \mathcal{L}(\boldsymbol{\theta})$ be the Hessian matrix.

- Necessary condition: If $\boldsymbol{\theta}^*$ is a local minimum, then we must have $\boldsymbol{g}^* = \mathbf{0}$ (i.e., $\boldsymbol{\theta}^*$ must be a **stationary point**), and $\mathbf{H}^*$ must be positive semi-definite.
- Sufficient condition: If $\boldsymbol{g}^* = 0$ and $\mathbf{H}^*$ is positive definite, then $\boldsymbol{\theta}^*$ is a local optimum.



$x^2 + y^2$ (definite) x^2 (semidefinite) $x^2 - y^2$ (indefinite)

Constrained / Unconstrained Optimization

Inequality constraints

Equality constraints

We define the **feasible set** as the subset of the parameter space that satisfies the constraints:

$$\mathcal{C} = \{\boldsymbol{\theta} : g_j(\boldsymbol{\theta}) \leq 0 : j \in \mathcal{I}, h_k(\boldsymbol{\theta}) = 0 : k \in \mathcal{E}\} \subseteq \mathbb{R}^D$$

Our **constrained optimization** problem now becomes

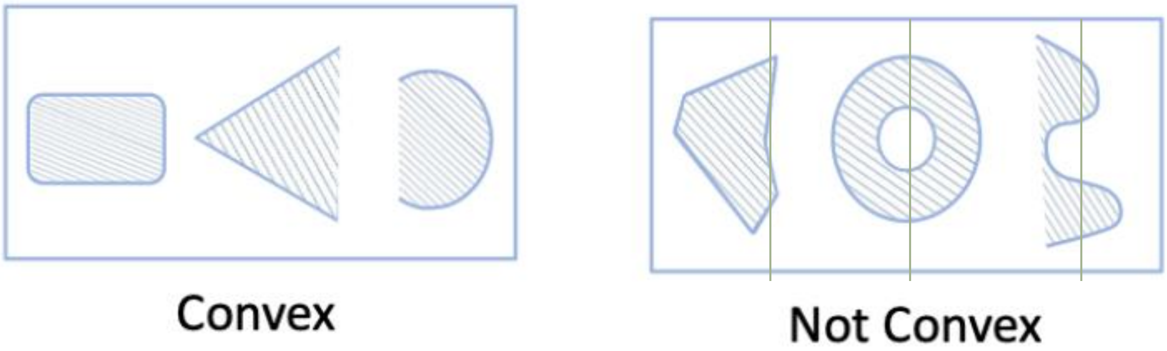
$$\boldsymbol{\theta}^* \in \operatorname*{argmin}_{\boldsymbol{\theta} \in \mathcal{C}} \mathcal{L}(\boldsymbol{\theta})$$

If $\mathcal{C} = \mathbb{R}^D$, it is called **unconstrained optimization**.

If too many constraints, empty feasible sets

A common strategy to solve constrained problem is add penalty to the loss function such as using Lagrangian multiplier.

Convex / Concave Optimization



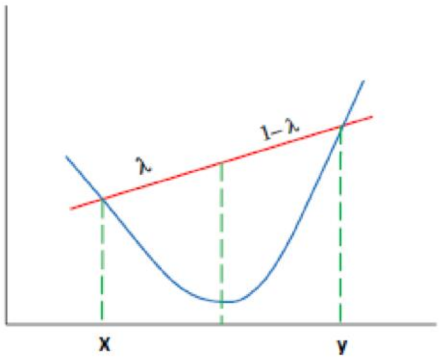
We say $\mathcal{S}$ is a **convex set** if, for any $x, x' \in \mathcal{S}$, we have

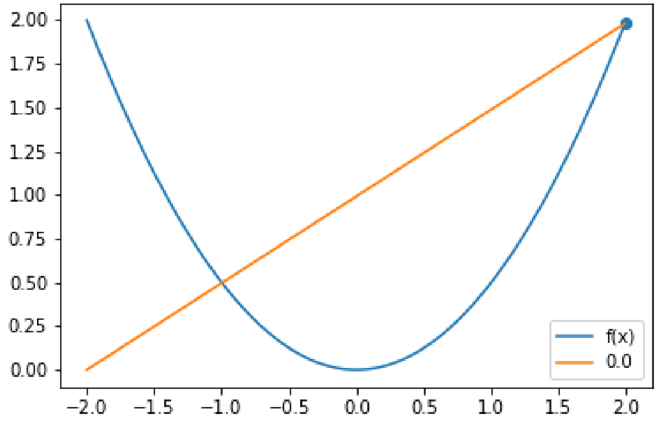
$$\lambda x + (1-\lambda)x' \in \mathcal{S}, \;\; \forall\, \lambda \in [0,1]$$

Convex Function

We say f is a **convex function** if its **epigraph** (the set of points above the function, illustrated in Figure 8.4a) defines a convex set. Equivalently, a function $f(\boldsymbol{x})$ is called convex if it is defined on a convex set and if, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$, and for any $0 \leq \lambda \leq 1$, we have

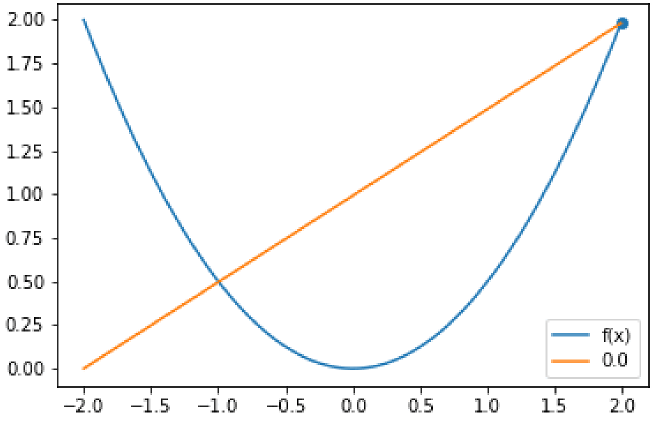
$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}) \tag{8.7}$$



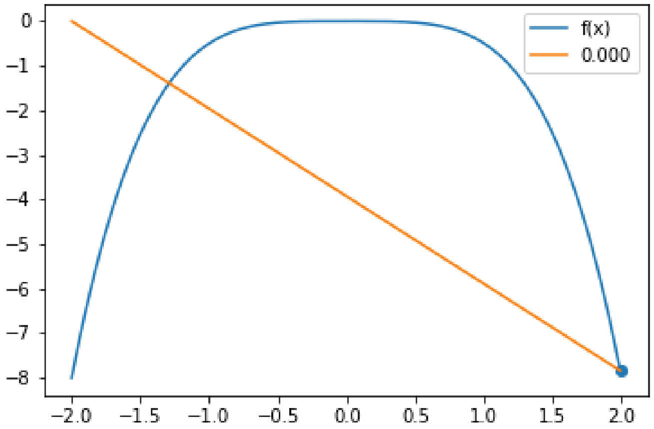


Convex Function

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$$



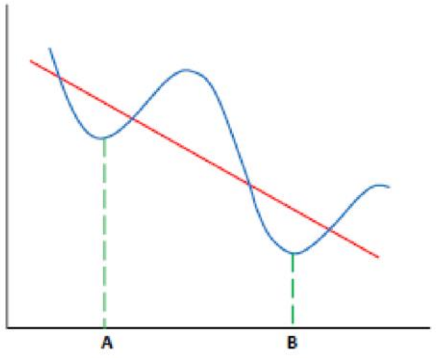
Convex

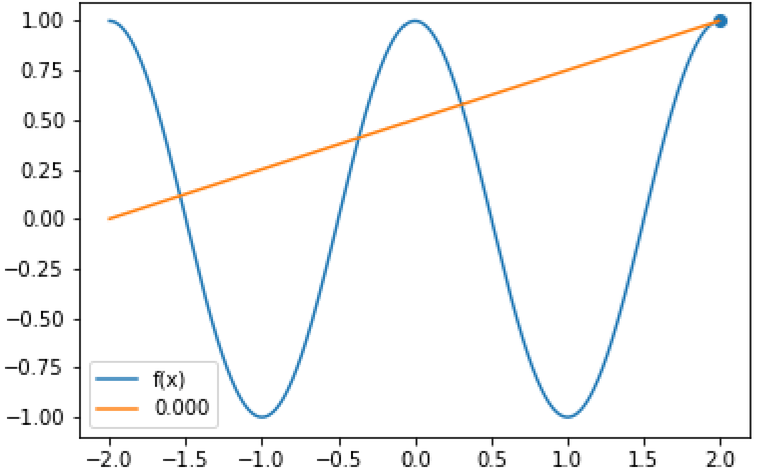


Concave

Convex Function

Neither convex nor concave

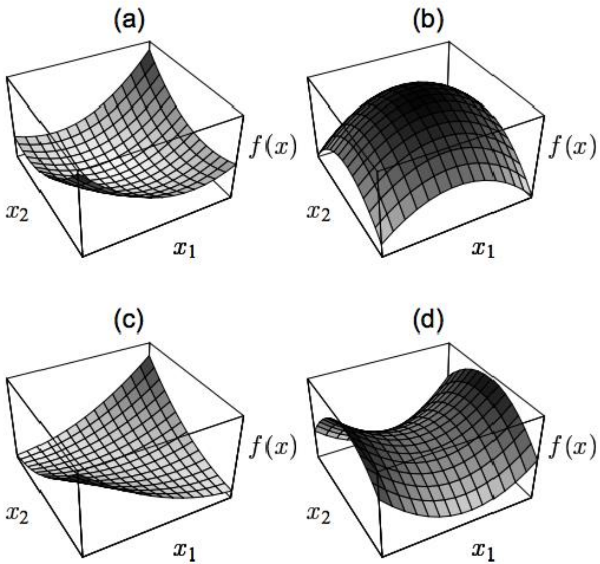




Types of Convex Function

Strictly convex

Strictly concave

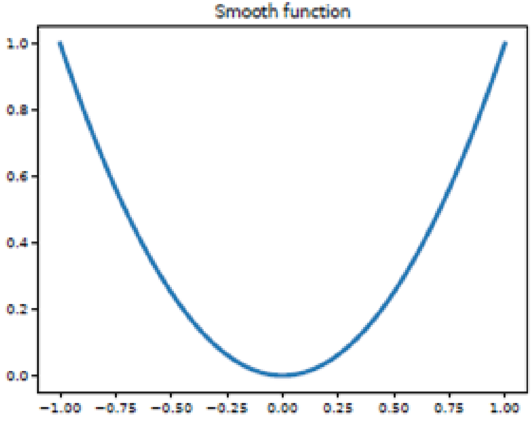


Convex but not strictly

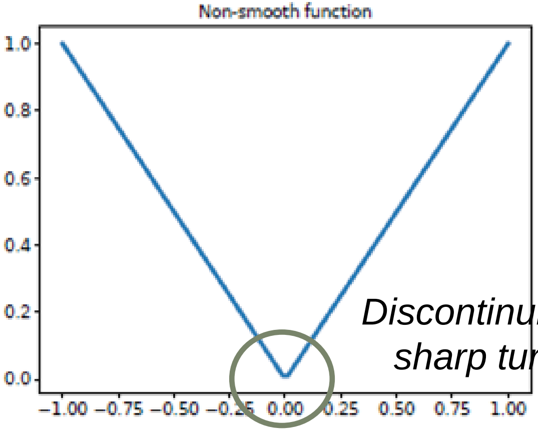
Neither convex nor concave - saddle point

Figure 8.6: The quadratic form $f(x) = x^\top \mathbf{A} x$ in 2d. (a) $\mathbf{A}$ is positive definite, so f is convex. (b) $\mathbf{A}$ is negative definite, so f is concave. (c) $\mathbf{A}$ is positive semidefinite but singular, so f is convex, but not strictly. Notice the valley of constant height in the middle. (d) $\mathbf{A}$ is indefinite, so f is neither convex nor concave. The stationary point in the middle of the surface is a saddle point. From Figure 5 of [She94].

Smooth and Non-Smooth Optimization



(a)



Discontinuity / sharp turn

(b)

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|$$

Lipschitz constant – quantify the degree of smoothness

First Order Method

- Gradient descent
- Step size/ learning rate
- Convergence rate
- Momentum Method

Taylor series: $$f(x+\Delta x) = f(x) + f'(x)\Delta x + \frac{1}{2!}f''(x)\Delta x^2 + \frac{1}{3!}f'''(x)\Delta x^3 + \ldots$$

Gradient Descent

We need $\mathcal{L}(\boldsymbol{\theta} + \eta \boldsymbol{d}) < \mathcal{L}(\boldsymbol{\theta})$

Gradient at current iterate: $\boldsymbol{g}_t \triangleq \nabla \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}_t} = \nabla \mathcal{L}(\boldsymbol{\theta}_t) = \boldsymbol{g}(\boldsymbol{\theta}_t)$

Descent direction: $\boldsymbol{d}^{\mathsf{T}} \boldsymbol{g}_t = ||\boldsymbol{d}||\ ||\boldsymbol{g}_t||\ \cos(\theta) < 0$

$$\text{pick } \boldsymbol{d}_t = -\boldsymbol{g}_t$$

First Order Method

- Gradient descent
- Step size/ learning rate
- Convergence rate
- Momentum Method

$$\theta_{t+1} = \theta_t + \rho_t d_t$$

Updated parameter

Step size

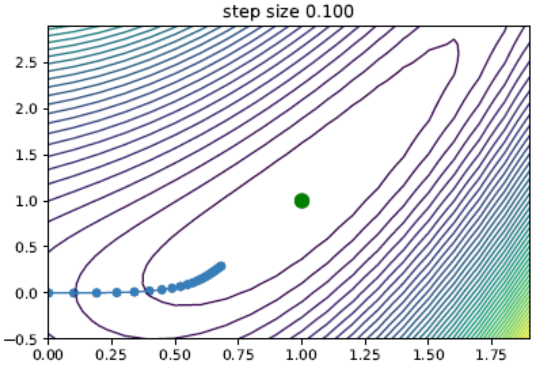
Descent direction

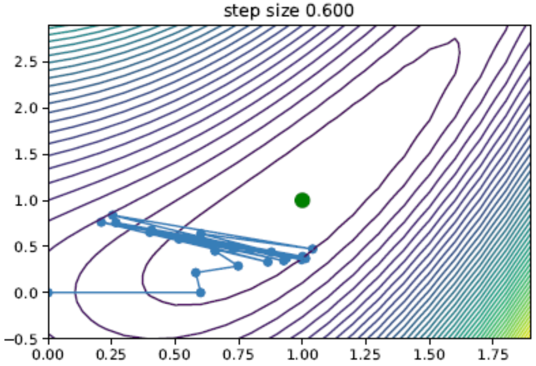
steepest descent will have global convergence iff the step size satisfies

$$\rho < \frac{2}{\lambda_{\max}(\mathbf{A})}$$

$$\mathcal{L}(\theta) = \frac{1}{2}\theta^\top \mathbf{A}\theta + b^\top\theta + c \text{ with } \mathbf{A} \succeq 0.$$

λmax = max eigenvalue





First Order Method

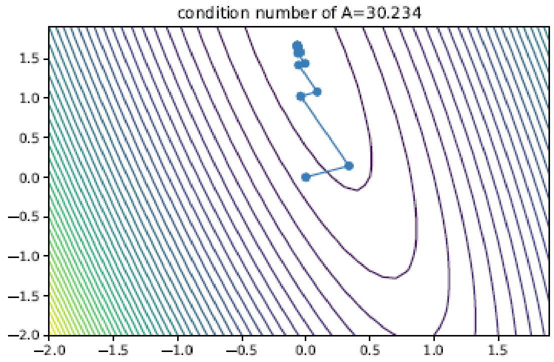
- Gradient descent
- Step size/ learning rate
- Convergence rate, μ
- Momentum Method

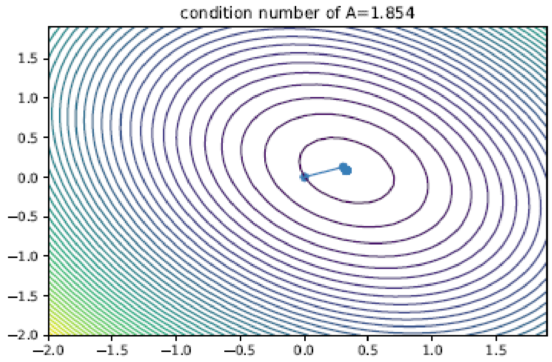
$$\mathcal{L}(\theta) = \frac{1}{2}\theta^\top \mathbf{A}\theta + b^\top\theta + c \text{ with } \mathbf{A} \succeq \mathbf{0}.$$

$$\mu = \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right)^2$$

$$\mu = \left(\frac{\kappa-1}{\kappa+1}\right)^2, \text{ where } \kappa = \frac{\lambda_{\max}}{\lambda_{\min}} \text{ is the condition number of } \mathbf{A}.$$

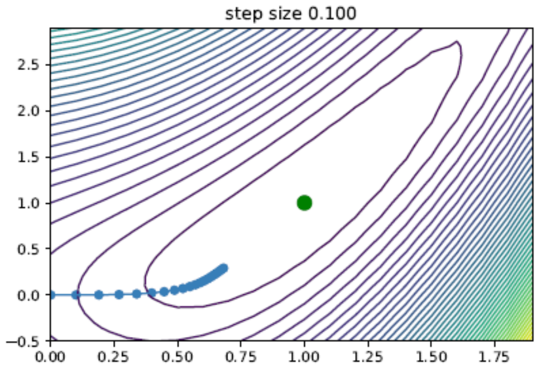
The condition number measures how "skewed" the space is, in the sense of being far from a symmetrical "bowl"





First Order Method

- Gradient descent
- Step size/ learning rate
- Convergence rate
- Momentum Method



Thinking like a ball rolling downward. At flat surface, it rolls down slowly. At sharp region, roll down faster.

momentum

$$m_t = \beta m_{t-1} + g_{t-1}$$

$$\theta_t = \theta_{t-1} - \rho_t m_t$$

(exponentially weighted moving average of past gradients)

Normally β=0.9, if β = 0 - gradient descent

Adaptive Moment Estimation (Adam)

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$s_t = \beta_2 s_{t-1} + (1 - \beta_2) g_t^2$$

$$\beta_1 = 0.9,\ \beta_2 = 0.999 \text{ and } \epsilon = 10^{-6}.$$

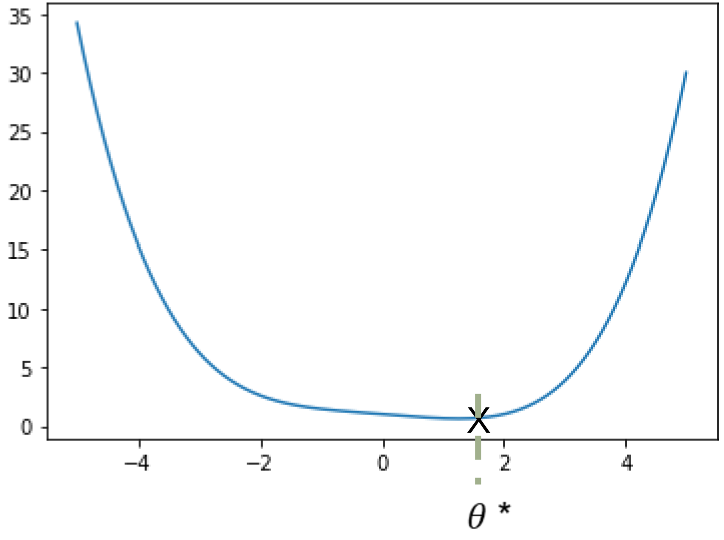
$$\eta_t = 0.001$$

$$\Delta\theta_t = -\eta_t \frac{1}{\sqrt{s_t} + \epsilon} m_t$$

First Order Method – Line Search

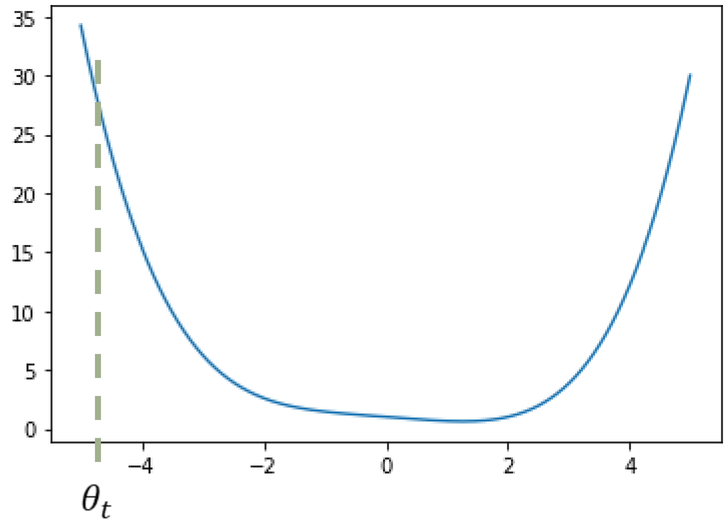
Consider we want to minimize this loss function, $f(\theta)$.

What if we use line search (iterative method to find θ *)



First Order Method – Line Search

Start with random number, θ_t

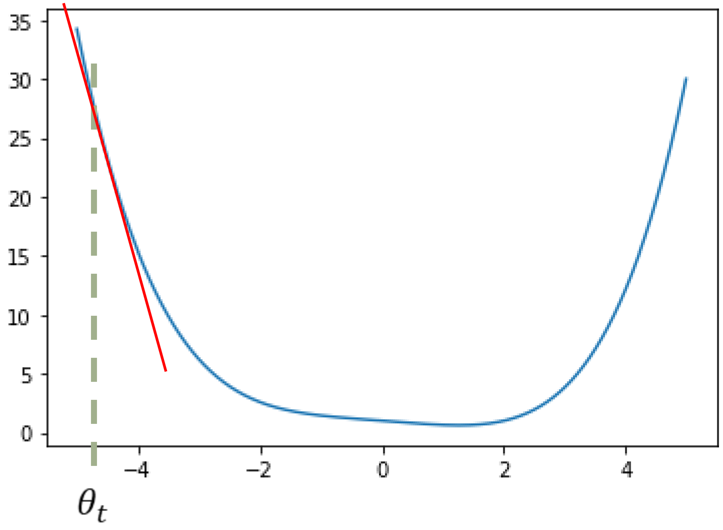


First Order Method – Line Search

Start with random number, θ_t

Compute the gradient at x_k

$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$



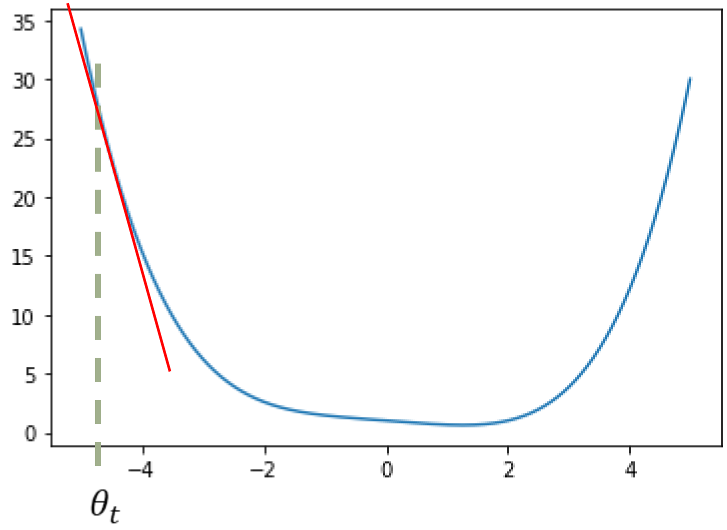
First Order Method – Line Search

Start with random number, θ_k

Compute the gradient at θ_k

$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$

If $f'(\theta_t)$ is negative, move θ_t to the right



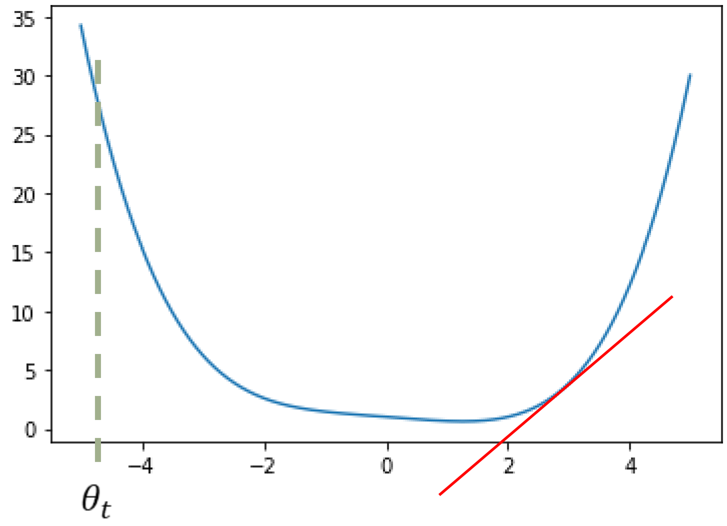
First Order Method – Line Search

Start with random number, θ_k

Compute the gradient at x_k

$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$

If $f'(\theta_t)$ is negative, move θ_t to the right
If $f'(\theta_t)$ is positive, move θ_t to the left



First Order Method – Line Search

Start with random number,θ

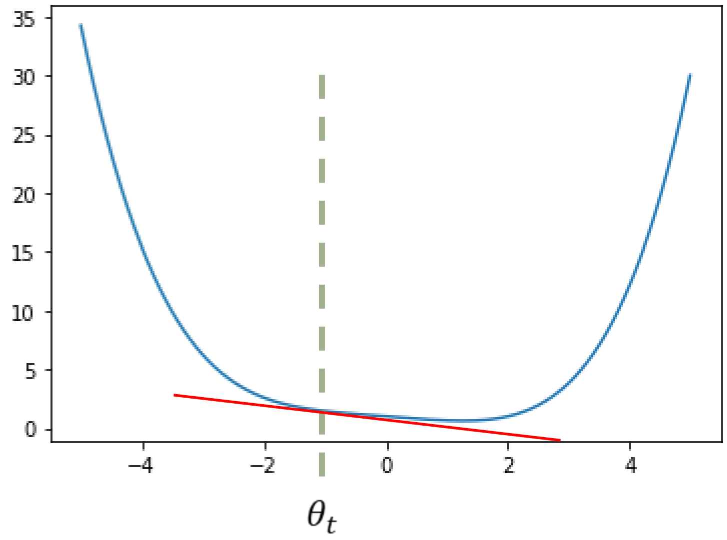
Compute the gradient at x_k

$$f'(\theta_t) = \frac{f(\theta) - f(\theta_t)}{\theta - \theta_t}$$

If $f'(\theta_t)$ is negative, move x_k to the right
If $f'(\theta_t)$ is positive, move x_k to the left

$$\theta_{t+1} = \theta_t - \alpha f'(\theta_t)$$

Large step size, α will overshoot
Small step size, α will be very slow



Second Order Method – Newton Method

Approximate with non linear graph

Compute the second derivative at θ_k

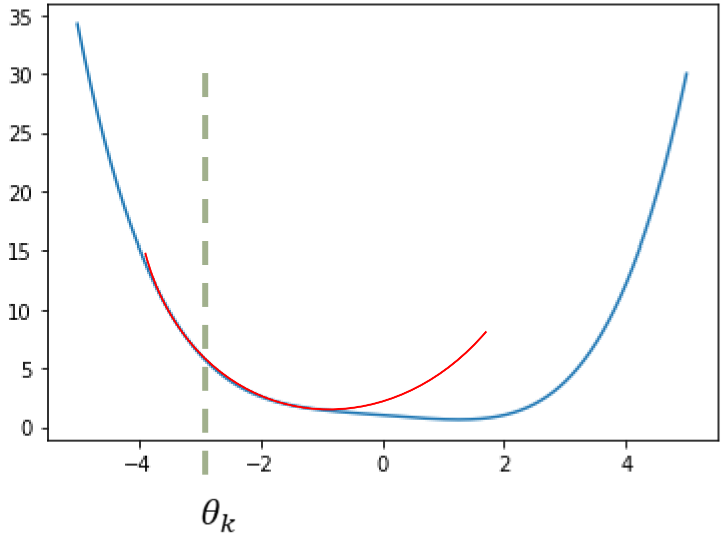
$$f''(\theta_k) = \frac{f'(\theta) - f'(\theta_k)}{\theta - \theta_k}$$

$$\theta_{t+1} = \theta_k - \alpha f'(\theta_k)$$

For Newton's method, the update formula is

$$\theta_{t+1} = \theta_k - \alpha \frac{1}{f''(\theta_k)} f'(\theta_k)$$

- Faster convergence



Second Order Method – Newton Method

Higher dimension

$$\theta_{t+1} = \theta_t - \rho_t \mathbf{H}_t^{-1} g_t$$

where

$$\mathbf{H}_t \triangleq \nabla^2 \mathcal{L}(\theta)|_{\boldsymbol{\theta}_t} = \nabla^2 \mathcal{L}(\theta_t) = \mathbf{H}(\theta_t)$$

H = Hessian matrix
ρ = step size
g_t = gradient

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

Consider a quadratic approximation

$$\mathcal{L}_{\text{quad}}(\theta) = \mathcal{L}(\theta_t) + g_t^{\mathsf{T}}(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^{\mathsf{T}}\mathbf{H}_t(\theta - \theta_t)$$

The minimum of $\mathcal{L}_{\text{quad}}$ is at

$$\theta = \theta_t - \mathbf{H}_t^{-1} g_t$$

Gradient Descent vs Newton's method

We need $\mathcal{L}(\theta + \eta d) < \mathcal{L}(\theta)$

Gradient at current iterate: $g_t \triangleq \nabla \mathcal{L}(\theta)|_{\theta_t} = \nabla \mathcal{L}(\theta_t) = g(\theta_t)$

Gradient Descent: $d^{\mathsf{T}} g_t = ||d||\ ||g_t||\ \cos(\theta) < 0$

$$\text{pick } d_t = -g_t$$

Consider only first two terms of Taylor series

Newton's method: $d_t = -\mathrm{H}_t^{-1} g_t$

Consider only first three term of Taylor series

Stochastic Gradient Descent

Stochastic Optimization: $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{z})}\left[\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{z})\right]$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{z}_t) = \boldsymbol{\theta}_t - \eta_t \boldsymbol{g}_t$$

(distribution of random variable is independent of parameter we are optimizing over)

Consider loss function: $\mathcal{L}(\boldsymbol{\theta}_t) = \frac{1}{N}\sum_{n=1}^{N} \ell(\boldsymbol{y}_n, f(\boldsymbol{x}_n; \boldsymbol{\theta}_t)) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}_n(\boldsymbol{\theta}_t)$

$$\boldsymbol{g}_t = \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}} \mathcal{L}_n(\boldsymbol{\theta}_t) = \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{y}_n, f(\boldsymbol{x}_n; \boldsymbol{\theta}_t))$$

This requires summing over all N training examples, and thus can be slow if N is large.

Minibatch: $\boldsymbol{g}_t \approx \frac{1}{|\mathcal{B}_t|}\sum_{n \in \mathcal{B}_t} \nabla_{\boldsymbol{\theta}} \mathcal{L}_n(\boldsymbol{\theta}_t) = \frac{1}{|\mathcal{B}_t|}\sum_{n \in \mathcal{B}_t} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{y}_n, f(\boldsymbol{x}_n; \boldsymbol{\theta}_t))$

Minibatch sampling, training epochs

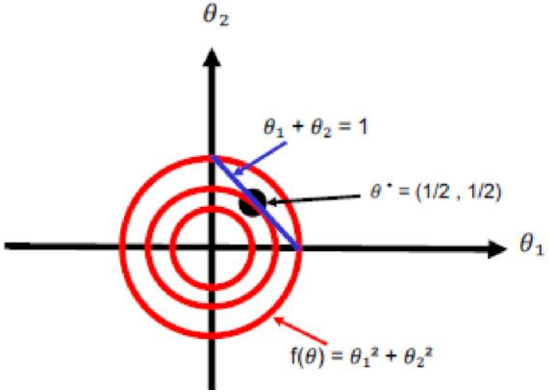
where $\mathcal{B}_t$ is a set of randomly chosen examples to use at iteration t.

Constrained Optimization

$$\theta^* = \arg\min_{\theta \in \mathcal{C}} \mathcal{L}(\theta)$$

where the feasible set, or constraint set, is

$$\mathcal{C} = \{\theta \in \mathbb{R}^D : h_i(\theta) = 0, i \in \mathcal{E},\ g_j(\theta) \leq 0, j \in \mathcal{I}\}$$



Taylor series: $$h(\theta + \epsilon) \approx h(\theta) + \epsilon^\top \nabla h(\theta)$$

Since both θ and $\theta + \epsilon$ are on the constraint surface, we must have $h(\theta) = h(\theta + \epsilon)$ and hence $\epsilon^\top \nabla h(\theta) \approx 0$. Since ϵ is parallel to the constraint surface, $\nabla h(\theta)$ must be perpendicular to it.

$\nabla \mathcal{L}(\theta)$ is also orthogonal to the constraint surface $\quad \nabla \mathcal{L}(\theta^*) = \lambda^* \nabla h(\theta^*)$

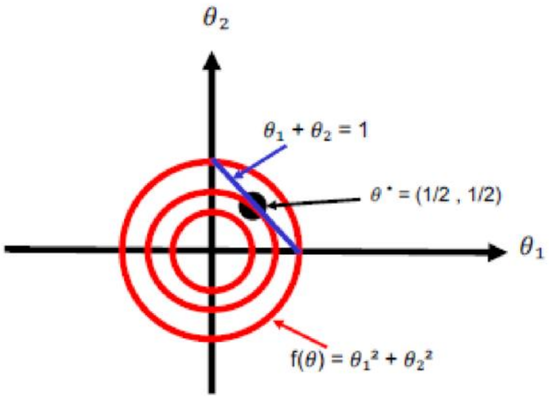
Lagrangian: $$L(\theta, \lambda) \triangleq \mathcal{L}(\theta) + \lambda h(\theta)$$

$$\nabla_{\theta,\lambda} L(\theta, \lambda) = 0 \iff \lambda \nabla_\theta h(\theta) = \nabla \mathcal{L}(\theta),\ h(\theta) = 0$$

Constrained Optimization

Lagrangian: $L(\theta, \lambda) \triangleq \mathcal{L}(\theta) + \lambda h(\theta)$

$$\nabla_{\boldsymbol{\theta},\lambda} L(\theta, \lambda) = \mathbf{0} \iff \lambda \nabla_{\boldsymbol{\theta}} h(\theta) = \nabla \mathcal{L}(\theta),\ h(\theta) = 0$$



$$L(\theta_1, \theta_2, \lambda) = \theta_1^2 + \theta_2^2 + \lambda(\theta_1 + \theta_2 - 1)$$

$$\frac{\partial}{\partial \theta_1} L(\theta_1, \theta_2, \lambda) = 2\theta_1 + \lambda = 0$$

$$\frac{\partial}{\partial \theta_2} L(\theta_1, \theta_2, \lambda) = 2\theta_2 + \lambda = 0$$

$$\frac{\partial}{\partial \lambda} L(\theta_1, \theta_2, \lambda) = \theta_1 + \theta_2 - 1 = 0$$

$2\theta_1 = -\lambda = 2\theta_2$, so $\theta_1 = \theta_2$.

$2\theta_1 = 1$

$\theta^* = (0.5, 0.5)$