CEE 616: Probabilistic Machine Learning
Foundations: Optimization

Jimi Oke

UMass Amherst
College of Engineering

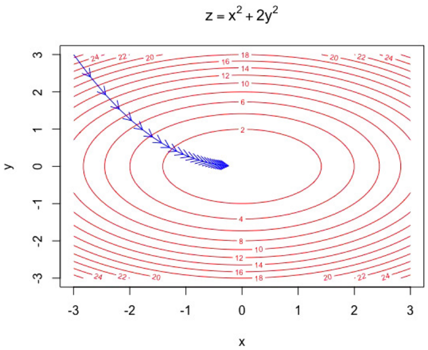
Thu, Sep 18, 2025

Outline

1. Inroduction
2. First-order methods
3. Second-order methods
4. Application: MLE
5. Constrained optimization
6. Outlook

Optimization

Optimization is the body of mathematics that deals with the theory and algorithms for characterizing the maximum/minimum values of functions.



We will consider two widely-used approaches in machine learning:

- First-order methods (e.g. gradient descent)
- Second-order methods (e.g. Newton's method)

Definitions

- Minimum of convex function

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta) : \frac{d\mathcal{L}(\theta^*)}{d\theta} = 0 \tag{1}$$

- Derivative of a function:

$$\mathcal{L}'(\theta) = \frac{d\mathcal{L}(\theta)}{d\theta} \tag{2}$$

- Gradient of a function (vector of partial derivatives):

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots \\ \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_n} \end{bmatrix} \tag{3}$$

Further definitions

- Second-order derivative:

$$\mathcal{L}''(\theta) = \frac{d^2\mathcal{L}(\theta)}{d\theta^2} \tag{4}$$

- Hessian (matrix of partial second derivatives):

$$\boldsymbol{H}_{\boldsymbol{\theta}}(\mathcal{L}(\boldsymbol{\theta})) = \nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial{\theta_1}^2} & \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_1\partial\theta_2} & \cdots & \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_1\partial\theta_n} \\ \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_2\partial\theta_1} & \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial{\theta_2}^2} & \cdots & \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_2\partial\theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_n\partial\theta_1} & \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_n\partial\theta_2} & \cdots & \frac{\partial^2\mathcal{L}(\boldsymbol{\theta})}{\partial{\theta_n}^2} \end{bmatrix} \tag{5}$$

Optimality conditions

For continuous, twice-differentiable functions:

- If $\boldsymbol{\theta}^*$ is a local minimum, then $\boldsymbol{g}^* = \mathbf{0}$ and $\boldsymbol{H}^*$ must be psd (necessary condition)
- If $\boldsymbol{g}^* = \mathbf{0}$ and $\boldsymbol{H}^*$ is pd, then $\boldsymbol{\theta}^*$ is a local optimum (suficient condition)

General approach

- Begin with an initial value $\boldsymbol{\theta}_0$
- At each iteration t, update $\boldsymbol{\theta}_{t+1}$
- Terminate when $\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t) = \epsilon$

Convex optimization

If the objective is convex and defined over a convex set, then every local minimum is a global minimum. pe

- Convex set: For any $\boldsymbol{x}$, $\boldsymbol{x}' \in \mathcal{S}$:

$$\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{x}' \in \mathcal{S}, \quad \forall \lambda \in [0,1] \tag{6}$$

- Convex function: For any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$ and for any $0 \leq \lambda \leq 1$:

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}) \tag{7}$$

Quadratic form

Given $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}$:

- f is convex if $\boldsymbol{A}$ is psd
- f is strictly convex if $\boldsymbol{A}$ is pd

Subgradients

Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{g} \in \mathbb{R}^n$ is a subgradient of f at $\boldsymbol{x} \in \mathrm{dom}(f)$ if:

$$f(\boldsymbol{z}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^\top(\boldsymbol{z} - \boldsymbol{x}) \quad \forall \boldsymbol{z} \in \mathrm{dom}(f) \tag{8}$$

- The set of subgradients is called the **subdifferential**: $\partial f(\boldsymbol{x})$
- f is subdifferentiable at $\boldsymbol{x}$ if it has at least one subgradient at that point

Subdifferential of ReLU

The rectified linear unit function (ReLU) is given by:

$$\mathrm{ReLU}(z) = \max(0, z), \quad z \in \mathbb{R} \tag{9}$$

Its subdifferential is:

$$\partial \mathrm{ReLU}(z) = \begin{cases} 0 & \text{if } z < 0 \\ [0, 1] & \text{if } z = 0 \\ 1 & \text{if } z > 0 \end{cases} \tag{10}$$

First-order methods

These methods only require first-order derivatives in order to compute an update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \rho_t \boldsymbol{d}_t \tag{11}$$

where:

- ρ_t: step size/learning rate
- $\boldsymbol{d}_t$: descent direction (e.g. negative gradient)
- t: index of iteration

The classic first-order method is **gradient descent**

Descent direction

A vector $\boldsymbol{d}$ is considered a descent direction if a nonzero step size ρ moved in its direction yields a decrease in the objective:

$$\mathcal{L}(\boldsymbol{\theta} + \rho \boldsymbol{d}) < \mathcal{L}(\boldsymbol{\theta}) \quad \forall 0 < \rho < \rho_{\text{max}} \tag{12}$$

Steepest descent

The direction of steepest descent is given by the negative gradient:

$$-\boldsymbol{g}_t = -\nabla \mathcal{L}(\boldsymbol{\theta}_t) \tag{13}$$

Step size

The choice of step size impacts the convergence of an optimization routine. It determines how far along in the descent direction the variable is updated at each step.

- **Constant step size**: ρ is fixed throughout
- **Adaptive step size**: ρ_t is modified at each iteration to satisfy certain conditions (e.g. line search methods)
- Sequence $\{\rho_t\}$ in an optimization algorithm is known as the learning rate schedule

Example: Gradient descent

Given the function

$$f(x) = (x-1)^4 - 3x + 4$$

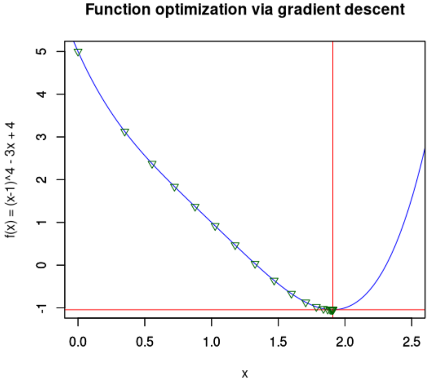
find the optimal point $(x^*, f(x^*))$.

First, we find the **gradient** and define the **update** step:

$$\begin{aligned} f'(x) &= 4(x-1)^3 - 3 \\ x_{k+1} &= x_k - \lambda \left[4(x_k - 1)^3 - 3\right] \end{aligned}$$

We choose $\lambda = 0.05$ and a random starting point between 0 and 1.

Example: Gradient descent (cont.)



Using the `gradient-descent.ipynb` notebook, we find the optimal point as $(\mathbf{1.908, -1.044})$. How does the learning rate impact the solution process?

Exact line search

Finds optimal ρ_t by:

$$\rho_t^* = \underset{\rho>0}{\arg\min}\, \phi_t(\rho) = \underset{\rho>0}{\arg\min}\, \mathcal{L}(\boldsymbol{\theta}_t + \rho \boldsymbol{d}_t) \tag{14}$$

If $\mathcal{L}(\boldsymbol{\theta})$ is quadratic, then we can write:

$$\phi_t(\rho) = \mathcal{L}(\boldsymbol{\theta}_t + \rho \boldsymbol{d}_t) = \frac{1}{2}(\boldsymbol{\theta} + \rho \boldsymbol{d})^\top \boldsymbol{A}(\boldsymbol{\theta} + \rho \boldsymbol{d}) + \boldsymbol{b}^\top(\boldsymbol{\theta} + \rho \boldsymbol{d}) + c \tag{15}$$

Solving for $\phi_t'(\rho) = 0$ gives the optimal update as:

$$\rho_t^* = -\frac{\boldsymbol{d}^\top(\boldsymbol{A}\boldsymbol{\theta}_t + \boldsymbol{b})}{\boldsymbol{d}_t^\top \boldsymbol{A}\boldsymbol{d}_t} \tag{16}$$

This approach introduces additional computational expense

Armijo backtracking

Finds a suitable ρ_t^* by iteratively shrinking ρ_t by $\beta \in (0, 1)$, i.e.

$$\rho_t^{k+1} = \beta \rho_t^k \tag{17}$$

until the **Armijo-Goldstein** condition is satisfied:

$$\mathcal{L}(\boldsymbol{\theta}_t + \rho \boldsymbol{d}_t) \leq \mathcal{L}(\boldsymbol{\theta}_t) + c\rho \boldsymbol{d}_t^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) \tag{18}$$

where $c \in [0, 1]$ is typically chosen as 10^{-4}

Momentum

Momentum methods are used to improve convergence. The momentum update is given by:

$$\boldsymbol{m}_{t+1} = \beta \boldsymbol{m}_t + \boldsymbol{g}_t \tag{19}$$

where $\beta \in (0, 1)$.
The momentum can also be expressed as an exponentially weighted average of past gradients:

$$\boldsymbol{m}_{t+1} = \sum_{\tau=0}^{t} \beta^{\tau} \boldsymbol{g}_{t-\tau} \tag{20}$$

The update is given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \rho_t \boldsymbol{m}_{t+1} \tag{21}$$

Nesterov momentum

In the Nesterov method, the momentum update computes the gradient at the new location, which can speed up convergence:

$$\boldsymbol{m}_{t+1} = \beta \boldsymbol{m}_t - \rho_t \nabla \mathcal{L}(\boldsymbol{\theta}_t + \beta \boldsymbol{m}_t) \tag{22}$$

The update is given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{m}_{t+1} \tag{23}$$

This method is also called **Nesterov accelerated gradient**

Stochastic gradient descent (SGD)

In stochastic optimization, goal is to minimize:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{z})}[\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{z})] \tag{24}$$

where $\boldsymbol{z}_t \sim q$ could be a training sample drawn from a set. Thus, the **stochastic gradient descent** update is given by:

$$\boldsymbol{\theta}_{t+1} =_t -\rho_t \nabla \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{z}_t) = \boldsymbol{\theta}_t - \rho_t \boldsymbol{g}_t \tag{25}$$

- Finite sum: $\mathcal{L}(\boldsymbol{\theta}_t) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}_n(\boldsymbol{\theta}_t)$

Newton's method

Second-order methods incorporate curvature via the second derivative to speed up convergence.
The basic method of this form is Newton's method:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \rho_t \boldsymbol{H}^{-1} \boldsymbol{g}_t \tag{26}$$

where:

$$\boldsymbol{H}_t := \nabla^2 \mathcal{L}(\boldsymbol{\theta}_t) = \boldsymbol{H}(\boldsymbol{\theta}_t) \tag{27}$$

Derivation of Newton's method

Taylor-expand $\mathcal{L}(\boldsymbol{\theta})$ around $\boldsymbol{\theta}_t$:

$$\mathcal{L}_{\text{quad}} = \mathcal{L}(\boldsymbol{\theta}_t) + \boldsymbol{g}_t^\top(\boldsymbol{\theta} - \boldsymbol{\theta}_t) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_t)^\top \boldsymbol{H}_t(\boldsymbol{\theta} - \boldsymbol{\theta}_t) \tag{28}$$

Taking the derivative of $\mathcal{L}_{\text{quad}}$ and setting it to zero to find the minimum, we obtain:

$$\boldsymbol{g}_t + \boldsymbol{H}_t(\boldsymbol{\theta} - \boldsymbol{\theta}_t) = \mathbf{0} \tag{29}$$

$$\boldsymbol{\theta} - \boldsymbol{\theta}_t = -\boldsymbol{H}_t^{-1}\boldsymbol{g}_t \tag{30}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta}_t - \boldsymbol{H}_t^{-1}\boldsymbol{g}_t \tag{31}$$

Thus, we set the descent diretion $\boldsymbol{d}_t$ as $-\boldsymbol{H}_t^{-1}\boldsymbol{g}_t$

BFGS

Broyden-Fletcher-Goldfarb-Shanno method. Approximates $\boldsymbol{H}_t \approx \boldsymbol{B}_t$:

$$\boldsymbol{B}_{t+1} = \boldsymbol{B}_t + \frac{\boldsymbol{y}_t \boldsymbol{y}_t^\top}{\boldsymbol{y}_t^\top \boldsymbol{s}_t} - \frac{(\boldsymbol{B}_t \boldsymbol{s}_t)(\boldsymbol{B}_t \boldsymbol{s}_t)^\top}{\boldsymbol{s}_t^\top \boldsymbol{B}_t \boldsymbol{s}_t} \tag{32}$$

$$\boldsymbol{s}_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1} \tag{33}$$

$$\boldsymbol{y}_t = \boldsymbol{g}_t - \boldsymbol{g}_{t-1} \tag{34}$$

- $\boldsymbol{B}_0$ is typically intialized as $\boldsymbol{I}$ (positive definite)
- To ensure $\boldsymbol{B}_{t+1}$ remains pd, ρ must satisfy the Wolfe conditions:

$$\mathcal{L}(\boldsymbol{\theta}_t + \rho \boldsymbol{d}_t) \leq \mathcal{L}(\boldsymbol{\theta}_t) + c_1 \rho \boldsymbol{d}_t^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) \tag{35}$$

$$\mathcal{L}(\boldsymbol{\theta}_t + \rho \boldsymbol{d}_t) \geq c_2 \rho \boldsymbol{d}_t^\top \nabla \mathcal{L}(\boldsymbol{\theta}_t) \tag{36}$$

where $0 < c_1 < c_2 < 1$

Limited memory BFGS

In practice, BFGS directly provides the inverse Hessian approximation:

$$\boldsymbol{C}_{t+1} = \left(\boldsymbol{I} - \frac{\boldsymbol{s}_t \boldsymbol{y}_t^\top}{\boldsymbol{y}_t^\top \boldsymbol{s}_t}\right) \boldsymbol{C}_t \left(\boldsymbol{I} - \frac{\boldsymbol{y}_t \boldsymbol{s}_t^\top}{\boldsymbol{y}_t^\top \boldsymbol{s}_t}\right) + \frac{\boldsymbol{s}_t \boldsymbol{s}_t^\top}{\boldsymbol{y}_t^\top \boldsymbol{s}_t} \tag{37}$$

where $\boldsymbol{C}_t \approx \boldsymbol{H}_t^{-1}$.

- The update can be reduced to a recurrence relation that depends explicitly on $\boldsymbol{C}_0$ and the history of $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$
- For computational efficiency, the M most recent $\boldsymbol{s}_t$, $\boldsymbol{y}_t$ may be used instead
- M is typically $\in [5, 20]$
- This approach is termed L-BFGS (limited memory BFGS)

Maximum likelihood estimation: Key definitions

Given a set of n independent observations $x_1, x_2, \ldots, x_n$ from a random sample, the **likelihood function** is:

$$L(x_1, x_2, \ldots, x_n; \theta) = f(x_1; \theta) f(x_2; \theta) \cdots f(x_n; \theta) \tag{38}$$

The **maximum likelihood estimator (MLE)**, $\hat{\theta}$, is the value of θ that maximizes the [log-]likelihood function:

$$\frac{\partial \ln L(x_1, x_2, \cdots, x_n; \theta)}{\partial \theta} = 0 \tag{39}$$

For multiple parameters, the likelihood function is:

$$L(x_1, \ldots, x_n; \theta_1, \ldots, \theta_1, \ldots, \theta_m) = \prod_{i=1}^{n} f(x_i; \theta_i; \theta_1, \ldots, \theta_m) \tag{40}$$

And the MLE's would be found by simultaneously solving the partial derivatives set to 0 for each parameter.

Maximum likelihood estimation (MLE) in logistic regression

The log-likelihood function for the binomial logistic regression case is

$$\ell(\beta) = \sum_i \left[y_i \left(\beta_0 + \beta_1 x_i \right) - \log \left(1 + e^{\beta_0 + \beta_1 x_i} \right) \right] \tag{41}$$

The optimal $\hat{\beta}$ which maximizes $\ell(\beta)$ is the **maximum likelihood estimate**.

Also recall the derivative of ℓ:

$$\nabla_{\boldsymbol{\beta}} \ell = \begin{pmatrix} \frac{\partial \ell}{\partial \beta_0} \\ \frac{\partial \ell}{\partial \beta_1} \end{pmatrix} = \begin{pmatrix} \sum_i \left[y_i - p(x_i) \right] \\ \sum_i \left[x_i \left(y_i - p(x_i) \right) \right] \end{pmatrix} \tag{42}$$

We can use either Newton-Raphson or gradient *ascent* to *maximize* ℓ.

Gradient ascent for MLE in logistic regression

This approach only requires the first derivative:

$$\beta_{k+1} = \beta_k + \lambda \nabla \ell(\beta_k) \tag{43}$$

Thus, to find $\hat{\beta}$ we iterate using:

$$\begin{pmatrix} \beta_{0,k} \\ \beta_{1,k} \end{pmatrix} = \begin{pmatrix} \beta_{0,k} \\ \beta_{1,k} \end{pmatrix} + \lambda \begin{pmatrix} \sum_i \left[y_i - p(x_i) \right] \\ \sum_i \left[x_i \left(y_i - p(x_i) \right) \right] \end{pmatrix} \tag{44}$$

Because the log-likelihood is *concave*, and thus a *maximization* problem, we *ascend* the function and thus *add* the scaled derivative.

- As we can see, the gradient ascent method does not require a second derivative
- However, it may require more iterations to converge than Newton-Raphson

Newton-Raphson approach for MLE in logistic regression

The optimal point $\hat{\beta}$ is given by the root of the equation $\nabla_{\boldsymbol{\beta}} \ell = 0$.

Applying Newton-Raphson, the update step is:

$$\beta_{k+1} = \beta_k - \boldsymbol{H}_{\beta_k}^{-1}(\ell) \nabla_{\beta_k} \ell(\beta_k) \tag{45}$$

The operator $\boldsymbol{H}^{-1}$ represents the inverse **Hessian** (second derivative) matrix of ℓ with respect to β:

$$\boldsymbol{H}_{\beta_k}(\beta_k) = \nabla^2_{\beta_k} \ell = \begin{pmatrix} \frac{\partial^2 \ell(\beta)}{\partial {\beta_0}^2} & \frac{\partial^2 \ell(\beta)}{\partial \beta_0 \beta_1} \\ \frac{\partial^2 \ell(\beta)}{\partial \beta_1 \beta_0} & \frac{\partial^2 \ell(\beta)}{\partial {\beta_1}^2} \end{pmatrix} \tag{46}$$

Note that the (45) is just the matrix representation of the 1-D case:

$$\beta_{k+1} = \beta_k - \frac{\ell'(\beta_k)}{\ell''(\beta_k)} \tag{47}$$

Newton-Raphson approach for MLE (cont.)

We can work out each component of the second derivative:

$$\frac{\partial^2 \ell(\beta)}{\partial {\beta_0}^2} = -\sum_i p(x_i)(1 - p(x_i)) \tag{48}$$

$$\frac{\partial^2 \ell(\beta)}{\partial \beta_0 \beta_1} = -\sum_i x_i p(x_i)(1 - p(x_i)) \tag{49}$$

$$\frac{\partial^2 \ell(\beta)}{\partial {\beta_1}^2} = -\sum_i x_i^2 p(x_i)(1 - p(x_i)) \tag{50}$$

The complete update can then be shown as:

$$\begin{pmatrix} \beta_{0,k+1} \\ \beta_{1,k+1} \end{pmatrix} = \begin{pmatrix} \beta_{0,k} \\ \beta_{1,k} \end{pmatrix} - \left[\begin{pmatrix} \frac{\partial^2 \ell(\beta)}{\partial {\beta_0}^2} & \frac{\partial^2 \ell(\beta)}{\partial \beta_0 \partial \beta_1} \\ \frac{\partial^2 \ell(\beta)}{\partial \beta_1 \partial \beta_0} & \frac{\partial^2 \ell(\beta)}{\partial {\beta_1}^2} \end{pmatrix}^{-1} \begin{pmatrix} \frac{\partial \ell}{\partial \beta_0} \\ \frac{\partial \ell}{\partial \beta_1} \end{pmatrix} \right]_{\beta_k} \tag{51}$$

Alternatively:

$$\beta_{k+1} = \beta_k - \left[\left(\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta} \right]_{\beta_k} \tag{52}$$

Constrained optimization

Deals with problems where we seek to minimize an objective:

$$\min_{\boldsymbol{\theta} \in \mathcal{C}} \mathcal{L}(\boldsymbol{\theta}) \tag{53}$$

subject to:

$$h_i(\boldsymbol{\theta}) = 0, \quad i \in \mathcal{E} \tag{54}$$

$$g_j(\boldsymbol{\theta}) \leq 0, \quad j \in \mathcal{I} \tag{55}$$

where:

- $\mathcal{C}$: constraint/feasible set
- $\mathcal{E}$: set of equality constraints
- $\mathcal{I}$: set of inequality constraints

Lagrange multipliers

Given an optimization problem with m equality contraints $\boldsymbol{h}(\boldsymbol{\theta})$, we can write the Lagrangian as:

$$L(\boldsymbol{\theta}, \lambda) := \mathcal{L}(\boldsymbol{\theta}) + \sum_{j=1}^{m} \lambda_j h_j(\boldsymbol{\theta}) \tag{56}$$

where λ_j are the Lagrange multipliers.
Thus, to find *, we solve:

$$\nabla_{\boldsymbol{\theta}, \lambda} L(\boldsymbol{\theta}, \lambda) = \mathbf{0} \tag{57}$$

- If $\boldsymbol{\theta} \in \mathbb{R}^D$, there are $D + m$ equations with an equal number of unknonwns

Karush-Kuhn-Tucker (KKT) conditions

Given a problem with equality constraints $\boldsymbol{h}(\boldsymbol{\theta}) = \mathbf{0}$ and inequality constraints $\boldsymbol{g}(\boldsymbol{\theta}) \leq \mathbf{0}$, the generalized Lagrangian is given by:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \mathcal{L}(\boldsymbol{\theta}) + \sum_i \mu_i g_i(\boldsymbol{\theta}) + \sum_j \lambda_j h_j(\boldsymbol{\theta}) \tag{58}$$

The optimization problem then becomes:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\mu} \geq \mathbf{0}, \boldsymbol{\lambda}} L(\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\lambda}) \tag{59}$$

When $\mathcal{L}$ and g are convex, then the KKT conditions are necessary and sufficient for global optimality:

- Feasibility: constraints satisfied
- Stationarity of solution: $\nabla_{\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\lambda}} L = \mathbf{0}$
- Dual feasibility: $\boldsymbol{\mu} \geq \mathbf{0}$
- Complementary slackness: $\boldsymbol{\mu} \odot \boldsymbol{g} = \mathbf{0}$

Further topics

Some to be covered in Advanced Probabilistic ML:

- Linear programming (simplex algorithm)
- Quadratic programming
- Proximal gradient method
- Bound optimization (majorize-minimize algorithms)
- Expectation maximization (EM); for MLE/MAP estimation

Reading assignments

- **PMLI** 8
- **PMLCE** 5