Lecture 12

Optimization

Problem: minimize a scalar-valued loss or cost function $\mathcal{L}(\vec{\theta})$:

$$\vec{\theta}^* = \underset{\vec{\theta}}{\operatorname{argmin}}\, \mathcal{L}(\vec{\theta})$$

assume that $\vec{\theta} \in R^D$ (D: # prms) $\Rightarrow$ continuous optimization

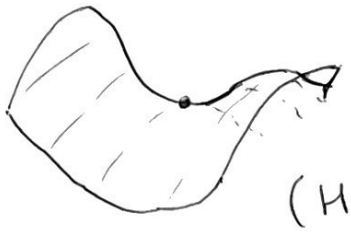
$\exists \delta > 0$ s.t. for $\|\vec{\theta} - \vec{\theta}^*\| < \delta$, $\mathcal{L}(\vec{\theta}^*) \leq \mathcal{L}(\vec{\theta})$

Define
$$\begin{cases} g_i(\vec{\theta}) = \frac{\partial}{\partial \theta_i} \mathcal{L}(\vec{\theta}), & \text{gradient} \\ H_{ij}(\vec{\theta}) = \frac{\partial^2}{\partial \theta_i \partial \theta_j} \mathcal{L}(\vec{\theta}) & \text{Hessian} \end{cases}$$

If $\vec{g}(\vec{\theta}^*) = \vec{0}$ and $H(\vec{\theta}^*)$ is pos. def. (that is, all its eigenvalues are >0),

$\vec{\theta}^*$ is a local min.

Saddle points:



some directions downhill, some uphill

($H(\vec{\theta}^*)$ has both pos. & neg. eigenvalues)

Constraints: $\begin{cases} g_j(\vec{\theta}) \leq 0, & j = 1, 2, .. \quad \text{inequality} \\ h_k(\vec{\theta}) = 0, & k = 1, 2, .. \quad \text{equality} \end{cases}$

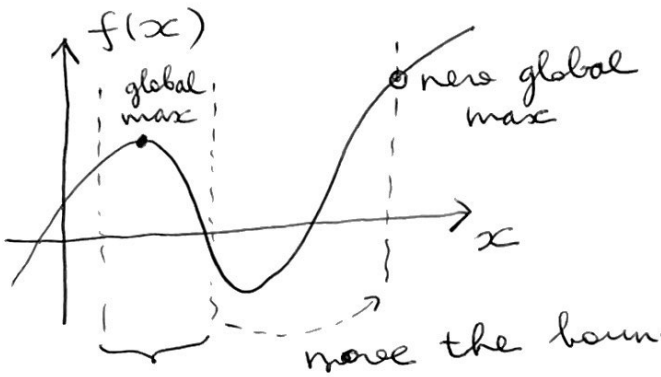
Ex. (a) $h(\vec{\theta}) = 1 - \sum_{i=1}^{D} \theta_i = 0$ prms sum to 1.0

(b) $g_i(\vec{\theta}) = -\theta_i \leq 0$ all prms pos.

$i = 1, .., D$

Feasible set = subset of $\vec{\theta}$ that satisfies all the contstraints.

Constraints can change the number of min/max of a function:

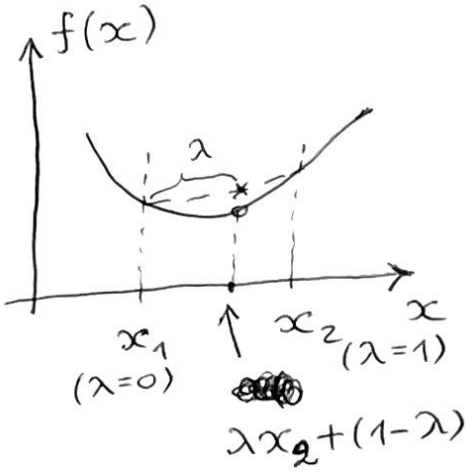


Convex functions: $f(\lambda\vec{x} + (1-\lambda)\vec{y}) \leq \lambda f(\vec{x}) + (1-\lambda) f(\vec{y})$

$\forall \vec{x}, \vec{y}; \quad 0 \leq \lambda \leq 1$

↑ strictly convex if <

[$f(\vec{x})$ is (strictly) concave
if $-f(\vec{x})$ is (strictly) convex]



If $f(\vec{x})$ is strictly convex in some domain $\Rightarrow$ $H = \nabla^2 f(\vec{x})$ is pos. def. in that domain.

First-order methods

(no curvature information)

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \eta_t \vec{d}_t \quad \leftarrow \text{descent direction}$$

$\vec{\theta}_0$ = starting point

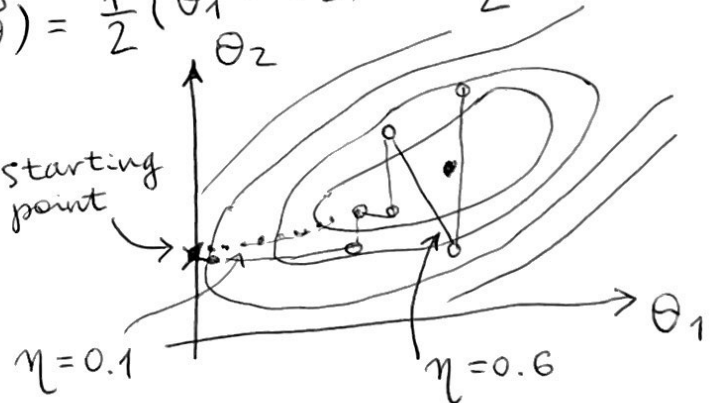
η_t ↑ learning rate (step size)

Usually, $\vec{d}_t = -\underbrace{\vec{\nabla}_{\vec{\theta}} \mathcal{J}(\vec{\theta})\Big|_{\vec{\theta}_t}}_{\vec{g}_t = \text{gradient}}$ ⇐ steepest descent

Learning rate dependence:

Ex. $\mathcal{J}(\vec{\theta}) = \frac{1}{2}(\theta_1^2 - \theta_2)^2 + \frac{1}{2}(\theta_1 - 1)^2$

$\eta = 0.6$ fails to converge, even on a convex problem



Line search: solve 1D minimization problem along the direction of $\vec{d}_t$:

$$\eta_t = \underset{\eta>0}{\operatorname{argmin}}\, \mathcal{L}(\vec{\theta}_t + \eta \vec{d}_t)$$

Ex. $\mathcal{L}(\vec{\theta}) = \frac{1}{2}\underbrace{\theta_i A_{ij} \theta_j + b_i \theta_i}$

sums over repeated indices

$$\frac{d}{d\eta}\mathcal{L}(\vec{\theta}+\eta\vec{d}) = d_i A_{ij}(\theta_j + \eta d_j) + d_i b_i =$$

$$= d_i(A_{ij}\theta_j + b_j) + \eta\, d_i A_{ij} d_j = 0, \quad \text{s.t.}$$

$$\eta = -\frac{d_i(A_{ij}\theta_j + b_j)}{d_i A_{ij} d_j}$$

Momentum methods

"heavy ball" method:

$$\begin{cases} \vec{m}_t = \beta \vec{m}_{t-1} + \vec{g}_{t-1}, & 0<\beta<1 \\ \vec{\theta}_t = \vec{\theta}_{t-1} - \eta_t \vec{m}_t & (\text{typically, } \beta = 0.9) \end{cases}$$

$\beta = 0 \Rightarrow$ steepest descent

note that

$$\vec{m}_t = \beta \vec{m}_{t-1} + \vec{g}_{t-1} = \beta^2 \vec{m}_{t-2} + \beta \vec{g}_{t-2} + \vec{g}_{t-1} = \dots$$

$$= \sum_{\tau=0}^{t-1} \beta^\tau \vec{g}_{t-\tau-1}$$

If $\vec{g}_t = \vec{g} = \text{const} \quad (\forall t)$,

$$\vec{m}_t = \vec{g} \sum_{\tau=0}^{t-1} \beta^\tau \simeq \vec{g} \underbrace{\sum_{\tau=0}^{\infty} \beta^\tau}_{\frac{1}{1-\beta}} = \frac{\vec{g}}{1-\beta} \quad \text{in this limit}$$

Nesterov accelerated gradient method:

$$\begin{cases} \tilde{\vec{\theta}}_{t+1} = \vec{\theta}_t + \beta(\vec{\theta}_t - \vec{\theta}_{t-1}), & \Leftarrow \text{extrapolation step} \\ \vec{\theta}_{t+1} = \tilde{\vec{\theta}}_{t+1} - \eta_t \vec{\nabla}_{\vec{\theta}} \mathcal{I}(\tilde{\vec{\theta}}_{t+1}) & 0 < \beta < 1 \end{cases}$$

One can write

$$\begin{cases} \vec{m}_{t+1} = \beta \vec{m}_t - \eta_t \vec{\nabla}_{\vec{\theta}} \mathcal{I}(\vec{\theta}_t + \beta \vec{m}_t), \\ \vec{\theta}_{t+1} = \vec{\theta}_t + \vec{m}_{t+1} \end{cases}$$

($\vec{m}_{t+1}$: new location for the gradient, updated by momentum; $\vec{\theta}_{t+1} - \vec{\theta}_t$; $\vec{m}_t = \vec{\theta}_t - \vec{\theta}_{t-1}$)

Second-order methods

Newton's method:

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \eta_t H_t^{-1} \vec{g}_t, \quad \text{where}$$

$$H_{ij,t} = \frac{\partial^2}{\partial\theta_i \partial\theta_j} \mathcal{I}(\vec{\theta}) \Big|_{\vec{\theta}_t}$$

Consider $\mathcal{L}(\vec{\theta}) = \mathcal{L}(\vec{\theta}_t) + g_{i,t}(\theta_i - \theta_{i,t}) +$

$$+ \frac{1}{2}(\theta_i - \theta_{i,t}) H_{ij,t} (\theta_j - \theta_{j,t}) .$$

↑ quadratic expansion

$$\frac{\partial}{\partial \theta_k} \mathcal{L}(\vec{\theta}) = g_{k,t} + H_{kj,t}(\theta_j - \theta_{j,t}) = 0 , \text{ or}$$

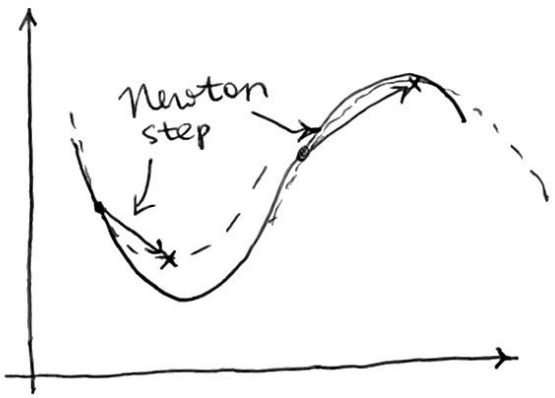
$$\vec{g}_t = - H_t (\vec{\theta} - \vec{\theta}_t) ,$$

$$\vec{\theta} = \vec{\theta}_t - H_t^{-1} \vec{g}_t$$

↑ "intelligent" step size

In practice, can use line search to find the empirical step size η_t.

Note that Newton's method finds the min of a quadratic function in <u>one step</u>, by construction.



Newton steps can go towards maxima instead of minima in non-convex functions (i.e., functions in which the sign of curvature changes)

Tikhonov regularization

Define $\delta_i = \theta_i - \theta_{i,t}$, then

$$\mathcal{L}(\vec{\theta}) \simeq \mathcal{L}(\vec{\theta}_t) + g_{i,t}\,\delta_i + \frac{1}{2}\delta_i H_{ij,t}\delta_j .$$

If we minimize $\mathcal{L}(\vec{\theta}) + \underbrace{\frac{\lambda}{2}\vec{\delta}^2}_{\text{penalty for large steps}}$ instead of $\mathcal{L}(\vec{\theta})$, we get $\vec{\delta} = -\underbrace{(H+\lambda\mathbb{I})^{-1}}_{\text{Tikhonov regularization}}\vec{g}$

$\lambda > 0$

Stochastic gradient descent (SGD)

Consider $\mathcal{L}(\vec{\theta}_t) = \frac{1}{N}\sum_{n=1}^{N} \underbrace{\mathcal{L}_n(\vec{\theta}_t)}_{\text{cost per datapoint}}$

idea: replace $\vec{g}_t = \frac{1}{N}\sum_{n=1}^{N} \vec{\nabla}_{\vec{\theta}}\mathcal{L}_n(\vec{\theta}_t)$

with $\vec{g}_t \simeq \frac{1}{|B_t|}\sum_{n\in B_t} \vec{\nabla}_{\vec{\theta}}\mathcal{L}_n(\vec{\theta}_t)$

B_t = subset of datapoints to use at iteration t;

$|B_t| \ll N$. Typically, $|B_t| \sim 10^1 - 10^2$

Ex. SGD for linear regression

$$\mathcal{L}(\vec{\theta}) = \frac{1}{2N} \sum_{n=1}^{N} (\vec{x}_n \cdot \vec{\theta} - y_n)^2 .$$

Then $$\vec{g}_t = \frac{1}{N} \sum_{n=1}^{N} (\vec{x}_n \cdot \vec{\theta} - y_n) \vec{x}_n$$

If we use $|\mathcal{B}| = 1$, we obtain: (diff. between model & data)

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \eta_t (\vec{x}_{n_t} \cdot \vec{\theta} - y_{n_t}) \vec{x}_{n_t} , \text{ where}$$

n_t = index of the example @ iteration t

preconditioned SGD:

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \eta_t M_t^{-1} \vec{g}_t , \text{ where}$$

M_t is a preconditioning matrix (typically pos. def.); a 'poor man's Hessian'

AdaGrad ('adaptive gradient')

$$\theta_{i,t+1} = \theta_{i,t} - \eta_t \frac{1}{\sqrt{s_{i,t} + \epsilon}} g_{i,t} ,$$

where $s_{i,t} = \underbrace{\sum_{\tilde{t}=1}^{t} g_{i,\tilde{t}}^2}_{\text{sum over past grad}^2}$, $\epsilon > 0$ is a small hyperprm

Typically, $\eta_t = \eta = \text{const}$.

clearly, $M_t = \text{diag}((\vec{s}_t + \vec{\epsilon})^{1/2})$, where

$$\vec{s}_t = \begin{pmatrix} s_{1,t} \\ s_{2,t} \\ \vdots \\ s_{D,t} \end{pmatrix} \quad \text{and} \quad \vec{\epsilon} = \begin{pmatrix} \epsilon \\ \epsilon \\ \vdots \\ \epsilon \end{pmatrix}$$

↑ dim of $\vec{\theta}$

[adaptive learning rate]

↳ components of $\vec{s}_t$ may grow large as $t \uparrow$ → learning rate ↘.

RMSProp

Idea: use a weighted average to compute $s_{i,t}$:

$$s_{i,t+1} = \beta s_{i,t} + (1-\beta) g_{i,t}^2 =$$

$$= (1-\beta) g_{i,t}^2 + \beta [\beta s_{i,t-1} + (1-\beta) g_{i,t-1}^2] =$$

$$= (1-\beta)[g_{i,t}^2 + \beta g_{i,t-1}^2] + \beta^2 s_{i,t-1} = \ldots =$$

$$= (1-\beta)[\underbrace{g_{i,t}^2 + \beta g_{i,t-1}^2 + \ldots + \beta^{t-1} g_{i,1}^2}_{\text{weighted sum of grad}^2}]$$

Typically, $\beta = 0.9$.

The update is as with AdaGrad:

$$\theta_{i,t+1} = \theta_{i,t} - \eta_t \frac{1}{\sqrt{s_{i,t} + \epsilon}} g_{i,t}.$$

AdaDelta

$$\theta_{i,t+1} = \theta_{i,t} - \eta_t \frac{\sqrt{\delta_{i,t-1} + \epsilon}}{\sqrt{s_{i,t} + \epsilon}} g_{i,t} \text{ , where}$$

$$\begin{cases} s_{i,t+1} = \beta s_{i,t} + (1-\beta) g_{i,t}^2 , & \leftarrow \text{same as RMSProp} \\ \delta_{i,t} = \beta \delta_{i,t-1} + (1-\beta)(\underbrace{\Delta\theta_{i,t}}_{\theta_{i,t+1} - \theta_{i,t}})^2 . \end{cases}$$

Adam = RMSProp + momentum

"adaptive moment estimation"

Compute $\begin{cases} m_{i,t} = \beta_1 m_{i,t-1} + (1-\beta_1) g_{i,t} , \\ s_{i,t} = \beta_2 s_{i,t-1} + (1-\beta_2) g_{i,t}^2 . \end{cases}$

Update: $\theta_{i,t+1} = \theta_{i,t} - \eta_t \frac{1}{\sqrt{s_{i,t}} + \epsilon} m_{i,t}$

Recommended values:

$$\begin{cases} \beta_1 = 0.9 , & \beta_2 = 0.999 \\ \epsilon = 10^{-6} , & \eta_t = \eta = 10^{-3} \end{cases}$$

Finally, one can 'bias-corrected' moments:

$$\begin{cases} \vec{m}_t \to \hat{\vec{m}}_t = \frac{\vec{m}_t}{1-\beta_1^t} , \\ \vec{s}_t \to \hat{\vec{s}}_t = \frac{\vec{s}_t}{1-\beta_2^t} . \end{cases}$$

These are only important for small t (in the beginning of optimization)