03-Linear Regression (Convexity)

Convexity

Definition of convexity

Convex set

Let X be a vector space, and $S \subseteq X$

S is a convex set $\overset{\text{def}}{\iff}$

$\forall \boldsymbol{x}, \boldsymbol{y} \in S \quad \forall \lambda \in (0,1)$

$\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in S$

Convex function

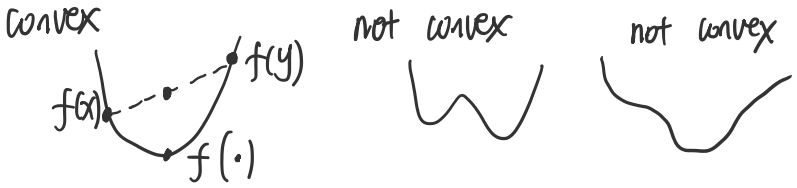
f is a convex function $\overset{\text{def}}{\iff}$

dom f is a convex set [domain of f]

AND

$\forall \boldsymbol{x}, \boldsymbol{y} \in \text{dom } f, \quad \forall \lambda \in (0,1)$

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

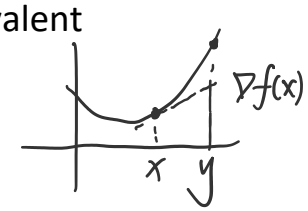


Theorem:

Let f be a twice-differentiable function on a convex open domain dom f.

Then, the following statements are equivalent

a. f is convex



b. (First-order condition)

$\forall \boldsymbol{x}, \boldsymbol{y} \in \text{dom} f, \quad f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + [\nabla f(\boldsymbol{x})]^T \cdot (\boldsymbol{y} - \boldsymbol{x})$

c. (Second-order condition)

$\forall \boldsymbol{x} \in \text{dom} f \qquad \nabla^2 f(\boldsymbol{x}) \succeq \boldsymbol{0}$ positive semi-definite

Calculus review

The gradient is the first-order partial derivatives arranged in the original shape, denoted by $\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$ or $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(\boldsymbol{x}) \\ \frac{\partial}{\partial x_2} f(\boldsymbol{x}) \\ \vdots \\ \frac{\partial}{\partial x_n} f(\boldsymbol{x}) \end{pmatrix} \in \mathbb{R}^n, \text{ where } \boldsymbol{x} \in \mathbb{R}^n$$

The Hessian is the second-order partial derivatives, arranged as a matrix denoted by $\boldsymbol{H}(\boldsymbol{x})$ or $\nabla^2 f(\boldsymbol{x})$

$$\boldsymbol{H}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \ddots & & \\ \vdots & & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}_{n \times n}$$

Proof: b⇒a; c⇒b in one-dimensional case. Others=homework

(Direction: b⇒a)

$\forall \boldsymbol{x}, \boldsymbol{y} \in \text{dom} f$ and $\forall \lambda \in (0,1)$

Def $\boldsymbol{z} := \lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}$

Applying condition b:

$$f(\boldsymbol{x}) \geq f(\boldsymbol{z}) + [\nabla f(\boldsymbol{z})]^T \cdot (\boldsymbol{x} - \boldsymbol{z}) \quad (1)$$

$$f(\boldsymbol{y}) \geq f(\boldsymbol{z}) + [\nabla f(\boldsymbol{z})]^T \cdot (\boldsymbol{y} - \boldsymbol{z}) \quad (2)$$

(1)·λ: $\lambda f(\boldsymbol{x}) \geq \lambda f(\boldsymbol{z}) + \lambda \cdot [\nabla f(\boldsymbol{z})]^T \cdot (\boldsymbol{x} - \boldsymbol{z})$ (3)

(2)·(1−λ): $(1-\lambda) f(\boldsymbol{y}) \geq (1-\lambda) f(\boldsymbol{z}) + (1-\lambda) [\nabla f(\boldsymbol{z})]^T \cdot (\boldsymbol{y} - \boldsymbol{z})$ (4)

(3) ⇐ (4):

$$\lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \geq f(\mathbf{z}) + \left[\nabla f(\mathbf{z})\right]^T \cdot \left(\lambda \cdot (\mathbf{x}-\mathbf{z}) + (1-\lambda)(\mathbf{y}-\mathbf{z})\right)$$
$$= f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y})$$

$\lambda x - \lambda z + y - z - \lambda y + \lambda z$

$\lambda x + (1-\lambda) y - z$

0

(Direction: (3) ⇒ b in one-dimensional case)

By Taylor's Theorem:

$$\forall\, x, y \in \text{dom } f,\ \exists\, z \in (x, y)$$
$$f(y) = f(x) + f'(x)(y-x) + \frac{1}{2} \underbrace{f''(z)}_{\geq 0 \text{ by condition (3)}} \underbrace{(y-x)^2}_{\geq 0}$$

$$\Rightarrow \quad f(y) \geq f(x) + f'(x) \cdot (y-x) \qquad \#$$

Proof sketch for high-dimensional cases:

We consider any 1D line in dom f. If a function cut by any line in the domain is convex, then the function is convex.

Convexity of MSE

First-order derivative

$$J(\mathbf{w}) = \frac{1}{2} \sum_{m=1}^{M} \left(\sum_{k=0}^{d} w_k \cdot x_k^{(m)} - t^{(m)} \right)^2$$

$$\frac{\partial}{\partial w_i} J(\mathbf{w}) = \frac{1}{2} \sum_{m=1}^{M} \cdot 2 \left(\sum_{k=0}^{d} w_k \cdot x_k^{(m)} - t^{(m)} \right) \cdot x_i^{(m)}$$
$$= \sum_{m=1}^{M} \left(\sum_{k=0}^{d} w_k \cdot x_k^{(m)} - t^{(m)} \right) \cdot x_i^{(m)}$$

From the element-wise derivative, we have

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \underbrace{\mathbf{X}^T}_{(d+1)\times M} \cdot \underbrace{(\mathbf{X}\mathbf{w} - \mathbf{t})}_{M \times 1} \qquad \in \mathbb{R}^{d+1}$$

Noticing that $\begin{pmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{pmatrix}$ (X^T) $\cdot$ $\begin{pmatrix} \\ \\ \end{pmatrix}$ ($Xw - t$) has the implicit summation over m.

The gradient can also be obtained from

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{t}\|^2$$

by matrix calculus.

$$\begin{aligned}\nabla_{\mathbf{w}} J(\mathbf{w}) &= \nabla_{\mathbf{w}}\left[\frac{1}{2}(\mathbf{X}\mathbf{w}-\mathbf{t})^T(\mathbf{X}\mathbf{w}-\mathbf{t})\right] \\ &= \nabla_{\mathbf{w}}\left[\frac{1}{2}\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{t}^T\mathbf{X}\mathbf{w} + \frac{1}{2}\mathbf{t}^T\mathbf{t}\right] \\ &= \mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{t}\end{aligned}$$

Second-order derivative

$$\nabla^2 J(\mathbf{w}) = \mathbf{X}^T\mathbf{X} \succeq 0 \qquad \text{(PSD)}$$

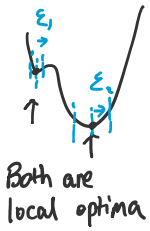
Conclusion: Linear regression is convex in the parameters

Optimality of a convex function

Local optimum

x is a local optimum of f $\Leftrightarrow$ (def)

$\exists \varepsilon > 0$, $\forall z \in \text{dom } f$, if $\|z - x\| < \varepsilon$ then $f(x) \le f(z)$



Global optimum

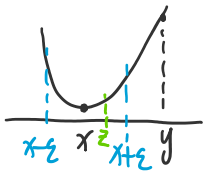
x is a (global) optimum of f (def)

$\forall z \in \text{dom } f \quad f(x) \le f(z)$

Theorem: Convex => local optimum is global

Proof. We pick such ε that $\forall z \in N_\varepsilon(x)$, $f(x) \le f(z)$

ε-Neighbor, fancy way of writing $\|x - z\| < \varepsilon$



If $\|y - x\| < \varepsilon$, we are already happy ☺

If otherwise $\|y - x\| \ge \varepsilon$,

we define $\lambda = \frac{\varepsilon}{2\|y - x\|} \in (0, 1)$

Define $z = (1-\lambda)x + \lambda y$

$\|z - x\| = \lambda \|x - y\| = \frac{\varepsilon}{2} < \varepsilon \Rightarrow f(x) \leq f(z)$

By convexity

$$(1-\lambda) f(x) + \lambda f(y) \geq f((1-\lambda)x + \lambda y)$$
$$= f(z)$$
$$\geq f(x)$$

Thus $f(y) \geq f(x)$. We are also happy ☺ #

Further reading on convexity analysis in unconstrained scenario:
http://www.princeton.edu/~aaa/Public/Teaching/ORF523/S16/ORF523_S16_Lec7_gh.pdf

Closed-form solution to linear regression

Set $\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbf{0}$

$$\mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{t} = \mathbf{0}$$

$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$ is the global optimum

Here, we assume $\mathbf{X}^T\mathbf{X}$ is invertible. If not use pseudo-inverse.
See A.5.4 in https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

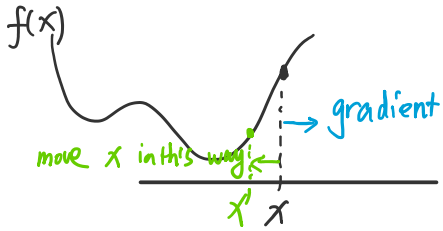
Complexity: $O(n^{2.3}) \sim O(n^3)$
https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations#endnote_blockinversion

Numerical solution - Gradient descent

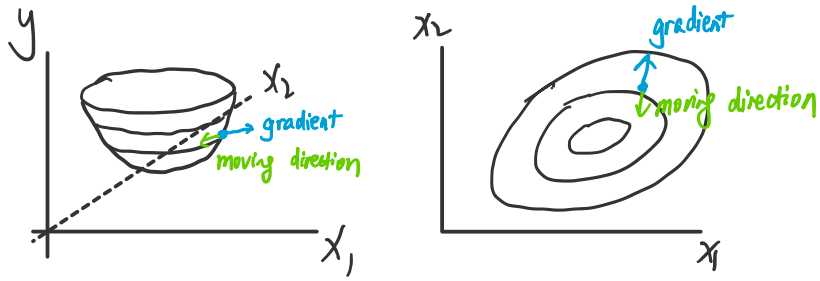
If we move a **small** step towards the opposite direction of gradient, Then, the function value decreases.

As $\alpha \to 0$, let $\boldsymbol{x}' = \boldsymbol{x}$

$$f(\boldsymbol{x}') = f(\boldsymbol{x}) + [\nabla_{\boldsymbol{x}} f(\boldsymbol{x})]^T [-\alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x})] + O(\alpha^2)$$
$$= f(\boldsymbol{x}) - \alpha \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|^2 + O(\alpha^2)$$
$$\Rightarrow f(\boldsymbol{x}') \leq f(\boldsymbol{x})$$



$f(x) \geq f(x')$



Iterative update: We compute the gradient and move a small step against the gradient. Repeat the above criteria for many times.

Gradient Descent Algorithm

Initialize $\boldsymbol{w} = \boldsymbol{w}^{(0)}$ randomly

Loop over epochs $t = 0, 1, 2, \ldots$:

Compute gradient $\nabla J(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}$

Update parameters

$$\boldsymbol{w}^{(t+1)} := \boldsymbol{w}^{(t)} - \alpha \cdot \nabla J(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}$$

Until stopping criterion satisfies

Newton's Method

2nd-order Taylor expansion is more precise than 1st-order.

$$f(\boldsymbol{x}+\boldsymbol{t}) = f(\boldsymbol{x}) + \left[\nabla f(\boldsymbol{x})\right]^{\mathsf{T}}\boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^{\mathsf{T}}\boldsymbol{H}(\boldsymbol{x})\boldsymbol{t} + O(\|\boldsymbol{t}\|^3)$$

Suppose f is indeed quadratic, then $\boldsymbol{t}$ can be solved by

$$\frac{\partial f}{\partial \boldsymbol{t}} = \nabla f(\boldsymbol{x}) + \boldsymbol{H}(\boldsymbol{x})\boldsymbol{t} \overset{\text{set}}{=} \boldsymbol{0}$$

$$\boldsymbol{t} = -\left[\boldsymbol{H}(\boldsymbol{x})\right]^{-1}\nabla f(\boldsymbol{x})$$

However, the objective function J may not be quadratic in $\boldsymbol{w}$, and thus multiple iterations are needed.

Newton's Method

Initialize $\boldsymbol{w} = \boldsymbol{w}^{(0)}$ randomly

Loop over epochs $t = 0, 1, 2, \ldots$:

Compute gradient and Hessian

$$\boldsymbol{g} = \nabla_{\boldsymbol{w}} J(\boldsymbol{w})\Big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}, \qquad \boldsymbol{H} = \nabla^2 J(\boldsymbol{w})\Big|_{\boldsymbol{w}=\boldsymbol{w}^{(t)}}$$

Update parameters $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \boldsymbol{H}^{-1}\boldsymbol{g}$

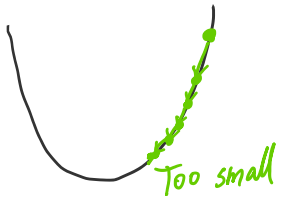
Until stopping criterion satisfies

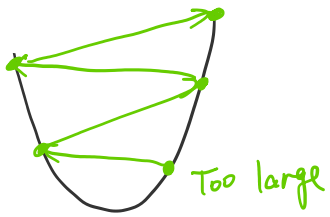
Comments:

- Initialization of $\boldsymbol{w}^{(0)}$:
  - Convex: Doesn't matter that much.
  - Non-convex: small random values around the origin $\mathbf{0}$

- Iteration over data
  - One epoch means one iteration over all data samples
  - Stopping criteria
    - Set MaxIter due to budget constraint
    - Until the convergence of J, i.e.,
      $$\left|J\left(\boldsymbol{w}^{(t)}\right) - J\left(\boldsymbol{w}^{(t-1)}\right)\right| < threshold$$
    - Early stop by validation (TBD in future lectures)

- α is the learning rate
  - Theoretical result: Gradient descent guarantees to converge with some assumptions: https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec6.pdf
  - In practice:
    - Too small ⟹ slow, and performance may not be very good
    - Too large ⟹ overshoot





  - Could be tuned in a multiplicative way, e.g.,
    …, 0.01, 0.03, 0.1, 0.3, 1, 3, …
  - Learning rate decay
    - Start with a relatively large α

    - Decrease α during training
  - Recent advances: Adaptive learning rate for different parameters

- Gradient
  - Full-batch gradient descent

$$J = \frac{1}{M}\sum_{i=1}^{M} J^{(i)}$$

  - Mini-batch gradient descent

In each iteration in the loop,

$$J = \frac{1}{B}\sum_{b=1}^{B} J^{(n_b)}$$

where $B < N$, and $n_1, n_2, \cdots, n_b \in \{x_1, \cdots, x_M\}$,

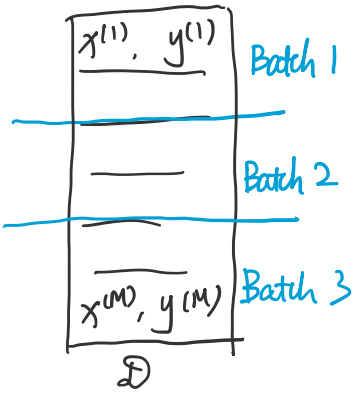
  - Stochastic gradient descent (SGD)

Mini-batch with $B = 1$

Convergence: https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L11.pdf

Essentially, a batch is a sampled subset of the entire dataset. Two settings:

  - Sample with replacement
  - Sample without replacement: In this case, all batches form a partition of the dataset



For epoch $= 1, 2, \cdots$:
  For batch $= 1, \cdots M/B$:
    Compute $\nabla J(\boldsymbol{w}, \mathcal{D}_b)\big|_{\boldsymbol{w}}$
    $\boldsymbol{w}^{new} := \boldsymbol{w} - \alpha \nabla J(\boldsymbol{w}, \mathcal{D}_b)\big|_{\boldsymbol{w}}$

Pros and Cons

	Full batch	SGD ($B = 1$)
Pros	Exact gradient	Gradient could be noisy
Cons	Slow in each param. update	Efficient for each update

Mini-batch gradient descent is a tradeoff. Besides efficiency concerns, evidence shows that small batch gradient descent is better (yields more generalizable solutions) than full batch. The theoretical reasons are not completely clear.

Comparing closed-form solution and numerical solution

- For linear regression:
  - Closed-form solution converges better (guaranteed global optimum) and is faster if the number of sample is not too large.
  - Numerical solution works regardless of the existence of matrix inverse. It is more efficient if we have too many samples.
- For other optimization/computation:
  - Having a closed-form solution is rare, but should always be considered before trying numerical methods. In gradient descent, for example, we compute the gradient by closed-form formula (although the optimum is not closed-form), instead of numerical gradient computing.
  - Numerical methods are more flexible. Gradient descent works for not only convex problems, but also non-convex ones. In non-convex gradient descent, we may be stuck at a local optimum. Restarting gradient descent by different initial parameters may alleviate this problem.