Machine Learning

Gradient Descent, Newton's Method

System of Linear Equations – Infeasible

Consider $\mathbf{A} \in \mathbb{C}^{M \times N}$. We want to **solve** the system of equations (M equations, N unknowns):

$$\mathbf{y} = \mathbf{A}\mathbf{x}.$$

That is, we want to find $\mathbf{x}$ such that the above equation is satisfied, given $\mathbf{A}$ and $\mathbf{y}$.

Case 1: Infeasible (no solution). The system is solvable iff $\mathbf{y} \in \text{span}(\mathbf{A})$. A sufficient (but not necessary) condition is $\text{rank}(\mathbf{A}) = M$.

SLE – General SVD Solution

Case 2: Feasible. If $\mathbf{y} \in \text{span}(\mathbf{A}) = \text{span}(\mathbf{U})$, the system $\mathbf{y} = \mathbf{Ax}$ is solvable.

For RSVD $\mathbf{A} = \mathbf{U\Sigma V}^H$, the general solution is given by

$$\mathbf{x} = \mathbf{V\Sigma}^{-1}\mathbf{U}^H\mathbf{y} + \mathbf{b}$$

for any $\mathbf{b} \in \mathcal{N}(\mathbf{A}) = \mathcal{N}(\mathbf{V}^H) = \text{span}(\mathbf{V})^{\perp} = \{\mathbf{z} \in \mathbb{C}^N : \mathbf{z}^H\mathbf{y} = 0 \; \forall \mathbf{y} \in \text{span}(\mathbf{V}) \}$.

Verify: $\mathbf{Ax} = \mathbf{A}(\mathbf{V\Sigma}^{-1}\mathbf{U}^H\mathbf{y} + \mathbf{b}) = \mathbf{U\Sigma V}^H(\mathbf{V\Sigma}^{-1}\mathbf{U}^H\mathbf{y} + \mathbf{b}) = \mathbf{UU}^H\mathbf{y} = \mathbf{y}$.

Thus, there are as many solutions as $|\mathcal{N}(\mathbf{A})|$. Subspaces come in two cardinalities: 0 and infinity.

SLE – Full Row-Rank

Special case 1: $\rho = \text{rank}(\mathbf{A}) = M < N$ (full-row rank). This implies that $\mathbf{A}$ is wide. Therefore, there are $|\mathcal{N}(\mathbf{A})| = \infty$ solutions to the system. Moreover,

$$\begin{aligned}\mathbf{A}^{\dagger R} &= \mathbf{A}^H(\mathbf{A}\mathbf{A}^H)^{-1} \\ &= \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^H(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^H\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^H)^{-1} \\ &= \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^H(\mathbf{U}^H)^{-1}\boldsymbol{\Sigma}^{-2}(\mathbf{U})^{-1} \\ &= \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H.\end{aligned}$$

More unknowns (N) than independent equations ($\rho = M$).

In this special case, the system solution takes the special form:

$$\mathbf{x} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H\mathbf{y} + \mathbf{b} \;=\; \mathbf{A}^{\dagger R}\mathbf{y} + \mathbf{b}, \qquad \forall \mathbf{b} \in \mathcal{N}(\mathbf{A})$$

and no decomposition of $\mathbf{A}$ (e.g., SVD) is needed for solving the system.

SLE – Full Column-Rank

Special case 2: $\rho = \text{rank}(\mathbf{A}) = N < M$ (full-column rank). This implies that $\mathcal{N}(\mathbf{A}) = \mathbf{0}_N$ and the system has a single solution. Moreover,

$$\begin{aligned}\mathbf{A}^{\dagger L} &= (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H \\ &= (\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^H\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^H)^{-1}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^H \\ &= (\mathbf{V}^H)^{-1}\boldsymbol{\Sigma}^{-2}(\mathbf{V})^{-1}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^H \\ &= \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H.\end{aligned}$$

More independent equations to satisfy (M) than unknowns to tune (N).

In this special case,

$$\mathbf{x} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H\mathbf{y} + \mathbf{0}_N = \mathbf{A}^{\dagger L}\mathbf{y}$$

and no decomposition of $\mathbf{A}$ (e.g., SVD) is needed for solving the system.

SLE – Square, Full Rank

Special case 3: $\rho = \text{rank}(\mathbf{A}) = M = N$ (square full rank). This implies that $\mathcal{N}(\mathbf{A}) = \mathbf{0}_N$ and the system again has a single solution. Moreover,

$$\begin{aligned}\mathbf{A}^{\dagger L} &= \mathbf{A}^{\dagger L} = \mathbf{A}^{-1} \\ &= (\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^H)^{-1} \\ &= (\mathbf{V}^H)^{-1}\boldsymbol{\Sigma}^{-1}(\mathbf{U})^{-1} \\ &= \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H\end{aligned}$$

In this special case,

$$\mathbf{x} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H\mathbf{y} + \mathbf{0}_N = \mathbf{A}^{-1}\mathbf{y}$$

and, again, no decomposition of $\mathbf{A}$ (e.g., SVD) is needed for solving the system.

Multivariate Derivative

Simple extension of the concept of univariate differentiation.

$$\frac{\partial X_{k,l}}{\partial X_{i,j}} = \delta_{i,k}\delta_{l,j}$$

Vector derivative w.r.t. scalar -> vector $\boldsymbol{a} = \frac{\partial \boldsymbol{x}}{\partial y}$, $a_i = \frac{\partial x_i}{\partial y}$

Scalar derivative w.r.t. vector -> vector $\boldsymbol{a} = \frac{\partial x}{\partial \boldsymbol{y}}$, $a_i = \frac{\partial x}{\partial y_i}$. Also written as $\boldsymbol{a} = \nabla_{\boldsymbol{y}} x$.

Vector derivative w.r.t. vector -> matrix $\boldsymbol{A} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{y}}$, $A_{i,j} = \frac{\partial x_i}{\partial y_j}$

Extends to matrix w.r.t to martrix…

(see more here: https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf)

Multivariate Derivative - Examples

Example 1: First order (Linear)

$$\nabla_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{y} = \boldsymbol{a}$$

$$a_i = \frac{\partial(\boldsymbol{x}^T \boldsymbol{y})}{\partial x_i} = \frac{\partial\left(\sum_j x_j y_j\right)}{\partial x_i} = y_i \Rightarrow \boldsymbol{a} = \boldsymbol{y}$$

Example 2: Second order (Quadratic)

$$\nabla_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} = \boldsymbol{a}$$

$$a_i = \frac{\partial(\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x})}{\partial x_i} = \frac{\partial\left(\sum_j \sum_k x_j A_{j,k} x_k\right)}{\partial x_i} = \cdots = \sum_j \left(A_{i,j} + A_{j,i}\right) x_j \Rightarrow \boldsymbol{a} = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{x}$$

Multivariate Optimization

$$\text{P: } \min_{\boldsymbol{x} \in F} f(\boldsymbol{x})$$

This is a multivariate optimization problem.

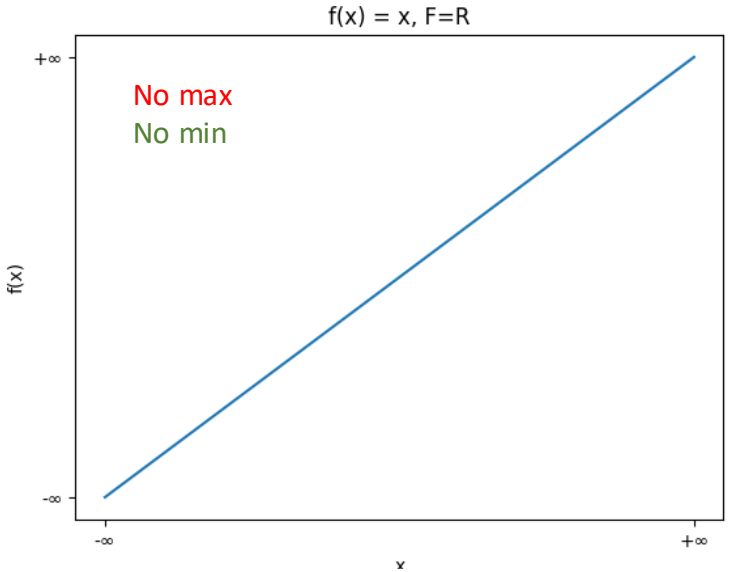
Find the argument $\boldsymbol{x}$ that minimizes objective function $f: F \to \mathbb{R}$

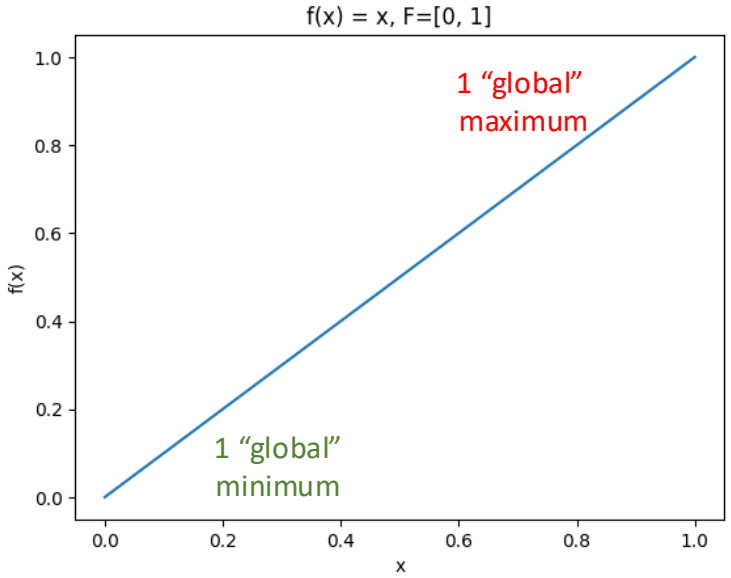
Can have no solution, unique solution, multiple solutions, infinite solutions.

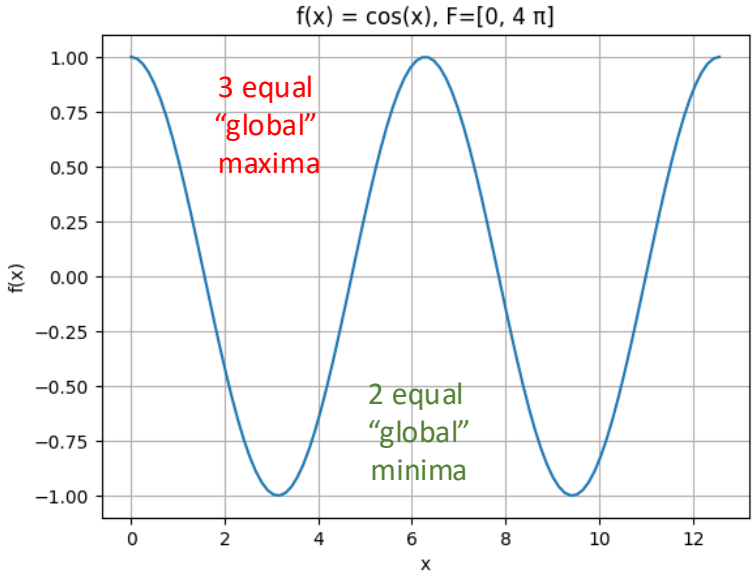
Examples:

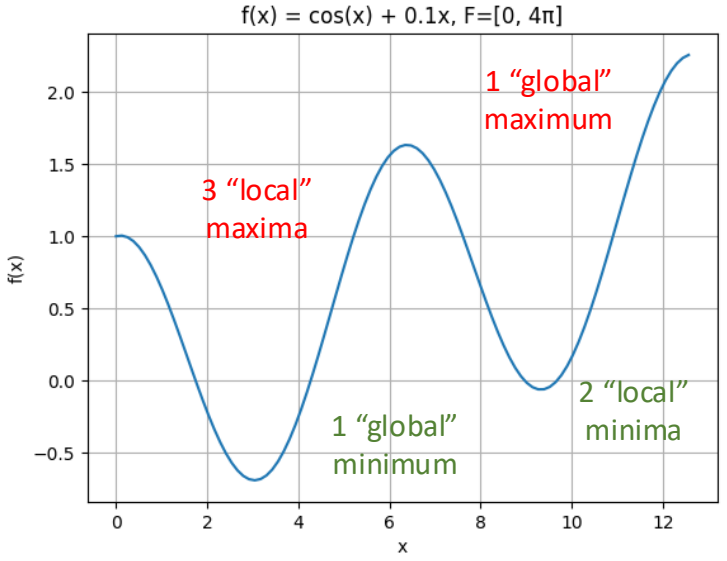
- $f(x) = x, F = \mathbb{R}$. No solution.
- $f(x) = x, F = [0,1]$. Single solution.
- $f(x) = x^2, F = \mathbb{R}$. Single solution.
- $f(x) = \cos(x), F = [0, 4\pi]$. Two solutions.
- $f(x) = \cos(x), F = \mathbb{R}$. Infinitely many solutions.

Multivariate Optimization









Multivariate Optimization

$$\text{P: } \min_{x \in F} \ f(\boldsymbol{x})$$

Therefore, the combination of F and f determine whether a problem is solvable or not; if it is solvable, they determine how many solutions it has and how easy/difficult it is to obtain any of the solutions.

Convex F and f make P an instance of **convex optimization.**

This is an important field of study.

Convex optimization problems have a single global minimum. No local minima.

See more here: https://web.stanford.edu/~boyd/cvxbook/

Least Squares (LS)

Least Squares problem: Given $\boldsymbol{A}$ and $\boldsymbol{y}$, solve:

$$\text{P: } \min_{\boldsymbol{x}\in\mathbb{R}^N} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$$

This is an instance of **convex** optimization (quadratic minimization objective, unconstrained).

Need an algorithm (i.e., methodology) for solving this problem (i.e., find min and argmin).

We start with inspecting the problem, in search of an efficient solution.

Based on our SLE understanding, there is a case where the algorithm can be fast: single-shot solution.

This is the case of $\boldsymbol{y} \in span(\boldsymbol{A})$…

LS – Special Case: $\boldsymbol{y} \in span(\boldsymbol{A})$

We first notice that: $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \geq 0$

Then, we recall that when $\boldsymbol{y} \in span(\boldsymbol{A})$, there exists $\boldsymbol{x}$ so that $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$.

Specifically, by SLE theory (considering R-SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$) it holds:

$$\mathbf{x} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H\mathbf{y} + \mathbf{b}, \mathbf{b} \in \text{null}(\boldsymbol{A})$$

$$\Rightarrow \mathbf{A}\mathbf{x} = \mathbf{y}$$

$$\Leftrightarrow \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} = \mathbf{0}$$

$$\Leftrightarrow \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = 0$$

LS – Special Case: $\boldsymbol{y} \in span(\boldsymbol{A})$

Therefore, when $\boldsymbol{y} \in span(\boldsymbol{A})$, for any $\mathbf{b} \in \text{null}(\boldsymbol{A})$,

$$\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{H}\mathbf{y} + \mathbf{b} \in \text{argmin}_{\boldsymbol{x}\in\mathbb{R}^{N}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$$

And

$$\min_{\boldsymbol{x}\in\mathbb{R}^{N}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = 0$$

This is an instance of **convex** optimization (quadratic minimization objective, unconstrained feasibility).

1) Perform R-SVD: $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \xleftarrow{RSVD} \boldsymbol{A}$

2) Return $\mathbf{x}_{\text{opt}} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{H}\mathbf{y} + \mathbf{b}$ for any $\mathbf{b} \in \mathcal{N}(\boldsymbol{A})$ // Or just return $\mathbf{x}_{\text{opt}} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{H}\mathbf{y}$

- Note: Clearly, the LS problem in this case has infinitely many optimal arguments since $|\mathcal{N}(\boldsymbol{A})| = \infty$. Since $\mathbf{0} \in \text{null}(\boldsymbol{A})$, $\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{H}\mathbf{y}$ is one of the infinitely many optimal arguments.

LS – General Case

Given $\boldsymbol{A}$ and $\boldsymbol{y} \in \mathbb{R}^D$, solve:

$$\text{P: } \min_{\boldsymbol{x} \in \mathbb{R}^N} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$$

This is an instance of **convex** optimization (quadratic minimization objective, unconstrained feasibility).

In calculus you learned that the stationary point of a function is where the function derivative becomes 0.

For unconstrained convex optimization (like the LS above), this stationary point is an argmin.

LS – General Case

So, in the general case, LS can be solved by setting derivative of objective $L(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ to 0.

To that end, we expand the quadratic objective as follows:

$$L(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y}$$

We calculate the derivative (gradient) as:

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y})$$

$$= \nabla_{\boldsymbol{x}}\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \nabla_{\boldsymbol{x}}\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x} - \nabla_{\boldsymbol{x}}\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{y} + \nabla_{\boldsymbol{x}}\boldsymbol{y}^T\boldsymbol{y}$$

$$= (\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{y})2$$

LS – General Case

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}) = (\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{y})2$$

Setting the gradient equal to 0:

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x})\Big|_{\boldsymbol{x}_{opt}} = 0$$

$$\left(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_{opt} - \boldsymbol{A}^T\boldsymbol{y}\right)2 = 0$$

$$\boldsymbol{A}^T\boldsymbol{y} = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_{opt}$$

This is an SLE. Since we do not want to make assumptions on the rank of $\boldsymbol{A}$ and $\boldsymbol{A}^T\boldsymbol{A}$, the general solution will be derived by the R-SVD of $\boldsymbol{A}^T\boldsymbol{A}$.

LS – General Case

We start with the R-SVD $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$.

Then we find $\boldsymbol{A}^T\boldsymbol{A} = (\boldsymbol{U\Sigma V}^T)^T\boldsymbol{U\Sigma V}^T = \boldsymbol{V\Sigma}^T\boldsymbol{U}^T\boldsymbol{U\Sigma V}^T = \boldsymbol{V\Sigma}^2\boldsymbol{V}^T$.

This constitutes R-SVD of $\boldsymbol{A}^T\boldsymbol{A}$.

Note: R-SVD solution is unique. Once a matrix is written as **orthonormal-columns** × **positive diagonal** × **orthonormal-rows**, this is the unique R-SVD of that matrix.

We also find that $\text{null}(\boldsymbol{A}) = \mathcal{N}(\boldsymbol{A}^T\boldsymbol{A}) = \mathcal{N}(\boldsymbol{V}^T)$

Therefore, the solution of this SLE $\boldsymbol{A}^T\boldsymbol{y} = \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_{opt}$ is:

$$\boldsymbol{x}_{opt} = \mathbf{V\Sigma}^{-2}\boldsymbol{V}^H\boldsymbol{A}^T\mathbf{y} + \mathbf{b} = \mathbf{V\Sigma}^{-2}\boldsymbol{V}^T\boldsymbol{V\Sigma}\boldsymbol{U}^T\mathbf{y} + \mathbf{b} = \mathbf{V\Sigma}^{-1}\boldsymbol{U}^T\mathbf{y} + \mathbf{b}$$

where $\mathbf{b} \in \mathcal{N}(\boldsymbol{A})$.

LS – General Case

So, the solution to the Least Squares problem $\min_{\boldsymbol{x}\in\mathbb{R}^N} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$ is always

$$\boldsymbol{x}_{opt} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T\mathbf{y} + \mathbf{b}, \text{ for any } \mathbf{b} \in \mathcal{N}(\boldsymbol{A})$$

whether $\boldsymbol{y}$ is in $span(\boldsymbol{A})$ or not!

If $\boldsymbol{y} \in span(\boldsymbol{A})$: $\boldsymbol{x} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T\mathbf{y} + \mathbf{b}$ minimizes $\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$, setting it equal to 0, since $\boldsymbol{x} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T\mathbf{y} + \mathbf{b}$ is the solution to the SLE $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$.

If $\boldsymbol{y} \notin span(\boldsymbol{A})$: $\boldsymbol{x} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T\mathbf{y} + \mathbf{b}$ minimizes $\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$, which will be positive (not zero), since $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ does not have a solution.

So, LS is broader than SLE and boils down to SLE pnly when $\boldsymbol{y} \in span(\boldsymbol{A})$

LS – Special Case (full-row or full-column rank)

When $\rho = \text{rank}(\mathbf{A}) = M < N$ (full-row rank), $|\mathcal{N}(\mathbf{A})| = \infty$ and

$$\mathbf{A}^{\dagger R} = \mathbf{A}^H(\mathbf{A}\mathbf{A}^H)^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H.$$

Thus, the LS solution is $\mathbf{x}_{opt} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H\mathbf{y} = \mathbf{A}^{\dagger R}\mathbf{y} + \mathbf{b}$, for any $\mathbf{b} \in \mathcal{N}(\mathbf{A})$.

Recall that $\mathbf{0} \in \mathcal{N}(\mathbf{A})$, so $\mathbf{x}_{opt} = \mathbf{A}^{\dagger R}\mathbf{y}$ is one of the infinitely many solutions.

When $\rho = \text{rank}(\mathbf{A}) = N < M$ (full-column rank), $\mathcal{N}(\mathbf{A}) = \mathbf{0}_N$ and

$$\mathbf{A}^{\dagger L} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H.$$

Thus, the unique LS solution is $\mathbf{x}_{opt} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^H\mathbf{y} = \mathbf{A}^{\dagger L}\mathbf{y}$.

Function Definitions

$L(x)$ is **Continuously Differentiable** (smooth or C1) iff (1) $\nabla L(x)$ exists for all x in its domain and (2) $\nabla L(x)$ is a continuous function.

Example: $L(x) = x^2$, $\nabla L(x) = 2x$ (continuous everywhere)

$L(x)$ is **Twice Continuously Differentiable** (C2) iff (1) It is continuously differentiable, (2) the second derivative $\nabla^2 L(x)$ exists for all x in its domain, and (3) the second derivative $\nabla^2 L(x)$ is a continuous function.

Example: $L(x) = x^2$, $\nabla^2 L(x) = 2$ (constant, thus continuous)

$L(x)$ is **Lipschitz Continuous** (LC) iff there exists a constant $C \geq 0$ such that $|L(x) - L(y)| \leq C\,|x - y|$ for all x, y in its domain.

Example: $L(x) = \max(0, x)$ (ReLU) is LC w/ Lipschitz constant $C = 1$.

$L(x)$ is **Convex** if

$$L(\alpha x + (1 - \alpha)y) \leq \alpha L(x) + (1 - \alpha)L(y)$$

for all x, y in its domain and $\alpha \in [0, 1]$.

Example: $L(x) = x^2$ (the line segment between any two points on the curve lies above the curve)

Gradient Descent (GD)

Above we presented closed-form solutions for LS via SVD or pseudoinversions. These solutions are possible because of the convexity of LS.

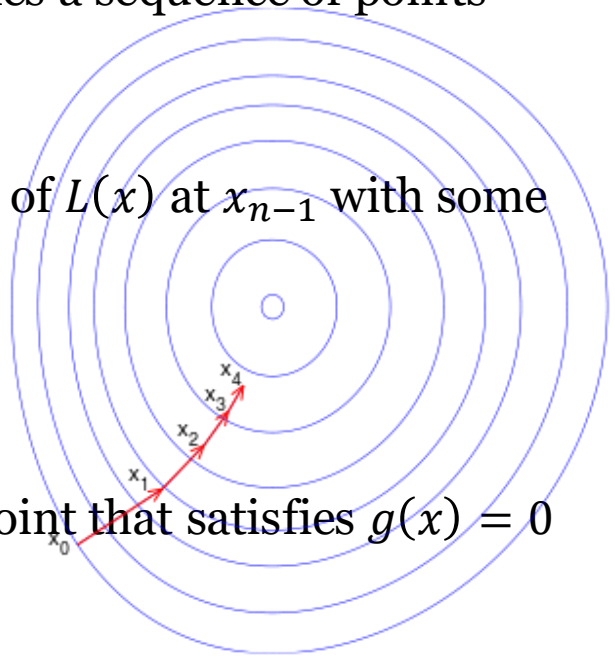
A more general approach to find local/global minima, even for non-convex problems, is the Gradient Descent (GD) method. This is applied when $L(x)$ is differentiable.

GD is an iterative method that is initialized to some point x_0 and determines a sequence of points $x_1, x_2, \ldots$

When at x_{n-1}, x_n is obtained by moving on the direction negative gradient of $L(x)$ at x_{n-1} with some step size.

If the step size is too large, the iterations will diverge.

If the step size is low enough, the iterations will converge to a stationary point that satisfies $g(x) = 0$ (it can be a local minimum or saddle point).



Gradient Descent (GD)

GD Algorithm

1. Start with arbitrary initial point $\boldsymbol{x}_0$,

2. For $n = 1, 2, \ldots$, perform:

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} - \gamma_n \, \boldsymbol{g}(\boldsymbol{x}_{n-1}), \text{ where } \boldsymbol{g}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} L(\boldsymbol{x})$$

3. Upon convergence/termination return $\boldsymbol{x}_n$

Termination: e.g., when $\|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\|_2 < \epsilon$ or when $n > \tau$

Step size γ_n must be chosen appropriately; too small: slow convergence; too large: divergence.

GD for LS

For LS, $\boldsymbol{g(x)} = \nabla_{\boldsymbol{x}} L(\boldsymbol{x}) = (\boldsymbol{A}^T\boldsymbol{Ax} - \boldsymbol{A}^T\boldsymbol{y})2$

GD for LS

1. Start with <u>arbitrary</u> initial point $\boldsymbol{x}_0$,
2. For $n = 1, 2, \ldots$, perform:

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} - \gamma_n(\boldsymbol{A}^T\boldsymbol{Ax}_{n-1} - \boldsymbol{A}^T\boldsymbol{y})2$$

3. Upon convergence/termination return $\boldsymbol{x}_n$

For LS, $g(x) = \nabla_x L(x) = (A^TAx - A^Ty)2$

GD for LS

1. Start with arbitrary initial point x_0,
2. For $n = 1, 2, \ldots$, perform:

$$x_n = x_{n-1} - \gamma_n(A^TAx_{n-1} - A^Ty)2$$

3. Upon convergence/termination return x_n

GD Step Size – Fixed Step Size

For all n,

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} - \gamma g(\boldsymbol{x}_{n-1})$$

That is, we keep constant step size for all updates.

Step size determines convergence or divergence.

If γ is too large, the iterations will diverge.

For convergence, constant step size much be **low enough at each step**!

That is, to guarantee convergence, step size must be upper bounded:

$$\gamma < T$$

where T captures the maximum rate of change of the gradient across the entire domain.

GD Step Size – Fixed Step Size (Theory)

Let $L(x)$ have a **Lipschitz Continuous (LC) gradient** with **Lipschitz constant** C being the minimum positive number for which:

$$\|g(\boldsymbol{x}) - g(\boldsymbol{y})\| \leq C \, \|\boldsymbol{x} - \boldsymbol{y}\| \; \forall \boldsymbol{x}, \boldsymbol{y}$$

That is, the magnitude of the gradient change between any two points is bounded by the distance of these two points, weighted by C. For the Lipschitz constant it holds:

$$C = \sup_{x,y \in F} \frac{\|g(x) - g(y)\|}{\|x - y\|}$$

Then, if $L(x)$ is smooth, $g(x)$ is LC, and we set $\gamma < 1/C$, then GD **converges** to a stationary point that satisfies $g(x) = 0$.

If $L(x)$ is also convex the convergence will be at global optimum.

Clearly, high C means that gradient changes fast; thus, lower γ is needed for convergence.

GD Step Size – Fixed Step Size (Approximation)

Since it is practically impossible to compute the supremum over all pairs of points in F, we find a lower bound for C by examining only the points in a finite search set $S \subset F$.

We set $\gamma < \frac{1}{\hat{C}}$ where

$$\hat{C} = \sup_{x,y \in S} \frac{\|g(x) - g(y)\|}{\|x - y\|}$$

and $S \subset F$ is a search set.

This approximation comes with no guarantees of convergence –but can work well in practice for appropriate sample set S.

This approximation can also work when the function is not Lipschitz continuous.

GD Step Size – Fixed Step Size (C2)

Special Case: Twice Differentiable Functions

If $L(x)$ is C2 with LC gradient, the Lipschitz constant of the gradient C can be found by SVD of the Hessian matrix $H(\boldsymbol{x}) = \boldsymbol{\nabla_{\mathbf{x}^{\mathbf{T}}} \nabla_{\mathbf{x}} f(x)} = \boldsymbol{\nabla}_{\boldsymbol{x}^T} g(\boldsymbol{x})$. Specifically, C is given by

$$C = \sup_{x \in F} \sigma_{\max}(H(\boldsymbol{x}))$$

In general, it is practically intractable to compute the supremum over all points in F.

Instead, we can find a lower bound for C by evaluating the largest eigenvalue of the Hessian at points in a finite search set $S \subset F$. We set $T < \frac{1}{\hat{C}}$, where:

$$\hat{C} = \max_{x \in S} \sigma_{\max}(H(\boldsymbol{x}))$$

GD Step Size – Fixed Step Size (C2)

Even More Special Case: Quadratic Functions: For quadratic functions, the Hessian $H(\boldsymbol{x}) = H$ is constant across all points $\boldsymbol{x}$ in the domain. Thus, $C = \sigma_{\max}(\boldsymbol{H})$.

For example, for LS objective,

$$H = \nabla_{\boldsymbol{x}} g(\boldsymbol{x}) = \nabla_{\boldsymbol{x}}(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^T\boldsymbol{y})2 = \boldsymbol{A}^T\boldsymbol{A}2$$

Considering RSVDs $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and $\boldsymbol{A}^T\boldsymbol{A}2 = \boldsymbol{V}2\boldsymbol{\Sigma}^2\boldsymbol{V}^T$, we find that

$$C = 2\sigma_{\max}(\boldsymbol{A})^2$$

GD Step Size – Gradient Lipschitz Step Size

If $L(x)$ is C2 with LC gradient, we can perform gradient descent

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} - \gamma_n g(\boldsymbol{x}_{n-1})$$

with step size $\gamma_n = 1/C_n$, where

$$C_n = \sigma_{\max}\left(\boldsymbol{H}\left(x_{(n-1)}\right)\right)$$

is the Lipschitz constant of $g(\boldsymbol{x})$ at $\boldsymbol{x}_{(n-1)}$.

Again, this guarantees convergence to a stationary point.

GD Step Size – Backtracking Line Search

1. fix a parameter $0 < \beta < 1$ (e.g., $\beta = 0.5$) and c > 0 (e.g., $c = 10^{-2}$)
2. initialize with arbitrary γ_n^{cand}
3. while $L(x_{n-1}) - L\left(x_{n-1} - \gamma_n^{cand} g(x_{n-1})\right) \leq c \cdot \gamma_n^{cand} \|g(x_{n-1})\|^2$ (Armijo Condition)
4. update $\gamma_n^{cand} \leftarrow \beta \cdot \gamma_n^{cand}$
5. return $\gamma_n = \gamma_n^{cand}$

For a smooth function with LC gradient, backtracking line search using Armijo's condition guarantees convergence to a stationary point.

See convergence proofs here: https://www.cs.cmu.edu/~ggordon/10725-F12/scribes/10725_Lecture5.pdf

GD Step Size – Exact Line Search

Step size can be optimized such that

$$\gamma_n = \operatorname{argmin}_{\gamma \geq 0} L(\boldsymbol{x}_n)$$

where $\boldsymbol{x}_n = \boldsymbol{x}_{n-1} - \gamma\, \boldsymbol{g}(\boldsymbol{x}_{n-1})$ and $\boldsymbol{g}(\boldsymbol{x}_{n-1}) = (\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_{n-1} - \boldsymbol{A}^T\boldsymbol{y})2$

We define $\boldsymbol{e}_n := \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_n \Rightarrow\ L(\boldsymbol{x}_n) = \|\boldsymbol{e}_n\|^2$

And we find that $\boldsymbol{e}_n = \boldsymbol{y} - \boldsymbol{A}\big(\boldsymbol{x}_{n-1} - \gamma\, \boldsymbol{g}(\boldsymbol{x}_{n-1})\big) = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{n-1} + \boldsymbol{A}\boldsymbol{g}(\boldsymbol{x}_{n-1})\gamma = \boldsymbol{e}_{n-1} + \boldsymbol{A}\boldsymbol{g}(\boldsymbol{x}_{n-1})\gamma$

We also find that $\boldsymbol{g}(\boldsymbol{x}_{n-1}) = (\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}_{n-1} - \boldsymbol{A}^T\boldsymbol{y})2 = -\boldsymbol{A}^T\boldsymbol{e}_{n-1}2$

From the above two, $\boldsymbol{e}_n = \boldsymbol{e}_{n-1} + \boldsymbol{A}\big(-\boldsymbol{A}^T\boldsymbol{e}_{n-1}2\big)\gamma = \boldsymbol{e}_{n-1} - \boldsymbol{A}\boldsymbol{A}^T\boldsymbol{e}_{n-1}2\gamma$

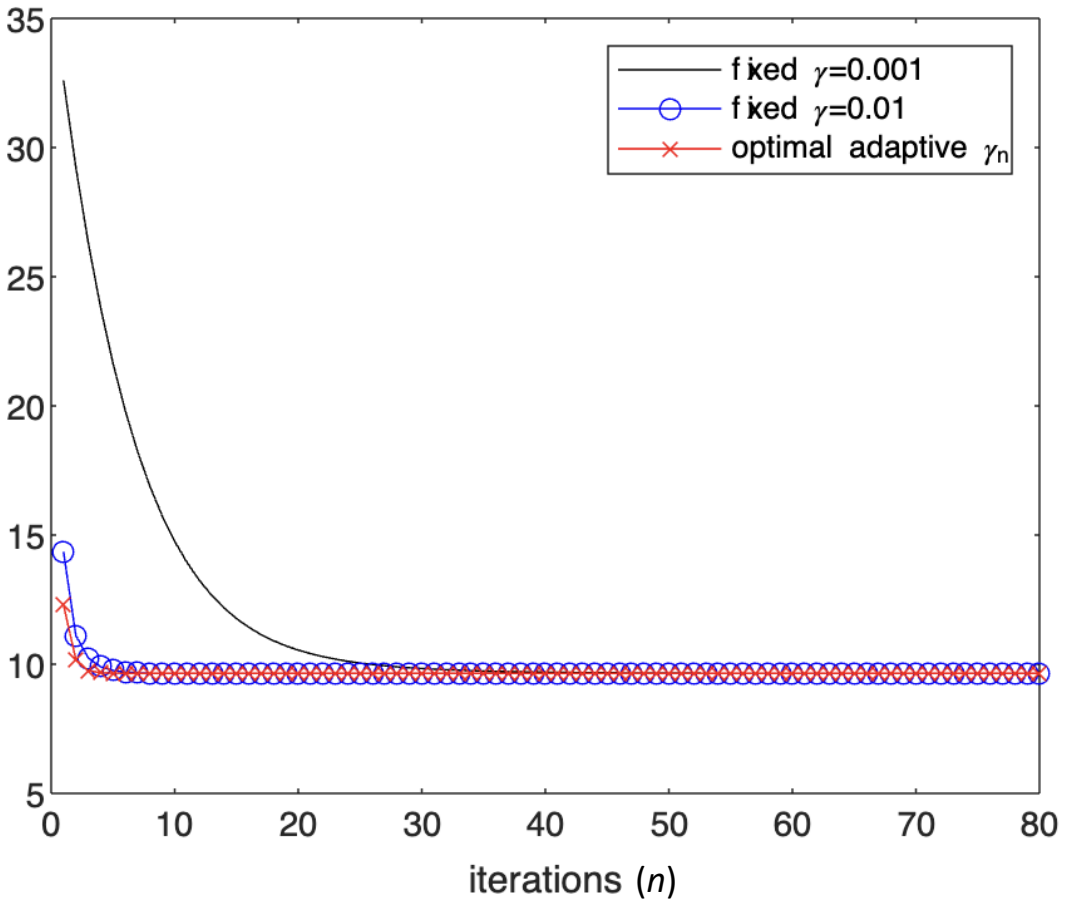
Accordingly, $L(\boldsymbol{x}_n) = \|\boldsymbol{e}_n\|^2 = \|\boldsymbol{e}_{n-1}\|^2 + 4\gamma^2\left\|\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{e}_{n-1}\right\|^2 - 4\gamma\boldsymbol{e}_{n-1}^T\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{e}_{n-1}$

Last, we solve the Exact Line Search as: $\frac{\partial}{\partial_\gamma} L(\boldsymbol{x}_n)\Big|_{\gamma_n} = 0 \Rightarrow \gamma_n = \frac{\left\|\boldsymbol{A}^T\boldsymbol{e}_{n-1}\right\|^2}{2\left\|\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{e}_{n-1}\right\|^2} \Rightarrow \gamma_n = \frac{\|\boldsymbol{g}(\boldsymbol{x}_{n-1})\|^2}{2\|\boldsymbol{A}\boldsymbol{g}(\boldsymbol{x}_{n-1})\|^2}$

GD for LS – Step Size (Example)

$$\boldsymbol{A} \in \mathbb{R}^{100 \times 10}$$

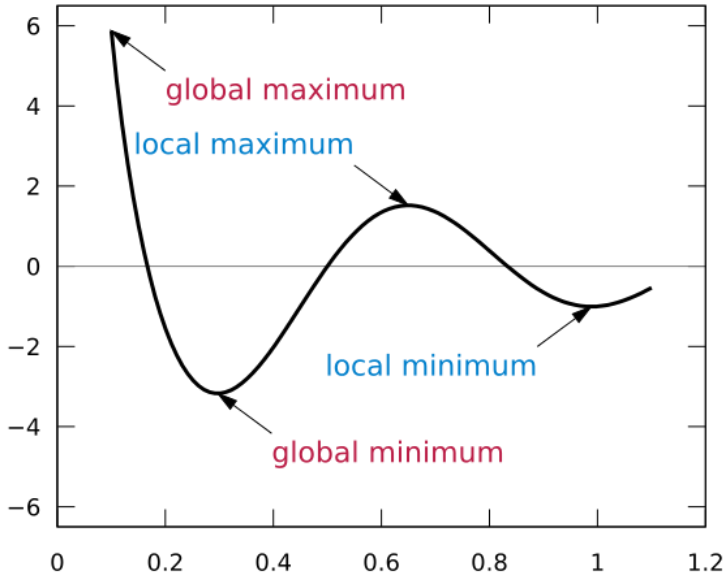
$L(\boldsymbol{x}_n)$



GD for LS – Convergence Guarantees

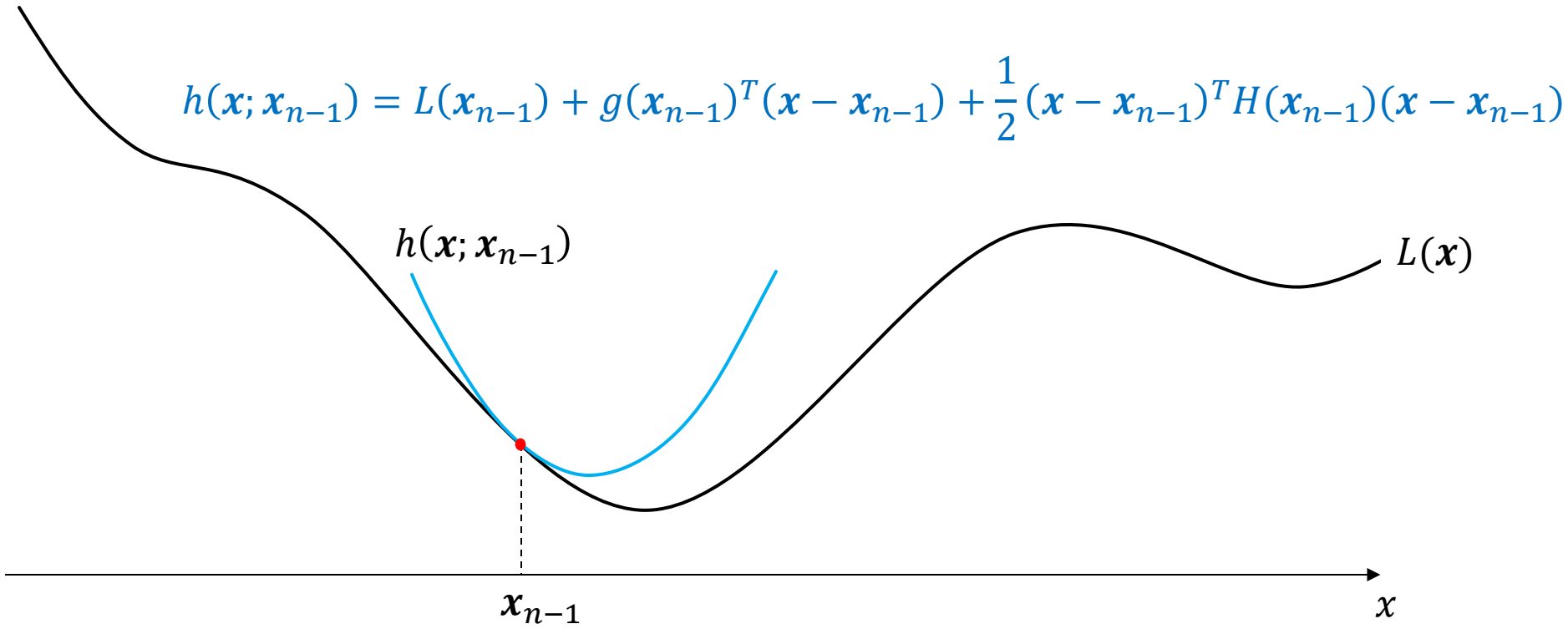
If the function is convex and step sizes appropriately chosen, GD guarantees convergence to global minimum $\boldsymbol{x}_{opt}$.

If the function is non-convex, GD converges to a stationary point $\hat{\boldsymbol{x}}$. Convergence to the global minimum is possible, subject to "lucky" initial guess, but not guaranteed by GD.



Newton's Method

- Iterative steps are taken based on the **2nd order** Taylor series approximation.
- The minimum is sought by traveling downward on the **quadratics** tangent to $L(\boldsymbol{x})$ at each iteration.



Newton's Method

- Given $\boldsymbol{x}_{n-1}$, set $\boldsymbol{x}_n$ to the value of $\boldsymbol{x}$ that minimizes

$$h(\boldsymbol{x}; \boldsymbol{x}_{n-1}) = L(\boldsymbol{x}_{n-1}) + g(\boldsymbol{x}_{n-1})^T(\boldsymbol{x} - \boldsymbol{x}_{n-1}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_{n-1})^T H(\boldsymbol{x}_{n-1})(\boldsymbol{x} - \boldsymbol{x}_{n-1})$$

- For any given $\boldsymbol{x}_{n-1}$, $h(\boldsymbol{x}; \boldsymbol{x}_{n-1})$ is quadratic over $\boldsymbol{x}$ (i.e., convex).

- Minimize over $\boldsymbol{x}$ by setting gradient to 0.

$$c(\boldsymbol{x}; \boldsymbol{v}) = \nabla h(\boldsymbol{x}; \boldsymbol{x}_{n-1}) = g(\boldsymbol{x}_{n-1}) + H(\boldsymbol{x}_{n-1})(\boldsymbol{x} - \boldsymbol{x}_{n-1})$$

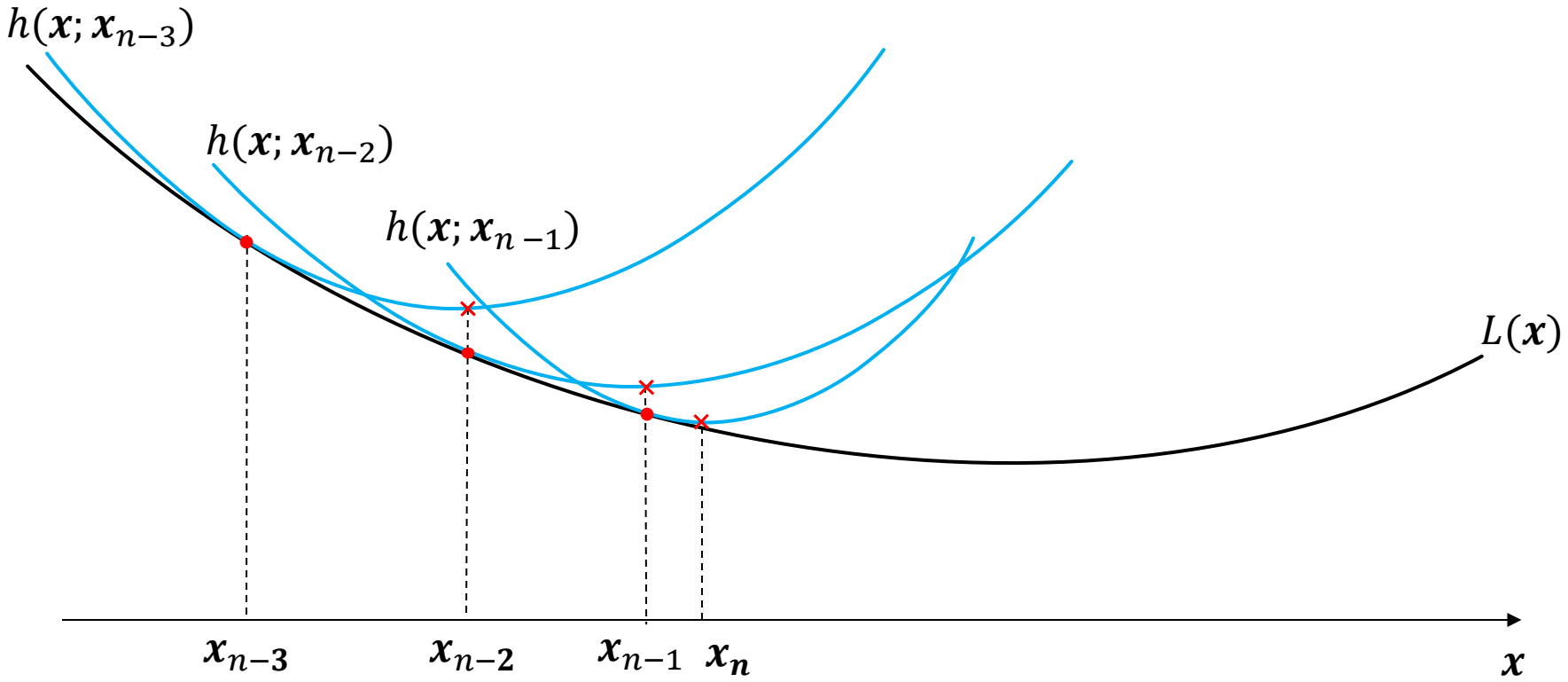
$$c(\boldsymbol{x}; \boldsymbol{x}_{n-1})|_{x=x_n} = g(\boldsymbol{x}_{n-1}) + H(\boldsymbol{v})(\boldsymbol{x}_n - \boldsymbol{v}) = 0$$

- Solve the SLE:

$$H(\boldsymbol{x}_{n-1})\boldsymbol{x}_n = H(\boldsymbol{x}_{n-1})\boldsymbol{x}_{n-1} - g(\boldsymbol{x}_{n-1})$$

Newton's Method

1. Begin at arbitrary point $\boldsymbol{x}_0$
2. For $n = 1,2, \ldots$:
3. Calculate $H(\boldsymbol{x_{n-1}})$ and $g(\boldsymbol{x}_{n-1})$
4. Obtain $\boldsymbol{x}_n$ that solves the SLE: $H(\boldsymbol{x}_{k-1})\boldsymbol{x}_k = H(\boldsymbol{x}_{k-1})\boldsymbol{x}_{k-1} - g(\boldsymbol{x}_{k-1})$
5. Upon convergence/termination, return $\boldsymbol{x}_n$



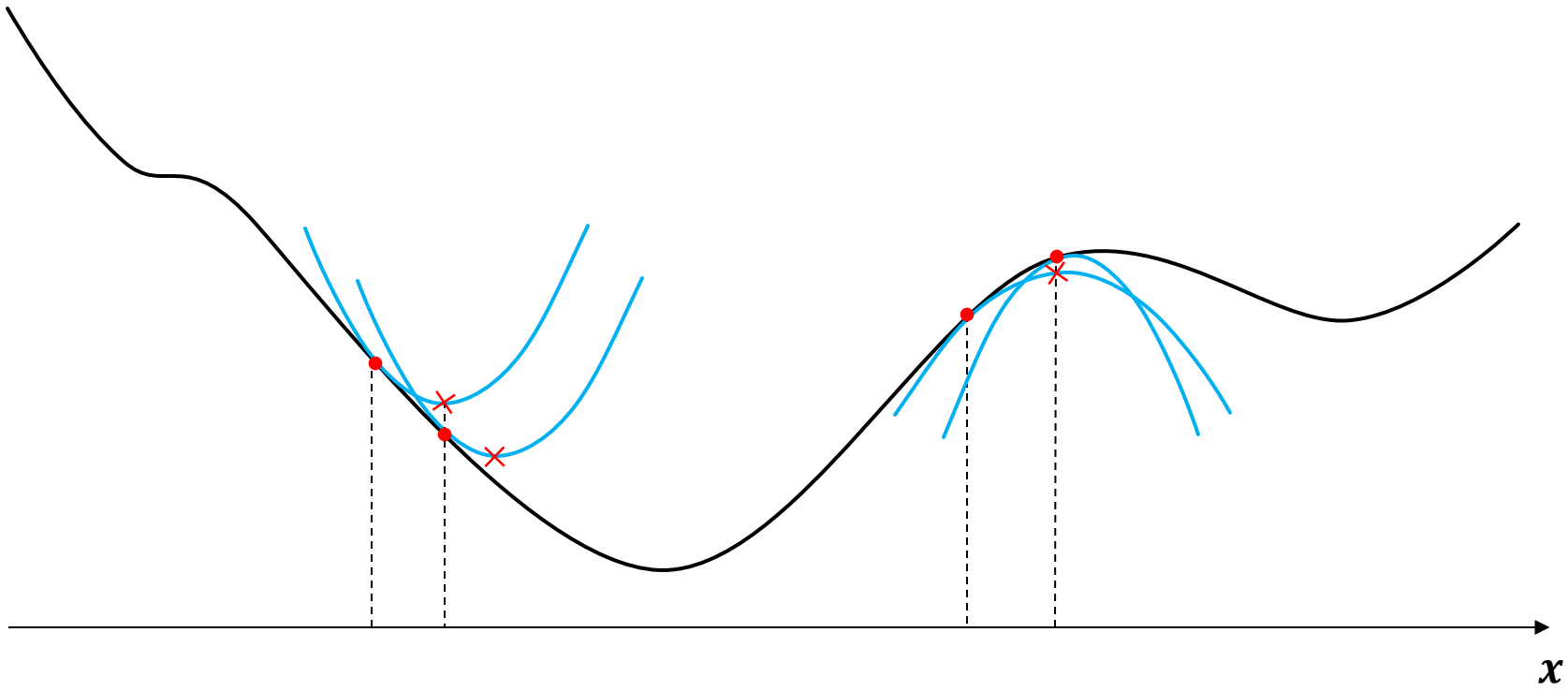
Newton's Method

For a convex problem, this will converge to global optimum.

For quadratic problem, solutions is obtained at first iteration.

For a non-convex function quadratics can be **concave**.

With such functions Newton's method can *climb* to a maximum.



LS uses a simple loss…

So far, we have studied the LS problem where the loss L(x) is a quadratic.

It has a standard closed form solution via SVD or (pseudo)-inversion.

Also, it is a convex problem, and GD returns the global optimum, upon appropriate step size.

Moreover, it has a single Hessian and Lipschitz constant across all values of x.

Since it is a quadratic, Newton's method solves the problem in a single iteration.

Next, let us look into a different loss function.

It is called "cross-entropy" loss and finds wide application in machine learning. We will see more about its connection to machine learning later in the course. For now, we study it a general optimization problem.

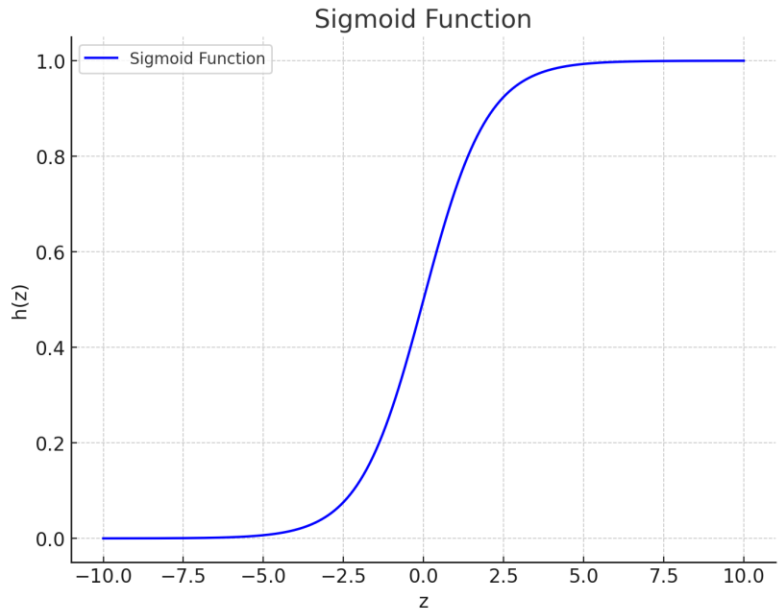
Cross-Entropy Minimization - Definition

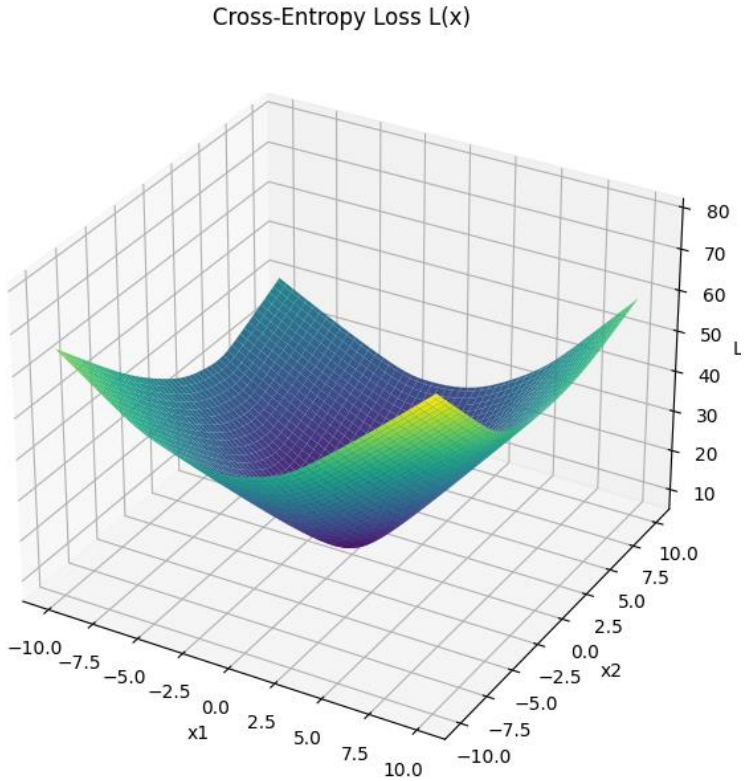
Given $\boldsymbol{A} \in R^{N \times D}$ and $\boldsymbol{y} \in \{0,1\}^N$, we define the CE objective:

$$L(\boldsymbol{x}) = -\sum_{i=1}^{N} \left[y_i \ln\left(h(A_{i,:}\,\boldsymbol{x})\right) + (1 - y_i) \ln\left(1 - h(A_{i,:}\boldsymbol{x})\right)\right]$$

where $A_{i,:}$ is the i-th row of matrix A and $h(z) = \frac{1}{1+\exp(-z)}$ is the **sigmoid** function.

L(x) is still convex but not a quadratic.





Cross-Entropy Minimization - Gradient

$$g(\boldsymbol{x}) := \nabla_{\boldsymbol{x}} L(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \left(-\sum_{i=1}^{N} \left[y_i \ln\left(h\left(A_{i,:}\,\boldsymbol{x}\right)\right) + (1 - y_i) \ln\left(1 - h\left(A_{i,:}\boldsymbol{x}\right)\right) \right] \right)$$

$$= -\sum_{i=1}^{N} \nabla_{\boldsymbol{x}} \left[y_i \ln\left(h\left(A_{i,:}\,\boldsymbol{x}\right)\right) + (1 - y_i) \ln\left(1 - h\left(A_{i,:}\boldsymbol{x}\right)\right) \right] = -\sum_{i=1}^{N} z_i(x)$$

where

$$z_i(\boldsymbol{x}) := \nabla_{\boldsymbol{x}} \left[y_i \ln\left(h\left(A_{i,:}\,\boldsymbol{x}\right)\right) + (1 - y_i) \ln\left(1 - h\left(A_{i,:}\boldsymbol{x}\right)\right) \right]$$

We find

$$z_i(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \left[y_i \ln\left(h\left(A_{i,:}\,\boldsymbol{x}\right)\right) + (1 - y_i) \ln\left(1 - h\left(A_{i,:}\boldsymbol{x}\right)\right) \right]$$

$$= y_i \nabla_{\boldsymbol{x}} \left(\ln\left(h\left(A_{i,:}\,\boldsymbol{x}\right)\right) \right) + (1 - y_i) \nabla_{\boldsymbol{x}} \left(\ln\left(1 - h\left(A_{i,:}\boldsymbol{x}\right)\right) \right)$$

By Chain Rule, for any function $m(\boldsymbol{x})$,

$$\nabla_{\boldsymbol{x}} \ln\left(m(\boldsymbol{x})\right) = \frac{1}{m(\boldsymbol{x})} \nabla_{\boldsymbol{x}} m(\boldsymbol{x})$$

and

$$\nabla_{\boldsymbol{x}}\, h\left(m(\boldsymbol{x})\right) = h\left(m(\boldsymbol{x})\right) \left(1 - h\left(m(\boldsymbol{x})\right)\right) \nabla_{\boldsymbol{x}} m(\boldsymbol{x})$$

Cross-Entropy Minimization - Gradient

Therefore,

$$\nabla_{\boldsymbol{x}}\left(\ln\left(h\left(A_{i,:}\,\boldsymbol{x}\right)\right)\right)=\frac{1}{h\left(A_{i,:}\,\boldsymbol{x}\right)}h\left(A_{i,:}\,\boldsymbol{x}\right)\left(1-h\left(A_{i,:}\,\boldsymbol{x}\right)\right)A_{i,:}^{T}$$
$$=\left(1-h\left(A_{i,:}\,\boldsymbol{x}\right)\right)A_{i,:}^{T}$$

$$\nabla_{\boldsymbol{x}}\left(\ln\left(1-h\left(A_{i,:}\,\boldsymbol{x}\right)\right)\right)=\frac{1}{1-h\left(A_{i,:}\,\boldsymbol{x}\right)}\left(-\left(\nabla_{\boldsymbol{x}}h\left(A_{i,:}\,\boldsymbol{x}\right)\right)\right)$$

$$=-\frac{1}{1-h\left(A_{i,:}\,\boldsymbol{x}\right)}h\left(A_{i,:}\,\boldsymbol{x}\right)\left(1-h\left(A_{i,:}\,\boldsymbol{x}\right)\right)A_{i,:}^{T}=-h\left(A_{i,:}\,\boldsymbol{x}\right)A_{i,:}^{T}$$

Accordingly,

$$z_i(\boldsymbol{x})=y_i\left(1-h\left(A_{i,:}\,\boldsymbol{x}\right)\right)A_{i,:}^{T}+(1-y_i)\left(-h\left(A_{i,:}\,\boldsymbol{x}\right)A_{i,:}^{T}\right)$$
$$=\left(y_i-y_ih\left(A_{i,:}\,\boldsymbol{x}\right)+y_ih\left(A_{i,:}\,\boldsymbol{x}\right)-h\left(A_{i,:}\,\boldsymbol{x}\right)\right)A_{i,:}^{T}$$
$$=\left(y_i-h\left(A_{i,:}\,\boldsymbol{x}\right)\right)A_{i,:}^{T}$$

Cross-Entropy Minimization – Gradient, Hessian, Lipschitz

We found $z_i(\boldsymbol{x}) = \left(y_i - h\left(A_{i,:}\,\boldsymbol{x}\right)\right) A_{i,:}^T \;\forall i$. Therefore, the **CE gradient** is

$$g(\boldsymbol{x}) = -\sum_{i=1}^{N} z_i(x) = -\sum_{i=1}^{N} \left(y_i - h\left(A_{i,:}\,\boldsymbol{x}\right)\right) A_{i,:}^T = \boldsymbol{A}^T(h(\boldsymbol{A}\boldsymbol{x}) - \boldsymbol{y})$$

Regarding the Hessian, $H(\boldsymbol{x}) = \nabla_{\boldsymbol{x}^T} g(\boldsymbol{x}) = \nabla_{\boldsymbol{x}^T} \boldsymbol{A}^T(h(\boldsymbol{A}\boldsymbol{x}) - \boldsymbol{y}) = \boldsymbol{A}^T \nabla_{\boldsymbol{x}^T} h(\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{A}^T \begin{bmatrix} \nabla_{\boldsymbol{x}^T} h\left(A_{1,:}\boldsymbol{x}\right) \\ \vdots \\ \nabla_{\boldsymbol{x}^T} h\left(A_{N,:}\boldsymbol{x}\right) \end{bmatrix} = \boldsymbol{A}^T \begin{bmatrix} h\left(A_{1,:}\boldsymbol{x}\right)\left(1 - h\left(A_{1,:}\boldsymbol{x}\right)\right) A_{1,:} \\ \vdots \\ h\left(A_{1,:}\boldsymbol{x}\right)\left(1 - h\left(A_{1,:}\boldsymbol{x}\right)\right) A_{N,:} \end{bmatrix}$

(notice that $\boldsymbol{\nabla_x a^T x} = \boldsymbol{a}$ and $\boldsymbol{\nabla_{x^T} a^T x} = \boldsymbol{a^T}$)

Therefore, the **CE Hessian** is

$$H(\boldsymbol{x}) = \boldsymbol{A^T} R(\boldsymbol{x}) \boldsymbol{A}$$

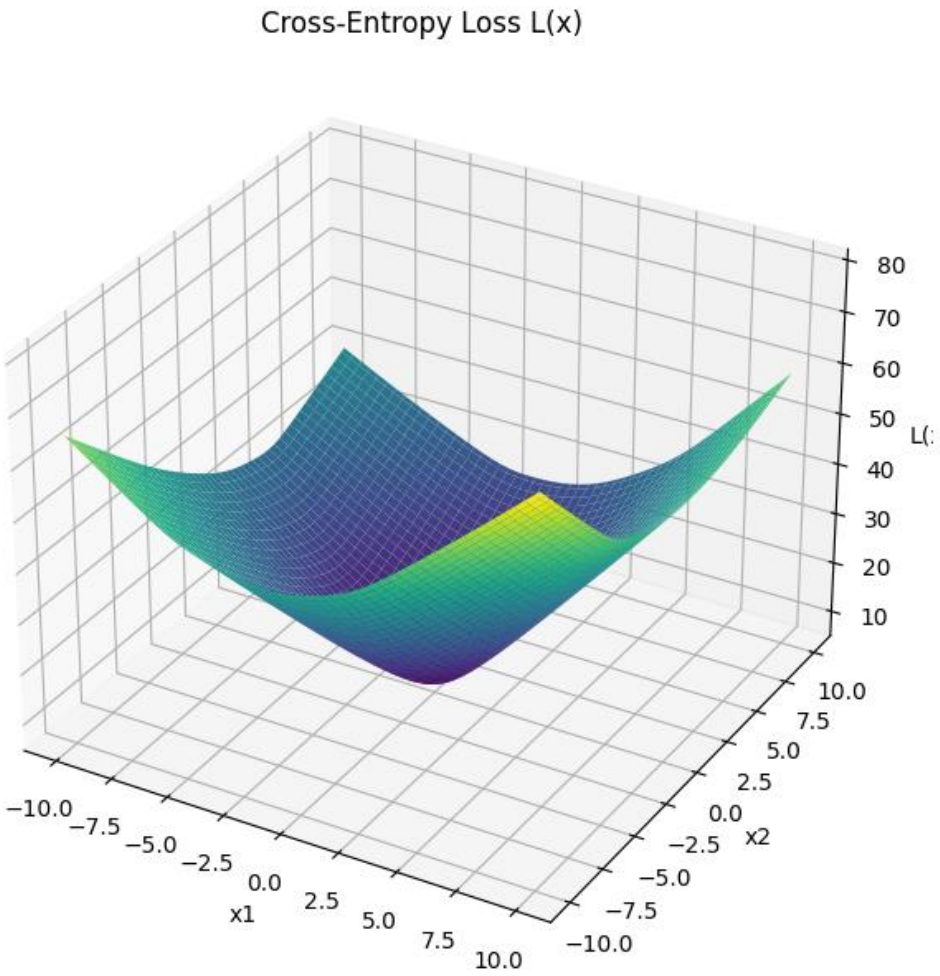
where

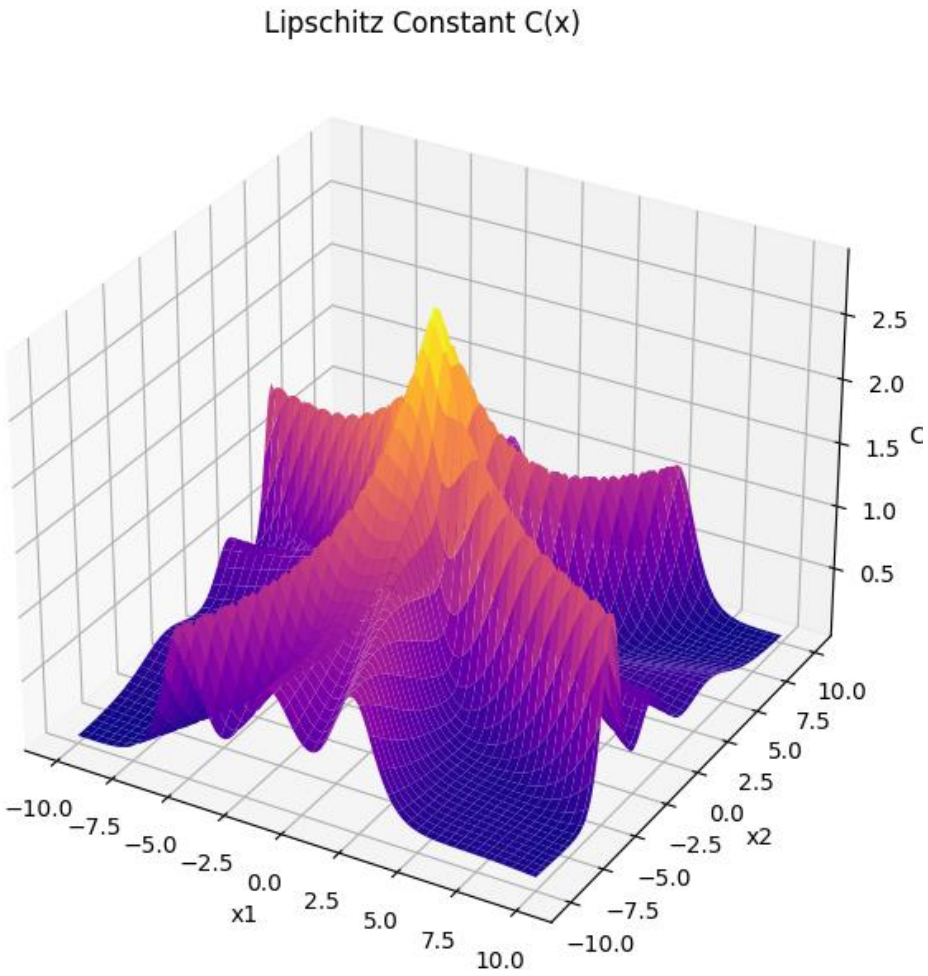
$$R(\boldsymbol{x}) = \mathrm{diag}\left(\left[h\left(A_{1,:}\boldsymbol{x}\right)\left(1 - h\left(A_{1,:}\boldsymbol{x}\right)\right), \ldots, h\left(A_{N,:}\boldsymbol{x}\right)\left(1 - h\left(A_{N,:}\boldsymbol{x}\right)\right)\right]\right)$$

The Lipschitz Constant of $L(\boldsymbol{x})$ at each point $\boldsymbol{x}$ is

$$C(x) = \sigma_{max}\left(\boldsymbol{A^T} R(\boldsymbol{x}) \boldsymbol{A}\right)$$

Cross-Entropy Minimization – Lipschitz





Cross-Entropy Minimization

How could we minimize CE?

No closed form minimizer. No closed form solution to system $g(\boldsymbol{x}) = \boldsymbol{A}^T(h(\boldsymbol{A}\boldsymbol{x}) - \boldsymbol{y}) = \boldsymbol{0}$

Can solve as:

1) GD with FSS. Set step size below $\frac{1}{C_{min}}$, where $C_{min} = \min_{\boldsymbol{x} \in R^D} C(x) = \min_{\boldsymbol{x} \in R^D} \sigma_{max}(H(\boldsymbol{x}))$.

   Can use instead $\hat{C}_{min} = \min_{\boldsymbol{x} \in S} \sigma_{max}(H(\boldsymbol{x}))$ where S is a subset of R^D

2) GD with ELS

3) GD with BLS

4) GD with Lipschitz constant steps; i.e., $\gamma_n = \sigma_{max}\left(H(\boldsymbol{x}_{n-1})\right)$

5) Newton's method (non-trivial, multiple iterations in contrast to LS)

...

and many more approaches such as stochastic variants of GD that we will talk about later on.