EECS 553 Lecture 06
Machine Learning (ECE)
[class version]

Jeff Fessler

University of Michigan
©Laura Balzano and Jeff Fessler, UM, September 16, 2026, 18:09

F26

Logistics

© 2026 Fessler

- HW1 solutions on HW solution server (link here and on Canvas)
- Python template for HW2 on gdrive (link here and on Canvas)
- HW3 released. Two autograder problems...
- iClicker scores on Canvas will *eventually* drop 4 lowest days
- Groups and "paper ideas" due Fri. Sep. 25!
  `project-team` label in Piazza.
  Prefer groups of 4! (75 students...)
- Details "paper ideas" format on Canvas

Recap / Outline

© 2026 Fessler

- Recap
  - Regression, MSE, MMSE, Regularized LS
  - ERM, surrogate losses
- Today
  - Unconstrained optimization fundamentals
    - Gradient
    - Lipschitz continuity
    - 1st-order approximation lemma
    - Local and global minimizers
    - Hessian
    - Convex functions
  - Optimization algorithms
    - Gradient descent
    - Convergence rates

Unconstrained optimization: motivation

© 2026 Fessler

- Many ML methods involve finding a minimizer of some cost function or a maximizer of some objective function.
- Often the cost function arises from a that defines the error metric we want to use, and the quantifies that error with respect to the joint distribution of $(\boldsymbol{X}, Y)$.
- An "ideal' (but usually impractical) approach is to design a predictor (classifier or regressor) by minimizing the risk

$$f_* = \arg\min_{f \in \mathcal{F}} R(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}\{L(Y, f(\boldsymbol{X}))\}$$

- Typically the set of predictors is parameterized:

$$\mathcal{F} = \{f_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathbb{R}^p\},$$

in which case we have a parametric **optimization problem**:

$$f_* = f_{\hat{\boldsymbol{\theta}}}, \quad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \quad J(\boldsymbol{\theta}) = \mathbb{E}\{L(Y, f_{\boldsymbol{\theta}}(\boldsymbol{X}))\}.$$

Unconstrained optimization: setup

© 2026 Fessler

- For example, in linear regression we use affine regressors:

$$\mathcal{F} = \{f \,:\, f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b,\ \boldsymbol{\theta} = (\boldsymbol{w}, b) \in \mathbb{R}^{\quad}\}.$$

- The (usually impractical) "ideal" approach would involve

$$\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{w}}, \hat{b}) = \underset{\boldsymbol{\theta}=(\boldsymbol{w},b)\in\mathbb{R}^{d+1}}{\arg\min}\ \mathbb{E}_{\boldsymbol{x},y}\left\{(\boldsymbol{w}^\top \boldsymbol{X} + b - Y)^2\right\}.$$

- In practice we use **empirical risk minimization** (ERM) with regularization:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \lambda\Omega(\boldsymbol{\theta}).$$

- Example: Regularized linear least squares regression:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}=(\boldsymbol{w},b)\in\mathbb{R}^{d+1}}{\arg\min}\ J(\boldsymbol{\theta}), \quad J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\qquad - y_i)^2 + \Omega(\boldsymbol{\theta}).$$

An **unconstrained optimization problem** has the general form

where $F : \mathbb{R}^p \mapsto \mathbb{R}$ is called the or

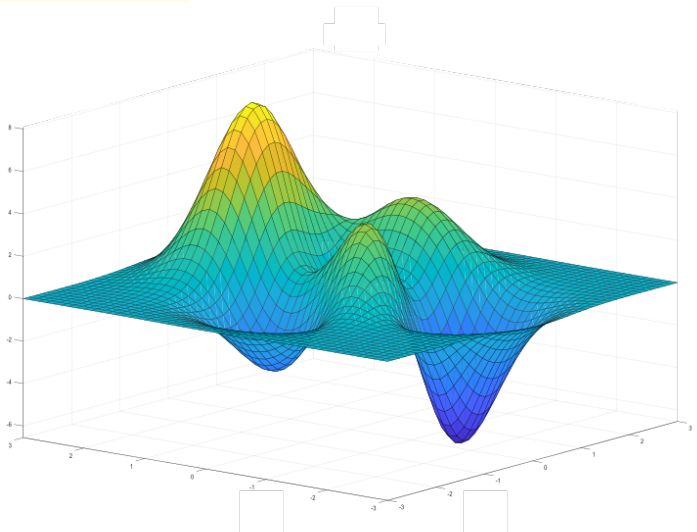


Image courtesy [2]

Key topics:
gradient methods, stationary points, convexity, non-convexity

ML examples of unconstrained optimization I

$$\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^p} F(\boldsymbol{\theta})$$

- Sparse regularized regression: $F(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1$
  where we are given $\boldsymbol{y}, \boldsymbol{X}$
  If the rows of $\boldsymbol{X}$ are $\boldsymbol{x}_i$, then we are given data $\{(\boldsymbol{x}_i, y_i)\}$.
- Logistic regression (for binary outputs $y = 0, 1$):
  $F(\boldsymbol{\theta}) = \sum_{i=1}^n -y_i \log\Big(\phi(\boldsymbol{x}_i^\top\boldsymbol{\theta})\Big) - (1-y_i)\log\Big(1-\phi(\boldsymbol{x}_i^\top\boldsymbol{\theta})\Big)$
  where $(\boldsymbol{x}_i, y_i)$ are our data and $\phi(z) \triangleq \frac{1}{1+e^{-z}}$ denotes the **standard logistic function**.
- PSD matrix completion: $F(\boldsymbol{\Theta}) = \sum_{(i,j)\in\Omega} |a_{ij} - (\boldsymbol{\Theta}\boldsymbol{\Theta}^\top)_{ij}|^2$
  where Ω is a subset of observed entries in the matrix $\boldsymbol{A}$.

ML examples of unconstrained optimization II

- Neural network optimization is generally formed as unconstrained nonconvex optimization:

$$F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} h(\boldsymbol{y}_i, \underbrace{\phi_L(\boldsymbol{W}_L(\phi_{L-1}(\boldsymbol{W}_{L-1}(\cdots(\phi_1(\boldsymbol{W}_1(\boldsymbol{x}_i)))\cdots)))))}_{}$$

  and, for example,
  - h is some loss function, *e.g.*, cross entropy between y and the second argument (output of the NN),
  - $\phi_1, \ldots, \phi_L$ are nonlinear functions in L layers such as ReLU, max pooling, or softmax, and
  - $\boldsymbol{\theta} = (\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L)$ are weight matrices of L layers (often with structure, e.g., convolutional layers; they are also sometimes functions of the input to that layer, *e.g.*, transformer layers).
- Here $\boldsymbol{\theta} \in \mathbb{R}^p$ where p can be very large.

Notation ambiguity

© 2026 Fessler

- When discussing optimization, we may use $J(\boldsymbol{\theta})$ or $F(\boldsymbol{\theta})$ or $F(\boldsymbol{x})$ or $f(\boldsymbol{x})$ to describe optimizing a function of a variable.
- In this context $\boldsymbol{x}$ is not data, but plays the role of the optimization variable.
- From context, determine whether $f(\boldsymbol{x})$ denotes a cost function, or a classification method.

Unconstrained optimization: context

© 2026 Fessler

- UM has several (general purpose?) optimization-oriented courses
  - ECE 559: Optimization methods for signal processing and ML
  - EECS/IOE 600: Function space methods in system theory
  - IOE 611: Nonlinear programming (aka, convex optimization)
  - Math 562 / IOE 511: Continuous optimization methods
  - Stats 606: Computation and optimization methods in statistics
- Yet we include in 553, and ML venues continue to have numerous optimization papers. Why?
  - large
  - large
  -
  -

Unconstrained optimization: key issues

© 2026 Fessler

- **Cost function** considerations
  - Minimizer
  - Minimizer
  -
  - Landscape: saddle points? If so, benign (strict) or degenerate?
  -
- **Algorithm** considerations
  - Convergence:
  - Convergence rate
  - Per iteration
  -
  - Tuning parameters
-

- Begin with **differentiable** cost functions for simplicity.
- For a **differentiable** cost function $F : \mathbb{R}^p \mapsto \mathbb{R}$, the **gradient** of F at $\boldsymbol{x} = [x_1 \ \cdots \ x_p]^\top \in \mathbb{R}^p$ is defined by:

$$\nabla F(\boldsymbol{x}) \triangleq$$

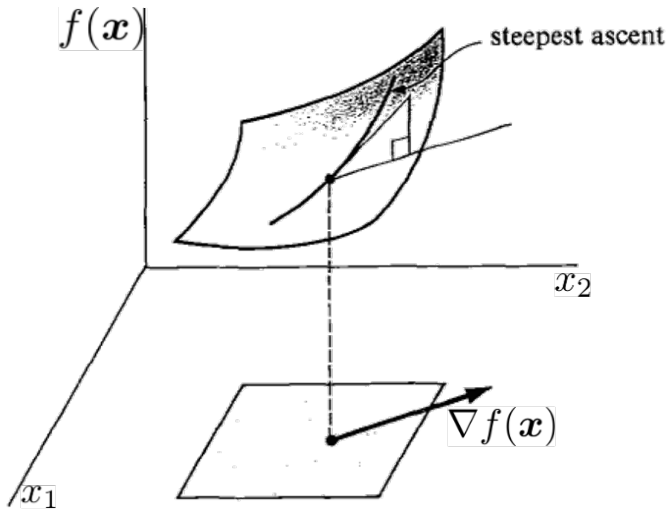


Image courtesy Clay Scott

- The gradient gives the direction of

For a 1d **differentiable** function $F : \mathbb{R} \mapsto \mathbb{R}$, **Taylor's remainder theorem** states

$$F(y) = F(x) + \int_x^y F'(z) \, \mathrm{d}z,$$

where F' denotes the derivative in **Lagrange's notation**

With the change of variable $z = tx + (1-t)y$ for $t \in [0, 1]$, this can equivalently be written as

$$F(y) = F(x) +$$

Theorem

*(**Multivariate remainder**) For a differentiable function $F : \mathbb{R}^p \mapsto \mathbb{R}$ we have for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$:*

$$F(\boldsymbol{y}) = F(\boldsymbol{x}) + \int_0^1 \langle \boldsymbol{y} - \boldsymbol{x}, \nabla F(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \rangle \, \mathrm{d}t \qquad (1)$$

where $\langle \cdot, \cdot \rangle$ denotes the standard **inner product** in $\mathbb{R}^p$.

Interpretation: the change in the function value equals the *aggregated projection of gradients along the path.*

Definition (Lipschitz)

- A function $f : \mathbb{R}^N \mapsto \mathbb{R}^M$ is called L-Lipschitz iff for all inputs $\boldsymbol{x}, \boldsymbol{y}$ in the domain, it obeys

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\| .$$

  We will use Euclidean norms unless otherwise specified.
- A function is called **Lipschitz continuous** or just **Lipschitz** iff there exists an $L < \infty$ for which it is L-Lipschitz.
- A function is called **locally Lipschitz** over a set $\mathcal{D} \subset \mathbb{R}^N$ if it is Lipschitz when restricted to $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{D}$.

Q06.1 Which function $f(x)$ is *not* **Lipschitz** continuous over $\mathbb{R}$?

A: 0 B: x C: $|x|$ D: $-|x|$ E: x^2 **??**

- If $F : \mathbb{R}^p \mapsto \mathbb{R}$ is differentiable and if its gradient is uniformly bounded, *i.e.*, $\|\nabla F(\boldsymbol{x})\| \leq L$ for all $\boldsymbol{x} \in \mathbb{R}^p$, then F is L-Lipschitz.
- Those are sufficient but not necessary conditions, *e.g.*, $f(x) = |x|$ is 1-Lipschitz but it is not differentiable.

Definition

A function $F : \mathbb{R}^p \mapsto \mathbb{R}$ is called ***L*-smooth** iff its **gradient** is ***L*-Lipschitz**:

$$\|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\| \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p \, .$$

- The choice of the norms above affects L, but does not affect whether it is smooth or not.
- Not all smooth functions are Lipschitz, *e.g.*, $f(x) = x^2$ is smooth, but not Lipschitz continuous.

1st-order approximation lemma I

Lemma. If $F : \mathbb{R}^p \mapsto \mathbb{R}$ is differentiable and L-smooth, then $\forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^p$:

$$\left| F(\boldsymbol{x}) - \underbrace{\left(F(\boldsymbol{z}) + \langle \nabla F(\boldsymbol{z}), \boldsymbol{x} - \boldsymbol{z} \rangle\right)}_{} \right| \leq \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{z}\|_2^2 .$$

Proof: Recall $F(\boldsymbol{x}) = F(\boldsymbol{z}) + \int_0^1 \langle \boldsymbol{x} - \boldsymbol{z}, \nabla F(t\boldsymbol{z} + (1-t)\boldsymbol{x}) \rangle \, \mathrm{d}t$.
By the 1st-order Taylor expansion (1), we can write:

$$\begin{aligned} e(\boldsymbol{x}; \boldsymbol{z}) &\triangleq F(\boldsymbol{x}) - \left(F(\boldsymbol{z}) + \langle \nabla F(\boldsymbol{z}), \boldsymbol{x} - \boldsymbol{z} \rangle\right) \\ &= \int_0^1 \langle \boldsymbol{x} - \boldsymbol{z}, \nabla F(t\boldsymbol{z} + (1-t)\boldsymbol{x}) - \nabla F(\boldsymbol{z}) \rangle \, \mathrm{d}t . \end{aligned}$$

Apply **Cauchy-Schwarz inequality**:

$$e(\boldsymbol{x}; \boldsymbol{z}) \leq \int_0^1 \|\boldsymbol{x} - \boldsymbol{z}\|_2 \, \|\nabla F(t\boldsymbol{z} + (1-t)\boldsymbol{x}) - \nabla F(\boldsymbol{z})\|_2 \, \mathrm{d}t .$$

1st-order approximation lemma II

Apply **Lipschitz continuity** of ∇F:

$$\|\nabla F(t\boldsymbol{z} + (1-t)\boldsymbol{x}) - \nabla F(\boldsymbol{z})\|_2 \leq L(1-t)\,\|\boldsymbol{x} - \boldsymbol{z}\|_2$$

to upper bound the RHS:

$$e(\boldsymbol{x};\boldsymbol{z}) \leq \|\boldsymbol{x} - \boldsymbol{z}\|_2^2\, L \int_0^1 (1-t)\,\mathrm{d}t = \frac{L}{2}\,\|\boldsymbol{x} - \boldsymbol{z}\|_2^2\,. \qquad \square$$

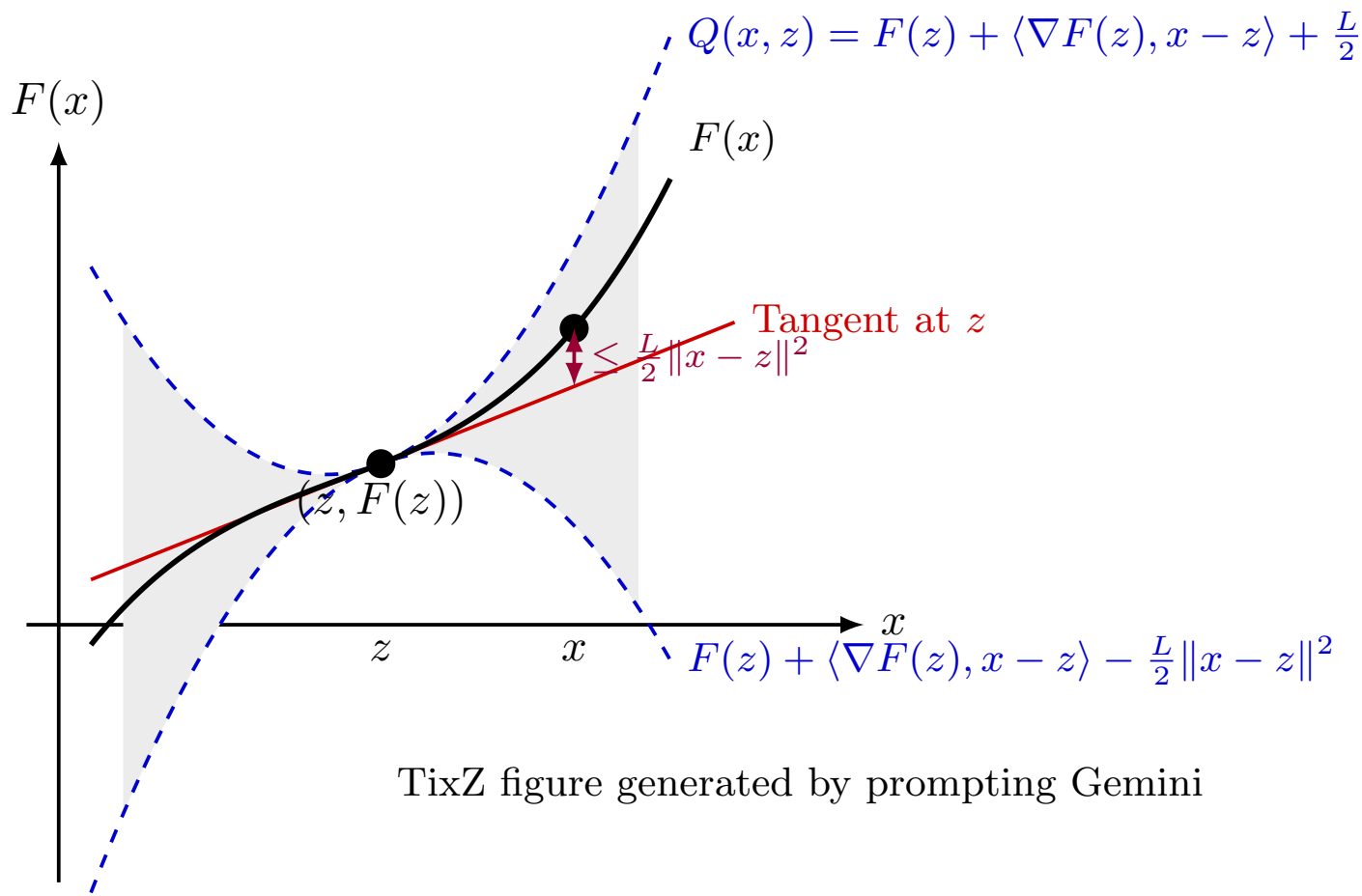
Thus we have the following bound on the accuracy of a 1st-order approximation of a smooth function:

$$\left|F(\boldsymbol{x}) - \underbrace{\left(F(\boldsymbol{z}) + \langle \nabla F(\boldsymbol{z}), \boldsymbol{x} - \boldsymbol{z}\rangle\right)}_{\text{1st-order approximation}}\right| \leq \frac{L}{2}\,\|\boldsymbol{x} - \boldsymbol{z}\|_2^2\ .$$

Useful for some optimization algorithm derivations, *e.g.*, proximal optimization methods for 1-norm regularizers, *e.g.*, for sparse regression and sparse logisitic regularization.

Descent lemma: illustration

© 2026 Fessler



TixZ figure generated by prompting Gemini

Key property: upper bound $Q(\boldsymbol{x}; \boldsymbol{z})$ is quadratic in $\boldsymbol{x}$ for given $\boldsymbol{z}$.
Some **majorize-minimize** (**MM**) algorithms exploit this property.

Let $F : \mathbb{R}^p \mapsto \mathbb{R}$ be a cost function.

- $\boldsymbol{x}_*$ is a **local minimizer** of F iff there is a ball $\mathcal{B}_\epsilon$ of radius $\epsilon > 0$ centered at $\boldsymbol{x}_*$ for which

$$F(\boldsymbol{x}_*) \leq F(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathcal{B}_\epsilon$$

  i.e., $\forall \boldsymbol{x} \; : \; \|\boldsymbol{x} - \boldsymbol{x}_*\| < \epsilon$.

- $\boldsymbol{x}_*$ is a **global minimizer** of F iff

$$F(\boldsymbol{x}_*) \leq F(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbb{R}^p.$$

- Global minimizers need not be unique in general. Example:
- Differentiability of F can start to help us identify (or at least rule out) candidate local minimizers.

- If F is differentiable and $\boldsymbol{x}^*$ is a **local minimizer** of F, then

$$\nabla F(\boldsymbol{x}^*) = \mathbf{0}.$$

- Note that $\nabla F(\boldsymbol{x}^*) = \mathbf{0}$ is necessary, but *not* sufficient for $\boldsymbol{x}^*$ to be a local minimizer of a differentiable function.
- If F is differentiable and $\nabla F(\boldsymbol{x}) = \mathbf{0}$ for some $\boldsymbol{x}$, then $\boldsymbol{x}$ is called a **stationary point** of F.
- If F is not differentiable everywhere, then wherever $\nabla F(\boldsymbol{x}) = \mathbf{0}$ or $\nabla F(\boldsymbol{x})$ is undefined, $\boldsymbol{x}$ is called a **critical point** of F.

- The Hessian matrix a real-valued function F quantifies its local curvature.
- It is useful for establishing optimality conditions.
- If $F : \mathbb{R}^p \mapsto \mathbb{R}$ is twice differentiable at $\boldsymbol{x} \in \mathbb{R}^p$, then the **Hessian matrix** of F at $\boldsymbol{x}$ is the $p \times p$ matrix:

$$\nabla^2 F(\boldsymbol{x}) \triangleq \begin{bmatrix} \frac{\partial^2 F(\boldsymbol{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 F(\boldsymbol{x})}{\partial x_1 \partial x_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F(\boldsymbol{x})}{\partial x_p \partial x_1} & \cdots & \frac{\partial^2 F(\boldsymbol{x})}{\partial x_p^2} \end{bmatrix}.$$

- If F is twice continuously differentiable at a point $\boldsymbol{x}$, then at that point: $\nabla^2 F(\boldsymbol{x})$ is a

Q06.2 Square matrix $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ is **positive semi-definite** (**PSD**) iff

A: $a_{ij} \geq 0$ for all $i, j = 1, \ldots, d$.

B: $\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$.

C: $\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^p$.

D: $\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{y} \geq 0$ for all $\boldsymbol{x} \geq 0, \boldsymbol{y} \geq 0$.

E: none of these

2nd-order necessary condition

© 2026 Fessler

- If F is twice continuously differentiable and $\boldsymbol{x}^*$ is a **local minimizer**, then $\nabla^2 F(\boldsymbol{x}^*)$ is **positive semi-definite**.
- This fact generalizes the result from single-variable calculus that the second derivative is nonnegative at a local minimizer.

Q06.3 Which is an example of a function $F : \mathbb{R} \mapsto \mathbb{R}$ and a **critical point** $\tilde{x}$ such that $\nabla^2 F(\tilde{x})$ is PSD but $\tilde{x}$ is not a local minimizer?

A: $F(x) = x^2$, $\tilde{x} = 0$

B: $F(x) = x^3$, $\tilde{x} = 0$

C: $F(x) = (x-3)^2$, $\tilde{x} = 0$

D: $F(x) = \cos(x)$, $\tilde{x} = 0$

E: none of these

For $\boldsymbol{x} \triangleq \begin{bmatrix} x \\ y \end{bmatrix}$, consider the function

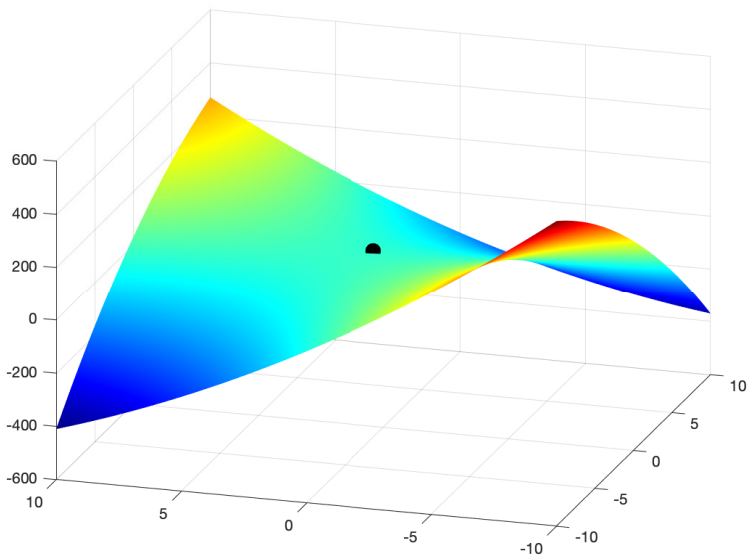
$$f(\boldsymbol{x}) = f(x, y) = x^2 + 4xy - y^2 - 8x - 6y + 10.$$

1. Determine $\nabla f(\boldsymbol{x}) =$
2. Determine $\nabla^2 f(\boldsymbol{x}) =$
3. Determine a **critical point** $\boldsymbol{x}^*$:

Q06.4 For this $f(\boldsymbol{x})$, the critical point $\boldsymbol{x}^*$ is a?
A: local maximizer B: local minimizer C: global minimizer D: global maximizer E: none of these

For $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2$, consider the function

$$f(\boldsymbol{x}) = f(x, y) = x^2 + 4xy - y^2 - 8x - 6y + 10.$$



- We say that $F : \mathbb{R}^p \mapsto \mathbb{R}$ is a **convex function** iff

$$F(t\boldsymbol{x} + (1-t)\boldsymbol{y}) \leq tF(\boldsymbol{x}) + (1-t)F(\boldsymbol{y}),$$
$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p \text{ and } t \in [0,1].$$

- We say $F : \mathbb{R}^p \mapsto \mathbb{R}$ is **strictly convex** iff

$$F(t\boldsymbol{x} + (1-t)\boldsymbol{y}) < tF(\boldsymbol{x}) + (1-t)F(\boldsymbol{y})$$
$$\forall \boldsymbol{x} \neq \boldsymbol{y} \text{ and } t \in (0,1).$$

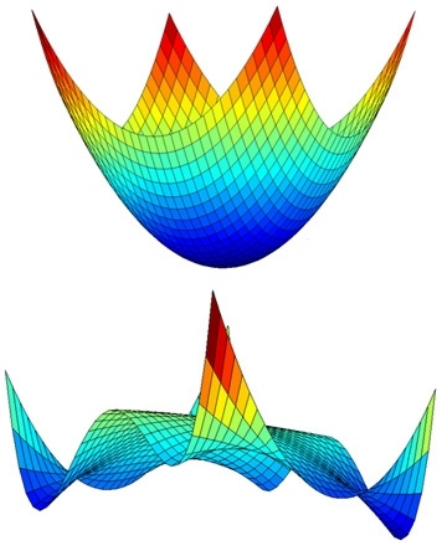


Image courtesy [3]
top: strictly convex
bottom: non-convex

CQ: Convex functions

© 2026 Fessler

Q06.5 If a function F is convex, then:

A: Every **local minimizer** is a **global minimizer**.

B: F has at most one **local minimizer** that is also the **global minimizer**.

C: Every stationary point is a **local maximum**.

D: F may have **saddle points**.

E: A & D

Suppose $\boldsymbol{x}$ is a

Then there exists $\boldsymbol{y}$ such that $F(\boldsymbol{y}) < F(\boldsymbol{x})$.

Using convexity this implies for any $0 < \epsilon < 1$:

$$F(\underbrace{\boldsymbol{x} + \epsilon\,(\boldsymbol{y} - \boldsymbol{x})}_{\tilde{\boldsymbol{x}}}) = F((1-\epsilon)\,\boldsymbol{x} + \epsilon\,\boldsymbol{y})$$

$$\leq (1-\epsilon)F(\boldsymbol{x}) + \epsilon F(\boldsymbol{y})$$

$$= F(\boldsymbol{x}) + \epsilon\,(F(\boldsymbol{y}) - F(\boldsymbol{x})) < F(\boldsymbol{x}).$$

This strict inequality provides a contradiction, by violating the definition of □

Also, $\boldsymbol{y} - \boldsymbol{x}$ is a strict **descent direction** at $\boldsymbol{x}$ (defined soon).

Raise your hand: Convex functions

© 2026 Fessler

Give an example of a function $f : \mathbb{R} \mapsto \mathbb{R}$ that is

- **convex** but not **strictly convex**
- **convex** and has more than one **global minimizer**
- **strictly convex** but has no **global minimizer**

CQ: Convex function combinations

© 2026 Fessler

Q06.6 Consider these three statements in this order. Are they true or false?

- The sum of convex functions is necessarily convex.
- The product of convex functions is necessarily convex.
- The composition of convex functions is necessarily convex.

A: True, True, False

B: True, False, True

C: False, True, False

D: True, False, False

E: none of these

1st- and 2nd-order conditions revisited

© 2026 Fessler

- Using convexity we can make stronger assertions about optimality conditions for differentiable functions.
  - If F is a **convex function**, then $\nabla F(\boldsymbol{x}^*) = \mathbf{0}$ is both necessary *and* sufficient for $\boldsymbol{x}^*$ to be a
  - If F is a **strictly convex function**, then $\nabla F(\boldsymbol{x}^*) = \mathbf{0}$ is both necessary *and* sufficient for $\boldsymbol{x}^*$ to be a

- For twice differentiable functions we have some (incomplete!) relationships between convexity and the Hessian.
  - F is convex $\iff \nabla^2 F(\boldsymbol{x})$ is positive semidefinite $\forall \boldsymbol{x} \in \mathbb{R}^p$
  - F is strictly convex $\nabla^2 F(\boldsymbol{x})$ is positive definite $\forall \boldsymbol{x} \in \mathbb{R}^p$

Other function/optimization problem properties © 2026 Fessler

- **Quasiconvex functions** [4]
- Functions that satisfy the **Kurdyka-Łojasiewicz inequality** [5]
- **self-concordant functions**
- Difference-of-Convex (DC) functions [6]

Modern optimization theory for ML tries to understand when non-convex functions are locally "nice" for optimization or have nice "landscape" properties that make them agreeable to (sub-)gradient optimization techniques.

Popular ML optimization techniques

© 2026 Fessler

- First-order gradient methods
- Second-order gradient methods (more complex and more memory due to Hessian computation, but there are various ways to approximate this step)
- Stochastic gradient methods
  - Stochastic variance reduced gradient methods
  - Adaptive learning rate: RMSprop, Adam
  - Matrix orthogonalization: Muon

- We have seen many ML problems involve **unconstrained optimization** problems of the form

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^p} J(\boldsymbol{\theta}).$$

- The cost function is $J : \mathbb{R}^p \mapsto \mathbb{R}$.
- Focus first on differentiable cost functions where we can compute the **gradient**:

$$\nabla J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial}{\partial \theta_1} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_p} J(\boldsymbol{\theta}) \end{bmatrix} \in \mathbb{R}^p.$$

- A representative example is the regularized least-squares cost function

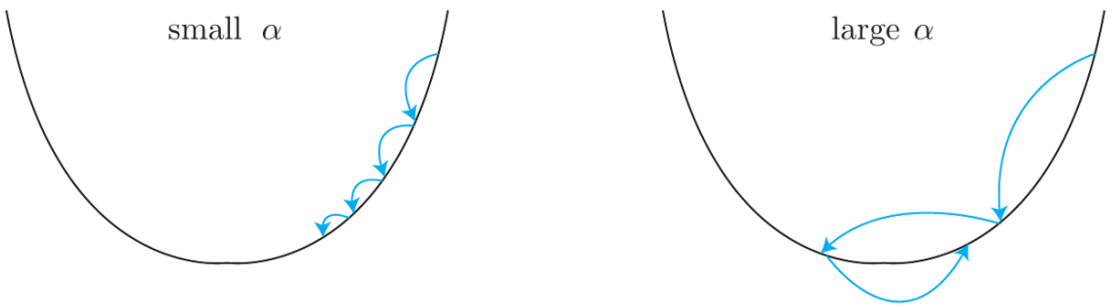
$$J(\boldsymbol{\theta}) = \frac{1}{2n} \|\bar{\boldsymbol{X}} \boldsymbol{\theta} - \boldsymbol{y}\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 .$$

- Consider minimizing a generic cost function $J(\boldsymbol{\theta})$
- Initial guess $\boldsymbol{\theta}_0$
- The **gradient descent** iteration is:
  For $t = 0, \ldots,$ `max_iter`

  $$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \nabla J(\boldsymbol{\theta}_t)$$

  If convergence condition satisfied,

  End
- Step-size matters:



Gradient descent for regularized linear regression

© 2026 Fessler

- The quadratically regularized least-squares (*i.e.*, ridge regression) cost function can be written (by expanding):

$$J(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} - \boldsymbol{r}^\top \boldsymbol{\theta} + c,$$

where

$$\boldsymbol{H} = \frac{1}{n}\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}} + \lambda \boldsymbol{I}, \quad \boldsymbol{r} = \frac{1}{n}\bar{\boldsymbol{X}}^\top \boldsymbol{y}, \quad c = \boldsymbol{y}^\top \boldsymbol{y}.$$

- Cost function gradient:

$$\nabla J(\boldsymbol{\theta}) =$$

- Compute complexity?

CQ: GD for regularized LS

© 2026 Fessler

Q06.7 What is the per-iteration compute complexity of GD?

A: $O(\max(d, n))$

B: $O(dn)$

C: $O(d^2 n)$

D: $O(dn^2)$

E: none of these

- vs
- If iterations needed, then same as matrix inversion.
- How many iterations are needed?

Sequence $\{\boldsymbol{x}_k\}_{k>0}$ **converges** to a limit $\boldsymbol{x}^*$, denoted $\boldsymbol{x}_k \to \boldsymbol{x}^*$, iff:

$$\lim_{k \to \infty} \|\boldsymbol{x}_k - \boldsymbol{x}^*\| = 0 \ ,$$

i.e., $\forall \varepsilon > 0$, $\exists K_\varepsilon$ such that $k > K_\varepsilon \implies \|\boldsymbol{x}_k - \boldsymbol{x}^*\| < \varepsilon$. What norm do we use here?

The **convergence rate** of such a convergent sequence is

- **Q-linear** (aka "linear") iff:

- **Q-superlinear** iff: $\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} \to 0$,
  e.g., $x_k = 1/k!$ for which $|x_{k+1} - 0|/|x_k - 0| = k!/(k+1)! = 1/(k+1) \to 0$
- **Q-sublinear** iff: $\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} \to 1$,
  e.g., $x_k = 1/k^p$ for $p > 0$ for which $|x_{k+1} - 0|/|x_k - 0| = k^p/(k+1)^p = \left(\frac{k}{k+1}\right)^p \to 1$

References I

[1] K. P. Murphy.
Probabilistic machine learning: An introduction.
MIT Press, 2022.

[2] Alberto Pirrera.
Bistable structures for morphing applications using anisotropic shells.
PhD thesis, Univ. of Bristol, 01 2011.

[3] Larry Hardesty.
After almost 20 years, math problem falls, 2011.
MIT News: https://news.mit.edu/2011/convexity-0715.

[4] David Eppstein.
Quasiconvex programming.
Combinatorial and Computational Geometry, 52(287-331):3, 2005.

[5] Jérôme Bolte, Aris Daniilidis, and Adrian Lewis.
The łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems.
SIAM Journal on Optimization, 17(4):1205–1223, 2007.

[6] Le Thi Hoai An and Pham Dinh Tao.
The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems.
Annals of operations research, 133(1):23–46, 2005.

[7] Jim Belk.
8.2 planes and hyperplanes, 2021.
https://e.math.cornell.edu/people/belk/writing/LinearODEs8.pdf.

References II

[8] C. Bishop.
Pattern recognition and machine learning.
Springer, 2006.

[9] T. Serafini, G. Zanghirati, and L. Zanni.
Gradient projection methods for quadratic programs and applications in training support vector machines.
Optimization Methods and Software, 20(2-3):353–78, 2005.

[10] R. O. Duda, P. E. Hart, and D. G. Stork.
Pattern classification.
Wiley, New York, 2001.

[11] Onmyphd.
Karush–kuhn–tucker conditions inequality constraint diagram, 2013.
http://www.onmyphd.com/?p=kkt.karush.kuhn.tucker, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=27331272.

[12] Stephen Boyd and Lieven Vandenberghe.
Convex Optimization.
Cambridge University Press, 2004.

[13] Emmanuel Candes and Benjamin Recht.
Exact matrix completion via convex optimization.
Communications of the ACM, 55(6):111–119, 2012.

[14] David Gross.
Recovering low-rank matrices from few coefficients in any basis.
IEEE Transactions on Information Theory, 57(3):1548–1566, 2011.

[15] Ali Jalali, Yudong Chen, Sujay Sanghavi, and Huan Xu.
Clustering partially observed graphs via convex optimization.
In *ICML*, 2011.

References III

[16] Paul Hand.
Conditions for dual certificate existence in semidefinite rank-1 matrix recovery.
In *Proceedings of the 10th International Conference on Sampling Theory and Applications*, 2013.

[17] Xiaodong Li and Vladislav Voroninski.
Sparse signal recovery from quadratic measurements via convex programming.
SIAM Journal on Mathematical Analysis, 45(5):3019–3033, 2013.

[18] Qiuwei Li, Ashley Prater, Lixin Shen, and Gongguo Tang.
Overcomplete tensor decomposition via convex optimization.
In *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 53–56. IEEE, 2015.

[19] Stanley Osher and Wotao Yin.
Sparse recovery via ℓ_1 and l_1 optimization.
Notices of the International Congress of Chinese Mathematicians, 3(1):4–10, 2015.

[20] Carlos Fernandez-Granda.
Super-resolution of point sources via convex programming.
Information and Inference: A Journal of the IMA, 5(3):251–303, 2016.

[21] Takayuki Iguchi, Dustin G Mixon, Jesse Peterson, and Soledad Villar.
Probably certifiably correct k-means clustering.
Mathematical Programming, 165(2):605–642, 2017.

[22] Chong You, Zhihui Zhu, Qing Qu, and Yi Ma.
Robust recovery via implicit bias of discrepant learning rates for double over-parameterization.
Advances in Neural Information Processing Systems, 33:17733–17744, 2020.

[23] Shunta Akiyama and Taiji Suzuki.
On learnability via gradient method for two-layer relu neural networks in teacher-student setting.
In *International Conference on Machine Learning*, pages 152–162. PMLR, 2021.

References IV

[24] Kyle Gilman, Sam Burer, and Laura Balzano.
A semidefinite relaxation for sums of heterogeneous quadratics on the stiefel manifold.
arXiv preprint arXiv:2205.13653, 2022.