EECS 553 Lecture 07
Machine Learning (ECE)
[class version]

Jeff Fessler

University of Michigan
©Laura Balzano and Jeff Fessler, UM, September 21, 2026, 07:52

F26

Logistics

© 2026 Fessler

- Groups and "paper ideas" due Fri. Sep. 25 to Canvas.
- HW3 due Wed. Sep. 30
  For HW3#3a, assume features have Gaussian distributions.
  For HW3#3c, may use JULIA QDA demo code or PYTHON translation from HW2#6.

Recap / Outline

© 2026 Fessler

- Recap
  - Optimization fundamentals: gradient, Lipschitz, 1st-order approximation, local/global, convex
- Today
  - Unconstrained optimization fundamentals
    - Convex functions
  - Unconstrained optimization algorithms
    - Gradient descent
    - Convergence rates
  - Descent methods
    - Fixed point
    - Newton's method
    - Convergence: global / local
  - Stochastic gradient descent
    - Subgradient method

- 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

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

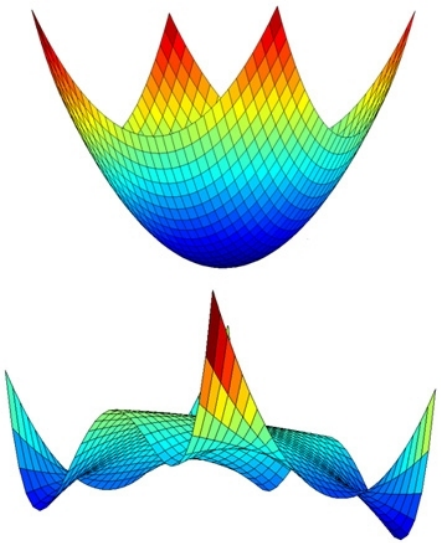


Image courtesy [2]
top: strictly convex
bottom: non-convex

CQ: Convex functions

© 2026 Fessler

Q07.1 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

Q07.2 Consider these three statements in this order. Are they true or false?

- Every sum of convex functions is convex.
- Every product of convex functions is convex.
- Every composition of convex functions is 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 $\nabla F(\boldsymbol{x}_*) = \boldsymbol{0}$ is both necessary *and* sufficient for $\boldsymbol{x}_*$ to be a
  - If F is a **strictly convex function**, then $\nabla F(\boldsymbol{x}_*) = \boldsymbol{0}$ 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** [3]
- Functions that satisfy the **Kurdyka-Łojasiewicz inequality** [4]
- **self-concordant functions**
- Difference-of-Convex (DC) functions [5]

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 methods

© 2026 Fessler

- First-order gradient (and subgradient) 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 [6]

- 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 for $p = d + 1$:

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

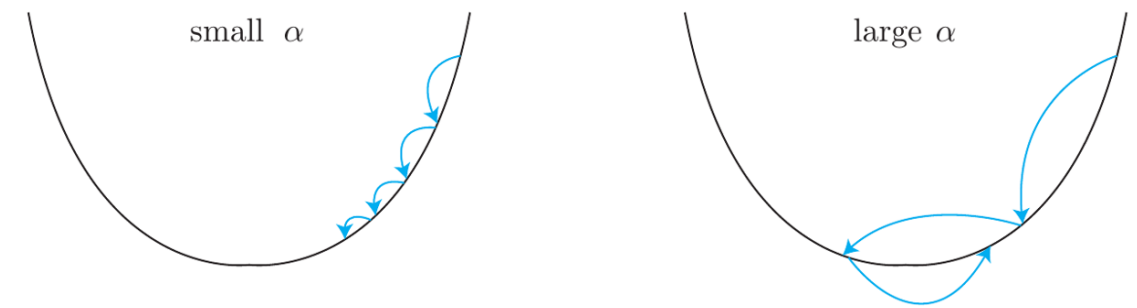
- Consider minimizing a differentiable cost function $J(\boldsymbol{\theta})$
- Start with an initial guess $\boldsymbol{\theta}_0$.
- The **gradient descent** (**GD**) iteration is recursive:
  For $t = 0, \ldots,$ `max_iter`

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

  If convergence condition satisfied,

  End

- Step-size matters:



JULIA code for seeking a minimizer of $f(\boldsymbol{x})$:

```
using LinearAlgebra: norm
function gd(∇f::Function, α::Real, x ; Niter::Int = 50, tol::Real = 1e-6)
    for iter in 1:Niter
        xnew = x - α * ∇f(x)
        if norm(xnew - x) < norm(x) * tol
            @show "stop at iteration $iter"
            return xnew
        else
            x = xnew
        end
    end
    return x
end
```

- This simple code causes several memory allocations each iteration.
- More efficient code would use in-place operations, *e.g.*, Optim.jl
- More professional code would have comments, a docstring, error checking ($\alpha > 0$), and handling of corner cases like $\|\boldsymbol{x}\| = 0$.

- 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

$$\boldsymbol{H} = \frac{1}{n}\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}} + \lambda \boldsymbol{I}, \quad \boldsymbol{r} = \frac{1}{n}\bar{\boldsymbol{X}}^\top \boldsymbol{y}, \quad c = \frac{1}{2n}\boldsymbol{y}^\top \boldsymbol{y}.$$

- Cost function gradient:

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

- Compute complexity of GD?

CQ: GD for regularized LS

© 2026 Fessler

Q07.3 What is the per-iteration compute complexity of GD for regularized LS regression? Hint. $\bar{\boldsymbol{X}}$ is $n \times (d+1)$.

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

B: $O(dn)$

C: $O(d^2 n)$

D: $O(dn^2)$

E: none of these

- vs for analytical solution: $\hat{\boldsymbol{\theta}} = \boldsymbol{H}^{-1}\boldsymbol{r} = (\bar{\boldsymbol{X}}^\top \bar{\boldsymbol{X}} + n\lambda \boldsymbol{I})^{-1} \bar{\boldsymbol{X}}^\top \boldsymbol{y}$.
- If iterations needed, then using GD requires similar computation as matrix "inversion" (via backslash).
- Number of iterations depends on **convergence rate**.

Sequence $\{\boldsymbol{x}_k\}$ **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 (quotient) **convergence rate** 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 $|x_{k+1} - 0|/|x_k - 0| = k^p/(k+1)^p = \left(\frac{k}{k+1}\right)^p \to 1$

Q07.4 For the sequence $\left\{ \begin{bmatrix} 1/(3^k+2) \\ 7/k! \end{bmatrix} \right\}$, the convergence rate is:

A: Q-linear with $\mu = 1/3$

B: Q-linear with $\mu = 2/3$

C: Q-sublinear

D: Q-superlinear

E: None of these

Hint: $\boldsymbol{x}_* = \mathbf{0}$.

Descent methods more generally

© 2026 Fessler

Goal: find a global minimizer $\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^p} J(\boldsymbol{\theta})$

using a simple iterative procedure like:

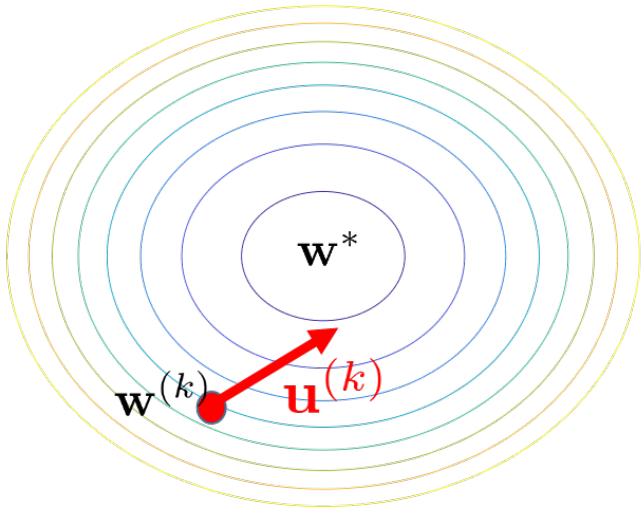
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \alpha \boldsymbol{u}_t.$$

- $\boldsymbol{u}_t$ is an **update** vector in $\mathbb{R}^p$, sometimes called a **search direction**
- $\alpha > 0$ is a **step size** parameter. Many options:
  - α : a predetermined constant
  - $\{\alpha_t\}$: a predetermined sequence
  - $\{\alpha_t\}$: found by a **back-tracking line search**
  - $\{\alpha_t\}$: other adaptive methods, often using **momentum**
- Definition: $\boldsymbol{u}$ is a **descent direction** at $\boldsymbol{\theta}$ iff

CQ: Descent direction

© 2026 Fessler

This figure shows contours of a **convex** cost function $F(\boldsymbol{w})$.



Constant contours $\{\mathbf{w} : F(\mathbf{w}) = c\}$

Q07.5 The $\boldsymbol{u}$ shown here is
[a descent descent direction, parallel to $-\nabla F(\boldsymbol{w}^{(k)})$] ??

A: (T,T) B: (T,F) C: (F,T) D: (F,F) **??**

Optimization method categories

© 2026 Fessler

Descent methods are categorized as:

- **Zeroth order**: $\boldsymbol{u}$ defined by cost function values $\{J(\boldsymbol{\theta}_t)\}$ for $J \in \mathcal{C}^0$

- **First order**: $\boldsymbol{u}$ defined by $\{\nabla J(\boldsymbol{\theta}_t)\}$ for $J \in \mathcal{C}^1$

  - For example, the **heavy ball** variant of GD uses both $\nabla J(\boldsymbol{\theta}_t)$ and $\nabla J(\boldsymbol{\theta}_{t-1})$ to define $\boldsymbol{u}_t$

- **Second order**: $\boldsymbol{u}$ defined by $\{\nabla J(\boldsymbol{\theta}_t),\ \nabla^2 J(\boldsymbol{\theta}_t)\}$ for $J \in \mathcal{C}^2$

- Consider minimizing the differentiable cost function $J(\boldsymbol{\theta})$ where $J : \mathbb{R}^p \mapsto \mathbb{R}$ is differentiable, the (first-order) GD method is:
  - Initial guess $\boldsymbol{\theta}_0 \in \mathbb{R}^p$
  - For $t = 1, \ldots,$ `max_iter`

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

    If convergence condition satisfied, exit

    End
- We can write the key step of this algorithm (and many algorithms) as a **recursive** mapping:

$$\boldsymbol{\theta}_{t+1} \leftarrow \mathcal{M}_{\mathsf{GD}}(\boldsymbol{\theta}_t), \quad \text{where} \quad \mathcal{M}_{\mathsf{GD}}(\boldsymbol{\theta}) \triangleq \boldsymbol{\theta} - \alpha \nabla J(\boldsymbol{\theta}).$$

- The **convergence** properties of the sequence $\{\boldsymbol{\theta}_t\}$ generated by the GD algorithm is governed by the properties of the mapping $\mathcal{M} : \mathbb{R}^p \mapsto \mathbb{R}^p$.

- In general, consider a recursive algorithm defined by a mapping function $\mathcal{M} : \mathbb{R}^p \mapsto \mathbb{R}^p$ as

$$\boldsymbol{\theta}_{t+1} = \mathcal{M}(\boldsymbol{\theta}_t).$$

- Any point $\bar{\boldsymbol{\theta}} \in \mathbb{R}^p$ such that

$$\bar{\boldsymbol{\theta}} = \mathcal{M}(\bar{\boldsymbol{\theta}})$$

is called a

- In general, the set of fixed points of a mapping can be anywhere from

Recall. $\boldsymbol{\theta}$ is called a **stationary point** of (differentiable) cost function J iff $\nabla J(\boldsymbol{\theta}) = \mathbf{0}$.

Q07.6 Every **stationary point** of a differentiable cost function J is a **fixed point** of the GD iteration mapping $\mathcal{M}_{\text{GD}}$ for minimizing J. The converse also holds. ??

A: (T,T) B: (T,F) C: (F,T) D: (F,F) **??**

Assume $J \in \mathcal{C}^2$, *i.e.*, it is twice continuously differentiable.
Newton's method (aka Newton-Raphson) for minimization selects update direction $\boldsymbol{u}_t$

$$\boldsymbol{\theta}_{t+1} =$$

Motivated by Taylor series when

Newton's update is identical to GD (for $\alpha = 1/a$) when contours of J are spherical

$$\nabla^2 J(\boldsymbol{\theta}) = a\boldsymbol{I}, \quad a > 0.$$

2nd-order Taylor approximation of twice-differentiable cost function J around $\boldsymbol{\theta}_t$:

$$J(\boldsymbol{\theta}) \approx J(\boldsymbol{\theta}_t) + \langle \nabla J(\boldsymbol{\theta}_t), \boldsymbol{\theta} - \boldsymbol{\theta}_t \rangle + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_t)^\top \nabla^2 J(\boldsymbol{\theta}_t)(\boldsymbol{\theta} - \boldsymbol{\theta}_t).$$

Minimizing the approximation by zeroing its gradient yields the Newton update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - [\nabla^2 J(\boldsymbol{\theta}_t)]^{-1} \nabla J(\boldsymbol{\theta}_t).$$

CQ: Newton's method

© 2026 Fessler

Q07.7 Assume J is a twice differentiable and strictly convex cost function.

Every **stationary point** of J is a **fixed point** of the Newton iteration.

The converse also holds. ??

A: (T,T) B: (T,F) C: (F,T) D: (F,F) **??**

Example: Quadratic cost function

© 2026 Fessler

$$\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \triangleq \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} - \boldsymbol{\theta}^\top \boldsymbol{b} + c =$$

Gradient: $\nabla J(\boldsymbol{\theta}) = \boldsymbol{H}\boldsymbol{\theta} - \boldsymbol{b}$ Hessian: $\nabla^2 J(\boldsymbol{\theta}) = \boldsymbol{H} \succ \mathbf{0}$

GD update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla J(\boldsymbol{\theta}_t)$$
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha (\boldsymbol{H}\boldsymbol{\theta}_t - \boldsymbol{b})$$
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{H} (\boldsymbol{\theta}_t - \boldsymbol{H}^{-1}\boldsymbol{b})$$
$$\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_* = (\boldsymbol{I} - \alpha \boldsymbol{H}) (\boldsymbol{\theta}_t - \boldsymbol{\theta}_*)$$

Newton update (with step size):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - [\nabla^2 J(\boldsymbol{\theta}_t)]^{-1} \nabla J(\boldsymbol{\theta}_t)$$
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{H}^{-1} (\boldsymbol{H}\boldsymbol{\theta}_t - \boldsymbol{b})$$
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha (\boldsymbol{\theta}_t - \boldsymbol{H}^{-1}\boldsymbol{b})$$

See [15, §9.2].

Example: Quadratic cost function

© 2026 Fessler

$$J(\boldsymbol{\theta}) = \tfrac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} - \boldsymbol{\theta}^\top \boldsymbol{b} + c$$

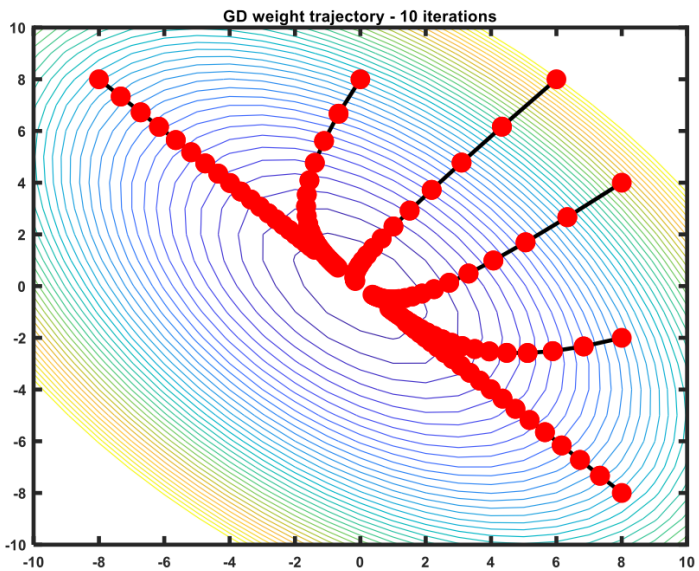
For this example,

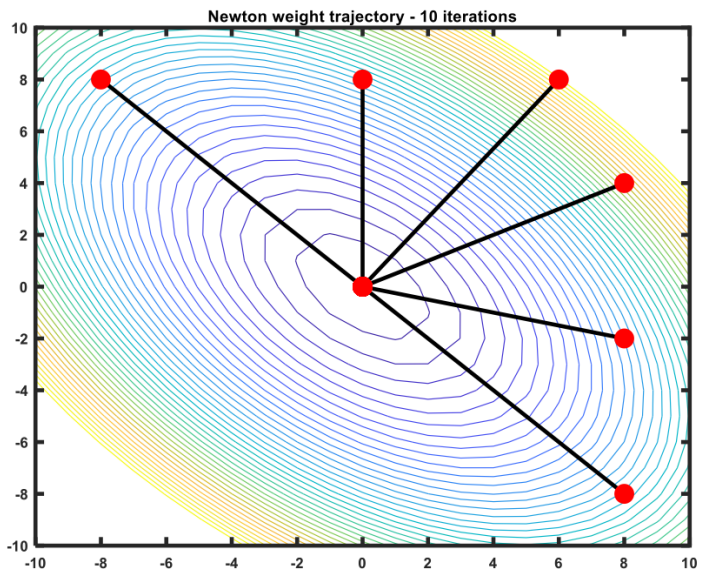
GD update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha\left(\boldsymbol{H}\boldsymbol{\theta}_t - \boldsymbol{b}\right)$$

Newton update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{H}^{-1}(\boldsymbol{H}\boldsymbol{\theta}_t - \boldsymbol{b})$$





Figures courtesy Al Hero, showing iterates from different $\boldsymbol{\theta}_0$. The axes are θ_1 and θ_2.

Example: Quadratic cost function

© 2026 Fessler

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

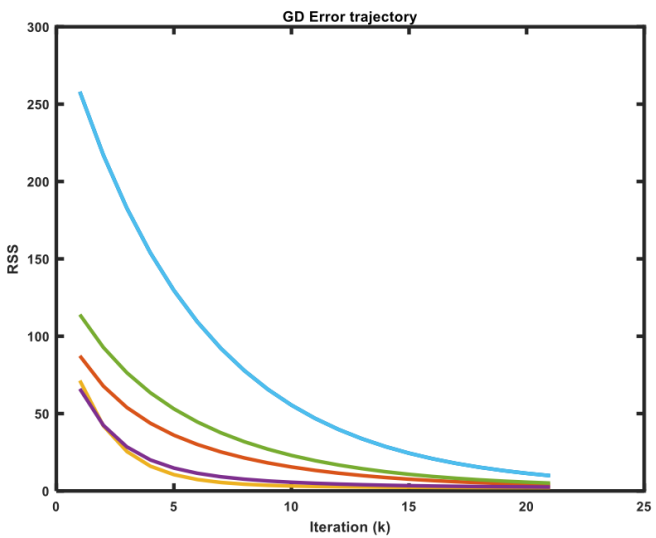
For this example, $\boldsymbol{\theta} \in \mathbb{R}^2$, $\text{eig}(\boldsymbol{H}) = \{\frac{1}{4}, \frac{1}{12}\}$. Step size $\alpha = 1$.

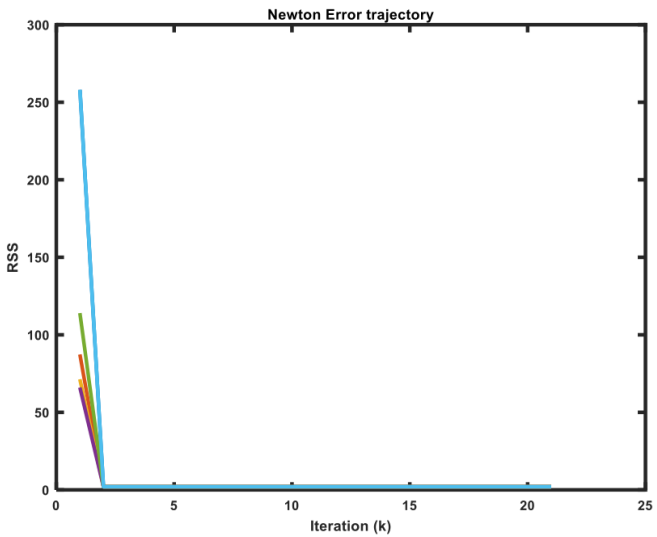
GD update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha\left(\boldsymbol{H}\boldsymbol{\theta}_t - \boldsymbol{b}\right)$$

Newton update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{H}^{-1}(\boldsymbol{H}\boldsymbol{\theta}_t - \boldsymbol{b})$$





- Vertical axis is $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_*\|_2$.
- Each curve is for a different $\boldsymbol{\theta}_0$.

Figures courtesy Al Hero;

From Newton to MM I

- In some applications (including logistic regression) we have a matrix upper bound

$$\nabla^2 J(\boldsymbol{\theta}) \preceq \boldsymbol{H}.$$

  This implies the following upper bound on the cost function:

$$J(\boldsymbol{\theta}) \leq Q(\boldsymbol{\theta}, \boldsymbol{\theta}_t) \triangleq J(\boldsymbol{\theta}_t) + \langle \nabla J(\boldsymbol{\theta}_t), \boldsymbol{\theta} - \boldsymbol{\theta}_t \rangle + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_t)^\top \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_t)$$

  This inequality looks akin to the 2nd-order Taylor approximation used in Newton's method.

- A majorize-minimize (MM) method updates $\boldsymbol{\theta}$ by minimizing the quadratic upper bound:

$$\boldsymbol{\theta}_{t+1} = \arg\min_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \boldsymbol{H}^{-1} \nabla J(\boldsymbol{\theta}_t).$$

From Newton to MM II

- To avoid performing matrix inversion every iteration, we precompute the inverse $\boldsymbol{H}^{-1}$ (just once) [1, §10.3.4].
- Multiplying by precomputed $\boldsymbol{H}^{-1}$ requires $O(p^2)$ FLOPs per iteration, whereas inverting the Hessian $\nabla^2 J(\boldsymbol{\theta}_t)$ is $O(p^3)$.
- If $\boldsymbol{H} = L\boldsymbol{I}$ for a Lipschitz constant L of the gradient ∇J, then the MM approach simplifies to

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{H}^{-1}\nabla J(\boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \frac{1}{L}\nabla J(\boldsymbol{\theta}_t).$$

ECE 551 uses this approach to logistic regression [15, §9.5].

CQ: Convergence of iterates

© 2026 Fessler

Q07.8 If we have a sequence $\boldsymbol{\theta}_t \to \boldsymbol{\theta}_*$ such that $\frac{\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_*\|}{\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_*\|} \to 0$, then it converges:

A: Linearly

B: Superlinearly

C: Sublinearly

D: Exponentially

E: None of these

"Convergence" of algorithms

© 2026 Fessler

- An algorithm's convergence is called **global** iff
- Convergence is **local** if
- The optimization community often bends terminology by discussing convergence of "algorithms" when what is really being discussed is convergence of the **sequence** $\{\boldsymbol{\theta}_t\}$ generated by the algorithm.
- These terms are distinct from convergence of a sequence to a **global minimizer** or **local minimizer** of some cost function, which is also sometimes specified, *e.g.*, "the algorithm generates a sequence that converges globally to a local minimizer."

Convergence properties of GD for quadratic J

© 2026 Fessler

For $J(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \boldsymbol{b} + c, \quad \boldsymbol{H} \succ \mathbf{0},$

GD has **global linear convergence** iff

Convergence condition $\gamma < 1$ requires

After K iterations of GD, the initial "error" $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}_*\| \triangleq \text{err}_0$

$$\text{err}_K \triangleq$$

To reduce error to at most ϵ at iteration K requires $\gamma^K \text{err}_0 \leq \epsilon$, *i.e.*, we need:

$$K \geq \log\left(\frac{\text{err}_0}{\epsilon}\right) \frac{1}{\log(1/\gamma)}.$$

Convergence of GD for (more) general J

© 2026 Fessler

If the gradient is

then using

$$\alpha =$$

guarantees progress by GD of [16]:

$$J(\boldsymbol{\theta}_{t+1}) \leq$$

More generally, if $0 < \alpha < 2/L$ then we have this (2025!) **tight upper bound** [17]:

$$J(\boldsymbol{\theta}_t) - J(\boldsymbol{\theta}_*) \leq \frac{L \left\| \boldsymbol{\theta}_0 - \boldsymbol{\theta}_* \right\|_2^2}{2} \max\left(\frac{1}{2L\alpha + 1}, (1 - L\alpha)^{2t} \right).$$

Effect of step size

© 2026 Fessler

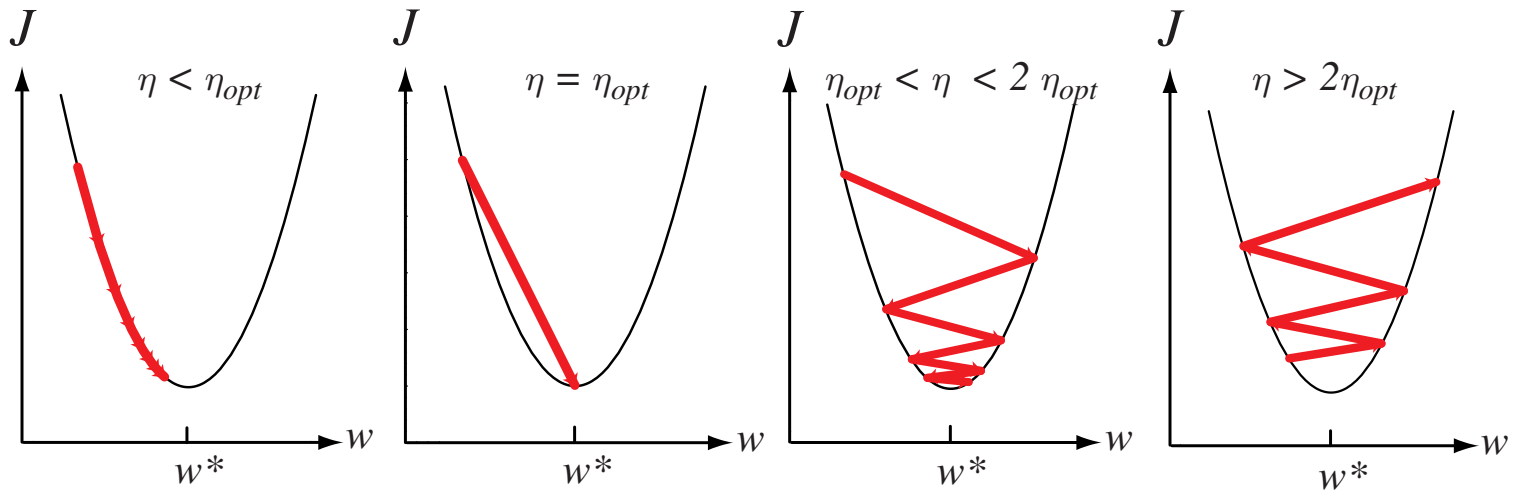


FIGURE 6.16. Gradient descent in a one-dimensional quadratic criterion with different learning rates. If $\eta < \eta_{opt}$, convergence is assured, but training can be needlessly slow. If $\eta = \eta_{opt}$, a single learning step suffices to find the error minimum. If $\eta_{opt} < \eta < 2\eta_{opt}$, the system will oscillate but nevertheless converge, but training is needlessly slow. If $\eta > 2\eta_{opt}$, the system diverges. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Figure from [18].

Convergence of Newton for locally quadratic J © 2026 Fessler

- **Local superlinear convergence** is possible if initialized near $\boldsymbol{\theta}_*$.
- **Newton's method** can diverge if not initialized close enough.
- Newton's method is also a **fixed-point iteration**:

$$\boldsymbol{\theta}_{t+1} = \qquad$$

where here

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \left[\nabla^2 J(\boldsymbol{\theta}_t)\right]^{-1} \nabla J(\boldsymbol{\theta}_t)$$

$$\implies \mathcal{M}(\boldsymbol{\theta}) = \qquad$$

- For non-quadratic $J(\boldsymbol{\theta})$, convergence of Newton's method is assured if (sufficient condition) the initial point $\boldsymbol{\theta}_0$ is in a sphere $\{\boldsymbol{\theta} : \|\boldsymbol{\theta} - \boldsymbol{\theta}_*\|_2 \leq r\}$ over which $\|\nabla \mathcal{M}(\boldsymbol{\theta})\|_2 < 1$.

- In ML, we often focus on cost functions (like ERM) that have an additive form:

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

where

- Or,

- **Stochastic gradient descent** (**SGD**) attempts to minimize the overall cost $J(\boldsymbol{\theta})$ by incrementally descending each J_i in the sum.

- Key property for SGD with expectation taken over random i:

$$\mathbb{E}\{\nabla J_i(\boldsymbol{\theta})\} =$$

Pseudo-code for minimizing $J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} J_i(\boldsymbol{\theta})$

- Initialize $\boldsymbol{\theta}_0$, set $t \leftarrow 0$
- For $j = 1, \ldots,$ `max_iter` **epoch** count

  For i `in permute` $(1, \ldots, n)$

  $\boldsymbol{\theta}_t \leftarrow$

  $t \leftarrow t + 1$

  End

  If convergence condition satisfied, exit

End

See also [7], incremental gradient descent [8], coordinate descent [9], Robbins-Monro algorithm [10] [11, p. 96].

Gradient descent v Stochastic gradient descent © 2026 Fessler

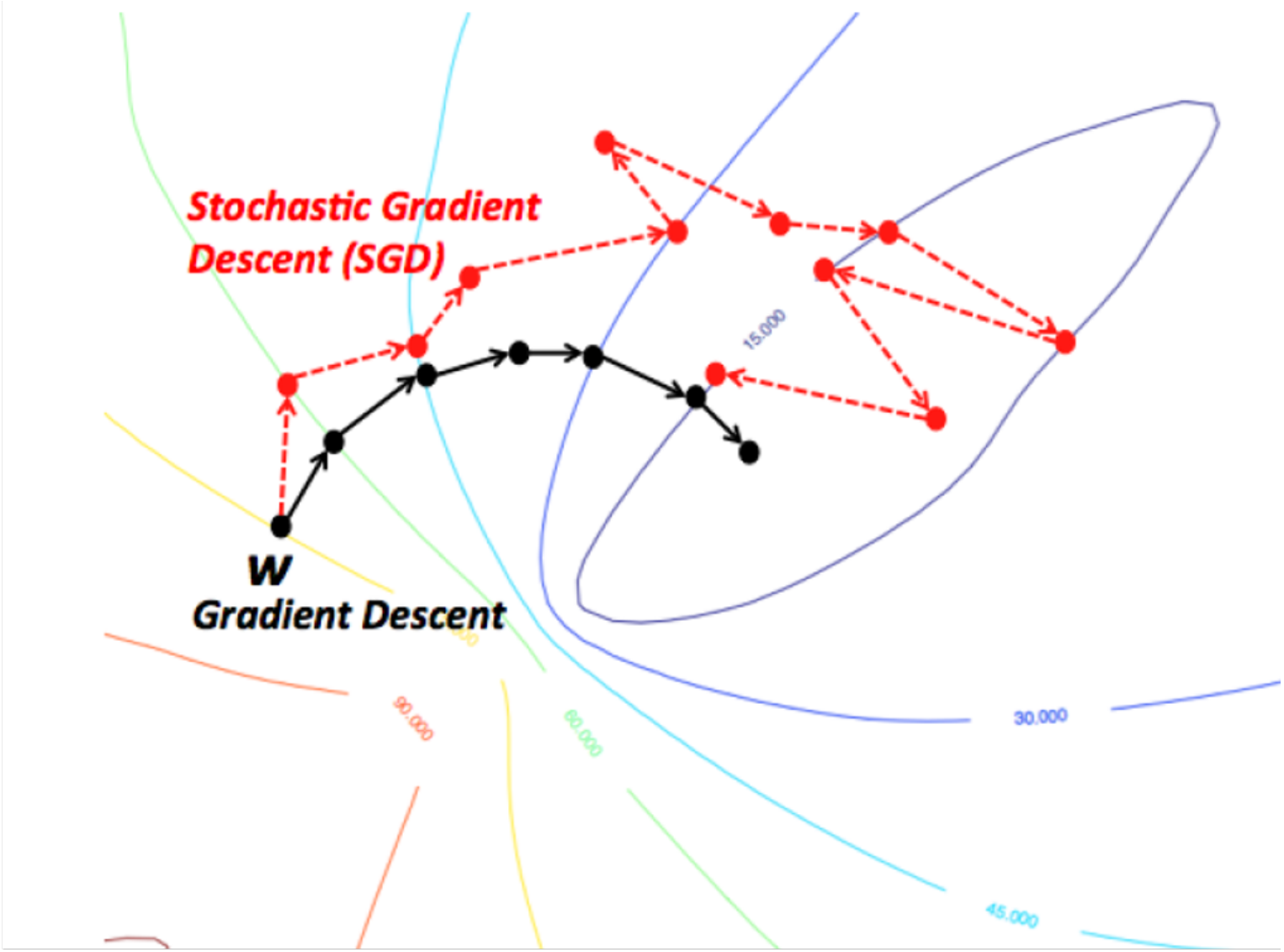


Image courtesy [12].

SGD: Momentum and variations

© 2026 Fessler

SGD variations:

- Stochastic variance reduced gradient methods
- Adaptive learning rate / momentum:
  - RMSprop [13]
  - **Adam** (adaptive moment estimation) [14]
- Matrix orthogonalization: Muon [6]

- The **subgradient method** is a generalization of gradient descent that applies to **nondifferentiable**, **convex** cost functions, like ERM with **hinge loss**.
- Let $J : \mathbb{R}^d \mapsto \mathbb{R}$ be convex, and let $\boldsymbol{\theta} \in \mathbb{R}^p$.
  If J is differentiable, then so far we've focused on the update direction

$$\boldsymbol{u} =$$

  which is the only vector such that this convexity inequality holds:

$$J(\boldsymbol{\theta}') \geq$$

Subgradient definition

© 2026 Fessler

- If J is convex but *not* differentiable, then for some $\boldsymbol{\theta}$, there may be many $\boldsymbol{u}$ satisfying the previous inequality.
- We define the **subdifferential** of J at $\boldsymbol{\theta}$, denoted $\partial J(\boldsymbol{\theta})$, to be the *set* of all $\boldsymbol{u}$ satisfying the inequality.
- A **subgradient** is any element of the **subdifferential**.

- In the figure, the subdifferential at the cusp is the interval $\left[J'_-(\theta), J'_+(\theta)\right]$ where $J'_-(\theta), J'_+(\theta)$ denote the left and right derivatives.

- A **subgradient method** replaces the gradient by *any* subgradient.
- Pseudo-code for minimizing $J(\boldsymbol{\theta})$:
  - initialize $\boldsymbol{\theta}_0$
  - $t \leftarrow 0$
  - Repeat:
    - select
    - $\boldsymbol{\theta}_t \leftarrow$
    - $t \leftarrow t + 1$
  - Until stopping criterion satisfied
- We can also have a **stochastic subgradient method**, analogous to SGD, when

$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} J_i(\boldsymbol{\theta})$$

Summary

© 2026 Fessler

- Least squares Linear regression: exact solution possible, but may be infeasible for large d
- Ridge penalty: makes objective strictly convex, solution unique
- Linear regression is the basis for several nonlinear regression methods
- Empirical risk minimization generalizes all we've seen so far.
- GD/SGD: efficient implementation
- SGD does *not* monotonically descend so the "'D" is a misnomer
- [19]

References I

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

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

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

[4] 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.

[5] 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.

[6] J. Bernstein and L. Newhouse.
Old optimizer, new norm: an anthology.
In *OPT 2024: Optimization for Machine Learning*, 2024.

[7] Léon Bottou.
Stochastic gradient descent tricks.
In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

References II

[8] Dimitri P Bertsekas et al.
Incremental gradient, subgradient, and proximal methods for convex optimization: A survey.
Optimization for Machine Learning, 2010(1-38):3, 2011.

[9] Stephen J Wright.
Coordinate descent algorithms.
Mathematical Programming, 151(1):3–34, 2015.

[10] H. Robbins and S. Monro.
A stochastic approximation method.
Ann. Math. Stat., 22(3):400–7, September 1951.

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

[12] Devin Soni.
Improving vanilla gradient descent.
Towards Data Science, 2018.
https://towardsdatascience.com/improving-vanilla-gradient-descent-f9d91031ab1d.

[13] Y. N. Dauphin, H. de Vries, J. Chung, and Y. Bengio.
Equilibrated adaptive learning rates for non-convex optimization, 2015.

[14] D. P. Kingma and J. Ba.
Adam: A method for stochastic optimization, 2014.

[15] J. A. Fessler and R. R. Nadakuditi.
Linear algebra for data science, machine learning, and signal processing.
Cambridge, 2024.

References III

[16] Sham Kakade.
Cse 546: Machine learning, lecture 9 on gradient descent, 2015.
https://courses.cs.washington.edu/courses/cse546/15au/lectures/lecture09_optimization.pdf.

[17] J. Kim.
A proof of the exact convergence rate of gradient descent, 2025.

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

[19] O. Gencoglu, M. van Gils, E. Guldogan, C. Morikawa, M. Suzen, M. Gruber, J. Leinonen, and H. Huttunen.
HARK side of deep learning – from grad student descent to automated machine learning, 2019.