EECS 553 Lecture 08
Machine Learning (ECE)
[class version]

Jeff Fessler

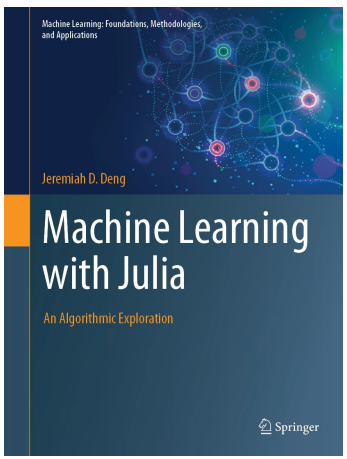
University of Michigan
©Laura Balzano and Jeff Fessler, UM, September 23, 2026, 22:35

F26

Logistics

© 2026 Fessler

- Project teams / paper ideas due Fri. Sep. 25
- After class meet-up for those still seeking teams?
- HW regrade requests due 3 days after scores released.
- HW3 due Wed. Sep. 30



[2] code

Q08.1 Which due date would you prefer for HW4?
Keep in mind 24-hour grace period and Exam 1 is Mon. Oct. 12.
A: Wed. Oct 8 B: Thu. Oct 9 C: Fri. Oct 10

Recap / Outline

© 2026 Fessler

- Recap
  - Convexity, convergence rates, descent methods, fixed point
- Today
  - Unconstrained optimization algorithms (continued)
    - Newton's method
    - Convergence: global / local
  - Stochastic gradient descent
    - Subgradient method
  - Logistic regression
    - Logistic regression: Likelihood
    - Logistic regression: Regularization

Assume $J \in \mathcal{C}^2$, *i.e.*, it is twice continuously differentiable.
Newton's method (aka Newton-Raphson) for minimization selects update direction $\boldsymbol{u}_t$ assuming $\nabla^2 J(\boldsymbol{\theta}_t)$ is invertible (or even positive definite) as a transformed steepest descent direction:

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

Motivated by Taylor series when $\boldsymbol{\theta}_t \approx \boldsymbol{\theta}_* =$ stationary point of J:

$$\nabla J(\boldsymbol{\theta}_*) = \mathbf{0} \approx \nabla J(\boldsymbol{\theta}_{t+1}) \approx \nabla J(\boldsymbol{\theta}_t) + \nabla^2 J(\boldsymbol{\theta}_t) (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t).$$

Newton's update is identical to GD (for $\alpha = 1/a$) in the rare cases where contours of J are spherical:

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

- Consider 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 this 2nd-order approximation by zeroing its gradient yields the Newton update:

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

-

CQ: Newton's method

© 2026 Fessler

Q08.2 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

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

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

GD update:

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

Convergence depends on
See [3, §9.2].

Newton update (with step size):

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

Convergence in one step if
(but we had to calculate $\boldsymbol{H}^{-1}$).

Example: Quadratic cost function

© 2026 Fessler

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

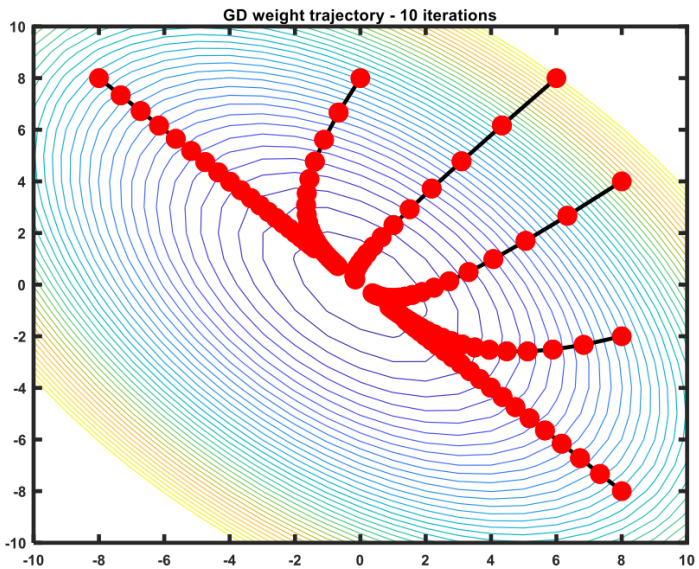
For this example, $\boldsymbol{\theta} \in \mathbb{R}^2$, $\text{eig}(\boldsymbol{H}) = \{\frac{1}{4}, \frac{1}{12}\}$. Step size $\alpha = 1$.

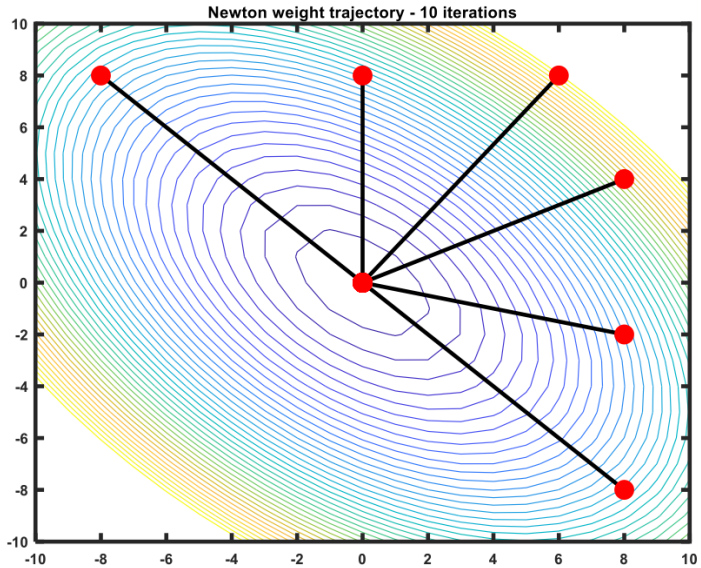
GD update:

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

Newton update:

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





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

Example: Quadratic cost function [Read]

© 2026 Fessler

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

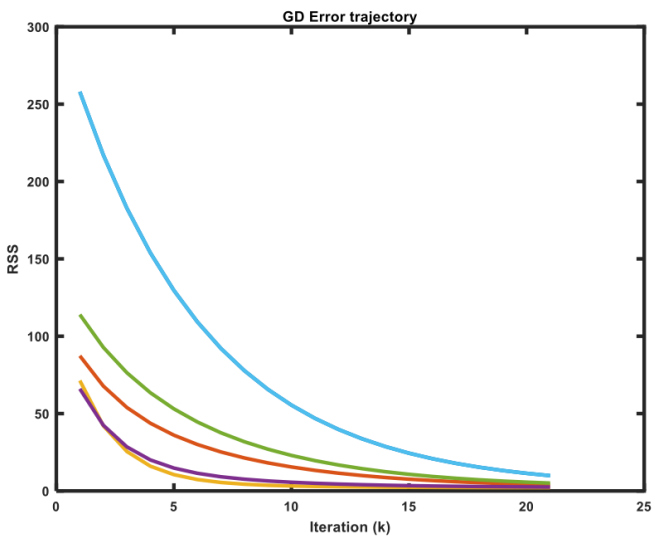
For this example, $\boldsymbol{\theta} \in \mathbb{R}^2$, $\text{eig}(\boldsymbol{H}) = \{\frac{1}{4}, \frac{1}{12}\}$. Step size $\alpha = 1$.

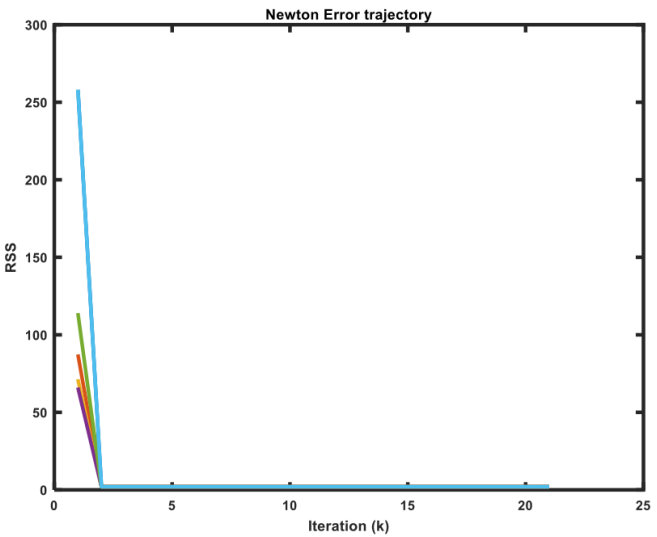
GD update:

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

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, but it is an upper bound.

- 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

- Fact. Due to upper bound, this approach ensures descent:

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

- 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 [3, §9.5].

CQ: Convergence of iterates

© 2026 Fessler

Q08.3 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 rate of Newton's method

© 2026 Fessler

If $J(\boldsymbol{\theta})$ is:

- twice differentiable
- a **strongly convex** function, meaning that $\nabla^2 J(\boldsymbol{\theta}) \succeq \mu \boldsymbol{I}$ for all $\boldsymbol{\theta}$ for some $\mu > 0$, and
- has a Lipschitz Hessian,

then provided that $\boldsymbol{\theta}_0$ is close enough to $\boldsymbol{\theta}_*$, the sequence $\{\boldsymbol{\theta}_t\}$ generated by Newton's method will converge to the (necessarily unique) minimizer $\boldsymbol{\theta}_*$ **quadratically fast** *i.e.*,

$$\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_*\| \leq \frac{1}{2} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_*\|^2 .$$

This convergence is the appeal of Newton's method.
Its drawback is inverting the Hessian, but **quasi-Newton methods** like LBFGS help address this with practical Hessian approximations.

"Convergence" of algorithms

© 2026 Fessler

- An algorithm's convergence is called **global** iff the sequence generated by the algorithm converges (to some limit) from
- Convergence is **local** if convergence (to some limit) is achieved only with an initialization in some given region called the
- 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 $J(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\theta}^\top \boldsymbol{b} + c, \quad \boldsymbol{H} \succ \boldsymbol{0},$

GD has **global linear convergence** iff

Convergence condition $\gamma < 1$ requires suitable step size

$$0 < \alpha <$$

where $\sigma_1 = \|\boldsymbol{H}\|_2$ is first **singular value** of $\boldsymbol{H}$.

After K iterations of GD, the initial "error" $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}_*\| \triangleq \text{err}_0$ reduces by at least γ^K:

$$\text{err}_K \triangleq \|\boldsymbol{\theta}_K - \boldsymbol{\theta}_*\| \leq$$

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 of J is **Lipschitz continuous**:

$$\|\nabla J(\boldsymbol{\theta}) - \nabla J(\boldsymbol{\gamma})\| \le L\|\boldsymbol{\theta} - \boldsymbol{\gamma}\|,$$

then using

$$\alpha =$$

guarantees (cost function) progress by GD of [4]:

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

More generally, if $0 < \alpha < 2/L$ then we have this (2025!) **tight upper bound** [5] on the cost function descent:

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

Effect of step size

© 2026 Fessler

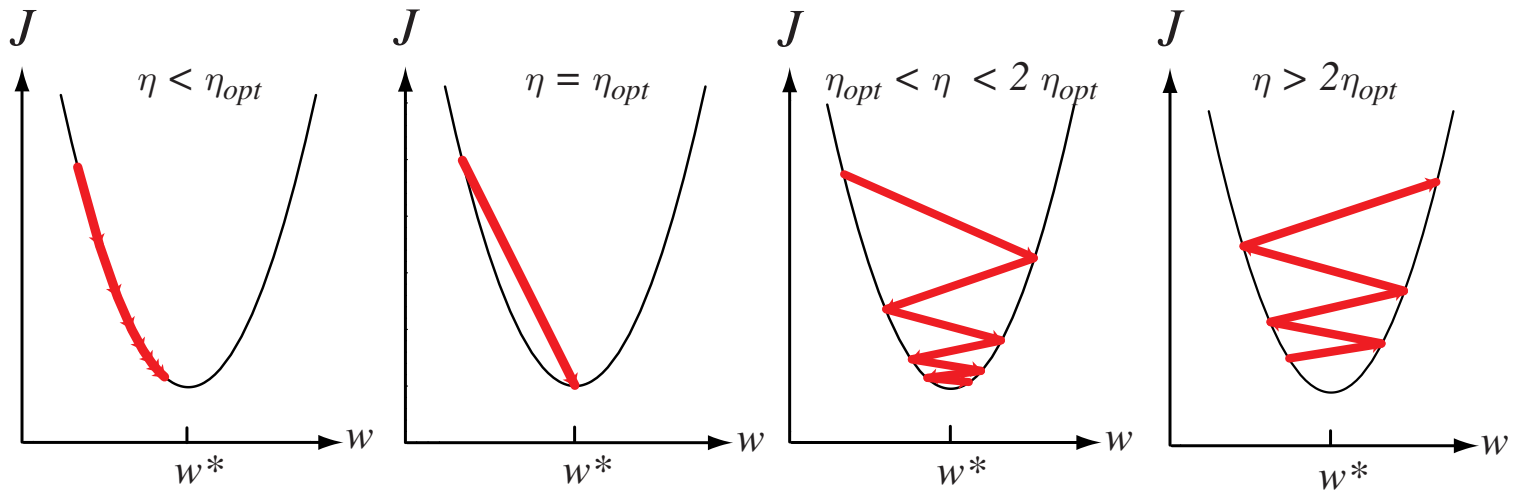


FIGURE 6.16. Gradient descent in a one-dimensional quadratic criterion with different learning rates. If $\eta < \eta_{opt}$, convergence is assured, but training can be needlessly slow. If $\eta = \eta_{opt}$, a single learning step suffices to find the error minimum. If $\eta_{opt} < \eta < 2\eta_{opt}$, 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 [6].

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**:

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

where here

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \alpha \left[\nabla^2 J(\boldsymbol{\theta}_t)\right]^{-1} \nabla J(\boldsymbol{\theta}_t) \\ \implies \mathcal{M}(\boldsymbol{\theta}) &= \boldsymbol{\theta} - \alpha \left[\nabla^2 J(\boldsymbol{\theta})\right]^{-1} \nabla J(\boldsymbol{\theta}). \end{aligned}$$

- 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:

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

  where $J_i(\boldsymbol{\theta})$ depends on the training data only through
- Or, we may also divide the data up in different ways, *i.e.*, by blocks of variables, called a **batch** or **mini-batch**, where **batch size** is often limited by GPU memory.
- **Stochastic gradient descent** (**SGD**) attempts to minimize the overall cost $J(\boldsymbol{\theta})$ by incrementally descending each J_i in the sum. Also called **incremental gradient descent**; "stochastic" is technically reserved for a setting where the subfunction J_i is selected probabilistically.
- 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].
The point of SGD is that the gradient step should be stochastic and be equal to ∇J in expectation.
For ridge regression we now have $O(d)$ per iteration, but each iteration uses only one data point. $O(nd)$ still to pass over the data once.

Gradient descent v Stochastic gradient descent © 2026 Fessler

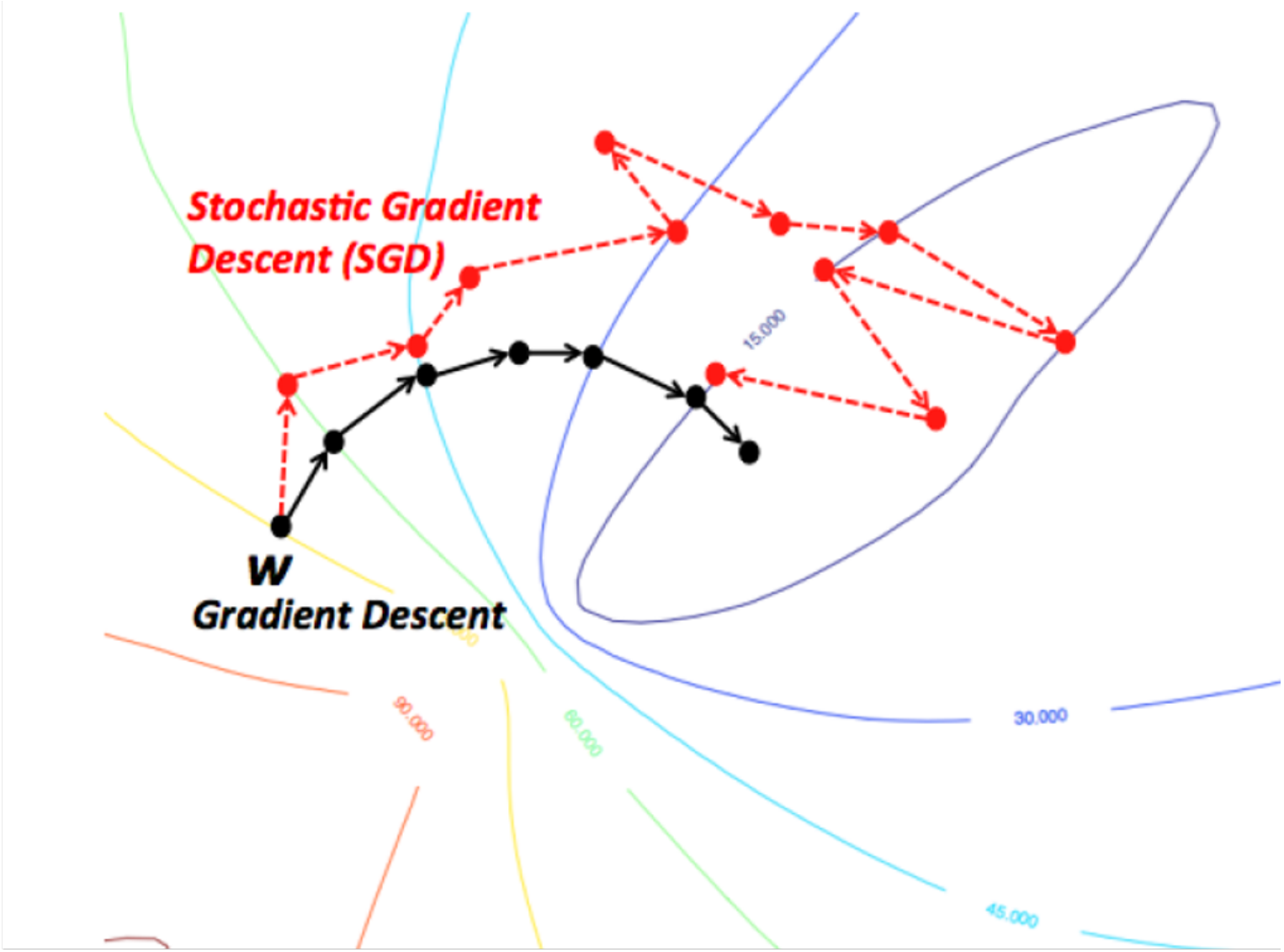


Image courtesy [12].

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 [15]

- 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.

Subgradient method

© 2026 Fessler

- 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
- [16]

Logistic regression: Between classification and regression ©

2026 Fessler

We first consider **logistic regression** in the binary case: $Y \in \{0, 1\}$. (There is a **multiclass** extension.)

This setting is typical for:

- Medical data:
  - The patient had a heart attack or did not.
  - The patient is alive or dead.
- Educational data:
  - The student passed or failed
  - The student was accepted to the program or not.
- Financial consumer data:
  - The transaction is legitimate or fraudulent.
  - The consumer defaulted on their loan or not.

Logistic regression: Classification or regression? © 2026 Fessler

- Consider the binary case for logistic regression: $Y \in \{0, 1\}$.
- Logistic regression *directly* builds and uses a model (or estimate) $\hat{\eta}(\boldsymbol{x}; \hat{\boldsymbol{\theta}})$ for the **posterior** probability (hence a kind of regression):

$$\eta(\boldsymbol{x}) \triangleq$$

- This estimate leads naturally to a **plug-in classifier** [1, §10.2.1]:

$$f(\boldsymbol{x}) \triangleq \begin{cases} 1, & \text{if} \\ 0, & \text{otherwise.} \end{cases}$$

- It combines two ideas:
  - Logistic probability estimation (a kind of **regression**)
  - "Linear" **classification**

Log. reg.: Probability model example

© 2026 Fessler

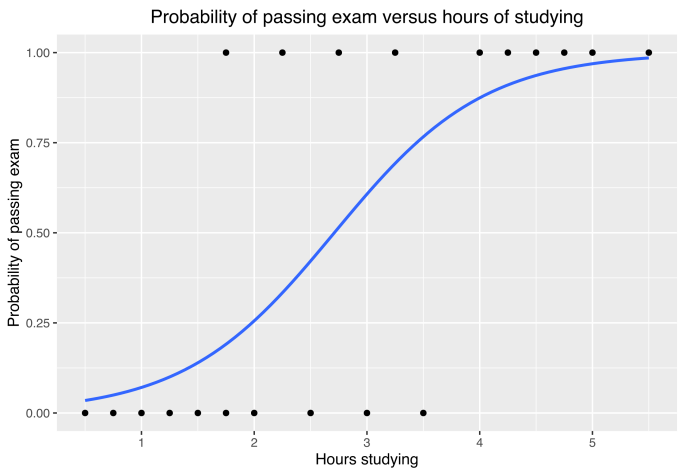
- Example taken from Wikipedia:
  A group of 20 students spends between 0 and 6 hours studying for an exam. How does the number of hours spent studying affect the probability of a student passing the exam?

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

If a (different) student studies for 3 hours, what is the probability of a pass?

Possible posterior probability model:

$$\begin{aligned} \mathrm{P}\{\text{Pass} \mid \text{Hours}\} &= \frac{1}{1 + \exp\Big(-(\qquad)\Big)} \\ &= \frac{1}{1 + \exp(-(1.5 \times \text{Hours} - 4))} \\ &= \frac{1}{1 + \mathrm{e}^{-(1.5 \times (\text{Hours} - 2.7))}} \end{aligned}$$



Plot courtesy Canley

- The blue curve is a **sigmoid function** or "S-shaped" curve.
- Here we are using the **standard logistic function**:

$$\sigma(x) \triangleq \frac{1}{1 + \mathrm{e}^{-x}}.$$

- Feed linear combination of features into **logistic** probability model:

$$\eta(\boldsymbol{x};\boldsymbol{\theta}) \triangleq \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b) = \frac{1}{1 + \mathrm{e}^{-(\boldsymbol{w}^\top \boldsymbol{x} + b)}} \approx$$

  where

$$\boldsymbol{\theta} = \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix} \in \mathbb{R}^{d+1}.$$

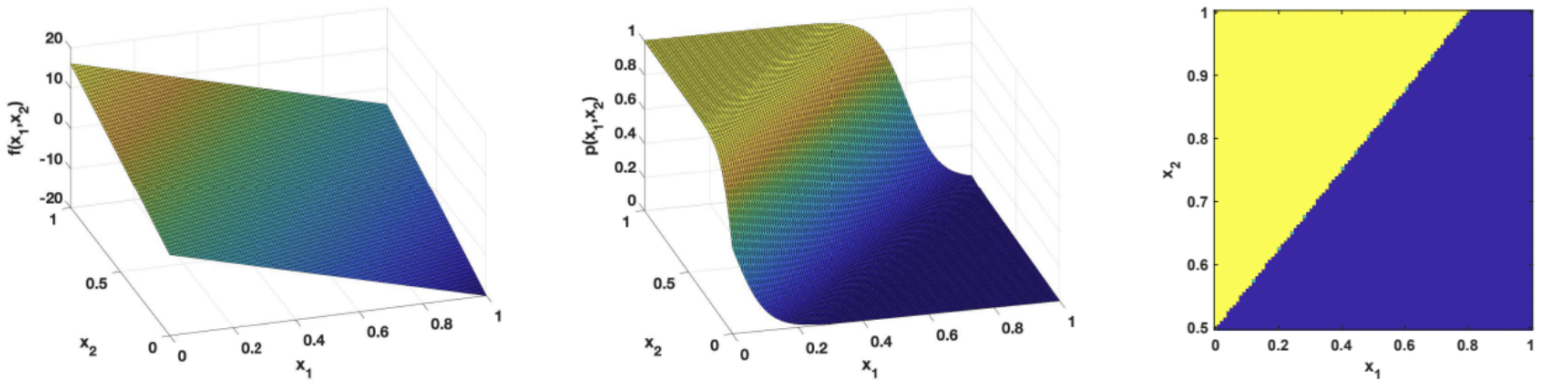
- Somehow estimate $\hat{\boldsymbol{\theta}} = \begin{bmatrix} \hat{b} \\ \widehat{\boldsymbol{w}} \end{bmatrix}$. (Forthcoming...)
- Then plug-in method is a "linear" classifier:

$$\hat{y} = f(\boldsymbol{x}) = \mathbb{I}_{\{\eta(\boldsymbol{x};\hat{\boldsymbol{\theta}}) \geq 1/2\}}, \quad \eta(\boldsymbol{x};\hat{\boldsymbol{\theta}}) \geq \frac{1}{2} \iff \widehat{\boldsymbol{w}}^\top \boldsymbol{x} + \hat{b} \geq 0.$$

- f is linear:

$$f(\boldsymbol{x}) = -20x_1 + 16x_2$$

$f(\boldsymbol{x})$ | $p(\boldsymbol{x}) = \sigma(f(\boldsymbol{x}))$ | classification regions



Above, "f" is our $\tilde{f}(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$
and $\sigma(\cdot)$ is the **standard logistic function**.

Figure courtesy Justin Romberg

- Given training data $\mathcal{D} = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n))$, we must estimate the model parameter vector

$$\boldsymbol{\theta} = \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix}.$$

- We need a criterion that quantifies how well

$$\eta(\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{1 + \mathrm{e}^{-(\boldsymbol{w}^\top \boldsymbol{x} + b)}}$$

explains the training data for any candidate $\boldsymbol{\theta}$.

- Let $p(y \mid \boldsymbol{x}; \boldsymbol{\theta})$ denote the conditional probability that $Y = y$ given $\boldsymbol{X} = \boldsymbol{x}$.
- By definition (and then writing without braces):

$$p(y \mid \boldsymbol{x}; \boldsymbol{\theta}) = \begin{cases} \eta(\boldsymbol{x}; \boldsymbol{\theta}), & y = 1 \\ 1 - \eta(\boldsymbol{x}; \boldsymbol{\theta}), & y = 0 \end{cases}$$

$$= \quad$$

- The **likelihood** of $\boldsymbol{\theta}$ is

$$L(\boldsymbol{\theta}) \triangleq \mathrm{p}(\mathcal{D}_{\text{train}}; \boldsymbol{\theta}) = \underbrace{\prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})}_{\text{independence}}$$

$$= \prod_{i=1}^{n}$$

- Notation (and rationale for the term "linear"):

$$\bar{\boldsymbol{x}}_i = \begin{bmatrix} 1 \\ \boldsymbol{x}_i \end{bmatrix} = [1 \; x_{i1} \; \cdots \; x_{id}]^\top$$

$$\boldsymbol{\theta} = \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix} = [b \; w_1 \; \cdots \; w_d]^\top$$

$$\implies \boldsymbol{w}^\top \boldsymbol{x}_i + b = \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i.$$

- The **log-likelihood** of $\boldsymbol{\theta}$ is

$$\begin{aligned} \ell(\boldsymbol{\theta}) &\triangleq \log L(\boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} \left[y_i \log\left(\frac{1}{1 + \mathrm{e}^{-\boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i}} \right) + (1 - y_i) \log\left(\frac{\mathrm{e}^{-\boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i}}{1 + e^{-\boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_i}} \right) \right] \\ &= \sum_{i=1}^{n} \left[y_i \log\Big(\sigma(\tilde{f}(\boldsymbol{x}_i; \boldsymbol{\theta})) \Big) + (1 - y_i) \log\Big(1 - \sigma(\tilde{f}(\boldsymbol{x}_i; \boldsymbol{\theta})) \Big) \right]. \end{aligned}$$

- If the parameter dimension d is small, especially relative to the number of samples n, we can compute the estimate $\hat{\boldsymbol{\theta}}$ by maximizing the log-likelihood (*i.e.*, minimizing the negative log-likelihood):

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^{d+1}}{\arg\min} -\frac{1}{n} \ell(\boldsymbol{\theta}).$$

- Dividing by n aligns with ERM formulation.
- This is a **convex** optimization problem (HW), but there is no closed-form solution, so iterative unconstrained optimization methods are required.

CQ: Logistic

© 2026 Fessler

Q08.4 Logistic regression is:

A: **generative** B: **discriminative** C: neither D: both

Q08.5 Logistic regression is:

A: **parametric** B: **nonparametric** C: neither D: both

- Unless $n \gg d$, it is typically preferable to minimize a **regularized** cost function like:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^{d+1}}{\arg\min}\, J(\boldsymbol{\theta}), \quad J(\boldsymbol{\theta}) \triangleq \underbrace{-\frac{1}{n}\ell(\boldsymbol{\theta})}_{\text{empirical risk}} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2, \quad \boldsymbol{\theta} = \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix},$$

  where $\lambda > 0$ is a used-specified constant called a **regularization parameter**.

- Why introduce the regularization term?
  - If $n < d$, the Hessian $\nabla^2 \ell(\boldsymbol{\theta})$ will be singular.
  - $J(\boldsymbol{\theta})$ is **strictly convex** for $\lambda > 0$, so it has a unique minimizer. (HW)
  - The regularization term encourages the solution to have small norm, which can prevent **overfitting** to the training data - important when the sample size n is small relative to d.

JULIA demo

- No ellipses...
- Optimization via Quasi-Newton
- Decision boundary is

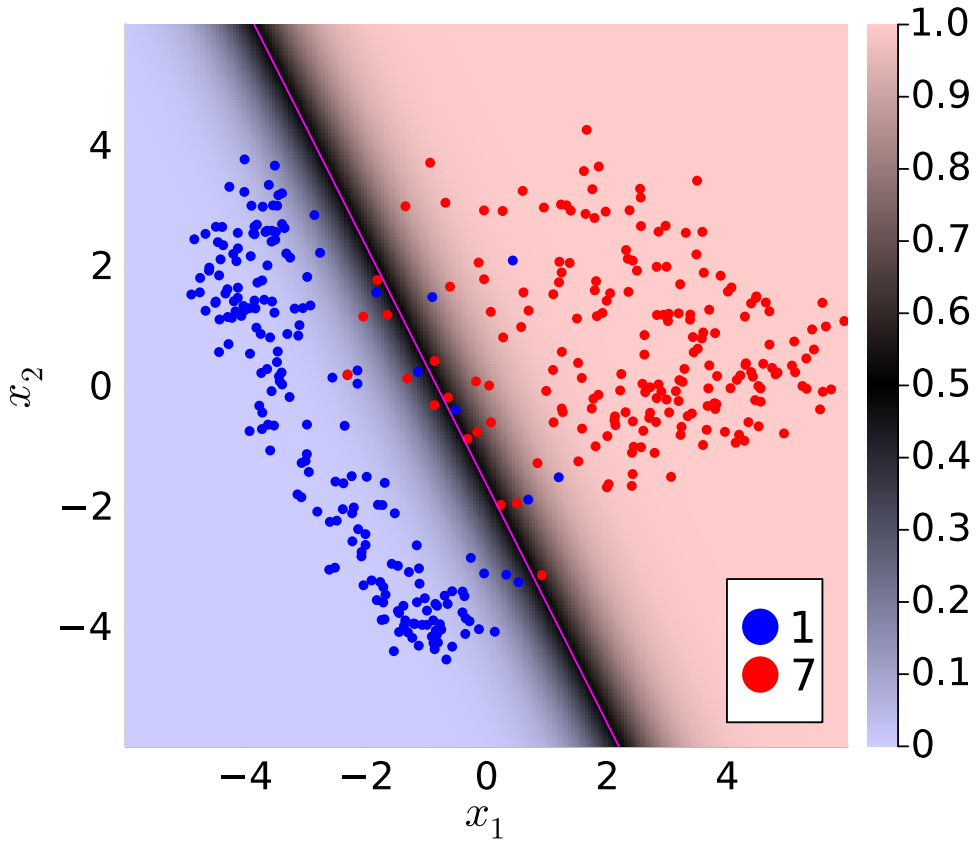
$$\{\boldsymbol{x} : \boldsymbol{w}^\top \boldsymbol{x} + b = 0\}$$

so

$$x_2 = \frac{-(b + w_1 x_1)}{w_2}.$$

Here

$$\hat{\boldsymbol{\theta}} = [1.76\ 2.12\ 1.07]^\top.$$



Color: predicted posterior prob's: $\hat{\eta}(\boldsymbol{x})$.
This is the "regression" in the name.

- When d is large, it can be more **interpretable** to select a subset of the features, instead of using all d features.
- Ideally we might solve a **feature selection** problem such as

$$\underset{\boldsymbol{\theta} \in \mathbb{R}^{d+1}}{\arg\min} -\ell(\boldsymbol{\theta}) \text{ s.t. } \|\boldsymbol{\theta}\|_0 \leq K$$

  for some user-selected (or cross-validation selected) $K \in \mathbb{N}$, where $\|\boldsymbol{x}\|_0 = \sum_i \mathbb{I}_{\{x_i \neq 0\}}$.
- $\|\cdot\|_0$ is discontinuous and nonconvex, making optimization very difficult
- A practical alternative is to use a 1-norm regularizer:

$$J(\boldsymbol{\theta}) = -\frac{1}{n}\ell(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1 .$$

  This cost function is **convex**, though it is non-smooth, which motivates advanced optimization approaches like accelerated **proximal gradient** methods [17].

Logistic regression: Summary

© 2026 Fessler

- More than a classifier – it predicts the probability of each class
- LR assumption is less restrictive than LDA assumption (but if data are Gaussian, LDA is more powerful)
- Widely used in application domains including health sciences
- Example: predict whether a patient will develop a disease (*e.g.*, diabetes, coronary disease) based on various attributes (age, sex, weight, BMI, etc.)
- For imbalanced classes, may want to use weighted ERM.
- Multiclass extension: may discuss later (or in HW)
- Nonlinear extension:

$$\eta(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{1 + \exp(-h(\boldsymbol{x}))}$$

where h is nonlinear.
We will see examples later (kernel methods, neural networks)

CQ: SGD

© 2026 Fessler

For $J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} J_i(\boldsymbol{\theta})$, the SGD algorithm is as follows:

- Initialize $\boldsymbol{\theta}_0$; set $t \leftarrow 0$
- For $j = 1, \ldots,$ `max_iter`
  For $i = 1, \ldots, n$ *in random order*
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} -$ _____
  $t \leftarrow t + 1$
  End
  If convergence condition satisfied, exit
  End

Q08.6 Fill in the blank:

A: $\alpha \nabla J(\boldsymbol{\theta}_{t-1})$ B: $\frac{1}{n} \nabla J_i(\boldsymbol{\theta}_{t-1})$ C: $\alpha_j \nabla J_k(\boldsymbol{\theta}_t)$

D: $\alpha_j \nabla J_i(\boldsymbol{\theta}_{t-1})$ E: none

References I

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

[2] J. D. Deng.
Machine learning with Julia: An algorithmic exploration.
Springer, Singapore, 2026.

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

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

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

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

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

[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.

References II

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

[16] 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.

References III

[17] D. Kim and J. A. Fessler.
Adaptive restart of the optimized gradient method for convex optimization.
J. Optim. Theory Appl., 178(1):240–63, July 2018.