EECS 553 Lecture 01
Machine Learning (ECE)
[class version]

Jeff Fessler

University of Michigan
©Laura Balzano and Jeff Fessler, UM, August 31, 2026, 16:57

F26

Outline

© 2026 Fessler

- Logistics
  - Instructors / office hours
  - Links
  - Exams and grade weighting
  - AI policy
  - Piazza
  - Pre-requisites
  - Homework
  - Book(s)
- Introduction to ML
- Classification introduction
- k-NN classifier
- Probabilistic methods
  - Loss and risk
  - 0-1 loss and error probability
- Next

Instructors

© 2026 Fessler



Jeff Fessler,
Professor in EECS

Research on medical imaging, optimization, ML for image reconstruction, ...



Jiayao Yang
ECE PhD student

Research on MRI
ML for MRI...



Hongze Yu
ECE PhD student

Research on MRI
ML for MRI...

Office hours and locations posted on Canvas (Mon-Fri)

ML Example: Low-dose X-ray CT imaging

© 2026 Fessler

From Song & Shen et al., ICLR 2022 [0], using a score-based diffusion prior $\nabla \log p(\boldsymbol{x})$
Trained with 47K 2D CT images. Recon 23 projection views ($\approx$ 17-fold dose reduction)

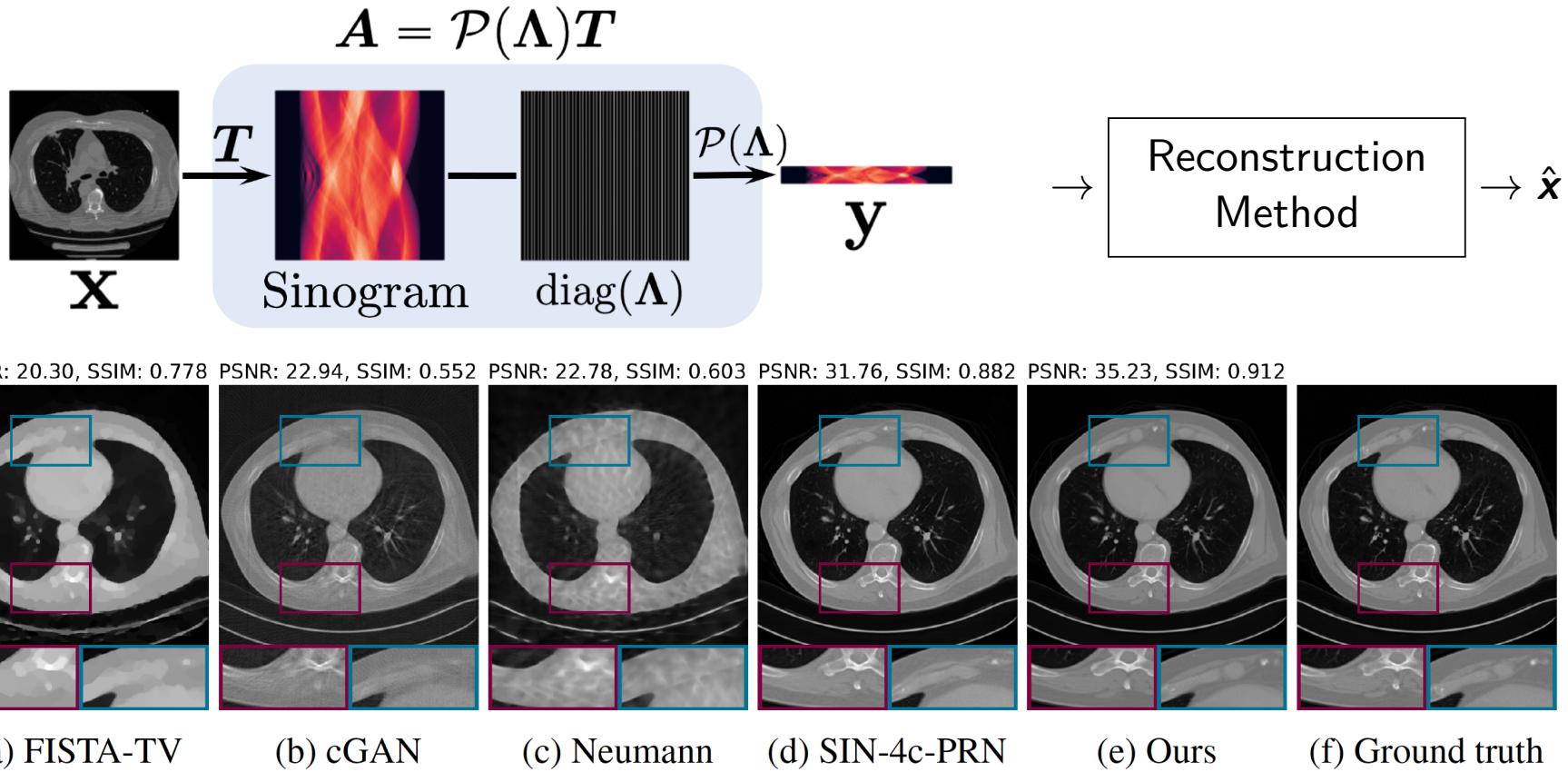


Figure from [2]

Logistics: links to sites

© 2026 Fessler

- All of these links are on Canvas:
  - Canvas is main hub / resource
  - Google Drive for template / annotated course slides (pdf)
  - Prof. Scott's excellent ML notes on Canvas/ Files / Scott-ML-Notes-W23
  - Gradescope for HW assignments / submission
  - Google Colab for demos (in Python and/or Julia).
  - Julia ML demos
  - gdrive folder of F26 colab demos
  - Great Lakes HPC cluster (compute resource for projects)
  - Overleaf for student-generated practice problems
  - iClicker for in-class active learning (check for email instructions)
  - Back-up material at `https://web.eecs.umich.edu/~fessler/course/553/index.html`
  - Evolving lecture-by-lecture topics
  - Syllabus link on Canvas
- A few syllabus highlights next

- Ask questions!
- iClicker https://iclicker.com for in-class active learning
  - clicker points start counting *next* class
  - submitting iclicker answers when absent is a HC violation
  - using AI for clicker questions is a HC violation
  - discussing with fellow students for clicker questions is expected
- Speaking of peer interaction:
  These slides are based on LaTeX source files provided by Prof. Laura Balzano, who in turn drew inspiration from PPT slides by Prof. Clay Scott and Prof. Samey Oymak. I gratefully acknowledge all of their generosity.

  Any errors are my responsibility. (Use Piazza to discuss!)

Grade weighting

© 2026 Fessler

25%	Exam1 **(Mon Oct 12 6-8 PM)**
25%	Exam2 **(Mon Nov 16 6-8 PM)**
10%	Best of your (possibly standardized) two exam scores
5%	HW (key preparation for exams)
	1 lowest dropped by Canvas; not autograder problems
5%	Clicker questions
	4 lowest days dropped
30%	Project on ML reproducibility
	(detailed score weighting on syllabus)

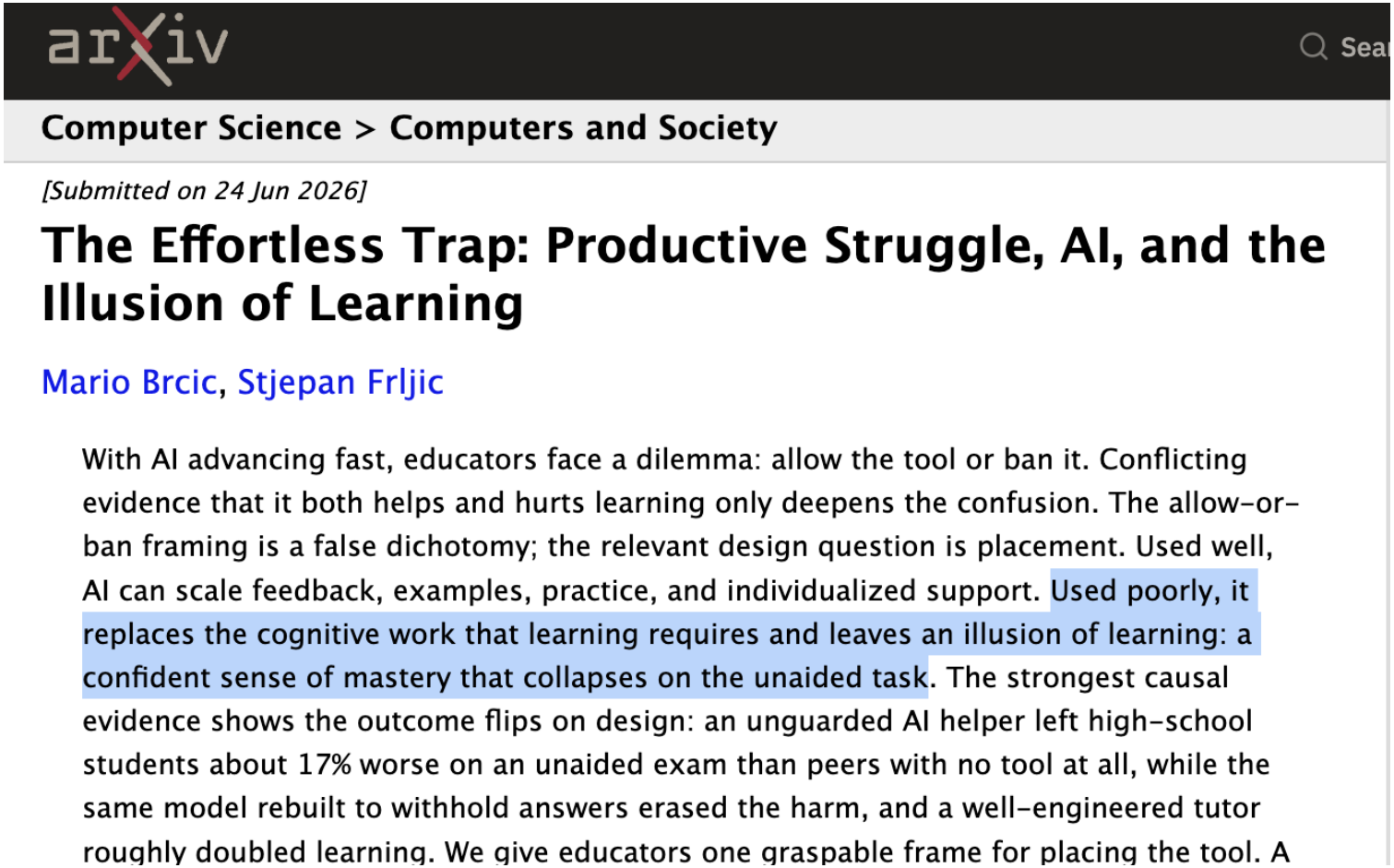
See syllabus for exam policies.

Project deadlines subject to change.
Dec. 17, 4-6PM: Possible oral or poster presentations during official "final exam" time

AI policy guiding principle

© 2026 Fessler

- Details on syllabus - read it!
- Over-arching principle:
  You can (and should) use GenAI in ways that enhance your learning, but you must not use GenAI in any way that hinders your learning in this class.
- I have tried to design the course policies to



arXiv Sea

Computer Science > Computers and Society

[Submitted on 24 Jun 2026]

The Effortless Trap: Productive Struggle, AI, and the Illusion of Learning

Mario Brcic, Stjepan Frljic

With AI advancing fast, educators face a dilemma: allow the tool or ban it. Conflicting evidence that it both helps and hurts learning only deepens the confusion. The allow-or-ban framing is a false dichotomy; the relevant design question is placement. Used well, AI can scale feedback, examples, practice, and individualized support. Used poorly, it replaces the cognitive work that learning requires and leaves an illusion of learning: a confident sense of mastery that collapses on the unaided task. The strongest causal evidence shows the outcome flips on design: an unguarded AI helper left high-school students about 17% worse on an unaided exam than peers with no tool at all, while the same model rebuilt to withhold answers erased the harm, and a well-engineered tutor roughly doubled learning. We give educators one graspable frame for placing the tool. A

Used poorly, it [AI] replaces the cognitive work that learning requires and leaves an illusion of learning: a confident sense of mastery that collapses on the unaided task. [3]

Piazza

© 2026 Fessler

- Use Piazza to ask questions about the course
  - Students can answer other students questions
  - Instructors will also answer questions and "endorse" student questions and answers
- Post anonymously (to other students) if you prefer
- Send "private" messages visible only to the instructors (for issues that are not relevant to other students)
- Please use Piazza rather than emailing the instructors (unless we request email); it helps us keep organized.
- Use Piazza to form small groups, find study mates, project partners – see designated pinned thread for this purpose ("Search for Teammates!")

CQ: Pre-requisites: Math

© 2026 Fessler

Q01.1 What courses have you taken in linear algebra and probability?

A: ECE 501 and ECE 551

B: Related *graduate* level classes, at least one was from another UM dept.

C: I took at least one of these graduate level classes at another university.

D: Uh oh, I took at least one of these courses at the undergraduate level only.

E: none of these

(Points for clicker questions start in the next class. Today is just practice.)

CQ: Pre-requisites: Coding

© 2026 Fessler

Q01.2 What is your software background?

A: Primarily adept at Python

B: Primarily adept at Julia

C: Adept at both Julia and Python

D: Uh oh, I pretty much only use Matlab

E: none of these

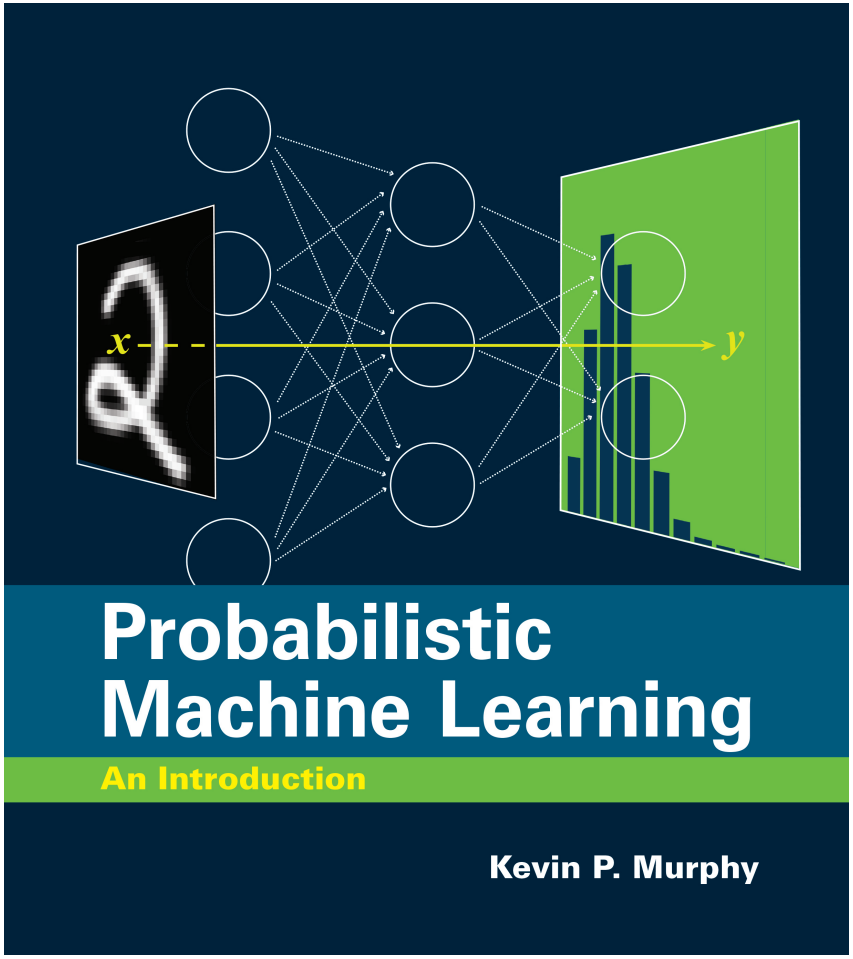
Homework

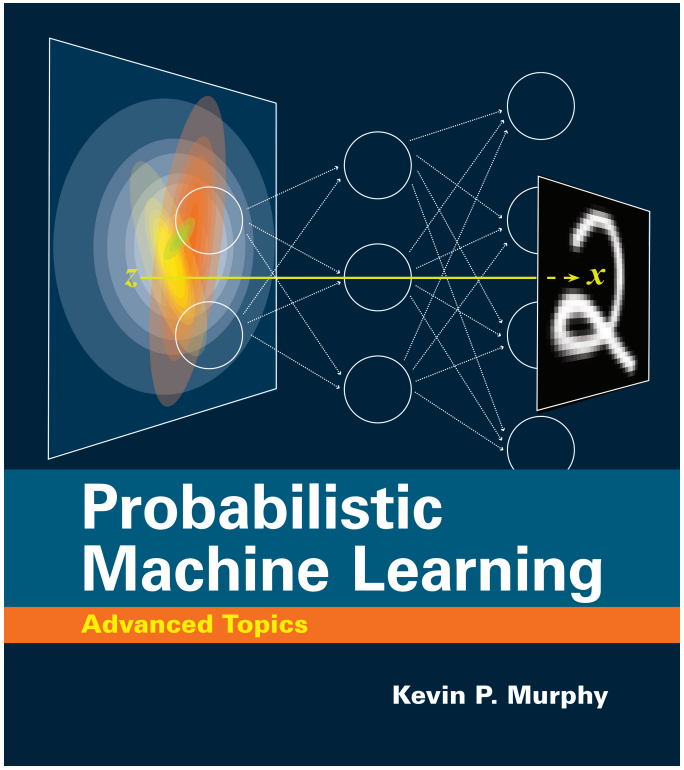
© 2026 Fessler

- HW 1 is released
  - Due Wed. Sep. 8, 11PM.
  - 24 grace period (no penalty) for upload to gradescope
  - It covers linear algebra (551) and probability (501) topics that are pre-requisite for this course.
- If you struggle with this homework and have not had the advisory pre-requisite courses, I suggest you drop 553 and take those pre-reqs first (ECE 501/551 offered now) or take a 400-level ML course instead (EECS 453 offered now).

Recommended book

© 2026 Fessler



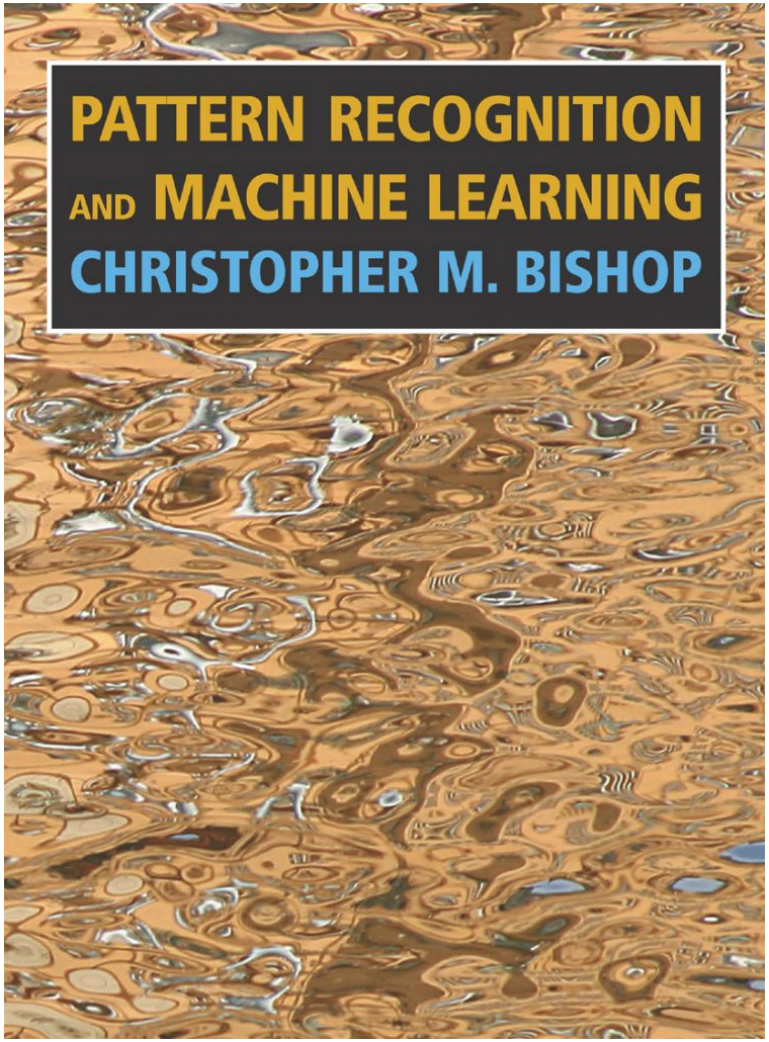


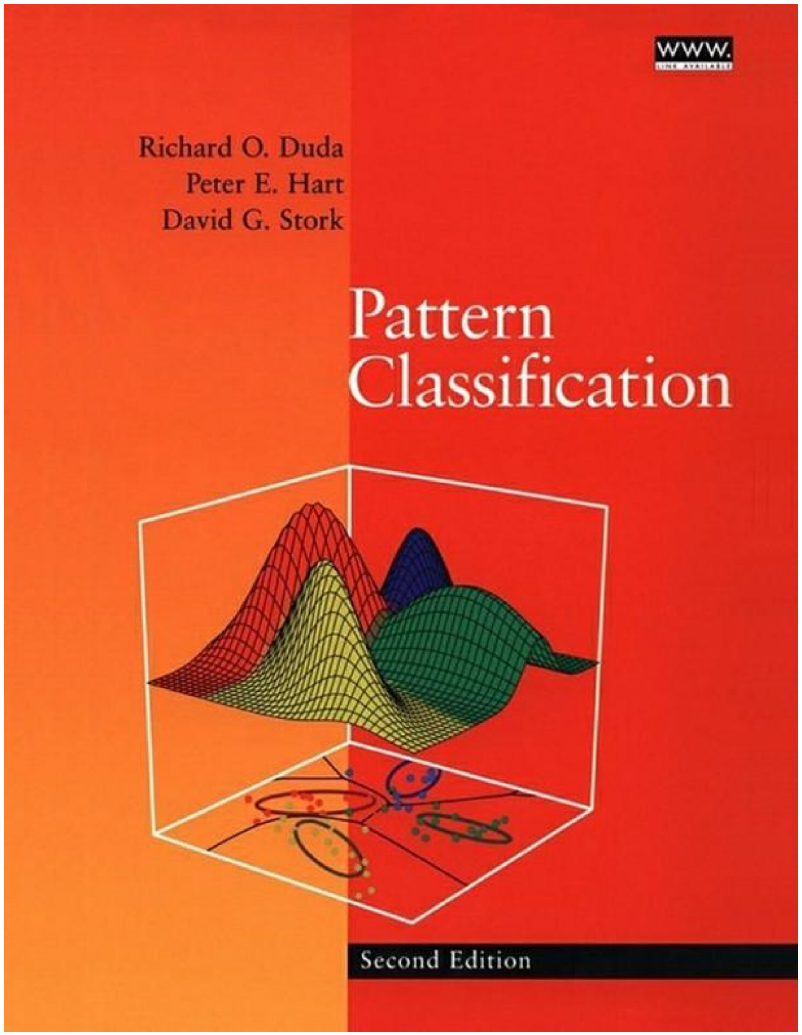
[1] https://probml.github.io/pml-book/book1.html

[4] https://probml.github.io/pml-book/book2.html

More books for reference

© 2026 Fessler



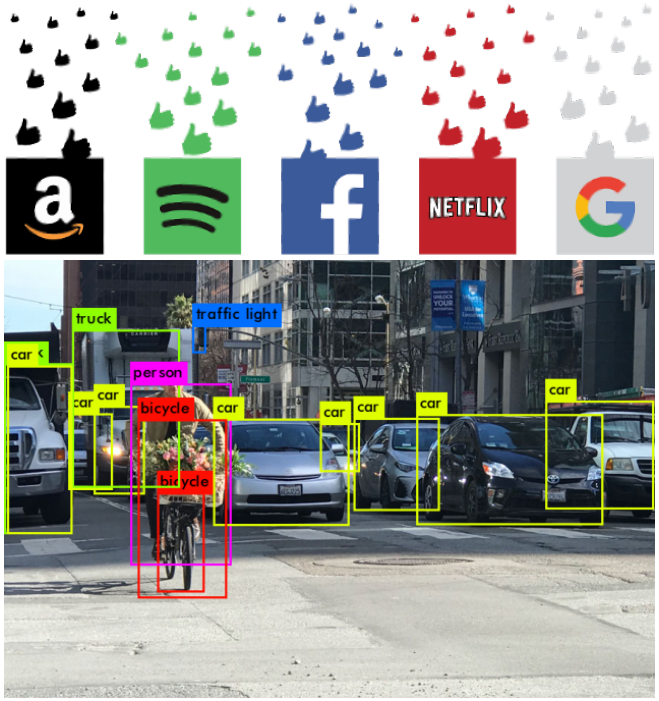


[5] https://www.microsoft.com/en-us/research/people/cmbishop/prml-book

[6] https://ebookcentral-proquest-com.proxy.lib.umich.edu/lib/umichigan/detail.action?docID=699526

ML is about "learning from data" or "learning from experience"

- Recommender systems (Netflix, Amazon)
- Collision avoidance systems
- Spam filtering
- Text-to-speech/Speech-to-text
- Machine translation (Google translate)
- Image filters / word identification in images
- Face detection/biometrics
- Chatbots (ChatGPT)
- Search engine refinement
- AlphaGo
- Personalized medicine
- Forecasting electricity demand
- Weather forecasting
- Credit risk assessment
- Financial market prediction
- Fraud detection
- Disaster Recovery guidance
- Route planning
- ...



Images courtesy:

https://medium.com/@igorirailean/a-recommender-system-using-fastai-in-google-colab-110d363d422f

https://centralcoastdatascience.org/projects/all/2021/how-pixel-differences-can-affect-sensors-self-driving-cars

Modern challenges in ML

- Scalability
  - Large sample size, high dimension
  - Self-supervision (scalable in labels)
    e.g., masked autoencoders, next word prediction
  - High velocity streaming data
- Applicability and accuracy
  - Identifying appropriate error metrics and objective functions
  - Improving error rates
- Social responsibility
  - Fairness, privacy, safety, sustainability, transparency, reproducibility, ...
- Low-quality data
  - Changing/drifting distributions, feature noise, label noise, missing data, ...
- Non-Euclidean data
  - Graphs, strings, functions, rank orderings, categorical, probability distributions, ...

This course

© 2026 Fessler

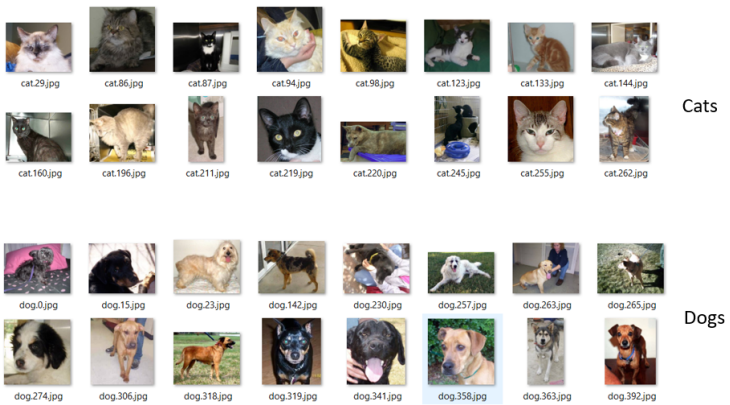
- Supervised, self-supervised, and unsupervised learning
- Deriving algorithms from basic principles
- Discussing general phenomena that pervade ML (modeling, generalization, overfitting, regularization, convergence, etc.)
- Implementing core algorithms on homework
- Brief exposure to modern challenges and applications
- Self-directed exploration through the ML reproducibility project
- Goal: A foundation for students who wish to read the ML literature, to do research in ML, and/or to apply ML ideas in their work or research.
- Further prepare for

Classification: a canonical ML problem

© 2026 Fessler

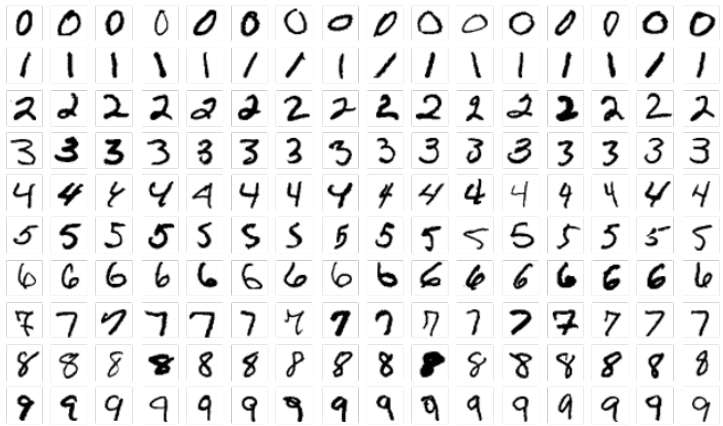
Supervised learning: training data with

Two class application:



https://github.com/melihaltun/cats_vs_dogs_recognition

Ten class application:



https://upload.wikimedia.org/wikipedia/commons/2/27/MnistExamples.png

- Goal: develop computer algorithms that take as inputs *similar* such data and return the labels correctly (with high probability)
- Key concept: **generalization** as opposed to **memorization**

Classification: problem formulation

© 2026 Fessler

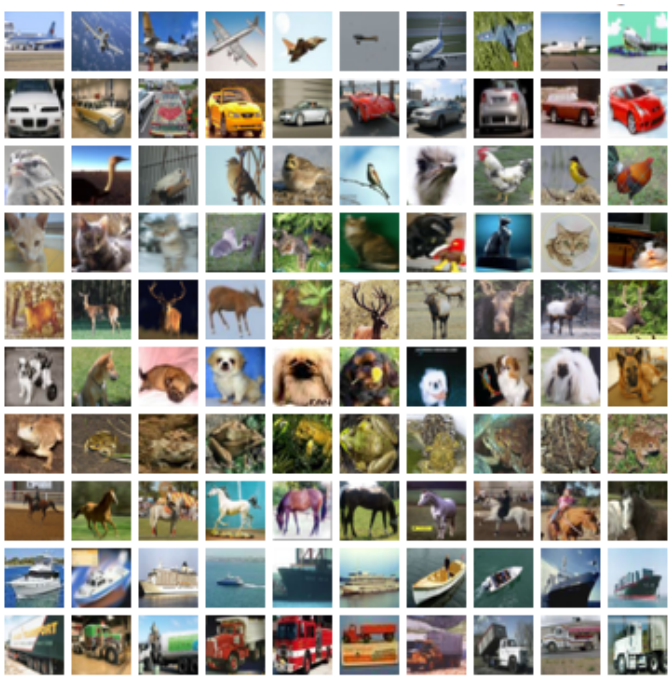
- Training data: pairs of **feature vectors** and **labels**:

$$\mathcal{D}_{\text{train}} = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)),$$

  where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{1, \ldots, K\}$ (or any finite set).
- d denotes the **ambient dimension** of each data point (vector).
- K denotes the number of **classes**, *e.g.*, $K = 10$ for digits 0:9
- $K = 2$ is **binary classification**, *e.g.*, cats vs dogs.
  Often, labels $\{+1, -1\}$ are used for binary classification, instead of $\{1, 2\}$. And sometimes labels
- A **classifier** is a **function** that maps *any* $\boldsymbol{x} \in \mathbb{R}^d$ to
  $f : \mathbb{R}^d \mapsto \mathcal{Y}$ having **domain**: $\mathbb{R}^d$, **co-domain** $\mathcal{Y} = \{1, \ldots, K\}$.
- A **trainer** or **learner** takes the training data $\mathcal{D}_{\text{train}}$ (and **hyperparameters**) as input and returns

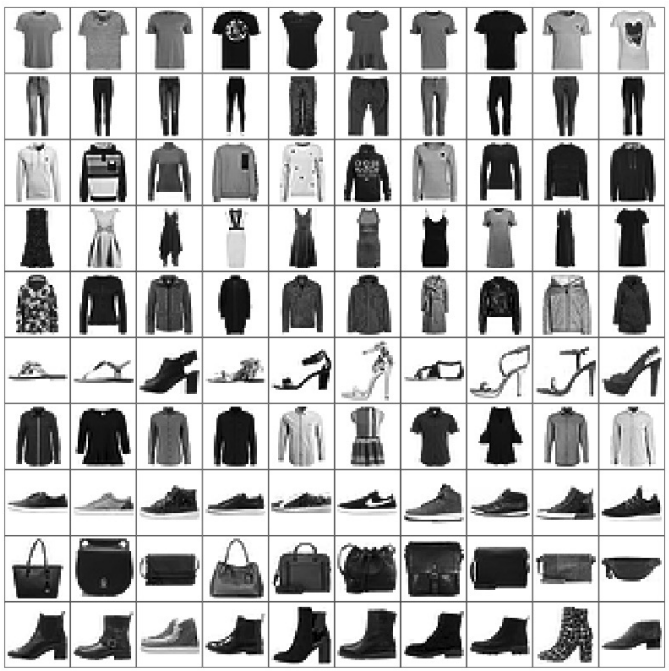
$K > 2$ is called **multiclass classification**.
More example problems with $K = 10$:

CIFAR-10 (2009) [7]



Airplanes, Cars, Birds, Cats, Deer, Dogs, Frogs, Horses, Ships, Trucks
60K 32×32 color images
https://www.kaggle.com/c/cifar-10

Fashion-MNIST (2017) [8]



https://www.christianhaller.me/blog/projectblog/2020-12-29-FashionMNIST

- The **nearest-neighbor classifier** is a classic **supervised learning** method.
- Requires labeled training data $\mathcal{D}_{\text{train}} = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n))$
- The classifier function $f : \mathbb{R}^d \mapsto \mathcal{Y}$ assigns any $\boldsymbol{x} \in \mathbb{R}^d$ the same label as the "closest" training instance $\boldsymbol{x}_i$.
- In math:

$$\hat{y} = f(\boldsymbol{x}) = y_{\hat{i}(\boldsymbol{x})}, \quad \hat{i}(\boldsymbol{x}) \triangleq$$

- aka 1-nearest-neighbor classifier

Q01.3 In 551 you learned that "all norms are equivalent." [9, §6.2]. The norm choice matters here?

A: True B: False **??**

The k-nearest-neighbors (k-NN) classifier: Idea © 2026 Fessler

- The ***k*-nearest-neighbor classifier** (**k-NN**) or (**KNN**) is a **supervised learning** method.
- It needs (almost!) no "learning" or "training" process
- For any input feature vector $\boldsymbol{x} \in \mathbb{R}^d$, the k-NN classifier assigns the *majority vote* label of the k closest training points $\{\boldsymbol{x}_i\}$
- k is odd, so no "tie" votes
- Caution:
  - "k" is number of
  - "K" is number of

  Determine which "k" is meant from context!
- Julia packages NearestNeighbors.jl & NearestNeighborModels.jl

The k-nearest-neighbors (k-NN) classifier: Math © 2026 Fessler

- Formula
  - Find subset $\mathcal{N}_k = \mathcal{N}_k(\boldsymbol{x}, \mathcal{D}_{\text{train}})$ of indices $\{1, \ldots, n\}$ having **cardinality** k corresponding to k nearest neighbors of $\boldsymbol{x}$, *i.e.*, such that

$$\max_{i \in \mathcal{N}_k} \|\boldsymbol{x} - \boldsymbol{x}_i\| \leq \min_{j \notin \mathcal{N}_k} \|\boldsymbol{x} - \boldsymbol{x}_j\| .$$

  - Majority vote of classes of those nearest neighbors:

$$\hat{y} = f(\boldsymbol{x}) = \underset{k' \in \{1,\ldots,K\}}{\arg\max} \sum_{i \in \mathcal{N}_k} \mathbb{I}_{\{y_i = k'\}}.$$

Q01.4 The set $\mathcal{N}_k$ is unique.

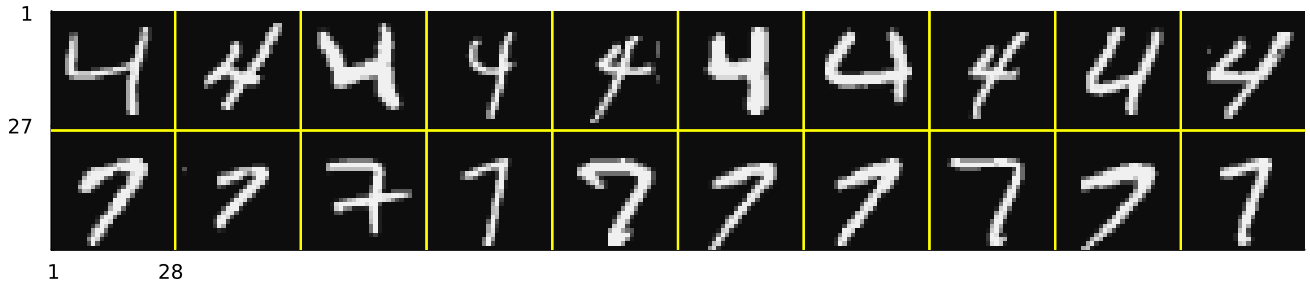
A: True B: False **??**

- What is the effect of k? (See demo)

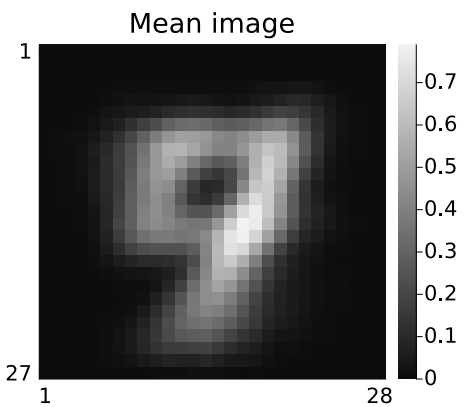
Demo: k-NN for binary classification - Prep

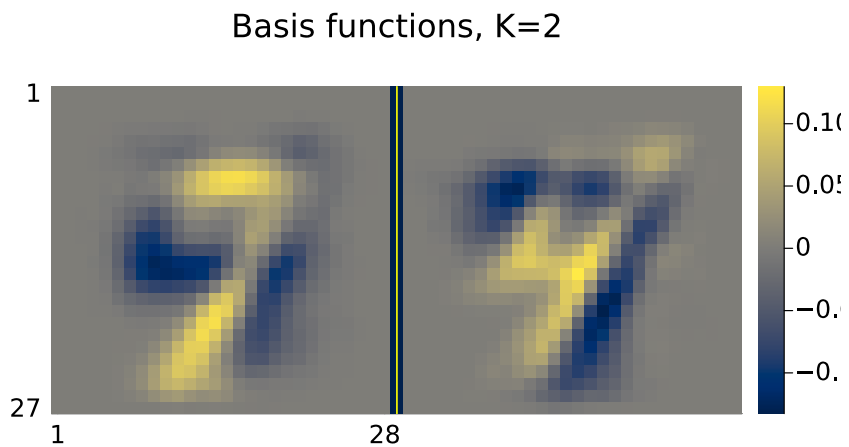
© 2026 Fessler

Julia demo with MNIST hand-written digit data for "4" and "7"
Data, d=756, N=2000



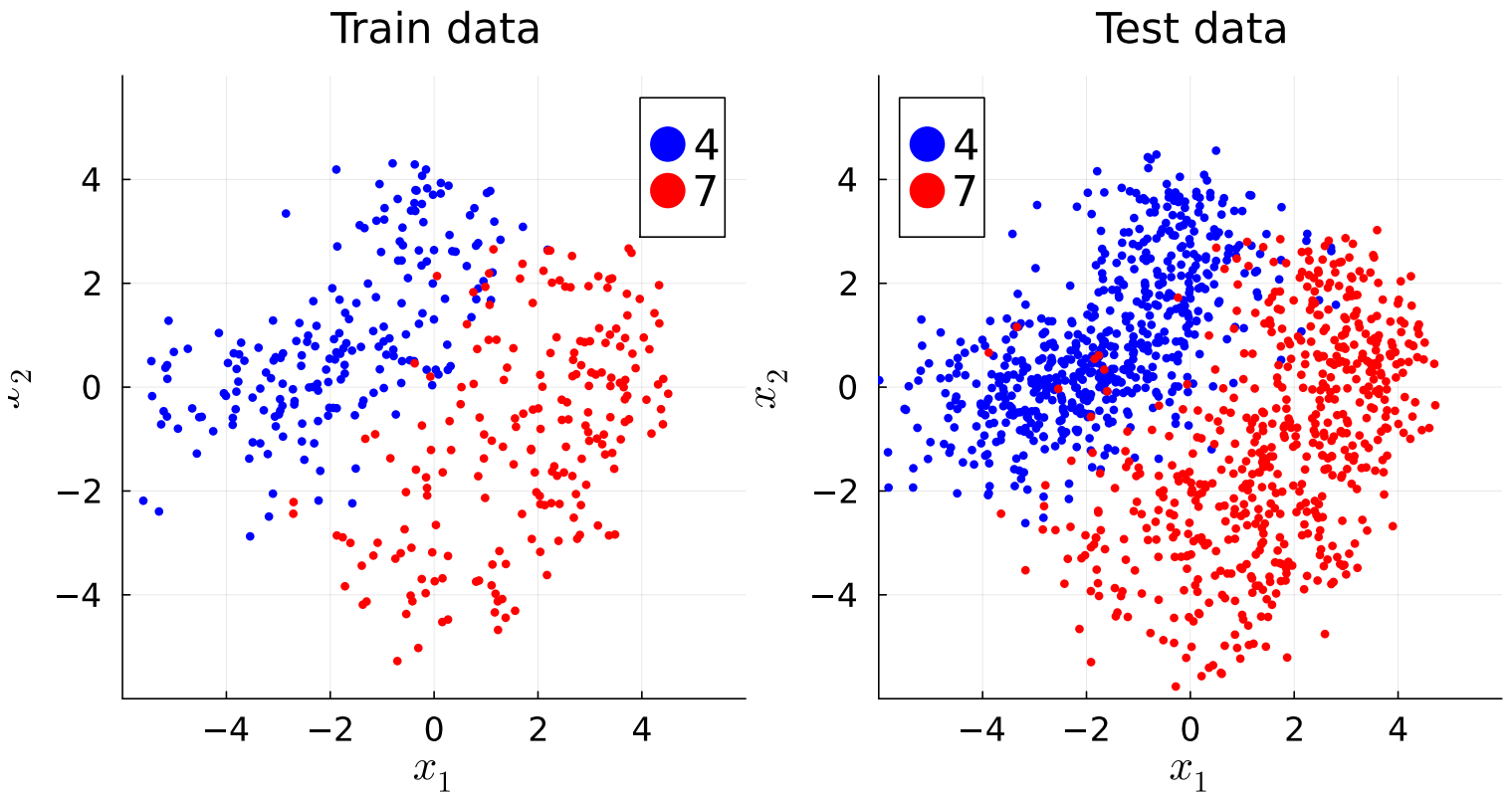
Use PCA to reduce dimension via $\boldsymbol{x} \leftarrow$





Demo: k-NN for binary classification - Data

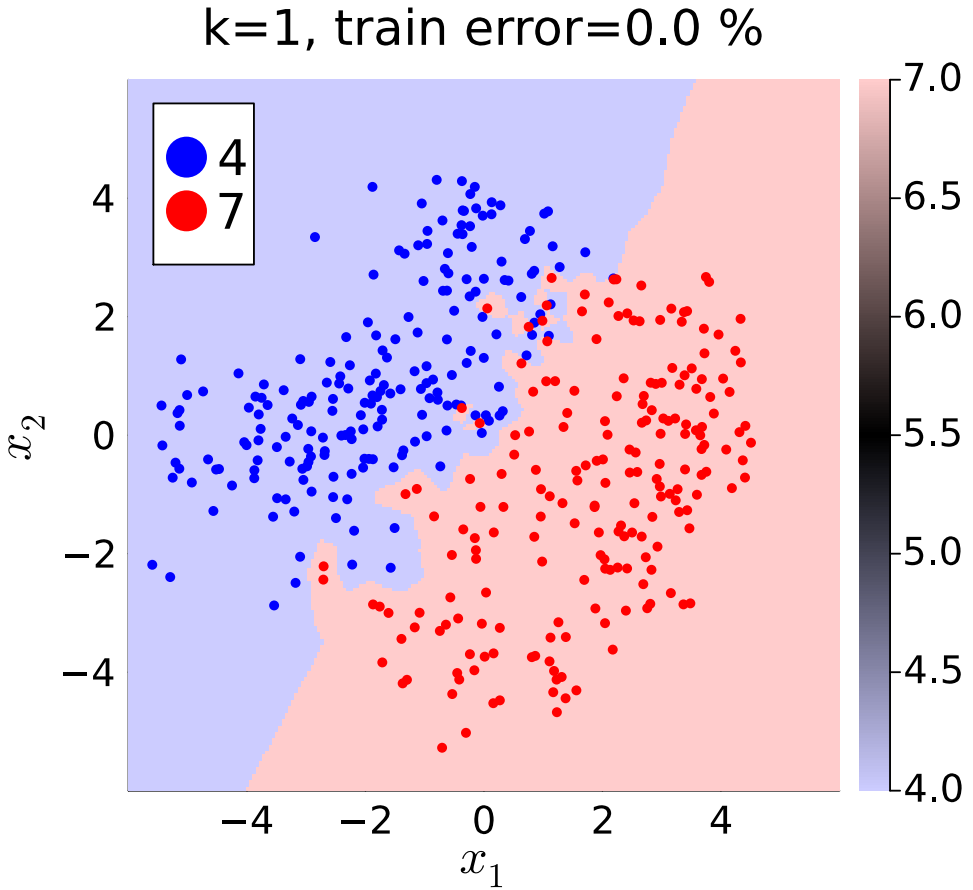
© 2026 Fessler



- Feature vector $\boldsymbol{x} \in \mathbb{R}^d$ where $d = 2$ in demo for visualization.
- Label $y \in \{\text{blue dot}, \text{red dot}\}$ for digits "4" and "7"
- $n = 200$ here (balanced data with 100 of each class)
- Our **binary classifier** is a function $f : \mathbb{R}^2 \mapsto \{4, 7\}$
- In 551 we did classification using

Demo: k-NN decision regions: $k = 1$

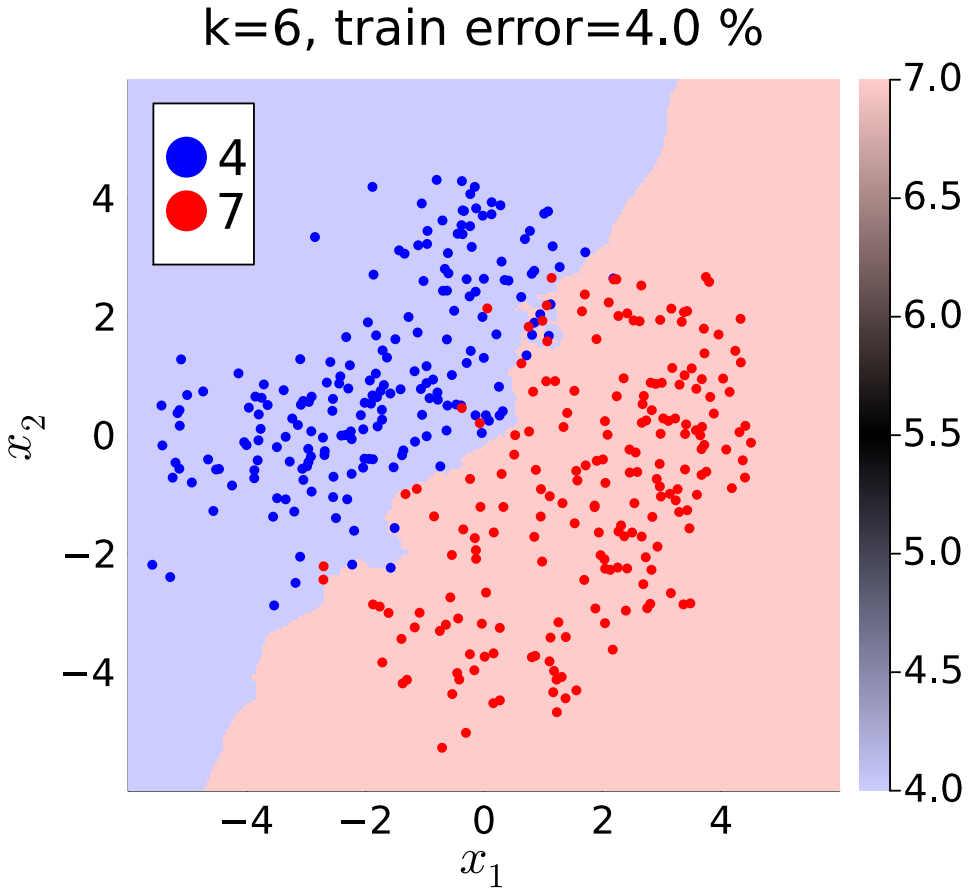
© 2026 Fessler



- Shaded **decision regions** of the NN classifier $f : \mathbb{R}^2 \mapsto \{4, 7\}$.
- The border is the **decision boundary** between class predictions for possible test data.
- Note isolated red points with small surrounding regions.

Demo: k-NN decision regions: $k = 6$

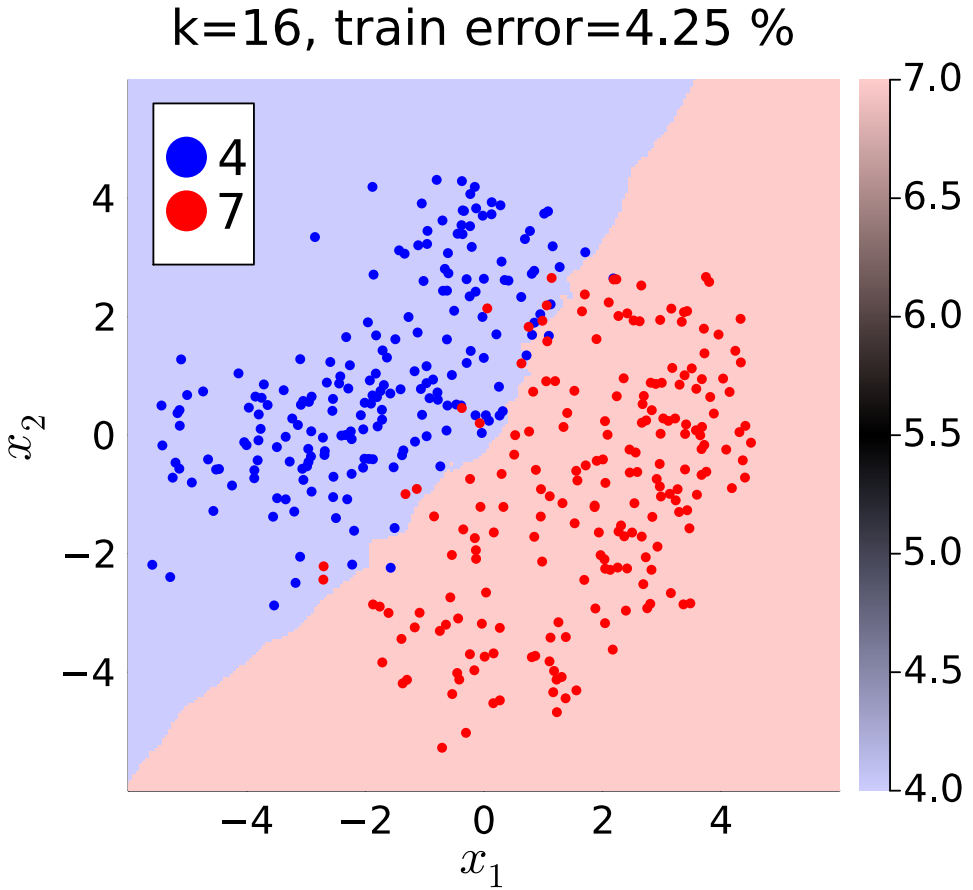
© 2026 Fessler



- Smoother decision boundaries as k increases.
- $k = 6$ happens to be optimal for the validation data

Demo: k-NN decision regions: $k = 16$

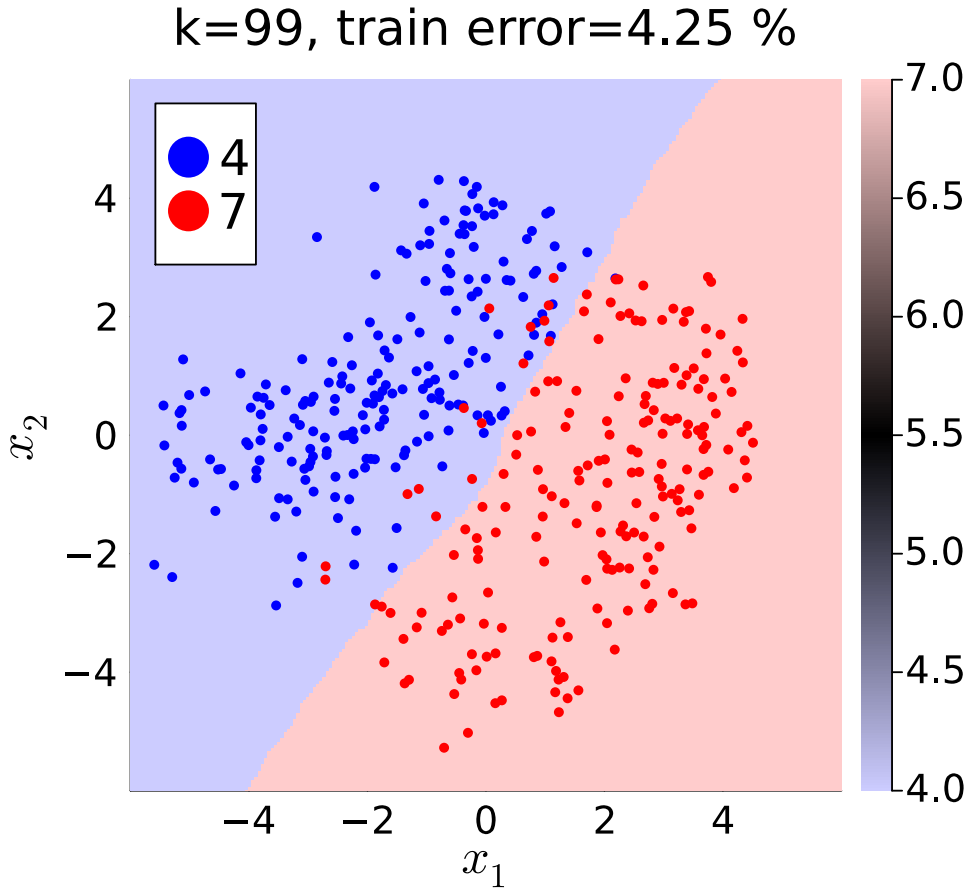
© 2026 Fessler



- Smoother decision boundaries as k increases.
- $k = 16$ happens to be optimal for the test data (see plot) but we would never know that in practice.

Demo: k-NN decision regions: $k = 99$

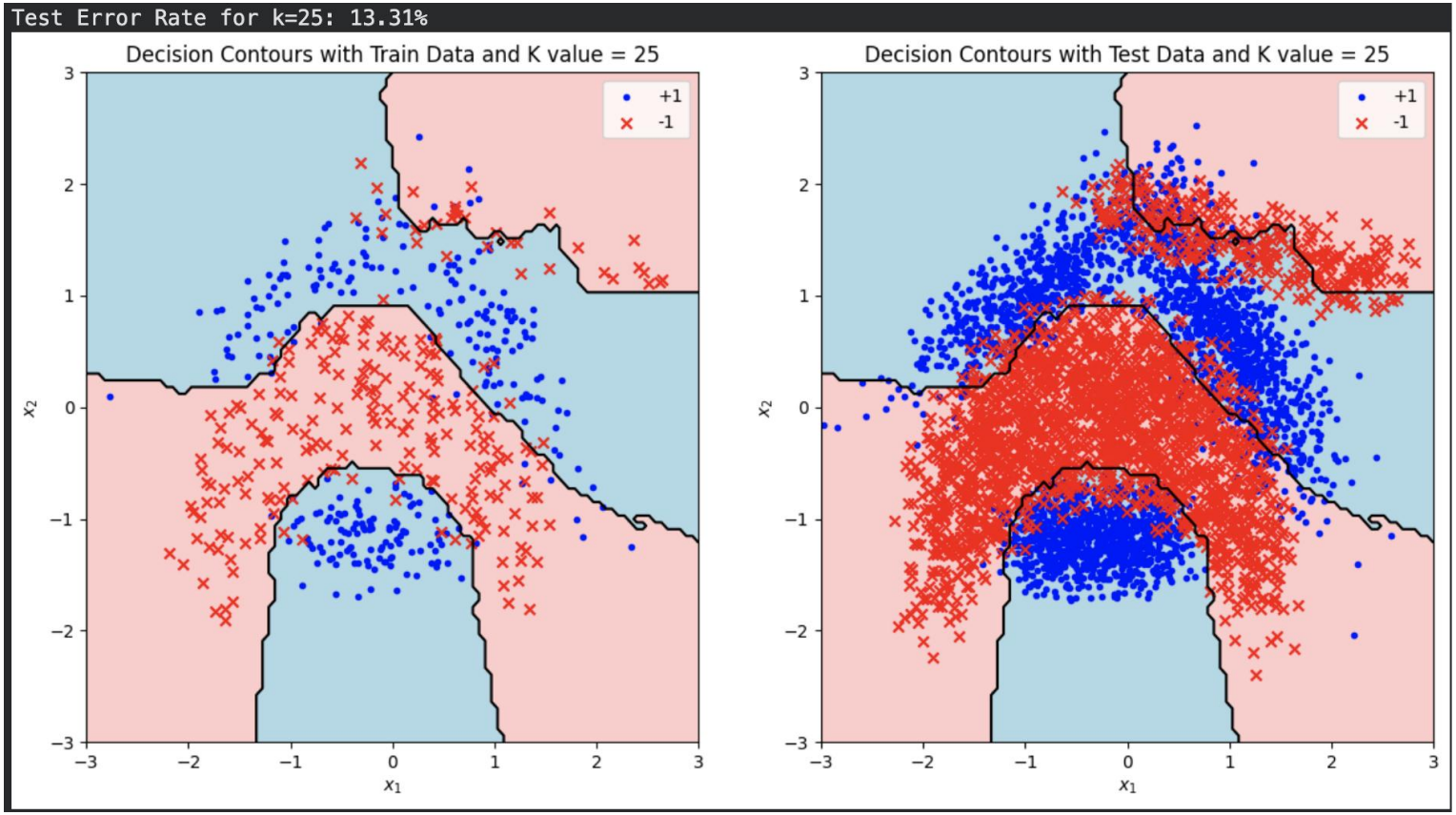
© 2026 Fessler



- Here the decision boundary is unexpectedly close to a line.

BTW, the python colab demo has synthetic data, an interactive slider and more complicated decision regions.

Colab demo: k-NN decision boundaries: $k = 25$ © 2026 Fessler



- Every classifier design involves choices.
- What are the design questions here?
  - How to choose
  - How to choose

k-NN classifier: Trade-offs

© 2026 Fessler

Training data: $\mathcal{D}_{\text{train}} = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n))$
where n denotes number of (labeled) training pairs

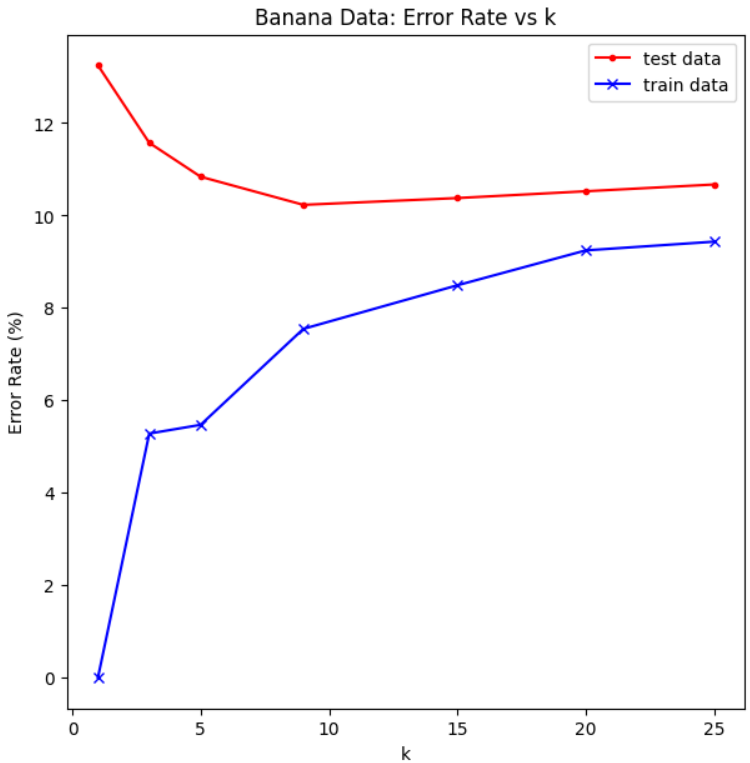
- Empirical evidence (from demo) suggests:
  - For fixed n, k is a parameter that affects the smoothness/complexity of the classifier: Larger k means more smoothness/less complexity.
  - So k controls a trade-off between
- Not seen in this demo:
  - As n increases,

k-NN classifier: Hyperparameter choice

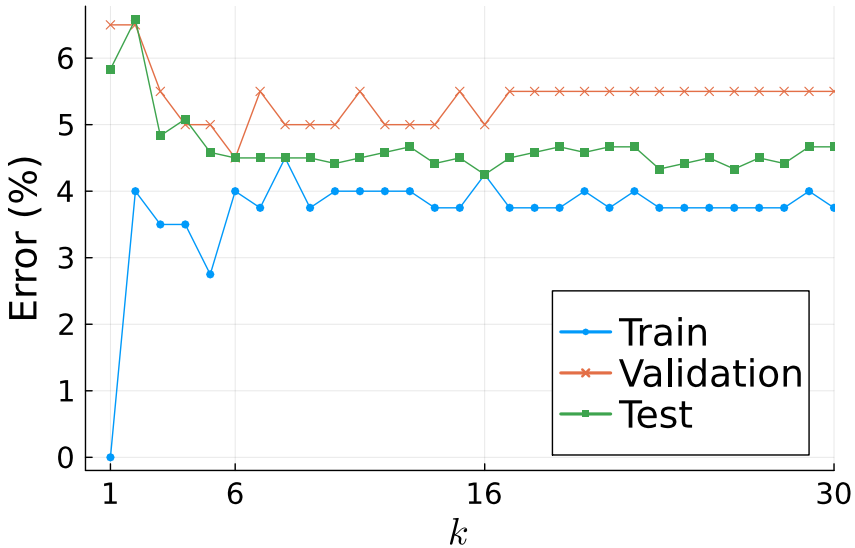
© 2026 Fessler

The k-NN method is classic, but already illustrates universal over-fit / under-fit trade-offs in ML.

Python "Banana" Demo



Julia MNIST Digit Demo



Later we will elaborate on hyperparameter selection methods like holdout, validation data, and **cross-validation**.

Q01.5 Suppose 55% of training data is class1 and 45% of data is class2. What happens to (class1, class2) percent *training* error of k-NN as $k \to n$?
A: (0, 100) B: (45, 55) C: (55, 45) D: (100, 0) E: None of these
??

Q01.6 An ML algorithm is **generative** if it is based on a full **probabilistic model** of the observed data.
An ML algorithm is **discriminative** if it assumes a partial or no probabilistic model.
The k-NN classifier is?
A: generative B: discriminative **??**

CQ: Parametric/nonparametric

© 2026 Fessler

Q01.7 A learning algorithm is **parametric** if the number of model parameters* is independent of sample size n.
An algorithm is **nonparametric** if the number of model parameters* grows with sample size n.
The k-NN classifier is?

A: parametric

B: nonparametric because of properties of k

C: nonparametric for another reason

D: none of these

*Model parameters = parameters needed to represent the function (=model) produced by the learning algorithm

Pros and cons of k-NN classifier

© 2026 Fessler

- Disadvantages
  - Evaluation time using **big O** notation.
  - Affected by curse of dimensionality: in high dimensional spaces, distances between points become more similar [1, §16.1.2].
- Advantages
  - Very simple, easy to implement
    (no training time if k and $\|\cdot\|$ are just picked by hand).
  - Often a good baseline or first method to try.
  - Works with any notion of distance $d(\boldsymbol{x}, \boldsymbol{x}_i)$ not just $\|\boldsymbol{x} - \boldsymbol{x}_i\|$, including *learned* distance metrics [1, §16.2].
  - Readily applies to multiclass classification (not just binary) and to **regression** via ***k*-nearest neighbor smoother** (local average)
  - Transforming to another coordinate system can help [10].
    We will discuss feature extraction / feature learning later.

k-NN scalability

© 2026 Fessler

- Approximate nearest neighbor search methods can have better scaling, such as **locality sensitive hashing** (**LSH**) [1, §16.1.3]. LSH has been used for efficiency in training deep networks [11], including for learnable sparse **attention** [1, p. 531].
- Condensed nearest neighbor rules: Remove training points that have little to no effect on the classifier
- Edited nearest neighbor rules: Remove training points whose label disagrees with its k-nearest neighbors
- k-NN has been studied theoretically for > 25 years
  As $n \to \infty$, the performance of k-NN becomes within a factor of 2 of the Bayes error [1, §16.1].

k-NN probability model

© 2026 Fessler

- We presented the k-NN method as a procedure.
- It can be also derived from a probability model [1, §16.1]:

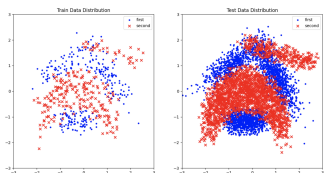
$$\mathsf{P}\{Y = c \mid \boldsymbol{x}, \mathcal{D}_{\text{train}}\} = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\boldsymbol{x}, \mathcal{D}_{\text{train}})} \mathbb{I}_{\{y_i = c\}}, \quad c = 1, \dots, K.$$

- The k-NN classifier returns the mode of this conditional distribution (the majority label) for any test vector $\boldsymbol{x} \in \mathbb{R}^d$.
- This feels like a bit of a "concocted" model, here just for completeness.
- We turn turn to probabilistic formulations of classifiers.

Goal: work towards designing *optimal* classifiers.
Optimizing requires some modeling assumptions.
ML literature abounds with probabilistic models.

- Feature vector $\boldsymbol{X} \in \mathbb{R}^d$
- Label $Y \in \{1, \ldots, K\}$
- Assume each pair $(\boldsymbol{X}, Y)$ is a

- Design classifier $f : \mathbb{R}^d \to \{1, \ldots, K\}$ in principled (optimal?) way using $\mathrm{p}(\boldsymbol{x}, y)$.
- Example: colab demo used synthetic data from some $\mathrm{p}(\boldsymbol{x}, y)$.



Q01.8 The three components of a probability space are:

A: The sample space, event space, and probability measure.

B: The sampling procedure, sigma algebra, and joint probability distribution.

C: The sample space, a partition of the real line into countably infinite disjoint sets, and the measure of those sets.

D: none of these

Probabilistic framework for classification

© 2026 Fessler

- We wish to learn/model/exploit the joint distribution of $(\boldsymbol{X}, Y)$.
  - Often (but not always) we will use "ECE 501 notation" $(\boldsymbol{X}, Y)$ for random vectors / random variables.
  - Here $\boldsymbol{X} \in \mathbb{R}^d$ is a random vector (not a matrix)
  - Later we often use $\boldsymbol{A}$ and $\boldsymbol{X}$ with bold **serif** fonts for **matrices**.
- Notation:
  - $\pi_k \triangleq \mathsf{P}\{Y = k\}$ class probabilities
  - $g_k(\boldsymbol{x}) \triangleq$ pdf/pmf of $\boldsymbol{X}$ given $Y = k$, *e.g.*, $g_k(\boldsymbol{x}) =$
    Why "g_k"?
- Learning / modeling:
  - $\{\pi_k\}$ rarely known; typically need to be learned
  - $\{g_k(\cdot)\}$ rarely known; typically need to be learned (via some parametric or nonparametric model)

Theorem (**Law of total probability**)

*Consider a probability space with a sample space, event space, and a probability measure. If $\{B_k\}$ is a finite or countably infinite **partition** of the sample space, then for any event A on the same probability space:*

$$P(A) =$$

$$= $$

For binary classification, use **partition** $B_1 = \{Y = 1\}$ and $B_2 = \{Y = 2\}$. So the (total) distribution (pdf or pmf) $g(\boldsymbol{x})$ is a **mixture distribution**:

$$g(\boldsymbol{x}) =$$

- We wish to learn/model/exploit the joint distribution of $(\boldsymbol{X}, Y)$.
- Notation:
  - $\pi_k \triangleq \mathsf{P}\{Y = k\}$
  - $g_k(\boldsymbol{x}) \triangleq$ pdf/pmf of $\boldsymbol{X}$ given $Y = k$
  - $\eta_k(\boldsymbol{x}) \triangleq$
    Why η?

Q01.9 The notation $\pi_k \triangleq \mathsf{P}\{Y = k\}$ represents the $\boxed{?}$ class probability and the notation $\eta_k(\boldsymbol{x}) \triangleq \mathsf{P}\{Y = k \mid \boldsymbol{X} = \boldsymbol{x}\}$ represents the $\boxed{?}$ class probability.

A: kth, $\boldsymbol{x}$ given k

B: conditional, marginal

C: prior, posterior

D: binary, multi-class

E: none of these

Foreshadowing the Bayes' classifier

© 2026 Fessler

- A **classifier** is a function $f : \mathbb{R}^d \mapsto \{1, \ldots, K\}$.
- Given the joint distribution of $(\boldsymbol{X}, Y)$, what is the "best" possible classifier?
- The best classifier depends on a chosen **performance measure**.
- The most common performance measure for **classification** problems is the **probability of error**, defined by:

$$R(f) \triangleq$$

i.e., the probability of the event

$$\{(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{1, \ldots, K\} \, : \, f(\boldsymbol{x}) \neq y\}.$$

- It treats all errors equally (both hard and easy cases).
- Given $\mathrm{p}(\boldsymbol{x}, y)$, a "best" classifier is $f_* \triangleq$ an **optimization problem**.

Loss and risk in general ML problems

© 2026 Fessler

- Consider a supervised learning problem with feature vectors / labels $(\boldsymbol{X}, Y)$ having joint distribution $\mathrm{p}(\boldsymbol{x}, y)$.
- Let $\mathcal{Y}$ denote the output space
  - Classification: $\mathcal{Y} =$
  - Regression: $\mathcal{Y} =$
- A **loss function** is a function $L : \mathcal{Y} \times \mathcal{Y} \mapsto [0, \infty)$:

$$L(y, \hat{y}) =$$

- Let $f : \mathbb{R}^d \mapsto \mathcal{Y}$ denote a **classifier** or **regressor**, aka **predictor**. The (statistical) **risk** of f is defined to be

$$R(f) = R_L(f) \triangleq$$

Loss and risk for binary classification

© 2026 Fessler

- For binary classification problems, often $\mathcal{Y} = \{1, -1\}$, and f is called a **decision function** or **decision rule** or **discriminant function**. The predicted label is

$$\hat{y} = f(\boldsymbol{x}) =$$

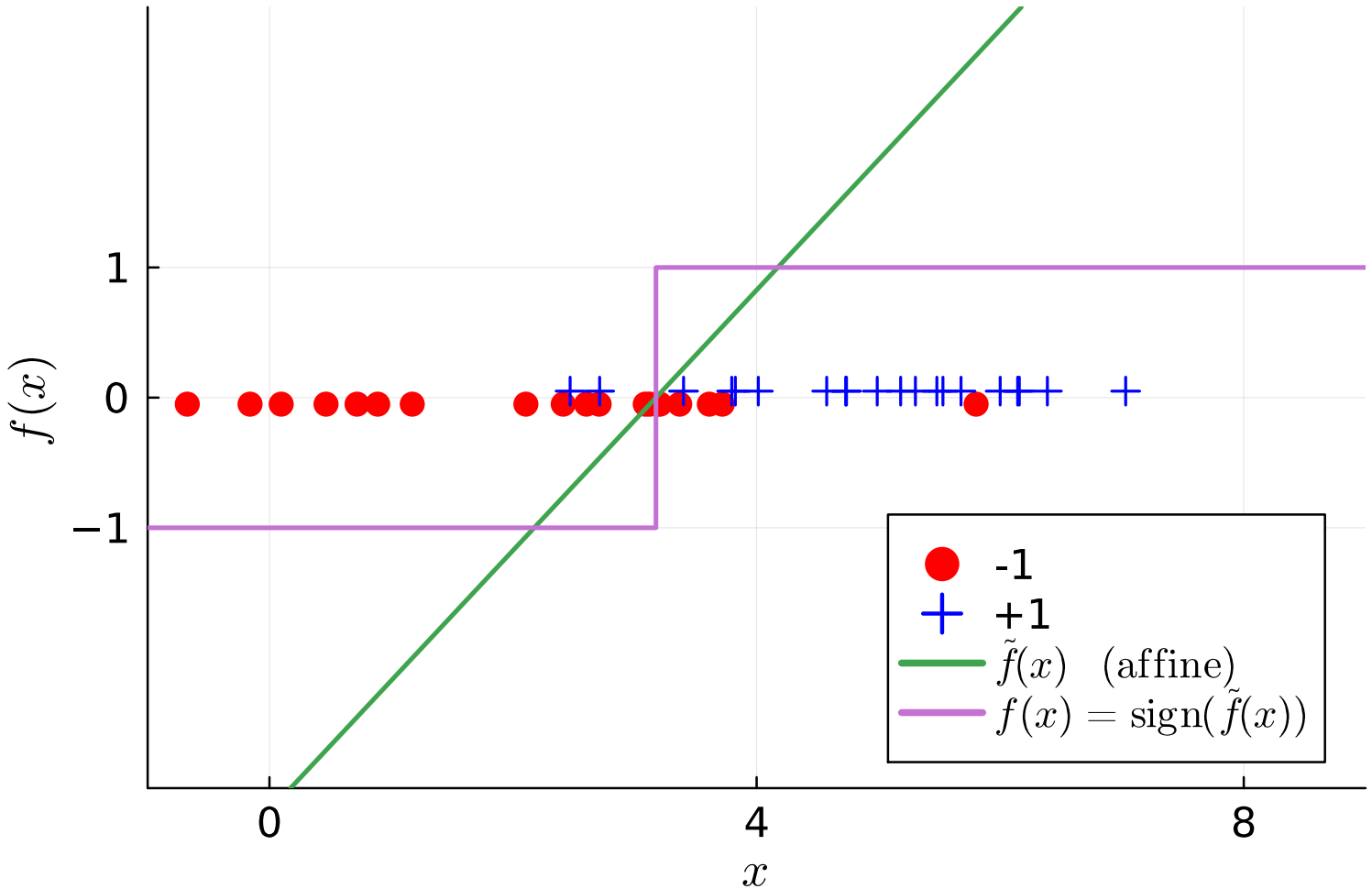
$$\tilde{f} : \mathbb{R}^d \mapsto \mathbb{R}$$
$$f : \mathbb{R}^d \mapsto \{\pm 1\}.$$

Pictures on next pages from decision boundary demo.

- For example, a "linear" classifier has $\tilde{f}(\boldsymbol{x}) =$
  - The $d + 1$ classifier parameters are $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.
  - We must **learn** these parameters from training data.

Decision boundary: 1D

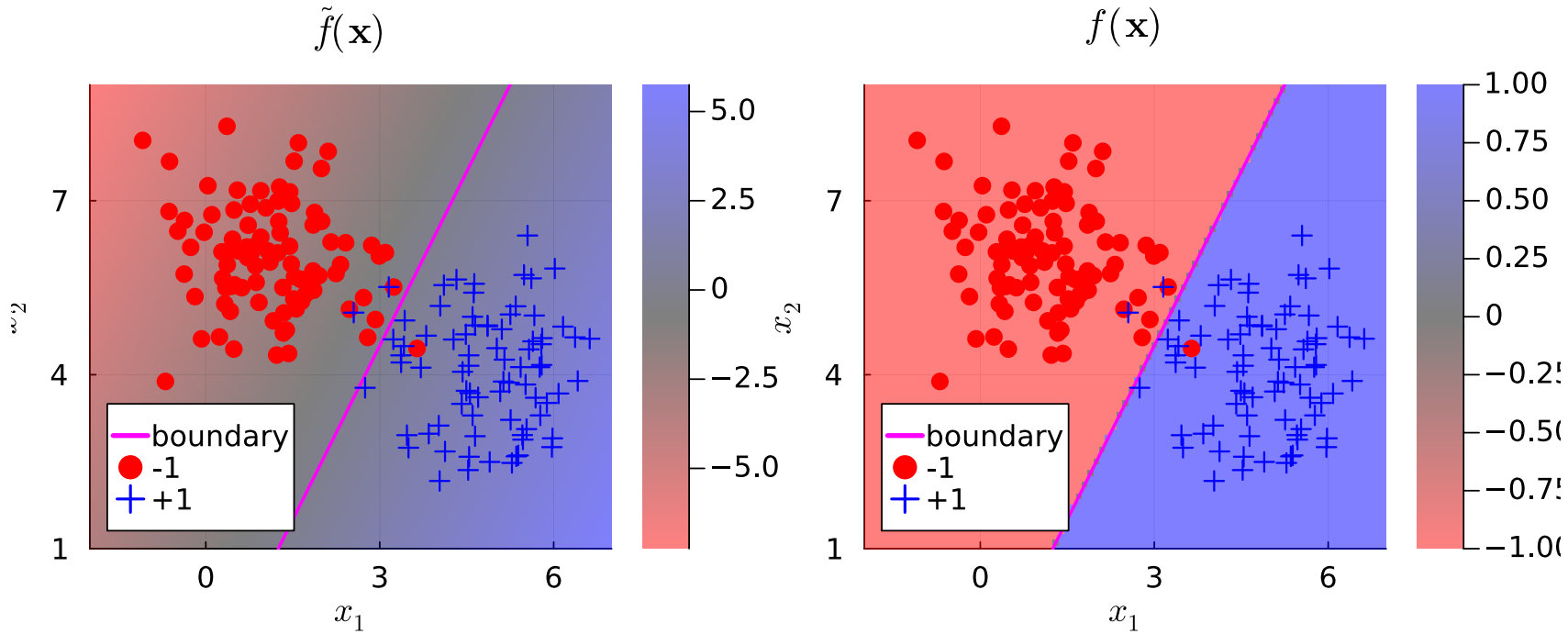
© 2026 Fessler



Here $\tilde{f}$ is an **affine function**. Geometrically the locus of points is a "line," but the function $\tilde{f}$ is not linear:

Decision boundary: 2D

© 2026 Fessler



We will describe several ways to define

The 0-1 loss and risk

© 2026 Fessler

- If L is the **0-1 loss** ("all or nothing"):

$$L_{\text{0-1}}(y, \hat{y}) \triangleq$$

$$=$$

- then the **risk** is the **probability of error**:

$$R(f)$$

- The expectation is w.r.t. the joint distribution $\mathrm{p}(\boldsymbol{x}, y)$.

Later we will discuss **surrogate losses** for the 0-1 loss for binary classification.

- Rewrite 0-1 loss:

$$L_{\text{0-1}}(y, f(\boldsymbol{x})) = \mathbb{I}_{\{y \neq f(\boldsymbol{x})\}} = \mathbb{I}_{\{y \neq \text{sign}(\tilde{f}(\boldsymbol{x}))\}} = \mathbb{I}_{\{y \cdot \tilde{f}(\boldsymbol{x}) \leq 0\}}.$$

- Hinge loss (used in **support vector machine** (**SVM**) methods):

$$L_{\text{hinge}}(y, \tilde{f}(\boldsymbol{x})) \triangleq \max\Big(0, 1 - y \cdot \tilde{f}(\boldsymbol{x})\Big).$$

- Logistic loss (used in **logistic regression**):

$$L_{\text{logistic}}(y, \tilde{f}(\boldsymbol{x})) \triangleq \log\Big(1 + \mathrm{e}^{-y \cdot \tilde{f}(\boldsymbol{x})}\Big).$$

  To be precise, this loss uses the negative log of the **standard logistic function** $z \mapsto 1/(1 + \exp(-z))$.

- They all are non-increasing functions of $y \cdot \tilde{f}(\boldsymbol{x})$.

Looking ahead

© 2026 Fessler

- Upcoming topics: Bayes classifier, LDA, QDA, ERM, optimization
  - Our goal in general will be to design ML methods that minimize risk, *i.e.*, minimize loss according to our chosen loss function.
  - Such minimization involves statistical models and optimization
- Next few weeks: the core introductory algorithms of supervised learning, derived from basic principles with applications to real data on the homework

References I

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

[2] Y. Song, L. Shen, L. Xing, and S. Ermon.
Solving inverse problems in medical imaging with score-based generative models.
In *Proc. Intl. Conf. on Learning Representations*, 2022.

[3] M. Brcic and S. Frljic.
The effortless trap: productive struggle, AI, and the illusion of learning, 2026.

[4] K. P. Murphy.
Probabilistic machine learning: Advanced topics.
MIT Press, 2023.

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

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

[7] A. Krizhevsky and G. Hinton.
Learning multiple layers of features from tiny images, 2009.

[8] H. Xiao, K. Rasul, and R. Vollgraf.
Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms, 2017.

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

References II

[10] O. Boiman, E. Shechtman, and M. Irani.
In defense of nearest-neighbor based image classification.
In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, pages 1–8, 2008.

[11] B. Chen, T. Medini, J. Farwell, S. Gobriel, C. Tai, and A. Shrivastava.
SLIDE: In Defense of Smart Algorithms over Hardware Acceleration for Large-Scale Deep Learning Systems.
In *MLSys*, 2020.