

Learning from data: tasks, representations and notation

Prof. Dr. Emre Onur Kahya · Department of Physics Engineering, Istanbul Technical University

Machine learning replaces hand-written rules by functions estimated from examples. This note fixes the vocabulary and notation used throughout the course: what a dataset is, how an observation becomes a vector, which kinds of learning problems exist and how success is measured. Later notes refer back to Section 2 (notation) and Table 2 (tasks and losses) instead of repeating them.

1 Learning from examples

Suppose we want a program that decides whether a photograph shows a horse. Writing the rules by hand — four legs, a mane, a long head, unless it is a donkey, unless the horse is lying down — quickly becomes hopeless. The machine-learning approach is different: collect many photographs labelled “horse” or “not horse” and let an algorithm *find* a function that maps images to labels. The principle is far more general than any single application: the same algorithm, given different examples, learns to recognise galaxies, transcribe speech or predict the energy of a particle shower.

Key idea Learning is not memorising. A table that stores every training image with its label answers perfectly on those images and is useless on a new one. What we want is *generalisation*: good predictions for cases never seen during training. Note 4 makes this precise and shows how to measure it honestly.

Modern connection — rules versus data

The largest systems built today follow this principle at an enormous scale. A large language model is not given a grammar of English or Turkish; it is trained to predict the next piece of text from the previous ones on trillions of words, and grammar, facts and some reasoning skills emerge as by-products [14, 2]. Image classifiers trained on ImageNet’s 1.28 million labelled photographs [13] replaced decades of hand-designed vision pipelines.

In physics — from cuts to classifiers

Particle-physics analyses traditionally separated signal from background with hand-tuned *cuts* on a few observables (“transverse momentum above 25 GeV, invariant mass within 10 GeV of the Z peak”). Today the same decision is often made by a classifier trained on simulated events; for identifying boosted top quarks, for example, learned taggers that read low-level detector information outperform the best hand-designed variables [8]. The physics knowledge does not disappear — it moves into the simulation, the choice of inputs and the validation of the classifier.

2 Data and notation

Examples. An *example* (observation, sample) is a pair $(\mathbf{x}, y)$. The *input* $\mathbf{x}$ is a vector of d real numbers, $\mathbf{x} = (x_1, \dots, x_d)^\top \in \mathbb{R}^d$, whose components are called *features*. The *target* (label, desired output) y is what we want to predict. A *training set* of n examples is

$$D_n = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}.$$

Stacking the inputs as rows gives the *design matrix* $X \in \mathbb{R}^{n \times d}$ whose i -th row is $\mathbf{x}_i^\top$; the targets form a vector $\mathbf{y} \in \mathbb{R}^n$ (or, for m classes, a matrix of one-hot rows). Subscripts index examples ($\mathbf{x}_i$), and when needed a bracket indexes features: $x_{i,j}$ or $\mathbf{x}_i[j]$ is feature j of example i . A *predictor* is a function $f : \mathbb{R}^d \rightarrow \mathcal{Y}$; a new input whose target is unknown is a *test point*.

Problem size. Three numbers decide which algorithms are even feasible: the number of examples n , the input dimension d and the number of outputs (for classification, the number of classes m). Table 1 gives orders of magnitude.

Table 1. Typical problem sizes. The cost of most algorithms grows with n , d and m in very different ways.

	n (examples)	d (input dimension)	m (classes)
Iris flowers (1936)	150	4	3
Handwritten digits (MNIST)	60 000	784 (28×28 pixels)	10
ImageNet classification	1.28×10^6	150 528 (224×224×3)	1000
Language-model pre-training	10^{12} – 10^{13} tokens	embedding size $\sim 10^3$ – 10^4	vocabulary $\sim 10^5$
Jet tagging at the LHC	10^6 – 10^7 simulated jets	10^2 – 10^3 constituents' features	2–10

The i.i.d. assumption. We model each example $Z_i = (\mathbf{X}_i, Y_i)$ as a draw from an *unknown* joint distribution $P(\mathbf{X}, Y)$ – “nature” or the experiment. Unless stated otherwise the examples are assumed *independent and identically distributed* (i.i.d.): every example comes from the same distribution, independently of the others. Two consequences are used constantly: the order of the examples carries no information (shuffling the dataset changes nothing), and a randomly chosen subset is a fair sample of the whole.

Not every dataset is i.i.d. Time series (tomorrow depends on today), grouped data (several images of the same galaxy or patient), data whose distribution changes over time, and models trained on simulation but applied to real detector data (the *simulation-to-reality gap*) all violate the assumption. Methods that ignore this produce optimistic results; Note 4 shows how to split such data correctly.

Geometric picture. A dataset is a cloud of n points in $\mathbb{R}^d$, each carrying a label. A test point is a new location in that space, and learning means deciding what to predict *everywhere* in $\mathbb{R}^d$, including in regions where no training point lies. This picture – points in a high-dimensional space – underlies every method in weeks 1–3, and its difficulties in large d are the subject of Note 3.

3 From raw observations to vectors

Most learning algorithms of this course require each example to be a vector of fixed dimension d . How we build that vector – the *representation* – often matters more than the choice of algorithm.

Raw representations. A grey-scale image of 28×28 pixels becomes a vector of $d = 784$ intensities; a colour image of 224×224 pixels has $d = 150\,528$. A waveform or a detector read-out can be used as is. Raw vectors are easy to produce but awkward: a shift of the object by one pixel changes almost every coordinate.

Hand-crafted features. Domain experts design a few informative quantities. For documents, the *bag of words* counts how often each word of a vocabulary occurs, so “the cat jumped” becomes a long vector of zeros with three ones; it forgets word order, so “dog bites man” and “man bites dog” get the same vector. In vision, histograms of oriented gradients and SIFT descriptors [4, 9] summarise local edge directions. In physics, features are often physical observables: invariant masses, transverse momenta, or jet-substructure variables built from the theory of QCD.

Learned representations. Deep networks learn the features themselves from raw inputs, layer by layer, from edges to parts to objects. The learned intermediate vectors — *embeddings* — are now reused far beyond the task they were trained for: a word or token is mapped to a vector of a few hundred to a few thousand dimensions (768 in BERT-base [5]), and models such as CLIP map images and captions into a *shared* space where matching pairs are close [12]. For text, the raw input is first split into *tokens* from a vocabulary of 10^4 – 10^5 pieces, and each token is looked up in an embedding table.

Modern connection — representations as a product

A common recipe in 2026 is to take a large pre-trained model, keep it frozen, and train only a simple classifier — or even a nearest-neighbour rule (Note 5) — on its embeddings. The algorithms of weeks 1–3 are therefore not museum pieces: they are exactly what is applied on top of modern representations.

4 Families of learning problems

Supervised learning. predicts a target y from an input x . In *classification* y is a category (binary, $y \in \{0, 1\}$ or $\{-1, +1\}$; or multiclass, $y \in \{1, \dots, m\}$). In *regression* y is a real number or a vector of real numbers.

Unsupervised learning. has no explicit target. *Density estimation* models the distribution $p(x)$ of the inputs; *clustering* groups similar examples; *dimensionality reduction* finds a few coordinates that describe the data (for visualisation or compression); *representation learning* builds useful features.

Between the two. *Semi-supervised* learning uses a few labelled and many unlabelled examples. *Self-supervised* learning manufactures targets from the data itself: predict the next token of a text, a masked patch of an image, or which caption belongs to which image. Next-token prediction is, formally, multiclass classification with m equal to the vocabulary size, repeated at every position of every text — which is why the simple losses below also drive the largest models.

Reinforcement learning. An agent chooses actions in an environment to maximise a cumulative reward; it is used, for example, to fine-tune language models from human preference judgements [11]. It is outside the scope of this course.

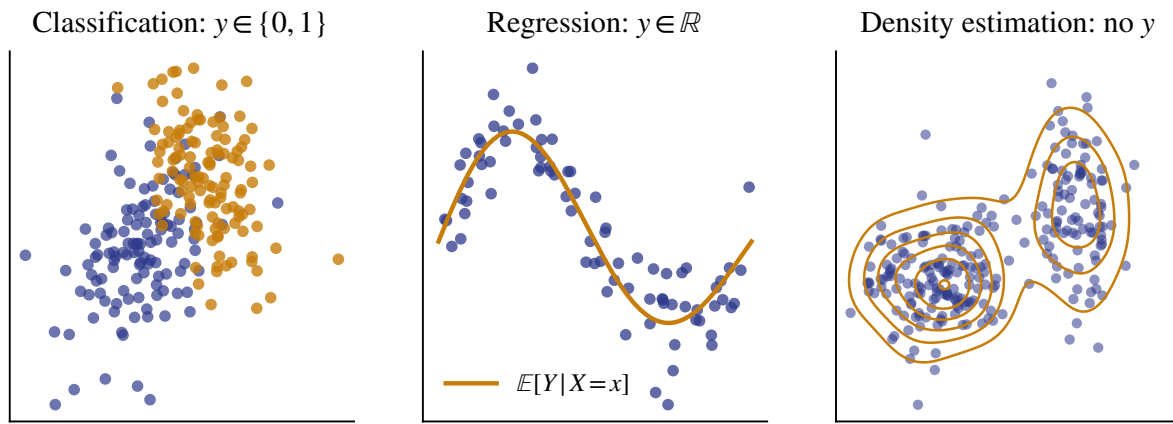


Figure 1. Three tasks. Left: classify events into signal (amber) and background (indigo) from two observables. Middle: regression; the curve is the conditional mean $\mathbb{E}[Y | X = x]$ that a good regressor approximates. Right: density estimation; contours of an estimate of $p(\mathbf{x})$ (Note 5 explains how it was computed).

Table 2. The three classic problems. This table is the reference for the rest of the course.

	Target	What $f(\mathbf{x})$ should be	Typical loss L
Classification	$y \in \{1, \dots, m\}$	the most probable class, or the vector of probabilities $P(Y = k \mathbf{X} = \mathbf{x})$	error $\mathbf{1}\{f(\mathbf{x}) \neq y\}$
Regression	$y \in \mathbb{R}$	the conditional mean $\mathbb{E}[Y \mathbf{X} = \mathbf{x}]$	squared error $(f(\mathbf{x}) - y)^2$
Density estimation	none	the density $p(\mathbf{x})$	negative log-likelihood $-\log f(\mathbf{x})$

Why these targets? For squared error, conditioning on $\mathbf{X} = \mathbf{x}$ and writing $\mu(\mathbf{x}) = \mathbb{E}[Y | \mathbf{X} = \mathbf{x}]$,

$$\mathbb{E}[(f(\mathbf{x}) - Y)^2 | \mathbf{X} = \mathbf{x}] = (f(\mathbf{x}) - \mu(\mathbf{x}))^2 + \text{Var}(Y | \mathbf{X} = \mathbf{x}),$$

so the best possible prediction is $\mu(\mathbf{x})$, and the remaining $\text{Var}(Y | \mathbf{X} = \mathbf{x})$ is noise that no predictor can remove. Similarly, the expected classification error at $\mathbf{x}$ is $1 - P(Y = f(\mathbf{x}) | \mathbf{X} = \mathbf{x})$, minimised by the most probable class (Exercise E3). The irreducible error that remains is called the *Bayes error*.

One dataset, several tasks. The task is a choice of the analyst, not a property of the data. From the same galaxy survey one can classify morphology (spiral or elliptical), regress the redshift from the measured colours (“photometric redshifts”), or estimate the density of objects in colour space and flag the rare ones that fall in low-density regions — a standard route to discovering unusual objects.

5 Anatomy of a machine-learning project

1. **Formulate.** Which task of Table 2 is it? Which loss reflects what you actually care about (a missed signal event and a false alarm rarely cost the same)?
2. **Understand the data.** How many examples, what is one example, how were labels obtained (by experts, by simulation, by an older algorithm?), and what is the unit of independence?
3. **Split first.** Set aside validation and test data *before* looking for patterns (Note 4).
4. **Pre-process using training data only.** Scaling constants, vocabularies and feature selections are part of the model; computing them on all the data leaks information from the test set.

5. **Start simple.** A constant predictor, a linear model or a nearest-neighbour rule gives a baseline. Debug on low-dimensional problems where the data and the decision boundary can be plotted and training takes seconds.
6. **Evaluate out of sample,** compare models fairly and report uncertainty (Note 4).
7. **Deploy and monitor.** The data distribution can drift after deployment.

Modern connection — leakage in published science

A survey of machine-learning-based science found data leakage — for example pre-processing on the full dataset, duplicates across splits, or features that encode the label — in hundreds of published papers across seventeen fields, often turning a useless model into an apparently excellent one [7]. Steps 3 and 4 above are the cheapest insurance against it. Documenting how a dataset was collected (“datasheets for datasets” [6]) helps others detect such problems.

6 A vector toolkit

For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ the *inner product* is $\mathbf{a}^T \mathbf{b} = \sum_{j=1}^d a_j b_j$. The L_p norm is $\|\mathbf{a}\|_p = (\sum_j |a_j|^p)^{1/p}$ for $p \geq 1$; $p = 2$ gives the Euclidean norm $\|\mathbf{a}\| = \sqrt{\mathbf{a}^T \mathbf{a}}$, $p = 1$ the sum of absolute values and $p \rightarrow \infty$ the largest absolute value. The Euclidean distance is $\|\mathbf{a} - \mathbf{b}\|$, and the *cosine similarity* $\cos(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{b} / (\|\mathbf{a}\| \|\mathbf{b}\|)$ compares directions only — the standard way to compare embeddings.

Distances in bulk. Many methods need the distances between q query points (rows of $Q \in \mathbb{R}^{q \times d}$) and n data points (rows of X). Expanding the square,

$$\|\mathbf{a} - \mathbf{b}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - 2 \mathbf{a}^T \mathbf{b},$$

so the whole $q \times n$ matrix of squared distances is two vectors of squared norms plus the single matrix product $-2QX^T$. Matrix products are what modern hardware does best; the loop over pairs disappears.

```
import numpy as np
def sq_distances(Q, X):
    # Q: (q, d), X: (n, d)
    D = (Q**2).sum(1)[:, None] + (X**2).sum(1)[None, :] - 2 * Q @ X.T
    return np.maximum(D, 0) # round-off can make tiny values negative
```

The cost is $O(qnd)$ operations either way; the gain is that they run as one optimised matrix multiplication. Note 5 builds the nearest-neighbour algorithm on this function.

Summary An example is a pair $(\mathbf{x}, y)$ with $\mathbf{x} \in \mathbb{R}^d$; a dataset is n such pairs, assumed i.i.d. from an unknown $P(\mathbf{X}, Y)$. Representations range from raw measurements through hand-crafted features to learned embeddings. Tasks are supervised (classification, regression), unsupervised (density estimation and others) or in between (self-supervised); each comes with a natural target and loss (Table 2). The goal is always generalisation to new data.

Exercises

- E1.** What is the input dimension d of (a) a 64×64 colour image, (b) a one-hot encoding of a word from a vocabulary of 50 000 words, (c) a sequence of 128 tokens, each embedded in $\mathbb{R}^{768}$, flattened

into one vector?

- E2. Prove the decomposition of the conditional expected squared error in Section 4 and conclude that $\mathbb{E}[Y \mid \mathbf{X} = \mathbf{x}]$ is the best regressor.
- E3. Show that the expected classification error at $\mathbf{x}$ equals $1 - P(Y = f(\mathbf{x}) \mid \mathbf{X} = \mathbf{x})$ and is minimised by $f(\mathbf{x}) = \arg \max_k P(Y = k \mid \mathbf{X} = \mathbf{x})$.
- E4. Is the i.i.d. assumption reasonable for a random train/test split of (a) collision events recorded in one run of an experiment, (b) consecutive frames of the same video, (c) data from 2024 used to predict 2025, (d) galaxy images where each galaxy was photographed several times? For each “no”, propose a better split.
- E5. Implement `sq_distances`, check it against a double loop on small random matrices, and compare running times for $q = 1000$, $n = 10\,000$, $d = 128$.
- E6. Give two sentences with identical bag-of-words vectors and different meanings. Which kind of representation of Section 3 can tell them apart, and why?

Further reading. Bishop [1], chapter 1; Murphy [10], chapter 1; for applications across physics, the review by Carleo *et al.* [3].

References

- [1] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer (2006).
- [2] T. Brown *et al.*, Language models are few-shot learners, *NeurIPS* (2020).
- [3] G. Carleo *et al.*, Machine learning and the physical sciences, *Rev. Mod. Phys.* 91, 045002 (2019).
- [4] N. Dalal and B. Triggs, Histograms of oriented gradients for human detection, *CVPR* (2005).
- [5] J. Devlin *et al.*, BERT: pre-training of deep bidirectional transformers for language understanding, *NAACL* (2019).
- [6] T. Gebru *et al.*, Datasheets for datasets, *Commun. ACM* 64(12), 86–92 (2021).
- [7] S. Kapoor and A. Narayanan, Leakage and the reproducibility crisis in machine-learning-based science, *Patterns* 4(9), 100804 (2023).
- [8] G. Kasieczka *et al.*, The machine learning landscape of top taggers, *SciPost Phys.* 7, 014 (2019).
- [9] D. G. Lowe, Distinctive image features from scale-invariant keypoints, *Int. J. Comput. Vis.* 60, 91–110 (2004).
- [10] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*, MIT Press (2022).
- [11] L. Ouyang *et al.*, Training language models to follow instructions with human feedback, *NeurIPS* (2022).
- [12] A. Radford *et al.*, Learning transferable visual models from natural language supervision, *ICML* (2021).
- [13] O. Russakovsky *et al.*, ImageNet large scale visual recognition challenge, *Int. J. Comput. Vis.* 115, 211–252 (2015).
- [14] A. Vaswani *et al.*, Attention is all you need, *NeurIPS* (2017).

The selection and order of topics in weeks 1–3 follow the course IFT 6390 *Fondements de l’apprentissage machine* at Université de Montréal (taught by Ioannis Mitliagkas, slides by Pascal Vincent). The text, examples, figures and exercises of this note are original. Figures are computed from synthetic data by `figures.py`, published with the sources of these notes.