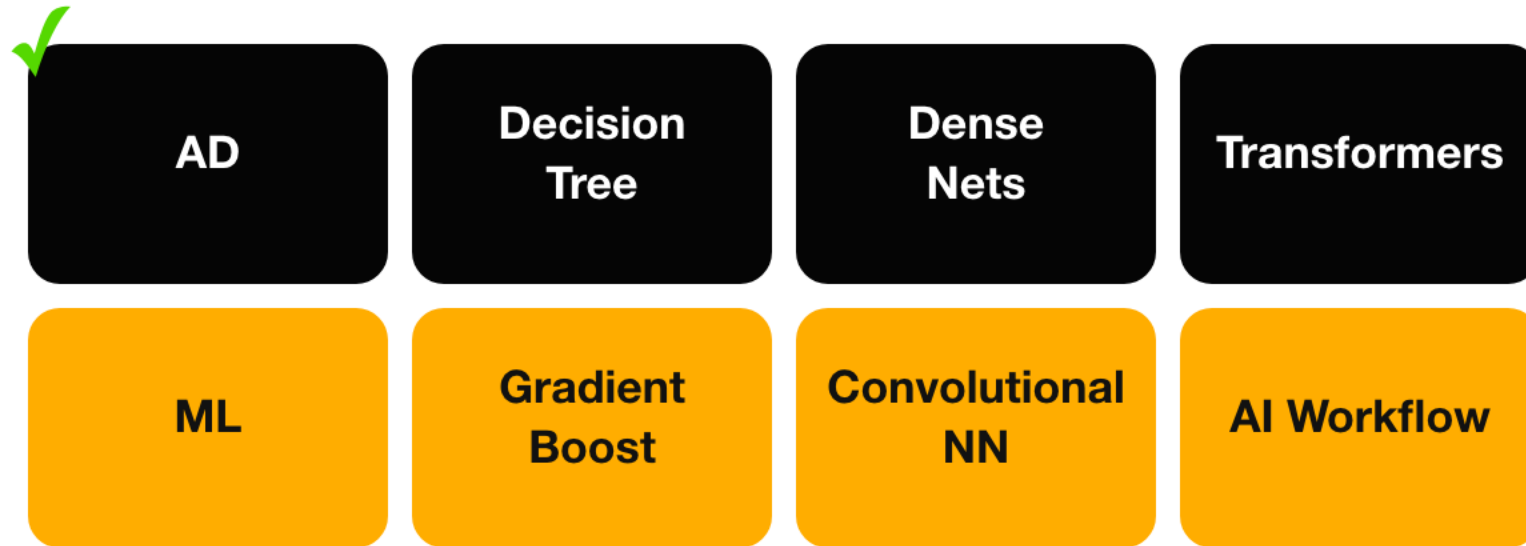


Key notions in ML

Introduction

Mikhail Mikhasenko, 29/07/2025

Plan for the course



Supervised learning

Labelled data

Machine-learning language:

- x is a **feature vector**, $x = (x^{f_1}, x^{f_2}, \dots, x^{f_N})$
- y is a **label**: a value, a vector, or a class such as cat/dog or signal/background
- The data are pairs: $\{(x_i, y_i)\}_{i=1}^n$

$$f(x) = y$$

we want to learn

Regression and classification

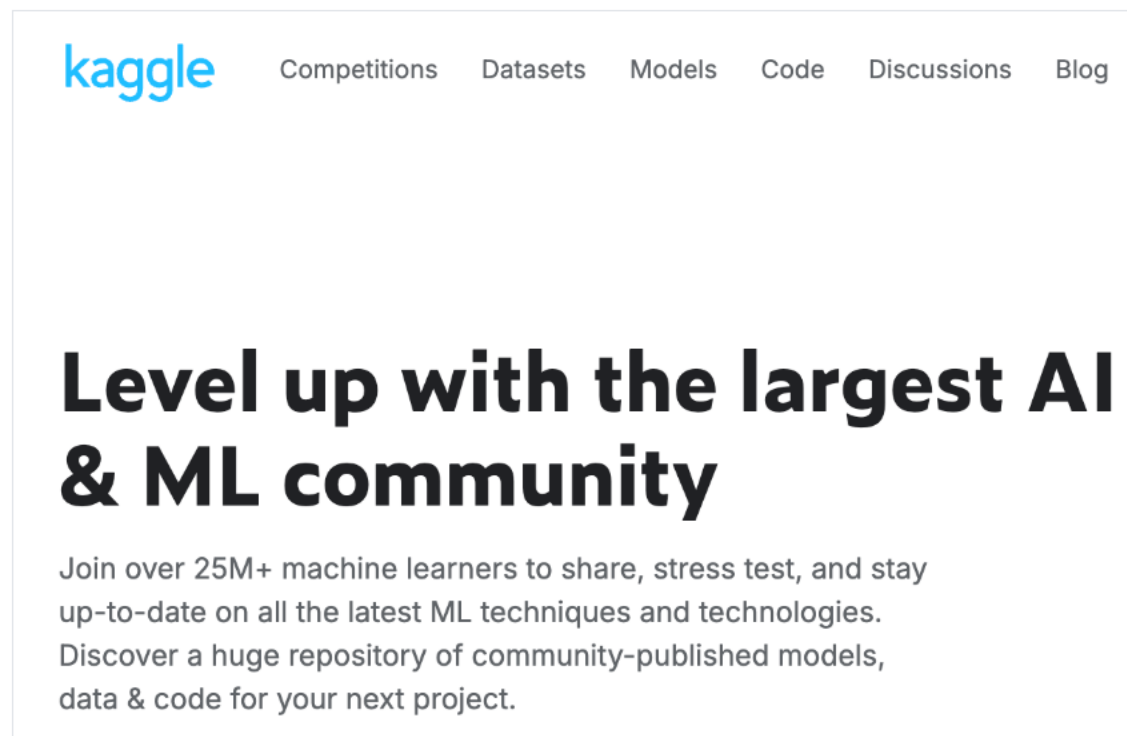
- **Classification:** y belongs to a discrete set of classes
- **Regression:** y belongs to a continuous space

$$f(x) = y$$

Labels and features

Kaggle and ML datasets

- Large collection of datasets and machine-learning challenges
- Public notebooks, benchmarks, and competitions

A screenshot of the Kaggle website's header and hero section. The header features the Kaggle logo in blue, followed by navigation links: Competitions, Datasets, Models, Code, Discussions, and Blog. The hero section has a large, bold heading "Level up with the largest AI & ML community" and a paragraph of text below it.

kaggle Competitions Datasets Models Code Discussions Blog

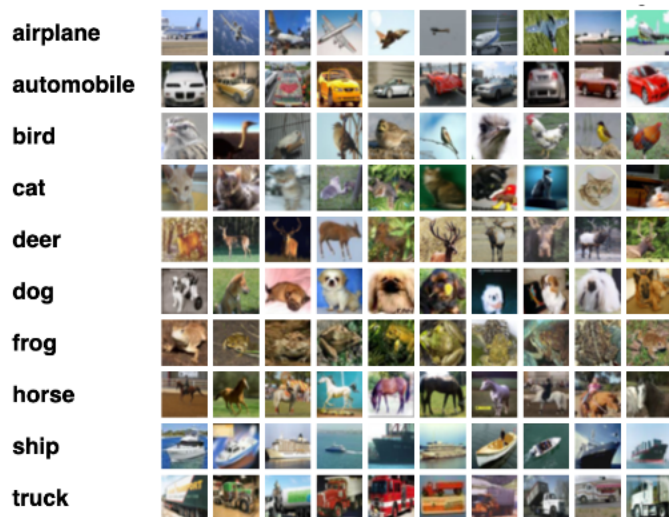
Level up with the largest AI & ML community

Join over 25M+ machine learners to share, stress test, and stay up-to-date on all the latest ML techniques and technologies. Discover a huge repository of community-published models, data & code for your next project.

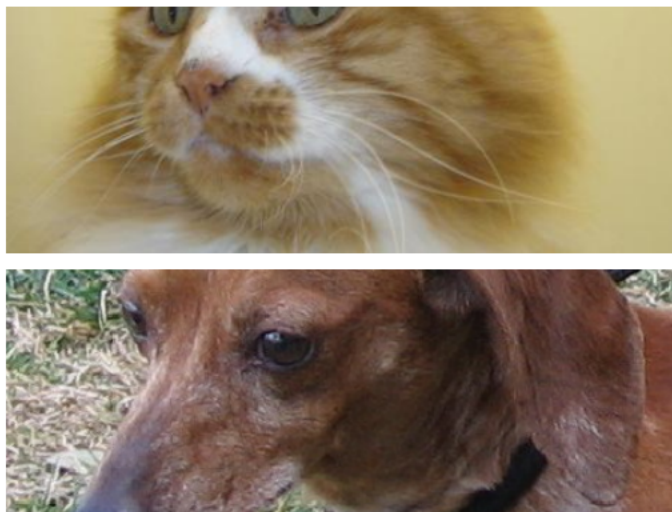
Example 1: Image classification

- **Problem:** recognize the object in an image
- **Label:** class name

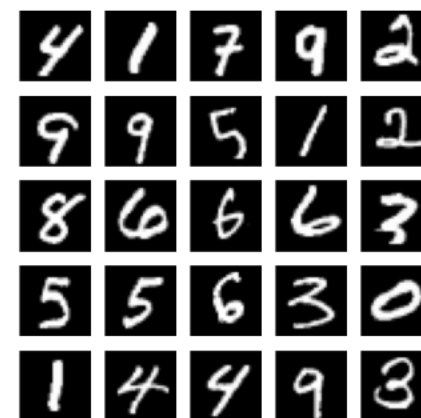
[CIFAR-10](#)



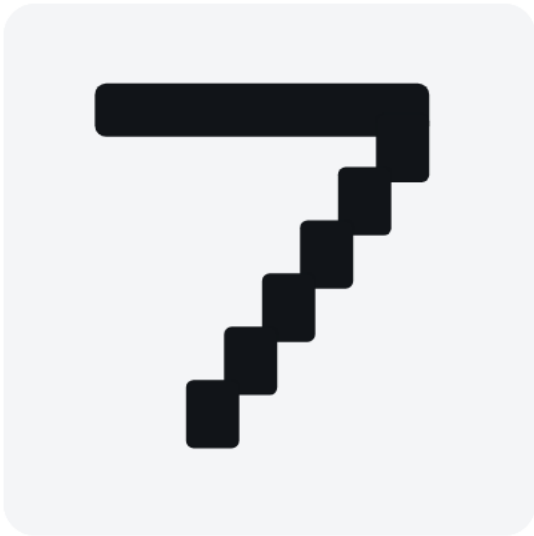
[Kaggle cats and dogs](#)



[MNIST](#)



Example 1: one MNIST image



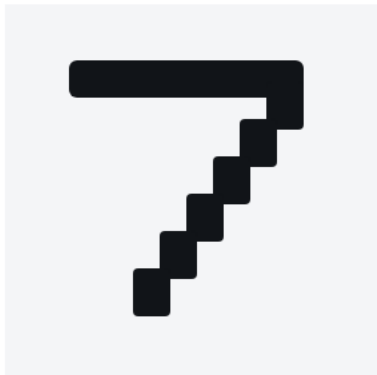
One training example is a **28 × 28** image: 784 grayscale pixel values.

The label is a single digit. MNIST has **10** classes, the digits 0 through 9.

[MNIST](#)

Example 1: input and output

INPUT



784 pixel values

ML

OUTPUT

0	1	2	3	4
5	6	7	8	9

10 class scores

Example 2: Predicting housing prices

- **Problem:** predict a house price from its features
- **Feature space:** rooms, location, area, condition, and similar descriptors
- **Label:** a real-valued price

[Kaggle House Prices competition](#)



Example 2: the data fields

79 data fields in, one price out

MSSubClass · MSZoning · LotFrontage · LotArea · Street · Alley · LotShape · LandContour · Utilities · LotConfig · LandSlope · Neighborhood · Condition1 · Condition2 · BldgType · HouseStyle · OverallQual · OverallCond · YearBuilt · YearRemodAdd · RoofStyle · RoofMatl · Exterior1st · Exterior2nd · MasVnrType · MasVnrArea · ExterQual · ExterCond · Foundation · BsmtQual · BsmtCond · BsmtExposure · BsmtFinType1 · BsmtFinSF1 · BsmtFinType2 · BsmtFinSF2 · BsmtUnfSF · TotalBsmtSF · Heating · HeatingQC · CentralAir · Electrical · 1stFlrSF · 2ndFlrSF · LowQualFinSF · GrLivArea · BsmtFullBath · BsmtHalfBath · FullBath · HalfBath · BedroomAbvGr · KitchenAbvGr · KitchenQual · TotRmsAbvGrd · Functional · Fireplaces · FireplaceQu · GarageType · GarageYrBlt · GarageFinish · GarageCars · GarageArea · GarageQual · GarageCond · PavedDrive · WoodDeckSF · OpenPorchSF · EnclosedPorch · 3SsnPorch · ScreenPorch · PoolArea · PoolQC · Fence · MiscFeature · MiscVal · MoSold · YrSold · SaleType · SaleCondition

[Id](#) is only an index. [SalePrice](#) is the label, not a feature. [Competition data](#)

Example 2: input and output

INPUT · 79 FIELDS

LotArea 8450

lot size, square feet

OverallQual 7

material and finish, 1–10

YearBuilt 2003

year of construction

GrLivArea 1710

above-ground living area, square feet

GarageCars 2

garage capacity, in cars

and 74 further fields

ML

OUTPUT

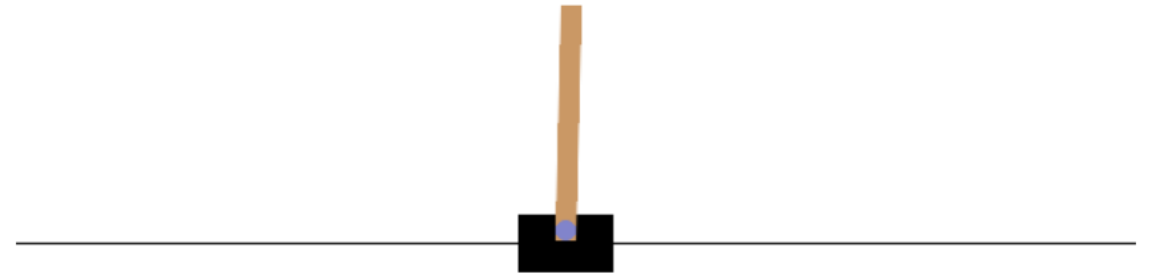
\$208,500

SalePrice

Example 3: Balancing a pole

- **Problem:** keep a pole upright by moving a cart left or right
- **Feature space:** cart position and velocity, pole angle and angular velocity
- **Label:** push left or push right
- **Setting:** reinforcement learning

[Gymnasium Cart Pole](#)



Example 3: input and output

INPUT

Cart position	0.036
Cart velocity	-0.089
Pole angle	-0.056
Pole angular velocity	-0.063

ML

OUTPUT

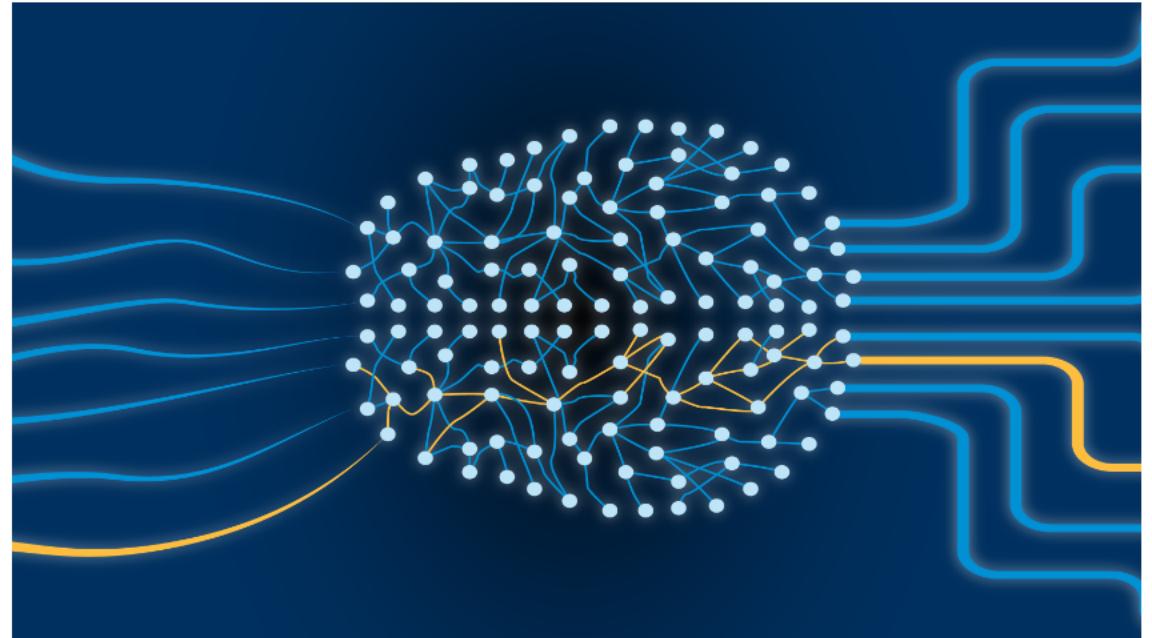
0 · push left

1 · push right

Learning the function

- A mathematical algorithm adjusts f so that $f(x)$ predicts y well on the data
- Learning resembles adaptation in a biological brain: many parameters change, even when their individual meaning remains obscure

$$f(x) = y$$



Loss function

or score function



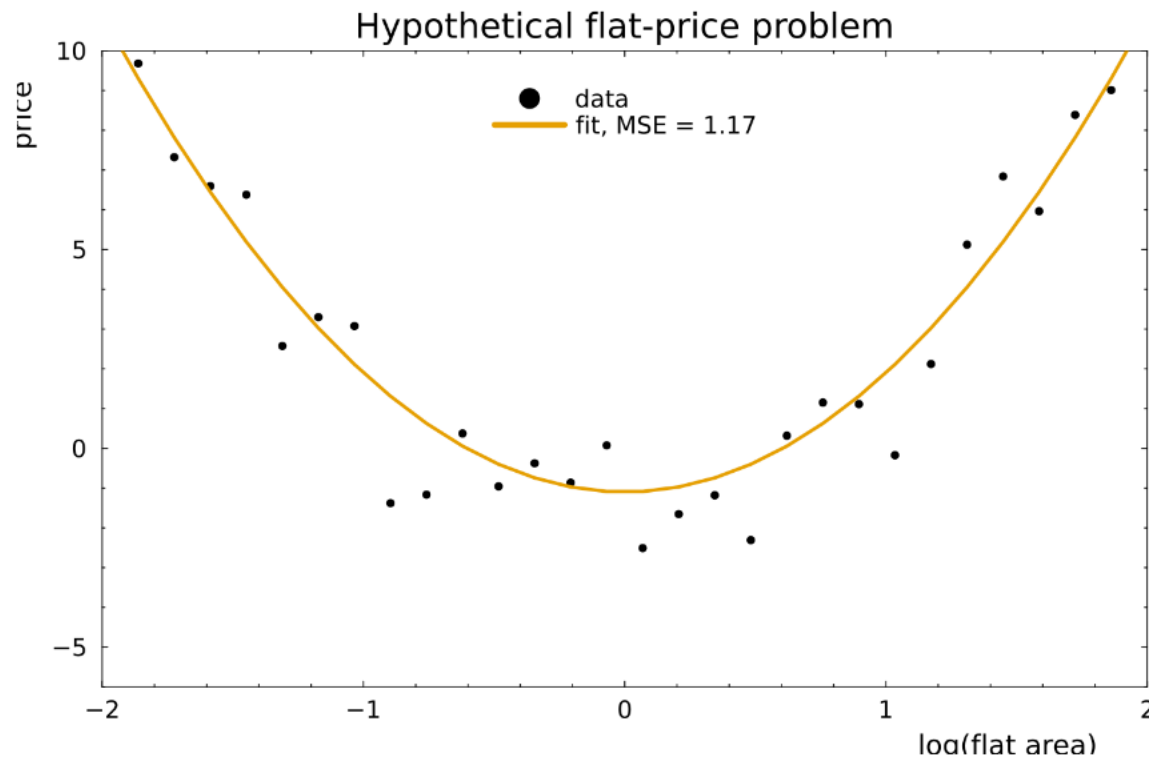
Loss function

or score function



Loss: Mean squared error

Hypothetical flat prices



$$f(x) = 3x^2 - a, \quad a = 1.1$$

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2 = 1.17$$

Thirty noisy prices. The curve is one setting of a ; training moves a to make this loss small.

Reinforcement learning loss

General policy-gradient loss

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^N \sum_{t=0}^{T_i-1} G_{i,t} \log \pi_{\theta}(a_{i,t} | s_{i,t})$$

$$G_{i,t} = \sum_{k=t}^{T_i-1} \gamma^{k-t} r_{i,k}$$

L loss minimized in training

N episodes; i episode index

t current frame; k future frame

$r_{i,k}$ reward; γ discount factor

$\pi_{\theta}(a | s)$ action probability from the network

θ neural-network parameters

T_i number of frames in episode i

$s_{i,t}$ state; $a_{i,t}$ chosen action

$G_{i,t}$ future reward after the action

CartPole: the pole stays up for 120 frames

Every frame gives $r_t = 1$. Choose $\gamma = 1$.

$$G_t = \sum_{k=t}^{119} 1 = 120 - t$$

Let $p_t = P(\text{right} | s_t)$ and define the probability of the action taken:

$$q_t = \begin{cases} p_t, & a_t = \text{right}, \\ 1 - p_t, & a_t = \text{left}. \end{cases}$$

$$L_{\text{episode}} = - \sum_{t=0}^{119} (120 - t) \log q_t$$

Minimizing this loss makes actions from the observed 120-frame episode more probable.

Loss: Binary cross entropy

Standard for binary classification

- Mean squared error can classify, but binary cross entropy better matches a probabilistic classifier
- Interpret $p(x) = f(x)$ as the probability of class $y = 1$
- The probability of class $y = 0$ is $1 - p(x)$

$$\text{BCE} = - \sum_i \left[y_i \log p(x_i) + (1 - y_i) \log (1 - p(x_i)) \right]$$

The loss grows rapidly when the model assigns low probability to the observed class.

Shannon entropy

How broad, and how uncertain, a distribution is

$$S = - \sum_i q_i \log q_i$$

$$S = - \int q(x) \log q(x) dx$$

Monte Carlo, with x_i drawn from q :

$$S \approx -\frac{1}{n} \sum_i \log q(x_i)$$

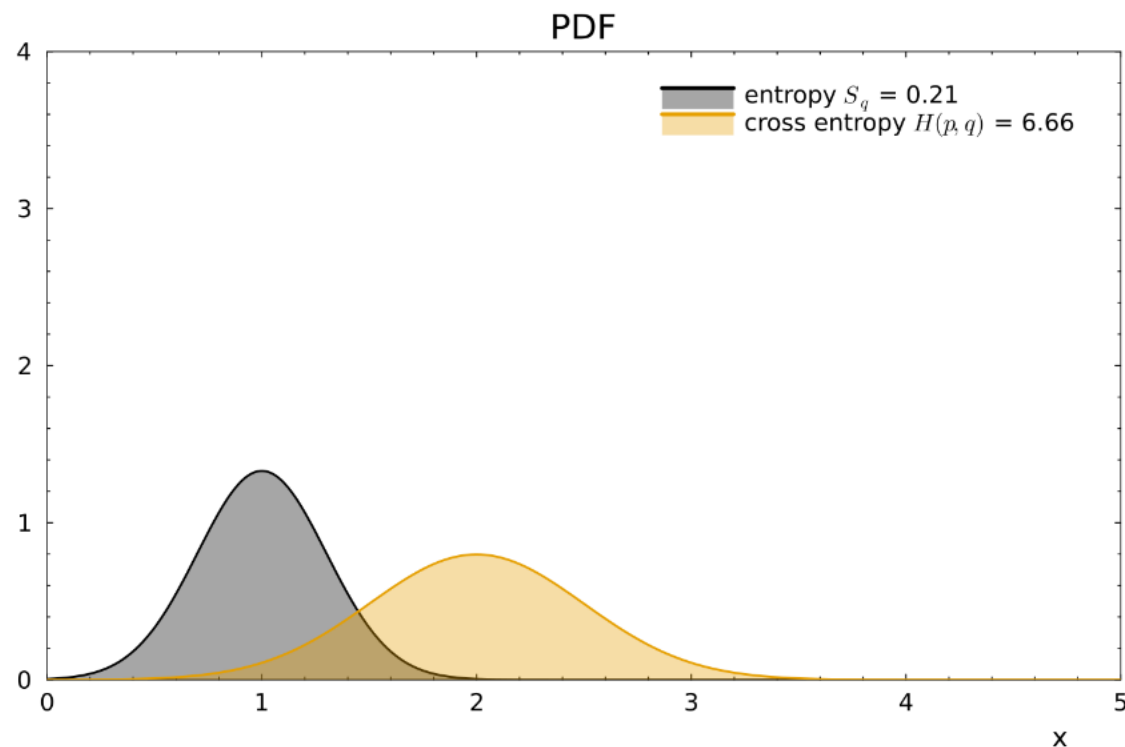
The narrow density has $S_q = 0.21$. A broader density is more uncertain, so its entropy is larger.

Cross entropy of two distributions

Likelihood of q , evaluated on a sample from p

$$H(p, q) = - \sum_i p_i \log q_i$$

- This is binary cross entropy for two densities
- It measures how different p and q are
- The minimum is $p = q$, and then $H(p, p)$ is the entropy of p



$q = \text{Normal}(1, 0.3)$, $p = \text{Normal}(2, 0.5)$

$H(p, q) = 6.66$

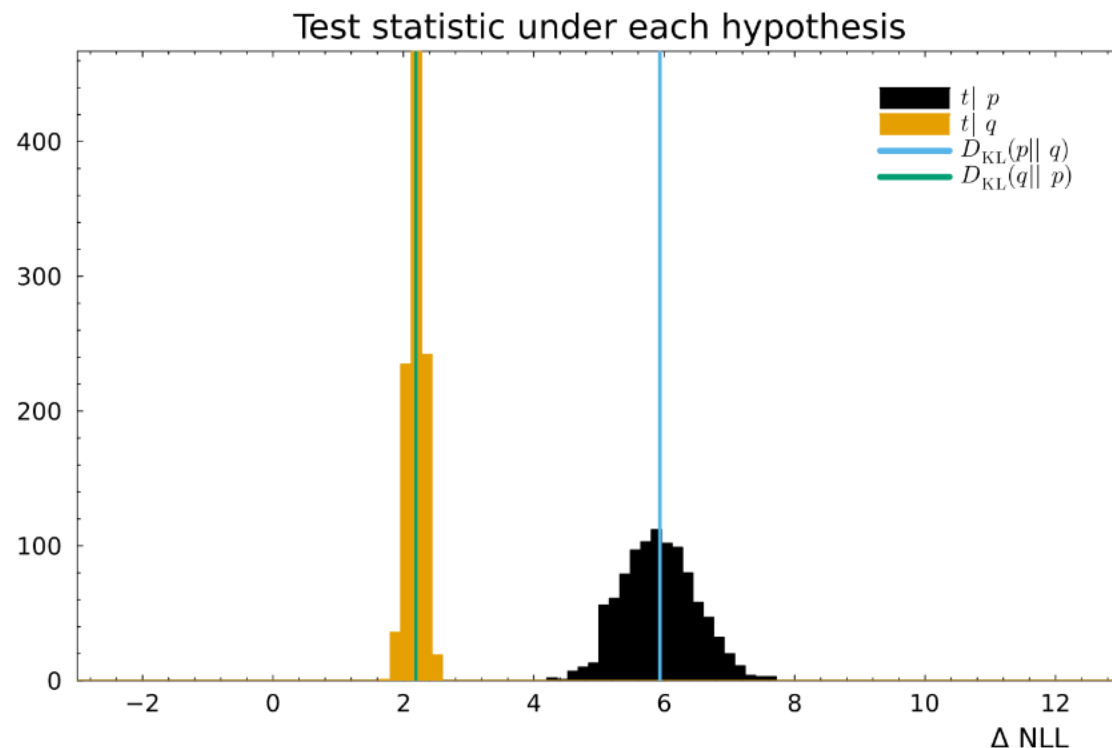
Relative entropy

Kullback–Leibler divergence

$$D_{\text{KL}}(p \parallel q) = H(p, q) - S_p$$

$$D_{\text{KL}}(p \parallel q) = -\sum_i p_i \log \frac{q_i}{p_i}$$

Expected likelihood difference when the truth is p and the model is q . The two directions are not equal.



$$D_{\text{KL}}(p \parallel q) = 5.93, D_{\text{KL}}(q \parallel p) = 2.19$$

Training ML system

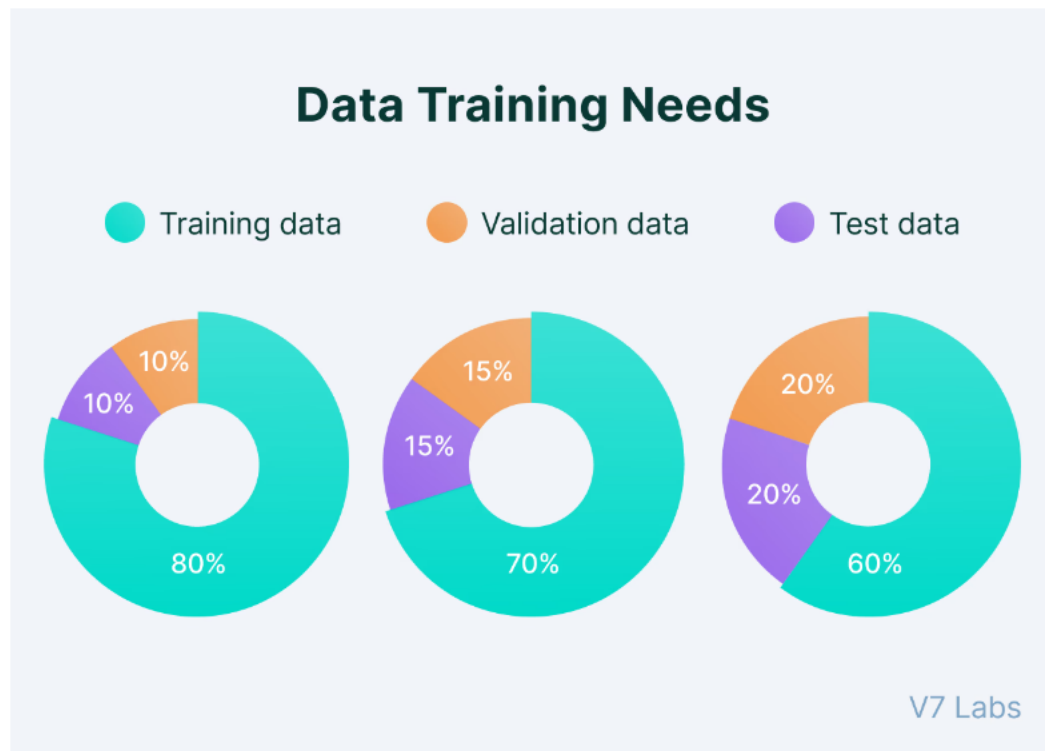
Training, validation, and test data

Minimal setup

- Train on the **training set**
- Evaluate once on the **test set**

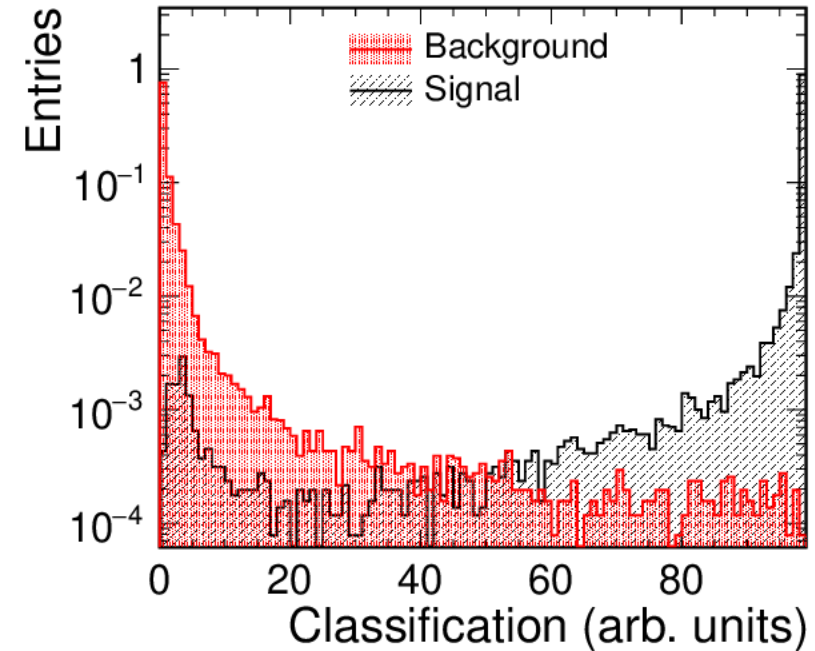
Comparing architectures

- Use a **validation set** for model and hyperparameter choices
- Keep the **test set** independent until the final evaluation



Classification decisions

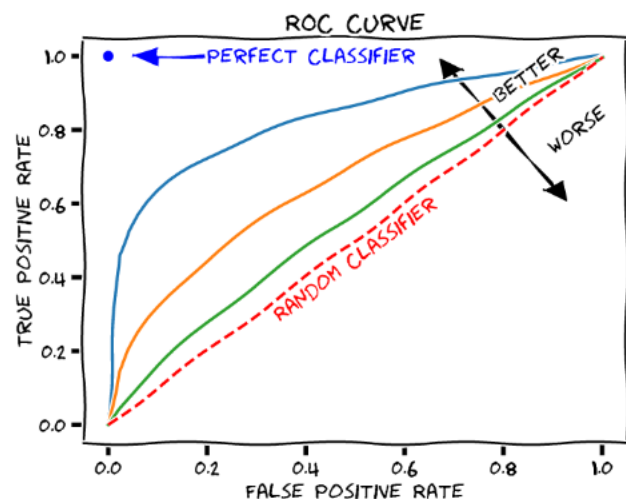
- A classifier usually returns a continuous score or probability
- A threshold converts the score into a class decision
- The error rate depends on the chosen threshold



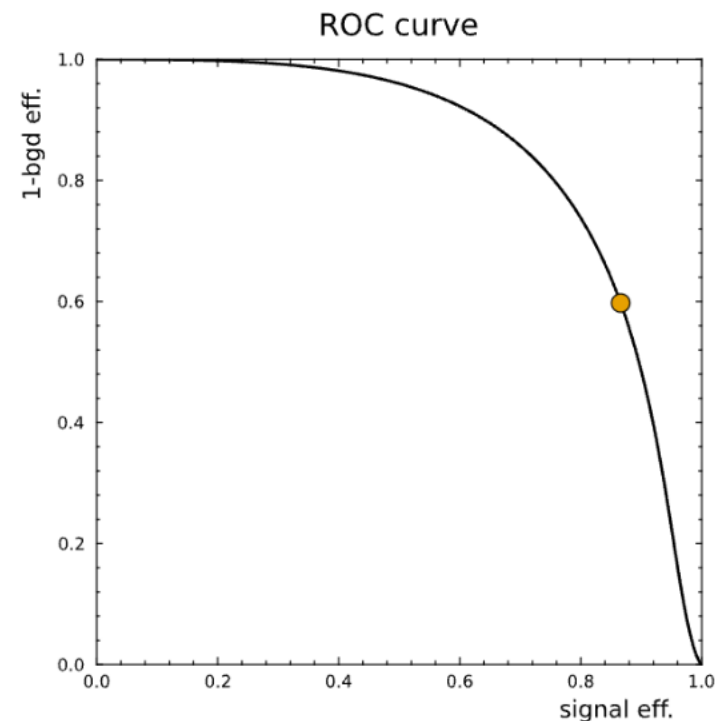
ROC curve

Receiver operating characteristic

- Sweep the classifier threshold
- Plot the true-positive rate against the false-positive rate
- Curves closer to the upper-left corner indicate stronger separation
- **AUC**, the area under the curve, summarizes the whole ROC: 1 is perfect, $\frac{1}{2}$ is chance



The same threshold sweep in different variables



Optimization and gradient descent

Training minimizes a loss $L(\theta)$ over model parameters θ .

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} L(\theta_t)$$

- **Learning rate** η : the size of each step along the gradient
- The gradient points toward the steepest local increase, so the update moves in the opposite direction

