Machine Learning – Fundamentals

Artificial Intelligence for Digital Health (AID)

M.Sc. in Digital Health – University of Pisa

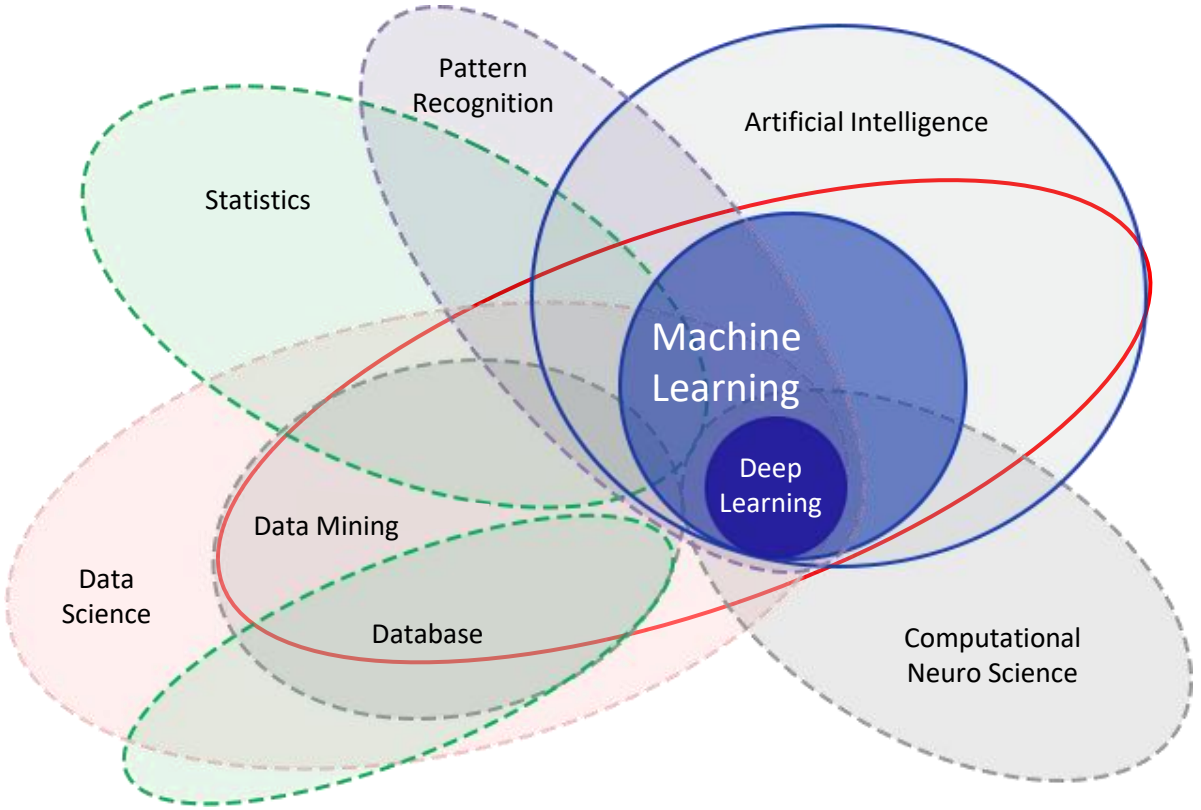
Davide Bacciu (davide.bacciu@unipi.it)

Lecture Outline

- Understand basic concepts of machine learning (ML)
- Differentiate between learning paradigms and fundamental ML tasks
- Discuss data types and their roles in ML
- Statistical Learning Theory: Learn about generalization, bias/variance tradeoff, and regularization
- Model Selection: How to evaluate a model and robustly assess its generalization

Introduction to Machine Learning

What is Machine Learning?



- Machine learning or Learning from Data
- Classical Computing: Data & Program → Results
- (Supervised) Machine Learning: Data & Results → Program
- In more formal terms, a program is some unknown function for which we can observe data (and corresponding results)
- Key Components:
  - Data
  - Model
  - Learning Algorithm

What is Learning?

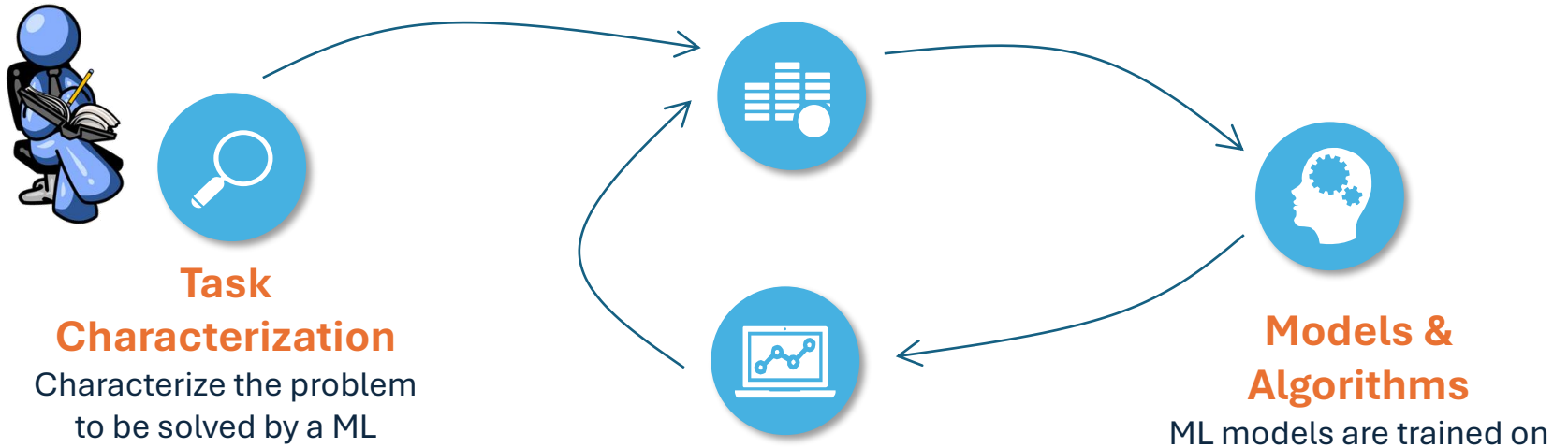
Tom Mitchell, 1997:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Three key components

- Task → a problem we would like to solve
- Experience → data to learn from
- Performance Measure → a measure of how well we learned

ML Lifecycle



Learning Data

A collection of relevant samples to train the ML model. Depending on task, should include ground truth predictions for example inputs.

Task Characterization

Characterize the problem to be solved by a ML model

- No «algorithmic» solution
- Excessive computational cost

Trained Model

A trained model can provide predictions and/or identify patterns in fresh data

Models & Algorithms

ML models are trained on available data using optimization algorithms that can modify predictions and knowledge captured by the model

Learning Paradigms

Supervised Learning

Learn an unknown function predicting an output in response to an input

- Diagnose disease given patient profile

(x, y)

Unsupervised Learning

Identification of structures, regularities associations, distribution and anomalies in the data

- Signaling anomalous physiological measurements

(x)

Reinforcement Learning

Learning of a policy or complex behaviour while being allowed to observe only partial responses from the interaction with the environment or the user

- Personalized treatment policy for a patient

(s, a, r)

And much… much more (semi-supervised, weakly supervised, continual, …)

Supervised Tasks: Classification and Regression

- **Definition**: Learning from labeled data to make predictions about categorical or continuous outcomes
- **Classification**:
  - Binary or multi-class.
  - Example: Predicting disease presence.
- **Regression**:
  - Predicting continuous variables.
  - Example: Forecasting patient survival times.

Unsupervised Tasks: Clustering

- **Definition**: Grouping similar data points
- **Common Algorithms**:
  - k-Means Clustering.
  - Hierarchical Clustering.
  - DBSCAN.
- **Applications**:
  - Identifying subtypes of cancer from genetic profiles (stratification)

Unsupervised Tasks: Dimensionality Reduction

- **Definition**: simplifying data representation (e.g., for visualizing high-dimensional data)
- **Common Algorithms**:
  - Principal Component Analysis (PCA).
  - t-SNE.
- **Applications**:
  - Genomics data analysis.
  - Visualizing patient features and their similarity

Data Vs Learning Models

The nature of data deeply influences the choice of the learning model and its efficacy

Type

Categorical

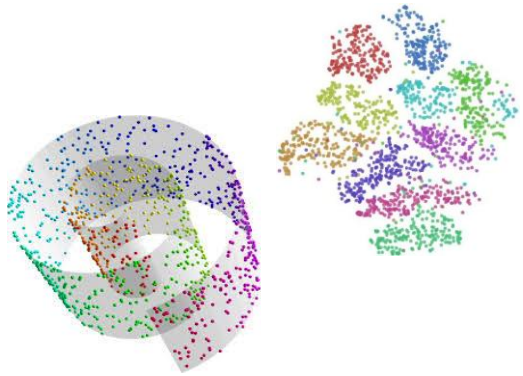
Ordinal

Continuous

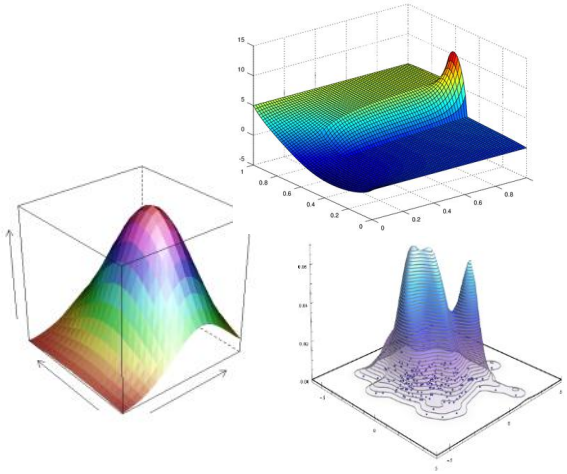
Mixed

Relational

Separability



Distribution



Which data for which tasks?

A general distinction:

- **Unstructured data**: e.g., tabular data. Features do not have particular relationships one another.
- **Structured data**: e.g., images, sequences. Features are related to one another (e.g., sequences have a temporal dimension).

The type of data influences the choice of the learning model:

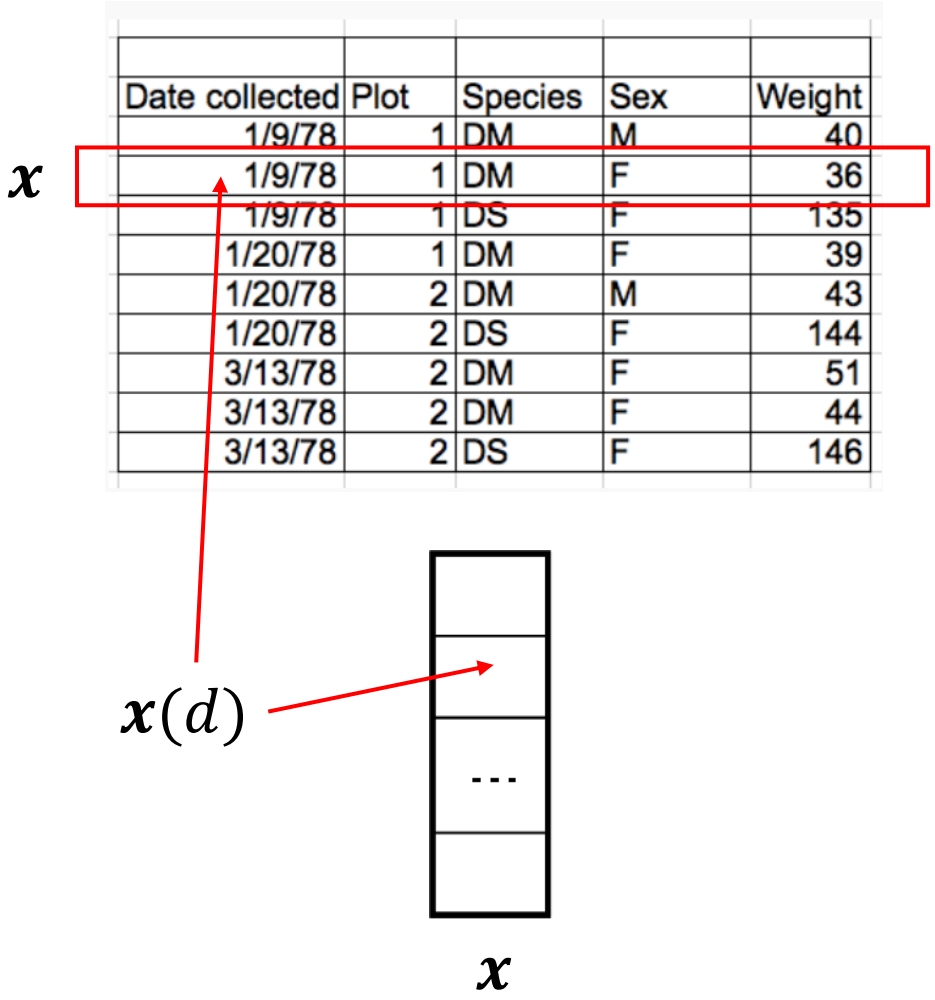
- E.g. convolutional networks for images.
- Data help us identify inductive biases for models

Inductive Bias
The incorporation of domain knowledge in model design (which influences its effectiveness on certain data types)

Data Types

- **Tabular Data**
  - Rows represent samples; columns represent features
  - Typically mix features of heterogenous type and scale
  - Example: Patient data with age, weight, and diagnosis.
- **Vectors**
  - Numerical arrays representing features
  - Apparently similar to tabular (but different from a fine-grained math perspective)
  - Example: spectral features of an EEG.
- **Images**
  - Pixel grids represented as matrices/tensors.
  - Example: X-ray images for disease diagnosis.
- **Sequential Data**
  - Ordered data with temporal or logical sequence.
  - Example: EEG signals, medical records across time
- **...and more**

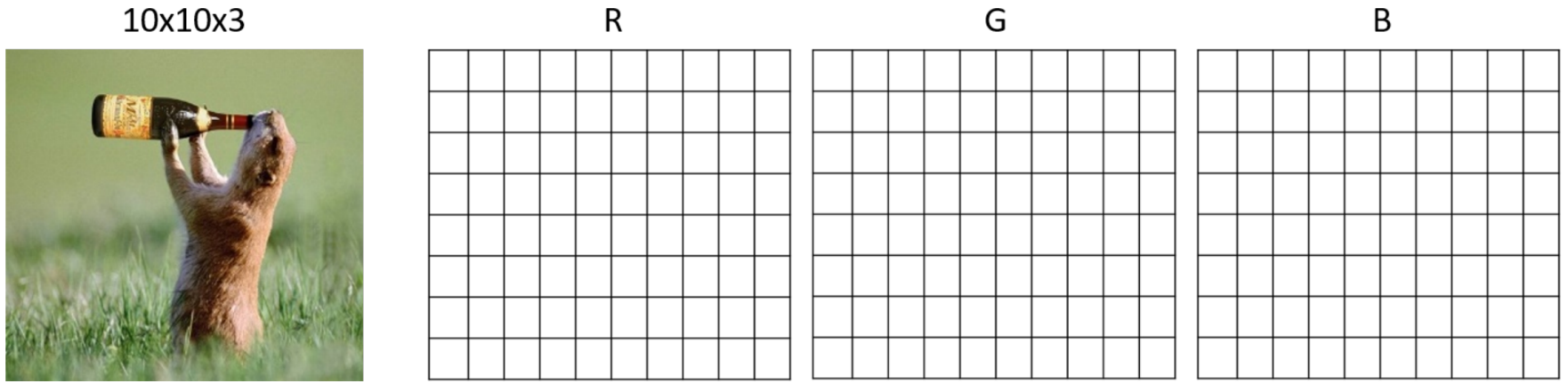
Vectorial/Tabular



- A *D*-dimensional numerical array
  - Continuous, categorical or mixed values
  - Describes an individual of our world of interest, e.g. patients in a biomedical application
- The single dimensions d are called features and numerically represent an attribute of the individual
  - E.g. if $\boldsymbol{x}$ describes a patient, $\boldsymbol{x}(d)$ (or x_d) can be his/her age

Images

Images are matrices of pixels intensity

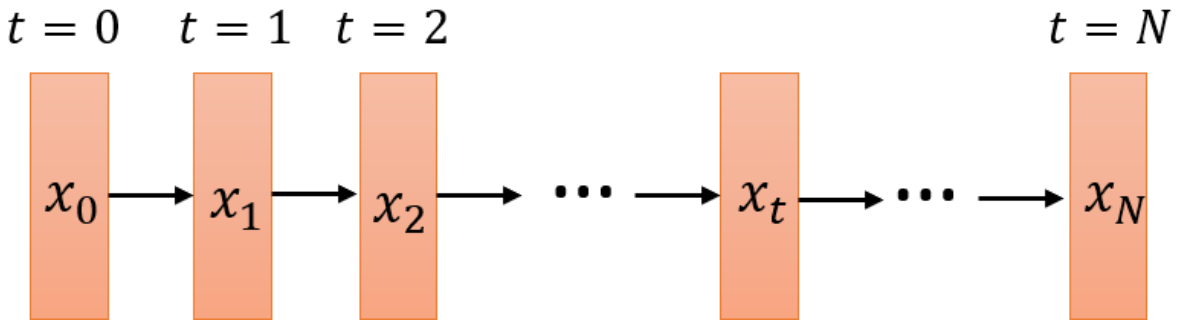


Each pixel has an associated (spatial) coordinate within the image

Inductive Bias
Nearby pixels tend to have the same color

Sequences

- Variable size data characterized by sequentially dependent information
- Each element of the sequence is (possibly) a vector (multivariate)
- Sequence elements can be sampled at irregular times
- In ML can be used both as input and output information



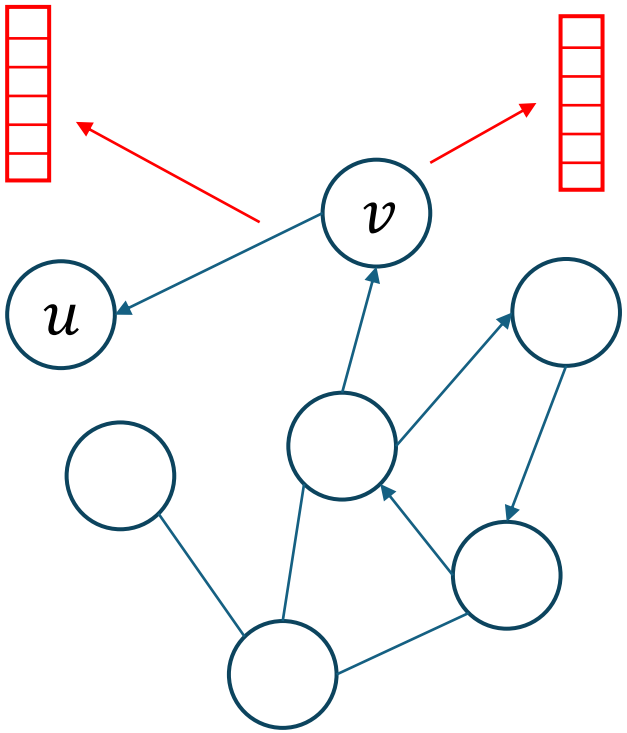
Inductive Bias

The element at time t in the sequence may depend only on its (more or less) recent past

Graphs

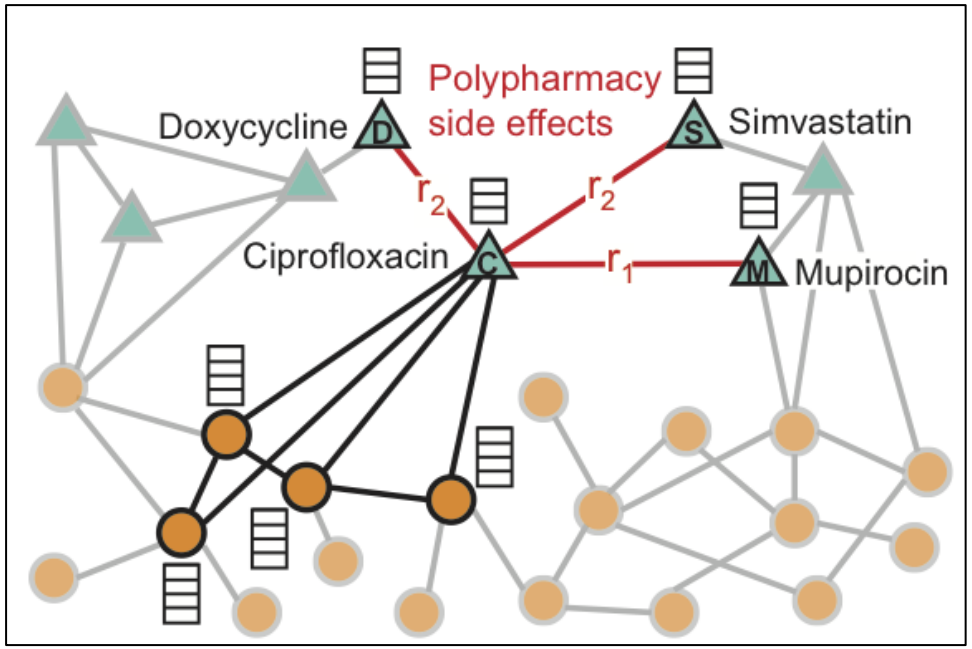
Allow to represent articulated relationships in compound data

Edge label x_{vu}



Node label

x_v



Basics of Statistical Learning Theory

Fundamental Definitions in ML

- **Sample/observation/example**: A single instance or observation $\boldsymbol{x}$

$$\text{E.g. a vector } \boldsymbol{x} = (x_1, \ldots, x_K)$$

- **Dataset**: A collection of samples $D = \{(\boldsymbol{x}^1, \boldsymbol{y}^1);\ \ldots\ (\boldsymbol{x}^N, \boldsymbol{y}^N)\}$
- **Features**: Attributes or variables describing each sample, e.g. x_k
- **Target**: Desired outcomes for supervised learning, e.g. $\boldsymbol{y}^n$

ML Models

- **ML model**: A parametric function $h_{\theta,\alpha}(\boldsymbol{x})$ that can be applied to data $\mathbf{x}$ and whose behavior is regulated by adaptive parameters θ (learned) and by hyperparameters α (externally set)
  - $h_{\theta,\alpha}$ defines a family of functions
  - Changing parameters (and hyperparameters) changes how the function behaves
- **Training**: Process through which the parameters θ of model $h_{\theta,\alpha}$ are modified to adapt to training dataset D by optimizing a cost/error function $E\left(h_{\theta,\alpha}\middle|D\right)$

Empirical Error (Supervised Learning)

Suppose we have a dataset of N samples

$$D = \{(\boldsymbol{x}^1, \boldsymbol{y}^1);\ \ldots\ (\boldsymbol{x}^N, \boldsymbol{y}^N)\}$$

The empirical (sample) error of model h_θ (omitting α for simplicity) with respect to the sample D is

$$E(h_\theta|D) = \frac{1}{N}\sum_{(\boldsymbol{x}^i,\boldsymbol{y}^i)} L(h_\theta(\boldsymbol{x}^i), \boldsymbol{y}^i)$$

where $L(h_\theta(\boldsymbol{x}^i), \boldsymbol{y}^i)$is the loss, i.e. a function measuring the discrepancy between the predicted $h_\theta(\boldsymbol{x}^i)$ and the target value$\boldsymbol{y}^i$

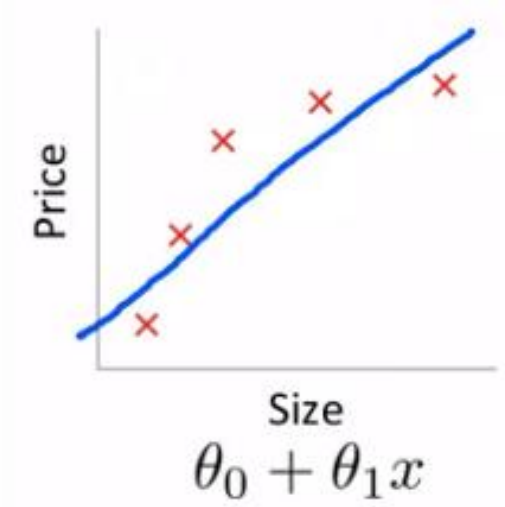
Learning is the process which identifies those parameters θ that minimize $E(h_\theta|D)$ ⇒ however, we would like to achieve something more…

Generalization

- Sought property of a model h_θ that, trained on D, generalizes well its output on new/fresh data D'
  - The goal of a ML model is to predict well on unseen data
  - Opposite of memorization
- Generalization states that the model can transfer its performance from a dataset (finite data) to new samples (infinite data)
- Statistical learning theory studies the conditions under which we can generalize starting from a finite sample
- How we can make sure that the empirical error $E(h_\theta|D)$ is a good estimator of the risk $R(h_\theta)$

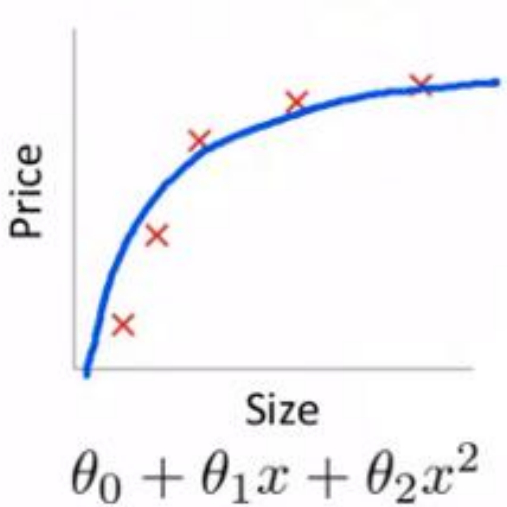
$$E(h_\theta|D) = \frac{1}{N}\sum_{(\boldsymbol{x}^i,\boldsymbol{y}^i)} L(h_\theta(\boldsymbol{x}^i), \boldsymbol{y}^i) \approx R(h_\theta) = \mathbb{E}_{x,y\sim \mathrm{P}}\left[L(\boldsymbol{x},\boldsymbol{y})\right] = \int L(h_\theta(\boldsymbol{x}),\boldsymbol{y})\, dP(\boldsymbol{x},\boldsymbol{y})$$

Empirical Risk & Model Complexity



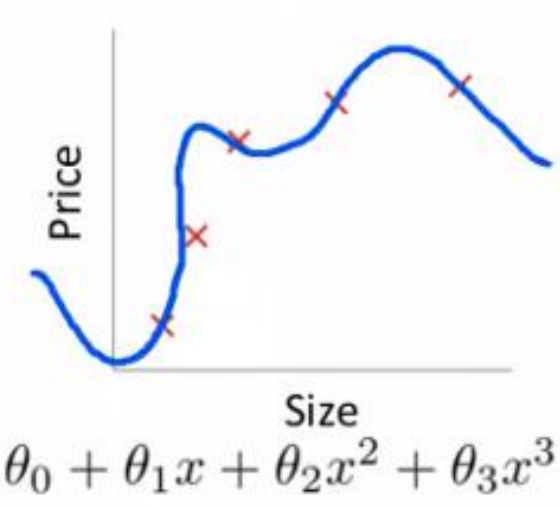
$\theta_0 + \theta_1 x$

Underfit: model is too simple and cannot learn the right function



$\theta_0 + \theta_1 x + \theta_2 x^2$

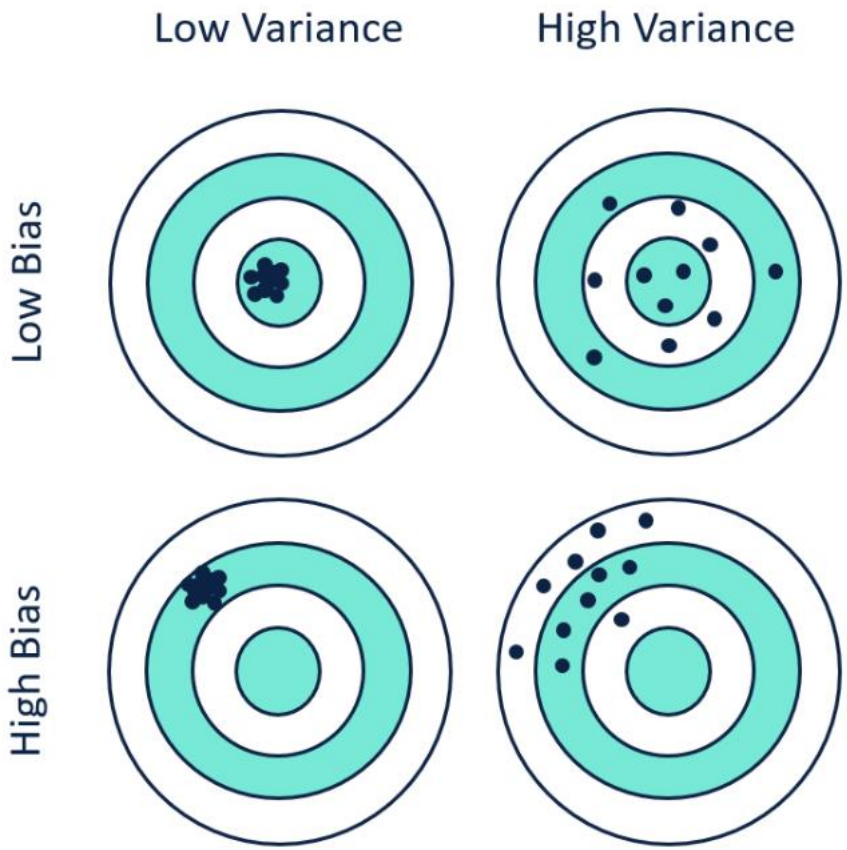
A balanced model complexity gives us generalization guarantees



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Overfit: model is too complex; it is memorizing the dataset but not generalizing

Bias-Variance



Bias: how close is the learned function from the target one?

Error from overly simplistic models.

Example: Linear regression on non-linear data.

Variance: How much dataset-dependent is my learned function?

Error from overly complex models capturing noise.

Example: Overfitting high-order polynomials.

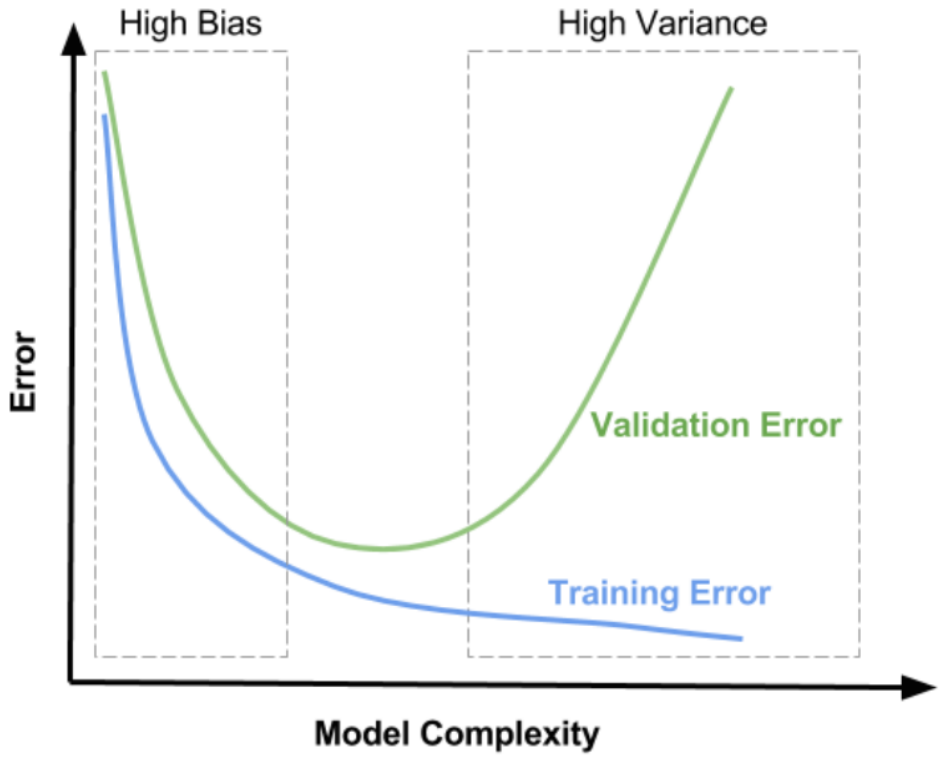
Trade-off

Balancing bias and variance to achieve optimal performance.

Need tools to measure trade-off and control model complexity

Measuring the trade-off

How do we measure overfitting/underfitting?



Given a finite dataset D we need to create (at least) 3 partitions

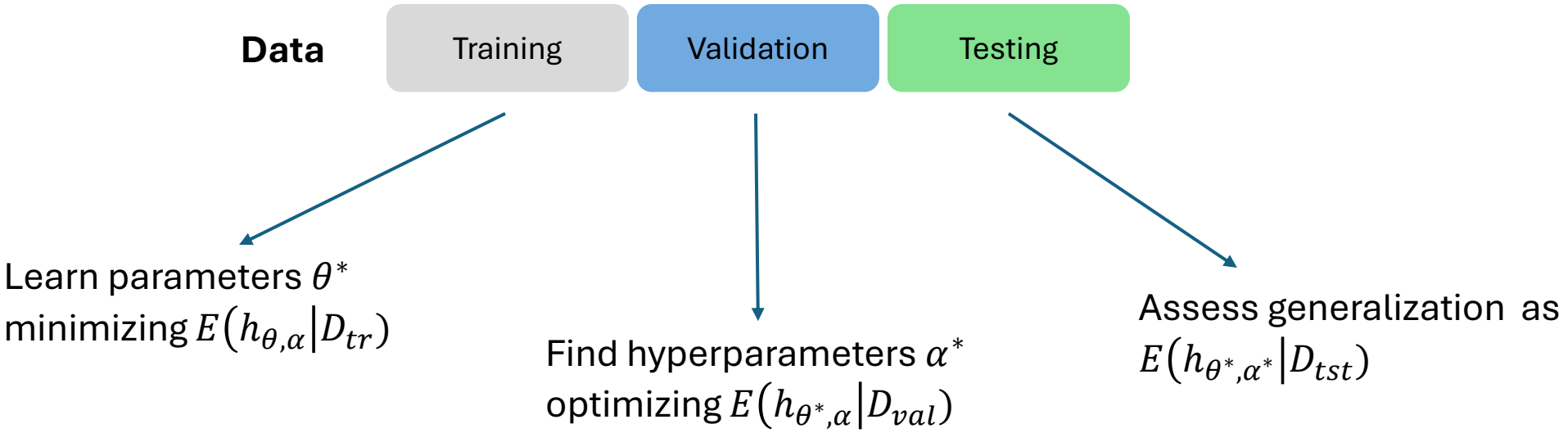
- Training set D_{tr} is used to learn model parameters
- Validation set D_{val} is used to measure overfit/underfit and take decisions on the model (hyperparameters α)
- Test set D_{tst} is used to reliably estimate $R(h_{\theta;\alpha})$(generalization)

Model Selection and Evaluation

Model Selection

Set of techniques from robust statistics to measure generalization, avoid overfitting and reduce the effect of model bias

Key catch: separate training phase, from the choice of model configuration (including hyperparameters α), from model generalization assessment



Parameters Vs Hyperparameters

- **Parameters θ**
  - Learned automatically from data through training
  - Contain model knowledge (data patterns)
  - Example: coefficients of linear regression; weights of a neural network
- **Hyperparameters α**
  - Can be set manually by the practitioner (can also be automated)
  - Tuned in model selection to optimize model generalization
  - Example: k in kNN; learning rates in neural networks; depth in decision trees

Model selection and inductive bias

- Hyperparameters are not the only aspect tuned during model selection
- Preprocessing and architectural design choices of an ML model influences deeply
  - The type of tasks it can solve
  - The type of data it can handle
  - The quality of generalization of its results
- Design choices (for model selection)
  - Data normalization (preprocessing more in general)
  - Neural network topology
  - Probability distributions
  - Activation functions
  - Distance metrics
  - Regularization strategies
  - Loss functions
  - ....

Inductive bias

Regularization

- Techniques to prevent overfitting by adding constraints to the model
- Implemented through mechanisms to control model complexity

A typical scheme entails modifying the training error

$$E\left(h_{\theta,\alpha}\middle|D_{tr}\right) = \frac{1}{N}\sum_{(\boldsymbol{x}^i,\boldsymbol{y}^i)} L(h_\theta(\boldsymbol{x}^i),\boldsymbol{y}^i) + \lambda P(h_{\theta,\alpha})$$

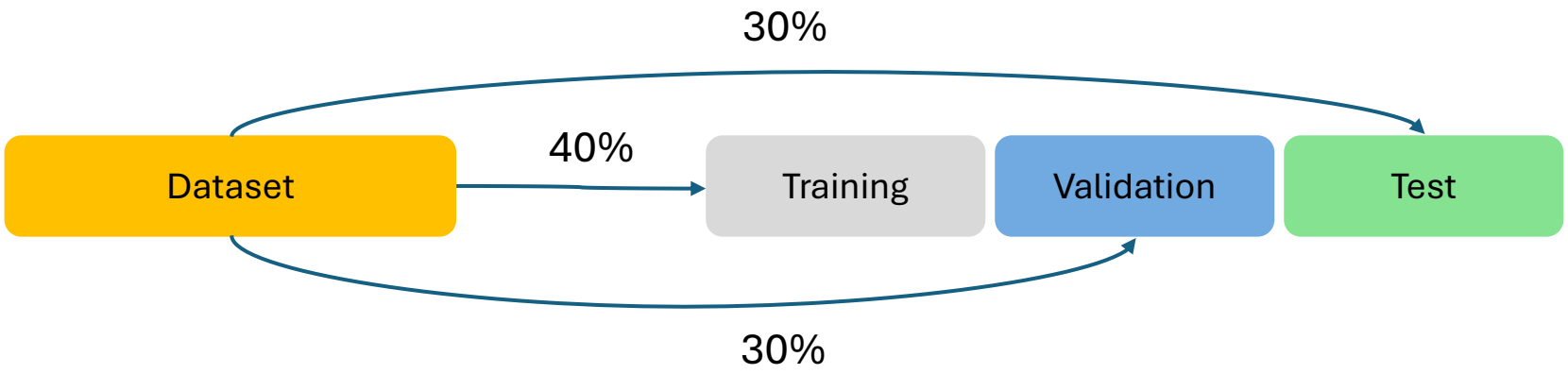
- $P(h_{\theta,\alpha})$ is a term penalizing if $h_{\theta,\alpha}$ is too complex
  - E.g.: using too many parameters θ
  - We will see example using norms of the parameter vectors as penalty functions
- λ is an hyperparameter ($\lambda \in \alpha$) regulating the trade-off between training error and penalization

Managing Model Selection - Holdout Splits

The preliminary action for model selection requires generating the dataset partition into training, validation and test

The holdout approach creates a static partition by selecting at random (without replacement!!!) which samples end-up in each of the three bins

Magic proportions used often: 40/30/30, 50/25/25, 60/20/20, 80/10/10



A word of caution about random splitting

The selection of samples cannot be fully random: it needs to preserve in the three partitions the same distributional properties of the original dataset!

- Proportion of samples from each class in classification tasks
- Distribution of relevant input features, such as patient gender or age

The solution typical amounts to do stratification or stratified sampling

1. Divide data in groups according to the variable w.r.t. which you want to stratify (e.g. the predicted class)
2. Sample at random training, validation and test samples from each of the group above according to the magic proportions

Managing Model Selection – Grid Hyperparameter Search

Assume you have two sets of hyperparameters and their candidate values

$$\alpha_1 = [0.1, 0.2, 0.3], \ \alpha_2 = [0.01, 0.001, 0.0001]$$

Given a (well designed) split in D_{tr}, D_{val}, D_{tst}

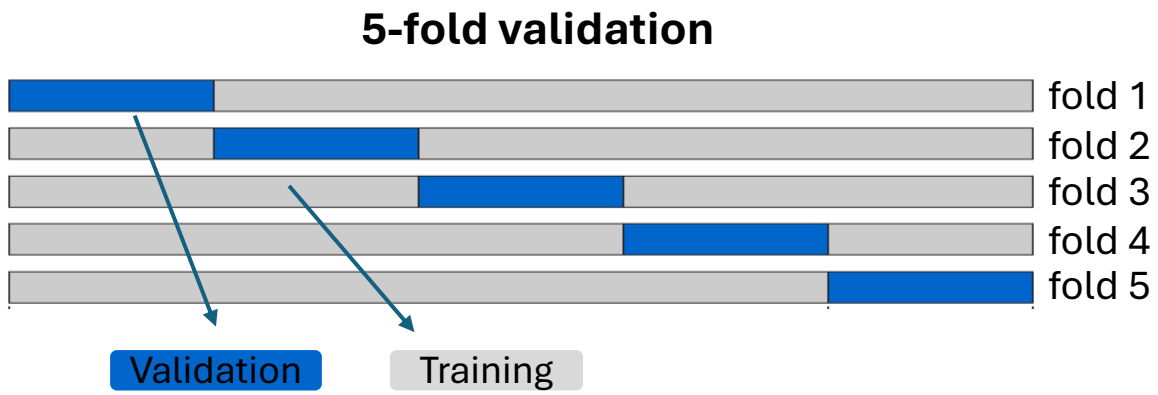
1. Create a grid of hyperparameter to search, with as many elements as the combinations of α_1 and α_2 values: e.g. $\alpha_1 = 0.1, \alpha_2 = 0.01$, then $\alpha_1 = 0.1, \alpha_2 = 0.001$, … then $\alpha_1 = 0.3, \alpha_2 = 0.0001$
2. Instantiate as many models as elements of the grid above
3. Train each model instance on D_{tr} and assess its performance on D_{val}
4. Select the best performing model (combination of hyperparameters) on D_{val}
5. Retrain the best performing model on $D_{tr} \cup D_{val}$ and test generalization performance on D_{tst}

k-fold cross-validation

Resampling procedure obtain more robust estimates of performance in ML models while remaining sample effective

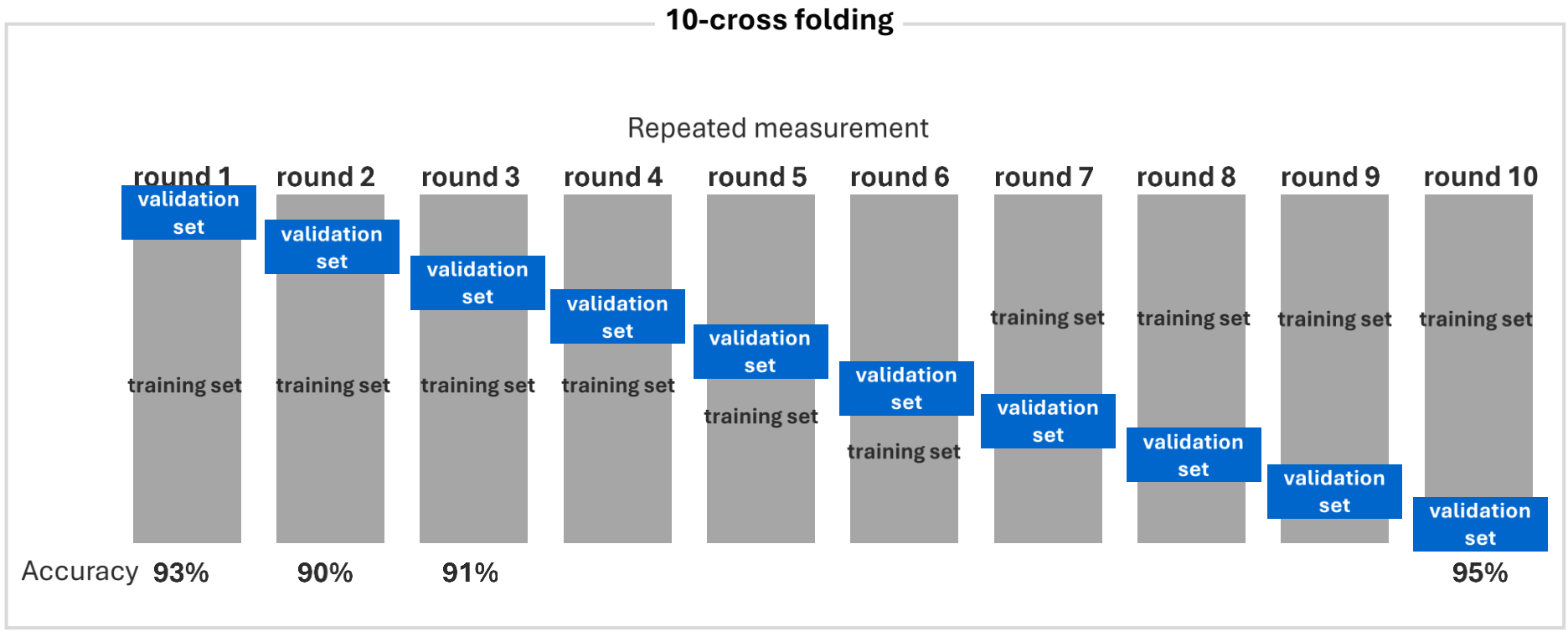
Reduces variability compared to a holdout train-validation-test split

1. Split the dataset into k folds.
2. Train the model on gray folds.
3. Validate on the blue fold.
4. Repeat for each fold and average the results.



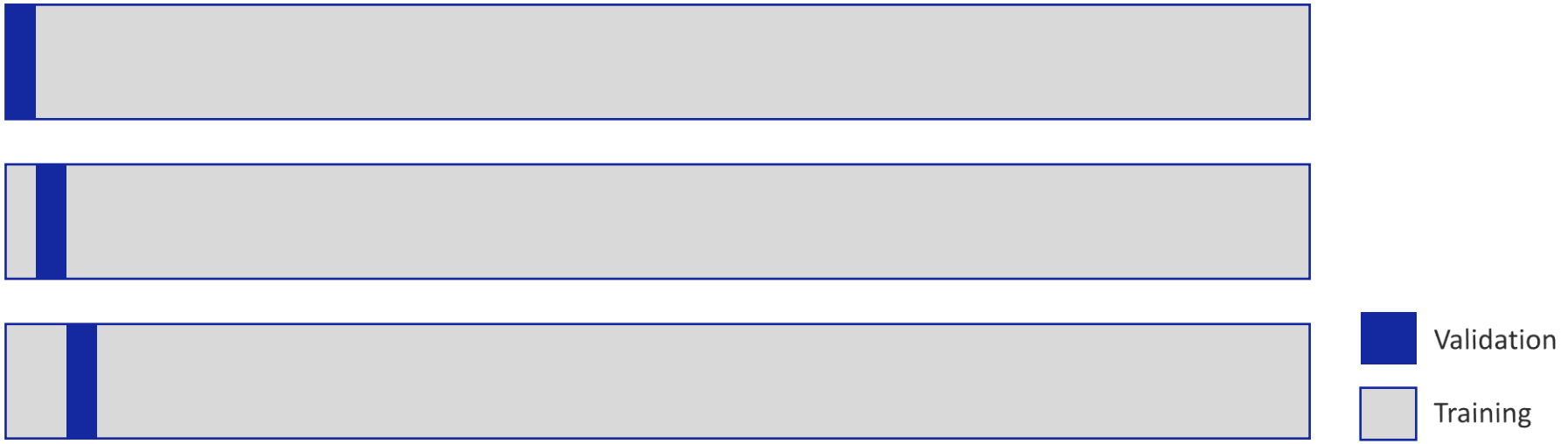
Test best model on hold-out → Test

Measuring Validation Performance with K-Fold



Final average accuracy (Round1, Round2... Round10)

Leave-One-Out Cross-Validation



- Leave only one observation for validation each time
- Maximum use of data (useful with small sample sizes)
- Worse use of time

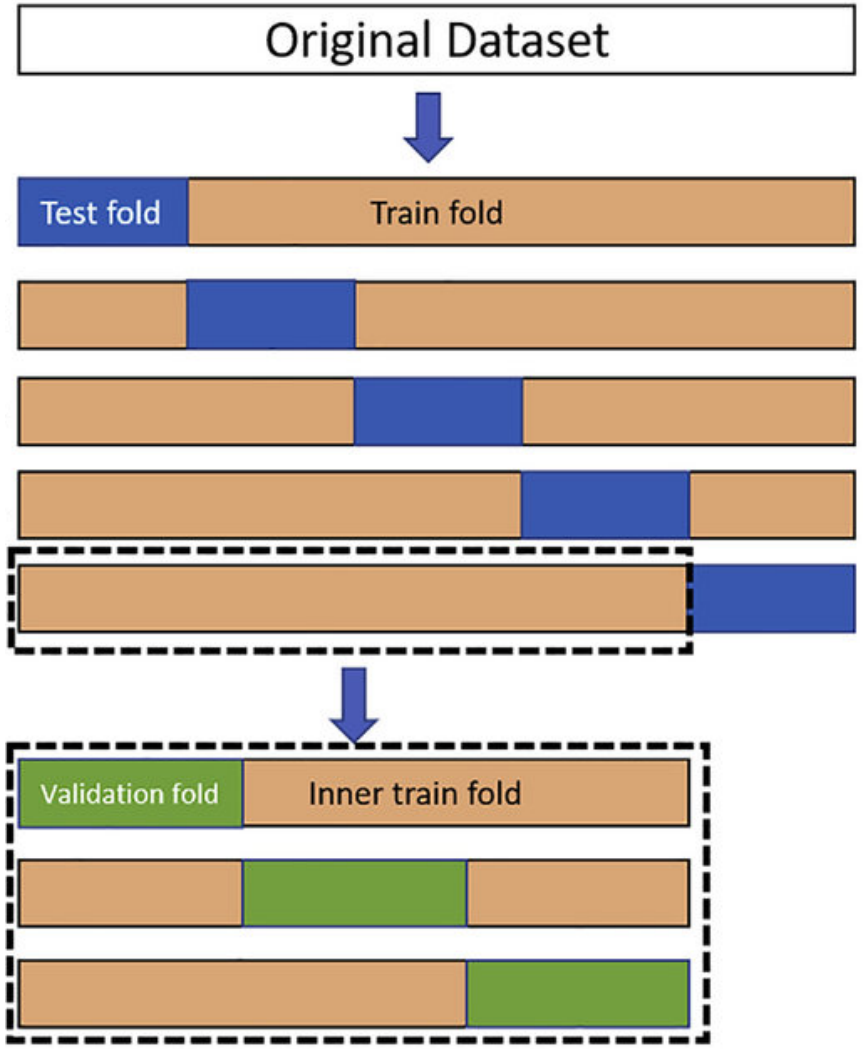
Nested Cross-Validation

A two-layered cross-validation method

- Outer loop: Splits the dataset for model evaluation.
- Inner loop: Performs -fold cross-validation for hyperparameter tuning.

Provides robust estimates of model generalization performance

- Average test performance and its dispersion



Measuring predictive performance

- Evaluating how well a model is doing (on training, validation or test) requires the definition of a performance metric
- The performance metric can be (and generally is) different from the model loss function
  - Loss function: serves to define the optimization problem that guides learning of the model parameters
  - Performance function: measure how adequate a model is for the specific application in which it will be used
- Performance metrics need to be tailored to the specific application and to the nature of the finite dataset
  - Example: highly imbalanced classes

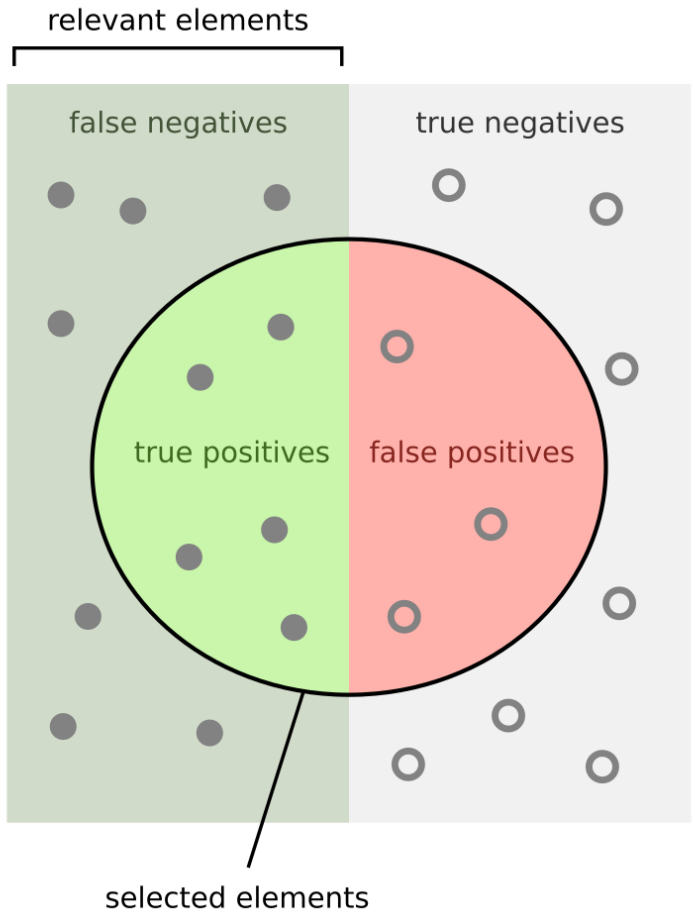
Some Basic Metrics

Focus on classification tasks (regression is a bit less interesting on this subject)

If we have a binary classification problem (true false)

- True Positives (TP): Correctly predicted positive cases.
- True Negatives (TN): Correctly predicted negative cases.
- False Positives (FP): Incorrectly predicted as positive.
- False Negatives (FN): Incorrectly predicted as negative.

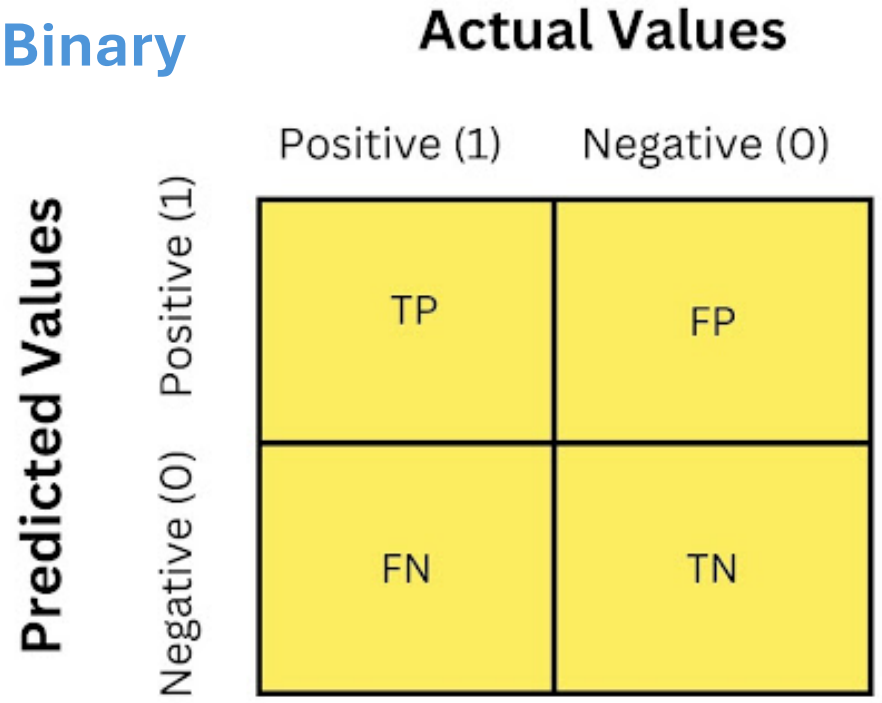
Img @ Wikipedia



Confusion Matrix

Provides insights into model performance across classes.

Binary



Multiclass

Precision and Recall

Precision is the proportion of true positive predictions among all positive predictions

$$precision = \frac{TP}{(TP + FP)}$$

Recall (sensitivity or TP Rate (TPR)) is the proportion of true positive cases detected among all actual positives

$$recall = \frac{TP}{(TP + FN)}$$

These metrics are crucial for imbalanced datasets

Example: Disease screening

High recall predictor:

- Ensures most diseased patients are identified.
- Minimizes false negatives but may increase false positives.

High precision predictor:

- Reduces false positives, ensuring accurate diagnosis.
- Important for reducing unnecessary treatments or tests.

F1 Metric

The harmonic mean of Precision and Recall balancing them into a single score

$$F1 = 2 * \frac{(precision * recall)}{(precision + recall)}$$

Useful when there is an uneven class distribution or when both Precision and Recall are important.

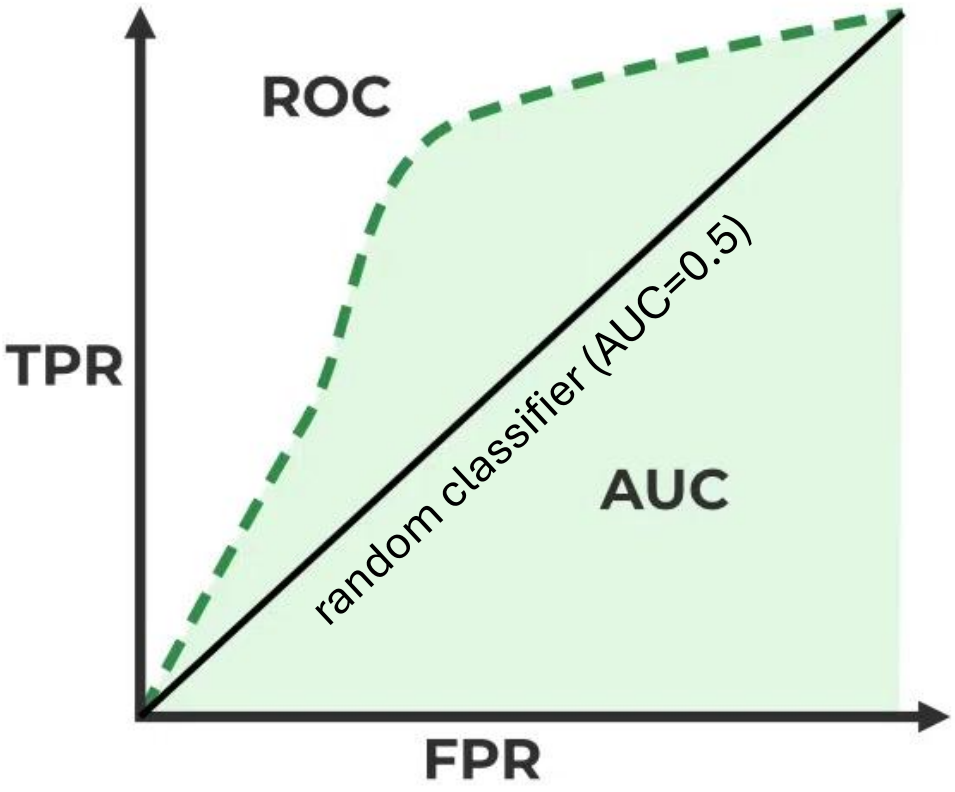
Specificity

- Specificity (or TN Rate (TNR)) is the proportion of true negative cases detected among all actual negatives

$$specificity = \frac{TN}{(TN + FP)}$$

- A complement to recall (sensitivity)

AUC-ROC Curve



AUC-ROC evaluates binary classifiers performance based on classification thresholds

- Receiver Operating Characteristic (ROC): Curve plotting TPR (Recall) vs. FPR
- Area Under Curve (AUC): Numerical representation of ROC quality
- AUC close to 1 indicates excellent performance.
- Useful for comparing multiple models.

Wrap-up

Take home lessons

- The nature of data and of the learning tasks guides the design choices of the learning model (Inductive bias)
- The goal of an ML model is to generalize well on new data, using a finite dataset in an optimal manner to estimate the theorical risk (error on infinite data)
- Model complexity needs to be controlled to avoid overfitting and achieve generalization (regularization)
- Model selection provides us with a toolset to obtain robust estimates from the empirical risk, identifying optimal model hyperparameters and design choices (grid search, holdout, k-fold cross validation, nested CV)
- Choosing model performance metric requires insight into the final application and the nature of the dataset
  - Confusion matrix: detailed class-wise analysis.
  - Precision, recall and F1: imbalanced datasets.
  - AUC-ROC: holistic view of classification thresholds.

Next Lecture

- Tomorrow: lab time!
- Next lecture (Thursday)
  - Linear models in machine learning
  - Linear regression models
  - Logistic regression models
  - Learning by least square minimization and gradient descent