UNIVERSIDADE CATÓLICA PORTUGUESA

BRAGA

Deep Learning

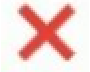
Session 7

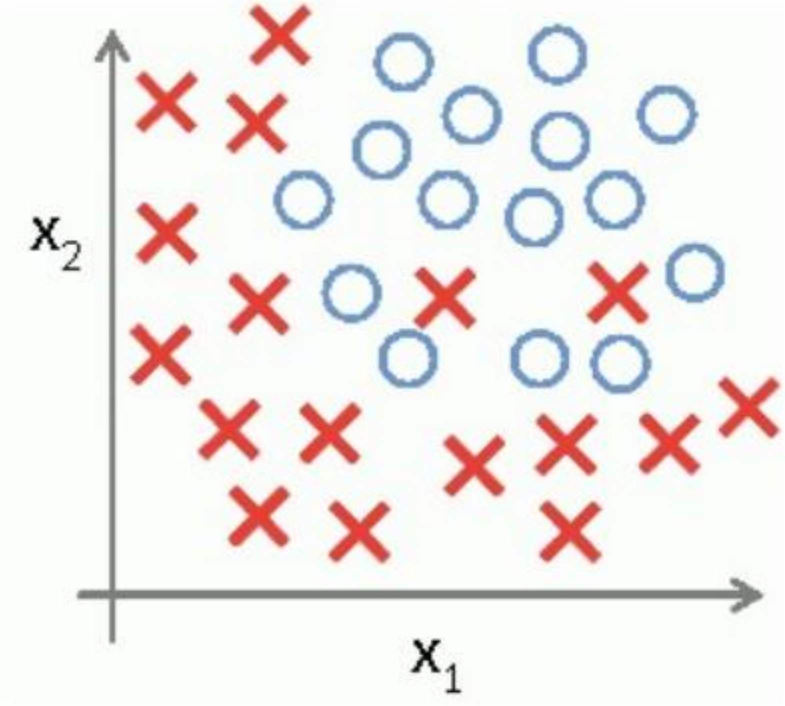
Model Selection and Hyperparameter Tuning

Applied Data Science

2024/2025

Exercise

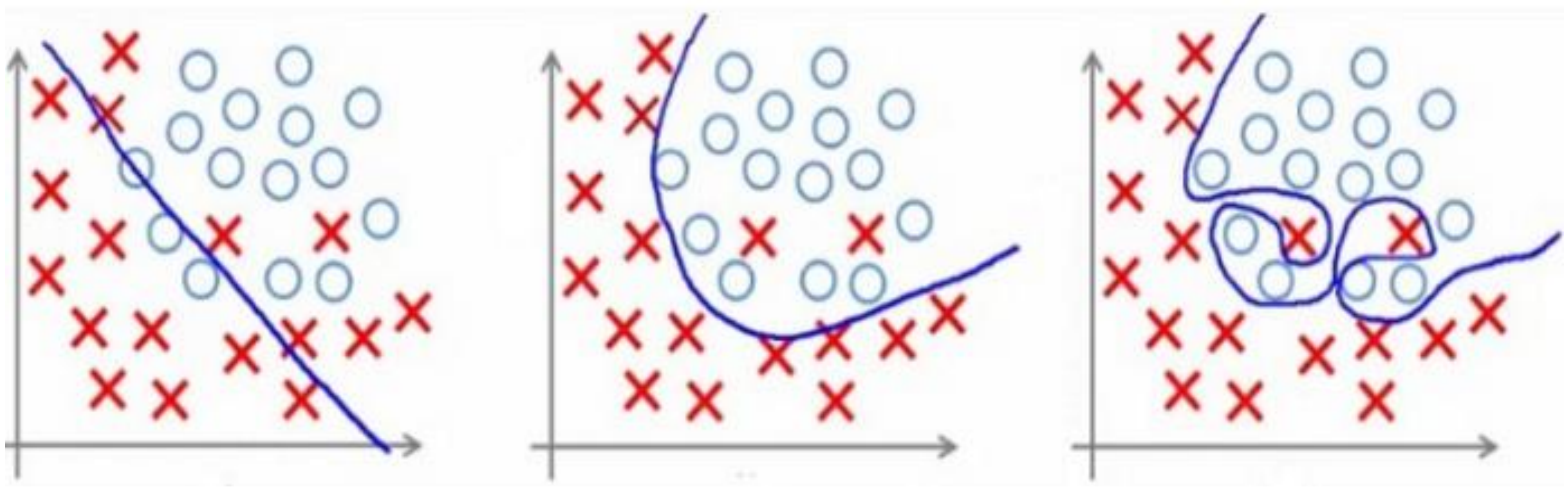
- Separate  from 



1. Using a straight line (linear function)
2. Using a parabola (quadratic function)
3. Using any curve

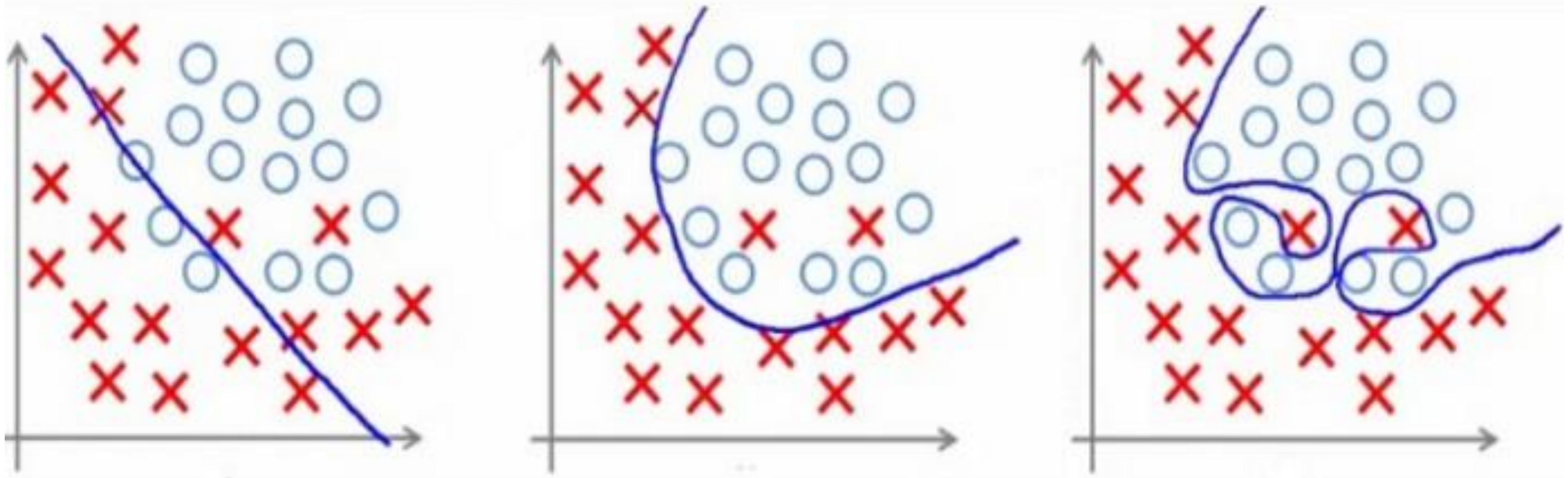
Exercise

- Which model would you choose?



Exercise

- Which model would you choose?



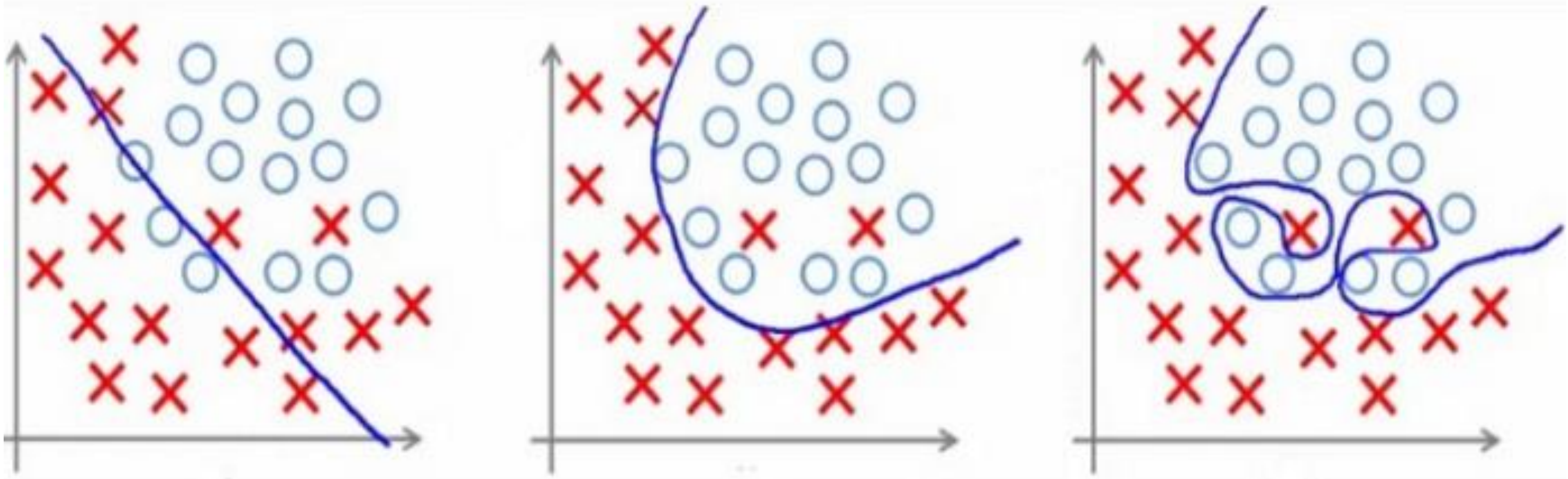
Underfits: too simple to explain the data!

Overfits: too complex to generalize to new data!

Exercise

- Which model would you choose?

Key challenge for neural networks since they have many parameters!

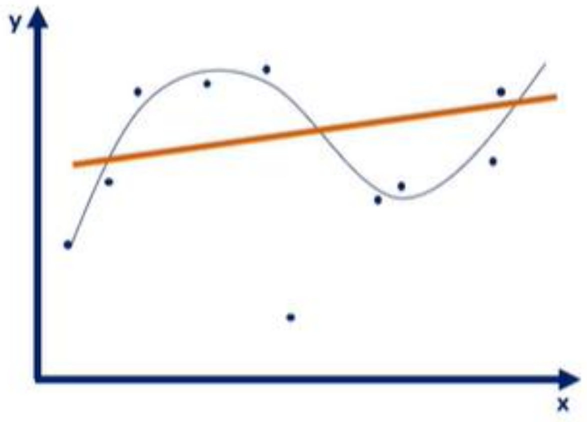


Underfits: too simple to explain the data!

Overfits: too complex to generalize to new data!

The Problem of Overfitting

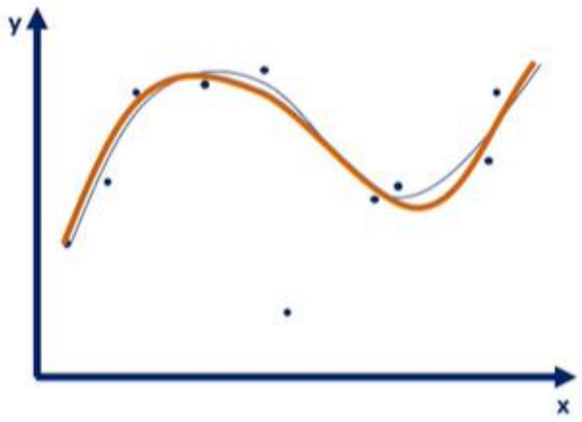
An **underfitted** model



Doesn't capture any logic

- High loss
- Low accuracy

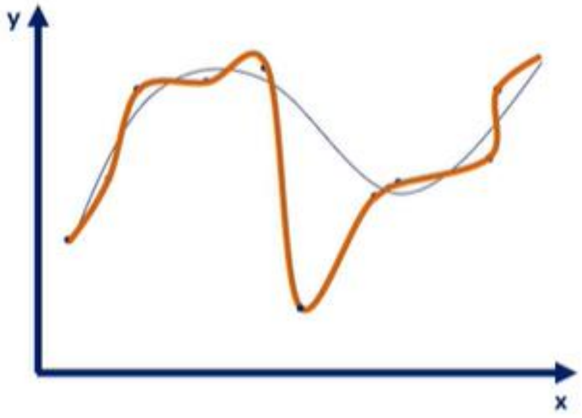
A **good** model



Captures the underlying logic of the dataset

- Low loss
- High accuracy

An **overfitted** model



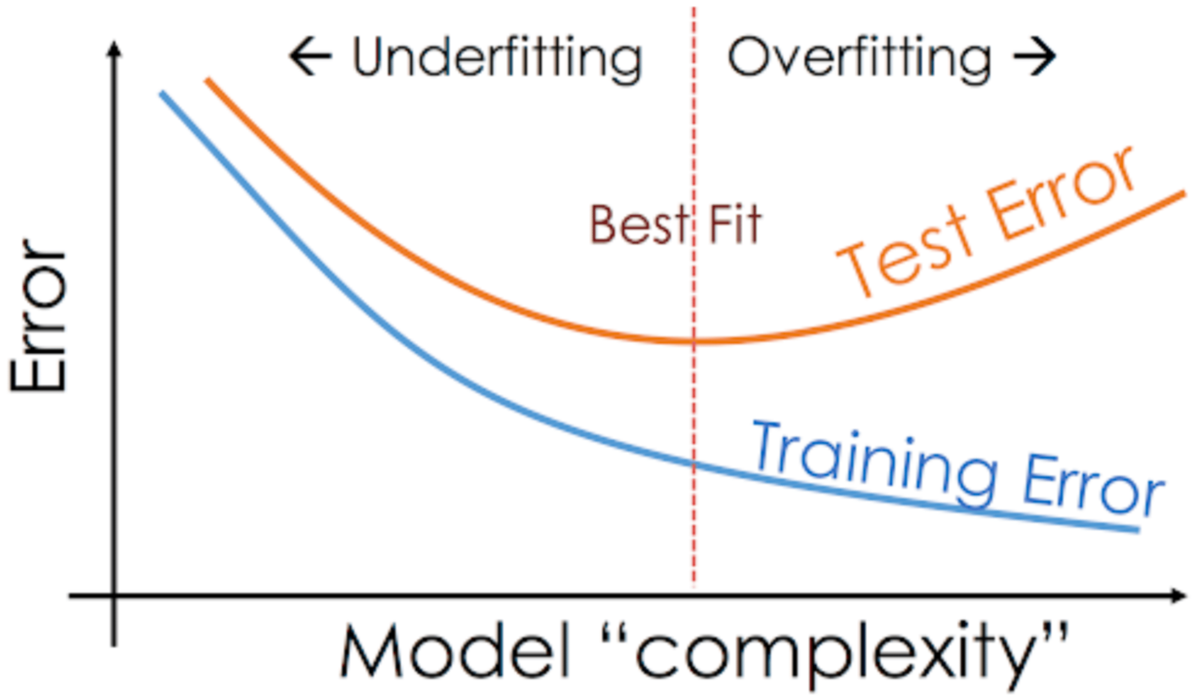
Captures all the noise, thus "missed the point"

- Low loss
- Low accuracy

Overfitting

- To detect overfitting, analyze error/loss for models tested on training data and test/validation data.
  - What happens to training data error as the number of training steps increases?

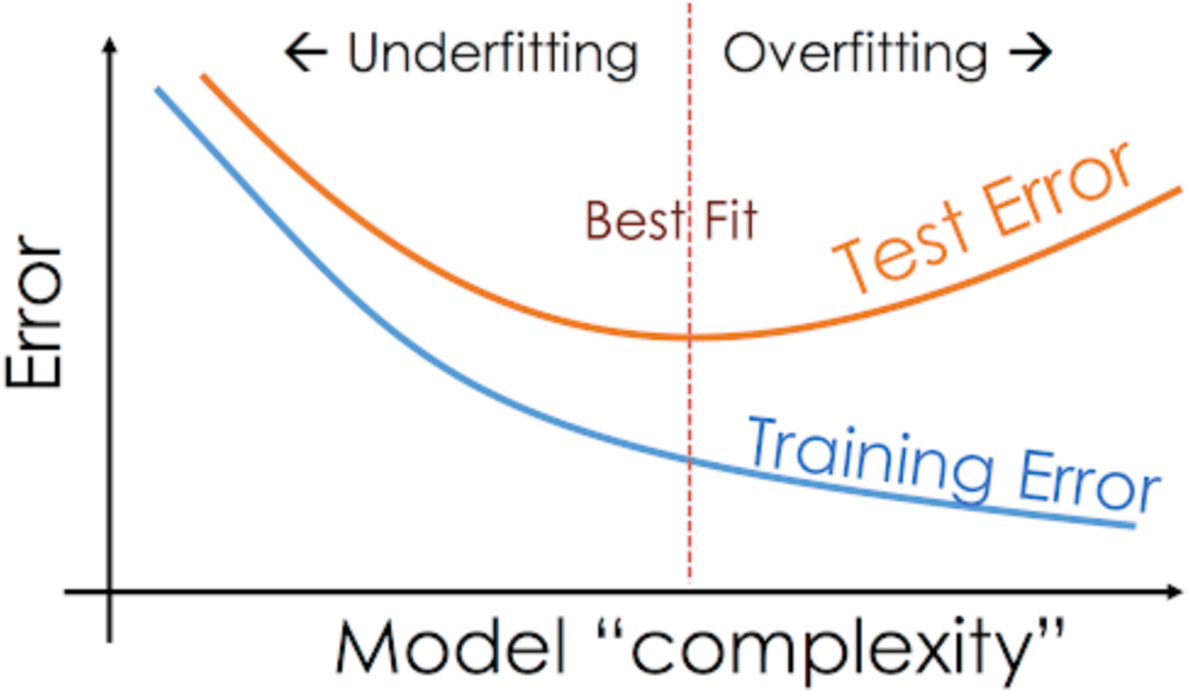
Error shrinks!



Overfitting

- To detect overfitting, analyze error/loss for models tested on training data and test/validation data.
  - What happens to test/validation error as the number of training steps increases?

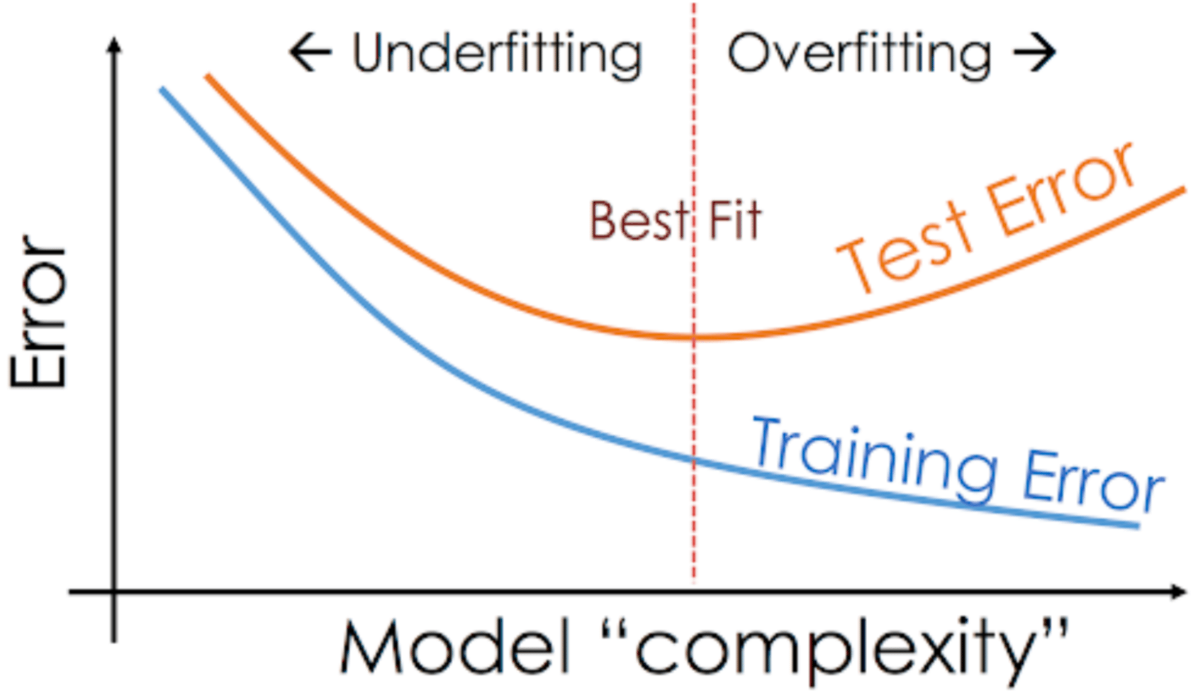
Error shrinks and then grows!



Overfitting

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

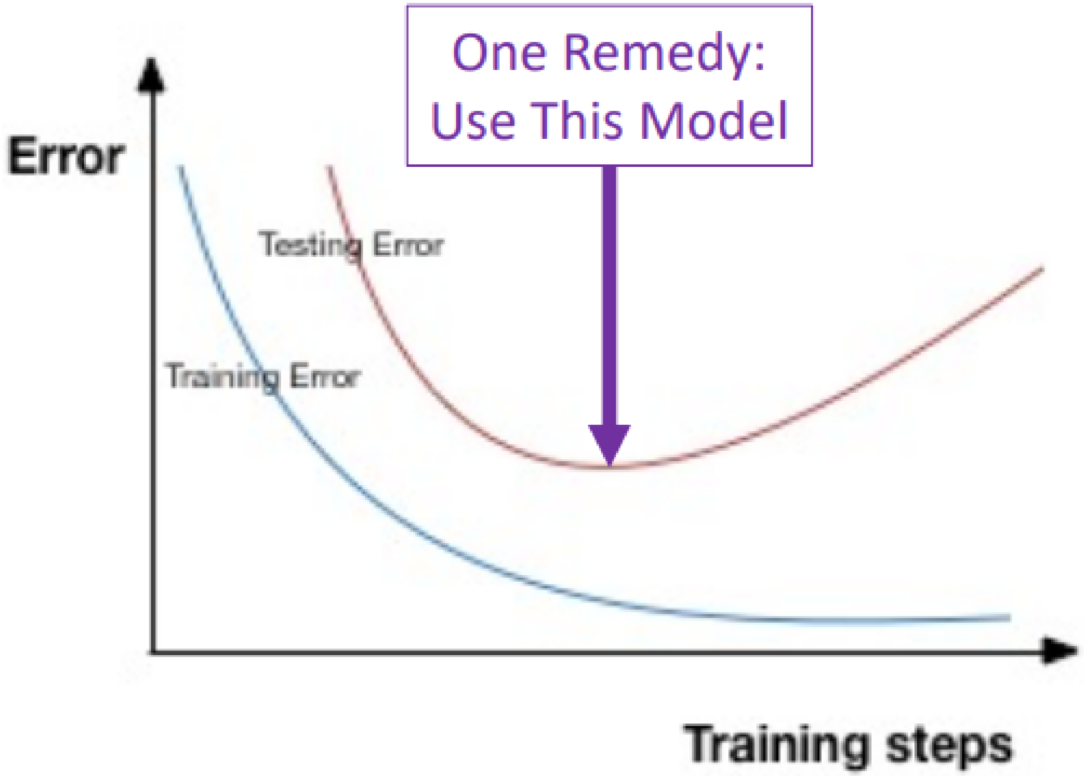
- To detect overfitting, analyze error/loss for models tested on training data and test/validation data.
  - Why does training error **shrink** and test/validation error **grow**?



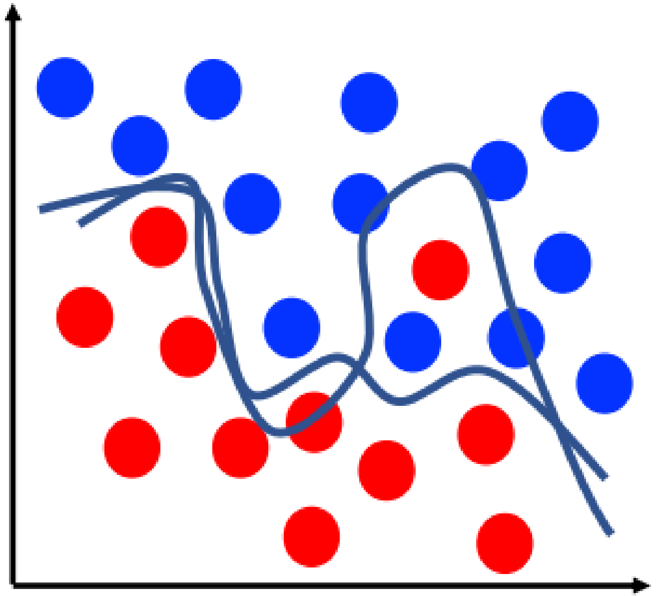
Modeling noise in the training data (i.e., overfitting) reduces training error and the expense of losing knowledge that generalizes to unobserved test data.

How to Avoid Overfitting?

Early stopping



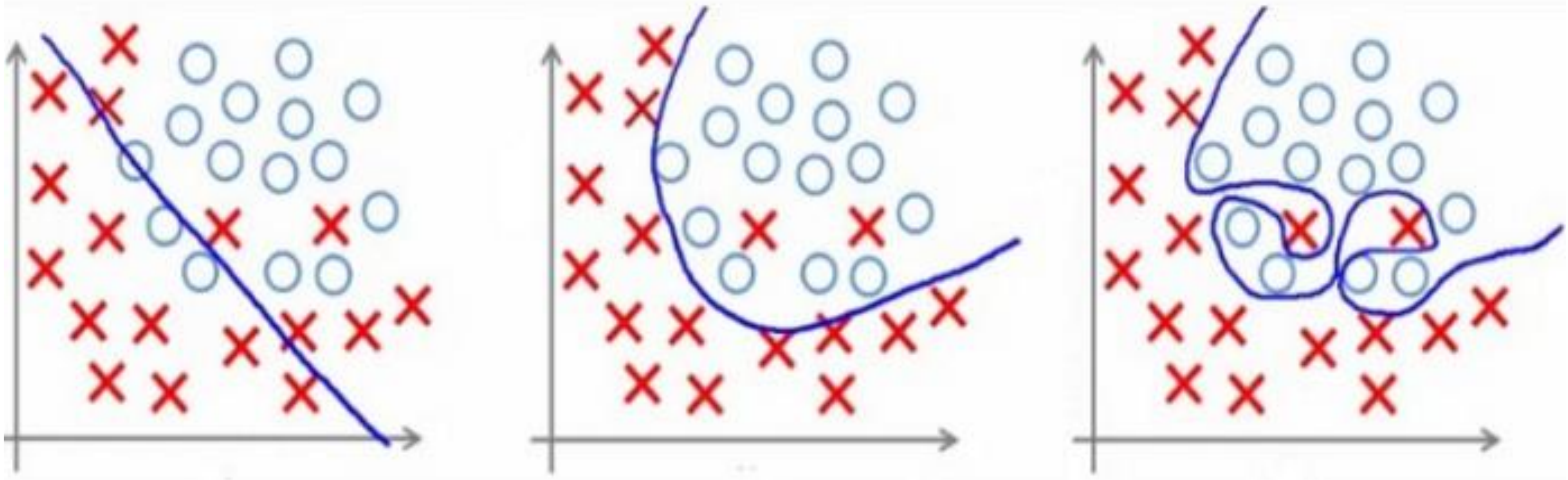
Add training data



Many more techniques to be discussed in this course…

Underfitting

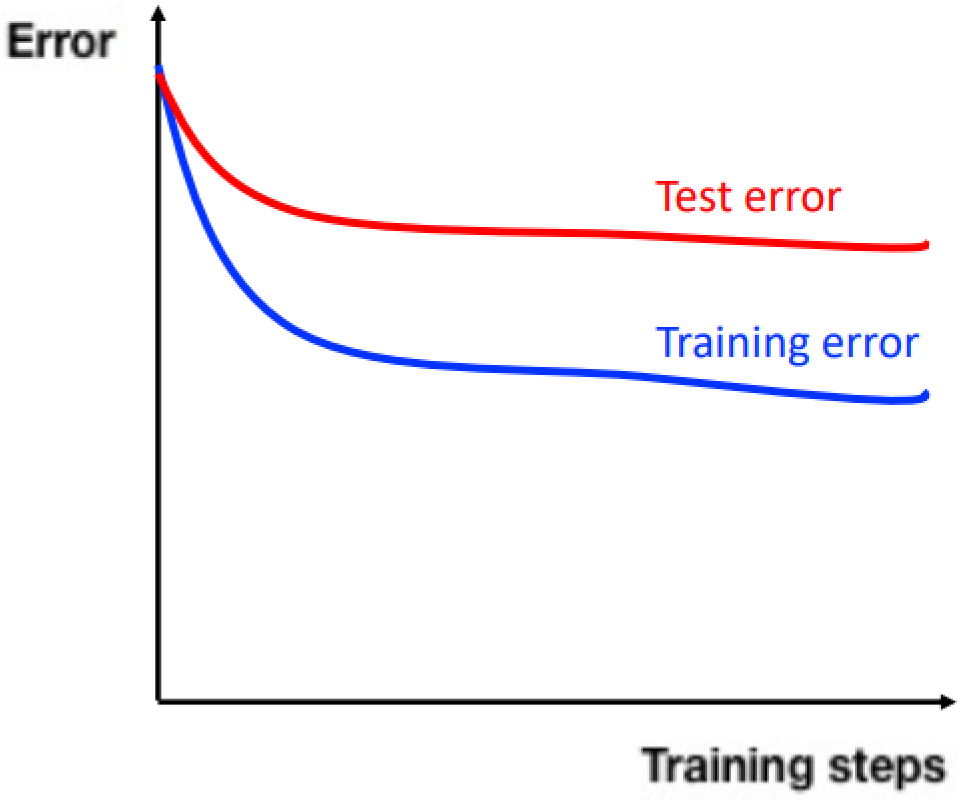
Underfits: too simple to explain the data!



Underfitting

- To detect overfitting, analyze error/loss for models tested on training data (and optionally test/validation data):
  - What happens to training data error as number of training steps increases?

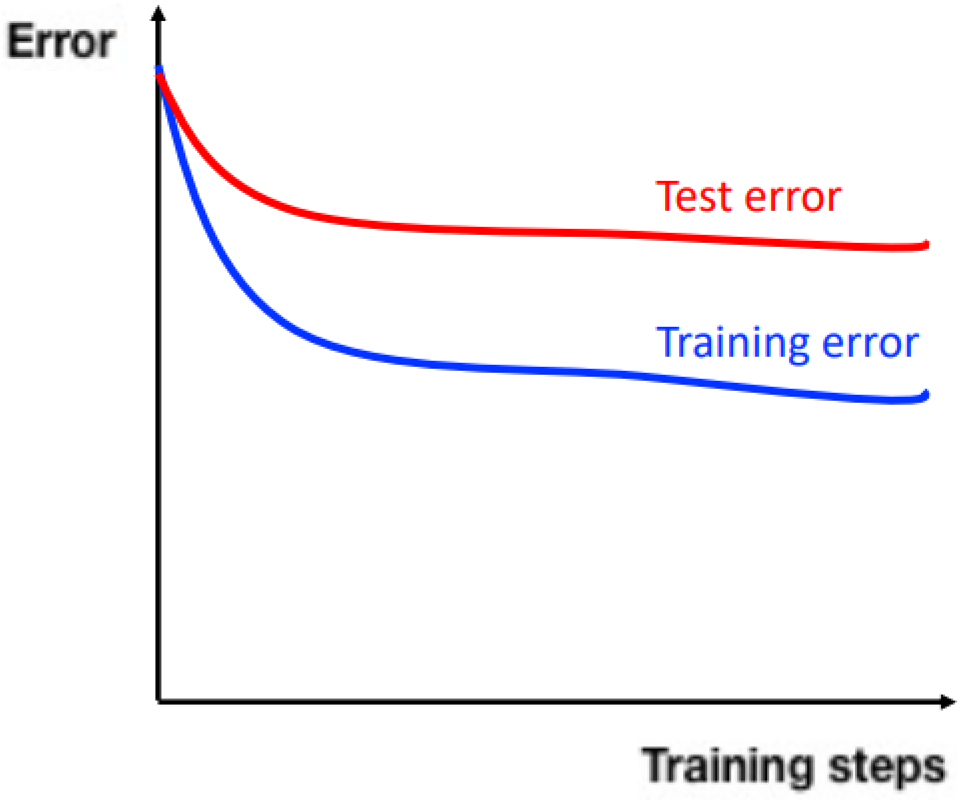
Error remains high!



Underfitting

- To detect overfitting, analyze error/loss for models tested on training data (and optionally test/validation data):
- What happens to test/validation data error as number of training steps increases?

Error remains high!



How to Avoid Underfitting?

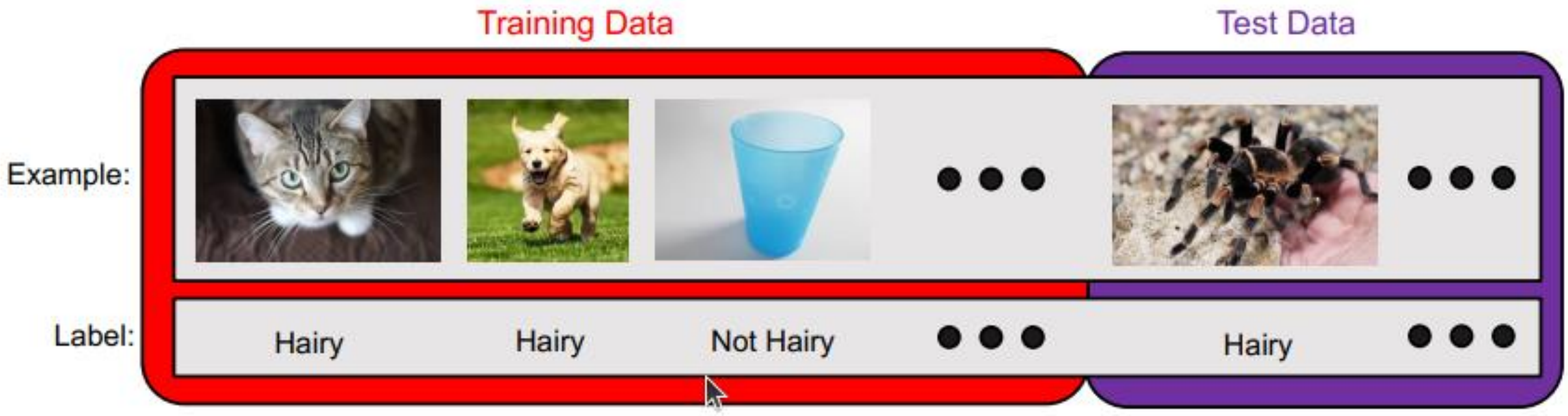
Increase representational complexity, for example increase the number of layers and/or units in a neural network.

Underfitting vs Overfitting

Goal: learn a model with a capacity that is neither too small nor too large so it can generalize well when predicting on previously unseen test data.

Selecting Model Hyperparameters

- Our goal is to design models that generalize well to new, previously unseen examples (Test Data).



Key Challenge: how to select a model without repeatedly observing the test data (which leads to overfitting)?

Model Design

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

Model hyperparameters (selected); e.g.,

- Number of layers
- Number of units in each layer
- Activation function
- Batch size
- Learning rate
- ...

Model parameters (learned)

- Weights
- Biases

Key Challenge: how to select a model without repeatedly observing the test data (which leads to overfitting)?

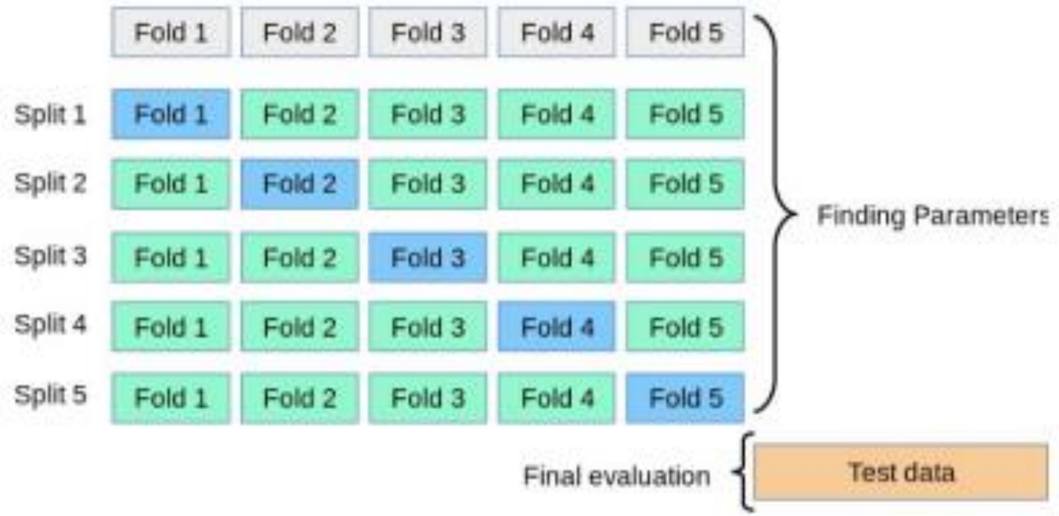
Hyperparameter Tuning

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

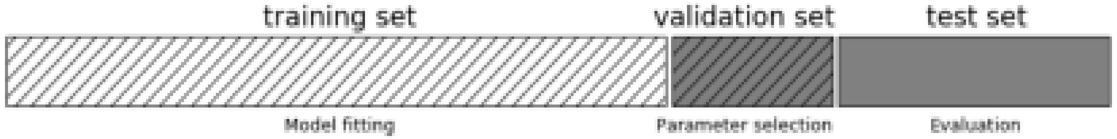
- Split the training set so it can ve used to test different hyperparameters

For statistically strong results:

Small training dataset: cross validation

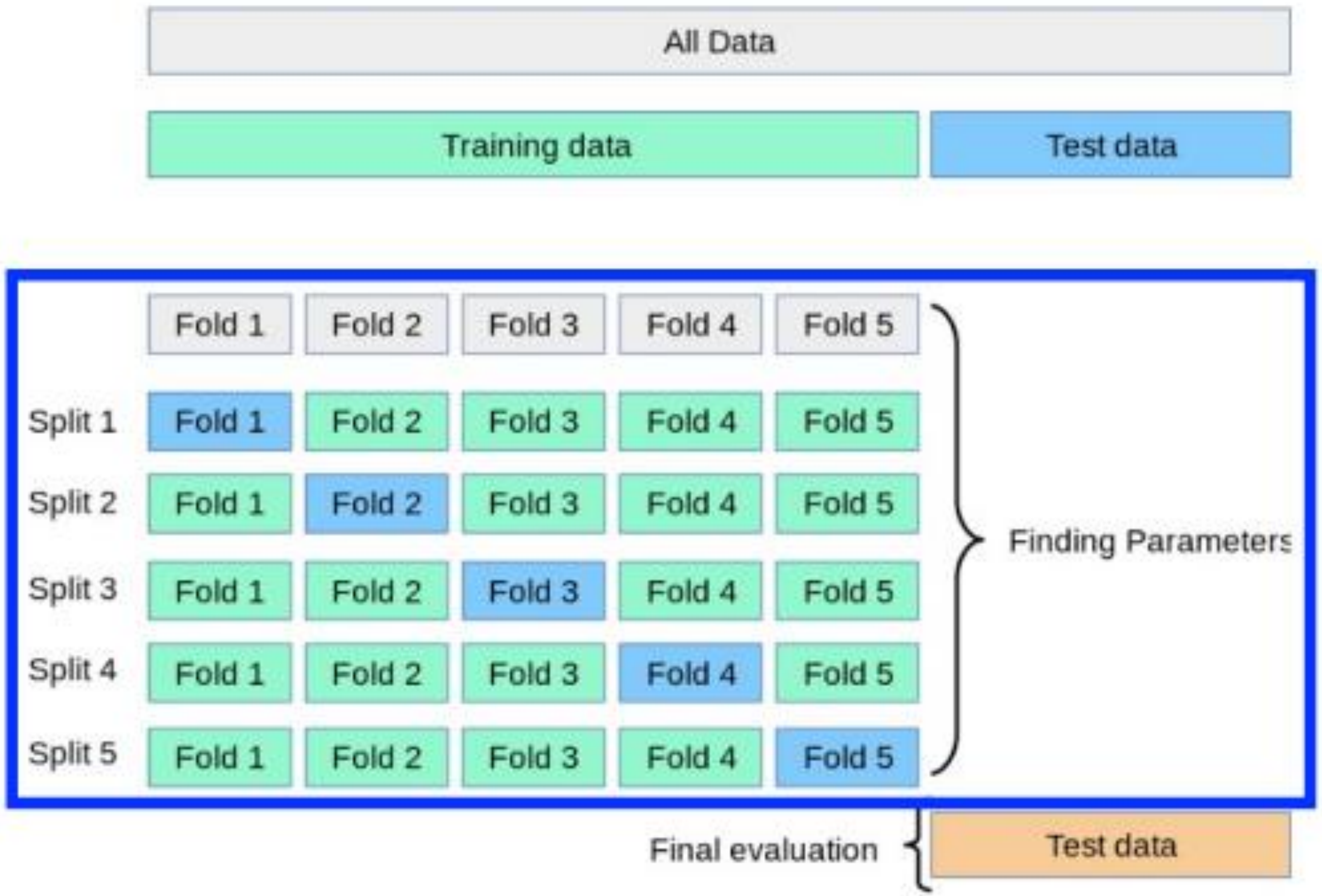


Else: train/validation split



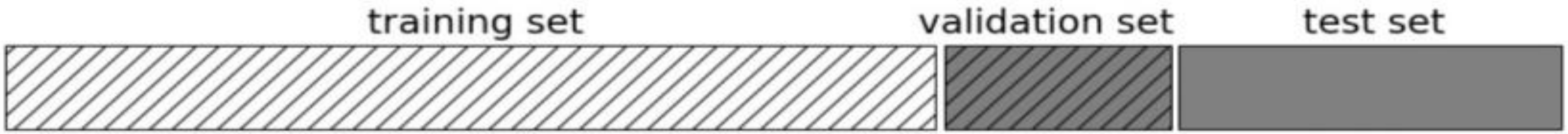
Cross-Validation

- Limit influence of dataset split



Validation Split

- Split data into "train" and "validation" datasets



- Hyperparameter selection: test models trained with different hyperparameter values on the validation set to find the best one

- Final model: retrain using the model hyperparameters selected from validation set testing using the data in the training AND validation splits.

Hyperparameters Summary

- So far we found the following hyperparameters:

- Model:
  - Network depth and width (number of layers, neurons per layer);
  - Type of activation functions (e.g., ReLU, Tanh).

- Training process:
  - Optimizer settings (learning rate, momentum, etc.);
  - Number of epochs and batch size;
  - Validation strategy (cross-validation or holdout).

- Dataset Considerations:
  - Data size (do we need more samples?);
  - Handling class imbalance.

Manual Hyperparameter Tuning

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Manual approach goal: achieve good performance on the test set

- Some examples of effect of hyperparamaters on model capacity:

  - Number of layers/nodes: increases capacity when increased.

  - Learning rate: increases capacity when tuned optimally.

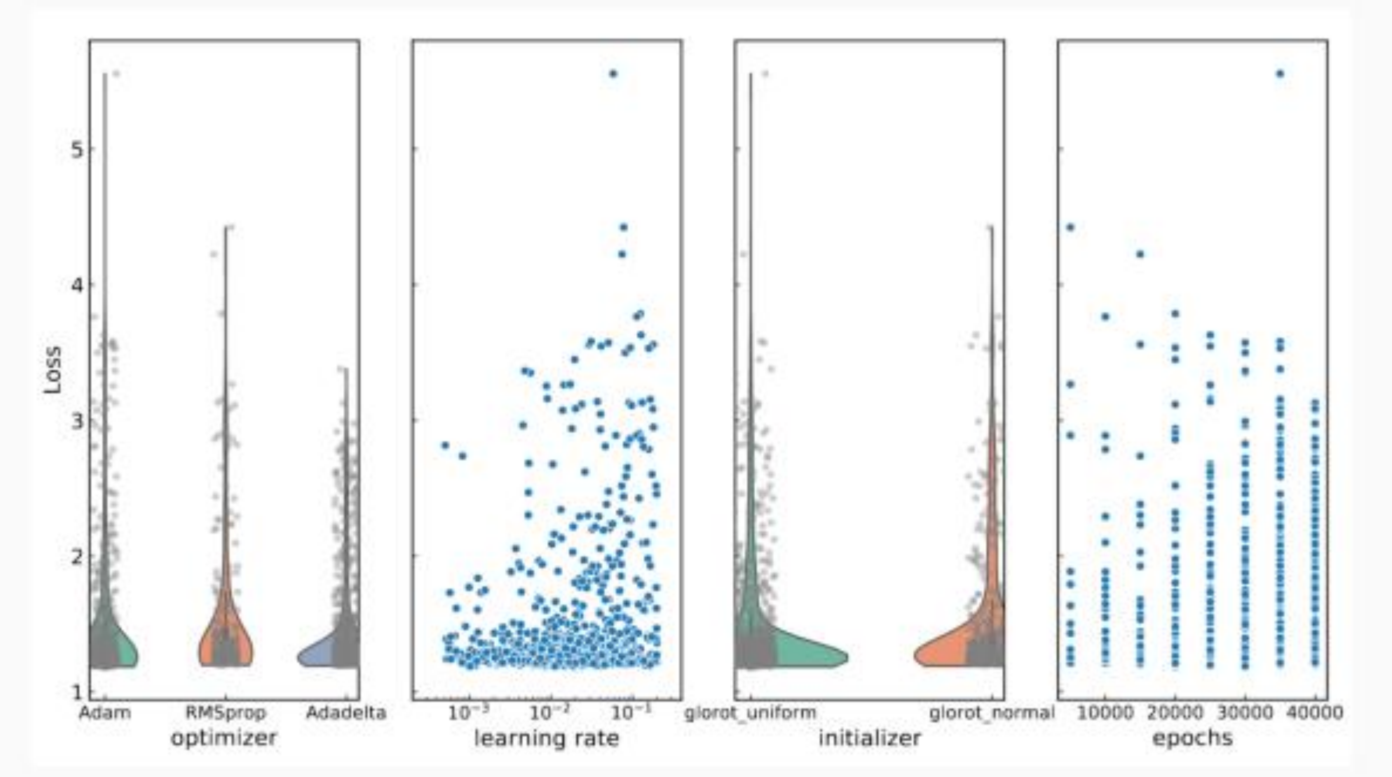
  - Weight decay: increases capacity when decreased.

  - Dropout rate: increases capacity when decreased.

Automatic Hyperparameter Optimization Algorithms

- Hyperparameter tuning is an optimization problem thus we can automate the process.

- Common algorithms:
  - Grid Search;
  - Random Search;
  - Bayesian Optimization;
  - Gradient-Based Optimization;
  - Evolutionary Optimization.

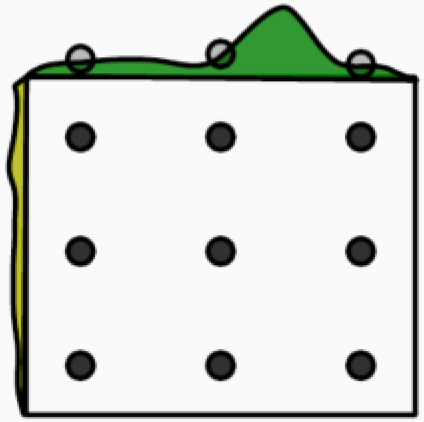


Grid Search

- Grid search: searching through a manually subset range of the hyperparameter space.

  - Train model for every grid point of the hyperparameter space.

  - Monitor the best validation set error → best hyperparameter values.



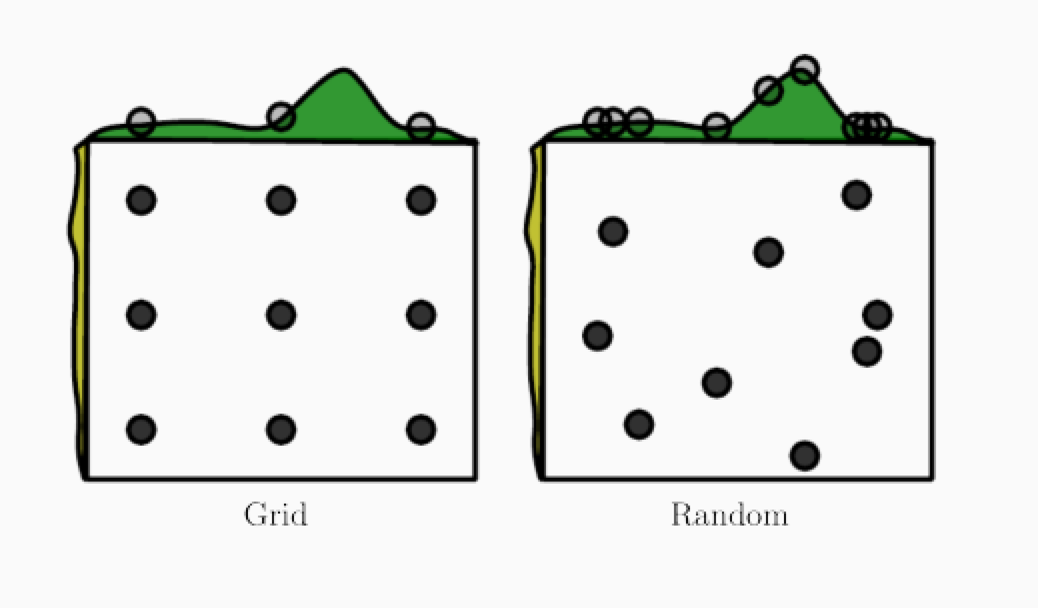
Grid

The disadvantage: with n values and m parameters the number of trials is $\mathcal{O}(n^m)$

Random Search

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

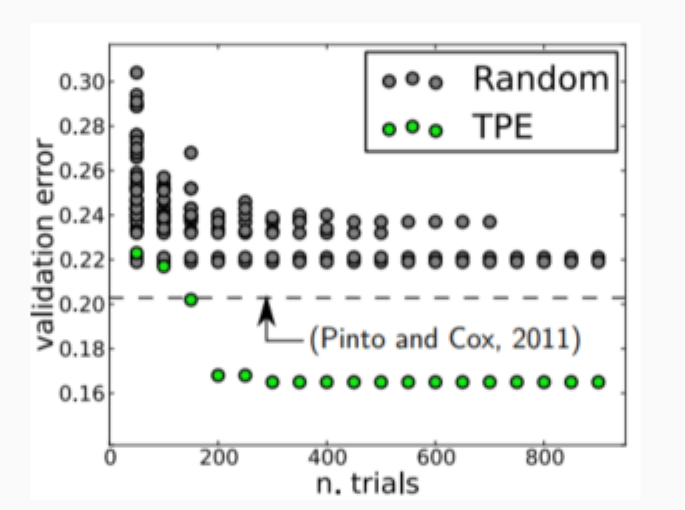
- Random search: sample trial points from a marginal distribution for each hyperparameter.
  - Do not discretize or bin the values of the hyperparameters.
  - The marginal distribution will perform independent explorations of hyperparameters.



Model-Based Hyperparameter Optimization

- Idea:
  - Perform a training using a set of hyperparameters
  - Define the cost function to be optimize as the validation set error
  - Use sequential model-based optimization (SMBO) approach, or algorithms which monitors the numerical gradient from the loss function.

- Example:
  - Bayesian Optimization
  - Tree-structured Parzen Estimator (TPE)



SMBO

SMBO minimizes functions $f : X \to \mathcal{R}$ where each evaluation is very expensive.

The f function is replaced by a **surrogate** function, $\bar{f}$, easier to manage.

The surrogate function proposes a new search point $\mathbf{x}_{i+1}$, $f(\mathbf{x}_{i+1})$ is computed and $\bar{f}$ updated or recomputed to approximate better the true loss function.

Data: loss function f, initial surrogate $\overline{f}_0$, number of trials T
Result: Candidate $\mathbf{x}_{best}$ for the minimum of f
Set trials history $H = \emptyset$;
for $i = 1$ **to** T **do**
    $x^\star \leftarrow \text{argmin}_{\mathbf{x}} L(\mathbf{x}, \overline{f}_{i-1})$;
    Compute $f(x^\star)$;
    $H \leftarrow H \cup \{\mathbf{x}^\star, f(\mathbf{x}^\star)\}$;
    Model a new surrogate function f_i using H;
end

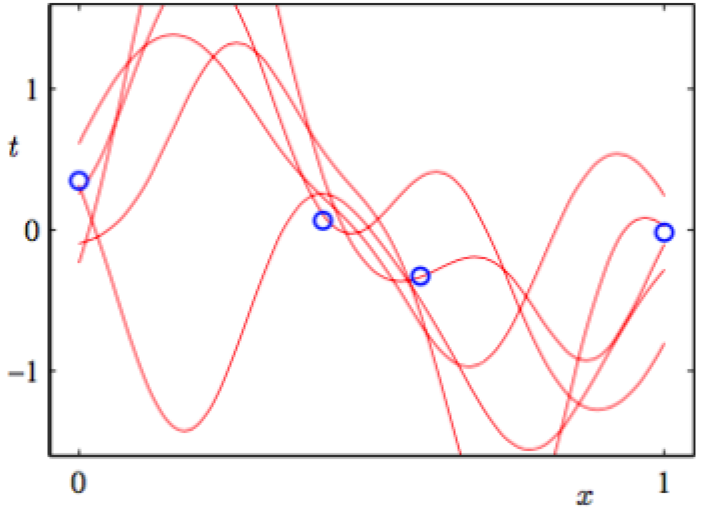
Where $L(\mathbf{x}, \bar{f})$, the criterion, and $\bar{f}$ depend on the specific algorithm.

Bayesian Optimization

- The function we're trying to optimize (e.g. loss as a function of hyperparameters) is really complicated.

- Bayesian Optimization tries to approximate it with a simpler function, called the surrogate function.

- After we've tried a certian number of hyperparameter combinations, we can condition on these hyperparameters to infer the posterior over the surrogate function using for instance Bayesian linear regression.



Bayesian Optimization

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- To choose the next point to query, we must define an acquisition function, which tells us how promising a candidate it is.

- **Candidate 1:** probability of improvement (PI)

$$\mathrm{PI} = \Pr(f(\theta) < \gamma - \epsilon),$$

where γ is the best value so far, and ϵ is small.

PI: Probability of Improvement.
Pr: Probability.
f(θ): The objective function value at a specific point θ in the parameter space. This function represents the performance measure being optimized (e.g., accuracy, loss).
γ: The best value obtained so far (the optimal value) during the optimization process.
ε: A small positive value that represents a margin of improvement. It is used to define how much better a new sample must be compared to the current best value γ to be considered an improvement.

- The problem with Probability of Improvement (PI): it queries points it is highly confident will have a small imporvement
  - Usually these are right next to ones we’ve already evaluated

Bayesian Optimization

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- A better choice: **Expected Improvement (EI)**

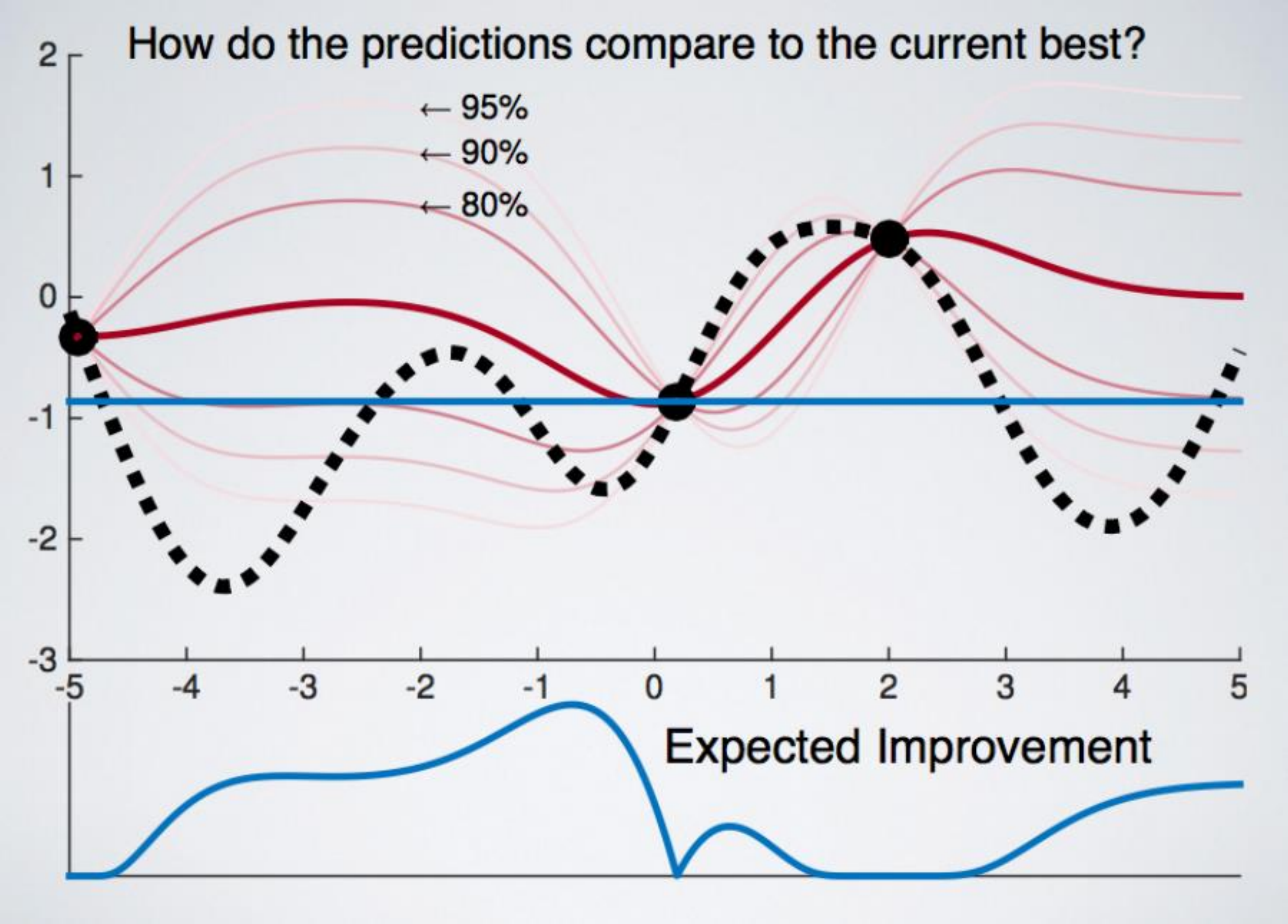
$$\mathrm{EI} = \mathbb{E}[\max(\gamma - f(\theta), 0)]$$

EI: Expected Improvement.
E: Expectation (or average) over the distribution of possible function values at **θ**.
max(γ - f(θ), 0): The improvement at a particular point **θ**, which is the difference between the best value **γ** and the predicted value **f(θ)**. If the predicted value is worse than **γ**, the improvement is set to 0 (i.e., no improvement).

  - The idea: if the new value is much better, we win by a lot; if it's much worse, we haven't lost anything

  - The Expected Improvement (EI) metric balances exploitation (sampling where the model predicts good performance) and exploration (sampling where uncertainty is high).

Bayesian Optimization



TPE

- Also a SMBO that uses surrogates.

- Key idea: model $P(x|y)$ instead of $P(y|x)$
  - x = value of single hyperparameter
  - y = loss

- Two surrogate models are mantained:
  - A distribution for bad values: $P(x|y > y^*) = P(x|\text{bad})$
  - A distribution for good values: $P(x|y \leq y^*) = P(x|\text{good})$

y* : threshold that determines good/bad splits

TPE

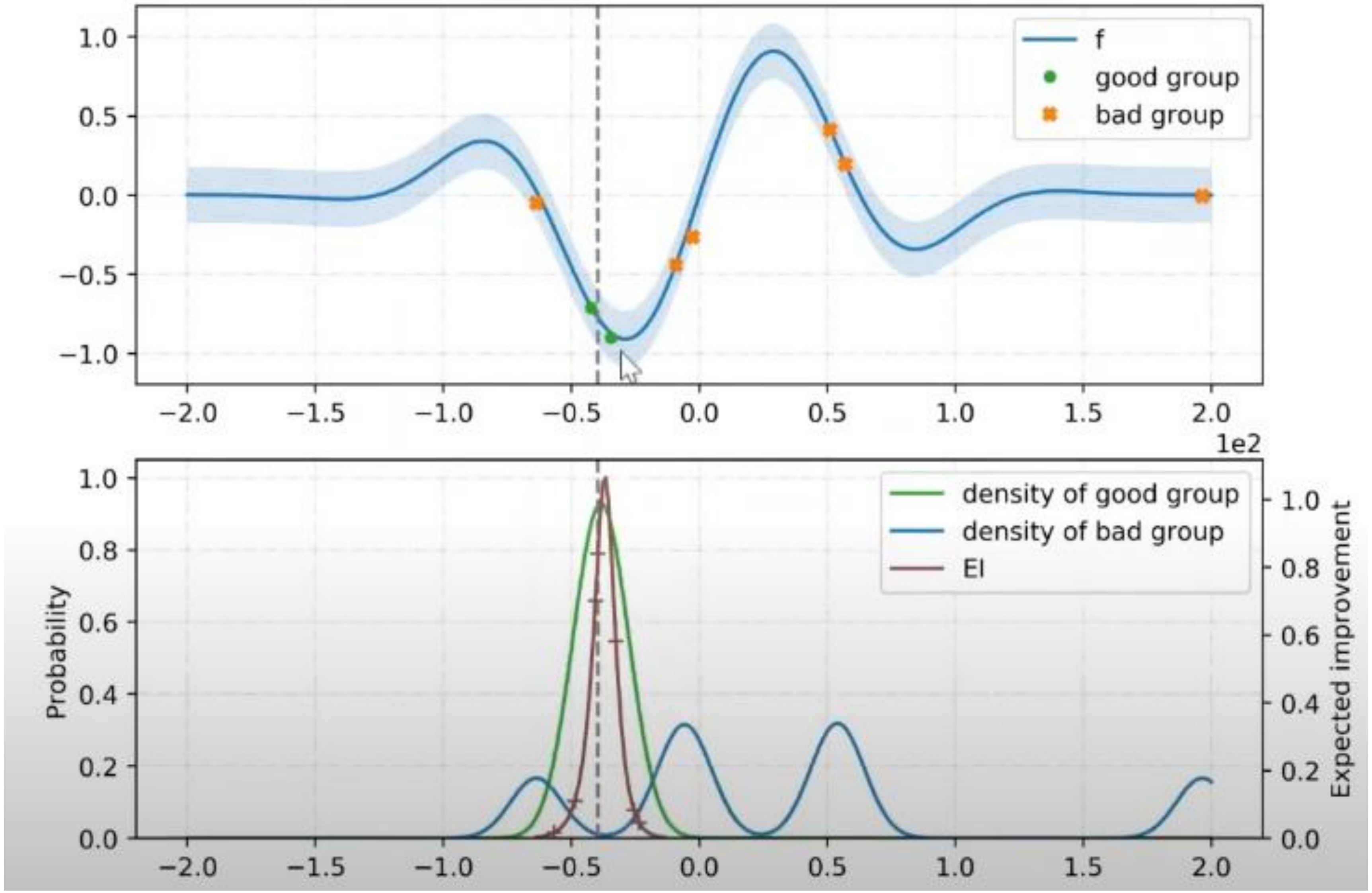
- How to get new promising candidates?

- Idea: a promising candidate is likely to
  - Have low probability under the bad distribution $P(x|\text{bad})$
  - Have high probability under the good distribution $P(x|\text{good})$

$$\text{“Promisingness”} \propto \frac{P(x|\text{good})}{P(x|\text{bad})}$$

⇒ **Proportional to Expected Improvement**

TPE


f
good group
bad group
1e2
density of good group
density of bad group
EI
Probability
Expected improvement

Next Session: Regularization

- What is it?
  - Technique that constrains our optimization problem to discourage complex models.

"any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."

- Ch. 5.2 of Goodfellow book on Deep Learning

- Why do we need it?
  - Improve generalization of our model on unseen data.