



UNIVERSIDADE
CATÓLICA
PORTUGUESA

BRAGA

Deep Learning

Session 7

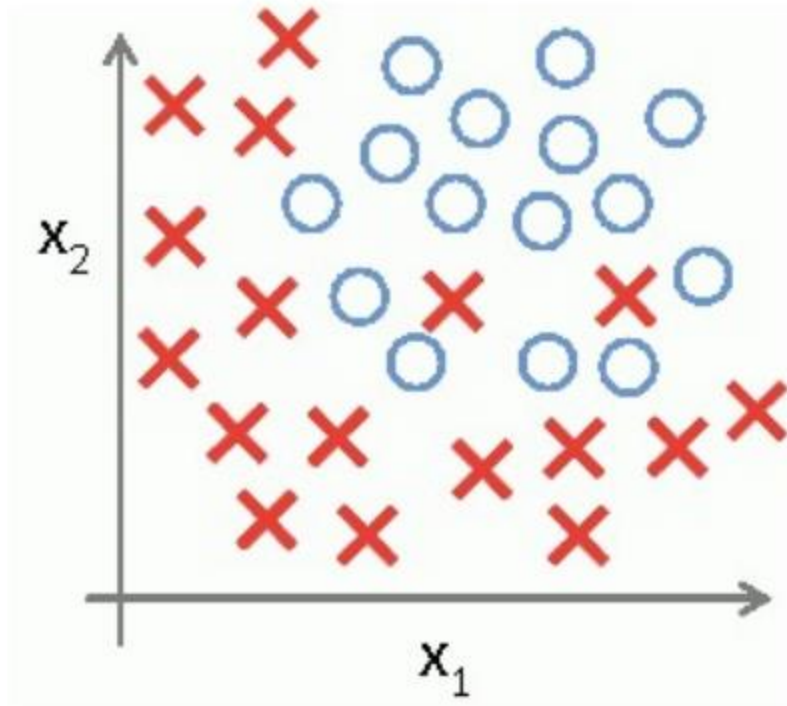
Model Selection and Hyperparameter Tuning

Applied Data Science

2024/2025

Exercise

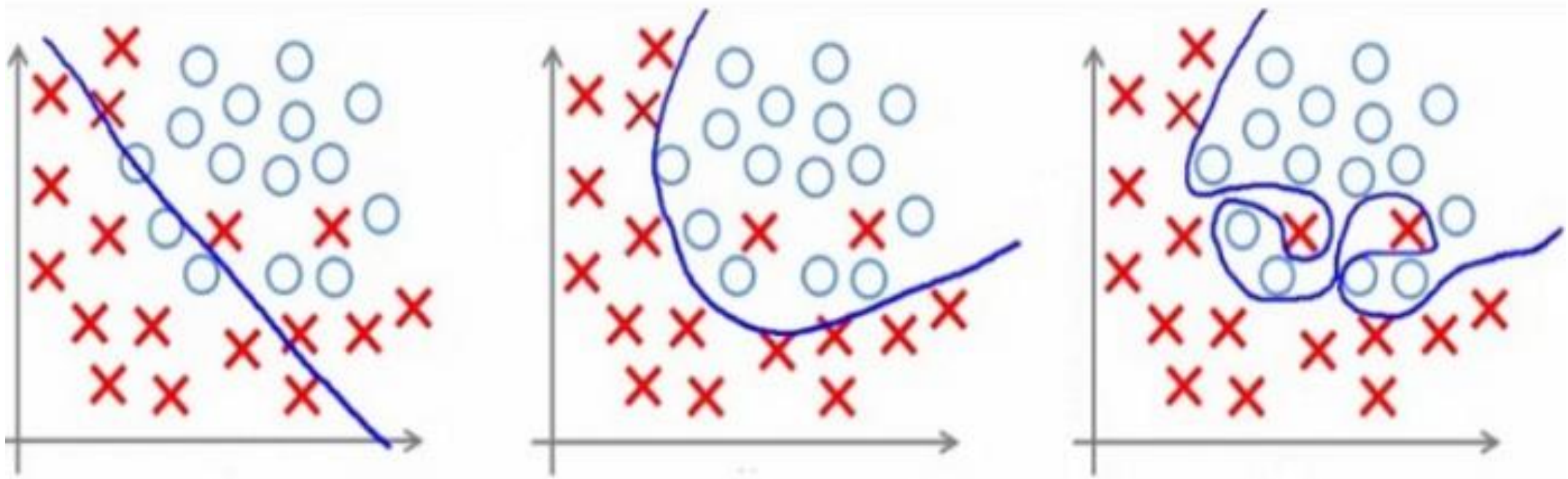
- Separate  from 



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

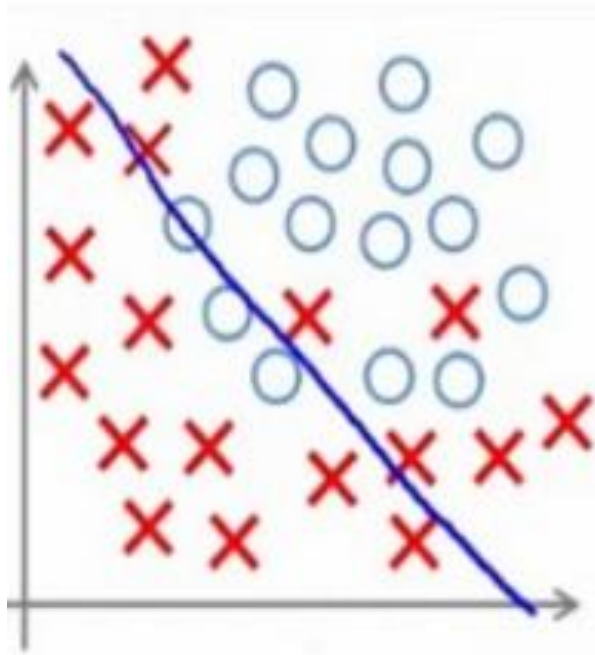
Exercise

- Which model would you choose?

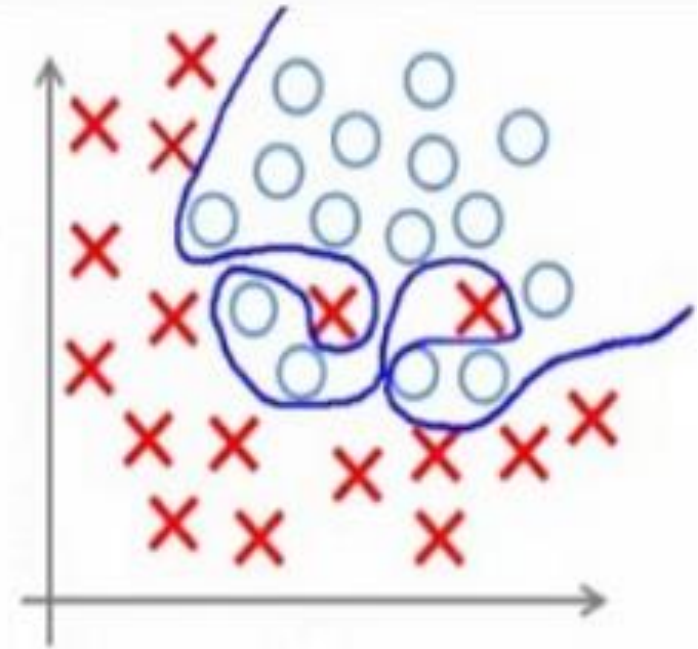
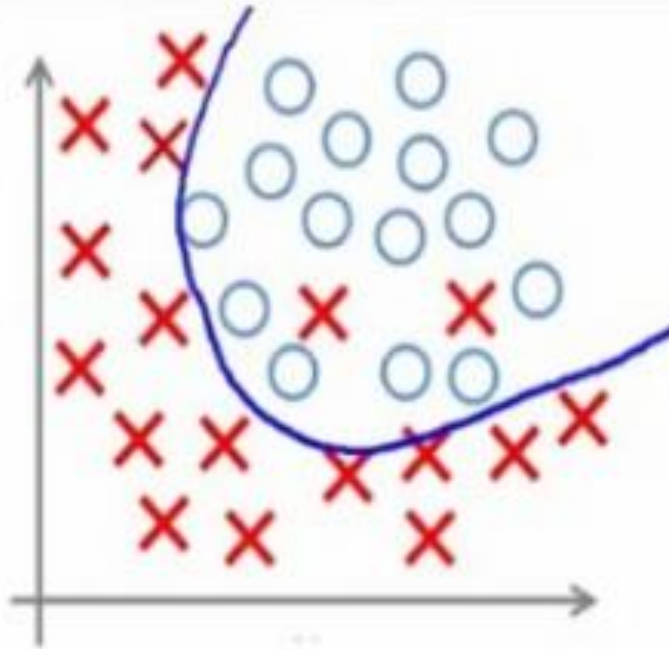


Exercise

- Which model would you choose?



Underfits: too simple to explain the data!

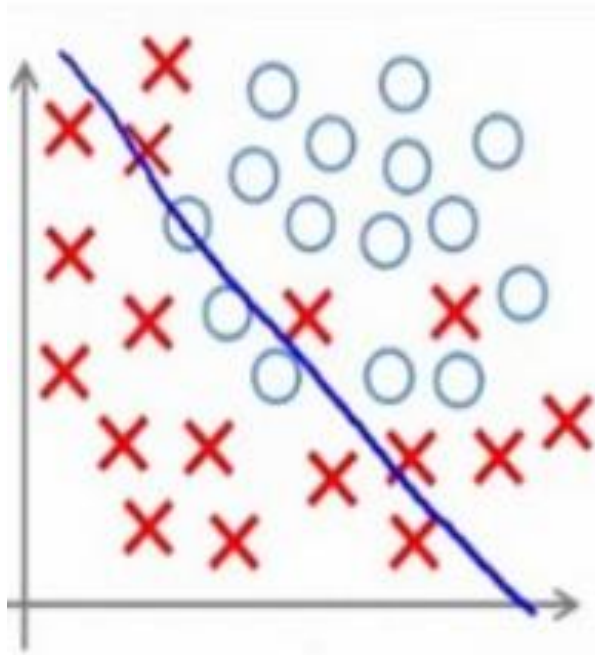


Overfits: too complex to generalize to new data!

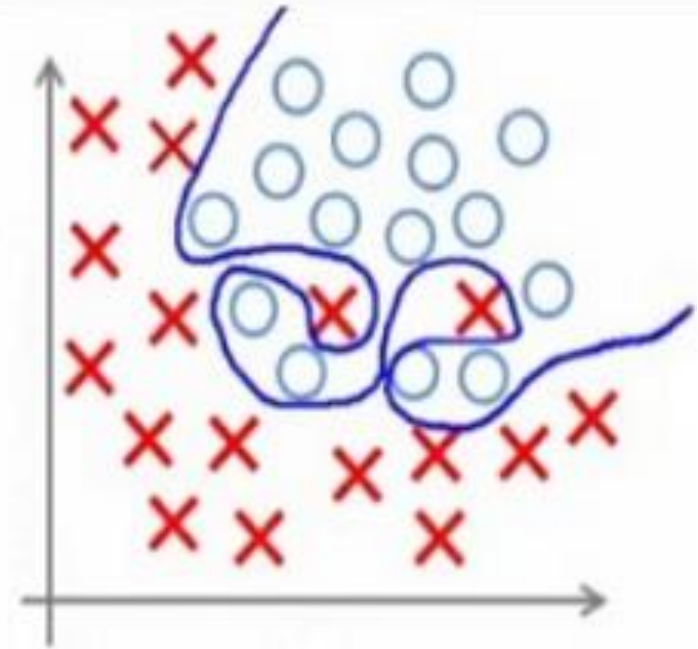
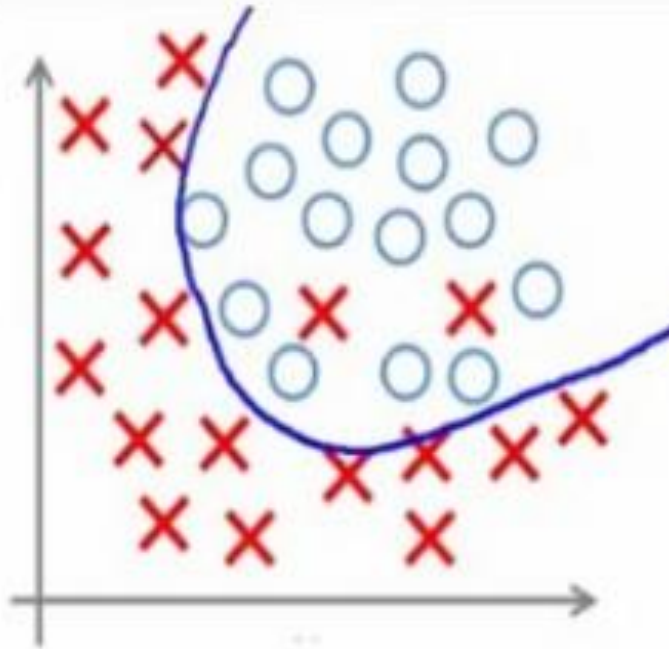
Exercise

- Which model would you choose?

Key challenge for neural networks since they have many parameters!



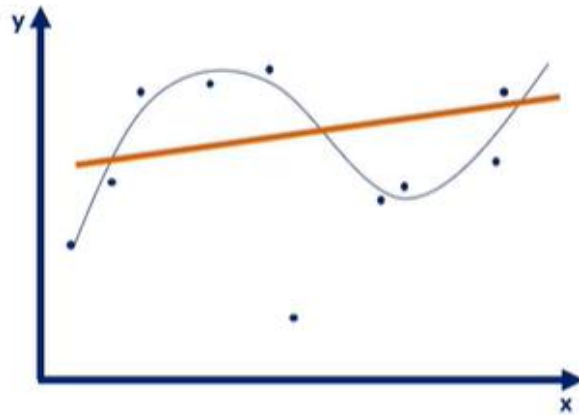
Underfits: too simple to explain the data!



Overfits: too complex to generalize to new data!

The Problem of Overfitting

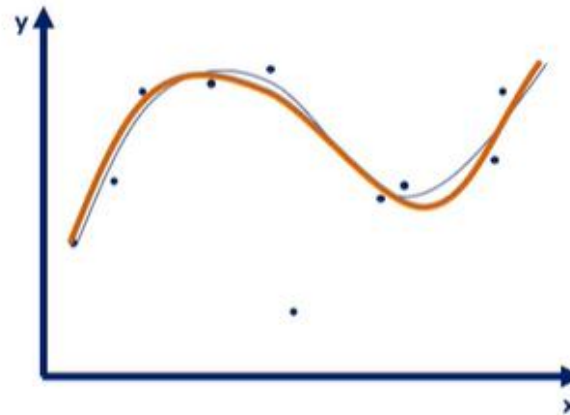
An **underfitted** model



Doesn't capture any logic

- High loss
- Low accuracy

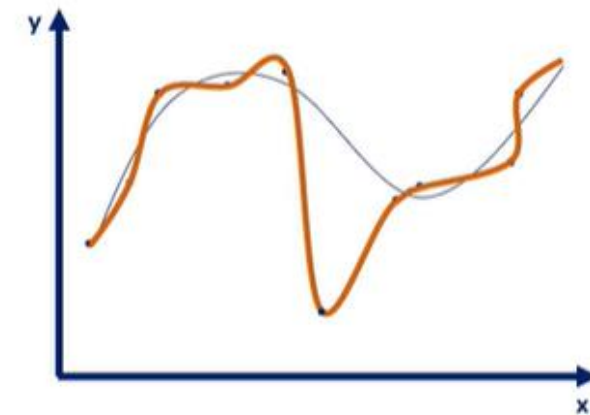
A **good** model



Captures the underlying logic of the dataset

- Low loss
- High accuracy

An **overfitted** model



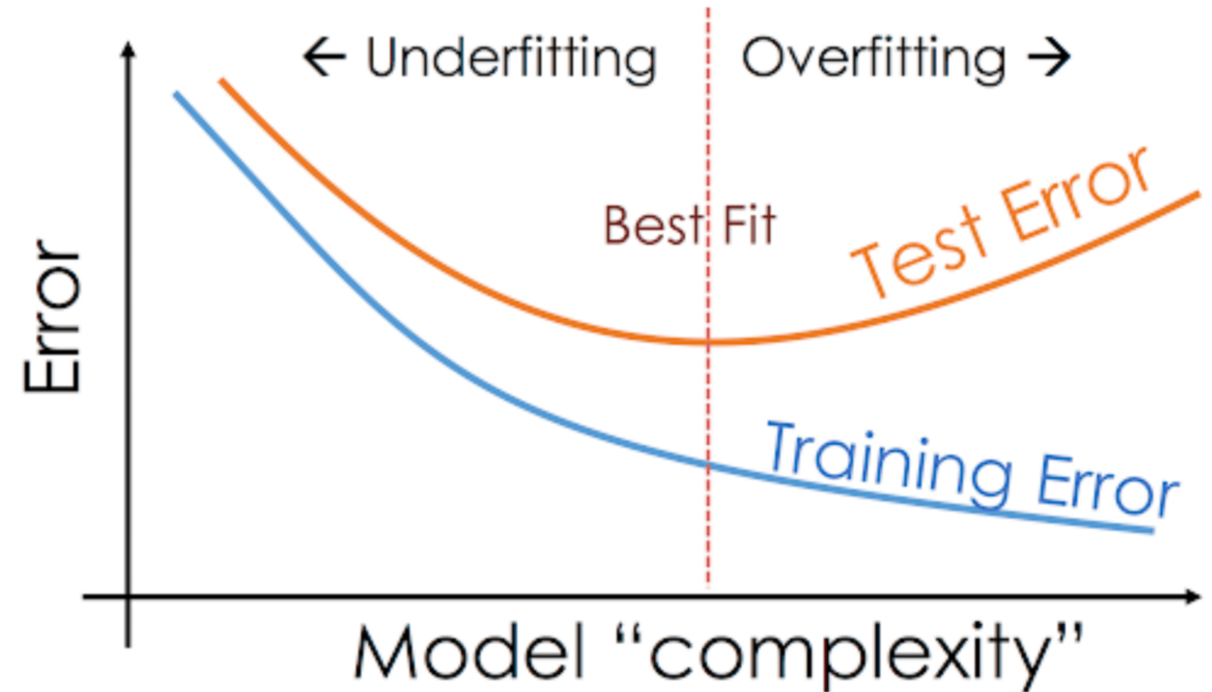
Captures all the noise, thus "missed the point"

- 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?

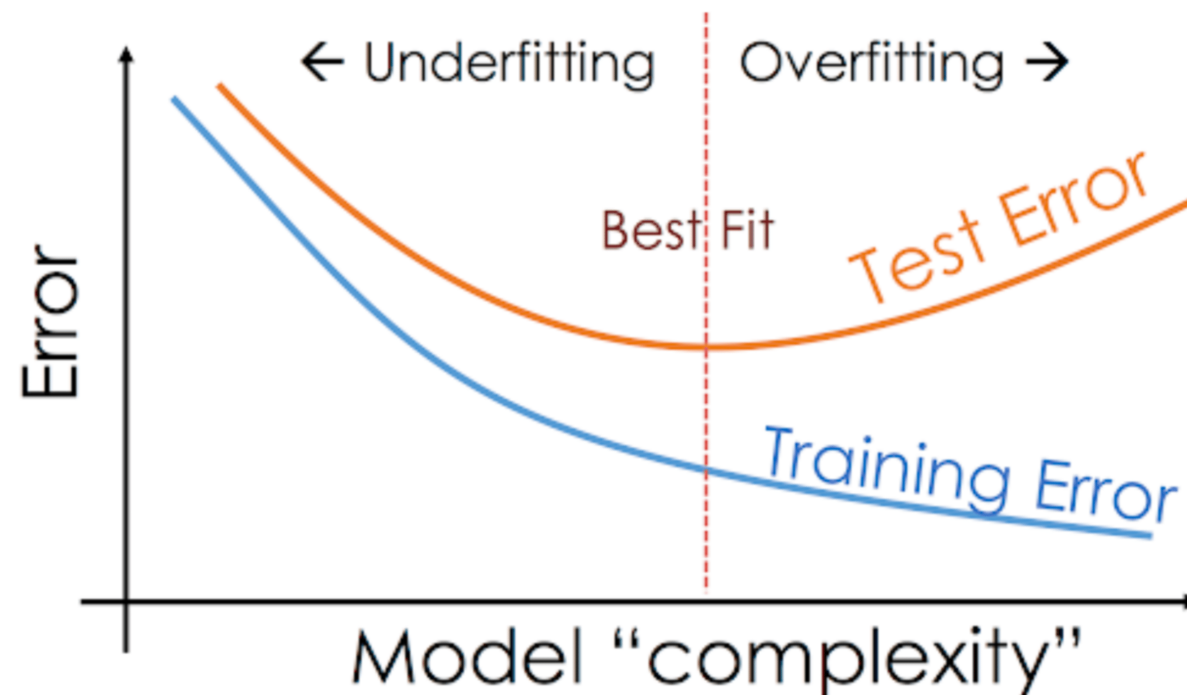
Error shrinks!



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?

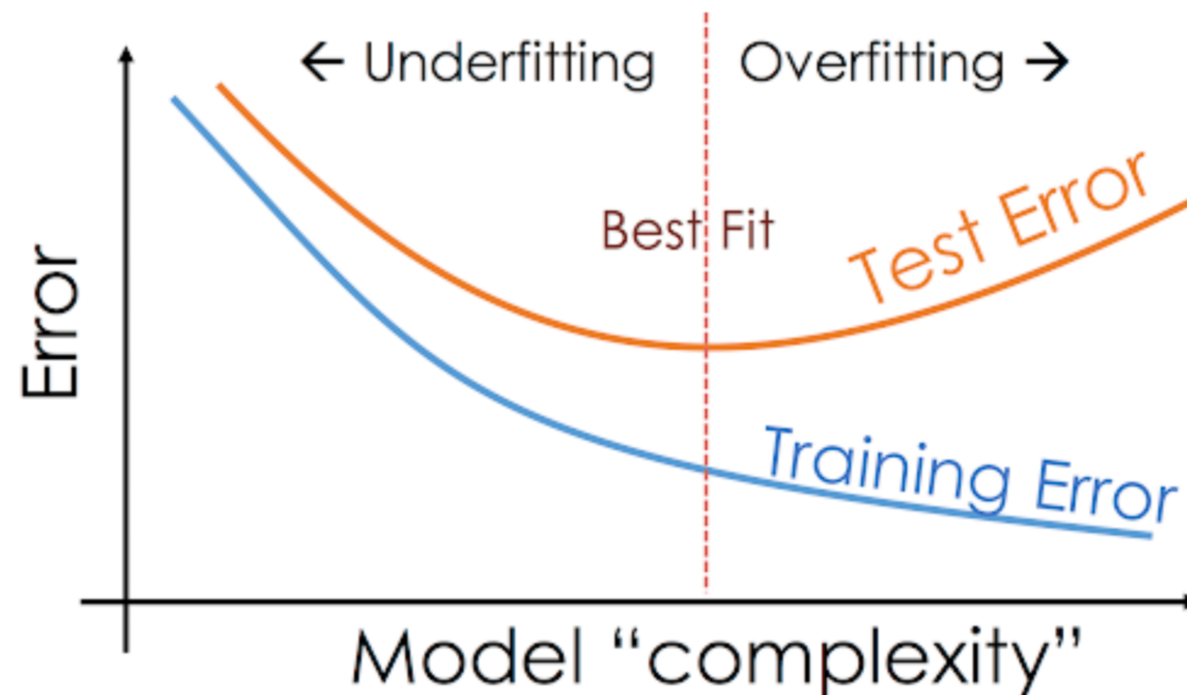
Error shrinks and then grows!



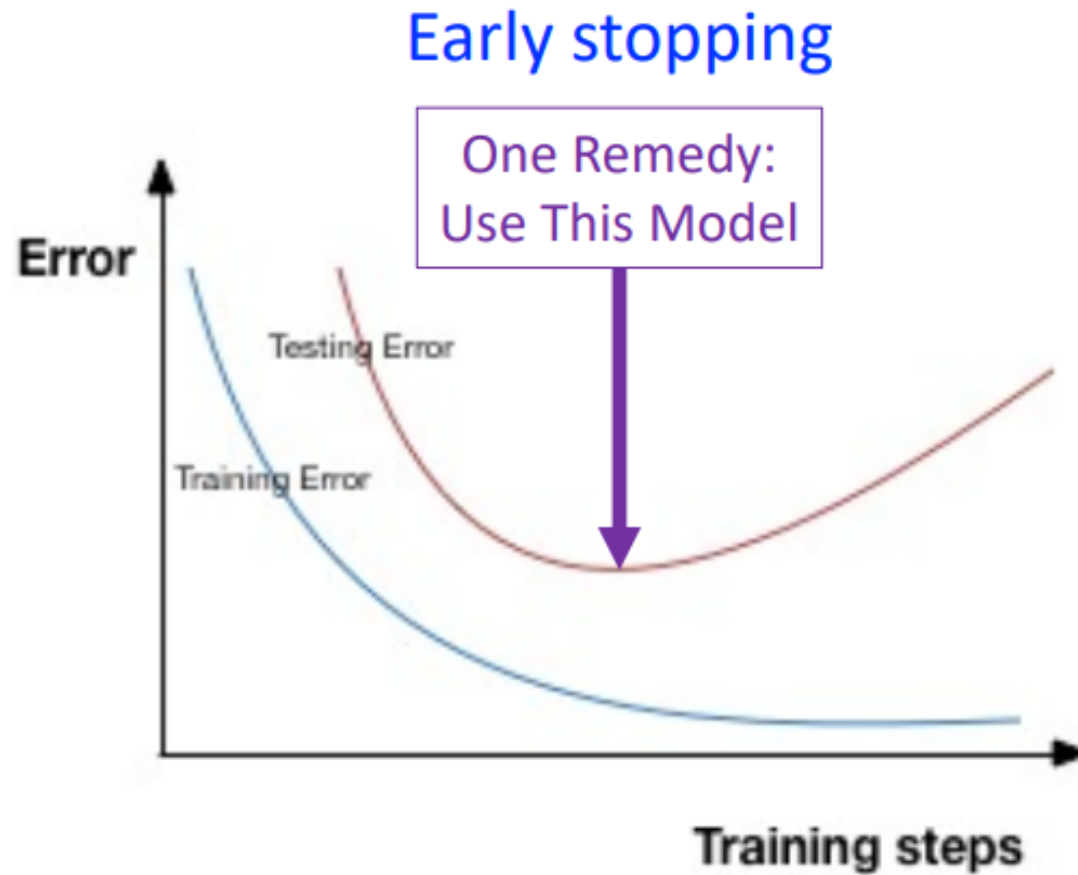
Overfitting

- 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**?

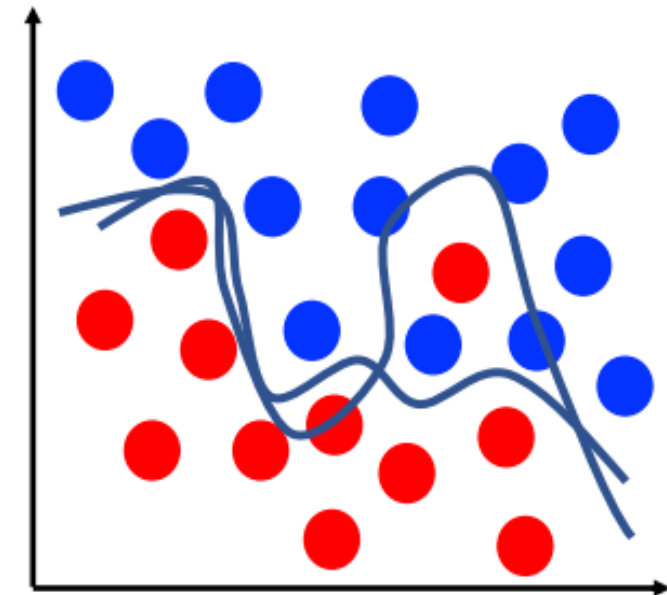
Modeling noise in the training data (i.e., overfitting) reduces training error and the expense of losing knowledge that generalizes to unobserved test data.



How to Avoid Overfitting?



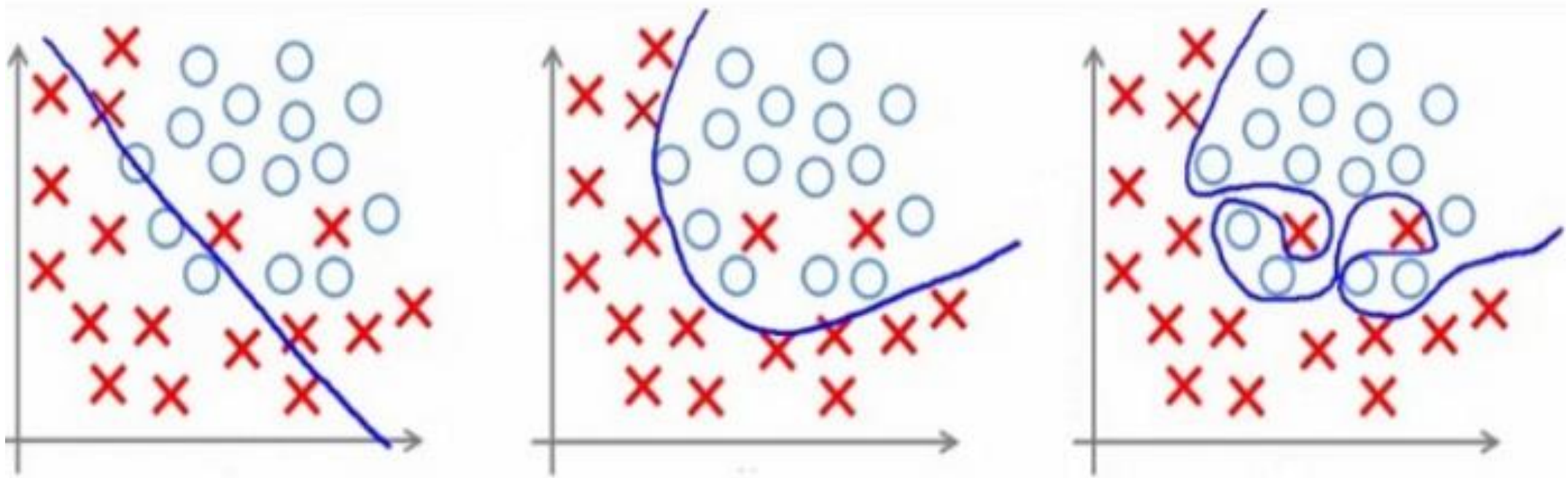
Add training data



Many more techniques to be discussed in this course...

Underfitting

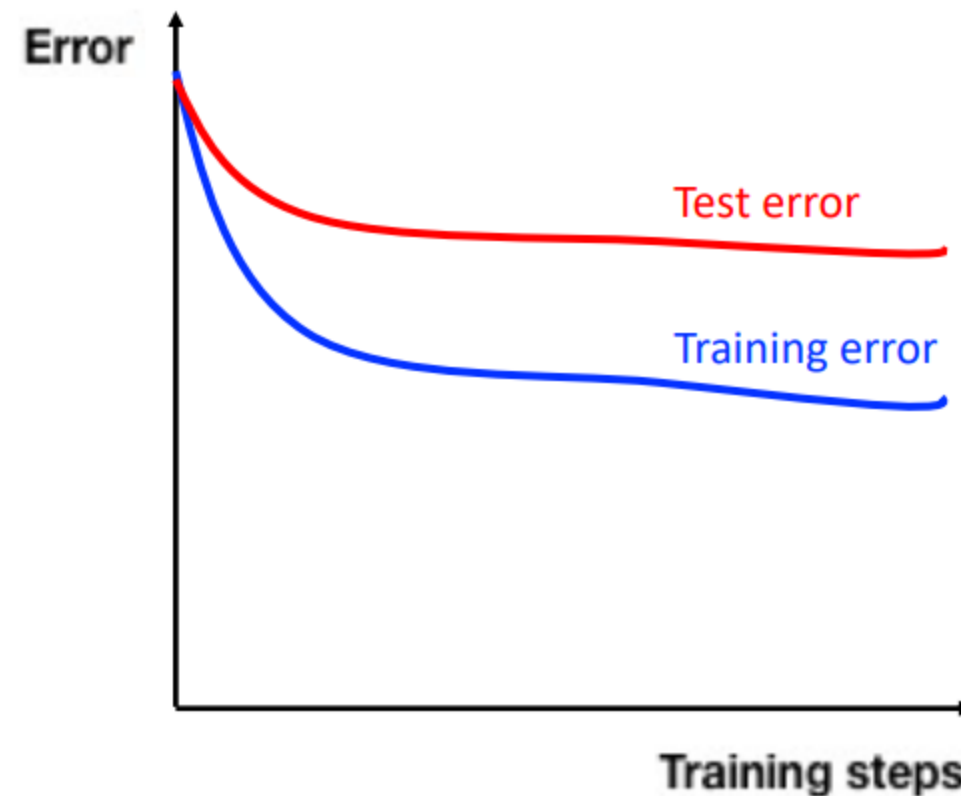
Underfits: too simple to explain the data!



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?

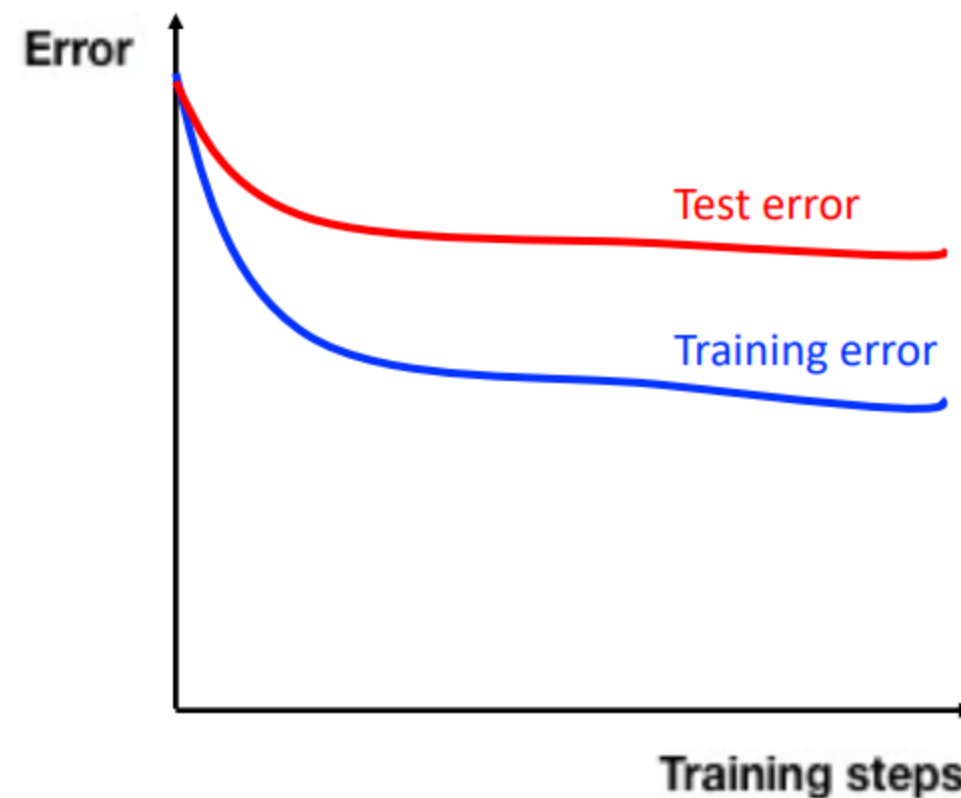
Error remains high!



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

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

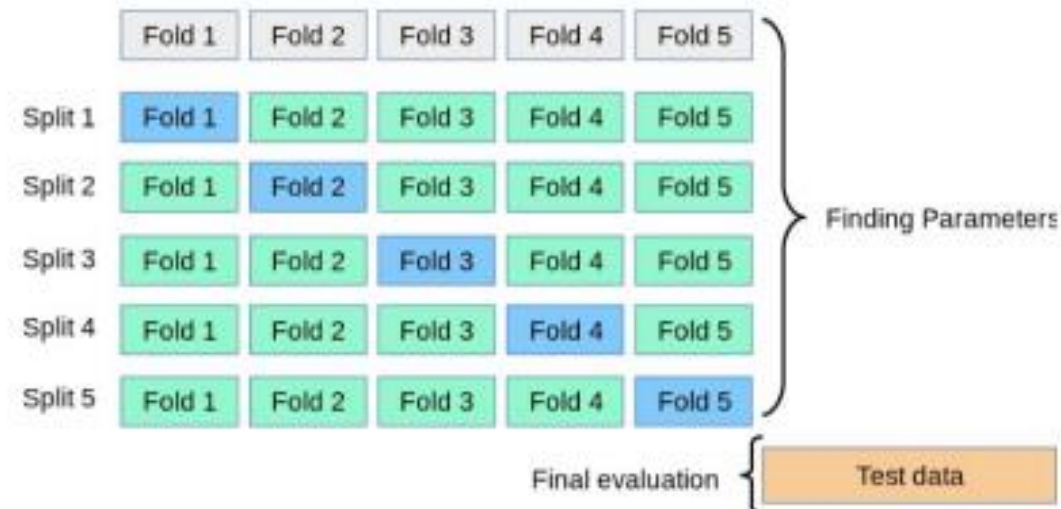
Key Challenge: how to select a model without repeatedly observing the test data (which leads to overfitting)?

Hyperparameter Tuning

- Split the training set so it can be 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

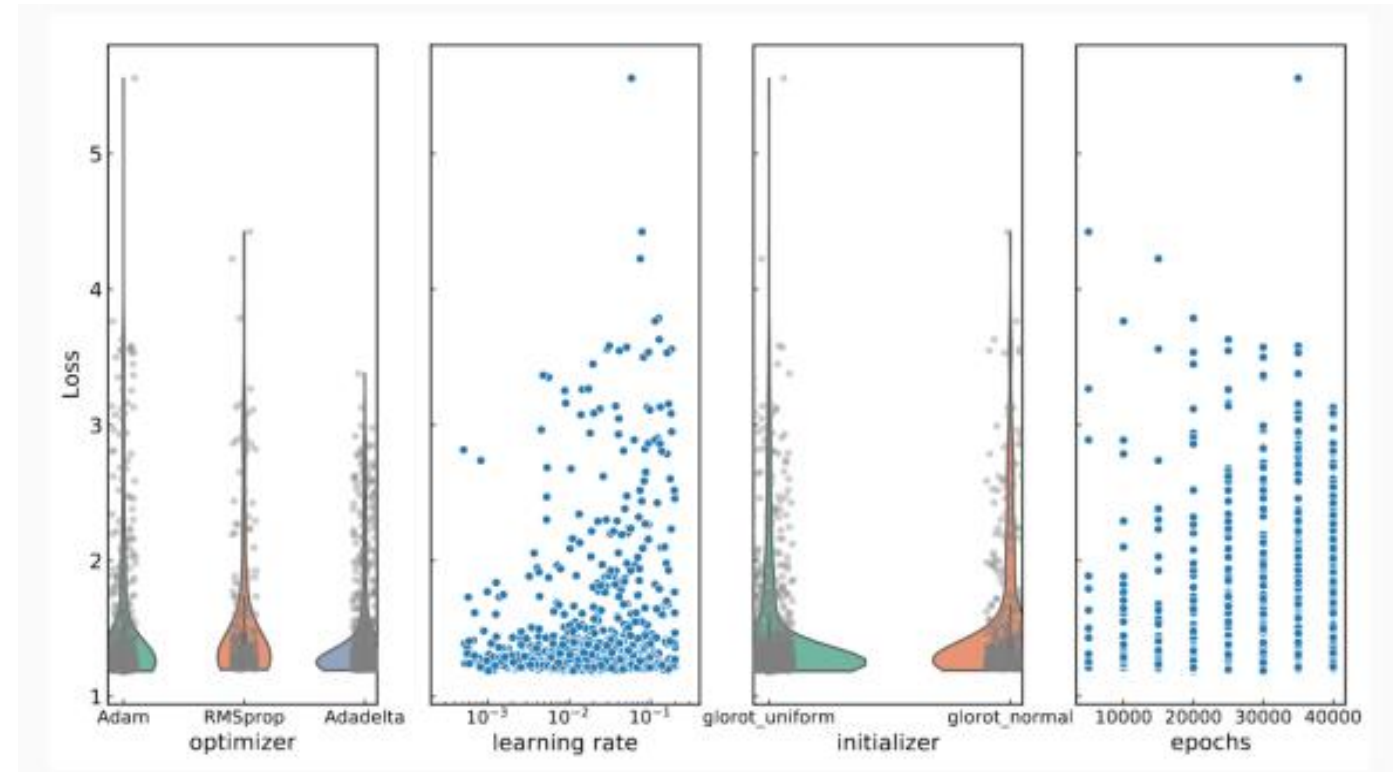
- So far we found the following hyperparameters:
- Model:
 - Network depth and width (number of layers, neurons per layer);
 - Type of activation functions (e.g., ReLU, Tanh).
- Training process:
 - Optimizer settings (learning rate, momentum, etc.);
 - Number of epochs and batch size;
 - Validation strategy (cross-validation or holdout).
- Dataset Considerations:
 - Data size (do we need more samples?);
 - Handling class imbalance.

Manual Hyperparameter Tuning

- Manual approach goal: achieve good performance on the test set
- Some examples of effect of hyperparameters 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.

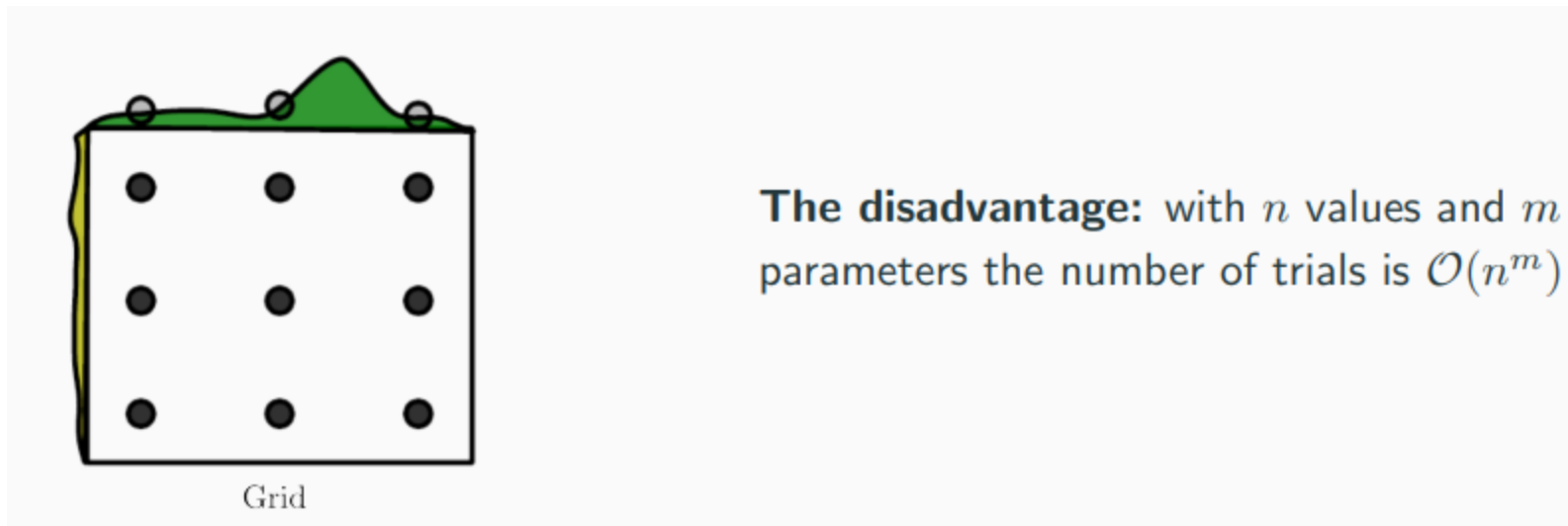
Automatic Hyperparameter Optimization Algorithms

- 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.



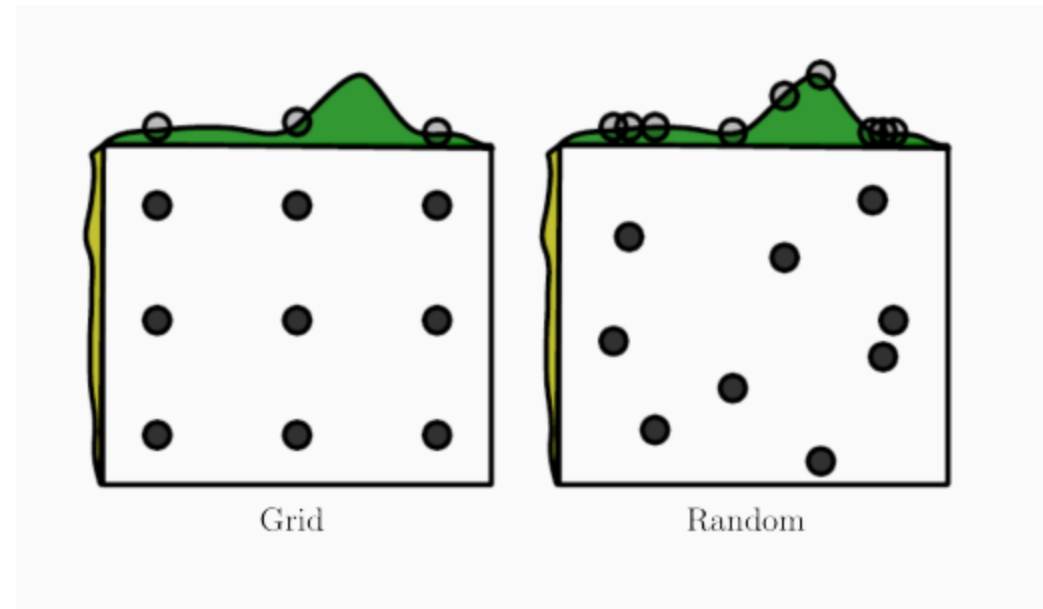
Grid Search

- 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.



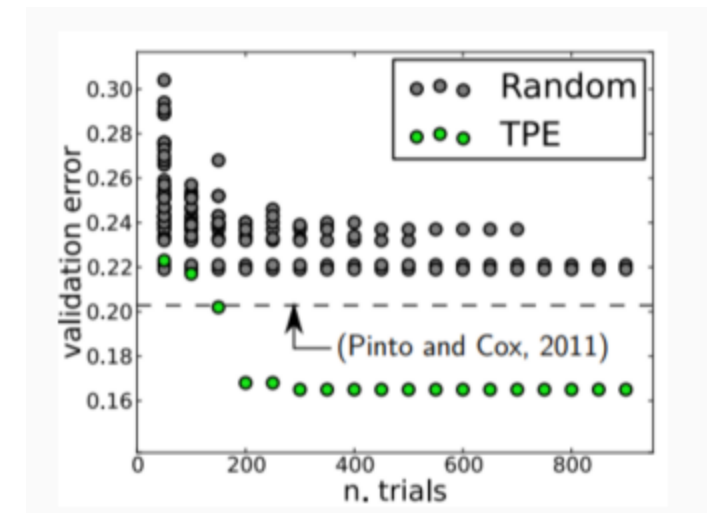
Random Search

- 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: Tree-structured Parzen Estimator (TPE)



SMBO minimizes functions $f : X \rightarrow \mathcal{R}$ 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 \bar{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**

$x^* \leftarrow \operatorname{argmin}_{\mathbf{x}} L(\mathbf{x}, \bar{f}_{i-1})$;

 Compute $f(x^*)$;

$H \leftarrow H \cup \{\mathbf{x}^*, f(\mathbf{x}^*)\}$;

 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.

- The Tree-structured Parzen Estimator (TPE) algorithm is a SMBO where the surrogate model is a probabilistic model $p(y|\mathbf{x})$, which chooses the next trial point by optimizing the Expected Improvement criterion:

$$\text{EI}_{y^*}(\mathbf{x}) = \int_{-\infty}^{\infty} \max(y^* - y, 0) p(y|\mathbf{x}) dy$$

which measures how much the loss function is expected to be lower than a threshold value y^* , chosen so that $p(y < y^*) = \gamma$ where γ is a parameter of the algorithm.

The $p(y|\mathbf{x})$ is computed via Bayes' theorem through $p(\mathbf{x}|y)$

$$p(\mathbf{x}|y) = l(\mathbf{x}) \quad \text{if } y < y^*; \quad g(\mathbf{x}) \quad \text{if } y \geq y^*$$

where $l(\mathbf{x})$ and $g(\mathbf{x})$ are probability distributions estimated by using the trials \mathbf{x}_i such that $f(\mathbf{x}_i)$ is respectively lower and higher or equal y^* .

The Expected Improvement for the TPE admits a closed form solution:

$$\text{EI}_{y^*}(\mathbf{x}) = \int_{-\infty}^{\infty} \max(y^* - y) \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} dy \propto \left(\gamma + \frac{g(\mathbf{x})}{l(\mathbf{x})} (1 - \gamma) \right)^{-1}$$

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.