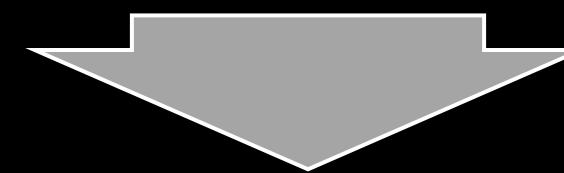


Introduction to Data Science and Machine Learning

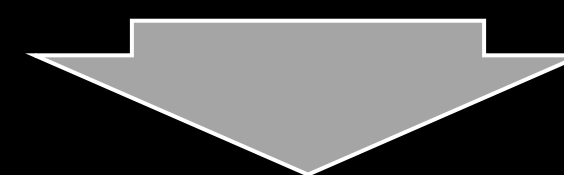
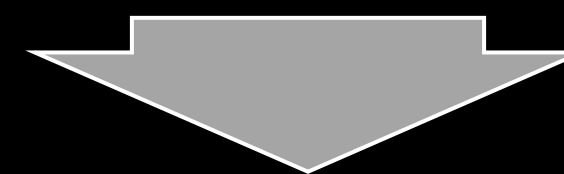
OVERFITTING AND MODEL EVALUATION

- **Review of Important Terms in Machine Learning**
- **Overfitting and Regularization**
- **Model Quality Criteria**
- **Introduction to Neural Networks**

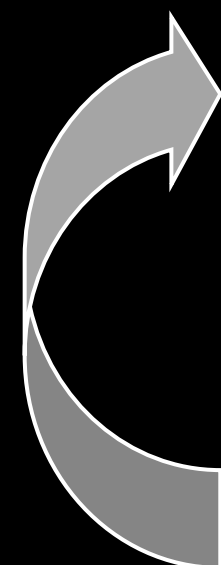
Selection of an ML Model



Splitting the Dataset (e.g. 70/20/10)



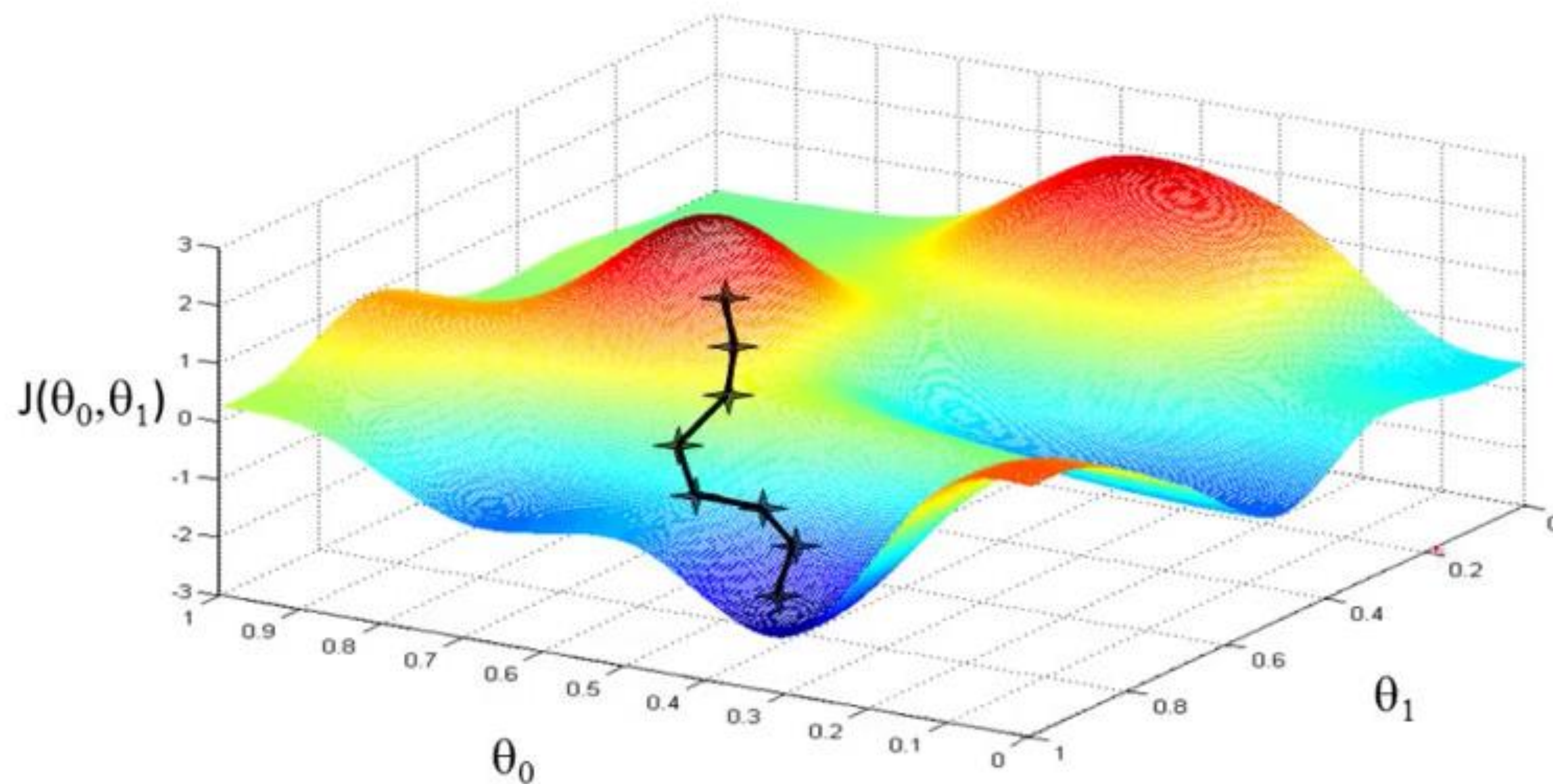
Changing the
hyperparameters
(model-centric
optimization)



Extension/
Improvement of
the dataset
(data-centric
optimization)



OPTIMIZATION FUNCTION (OPTIMIZER)



- **Iterative method (Gradient Descent) to find the minimum of the cost function.**
- **The learning rate ("learning parameter") describes the step size for approaching the minimum.**

OPTIMIZATION OF THE MODEL PARAMETERS

Forward Propagation

- Computation of predictions for a dataset using the current model parameters (weights and biases)
- Computation of the total cost or loss based on the difference between predictions and actual values

Backward Propagation

- Calculation of the contribution of each model parameter to the total loss
- Computation of gradients (i.e., partial derivatives of the loss with respect to parameter) to determine how to update the parameters to reduce the loss

Gradient Descent (Optimization Step)

- Use the derivatives to update the parameters
- Typical update rule:
$$\text{New Value} = \text{Old Value} - \text{Learning Rate} \times \text{Partial Derivative}$$

MODEL PARAMETERS VS. HYPER PARAMETERS

Model Parameters

- Parameters that are optimized during training (the weights and biases).

Hyper Parameters

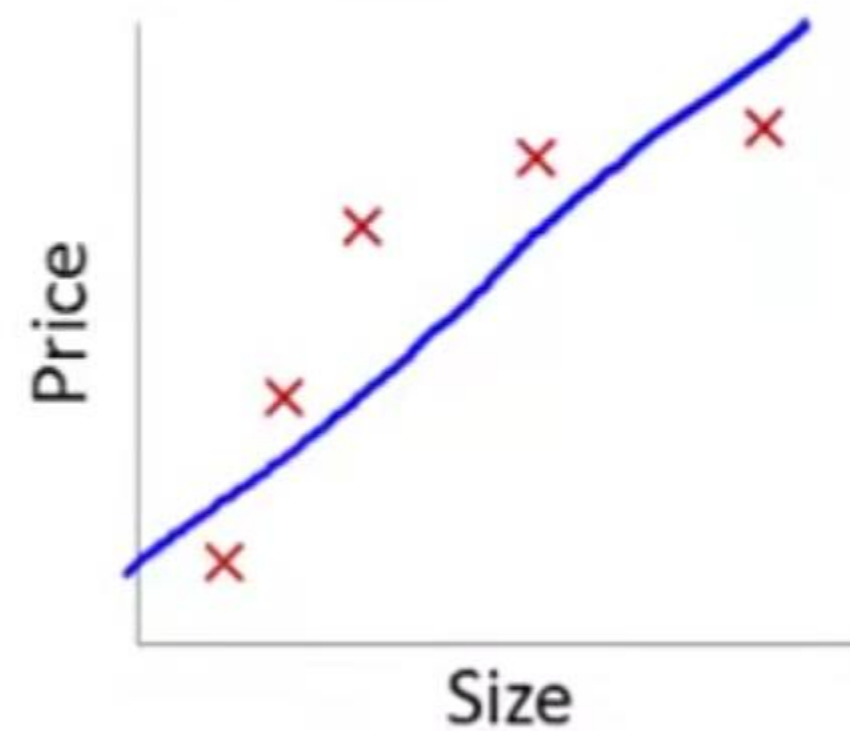
- Parameters set before training (e.g., on how the parameters are optimized, how many are optimized, or how they relate to each other).

FEATURES, LABELS, PARAMETER

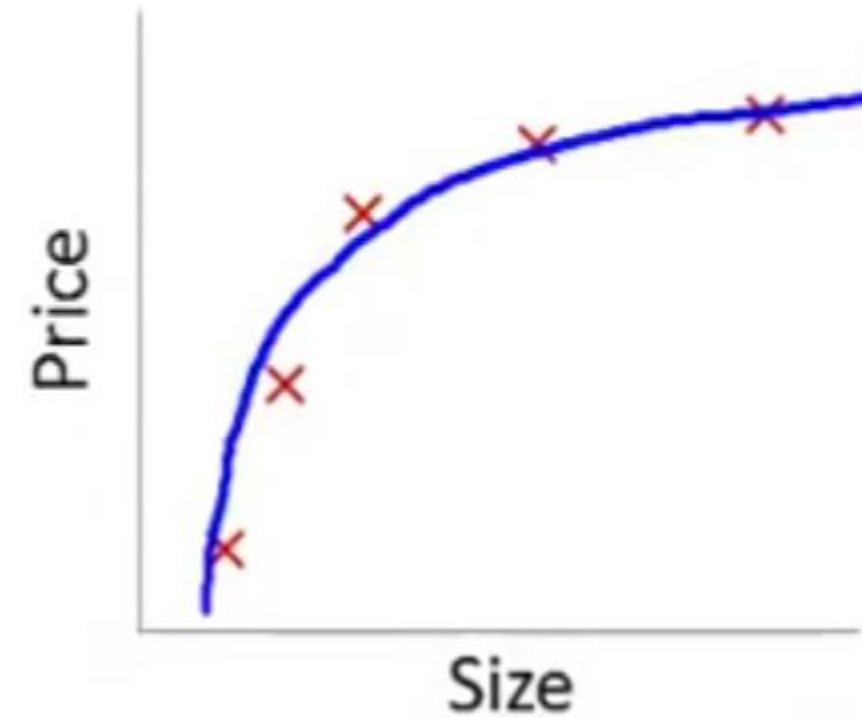
Machine Learning	Statistics	Description
Feature, Input Variable	Independent Variable	Input data used for prediction
Label, Target Variable, Output	Dependent Variable	Observed outcomes used to train the model
Weights, (Model) Parameter	Coefficients	Are optimized ("learned") during training

OVERFITTING

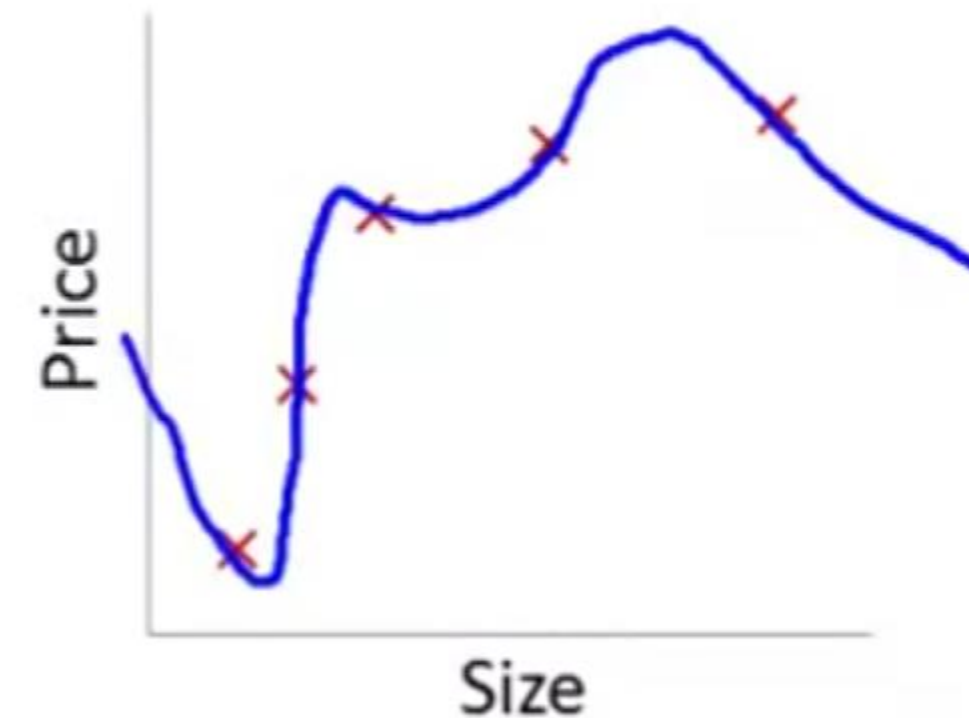
Example: Linear regression (housing prices)



$\rightarrow \theta_0 + \theta_1 x$
"Underfit" "High bias"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$
"Just right"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
"Overfit" "High variance"

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

STRATEGIES TO AVOID OVERFITTING

Options:

1. Reduce number of features.

→ — Manually select which features to keep.

→ — Model selection algorithm

2. Regularization.

→ — Keep all the features, but reduce magnitude/values of parameters θ_j .

— Works well when we have a lot of features, each of which contributes a bit to predicting y .

REGULARIZATION

"Penalizing" the Use of Variable Information within the Cost Function

Linear Model with Multiple Variables x_1, x_2 and possibly many others:

$$h_x(\theta_0, \theta_1, \theta_2, \dots) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$$

(with $\theta_0, \theta_1, \theta_2, \dots$ as the model parameters to be estimated)

Cost function with regularization:

$$J_x(\theta_0, \theta_1, \theta_2, \dots) = \frac{1}{m} \left[\sum_m (h_x(\theta_0, \theta_1, \theta_2, \dots) - y)^2 + \lambda (|\theta_0| + |\theta_1| + |\theta_2| + \dots) \right]$$

(with λ as regularization parameter)

EXAMPLE NOTEBOOK

The screenshot displays a Jupyter Notebook interface within a web browser. The browser's address bar shows the URL 'repo-template-intro-to-data-science-and-ml [Codespaces: fictional waddle]'. The notebook's tab bar includes 'overfitting.py U', 'overfitting.ipynb U X', 'interaction effects.py U', and 'interaction effect.ipynb U'. The active notebook, 'overfitting.ipynb', has a toolbar with options: '+ Code', '+ Markdown', 'Run All', 'Restart', 'Clear All Outputs', 'Variables', 'Outline', and a dropdown menu. The top right corner indicates the Python version 'Python 3.10.13' with edit, view, and delete icons. A left sidebar contains navigation icons: home, search, recent (10 items), view, file explorer, console, GitHub, Python, and a settings gear. The notebook content features a title 'Demonstrating Overfitting and the Importance of Feature Selection' with a dropdown arrow. Below the title is a paragraph explaining overfitting: 'This notebook illustrates the phenomenon of overfitting in statistical models, particularly in the context of regression analysis. Overfitting occurs when a model learns not only the underlying pattern but also the noise in the training data, leading to poor performance on new, unseen data. This is often a result of using excessively complex models or including irrelevant features in the model.' This is followed by another paragraph: 'We will explore how different regression models respond to a mix of relevant and irrelevant features and demonstrate the utility of techniques like Ridge Regression for mitigating overfitting.' A section header 'Import Libraries' precedes a code cell. The code cell, labeled '[1]' on the left and 'Python' on the right, contains the following Python code:

```
# Importing necessary libraries for data handling, mathematical operations, and plotting
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import mean_squared_error
```

 The bottom status bar shows 'Codespaces: fictional waddle', a file explorer icon, 'main*', a refresh icon, '53', a warning icon, '57', a speech bubble icon, '8', 'Cell 1 of 14', a grid icon, 'Layout: German', and a bell icon.

repo-template-intro-to-data-science-and-ml [Codespaces: fictional waddle]

overfitting.py U overfitting.ipynb U X interaction effects.py U interaction effect.ipynb U

overfitting.ipynb

+ Code + Markdown Run All Restart Clear All Outputs Variables Outline ...

Python 3.10.13

Demonstrating Overfitting and the Importance of Feature Selection

This notebook illustrates the phenomenon of overfitting in statistical models, particularly in the context of regression analysis. Overfitting occurs when a model learns not only the underlying pattern but also the noise in the training data, leading to poor performance on new, unseen data. This is often a result of using excessively complex models or including irrelevant features in the model.

We will explore how different regression models respond to a mix of relevant and irrelevant features and demonstrate the utility of techniques like Ridge Regression for mitigating overfitting.

Import Libraries

```
# Importing necessary libraries for data handling, mathematical operations, and plotting
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import mean_squared_error
```

[1] Python

Codespaces: fictional waddle main* 53 57 8 Cell 1 of 14 Layout: German

BATCHES, STEPS AND EPOCHS

Batch

- The entire set of training data is divided into separate subgroups of equal size.
- The standard batch size in TensorFlow is 32.

Step

- A single iteration of gradient descent performed on one batch of data, during which all model weights are updated once.

Epoche

- Optimization of the model using the complete training data:
Number of Steps × Batch Size = Training Sample Size
- Depending on the model, very few epochs may suffice, or several hundred or thousand may be needed for optimization.

MODEL QUALITY CRITERIA FOR REGRESSION TASKS

errors: **forecast – actual** **(also: residuals)**

mae: **mean(abs(errors))**

mape: **mean(abs(errors/actual))**

mse: **mean(errors²)**

rmse: **sqrt(mean(errors²))**

rse: **sum(errors²) / sum((actual-mean(actual))²)**

r² = **1 – rse**

Video (3 minutes) with explanations of the criteria:

<https://www.coursera.org/lecture/machine-learning-with-python/evaluation-metrics-in-regression-models-5SxtZ>

MODEL QUALITY CRITERIA FOR CLASSIFICATION TASKS

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

$$\text{Precision} = \frac{\text{Number of Correct Positive Predictions}}{\text{Total Number of Predictions}}$$

e.g., for spam
detection

$$\text{Recall} = \frac{\text{Number of Correct Positive Predictions}}{\text{Number of Actual Positive Cases}}$$

e.g., credit card
fraud detection

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Actual Observations			
		<i>positive</i>	<i>negative</i>
Predictions	<i>positive</i>	true	false
	<i>negative</i>	false	true

Blog with more explanations on the above metrics:

<https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262/>

BREAKOUT

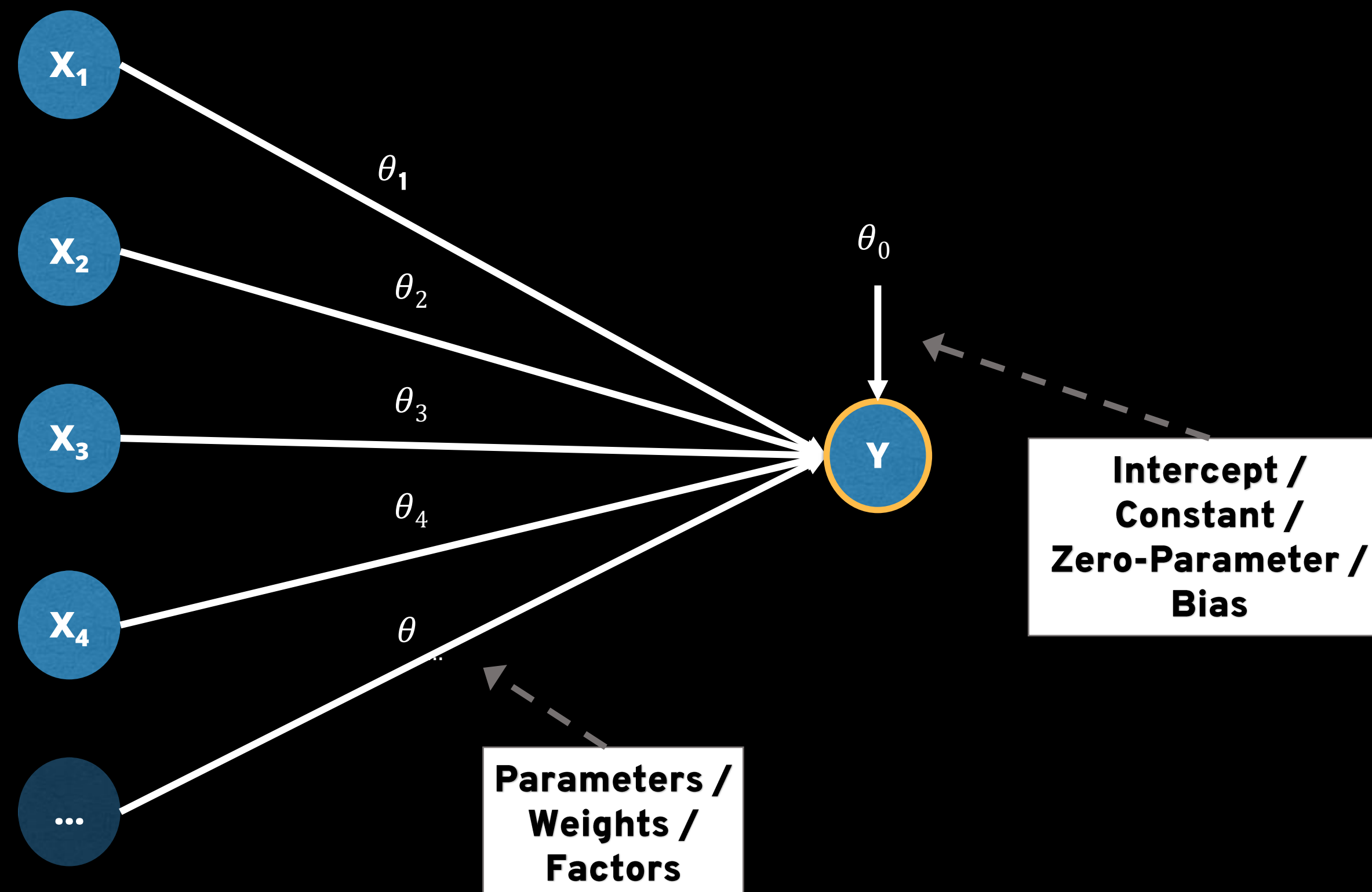
Each team writes a post in Mattermost with answers to the following questions:

- **What is the r^2 of your currently best model?**
- **Have you tested a model on Kaggle?**
→ **If “No”: What problem are you currently facing?**

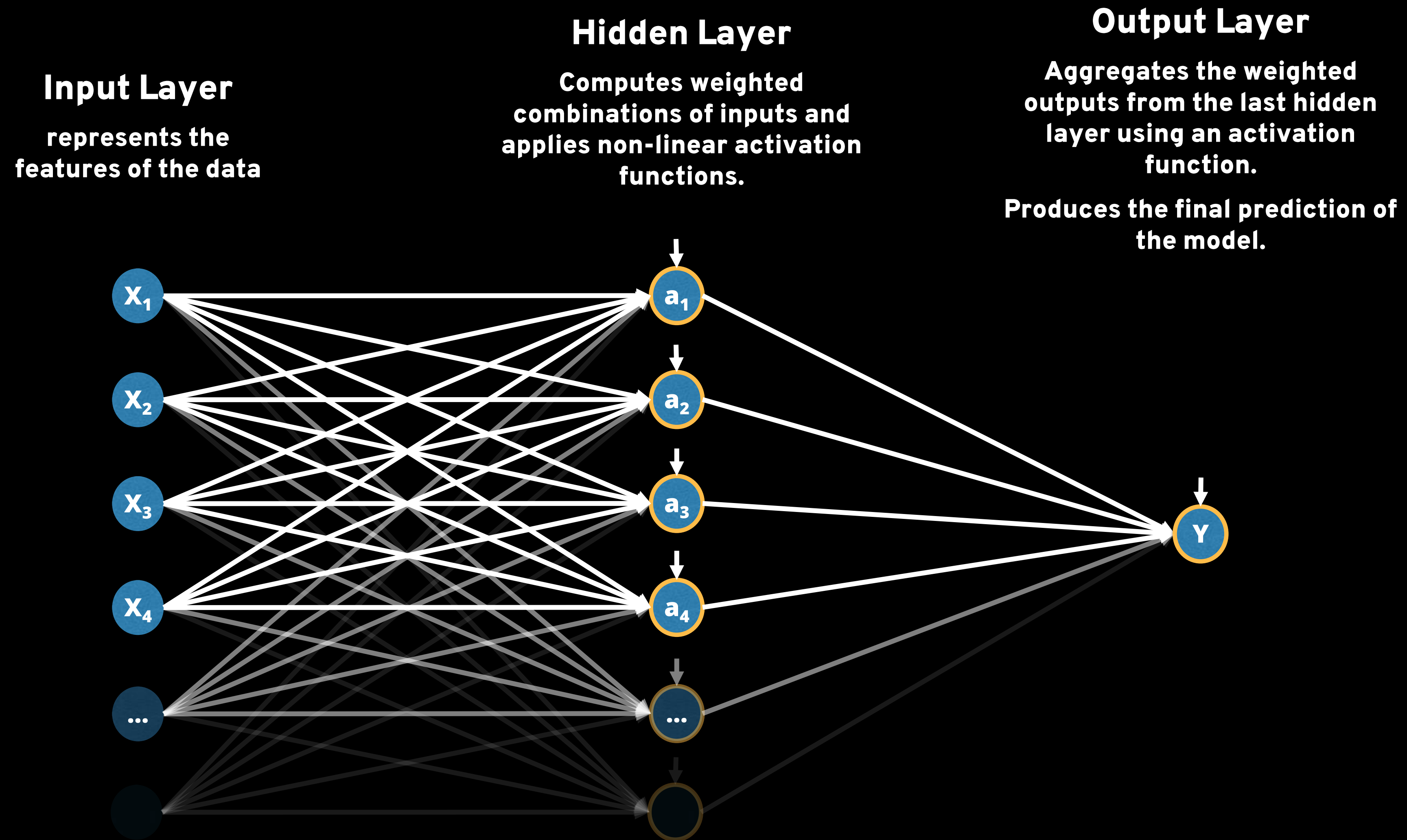
LINEAR REGRESSION

Input Layer
represents the features of the data
– each input node corresponds to
one variable (feature or dimension).

Output Layer
Applies an activation function (in this case, a
linear function) to aggregate the weighted
inputs (parameters θ) from the previous layer.



NEURAL NETS



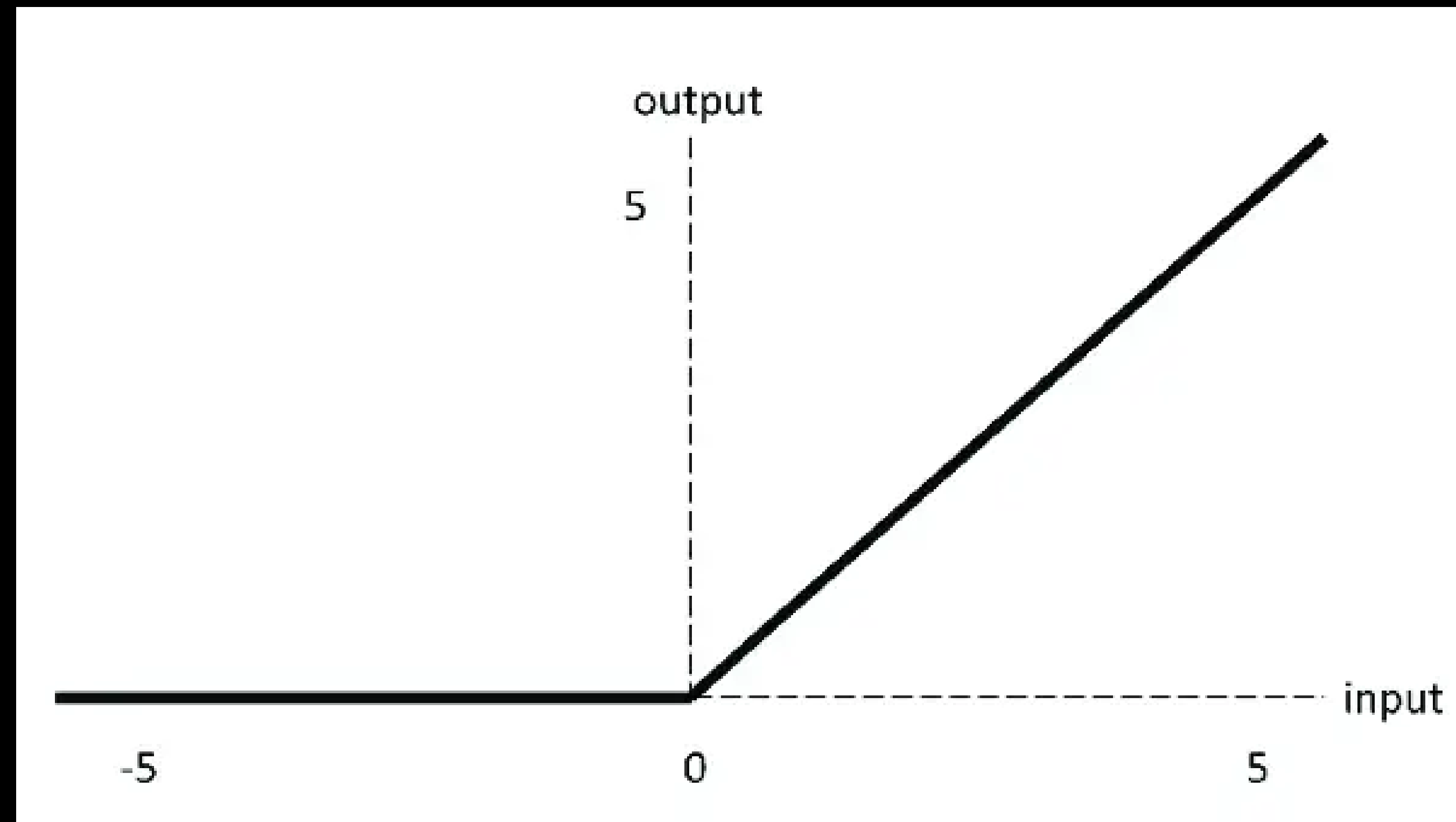
PROPERTIES OF THE LINEAR MODEL

- **Both with and without regularization, parameter optimization for the linear model is very easy and fast.**
 - **For simple models, it is easy to obtain optimized parameters.**
- **Models that are easier to optimize typically rely on stronger assumptions about the relationships between variables (in this case, linear relationships).**
 - **Therefore, optimal selection and encoding of variables according to these assumptions becomes even more important.**
 - **In some cases, the actual relationships may not be captured by the model.**

PROPERTIES OF NEURAL NETS

- **Definition of additional "hidden layers" between input and output layer.**
 - **Statistics: Definition of latent variables, usually with unknown meaning**
 - **Allows implicit modelling of interaction effects**
- **Use of non-linear activation functions.**
 - **Allows modeling of non-linear effects**

THE NON-LINEAR ACTIVATION FUNCTION RELU



$$f(x) = \max(0, x)$$



Epoch
000,000

Learning rate

0.03

Activation

ReLU

Regularization

None

Regularization rate

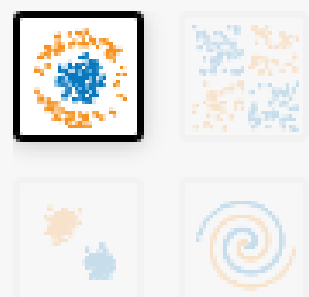
0

Problem type

Classification

DATA

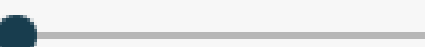
Which dataset do you want to use?



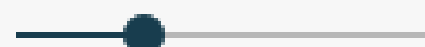
Ratio of training to test data: 50%



Noise: 0



Batch size: 10



REGENERATE

FEATURES

Which properties do you want to feed in?

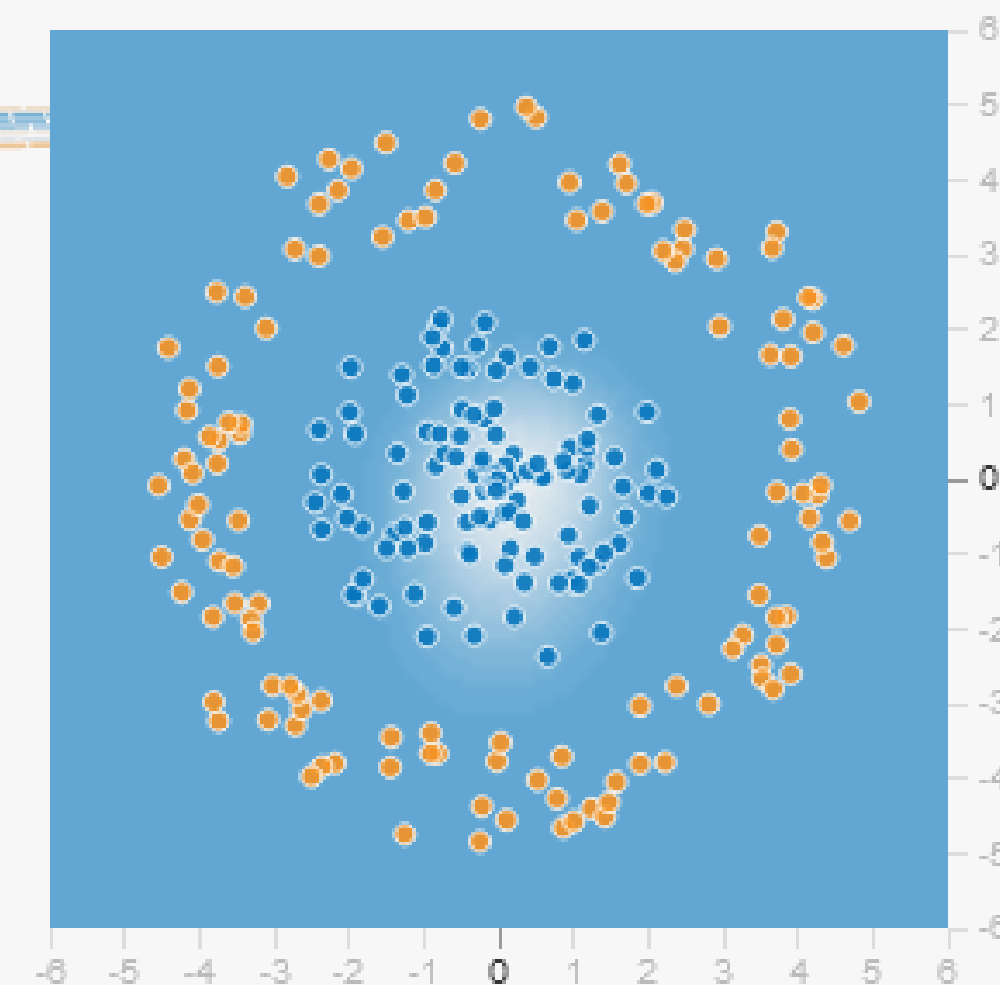
- X1
- X2
- X12
- X22
- X1X2
- $\sin(X1)$
- $\sin(X2)$

+ - 0 HIDDEN LAYERS

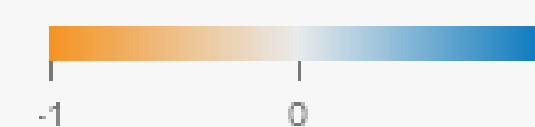
OUTPUT

Test loss 1.068

Training loss 1.082



Colors shows data, neuron and weight values.



☐ Show test data

☐ Discretize output

BREAKOUT

- **Open the following tool: <https://playground.tensorflow.org/>**
 - **Define two hidden layers and try changing the number of neurons so that you can predict the spiral-shaped dataset.**
- **Have you found a systematic approach to arrive at a solution?**
- **To what extent can you interpret the result in terms of the features used?**

HYPERPARAMETERS

- **Choice of model or model architecture**
- **Choice of activation functions**
- **Choice of loss function**
 - **MAE, MSE, ...**
- **Choice of optimization function**
 - **Learning rate size**
- **Choice of batch size**
- **Depending on the model architecture and selected components, many more...**

LEARNING RESOURCES

- Enroll into the Coursera course “[Advanced Learning Algorithms](#)” and watch the videos of the sections “Neural networks intuition” and “TensorFlow implementation”

Advanced Learning Algorithms

Limited access ⓘ

Course Material

Week 1

Week 2

Week 3

Week 4

Grades

Notes

Messages

Course Info

Like this course? Become an expert by joining the [Machine Learning Specialization](#).

Neural Networks

1h of videos left 2 min of readings left 5 graded assessments left

This week, you'll learn about neural networks and how to use them for classification tasks. You'll use the TensorFlow framework to build a neural network with just a few lines of code. Then, dive deeper by learning how to code up your own neural...

Show Learning Objectives

Neural networks intuition

Practice quiz: Neural networks intuition

1 graded assessment left

Neural network model

Practice quiz: Neural network model

1 graded assessment left

TensorFlow implementation

Practice quiz: TensorFlow implementation

1 graded assessment left

Upgrade to this Specialization

- Full access to all courses in this Specialization
- Graded assessments to measure your progress
- A certificate to feature on your resume
- €43/mo after a free trial, cancel anytime



Siddhant S.

"I got into two summer internships and received a job offer for a data scientist role by sharing my Coursera certificates."

Start 7-day free trial



TASKS

- **Further extend the dataset with additional variables that could be relevant for estimating revenue.**
- **Test your model's predictive performance on Kaggle!**