# Introduction to Machine Learning: Part II

Prof. Sean Dobbs[1] & Daniel Lersch[2]

April 8, 2021

[1] (sdobbs@fsu.edu)
[2] (dlersch@jlab.org)

# About this Lecture

- **Part I:** (Covered by Prof. Dobbs)
  - Basic concepts of machine learning
    (with focus on feedforward neural networks)
  - Data manipulation and visualization with pandas dataframes
  - Training a neural network with scikit
- **Part II:** (Today)
  - Overfitting and validation data
  - Gaussian processes

**The individual contents might be subject to change**

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

... introduces a few machine learning algorithms

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

... introduces a few machine learning algorithms

... aims to familiarize with machine learning jargon / vocabulary

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

... introduces a few machine learning algorithms

... aims to familiarize with machine learning jargon / vocabulary

... does NOT cover all aspects of machine learning (further reading required)

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

... introduces a few machine learning algorithms

... aims to familiarize with machine learning jargon / vocabulary

... does NOT cover all aspects of machine learning (further reading required)

... will NOT turn you into a machine learning specialist

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

... introduces a few machine learning algorithms

... aims to familiarize with machine learning jargon / vocabulary

... does NOT cover all aspects of machine learning (further reading required)

... will NOT turn you into a machine learning specialist

... was held last year in a different format $\rightarrow$ revised material for this edition

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

... introduces a few machine learning algorithms

... aims to familiarize with machine learning jargon / vocabulary

... does NOT cover all aspects of machine learning (further reading required)

... will NOT turn you into a machine learning specialist

... was held last year in a different format $\rightarrow$ revised material for this edition

... mainly utilizes the **scikit-learn** library

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

... introduces a few machine learning algorithms

... aims to familiarize with machine learning jargon / vocabulary

... does NOT cover all aspects of machine learning (further reading required)

... will NOT turn you into a machine learning specialist

... was held last year in a different format $\rightarrow$ revised material for this edition

... mainly utilizes the **scikit-learn** library

... uses **repl.it** for the hands-on sessions

# This Lecture...

... focusses on the basic concepts and ideas behind machine learning

... introduces a few machine learning algorithms

... aims to familiarize with machine learning jargon / vocabulary

... does NOT cover all aspects of machine learning (further reading required)

... will NOT turn you into a machine learning specialist

... was held last year in a different format $\rightarrow$ revised material for this edition

... mainly utilizes the **scikit-learn** library

... uses **repl.it** for the hands-on sessions

... most likely contain several errors ($\rightarrow$ Please send a mail to `dlersch@jlab.org`)

# Homework and Literature

- Machine learning can be learned best by simply doing it!

# Homework and Literature

- Machine learning can be learned best by simply doing it!
- Homework aims to perform a simple analysis and getting familiar with machine learning

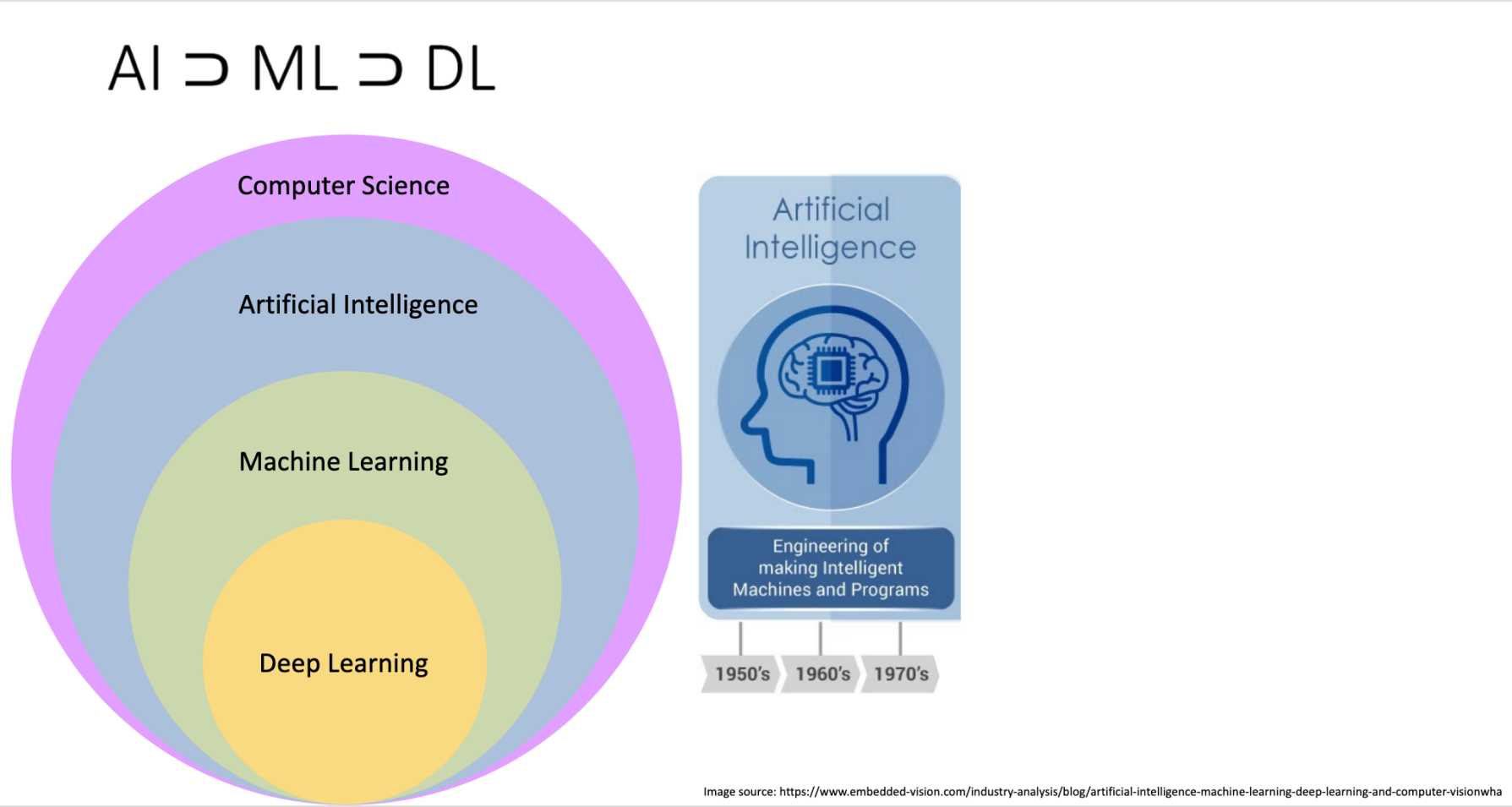
# Homework and Literature

- Machine learning can be learned best by simply doing it!
- Homework aims to perform a simple analysis and getting familiar with machine learning
- Helpful literature:
  - The **scikit-learn** documentation
  - Talks from
    - ★ The **deep learning for science school 2020**
    - ★ The **deep learning for science school 2019**[3]
  - **Distill.pub** (many articles about state-of-the-art machine / deep learning)
  - "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow", by Aurélien Géron
  - The internet is full of good (but also very bad!) literature[4] $\rightarrow$ browse with caution
  - Slides and scripts available at:
    `http://hadron.physics.fsu.edu/~dlersch/Intro_To_ML_2021/`

[3] Very good and detailed explanation of (deep) neural networks

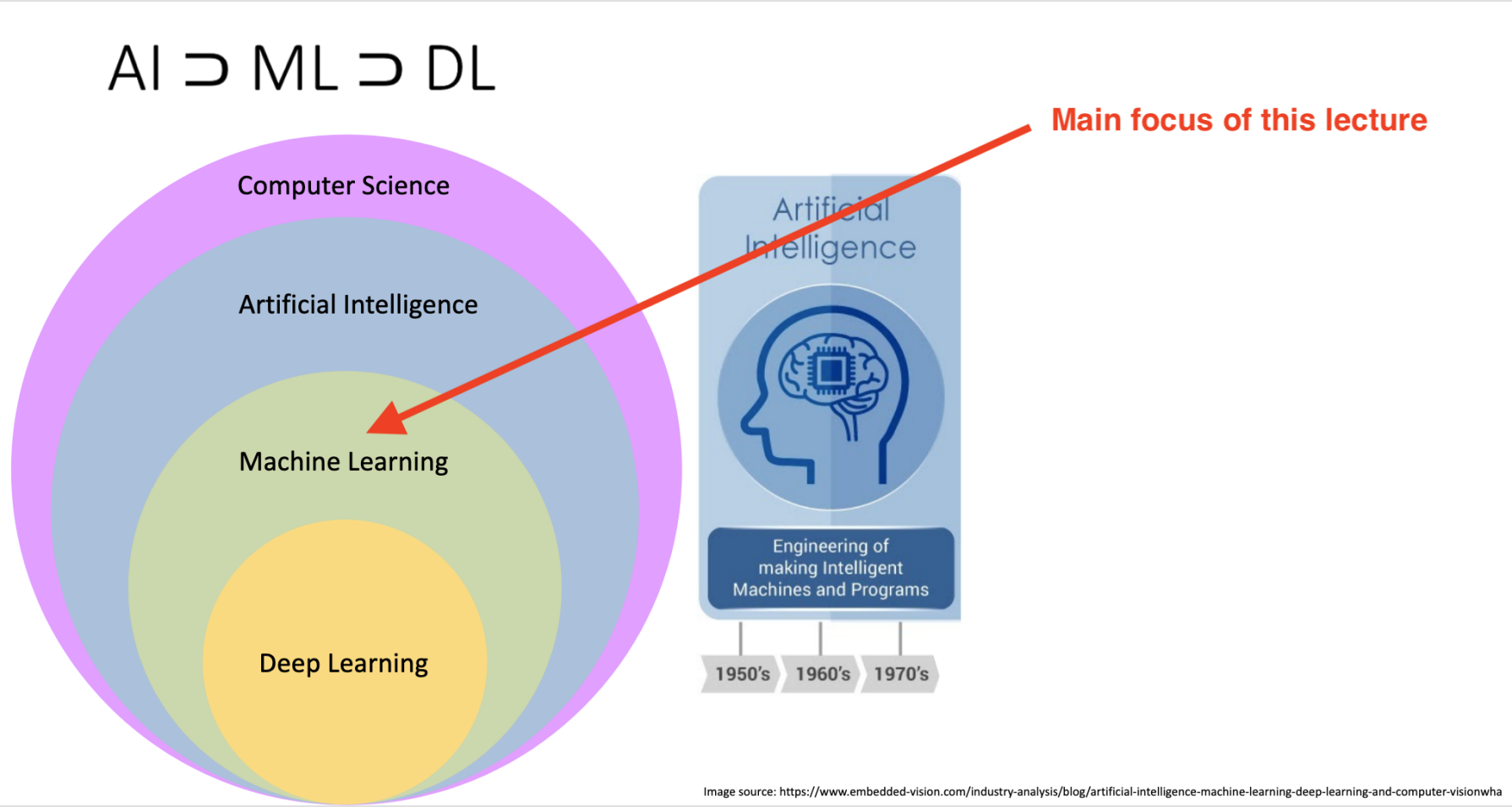
[4] Any document claiming that there is a quick way to understand machine learning without any theory / math is considered as bad

# AI, ML and DL



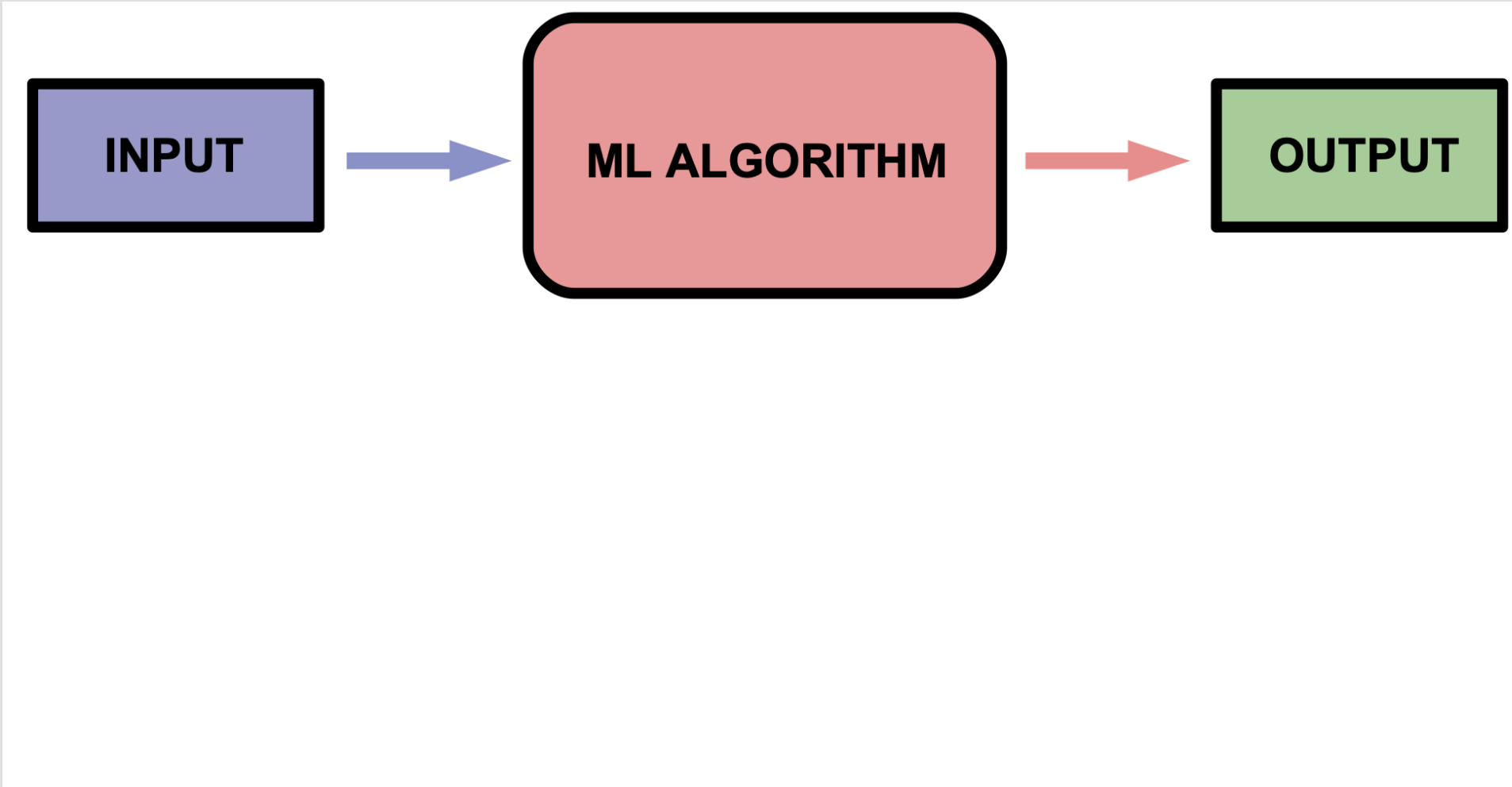

Slide taken from Brenda Ngs introductory talk at the: **deep learning for science school 2019**

# AI, ML and DL

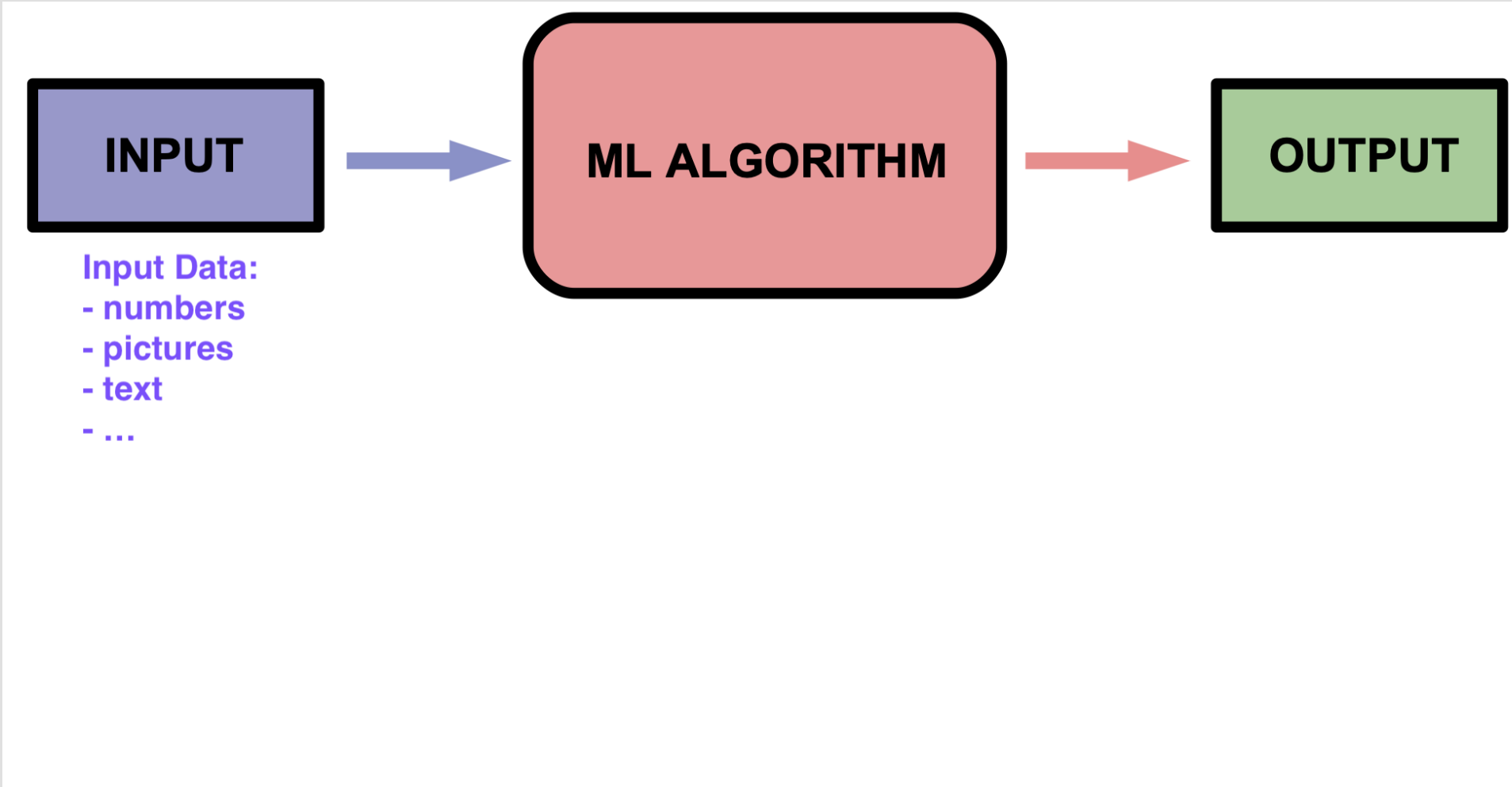



Slide taken from Brenda Ngs introductory talk at the: **deep learning for science school 2019**

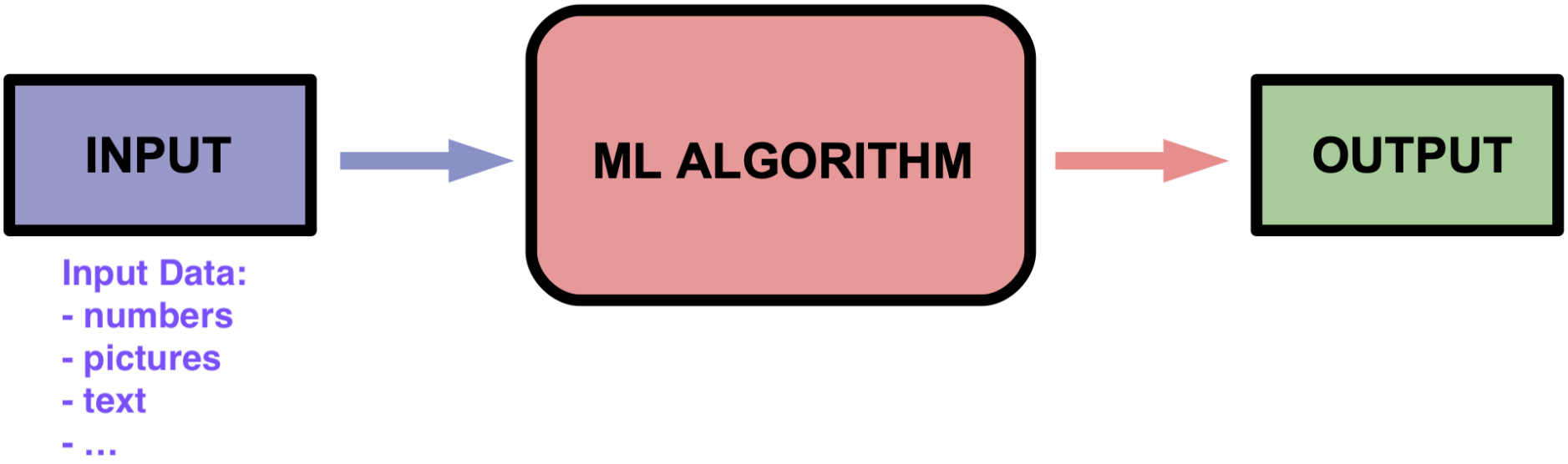
# Basic Components of Machine Learning



# Basic Components of Machine Learning

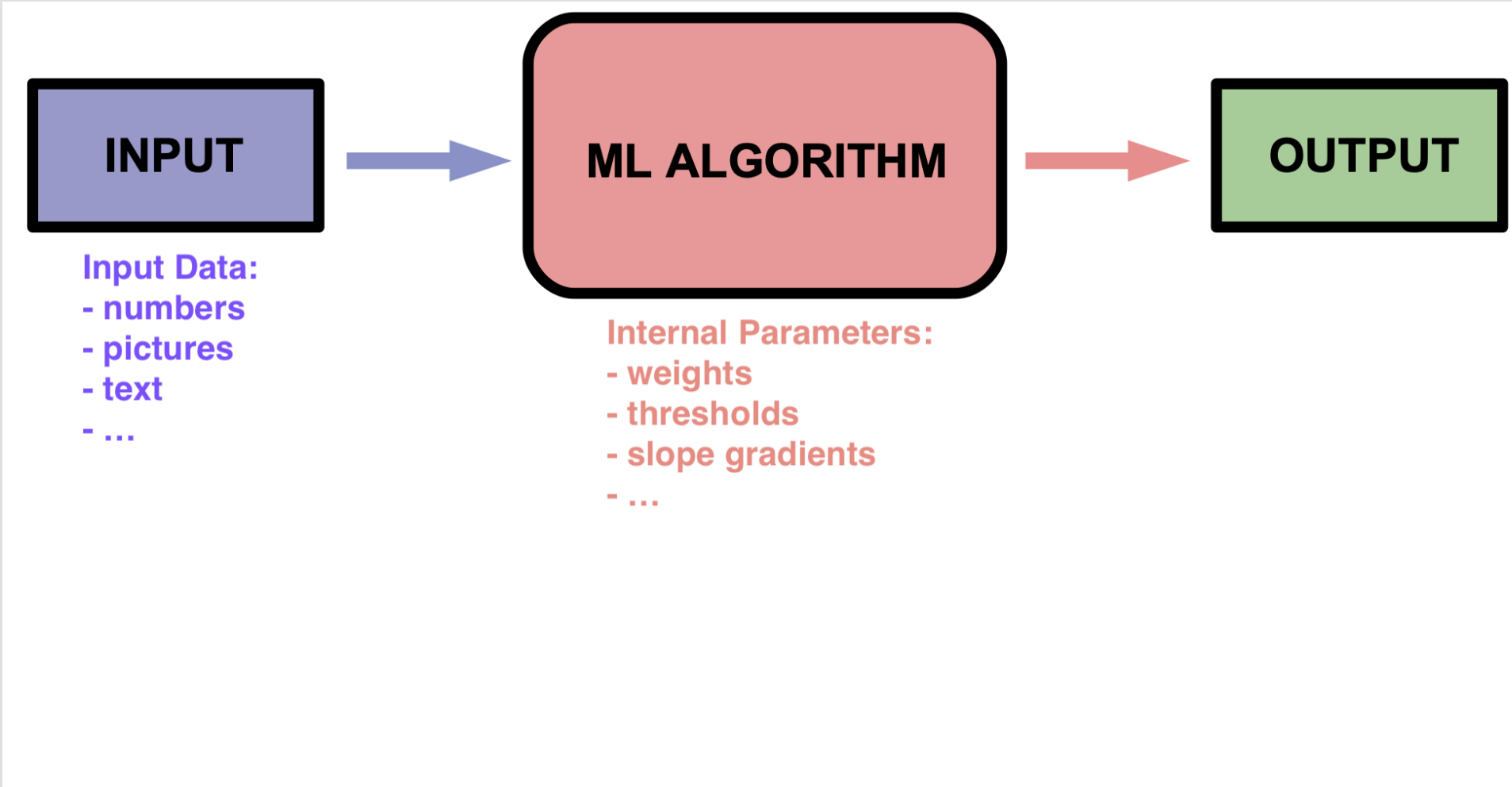


# Basic Components of Machine Learning

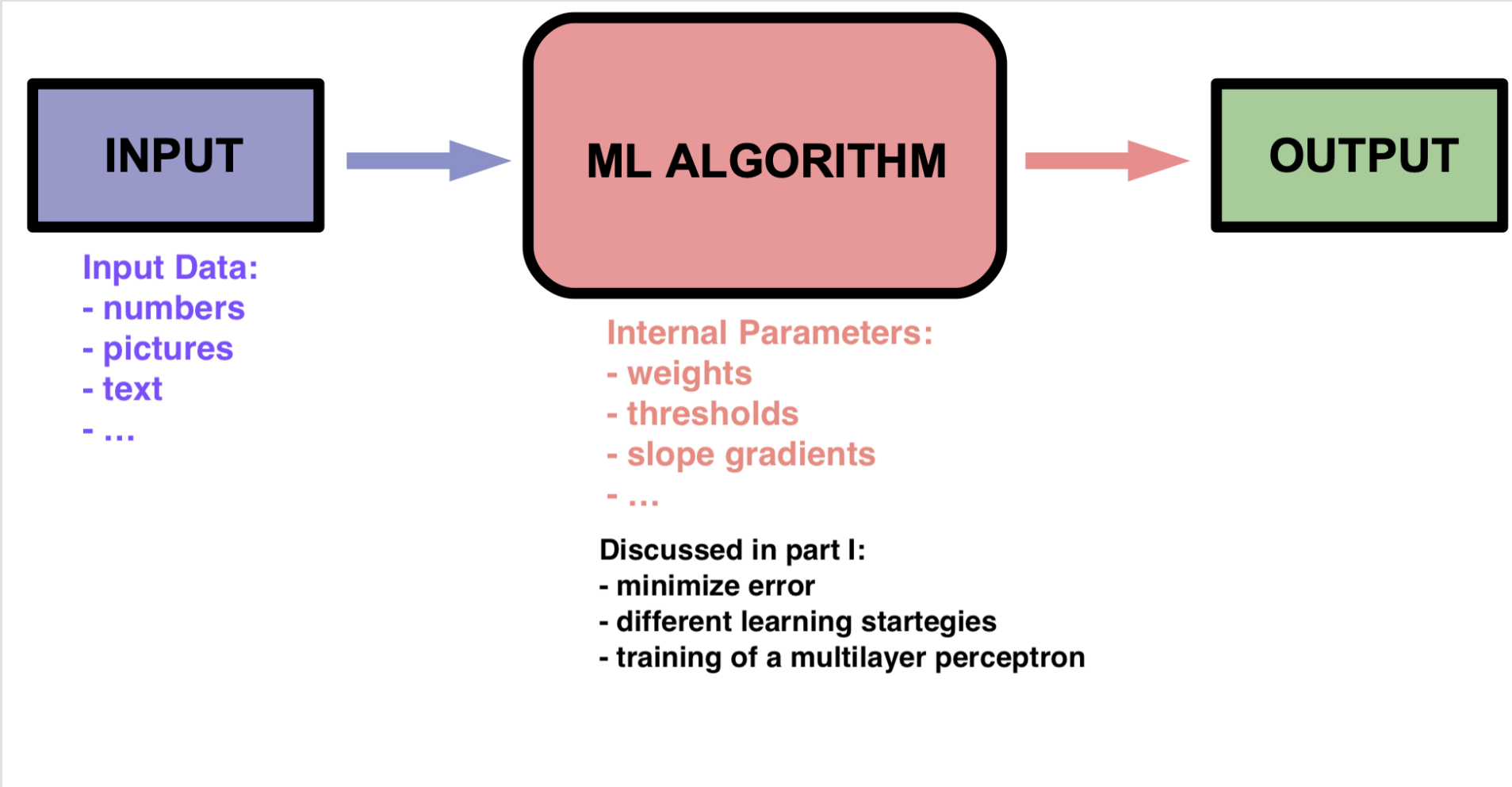



**Introduced in part I: DataFrames —> handle and manipulate data**

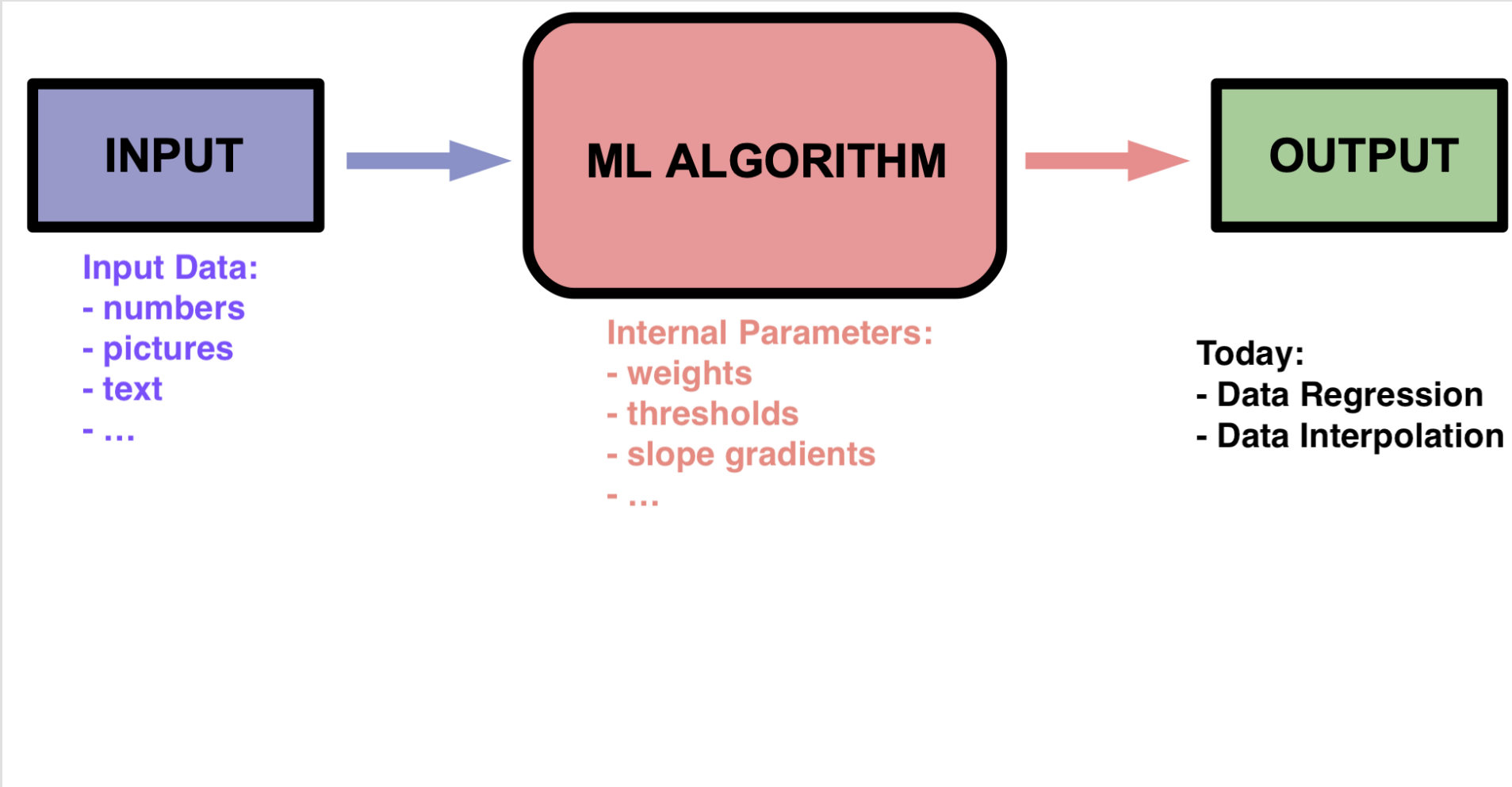
# Basic Components of Machine Learning



# Basic Components of Machine Learning



# Basic Components of Machine Learning

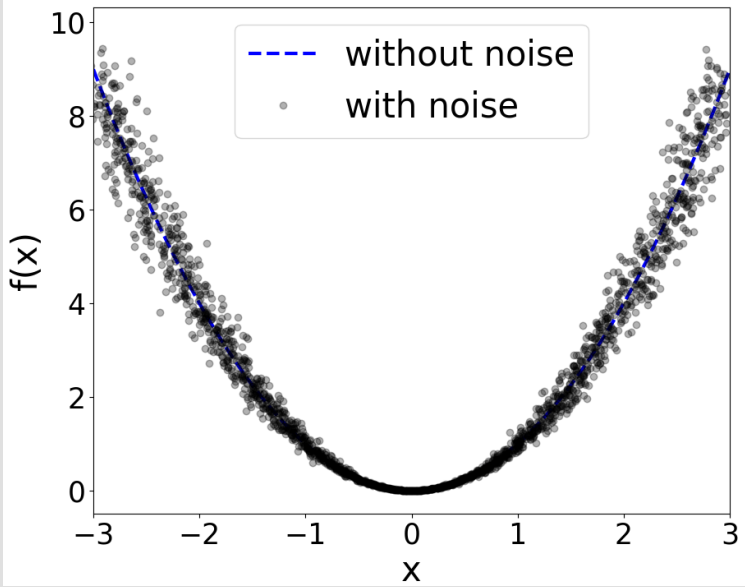


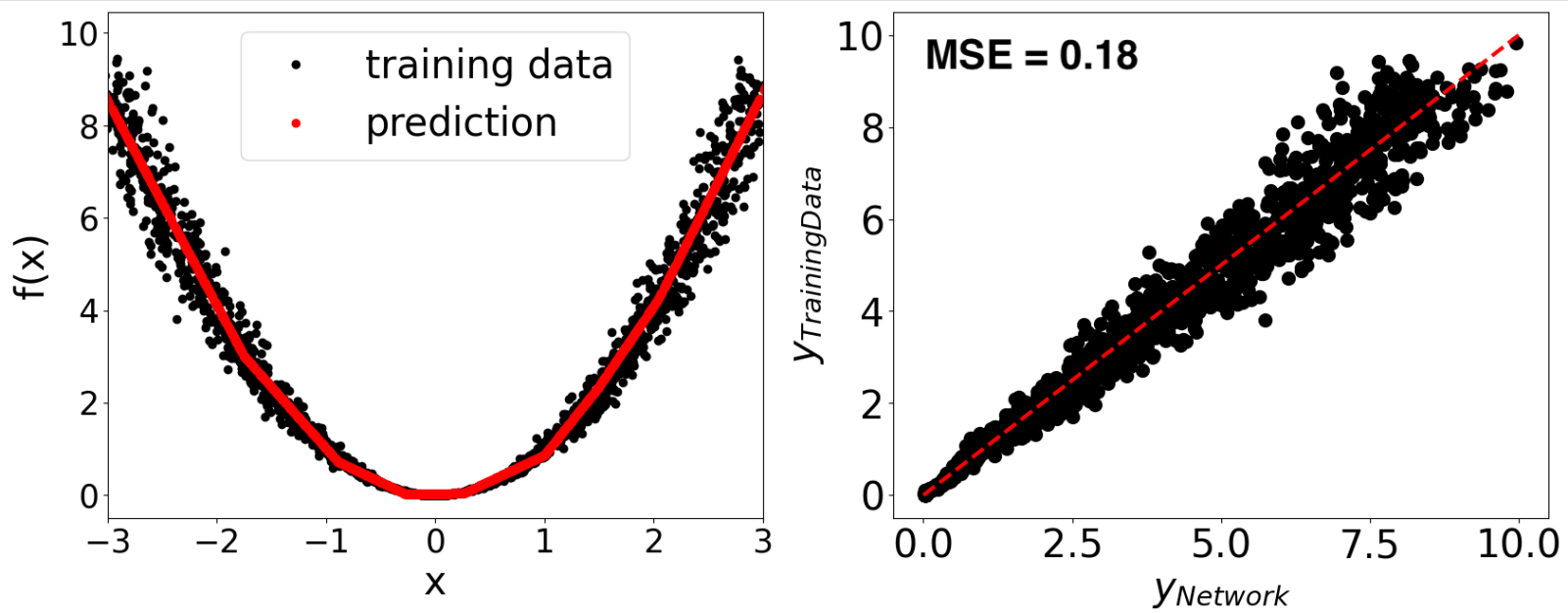
# Fitting a noisy quadratic Function with a Neural Network

- Already discussed in part1
- Now add noise to the data:

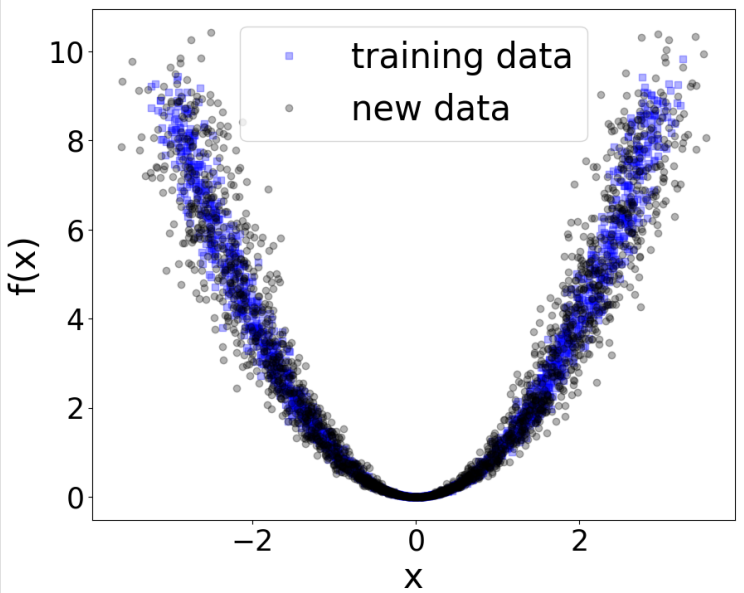
  $x \mapsto x \cdot \text{Gauss}(1, 5\%)$

  $f(x) \mapsto f(x) \cdot \text{Gauss}(1, 5\%)$
- Use neural network from part1 to fit the data
- Judge quality of fit with:
  Mean Squared Error (MSE) =
  $\frac{1}{N} \sum [y_{Data} - y_{Network}]^2$

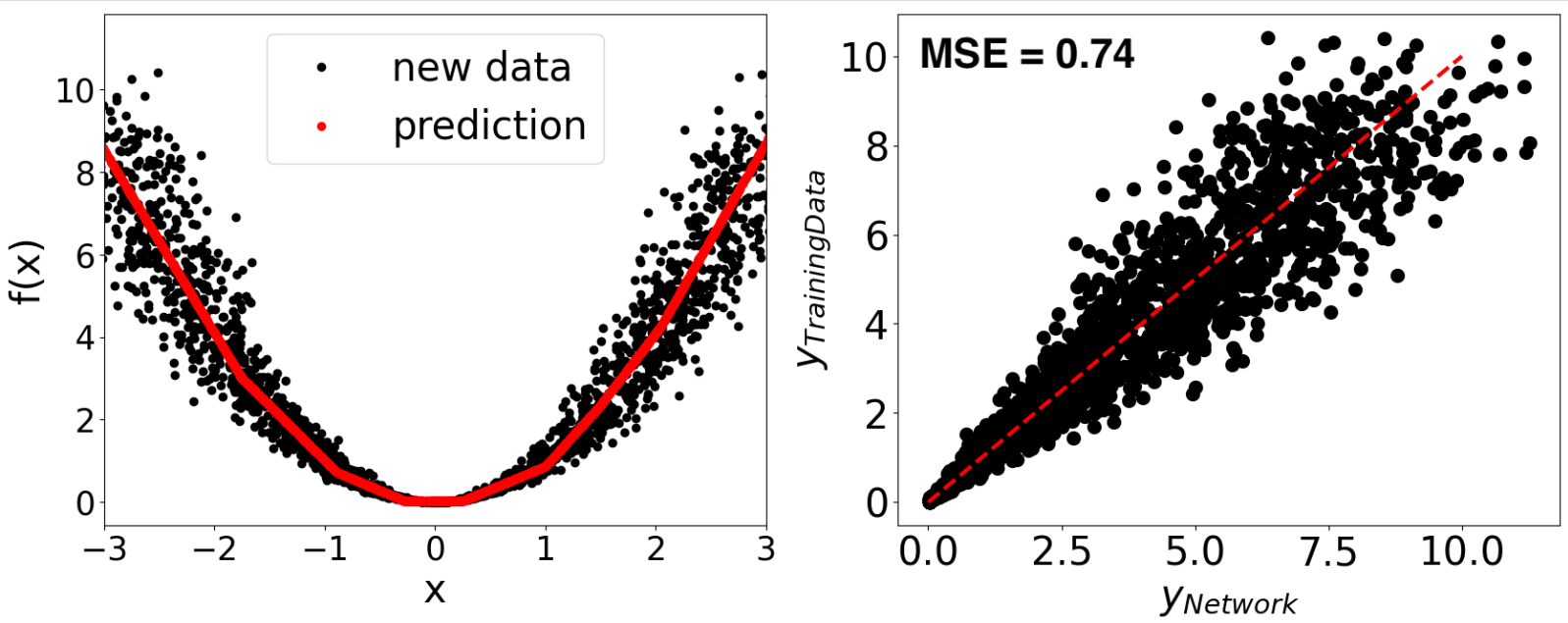


# Feeding "new" Data into the Network




- Look again data with $f(x) \approx x^2$
- Noise is 2 times higher than in training data
- Feed this data into trained network
- Predictive performance worse
  → not surprising, network only "knows" what it has been trained to!



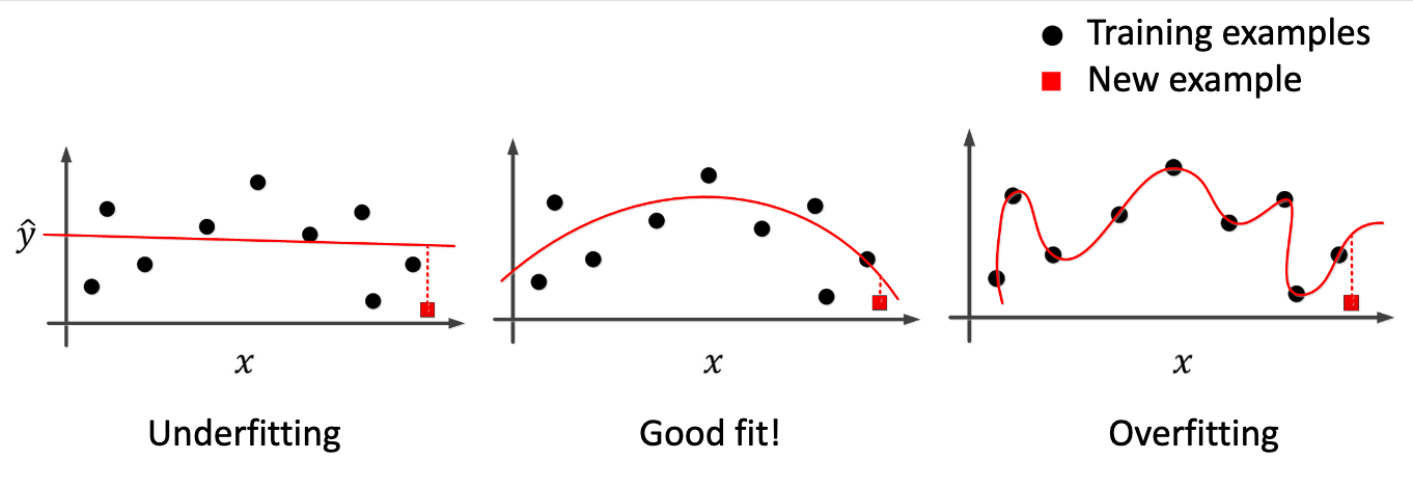
# Validation Data

- Want to enable network to abstract / generalize on unknown data AND avoid overfitting (i.e. avoid that network reproduces features from training data only)


Picture taken from Brenda Ngs introductory talk at the: deep learning for science school 2019

# Validation Data

- Want to enable network to abstract / generalize on unknown data AND avoid overfitting (i.e. avoid that network reproduces features from training data only)
- **Validation Data:** Part of training data that is NOT used to update internal parameters[5], but used to determine when training is complete

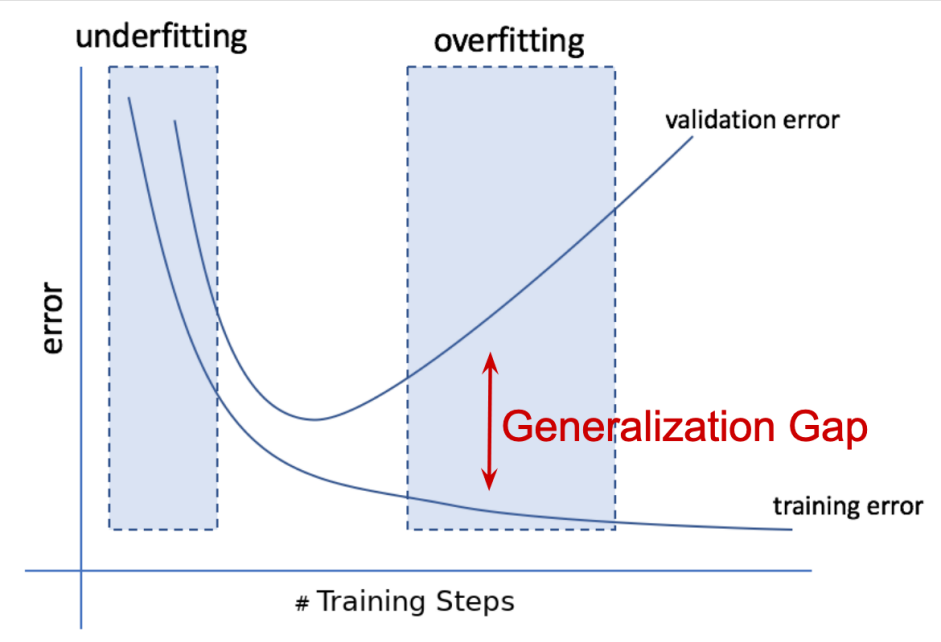



Picture taken from Brenda Ngs introductory talk at the: deep learning for science school 2019

[5] This data is "unseen" by the algorithm during the training stage

# Validation Data

- Want to enable network to abstract / generalize on unknown data AND avoid overfitting (i.e. avoid that network reproduces features from training data only)
- **Validation Data:** Part of training data that is NOT used to update internal parameters[5], but used to determine when training is complete

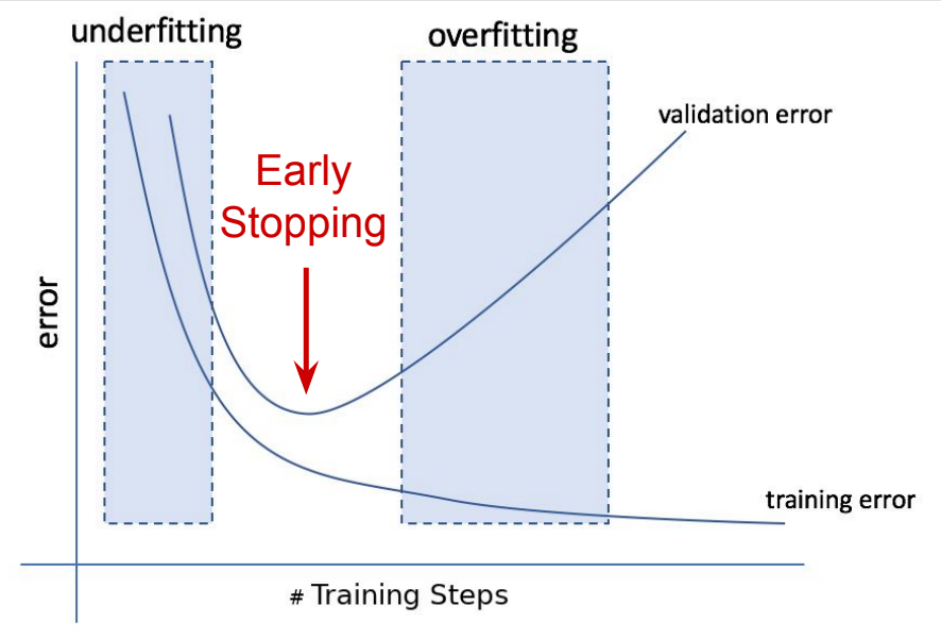



Picture taken from Mustafa Mustafas talk at the: deep learning for science school 2019

[5] This data is "unseen" by the algorithm during the training stage

# Validation Data

- Want to enable network to abstract / generalize on unknown data AND avoid overfitting (i.e. avoid that network reproduces features from training data only)
- **Validation Data:** Part of training data that is NOT used to update internal parameters[5], but used to determine when training is complete



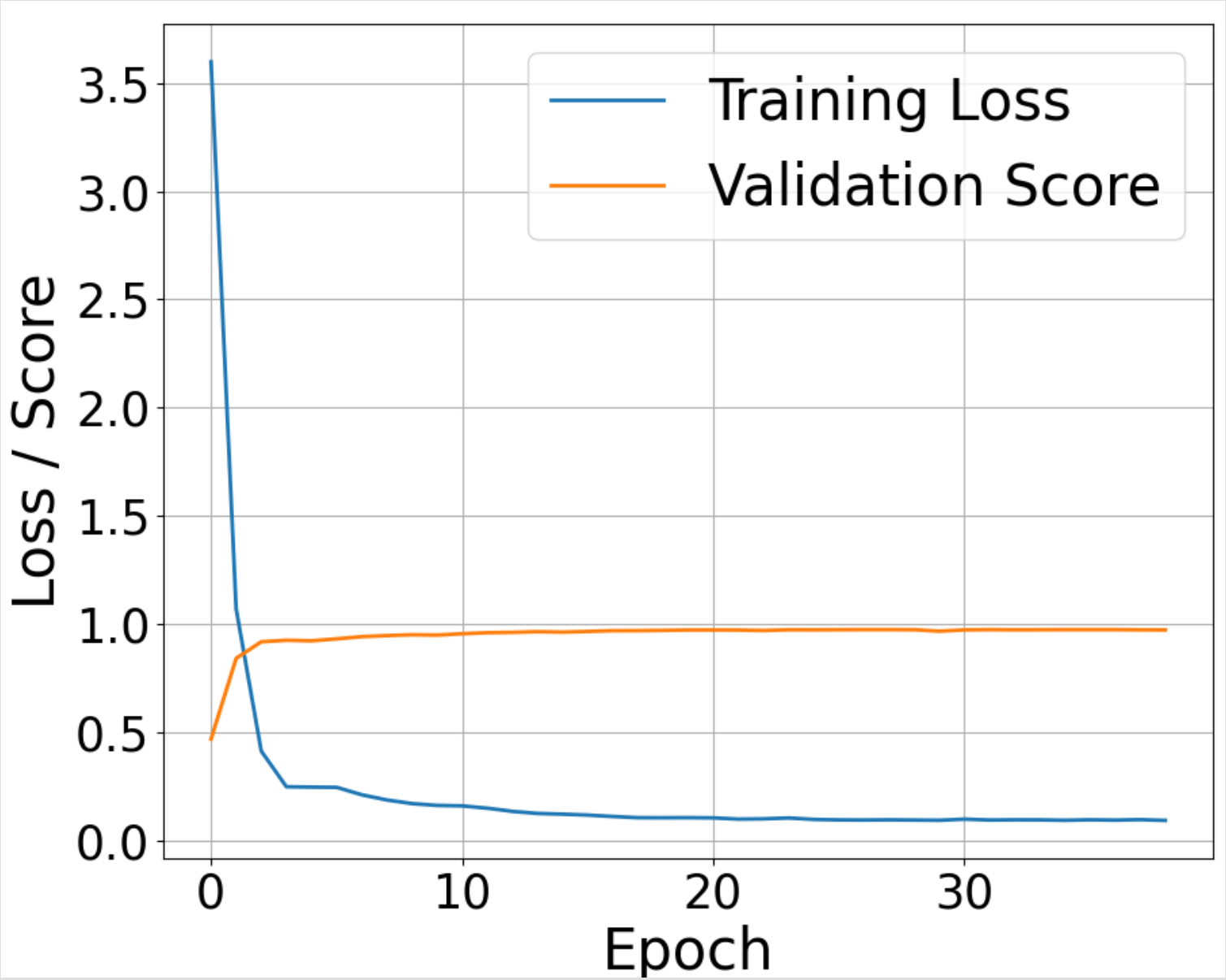

Picture taken from Mustafa Mustafas talk at the: deep learning for science school 2019

[5] This data is "unseen" by the algorithm during the training stage

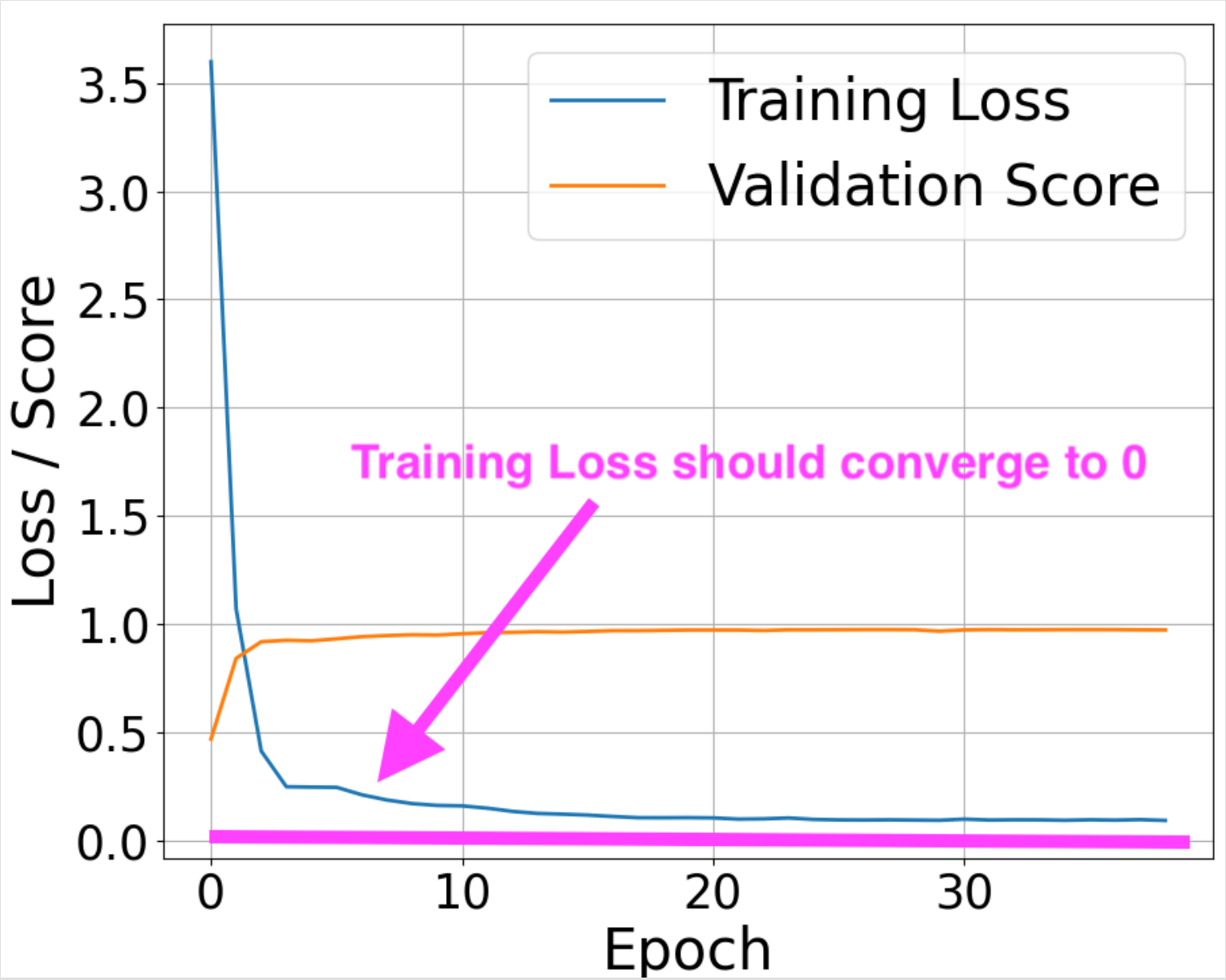
# Implementing Early Stopping and Validation Data in the scikit MLPRegressor

```
my_mlp =   MLPRegressor(
             hidden_layer_sizes=(10),
             activation='relu',
             solver='sgd',
             warm_start=True,
             max_iter = 1000,
             shuffle=True,
             tol=1e-6,
             validation_fraction=0.5, #---> Define the percentage of
             #training data that shall be kept aside
             early_stopping=True, #---> Enable early stopping
             random_state=0,
             learning_rate_init = 0.05
       )
```

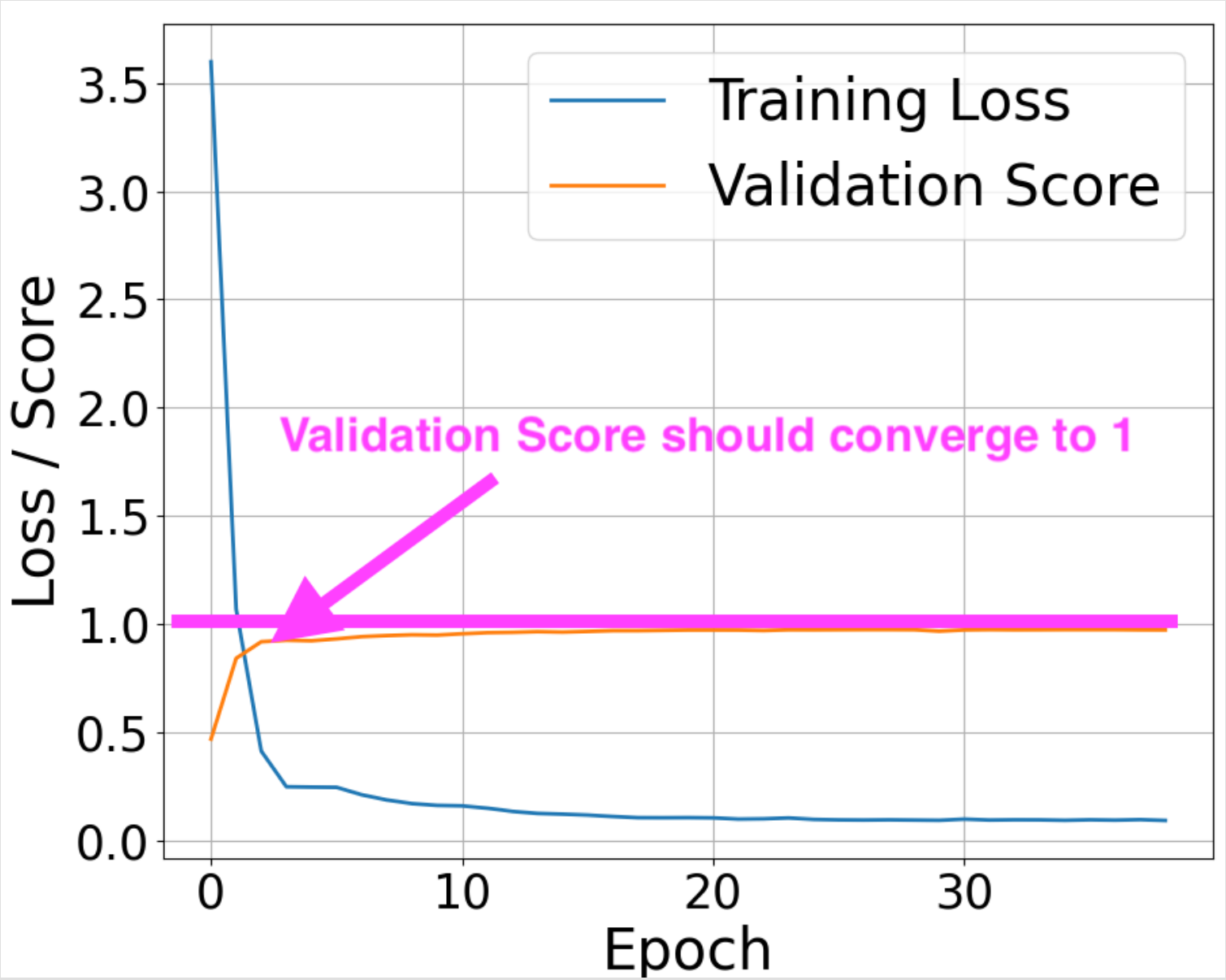
# Understanding the Learning Curve from scikit



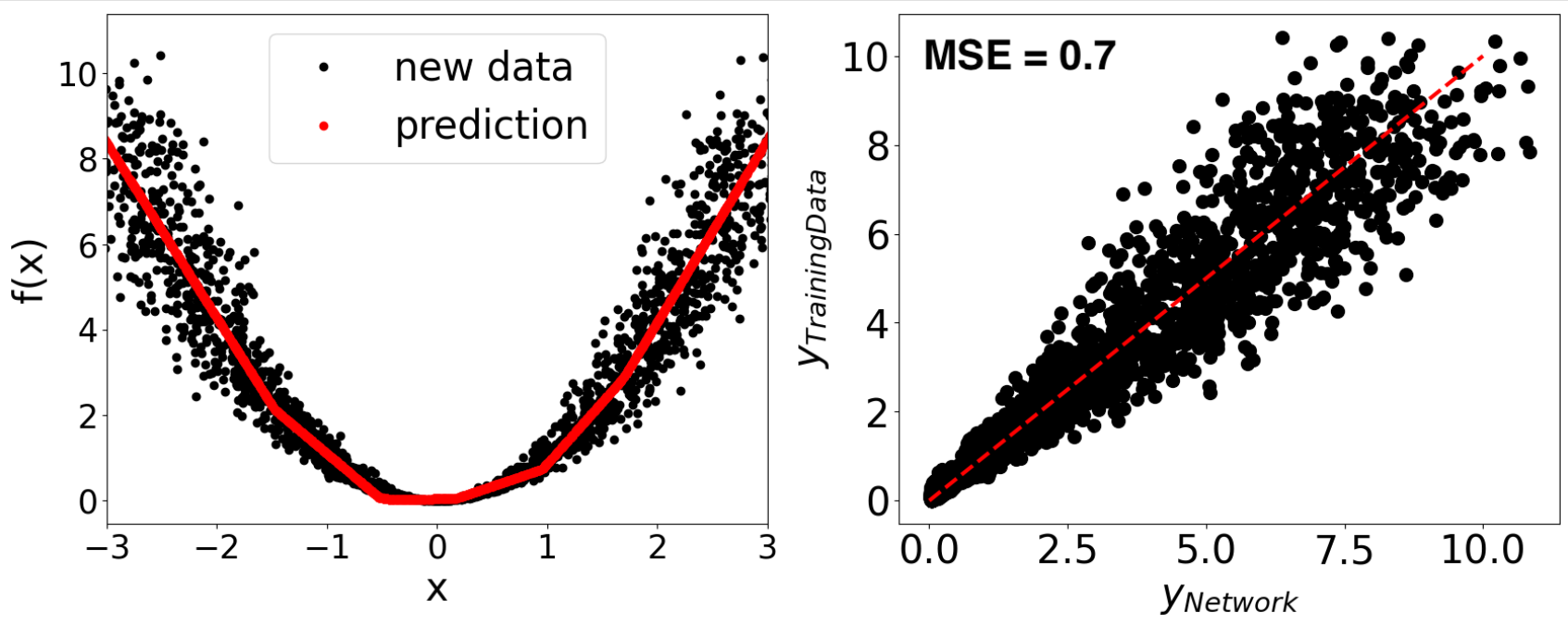
# Understanding the Learning Curve from scikit



# Understanding the Learning Curve from scikit

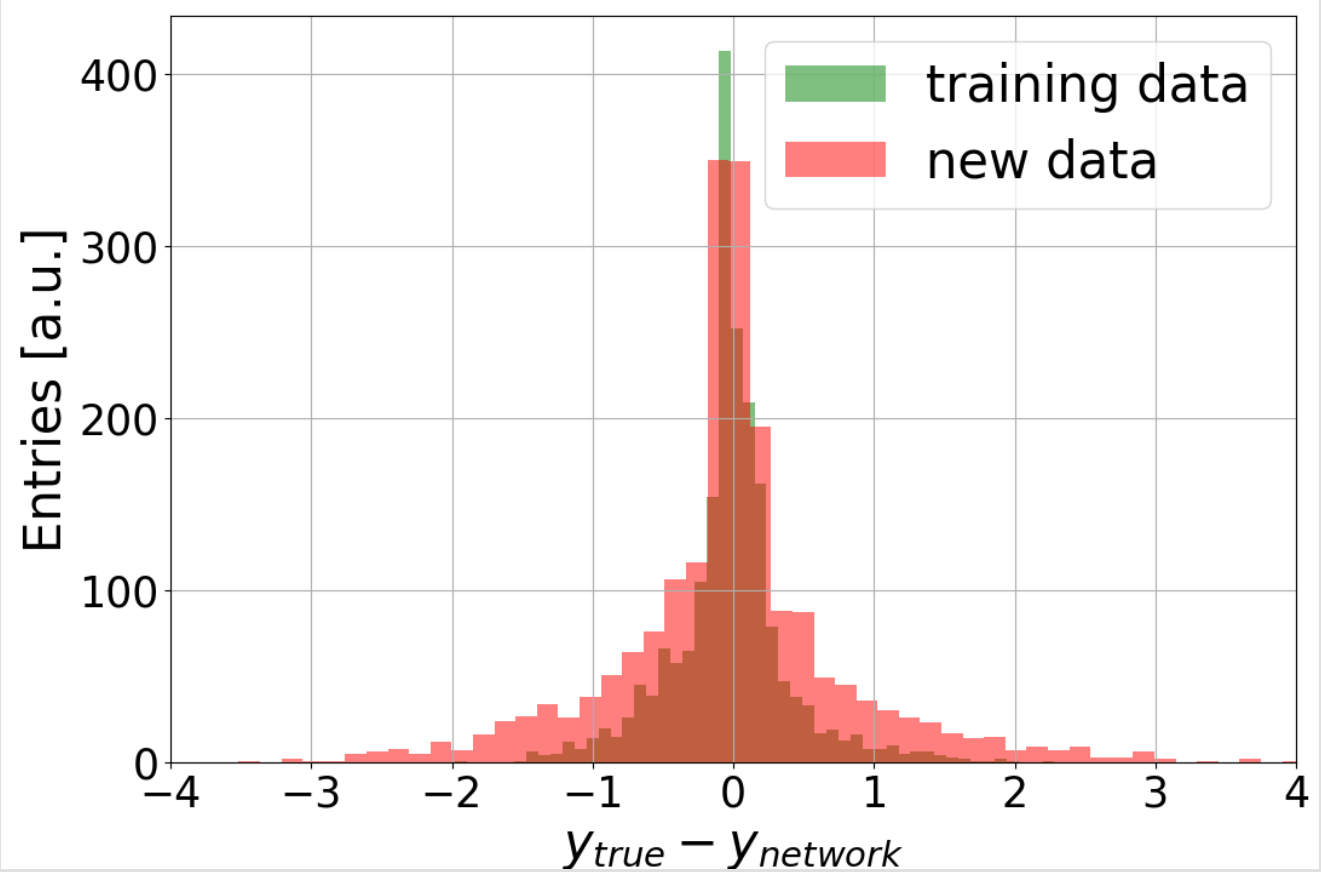


# Feed "new" Data into the re-trained Neural Network




- Trained neural network
  - Include validation data
  - Use early stopping
- Performance improvement $\sim 5\%$

# Judging Regression Performance with Residuals




- Residual: $y_{true} - y_{network}$
- Should be centered at 0 $\rightarrow$ you are in trouble if not!
- In general[6]: Small residual width $\rightarrow$ good regression performance

[6] There are cases where this not true: Too small width on training data $\leftrightarrow$ overfitting

# DIY: Regression with Validation Data

1.) Go to: `https://replit.com/@daniel49/FSUMLLecture2`

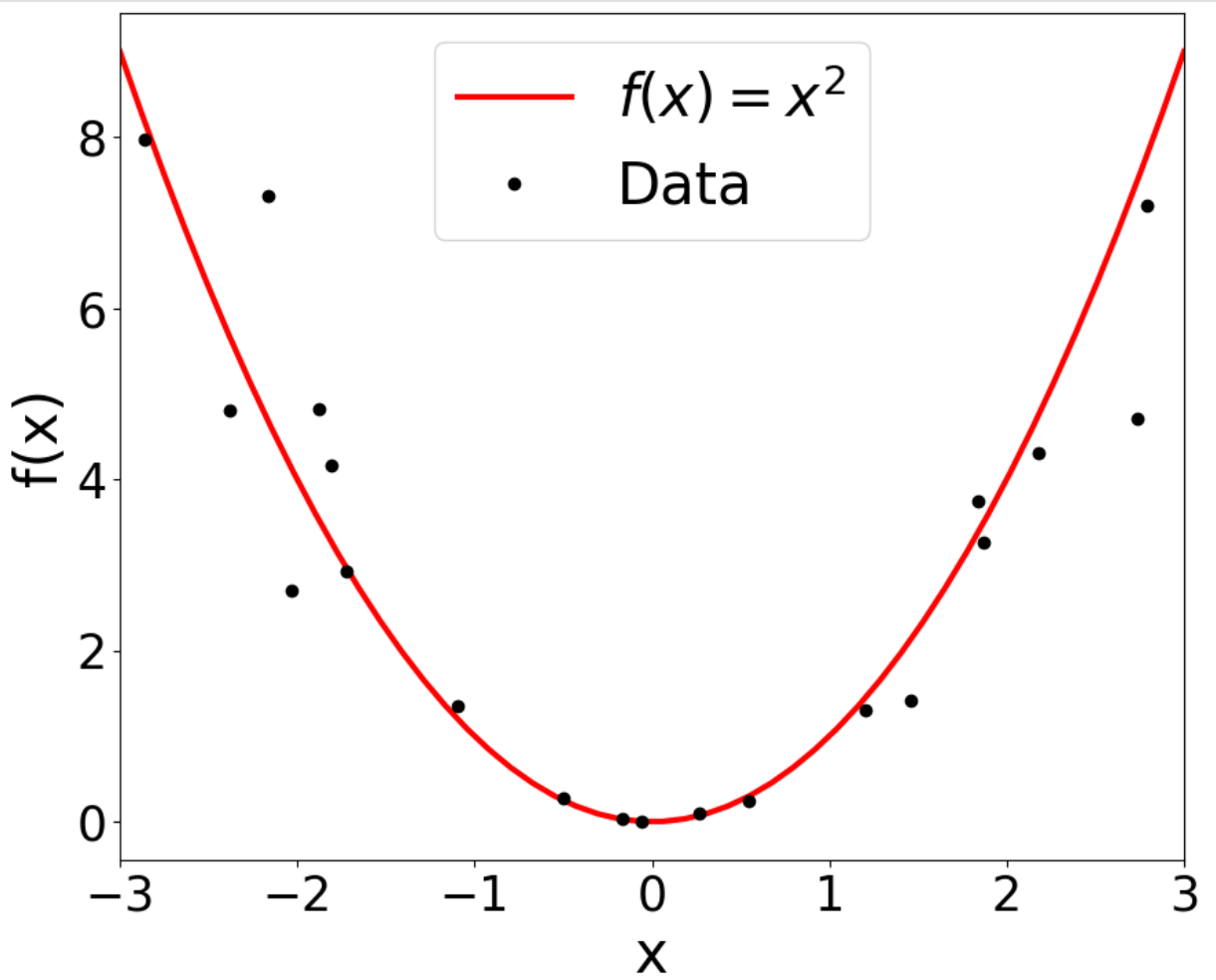
2.) Klick on the **Fork** button

3.) Sign in or log in with your credentials (repl is free)

4.) Follow instructions in **main.py**

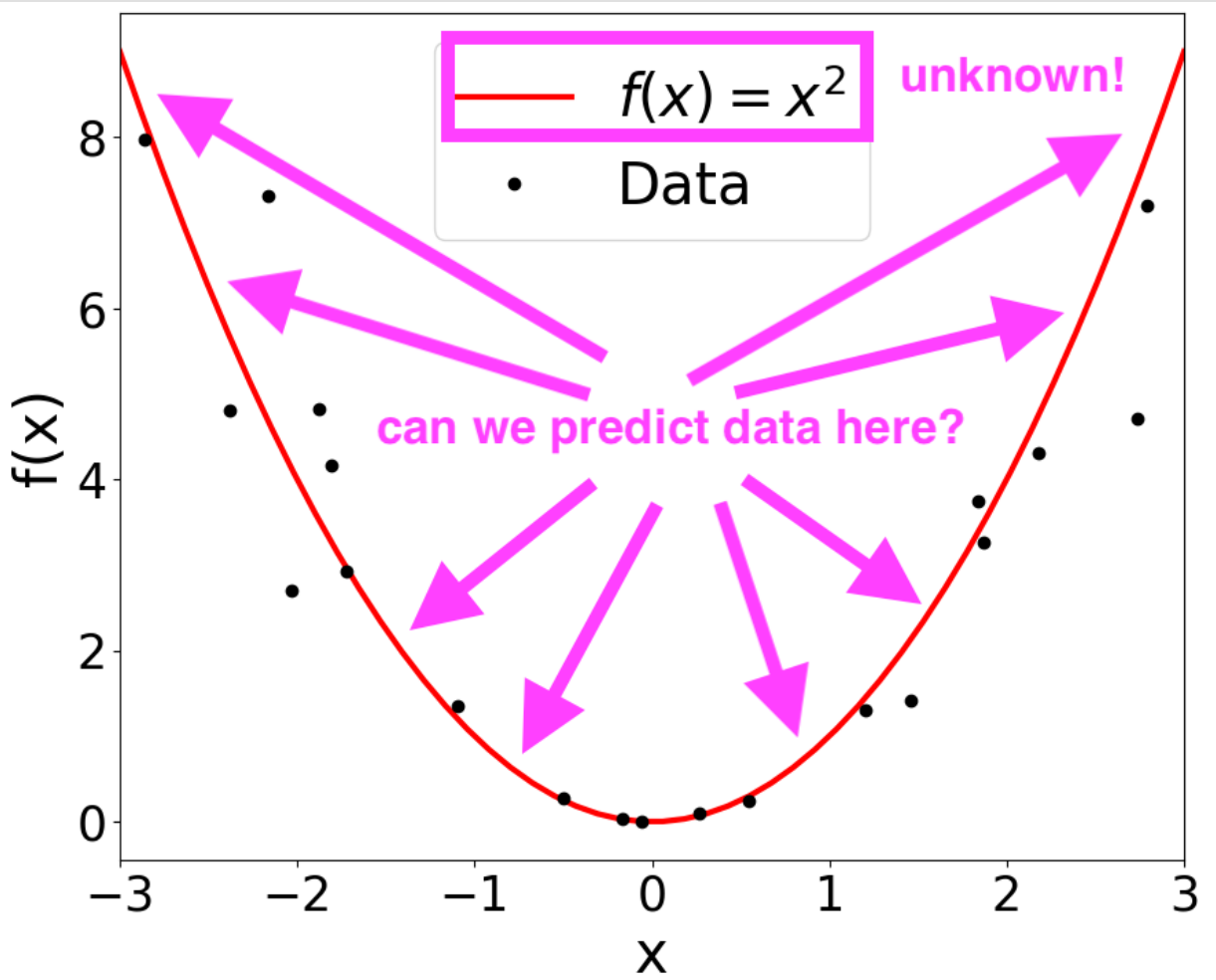
# Data Interpolation and Sampling with Gaussian Processes

- **Given:** Few data points which stem from an unknown function
- **Goal:** Try to find underlying (true) distribution
  - Interpolate existing data
  - Sample data from fitted distribution

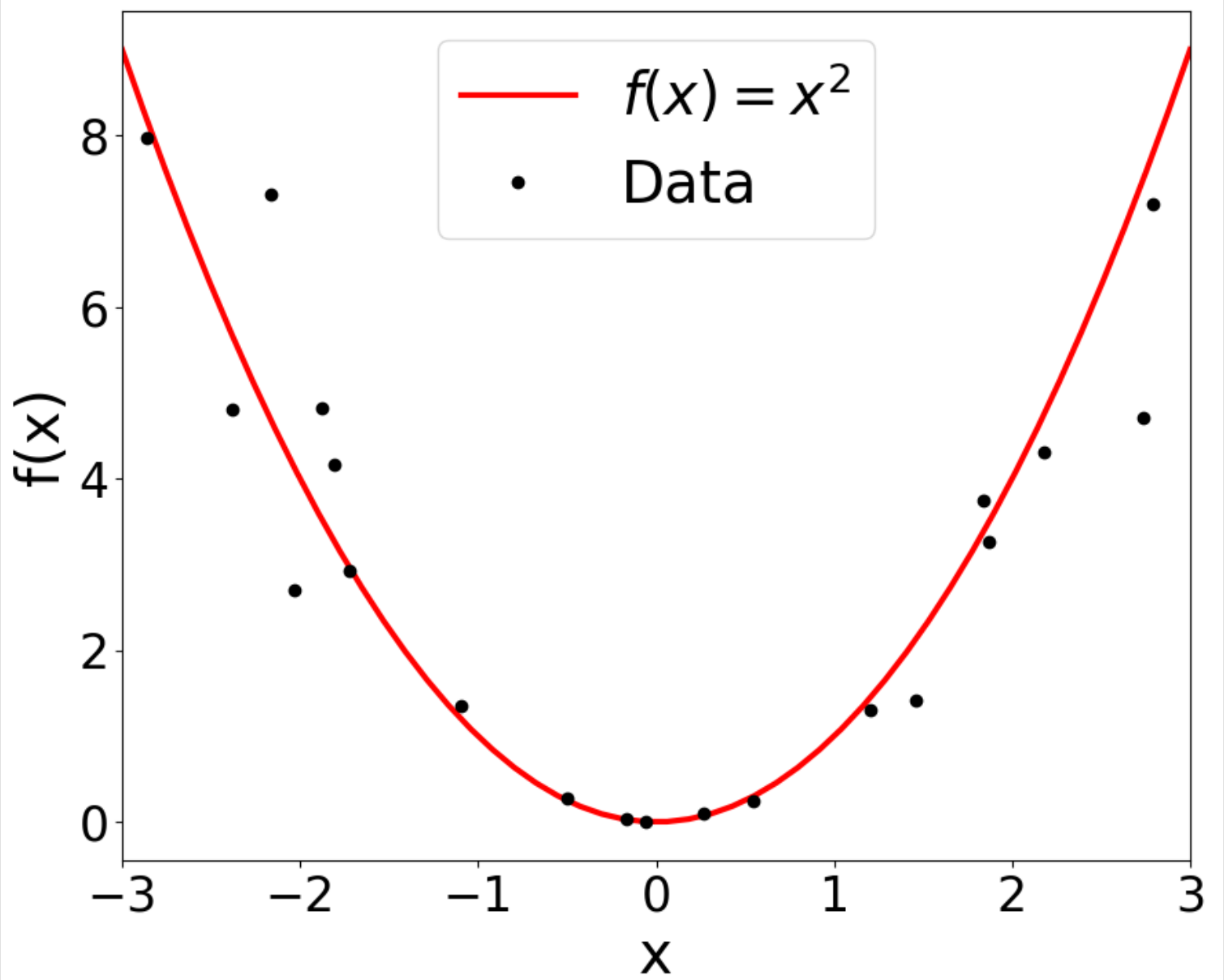


# Data Interpolation and Sampling with Gaussian Processes

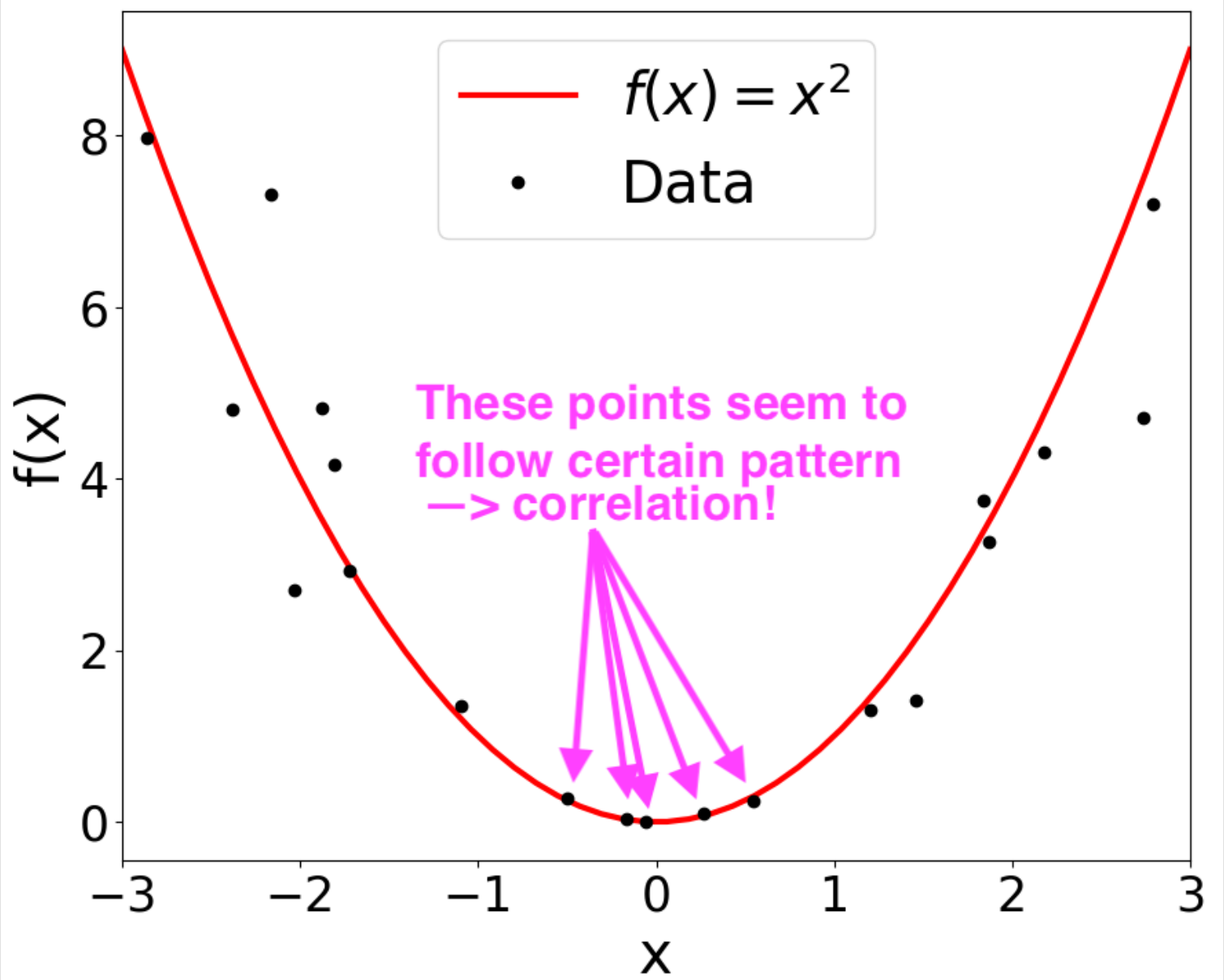
- **Given:** Few data points which stem from an unknown function
- **Goal:** Try to find underlying (true) distribution
  - Interpolate existing data
  - Sample data from fitted distribution



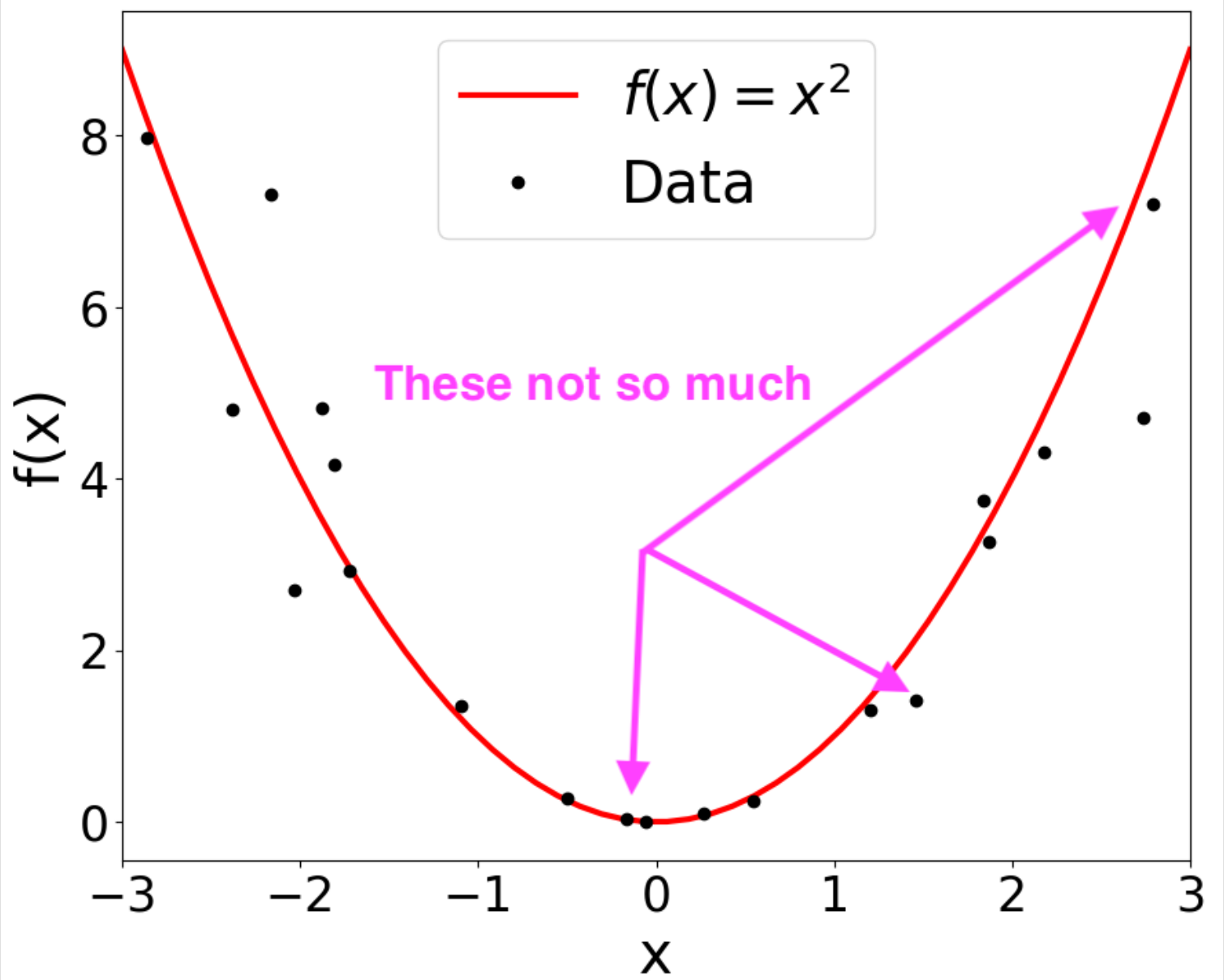
# Correlated Data Points



# Correlated Data Points



# Correlated Data Points

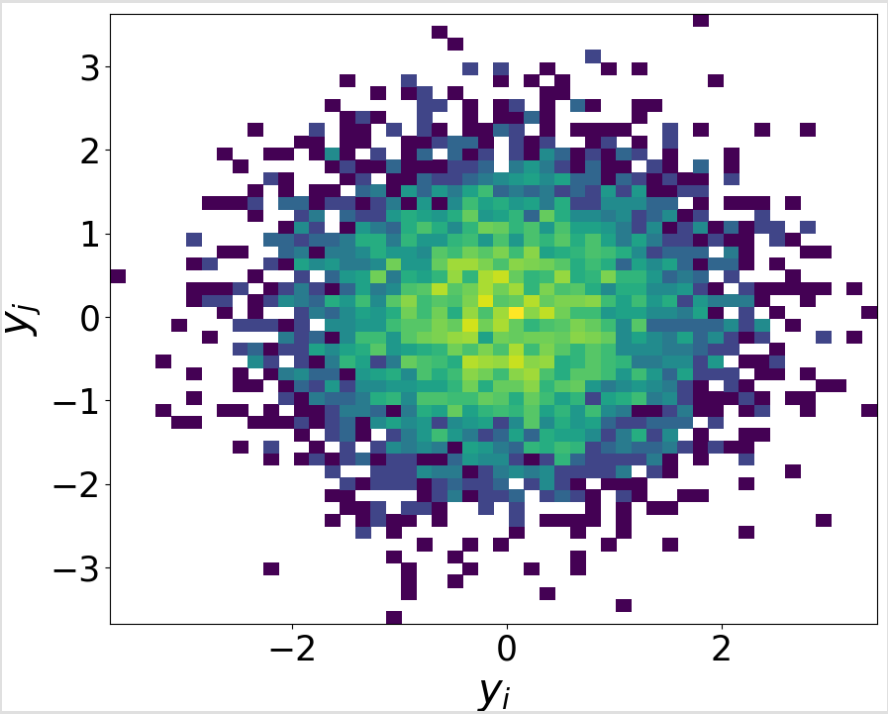


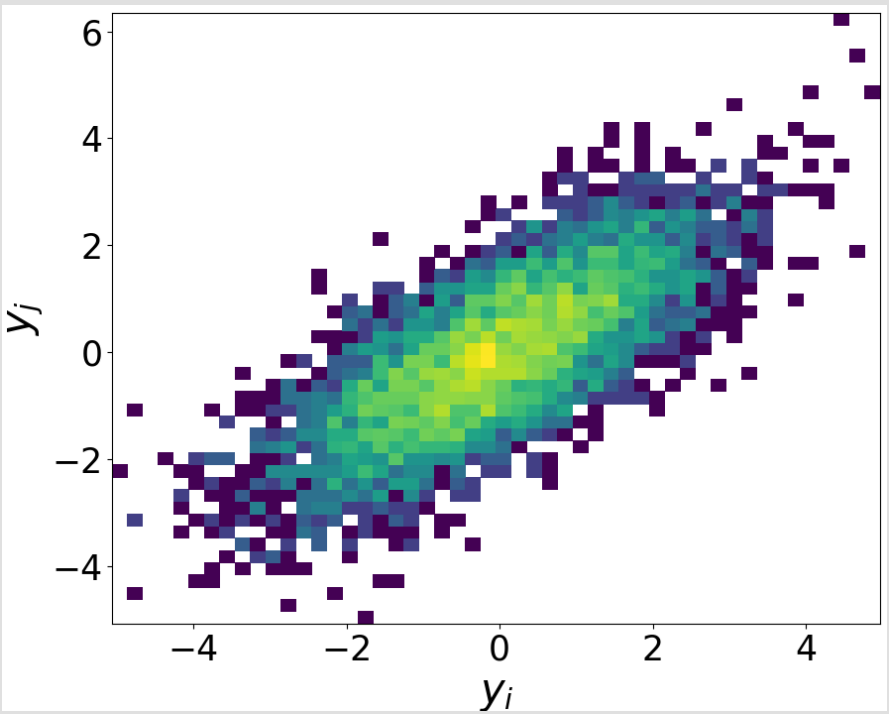
# Sampling correlated Data from a multidimensional Gaussian Distribution

- Sample two points $y_i$, $y_j$ with known correlation $\rho_{ij}$ and variances $\sigma_i$, $\sigma_j$
- Use multidimensional Gaussian distribution (centered at zero):

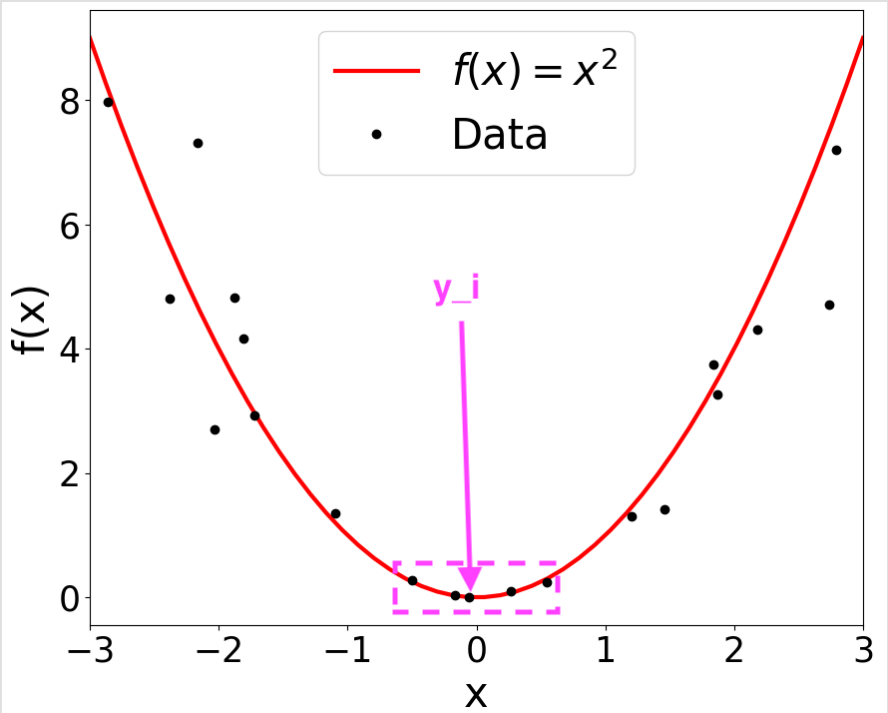
$$f(y_i, y_j) = \frac{1}{2\pi\sigma_i\sigma_j\sqrt{1-\rho_{ij}^2}} \cdot \exp\left[\left(\frac{y_i}{\sigma_i}\right)^2 + \left(\frac{y_j}{\sigma_j}\right)^2 - \left(\frac{2\rho_{ij}y_iy_j}{\sigma_i\sigma_j}\right)\right]$$

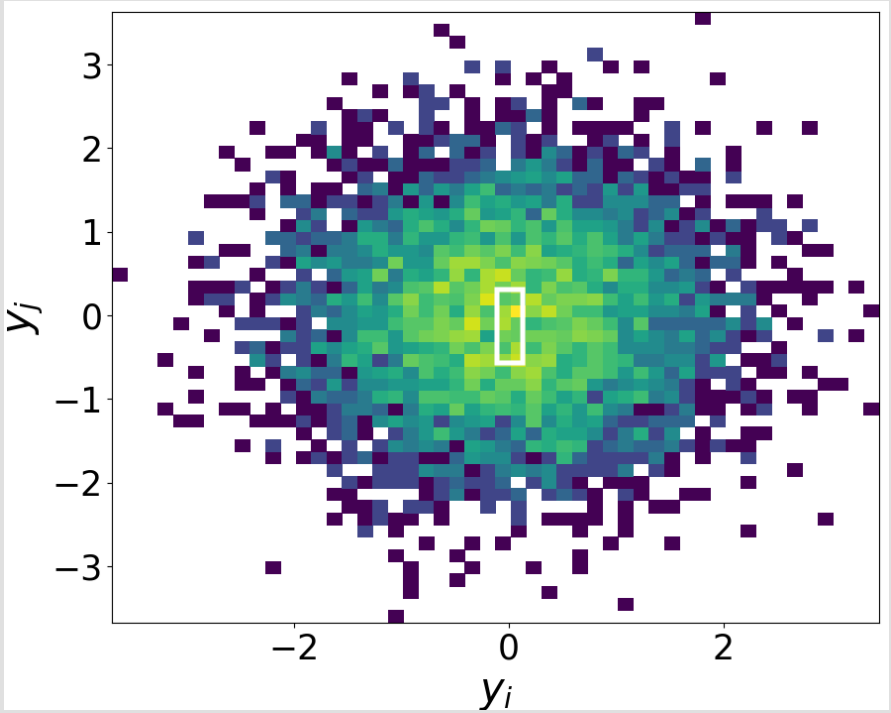
- **Left panel:** $\rho_{ij} = 0.01$, $\sigma_i = \sigma_j = 1$
- **Right panel:** $\rho_{ij} = 1.5$, $\sigma_i = \sigma_j = 2$



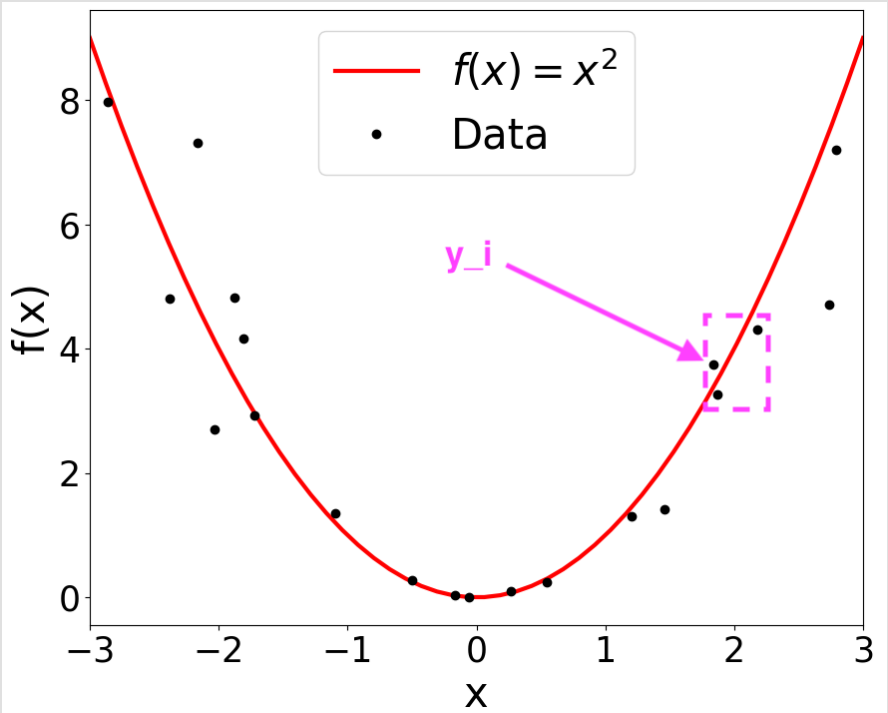
# Application to our Problem

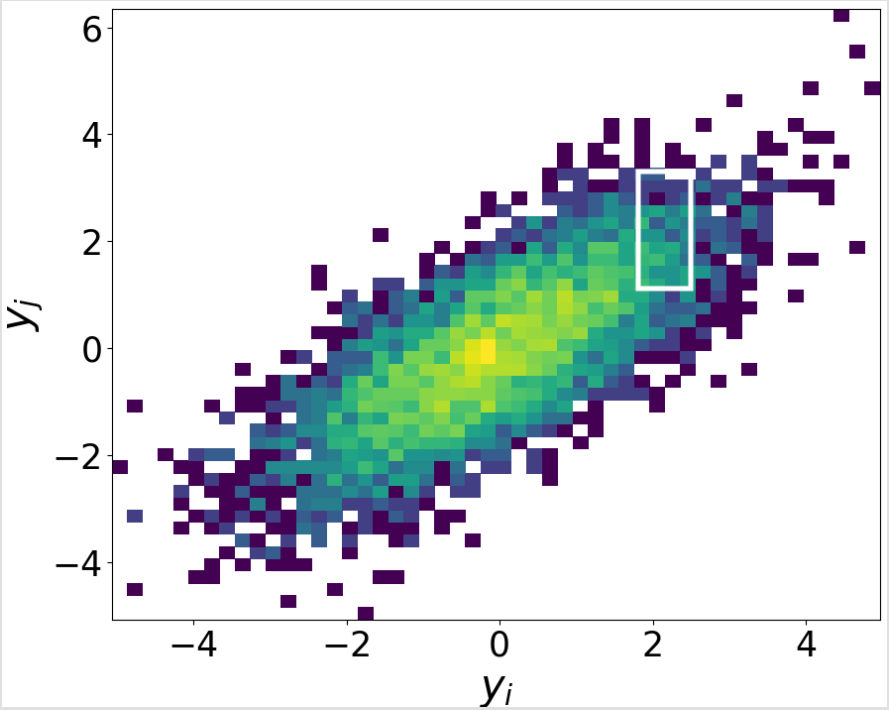



- Sampling (right panel) done with: $\rho_{ij} = 0.01$, $\sigma_i = \sigma_j = 1$
- Sampled data (inside white square) $\approx$ fraction of observed data (magenta square)
- Describe parts of our data by multidimensional Gaussian with known correlation $\rightarrow$ **Gaussian Processes**

# Application to our Problem




- Sampling (right panel) done with: $\rho_{ij} = 1.5$, $\sigma_i = \sigma_j = 2$
- Sampled data (inside white square) $\approx$ fraction of observed data (magenta square)
- Describe parts of our data by multidimensional Gaussian with known correlation $\rightarrow$ **Gaussian Processes**

# Gaussian Processes: General Idea

1.) If correlation matrix $\rho$ is known: Generate data $\boldsymbol{y} = \{y_1, y_2, ..., y_N\}$ via:

$f(\boldsymbol{y}) = \frac{1}{2\pi|\rho|} \cdot \exp[-0.5 \boldsymbol{y} \rho^{-1} \boldsymbol{y}]$

# Gaussian Processes: General Idea

1.) If correlation matrix $\rho$ is known: Generate data $\boldsymbol{y} = \{y_1, y_2, ..., y_N\}$ via:
$f(\boldsymbol{y}) = \frac{1}{2\pi|\rho|} \cdot \exp[-0.5 \boldsymbol{y} \rho^{-1} \boldsymbol{y}]$

2.) $\rho$ is not known $\rightarrow$ Parameterize $\rho$ via Kernel Function

# Gaussian Processes: General Idea

1.) If correlation matrix $\rho$ is known: Generate data $\boldsymbol{y} = \{y_1, y_2, ..., y_N\}$ via:
$f(\boldsymbol{y}) = \frac{1}{2\pi|\rho|} \cdot \exp[-0.5 \boldsymbol{y} \rho^{-1} \boldsymbol{y}]$

2.) $\rho$ is not known $\rightarrow$ Parameterize $\rho$ via Kernel Function

3.) Fit $f(\boldsymbol{y})$ to observed data

# Gaussian Processes: Kernel Functions

- Most commonly used:
  Exponential Squared: $k(x_i, x_j) = \sigma_f^2 \exp\left[ -\frac{(x_i - x_j)^2}{\Delta^2} \right] + \sigma_n^2 \delta(x_i, x_j)$

# Gaussian Processes: Kernel Functions

- Most commonly used:

  Exponential Squared: $k(x_i, x_j) = \sigma_f^2 \exp\left[ -\frac{(x_i - x_j)^2}{\Delta^2} \right] + \sigma_n^2 \delta(x_i, x_j)$

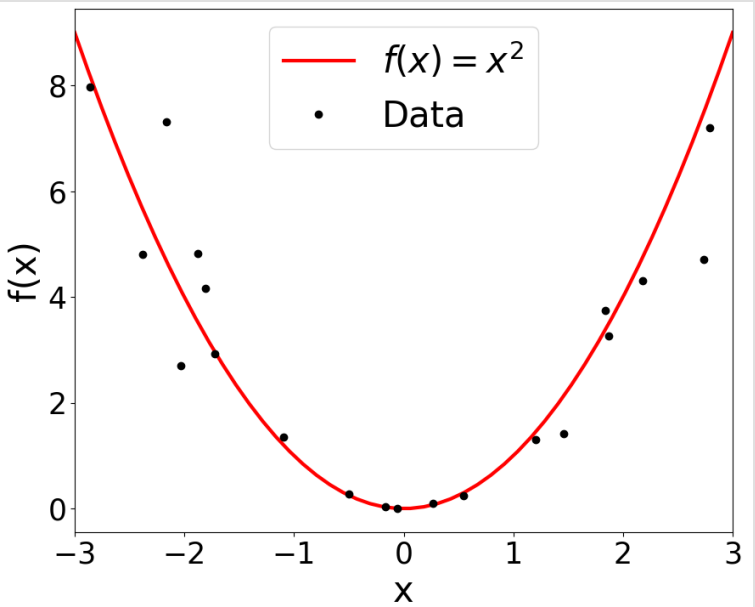
- Crucial parameters:
  - Length scale $\Delta$: Defines "far" / "close" points
  - $\sigma_f$: global variance
  - $\sigma_n$: Noise (pick up statistical fluctuations in data)

# Gaussian Processes: Kernel Functions

- Most commonly used:

  Exponential Squared: $k(x_i, x_j) = \sigma_f^2 \exp\left[ -\frac{(x_i - x_j)^2}{\Delta^2} \right] + \sigma_n^2 \delta(x_i, x_j)$

- Crucial parameters:
  - Length scale $\Delta$: Defines "far" / "close" points
  - $\sigma_f$: global variance
  - $\sigma_n$: Noise (pick up statistical fluctuations in data)
- Properties:
  - $\lim_{|x_i - x_j| \to 0} k(x_i, x_j) \to \sigma_f^2 + \sigma_n^2$
  - $\lim_{|x_i - x_j| \to \infty} k(x_i, x_j) \to 0$

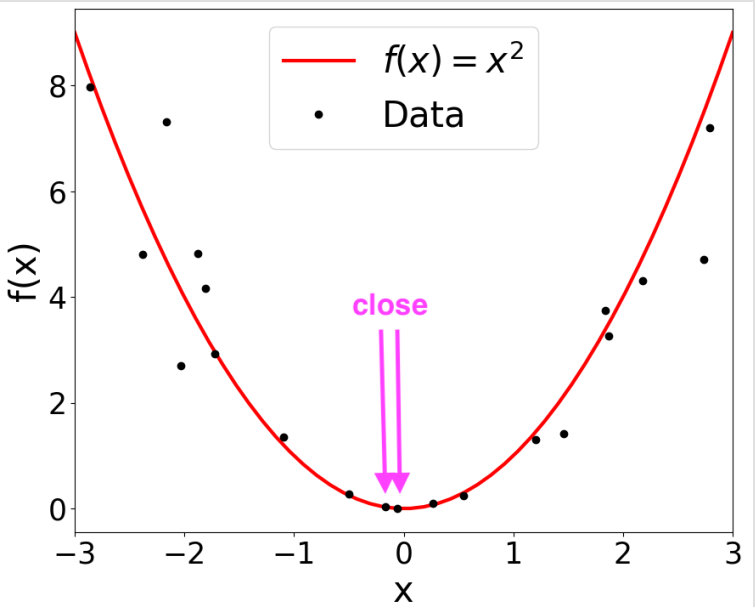


# Gaussian Processes: Kernel Functions

- Most commonly used:

  Exponential Squared: $k(x_i, x_j) = \sigma_f^2 \exp\left[-\frac{(x_i - x_j)^2}{\Delta^2}\right] + \sigma_n^2 \delta(x_i, x_j)$

- Crucial parameters:
  - Length scale $\Delta$: Defines "far" / "close" points
  - $\sigma_f$: global variance
  - $\sigma_n$: Noise (pick up statistical fluctuations in data)
- Properties:
  - $\lim_{|x_i - x_j| \to 0} k(x_i, x_j) \to \sigma_f^2 + \sigma_n^2 \to$ Close points share features
  - $\lim_{|x_i - x_j| \to \infty} k(x_i, x_j) \to 0$

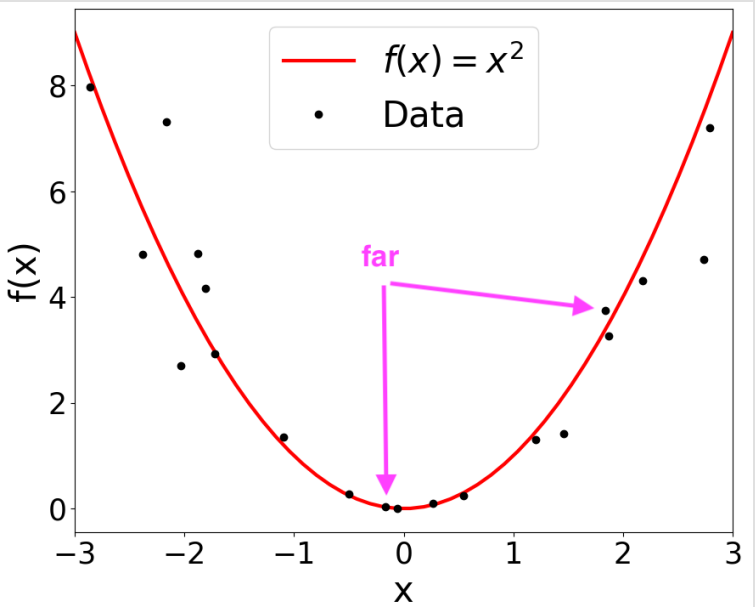


# Gaussian Processes: Kernel Functions

- Most commonly used:

  Exponential Squared: $k(x_i, x_j) = \sigma_f^2 \exp\left[ -\frac{(x_i - x_j)^2}{\Delta^2} \right] + \sigma_n^2 \delta(x_i, x_j)$

- Crucial parameters:
  - Length scale $\Delta$: Defines "far" / "close" points
  - $\sigma_f$: global variance
  - $\sigma_n$: Noise (pick up statistical fluctuations in data)
- Properties:
  - $\lim_{|x_i - x_j| \to 0} k(x_i, x_j) \to \sigma_f^2 + \sigma_n^2$
  - $\lim_{|x_i - x_j| \to \infty} k(x_i, x_j) \to 0 \to$ Distant points do not "know" each other



# Setting up Gaussian Processes in scikit

$$k(x_i, x_j) = \underbrace{\left[\exp\left\{-\frac{1}{2}\left(\frac{x_i - x_j}{\text{length\_scale}}\right)^2\right\}\right]}_{\text{RBF}} + \alpha(x_i, x_j)\delta(x_i, x_j) + \underbrace{\text{noise\_level} \cdot \delta(x_i, x_j)}_{\text{WhiteKernel}}$$

```
#1.) Define the kernel:
#RBF: Radial Basis Function = exponential squared
kernel = RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e2)) #sigma_f = 1.
#White kernel: Corresponds to sigma_n --> constant noise
   + WhiteKernel(noise_level=1.0, noise_level_bounds=(1e-10, 1e+1))
```

# Setting up Gaussian Processes in scikit

$$k(x_i, x_j) = \underbrace{\left[\exp\left\{-\frac{1}{2}\left(\frac{x_i - x_j}{\text{length\_scale}}\right)^2\right\}\right]}_{\text{RBF}} + \alpha(x_i, x_j)\delta(x_i, x_j) + \underbrace{\text{noise\_level} \cdot \delta(x_i, x_j)}_{\text{WhiteKernel}}$$

```
#1.) Define the kernel:
#RBF: Radial Basis Function = exponential squared
kernel = RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e2)) #sigma_f = 1.
#White kernel: Corresponds to sigma_n --> constant noise
   + WhiteKernel(noise_level=1.0, noise_level_bounds=(1e-10, 1e+1))

#2.) Setting the processes:
my_gp = GaussianProcessRegressor(
       kernel=kernel,
       n_restarts_optimizer=10, #--> How many times to run the minimization
       alpha=0.0 #--> similar to sigma_n if constant,
       #can be set for each data point individually--> individual error
 )
```

# Setting up Gaussian Processes in scikit

$$k(x_i, x_j) = \underbrace{\left[\exp\left\{-\frac{1}{2}\left(\frac{x_i - x_j}{\text{length\_scale}}\right)^2\right\}\right]}_{\text{RBF}} + \alpha(x_i, x_j)\delta(x_i, x_j) + \underbrace{\text{noise\_level} \cdot \delta(x_i, x_j)}_{\text{WhiteKernel}}$$

```
#1.) Define the kernel:
#RBF: Radial Basis Function = exponential squared
kernel = RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e2)) #sigma_f = 1.
#White kernel: Corresponds to sigma_n --> constant noise
   + WhiteKernel(noise_level=1.0, noise_level_bounds=(1e-10, 1e+1))

#2.) Setting the processes:
my_gp = GaussianProcessRegressor(
       kernel=kernel,
       n_restarts_optimizer=10, #--> How many times to run the minimization
       alpha=0.0 #--> similar to sigma_n if constant,
       #can be set for each data point individually--> individual error
 )

 #3.) Set the parameters:
 my_gp.fit(x_values,y_values)
```

# Setting up Gaussian Processes in scikit

$$k(x_i, x_j) = \underbrace{\left[\exp\left\{-\frac{1}{2}\left(\frac{x_i - x_j}{\text{length\_scale}}\right)^2\right\}\right]}_{\text{RBF}} + \alpha(x_i, x_j)\delta(x_i, x_j) + \underbrace{\text{noise\_level} \cdot \delta(x_i, x_j)}_{\text{WhiteKernel}}$$

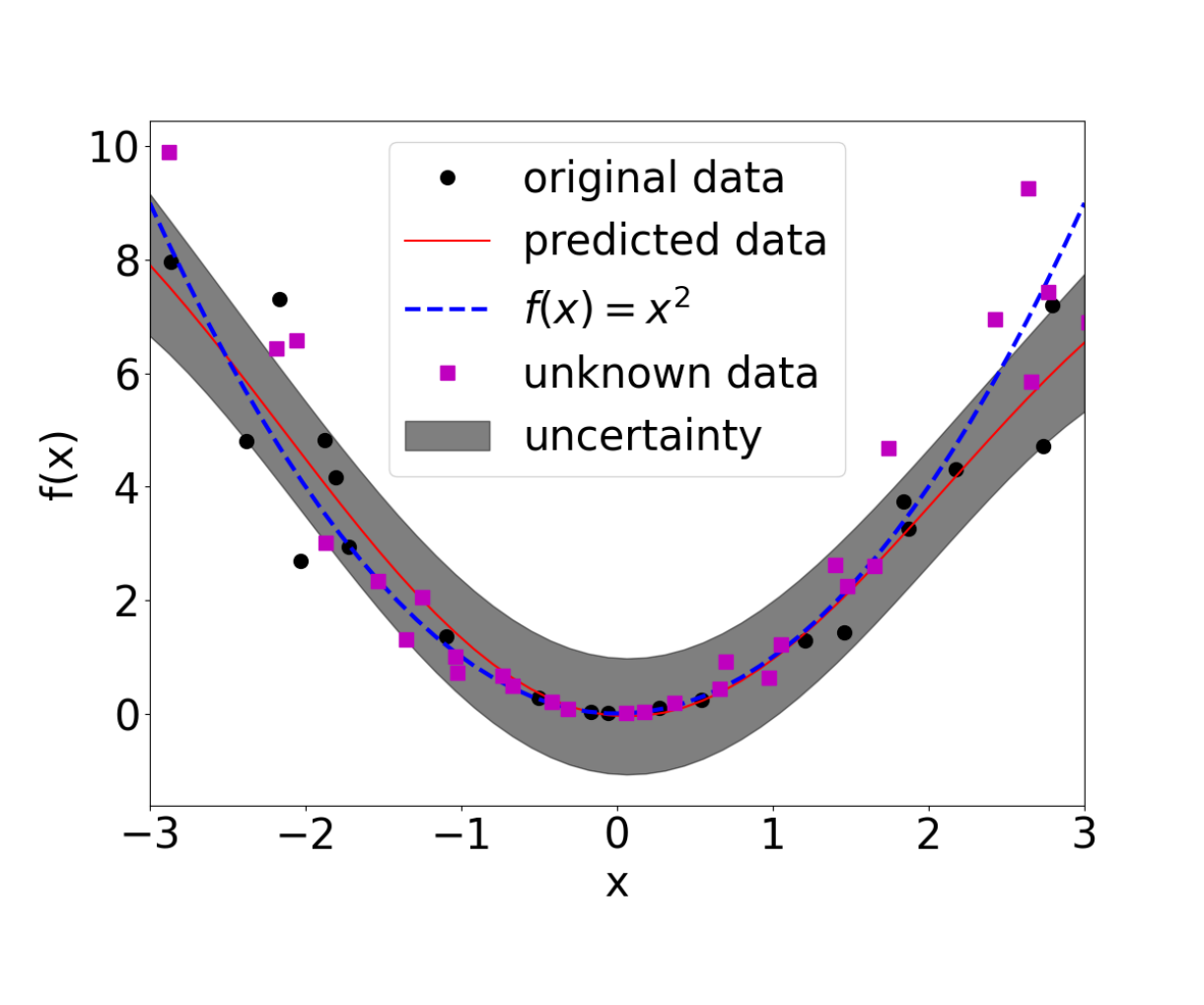
```
#1.) Define the kernel:
#RBF: Radial Basis Function = exponential squared
kernel = RBF(length_scale=1.0, length_scale_bounds=(1e-2, 1e2)) #sigma_f = 1.
#White kernel: Corresponds to sigma_n --> constant noise
   + WhiteKernel(noise_level=1.0, noise_level_bounds=(1e-10, 1e+1))

#2.) Setting the processes:
my_gp = GaussianProcessRegressor(
       kernel=kernel,
       n_restarts_optimizer=10, #--> How many times to run the minimization
       alpha=0.0 #--> similar to sigma_n if constant,
       #can be set for each data point individually--> individual error
 )

 #3.) Set the parameters:
 my_gp.fit(x_values,y_values)

 #4.) Get the predictions:
 predictions, covariances = my_gp.predict(x_values,return_cov=True)
```

# Applying Gaussian Processes to our Problem




- Unknown and original data have been generated together
- Could improve performance by including individual uncertainties ($\alpha$-parameter in scikit)

# Gaussian Processes: Short Summary

- Gaussian processes are powerful tools
- Purely driven by measured data
- Highly depend on kernel function
  $\rightarrow$ Problem if you can not formulate a proper one
- Computationally expensive for large data sets with $N_{points}$:
  Kernel matrix is of size $N_{points} \times N_{points}$
- Further reading:
  - **A Visual Exploration of Gaussian Processes**
    (from Distill.pub, very nice explanation)
  - **Gaussian Processes, not quite for dummies**
    (from The Gradient)
  - **scikit-Documentation**
  - **Gaussian Processes for Machine Learning**
    (Carl Eduard Rasmussen and Christopher K.I. Williams, MIT Press 2006)

# DIY: Gaussian Processes

1.) Go to: `https://replit.com/@daniel49/FSUMLLecture2`

2.) Klick on the **Fork** button

3.) Sign in or log in with your credentials (repl is free)

4.) Follow instructions in **main.py**