PAL Machine Learning Workshop Week 4: Polynomial Regression & Regularization

24/02/2023 18:00 Future Technologies Lab

Contents

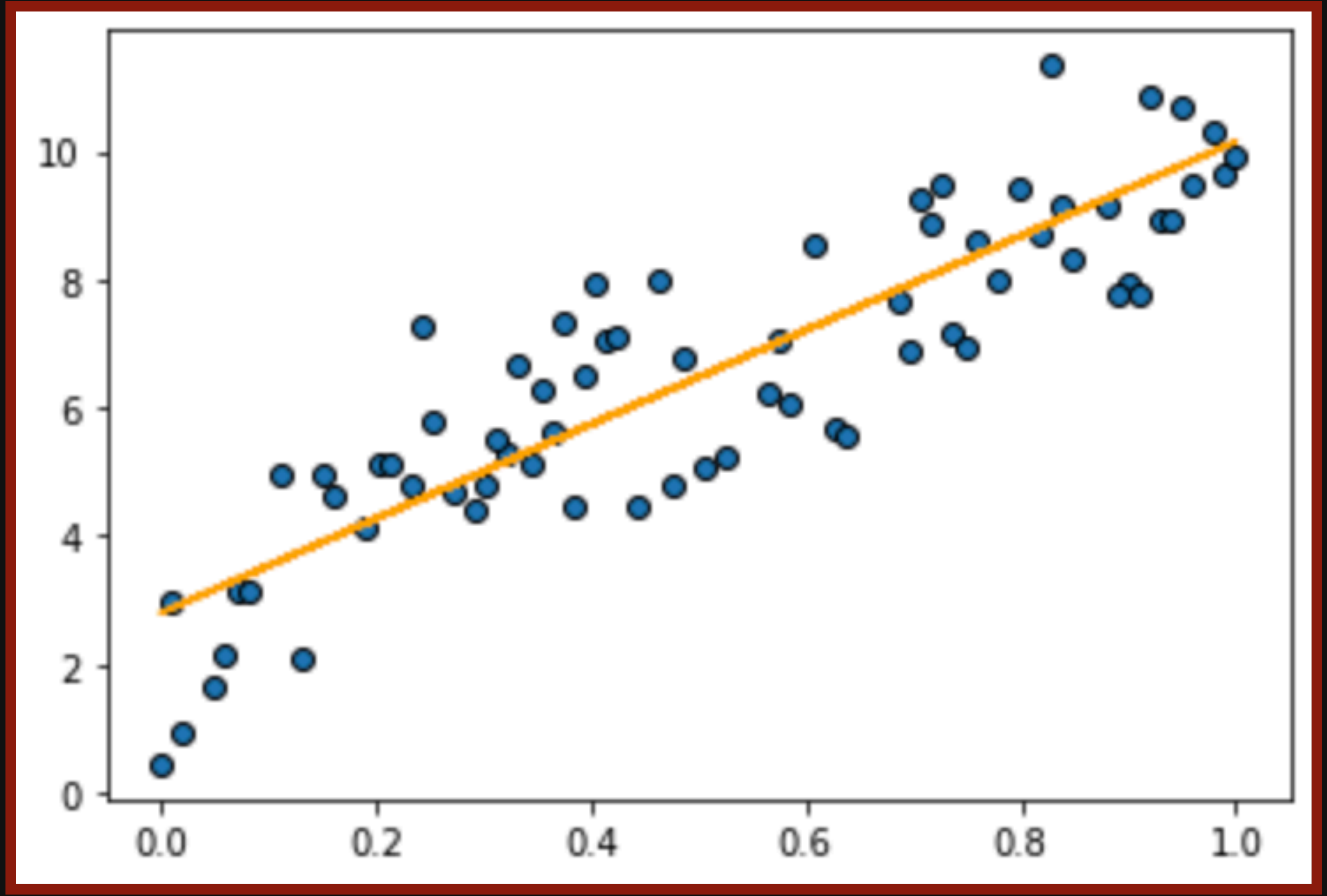
- Recap: Linear Regression
- Polynomial Regression
- Overfitting & Underfitting
- L2 Regularisation

Recap: Linear Regression

- We could assume that y is some linear function of x. In other words, for some unknown $w_0, w_1 \in \mathbb{R}$, we have:

$$f(x) = w_0 + w_1 x$$

- We will refer to w_0, w_1 as the *parameters,* where x is the *independent* variable and y is the *dependent* variable.

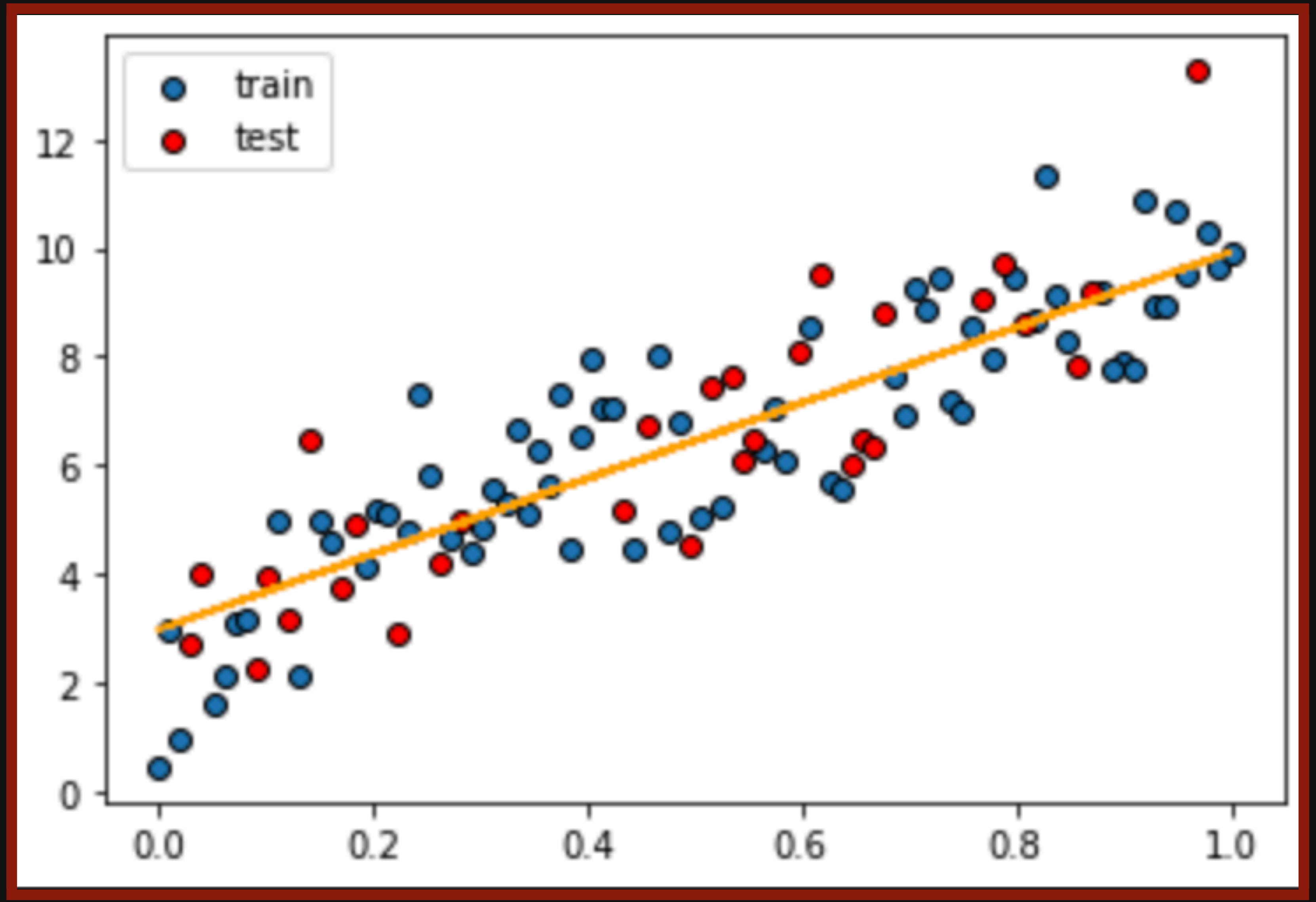


Recap: Linear Regression

- Our aim is to find the best set of parameters $W^* = \{w_0^*, w_1^*\}$, i.e. the one that gives us the best result on our training data.

- Once we've learned the predictive model $f(x)$, we want to use it to make predictions y' for the new data x' that we haven't seen before:

$$y' = f(x') = w_0^* + w_1^* x$$



Recap: Linear Regression

- We can represent our data, parameters and target values using matrix notation:

$$X = \begin{array}{cc} \begin{bmatrix} x_{10} & \cdots & x_{1d} \\ \vdots & \ddots & \\ x_{n0} & & x_{nd} \end{bmatrix} & n \\ d+1 & \end{array}$$

- x is a *design matrix*, where d is the number of features and n is the number of examples
- We add a column of ones to capture an intercept

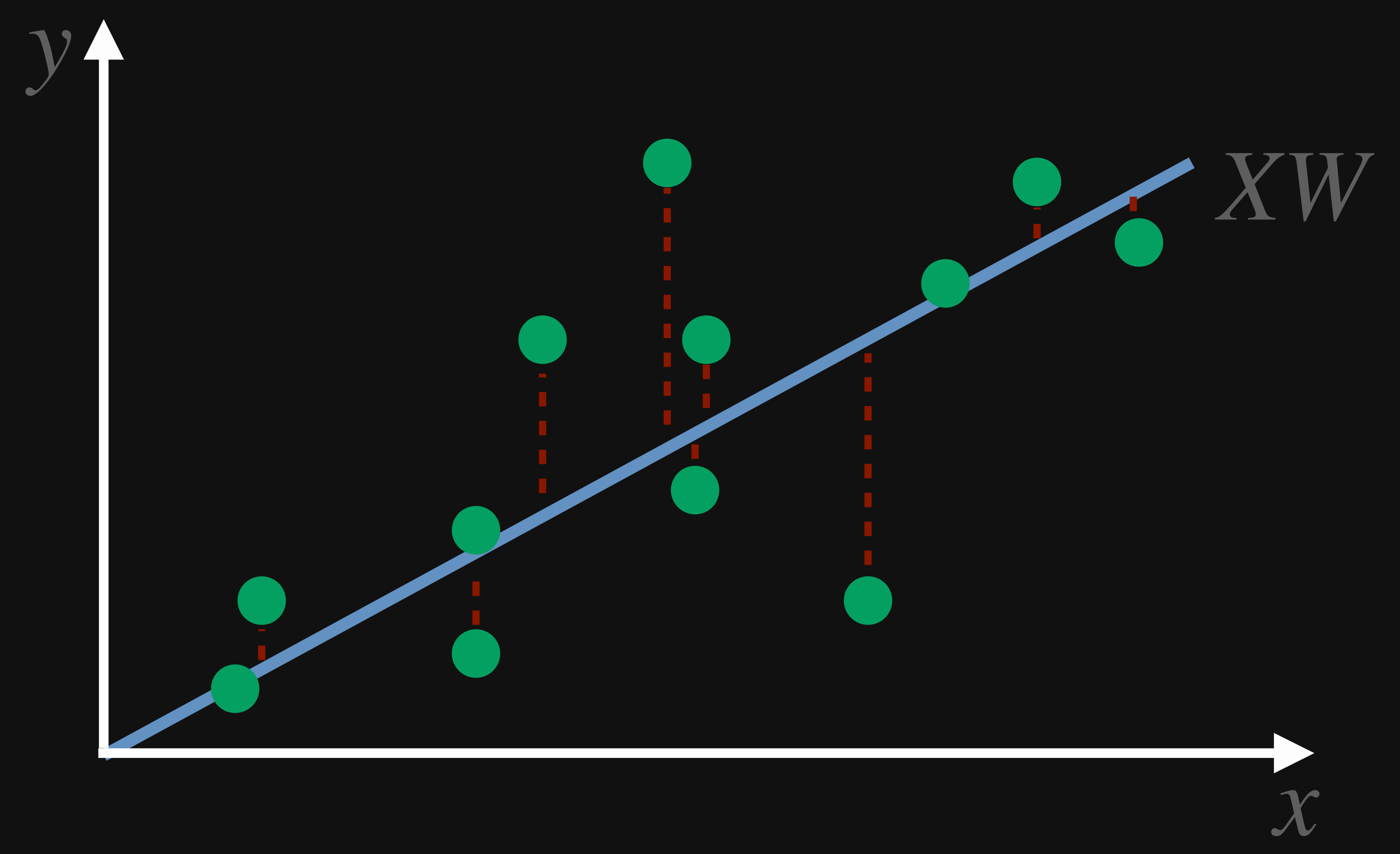
$$= \begin{array}{cc} \begin{bmatrix} 1 & \cdots & x_{1d} \\ \vdots & \ddots & \\ 1 & & x_{nd} \end{bmatrix} & n \\ d+1 & \end{array} \qquad W = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix} d+1 \qquad \hat{y} = \begin{bmatrix} \hat{y}_0 \\ \vdots \\ \hat{y}_n \end{bmatrix} n \qquad \hat{y} = XW$$

Recap: Loss Functions

- Loss function measures deviation of the model's prediction from the ground truth.
- Allows to evaluate the fit of a machine learning model.
- MSE is defined as the average sum of the squared differences between the prediction and the ground truth.

$$RSS = \frac{1}{2}\sum_{i=1}^{n}(y - XW)^2$$

$$MSE = \frac{1}{n}RSS$$



Recap: Optimization using Normal Equations

- The gradient of RSS can be defined as follows:

$$
\begin{aligned}
\nabla_W RSS &= \frac{1}{2}\nabla_W (y - XW)^T(y - XW) \\
&= \frac{1}{2}\nabla_W \left((XW)^T(XW) - y(XW)^T - (XW)^T y + y^T y\right) \\
&= \frac{1}{2}\nabla_W \left(W^T X^T XW - 2(XW)^T y + y^T y\right) \\
&= \frac{1}{2}\left(2(X^T X)W - 2X^T y\right) \\
&= (X^T X)W - X^T y
\end{aligned}
$$

$(AB)^T = B^T A^T$

$a^T b = b^T a$

$\nabla_x x^T Ax = 2Ax$ for a symmetric matrix A

$\nabla_x b^T x = b$

Optimization using Normal Equations

- The derivative becomes 0 at the minimum of a function.
- Since $RSS(W)$ is a quadratic function it will only have one minimum.
- If we solve the above expression for W we will get an expression for the minimum of our MSE loss function:

$$(X^T X)W - X^T y = 0$$

$$(X^T X)W = X^T y$$

- Hence the value $W*$ that minimises the objective is given by:

$$W* = (X^T X)^{-1} X^T y = X^\dagger y$$

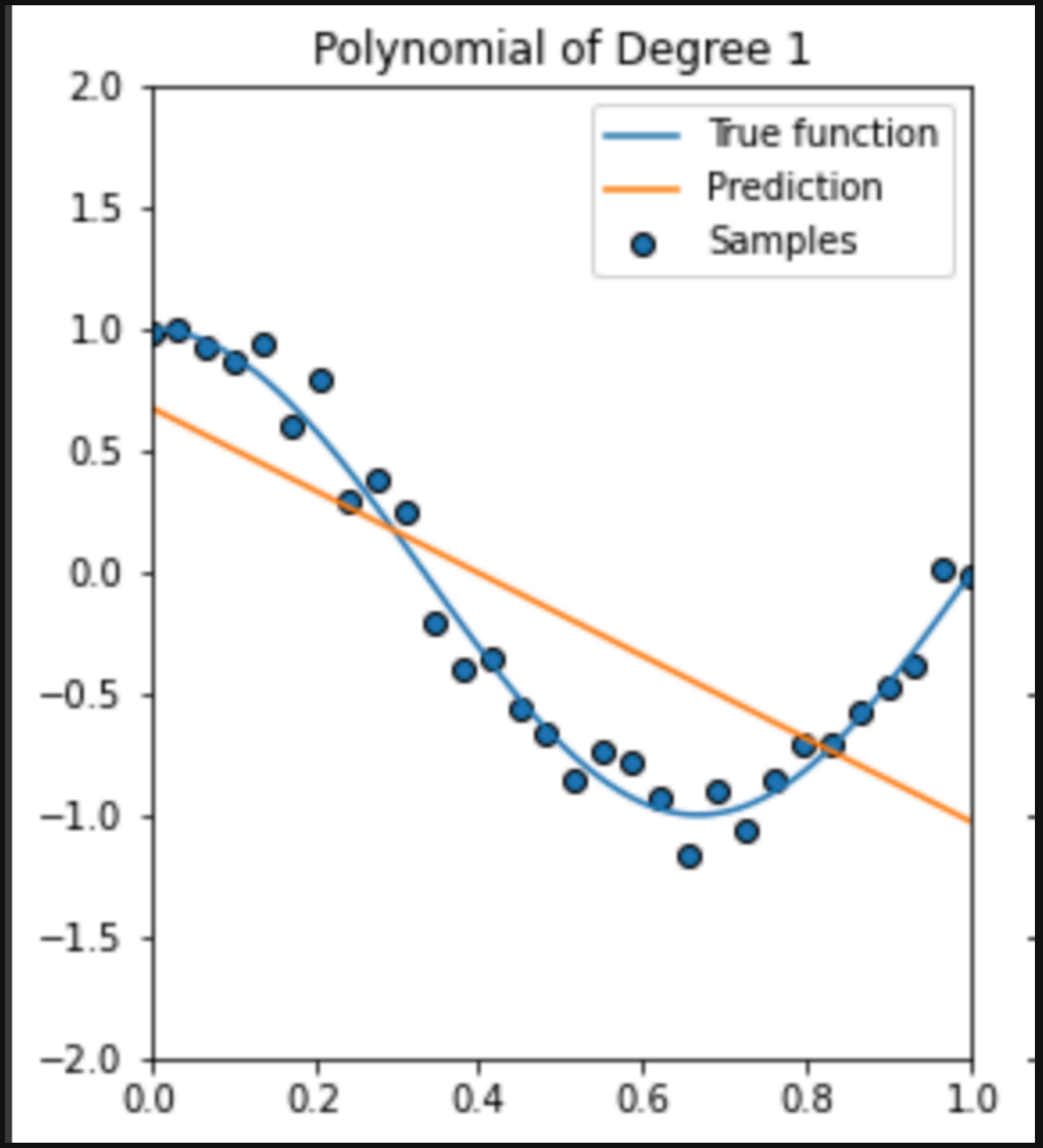
Polynomial Regression

- In practice, the data will often have a non-linear relationship with the targets.
- We can use polynomial regression to model more complex relationships.
- For example, if we have two features x_1, x_2 and we use a polynomial of degree 2, the prediction will be defined by:

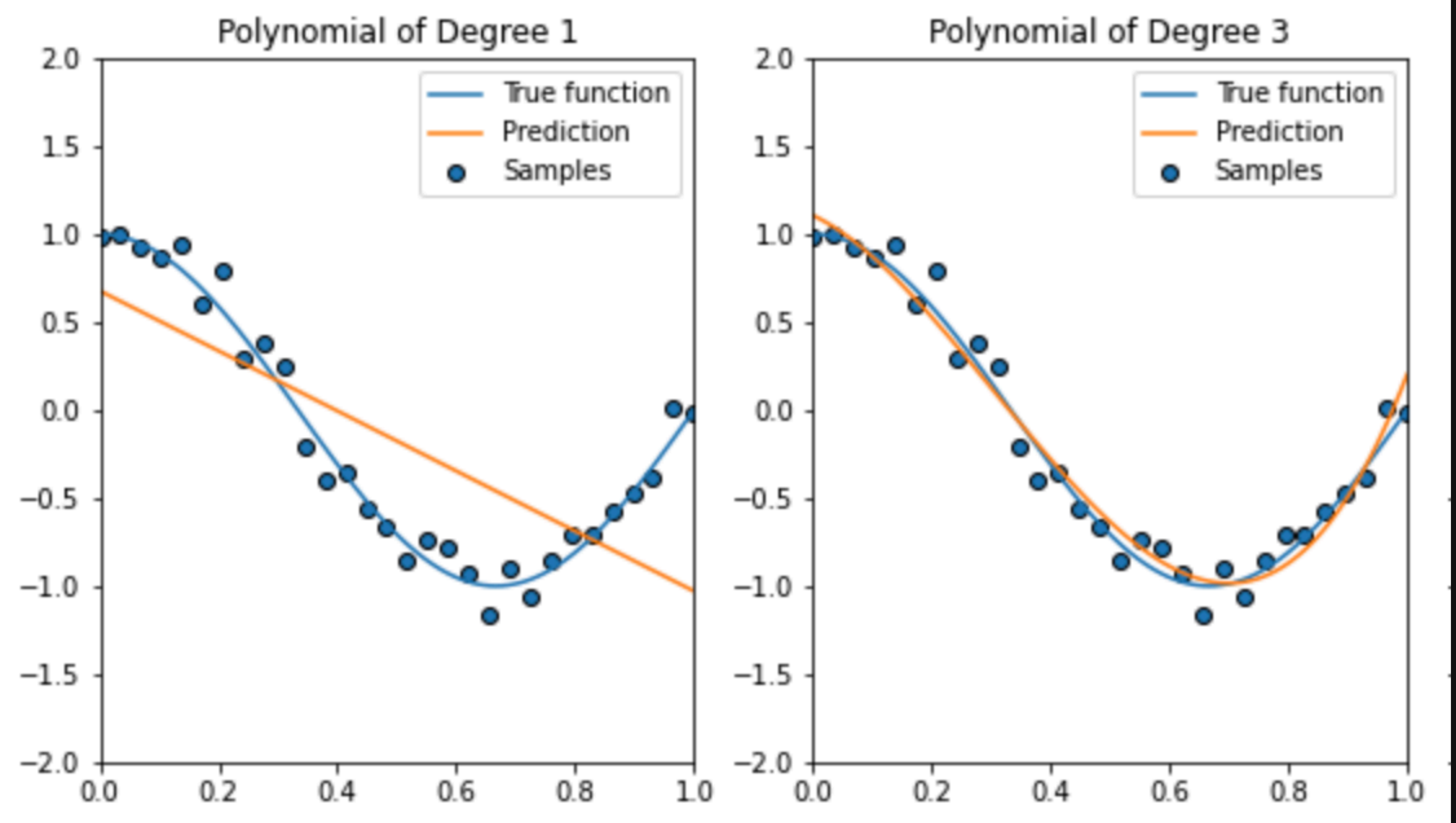
$$f(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2$$

- Important: The model will be highly non-linear in x, but still linear in W !

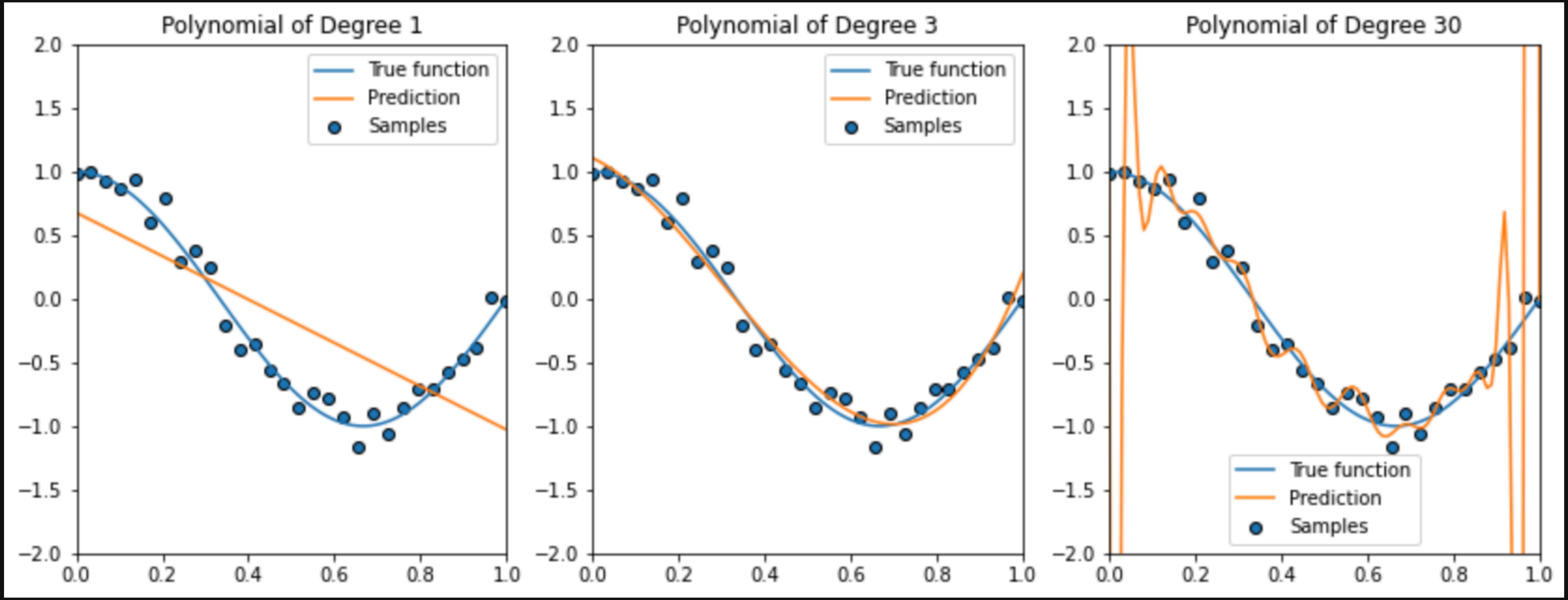
Polynomial Regression



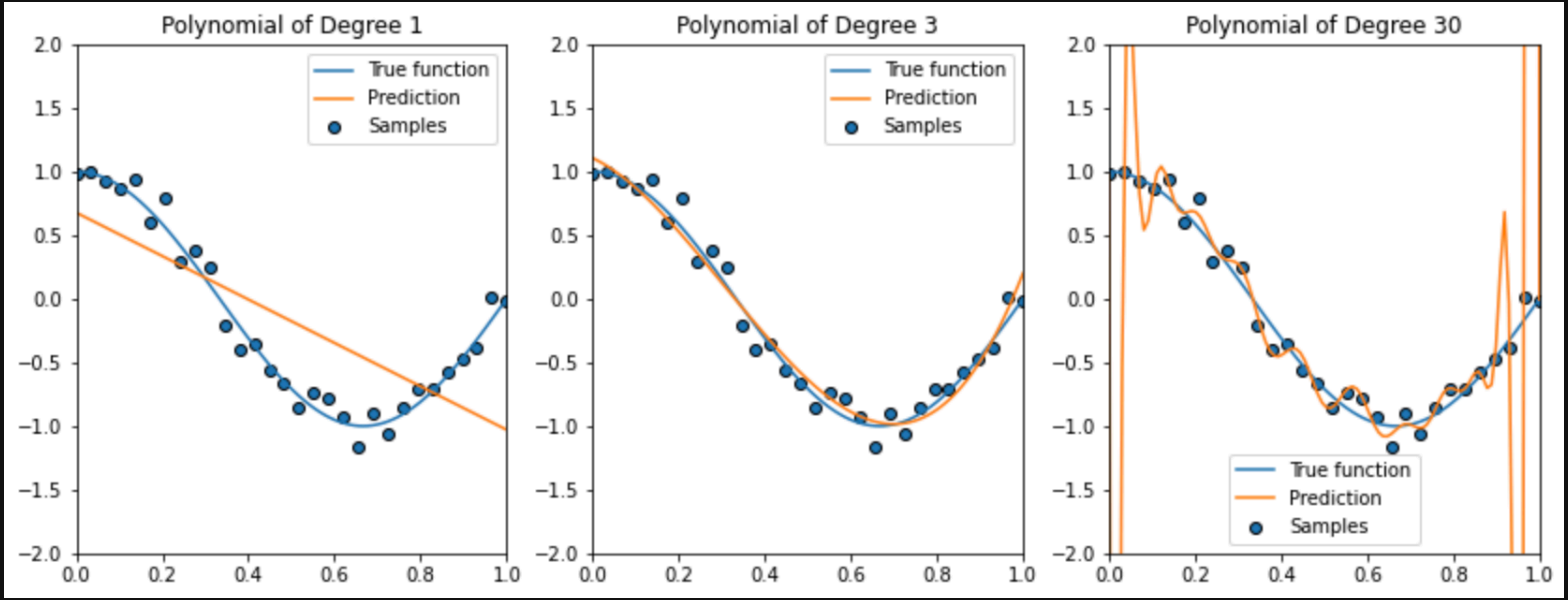
Polynomial Regression



Polynomial Regression

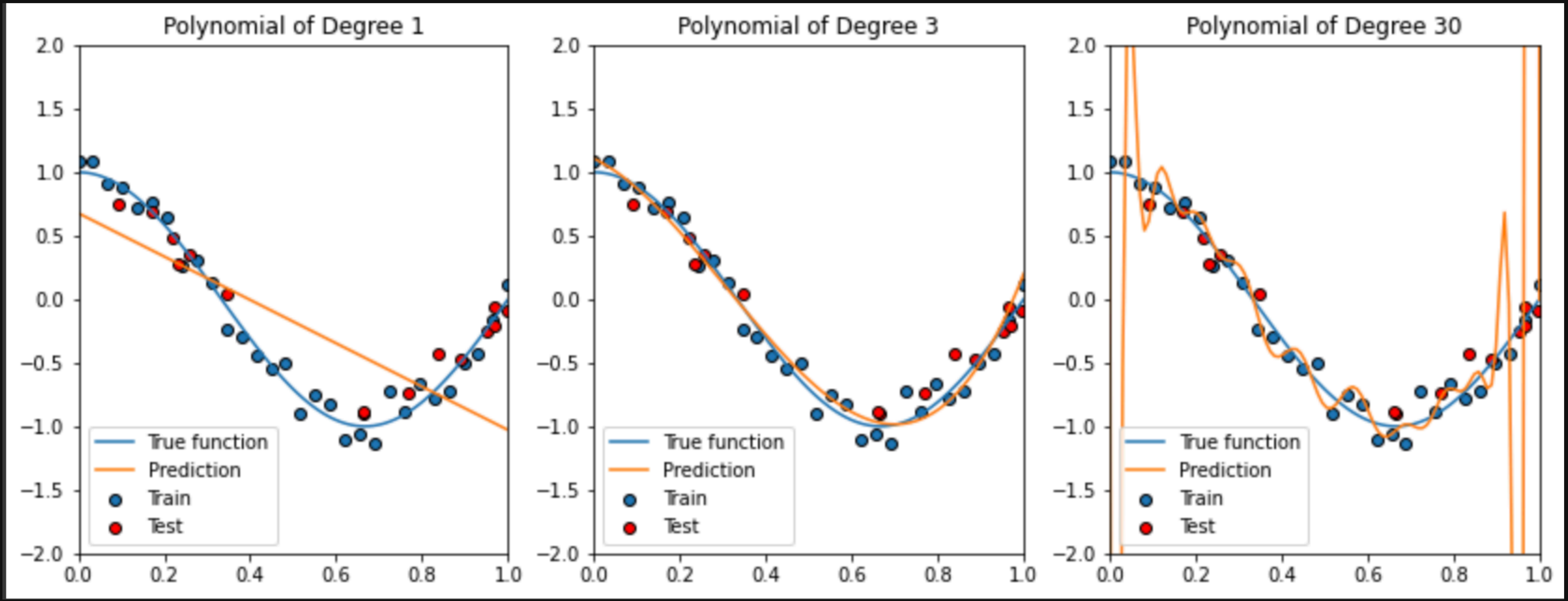


Polynomial Regression

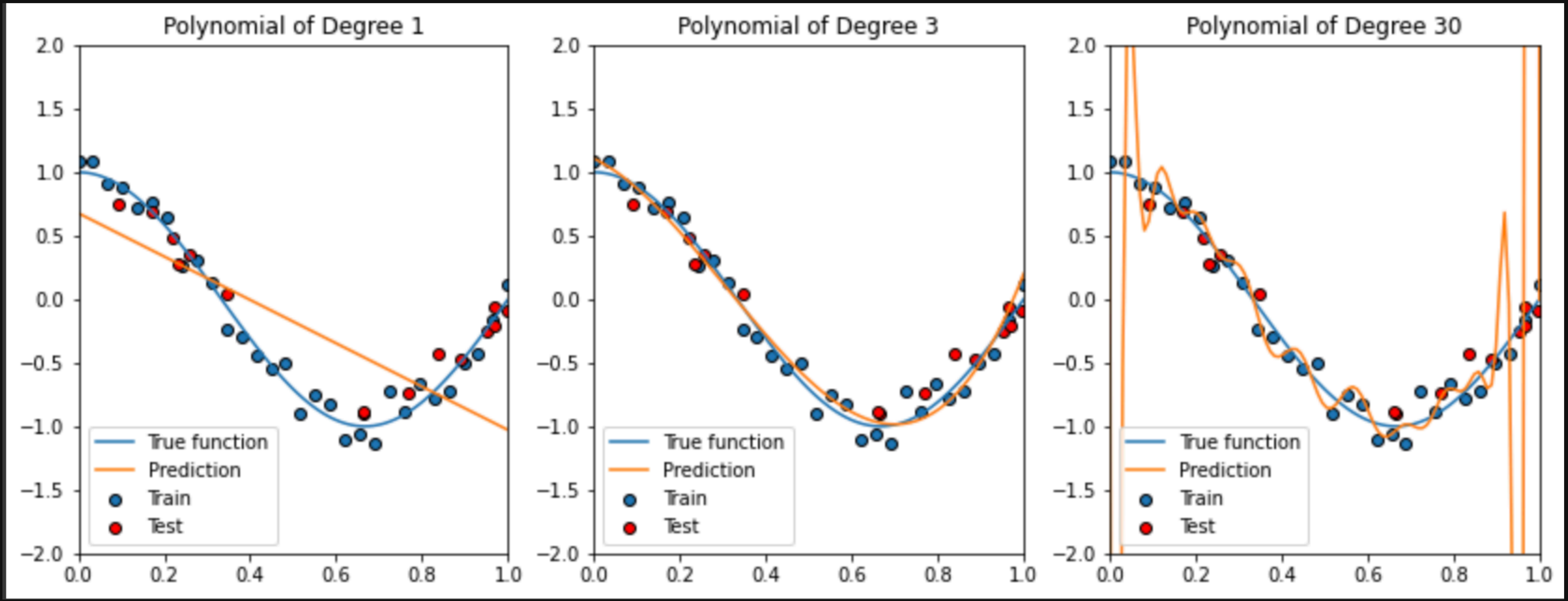


$MSE_{train} = 0.21$ $MSE_{train} = 0.02$ $MSE_{train} = 0.01$

Polynomial Regression

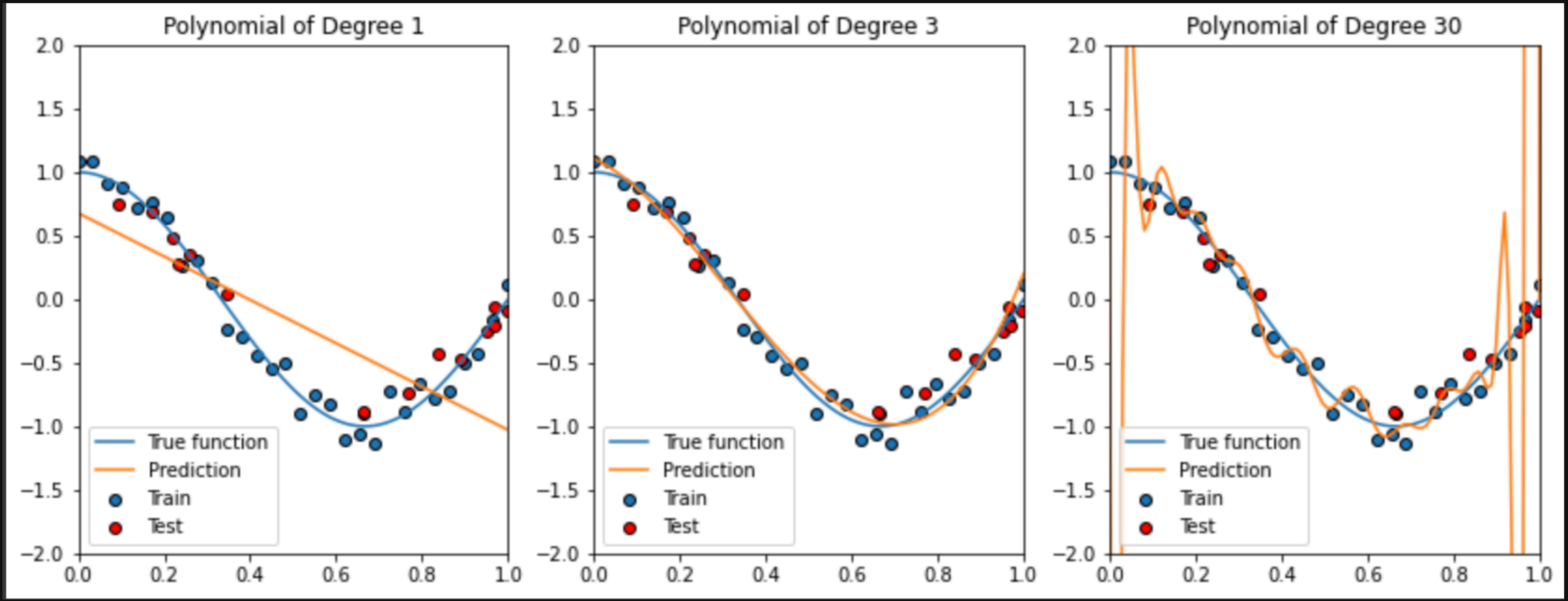


Polynomial Regression



$MSE_{train} = 0.21$ $MSE_{train} = 0.02$ $MSE_{train} = 0.01$

Polynomial Regression



$MSE_{train} = 0.21$

$MSE_{test} = 0.24$

$MSE_{train} = 0.02$

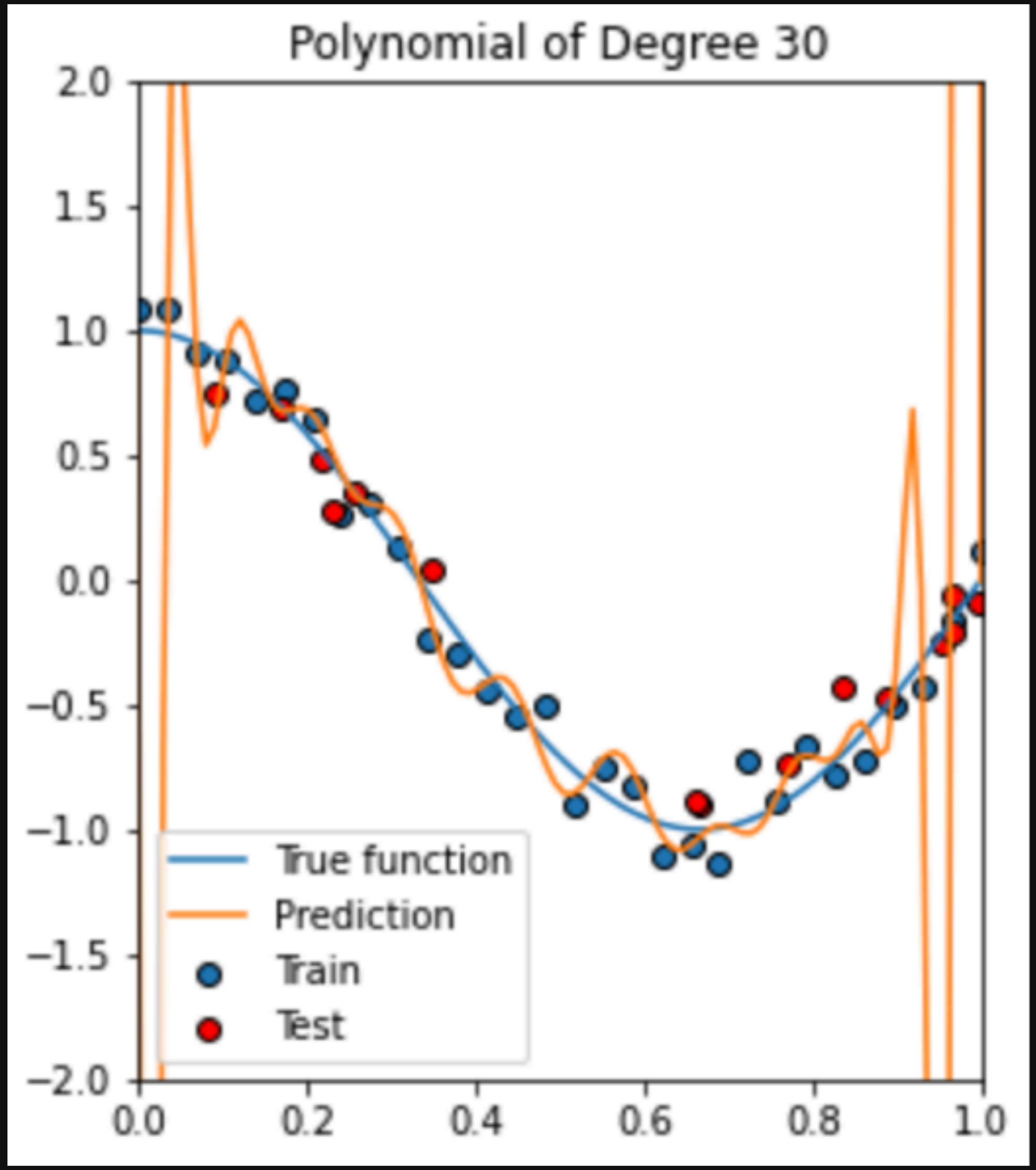
$MSE_{test} = 0.04$

$MSE_{train} = 0.01$

$MSE_{test} = 669$

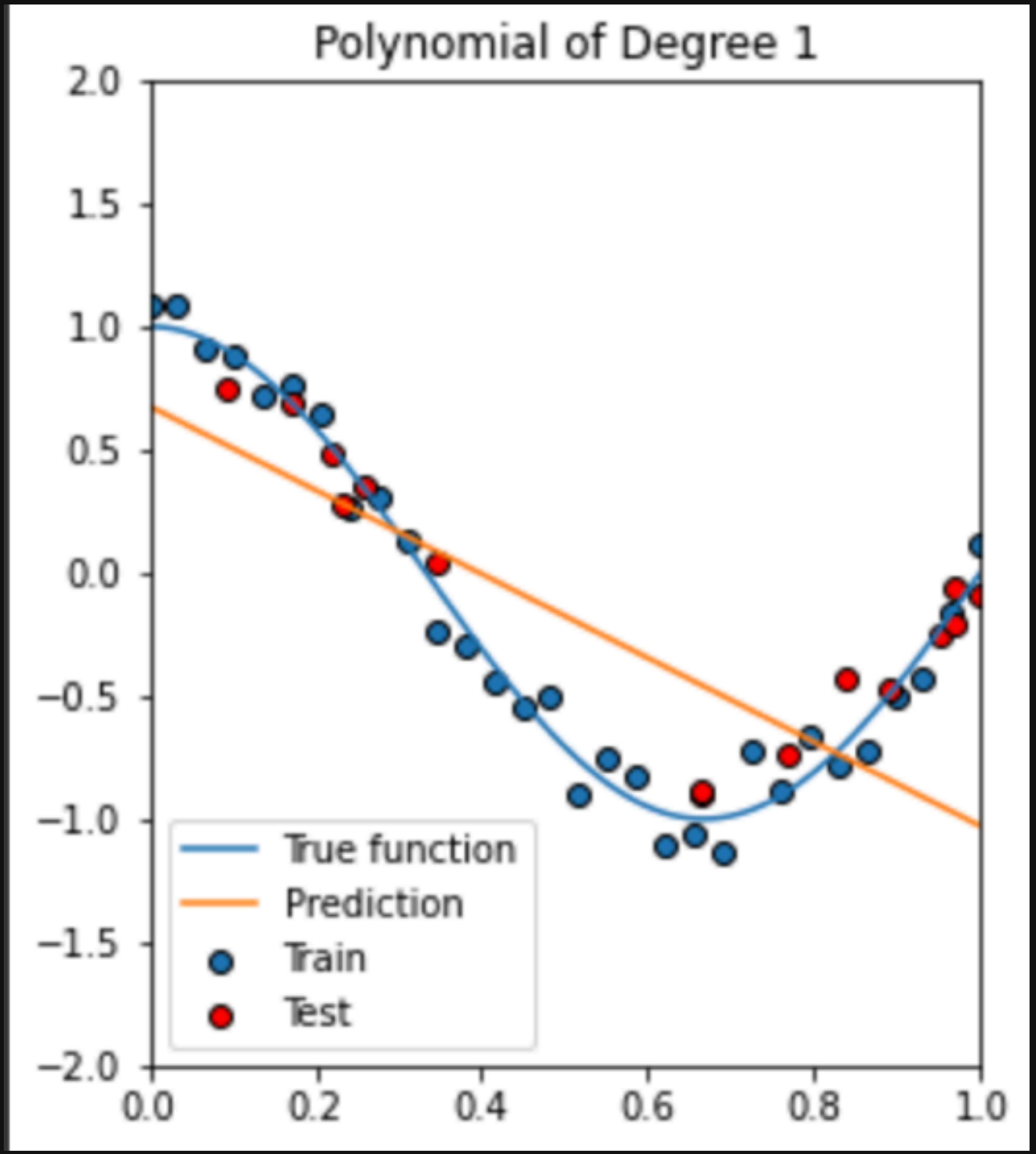
Overfitting

- A very expressive model fits the training dataset perfectly.
- The model also makes wildly incorrect prediction outside the dataset and doesn't generalize.
- Dealing with Overfitting:
  - Reduce the complexity class of the model (going from polynomial to linear)
  - Modify the loss function to penalise complex models that may overfit the data



Underfitting

- A small model (e.g. a straight line) will not fit the training data well.
- For held-out data it will not be accurate neither.
- Dealing with Underfitting:
  - Increase model complexity class
  - Create richer features that will make the dataset easier to fit



Regularization

- The idea of regularization is to penalize models that may overfit the data
- This could be done by changing the objective to include a term that penalizes complex models
- $J(W) = \frac{1}{2}(X\theta - y)^T(X\theta - y) + \frac{1}{2}\lambda||\theta||_2^2$
- The first part is a usual loss function, such as MSE.
- The second part is a regularizerthat penalizes models that are overly complex.
- A regularization coefficient $\lambda > 0$ controls the strength of a regularizer.

Regularization

- The derivative can then be calculated as:

$$
\begin{aligned}
\nabla_W J(W) &= \nabla_W\left(\frac{1}{2}(XW - y)^T(XW - y) + \lambda\|W\|_2^2\right) \\
&= \nabla_W\left(RSS(W) + \frac{1}{2}\lambda\|W\|_2^2\right) \\
&= \nabla_W RSS(W) + \lambda W \\
&= (X^T X)W - X^T y + \lambda W \\
&= (X^T X + \lambda I)W - X^T y
\end{aligned}
$$

The value of W that minimises the loss will then be: $W* = (X^T X + \lambda I)^{-1} + X^T y$

Credits

- https://github.com/kuleshov/cornell-cs5785-2022-applied-ml