

Machine Learning

Overfitting and Underfitting

Mostafa S. Ibrahim

Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher

PhD from Simon Fraser University - Canada

Bachelor / MSc from Cairo University - Egypt

Ex-(Software Engineer / ICPC World Finalist)

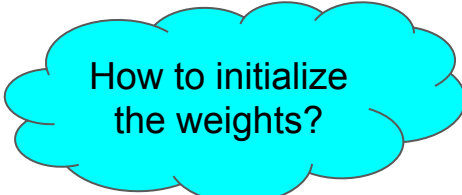


© 2023 All rights reserved.

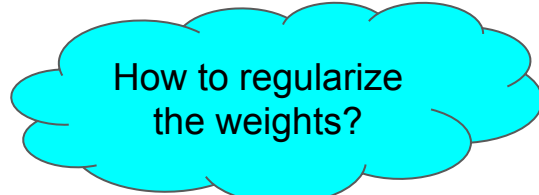
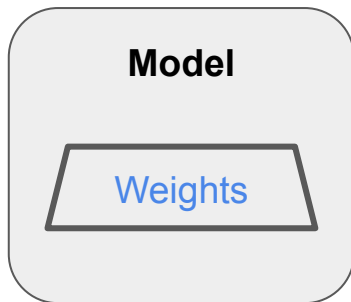
Please do not reproduce or redistribute this work without permission from the author

The weights!

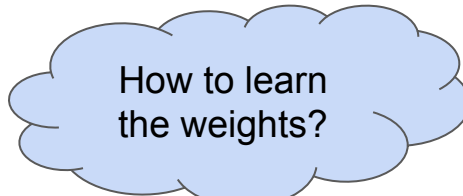
- There are 3 important research areas about the weights of a model



How to initialize
the weights?



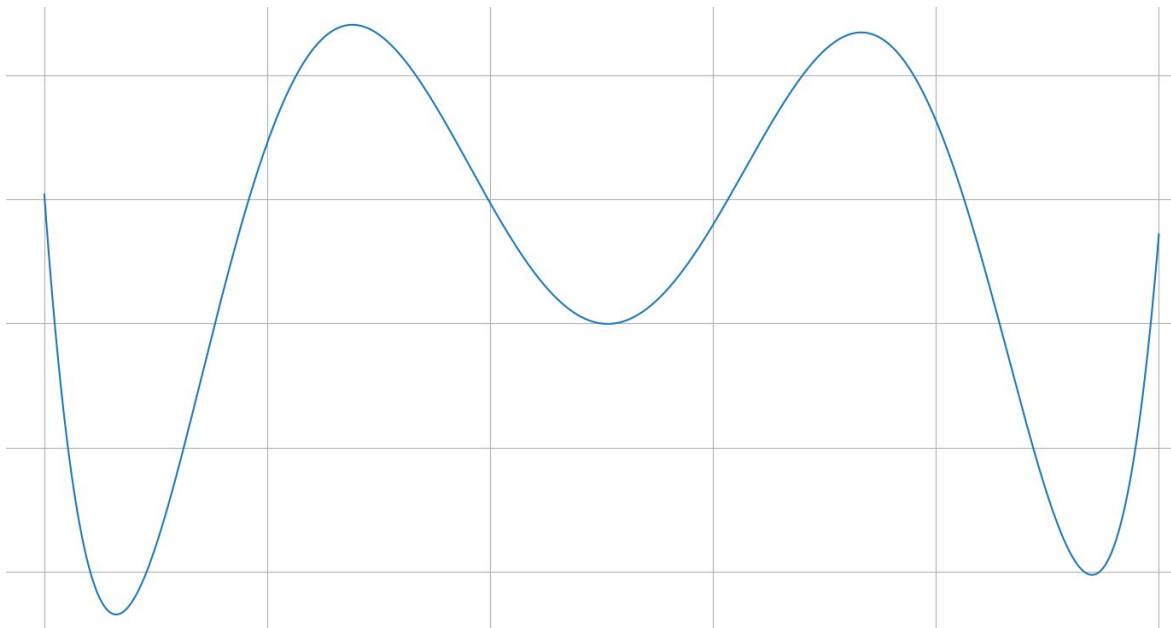
How to regularize
the weights?



How to learn
the weights?

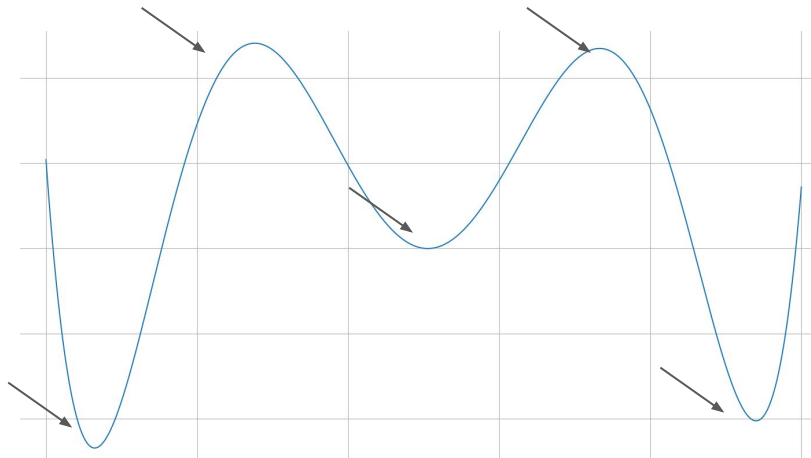
Background: Critical Points

- How many critical points are there in this graph?
- What is the minimum function degree of this graph?



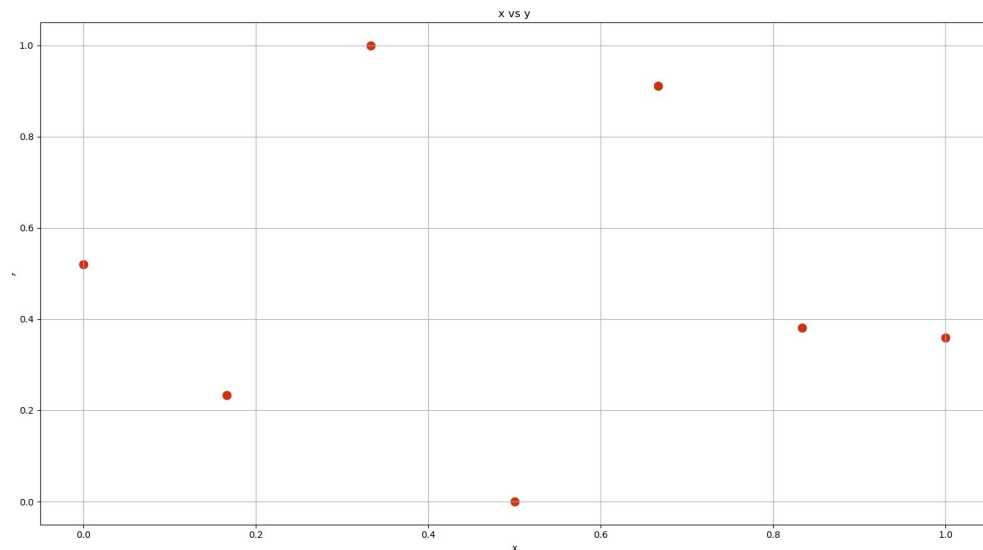
Background: Critical Points

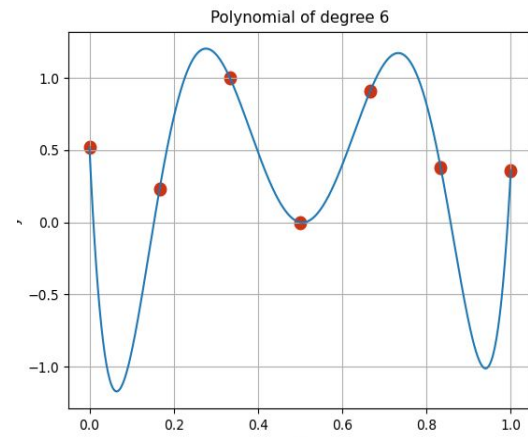
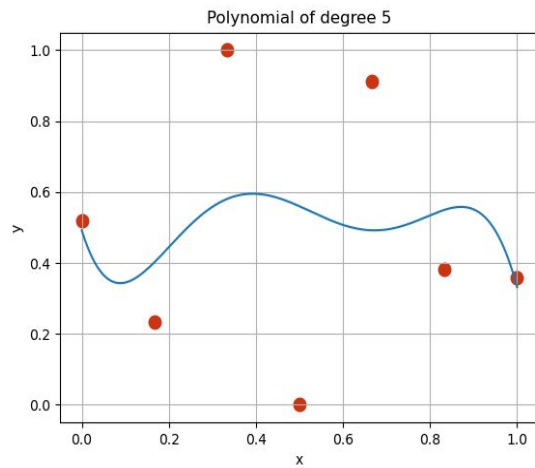
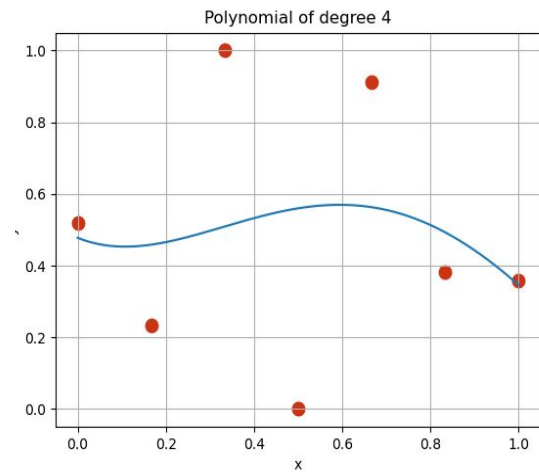
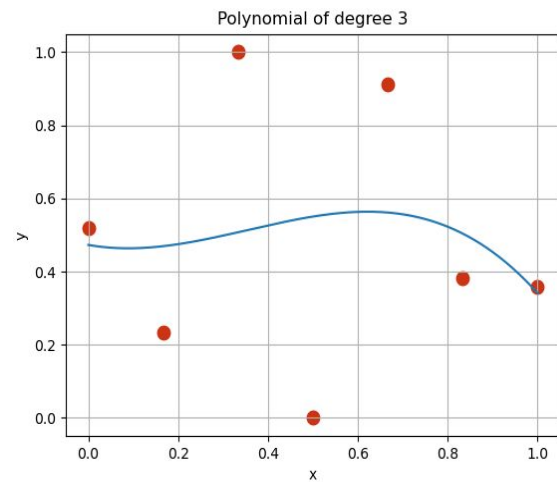
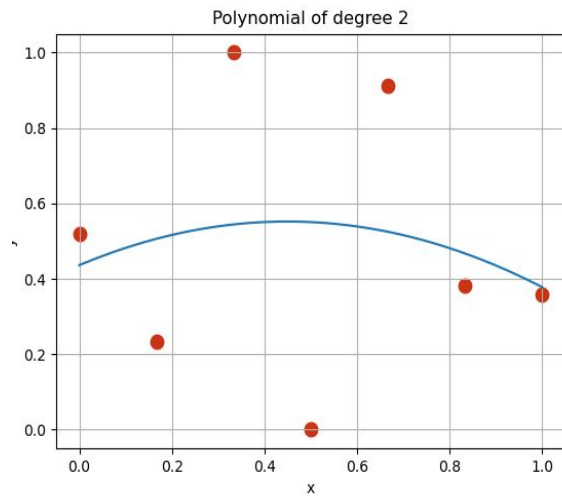
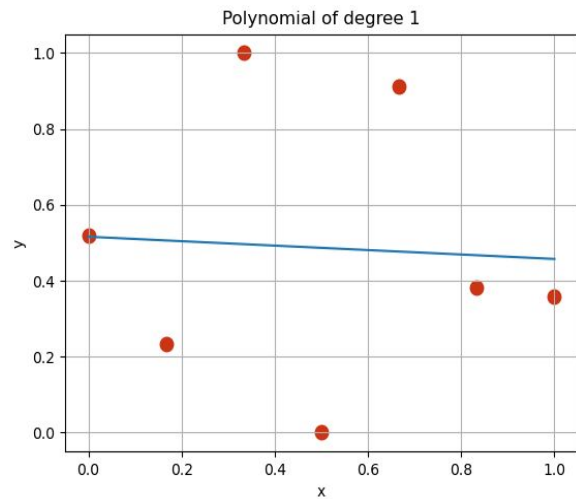
- Critical points are the points on the graph where the function's rate of change is altered (e.g. it could change from increasing to decreasing or vice versa)
 - In the given example, there are 5 critical points
- A polynomial function of degree n can have at **most** $n-1$ critical points
 - The function in this example must have a degree of at least 6



What is the best model?

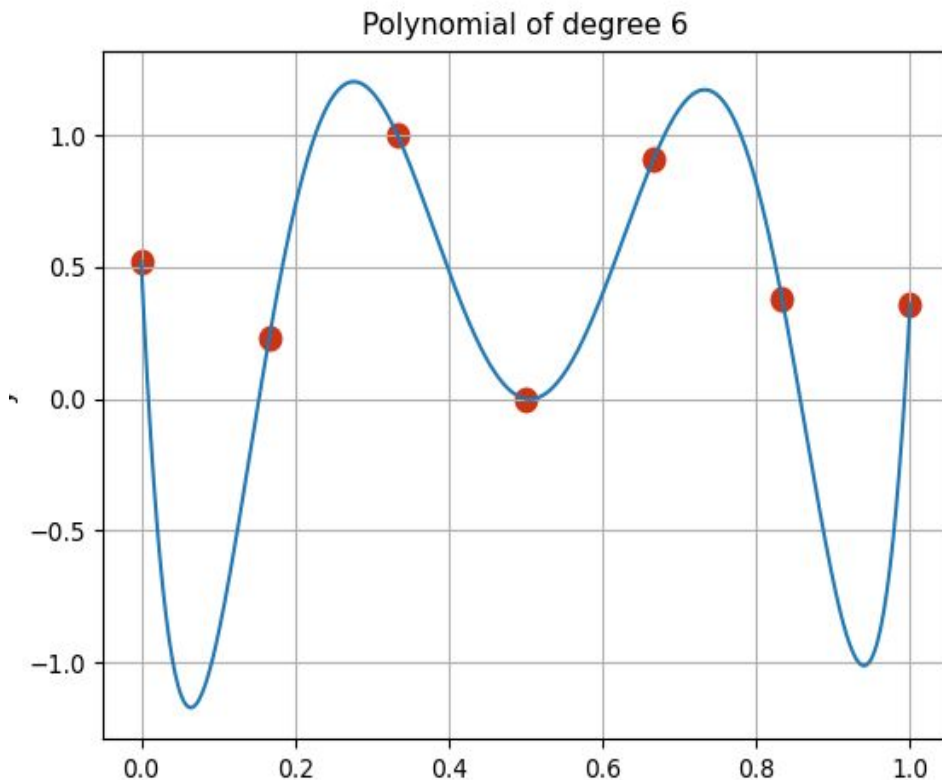
- Assume our dataset only has these 7 points
- What is the best model?
 - Line?
 - Polynomial?
 - Of which degree?!





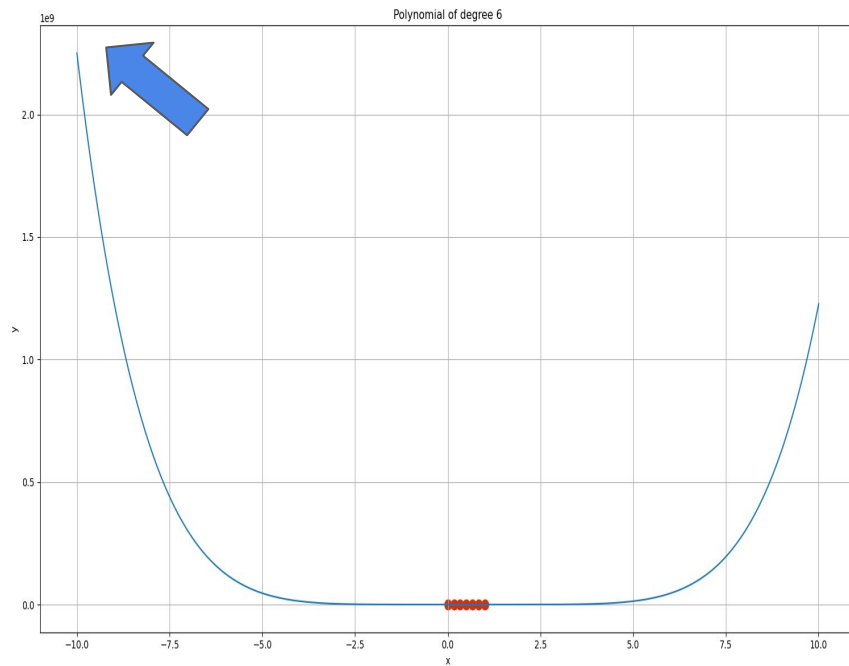
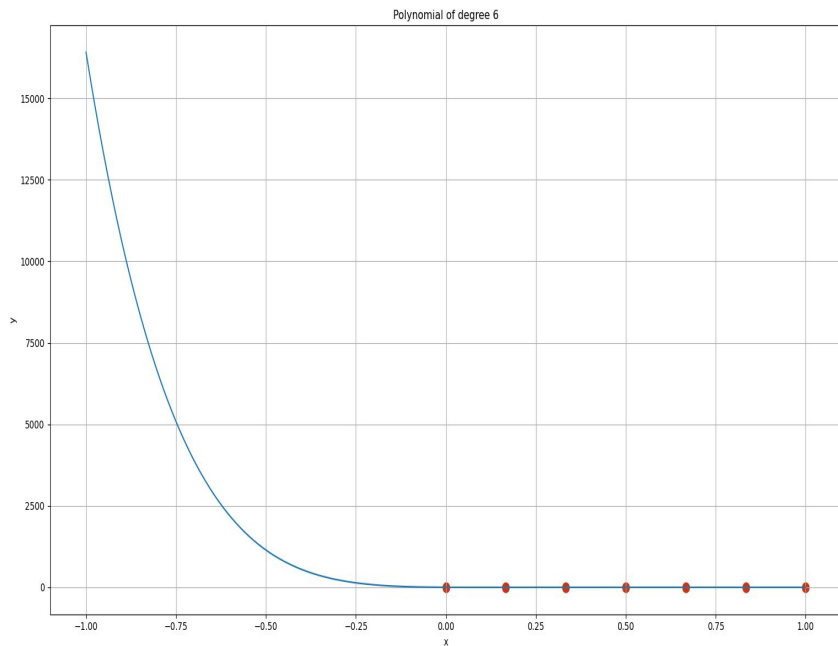
Generalization

- This model has a **training error of zero**
- However, remember our goal is **generalization**
- Do you think this model will generalize?



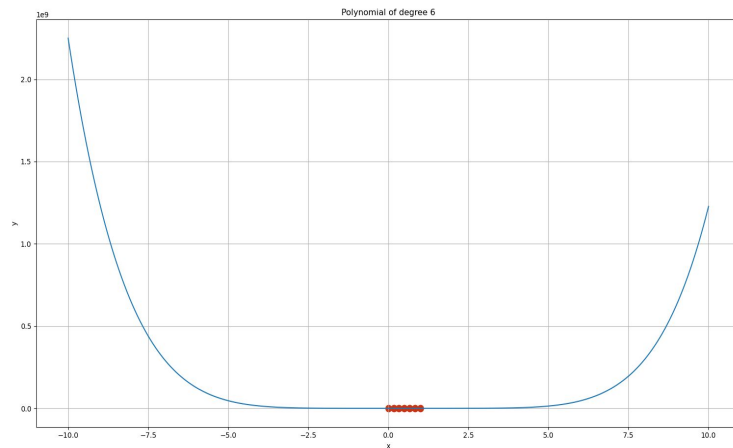
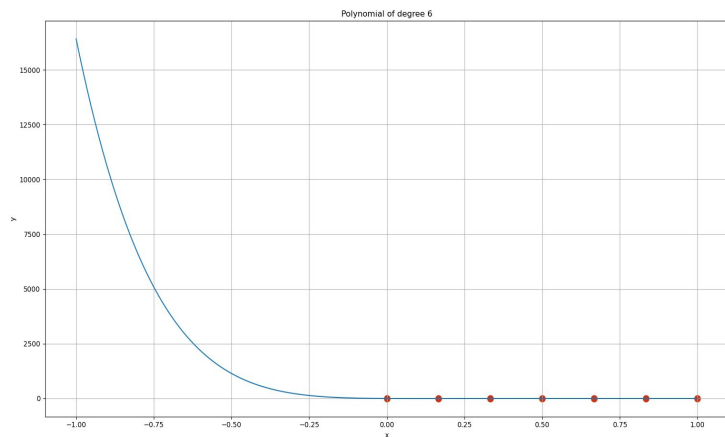
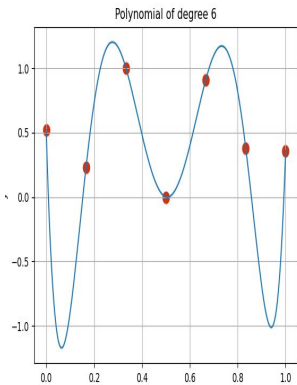
Curve Investigations

- Let's view the curve on a wider x scale: $[-1, 1]$ or $[-10, 10]$



Curve Investigations

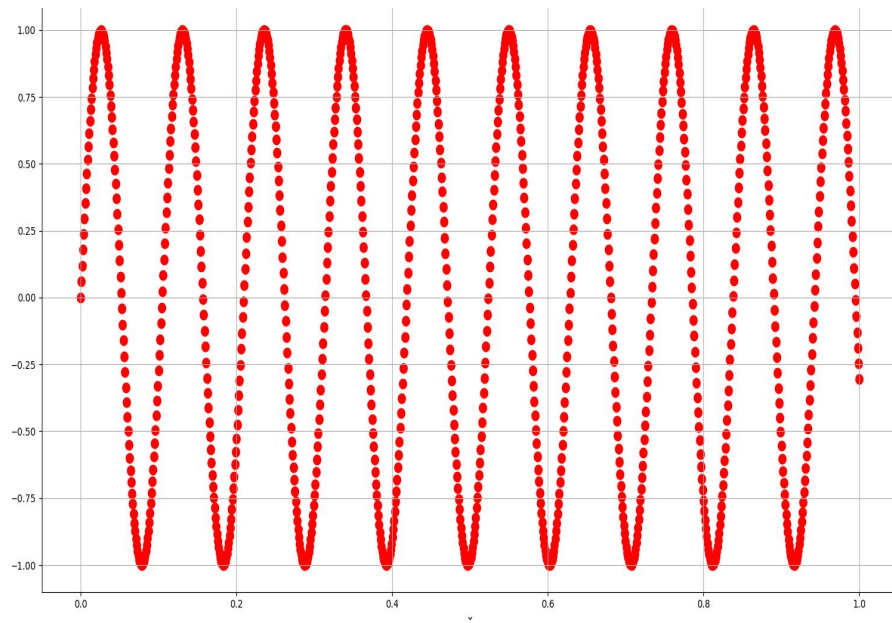
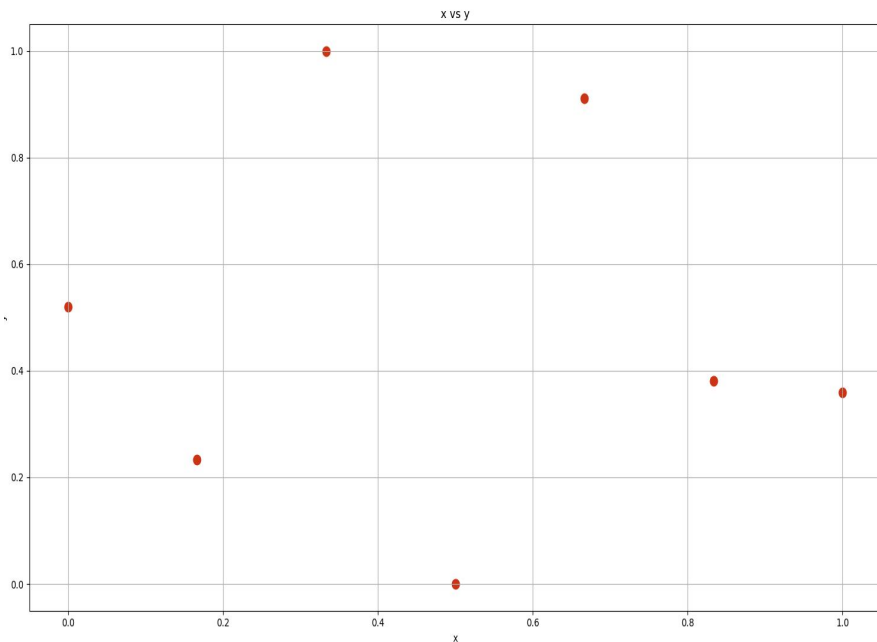
- Observe: beyond the given points, the $f(x)$ changes drastically
 - Visually, it could be tricky to see them on a large range [appears as points on a line]
- This function will completely fail to **generalize** beyond our given points, although training error is **zero**



Sample vs Population

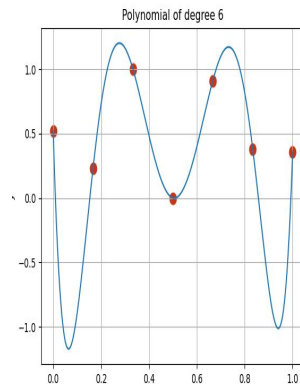
- This data is actually sampled from the $\sin(x)$ function
 - It is not a polynomial function, but its Taylor expansion is :

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$



Overfitting

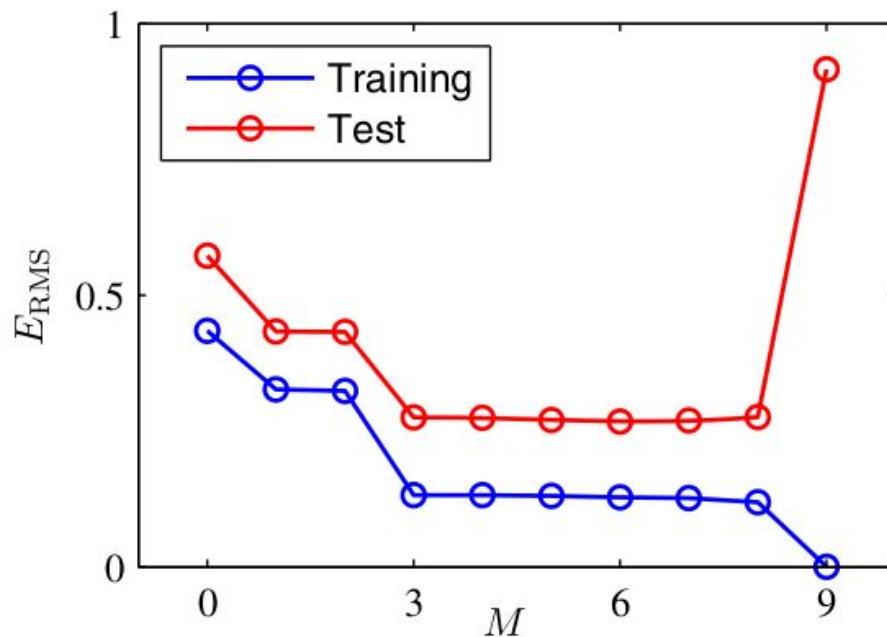
- Our sample contained only 7 noisy samples. We fitted a polynomial of 6th degree which might have 5 critical points and there is a chance some crazy polynomial is generated to **fit them perfectly**
 - Remember: some noise is always present in data, and the model absorbed it!
 - But the actual population's graph is very different to that perfect fit
- So
 - The data was **not sufficient** to capture the data patterns
 - **Our model was too complex relative to the data** and **memorized** the data rather than learning its pattern
- This phenomenon is known as **Overfitting**
 - A model that corresponds *too closely or exactly* to a particular set of data and fails to *generalize* beyond it



Train vs Validation

Figure 1.5 Graphs of the root-mean-square error, defined by (1.3), evaluated on the training set and on an independent test set for various values of M .

- How do you understand the figure?
- Once we overfitted, the train error decreases but the validation error increases

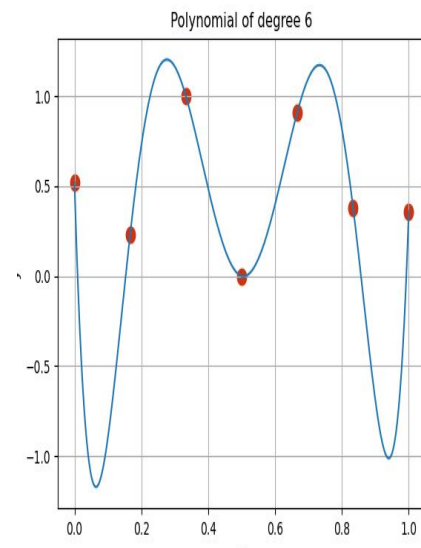


Investigating weights

- The key ingredients of our ML models are the weights/parameters
- It's important to consider the following concerns:
 - How the weights are initially initialized
 - Are they almost zero? Then they are cancelling the input features
 - Are the absolute value of the **weights too big**?
 - Then we are learning something VERY specific
 - This can lead to overfitting

Investigating weights

Polynomial Degree	MSE	Average Absolute Weight
1	0.1174	0.0234
2	0.1147	0.3446
3	0.1138	0.6222
4	0.1136	2.366
5	0.1106	49.03
6	0	2336.03
7	0	997.93
8	0	619.2
9	0	387.18



6th degree weights: [0, 61.9, 731.2, -3045.2, 5767.0, -5069.0 1677.7]

Weights toward zeros!

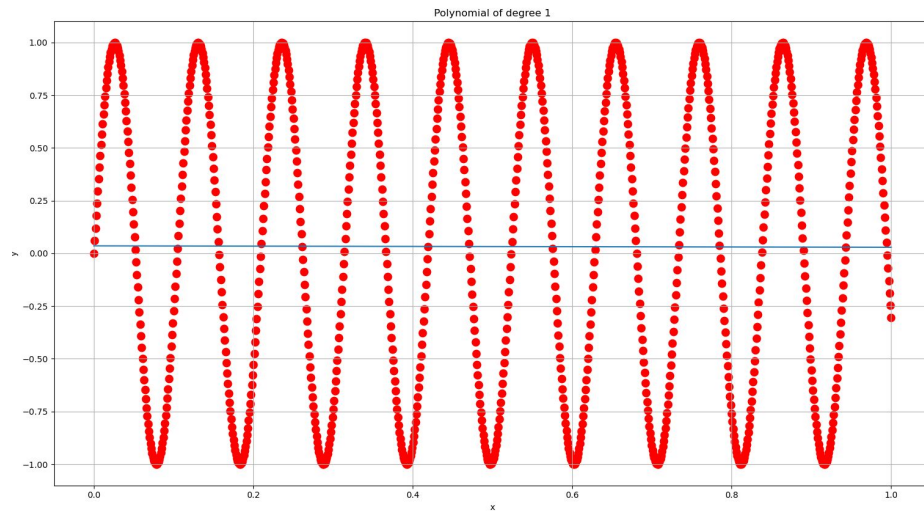
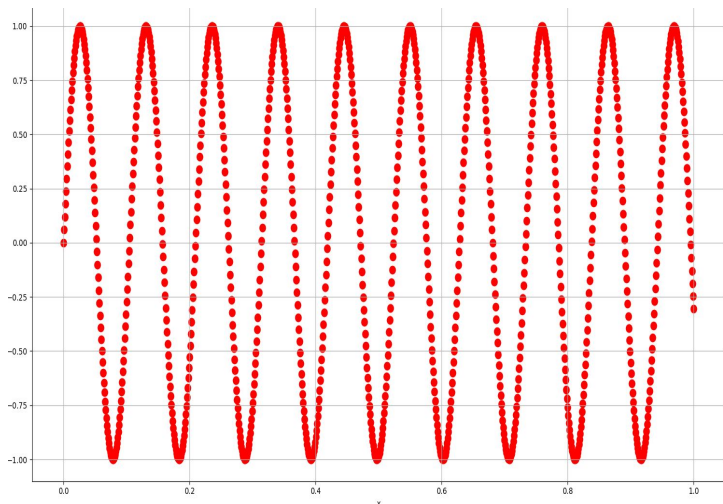
- Now we understand what happened
- The learned model learned **very large weights** to **perfectly fit** the data
- A good model should have small weights, even if it means being a bit less accurate for some examples (x, y) . Why?
- We need to push our model to find the optimal weights such that the weights are close to zero (e.g. *0.2165 NOT 6512.20*)
- This is called **weights regularization**: a **shrinkage** method that constrain the weights (another similar term: weights decay)

Why a small weight is preferred over a large weight?

- **Preventing Overfitting:** Large weights allow the network to **memorize** the training examples (too specific), leading to poor generalization.
 - By using smaller weights, the model's **capacity** is effectively **limited**, which helps in reducing overfitting and promoting better generalization to unseen data.
- **Efficient Optimization:** Gradient-based optimization algorithms rely on the **magnitude** of the **gradients** to update the weights effectively.
 - If the weights are large, the gradients can also become large, causing slow convergence or even divergence. Smaller weights help to maintain well-scaled gradients
 - This is why also we use learning rate, to keep the gradients small
- **Numerical Instability:** Using large weights can lead to numerical instability during the training process

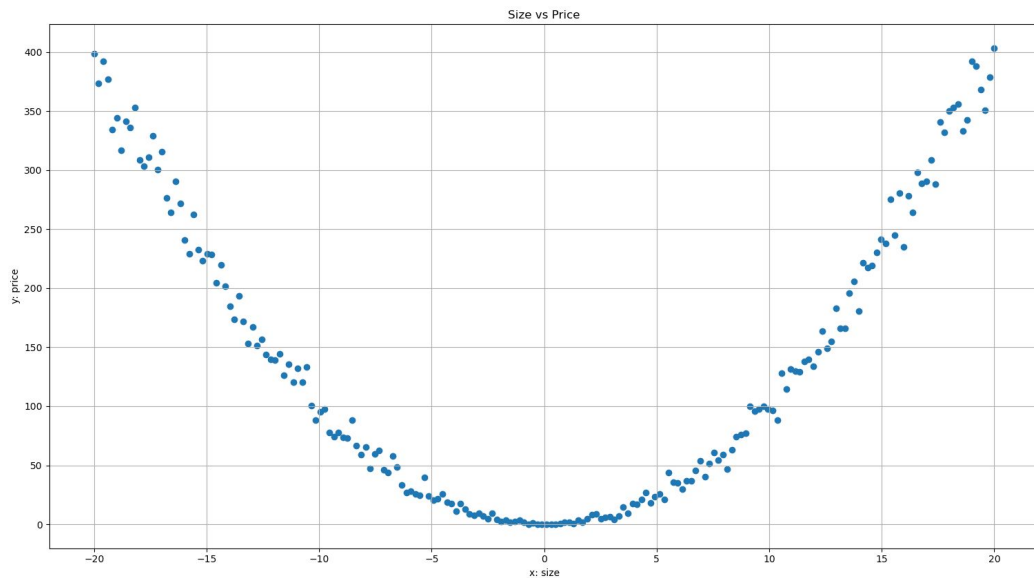
Question!

- Assume we are given the data **pattern** below and we have to use a polynomial regression, what is the best degree?
- The best thing? A line that passes through the middle of the data will have the lowest generalization error.



Question!

- Assume we have the following data. What is the best modeling for it?
- 1) A line 2) Polynomial of 2nd degree 3) Polynomial of 10th degree



- Polynomial of 2nd degree
 - Perfect fit
- Polynomial of 10th degree
 - Very complex relative to data
 - **Overfitting**
- Line
 - Too simple for data of this complexity
 - This causes **underfitting**

Underfitting

- The model is unable to capture the relationship between the input and output variables accurately
- How to detect underfitting?
 - Look for LOW prediction error on the **training** data
- How to prevent underfitting?
 - Use a more complex model relative to the data (e.g. higher polynomial degree)
 - Increase the number of features is one way increasing the model complexity
 - A good model's complexity but we need to solve other issues:
 - Avoid **too** early **stopping** of training
 - Lower the regularization penalty to reduce limitations (bias): next lecture!
 - Very noisy data can be reason

Summary

- Recall: to **measure the performance** of a model, we use a train/test split or cross-validation
- **Overfitting**: the model performs well on training data but poorly on new, unseen data (i.e. poor generalization)
 - **Why?** The model is too complex relative to the data size/diversity
 - **How** to detect? Having very low training error, but high validation error
- **Underfitting**: **Poor** performance on training data **and poor** generalization
- **Generalization** is the model's ability to give reasonable outputs to unseen data. Generalization is our **goal**
 - A good model should neither underfit nor overfit
- Always investigate your weights:
 - Are their absolute values huge?
 - Or are they close to zero?

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”

