ED5340 - Data Science: Theory and Practise

L20 - Regularization

Ramanathan Muthuganapathy (https://ed.iitm.ac.in/~raman)
Course web page: https://ed.iitm.ac.in/~raman/datascience.html
Moodle page: Available at https://courses.iitm.ac.in/

Linear Regression

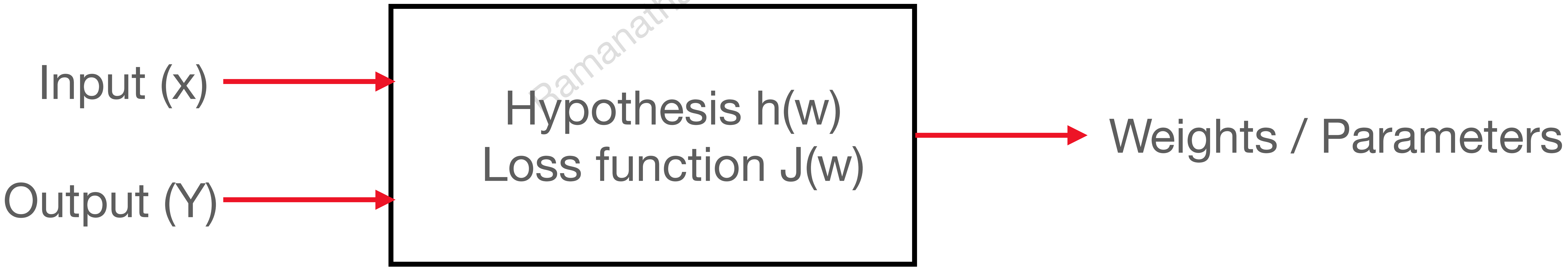
Univariate

- Ground truth data - Input feature / output $(\mathbf{x}, \mathbf{y})$ are the knowns
- Use a model / hypothesis as $h(w)$
- Develop an error / cost / loss function $J(w) = J(\mathbf{y}, \bar{\mathbf{y}}) = J(\mathbf{y}, h(w))$
- The weights are identified by
    - min $J(w)$
- Essentially, ML problem is now reduced to an optimization problem.
- Weights are identified using Optimization.

Linear Regression

Univariate case

- Ground truth data - Input feature / output $(\mathbf{x}, \mathbf{y})$ are the knowns
- Use a model / hypothesis as $h(w)$ and cost function $J(w)$
-

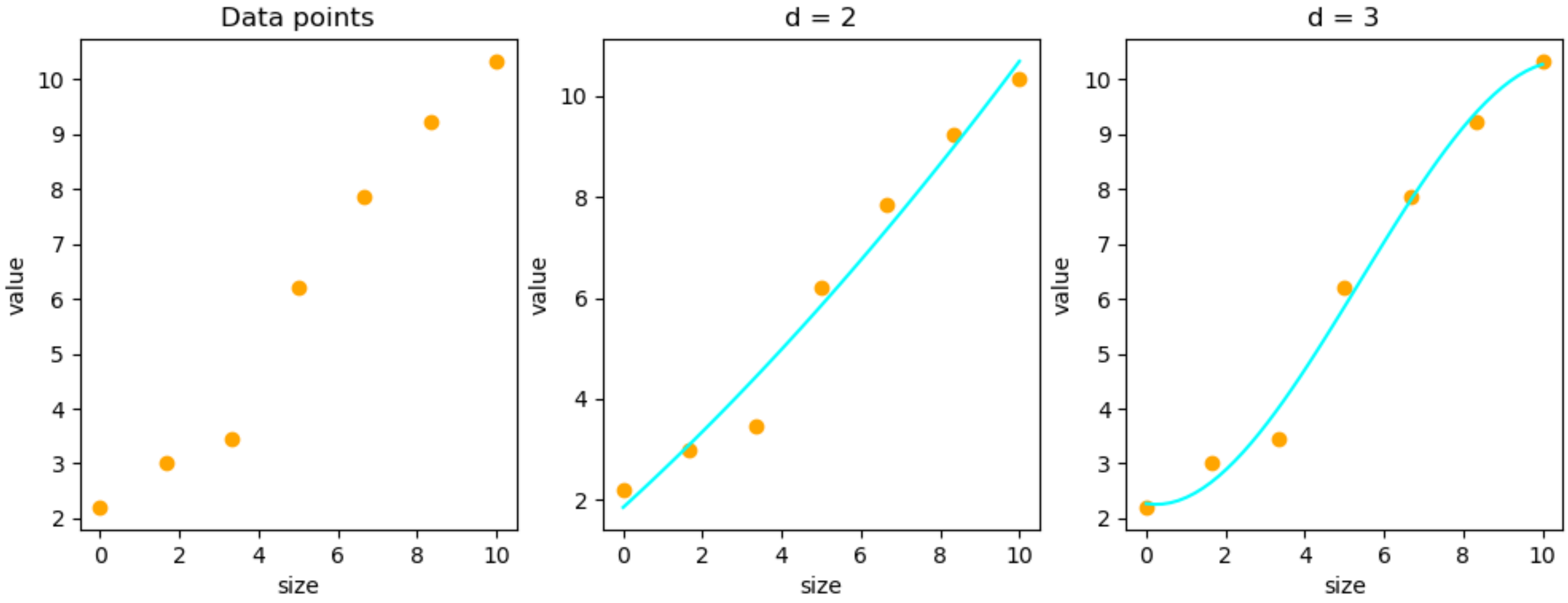


Linear Regression

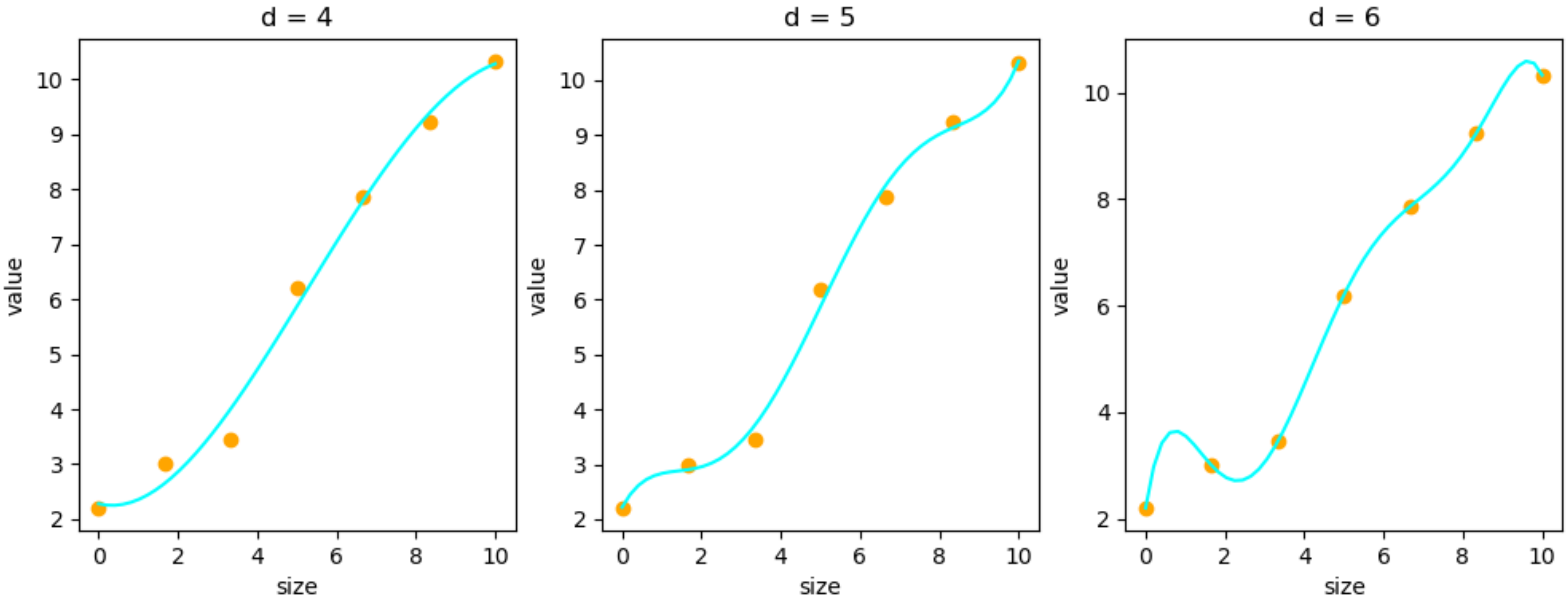
Cost function

- $J(w) = \sum_{i=1}^{m} \frac{1}{2m}(h_w(x^{(i)}) - y^{(i)})^2$

Result of varying the degree

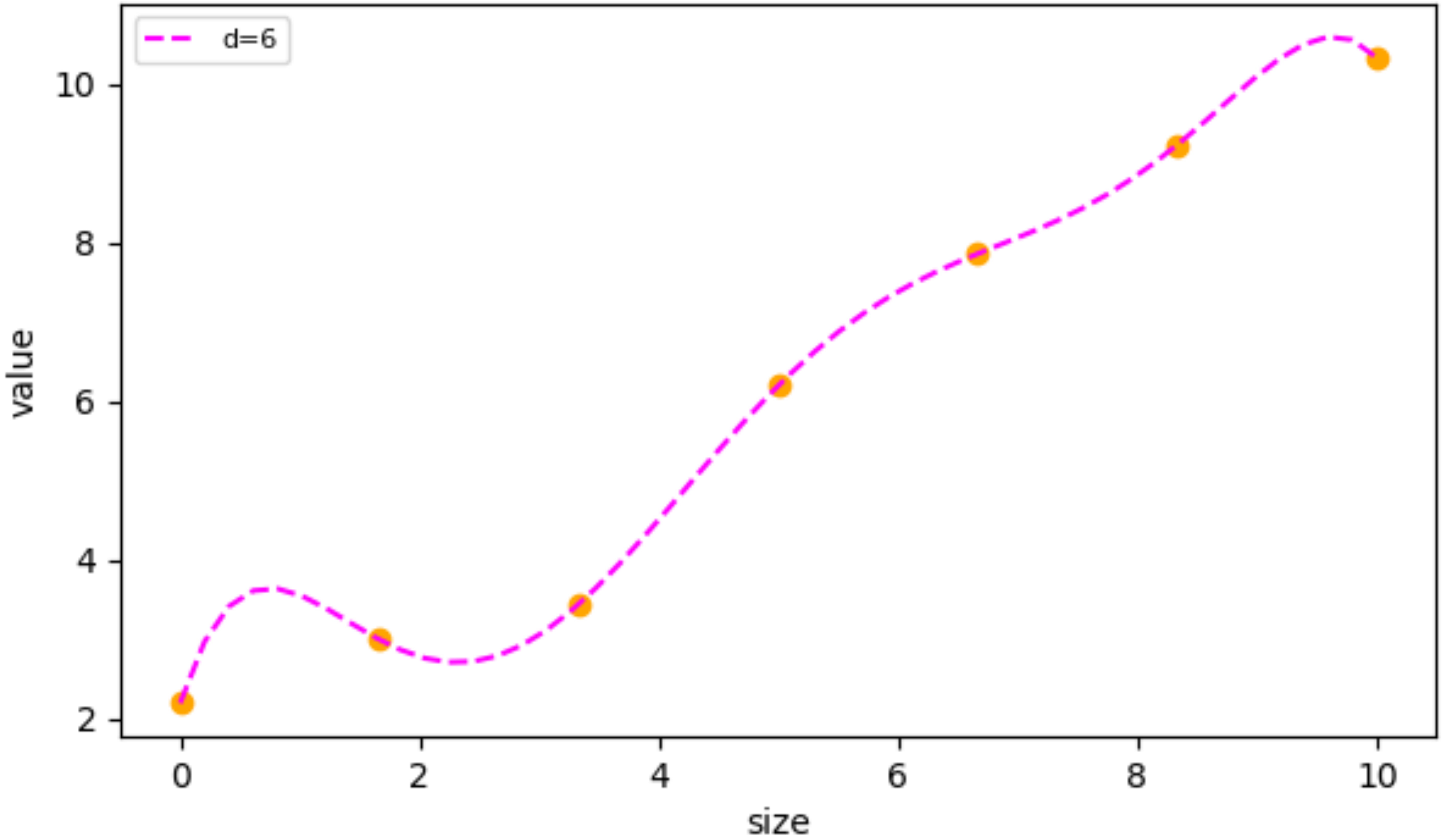


Result of varying the degree



Polynomial Regression

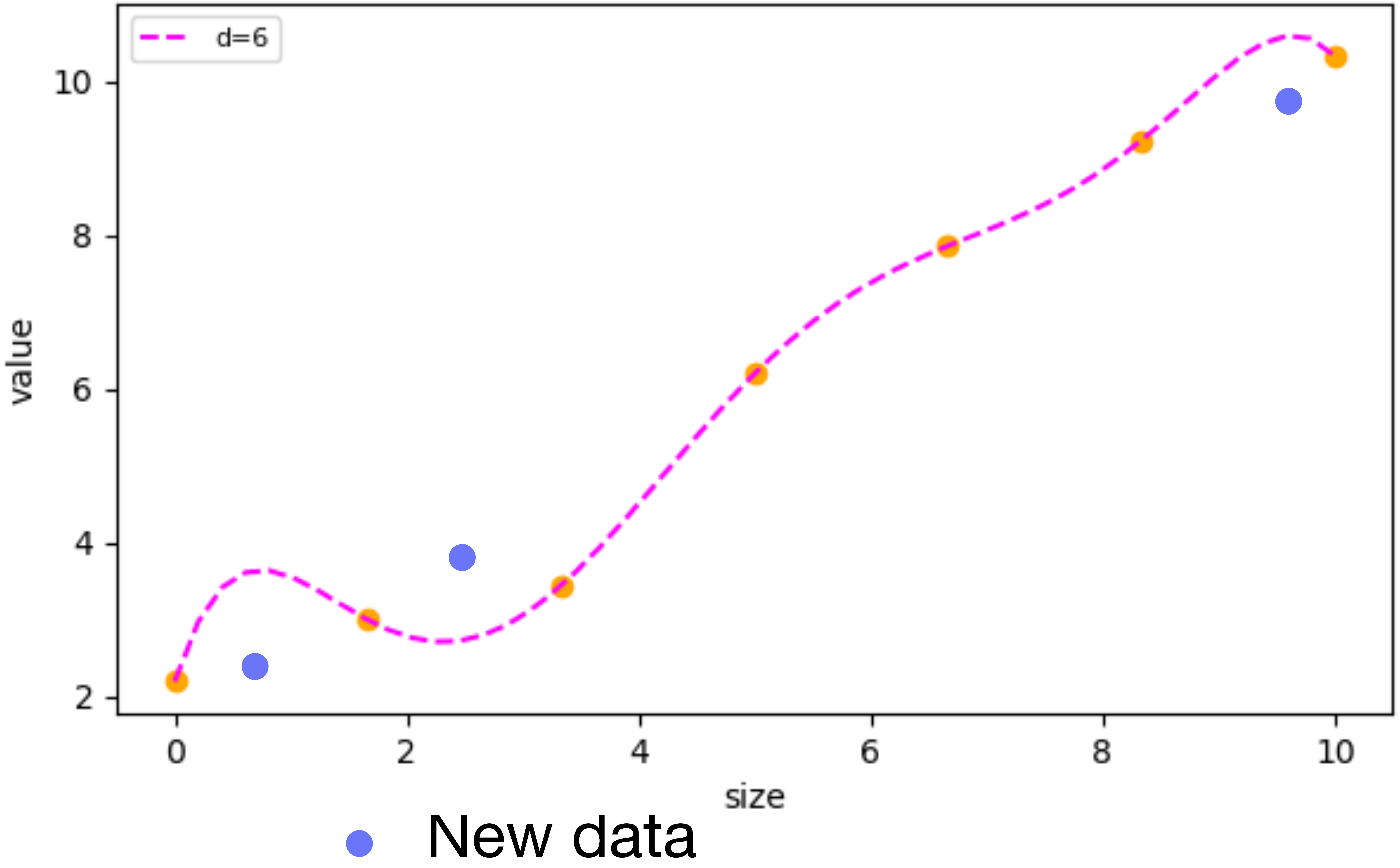
d = 6 $h_w(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$



- Great fit for the training data

Polynomial Regression

d = 6 $h_w(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$



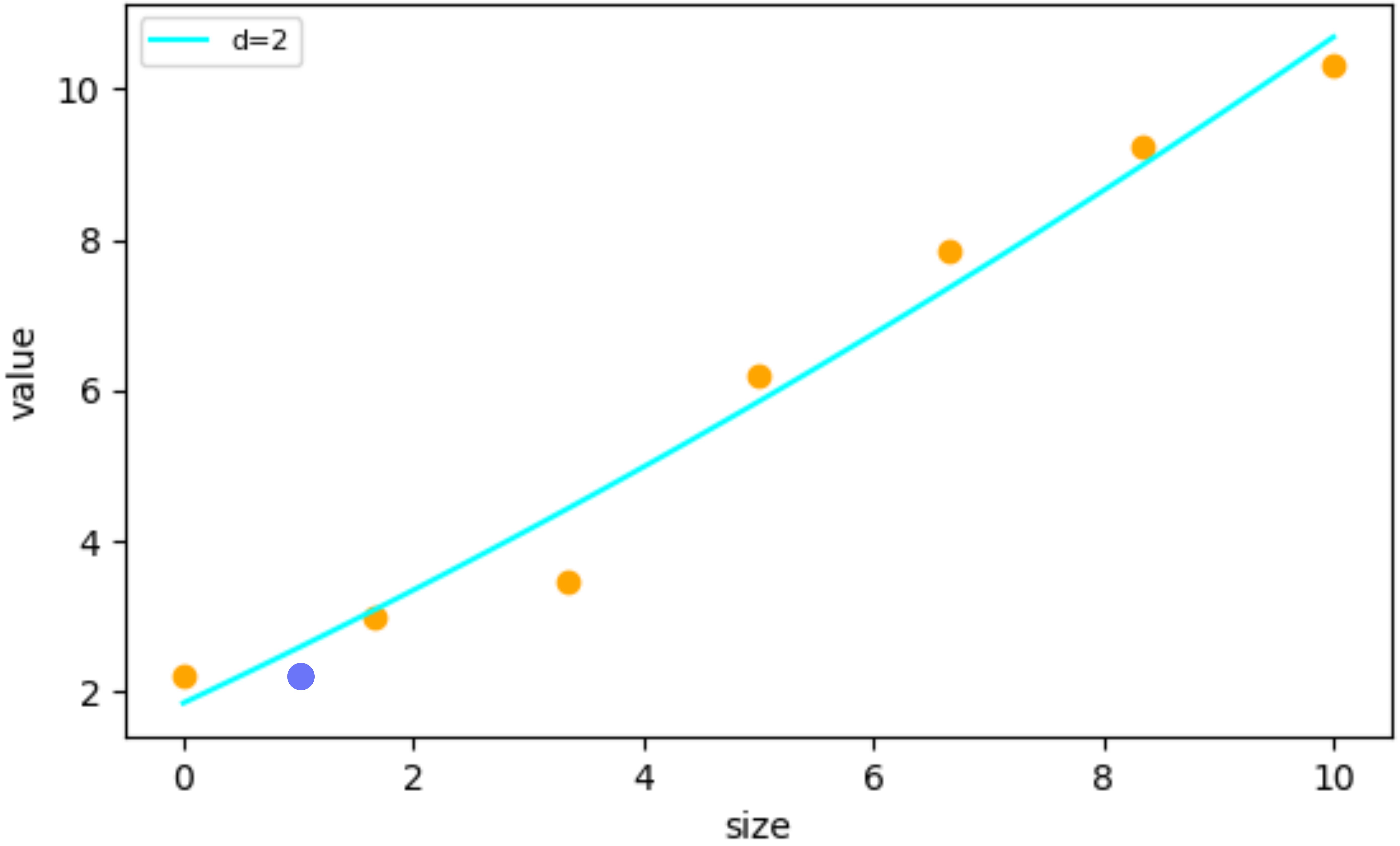
Overfitting

High variance

- Hypothesis fits the training data well.
- Fails for the ‘test’ data
- Fails to generalize
- Largely due to oscillations (curve-fitting problem!)
- Overfitting

Polynomial Regression

d = 2



- $h_w(x) = w_0 + w_1 x + w_2 x^2$

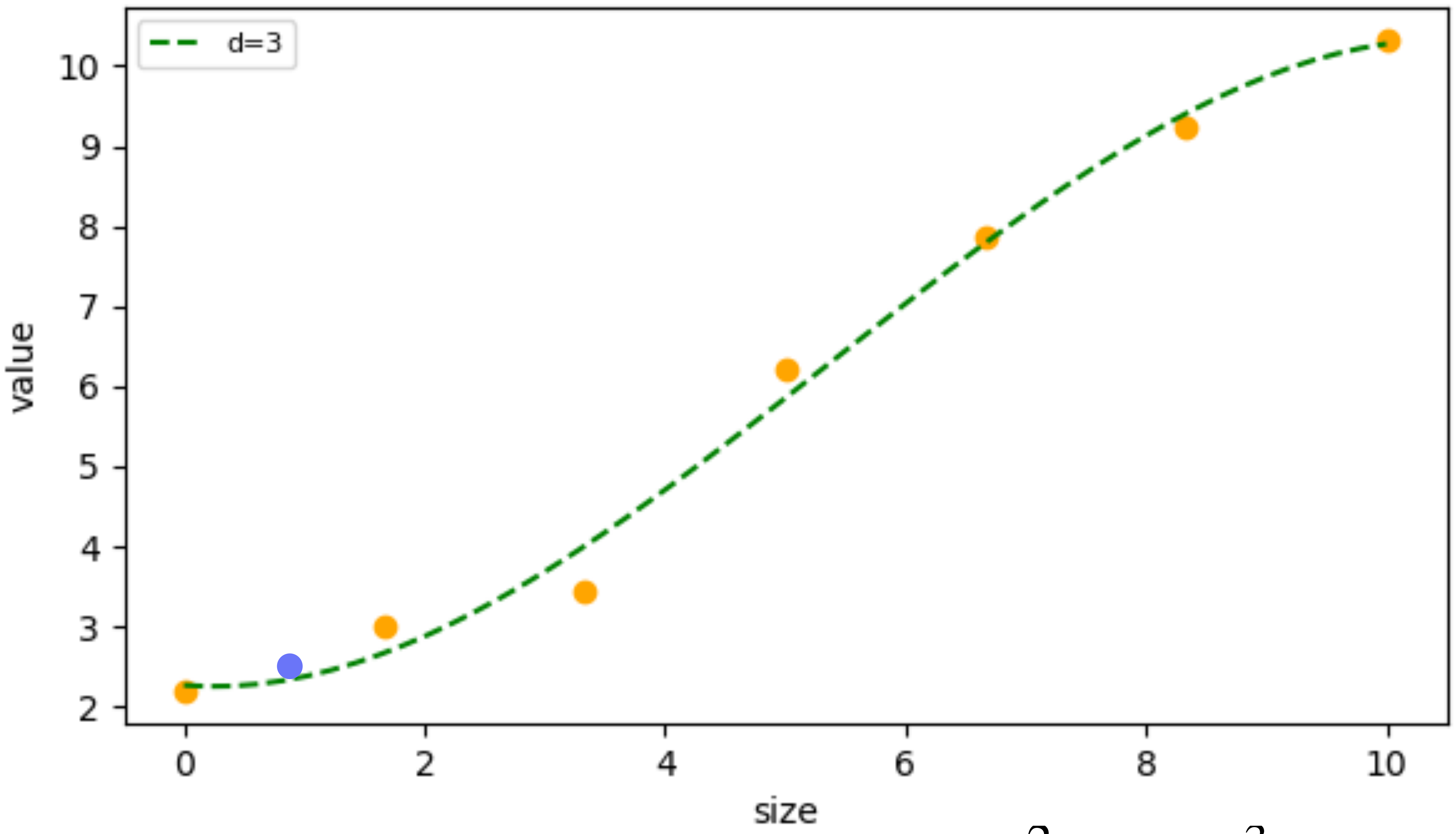
Underfitting

High bias

- Hypothesis does not fit the training data that well.
- Fails for the ‘test’ data
- Fails to generalize
- Much flatter overall!
- Under-fit

Polynomial Regression - better, overall

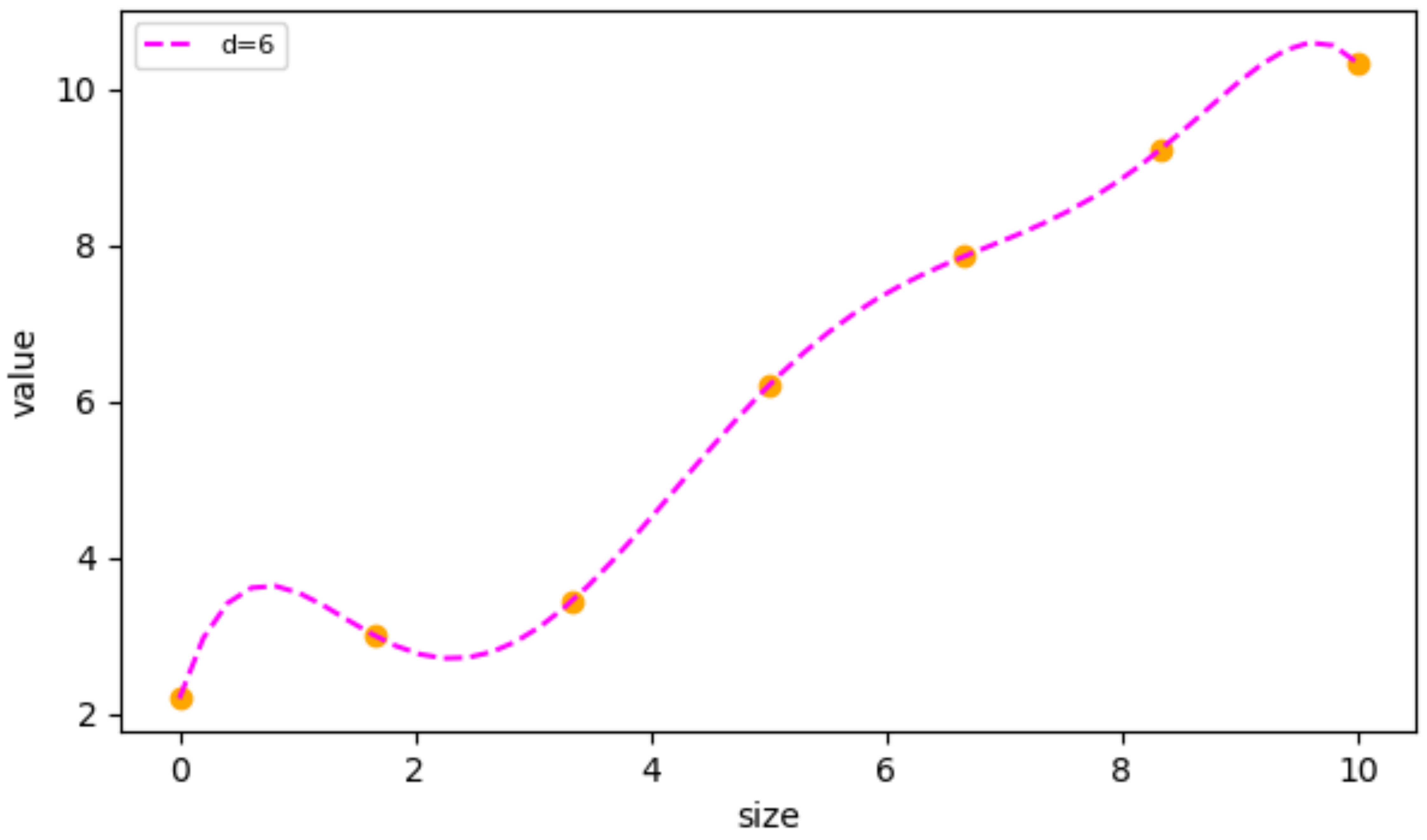
d = 3



- $h_w(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

Addressing overfitting

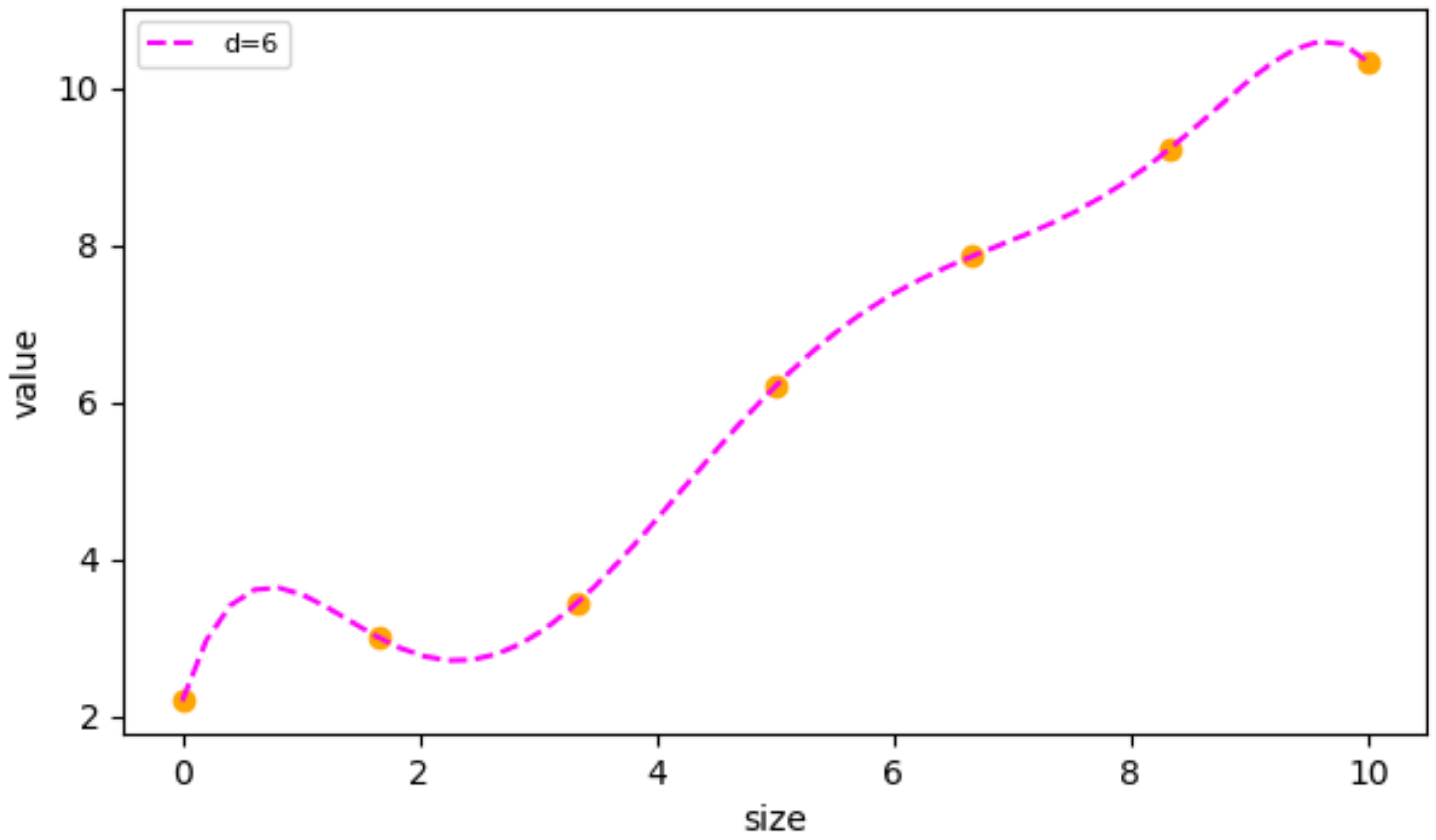
- Plotting is difficult (more features)
- Manually reduce number of features
- Regularization
- Model selection



Addressing overfitting

Regularization

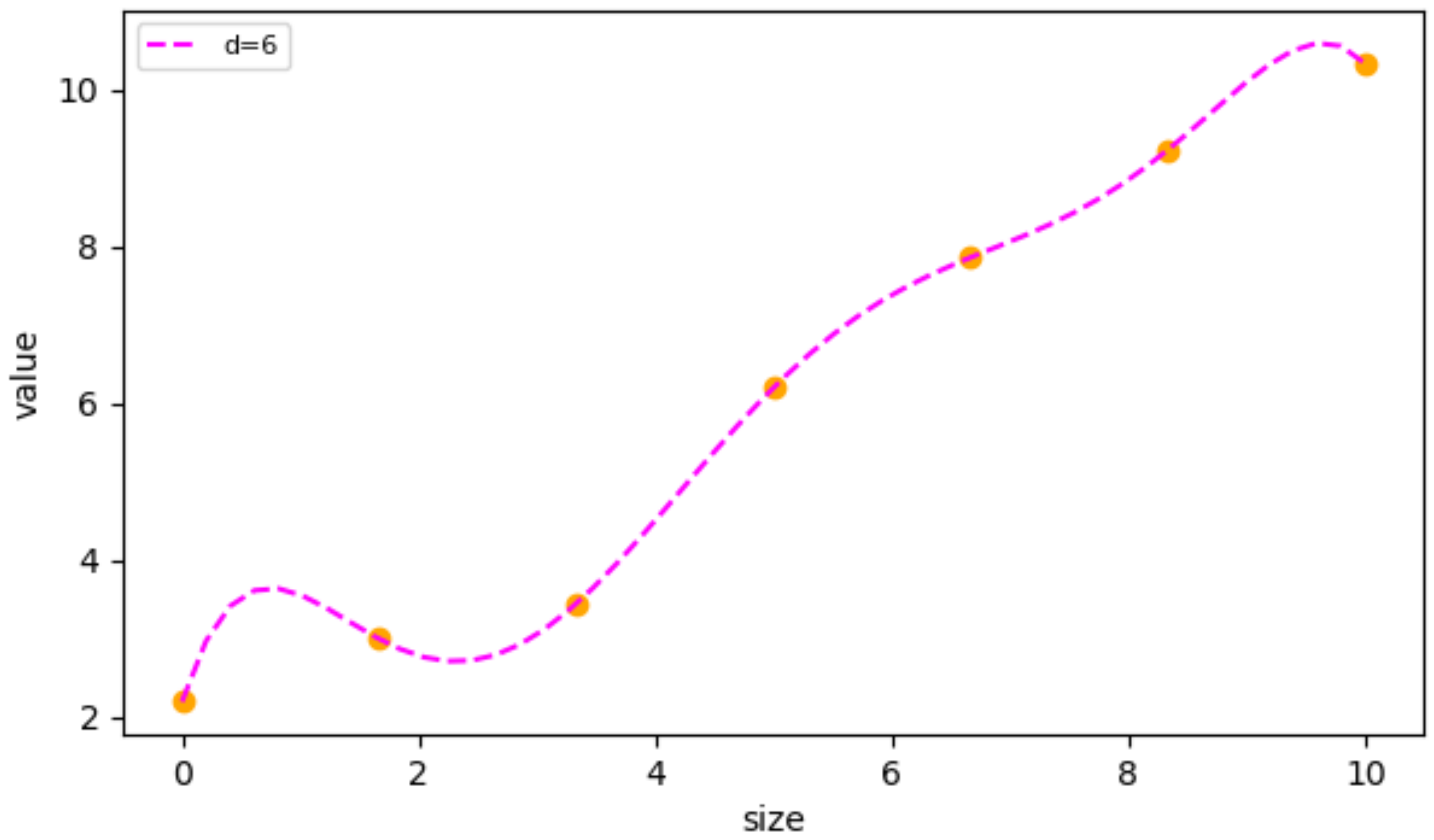
- Keep all the features
- Typically, that means to reduce oscillations
- Smoothen the curve by appropriately reducing the weights.



Polynomial Regression

d = 6 $h_w(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$

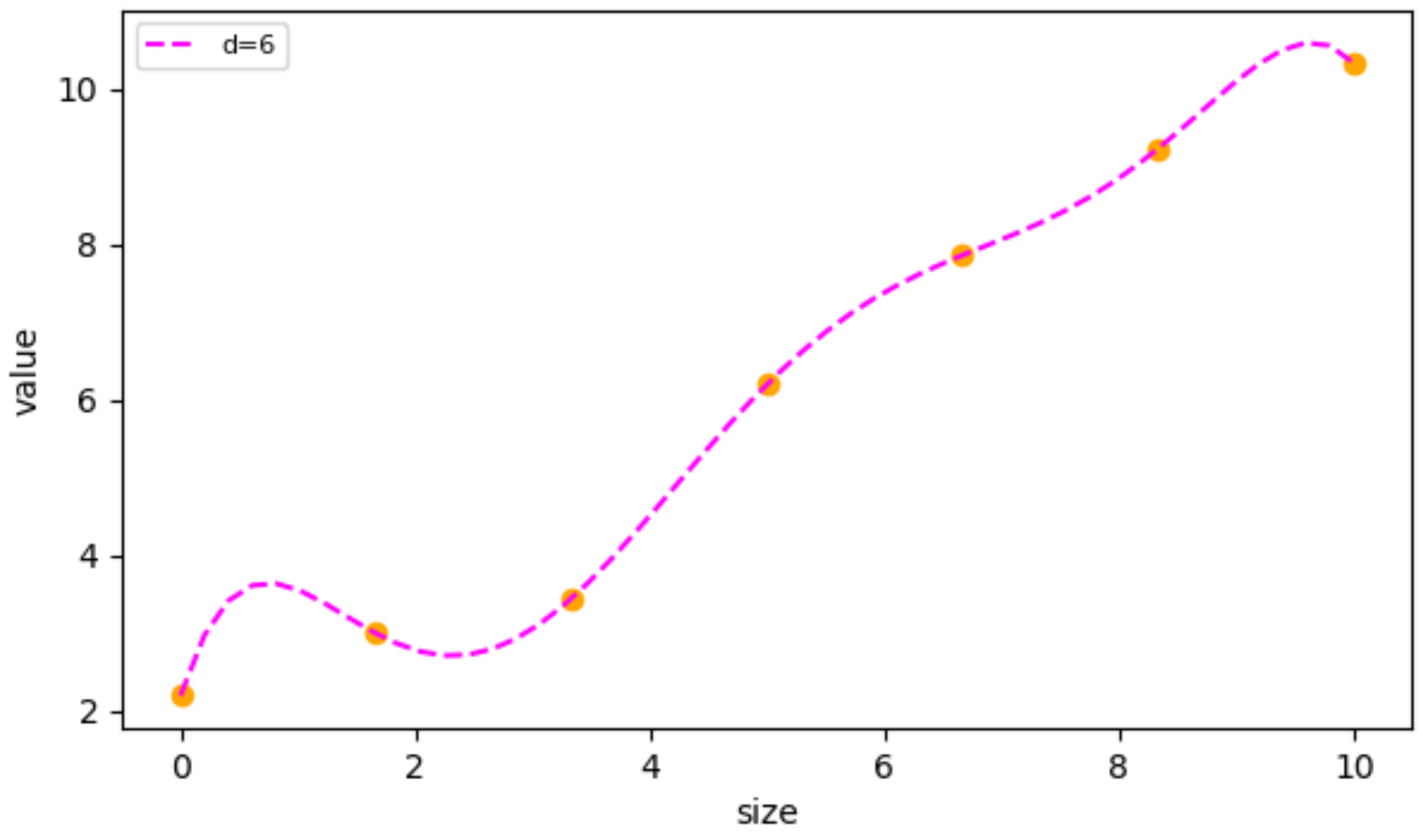
- Penalizing the weights
- smaller values for the weights
- Essentially behaves more like a lower order curve!
- Adding a 'regularization' term in the cost function.
- Needs to implement to get a better picture!



Polynomial Regression

d = 6 $h_w(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6$

- If number of features are large, which weights to be penalised?
- Add a regularisation term to the existing cost function.



Linear Regression

Cost function without regularisation

- $J(w) = \dfrac{1}{2m}\left[\sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2\right]$

Linear Regression

Cost function with regularisation

- $$J(w) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{i=1}^{m} w_j^2\right]$$
- λ is the 'regularization' parameter (another hyper parameter)

Linear Regression

Cost function with regularisation

- $$J(w) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{i=1}^{m} w_j^2\right]$$
- λ is the 'regularization' parameter
- If λ is very large, you will penalise all the weights (except w_0)
  - all the weights could go close to 0
  - Under-fit case

Linear Regression

Cost function with regularisation

- $J(w) = \dfrac{1}{2m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^{m} w_j^2\right]$
- λ is the 'regularization' parameter
- If λ is very small, original cost function prevails
- ℓ_2 regularisation

Linear Regression

Optimization

- $J(w) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{i=1}^{m}w_j^2\right]$
- $\frac{\partial J}{\partial w_0} = \frac{1}{m}\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)}) \,.\, x^{(0)}$
- $\frac{\partial J}{\partial w_j} = \frac{1}{m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)}) \,.\, x_j^{(i)} + \lambda w_j\right]$

Linear Regression

Optimization - Update step

- $$J(w) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{i=1}^{m} w_j^2\right]$$
- $$w_0 = w_0 - \alpha\frac{\partial J}{\partial w_0} = w_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})\,.\,x^{(0)}$$
- $$w_j = w_j - \alpha\frac{\partial J}{\partial w_j} = w_j - \alpha\frac{1}{m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})\,.\,x_j^{(i)} + \lambda w_j\right]$$

Linear Regression

Optimization - Update step

- $w_j = w_j - \alpha \frac{1}{m} \left[\sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) \, . \, x_j^{(i)} + \lambda w_j \right]$
- $w_j = \left(1 - \alpha \frac{\lambda}{m} \right) w_j - \alpha \frac{1}{m} \left[\sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)}) \, . \, x_j^{(i)} \right]$
- shrinking the weights

HW: Workout the update step for logistic regression with regularisation

Evaluating a model / hypothesis

Without regularisation

- Split the data into train / test set (80 / 20) or (70, 30)
- m_{tr}, m_{te} - number of samples in each of them.
- Compute training $J_{tr}(w)$ (without the regularisation term).
- $J_{tr}(w) = \sum_{i=1}^{m_{tr}} \frac{1}{2m_{tr}}(h_w(x^{(i)}) - y^{(i)})^2$

Evaluating a model / hypothesis

- $h_w(x)$
    - for each degree d, compute the training error ($J_{tr}(w)$)
    - pick the model with the lowest error
- Computer test set cost function $J_{te}(w)$

Evaluating a model / hypothesis

Without regularisation

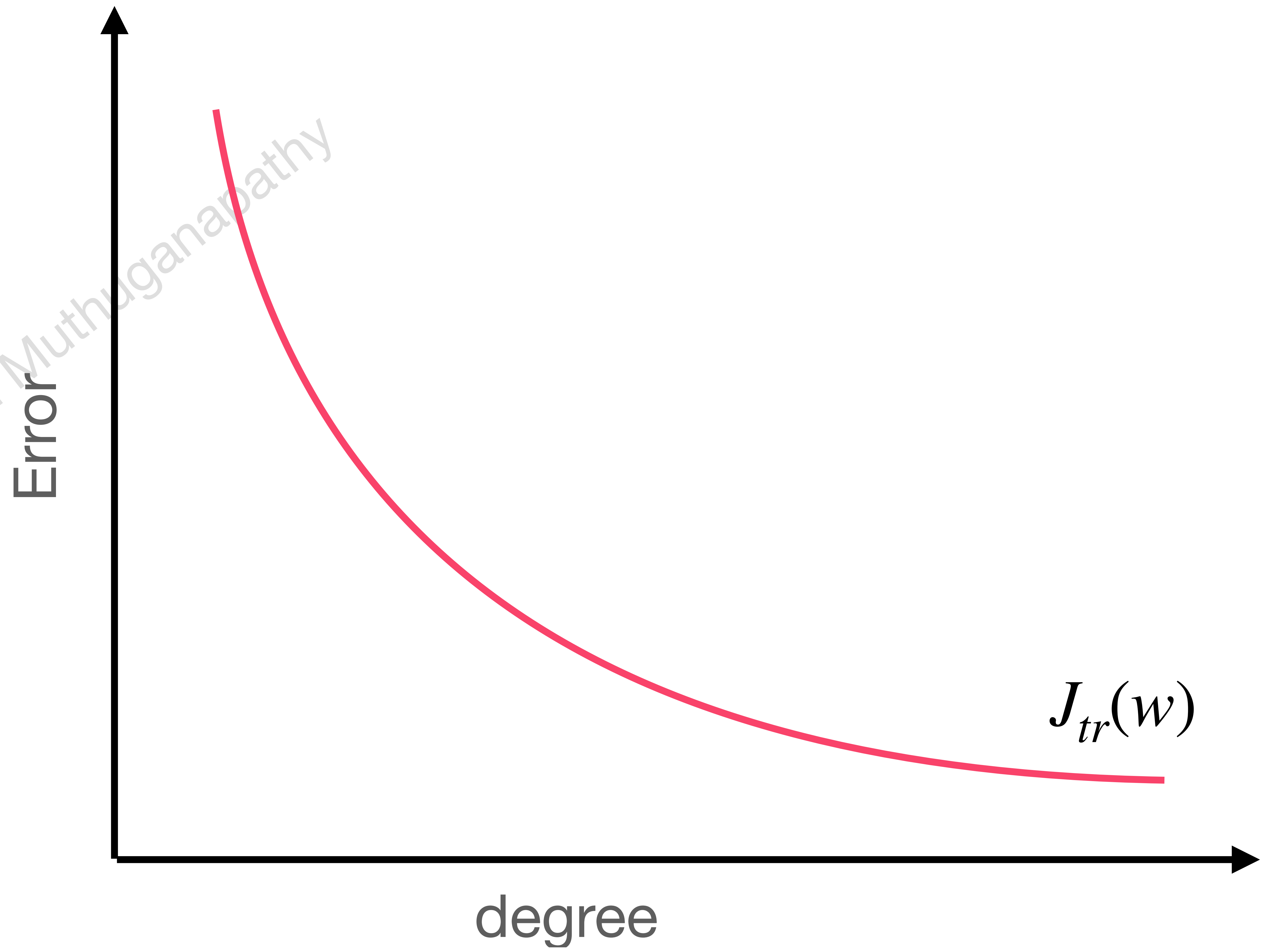
- If you have considerable, split data into train / (cross) validate / test (70 / 15 / 15 or 60 / 20 / 20)
- m_{tr}, m_v, m_{te} - number of samples in each of them.
- Compute training $J_{tr}(w)$ error (without the regularisation term).
- $$J_{tr}(w) = \sum_{i=1}^{m_{tr}} \frac{1}{2m_{tr}}(h_w(x^{(i)}) - y^{(i)})^2$$

Evaluating a model / hypothesis

- $h_w(x)$
  - for each degree d, compute the (cross) validation error ($J_v(w)$)
  - pick the model with the lowest validation error
- Using the test set cost function, compute $J_{te}(w)$
- Typically, the validation and test sets should be different.

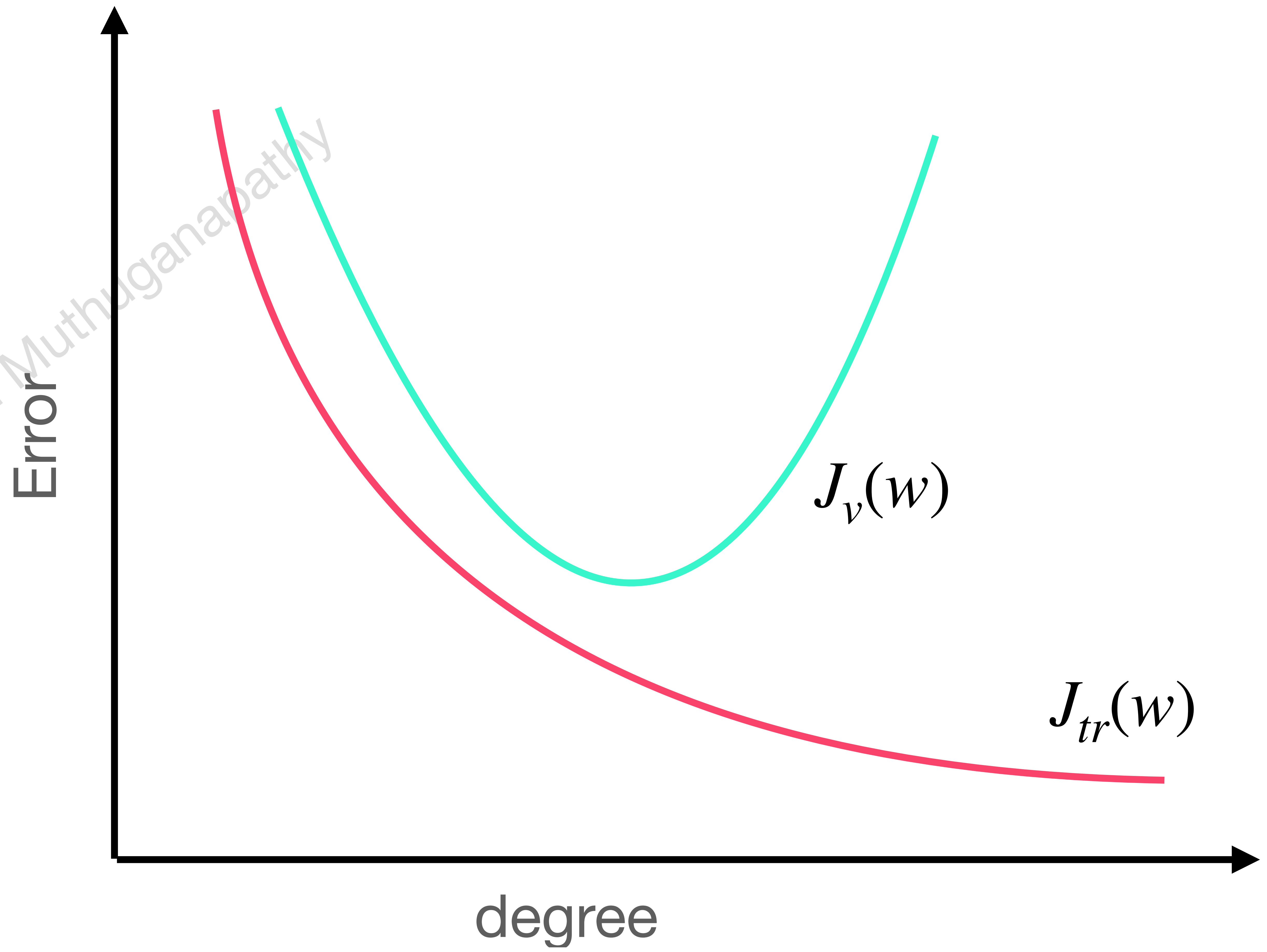
Bias or Variance

- Plot degree vs Error for $J_{tr}(w)$



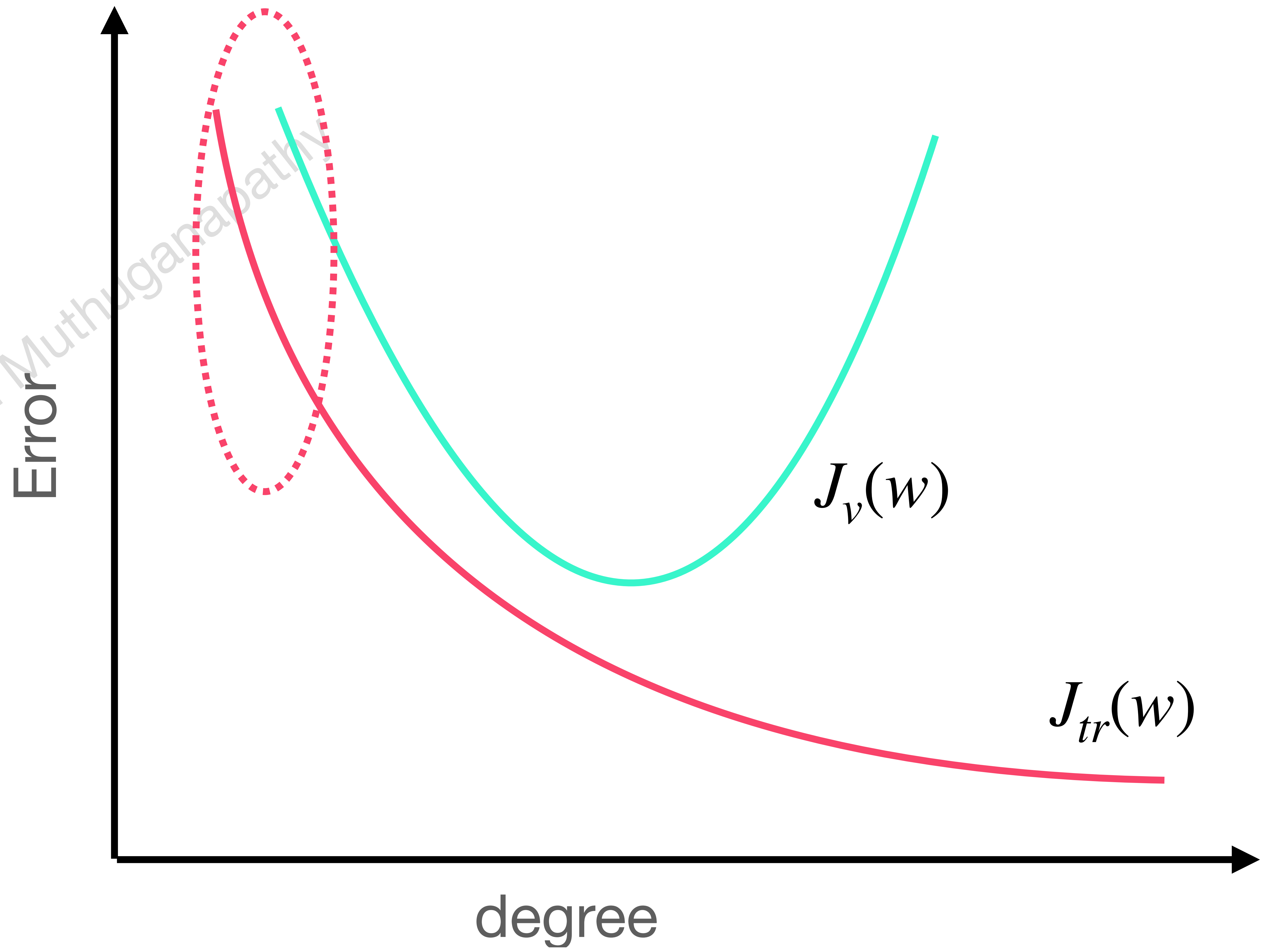
Bias or Variance

- Plot degree vs Error for $J_v(w)$



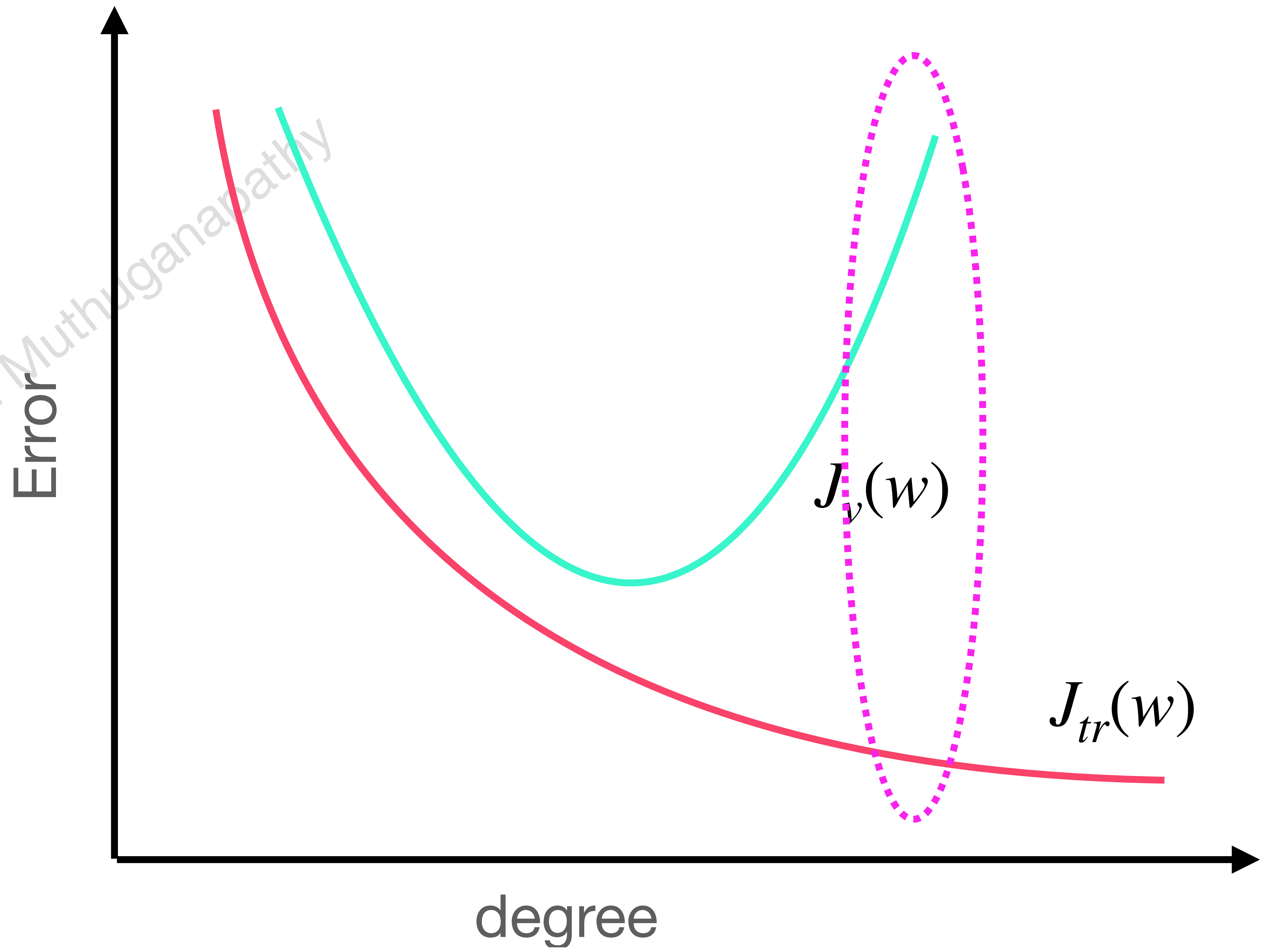
Is it bias or variance?

- Bias (Underfit)
  - $J_{tr}(w)$ will be high
  - $J_{tr}(w) \approx J_v(w)$



Is it bias or variance?

- Variance (Overfit)
    - $J_{tr}(w)$ will be low
    - $J_v(w) \ggg J_{tr}(w)$



Choosing λ

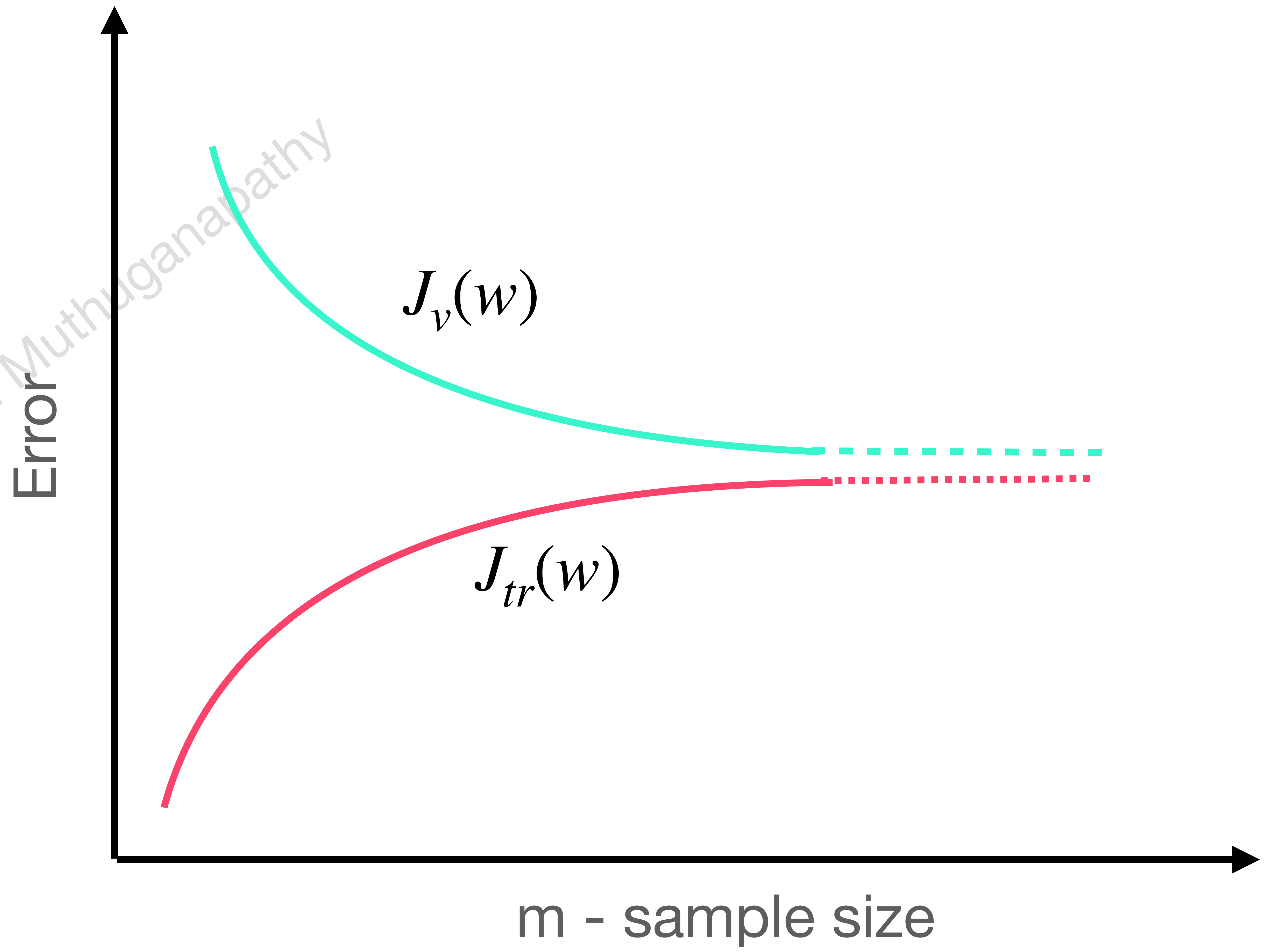
- $J(w) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{i=1}^{m} w_j^2\right]$
- $h_w(x) = w_0 + w_1x + w_2x^2 + w_3x^3$

Choosing λ

- Compute training $J_{tr}(w)$, validation $J_v(w)$ and test set cost function $J_{te}(w)$ (without regularization term)
- $J(w) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^{m} w_j^2\right]$
- Use various λ (0, 0.01 to 5, incrementing twice of the previous one)
- Compute $J_v(w)$ for each of them (and compare with $J(w)$)
- Pick the λ that has the closest match between $J_v(w)$ and $J(w)$.
- Check with the test set!

Learning curves

- Bias
    - $J_{tr}(w)$ and $J_v(w)$ reach a saturation level.
    - Increasing the sample size will not have any improvement.



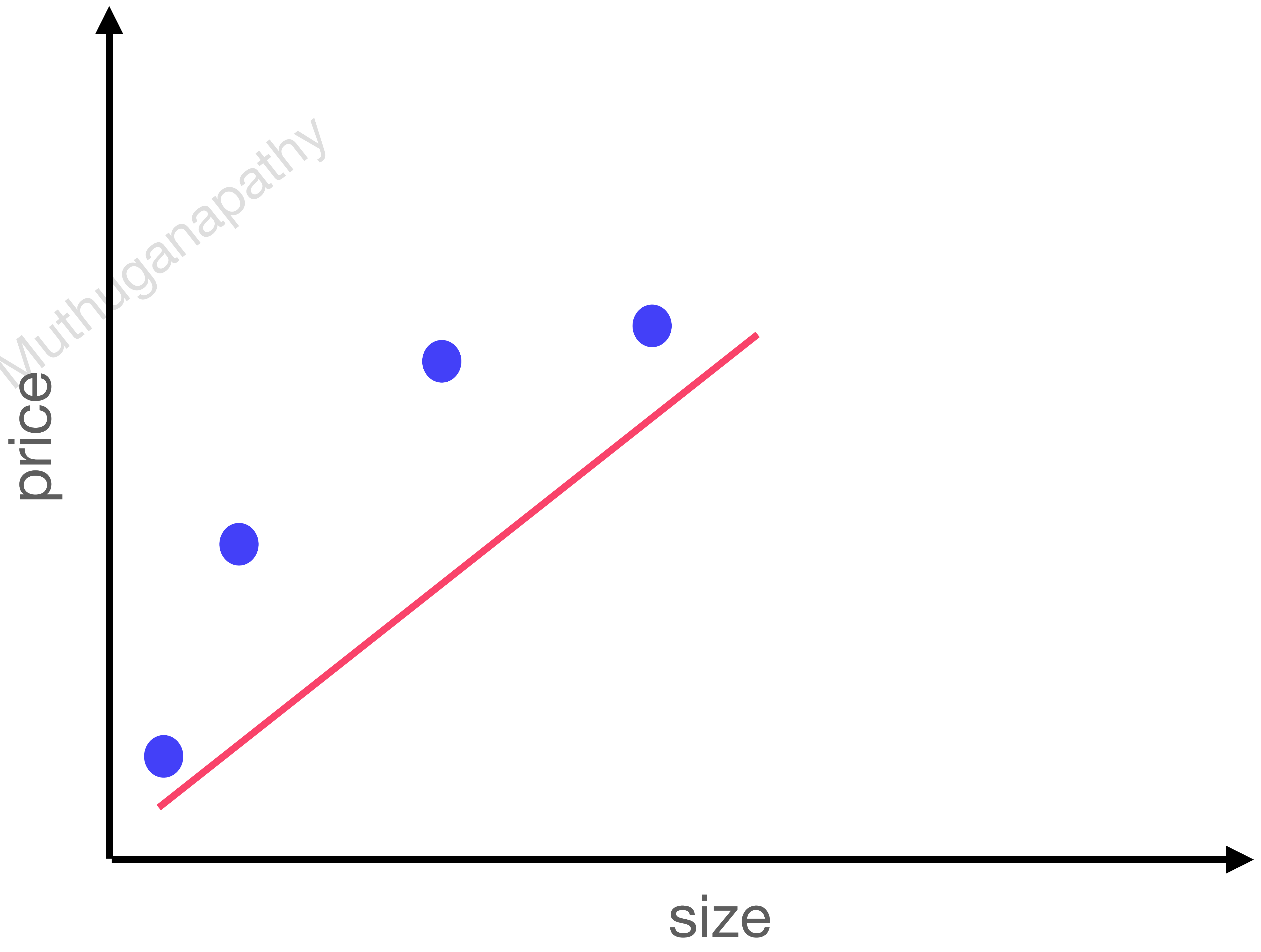
Learning curves

Why?

- Bias

    - $J_{tr}(w)$ and $J_v(w)$ reach a saturation level.

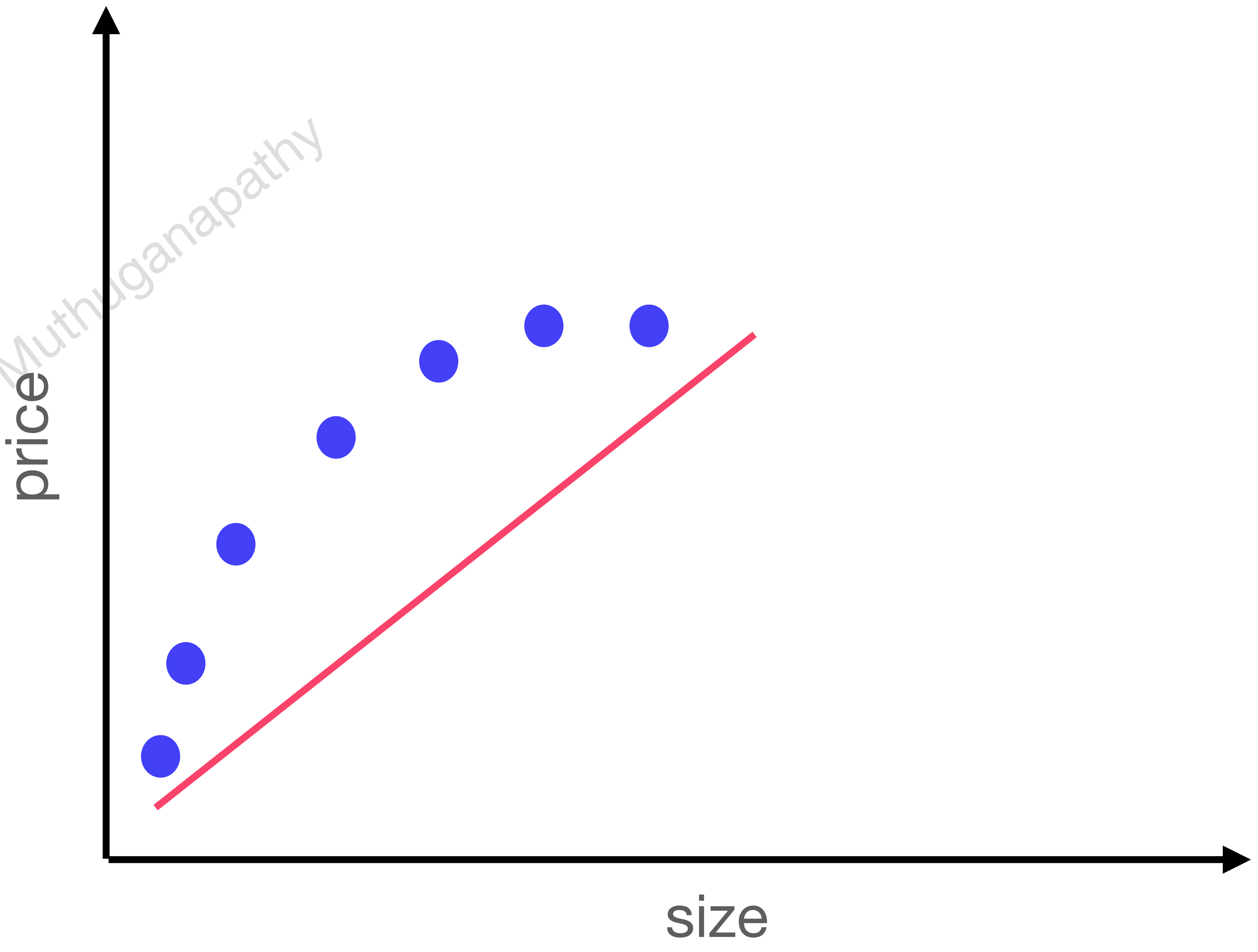
    - Increasing the sample size will not have any improvement.



Learning curves

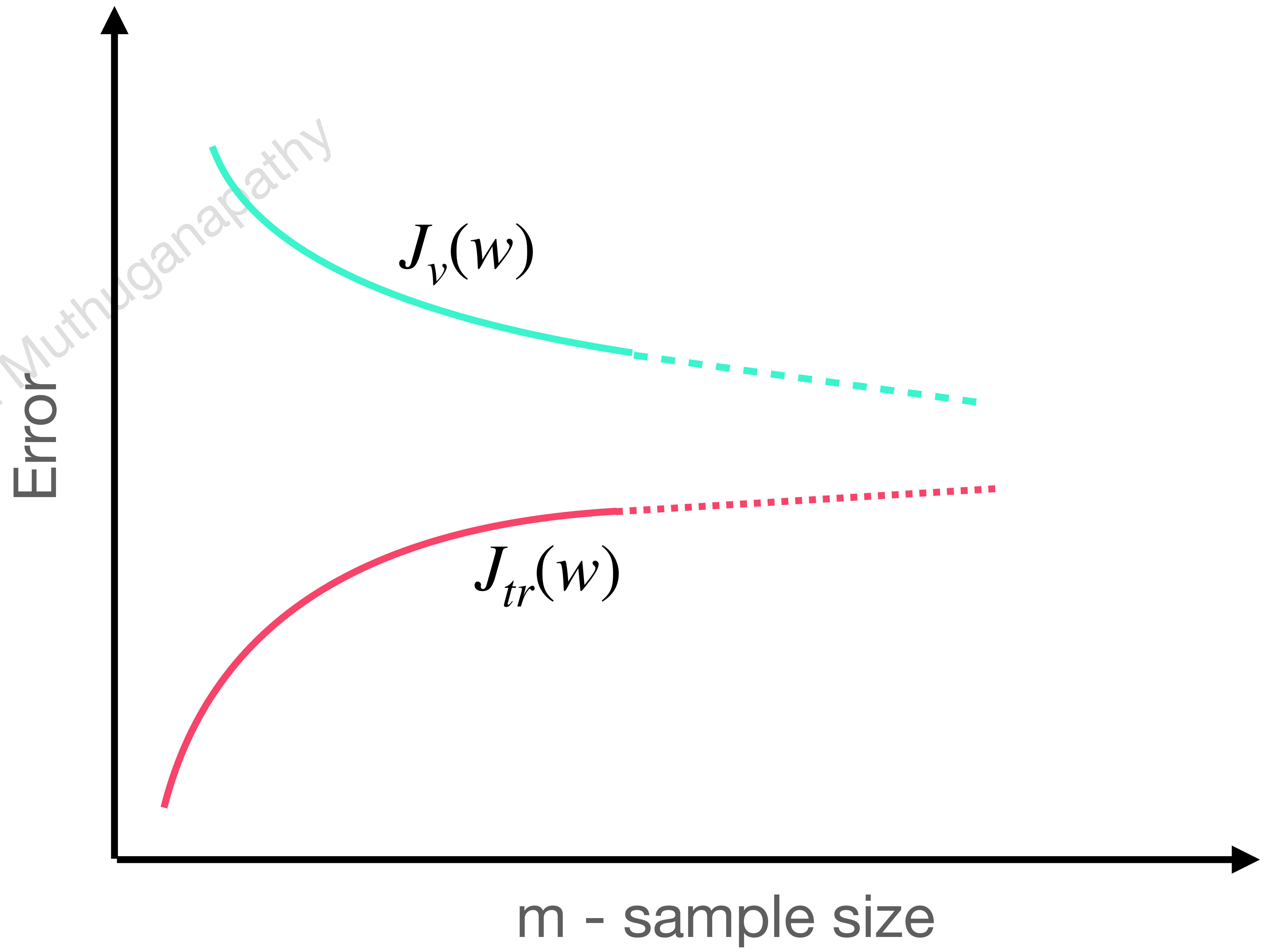
Why?

- Bias
  - $J_{tr}(w)$ and $J_v(w)$ reach a saturation level.
  - Increasing the sample size will not have any improvement.



Learning curves

- Variance
  - $J_{tr}(w)$ and $J_v(w)$ - gap is larger
  - Increasing the sample size likely to improve the performance.



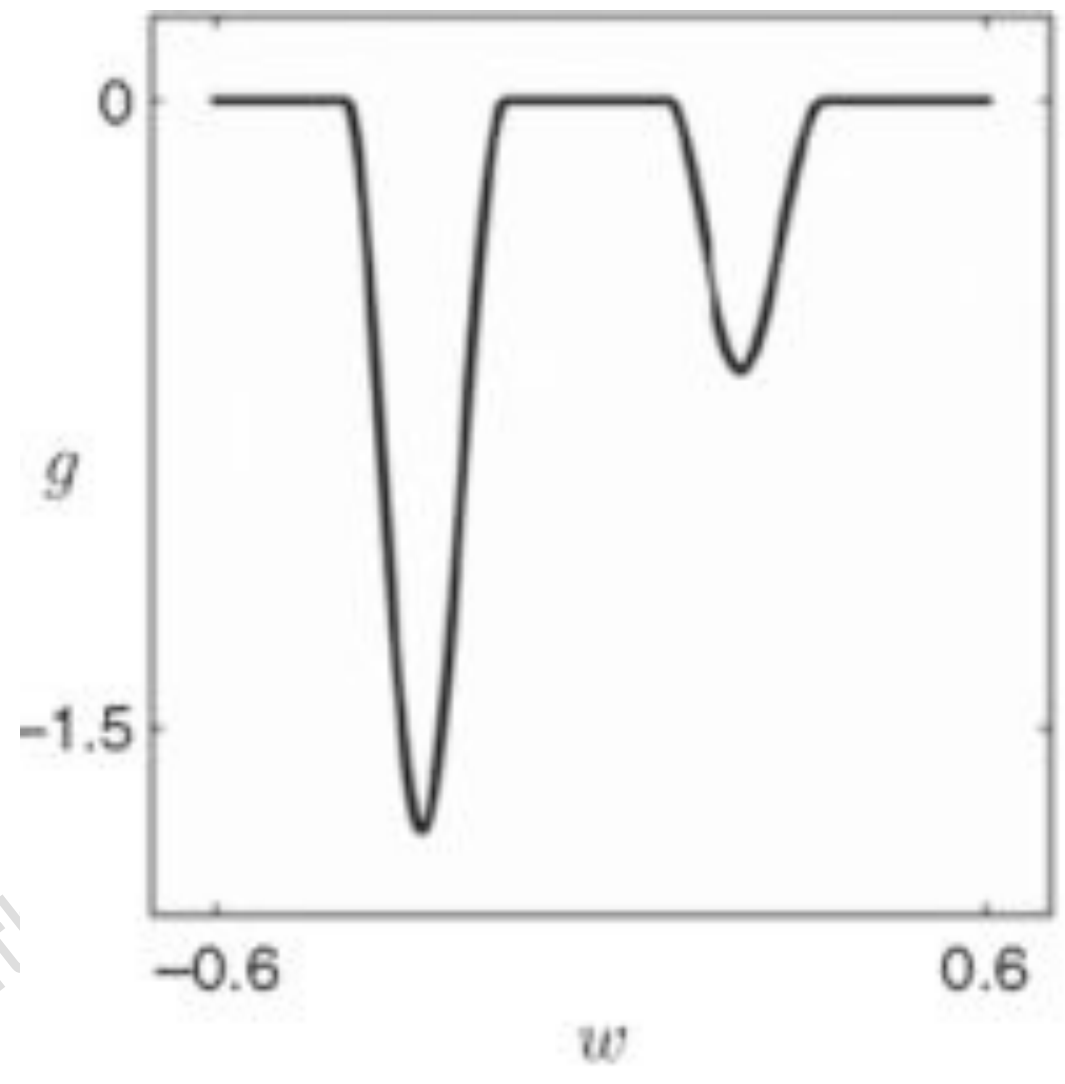
Things that can be tried out

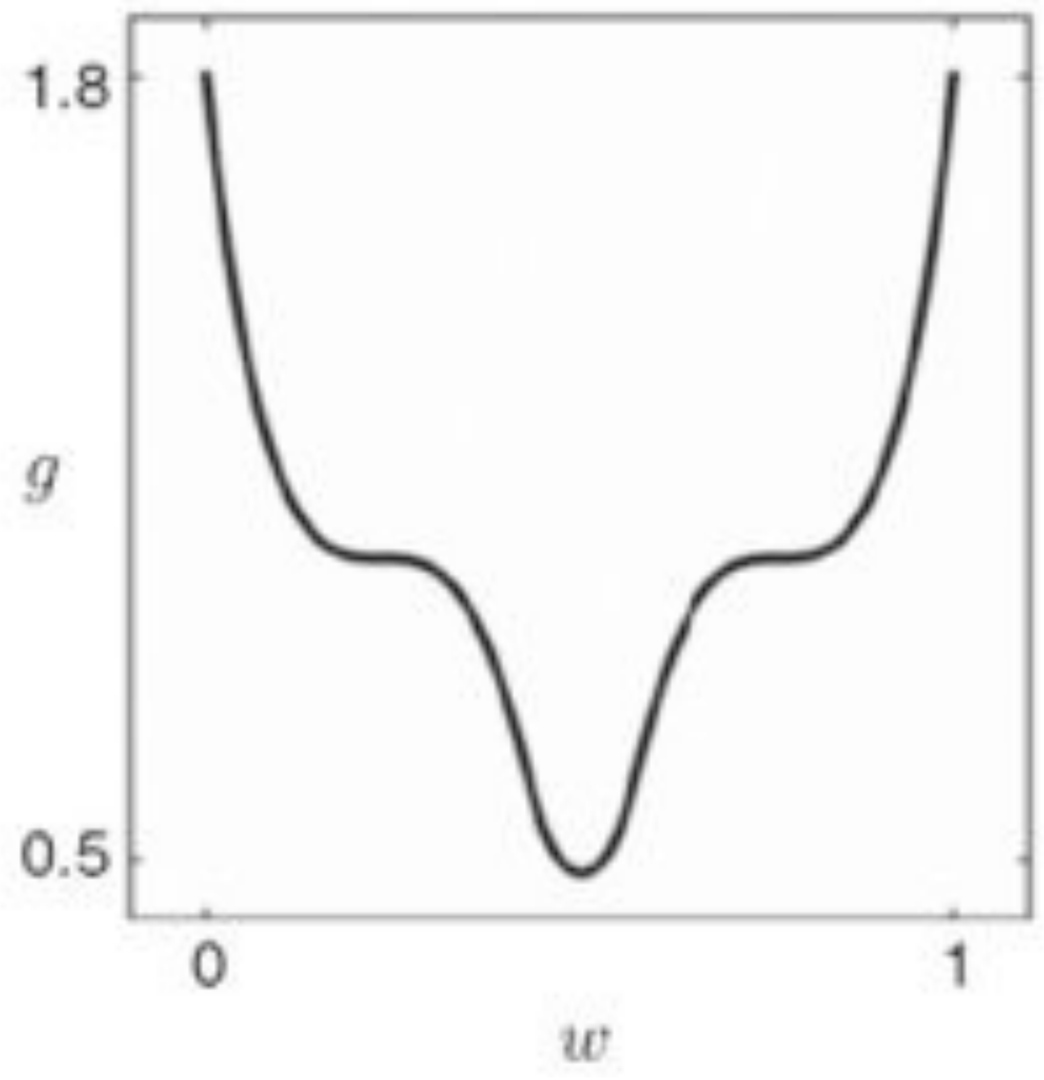
to fix bias or variance problem

- Increase the sample size - Fixes high variance
- Smaller set of features - Fixes high variance
- More / adding features - Fixing high bias problem
- Change the model - Polynomial (Fixes high bias)
- Increase the value for λ - Fixes high variance
- Decrease the value for λ - Fixes high bias

Cost function
MLR

- Non-convex cost function
- Flat regions
- inflection / saddle point





Regularized Cost function

removing flatness

- flat regions can be eliminated.
- More local optimum can come up.