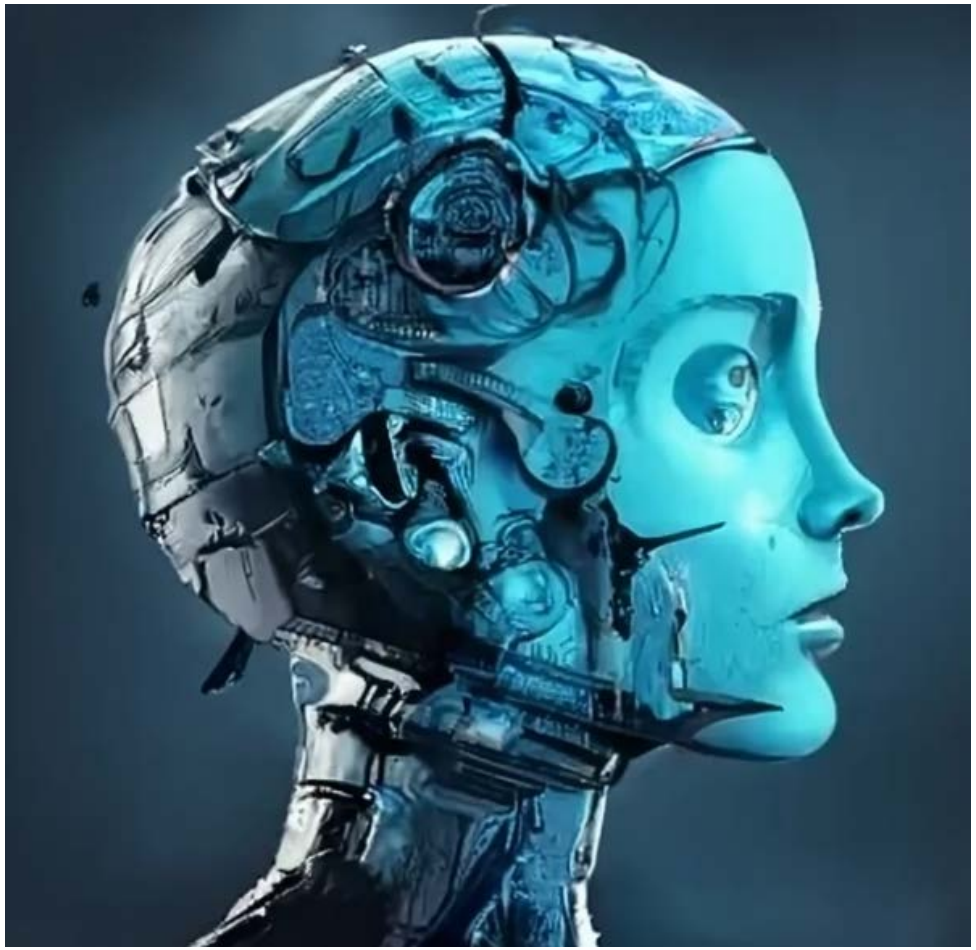
Machine Intelligence

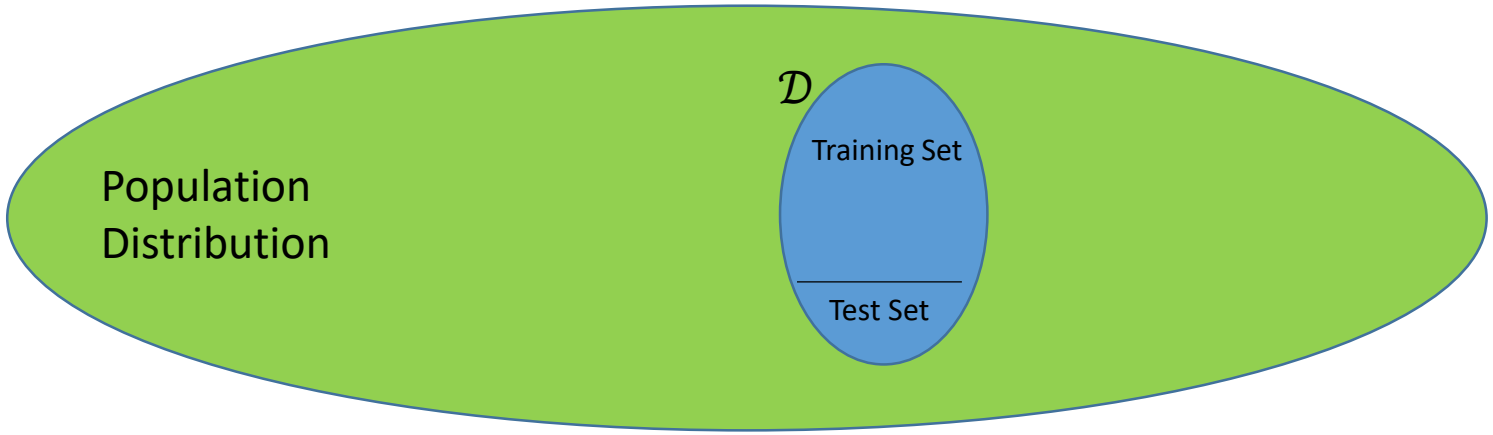
Linear Regression Regularization

Dr. Cory Merkel

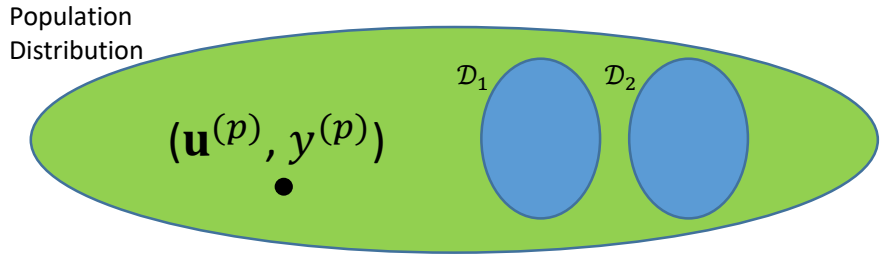


Model Generalization

- Almost always, the goal of machine learning is to get the model to *generalize*
  - Want to learn something about the full underlying distribution of data (population distribution) from a finite set of samples (training set)
  - We typically estimate how well the model generalizes by looking at its performance on a held out test set



Bias and Variance



- What can we say about the expected performance of our model on the test data?
- Assume we average the MSE over all possible $\mathcal{D}$ in the population:

$$\mathbb{E}_{\mathcal{D}}[MSE] = \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_p\left[\left(y^{(p)} - \hat{y}^{(p)}\right)^2\right]\right] = \mathbb{E}\left[\mathbb{E}_{\mathcal{D}}[(y - \hat{y})^2]\right]$$

Switch order of expectations and drop p script for simplicity.

$$\mathbb{E}_x[f(x)] \equiv \int_x p(x) f(x) \mathrm{d}x$$

$$= \mathbb{E}\left[\mathbb{E}_{\mathcal{D}}[y^2] - 2\mathbb{E}_{\mathcal{D}}[y\hat{y}] + \mathbb{E}[\hat{y}^2]\right]$$

$$= \mathbb{E}[\mathbb{E}_{\mathcal{D}}[(y - \mathbb{E}_{\mathcal{D}}[y])^2] + \mathbb{E}_{\mathcal{D}}[y]^2 - 2\mathbb{E}_{\mathcal{D}}[y\hat{y}] + \mathbb{E}_{\mathcal{D}}[(\hat{y} - \mathbb{E}_{\mathcal{D}}[\hat{y}])^2] + \mathbb{E}_{\mathcal{D}}[\hat{y}]^2]$$

$$= \mathbb{E}\left[(\mathbb{E}_{\mathcal{D}}[\hat{y}] - \mathbb{E}_{\mathcal{D}}[y])^2 + \mathbb{E}_{\mathcal{D}}[(\hat{y} - \mathbb{E}_{\mathcal{D}}[\hat{y}])^2] + \mathbb{E}_{\mathcal{D}}[(y - \mathbb{E}_{\mathcal{D}}[y])^2]\right] = bias^2 + Var + noise$$

Squared Bias **Variance** **Noise**

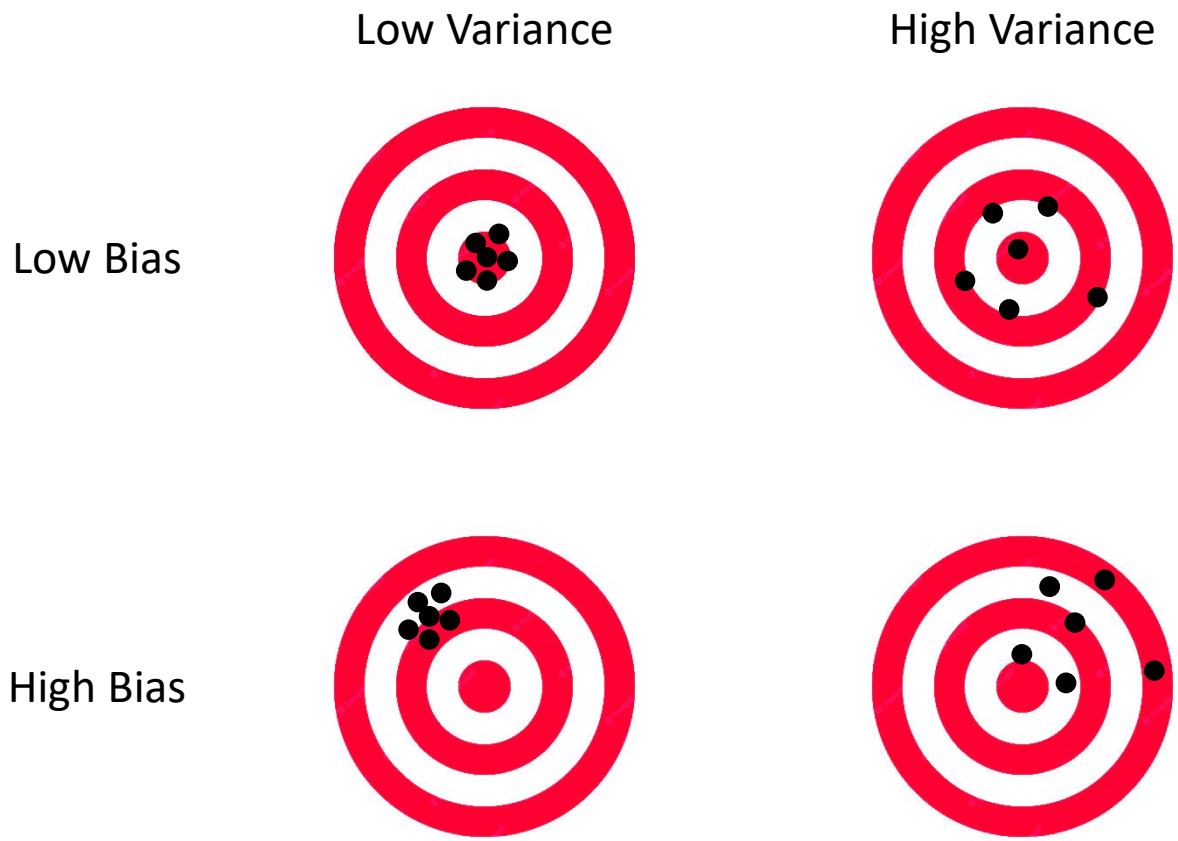
Bias-Variance Decomposition

- We can decompose the expected squared error of any machine learning model into the sum of three terms

$$\mathbb{E}_{\mathcal{D}}[MSE] = bias^2 + Var + noise$$

- Bias: High bias means the model is not appropriate for the data being modeled
  - i.e., the true function generating the observed data is different from the form of the function being used to fit the data
- Variance: High variance likely means we are overfitting to training data
- High noise: Nothing we can do about this except improve our experimental measurements
  - Therefore, the best we can hope for is $\mathbb{E}_{\mathcal{D}}[MSE] = noise$

Bias-Variance Decomposition



Bias-Variance Decomposition

- What does this look like for linear regression?
- Assume our observed data come from the following function:

$$\mathbf{y} = \mathbf{U}\tilde{\mathbf{w}} + \boldsymbol{\epsilon}$$

and, our hypothesis function is $\hat{\mathbf{y}} = \mathbf{U}\mathbf{w}$, with $\mathbf{w} = \mathbf{U}^{\dagger}\mathbf{y}$

also, $\epsilon^p \sim \mathcal{N}(0, \sigma_\epsilon^2)$

- With a bit of work, we can derive the following:

$$bias^2 = 0$$

$$Var = \frac{\sigma_\epsilon^2 n}{n - N - 1}, n - N - 1 > 0$$

$$noise = \sigma_\epsilon^2$$

$$\mathbb{E}_{\mathcal{D}}[MSE] = \sigma_\epsilon^2 \frac{n-1}{n-N-1}$$

Recall that n is the number of points in our training set and N is the number of features.

Bias-Variance Decomposition

- Some important things to notice about the bias-variance decomposition for linear regression:

$$\mathbb{E}_{\mathcal{D}}[MSE] = \sigma_{\epsilon}^2 \frac{n-1}{n-N-1}$$

  - More training data → less error
  - Higher dimensional data (i.e., # of features) → higher error

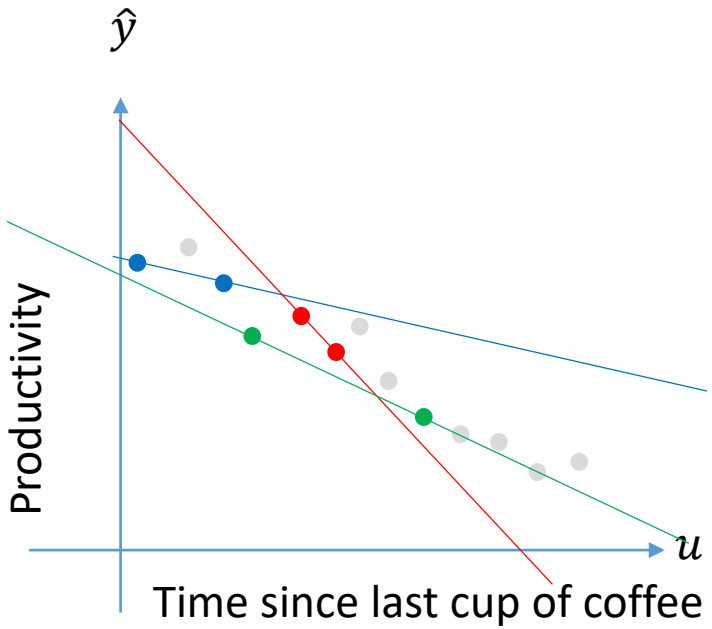
$$n = \frac{\mathbb{E}_{\mathcal{D}}[MSE](N+1) - \sigma_{\epsilon}^2}{\mathbb{E}_{\mathcal{D}}[MSE] - \sigma_{\epsilon}^2}$$

  - For a given error level, we need more training data as the dimension of the input grows (N)
    - ❑This is related to an important concept called the *curse of dimensionality*

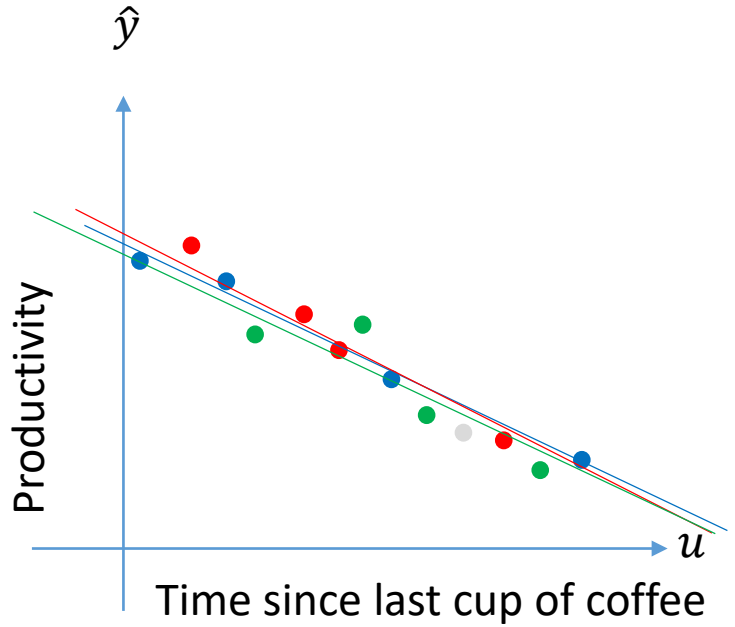
Bias-Variance Decomposition

- Let's get some intuition about the influence of training set size on expected MSE

$n = 2$: If we only use 2 points to train our model, then the hypothesis function will have high variance. (Very different hypotheses depending on which 2 datapoints we pick.)



$n = 4$: As we use more points for training, the training data have better representation of the population distribution, and variance decreases.



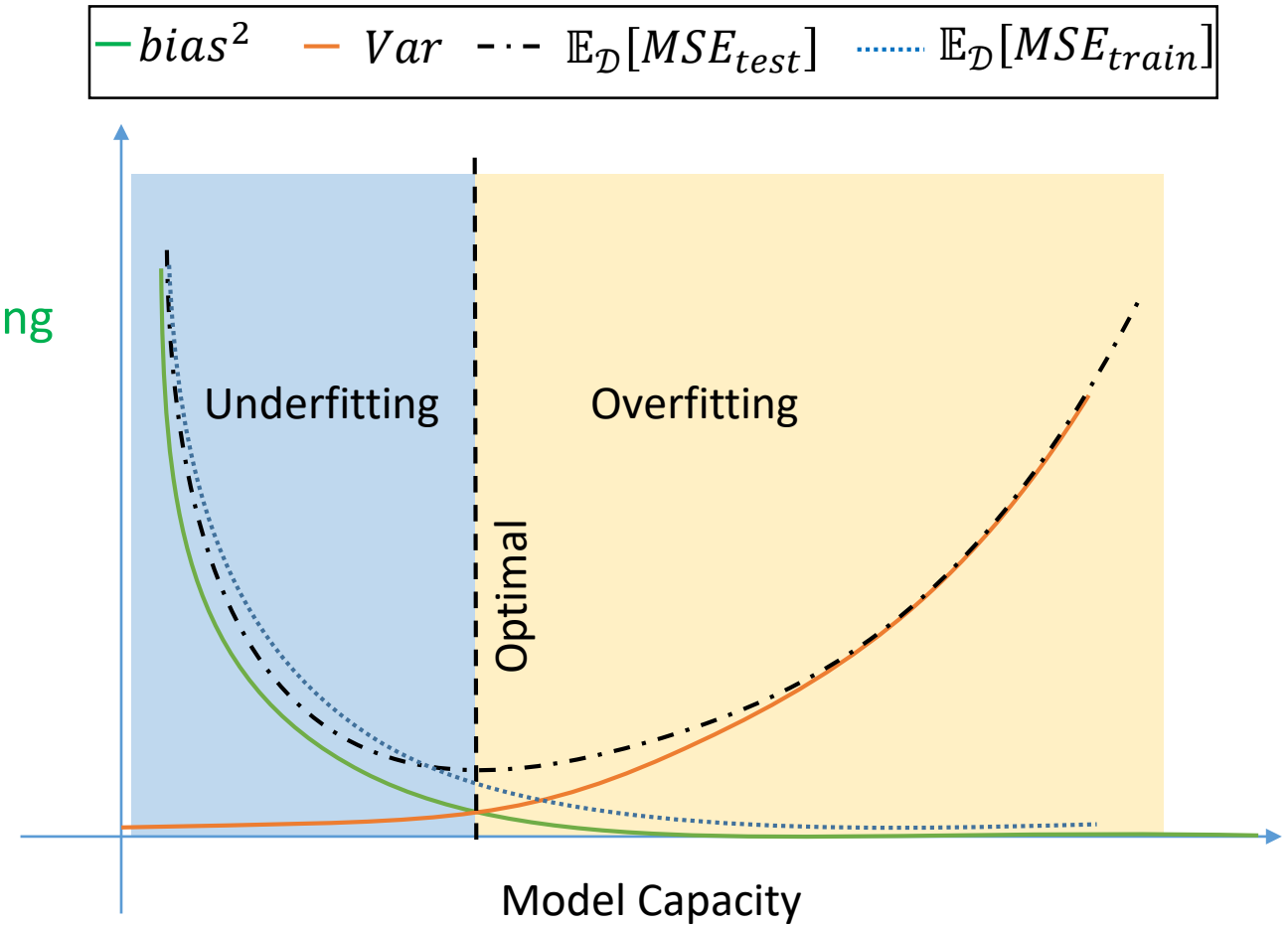
Finding the Right Model Capacity (Model Selection)

$$\mathbb{E}_{\mathcal{D}}[MSE] = bias^2 + Var + noise$$

- For linear regression, we can reduce the variance of our model by
  - Using more data
  - Using fewer features
- More generally, we may be working with data that are not linear, and we will have a bias term
  - Highly flexible (high capacity) models will have high variance, but low bias
  - Less flexible (low capacity) will have high bias, but low variance
  - We want to use a model with the “just right” level of flexibility

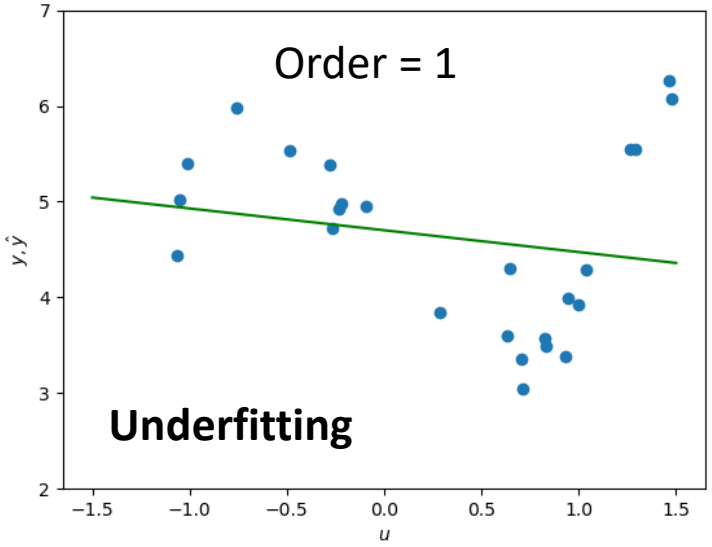
Model Capacity and Bias-Variance

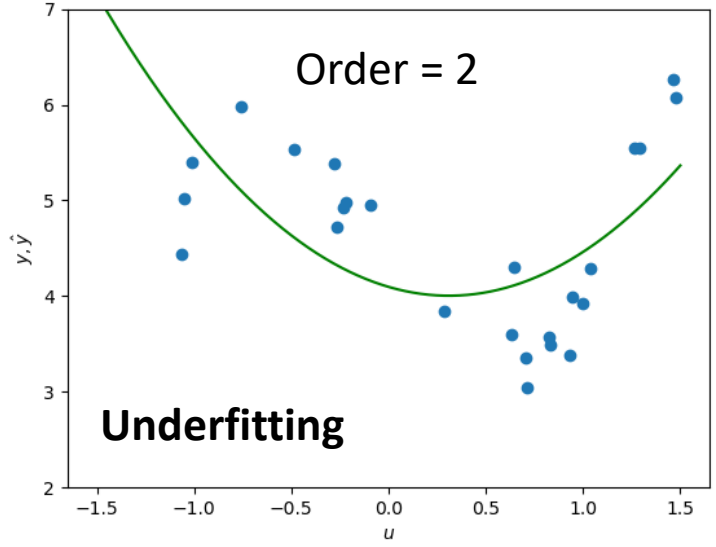
- Generally, bias decreases and variance increases as model capacity increases
  - Then, why does deep learning work so well, where model capacity is huge?
    - ❑Hint: Something really interesting happens as we keep moving to the right in this plot…

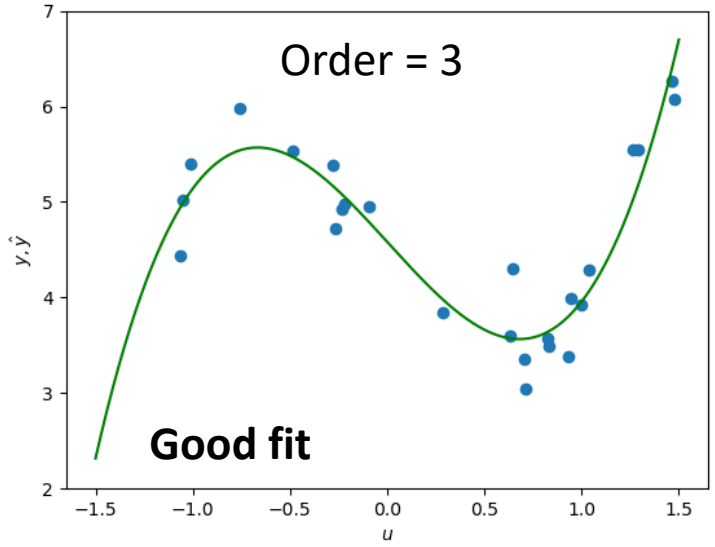


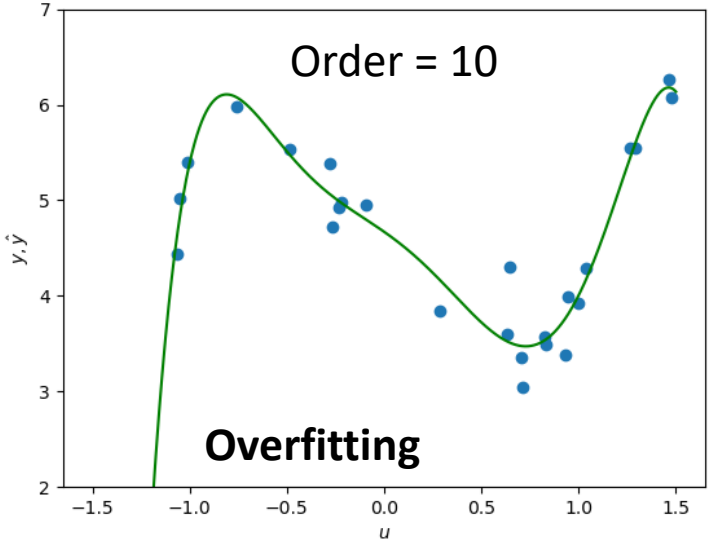
Example: $y = \tilde{w}_0 + \tilde{w}_1 u + \tilde{w}_2 u^3$

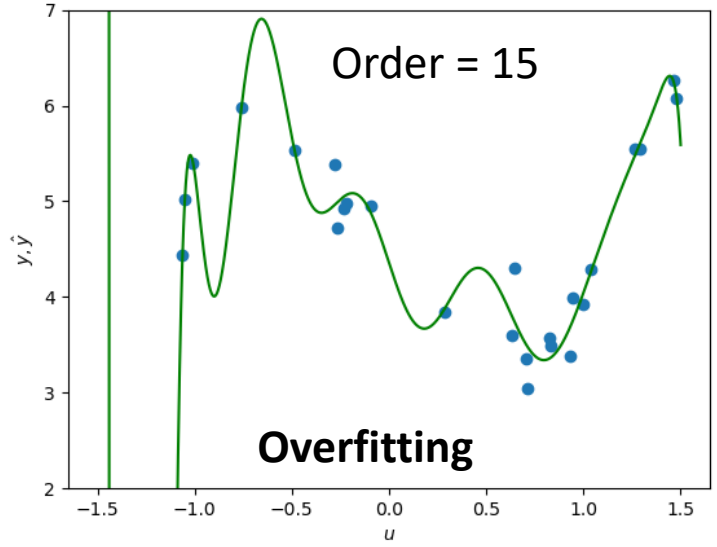
- Linear regression using different polynomial features of u:





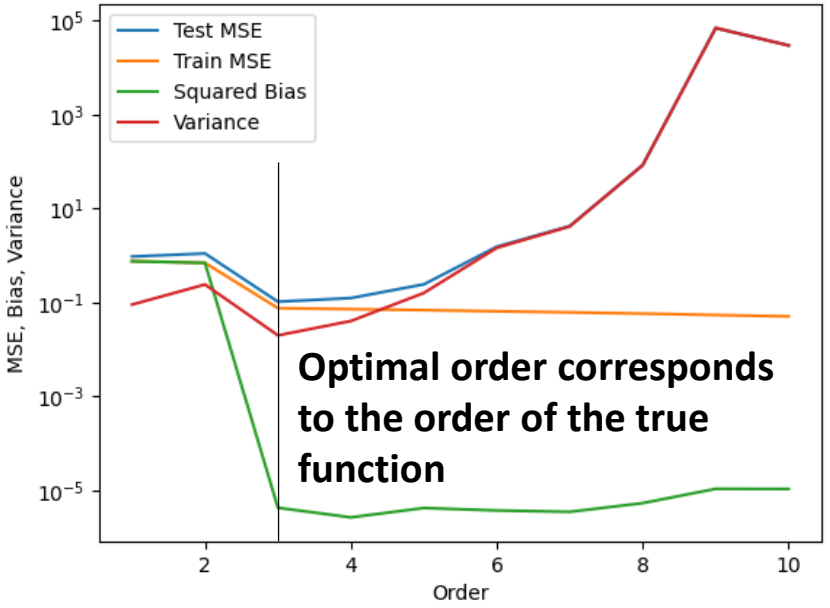






Model Capacity and Bias-Variance

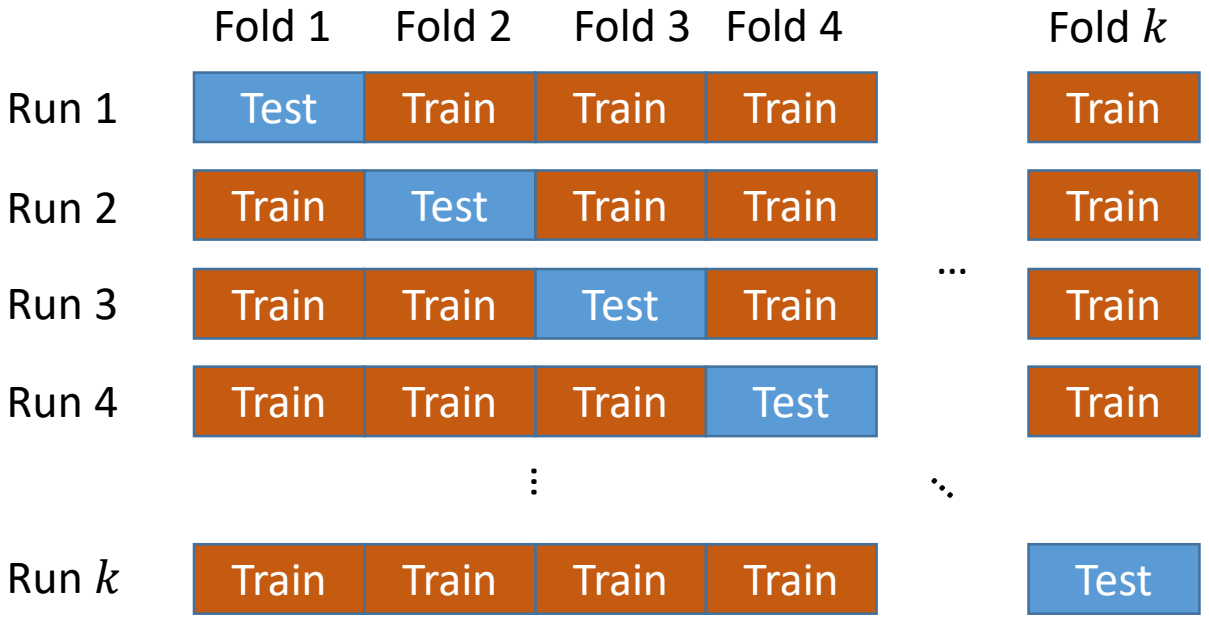
- The real bias and variance curves won't look exactly like the ideal ones
  - Remember, to take the expected values in the definitions of bias and variance, we would have to perform an infinite number of simulations



- In this example, we knew the true underlying function ($y = \widetilde{w}_0 + \widetilde{w}_1 u + \widetilde{w}_2 u^3$), so we could generate as much data as we want
  - In practice, we won't know the true function, so we have to estimate bias, variance, MSE, and other metrics using the data we are given in $\mathcal{D}$

How Do We Get a Good Estimate of Expected Performance with Limited Data?

- k-Fold cross validation:
  - Split $\mathcal{D}$ into k non-overlapping sets
    - ❑Use the union of $k-1$ sets for training and 1 set for testing
    - ❑Average results over k different combinations of train/test
      - When $k = n$, this is called leave-one-out cross validation (LOOCV)



Bias, Variance, and MSE with k-Fold Cross Validation

$$bias^2 \approx \frac{1}{n}\sum_{p=1}^{n}\left[\left(\frac{1}{k}\sum_{i=1}^{k}\hat{y}^{(p,k)}\right) - y^{(p)}\right]^2$$

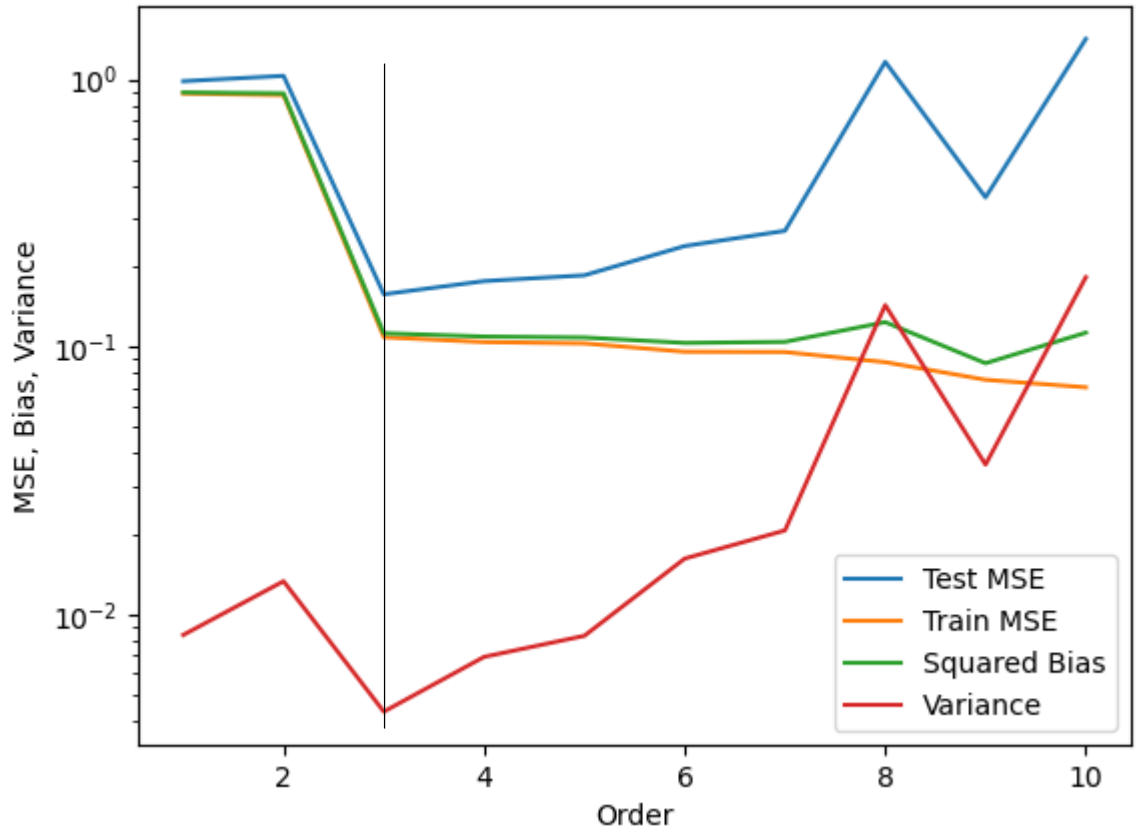
$$Var \approx \frac{1}{n}\sum_{p=1}^{n}\frac{1}{k}\sum_{i=1}^{k}\left[\hat{y}^{(p,k)} - \left(\frac{1}{k}\sum_{i=1}^{k}\hat{y}^{(p,k)}\right)\right]^2$$

$$\mathbb{E}[MSE] = \frac{1}{k}\sum_{i=1}^{k}\sum_{p\in Fold\ i}^{N/k}\frac{1}{N/k}\left(\hat{y}^{(p,k)} - y^{(p)}\right)^2$$

Average test MSE over all Folds.

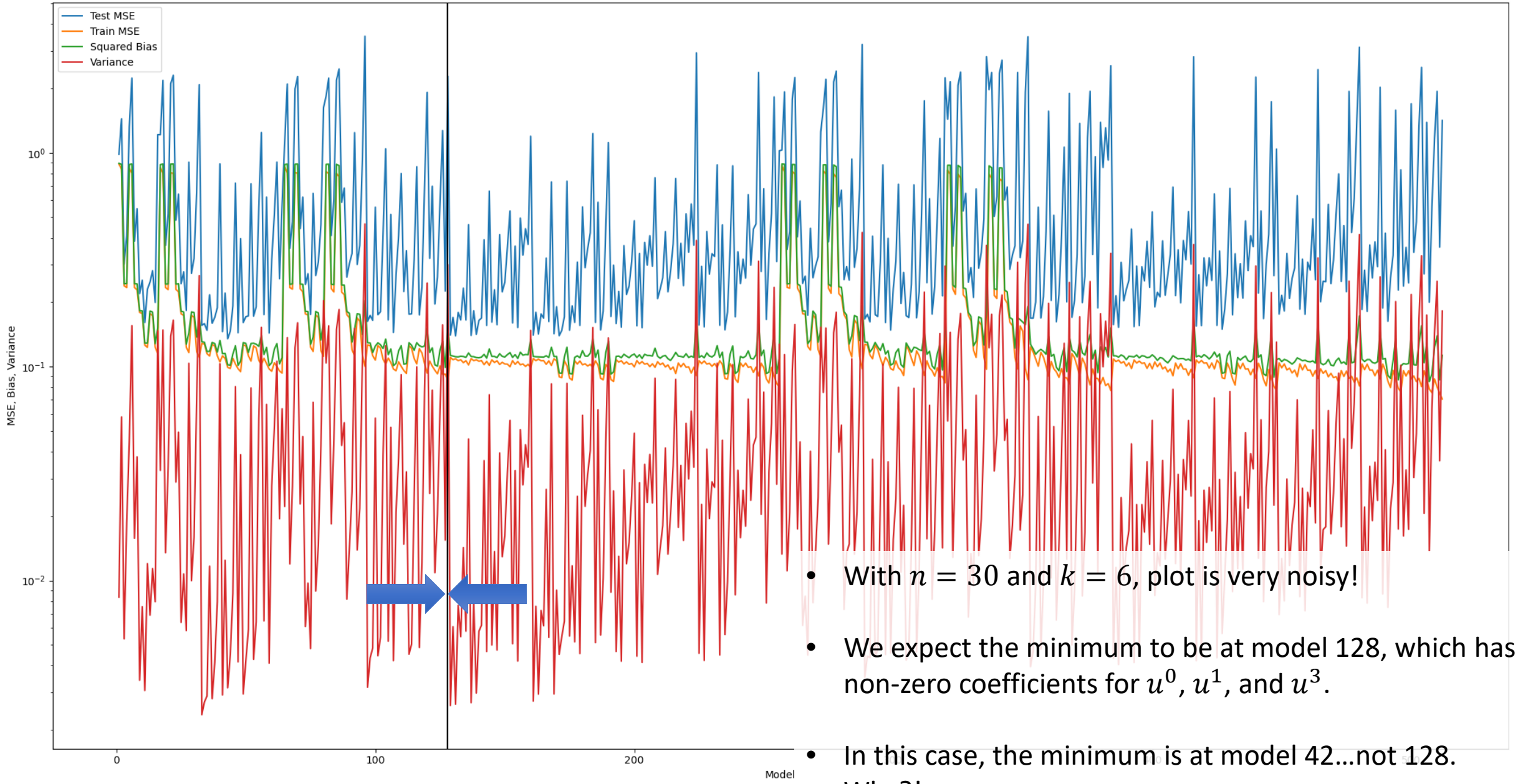
Bias, Variance, and MSE with k-Fold Cross Validation

- $n = 30, k = 6$
  - Even with a fairly small amount of data, we can get a good idea of the best model using cross validation

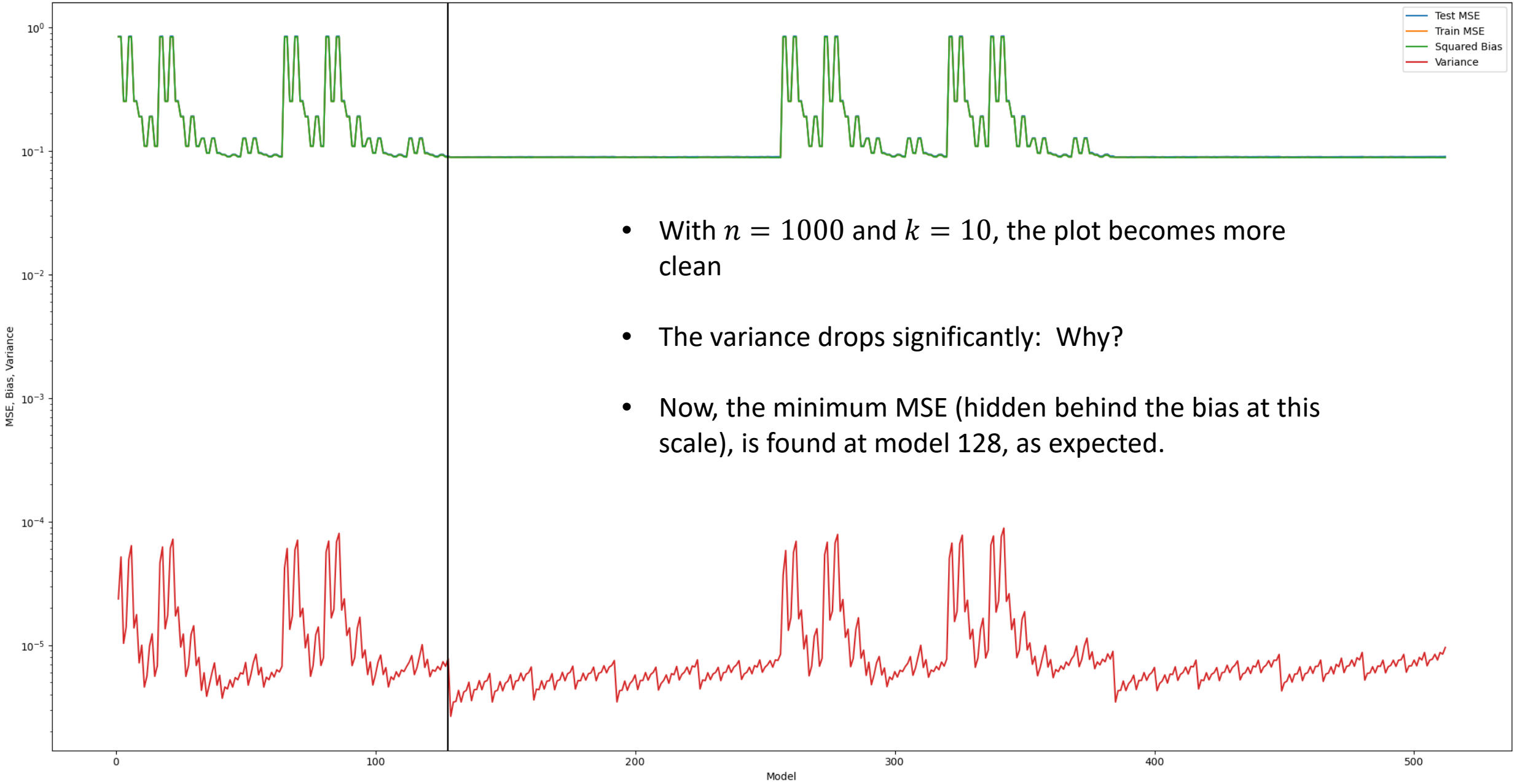


Bias, Variance, and MSE with k-Fold Cross Validation

- We could also try out all possible combinations of non-zero coefficients
  - For maximum order of 10, we have bias + first order + 2^9 combinations of others:
    - e.g.: $\hat{y} = w_0 + w_1 u + w_4 u^4 + w_7 u^7$



- With $n = 30$ and $k = 6$, plot is very noisy!
- We expect the minimum to be at model 128, which has non-zero coefficients for u^0, u^1, and u^3.
- In this case, the minimum is at model 42…not 128. Why?!



- With $n = 1000$ and $k = 10$, the plot becomes more clean
- The variance drops significantly: Why?
- Now, the minimum MSE (hidden behind the bias at this scale), is found at model 128, as expected.

Regularization

- The model/feature selection scheme we have just seen is a type of *regularization*
  - Limiting the model capacity to avoid overfitting the training data
- In practice, exhaustive search of all possible models or feature combinations is not feasible
  - In this case, regularization amounts to using a high capacity model/large number of features, but limiting the influence of less important features on the hypothesis

Regularization

- We can limit the influence of certain features by making sure their associated coefficients don't become too large:
  - Make large parameters/coefficient equate to larger loss

Regular mean squared error loss that we have used previously

Scalar hyperparameter

$$\mathcal{L}(h_{\mathbf{w}}(\mathbf{u}), y) = \mathcal{L}_{MSE} + \lambda \mathcal{L}_{\mathbf{w}}$$

Regularization term of the loss function that penalizes large parameters

name	function			
l_0 / Best Subset	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_0$	$=$	$\sum_i I[w_i \neq 0]$
l_1 / Lasso	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_1$	$=$	$\sum_i \lvert w_i \rvert$
l_2^2 / Ridge	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_2^2$	$= \mathbf{w} \cdot \mathbf{w} =$	$\sum_i w_i^2$
Elastic Net	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_1 + \alpha\lvert\mathbf{w}\rvert_2^2$	$=$	$\sum_i \lvert w_i \rvert + \alpha \sum_i w_i^2$
l_∞	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_\infty$	$=$	$\max_i \lvert w_i \rvert$

Slide based on lecture notes from R. Ptucha, A. Loui, A. Kwasinski, A. Savakis

Regularization

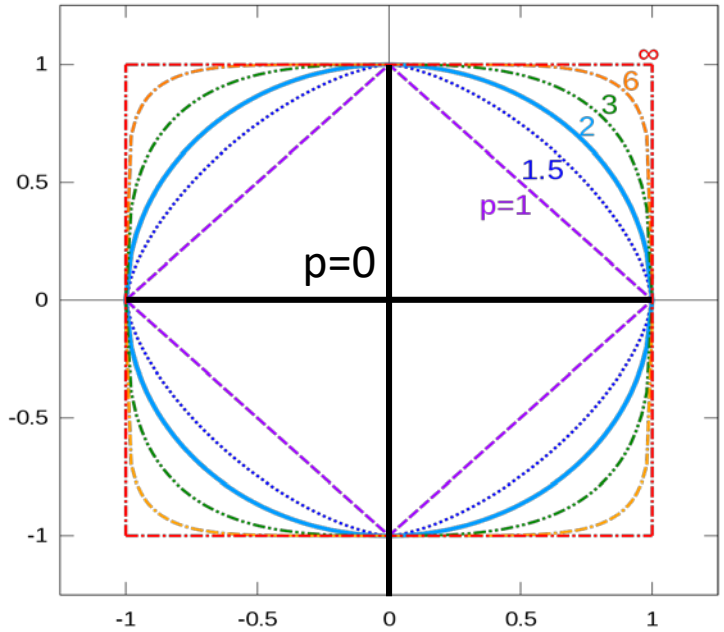
name	function			
l_0 / Best Subset	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_0$	$=$	$\sum_i I[w_i \neq 0]$
l_1 / Lasso	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_1$	$=$	$\sum_i \lvert w_i\rvert$
l_2^2 / Ridge	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_2^2$	$= \mathbf{w} \cdot \mathbf{w} =$	$\sum_i w_i^2$
Elastic Net	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_1 + \alpha\lvert\mathbf{w}\rvert_2^2$	$=$	$\sum_i \lvert w_i\rvert + \alpha \sum_i w_i^2$
l_∞	$h(\mathbf{w}) =$	$\lvert\mathbf{w}\rvert_\infty$	$=$	$\max_i \lvert w_i\rvert$

- Regularization terms are based on ℓ_p norms: $\|\mathbf{x}\|_p \equiv \left(\sum |x_i|^p\right)^{1/p}$
- ℓ_0: Best subset is a count of the number of non-zero parameters
  - Minimizes the number of parameters that can participate
- ℓ_1: LASSO, minimizes sum of absolute value of coefficients, "L-1 norm"
  - **L**east **A**bsolute **S**hrinkage and **S**election **O**perator
  - Forces coefficients to get driven towards zero
  - Good when lots of features and we want to remove many of them
- ℓ_2: Ridge minimizes sum of square of coefficients
  - Does not force coefficients as close to zero as ℓ_1 norm
  - Can compensate for over fitting without removing features from model
- Elastic net: Weighted average of LASSO and Ridge
- ℓ_∞: The max absolute value of any coefficient
  - Limits the largest coefficient

Slide based on lecture notes from R. Ptucha, A. Loui, A. Kwasinski, A. Savakis

Regularization

- Different regularization terms base on ℓ_p norms can be thought of geometrically:
  - The ℓ_p norm of any vector from the origin to any point on one of the $p = \cdots$ curves is 1



https://en.wikipedia.org/wiki/Lp_space#/media/File:Vector-p-Norms_qtl1.svg

Pseudoinverse Method with ℓ_2 Regularization

$$\mathcal{L}(h_{\mathbf{w}}(\mathbf{u}), y) = \mathcal{L}_{MSE} + \lambda \sum_{i=1}^{N} w_i^2$$

Note that we usually don't penalize w_0 (the bias)

Identity matrix with element 1,1 equal to 0

$$\mathbf{w} = \left(\mathbf{U}^T\mathbf{U} + \lambda \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & \\ 0 & & \ddots \\ & & & 1 \end{bmatrix} \right)^{-1} \mathbf{U}^T\mathbf{y}$$

Gradient Descent with Regularization

$$\mathcal{L}(h_{\mathbf{w}}(\mathbf{u}), y) = \mathcal{L}_{MSE} + \lambda \mathcal{L}_{\mathbf{w}}$$

$$\mathbf{w} \coloneqq \mathbf{w} - \alpha[\nabla_{\mathbf{w}} \mathcal{L}_{MSE} + \lambda \nabla_{\mathbf{w}} \mathcal{L}_{\mathbf{w}}]$$

$$w_i \coloneqq w_i - \alpha \left[\frac{1}{n} \sum_{p=1}^{n} (\hat{y}^p - y^p) u_i + \lambda \frac{\partial \mathcal{L}_{\mathbf{w}}}{\partial w_i} \right]$$

Example: ℓ_2 Regularization:

- Consider our previous polynomial fitting problem: $\hat{y} = \sum_{i=0}^{10} w_i u^i$