Data 100, Discussion 10

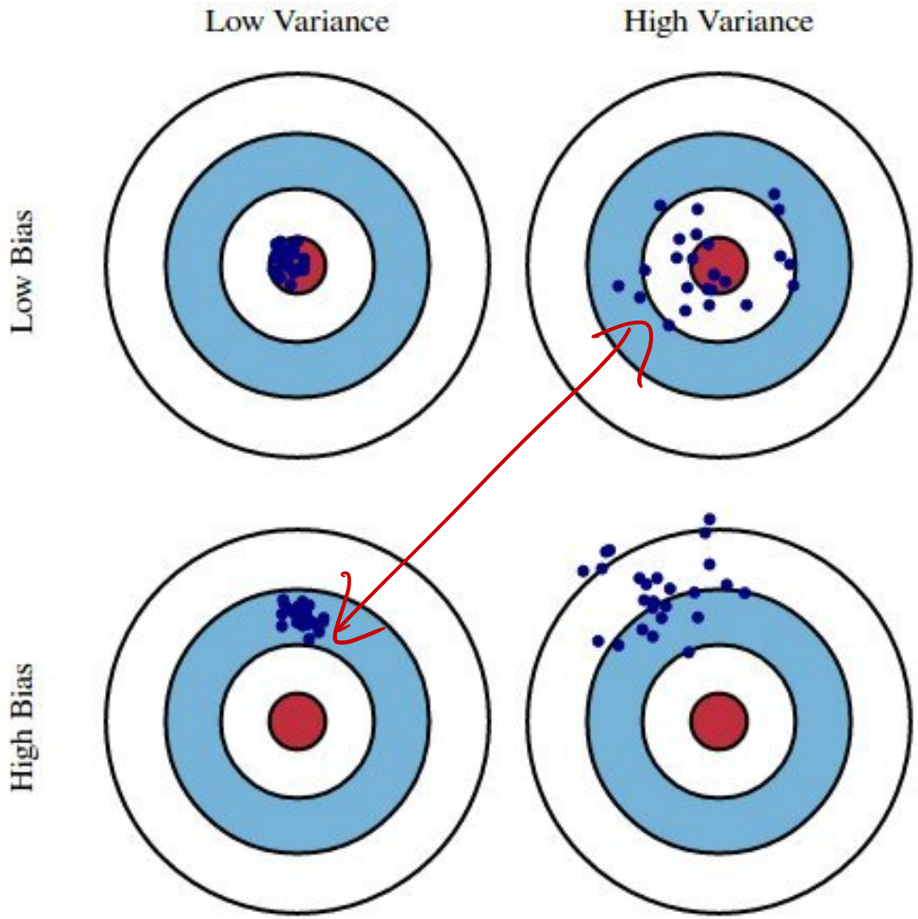
Suraj Rampure

Wednesday, October 30th, 2019

Agenda

- Bias-Variance Trade-off
- Regularization
- Cross Validation

Bias-Variance



Bias-Variance Decomposition

Suppose ϵ is some random variable such that $\mathbb{E}[\epsilon] = 0$ and $var[\epsilon] = \sigma^2$. Also, suppose we have Y generated as follows:

$$Y = h(x) + \epsilon$$

We collect some sample points $\{(x_i, y_i)\}_{i=1}^n$, and want to fit a model $f_\beta(x)$. We define the model **risk** as $\mathbb{E}[(Y - f_\beta(x))^2]$.

$$\mathbb{E}[(Y - f_\beta(x))^2] = \sigma^2 + (h(x) - \mathbb{E}[f_\beta(x)])^2 + \mathbb{E}[(\mathbb{E}[f_\beta(x)] - f_\beta(x))^2]$$

model risk — obs. variance — model bias^2 — model variance

This is sometimes referred to as the **bias-variance decomposition.**

Bias and Variance

Note: Both of the following depend on our prediction $f_\beta(x)$ (and hence, our choice of $\hat{\beta}$.)

Bias

$$h(x) - \mathbb{E}[f_\beta(x)]$$

- The difference between the true value and our expected prediction
- High bias typically indicates **underfitting**
- **Intuitively:** Model may be too basic to capture the underlying relationship

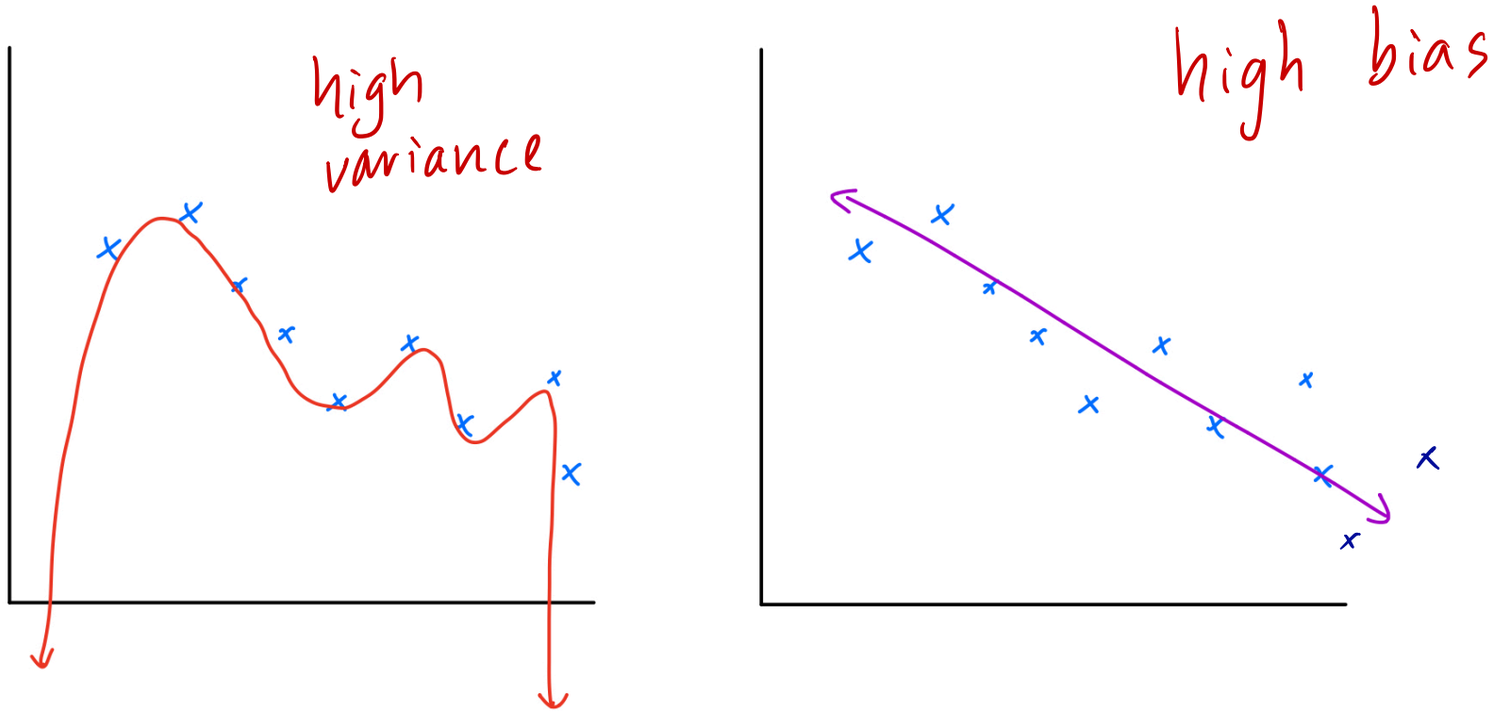
Model Variance:

Recall:
$Var[X] = E[(X - E[X])^2]$

$$\mathbb{E}[(\mathbb{E}[f_\beta(x)] - f_\beta(x))^2]$$

- Variance of $f_\beta(x)$, our prediction
- **Intuitively:** How much our predictions vary, given unseen data
- High variance indicates **overfitting** to training data

Polynomial regression with large d, small d:



The high degree polynomial model has lower bias, but higher variance, than the model on the right.

One way to interpret variance: In the model on the left, if we were to introduce a new point, our polynomial model would change significantly. However, on the right, introducing a new point is unlikely to change our model by much.

Model Complexity

Observation: We can make our training error arbitrarily close to 0, by adding more and more features.

Why don't we do this?

Overfitting to our data! We need to generalize.

Pitfalls of Ordinary Least Squares

In **Ordinary Least Squares**, our goal was to find the vector β that minimizes the following empirical risk:

$$R(\beta) = \frac{1}{n}||y - X\beta||_2^2$$

The **optimal value of** β (i.e. $\hat{\beta}$) is given by

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

Issues with OLS:

- Solution doesn't always exist (if X is not full-rank, $X^T X$ will not be full rank)
- Potential overfitting to training set – model can be too complex

values in $\vec{\beta}$ may be too large

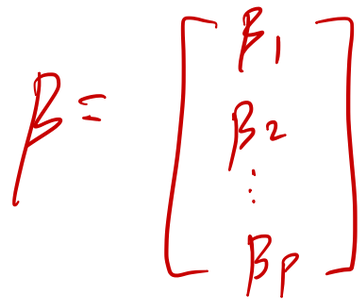
Pitfalls of Ordinary Least Squares

Solution: **Add penalty on magnitude of β.**

Now, our optimization problem is to find the β that minimizes

$$R(\beta) = \frac{1}{n}||y - X\beta||_2^2 + \lambda S(\beta)$$

- If $S(\beta) = \sum_{i=1}^{p} \beta_i^2 = ||\beta||_2^2$, we are performing L_2 regularization, called **ridge regression**
- If $S(\beta) = \sum_{i=1}^{p} |\beta_i| = ||\beta||_1$, we are performing L_1 regularization, called **LASSO regression**



- Note: $||x||_2 = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$, and $||x||_1 = |x_1| + |x_2| + ... + |x_n|$.

Ridge Regression

When we use the L_2 vector norm for the penalty term, our objective function becomes

$$R(\beta) = \frac{1}{n}||y - X\beta||_2^2 + \lambda||\beta||_2^2$$

$= \beta_1^2 + \beta_2^2 + \cdots + \beta_p^2$

Solution can also be determined using vector calculus.

$$\Rightarrow \hat{\beta_{ridge}} = (X^T X + \lambda I)^{-1} X^T y$$

- λ represents the penalty on the size of our model. It is a **hyperparameter**, in that we get to choose it as opposed to learn it from our data. We will discuss this more shortly.
- Unlike OLS, Ridge Regression always has a unique solution!

LASSO Regression

When we use the L_1 vector norm for the penalty term, our objective function becomes

$$R(\beta) = \frac{1}{n}\left|\left|y - X\beta\right|\right|_2^2 + \lambda\left|\left|\beta\right|\right|_1 \quad = |\beta_1| + |\beta_2| + \cdots + |\beta_p|$$

Unlike OLS and Ridge Regression, there is (in general) no closed form solution. Need to use a numerical method, such as gradient descent.

- Again, λ represents the penalty on the size of our model.
- LASSO regression encourages sparsity, that is, it sets many of the entries in our β vector to 0. LASSO effectively selects features for us, and also makes our model less complex (many weights set to 0 ——–> less features used ———> less complex)

Fun fact: LASSO stands for Least Absolute Shrinkage and Selection Operator.

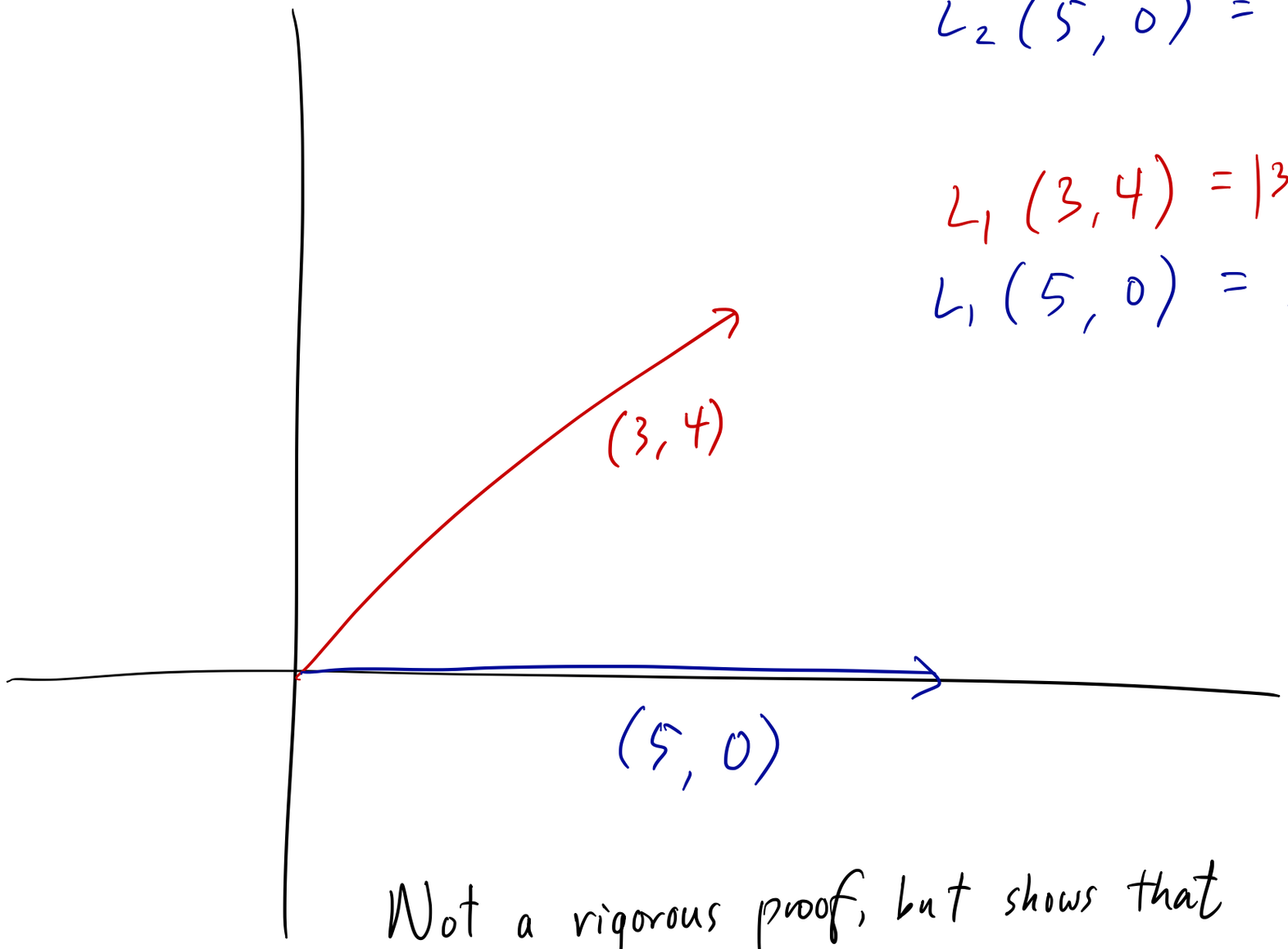
L_1 vs. L_2 vector norms: $(3, 4)$ vs. $(5, 0)$

$L_2(3, 4) = 5$

$L_2(5, 0) = 5$

$L_1(3, 4) = |3| + |4| = 7$

$L_1(5, 0) = 5$



Not a rigorous proof, but shows that

L_1 prefers $\vec{\beta}$ with some entries being 0

Regularization and Bias / Variance

Let's analyze the objective function for ridge regression (however, the analysis is the same for LASSO).

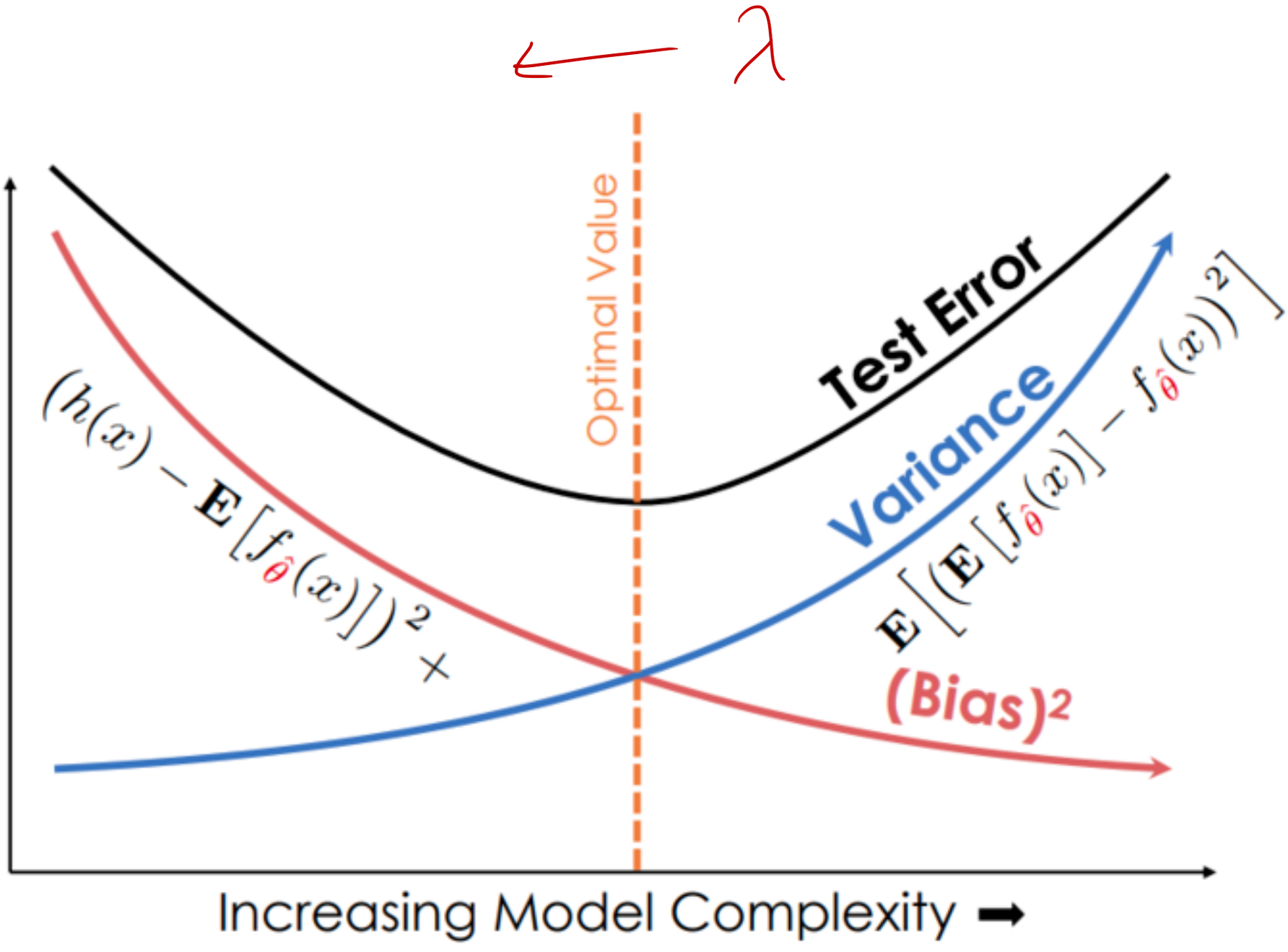
$$R(\beta) = \frac{1}{n}\left|\left|y - X\beta\right|\right|_2^2 + \lambda\left|\left|\beta\right|\right|_2^2$$

As λ increases, model complexity decreases.

- This is because increasing λ increases the penalty on the magnitude of β.
- Since we are trying to minimize the objective, if λ increases, $\left|\left|\beta\right|\right|_2^2$ must decrease.

As a result, as λ increases, bias increases, and model variance decreases.

- Bias increases because our model becomes less complex, and thus more general.
- Variance decreases because, again, our model becomes more general.



Reminder: *Model complexity and λ are inversely related!*

Motivating Cross Validation

Question: How do we select a subset of features to use? How do we select a value of λ?

One approach:

1. Select several different candidate values of λ, and train our model on each of them
2. Select the λ of the model that had the lowest training error

Why is this not a great approach?

Overfitting to our data: no indication of if this will generalize

Motivating Cross Validation

A slightly more robust approach:

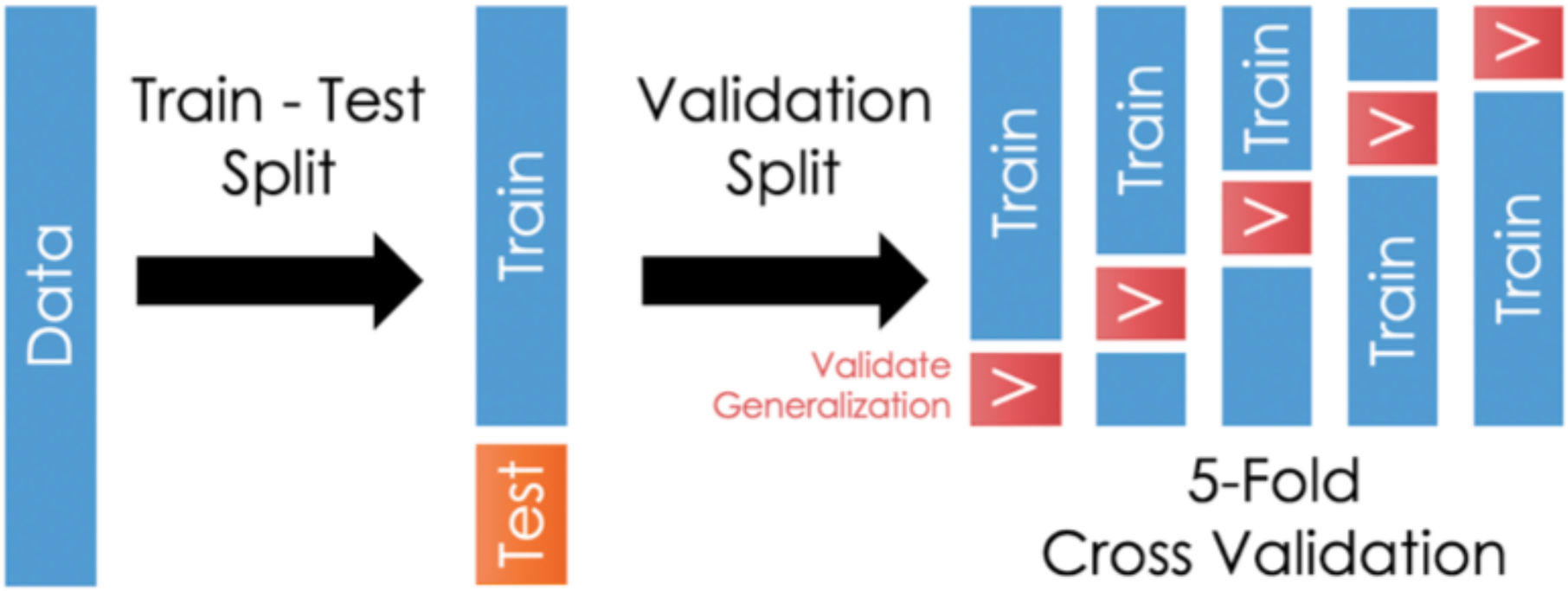
1. Split data into `train` and `test` sets
2. Select several different candidate values of λ, and train our model on each of them using the `train` data
3. Select the λ of the model that had the lowest error on the `test` set

Why is this better than the previous approach? evaluating on some unseen data

Why is it still not great? still could overfit to test data

Cross Validation

Cross validation simulates multiple train-test splits on the training data.



Pseudocode for Cross Validation

```
split data into train and test
split train into n disjoint partitions (folds)

errors_for_lambda = []

for each candidate value of lambda:
        errors_for_folds = []
        for i = 1, 2, ..., n:
                set fold i to be "validation"
                set all other folds to be "train"
                train model on "train"
                evaluate error using "validation", and add to errors_for_folds
        put mean(errors_for_folds) into errors_for_lambda


select lambda with lowest entry in errors_for_lambda
```

Note: *Here, we used the specific example of choosing different λ values. However, this exact procedure holds for choosing different subsets of features.*