

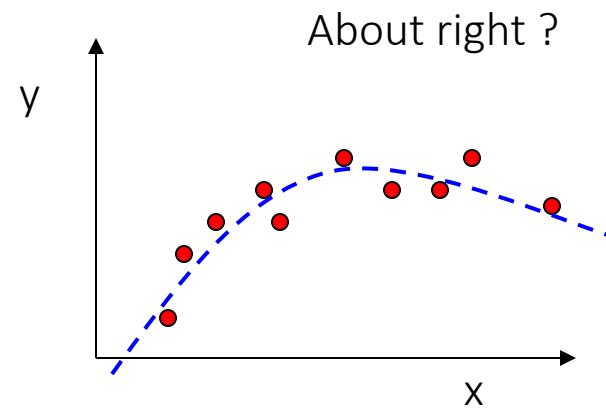
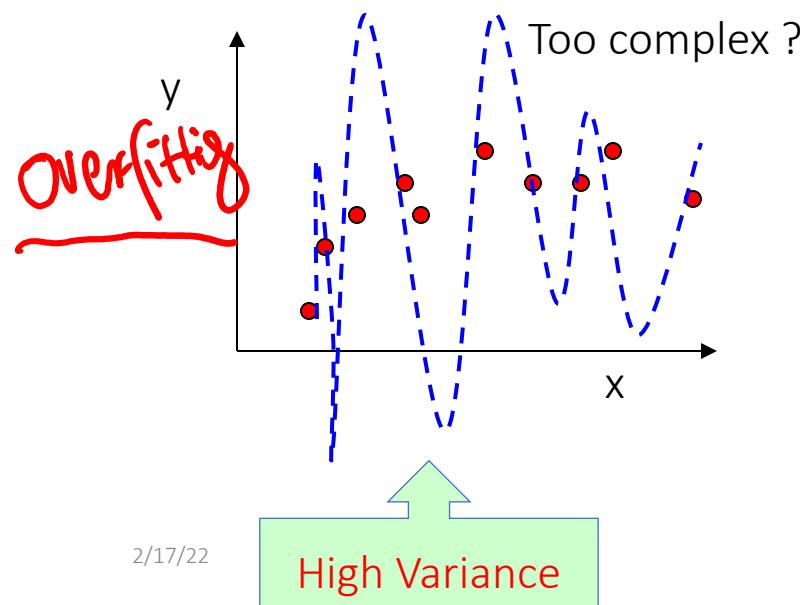
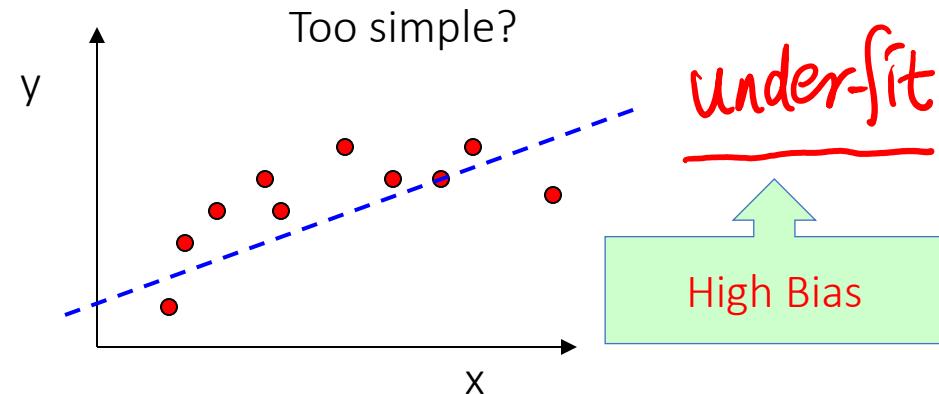
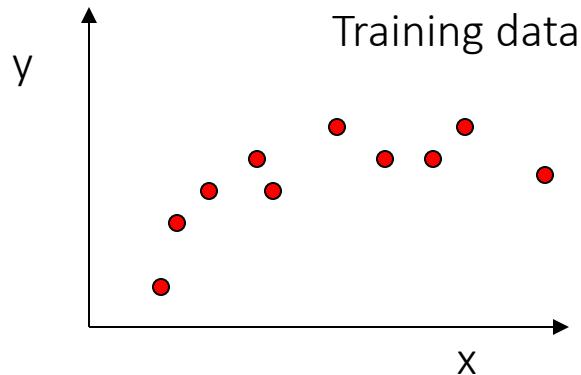
# UVA CS 4774: Machine Learning

## Lecture 9: Bias-Variance Tradeoff

Dr. Yanjun Qi

University of Virginia  
Department of Computer Science

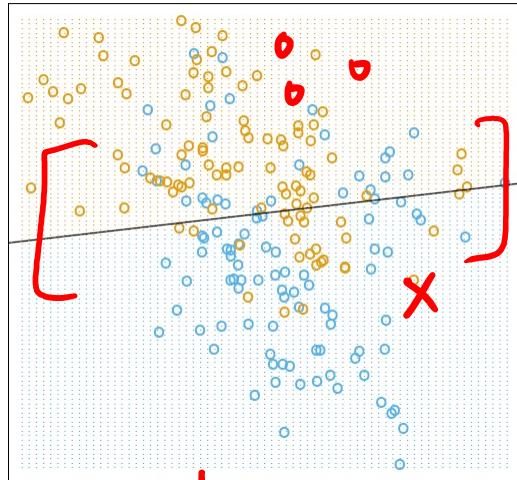
# Complexity / Goodness of Fit / Generalization



What ultimately matters: GENERALIZATION

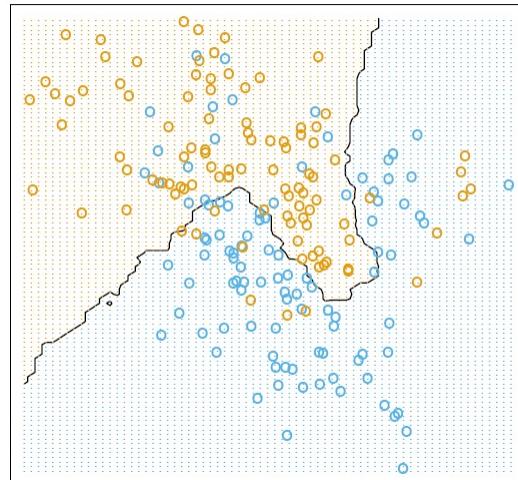
# Complexity / Goodness of Fit / Generalization: Decision boundaries in global vs. local models

$y \in \{\text{orange, blue}\}$



**underfit**  
Linear classification

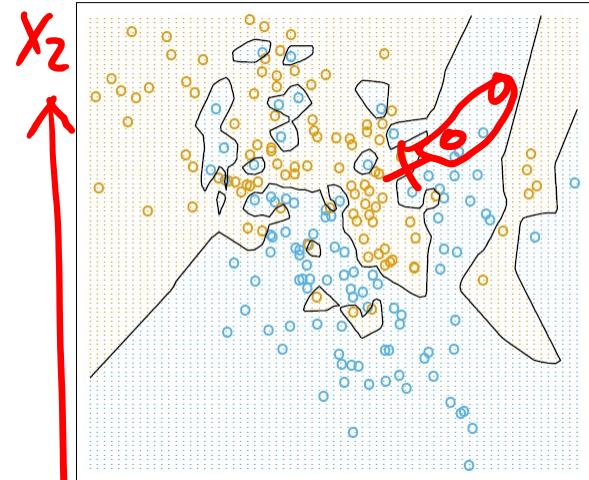
- global
- stable
- can be inaccurate



15-nearest neighbor

- K acts as a smoother

- local
- accurate
- unstable



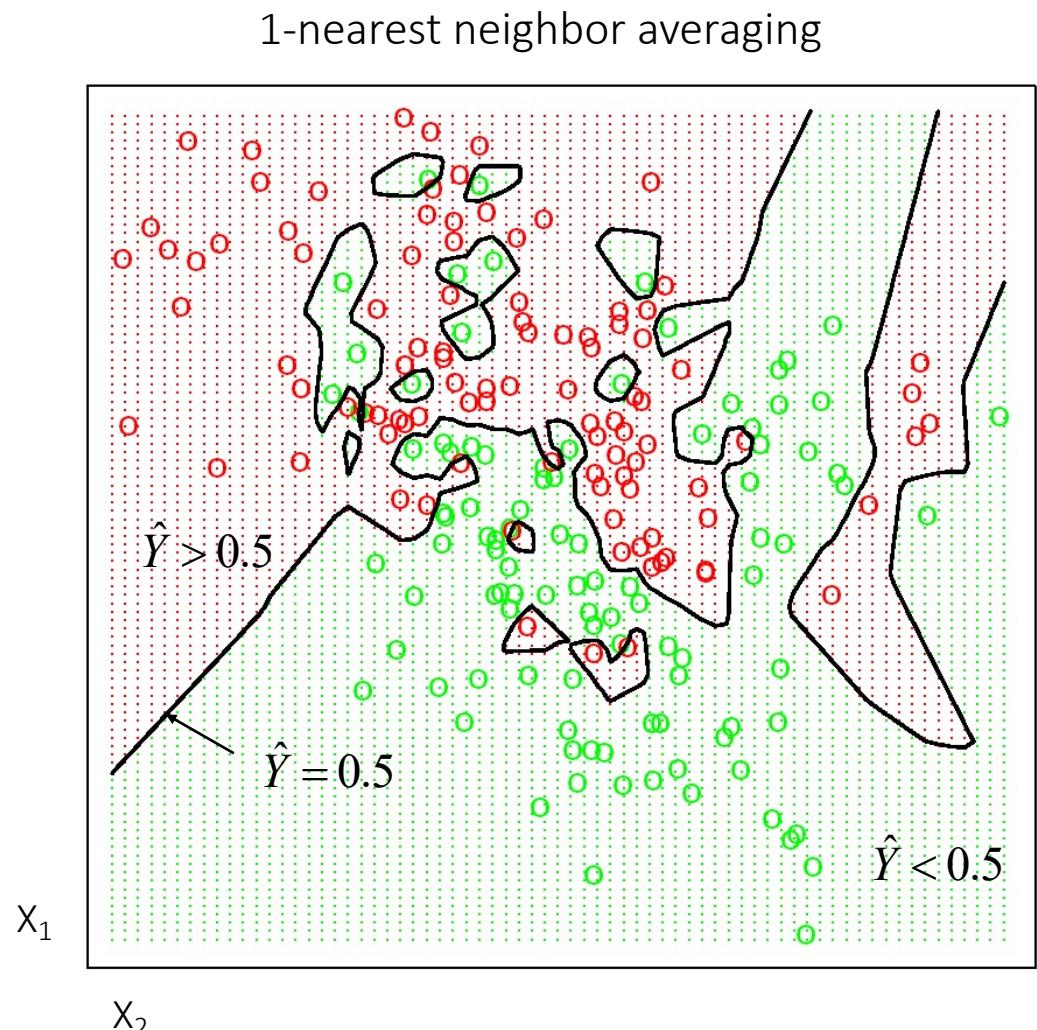
1-nearest neighbor

**Overfitting**

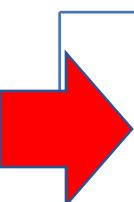
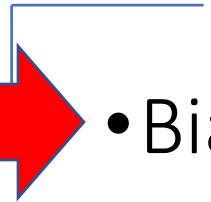
What ultimately matters: GENERALIZATION

# Lesson Learned : Training Error from KNN

- When  $k = 1$ ,
- No misclassifications (on training): **Overfit**
- Minimizing training error is not always good (e.g., 1-NN)



# Roadmap

- 
- 
- 
- Bias-variance decomposition
  - Bias-Variance Tradeoff / Model Selection
  - Remedy when Overfit / Underfit

# Review: Mean and Variance of Random Variable (RV)

$X$ : random variables written in capital letter

$\Rightarrow \vec{X} = [X_1, X_2, \dots, X_p]$  random vector

$\Rightarrow Y$  random variable

$\Rightarrow f(\vec{X})$  random variable  
e.g.  $\theta^T \vec{X}$

$\Rightarrow \vec{\theta}$  random variable

# Review: Mean and Variance of Random Variable (RV)

Bernoulli:  $t \in \{0, 1\}$

$$\begin{cases} P(t=0) = 0.1 \\ P(t=1) = 0.9 \end{cases}$$

- Mean (Expectation):

- Discrete RVs:

$$E(X) = \sum_{v_i} v_i * P(X = v_i) = (-1) \times 0.1 + 1 \times 0.9$$
$$E(t) = -0.1 + 0.9$$

$$= 0.8$$

- Continuous RVs:

$$E(X) = \int_{-\infty}^{+\infty} x * p(x) dx$$
$$X \sim N(2, \sigma^2)$$
$$E(X) = 2$$

# Review: Mean and Variance of Random Variable (RV)

Bernoulli:  $t \in \{0, 1\}$

$$\begin{cases} -1 : 0.1 \\ 1 : 0.9 \end{cases} \Rightarrow g(t) = 2t$$

- Mean (Expectation):

- Discrete RVs:

$$E(g(t)) = \frac{2 \times (-1) \times 0.1 + 2 \times (1) \times 0.9}{1.6}$$

$$E[aX] = aE[X] \quad E(g(X)) = \sum_{v_i} \underline{g(v_i)} P(X = v_i)$$

- Continuous RVs:

$$E(g(X)) = \int_{-\infty}^{+\infty} g(x) * p(x) dx$$

# Review: Mean and Variance of RV

- Variance:

$$Var(X) = E((X - \mu)^2)$$

$\boxed{\mu = E(X)}$

- Discrete RVs:

$$V(X) = \sum_{v_i} (v_i - \mu)^2 P(X = v_i)$$

- Continuous RVs:

$$V(X) = \int_{-\infty}^{+\infty} (x - \mu)^2 p(x) dx$$

# Statistical Decision Theory (Extra)

- Random input vector:  $X$

- Random output variable:  $Y$

- Joint distribution:  $\Pr(X, Y)$

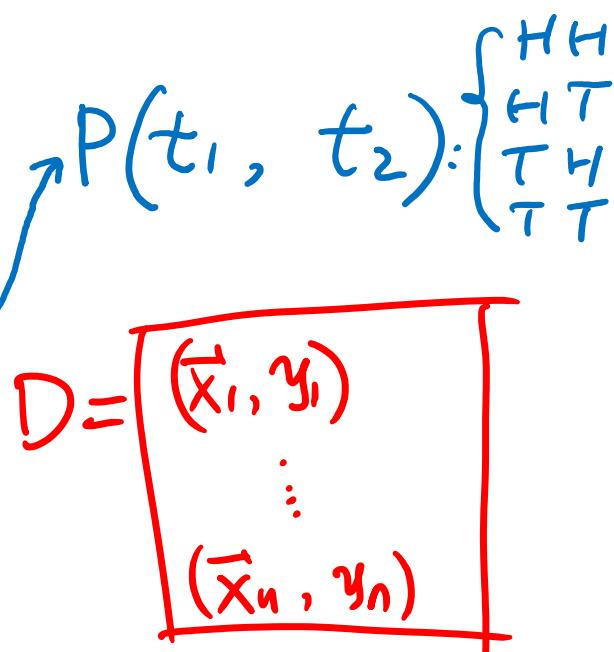
- Loss function  $L(Y, f(X))$

- Expected prediction error (EPE):

$$\text{EPE}(f) = \underbrace{\mathbb{E}(L(Y, f(X)))}_{\text{e.g.}} = \int L(y, f(x)) \Pr(dx, dy)$$

$$\text{e.g.} = \int (y - f(x))^2 \Pr(dx, dy)$$

e.g. Squared error loss (also called L2 loss )

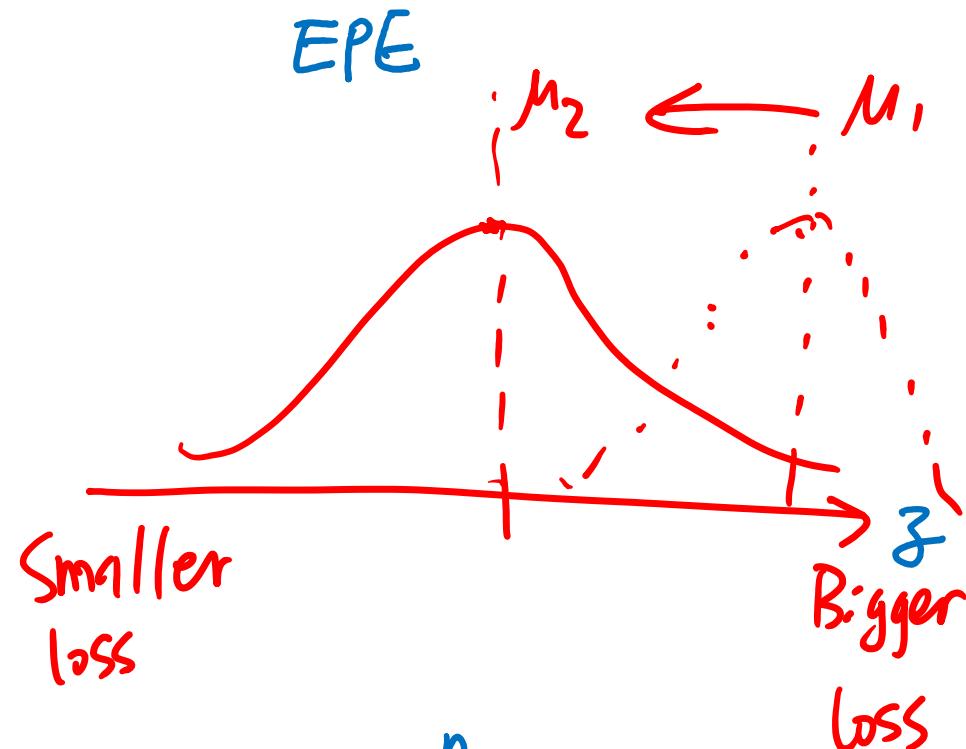


Consider population distribution

# Test Error to EPE: (Extra)

(Almost) the same definition:

- Expected Prediction Error:
- Expected Test Error:
- Expected Risk of a hypothesis
- Empirical Risk Minimization



- Expected prediction error (EPE):

$$\text{EPE}(f) = \mathbb{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

$$\text{e.g. } \int (y - f(x))^2 \Pr(dx, dy)$$

e.g. Squared error loss (also called L2 loss )

$$\sum_{i=1}^n L(y_i, f(x_i))$$

One way to define generalization: by considering the joint population distribution

# Decomposition of EPE

- When additive error model:  $Y = \underline{f}(X) + \epsilon, \epsilon \sim (0, \sigma^2)$
- Notations

- Output random variable:  $\underline{Y}$

- True function:  $f \rightarrow \text{true}$

- Prediction estimator:  $\hat{f} \rightarrow D \rightarrow \hat{f}$

$$\begin{aligned}
 EPE(x) &= E[(Y - \hat{f})^2 | X = x] \\
 &= E[((Y - f) + (f - \hat{f}))^2 | X = x] \\
 &= E[\underbrace{(Y - f)^2}_{\text{Irreducible / Bayes error}} | X = x] + E[(f - \hat{f})^2 | X = x] \\
 &= \sigma^2 + \text{Var}(\hat{f}) + \text{Bias}^2(\hat{f})
 \end{aligned}$$

Irreducible / Bayes error

# Bias-Variance Trade-off for EPE:

$$\text{EPE}(x) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

```
graph TD; A[noise2] --> B[Unavoidable error]; A --> C[Error due to incorrect assumptions]; A --> D[Error due to variance of training samples]; B --- E[bias2]; C --- F[variance]
```

$$\left\{ \begin{array}{l} Y = f(x) + \varepsilon \quad \varepsilon \sim N(0, \sigma^2) \\ \hat{f}_D(x) \end{array} \right. \Rightarrow \begin{array}{l} \text{---} \text{---} \text{---} \\ \text{f true} \\ \hat{f} \text{ estimated} \\ \bar{f} \text{ Expected} \\ \hat{f} \text{ Estimated} \end{array}$$

$$\begin{aligned} EPE &= E_{(X,Y)} [(Y - \hat{f}(x))^2] \\ &= E_{(X,Y)} [((Y - f) + (f - \hat{f}))^2] \\ &= \underbrace{E[(Y - f)^2]}_{\sigma^2} + E[(f - \hat{f})^2] \\ \text{Bayes error} &= \underbrace{\sigma^2}_{+} \underbrace{E[f - \hat{f}]^2}_{+} \end{aligned}$$

$$E[(f - \hat{f})^2]$$

$$= E((f - \bar{f}) + (\bar{f} - \hat{f}))^2]$$

$$= E[(f - \bar{f})^2] + E[(\bar{f} - \hat{f})^2]$$

*Bias<sup>2</sup>*                            *Variance*

---


$$E[2(f - \bar{f})(\bar{f} - \hat{f})] = 0$$

$$E(\hat{f}) = \bar{f} \Rightarrow \cancel{2E[\bar{f}\bar{f}] - 2E[\bar{f}\hat{f}]} + \cancel{- 2E[\bar{f}\bar{f}] + 2E[\bar{f}\hat{f}]}$$

- More so than just these intuitive descriptions, the expected test error mathematically decomposes into a sum of three corresponding parts. Begin by writing the model

$$Y = f(X) + \varepsilon,$$

where  $\varepsilon$  has mean zero, variance  $\sigma^2$ , and is independent of  $X$ . Note that the independence condition is the an actual (nontrivial) assumption. Recall that  $(x_i, y_i)$ ,  $i = 1, \dots, n$  are independent of each other and of  $(X, Y)$ , all with the same distribution. We'll look at the expected test error, conditional on  $X = x$  for some arbitrary input  $x$ . It follows that

$$\mathbb{E}[(Y - \hat{f}(x))^2 | X = x] = \sigma^2 + \underbrace{\mathbb{E}[(f(x) - \hat{f}(x))^2]}_{\text{Risk}(\hat{f}(x))}. \quad \leftarrow$$

The first term  $\sigma^2$  is the *irreducible error*, or sometimes referred to as the *Bayes error*, and the second term is called the risk, or mean squared error (MSE). The risk further decomposes into two parts, so that

$$\mathbb{E}[(Y - \hat{f}(x))^2 | X = x] = \underbrace{\sigma^2}_{\text{EPE}} + \underbrace{(f(x) - \mathbb{E}[\hat{f}(x)])^2}_{\text{BayesE}} + \underbrace{\mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]}_{\text{Var}(\hat{f}(x))}, \quad (2)$$

the latter terms being the squared *estimation bias* or simply *bias*, and the *estimation variance* or simply *variance*, respectively. The decomposition (2) is called the *bias-variance decomposition* or *bias-variance tradeoff*

$$E \left[ (Y - \hat{f}(x))^2 \right] = E \left[ (f(X) + \epsilon - \hat{f}(x))^2 \right]$$

$$= E \left[ (f(X) - \hat{f}(x))^2 \right] + 2E[\epsilon(f(x) - \hat{f}(x))] + E[\epsilon^2] = MSE(f, \hat{f}) + Var(\epsilon)$$

Assuming the Bayes error is independent of  $\hat{f}(x)$ ,

$$E[\epsilon(f(x) - \hat{f}(x))] = E[\epsilon]E[f(x) - \hat{f}(x)] = 0$$

$$E[\epsilon^2] = \sigma^2 + E[\epsilon]^2 = \sigma^2$$

$$E \left[ (f(X) - \hat{f}(x))^2 \right] = E \left[ ((f(X) - E[\hat{f}(x)]) + (E[\hat{f}(x)] - \hat{f}(x)))^2 \right]$$

$$= E \left[ (f(X) - E[\hat{f}(x)])^2 + 2(f(X) - E[\hat{f}(x)])(E[\hat{f}(x)] - \hat{f}(x)) + (E[\hat{f}(x)] - \hat{f}(x))^2 \right]$$

$$= E \left[ (f(X) - E[\hat{f}(x)])^2 \right] + 2E \left[ (f(X) - E[\hat{f}(x)])(E[\hat{f}(x)] - \hat{f}(x)) \right] + E \left[ (E[\hat{f}(x)] - \hat{f}(x))^2 \right]$$

We can show:

$$2E \left[ (f(X) - E[\hat{f}(x)])(E[\hat{f}(x)] - \hat{f}(x)) \right] = 2(f(X) - E[\hat{f}(x)])E[E[\hat{f}(x)] - \hat{f}(x)] = 0$$

Finally,

$$E \left[ (f(X) - \hat{f}(x))^2 \right] = E \left[ (f(X) - E[\hat{f}(x)])^2 \right] + E \left[ (E[\hat{f}(x)] - \hat{f}(x))^2 \right]$$

$$= Bias(f(x), \hat{f}(x))^2 + Var(\hat{f}(x))$$

Putting it all together:

$$E \left[ (Y - \hat{f}(x))^2 \right] = Bias(f(x), \hat{f}(x))^2 + Var(\hat{f}(x)) + \sigma^2$$

## Another View: BIAS AND VARIANCE TRADE-OFF for parameter estimation (Extra)

$$D_1 \rightarrow \hat{\theta}_1 / D_2 \rightarrow \hat{\theta}_2 / \dots / D_T \rightarrow \hat{\theta}_T$$

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2]$$

$$= E[((\hat{\theta} - \bar{\theta}) + (\bar{\theta} - \theta))^2]$$

$$= E[(\hat{\theta} - \bar{\theta})^2] + E[(\bar{\theta} - \theta)^2] + 2E[(\hat{\theta} - \bar{\theta})(\bar{\theta} - \theta)]$$

$$= \underbrace{Var(\hat{\theta})}_{\text{Error due to variance of training samples}} + \underbrace{Bias^2(\hat{\theta})}_{\text{Error due to incorrect assumptions}} + 0$$

$$E(\hat{\theta}) = \bar{\theta}$$

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] = Bias^2(\hat{\theta}) + Var(\hat{\theta})$$

## BIAS AND VARIANCE TRADE-OFF for Parameter Estimation (Extra)

- $\theta$ : true value (normally unknown)
  - $\hat{\theta}$ : estimator
  - $\bar{\theta} := E[\hat{\theta}]$  (mean, i.e. expectation of the estimator)
- Bias  $E[(\bar{\theta} - \theta)^2]$ 
    - measures **accuracy** or **quality** of the estimator
    - low bias implies on average we will accurately estimate **true parameter** from training data
  - Variance  $E[(\hat{\theta} - \bar{\theta})^2]$ 
    - Measures **precision** or **specificity** of the estimator
    - Low variance implies the estimator does not **change** much as **the training set varies**

# Model “bias” & Model “variance”

- Middle RED:
  - TRUE function

$\theta$ : red dot  
 $\hat{\theta}_{\text{if}}$ : blue dots

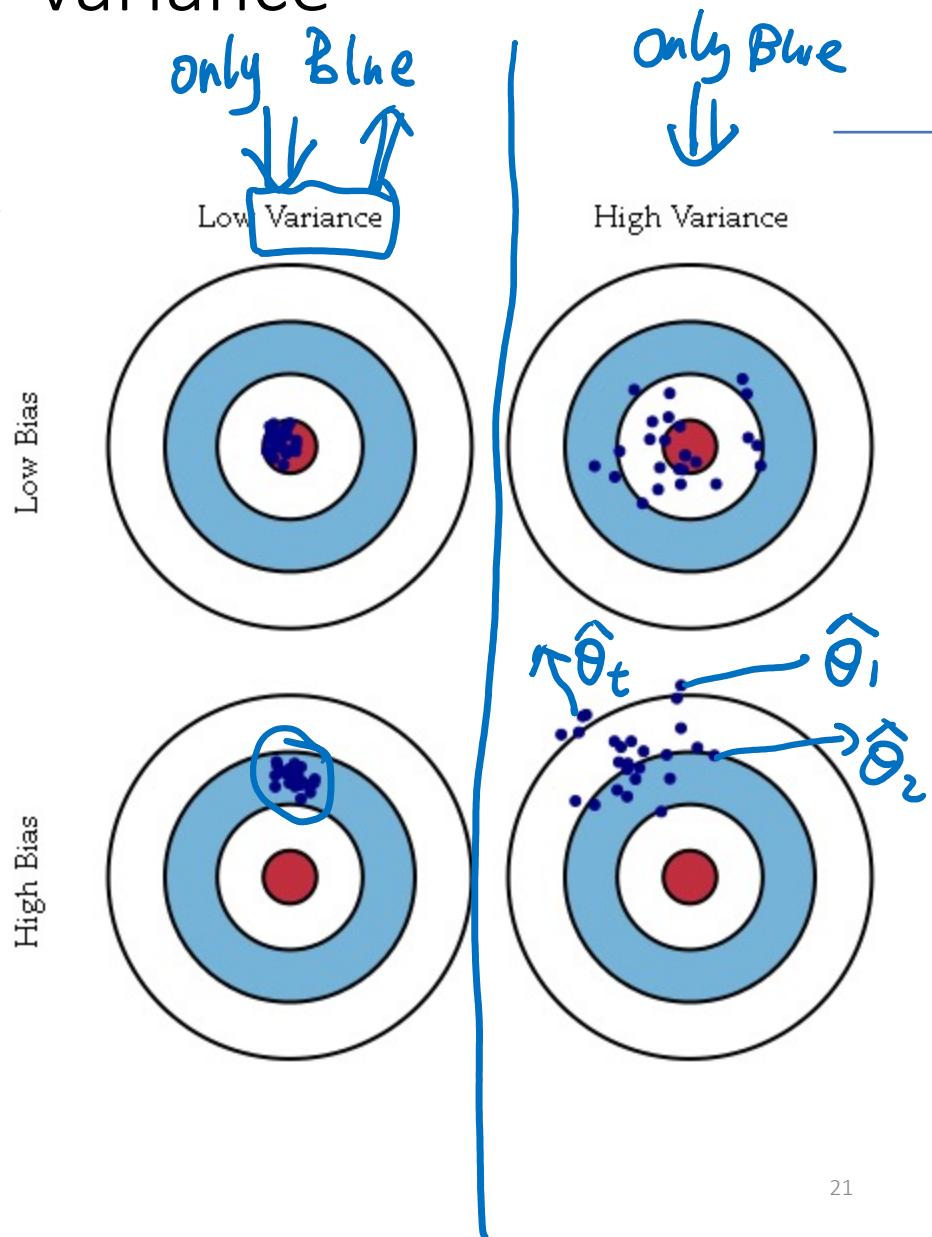
$\bar{\theta}$ : mean  
point all of Blue

- Error due to bias:
  - How far off in general from the middle red

$$E[(\bar{\theta} - \theta)^2]$$

- Error due to variance:
  - How wildly the blue points spread

$$E[(\hat{\theta} - \bar{\theta})^2]$$



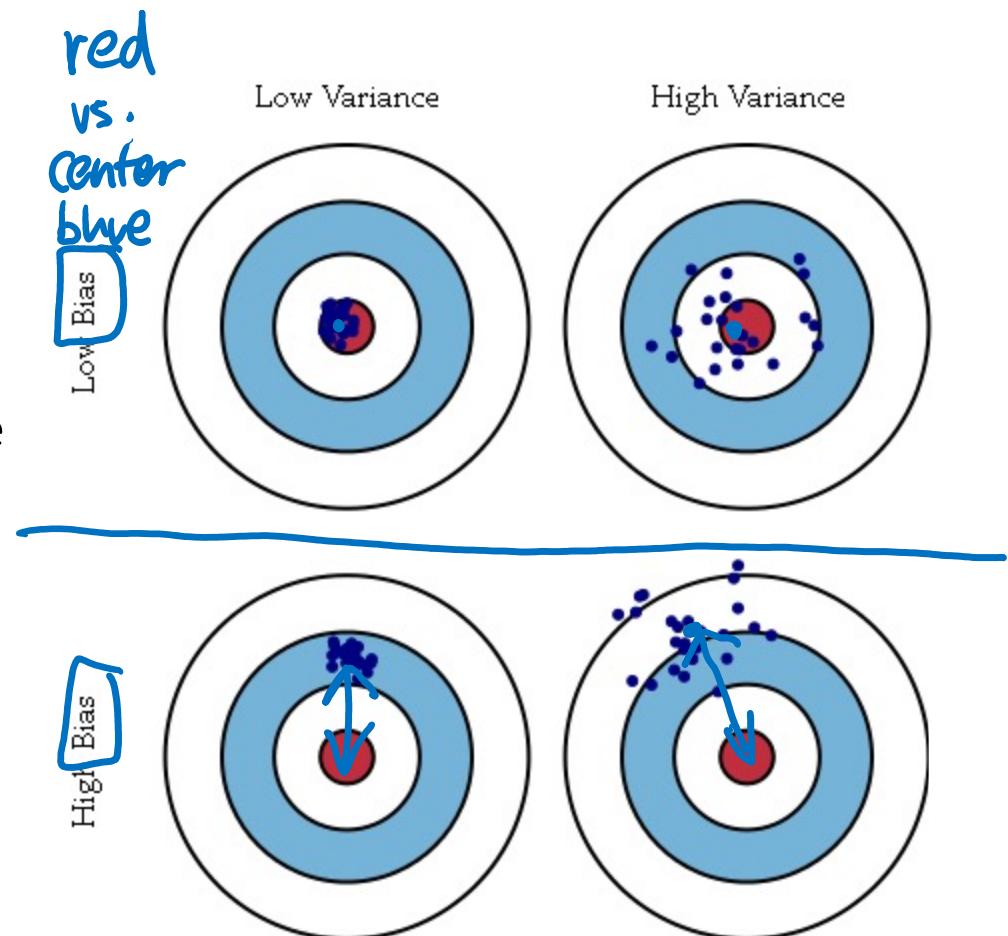
# Model “bias” & Model “variance”

- Middle RED:
  - TRUE function
- Error due to bias:
  - How far off in general from the middle red

$$E[(\bar{\theta} - \theta)^2]$$

- Error due to variance:
  - How wildly the blue points spread

$$E[(\hat{\theta} - \bar{\theta})^2]$$



# Model “bias” & Model “variance”

- Middle RED:
  - TRUE function

$\theta$   
(middle red)

- Error due to bias:
  - How far off in general from the middle red

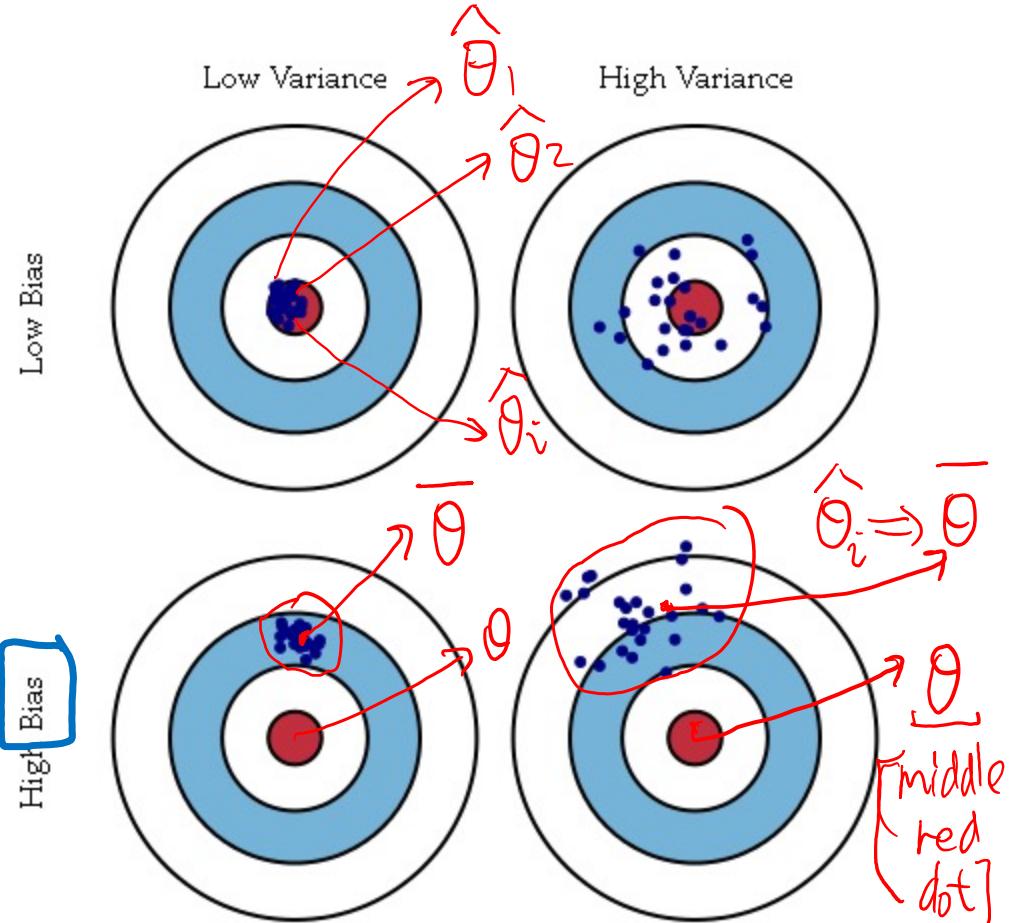
$$E(\theta - \bar{\theta})$$

mean of  $\hat{\theta}$

- Error due to variance:
  - How wildly the blue points spread

$$E((\hat{\theta} - \bar{\theta})^2)$$

$\{\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \dots\}$  Blue dots

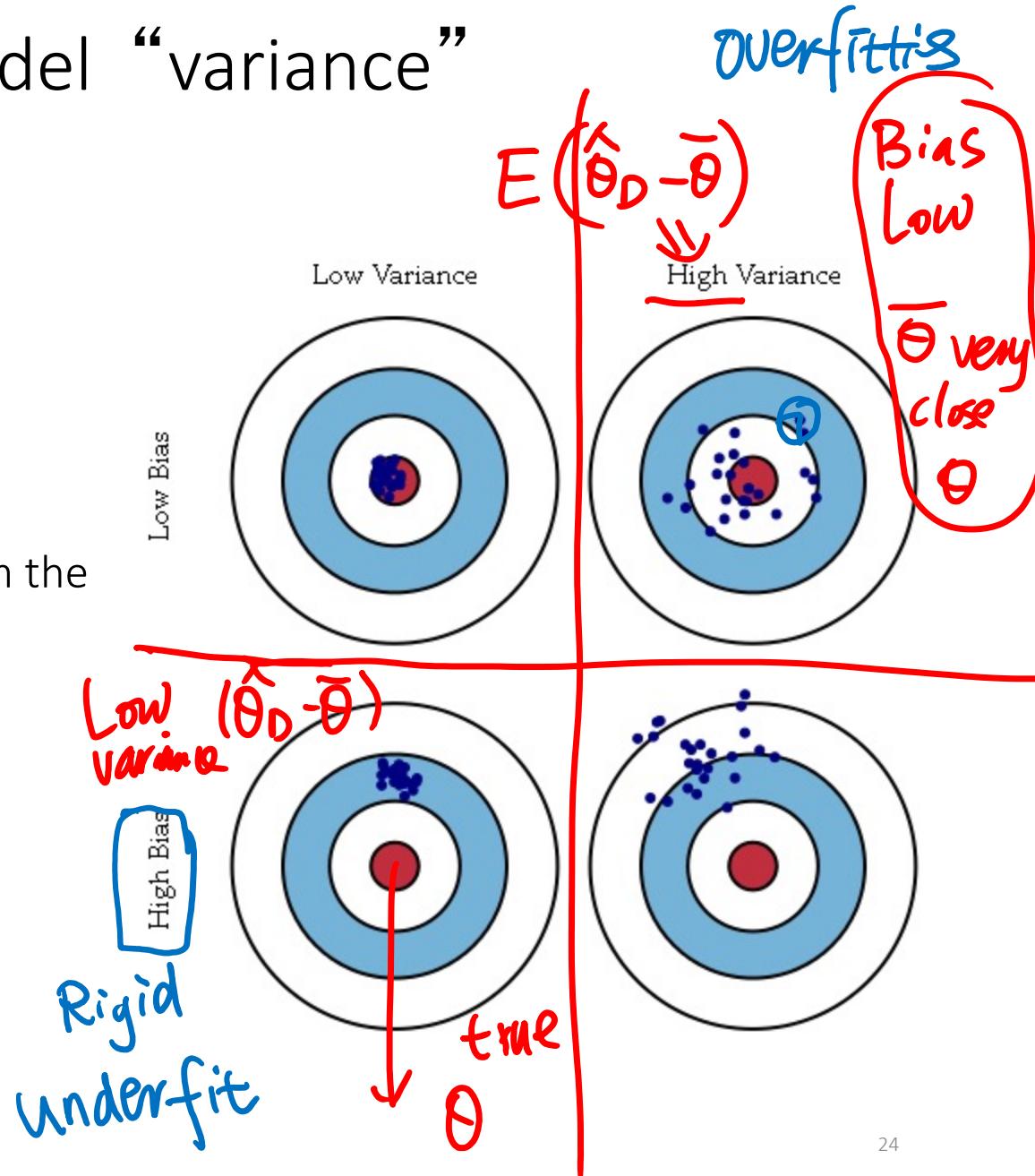


# Model “bias” & Model “variance”

- Middle RED:
  - TRUE function
- Error due to bias:
  - How far off in general from the middle red
- Error due to variance:
  - How wildly the blue points spread

$$E[(\bar{\theta} - \theta)^2]$$

$$E[(\hat{\theta} - \bar{\theta})^2]$$



# need to make assumptions that are able to generalize

- Underfitting: model is too “simple” to represent all the relevant characteristics
  - High bias and low variance
  - **High training error and high test error**
- Overfitting: model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - **Low training error and high test error**

# Thank You



# UVA CS 4774: Machine Learning

## Lecture 9: Bias-Variance Tradeoff

Module II

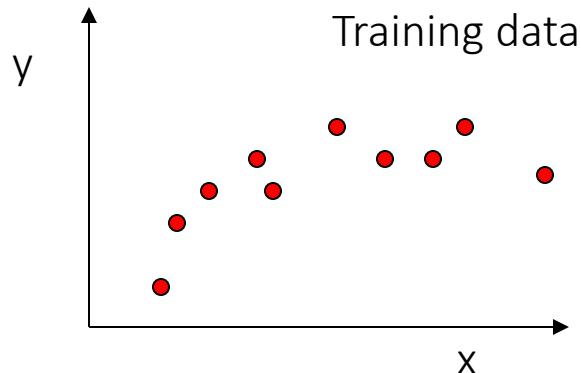
Dr. Yanjun Qi

University of Virginia  
Department of Computer Science

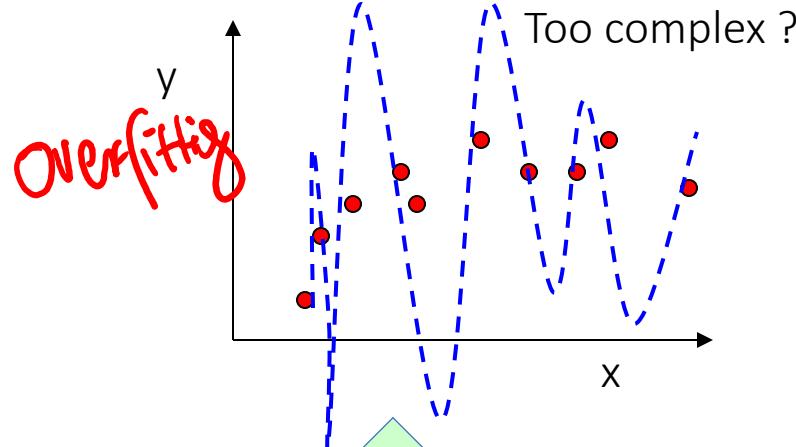
# Roadmap

- Bias-variance decomposition
- Bias-Variance Tradeoff / Model Selection
- Remedy when Overfit / Underfit

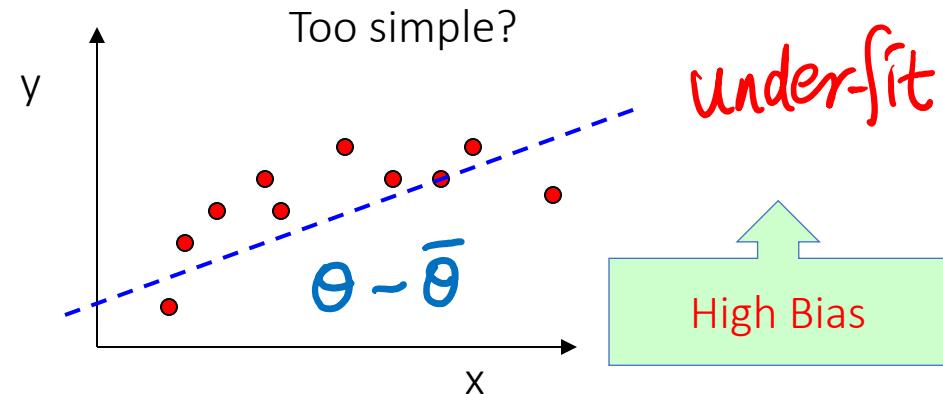
# Complexity / Goodness of Fit / Generalization



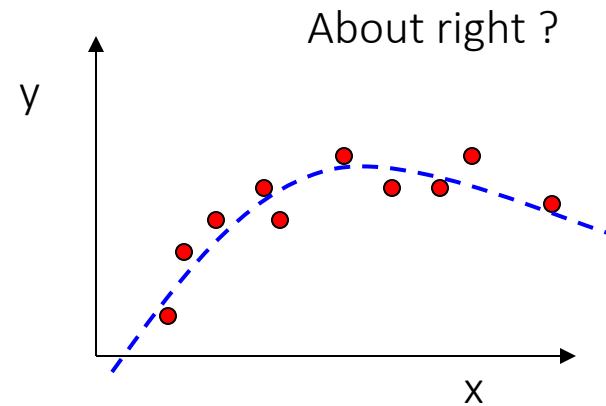
$$\hat{\theta}_D - \bar{\theta}$$



High Variance

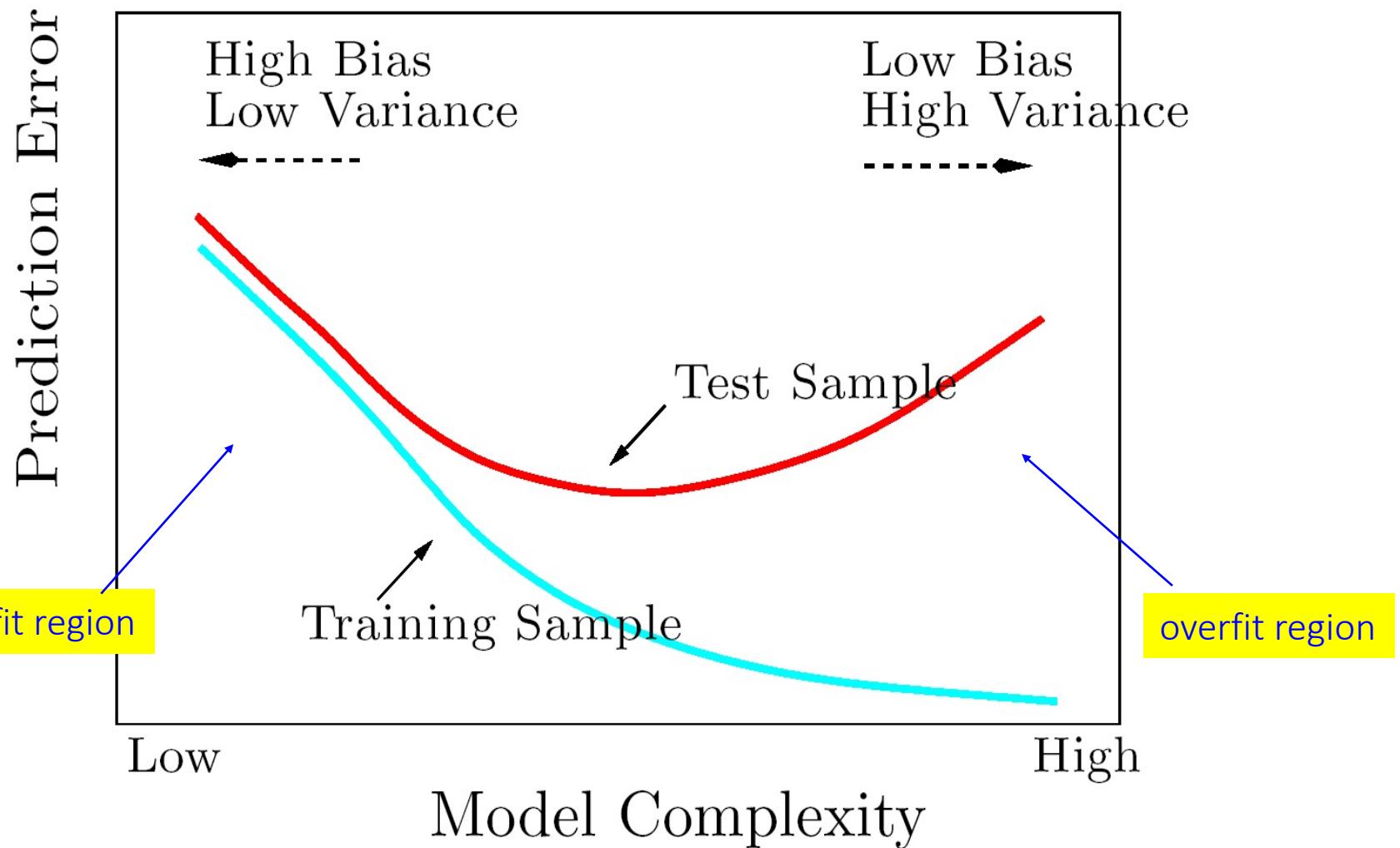


under-fit



What ultimately matters: GENERALIZATION

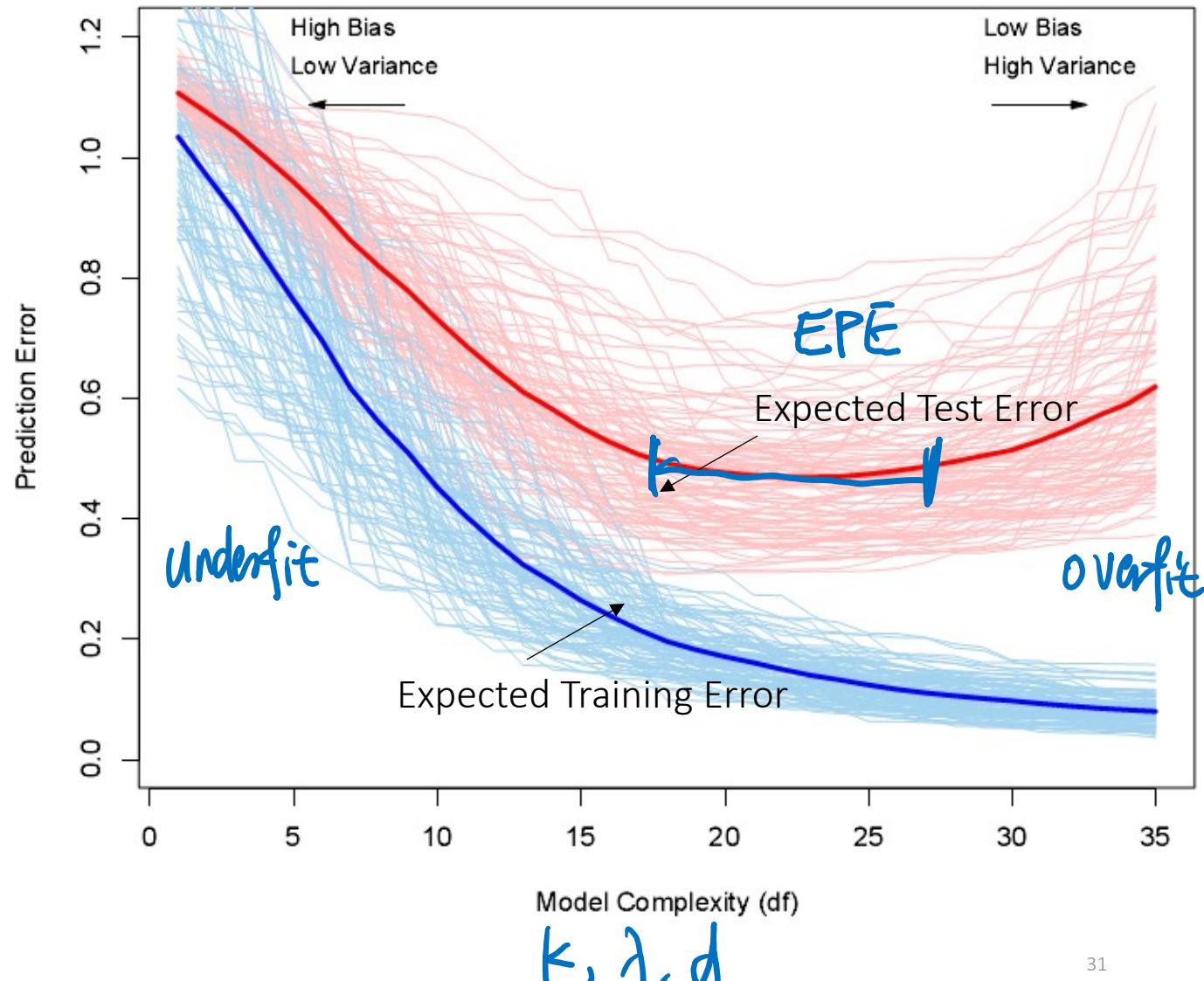
# Bias-Variance Tradeoff / Model Selection



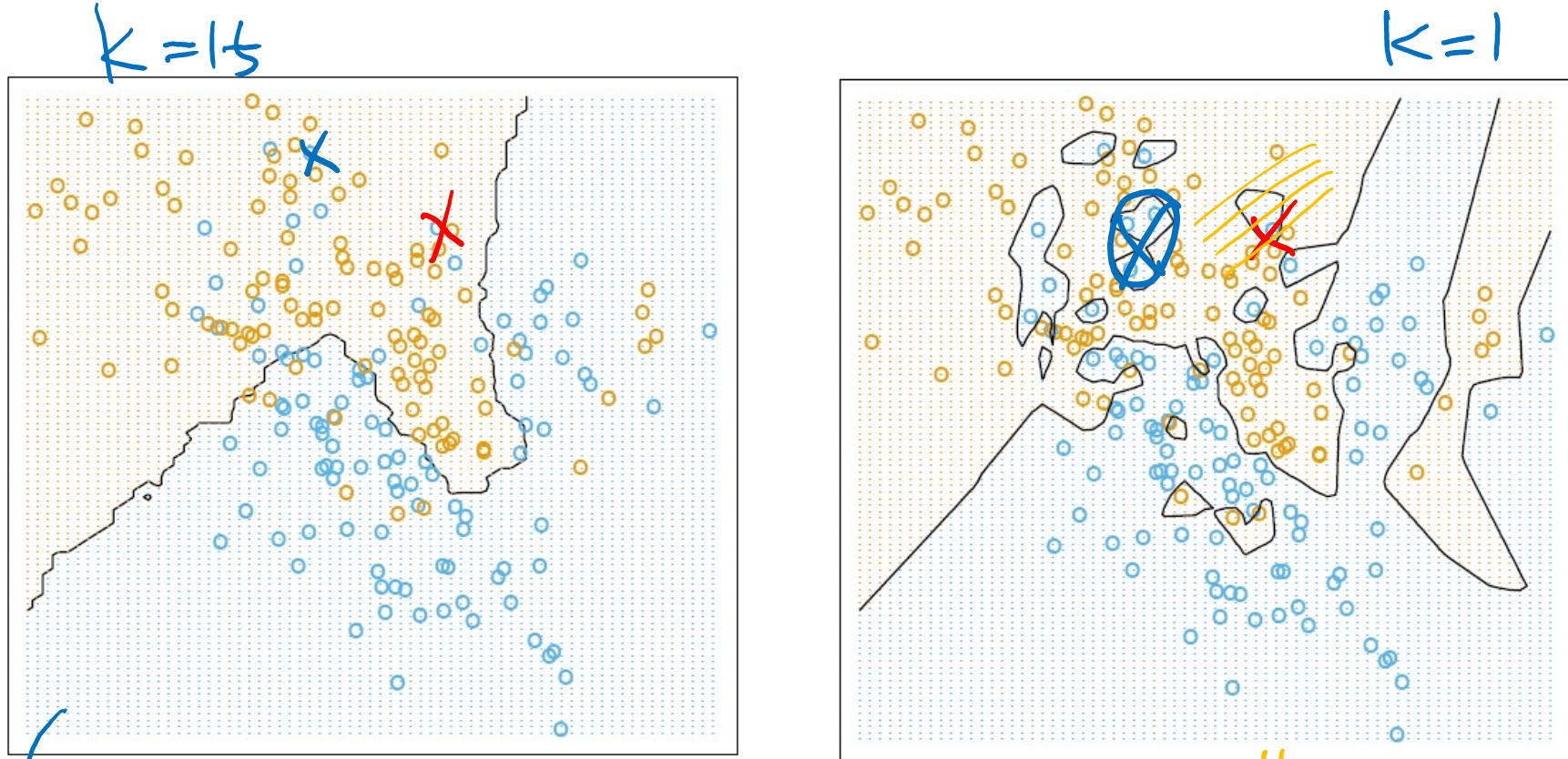
# (1) Randomness of Training Sets

$$\Pr(x, \tilde{y})$$

randomness  
of train  
test set



# Randomness of Train Set => Variance of Models, e.g.,



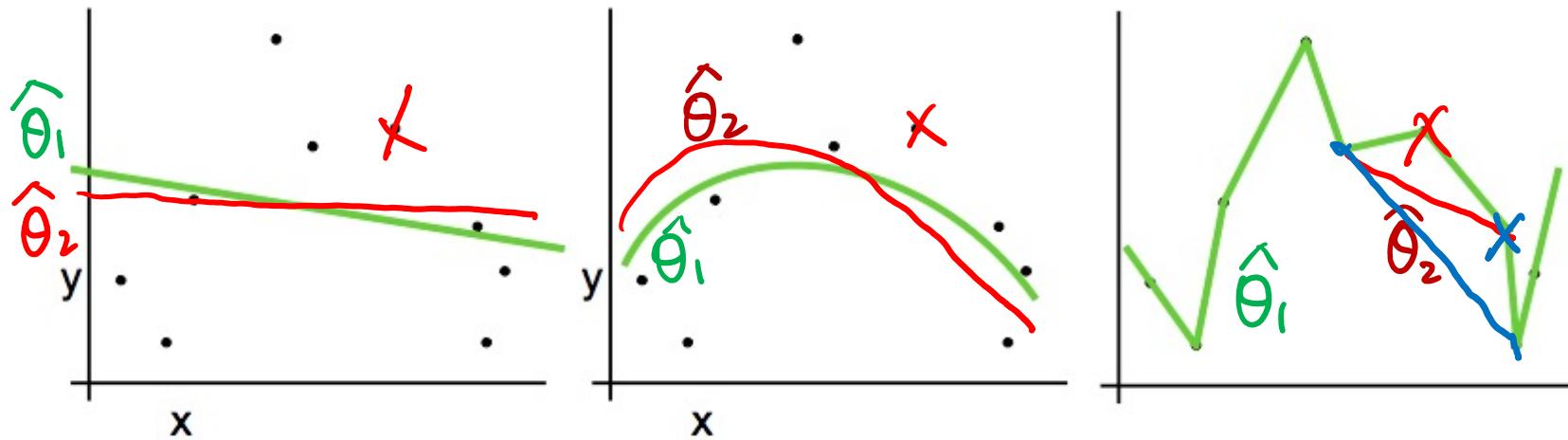
e.g. removing one train sample  
No change of decision boundary

2/17/22

↓  
decision boundary changed

32

Randomness of Train Set  
=> Variance of Models, e.g.,

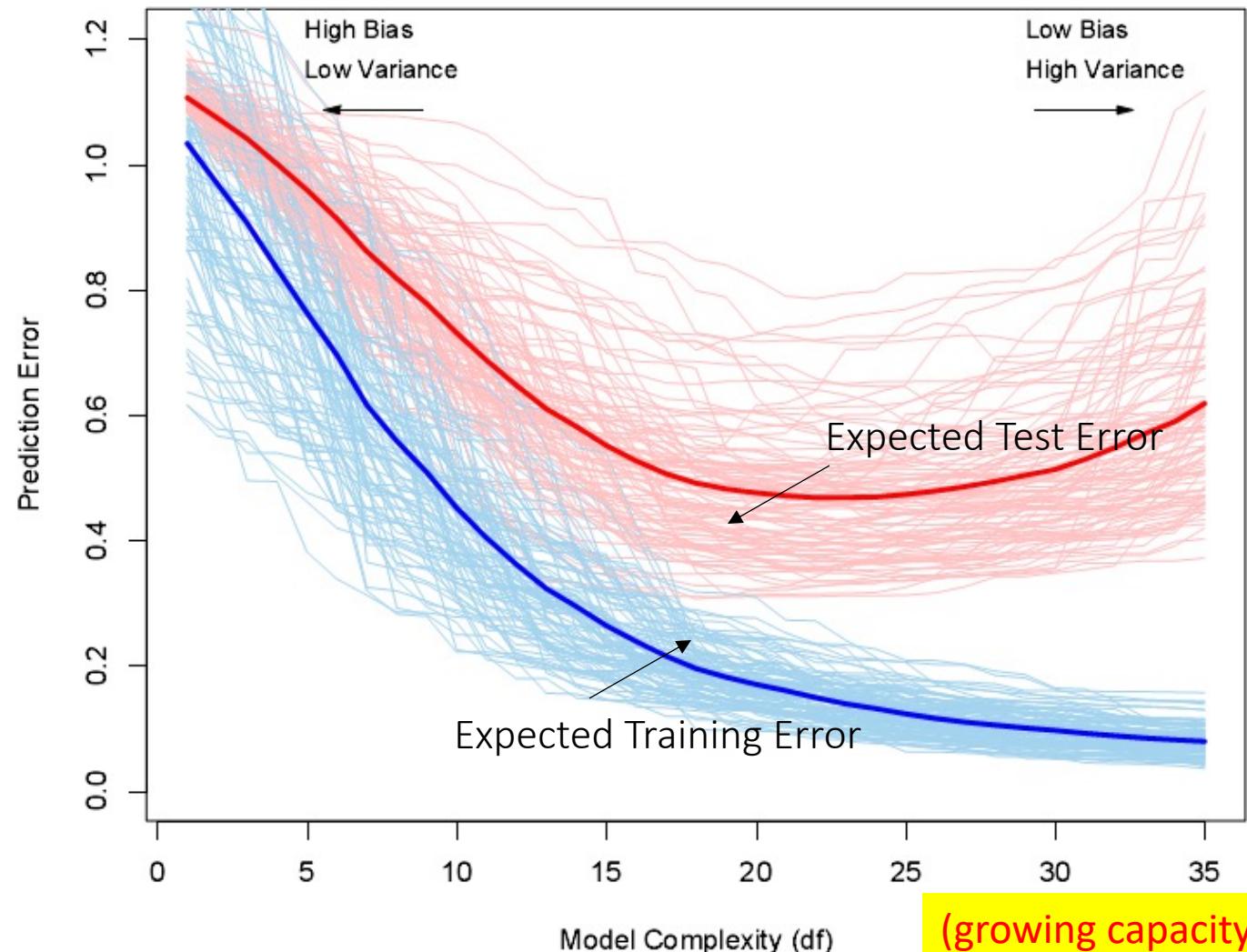


e.g. removing  
one training sample

when model complexity  $\uparrow \Rightarrow$  model variance  $\uparrow$

(2) Training error can always be reduced when increasing model complexity

$$\kappa = 1 \\ \text{Tr}(\kappa_{NN}) \approx 0$$

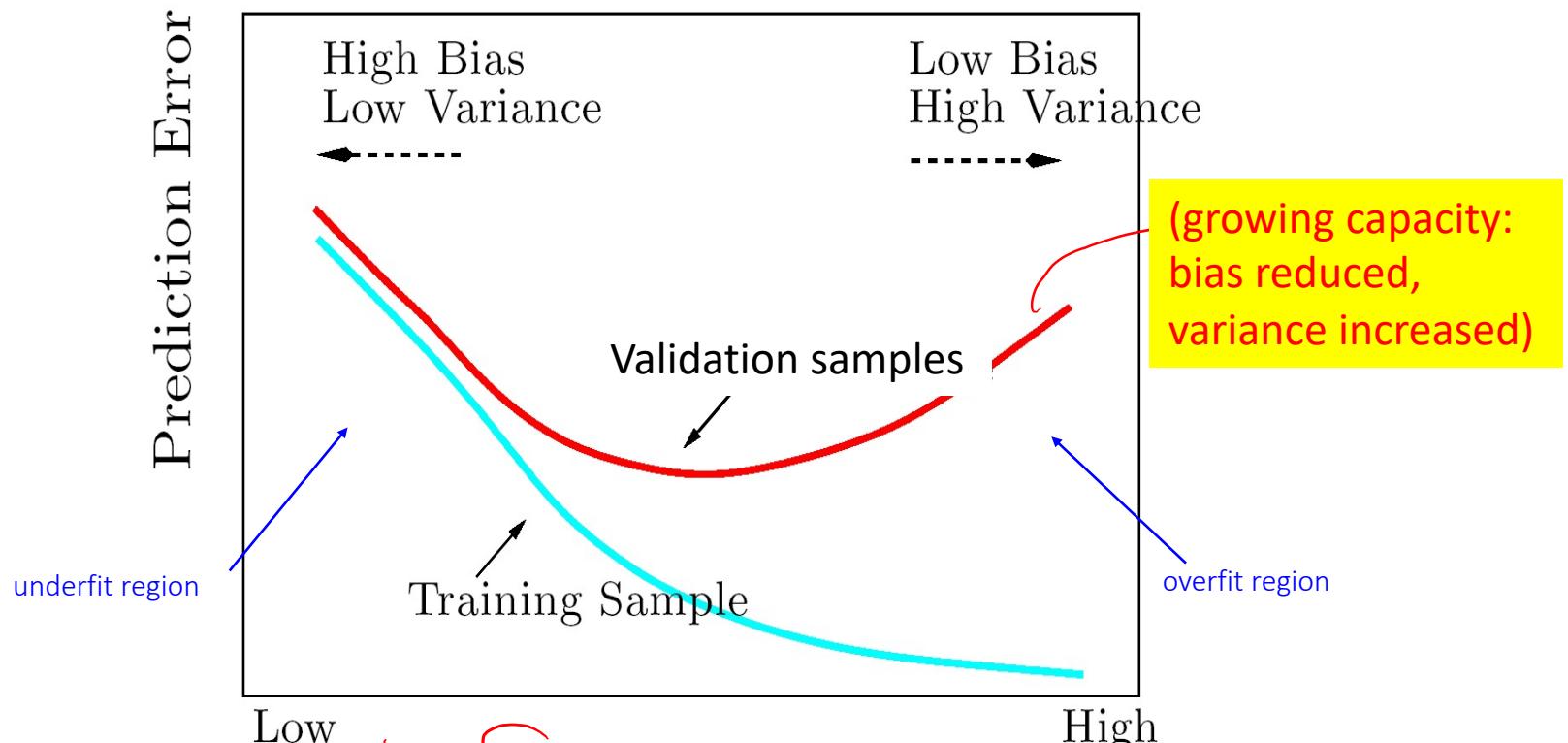


(growing capacity:  
bias reduced,  
variance increased)

# Bias-Variance Tradeoff / Model Selection

Rigid  $\xrightarrow{\hspace{10cm}}$  Complex

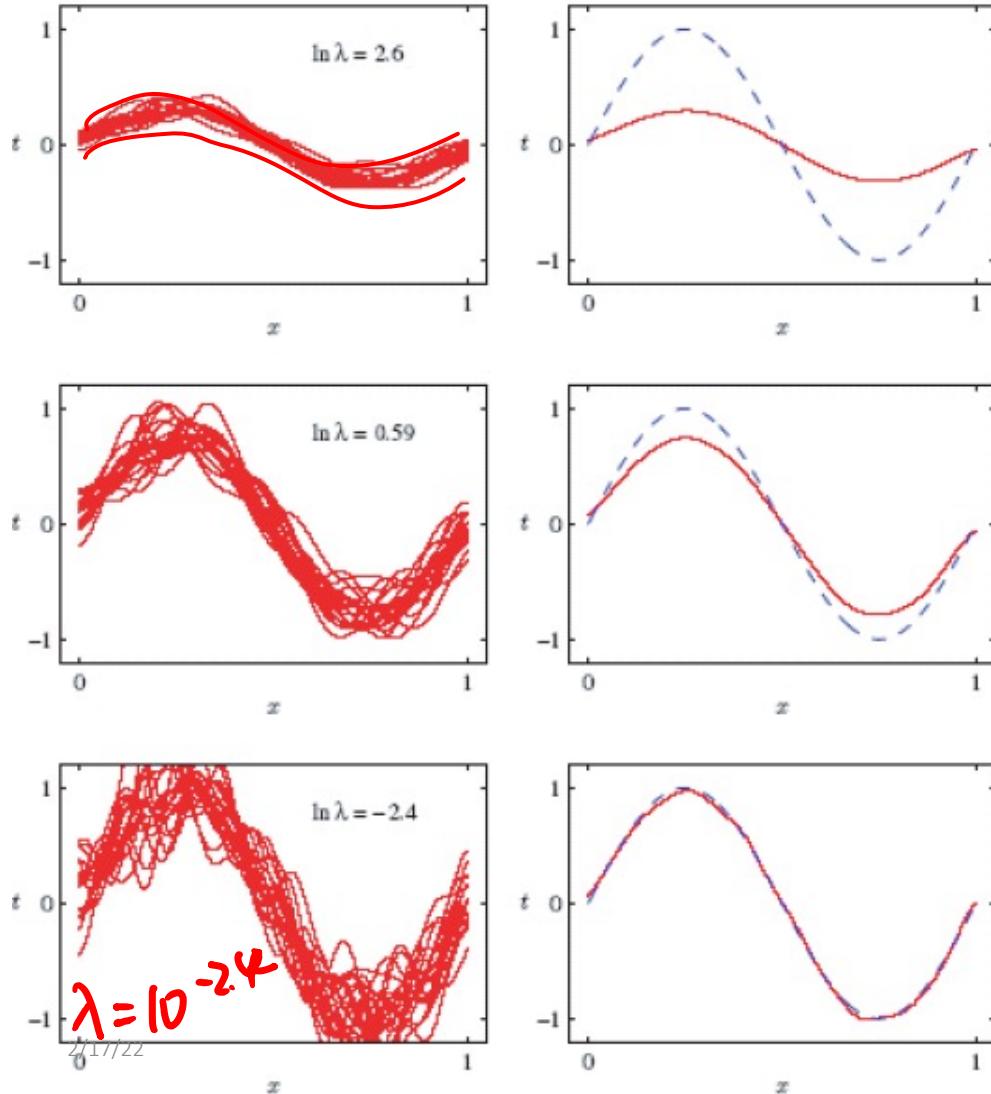
$K \leftarrow \rightarrow$   
 $d \uparrow \rightarrow$   
 $\lambda \downarrow \rightarrow$



KNN: large  $k \leftarrow$  [Model Complexity]  $\rightarrow$  Small  $k$   
Regression: small  $d \xrightarrow{35} \rightarrow$  large  $d$

### (3) Randomness in the Testing Error!!!

$D_{ts} \rightarrow E[D_{ts}]$



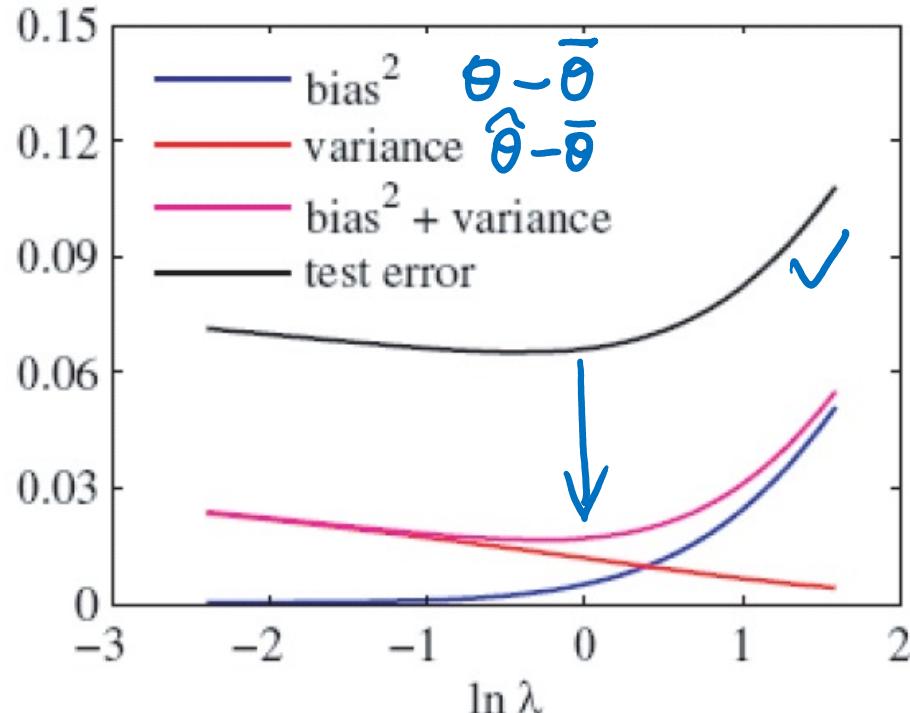
- lambda hyperparameter controls "regularization" terms in RLR, the smaller the lambda, is the more complex the model (why?)
  - Simple (highly regularized) models have low variance but high bias.
  - Complex models have low bias but high variance.
- You are inspecting an empirical average over 100 training set.

e.g. Regularized LR as an example.

(growing capacity:  
bias reduced,  
variance increased)

# (4) Generalization Error as Bias<sup>2</sup>+variance / Model Selection? → Expected Testing Error

- bias decrease with model capacity,
- Variance increase with model capacity
- Sum of Bias<sup>2</sup>+Variance has a valley shape

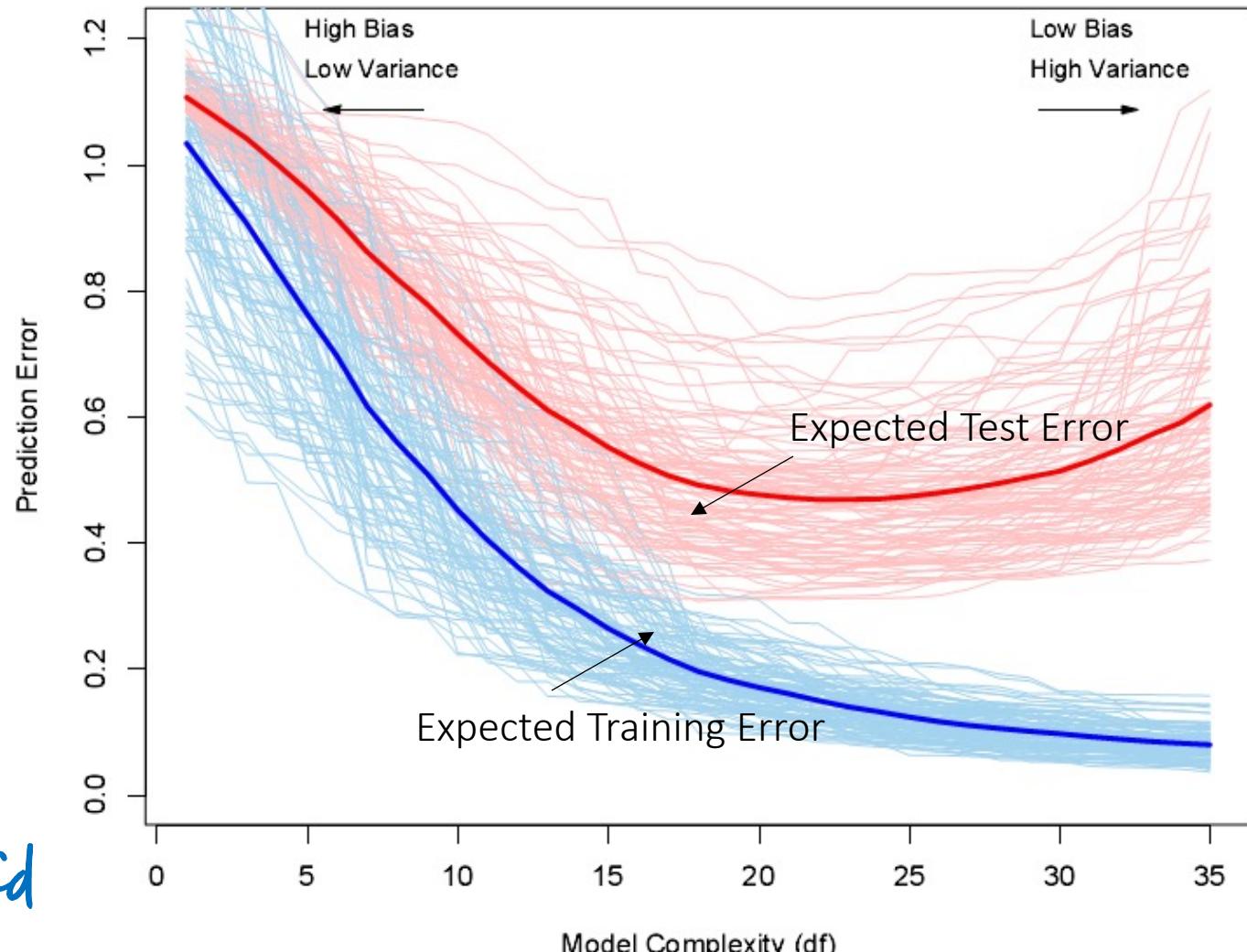


$p(x, y)$   
 $\theta_{\text{true}}$  :  
unknown

- Bias<sup>2</sup>+variance predicts (shape of) test error quite well.
- However, bias and variance cannot be computed since it relies on knowing the true distribution of x and y

# Cross Validation Error as good approximation for Expected Test error → good appx of generalization

See proof  
proving  
Cross Valid  
error as good  
approximates  
for Expected  
Test Error  
  
in Extra



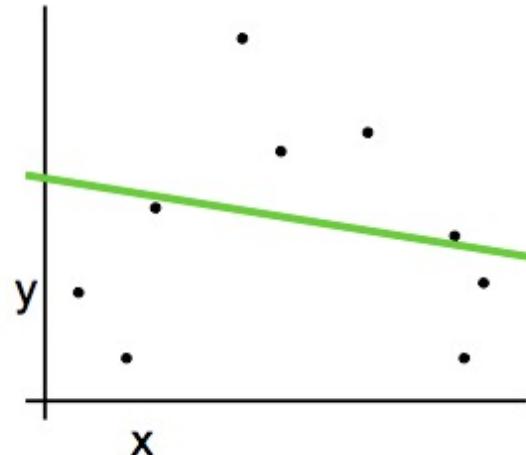
*Cross Valid*

$\approx EPE$

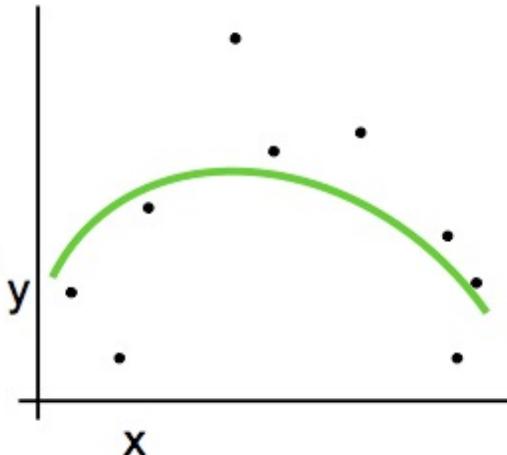
# Bias-Variance Trade-off

- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).
- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample randomness).

# Regression: Complexity versus Goodness of Fit



Low Variance /  
High Bias



Medium Bias  
Medium Variance  
Model complexity = medium

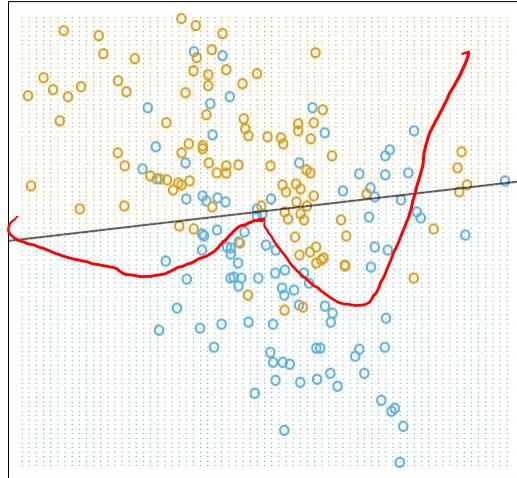


Low Bias  
/ High Variance

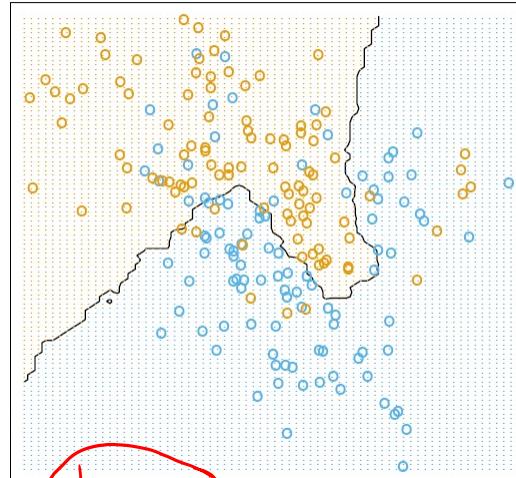
Highest Bias  
Lowest variance  
Model complexity = low

Smallest Bias  
Highest variance  
Model complexity = high

# Classification, Decision boundaries in global vs. local models



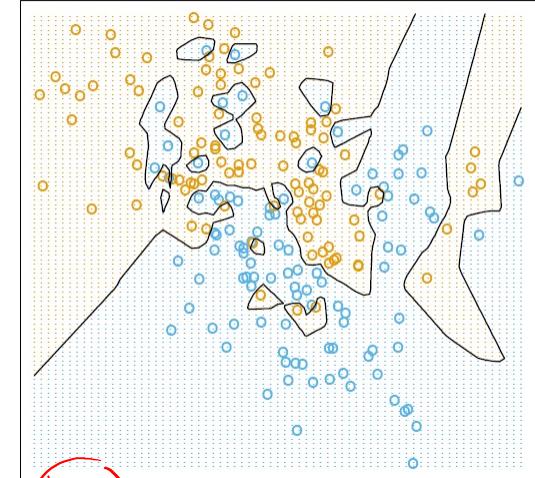
Low Variance /  
High Bias



$K=15$   
15-nearest neighbor

Highest Bias  
Lowest variance  
Model complexity = low

Medium Bias  
Medium Variance  
Model complexity = medium



$K=1$   
1-nearest neighbor

Low Bias / High Variance

Smallest Bias  
Highest variance  
Model complexity = high

# Thank You



# UVA CS 4774: Machine Learning

## Lecture 9: Bias-Variance Tradeoff

Module III

Dr. Yanjun Qi

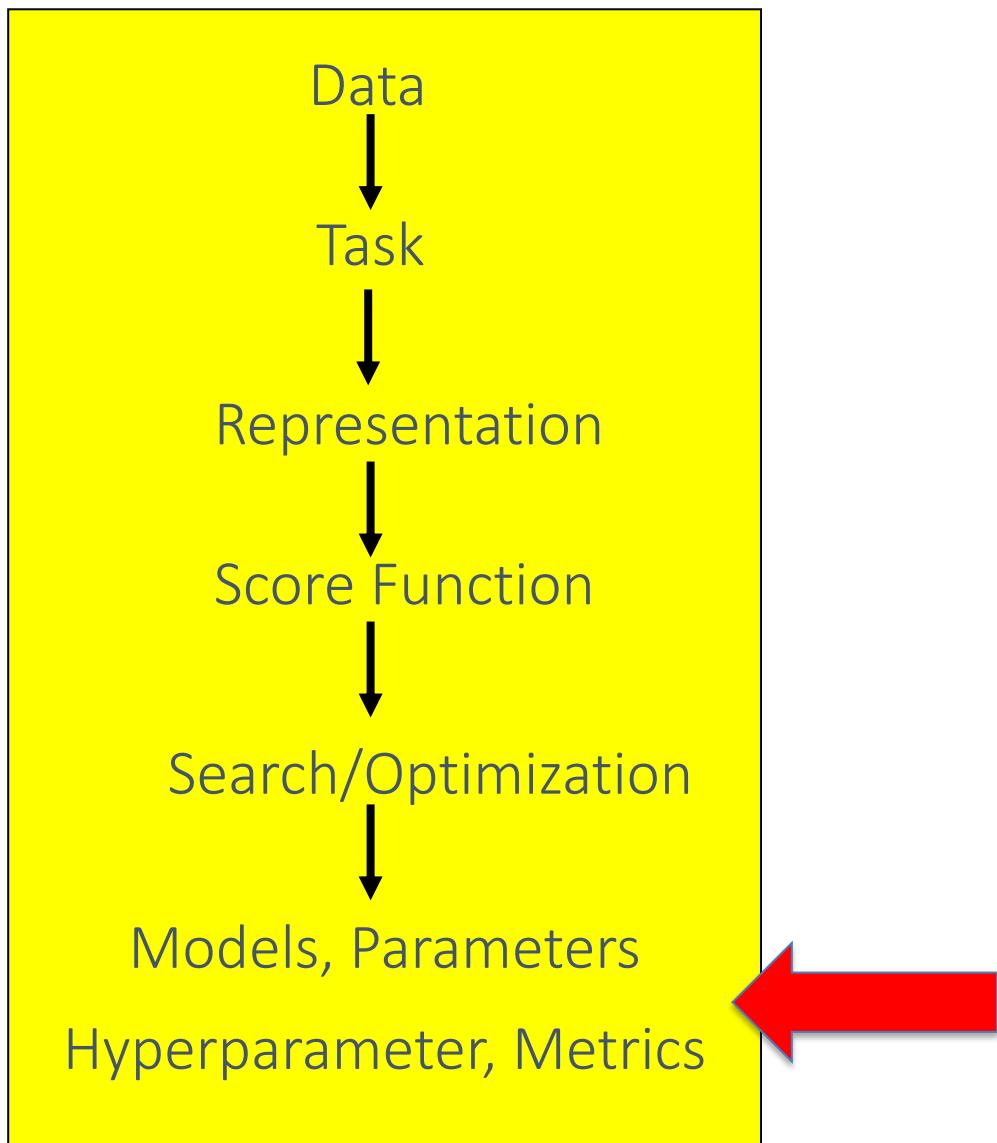
University of Virginia  
Department of Computer Science

# Roadmap

- Bias-variance decomposition
- Bias-Variance Tradeoff / Model Selection
- Remedy when Overfit / Underfit

# Machine Learning in a Nutshell

ML grew out of work in AI  
Optimize a performance criterion using example data or past experience, Aiming to generalize to unseen data



Review:

Expected Test Error as Bias<sup>2</sup>+variance + Bayes Error

$$Pr(x, Y)$$

$$EPE(x) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

Unavoidable  
error

Error due to  
incorrect  
assumptions

Error due to variance  
of training samples

$$(\theta, \bar{\theta})$$

$$(\bar{\theta}, \hat{\theta}_D)$$

# Why causes bad generalization?

- Components
  - Bias: how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - Variance: how much models estimated from different training sets differ from each other

# Two Types of bad generalization

- Underfitting: model is too “simple” to represent all the relevant characteristics

- High bias] and low variance

Bias

- High training error and high test error

- Overfitting: model is too “complex” and fits irrelevant characteristics (noise) in the data

- Low bias and high variance

Variance

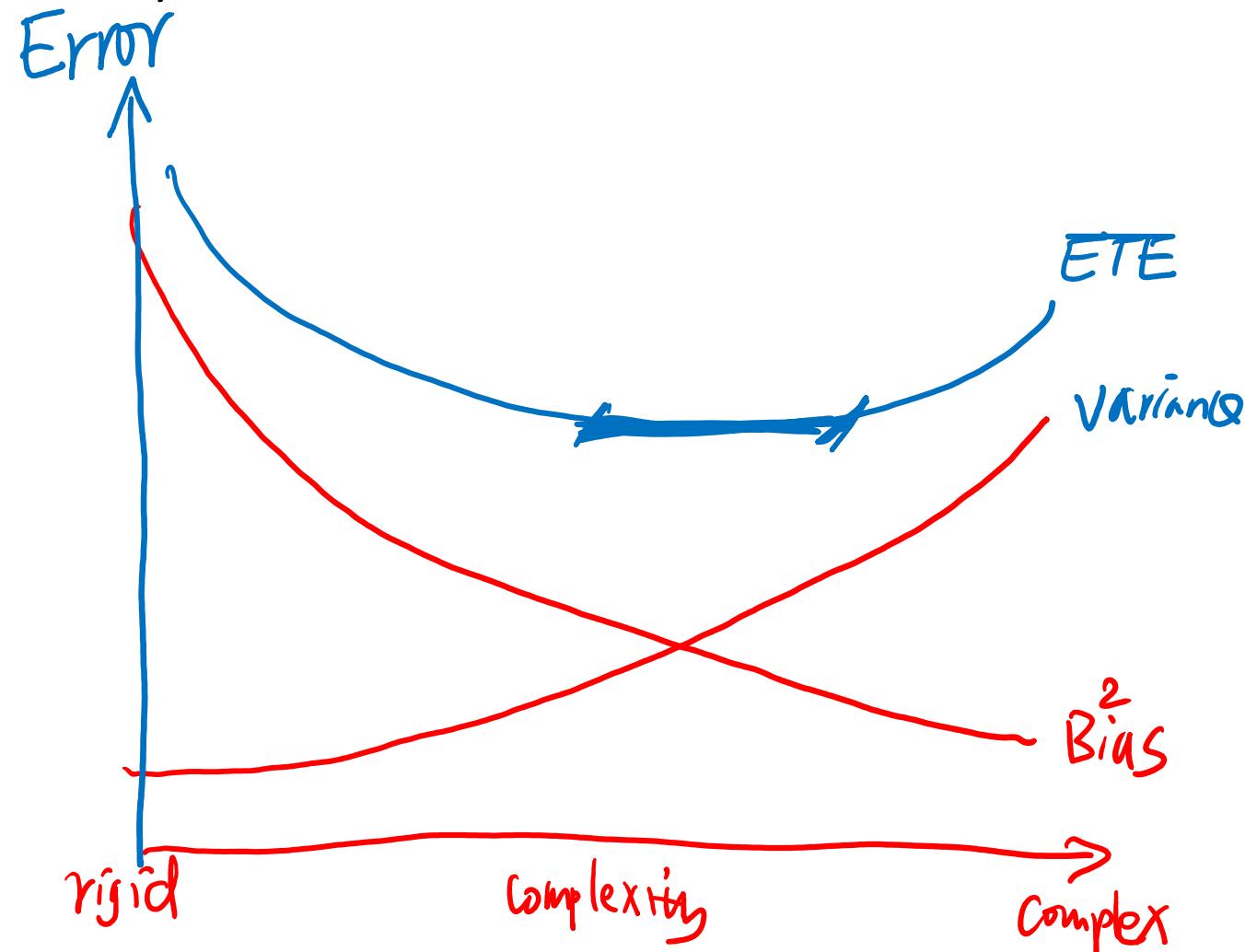
- Low training error and high test error

Review:

One important Control of Bias Variance Tradeoff

→ Model Complexity

- bias decrease with model gets more complex;
- Variance increase with bigger model capacity
- Sum of  $\text{Bias}^2 + \text{Variance}$

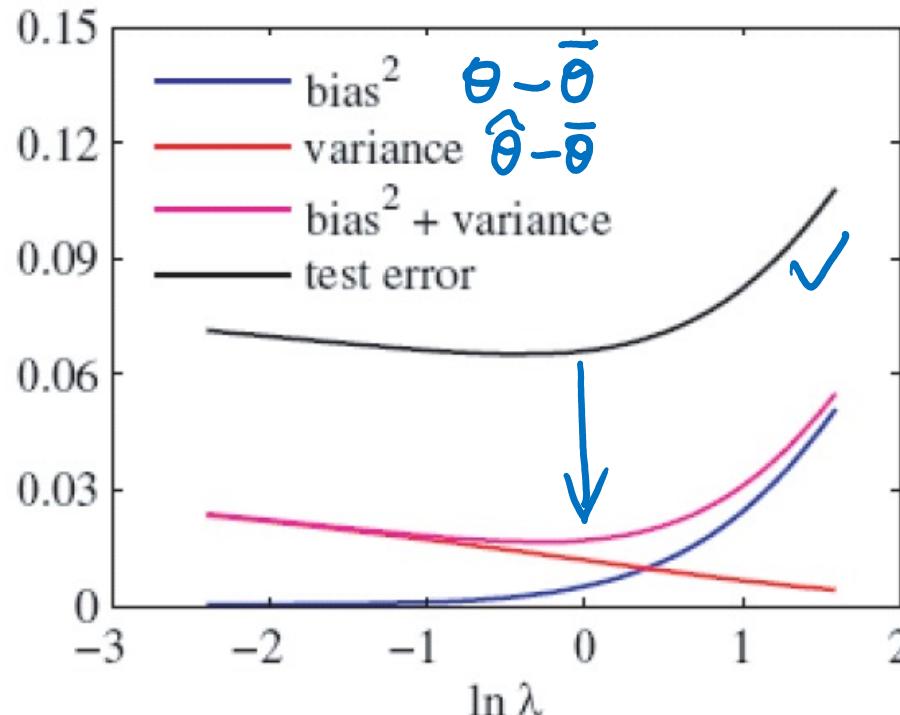


## Review:

One important Control of Bias Variance Tradeoff

→ Model Complexity

- bias decrease with model capacity,
- Variance increase with model capacity
- Sum of Bias<sup>2</sup>+Variance has a valley shape



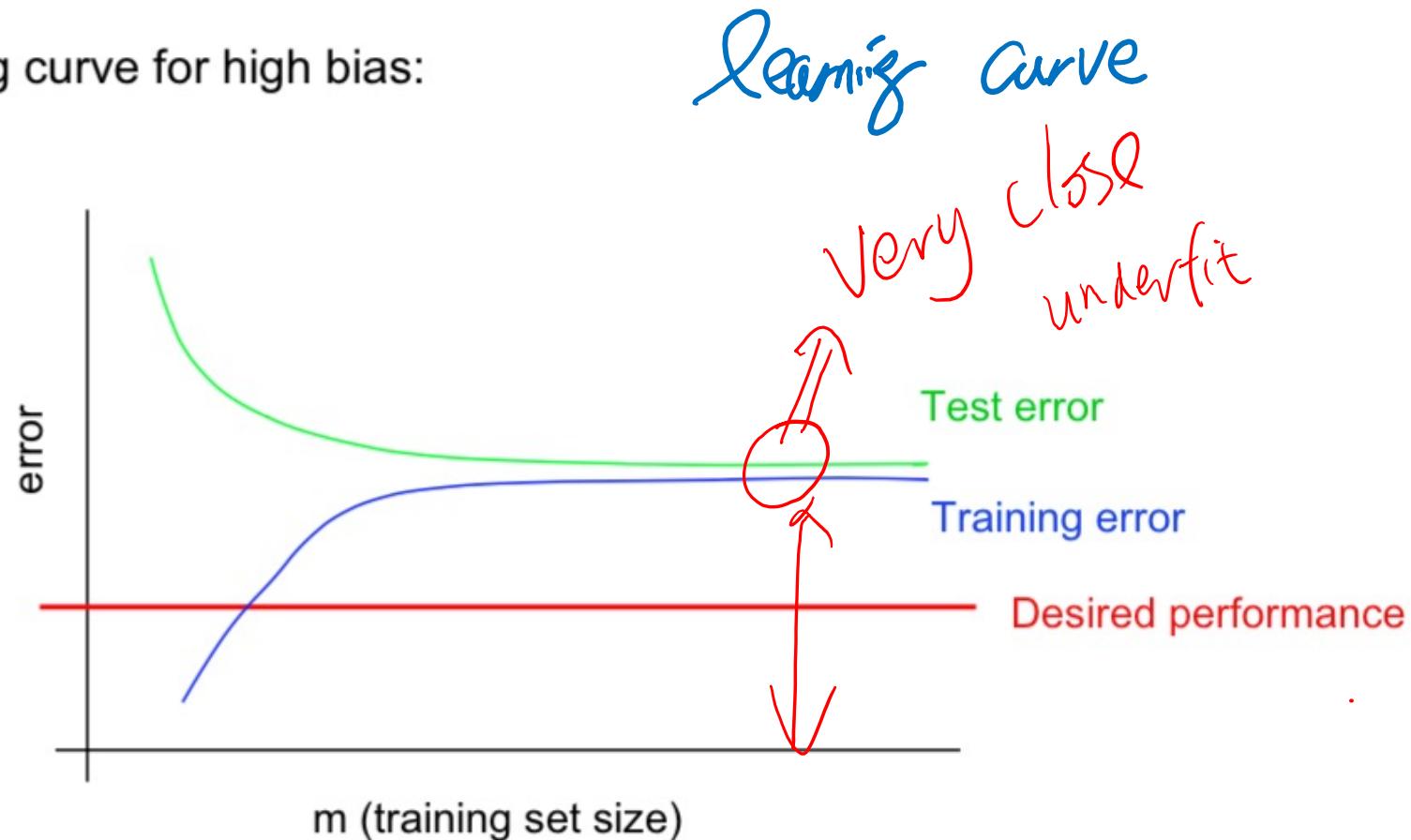
$p(x, y)$   
 $f(x)$   
 $\theta_{\text{true}}$ :  
unknown

- Bias<sup>2</sup>+variance predicts (shape of) test error quite well.
- However, bias and variance cannot be computed since it relies on knowing the true distribution of x and y

# Another important Control of Bias Variance Tradeoff

→ Training Size (Extra)

Typical learning curve for high bias:



- Even training error is unacceptably high.
- Small gap between training and test error.

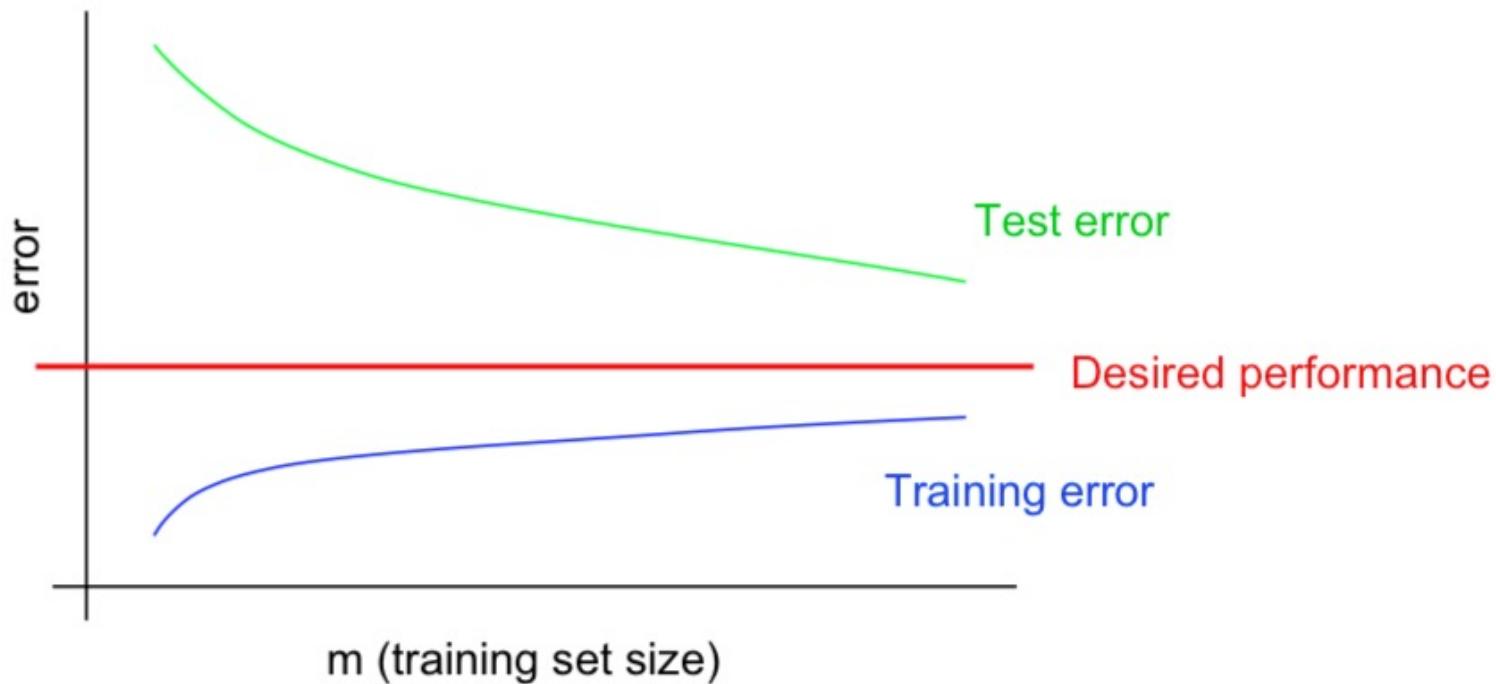
High training error and high test error

# Another important Control of Bias Variance Tradeoff

→ Training Size (Extra)

Typical learning curve for high variance:

*Learning Curve*



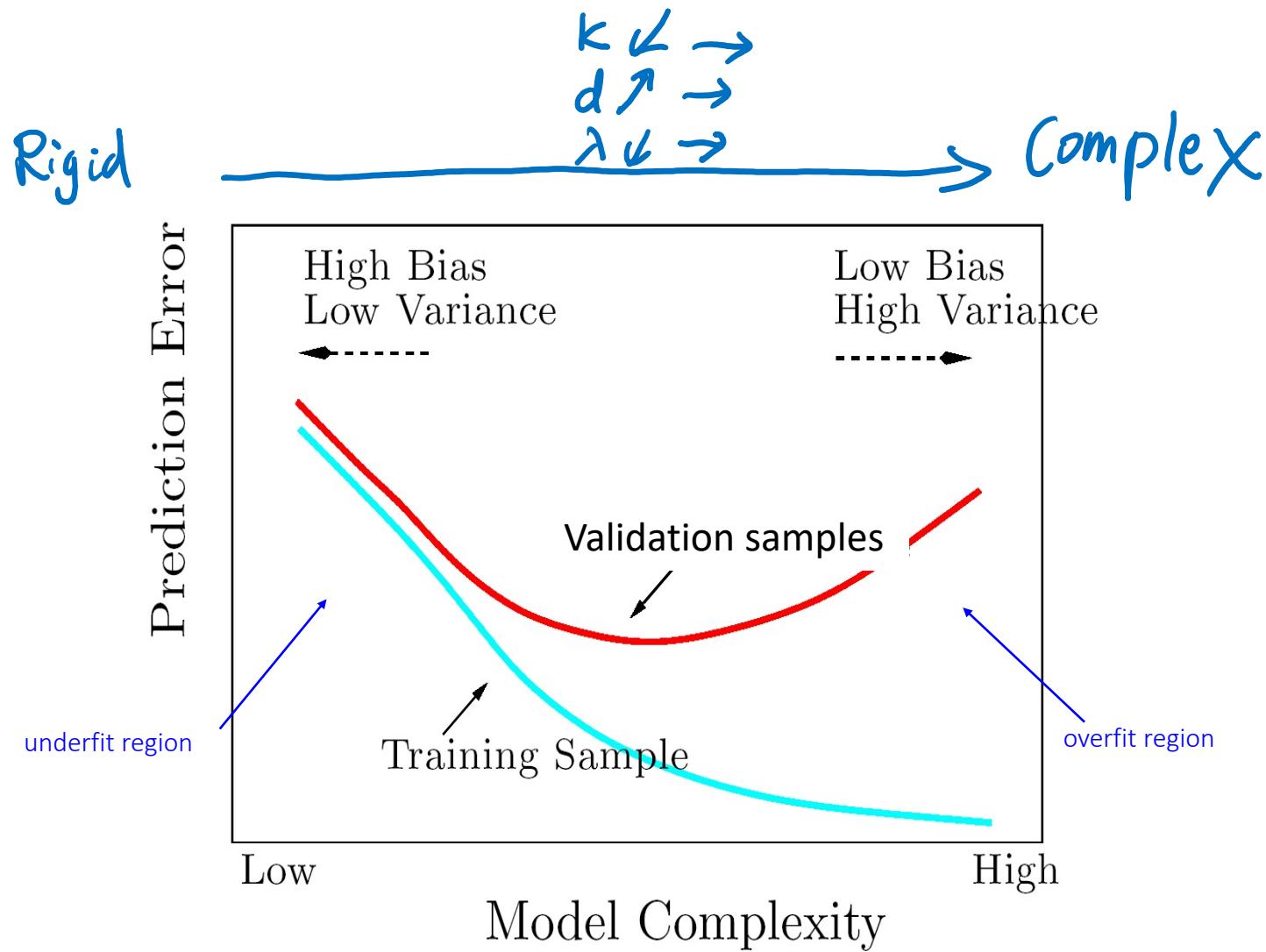
# Is the bias-variance trade off dependent on the number of samples? (EXTRA)

$n \nearrow$   
variance  $\downarrow$

In the usual application of linear regression, your coefficient estimators are unbiased so sample size is irrelevant. But more generally, you can have bias that is a function of sample size as in the case of the variance estimator obtained from applying the population variance formula to a sample (sum of squares divided by n).....

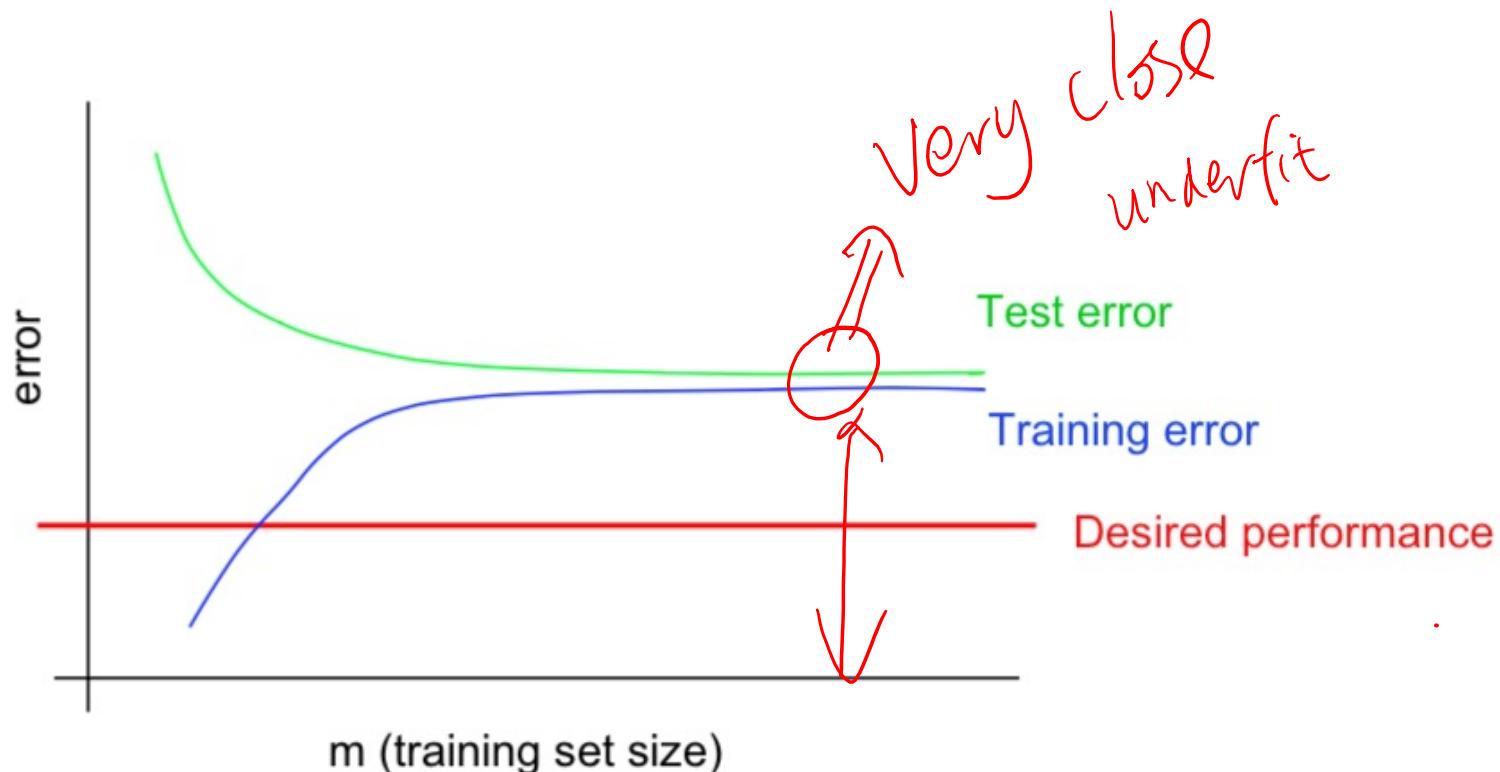
... the bias and variance for an estimator are generally a decreasing function of training size  $n$ . Dealing with this is a core topic in nonparametric statistics. For nonparametric methods with tuning parameters a very standard practice is to theoretically derive rates of convergence (as sample size goes to infinity) of the bias and variance as a function of the tuning parameter, and then you find the optimal (in terms of MSE) rate of convergence of the tuning parameter by balancing the rates of the bias and variance. Then you get asymptotic results of your estimator with the tuning parameter converging at that particular rate. Ideally you also provide a data-based method of choosing the tuning parameter (since simply setting the tuning parameter to some fixed function of sample size could have poor finite sample performance), and then show that the tuning parameter chosen this way attains the optimal rate.

(0) If your model complexity can be ordered as X-axis



# (1) If you use learning curve find: Underfitting / High bias / Model too Simple

Typical learning curve for high bias:



- Even training error is unacceptably high.
- Small gap between training and test error.

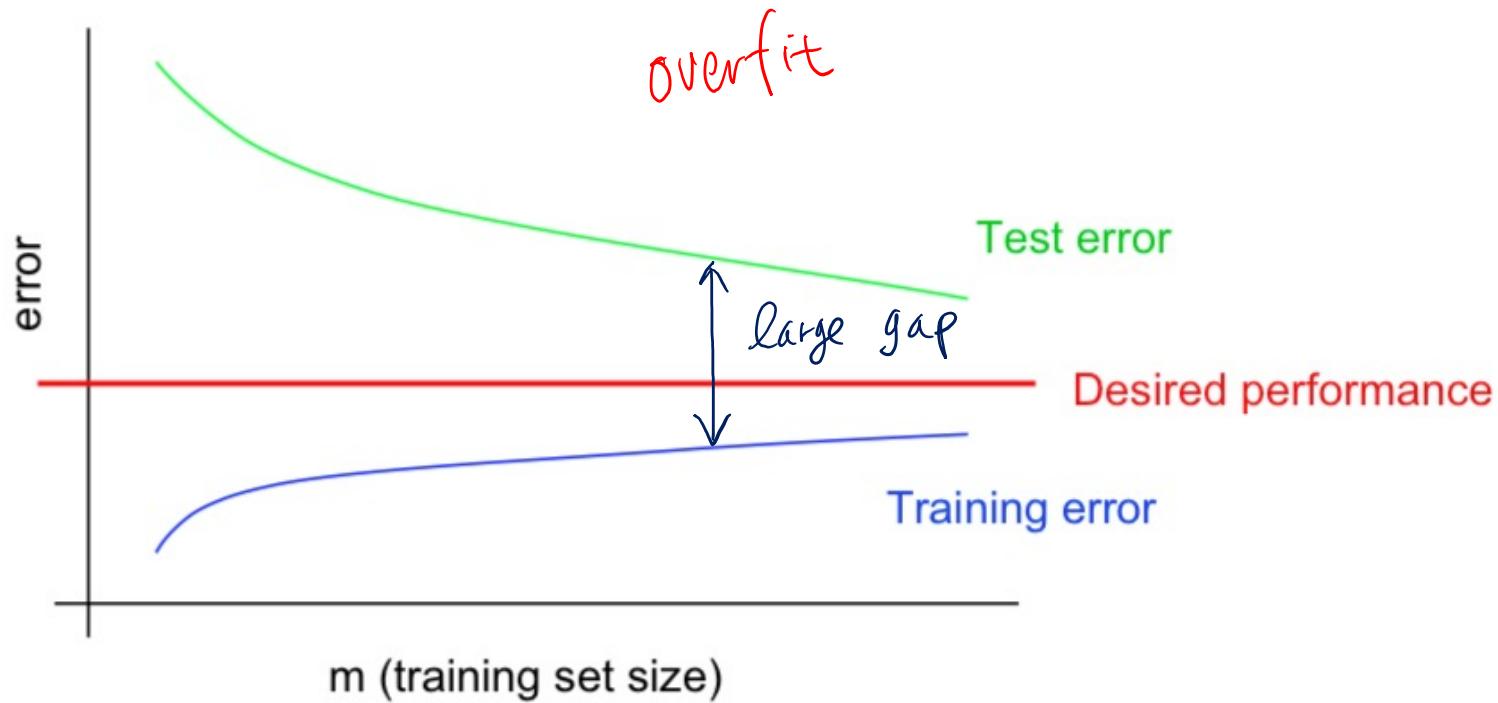
High training error and high test error

# How to reduce Model High Bias ?

- E.g.
  - Get additional features
  - Try more complex learner

(2) If you use learning curve find: Overfitting / High variance / Model too Complex

Typical learning curve for high variance:



- Test error still decreasing as  $m$  increases. Suggests larger training set will help.
- Large gap between training and test error.
- Low training error and high test error

# How to reduce Model High Variance?

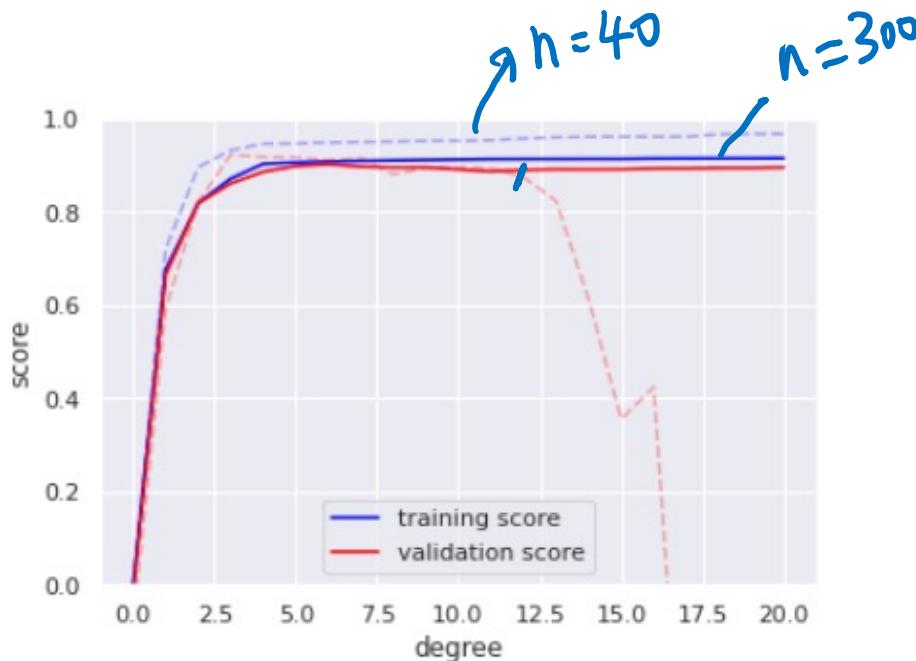
- Choose a simpler classifier
- Regularize the parameters
- Get more training data
- Try smaller set of features *feature Selection*  $n < p$
- Try feature engineering
- Try multiple models and then use all as ensemble

# Take Away : Three types of plots

- (1) Sanity check (S)GD type Optimization
  - Train / Vali Loss vs. Epochs to help you
  - [https://scikit-learn.org/stable/auto\\_examples/linear\\_model/plot\\_sgd\\_early\\_stopping.html#sphx-glr-auto-examples-linear-model-plot-sgd-early-stopping-py](https://scikit-learn.org/stable/auto_examples/linear_model/plot_sgd_early_stopping.html#sphx-glr-auto-examples-linear-model-plot-sgd-early-stopping-py)
- (2) Sanity check hyperparameter tuning
  - Train / Vali Loss vs. hyperparameter Values  
**from sklearn.model\_selection import validation\_curve**
- (3) Sanity check if your current model overfits or underfits
  - Train / Vali Loss vs. Varying Size of Training
  - [https://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_learning\\_curve.html#sphx-glr-auto-examples-model-selection-plot-learning-curve-py](https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html#sphx-glr-auto-examples-model-selection-plot-learning-curve-py)

I will Code run

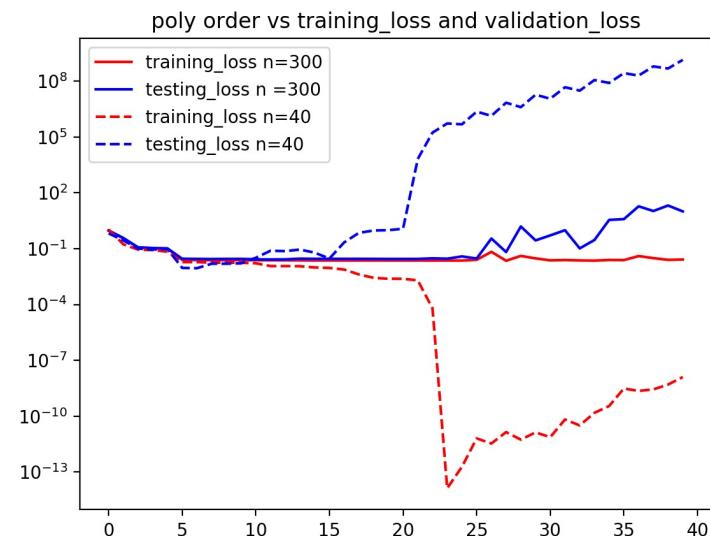
<https://colab.research.google.com/drive/1TvHQoJpYwc5XKz0QGOY3yu1OTIjSY5gn?usp=sharing>



### (1) Validation\_curve

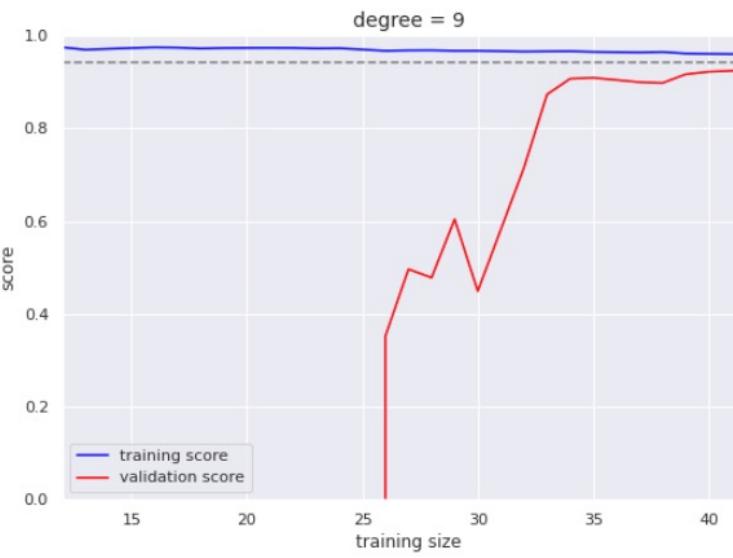
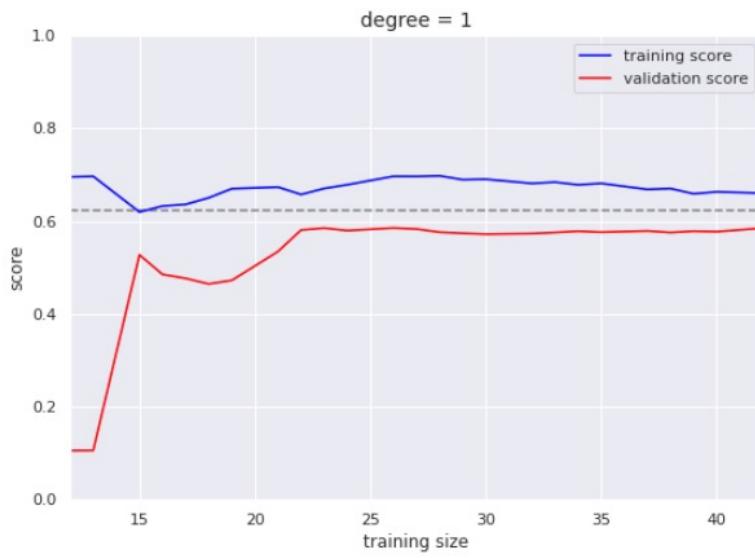
By scikitlearn Validation\_curve function  
(normalize all metrics to positive range)

[https://scikit-learn.org/stable/modules/model\\_evaluation.html#the-scoring-parameter-defining-model-evaluation-rules](https://scikit-learn.org/stable/modules/model_evaluation.html#the-scoring-parameter-defining-model-evaluation-rules)



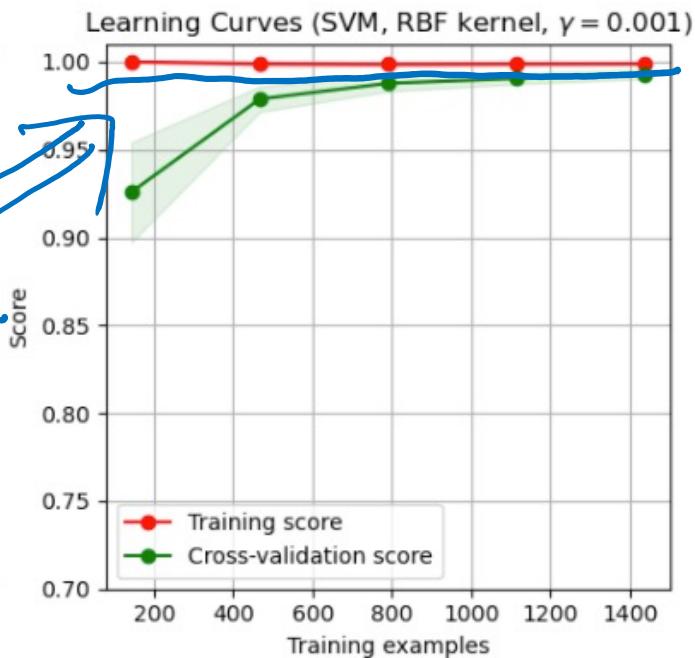
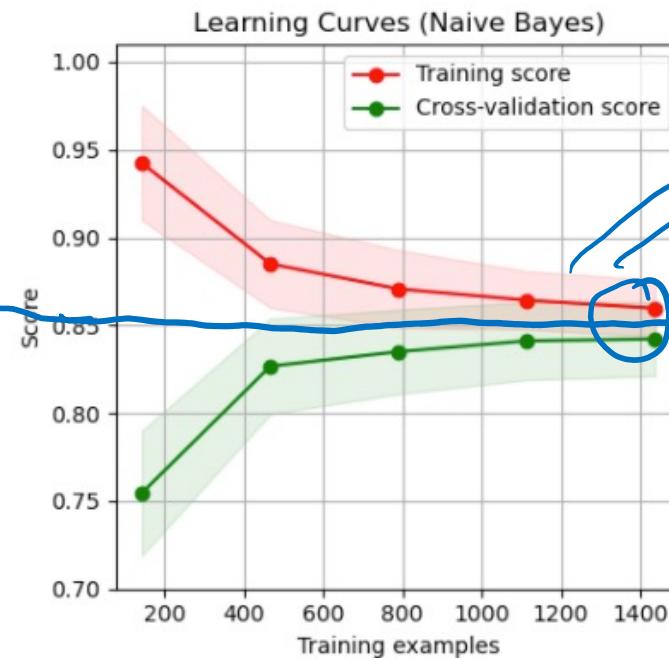
### (1) Validation\_curve

By our HW2 ( more close to modern deep learning library style )



## (1) Learning Curves for polynomial regression (up) and classification (down) / by scikitlearn

underfit



# References

- Prof. Tan, Steinbach, Kumar's "Introduction to Data Mining" slide
- Prof. Andrew Moore's slides
- Prof. Eric Xing's slides
- Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.

	<b>Underfitting</b>	<b>Just right</b>	<b>Overfitting</b>
<b>Symptoms</b>	<ul style="list-style-type: none"> <li>- High training error</li> <li>- Training error close to test error</li> <li>- High bias</li> </ul>	<ul style="list-style-type: none"> <li>- Training error slightly lower than test error</li> </ul>	<ul style="list-style-type: none"> <li>- Low training error</li> <li>- Training error much lower than test error</li> <li>- High variance</li> </ul>
<b>Regression</b>			
<b>Classification</b>			
<b>Remedies</b>	<ul style="list-style-type: none"> <li>- Complexify model</li> <li>- Add more features</li> <li>- Train longer</li> </ul>		<ul style="list-style-type: none"> <li>- Regularize</li> <li>- Get more data</li> <li>- Feature selection</li> </ul>

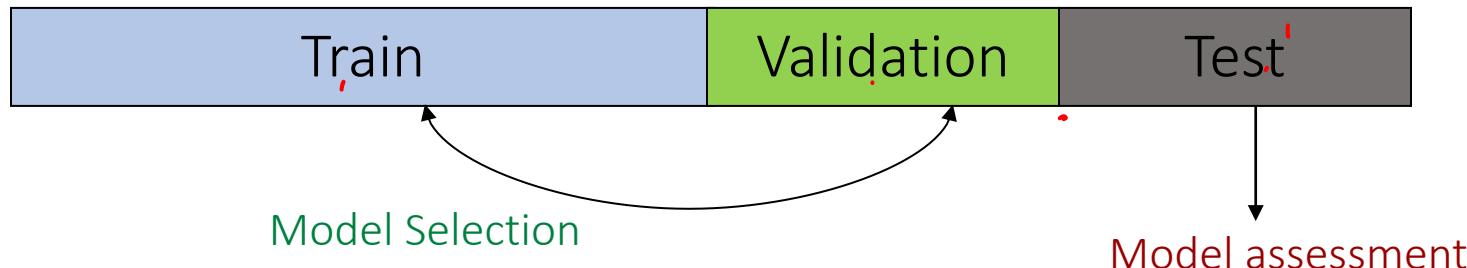
Credit: Stanford Machine Learning

# Review: Model Selection and Assessment

- Model Selection
  - Estimating performances of different models to choose the best one
- Model Assessment
  - Having chosen a model, estimating the prediction error on new data

# Model Selection and Assessment

- When Data Rich Scenario: Split the dataset



- When Insufficient data to split into 3 parts
  - Approximate validation step analytically
    - AIC, BIC, MDL, SRM
  - **Efficient reuse of samples**
    - Cross validation, bootstrap

# Model Selection (Hyperparameter Tuning) & Model Assessment Pipelines in HW2

- (1) train / Validation / test
- (2) k-CV on train to choose hyperparameter / then test

# The battle against overfitting (Extra) :

- Cross validation
- Regularization
- Feature selection
- Model selection --- Occam's razor
- Model averaging
  - The Bayesian-frequentist debate
  - Bayesian learning (weight models by their posterior probabilities)

For instance, if trying to solve “spam detection” using (Extra)

L2 - logistic regression, implemented with gradient descent.

Fixes to try: If performance is not as desired

- Try getting more training examples.
- Try a smaller set of features.
- Try a larger set of features.
- Try email header features.
- Run gradient descent for more iterations.
- Try Newton’s method.
- Use a different value for  $\lambda$ .
- Try using an SVM.

Fixes high variance.  
Fixes high variance.  
Fixes high bias.  
Fixes high bias.  
Fixes optimization algorithm.  
Fixes optimization algorithm.  
Fixes optimization objective.  
Fixes optimization objective.

# kNN estimator and L2-EPE

# Expected prediction error (EPE)

Consider joint distribution

$$\text{EPE}(f) = \mathbb{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

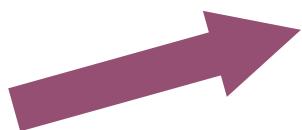
- For L2 loss:

$$\text{e.g.} = \int (y - f(x))^2 \Pr(dx, dy)$$

under L2 loss, best estimator for EPE (Theoretically) is :

Conditional mean  $\hat{f}(x) = \mathbb{E}(Y | X = x)$

e.g. KNN



NN methods are the direct implementation (approximation )

# kNN for minimizing EPE

- We know under L2 loss, best estimator for minimize EPE (theoretically) is :

Conditional

mean  $f(x) = \mathbb{E}(Y | X = x)$

- Nearest neighbours assumes that  $f(x)$  is well approximated by a locally constant function.

# Minimize EPE using L2

- Expected prediction error (EPE) for L2 Loss:

$$\text{EPE}(f) = \mathbb{E}(Y - f(X))^2 = \int (y - f(x))^2 \Pr(dx, dy)$$

- Since  $\Pr(X, Y) = \Pr(Y | X)\Pr(X)$ , EPE can also be written as

$$\text{EPE}(f) = \mathbb{E}_X \mathbb{E}_{Y|X}([Y - f(X)]^2 | X)$$

- Thus it suffices to minimize EPE pointwise

Best estimator under L2 loss:  
conditional expectation

$$f(x) = \arg \min_c \mathbb{E}_{Y|X}([Y - c]^2 | X = x)$$

Conditional

mean

Solution for Regression:

2/17/22

Solution for kNN:



$$f(x) = \mathbb{E}(Y | X = x)$$

# Minimize EPE using L2 (another proof)

- Let  $t$  be the **true** (target) output and  $y(x)$  be our estimate. The **expected squared loss** is

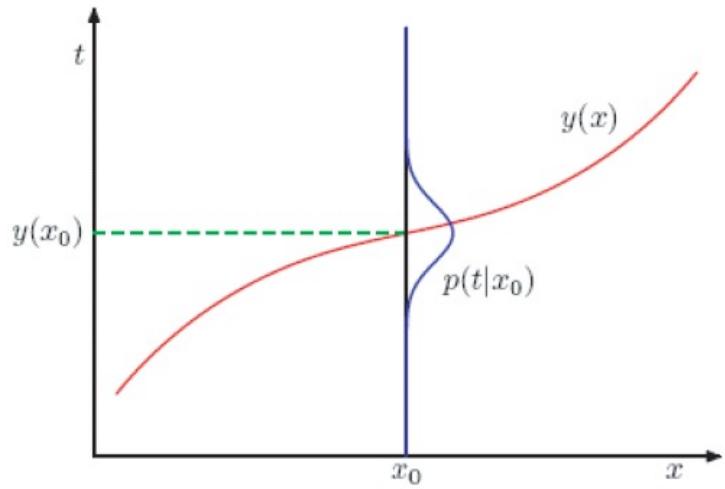
$$\begin{aligned}E(L) &= \iint L(t, y(x)) p(x, t) dx dt \\&= \iint (t - y(x))^2 p(x, t) dx dt\end{aligned}$$

- Our goal is to choose  $y(x)$  that minimizes  $E(L)$ :
  - Calculus of variations:

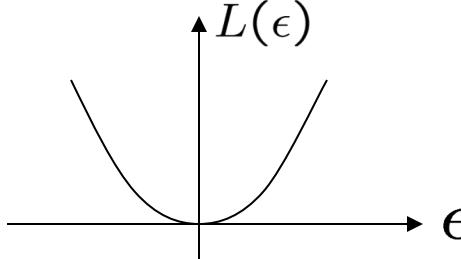
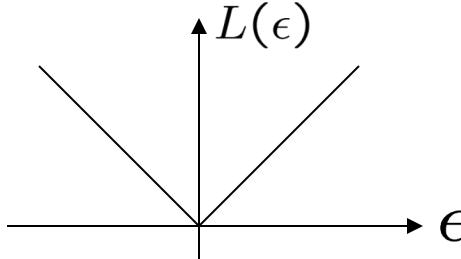
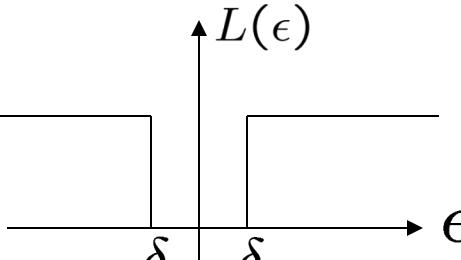
$$\frac{\partial E(L)}{\partial y(x)} = 2 \int (t - y(x)) p(x, t) dt = 0$$

$$\int y(x) p(x, t) dt = \int t p(x, t) dt$$

$$y^*(x) = \int \frac{tp(x, t)}{p(x)} dt = \int tp(t | x) dt = E_{t|x}[t] = E[t | x]$$



# Review : EPE with different loss

Loss Function	Estimator $\hat{f}(x)$
$L_2$  $L(\epsilon) = \epsilon^2$	$\hat{f}(x) = E[Y X = x]$
$L_1$  $L(\epsilon) =  \epsilon $	$\hat{f}(x) = \text{median}(Y X = x)$
$0-1$  $L(\epsilon) = \begin{cases} 0 & \text{if } \epsilon < -\delta \\ 1 & \text{if } -\delta \leq \epsilon \leq \delta \\ 0 & \text{if } \epsilon > \delta \end{cases}$	$\hat{f}(x) = \arg \max_Y P(Y X = x)$ (Bayes classifier / MAP)

# Expected prediction error (EPE)

Consider joint distribution

$$\text{EPE}(f) = \mathbb{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

For 0-1 loss:  $L(k, \ell) = 1 - d_{kl}$

Bayes Classifier



$$\hat{f}(X) = C_k \text{ if } \Pr(C_k | X = x) = \max_{g \in \mathcal{C}} \Pr(g | X = x)$$

# More for Overfitting

# Bayesian and Frequentist (Extra)

- Frequentist interpretation of probability
  - Probabilities are objective properties of the real world, and refer to limiting relative frequencies (e.g., number of times I have observed heads). Hence one cannot write  $P(\text{Katrina could have been prevented}/D)$ , since the event will never repeat.
  - Parameters of models are *fixed, unknown constants*. Hence one cannot write  $P(\vartheta/D)$  since  $\vartheta$  does not have a probability distribution. Instead one can only write  $P(D/\vartheta)$ .
  - One computes point estimates of parameters using various *estimators*,  $\vartheta^* = f(D)$ , which are designed to have various desirable qualities when *averaged over future data D* (assumed to be drawn from the “true” distribution).
- Bayesian interpretation of probability
  - Probability describes degrees of belief, not limiting frequencies.
  - Parameters of models are *hidden variables*, so one can compute  $P(\vartheta/D)$  or  $P(f(\vartheta)/D)$  for some function  $f$ .
  - One estimates parameters by computing  $P(\vartheta/D)$  using Bayes rule:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

# Cross Validation and Variance Estimation

- Cross-validation (CV) is quite a general tool for estimating the expected test error (1), that makes minimal assumptions—i.e., it doesn't assume that  $Y = f(X) + \varepsilon$  with  $\varepsilon$  independent of  $X$ , it doesn't assume that the training inputs  $x_1, \dots, x_n$  are fixed, all it really assumes is that the training samples  $(x_1, y_1), \dots, (x_n, y_n)$  are i.i.d.

We split up our training set into  $K$  divisions or folds, for some number  $K$ ; usually this is done randomly. Write these as  $F_1, \dots, F_K$ , so  $F_1 \cup \dots \cup F_K = \{1, \dots, n\}$ . Now for each  $k = 1, \dots, K$ , we fit our prediction function on all points but those in the  $k$ th fold, denoted  $\hat{f}^{-(k)}$ , and evaluate squared errors on the points in the  $k$ th fold,

$$\text{CV}_k(\hat{f}^{-(k)}) = \frac{1}{n_k} \sum_{i \in F_k} (y_i - \hat{f}^{-(k)}(x_i))^2.$$

<http://www.stat.cmu.edu/~ryantibs/statml/review/modelbasics.pdf>

- What is the difference between choosing say  $K = 5$  (a common choice) versus  $K = n$ ?
  - When  $K = 5$ , the function  $\hat{f}^{-(k)}$  in each fold  $k$  is fit on about  $4/5 \cdot n$  samples, and so we are looking at the errors incurred by a procedure that is trained on less data than the full  $\hat{f}$  in (1). Therefore the mean of the CV estimate (7) could be off. When  $K = n$ , this is not really an issue, since each  $\hat{f}^{-(k)}$  is trained on  $n - 1$  samples
  - When  $K = n$ , the CV estimate (7) is an average of  $n$  extremely correlated quantities; this is because each  $\hat{f}^{-(k)}$  and  $\hat{f}^{-(\ell)}$  are fit on  $n - 2$  common training points. Hence the CV estimate will likely have very high variance. When  $K = 5$ , the CV estimate will have lower variance, since it is the average of quantities that are less correlated, as the fits  $\hat{f}^{-(k)}$ ,  $k = 1, \dots, 5$  do not share as much overlapping training data

This is tradeoff (the bias-variance tradeoff, in fact!). Usually, a choice like  $K = 5$  or  $K = 10$  is more common in practice than  $K = n$ , but this is probably an issue of debate

- For  $K$ -fold CV, it's can be helpful to assign a notion of variability to the CV error estimate. We argue that

$$\text{Var}(\text{CV}(\hat{f})) = \text{Var}\left(\frac{1}{K} \sum_{k=1}^K \text{CV}_k(\hat{f}^{-(k)})\right) \approx \frac{1}{K} \text{Var}(\text{CV}_1(\hat{f}^{-(1)})). \quad (8)$$

Why is this an approximation? This would hold exactly if  $\text{CV}_1(\hat{f}^{-(1)}), \dots, \text{CV}_K(\hat{f}^{-(K)})$  were i.i.d., but they're not. This approximation is valid for small  $K$  (e.g.,  $K = 5$  or  $10$ ) but not really for big  $K$  (e.g.,  $K = n$ ), because then the quantities  $\text{CV}_1(\hat{f}^{-(1)}), \dots, \text{CV}_K(\hat{f}^{-(K)})$  are highly correlated

## Extra: Practical issues for Cross Validation

- How to decide the values for K in K-CV:
  - Also a bias-variance tradeoff issue
  - Commonly used  $K = 10$
  - when data sets are small relative to the number of models that are being evaluated, we need to increase K
  - K needs to be large for the variance to be small enough, but this makes it time-consuming.

# Practical issues for CV

- How to decide the values for K in KCV and  $a = 1/K$ 
  - Commonly used  $K = 10$  and  $a = 0.1$ .
  - when data sets are small relative to the number of models that are being evaluated, we need to decrease  $a$  and increase  $K$
  - $K$  needs to be large for the variance to be small enough, but this makes it time-consuming.
- Bias-variance trade-off
  - Small  $a$  usually lead to low bias. In principle, *LOOCV* provides an almost unbiased estimate of the generalization ability of a classifier, especially when the number of the available training samples is severely limited; but it can also have high variance.
  - Large  $a$  can reduce variance, but will lead to under-use of data, and causing high-bias.
- One important point is that the test data  $D_{\text{test}}$  is never used in CV, because doing so would result in overly (*indeed dishonest*) optimistic accuracy rates during the testing phase.