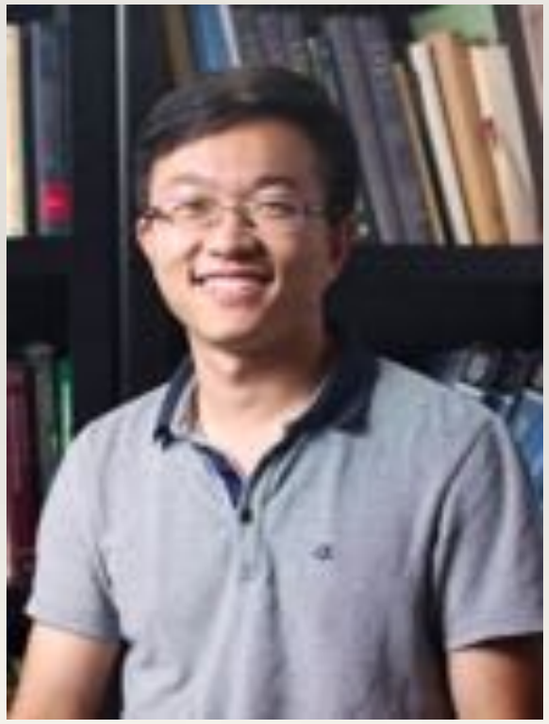
Practical Statistical Learning (F18)

Feng Liang



Changbo Zhu

Yinyin (Elaine) Chen

Joshua Daniel Loyal

Overview

- Types of statistical learning problems
- Why learning is difficult?
- Bias variance tradeoff
- An example: kNN *vs* Linear Regression (in a separate pdf file)
- Not all about prediction

Problems (I)

- Project 1 (Ames Housing Data): Predict the sale price of a house given its features.
- Project 2 (Sales Forecasting): Provide sales forecasting for Walmart for each department in each store based on historical data.

Y: Target
X: Feature / Covariates

Problems (II)

- Project 3 (Lending Club): Determine the chance that a borrower will miss a payment next month given various characteristics of the borrower and the loan.
- Project 4 (Sentiment Analysis): Determine whether a movie review is positive or negative.

Y: Target
X: Feature/Covariates

Problems (III)

- Based on the recent real estate transactions at Ames, Iowa, can we identify any home buying/selling trends? Further, can we identify distinctive groups of buyers?

- Based on the transaction data at Walmart, can we recommend any marketing strategies to Walmart?

Association Rule (chap 14.2 of ESL)
Market Segmentation (cluster customers)

Problems (III)

- Based on the recent real estate transactions at Ames, Iowa, can we identify any home buying/selling trends? Further, can we identify distinctive groups of buyers?
- Based on the transaction data at Walmart, can we recommend any marketing strategies to Walmart?

Y: ~~Target~~

X: Feature/Covariates

Types of Statistical Learning Problems

- Supervised Learning
  - Regression: response is a number
  - Classification: response is a label (binary or multi-class)

Semi-supervised Learning *Recommender System*

- Unsupervised Learning: identify latent structures in the data, e.g., clustering, association rule, HMM, etc.

How does Supervised Learning Work?

Data: $\{x\_i, y\_i\}$
i=1, 2, …, n

Goal: learn w
y <—- $f(x; w)$

Minimize an objective function

$$F(w) = \sum_{i=1}^{n} \text{loss}(y_i, f(x_i; w))$$

$$\text{loss} = \left[y_i - f(x_i; w) \right]^2$$

$$\text{loss} = \mathbf{1}\left\{ y_i \neq f(x_i; w) \right\}$$

$$\text{loss} = \begin{cases} -\log f(x_i; w) & \text{if } y_i = 1 \\ -\log(1 - f(x_i; w)) & \text{if } y_i = 0 \end{cases}$$

How does Supervised Learning Work?

Data: $\{x\_i, y\_i\}$
i=1, 2, …, n

Goal: learn w
y <—- $f(x; w)$

Minimize an objective function

$$F(w) = \sum_{i=1}^{n} \text{loss}(y_i, f(x_i; w))$$

1. The minimizer w^* may be in closed form.
2. Try optimization algorithms that can guarantee to converge to the global minimizer.
3. In the worst case, try *gradient descent*.

Overview

- Types of statistical learning problems
- Why learning is difficult?
- Bias variance tradeoff
- An example: kNN *vs* Linear Regression (in a separate pdf file)
- Not all about prediction

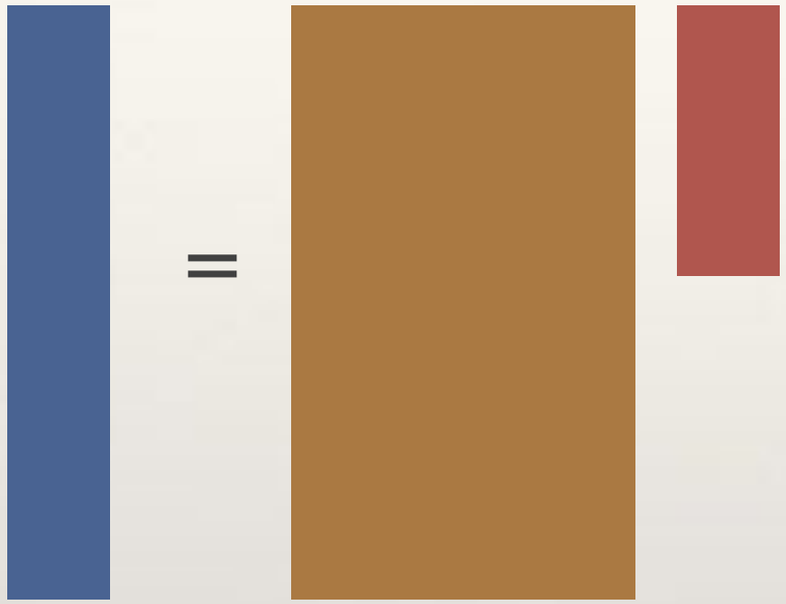
Challenges

- Training error underestimates test/generalization error.
- Overfitting: perform well on the training data but not on the future (unseen) data.
- p denotes the number of parameters the regression/classification function f has, i.e., the number of parameters we need to learn from the data.
- The gap between the two errors (training *vs.* test) gets large when p is large.

Curse of Dimensionality

- Curse of Dimensionality in Classification:
  - 1NN (one-nearest-neighbor) predicts perfectly on the training data
  - Illustration on how dimensionality changes the performance of linear classifiers: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

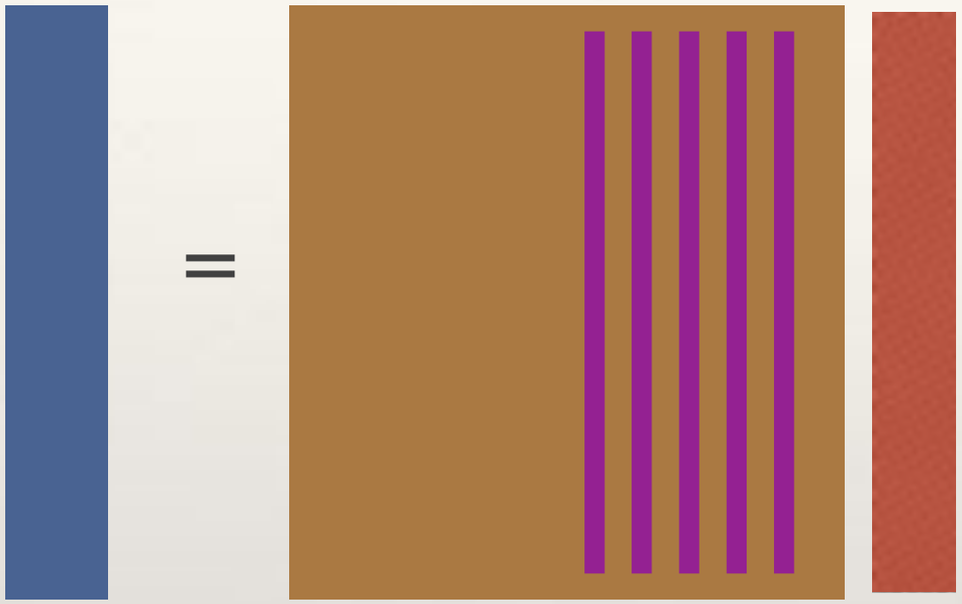
Curse of Dimensionality



$$y_{n\times 1} = X_{n\times p} w_{p\times 1}$$

Curse of Dimensionality in Regression

Curse of Dimensionality



$$y_{n\times 1} = X_{n\times n} w_{n\times 1}$$

n equations and *n* parameters

Perfect fit on the training data!

Overview

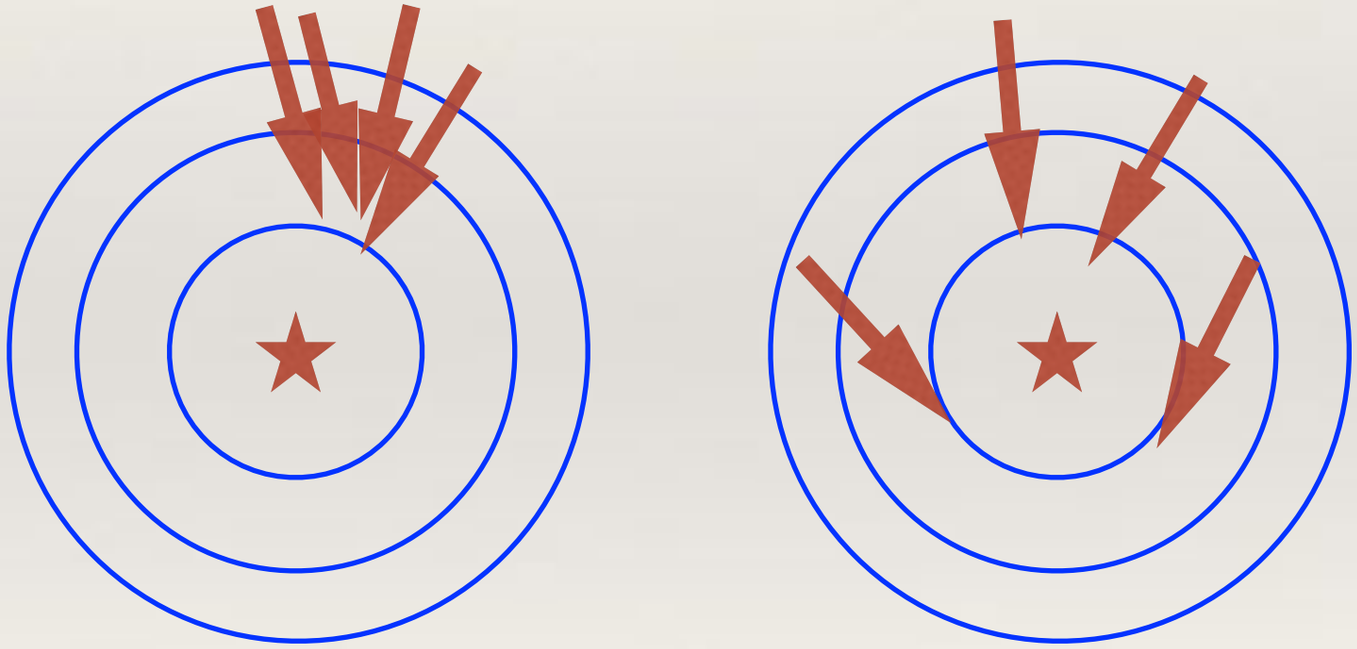
- Types of statistical learning problems
- Why learning is difficult?
- Bias variance tradeoff
- An example: kNN *vs* Linear Regression (in a separate pdf file)
- Not all about prediction

Bias Variance Tradeoff

Goal of ML: Minimize *generalization error* (i.e., error on unseen future datasets), not training error.

Source of errors:

- Bias
- Variance

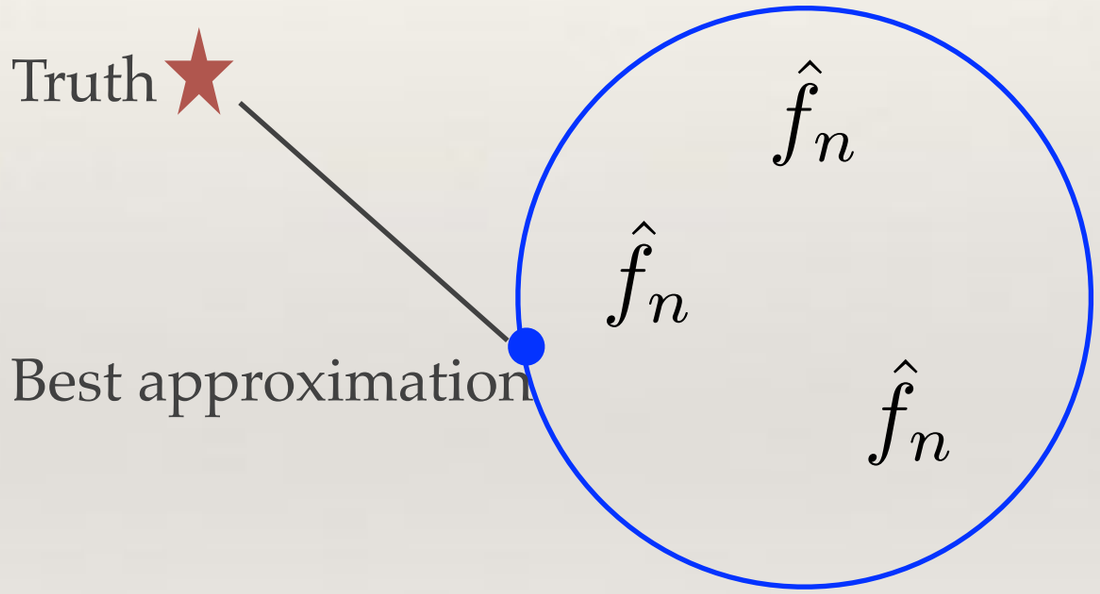


Bias Variance Tradeoff

Goal of ML: Minimize *generalization error* (i.e., error on unseen future datasets), not training error.

Source of errors:

- Bias
- Variance

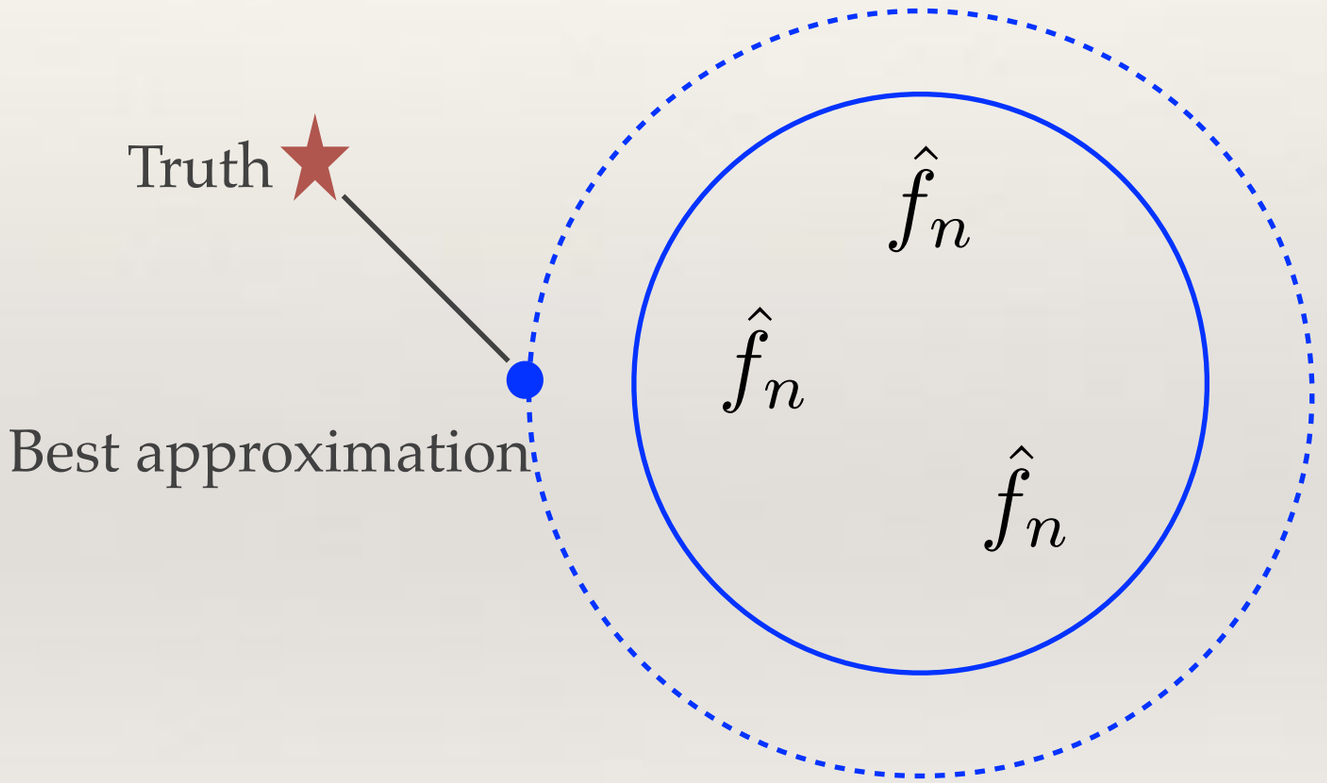


Bias Variance Tradeoff

Goal of ML: Minimize *generalization error* (i.e., error on unseen future datasets), not training error.

Source of errors:

- Bias
- Variance

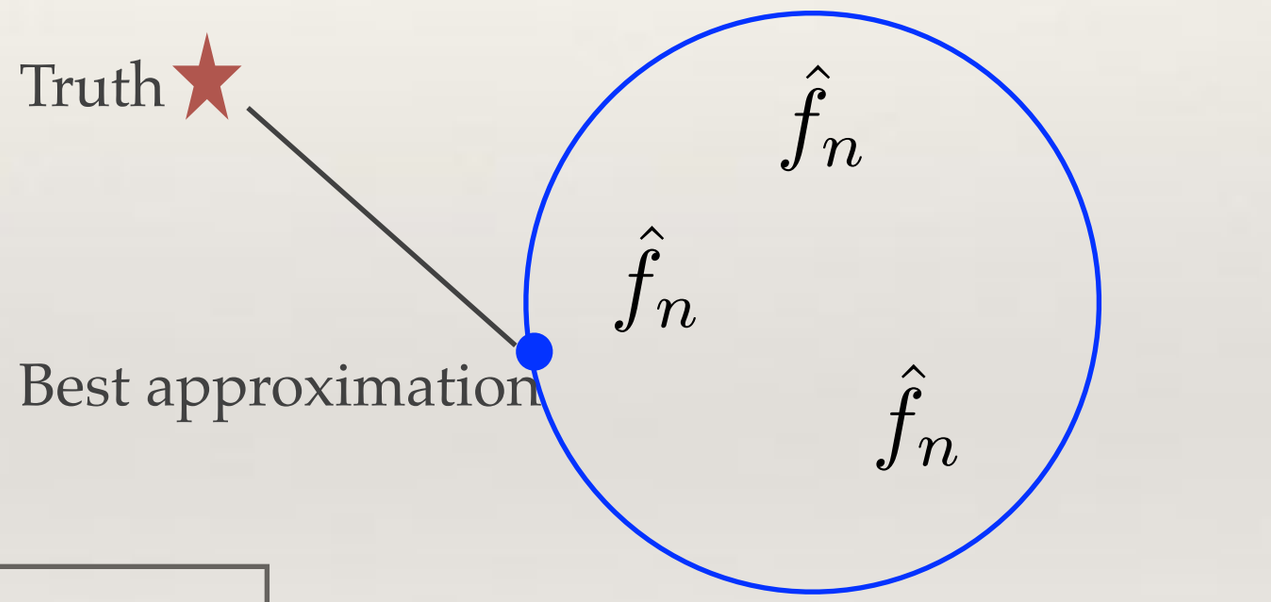


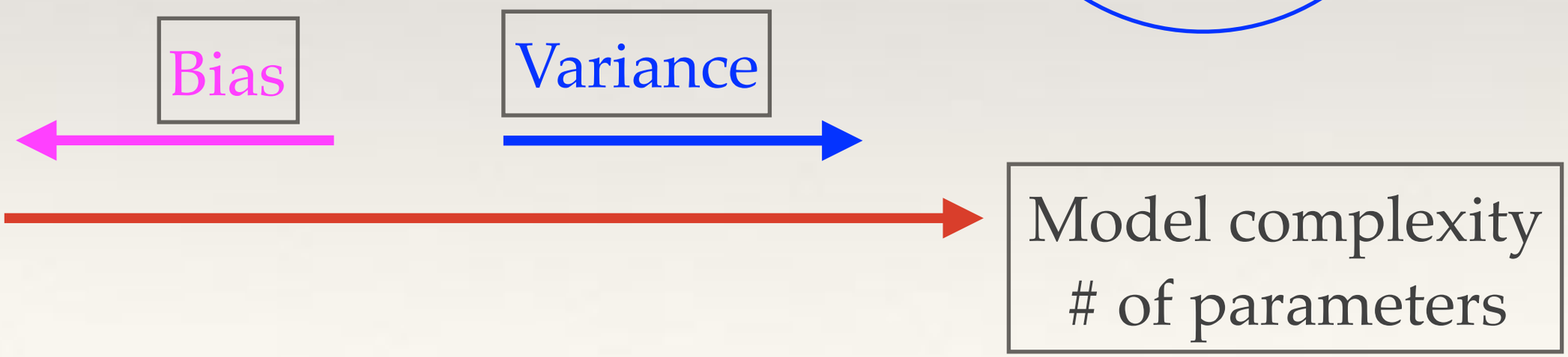
Bias Variance Tradeoff

Goal of ML: Minimize *generalization error* (i.e., error on unseen future datasets), not training error.

Source of errors:

- Bias
- Variance





Bias Variance Tradeoff

Goal of ML: Minimize *generalization error* (i.e., error on unseen future datasets), not training error.

Source of errors:

- Bias
- Variance

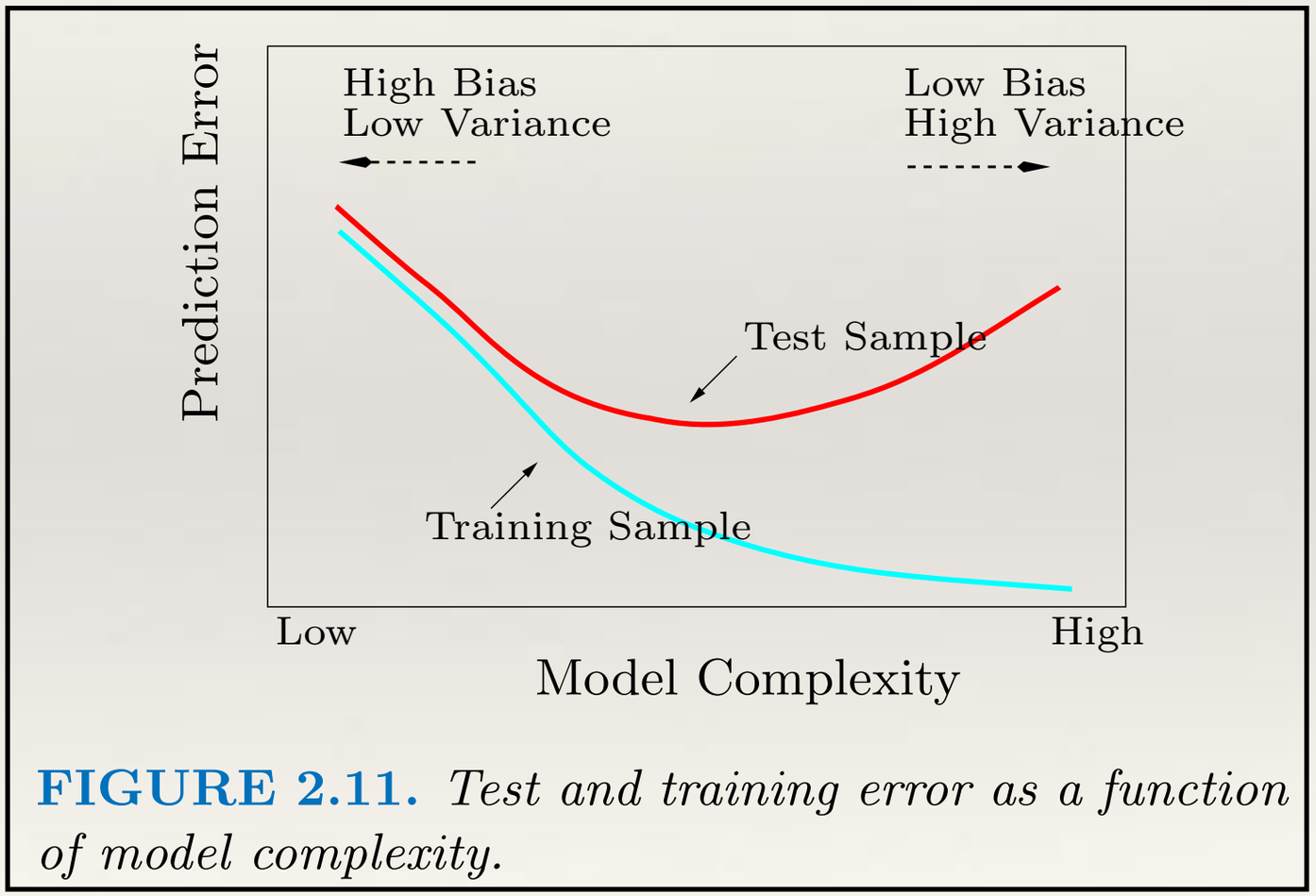


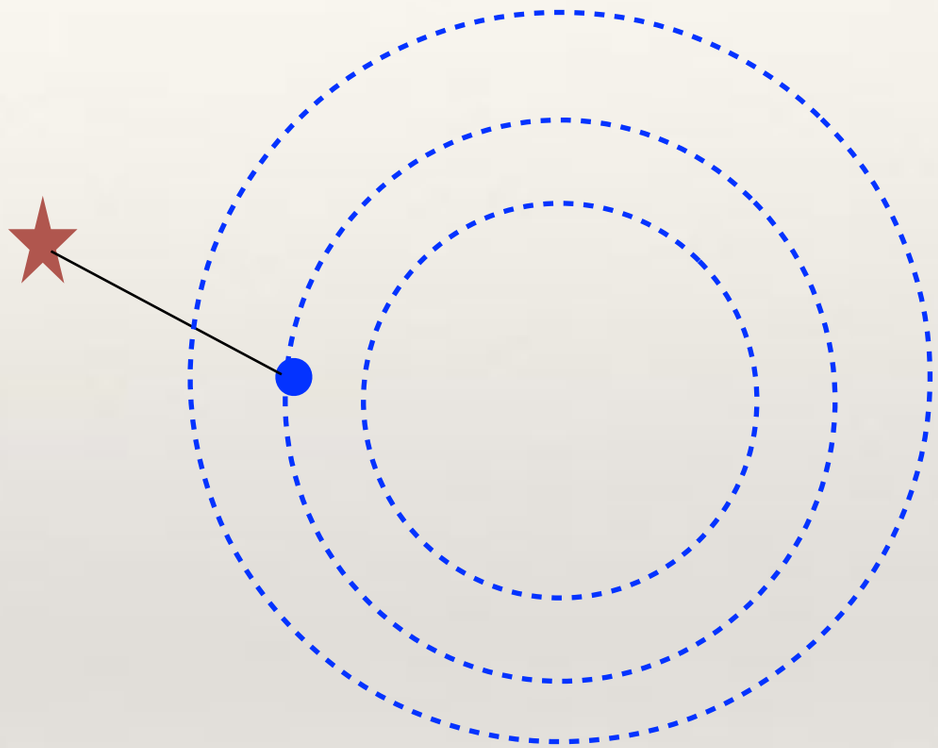
FIGURE 2.11. *Test and training error as a function of model complexity.*

What'll be Covered in Stat542

- Flexible modeling techniques to reduce bias
- Useful strategies to achieve the tradeoff between bias and variance

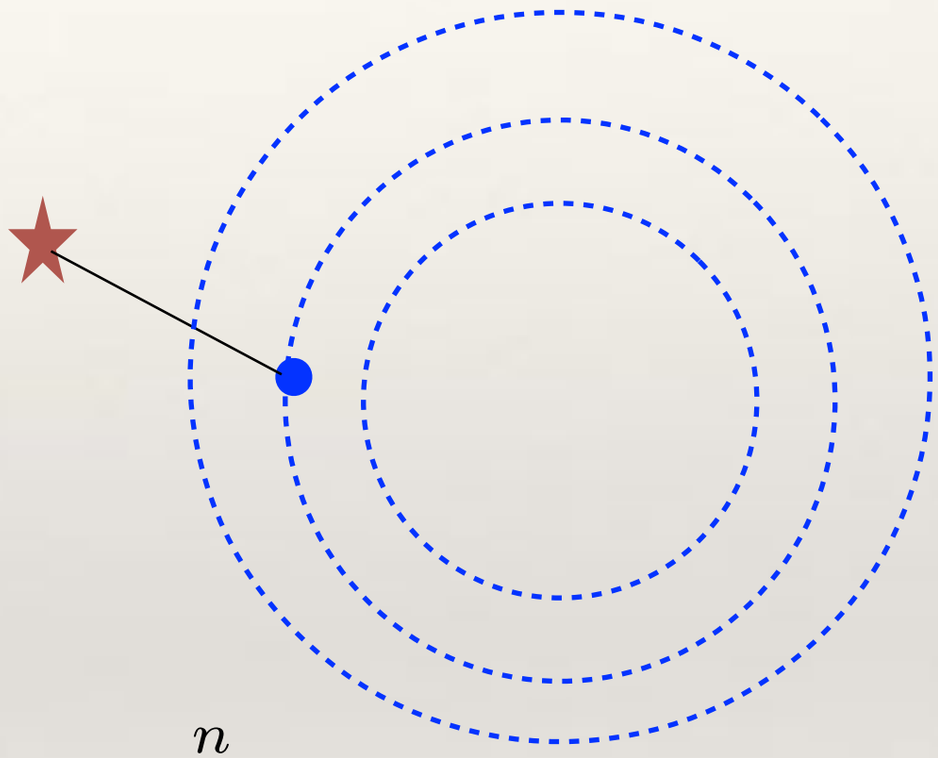
Two Successful Strategies

- Regularization: Restrict the parameters to a low-dimensional space, which is *adaptively* determined by the data.



Two Successful Strategies

- Regularization: Restrict the parameters to a low-dimensional space, which is *adaptively* determined by the data.



$$\min_w \left[\sum_{i=1}^{n} \text{loss}(y_i, f(x_i; w) + \lambda |w| \right]$$

LASSO

Two Successful Strategies

- Regularization: Restrict the parameters to a low-dimensional space, which is *adaptively* determined by the data.
- Ensemble: Average many low-bias high-variance models; averaging reduces variance.

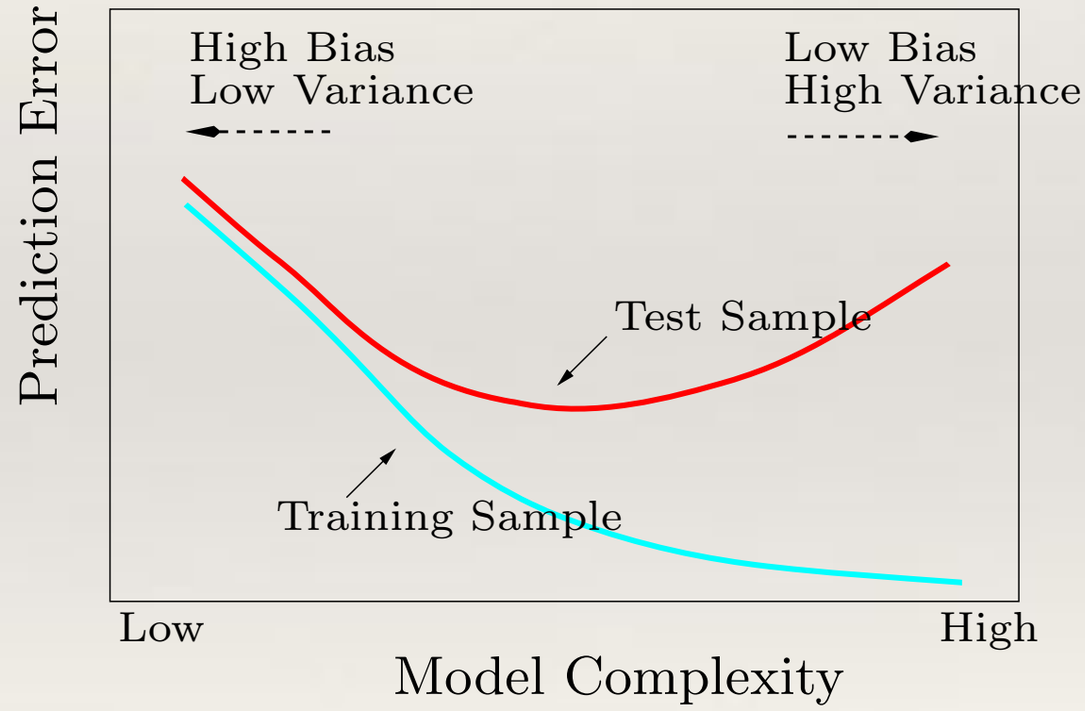


FIGURE 2.11. *Test and training error as a function of model complexity.*

Overview

- Types of statistical learning problems
- Why learning is difficult?
- Bias variance tradeoff
- An example: kNN *vs* Linear Regression
- Not all about prediction

Not All About Prediction

- Although the focus of this course is prediction, statistical learning $\neq$ prediction
- Exploration *vs.* Prediction
- Data product *vs.* decision making
- Make your model to generate actionable insights

Simulation: kNN vs Linear Regression

- Review two simple approaches for supervised learning:
  - k-Nearest Neighbors (kNN), and
  - Linear regression
- Then examine their performance on two simulated experiments to highlight the trade-off between bias and variance.

k-Nearest Neighbors

$$\hat{y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i,$$

where $N_k(x)$ denotes k samples from the training data, which (in terms of their x values) are closest to x.

Regression: kNN predicts y by a local average.

Classification: kNN can return the majority vote in $N_k(x)$, e.g., $\hat{y} = 1$ if $\frac{1}{k} \sum_{x_i \in N_k(x)} y_i > 0.5$ assuming $y \in \{1, 0\}$, or return a probability vector calculated based on the frequencies in $N_k(x)$.

What are the input parameters for kNN?

One input parameter is k, the neighborhood size.

- For 1NN ($k = 1$), the prediction at x_i (the i-th training sample) is exactly y_i, i.e., zero training error.
- For nNN ($k = n$), every neighborhood contains all the n training samples, so the prediction is the same no matter what value x takes.

The complexity or the dimension of kNN is roughly equal to n/k.

No magic value for k. It is a tuning parameter of the algorithm and is usually chosen by cross-validation.[a]

[a] If you don't know what cross-validation is, read chap 5.1 in ISLR.

The other input parameter is the metric, which we use to define the neighborhood.

The default is the Euclidean distance on the p-dimension feature vector $x \in \mathbb{R}^p$. However, it could be the weighted Euclidean, e.g.,

$$d(x, \tilde{x}) = \sum_{j=1}^{p} w_j (x_j - \tilde{x}_j)^2,$$

and we would like to learn the weights w_j's from the data.

It does not need to be Euclidean, as long as it is a similarity measure for any two samples, e.g., in image classification (from Flickr), we can measure the similarity of two images by their physical similarity, or by the similarity of their tags, or by the percentage of people who like both images.

Linear Regression

- In linear regression models, we approximate Y by a linear function of X:

$$f(\mathbf{x}) \approx \beta_0 + x_1\beta_1 + \cdots x_p\beta_p,$$

and estimate $\hat{\beta}_j$'s using the so-called Least Squares (LS) principle

$$\min_{\beta_0,\ldots,\beta_p} \sum_{i=1}^{n} (y_i - \beta_0 - x_{i1}\beta_1 \cdots - x_{ip}\beta_p)^2.$$

The solution is easy to compute – call command `lm` in `R`.

- We can also apply linear regression on classification problems with $Y = 0/1$, and predict Y to be 1 if the LS prediction $f(x)$ is bigger than 0.5 and 0 otherwise.

- There are some drawbacks with LS for classification:

  1) The squared difference $(y_i - f(\mathbf{x}_i))^2$ is not a good evaluation metric for classification;

  2) Ideally we would like to estimate the $\mathbb{P}(Y = 1 \mid X = \mathbf{x})$, however the linear function $f(\mathbf{x})$ could return us values outside $[0, 1]$.

  Later we'll learn a generalization of LS, called logistic regression model, where we assume the logit of the probability is a linear function:

$$\log \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} \approx \beta_0 + x_1\beta_1 + \cdots x_p\beta_p.$$

- Despite some of the drawbacks, the LS approach for classification works reasonably well in practice; plus its computation is very fast. So we'll apply LS on the two toy examples.

Two Toy Examples

- Let's look at the performance of these two approaches on two simulated binary classification examples from chap 2 (ESL).
- Example I: The data in each class are generated from a Gaussian distribution and the two Gaussians have different means.
- Example II: The data in each class are generated from a mixture of 10 Gaussians in each class.
- Check the `R` code and lec_Introduction_kNN_vs_LinearReg_figs.pdf posted on the course website.

Compute the Bayes Rule: Example 1

$$
\begin{aligned}
Y &\sim \text{Bern}(p), \\
X \mid Y = 0 &\sim \text{N}(\mu_0, \sigma^2 \mathbf{I}_p), \\
X \mid Y = 1 &\sim \text{N}(\mu_1, \sigma^2 \mathbf{I}_p).
\end{aligned}
$$

The joint dist can be factorized as $P(Y, X) = P(Y) \times P(X \mid Y)$. All the calculation on the next slide is to use Bayes' theorem to compute $P(Y \mid X)$.

Note that $P(Y, X)$ is neither a pmf (for discrete r.v.) nor a pdf (for continuous r.v.). It will get a little technical if I tend to rigorously justify my calculation on the next slide. Let's ignore the technicality, and just treat X as discrete with pmf same as its density function. Trust me, the result is correct.

$$P(Y=1 \mid X=x) = \frac{P(Y=1, X=x)}{P(X=x)}$$

$$= \frac{P(Y=1, X=x)}{P(Y=1, X=x) + P(Y=0, X=x)}$$

$$= \frac{P(Y=1)P(X=x|Y=1)}{P(Y=1)P(X=x|Y=1) + P(Y=0)P(X=x|Y=0)}$$

$$= \frac{(p)\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ - \frac{\|x-\mu_1\|^2}{2\sigma^2} \right\}}{(p)\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ - \frac{\|x-\mu_1\|^2}{2\sigma^2} \right\} + (1-p)\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ - \frac{\|x-\mu_0\|^2}{2\sigma^2} \right\}}$$

$$= \left[1 + \exp\left\{ \frac{1}{2\sigma^2}(\|x-\mu_1\|^2 - \|x-\mu_0\|^2) - \log\frac{p}{1-p} \right\} \right]^{-1}$$

kNN *vs.* Linear Regression

- Linear regression: f is linear
  - low variance: need to estimate $p = 3$ parameters
  - high bias (underfit): linear assumption is very restrictive
- kNN: no assumption on f, except local some smoothness.
  - low bias (overfit): flexible and adaptive. It can be shown that as $k, n \to \infty$ such that $k/n \to 0$, kNN is consistent.
  - high variance: num of parameters for kNN is roughly n/k, which goes to ∞ in order to achieve consistency.