# Parameter estimation, Bias & Variance

## Data Mining II

Year 2009-10

Lluís Belanche
Alfredo Vellido

*Introduction to Pattern Analysis*
*Ricardo Gutierrez-Osuna*
*Texas A&M University*

# Maximum Likelihood (1)

- **Suppose we consider estimating a density function p(x) which depends on a number of parameters $\theta=[\theta_1, \theta_2,\ldots\theta_M]^T$**
  - For a Gaussian pdf $\theta_1=\mu$, $\theta_2=\sigma$ and $p(x)=N(\mu, \sigma)$
  - To make the dependence on the parameters $\theta$ explicit we write $p(x|\theta)$
- **Assume that we have a number of examples $X=\{x^{(1}, x^{(2},\ldots x^{(N}\}$ drawn independently from the distribution $p(x|\theta)$ (an i.i.d. set)**
  - Then we can write

$$p(X|\theta)=\prod_{k=1}^{N} p\left(x^{(k}|\theta\right)$$

  - The ML estimate of $\theta$ is the value that maximizes the likelihood $p(X|\theta)$

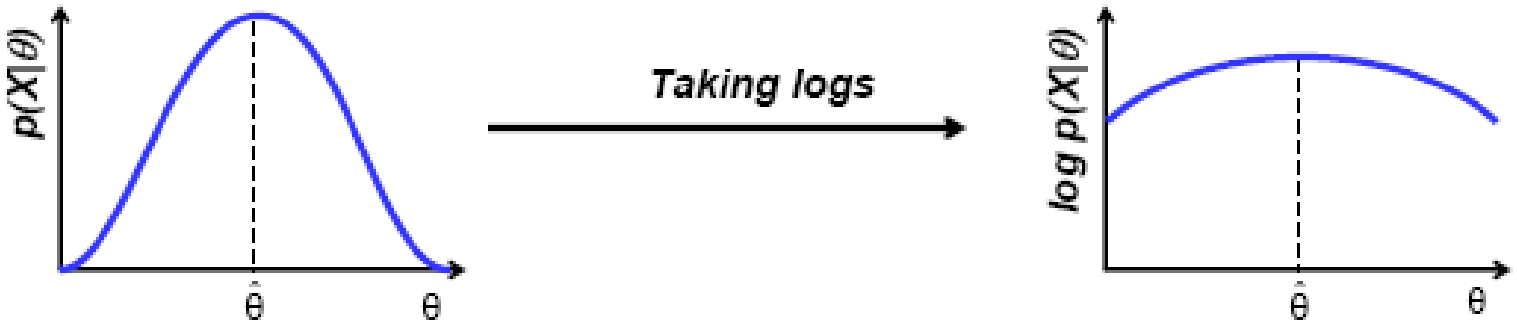
$$\hat{\theta}=\text{argmax}\left[p(X|\theta)\right]$$

    - This corresponds to the intuitively pleasing idea of choosing the value of $\theta$ that is most likely to give rise to the data!

# Maximum Likelihood (2)

- **For analytical purposes it is convenient to work with the log of the likelihood**
  - Since the log is a monotonic function

$$\hat{\theta} = \text{argmax}\left[p(X \mid \theta)\right] = \text{argmax}\left[\log p(X \mid \theta)\right]$$




- **Then the Maximum Likelihood estimate of the parameter θ can be written as**

$$\hat{\theta} = \text{argmax}\left[\log \prod_{k=1}^{N} p\left(x^{(k} \mid \theta\right)\right] = \text{argmax}\left[\sum_{k=1}^{N} \log p\left(x^{(k} \mid \theta\right)\right]$$

# Example 1: Gaussian case
## (σ known but μ unknown)

- **Assume a dataset $X=\{x^{(1}, x^{(2},\ldots x^{(N}\}$ and a density of the form $p(x)=N(\mu,\sigma)$ where the standard deviation $\sigma$ is known**
- **What is the Maximum Likelihood estimate of the mean?**

$$\theta=\mu\Rightarrow\hat{\theta}=\operatorname{argmax}\sum_{k=1}^{N}\log p\left(x^{(k}\mid\theta\right)$$

$$=\operatorname{argmax}\sum_{k=1}^{N}\log\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}\left(x^{(k}-\mu\right)^2\right)\right)$$

$$=\operatorname{argmax}\sum_{k=1}^{N}\left\{\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right)-\frac{1}{2\sigma^2}\left(x^{(k}-\mu\right)^2\right\}$$

- The maxima (or minima) of a function are defined by the zeros of its derivative:

$$\frac{\partial\sum_{k=1}^{N}\log p\left(x^{(k}\mid\theta\right)}{\partial\theta}=\frac{\partial}{\partial\mu}\sum_{k=1}^{N}\{\bullet\}=0\Rightarrow\mu=\frac{1}{n}\sum_{k=1}^{N}x^{(k}$$

- So the ML estimate of the mean is the average value of the training data, a very intuitive result!

# Example 2: Gaussian case (both μ and σ unknown)

- **This is a more general case when neither the mean nor the standard deviation are known**
  - Fortunately, the problem can be solved in the same fashion
  - In this case, the derivative becomes a gradient since we have two variables

$$\hat{\theta} = \begin{bmatrix} \theta_1 = \mu \\ \theta_2 = \sigma^2 \end{bmatrix} \Rightarrow \nabla_\theta = \begin{bmatrix} \frac{\partial}{\partial \theta_1} \sum_{k=1}^{N} \log p(x^{(k)} \mid \theta) \\ \frac{\partial}{\partial \theta_2} \sum_{k=1}^{N} \log p(x^{(k)} \mid \theta) \end{bmatrix} = \sum_{k=1}^{N} \begin{bmatrix} \frac{1}{\theta_2}(x^{(k)} - \theta_1) \\ -\frac{1}{2\theta_2} + \frac{(x^{(k)} - \theta_1)^2}{2\theta_2^2} \end{bmatrix} = 0$$

  - Solving for $\theta_1$ and $\theta_2$ yields

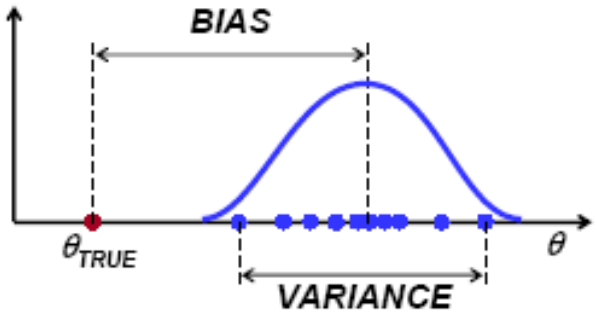
$$\hat{\theta}_1 = \frac{1}{N} \sum_{k=1}^{N} x^{(k)}; \quad \hat{\theta}_2 = \frac{1}{N} \sum_{k=1}^{N} (x^{(k)} - \hat{\theta}_1)^2$$

    - Therefore, the ML of the variance is the sample variance of the dataset, again a very pleasing result
  - Similarly, it can be shown that the Maximum Likelihood parameter estimates for the multivariate Gaussian are also the sample mean vector and sample covariance matrix

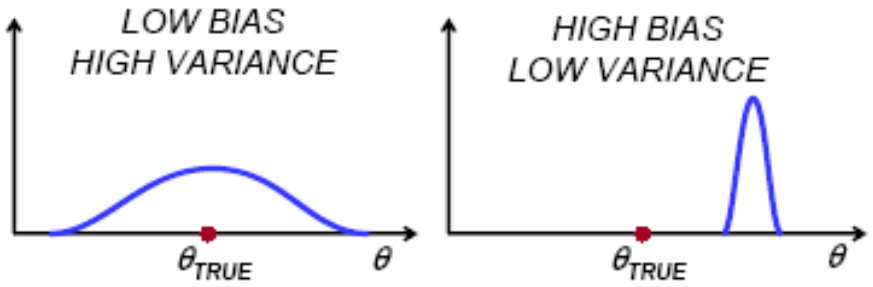
$$\hat{\mu} = \frac{1}{N} \sum_{k=1}^{N} x^{(k)}; \quad \hat{\Sigma} = \frac{1}{N} \sum_{k=1}^{N} (x^{(k)} - \hat{\mu})(x^{(k)} - \hat{\mu})^T$$

# Bias and variance (1)

- **How good are these estimates? Two measures of "goodness" are used for statistical estimates**
  - **BIAS**: how close is the estimate to the true value?
  - **VARIANCE**: how much does the estimate change for different runs (e.g. different datasets)?




- **The bias-variance tradeoff**
  - In most cases, you can only decrease one of them at the expense of the other



# Bias and variance (2)

- **What is the bias of the ML estimate of the mean?**

$$E[\hat{\mu}] = E\left[\frac{1}{N}\sum_{k=1}^{N} x^{(k}\right] = \frac{1}{N}\sum_{k=1}^{N} E[x^{(k}] = \mu$$

  - Therefore the mean is an unbiased estimate

- **What is the bias of the ML estimate of the variance?**

$$E[\hat{\sigma}^2] = E\left[\frac{1}{N}\sum_{k=1}^{N}(x^{(k} - \hat{\mu})^2\right] = \frac{N-1}{N}\sigma^2 \neq \sigma^2$$

  - Thus, the ML estimate of variance is BIASED
    - The problem is that the ML estimate of variance uses the ML estimate of the mean instead of its true value
  - How "bad" is this bias?
    - For $N \to \infty$ the bias becomes zero asymptotically
    - The bias is only noticeable when we have very few samples, in which case we should not be doing statistics in the first place
  - Notice that MATLAB uses an unbiased estimate of the co-variance

$$\hat{\Sigma}_{UNBIASED} = \frac{1}{N-1}\sum_{k=1}^{N}(x^{(k} - \hat{\mu})(x^{(k} - \hat{\mu})^T$$

# Bias-Variance in Regression

- True function is $y = f(x) + \varepsilon$, where $\varepsilon \sim N(0,\sigma^2)$
- Given a set of examples $X=\{(x_i, y_i)\}$ we fit an hypothesis $h(x)$ to X to minimize the squared error $\Sigma_i [y_i - h(x_i)]^2$

For a new data point x* with observed value $y^* = f(x^*) + \varepsilon$, we would like to understand the expected prediction error or EPE of h in x*:

$$EPE(h; x^*) = E[ (y^* - h(x^*))^2 ]$$

which decomposes into "squared bias"+"variance"+ "noise":

$$EPE(h; x^*) = E[ (y^* - h(x^*))^2 ] = (E[h(x^*)] - f(x^*))^2 + E[ (h(x^*) - E[h(x^*)])^2 ] + E[ (y^* - f(x^*))^2 ]$$

# Bias, Variance, and Noise

- Bias: $E[h(x^*)] - f(x^*)$

  → How much average estimates deviate from the truth: describes the systematic error of $h(x^*)$

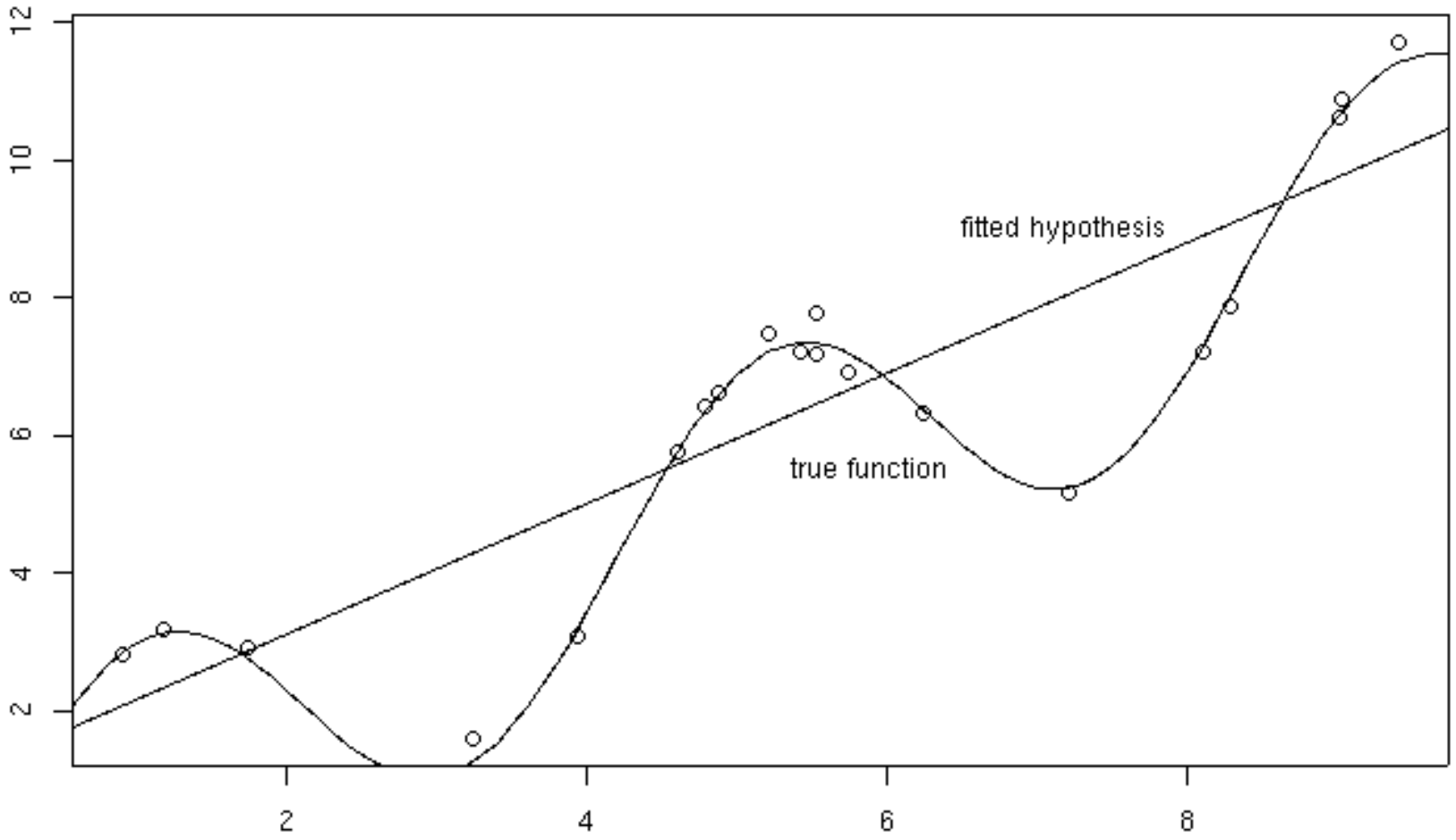
- Variance: $E[ (h(x^*) - E[h(x^*)])^2 ]$

  → Variability of the estimates upon changing the data sample: describes how much $h(x^*)$ varies from one training sample X to another

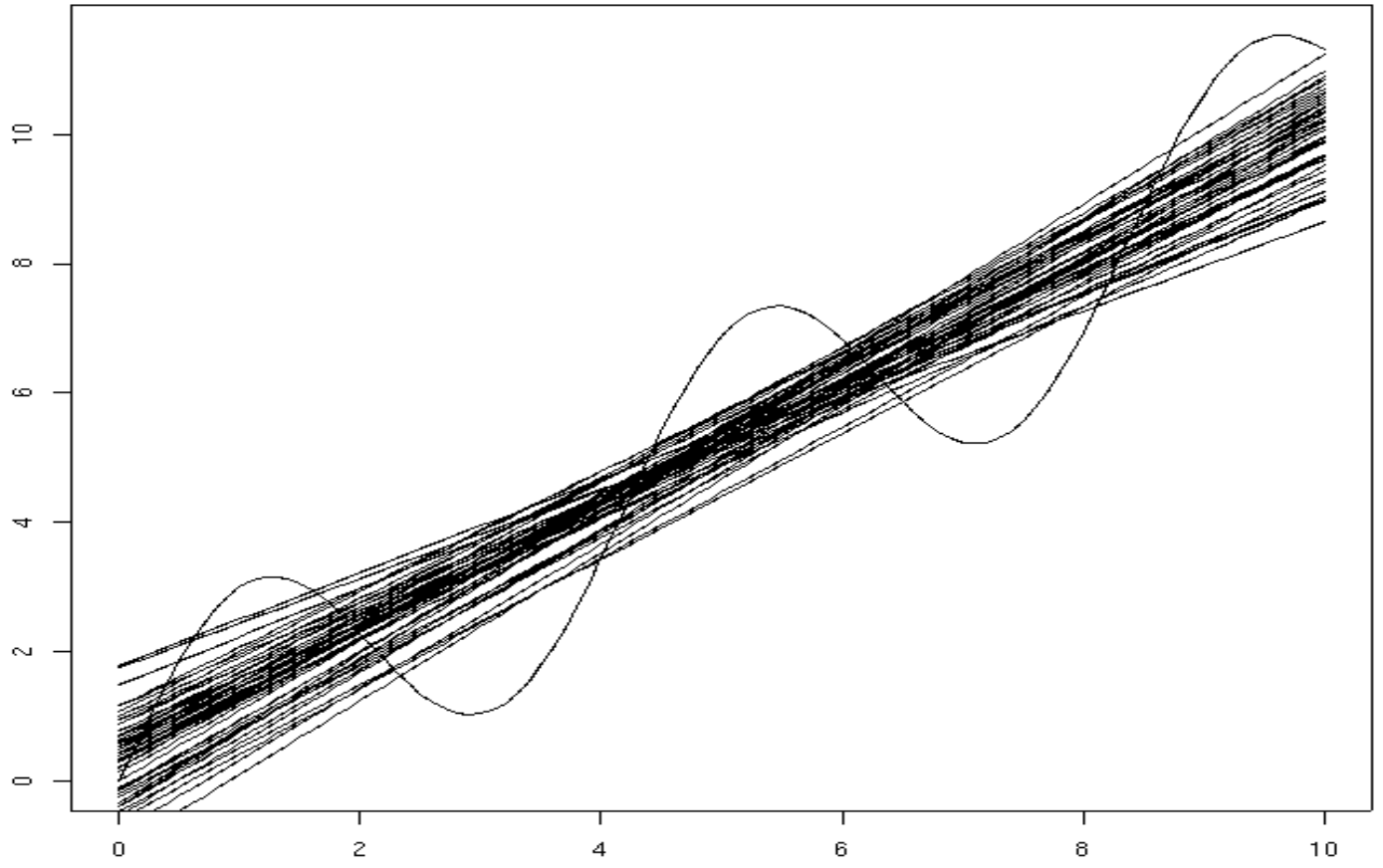
- Noise: $E[ (y^* - f(x^*))^2 ] = E[\varepsilon^2] = \sigma^2$

  → Inherent random component: describes how much $y^*$ varies from $f(x^*)$. Noise due to ignorance

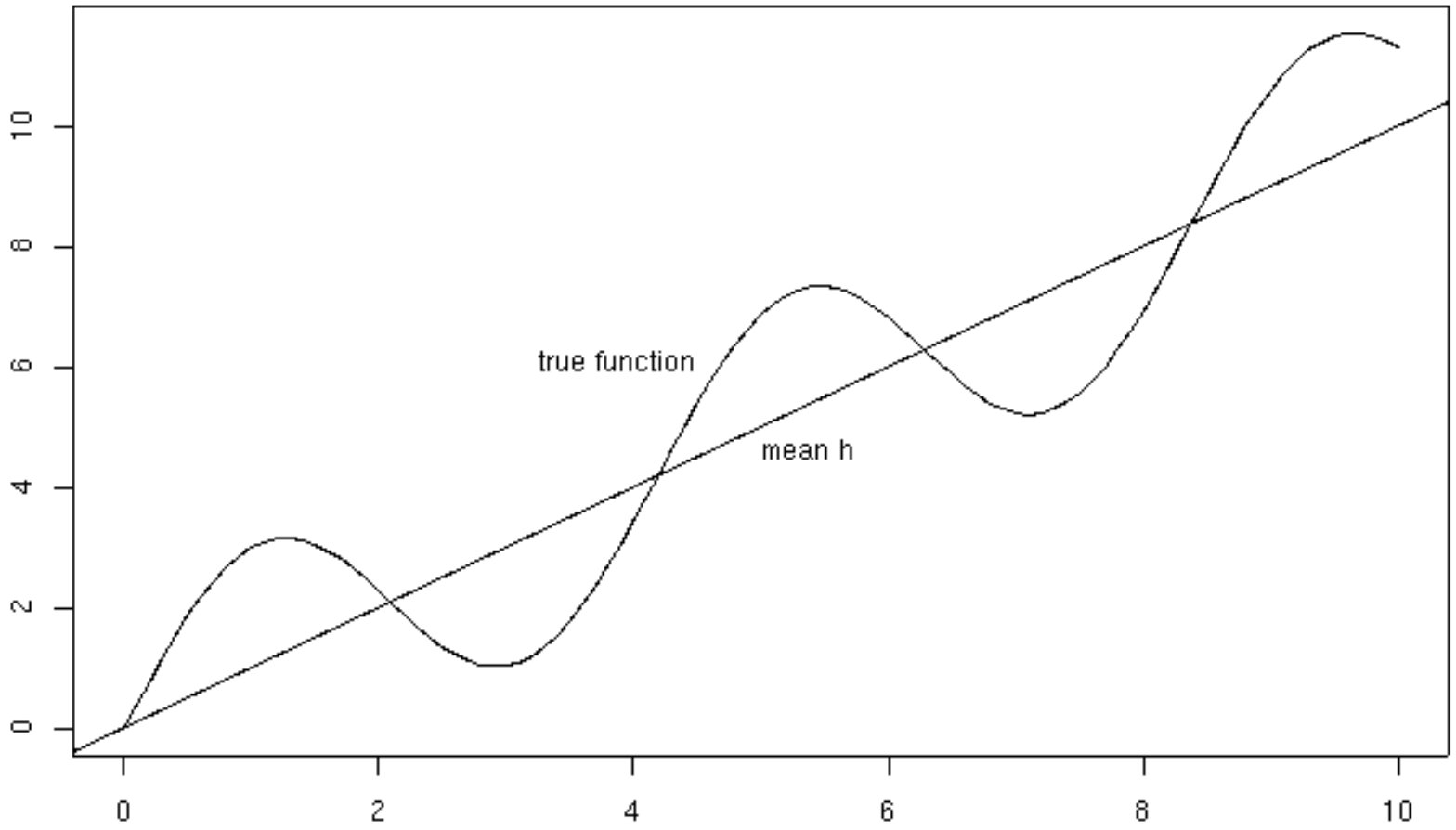
# Example: 20 points; linear fit h(x)
# y = x + 2 sin(1.5x) + N(0,0.2)



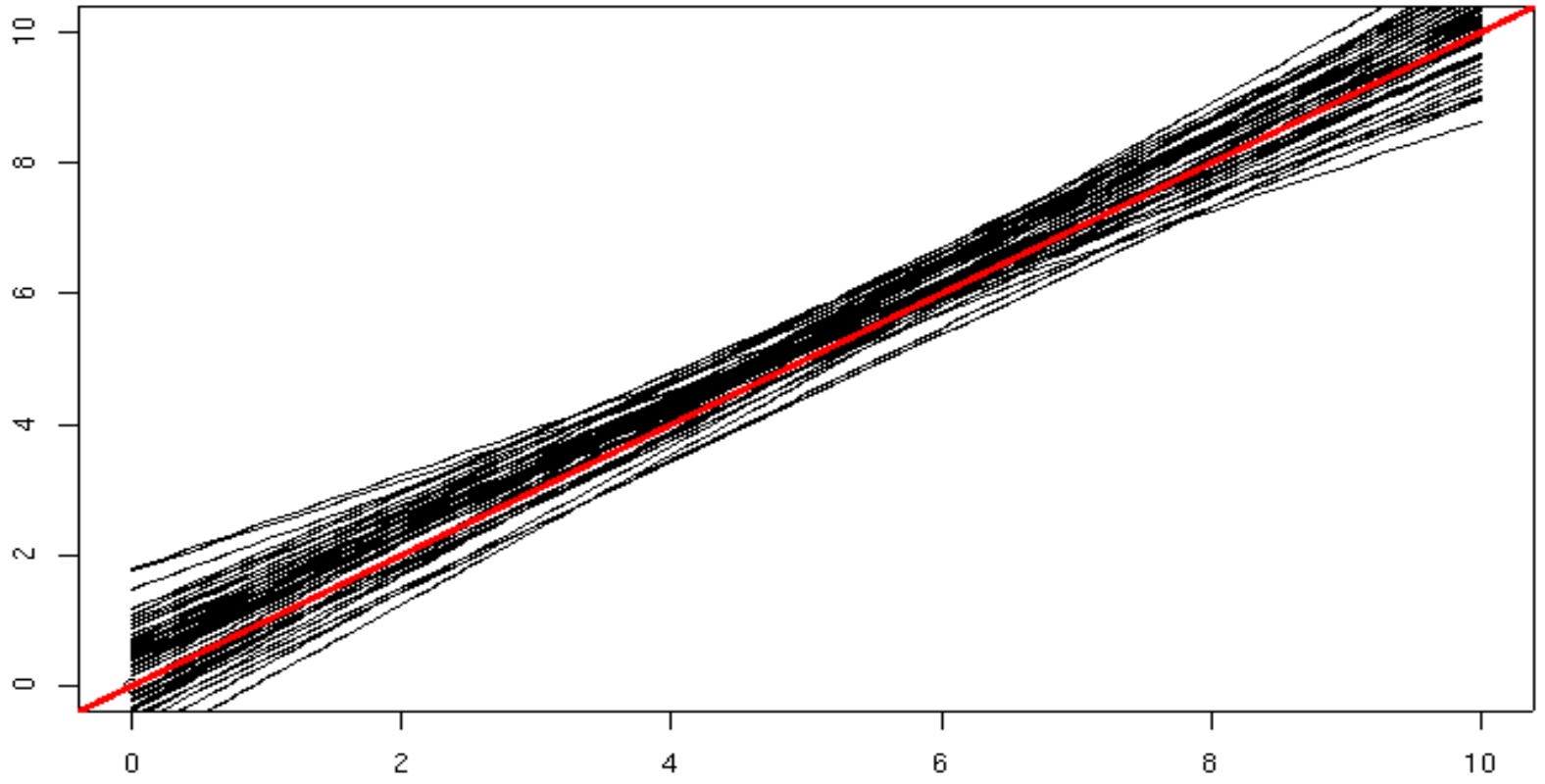
# 50 fits (20 examples each)



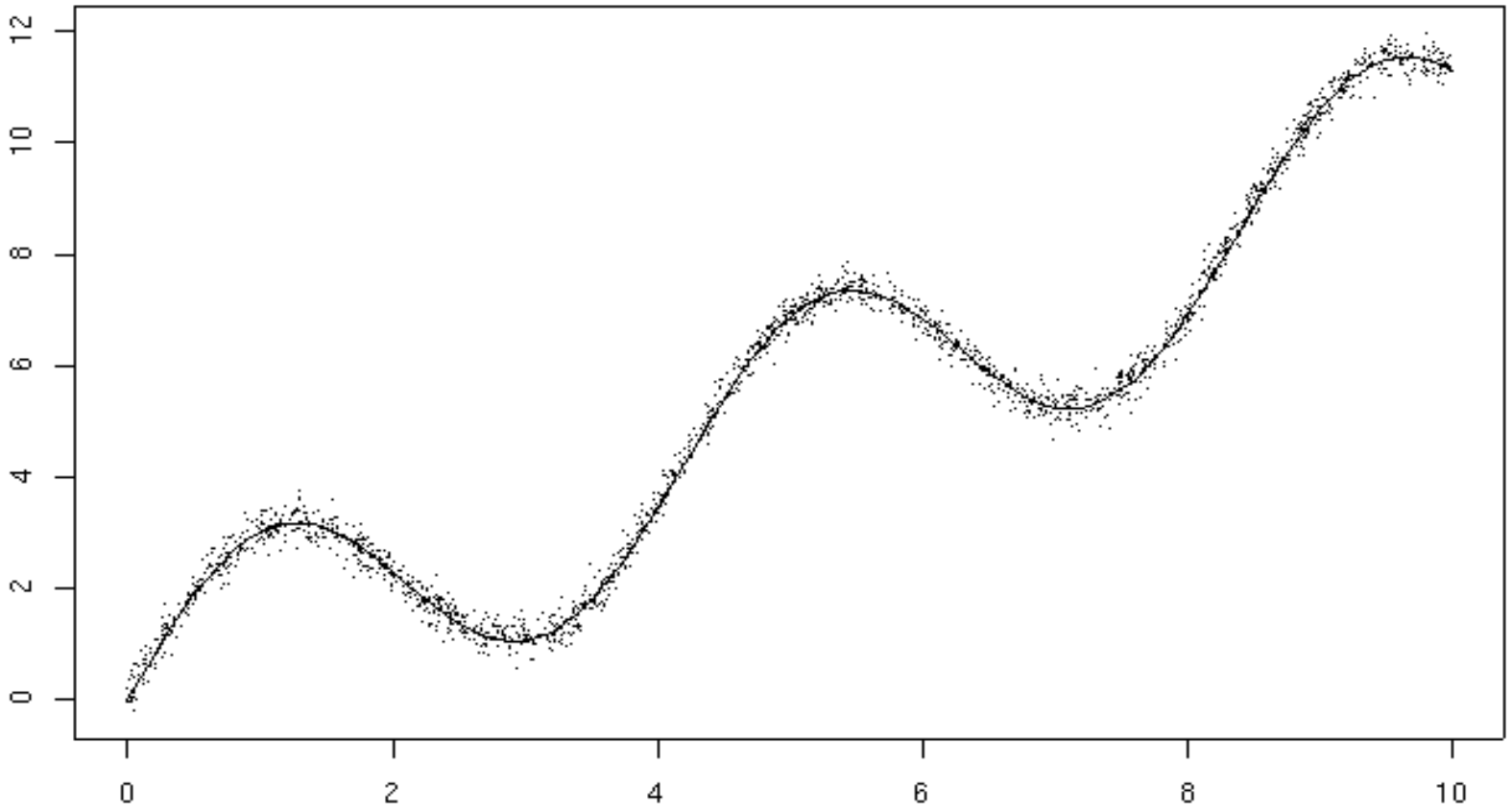
# Bias



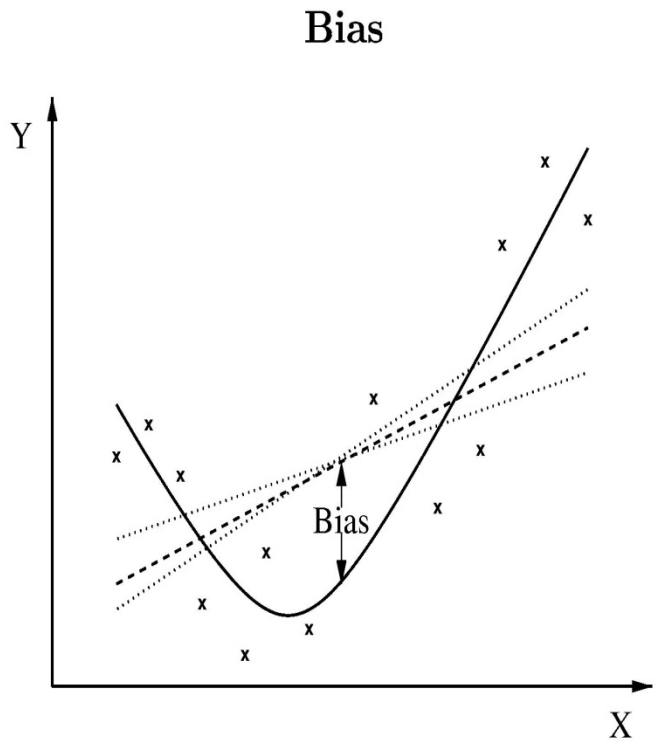
# Variance

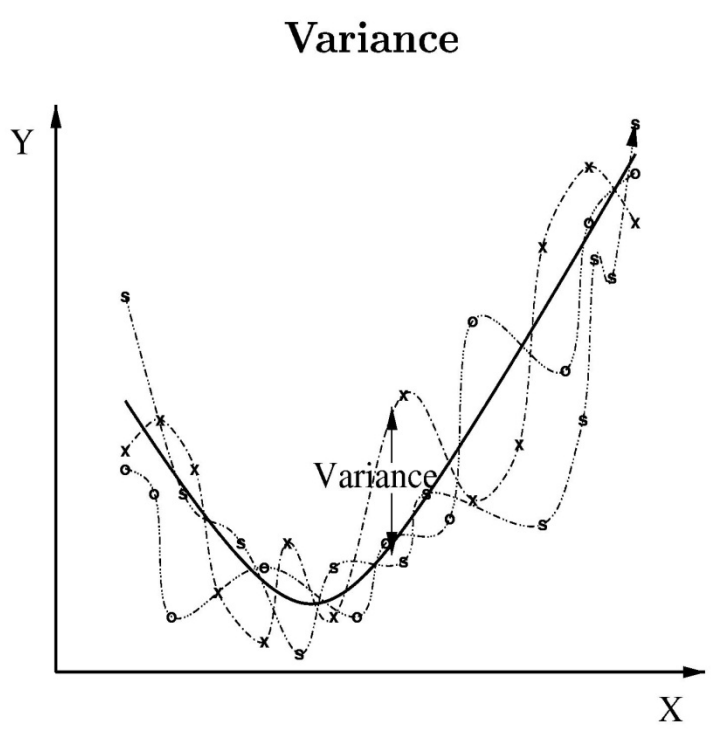


# Noise

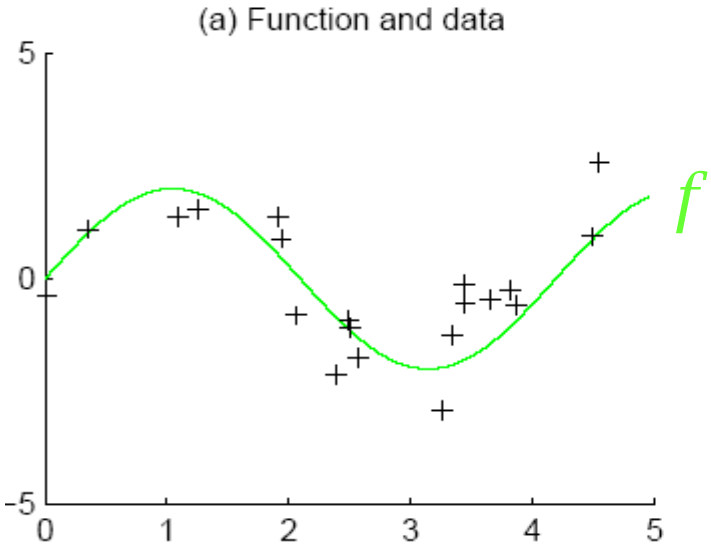


# Yet again?

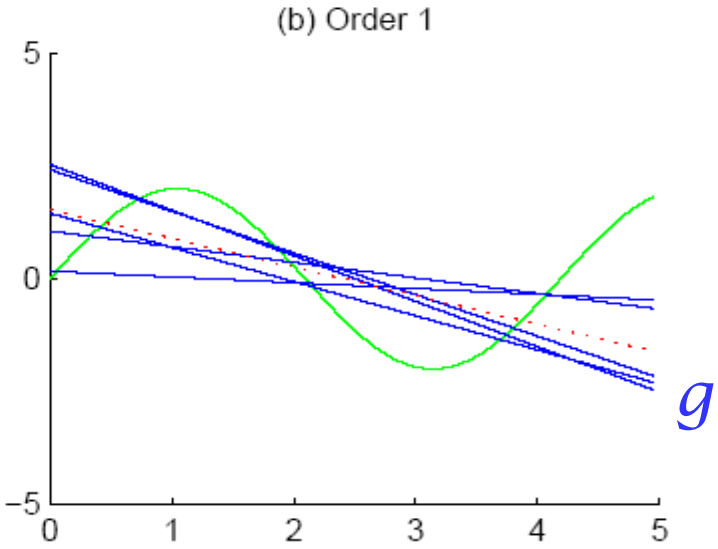


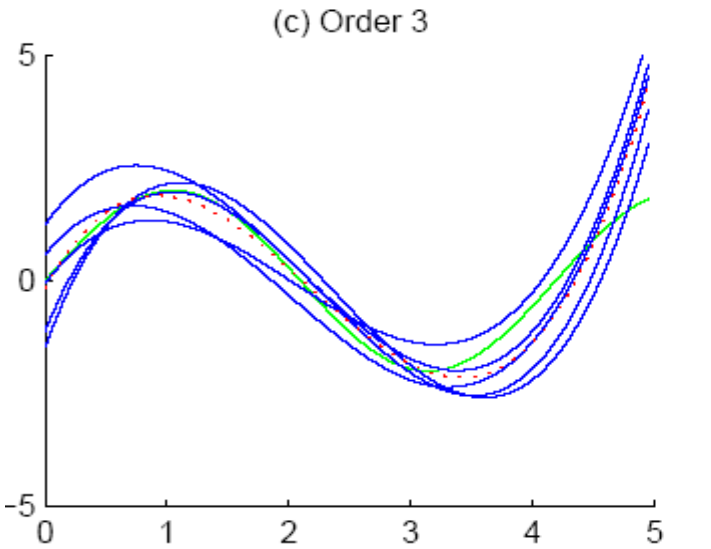
# The Bias-Variance Dilemma

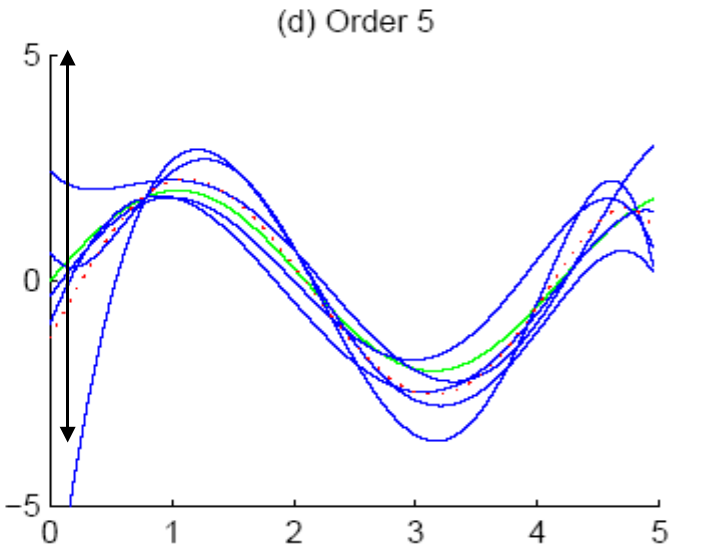



$f$

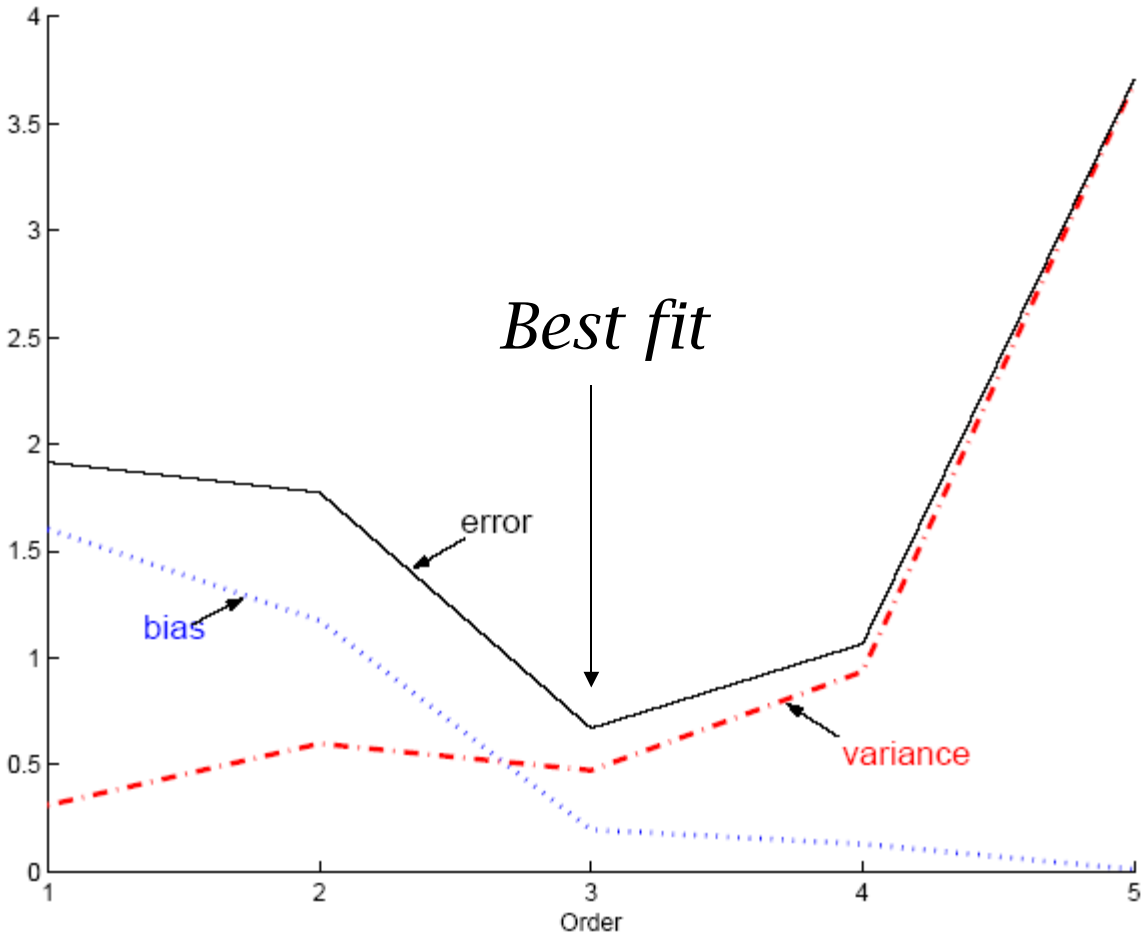



$g$



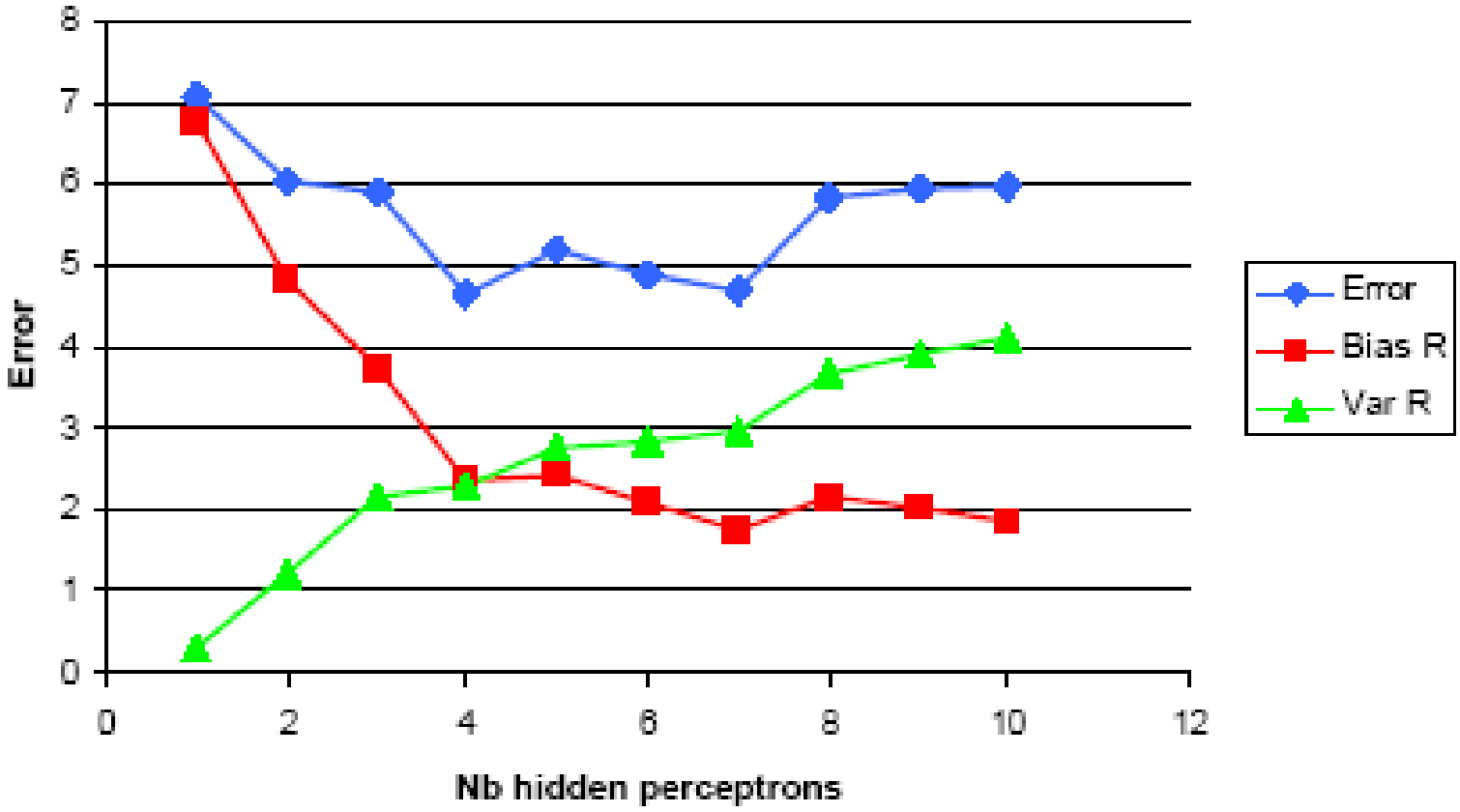



# The Bias-Variance Dilemma

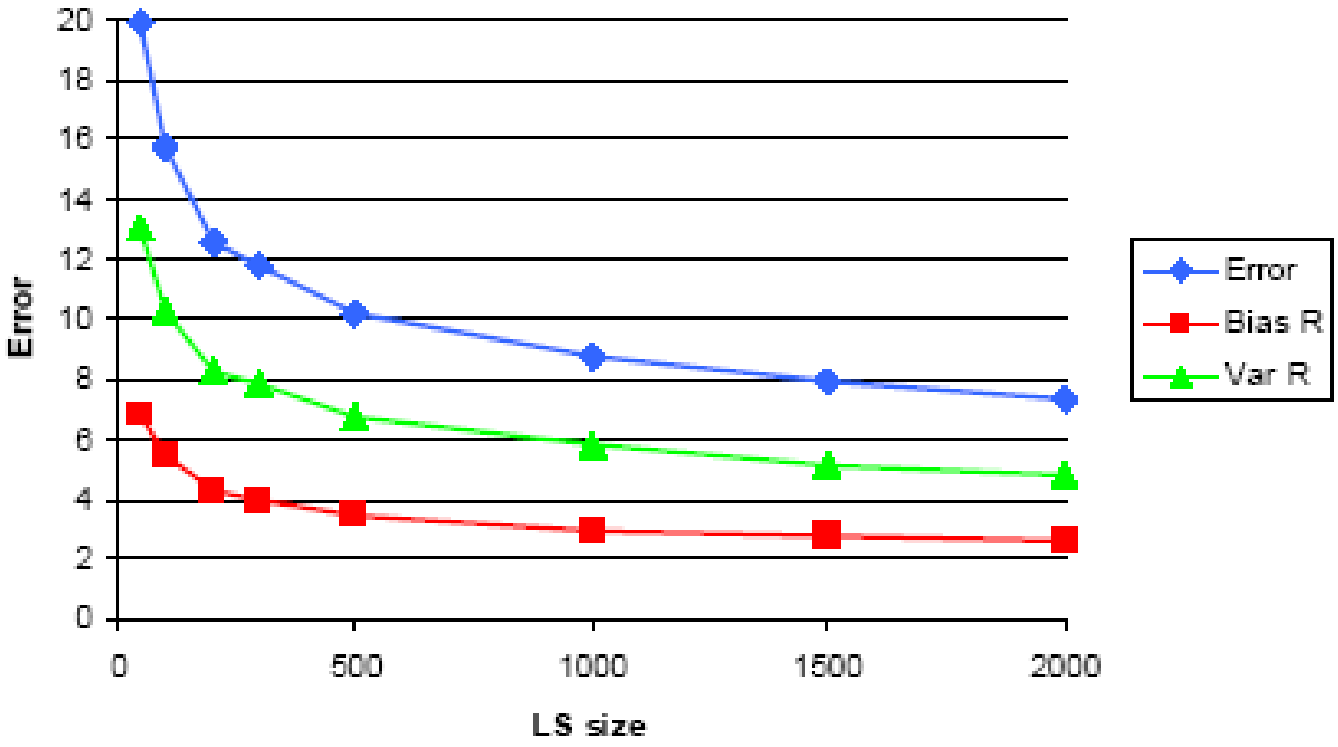



Usually, the bias is a decreasing function of model complexity, while variance is a increasing function of the complexity

# Example 1: the case of ANNs



# Example 2: the case of 1NN




- When model complexity is dependent on training sample size, then both bias and variance decrease with sample size.

# Measuring Bias & Variance (1)

- In practice (unlike in theory), we have only ONE training set X.

- We can simulate multiple training sets by bootstrap replicates
  - X' = {x | x is drawn at random with replacement from X} and |X'| = |X|.

# Measuring Bias & Variance (2)

- Assume a noiseless environment
- Construct B bootstrap replicates of X (e.g., B = 200): $X_1$, …, $X_B$
- Apply learning algorithm to each replicate $X_i$ to obtain hypothesis $h_i$
- Let $T_i = X \setminus X_i$
- Compute prediction $h_i(x)$ for each x in $T_i$

# Measuring Bias & Variance (3)

For each original data point x*, we now have the observed corresponding value y* and a number k≤B of predictions $y_j=h_j(x^*)$, $j=1,\ldots k$

- Compute the average prediction h*
- Estimate <u>bias</u> as (h* – y*)
- Estimate <u>variance</u> as $\Sigma^k_{j=1} (y_j – h^*)^2/(k – 1)$

# But …

- Bootstrap replicates are not real data
- We ignore the noise
  - If we have multiple data points with the same x value, then we can estimate the noise
  - We can also estimate noise by pooling y values from nearby x values

# In conclusion …

- Prediction error can be decomposed into bias and variance (and noise)
  - Bias arises when the classifier cannot represent the true function: the classifier **underfits** the data
  - Variance arises when the classifier depends too much on the particular training set: the classifier **overfits** the data
- There is an inherent trade-off between the two