# Regression: Bias and Variance

Jiayu Zhou

[1]Department of Computer Science and Engineering
Michigan State University
East Lansing, MI USA

February 16, 2016

# Table of contents

1. Decision Theory
2. Cross Validation

# Overfitting

Different perspectives of the same problem (not enough data):

- Huge magnitude of model parameters
- Close-to-singular design matrix
- Model complexity

Regularization provides an effective way to prevent overfitting.

$$\min \text{Loss}(\mathbf{w}) + \text{Regularization}(\mathbf{w})$$

# Squared Loss and Conditional Mean

Choosing a specific estimate $y(\mathbf{x}) = y(\mathbf{x}; \mathbf{w})$ of the value of $t$ for each input $\mathbf{x}$, given the squared loss $L(t, y(\mathbf{x})) = \{t - y(\mathbf{x})\}^2$

$$\mathbb{E}[L] = \iint L(t, y(\mathbf{x}))p(\mathbf{x}, t)\ \mathrm{d}\mathbf{x}\ \mathrm{d}t = \iint \{t - y(\mathbf{x})\}^2 p(\mathbf{x}, t)\ \mathrm{d}\mathbf{x}\ \mathrm{d}t$$

To obtain the optimal prediction function $y(x)$, we set:

$$\frac{\delta \mathbb{E}[L]}{\delta y} = 2 \int \{y(\mathbf{x}) - t\} p(\mathbf{x}, t)\ \mathrm{d}t = 0 \Rightarrow y(\mathbf{x}) = \mathbb{E}_t[t|\mathbf{x}]$$

where $\mathbb{E}_t[t|\mathbf{x}] = \int t p(t|\mathbf{x}) dt$ is the *regression function*: conditional average of $t$ conditioned on $\mathbf{x}$.

# Decomposition I

Given the knowledge $y^*(\mathbf{x}) = \mathbb{E}_t[t|\mathbf{x}]$, by properly manipulate the loss, we can get:

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x} + \int \text{var}[t|\mathbf{x}] p(\mathbf{x}) d\mathbf{x} \quad (1)$$

- First term: mismatch between OUR hypothesis and target. Will vanish when $y(\mathbf{x}) = \mathbb{E}_t[t|\mathbf{x}]$.
- Second term: intrinsic variability of the target data (noise). This is the irreducible minimum value of the loss function.

# Decomposition I

Given the knowledge $y^*(\mathbf{x}) = \mathbb{E}_t[t|\mathbf{x}]$, by properly manipulate the loss, we can get:

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x}) d\mathbf{x} + \int \text{var}[t|\mathbf{x}] p(\mathbf{x}) d\mathbf{x} \quad (1)$$

- First term: mismatch between OUR hypothesis and target. Will vanish when $y(\mathbf{x}) = \mathbb{E}_t[t|\mathbf{x}]$.
- Second term: intrinsic variability of the target data (noise). This is the irreducible minimum value of the loss function.
- Can we obtain $\mathbb{E}_t[t|\mathbf{x}] = \int t p(t|\mathbf{x}) dt$?

# Decomposition I (cont.)

- In practice, we have a data set $\mathcal{D}$ with a finite number $N$ of data points, so we do not know the regression function $h(x) = \mathbb{E}_t[t|\mathbf{x}]$ exactly.
- We learn/estimate $y(\mathbf{x})$ from a particular dataset $\mathcal{D}$, and we write $y(\mathbf{x}) = y(\mathbf{x}; \mathcal{D})$ to show the dependence $\mathcal{D}$.

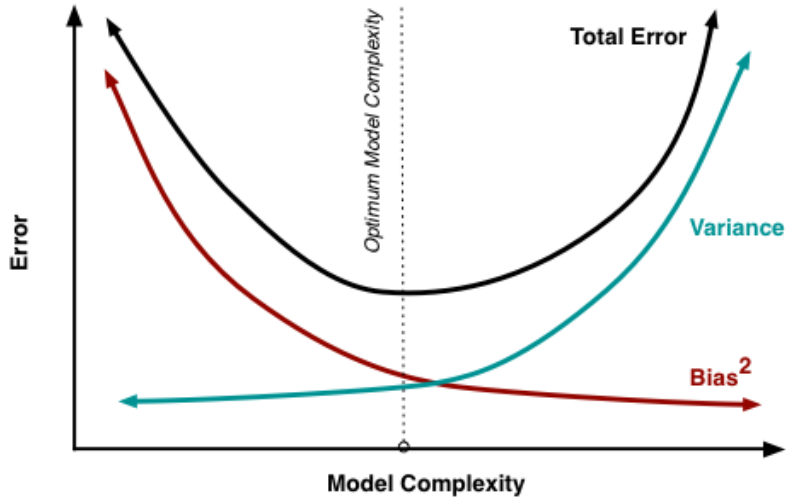
# Decomposition II

We can further decompose the first term:

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2] \\
&= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2]}_{\text{variance}}
\end{aligned}$$

- Bias: the extent to which the average prediction over all data sets differs from the desired regression function.
- Variance: the extent to which the solutions for individual data sets vary around their average

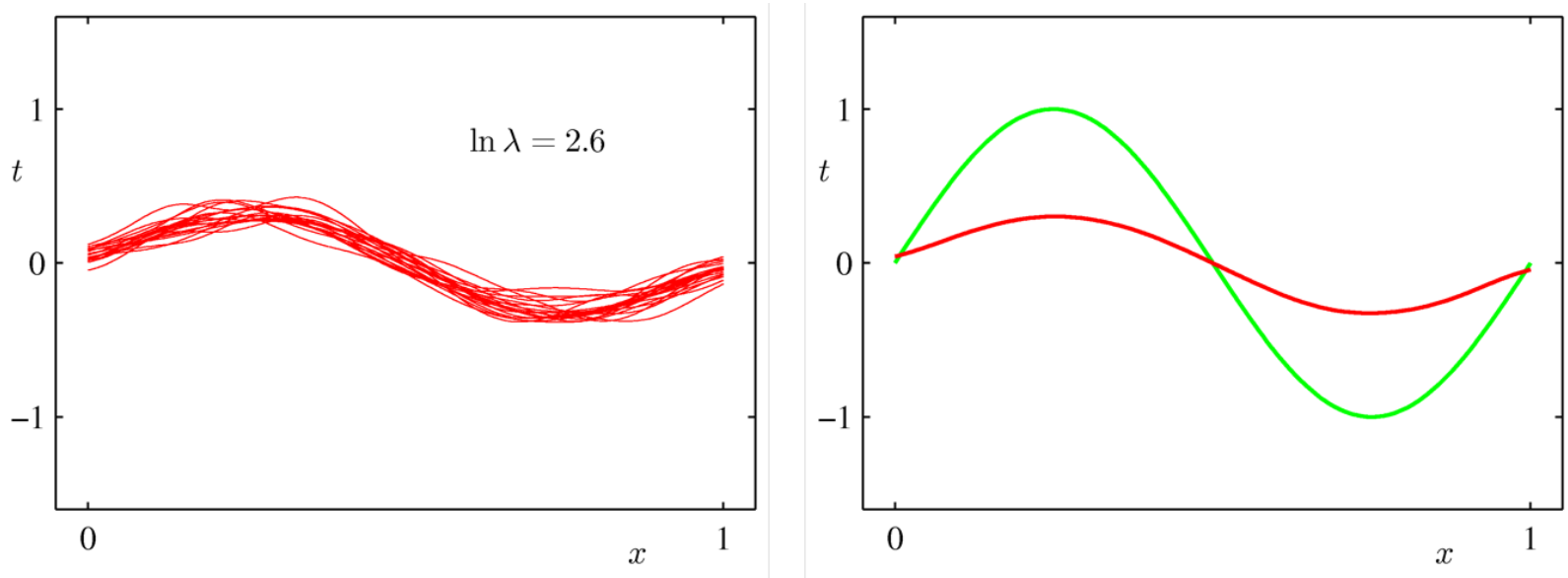
# Decomposition II (cont.)




$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{irreducible error (noise)}$$

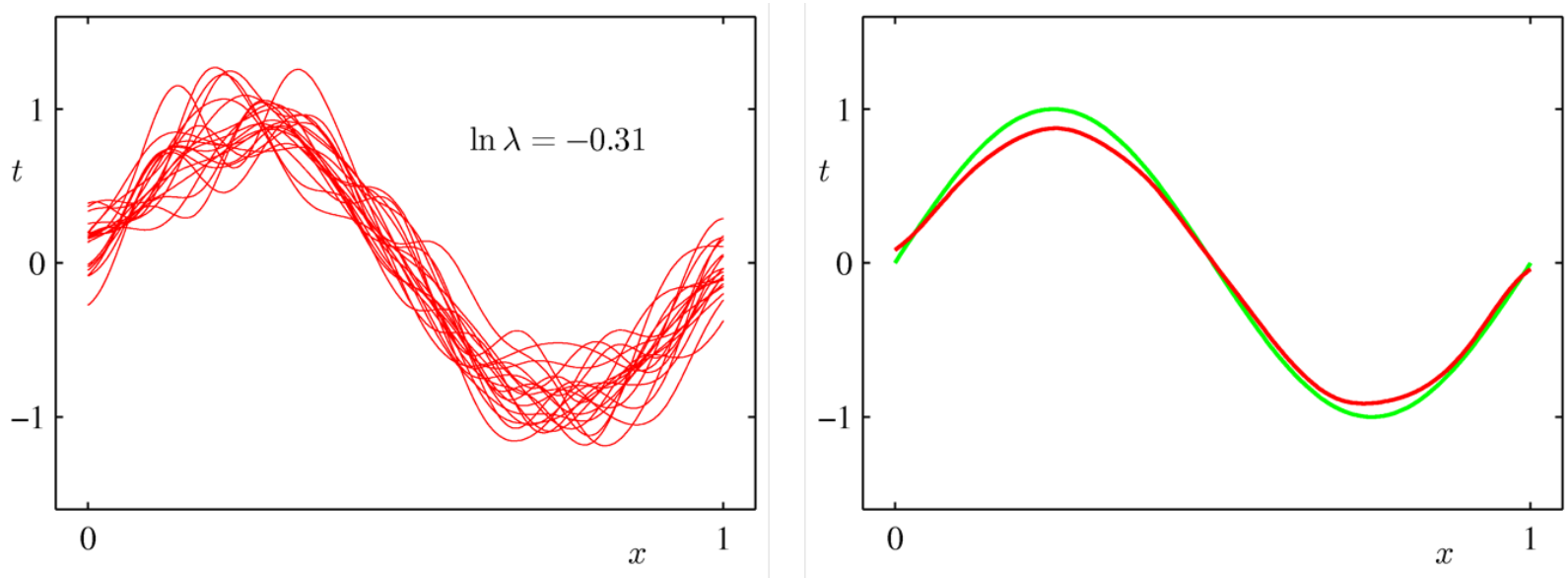
- Flexible models/Strong approximators (e.g., high degree polynomials): low bias and high variance
- Rigid models/Weak approximatiors (e.g., low degree polynomials): high bias and low variance

# Decomposition Example



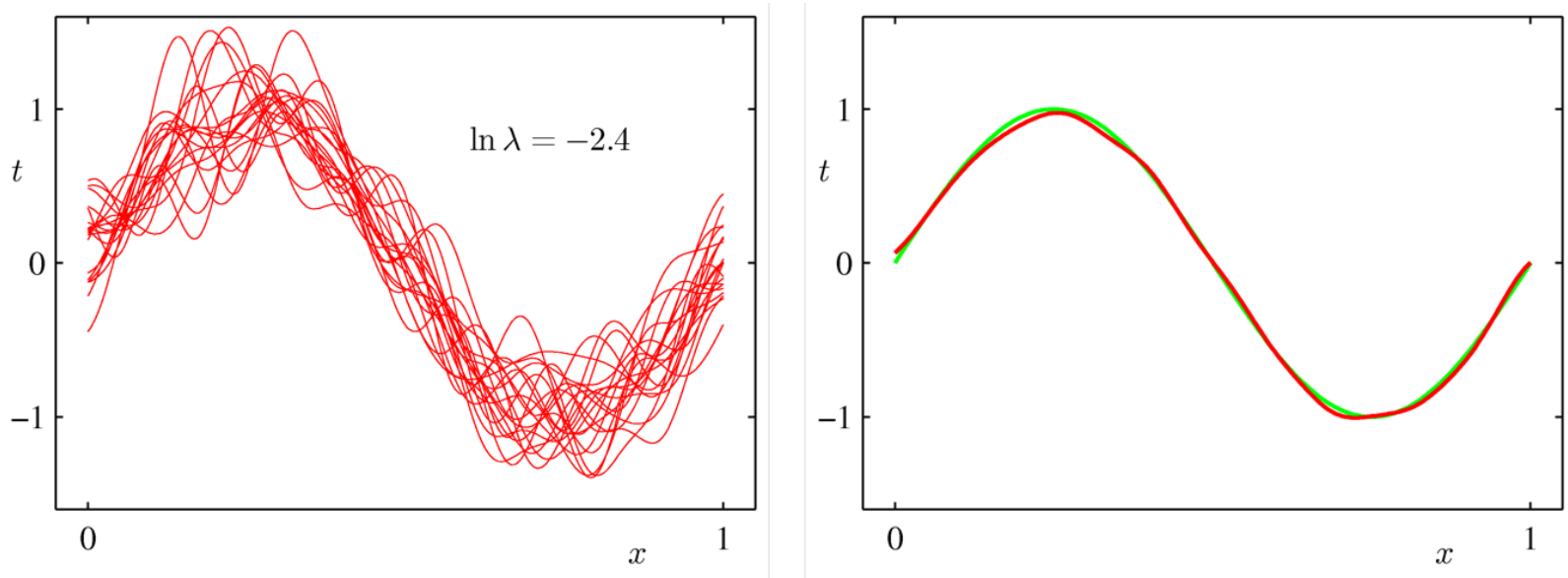

Example: 25 data sets from the sinusoidal, varying the degree of regularization.

# Decomposition Example (cont.)




Example: 25 data sets from the sinusoidal, varying the degree of regularization.

# Decomposition Example (cont.)




Example: 25 data sets from the sinusoidal, varying the degree of regularization.

# Bias-variance trade-off

- We may not learn a good flexible model given limited sample
- We would like to pay a little bias to save a lot of variance
- We can pay in model bias, or estimation bias
  - Feature Selection: More model bias, less parameters to estimate
  - Regularization: More esimation bias, lower variance
- Feature selection can often be formulated as special case of regularization [e.g. $\ell_0$-norm]

# Least Squares Estimator

- Truth $f(x) = x^T \mathbf{w}$, observed $y = f(x) + \epsilon$, $\mathbb{E}[\varepsilon] = 0$
- Least squares estimator $\mathbf{w}^{\text{ls}} = (X^T X)^{-1} X^T y$, $\hat{f}(x_0) = x_0^T \mathbf{w}^{\text{ls}} = x_0^T (X^T X)^{-1} X^T \mathbf{y}$
- Show that the least squares estimator is unbiased, i.e.,$f(x) = \mathbb{E}[\hat{f}(x)]$

# Least Squares Estimator

- Truth $f(x) = x^T \mathbf{w}$, observed $y = f(x) + \epsilon$, $\mathbb{E}[\varepsilon] = 0$
- Least squares estimator $\mathbf{w}^{\text{ls}} = (X^T X)^{-1} X^T y$, $\hat{f}(x_0) = x_0^T \mathbf{w}^{\text{ls}} = x_0^T (X^T X)^{-1} X^T \mathbf{y}$
- Show that the least squares estimator is unbiased, i.e.,$f(x) = \mathbb{E}[\hat{f}(x)]$

$$
\begin{aligned}
& f(x_0) - \mathbb{E}[\hat{f}(x_0)] \\
=& x_0^T \mathbf{w} - \mathbb{E}[x_0^T (X^T X)^{-1} X^T \mathbf{y}] \\
=& x_0^T \mathbf{w} - \mathbb{E}[x_0^T (X^T X)^{-1} X^T (x^T \mathbf{w} + \varepsilon)] \\
=& x_0^T \mathbf{w} - \mathbb{E}[x_0^T \mathbf{w} + x_0 (X^T X)^{-1} X^T \varepsilon] \\
=& x_0^T \mathbf{w} - x_0^T \mathbf{w} + x_0 (X^T X)^{-1} X^T \mathbb{E}[\varepsilon] = 0
\end{aligned}
$$

# Least Squares Estimator (cont.)

- Truth $f(x) = x^T \mathbf{w}$, observed $y = f(x) + \epsilon$, $\mathbb{E}[\varepsilon] = 0$, $\text{var}(\varepsilon) = \sigma^2$
- $E[(\hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)])^2] = \sigma^2 d/N$,
- Variance is $O(d/N)$
- where $d$ the is number of features and $N$ is the number of samples.

# Ridge regression

- Let $R = X^T X$

$$\begin{aligned}
\hat{\mathbf{w}}_\lambda^{\text{ridge}} &= (X^T X + \lambda \mathbf{I}_d)^{-1} X^T \mathbf{y} \\
&= (R + \lambda \mathbf{I}_d)^{-1} R R^{-1} X^T \mathbf{y} \\
&= [R(\mathbf{I}_d + \lambda R^{-1})]^{-1} R[(X^T X)^{-1} X^T \mathbf{y}] \\
&= (\mathbf{I}_d + \lambda R^{-1})^{-1} R^{-1} R \mathbf{w}^{\text{ls}} \\
&= (\mathbf{I}_d + \lambda R^{-1})^{-1} \mathbf{w}^{\text{ls}}
\end{aligned}$$

- Therefore

$$\begin{aligned}
\mathbb{E}(\mathbf{w}_\lambda^{\text{ridge}}) &= \mathbb{E}\{(\mathbf{I}_d + \lambda R^{-1})^{-1} \mathbf{w}^{\text{ls}}\} \\
&= (\mathbf{I}_d + \lambda R^{-1})^{-1} \mathbf{w} \\
&\underbrace{\neq}_{\text{if}(\lambda \neq 0)} \mathbf{w}
\end{aligned}$$

# Effect of Algorithm Parameters on Bias and Variance

- $k$-nearest neighbor:
  - increasing $k$ typically increases bias and reduces variance
- decision trees of depth $D$:
  - increasing $D$ typically increases variance and reduces bias
- RBF SVM with parameter $\sigma$:
  - increasing $\sigma$ typically increases bias and reduces variance
- Matrix Completion rank $r$
  - increasing $r$ typically increases variance and reduces bias
- Multi-task learning
  - increasing task relatedness typically increases bias and reduces variance

Many more...

# Model Selection

- How do we choose algorithm parameters (e.g., $\lambda$ in ridge regression) in algorithms?
- We need a disciplined way of choosing $\lambda$
- Obviously want to choose $\lambda$ that minimizes the mean squared error
- Issue is part of the bigger problem of **model selection**

# Training sets versus test sets

- If we have a good model, it should predict well when we have new data.
- In machine learning terms, we compute our statistical model $\hat{f}(.)$ from the **training set**.
- A good estimator $\hat{f}(.)$ should then perform well on a new, independent set of data.
- We "test" or assess how well $\hat{f}(.)$ performs on the new data, which we call the **test set**.

# Training sets versus test sets

- Ideally, we would separate our available data into both training and test sets
  - Of course, this is not always possible, especially if we have a few observations
- Hope to come up with the best-trained algorithm that will stand up to the test
  - Example: The Netflix contest
- How can we try to find the best-trained algorithm?

# $K$-fold cross validation

- Most common approach is $K$-**fold cross validation**:
  - Partition the training data $\mathcal{D}$ into $K$ separate sets of equal size
    - Suppose $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_K)$
    - Commonly chosen $K$s are $K = 5$ and $K = 10$
  - For each $k = 1, 2, ..., K$, fit the model $\hat{f}_{-k}^{(\lambda)}(\mathbf{x})$ to the training set excluding the $k$th-fold $\mathcal{D}_k$
  - Compute the fitted values for the observations in $\mathcal{D}_k$ , based on the training data that excluded this fold
  - Compute the cross-validation (CV) error for the $k$-th fold:

$$(\text{CV Error})_k^{(\lambda)} = |\mathcal{D}_k|^{-1} \sum_{(\mathbf{x},y)\in\mathcal{D}_k} (y - \hat{f}_{-k}^{(\lambda)}(\mathbf{x}))^2$$

# $K$-fold cross validation

- The model then has overall cross-validation error:

$$(\text{CV Error})^{(\lambda)} = K^{-1} \sum_{k=1}^{K} (\text{CV Error})_k^{(\lambda)}$$

- Select $\lambda^*$ as the one with minimum $(\text{CV Error})^{(\lambda)}$
- Compute the chosen model $\hat{f}_{-k}^{(\lambda^*)}(\mathbf{x})$ on the entire training set $T = (\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k)$
- Apply $\hat{f}_{-k}^{(\lambda^*)}(\mathbf{x})$ to the test set to assess test error

# $K$-fold cross validation

- Remark: Our data set might be small, so we might not have enough observations to put aside a test set:
  - In this case, let all of the available data be our training set
  - Still apply $K$-fold cross validation
  - Still choose $\lambda^*$ as the minimizer of CV error
  - Then refit the model with $\lambda^*$ on the entire training set

# Leave-one-out CV

- What happens when $K = |\mathcal{D}|$?
- This is called **leave-one-out cross validation**
- Widely used in many applications with limited amount of data.