Overfitting, Underfitting and Cross-Validation

Prof. Mingkui Tan

South China University of Technology
Southern Artificial Intelligence Laboratory(SAIL)

Content

- Problem of Model Validation
  - Underfitting and Overfitting
  - Bias-Variance Tradeoff
- Cross-Validation
  - Training Data, Validation Data, Testing Data
  - Performance Report
  - Parameter Tuning
  - K-Fold Cross-Validation

Content

- Problem of Model Validation
    - Underfitting and Overfitting
    - Bias-Variance Tradeoff
- Cross-Validation
    - Training Data, Validation Data, Testing Data
    - Performance Report
    - Parameter Tuning
    - K-Fold Cross-Validation

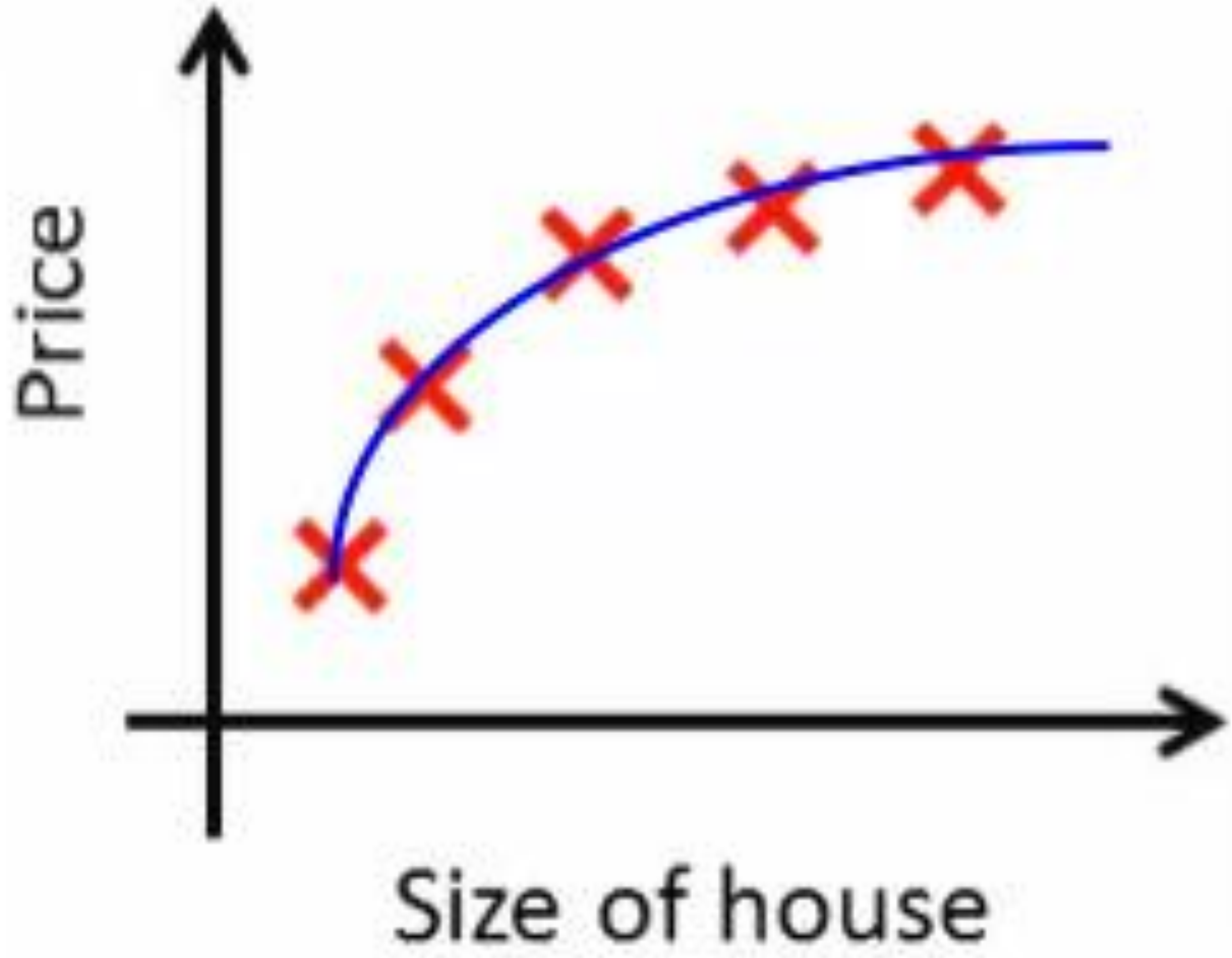
Why We All Need Validation

1. Business Reasons
   - Need to choose the best model.
   - Measure accuracy/power of selected model.
   - Good to measure ROI of the modeling project.
2. Statistical Reasons
   - Model building techniques are inherently designed to minimize “loss” or “bias”.
   - To an extent, a model will always fit “noise” as well as “signal”.
   - ➔ If you just fit a bunch of models on a given dataset and choose the “best” one, it will likely be overly “optimistic”.

Some Definitions

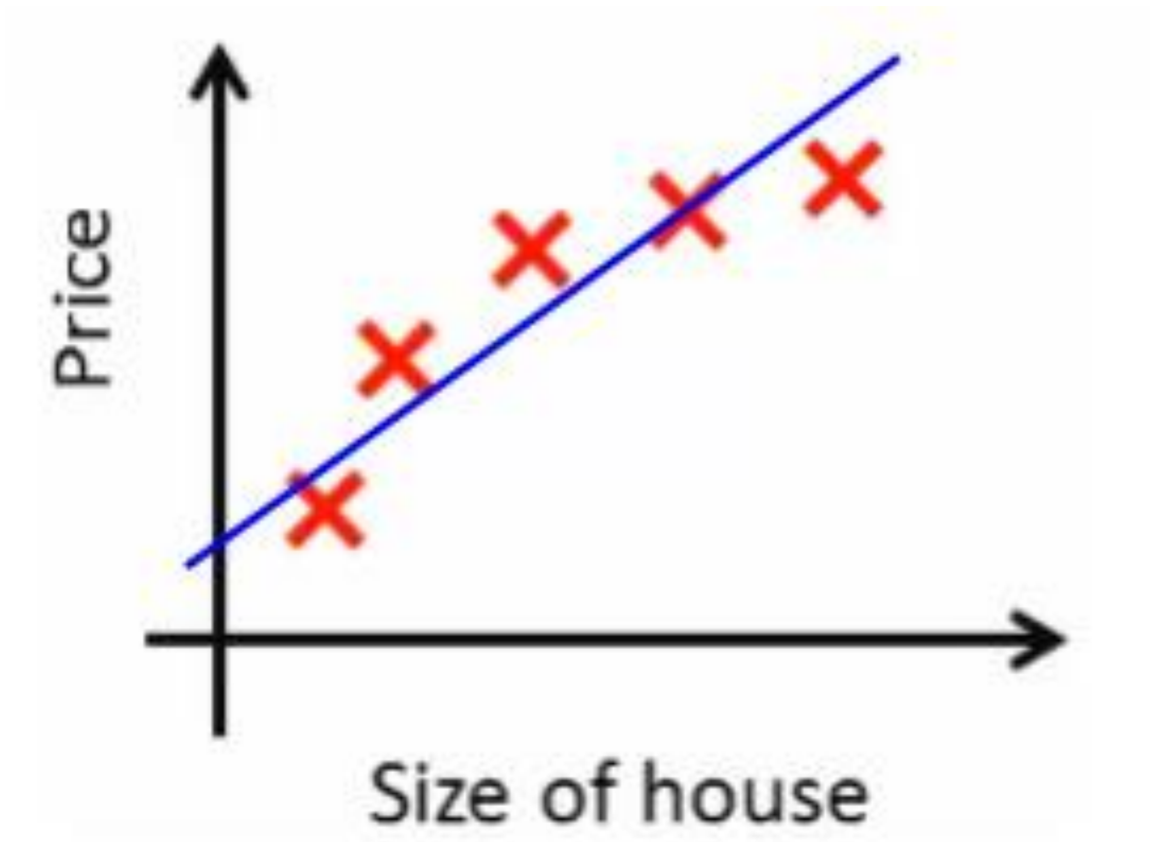
- Target Variable: Y
  - What we are trying to predict.
    - House price,…
- Input Variables: $\{X_1, X_2, \ldots, X_N\}$
  - Input features used to make predictions.
    - Size of house, ….
- Predictive Model: $Y = f(X_1, X_2, \ldots, X_N)$
  - Estimates the unknown value Y based on known values $\{X_i\}$.

General Fitting Scheme



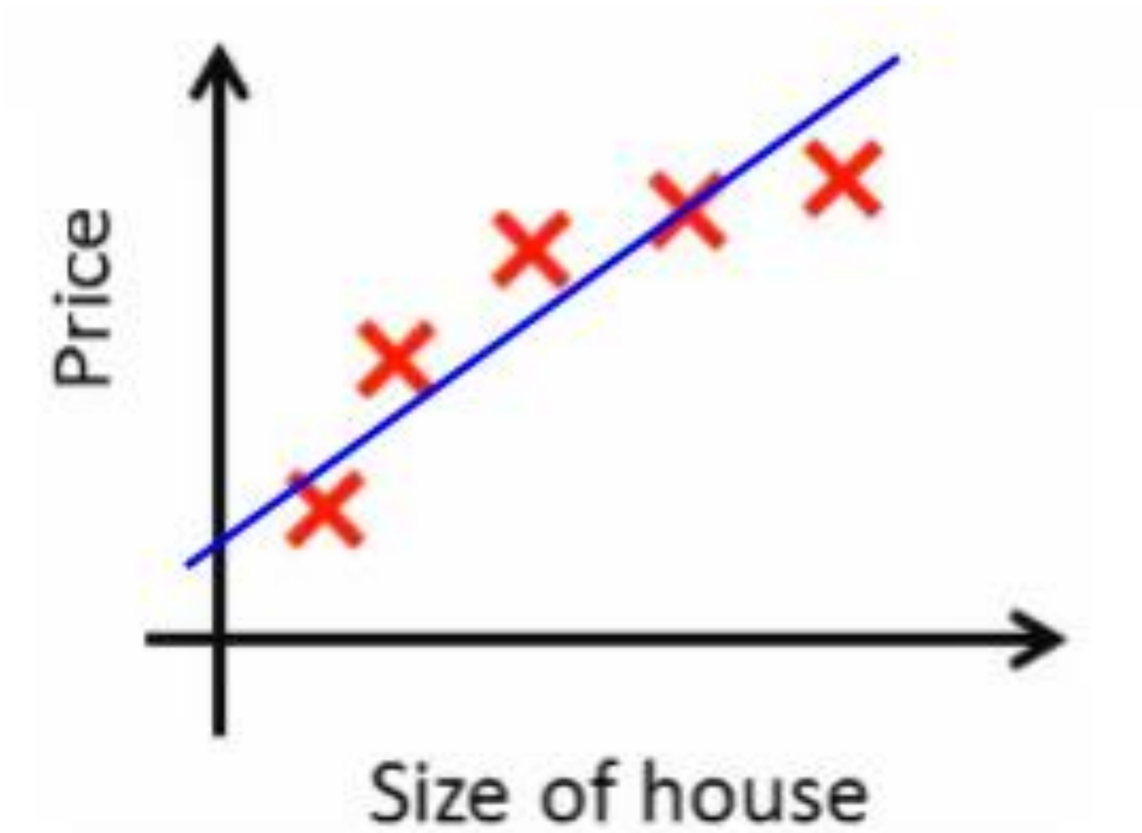
Underfitting

- Model cannot capture the underlying trend of the data



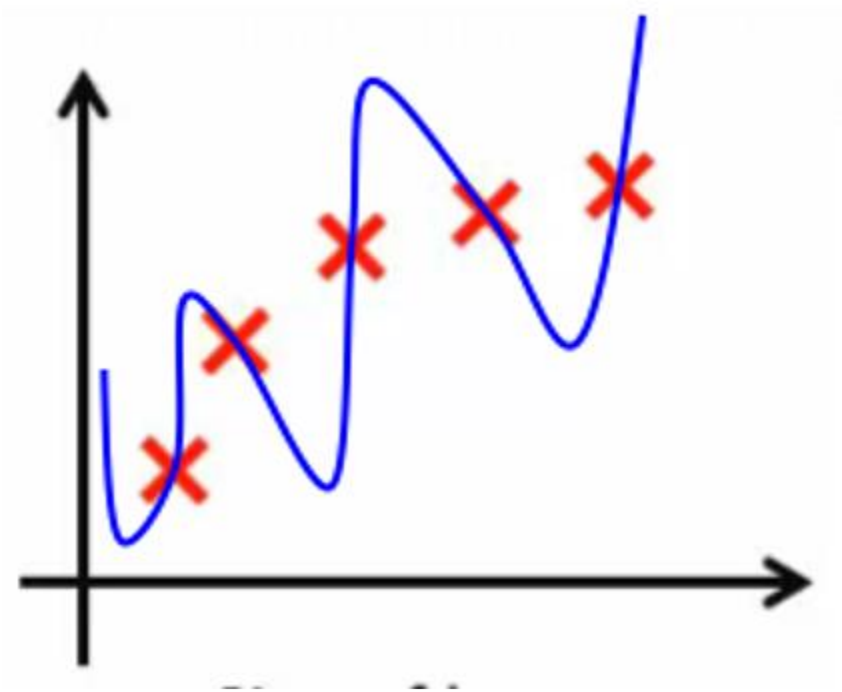
Solution to Underfitting

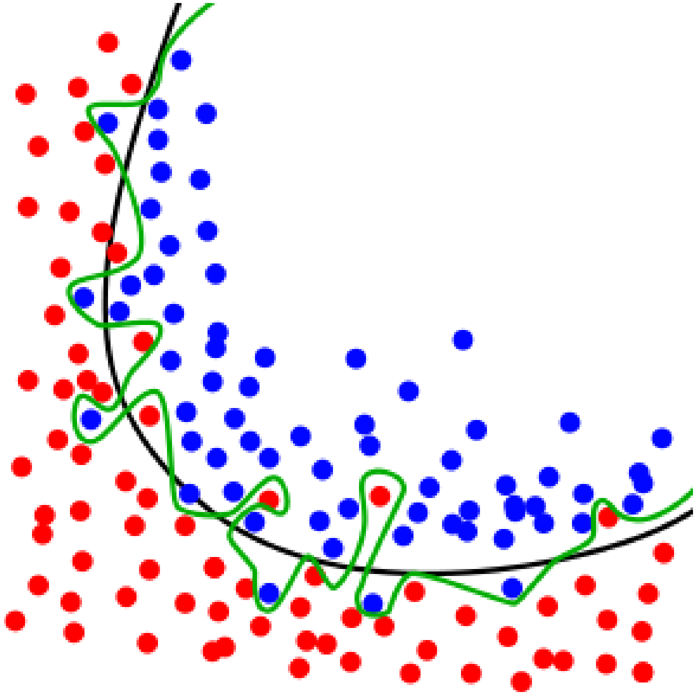
- Model cannot capture the underlying trend of the data



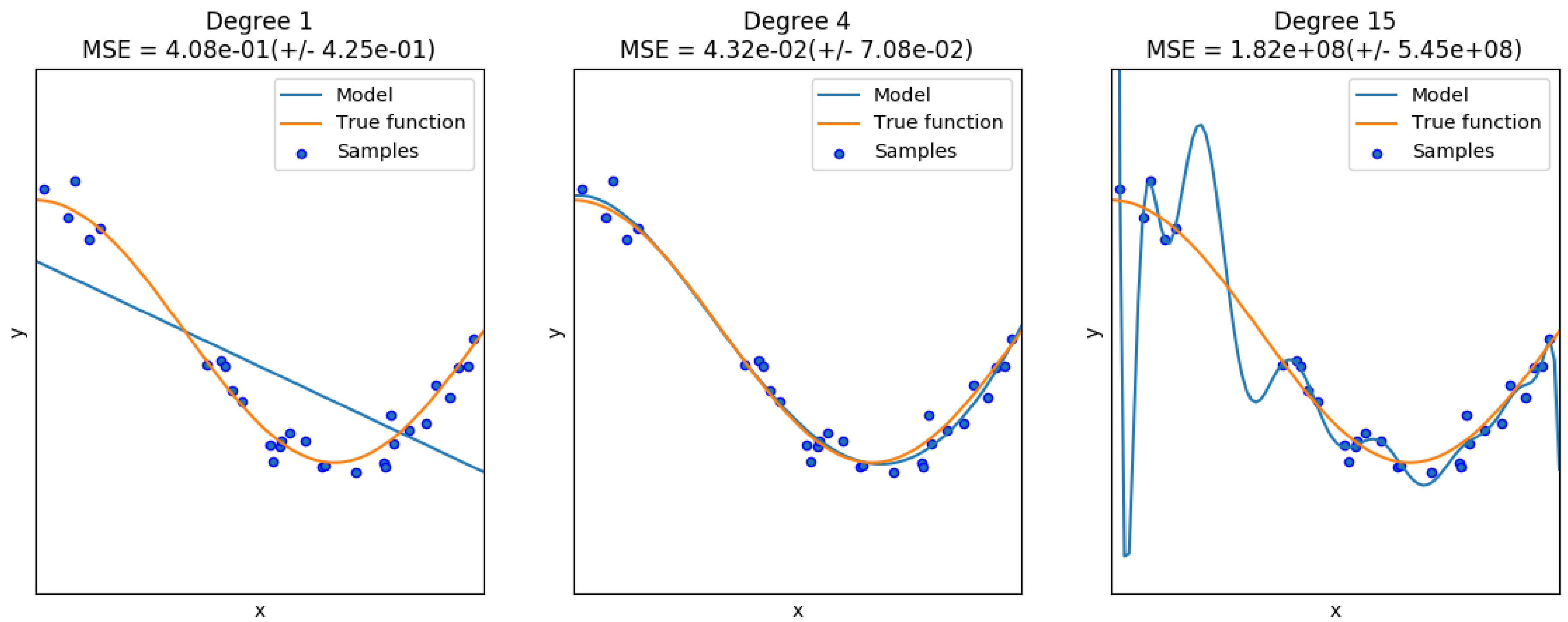
The Problem of Overfitting

- Model is too complex to capture the true pattern
- Model seeks to fit the noise or outlier of the data

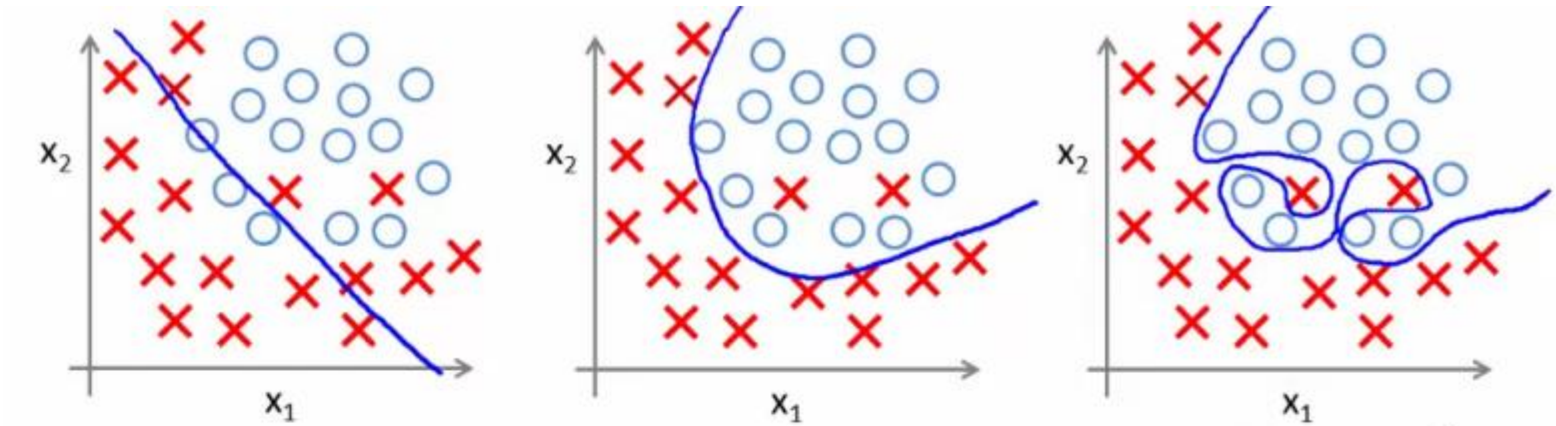




Underfitting vs Overfitting



Underfitting vs Overfitting



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2$
$+\theta_3 x_1^2 + \theta_4 x_2^2$
$+\theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$
$+\theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2$
$+\theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots$

UNDERFITTING
(high bias)

OVERFITTING
(high variance)

The Perils of Optimism

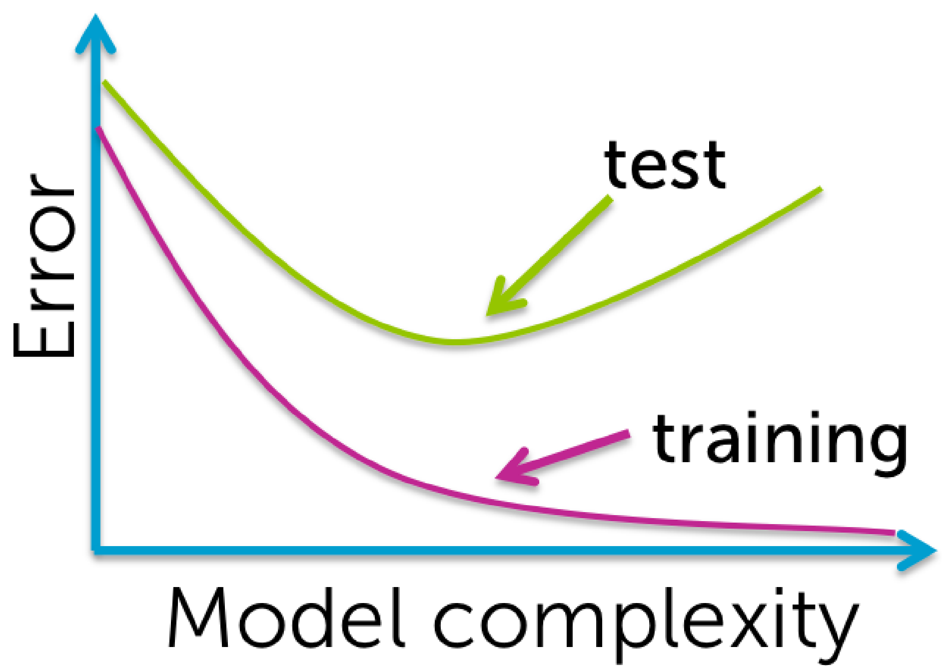
- Error on the dataset used to *fit* the model can be misleading
    - Doesn’t predict future performance.
- Too much complexity can diminish model’s accuracy on future data.
    - Sometimes called the Bias-Variance Tradeoff.

The Bias-Variance Tradeoff

- Complex model:
  - Low “bias”:
    - The model fit is good on the *training data*.
    - The model value is close to the data’s expected value.
  - High “Variance”:
    - Model more likely to make a wrong prediction.

Signs of Underfitting/Overfitting

- **How do we know if we are underfitting or overfitting?**
  - If by increasing capacity we decrease generalization error, then we are underfitting, otherwise we are overfitting.
  - If the error in representing the training set is relatively large and the generalization error is large, then underfitting;



Signs of Underfitting/Overfitting

- **Need to increase capacity (complexity of models)**.

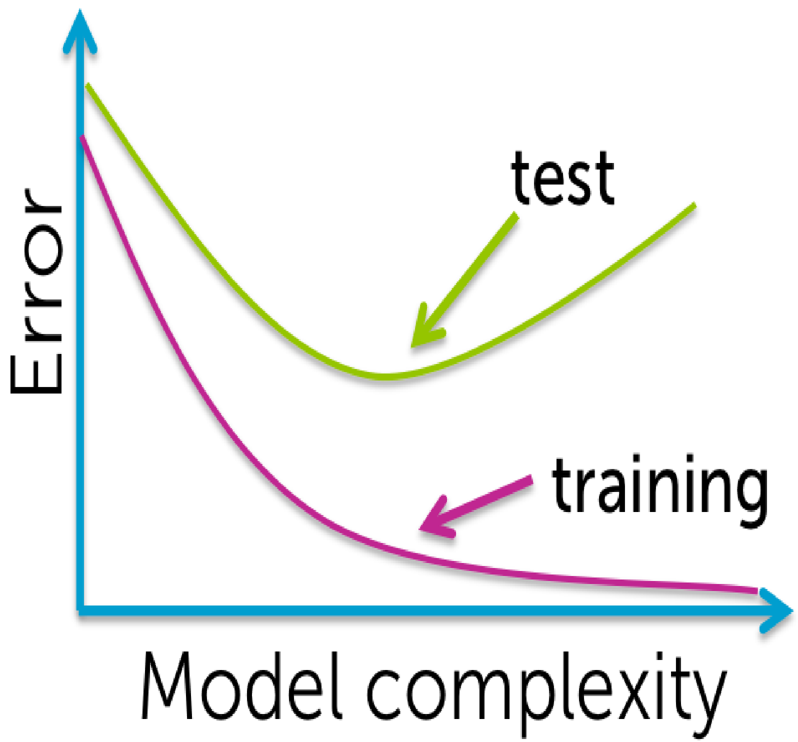
  – If the error in representing the training set is relatively small and the generalization error is large, then overfitting;

- **Need to decrease capacity or increase training set**.

  – Many features and relatively small training set.

  – if you have chosen a large capacity to complement the many features, then you might overfit the data:

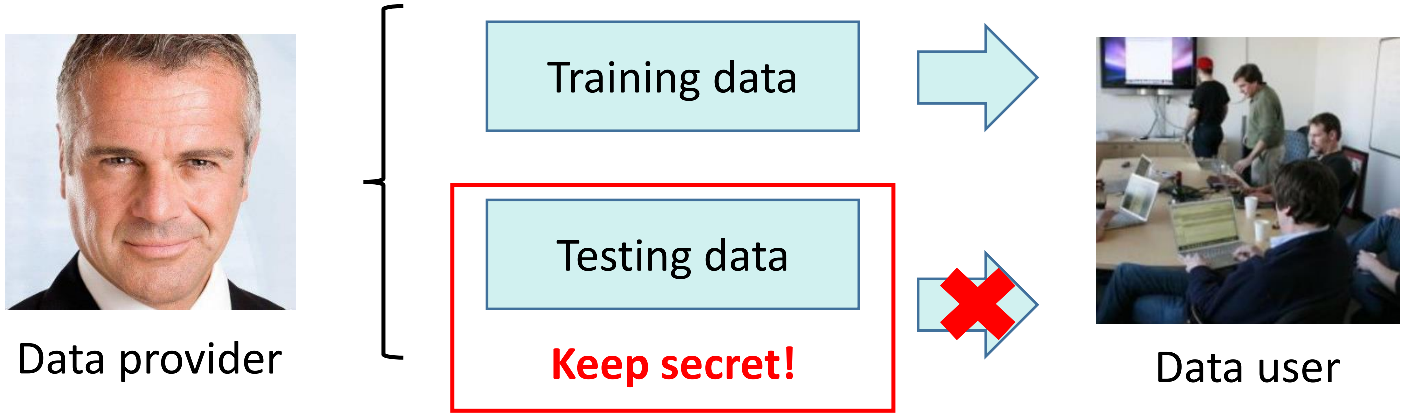
- **need to decrease the capacity.**



Content

- Problem of Model Validation
  - Underfitting and Overfitting
  - Bias-Variance Tradeoff
- Cross-Validation
  - Training Data, Validation Data, Testing Data
  - Performance Report
  - Parameter Tuning
  - K-Fold Cross-Validation

Data Split



- Simplest idea: Divide data into 2 pieces.
  - Training data: data used to fit model
  - Test data: “fresh” data used to evaluate model
- Test data contains:
  - Actual target value *Y*
  - Model prediction *Y**
- We can find clever ways of displaying the relation between *Y* and *Y**.
  - Lift curves, gains charts, ROC curves ……

Testing Errors

How you can tell that a hypothesis overfits?

- Plotting-not always good

We can split all data into 2 subsets

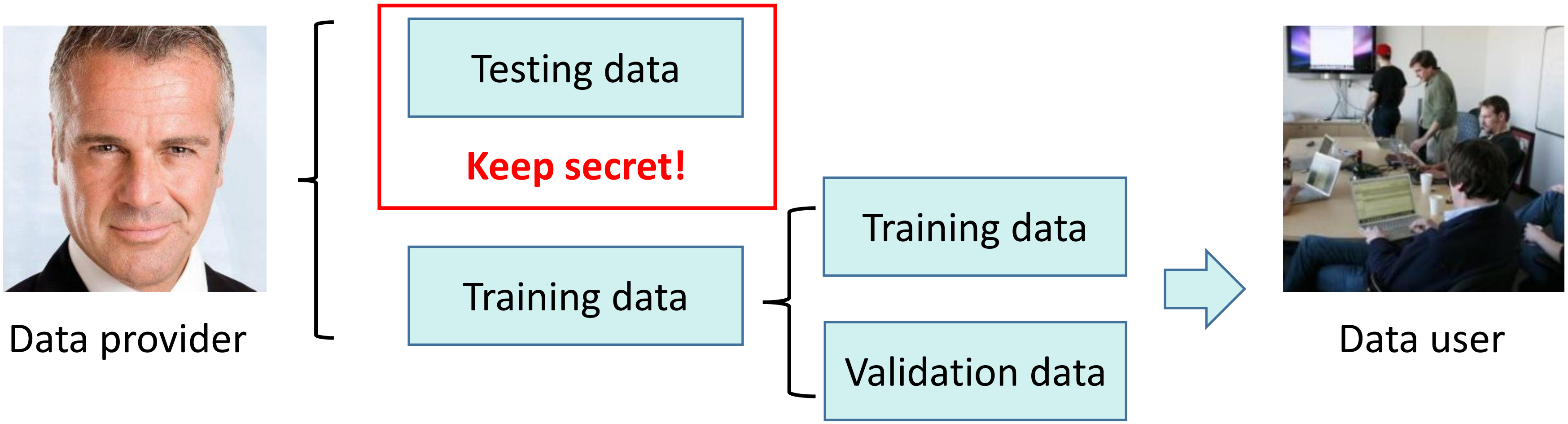
- Training set ≈ 70% of data, m-number of examples in the training set
- Testing set ≈ 30% of data, m_{test}-number of examples in the testing set

It's better to choose examples for training /testing sets randomly

Error Metrics

	Prediction	**Classification**
Example Model	Linear Regression	Logistic Regression
Test Error	$J_{\text{test}}(\theta) = \frac{1}{m_{\text{test}}} \sum \text{error}\left(h_\theta(x^{(i)}_{\text{test}}), y^{(i)}_{\text{test}}\right)$	
$\text{error}(h_\theta(x), y)$	Average Square Error $\text{error}(h_\theta(x), y) = \frac{1}{2}(h_\theta(x) - y)^2$	Misclassification Error $\text{error}(h_\theta(x), y) = \begin{cases} 0 \text{ if classification is correct} \\ 1 \text{ otherwise} \end{cases}$

Validation



Generally cross-validation is used to find the best value of some parameter

- We still have training and testing sets
- But additionally we have a cross-validation set to test the performance of our model depending on the parameter

Cross-Validation for Evaluation

- Instead of splitting the data set into 2 categories, we split into 3 sets:
- Training set (≈60%)
  - $\boldsymbol{x}^{(i)}, y^{(i)}$, total m examples
- Cross-validation set(or cv, ≈20%)
  - $\boldsymbol{x}_{cv}^{(i)}, y_{cv}^{(i)}$, total m_{cv} examples
- Test set(≈20%)
  - $\boldsymbol{x}_{\text{test}}^{(i)}, y_{test}^{(i)}$, total m_{test} examples

Cross-Validation for Evaluation

- Training error

$$J_{train}(\boldsymbol{\theta}) = \frac{1}{2m}\sum cost(\boldsymbol{x}^{(i)}, y^{(i)})$$

- Cross-Validation error

$$J_{cv}(\boldsymbol{\theta}) = \frac{1}{2m_{cv}}\sum cost(\boldsymbol{x}_{cv}^{(i)}, y_{cv}^{(i)})$$

- Test Error

$$J_{test}(\boldsymbol{\theta}) = \frac{1}{2m_{test}}\sum cost(\boldsymbol{x}_{test}^{(i)}, y_{test}^{(i)})$$

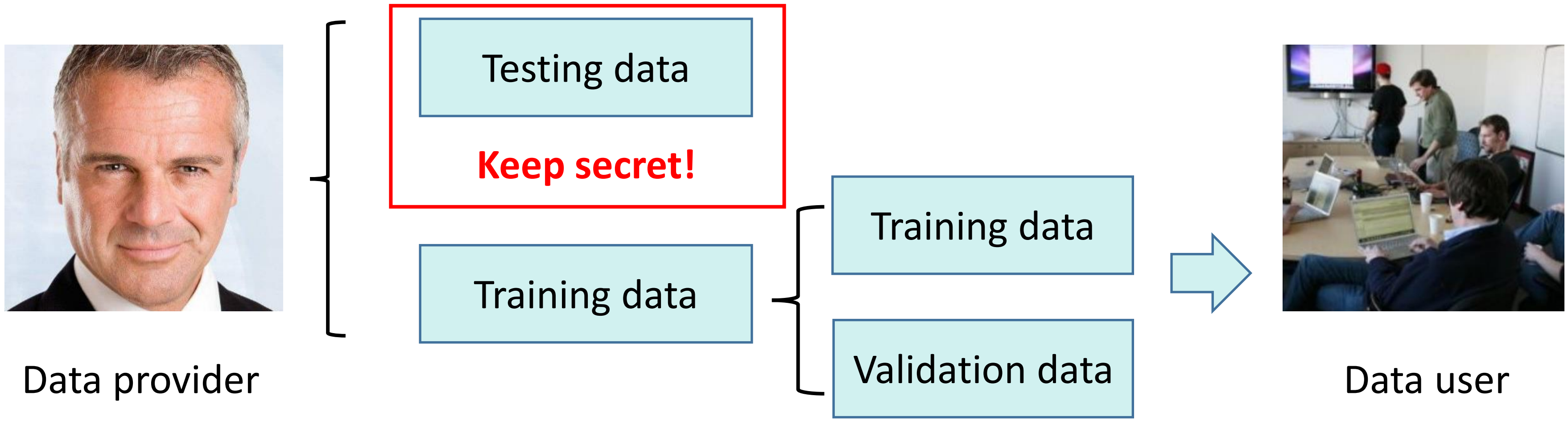
For model selection, we

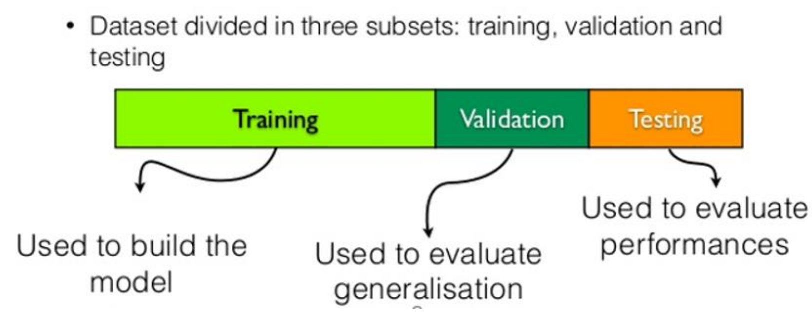
- Obtain $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(d)}$ and select best (lowest) $J_{cv}(\boldsymbol{\theta}^{(i)})$

For final evaluation, we

- Evaluate generalization error $J_{test}(\boldsymbol{\theta}^{(i)})$ on testing set

Tuning Learning Parameter





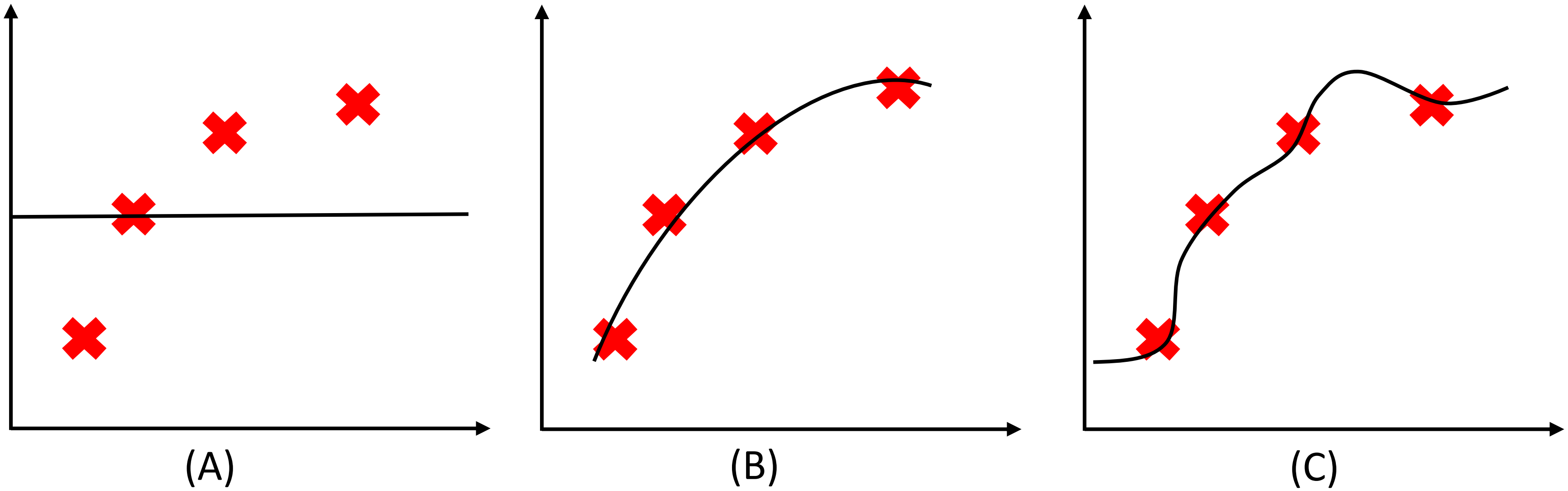
- ➢ Split data set into
  - Learning set
  - Validation set
  - Test set
- ➢ Use validation set for tuning hyper-parameters
- ➢ Use testing set only for final evaluation

Example: Tuning Regularization Parameter λ

Suppose now we're fitting a model with high-order polynomial

- $h_{\boldsymbol{\theta}}(\boldsymbol{x})=\theta_0 + \theta_1 x + \cdots + \theta_4 x^4$
- To prevent overfitting we use regularization
- $J(\boldsymbol{\theta})=\frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(\boldsymbol{x}_i), y_i)+\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

Regularization Parameter λ

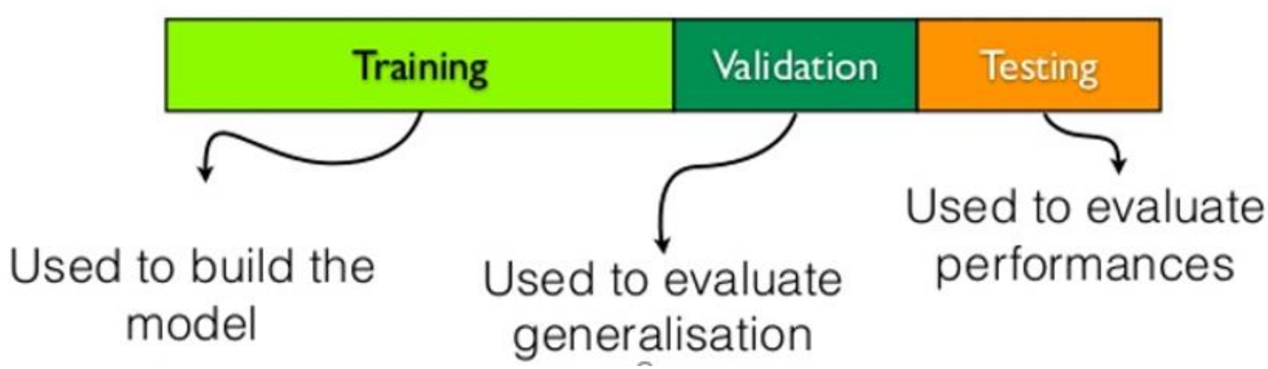


- (a)If λ is large (say λ =10000), all $\boldsymbol{\theta}$ are penalized and $\theta_1 \approx \theta_2 \approx \cdots \approx 0$, $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx \theta_0$.
- (b)If λ is intermediate, we fit well.
- (c)If λ is small(close to 0), we fit too well, i.e. we overfit.

Chose good λ on validation set

- Choose a range of possible values for λ (0.02, 0.04,...,0.24)
- That gives us 12 models to checks
- For each λ_i
  - Calculate $\boldsymbol{\theta}_i$
  - Calculate $J_{cv}(\boldsymbol{\theta}_i)$
  - And take λ_i with lowest $J_{cv}(\boldsymbol{\theta}_i)$
- Finally, we report the test error as $J_{test}(\boldsymbol{\theta}_i)$

  - Dataset divided in three subsets: training, validation and testing

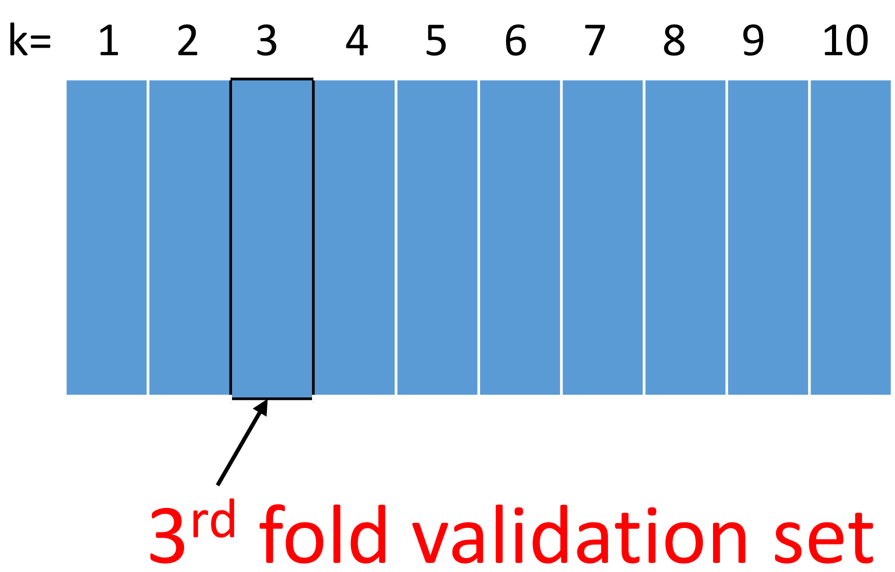


K-Fold Cross-Validation

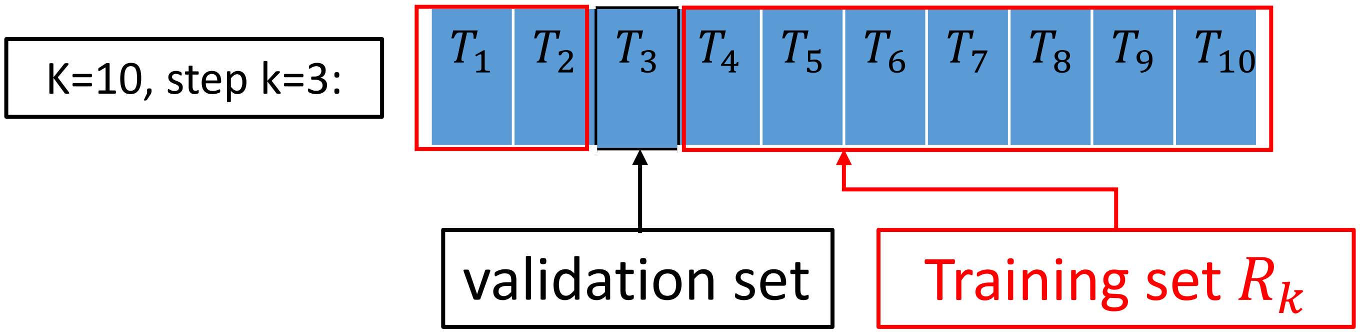
If we want to reduce variability in the data

- We can use multiple rounds of cross-validation using different partitions
- And then average the result over all rounds

We're given a dataset S sampled from the population *D*



K-Fold Cross-Validation



- Partition data S into K equal disjoint subsets (T_1,…, T_k)(typically 5-10 subsets)
- Perform following K steps for k=1…K
  - Use R_k=S-T_k as the training set
  - Build classifier C_k using R_k
  - Use T_k as the validation set, compute error Err_k=*error*(C_k, T_k)
- Let $Err^{ave}=\frac{1}{K}\sum_{k=1}^{K} Err_k$
  - This is the averaged error rate

Tuning Learning Parameter

Choosing best parameter λ with K-Fold Cross-Validation

- Split your data into training set and validation set
- For every possible value λ, estimate the error rate
- Select λ with least average error rate Err^{ave}

Final evaluation on testing set