COMP 5630/6630:Machine Learning

Lecture 5: Model Selection, Cross Validation, Bias Variance Tradeoff

Slide Adapted: Drs. Aakur , Andrew Ng

Predicting House Price: Generalize to Unseen Data

Dataset of the living areas, bedrooms, and prices of 47 houses

Living area (ft^2)	# bedrooms	Price (1000$s)
1643	4	256
1356	3	202
1678	3	287
…	…	…
3000	4	400

- Predicting house price with a regularized linear regression
- Model does not perform well on new, unseen data

What to do?

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

Predicting House Price: Generalize to Unseen Data

Dataset of the living areas, bedrooms, and prices of 47 houses

Living area (ft^2)	# bedrooms	Price (1000$s)
1643	4	256
1356	3	202
1678	3	287
…	…	…
3000	4	400

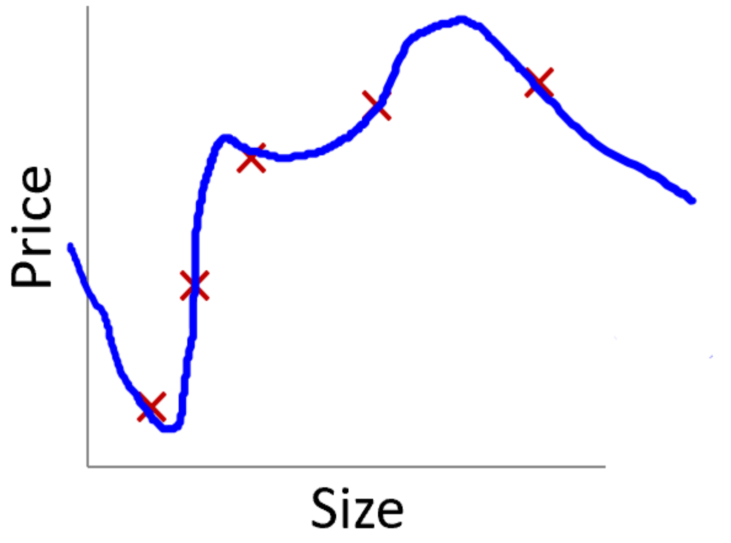
$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- Predicting house price with a regularized linear regression
- Model does not perform well on new, unseen data

What to do?

1. More training examples?
2. More features?
3. Less features?
4. Add polynomial features (e.g., sqrt(living area))
5. Smaller value of λ?
6. Large value of λ?

Evaluating Hypothesis



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Does not generalize to new examples not in training set.

$x_1 =$ size of house

$x_2 =$ no. of bedrooms

$x_3 =$ avg. income in the neighborhood

$x_4 =$ age of house

Evaluating Hypothesis: Dataset

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

Training Data, 70%

$$\longrightarrow \begin{array}{c} (x^{(1)}, y^{(1)}) \\ (x^{(2)}, y^{(2)}) \\ \vdots \\ (x^{(m)}, y^{(m)}) \end{array}$$

Test Data, 30%

$$\longrightarrow \begin{array}{c} (x_{test}^{(1)}, y_{test}^{(1)}) \\ (x_{test}^{(2)}, y_{test}^{(2)}) \\ \vdots \\ (x_{test}^{(m_{test})}, y_{test}^{(m_{test})}) \end{array}$$

Training/testing Procedure: Linear Regression

- Learn parameter θ from training data by minimizing $J_{\text{training}}(\Theta)$

$$J_{\text{training}}(\Theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- Compute test set error:

$$J_{\text{test}}(\Theta) = \frac{1}{2mtest}\sum_{i=1}^{mtest}(h_\Theta(x_{test}{}^{(i)}) - y_{test}{}^{(i)})^2$$

Training/testing Procedure: Logistic Regression

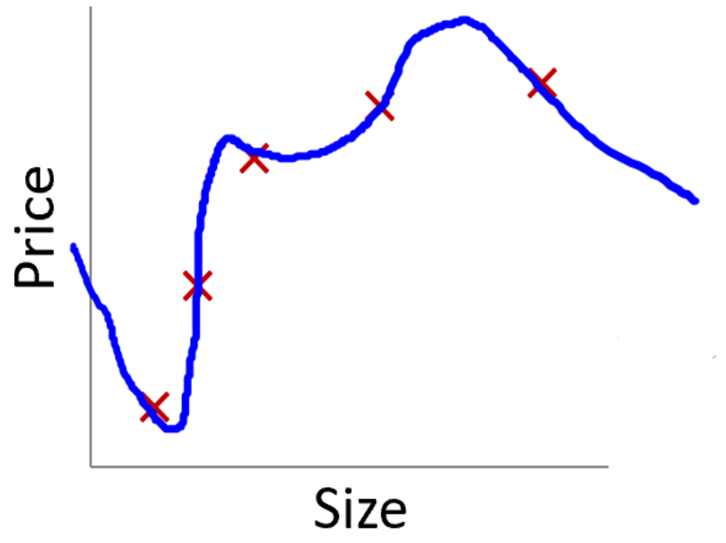
- Learn parameter θ from training data

- Compute test set error:

$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log h_\theta(x_{test}^{(i)})$$

- Misclassification error (0/1 misclassification error)

Overfitting



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

- Once the model is trained, parameters $\theta_0, \theta_1, \ldots, \theta_4$ are fit to the training dataset
- The training error, $J_{\text{training}}(\Theta)$ will likely be much lower than the generalization error
- *How well does the model generalize?*

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ avg. income in the neighborhood
$x_4 =$ age of house

Model Selection: Using Data

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

$\vdots$

10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

Choose $\theta_0 + \ldots \theta_5 x^5$

Measure generalization of model: Report test set error $J_{test}(\theta^{(5)})$

Problem: Performance on training set is not a good indicator of generalization on test set.

Model Selection Using Data: Validation Set

- Divide training set into *training* and *validation* set.
- Now we have split data into three parts: training, validation, and test set

  - Training step
    - use parts of the data to train a range of models
      - Or a given model with a range of values for its parameters
    - Compare them on an independent set, called validation set

- Estimate model’s performance on test set

Evaluating Hypothesis: Dataset

Size	Price
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

Training Data, 60%

$$(x^{(1)}, y^{(1)})$$
$$(x^{(2)}, y^{(2)})$$
$$\vdots$$
$$(x^{(m)}, y^{(m)})$$

Validation Data, 20%

$$(x_{cv}^{(1)}, y_{cv}^{(1)})$$
$$(x_{cv}^{(2)}, y_{cv}^{(2)})$$
$$\vdots$$
$$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$$

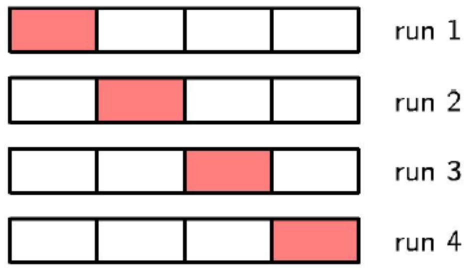
Test Data, 20%

$$(x_{test}^{(1)}, y_{test}^{(1)})$$
$$(x_{test}^{(2)}, y_{test}^{(2)})$$
$$\vdots$$
$$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$$

K-fold Cross Validation

- Limited data
- All available data is partitioned into K non-overlapping groups
- K-1 groups are used to train and evaluated on remaining group (test)
- Repeat for all K choices of held-out group
- Performance scores from K runs are averaged

K=4



If K=N, then it is called the leave-one-out cross validation

Disadvantage of Cross-Validation

- No. of training runs is increased by factor of K
- Problematic if training itself is expensive
- Different data sets can yield different complexity parameters for a single model
- Combinations of parameters are exponential

Bias-Variance Tradeoff

Definitions

- The *generalization* of a machine learning algorithm:
  - Performance (classification, regression, density estimation) on test data, not used for training, but drawn from the same (joint) distribution as the training data.
  - Often, our goal is to get good generalization.
- *Overfitting* or model *variance*:
  - When our model is too complex for the amount of training data we have, it memorizes parts of the noise as well as learning the true problem structure. This is called *overfitting or model variance*.
- *Underfitting or* model *bias*
  - When our model is not complex enough, it cannot capture the structure in our data, no matter how much data we give it. This is called *underfitting or model bias*.

What is Bias and Variance?

- The ***bias error*** is an error from ***erroneous assumptions*** in the learning algorithm. ***High bias*** can cause an algorithm to miss the relevant relations between features and target outputs (***underfitting***).

- The ***variance*** is an error from ***sensitivity to small fluctuations*** in the training set. ***High variance*** can cause an algorithm to ***model*** the ***random noise*** in the training data, rather than the intended outputs (***overfitting***).

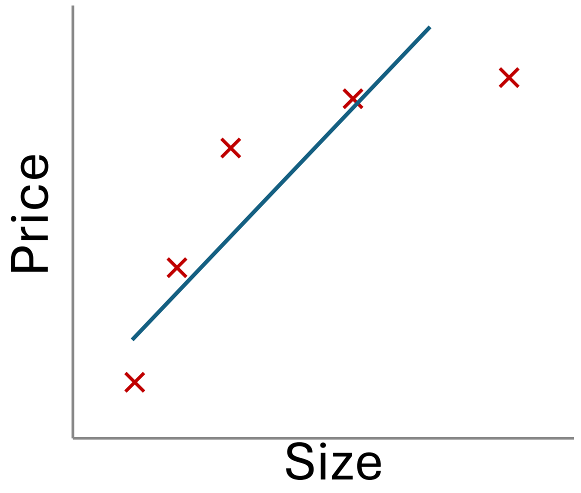
What are we trying to do?

- What basic problems are we trying to avoid with model selection?

- *Overfitting*:
    - if we chose a model that is too complex, it will overfit to the noise in our training set. Another way of saying this is that the machine we end up with is very sensitive to the particular training sample we use. The model has a lot of variance across training samples of a fixed size.
- *Underfitting*:
    - if we chose a model that is not complex enough, it cannot fit the true structure, and so no matter what training sample we use there is some error between the true function and our model approximation. The model has a lot of bias.

- Intuitively, we need the right balance. How can we formalize this?

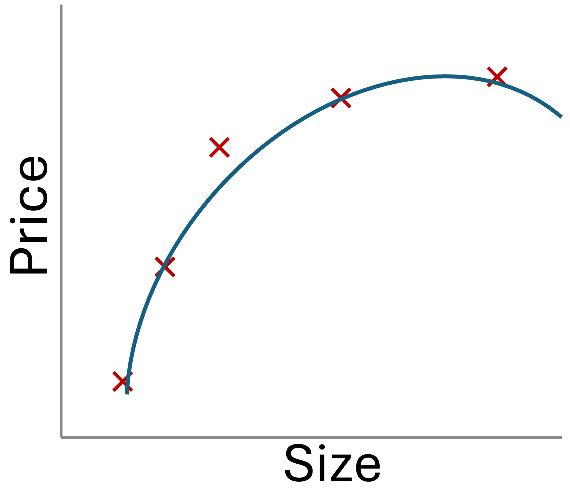
Example: Linear regression (housing prices)



$$\theta_0 + \theta_1 x$$

Underfit

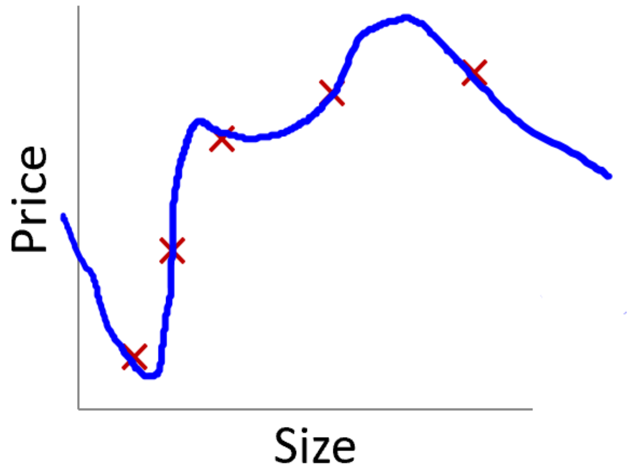
degree of polynomial d = 1



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

Just right

degree of polynomial d = 2



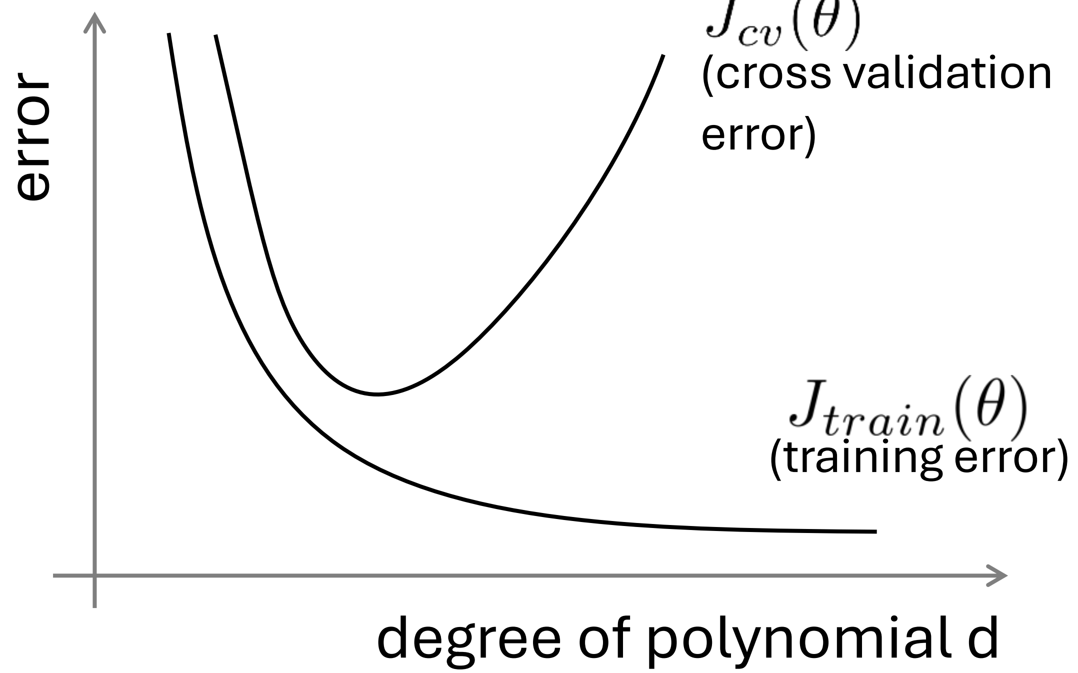
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Overfit

degree of polynomial d = 4

Diagnosing bias vs. variance

Suppose your learning algorithm is performing less well than you were hoping. ($J_{cv}(\theta)$ or $J_{test}(\theta)$ is high.) Is it a bias problem or a variance problem?

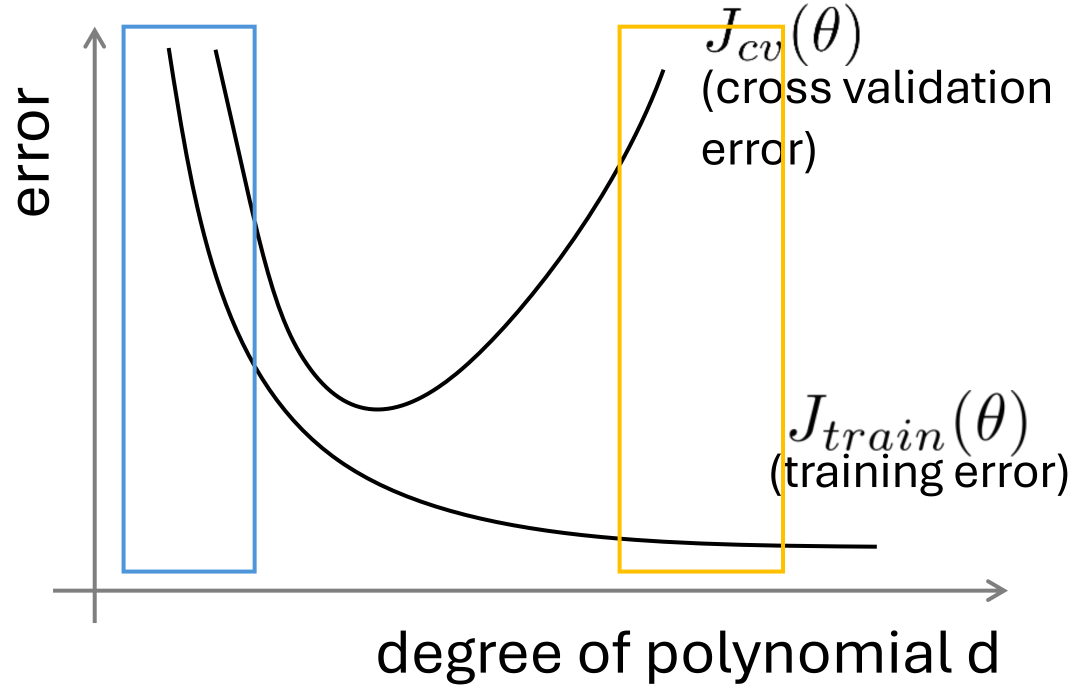


Bias (underfit):

Variance (overfit):

Diagnosing Bias vs. Variance

Suppose your learning algorithm is performing less well than you were hoping. ($J_{cv}(\theta)$ or $J_{test}(\theta)$ is high.) Is it a bias problem or a variance problem?



Bias (underfit):

$J_{train}(\theta)$ will be high

$J_{cv}(\theta) \approx J_{train}(\theta)$

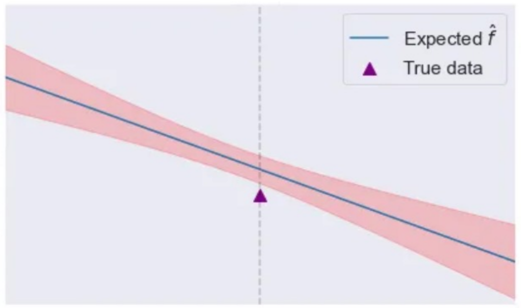
Variance (overfit):

$J_{train}(\theta)$ will be low

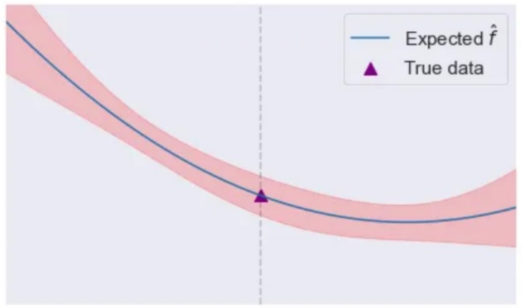
$J_{cv}(\theta)$ will be high

Bias and Variance Example

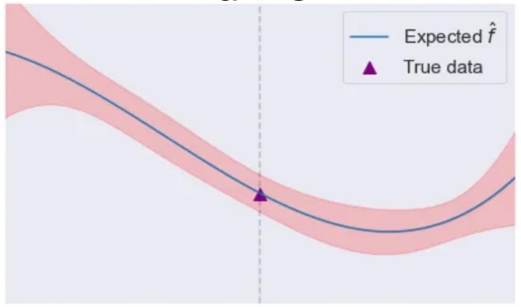
d = 1



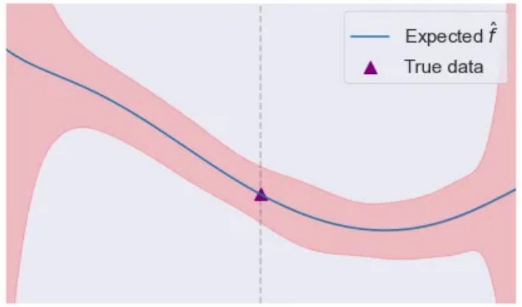
d = 2



d = 3



d = 7



Bias is the “distance” between the true data (triangle) and the expected value (blue line). Variance is how “wide” the different predictions are (red band).

d = Degree of polynomial

Bias and variance of each model?

Example Credit: https://towardsdatascience.com/a-visual-understanding-of-bias-and-variance-66179f16be32

Recap: Model Selection & Performance Estimation

- Model selection: Given a set of models, choose the one which will perform best on new, unseen dataset
- Model Assessment: Given the selected model, estimate its generalization error on new, unseen data
- If we have enough data, the above two problems can be solved by splitting data into three parts
  - Training data: use to train model
  - Validation data: measure performance of each trained model to select the best model
  - Test data: used only once, on the selected model
  - Typical split: training: 60%, validation: 20%, test: 20%

Recap: When Data is Limited

- Try to approximate the validation and test error.
- Two basic approaches

1. Analytic Method
   - Derive mathematic expression to approximate the test error
   - Example: BIC, AIC, VC Dimension
2. Sample Recycling Method
   - Estimate the test error using the same data trained on previous cycle
   - Example: K-fold cross validation, bootstrapping

Recap: Regularization and Capacity Control

- Improve ML model generalization: reducing bias and variance

- Solution: Using complex model and gradually adding more data will reduce both bias and variance

- What if we don’t have more data?
  - Goal: reduce variance (by using a simpler model) while not increasing bias too much (not too simple model)

  - Control this tradeoff by monitoring the regularization parameter

Regularization: Parameter Sharing

- Another way to control model complexity

  - Parameter sharing:
    - Tie together or share different parameters during the training

    - Example:
      1. training neural network models in computer vision applications
      2. Mixtures of Gaussians to jointly estimate cluster covariance

Regularization: Method with Local Support

- Locally weighted method
  - Restrict the training data to predict output on new data
  - “Semi-parametric model”: try to use their locally allocated training data to best fit the model and have a few “metaparameters” which control how to use the allocated data at the test time
  - Example: K-NN Classifier, Locally weighted regression

Regularization: Adding a Penalty Term

- Add a penalty term to the cost function (error, log likelihood)to measure the complexity of the model

$$\text{cost}(\theta) = \text{error}(\text{data}, \theta) + \lambda \text{penalty}(\theta)$$

- Goal: minimize modified, penalty added cost function

- The larger the penalty weight, λ, the simpler our model will be

- How to set the value of λ?
  - Use validation set data to determine the value

Example Penalty: Early Stopping

- Used in training neural networks

- Monitor validation loss and terminate training if the validation loss is not changing for a predefined number