CENG 463
Machine Learning

Lecture 07 - Experiment Design and Performance Measurement

Slides were prepared using the material provided in Stanford's Machine Learning Course by Andrew Ng.

Diagnosis for Learning Algorithms

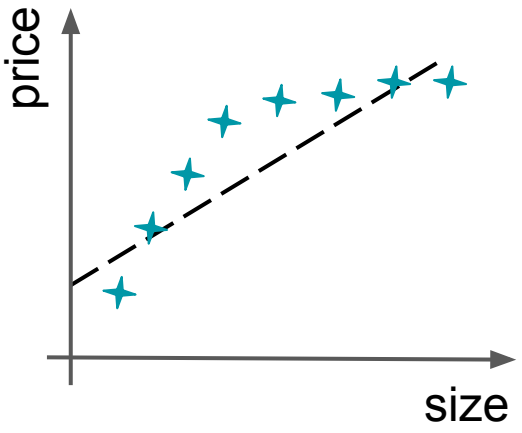
- Suppose you have implemented linear regression to predict housing prices.
- However, when you test your hypothesis on a new set of houses, you find that it makes very large errors in its predictions. What should you try next?
  - Get more training examples?
  - Try smaller sets of features?
  - Try getting additional features?
  - Try adding polynomial features?
- To decide on what to do, one should perform a **diagnostic test.**

Diagnosis for Learning Algorithms

- **Diagnostic test:** A test that you can run to gain insight what is/isn’t working with a learning algorithm, and gain guidance on how to improve its performance.
- Diagnostics can take time to implement, but it is needed to make a healthy decision.

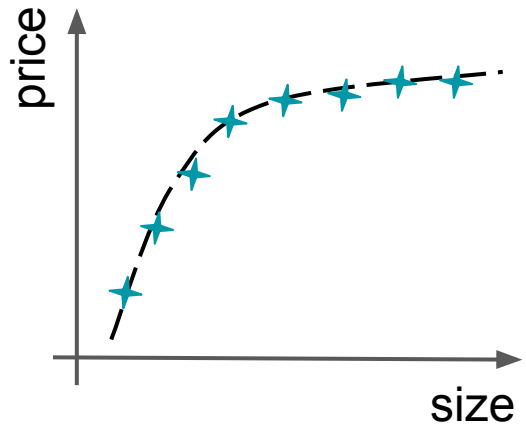
Underfit / Overfit

- Example: Polynomial regression (housing prices)



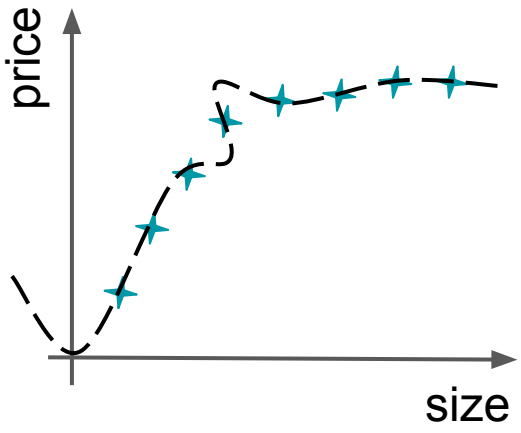
$\theta_0 + \theta_1 x$

underfit (high bias)



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

“just right”



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

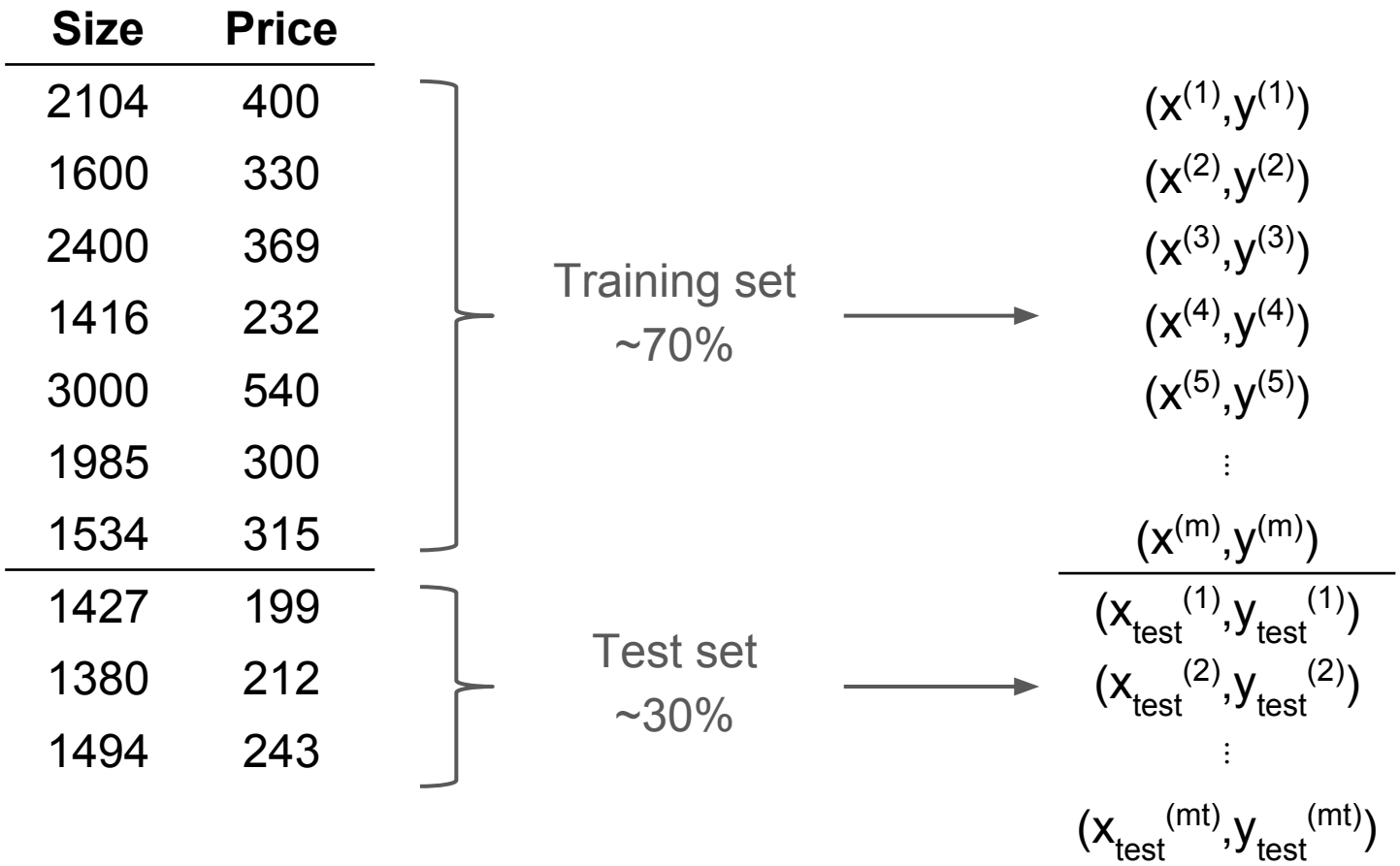
overfit (high variance)

Underfit / Overfit

- **Underfitting:** If we have very few features, the hypothesis fails to model the samples well.
- **Overfitting:** If we have too many features, the hypothesis may fit the training set very well but fail to generalize to new samples (predict prices on new examples).
- When we have many variables (for instance, size, number of bedrooms, age, kitchen size, number of floors, etc.), we can not observe overfitting visually, so we need to develop some other evaluation methodologies.

Evaluating the Hypothesis

- Using the dataset for evaluation of linear regression:



Evaluating the Hypothesis

- Training/testing procedure for linear regression.
  - Learn parameter θ from training set (~70%) by minimizing the training error J(θ).
  - Compute test set error (for ~30% of data):

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left| h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right|^2$$

  - Test set error shows us the **generalization ability** of our hypothesis.

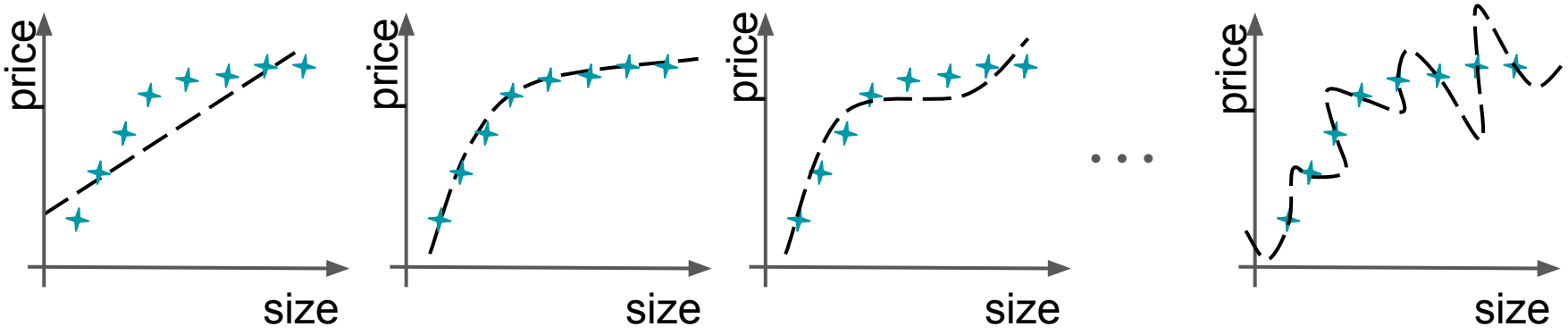
Model Selection

- If we have options for the model to be trained, we also need to decide on its **complexity**. E.g. Polynomial regression:
  1. $h_\theta(x) = \theta_0 + \theta_1 x$
  2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
  3. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

  ⋮

  10. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + ... + \theta_n x^n$
- Here, in addition to the parameter set θ, we need to decide on a complexity parameter: **d = degree of polynomial**. Assume we train each of these models (with a different **d**) with training set and obtain different parameter sets: $\theta^{(1)}$, $\theta^{(2)}$, $\theta^{(3)}$,..., $\theta^{(10)}$

Model Selection



d=1.
Train with training set.
Obtain $\theta^{(1)}=(\theta_0, \theta_1)$

d=2.
Train with training set.
Obtain $\theta^{(2)}=(\theta_0, \theta_1,\theta_2)$

d=3.
Train with training set.
Obtain $\theta^{(3)}=(\theta_0, \theta_1,\theta_2,\theta_3)$

. . .

. . .

d=10.
Train with training set.
Obtain $\theta^{(10)}=(\theta_0, \theta_1,...,\theta_{10})$

Then we can evaluate each of these models using the test set and compute $J_{test}(\theta^{(1)})$, $J_{test}(\theta^{(2)})$,..., $J_{test}(\theta^{(10)})$.

Model Selection

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x^2 + \theta_3 x^3$

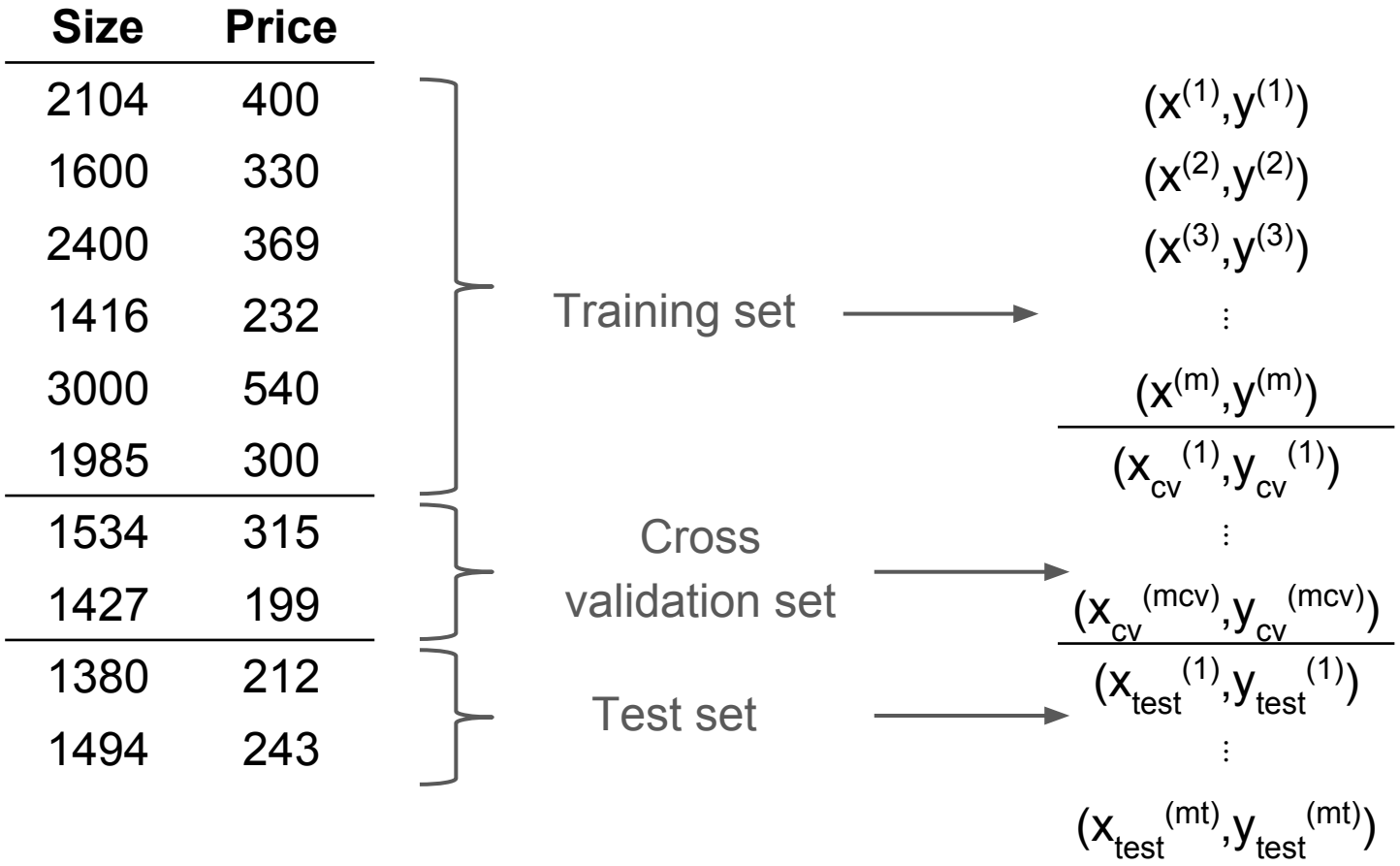
⋮

10. $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x^n$

- Let's say for this example $J_{test}(\theta^{(5)})$ is the lowest error. Then, we choose fifth order polynomial: $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_5 x^5$
- How well does the model generalize? **Not very well !**
- **The problem:** If we select the extra parameter (d) according to its fit to the test set, it will be overly optimistic, because for another test set, the performance will change.

Model Selection with Cross Validation

- To solve this problem, we use another partitioning scheme.



Model Selection with Cross Validation

- Training error:

$$J_{train}(\theta) = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left| h_\theta(x_{train}^{(i)}) - y_{train}^{(i)} \right|^2$$

- Cross validation error:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} \left| h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)} \right|^2$$

- Test error:

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left| h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right|^2$$

Model Selection with Cross Validation

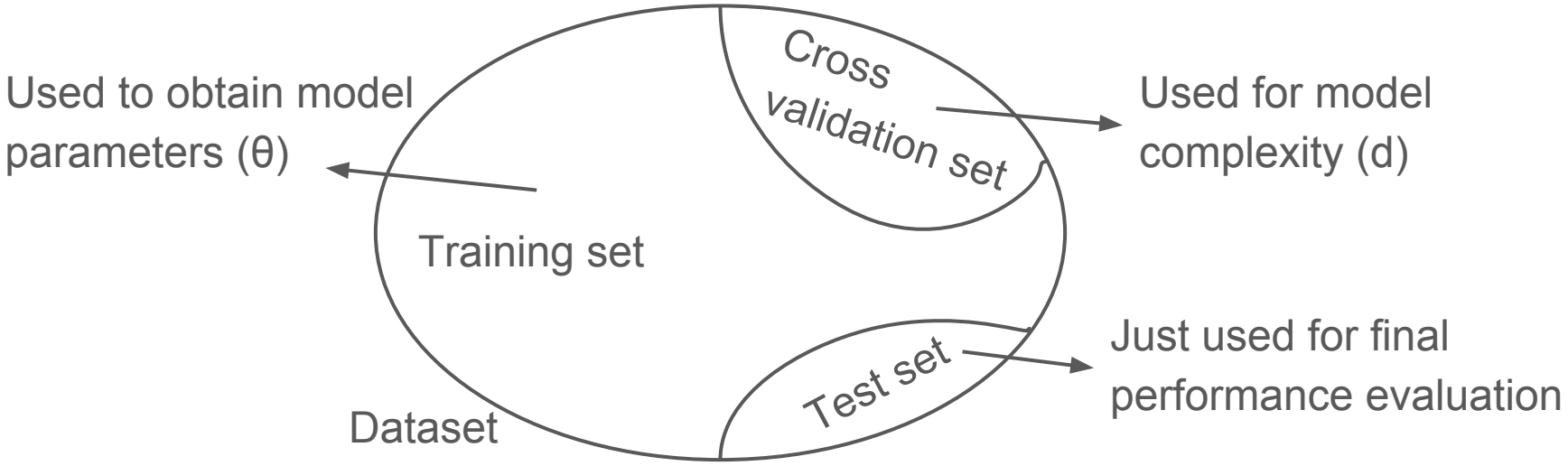
- We again work on the polynomial regression example:
  1. $h_\theta(x) = \theta_0 + \theta_1 x$
  2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
  3. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

  ⋮

  10. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + ... + \theta_n x^n$
- After training models with different **d**, now we compare the cross validation errors: $J^{cv}(\theta^{(1)})$, $J_{cv}(\theta^{(2)})$,..., $J_{cv}(\theta^{(10)})$
- Let's choose the one with minimum error. Let's say $J_{cv}(\theta^{(4)})$ is the lowest (d=4) for this example.

Model Selection with Cross Validation

- Since the selection for **d** is done using the cross validation set ($J_{cv}(\theta^{(i)})$), it is fit to that set.
- Thus, computing $J_{test}(\theta^{(4)})$ gives us a fair performance measure of the model selected. Because, test set is not used for any model parameter (θ) or complexity (d) selection.



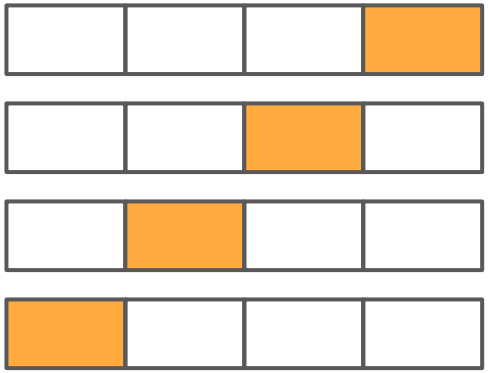
Randomization

- It requires that partitioning of the data should be randomly determined so that the results are independent.
- This is a typical problem in real-world experiments.
  - E.g.1: A part of a text dataset may be collected from a newspaper and another part from a monthly magazine.
  - E.g.2: A part of a street image dataset may have been collected when it is sunny and another part when it is cloudy.
- We have to make sure the variations are randomly distributed in dataset. Training with one part and validating with the other is not healthy.

K-fold Cross Validation

- Sometimes, random sampling of training and test sets is accomplished by repeated use of the same data split differently.
  - The dataset is divided into K equal-sized parts.
  - K-1 parts are used for training.
  - Remaining one part is used as a validation set.

- 4-fold cross validation:



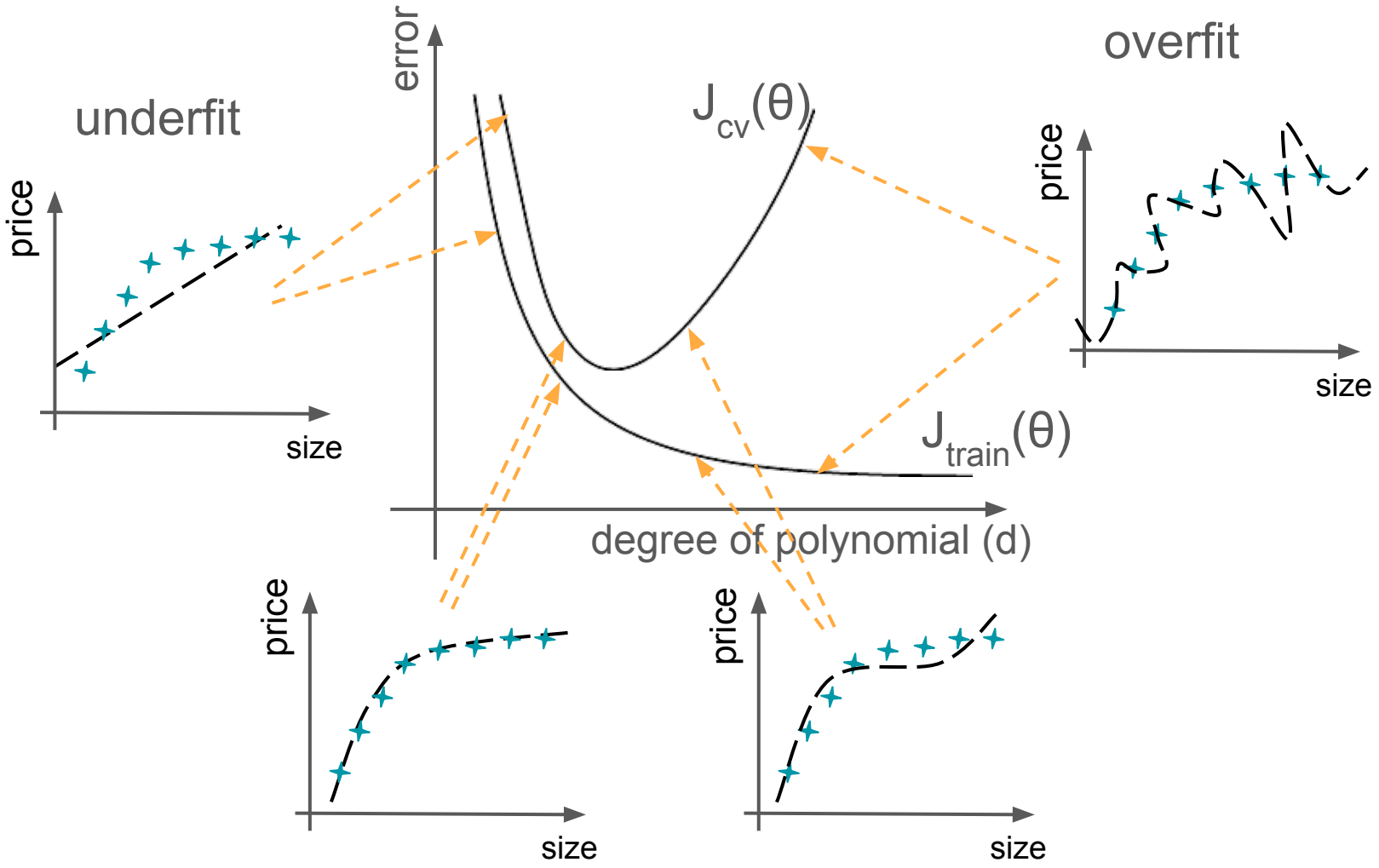
Training with the first three parts and validation with the fourth

Training with another set of three parts and validation with the third

Training with another set of three parts and validation with the second

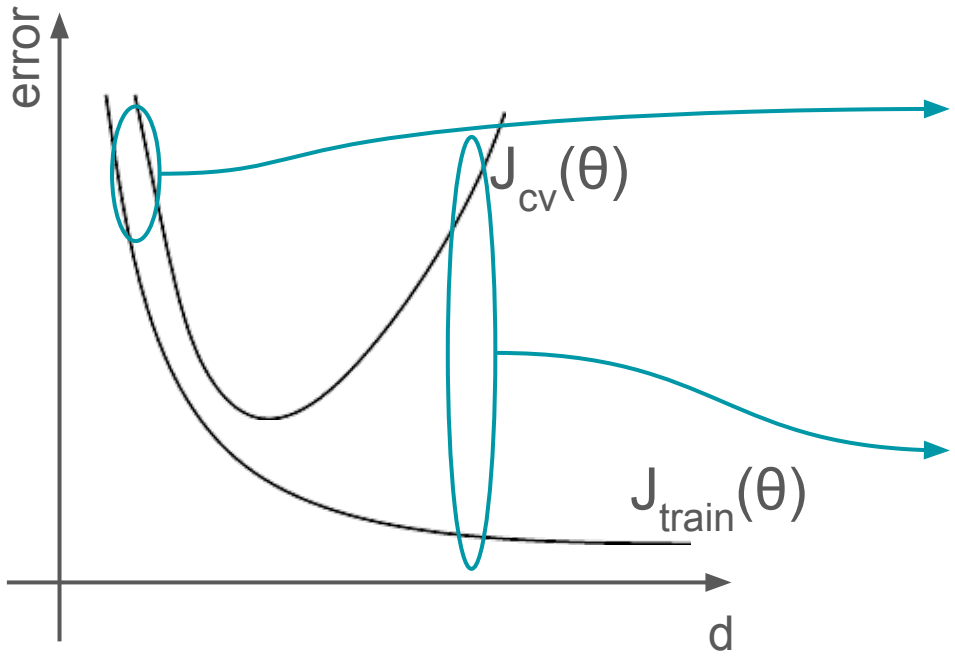
Training with the last three parts and validation with the first

Diagnosing overfit / underfit



Diagnosing overfit / underfit

- Suppose that your learning algorithm is performing less well than you were hoping. ($J_{cv}(\theta)$ or $J_{test}(\theta)$ is high.)
- Is it an underfitting problem or an overfitting problem?



- $J_{train}(\theta)$ is high.
- $J_{cv}(\theta) \approx J_{train}(\theta)$

➡ **Underfitting**

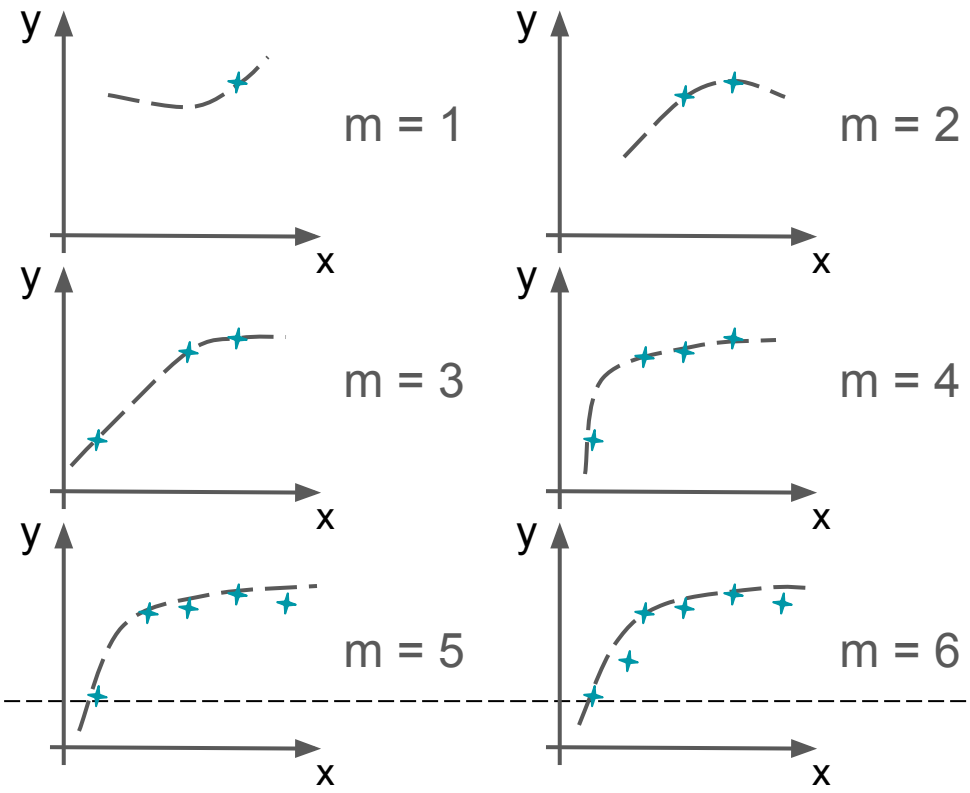
- $J_{train}(\theta)$ is low.
- $J_{cv}(\theta) >> J_{train}(\theta)$

➡ **Overfitting**

Learning Curves

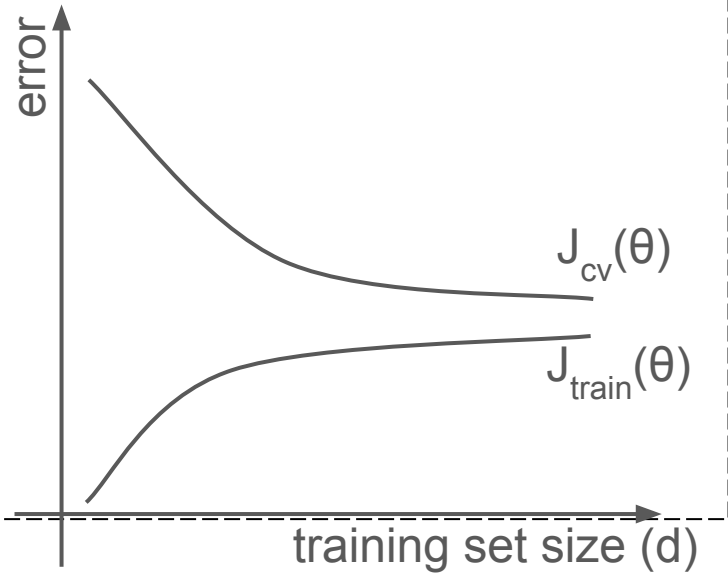
- Learning curves (diagnosis for fixed model complexity)

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$



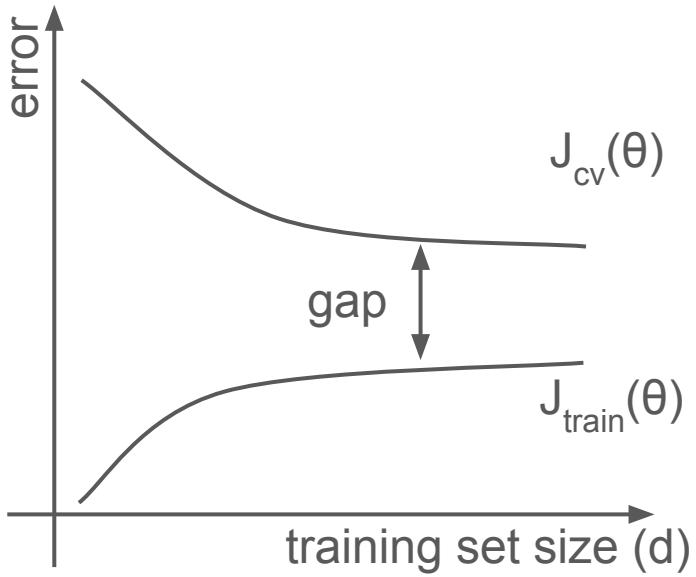
$$J_{train}(\theta) = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left| h_\theta(x_{train}^{(i)}) - y_{train}^{(i)} \right|^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} \left| h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)} \right|^2$$

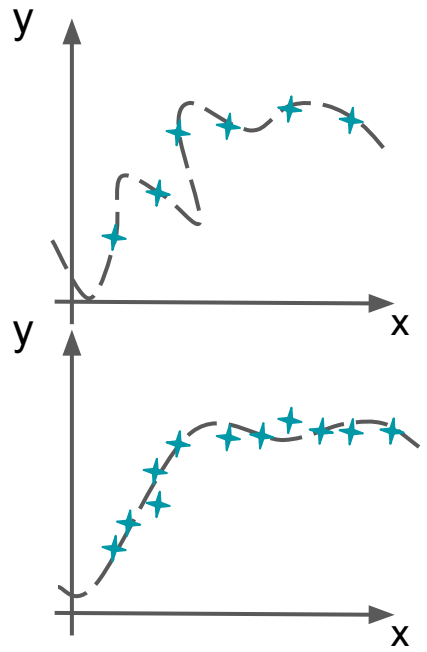


Learning Curves

- If a learning algorithm is suffering from overfitting, getting more training data will help (the gap will decrease).



$$h_\theta(x) = \theta_0 + \theta_1 x + \ldots + \theta_{100} x^{100}$$



Diagnosis

- Suppose you have implemented linear regression to predict housing prices. However, when you test your hypothesis on a new set of houses, you find that it makes very large errors in its predictions.
- When the following actions help?
  - **Get more training examples** fixes **overfitting**.
  - **Try smaller sets of features** fixes **overfitting**.
  - **Try getting additional features** fixes **underfitting**.
  - **Try adding polynomial features** fixes **underfitting**.

Performance Metrics

- Confusion matrix:

	Predicted Class	**Actual Positive**	**Actual Negative**
Actual Class		1	0
Predicted Positive	1	True Positive	False Positive
Predicted Negative	0	False Negative	True Negative

- There are four possible cases:
  - For an actually positive sample,
    - if the prediction is also positive, this is a **true positive**.
    - if the prediction is negative, then it is a **false negative**.
  - For an actually negative sample,
    - if the prediction is also negative, that is a **true negative**.
    - if the prediction is positive, we have a **false positive**.

Problem

- Why do we need different metrics to measure performance?
- E.g. cancer classification:
  - Assume we trained logistic regression model $h_\theta(x)$. (y=1 if cancer, y=0 otherwise)
  - We achieve 1% error on test set (99% correct diagnoses).
  - Now assume that only 0.50% of patients have cancer.

```
function y = predictCancer(x)
    y = 0; %ignore x!
return
```

  - Using the above algorithm, you already have 99.5% accuracy, better than the trained model!
  - This does NOT reveal the real performance.

Solution

- **<u>Precision and Recall:</u>**
  - We want our performance metric to be robust even when y=1 is a rare class.
  - **Precision:** The fraction of patients that actually have cancer among all patients where we predicted y=1.

$$\frac{\#\,\text{True positives}}{\#\,\text{Predicted positives}}$$

  - **Recall:** The fraction of patients that we correctly detect as having cancer among all patients that actually have cancer.

$$\frac{\#\,\text{True positives}}{\#\,\text{Actual positives}}$$

The trade-off

- Precision and recall change as we modify our classifier's threshold, that is at which probability we label a sample as 'positive'.
- Different threshold probability values can be chosen for different tasks or for preferred behavior regarding the same task.
- For instance, straightforward prediction in logistic regression is to set threshold to 0.5:
  - predict 1 if $h_\theta(x) > 0.5$
  - predict 0 if $h_\theta(x) \leq 0.5$

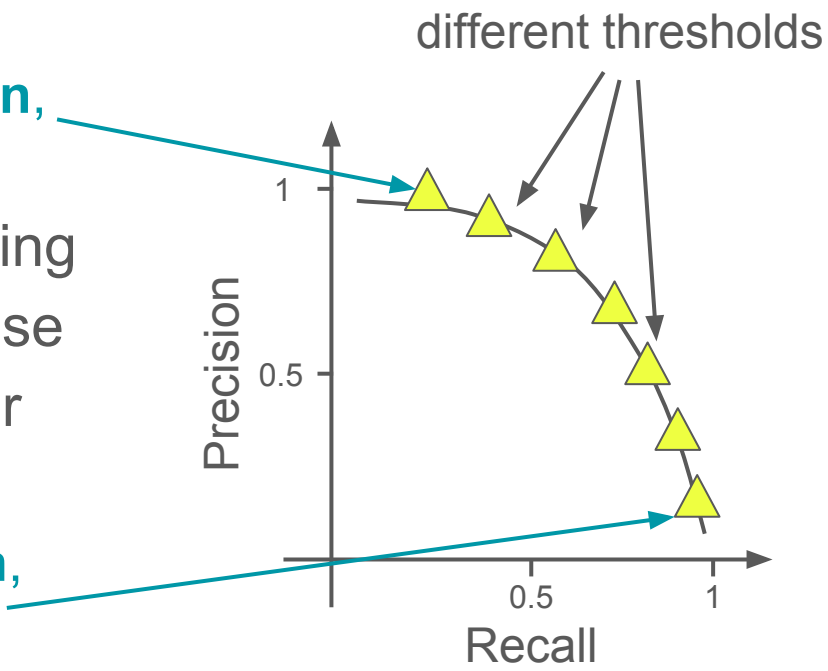
  but this threshold is not mandatory.

The trade-off

- Suppose we want to predict y=1 (cancer) only if we are very confident. So, we choose higher threshold (e.g. 0.9)
  - This results in **higher precision**, **lower recall**.
- Suppose we want to avoid missing some cases of cancer (avoid false negatives). So, we choose lower threshold (e.g. 0.1)
  - This results in **lower precision**, **higher recall**.

$$Precision = \frac{\#\text{ True positives}}{\#\text{ Predicted positives}}$$

$$Recall = \frac{\#\text{ True positives}}{\#\text{ Actual positives}}$$



Comparing Precision/Recall Pairs

- Different thresholds/algorithms produce different results in terms of precision and recall.
- Ideally we want to keep both precision and recall high. How can we compare these precision/recall pairs?
  - A measure that was proposed to compare different precision-recall pairs (different thresholds) is F Score:

$$F_\beta = (1+\beta^2)\frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

  - Most commonly, F1 is used (where β=1), the F score corresponding to the harmonic mean of precision and recall:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Comparing Precision/Recall Pairs

- Results of ‘Average’ and ‘F1 score’ for three algorithms:

	Precision (P)	**Recall (R)**	**Average**	**F_1 score**
Algorithm 1	0.50	0.40	0.45	0.444
Algorithm 2	0.70	0.10	0.40	0.175
Algorithm 3	0.02	1.00	0.51	0.039

- Since we want to keep both precision and recall high, Algorithm 1 in this example is better.
- Unlike ‘Average’, F1 score gives Algorithm 1 the highest score.

Summary

- We have learned about:
  - Underfit / Overfit
  - Using the Dataset for Evaluation
  - Model Complexity and Model Selection
  - Randomization
  - Diagnosing Underfit / Overfit
  - Learning Curves
  - Precision, Recall and the Trade-off in Between
  - F_1 Score