# CS 60050
# Machine Learning

Evaluation and Error analysis

Validation and Regularization

Some slides taken from course materials of Andrew Ng

# How to evaluate a model?

- Regression
  - Some measure of differences between actual values and values predicted by model
- Classification
  - Whether predicted classes match the actual classes

# Evaluation metrics for classification

- Let y = actual class, h = predicted class for an example

- Accuracy: Out of all examples, for what fraction is h = y?

- But accuracy is often not sufficient to indicate performance in practice

# Skewed classes

- Often the class of interest is a rare class (y=1)
  - Spam emails / social network accounts
  - Cancerous cells
  - Fraud credit card transactions

# Skewed classes

- Often the class of interest is a rare class (y=1)
  - Spam emails / social network accounts
  - Cancerous cells
  - Fraud credit card transactions
- Precision: Out of all examples for which model predicted h=1, for what fraction is y=1?
- Recall: Of all examples for which y=1, for what fraction did model correctly predict h=1?

# Precision vs. Recall: tradeoff

- Predict y=1 if h > some threshold
- Predict y=1 only if highly confident: high precision, lower recall
- Avoid missing too many cases with y=1: high recall, lower precision

- F-score: harmonic mean of Precision and Recall

$$2\frac{PR}{P+R}$$

# Confusion Matrix

| | | Predicted Label | |
|---|---|---|---|
| | | h = 1 | h = -1 |
| True Label | $y = 1$ | True positive | False negative |
| | $y = -1$ | False positive | True negative |

Precision: (True positive) / (True positive + False positive)

Recall: (True positive) / (True positive + False negative)

# Another format of confusion matrix

| | | y = +1 | y = −1 |
|---|---|---|---|
| h | +1 | no error | *false accept* |
| | −1 | *false reject* | no error |

- Two types of errors:
  - False positive/accept: hypothesis +1, true label -1
  - False negative/reject: hypothesis -1, true label +1

# Two types of errors

- How do we penalize the two types of errors?
- Which is more important – higher Precision or higher Recall?
- Depends on the specific application

# Example: Fingerprint verification

- Input fingerprint, classify as known identity or intruder
- Application 1: Supermarket verifies customers for giving a discount
- Application 2: For entering into RAW, GoI

# Example: Fingerprint verification

- Input fingerprint, classify as known identity or intruder
- Application 1: Supermarket verifies customers for giving a discount
- Application 2: For entering into RAW, GoI

| | | y | |
|---|---|---|---|
| | | $+1$ | $-1$ |
| $h$ | $+1$ | 0 | 1 |
| | $-1$ | 10 | 0 |

# Example: Fingerprint verification

- Input fingerprint, classify as known identity or intruder
- Application 1: Supermarket verifies customers for giving a discount
- Application 2: For entering into RAW, GoI

| | | y | |
|---|---|---|---|
| | | $+1$ | $-1$ |
| $h$ | $+1$ | 0 | 1 |
| | $-1$ | 10 | 0 |

| | | y | |
|---|---|---|---|
| | | $+1$ | $-1$ |
| $h$ | $+1$ | 0 | 1000 |
| | $-1$ | 1 | 0 |

# On what data to measure precision, recall, error rate, ..?

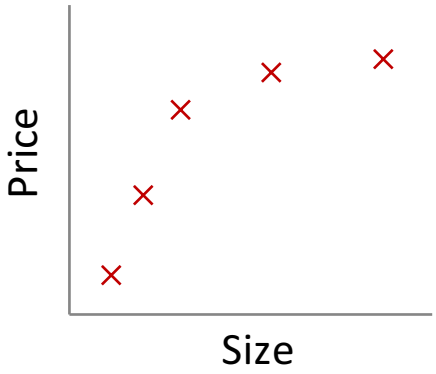
- Option 1: training set
- Option 2: some other set of examples that was unknown at the time of training (test set)

- Motivation for ML: learn a model that performs well (generalizes well) to unknown examples
- Option 2 gives better guarantees for generalization of a learnt model

# Error Analysis

# Bias and Variance

# Example: Linear regression (housing prices)




$\theta_0 + \theta_1 x$

Fitting a linear function

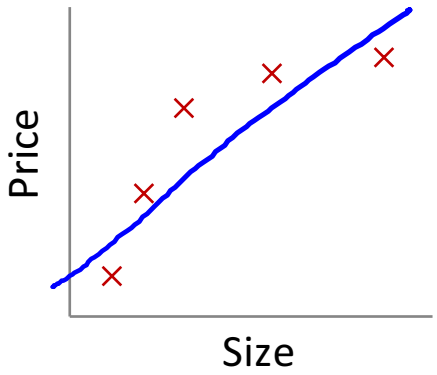
$\theta_0 + \theta_1 x + \theta_2 x^2$

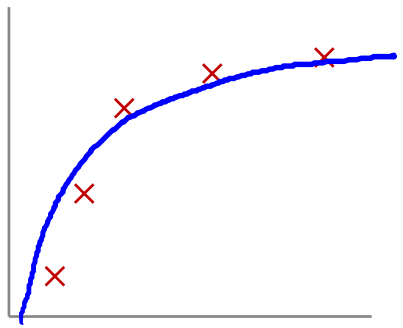
Fitting a quadratic function

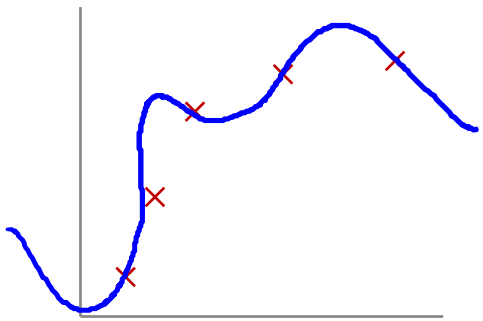
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Fitting a higher order function

# Bias vs. variance in linear regression





$d=1$

$\theta_0 + \theta_1 x$

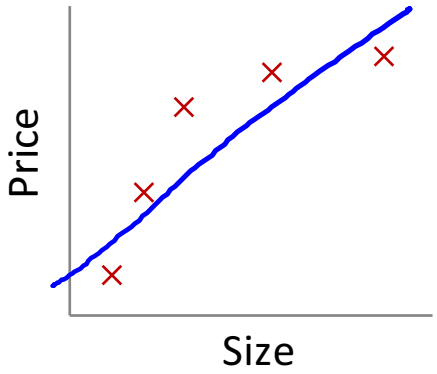
$d=2$

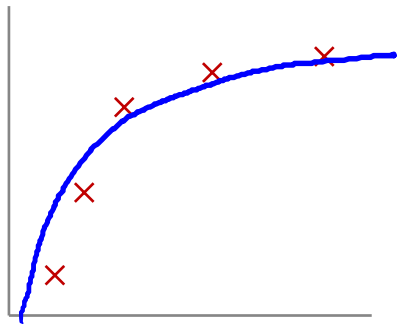
$\theta_0 + \theta_1 x + \theta_2 x^2$

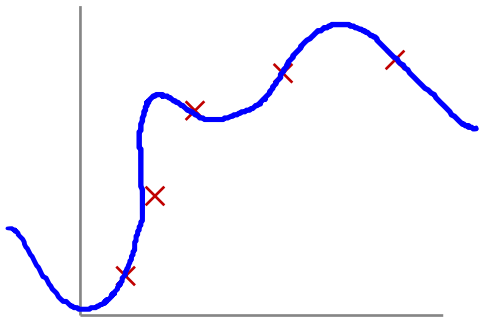
$d=4$

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

# Bias vs. variance in linear regression





High bias (underfitting)

$d = 1$

$\theta_0 + \theta_1 x$

"Just right"

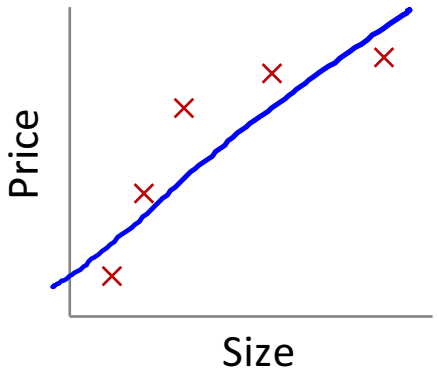
$d = 2$

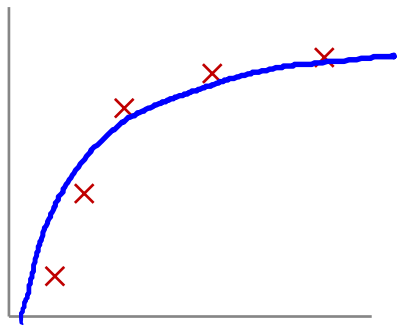
$\theta_0 + \theta_1 x + \theta_2 x^2$

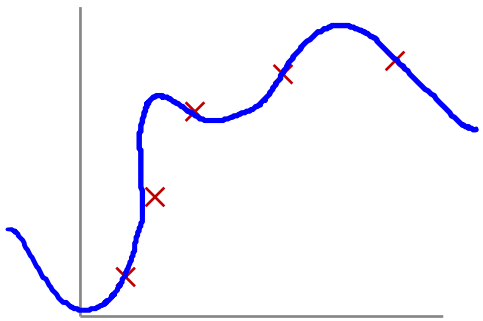
$d = 4$

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

# Bias vs. variance in linear regression





High bias (underfitting)

$d = 1$

$\theta_0 + \theta_1 x$

"Just right"

$d = 2$

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

High variance (overfitting)

$d = 4$

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

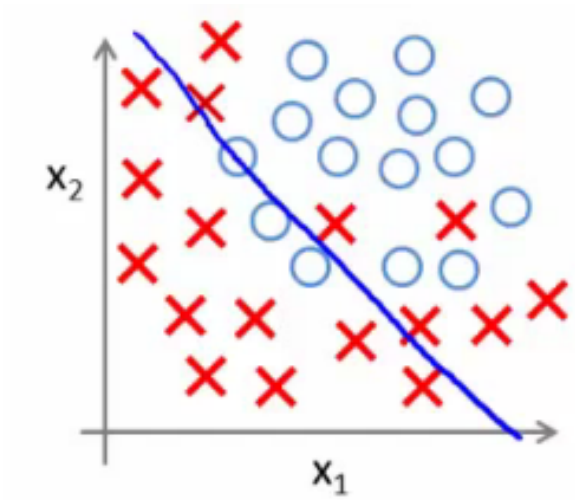
# Overfitting

If we have too many features, the learned hypothesis may fit the training set very well

but fail to generalize to new examples.

# Bias vs. variance in logistic regression

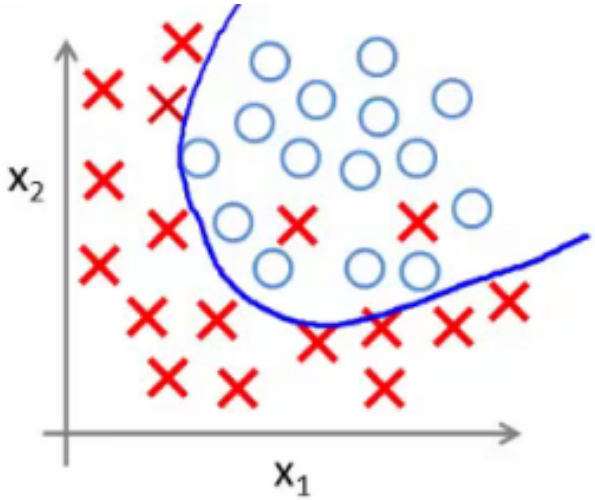
Example: Logistic regression



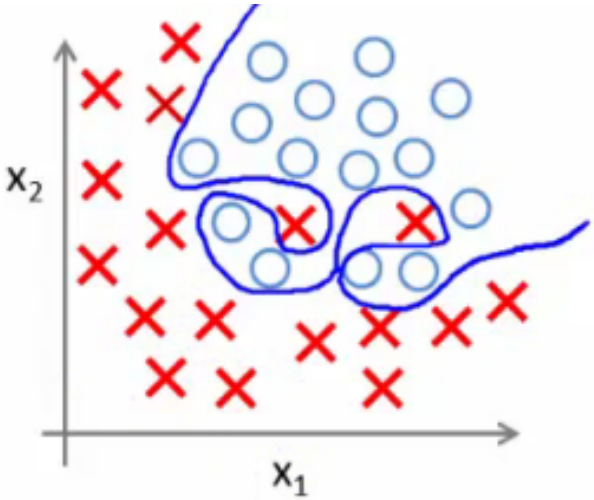

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

( $g$ = sigmoid function)

**UNDERFITTING**
**(high bias)**




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




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

**OVERFITTING**
**(high variance)**

# Sources of noise and error

- While learning a target function using a training set
- Two sources of noise
  - Some training points may not come exactly from the target function: stochastic noise
  - The target function may be too complex to capture using the chosen hypothesis set: deterministic noise

- Generalization error: Model tries to fit the noise in the training data, which gets extrapolated to the test set

# Ways to handle noise

- Validation
  - Check performance on data other than training data, and tune model accordingly
- Regularization
  - Constraint the model so that the noise cannot be learnt too well

# Validation

# Validation

- Divide given data into train set and test set
  - E.g., 80% train and 20% test
  - Better to select randomly
- Learn parameters using training set
- Check performance (validate the model) on test set, using measures such as accuracy, misclassification rate, etc.
- Trade-off: more data for training vs. validation

# An example: model selection

- Which order polynomial will best fit a given data? Polynomials available: $h_1$, $h_2$, …, $h_{10}$
- As if an extra parameter - degree of the polynomial - is to be learned
- Approach
  - Divide into train and test set
  - Train each hypothesis on train set, measure error on test set
  - Select the hypothesis with minimum test set error

# An example: model selection

- Problem with the previous approach
  - The test set error we computed is not a true estimate of generalization error
  - Since our extra parameter (order of polynomial) is fit to the test set

# An example: model selection

- Approach 2
  - Divide data into train set (60%), validation set (20%) and test set (20%)
  - Select that hypothesis which gives lowest error on validation set
  - Use test set to estimate generalization error

- Note: Test set not at all seen during training

# Popular methods of evaluating a classifier

- Holdout method
  - Split data into train and test set (usually 2/3 for train and 1/3 for test). Learn model using train set and measure performance over test set
  - Usually used when there is sufficiently large data, since both train and test data will be a part

# Popular methods of evaluating a classifier

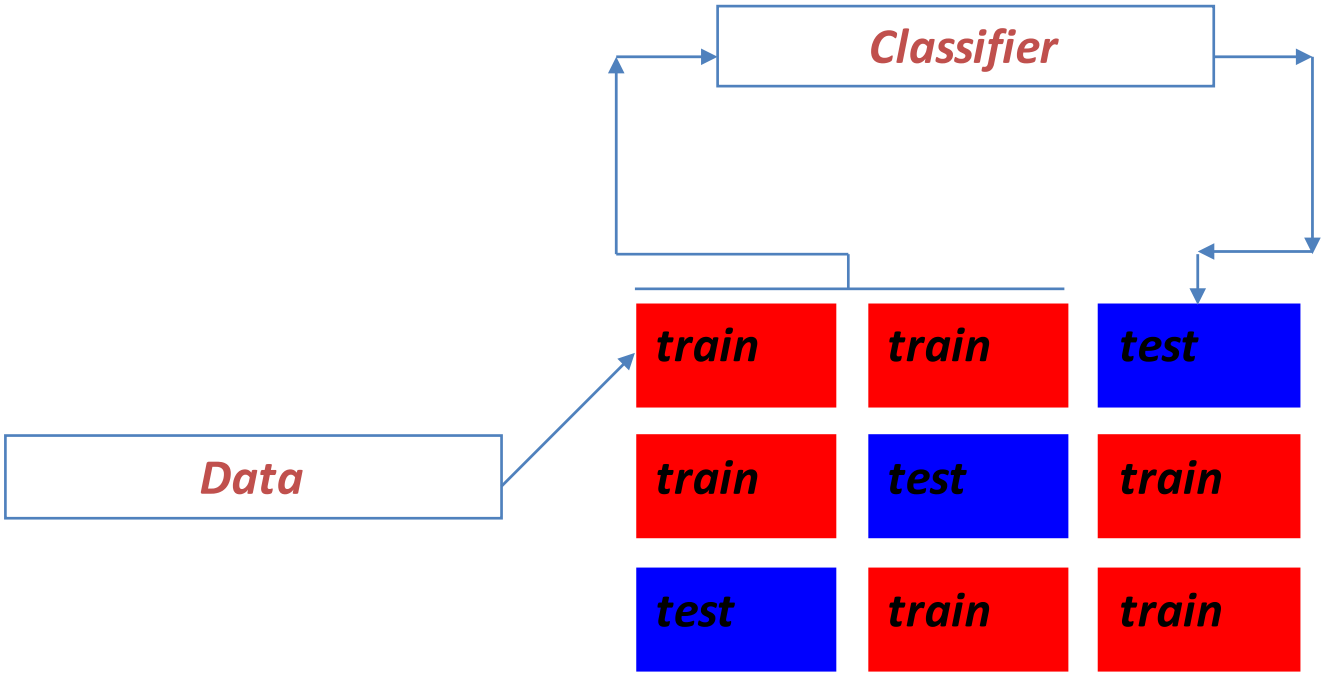
- Repeated Holdout method
  - Repeat the Holdout method multiple times with different subsets used for train/test
  - In each iteration, a certain portion of data is randomly selected for training, rest for testing
  - The error rates on the different iterations are averaged to yield an overall error rate
  - More reliable than simple Holdout

# Popular methods of evaluating a classifier

- k-fold cross-validation
  - *First step*: data is split into *k* subsets of equal size;
  - *Second step*: each subset in turn is used for testing and the remainder for training
  - Performance measures averaged over all folds

- Popular choice for k: 10 or 5
- Advantage: all available data points being used to train as well test model

# k-fold cross validation (shown for k=3)



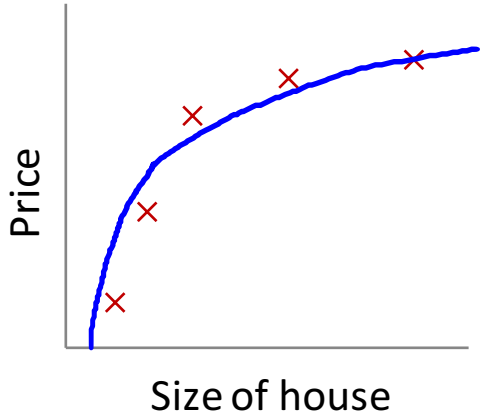
# Regularization

# Addressing overfitting: Two ways

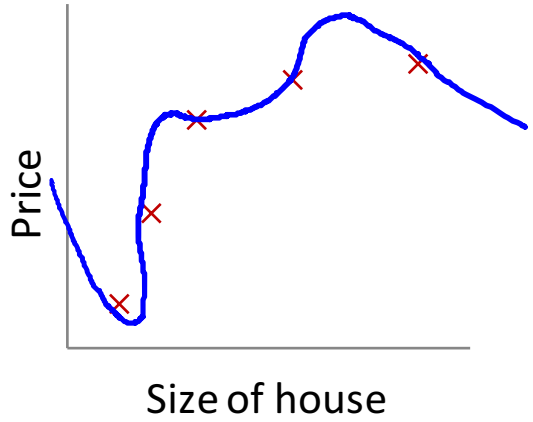
1. Reduce number of features
   - Manually select which features to keep
   - Problem: loss of some information (discarded features)

2. Regularization
   - Keep all the features, but reduce magnitude/values of parameters
   - Works well when we have a lot of features, each of which contributes a bit to predicting

# Intuition of regularization




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




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

Suppose we penalize and make $\theta_3, \theta_4$ really small.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \quad + K\,\Theta_3^2 + K\,\Theta_4^2$$

# Regularization for linear regression

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

$$\min_\theta J(\theta)$$

By convention, regularization is not applied on $\theta_0$ (makes little difference to the solution)

λ: Regularization parameter

Smaller values of parameters lead to more generalizable models, less overfitting

# Regularization for linear regression

In regularized linear regression, we choose $\theta$ to minimize

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

Regularization parameter $\lambda$

- Controls trade-off between our two goals
- (1) fitting the training data well
- (2) keeping values of parameters small

- What if λ is too large? Underfitting

# L1 and L2 regularization

- What we are discussing is called L2 regularization or “ridge” regularization
  - adds *squared magnitude* of parameters as penalty term
- Look up L1 or “Lasso” regularization
  - adds *absolute value of magnitude* of parameters as penalty term

# Regularized linear regression

## Gradient Descent for ordinary linear regression

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(j = 0, 1, 2, 3, \ldots, n)$$

$\}$

## Regularized linear regression

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

$$\min_\theta J(\theta)$$

## Gradient Descent for Regularized Linear Regression

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

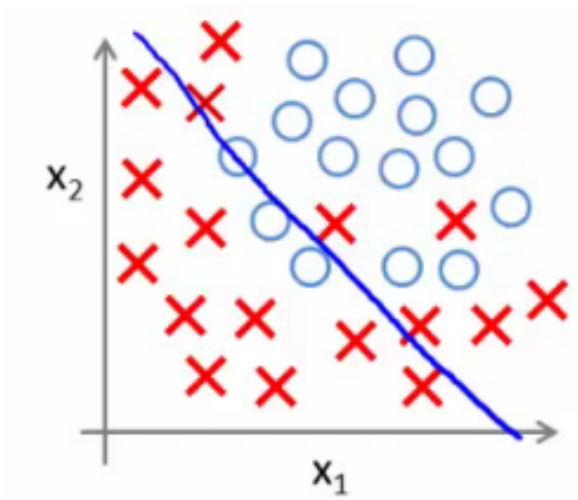
$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

$\}$

# Regularized logistic regression

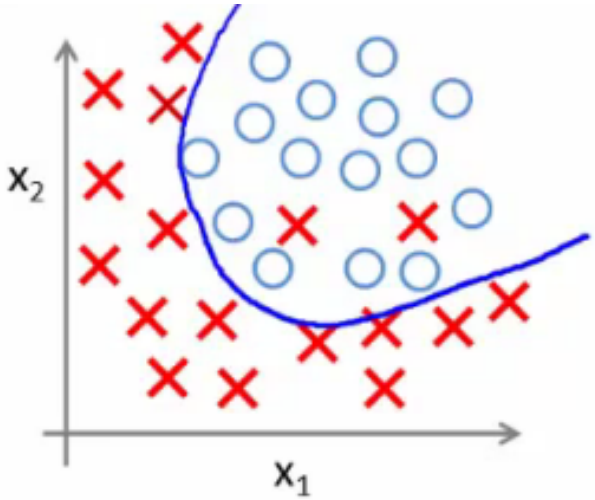
Example: Logistic regression



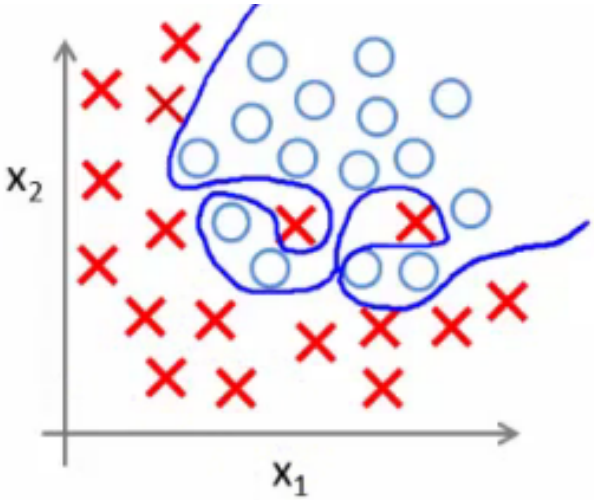

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

( $g$ = sigmoid function)

**UNDERFITTING**
**(high bias)**




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




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

**OVERFITTING**
**(high variance)**

## Gradient descent for ordinary logistic regression

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(j = 0, 1, 2, 3, \ldots, n)$$

$\}$

## Gradient Descent for Regularized Logistic Regression

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

$$+ \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

## Gradient Descent for Regularized Logistic Regression

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

$$+ \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

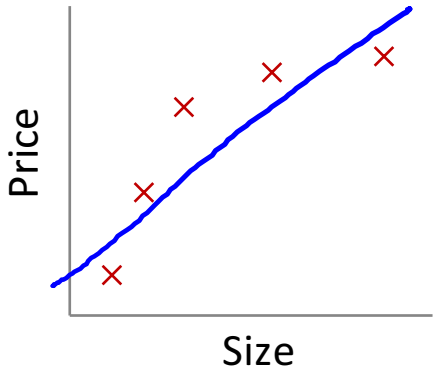
$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

$\}$

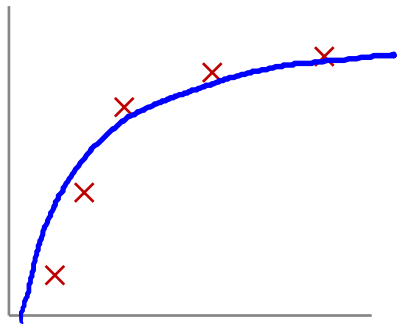
# Bias vs. Variance
# A closer look

Example: Linear regression



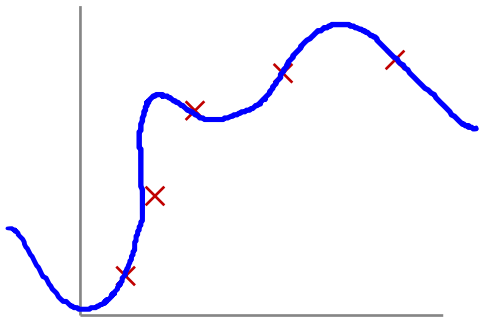

High bias
(underfit)

$d=1$



"Just right"

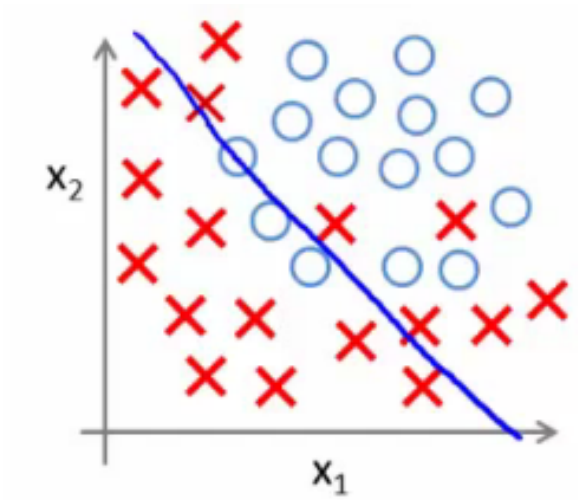
$d=2$



High variance
(overfit)

$d=4$

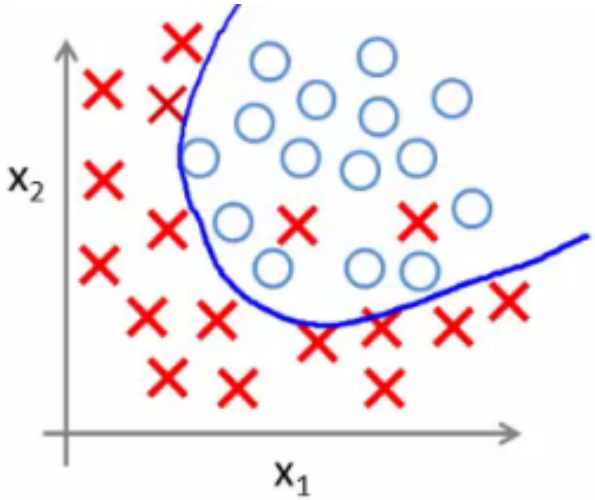
Example: Logistic regression



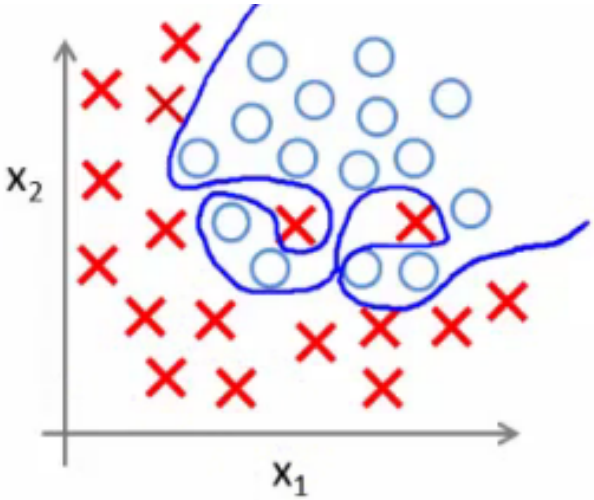

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

( $g$ = sigmoid function)

**UNDERFITTING**
**(high bias)**




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

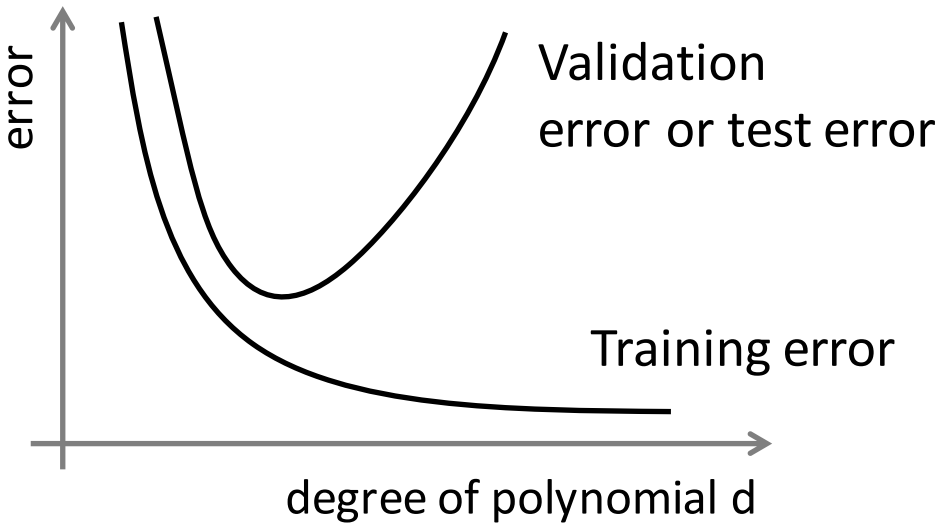



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots)$

**OVERFITTING**
**(high variance)**

# Analysing bias vs. variance

- Suppose your model is not performing as well as expected. Is it a bias problem or a variance problem?




Bias (underfit):
Both training error and validation / test error are high

Variance (overfit):
Low training error
High validation / test error

# Bias vs. Variance

- Bias and variance both contribute to the error of classifier
- Variance is error due to randomness in how the training data was selected (variance of an estimate refers to how much the estimate will vary from sample to sample)
- Bias is error due to something systematic, not random

# Will more training data help?

- A learnt model is not performing as well as expected. Will having more training data help?

- Note that there can be substantial cost for getting more training data.

# Will more training data help?

- A learnt model is not performing as well as expected. Will having more training data help?

- Note that there can be substantial cost for getting more training data.

- If model is suffering from high bias, getting more training data will not (by itself) help much.
- If model is suffering from high variance, getting more training data is likely to help

# Practical approach

- Divide data into training set and validation set
- Start with simple algorithm, train on different amounts of training data, test performance on validation set
- Plot learning curves to decide if more training data, more features likely to help
- Error analysis: Manually examine the examples (in validation set) where algorithm made errors. Any systematic trend in what type of examples it is making errors on?

# Learning curves

- How do training error (in-sample error) and test or validation error (out-of-sample error) generally vary with number of training points?