6.0002 Exam Review

*Not comprehensive. All material covered in the course is fair game.

Machine Learning

Machine learning

Field of study that gives computers the ability to learn without being explicitly programmed

- General paradigm for data modeling
    - when designing a model, rather than choosing parameters we let the data show us the best parameters
    - Makes data analysis much more scalable
    - Ex: Predicting whether a tumor is benign or malignant

Choosing Features

- Want to choose features that help describe what’s important in the data without overfitting (i.e. want **high Signal-to-Noise Ratio**).
- Need to decide how to measure distance based on features
  - Sometimes we use **Minkowski Metric, Earth Mover’s Distance, or Dynamic Time Warping** as a measure of distance
- Need to decide how to weight different features

Feature scaling

- A way to standardize the range of features of data
- Allows us to have each number contribute equally to objective function
- Z-scaling
  - each feature has an average of 0 and a standard deviation of 1
- Interpolation
  - map min to 0, max to 1, and linearly interpolate

Confusion matrix (because accuracy is not enough)

	Predicted +	Predicted -
Actual +	True Positive (tp)	False Negative (fn)
Actual -	False Positive (fp)	True Negative (tn)

- accuracy = (tp + tn) / (tp + fp + tn + fn)
- sensitivity (recall) = tp / (tp + fn)
- Specificity (true negative rate or precision) = tn / (tn + fp)
- positive predictive value= tp / (tp + fp)
- Which metric we value more depends on situation
    - class imbalance: consider a disease that occurs in 0.1% of population

Datasets and Validation

- Split into **training**, **validation** (to help choose parameter values), and **testing** datasets.
- When we don't have a huge amount of data we use **cross-validation**
  - Repeated Random Sampling - Randomly split data into testing and training datasets k times. For each train its own model, and take the average of their error rates.
  - K-fold Cross-Validation - Divide data into N/k testing datasets. Train k models that each use one of the N/k testing datasets and all the remaining data for training. Compute the average error rate.
  - Leave-one-out - Same as above, but k = N

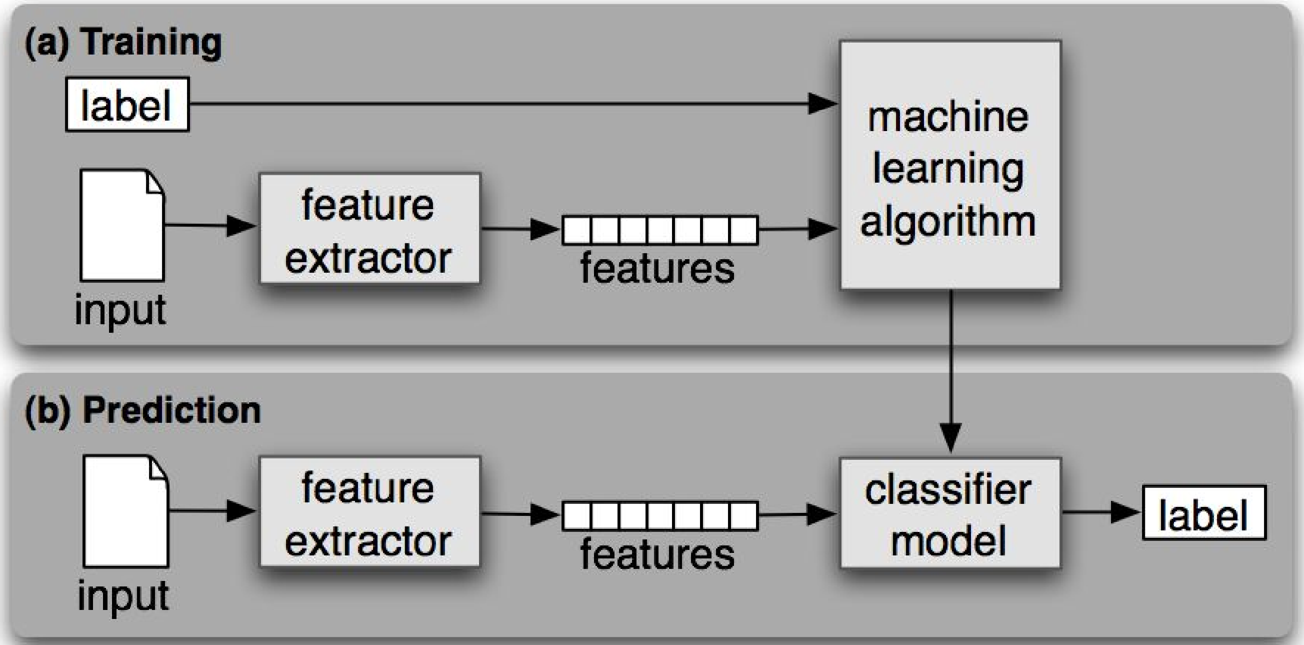
Supervised learning

- Each data point = (feature vector, annotation)
- Given: points + discrete labels (*Classification*)
  - Goal: predict labels for new points

or

- Given: points + real-valued scores (*Regression*)
  - Goal: predict scores for new points

Supervised Learning

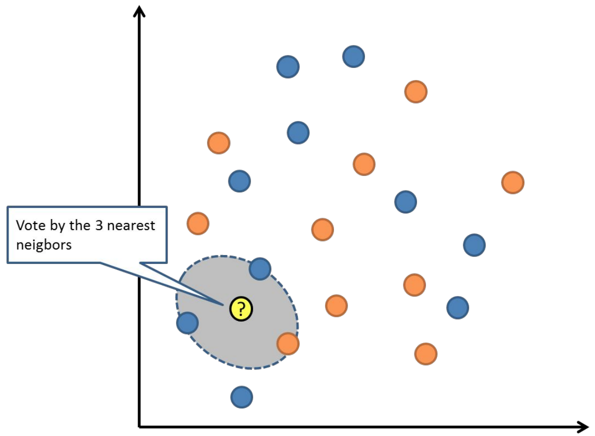


http://www.nltk.org/book/ch06.html

Supervised classification

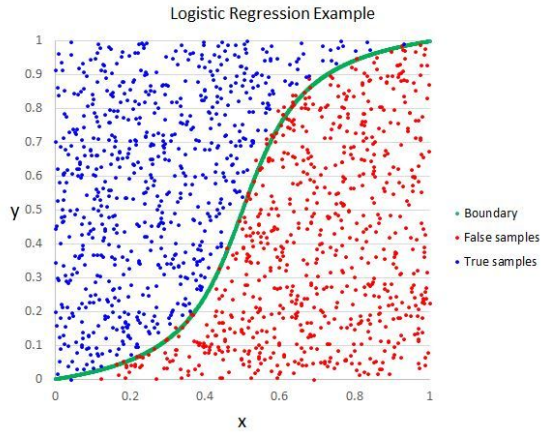
K-Nearest Neighbors

- For new dataset look at k nearest neighbors and assign it the most common label you see



Logistic Regression

- Designed to predict the probability of an event (finite number of labels, usually just 0 and 1) by finding weights for each feature



Test Yourself on Supervised Learning!

- True or False: Logistic Regression is useful when we want to predict continuous values.

- True or False: KNN is a deterministic algorithm

Test Yourself on Supervised Learning!

- Logistic Regression is useful when we want to predict continuous values. **FALSE**

- True or False: KNN is a deterministic algorithm. **TRUE**

Unsupervised learning

- Given: points (no labels/scores)
  - Goal: infer underlying structure of data
- A common unsupervised learning problem is *clustering*
  - Cluster data points
  - E.g., type 1 tumor, type 2 tumor, ... instead of malignant and benign
  - Algorithm: agglomerative Hierarchical clustering, k-means
- Choose k
  - Domain knowledge
  - Plot your clustering results, understand it, trial and error

Hierarchical Clustering

Greedy, deterministic, but *very* slow

Agglomerative Hierarchical clustering

1. Start by assigning each item to its own cluster

2. Find the closest (by linkage) pair of clusters and merge them into a single cluster

3. Continue the process until all items are clustered into a single cluster of size N.

Linkages:

- ***Average linkage:*** average distance between two data points, one from each clusters
- ***Complete linkage:*** max dist between two data points, one from each cluster
- ***Single-linkage:*** min dist between two data points, one from each cluster

Test Yourself on Hierarchical Clustering!

- True or False: The results of hierarchical clustering depend on the linkage criteria used.

- True or False: In hierarchical clustering there are O(n) iterations (combinations of clusters) before the algorithm terminates.

Test Yourself on Hierarchical Clustering!

- The results of hierarchical clustering depend on the linkage criteria used. **TRUE**

- In hierarchical clustering there are O(n) iterations (combinations of clusters) before the algorithm terminates. **TRUE**

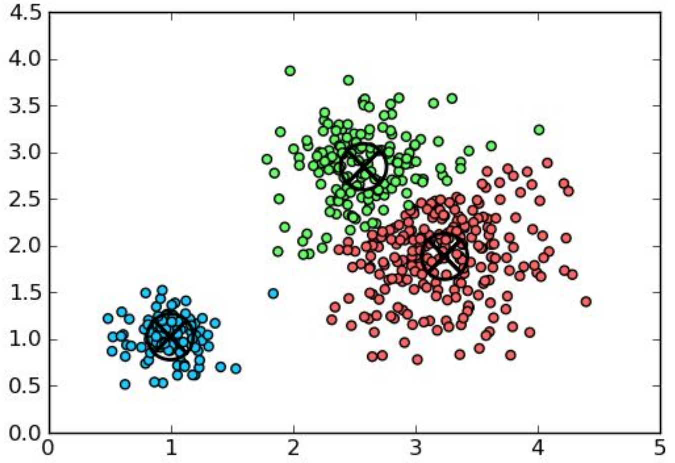
K-Means

Approximate solution

Fast (~linear in number of data point)

Select K by:

- Trial-and-error with cross validation
- a priori knowledge
- hierarchical clustering

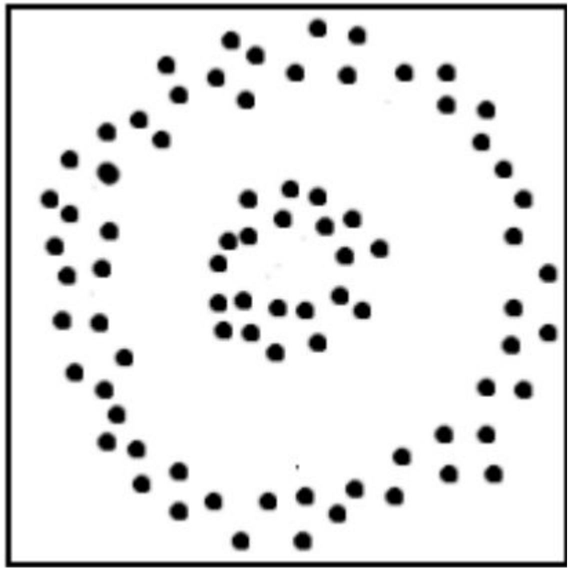


Test Yourself on K-means!

Given the 2-D feature vector representation shown on the right and running a k-means clustering algorithm (with k = 2), which distance metric separates the clusters such that one cluster contains all the points in the outer ring and the other contains all the points in the inner ring?

a) Euclidean distance b) Manhattan distance

c) Both of the above d) None of the above



Comparing ML Algorithms

	Supervised or Unsupervised	Deterministic or Nondeterministic	Complexity
Hierarchical Clustering	Unsupervised	Deterministic	~ O(n^3)
K-Means	Unsupervised	Nondeterministic	~ O(n) per iteration
K Nearest Neighbors	Supervised	Deterministic	~ O(n)
Logistic Regression	Supervised	Deterministic	Reasonably fast, we haven't given you enough details to know exactly

Test Yourself!

- The final results of k-means depends on the initial centroids

- Training an algorithm on data set A and testing it on the same data set A is an example of supervised learning.

- Hierarchical and k-means clustering are both greedy algorithms.

Test Yourself!

- The final results of k-means depends on the initial centroids **TRUE**

- Training an algorithm on data set A and testing it on the same data set A is an example of supervised learning. **FALSE**

- Hierarchical and k-means clustering are both greedy algorithms. **TRUE**

Optimization

Structure of optimization problems

- **Objective function**
  - What we want to optimize
  - Evaluate the goodness of each possible solution
- Set of **constraints**
  - Conditions that <u>must</u> be satisfied by solution

The ‘knapsack problem’

- Given:
  - Budget (e.g., how much weight you can carry / how much space your bag has)
  - collection of items, each with a value and cost (e.g., weight / space)
- Goal: find set of items yielding highest value <u>within budget</u>

General and common optimization problem: most optimization problems seen in class are instances of knapsack

- Objective function:
- Constraint:

The ‘knapsack problem’

- Given:
  - Budget (e.g., how much weight you can carry / how much space your bag has)
  - collection of items, each with a value and cost (e.g., weight / space)
- Goal: find set of items yielding highest value <u>within budget</u>

General and common optimization problem: most optimization problems seen in class are instances of knapsack

- Objective function: sum of values of the items you choose
- Constraint: items you choose can’t exceed the budget

The change making problem

How can a given amount of money be made with the least number of coins of given denominations? Assume infinite supply of each type of coins and the order of coins doesn't matter.

- Objective function:
- Constraints:
- Goal:

The change making problem

How can a given amount of money be made with the least number of coins of given denominations? Assume infinite supply of each type of coins and the order of coins doesn't matter.

- Objective function:
  - number of coins needed to make the given amount of money
- Constraints:
  - Sum of coins == required change, use coins with valid denominations, #coins of each denomination >= 0
- Goal:
  - Minimize the objective function

Basic Approaches

- Brute Force
- Greedy Algorithm
- Dynamic Programming

Brute-force algorithms

- enumerate every possible solution
- filter those that violate constraints
- pick one with best objective value

Brute-force algorithms

- enumerate every possible solution
- filter those that violate constraints
- pick one with best objective value

waaaay too slow

Test Yourself on Brute Force Algorithms!

- In a knapsack problem with 100 items, how many potential solutions must be considered by a brute force algorithm?

- True or False: All brute force algorithms are exponential.

Test Yourself on Brute Force Algorithms!

- In a knapsack problem with 100 items, how many potential solutions must be considered by a brute force algorithm? **2^100**

- True or False: All brute force algorithms are exponential. **FALSE**

Greedy algorithms

- repeatedly make ‘locally best’ choice
    - what is ‘locally best’ ?
- Fast! Easy to implement.
- <u>no guarantee</u> of *globally* optimal solution

Test Yourself on Greedy Algorithms!

- True or False: Greedy algorithms never achieve globally optimal results

- True or False: Running a greedy algorithm and then sorting the dataset and running the algorithm again could produce a different result

Test Yourself on Greedy Algorithms!

- Greedy algorithms never achieve globally optimal results **FALSE**

- Running a greedy algorithm and then sorting the dataset and running the algorithm again could produce a different result **TRUE**

Dynamic programming

When can you use it? **overlapping subproblems** and **optimal substructure**

- **Optimal substructure**: a globally optimal solution can be found by combining optimal solutions to local subproblems
- **Overlapping subproblems**: an optimal solution involves solving the same problem multiple times

- Implement solutions using **memoization**, either top-down or bottom-up
- Guarantees a *globally* optimal solution with polynomial complexity (fast!)
- Trade time for space
- Calculate results for each subproblem only once and store them for later.

Top-Down

```
def fib(n, lookup):

    # Base case
    if n == 0 or n == 1 :
        lookup[n] = n

    # If the value is not calculated previously then
    # calculate it
    if lookup[n] is None:
        lookup[n] = fib(n-1 , lookup)  + fib(n-2 , lookup)

    # return the value corresponding to that value of n
    return lookup[n]
```

Bottom-Up

```
def fib(n):

    # array declaration
    f = [0]*(n+1)

    # base case assignment
    f[1] = 1

    # calculating the fibonacci and
storing the values
    for i in xrange(2 , n+1):
        f[i] = f[i-1] + f[i-2]
    return f[n]
```

Source: http://www.geeksforgeeks.org/dynamic-programming-set-1/

Test Yourself on Dynamic Programming!

- True or False: Dynamic programming can be used to reduce the order of algorithmic complexity of sorting a list of integers to something below n log n, where n is the length of the list to be sorted.

- True or False: Dynamic programming can be used to solve any problem that a greedy algorithm can be used for.

Test Yourself on Dynamic Programming!

- Dynamic programming can be used to reduce the order of algorithmic complexity of sorting a list of integers to something below n log n, where n is the length of the list to be sorted. **FALSE**

- Dynamic programming can be used to solve any problem that a greedy algorithm can be used for. **FALSE**

Comparison Summary

	Complexity	Guaranteed Globally Optimal	Violates Constraints
Brute Force	O(2^n)	Yes	No
Greedy Algorithms	O(n)	No	No
Dynamic Programming	O(n^c), c constant	Yes	No

Lambda Functions

Lambda Functions

- “Anonymous” Functions
- Useful when only plan on using a function once
- You don’t ever *need* to use a lambda, sometimes it is just more concise or easier to understand
- `lambda arguments: expression`

- `div = lambda x, y : x//y`

-
  ```
  def div(x,y):
      return x//y
  ```

Randomness

Randomness

- When do we use it?
  - Generate large, unbiased datasets.
  - Model systems with choices in state transitions.
  - Draw statistical conclusions about the likelihood of outcomes.
  - Model imprecision in measurements
- Useful functions
  - random.random(...), random.choice(...), random.seed(...), random.randint(...)

Test Yourself!

Consider the following function. Is it deterministic?

```
def some_number():

    random.seed(0)

    return 10 * random.randint(1,2)
```

Test Yourself!

Consider the following function. Is it deterministic?

```
def some_number():

    random.seed(0)

    return 10 * random.randint(1,2)
```

YES.

Graphs & Trees

Graphs

- Undirected vs. directed (digraph)

- Weighted vs. unweighted

- Adjacency matrix vs. list (of neighbors)

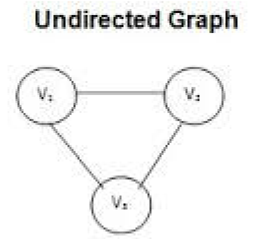


Figure 1: An Undirected Graph

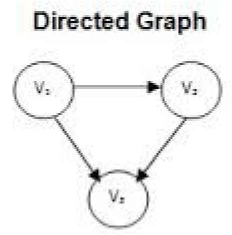
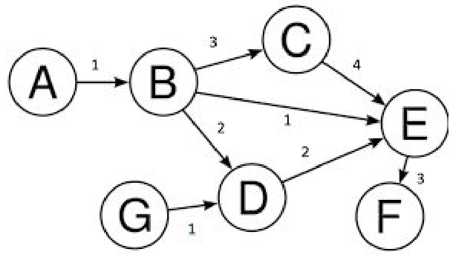
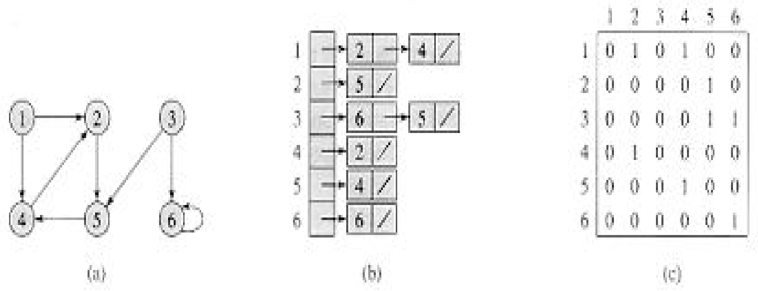


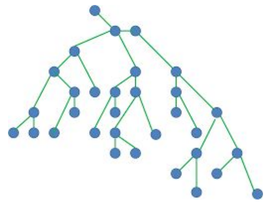
Figure 2: A Directed Graph



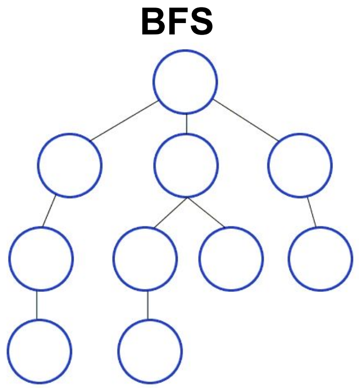


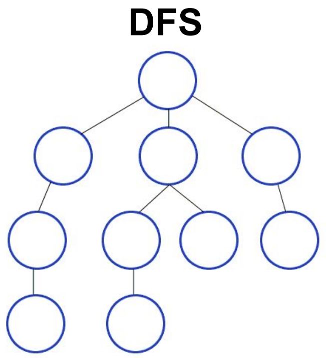
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Trees and Search



- Tree: undirected graph without cycles, every pair of nodes has only a single path between them
- Breadth vs. depth-first search
    - BFS (queue, FIFO): guaranteed shortest path for unweighted graphs
    - DFS (stack, LIFO): returns a path (not necessarily optimal)





Test Yourself!

Suppose you have a weighted directed graph and want to find a path between nodes A and B with the smallest total weight. Select the most accurate statement.

a) If some edges have negative weights, depth-first search finds a correct solution.

b) If all edges have weight 2, depth-first search guarantees that the first path found to be is the shortest path.

c) If some edges have negative weights, breadth-first search finds a correct solution.

d) If all edges have weight 2, breadth-first search guarantees that the first path found to be is the shortest path.

Test Yourself!

Suppose you have a weighted directed graph and want to find a path between nodes A and B with the smallest total weight. Select the most accurate statement.

a) If some edges have negative weights, depth-first search finds a correct solution.

b) If all edges have weight 2, depth-first search guarantees that the first path found to be is the shortest path.

c) If some edges have negative weights, breadth-first search finds a correct solution.

d) If all edges have weight 2, breadth-first search guarantees that the first path found to be is the shortest path.

Central limit theorem, SE, confidence interval

Sampling

- Important because it is often not possible to observe entire population
- It’s never possible to guarantee perfect accuracy through sampling
- How many samples do we need to look at before we can have justified confidence in our answer?
  - Depends on variability in underlying distribution

Example = 1 data point, sample = set of examples

Variation in Data (Variance, Standard Deviation)

$$variance(X) = \frac{\sum_{x \epsilon X}(x - \mu)^2}{|X|}$$

$$\sigma(X) = \sqrt{\frac{1}{|X|}\sum_{x \in X}(x - \mu)^2}$$

Measure of how much spread there is in the data

Confidence interval

- Range that measures how uncertain we are about our estimate
- The 95% confidence level of sample mean says that 95% of the time when this process is repeated (a sample is taken, the mean is computed, and the 95% confidence interval is computed), the confidence interval will encompass the true (population) mean

Standard Error of the Mean (SEM)

- SEM = $\sigma/\sqrt{n}$ (σ is population s.d. -> unknown)
- Approximation
  - With one sample: $s/\sqrt{n}$
    - s: sample s.d. (s.d. of examples in this sample)
    - Reasonable approximation when n is large enough
  - Many samples (m): CLT
    - *For large enough m and n, mean of these m samples will be approximately normally distributed, with mean close to population mean and s.d. close to $\sigma/\sqrt{n}$*
    - Use s.d. of these samples as SEM

Central Limit Theorem

1) The means of the samples in a set of samples (the sample means) will be approximately normally distributed,
2) This normal distribution will have a mean close to the mean of the population, and
3) The variance of the sample means will be close to the variance of the population divided by the sample size.

Test Yourself!

- True or False: For a given standard error, lower confidence levels produce wider confidence intervals.

- True or False: The statement, "the 95% confidence interval for the population mean is (350, 400)", is equivalent to the statement, "there is a 95% probability that the population mean is between 350 and 400".

Test Yourself!

- For a given standard error, lower confidence levels produce wider confidence intervals. **FALSE**

- The statement, "the 95% confidence interval for the population mean is (350, 400)", is equivalent to the statement, "there is a 95% probability that the population mean is between 350 and 400". **FALSE**

Curve Fitting and Linear Regression

Curve Fitting and Linear Regression

• Finding a best fit curve is an optimization problem, where the objective function is usually least squares $\sum_{i=1}^{n}(observed_i - predicted_i)^2$

• pylab.polyfit(x,y,degree) returns the coefficients of a polynomial of degree "degree" that best fits the x,y data.

• Example of linear regression – fitting a polynomial of any order

Curve Fitting and Linear Regression

- There are numerical ways to find the “goodness” of a fit curve
- Can use the Mean Squared Error (least squares)
  - No upper limit
  - Only good for comparing two curves

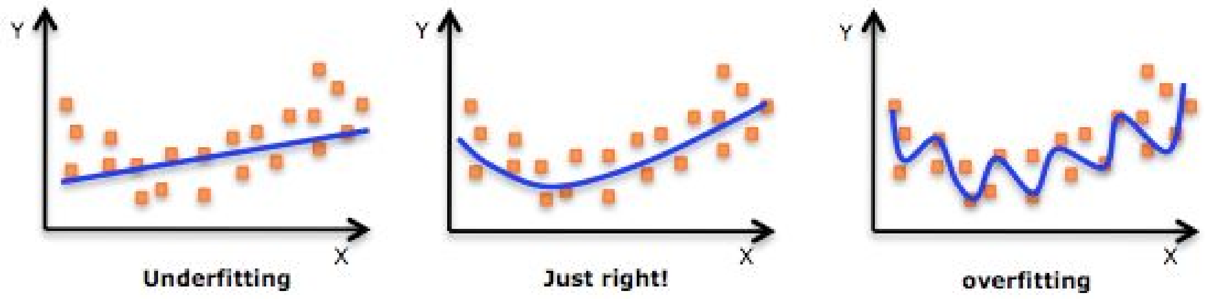
$$\frac{1}{n}\sum_{i=1}^{n}(observed_i - predicted_i)^2$$

- Can use the Coefficient of Determination (R^2) – a measure of how much of the variance is explained by the model
  - 0 means none of the variance is explained
  - 1 means all of the variance is explained (which is not necessarily good -- noise!!!)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(observed_i - predicted_i)^2}{\sum_{i=1}^{n}(observed_i - meanOfObserved)^2}$$

Curve Fitting and Linear Regression

- We can also describe a fit as underfitting or overfitting
    - **underfitting** is basically not capturing the trend
    - **overfitting** is basically capturing noise as well as the trend



Test Yourself on Regression!

- A linear regression model with an R^2 value of 0.8 for the training data explains 80% of the variation in the training data.

- When choosing a degree for polyfit, it is good practice to choose a degree that maximizes R^2 on the training data.

- Given a finite set of data points there exists a polynomial fit such that the polynomial curve goes through each point in the data.

Test Yourself on Regression!

- A linear regression model with an R^2 value of 0.8 for the training data explains 80% of the variation in the training data. **TRUE**

- When choosing a degree for polyfit, it is good practice to choose a degree that maximizes R^2 on the training data. **FALSE**

- Given a finite set of data points there exists a polynomial fit such that the polynomial curve goes through each point in the data. **TRUE**

Probability

Probability

- Always in the range [0,1]
    - 0 if impossible, 1 if certain
- If a certain event has probability p, the inverse event has probability 1-p
    - Useful when a probability is not computable in a straightforward way
- **Example:** given a deck of cards, what's the probability that you draw a royal flush?

Probability

- Always in the range [0,1]
  - 0 if impossible, 1 if certain
- If a certain event has probability p, the inverse event has probability 1-p
  - Useful when a probability is not computable in a straightforward way
- **Example**: given a deck of cards, what's the probability that you draw a royal flush?

  Number of different 5-card sets: (52*51*50*49*48)/(5*4*3*2*1) = 2,598,960

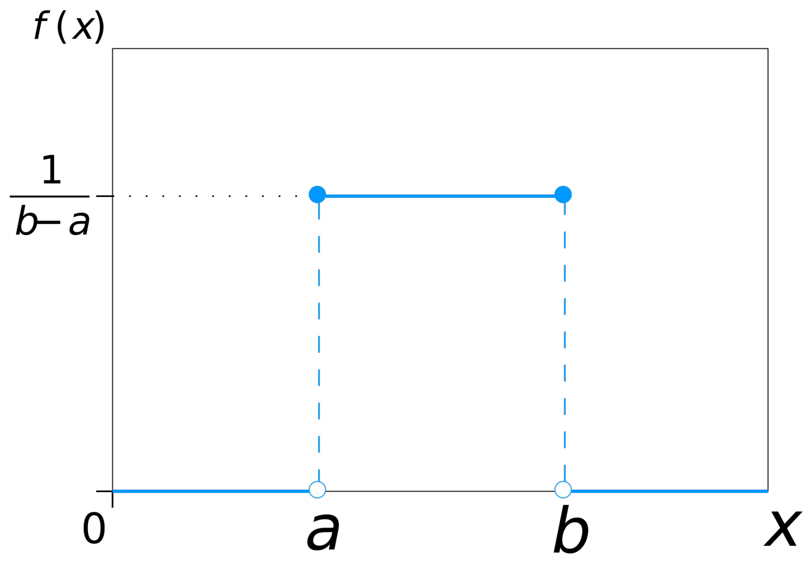
  Number of royal flushes: 4
  Probability: 4/2,598,960 = 0.0000015

Probability Distributions

- Probability Distribution Function (**PDF**)
- A function that defines the probability of the value of a random variable
- Area under the curve gives the probability of the value being in a *range*
  - The value of the function itself is *not* the probability!
- Area under the curve over the whole x-axis always adds up to 1
- We know Uniform, Gaussian (Normal), but many others also exist

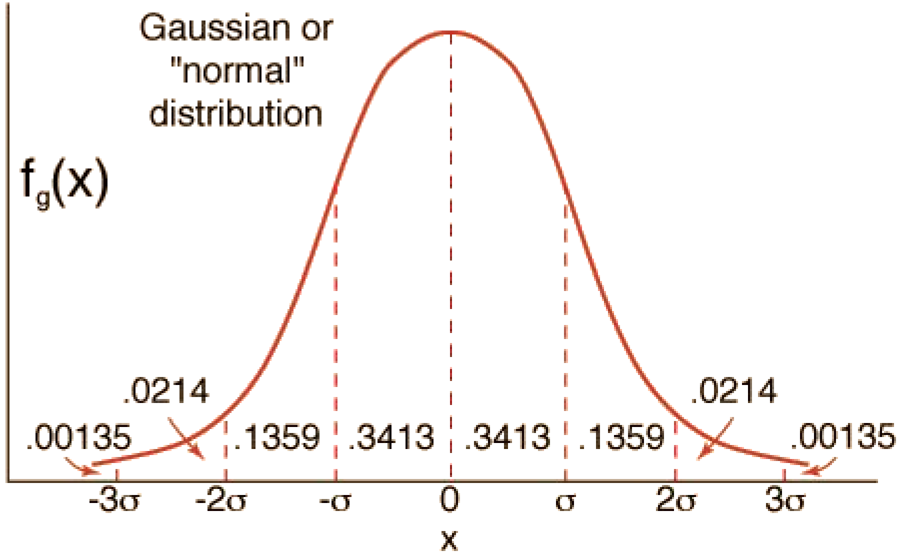
Probability Distributions: Uniform

- Has a constant value over a continuous range
- Simplest probability distribution



Probability Distributions: Gaussian

- “Bell curve”
- Many examples in nature, data
- Usually defined as (mean, std)
    - (0,1) is the standard *normal* distribution
- “Empirical rule” allows us to relate data points to overall structure
    - 68-95-99.7



Probability Distributions: Others

- Binomial
  - Probability k out of n independent trials succeed
  - Useful to model: Rolls of a die, flips of a coin.
- Exponential
  - Probability that event with prob p in each timestep has not happened after t timesteps
  - Useful to model: Arrival times, molecules decaying, etc.

Test Yourself!

- True or False: The standard deviation fully defines a uniform distribution.

- An unfair coin with p(heads) = 3/4 is tossed 10 times. What is the probability that exactly 6 heads will occur?

Test Yourself!

- The standard deviation fully defines a uniform distribution **FALSE**.

- An unfair coin with p(heads) = 3/4 is tossed 10 times. What is the probability that exactly 6 heads will occur? **10c6 * (3/4)^6 * (1/4)^4**

Probability Distributions: Demo

Demo

List Comprehension

List Comprehension

- Creating a list that is an arithmetic expression of another list or range
- Faster than using a for loop
- You don’t ever *need* to use list comprehension, sometimes it is just more concise or easier to understand

-
  ```
  Doubled = [2*x for x in range(10)]
  ```

-
  ```
  Doubled = []

  for element in range(10):

      Doubled.append(2*x)
  ```

Monte Carlo Simulations and Law of Large Numbers

Law of Large Numbers (Bernoulli’s Theorem)

In repeated independent experiments, the average value of the experiment approaches the expected value as the number of experiments goes to infinity.

Note: This does not imply that if deviations from expected behavior occur, these deviations are likely to be evened out by opposite deviations in the future. (**gambler’s fallacy**)

Law of Large Numbers (Bernoulli's Theorem)

[Example] Flipping a Fair Coin

- Expected fraction of Heads: 0.5

Law of Large Numbers (Bernoulli’s Theorem)

[Example] Flipping a Fair Coin

- Expected fraction of Heads: 0.5
- The Law of Large Numbers suggest that as you perform more trials, the fraction of Heads will get closer to 0.5.

Law of Large Numbers (Bernoulli's Theorem)

[Example] Flipping a Fair Coin

- Expected fraction of Heads: 0.5
- The Law of Large Numbers suggest that as you perform more trials, the fraction of Heads will get closer to 0.5.
- Getting 10 Heads in a row does not increase the probability of getting a Tail on the next flip. (gambler's fallacy)

Law of Large Numbers (Bernoulli’s Theorem)

[Example] Roulette

- Following an extreme random event, the next random event is likely to be less extreme **(regression to mean)**
- If we spin the wheel ten times and get 9 blacks, we are likely to get fewer than 9 blacks on the next ten spins
- Is this not gambler’s fallacy???

Law of Large Numbers (Bernoulli's Theorem)

[Example] Roulette

- Following an extreme random event, the next random event is likely to be less extreme **(regression to mean)**
- If we spin the wheel ten times and get 9 blacks, we are likely to get fewer than 9 blacks on the next ten spins
- Is this not gambler's fallacy??? **No**

Monte Carlo

Can help in estimating the value of an unknown quantity using the principles of inferential statistics.

Population: a set of examples

Sample: a proper subset of a population

A **random sample** tends to exhibit the same properties as the population from which it is drawn

Useful when analytical solution is difficult or impossible

Lies, damned lies, and statistics

- Be careful in making conclusions from your data or reading others' conclusion
- Truncating y-axis
- Non-representative Sampling
  - Land-lines
  - Course evaluations
  - Survivor's Bias