APS1070

Foundations of Data Analytics and Machine Learning

Winter 2022

Week 12:
- *More Concepts*
- *Final Assessment Details*
- *Past Final Exam Questions*

Sinisa Colic and Samin Aref

More Concepts Con’t

Overview

➢ There are several concepts related to data science that deserve some consideration:

- **Discrete Optimization**
- Sampling Methods
- Monte Carlo Sampling
- Hypothesis Testing

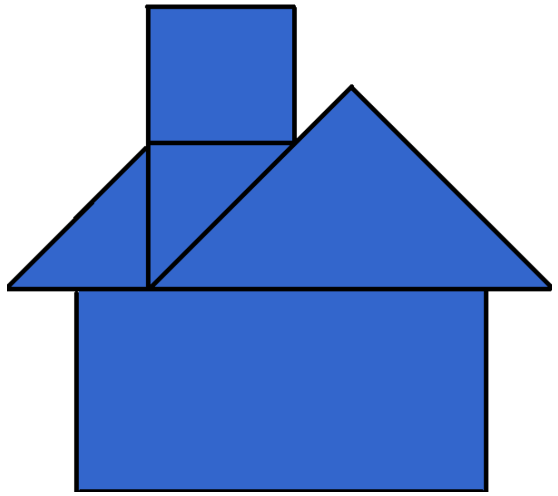
This Week

Sampling Methods

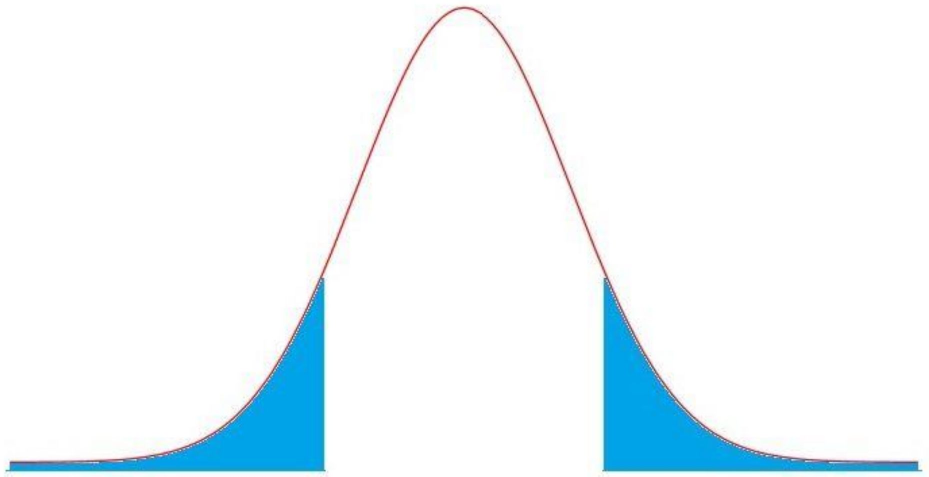
Why Sampling Methods?

- Q: How would you solve the following problems?

1. Compute area



2. Compute the blue area



3. Integrate f(x)

$$f(x) = x^2$$

$$\int f(x)dx$$

- Most of the examples we've seen so far can be solved analytically.

Why Sampling Methods?

- Q: How would you solve the following problems?

Compute Area

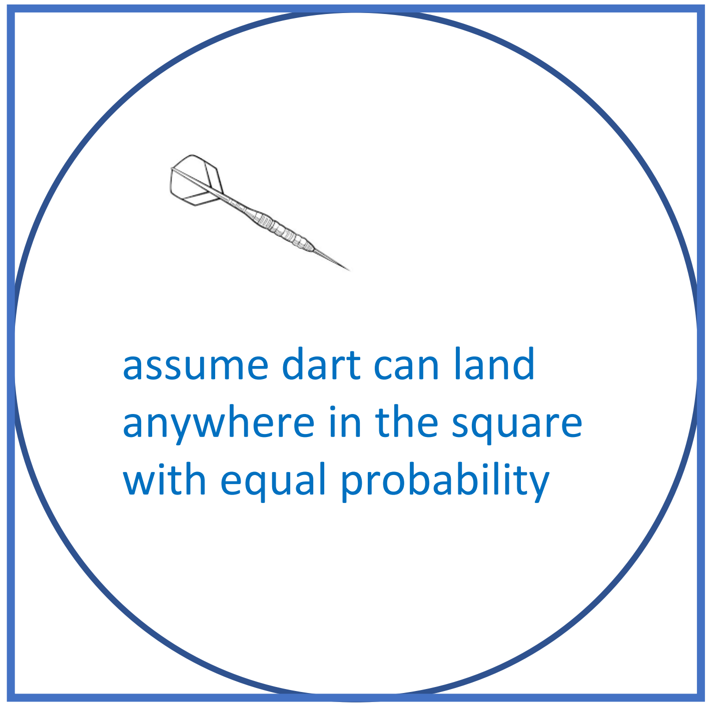
Integrate a function

$$\int_0^4 \sqrt[4]{15x^3 + 21x^2 + 41x + 3}.e^{-0.5x}dx$$

- For most problems of practical interest, the analytical solution is intractable, and so we have to resort to some form of approximation.

Simple Illustration

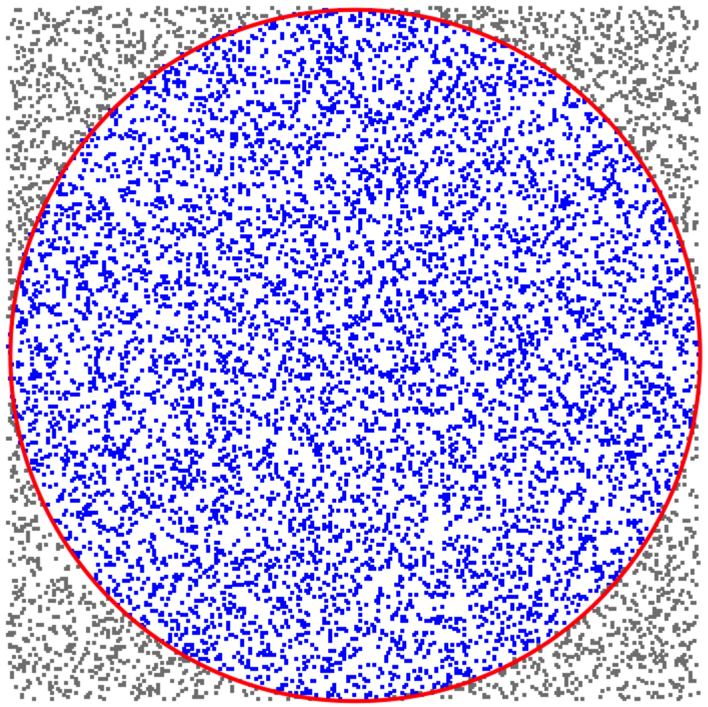
- Q: What is the probability of a dart landing in the circle?


assume dart can land anywhere in the square with equal probability

- Simulate many random 2D points
- Use $x^2 + y^2 \leq r^2$
- Then apply $\frac{N_{in}}{N_{tot}}$ to estimate area

Simple Illustration: Simulation

- Q: What is the probability of a dart landing in the circle?



------------ Simulation Code ------------

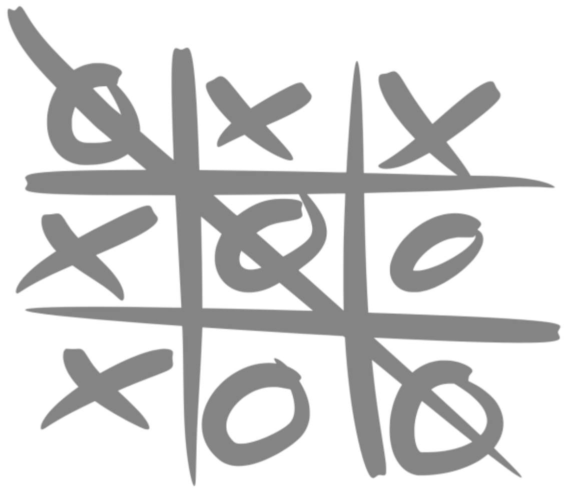
```
def throwDarts(num_darts):
   in_circle = 0
   for darts in range(1,num_darts + 1, 1):
      x = random.random()
      y = random.random()
      if (x*x + y*y)**0.5 <= 1.0:
         in_circle +=1
   return (in_circle/num_darts)
```

```
Estimate:   0.78308
Truth:  0.7853981633974483
```

How can we obtain π?

Harder Example

- Q: Assume you are playing tic-tac-toe and you have a P(win) = *p*. The game ends when you lose 2 times in a row. What is the expected number of rounds we will play the game before losing?
- Coming up with an analytical solution can take a long time. In the process we may make a mistake without realizing it.
- Is there a way we can obtain an estimate?



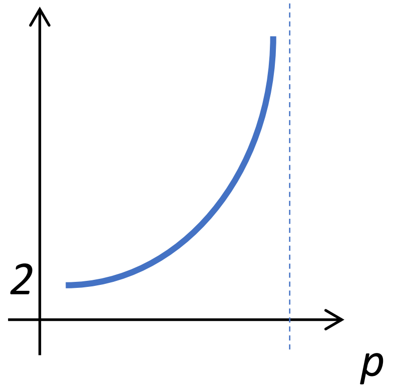
Harder Example: Simulation

------------- Simulation Code ------------

```python
import numpy as np
def mc(n=10**6,p=0.5):
    rounds= []
    for _ in range(n):
        r,losses = 0,0
        while losses != 2:
            r+=1
            if np.random.random() <= p:
                losses = 0
            else:
                losses +=1
        rounds.append(r)
    return np.mean(rounds)
```

Analytical Solution

$$\text{rounds} = \frac{2-p}{(1-p)^2}$$



```
Estimate: 5.996614
Truth: 6
```

Source: ritvikmath

Monte Carlo Sampling Methods

- It was invented by Stanislaw Ulam during the Manhattan Project.
  - e.g., asses the risk of a runaway chain reaction that could blow up the earth.
- Used extensively in:
  - risk analysis,
  - finance,
  - supply chain logistics,
  - computational physics,
  - robotics

Monte Carlo is not a single method but instead any type of method that relies on **simulations and randomness** to get the solution to a problem.

Law of Large Numbers

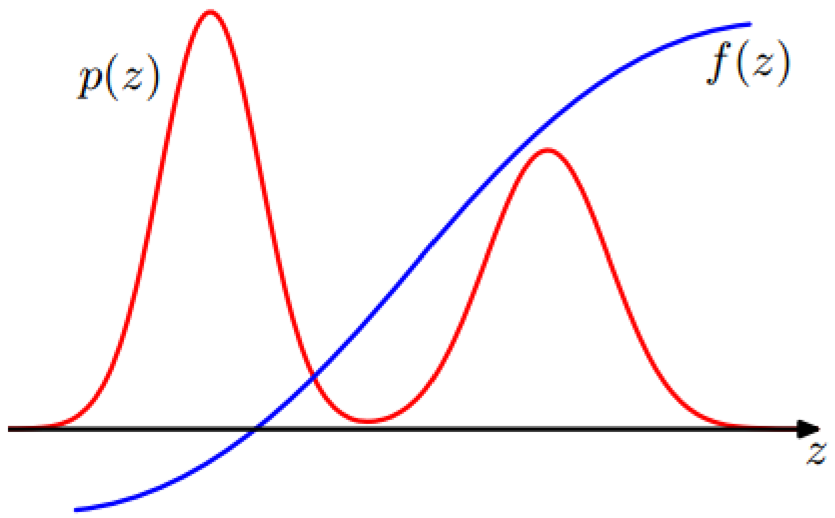
- The Law of Large Numbers describes what happens when performing the same experiment many times.
- After many trials, the average of the results should be close to the expected value and will be more accurate with more trials.
- For Monte Carlo simulation, this means that we can learn properties of a random variable (mean, variance, etc.) simply by simulating it over many trials.

Pros and Cons

- Pros:
  - Monte Carlo simulation is often easy to implement.
  - Doesn’t matter if you are subject matter expert. It converges to similar results without requiring analytical solutions.
- Cons:
  - It isn’t always fast, will depends on the problem (e.g., p=0.999 in our game example will take a long time to finish).
  - Not generalizable, just because you have one result for one set of parameters, it doesn’t say anything about other parameters unlike analytical solution.
  - Not interpretable.

Why Sampling Methods?

- For most probabilistic models of practical interest, exact inference is intractable, and so we have to resort to some form of approximation.
- Here we will focus primarily on the problem of estimating expectations for intractable distributions:



$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}$$

Comes up frequently in Bayesian modelling

Partition Function

- A common problem in machine learning is that the partition function is often intractable (or difficult to compute):

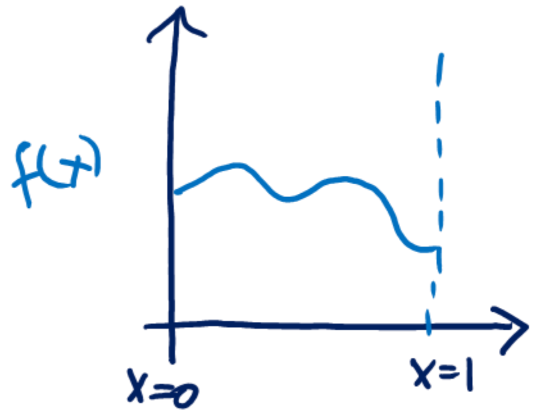
$$p(z|x) = \frac{p(x|z)p(z)}{\underbrace{\int_z p(x|z)p(z)dz}_{p(x)}}$$

partition function

$p(x)$

In circumstances where we are unable to sample directly from the distribution *p*(z) we can use an alternative approach known as importance sampling.

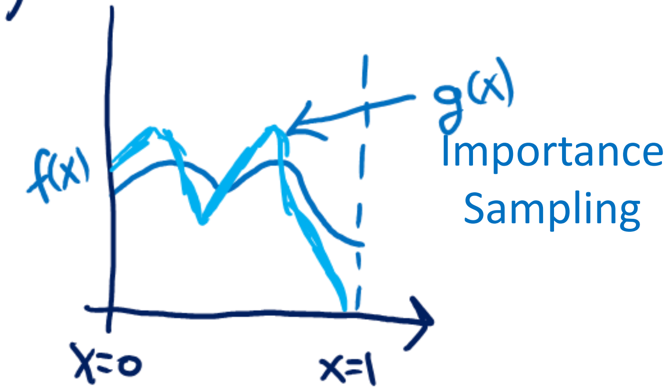
Example: Sampling (i)



$$\int_0^1 f(x)\,dx = ?$$

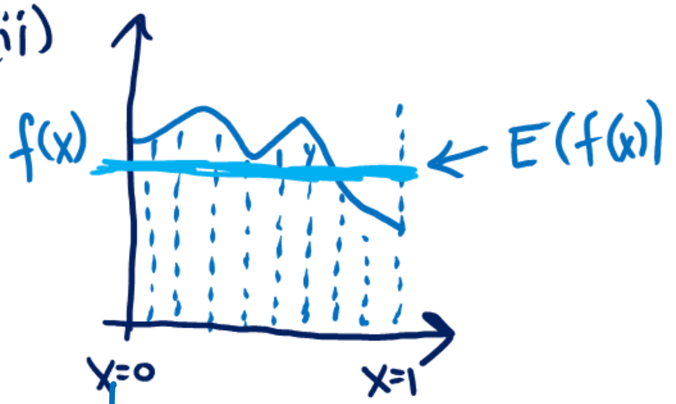
Approx. using g(x)

(i)



Approx. by taking samples

(ii)



$$\int_0^1 f(x)\,dx \approx E(f(x))$$

(ii) estimate average by taking N samples

$$\int_0^1 f(x)\,dx = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

if we have $I = \int_a^b f(x)\,dx$

then

$$I_m = (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Monte Carlo estimate

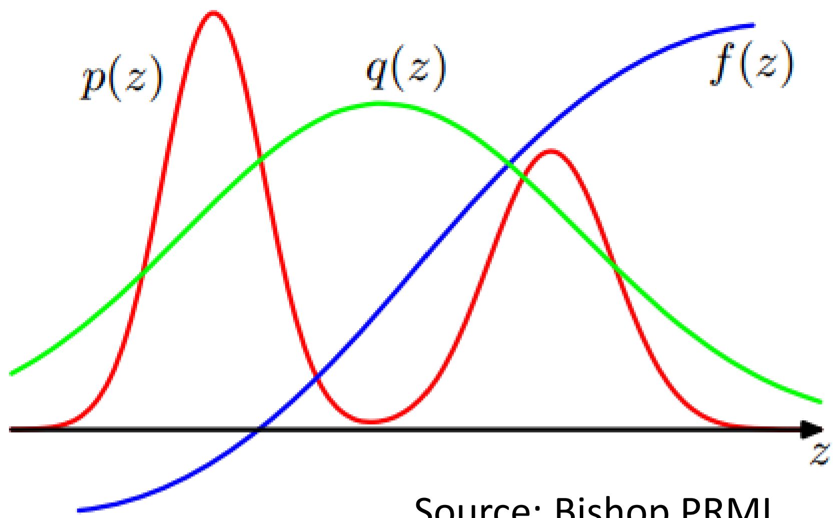
$x_1, x_2 \ldots x_N$ are uniformly distributed between a & b

$$\lim_{N\to\infty} I_m = I$$

Importance Sampling

- Draws samples from a simpler distribution q(z) to estimate expectation of f(x)
- Corresponding terms in the summation are weighted by the ratios p(x)/q(x)

$$\begin{aligned}\mathbb{E}_p[f(x)] &= \sum_x p(x) f(x) \\ &= \sum_x q(x) \frac{p(x) f(x)}{q(x)} \\ &= \mathbb{E}_q\left[\frac{p(x)}{q(x)} f(x)\right],\end{aligned}$$



Source: Bishop PRML

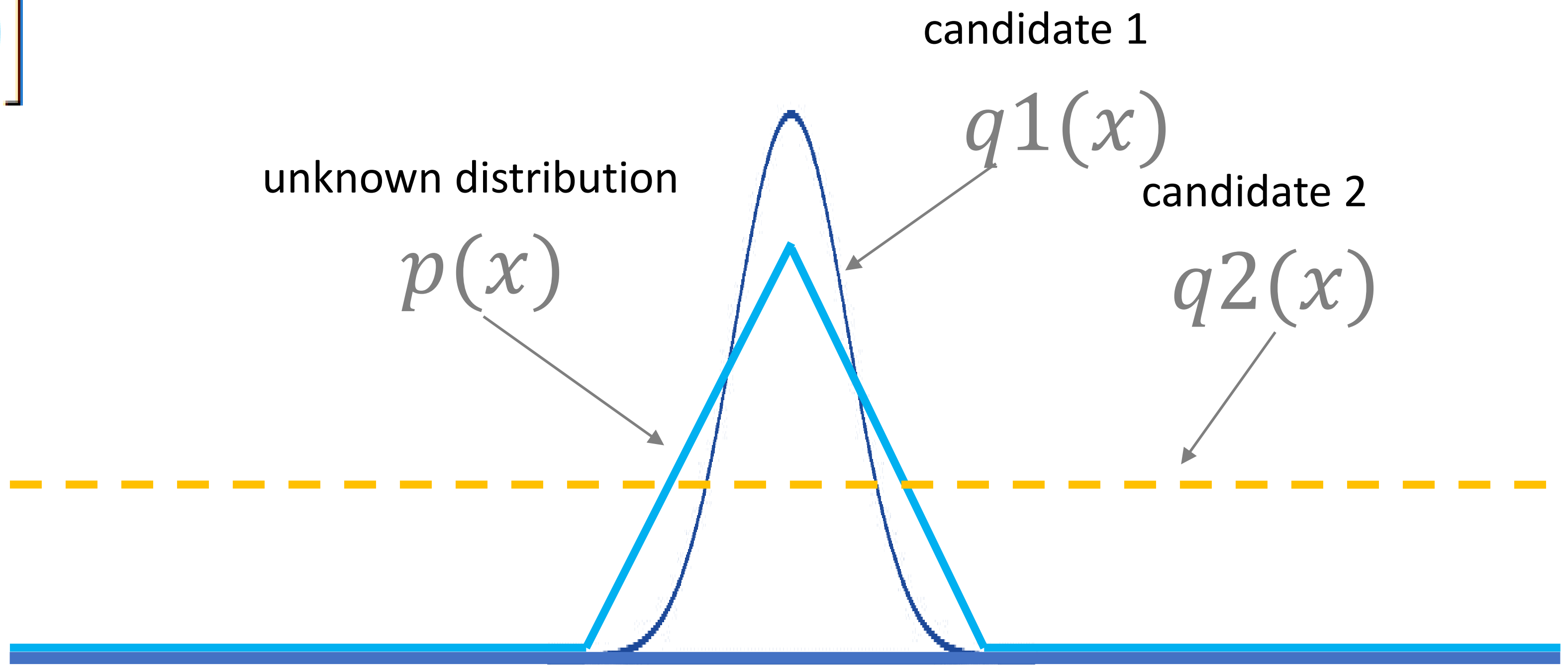
Limitations Con’t

- Depends on how well the sampling distribution q(z) matches the desired distribution p(z).
- **Importance weights may be dominated by a few examples** having large values (remaining weights insignificant)
- Major drawback of the importance sampling method is the potential to produce results that are arbitrarily in error and with **no diagnostic indication**.
- Should not be small or zero where p(z) may be significant.

Source: Bishop PRML

Which candidate distribution is best?

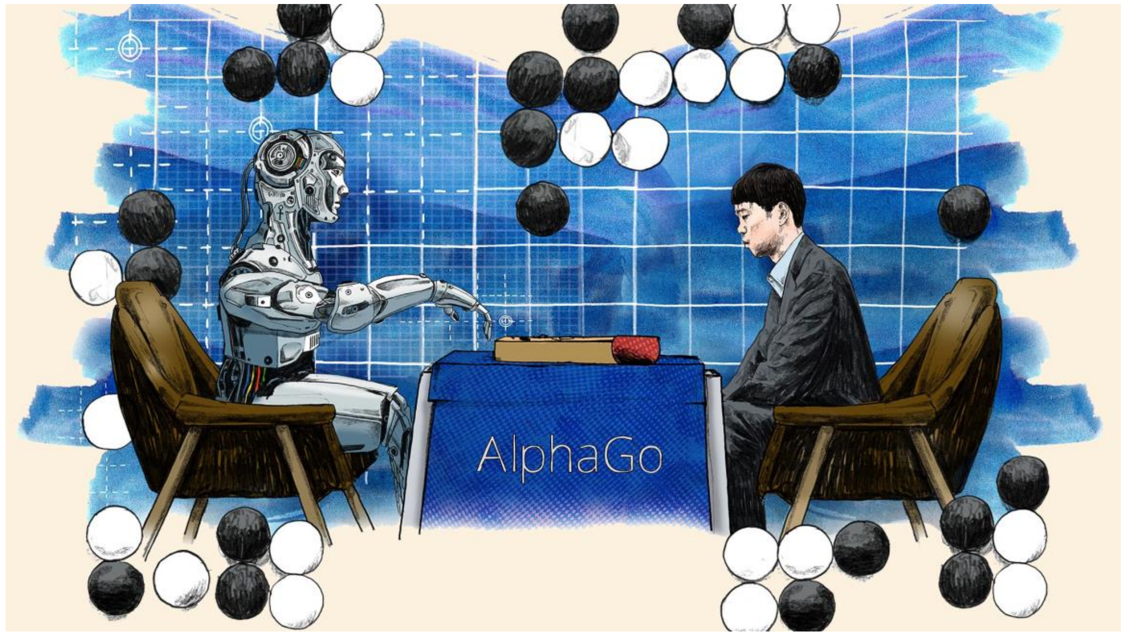
$$\mathbb{E}_q\left[\frac{p(x)}{q(x)}f(x)\right]$$



Source: Ben Lambert

Link to Deep Learning

- Leads into more advanced methods such as Markov Chain Monte Carlo (MCMC).
- A whole course could be devoted to this topic.
- DeepMind used a version of Monte Carlo Sampling in AlphaGo.



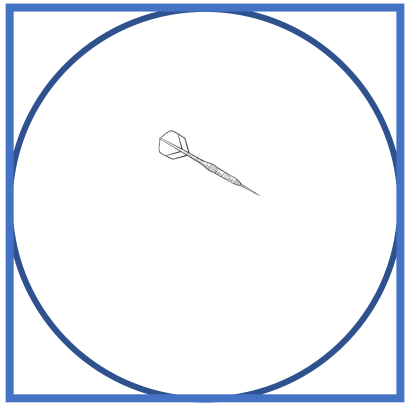
Source: Level Up Coding

Hypothesis Testing

Confidence Intervals

- Let us consider our example from earlier, but this time we will use it to estimate pi = 3.14159…

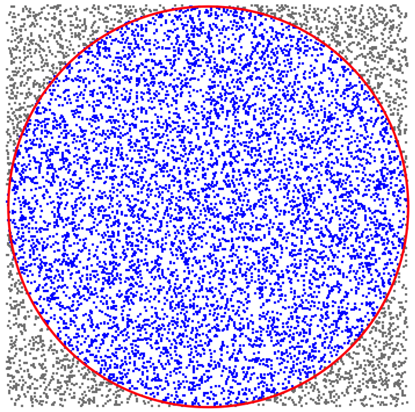
------------- Simulation Code ------------



```
def throwDarts(num_darts):
  in_circle = 0
  for darts in range(1,num_darts + 1, 1):
    x = random.random()
    y = random.random()
    if (x*x + y*y)**0.5 <= 1.0:
      in_circle +=1
  return (4*in_circle/num_darts)

Estimate:  3.13868
Truth:  3.141592653589793
```

one estimate



Confidence Intervals

- The equation for obtaining the confidence intervals is:

sample mean

$$\bar{x} = \frac{1}{n}\sum_i x_i$$

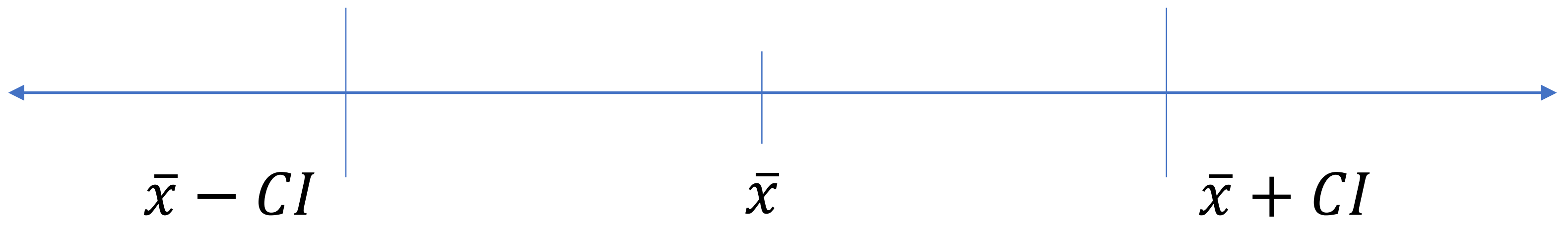
sample standard deviation

$$s^2 = \sum_i \frac{(x_i - \bar{x})^2}{n-1}$$

$$CI = \bar{x} \pm z\frac{s}{\sqrt{n}}$$

confidence level value
(typically set at 95%)

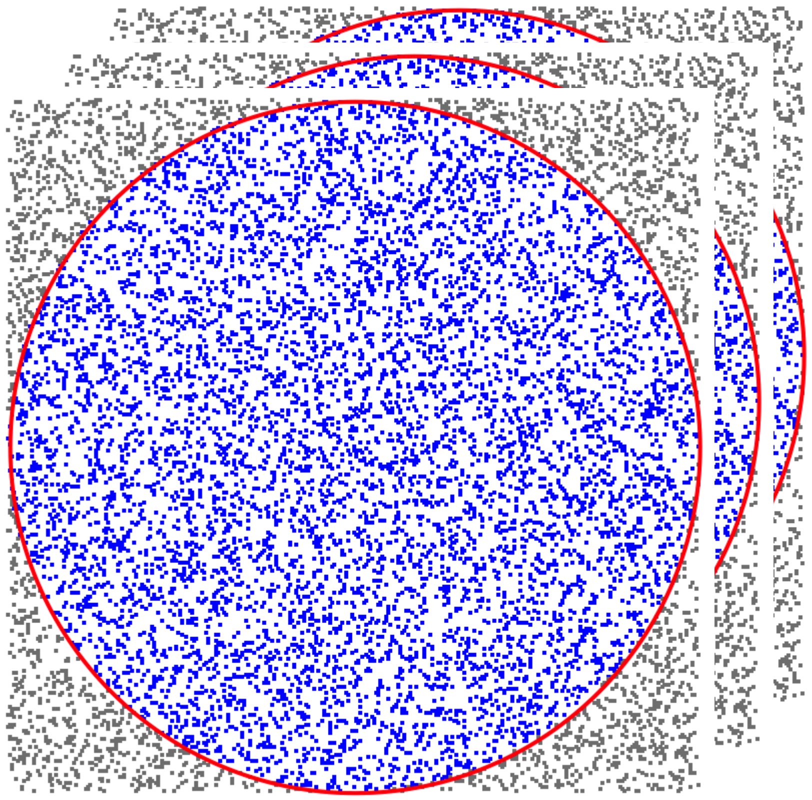
sample size



Obtain More Samples

- Let us test this by repeating our simulation 100 times:

Repeat multiple times



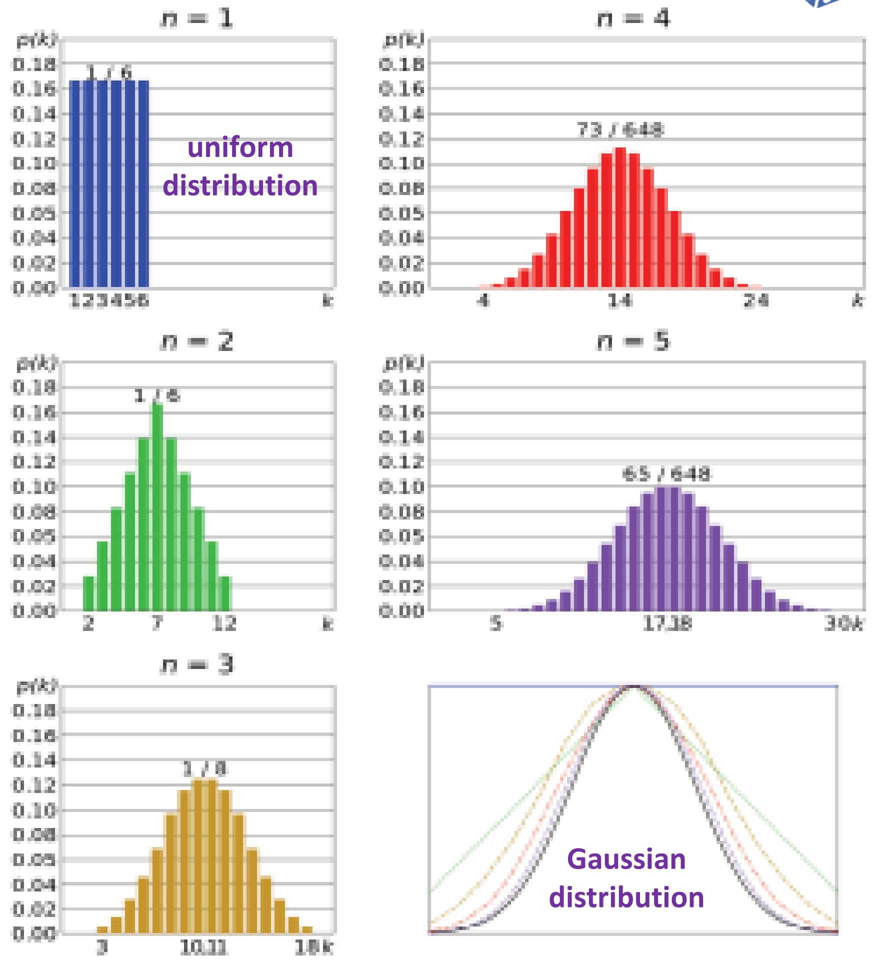
------------ Simulation Code -----------

```
def getEst(num_darts, num_trials):
  estimates = []
  for t in range(num_trials):
    guess = throwDarts(num_darts)
    estimates.append(guess)
  s_dev = np.std(np.array(estimates),ddof=1)
  s_err = s_dev/(len(estimates)**.5)
  cur_est = sum(estimates)/len(estimates)
  return (s_mean,s_err)
```

Central Limit Theory

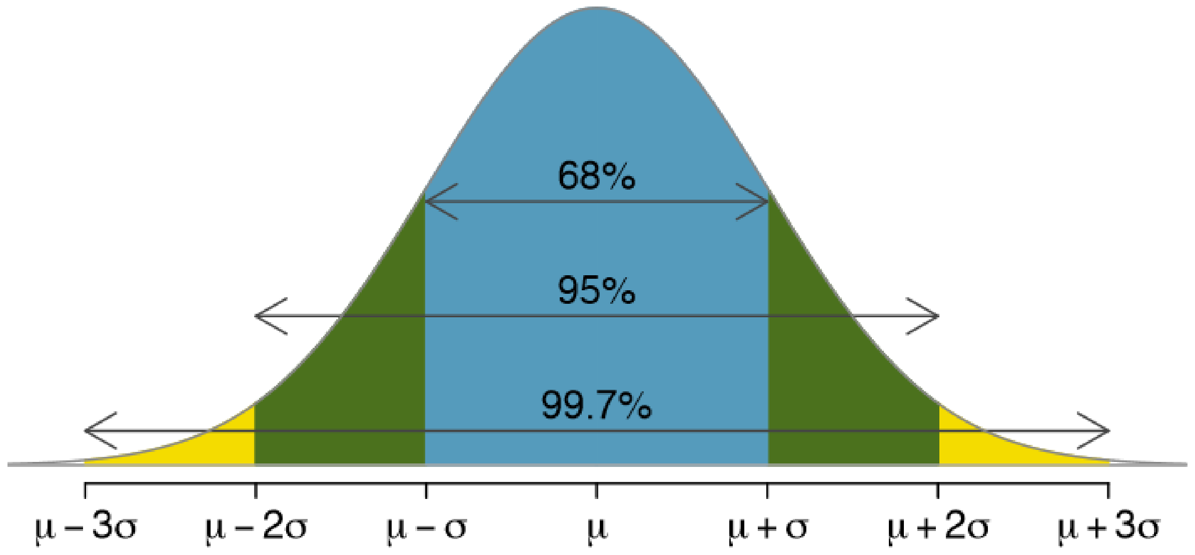
- Q: What happens when you perform a measurement multiple times and compute the average.
- A: By the central limit theory the **distribution will become Gaussian**, and the mean will be close to the population (true) mean.

Example: rolling a die many times

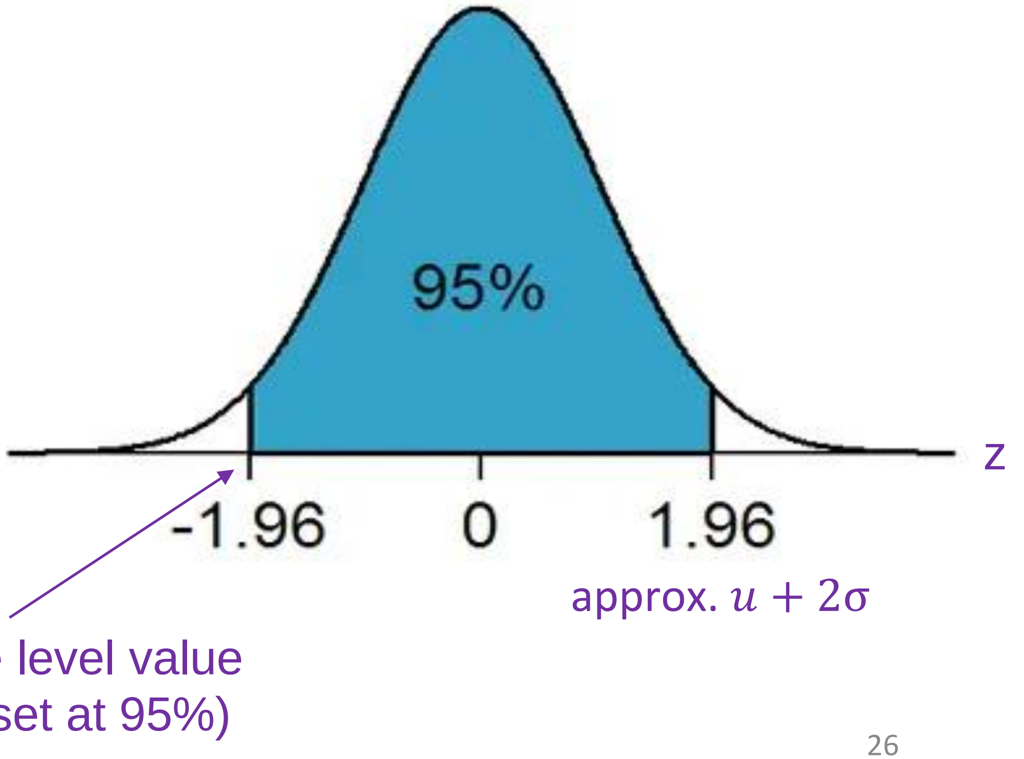


Gaussian Distribution

- What does a 95% confidence really mean?

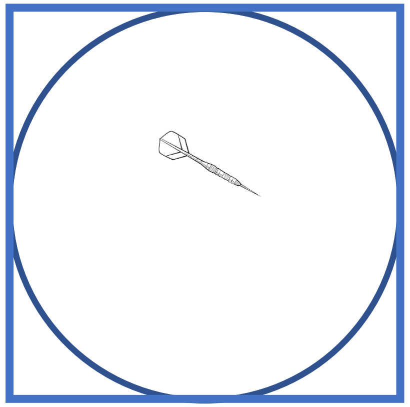


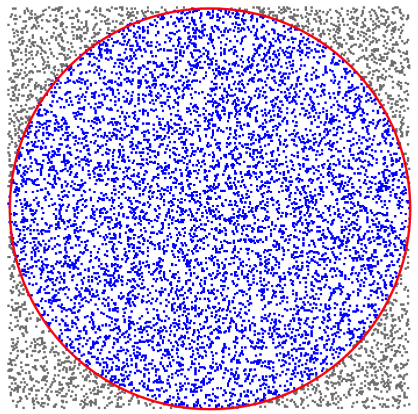
Normalized Gaussian
$(\mu = 0, \sigma^2 = 1)$



Confidence Intervals

- Calculating for our example…





one sample mean

```
Sample Mean:  3.141528
CI:  3.140608  to  3.142448
True Mean:  3.141592653589793
```

sample mean

$$\bar{x} = \frac{1}{n}\sum_i x_i$$

3.1409936

sample standard deviation

$$s^2 = \sum_i \frac{(x_i - \bar{x})^2}{n-1}$$

s = 0.005649

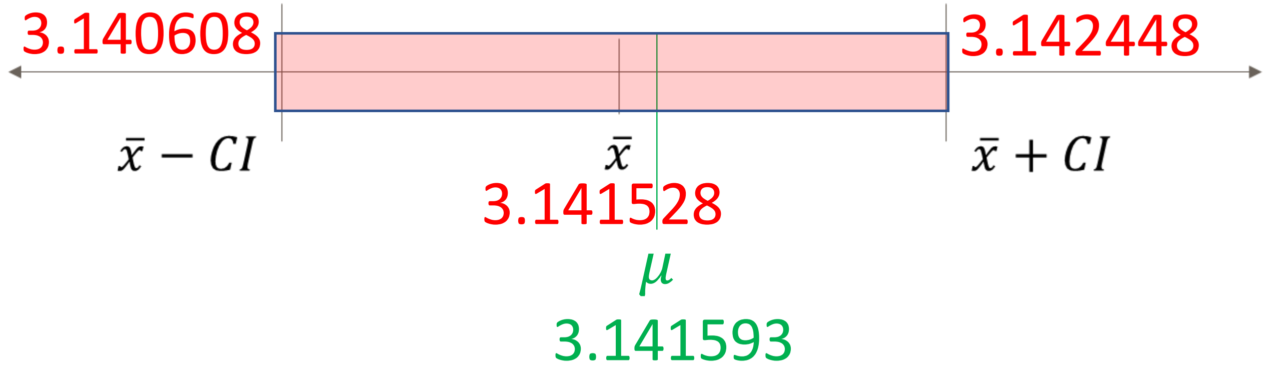
$$CI = \bar{x} \pm z\frac{s}{\sqrt{n}}$$

confidence level value (typically set at 95%)

1.96

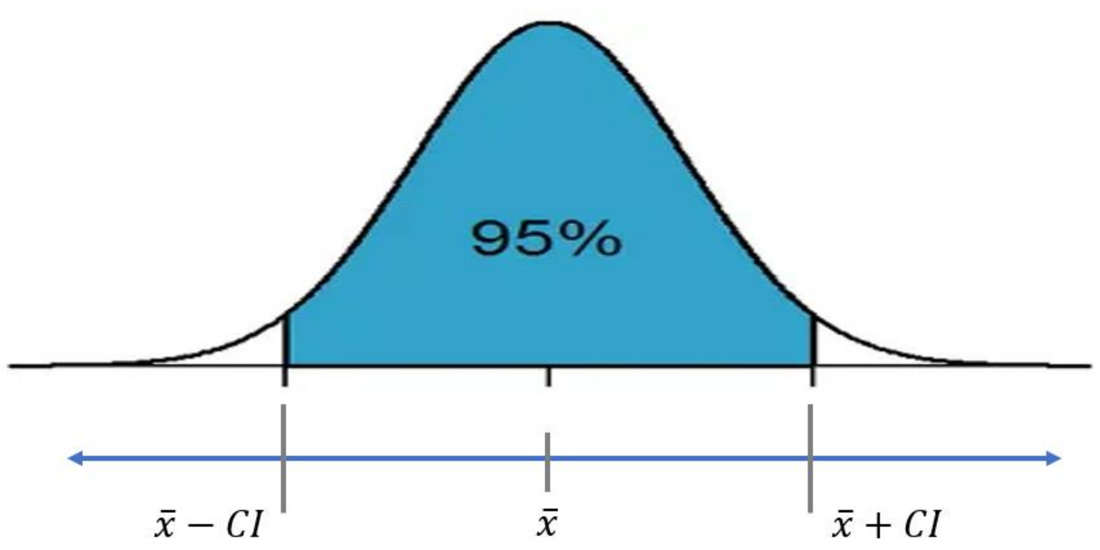
sample size

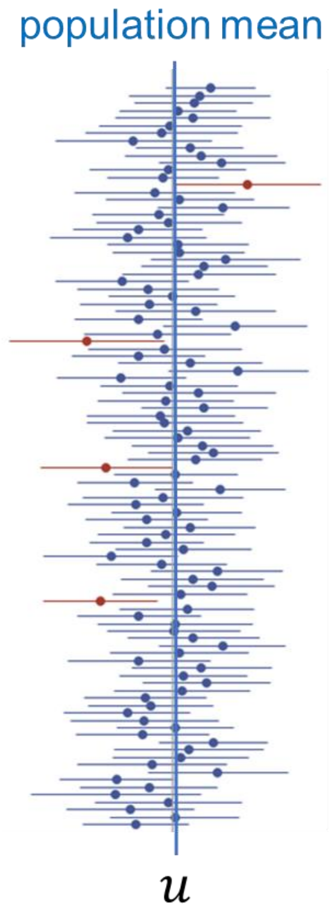
100



Confidence Intervals

- What does a 95% confidence really mean?



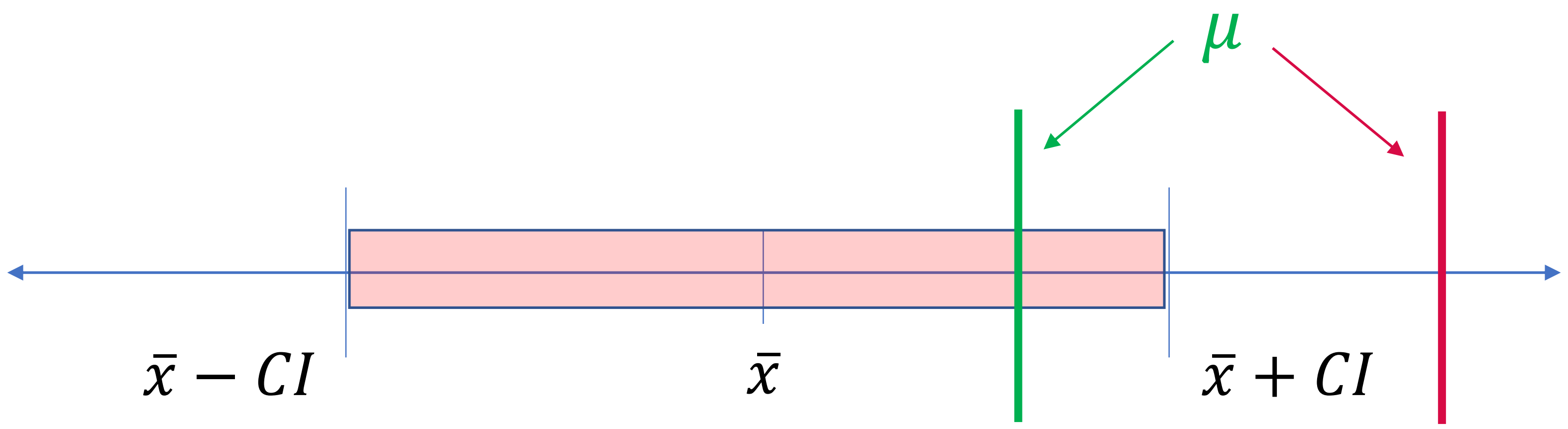


- If we were to repeat the simulation to obtain 100 sample means, then the confidence interval will overlap with the population mean 95 times.
- And 5 of the times the population mean will be outside the interval range.

Let’s test it out in code!

Hypothesis Testing

- Q: How does confidence interval relate to hypothesis testing?



- A: Null hypothesis is that the sample and population mean come from the same distribution. If CI overlaps with population mean, then we accept the hypothesis. Otherwise, we reject the hypothesis.

Hypothesis Testing

- In the example we implanted what is referred to as a z-test. One of the **requirements of a z-test is that we know the population variance**.
  - or often if we have $n > 30$ samples we can assume that the sample variance is close enough.

- Alternatively, we can **apply a t-test** which uses a slightly different distribution that works well when population variance is unknown.

Two or More samples

- We can also do a hypothesis test between two or more samples. That is, we can test if samples have the same population mean (Null Hypothesis):

H_0: $\mu_1 = \mu_2$

H_a: $\mu_1 \neq \mu_2$

two sample t-test
or ANOVA

H_0: $\mu_1 = \mu_2 \ldots = \mu_m$

H_a: $\mu_1 \neq \mu_2 \ldots \neq \mu_m$

ANOVA

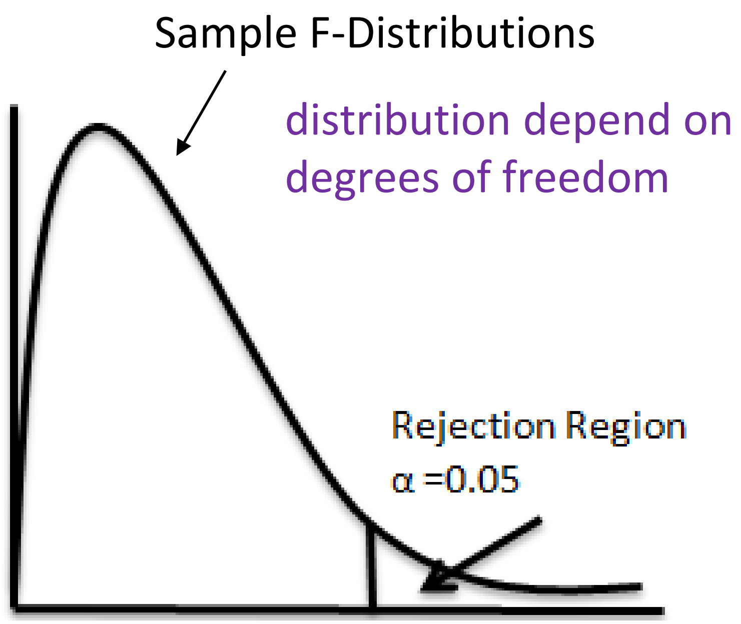
- If we reject the hypothesis then the means are different.

ANOVA

- To perform a statistical test on whether **more than three population means are equal** we can use Analysis of Variance (ANOVA).

- The approach:
  1. Find sum of squared **differences within groups** div. by dof
  2. Find sum of square **differences among group** div. by dof
  3. Ratio provides F value which is compared to the F distribution based on desired critical value or significance.

$$F = \frac{\text{variance among groups}}{\text{variance within groups}} = \frac{\mathrm{SS}_{\text{among}}/(\mathrm{k}-1)}{\mathrm{SS}_{\text{within}}/(\mathrm{N}-\mathrm{k})}$$

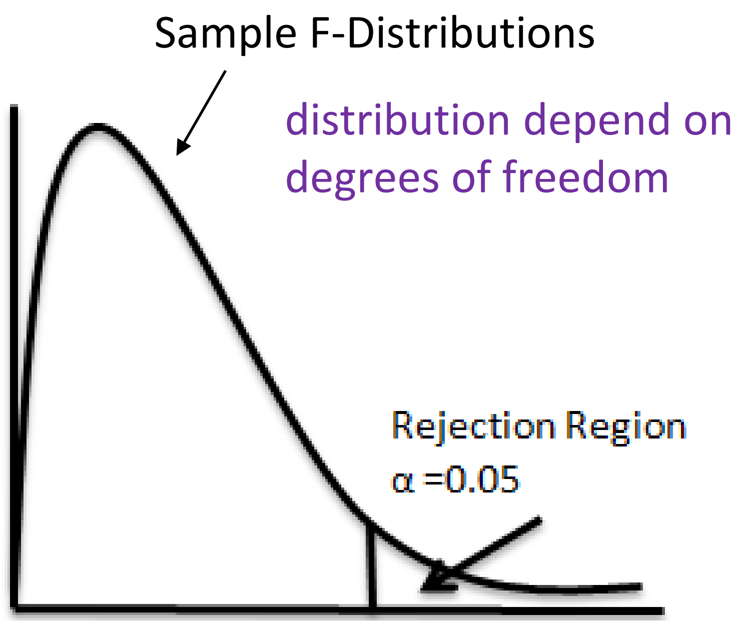


Example 1: ANOVA

Given scores from a test for 9 students: {1,3,4,5,5,5,6,7,9}

	Group I	**Group II**	**Group III**
	{1,5,9}	{4,5,6}	{3,5,7}
	$\overline{X_1} = 5$	$\overline{X_2} = 5$	$\overline{X_3} = 5$
SS_{within} =	$(-4)^2+0^2+4^2$	$(-1)^2+0^2+1^2$	$(-2)^2+0^2+2^2$
	32	2	8
SS_{among} =	$3(0^2)$	$3(0^2)$	$3(0^2)$
	0	0	0

$$F = \frac{\text{variance among groups}}{\text{variance within groups}} = \frac{0/(3-1)}{42/(9-3)} = 0$$

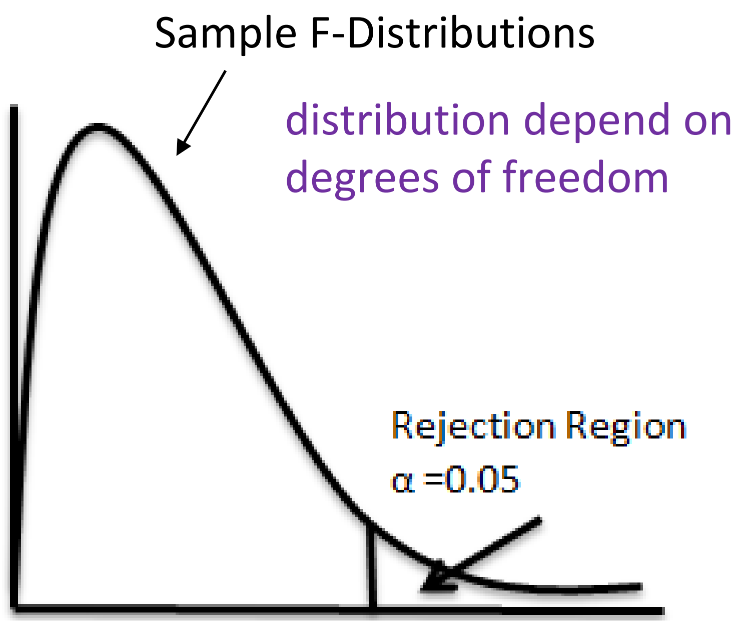


Example 2: ANOVA

Given scores from a test for 9 students: {1,3,4,5,5,5,6,7,9}

	Group I	**Group II**	**Group III**
	{1,3,5}	{5,7,9}	{4,5,6}
	$\overline{X_1} = 3$	$\overline{X_2} = 7$	$\overline{X_3} = 5$
SS_{within} =	$(-2)^2+0^2+2^2$ 8	$(-2)^2+0^2+2^2$ 8	$(-1)^2+0^2+1^2$ 2
SS_{among} =	$3(3-5)^2$ 12	$3(7-5)^2$ 12	$3(5-5)^2$ 0

$$F = \frac{\text{variance among groups}}{\text{variance within groups}} = \frac{24/(3-1)}{18/(9-3)} = 4$$



Application: Feature Selection

- ANOVA can also be used for feature selection
- F-value score -> checks that when grouping the numerical feature by target vector, the means of each group is significantly different.

- Examples of ANOVA for Feature Selection in Machine Learning:
  - https://chrisalbon.com/machine_learning/feature_selection/anova_f-value_for_feature_selection/
  - https://towardsdatascience.com/anova-for-feature-selection-in-machine-learning-d9305e228476

The End

or Just the Beginning…

- The MEng in MIE with Emphasis in Analytics builds on the foundations covered in APS1070.
- Courses to consider:
  - MIE1624 – Introduction to Data Science and Analytics
  - ECE1513 – Introduction to Machine Learning
  - MIE1628 – Big Data Science
  - MIE1517 – Introduction to Deep Learning
  - APS1080 – Introduction to Reinforcement Learning
  - and many more…

Short Break

Review

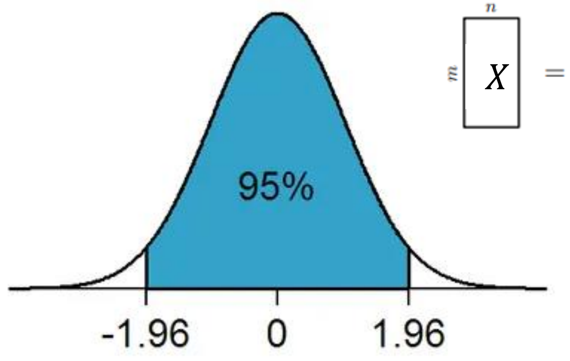
Programming Checklist

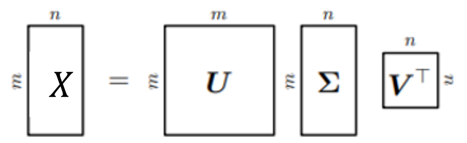
- ❑ Python Basics
  - ❑ int, float, bool, operations, display,
  - ❑ lists, tuples, dictionaries, sets and mutability
  - ❑ conditionals, loops, functions, list comprehension, files
  - ❑ object-oriented programming, class, methods, polymorphism
- ❑ Algorithms and Big O Notation
  - ❑ array search, sorting, hashing
- ❑ Common Data Science Packages
  - ❑ Jupyter notebook
  - ❑ pandas, numpy, scipy, matplotlib, scikit-learn
- ❑ Automatic Differentiation
- ❑ Github Basics

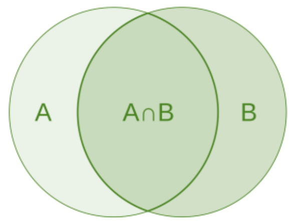
Mathematics Checklist

- ❑ Discrete Math
  - ❑ sets, union and intersection, combinations, permutations
- ❑ Probability Theory
  - ❑ summary statistics, expectation, covariance,
  - ❑ multivariate gaussians and basic distributions
  - ❑ Monte Carlo sampling methods, importance sampling
  - ❑ confidence intervals, hypothesis testing, ANOVA
- ❑ Linear Algebra
  - ❑ change of variables, projections, data augmentation
  - ❑ determinant, eigenvalue decomposition, trace, rank
  - ❑ matrix inverse, PCA, SVD and dimensionality reduction
- ❑ Vector Calculus
  - ❑ Jacobians/gradients, matrix derivatives

$$B = \begin{bmatrix} -7 & -7 & 6 \\ 2 & 1 & -1 \\ 4 & 5 & -4 \end{bmatrix}$$



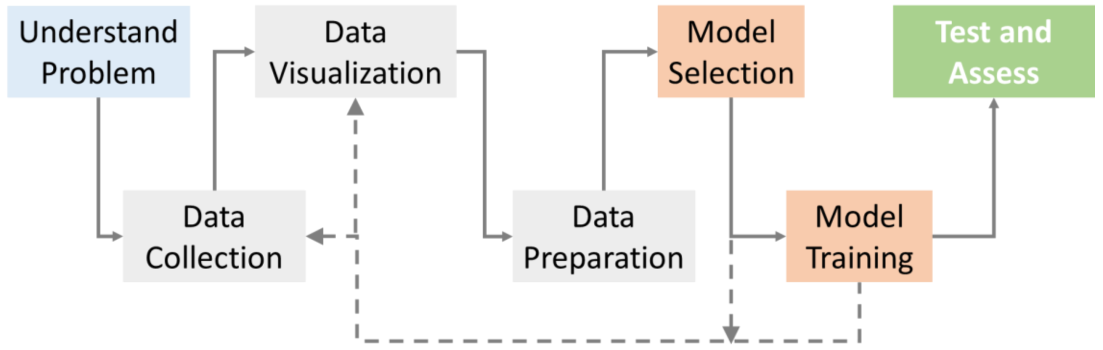




$$\underset{\uparrow\ \text{CDF}}{F_X(x)} = \int_{-\infty}^{x} \underset{\uparrow\ \text{PDF}}{f_X(t)}\,dt.$$

Machine Learning and Analytics Checklist

- ❑ Instance Based Learning
  - ❑ KNN, decision trees, clustering strategies
- ❑ Model Based Learning
  - ❑ linear/logistic regression, MLE approach
  - ❑ feature mapping and polynomial regression
  - ❑ mixture of Gaussians for anomaly detection
  - ❑ deep learning overview and PyTorch
- ❑ Machine Learning Taxonomy
  - ❑ parametric/nonparametric,
  - ❑ classification/ regression,
  - ❑ supervised/unsupervised,
  - ❑ model/instance –based learning
- ❑ Discrete and Continuous Optimization
  - ❑ gradient descent, regularization and convexity
  - ❑ greedy algorithms and set cover
- ❑ Performance Metrics
  - ❑ generalization, overfitting, bias/variance
  - ❑ precision and recall, Confusion Matrix, ROC
  - ❑ nested cross-validation



Final Assessment Details

Final Assessment Details

- **When**
  - Start Time: Tuesday, Apr 12 at 9am (Toronto Time)
  - End Time: Wednesday, Apr 13 at 3pm (Toronto Time)
- **Where**
  - Crowdmark – invites will be sent by email.
  - Submit answers page by page.
- **Duration**
  - Timed exam
  - 2.5 hours to required complete (**3.5 hours available**)

Final Assessment Details

- **Late submissions will receive a grade of 0.** Please ensure you have enough time to upload your work to Crowdmark.
- Sign Honour Code
  - **All work must be your own; all work must be completed without consulting with anyone else.**
  - Students suspected of plagiarism on a project, midterm or final assessment will be referred to the department for formal discipline for breaches of the Student Code of Conduct.
- No access to Piazza – email instructor instead
  - Announcements will be made for any typos or clarifications

Final Assessment Topics

- **Primary Topics:**
  - Matrix Decompositions, SVD, PCA, Projections
  - Vector Calculus, Matrix Derivatives, Identities
  - Linear Regression, Maximum Likelihood Estimation
  - Logistic Regression, Binary and Multiclass Classification
  - Gradient Descent, Regularization, Vectorization and Convexity
  - Neural Network Architectures and Automatic Differentiation
  - Discrete Optimization and Monte Carlo Sampling Methods
  - **Also included are concepts covered before the midterm.**
- **Secondary Topics:**
  - Concepts from Tutorials 0 - 4 and Projects 1 – 4

- **topics before midterm**

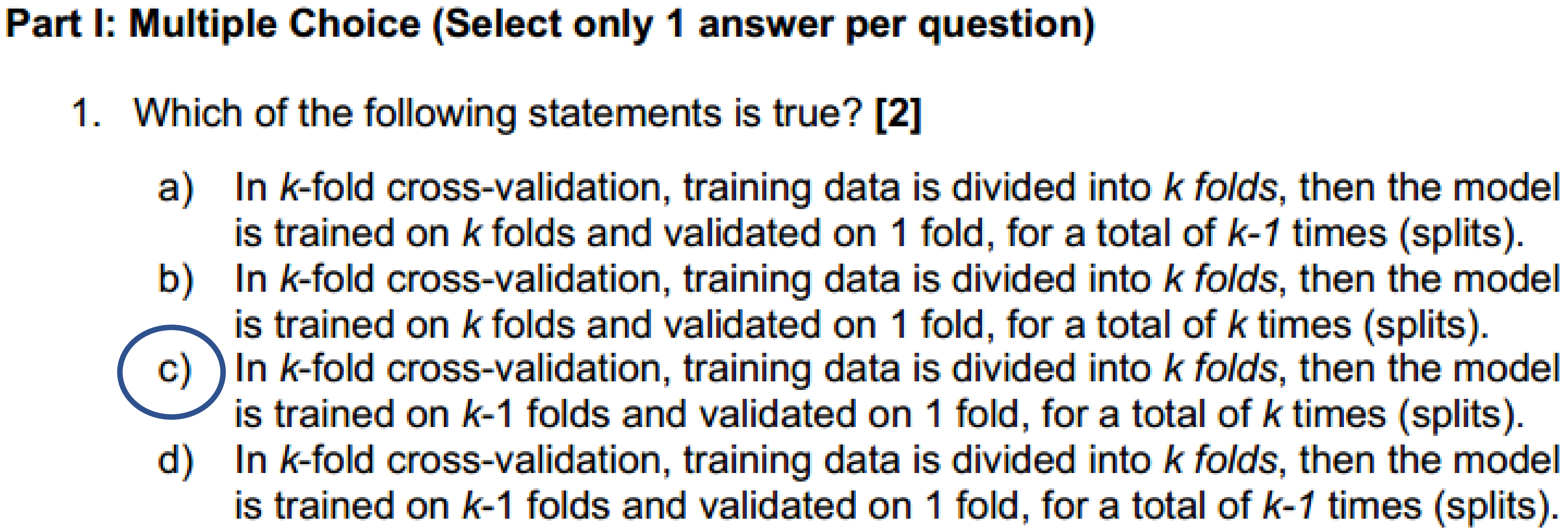
- Overview of Machine Learning, Generalization, Cross-validation, etc.
- Big-O Notation and General Understanding of Python and Python Libraries
- k-NNs, Decisions Trees, and Clustering Techniques
- Probability Theory, Expectation, Covariance, Gaussians
- Precision and Recall, Confusion Matrix, ROCs
- Basic Linear Algebra, Analytic Geometry and Transformations

Similar to the Midterm Format

- Multiple Choice
- True and False
- Short Answer
- Coding
  - You **will not be expected to write entire code from scratch**, but you could be asked to provide pseudo(or partial) code, **modify or interpret sample code** similar to what was asked in the midterm.
  - Not graded on Python syntax, order of parameters
  - You **will not be required to use Google Colab**, but it is available to check you work (does not count as showing work).

Sample Questions

Fall 2019 – Part I



Part I: Multiple Choice (Select only 1 answer per question)

1. Which of the following statements is true? **[2]**

   a) In *k*-fold cross-validation, training data is divided into *k folds*, then the model is trained on *k* folds and validated on 1 fold, for a total of *k-1* times (splits).
   b) In *k*-fold cross-validation, training data is divided into *k folds*, then the model is trained on *k* folds and validated on 1 fold, for a total of *k* times (splits).
   [✓ c)] In *k*-fold cross-validation, training data is divided into *k folds*, then the model is trained on *k*-1 folds and validated on 1 fold, for a total of *k* times (splits).
   d) In *k*-fold cross-validation, training data is divided into *k folds*, then the model is trained on *k*-1 folds and validated on 1 fold, for a total of *k-1* times (splits).

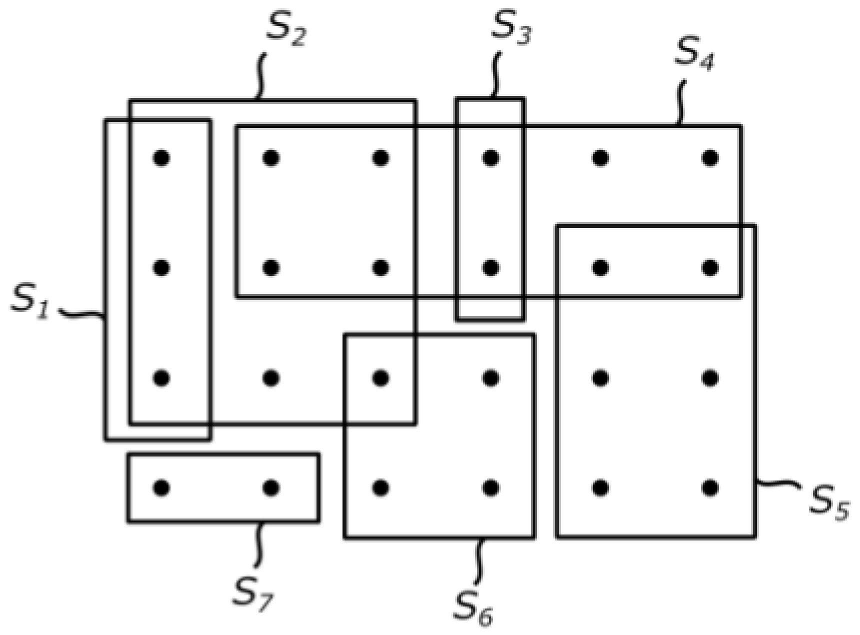
Fall 2019 – Part I

2. When performing PCA, the goal is to accomplish which of the following? **[2]**

   a) Maximize the variance of the primary components and maximize the residuals.
   b) Minimize the variance of the primary components and minimize the residuals.
   [✓ c)] Maximize the variance of the primary components and minimize the residuals.
   d) Minimize the variance of the primary components and maximize the residuals.

Fall 2019 – Part II

3. The schematic to the right has sets S_1, S_2, S_3, S_4, S_5, S_6 and S_7. What sets, and in what order, would a greedy algorithm select to cover the universe (i.e., cover each point) if all sets are weighted equally? **[2]**



Answer

In order of selection: S4, S2, S5, S6, S7

We exclude S1, S3

Fall 2019 – Part II

4. Is the matrix $A = \begin{pmatrix} 1 & 2 & -1 \\ 1 & 0 & 3 \\ -2 & -4 & 2 \end{pmatrix}$ invertible? **[2]**

Answer

For $n = 3$ (known as Sarrus' rule),

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{21}a_{32}a_{13} + a_{31}a_{12}a_{23}$$
$$- a_{31}a_{22}a_{13} - a_{11}a_{32}a_{23} - a_{21}a_{12}a_{33} .$$

=1*0*2+1*(-4)*(-1)+(-2)*2*3-(-2)*0*(-1)-1*(-4)*3-1*2*2=0+4-12+0+12-4=0

No -> determinant is 0

Fall 2019 – Part II

5. Given that $\binom{n}{r} = \frac{n!}{r!(n-r)!}$, prove that $\binom{n}{r} = \binom{n}{n-r}$. **[2]**

Answer

$$= \frac{n!}{(\boldsymbol{n-r})!\,(n-(\boldsymbol{n-r}))!}$$

$$= \frac{n!}{(n-r)!\,r!}$$

Fall 2019 – Part II

6. We have $x = [1,1,2]^T \in \mathbb{R}^3$ and $y = [1,0,3]^T \in \mathbb{R}^3$. What is the angle between vectors? **[2]**

Answer

$$\cos(w) = \frac{x^T y}{\sqrt{x^T x y^T y}} = \frac{7}{\sqrt{6*10}} \rightarrow w = 25.4\ degrees$$

Fall 2019 – Part II

7. Calculate the Jacobian $\boldsymbol{J_F}(x_1, x_2, x_3)$ of the function $\boldsymbol{F} : \mathbb{R}^3 \to \mathbb{R}^4$, which has components: **[2]**

$$y_1 = x_1 + 2x_3^2 \ ; \ y_2 = x_1 \sin x_2 \ ; \ y_3 = x_2 \exp(x_3) \ ; \ y_4 = x_1 + x_2$$

Answer

$$\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} \\ \frac{\partial y_4}{\partial x_1} & \frac{\partial y_4}{\partial x_2} & \frac{\partial y_4}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 4x_3 \\ \sin(x_2) & x_1\cos(x_2) & 0 \\ 0 & \exp(x_3) & x_2\exp(x_3) \\ 1 & 1 & 0 \end{bmatrix}$$

Fall 2019 – Part II

8. Consider functions $f : \mathbb{R}^3 \rightarrow \mathbb{R}^1$ and $\boldsymbol{x} : \mathbb{R}^1 \rightarrow \mathbb{R}^3$, where:

$$f(\boldsymbol{x}) = x_1 + 2x_2^2 + x_3 \ ; \ \boldsymbol{x}(t) = \begin{bmatrix} t \\ 3t \\ \exp(t) \end{bmatrix}$$

Calculate the gradient $\frac{df}{dt}$ using the chain rule. **[2]**

Answer

$$\begin{bmatrix} \frac{df}{dx_1} & \frac{df}{dx_2} & \frac{df}{dx_3} \end{bmatrix} \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \frac{dx_3}{dt} \end{bmatrix} = \begin{bmatrix} 1 & 12t & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ \exp(t) \end{bmatrix} = 1 + 36t + \exp(t)$$

Fall 2019 – Part III

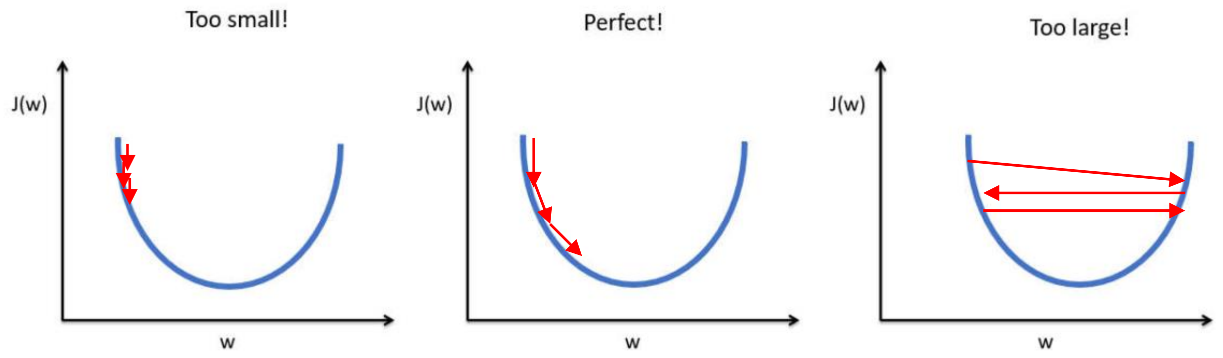
9. In general, is using a gradient descent algorithm a good choice for finding optimal hyperparameters? Why or why not? **[2]**

   Answer

   No. Gradient descent requires a smooth function to optimize, and ideally a convex function in order to find the global minimum. This is usually not the case for hyperparameters.

10. What is a learning rate? Explain what happens if it is set too high or too low (you may use a schematic/drawing if helpful). **[2]**

    Answer



Learning rate is the step size taken in the opposite direction of the gradient.

Fall 2019 – Part IV

11. You have the following code:

```
for i in range (n):
    for j in range ((i+1)**3):
        x = x + 1
        y = y + 1
```

How efficient is this code in terms of big O notation (what is the order of the algorithm's runtime)? [hint: $\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4}$] **[2]**

Answer

The loop is performed 1, 8, 27, … , n^3 times, $\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4} = \frac{n^2(n^2+2n+1)}{4} = \frac{n^4+2n^3+n^2}{4} \rightarrow O(n^4)$

Fall 2019 – Part IV

12. The input parameter *x* of "my_function" is training data with features as columns and examples as rows.

```
def my_function(x):
  return (x - x.mean()) / x.std
```

a) What is the purpose of this function, specifically? **[2]**

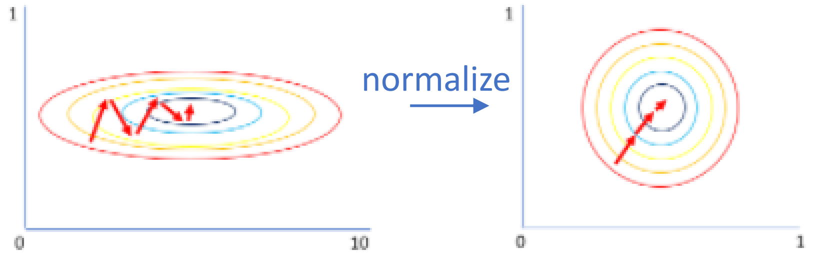
b) Describe why this type of process is important to gradient descent and K-Nearest Neighbor algorithms. **[2]**

Answer

a. Normalize each dimension by zero mean and unit variance.

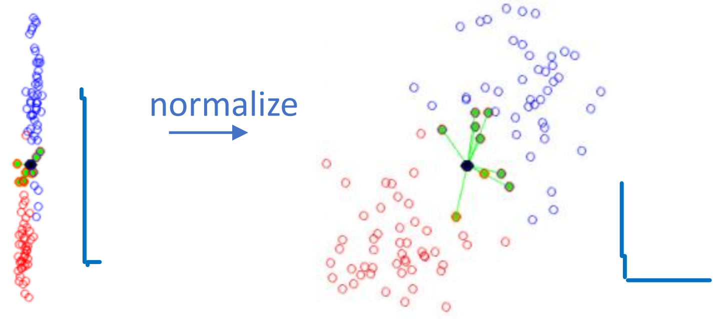
b. Gradient descent

gradient does not point towards minimum, hence takes longer to converge



K-Nearest Neighbours

All nearest neighbours align in direction of the axis with the smaller range.



Fall 2019 – Part IV

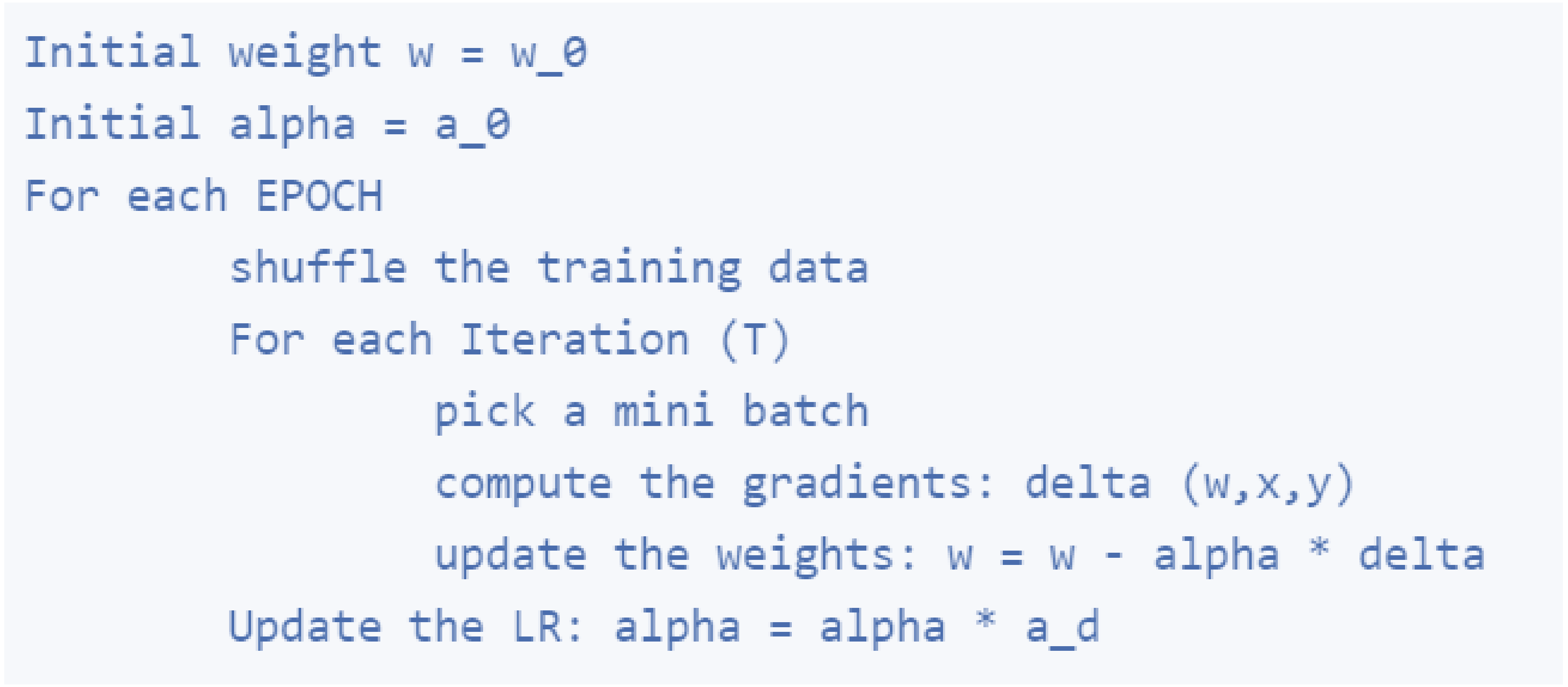
13. Consider a differentiable loss function $l(\boldsymbol{w}, \boldsymbol{x}, y)$ and a dataset D. You're asked to optimize the average loss $\frac{1}{N}\sum_{i=1}^{N} l\left(\boldsymbol{w}, \boldsymbol{x}^{(i)}, y^{(i)}\right)$ with respect to $\boldsymbol{w}$. Write pseudo-code implementing mini-batch gradient descent optimization using a decaying learning rate (i.e., a learning rate that is reduced by α_D every epoch), with the following inputs: **[4]**

- Differentiable loss $l(\boldsymbol{w}, \boldsymbol{x}, y)$ with gradient $\nabla_{\boldsymbol{w}} l(\boldsymbol{w}, \boldsymbol{x}, y)$.
- Dataset $D = \left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \ldots \left(\boldsymbol{x}^{(N)}, y^{(N)}\right)$.
- Mini-batch size m, initial weights $\boldsymbol{w}_0$ and number of steps T.
- Initial learning rate α_0 and learning rate decay α_D.

Note that your pseudo-code is just a representation of your algorithm written in plain English. Keep your pseudo-code simple and concise. Please use indentation and control structures. State any assumptions you make.

Fall 2019 – Part IV

Answer



```
Initial weight w = w_0
Initial alpha = a_0
For each EPOCH
        shuffle the training data
        For each Iteration (T)
                pick a mini batch
                compute the gradients: delta (w,x,y)
                update the weights: w = w - alpha * delta
        Update the LR: alpha = alpha * a_d
```

Summer 2020

2. [2] We have that

$$f(\boldsymbol{x}) = f(x_1, x_2, x_3) = ln(x_1 x_2) - \frac{x_3}{2}$$

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \begin{bmatrix} e^t \\ -cos(t) \\ 2t \end{bmatrix}$$

Calculate the gradient $\frac{df}{dt}$. Show all your calculations.

Answer

Given that $ln(x_1 x_2) = ln(x_1) + ln(x_2)$

$$\frac{df}{dt} = \frac{df}{dx}\frac{dx}{dt} = \begin{bmatrix} \frac{df}{dx_1} & \frac{df}{dx_2} & \frac{df}{dx_3} \end{bmatrix} \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \frac{dx_3}{dt} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{x_1} & \frac{1}{x_2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} e^t \\ sin(t) \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{e^t} & -\frac{1}{cos(t)} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} e^t \\ sin(t) \\ 2 \end{bmatrix} = 1 - \tan(t) - 1$$

$$= -tan(t)$$

Summer 2020

3. [3] Binary classification Model A achieves a top F1 score of $F1_A$ and an area under the ROC curve (AUC) of AUC_A, whereas binary classification Model B achieves a top F1 score of $F1_B$ and an AUC of AUC_B, on the same dataset. If $AUC_A > AUC_B$., is it implied that $F1_A > F1_B$? Why?

Answer

No, $F1_A > F1_B$ is not implied. AUC is the area under the ROC curve, which plots TPR ($TPR = \frac{TP}{TP+FN}$) vs FPR ($FPR = \frac{FP}{FP+TN}$). F1 score is ($\frac{2TP}{2TP+FP+FN}$) calculated at a single threshold. ***F1 score is sensitive to changes at the "best" threshold, whereas AUC is impacted by changes anywhere on the ROC curve.***

Example. Let's assume two identical models, and thus $AUC_A = AUC_B$ and the top $F1_A$ = top $F1_B$ at the same threshold *t*. Let's now modify Model B by altering a single prediction at a threshold that's not *t*, changing a TP at that threshold to a FN. Doing this will reduce the F1 score at that threshold, but not alter the best F1 score for that model (a worst F1 score at some other threshold has no impact on the best F1 score!). The ROC curve, however, will change for Model B (it now has 1 more bad prediction), resulting in reduced AUC. In this example, $AUC_A > AUC_B$ and (max) $F1_A$ = (max) $F1_B$.

Summer 2020

4. [3] *StandardScaler* (in *sklearn*) is often used in linear regression models.
   a. What does *StandardScaler* do?
   b. Is it best practice to apply StandardScaler before or after splitting data into training and testing groups? Why?
   c. Typically, is it useful to apply StandardScaler on targets, feature data, or both?

Answer

a. StandardScaler scales features to 0 mean and unit variance.

b. It's best to apply StandardScaler after splitting data, since you want to keep test data completely separate. If you scale before splitting, you're incorporating knowledge of test data properties into your model.

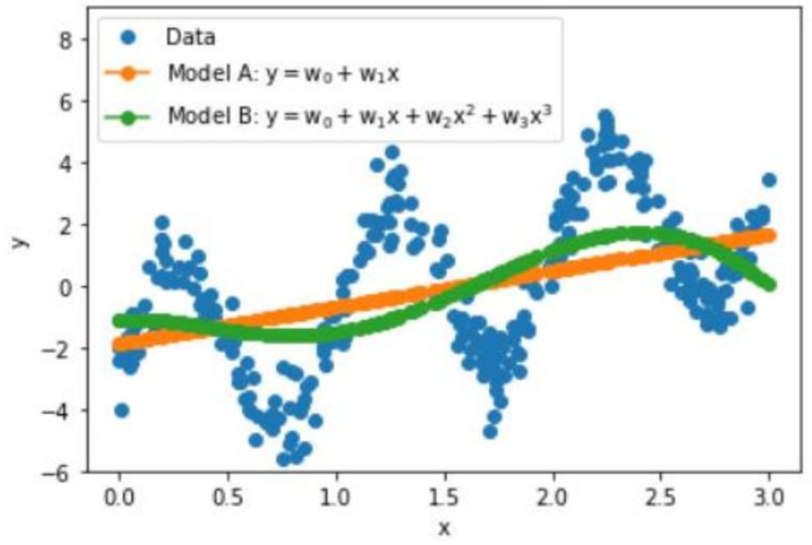
c. StandardScaler is typically applied only on features (so any one feature does not dominate the others). There's usually no benefit to applying it on targets.

Summer 2020

5. [3] Sam applies linear regression to data that has a single feature, *x*. A first model, Model A, makes predictions *y* using only a bias (w_0) and a weighted feature (w_1x), i.e. $y = w_0 + w_1x$. Unsatisfied, Sam then creates new features that are the *x* feature squared and cubed, yielding the model $y = w_0 + w_1x + w_2x_2^2 + w_3x_3^3$ (Model B), which still isn't a great fit.

   a. What minimum degree polynomial is needed to fit the data reasonably well? Why? [hint: count turning points (extremums) of polynomial]

   b. Can you conceive of an alternate feature mapping scheme, where a single additional feature is added to Model A, that would fit the data well? Explain your reasoning.



Answer

a. You'll need a polynomial of at least degree 7 or 9 to fit the data well – there's an explanation here – 1 more degree than the number of bumps.

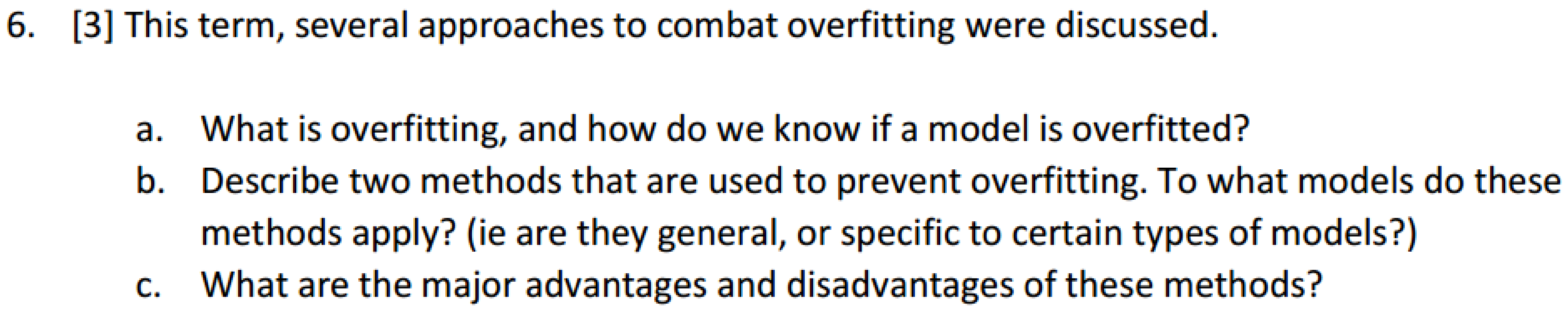
b. The idea here is to add a sinusoidal feature to Model A. The general equation for a sine function is:

$$y = a * sin(b * \underline{x})$$

Here the period is clearly 1, so $1 = \frac{2\pi}{b}$ and $b = 2\pi$

We could thus add a feature sin(2πx), and expect a good fit.

Summer 2020



6. [3] This term, several approaches to combat overfitting were discussed.

   a. What is overfitting, and how do we know if a model is overfitted?
   b. Describe two methods that are used to prevent overfitting. To what models do these methods apply? (ie are they general, or specific to certain types of models?)
   c. What are the major advantages and disadvantages of these methods?

Answer

a. Model has fit the subtleties in the training data that does not generalize to the test data.

b. Cross-validation and regularization are the most obvious methods here, but you could also discuss the removal of features, adding more data, etc. These generalize to all models.

c. Cross-validation: allows you to use your data more effectively, but at the price of increased computation
Regularization: limits the flexibility of the parameters which can be good and bad. Good in that it will limit the model's ability to overfit, but bad in that sometimes it may be too restrictive.