Curse of Dimensionality

Numerical Integration (Quadrature)

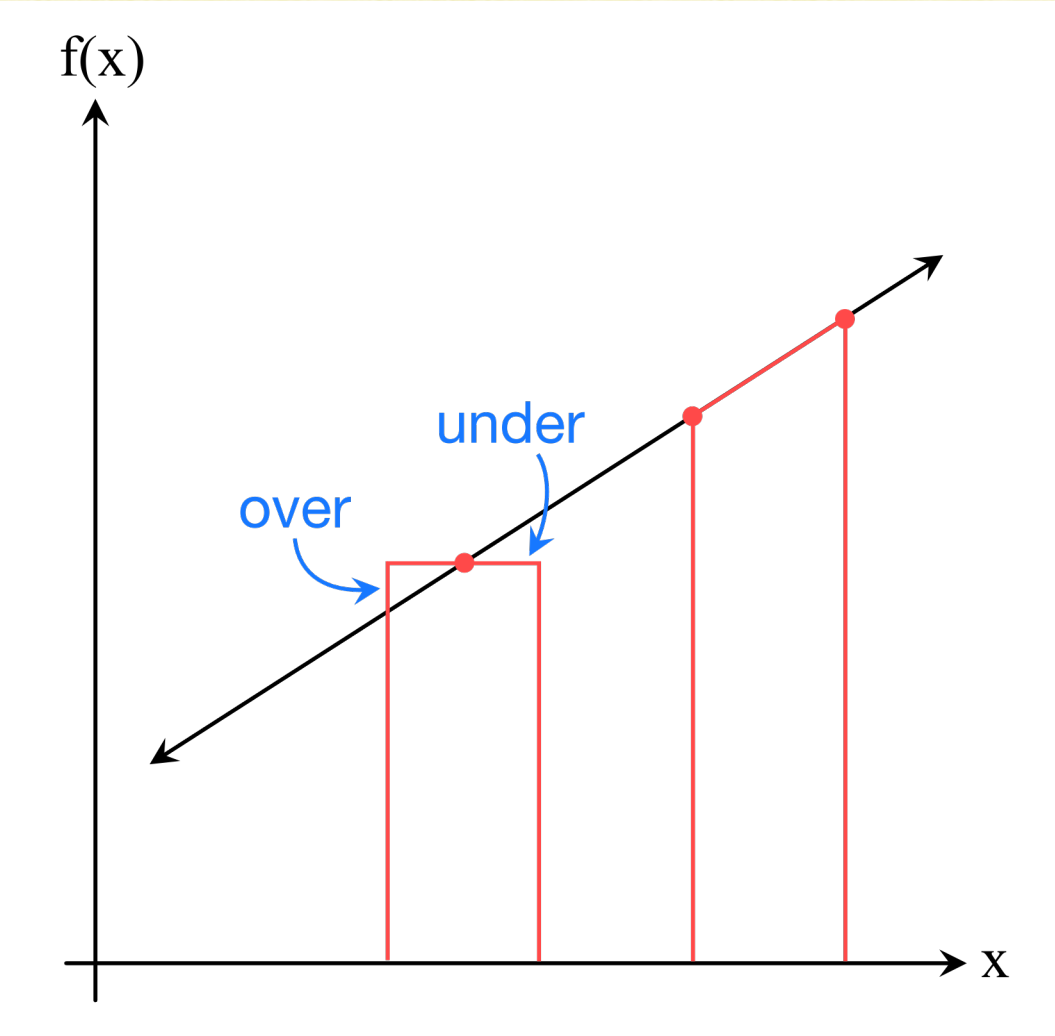
- Approximate $\int_{x_L}^{x_R} f(x)dx$ numerically
- Break up $[x_L, x_R]$ into subintervals, and consider each subinterval separately
- On each subinterval:
  - Reconstruct the function
  - Analytically find the area under the reconstructed curve
- These two steps can be combined in various ways (for efficiency)

- f is often not explicitly known
- I.e., often only have access to output values $f(x_i)$ given input values x_i
- In addition, it could be "expensive" to evaluate $f(x_i)$, especially when it requires running code

Newton-Cotes Quadrature

- On each subinterval, choose p equally spaced points and use $p-1$ degree polynomial interpolation to reconstruct the function and approximate the area under the curve
- Obtains the exact solution when f is a degree $p-1$ polynomial (as expected)
- When the number of points p is odd, symmetric cancellation gives the exact solution on a degree p polynomial (1 degree higher than expected)

Symmetric Cancellation



- When p = 2 points, the 1st degree piecewise linear approximation integrates piecewise linear functions exactly
- When p = 1 point, the 0th degree piecewise constant approximation (also) integrates piecewise linear functions exactly
  - Note the cancellation of under/over approximations in the figure

Newton-Cotes Quadrature

- Consider a total of m subintervals
- Piecewise constant approximation ($p = 1$ point) uses m points to integrate piecewise linear functions exactly
- Piecewise linear approximation ($p = 2$ points) uses $m + 1$ points to integrate piecewise linear functions exactly
  - points on the boundary between intervals are used for both intervals
- Piecewise quadratic approximation ($p = 3$ points) uses $2m + 1$ points to integrate piecewise cubic functions exactly
- Piecewise cubic approximation ($p = 4$ points) uses $3m + 1$ points to integrate piecewise cubic functions exactly

Local and Global Error

- Degree p polynomial reconstruction captures the Taylor expansion terms up to and including $\frac{h^p}{p!} f^{(p)}(x)$, with $O(h^{p+1})$ errors
- This $O(h^{p+1})$ error in the height of the function multiplied times the $O(h)$ width of the interval gives a per interval area error (local error) of $O(h^{p+2})$
- The total number of intervals is $\frac{x_R - x_L}{O(h)} = O\left(\frac{1}{h}\right)$, so the total error (global error) is $O\left(\frac{1}{h}\right) O(h^{p+2}) = O(h^{p+1})$
- Doubling the number of intervals halves their size leading to $\left(\frac{1}{2}\right)^{p+1}$ as much error, which is denoted by an order of accuracy of $p + 1$

Newton-Cotes Quadrature (Examples)

- Midpoint Rule: $\sum_i h_i f(x_i^{mid})$
  - 1 point, piecewise constant, exact for piecewise linear, 2nd order accurate, $O(h^2)$ error
- Trapezoidal Rule: $\sum_i h_i \frac{f\left(x_i^{left}\right)+f\left(x_i^{right}\right)}{2}$
  - 2 points, piecewise linear, exact for piecewise linear, 2nd order accurate, $O(h^2)$ error
- Simpson's Rule: $\sum_i h_i \frac{f\left(x_i^{left}\right)+4f\left(x_i^{mid}\right)+f\left(x_i^{right}\right)}{6}$
  - 3 points, piecewise quadratic, exact for piecewise cubic, 4th order accurate, $O(h^4)$ error

Gaussian Quadrature

- Use p optimally chosen points to obtain a method that is exact on degree $2p-1$ polynomials, and thus has an order of accuracy of $2p$
- For example: $\sum_i h_i \frac{f\left(x_i^{mid}-\frac{h_i}{2\sqrt{3}}\right)+f\left(x_i^{mid}+\frac{h_i}{2\sqrt{3}}\right)}{2}$
- 2 points, piecewise cubic, exact for piecewise cubic, 4th order accurate, $O(h^4)$ error
- Same accuracy as the 3 point Simpson's Rule
  - Simpson has 1 point on shared boundaries, so only $2m+1$ total points are required
  - That is, Gaussian quadrature only saves 1 point in total ($2m$ total points)

Two Dimensions

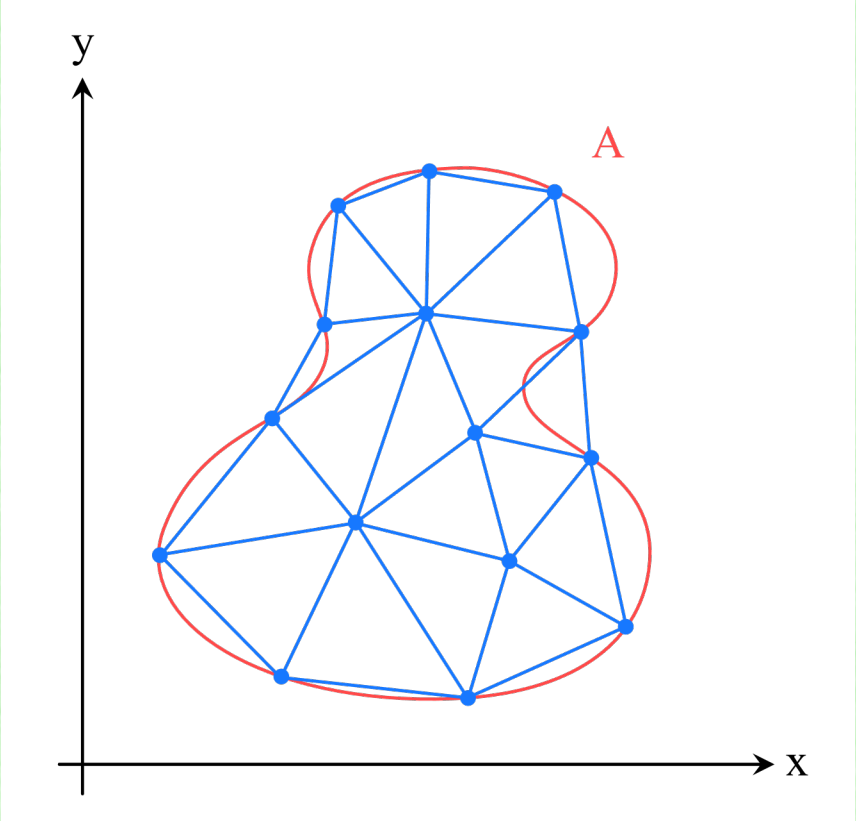
- $\iint_A \ f(x,y)dA$ where sub-regions dA of area A are considered separately

- When A is rectangular, it can be broken into sub-rectangles and addressed dimension-by-dimension using 1D techniques

- When A is more interesting, triangle sub-regions can be used to approximate it
- The difference between A and its approximation leads to a new source of error not seen in 1D (where the interval boundaries were merely points)

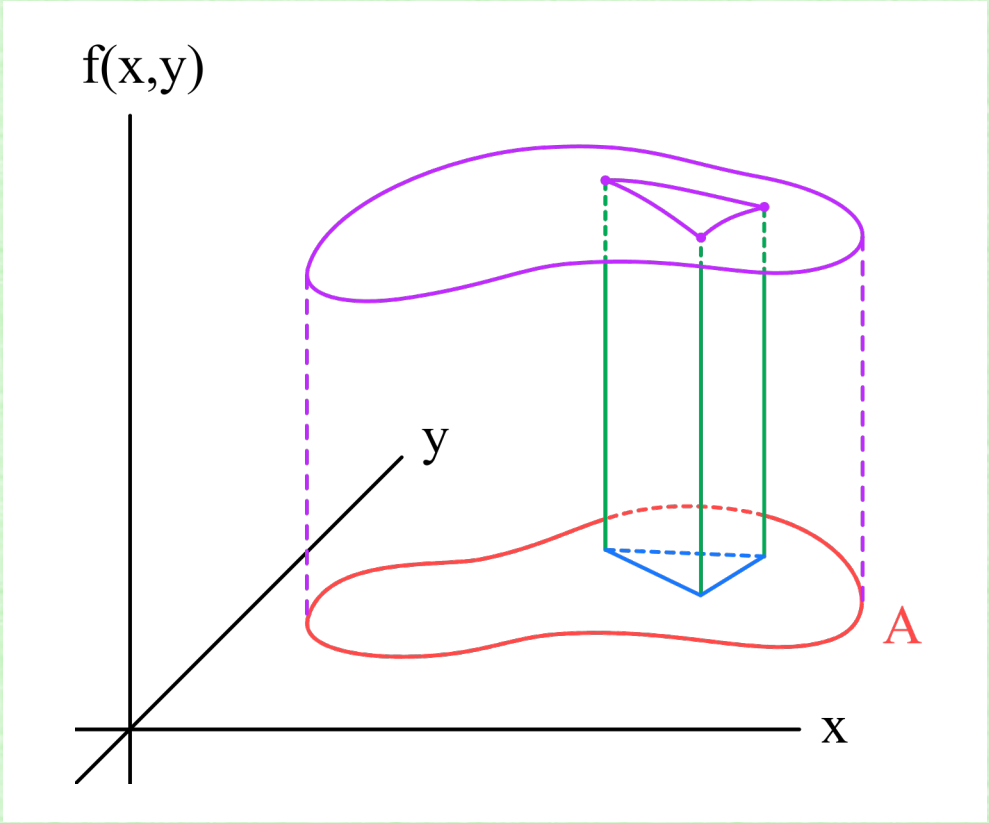
Domain Approximation Errors

- The difference between A and its approximation (via triangles here) leads to a new source of error in the integral (missing/extra area)



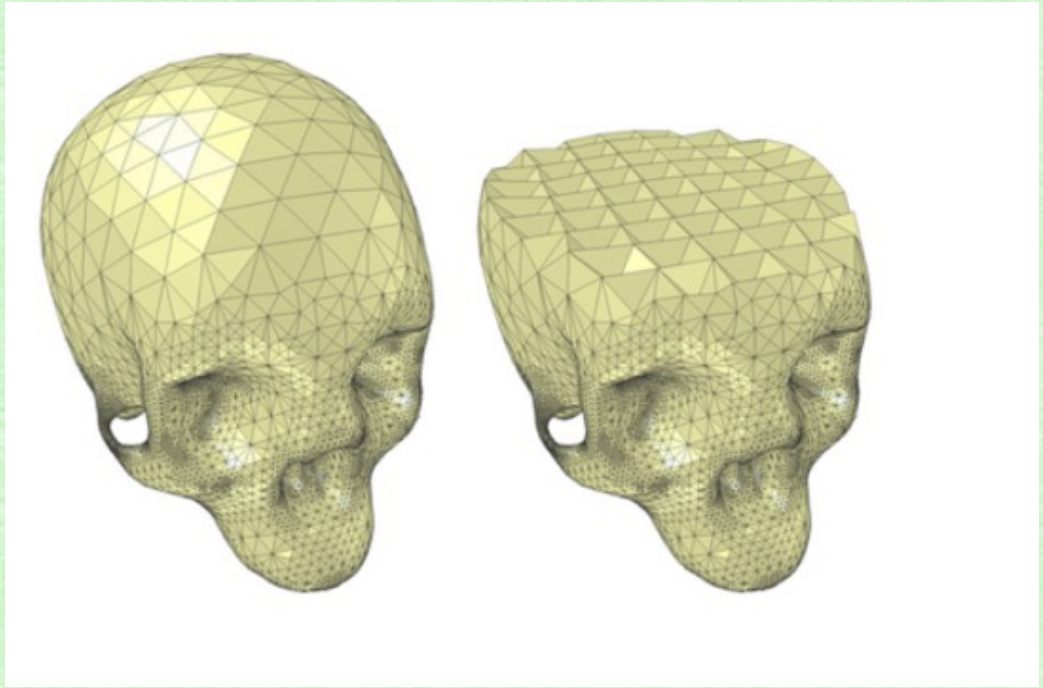
Integrating over Sub-regions

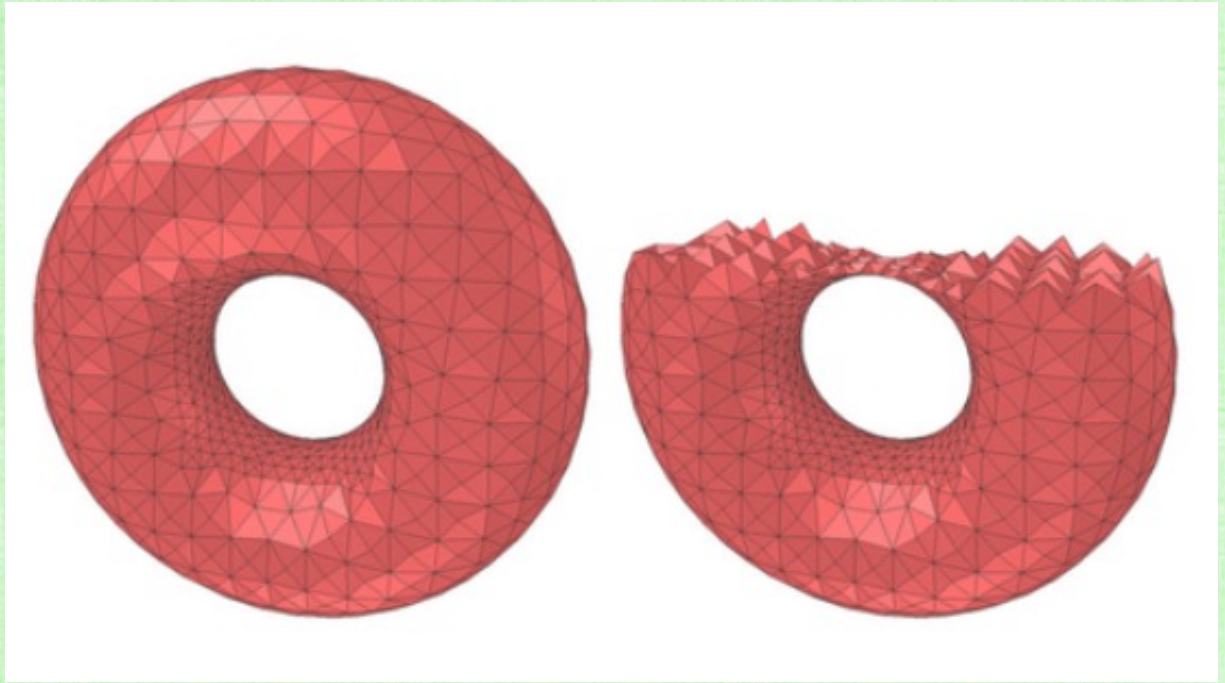
- Each triangle sub-region utilizes optimally chosen Gaussian quadrature points to compute sub-volumes



Three Dimensions

- $\iiint_V \ f(x, y, z)dV$ where tetrahedral sub-regions dV of volume V are each considered separately (with Gaussian quadrature points)





Curse of Dimensionality

- Consider a 1st order accurate method
- 1D: doubling the number of intervals cuts the error in half (2x work = ½ error)
- 2D: halving interval size requires 4 times the rectangles/triangles (4x work = ½ error)
- 3D: halving interval size requires 8 times the cubes/boxes/tets (8x work = ½ error)
- 4D: 16x work = ½ error, 5D: 32x work = ½ error, etc.
- Cutting error by a factor of 4 in 5D takes 32^2=1024x work
- Cutting error by a factor of 8 in 5D takes 32^3=32,768x work
- If the original code took 1 sec to run in 5D, cutting error by a factor of 8 takes 9 hours
- And cutting error by a factor of 16 takes 12 days
- And cutting the error by a factor of 32 takes over a year….

Yep, you're cursed

Curse of Dimensionality

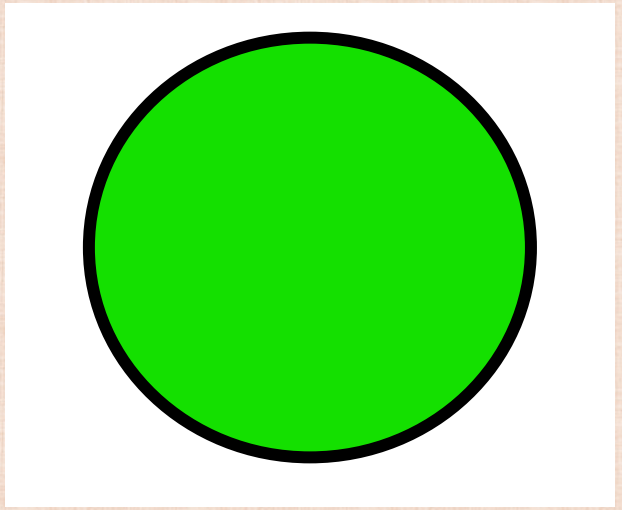
- Consider a 2nd order accurate method
- In 1D/2D/3D/4D/5D/etc. halving the interval size gives 4 times less error
- Cutting error by a factor of 4 in 5D takes 32x work
- If the original code took 1 sec to run in 5D, cutting error by a factor of 16 takes only 17 min (much faster than the 12 days for the 1st order accurate method)
- Cutting error by a factor of 1024 (3 decimal places more accuracy) takes over a year…
- In 10D, cutting error by a factor of 4 takes 1024x work
- Second order is better than first, but still intractable in higher dimensions
- Moreover, it's difficult (or impossible) to construct higher order methods in higher dimensions (and overfitting is a concern too)

Conclusion

- Newton-Cotes style approaches are only practical for 1D/2D/3D
  - or 1D/2D/3D + time
- Sometimes they can work ok in 4D

A Simple Example

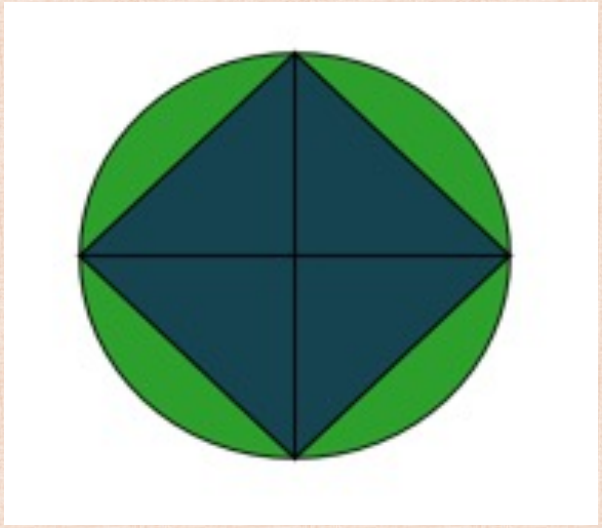
- Consider approximating $\pi = 3.1415926535\ldots$
- Use a compass to construct a circle with radius = 1
- Since $A = \pi r^2$, the area of this unit circle is π
- Integrate $f(x, y) = 1$ over the unit circle to obtain $\iint_A f(x, y)dA = \pi$



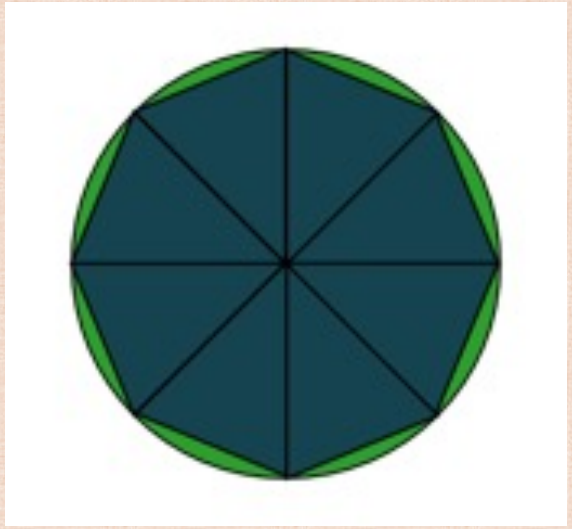
$Area = \pi$

Newton-Cotes Approach

- Inscribe triangles inside the circle
- Sum the area of all the triangles (no need to trivially multiply by the height = 1)
- The difference between the area A and its approximation with triangles leads to errors



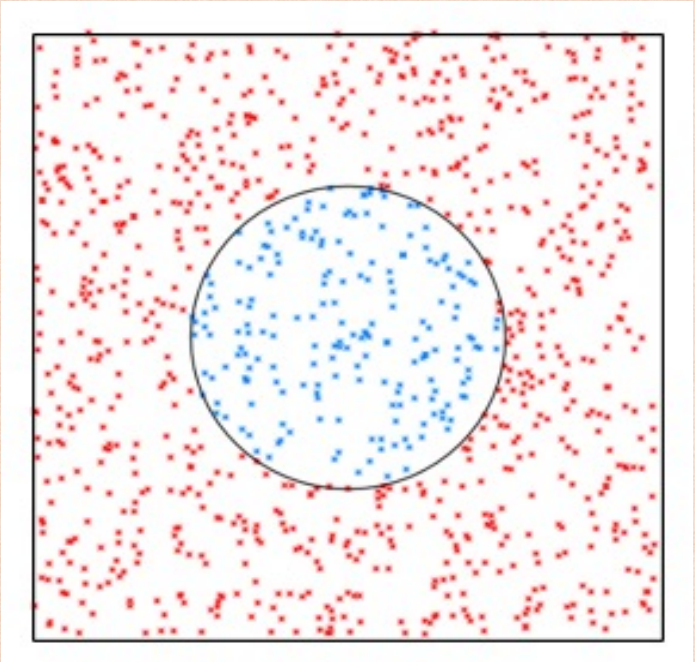
$\pi \approx 2$



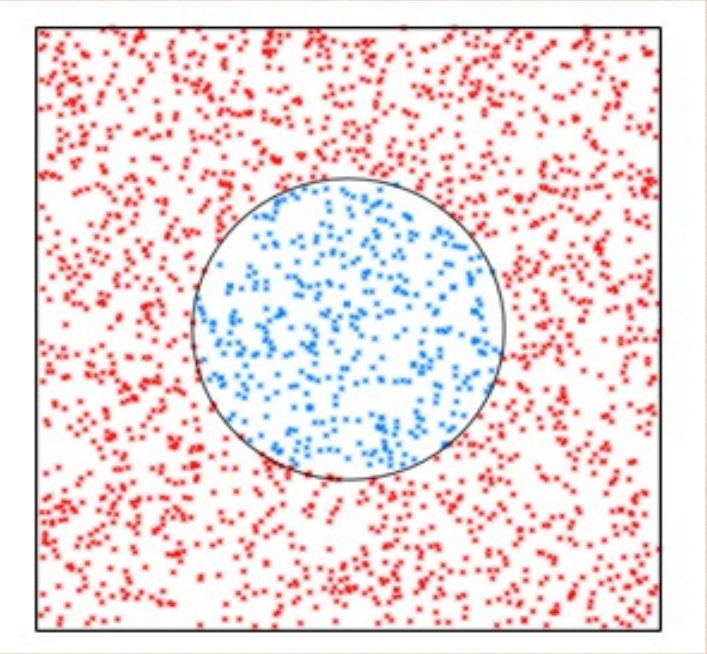
$\pi \approx 2.8284$

Monte Carlo Approach

- Construct a square with side length 4 containing the circle
- Randomly generate N points in the square (color points inside the circle blue)
- Since $\frac{A_{circle}}{A_{box}} = \frac{\pi}{16}$, can approximate $\pi \approx 16\left(\frac{N_{blue}}{N_{blue}+N_{red}}\right)$



$\pi \approx 3.136$



$\pi \approx 3.1440$

Monte Carlo Methods

- Typically used in higher dimensions (5D or more)
- Random (pseudo-random) numbers generate sample points that are multiplied by “element size” (e.g. length, area, volume, etc.)
- Error decreases like $\frac{1}{\sqrt{N}}$ where N is the number of samples (only ½ order accurate)
  - E.g. 100 times more sample points are needed to gain one more digit of accuracy

- Very slow convergence, but independent of the number of dimensions!
- Not competitive for lower dimensional problems (i.e., 1D, 2D, 3D), but the only tractable approach for high dimensional problems

Machine Learning Implications

- Consider $y = f(x)$ where $x \in R^n$ with large n
- Newton-Cotes style approaches would first do polynomial interpolation, and then analytically integrate the result
- An enormous number of points (and control volumes) is required to construct polynomial functions in higher dimensions (curse of dimensionality)
- The same is true when constructing $y = f(x)$ for function interpolation (in order to inference), i.e. a high dimensional x is intractable
- Thus, Monte Carlo approaches are far more efficient!
- This is a major reason for the close collaborations between ML/DL and Statistics departments
  - as compared to classical engineering, which operates in a lower dimensional 3D model of the physical world (and thus has closer ties to Applied Mathematics)