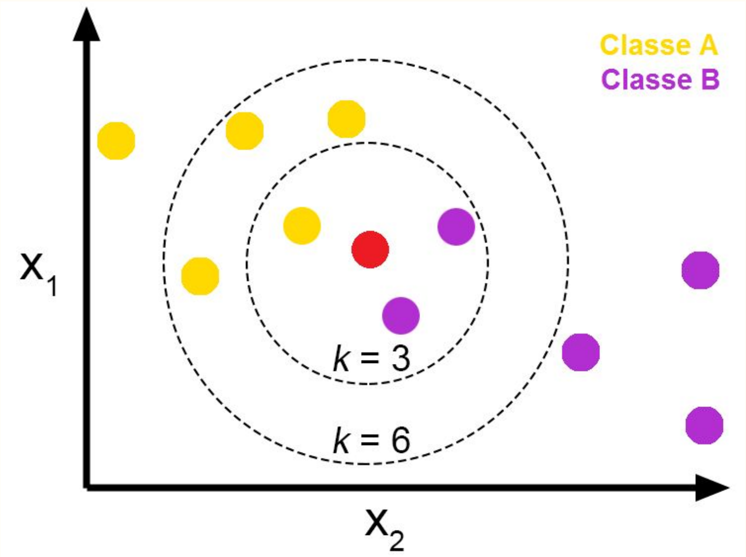
Speeding-up KNN with Dataspace Latticing and Precomputation using Distance Weighting

Alan Zhu and Leonardo Valli

Introduction

- KNN is a simple algorithm good at classifying clusters
- However, it has O(N) classification runtime
- Previous implementations have sped it up to O(log N)
- We want to speed it up to O(1) using a lattice



Related Work

- Proximity Search / Approximate Nearest Neighbors (ANN)
  - Annoy by Spotify
  - Faiss by Meta
  - O(log N) classification
- Product-Quantization KNN (PQKNN)
  - k-Means centroiding
  - O(N) but can be sublinear dependent on hyperparameters
- Distance-weighted KNN

Dataset

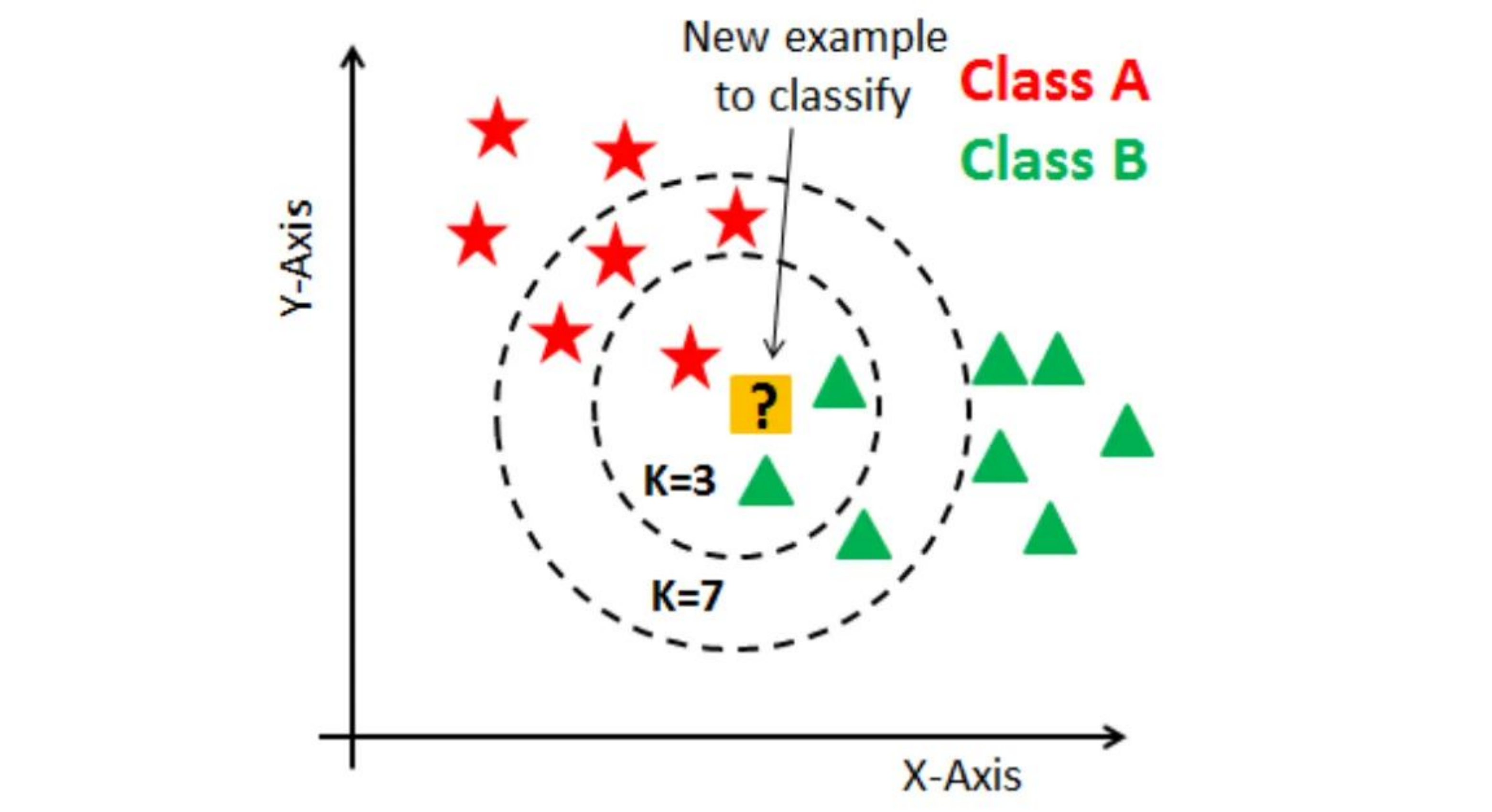
- Transiting Exoplanet Survey Satellite (TESS) Dataset
- 61 attributes - identification, position, planet properties, stellar properties, dates.
- Class - TESS Follow-Up Working Group Disposition (TFOPWG): Confirmed or rejected
- https://exofop.ipac.caltech.edu/tess/

Preprocessing

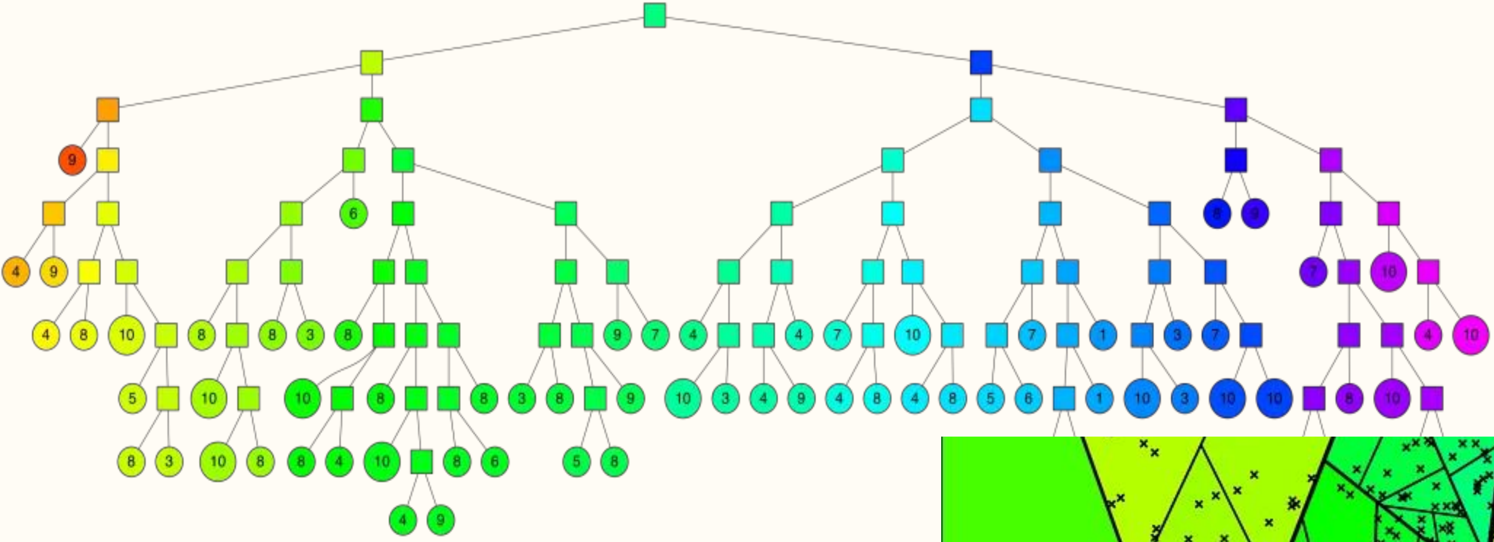
- Made class binary
- Intuitively removed attribute
- Removed 1 mostly empty attribute
- Z-score normalized due to outliers
- CFS subset attribute selection

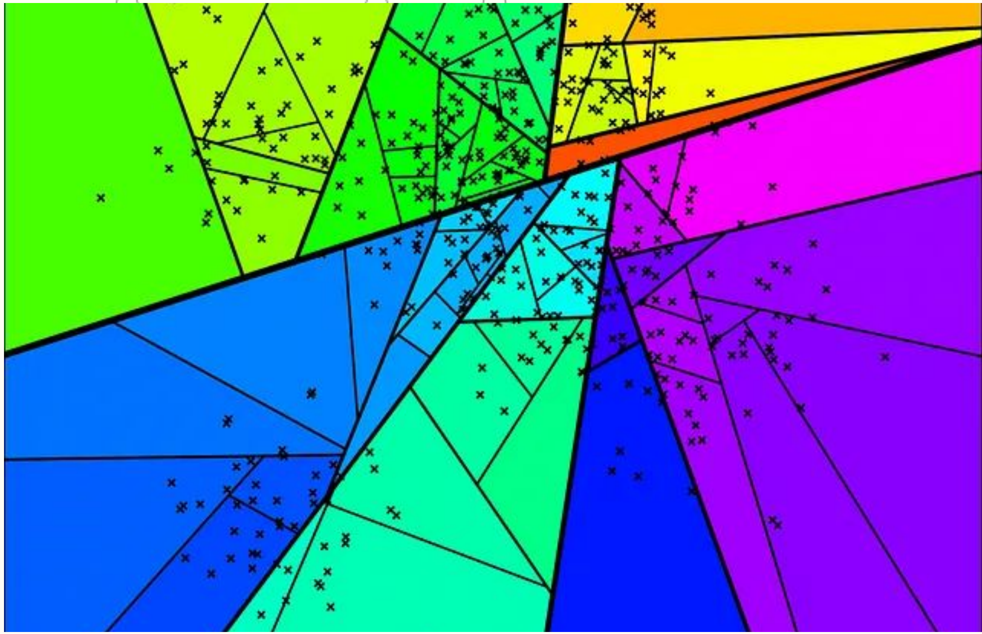
```
Selected attributes: 4,5,6,8,10,12,13,14,15,16,18,20 : 12
                     Time Series Observations
                     Spectroscopy Observations
                     Imaging Observations
                     Period (days)
                     Depth (mmag)
                     Planet Radius (R_Earth)
                     Planet Insolation (Earth Flux)
                     Planet Equil Temp (K)
                     Planet SNR
                     Stellar Distance (pc)
                     Stellar log(g) (cm/s^2)
                     Stellar Mass (M_Sun)
```

KNN



Goal: Achieve O(1) classification with KNN without sacrificing accuracy



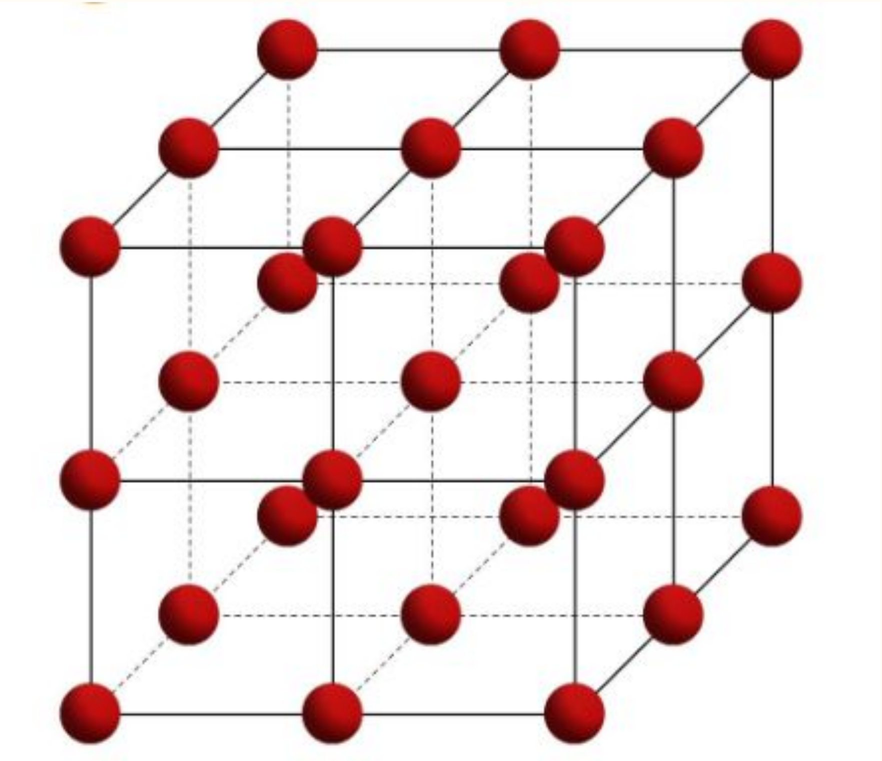


Proximity Search

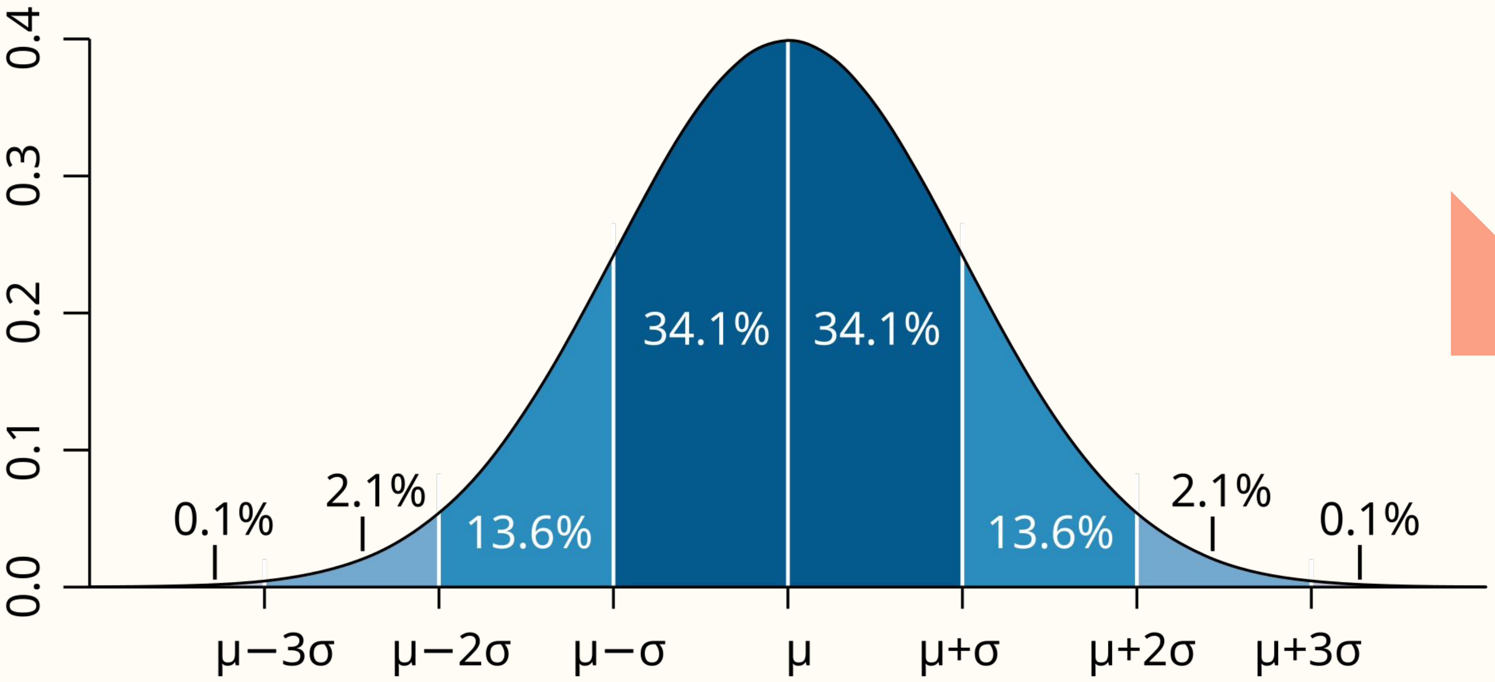
- O(log N)

Dataspace Latticing

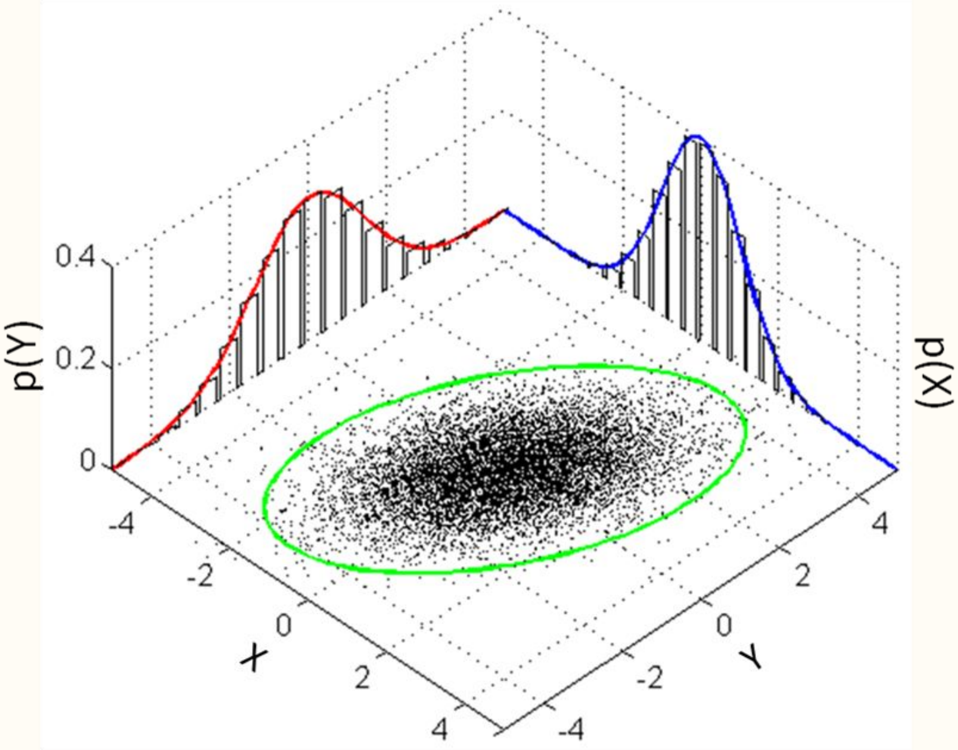
- O(1)

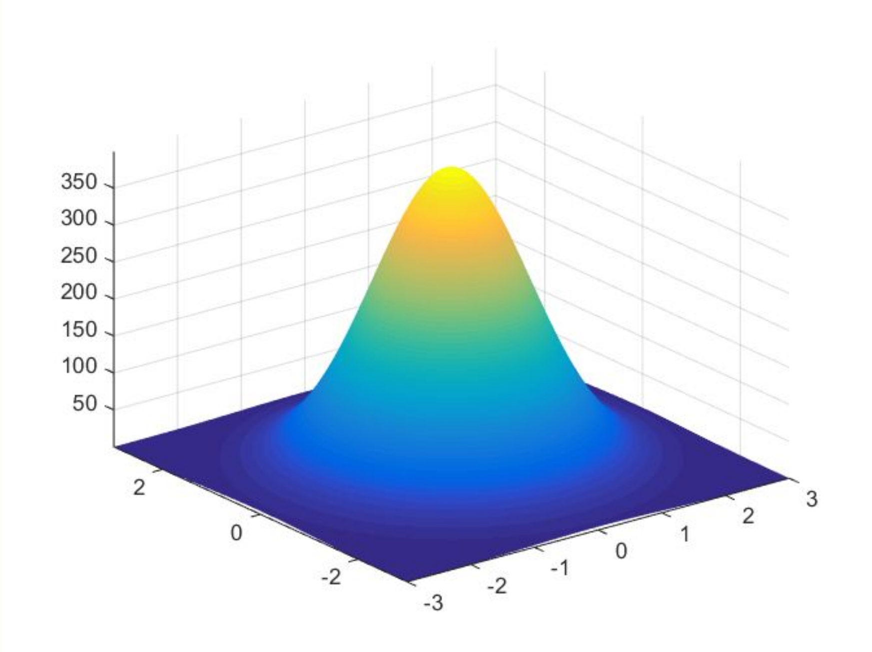


Gaussian Distribution

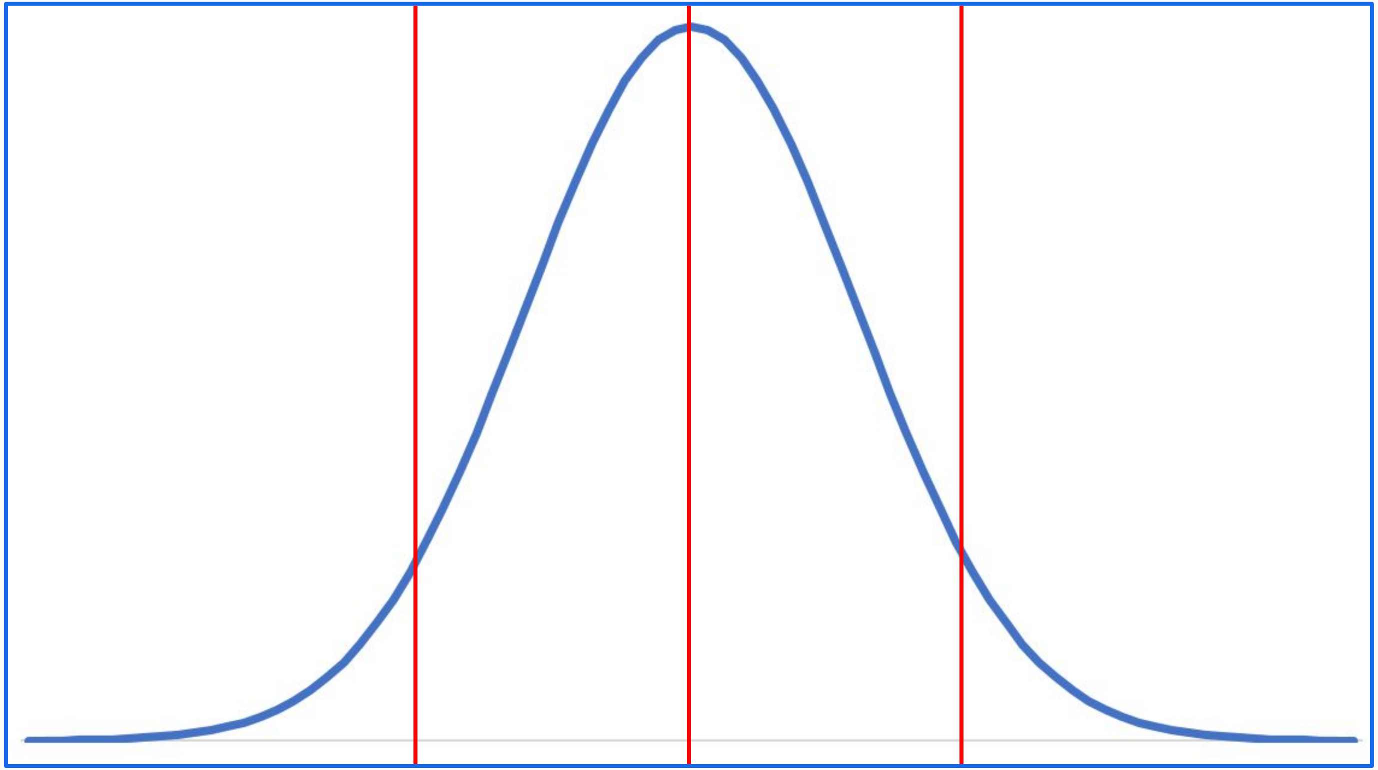


Multivariate Gaussian Distribution

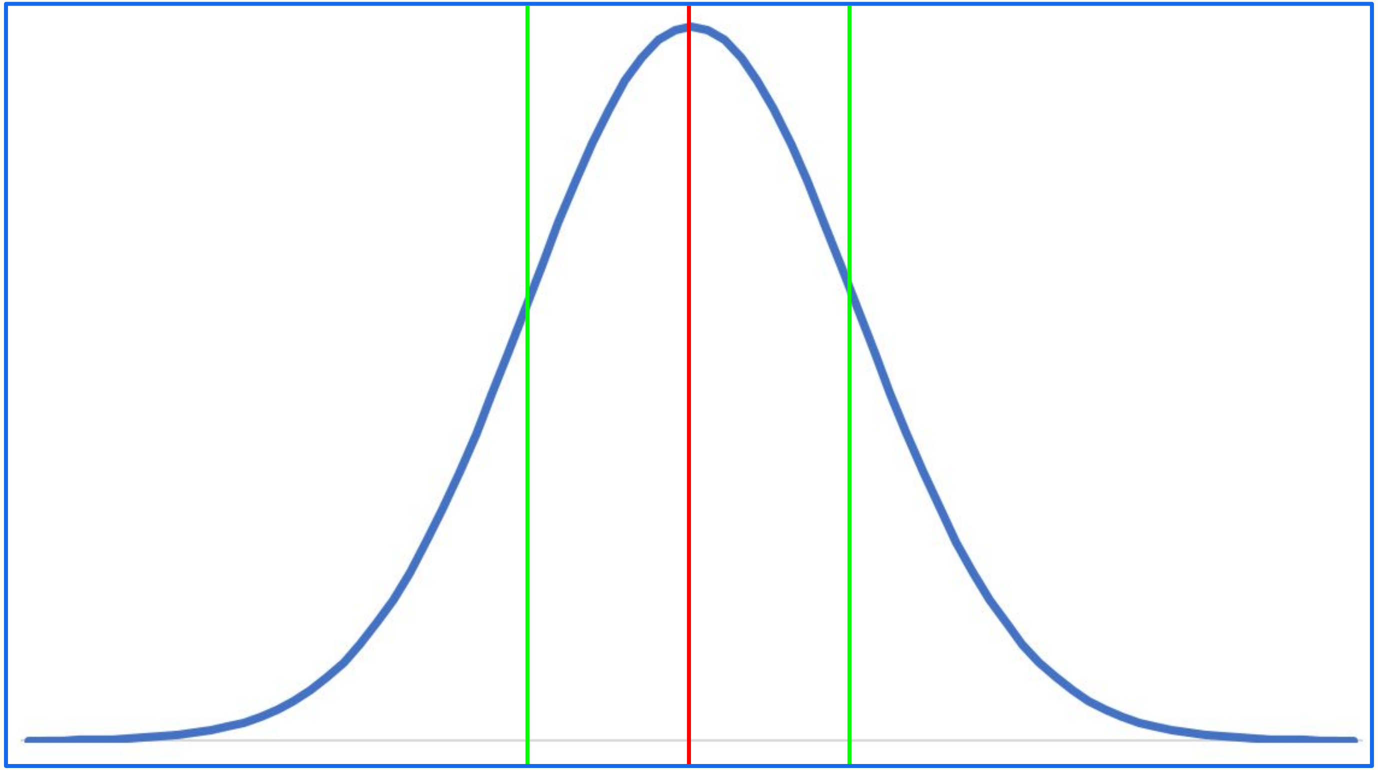




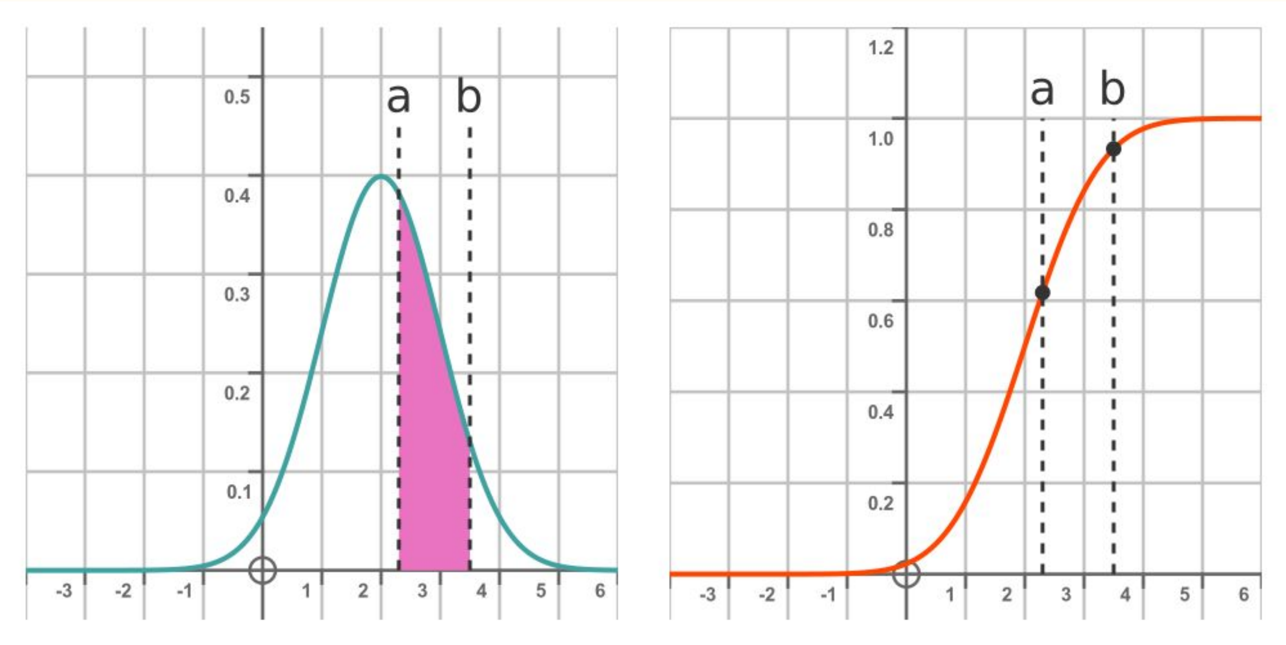
Optimize Lattice Points - Evenly Divide Dataspace



Optimize Lattice Points - Evenly Divide Dataspace

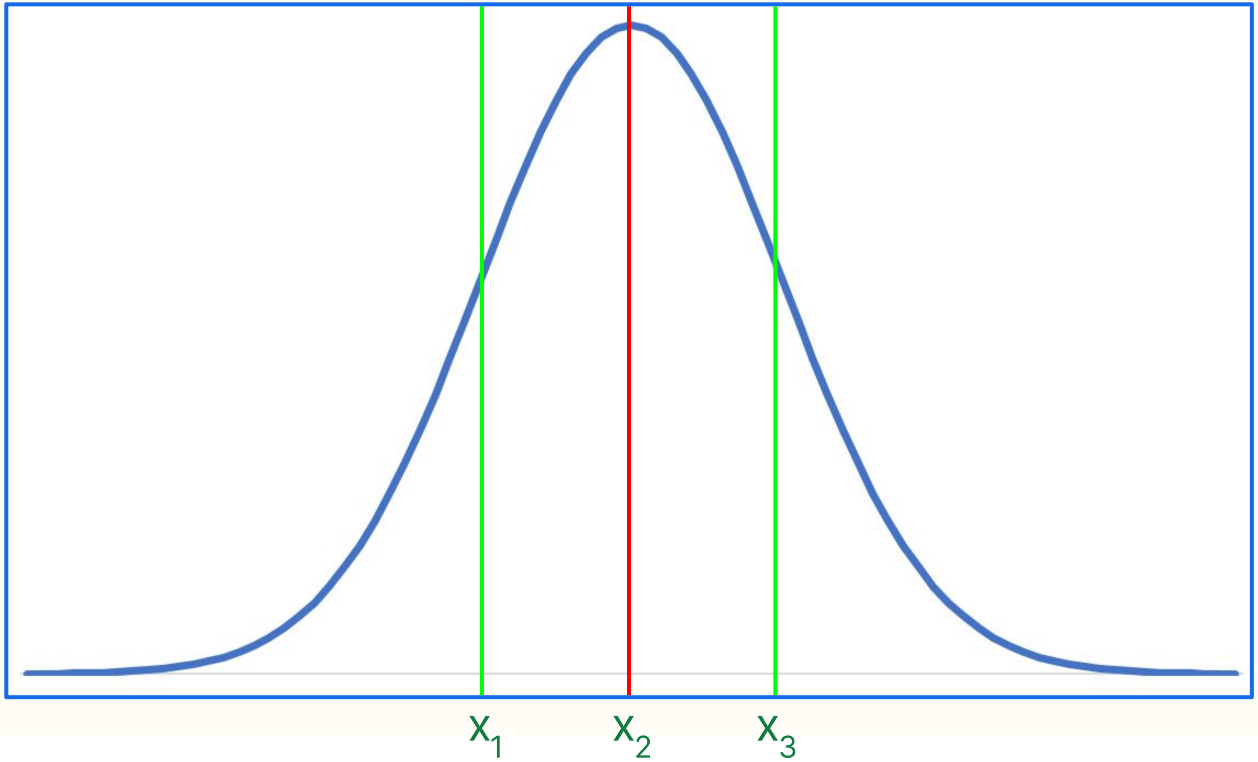


NormalCDF



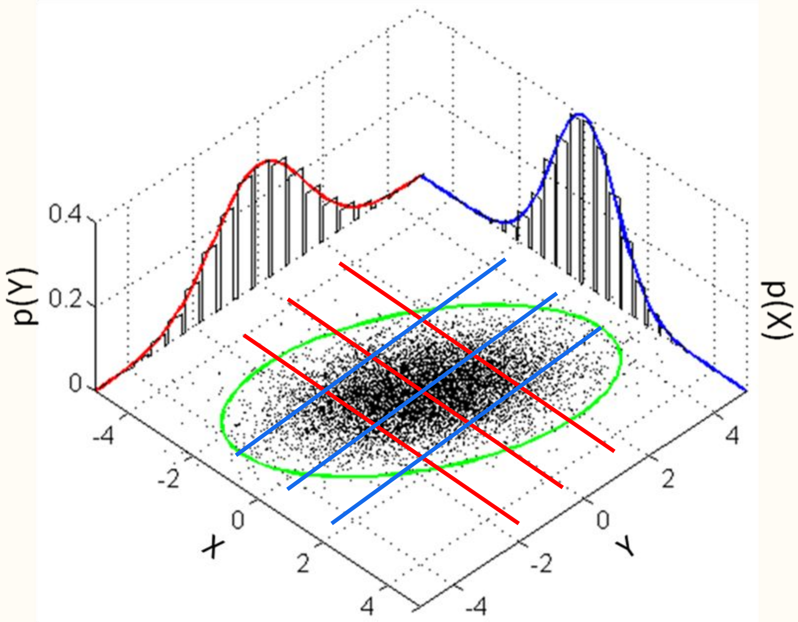
$$normalcdf(a,\ b) = \int_{a}^{b} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\, dx$$

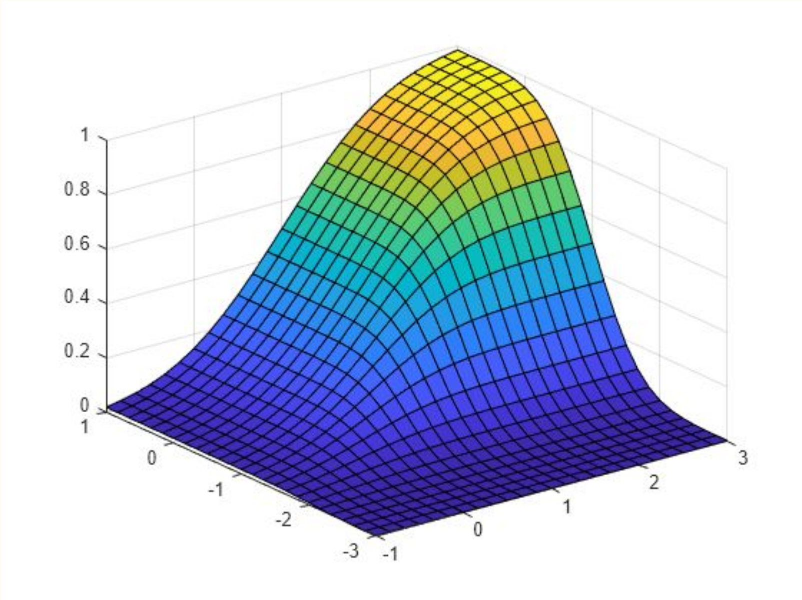
NormalCDF



$$normalcdf(-\infty, x_1) = normalcdf(x_1, x_2) = normalcdf(x_2, x_3) = \ldots\ normalcdf(x_n, \infty) = \frac{1}{n}$$

Multivariate NormalCDF





$$\int \cdots \int_m \frac{e^{\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}}{\sqrt{(2\pi)^m|\boldsymbol{\Sigma}|}} d^m x = \frac{1}{n^m}$$

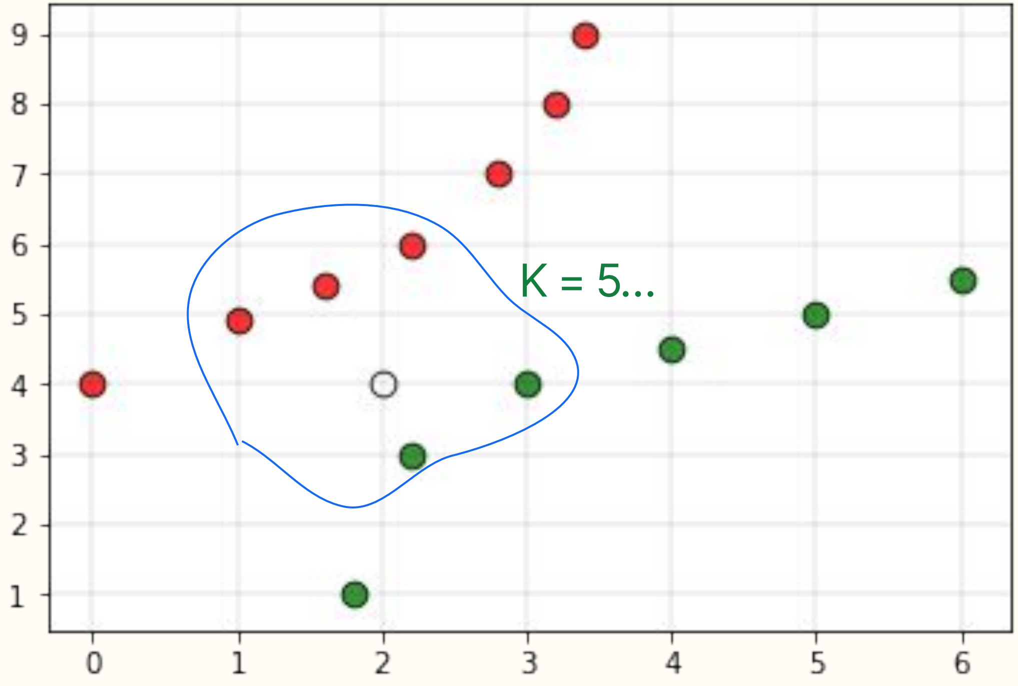
Lattice Build and Classification

- Generate the lattice points
- Compute the KNN classification at each point using proximity search
- Store the classifications in a lattice (dictionary)
  - Can be saved/loaded
- Classify instances by looking up the classification in the lattice

$$L_i = \frac{normalcdf(-\infty, x_i) \times (n+1)}{S}$$

0.1, 0.2, 0.3

Distance Weighting



The problem with a simple majority-voting approach

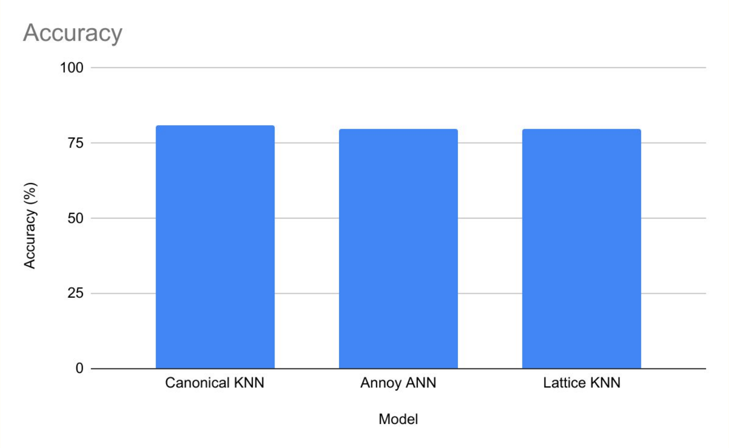
$$w = \frac{1}{(d + b_1)^{b_2}}$$

Distance Weighting

- Distance-weighting didn’t change much
    - Could be because k=3
    - Could be because there wasn’t much performance loss in the first place

Results - Accuracies

Model	**Accuracy**
Base KNN	80.8%
Proximity Search KNN	79.9%
3-slice Latticing Precomputation KNN	77.7%
4-slice Latticing Precomputation KNN	79.6%



Results - Confusion Matrices

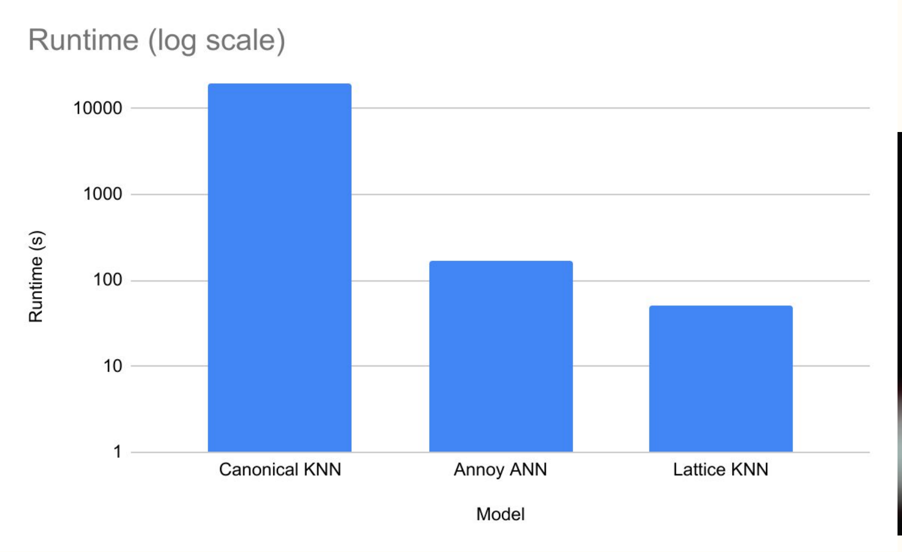
Model	Confusion Matrix
Base KNN	$\begin{array}{ccc} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & 254 & 77 \\ \mathbf{1} & 59 & 260 \end{array}$
Proximity Search KNN	$\begin{array}{ccc} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & 268 & 75 \\ \mathbf{1} & 58 & 249 \end{array}$
3-slice Latticing Precomputation KNN	$\begin{array}{ccc} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & 253 & 90 \\ \mathbf{1} & 44 & 263 \end{array}$

Now for the runtimes!

Drumroll please....

Results - Classification Runtimes (10000 runs)

Model	**Runtime (s)**
Base KNN	19500
Proximity Search KNN	171
Latticing Precomputation KNN	51



Conclusions

- Model building takes time
- Classification is sped up immensely
- Computation time is no longer dependent on number of instances: O(1)
- Minimal accuracy loss