Introduction to Modern AI
Week 6: Unsupervised Learning

Gavin Hartnett

PRGS, Winter Quarter 2022

Topics we will cover this week

- Information Theory
- K-Means Clustering
- ML in Large Dimensions
- PCA and Dimensional Reduction
- Generative Modeling and Density Estimation

Unsupervised Learning

- Unsupervised Learning comprises a wide range of modeling problems
- Defining characteristic: data is unlabeled (no Y variable)
- If there is no label/target, what is the goal exactly?
- This is why the term encompasses such a wide range of problems/models

Motivation

What fraction of your learning has been supervised vs unsupervised?

Source: https://research.umn.edu/inquiry/post/here-are-rules-overcoming-barriers-language-development

Source: https://mybabynursery.com.au/do-you-leave-your-baby-to-play-alone/

Motivation

- Most learning in nature seems to be close to the UL paradigm rather than the SL paradigm
- Clean, labeled data is expensive
- Unlabeled data is cheap, abundant, especially to large tech companies
- Unlabeled data still contains a wealth of information, surely there is some way to extract this

Information Theory

Information Theory

- Shannon: Information theory is the scientific study of the quantification, storage, and communication of digital information
- As such, Information Theory motivates many key algorithms and ideas in ML
- How should we attempt to quantify a concept such as "information"?
  - Likely events should have low information content
  - Unlikely events should have high information content
  - Information should be additive for independent events
- Motivates the concept of "self-information": $I(x) = -\log P(x)$

Entropy

- Entropy is a measure of the disorder, or uncertainty
- Initially introduced in physics as part of the development of thermodynamics
- *Shannon Entropy* applies to probability distributions
- Self-Information, averaged over an entire distribution: $H[P] = -\mathbb{E}_{x\sim P} \log P(x)$. Or:
- Discrete:

$$H[P] = -\sum_{x\in\mathcal{X}} P(x) \log P(x)$$

- Continuous:

$$H[P] = -\int_{x} P(x) \log P(x)$$

- Discrete entropy is always non-negative
- Continuous may be negative

Entropy: Examples

$$H[P] = -\mathbb{E}_{x\sim P} \log P(x)$$

- Deterministic distribution (minimal entropy):

$$P(x) = \begin{cases} 1 & x = x^* \\ 0 & \text{else} \end{cases}, \qquad H = -P(x^*) \log P(x^*) = -\log(1) = 0$$

- Uniform distribution (maximal entropy):

$$P(x) = \frac{1}{|\mathcal{X}|}, \qquad H = -\frac{1}{|\mathcal{X}|} \sum_{x\in\mathcal{X}} \log\left(\frac{1}{|\mathcal{X}|}\right) = \log|\mathcal{X}|$$

Entropy: Examples

- The Gaussian (normal) distribution $\mathcal{N}(\mu, \sigma^2)$ has the special property of being the distribution that maximizes entropy among all distributions with mean μ and std σ
- Maximally entropic distributions are special and widely used in statistics/ML
- Principal of Maximum Entropy
  - Among all distributions which are consistent with data, you should select the one with maximal entropy
  - In attempting to model a distribution, you should endeavor to make as few assumptions as possible (Occam's Razor)
  - Jaynes, Edwin T. "Information theory and statistical mechanics." Physical review 106.4 (1957): 620.
    Jaynes, Edwin T. "Information theory and statistical mechanics. II." Physical review 108.2 (1957): 171.

Kullback-Liebler (KL) Divergence

- KL divergence is a useful concept in information theory
- Statistical distance measure between two probability distributions
- Discrete:

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

- Continuous:

$$D_{KL}(P||Q) = \int_{x} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

- Some properties
  - $D_{KL}(P||Q) \geq 0$ with equality iff $P = Q$ almost everywhere
  - Asymmetric: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$
  - Invariant to changes of variables/reparametrizations
  - Simple relation to negative log likelihood (NLL):

$$D_{KL}(P||Q) = -\mathbb{E}_{x \sim P(x)} \log Q(x) + c_P$$

Kullback-Liebler (KL) Divergence

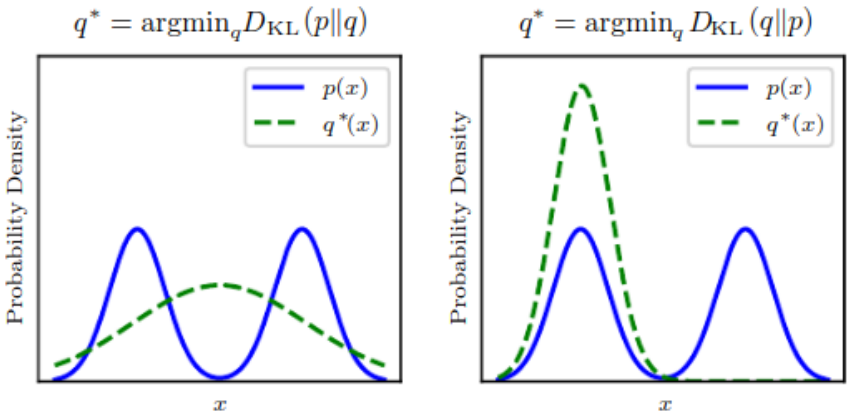


Figure 3.6: The KL divergence is asymmetric. Suppose we have a distribution $p(x)$ and wish to approximate it with another distribution $q(x)$. We have the choice of minimizing either $D_{\text{KL}}(p\|q)$ or $D_{\text{KL}}(q\|p)$. We illustrate the effect of this choice using a mixture of two Gaussians for p, and a single Gaussian for q. The choice of which direction of the KL divergence to use is problem dependent. Some applications require an approximation that usually places high probability anywhere that the true distribution places high probability, while other applications require an approximation that rarely places high probability anywhere that the true distribution places low probability. The choice of the direction of the KL divergence reflects which of these considerations takes priority for each application. *(Left)*The effect of minimizing $D_{\text{KL}}(p\|q)$. In this case, we select a q that has high probability where p has high probability. When p has multiple modes, q chooses to blur the modes together, in order to put high probability mass on all of them. *(Right)*The effect of minimizing $D_{\text{KL}}(q\|p)$. In this case, we select a q that has low probability where p has low probability. When p has multiple modes that are sufficiently widely separated, as in this figure, the KL divergence is minimized by choosing a single mode, to avoid putting probability mass in the low-probability areas between modes of p. Here, we illustrate the outcome when q is chosen to emphasize the left mode. We could also have achieved an equal value of the KL divergence by choosing the right mode. If the modes are not separated by a sufficiently strong low-probability region, then this direction of the KL divergence can still choose to blur the modes.

Clustering

Clustering

- Goal in clustering is to assign data points $\boldsymbol{x}$ to one of K clusters
- Similar to classification, but there is no "correct" cluster assignment
- Goal is to learn natural groupings within the data
- Items within a cluster should be "closer" to one another than to items in other clusters
- Many clustering algorithms exist, most popular is K-Means Clustering

K-Means Clustering

- Given
  - N observations in $\mathbb{R}^d$: $(\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$
  - Number of clusters K
- Find: Cluster assignments $c(i)$ and K mean vectors $\boldsymbol{\mu}_k$ that minimize

$$\text{loss}(c, \boldsymbol{\mu}) = \sum_{i=1}^{N} \sum_{k:c(i)=k} ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||_2^2$$

- Each cluster assigned a mean vector $\boldsymbol{\mu}_k$
- Each observation $\boldsymbol{x}_i$ is assigned to a cluster $c(i)$
- Assignments will minimize the variation within each cluster

K-Means Clustering

- loss function depends on K continuous vectors $\boldsymbol{\mu}_k$ and discrete cluster assignment function $c(i)$:

$$\text{loss}(c, \boldsymbol{\mu}) = \sum_{i=1}^{N} \sum_{k:c(i)=k} ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||_2^2$$

- So gradient descent not possible
- Instead, let's alternate b/w updating c and $\boldsymbol{\mu}_k$
  - For each $i = 1, ..., N$, update c:

$$c(i) = \text{argmin}_k ||\boldsymbol{x}_i - \boldsymbol{\mu}_k||^2$$

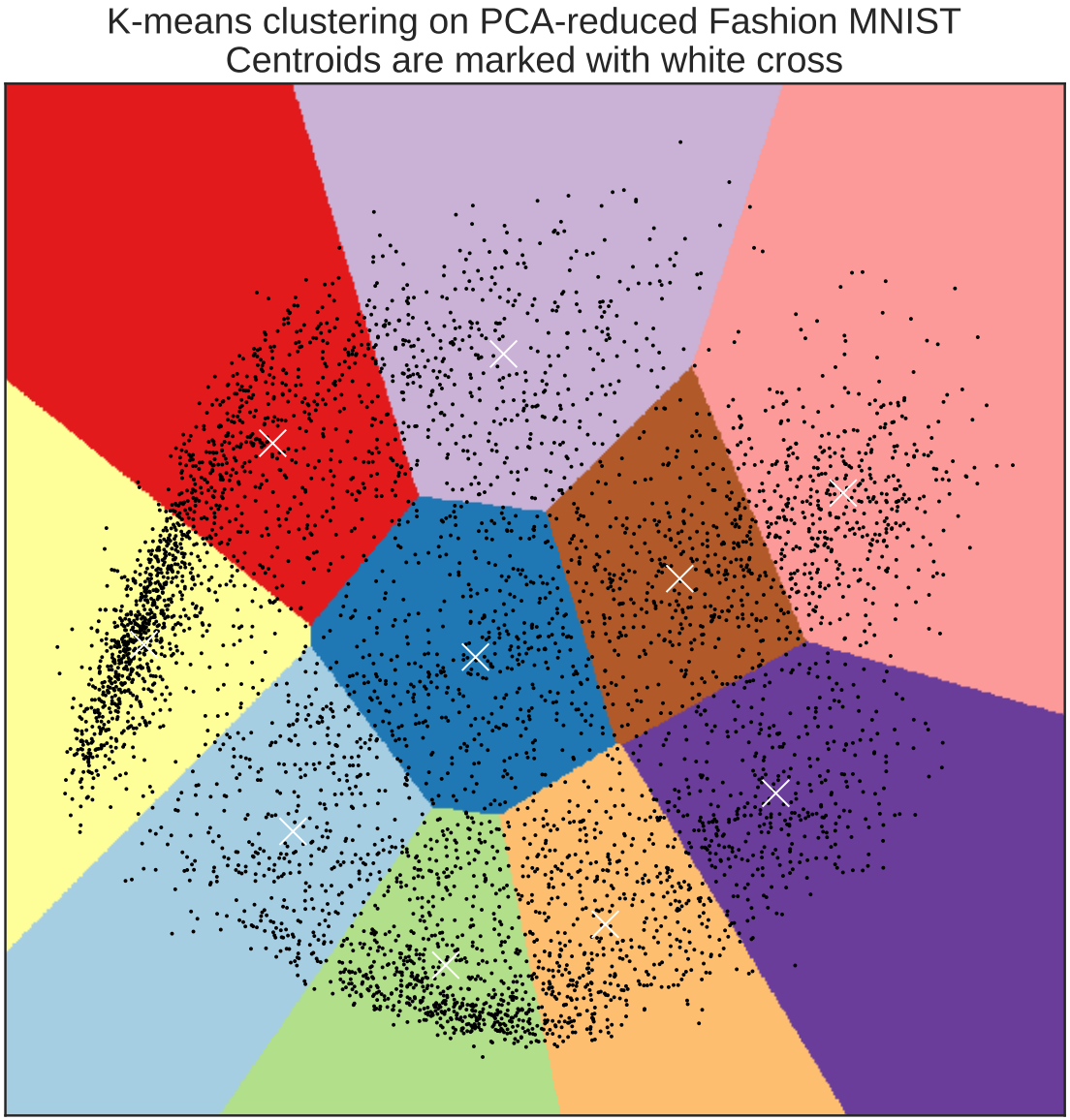
  - For each $k = 1, .., K$, update $\boldsymbol{\mu}_k$:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:c(i)=k} \boldsymbol{x}_i$$

K-Means Clustering

- The above algorithm does not always yield the global minimum
- In practice one typically runs the algorithm for many random initializations and takes best solution
- Assigning each point to the nearest mean produces a Voronoi diagram/tesselation

K-Means Clustering



ML in Large Dimensions

The Curse of Dimensionality

- There are a number of phenomena associated with high-dimensional data that are collectively referred to as the *curse of dimensionality*
- These afflict all ML problems, especially relevant for density estimation/generative modeling

The Curse of Dimensionality: Sampling

- As dimension of data grows, the coverage provided by a finite-sized sample decreases rapidly
- Consider $\boldsymbol{x} \in \mathbb{R}^d$. If we want 100 evenly spaced points for each dimension (x_i value), we need 100^d points - increases exponentially with d
- Consider an 256-bit color image with n pixels, then there are 256^{3n^2} possible images. Grows doubly exponentially in n. Thus, a realistically dataset $|\mathcal{D}|$ rapidly represents a vanishing small fraction of all possible images, and each image can be expected to appear only once.
  - Histograms quickly become uninformative in high D

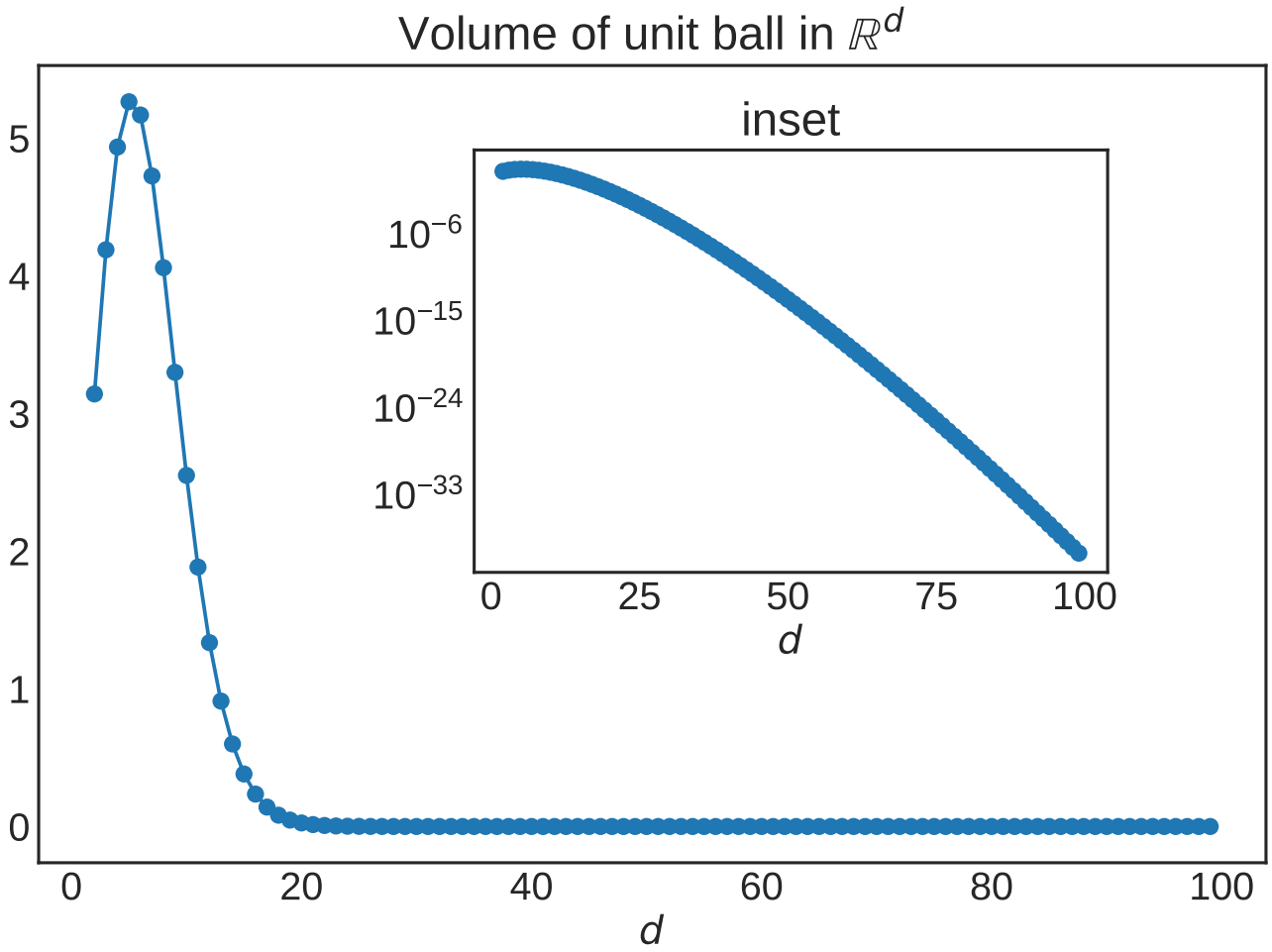
The Curse of Dimensionality: Geometric Considerations

- There are also a number of simple geometric considerations that play a role here
- Basic idea is that your $2d$ or $3d$ spatial intuition can be misleading in large d
- Consider the volume of a d-dimensional radius R ball:

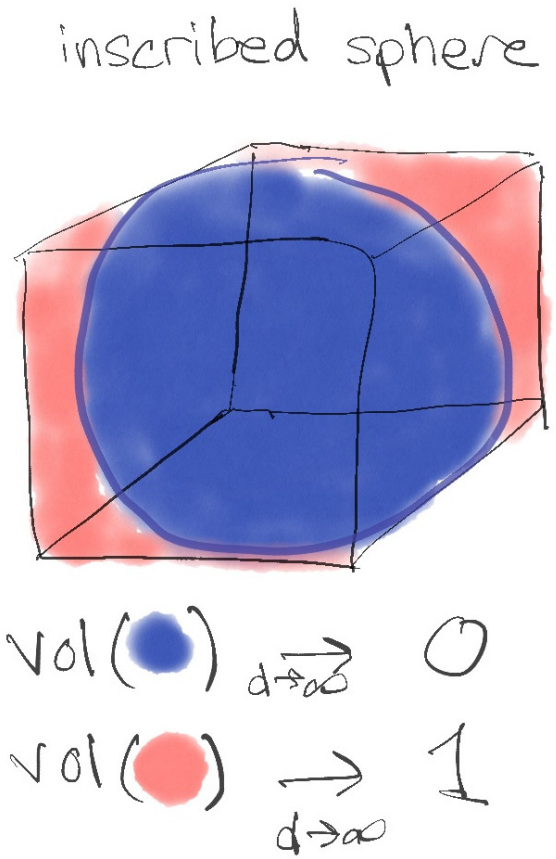
$$V_d(R) = \frac{\pi^{d/2}}{\Gamma(1 + \frac{d}{2})} R^d$$

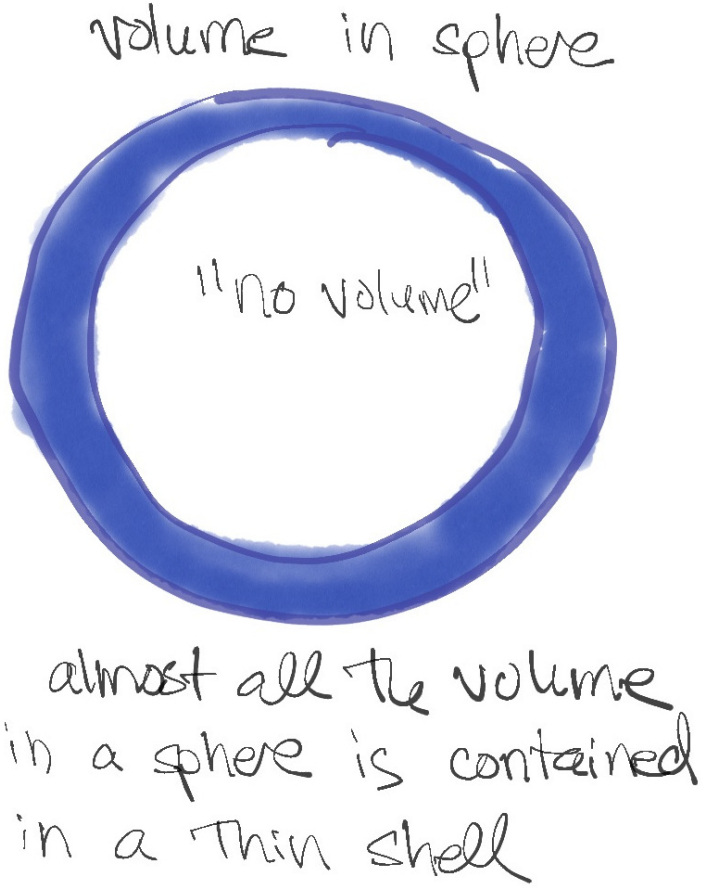
- Fixing R and letting d grow large, $V_d(R) \sim d^{-d/2}$: decreases exponentially
- Fixing d and letting R grow, $V_d(R) \sim R^d$: increases exponentially

The Curse of Dimensionality: Geometric Considerations



The Curse of Dimensionality: Geometric Considerations





The Curse of Dimensionality: Implications

- Underscores the difficulty of density estimation
  - Almost all histogram bins contain 0 observations
  - A small handful have 1 observation
  - Most likely none have 2 observations or more
- Signals a difficulty with nearest-neighbor or distance-based methods
  - Hard to get close in large-d

The Manifold Hypothesis

- Why does ML work at all?
  - Most ML applications involve very large dimensions
  - Data represents a very sparse sample in the space of all possibilities
  - Why should we expect these models to generalize at all?
- *Manifold Hypothesis*: probability distributions over real-world data often lies on a low-dimensional manifold
  - (Weak) point in favor: uniform sampling almost never produces a realistic image/document/etc (think of the infinite monkey theorem)
  - Also requires that space of realistic observations is connected

PCA and Dimensional Reduction

Dimensional Reduction

- There are many reasons why we might like to reduce the dimensionality of the data
- Visualizations
- Some dimensions might be noisier than others - dropping these could help
- Might be more convenient to work with fewer variables
- Data might be intrinsically low-dimensional (manifold hypothesis)

Principal Components Analysis (PCA)

- Principal components provide best linear approximation to data in lower dimension
- Relies on a simple matrix factorization called the Singular Value Decomposition (SVD)
- Let $\boldsymbol{X}$ be $(N \times p)$ matrix. Then the SVD is

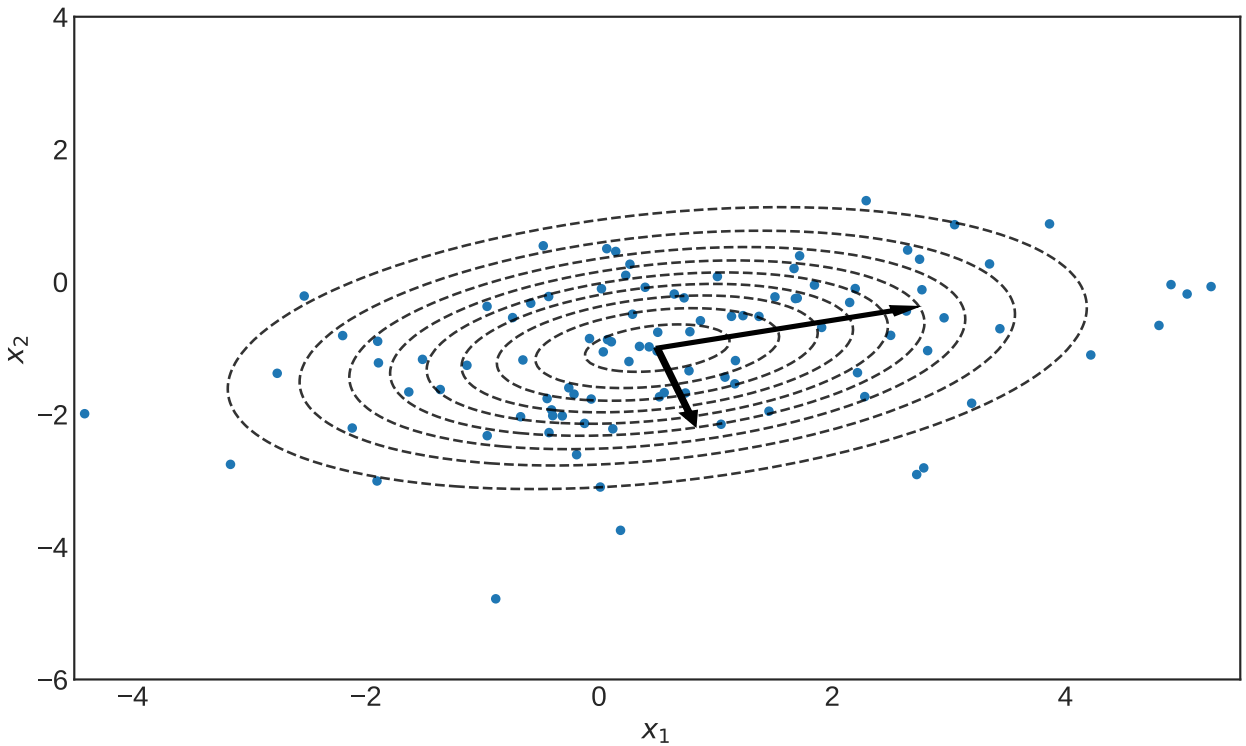
$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

  - $\boldsymbol{U}$ $(N \times N)$ (columns are left singular vectors)
  - $\boldsymbol{V}$ $(p \times p)$ (columns are right singular vectors)
  - $\boldsymbol{D}$ is $(N \times p)$ diagonal matrix, entries are called singular values
  - Singular values are sorted: $0 \leq d_p \leq d_{p-1} \leq ... \leq d_1$
  - Note: different conventions exist for SVD!
- When $\boldsymbol{X}$ is a data matrix with mean removed, singular vectors are the principal components

Principal Components Analysis (PCA): Example

- Consider 2D Gaussian

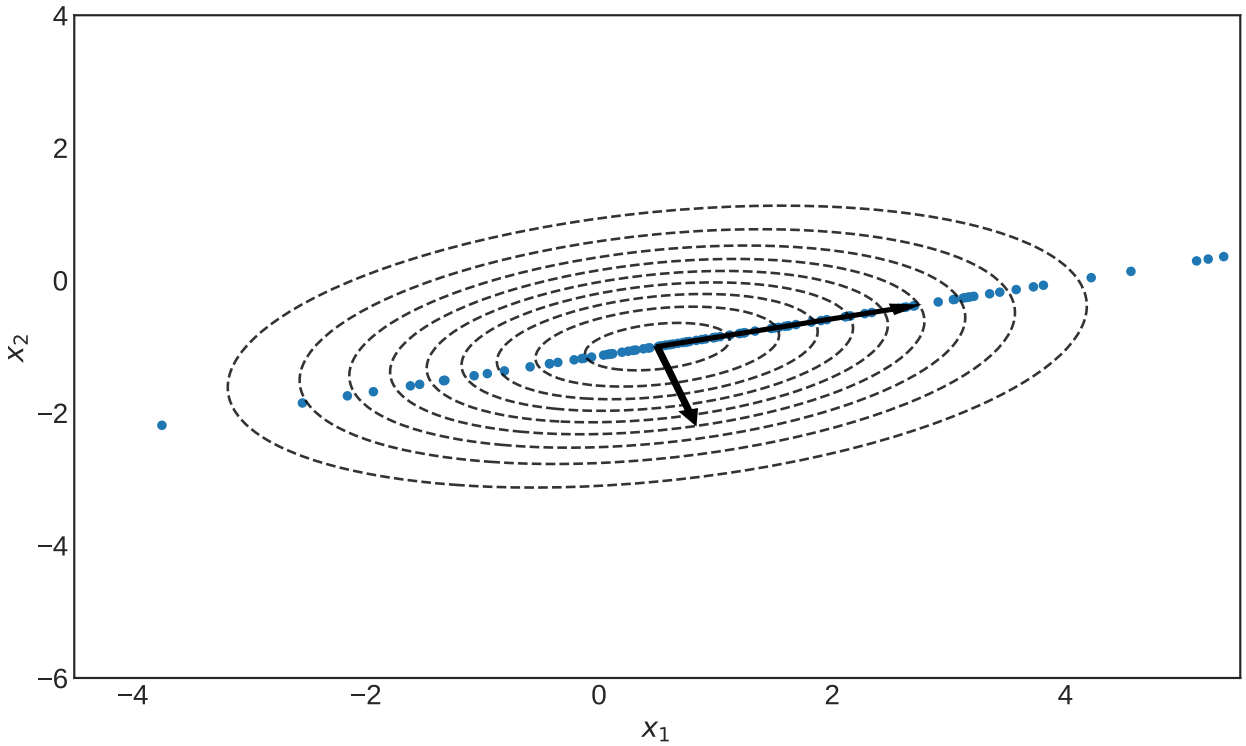
$$\boldsymbol{\mu} = \begin{pmatrix} 0.5 \\ -1 \end{pmatrix}, \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 3 & 0.5 \\ 0.5 & 1 \end{pmatrix}.$$



Principal Components Analysis (PCA)

- Key idea: project data down to smaller dimensions using $q \ll p$ of the principal components
- Let $\boldsymbol{V}_q$ be the matrix $\boldsymbol{V}$ with only the first q singular vectors retained
- Can project down from $\mathbb{R}^p$ to $\mathbb{R}^q$ via:

$$\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{V}_q$$



Generative Modeling and Density Estimation

Discriminative vs. Generative Modeling

- ML methods are sometimes divided into *discriminative* and *generative* approaches
  - discriminative: model $P(Y|X = x)$
  - generative: model $P(X|Y = y)$ or $P(X, Y)$
- The definition is not universally agreed upon
- Two definitions of generative related by a simple factor:

$$P(X, Y) = P(X|Y)\underbrace{P(Y)}_{\text{prior}}$$

- It's generally understood that Y is low-dimensional (scalar output or class labels) and X is the higher-dimensional feature space

Discriminative vs. Generative Modeling

- Bayes theorem can be used to convert from one model to another:

$$P(Y|X) = P(X|Y)\frac{P(Y)}{P(X)}$$

- $P(Y)$ - prior, easy to estimate
- $P(X)$ - hard to estimate, but doesn't depend on classes (Y)
- These discussion assumes that the Y variable exists, i.e., it applies to supervised learning
- Examples (we'll encounter these next week)
  - Naive Bayes
  - Latent Dirichlet Allocation (LDA)
- Focus this week: *generative modeling* is also used to refer to the problem of modeling $P(X)$ (unsupervised learning)

Generative Modeling

- This problem of estimating $P(X)$ is also called *density estimation*
- (Though much of ML involves estimating a distribution)
- The word generative refers to the idea that we have learned a model capable of generating the data, not just modeling $P(X)$
- In other words, we should be able to efficiently sample $P(X)$ also

Kernel Density Estimation (KDE)

- Given a set of data points $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$ drawn from a distribution $f(\boldsymbol{x})$, build an estimate of that distribution $\hat{f}(\boldsymbol{x})$
- Not generative, does not allow for $\hat{f}(\boldsymbol{x})$ to be easily sampled
- Also called Parzen windows or Parzen–Rosenblatt windows

$$\hat{f}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_i)$$

- Here K_h is the kernel, typically a Gaussian kernel is used

$$K_h(\boldsymbol{x}, \boldsymbol{x}') = \frac{\exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{x}'||_2^2}{2h^2}\right)}{\sqrt{2\pi h^2}}$$

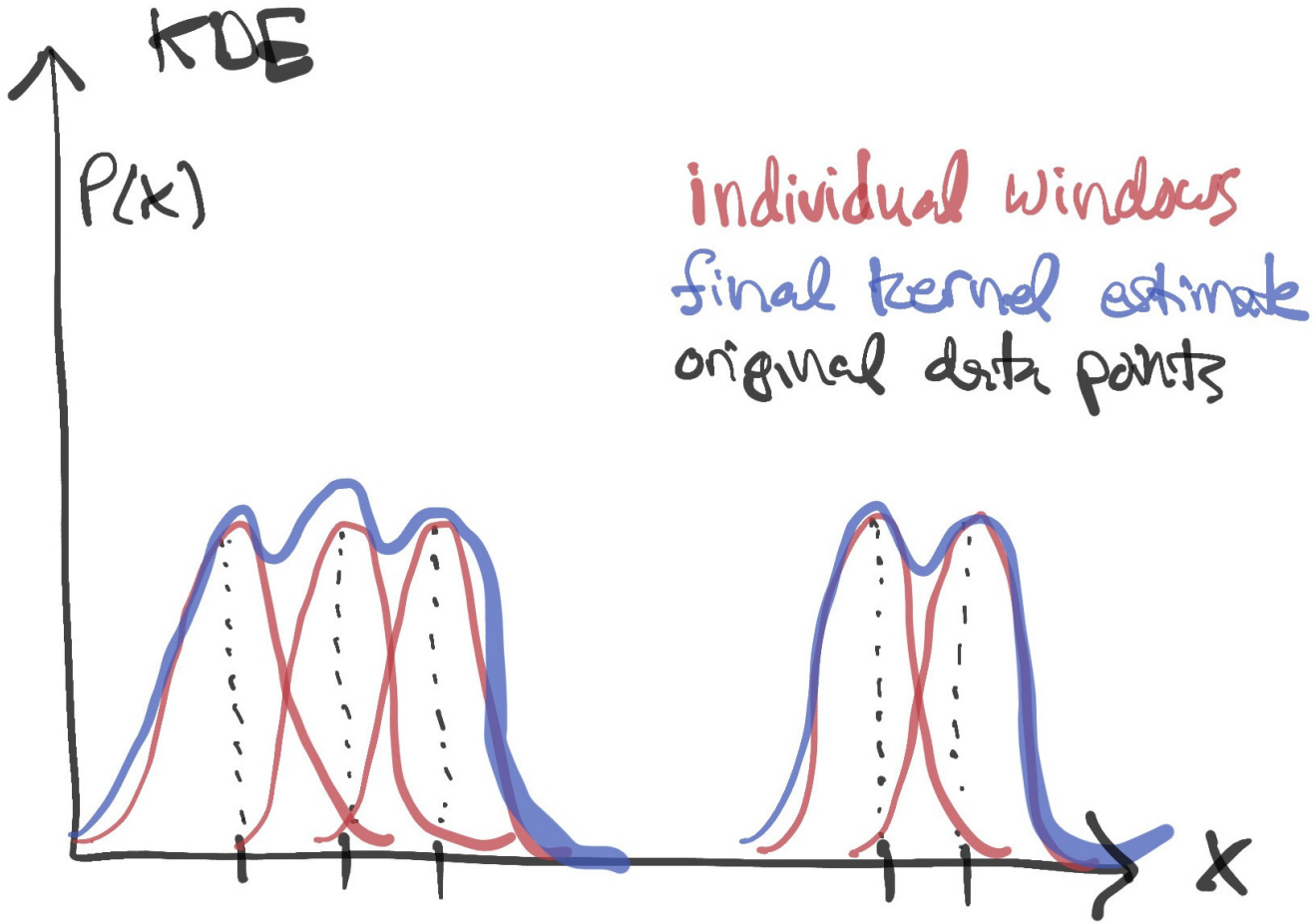
Kernel Density Estimation (KDE)

- Density estimate around point $\boldsymbol{x}$ is a Gaussian-weighted average of the nearby points in the dataset

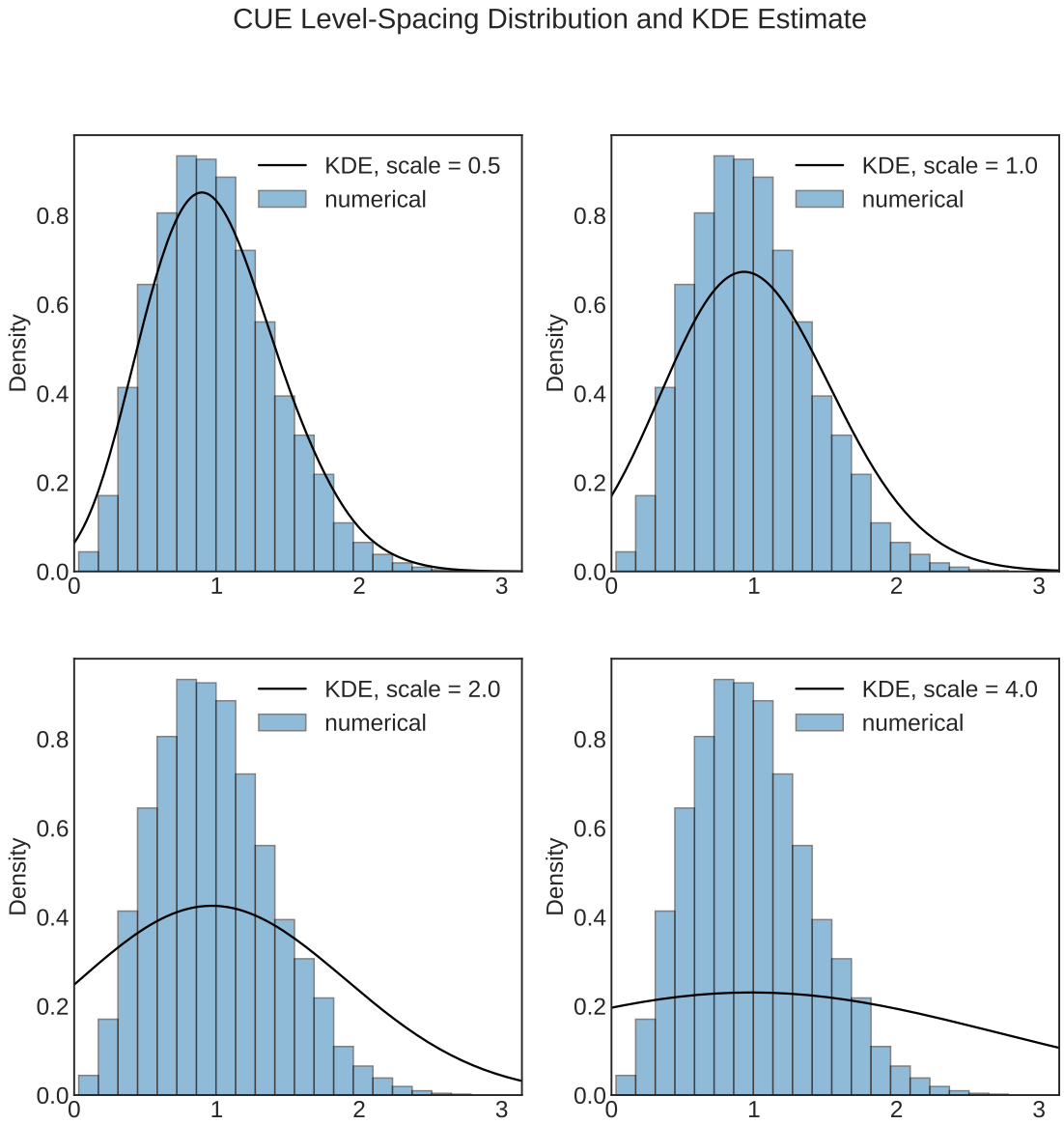
$$\hat{f}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_i)\,, \qquad K_h(\boldsymbol{x}, \boldsymbol{x}') = \frac{\exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{x}'||_2^2}{2h^2}\right)}{\sqrt{2\pi h^2}}\,.$$

- Closer points count more, contribution rapidly falls as distance $||\boldsymbol{x} - \boldsymbol{x}'||_2/h$ increases
- h is the width, sets the scale for what counts as close vs far

Kernel Density Estimation (KDE)



Kernel Density Estimation (KDE)



Deep Generative Modeling

- KDE only models a distribution, does not allow us to sample it efficiently
- KDE distribution is a mixture of Gaussians, hard to sample in general
- Many deep neural network-based generative models exist which do allow for efficient sampling:
  - Generative Adversarial Networks (GANs)
  - Normalizing Flows (NFs)
  - Boltzmann Machines (BMs)
  - Variational Auto-Encoders (VAE)
- We will cover these in the advanced class