STA 314: Statistical Methods for Machine Learning I

Lecture 7 - Unsupervised Learning, K-Means

Chris J. Maddison

University of Toronto

Unsupervised learning

- What can we do without labels?
- How can we even define what learning means without labels?
- Unsupervised learning is the study of learning without labels.
- In some sense, the ML community does not exactly agree on what it means to do unsupervised learning, but intuitively, unsupervised learning is the task of grouping, explaining, and finding structure data.

Motivating Examples

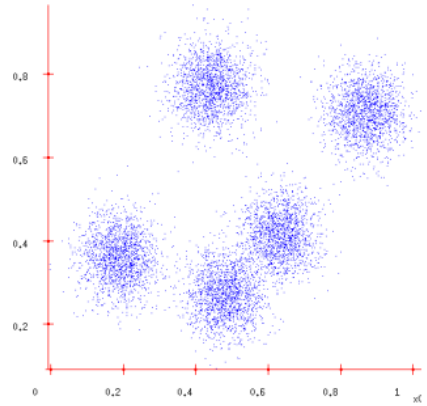
- Some examples of situations where you'd use unupservised learning
  - You want to understand how a scientific field has changed over time. You want to take a large database of papers and model how the distribution of topics changes from year to year. But what are the topics?
  - You're a biologist studying animal behavior, so you want to infer a high-level description of their behavior from video. You don't know the set of behaviors ahead of time.
  - You want to reduce your energy consumption, so you take a time series of your energy consumption over time, and try to break it down into separate components (refrigerator, washing machine, etc.).
- Common themes:
  - you have some data, and you want to infer some structure underlying the data.
  - you have some data, and you want to be able to make more data that looks similar

Clustering

- Today we will look at the simplest type of unsupervised learning: clustering.
- Clustering is the task of organizing data into groups or clusters.
- We will study the simplest method for doing this: the K-means algorithm.

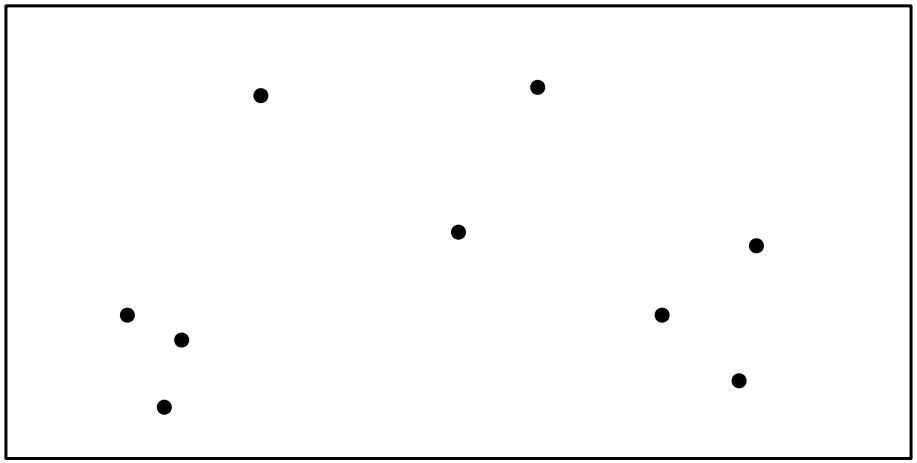
Clustering

- Sometimes the data form clusters, where samples within a cluster are similar to each other, and samples in different clusters are dissimilar:



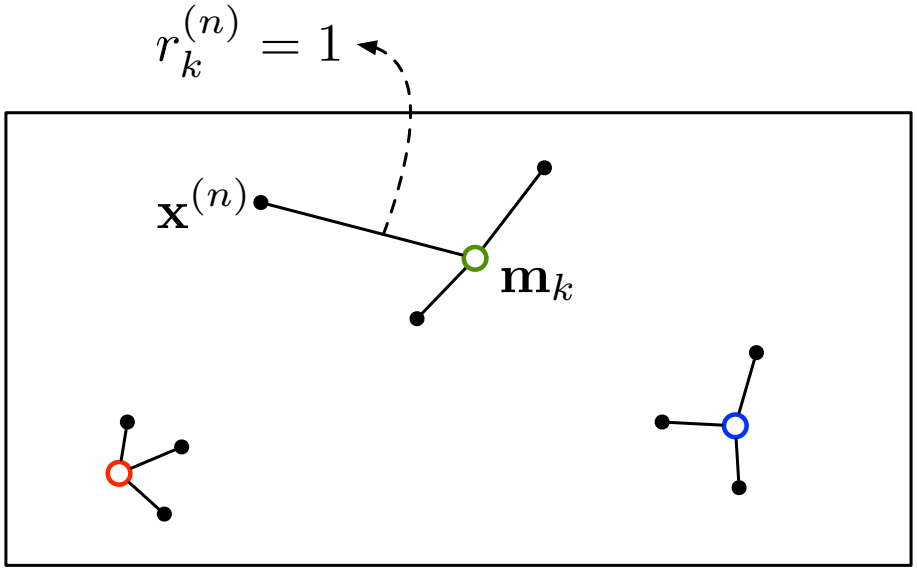
- Such a distribution is multimodal, since it has multiple modes, or regions of high probability mass.
- Grouping data points into clusters, with no observed labels, is called clustering. It is an unsupervised learning technique.
- E.g. clustering machine learning papers based on topic (deep learning, Bayesian models, etc.)
  - But topics are never observed (unsupervised).

Clustering problem



- Assume the data $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ lives in a Euclidean space, $\mathbf{x}^{(n)} \in \mathbb{R}^D$.
- Assume each data point belongs to one of K clusters
- Assume the data points from same cluster are similar, i.e. close in Euclidean distance.
- How can we identify those clusters (data points that belong to each cluster)? Let's formulate as an optimization problem.

K-means Objective



K-means Objective: Find centers $\{\mathbf{m}_k\}_{k=1}^K$ and assignments $\{\mathbf{r}^{(n)}\}_{n=1}^N$ to minimize the sum of squared distances of data points $\{\mathbf{x}^{(n)}\}$ to their assigned centers.

- Data sample $n = 1, .., N$: $\mathbf{x}^{(n)} \in \mathbb{R}^D$ (observed),
- Cluster center $k = 1, .., K$: $\mathbf{m}_k \in \mathbb{R}^D$ (not observed),
- Responsibilities: Cluster assignment for sample n: $\mathbf{r}^{(n)} \in \mathbb{R}^K$ 1-of-K encoding (not observed)

K-means Objective

- K-means Objective: Find cluster centers $\{\mathbf{m}_k\}_{k=1}^K$ and assignments $\{\mathbf{r}^{(n)}\}_{n=1}^N$ to minimize the sum of squared distances of data points $\{\mathbf{x}^{(n)}\}$ to their assigned centers.
  - Data sample $n = 1, .., N$: $\mathbf{x}^{(n)} \in \mathbb{R}^D$ (observed),
  - Cluster center $k = 1, .., K$: $\mathbf{m}_k \in \mathbb{R}^D$ (not observed),
  - Responsibilities: Cluster assignment for sample n: $\mathbf{r}^{(n)} \in \mathbb{R}^K$ 1-of-K encoding (not observed)
- Mathematically:

$$\min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} \hat{\mathcal{R}}(\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}) = \min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

  where $r_k^{(n)} = \mathbb{I}[\mathbf{x}^{(n)}$ is assigned to cluster $k]$, i.e., $\mathbf{r}^{(n)} = [0, .., 1, .., 0]^\top$
- Finding an optimal solution is an NP-hard problem!

K-means Objective

- Optimization problem:

$$\min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} \sum_{n=1}^{N} \underbrace{\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2}_{\substack{\text{distance between } x^{(n)} \\ \text{and its assigned cluster center}}}$$

- Since $r_k^{(n)} = \mathbb{I}[\mathbf{x}^{(n)}$ is assigned to cluster $k]$, i.e., $\mathbf{r}^{(n)} = [0, .., 1, .., 0]^\top$
- inner sum is over K terms but only one of them is non-zero.
- E.g. say sample $\mathbf{x}^{(n)}$ is assigned to cluster $k = 3$, then

$$\mathbf{r}^n = [0, 0, 1, 0, ...]$$

$$\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2 = ||\mathbf{m}_3 - \mathbf{x}^{(n)}||^2$$

How to optimize?: Alternating Minimization

Optimization problem:

$$\min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

- Problem is hard when minimizing jointly over the parameters $\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}$
- But note that if we fix one and minimize over the other, then it becomes easy.
- Doesn't guarantee the same solution!

How to optimize?: Alternating Minimization

Optimization problem:

$$\min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

- Fix the centers $\{\mathbf{m}_k\}$ then we can easily find the optimal assignments $\{\mathbf{r}^{(n)}\}$ for each sample n

$$\min_{\mathbf{r}^{(n)}} \sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

  - Assign each point to the cluster with the nearest center

$$r_k^{(n)} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\mathbf{x}^{(n)} - \mathbf{m}_j||^2 \\ 0 & \text{otherwise} \end{cases}$$

  - E.g. if $\mathbf{x}^{(n)}$ is assigned to cluster $\hat{k}$,

$$\mathbf{r}^{(n)} = \underbrace{[0, 0, ..., 1, ..., 0]^\top}_{\text{Only } \hat{k}\text{-th entry is 1}}$$

Alternating Minimization

- Likewise, if we fix the assignments $\{\mathbf{r}^{(n)}\}$ then can easily find optimal centers $\{\mathbf{m}_k\}$
  - Set each cluster's center to the average of its assigned data points: For $l = 1, 2, ..., K$

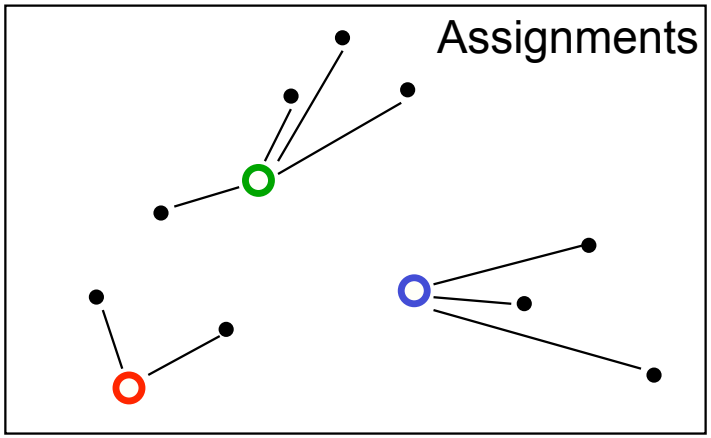
$$
\begin{aligned}
0 =& \frac{\partial}{\partial \mathbf{m}_l} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2 \\
=& 2 \sum_{n=1}^{N} r_l^{(n)} (\mathbf{m}_l - \mathbf{x}^{(n)}) \quad \Longrightarrow \quad \mathbf{m}_l = \frac{\sum_n r_l^{(n)} \mathbf{x}^{(n)}}{\sum_n r_l^{(n)}}
\end{aligned}
$$

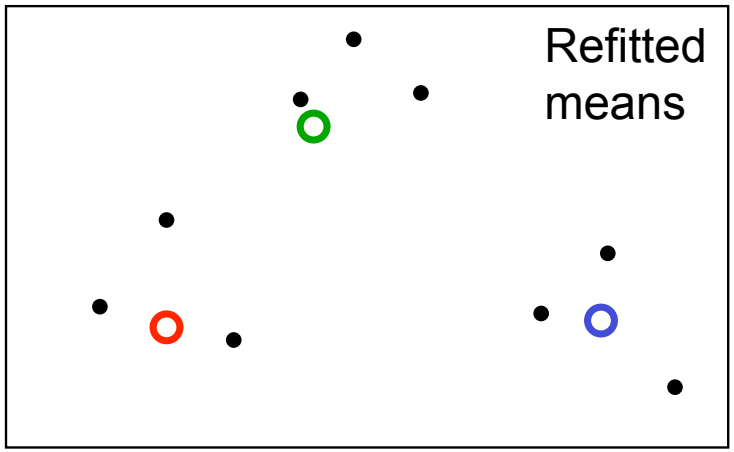
- Let's alternate between minimizing $\hat{\mathcal{R}}(\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\})$ with respect to $\{\mathbf{m}_k\}$ and $\{\mathbf{r}^{(n)}\}$
- This is called alternating minimization

K-means Algorithm

High level overview of algorithm:

- Initialization: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - Assignment step: Assign each data point to the closest cluster
  - Refitting step: Move each cluster center to the mean of the data assigned to it





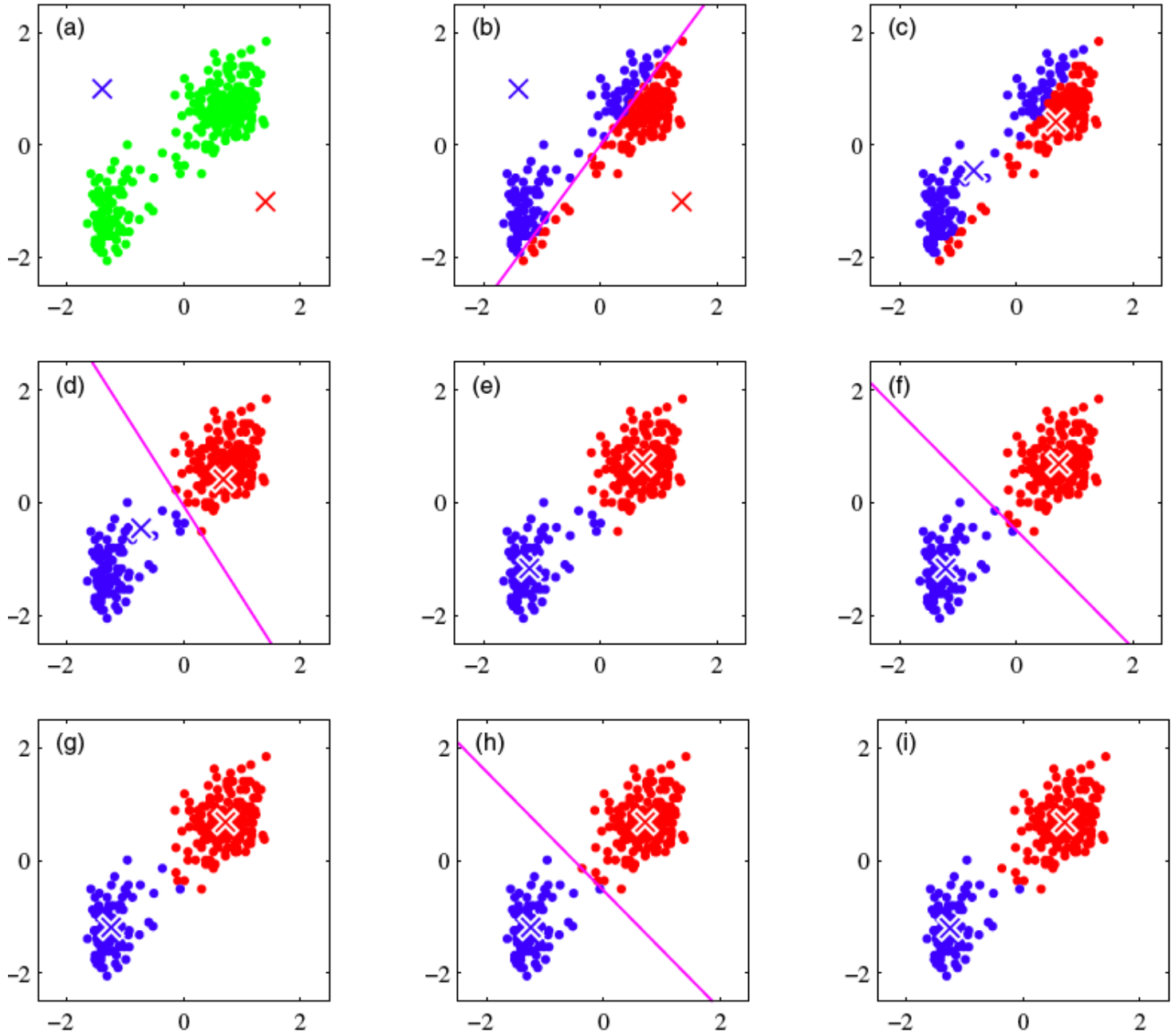


Figure from Bishop

Simple demo: http://syskall.com/kmeans.js/

The K-means Algorithm

- Initialization: Set K cluster means $\mathbf{m}_1, \ldots, \mathbf{m}_K$ to random values
- Repeat until convergence (until assignments do not change):
  - Assignment: Optimize $\hat{\mathcal{R}}$ w.r.t. $\{\mathbf{r}\}$: Each data point $\mathbf{x}^{(n)}$ assigned to nearest center

$$\hat{k}^{(n)} = \arg\min_k ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

    and Responsibilities (1-hot or 1-of-K encoding)

$$r_k^{(n)} = \mathbb{I}[\hat{k}^{(n)} = k] \quad \text{for} \quad k = 1, .., K$$

  - Refitting: Optimize $\hat{\mathcal{R}}$ w.r.t. $\{\mathbf{m}\}$: Each center is set to mean of data assigned to it

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}.$$

K-means for Vector Quantization

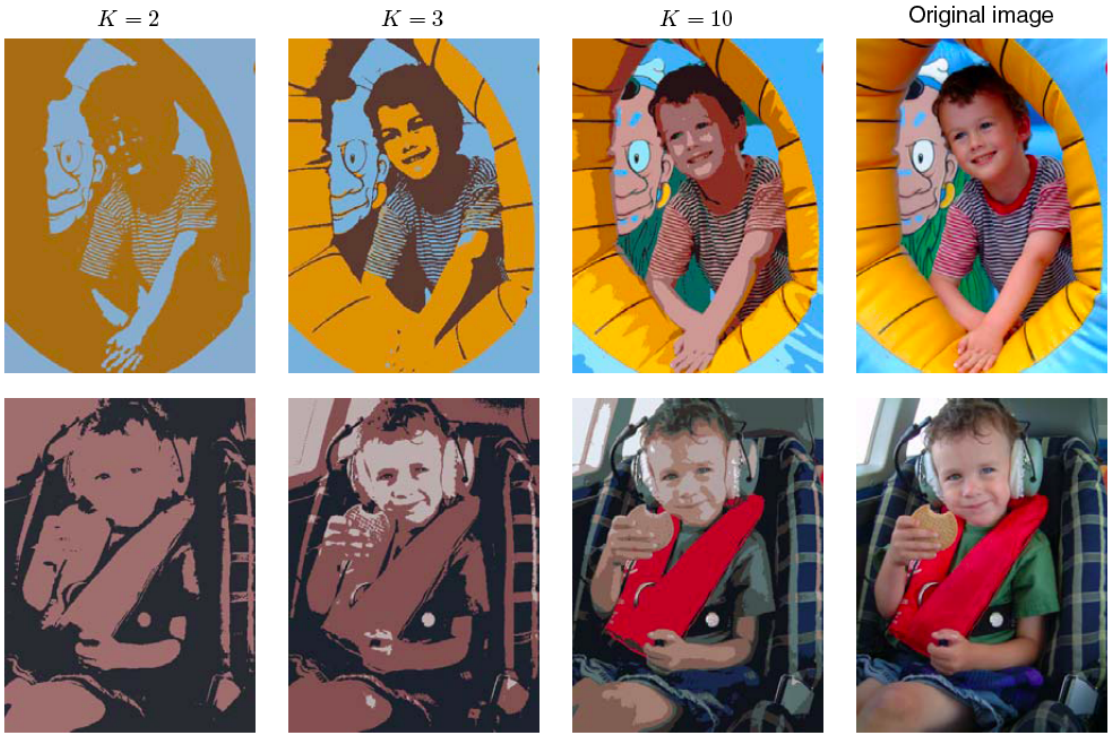
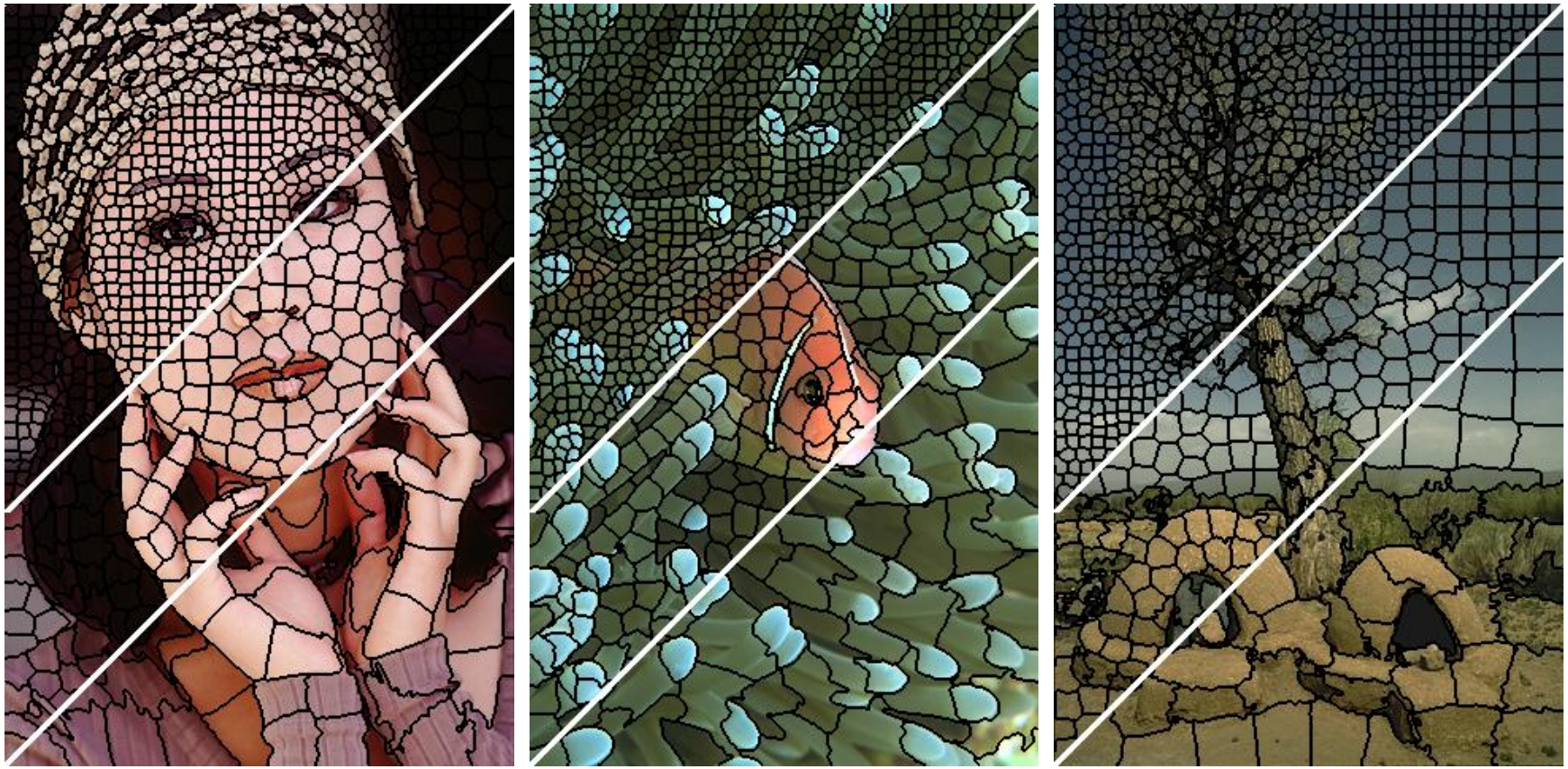


Figure from Bishop

- Given image, construct "dataset" of pixels represented by their RGB pixel intensities
- Run k-means, replace each pixel by its cluster center

K-means for Image Segmentation



- Given image, construct "dataset" of pixels, represented by their RGB pixel intensities and grid locations
- Run k-means (with some modifications) to get superpixels

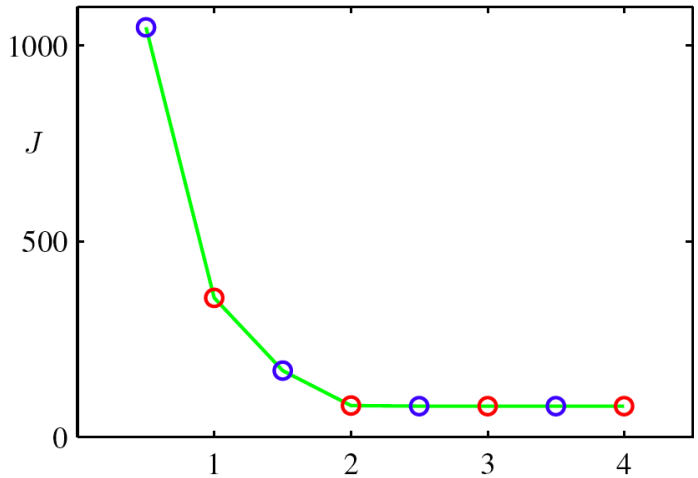
Questions about K-means

- Why does update set $\mathbf{m}_k$ to mean of assigned points?
- What if we used a different distance measure?
- How can we choose the best distance?
- How to choose K?
- Will it converge?

Hard cases – unequal spreads, non-circular spreads, in-between points

Why K-means Converges

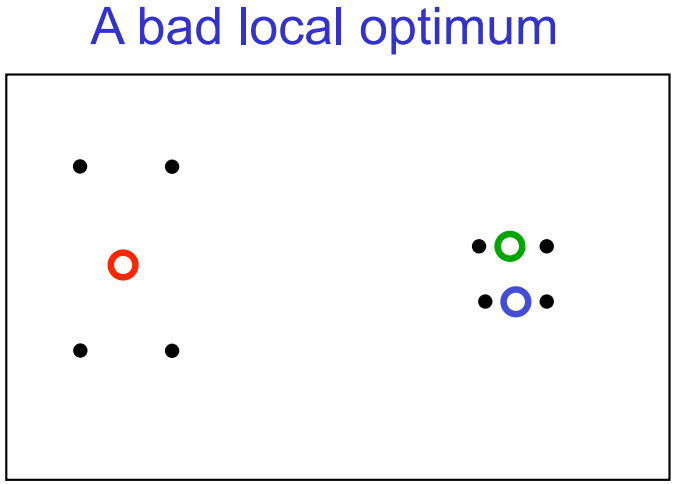
- K-means algorithm reduces the cost at each iteration.
  - Whenever an assignment is changed, the sum squared distances $\hat{\mathcal{R}}$ of data points from their assigned cluster centers is reduced.
  - Whenever a cluster center is moved, $\hat{\mathcal{R}}$ is reduced.
- Test for convergence: If the assignments do not change in the assignment step, we have converged (to at least a local minimum).
- This will always happen after a finite number of iterations, since the number of possible cluster assignments is finite



- K-means cost function after each assignment step (blue) and refitting step (red). The algorithm has converged after the third refitting step.

Local Minima

- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points



Soft K-means

- Instead of making hard assignments of data points to clusters, we can make soft assignments. One cluster may have a responsibility of .7 for a datapoint and another may have a responsibility of .3.
  - Allows a cluster to use more information about the data in the refitting step.
  - How do we decide on the soft assignments?
  - We already saw this in multi-class classification:
    - 1-of-K encoding vs softmax assignments

Soft K-means Algorithm

- Initialization: Set K means $\{\mathbf{m}_k\}$ to random values
- Repeat until convergence (measured by how much $\hat{\mathcal{R}}$ changes):
  - Assignment: Each data point n given soft "degree of assignment" to each cluster mean k, based on responsibilities

$$r_k^{(n)} = \frac{\exp[-\beta \|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2]}{\sum_j \exp[-\beta \|\mathbf{m}_j - \mathbf{x}^{(n)}\|^2]}$$

$$\implies \mathbf{r}^{(n)} = \text{softmax}(-\beta\{\|\mathbf{m}_k - \mathbf{x}^{(n)}\|^2\}_{k=1}^K)$$

  - Refitting: Model parameters, means, are adjusted to match sample means of datapoints they are responsible for:

$$\mathbf{m}_k = \frac{\sum_n r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

Questions about Soft K-means

Some remaining issues

- How to set β?
- Clusters with unequal weight and width?

These aren't straightforward to address with K-means. Instead, in the sequel, we'll reformulate clustering using a generative model.

As $\beta \to \infty$, soft k-Means becomes k-Means! (Exercise)

PCA Overview

- We now turn to the second unsupervised learning algorithm for this course: principal component analysis (PCA)
- Dimensionality reduction: map data to a lower dimensional space
  - Save computation/memory
  - Reduce overfitting, achieve better generalization
  - Visualize in 2 dimensions
- PCA is a linear model. It's useful for understanding lots of other algorithms.
  - Autoencoders
  - Matrix factorizations (next week)
- PCA is finding linear structure in data.

Gaussian distribution

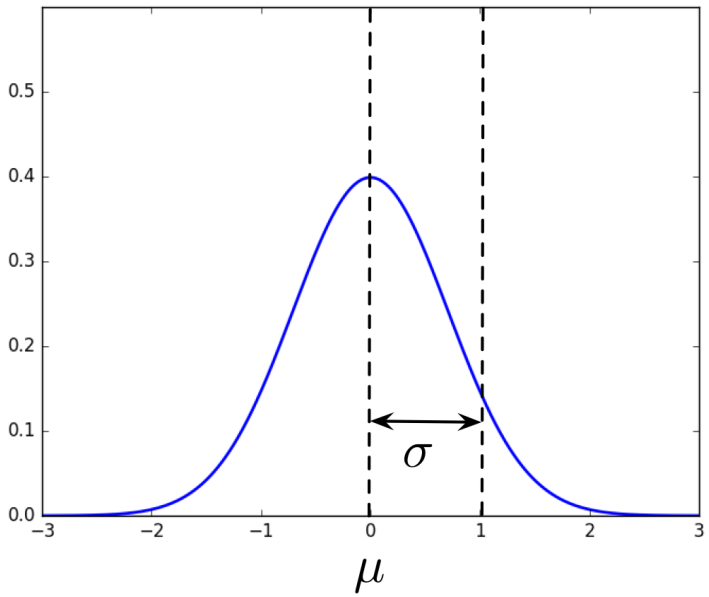
- First, we will review the Gaussian, or normal, distribution.
- The Central Limit Theorem says that sums of lots of independent random variables are approximately Gaussian. So, Gaussians show up everywhere in nature.
- In machine learning, we use Gaussians a lot because they make the calculations easy.

Gaussian distribution

- The normal distribution is parameterized by $\mu \in \mathbb{R}$ and $\sigma^2 > 0$:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- If you know the mean and the variance, then you can identify the Gaussian distribution exactly.



Shifting + scaling univariate Gaussians

All univariate Gaussians are shifted and scaled version of the standard normal: if $\epsilon \sim \mathcal{N}(0, 1)$, then $x = \mu + \sigma\epsilon$ has distribution $\mathcal{N}(\mu, \sigma^2)$.

Proof:

$$\mathbb{E}[x] = \mu + \sigma\mathbb{E}[\epsilon] = \mu$$

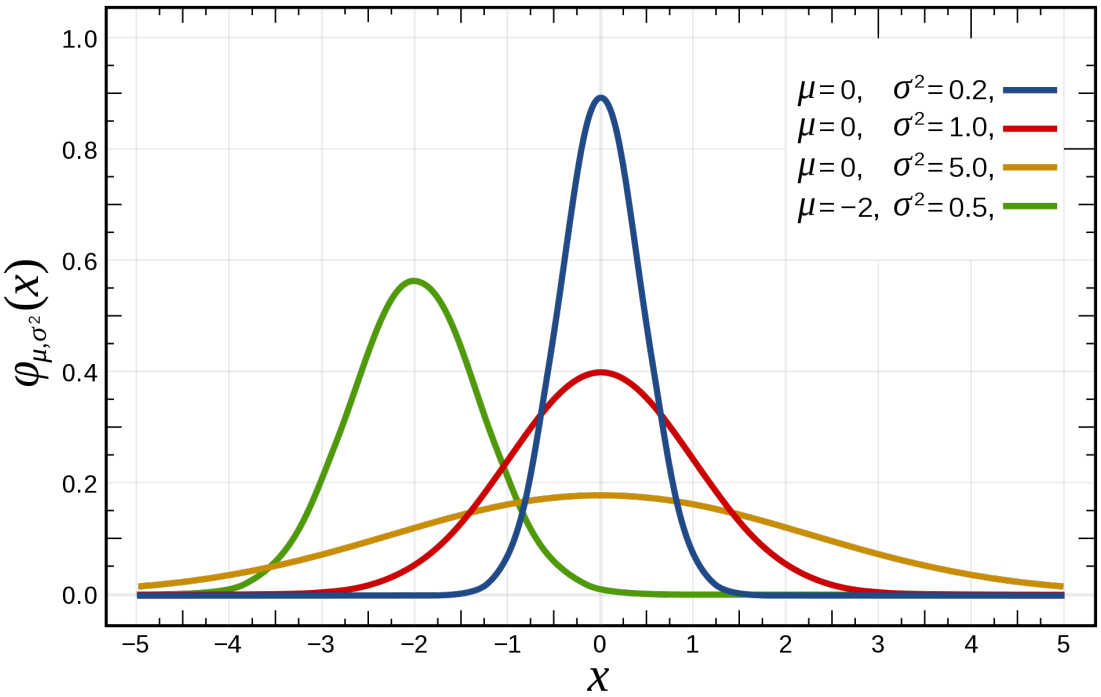
and

$$\mathrm{Var}(x) = \sigma^2\mathrm{Var}(\epsilon) = \sigma^2$$

Since a Gaussian is defined by its mean and its variance, we've shown that $x \sim \mathcal{N}(\mu, \sigma^2)$.

Shifting + scaling univariate Gaussians

I.e., densities are related to the standard normal by scaling by the square root σ and shifting by the mean μ:



Multivariate Gaussian statistics

- Similar intuitions apply to multivariate Gaussians!
- A multivariate Gaussian (or multivariate Normal) distribution is uniquely defined by a mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, denoted $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ or $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$
- Mean is a vector

$$\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}] = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_d \end{pmatrix}$$

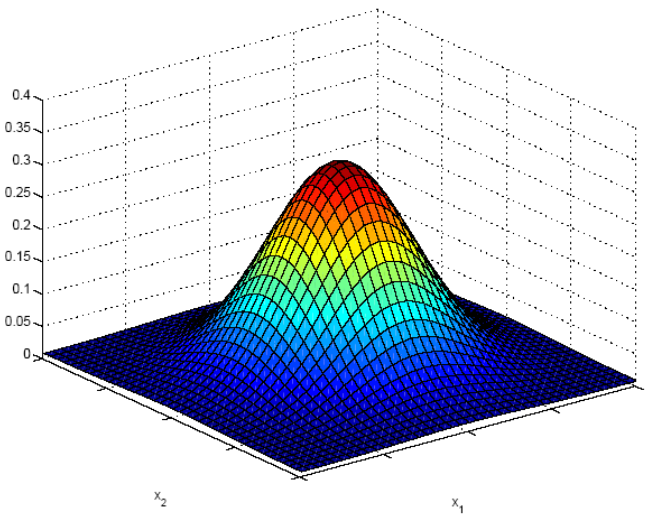
- Covariance is a matrix

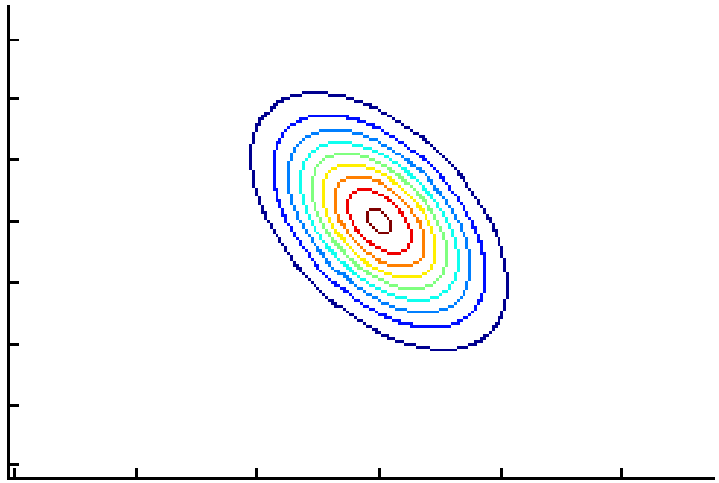
$$\boldsymbol{\Sigma} = \text{Cov}(\mathbf{x}) = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top] = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{D1} & \sigma_{D2} & \cdots & \sigma_D^2 \end{pmatrix}$$

Multivariate Gaussian Distribution

- Normally distributed variable $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ has distribution:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$





Key properties of covariances

Covariances are

1. symmetric, i.e., $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^\top$
2. positive semi-definite, i.e.,
   $\mathbf{v}^\top \boldsymbol{\Sigma} \mathbf{v} = \mathbf{v}^\top \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top]\mathbf{v} = \mathbb{E}[((\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{v})^2] \geq 0$ for all $\mathbf{v} \neq 0$
   - Symmetric matrix A is positive semidefinite if $\mathbf{v}^\top \mathbf{A}\mathbf{v} \geq 0$ for all non-zero $\mathbf{v}$. It is positive definite if $\mathbf{v}^\top \mathbf{A}\mathbf{v} > 0$ for all non-zero $\mathbf{v}$.
3. For a Gaussian distribution with density, $\boldsymbol{\Sigma}$ is positive definite.

Some other key properties:

3. $\boldsymbol{\Sigma} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] - \boldsymbol{\mu}\boldsymbol{\mu}^\top$ (generalizes $\text{Var}(x) = \mathbb{E}[x^2] - \mu^2$))
4. $\text{Cov}(\mathbf{A}\mathbf{x} + \mathbf{b}) = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top$ (generalizes $\text{Var}(ax + b) = a^2 \text{Var}(x)$)

Shifting and scaling a multivariate Gaussian

- Multivariate Gaussians are also shifted and scaled version of the standard multivariate normal distribution!
  - The standard multivariate normal has $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma} = \mathbf{I}$
- Multivariate analog of the shift is simple: it's a vector $\boldsymbol{\mu}$
- But what about the scale?
  - But in the multivariate case, the covariance $\boldsymbol{\Sigma}$ is a matrix! Does $\boldsymbol{\Sigma}^{\frac{1}{2}}$ exist, and can we scale by it?
- First, what does it mean to scale by a matrix?

Multivariate Scaling (Formal)

Recall from linear algebra (without proof): (*Spectral Theorem*). If $\mathbf{A} \in \mathbb{R}^{d \times d}$ is symmetric, then

1. $\mathbb{R}^D$ has an orthonormal basis consisting of the eigenvectors of $\mathbf{A}$.
2. There exists orthonormal matrix $\mathbf{Q}$ and diagonal matrix $\mathbf{\Lambda}$ such that $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. This is called the spectral decomposition of $\mathbf{A}$.
   - The columns of $\mathbf{Q}$ are (unit) eigenvectors of $\mathbf{A}$.
   - The diagonal entries of $\mathbf{\Lambda}$ are the eigenvalues of $\mathbf{A}$ in order corresponding to the eigenvector in $\mathbf{Q}$.

So, scaling $\mathbf{v}$ by a matrix $\mathbf{A}$ is $\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{v}$, which consists of rotating $\mathbf{v}$ by $\mathbf{Q}^T$, scaling by the eigenvalues, then rotating back by $\mathbf{Q}$. This can be thought of intuitively as stretching / compressing space along the eigenvectors of $\mathbf{A}$ by the corresponding eigenvalues.

Note: positive definite matrices have positive eigenvalues, and positive semidefinite matrices have non-negative eigenvalues.

Multivariate Scaling (Intuitive)

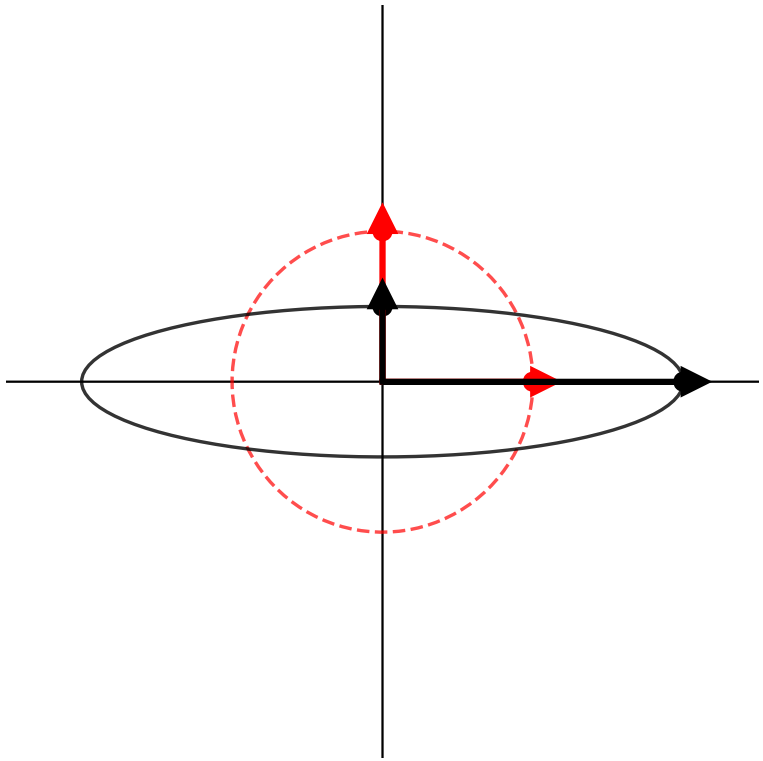
To "scale" by a positive definite matrix, you scale along each eigenvector by the corresponding eigenvalues:

Consider matrix (note it's symmetric!):

$$\begin{pmatrix} 2 & 0 \\ 0 & 0.5 \end{pmatrix}$$

Consider action on eigenvectors:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \perp \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



Multivariate Scaling (Intuitive)

(optional draw-on slide for intuition)

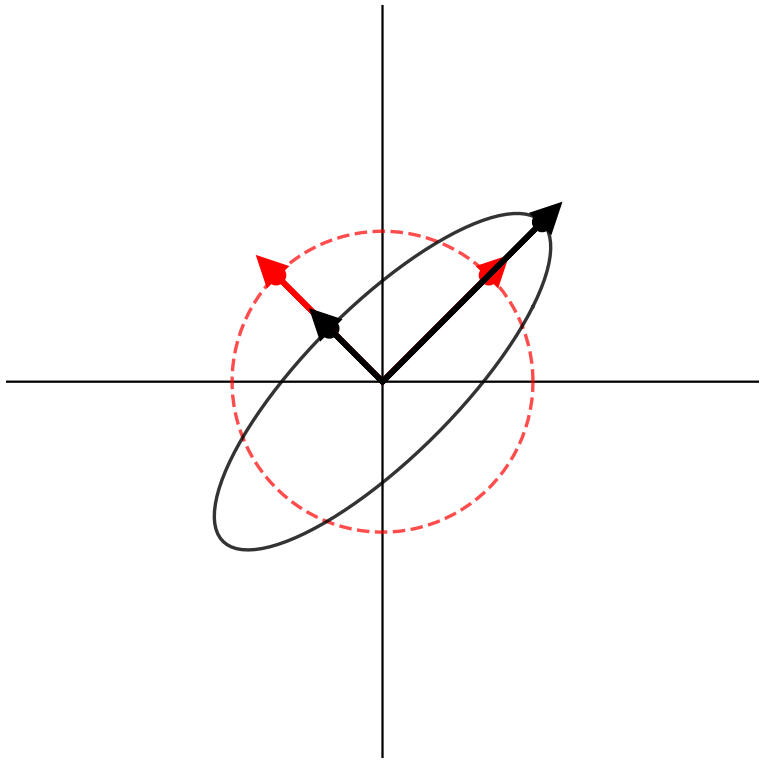
To "scale" by a positive definite matrix, you scale along each eigenvector by the corresponding eigenvalues:

Consider matrix (note it's symmetric!):

$$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

Consider action on eigenvectors:

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \perp \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$



Shifting and scaling a multivariate Gaussian

So here is the "scale" intuition:

- For positive definite $\mathbf{\Sigma}$, consider a spectral decomposition $\mathbf{Q\Lambda Q}^\top$.
- Define $\mathbf{\Sigma}^{1/2} = \mathbf{Q\Lambda}^{1/2}\mathbf{Q}^\top$, where $\mathbf{\Lambda}^{1/2}$ is a diagonal matrix whose diagonal entries are the square roots of the eigenvalues of $\mathbf{\Sigma}$ corresponding to the eigenvectors in $\mathbf{Q}$.
- Now, if $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ then $x = \boldsymbol{\mu} + \mathbf{\Sigma}^{1/2}\epsilon \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$. To see why:

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} + \mathbf{\Sigma}^{1/2}\mathbb{E}[\epsilon] = \boldsymbol{\mu}$$

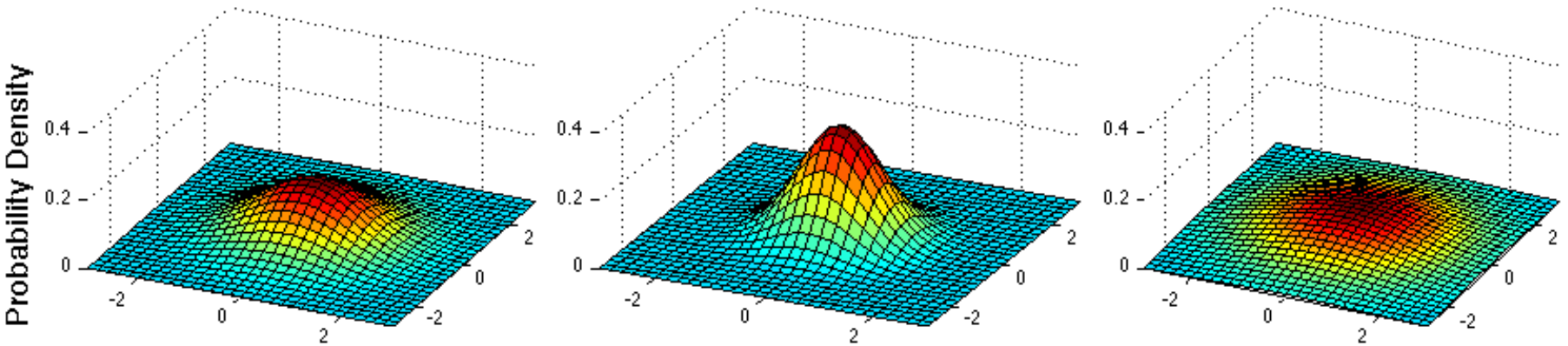
and

$$\begin{aligned}\text{Cov}(\mathbf{x}) = \mathbf{\Sigma}^{1/2}\,\text{Cov}(\epsilon)(\mathbf{\Sigma}^{1/2})^\top = \mathbf{\Sigma}^{1/2}(\mathbf{\Sigma}^{1/2})^\top &= \mathbf{Q\Lambda}^{1/2}\mathbf{Q}^\top\mathbf{Q\Lambda}^{1/2}\mathbf{Q}^\top \\ &= \mathbf{Q\Lambda Q}^\top = \mathbf{\Sigma}\end{aligned}$$

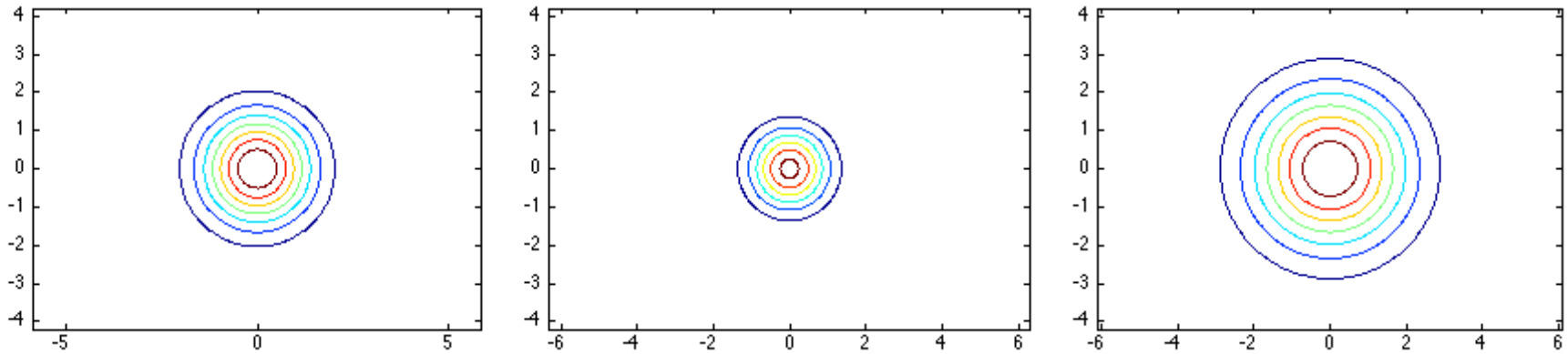
- What effect does this have on the density? Let's explore.

Bivariate Gaussian

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = 0.5 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = 2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



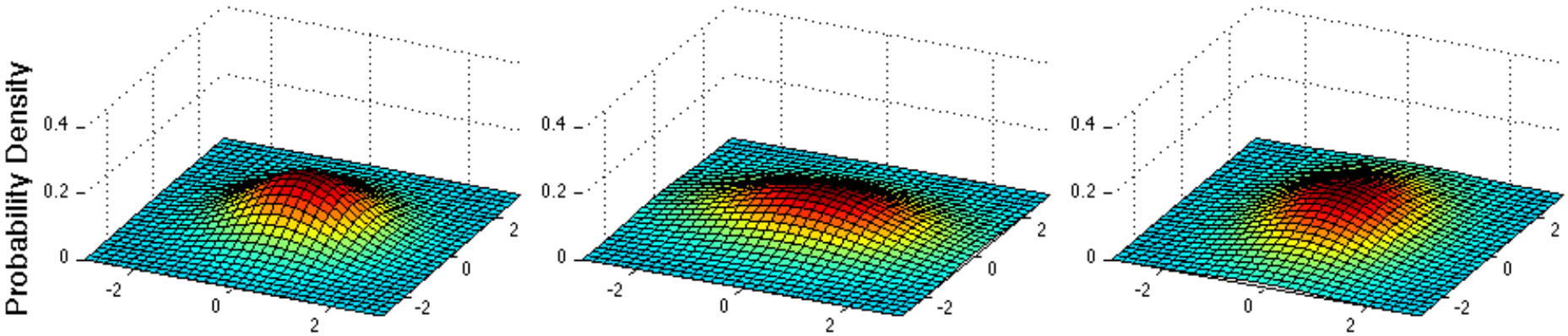
Probability density function



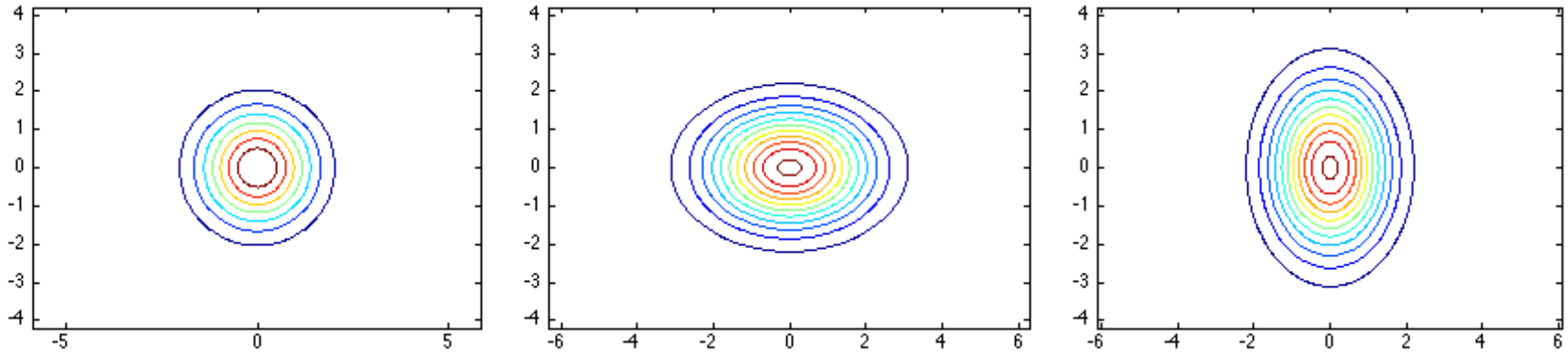
Contour plot of the pdf

Bivariate Gaussian

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$



Probability density function



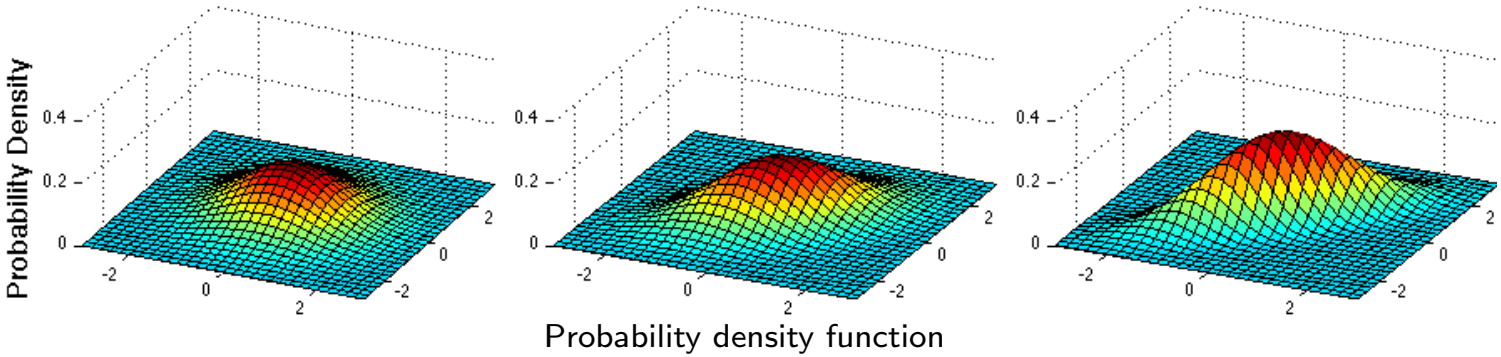
Contour plot of the pdf

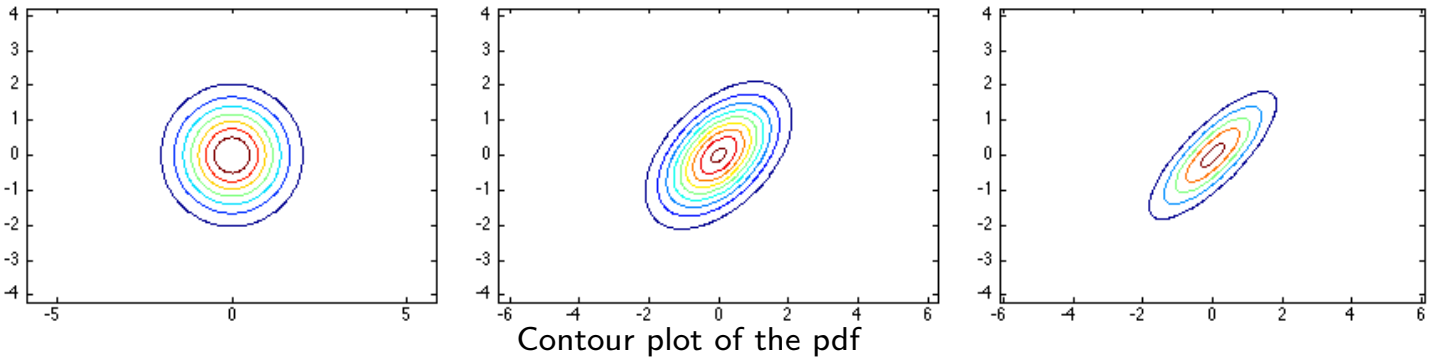
Bivariate Gaussian

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$$

$$= \mathbf{Q}_1 \begin{pmatrix} 1.5 & 0. \\ 0. & 0.5 \end{pmatrix} \mathbf{Q}_1^\top \qquad = \mathbf{Q}_2 \begin{pmatrix} 1.8 & 0. \\ 0. & 0.2 \end{pmatrix} \mathbf{Q}_2^\top$$

Test your intuition: Does $Q_1 = Q_2$?



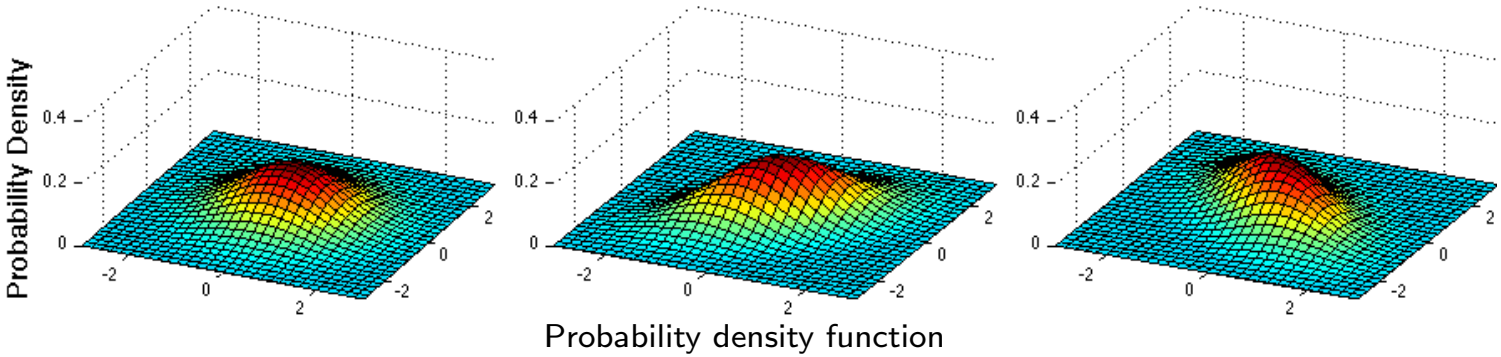


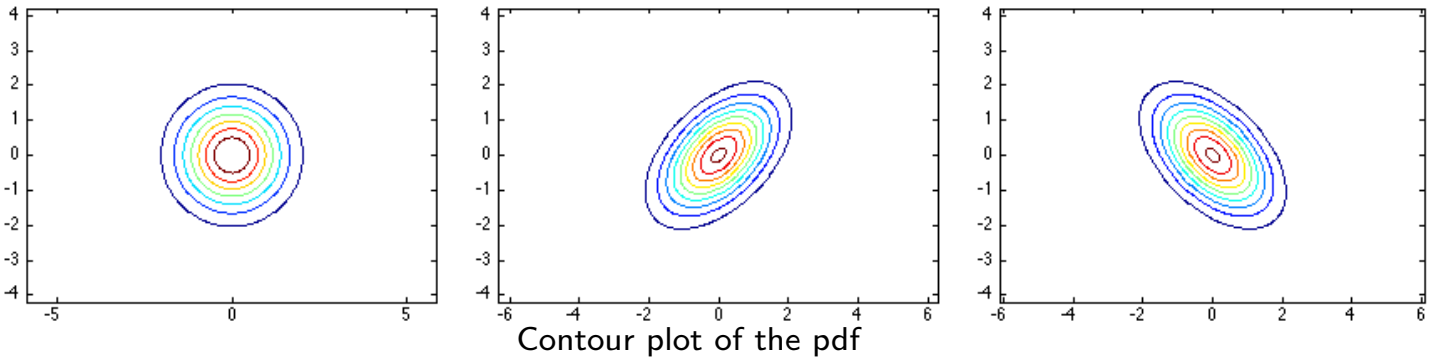
Bivariate Gaussian

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$$

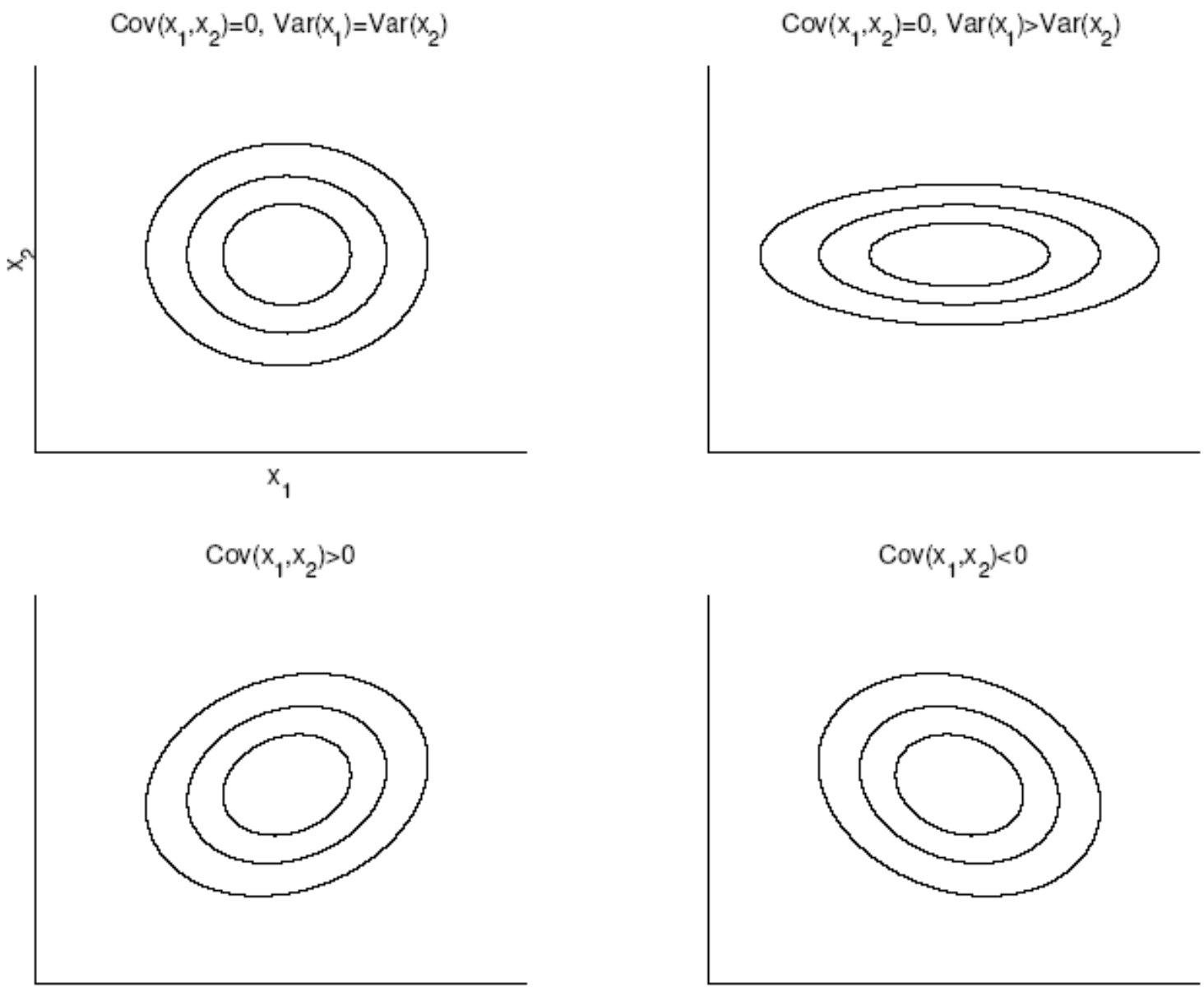
$$= \mathbf{Q}_1 \begin{pmatrix} 1.5 & 0. \\ 0. & 0.5 \end{pmatrix} \mathbf{Q}_1^\top \qquad = \mathbf{Q}_2 \begin{pmatrix} \lambda_1 & 0. \\ 0. & \lambda_2 \end{pmatrix} \mathbf{Q}_2^\top$$

Test your intuition: Does $Q_1 = Q_2$? What are λ_1 and λ_2?



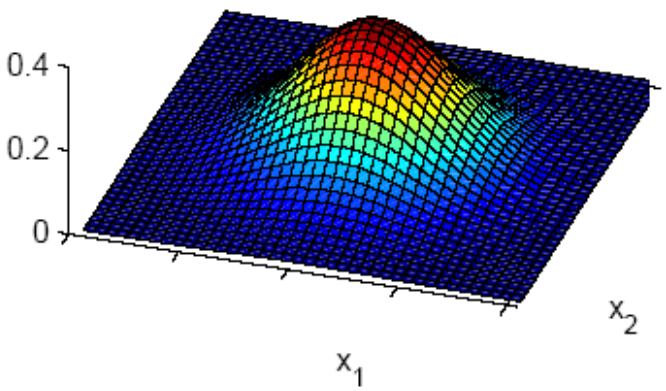


Bivariate Gaussian

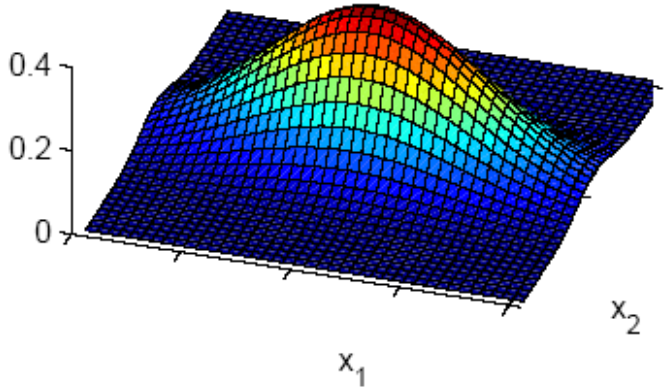


Bivariate Gaussian

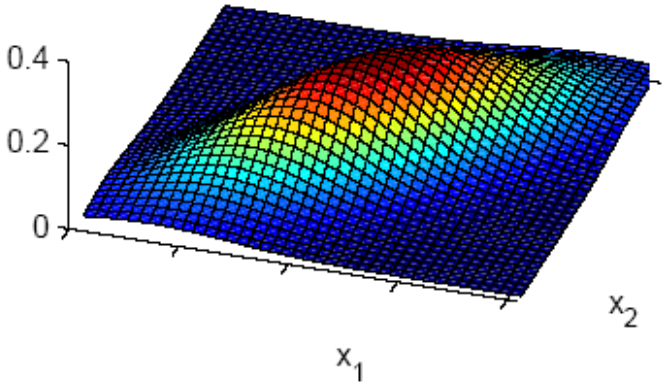
$\text{Cov}(x_1,x_2)=0$, $\text{Var}(x_1)=\text{Var}(x_2)$



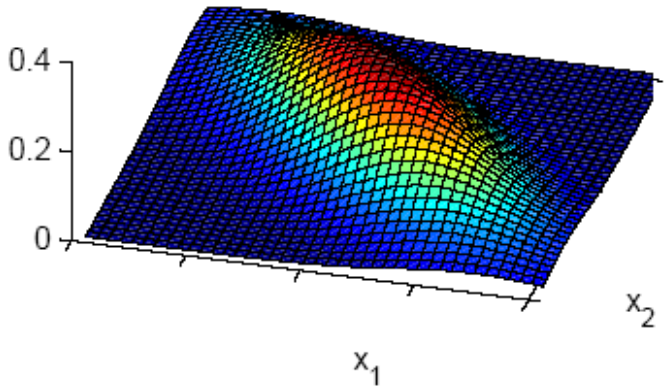
$\text{Cov}(x_1,x_2)=0$, $\text{Var}(x_1)>\text{Var}(x_2)$



$\text{Cov}(x_1,x_2)>0$



$\text{Cov}(x_1,x_2)<0$



PCA

- Next week: PCA.
- Dimensionality reduction: map data to a lower dimensional space
- PCA is finding linear structure in data.