COMP 432 Machine Learning

Probability Density Estimation

Computer Science & Software Engineering
Concordia University, Fall 2024

Random Variables

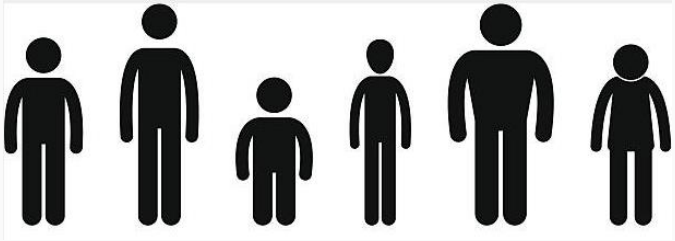
- In machine learning algorithms we have to deal with **random variables.**
- A random variable is a variable that can take **different values** (discrete or continuous) according to some probability.

Coin Toss

(discrete)

Roll a Dice

(discrete)

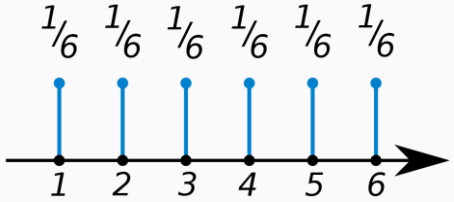


Human height distribution

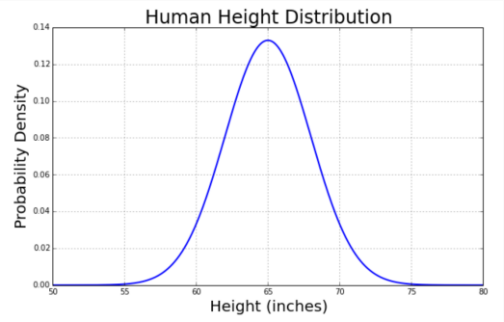
(continuous)

Random Variables

- Each random variable is associated with a **Probability Density Function** (PDF).
- The Probability Density Function (PDF) describes which outcomes are more likely and which ones are less.



PDF for the dice roll (discrete)



PDF for the human height *(continuous)*

Properties:

$$f_X(x) \geq 0 \text{ for all x}$$

$$\int_{-\infty}^{+\infty} f_X(x)dx = 1 \longrightarrow$$

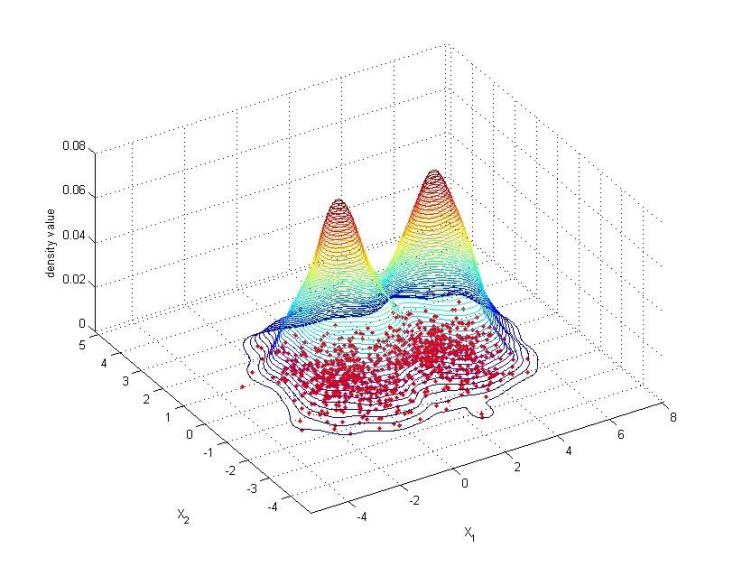
Note that $f_X(x)$ can be > 1 (the constraint is on its integral)

$$P(A \leq X \leq B) = \int_A^B f_X(x)dx$$

f_X is the pdf function for the (1D) random variable x

Random Variables

- In **machine learning**, random variables are everywhere.
- For instance, we can think of the input **features** are if they were drawn from a **probability density function** (data generation process).



- In most machine learning processes, this data generation process is **unknown.**
- However, we can try to estimate it from the available observations (features)
- This process is called **density estimation**.

Density Estimation

- **Goal**: given a dataset $D=\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_N\}$ we want to estimate $p(\mathbf{x})$.

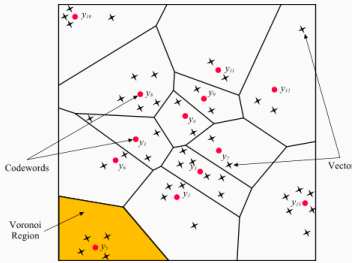
No labels, only inputs

At test/inference time, we can assign a **probability** to each **new sample**.

Why this can be helpful?

Unsupervised learning

- Anomaly (Outliers) detection → If the probability is low, we might have an outlier.
- Ranking → We can rank entries based on their probabilities.
- Vector quantization → Can be achieved by assigning shorter code to high $p(\mathbf{x})$.
- Clustering → We can identify groups of samples with high probabilities.



Density Estimation

- Beyond unsupervised learning, density estimation can be useful for **supervised learning** as well!

Prior Probability

Likelihood

$$k^* = \underset{k}{\operatorname{argmax}}\ P(y_k)p(x_1, ..., x_D \mid y_k)$$

This term can be simply estimated by counting:

$$P(y_k) \approx \frac{N_k}{N}$$

N_k → Training samples belong to class k.

N → Total number of training samples.

The likelihood can be estimated with a **density estimation technique**.

Notation:

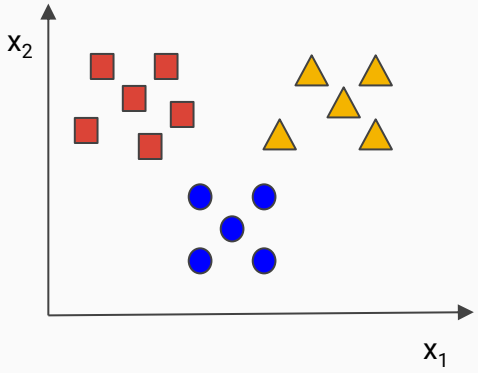
- P() for probabilities over **discrete variables**.
- p() for probabilities over **continuous variables** (probability density functions)

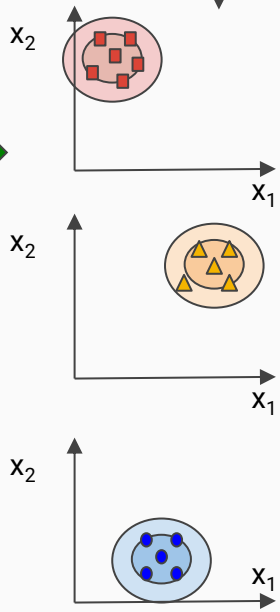
Density Estimation

Prior Probability

Likelihood

$$k^* = \underset{k}{\operatorname{argmax}}\ P(y_k)p(x_1, \ldots, x_D \mid y_k)$$





$p(\mathbf{x}|y_1)\ P(y_1) = 0.01$

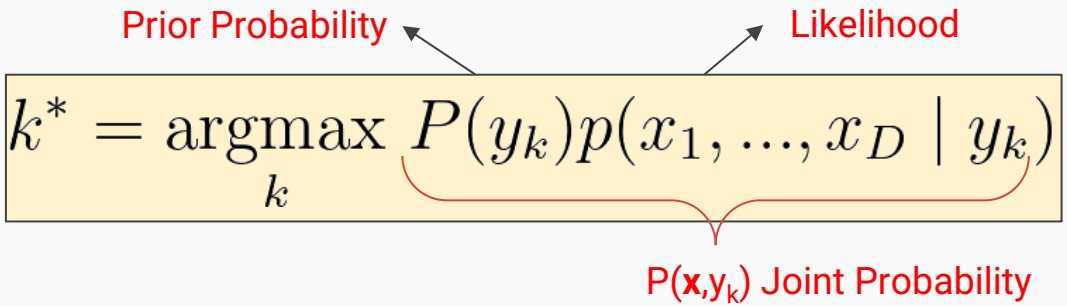
$p(\mathbf{x}|y_2)\ P(y_2) = 0.05$

$p(\mathbf{x}|y_3)\ P(y_3) = 0.02$

MAX

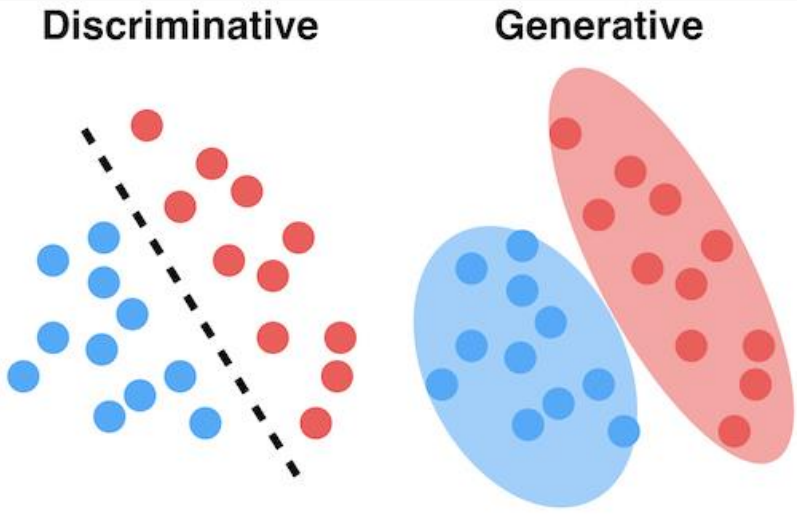
Class 2

Generative Models



- What is the main difference with the supervised approaches seen so far?
- So far, we have seen models that estimate the posterior **probability directly** (e.g., logistic regression, neural networks)**.**
- These models are called **discriminative models**.
- With the approach mentioned here, we estimate the **joint probability** (i.e., we estimate the **likelihood**, and then we combine it with the prior probability).
- These models are called **generative models**.

Generative Models



- **Discriminative models** are trained to **draw boundaries** in the input space.
- They estimate the **posterior probability** directly.
- **Generative models** try to model how data of each class is generated.

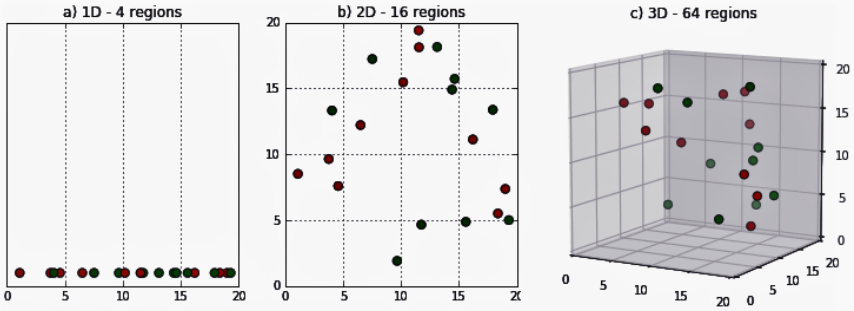
Decision trees? *K-nearest neighbor?* *Naive Bayes?*

- At training time, we estimate the “data generation” probability density function of each class instead of drawing boundaries.
- Unlike discriminative models, these models are also capable of **generating new data points** from the estimated data generation function

Curse of Dimensionality

- Density estimation is challenging, especially with **high-dimensional** inputs.

- We live in a 4D world (time+space) and is hard for us to imagine what happens in high-dimensional spaces like the ones where machine learning models live.

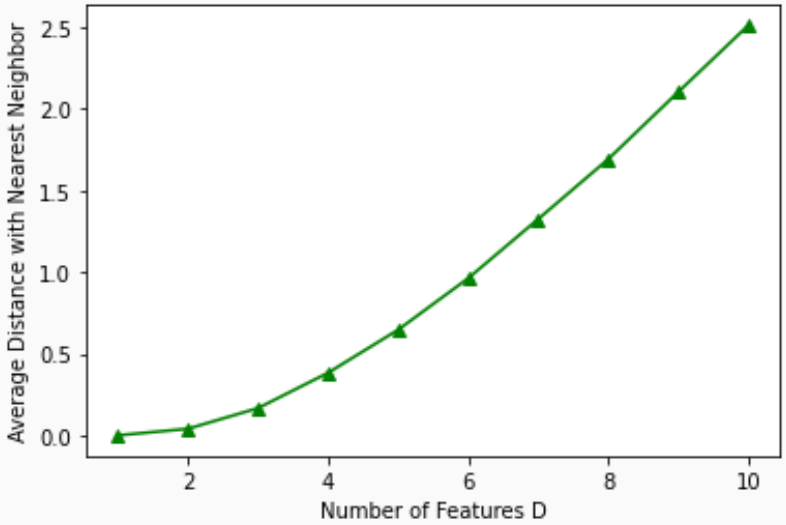


When we **increase the dimensionality** D of the input features, the data tends to be more and more **sparse**!

This problem is known as the **curse of dimensionality**.

Curse of Dimensionality

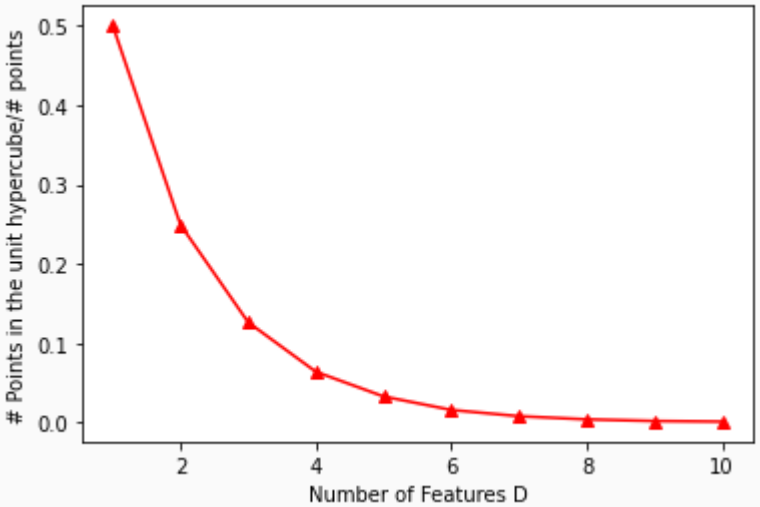
- Let's sample some points of different dimensionality and measure the average Euclidean distance with the nearest neighbor.



Points tend to be **far away** from each other in high-dimensional spaces.

Curse of Dimensionality

- Let's now plot the fraction of points that end up in the unitary hypercube over the total number of points.



High-dimensional spaces tend to be very "empty".

The number of points within a predefined volume **decreases exponentially**.

This means that we need **exponentially more data** to have the same density of points.

Curse of Dimensionality

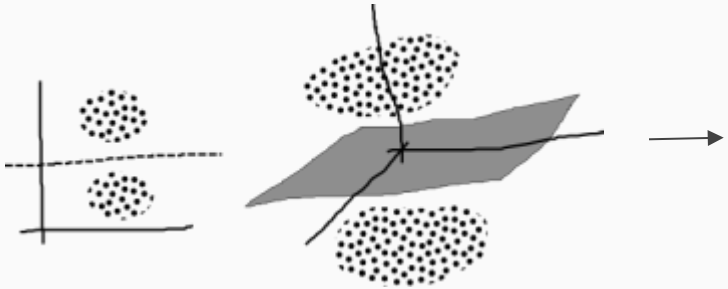
- This poses challenges not only to density estimation algorithms but in general to **any machine learning algorithm** operating on high-dimensional inputs.
- Nevertheless, not all the machine learning algorithms are equally sensitive to the curse of dimensionality issue:
- Methods that measure **distances** in the **input space** (e.g, *K-means*, *KNN*) are particularly sensitive to the curse of dimensionality.
- **Linear models** (e.g, linear regression, logistic regression) suffer from the curse of dimensionality as well. This can be mitigated by **regularization** (in particular L1 regularization because it forces sparsity by “dropping” some features).
- **Decision Trees** and **Random Forest** are less sensitive to this problem (especially when they test **one feature at a time**)

Curse of Dimensionality

- *What about support vector machines?*
- In principle, support vector machines should suffer a lot from the curse of dimensionality problem.
- We indeed transform our input features into a very **high-dimensional space.**
- In practice, it has been shown that support vector machines are not too much affected by the curse of dimensionality.

Curse of Dimensionality

- When we transform our input features, only a **small subset** of them are actually helpful to separate the classes.



Adding **irrelevant dimensions** does not matter at all, since the hyperplane would just lie parallel to the irrelevant dimensions.

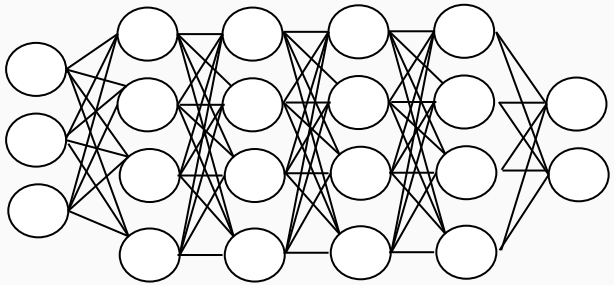
Only a subset of features has a strong impact on the hyperplane.

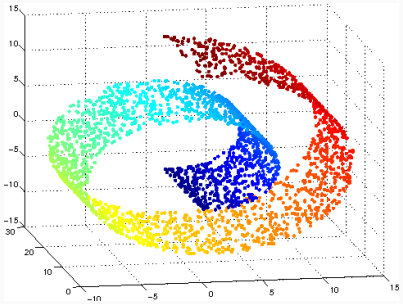
Features that matter lies in a **low-dimensional manifold**.

- Moreover, it can be shown that the **generalization** does **not depend on the dimensionality**.
- Finally, remember that SVMs use **L2 regularization**.

Curse of Dimensionality

- *What about neural networks?*
- Neural Networks do not suffer too much from the curse of dimensionality as well.

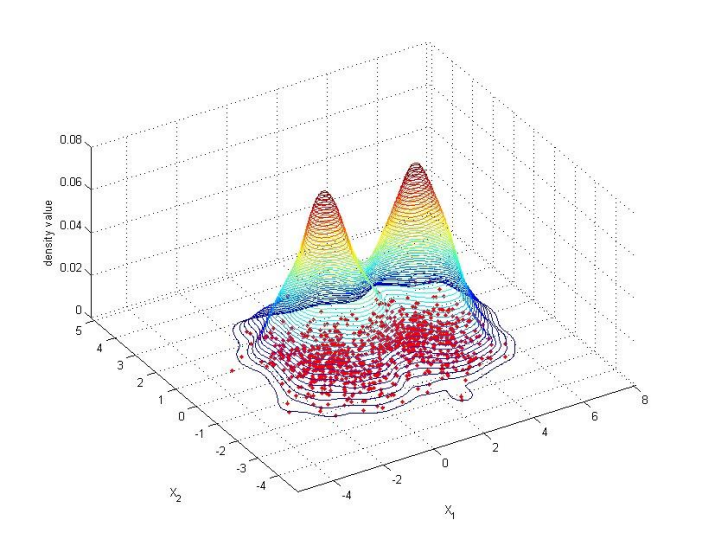




- Each neuron reacts only to a **small subset of features** (sparsity) in each layer.
- This is particularly evident for convolutional neural networks, which are based on **local connections.**
- Moreover, regularization techniques help improve the robustness against this problem.

Parametric Estimation

- **Goal**: given a dataset **X**, we want to estimate p(**x**).



- One way is to **parametrize the probability density function** with some parameters **θ**:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathbf{X} \mid \boldsymbol{\theta})$$

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_M]^T$$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \ldots & x_{1,D} \\ x_{2,1} & \ldots & x_{2,D} \\ \ldots & \ldots & \ldots \\ x_{N,1} & \ldots & x_{N,D} \end{bmatrix}$$

Maximum Likelihood Estimation (MLE)!

We look for the parameters **θ** that maximizes p(**X**|**θ**)

Parametric Estimation

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathbf{X} \mid \boldsymbol{\theta})$$

- This is another example of **parametric model**.
- In a parametric model, we have a **fixed number of adaptable parameters**, independent of the amount of data.

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_M]^T$$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,D} \\ x_{2,1} & \cdots & x_{2,D} \\ \cdots & \cdots & \cdots \\ x_{N,1} & \cdots & x_{N,D} \end{bmatrix}$$

- $\hat{\boldsymbol{\theta}}$ are the estimated parameters. The estimated parameters depend on the observations **X**
- If we change the observations **X,** the estimated parameters will change as well.
- We can thus interpret $\hat{\boldsymbol{\theta}}$ as a **random variable.**

Bias and Variance

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathbf{X} \mid \boldsymbol{\theta})$$

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_M]^T$$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,D} \\ x_{2,1} & \cdots & x_{2,D} \\ \cdots & \cdots & \cdots \\ x_{N,1} & \cdots & x_{N,D} \end{bmatrix}$$

- We want our estimated parameters to be as close as possible to the true ones.
- The **bias** of an estimator is defined as:

$$\text{bias}(\hat{\boldsymbol{\theta}}_{\mathbf{X}}) = \mathbb{E}_{\mathbf{X}}(\hat{\boldsymbol{\theta}}_{\mathbf{X}} - \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}}(\hat{\boldsymbol{\theta}}_{\mathbf{X}}) - \boldsymbol{\theta}$$

$\hat{\boldsymbol{\theta}}_{\mathbf{X}} \longrightarrow$ Parameters estimated using the dataset **X**

The bias can be computed in this way:

1. Sample from the data generation process **all the possible datasets X** composed of N data points.
2. For each dataset, compute the **difference** between the estimated and true parameters.
3. The bias is the **average** difference between the estimated and true parameters.

Bias and Variance

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathbf{X} \mid \boldsymbol{\theta})$$

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_M]^T$$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,D} \\ x_{2,1} & \cdots & x_{2,D} \\ \cdots & \cdots & \cdots \\ x_{N,1} & \cdots & x_{N,D} \end{bmatrix}$$

- An estimator is called **unbiased** if:

$$\operatorname{bias}(\hat{\boldsymbol{\theta}}_{\mathbf{X}}) = 0$$

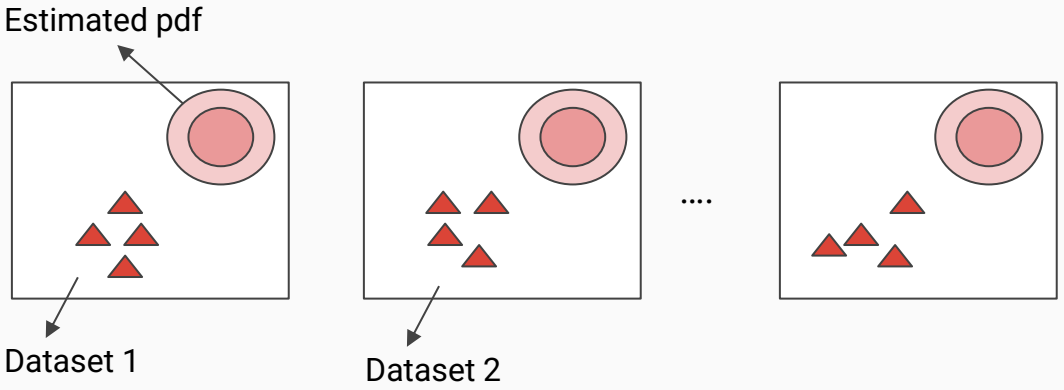
- An estimator is called **asymptotically unbiased** if:

$$\lim_{N \to \infty} \operatorname{bias}(\hat{\boldsymbol{\theta}}_{\mathbf{X}}) = 0$$

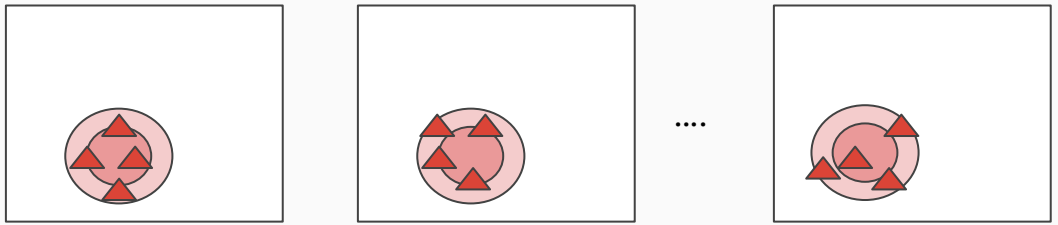
The bias gets zero if the dataset size N grows to infinity

Ideally, we want our estimator to be **unbiased** (or at least asymptotically unbiased).

Bias and Variance



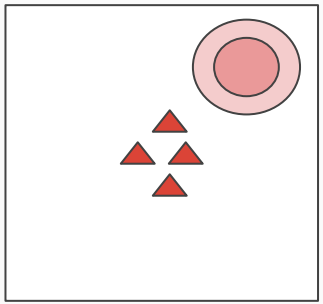
This is an estimator with **high bias** because, on average, the estimated parameters are far from true ones.



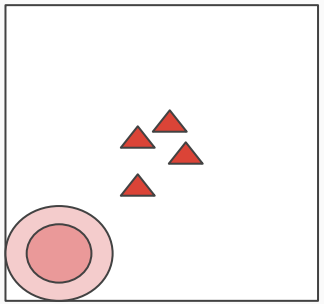
This is an estimator with **low bias** because, on average, the estimated parameters are close to the true ones.

Bias and Variance

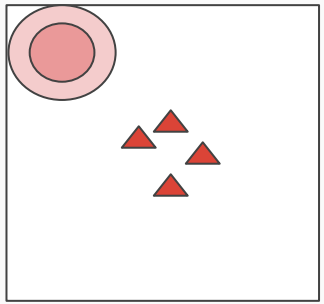
- Low bias is not all what we want.
- In some cases, estimators with low bias can be very bad!



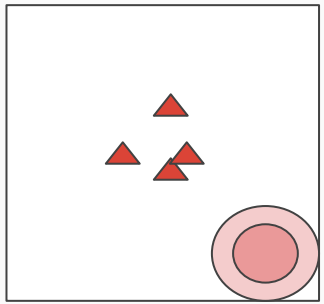
Dataset 1



Dataset 2



...

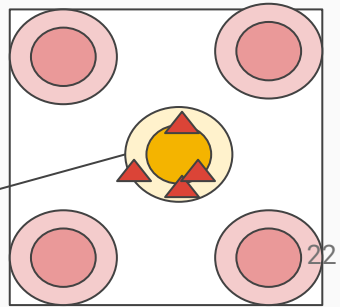


...

Do we have a low bias or a high bias here?

We have a **low bias** because the average parameters end up being close to the true ones

Model with the average parameters



Bias and Variance

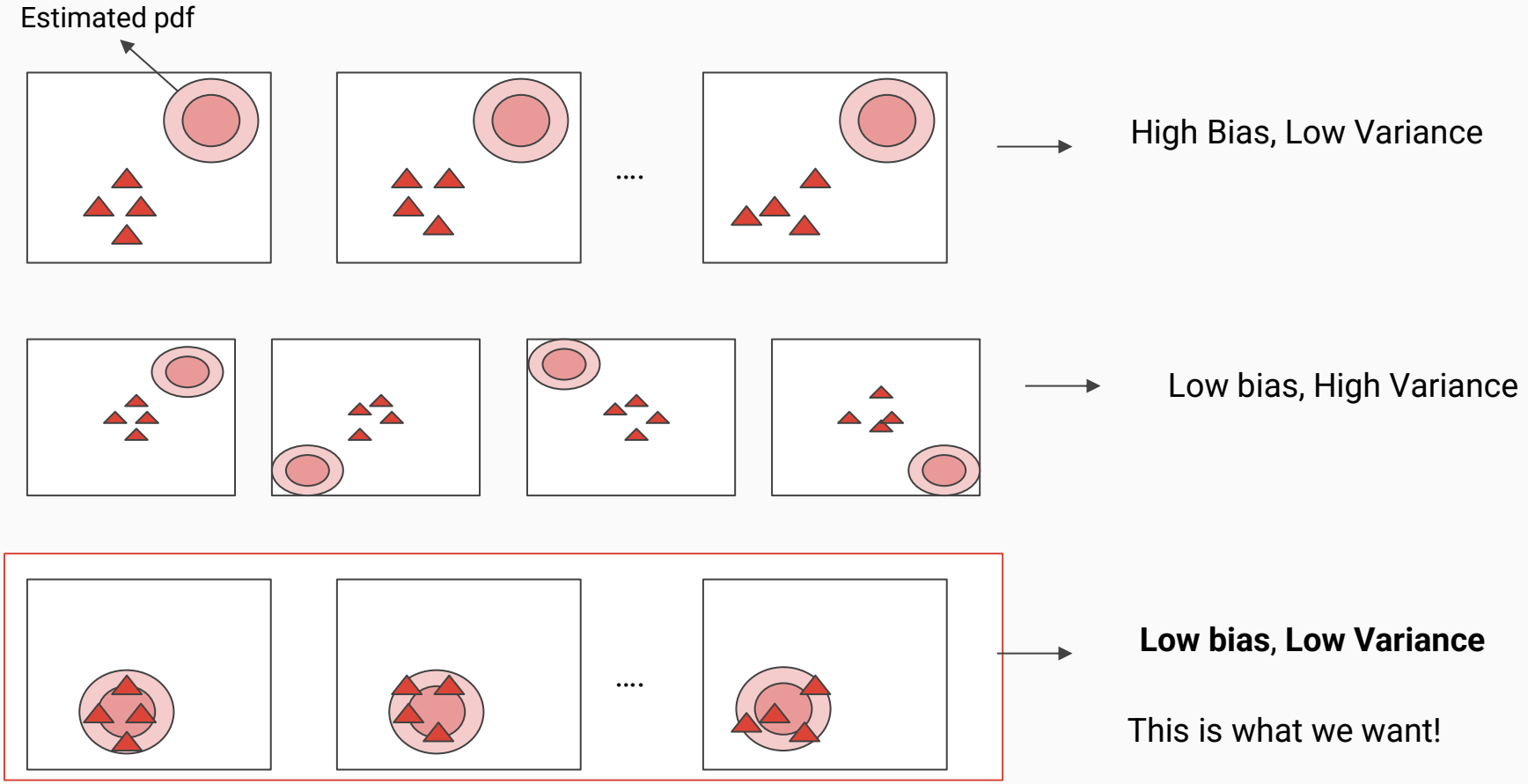
- We also want our estimator to have **low variance**.
- The **variance** of an estimator is defined as:

$$\mathrm{var}(\hat{\boldsymbol{\theta}}_{\mathbf{X}}) = \mathbb{E}_{\mathbf{X}}\left[(\hat{\boldsymbol{\theta}}_{\mathbf{X}} - \mathbb{E}_{\mathbf{X}}[\hat{\boldsymbol{\theta}}_{\mathbf{X}}])^2\right]$$

Mean value of the estimated parameters

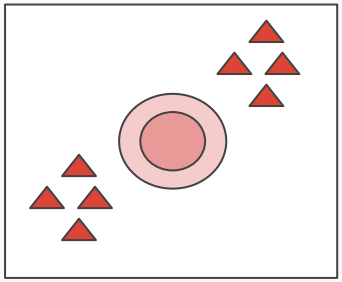
- The variance is low if the estimated parameters do not change too much when using different datasets X.
- Ideally, we want an estimator with **zero bias** and **zero variance**!

Bias and Variance

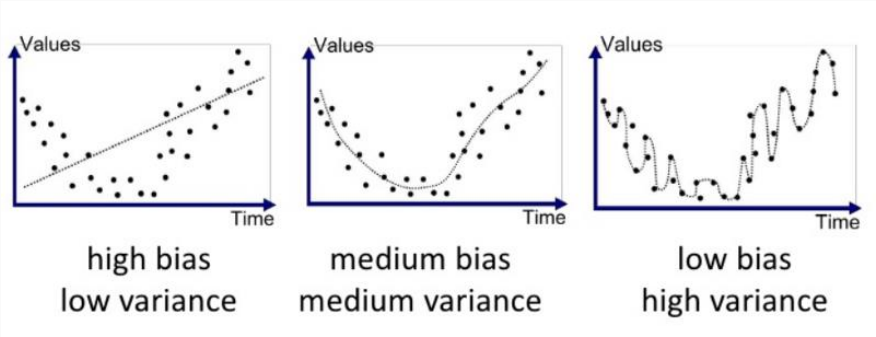


Bias and Variance

- A high bias can be caused by **wrong assumptions** about the data and models



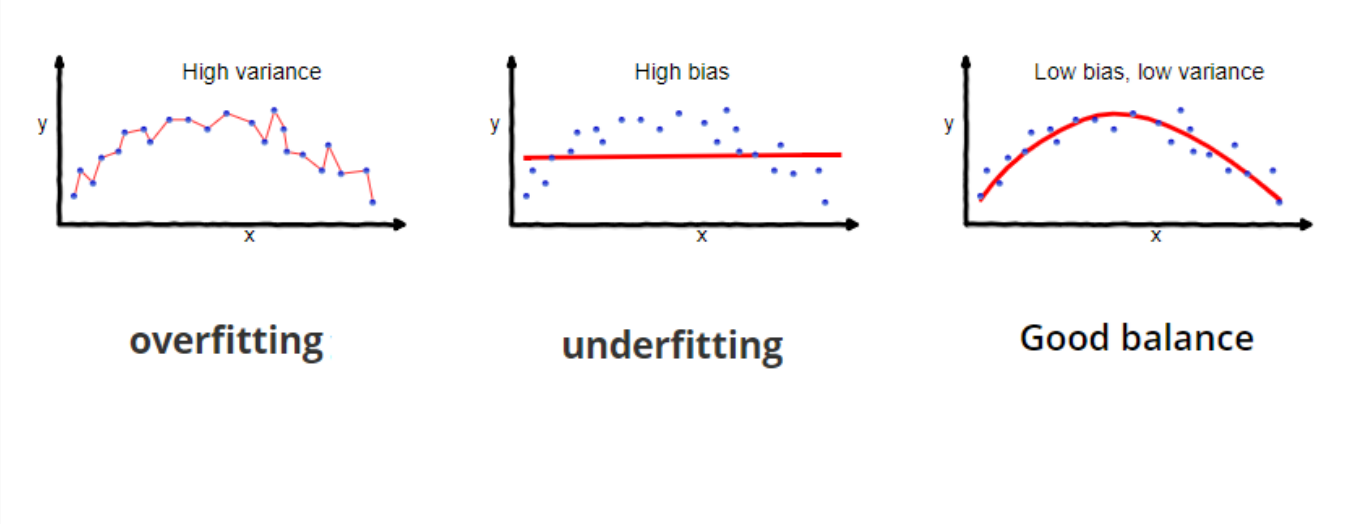
- We have a high bias if we use a Gaussian probability density function to describe observations that are not Gaussians.



- In the context of regression (supervised learning), high bias might happen if we assume the data to be linear while they aren't.

Bias and Variance

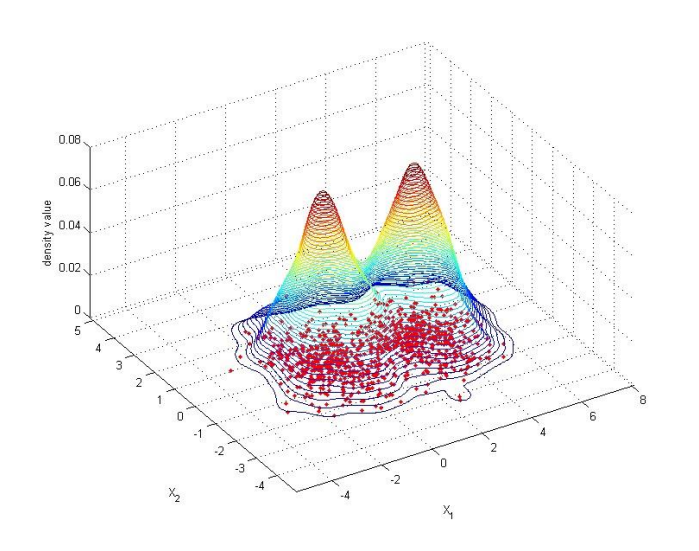
- A high variance might be caused by a model that **overfits** the training data.
- The concepts of **bias** and **variance** are tightly related to **underfitting** and **overfitting**.



There is a trade-off between **bias** and **variance** (just like there is a trade-off between underfitting and overfitting).

Parametric Estimation

- Let's now go back to our problem. We what to estimate the parameters such that:



$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ p(\mathbf{X} \mid \boldsymbol{\theta})$$

Maximum Likelihood Estimation (MLE)

- Since our observations (input features) are **independent** and **identically distributed** (i.i.d) we can write:

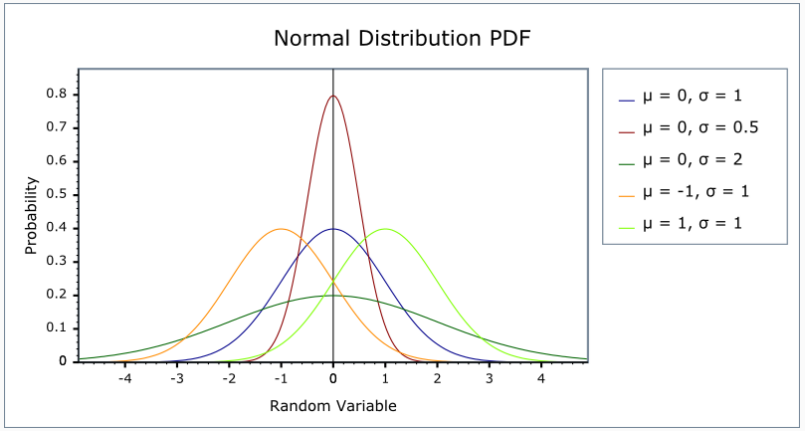
$$p(\mathbf{X} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\theta})$$

Parametric Estimation

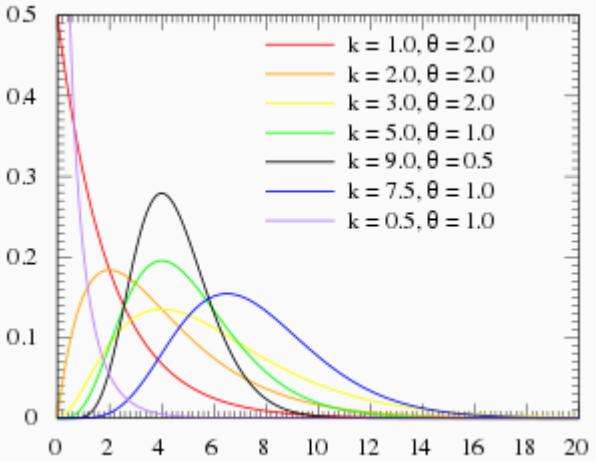
$$p(\mathbf{X} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\theta})$$

- Now, we have to choose a **statistical distribution** that can describe well our data.

- **Gaussian (normal) distribution**



- **Gamma distribution**

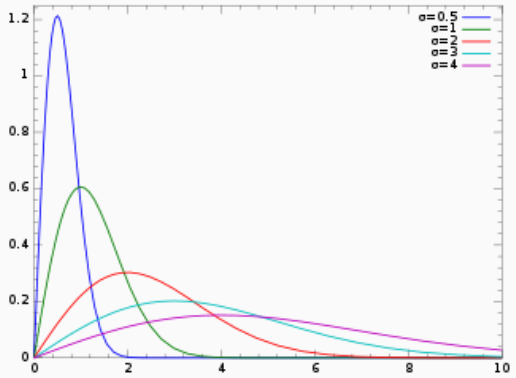


Parametric Estimation

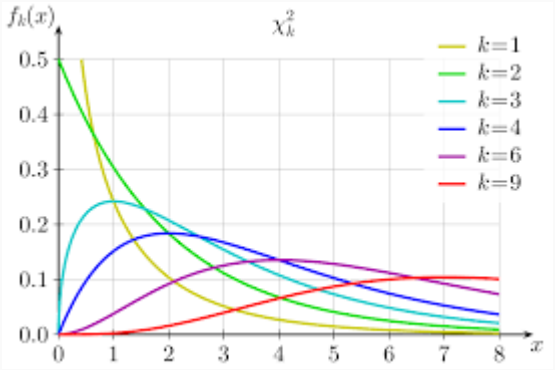
$$p(\mathbf{X} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\theta})$$

- Now, we have to choose a **statistical distribution** that can describe well our data.

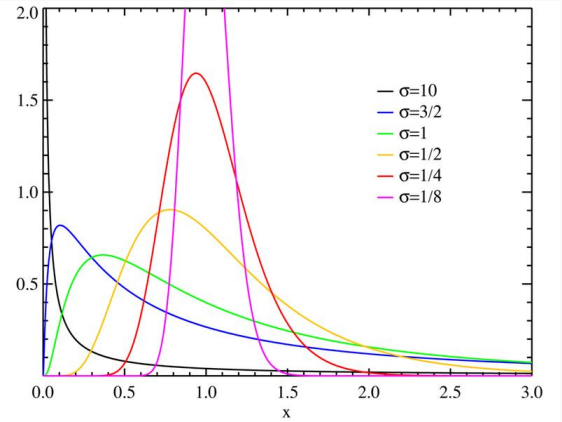
Rayleigh distribution



Chi-Square distribution



Log-Normal distribution



The choice of the distribution is an important **assumption**. If it is wrong our estimator will have a **high bias**!

.....

Multivariate Gaussian

$$p(\mathbf{X} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\theta})$$

- Let's assume our data can be described well by a **Gaussian distribution**.
- This is a popular choice that naturally comes in many applications.

- This is due to the **central limit theorem** which states:

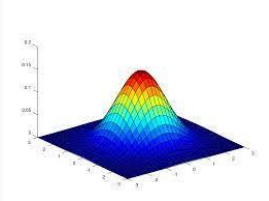
If the sum of many independent and iid variables has a finite variance, then it will be **approximately normally distributed** (i.e., Gaussian).

- Many real processes take origin from many independent causes and can thus be modeled with Gaussian distributions.

Multivariate Gaussian

- The multi-dimensional (multivariate) **Gaussian distribution** takes the following analytical expression:

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



Input features

$$\mathbf{x} = [x_1, x_2, \ldots, x_D]^T$$

Mean vector

$$\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_D]^T$$

This is the point where the gaussian is centered.

Covariance Matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma^2_{x_1} & cov(x_1, x_2) & \ldots & cov(x_1, x_D) \\ cov(x_2, x_1) & \sigma^2_{x_2} & \ldots & cov(x_2, x_D) \\ cov(x_D, x_1) & cov(x_D, x_2) & \ldots & \sigma^2_{x_D} \end{bmatrix}$$

The covariance matrix determines “how large” is the gaussian in each direction and where it is “oriented”.

The mean vector **μ** and the covariance matrix **Σ** are the parameters **θ** that we want to estimate.

Multivariate Gaussian

Covariance Matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma^2_{x_1} & cov(x_1, x_2) & \dots & cov(x_1, x_D) \\ cov(x_2, x_1) & \sigma^2_{x_2} & \dots & cov(x_2, x_D) \\ cov(x_D, x_1) & cov(x_D, x_2) & \dots & \sigma^2_{x_D} \end{bmatrix}$$

The covariance matrix collects all the covariances across the input features.

Properties of the covariance matrix:

- It is a **square matrix** with dimension (D x D)
- It is **symmetric**: $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$
- It is positive **semidefinite** (all eigenvalues are nonnegative)

$cov(x_i, x_j)$ quantifies how much x_i and x_j are **statistically related** (i.e, how much they are **correlated**)

$$cov(x_i, x_j) = \mathbb{E}[x_i - \overline{x_i}] \cdot \mathbb{E}[x_j - \overline{x_j}] = \mathbb{E}[x_i x_j] - \overline{x_i} \cdot \overline{x_j}$$

We try all the possible points $\mathbf{x} = [x_i, x_j]^T$

- $cov(x_i, x_j) > 0$ — If I increase x_i and on average x_j increases.
- $cov(x_i, x_j) = 0$ — If x_i and x_j are independent.
- $cov(x_i, x_j) < 0$ — If I increase x_i and on average x_j decreases.

Multivariate Gaussian

1D Gaussian Distribution

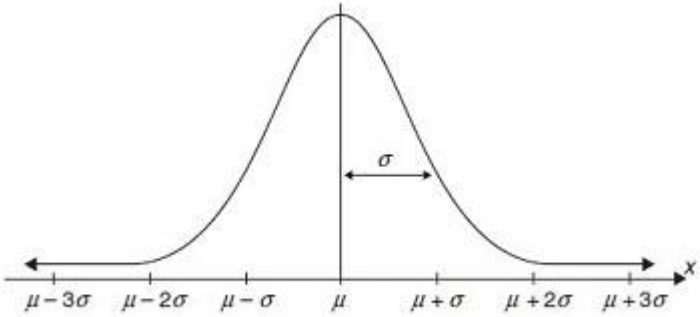
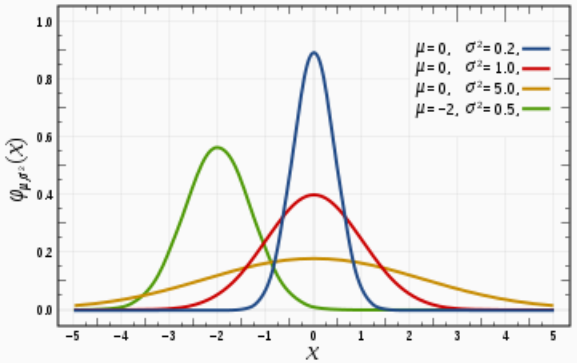


Figure 9.1 Normal curve with mean μ and standard deviation σ.



For 1D inputs, the expression for the Gaussian distribution becomes:

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right]$$

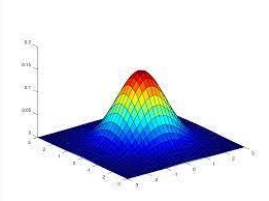
- μ is a scalar, while the covariance matrix Σ reduces to Σ the variance σ^2 .

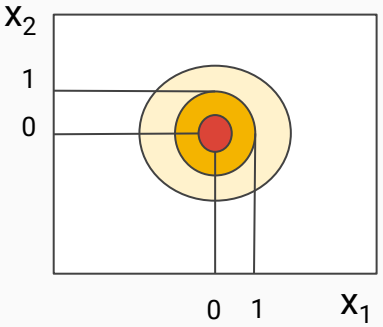
- If we change μ, we translate the center of the Gaussian.
- If we increase σ^2, we make the bell larger.

Multivariate Gaussian

2D Gaussian Distribution

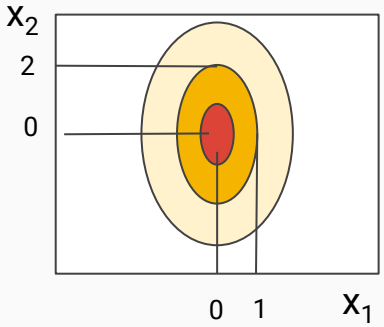
$$p(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$





$$\boldsymbol{\mu} = [0, 0]^T$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\boldsymbol{\mu} = [0, 0]^T$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

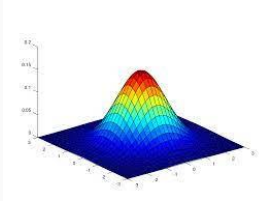
If the features are independent, the covariance matrix is **diagonal**.

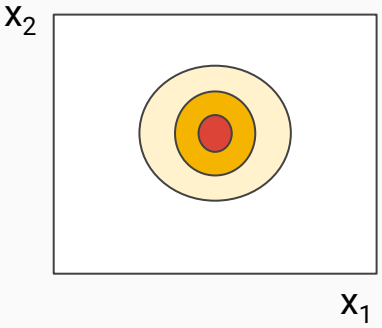
In general, the levels corresponding to the same probability (isolevels) are **ellipses**.

Multivariate Gaussian

2D Gaussian Distribution

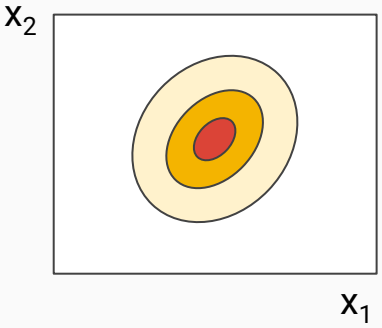
$$p(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$





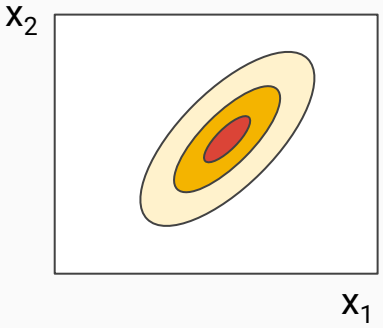
$$\boldsymbol{\mu} = [0,0]^T$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\boldsymbol{\mu} = [0,0]^T$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$



$$\boldsymbol{\mu} = [0,0]^T$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

If the features are correlated, the covariance matrix is **not diagonal**.

The **eigenvectors** define the main axis of the ellipse.

Multivariate Gaussian

D-dimensional Gaussian Distribution

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

The observations done for the 2D case can be extended to higher-dimensional Gaussian distributions:

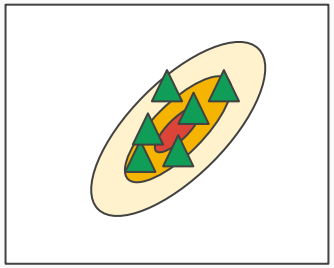
- The **eigenvalues** are all **positive**.
- The eigenvectors form an **orthogonal basis**.
- The isolevels of p(**x**) are hyperellipses whose axis directions are governed by the **eigenvectors**.
- The eigenvector corresponding to the largest eigenvalue defines the principal axis of the hyperellipses, while the others define the smaller axis.

Multivariate Gaussian

- Now, the problem is the following:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\theta})$$

Maximum Likelihood Estimation (MLE)



With a Gaussian distribution:

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

Let's try to solve this problem in a **closed-form way**!

Multivariate Gaussian

- For simplicity, let's solve the problem for the 1D Gaussian:

$$\hat{\mu}, \hat{\sigma} = \underset{\mu,\sigma}{\operatorname{argmax}} \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

Remember: monotonically increasing functions do not change the optimization outcome!

$$= \underset{\mu,\sigma}{\operatorname{argmax}} \sum_{i=1}^{N} -\ln(\sqrt{2\pi}) - \ln(\sigma) - \frac{(x_i - \mu)^2}{2\sigma^2}$$

(*Apply Logarithm*)

$$= \underset{\mu,\sigma}{\operatorname{argmax}} \sum_{i=1}^{N} -\ln(\sigma) - \frac{(x_i - \mu)^2}{2\sigma^2}$$

(*Remove Constant Term*)

$$= \underset{\mu,\sigma}{\operatorname{argmax}} -\sum_{i=1}^{N} \ln(\sigma) + \frac{(x_i - \mu)^2}{2\sigma^2}$$

(Algebraic manipulation)

Multivariate Gaussian

$$= \underset{\mu,\sigma}{\operatorname{argmax}} \; -\sum_{i=1}^{N} \ln(\sigma) + \frac{(x_i - \mu)^2}{2\sigma^2}$$

(Algebraic manipulation)

$$= \underset{\mu,\sigma}{\operatorname{argmin}} \; \underbrace{\sum_{i=1}^{N} \ln(\sigma) + \frac{(x_i - \mu)^2}{2\sigma^2}}_{\text{NLL}}$$

(Turning the problem to a minimization one)

- We can now compute the following derivatives and set them to zero:

$$\frac{\partial NLL}{\partial \mu} = 0 \qquad \frac{\partial NLL}{\partial \sigma} = 0$$

Multivariate Gaussian

$$\hat{\mu}, \hat{\sigma} = \underset{\mu,\sigma}{\operatorname{argmin}} \sum_{i=1}^{N} \ln(\sigma) + \frac{(x_i - \mu)^2}{2\sigma^2}$$

- Let's start with the **mean**:

$$\frac{\partial NLL}{\partial \mu} = \sum_{i=1}^{N} -\frac{2}{2\hat{\sigma}^2}(x_i - \hat{\mu}) = 0$$ (Derivative computation)

$$\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} \hat{\mu} = 0 \quad \Rightarrow \quad \hat{\mu} \sum_{i=1}^{N} = \sum_{i=1}^{N} x_i$$ (Algebraic manipulation)

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- This is the equation for the maximum likelihood estimation of the mean of a Gaussian.
- Not an unexpected result: this is the formula for computing the mean of the N data samples.

Multivariate Gaussian

- Let's now consider the **variance** term:

$$\hat{\mu}, \hat{\sigma} = \underset{\mu,\sigma}{\operatorname{argmin}} \sum_{i=1}^{N} \ln(\sigma) + \frac{(x_i - \mu)^2}{2\sigma^2}$$

$$\frac{\partial NLL}{\partial \sigma} = \sum_{i=1}^{N} \frac{1}{\hat{\sigma}} + \frac{(x_i - \hat{\mu})^2}{2} \frac{-2}{\hat{\sigma}^3} = 0$$ (Derivative computation)

$$\sum_{i=1}^{N} \frac{1}{\hat{\sigma}} - \frac{(x_i - \hat{\mu})^2}{\hat{\sigma}^3} = 0 \Rightarrow \sum_{i=1}^{N} \frac{\hat{\sigma}^2 - (x_i - \hat{\mu})^2}{\hat{\sigma}^3} = 0$$

(Algebraic manipulations)

$$\sum_{i=1}^{N} \hat{\sigma}^2 - (x_i - \hat{\mu})^2 = 0 \Rightarrow \sum_{i=1}^{N} \hat{\sigma}^2 = \sum_{i=1}^{N} (x_i - \hat{\mu})^2$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2$$

- This is the equation for the maximum likelihood estimation of the variance of a Gaussian.

This is the formula for computing the **variance** of the N data samples. Now you know where does it come from!

Multivariate Gaussian

$$\hat{\mu}, \hat{\sigma} = \underset{\mu,\sigma}{\operatorname{argmax}} \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Is *that unbiased?*

Yes!

$$\begin{aligned}
\text{bias}(\hat{\mu}_{\mathbf{x}}) &= \mathbb{E}_{\mathbf{x}}(\hat{\mu}_{\mathbf{x}}) - \mu \\
&= \mathbb{E}_{\mathbf{x}}\left[\frac{1}{N} \sum_{i=1}^{N} x_i \right] - \mu \\
&= \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathbf{x}}[x_i] - \mu \\
&= \frac{1}{N} \sum_{i=1}^{N} \mu - \mu \\
&= \mu - \mu = 0
\end{aligned}$$

Multivariate Gaussian

$$\hat{\mu}, \hat{\sigma} = \underset{\mu,\sigma}{\operatorname{argmax}} \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2$$

Is *that unbiased?*

$$\text{bias}(\hat{\sigma}^2_{\mathbf{x}}) = \mathbb{E}_{\mathbf{x}}(\hat{\sigma}^2_{\mathbf{x}}) - \sigma^2$$

It can be shown that:

$$\mathbb{E}_{\mathbf{x}}(\hat{\sigma}^2_{\mathbf{x}}) = \frac{N-1}{N}\sigma^2 \longrightarrow$$

The estimator is **biased** (but **asymptotically unbiased**)

However, we can make it unbiased by slightly changing the definition of the variance:

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \hat{\mu})^2$$

Multivariate Gaussian

- Next week, we will continue our discussion on density estimation and we will introduce models composed of multiple Gaussians (**Gaussian mixture models**).

- We will also mention **non-parametric methods** such as **Parzen windows**.

Lab Session

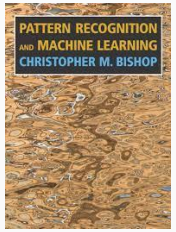
- During the weekly lab session, we will do:

  Tutorial on Clustering (implemented from scratch)

Clustering

11th Lab Assignment

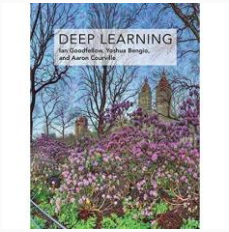
Additional Material



9.0.0 Mixture Models and EM

9.1.0 K-means Clustering

9.1.1 Image segmentation and compression

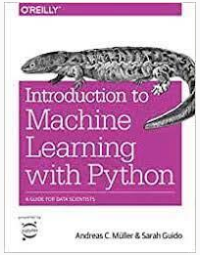


3.9: Common Probability Distributions

5.4: Estimators, Bias, and Variance

5.8.2: K-mean clustering

5.11.1: Curse of dimensionality



3.5 Clustering