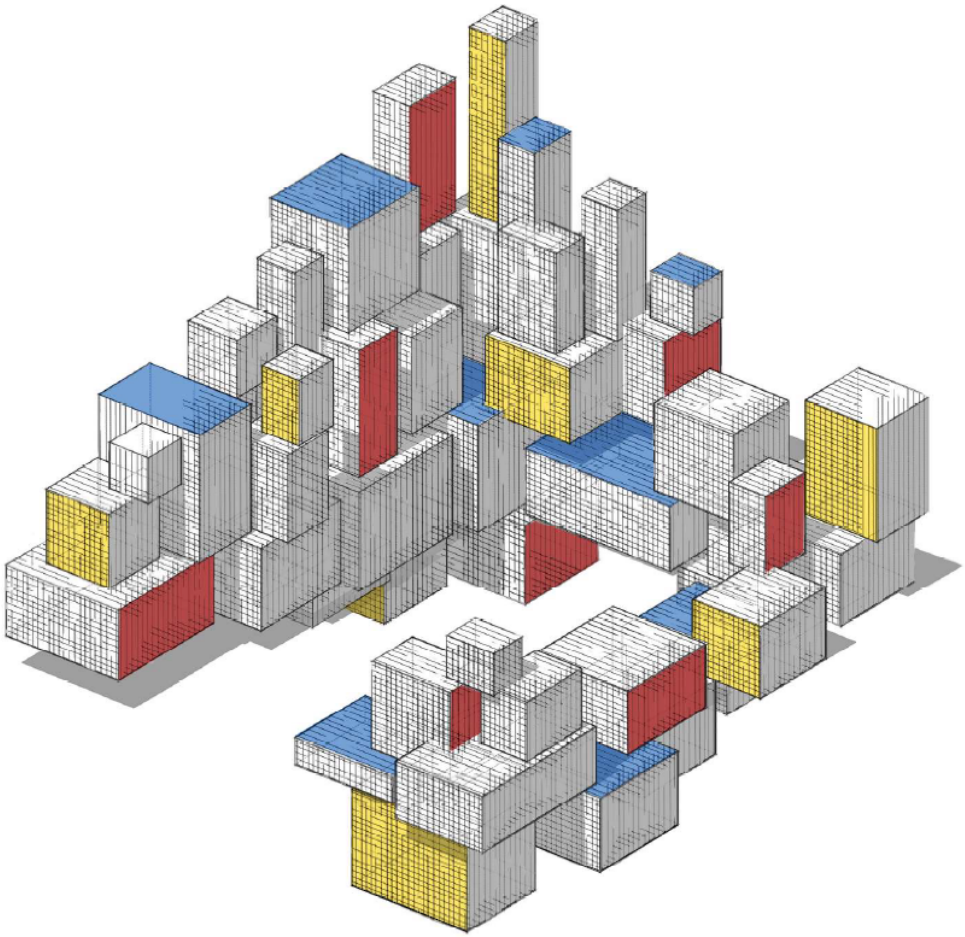
Modeling Data



Purpose

In this lecture we discuss:

- The role of modeling in data analysis
- Models with independent and identically distributed (iid) data
- The modeling dilemma
- Linear models
- Multivariate normal models

Modeling Data

The first step in any data analysis is to model the data in one form or another.

For example, in an unsupervised learning setting with data represented by a vector $\boldsymbol{x} = [x_1, \ldots, x_p]^\top$, a very general model is to assume that $\boldsymbol{x}$ is the outcome of a random vector $\boldsymbol{X} = [X_1, \ldots, X_p]^\top$ with some unknown pdf f.

The model can then be refined by assuming a specific form of f.

Iid Data

When given a sequence of such data vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, one of the simplest models is to assume that the corresponding random vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ are independent and identically distributed (iid).

We write

$$\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \overset{\text{iid}}{\sim} f \quad \text{or} \quad \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \overset{\text{iid}}{\sim} \mathsf{Dist},$$

to indicate that the random vectors form an iid sample from a sampling pdf f or sampling distribution Dist.

This model formalizes the notion that the knowledge about one variable does not provide extra information about another variable.

Independent Data Models

The main theoretical use of independent data models is that the joint density of the random vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ is the *product* of the marginal ones:

$$f_{\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = f(\boldsymbol{x}_1) \cdots f(\boldsymbol{x}_n).$$

Usually, the "model" $g(\boldsymbol{x})$ for $f(\boldsymbol{x})$ is specified up to a small number of parameters, i.e., $g(\boldsymbol{x} \mid \boldsymbol{\beta})$ for some parameter vector $\boldsymbol{\beta}$, corresponding to common probability distributions:

- $\mathcal{N}(\mu, \sigma^2)$
- $\mathsf{Bin}(n, p)$
- $\mathsf{Exp}(\lambda)$

Typically, the parameters are unknown and must be estimated from the data.

Exchangeable Data

If the order in which the data were collected (or their labeling) is not informative or relevant, then the joint pdf of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ satisfies the symmetry:

$$f_{\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = f_{\boldsymbol{X}_{\pi_1},\ldots,\boldsymbol{X}_{\pi_n}}(\boldsymbol{x}_{\pi_1}, \ldots, \boldsymbol{x}_{\pi_n})$$

for any permutation $\pi_1, \ldots, \pi_n$ of the integers $1, \ldots, n$.

We say that the infinite sequence $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$ is exchangeable if this permutational invariance holds for any finite subset of the sequence.

In Bayesian learning it is common to assume that the random vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ are a subset of an exchangeable sequence.

Modeling Dilemma

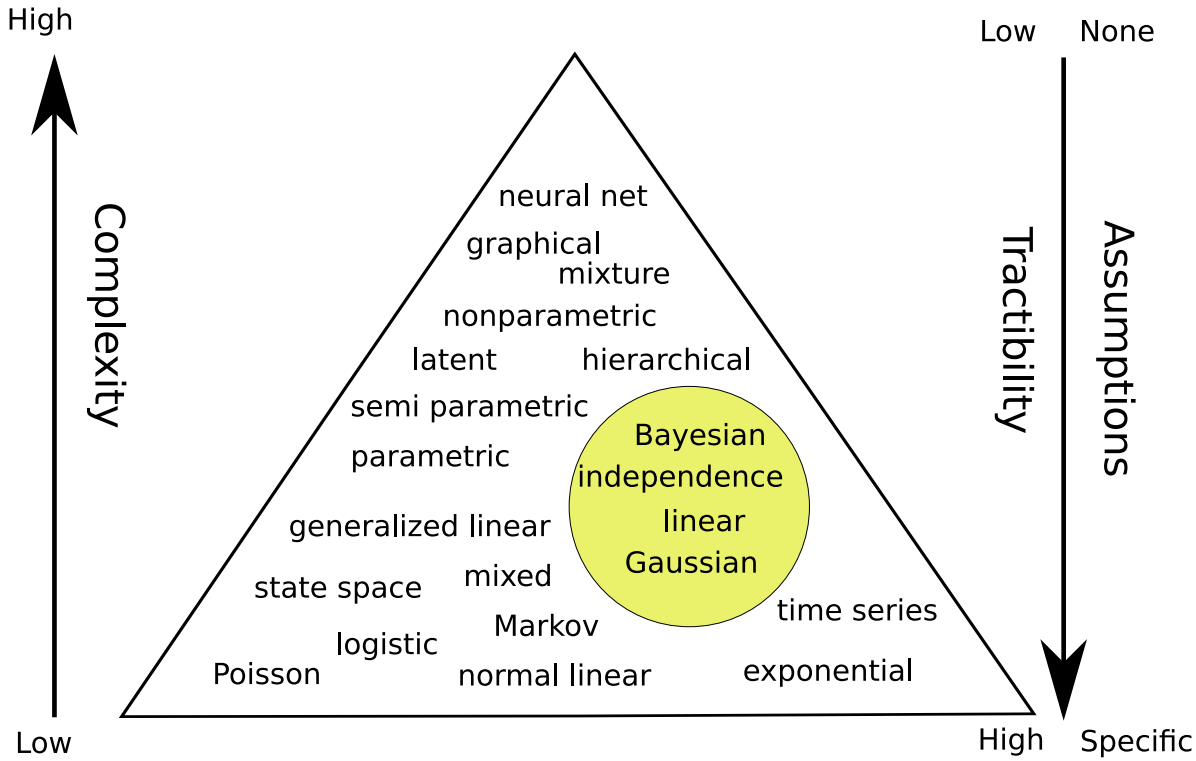


Figure: Complex models are more generally applicable, but may be difficult to analyze. Simple models may be highly tractable, but may not describe the data accurately. The triangular shape signifies that there are a great many specific models but not so many generic ones.

Tradeoff

There is a tradeoff between model tractability and applicability:

- Models that make few assumptions are more widely applicable, but at the same time may not be very mathematically tractable or provide insight into the nature of the data.
- Very specific models may be easy to handle and interpret, but may not match the data very well.

In the typical unsupervised setting we have a training set $\tau = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ that is viewed as the outcome of n iid random variables $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ from some unknown pdf f.

The objective is then to learn or estimate f from the finite training data.

Unsupervised Learning

Similar to the supervised learning framework, for unsupervised learning we reason as follows:

Specify a class of pdfs $\mathcal{G}_p := \{g(\cdot \mid \boldsymbol{\theta}), \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$, where $\boldsymbol{\theta}$ is a parameter in some subset $\boldsymbol{\Theta}$ of $\mathbb{R}^p$.

Seek the best g in $\mathcal{G}_p$ to minimize some risk. Note that $\mathcal{G}_p$ may not necessarily contain the true f even for very large p.

> We stress that our notation $g(\boldsymbol{x})$ has a different meaning in the supervised and unsupervised case. In the supervised case, g is interpreted as a prediction function for a response y; in the unsupervised setting, g is an approximation of a density f.

Loss Function for Unsupervised Learning

For each $\boldsymbol{x}$ we measure the discrepancy between the true model $f(\boldsymbol{x})$ and the hypothesized model $g(\boldsymbol{x} \mid \boldsymbol{\theta})$ using the loss function

$$\operatorname{Loss}(f(\boldsymbol{x}), g(\boldsymbol{x} \mid \boldsymbol{\theta})) = \ln \frac{f(\boldsymbol{x})}{g(\boldsymbol{x} \mid \boldsymbol{\theta})} = \ln f(\boldsymbol{x}) - \ln g(\boldsymbol{x} \mid \boldsymbol{\theta}).$$

The expected value of this loss (that is, the risk) is thus

$$\ell(g) = \mathbb{E} \ln \frac{f(\boldsymbol{X})}{g(\boldsymbol{X} \mid \boldsymbol{\theta})} = \int f(\boldsymbol{x}) \ln \frac{f(\boldsymbol{x})}{g(\boldsymbol{x} \mid \boldsymbol{\theta})} \, \mathrm{d}\boldsymbol{x}.$$

This integral defines the Kullback–Leibler (KL) divergence between f and $g(\cdot \mid \boldsymbol{\theta})$.

Note that the KL divergence is not symmetric in f and $g(\cdot \mid \boldsymbol{\theta})$. Moreover, it is always $\geqslant 0$ and equal to 0 when $f = g(\cdot \mid \boldsymbol{\theta})$.

Training Loss for Unsupervised Learning

Define $g^{\mathcal{G}_p}$ as the global minimizer of the risk in the class $\mathcal{G}_p$; that is, $g^{\mathcal{G}_p} = \operatorname{argmin}_{g \in \mathcal{G}_p} \ell(g)$. Let

$$\begin{aligned}\boldsymbol{\theta}^* &:= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}\,\mathrm{Loss}(f(\boldsymbol{X}), g(\boldsymbol{X} \mid \boldsymbol{\theta})) = \operatorname*{argmin}_{\boldsymbol{\theta}} \int \big(\ln f(\boldsymbol{x}) - \ln g(\boldsymbol{x} \mid \boldsymbol{\theta})\big) f(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \int f(\boldsymbol{x}) \ln g(\boldsymbol{x} \mid \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{x} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E} \ln g(\boldsymbol{X} \mid \boldsymbol{\theta}),\end{aligned}$$

then $g^{\mathcal{G}_p} = g(\cdot \mid \boldsymbol{\theta}^*)$ and learning $g^{\mathcal{G}_p}$ is equivalent to learning (or estimating) $\boldsymbol{\theta}^*$. To learn $\boldsymbol{\theta}^*$ from a training set $\tau = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ we minimize the training loss,

$$\frac{1}{n}\sum_{i=1}^{n} \mathrm{Loss}(f(\boldsymbol{x}_i), g(\boldsymbol{x}_i \mid \boldsymbol{\theta})) = -\frac{1}{n}\sum_{i=1}^{n} \ln g(\boldsymbol{x}_i \mid \boldsymbol{\theta}) + \frac{1}{n}\sum_{i=1}^{n} \ln f(\boldsymbol{x}_i),$$

giving:

$$\widehat{\boldsymbol{\theta}}_n := \operatorname*{argmax}_{\boldsymbol{\theta}} \frac{1}{n}\sum_{i=1}^{n} \ln g(\boldsymbol{x}_i \mid \boldsymbol{\theta}).$$

Maximum Likelihood Estimate

As the logarithm is an increasing function, this is equivalent to

$$\widehat{\boldsymbol{\theta}}_n := \operatorname*{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^{n} g(\boldsymbol{x}_i \mid \boldsymbol{\theta}),$$

where $\prod_{i=1}^{n} g(\boldsymbol{x}_i \mid \boldsymbol{\theta})$ is the likelihood of the data; that is, the joint density of the $\{\boldsymbol{X}_i\}$ evaluated at the points $\{\boldsymbol{x}_i\}$.

This is the classical maximum likelihood estimate of $\boldsymbol{\theta}^*$.

When the risk $\ell(g(\cdot \mid \boldsymbol{\theta}))$ is convex in $\boldsymbol{\theta}$ over a convex set $\boldsymbol{\Theta}$, we can find the maximum likelihood estimator by setting the gradient of the training loss to zero; that is, we solve

$$-\frac{1}{n} \sum_{i=1}^{n} \boldsymbol{S}(\boldsymbol{x}_i \mid \boldsymbol{\theta}) = \mathbf{0},$$

where $\boldsymbol{S}(\boldsymbol{x} \mid \boldsymbol{\theta}) := \frac{\partial \ln g(\boldsymbol{x} \mid \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ is the gradient of $\ln g(\boldsymbol{x} \mid \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ and is often called the score.

Example: Exponential Model

Training data: $\tau_n = \{x_1, \ldots, x_n\}$.
Model: $X_1, \ldots, X_n \sim_{\text{iid}} f(x)$.
Approximating class $\mathcal{G} := \{g : g(x \mid \theta) = \theta \exp(-x\,\theta), x > 0, \theta > 0\}$.

The likelihood of the data is

$$\prod_{i=1}^{n} g(x_i \mid \theta) = \prod_{i=1}^{n} \theta \exp(-\theta x_i) = \exp(-\theta\, n\, \overline{x}_n + n \ln \theta),$$

with score is $S(x \mid \theta) = -x + \theta^{-1}$. Maximizing the likelihood with respect to θ is the same as solving $-\sum_{i=1}^{n} S(x_i \mid \theta)/n = \overline{x}_n - \theta^{-1} = 0$.

Thus, the minimal training loss is attained at the maximum likelihood estimate $\widehat{\theta}_n = 1/\overline{x}_n$.

Model for Supervised Learning

In a supervised setting, where the data is represented by a vector $\boldsymbol{x}$ of explanatory variables and a response y, the general model is that $(\boldsymbol{x}, y)$ is an outcome of $(\boldsymbol{X}, Y) \sim f$ for some unknown f.

For a training sequence $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ the default model assumption is that $(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n) \sim_{\text{iid}} f$.

As explained earlier, the analysis primarily involves the conditional pdf $f(y \mid \boldsymbol{x})$ and in particular (when using the squared-error loss) the conditional expectation $g^*(\boldsymbol{x}) = \mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}]$.

Thus, conditional on $\boldsymbol{X} = \boldsymbol{x}$, we may write $Y = g^*(\boldsymbol{x}) + \varepsilon(\boldsymbol{x})$.

Linear Model

This leads to the simplest and most important model for supervised learning, where we choose a linear class $\mathcal{G}$ of prediction functions and assume that it is rich enough to contain the true g^*.

If we further assume that, conditional on $\boldsymbol{X} = \boldsymbol{x}$, the error term ε does not depend on $\boldsymbol{x}$, that is, $\mathbb{E}\,\varepsilon = 0$ and $\mathbb{V}\mathrm{ar}\,\varepsilon = \sigma^2$, then we obtain the following model.

Definition: Linear Model

In a linear model the response Y depends on a p-dimensional explanatory variable $\boldsymbol{x} = [x_1, \ldots, x_p]^\top$ via the linear relationship

$$Y = \boldsymbol{x}^\top \boldsymbol{\beta} + \varepsilon,$$

where $\mathbb{E}\,\varepsilon = 0$ and $\mathbb{V}\mathrm{ar}\,\varepsilon = \sigma^2$.

Model Matrix

Note that we formulated tthe linear model for a single pair $(\boldsymbol{x}, Y)$.

The model for the training set $\{(\boldsymbol{x}_i, Y_i)\}$ is simply that each $Y_i = \boldsymbol{x}_i^\top + \varepsilon_i$ and that the $\{\varepsilon_i\}$ are independent.

Gathering all responses in the vector $\boldsymbol{Y} = [Y_1, \ldots, Y_n]^\top$, we can write:

$$\boldsymbol{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where $\boldsymbol{\varepsilon} = [\varepsilon_1, \ldots, \varepsilon_n]^\top$ is a vector of iid copies of ε and $\mathbf{X}$ is the so-called model matrix, with rows $\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top$.

Linear models are fundamental building blocks of statistical learning algorithms.

Example: Polynomial Regression (cont.)

For our running example, we used the original explanatory variables $\{u_i\}$ to formulate the new explanatory (feature) variables $\boldsymbol{x}_i = [1, u_i, u_i^2, \ldots, u_i^{p-1}]^\top$, $i = 1, \ldots, n$.

We then considered linear prediction functions of these new features, to yield the linear model

$$\boldsymbol{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

with

$$\mathbf{X} = \begin{bmatrix} 1 & u_1 & u_1^2 & \cdots & u_1^{p-1} \\ 1 & u_2 & u_2^2 & \cdots & u_2^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_n & u_n^2 & \cdots & u_n^{p-1} \end{bmatrix}.$$

About Modeling

We would like to emphasize a number of points about modeling.

- *Any* model for data is likely to be *wrong*. For example, real data (as opposed to computer-generated data) are often assumed to come from a normal distribution, which is never exactly true. However, an important advantage of using a normal distribution is that it has many nice mathematical properties.
- Most data models depend on a number of unknown parameters, which need to be estimated from the observed data.
- Any model for real-life data needs to be *checked* for suitability. An important criterion is that data simulated from the model should resemble the observed data, at least for a certain choice of model parameters.

Guidelines for Choosing a Model

Here are some guidelines for choosing a model. Think of the data as a spreadsheet or data frame, where rows represent the data units and the columns the data features (variables, groups).

- First establish the *type* of the features (quantitative, qualitative, discrete, continuous, etc.).
- Assess whether the data can be assumed to be independent across rows or columns.
- Decide on the level of generality of the model. For example, should we use a simple model with a few unknown parameters or a more generic model that has a large number of parameters? Simple specific models are easier to fit to the data (low estimation error) than more general models, but the fit itself may not be accurate (high approximation error). The tradeoffs discussed before play an important role here.
- Decide on using a classical (frequentist) or Bayesian model.

Multivariate Normal Models

Recall that $X \sim \mathcal{N}(\mu, \sigma^2)$ means that $X = \mu + \sigma Z, \quad Z \overset{\text{iid}}{\sim} \mathcal{N}(0, 1)$.

Let $\boldsymbol{Z} = [Z_1, \ldots, Z_n]^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$ be a vector of iid $\mathcal{N}(0, 1)$ components.

Definition: Multivariate Normal Distribution

An m-dimensional random vector $\boldsymbol{X}$ that can be written in the form

$$\boldsymbol{X} = \boldsymbol{\mu} + \mathbf{B}\,\boldsymbol{Z}$$

for some m-dimensional vector $\boldsymbol{\mu}$ and $m \times n$ matrix $\mathbf{B}$, with $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n)$, is said to have a multivariate normal or multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\Sigma = \mathbf{B}\mathbf{B}^\top$. We write $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$.

Probability Density

The probability density of an m-dimensional multivariate normal distribution has a similar form to the density of the one-dimensional normal distribution:

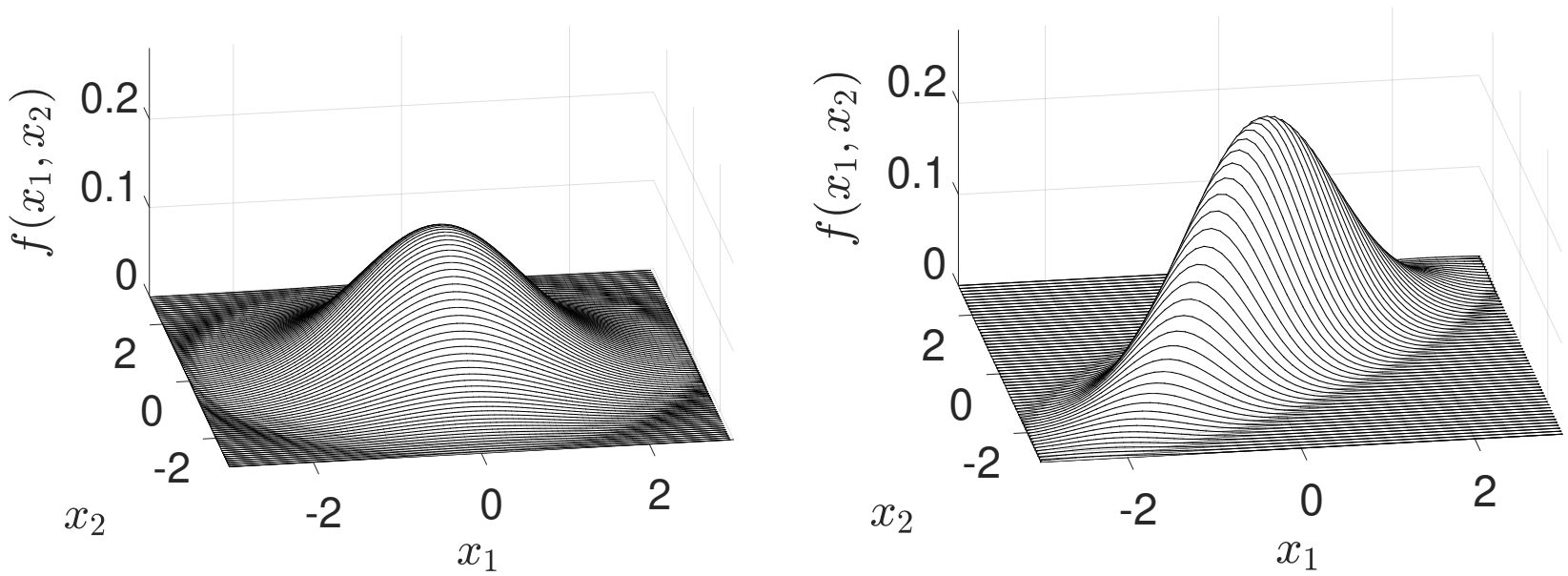
Theorem: Density of a Multivariate Random Vector

Let $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where the $m \times m$ covariance matrix Σ is invertible. Then $\boldsymbol{X}$ has pdf

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^m \, |\Sigma|}} \, \mathrm{e}^{-\frac{1}{2} \, (\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}, \quad \boldsymbol{x} \in \mathbb{R}^m.$$

Probability Density

Here are the pdfs of two bivariate (that is, two-dimensional) normal distributions. In both cases the mean vector is $\boldsymbol{\mu} = [0, 0]^\top$ and the variances (the diagonal elements of Σ) are 1. The correlation coefficients (or, equivalently here, the covariances) are respectively $\varrho = 0$ and $\varrho = 0.8$.



Properties of the Multivariate Normal Distribution

The main reason why the multivariate normal distribution plays an important role in data science and machine learning is that it satisfies the following properties:

1. Affine combinations are normal.
2. Marginal distributions are normal.
3. Conditional distributions are normal.

More on this later!