Crash Course on ML & AI for Economists:

What Mathematics Offers for Economics & Business

Iker Caballero Bragagnini[1]

[1] In representation of Virus Matemático

16/10/2024

Outline

Motivation

This series of talks is intended to give young economics and business students a good sense and understanding of the statistical foundations of Machine Learning and Artificial Intelligence techniques applied in economics and business. We hope this can be a good introduction to new techniques that can be helpful for students' future careers and research.

The main goal of Virus Matemático is to raise awareness of the importance of mathematics and its usefulness in daily life.

Virus Matemático

Virus Matemático is a student-led initiative in Spain to promote interest and mathematics education through playful and creative activities. The idea is to "infect" participants with a passion for mathematics by using innovative and collaborative strategies.

Figure: Members and collaborators in a street event with children.

About Me

- Education:
  - MSc in Computational and Mathematical Engineering
  - MSc in Statistics and Operations Research
  - BSc in Economics with Minor in Applied Maths
- Work Experience:
  - Quantitative Researcher & Developer - Metron Trading
  - Quantitative Credit Risk Analyst - Deloitte
  - Research Intern - Centre de Recerca Matemàtica
  - Research Intern - RISKcenter (Universitat de Barcelona)
- Interests:
  - Financial Mathematics & Statistics
  - Education
  - Academic Research
- Contact:
  - LinkedIn
  - Twitter

Summary of this Crash Course

1. Foundations of Machine Learning and Artificial Intelligence
   - What is ML & AI?
   - What is their relationship to Mathematics and Statistics?
   - Why are they important for economics and business?
   - Why are they important to you?
   - Multivariate Statistics
   - Statistical Learning Theory
2. ML & AI for Macroeconomics
   - How and why ML and AI are applied to Macroeconomics?
   - Linear Models
   - Supervised Algorithms
   - Neural Networks

Summary of this Crash Course

3. ML & AI for Microeconomics
   - How and why ML and AI are applied to Microeconomics?
   - Causality
   - Simulation and Reinforcement Learning
4. ML & AI for Finance
   - How and why ML and AI are applied to Finance?
   - Time Series
   - Statistical Finance
   - Advances in Financial Machine Learning

Outline of this section

1. What is ML & AI?
2. What is their relationship to Mathematics and Statistics?
3. Why are they important for economics and business?
4. Why are they important to you?
5. Multivariate Statistics

What is ML & AI?

You might commonly hear people or even experts use Machine Learning (ML) and Artificial Intelligence (AI) interchangeably, but even though both fields are closely related, they differ in some aspects. It is important to understand first what we are dealing with.

- **Artificial Intelligence:** Simulation (?) of human intelligence by machines, enabling them to perform tasks that would require human intelligence such as reasoning, learning, problem-solving, or even more advanced tasks given the latest advances.
- **Machine Learning:** Subset of AI that enables machines to learn from data and improve their performance over time without being explicitly programmed. These algorithms identify patterns, make predictions, and adapt based on experience (new data).

What is ML & AI?

Some similarities

- Both aim to create intelligent systems capable of performing tasks autonomously.
- Both rely on datasets to identify patterns and improve performance.

Some differences

- AI is a broad field encompassing ML, robotics, neuroscience, and more, while ML specifically focuses on learning from the data.
- AI aims to replicate cognitive functions, while ML is more about pattern recognition and optimization based on data (no need to replicate human intelligence).

What is its relation to Mathematics and Statistics?

Mathematics plays a crucial role in the development of different techniques and models in ML and AI, encompassing areas such as:

- **Linear Algebra:** Matrices and vector operations.
- **Calculus:** Gradients and numerical integration.
- **Probability and Combinatorics**: Distributions and counting.
- **Optimization**: Optimization algorithms and loss functions.

Moreover, statistics is a key tool that uses mathematical tools and results from areas such as:

- **Inference:** Allows to conclude and predict based on data.
- **Probability**: Allows to model randomness for different tasks.
- **Many more...**

Why is it important for economics and business?

ML & AI enhance the ability to handle large datasets, provide more accurate predictions, automate complex tasks, and uncover insights.

AI & ML

- Handle massive datasets and real-time information processing.
- Uses complex algorithms to capture non-linear patterns to forecast
- Greater personalization based on user data.
- Scalable across datasets and complex environments.
- Automation of complex tasks.

Classical methods

- Manual or rule-based systems that use smaller datasets.
- Rely upon econometric models and regressions.
- Broadly segmented based on limited data.
- Difficult to scale as data grows
- Automation of simple repetitive tasks.

Why are they important to you?

New job opportunities for economics/business students:

- Data Scientists
- Quantitative Analyst
- Policy Analysts
- Business Intelligence
- Economic Forecasters
- Supply Chain Analysts
- Marketing Analysts
- ...

All of these opportunities are fresh out of the oven for you, and most of them now use ML & AI techniques. Therefore, **you need to understand how the techniques and models used in these roles work.**

Translation: You need to understand the maths and the statistics behind them.

Why are they important to you?

A **common mistake** of economics and business students (and surely other students as well) who want to get these quantitative roles is that they hustle their way to the interview phase just to get demolished by **technical** (but most times **conceptual**) questions about **statistics** and **mathematics**.

In this series, we tackle this problem, so that each of the students can conceptually understand the mathematics and statistics used in this role and that create the building blocks of ML & AI. **Stay tuned**.

Multivariate Statistics

Before delving into the specific techniques and models in ML & AI, we first need to review some basic statistics that will help us throughout the discussions.

Because we normally deal with various variables/attributes, we will start with multivariate statistics, where we will understand:

- Multivariate Data
- Random Vectors
- Multivariate Statistics
- Distances
- Multivariate Distributions
- The Curse of Dimensionality

Multivariate Statistics: Multivariate Data

The databases you normally deal with are multivariate by nature, which means that you will always have many observations of different characteristics or attributes. These are normally represented in a table like the following (is this the only way?):

	Variable 1	Variable 2	$\cdots$	Variable k	$\cdots$	Variable p
Item 1:	x_{11}	x_{12}	$\cdots$	x_{1k}	$\cdots$	x_{1p}
Item 2:	x_{21}	x_{22}	$\cdots$	x_{2k}	$\cdots$	x_{2p}
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
Item j:	x_{j1}	x_{j2}	$\cdots$	x_{jk}	$\cdots$	x_{jp}
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
Item n:	x_{n1}	x_{n2}	$\cdots$	x_{nk}	$\cdots$	x_{np}

Figure: Table of multiple measurements for n items and p attributes.

Multivariate Statistics: Multivariate Data

We can take this table representation of the data as a two-dimensional $n \times p$ array, which we can think about as a matrix:

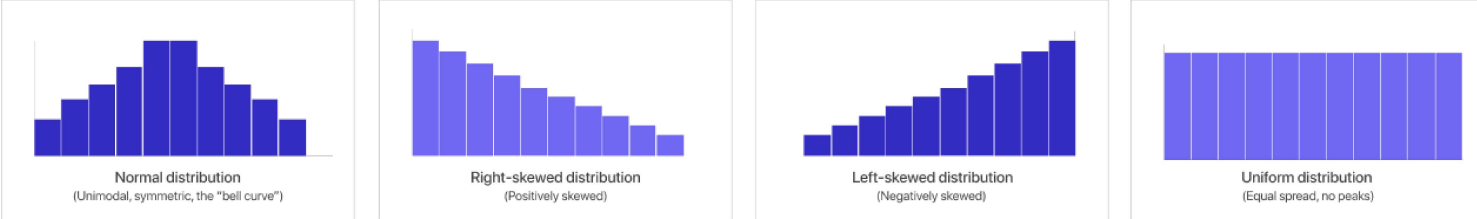
$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2k} & \dots & x_{2p} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{j1} & x_{j2} & \dots & x_{jk} & \dots & x_{jp} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nk} & \dots & x_{np} \end{bmatrix}$$

where x_{jk} is the measurement of the kth attribute for the jth item or individual. Given this representation, we can use **vectors** and **matrices** to easily compute different results.

Multivariate Statistics: Multivariate Data

However, the data does not tell us anything by putting it into a matrix or a table, we need to use **probability** and **statistics** to obtain useful insights. The following logic motivates their usage:

- We have n instances for each of the p attributes. This means we have n examples of each characteristic, which will be distributed o organized in some way (depending on the nature of the process generating the data).



- Hence, we can model an attribute k as **random variable** with some distribution, for which an observation j is a draw of this underlying distribution.

$$x_k \sim Dist_k(\theta) \quad \text{for} \quad k = 1, 2, ..., p \;\Rightarrow\; x_{1k}, x_{2k}, ..., x_{nk}$$

Multivariate Statistics: Random Vectors

Most of you will have notions of univariate and bivariate statistics, but we commonly need to deal with $p > 2$ attributes in ML and AI. Therefore, we go one step further and abstract to **random vectors**, which is simply a p-dimensional vector with p random variables as entries.

$$\mathbf{x}_j = \begin{bmatrix} x_{j1} & x_{j2} & \dots & x_{jk} & \dots & x_{jp} \end{bmatrix} \quad \text{for} \quad j = 1, 2, ..., n$$

If we take into account that in our database we would have n draws of the p variables (results are realized), we can obtain the matrix $\mathbf{X}$ of data again.

Now, this allows us to operate with multiple random variables at the same time using **linear algebra**. Most ML and AI techniques are explained using vectors and matrices because it is easier to understand.

We can compute descriptive statistics in more than 1 dimension (not just a number):

- Sample Mean:

$$\bar{\mathbf{x}} = \frac{1}{n}\begin{bmatrix}\mathbf{1}^T\mathbf{x}_1 & \mathbf{1}^T\mathbf{x}_2 & \dots & \mathbf{1}^T\mathbf{x}_p\end{bmatrix}$$

$$\text{where} \quad \mathbf{1}^T\mathbf{x}_k = \frac{\sum_{j=1}^{n} x_{jk}}{n} \quad \text{for} \quad k = 1, 2, ..., p \quad \& \quad \mathbf{1} = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}$$

- Sample Covariance:

$$\widehat{Cov}(\mathbf{x}_i, \mathbf{x}_k) = \sigma_{i,k} = \frac{1}{n-1}\tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_k = \frac{\sum_{j=1}^{n}(x_{ji} - \bar{x}_i)(x_{jk} - \bar{x}_k)}{n-1}$$

$$\text{where} \quad \tilde{\mathbf{x}}_k = \begin{bmatrix} x_{1k} - \bar{x}_k & \dots & x_{nk} - \bar{x}_k \end{bmatrix} \quad \text{for} \quad k = 1, 2, ..., p$$

$$\textbf{recall:} \quad Cov(\mathbf{x}_k, \mathbf{x}_k) = Var(\mathbf{x}_k)$$

- Sample Covariance Matrix:

$$\tilde{\mathbf{S}} = \left[\sigma_{ik} = \frac{1}{n-1}\tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_k\right] = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \dots & \sigma_{1,k} & \dots & \sigma_{1,p} \\ \sigma_{1,2} & \sigma_2^2 & \dots & \sigma_{2,k} & \dots & \sigma_{2,p} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \sigma_{1,k} & \sigma_{2,k} & \dots & \sigma_k^2 & \dots & \sigma_{k,p} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{1,p} & \sigma_{2,p} & \dots & \sigma_{k,p} & \dots & \sigma_p^2 \end{bmatrix}$$

- Sample Correlation Matrix:

$$\mathbf{R} = \left[r_{ik} = \frac{\sigma_{ik}}{\sqrt{\sigma_i^2}\sqrt{\sigma_k^2}} \right] = \begin{bmatrix} 1 & r_{1,2} & \dots & r_{1,k} & \dots & r_{1,p} \\ r_{1,2} & 1 & \dots & r_{2,k} & \dots & r_{2,p} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ r_{1,k} & r_{2,k} & \dots & 1 & \dots & r_{k,p} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{1,p} & r_{2,p} & \dots & r_{k,p} & \dots & 1 \end{bmatrix}$$

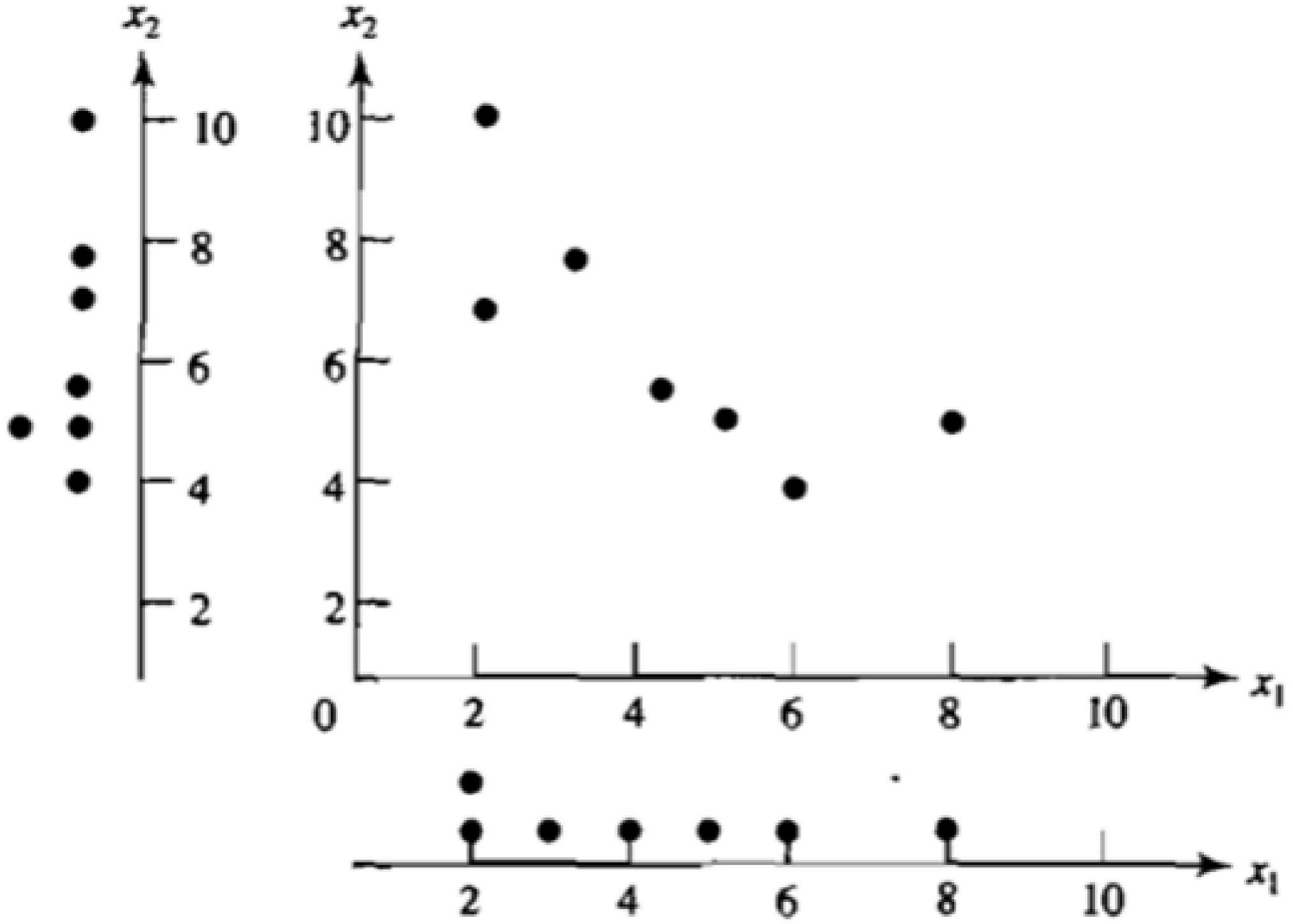


Figure: Scatter plot 2D

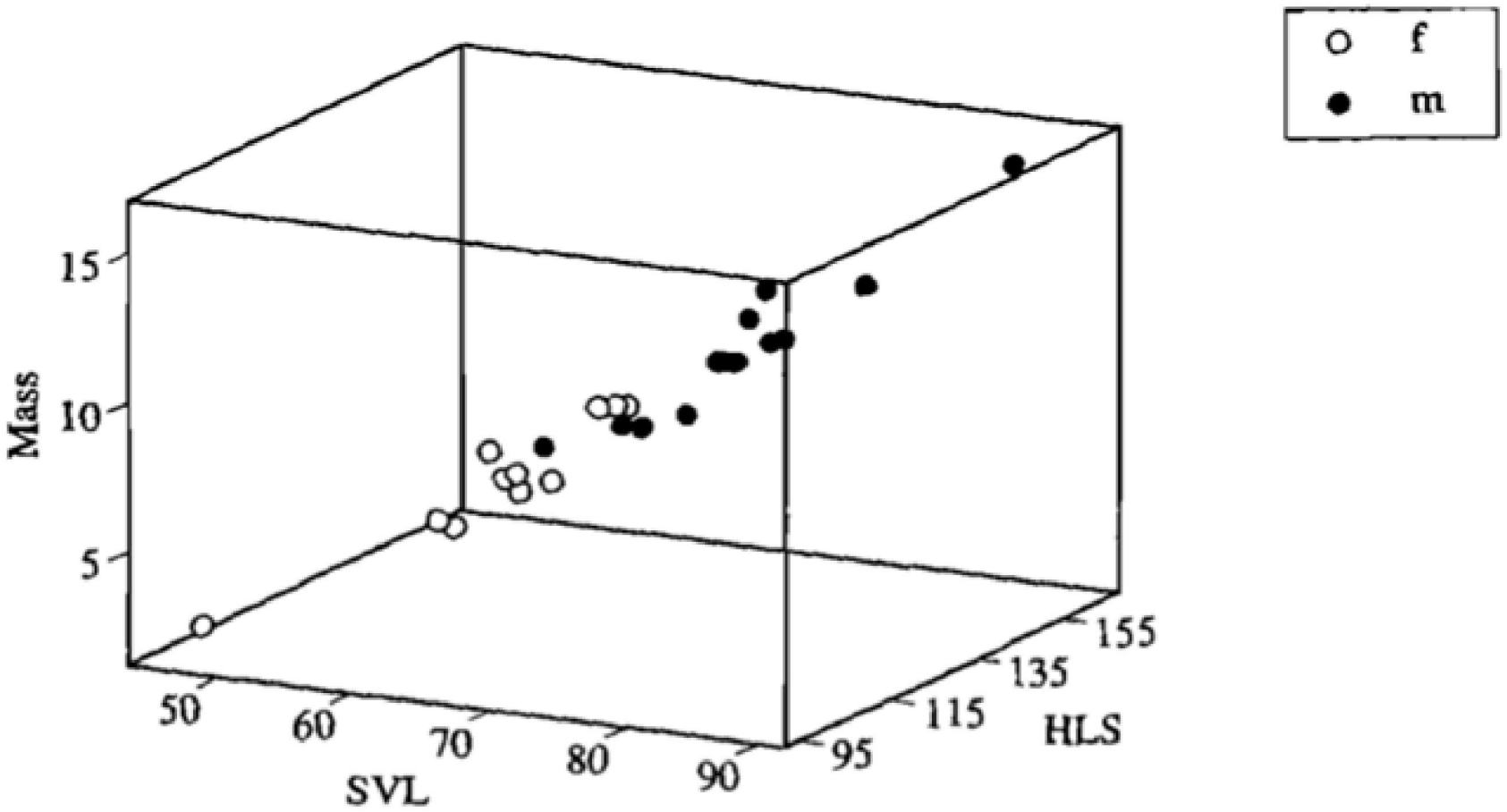


Figure: Scatter plot 3D

Multivariate Statistics: Multivariate Statistics

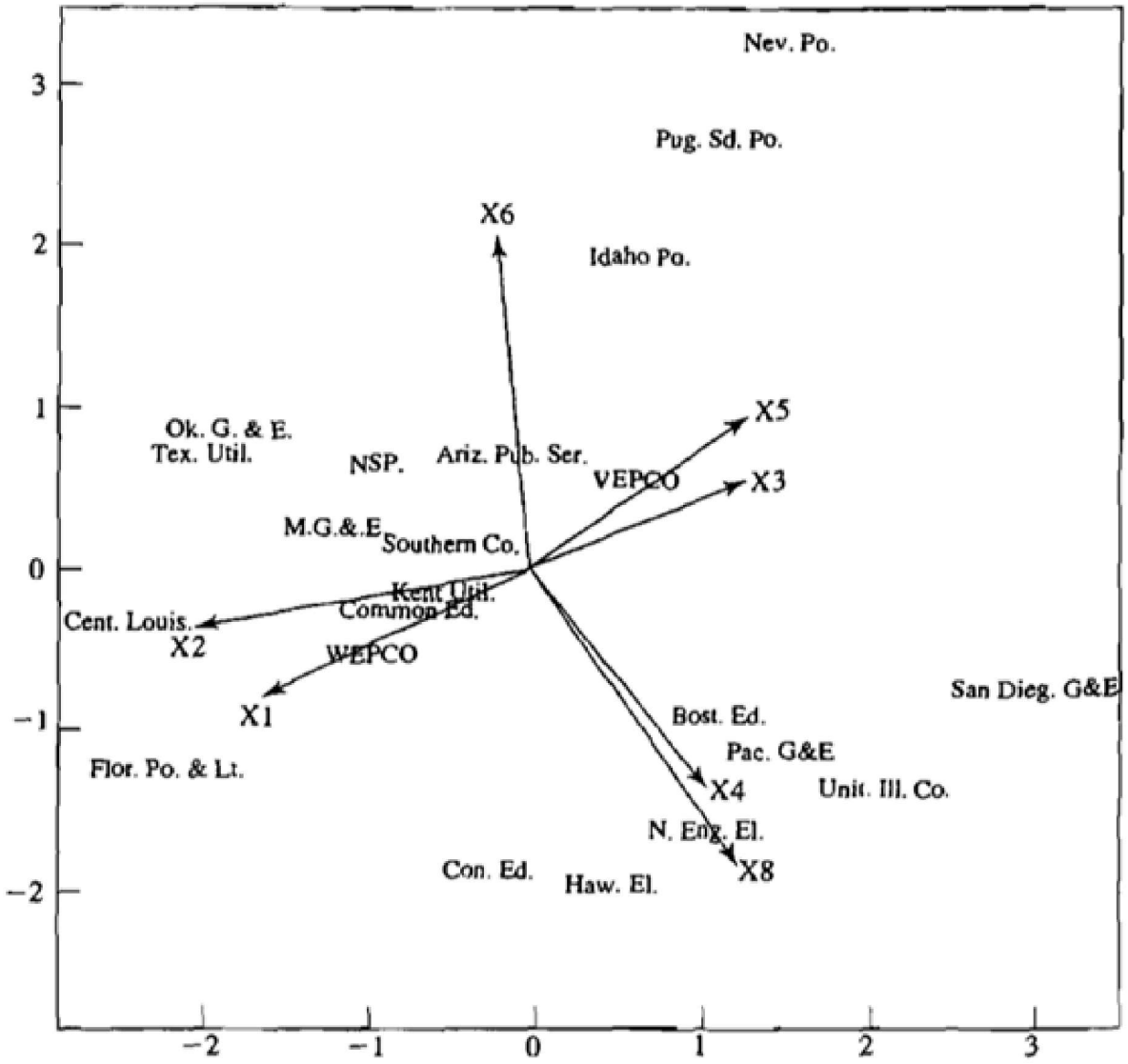


Figure: Biplot

Multivariate Statistics: Multivariate Statistics

Important things to think about with these graphs:

- You can see why the dimensions of the results of usual statistics like the mean, the variance, etc. increase when we use more variables.
- Matrices and vectors are important for manipulating a lot of numbers at the same time without doing various computations.
- From two variables with their own distribution, we can observe some joint behavior when we analyze them together and we can measure and study these relations.
- We can think about methods for solving some problems such as prediction or classification that include some or all of the attributes in the data.
- Because is useful for visualizing data, we might want to find methods to reduce dimensionality and represent multivariate data.

Multivariate Statistics: Distances

Various techniques in statistics and ML & AI use the concept of **distance** between different points P and Q (understood as data vectors or observations). There are different distance measures $d(P,Q)$, but let's delve into the Euclidean.

$$d(P,Q) = \sqrt{(P-Q)^T(P-Q)} = \sqrt{(x_{1k}-x_{1i})^2 + ... + (x_{nk}-x_{ni})^2}$$

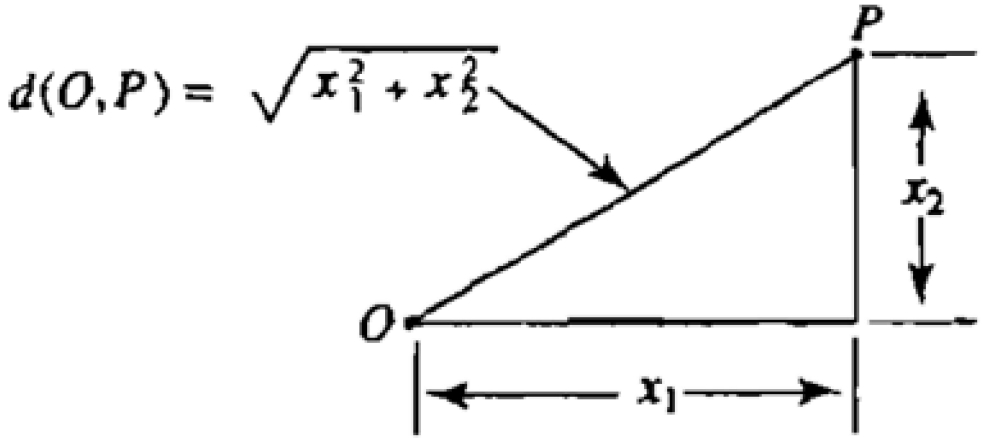


Figure: Euclidean Distance

Multivariate Statistics: Distances

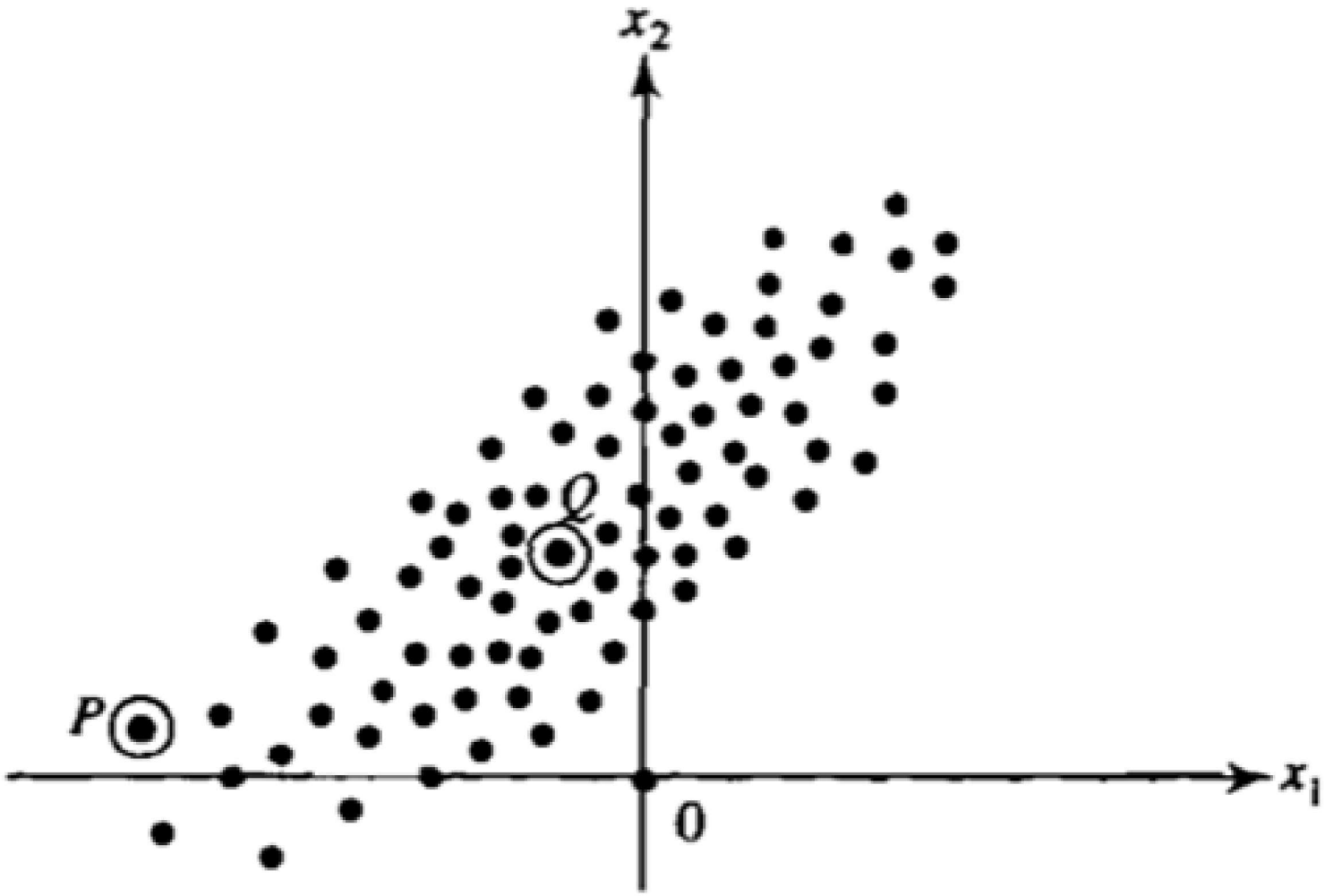


Figure: Cluster of points

Multivariate Statistics: Distances

When we look at the data, we might find that the distance between some observations might tell us something about the underlying distribution or process. Many ML & AI techniques, such as **clustering**, use distances to compute different elements using data observations.

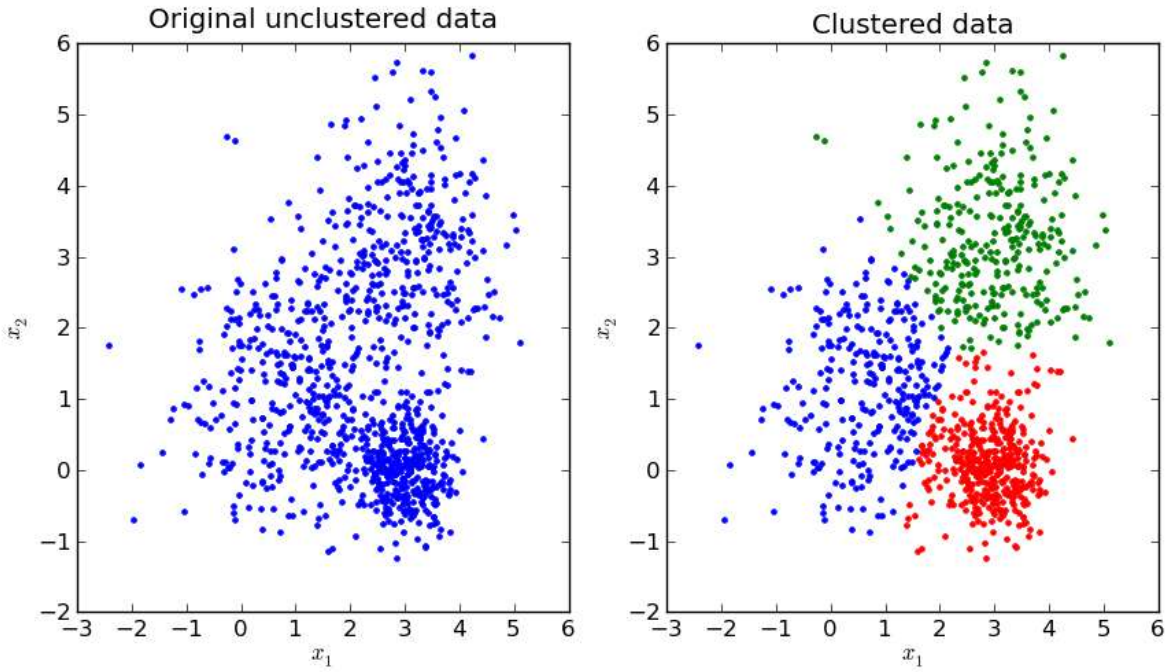


Figure: Clusters of points

Multivariate Statistics: Multivariate Distributions

Understanding random vectors as a group of random variables, it is obvious we can think about distributions with more than one dimension, leading to important distributions such as the **multivariate normal distribution**, in which various ML & AI algorithms are based.

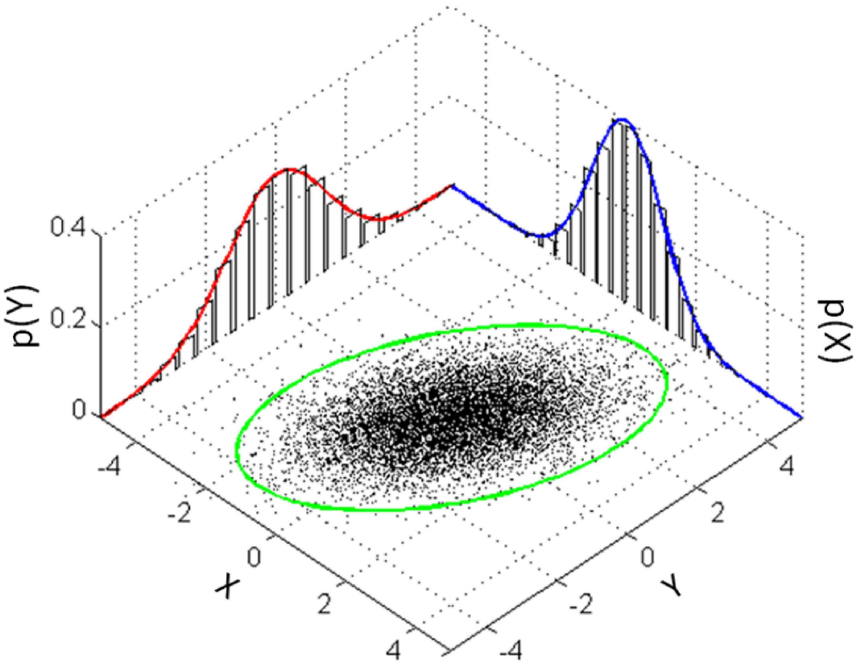


Figure: Bivariate Normal Distribution, with normal marginal distributions.

Multivariate Statistics: Multivariate Distributions

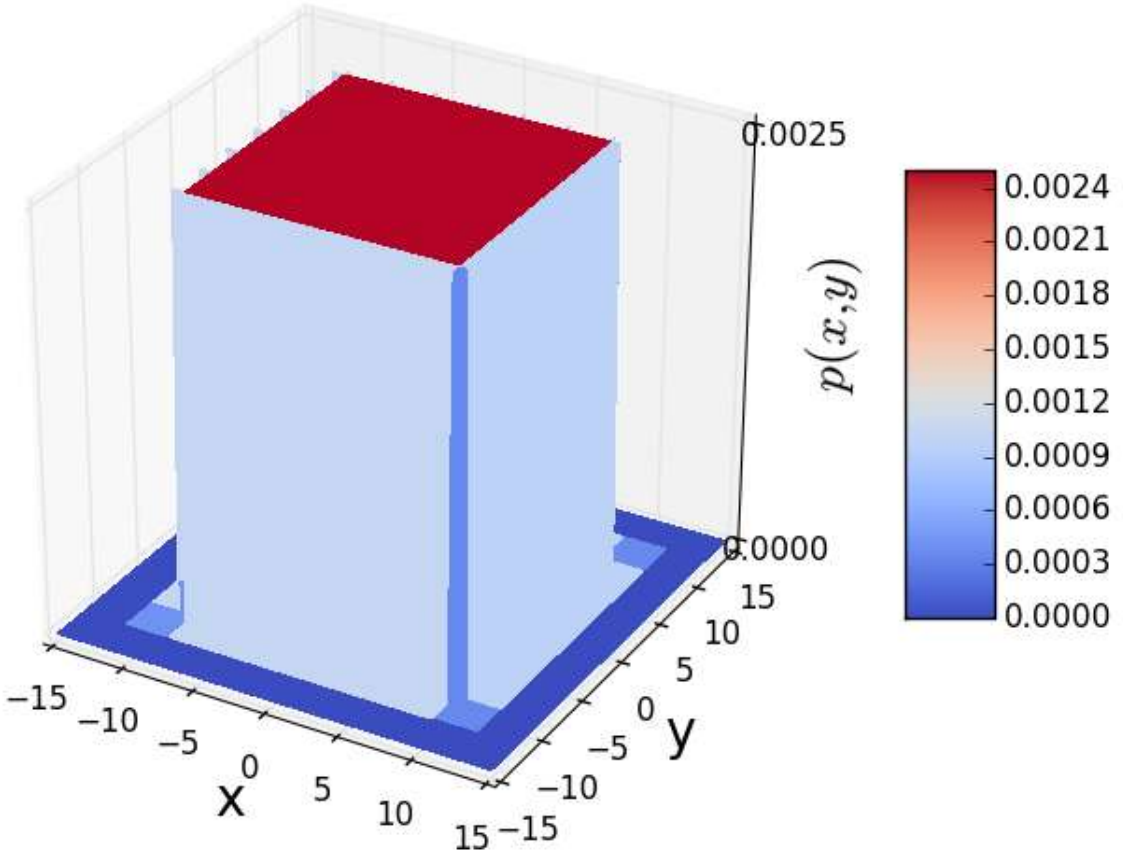


Figure: Multivariate Uniform Distribution

Multivariate Statistics: The Curse of Dimensionality

It seems like using multiple variables gives us a lot of information if we use adequate analyses and methods, and will make our ML & AI algorithms work better and better. Still, problems appear due to the **curse of dimensionality** such as the following:

- **Data Sparsity:** makes it difficult to find significant patterns because the available data cannot populate the space densely.
- **Distances**: distances become less intuitive and, because we consider many dimensions, the difference between farthest and nearest points diminishes, affecting algorithms based on them.
- **Computational Resources**: More variables mean more numbers and more numbers mean we need more computational power to compute different elements.
- **Overfitting**: Algorithms capture more noise than underlying patterns.

Multivariate Statistics: The Curse of Dimensionality

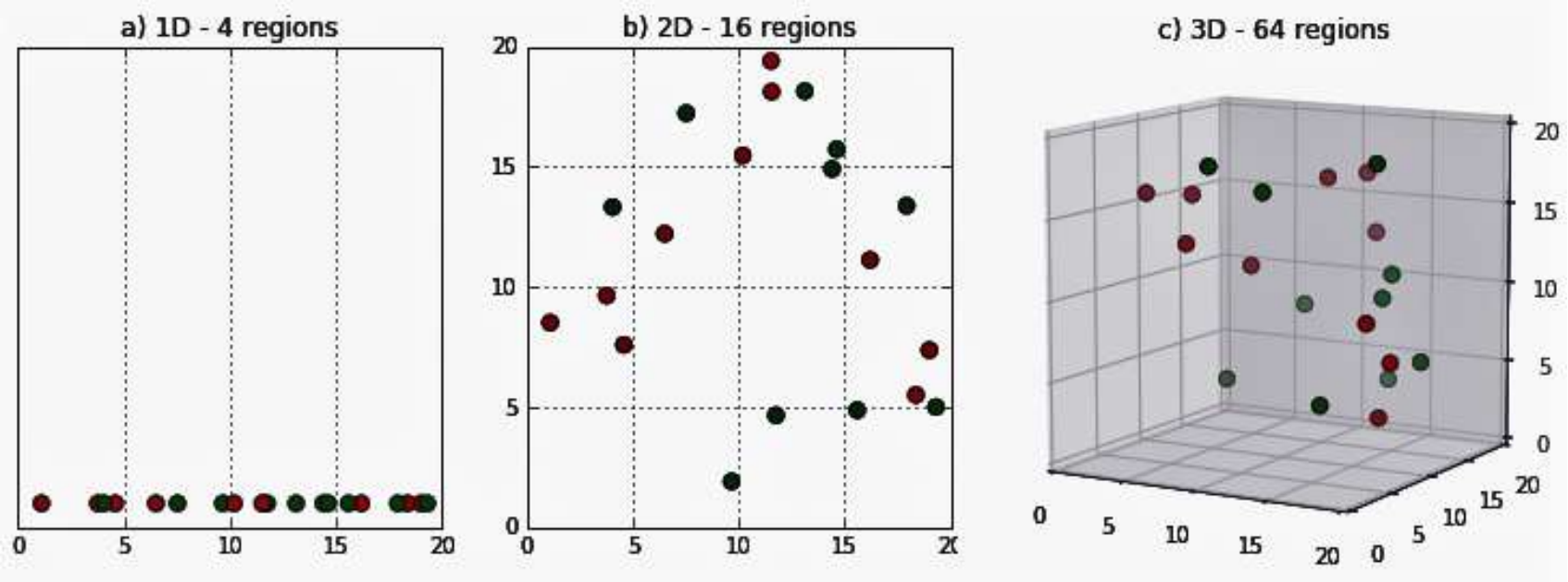


Figure: The regions the algorithm needs to analyze increases with every dimension.

Multivariate Statistics: The Curse of Dimensionality

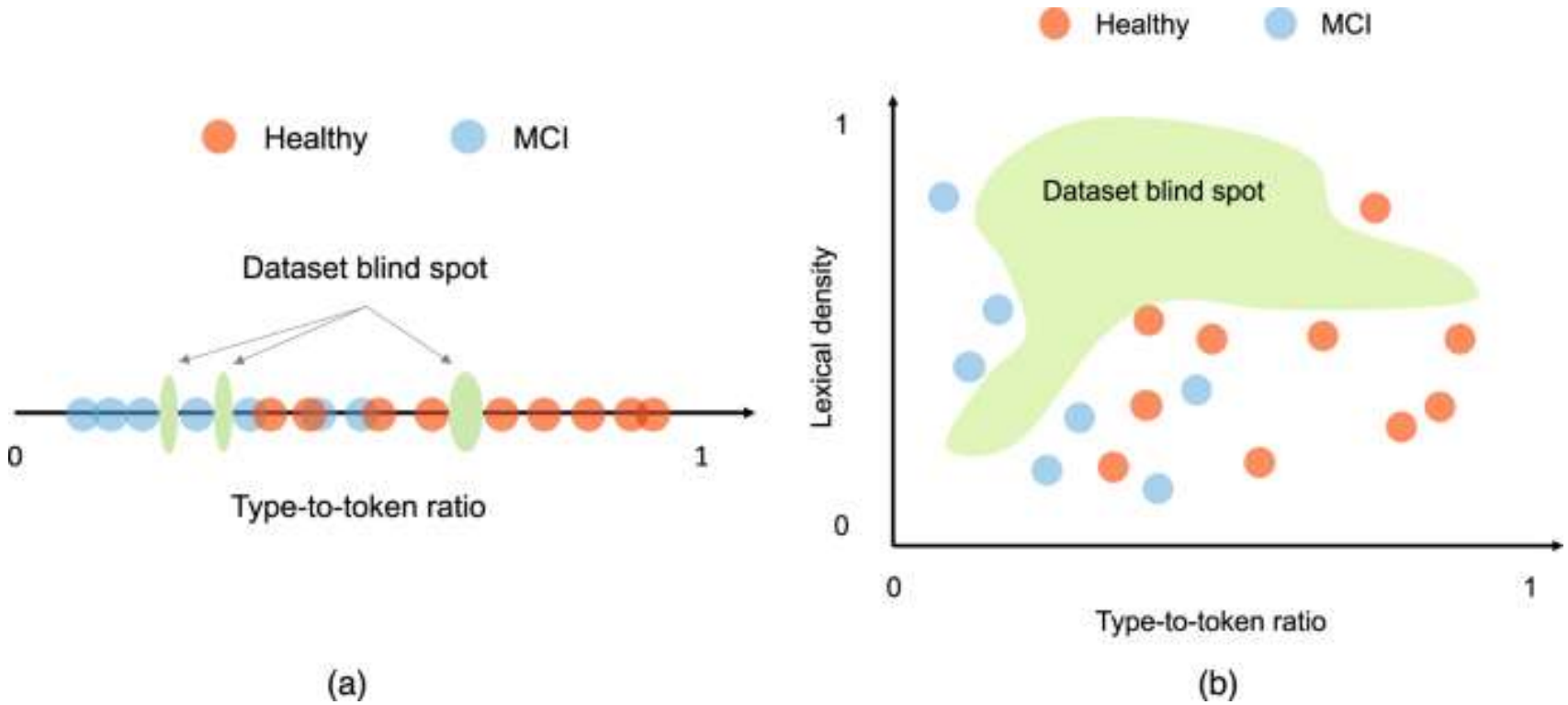


Figure: Datasets shows just a portion of the spaces we want to learn about.

Statistical Learning Theory

Now that we have the basic tools to understand the fundamental statistical and mathematical aspects of modelling with various data, which is foundational for ML & AI, we can go to one of the most interesting questions for newbies: **How do algorithms "learn"?**. The mathematical theory that deals with how algorithms can learn patterns from data and make predictions is called **statistical learning theory**, and we will take a quick look at some important aspects that characterize ML & AI and differentiates them from fields like statistical inference and econometrics.

- Learning Model and Risk Minimization
- Learning Problems
- Consistency
- Generalization

Statistical Learning Theory: Learning Model & Risk Minimization

The general model for learning is the following:

- A generator (G) of random vectors $\mathbf{x} \in \mathbb{R}^n$, drawn independently from a fixed but unknown probability distribution $F(\mathbf{x})$.
- A supervisor (S) that returns a result y for each vector $\mathbf{x}$, according to the conditional distribution $F(y|\mathbf{x})$, also fixed but unknown. This is the general case, which includes the supervisor using a function such that $y = f(\mathbf{x})$
- A learning machine LM capable of implementing a set of functions $f(\mathbf{x}, \alpha)$ for $\alpha \in \Lambda$, where Λ is the set of parameters. The elements of α might not need to be a vector but any abstract parameter.

Statistical Learning Theory: Learning Model & Risk Minimization

See that I have assumed a supervisor S? This is because this theory was developed for what is called **supervised learning**, but there are other forms:

- **Supervised Learning:** Learning from labeled data to make predictions (e.g., classification and regression).
- **Unsupervised Learning:** Learning from unlabeled data to find structure or patterns (e.g., clustering and dimensionality reduction).
- **Reinforcement Learning:** Learning through rewards and penalties in an environment (e.g., game playing and robotics).

Unsupervised learning is also called **multivariate analysis**. We will focus in supervised methods, given that they are the most famous ones.

Statistical Learning Theory: Learning Model & Risk Minimization

The problem of learning is to select the function from the function set $f(\mathbf{x}, \alpha)$ for $\alpha \in \Lambda$ that best approximates the supervisor's S output y.

The election of the function is based on the training set of l i.i.d observations drawn from $F(y, \mathbf{x}) = F(y|\mathbf{x})F(\mathbf{x})$

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_l, y_l)$$

During the learning process, the LM (learning machine) observes pairs $(\mathbf{x}, y)$ (the training set) and, after training, returns an output $\bar{y}$ for any $\mathbf{x}$. Its goal is for $\bar{y}$ to be as similar as y as possible in some sense.

Statistical Learning Theory: Learning Model & Risk Minimization

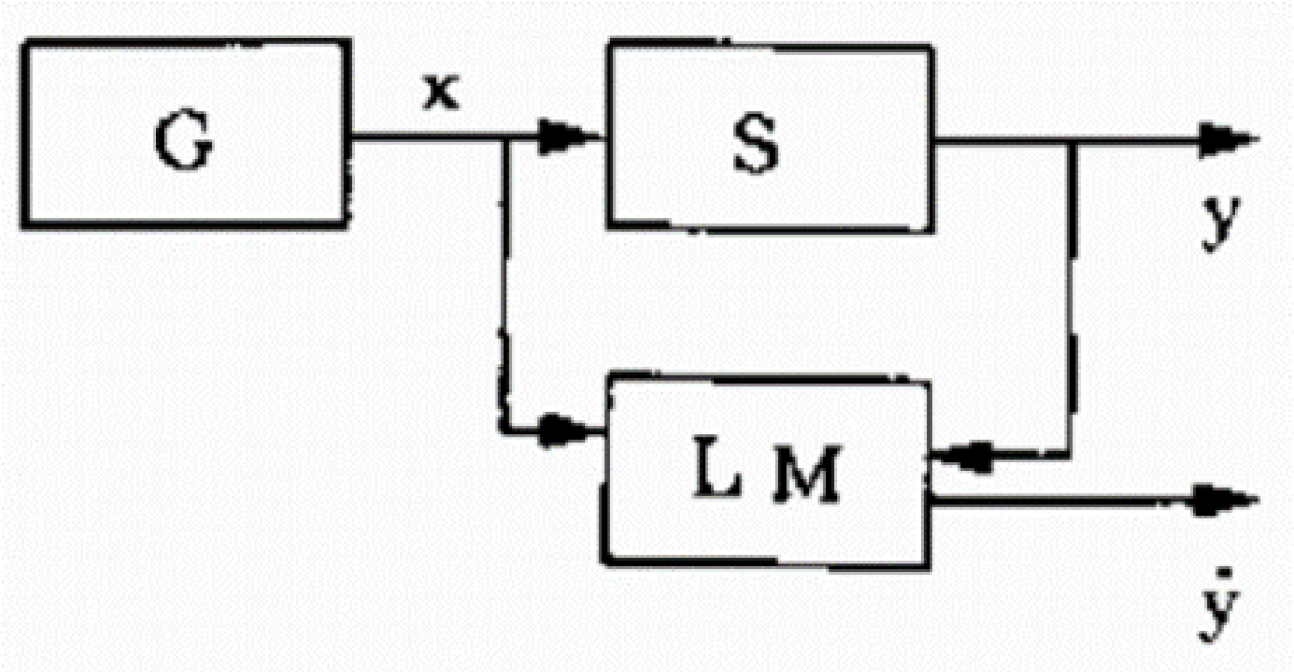


Figure: General Learning Model

Statistical Learning Theory: Learning Model & Risk Minimization

In order to choose the best available approximation to the supervisor's output, one measures the loss or discrepancy $L(y, f(\mathbf{x}, \alpha))$ between S's output y for a given input $\mathbf{x}$ and the response $f(\mathbf{x}, \alpha)$ provided by the LM. Considering the expected risk of this loss, given by the risk functional, the goal is to find a function $f(\mathbf{x}, \alpha_0)$ (from the function class) that minimizes risk $R(\alpha)$, defined as

$$R(\alpha) = \mathbb{E}_{x,y}[L(y, f(\mathbf{x}, \alpha))] = \int L(y, f(\mathbf{x}, \alpha)) dF(y, \mathbf{x})$$

whenever $F(y, \mathbf{x})$ is unknown but we just have a set of data $(\mathbf{x}, y)$. Because we deal with a discrete number of data points, we use the empirical risk functional for empirical risk minimization:

$$R_{emp}(\alpha) = (1/l) \sum_{i=1}^{l} Q(z_i, \alpha) \quad \text{where} \quad Q(z_i, \alpha) = L(y_i, f(\mathbf{x}_i, \alpha))$$

Statistical Learning Theory: Learning Problems

This model for learning is very general and encompasses different problems. The most important ones are classification and regression problems (even though we can consider other types).

- **Regression Problems**: Problems in which $y \in \mathbb{R}$ and $f(\mathbf{x}, \alpha)$ is a set of real functions. We predict a continuous output based on input variables (e.g., predicting house prices).
- **Classification Problems**: Problems in which $y \in \{0, ..., m\}$ and $f(\mathbf{x}, \alpha)$ is a set of discrete functions (functions that take a discrete number of values m). We assign data points to predefined categories (e.g., spam vs. non-spam emails).

In economics, most of the times we are concerned with these types, but other fields like non-parametric statistics also consider the "density estimation problem".

Statistical Learning Theory: Consistency & Generalization

Consistency in statistical inference is a property of estimators in which the estimator for some parameter $\bar{\alpha}_n$ converges in probability to the real value α, which can be expressed as

$$\lim_{n\to\infty} P(|\bar{\alpha}_n - \alpha| > \epsilon) = 0, \quad \forall \epsilon > 0 \quad \Leftrightarrow \quad \text{plim}_{n\to\infty}\ \bar{\alpha}_n = \alpha$$

This means that our estimator tends (in probability) to the "real" value when we increase observations. We are interested in when a learning machine that minimizes empirical risk can actually achieve a small value of risk (can generalize) and when it cannot.

Statistical Learning Theory: Consistency & Generalization

We say that a method of empirical risk minimization is consistent for the set of functions $Q(z, \alpha)$ for $\alpha \in \Lambda$ and for the probability distribution $F(z)$ whenever the following convergences hold:

$$R(\alpha_\ell) \xrightarrow{P} \inf_{\alpha \in \Lambda} R(\alpha) \quad \text{and} \quad R_{\text{emp}}(\alpha_\ell) \xrightarrow{P} \inf_{\alpha \in \Lambda} R(\alpha).$$

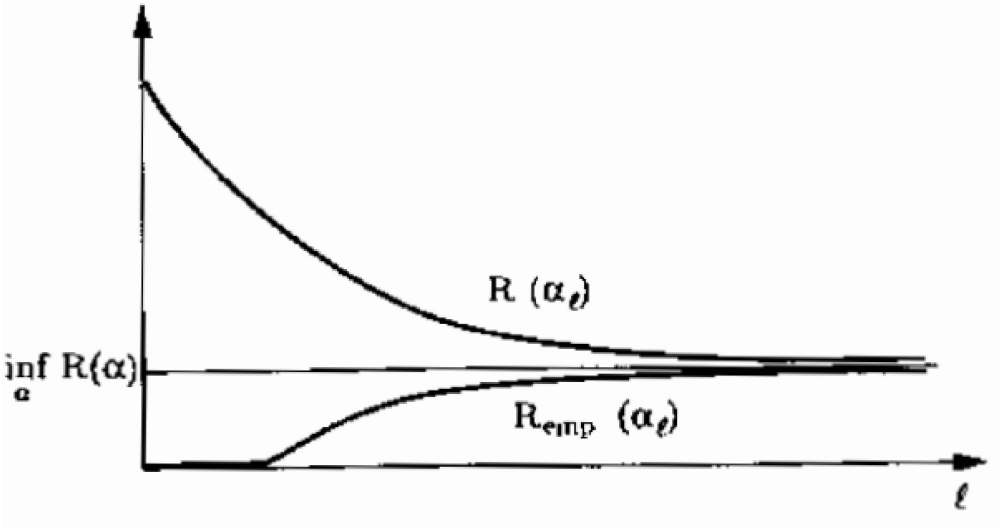


Figure: Convergence of risk and empirical risk to $\inf_\alpha R(\alpha)$

Statistical Learning Theory: Consistency & Generalization

Hence, as our training data set increases, the empirical risk of the learned parameters converge in probability to the true optimal risk (the infimum) and we would be approximating the unknown pattern in an optimal way.

But, why is this important at all? Well, the answer is rooted in the main goal of ML&AI. **We want to generalize to data we have not seen**. Therefore, we just do not care about the **error** we make during training (the **training error**), but also about the error we make on data we have not previously seen, called the **test set**. We would look at this aspect again in the future.

This theory allows us to establish some bounds to the test error and to the expected risk, but this is advanced theory we should not delve into.

Outline

How and Why ML & AI are applied to Macroeconomics

Imagine you are an economist working for the BCE. Some of the most important tasks you might need to develop in your position are:

- Forecast the evolution of important economic indicators such as GDP, unemployment, inflation, etc.
- Analyze quantitatively the policies implemented or designed by monetary and fiscal authorities.
- Read and analyze information sources such as news, social media posts, official documents, etc.
- Study interactions between different variables and agents in various markets and systems to understand the effects and relationships.

We have long used econometrics for modeling in this context, but ML & AI have a better performance for developing these types of tasks and now economists are looking for these skills in their new hires.

How and Why ML & AI are applied to Macroeconomics

Econometric models traditionally relied on **linear relationships** and **specific assumptions** (homoscedasticity, normality, etc.), as they allow for **easier handling** of mathematics and data for our purposes (computational requirements, interpretability, etc.).

However, **complex systems** (in the physical sense) such as financial markets and other types of economic systems exhibit behavior that is contrary to some of the assumptions and types of relationships used.

ML & AI are more **flexible tools**, as they can scale better to **higher dimensions** and have **minimal assumptions**. Let us look at some advantages and disadvantages of using ML & AI in macroeconomics.

How and Why ML & AI are applied to Macroeconomics

Advantages

- Complex pattern recognition
- Scalability with Big Data
- Automation
- Increased Predictive Power

Disadvantages

- Interpretability
- Overfitting
- Data Requirements
- Computational Intensity

Hence, we can see that ML & AI would also create some challenges for economists. This is the reason why econometrics and traditional statistics studied in economics are still used in some areas instead of ML & AI.

How and Why ML & AI are applied to Macroeconomics

The areas in which these state-of-the-art techniques are still not used as much as econometrics are:

- **Causal Inference:** Econometrics has developed methods to isolate cause-effect relationships.
- **Structural Modeling:** Econometrics is still used for developing structural models that explain the economy's mechanisms.
- **Small-Sample Analysis:** Econometrics offers models and techniques specially designed for using small datasets.
- **Economic Theory Integration:** Econometrics allows to incorporate economic theory restrictions to some models.

Linear Regression

Before delving into ML & AI algorithms, let's first review the most basic econometrics we need to know to compare them with these new techniques and understand what we are dealing with.

This is just a kind of refresher or some quick introduction. We will touch on the basics but there is plenty more to know if you want to be a great economist and expert in ML & AI.

Types of Data

The type of models that we use are heavily determined by the type of data we are trying to model. The most common types of data are the following:

- **Cross-sectional Data:** A sample of individuals taken at a certain point in time. [**In this section**]
- **Time Series Data:** An single individual is taken at different points in time. [**ML & AI in Finance**]
- **Pooled Data:** Combines cross-sectional and time series data. Different individuals can be selected at a certain point in time, where various points are measured. [**Out of topic**]
- **Panel Data:** Combines cross-sectional and time series data. For a sample of individuals, we obtain a time series for each one. [**Out of topic**]

Linear Regression: Basics

- From multivariate statistics, we know that when we assume that random variables y and $x_1, x_2, ..., x_p$ are related through some multivariate probability distribution:

$$p(y, \mathbf{x}) \quad \text{where} \quad \mathbf{x} = [x_1, x_2, ..., x_p]$$

- If this is the case, then by Bayes Theorem (which allows us to consider the probability of some variable given the values of other elements) we can see that:

$$p(y|\mathbf{x}) = \frac{p(y, \mathbf{x})}{p(\mathbf{x})} \Rightarrow p(y, \mathbf{x}) = p(y|\mathbf{x})p(\mathbf{x})$$

- We want to study how some variables of interest y vary when variables $\mathbf{x}$ vary, as we want to study the relationships and effects. Hence, we are focused on $p(y|\mathbf{x})$.

Linear Regression: Basics

- For practical purposes, we are not exactly interested in knowing or approximating $p(y|\mathbf{x})$, but on the expected value y given some variables $\mathbf{x}$ that allow us to explain y.

$$p(y|\mathbf{x}) \Rightarrow E(y|\mathbf{x})$$

- Because of this, we first design a **population model** which allows us to model our data. We suppose that there is a dependent variable y which is equal to some function f of $\mathbf{x}$ added to an error term ε which is taken as random (its distribution p depends on parameters $\boldsymbol{\theta}$).

$$y = f(\mathbf{x}) + \varepsilon \quad \varepsilon \sim p(\boldsymbol{\theta})$$

Linear Regression: Basics

- In econometrics, we make a set of assumptions such as linearity, normality of errors, exogeneity, iid observations, and others that allow us to work with a very simple model. For an individual i, the model is:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_j = \boldsymbol{\beta}^T \mathbf{x}_i + \varepsilon_i \quad \text{for} \quad i = 1, 2, ..., n$$

$$\varepsilon_i \sim N(0, \sigma^2) \quad \text{for} \quad i = 1, 2, ..., n$$

- Because we have the data for n individuals for all $p + 1$ variables, then we can express this with vectors and matrices, which will be important to estimate those $\boldsymbol{\beta}$ parameters:

$$y = \boldsymbol{\beta}^T \mathbf{X} + \boldsymbol{\varepsilon}$$

Linear Regression: Basics

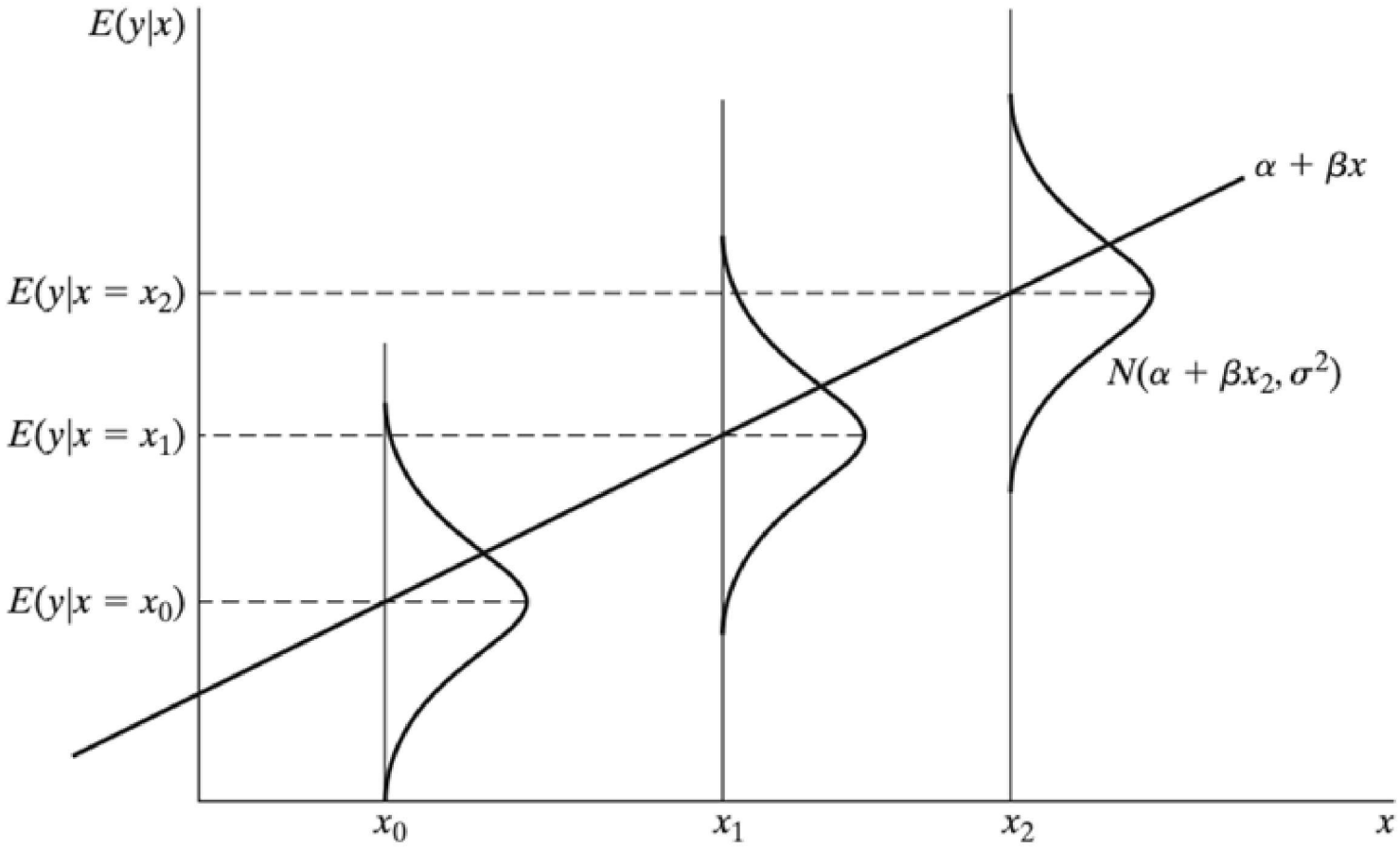


Figure: Population regression model.

Linear Regression: Basics

- Because of our model and our assumptions, we can see that the only thing we need to model $E(y|\mathbf{x})$ is a linear function composed by $\boldsymbol{\beta}$ and the variables in $\mathbf{x}$!

$$E(y|\mathbf{x}) = E(\boldsymbol{\beta}^T\mathbf{x} + \varepsilon|\mathbf{x}) = \boldsymbol{\beta}^T E(\mathbf{x}|\mathbf{x}) + E(\varepsilon|\mathbf{x}) = \boldsymbol{\beta}^T\mathbf{x}$$

- Then we only need to estimate $\boldsymbol{\beta}$ from our (large enough) dataset. Applying ordinary least squares, which tries to minimize the square distances between the real observations and our function, we obtain the OLS estimator:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T\mathbf{x}_i)^2 \Rightarrow \boldsymbol{\beta}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Linear Regression: Basics

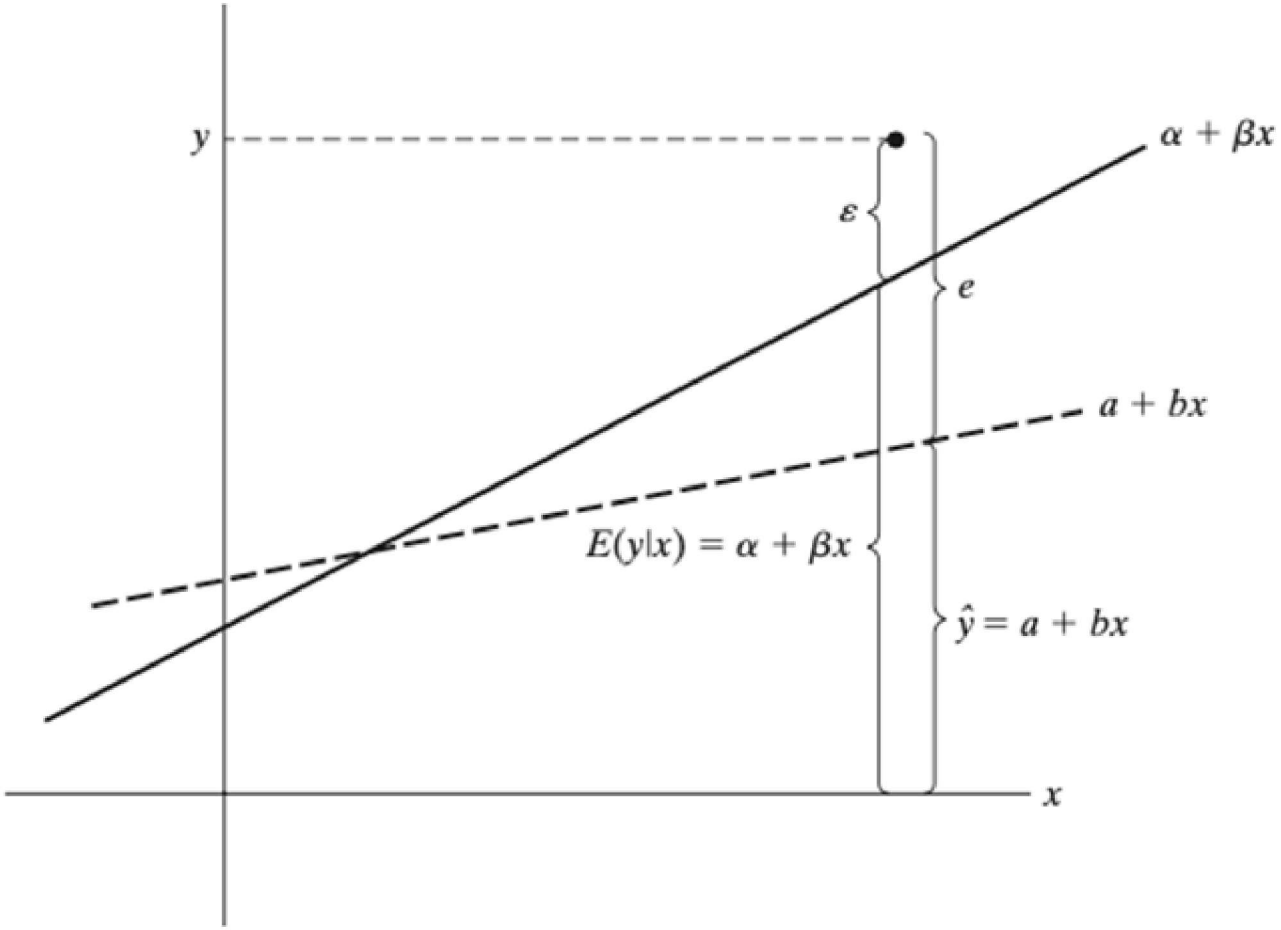


Figure: Sample regression model compared to the real population model.

Linear Regression: Basics

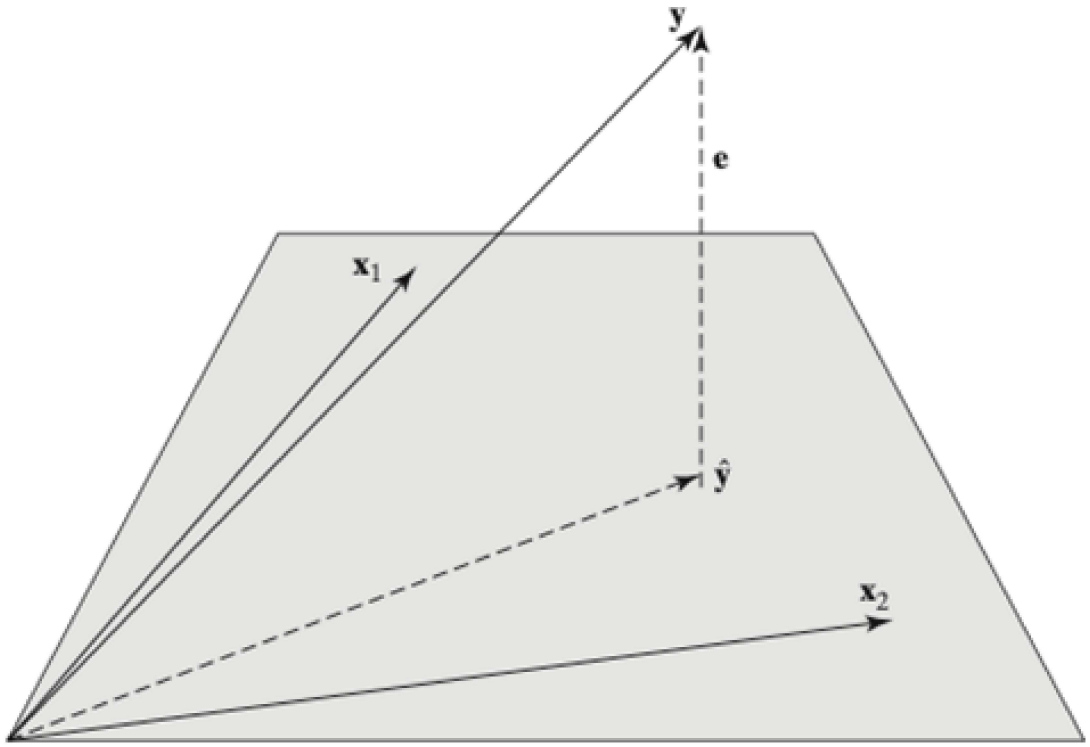


Figure: Linear regression projects $\mathbf{y}$ into the space of explanatory variables

Linear Regression: Loss and Regularization

Now that you have refreshed the basic knowledge teachers provide in any introductory econometrics course, we can talk about loss functions and regularization, which are concepts that can be clearly understood in linear regression but are generalized to many ML & AI techniques. Let us start from the loss function.

- We have talked about how, when **learning** from the data, we use the **ERM** (empirical risk minimization) principle for obtaining our approximating function. Because risk is based on the loss function, then the loss function is very important for the model we will obtain.
- Let us think about the implications of different loss functions.

Linear Regression: Loss and Regularization

- What happens if instead of an L_2 loss function (least squares loss) we use an L_1?

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^T \mathbf{x}_i| \Rightarrow \boldsymbol{\beta}^* = median(\mathbf{x}_i)$$

- What happens if we use a weighted version of the least squares regression?

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} w_i (y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2 \Rightarrow \boldsymbol{\beta}^* = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y}$$

- Have you thought about what happens when instead of directly using convex optimization techniques, you use iterative algorithms?

Linear Regression: Loss and Regularization

Apart from changing the main loss function for different purposes, we might also want our model to stick to relevant variables in order to have better performance. This naturally arises when dealing with multiple variables (think about the curse of dimensionality). A method that helps us deal with this is regularization.

- We want our linear regression or model to reduce the space of explanatory variables or the number of parameters, so that we do not have a very complex model that might not serve our purposes (like generalization).
- Hence, we need to introduce something to the loss function such that our β estimator takes into account the relevant features but ignores or reduces the effect of the irrelevant ones.

Linear Regression: Loss and Regularization

- What happens when we add a L_2 term (Ridge regression) for the parameters β to our loss function?

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \Rightarrow \boldsymbol{\beta}^* = (\mathbf{X}^T \mathbf{X} + \lambda n \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

- What happens when we add a L_1 term (Lasso regression) for the parameters β to our loss function?

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} w_i (y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2 \Rightarrow \boldsymbol{\beta}^* \text{ could be sparse!}$$

- Let's visualize this in order to get the idea of what regularization does...

Linear Regression: Loss and Regularization

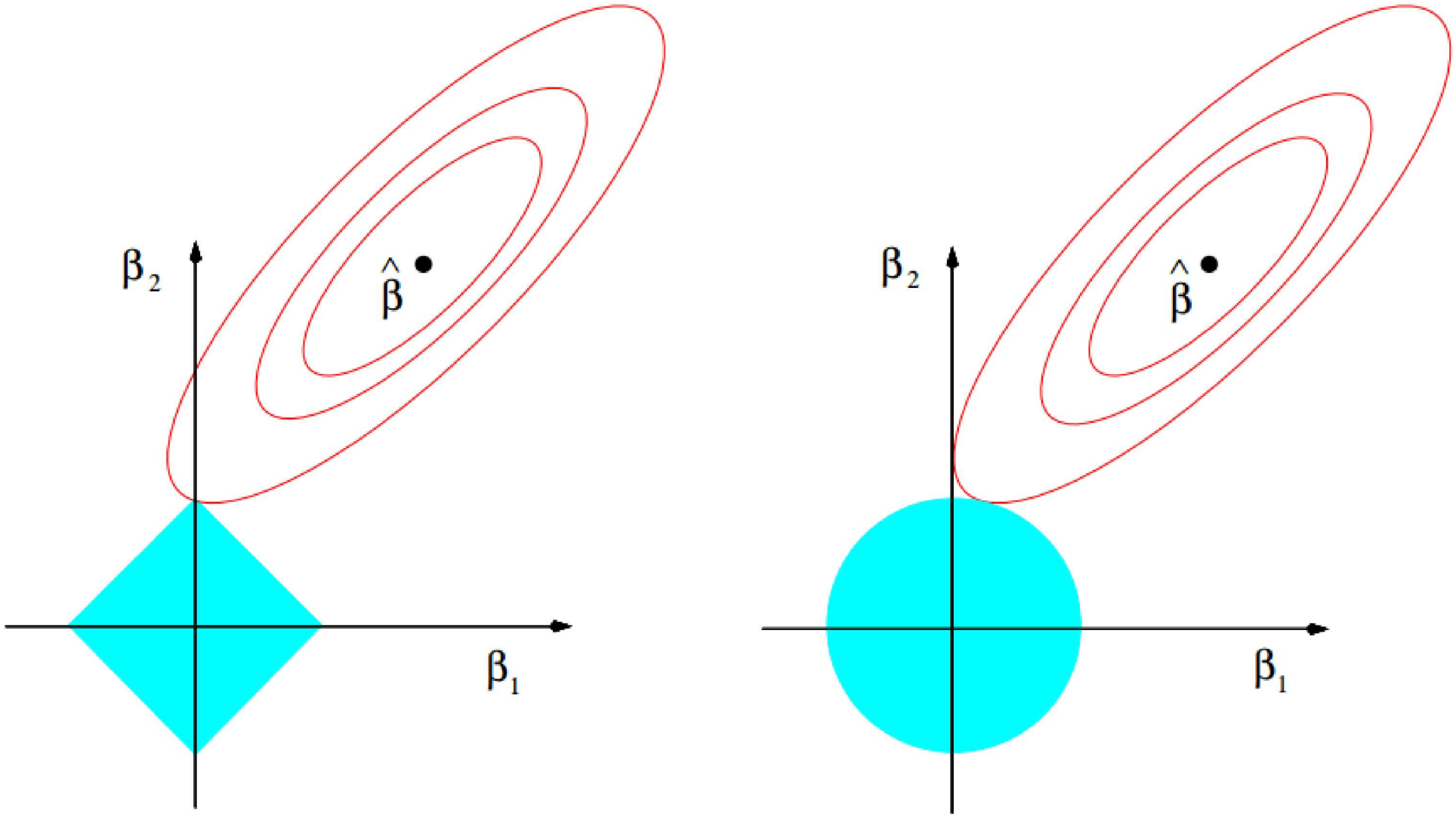


Figure: L_1 and L_2 regularization effect on parameter values.

Linear Regression: Loss and Regularization

Hence, one should keep the following ideas:

- Even though we would love to approximate the distribution, our concern is approximating the conditional expectation as a linear model that satisfies our assumptions.
- We normally use the quadratic loss function, but other loss functions can be used for empirical risk minimization, which lead to different results.
- To improve generalization and handle high-dimensionality, regularization techniques like Ridge and Lasso introduce penalty terms to the loss function.

Linear Classifiers

Now, what if we do not have a continuous target variable Y, but instead we have some discrete target variable that can only take a discrete set of values $\{1, 2, 3, ..., K\}$? Is the previous regression still valid? To apply linear regressions directly poses many problems, but in turn we can use **linear classification** methods. We will explain both the logistic regression and linear discriminant analysis.

- The logistic regression is a generalized linear model that allows to model a discrete or categorical variables through logit transformations.
- Linear discriminant analysis is also based on logit transformations, but assumes that observations from each category follow a Guassian distribution.

Linear Classifiers: Logistic Regression

- Logistic regression is a classification algorithm used for binary and multiclass classification. Unlike linear regression, it models the probability that a given input belongs to a particular class.
- We say that is based in logit transformations because the whole idea is to model the following:

$$\log \frac{P(y=k|\mathbf{X}=\mathbf{x})}{P(y=K|\mathbf{X}=\mathbf{x})} = \beta_{k0} + \beta_k^T \mathbf{x} \quad \text{for} \quad k = 1, 2, ..., K-1$$

- We just need to express the posterior probability of the Kth class like a logistic distribution, so that we obtain the logistic regression:

$$\Rightarrow P(y=k|\mathbf{X}=\mathbf{x}) = P(y=K|\mathbf{X}=\mathbf{x}) \exp(\beta_{k0} + \beta_k^T \mathbf{x})$$

$$\text{assume} \quad P(y=K|\mathbf{X}=\mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T \mathbf{x})}$$

Linear Classifiers: Logistic Regression

- It estimates the probability of an instance belonging to class k using the sigmoid function:

$$P(y = k|\mathbf{x}) = \frac{\exp(\beta_{k0} + \beta_k^T \mathbf{x})}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T \mathbf{x})}$$

$$P(y = K|\mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \beta_l^T \mathbf{x})}$$

  where $\boldsymbol{\beta}$ are the model parameters.
- A threshold (e.g., 0.5) is used to classify the output into one of the two categories.
- Parameters $\boldsymbol{\beta}$ are estimated using MLE:

$$\log[L(\boldsymbol{\beta})] = \sum_{i=1}^{N} \log[P(y_i|\mathbf{x}_i; \boldsymbol{\beta})] = \sum_{i=1}^{N} y_i \beta^T \mathbf{x}_i - \log(1 + e^{\beta^T \mathbf{x}_i})$$

- Because the likelihood is non-linear in β, we use numerical optimization techniques.

Linear Classifiers: Linear Discriminant Analysis

- Linear Discriminant Analysis (LDA) is a classification algorithm based on finding a linear boundary that best separates classes while maximizing variance between classes.
- Assumes each class follows a multivariate normal distribution (we need to estimate the multivariate parameters):

$$p(\mathbf{x}|y = k) \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

  where all classes share the same covariance matrix $\boldsymbol{\Sigma}$.
- The decision rule is derived from Bayes' theorem:

$$P(y = k|\mathbf{x}) = \frac{P(\mathbf{x}|y = k)P(y = k)}{\sum_{l=1}^{K} P(\mathbf{x}|y = l)P(y = l)}$$

Linear Classifiers: Linear Discriminant Analysis

- If we look at the log-odds ratio for comparing the posterior probability of two given classes, we obtain the following:

$$\log \frac{P(y=k|\mathbf{x})}{P(y=l|\mathbf{x})} = \mathbf{x}^T \boldsymbol{\Sigma}^{-1}(\mu_k - \mu_l)^T - \frac{1}{2}(\mu_k + \mu_l)^T \boldsymbol{\Sigma}^{-1}(\mu_k - \mu_l)$$

$$+ \log \frac{P(y=k)}{P(y=l)}$$

- The equal covariance matrices cause the normalization factors and the quadratic parts to cancel. Hence, the decision boundary will be linear between both classes.
- The classifier assigns $\mathbf{x}$ to the class k that maximizes the linear discriminant function, because it is the same as finding the class with the biggest log-odds ratio:

$$\delta_k(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log P(y=k)$$

Linear Classifiers: Linear Discriminant Analysis

- **Linear Discriminant Analysis (LDA):**
  - Assumes same covariance matrix $\mathbf{\Sigma}$ for all classes.
  - Works well when class distributions are similar.
  - Results in a linear decision boundary.
- **Quadratic Discriminant Analysis (QDA):**
  - Allows each class to have its own covariance matrix $\mathbf{\Sigma}_k$.
  - More flexible than LDA but requires more parameters.
  - Results in a quadratic decision boundary.
  - Works better when class variances differ significantly.
- **Pros and Cons:**
  - LDA is simple and robust but assumes equal covariance across classes.
  - QDA is more flexible but requires more data to estimate separate covariances.

Linear Classifiers: Linear Discriminant Analysis

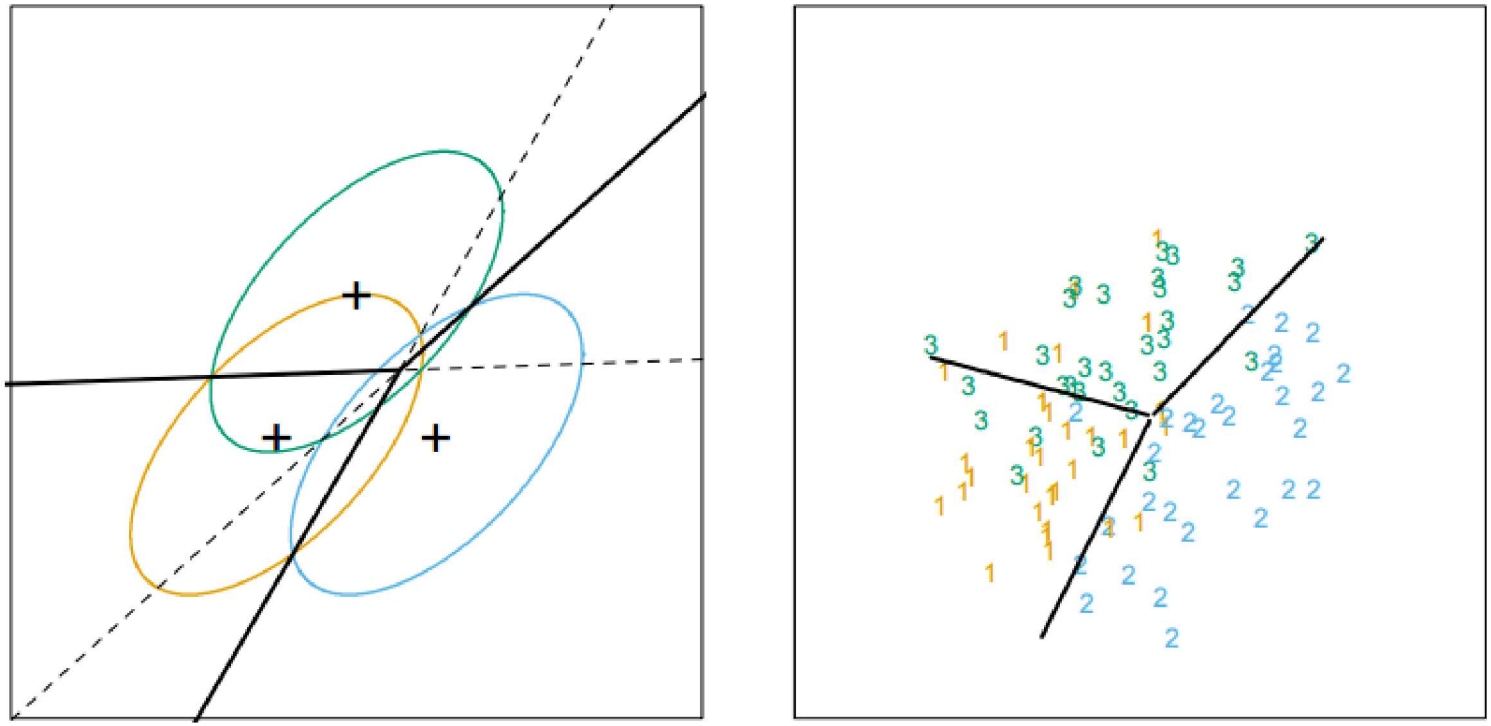
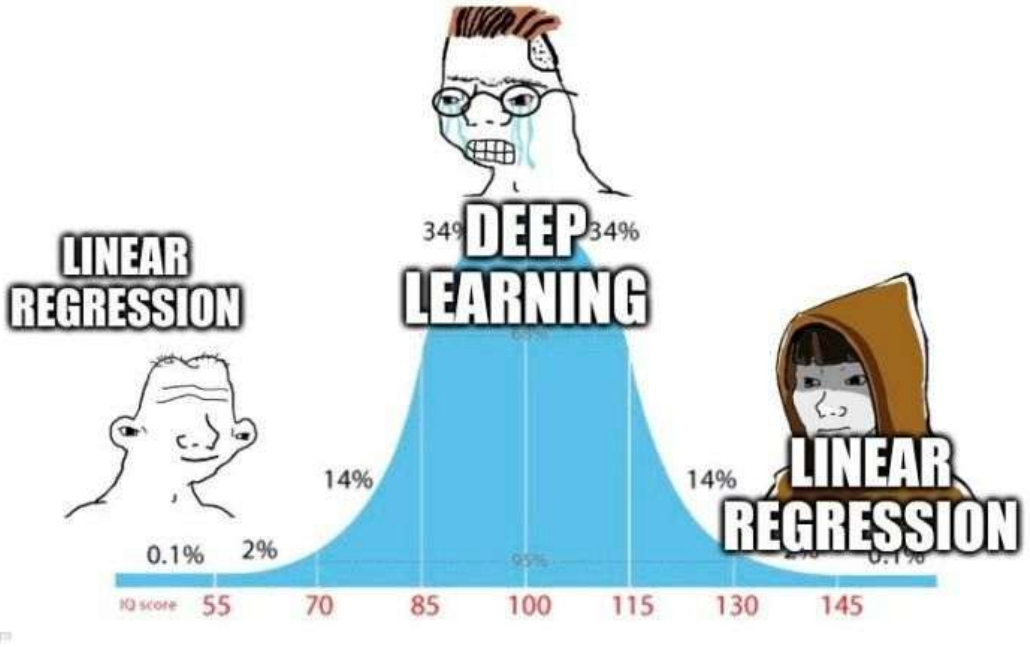


Figure: Decision frontiers in a LDA with 3 different categories

Why linear models first?

We have seen the most basic models from econometrics and mathematical statistics to model different types of problems. Now, you can better understand the use cases we are dealing with and the approaches in ML & AI. **You cannot walk without crawling first**.

You just have learned 75% of the type of models you might use. Most of the time in your job you will never apply a complex ML algorithm, but simple linear regressions.



Supervised Algorithms

Now let us fully delve into the domain of ML & AI algorithms... at last. We will look at the most important aspects of the techniques, but always keep in mind that there are various details apart from those laid out in this presentation, and I skip them for your mental health and for a clearer understanding of the core.

- Machine Learning Basics
- k-Nearest Neighbors
- Additive Models
- Decision Trees
- Bagging and Boosting

Supervised Algorithms: Basics

Let us have a quick flashback of the theoretical foundations of how supervised algorithms learned, based on the model seen in statistical learning theory:

- The capability of machines to learn from data is fundamentally based on the theory of statistical learning, which establishes useful results to understand how we learn in the context of problems such as regression and classification.

- Assuming an appropriate loss function L (which represents a loss or the cost of something), we can see that the regression and classification problems can be represented as:

$$\min_f E[L(y, f(\mathbf{z}))|\mathbf{x} = \mathbf{z}] \quad \text{(regression)}$$

$$\min_g E[L(y, g(\mathbf{z}))|\mathbf{x} = \mathbf{z}] \quad \text{(classification)}$$

Supervised Algorithms: Basics

- For certain loss functions, the optimal function f will be the expected value $E(y|\mathbf{x})$ and the optimal g will be $E(y = k|\mathbf{x}) = P(y = k|\mathbf{x})$, where k is a category. We used this in the standard econometric models we discussed!

- However, we could be using other loss functions, and the optimal functions we would want to model will correspond to other moments or other functions/quantities we want to model. The important thing is that, in most economic applications, **we are trying to model the same** from both econometrics and ML & AI.

Supervised Algorithms: Basics

- Nevertheless, we would want to forecast using these new methods, and consequently, we must care about how we can estimate a model that is **generalizable** and can learn from new data at the same time without hurting this capability of being used for the broad population.
- In this setting, we are not interested in using as many observations as we have in our dataset because that would mean we are adapting our estimated function $\hat{f}$ to our current data, and we might be capturing unique or random patterns (noise) that might not be present in other samples (the problem of **overfitting**).
- It is also important to understand that the complexity of our model is related to the probability that we are overfitting, as more complex models can adapt to weirder and weirder patterns. This is why sometimes we **regularize** our model to reduce complexity.

Supervised Algorithms: Basics

- To avoid overfitting, what we do is divide the sample into two differentiated sets of observations: the **training set** and the **test set**. The training set is used to estimate the parameters of or to obtain the desired estimator, while the test set is used for testing whether this estimator has a good performance when forecasting observations it has not seen.

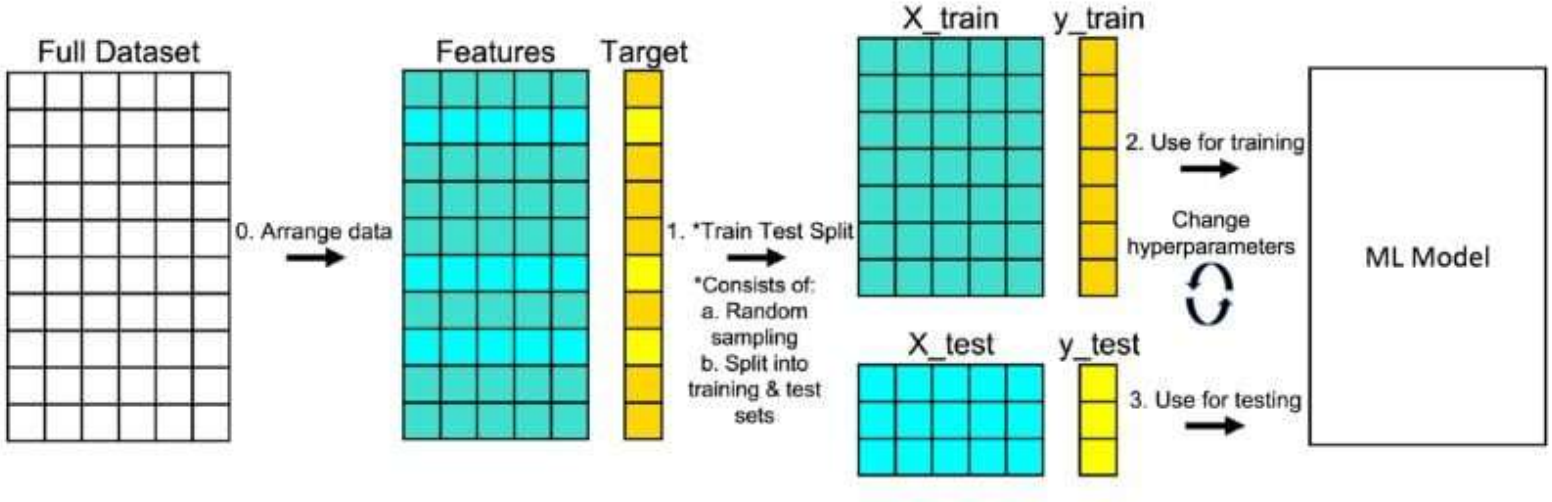


Figure: Train set and test set usage.

Supervised Algorithms: Basics

- In forecasting, we want our estimators to have **low variance** (we do not want very different ones for different samples) but to have a **low bias** also (we want to be as close as possible to the real value we want to forecast).
- The problem is that when we try to decrease bias, variance increases, and when we try to decrease variance, bias increases, so that there is a trade-off, known as the **bias-variance trade-off**.
- To see why this matters, we can consider the quadratic loss function and see how the error depends on both bias and variance:

$$E[(y - \hat{f}(\mathbf{x}_0))^2 | \mathbf{x} = \mathbf{x}_0] = \sigma_\varepsilon^2 + Bias^2[\hat{f}(\mathbf{x}_0)] + Var[\hat{f}(\mathbf{x}_0)]$$

Supervised Algorithms: Basics

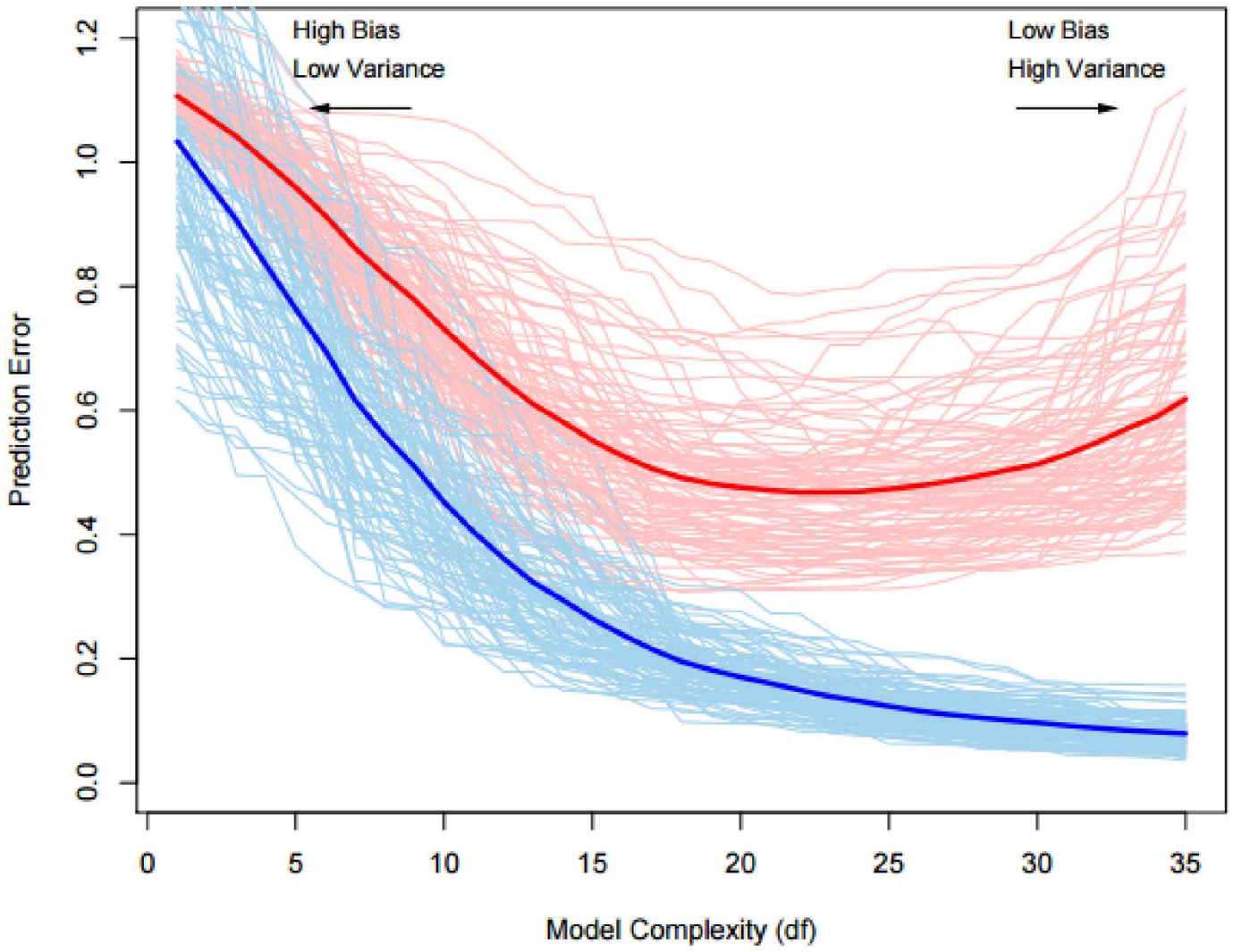


Figure: Train set and test set prediction error behavior.

Supervised Algorithms: Machine Learning Basics

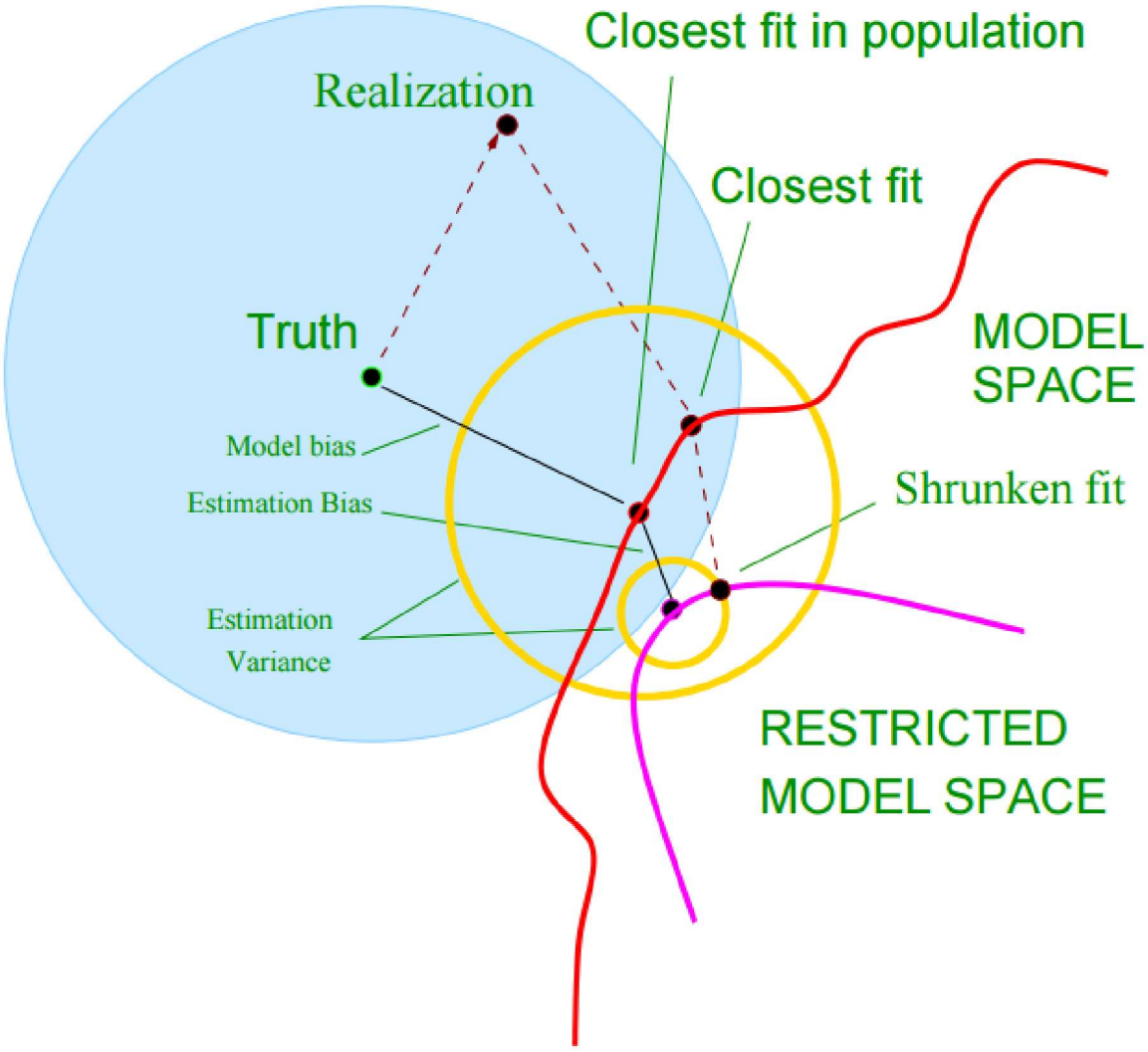


Figure: Schematic bias-variance behavior.

Supervised Algorithms: k-Nearest Neighbors

- k-Nearest Neighbours (k-NN) is a simple, non-parametric algorithm used for regression and classification.
  - In regression, k-NN predicts the value of a data point based on the average (or weighted average) of its k closest neighbors.
  - In classification, k-NN predicts the label of a data point based on the labels of its k closest neighbors.
- How it works:
  - Select a value for k (number of neighbors to consider).
  - Find the k data points closest to the target point.
  - Calculate the average of the values of these k neighbors to predict the target's value.
- **Pros**: Simple, intuitive, and works well with small datasets.
- **Cons**: Computationally expensive with large datasets, sensitive to the choice of k and outliers.

Supervised Algorithms: k-Nearest Neighbors

The k-NN estimator for regression and classification problems is:

$$\hat{y}(\mathbf{x}) = \frac{1}{k} \sum_{x_i \in N_k(\mathbf{x})} y_i$$

where y is continuous or a label and $N_k(\mathbf{x})$ is the neighbourhood of $\mathbf{x}$ defined by the k closest points $\mathbf{x}_i$ in the training sample.

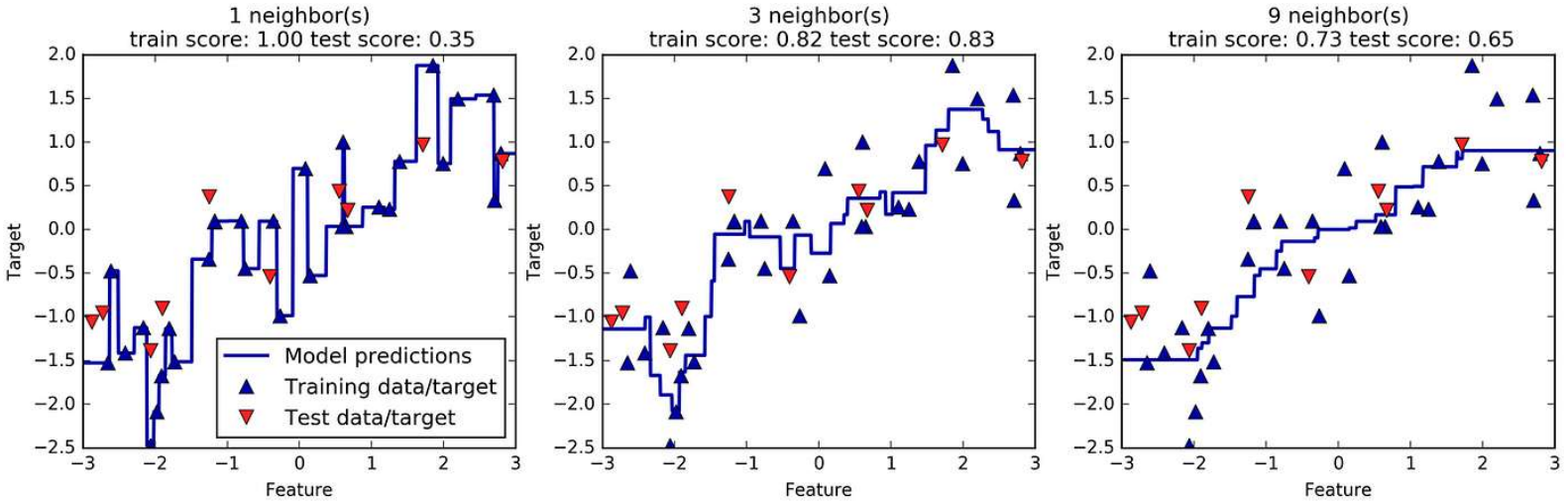


Figure: k-NN results for different k values.

Supervised Algorithms: Additive Models

- General Additive Models (GAMs) are a type of statistical model used to capture non-linear relationships in data by combining multiple functions of individual variables.
  - They are often applied in regression contexts to improve flexibility over linear models without becoming too complex.
- The general form of a General Additive Model is:

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i \quad \text{for} \quad i = 1, 2, ..., n$$

  where f_j for $j = 1, 2, ..., p$ are smooth functions estimated by the algorithm applied to each predictor x_j and ε is the error term.
- **Pros**: Flexible, interpretable, and allows to capture non-linearity.
- **Cons**: Can be computationally intensive with large datasets and may overfit if the functions are too flexible.

Supervised Algorithms: Additive Models

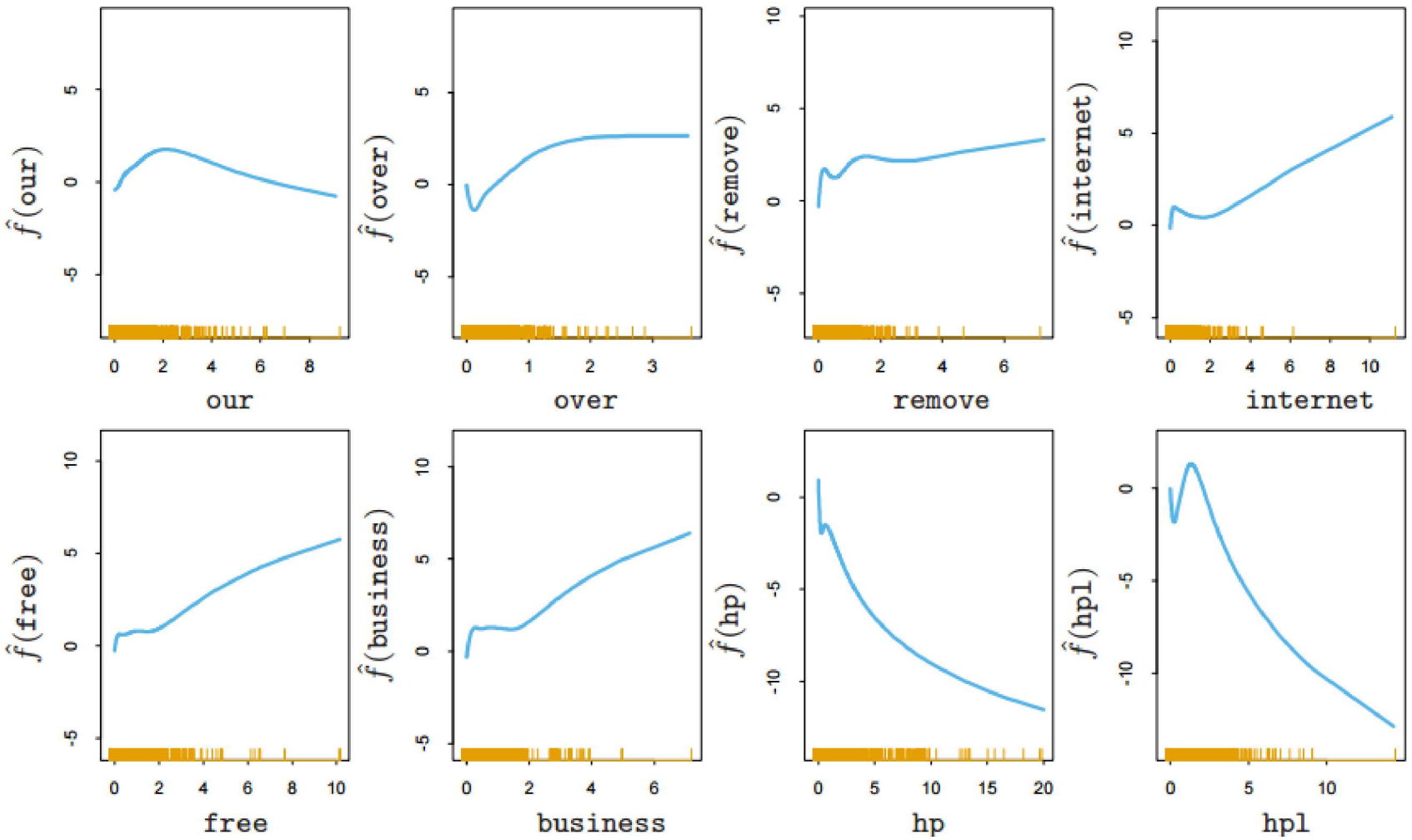


Figure: Function plots for the different variables.

Supervised Algorithms: Additive Models

Name	Num.	df	Coefficient	Std. Error	Z Score	Nonlinear P-value
Positive effects						
our	5	3.9	0.566	0.114	4.970	0.052
over	6	3.9	0.244	0.195	1.249	0.004
remove	7	4.0	0.949	0.183	5.201	0.093
internet	8	4.0	0.524	0.176	2.974	0.028
free	16	3.9	0.507	0.127	4.010	0.065
business	17	3.8	0.779	0.186	4.179	0.194
hpl	26	3.8	0.045	0.250	0.181	0.002
ch!	52	4.0	0.674	0.128	5.283	0.164
ch$	53	3.9	1.419	0.280	5.062	0.354
CAPMAX	56	3.8	0.247	0.228	1.080	0.000
CAPTOT	57	4.0	0.755	0.165	4.566	0.063
Negative effects						
hp	25	3.9	−1.404	0.224	−6.262	0.140
george	27	3.7	−5.003	0.744	−6.722	0.045
1999	37	3.8	−0.672	0.191	−3.512	0.011
re	45	3.9	−0.620	0.133	−4.649	0.597
edu	46	4.0	−1.183	0.209	−5.647	0.000

Figure: Table of results of the estimation of a GAM.

Supervised Algorithms: Decision Trees

- Decision trees are supervised learning algorithms used for both regression and classification tasks. They split the data into subsets based on feature values, creating a tree-like structure where each node represents a decision based on an attribute.
- How It Works:
  - The algorithm recursively splits the dataset at each node based on the best feature, aiming to maximize "purity" (classification) or "homogeneity" (regression) within the subsets.
  - The algorithm greedily evaluates all conditions and scores them, so that it stays with the split that divides data in different homogeneous regions.
- **Pros**: Easy to interpret, requires little data preprocessing and handles both categorical and continuous data.
- **Cons**: Prone to overfitting, sensitive to small variations in data, and can be less accurate than ensemble methods.

Supervised Algorithms: Decision Trees

- Classification Trees:
  - Common criteria:

$$\text{Gini Impurity} = 1 - \sum_{i=1}^{C} p_i^2 \qquad \text{Entropy} = -\sum_{i=1}^{C} p_i \log(p_i)$$

    where p_i is the probability of class i in the node.
  - The algorithm selects splits that minimize Gini or Entropy, leading to "purer" nodes.
- Regression Trees:
  - Aim to minimize the variance within each node.
  - Common criterion:

$$\text{Mean Squared Error (MSE)} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y})^2$$

  - The algorithm selects splits that minimize MSE, creating nodes with more homogeneous values.

Supervised Algorithms: Decision Trees

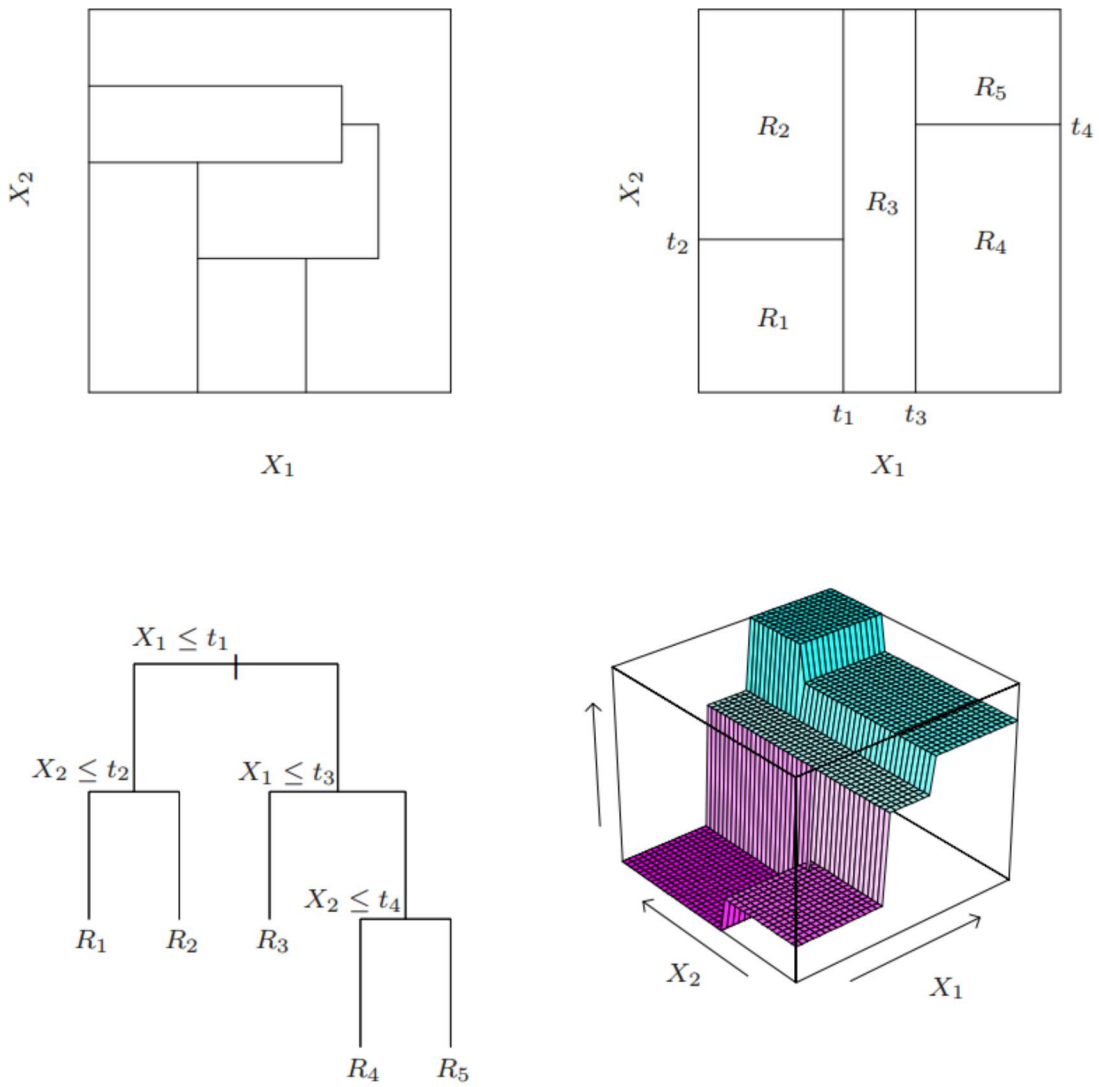


Figure: Functioning of decision trees.

Supervised Algorithms: Bagging and Boosting

- Bagging is an ensemble method that builds multiple models independently using different random subsets of the data.
  - Generate multiple datasets by sampling with replacement (bootstrap samples) from the original dataset.
  - Train a model on each bootstrap sample.
  - Combine predictions from each model:

$$\hat{y}_{\text{final}} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m$$

    where M is the number of models, and $\hat{y}_m$ is the prediction from model m.

- Applications and Benefits:
  - Bagging reduces variance and is particularly useful for high-variance models, like decision trees.
  - Commonly used in methods like Random Forests.

Supervised Algorithms: Bagging and Boosting

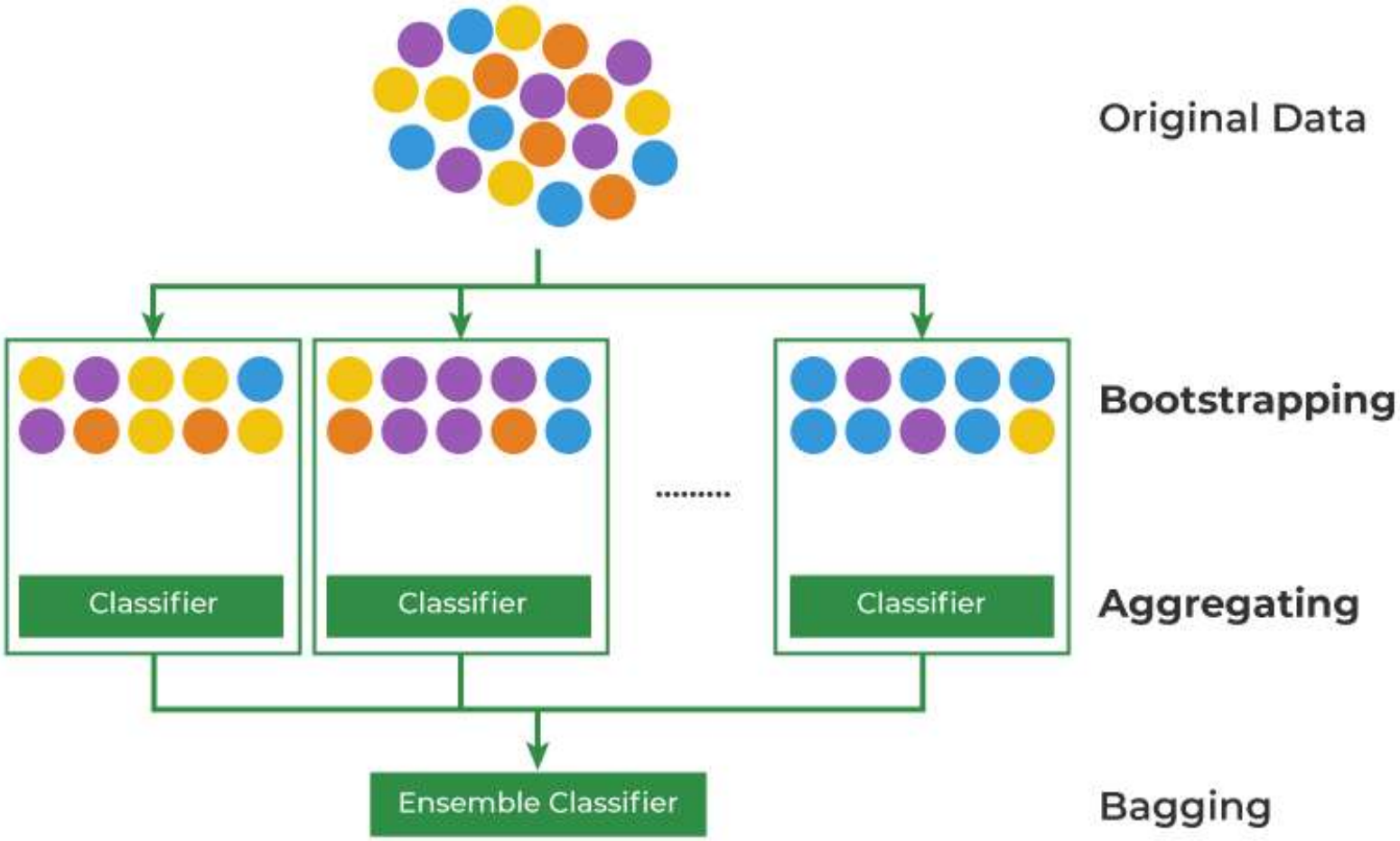


Figure: Bagging schematic functioning.

Supervised Algorithms: Bagging and Boosting

- Boosting is an ensemble technique that builds models sequentially, with each model focusing on correcting errors from the previous ones. It reduces bias by gradually improving the overall model through these corrections.
  - Start with an initial model and calculate its errors.
  - Train the next model to focus on misclassified samples or areas with high error.
  - Final prediction is a weighted sum of the individual model predictions:

$$\hat{y}_{\text{final}} = \sum_{m=1}^{M} \alpha_m \cdot \hat{y}_m$$

    where α_m is the weight assigned to model m based on its accuracy.

- **Pros**: Improves accuracy, reduces bias, and works well with weak models.
- **Cons**: Can overfit on noisy datasets, computationally intensive due to sequential training.

Supervised Algorithms: Bagging and Boosting

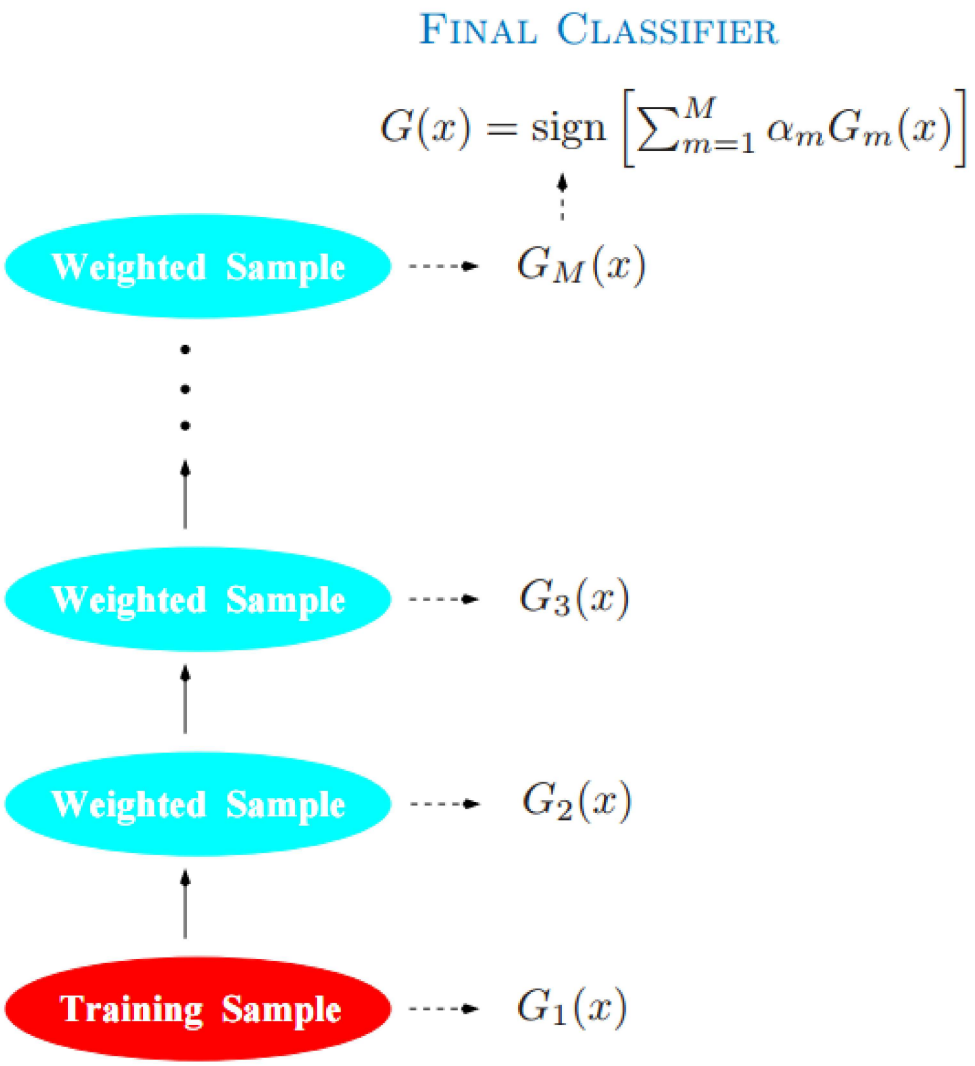


Figure: AdaBoost Boosting algorithm schematic functioning.

Neural Networks

- Neural Networks are computational models inspired by the human brain. They are designed to recognize patterns in data and learn from examples.
  - Neural networks are used in tasks like image recognition, speech processing, and predictive modeling.
- Basic Structure:
  - A neural network consists of interconnected nodes or "neurons" organized in layers.
  - Each layer processes information and passes it to the next layer.
- Why are they widely used?
  - Neural networks learn patterns from data without explicit programming.
  - They can generalize from examples to accurately predict new data.

Neural Networks: Structure

- Layers of a simple Neural Network:
  - **Input Layer**: Takes in data features as input.
  - **Hidden Layers**: Process information and extract patterns.
  - **Output Layer**: Produces the final prediction.
- Connections in Neural Networks:
  - Each neuron is connected to neurons in the next layer through "weights" (w), which determine the importance of each input.
- Formula for a single neuron:

$$\mathbf{h}^{(l)} = g^{(l)}\left((\mathbf{W}^{(l)})^T\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \quad \text{for} \quad l = 1, 2, ..., n$$

where $\mathbf{h}$ is the units vector, $\mathbf{W}$ is the weights matrix, $\mathbf{b}$ is the bias vector, each depending on the layer l, and the units are transformations of the inputs $\mathbf{x}$.

Neural Networks: Structure

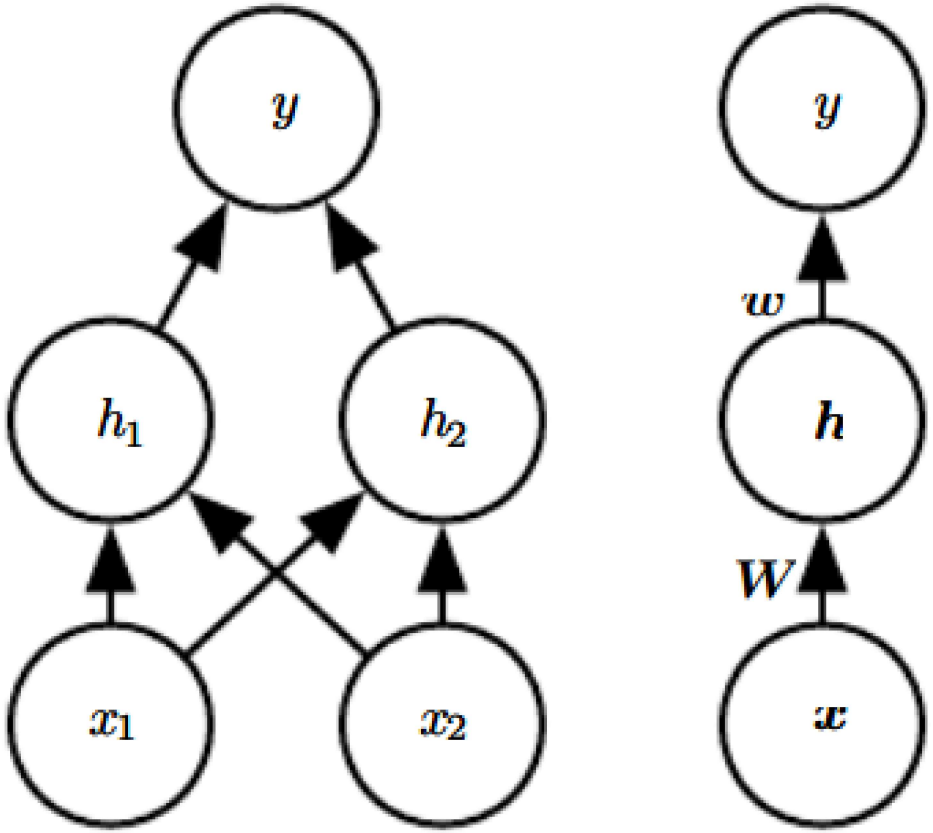


Figure: Schematic overview of a simple Neural Network.

Neural Networks: Activation Functions

- One of the most important elements of neural networks is its activation functions
  - Activation functions add non-linearity, allowing neural networks to learn complex patterns.
  - Activation functions help determine the final output of each neuron and, by extension, the entire network.
- Some common activation functions are:
  - **Sigmoid**: Maps output to a range between 0 and 1.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

  - **ReLU** (Rectified Linear Unit): Outputs positive values or 0.

$$\text{ReLU}(z) = \max(0, z)$$

Neural Networks: Activation Functions

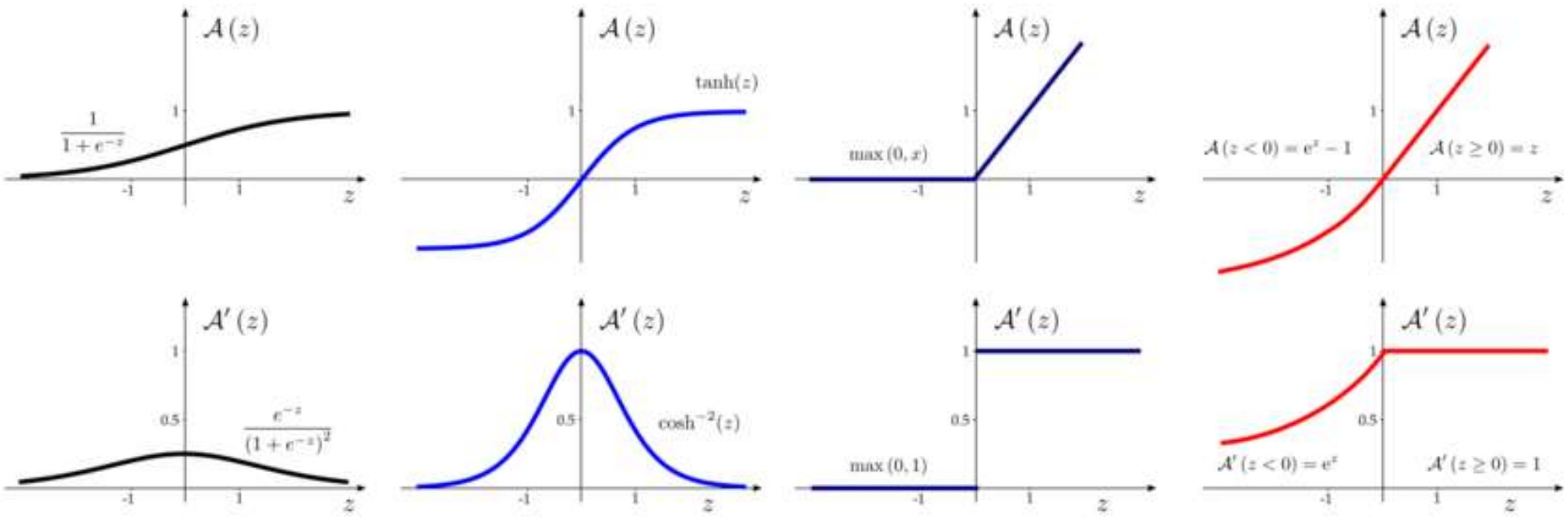


Figure: Graphs for different activation functions.

Neural Networks: Approximation

- One of the most important results in Deep Learning is the Universal Approximation Theorem, which states that a neural network with one hidden layer can approximate any continuous function, given enough neurons.
  - This theorem shows the power of neural networks to model complex relationships in data.
- How can it learn such complex patterns?
  - It does it in a similar way as we did with other models: we adjust our neural network to be close enough to the results of the desired variables in the dataset, but we do so in an iterative way.
  - By composing various non-linear functions, the activation functions allow us to learn representations of the data that pass to the next neuron and convey some information (learn something).

Neural Networks: Learning

- How do we fit a Neural Network? Just like other models, but with slight differences.
  - During training, the network adjusts its weights to minimize prediction errors. We do so through the backpropagation algorithm, which calculates errors and updates weights to improve accuracy.
  - The algorithm's idea is to pass forward the data to generate an arbitrary prediction, computing errors and propagating them backward layer by layer.
  - Finally, we adjust the weights and biases based on the gradients of the errors for each layer (through gradient-descent).
  - The backpropagation algorithm is the part that allows a network to "learn" from data.
- But how do Neural Networks behave exactly?
  - We know the mathematics of Neural Networks, so we know how they work, but we do not know how they behave!

Neural Networks: Learning

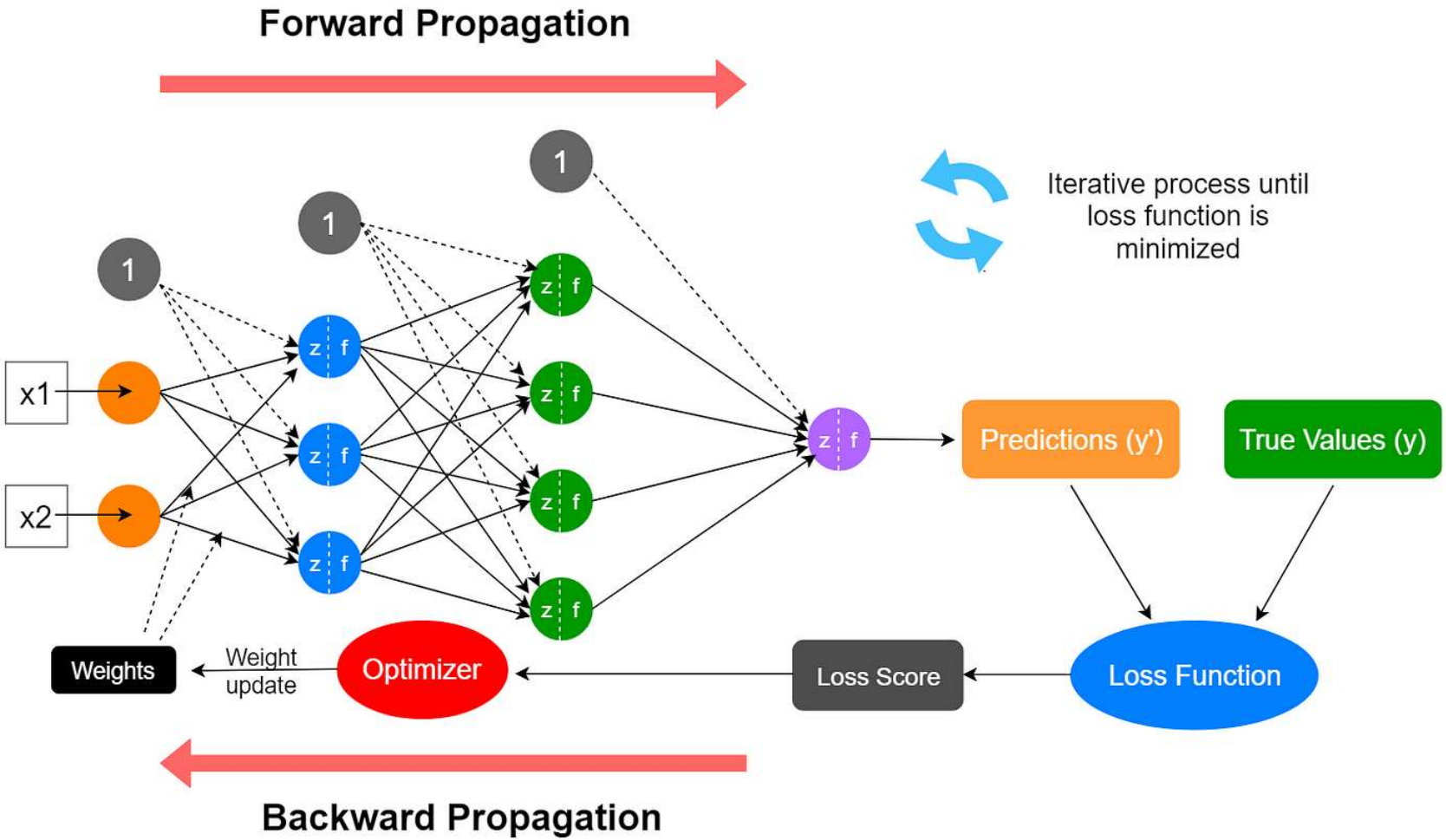
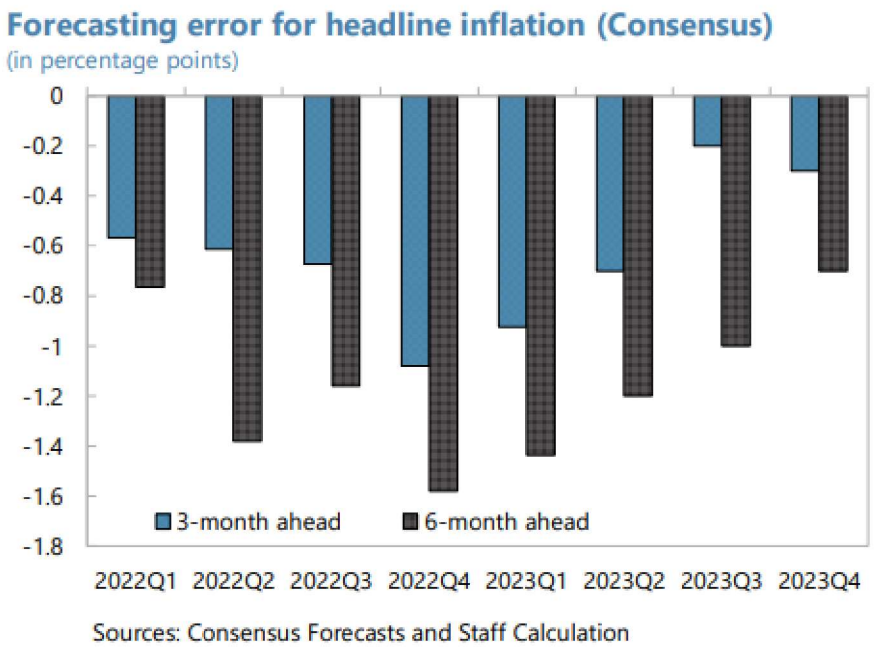


Figure: Backpropagation schematic functioning.

Application: Inflation Forecasting

- **Objectives of the paper**:
  - To improve inflation forecasting in Japan by applying machine learning models.
  - Compare machine learning models with traditional benchmarks in handling Japan's post-pandemic inflation.
- **Challenges in Traditional Forecasting**:
  - Traditional models faced difficulties with the new nonlinear and complex inflation dynamics.
  - Japan's post-2022 inflation required models that could adapt to non-linear patterns.

Application: Inflation Forecasting



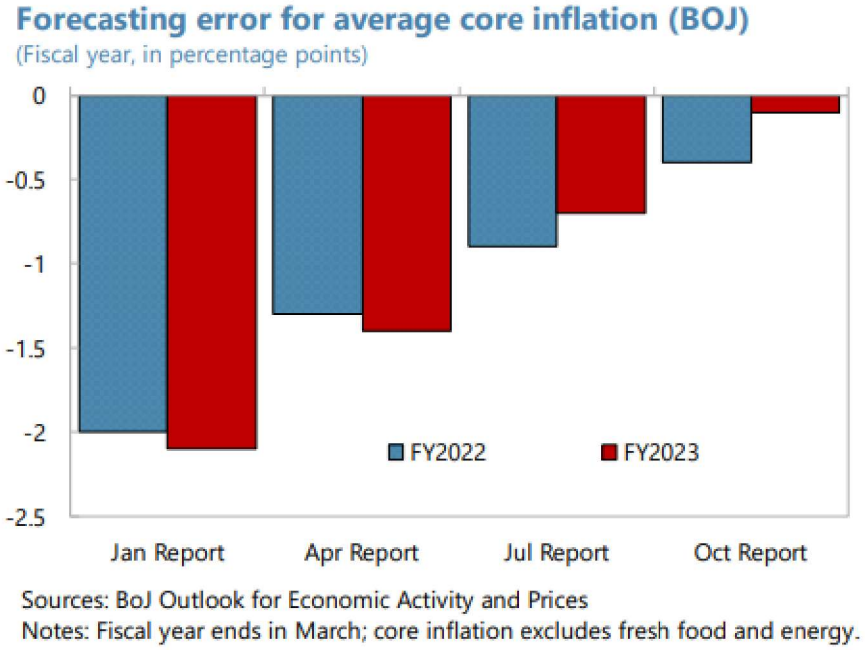


Figure: Inflation Forecasting Errors by Professional Forecasters

Application: Inflation Forecasting

Even though the authors also propose other models, let's talk about the random forest model.

- **Random Forest Overview**:
  - Ensemble method that builds multiple decision trees on random samples of data.
  - Aggregates outputs from all trees for final prediction, helping reduce overfitting common in single decision trees.
- **Advantages for Inflation Forecasting**:
  - Handles complex relationships between predictors without needing pre-specified equations.
  - Can estimate the importance of variables, which helps identify key inflation drivers.
- **Variables Used**:
  - Variables include consumer inflation expectations, exchange rates, GDP growth, and output gap.

Application: Inflation Forecasting

- **Performance of Random Forest**:
  - Although effective in capturing non-linear relationships, Random Forest had higher errors (RMSE) compared to simpler models like LASSO (regularized regression model) or an AR(1) (autoregressive time series model).
  - Random Forest's complex structure led to overfitting, especially given Japan's traditionally stable inflation environment.
- **Key Findings**:
  - Random Forest successfully identified key inflation drivers, including household inflation expectations and the output gap.
  - However, simpler models outperformed Random Forest by avoiding the overfitting seen in more complex ML models.

Application: Inflation Forecasting

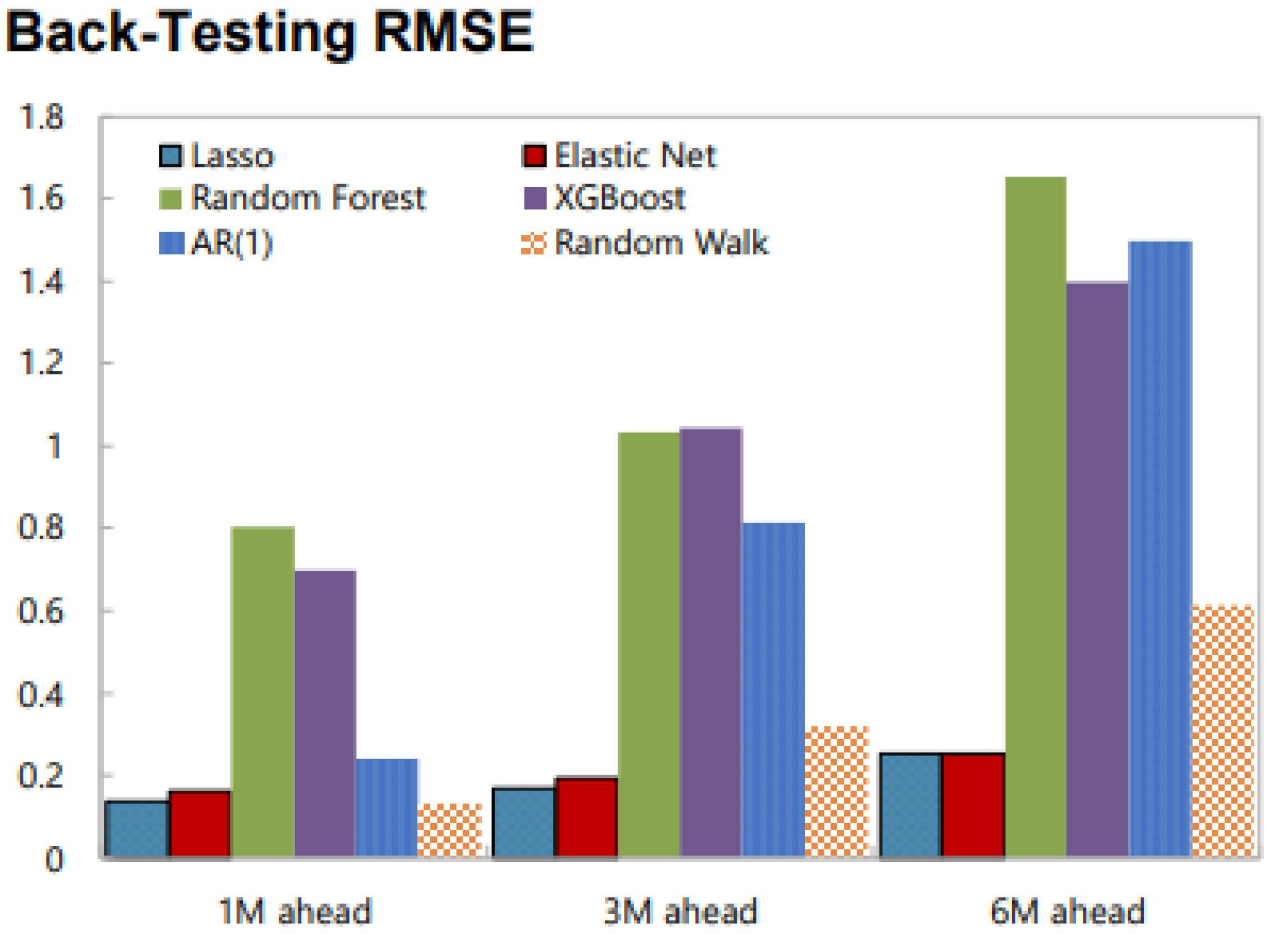


Figure: Back-testing Root Mean Square Errors for each model and forecast interval.