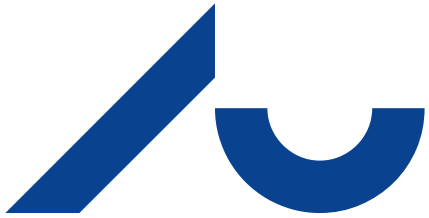


AARHUS
UNIVERSITY

Class 4: Learning From Data

Topic 1: ML Basics

Computational Analysis of Text, Audio, and Images, Fall 2023

Aarhus University

Mathias Rask (mathiasrask@ps.au.dk)

Aarhus University

Today's Menu

Introduction

Today's Menu

Introduction

Supervised Learning

Today's Menu

Introduction

Supervised Learning

Unsupervised Learning

Today's Menu

Introduction

Supervised Learning

Unsupervised Learning

Neural Nets

Table of Contents

Why Do We Need *Machine* Learning?

Why Do We Need *Machine* Learning?

Imagine we want to study how Danish news media frames "deservingness":

1. Random or intentional sampling of outlets

Why Do We Need *Machine* Learning?

Imagine we want to study how Danish news media frames "deservingness":

1. Random or intentional sampling of outlets
2. Manually pick out articles that contain news on social benefits

Why Do We Need *Machine* Learning?

Imagine we want to study how Danish news media frames "deservingness":

1. Random or intentional sampling of outlets
2. Manually pick out articles that contain news on social benefits
3. Create a coding scheme that, in detail, specifies how deservingness is framed as either 'deserving' or 'not deserving'

Why Do We Need *Machine* Learning?

Imagine we want to study how Danish news media frames "deservingness":

1. Random or intentional sampling of outlets
2. Manually pick out articles that contain news on social benefits
3. Create a coding scheme that, in detail, specifies how deservingness is framed as either 'deserving' or 'not deserving'
4. Manually read through articles to classify articles into the categories defined by the coding scheme

Example of Categories (Esmark and Schoop, 2017)

Table 1. Applied frames and frame dimensions: deserving and undeserving.

	Lack of incentive		Amoral behaviour	Lack of jobs	Lack of qualifications	Marginalization
Definition of problem	Lack of incentive to *work*	Lack of incentive to pursue further *education*	Lack of moral, disregard for the community	Lack of jobs	Skill set does not match current demands	Social and/ or economic marginalization
Causes	Economic structures, rational economic behaviour		Degradation of values, breakdown of norms	Economic cycles, insufficient demand and job creation	Innovation and technological development	Poverty, social, psychological and/or physical challenges
Responsibility and moral judgement	Collective responsibility for the state of the workforce Moral judgement of 'political will' to act on economic necessities		Individual responsibility for unemployment Moral judgement of individual attributes (laziness, etc.)	Collective responsibility for unemployment Moral judgement of society (solidarity, cohesiveness, etc.)	Collective responsibility for unemployment Moral judgement of society (solidarity, cohesiveness, etc.)	Collective responsibility for unemployment Moral judgement of society (solidarity, cohesiveness, etc.)
Solutions (logic of social benefits)	Reduction of benefits in order to increase incentives		Reduction of benefits as justice (punishment)	High level of social benefits as temporary compensation	High level of social benefits as means to upgrade qualifications (specific or general)	High level of social benefits reduces marginalization
	↓ Undeserving			↓ Deserving		

ML in Social Science

ML in Social Science

Social scientists are very often interested in categorizing data into

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories
  - Which topics do parties discuss in party programs?

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories
  - Which topics do parties discuss in party programs?
- Latent classes

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories
  - ▷ Which topics do parties discuss in party programs?
- Latent classes
  - Do state leaders use a “peacemaking” or “hostile“ tone towards Russia in the UN?

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories
  - ▷ Which topics do parties discuss in party programs?
- Latent classes
  - Do state leaders use a “peacemaking” or “hostile“ tone towards Russia in the UN?
  - Hate-speech detection on social media

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories
  - ▷ Which topics do parties discuss in party programs?
- Latent classes
  - Do state leaders use a “peacemaking” or “hostile“ tone towards Russia in the UN?
  - Hate-speech detection on social media
- Latent positions

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories
  - ▷ Which topics do parties discuss in party programs?
- Latent classes
  - Do state leaders use a “peacemaking” or “hostile“ tone towards Russia in the UN?
  - Hate-speech detection on social media
- Latent positions
  - Policy positions on economic vs. social dimension

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories
  - ▷ Which topics do parties discuss in party programs?
- Latent classes
  - Do state leaders use a “peacemaking” or “hostile“ tone towards Russia in the UN?
  - Hate-speech detection on social media
- Latent positions
  - Policy positions on economic vs. social dimension
  - Inter- and intra-party differences
  - . . .

ML in Social Science

Social scientists are very often interested in categorizing data into

- Content categories
  - ▷ Which topics do parties discuss in party programs?
- Latent classes
  - Do state leaders use a “peacemaking” or “hostile“ tone towards Russia in the UN?
  - Hate-speech detection on social media
- Latent positions
  - Policy positions on economic vs. social dimension
  - Inter- and intra-party differences
  - . . .

Terms

Terms

ML has its own vocab compared to econometrics, stats, and computer science

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y
- Unsupervised learning:

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y
- Unsupervised learning:We know X but *not* Y

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y
- Unsupervised learning:We know X but *not* Y
- Transfer learning:

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y
- Unsupervised learning:We know X but *not* Y
- Transfer learning:Take existing model h and use on $\mathcal{D}$

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y
- Unsupervised learning:We know X but *not* Y
- Transfer learning:Take existing model h and use on $\mathcal{D}$
    - ▷ Tuning: Fine-tune h and apply on $\mathcal{D}$

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y
- Unsupervised learning:We know X but *not* Y
- Transfer learning:Take existing model h and use on $\mathcal{D}$
  - Tuning: Fine-tune h and apply on $\mathcal{D}$
  - Pretraining: Apply h on $\mathcal{D}$ without any fine-tuning

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y
- Unsupervised learning:We know X but *not* Y
- Transfer learning:Take existing model h and use on $\mathcal{D}$
  - Tuning: Fine-tune h and apply on $\mathcal{D}$
  - Pretraining: Apply h on $\mathcal{D}$ without any fine-tuning
  - Extracting: Use h's learned feature representation of X on a new task

Terms

ML has its own vocab compared to econometrics, stats, and computer science

- Features: Input variables, the column names of the input data, or X
- Labels: Output variable, the column you are trying to predict, or Y
- Supervised learning:We know *both* X and Y
- Unsupervised learning:We know X but *not* Y
- Transfer learning:Take existing model h and use on $\mathcal{D}$
  - ▷ Tuning: Fine-tune h and apply on $\mathcal{D}$
  - ▷ Pretraining: Apply h on $\mathcal{D}$ without any fine-tuning
  - ▷ Extracting: Use h's learned feature representation of X on a new task
- A bunch of others: weakly-supervised, semi-supervised, self-supervised, ...

Goals

Goals

Machine learning is about *learning* from data using *machines*

Goals

Machine learning is about *learning* from data using *machines*

1. *How* do we learn? (i.e. type of learning and method)

Goals

Machine learning is about *learning* from data using *machines*

1. *How* do we learn? (i.e. type of learning and method)
2. Is learning *feasible*?

Goals

Machine learning is about *learning* from data using *machines*

1. *How* do we learn? (i.e. type of learning and method)
2. Is learning *feasible*?
    - Data quality (garbage in, garbage out)

Goals

Machine learning is about *learning* from data using *machines*

1. *How* do we learn? (i.e. type of learning and method)
2. Is learning *feasible*?
    - Data quality (garbage in, garbage out)
    - Data signal (complexity)

Goals

Machine learning is about *learning* from data using *machines*

1. *How* do we learn? (i.e. type of learning and method)
2. Is learning *feasible*?
    - Data quality (garbage in, garbage out)
    - Data signal (complexity)
3. *Why* do we learn? (prediction vs. measurement)

Goals

Machine learning is about *learning* from data using *machines*

1. *How* do we learn? (i.e. type of learning and method)
2. Is learning *feasible*?
    - Data quality (garbage in, garbage out)
    - Data signal (complexity)
3. *Why* do we learn? (prediction vs. measurement)
    - ▷ Our goal as social scientists is primarily measurement

Goals

Machine learning is about *learning* from data using *machines*

1. *How* do we learn? (i.e. type of learning and method)
2. Is learning *feasible*?
   - Data quality (garbage in, garbage out)
   - Data signal (complexity)
3. *Why* do we learn? (prediction vs. measurement)
   - ▷ Our goal as social scientists is primarily measurement
     → **But** prediction is a prerequisite

Goals

Machine learning is about *learning* from data using *machines*

1. *How* do we learn? (i.e. type of learning and method)
2. Is learning *feasible*?
   - Data quality (garbage in, garbage out)
   - Data signal (complexity)
3. *Why* do we learn? (prediction vs. measurement)
   - ▷ Our goal as social scientists is primarily measurement
     → **But** prediction is a prerequisite

⇝ The goal of this course is to get the tools to measure politically relevant concepts using machine learning tools

Table of Contents

Setup

Setup

- $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$

Setup

- $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$
- Classification: When Y is discrete/categorical
    - Binary vs. multi-class

Setup

- $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$
- Classification: When Y is discrete/categorical
  - Binary vs. multi-class
- Regression: When Y is continuous

Setup

- $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$
- Classification: When Y is discrete/categorical
  - Binary vs. multi-class
- Regression: When Y is continuous
  - When used to estimate positions in a latent space (e.g. the ideology of politicians), this is called scaling – more about this in 'Main Topic 2'

Setup

- $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$
- Classification: When Y is discrete/categorical
  - Binary vs. multi-class
- Regression: When Y is continuous
  - When used to estimate positions in a latent space (e.g. the ideology of politicians), this is called scaling – more about this in 'Main Topic 2'
- We want to learn the mapping $f : X \rightarrow Y$

Model Selection

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?
- Select h with the “best” performance

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?
- Select h with the "best" performance
  - Loss function: $\ell(Y', Y)$

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?
- Select h with the "best" performance
  - Loss function: $\ell(Y', Y)$
    - $(Y' - Y)^2$ (regression)

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?
- Select h with the "best" performance
  - Loss function: $\ell(Y', Y)$
    - $(Y' - Y)^2$ (regression)
    - $\mathbb{1}(Y' \neq Y)$ (classification)

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?
- Select h with the "best" performance
  - Loss function: $\ell(Y', Y)$
    - $(Y' - Y)^2$ (regression)
    - $\mathbb{1}(Y' \neq Y)$ (classification)
  - Empirical loss: $\hat{L}(h, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \ell(h(X_i), Y_i)$

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?
- Select h with the "best" performance
  - Loss function: $\ell(Y', Y)$
    - $(Y' - Y)^2$ (regression)
    - $\mathbb{1}(Y' \neq Y)$ (classification)
  - Empirical loss: $\hat{L}(h, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \ell(h(X_i), Y_i)$
  - Expected loss: $L(h) = \mathbb{E}[\ell(h(X), Y)]$

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?
- Select h with the "best" performance
  - Loss function: $\ell(Y', Y)$
    - $(Y' - Y)^2$ (regression)
    - $\mathbb{1}(Y' \neq Y)$ (classification)
  - Empirical loss: $\hat{L}(h, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \ell(h(X_i), Y_i)$
  - Expected loss: $L(h) = \mathbb{E}[\ell(h(X), Y)]$
  - The optimal model $h^*_{\mathcal{D}}$ is selected from $\mathcal{H}$ based on the empirical loss $\hat{L}(h, \mathcal{D})$

Model Selection

- Select model h with $h \in \mathcal{H}$ that approximates the unknown f
  - h: a model that learns how X is connected to Y
  - $\mathcal{H}$: Contains all possible models/solutions that connects X and Y
  - How do we select a single h based on the $|\mathcal{H}|$ models?
- Select h with the "best" performance
  - Loss function: $\ell(Y', Y)$
    - $(Y' - Y)^2$ (regression)
    - $\mathbb{1}(Y' \neq Y)$ (classification)
  - Empirical loss: $\hat{L}(h, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \ell(h(X_i), Y_i)$
  - Expected loss: $L(h) = \mathbb{E}[\ell(h(X), Y)]$
  - The optimal model $h^*_{\mathcal{D}}$ is selected from $\mathcal{H}$ based on the empirical loss $\hat{L}(h, \mathcal{D})$

    $\rightarrow$ In reality, the model is evaluated using performance metrics – more on those in a second!

Exercise

Discuss the following questions:

1. Is the empirical loss $\hat{L}(h^*_{\mathcal{D}}, \mathcal{D})$ an unbiased estimate of the expected loss $L(h^*)$? Why/why not?
2. How can we make sure that our empirical loss is an unbiased estimate of the expected loss?

Hints: Apply what you already know about sampling theory!

Exercise

Exercise

1. $\hat{L}(h^*_{\mathcal{D}}, \mathcal{D})$ is always a biased estimate of $L(h^*)$ since the optimal model $h^*_{\mathcal{D}}$ is selected based $\mathcal{D}$

Exercise

1. $\hat{L}(h^*_{\mathcal{D}}, \mathcal{D})$ is always a biased estimate of $L(h^*)$ since the optimal model $h^*_{\mathcal{D}}$ is selected based $\mathcal{D}$
2. Model h must be selected independently of $\mathcal{D}$

Exercise

1. $\hat{L}(h^*_{\mathcal{D}}, \mathcal{D})$ is always a biased estimate of $L(h^*)$ since the optimal model $h^*_{\mathcal{D}}$ is selected based $\mathcal{D}$
2. Model h must be selected independently of $\mathcal{D}$

   $\rightarrow$ Test sets to the rescue!

Generalization and Model Capacity

Generalization and Model Capacity

The ultimate goal of any model h is to perform well on previously unobserved input $X^{\texttt{new}}$ – i.e. generalization

Generalization and Model Capacity

The ultimate goal of any model h is to perform well on previously unobserved input $X^{\texttt{new}}$ – i.e. generalization

$$\text{generalization error} = g_{error} = |E_{in}(h) - E_{out}(h)|$$

Generalization and Model Capacity

The ultimate goal of any model h is to perform well on previously unobserved input $X^{\texttt{new}}$ – i.e. generalization

$$\text{generalization error} = g_{error} = |E_{in}(h) - E_{out}(h)|$$

g_{error} can be large for two reasons. How?

Generalization and Model Capacity

The ultimate goal of any model h is to perform well on previously unobserved input $X^{\texttt{new}}$ – i.e. generalization

$$\text{generalization error} = g_{error} = |E_{in}(h) - E_{out}(h)|$$

g_{error} can be large for two reasons. How?

1. Overfitting: When $E_{in}(h)$ is much smaller than $E_{out}(h)$

Generalization and Model Capacity

The ultimate goal of any model h is to perform well on previously unobserved input $X^{\texttt{new}}$ – i.e. generalization

$$\text{generalization error} = g_{error} = |E_{in}(h) - E_{out}(h)|$$

g_{error} can be large for two reasons. How?

1. Overfitting: When $E_{in}(h)$ is much smaller than $E_{out}(h)$
    - Biased selection

Generalization and Model Capacity

The ultimate goal of any model h is to perform well on previously unobserved input $X^{\texttt{new}}$ – i.e. generalization

$$\text{generalization error} = g_{error} = |E_{in}(h) - E_{out}(h)|$$

g_{error} can be large for two reasons. How?

1. Overfitting: When $E_{in}(h)$ is much smaller than $E_{out}(h)$
    - Biased selection
    - Capacity

Generalization and Model Capacity

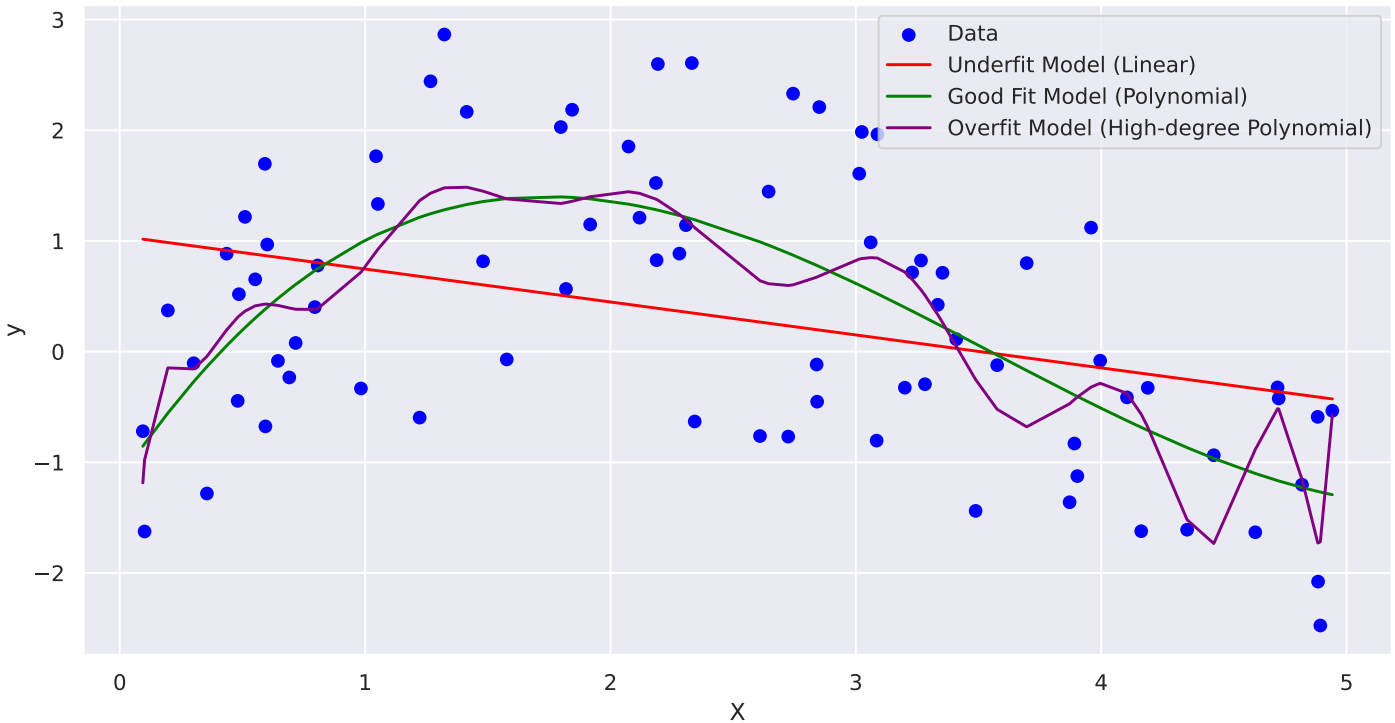
The ultimate goal of any model h is to perform well on previously unobserved input $X^{\texttt{new}}$ – i.e. generalization

$$\text{generalization error} = g_{error} = |E_{in}(h) - E_{out}(h)|$$

g_{error} can be large for two reasons. How?

1. Overfitting: When $E_{in}(h)$ is much smaller than $E_{out}(h)$
    - Biased selection
    - Capacity
2. Underfitting: When $E_{in}(h)$ is large

Illustration of Capacity



Training and Test Set

Training and Test Set

We can quantify the in- and out-of-sample errors using a training and a test set: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{test}\}$.

Training and Test Set

We can quantify the in- and out-of-sample errors using a training and a test set: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{test}\}$.

Assume we have $|\mathcal{H}| = 5$: $h_1, \ldots, h_i$ with $i = 5$

Training and Test Set

We can quantify the in- and out-of-sample errors using a training and a test set: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{test}\}$.

Assume we have $|\mathcal{H}| = 5$: $h_1, \ldots, h_i$ with $i = 5$

- Compute training losses: $\hat{L}(h_1, \mathcal{D}_{train}), \ldots, \hat{L}(h_5, \mathcal{D}_{train}) \rightarrow$ select h^* that minimizes the empirical loss = $E_{in}(h^*_{\mathcal{D}_{train}})$

Training and Test Set

We can quantify the in- and out-of-sample errors using a training and a test set: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{test}\}$.

Assume we have $|\mathcal{H}| = 5$: $h_1, \ldots, h_i$ with $i = 5$

- Compute training losses: $\hat{L}(h_1, \mathcal{D}_{train}), \ldots, \hat{L}(h_5, \mathcal{D}_{train}) \rightarrow$ select h^* that minimizes the empirical loss = $E_{in}(h^*_{\mathcal{D}_{train}})$
- Compute test loss: $\hat{L}(h^*_{\mathcal{D}_{train}}, \mathcal{D}_{test}) = E_{out}(h^*_{\mathcal{D}_{train}})$

Hyperparameters and Validation

Hyperparameters control the behavior of algorithms and are fixed external to the learning itself (e.g. regularization λ or polynomials).

Hyperparameters and Validation

Hyperparameters control the behavior of algorithms and are fixed external to the learning itself (e.g. regularization λ or polynomials).

Imagine we want to find the optimal regularization parameter λ for a Lasso regression (L1-regularization):

Hyperparameters and Validation

Hyperparameters control the behavior of algorithms and are fixed external to the learning itself (e.g. regularization λ or polynomials).

Imagine we want to find the optimal regularization parameter λ for a Lasso regression (L1-regularization):

- Split data: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test}\}$

Hyperparameters and Validation

Hyperparameters control the behavior of algorithms and are fixed external to the learning itself (e.g. regularization λ or polynomials).

Imagine we want to find the optimal regularization parameter λ for a Lasso regression (L1-regularization):

- Split data: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test}\}$
- Train $h_\lambda^1, h_\lambda^2, \ldots, h_\lambda^n$ on $\mathcal{D}_{train}$

Hyperparameters and Validation

Hyperparameters control the behavior of algorithms and are fixed external to the learning itself (e.g. regularization λ or polynomials).

Imagine we want to find the optimal regularization parameter λ for a Lasso regression (L1-regularization):

- Split data: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test}\}$
- Train $h_\lambda^1, h_\lambda^2, \ldots, h_\lambda^n$ on $\mathcal{D}_{train}$
- Compute validation losses: $\hat{L}(h_\lambda^1, \mathcal{D}_{val}), \ldots, \hat{L}(h_\lambda^n, \mathcal{D}_{val}) \rightarrow$ select h^* that minimizes the validation loss

Hyperparameters and Validation

Hyperparameters control the behavior of algorithms and are fixed external to the learning itself (e.g. regularization λ or polynomials).

Imagine we want to find the optimal regularization parameter λ for a Lasso regression (L1-regularization):

- Split data: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test}\}$
- Train $h_\lambda^1, h_\lambda^2, \ldots, h_\lambda^n$ on $\mathcal{D}_{train}$
- Compute validation losses: $\hat{L}(h_\lambda^1, \mathcal{D}_{val}), \ldots, \hat{L}(h_\lambda^n, \mathcal{D}_{val}) \rightarrow$ select h^* that minimizes the validation loss
- Compute test error $\hat{L}(h^*_{\mathcal{D}_{val}}, \mathcal{D}_{test})$ to get an unbiased estimate of $L(h^*_{\mathcal{D}_{val}})$

Hyperparameters and Validation

Hyperparameters control the behavior of algorithms and are fixed external to the learning itself (e.g. regularization λ or polynomials).

Imagine we want to find the optimal regularization parameter λ for a Lasso regression (L1-regularization):

- Split data: $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test}\}$
- Train $h_\lambda^1, h_\lambda^2, \ldots, h_\lambda^n$ on $\mathcal{D}_{train}$
- Compute validation losses: $\hat{L}(h_\lambda^1, \mathcal{D}_{val}), \ldots, \hat{L}(h_\lambda^n, \mathcal{D}_{val}) \rightarrow$ select h^* that minimizes the validation loss
- Compute test error $\hat{L}(h^*_{\mathcal{D}_{val}}, \mathcal{D}_{test})$ to get an unbiased estimate of $L(h^*_{\mathcal{D}_{val}})$

k-fold Cross-Validation

k-fold Cross-Validation

1. Split $\mathcal{D}$ into $\{\mathcal{D}_{train}, \mathcal{D}_{test}\}$
2. Split $\mathcal{D}_{train}$ into non-overlapping folds $\mathcal{D}^1_{train}, \mathcal{D}^2_{train}, \ldots, \mathcal{D}^k_{train}$
3. For each fold $\mathcal{D}^i_{train}$ for $i \in \{1, \ldots k\}$, we train on all folds except the i-th fold
4. Compute k losses, take the average, and select $h^*_{\mathcal{D}_{val}}$
5. Compute test error $\hat{L}(h^*_{\mathcal{D}_{val}}, \mathcal{D}_{test})$

	Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Split 1	Validation	Train	Train	Train	Train
Split 2	Train	Validation	Train	Train	Train
Split 3	Train	Train	Validation	Train	Train
Split 4	Train	Train	Train	Validation	Train
Split 5	Train	Train	Train	Train	Validation

Table 1: 5-Fold Cross-Validation

Metrics

Metrics

While loss functions are used in training, the actual model selection and evaluation are based on performance metrics. A widely used metric is the accuracy:

Metrics

While loss functions are used in training, the actual model selection and evaluation are based on performance metrics. A widely used metric is the accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Metrics

While loss functions are used in training, the actual model selection and evaluation are based on performance metrics. A widely used metric is the accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Bounded between 0 and 1 – can be seen as a proportion
- Interpretation: Share of correct predictions.

Metrics

While loss functions are used in training, the actual model selection and evaluation are based on performance metrics. A widely used metric is the accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- Bounded between 0 and 1 – can be seen as a proportion
- Interpretation: Share of correct predictions.
- In what cases does the accuracy fall short?

Metrics for Imbalanced Data

Metrics for Imbalanced Data

Accuracy is not very informative in case of imbalanced data. Instead, we can use precision and recall, or a combination:

Metrics for Imbalanced Data

Accuracy is not very informative in case of imbalanced data. Instead, we can use precision and recall, or a combination:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metrics for Imbalanced Data

Accuracy is not very informative in case of imbalanced data. Instead, we can use precision and recall, or a combination:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metrics for Imbalanced Data

Accuracy is not very informative in case of imbalanced data. Instead, we can use precision and recall, or a combination:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Supervised Algorithms

- Linear models (e.g. linear or logistic regression)
- Support Vector Machines (SVMs)
- Naive Bayes
- Trees
- Neural nets
- . . .

→ All have excellent support using scikit-learn

Estimating Probabilities From Classification

Estimating Probabilities From Classification

Classification deals with finding a function f that maps certain values of X to certain values of Y.

Estimating Probabilities From Classification

Classification deals with finding a function f that maps certain values of X to certain values of Y.

Probabilistic algorithms (e.g. logistic regression) do this by estimating the conditional probability $p(y \mid x)$

Estimating Probabilities From Classification

Classification deals with finding a function f that maps certain values of X to certain values of Y.

Probabilistic algorithms (e.g. logistic regression) do this by estimating the conditional probability $p(y \mid x)$

The output in these algorithms is probabilities, which we assign to class 1 if $p \geq \tau$ and class 0 otherwise.

Estimating Probabilities From Classification

Classification deals with finding a function f that maps certain values of X to certain values of Y.

Probabilistic algorithms (e.g. logistic regression) do this by estimating the conditional probability $p(y \mid x)$

The output in these algorithms is probabilities, which we assign to class 1 if $p \geq \tau$ and class 0 otherwise. The trick is to use the classification probabilities to generate scales!

Estimating Probabilities From Classification

Classification deals with finding a function f that maps certain values of X to certain values of Y.

Probabilistic algorithms (e.g. logistic regression) do this by estimating the conditional probability $p(y \mid x)$

The output in these algorithms is probabilities, which we assign to class 1 if $p \geq \tau$ and class 0 otherwise. The trick is to use the classification probabilities to generate scales!

Protest Dynamics (Steinert-Threlkeld et al. 2022)

Figure 1. *A*, Sample images and the protest classifier's rating of them. *B*, Protest images with their state violence rating. *C*, Protest images' protester violence rating. Labels specify each image's city and label probability. The use of hard negatives in the training set ensures that scenes that contain crowds (*A*, *left*), individuals walking on streets (*C*, *left*), or a nonprotest sign (*A*, *second*) are not included in analysis. Color version available as an online enhancement.

Polarization (Peterson and Spirling 2018)

PA

Classification Accuracy as a Substantive Quantity of Interest: Measuring Polarization in Westminster Systems

Andrew Peterson[1] and Arthur Spirling[2]

[1] *Postdoctoral Researcher, University of Geneva, Switzerland. Email: andrew.peterson@unige.ch*
[2] *Associate Professor of Politics and Data Science, New York University, USA. Email: arthur.spirling@nyu.edu*

Abstract

Measuring the polarization of legislators and parties is a key step in understanding how politics develops over time. But in parliamentary systems—where ideological positions estimated from roll calls may not be informative—producing valid estimates is extremely challenging. We suggest a new measurement strategy that makes innovative use of the "accuracy" of machine classifiers, i.e., the number of correct predictions made as a proportion of all predictions. In our case, the "labels" are the party identifications of the members of parliament, predicted from their speeches along with some information on debate subjects. Intuitively, when the learner is able to discriminate members in the two main Westminster parties well, we claim we are in a period of "high" polarization. By contrast, when the classifier has low accuracy—and makes a relatively large number of mistakes in terms of allocating members to parties based on the data—we argue parliament is in an era of "low" polarization. This approach is fast and substantively valid, and we demonstrate its merits with simulations, and by comparing the estimates from 78 years of House of Commons speeches with qualitative and quantitative historical accounts of the same. As a headline finding, we note that contemporary British politics is approximately as polarized as it was in the mid-1960s—that is, in the middle of the "postwar consensus". More broadly, we show that the technical performance of supervised learning algorithms can be directly informative about substantive matters in social science.

Keywords: statistical analysis of texts, polarization, learning

Table of Contents

Setup

Setup

- Start: $\mathcal{D} = \{X_1, \ldots, X_N\}$ where $X_i = \{x^1, x^2, \ldots, x^j\}'$

Setup

- Start: $\mathcal{D} = \{X_1, \ldots, X_N\}$ where $X_i = \{x^1, x^2, \ldots, x^j\}'$
- Two dominant approaches:

Setup

- Start: $\mathcal{D} = \{X_1, \ldots, X_N\}$ where $X_i = \{x^1, x^2, \ldots, x^j\}'$
- Two dominant approaches:
  - Clustering: $\tilde{\mathcal{D}} = \{X_1, \ldots, X_N\} \rightarrow \{(X_1, \tilde{Y}_1), \ldots, (X_N, \tilde{Y}_N)\}$

Setup

- Start: $\mathcal{D} = \{X_1, \ldots, X_N\}$ where $X_i = \{x^1, x^2, \ldots, x^j\}'$
- Two dominant approaches:
  - Clustering: $\tilde{\mathcal{D}} = \{X_1, \ldots, X_N\} \rightarrow \{(X_1, \tilde{Y}_1), \ldots, (X_N, \tilde{Y}_N)\}$
  - Dimensionality reduction: $\bar{\mathcal{D}} = \{\bar{X}_1, \ldots, \bar{X}_N\}$ where $\bar{X}_i = \{\bar{x}^1, \ldots, \bar{x}^k\}'$ with $k \ll j$

Setup

- Start: $\mathcal{D} = \{X_1, \ldots, X_N\}$ where $X_i = \{x^1, x^2, \ldots, x^j\}'$
- Two dominant approaches:
  - Clustering: $\tilde{\mathcal{D}} = \{X_1, \ldots, X_N\} \rightarrow \{(X_1, \tilde{Y}_1), \ldots, (X_N, \tilde{Y}_N)\}$
  - Dimensionality reduction: $\bar{\mathcal{D}} = \{\bar{X}_1, \ldots, \bar{X}_N\}$ where $\bar{X}_i = \{\bar{x}^1, \ldots, \bar{x}^k\}'$ with $k \ll j$

Recast $\mathcal{D}$ as a X_{ij} matrix with dimensions $N \times M$:

- i: Row index
- j: Column index
- x_i: Row i
- x_j: Column j

Why Cluster?

Why Cluster?

Clustering is the process of dividing observations in data into groups.
Used for two reasons:

Why Cluster?

Clustering is the process of dividing observations in data into groups.
Used for two reasons:

1. *Discovery*: We have a priori expectation that "groups" exist and we want to identify those
    - Intra-party unity using roll-call votes
    - Intra-party unity of voters using surveys
    - Identify speakers in an audio recording based on vocal characteristics
    - Weak labeling?

Why Cluster?

Clustering is the process of dividing observations in data into groups.
Used for two reasons:

1. *Discovery*: We have a priori expectation that "groups" exist and we want to identify those
    - Intra-party unity using roll-call votes
    - Intra-party unity of voters using surveys
    - Identify speakers in an audio recording based on vocal characteristics
    - Weak labeling?
2. *Exploration*: Getting to know your data

Discovery of Legislative Styles (Sulkin et al. 2017)

WILLIAM BERNHARD
University of Illinois
DANIEL SEWELL
University of Iowa
TRACY SULKIN
University of Illinois

A Clustering Approach to Legislative Styles

Once elected to office, members of Congress (MCs) face choices about how to prioritize their activities. What is the right balance between working in the district and on Capitol Hill? How active will they be in introducing and cosponsoring legislation? Will they give speeches and interviews? On which issues will they focus their attention? With whom will they collaborate and form coalitions? How much time will they spend raising money? To what extent will they toe the party line or chart their own course? Collectively, these decisions make up a member of Congress's "legislative style." Legislators' styles are central to congressional life, with the potential to shape their career trajectories, the representation they provide to constituents, and the nature of the policies passed by Congress.

In this article, we analyze legislative style, testing our theory about its correlates and consequences. We focus on the 1,049 legislators who served in the 101st–110th Congresses (1989–2008), gathering data on the activities of members of Congress, categorizing these into indices that reflect components of legislative style, and using longitudinal

k-means

k-means

Multiple algorithms, but *k*-means is the most widely used.

k-means

Multiple algorithms, but *k*-means is the most widely used.

Assumptions:

k-means

Multiple algorithms, but k-means is the most widely used.
Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:

k-means

Multiple algorithms, but k-means is the most widely used.

Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other

k-means

Multiple algorithms, but k-means is the most widely used.

Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other

k-means

Multiple algorithms, but k-means is the most widely used.

Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other
- $K < M$ and often $K \ll M$

k-means

Multiple algorithms, but k-means is the most widely used.

Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other
- $K < M$ and often $K \ll M$
- Single vs. multiple membership

k-means

Multiple algorithms, but k-means is the most widely used.

Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other
- $K < M$ and often $K \ll M$
- Single vs. multiple membership

$\rightarrow$ How do we choose K?

k-means

Multiple algorithms, but k-means is the most widely used.
Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other
- $K < M$ and often $K \ll M$
- Single vs. multiple membership

$\rightarrow$ How do we choose K?

Algorithm:

1. Randomly select K centroids $\mu_{k \in K}$

k-means

Multiple algorithms, but k-means is the most widely used.
Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other
- $K < M$ and often $K \ll M$
- Single vs. multiple membership

$\rightarrow$ How do we choose K?

Algorithm:

1. Randomly select K centroids $\mu_{k \in K}$
2. Assign x_i by finding the nearest centroid $\mu_{k \in K}$ (by minimizing the within-cluster-sum-of-squares)

k-means

Multiple algorithms, but k-means is the most widely used.
Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other
- $K < M$ and often $K \ll M$
- Single vs. multiple membership

$\rightarrow$ How do we choose K?

Algorithm:

1. Randomly select K centroids $\mu_{k \in K}$
2. Assign x_i by finding the nearest centroid $\mu_{k \in K}$ (by minimizing the within-cluster-sum-of-squares)
3. Re-calculate the centroids $\mu_{k \in K}$ by taking the mean of each cluster

k-means

Multiple algorithms, but k-means is the most widely used.
Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other
- $K < M$ and often $K \ll M$
- Single vs. multiple membership

$\rightarrow$ How do we choose K?

Algorithm:

1. Randomly select K centroids $\mu_{k \in K}$
2. Assign x_i by finding the nearest centroid $\mu_{k \in K}$ (by minimizing the within-cluster-sum-of-squares)
3. Re-calculate the centroids $\mu_{k \in K}$ by taking the mean of each cluster

$\rightarrow$ Terminate when x_i is not re-assigned

k-means

Multiple algorithms, but k-means is the most widely used.
Assumptions:

- Data contains $k = \{1, \ldots, K\}$ different groups where:
  1. Observations in cluster k is similar to each other
  2. Observations in different clusters are dissimilar to each other
- $K < M$ and often $K \ll M$
- Single vs. multiple membership

$\rightarrow$ How do we choose K?

Algorithm:

1. Randomly select K centroids $\mu_{k \in K}$
2. Assign x_i by finding the nearest centroid $\mu_{k \in K}$ (by minimizing the within-cluster-sum-of-squares)
3. Re-calculate the centroids $\mu_{k \in K}$ by taking the mean of each cluster

$\rightarrow$ Terminate when x_i is not re-assigned

Why Reduce?

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the “good” variation

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the “good” variation
2. Eliminate the “bad” variation

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the “good” variation
2. Eliminate the “bad” variation

$\rightarrow$ Result: A compressed data $\bar{X}_{ik}$ where $k \ll j$

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the “good” variation
2. Eliminate the “bad” variation

$\rightarrow$ Result: A compressed data $\bar{X}_{ik}$ where $k \ll j$

Widely used for:

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the “good” variation
2. Eliminate the “bad” variation

$\rightarrow$ Result: A compressed data $\bar{X}_{ik}$ where $k \ll j$

Widely used for:

- Feature selection

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the “good” variation
2. Eliminate the “bad” variation

$\rightarrow$ Result: A compressed data $\bar{X}_{ik}$ where $k \ll j$

Widely used for:

- Feature selection
- Latent space discovery

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the “good” variation
2. Eliminate the “bad” variation

$\rightarrow$ Result: A compressed data $\bar{X}_{ik}$ where $k \ll j$

Widely used for:

- Feature selection
- Latent space discovery
- Computational efficiency

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the "good" variation
2. Eliminate the "bad" variation

$\rightarrow$ Result: A compressed data $\bar{X}_{ik}$ where $k \ll j$

Widely used for:

- Feature selection
- Latent space discovery
- Computational efficiency

$\rightarrow$ A combination

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the "good" variation
2. Eliminate the "bad" variation

$\rightarrow$ Result: A compressed data $\bar{X}_{ik}$ where $k \ll j$

Widely used for:

- Feature selection
- Latent space discovery
- Computational efficiency

$\rightarrow$ A combination

Popular algorithms:

Why Reduce?

Dimensionality reduction aims to find a lower-dimensional representation of X_{ij} that:

1. Maintains the "good" variation
2. Eliminate the "bad" variation

$\rightarrow$ Result: A compressed data $\bar{X}_{ik}$ where $k \ll j$

Widely used for:

- Feature selection
- Latent space discovery
- Computational efficiency

$\rightarrow$ A combination

Popular algorithms:

- PCA
- t-SNE
- UMAP

PCA (Principal Components Analysis)

PCA (Principal Components Analysis)

PCA learns a lower-dimensional of X_{ij}, returning $\bar{X}_{ik}$ with $k \ll j$ where each column k is a principal component.

PCA (Principal Components Analysis)

PCA learns a lower-dimensional of X_{ij}, returning $\bar{X}_{ik}$ with $k \ll j$ where each column k is a principal component. Each component $k = \{1, \ldots K\}$ has no linear correlation (= orthogonal) with each other while preserving as much variance as possible:

PCA (Principal Components Analysis)

PCA learns a lower-dimensional of X_{ij}, returning $\bar{X}_{ik}$ with $k \ll j$ where each column k is a principal component. Each component $k = \{1, \ldots K\}$ has no linear correlation (= orthogonal) with each other while preserving as much variance as possible:

- PC1 maps X_{ij} onto an axis that contains the highest variance

PCA (Principal Components Analysis)

PCA learns a lower-dimensional of X_{ij}, returning $\bar{X}_{ik}$ with $k \ll j$ where each column k is a principal component. Each component $k = \{1, \ldots K\}$ has no linear correlation (= orthogonal) with each other while preserving as much variance as possible:

- PC1 maps X_{ij} onto an axis that contains the highest variance
- PC2 maps X_{ij} onto an axis orthogonal to PC1 and that contains the highest amount of leftover variance.

PCA (Principal Components Analysis)

PCA learns a lower-dimensional of X_{ij}, returning $\bar{X}_{ik}$ with $k \ll j$ where each column k is a principal component. Each component $k = \{1, \dots K\}$ has no linear correlation (= orthogonal) with each other while preserving as much variance as possible:

- PC1 maps X_{ij} onto an axis that contains the highest variance
- PC2 maps X_{ij} onto an axis orthogonal to PC1 and that contains the highest amount of leftover variance.

$\rightsquigarrow$ We maximize variance

PCA (Principal Components Analysis)

PCA learns a lower-dimensional of X_{ij}, returning $\bar{X}_{ik}$ with $k \ll j$ where each column k is a principal component. Each component $k = \{1, \dots K\}$ has no linear correlation (= orthogonal) with each other while preserving as much variance as possible:

- PC1 maps X_{ij} onto an axis that contains the highest variance
- PC2 maps X_{ij} onto an axis orthogonal to PC1 and that contains the highest amount of leftover variance.

$\rightsquigarrow$ We maximize variance

Interpretation:

PCA (Principal Components Analysis)

PCA learns a lower-dimensional of X_{ij}, returning $\bar{X}_{ik}$ with $k \ll j$ where each column k is a principal component. Each component $k = \{1, \dots K\}$ has no linear correlation (= orthogonal) with each other while preserving as much variance as possible:

- PC1 maps X_{ij} onto an axis that contains the highest variance
- PC2 maps X_{ij} onto an axis orthogonal to PC1 and that contains the highest amount of leftover variance.

$\rightsquigarrow$ We maximize variance

Interpretation:

- Sometimes straightforward, sometimes difficult
- Intuition: PC1 maps the data into the latent dimension along which observations vary the most $\rightsquigarrow$ Data specific!

PCA (Principal Components Analysis)

PCA learns a lower-dimensional of X_{ij}, returning $\bar{X}_{ik}$ with $k \ll j$ where each column k is a principal component. Each component $k = \{1, \dots K\}$ has no linear correlation (= orthogonal) with each other while preserving as much variance as possible:

- PC1 maps X_{ij} onto an axis that contains the highest variance
- PC2 maps X_{ij} onto an axis orthogonal to PC1 and that contains the highest amount of leftover variance.

$\rightsquigarrow$ We maximize variance

Interpretation:

- Sometimes straightforward, sometimes difficult
- Intuition: PC1 maps the data into the latent dimension along which observations vary the most $\rightsquigarrow$ Data specific!

Discuss an application for which PCA could be used to identify latent dimensions.

What Makes Party Systems Different? (Magyar 2021

PA

What Makes Party Systems Different? A Principal Component Analysis of 17 Advanced Democracies 1970–2013

Zsuzsanna B. Magyar

University of Lucerne, Lucerne, Switzerland. Email: zmagyar@ucla.edu

Abstract

Party systems, that is, the number and the size of all the parties within a country, can vary greatly across countries. I conduct a principal component analysis on a party seat share dataset of 17 advanced democracies from 1970 to 2013 to reduce the dimensionality of the data. I find that the most important dimensions that differentiate party systems are: "the size of the biggest two parties" and the level of "competition between the two biggest parties." I use the results to compare the changes in electoral and legislative party systems. I also juxtapose the results to previous party system typologies and party system size measures. I find that typologies sort countries into categories based on variation along both dimensions. On the other hand, most of the current political science literature use measures (e.g., the effective number of parties) that are correlated with the first dimension. I suggest that instead of these, indices that measure the opposition structure and competition could be used to explore problems pertaining to the competitiveness of the party systems.

Keywords: principal component analysis, party system, typology, effective number of parties

Results (Magyar 2021)

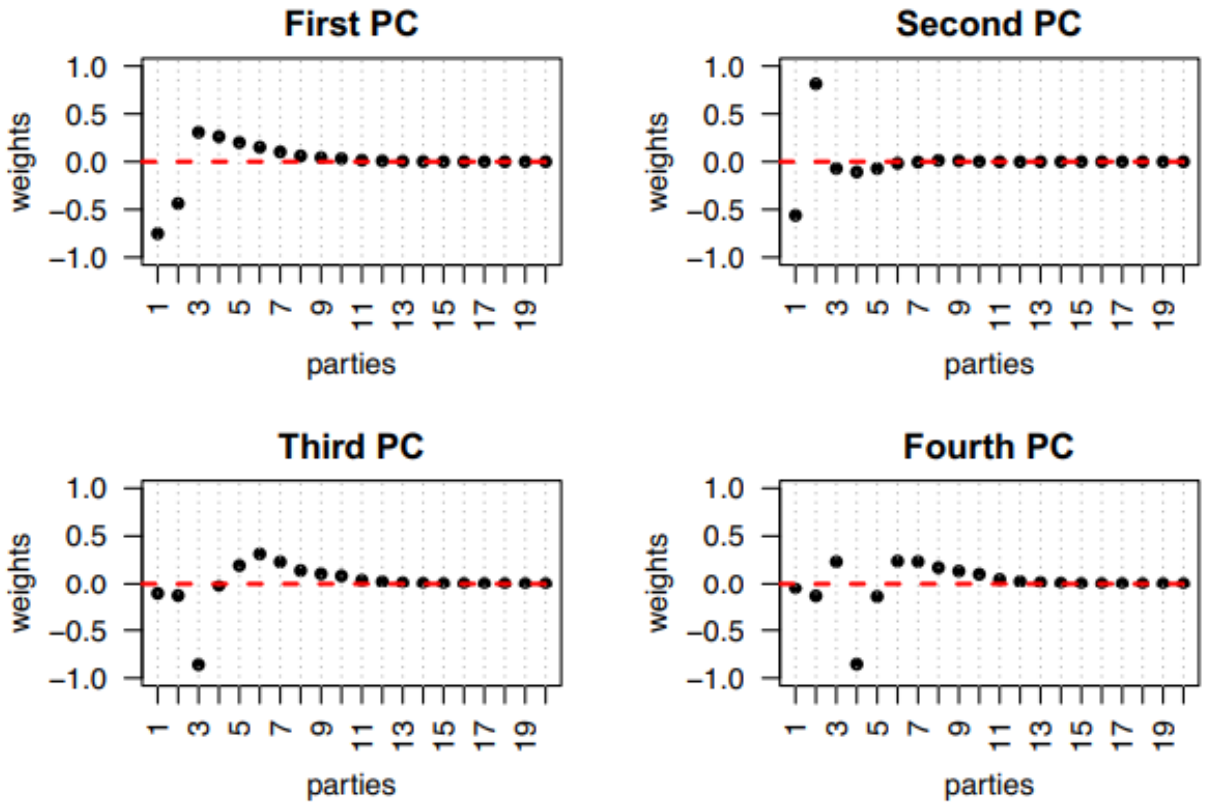


Figure 1. Loadings, PCA.*Notes:* The plot shows the loadings, the weight of parties in determining the principal components in the PCA.

Compound Measures (Abramson et al. 2022

Historical Border Changes, State Building, and Contemporary Trust in Europe

SCOTT F ABRAMSON *University of Rochester, United States*
DAVID B. CARTER *Washington University in St. Louis, United States*
LUWEI YING *University of California, Los Angeles, United States*

Political borders profoundly influence outcomes central to international politics. Accordingly, a growing literature shows that historical boundaries affect important macro-outcomes such as patterns of interstate disputes and trade. To explain these findings, existing theories posit that borders have persistent effects on individual-level behavior, but the literature lacks empirical evidence of such effects. Combining spatial data on centuries of border changes in Europe with a wide range of contemporary survey evidence, we show that historical border changes have persistent effects on two of the most politically significant aspects of behavior: individuals' political and social trust. We demonstrate that in areas where borders frequently changed, individuals are, on average, less trusting of others as well as their governments. We argue that this occurs because border changes disrupt historical state-building processes and limit the formation of interpersonal social networks, which leads to lower levels of trust.

Table of Contents

From ML to DL

From ML to DL

ML

- Feature extraction
- Input processed data
- “Small data”

From ML to DL

ML

- Feature extraction
- Input processed data
- “Small data”

DL

- Feature learning
- Input raw data
- “Big data”

Regression as a Neural Network I

Regression as a Neural Network I

Neural networks are complex when they get *deep*...

Regression as a Neural Network I

Neural networks are complex when they get *deep*... But in reality, they are just multiple regressions combined together!

Regression as a Neural Network I

Neural networks are complex when they get *deep*... But in reality, they are just multiple regressions combined together! Let us revisit the classical logistic regression:

Regression as a Neural Network I

Neural networks are complex when they get *deep*... But in reality, they are just multiple regressions combined together! Let us revisit the classical logistic regression:

$$\hat{y} = \underbrace{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_k x_k}_{z}$$

$$\hat{y} = \sigma(z)$$

where $\sigma(\cdot)$ is the sigmoid function:

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

When we input x, what do we get out? A (smooth) probability bounded between 0 and 1

Regression as a Neural Network I

Neural networks are complex when they get *deep*... But in reality, they are just multiple regressions combined together! Let us revisit the classical logistic regression:

$$\hat{y} = \underbrace{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_k x_k}_{z}$$

where $\sigma(\cdot)$ is the sigmoid function:

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

When we input x, what do we get out? A (smooth) probability bounded between 0 and 1

Regression as a Neural Network I

Neural networks are complex when they get *deep*... But in reality, they are just multiple regressions combined together! Let us revisit the classical logistic regression:

$$\hat{y} = \underbrace{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_k x_k}_{z}$$

$$\hat{y} = \sigma(z)$$

where $\sigma(\cdot)$ is the sigmoid function:

Regression as a Neural Network I

Neural networks are complex when they get *deep*... But in reality, they are just multiple regressions combined together! Let us revisit the classical logistic regression:

$$\hat{y} = \underbrace{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_k x_k}_{z}$$

$$\hat{y} = \sigma(z)$$

where $\sigma(\cdot)$ is the sigmoid function:

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

When we input x, what do we get out? A (smooth) probability bounded between 0 and 1

Regression as a Neural Network I

Neural networks are complex when they get *deep*... But in reality, they are just multiple regressions combined together! Let us revisit the classical logistic regression:

$$\hat{y} = \underbrace{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_k x_k}_{z}$$

$$\hat{y} = \sigma(z)$$

where $\sigma(\cdot)$ is the sigmoid function:

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

When we input x, what do we get out?

Regression as a Neural Network I

Neural networks are complex when they get *deep*... But in reality, they are just multiple regressions combined together! Let us revisit the classical logistic regression:

$$\hat{y} = \underbrace{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_k x_k}_{z}$$

$$\hat{y} = \sigma(z)$$

where $\sigma(\cdot)$ is the sigmoid function:

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

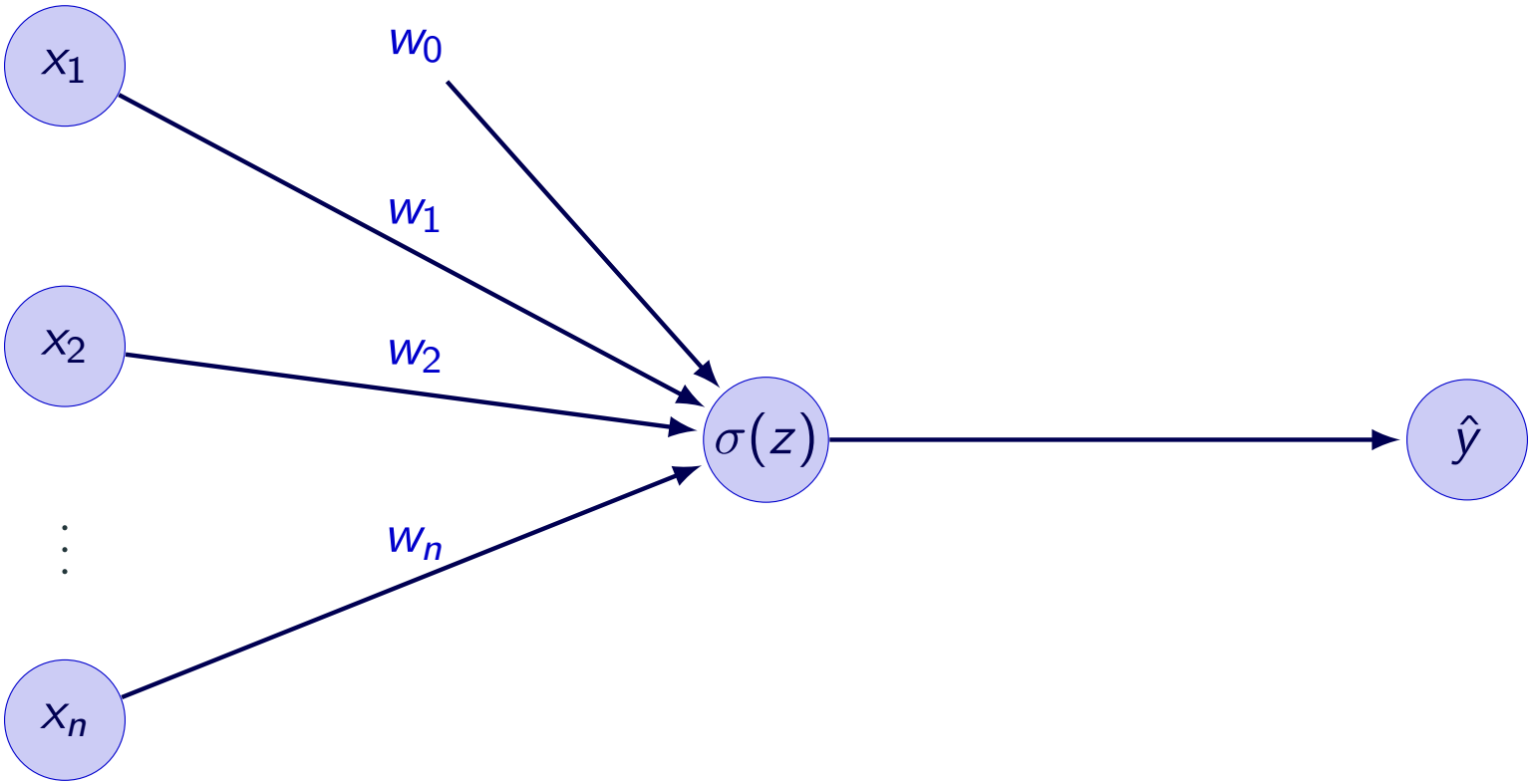
When we input x, what do we get out? A (smooth) probability bounded between 0 and 1

Logistic as a Net

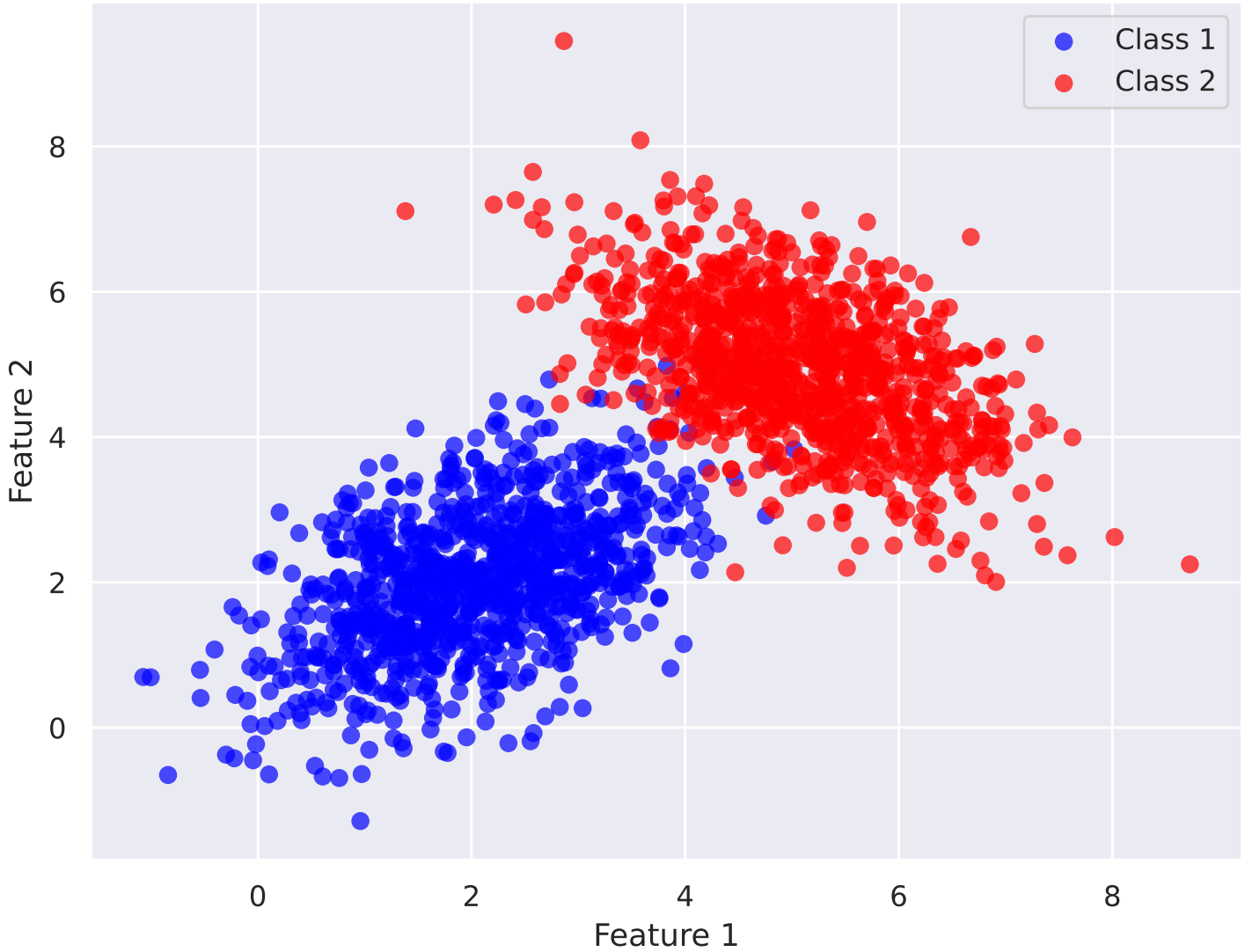
We can cast the logistic regression as a network:

Logistic as a Net

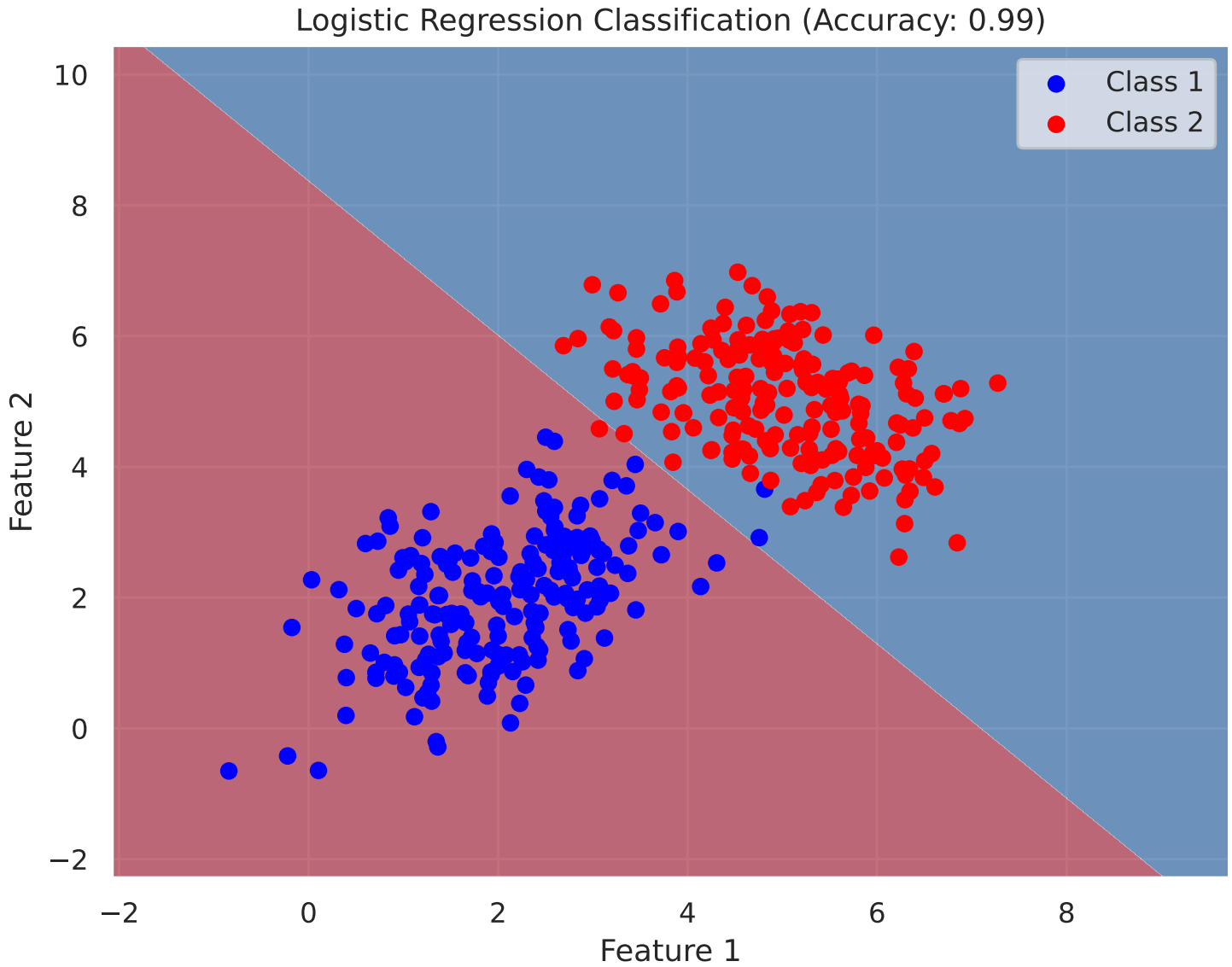
We can cast the logistic regression as a network:



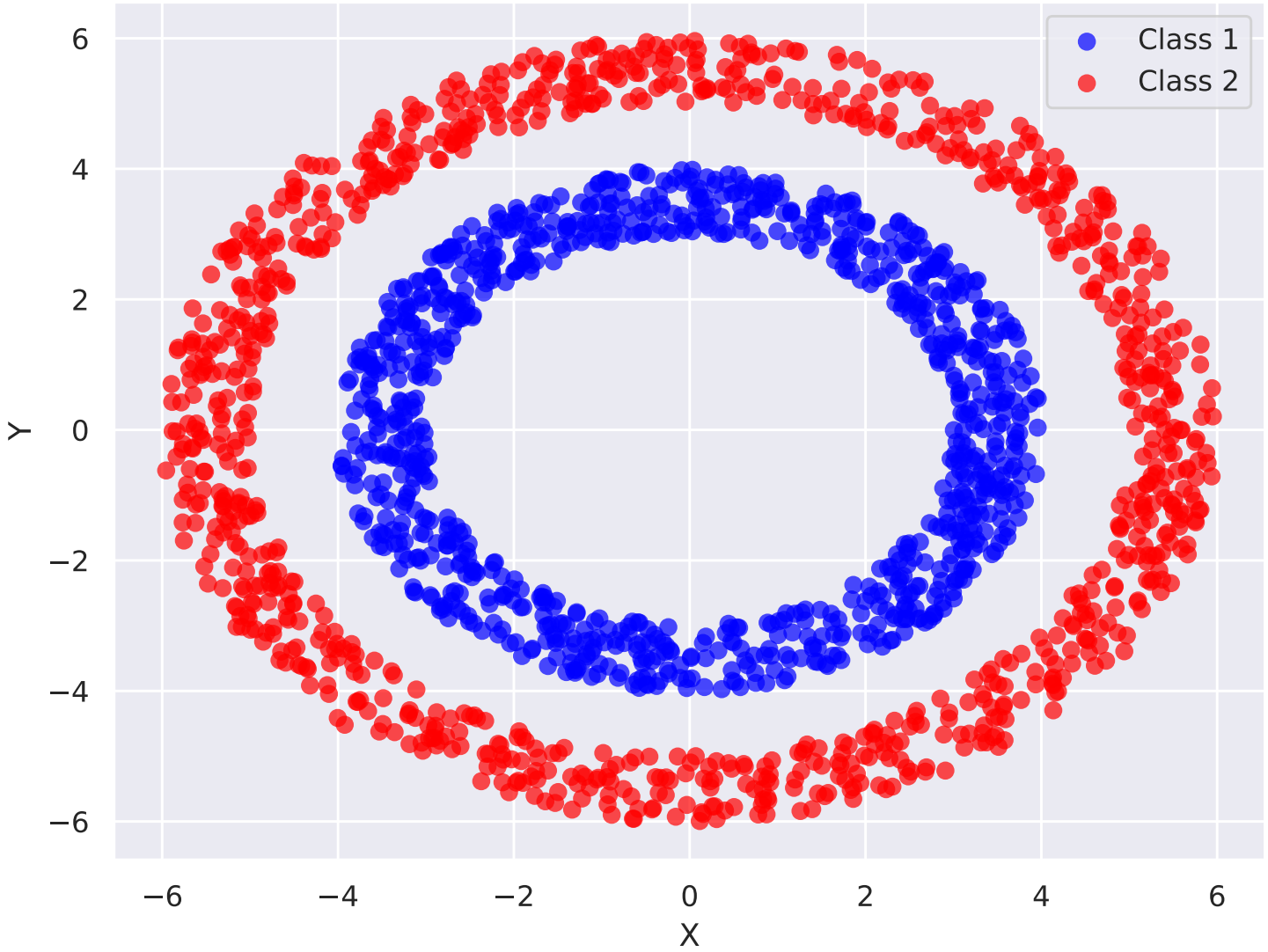
Linear Separable Data



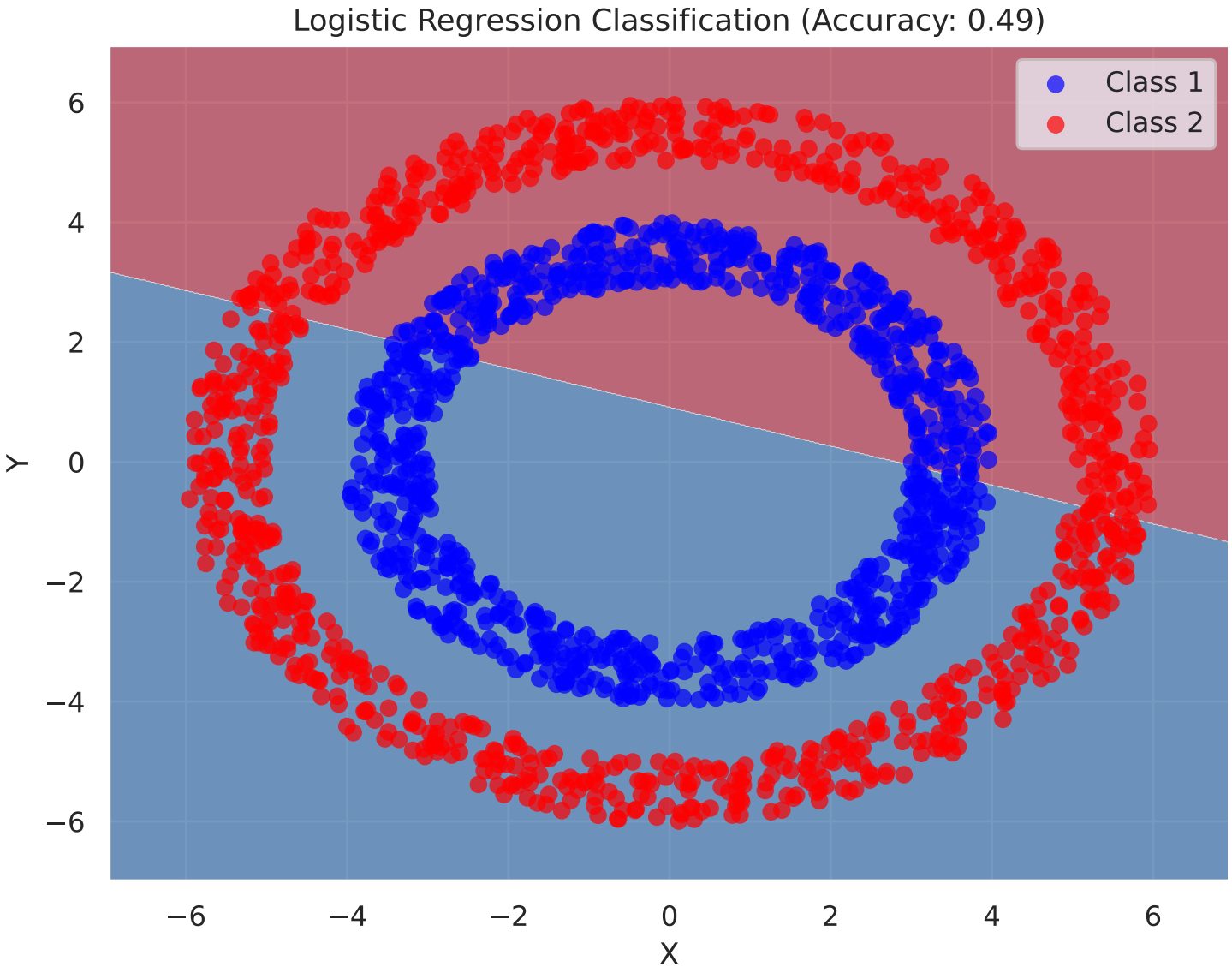
Decision Boundary



Non-linear Separable Data



Decision Boundary



Solving Nonlinear Problems

Solving Nonlinear Problems

Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

Solving Nonlinear Problems

Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

⇝ Neural networks allow us to handle that flexibly.

Solving Nonlinear Problems

Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

⇝ Neural networks allow us to handle that flexibly.

- Forward-nested regressions

Solving Nonlinear Problems

Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

⇝ Neural networks allow us to handle that flexibly.

- Forward-nested regressions
- Activation functions required to solve non-linear tasks

Solving Nonlinear Problems

Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

⇝ Neural networks allow us to handle that flexibly.

- Forward-nested regressions
- Activation functions required to solve non-linear tasks
- Number of layers: Layers with weights

Solving Nonlinear Problems

Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

⇝ Neural networks allow us to handle that flexibly.

- Forward-nested regressions
- Activation functions required to solve non-linear tasks
- Number of layers: Layers with weights
- Can be deep, very deep

Solving Nonlinear Problems

Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

⇝ Neural networks allow us to handle that flexibly.

- Forward-nested regressions
- Activation functions required to solve non-linear tasks
- Number of layers: Layers with weights
- Can be deep, very deep ⇝ transfer learning!

Solving Nonlinear Problems

Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

⇝ Neural networks allow us to handle that flexibly.

- Forward-nested regressions
- Activation functions required to solve non-linear tasks
- Number of layers: Layers with weights
- Can be deep, very deep ⇝ transfer learning!
  - Reduces training time
  - Reduces the need for GPU
  - Reduces CO_2 footprint
- Can have multiple output neurons

Solving Nonlinear Problems

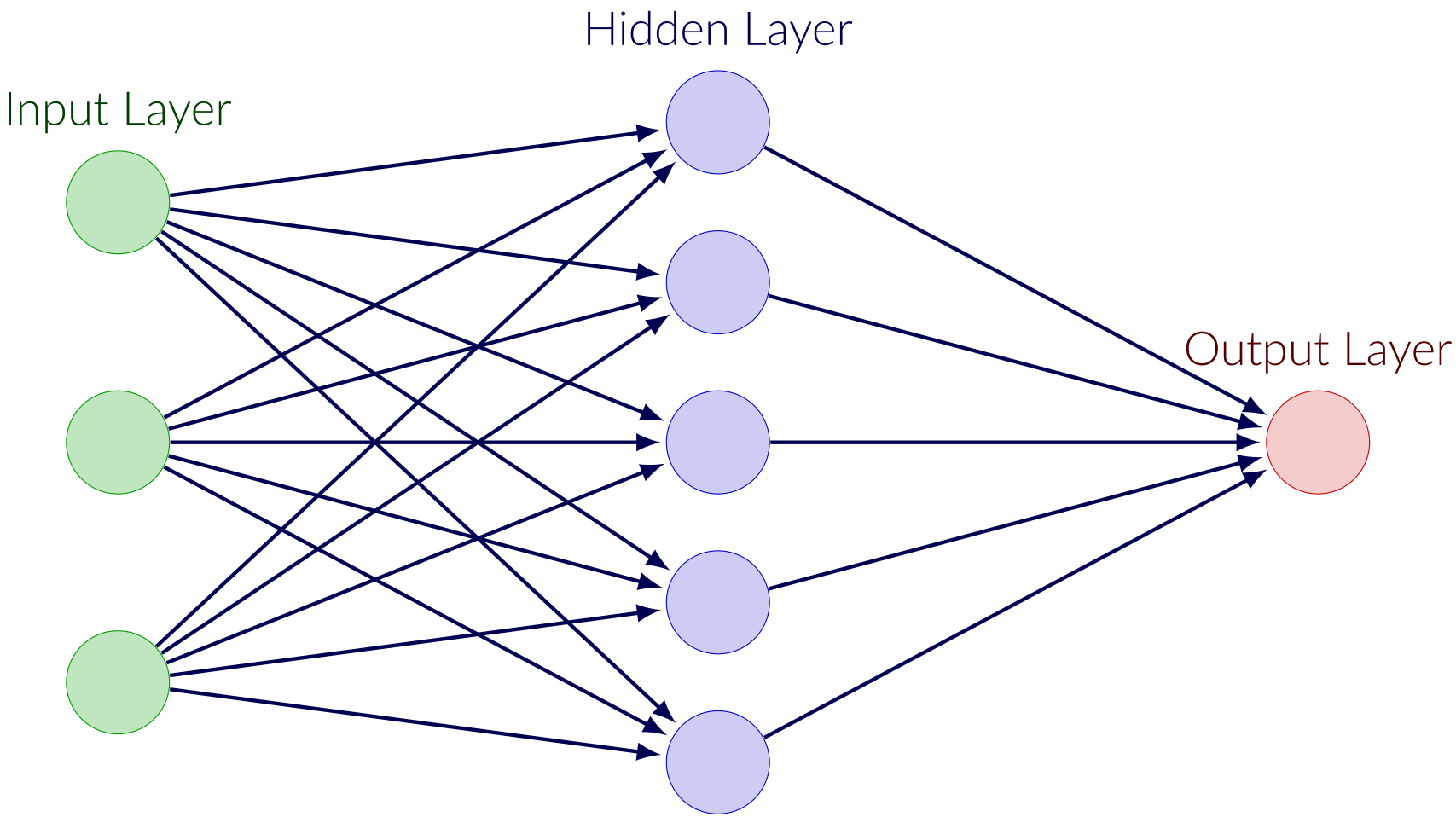
Unstructured and complex data like text, audio, and images are highly nonlinear and operate in high dimensions.

⇝ Neural networks allow us to handle that flexibly.

- Forward-nested regressions
- Activation functions required to solve non-linear tasks
- Number of layers: Layers with weights
- Can be deep, very deep ⇝ transfer learning!
  - Reduces training time
  - Reduces the need for GPU
  - Reduces CO_2 footprint
- Can have multiple output neurons
- Find weights using backpropagation and gradient descent (not important for us)
- BUT: Prone to overfitting!

Neural Network Diagram

Neural Network Diagram



How Many Weights?

How Many Weights?

Let's use our example from before with 3 input neurons, one hidden layer, five hidden neurons, and a single output neuron.

How Many Weights?

Let's use our example from before with 3 input neurons, one hidden layer, five hidden neurons, and a single output neuron.

How many weights are there? Remember to include the biases!

How Many Weights?

Let's use our example from before with 3 input neurons, one hidden layer, five hidden neurons, and a single output neuron.

How many weights are there? Remember to include the biases!

Result: $3 \times 5 + 5 \times 1 = 15 + 5 = 20$

How Many Weights?

Let's use our example from before with 3 input neurons, one hidden layer, five hidden neurons, and a single output neuron.

How many weights are there? Remember to include the biases!

Result: $3 \times 5 + 5 \times 1 = 15 + 5 = 20$

Including biases: $20 + 5 + 1 = 26$ weights

How Many Weights?

Let's use our example from before with 3 input neurons, one hidden layer, five hidden neurons, and a single output neuron.

How many weights are there? Remember to include the biases!

Result: $3 \times 5 + 5 \times 1 = 15 + 5 = 20$

Including biases: $20 + 5 + 1 = 26$ weights

If we write this up mathematically, we can see that it is just nested logistic regression:

How Many Weights?

Let's use our example from before with 3 input neurons, one hidden layer, five hidden neurons, and a single output neuron.

How many weights are there? Remember to include the biases!

Result: $3 \times 5 + 5 \times 1 = 15 + 5 = 20$

Including biases: $20 + 5 + 1 = 26$ weights

If we write this up mathematically, we can see that it is just nested logistic regression:

$$
\begin{aligned}
h_1 &= \sigma(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + b_1) \\
h_2 &= \sigma(w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + b_2) \\
h_3 &= \sigma(w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + b_3) \\
h_4 &= \sigma(w_{14}x_1 + w_{24}x_2 + w_{34}x_3 + b_4) \\
h_5 &= \sigma(w_{15}x_1 + w_{25}x_2 + w_{35}x_3 + b_5) \\
\hat{y} &= \sigma(w_{y1}h_1 + w_{y2}h_2 + w_{y3}h_3 + w_{y4}h_4 + w_{y5}h_5 + b_y)
\end{aligned}
$$

Architectures and Layers

Architectures and Layers

Neural networks come in many different varieties:

Architectures and Layers

Neural networks come in many different varieties:

- Fully-connected (the default)

Architectures and Layers

Neural networks come in many different varieties:

- Fully-connected (the default)
- Convolutional: Images (and audio)

Architectures and Layers

Neural networks come in many different varieties:

- Fully-connected (the default)
- Convolutional: Images (and audio)
- Recurrent: Text and audio

Architectures and Layers

Neural networks come in many different varieties:

- Fully-connected (the default)
- Convolutional: Images (and audio)
- Recurrent: Text and audio
- Transformers: Text, audio, and images

Architectures and Layers

Neural networks come in many different varieties:

- Fully-connected (the default)
- Convolutional: Images (and audio)
- Recurrent: Text and audio
- Transformers: Text, audio, and images
- Generative models: Text, audio, and images)

Architectures and Layers

Neural networks come in many different varieties:

- Fully-connected (the default)
- Convolutional: Images (and audio)
- Recurrent: Text and audio
- Transformers: Text, audio, and images
- Generative models: Text, audio, and images)

We will primarily use the fully-connected layers and convolution nets.

See you next week!

Topic 1: ML Basics

Computational Analysis of Text, Audio, and Images, Fall 2023

Aarhus University