CS 474/574 Machine Learning
1. Introduction

Prof. Dr. Forrest Sheng Bao
Dept. of Computer Science
Iowa State University
Ames, IA, USA

January 27, 2021

Why Machine Learning (ML)

- How computers know how to do things?

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer
  2. learning: without being specifically told

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer
  2. learning: without being specifically told
- Example 1: machine translation

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer
  2. learning: without being specifically told
- Example 1: machine translation
  1. programming: writing many rules to replace and reposition words, e.g., *Do you speak Julia? Sprechen Sie Julia?*

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer
  2. learning: without being specifically told
- Example 1: machine translation
  1. programming: writing many rules to replace and reposition words, e.g., *Do you speak Julia? Sprechen Sie Julia?*
  2. learning: feeding the computer many bilingual documents

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer
  2. learning: without being specifically told
- Example 1: machine translation
  1. programming: writing many rules to replace and reposition words, e.g., *Do you speak Julia? Sprechen Sie Julia?*
  2. learning: feeding the computer many bilingual documents
- Example 2: sorting

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer
  2. learning: without being specifically told
- Example 1: machine translation
  1. programming: writing many rules to replace and reposition words, e.g., *Do you speak Julia? Sprechen Sie Julia?*
  2. learning: feeding the computer many bilingual documents
- Example 2: sorting
  1. programming: Quicksort, etc.

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer
  2. learning: without being specifically told
- Example 1: machine translation
  1. programming: writing many rules to replace and reposition words, e.g., *Do you speak Julia? Sprechen Sie Julia?*
  2. learning: feeding the computer many bilingual documents
- Example 2: sorting
  1. programming: Quicksort, etc.
  2. learning: feeding the computer many pairs of unsorted and sorted list of numbers.

Why Machine Learning (ML)

- How computers know how to do things?
- Two ways:
  1. programming: steps detailed by human programmer
  2. learning: without being specifically told
- Example 1: machine translation
  1. programming: writing many rules to replace and reposition words, e.g., *Do you speak Julia? Sprechen Sie Julia?*
  2. learning: feeding the computer many bilingual documents
- Example 2: sorting
  1. programming: Quicksort, etc.
  2. learning: feeding the computer many pairs of unsorted and sorted list of numbers.
- The first approach in the context of AI is also called rule-based system or expert system, e.g. MyCin, Grammarly.

Why ML is attractive

- We are lazy. We want to shift the heavy lifting to the computers.

Why ML is attractive

- We are lazy. We want to shift the heavy lifting to the computers.
- We are incompetent. No kidding! Sometimes it is very difficult to come up with step-by-step instructions.

Why ML is attractive

- We are lazy. We want to shift the heavy lifting to the computers.
- We are incompetent. No kidding! Sometimes it is very difficult to come up with step-by-step instructions.
- Examples: Self-driving, AlphaGo, Automated circuit routing, Machine translation, Commonsense reasoning, text entailment, Document generation, auto-reply of messages/emails, fly a helicoper inversely, van-Gogh-lize paints.

Why ML is attractive

- We are lazy. We want to shift the heavy lifting to the computers.
- We are incompetent. No kidding! Sometimes it is very difficult to come up with step-by-step instructions.
- Examples: Self-driving, AlphaGo, Automated circuit routing, Machine translation, Commonsense reasoning, text entailment, Document generation, auto-reply of messages/emails, fly a helicoper inversely, van-Gogh-lize paints.
- It is a dream. "Creating an artificial being has been the dream since the beginning of science." – Movie A.I., Spielberg et al., 2001

Functions

- A function is a many(incl. one)-to-one mapping between data!

Functions

- A function is a many(incl. one)-to-one mapping between data!
- It does not need to have an analytic form and the domain/range does not have to be of real numbers.

Functions

- A function is a many(incl. one)-to-one mapping between data!
- It does not need to have an analytic form and the domain/range does not have to be of real numbers.
- An example of edge detection in Wolfram Mathematica:

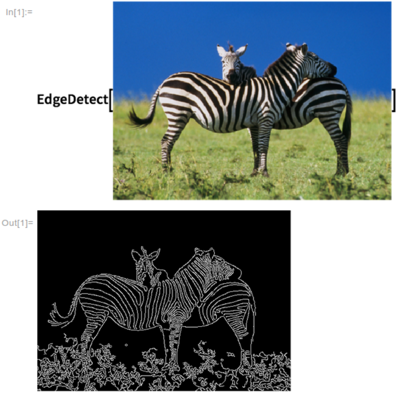
In[1]:= EdgeDetect[

]

Out[1]=

Functions

- A function is a many(incl. one)-to-one mapping between data!
- It does not need to have an analytic form and the domain/range does not have to be of real numbers.
- An example of edge detection in Wolfram Mathematica:



- Another example in machine translation

Supervised ML as function approximation

Three types of MLs

ML (in current approaches) is about finding/approximating functions.

- Supervised, finding $\hat{f}(x) \approx f(x)$ with ground truth provided by human.

Three types of MLs

ML (in current approaches) is about finding/approximating functions.

- Supervised, finding $\hat{f}(x) \approx f(x)$ with ground truth provided by human.
    - Let x and y be two (vectors of) variables, and a function connecting them $y = f(x)$ But only god knows f.

Three types of MLs

ML (in current approaches) is about finding/approximating functions.

- Supervised, finding $\hat{f}(x) \approx f(x)$ with ground truth provided by human.
  - Let x and y be two (vectors of) variables, and a function connecting them $y = f(x)$ But only god knows f.
  - We construct another function $\hat{f}$ to approximate f such that $\hat{y} = \hat{f}(x) \approx y = f(x)$ for a(ny) given x.

Three types of MLs

ML (in current approaches) is about finding/approximating functions.

- Supervised, finding $\hat{f}(x) \approx f(x)$ with ground truth provided by human.
  - Let x and y be two (vectors of) variables, and a function connecting them $y = f(x)$ But only god knows f.
  - We construct another function $\hat{f}$ to approximate f such that $\hat{y} = \hat{f}(x) \approx y = f(x)$ for a(ny) given x.
  - **Supervised** because we provide many pairs of x's and y's for the computer to know the difference between $\hat{y}$ and y on a large pool of samples.

Three types of MLs

ML (in current approaches) is about finding/approximating functions.

- Supervised, finding $\hat{f}(x) \approx f(x)$ with ground truth provided by human.
  - Let x and y be two (vectors of) variables, and a function connecting them $y = f(x)$ But only god knows f.
  - We construct another function $\hat{f}$ to approximate f such that $\hat{y} = \hat{f}(x) \approx y = f(x)$ for a(ny) given x.
  - **Supervised** because we provide many pairs of x's and y's for the computer to know the difference between $\hat{y}$ and y on a large pool of samples.
  - Examples: object detection from images, Flavia, CPU branch prediction, COVID-19 diagnosis from blood profile, Epileptic EEG recognition, depression treatment from brain shapes.

Three types of MLs

ML (in current approaches) is about finding/approximating functions.

- Supervised, finding $\hat{f}(x) \approx f(x)$ with ground truth provided by human.
  - Let x and y be two (vectors of) variables, and a function connecting them $y = f(x)$ But only god knows f.
  - We construct another function $\hat{f}$ to approximate f such that $\hat{y} = \hat{f}(x) \approx y = f(x)$ for a(ny) given x.
  - **Supervised** because we provide many pairs of x's and y's for the computer to know the difference between $\hat{y}$ and y on a large pool of samples.
  - Examples: object detection from images, Flavia, CPU branch prediction, COVID-19 diagnosis from blood profile, Epileptic EEG recognition, depression treatment from brain shapes.
  - Beyond categorization/classification: Mflux, Review helpfulness prediction, Document summarization, predict house prices

Three types of MLs

ML (in current approaches) is about finding/approximating functions.

- Supervised, finding $\hat{f}(x) \approx f(x)$ with ground truth provided by human.
  - Let x and y be two (vectors of) variables, and a function connecting them $y = f(x)$ But only god knows f.
  - We construct another function $\hat{f}$ to approximate f such that $\hat{y} = \hat{f}(x) \approx y = f(x)$ for a(ny) given x.
  - **Supervised** because we provide many pairs of x's and y's for the computer to know the difference between $\hat{y}$ and y on a large pool of samples.
  - Examples: object detection from images, Flavia, CPU branch prediction, COVID-19 diagnosis from blood profile, Epileptic EEG recognition, depression treatment from brain shapes.
  - Beyond categorization/classification: Mflux, Review helpfulness prediction, Document summarization, predict house prices
- Unsupervised, finding $\hat{f}(x)$ without ground truth

Three types of MLs

ML (in current approaches) is about finding/approximating functions.

- Supervised, finding $\hat{f}(x) \approx f(x)$ with ground truth provided by human.
  - Let x and y be two (vectors of) variables, and a function connecting them $y = f(x)$ But only god knows f.
  - We construct another function $\hat{f}$ to approximate f such that $\hat{y} = \hat{f}(x) \approx y = f(x)$ for a(ny) given x.
  - **Supervised** because we provide many pairs of x's and y's for the computer to know the difference between $\hat{y}$ and y on a large pool of samples.
  - Examples: object detection from images, Flavia, CPU branch prediction, COVID-19 diagnosis from blood profile, Epileptic EEG recognition, depression treatment from brain shapes.
  - Beyond categorization/classification: Mflux, Review helpfulness prediction, Document summarization, predict house prices
- Unsupervised, finding $\hat{f}(x)$ without ground truth
- Reinforcement, let the machine find ground truth itself

Representation of x

- x is usually not a simple (vector of) number(s). How to tell it to a computer?

Representation of x

- x is usually not a simple (vector of) number(s). How to tell it to a computer?
- Example: bananas vs. apples

Representation of x

- x is usually not a simple (vector of) number(s). How to tell it to a computer?
- Example: bananas vs. apples
- **Feature engineering**: manually craft functions to **extract** features from raw data, e.g,. SIFT, bag-of-words.

Representation of x

- x is usually not a simple (vector of) number(s). How to tell it to a computer?
- Example: bananas vs. apples
- **Feature engineering**: manually craft functions to **extract** features from raw data, e.g,. SIFT, bag-of-words.
- Automated feature extraction in deep learing: E.g., filters in CNNs.

Representation of x

- x is usually not a simple (vector of) number(s). How to tell it to a computer?
- Example: bananas vs. apples
- **Feature engineering**: manually craft functions to **extract** features from raw data, e.g,. SIFT, bag-of-words.
- Automated feature extraction in deep learing: E.g., filters in CNNs.
- If x involves categorical values (e.g., gender), there are usually two approaches: **One-hot encoding** and **embedding** (in DL context, to be discussed later).

Supervised ML

- Given many pairs of inputs and outputs:
  $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \ldots, (\mathbf{X_N}, \mathbf{y_N})\}$,

Supervised ML

- Given many pairs of inputs and outputs: $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \ldots, (\mathbf{X_N}, \mathbf{y_N})\}$,
- that underline a "black-box" function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that $\forall i \in [1..n], f(\mathbf{X}_i) = \mathbf{y}_i$,

Supervised ML

- Given many pairs of inputs and outputs: $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \ldots, (\mathbf{X_N}, \mathbf{y_N})\}$,
- that underline a "black-box" function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that $\forall i \in [1..n], f(\mathbf{X}_i) = \mathbf{y}_i$,
- construct a function $\hat{f}$ that approximates the function f.

Supervised ML

- Given many pairs of inputs and outputs: $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \dots, (\mathbf{X_N}, \mathbf{y_N})\}$,
- that underline a "black-box" function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that $\forall i \in [1..n], f(\mathbf{X}_i) = \mathbf{y}_i$,
- construct a function $\hat{f}$ that approximates the function f.
- "approximate": usually $\min ||\hat{f}(x) - f(x)||^p$ where p is usually 1 or 2. See ℓ_p-norm .

Supervised ML

- Given many pairs of inputs and outputs: $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \ldots, (\mathbf{X_N}, \mathbf{y_N})\}$,
- that underline a "black-box" function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that $\forall i \in [1..n], f(\mathbf{X}_i) = \mathbf{y}_i$,
- construct a function $\hat{f}$ that approximates the function f.
- "approximate": usually $\min ||\hat{f}(x) - f(x)||^p$ where p is usually 1 or 2. See ℓ_p-norm .
- The process of finding the approximation function $\hat{f}$ is called **training** or **learning**.

Supervised ML

- Given many pairs of inputs and outputs: $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \ldots, (\mathbf{X_N}, \mathbf{y_N})\}$,
- that underline a "black-box" function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that $\forall i \in [1..n], f(\mathbf{X}_i) = \mathbf{y}_i$,
- construct a function $\hat{f}$ that approximates the function f.
- "approximate": usually $\min ||\hat{f}(x) - f(x)||^p$ where p is usually 1 or 2. See ℓ_p-norm .
- The process of finding the approximation function $\hat{f}$ is called **training** or **learning**.
- $\hat{f}$ is called a **model** or an **estimator**.

Supervised ML

- Given many pairs of inputs and outputs: $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \ldots, (\mathbf{X_N}, \mathbf{y_N})\}$,
- that underline a "black-box" function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that $\forall i \in [1..n], f(\mathbf{X}_i) = \mathbf{y}_i$,
- construct a function $\hat{f}$ that approximates the function f.
- "approximate": usually $\min ||\hat{f}(x) - f(x)||^p$ where p is usually 1 or 2. See ℓ_p-norm .
- The process of finding the approximation function $\hat{f}$ is called **training** or **learning**.
- $\hat{f}$ is called a **model** or an **estimator**.
- $\mathbf{X_i}$: an **input** (especially when raw data is used as the input) or **feature vector** (if using feature engineering).

Supervised ML

- Given many pairs of inputs and outputs: $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \ldots, (\mathbf{X_N}, \mathbf{y_N})\}$,
- that underline a "black-box" function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that $\forall i \in [1..n], f(\mathbf{X}_i) = \mathbf{y}_i$,
- construct a function $\hat{f}$ that approximates the function f.
- "approximate": usually $\min ||\hat{f}(x) - f(x)||^p$ where p is usually 1 or 2. See ℓ_p-norm .
- The process of finding the approximation function $\hat{f}$ is called **training** or **learning**.
- $\hat{f}$ is called a **model** or an **estimator**.
- $\mathbf{X_i}$: an **input** (especially when raw data is used as the input) or **feature vector** (if using feature engineering).
- $\mathbf{y_i}$, often $\in \mathbb{R}^1$ a **label** (in classification) or **target** (used more generally and lately).

Supervised ML

- Given many pairs of inputs and outputs: $\{(\mathbf{X_1}, \mathbf{y_1}), (\mathbf{X_2}, \mathbf{y_2}), \ldots, (\mathbf{X_N}, \mathbf{y_N})\}$,
- that underline a "black-box" function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that $\forall i \in [1..n], f(\mathbf{X}_i) = \mathbf{y}_i$,
- construct a function $\hat{f}$ that approximates the function f.
- "approximate": usually $\min ||\hat{f}(x) - f(x)||^p$ where p is usually 1 or 2. See ℓ_p-norm .
- The process of finding the approximation function $\hat{f}$ is called **training** or **learning**.
- $\hat{f}$ is called a **model** or an **estimator**.
- $\mathbf{X_i}$: an **input** (especially when raw data is used as the input) or **feature vector** (if using feature engineering).
- $\mathbf{y_i}$, often $\in \mathbb{R}^1$ a **label** (in classification) or **target** (used more generally and lately).
- Classification vs. Regression: When y is continuous or discrete. In modern DL context, such division is usually no mentioned, expecially in generative tasks.