Supervised Machine Learning and Learning Theory

Lecture 1: Introduction, course information, syllabus, and examples

September 6, 2024

Lecture plan

- Course logistics
- Course information
- Coursework and grading policy
- Several examples

Motivation

- **This semester we'll go through an introductory class on machine learning**

- **Machine learning** is a subarea of artificial intelligence and computer science, traditionally speaking

- Increasingly, machine learning is being used outside computer science, including transportation, healthcare, biology, to name a few

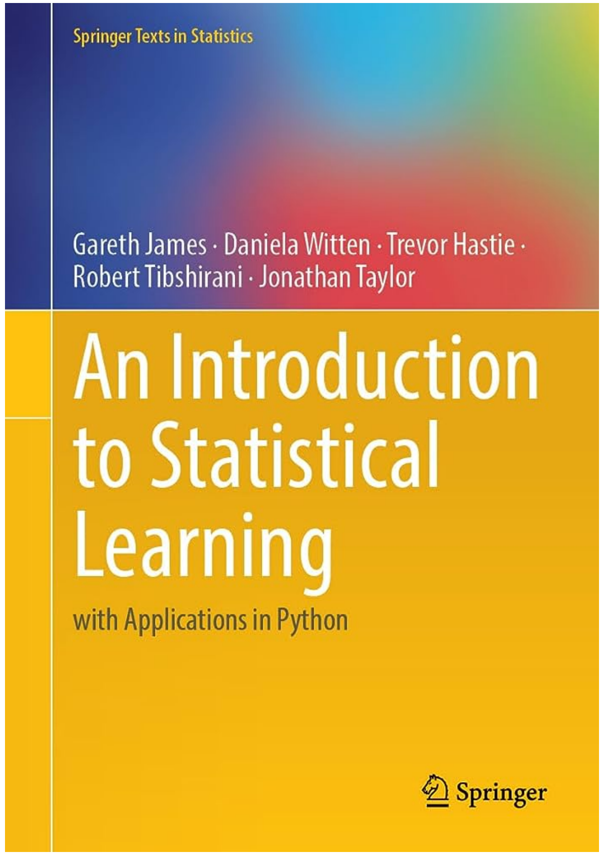
- **The rise of large language models** has spurred new interests in the subject of machine learning

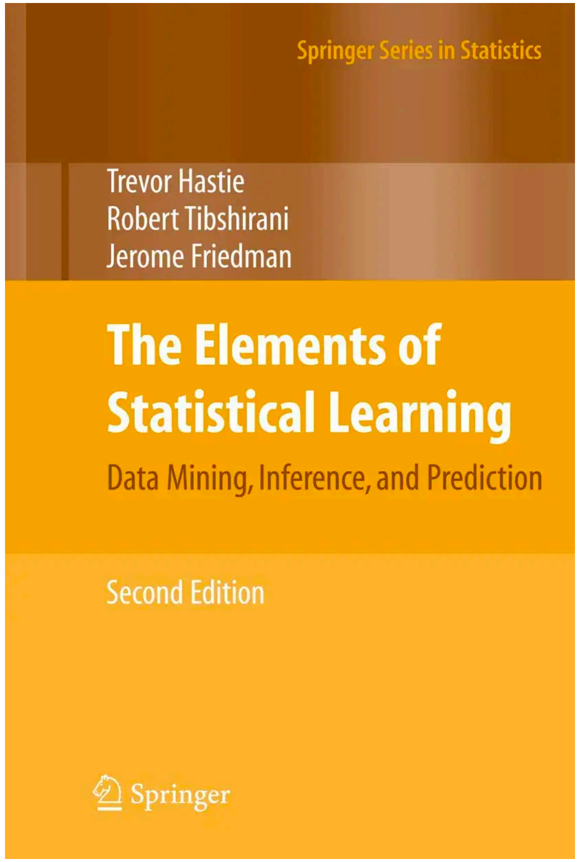
Syllabus

- **Part I:** Regression and classification
- **Part II:** Neural networks and deep learning
  - Convolutional neural networks
  - Transformer neural networks
  - Language modeling
- **Part III**: From prediction to inference, and beyond
  - Causality
  - Unsupervised learning
- **Prerequisites**: Linear Algebra, Applied Probability, Python

A complete syllabus can be found on canvas for your reference
https://northeastern.instructure.com/courses/193108/files/29529231?module_item_id=11012198

Recommended textbooks





I will try to write down some supporting notes and send to the class when possible

Course information

- **Note: This class is for students taking machine learning for the first time**

- **If you've taken an introductory machine learning class, I recommend wait until the spring semester. I will be teaching CS 7140 Advanced Machine Learning ☺**

- **Instructor**: Ryan Zhang
- **Location**: Shillman Hall 305
- **Time**: Tuesdays and Fridays from 9:50 AM to 11:30 AM

Coursework and grading policy

- Homework 40%
  - 5 problem sets with a mix of theoretical and empirical questions
- Midterm exam 30%
  - Nov 6 – Nov 9, take home, pick 24 hours from Wednesday until Saturday
- Course project proposal presentation + final project presentation 15%
  - In class, Oct 29, Nov 1; then Dec 6, Dec 9
- Final project report 15%
  - We will provide a template project around developing and applying language models. Another option is to pick a problem you are interested and develop a machine-learning solution to solve it

Lecture plan

- **Examples**

Predicting apartment/housing values

- **Example**: suppose you want to predict the apartment/housing values/rental cost in one of suburb towns of Boston
- **Dataset**: we have a dataset that contains n samples
- **Problem setup:** $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), \ldots, (\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})$
  - $\boldsymbol{x}^{(i)}$ is a feature vector: number of bedrooms, number of bathrooms, sqft, zip code, distance to nearest train station, number of restaurants nearby…
  - $\boldsymbol{y}^{(i)}$ is the valued/selling/rental price of the apartment/house
- **Question**: For an unseen house/apartment, predict the market price of this property?

Predicting apartment/housing values

- Recall our problem setup: Given a dataset that contains n samples $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), \ldots (\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})$, we would like to develop a predictive model that can map the $x's$ to the $y's$
  - **Each $\boldsymbol{x}^{(i)}$** includes a list of relevant features
  - **Each $\boldsymbol{y}^{(i)}$** is a label/response/target we would like to predict
- **Note**
  - If the labels y take values in a continuous range, the problem is called a **regression problem**
  - If the labels y take values in a discrete range, the problem is called a **classification problem**

Linear regression

- **Let us look at the simplest, and perhaps most widely used machine learning model**
- **Step 1: divide the dataset into train/validation/test splits (rule of thumb 80/10/10)**
- **Step 2: take the training split**, and derive the so called **mean squared error metric**:

$$\hat{\boldsymbol{L}}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta^\top x^{(i)} - y^{(i)}\right)^2$$

  - Above, let us use θ to denote a vector that has the same amount of values as the feature vector $x's$
  - For θ and $x^{(i)}$, let us use $\theta^\top x^{(i)} = \langle\theta, x^{(i)}\rangle$ to denote the vector inner product
- **Step 3:** minimize the MSE metric to obtain the θ; call it $\hat{\boldsymbol{\theta}}$

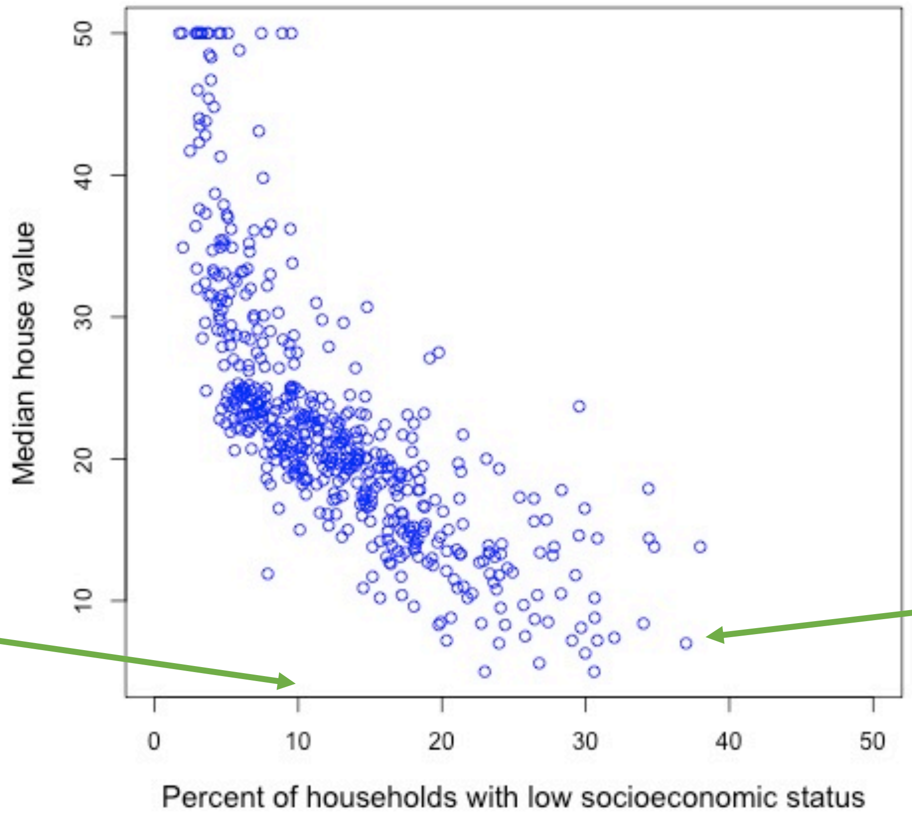
Illustrative example

- We will consider using a single feature, % lower status of the population (LSTAT), to predict median value of owner-occupied homes (MEDV)
- Note: this dataset is online at Kaggle https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset/input
- There are over ten different features in the dataset: CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B

Illustrative example

- This is a scatter plot of the dataset: LSTAT vs. MEDV

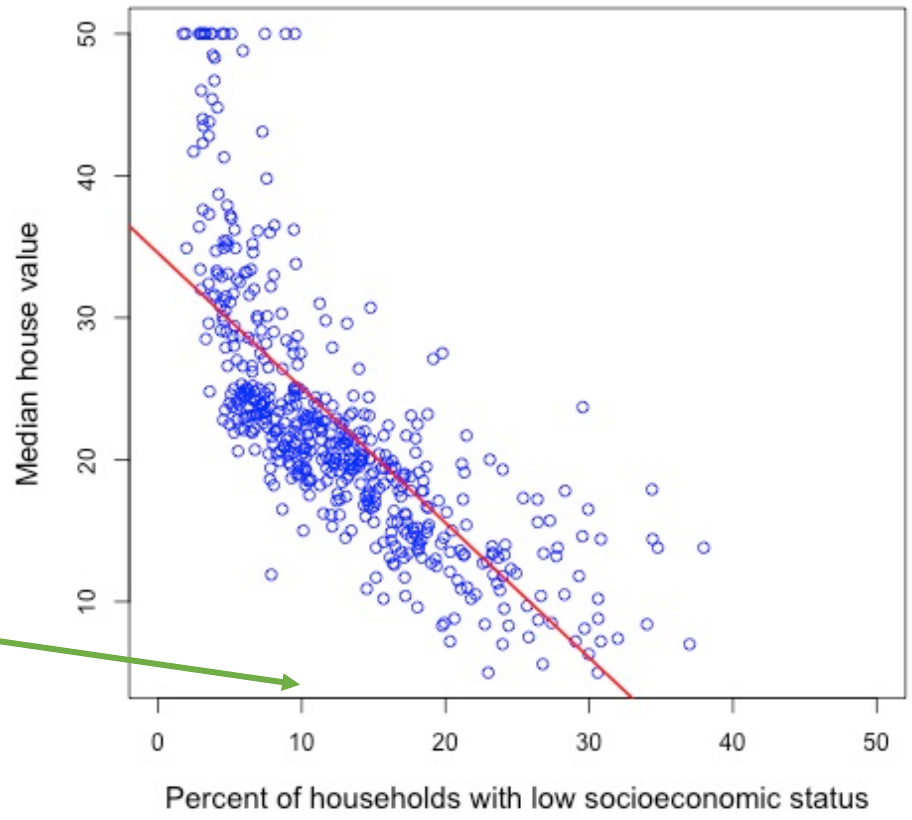
$x = 10$
$y = ?$



i-th sample
$(x^{(i)}, y^{(i)})$

Illustrative example

- **Fit a line by minimizing the MSE metric**



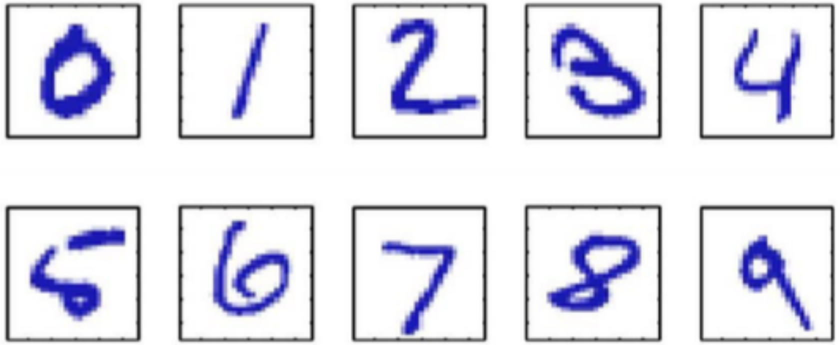
Fit a linear model to the data

$x = 10$

$y = ?$

Handwritten digit classification

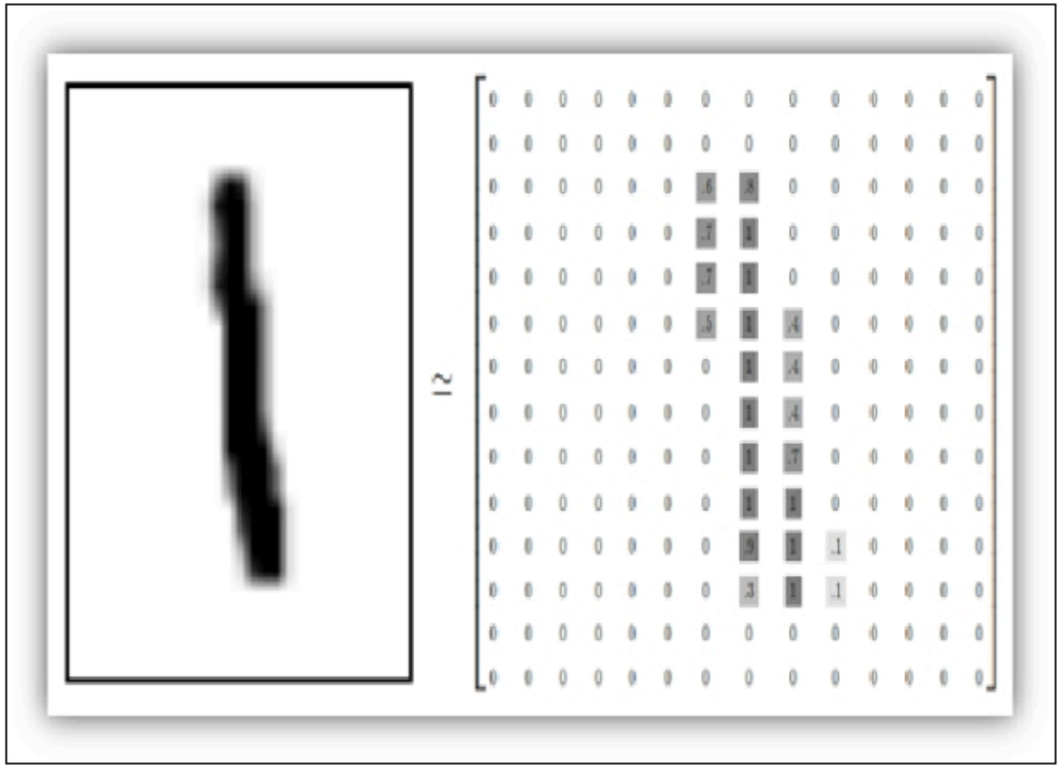
- Each image is 28 by 28 pixels, corresponding to a 784-dimensional vector



- **MNIST dataset**: 50k/5k/5k split, 60k in total
- Available online at https://yann.lecun.com/exdb/mnist/

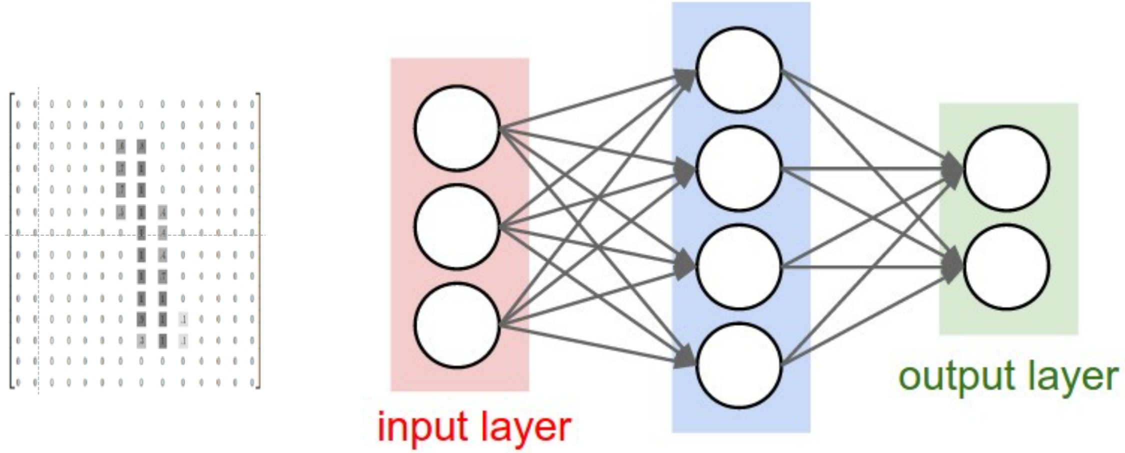
Preprocessing

- First need to map the image to an encoding: Here we'll use a matrix to represent of a black-n-white digit



Setting up a neural network

- **Feedforward neural networks:** Similar to convolutional neural networks but simpler

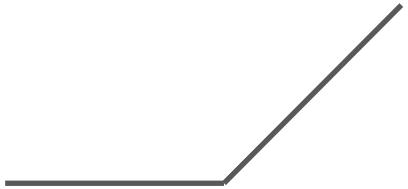


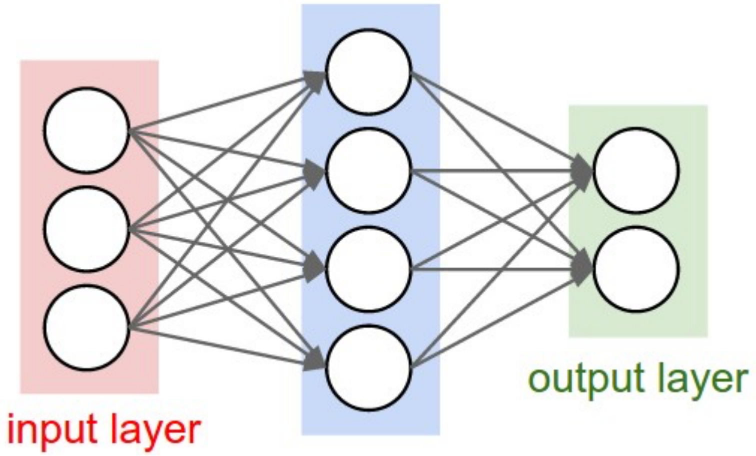
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

- **Input layer:** Receives the encoding of the handwritten digit 1
- **Hidden layer:** Maps the input encoding to a representation through a nonlinear transformation
- **Output layer:** Maps the representations to class probabilities

Feedforward neural networks

- Input layer, hidden layer, and output layer.
  - Nonlinear activation function: $\mathrm{ReLU}(x) = \max(x, 0)$





Understanding the working of neural networks

- Training split: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(n)}, y^{(n)})$

- In general, we may consider a loss function ℓ, that, takes a neural network f_W parameterized by W, and an input x, y, outputs a loss value for this input: $\ell(f_W(x), y)$

- Then, we may write the averaged training loss of a neural network f_W as follows

$$\hat{L}(f_W) := \frac{1}{n} \sum_{i=1}^{n} \ell(f_W(x^{(i)}), y^{(i)})$$

$\sum$ means we are taking a sum from $i = 1$ to n

Understanding the test loss

- **The test loss** of a neural network f_W should be measured on the test split. This must happen on a new test example (x, y) that the network has not seen during the training
- We may write the test loss as the expectation over an infinite population of unseen examples as follows

$$\boldsymbol{L(f_W) \coloneqq \mathbb{E}_{(x,y)\sim D}[\ell(f_W(x), y)]}$$

where D is the distribution of (x, y) pairs

- Mathematically speaking, the **expectation** of $\hat{L}(f_W)$ should be $\boldsymbol{L(f_W)}$
- Empirically, this means if we draw our dataset again, the test loss should be a faithful representation of the performance of the neural network, since we have not touched the test samples during training time

Some basic concepts

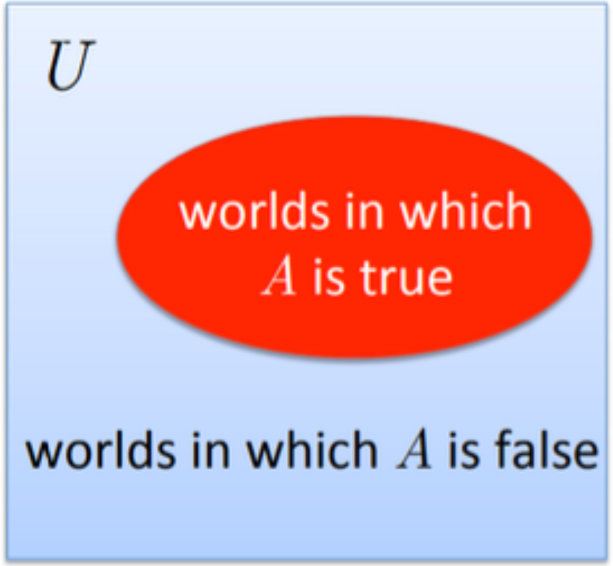
- **A random variable consists of a set of events and the corresponding probability value that each event will happen**
- Discrete random variables
  - Suppose you go fishing, the specie of the fish that you catch is a discrete rv
  - Suppose you are taking a flight, whether the flight will depart on time or not is a discrete rv
- Continuous random variables
  - Isotropic Gaussian with mean zero and variance 1: $\Pr[x] = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$ (check out Wikipedia page https://en.wikipedia.org/wiki/Gaussian_function)

Visualization

- Fishing: the **universe** $U = \{\text{salmon}, \text{tuna}, \text{yellowfin}\}$
  - Its area is the entire figure: $\Pr(A) = 1$

- More generally, $\Pr(A) = \text{Area of red oval}$

- Therefore: $\Pr(A) + \Pr(\neg A) = 1$



ID card verification

- Imagine that you are working on a project that requires detecting the bounding boxes of ID photos. Part of this is for authenticating a user's identity and extracting user's demographic information
- Without using machine learning: Hire a human annotation team or outsource to Amazon Mechanical Turk
- With machine learning: Build a prediction model to predict the region of the bounding boxes

Step 1

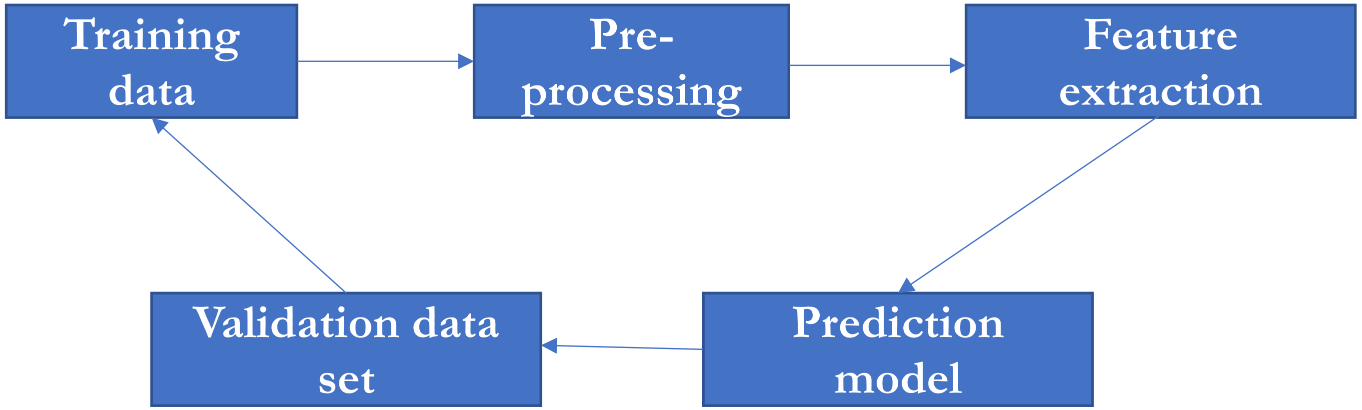
- Collect a dataset (a.k.a., data is the new oil)

Step 2

- Choose a model from your favorite package (pytorch, tensorflow, pandas, etc)

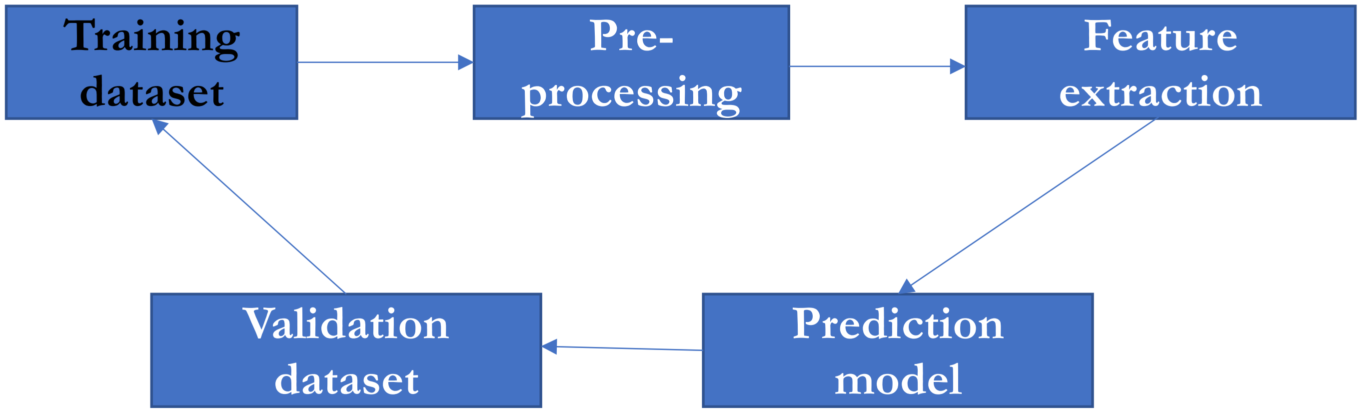
Step 3

- Train the model using the collected data



Step 3.1

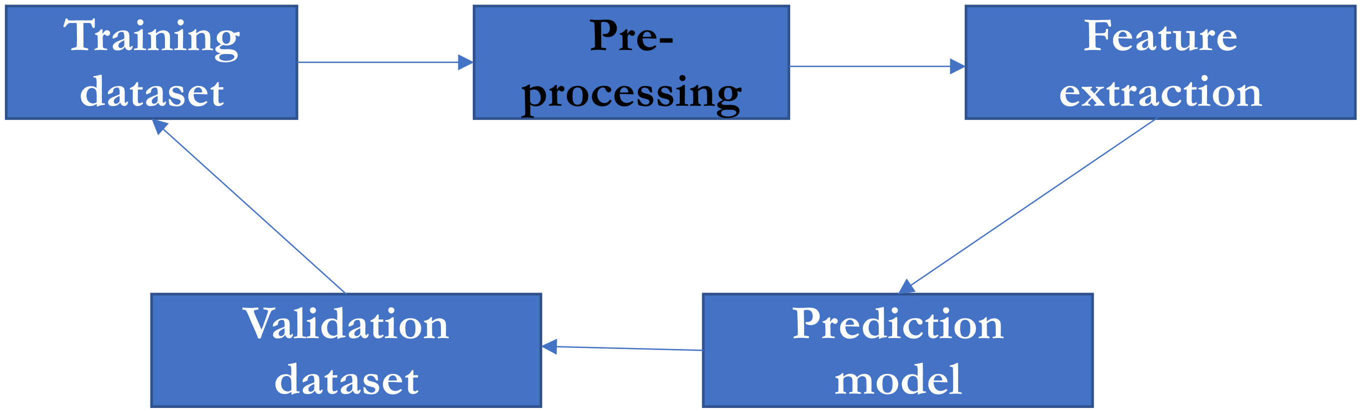
- Train the model using the collected data



- Training data
  - A slice of the data we have collected for training the model (Usually about 80%)

Step 3.2

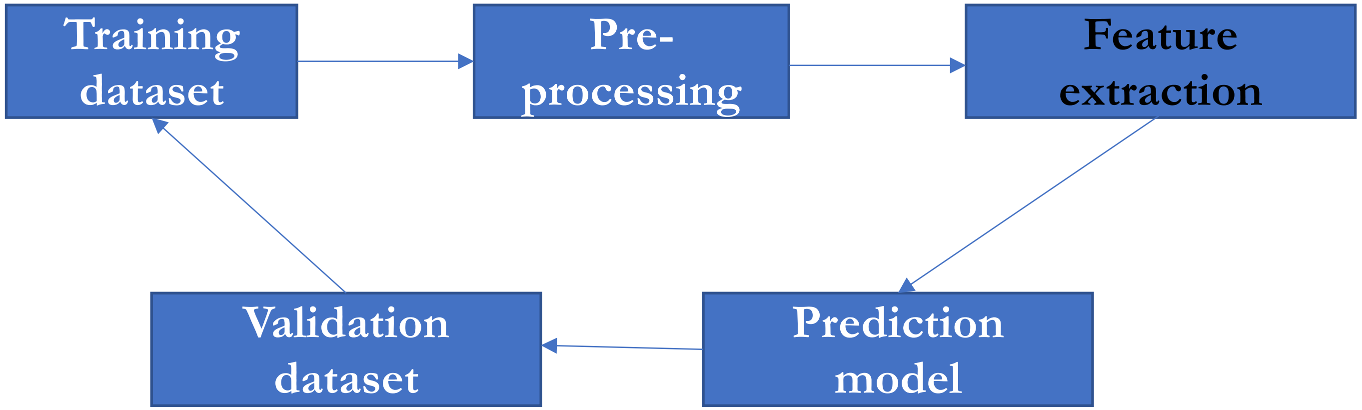
- Train the model using the collected data



- Preprocessing: Label the pixel positions within the bounding. The model will be trained to predict whether a pixel position is inside the bounding box

Step 3.3

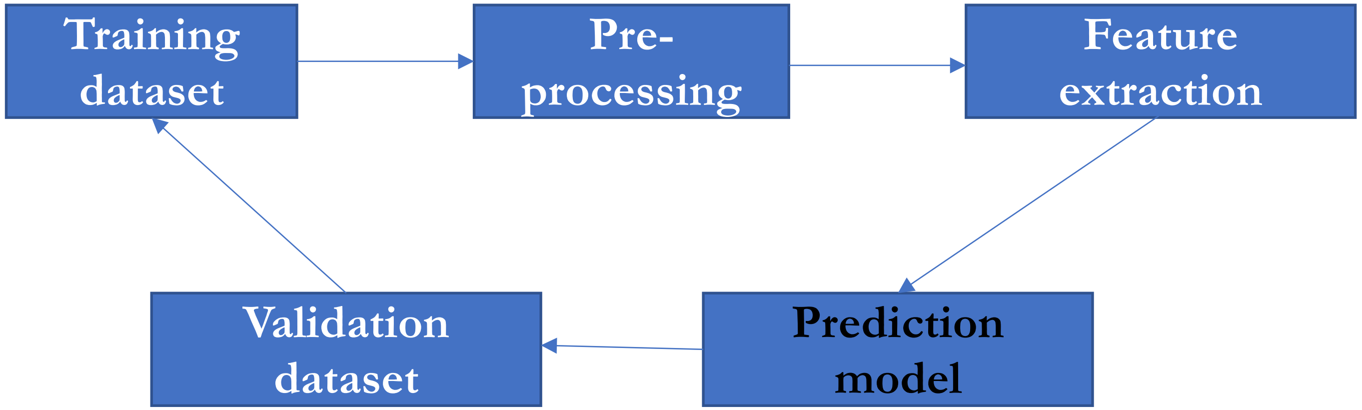
- Train the model using the collected data



- Feature extraction: Color, spatial pattern, etc

Step 3.4

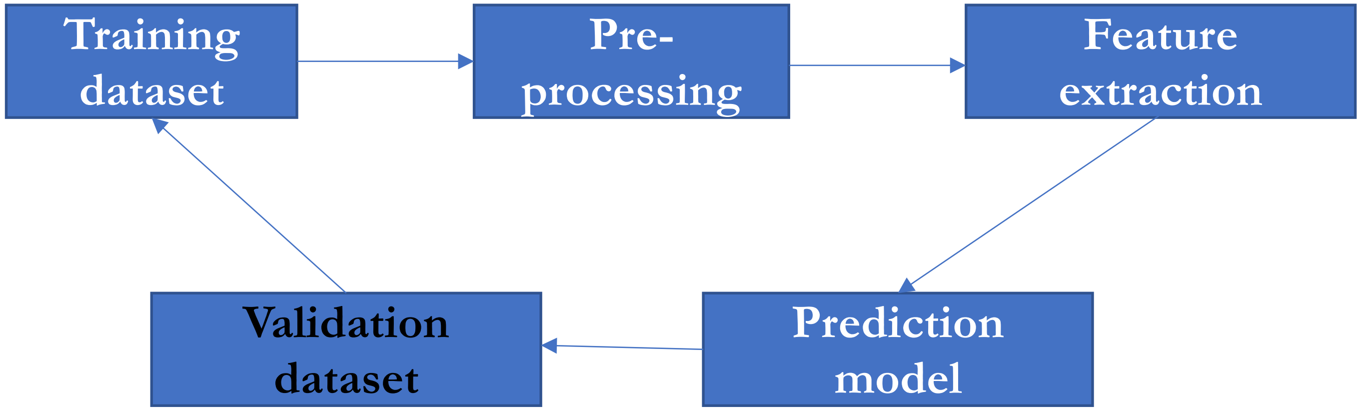
- Train the model using the collected data



- Learn a prediction model: using a convolutional neural network and update its weight parameters during training

Step 3.5

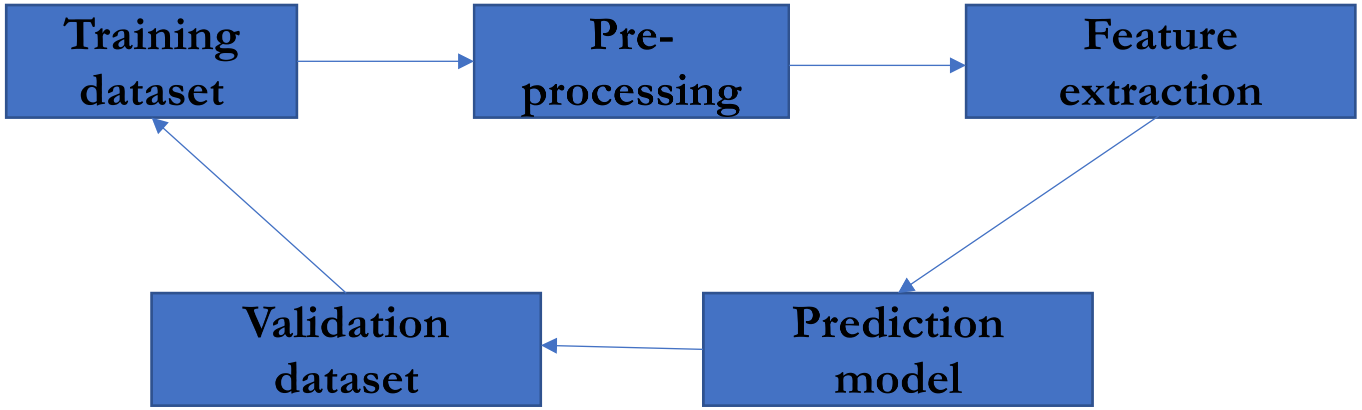
- Train the model using the collected data



- Evaluate on validation dataset: A slice of the collected dataset, usually about 10%

Step 4

- Train the model using the collected data



- **Performance on the test dataset:** usually about 10%

Questions

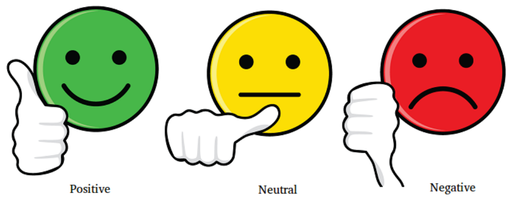
- Too little data?
  - Transfer learning: Apply a pretrained human face detection model
  - Synthetic data : Replace the human face of one ID with another human face
- Adaptation/new data/different states/countries?
  - Domain adaptation
  - Learning to learn (very relevant in robotics)

Lecture plan

- **Language modeling and naïve Bayes**

Naïve Bayes

- Sentiment prediction: "A very busy, but rewarding first week of the fall semester." The sentence consists of a list of eleven words: $\{A_1, A_2, \dots, A_{11}\}$



- Prediction rule: Choose the most likely hypothesis given the list of words
  - Hypothesis y is Positive, Neural, or Negative

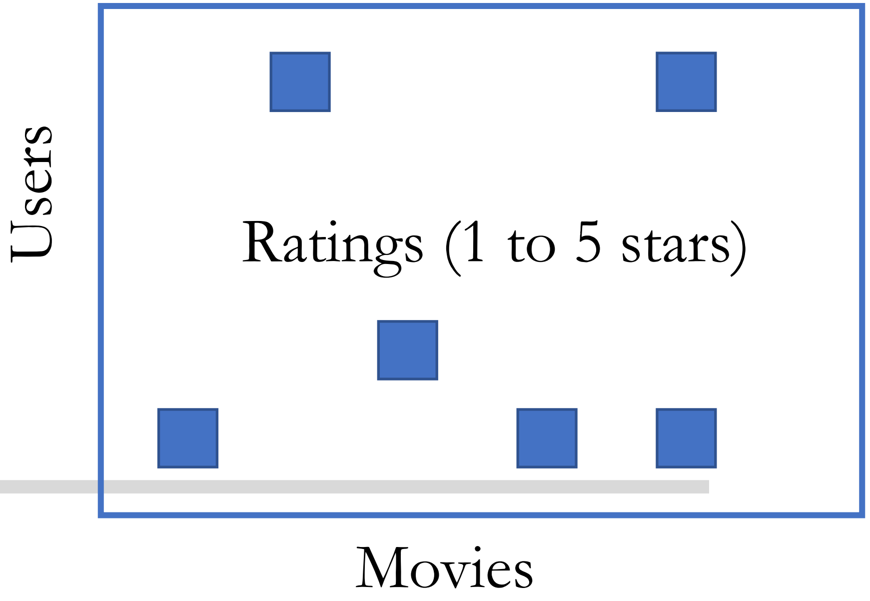
$$\arg\max_y \Pr(A_1, A_2, \dots, A_n | y) = \frac{\Pr(A_1, A_2, \dots, A_n, y)}{\Pr(y)}$$

- Naïve Bayes assumes conditional independence: Prior knowledge with a single word is easier to obtain

$$\Pr(A_1, A_2, \dots, A_n | y) \approx \Pr(A_1 | y) \cdot \Pr(A_2 | y) \cdot \dots \cdot \Pr(A_n | y) = \frac{\Pr(A_1, y)}{\Pr(y)} \cdot \dots \cdot \frac{Pr(A_1, y)}{Pr(y)}$$

Movie recommendation

- Predict the rating reference of a user for an unseen movie (similar for music)
  - Netflix challenge: $1 million for teams that beat their own prediction system by 10% over 100 million movie ratings
  - Most ratings are missing: $500K \times 18K = 9{,}000M \gg 100M$
- Can we build a machine learning model to predict missing ratings?
  - Learn users' preferences
  - Recommend movies to users
  - This is an example of unsupervised learning



Expected outcomes and deliverables

- You will learn state-of-the-art machine-learning techniques during the semester, and solid skills for implementing and applying them
- During the lectures, my goal is to help you understand the principles of how these methods work and when we should use which methods
  - Inevitably, there will be lots of statistics and linear algebra
  - I will try my best to explain concepts and methods with examples. **Feel free to ask questions**!
- The homework is designed to help you learn how to use these methods in practice
  - Most questions are coding questions based on Python, and lots of self-studying will be involved. To compensate for that, **we strongly encourage you to form groups to complete the homework**
  - **Every month** we'll have an in-class exercise session focused on debugging your coding skills. See course schedule for the dates

Expected outcomes of course project

- Learn how to develop and apply state-of-the-art language models to downstream tasks, such as prompt tuning, instruction following, QA, etc

Announcements

- **Office hours:** Tuesdays 12:30 PM to 1:30 PM at 177 Huntington Ave, #2211
- **Piazza:** See canvas
  - **Signup link** https://piazza.com/northeastern/fall2024/ds522020725202510
  - **Access code** 8128pzbevas
  - **Class link** https://piazza.com/northeastern/fall2024/ds522020725202510/home
- **Homework 1** will be released next week, due two weeks after. Submit on **gradescope. See syllabus for schedule**