Course Overview

Tal Linzen

CDS, NYU

Jan 25, 2022

Contents

Logistics

Course Staff

- Instructors:
  - Tal Linzen (Linguistics and Data Science)
  - He He (Computer Science and Data Science)
- Section leaders:
  - Vishakh Padmakumar
  - Colin Wan
- Graders:
  - Congyun Jin
  - Bella Lyu
  - Gavin Nan
  - Ziyi Xie
  - Namrata Mukhija

Logistics

- Class webpage: https://nyu-ds1003.github.io/spring2022
  - Course materials (lecture slides, homework assignments) will be made available on the website
- Announcements via Brightspace
- Discussion / questions on CampusWire: https://campuswire.com/c/G6A12AE75/feed
- Sign up to Gradescope to submit homework assignments (entry code **V8K3XW**)
- Office Hours:
  - The professors' office hours will be on Zoom (Tal: Tuesday 3:30-4:30 pm; He: Thursday 4:30-5:30 pm)
  - The section leaders' office hours will be in person (Vishakh: Wednesday 6-7 pm; Colin: Monday 1-2 pm; Room 204, 60 5th Ave)

Assessment

- 7 assignments ($1 \times 4\% + 6 \times 6\% = 40\%$)
- Two tests (60%)
  - Midterm Exam (30%) in Week 7 (March 8th), covering material up to Week 6
  - Final Exam (30%), schedule hasn't been announced yet, covering all material
- Typical grade distribution: A (40%), A- (20%), B+ (20%), B (10%), B- (5%), <B- (5%)

Homework

- Assignment 0: Help you get farmiliar with the format (not submitted or graded)
- First assignment out now – due on **Feb 1**
- Submit through Gradescope as a **PDF document**
- Late policy: Assignments are accepted up to **48 hours** late (see more details on website)
- You can collaborate with other students on the homework assignments, but please:
  - Write up the solutions and code on your own;
  - And list the names of the students you discussed each problem with.

Exams (60%)

- Exam format TBD: either in-person or submitted through Gradescope, like the assignments
- We'll make this decision based on NYU policy at the time – stay tuned
- Before each exam, we will post exams from previous years

Prerequisites

- DS-GA 1001: Introduction to Data Science
- DS-GA 1002: Statistical and Mathematical Methods
- Math
  - Multivariate Calculus
  - Linear Algebra
  - Probability Theory
  - Statistics
  - [Preferred] Proof-based linear algebra or real analysis
- Python programming (numpy)

Course Overview and Goals

Syllabus (Tentative)

13 weeks of instruction + 1 week midterm exam

- 2 weeks: introduction to **statistical learning theory**, **optimization**
- 2–3 weeks: **Linear** methods for binary classification and regression (also **kernel methods)**
- 2 weeks: **Probabilistic models**, **Bayesian** methods
- 1 week: **Multiclass** classification and introduction to **structured prediction**
- 3–4 weeks: **Nonlinear** methods (**trees**, **ensemble** methods, and **neural networks**)
- 2 weeks: **Unsupervised** learning: **clustering** and **latent variable** models
- More detailed schedule on the course website (still subject to change)
- Certain applications and practical algorithms may be covered in the labs

The high level goals of the class

- Our focus will be on the fundamental building blocks of machine learning
- ML methods have a lot of names; our goal is for you to notice that

  fancy new method A "is just" familiar thing B + familiar thing C + tweak D

  - SVM "**is just**" ERM with hinge loss with ℓ_2 regularization
  - Pegasos "**is just**" SVM with SGD with a particular step size rule
  - Random forests "**are just**" bagging with trees, with a different approach to choosing splitting variables

The level of the class

- We will learn how to implement each ML algorithm **from scratch** using numpy alone, without any ML libraries.
- Once we have implemented an algorithm from scratch once, we will use the sklearn version.

What is Machine Learning

Based on David Rosenberg and He He's materials

Tal Linzen

Center for Data Science, NYU

Jan 25, 2022

Machine Learning Problems

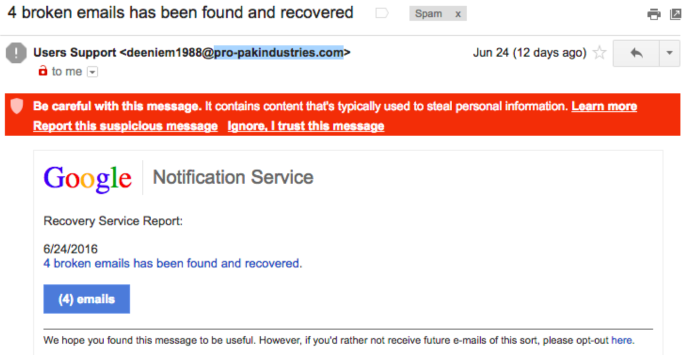
Typically our goal is to solve a prediction problem of the format:

- Given an **input** x,
- **Predict** an **output** y.

We'll start with a few canonical examples.

Example: Spam Detection

- **Input:** Incoming email



- **Output:** “SPAM” or “NOT SPAM”
- This is a **binary classification** problem: there are two possible outputs.

Example: Medical Diagnosis

- **Input**: Symptoms (fever, cough, fast breathing, shaking, nausea, ...)
- **Output:** Diagnosis (pneumonia, flu, common cold, bronchitis, ...)
- A **multiclass classification** problem: choosing an output out of a *discrete* set of possible outputs.

How do we express uncertainty about the output?

- **Probabilistic classification** or **soft classification**:

$$\begin{aligned} \mathbb{P}(\text{pneumonia}) &= 0.7 \\ \mathbb{P}(\text{flu}) &= 0.2 \\ \vdots \quad & \quad \vdots \end{aligned}$$

Example: Predicting a Stock Price

- **Input:** History of the stock's prices
- **Output:** The price of the stock at the close of the next day
- This is called a **regression** problem (for historical reasons): the output is *continuous*.

Comparison to Rule-Based Approaches (Expert Systems)

- Consider the problem of medical diagnosis.
    1. Talk to experts (in this case, medical doctors).
    2. Understand how the experts come up with a diagnosis.
    3. Implement this process as an algorithm (a **rule-based system**): e.g., a set of symptoms $\rightarrow$ a particular diagnosis.
    4. Potentially use logical deduction to infer new rules from the rules that are stored in the knowledge base.

Rule-Based Approach

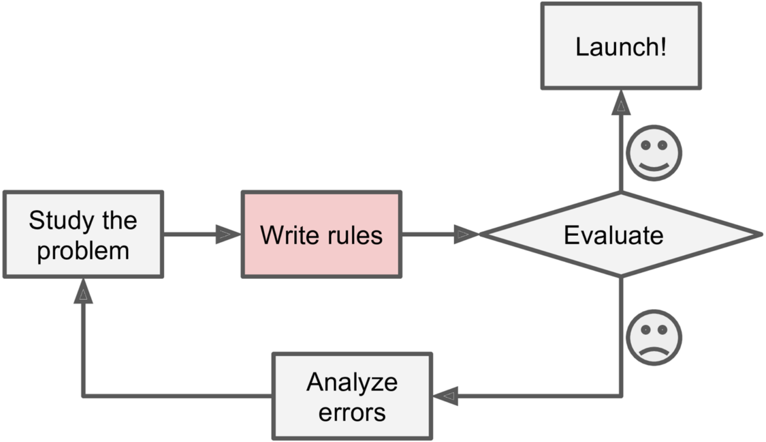


Fig 1-1 from *Hands-On Machine Learning with Scikit-Learn and TensorFlow* by Aurelien Geron (2017).

Advantages of Rule-Based Approaches

- Leverage existing domain expertise.
- Generally **interpretable**: We can describe the rule to another human
- Produce reliable answers for the scenarios that are included in the knowledge bases.

Limitations of Rule-Based Systems

- Labor intensive to build: experts' time is expensive.
- Rules work very well for areas they cover, but often do not **generalize** to unanticipated input combinations.
- Don't naturally handle uncertainty.

The Machine Learning Approach

- Instead of explicitly engineering the process that a human expert would use to make the decision...
- We have the machine **learn** on its own from inputs and outputs (decisions).
- We provide **training data**: many examples of (input x , output y) pairs, e.g.
  - A set of videos, and whether or not each has a cat in it.
  - A set of emails, and whether or not each one should go to the spam folder.
- Learning from training data of this form (inputs and outputs) is called **supervised learning**.

Machine Learning Algorithm

- A **machine learning algorithm** learns from the training data:
  - **Input**: Training Data (e.g., emails x and their labels y)
  - **Output**: A prediction function that produces output y given input x.
- The goal of machine learning is to find the "best" (to be defined) prediction function **automatically, based on the training data**
- The success of ML depends on
  - The availability of large amounts of data;
  - **Generalization** to unseen samples (the test set): just memorizing the training set will not be useful.

Machine Learning Approach

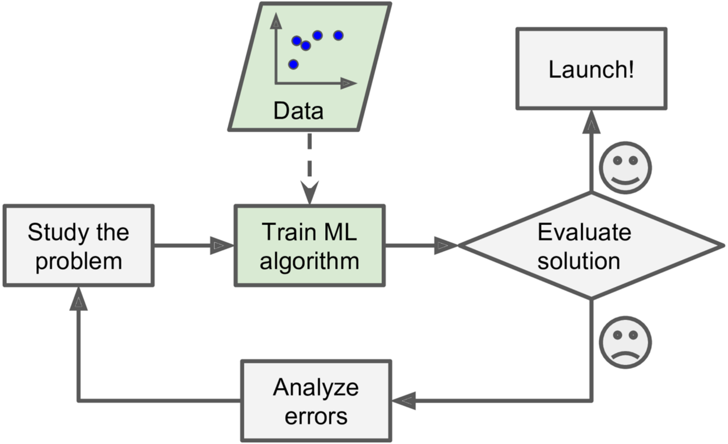


Fig 1-2 from *Hands-On Machine Learning with Scikit-Learn and TensorFlow* by Aurelien Geron (2017).

Key concepts

- The most common **ML problem types**:

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)
  - Regression

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)
  - Regression
- **Prediction function**: predicts output y (e.g. spam or not?) given input x (e.g. email)

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)
  - Regression
- **Prediction function**: predicts output y (e.g. spam or not?) given input x (e.g. email)
- **Training data**: a set of (input x, output y) pairs

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)
  - Regression
- **Prediction function**: predicts output y (e.g. spam or not?) given input x (e.g. email)
- **Training data**: a set of (input x, output y) pairs
- **Supervised learning algorithm**: takes training data and produces a prediction function

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)
  - Regression
- **Prediction function**: predicts output y (e.g. spam or not?) given input x (e.g. email)
- **Training data**: a set of (input x, output y) pairs
- **Supervised learning algorithm**: takes training data and produces a prediction function
- Beyond prediction

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)
  - Regression
- **Prediction function**: predicts output y (e.g. spam or not?) given input x (e.g. email)
- **Training data**: a set of (input x, output y) pairs
- **Supervised learning algorithm**: takes training data and produces a prediction function
- Beyond prediction
  - **Unsupervised learning**: finding structures in data, e.g. clustering

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)
  - Regression
- **Prediction function**: predicts output y (e.g. spam or not?) given input x (e.g. email)
- **Training data**: a set of (input x, output y) pairs
- **Supervised learning algorithm**: takes training data and produces a prediction function
- Beyond prediction
  - **Unsupervised learning**: finding structures in data, e.g. clustering
  - **Reinforcement learning**: optimizing long-term objective, e.g. Go

Key concepts

- The most common **ML problem types**:
  - Classification (binary and multiclass)
  - Regression
- **Prediction function**: predicts output y (e.g. spam or not?) given input x (e.g. email)
- **Training data**: a set of (input x, output y) pairs
- **Supervised learning algorithm**: takes training data and produces a prediction function
- Beyond prediction
  - **Unsupervised learning**: finding structures in data, e.g. clustering
  - **Reinforcement learning**: optimizing long-term objective, e.g. Go
  - **Representation learning**: learning good features of real-world objects, e.g. text

Core Questions in Machine Learning

Given any task, the following questions need to be answered:

- **Modeling**: What class of prediction functions are we considering?
- **Learning**: How do we learn the "best" prediction function in this class from our training data?
- **Inference**: How do we compute the output of the prediction function for a new input?

Statistical Learning Theory

Based on David Rosenberg and He He's materials

Tal Linzen

CDS, NYU

Jan 25, 2022

Decision theory

- In data science problems, we generally need to:
  - Make a decision:
    - Move email to spam folder?

Decision theory

- In data science problems, we generally need to:
  - Make a decision:
    - Move email to spam folder?
  - Take an action:
    - In a self-driving car, make a right turn
    - Reject the hypothesis that $\theta = 0$ (classical statistics)

Decision theory

- In data science problems, we generally need to:
  - Make a decision:
    - Move email to spam folder?
  - Take an action:
    - In a self-driving car, make a right turn
    - Reject the hypothesis that $\theta = 0$ (classical statistics)
  - Produce some output:
    - Whose face is it in the image?
    - The Hindi translation of a Japanese input sentence

Decision theory

- In data science problems, we generally need to:
  - Make a decision:
    - Move email to spam folder?
  - Take an action:
    - In a self-driving car, make a right turn
    - Reject the hypothesis that $\theta = 0$ (classical statistics)
  - Produce some output:
    - Whose face is it in the image?
    - The Hindi translation of a Japanese input sentence
  - Predicting where a storm will be in an hour (what forms of output are possible here?)

Decision theory

- In data science problems, we generally need to:
  - Make a decision:
    - Move email to spam folder?
  - Take an action:
    - In a self-driving car, make a right turn
    - Reject the hypothesis that $\theta = 0$ (classical statistics)
  - Produce some output:
    - Whose face is it in the image?
    - The Hindi translation of a Japanese input sentence
  - Predicting where a storm will be in an hour (what forms of output are possible here?)
  - An **action** is the generic term for what is produced by our system.

Inputs

We make our decision based on context:

- Inputs [ML]
- Covariates [Statistics]

Examples of inputs

- A picture
- The location of the storm in the last 24 hours, other weather-related measurements
- A search query

Outcome

Inputs are often paired with **outputs** or **labels**.

Examples of outcomes/outputs/labels

- Whether or not the picture actually contains an animal
- The storm's location one hour after they query
- Which, if any, of the suggested URLs were selected

Evaluation Criterion

Decision theory is about finding "optimal" actions, under various definitions of optimality.

Examples of Evaluation Criteria

- Is the classification correct?
- Does the transcription exactly match the spoken words?
  - Should we give partial credit (for getting only some of the words right)? How?
- How far is the storm from the predicted location? (If we're producing a point estimate)
- How likely is the storm's actual location under the predicted distribution? (If we're doing density prediction)

Typical Sequence of Events

Many problem domains can be formalized as follows:

1. Observe input x.
2. Take action a.
3. Observe outcome y.
4. Evaluate action in relation to the outcome.

Three spaces:

- Input space: $\mathcal{X}$
- Action space: $\mathcal{A}$
- Outcome space: $\mathcal{Y}$

Formalization

Prediction Function

A **prediction function** (or **decision function**) gets input $x \in \mathcal{X}$ and produces an action $a \in \mathcal{A}$:

$$\begin{aligned} f: \quad \mathcal{X} &\to \mathcal{A} \\ x &\mapsto f(x) \end{aligned}$$

Formalization

Prediction Function

A **prediction function** (or **decision function**) gets input $x \in \mathcal{X}$ and produces an action $a \in \mathcal{A}$:

$$\begin{aligned} f: \quad \mathcal{X} & \rightarrow \mathcal{A} \\ x & \mapsto f(x) \end{aligned}$$

Loss Function

A **loss function** evaluates an action in the context of the outcome y.

$$\begin{aligned} \ell: \quad \mathcal{A} \times \mathcal{Y} & \rightarrow \mathbf{R} \\ (a, y) & \mapsto \ell(a, y) \end{aligned}$$

Evaluating a Prediction Function

Goal: Find the optimal prediction function.

Intuition: If we can evaluate how good a prediction function is, we can turn this into an optimization problem.

- The loss function ℓ evaluates a *single* action
- How do we evaluate the prediction function *as a whole*?
- We will use the standard **statistical learning theory** framework.

Define a space where the prediction function is applicable

- Assume there is a **data generating distribution** $P_{\mathcal{X}\times\mathcal{Y}}$.
- All input/output pairs (x, y) are generated i.i.d. from $P_{\mathcal{X}\times\mathcal{Y}}$.

One common desideratum is to have a prediction function $f(x)$ that "does well on average":

$$\ell(f(x), y) \text{ is usually small, in some sense}$$

How can we formalize this?

Risk

Definition

The **risk** of a prediction function $f : \mathcal{X} \to \mathcal{A}$ is

$$R(f) = \mathbb{E}_{(x,y) \sim P_{\mathcal{X} \times \mathcal{Y}}} \left[\ell(f(x), y) \right].$$

In words, it's the **expected loss** of f over $P_{\mathcal{X} \times \mathcal{Y}}$.

We can't actually compute the risk function:

Since we don't know $P_{\mathcal{X} \times \mathcal{Y}}$, we cannot compute the expectation.
But we can **estimate** it.

The Bayes Prediction Function

Definition

A **Bayes prediction function** $f^* : \mathcal{X} \to \mathcal{A}$ is a function that achieves the *minimal risk* among all possible functions:

$$f^* \in \underset{f}{\arg\min}\, R(f),$$

where the minimum is taken over all functions from $\mathcal{X}$ to $\mathcal{A}$.

- The risk of a Bayes prediction function is called the **Bayes risk**.
- A Bayes prediction function is often called the "**target function**", since it's the best prediction function we can possibly produce.

Example: Multiclass Classification

- Spaces: $\mathcal{A} = \mathcal{Y} = \{1, \ldots, k\}$
- 0-1 loss:

$$\ell(a, y) = 1(a \neq y) := \begin{cases} 1 & \text{if } a \neq y \\ 0 & \text{otherwise.} \end{cases}$$

Example: Multiclass Classification

- Spaces: $\mathcal{A} = \mathcal{Y} = \{1, \ldots, k\}$
- 0-1 loss:

$$\ell(a, y) = 1(a \neq y) := \begin{cases} 1 & \text{if } a \neq y \\ 0 & \text{otherwise.} \end{cases}$$

- Risk:

$$\begin{aligned} R(f) &= \mathbb{E}\left[1(f(x) \neq y)\right] \quad = 0 \cdot \mathbb{P}(f(x) = y) + 1 \cdot \mathbb{P}(f(x) \neq y) \\ &= \mathbb{P}(f(x) \neq y), \end{aligned}$$

  which is just the misclassification error rate.

- The Bayes prediction function returns the most likely class:

$$f^*(x) \in \underset{1 \leqslant c \leqslant k}{\arg\max}\, \mathbb{P}(y = c \mid x)$$

But we can't compute the risk!

- Can't compute $R(f) = \mathbb{E}\left[\ell(f(x), y)\right]$ because we **don't know** $P_{\mathcal{X}\times\mathcal{Y}}$.

But we can't compute the risk!

- Can't compute $R(f) = \mathbb{E}\left[\ell(f(x), y)\right]$ because we **don't know** $P_{\mathcal{X}\times\mathcal{Y}}$.
- One thing we can do in ML/statistics/data science is **estimate** it:

But we can't compute the risk!

- Can't compute $R(f) = \mathbb{E}[\ell(f(x), y)]$ because we **don't know** $P_{\mathcal{X} \times \mathcal{Y}}$.
- One thing we can do in ML/statistics/data science is **estimate** it:

Assume we have sample data:

Let $\mathcal{D}_n = ((x_1, y_1), \ldots, (x_n, y_n))$ be drawn i.i.d. from $\mathcal{P}_{\mathcal{X} \times \mathcal{Y}}$.

But we can't compute the risk!

- Can't compute $R(f) = \mathbb{E}[\ell(f(x), y)]$ because we **don't know** $P_{\mathcal{X}\times\mathcal{Y}}$.
- One thing we can do in ML/statistics/data science is **estimate** it:

Assume we have sample data:

Let $\mathcal{D}_n = ((x_1, y_1), \ldots, (x_n, y_n))$ be drawn i.i.d. from $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$.

- We draw inspiration from the strong law of large numbers:
  If $z_1, \ldots, z_n$ are i.i.d. with expected value $\mathbb{E}z$, then

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} z_i = \mathbb{E}z,$$

with probability 1.

The Empirical Risk

Let $\mathcal{D}_n = ((x_1, y_1), \ldots, (x_n, y_n))$ be drawn i.i.d. from $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$.

Definition

The **empirical risk** of $f : \mathcal{X} \to \mathcal{A}$ with respect to $\mathcal{D}_n$ is

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i).$$

By the strong law of large numbers,

$$\lim_{n\to\infty} \hat{R}_n(f) = R(f),$$

almost surely.

Empirical Risk Minimization

Definition

A function $\hat{f}$ is an **empirical risk minimizer** if

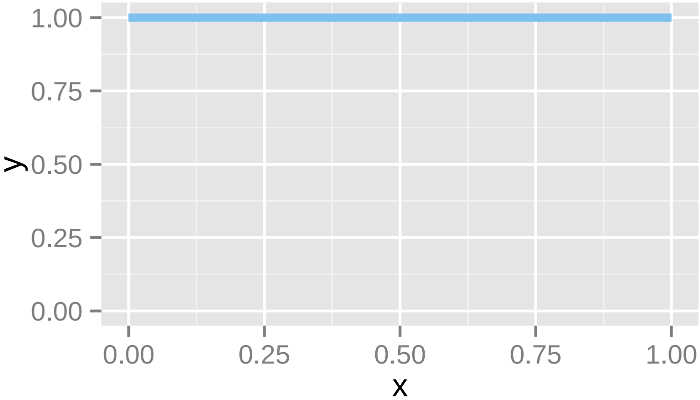
$$\hat{f} \in \arg\min_{f} \hat{R}_n(f),$$

where the minimum is taken over all functions $f : \mathcal{X} \to \mathcal{A}$.

- In an ideal world we'd want to find the risk minimizer.
- Is the empirical risk minimizer close enough?
- In practice, we always only have a finite sample...

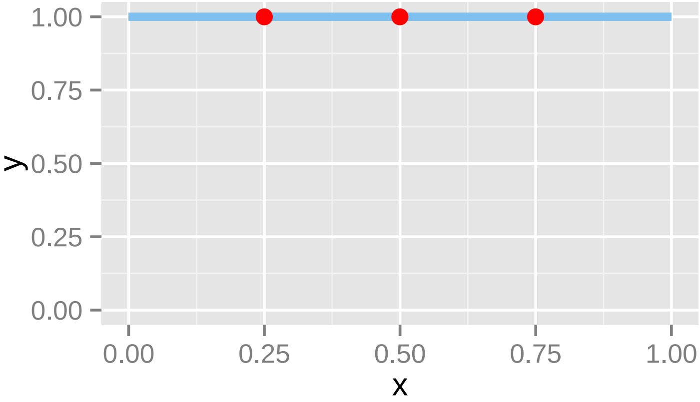
Empirical Risk Minimization

- $P_{\mathcal{X}} = \text{Uniform}[0,1]$, $Y \equiv 1$ (i.e. Y is always 1).
- A plot of $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$:



Empirical Risk Minimization

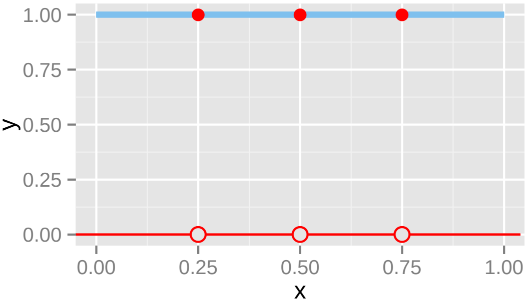
$P_{\mathcal{X}} = \text{Uniform}[0,1]$, $Y \equiv 1$ (i.e. Y is always 1).



A sample of size 3 from $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$.

Empirical Risk Minimization

$P_{\mathcal{X}} = \text{Uniform}[0,1]$, $Y \equiv 1$ (i.e. Y is always 1).

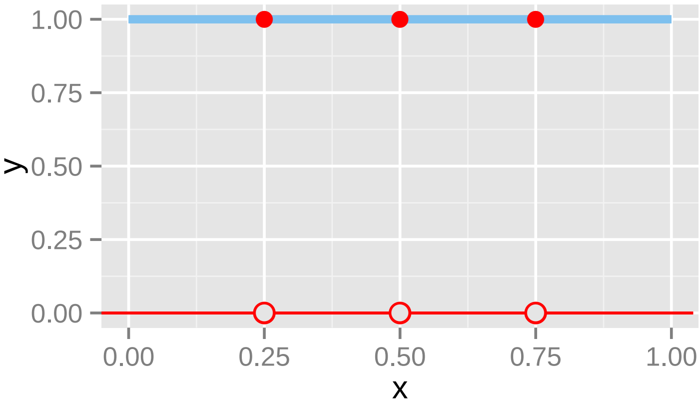


A proposed prediction function:

$$\hat{f}(x) = 1(x \in \{0.25, 0.5, 0.75\}) = \begin{cases} 1 & \text{if } x \in \{0.25, .5, .75\} \\ 0 & \text{otherwise} \end{cases}$$

Empirical Risk Minimization

$P_{\mathcal{X}} = \text{Uniform}[0,1]$, $Y \equiv 1$ (i.e. Y is always 1).



Under either the square loss or the 0/1 loss, $\hat{f}$ has Empirical Risk = 0 and Risk = 1.

Empirical Risk Minimization

- In this case, ERM led to a function f that just memorized the data.
- How can we improve **generalization** from the training inputs to new inputs?
- We need to smooth things out somehow!
  - A lot of modeling is about spreading and extrapolating information from one part of the input space $\mathcal{X}$ into unobserved parts of the space.
- One approach is **constrained ERM**:
  - Instead of minimizing empirical risk over *all* prediction functions,
  - We constrain our search to a particular subset of the space of functions, called a **hypothesis space**.

Hypothesis Spaces

Definition

A **hypothesis space** $\mathcal{F}$ is a set of prediction functions $\mathcal{X} \to \mathcal{A}$ that we consider when applying ERM.

Desirable properties of a hypothesis space:

- Includes only those functions that have the desired "regularity", e.g. smoothness, simplicity
- Easy to work with (e.g., we have efficient algorithms to find the best function within the space)

Most applied work is about designing good hypothesis spaces for specific tasks.

Constrained Empirical Risk Minimization

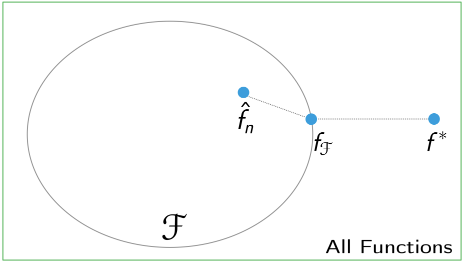
- Given a hypothesis space $\mathcal{F}$, a set of prediction functions mapping $\mathcal{X} \to \mathcal{A}$,
- An **empirical risk minimizer** (ERM) in $\mathcal{F}$ is a function $\hat{f}_n$ such that

$$\hat{f}_n \in \underset{f \in \mathcal{F}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

- A **Risk minimizer** in $\mathcal{F}$ is a function $f^*_{\mathcal{F}} \in \mathcal{F}$ such that

$$f^*_{\mathcal{F}} \in \underset{f \in \mathcal{F}}{\arg\min} \mathbb{E}\left[\ell(f(x), y)\right].$$

Excess Risk Decomposition



$$f^* = \arg\min_f \mathbb{E}\left[\ell(f(x), y)\right]$$

$$f_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathbb{E}\left[\ell(f(x), y)\right]$$

$$\hat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

- **Approximation error** (of $\mathcal{F}$) $= R(f_{\mathcal{F}}) - R(f^*)$
- **Estimation error** (of $\hat{f}_n$ in $\mathcal{F}$) $= R(\hat{f}_n) - R(f_{\mathcal{F}})$

Excess Risk Decomposition for ERM

Definition

The **excess risk** compares the risk of f to the Bayes optimal f^*:

$$\textbf{Excess Risk}(f) = R(f) - R(f^*)$$

- Can excess risk ever be negative?

The excess risk of the ERM $\hat{f}_n$ can be decomposed:

$$\begin{aligned}\textbf{Excess Risk}(\hat{f}_n) &= R(\hat{f}_n) - R(f^*) \\ &= \underbrace{R(\hat{f}_n) - R(f_{\mathcal{F}})}_{\text{estimation error}} + \underbrace{R(f_{\mathcal{F}}) - R(f^*)}_{\text{approximation error}}.\end{aligned}$$

- There is a tradeoff between estimation error and approximation error

Approximation Error

Approximation error $R(f_{\mathcal{F}}) - R(f^*)$ is

- a property of the class $\mathcal{F}$
- the penalty for restricting to $\mathcal{F}$ (rather than considering all possible functions)

Bigger $\mathcal{F}$ mean *smaller* approximation error.

Concept check: Is approximation error a random or non-random variable?

Estimation Error

Estimation error $R(\hat{f}_n) - R(f_{\mathcal{F}})$

- is the performance hit for choosing f using finite training data
- is the performance hit for minimizing empirical risk rather than true risk

With *smaller* $\mathcal{F}$ we expect *smaller* estimation error.

Under typical conditions: "With infinite training data, estimation error goes to zero."

Concept check: Is estimation error a random or non-random variable?

ERM in Practice

- What have we been glossing over by writing "argmin"?

ERM in Practice

- What have we been glossing over by writing “argmin”?
- In practice, we need a method to find $\hat{f}_n \in \mathcal{F}$: this can be very difficult!

ERM in Practice

- What have we been glossing over by writing "argmin"?
- In practice, we need a method to find $\hat{f}_n \in \mathcal{F}$: this can be very difficult!
- For nice choices of loss functions and classes $\mathcal{F}$, we can get arbitrarily close to the exact minimizer
  - But that takes time – is it always worth it?

ERM in Practice

- What have we been glossing over by writing "argmin"?
- In practice, we need a method to find $\hat{f}_n \in \mathcal{F}$: this can be very difficult!
- For nice choices of loss functions and classes $\mathcal{F}$, we can get arbitrarily close to the exact minimizer
    - But that takes time – is it always worth it?
- For some hypothesis spaces (e.g. neural networks), we don't know how to find $\hat{f}_n \in \mathcal{F}$.

Optimization Error

- In practice, we don't find the ERM $\hat{f}_n \in \mathcal{F}$.
- We find $\tilde{f}_n \in \mathcal{F}$ that we hope is good enough.
- **Optimization error:** If $\tilde{f}_n$ is the function our optimization method returns, and $\hat{f}_n$ is the empirical risk minimizer, then

$$\text{Optimization Error} \; = \; R(\tilde{f}_n) - R(\hat{f}_n).$$

Error Decomposition in Practice

- Excess risk decomposition for function $\tilde{f}_n$ returned by an optimization algorithm in practice:

$$\begin{aligned} \textbf{Excess Risk}(\tilde{f}_n) &= R(\tilde{f}_n) - R(f^*) \\ &= \underbrace{R(\tilde{f}_n) - R(\hat{f}_n)}_{\text{optimization error}} + \underbrace{R(\hat{f}_n) - R(f_{\mathcal{F}})}_{\text{estimation error}} + \underbrace{R(f_{\mathcal{F}}) - R(f^*)}_{\text{approximation error}} \end{aligned}$$

- It would be nice to observe the error decomposition for a practical $\tilde{f}_n$!
- How would we address each type of error?
- Why is this usually impossible?
- But we could constuct an artificial example, where we know $P_{\mathcal{X}\times\mathcal{Y}}$ and f^* and $f_{\mathcal{F}}$...

ERM Overview

- Given a loss function $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbf{R}$,

ERM Overview

- Given a loss function $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbf{R}$,
- Choose a hypothesis space $\mathcal{F}$.

ERM Overview

- Given a loss function $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbf{R}$,
- Choose a hypothesis space $\mathcal{F}$.
- Use an optimization method to find an empirical risk minimizer $\hat{f}_n \in \mathcal{F}$:

$$\hat{f}_n = \underset{f \in \mathcal{F}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

ERM Overview

- Given a loss function $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbf{R}$,
- Choose a hypothesis space $\mathcal{F}$.
- Use an optimization method to find an empirical risk minimizer $\hat{f}_n \in \mathcal{F}$:

$$\hat{f}_n = \underset{f \in \mathcal{F}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

- (Or find a $\tilde{f}_n$ that comes close to $\hat{f}_n$)

ERM Overview

- Given a loss function $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbf{R}$,
- Choose a hypothesis space $\mathcal{F}$.
- Use an optimization method to find an empirical risk minimizer $\hat{f}_n \in \mathcal{F}$:

$$\hat{f}_n = \underset{f \in \mathcal{F}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

- (Or find a $\tilde{f}_n$ that comes close to $\hat{f}_n$)
- The data scientist's job:
  - Choose $\mathcal{F}$ that balances approximation and estimation error.
  - As we get more training data, we can use a bigger $\mathcal{F}$.