Statistical Learning Theory

He He

Slides based on Lecture 1b, 1c from David Rosenberg's course material.

CDS, NYU

Feb 2, 2021

Contents

Decision Theory

What types of problems are we solving?

- In data science problems, we generally need to:
  - Make a decision
  - Take an action
  - Produce some output
- Have some evaluation criterion

Actions

An **action** is the generic term for what is produced by our system.

Examples of Actions

- Produce a 0/1 classification (classical ML)
- Reject hypothesis that $\theta = 0$ (classical Statistics)
- Generate text (image captioning, speech recognition, machine translation)
- What's an action for predicting where a storm will be in 3 hours?

Inputs

In order to make the decision, we typically have additional context:

- Inputs [ML]
- Covariates [Statistics]

Examples of Inputs

- A picture
- A storm's historical location and other weather data
- 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 query
- Which, if any, of suggested the 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 text transcription exactly match the spoken words?
  - Should we give partial credit? How?
- How far is the storm from the predicted location? (for point prediction)
- How likely is the storm's location under the predicted distribution? (for 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 $a \in \mathcal{A}$:

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

Loss Function

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

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

Evaluating a Prediction Function

Goal: find the optimal prediction function

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

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

Statistical Learning Theory

Setup for Statistical Learning Theory

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 $P_{\mathcal{X}\times\mathcal{Y}}$.

Want 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 $P_{\mathcal{X}\times\mathcal{Y}}$.

Risk function cannot be computed

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:

$$f^* \in \arg\min_f R(f),$$

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.

- Bayes prediction function is just the assignment to 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 $R(f) = \mathbb{E}[\ell(f(x), y)]$ because we **don't know** $P_{\mathcal{X} \times \mathcal{Y}}$.

But we can't compute the risk!

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

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

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

assume we have sample data.

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

- Let's draw some 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.

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

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

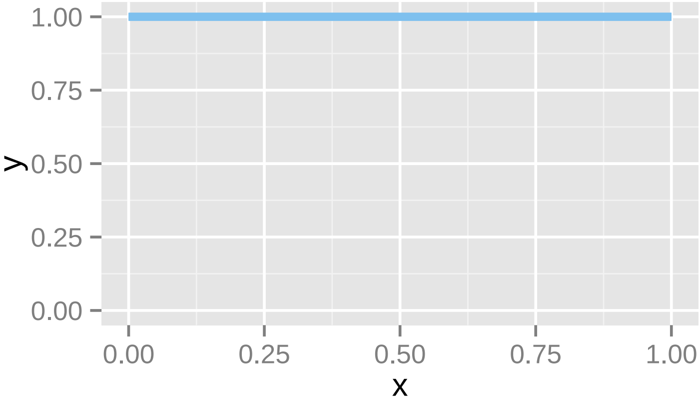
where the minimum is taken over all functions.

We want risk minimizer, is empirical risk minimizer close enough?

In practice, we only have a finite sample.

Empirical Risk Minimization

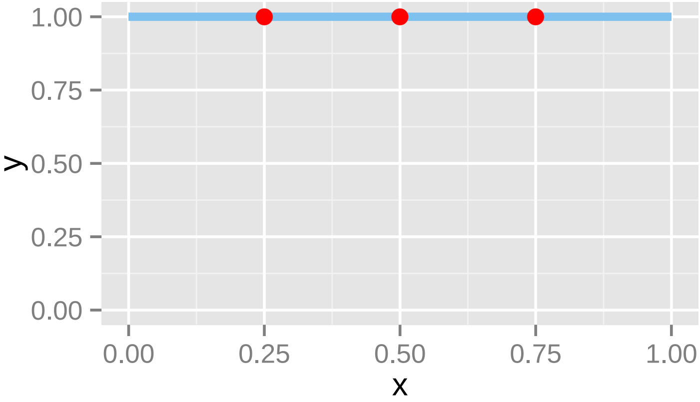
$P_{\mathcal{X}} = \text{Uniform}[0,1]$, $Y \equiv 1$ (i.e. Y is always 1).



$\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$.

Empirical Risk Minimization

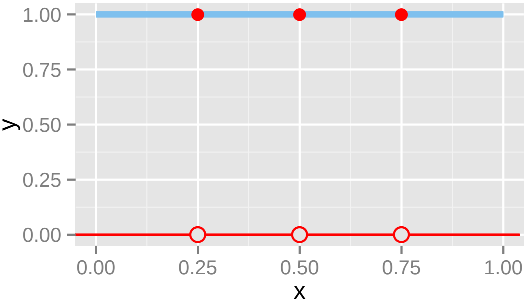
$P_{\mathcal{X}} = \text{Uniform}[0,1]$, $Y \equiv 1$ (i.e. Y is always 1).



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).

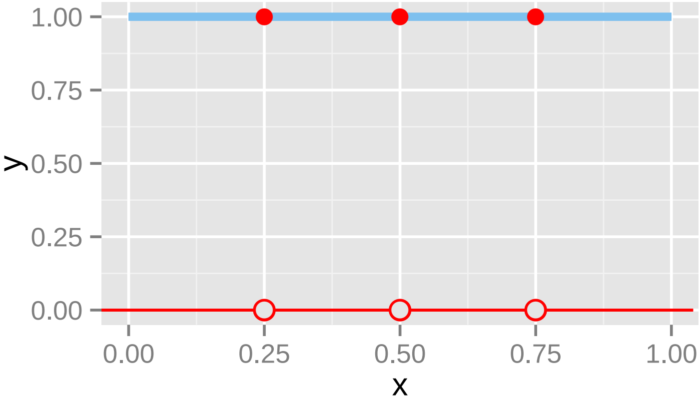


A proposed prediction function:

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

Empirical Risk Minimization

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



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

Empirical Risk Minimization

- ERM led to a function f that just memorized the data.
- How to spread information or **generalize** from training inputs to new inputs?
- 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: "Constrained ERM"
  - Instead of minimizing empirical risk over *all* prediction functions,
  - constrain to a particular subset, called a **hypothesis space**.

Hypothesis Spaces

Definition

A **hypothesis space** $\mathcal{F}$ is a set of functions mapping $\mathcal{X} \to \mathcal{A}$. It is the collection of prediction functions we are choosing from.

Want Hypothesis Space that

- Includes only those functions that have desired "regularity", e.g. smoothness, simplicity
- Easy to work with

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

Constrained Empirical Risk Minimization

- Hypothesis space $\mathcal{F}$, a set of prediction functions mapping $\mathcal{X} \to \mathcal{A}$
- **Empirical risk minimizer** (ERM) in $\mathcal{F}$ is

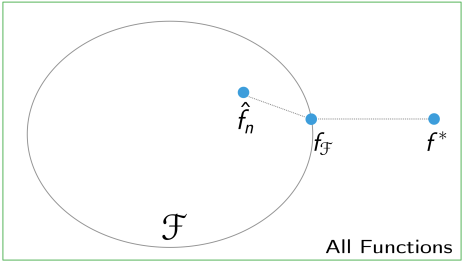
$$\hat{f}_n \in \underset{f \in \mathcal{F}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

- **Risk minimizer** in $\mathcal{F}$ is $f^*_{\mathcal{F}} \in \mathcal{F}$, where

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

Excess Risk Decomposition

Error 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:

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

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

- We've been cheating a bit by writing "argmin".
- In practice, we need a method to find $\hat{f}_n \in \mathcal{F}$.
- For nice choices of loss functions and classes $\mathcal{F}$, we can get arbitrarily close to a minimizer
  - But takes time – is it 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

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

- Can optimization error be negative? Yes!
- But

$$\hat{R}(\tilde{f}_n) - \hat{R}(\hat{f}_n) \geqslant 0.$$

Error Decomposition in Practice

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

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

- Concept check: It would be nice to have a concrete example where we find an $\tilde{f}_n$ and look at it's error decomposition. 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 $\ell : \mathcal{A} \times \mathcal{Y} \to R$.
- Choose hypothesis space $\mathcal{F}$.
- Use an optimization method to find ERM $\hat{f}_n \in \mathcal{F}$:

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

- Data scientist's job:
  - choose $\mathcal{F}$ to balance between approximation and estimation error.
  - as we get more training data, use a bigger $\mathcal{F}$