CPSC 532D — 1. SETUP; ERM

Danica J. Sutherland
University of British Columbia, Vancouver
Fall 2023

For syllabus-type material, see the course website.

This course is about: When should we expect machine learning algorithms to work? What kind of problems are possible for machine learning models to represent? What is possible to learn from data?

To phrase these questions more precisely, we'll start by formalizing a bit a default problem setup for the course.

1 PROBLEM SETUP

Our default learning problem is as follows:

- We have a data distribution $\mathcal{D}$ over some domain $\mathcal{Z}$. For *supervised learning*, this is often (but not always) actually a product space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ of (x, y) pairs, where x is an input object (e.g. an image) and y is a label (e.g. whether the image contains a dog).
- We have m independent, identically distributed samples $z_1, \dots, z_m \sim \mathcal{D}$.
- The sequence $S = (z_1, \dots, z_m) \sim \mathcal{D}^m$ is our training "set."
  - The "set" terminology is extremely well-established, so we'll use it. But we want to allow repeats, and occasionally (e.g. in online learning) we might want to care about the order, so we'll mathematically treat it as an m-tuple and not actually a set.
- We have a *hypothesis class* $\mathcal{H}$. In supervised learning, this is usually a set of predictors $h : \mathcal{X} \to \hat{\mathcal{Y}}$, a space of prediction functions.
  - Often, we might have $\hat{\mathcal{Y}} = \mathcal{Y}$, but we also might not; for example, it's common to have a problem with binary labels so $\mathcal{Y} = \{0, 1\}$, but make probabilistic predictions in $\hat{\mathcal{Y}} = [0, 1]$, or general confidence predictions in $\mathbb{R}$.
  - An example $\mathcal{H}$ might be a set of linear predictors, e.g. $\{x \mapsto w \cdot x : w \in \mathbb{R}^d\}$, or $\{x \mapsto w \cdot x : \|w\| \le B\}$.

$x \mapsto 2x + 3$ means "the function which, given the argument x, returns $2x + 3$"; $\mathcal{H}$ is a set of functions. This is like `lambda x: 2*x+3` in Python.

- We have a *loss function* $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$. In supervised learning, this often takes the form $\ell(h, (x, y)) = l(h(x), y)$ for some $l : \hat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$. Some common examples:
  - Zero-one loss: $l(\hat{y}, y) = \mathbb{1}(\hat{y} \neq y)$, usually used for $\mathcal{Y} = \hat{\mathcal{Y}}$ a discrete set of labels. This corresponds to one minus the *accuracy* of a predictor.
  - Logistic loss: $l(\hat{y}, y) = \log(1 + \exp(-\hat{y}y))$ for $\hat{\mathcal{Y}} = \mathbb{R}$, $\mathcal{Y} = \{-1, 1\}$. This loss $\to 0$ if $\hat{y} \to \infty y$ (very confidently right), is $\log 2$ if $\hat{y} = 0$ (a totally ambiguous prediction), and $\to \infty$ if $\hat{y} \to -\infty y$ (very confidently wrong).

The function $\mathbb{1}$ returns one if its boolean argument is true, and zero if not.

For more, visit https://cs.ubc.ca/~dsuth/532D/23w1/.

  - Square loss: $l(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$. (Sometimes the $\frac{1}{2}$ isn't included.)

- $L_{\mathcal{D}}(h) = \mathbb{E}_{z\sim\mathcal{D}}\, \ell(h, z) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\, l(h(x), y)$ is called the *risk*, the *population loss*, the *true loss*, etc.
- $L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i) = \frac{1}{m}\sum_{i=1}^{m} l(h(x_i), y_i)$ is the *empirical risk*, the *sample loss*, the *training loss* (if S is the training set), etc.
- A learning algorithm $\mathcal{A}$ is a function that takes in a sample S and returns a hypothesis in $\mathcal{H}$. Ideally, one with low risk.

Translating notation across relevant sources, for reference:

	These notes	[SSBD]	[MRT]	[Bach23]	[Zhang23]
Number of samples	m	m	m	n	n
Sample set	S	S	S	$\mathcal{D}_n$	S_n
Distribution over $\mathcal{X}\times\mathcal{Y}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$
Hypothesis class	$\mathcal{H}$	$\mathcal{H}$	$\mathcal{H}$	–	–
Parameter set[1]	–	–	–	Θ	Ω
Loss $\mathcal{H}\times\mathcal{Z}\to\mathbb{R}$	ℓ	ℓ	–	–	$\phi(w, z)$
Loss $\hat{\mathcal{Y}}\times\mathcal{Y}\to\mathbb{R}$	l	–	L	ℓ[2]	L
Empirical risk	$L_S(h)$	$L_S(h)$	$\hat{R}_S(h)$	$\hat{\mathcal{R}}(\theta)$	$\phi(w, \mathcal{D})$
Population risk	$L_{\mathcal{D}}(h)$	$L_{\mathcal{D}}(h)$	$R(h)$	$\mathcal{R}(\theta)$	$\phi(w, S_n)$

2 Emprical risk minimization

The most common learning algorithm we'll think about is *empirical risk minimization*: $\mathrm{ERM}(S) \in \arg\min_{h\in\mathcal{H}} L_S(h)$. (If there are ties, by default we think of the algorithm returning an arbitrary choice.)

If $\mathcal{H}$ is infinite, there might be not be a minimizer; we usually won't worry about this explicitly, but basically everything we talk about could be generalized to approximate *minimizers.*

The returned hypothesis, ERM(S), which we will also often denote $\hat{h}_S$, is called an empirical risk minimizer ("an ERM").

For example, ordinary least squares is ERM with the squared loss and $\mathcal{H} = \{x \mapsto w \cdot x\}$:

$$\begin{aligned}
\mathrm{ERM}(S) &\in \operatorname*{arg\,min}_{h\in\{x\mapsto w\cdot x\,:\, w\in\mathbb{R}^d\}} L_S(h) \\
&= \operatorname*{arg\,min}_{h\in\{x\mapsto w\cdot x\,:\, w\in\mathbb{R}^d\}} \frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i) \\
&= \operatorname*{arg\,min}_{h\in\{x\mapsto w\cdot x\,:\, w\in\mathbb{R}^d\}} \frac{1}{m}\sum_{i=1}^{m} l(h(x_i), y_i) \\
&= \left\{ x \mapsto x\cdot\hat{w} \,:\, \hat{w} \in \operatorname*{arg\,min}_{w\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} l(w\cdot x_i, y_i) \right\} \\
&= \left\{ x \mapsto x\cdot\hat{w} \,:\, \hat{w} \in \operatorname*{arg\,min}_{w\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} \frac{1}{2}(w\cdot x_i - y_i)^2 \right\},
\end{aligned}$$

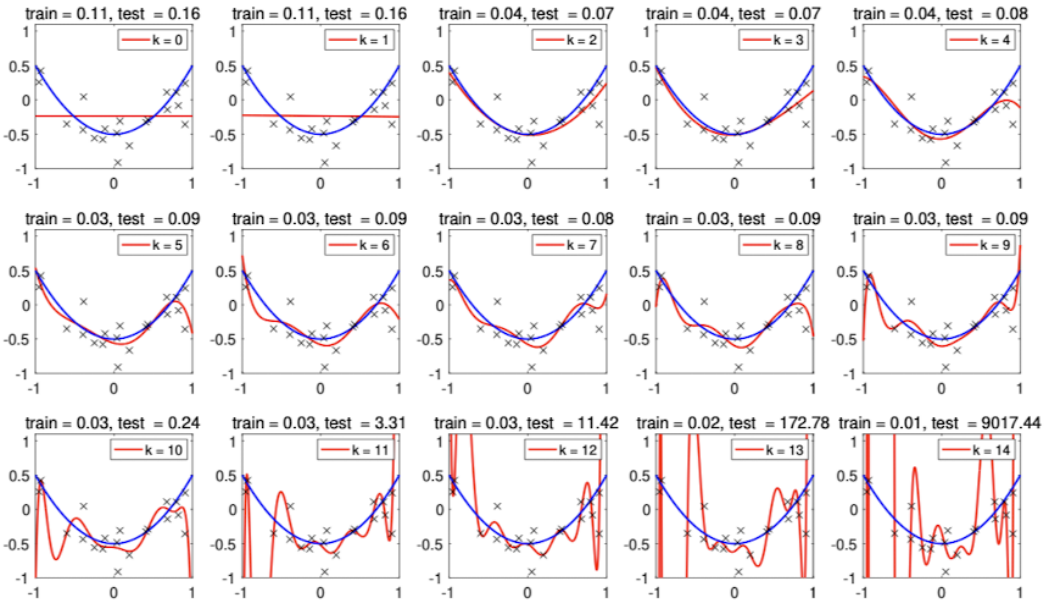
In our notation here, $\mathrm{ERM}(S)$ *is returning a function (which makes these last couple of lines slightly tedious); we could equally well have let $\mathcal{H}$ be a set of parameter vectors and define a loss on parameters, $\ell(w, (x, y)) = \frac{1}{2}(x\cdot w - y)^2$.*

and now $\hat{w}$ in the last line is probably what your intro stats class wrote down in the first place as the definition of linear regression.

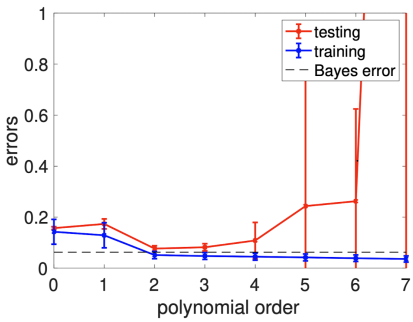
[1] For these sources, the prediction function $\mathcal{X}\to\hat{\mathcal{Y}}$ is obtained by $f_\theta(x)$ or $f(w, x)$.
[2] Arguments are in the other order: $\ell(y, \hat{y})$.

We know that $L_S(\text{ERM}(S))$ is small by definition, but when can we expect $L_{\mathcal{D}}(\text{ERM}(S))$ to be small? The first big chunk of this course is about this question in particular. The vital question is about choosing an appropriate hypothesis class $\mathcal{H}$: if it's too simple, you'll never be able to learn the pattern you're looking for, but if it's too complicated, you'll *overfit* and pick one that seems good by chance, i.e. has good $L_S(\text{ERM}(S))$ but bad $L_{\mathcal{D}}(\text{ERM}(S))$.

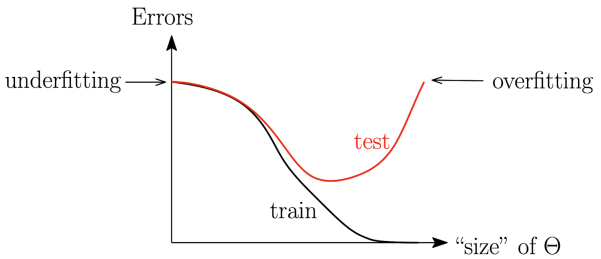
Figure 1 illustrates this trade-off for polynomial regression. This is similar to what you saw in your intro machine learning class, but one of the things we'll do in this course is formalize this general intuition and prove theorems about it.



(a) Polynomial regression, $h(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_k x^k$, for increasing k. Blue is $\mathbb{E}[y \mid x]$ (a quadratic function); red is estimates. [Bach23, Figure 2.1]



(b) Training and test errors from Figure 1a. [Bach23, Figure 2.2]



(c) Cartoon version of Figure 1b for general $\mathcal{H}$ (he writes Θ). [Bach23, Chapter 2]

Figure 1: Underfitting to overfitting as $\mathcal{H}$ gets bigger.

2.1 *ERM bounds*

The basic way to prove when ERM generalizes well is based on this decomposition:

h^* can be anything *in $\mathcal{H}$ that we want to compare to*

$$\begin{aligned} L_{\mathcal{D}}(\hat{h}_S) &= L_{\mathcal{D}}(\hat{h}_S) \underbrace{- L_S(\hat{h}_S) + L_S(\hat{h}_S)}_{0} \underbrace{- L_S(h^*) + L_S(h^*)}_{0} \underbrace{- L_{\mathcal{D}}(h^*) + L_{\mathcal{D}}(h^*)}_{0} \\ &= \underbrace{L_{\mathcal{D}}(\hat{h}_S) - L_S(\hat{h}_S)}_{\text{A: } \hat{h}_S\text{'s overfitting}} + \underbrace{L_S(\hat{h}_S) - L_S(h^*)}_{\leq 0\text{: } \hat{h}_S \text{ minimizes } L_S} + \underbrace{L_S(h^*) - L_{\mathcal{D}}(h^*)}_{\text{B: } h^*\text{'s ``underfitting''}} + L_{\mathcal{D}}(h^*) \\ &\leq L_{\mathcal{D}}(h^*) + A + B. \end{aligned}$$

So, if we can bound A and B, then we can say that $\hat{h}_S$ isn't too much worse than h^*. But since h^* was arbitrary (as long as our bound of B didn't depend on it), then this means that

$$L_{\mathcal{D}}(\hat{h}_S) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq A + B.$$

If you aren't familiar with inf, *it's like* min *but makes sense even if there isn't a minimizer. For example,* $\inf_{x \in \mathbb{R}: x > 0} x = 0$ *even though* 0 *isn't in that set.*

This term on the left is called the *excess error*: it's how much worse $\hat{h}_S$ is than the best thing in $\mathcal{H}$. The next few weeks will be denoted to different ways to bound A + B.

REFERENCES

[Bach23] Francis Bach. *Learning Theory from First Principles*. April 2023 draft.

[MRT] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talkwalkar. *Foundations of Machine Learning*. 2nd edition. MIT Press, 2018.

[SSBD] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[Zhang23] Tong Zhang. *Mathematical Analysis of Machine Learning Algorithms*. 2023 pre-publication version.