

Optimization for Data Science

ETH Zürich, FS 2023

261-5110-00L

Lecture 1: Introduction & Optimization

Bernd Gärtner
Niao He

<https://www.ti.inf.ethz.ch/ew/courses/ODS23/index.html>

February 26, 2023

Course Organization

- ▶ Lectures (Mon 13.15-14.00, NO C 60 and Tue 10.15-11.55, ETF C 1); recordings and live streaming will be available through the ETH video portal.
- ▶ Live questions from you are always welcome!
- ▶ We will frequently run clicker questions in the EduApp to allow you to assess your understanding during the lectures.
- ▶ Exercise classes (Tue 14.15-16, Fri 14.15-16), starting February 21; no recording or live streaming
- ▶ Weekly non-graded exercises; submission not mandatory but highly recommended, to support the learning experience, and as preparation for the written assignments and the exam

Course Organization II

- ▶ **Four** graded written assignments (10% of the grade each)
- ▶ Written final exam, 180 minutes, **closed** book (60% of the grade, 4 pages of prepared notes allowed)
- ▶ Material relevant for the exam: **slides, blackboard notes, exercises, written assignments**
- ▶ Lecture notes are available (also contain additional material and more in-depth discussions)
- ▶ All static information (course overview, formalities, dates, rooms, lecturers and teaching assistants, . . .) can be found on the course web page.
- ▶ All announcements and teaching materials as well as **discussion forums** can be found in the course moodle.

Overview (from course catalog)

This course provides an in-depth theoretical treatment of classical and modern optimization methods that are relevant in data science. The focus is on

- ▶ motivations and design principles behind the algorithms;
- ▶ provable performance bounds;
- ▶ mathematical tools and techniques to prove them.

You will understand

- ▶ why optimization algorithms work;
- ▶ what their limits are.

This understanding will be of help in selecting suitable algorithms in a given application, but providing concrete practical guidance is not our focus.

This lecture: The role of optimization for Data Science

Classical algorithm theory:

- ▶ An optimization algorithm (for example Kruskal's) solves a well-defined **optimization problem** (for example minimum spanning tree)

Data Science:

- ▶ Starting point is a **learning problem**, not always well-defined.
- ▶ **Optimization typically happens on training data**, but even a perfect result may fail to solve the learning problem.
- ▶ This is not a failure of the optimization algorithm but of the model in which it was applied.
- ▶ **Optimization is only one ingredient in solving a learning problem.**

Studying optimization will not turn you into a holistic data scientist... but lets you understand and explore the (after all not so small) optimization corner in the Data Science landscape.

Greater Data Science

In his article 50 years of Data Science, David Donoho outlines [six divisions](#) [Don17]:

1. Data Gathering, Preparation, and Exploration
2. Data Representation and Transformation
3. Computing with Data ← doing optimization
4. Data Modeling ← understanding optimization
5. Data Visualization and Presentation
6. Science about Data Science

Optimization is done within Computing with Data.

Optimization needs to be understood in Data Modeling (we need to come up with models that we know how to optimize).

Algorithms: Classical view

- ▶ A computational procedure that transforms input into output.
- ▶ Example [CLRS09, Section 23.1]:
Assume that we have a connected, undirected graph $G = (V, E)$ with a weight function $w : E \rightarrow \mathbb{R}$ and wish to find a minimum spanning tree for G .
- ▶ Kruskal's algorithm: worst-case runtime $O(|E| \log |E|)$
- ▶ Prim's algorithm with Fibonacci heaps: worst-case runtime $O(|E| + |V| \log |V|)$
- ▶ So Prim's algorithm is faster on dense graphs.
- ▶ Case closed.

Algorithms: Data Science view

- ▶ The starting point are **data** which we want to explain in some (not always prespecified) **way**.
- ▶ An algorithm is a tool to help us in **finding such an explanation**.
- ▶ There is not a single input, but **a typically unknown input distribution** from which a measurement or an experiment draws a (noisy) sample.
- ▶ We are not interested in explaining a concrete sample, but the nature of the distribution.
- ▶ The main quality measure of an algorithm is its **explanation ability**.
- ▶ Runtime is secondary (we need to ensure that the algorithm can be run).

Minimum spanning tree as a learning problem [Gro18, Chapter 4]

Synthetic setup:

- ▶ Ground truth weights on the edges of a complete graph, all roughly the same.
- ▶ Weights are perturbed by random noise.
- ▶ As a consequence, the MST may differ from one sample to the next.
- ▶ Goal: learn the ground truth MST.

Low noise:

- ▶ Any MST algorithm produces the ground truth MST from one sample, with high probability.

High noise:

- ▶ The MST of the sample will significantly differ from the ground truth MST, we need more samples.

Algorithmic information sets

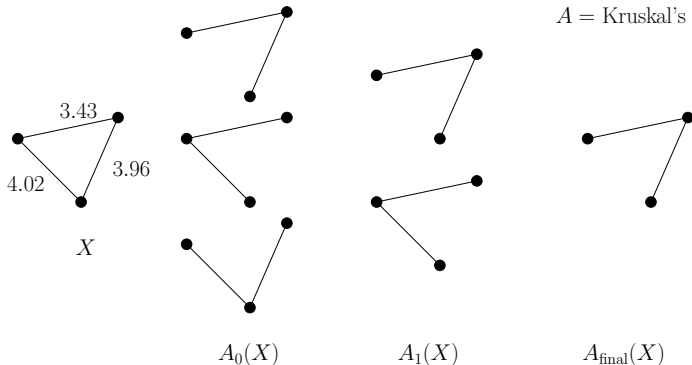
X : a sample (from the ground truth graph, perturbed by random noise)

A : Some stepwise **MST** algorithm (think of Kruskal's or Prim's)

- ▶ $A_t(X)$: the set of spanning trees still possible after step t , taking the edges processed so far into account.
- ▶ $A_0(X)$: the set of all spanning trees.

$A_{\text{final}}(X)$: the MST of X .

$A = \text{Kruskal's}$



Working with two samples

X, X'' : two samples (from the ground truth graph, perturbed by random noise)

A : Some stepwise MST algorithm

- ▶ Run A in parallel on X', X'' .
- ▶ Compute t maximizing the joint information: the probability that two spanning trees, chosen uniformly at random from $A_t(X')$ and $A_t(X'')$, respectively, are equal.
- ▶ Return a spanning tree randomly chosen from $A_t(X') \cap A_t(X'')$.

Results of this early stopping algorithm: The algorithm matters!

- ▶ With $A = \text{Kruskal's algorithm}$, we get closer to the ground truth than with $A = \text{Prim's}$.
- ▶ A third algorithm, Reverse Delete, is even better than Kruskal's. In the classical view, this algorithm is not competitive due to its bad worst-case runtime.

A learning framework

Ingredients:

- ▶ A data source \mathcal{X} (probability space, equipped with a probability distribution that we in general do not know)
- ▶ Independent samples $X_1, X_2, \dots \sim \mathcal{X}$, as many as we want (or can afford)
- ▶ Goal: “explain” \mathcal{X} through these samples!

Simple example: Biased coin flip.

- ▶ $\mathcal{X} = \{0, 1\}$ (1 = head, 0 = tail)
- ▶ Head comes up with (unknown) probability p^* , and the goal is to learn p^* .
- ▶ $p^* = \lim_{n \rightarrow \infty} |\{i : X_i = 1\}|/n$ (law of large numbers), but this limit is not computable exactly. 大数定律下的值大概为0.5
- ▶ The Chernoff bound says how large n needs to be for the empirical estimate $|\{i : X_i = 1\}|/n$ to yield a good approximation of p^* with high probability.

Expected risk minimization

Ingredients (cont'd):

- ▶ A class \mathcal{H} of hypotheses (possible explanations of \mathcal{X})
- ▶ Goal: select the hypothesis $H \in \mathcal{H}$ that best explains \mathcal{X} !
- ▶ Quality measure: risk or loss function $\ell : \mathcal{H} \times \mathcal{X} \rightarrow \mathbb{R}$
- ▶ $\ell(H, X)$: by how much does hypothesis H fail to explain data X ?

Expected risk of hypothesis H :

$$\ell(H) := \mathbb{E}_{\mathcal{X}}[\ell(H, X)]$$

Best explanation is the expected risk minimizer (assuming that it exists):

$$H^* = \operatorname{argmin}_{H \in \mathcal{H}} \ell(H).$$

Expected risk minimization: Biased coin

- ▶ $\mathcal{X} = \{0, 1\}$ (1 = head, 0 = tail)
- ▶ $\mathcal{H} = [0, 1]$ (the candidates for the bias p^*)
- ▶ $\ell(H, X) = (X - H)^2$
- ▶ Exercise 1: the expected risk minimizer H^* exists and equals p^* .
- ▶ **Mathematically**, we may be able to argue about expected risk.

PAC Learning

Computationally, we are typically at a loss.

Not knowing the distribution over the data source \mathcal{X} , we cannot compute

- ▶ the expected risk $\ell(H)$ of a hypothesis H ;
- ▶ the expected risk minimizer H^* , even if it exists.

More modest goal: try to be **probably approximately correct** (PAC)!

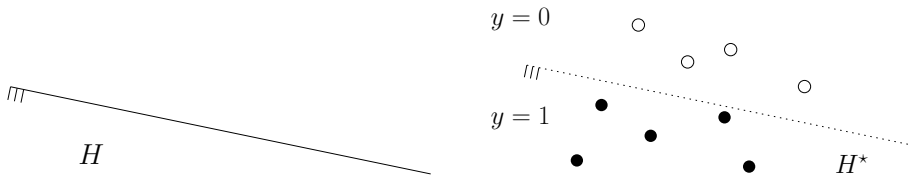
Given tolerances $\delta, \varepsilon > 0$, produce with probability at least $1 - \delta$ a hypothesis $\tilde{H} \in \mathcal{H}$ such that

$$\ell(\tilde{H}) \leq \inf_{H \in \mathcal{H}} \ell(H) + \varepsilon.$$

This means that with **high probability**, we **approximately** solve the expected risk minimization problem.

Running example: Learning a halfplane

- ▶ $\mathcal{X} = \mathbb{R}^2 \times \{0, 1\}$
- ▶ \mathcal{X} comes with an unknown **halfplane** H^* . The goal is to learn H^* .
- ▶ Halfplane: set of the form $H = \{(x_1, x_2) \in \mathbb{R}^2 : ax_1 + bx_2 \leq c\}$ for given parameters $a, b, c \in \mathbb{R}$ (“everything on one side of a line”):



- ▶ \mathcal{H} : the set of all halfplanes
- ▶ Samples are of the form $X = (\mathbf{x}, y) \in \mathcal{X}$, where $\mathbf{x} \in \mathbb{R}^2$ is a point in the plane and $y = \mathbb{I}(\{\mathbf{x} \in H^*\}) \in \{0, 1\}$ its **label**, telling us whether \mathbf{x} is in the unknown halfplane H^* or not.
- ▶ This is **supervised learning**: we obtain **labeled** training samples from the data source.

Running example: Learning a halfplane II

0-1-loss (tells us whether $X = (\mathbf{x}, y)$ is misclassified by H):

$$\ell(H, X) = \mathbb{I}(\{\mathbb{I}(\{\mathbf{x} \in H\}) \neq y\}) = \mathbb{I}(\mathbf{x} \in H \oplus H^*),$$

where \oplus denotes symmetric difference of two sets.

Expected risk (probability that H misclassifies a point \mathbf{x} randomly sampled from \mathcal{X}):

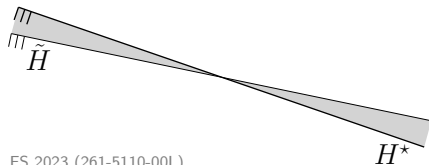
► $\ell(H) = \text{prob}(H \oplus H^*)$

Optimal halfplane H^* (no misclassified points):

► $\ell(H^*) = 0$

PAC halfplane \tilde{H} (measure of misclassified points is small with high probability):

► $\ell(\tilde{H}) \leq \varepsilon$ with probability at least $1 - \delta$



Empirical risk minimization

The most prominent “customer” of optimization in Data Science.

- ▶ Expected risk values $\ell(H)$, $H \in \mathcal{H}$ are not available to us.
- ▶ But we can draw n independent samples X_1, X_2, \dots, X_n from \mathcal{X} which form our **training data**.
- ▶ Given these data, the **empirical risk** is

$$\ell_n(H) = \frac{1}{n} \sum_{i=1}^n \ell(H, X_i).$$

- ▶ $\ell_n(H)$ is a random variable.

(Weak) law of large numbers (Lemma 1.1)

Let $H \in \mathcal{H}$ be a hypothesis. For any $\delta, \epsilon > 0$, there exists $n_0 \in \mathbb{N}$ such that for $n \geq n_0$,

$$|\ell_n(H) - \ell(H)| \leq \epsilon \tag{1}$$

with probability at least $1 - \delta$.

Empirical risk minimization II

For $n \in \mathbb{N}$ and given training data X_1, X_2, \dots, X_n from \mathcal{X} , produce a hypothesis \tilde{H}_n such that

$$\ell_n(\tilde{H}_n) \leq \inf_{H \in \mathcal{H}} \ell_n(H) + \varepsilon.$$

- ▶ This may still be a hard problem, but at least, we have all the data that we need in order to solve it.
- ▶ \tilde{H}_n is (almost) the best explanation that we can find for the given training data.
- ▶ \tilde{H}_n is a (hypothesis-valued) random variable.

Here, we are in the “classical algorithm theory view” of an optimization algorithm as solving a well-defined optimization problem:

Input: training data X_1, X_2, \dots, X_n

Output: an almost optimal hypothesis \tilde{H}_n

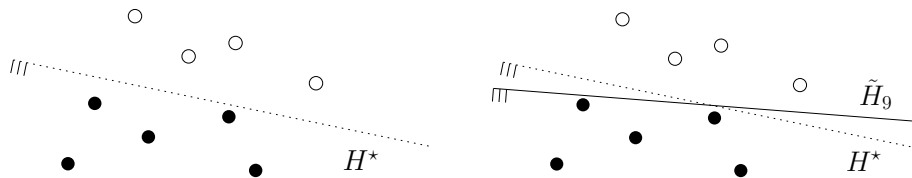
Empirical risk minimization: Biased coin

- ▶ $\mathcal{X} = \{0, 1\}$ (1 = head, 0 = tail)
- ▶ $\mathcal{H} = [0, 1]$ (the candidates for the bias p^\star)
- ▶ $\ell(H, X) = (X - H)^2$
- ▶ $\ell_n(H) = \frac{1}{n} \sum_{i=1}^n (X_i - H)^2$
- ▶ Highschool calculus: this is (exactly and unsurprisingly) minimized by

$$\tilde{H}_n = \frac{1}{n} \sum_{i=1}^n X_i,$$

the relative frequency of heads in a sequence of n biased coin flips.

Empirical risk minimization: Learning a halfplane



- ▶ $\mathcal{X} = \mathbb{R}^2 \times \{0, 1\}$ (points labeled with the information whether they are in the unknown halfplane H^* or not.)
- ▶ \mathcal{H} the set of all halfplanes
- ▶ $\ell(H, X) = \mathbb{I}(\mathbf{x} \in H \oplus H^*)$ (is \mathbf{x} misclassified by H ?)
- ▶ $\ell_n(H) = \frac{1}{n} \sum_{i=1}^n \mathbb{I}(\mathbf{x}_i \in H \oplus H^*)$
- ▶ This is minimized (with value 0) by **any** halfplane \tilde{H}_n that separates the training points with label 1 from the ones with label 0. Such a halfplane can efficiently be found through linear programming.

Empirical versus expected risk

- ▶ \tilde{H}_n : almost the best explanation for the training data
- ▶ In an ideal world: \tilde{H}_n is also almost the best explanation for the data source \mathcal{X} :

$$\ell(\tilde{H}_n) \leq \inf_{H \in \mathcal{H}} \ell(H) + \varepsilon$$

with probability at least $1 - \delta$.

Biased coin flip:

- ▶ \tilde{H}_n is the relative frequency of heads in a sequence of n biased coin flips.
- ▶ By Exercise 1 (i):

$$\ell(\tilde{H}_n) = \ell(p^\star) + \left(p^\star - \frac{1}{n} \sum_{i=1}^n X_i \right)^2.$$

- ▶ Chernoff bound: for n sufficiently large, the term in brackets becomes small, with high probability \Rightarrow ideal world scenario

Empirical versus expected risk II

The law of large numbers seems to ensure that we are **always** in an ideal world scenario:

- ▶ \tilde{H}_n : an almost best explanation of the training data (computed through empirical risk minimization)
- ▶ \tilde{H} : an almost best explanation of the data source (just needed for the proof)

$$\begin{aligned}\ell(\tilde{H}_n) &\leq \ell_n(\tilde{H}_n) + \varepsilon && \text{(weak law of large numbers for } \tilde{H}_n, \text{ Lemma 1.1)} \\ &\leq \inf_{H \in \mathcal{H}} \ell_n(H) + 2\varepsilon && (\tilde{H}_n \text{ is an almost best explanation of the training data)} \\ &\leq \ell_n(\tilde{H}) + 2\varepsilon && (\dots \text{ and } \tilde{H} \text{ can't be better than the best explanation)} \\ &\leq \ell(\tilde{H}) + 3\varepsilon && \text{(weak law of large numbers for } \tilde{H}, \text{ Lemma 1.1)} \\ &\leq \inf_{H \in \mathcal{H}} \ell(H) + 4\varepsilon && (\tilde{H} \text{ is an almost best explanation of the data source)}\end{aligned}$$

with probability at least $1 - 2\delta$ (union bound: each of the two weak laws fails with probability at most δ) \Rightarrow ideal world scenario

This was complete nonsense!

Empirical versus expected risk III

Lemma 1.1 (the weak law of large numbers) can only be applied to a **fixed** hypothesis but not to the **data-dependent** hypothesis \tilde{H}_n . \Rightarrow the above proof fails.

A **true** sufficient condition for an ideal world scenario is that the weak law of large numbers **uniformly** holds for **all** hypotheses (in particular for \tilde{H}_n), with high probability:

Theorem 1.2

Suppose that for any $\delta, \epsilon > 0$, there exists $n_0 \in \mathbb{N}$ such that for $n \geq n_0$,

$$\sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| \leq \epsilon$$

with probability at least $1 - \delta$. Then, for $n \geq n_0$, an approximate empirical risk minimizer \tilde{H}_n is PAC for expected risk minimization, meaning that it satisfies

$$\ell(\tilde{H}_n) \leq \inf_{H \in \mathcal{H}} \ell(H) + 3\epsilon$$

with probability at least $1 - \delta$.

Empirical versus expected risk IV

Proof.

$$\begin{aligned}\ell(\tilde{H}_n) &\leq \ell_n(\tilde{H}_n) + \varepsilon && \text{(follows from } \sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| \leq \varepsilon) \\ &\leq \inf_{H \in \mathcal{H}} \ell_n(H) + 2\varepsilon && (\tilde{H}_n \text{ is an almost best explanation of the training data)} \\ &\leq \inf_{H \in \mathcal{H}} \ell(H) + 3\varepsilon && \text{(follows from } \sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| \leq \varepsilon)\end{aligned}$$

with probability at least $1 - \delta$. □

The condition $\sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| \leq \varepsilon$:

- ▶ Optimistic guess: It's always true!
- ▶ Pessimistic guess: It's certainly too strong to hold in relevant cases!

The condition turns out to hold in **many** relevant cases (we'll get to one of them), but it's far from being always true.

Clicker Question

Which of the following are non-constant random variables (meaning that they actually depend on the samples that we draw from \mathcal{X})?

- (a) Training Data: X_1, \dots, X_n
- (b) Expected risk of a fixed hypothesis: $\ell(H)$
- (c) Empirical risk of a fixed hypothesis: $\ell_n(H)$
- (d) Set of approximately optimal hypotheses w.r.t. expected risk
- (e) Set of approximately optimal hypotheses w.r.t. empirical risk

- ▶ a, b, c, d, e
- ▶ a, b, d
- ▶ a, c, e
- ▶ b, c, d, e

A counterexample

The inequality

$$\ell(\tilde{H}_n) \leq \ell_n(\tilde{H}_n) + \varepsilon$$

in our nonsense derivation can in fact fail. As a consequence of Theorem 1.2, the condition $\sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)| \leq \varepsilon$ can also fail.

Example:

- ▶ $\mathcal{X} = [0, 1]$, equipped with the uniform distribution.
- ▶ \mathcal{H} the collection of all subsets of $[0, 1]$ of length (Lebesgue measure) $1/2$
- ▶ $\ell(H, X) = \mathbb{I}(\{X \notin H\}) \Rightarrow$ expected risk $\ell(H) = 1/2$ for all $H \in \mathcal{H}$

Observation

For all training data $X_1, X_2, \dots, X_n \in \mathcal{X}$, there exists a hypothesis $\tilde{H}_n \in \mathcal{H}$ with $\tilde{H}_n \supseteq \{X_1, X_2, \dots, X_n\}$, so that empirical risk $\ell_n(\tilde{H}_n) = 0$.

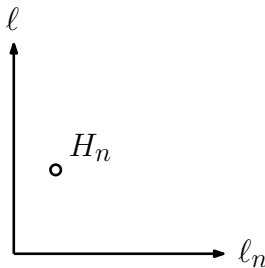
Hence, $\ell(\tilde{H}_n) \leq \ell_n(\tilde{H}_n) + \varepsilon$ cannot be achieved for $\varepsilon < 1/2$.

Exercise 2: \mathcal{H} can be made countable (a finite \mathcal{H} cannot lead to a counterexample).

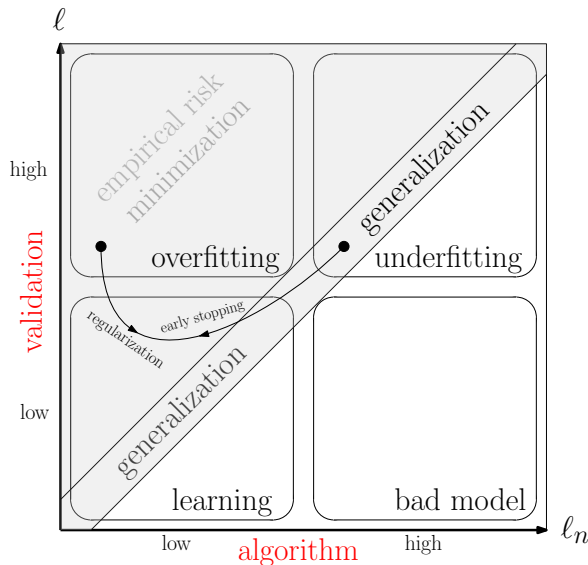
The map of learning

Learning algorithm:

- ▶ Based on training data X_1, X_2, \dots, X_n , select a data-dependent hypothesis H_n (this is a random variable).
- ▶ H_n has empirical risk $\ell_n(H_n)$ and expected risk $\ell(H_n)$.



The map of learning: Algorithm and Validation



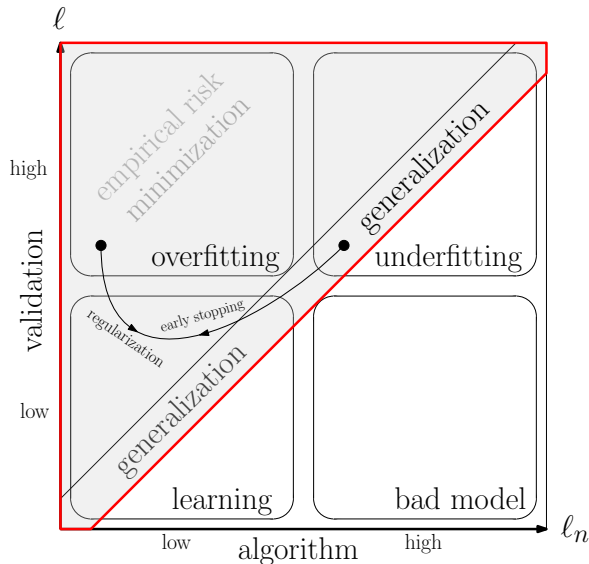
Algorithm:

- ▶ maps training data (input) to hypothesis H_n (output)
- ▶ this controls empirical risk and locates H_n in the ℓ_n -dimension

Validation:

- ▶ computes the empirical risk of H_n on fresh **test data** that the algorithm has not seen
- ▶ this approximates expected risk (weak law of large numbers!) and locates H_n in the ℓ -dimension.

The map of learning: Empirical risk minimization



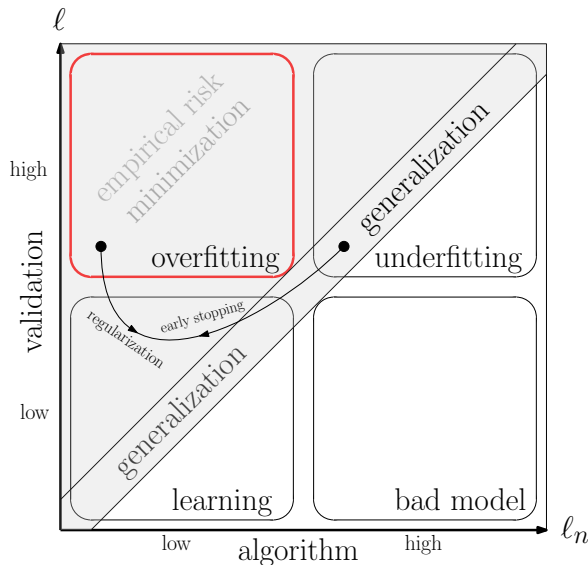
Empirical risk minimization:

- ▶ The area where an almost best explanation \tilde{H}_n of the training data ends up, w.h.p.
- ▶ Proof (implicitly seen before): Let \tilde{H} be an almost best explanation of the data source. Then

$$\begin{aligned}\ell_n(\tilde{H}_n) &\leq \inf_{H \in \mathcal{H}} \ell_n(H) + \varepsilon \\ &\leq \ell_n(\tilde{H}) + \varepsilon \\ &\leq \ell(\tilde{H}) + 2\varepsilon \quad (\text{w.h.p.}) \\ &\leq \ell(\tilde{H}_n) + 3\varepsilon\end{aligned}$$

So $\ell(\tilde{H}_n) \geq \ell_n(\tilde{H}_n) - 3\varepsilon$, w.h.p.

The map of learning: Overfitting

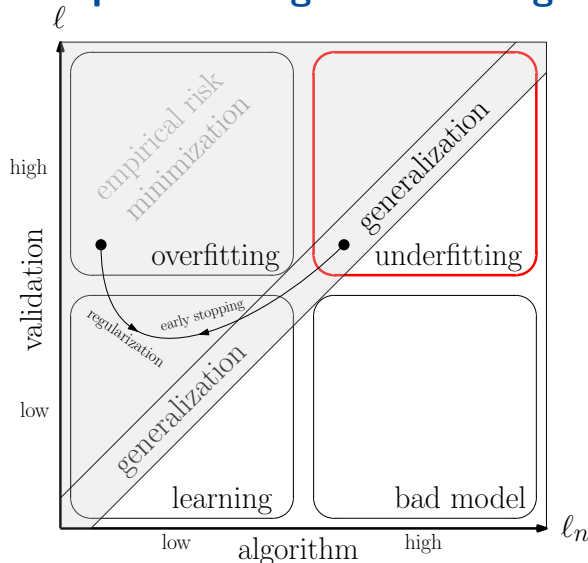


Overfitting:

- ▶ Low empirical risk, high expected risk
- ▶ Explanation quality on the data source is much worse than on the training data.
- ▶ Main reason: theory (\mathcal{H} and ℓ) is so complex that it can explain almost anything.

“I can explain the results of my favorite sports team: player X wore black socks in exactly the won games.”

The map of learning: Underfitting and Bad model



Underfitting:

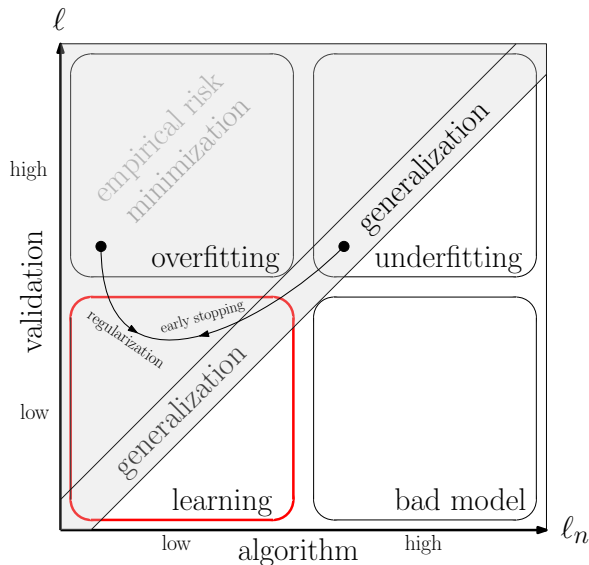
- ▶ High empirical risk, high expected risk
- ▶ We neither explain the training data nor the data source.
- ▶ Main reason: theory (\mathcal{H} and ℓ) is too simple to capture the nature of the data.

“I can explain the results of my favorite sports team: it always wins.”

Bad model:

- ▶ We can explain the data source but not the training data; we trained for the wrong goal.

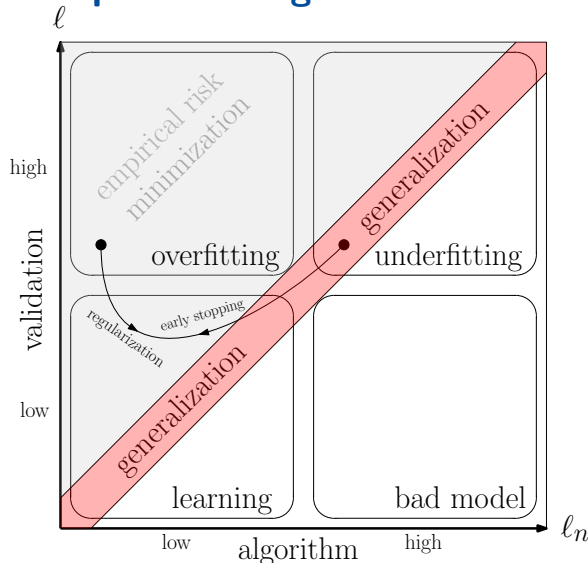
The map of learning: Learning



Learning:

- ▶ Low empirical risk, low expected risk
- ▶ Training was successful: explanation quality on the data source is good.
- ▶ Although possibly not as good as on the training data.

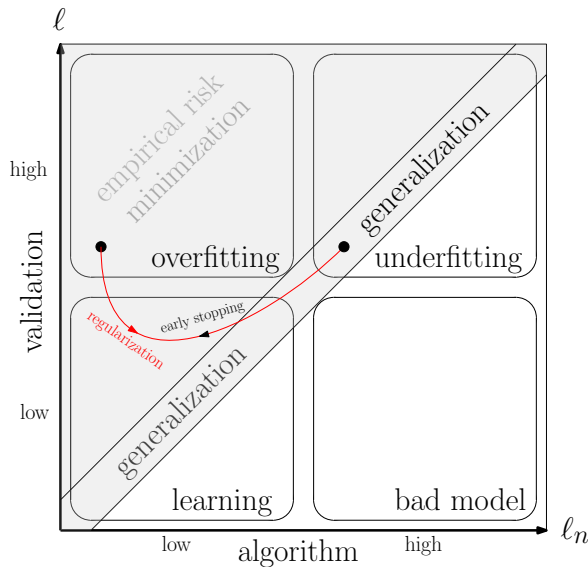
The map of learning: Generalization



Generalization:

- ▶ Expected risk is close to the empirical risk
- ▶ With respect to our explanation, training data are representative for the data source.
- ▶ This **doesn't** mean that our explanation is good.
- ▶ Any “blind explanation” (not depending on the training data) will generalize very well (weak law of large numbers).
- ▶ Ideally, we want generalization **and** learning.

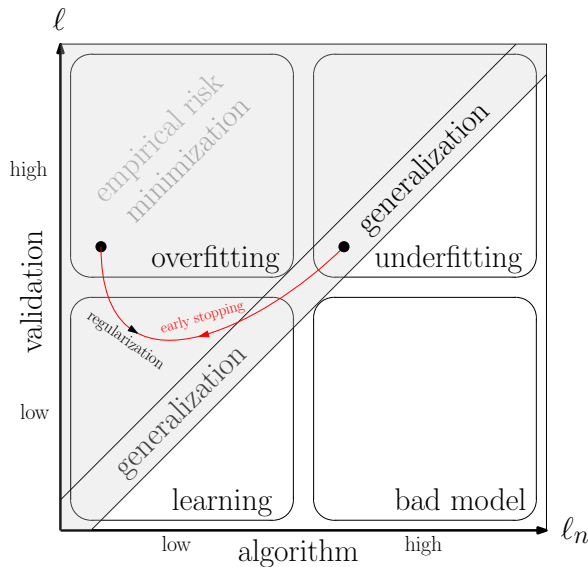
The map of learning: Regularization



Regularization (to cure overfitting):

- ▶ Make part of the complex theory (the loss function ℓ) less complex
- ▶ $\ell' = \ell + \lambda r$, for a function r that “punishes” complex hypotheses
- ▶ As λ grows, we introduce more and more **bias** (deviation from the theory) \rightarrow empirical risk increases.
- ▶ At the same time, the **variance** (sensitivity to the training data) decreases \rightarrow expected risk may decrease.
- ▶ Bias-variance tradeoff: find λ minimizing the expected risk!

The map of learning: Early stopping



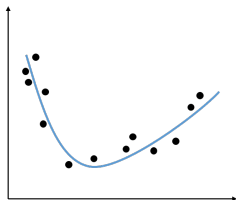
Early stopping (to cure overfitting):

- ▶ Assume we compute \tilde{H}_n by decreasing the empirical risk in every step.
- ▶ Stop at some step t and output the current hypothesis!
- ▶ As t grows, the **bias** (deviation from the theory) decreases \rightarrow empirical risk decreases.
- ▶ At the same time, the **variance** (sensitivity to the training data) increases. \rightarrow expected risk may increase.
- ▶ Bias-variance tradeoff: find t minimizing the expected risk!

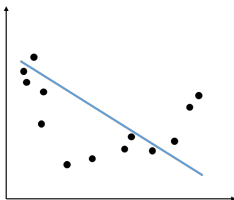
Clicker Question

Put the plots below into the right order such that they represent *generalization*, *overfitting*, and *underfitting*, respectively!

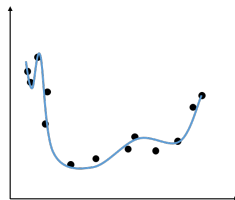
- ▶ a, b, c
- ▶ a, c, b
- ▶ b, c, a
- ▶ c, a, b



(a)



(b)



(c)

Takeaway message at this point

Empirical risk minimization is widely and successfully used in practice, but it should not be used blindly.

- ▶ In any given application, there are potentially many sensible theories (hypothesis classes \mathcal{H} and loss functions ℓ).
- ▶ Conflicting goals:
 - ▶ Theory should be **informative** (“makes sense” for the given problem)
 - ▶ Theory should be **robust** (empirical risk is a good “proxy” for expected risk).
 - ▶ Empirical risk minimization should be **efficient** in this theory.
- ▶ **Not** in this course: choosing the theory.
- ▶ **In** this course: solving the optimization problems arising in empirical risk minimization **after** the theory has been chosen.
- ▶ Other courses (at ETH) deal with choosing the theory; see Section 1.12 (Further listening) in the lecture notes.

The VC theory (more details in lecture notes)

Provides sufficient conditions under which we have **generalization**, meaning that $\ell(\tilde{H}_n) \approx \ell_n(\tilde{H}_n)$.

- ▶ $\mathcal{X} = \mathcal{D} \times \{0, 1\}$.
 - ▶ unknown (and arbitrary) probability distribution over data \mathcal{D}
 - ▶ unknown “optimal” subset $H^* \subseteq \mathcal{D}$
 - ▶ samples of the form $X = (\mathbf{x}, y)$ where $\mathbf{x} \in \mathcal{D}$ and **label** y tells us whether $\mathbf{x} \in H^*$.
 - ▶ \mathcal{H} a collection of subsets of \mathcal{D} (candidates for H^*).
 - ▶ 0-1-loss
- ▶ Example that we have seen: $\mathcal{D} = \mathbb{R}^2$, \mathcal{H} the set of all halfplanes.
- ▶ If \mathcal{H} has finite **VC-dimension**, then $\sup_{H \in \mathcal{H}} |\ell_n(H) - \ell(H)|$ can be made arbitrarily small, with arbitrarily high probability, by choosing the number of samples n large enough. Generalization of the empirical risk minimizer follows.
- ▶ For halfplanes (VC-dimension 3), this works.
- ▶ See Section 1.7 in the lecture notes, as well as **263-5300-00L Guarantees for Machine Learning**, and **401-2684-00L Mathematics of Machine Learning**

Distribution-dependent guarantees (more details in lecture notes)

We assume something about the probability distribution underlying our data source.

- ▶ Example: **Linear Regression with fixed design and centered noise**.
 - ▶ Ground truth vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathbb{R}^d, n \geq d$
 - ▶ Data source $\mathcal{Y} = \mathbb{R}^n$
 - ▶ unknown “optimal” vector $\beta^* \in \mathbb{R}^d$
 - ▶ sample $\mathbf{y} \sim \mathcal{Y}$ has entries $y_i = \mathbf{x}_i^\top \beta^* + w_i$, where the w_i are independent noise terms with expectation 0 and variance σ^2 each.
 - ▶ $\mathcal{H} = \mathbb{R}^d$ (candidates for β^*)
 - ▶ Squared loss $\ell(\beta, \mathbf{y}) = \frac{1}{n}(y_i - \mathbf{x}_i^\top \beta)^2$
- ▶ Then the empirical risk minimizer $\tilde{\beta} = \tilde{\beta}_1$ (from **one** sample with n values) has almost optimal expected risk:

$$\ell(\tilde{\beta}) = \ell(\beta^*) + O\left(\sigma^2 \cdot \frac{d + \log(1/\delta)}{n}\right),$$

with probability at least $1 - \delta$.

- ▶ See Section 1.8 in the lecture notes, as well as **263-4508-00L Algorithmic Foundations of Data Science**

Worst-case versus average-case complexity

In empirical risk minimization, training data are independent samples from the data source \mathcal{X} .

What we (should) care about is not the **worst-case** complexity of our optimization algorithm, but its **average-case** complexity (taken over the joint distribution of the training samples).

Example: Quicksort for sorting n numbers

- ▶ Worst-case complexity (maximum runtime over all inputs) is $\Theta(n^2)$.
- ▶ Average-case complexity (expected runtime over randomly shuffled input) is $O(n \log n)$.

Still, even in Data Science, many sources (including this course) analyze optimization algorithms in terms of their worst-case complexity.

Why?

Worst-case versus average-case complexity II

In most cases, it's very difficult to understand the average-case complexity.

Problem 1: Lack of Knowledge.

- ▶ We may have no information about the probability distribution over \mathcal{X} .
- ▶ As in Quicksort, we can use that the training samples come in random order, but this doesn't go beyond **expected** worst-case analysis.
- ▶ A real average-case analysis requires knowledge about the distribution over \mathcal{X} .

Problem 2: Too specific knowledge.

- ▶ We may be able to compute the average-case complexity of the algorithm **over the specific distribution** in a given application.
- ▶ This analysis will typically not work for the distributions in other applications.
- ▶ There are infinitely many different average-case complexities.
- ▶ **There is only one worst-case complexity** that covers all applications (possibly with overly pessimistic bounds, though).

The story of the simplex method (more details in lecture notes)

The complexity of this optimization method for linear programming has puzzled (and is still puzzling) researchers since its invention in the 1940s.

A great showcase for the gap between worst-case and average-case complexity and its eventual bridging via [smoothed analysis](#).

See Section 1.10 in the lecture notes.

The estimation-optimization tradeoff

Recall:

- ▶ In empirical risk minimization, we don't want the **absolutely** best explanation of the training data.
- ▶ An almost best one \tilde{H}_n suffices.

Rationale:

- ▶ empirical risk $\ell_n(\tilde{H}_n)$ is only an approximation of expected risk $\ell(\tilde{H}_n)$ (and we care about the latter).
- ▶ We anyway lose precision in going from $\ell_n(\tilde{H}_n)$ to $\ell(\tilde{H}_n)$.
- ▶ It doesn't help to optimize $\ell_n(\tilde{H}_n)$ to a significantly higher precision.

Estimation error:

- ▶ precision that we lose in going from empirical to expected risk

Optimization error:

- ▶ precision that we lose in finding only an almost best explanation of the training data

The estimation-optimization tradeoff II

We may have constraints on how many training data we can sample.

- ▶ Reason: Training data are expensive to obtain.
- ▶ Physical experiment or measurement with **significant cost**.
- ▶ **Human intervention** (manual labeling of training data)
- ▶ **Small-scale learning**

We may have constraints on how much time we can afford to spend on optimization.

- ▶ Reason: Large data
- ▶ “Good old days” where every algorithm of polynomial runtime was considered efficient are gone.
- ▶ We typically need **linear-time** or even **sublinear-time** algorithms in order to cope with the data.
- ▶ **Large-scale learning**

The estimation-optimization tradeoff III

Small-scale learning:

- ▶ it doesn't hurt to go for as small an optimization error as we can.

Large-scale learning:

- ▶ we may need to give up on some optimization precision in order to be able to stay within the optimization time budget.

Estimation-optimization tradeoff:

- ▶ find the most efficient way of spending the resources under the given constraints!

Algorithms: runtime guarantees support the tradeoff

We employ stepwise methods, gradually improving a candidate solution. Guarantees:





- ▶ “on data of this and that type, this and that algorithm is guaranteed to have optimization error at most ε after at most $c(\varepsilon)$ many steps.”
- ▶ c is a function that grows as $\varepsilon \rightarrow 0$
- ▶ c is typically not even defined at $\varepsilon = 0$.
- ▶ Algorithms cannot be used to “optimize to the end.”
- ▶ Analysis: bound the growth of c as $\varepsilon \rightarrow 0$!
- ▶ Algorithms can significantly differ in this growth.
- ▶ $c(\varepsilon) = O(\log(1/\varepsilon))$ is better than $c(\varepsilon) = O(1/\varepsilon)$ is better than $c(\varepsilon) = O(1/\varepsilon^2)$.

Bounds we give are usually **worst-case**.

- ▶ may be overly pessimistic in a concrete application
- ▶ but still provide some (and sometimes the only) useful guidance concerning optimization time.

Further reading on the tradeoff: Bottou and Bousquet [BB07].

Bibliography

-  Léon Bottou and Olivier Bousquet.
The tradeoffs of large scale learning.
Advances in neural information processing systems, 20, 2007.
-  Thomas H. Cormen, Charles E. Leiserson, Ron L. Rivest, and Clifford Stein.
Introduction to algorithms.
MIT Press, Cambridge, Mass, 3rd ed. edition, 2009.
-  David Donoho.
Fifty years of data science.
Journal of Computational and Graphical Statistics, 24(6):745–766, 2017.
-  Alexey Gronskiy.
Statistical Mechanics and Information Theory in Approximate Robust Inference.
PhD thesis, ETH Zurich, Zurich, 2018.