Advanced Topics in ML part I

Amir Yehudayoff, DIKU

Contents

Chapter 1

Introduction

Remark. *This text probably contains errors.*

High-level goals

This is a second or a third course you see on ML. This course is theoretical. We will discuss many ideas. These ideas can improve your understanding of ML and allow you to make better choices. It can also allow you to communicate more accurately and effectively with others.

Remark. *I'll try to not assume too much, and if you don't understand something please ask.*

Mathematics

Math is a language for thought. Mathematics consists of three main ingredients: definitions, theorems and proofs. Each serves a unique purpose in the thought process. Definition – who? Theorem – what? Proofs – why? Many proofs come with algorithmic ideas.

Topics

Our focus will be on several related topics in ML:

1. Compression.
2. Stability.
3. Privacy.
4. Replicability.

Reading

1. This text.
2. Understanding Machine Learning: From Theory to Algorithms / Shalev-Shwartz and Ben-David.
3. Papers.

Exercises

We have three weeks together, and there are three chapters in the text. In each chapter, you can find exercises that are meant to help you digest the material. You should submit the exercise (according to the guidelines). I recommend to try to solve them with friends, and even if you fail to find a full solutions, please try to write your thoughts and submit the outcome.

Chapter 2

Basic definitions

PAC

We start by recalling basic notions and set some notation.

Remark. *The domain is denoted by X and the label space by Y. The domain can be the collection of pictures, and the labeled space can be $\{yes, no\} \equiv \{\pm\} \equiv \{0, 1\}$. We think of X, Y as finite sets (this allows to ignore some subtle probability theory issues).*

Definition. *The input to a learning algorithm (a.k.a. sample or dataset) is a list of the form*

$$(x_1, y_1), \ldots, (x_n, y_n) \in X \times Y.$$

We sometimes denote it by S. The output of a learning algorithm is predictions on all the domain:

$$h : X \to Y.$$

A learning algorithm (with sample size n) is a function

$$A : (X \times Y)^n \to Y^X.$$

So $A(S)$ is a function from X to Y.

Remark. *This is an abstract approach. In practice, we also care about the computational complexity of A, and other issues.*

What is a "learning problem"? A standard way to formally define this notion is via a collection of functions $H \subset Y^X$, sometimes referred to as an hypothesis class. We think of H as known to A. There are a few ways to think of H: (1) the function we

are trying to learn is in H (a.k.a. the realizable setting), (2) the function we are going to output is in H (e.g. neural network with a certain architecture), and (3) we think of H as the functions we want to compete with (a.k.a. the agnostic setting).

Example. *The domain is $X = \mathbb{R}^2$ and H is the collection of axis parallel rectangles. Draw a picture of a typical input data (in the realizable case), and a corresponding reasonable output.*

Definition. *A learning algorithm A is proper (w.r.t. H) if $A(S) \in H$ for all S.*

How do we asses the quality of our predictions? There is a loss function for that. We shall not define the most general framework, but just mention two examples. The zero-one loss function

$$L(h, (x, y)) = 1_{h(x) \neq y}.$$

If $Y \subseteq \mathbb{R}$ then we can have

$$L(h, (x, y)) = (h(x) - y)^2.$$

Remark. *We are going to focus on zero-one loss. When $|Y| = 2$, this is called binary classification. When $|Y| > 2$, this is called multiclass classification.*

How is the input data generated? The standard assumption (pros and cons) is that there is some unknown distribution μ on $X \times Y$ and $S \sim \mu^n$.

Definition (losses of h)**.** *The population loss (w.r.t. μ) is*

$$L_\mu(h) = \mathbb{E}_{(x,y)\sim\mu} L(h, (x, y)).$$

The empirical loss on data S is

$$L_S(h) = \frac{1}{n} \sum_{i \in [n]} L(h, (x_i, y_i)).$$

Definition (losses of A)**.** *The population loss (true loss) of an algorithm A with sample size n w.r.t. μ is*

$$L_\mu(A) = \mathbb{E}_{S\sim\mu^n}[L_\mu(A(S))].$$

The population loss of a concept class H w.r.t. μ is

$$L_\mu(H) = \inf\{L_\mu(h) : h \in H\}.$$

Remark. *The algorithm does not know (a priori) any of the population losses. It can, however, compute the empirical loss. This leads to a compelling algorithmic principle: Empirical Risk Minimization. Output a function with smallest empirical loss. The ERM principle is not always optimal.*

Definition (realizable)**.** *The distribution μ is called realizable if $L_\mu(H) = 0$.*

Definition (realizable PAC)**.** *A class H is PAC learnable in the realizable setting if for every $\epsilon > 0$ there is an algorithm A with sample size n so that for all realizable μ,*

$$L_\mu(A) < \epsilon.$$

Remark. *There are other (similar) formulations of this definition.*

Remark. *This is a "worst case" definition; the algorithm should work uniformly well for all μ. One can think of other reasonable definitions.*

Exercise 1. *This is about the case when we know the distribution μ, when we can use a Bayesian perspective. The Bayes optimal classifier $B : X \to Y$ is defined as follows: the value $B(x)$ is a maximizer y of $\Pr_{(x',y')\sim\mu}[y' = y|x' = x]$. Prove that for any $f : X \to Y$,*

$$L_\mu(B) \leq L_\mu(f).$$

Definition (agnostic PAC)**.** *A class H is PAC learnable in the agnostic setting if for every $\epsilon > 0$ there is an algorithm A with sample size n so that for all μ,*

$$L_\mu(A) < \epsilon + L_\mu(H).$$

Remark. *We try to use different languages to answer the question:*

When is H learnable?

Uniform convergence

For every fixed h, standard concentration inequalities say that if $n \gg \frac{1}{\epsilon^2}$ then for every μ,

$$\mathbb{E}_{S\sim\mu^n}[|L_\mu(h) - L_S(h)|] < \epsilon.$$

If we take enough samples, then the true loss is typically close to the empirical loss. In this case, we can use the empirical loss as an effective proxy to the true loss. But we do not care about a single h. We want to argue on all of H at once.

Definition (uniform convergence)**.** *The class H satisfies uniform convergence if for every $\epsilon > 0$, there is n so that for all μ,*

$$\mathbb{E}_{S\sim\mu^n}[\sup_{h\in H}|L_\mu(h) - L_S(h)|] < \epsilon.$$

In words, the empirical loss is typically close to the true loss for all of H simultaneously.

Remark. *Again, there are other formulations.*

Lemma 1. *If H satisfies uniform convergence then any ERM learns H.*

Proof. Fix μ and let h_* be a function in H with minimum loss:

$$L_\mu(h_*) = L_\mu(H).$$

Partition the loss to three parts:

$$\begin{aligned}&\mathbb{E}_{S\sim\mu^n}[L_\mu(A(S))]\\ &= \mathbb{E}_{S\sim\mu^n}[L_\mu(A(S)) - L_S(A(S))] + \mathbb{E}_{S\sim\mu^n}[L_S(A(S)) - L_\mu(h_*)] + L_\mu(H).\end{aligned}$$

The first term is "how much can we rely on the empirical loss", the second is small for all ERMs, and the third is the "unavoidable loss". Because A is an ERM,

$$\begin{aligned}&\leq \mathbb{E}_{S\sim\mu^n}[\sup_{h\in H}|L_\mu(h) - L_S(h)|] + \mathbb{E}_{S\sim\mu^n}[L_S(h_*) - L_\mu(h_*)] + L_\mu(H)\\ &\leq \epsilon + \epsilon + L_\mu(H).\end{aligned}$$

□

Remark. *The term $L_\mu(A(S)) - L_S(A(S))$ is related to a very important notion: overfitting. Overfitting means that the algorithm produced very good results on the empirical data, but these results are too good to be true—on the true data the results are poor. Whenever we devise a learning algorithm, we should ask ourselfs "did we overfit?". More on this later on.*

A metric perspective

We can think of a learning problem as a metric space. The points in this space are the functions $h \in H$. Given a distribution ν on X, we can measure the "distance" between points via

$$\text{dist}(h, h') = \Pr_{x\sim\nu}[h(x) \neq h'(x)].$$

Remark. *Draw as points in the plane as a mental model.*

Remark. *We can now connect "ML properties" to "metric properties".*

Definition (a ball)**.** *For $r > 0$, the ball of radius r around h is*

$$B(r, h) = \{h' : \mathrm{dist}(h', h) < r\}.$$

Definition (cover numbers)**.** *For $\epsilon > 0$, the cover number $N_\nu(\epsilon) = N_{H,\nu}(\epsilon)$ of this metric space is the minimum number of balls of radius ϵ that cover the space.*

Remark. *Cover numbers play a very important role in many areas of math. In the context of ML, the main statement is something like:*

$$\textit{the sample complexity of PAC learning } H \textit{ is roughly } \sup_\nu \log N_\nu(\epsilon).$$

This is similar to various notions of dimensions.

Approximations

For a fixed $h \in H$, and distribution ν on X we can estimate $\mathbb{E}_\mu[h] = \Pr_{x\sim\nu}[h(x) = 1]$ by taking some samples from ν and computing the average.

$$\textit{Can we estimate } \mathbb{E}_\mu[h] \textit{ for all } h \in H \textit{ together?}$$

We shall relate this question to learning later on. For now, let us keep this idea via the following definition.

Definition (approximation)**.** *The class H has approximation parameter k if for every distribution ν there are $x_1, \ldots, x_k \in X$ so that for all $h \in H$,*

$$\left| \Pr_{x\sim\nu}[h(x) = 1] - \Pr_{i\sim[k]}[h(x_i) = 1] \right| < \tfrac{1}{10},$$

where $i \sim [k]$ is uniform in $[k]$.

Remark. *In words, the empirical mean of every $h \in H$ is close to its true mean.*

Remark. *The constant $\frac{1}{10}$ can be changed to be a parameter ϵ.*

Remark. *We mentioned several notions that are related to PAC learnability. There are more notions of this form, but we stop for now.*

VC dimension

The notions we introduce above (PAC, uniform convergence, metric, approximations) used both H and the space of distributions on $X \times Y$.

Is there a clean property of H alone that allows
to decide our ability to solve learning the problem defined by H?

Vapnik and Chervonenkis identified a basic mechanism doing this.

Definition (shattering)**.** *A set $S \subset X$ is shattered by H if*

$$H|_S = \{0,1\}^S$$

where

$$H|_S = \{h|_S : h \in H\}$$

and $h|_S$ is the restriction of h to S.

Example. *If H is the collection of half-planes in $\mathbb{R}^2$ and S is three generic points in $\mathbb{R}^2$ then S is shattered by H.*

Definition (VC dimension)**.** *The VC dimension of H is the maximum size of a shattered set.*

Example. *If H is the collection of half-planes in $\mathbb{R}^2$ then its VC dimension is three. There are three shattered points. No four points are shattered (there are two cases).*

Exercise 2. *What is the VC dimension of the class of half-spaces in $\mathbb{R}^n$?*

Remark. *The VC dimension captures many important properties of H.*

Theorem 2. *The following are equivalent:*

1. *The class H is PAC learnable.*
2. *The VC dimension of H is finite.*
3. *The class H satisfies uniform convergence.*

Remark. *This theorem also says that a "universal" learning rule for binary classification is an ERM. One problem is that ERMs are often not efficiently computable.*

Theorem 3 (Haussler)**.** *There is a constant $C > 1$ so that if the VC dimension of H is d then for every distribution ν on X,*

$$N_{H,\nu}(\epsilon) \leq (\tfrac{C}{\epsilon})^d.$$

Remark. *Using the metric space perspective, the theorem says that the sample complexity of learning H is at most $\approx d$.*

Exercise 3. *Here we prove a weak version of Haussler's theorem. The Sauer-Shelah-Perles bound says that if H has VC dimension d and $|X| = n$ then*

$$|H| \leq \sum_{i=0}^{d} \binom{n}{i} \leq \left(\frac{en}{d}\right)^d.$$

Prove that for every d and $\epsilon > 0$, there is $\delta = \delta(d, \epsilon) > 0$ so that for every H with VC dimension d, and for every distribution ν on X,

$$N_{H,\nu}(\epsilon) \leq \left(\frac{1}{\delta}\right)^d.$$

Theorem 4. *Every H with $VC(H) = d$ has approximation parameter $O(d)$.*

Remark. *To prove the theorem, we can choose the points $x_1, \ldots, x_k$ in the approximation for ν by (i.i.d.) sampling from ν. The analysis is not trivial.*

Exercise 4. *Prove that if H has $VC(H) = d$ and approximation parameter k then $k \geq \Omega(d)$, where $\Omega(\cdot)$ is some universal constant.*

Chapter 3

Sample compression schemes

Basic definitions

Intuitively, our ability to learn is related to our ability to understand. What does "understand" mean? One possibility is that "understanding" is about providing succinct explanations to the observe phenomenon. It is about our ability to compress information. Let us start with a motivation example.

Example. *Draw a few linearly separated points in the plane (labelled by $\pm$). Some points are more important than other in this drawing. Which? Why?*

Example. *Draw points in the plane labelled by an axis parallel rectangle. Some points are more important than other in this drawing. Which? Why?*

Littlestone and Warmuth identified a common property to many learning algorithm and abstract it as follows.

Definition. *Let $H \subset Y^X$. A sample compression scheme of size s is comprised of two map: a compression map κ and a reconstruction map ρ. The compression map κ maps a sample S to a subsample of S of size at most s;*

$$\kappa((x_1, y_1), \ldots, (x_n, y_n)) = ((x_{i_1}, y_{i_1}), \ldots, (x_{i_k}, y_{i_k}))$$

for $k \leq s$. The reconstruction map takes a sample S of size at most s and output a full function from X to Y. The two maps should satisfy the following property: For every $S = ((x_i, y_i))_{i=1}^n$ that is realizable by H, for every $i \in [n]$ we have

$$[\rho(\kappa(S))](x_i) = y_i.$$

In words, if we compress and then reconstruct then we get the correct answers.

Remark. *The first observation is that sample compression scheme lead to learning algorithms. Given input S, the algorithm is $A = \rho \circ \kappa$; that is, for all S,*

$$A(S) = \rho(\kappa(S)).$$

In words, the algorithm first compresses its data, then it "forgets" the original data and decompress it. It has the additional property that for all S

$$L_S(A(S)) = 0.$$

This holds even thought it is "not an ERM".

Example. *Half-planes in the plane have a sample compression scheme of size three. What are κ and ρ?*

Remark. *The perceptron algorithm and SVM are sample compression schemes.*

Remark. *Encoder-decoder neural nets can be thought of as implementing this high-level idea.*

Exercise 1. *Show that the class of half-spaces is $\mathbb{R}^d$ has a sample compression scheme of size $2d$.*

Compression and learning

Sample compression schemes yield good learning algorithms.

Theorem 5. *If (κ, ρ) is a sample compression scheme for H of size s and μ is realizable, then for every $\epsilon > 0$ with $n \gg \frac{s}{\epsilon}$ we have*

$$L_\mu(A) < \epsilon$$

where $A = \rho \circ \kappa$.

Proof. For every $S \in (X \times Y)^n$ and $T \subset [n]$ of size at most s, denote by $S|_T$ the sub-sample of S that corresponds to T, and define the function

$$f_T = f_{T,S} = \rho(S|_T).$$

For each T, define the event

$$E_T = \{S \in (X \times Y)^n : L_S(f_T) = 0, L_\mu(f_T) \geq \epsilon\}.$$

In words, E_T comprises of sample that if we compress using T we get something with low empirical loss but large true loss. Denote by F the event that the prediction has large loss:

$$F = \{S : L_\mu(A(S)) \geq \epsilon\}.$$

Because we have a sample compression scheme,

$$F \subset \bigcup_T E_T.$$

Indeed, if the prediction of the algorithm has loss $\geq \epsilon$ on sample S then with T so that $(s_i : i \in T) = \kappa(S)$ we have $f_T = A(S)$ so that

$$L_S(f_T) = L_S(A(S)) = 0 \qquad \text{and} \qquad L_\mu(f_T) = L_\mu(A(S)) \geq \epsilon.$$

A different way of seeing that is: for all S,

$$A(S) \in \{f_T : T\}$$

and

$$L_S(A(S)) = 0.$$

Now, using the union bound,

$$\Pr[F] \leq \Pr\left[\bigcup_T E_T\right] \leq \sum_T \Pr[E_T].$$

For each fixed T, the function f_T depends on $S|_T$ which is $\leq s$ samples out of the total n samples. The two datasets $S|_T$ and $S|_{T^c}$ are independent (where $T^c = [n] \setminus T$).

Remark. *It could help imagining first that $T = [s]$.*

Conditioned on the value of $S|_T$, the event E_T says that

$$f_T \text{ is correct on all of } T^c \subset [n]$$

and on the other hand

$$\text{the true loss of } f_T \text{ is } \geq \epsilon.$$

There are $\geq n - s$ points in T^c, so that change that this happens is

$$\Pr[E_T] \leq (1 - \epsilon)^{n-s}.$$

The number of T's is at most

$$\sum_{i=0}^{s} \binom{n}{i} \leq \left(\frac{en}{s}\right)^s. \tag{3.1}$$

Remark. *The fact that there are only a few T's is where the compressing nature of the algorithm is used.*

Altogether,

$$\Pr[F] \leq (1-\epsilon)^{n-s}\left(\frac{en}{s}\right)^s < \epsilon, \tag{3.2}$$

where the last inequality holds if n is large enough. □

Remark. *The proof shows that if an algorithm does not use "all of its data" but still has low empirical loss then it generalizes.*

Exercise 2. *Prove* (3.1).

Exercise 3. *Find a concrete bound on n that makes the right inequality in 3.2 correct.*

Exercise 4. *Write the proof above without looking at the text.*

Learning and compression

We just saw that every compression scheme allows to learn. *Is it true that every problem that we can solve via ML can be explained by compression?* In a work with Moran, we proved that (in some sense) the answer is yes.

Remark. *Recall that the sample complexity of PAC learning H is approximately $VC(H)$.*

Theorem 6. *If $VC(H) = d$ then there is a sample compression scheme for H of size $s \leq 2^{O(d)}$.*

Remark. *The sample compression scheme above uses a small amount of "additional information"; more on this below.*

Remark. *It is not known if sample compression schemes of size $O(d)$ exist or not.*

We will state a slightly more informative theorem.

Definition. *The dual class of $H \subset Y^X$, denoted by H^*, is the class $H^* \subset Y^H$ defined as follows. The names of the functions in H^* is X, and $x \in H^*$ defines the function $x(h) = h(x)$. Thinking of H as an $H \times X$ matrix with entries in Y, the class H^* is obtained by transposing the matrix. The dual VC dimension of H is $VC^*(H) = VC(H^*)$.*

Exercise 5. *Prove that $VC^*(H) < 2^{VC(H)+1}$ for all H.*

Theorem 7. *If $VC(H) = d$ and $VC^*(H) = d^*$ then there is a sample compression scheme for H of size $s \leq O(d \cdot d^*)$.*

We are going to also use the well-known von Neumann's minimax theorem from game theory.

Theorem 8. *Let M be a real matrix with I rows and J columns. Assume that for every distribution α on I, there is $j \in J$ so that*

$$\sum_{i \in I} \alpha_i M_{i,j} \leq v.$$

Then, there is a distribution β on J so that for every α,

$$\sum_{j \in J} \alpha_i M_{i,j} \beta_j \leq v.$$

Remark. *The theorem is basic in game theory. We shall not discuss this connection in detail (although it is very interesting). The rows I are pure strategies for the row player. The columns J are pure strategies for the column player. The value $M_{i,j}$ is the loss of the column player. The column player wishes to minimize her loss. Assume that for every mixed strategy α, there is a response j so that the expected loss $\sum_i \alpha_i M_{i,j}$ is small. The response can depend on knowledge of the strategy! Then, there is a mixed strategy β, so that for every α there is small loss.*

The conclusion is that certain two-player games (e.g., that can be represented by matrices) have an equilibrium point. Namely, mixed strategies for the two players that are "optimal" or "stable". Each of the players does not have any incentive to change his/hers strategy (even after playing the game many times). The minimax theorem has found applications in many areas of mathematics (and was also generalized).

Proof sketch of Theorem 7. Let $S = ((x_1, h(x_1)), \ldots, (x_n, h(x_n)))$ with $h \in H$. Let $H' \subset Y^n$ be the restriction of H to $x_1, \ldots, x_n$. The VC dimension of H' is at most d.

Using learnability. It follows that the sample complexity of PAC learning H' is at most $t \approx d$. More specifically, let A be the algorithm that takes $T \subset [n]$ of size t, and outputs an ERM, denoted by A_T, in H' for the dataset $((x_i, h(x_i))_{i \in T}$. For every distribution ν on $[n]$, we know

$$L_\nu(A) = \mathbb{E}_T L_\nu(A_T) \leq \frac{1}{3};$$

in L_ν we interpret ν as the distribution of $(x, h(x))$ for $x \sim \nu$. In particular, there is T so that

$$L_\nu(A_T) \leq \frac{1}{3}.$$

The game. Let us represent this by a matrix (or a game).

Remark. *There are two players. An input player that chooses x_i. An output player that replies with T; that is, the answer is A_T. The input player wins if $A_T(x_i) \neq h(x_i)$, and the output player wins otherwise. This is captured by the matrix*

$$M_{i,T} = 1_{A_T(i) \neq h(x_i)}.$$

The row of the matrix are $i \in [n]$, and the columns are the sets T.

The distribution ν is the α in the minimax theorem. We know that for all ν, there is T so that

$$1/3 \geq L_\nu(A) = \sum_{i \in [n]} \nu(i) 1_{A_T(i) \neq h(x_i)}.$$

The minimax theorem implies that there is a distribution β on sets T so that for every $i \in [n]$,

$$\sum_T \beta(T) 1_{A_T(i) \neq h(x_i)} \leq \frac{1}{3}.$$

In other words, for every row i in the matrix, if we average the columns according to β then we can recover $h(x_i)$:

$$\text{the average is } > \tfrac{1}{2} \iff h(x_i) = 1.$$

Concluding: high-level The dual VC dimension of the matrix of H' is at most $d^* = VC^*(H)$. From this, we can deduce (similar to uniform convergence) that if we choose i.i.d. samples $T_1, \ldots, T_k$ from β for $k \approx d^*$ then for all $i \in [n]$,

$$\Pr_{j \sim [k]}[A_{T_j}(i) = h(x_i)] \geq \frac{2}{3} - \frac{1}{10} > \frac{1}{2}$$

or in other words

$$\mathrm{MAJ}(A_{T_1}(i), \ldots, A_{T_k}(i)) = h(x_i).$$

Summary. The compression of S is the $k \cdot t$ points in $T_1, \ldots, T_k$. The reconstruction is obtained by the majority function above.

Remark. *The reconstruction function in the compression scheme described above uses a small amount of additional information; we need to know the partition of the s data points to the k parts $T_1, \ldots, T_k$.*

□

Remark. *The proof tells us that sub-sampling the dataset S can lead to not overfitting the data.*

Chapter 4

Information complexity and stability

Information theory

Information theory provides useful and powerful methods for thinking and arguing about compression. This theory was initiated by Shannon in his seminal work on coding theory. The most basic concept in it is entropy. Entropy has many intuitive properties, which allow to argue about relatively complicated situations. Our presentation is going to be quick, because you have already seen this at some point.

Definition. *Let A be a random variable taking values finite set. The entropy of A is*

$$H(A) = \sum_a \Pr[A = a] \log \frac{1}{\Pr[A = a]},$$

where $\log = \log_2$ *and by convention* $0 \log \frac{1}{0} = 0$.

Remark. *Think of A as some information we want to communicate. The entropy of A is the amount of bits we need to communicate in order to describe A.*

Exercise 1. *Prove that $H(A) \geq 0$.*

Exercise 2. *Let A be a random variable taking values in $[n]$, and let U be a random variable uniformly distributed in $[n]$. Prove that*

$$H(A) \leq H(U) = \log n.$$

When we have two types of information A, B, we can measure the description length of A when we already know B as follows.

Definition. *The conditional entropy of A conditioned on B is*

$$H(A|B) = H(A,B) - H(B).$$

Exercise 3. *Prove that*

$$H(A|B) = \sum_b \Pr[B=b] H(A_{B=b}),$$

where $A_{B=b}$ is a random variable distributed like A conditioned on the event $B=b$.

Exercise 4. *Prove the data processing inequality:*

$$H(A|B) \leq H(A).$$

We can also measure the amount of mutual information between two pieces of data.

Definition. *The mutual information between A, B is*

$$I(A;B) = H(A) + H(B) - H(A,B) = H(A) - H(A|B) = H(B) - H(B|A).$$

Exercise 5. *Prove that $I(A;B) = 0$ iff A, B are independent.*

Definition. *The conditional mutual information is*

$$I(A;B|C) = H(A|C) + H(B|C) - H(A,B|C).$$

Exercise 6. *Prove the chain rule for mutual information is that*

$$I(A;B,C) = I(A;B) + I(A;C|B).$$

Information cost

Mutual information allows to measure the amount of information an algorithm reveals on its input (we shall use notation that is suitable for learning, but this definition makes sense in general).

Definition. *The information cost of algorithm A is*

$$IC(A) = I(A(S);S)$$

where S is the input to A.

Remark. *This definition makes sense only when S is random.*

Remark. *When A is a learning algorithm, then $A(S)$ is a function.*

Remark. *The number $I(A(S);S)$ is intuitively related to privacy. Think of S as sensitive data that is fed into A. The smaller the cost is, the more private the algorithm is. There are other ways to measure privacy; more on this, later on the course.*

Overfitting

When a learning algorithm has low information cost, it is "compressing". In a work with Bassily, Moran, Nachum and Shafer, we showed that this type of compression means that the algorithm does not overfit its data. Here is a concrete definition for overfitting.

Definition. *An learning algorithm A does not overfit (w.r.t. input distribution μ) if*

$$\mathbb{E}_{S\sim\mu^n}[|L_S(A(S)) - L_\mu(A(S))|] \leq \tfrac{1}{10}.$$

In words, the true loss of the output is close to the empirical loss.

Remark. *We can add two parameters (ϵ, δ) to the definition.*

Theorem 9. *There is a (small) constant $c > 0$ so that the following holds. If the sample size of A is n, and if for a given input distribution μ we know that*

$$IC(A) \leq cn$$

then A does not overfit (w.r.t. μ).

Exercise 7. *Let $H \subseteq \{0,1\}^X$ be so that $VC(H) = d$. Let ν be a distribution on X that is realizable by H. For $h \in H$, denote by μ_h be the distribution on $(x, h(x))$ where x is sampled from ν. For every $n > 1$, describe an algorithm with sample size n so that for every $h \in H$, if $S \sim \mu_h^n$ then*

$$\Pr[L_S(A(S)) > \tfrac{1}{10}] \geq \tfrac{9}{10}$$

and

$$I(S, A(S)) \leq O(d \log n)$$

where $O(\cdot)$ is a universal constant.

Stability

We want our algorithms to be stable. That a small perturbation in the input data will not lead to a huge change in their behavior. There are many ways to define this (and it is related to privacy and replicability). Let us build some formalism for studying it.

Type of perturbations. The type of perturbation to the input data we allow is:

Definition. *For an input sample $S = (z_1, \ldots, z_n)$, for z, and for $i \in [n]$, denote by $S^{(z,i)}$ the sample obtained from S be replacing z_i by z.*

Measures of distance. We measure the affect of a perturbation as follows.

Remark. *We allow algorithms to be randomized, so that $A(S)$ is not a single function but a distribution on functions.*

Definition. *The total variation distance (a.k.a. statistical distance) between two distributions ν, ν' over the same domain is defined by*

$$d_{TV}(\nu, \nu') = \max\{\nu(E) - \nu'(E) : E \text{ is an event}\}.$$

Remark. *If this distance is small, it means that every event occurs in both distribution with similar probabilities.*

Exercise 8. *For two distributions ν, ν' over $[n]$, define*

$$\|\nu - \nu'\|_1 = \sum_{i \in [n]} |\nu(i) - \nu'(i)|.$$

Prove that

$$2d_{TV}(\nu, \nu') = \|\nu - \nu'\|_1.$$

Definition. *We say that A is stable (w.r.t. μ) if*

$$\mathbb{E}_{z \sim \mu, i \sim [n]}[d_{TV}(A(S), A(S^{(z,i)}))] < \tfrac{1}{10}$$

where i is uniform in $[n]$, where z, i are independent and where d_{TV} is with respect to the distribution with $S \sim \mu^n$, which is independent of z, i.

Remark. *In words, the statistical distance between the distribution of $A(S)$ and of $A^{(z,i)}$ is small. The two distributions are close.*

Exercise 9. *Prove that if A is stable w.r.t. μ then A does not overfit w.r.t. μ.*

Theorem 10. *If A has sample size n and $IC(A) < \frac{n}{100}$ w.r.t. μ, then A is stable.*

Remark. *The exercise and the theorem imply that algorithms with low information cost do not overfit.*

A crucial property of mutual information that is part of the proof of the theorem is (recall that $S = (z_1, \ldots, z_n) \sim \mu^n$):

Lemma 11.

$$\sum_{i \in [n]} I(A; z_i) \leq I(A; S).$$

Remark. *This is a somewhat surprising inequality: the sum of many things is smaller than a single number. This property is extremely powerful in many settings.*

Proof of lemma. By the chain rule,

$$I(A; S) = \sum_{i=1}^{n} I(A; z_i | z_1, \ldots, z_{i-1}).$$

For each i, using independence and that condition reduces entropy:

$$\begin{aligned} I(A; z_i | z_1, \ldots, z_{i-1}) &= H(z_i | z_1, \ldots, z_{i-1}) - H(A, z_i | z_1, \ldots, z_{i-1}) \\ &\geq H(z_i) - H(A, z_i) = I(A; z_i). \end{aligned}$$

□

To relate stability to information cost, we use Pinsker's inequality (we shall not prove it here).

Lemma 12. *Let $p_{A,B}$ be a distribution on pairs (A, B). Denote by p_A, p_B the two marginal distributions. Denote by $p_A \times p_B$ the product distribution. It holds that*

$$d_{TV}(p_A \times p_B, p_{A,B}) \leq \sqrt{I(A; B)}.$$

Remark. *The proof of Pinsker's inequality is elementary. There is a way to reduce the claim for general random variables to a boolean one. The proof then becomes an exercise in one-dimensional calculus.*

Proof of Theorem 10. Recall that $S = (z_1, \ldots, z_n)$. First, fix $i \in [n]$, and re-write

$$\begin{aligned} \mathbb{E}_{z \sim \mu}[d_{TV}(A(S), A(S^{(z,i)}))] &= d_{TV}((A(S), z), (A(S^{(z,i)}), z)) \\ &= d_{TV}((A(S), z), (A(S), z_i)). \end{aligned}$$

Pinsker's inequality implies that

$$d_{TV}((A(S), z), (A(S, z_i)) \leq \sqrt{I(A(S); z_i)}.$$

Second, average also over i and use that $\sqrt{\ }$ is concave:

$$\begin{aligned}\mathbb{E}_{z,i}[d_{TV}(A(S), A(S^{(z,i)}))] &\leq \mathbb{E}_i \sqrt{I(A(S); z_i)} \\ &\leq \sqrt{\mathbb{E}_i I(A(S); z_i)} \\ &= \sqrt{\tfrac{1}{n} \sum_i I(A; z_i)}.\end{aligned}$$

Lastly, use the lemma from before, and the assumption that the cost is small:

$$\tfrac{1}{n} \sum_i I(A; z_i) \leq \tfrac{1}{n} I(A; S) < \tfrac{1}{100}.$$

□

Other notions

There are other notions of stability, and other notions of entropy as well. Let us briefly mention a couple.

Replicability. Replicability refers to our ability to repeat a process or experiment without changing the result. This is an important notion in science, and we also want our algorithms to be replicable. Here is a definition that was suggested recently (by Impagliazzo, Lei, Pitassi and Sorrell).

Definition. *Let A be a learning algorithm. For this definition, we take extra care in the way we formalize how A uses its internal randomness. Denote by S the input sample to A, and by R the internal randomness of A. The two variables S and R are assumed independent. For a distribution μ on S, the algorithm A is ρ-replicable if*

$$\Pr_{R,S,S'}[A(S, R) = A(S', R)] \geq \rho$$

where S, S' are independent. In words, if we run the algorithm on two independent inputs but with the same internal randomness, then we expect the outcome to be the same.

Remark. *The closer ρ is to one, the more replicable A is.*

Global stability. A different notion of stability is that of collision probability global stability (introduced by Bun, Livni and Moran).

Definition. *Let A be a learning algorithm. For a distribution μ on S, the algorithm A is ρ-globally-stable if there is a function f so that*

$$\Pr[A(S,R) = f] \geq \rho.$$

In words, at least one of the outputs is attained with a decent chance.

The notion of global stability is captured by a different notion of entropy.

Definition. *The min-entropy of a distribution ν on $[n]$ is*

$$H_\infty(\nu) = \min_i \log \frac{1}{\nu(i)}.$$

Remark. *This notion of entropy plays a central role in the study of randomness extractors (these are algorithms that "improve the quality of randomness").*

Exercise 10. *Prove that for every distribution ν on $[n]$,*

$$H_\infty(\nu) \leq H_\infty(U)$$

where U is the uniform distribution on $[n]$. Compute $H_\infty(U)$.

Remark. *The algorithm A is ρ-globally-stable iff the distribution ν of its output $A(S,R)$ satisfies $H_\infty(\nu) \leq \log \frac{1}{\rho}$.*

Exercise 11. *Prove that if A is ρ-globally-stable then A is ρ^4-replicable.*

Remark. *The converse is not true. In a joint work with Chase and Moran, we proved that there are problems with replicable algorithms but no globally-stable algorithms.*

Collides. There is one more notion of stability (implicitly defined in the work with Chase and Moran).

Definition. *Let A be a learning algorithm. For a distribution μ on S, the algorithm A is ρ-colliding if*

$$\Pr_{R,R',S,S'}[A(S,R) = A(S',R')] \geq \rho.$$

In words, if we run the algorithm on two independent inputs and internal randomnesses, then we expect the outcome to be the same.

This notions is also captured by some entropy.

Definition. *The Renyi entropy (a.k.a. collision probability entropy) of a distribution ν on $[n]$ is*

$$H_2(\nu) = \log \frac{1}{\sum_{i\in[n]} \nu^2(i)}.$$

Exercise 12. *Prove that for every distribution ν on $[n]$,*

$$H_2(\nu) \leq H_2(U)$$

where U is the uniform distribution on $[n]$. Compute $H_2(U)$.

Remark. *The algorithm A is ρ-colliding iff the distribution ν of its output $A(S,R)$ satisfies $H_2(\nu) \leq \log \frac{1}{\rho}$.*

Remark. *In a nutshell, the message is:*

To define stability, decide how to measure the entropy of your algorithm. Stability is then defined by low entropy of the output.