Statistical learning theory

Carlos Cotrini
Department of Computer Science
ETH Zürich
Switzerland

February 2, 2021

Chapter 1

Maximum entropy and posterior agreement

1.1 Introduction

We present here an alternative statistical learning method called ME+PA (maximum entropy + posterior agreement). This method is robust and can be applied to a wide variety of learning problems. In addition, it has been experimentally demonstrated that it generalizes better than empirical risk minimization techniques in different domains, including brain image analysis [?, ?], access control [?, ?], minimum spanning trees [?], and predicting rankings in chess tournaments [?]. In particular, the notion of posterior agreement captures a notion similar to that of mutual information, when seeing learning algorithms as communication channels that use models to communicate information about an underlying phenomenon.

1.2 Overview of ME+PA

Before making a formal presentation, we give an intuitive overview. ME+PA consists of the following steps:

Step 1: Hypothesis class Define a hypothesis class $\mathcal{C}$. This indicates the class of candidate models you want to train. Examples of hypothesis classes are the class of linear regression models, the class of classification trees, and the class of neural networks with a specific architecture.

Step 2: Cost function The next step is to define a cost function R. This function takes as input a model $c \in \mathcal{C}$ and an observation X and produces a cost value $R(c, X)$ measuring how well the model fits the given observation. Since cost functions are usually defined by humans, we require cost functions to be non-negative.

Step 3: Maximum entropy For an arbitrary observation X, we now define a posterior probability distribution $p(\cdot \mid X)$ over $\mathcal{C}$, conditioned on X. For $c \in \mathcal{C}$, $p(c \mid X)$ should measure how much we believe that c is the right model for X. This belief is quantified by $R(c, X)$. The lower this cost is, the more we believe that c is the right model for X. We argue later in Section 1.5 that a natural posterior distribution that fulfils this requirement is the *Gibbs distribution induced by* R:

$$p(c \mid X) \propto \exp\left(-\frac{1}{T} R(c, X)\right). \tag{1.1}$$

where $T > 0$ is a hyper-parameter called the *temperature*.

Step 4: Posterior agreement We show in Section 1.4 that merely picking the "most likely" model c according to $p(c \mid X)$ amounts to overfitting and, therefore, it is not enough to generalize well. We use instead two independent observations X' and X'' and define the *posterior agreement kernel* between X' and X'':

$$\kappa(X', X'') := \sum_{c \in C} p(c \mid X) p(c \mid X''). \tag{1.2}$$

This kernel can be used to select a value for T from a set of candidate temperatures $\mathcal{T}$. It can even be used to select a cost function R among a set of candidate cost functions $\mathcal{R}$:

$$T^*, R^* = \operatorname*{arg\,max}_{T \in \mathcal{T}, R \in \mathcal{R}} \kappa(X', X'') \tag{1.3}$$

Finally, the two Gibbs distributions $p(\cdot \mid X')$ and $p(\cdot \mid X'')$ can be aggregated together to yield a final distribution over models

$$p^*(c \mid X', X'') \propto p(c \mid X') p(c \mid X''). \tag{1.4}$$

Observe that this distribution gives high probability only to those models c for which both $p(c \mid X')$ and $p(c \mid X'')$ are high. Therefore, $p^*(c \mid X', X'')$ is more robust to noise interference coming from either X' or X''.

1.3 Formalization

We assume that we are interested in computing a model for a *phenomenon* of interest. The phenomenon is represented with a probability distribution p over an *instance space* $\mathcal{X}$. We assume that p is unknown to us.

We perceive the phenomenon through *observations*, which we represent with elements in $\mathcal{X}$ drawn from p. In particular, we are interested in complex phenomena where making an observation is expensive, and in the worst case, we only have two independent observations X' and X''. Depending on the context, we sometimes view X' and X'' instead as random variables with distribution p.

We also assume given a finite[1] hypothesis class $\mathcal{C}$ comprising all candidate models.

Definition 1. *A* cost function *is a function $R : \mathcal{C} \times \mathcal{R} \to [0, \infty)$.*

Definition 2. *For a cost function R and an observation X, a* Gibbs distribution (induced by R and X) *is defined by*

$$p(c \mid X) \propto \exp\left(-\frac{1}{T} R(c, X)\right), \tag{1.5}$$

where $T > 0$ is a hyper-parameter called the temperature*.*

Definition 3. *For a cost function R and a temperature $T > 0$. The* posterior agreement kernel (induced by R and T) *is a function $\kappa : \mathcal{X} \times \mathcal{X} \to [0, 1]$ such that, for any two observations $X', X'' \in \mathcal{X}$,*

$$\kappa(X', X'') = \sum_{c \in \mathcal{C}} p(c \mid X') p(c \mid X''). \tag{1.6}$$

Here, $p(\cdot \mid X')$ and $p(\cdot \mid X'')$ are two Gibbs distributions induced by R and with temperature T.

Definition 4 (The combined Gibbs distribution)**.** *Assume given two Gibbs distributions induced by a same cost function, but two different observations X' and X'', and having the same temperature. The* combined Gibbs distribution *is defined as follows*

$$p(c \mid X', X'') \propto p(c \mid X') p(c \mid X''). \tag{1.7}$$

[1] ME+PA can also be applied to continuous hypothesis classes, but the foundations of this method have only been established for finite hypothesis classes.

Definition 5 (The posterior agreement principle)**.** *Assume given two independent observations X' and X'' of a phenomenon. Whenever making a choice ω from a set Ω of options concerning a particular learning algorithm, one must choose the option that* maximizes the posterior agreement kernel*:*

$$\omega^* = \arg\max_{\omega \in \Omega} \kappa(X', X''). \tag{1.8}$$

Statistical learning

Statistical learning proposes to train models via *empirical risk minimization (ERM).* In ERM, we define an instance space $\mathcal{X}$, a hypothesis class $\mathcal{C}$, and a cost function R.

ERM trains a model in $c \in \mathcal{C}$ that minimizes *the expected cost* $\mathbb{E}_X[R(c, X)]$, where X is a random observation of the phenomenon. However, this requires the distribution p behind the phenomenon, which we assume to be unavailable. In consequence, statistical learning advocates to approximate this expected cost with the *empirical cost* $\frac{1}{n}\sum_{i \leq n} R(c, X_i)$, where $X_1, \ldots, X_n$ is a sample of p.

We define the *empirical risk minimizer* as a model $\hat{c}$ that minimizes the empirical cost. That is,

$$\hat{c} = \arg\min_{c \in \mathcal{C}} \sum_{i \leq n} R(c, X_i). \tag{1.9}$$

We assume for simplicity that $\hat{c}$ always exist, although this is not true in general.

1.4 The random array

We present an artificial toy example, called *the random array* that illustrates the power of ME+PA. In particular, it illustrates the ways in which empirical risk minimization overconfidently generalizes when there is insufficient data and noise interference.

1.4.1 Setup

Consider a random array $\mathfrak{X} = (\mathfrak{X}_0, \ldots, \mathfrak{X}_{n-1})$ with range $\mathbb{R}^n$. Suppose that $\mathfrak{X}_i \sim \mathcal{N}(i, \sigma_i)$. For convenience, we let σ_i be such that $\mathbf{P}(\mathfrak{X}_i < 0) = 0.05$. The random array $\mathfrak{X}$ is our phenomenon.

Assume that we only have two observations X' and X'', consisting of two arrays drawn at random from $\mathfrak{X}$'s distribution. We want to estimate, from just X' and X'',

$$i^* := \arg\min_{i \leq n} \mathbb{E}_{\mathfrak{X}}[\mathfrak{X}_i]. \tag{1.10}$$

Take a moment to convince yourself that $i^* = 0$. To give some intuition on the problem, consider Figure 1.1, which shows a realization of X' and X''.

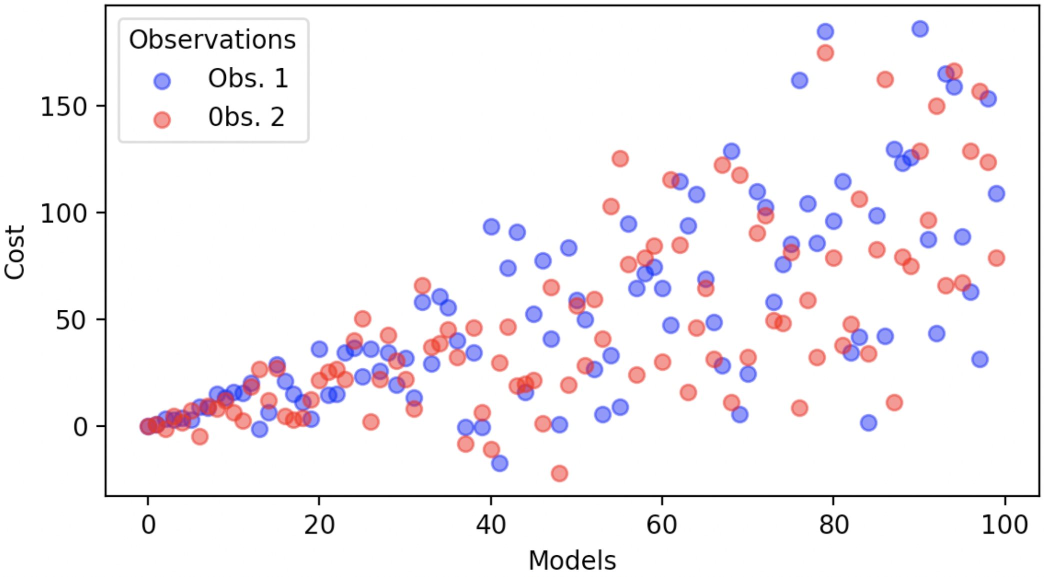


Figure 1.1

1.4.2 What happens if we do ERM?

If we follow standard ERM, then we would estimate i^* with

$$\hat{i} := \arg\min_{i \leq n} X'_i + X''_i.$$

That is, the popular wisdom in statistical learning advocates to estimate i^* with the index where the minimum of $X' + X''$ is. However, this approach fails in probability as n increases. This is because, with high probability, there is some $i > 0$ for which $X'_i + X''_i < X'_0 + X''_0$.

1.4.3 What happens if we do ME+PA?

ME+PA correctly estimates i^* much more often than ERM. To demonstrate this, we set $n = 1000$ and conducted an experiment where we drew at random

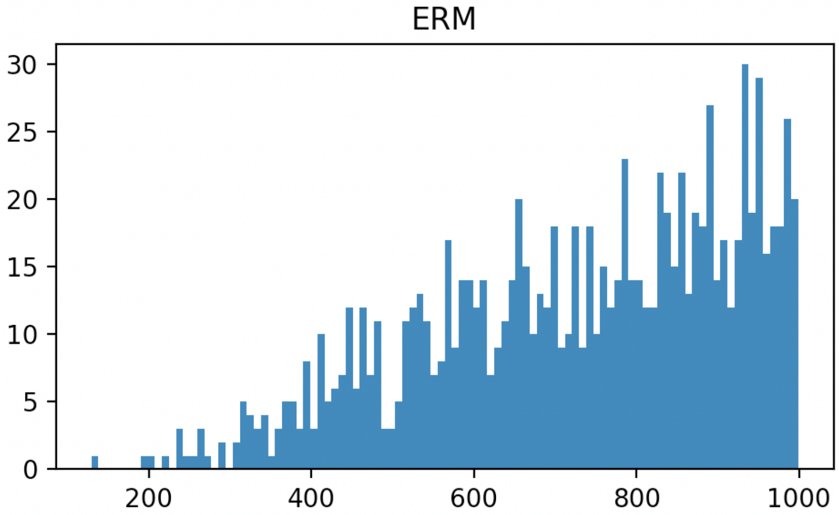


Figure 1.2: Histogram counting how often each estimate in $\{0, \ldots, 999\}$ was picked by ERM as the estimate for i^* across 1000 trials. Observe that ERM never picked the correct estimate.

1000 pairs (X', X'') of independent observations from $\mathfrak{X}$'s distribution. For each pair, we executed ERM and ME+PA to estimate i^*. We counted how often each value $i \leq n$ was chosen as the estimate by each method. Figures 1.2 and 1.3 demonstrate that in 20% of the cases, ME+PA selects i^* as the estimate, whereas ERM never picks it.

1.4.4 Application of ME+PA

We now illustrate the steps to apply ME+PA to this problem.

First, the hypothesis class is $\mathcal{C} = \{0, \ldots, n-1\}$ and the set $\mathcal{X}$ is $\mathbb{R}^n$.

Let $X \in \mathcal{X}$ be an observation. The cost function is $R(c, X) = X_c$. The Gibbs distribution is given by $p(c \mid X) \propto \exp\left(-1/T X_c\right)$. For the two given observations X' and X'', a combined Gibbs distribution is $p(c \mid X', X'') \propto \exp\left(-1/T\left(X'_c + X''_c\right)\right)$.

We now need to define a value for T. We guess a set of candidate values and, for each of them, we compute the posterior agreement kernel $\kappa(X', X'')$. We then pick the value that yielded the maximum posterior agreement kernel. Finally, we just choose the $\hat{c}$ with largest $p(\hat{c} \mid X', X'')$.

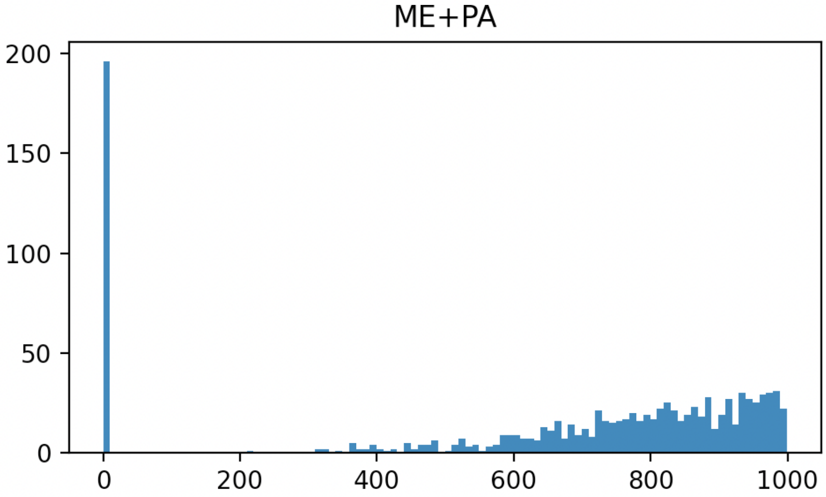


Figure 1.3: Histogram counting how often each estimate in $\{0, \ldots, 999\}$ was picked by ME+PA as the estimate for i^* across 1000 trials. In contrast to ERM, ME+PA picks the correct estimate around 20% of the time.

1.5 Maximum entropy

In this section, we explain the rationale behind the Gibbs distribution. Remember that for an observation X, we want to have a distribution $p(\cdot \mid X)$ such that, for $c \in \mathcal{C}$, $p(c \mid X)$ measures how much we believe that c is the right model for X. The base for that measure is the cost $R(c, X)$. The lower this cost is, the more we believe that c is the right model for X. We can therefore elicit the following requirement:

Requirement 1. *For any* $X \in \mathcal{X}$ *and any* $c_1, c_2 \in \mathcal{C}$, $R(c_1, X) \leq R(c_2, X) \Leftrightarrow P(c_1 \mid X) \leq P(c_2 \mid X)$.

There are too many distributions that fulfil this requirement. To narrow the set of candidate distributions, we follow the *maximum entropy principle* [**?**, **?**].

The maximum entropy principle states that when choosing between two distributions that equally fulfil some criteria, choose the one with the highest entropy. This is equivalent to choosing the distribution that is “closest” to the uniform distribution with respect to the Kullback-Leibler divergence. By aiming for the “most uniform” distribution, we aim for a distribution that does not show arbitrary preferences for one model over the other. The distribution should only show preferences based on the cost function.

Requirement 2. $P(\cdot \mid X)$ *shall have maximum entropy.*

Finally, we add some regularity requirements.

Requirement 3. *$P(\cdot \mid X)$ is a regular distribution. That is, $P(c \mid X) \geq 0$, for any $c \in \mathcal{C}$, $\sum_{c \in \mathcal{C}} P(c \mid X) = 1$, and $\mathbb{E}_{C \sim p(\cdot|X)}[R(C, X)] < \infty$.*

These requirements define a constrained optimization problem

$$\max_{p} \quad H[p] \tag{1.11}$$

$$s.t. \quad p(c_1) \leq p(c_2), \text{ for any } c_1, c_2 \in \mathcal{C} \text{ with } R(c_1, X) \leq R(c_2, X), \tag{1.12}$$

$$p(c) \geq 0, \text{ for } c \in \mathcal{C}, \tag{1.13}$$

$$\sum_{c \in C} p(c) = 1, \tag{1.14}$$

$$\mathbb{E}_{C \sim p}[R(C, X)] = \mu. \tag{1.15}$$

Here, μ is a hyper-parameter ensuring that the expectation is finite. We will later see that its exact value is not important and can be chosen using the posterior agreement principle.

We solve this problem by using Lagrange multipliers, forgetting about the inequality constraints, and then showing that the solution happens to fulfil those inequality constraints. The Lagrangian in this case is

$$\mathcal{L}(p, \lambda_1, \lambda_2) = H[p] + \lambda_1 \left(1 - \sum_{c \in \mathcal{C}} p(c)\right) + \lambda_2 \left(\mathbb{E}_{C \sim p}[R(C, X)] - \mu\right). \tag{1.16}$$

Therefore, for $c \in \mathcal{C}$,

$$\frac{\partial \mathcal{L}}{\partial p(c)} = -1 - \log p(c) + \lambda_1 + \lambda_2 R(c, X). \tag{1.17}$$

Setting this expression equal to zero and solving for $p(c)$, while using the constraint that $\sum_c p(c) = 1$, yields that

$$p(c) \propto \exp(-\lambda_2 R(c, X)). \tag{1.18}$$

Observe that $p(\cdot)$ already fulfils the constraint (1.13). If we enforce the constraint (1.12), then λ_2 must be non-negative. Furthermore, if $\lambda_2 = 0$, which is the case when $\mu = 1/\left|\mathcal{C}\right| \sum_c R(c, X)$, then $p(\cdot)$ is the uniform distribution over $\mathcal{C}$, which is not interesting for us as $p(\cdot)$ does not discriminate the different models in $\mathcal{C}$ in this case. We can therefore agree that $\lambda_2 > 0$. To make this distribution look more like the Gibbs distribution used in statistical physics to model particle systems, we define $T := 1/\lambda_2$. Hence, the solution of the constrained optimization problem is

$$p(c) \propto \exp\left(-\frac{1}{T}R(c, X)\right), \text{ for some } T > 0. \tag{1.19}$$

We refrain from computing the exact value for T as it is analytically very difficult and unnecessary. The value of T is defined by μ, which is a hyper-parameter. Hence, we just leave T as another hyper-parameter. T can also be seen as a concentration hyper-parameter, defining how concentrated $p(\cdot)$ is around the global minima of $R(c, X)$. To see this, we have the following exercise.

Exercise 1. *Prove that*

$$\lim_{T\to\infty} p(c) = \frac{1}{|\mathcal{C}|}. \tag{1.20}$$

$$\lim_{T\to 0} p(c) = \begin{cases} a & \textit{if } c \in \arg\min_{c\in\mathcal{C}} R(c, X) \\ 0 & \textit{otherwise.} \end{cases} \tag{1.21}$$

Here, a is a constant value.

Observe that p is unique. Any other p' that solves the constrained problem must also have the form given by Equation (1.19). Hence, the difference between p' and p must be in their corresponding values for T, but observe that a change in T induces a change in $\mathbb{E}_C[R(C, X)]$. As a result, it is impossible that $p \neq p'$ if both happen to solve the constrained optimization problem and have $\mathbb{E}_{C\sim p}[R(C, X)] = \mathbb{E}_{C\sim p'}[R(C, X)] = \mu$.

The results in this section can be naturally adapted in some cases when $\mathcal{C}$ is not discrete, like $\mathbb{R}^m$. In this case, one of the main challenges is to ensure that the Gibbs distribution has a bounded normalization constant. This is usually satisfied if the cost function goes to ∞ fast enough as $\|c\| \to \infty$.

Exercise 2. *Let $\mathcal{C} = \mathbb{R}^m$ and $\mathcal{X}$ denote all subsets of pairs $(x, y) \in \mathbb{R}^m \times \mathbb{R}$. Consider the sum-of-squares cost function used for linear regreesion, which is defined for $c \in \mathcal{C}$ and $X \in \mathcal{X}$ as follows:*

$$R(c, X) = \sum_{(x,y)\in\mathcal{X}} \left(y - c^\top x\right)^2 + \lambda \|c\|^2. \tag{1.22}$$

Demonstrate that the solution to the constrained optimization problem given by Equations 1.11—1.15 is given by

$$p(\cdot \mid X) \propto \exp\left(-\frac{1}{T}R(c, X)\right), \quad \textit{with } T > 0.$$

Observe that without the regularization term $\lambda \|c\|^2$, it would not be possible to satisfy all the constraints.

1.6 Relation between the holdout and ME+PA

Many training algorithms require hyper-parameters to define the hypothesis class $\mathcal{C}$. A common technique to select those hyper-parameters is *the hold-out.*

In the hold-out, we draw two samples X' and X'', usually called the *training sample* and the *testing sample* respectively. Models are trained in X' and then evaluated in X''. A natural evaluation metric is *the out-of-sample error*:

$$\mathbb{E}_{C|X'}\left[R(C, X'')\right] = \mathbb{E}_{C|X'}\left[-T \log p(C \mid X'')\right] \tag{1.23}$$

$$\propto - \int p(c \mid X') \log p(c \mid X'')\, dc. \tag{1.24}$$

Here, $p(C \mid X')$ and $p(C \mid X'')$ are Gibbs distributions induced by R and some temperature T.

The hold-out advocates to select the hyper-parameter that minimizes this error.

In contrast, posterior agreement advocates to select the hyper-parameter that maximizes the posterior agreement kernel induced by R and some temperature T:

$$\mathbb{E}_{C|X'}\left[p(C \mid X'')\right]. \tag{1.25}$$

We argue here that the posterior agreement kernel is less sensitive to noise than the out-of-sample error. To see this suppose that $X' = X + \Delta'$ and $X'' = X + \Delta''$, where $X \in \mathbb{R}^m$ and that Δ', Δ'' are distributed according to a Gaussian. Furthermore, we assume that $\|\Delta'\| < 1$ and $\|\Delta''\| < 1$.

Exercise 3. *Show that*

$$\mathbb{E}_{C|X'}[R(C, X'')] = \mathbb{E}_{C|X}[R(C, X)] + O(\|\Delta\|) \text{ and} \tag{1.26}$$

$$\mathbb{E}_{C|X'}[p(C \mid X'')] = \mathbb{E}_{C|X}[R(C, X)] + O(\|\Delta\|^2). \tag{1.27}$$

Use the following second-order Taylor approximation:

$$p(C \mid X') = p(C \mid X) + \left(\nabla_X p(C \mid X)\right)^\top \Delta + O(\|\Delta\|^2). \tag{1.28}$$

1.7 Informal justification and motivation

ToDo: Prepare intuitive explanation

ToDo: Demonstrate the entire workflow of ME+PA for clustering.

Chapter 2

Deterministic annealing

2.1 The problem of centroid-based cluster- ing

In this chapter, we consider the problem of centroid-based clustering. In this problem, an observation is a sample $X = \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^d$ of points and our objective is to compute a set $\{\theta_1, \ldots, \theta_K\} \subseteq \mathbb{R}^d$ of K centroids and a *cluster assignment* function $c : \{1, \ldots, N\} \to \{1, \ldots, K\}$, where $K \in \mathbb{N}$ is a hyperparameter. We let the hypothesis class $\mathcal{C}$ be all possible cluster assignment functions with fixed K.

As cost function, we choose the K-means cost function:

$$R(c, \theta, X) := \sum_{i \leq n} \left\| x_i - \theta_{c(i)} \right\|^2 . \tag{2.1}$$

Note that, in contrast to K-means, we neither require θ_k to be the mean of all points in X assigned to cluster k nor that $c(x)$ is the index of the centroid closest to x.

2.2 Chapter outline

In this chapter, we study how to apply simulated annealing to the problem of clustering, and derive a simplified version called deterministic annealing, proposed by Kenneth Rose [**?**]. We compare deterministic annealing against a popular ERM approach, K-means, and empirically show that deterministic annealing often provides solutions that generalize better than K-means. Furthermore, combining deterministic annealing with posterior agreement gives a method to estimate the number of clusters while executing deterministic annealing.

Deterministic annealing

Deterministic annealing originates from applying simulated annealing to clustering. We can summarize simulated annealing in this context in the following three steps.

1. Pick a first model c and an initial (high) temperature.
2. Use MCMC to draw a sample from $p(\cdot \mid X)$, the Gibbs distribution induced by $R(\cdot, \cdot, X)$, starting from c as the first model in the sample.
3. Decrease the temperature, let c be the last model in the MCMC sample, and go to Step 2.

We see in Section **??** that the Gibbs distribution induced by the cost function $R(\cdot, \cdot, X)$ is actually tractable. This means that we do not need to do MCMC sampling. We can instead directly computing the Gibbs distribution. The three steps become the following procedure, called *deterministic annealing*:

1. Pick an initial set of centroids and an initial (high) temperature.
2. Compute the Gibbs distribution $p(\cdot \mid X)$, for θ arbitrary.
3. Decrease the temperature and go to Step 2.

In Section **??**, we show that the centroids can then be computed as those that maximize the Gibbs distribution's entropy. However, this maximization is intractable, so we use the EM algorithm to efficiently compute an approximation. We combine all these insights to formulate the algorithm for deterministic annealing in Section 2.5.

In Section 2.6, we provide some experimental results comparing deterministic annealing against K-means.

In Section **??**, we describe an interesting behavior of deterministic annealing. When the temperature is high, there is only one cluster containing all points and whose centroid is just the sample mean. As the temperature decreases, this cluster decomposes into several clusters that continue to decompose as the temperature keeps decreasing. One could argue that at a sufficiently low temperature, each point becomes its own cluster.

Finally, in Section 2.8, we exploit this behavior and posterior agreement to determine the number of clusters while executing deterministic annealing. This yields an extra advantage of deterministic annealing over other standard clustering methods like K-means which demand to set the number of clusters in advance.

2.3 Tractability of the Gibbs distribution

We assume that our hypothesis class $\mathcal{C}$ is the set of all cluster assignments $c : \{1, \dots, N\} \to \{1, \dots, K\}$.ToDo: Observe that c's domain must be the set of numbers up to N and not an Euclidean domain. Otherwise, the Gibbs distribution is a continuous distribution! We define then the family of Gibbs distributions induced by the K-means cost function as all distributions over $\mathcal{C}$ of the form

$$p(\cdot \mid \theta, X) \propto \exp\left(-\frac{1}{T} R(c, \theta, X)\right). \tag{2.2}$$

Here, $T > 0$ is a hyper-parameter denoting the temperature and $\theta \in \mathbb{R}^{K \times d}$ are parameters denoting the cluster centroids.

The centroids are chosen to be hyper-parameters just for convenience. One could treat them as part of the hypothesis class, but this substantially complicates the analysis.

Theorem 6. *A Gibbs distribution induced by the K-means cost function factorizes as follows:*

$$p(c \mid \theta, X) = \prod_{i \le N} p(c(i) \mid \theta, X), \tag{2.3}$$

where

$$p(c(i) \mid \theta, X) \propto \exp\left(-\frac{1}{T} \left\|x_i - \theta_{c(i)}\right\|^2\right). \tag{2.4}$$

Proof. The Gibbs distribution is

$$p(c \mid \theta, X) = \frac{\exp\left(-\frac{1}{T} \sum_{i \le N} \left\|x_i - \theta_{c(i)}\right\|^2\right)}{\sum_{c \in \mathcal{C}} \exp\left(-\frac{1}{T} \sum_{i \le N} \left\|x_i - \theta_{c(i)}\right\|^2\right)}. \tag{2.5}$$

The numerator can be rewritten as follows:

$$\exp\left(-\frac{1}{T} \sum_{i \le N} \left\|x_i - \theta_{c(i)}\right\|^2\right) = \prod_{i \le N} \exp\left(-\frac{1}{T} \left\|x_i - \theta_{c(i)}\right\|^2\right). \tag{2.6}$$

We now apply the combinatorial trick that we studied in the lecture to rewrite the denominator. Unfortunately, I do not see a better way to explain this trick without the diagrams from the lecture.

$$\sum_{c \in \mathcal{C}} \exp\left(-\frac{1}{T} \sum_{i \le N} \left\|x_i - \theta_{c(i)}\right\|^2\right) = \dots = \prod_{i \le N} \sum_{k \le K} \exp\left(-\frac{1}{T} \left\|x_i = \theta_i\right\|^2\right). \tag{2.7}$$

Putting these results together yields that

$$p(c \mid \theta, X) = \frac{\prod_{i \leq N} \exp\left(-\frac{1}{T}\left\|x_i - \theta_{c(i)}\right\|^2\right)}{\prod_{i \leq N} \sum_{k \leq K} \exp\left(-\frac{1}{T}\left\|x_i - \theta_k\right\|^2\right)} \tag{2.8}$$

$$= \prod_{i \leq N} \frac{\exp\left(-\frac{1}{T}\left\|x_i - \theta_{c(i)}\right\|^2\right)}{\sum_{k \leq K} \exp\left(-\frac{1}{T}\left\|x_i - \theta_k\right\|^2\right)} \tag{2.9}$$

$$= \prod_{i \leq N} p(c(i) \mid \theta, X). \tag{2.10}$$

□

Theorem 6 shows that we can compute $p(c \mid \theta, X)$ in just $O(NK)$-time. You just have to compute $\exp\left(-\frac{1}{T}\left\|x_i - \theta_k\right\|^2\right)$, which is $O(1)$-time, for $i \leq N$ and $k \leq K$. Hence, computing $p(c(i) \mid \theta, X)$ takes $O(K)$-time, for $i \leq N$, yielding a total computing time of $O(NK)$.

2.4 Computing the centroids

The calculations from the previous section do not tell us the values of the centroids. Remember that we follow the maximum-entropy principle, so we compute the centroids that maximize the entropy of $p(\cdot \mid \theta, X)$. In this section, we use C to denote a random cluster assignment whose distribution is $p(\cdot \mid \theta, X)$.

We assume here that $\mathbb{E}_{C \sum p(\cdot \mid \theta, X)} = \mu$, for some fixed $\mu \in \mathbb{R}^+$ and that the temperature hyper-parameter of $p(\cdot \mid \theta, X)$ depends exclusively on μ.

Lemma 1. *If, for a set θ^* of centroids,*

$$\frac{\partial H\left[p(\cdot \mid \theta, X)\right]}{\partial \theta}\Big|_{\theta^*} = 0, \tag{2.11}$$

then

$$\mathbb{E}_{C \sim p(\cdot \mid \theta, X)}\left[\frac{\partial}{\partial \theta} R(C, \theta, X)\Big|_{\theta^*}\right] = 0. \tag{2.12}$$

Proof. We leave the proof details as an exercise and provide just a vague proof. For convenience, all expectations are with respect to a random cluster assignment $C \sim p(\cdot \mid \theta, X)$.

First, show that

$$H\left[p(\cdot \mid \theta, X)\right] = \frac{1}{T}\mathbb{E}\left[R(C,\theta,X)\right] + \mathbb{E}\left[\log\sum_{c\in\mathcal{C}}\exp\left(-\frac{1}{T}R(c,\theta,X)\right)\right]. \quad (2.13)$$

Recall that $\mathbb{E}\left[R(C,\theta,X)\right]$ is fixed to be constant, by construction. Hence, the first term on the right-hand side of the equation above is constant with respect to θ. Moreover, the second term does not depend on the random variable C. We get then that

$$H[p(\cdot \mid \theta, X)] = \text{const} + \log\sum_{c\in\mathcal{C}}\exp\left(-\frac{1}{T}R(c,\theta,X)\right). \quad (2.14)$$

Take the derivative with respect to on both sides and show that

$$\frac{\partial H[p\left(\cdot \mid \theta, X\right)]}{\partial\theta} = \mathbb{E}_{C\sim p(\cdot|\theta^*,X)}\left[\frac{\partial}{\partial\theta}R(C,\theta,X)\right]. \quad (2.15)$$

This equality yields the desired result. □

For an event A, we now denote by $\mathbf{1}A$ the indicator function. That is,

$$\mathbf{1}A(\omega)\begin{cases}1 & \text{if } \omega\in A \text{ and}\\ 0 & \text{otherwise.}\end{cases} \quad (2.16)$$

Lemma 2. *Let C denote a random cluster assignment and $C(i)$, for $i\leq N$, the cluster that C assigns to x_i. If $R(\cdot,\cdot,X)$ is the K-means cost function, then*

$$\frac{\partial}{\partial\theta_k}R(C,\theta,X) = 2\theta_k\sum_{i\leq N}\mathbf{1}\left\{C(i)=k\right\} - 2\sum_{i\leq N}x_i\mathbf{1}\left\{C(i)=k\right\}. \quad (2.17)$$

Proof. The proof is a straightforward use of vector calculus and is left as an exercise. □

Theorem 7. *The set θ^* of centroids that maximize the entropy of $p(\cdot \mid \theta, X)$ must satisfy the following conditions.*

$$\theta_k^* = \frac{\sum_{i\leq N}x_i\mathbf{P}\left(C(i)=k \mid \theta^*, X\right)}{\sum_{i\leq N}\mathbf{P}\left(C(i)=k \mid \theta^*, X\right)},\ \textit{for } k\leq K, \quad (2.18)$$

where

$$\mathbf{P}\left(C(i)=k \mid \theta^*, X\right) \propto \exp\left(-\frac{1}{T}\left\|x_i-\theta_k^*\right\|^2\right). \quad (2.19)$$

Proof. If θ^* maximizes the entropy of $p(\cdot \mid \theta, X)$, then, by Lemma 1,

$$\mathbb{E}_{C\sim p(\cdot|\theta^*,X)}\left[\frac{\partial}{\partial\theta}R(C,\theta,X)\big|_{\theta^*}\right] = 0. \tag{2.20}$$

Plugging Equation 2.17 in the equation above yields that

$$\mathbb{E}_{C\sim p(\cdot|\theta^*,X)}\left[\theta^*_k\sum_{i\le N}\mathbf{1}\left\{C(i)=k\right\} - \sum_{i\le N}x_i\mathbf{1}\left\{C(i)=k\right\}\right] = 0. \tag{2.21}$$

Apply now linearity of the expectation and the fact that $\mathbb{E}\left[\mathbf{1}A\right] = \mathbf{P}\left(A\right)$ to get that

$$\theta^*_k\sum_{i\le N}\mathbf{P}\left(C(i)=k \mid \theta^*, X\right) - \sum_{i\le N}x_i\mathbf{P}\left(C(i)=k \mid \theta^*, X\right) = 0. \tag{2.22}$$

The desired condition follows from this equation. □

Unfortunately, Equation **??** does not give us a closed formula to compute θ^*. We can still attempt to estimate θ^* iteratively, with the following procedure.

1. Set $t \leftarrow 0$ and θ^*_t to an arbitrary value.
2. Set $t \leftarrow t+1$ and

$$\theta^*_{t+1} \leftarrow \frac{\sum_{i\le N}\mathbf{P}\left(C(i)=k \mid \theta^*_t, X\right)x_i}{\mathbf{P}\left(C(i)=k \mid \theta^*_t, X\right)}. \tag{2.23}$$

This procedure converges to a local maximum of $H[p(\cdot \mid \theta, X)]$.

Exercise 4. *Demonstrate that the procedure above also results from applying the EM-algorithm to the following maximization problem:*

$$\max_\theta \log\sum_{c\in\mathcal{C}}p(X, c \mid \theta), \tag{2.24}$$

where $p(X, c \mid \theta) := p(c \mid \theta, X)p(X)$ and $p(X)$ is the pdf of the phenomenon where X comes from. Use this to demonstrate why the procedure above converges.

2.5 Deterministic annealing

We now collect all insights from this chapter and use them to propose an improved version of simulated annealing, called *deterministic annealing.* Algorithm **??** provides the details. The idea of this procedure is the following.

1. We set a high value for the temperature and start with arbitrary centroids.
2. We alternate between (i) computing the maximum-entropy distribution for a random cluster assignment while fixing the centroids and (ii) computing the centroids that maximize the entropy of that distribution. This alternation is repeated until convergence of the centroids.
3. We reduce the temperature and repeat Step 2. To avoid having repeated centroids, we add a small amount of noise.

We remark some differences between deterministic annealing and simulated annealing:

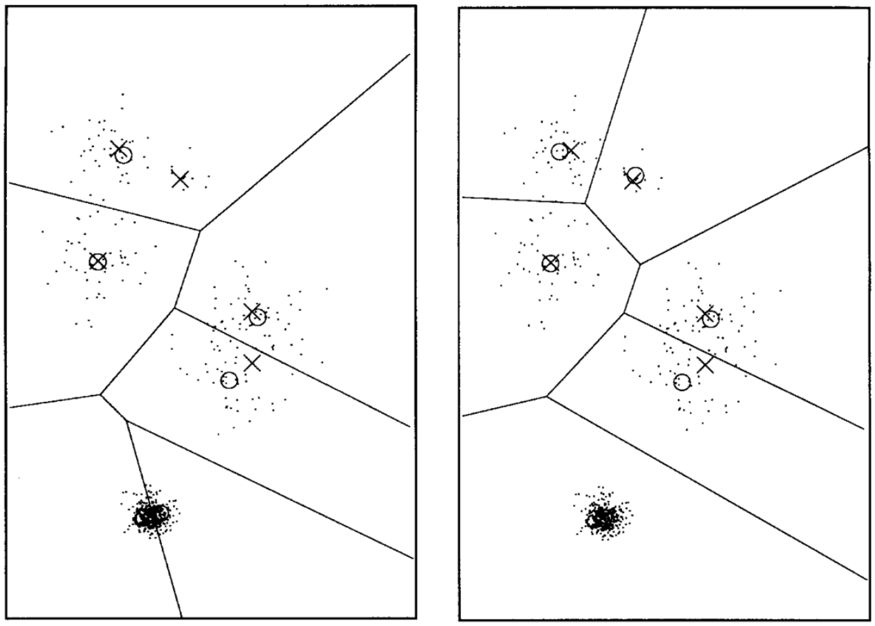
- There is no MCMC sampling. This is because the Gibbs distribution induced by the K-means cost is tractable. So we can just compute it directly.
- The centroids are treated as parameters of the Gibbs distribution and not as part of the hypothesis class. This is done mainly for convenience. A Gibbs distribution that also treats the centroids as random variables makes the whole procedure intractable.

2.6 Experimental results

Figure **??**, from Rose [**?**, **?**], compares deterministic annealing with K-means on a particular dataset X which is a sample from a mixture of six Gaussians with different means and different covariance matrices. K-means depends on the initialization, so it was run 25 times. The result shown in the figure was obtained only once, and in more than 80% of the runs, K- means was stuck in a local minimum. In contrast, deterministic annealing is independent of the initialization and finds the global minimum more often than K-means. For both K-means and deterministic annealing, 6 centroids were used.

Algorithm 1 Deterministic annealing

```
Let
      $\epsilon > 0$ be a temperature threshold,
      reduce$(\cdot)$ be a function for decreasing the temperature.
      close$(\cdot, \cdot)$ be a function that evaluates if two matrices are close.
function DetAnn($\epsilon$, reduce, close)
    $T \leftarrow \infty$                          ▷ $\infty$ is a sufficiently large value.
    $\theta \leftarrow \$$                          ▷ Define arbitrary initial centroids.
    while $T > \epsilon$ do
        repeat
            $\theta_0 \leftarrow \theta$
            Compute $\mathbf{P}\left(C(i) = k \mid \theta_0, X\right)$, for $i \leq N$ and $k \leq K$
            $\theta \leftarrow \dfrac{\sum_{i \leq N} \mathbf{P}\left(C(i) = k \mid \theta_0, X\right) x_i}{\sum_{i \leq N} \mathbf{P}\left(C(i) = k \mid \theta_0, X\right)}$
        until close$(\theta, \theta_0)$
        Add a small amount of noise to each centroid in $\theta$.
        $T \leftarrow$ reduce$(T)$
    end while
    return $\theta$, $\{\mathbf{P}\left(C(i) = k \mid \theta, X\right) \mid i \leq N, k \leq K\}$
end function
```



(a) K-means

(b) Deterministic annealing

Figure 2.1: Comparison of (a) K-means and (b) deterministic annealing. The $\times$ marks are the means of the Gaussian components. The $\bigcirc$ marks are the centroids computed by the algorithm.

2.7 Phase transitions

We now study the behavior of the centroids as the temperature decreases. We will see that, at the beginning when the temperature is high, all centroids are equal to the sample mean. As a result one can assume that there is only one cluster. As the temperature decreases, this cluster eventually decomposes into a few clusters, which continue decomposing into more clusters as the temperature keeps decreasing.

Figure 2.2, from Rose [**?**], illustrates this behavior. The four subfigures show the centroids (× marks) at four different points of the execution of deterministic annealing with 9 centroids. Here, X is a sample from a mixture of 9 Gaussians. The subfigures are sorted by decreasing temperature. The curves in the graphic represent points that have all the same probability of belonging to one particular cluster. Observe how the apparent number of centroids change as the temperature decreases. In Figure 2.2a, there seems to be only three centroids, but what actually happens is that some of the 9 centroids are very close to each other. Figure 2.2b shows the centroids after the temperature has decreased. There seems to be now 5 centroids. In Figure 2.2c we have a lower temperature and now 7 apparent centroids. Finally, in Figure 2.2d, all 9 centroids are at different locations and close to the means of the Gaussian mixture. We now investigate this behavior analytically. We do this through a sequence of lemmas whose proofs are left as exercises.

2.7.1 There is only one cluster at a high temperature

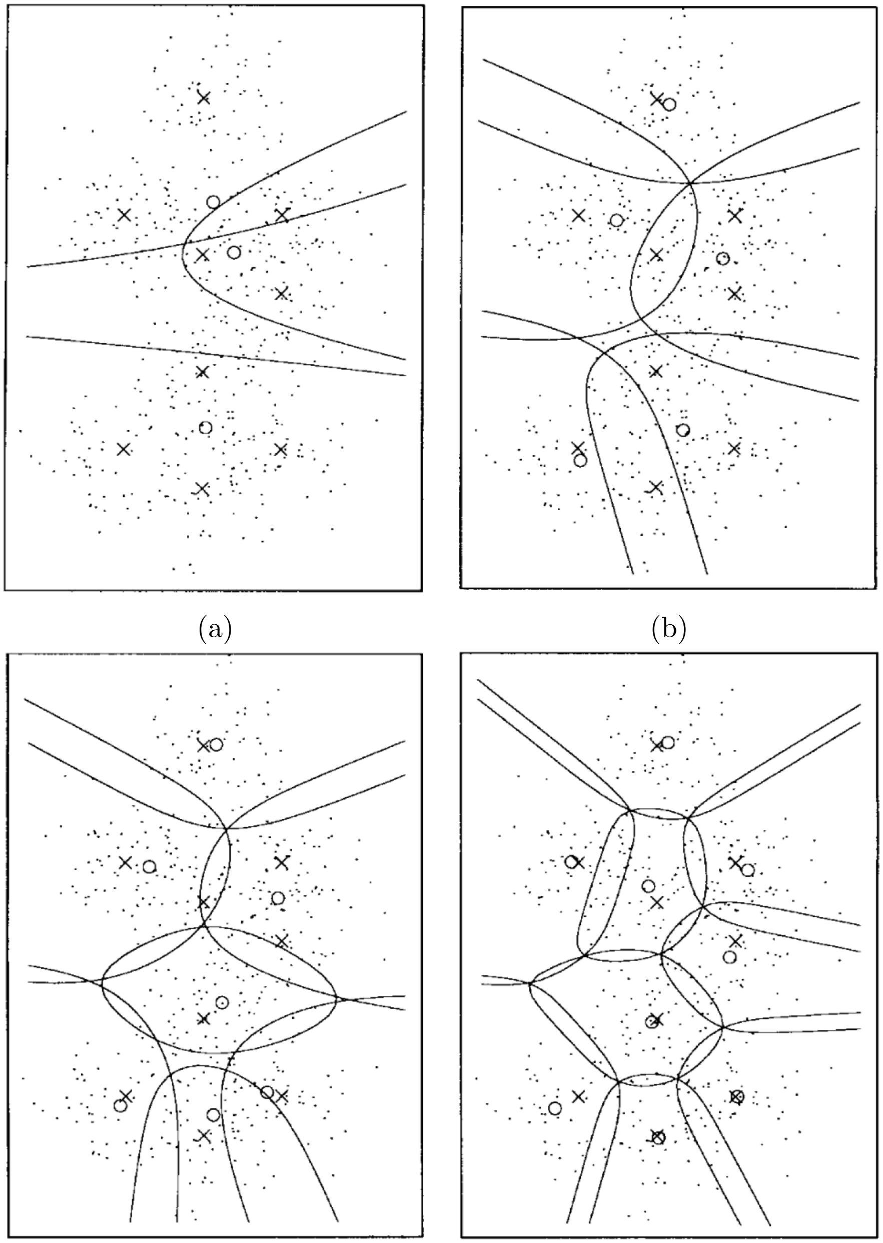
Lemma 3.

$$\lim_{T\to\infty} \mathbf{P}\left(C(i)=k \mid \theta, X\right) = \frac{1}{K}. \tag{2.25}$$

$$\lim_{T\to 0} \mathbf{P}\left(C(i)=k \mid \theta, X\right) = \begin{cases} 1 & \textit{if } \theta_k \textit{ is the centroid closest to } x_k \textit{ and} \\ 0 & \textit{otherwise.} \end{cases} \tag{2.26}$$

Proof. This is a consequence of Exercise □

2.8 Deterministic annealing and posterior agreement



(a)

(b)

(c)

(d)

Figure 2.2

Chapter 3

An information-theoretic foundation for posterior agreement

Assumption

3.1 Introduction

This chapter proposes an information-theoretic foundation for posterior agreeement, originally proposed by Joachim Buhmann [?] and studied by Alex Gronskiy as part of his doctoral thesis [?]. Posterior agreement can be understood as a method for *validating algorithms* that solve stochastic optimization problems of the form

$$\min_{c} \mathbb{E}_X \left[R(c, X) \right].$$

Here, $R(c, X)$ is the result of evaluating a cost function on a candidate solution c on an instance of the problem described by the random variable X. The expectation is computed with X's distribution.

In posterior agreement, we assume that an algorithm is a function that computes, from a given observation X', a *posterior distribution* $p(\cdot \mid X')$ over a finite space $\mathcal{C}$ of feasible solutions. Every algorithm can be converted into such an algorithm, even if it is deterministic. In this case, the posterior distribution is just a distribution that assigns probability mass one to the algorithm's output.

Posterior agreement assesses the performance of an algorithm by computing the *expected log posterior agreement*:

$$\mathbb{E}_{X',X''} \log \left(|\mathcal{C}|\, \kappa\left(X', X''\right)\right), \tag{3.1}$$

where $\kappa\left(X', X''\right)$ is the *posterior agreement kernel*:

$$\kappa\left(X', X''\right) := \sum_{c \in \mathcal{C}} p(\cdot \mid X') p(\cdot \mid X'').$$

The expected log posterior agreement requires the joint probability distribution of X' and X'', which is often unknown. One only has access to a handful of observations, at least two: X' and X''. In this case, one can use the *empirical log posterior agreement*:

$$\log \left(|\mathcal{C}|\, \kappa\left(X', X''\right)\right). \tag{3.2}$$

Moreover, the empirical log posterior agreement is intended to be a metric to compare different algorithms. Therefore, it is often sufficient to measure $\kappa\left(X', X''\right)$.

We show how posterior agreement can compare, for example, Prim's, Kruskal's, and the reverse-delete algorithm for computing spanning trees for graphs whose edge weights are defined by random variables, as in the example above. Posterior agreement can also compare among different cost functions and hyperparameter values, when training machine learning models.

Therefore, posterior agreement advocates that, in the context of stochastic optimization, *algorithms should aim for maximizing the posterior agreement kernel, given two instances of the problem*, rather than minimizing the cost function for either instance or an aggregate of these instances.

ToDo: Organization of these notes.

3.2 Motivation

Figure 3.1 gives two graphs modeling a city with four main locations: North, East, South, and West. There is a road connecting any two different locations and, depending on the day, traversing that road by car takes some amount of time. The number labeling each edge indicates that time. Observe that the time varies with each day.

Consider the minimum spanning tree of each of these two graphs. For Monday's graph, the tree is the one with North at the root and all other locations as children of North. For Tuesday's graph, the tree is the one with

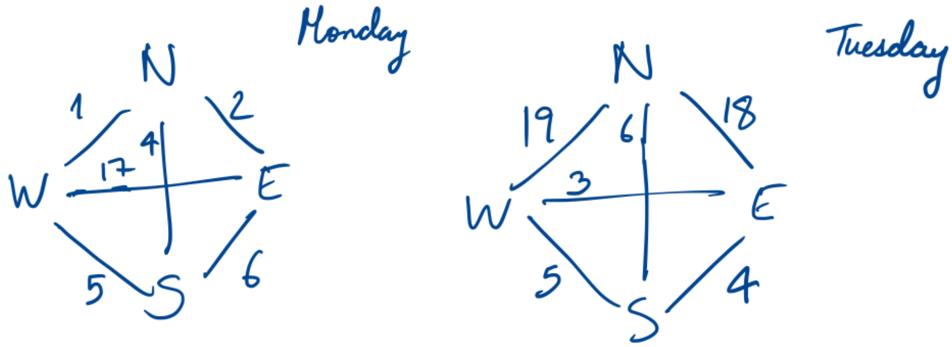


Figure 3.1: A graph with its observed edge weights for two different days.

South at the root and all other locations as children of South. We call these two trees the North and South trees, respectively.

Can we estimate from this information what Wednesday's graph's minimum spanning tree will look like?

The city graph can be understood as a random variable X. More precisely, it is a collection of 6 random variables, each of them defining an edge's weight on a particular day. Hence, the weight $R(c, X)$ of a tree c in X, which is the sum of the edge weights in c, is also a random variable. We can then formulate the problem of the minimum spanning tree for our case as the following stochastic optimization problem:

$$\arg\min_{c \in \mathcal{C}} \mathbb{E}\left[R(c, X)\right], \tag{3.3}$$

where $\mathcal{C}$ denotes all spanning trees of the city graph.

In practice, the main challenge in this type of optimization problems is that we do not know X's probability distribution. One natural way to go around this issue and approximately solve Problem 3.3 is to substitute $\mathbb{E}\left[R(c, X)\right]$ with an *empirical estimate*:

$$\mathbb{E}\left[R(c, X)\right] \approx \sum_{i \leq N} R(c, X_i),$$

where $\{X_1, \ldots, X_n\}$ is a sample of observed values of X. When n is sufficiently large and each X_i is independent from the others, then by the law of large numbers, this becomes a good approximation of $\mathbb{E}\left[R(c, X)\right]$. We call the *empirical risk minimizer* the solution of the problem:

$$\arg\min_{c \in \mathcal{C}} \sum_{i \leq n} \left[R(c, X_i)\right], \tag{3.4}$$

We present later scenarios where the empirical risk minimizer is not the best solution we can obtain, especially when we have only very few observations-Demonstrate this. One example is the problem of computing the minimum of an array $X = (X_1, \ldots, X_n)$ of random variables. Given just two observations X^1 and X^2, approximating $\min_i \mathbb{E}X_i$ with $\min_i X_i^1 + X_i^2$ is not the best we can do.

3.3 Overview

Posterior agreement originates from formalizing an algorithm $\mathcal{A}$ *as a communication channel* by which a sender and a receiver communicate outputs from $\mathcal{A}$. The robustness of an algorithm can be measured by the *capacity* of that communication channel, where the capacity defines the maximum number of distinguishable messages that can be communicated through the channel.

Shannon's channel coding theorem

Consider a channel by which a sender can transmit bits to a receiver. Assume that the channel is noisy in the sense that a bit can be flipped during transmission with probability $\epsilon < 0.5$. The sender and the receiver agree on transmitting only bitstrings of length n. We call these bitstrings *codewords*. Observe that if $\epsilon = 0$, then the sender can reliably communicate 2^n different messages to the receiver, by agreeing in advance with the receiver on a way to encode each message as one codeword of length n. This correspondence is called a *code*.

In practice, $\epsilon > 0$. Therefore, the sender and the receiver need to agree on a code that is robust to the channel's noise. We now show two examples of codes:

- They could agree on just 2 messages, encoded as $00\ldots0$ and $11\ldots1$, respectively. If the codeword length is sufficiently large, then with high probability, less than half of the bits will be flipped during transmission. As a result, the receiver can almost surely identify the message from the received codeword, by just counting the frequency of each bit in the codeword.

- They agree on sending 2^n messages, each encoded with a unique codeword of length n. With high probability, some of the bits will be flipped during transmission and the receiver will fail to identify the message that the sender tried to communicate.

Observe that these two codes represent two extremes, as $n \to \infty$. On one hand, the first code communicates only 2 messages, but with high probability of success. On the other hand, the second code communicates 2^n messages, but with low probability of success. One can also imagine other codes that strike a balance between the number of messages to be communicated and the success probability.

Shannon's coding theorem answers the following question. *What is the code that maximizes the number of messages that the sender can communicate to the receiver, while attaining a probability of success close to 1, as $n \to \infty$?* When $\epsilon > 0$, this number is clearly below 2^n, but since $\epsilon < 0.5$, this number must be positive. Shannon shows that this number is 2^{nc}, where c is *the channel's capacity*, a quantity that is defined by the channel. Therefore, channels with higher capacity allow the communication of more messages at the same codeword length.

Posterior agreement

Posterior agreement originates from modeling an algorithm $\mathcal{A}$ as a communication channel $C_{\mathcal{A}}$, where a sender communicates outputs from $\mathcal{A}$ to a receiver. We argue that the capacity of $C_{\mathcal{A}}$ is defined by $\mathcal{A}$'s robustness to noise in the input. That is, if $\mathcal{A}$ is robust to noise, then it is possible to communicate many more messages through $C_{\mathcal{A}}$ than when $\mathcal{A}$ is sensitive to noise.

We give an overview of this argument next. A rigorous argument is given in Section 3.5.

We assume given an *instance space* $\mathcal{X}$, comprising all possible observations, and a *solution space* $\mathcal{C}$, comprising all possible solutions. A phenomenon is then a probability distribution over $\mathcal{X}$. We assume that algorithms intending to solve Problem 3.3 receive in the input an observation X' and output a distribution $p(\cdot \mid X')$ over $\mathcal{C}$.

Example 1. *In the minimum spanning tree problem, a codeword is a spanning tree, a phenomenon is a distribution governing a graph's edge weights, and an observation is a graph whose edge weights are drawn from a fixed distribution.*

Example 2. *In the centroid-based clustering problem, a codeword is a cluster assignment function and a set of centroids, an observation is a set of points to be clustered, and a phenomenon is a distribution where the points are drawn from.*

For our analysis, we assume that each instance space contains all observations of a given "size" $n \in \mathbb{N}$. This size n is a notion that measures the observations' and phenomena's complexity. For example, in the minimum spanning tree problem, an instance space contains only all weighted graphs with a fixed number n of vertices.

For an algorithm $\mathcal{A}$, we define a communication channel $C_{\mathcal{A}}$ that works as follows. To use the channel, a sender picks an instance X', drawn from a phenomenon p_X, computes and inputs $p(\cdot \mid X')$ to the channel. The channel replaces $p(\cdot \mid X')$ with $p(\cdot \mid X'')$, where X'' is a fresh new instance drawn from p_X. The channel outputs $p(\cdot \mid X'')$ to the receiver.

We now emphasize the key insight of this modeling. If $\mathcal{A}$ is robust to the fluctuations in X', then there should not be much difference between $p(\cdot \mid X')$ and $p(\cdot \mid X'')$. In contrast, if $\mathcal{A}$ is very sensitive to the fluctuations in X', then $p(\cdot \mid X')$ and $p(\cdot \mid X'')$ may be very different. Hence, $\mathcal{A}$'s robustness to noise defines how many different "messages" can we send through this channel. We conclude then that $\mathcal{A}$'s robustness is measured by the capacity of this channel $C_{\mathcal{A}}$. This capacity, as we show in Section 3.5, can be estimated by the expected log posterior agreement. For this reason, we argue that algorithms intended to solve Problem 3.3 shall be measured by their expected log posterior agreement.

Observe the following analogies to Shannon's coding theory. Channels have a capacity that define the maximum number of distinguishable messages that can be communicated. Channels with higher capacity are preferable, as they allow more different messages to be communicated. Analogously, we argue that algorithms can be modeled as channels and, therefore, have a capacity, which we later show to be the expected log posterior agreement. Hence, we argue that algorithms with higher expected log posterior agreement are preferable, as they allow more different messages to be communicated.

Protocol overview

The protocol describes a code by which a sender can communicate messages to a receiver. Let $\mathcal{A}$ be an algorithm intending to solve Problem 3.3. In our case, a message is a phenomenon and a codeword is the output $p(\cdot \mid X')$ of $\mathcal{A}$ when given an observation X' from a phenomenon as input. Analogous to Shannon's coding theorem, the sender and the receiver aim to maximize the number of different messages that the sender can communicate, while ensuring that the receiver's probability of success goes to 1 as $n \to \infty$.

Figure 3.2 gives an overview of the protocol. The sender must describe a phenomenon q to a receiver through a noisy channel. The sender makes an observation X' from q, uses $\mathcal{A}$ to compute $p(\cdot \mid X')$, and sends through the channel to the receiver. The channel is noisy and we represent its noise by replacing $p(\cdot \mid X')$ with $p(\cdot \mid X'')$, where X'' is another observation from q.

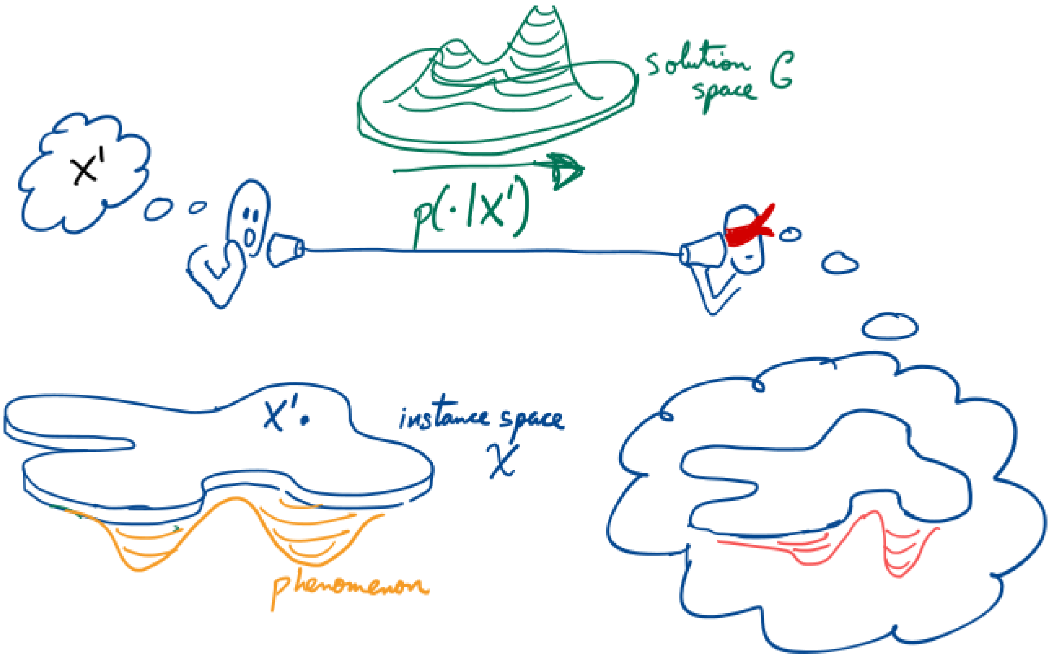


Figure 3.2

The receiver succeeds if he is able, using $p(\cdot \mid X'')$, to distinguish X' from observations from other different phenomena.

We remark that this protocol is just a thought experiment. The protocol is computationally impossible for many interesting optimization problems. This is because the protocol requires that we know the underlying distribution behind the observations of one phenomenon. This is not possible in most of the cases. Nonetheless, the protocol provides a formal justification and motivation for posterior agreement.

Protocol example

We now give a more precise overview by showing how posterior agreement works with a simple example. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two algorithms that estimate the mean of a univariate distribution, given only a sample from that distribution. For example, $\mathcal{A}_1$ fits a Gaussian to the sample via maximum-likelihood estimation whereas $\mathcal{A}_2$ does the same, but only using only the sample maximum and minimum. Suppose that both algorithms output Gaussian distributions that indicate where they believe that the mean is.

The protocol works as follows. Fix $n \in \mathbb{N}$, which denote the size of all the observations in the instance space. In our case, n denotes the sample size.

The sender and the receiver are given the algorithm under evaluation $\mathcal{A}_i$ and then they choose the size $m_n \in \mathbb{N}$ of the set of messages that they the sender will attempt to communicate. Recall that they want to choose m_n as large as possible. However, a large m_n increases the probability P_n that the receiver fails to recognize the message from the codeword transmitted by the sender. We later see that the best choice for m_n is defined by $\mathcal{A}_i$.

The protocol proceeds then as follows:

1. (Figure 3.3) The sender and the receiver agree on a set of $m := m_n$ phenomena, which we represent with the probability distributions $q_1, q_2, \ldots, q_m$.

2. (Figure 3.3) The sender and the receiver together make one observation X'_i of each phenomenon q_i, with $i \leq m$. In this case, an observation is a sample of points from q_i. Afterwards, they use $\mathcal{A}_i$ to compute a distribution $p(\cdot \mid X')$ for each observation X'.

3. (Figure 3.4) An observation X' is chosen out of these m observations uniformly at random. X' is given to the sender, but kept secret from the receiver.

4. (Figure 3.5) The sender sends $p(\cdot \mid X')$ to the receiver through a *noisy communication channel*. This channel replaces $p(\cdot \mid X')$ with $p(\cdot \mid X'')$, where X'' is a fresh new observation from the same phenomenon where X' comes from.

5. (Figure 3.6) The receiver gets $p(\cdot \mid X'')$ and must now guess which observation in $\{X'_1, \ldots, X'_m\}$ the sender chose in Step 3. For this, the receiver uses the natural approach of guessing the observation $\hat{X}$ for which $p(\cdot \mid \hat{X})$ overlaps the most with $p(\cdot \mid X'')$. In other words, the receiver guesses the observation $\hat{X}$ that fulfills:

$$\kappa\left(\hat{X}, X'\right) \geq \kappa\left(Y, X'\right), \text{ for all Y.}$$

6. If $\hat{X} = X'$, the receiver has succeeded.

Receiver's probability of failure

We show in Section 3.5.4 that the receiver's failure probability is bounded above by

$$\exp\left(-\mathbb{E}_{X',X''}\left[\log\left(|\mathcal{C}|\,\kappa\left(X', X''\right)\right)\right] + \epsilon \log|\mathcal{C}| + \log m\right),$$

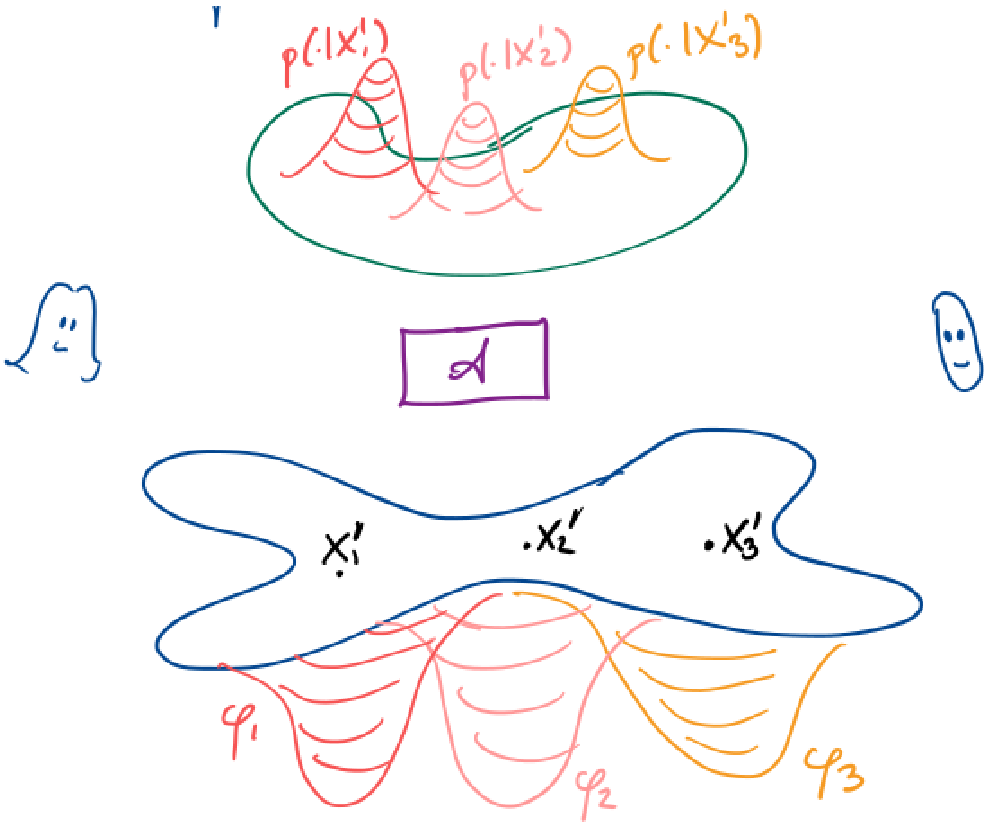

$p(\cdot|X'_1)$
$p(\cdot|X'_2)$
$p(\cdot|X'_3)$
$\mathcal{A}$
X'_1
X'_2
X'_3
φ_1
φ_2
φ_3

Figure 3.3

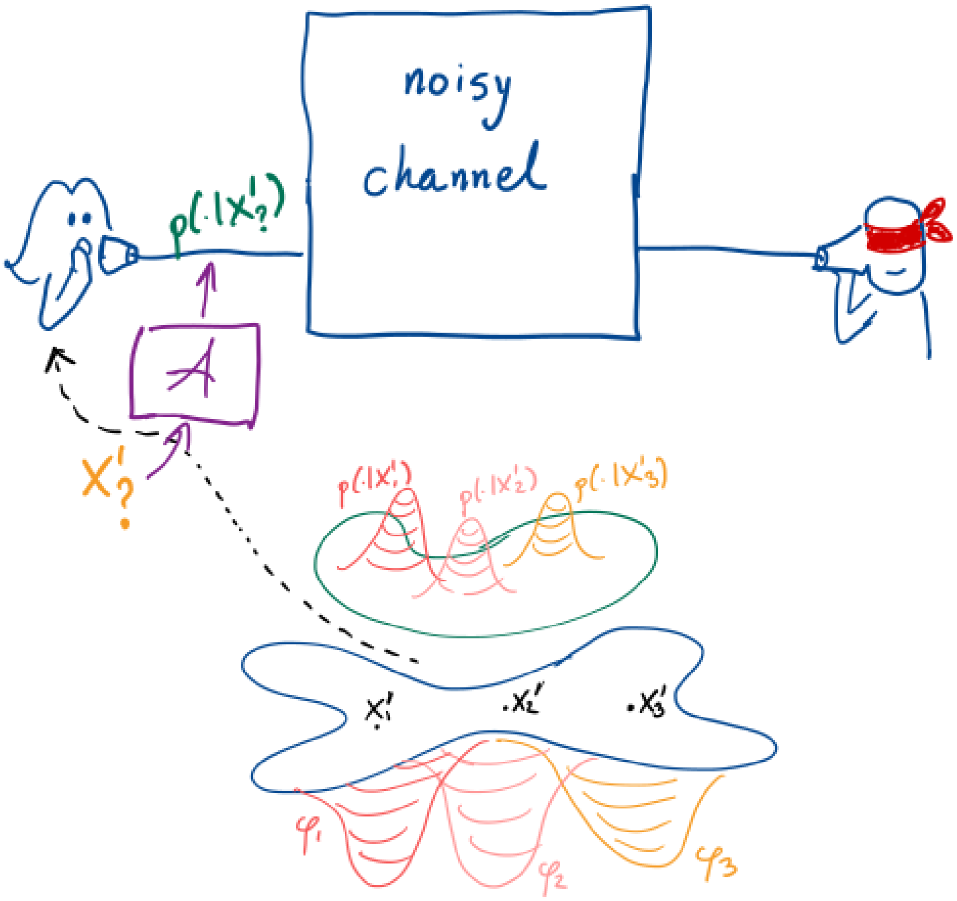

noisy
channel
$p(\cdot|X'_?)$
$\mathcal{A}$
$X'_?$
$p(\cdot|X'_1)$
$p(\cdot|X'_2)$
$p(\cdot|X'_3)$
X'_1
X'_2
X'_3
φ_1
φ_2
φ_3

Figure 3.4

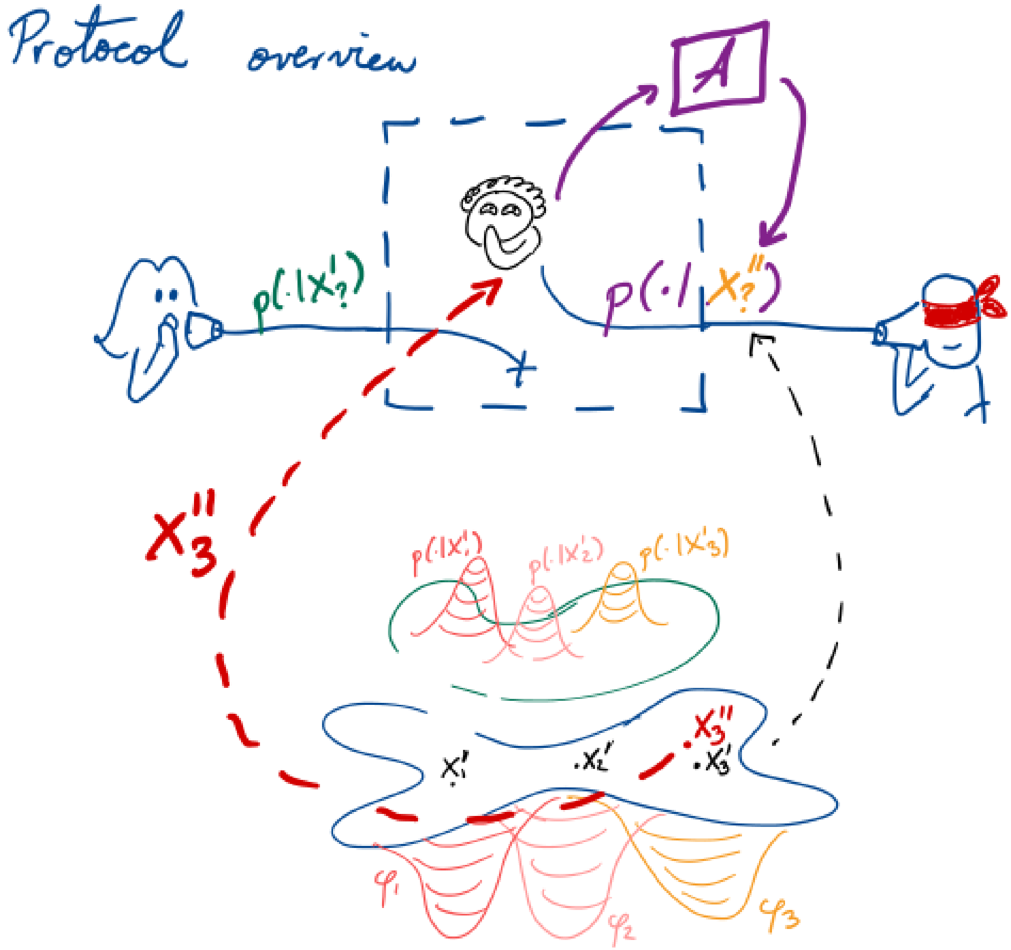


Figure 3.5

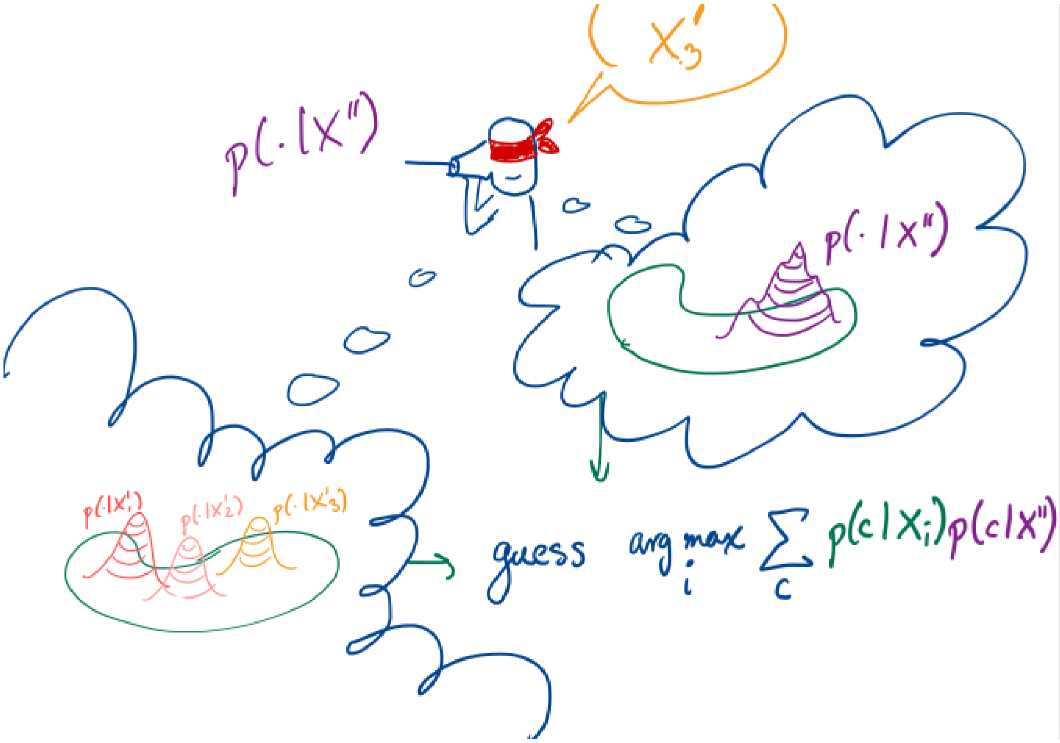

X'_3
$p(\cdot|X'')$
$p(\cdot|X'')$
$p(\cdot|X'_1)$
$p(\cdot|X'_2)$
$p(\cdot|X'_3)$
guess $\arg\max_i \sum_c p(c|X_i)p(c|X'')$

Figure 3.6

where $\epsilon > 0$ is arbitrary, $\mathcal{C}$ is the solution space, and m is the number of observations defined in Step 1.

Assume now that $\log|\mathcal{C}| = \Omega(n)$. This is a reasonable assumption, as for many interesting problems, $\mathcal{C}$ grows exponentially on n. Observe that the algorithm can only influence $\kappa(X', X'')$. If ϵ is sufficiently small and the algorithm ensures that

$$\mathbb{E}_{X',X''} \log\left(|\mathcal{C}|\,\kappa(X', X'')\right) - \log m = \Omega(n),$$

then the receiver's failure probability becomes 0 as $n \to \infty$. The algorithm can ensure this by *maximizing the expected log posterior agreement*. The larger this quantity is, the higher m can be and the more messages the sender can communicate to the receiver.

3.4 Shannon's channel coding theorem

To properly formulate and analyze the communication protocol above, we build upon Shannon's channel coding theorem. This theorem measures how much information we can optimally send through a communication channel. This section is mainly based on Chapter 7 from Thomas and Cover [**?**].

3.4.1 Channels

We understand a channel as a medium by which one sender can send symbols from a fixed set $\mathfrak{A}$ to a receiver. We allow channels to be noisy, meaning that the symbol a can be altered to another symbol b with probability $p(b \mid a)$ during the transmission through the channel. We do not allow the receiver to give feedback to the sender.

Definition 8. *A* (noisy) channel *is a pair* $\left(\mathfrak{A}, \{p(\cdot \mid a)\}_{a\in\mathfrak{A}}\right)$*, where* $\mathfrak{A}$ *is a set and* $p(\cdot \mid a)$*, for* $a \in \mathfrak{A}$*, is a conditional distribution on* $\mathfrak{A}$*.*

For convenience, we sometimes write just $\mathcal{P}$ to denote the family of distributions $\{p(\cdot \mid a)\}_{a\in\mathfrak{A}}$. Observe that we assume that transmissions are independent from each other. What the receiver gets does not influence what the user sends or the channel's conditional probabilities in the future.

Example 3. *The* binary channel *is the channel with* $\mathfrak{A} = \{0, 1\}$ *and* $p(b \mid a) = \mathbb{I}\{a = b\}$*, for* $a, b \in \mathfrak{A}$*, where* $\mathbb{I}$ *is the indicator function. This is a channel where there is no noise interference.*

Example 4. *The* noisy binary channel *is the binary channel, but with*

$$p(b \mid a) \begin{cases} 1-\epsilon & \textit{if } a = b \textit{ and} \\ \epsilon & \textit{if } a \neq b. \end{cases}$$

Unless stated otherwise, we assume that $0 < \epsilon \leq 0.5$.

Example 5. *The* typewriter channel *is the channel with* $\mathfrak{A} = \{\mathfrak{a}, \mathfrak{b}, \ldots, \mathfrak{z}\}$ *and* $p(b \mid a) = \mathbb{I}\{a = b\}$, *for* $a, b \in \mathfrak{A}$, *where* $\mathbb{I}$ *is the indicator function.*

Example 6. *The* noisy typewriter channel *is the channel with* $\mathfrak{A} = \{\mathfrak{a}, \mathfrak{b}, \ldots, \mathfrak{z}\}$, *but with*

$$p(b \mid a) \begin{cases} 1-\epsilon & \textit{if } a = b, \\ \epsilon & \textit{if } a = b + 1 \textit{ mod } 26. \end{cases}$$

3.4.2 Channel capacity

Which of the channels above sends *the most information per transmission*? Intuitively, a letter has more information than a bit and the presence of noise affects the amount of information we send in one transmission. Indeed, sending one letter through the typewriter channel provides more information than sending one letter through a noisy typewriter channel. Also, a letter has more information than a bit. Hence, we say that the typewriter channel has *the most capacity*: one transmission through this channel carries in average more information than one transmission through any of the other three channels. Similarly, we say that the noisy binary channel has *the least capacity.*

We now formally define channel capacity.

Definition 9. *For a channel* $\left(\mathfrak{A}, \{p(\cdot \mid a)\}_{a \in \mathfrak{A}}\right)$, *its capacity is*

$$\max_{p(\cdot)} I(S; \hat{S}),$$

where S *and* $\hat{S}$ *are random variables denoting a symbol input to the channel and the output symbol, respectively, when* S *is distributed according to* $p(\cdot)$. *We refer to* $I(S; \hat{S})$ *as the channel's* input-output mutual information *and the joint distribution of* S *and* $\hat{S}$ *as the* input-output distribution.

Example 7. *For the binary channel, we can reliably send one bit per transmission. This corresponds to the channel's capacity,*

$$\max_{p(\cdot)} I(S; \hat{S}) = \max_{p(\cdot)} \{H(S) - H(\hat{S} \mid S)\} = \max_{p(\cdot)} H(S) = \max_{p(\cdot)} H(S) = 1.$$

The second equality follows from the fact that $\hat{S} = S$, so $H(\hat{S} \mid S) = 0$. The last equality follows from the fact that the distribution that maximizes the entropy of a Bernoulli random variable is the uniform distribution, which yields an entropy of 1 bit.

Example 8. *A similar line of reasoning shows that the typewriter channel's capacity is $\max_{p(\cdot)} H(S) = \log 26 \approx 4.7$. This corresponds to the intuition that we can reliably send one letter per transmission, which contains around 4.7 bits of information.*

Example 9. *Consider now the noisy typewriter with $\epsilon = 0.5$. How many bits can we reliably send per transmission? Observe that one way to reliably send information is by agreeing to only send letters at even positions in the alphabetic order. In that way, if you receive, for example, `a` or `b`, you know for sure that the sender input `a` to the channel. However, by sending only the "even" letters, you need to double the efforts with respect to the typewriter channel without noise. As a consequence, the noisy typewriter has less capacity. One can actually show that $\max_{p(\cdot)} I(S; \hat{S}) = -1 + \log 26$, where a maximizing $p(\cdot)$ is the uniform distribution over the "even" letters.*

3.4.3 Codes

Intuition on codes and rates

Why aren't we taking into account loss bits or duplicated bits? This can happen during transmission.

Consider the noisy binary channel. By sending several bits in a specific pattern to the channel, one can come up with sophisticated ways to transmit complex information through the channel, like images or spreadsheets. We illustrate this by showing how to use the binary channel to send letters in $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$. The sender and the receiver must first agree on a *code* for those letters. One such code, which we call *naïve code*, encodes the letter `a` as the *codeword* `00000`, `b` as `00001`, and so on. That is, the codeword for the i-th alphabet letter is number i in base 2, written as a bit string of length 5. In a similar fashion, we can conceive codes for communicating more complex data like images and spreadsheets.

Unfortunately, the code mentioned in the previous paragraph is sensitive to the channel's noise. If we send the codeword for `a`, the receiver may get the codeword for `b` with probability $\epsilon\,(1-\epsilon)^4$, yielding a *communication error*. Information theory has came up with smarter codes that reduce the probability of such a communication error, but at the cost of longer codewords. For example, we can use a code, which we call *the 5-redundant code*. This code encodes `a` as `00000` and `b` as `11111`. The receiver would then take the received codeword $\hat{w}$ and search for the codeword w that closest codeword with respect to the Hamming distance. The receiver would then assume that the sender sent the letter associated to w. For example, if the receiver gets `11010`, then she assumes that the sender sent `11111`, which is the codeword for `b`. This code is more robust to noise than the naïve alphabet code. In comparison with the naïve code, more bits need to be flipped by noise in order to get a communication error. This is less likely than having one bit flipped in the naïve code's codewords.

Unfortunately, the robustness comes at the price of less messages. If we use codewords of length n only. Using the naïve code, the sender can communicate 2^n different messages. However, using the n-redundant code, the sender can communicate only 2. Assuming that the noise does not cause a communication error, we manage to transmit at a *rate* of one bit per transmission in the naïve code and one bit per n transmissions in the n-redundant code. In the limit, as $n \to \infty$, the naïve code attains a rate of 1 bit per transmission, but a probability of a communication error equal to 1. On the other hand, the n-redundant code attains a rate of 0 bits per transmission, but a probability of a communication error equal to 0. This comparison is summarized in Figure 3.7

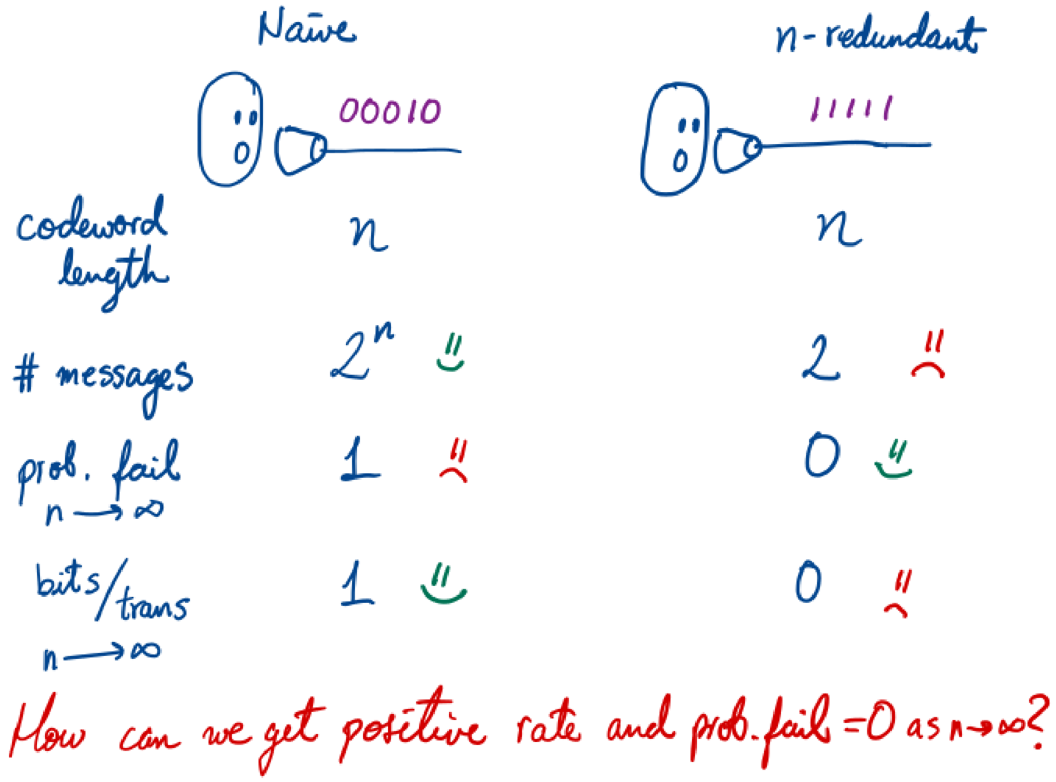
Intuition on Shannon's channel coding theorem

We started by using the binary noisy channel to send bits and we are now devising codes to send a finite set of letters through the binary noisy channel. In a similar manner, we can devise codes to send words through the binary noisy channel, by creating a code that maps words to bit strings of fixed length.Why are we talking only of finite sentences S? In practice, I want to send sentences of variable lengths. Also, why don't we allow encodings of different lengths?

We can continue in this way to create codes to send images of a fixed size, videos of a fixed length, and so on.Why the obsession with fixed size? All this is done by using longer codewords.

Shannon's coding theorem concerns with the problem of finding sustainable strategies for building codes. By sustainable, we mean that the strategy should yield codes that fulfill two conditions.

- First, the codes attain and maintain a positive rate r of bits of information per transmission as $n \to \infty$.
- Second, the probability of a communication error goes to zero as $n \to \infty$.



	Naïve	n-redundant
codeword length	n	n
# messages	2^n ☺	2 ☹
prob. fail $n \to \infty$	1 ☹	0 ☺
bits/trans $n \to \infty$	1 ☺	0 ☹

How can we get positive rate and prob. fail $= 0$ as $n \to \infty$?

Figure 3.7

Shannon's channel coding theorem states that there is a sustainable strategy for building codes, as long as the targeted rate r is below the channel's capacity. For a fixed codeword length n, the maximum number of messages that can be communicated with this strategy is $\lfloor 2^{nr} \rfloor$.

Formalization

For the definitions below, let $M, n \in \mathbb{N}$ and let $(\mathfrak{A}, \mathcal{P})$ be a channel.

Definition 10. *An* (M, n)-code *is a pair* (Enc, Dec) *of functions with* $Enc : \{1, 2, \ldots, M\} \to \mathfrak{A}^n$ *and* $Dec : \mathfrak{A}^n \to \{1, 2, \ldots, M\}$*.*

Definition 11. *The* rate of a (M, n)-code *is*

$$\frac{\log M}{n}.$$

Observe that if a (M, n)-code has rate r, then $M = 2^{nr}$.

Definition 12. *For an* (M, n)*-code* (Enc, Dec)*, its* probability of a communication error *is*

$$\frac{1}{M} \sum_{i \leq M} \mathbf{P}\left(Dec(\hat{W}) \neq i \mid W = Enc\,(i)\right),$$

where $\mathbf{P}\left(Dec(\hat{W}) \neq i \mid W = Enc\,(i)\right)$ *is the probability that the receiver decodes something different to* i*, given that we sent* $Enc\,(i)$ *through the channel.*

Intuitively, the probability of a communication error is the probability that the receives decodes a wrong message when we send him the codeword of a message chosen uniformly at random.

Example 10. *The* n*-redundant code discussed above is an example of a* $(2^n, n)$*-code, whose rate is* $\log 2^n / n = 1$*.*

Definition 13. *A rate* r *is* attainable *if there is a sequence of* $(\lfloor 2^{nr} \rfloor, n)$*-codes, indexed by* n*, such that the probability of a communication error goes to zero as* $n \to \infty$*.*

3.4.4 Shannon's coding theorem

Typicality

We now recall some important notions from Thomas and Cover [**?**]

Theorem 14. *(The asymptotic equipartition property) Let $S, S_1, S_2, \ldots, S_n$ be identically and independently distributed random variables with distribution $p(\cdot)$ over a space $\mathcal{S}$, then*

$$-\frac{1}{n}p(S_1, \ldots, S_n) \to H(S), \quad \textit{in probability as } n \to \infty.$$

This theorem follows from the weak law of large numbers. In our context, $S, S_1, S_2, \ldots, S_n$ denote symbols from a channel's alphabet and $p(\cdot)$ is $\arg\max_p I(S; \hat{S})$. That is, $p(\cdot)$ is the distribution that achieves the channel's capacity.

Definition 15. *For $n \in \mathbb{N}$ and $\epsilon > 0$, the* typical set $A_\epsilon^{(n)}$ *with respect to $p(\cdot)$ is the set of sequences $(s_1, \ldots, s_n) \in \mathcal{S}^n$ such that*

$$H(S) - \epsilon \leq -\frac{1}{n}\log p(s_1, s_2, \ldots, s_n) \leq H(S) + \epsilon.$$

A sequence in $A_\epsilon^{(n)}$ is called a typical sequence*.*

In our context, codewords will consist of typical sequences. The next theorem justifies the following intuitions:

1. All typical sequences have approximately the same probability $\approx 2^{-nH(S)}$.
2. If you draw a sequence in $\mathcal{S}^n$, using $p(\cdot)$, then the resulting sequence is typical with high probability.
3. $\left|A_\epsilon^{(n)}\right| \approx 2^{nH(S)}$.

Theorem 16.

1. *If $(s_1, s_2, \ldots, s_n) \in A_\epsilon^{(n)}$, then $2^{-n(H(S)+\epsilon)} \leq p(s_1, \ldots, s_n) \leq 2^{-n(H(S)-\epsilon)}$.*
2. *$\mathbf{P}\left(A_\epsilon^{(n)}\right) > 1 - \epsilon$, for sufficiently large n.*
3. *$(1-\epsilon)2^{n(H(S)-\epsilon)} \leq \left|A_\epsilon^{(n)}\right| \leq 2^{n(H(S)+\epsilon)}$.*

The reader can take it as an exercise to proof these claims. The proofs are in Thomas and Cover [**?**].

Definition 17. *Let $n \in \mathbb{N}$ and let $p_{S\hat{S}}(\cdot, \cdot)$ be the joint distribution of two random variables S and $\hat{S}$, whose ranges are $\mathcal{S}$ and $\hat{\mathcal{S}}$, respectively. The set $A_\epsilon^{(n)}$ of* jointly typical sequences *with respect to $p_{S\hat{S}}$ is the set of pairs $(\mathbf{s}^n, \hat{\mathbf{s}}^n)$ of sequences that fulfill the following:*

1. $\left|-\frac{1}{n}\log p_{S^n}(\mathbf{s}^n) - H(S)\right| < \epsilon.$

2. $\left|-\frac{1}{n}\log p_{\hat{S}^n}(\hat{\mathbf{s}}^n) - H(\hat{S})\right| < \epsilon.$

3. $\left|-\frac{1}{n}\log p_{S^n\hat{S}^n}(\mathbf{s}^n, \hat{\mathbf{s}}^n) - H(S, \hat{S})\right| < \epsilon.$

A pair in $A_\epsilon^{(n)}$ *is called a* jointly typical pair of sequences*.*

We clarify that, for $\mathbf{s}^n = (s_1, \ldots, s_n)$ and $\hat{\mathbf{s}}^n = (\hat{s}_1, \ldots, \hat{s}_n)$,

$$p_{S^n}(\mathbf{s}^n) = \prod_{i\leq n} p_S(s_i),$$

$$p_{\hat{S}^n}(\hat{\mathbf{s}}^n) = \prod_{i\leq n} p_{\hat{S}}(\hat{s}_i) = \prod_{i\leq n}\sum_s p_S(s)p_{\hat{S}|S}\left(\hat{s}_i \mid s\right), \text{and}$$

$$p_{S^n\hat{S}^n}(\mathbf{s}^n, \hat{\mathbf{s}}^n) = \prod_{i\leq n} p_{S\hat{S}}(s_i, \hat{s}_i) = \prod_{i\leq n} p_S(s_i)p_{\hat{S}|S}(\hat{s}_i \mid s_i).$$

In the context of communication via a channel, $p_{S^n\hat{S}^n}$ represents the joint distribution of $\mathbf{S}^n$ and $\hat{\mathbf{S}}^n$, where

- $\mathbf{S}^n$ denotes a random codeword, where each symbol was chosen at random according to the distribution $p_S = \arg\max_p I(S; \hat{S})$ that achieves channel capacity.

- $\hat{\mathbf{S}}^n$ denotes a codeword that the channel would output, after we send $\mathbf{S}^n$ as input.

We call $p_{S^n\hat{S}^n}$ the *codeword input-output distribution*.

In the context of communication via a channel, the following theorem justifies the following intuitions:

1. Suppose that we build a codeword $\mathbf{s}^n$ at random by choosing each of its symbols at random according to p_S, the distribution that attains channel capacity. Then we send $\mathbf{s}^n$ through the channel and let $\hat{\mathbf{s}}^n$ be the output codeword. Then $(\mathbf{s}^n, \hat{\mathbf{s}}^n)$ is jointly typical with high probability.

2. Suppose now that we build another codeword $\mathbf{q}^n$ at random using the same procedure. Then it is very *unlikely* that $(\mathbf{q}^n, \mathbf{y}^n)$ is jointly typical.

Theorem 18.

1. $\mathbf{P}\left(\left(\mathbf{S}^n, \hat{\mathbf{S}}^n\right) \in A_\epsilon^{(n)}\right) \to 1$, *as* $n \to \infty$.

2. *If* $\mathbf{Q}^n \sim p_{S^n}(\cdot)$ *and* $\hat{\mathbf{S}}^n \sim p_{\hat{S}^n}(\cdot)$ *(i.e., they are drawn independently at random from the marginal distributions of* $p_{S^n\hat{S}^n}$*), then*

$$(1-\epsilon)2^{-n\left(I(S;\hat{S})+3\epsilon\right)} \leq \mathbf{P}\left(\left(\mathbf{Q}^n, \hat{\mathbf{S}}^n\right) \in A_\epsilon^{(n)}\right) \leq 2^{-n\left(I(S;\hat{S})-3\epsilon\right)}$$

These two intuitions justify the effectiveness of a very simple code, called *Shannon's random code.*

Shannon's random code

Theorem 19. *A rate is attainable iff it is below the channel's capacity.*

We only focus here on proving the following direction: if a rate is below the channel's capacity, then it is attainable, as it illustrates how to propose a $(\lfloor 2^{nr}\rfloor, n)$-code for communicating $\lfloor 2^{nr}\rfloor$ messages.

To prove this, we present, for $n > 1$, a $(\lfloor 2^{nr}\rfloor, n)$-code whose probability of a communication error is at most $p_n = 2^{-n(cap-3\epsilon-r)}$, where ϵ is chosen to be sufficiently small. Hence, if $r < cap$, we get that the probability of a communication error goes to zero as $n \to \infty$.

The code's encoder function *Enc* is defined as follows. For a message $m \leq \lfloor 2^{nr}\rfloor$, we define $Enc(m)$ as a string in $\mathbf{s}^n$ where each symbol was drawn from a distribution $p^* = \arg\max_{p(\cdot)} I(S;\hat{S})$. That is, a distribution that maximizes the channel's input-output mutual information and attains the channel's capacity.

The code's decoder function *Dec* is defined as follows. Given the string $\mathbf{s}^n$ output by the channel, *Dec* goes through each message m and tests if $(Enc(m), \mathbf{s}^n)$ is jointly typical with respect to the codeword input-output distribution $p_{S^n\hat{S}^n}$. *Dec* outputs the first message for which this test succeeds. If no message succeeds on the test, then *Dec* outputs an arbitrary message.

Remind the reader that the sender and the receiver agree in advance on *Enc* and *Dec*. Therefore, the receiver also knows both *Enc* and the set $\{1, 2, \ldots, \lfloor 2^{nr}\rfloor$ of messages.

Theorem 20. *The probability* $\mathbf{P}(\mathcal{E})$ *of a communication error for Shannon's random code goes to 0 as* $n \to \infty$.

Proof. Let $\mathcal{K}$ be a random variable representing a possible code. Then

$$\mathbf{P}(\mathcal{E}) = \sum_{\mathcal{K}} \mathbf{P}(\mathcal{K})\, P_e(\mathcal{K}),$$

where $P_e(\mathcal{K})$ is the probability of a communication error for code $\mathcal{K}$. Observe now that

$$\begin{aligned}\mathbf{P}\left(\mathcal{E}\right) &= \sum_{\mathcal{K}} \mathbf{P}\left(\mathcal{K}\right) P_e(\mathcal{K}) \\ &= \sum_{\mathcal{K}} \mathbf{P}\left(\mathcal{K}\right) \frac{1}{\lfloor 2^{nr} \rfloor} \sum_{w \leq \lfloor 2^{nr} \rfloor} \mathbf{P}\left(Dec(\hat{\mathbf{S}}^n) \neq w \mid \mathbf{S}^n = Enc(w)\right) \\ &= \frac{1}{\lfloor 2^{nr} \rfloor} \sum_{\mathcal{K}} \sum_{w \leq \lfloor 2^{nr} \rfloor} \mathbf{P}\left(\mathcal{K}\right) \mathbf{P}\left(Dec(\hat{\mathbf{S}}^n) \neq w \mid \mathbf{S}^n = Enc(w)\right).\end{aligned}$$

Observe now that all codewords were chosen independently at random, so

$$\mathbf{P}\left(Dec(\hat{\mathbf{S}}^n) \neq w \mid \mathbf{S}^n = Enc(w)\right) = \mathbf{P}\left(Dec(\hat{\mathbf{S}}^n) \neq 1 \mid \mathbf{S}^n = Enc(1)\right),$$

for $w > 1$. Hence,

$$\begin{aligned}\mathbf{P}\left(\mathcal{E}\right) &= \frac{1}{\lfloor 2^{nr} \rfloor} \sum_{\mathcal{K}} \sum_{w \leq \lfloor 2^{nr} \rfloor} \mathbf{P}\left(\mathcal{K}\right) \mathbf{P}\left(Dec(\hat{\mathbf{S}}^n) \neq w \mid \mathbf{S}^n = Enc(w)\right) \\ &= \sum_{\mathcal{K}} \mathbf{P}\left(\mathcal{K}\right) \mathbf{P}\left(Dec(\hat{\mathbf{S}}^n) \neq 1 \mid \mathbf{S}^n = Enc(1)\right) \\ &= \mathbf{P}\left(\mathcal{E} \mid w = 1\right).\end{aligned}$$

This means that the probability of a communication error is equal to the probability of a communication error, assuming that the sender sent the codeword for message 1.

Note that, in Shannon's random code, the event of a communication error implies at least one of the following events: the received codeword $\hat{\mathbf{S}}^n(1)$ is not jointly typical with the codeword $\mathbf{S}^n(1)$ for message 1 or the received codeword $\hat{\mathbf{S}}^n(1)$ is jointly typical with the codeword $\mathbf{S}^n(w)$ for a message $w > 1$. More precisely,

$$\mathbf{P}\left(\mathcal{E} \mid M = 1\right) = \mathbf{P}\left(\begin{array}{c} \left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)} \text{ or} \\ \left(\mathbf{S}^n(2), \hat{\mathbf{S}}^n(1)\right) \in A_\epsilon^{(n)} \text{ or} \\ \left(\mathbf{S}^n(3), \hat{\mathbf{S}}^n(1)\right) \in A_\epsilon^{(n)} \text{ or} \\ \vdots \\ \left(\mathbf{S}^n(\lfloor 2^{nr} \rfloor), \hat{\mathbf{S}}^n(1)\right) \in A_\epsilon^{(n)}. \end{array}\right).$$

By the union bound,

$$\mathbf{P}\left(\mathcal{E} \mid M=1\right) \leq \mathbf{P}\left(\left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)}\right) + \sum_{w>1} \mathbf{P}\left(\left(\mathbf{S}^n(w), \hat{\mathbf{S}}^n(1)\right) \in A_\epsilon^{(n)}\right).$$

We now apply Theorem 18, which implies the following:

- $\mathbf{P}\left(\left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)}\right) \to 1$, as $n \to \infty$. In other words, $\mathbf{P}\left(\left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)}\right) \to 0$, as $n \to \infty$.
- $\mathbf{P}\left(\left(\mathbf{S}^n(w), \hat{\mathbf{S}}^n(1)\right) \in A_\epsilon^{(n)}\right) \leq 2^{-n\left(I(S;\hat{S})-3\epsilon\right)}$. This is because, for $w > 1$, $\mathbf{S}^n(1)$ and $\mathbf{S}^n(w)$ were independently drawn from p_{S^n} and, therefore, $\hat{\mathbf{S}}^n(1)$ and $\mathbf{S}^n(w)$ were independently drawn from $p_{\hat{S}^n}$ and p_{S^n}, respectively.

Using these observations we get that

$$\begin{aligned}
\mathbf{P}\left(\mathcal{E} \mid M=1\right) &\leq \mathbf{P}\left(\left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)}\right) + \sum_{w>1} \mathbf{P}\left(\left(\mathbf{S}^n(w), \hat{\mathbf{S}}^n(1)\right) \in A_\epsilon^{(n)}\right) \\
&\leq \mathbf{P}\left(\left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)}\right) + \sum_{w>1} 2^{-n\left(I(S;\hat{S})-3\epsilon\right)} \\
&= \mathbf{P}\left(\left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)}\right) + \left(\lfloor 2^{nr} \rfloor - 1\right) 2^{-n\left(I(S;\hat{S})-3\epsilon\right)} \\
&\leq \mathbf{P}\left(\left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)}\right) + 2^{nr} 2^{-n\left(I(S;\hat{S})-3\epsilon\right)} \\
&= \mathbf{P}\left(\left(\mathbf{S}^n(1), \hat{\mathbf{S}}^n(1)\right) \notin A_\epsilon^{(n)}\right) + 2^{-n\left(I(S;\hat{S})-r-3\epsilon\right)}.
\end{aligned}$$

Observe that we chose p_S as $\arg\max_p I(S;\hat{S})$, so $I(S;\hat{S}) = \mathit{cap}$, the channel's capacity. So, if the rate r is below the channel's capacity and ϵ is sufficiently small, then $\mathbf{P}\left(\mathcal{E} \mid M=1\right) \to 0$ as $n \to \infty$. □

3.5 Communication protocol for algorithm validation

We now formalize the communication protocol where posterior agreement originates. We first explain an *ideal variant* where we know X's probability distribution p_X. Furthermore, we assume that p_X is from a parameterized family $\mathfrak{P}$ of probability distributions. Afterwards, we explain the *empirical variant* where we do not know anything about p_X and, even worse, we only have *two observations* X' and X'' drawn from p_X. Recall that, in practice, we only have access to observations and have no information of the nature of the distribution where these observations came from.

3.5.1 Assumptions

For our statements to hold, we make the following assumptions.

Exponential solution space: We assume that $\mathcal{C}$ is discrete and that $\log |\mathcal{C}| = \Theta(n)$, where n measures the "size" of an observation from a phenomenon. For example, n measures the number of edges in a graph or the number of datapoints in a clustering instance. Intuitively, we assume that the solution space's size grows exponentially in n.

Probabilistic outputs: We assume that any algorithm for stochastic optimization, when given an input X' from an instance space $\mathcal{X}$, outputs a distribution $p(\cdot \mid X')$ over a discrete solution space $\mathcal{C}$. Every algorithm can be thought to be as such, even when it only outputs a fixed value $c_{X'} \in \mathcal{C}$, when given X' as input. In this case, $p(c \mid X') = 1$ if $c = c_{X'}$ and 0 otherwise.

3.5.2 Ideal variant

Let $\mathcal{A}$ be an algorithm intended for solving Problem 3.3. Posterior agreement originates from a communication protocol between a sender and a receiver. The sender must communicate, using $\mathcal{A}$'s outputs, the nature of a phenomenon to the receiver. The receiver must be able, using the information sent from the sender, to identify the phenomenon. The communication from the sender to the receiver is done through a noisy channel, whose noise is defined by $\mathcal{A}$ and the phenomenon.

Messages Fix $n \in \mathbb{N}$. Present the instance space $\mathcal{X}$ to the sender and the receiver. Then agree draw at random a set $\mathcal{F} \subseteq \mathfrak{P}$ of m phenomena and, for each $p \in \mathcal{F}$, draw an observation X' of size n. Let $\mathcal{M} = \{X'_1, \ldots, X'_m\}$. Agreeing on $\mathcal{F}$ is often not possible in practice, so we explain in the empirical variant how to deal with this. This $\mathcal{M}$ constitutes the set of possible messages that the sender may try to communicate to the receiver.

For the moment, we leave the value of m undefined. We leave for later to figure out what is the maximum value of m that we can use. The choice of m must still ensure that the probability of a communication error goes to 0, as $n \to \infty$.

Code Present algorithm $\mathcal{A}$ to the sender and the receiver. For each $X' \in \mathcal{M}$, use $\mathcal{A}$ to compute $p(\cdot \mid X')$. Define the code $(Enc_{\mathcal{A}}, Dec_{\mathcal{A}})$ as follows:

- $Enc_{\mathcal{A}}$ encodes $X' \in \mathcal{M}$ as the codeword $p(\cdot \mid X')$.
- $Dec_{\mathcal{A}}$ decodes a probability distribution $p(\cdot \mid Y)$ over $\mathcal{C}$ as any $\hat{X} \in \mathcal{M}$ such that $k\left(Y, \hat{X}\right) \geq k\left(Y, X\right)$, for all $X \in \mathcal{M}$. Here,

$$k\left(Y, X'\right) := \sum_{c} p(c \mid Y) p(c \mid X').$$

Channel The channel is the pair $\left(\{p(\cdot \mid X')\}_{X' \in \mathcal{M}}, \mathcal{P}\right)$ where $\mathcal{P}$ is defined by the following probabilistic procedure. Assume that the sender inputs $p(\cdot \mid X')$ to the channel. The channel then replaces X' with a fresh new observation X'' from the same phenomenon where X' comes from. Then it uses the algorithm $\mathcal{A}$ to compute $p(\cdot \mid X'')$ and outputs that to the receiver.

Communication A message X' is selected uniformly at random from $\mathcal{M}$ and without the receiver's knowledge. The sender then sends the codeword $p(\cdot \mid X')$ through the channel. The receiver gets $p(\cdot \mid X'')$ and uses the decoding function to guess which message the sender sent. The receiver succeeds by correctly guessing X'.

Observe how the algorithm and the phenomenon define the channel's noise. Hence, the algorithm can be evaluated by *the capacity* of the resulting channel. This capacity is measured by its maximum attainable rate, which is obtained by figuring out how to maximize the number m of messages that can be used in the protocol, while making the probability of a communication error go to zero as n, the size of the observations, go to infinity. We carry this analysis in the empirical variant.

3.5.3 Empirical variant

The capacity of the channel built above cannot be computed, as we often only have access to a set of observations and not to the phenomena behind them. This means that we cannot compute the set $\mathcal{M}$ of messages, as described in the ideal variant. For this reason, we use an empirical variant, where we assume that we are given at least two observations X' and X'' of a same phenomenon.

Messages (Figure 3.8) We create a set of messages by producing *transformed copies* of X'. We take a set $\mathbb{T}$ of transformations, where each transformation transforms instances into instances. Afterwards, we draw m transformations $\tau_1, \tau_2, \ldots, \tau_m$ from $\mathbb{T}$ uniformly at random. In this way, we can create a set

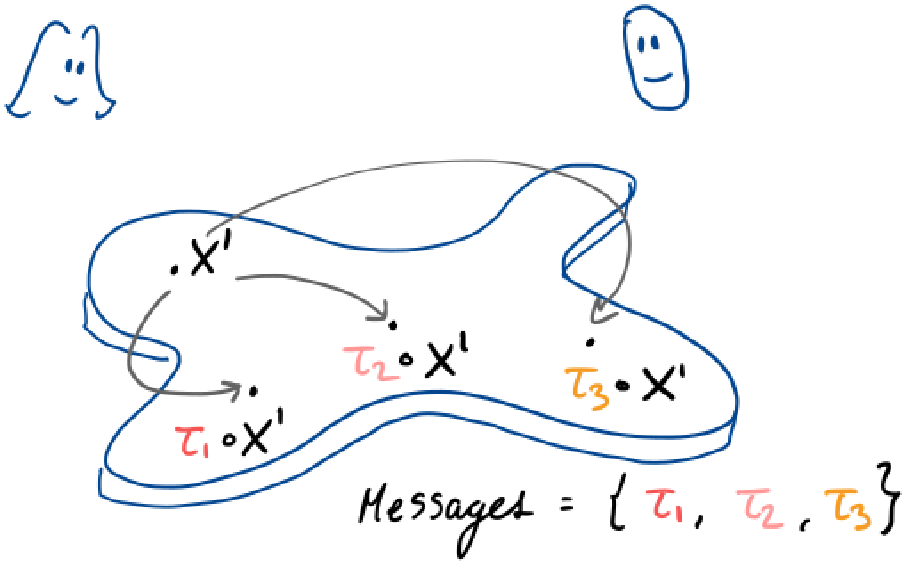


Figure 3.8

$\mathcal{M} = \{\tau_1 \circ X', \tau_2 \circ X', \ldots, \tau_m \circ X'\}$ of messages that are analogous to the set of messages that we created in the ideal variant. We impose the following requirement on $\mathbb{T}$ so that $\mathcal{M}$ looks like the set $\mathcal{M}$ that we would obtain in the ideal variant.

- $\sum_\tau p(c \mid \tau \circ X') \in \left[\frac{|\mathbb{T}|}{|\mathcal{C}|}(1-\rho), \frac{|\mathbb{T}|}{|\mathcal{C}|}(1+\rho)\right]$, for some small $\rho > 0$. The reason for this assumption is that we we do not want the probability mass of all these codewords to be concentrated in a narrow subset of $\mathcal{C}$, as this reduces the number of different messages that the sender can communicate to the receiver. This can be ensured that, for each $c \in \mathcal{C}$, the total probability mass c gets is $\sum_\tau p(c \mid \tau \circ X') \approx \frac{|\mathbb{T}|}{|\mathcal{C}|}$.
- The transformations do not alter an instance's randomness.

Observe that such a set of transformations does not necessarily always exist. For example, if the algorithm under evaluation always produces the same constant distribution $p(\cdot \mid X')$, independent of X', then no set of transformations will be able to achieve the requirement above. In this case, however, there is no need to build a channel to evaluate such an algorithm, as it is clear that the algorithm is not producing any value from the data.

Code (Figure 3.9) The encoding and decoding functions are the analogous of the ideal variant. The codeword of a message $\tau \in \mathcal{M}$ is $p(\cdot \mid \tau \circ X')$. When

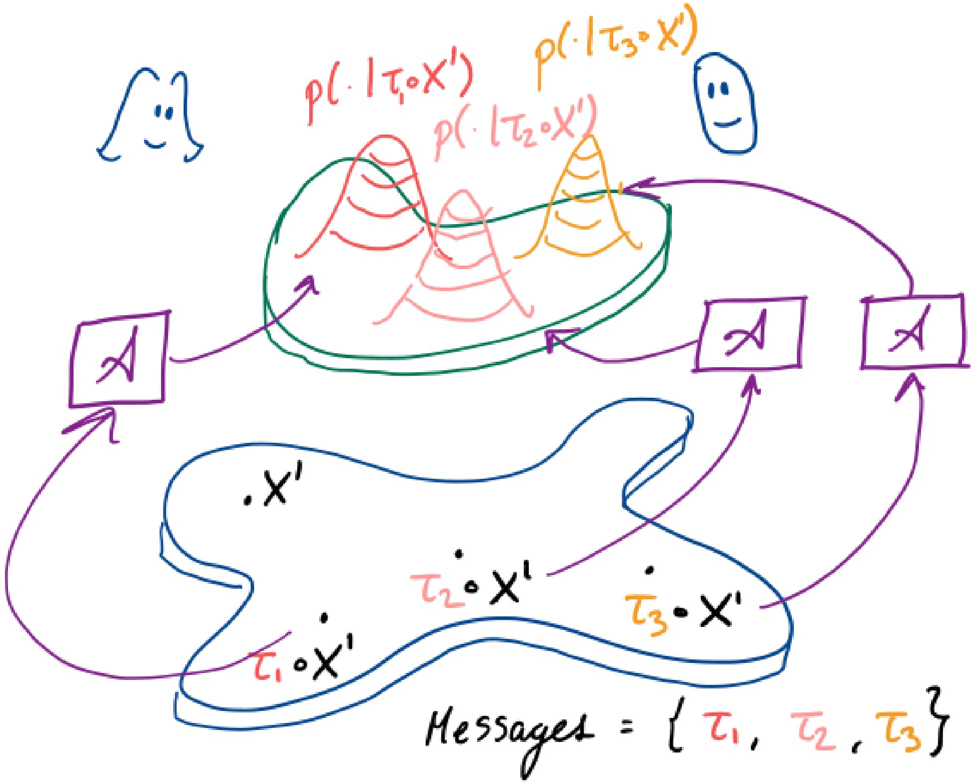

$p(\cdot \mid \tau_1 \circ X')$
$p(\cdot \mid \tau_3 \circ X')$
$p(\cdot \mid \tau_2 \circ X')$
$\mathcal{A}$
$\mathcal{A}$
$\mathcal{A}$
$.X'$
$\tau_2 \circ X'$
$\tau_3 \circ X'$
$\tau_1 \circ X'$
Messages = $\{\tau_1, \tau_2, \tau_3\}$

Figure 3.9

given a codeword $p(\cdot \mid Y)$, the decoding function outputs the message τ for which $k\,(Y, \tau \circ X') \geq k\,(Y, \sigma \circ X')$, for every $\sigma \in \mathcal{M}$.

Channel (Figure 3.10) When the sender inputs $p(\cdot \mid \tau \circ X')$ to the channel, the channel outputs $p(\cdot \mid \tau \circ X'')$, as in the ideal variant.

Protocol The sender and the receiver agree on a set $\mathbb{T}$ of transformations and use the algorithm under evaluation $\mathcal{A}$ to compute the code's encoding and decoding functions. A set $\mathcal{M} = \{\tau_1, \ldots, \tau_m\}$ of m messages is drawn uniformly at random from $\mathbb{T}$. A message $\tau \in \mathcal{M}$ is selected uniformly at random and without the receiver's knowledge. The sender then sends the codeword $p(\cdot \mid \tau \circ X')$ through the channel. The receiver gets $p(\cdot \mid \tau \circ X'')$ and uses the decoding function to guess which message the sender sent (Figure 3.6). The receiver succeeds by correctly guessing τ.

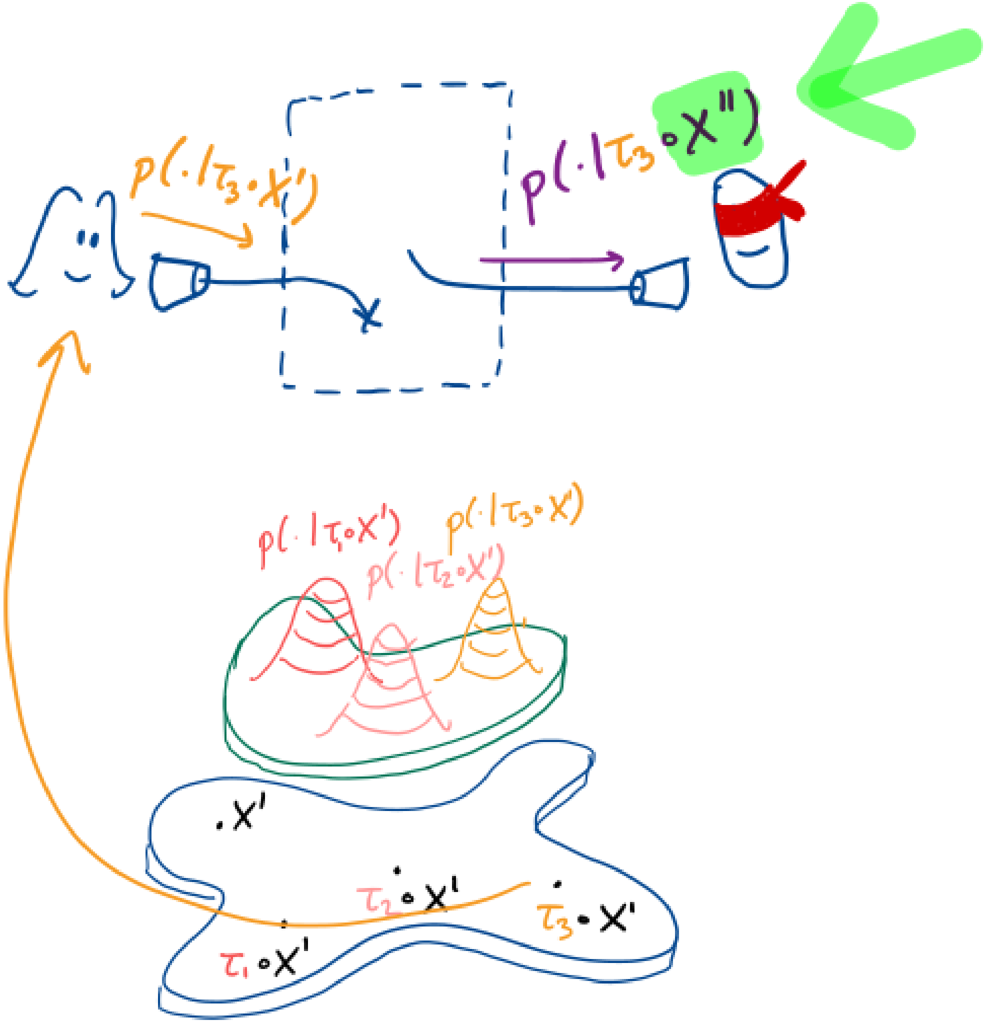

$p(\cdot|\tau_3\circ X')$
$p(\cdot|\tau_3\circ X'')$
$p(\cdot|\tau_1\circ X')$
$p(\cdot|\tau_3\circ X')$
$p(\cdot|\tau_2\circ X')$
$\cdot X'$
$\tau_2\circ X'$
$\tau_3\circ X'$
$\tau_1\circ X'$

Figure 3.10

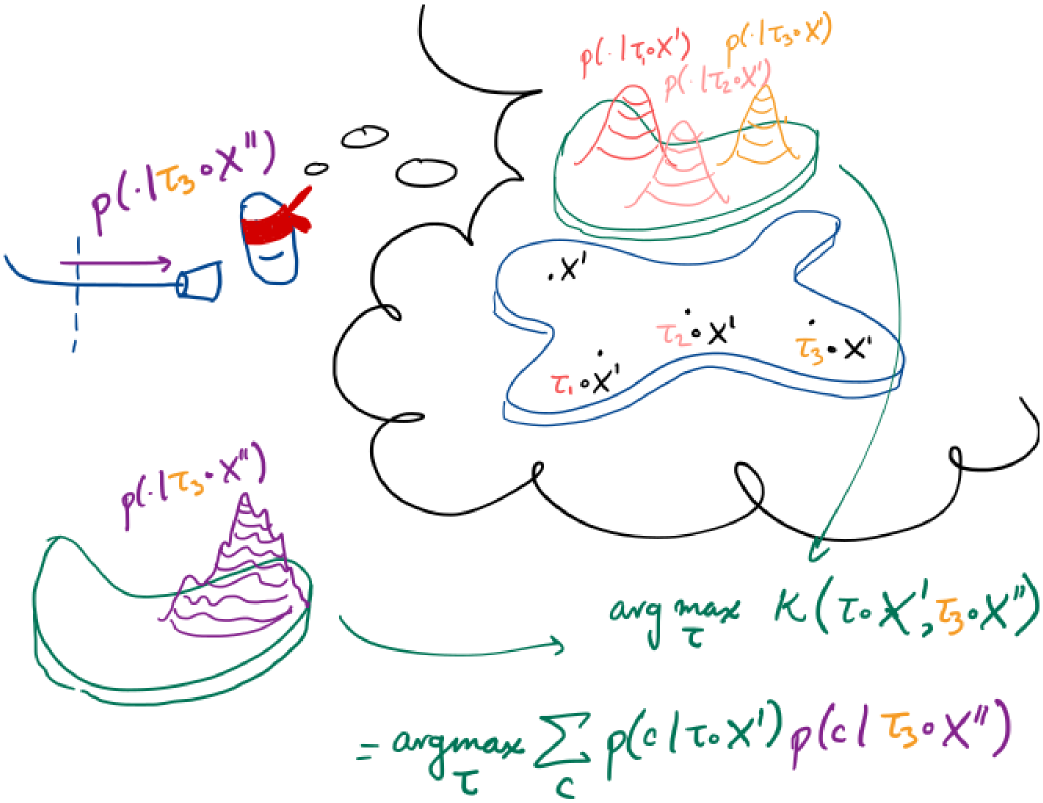

$p(\cdot|\tau_1\circ X')$
$p(\cdot|\tau_3\circ X')$
$p(\cdot|\tau_2\circ X')$
$p(\cdot|\tau_3\circ X'')$
$\cdot X'$
$\tau_2\circ X'$
$\tau_3\circ X'$
$\tau_1\circ X'$
$p(\cdot|\tau_3\circ X'')$
$\arg\max_{\tau} K(\tau\circ X', \tau_3\circ X'')$
$= \arg\max_{\tau} \sum_{c} p(c|\tau\circ X')\, p(c|\tau_3\circ X'')$

Figure 3.11

3.5.4 Probability of a communication error

Theorem 21. *The probability of a communication error is bounded above by*

$$P_{(n)} = \exp\left(-I + \log m + \epsilon \log |\mathcal{C}|\right), \tag{3.5}$$

where

$$I := \mathbb{E}_{X',X''}\left[\log\left(|\mathcal{C}|\, k(X',X'')\right)\right] \tag{3.6}$$

One can show that $P_{(n)} \to 0$ as $n \to \infty$ by choosing ϵ sufficiently small and ensuring that $I - \log m = \Omega(n)$.

Proof. Let τ_s be the message sent by the sender and let $\hat{\tau}$ be the message guessed by the receiver. Just as in the proof of Theorem 20, one can show that $\mathbf{P}\left(\hat{\tau} \neq \tau_s\right) = \mathbf{P}\left(\hat{\tau} \neq id\right)$, where id is some arbitrary transformation. Without loss of generality, we assume that id is the identity transformation.

Using the definition of communication error, we get that

$$\mathbf{P}\left(\hat{\tau} \neq id\right) = \mathbf{P}\left(\max_{\tau \neq id} \kappa\left(\tau \circ X', X''\right) \geq \kappa\left(X', X''\right) \mid id\right).$$

Applying the union bound, we get

$$\mathbf{P}\left(\hat{\tau} \neq id\right) \leq \sum_{\tau \neq id} \mathbf{P}\left(\kappa\left(\tau \circ X', X''\right) \geq \kappa\left(X', X''\right) \mid id\right).$$

We now rewrite probabilities as expectations

$$\begin{aligned}
\mathbf{P}\left(\hat{\tau} \neq id\right) &\leq \sum_{\tau \neq id} \mathbf{P}\left(\kappa\left(\tau \circ X', X''\right) \geq \kappa\left(X', X''\right) \mid id\right) \\
&= \sum_{\tau \neq id} \mathbb{E}_{X',X''}\left[\mathbb{E}_{\tau}\left[\mathbb{I}\left\{\kappa\left(\tau \circ X', X''\right) \geq \kappa\left(X', X''\right)\right\} \mid id, X', X''\right]\right] \\
&= \sum_{\tau \neq id} \mathbb{E}_{X',X''}\left[\mathbf{P}\left(\kappa\left(\tau \circ X', X''\right) \geq \kappa\left(X', X''\right) \mid id, X', X''\right)\right].
\end{aligned}$$

Here, $\mathbb{I}$ is the indicator function.

We now apply Markov's inequality.

$$\begin{aligned}
\mathbf{P}\left(\hat{\tau} \neq id\right) &\leq \sum_{\tau \neq id} \mathbb{E}_{X',X''}\left[\mathbf{P}\left(\mathbb{I}\left\{\kappa\left(\tau \circ X', X''\right) \geq \kappa\left(X', X''\right)\right\} \mid id, X', X''\right)\right] \\
&\leq \sum_{\tau \neq id} \mathbb{E}_{X',X''}\left[\frac{\mathbb{E}_{\tau}\left[\kappa\left(\tau \circ X', X''\right) \mid id, X', X''\right]}{\kappa\left(X', X''\right)}\right].
\end{aligned}$$

We now compute an upper bound for the numerator.

$$\begin{aligned}\mathbb{E}_\tau\left[\kappa\left(\tau\circ X', X''\right) \mid X', X''\right] &= \mathbb{E}_\tau\left[\sum_c p(c\mid\tau\circ X')p(c\mid X'') \mid X', X''\right] \\ &= \sum_c p(c\mid X'')\mathbb{E}_\tau\left[p(c\mid\tau\circ X')\right] \\ &= \sum_c p(c\mid X'')\sum_\tau \frac{1}{|\mathbb{T}|}p(c\mid\tau\circ X') \\ &\leq \frac{1}{|\mathbb{T}|}\frac{|\mathbb{T}|}{|\mathcal{C}|}(1+\rho)\sum_c p(c\mid X'') = (1+\rho)\frac{1}{|\mathcal{C}|}.\end{aligned}$$

For the last inequality, we used the assumption that $\sum_\tau p(c \mid \tau\circ X') \in \left[\frac{|\mathbb{T}|}{|\mathcal{C}|}(1-\rho), \frac{|\mathbb{T}|}{|\mathcal{C}|}(1+\rho)\right]$.

We have then that

$$\begin{aligned}\mathbf{P}\left(\hat{\tau}\neq id\right) &\leq \sum_{\tau\neq id}\mathbb{E}_{X',X''}\left[\frac{1+\rho}{|\mathcal{C}|\,\kappa\left(X',X''\right)}\right] \\ &= (1+\rho)\left(m-1\right)\mathbb{E}_{X',X''}\exp\left(-\log\left(|\mathcal{C}|\,\kappa\left(X',X''\right)\right)\right) \\ &\leq m(1+\rho)\;\mathbb{E}_{X',X''}\exp\left(-\log\left(|\mathcal{C}|\,\kappa\left(X',X''\right)\right)\right) \\ &= (1+\rho)\mathbb{E}_{X',X''}\exp\left(-\log\left(|\mathcal{C}|\,\kappa\left(X',X''\right)\right)+\log m\right) \\ &\approx \;\mathbb{E}_{X',X''}\exp\left(-\hat{I}+\log m\right).\end{aligned}$$

Here, for simplicity, we assumed $1+\rho\approx 1$.

To provide an upper bound to $\mathbf{P}\left(\hat{\tau}\neq id\right)$, we must bound the behavior of the random variable $\hat{I} = \log\left(|\mathcal{C}|\,\kappa\left(X',X''\right)\right)$. To do this, we assume that $\hat{I}$ satisfies *an asymptotic equipartition property* in the sense that, as $n\to\infty$, $\hat{I}\to I$, where $I = \mathbb{E}_{X',X''}\log\left(|\mathcal{C}|\,\kappa\left(X',X''\right)\right)$, the expected log posterior agreement. Under this assumption, for every $\epsilon,\delta > 0$, there is $n_0\in\mathbb{N}$ such that for any $n > n_0$,

$$\mathbf{P}\left(\left|\hat{I}-I\right|\leq\epsilon\log|\mathcal{C}|\right) > 1-\delta.$$

ToDo: Explain why this assumption is reasonable. In particular, explain its connection to typicality.

With this assumption, we can derive the following:

$$\begin{aligned}
\mathbf{P}\left(\hat{\tau} \neq id\right) &\leq \mathbb{E}_{X',X''} \exp\left(-\hat{I} + \log m\right) \\
&= \mathbb{E}_{X',X''} \exp\left(-\hat{I} + (I - I) + \log m\right) \\
&= \mathbb{E}_{X',X''} \exp\left(-I + (I - \hat{I}) + \log m\right) \\
&\leq \mathbb{E}_{X',X''} \exp\left(-I + \left|I - \hat{I}\right| + \log m\right) \\
&\leq \mathbb{E}_{X',X''} \exp\left(-I + \epsilon \log |\mathcal{C}| + \log m\right) \\
&= \exp\left(-I + \epsilon \log |\mathcal{C}| + \log m\right).
\end{aligned}$$

This concludes the proof. □

Recall that the goal is to be able to maximize the number of distinguishable messages that can be sent through the channel. Hence, we must aim to make both m and I as large as possible. The algorithm can only influence I and, therefore, good algorithms *shall maximize the expected log posterior agreement.*

Computing I requires the underlying distribution of X' and X'', which we assume to be unknown. In this case, we can approximate I with *the empirical log posterior agreement*

$$\frac{1}{L} \sum_{\ell \leq L} \left[\log\left(|\mathcal{C}|\, k(X'_\ell, X''_\ell)\right)\right], \tag{3.7}$$

where $\{X'_1, X''_1, \ldots, X'_L, X''_L\}$ is a set of observations.

Finally, we remark some analogies with Shannon's channel coding theorem. The quantity $\frac{1}{n}\mathbb{E}_{X',X''} \log\left(|\mathcal{C}|\, \kappa\left(X', X''\right)\right)$ plays the role the input-output mutual information. The value $\log m/n$ plays the role of the code rate.

Appendix A

The EM-algorithm

A.1 Introduction

We motivate and present the EM (expectation-maximization) algorithm, an algorithm used for approximately computing parameter values for probability distributions in maximum-likelihood estimation. The reader is expected to have knowledge of undergraduate probability theory and to be familiar with maximum-likelihood estimation.

The presentation is divided in three parts. Section A.2 presents an optimization problem regarding a vegan flea. We present an algorithm that finds an approximate solution and provides intuitions to understand why the EM-algorithm works. Section A.3 presents the problem of building a simple movie recommendation system. We show that movie ratings can be understood as samples from a probabilistic model that is defined by a set of multivariate Bernoulli distributions. Estimating the parameters of these distributions via maximum-likelihood turns out to be very hard using analytical methods. In Section A.4, we show that this maximum-likelihood estimation problem is an instance of the vegan-flea optimization problem and derive the EM-algorithm from the approximation algorithm presented in Section A.2.

A.2 The vegan-flea optimization problem

A.2.1 The dog

We introduce a two-dimensional dog, depicted in Figure A.1a. Although, in practice, dogs are three-dimensional entities, a two-dimensional dog makes easier the presentation of some ideas later. Observe that this two-dimensional dog has only two legs, since two legs suffice to keep balance, and one eye, since there is no need for perspectives in a two-dimensional space.

Figure A.1b shows some of the dog's cardiovascular system. Observe that a blood vessel of a two-dimensional being does not have the shape of a cilinder. They still naturally expand when a surge of blood flows from a heart's pump. For the rest of these notes we focus only on a small area of this figure; namely, the tiny green square shown in the figure.

Figure A.1c shows the area marked by the green square in detail. We have placed some Cartesian axes there for reference. There is a flea at coordinates (0.25, 6). The dog's skin is the brown curve, and the upper border of a blood vessel is the red curve. Observe that, in a two-dimensional space, skins are lines instead of surfaces.

A.2.2 The skin and the blood vessel's upper border

We now mathematically model the skin and the blood vessel's upper border. Let $\mathit{skin} : [0,1] \times [0,\infty) \to \mathbb{R}$, and $\mathit{vessel} : [0,1] \times [0,\infty) \to \mathbb{R}$ be two functions. For $t \in [0,\infty)$, let $\mathit{skin}(\cdot,t) : [0,1] \to \mathbb{R}$ be the function that maps x to $\mathit{skin}(x,t)$. We define the function $\mathit{vessel}(\cdot,t)$ analogously. Intuitively, for $t \in [0,\infty)$, the functions $\mathit{skin}(\cdot,t)$ and $\mathit{vessel}(\cdot,t)$ describe the skin and the blood vessel's upper border at time t, as depicted in Figure A.2. Hence, $\mathit{skin}(x,t) \geq \mathit{vessel}(x,t)$, for any $(x,t) \in [0,1] \times [0,\infty)$. Observe that $\mathit{skin}(\cdot,t)$ and $\mathit{vessel}(\cdot,t)$ vary with t. This is to model the fact that blood flows through the vessel and, consequently, makes the skin surface and the blood vessel's upper border vary with time. The animated .gif file attached with these notes[1] illustrate the setting. We strongly encourage the reader to look the .gif file before proceeding.

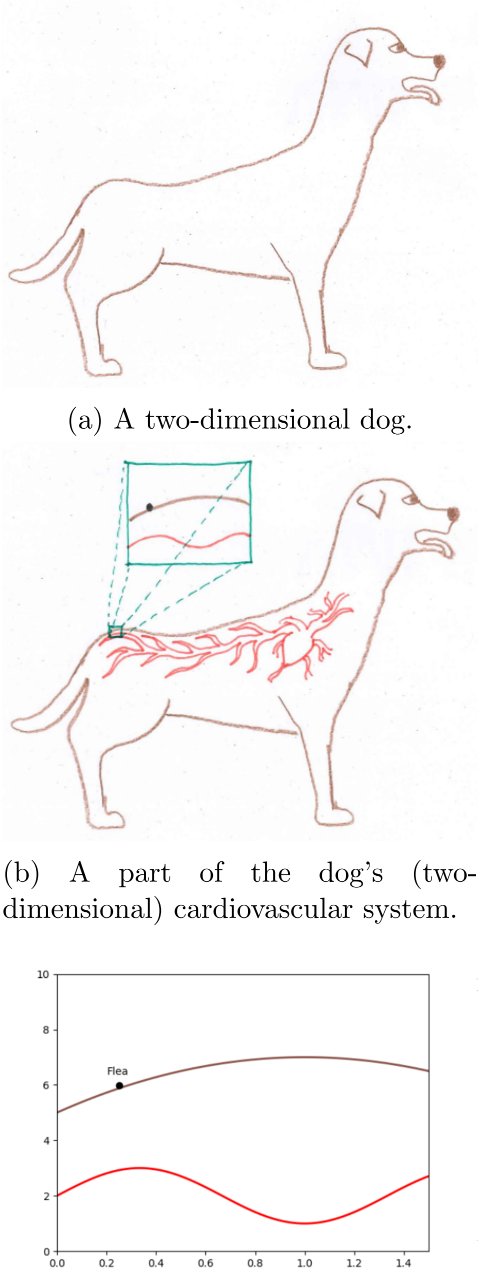
Assumption 1. *We assume that for any $x \in [0,1]$ and any two time points $t_1, t_2 \in [0,\infty)$, $\mathit{skin}(x,t_1) - \mathit{vessel}(x,t_1) = \mathit{skin}(x,t_2) - \mathit{vessel}(x,t_2)$.*

This assumption states that the skin surface changes by the same amount that the blood vessel's upper border changes. This allows us to define a function d such that $d(x) = \mathit{skin}(x,t) - \mathit{vessel}(x,t)$, for any $x \in [0,1]$ and $t \in [0,\infty)$.

Blood flows periodically through the vessel and therefore makes the vessel shape change. Moreover, we assume the following:

Assumption 2. *For any $x \in [0,1]$ and any $t \in [0,\infty)$, there is $t' \geq t$ such that $\mathit{vessel}(x,t') = \max_{x'} \mathit{vessel}(x',t')$.*

[1] `https://people.inf.ethz.ch/ccarlos/assets/em/vegan_flea.gif`

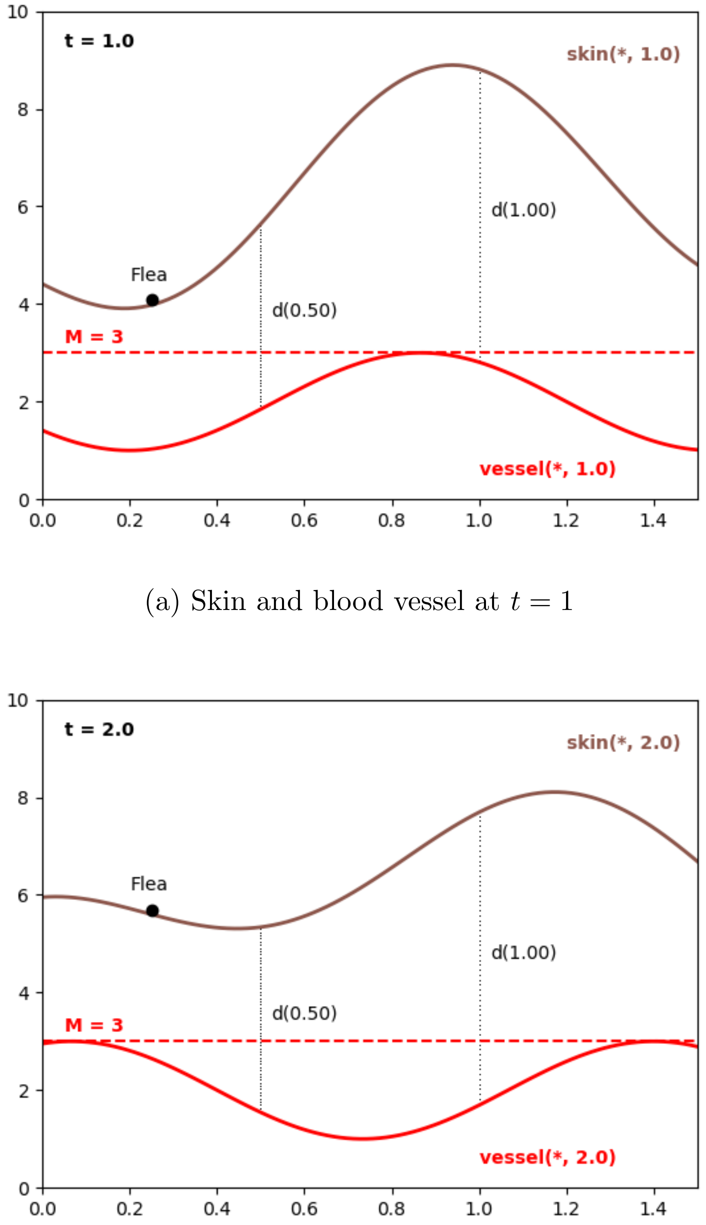


(a) A two-dimensional dog.

(b) A part of the dog's (two-dimensional) cardiovascular system.

(c) The flea, the skin, and the upper border of a blood vessel.

Figure A.1



(a) Skin and blood vessel at $t = 1$

(b) Skin and blood vessel at $t = 2$

Figure A.2: An illustration of the functions $skin(\cdot, t)$ and $vessel(\cdot, t)$, for $t \in \{1, 2\}$.

If you observe the .gif animation, you can see that we defined a constant M. You can also see that, for any x and any t, $\mathit{vessel}(x, t) \leq M$ and that there is $t' \geq t$ such that $\mathit{vessel}(x, t') = M$. We could then make Assumption 2 stronger by stating that, for any $x \in [0, 1]$ and any $t \in [0, \infty)$, $\mathit{vessel}(\cdot, t)$ is bounded by M and that there is $t' \geq t$ such that $\mathit{vessel}(x, t) = M$. However, Assumption 2 is enough for our purposes. Moreover, it is weaker and, hence, more general.

A.2.3 The vegan flea

Imagine now that there is a flea resting on the skin surface at $(x_0, \mathit{skin}(x_0, 0))$, for some $x_0 \in [0, 1]$. The flea has decided to become vegan and wishes to be as far away from the blood vessel as possible, to avoid the temptation of the blood. More precisely, the flea's goal is the following.

Objective 1. Compute a value x^* that maximizes d.

A look at the .gif file shows that $x^* = 1.0$. This is easy for us as we, three-dimensional creatures, have an omniscient view of the flea's universe. The flea, however, cannot see that as she knows nothing about *vessel*. In spite of this, we illustrate how the flea can partially achieve its objective.

We make two assumptions about the flea's computation abilities.

Assumption 3. *For any $t \in [0, \infty)$, the flea can efficiently compute*

$$x^* = \arg\max_{x'} \mathit{skin}(x, t).$$

This assumption bases on the idea that the flea can see the dog's skin and can therefore maximize $\mathit{skin}(\cdot, t)$. The next assumption states that the flea, located at $(x_0, (\mathit{skin}(x_0, 0)))$, can identify the moment t' when $\mathit{vessel}(x_0, t') = M$, the maximum of $\mathit{vessel}(\cdot, t)$.

Assumption 4. *For any $x \in [0, 1]$ and any $t \in [0, \infty)$, the flea can efficiently compute some $\hat{t} \geq t$ such that $\mathit{vessel}(x, \hat{t}) = \max_{x'} \mathit{vessel}(x', \hat{t})$.*

We give some justification for this assumption. Blood flows through the vessel in a periodic way and $\mathit{skin}(\cdot, t)$ changes in the same way as $\mathit{vessel}(\cdot, t)$ does. Hence, the flea can learn the blood pulse and then wait for a time $\hat{t}$ where $\mathit{vessel}(x_0, \hat{t}) = M$.

A.2.4 An approximate maximization algorithm

We describe a strategy by which the flea can compute a value x^* such that $d(x^*) \geq d(x_0)$, where $(x_0, \mathit{skin}(x_0, 0))$ is the flea's current position.

[**E-step**] The flea waits for a time $\hat{t}$ at which $\mathit{vessel}(x_0, \hat{t}) = \max_x \mathit{vessel}(x, \hat{t})$ (Figure A.3a). This is possible by Assumption 4.

[**M-step**] At time $\hat{t}$, the flea computes a value x^* such that

$$x^* = \arg\max_x \mathit{skin}(x, \hat{t}).$$

(Figure A.3b). This is possible by Assumption 3.

[**Move**] The flea moves to $(x^*, \mathit{skin}(x^*, \hat{t}))$ (Figure A.3c).

Figure A.4 illustrates why $d(x^*) \geq d(x_0)$. Observe that $\mathit{skin}(x^*, \hat{t}) \geq \mathit{skin}(x_0, \hat{t})$, since x^* is a maximum of $\mathit{skin}(\cdot, \hat{t})$. Observe also that $\mathit{vessel}(x^*, \hat{t}) \leq \mathit{vessel}(x_0, \hat{t})$, since $\mathit{vessel}(x_0, \hat{t})$ is a maximum of $\mathit{vessel}(\cdot, \hat{t})$. Hence, $d(x^*) = \mathit{skin}(x^*, \hat{t}) - \mathit{vessel}(x^*, \hat{t}) \geq \mathit{skin}(x_0, \hat{t}) - \mathit{vessel}(x_0, \hat{t}) = d(x_0)$.

Notice that the flea can set $x_0 = x^*$ and repeat this procedure to find another value x^{**} such that $d(x^{**}) \geq d(x^*)$. The flea can repeat this process as long as the computed values increase d.

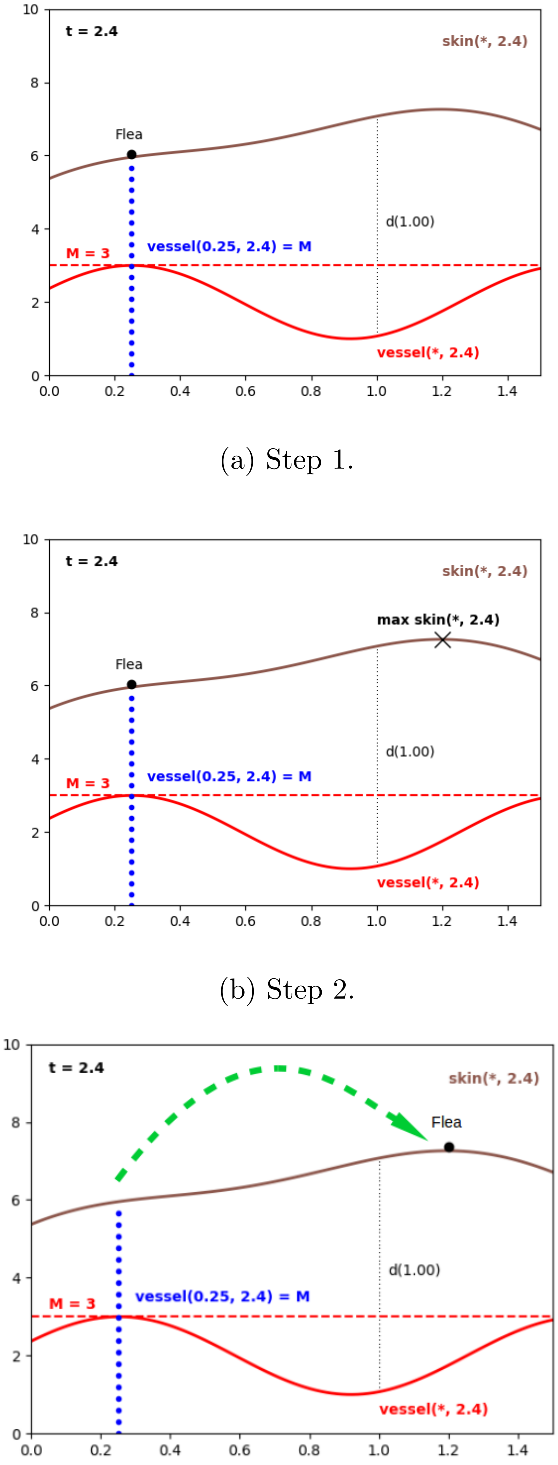
We summarize these insights into Algorithm 2, which computes a sequence of values x_0, $x_1, \ldots$ such that $d(x_0) \leq d(x_1) \leq \ldots$ Observe that this algorithm converges.

One can also relax the assumptions above so that the algorithm works even when *vessel* and *skin*'s domains are a product $\mathcal{X} \times \mathcal{T}$ of any two sets $\mathcal{X}$ and $\mathcal{T}$.

With an argumentation similar to the one above, one can prove that $d(x^*)$ never decreases between two iterations of Algorithm 2's loop.

A.3 Building a movie-recommendation system with a mixture of multivariate Bernoulli distributions

Table A.1 shows the ratings that 10 (fictitious) individuals gave to 6 popular movies. To keep it simple, we assume only binary ratings (good or bad). After a close look to the ratings, the reader can see that Alice and Bob have the exact same taste for movies: *sci-fi* movies. The next four people have



(a) Step 1.

(b) Step 2.

(c) Step 3.

Figure A.3

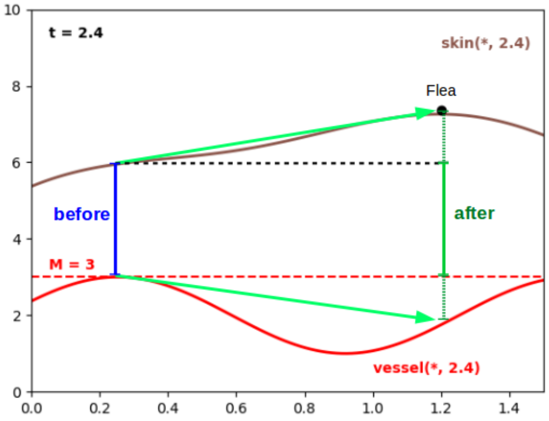


Figure A.4

	Star Wars	Star Trek	Titanic	Pretty Woman	007	Mission Impossible
Alice	✓	✓	×	×	×	×
Bob	✓	✓	×	×	×	×
Carlos	×	×	✓	✓	×	×
David	×	×	✓	✓	×	×
Ellen	×	×	✓	×	×	✓
Fabian	×	×	✓	✓	×	×
Gabriel	✓	✓	×	×	✓	✓
Hector	✓	×	×	×	✓	✓
Ian	✓	✓	×	×	✓	✓
Zelya	✓	×	✓	×	×	✓
John	?	?	?	✓	?	?

Table A.1: Ratings from 10 individuals for 6 movies. According to the table, everyone who likes Pretty Woman also liked Titanic. Therefore, it is likely that John would also like Titanic.

Algorithm 2

```
Require: $\mathcal{X}$ and $\mathcal{T}$ two sets and real functions $d$, $vessel$, and $skin$ satisfying the following.
    A1 $d(x) = skin(x,t) - vessel(x,t)$, for any $x \in \mathcal{X}$ and $t \in \mathcal{T}$.
    A2 For any $x \in \mathcal{X}$, one can efficiently compute $\hat{t} \in \mathcal{T}$ such that $vessel(x,\hat{t}) = \max_{x'} vessel(x',\hat{t})$.
    A3 For any $t \in \mathcal{T}$, one can efficiently compute $\arg\max_x skin(x,t)$.
function DistanceMax($vessel : \mathcal{X} \times \mathcal{T} \to \mathbb{R}$, $skin : \mathcal{X} \times \mathcal{T} \to \mathbb{R}$)
    Choose any $x_0 \in \mathcal{X}$.
    for $i = 0, 1, \ldots$ do
        [E-step] Compute $\hat{t} \in \mathcal{T}$ s.t. $vessel(x_0,\hat{t}) = \max_x vessel(x,\hat{t})$.
        [M-step] Compute $x^* = \arg\max_x skin(x,\hat{t})$.
        Print $x^*$ and $d(x^*)$.
        $x_0 \leftarrow x^*$.
    end for
end function
```

a strong interest for *romantic* movies. The next three people like sci-fi and action movies. Zelya's tastes seem to be different from everyone else.

Consider now John, he likes Pretty Woman, but does not like Star trek. He has not seen any of the other movies. What movie could we recommend to him? Since he likes Pretty Woman and everyone who liked Pretty Woman also liked Titanic, we can recommend him to watch Titanic.

From all 2^6 combinations of ratings, the table contains only a few of them. Moreover, a large majority of the people in the table seem to belong to one of very few taste categories: sci-fi, romantic, or sci-fi+action. Real life is not so different: a large majority of people can partitioned into very few categories and people within a same category have very similar preferences. To recommend a movie to someone, we estimate the category where this person belongs and then search for a movie that people in this category liked.

A.3.1 Movie ratings as samples from probability distributions

We can view Table A.1 as the result of a sampling process. Initially, the table was empty and then one person at random appeared (Alice, in our case) and filled the first row of the table. Then Bob appeared and so on. To sample the film ratings of one person, we first *sample the category* where this person belongs and then, for each movie, we *sample the rating* this person gave to that movie, conditioned on the person belonging to the sampled category. This sampling process is then defined by the following probability distributions:

- A distribution over categories.
- For each category and each movie, a distribution defining the probability that a person in the category likes the movie.

From these two distributions, we build a new probability distribution with which we can answer the following question: *if a person watched and liked movies m_1, $m_2, \ldots$, and m_k, how likely is that she will like a movie m' that she has not seen?* This probability distribution constitutes then our recommendation system. To decide which movie to recommend, we take the ratings the person has given to previously watched movies. Then, for each movie in the database she has not seen, we compute the probability that she likes that movie. Finally, we recommend the movie that she will most likely like.

We first formalize the two distributions mentioned above. Suppose we have K categories and D movies. We identify categories with the numbers 1, $2, \ldots$, K and movies with the numbers 1, $2, \ldots$, D. We can model the distribution of K categories using a discrete distribution with a set $\{\nu_1,\ \nu_2, \ldots,\ \nu_K\}$ of K parameters that add up to 1. For the k-th category and the movie j, we define a value μ_{kj} indicating what the probability is that a person in the k-th category likes j. We leave the values ν_k and μ_{kj}, for $k \leq K$ and $j \leq D$, undefined for the moment.

Having defined these distributions, we can now assign a probability to the ratings a person gave to all movies in the database. We model these ratings with a vector $x \in \{0,1\}^D$, where x_j, for $j \leq D$, indicates whether the movie was rated good ($x_j = 1$) or bad ($x_j = 0$). We leave as an exercise to show that the probability $p(x)$ of a vector $x \in \{0,1\}^D$ is as follows:

$$p(x) = \sum_{k \leq K} \left(\nu_k \prod_{j \leq D} \mu_{kj}^{x_j} \left(1 - \mu_{kj}\right)^{1-x_j} \right). \tag{A.1}$$

We can now define probabilities for movie-rating databases. We model a movie-rating database with a matrix $X \in \{0,1\}^{N \times D}$. Each row X_i, for $i \leq N$, represents a person and each entry $X_{i,j}$, for $j \leq D$, represents the rating person i gave to movie j. Assuming that the ratings of two different people are independent, we can show that the probability $p(X)$ is given by the following.

$$p(X) = \prod_{i \leq N} p(X_i) = \prod_{i \leq N} \sum_{k \leq K} \left(\nu_k \prod_{j \leq D} \mu_{kj}^{X_{i,j}} \left(1 - \mu_{kj}\right)^{1 - X_{i,j}} \right). \tag{A.2}$$

A.3.2 Maximum-likelihood estimation

We now choose values for ν_k and μ_{kj}, for $k \leq K$ and $j \leq D$. Here, we use *maximum-likelihood estimation*, which argues that the best values for our parameters are those that maximize $p(X)$, for X the movie database we have. For computational reasons, one searches instead for the parameters that maximize $\log p(X)$. Using basic logarithm properties, we can show that

$$\log p(X) = \sum_{i \leq N} \log \left(\sum_{k \leq K} \left(\nu_k \prod_{j \leq D} \mu_{kj}^{x_j} \left(1 - \mu_{kj}\right)^{1 - x_j} \right) \right). \tag{A.3}$$

The value $\log p(X)$ is called *X's log likelihood.*

Finding the parameter values that maximize this log likelihood is difficult, even with approximation methods. Nonetheless, it is possible to find a set of parameter values that locally maximize the log likelihood, using Algorithm 2. To do this, we introduce a new set $\mathbf{Z} = \{\mathbf{Z}(i) \mid i \leq N\}$ of random variables. For $i \leq N$, $\mathbf{Z}(i)$ indicates to which category person i belongs. Assume for a moment that, in addition of X, we also know, for $i \leq N$, the category $Z(i)$ where person i belongs. One can show that (X, Z)'s log likelihood is given by the following.

$$\log p(X, Z) = \sum_{i \leq N} \left(\begin{array}{l} \log \nu_{Z(i)} + \\ x_{i,Z(i)} \log \mu_{Z(i),j} + \\ \left(1 - x_{i,Z(i)}\right) \log \left(1 - \mu_{Z(i),j}\right) \end{array} \right). \tag{A.4}$$

The log likelihood of (X, Z) is much easier to maximize with respect to the parameters than X's log likelihood; it can be maximized using standard calculus. However, observe that the movie database does not tell us to which category each person belongs, so we do not know Z and there is no clear way how to obtain it.

It is common that log likelihood maximization problems become easier when we introduce additional random variables to the probabilistic model. It is also common that the values that such random variables take are not available. For these situations, the EM-algorithm was proposed.

A.4 Derivation of the EM-algorithm

We generalize the problem we addressed in the previous section. Let $\mathbf{X}$ and $\mathbf{Z}$ be random variables and let X be an observed value for $\mathbf{X}$. Assume that the joint pdf $p(\cdot, \cdot \mid \theta)$ for $(\mathbf{X}, \mathbf{Z})$ is parameterized by θ, which can take values in Θ. Our goal is to compute

$$\arg\max_{\theta \in \Theta} \log p(X \mid \theta). \tag{A.5}$$

Assume now that it is preferrable to work with $\log p\,(X, Z)$ than with $\log p(X)$, for any Z in $\mathbf{Z}$'s range. If we knew the value Z that $\mathbf{Z}$ took when we obtained $\mathbf{X} = X$, then we could state that

$$\log p(X \mid \theta) = \log p(X, Z \mid \theta) - \log p(Z \mid X, \theta). \tag{A.6}$$

This identity follows from the definition of conditional pdfs. We can make Z irrelevant by computing expectations on both sides with respect to some pdf $\tilde{p}$ for $\mathbf{Z}$. We leave for later the problem of defining $\tilde{p}$.

$$\begin{aligned}\int \tilde{p}(Z) \log p(X \mid \theta) dZ &= \int \tilde{p}(Z) \log p(X, Z \mid \theta) dZ - \int \tilde{p}(Z) \log p(Z \mid X, \theta) dZ \\ &= \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(X, \mathbf{Z} \mid \theta)\right] - \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(\mathbf{Z} \mid X, \theta)\right].\end{aligned}$$

Observe that $\log p(X \mid \theta)$ does not depend on Z or $\tilde{p}$. Therefore, the left-hand side equals $\log p(X \mid \theta)$. As a result,

$$\log p(X \mid \theta) = \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(X, \mathbf{Z} \mid \theta)\right] - \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(\mathbf{Z} \mid X, \theta)\right]. \tag{A.7}$$

We now show that the maximization problem in Equation A.5 is an instance of the vegan-flea problem. Let $\mathcal{X} = \Theta$ and $\mathcal{T}$ be the set of all pdfs for $\mathbf{Z}$. For $\theta \in \Theta$ and $\tilde{p} \in \mathcal{T}$, let

$$\begin{aligned} d(\theta) &= \log p(X \mid \theta) \\ \mathit{skin}(\theta, \tilde{p}) &= \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(X, \mathbf{Z} \mid \theta)\right] \\ \mathit{vessel}(\theta, \tilde{p}) &= \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(\mathbf{Z} \mid X, \theta)\right]. \end{aligned}$$

We now derive sufficient conditions for the assumptions [**A1**], [**A2**], and [**A3**] to hold.

A1 This assumption follows from Equation A.7, so no condition is necessary.

A2 In our case, this assumption means the following: for any $\theta \in \Theta$, one can efficiently compute $\tilde{p} \in \mathcal{T}$ such that for any $\theta' \in \Theta$,

$$\mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(\mathbf{Z} \mid X, \theta)\right] \geq \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(\mathbf{Z} \mid X, \theta')\right]. \tag{A.8}$$

We can fulfill this inequality by setting $\tilde{p}\left(\mathbf{Z}\right) = p(\mathbf{Z} \mid X, \theta)$. This follows from Gibbs's inequality, which states that for any two pdfs p and q for a random variable $\mathbf{Z}$,

$$\mathbb{E}_{p(\mathbf{Z})}\left[\log p(\mathbf{Z})\right] \geq \mathbb{E}_{p(\mathbf{Z})}\left[\log q(\mathbf{Z})\right]. \tag{A.9}$$

Hence, for **[A2]** to hold, we require the pdf $p(\mathbf{Z} \mid X, \theta)$ *to be efficiently computable.*

A3 This assumption requires that, for any $\tilde{p} \in \mathcal{T}$, we can efficiently compute

$$\arg\max_{\theta \in \Theta} \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(X, \mathbf{Z} \mid \theta)\right].$$

In summary, to apply Algorithm 2 to compute $\arg\max_\theta \log p\left(X \mid \theta\right)$, we require the following.

AE1 One can efficiently compute the pdf $p(\mathbf{Z} \mid X, \theta)$.

AE2 One can efficiently compute $\arg\max_{\theta \in \Theta} \mathbb{E}_{\tilde{p}(\mathbf{Z})}\left[\log p(X, \mathbf{Z} \mid \theta)\right]$, for any pdf $\tilde{p}$ for $\mathbf{Z}$.

Instantiating Algorithm 2 to our particular problem, we obtain *the EM algorithm.*

Algorithm 3

Require:

- Θ a set of parameters.
- A joint pdf $p(\mathbf{X}, \mathbf{Z} \mid \theta)$ over two random variables $\mathbf{X}$ and $\mathbf{Z}$, governed by a parameter θ that ranges over Θ.
- A value X in $\mathbf{X}$'s range.

AE1 One can efficiently compute the pdf $p(\mathbf{Z} \mid X, \theta)$.

AE2 One can efficiently compute $\arg\max_{\theta \in \Theta} \mathbb{E}_{\tilde{p}(\mathbf{Z})} \left[\log p(X, \mathbf{Z} \mid \theta)\right]$, for any pdf $\tilde{p}$ for $\mathbf{Z}$.

```
function EM($X$, $p(\mathbf{X}, \mathbf{Z} \mid \theta)$, $\Theta$)
    Choose any $\theta_0 \in \Theta$.
    for $i = 0, 1, \ldots$ do
        [E-step] Compute $p(\mathbf{Z} \mid X, \theta_i)$.
        [M-step] Compute $\theta_{i+1} = \arg\max_\theta \mathbb{E}_{p(\mathbf{Z} \mid X, \theta_i)} \left[\log p(X, \mathbf{Z} \mid \theta)\right]$.
        Print $\theta_{i+1}$ and $\log p(X \mid \theta_{i+1})$.
    end for
end function
```