Offline to Online Conversion

Marcus Hutter

Research School of Computer Science
Australian National University
Canberra, ACT, 0200, Australia
http://www.hutter1.net/

12 July 2014

Abstract

We consider the problem of converting offline estimators into an online predictor or estimator with small extra regret. Formally this is the problem of merging a collection of probability measures over strings of length 1,2,3,... into a single probability measure over infinite sequences. We describe various approaches and their pros and cons on various examples. As a side-result we give an elementary non-heuristic purely combinatoric derivation of Turing's famous estimator. Our main technical contribution is to determine the computational complexity of online estimators with good guarantees in general.

Contents

Keywords

offline, online, batch, sequential, probability, estimation, prediction, time-consistency, normalization, tractable, regret, combinatorics, Bayes, Laplace, Ristad, Good-Turing.

1 Introduction

A standard problem in statistics and machine learning is to estimate or learn an in general non-i.i.d. probability distribution $q_n:\mathcal{X}^n\to[0,1]$ from a batch of data $x_1,...,x_n$. q_n might be the Bayesian mixture over a class of distributions $\mathcal{M}$, or the (penalized) maximum likelihood (ML/MAP/MDL/MML) distribution from $\mathcal{M}$, or a combinatorial probability, or an exponentiated code length, or else. This is the batch or *offline* setting. An important problem is to predict x_{n+1} from $x_1,...,x_n$ sequentially for $n=0,1,2...$, called *online* learning if the predictor improves with n. A stochastic prediction $\tilde{q}(x_{n+1}|x_{1:n})$ can be useful in itself (e.g. weather forecasts), or be the basis for some decision, or be used for data compression via arithmetic coding, or otherwise. We use the prediction picture, but could have equally well phrased everything in terms of log-likelihoods, or perplexity, or code-lengths, or log-loss.

The naive predictor is $\tilde{q}^{\text{rat}}(x_{n+1}|x_1...x_n):=q_{n+1}(x_1...x_{n+1})/q_n(x_1...x_n)$ is not properly normalized to 1 if q_n and q_{n+1} are not compatible. We could fix the problem by normalization $\tilde{q}^{\text{n1}}(x_{n+1}|x_1...x_n):=\tilde{q}^{\text{rat}}(x_{n+1}|x_1...x_n)/\sum_{x_{n+1}}\tilde{q}^{\text{rat}}(x_{n+1}|x_1...x_n)$, but this may result in a very poor predictor. We discuss two further schemes, $\tilde{q}^{\text{lim}}$ and $\tilde{q}^{\text{mix}}$, the latter having good performance guarantees (small regret), but a direct computation of either is prohibitive.

A major open problem is to find a computationally tractable online predictor $\tilde{q}$ with provably good performance given offline probabilities (q_n). A positive answer would benefit many applications.

Applications. (i) Being able to use an offline estimator to make stochastic predictions (e.g. weather forecasts) is of course useful. The predictive probability needs to sum to 1 which $\tilde{q}^{\text{n1}}$ guarantees, but the regret should also be small, which only $\tilde{q}^{\text{mix}}$ guarantees.

(ii) Given a parameterized class of (already) online estimators $\{\tilde{q}^\theta\}$, estimating the parameter θ from data $x_1...x_n$ (e.g. maximum likelihood) for $n=1,2,3,...$ leads to a sequence of parameters $(\hat{\theta}_n)$ and a sequence of estimators $(q_n):=(\tilde{q}^{\hat{\theta}_n})$ that is usually *not* online. They need to be reconverted to become online to be useful for prediction or compression, etc.

(iii) Arithmetic coding requires an online estimator, but often is based on a class of distributions as described in (ii). The default 'trick' to get a fast and online estimator is to use $\tilde{q}^{\hat{\theta}_n}(x_{n+1}|x_{1:n})$ which is properly normalized and often very good.

(iv) Online conversions are needed even for some offline purposes. For instance, computing the cumulative distribution function $\sum_{y_{1:n}\leq x_{1:n}}q_n(y_{1:n})$ can be hard in general, but can be computed in time $O(n)$ if (q_n) is (converted to) online.

Contributions & contents. The main purpose of this paper is to introduce and discuss the problem of converting offline estimators (q_n) to an online predictor $\tilde{q}$ (Section 2).

We compare and discuss the pros and cons of the four conversion proposals (Section 3). We also define the worst-case extra regret of online $\tilde{q}$ over offline (q_n), measuring the conversion quality.

We illustrate their behavior for various classical estimators (Bayes, MDL, Laplace, Good-Turing, Ristad) (Section 4). Naive normalization of the triple uniform estimator interestingly leads to the Good-Turing estimator, but induces huge extra regret, while naive normalization of Ristad's quadruple uniform estimator induces negligible extra regret.

Given that $\tilde{q}^{\text{n1}}$ can fail for interesting offline estimators, natural questions to ask are: whether the excellent predictor $\tilde{q}^{\text{mix}}$ can be computed or approximated (yes), by an efficient algorithm (no), whether for every (q_n) there exists any fast $\tilde{q}$ nearly as good as $\tilde{q}^{\text{mix}}$ (no), or whether there exist (q_n) for which no fast $\tilde{q}$ can even slightly beat the trivial uniform predictor (yes) (Section 5).

The proofs for these computational complexity results are deferred to the next section (Section 6).

These results do not preclude a satisfactory positive solution in practice, in particular given the contrived nature of the constructed (q_n), but as any negative complexity result they show that a solution requires extra assumptions or to moderate our demands. This leads to some precise open problems to this effect (Section 7).

Proofs for the regret bounds can be found in Appendix A and a list of notation in Appendix B.

As a side-result we give the arguably most convincing (simplest and least heuristic) derivation of the famous Good-Turing estimator. Other attempts at deriving the estimator Alan Turing suggested in 1941 to I.J. Good are less convincing (to us) [Goo53]. They appear more heuristic or convoluted, or are incomplete, often assuming something close to what one wants to get out [Nad85]. Our purely combinatorial derivation also feels right for 1941 and Alan Turing.

2 Problem Formulation

We now formally state the problem of offline to online conversion in three equivalent ways and the quality of a conversion. Let $x_t \in \mathcal{X}$ for $t \in \{1,...,n\}$ and $x_{t:n} := x_t...x_n \in \mathcal{X}^{n-t+1}$, $x_{<n} := x_1...x_{n-1} \in \mathcal{X}^{n-1}$, and $x_{1:0} = x_{<1} = \epsilon$ be the empty string. ln denotes the natural logarithm and log the binary logarithm. $\tilde{q}_{|\mathcal{X}^n}$ constrains the domain $\mathcal{X}^*$ of $\tilde{q}$ to $\mathcal{X}^n$.

Formulation 1 (measures). Given probability measures Q_n on $\mathcal{X}^n$ for $n=1,2,3,...$, find a probability measure $\tilde{Q}$ on $\mathcal{X}^\infty$ close to all Q_n in the sense of $\tilde{Q}(\mathcal{A} \times \mathcal{X}^\infty) \approx Q_n(\mathcal{A})$ for all measurable $\mathcal{A} \subseteq \mathcal{X}^n$ and all n.

For simplicity of notation, we will restrict to countable $\mathcal{X}$, and all examples will be for finite $\mathcal{X} = \{1,...,d\}$. This allows us to reformulate the problem in terms of probability (mass) functions and predictors. A choice for $\approx$ will be given below.

Formulation 2 (probability mass function). Given probability mass functions $q_n : \mathcal{X}^n \to [0;1]$, i.e. $\sum_{x_{1:n}} q_n(x_{1:n}) = 1$, find a function $\tilde{q} : \mathcal{X}^* \to [0;1]$ which is *time-*

consistent (TC) in the sense

$$\sum_{x_n} \tilde{q}(x_{1:n}) \;=\; \tilde{q}(x_{<n}) \;\forall n, x_{<n} \quad \text{and} \quad \tilde{q}(\epsilon) = 1 \tag{TC}$$

and is close to q_n i.e. $\tilde{q}(x_{1:n}) \approx q_n(x_{1:n})$ for all n and $x_{1:n}$.

This is equivalent to Formulation 1, via $q_n(x_{1:n}) := Q_n(\{x_{1:n}\})$, and since $\tilde{q}$ is TC iff there exists $\tilde{Q}$ with $\tilde{q}(x_{1:n}) = \tilde{Q}(\{x_{1:n}\} \times \mathcal{X}^\infty)$ [LH14, Appendix]. We will use the following equivalent predictive formulation, discussed in the introduction, whenever convenient:

Formulation 3 (predictors). Given q_n as before, find a predictor $\tilde{q}: \mathcal{X} \times \mathcal{X}^* \to [0;1]$ which must be *normalized* as

$$\sum_{x_n} \tilde{q}(x_n | x_{<n}) \;=\; 1 \;\; \forall n, x_{<n} \tag{Norm}$$

such that its joint probability

$$\tilde{q}(x_{1:n}) \;:=\; \prod_{t=1}^{n} \tilde{q}(x_t | x_{<t})$$

is close to q_n as before.

$\tilde{q}(x_{1:n})$ is the probability that an (infinite) sequence starts with $x_{1:n}$ and $\tilde{q}(x_n | x_{<n}) \equiv \tilde{q}(x_{1:n}) / \tilde{q}(x_{<n})$ is the probability that x_n follows given $x_{<n}$. Conditions (TC) and (Norm) are equivalent, and are the formal requirement(s) for an estimator to be *online*. We also speak of (q_n) being (not) Norm or TC.

Performance/distance measure. For modelling and coding we want $\tilde{q}$ as large as possible, which suggests the worst-case regret or log-loss regret

$$R_n \;\equiv\; R_n(\tilde{q}) \;\equiv\; R_n(\tilde{q} || q_n) \;:=\; \max_{x_{1:n}} \ln \frac{q_n(x_{1:n})}{\tilde{q}(x_{1:n})} \tag{1}$$

For our qualitative considerations, other continuous $R_n \geq 0$ with $R_n = 0$ iff $\tilde{q}_{|\mathcal{X}^n} = q_n$ would also do. The R_n quantification of $\approx$ above has several convenient properties: Since an online arithmetic code of $x_{1:n}$ w.r.t. $\tilde{q}$ has code length $|\log_2 \tilde{q}(x_{1:n})|$, and an offline Shannon-Fano or Huffman code for $x_{1:n}$ w.r.t. q_n has code length $|\log_2 q_n(x_{1:n})|$, this shows that the online coding of $x_{1:n}$ w.r.t. $\tilde{q}$ leads to codes at most $R_n \ln 2$ bits longer than offline codes w.r.t. q_n. Naturally we are interested in $\tilde{q}$ with small R_n, and indeed we will see that this is always achievable. Also, if q_n is an offline approximation of the true sampling distribution μ, then R_n upper bounds the *extra regret* of a corresponding online approximation $\tilde{q}$:

$$R_n^{\text{online}} - R_n^{\text{offline}} \;\equiv\; R_n(\tilde{q} || \mu) - R_n(q_n || \mu) \;\leq\; R_n(\tilde{q} || q_n) \;\equiv\; R_n \tag{2}$$

Extending $\boldsymbol{q_s}$ from $\boldsymbol{\mathcal{X}^s}$ to $\boldsymbol{\mathcal{X}^\infty}$. Some (natural) offline $(q_n)_{n \in \mathbb{N}}$ considered later are automatically online in the sense that $\tilde{q}$ defined by $\tilde{q}(x_{1:n}) := q_n(x_{1:n}) \; \forall n, x_{1:n}$ is

TC and hence $R_n = 0$ for all n. Note that it is *always* possible to choose $\tilde{q}$ such that $R_n = 0$ for *some* n: For some fixed $s \in \mathbb{N}_0$ define

$$\bar{q}_s(x_{1:n}) \;:=\; \begin{cases} q_s(x_{1:s}) & \text{if} \quad n = s, \\ \sum_{x_{n+1:s}} q_s(x_{1:s}) & \text{if} \quad n < s, \\ q_s(x_{1:s}) Q(x_{s+1:n}|x_{1:s}) & \text{if} \quad n > s \end{cases} \tag{3}$$

where Q can be an arbitrary measure on $\mathcal{X}^\infty$, e.g. uniform $Q(x_{s+1:n}|x_{1:s}) = |\mathcal{X}|^{n-s}$. It is easy to see that $\tilde{q} := \bar{q}_s$ is TC with $R_s(\tilde{q}) = R_s(\bar{q}_s) = R_s(q_s) = 0$, but in general $R_n(\bar{q}_s) > 0$ for $n \neq s$. Therefore naive minimization of R_n w.r.t. $\tilde{q}$ does not work. Minimizing $\lim_{n\to\infty} R_n$ can also fail for a number of reasons: the limit may not exist or is infinite, or minimizing it leads to poor finite-n performance or is not analytically possible or computationally intractable.

3 Conversion Methods

We now consider four methods of converting offline estimators to online predictors and discuss their pros and cons. They illustrate the difficulties and serve as a starting point to a more satisfactory solution.

Naive ratio. The simplest way to define a predictor $\tilde{q}$ from q_n is via *rat*io

$$\tilde{q}^{\text{rat}}(x_t|x_{<t}) \;:=\; \frac{q_t(x_{1:t})}{q_{t-1}(x_{<t})} \quad \text{or equivalently} \quad \tilde{q}^{\text{rat}}(x_{1:n}) \;:=\; q_n(x_{1:n}) \tag{4}$$

While this "solution" is tractable, it obviously only works when q_n already is TC. Otherwise $\tilde{q}^{\text{rat}}$ violates (TC). The deviation of

$$\mathcal{N}(x_{<t}) \;:=\; \sum_{x_t} \tilde{q}^{\text{rat}}(x_t|x_{<t}) \;\equiv\; \frac{\sum_{x_t} q_t(x_{1:t})}{q_{t-1}(x_{<t})} \tag{5}$$

from 1 measures the degree of violation. Note that the expectation of $\mathcal{N}(x_{<t})$ w.r.t. q_{t-1} is 1, so if $\mathcal{N}(x_{<t})$ is smaller than 1 for some $x_{<t}$ it must be larger for others, hence $\max_{x_{<t}} \mathcal{N}(x_{<t}) = 1$ iff $\mathcal{N}(x_{<t}) = 1$ for all $x_{<t} \in \mathcal{X}^{t-1}$.

Naive normalization. Failure of $\tilde{q}^{\text{rat}}(x_t|x_{<t})$ to satisfy (Norm) is easily corrected by normalization [Sol78]:

$$\tilde{q}^{\text{n1}}(x_t|x_{<t}) \;:=\; \frac{q_t(x_{1:t})}{\sum_{x_t} q_t(x_{1:t})} \;\equiv\; \frac{\tilde{q}^{\text{rat}}(x_t|x_{<t})}{\mathcal{N}(x_{<t})} \quad \text{and} \tag{6}$$

$$\tilde{q}^{\text{n1}}(x_{1:n}) \;:=\; \prod_{t=1}^{n} \tilde{q}^{\text{n1}}(x_t|x_{<t}) \;\equiv\; \frac{q_n(x_{1:n})}{\prod_{t=1}^{n} \mathcal{N}(x_{<t})} \tag{7}$$

This guarantees TC and for small $\mathcal{X}$ is still tractable, but note that $\tilde{q}^{\text{n1}}_{|\mathcal{X}^n} \not\equiv q_n$ unless q_n is already TC. Unfortunately, this way of normalization can result in poor performance and very large regret R_n for finite n and asymptotically. Even if performance is good, computing R_n or finding good upper bounds can be very hard. Using (1) and (7), the regret can be represented and upper bounded as follows:

$$R_n(\tilde{q}^{\rm n1}) \;=\; \max_{x_{1:n}}\sum_{t=1}^n \ln\mathcal{N}(x_{<t}) \;\leq\; \sum_{t=1}^n \ln\max_{x_{<t}}\mathcal{N}(x_{<t}) \tag{8}$$

If q_n is TC, then $\mathcal{N}\equiv 1$, hence R_n as well as the upper bound are 0.

Let us consider here a simple but artificial example how bad things can get, following up with important practical examples in the next section. For an i.i.d. estimator $q_n(x_{1:n})=q_n(x_1)\cdot...\cdot q_n(x_n)$, where we slightly overloaded notation, $\tilde{q}^{\rm n1}(x_t|x_{<t})=q_t(x_t)$ and $\tilde{q}^{\rm n1}(x_{1:n})=q_1(x_1)\cdot...\cdot q_n(x_n)$, therefore by definition (1)

$$R_n(\tilde{q}^{\rm n1}) \;=\; \max_{x_{1:n}}\ln\prod_{t=1}^n \frac{q_n(x_t)}{q_t(x_t)} \;=\; \sum_{t=1}^n \ln\max_{x_t}\frac{q_n(x_t)}{q_t(x_t)}$$

We now consider $\mathcal{X}=\{0,1\}$ with concrete Bernoulli($^2\!/\!_3$) probability $q_n(x_t=1)={}^2\!/\!_3$ for even n and Bernoulli($^1\!/\!_3$) probability $q_n(x_t=1)={}^1\!/\!_3$ for odd n. We see that for even t,

$$\tilde{q}^{\rm rat}(1_t|1_{<t}) \;=\; \frac{q_t(1_1)\cdot...\cdot q_t(1_{t-1})\cdot q_t(1_t)}{q_{t-1}(1_1)\cdot...\cdot q_{t-1}(1_{t-1})} \;=\; 2^{t-1}\cdot\frac{2}{3}$$

is very badly unnormalized. Indeed $R_n(\tilde{q}^{\rm n1})$ grows linearly with n, i.e. becomes very large:

$$R_n(\tilde{q}^{\rm n1}) \;=\; \sum_{t=1}^n \ln\left\{\begin{matrix} 1 & \text{if} & n-t \text{ is even} \\ 2 & \text{if} & n-t \text{ is odd}\end{matrix}\right\} \;=\; \lfloor\frac{n}{2}\rfloor\ln 2$$

Limit. We have seen how to make $R_s=0$ for any fixed s using $\bar{q}_s$ (3). A somewhat natural idea is to define

$$\tilde{q}^{\rm lim}(x_{1:n}) \;:=\; \lim_{s\to\infty}\bar{q}_s(x_{1:n}) \;=\; \lim_{s\to\infty}\sum_{x_{n+1:s}} q_s(x_{1:s})$$

in the hope to make $\lim_{s\to\infty}R_s=0$. Effectively what $\tilde{q}^{\rm lim}$ does is to use q_s for very large s also for short strings of length n by marginalization. Problems are plenty: The limit may not exist, may exist but be incomputable, R_n may be hard to impossible to compute or upper bound, and even if the limit exists, $\tilde{q}^{\rm lim}$ may perform badly.

For instance, for the above Bernoulli($^1\!/\!_3|^2\!/\!_3$) example, the argument of the limit

$$\tilde{q}^{\rm lim}(x_{1:n}) \;=\; \lim_{s\to\infty}\sum_{x_{n+1:s}} q_s(x_1)\cdot...\cdot q_s(x_s) \;=\; \lim_{s\to\infty}[q_s(x_1)\cdot...\cdot q_s(x_n)]$$

oscillates indefinitely (except if $x_1+...+x_n={}^n\!/\!_2$). A template leading to a converging but badly performing $\tilde{q}^{\rm lim}$ is $q_n(x_{1:n})=\text{Bad}(x_{<\lfloor n/2\rfloor})\cdot\text{Good}(x_{\lfloor n/2\rfloor:n})$. While offline $q_n(x_{1:n})$ is a "good" estimator on half of the data, $\tilde{q}^{\rm lim}(x_{1:n})=\text{Bad}(x_{1:n})$ is "bad" on all data. For example, $\text{Bad}(x_{1:n}):=|\mathcal{X}|^{-n}$ (see *Uniform* next Section) and $\text{Good}(x_{1:n})=\binom{n+d-1}{n_1...n_d\ d-1}$ (see *Laplace* next Section) or simpler $\text{Good}(1_{1:n})=1$, lead to $R_n(\tilde{q}^{\text{lim}})\propto n$.

Mixture. Another way of exploiting $\bar{q}_s$ is as follows: Rather than taking the limit $s\to\infty$ let us consider the class $\{\bar{q}_1,\bar{q}_2,...\}$ of *all* $\bar{q}_s$. This corresponds to a set of measures on $\mathcal{X}^\infty$, each good in a particular circumstance, namely $\bar{q}_s$ is good and indeed perfect at time s. It is therefore natural to consider a Bayesian mixture over this class [San06]

$$\tilde{q}^{\text{mix}}(x_{1:n}) \;:=\; \sum_{s=0}^{\infty}\bar{q}_s(x_{1:n})w_s \quad \text{with prior} \quad w_s>0,\; \sum_{s=0}^{\infty}w_s=1. \tag{9}$$

$\tilde{q}^{\text{mix}}$ is TC and its regret can easily be upper bounded [San06]:

$$R_n(\tilde{q}^{\text{mix}}) \;=\; \max_{x_{1:n}}\ln\frac{q_n(x_{1:n})}{\sum_{s=0}^{\infty}\bar{q}_s(x_{1:n})w_s} \;\leq \max_{x_{1:n}}\ln\frac{q_n(x_{1:n})}{\bar{q}_n(x_{1:n})w_n} \;=\; \ln w_n^{-1} \tag{10}$$

For e.g. $w_n:=\frac{1}{(n+1)(n+2)}$ we have $\ln w_n^{-1}\leq 2\ln(n+2)$ which usually can be regarded as small. This shows that *any* offline estimator can be converted into an online predictor with very small extra regret (2). Note that while $\tilde{q}^{\text{mix}}$ depends on arbitrary Q defined in (3), the upper bound (10) on R_n does not. Unfortunately it is unclear how to convert this heavy construction into an efficient algorithm.

A variation is to set $Q\equiv 0$, which makes $\tilde{q}^{\text{mix}}$ a semi-measure, which could be made TC by naive normalization (7). Bound (10) still holds since for $\tilde{q}^{\text{mix}}$ with $Q\equiv 0$ the normalizer $\mathcal{N}\leq 1$. Another variation is as follows. Often q_n violates TC only weakly, in which case a sparser prior, e.g. $w_{2^k}:=\frac{1}{(k+1)(k+2)}$ and $w_n=0$ for all other n, can lead to even smaller regret.

Further choices for $\tilde{q}$. Of course the four presented choices for $\tilde{q}$ do not exhaust all options. Indeed, finding a tractable $\tilde{q}$ with good properties is a major open problem. Several estimation procedures do not only provide q_n on $\mathcal{X}^n$, but measures on $\mathcal{X}^\infty$ or equivalently for *each n separately* a TC $q_n:\mathcal{X}^*\to[0;1]$ (see Bayes and crude MDL below). While this opens further options for $\tilde{q}$, e.g. $\tilde{q}(x_{n+1}|x_{1:n}):=q_n(x_{1:n+1})/q_n(x_{1:n})$ with some (weak) results for MDL [PH05], it does not solve our main problem.

Notes. Each solution attempt has its down-sides, and a solution satisfying all our criteria remains open.

It is easy to verify that, if q_n is already TC, the first three definitions of $\tilde{q}$ coincide, and $R_n=0$, which is reassuring, but $\tilde{q}_n^{\text{mix}}$ in general differs due to the arbitrary w in (9) and arbitrary Q in $\bar{q}$ in (3).

4 Examples

All examples below fall in one of two major strategies for designing estimators (the introduction mentions others we do not consider). One strategy is to start with a class $\mathcal{M}$ of probability measures ν on $\mathcal{X}^\infty$ in the hope one of them is good. For instance, $\mathcal{M}$ may contain (a subset of) i.i.d. measures $\nu_{\boldsymbol{\theta}}(x_{1:n}) := \theta_{x_1}\cdot...\cdot\theta_{x_n}$ with $\theta_i \geq 0$ and $\theta_1+...+\theta_d = 1$ and $d := |\mathcal{X}|$. One may either select a ν from $\mathcal{M}$ informed by given data $x_{1:n}$ or take an average over the class. The other strategy assigns uniform probabilities over subsets of $\mathcal{X}^n$. This combinatorial approach will be described later. Some strategies lead to TC and some examples are TC. For the others we will discuss the various online conversions $\tilde{q}$.

Bayes. The Bayesian mixture over $\mathcal{M}$ w.r.t. some prior (density) $w()$ is defined as

$$q_n(x_{1:n}) \;:=\; \int_{\mathcal{M}} \nu(x_{1:n})\, w(\nu)\, d\nu$$

Since q_n is TC, $(q_n^{\text{rat}}) \equiv (q_n^{\text{n1}}) \equiv (q_n^{\text{lim}})$ coincide with $\tilde{q}$, $R_n = 0$, and $\tilde{q}^{\text{rat}}$ is tractable if the Bayes mixture is. Note that $\tilde{q} \notin \mathcal{M}$ in general, in particular it is not i.i.d. Assume the true sampling distribution μ is in $\mathcal{M}$. For countable $\mathcal{M}$ and counting measure $d\nu$, we have $q_n(x_{1:n}) \geq \mu(x_{1:n}) w(\mu)$, hence $R_n^{\text{online}} = R_n^{\text{offline}} \leq \ln w(\mu)^{-1}$. For continuous classes $\mathcal{M}$ we have $R_n^{\text{online}} = R_n^{\text{offline}} \lesssim \ln w(\mu)^{-1} + O(\ln n)$ under some mild conditions [BC91, Hut03, RH07].

MDL/NML/MAP. The MAP or MDL estimator is

$$\hat{q}_n(x_{1:n}) \;:=\; \sup_{\nu\in\mathcal{M}} \{\nu(x_{1:n})\, w(\nu)\} \quad \text{and} \quad q_n(x_{1:n}) \;:=\; \frac{\hat{q}_n(x_{1:n})}{\sum_{x_{1:n}} \hat{q}_n(x_{1:n})}$$

Since $\hat{q}_n$ is not even a probability on $\mathcal{X}^n$, we have to normalize it to q_n. For uniform prior density $w()$, $\hat{q}_n$ is the maximum likelihood (ML) estimator, and q_n is known under the name normalized maximum likelihood (NML) or modern minimum description length (MDL). Unlike Bayes, q_n is *not* TC, which causes all kinds of complications [Grü07, Hut09, LH14], many of them can be traced back to our main open problem and the unsatisfactory choices for $\tilde{q}$ [PH05]. R_n^{offline} is essentially the same as for Bayes under similar conditions, but R_n^{online} depends on the choice of $\tilde{q}$. Crude MDL simply selects $q_n := \text{argmax}_{\nu\in\mathcal{M}}\{\nu(x_{1:n})\, w(\nu)\}$ at time n, which is a probability measure on $\mathcal{X}^\infty$. While this opens additional options for defining $\tilde{q}$, they also can perform poorly in the worst case [PH05]. Note that most versions of MDL perform often very well in practice, comparable to Bayes; robustness and proving guarantees are the open problems.

Uniform. The uniform probability $q_n(x_{1:n}) := |\mathcal{X}|^{-n}$ is TC, hence all four $\tilde{q}$ coincide and $R_n = 0$ (only for uniform Q in case of q_n^{mix}). Unless data is uniform, this is a lousy estimator, since predictor $\tilde{q}(x_t|x_{<t}) = 1/|\mathcal{X}|$ is indifferent and ignores all evidence $x_{<t}$ to the contrary. But the basic idea of uniform probabilities is sound, if applied smartly: The general idea is to partition the sample space (here $\mathcal{X}^n$) into $\mathcal{P} = \{S_1,...,S_{|\mathcal{P}|}\}$ and assign uniform probabilities to each partition: $q_n(x_{1:n}|S_r) = 1/|S_r|$ and a (possibly) uniform probability to the parts themselves $q_n(S_r) = 1/|\mathcal{P}|$. For small $|\mathcal{P}|$, $q_n(x_{1:n}) = q_n(x_{1:n}|S_r) q_n(S_r)$ is never more than a small factor $|\mathcal{P}|$

smaller than uniform $|\mathcal{X}|^{-n}$ but may be a huge factor of $|\mathcal{X}|^n/|S_r||\mathcal{P}|$ larger. The Laplace rule can be derived that way, and the Good-Turing and Ristad estimators by further sub-partitioning.

Laplace. More interesting than the uniform probability is the following double uniform combinatorial probability: Let $n_i:=|\{t:x_t=i\}|$ be the number of times, symbol $i\in\mathcal{X}=\{1,...,d\}$ appears in $x_{1:n}$. We assign a uniform probability to all sequences $x_{1:n}$ with the same counts $\boldsymbol{n}:=(n_1,...,n_d)$, therefore $q_n(x_{1:n}|\boldsymbol{n})=\binom{n}{n_1...n_d}^{-1}$. We also assign a uniform probability to the counts $\boldsymbol{n}$ themselves, therefore $q_n(\boldsymbol{n})=|\{\boldsymbol{n}:n_1+...+n_d=n\}|^{-1}=\binom{n+d-1}{d-1}^{-1}$. Together

$$q_n(x_{1:n}) \;=\; \binom{n}{n_1\;...\;n_d}^{-1}\binom{n+d-1}{d-1}^{-1} \;=\; \binom{n+d-1}{n_1\;...\;n_d\;\;d-1}^{-1}$$

$$\text{hence}\;\; \tilde{q}^{\text{rat}}(x_{n+1}=i|x_{1:n}) \;=\; \frac{q_{n+1}(x_{1:n}i)}{q_n(x_{1:n})} \;=\; \frac{n_i+1}{n+d}$$

is properly normalized (Norm), so $\tilde{q}^{\text{rat}}$ is TC, and $(q_n^{\text{rat}})\equiv(q_n^{\text{n1}})\equiv(q_n^{\text{lim}})$ coincide with $\tilde{q}$ and $R_n=0$. $\tilde{q}^{\text{rat}}$ is nothing but Laplace's famous rule.

Good-Turing. Even more interesting is the following triple uniform probability: Let $M_r:=\{i:n_i=r\}$ be the symbols that appear exactly $r\in\mathbb{N}_0$ times in $x_{1:n}$, and $m_r:=|M_r|$ be their number. Clearly $m_r=0$ for all $r>n$, but due to $\sum_{r=0}^n r\cdot m_r=n$, $m_r=0$ also for many $r<n$. We assign uniform probabilities to $q_n(x_{1:n}|\boldsymbol{n})$ as before and to $q_n(\boldsymbol{n}|\boldsymbol{m})$ and to $q_n(\boldsymbol{m})$, where $\boldsymbol{m}:=(m_0,...,m_n)$. There are $\binom{d}{m_0...m_n}$ ways to distribute symbols $1,...,d$ into sets $(M_0,...,M_n)$ (many of them empty) of sizes $m_0,...,m_n$. Therefore $q_n(\boldsymbol{n}|\boldsymbol{m})=\binom{d}{m_0...m_n}^{-1}$. Each $\boldsymbol{m}$ constitutes a decomposition of n into natural summands with repetition but without regard to order. The number of such decompositions is a well-known function [AS74, §24.2.2] which we denote by $\text{Part}(n)$. Therefore $q_n(\boldsymbol{m})=\text{Part}(n)^{-1}$. Together

$$q_n(x_{1:n}) \;=\; \binom{n}{n_1\;...\;n_d}^{-1}\binom{d}{m_0\;...\;m_n}^{-1}\text{Part}(n)^{-1}, \quad\text{hence} \tag{11}$$

$$\tilde{q}^{\text{rat}}(x_{n+1}=i|x_{1:n}) \;=\; \frac{q_{n+1}(x_{1:n}i)}{q_n(x_{1:n})} = \frac{n_i+1}{n+1}\cdot\frac{m_{r+1}+1}{m_r}\cdot\frac{\text{Part}(n)}{\text{Part}(n+1)},\;\; r=n_i \tag{12}$$

This is not TC as can be verified by example, but is a very interesting predictor: The first term is close to a frequency estimate n_i/n. The second term is close to the Good-Turing (GT) correction m_{r+1}/m_r. The intuition is that if e.g. many symbols have appeared once (m_1 large), but few twice (m_2 small), we should be skeptical of observing a symbol that has been observed only once another time, since it would move from a likely category to an unlikely one. The third term $\frac{\text{Part}(n)}{\text{Part}(n+1)}\to 1$ for $n\to\infty$. The normalized version

$$\tilde{q}^{\text{n1}}(x_{n+1}=i|x_{1:n}) \;=\; \frac{\tilde{q}^{\text{rat}}(x_{n+1}=i|x_{1:n})}{\sum_{x_{n+1}}\tilde{q}^{\text{rat}}(x_{n+1}|x_{1:n})} \;=\; \frac{1}{\mathcal{N}_n}\cdot\frac{r+1}{n+1}\cdot\frac{m_{r+1}+1}{m_r} \tag{13}$$

$$\text{where}\;\; \mathcal{N}_n \;:=\; \frac{1}{n+1}\sum_{r=0,m_r\neq 0}^{n}(r+1)(m_{r+1}+1) \tag{14}$$

is even closer to the GT estimator. We kept $\frac{1}{n+1}$ as in [Goo53, Eq.(13)], while often $\frac{1}{n}$ is seen due to [Goo53, Eq.(2)]. Anyway after normalization there is no difference. The only difference to the GT estimator is the appearance of $m_{r+1}+1$ instead of m_{r+1}. Unfortunately its regret is very large:

Theorem 1 (Naively normalized triple uniform estimator) *Naive normalization of the triple uniform combinatorial offline estimator q_n defined in* (11) *leads to the (non-smoothed) Good-Turing estimator $\tilde{q}^{n1}$ given in* (13) *with regret*

$$R_n(\tilde{q}^{n1}||q_n) = \max_{x_{1:n}}\Big\{\sum_{t=1}^n \ln\mathcal{N}_{t-1}\Big\} - \ln(Part(n)) \begin{cases} = n\ln 2 \pm O(\sqrt{n}) \text{ for } |\mathcal{X}| = \infty \\ \geq 0.43n - O(\sqrt{n}) \text{ for } |\mathcal{X}| \geq 3 \end{cases} \tag{15}$$

Inserting (12) and (14) into (6) we get $\mathcal{N}(x_{1:n}) = \frac{\tilde{q}^{\text{rat}}(x_{n+1}|x_{1:n})}{\tilde{q}^{\text{n1}}(x_{n+1}|x_{1:n})} = \frac{\text{Part}(n)}{\text{Part}(n+1)}\mathcal{N}_n$ which by (8) implies the first equality. We prove the last equality in Appendix A by showing that the maximizing sequence is $x_{1:\infty} = 1223334444...$ with $\mathcal{N}_n = 2\pm O(n^{-1/2})$ which requires infinite d or at least $d \geq \sqrt{2n}$. We also show that $R_n \geq 0.43n - O(\sqrt{n})$ for every $d \geq 3$. The linearly growing R_n shows that naive normalization severely harms the offline triple uniform estimator q_n.

Indeed, raw GT performs very poorly for large r in practice, but smoothing the function $m_{()}$ leads to an excellent estimator in practice [Goo53], e.g. Kneser-Ney smoothing for text data [CG99]. Our $m_{r+1} \rightsquigarrow m_{r+1}+1$ is a kind of albeit insufficient smoothing. $\tilde{q}^{\text{mix}}$ may be regarded as an (unusual) kind of smoothing, which comes with the strong guarantee $R_n \leq 2\ln(n+2)$, but a direct computation is prohibitive. [San06] gives a low-complexity smoothing of the original GT that comes with guarantees, namely sub-linear $O(n^{2/3})$ log worst-case sequence attenuation, but this is different from R_n in various respects: Log worst-case sequence-attenuation is relative to i.i.d. coding and unlike R_n lower bounded by $O(n^{1/3})$. Still a similar construction may lead to sublinear and ideally logarithmic R_n.

Ristad [Ris95] designed an interesting quadruple uniform probability motivated as follows: If $\mathcal{X}$ is the set of English words and $x_{1:n}$ some typical English text, then most symbols=words will not appear ($d \gg n$). In this case, Laplace assigns not enough probability ($\frac{n_i+1}{n+d} \ll \frac{n_i}{n}$) to observed words. This can be rectified by treating symbols $\mathcal{A} := \{i : n_i > 0\}$ that do appear different from symbols $\mathcal{X}\setminus\mathcal{A}$ that don't. For $n > 0$, $x_{1:n}$ may contain $m \in \{1,...,\min\{n,d\}\}$ different symbols, so we set $q_n(m) = 1/\min\{n,d\}$. Now choose uniformly which m symbols $\mathcal{A}$ appear, $q_n(\mathcal{A}|m) = \binom{d}{m}^{-1}$ for $|\mathcal{A}| = m$. There are $\binom{n-1}{m-1}$ ways of choosing the frequency of symbols consistent with $n_1+...+n_d = n$ and $n_i > 0 \Leftrightarrow i \in \mathcal{A}$, hence $q_n(\boldsymbol{n}|\mathcal{A}) = \binom{n-1}{m-1}^{-1}$. Finally, $q_n(x_{1:n}|\boldsymbol{n}) = \binom{n}{n_1...n_d}^{-1}$ as before. Together

$$q_n(x_{1:n}) \;=\; \binom{n}{n_1\;...\;n_d}^{-1}\binom{n-1}{m-1}^{-1}\binom{d}{m}^{-1}\frac{1}{\min\{n,d\}}, \quad \text{which implies} \tag{16}$$

$$\tilde{q}^{\rm rat}(x_{n+1}=i|x_{1:n}) \;=\; \frac{\min\{n,d\}}{\min\{n+1,d\}}\cdot\left\{ \begin{array}{ll} \frac{(n_i+1)(n-m+1)}{n(n+1)} & \text{if} \quad n_i>0 \\ \frac{m(m+1)}{n(n+1)}\cdot\frac{1}{d-m} & \text{if} \quad n_i=0 \end{array} \right.$$

This is not TC, since

$$\mathcal{N}(x_{1:n}) \;=\; \frac{\min\{n,d\}}{\min\{n+1,d\}}\cdot\left\{ \begin{array}{ll} 1+\frac{2m}{n(n+1)} & \text{if} \quad m<d \\ 1-\frac{m(m-1)}{n(n+1)} & \text{if} \quad m=d \end{array} \right.$$

is not identically 1. Normalization leads to

$$\tilde{q}^{\rm n1}(x_{n+1}=i|x_{1:n}) \;=\; \left\{ \begin{array}{ll} \frac{(n_i+1)(n-m+1)}{n(n+1)+2m} & \text{if} \quad n_i>0 \text{ and } m<d \\ \frac{m(m+1)}{n(n+1)+2m}\cdot\frac{1}{d-m} & \text{if} \quad n_i=0 \\ \frac{n_i+1}{n+m} & \text{if} \quad m=d \;\; [\Rightarrow n_i>0] \end{array} \right. \tag{17}$$

For $n\!=\!0$ we have $\tilde{q}^{\rm rat}(x_1)\!=\!\tilde{q}^{\rm n1}(x_1)\!=\!q_n(x_1)\!=\!1/d$ and $\mathcal{N}(\epsilon)\!=\!1$. While by construction, the offline estimator should have good performance (in the intended regime), the performance of the online version depends on how much the normalizer exceeds 1. The first factor in $\mathcal{N}$ is ≤ 1 and the $m=d$ case is ≤ 1. Therefore $\mathcal{N}(x_{1:n})\leq 1\!+\!\frac{2m}{n(n+1)}\leq 1\!+\!\frac{2}{n+1}$, where we have used $m\!\leq\! n$ in the second step. The regret can hence be bounded by

$$R_n(\tilde{q}^{\rm n1}) \;\leq\; \sum_{t=1}^n \ln\max_{x_{<t}}\mathcal{N}(x_{<t}) \;\leq\; \sum_{t=2}^n \ln(1\!+\!\tfrac{2}{t}) \;\leq\; \sum_{t=2}^n \tfrac{2}{t} \;\leq\; 2\ln n$$

Theorem 2 (Quadruple uniform estimator) *Naive normalization of Ristad's quadruple uniform combinatorial offline estimator q_n defined in* (16) *leads to Ristad's natural law $\tilde{q}^{n1}$ given in* (17) *with regret $R_n(\tilde{q}^{n1}||q_n)\!\leq\! 2\ln n$.*

This shows that simple normalization does not ruin performance. Indeed, the regret bound is as good as we are able to guarantee in general via $\tilde{q}^{\rm mix}$.

5 Computational Complexity of $\tilde{q}$

Computability and complexity of $\tilde{q}^{\rm mix}$. From the four discussed online estimators only $q_n^{\rm mix}$ guarantees small extra regret over offline (q_n) in general, but the definition of $\tilde{q}^{\rm mix}$ is quite heavy and at first it is not even clear whether it is computable. The following theorem shows that $\tilde{q}^{\rm mix}$ can be computed to relative accuracy ε in double-exponential time:

Theorem 3 (Computational complexity of $\tilde{q}^{\rm mix}$) *There is an algorithm A that computes $\tilde{q}^{mix}$ (with uniform choice for Q) to relative accuracy $|A(x_{1:n},\varepsilon)/\tilde{q}^{mix}(x_{1:n})-1|<\varepsilon$ in time $O(|\mathcal{X}|^{4|\mathcal{X}|^n/\varepsilon})$ for all $\varepsilon\!>\!0$.*

The relative accuracy ε allows us to compute the predictive distribution $\tilde{q}^{\text{mix}}(x_t|x_{<t})$ to accuracy ε, ensures $A(x_{1:n},\varepsilon) > (1-\varepsilon)\tilde{q}_n^{\text{mix}}(x_{1:n})$, hence $R_n(A(\cdot,\varepsilon)||q_n) \leq R_n(\tilde{q}^{\text{mix}}||q_n) + \frac{\varepsilon}{1-\varepsilon}$, and approximate normalization $|1-\sum_{x_{1:n}}A(x_{1:n},\varepsilon)|<\varepsilon$.

Computational complexity of general $\tilde{q}$. The existence of $\tilde{q}^{\text{mix}}$ shows that any offline estimator can be converted into an online estimator with minimal extra regret $R_n \leq 2\ln(n+2)$. While encouraging and of theoretical interest, the provided algorithm for $\tilde{q}^{\text{mix}}$ is prohibitive. Indeed, Theorem 4 below establishes that there exist offline (q_n) computable in polynomial time for which the fastest algorithm for *any* online (=TC) $\tilde{q}$ with $R_n \leq O(\log n)$ is at least exponential in time.

Trivially $R_n \leq n\ln|\mathcal{X}|$ can always be achieved for any (q_n) by uniform $\tilde{q}(x_{1:n}) = |\mathcal{X}|^{-n}$. So a very modest quest would be $R_n \leq (1-\varepsilon)n\ln|\mathcal{X}|$. If we require $\tilde{q}$ to run in polynomial time but with free oracle access to (q_n), Theorem 5 below shows that this is also not possible for some exponential time (q_n).

Together this does not rule out that for every fast (q_n) there exists a fast $\tilde{q}$ with e.g. $R_n \leq \sqrt{n}$. This is our main remaining open problem to be discussed in Section 7.

The main proof idea for both results is as follows: We construct a deterministic (q_s) that is 1 on the sequence of quasi-independent quasi-random strings $\dot{x}^1_{1:1}$, $\dot{x}^2_{1:2}$, $\dot{x}^3_{1:3}$, ... The only way for $\tilde{q}(x_{1:n})$ to be not too much smaller than $\bar{q}_s(\dot{x}^s_{1:n})$ is to know $\dot{x}^s_{1:s}$. If $s=s(n)$ is exponential in n this costs exponential time. If $\tilde{q}$ has only oracle access to (q_s), it needs exponentially many oracle calls even for linear $s(n)=(1+\varepsilon)n$.

The general theorem is a bit unwieldy and is stated and proven in the next section. Here we present and discuss the most interesting special cases. $\text{TIME}(g(n))$ is defined as the class of all algorithms that run in time $O(g(n))$ on inputs of length n. Real-valued algorithms produce for any rational $\varepsilon>0$ given as an extra argument, an ε-approximation in this time, as did $A(x_{1:n},\varepsilon)$ for $\tilde{q}^{\text{mix}}$ above. Algorithms in $\text{E}^c := \text{TIME}(2^{cn})$ run in exponential time, while $\text{P} := \bigcup_{k=1}^{\infty}\text{TIME}(n^k)$ is the classical class of all algorithms that run in polynomial time (strictly speaking Function-P or FP [AB09]). The theorems don't rest on any complexity separation assumptions such as $\text{P}\neq\text{NP}$. We only state and prove the theorems for binary alphabet $\mathcal{X}=\mathbb{B}=\{0,1\}$. The generalization to arbitrary finite alphabet is trivial. 'For all large n' shall mean 'for all but finitely many n', denoted by $\forall' n$. $m>0$ is a constant that depends on the machine model, e.g. $m=1$ for a random access machine (RAM).

Theorem 4 (Sub-optimal fast online for fast offline) *For all $r>0$ and $c>0$ and $\varepsilon>0$*

- (*i*) $\exists(q_s) \in \mathit{TIME}(s^{b+m})\ \forall\tilde{q} \in E^c : R_n(\tilde{q}||q_n) \geq r\ln n\ \forall' n,$ *where* $b := \frac{c+1+\varepsilon}{1-\varepsilon}r$
- (*ii*) *in particular for large c and r:* $\exists(q_s) \in P\ \forall\tilde{q} \in E^c : R_n \geq r\ln n\ \forall' n$
- (*iii*) *in particular for small c,ε:* $\exists(q_s) \in \mathit{TIME}(s^{r+m+\varepsilon})\ \forall\tilde{q}\in P : R_n \geq r\ln n\ \forall' n$
- (*iv*) *in particular for $\tilde{q}^{mix}$:* $\exists(q_s) \in P : \tilde{q}^{mix} \notin E^c$

In particular (iii) implies that there is an offline estimator (q_s) computable in quartic time s^4 on a RAM for which no polynomial-time online estimator $\tilde{q}$ is as good as $\tilde{q}^{\text{mix}}$.

The slower (q_s) we admit (larger r), the higher the lower bound gets. (ii) says that even algorithms for $\tilde{q}$ running in exponential time 2^{cn} cannot achieve logarithmic regret for all $(q_s)\in\mathrm{P}$. In particular this implies that (iv) any algorithm for $\tilde{q}^{\text{mix}}$ requires super-exponential time for some $(q_s)\in\mathrm{P}$ on some arguments.

The next theorem is much stronger in the sense that it rules out even very modest demands on R_n but is also much weaker since it only applies to online estimators for slow (q_s) used as a black box oracle. That is, $\tilde{q}^o(x_{1:n})$ can call $q_s(z_{1:s})$ for any s and $z_{1:s}$ and receives the correct answer. We define $\mathrm{TIME}^o(g(n))$ as the class of all algorithms with such oracle access that run in time $O(g(n))$, where each oracle call is counted only as one step, and similarly P^o and $\mathrm{E}^{c,o}$.

Theorem 5 (Very poor fast online using offline oracle) *For all* $\varepsilon>0$

$$\exists o\equiv(q_s)\in E^1\ \forall\tilde{q}^o\in E^{\varepsilon/2,o}: R_n(\tilde{q}^o||q_n)\geq(1-\varepsilon)n\ln 2\ \forall' n$$
$$\textit{or cruder: } \exists o\equiv(q_s)\ \forall\tilde{q}^o\in P^o: R_n(\tilde{q}^o||q_n)\geq(1-\varepsilon)n\ln 2\ \forall' n$$

The second line states that the trivial bound $R_n\leq n\ln 2$ achieved by the uniform distribution can in general not be improved by a fast $\tilde{q}^o$ that (only) has oracle access to the offline estimator.

Usually one Does not state the complexity of the oracle, since it does not matter, but knowing that an $o\in\mathrm{E}^1$ is sufficient (first line) tells us something: First, the negative result is not an artifact of some exotic non-computable offline estimator. On the other hand, if an exponential time offline o is indeed needed to make the result true, the result wouldn't be particularly devastating. It is an open question whether an $o\in\mathrm{P}$ can cause such bad regret.

6 Computational Complexity Proofs

Proof of Theorem 3. The design of an algorithm for $\tilde{q}^{\text{mix}}$ and the analysis of its run-time follows standard recipes, so will only be sketched. A real-valued function $\tilde{q}^{\text{mix}}:\mathcal{X}^*\to[0;1]$ is (by definition) computable (also called estimable [Hut05]), if there is an always halting algorithm $A:\mathcal{X}^*\times\mathbb{Q}^+\to\mathbb{Q}$ with $|A(x_{1:n},\varepsilon)-\tilde{q}^{\text{mix}}(x_{1:n})|<\varepsilon$ for all rational $\varepsilon>0$. We assume there is an oracle q_t^ε that provides q_t to ε-accuracy in time $O(1)$. We assume that real numbers can be processed in unit time. In reality we need $O(\ln{}^1\!/\!_\varepsilon)$ bits to represent, and time to process, real numbers to accuracy ε. This leads to some logarithmic factors in run-time which are dwarfed by our exponentials, so will be ignored. To compute $\bar{q}_s(x_{1:n})$ to accuracy ${}^\varepsilon\!/\!_2$ we need to call $q_s^{\varepsilon/2N}$ oracle $N:=\max\{|\mathcal{X}|^{s-n},1\}$ times and add up all numbers. We can compute $\tilde{q}^{\text{mix}}$ to ε-accuracy by the truncated sum $\sum_{s=0}^{2/\varepsilon}\bar{q}_s^{\varepsilon/2}(x_{1:n})w_s$ with $w_s=\frac{1}{(s+1)(s+2)}$, since the tail sum is bounded by ${}^\varepsilon\!/\!_2$. Hence overall runtime is $O(|\mathcal{X}|^{2/\varepsilon-n})$. But this is not sufficient. For large n, $\tilde{q}^{\text{mix}}(x_{1:n})$ is typically small, and we need a *relative* accuracy of ε, i.e. $|A(x_{1:n},\varepsilon')/\tilde{q}^{\text{mix}}(x_{1:n})-1|<\varepsilon$. For $Q(x_{1:n})=|\mathcal{X}|^{-n}$, we have $\tilde{q}^{\text{mix}}(x_{1:n}) \geq \frac{1}{2}Q(x_{1:n}) = \frac{1}{2}|\mathcal{X}|^{-n}$, hence $\varepsilon' = \frac{\varepsilon}{2}|\mathcal{X}|^{-n}$ suffices. Run time becomes $O(|\mathcal{X}|^{\frac{4}{\varepsilon}|\mathcal{X}|^n - n}) \leq e^{e^{O(n)}/\varepsilon}$. ■

Theorem 6 (Fast offline can imply slow online (general)) *Let $s(n)$ and $f(n)$ and $g(n)$ be monotone increasing functions. $s(n)$ shall be injective and $\geq n$ for large n with inverse $n(s) := \max\{n : s(n) \leq s\}$ and $g(n) < \frac{1}{2}n^{-\delta}h(n)$, where $h(n) := 2^{s(n)-n}[n^{-\gamma} - 2^{f(s(n))-n}]$. $m > 0$ is a constant depending on the machine model, e.g. $m=1$ for a RAM. Then for all $\gamma > 0$ and $\delta > 0$ it holds that*

$$\exists o \equiv (q_s) \in \mathit{TIME}(n(s)^{\gamma+\delta} 2^{n(s)} g(n(s)) s^m)$$
$$\forall \tilde{q}^o \in \mathit{TIME}^o(g(n)) : R_n(\tilde{q}^o || q_n) \geq f(n) \ln 2 \ \forall' n$$

Proof of Theorem 6.

Effective quasi-sparse sets. We need a single set $\{\dot{x}^s_{1:s}\}_{s\in\mathbb{N}} \equiv \{\dot{x}^1_1, \dot{x}^2_{1:2}, \dot{x}^3_{1:3}, ...\}$ of sequences that is "safe" against *every* polynomial time $\tilde{q}$ in a sense to be clarified below. Let $o = (q_s)$ be any deterministic oracle, i.e. for every s, q_s is 1 on exactly one string, namely $\dot{x}^s_{1:s}$. Let $T^o_1, T^o_2, ...$ be an enumeration of all Turing machines with access to oracle o, but each $T^o_k(z_{1:n})$ is terminated after time $k^{\delta/\gamma} g(n)$. Any $\delta > 0$ and $\gamma > 0$ will do. Therefore (T^o_k) enumerates all time-bounded machines.

The idea of the following construction is to return an $\dot{x}^s_{1:s}$ that is not in any effective quasi-sparse set of the form

$$L^{n,o}_k := \begin{cases} \tilde{L}^{n,o}_k & \text{if} \quad |\tilde{L}^{n,o}_k| \leq 2^{f(s(n))} \\ \{\} & \text{else} \end{cases} , \quad \text{where} \quad \tilde{L}^{n,o}_k := \{z \in \mathbb{B}^n : T^o_k(z) = 1\}$$

and $f(s)$ is some (linear/logarithmic) monotone increasing function and $s(n)$ is some injective (linear/exponential) monotone increasing function.

Constructing quasi-random sequences $\dot{x}^s_{1:s}$. For the construction to work, $\dot{x}^s_{1:s}$ should also not be probed by any fast algorithm on any input. Since the algorithms can probe oracle $o = (q_s)$ before q_s has been constructed, we need a careful construction in stages $s = 1, 2, 3, ...$ Assume $\dot{x}^{s'}_{1:s'}$ and $q_{s'}$ have already been constructed for all $s' < s$. We now construct $\dot{x}^s_{1:s}$. For this we define a fake oracle o_s that coincides with o whenever queried with a string of length less than s (the already constructed $q_{s'}$), but always returns 0 when queried with a string of length s or larger (for which $q_{s'}$ has yet to be constructed). Let $n := n(s)$ and

$$C_{\geq s} \ := \ \{y_{1:s'} : s' \geq s, \exists z_{1:n} \exists k \leq n^\gamma : T^{o_s}_k(z_{1:n}) \text{ calls } o_s \text{ on } y_{1:s'}\}$$

be the set of sequences $y_{1:s'}$ longer or equal than s (this is important) that are queried by any of the first n^γ (any $\gamma > 0$ will do) Turing machines $T^{o_s}_k$ on any input $z_{1:n}$. Now let

$$F_s \;:=\; \mathbb{B}^s \setminus \Big(\bigcup_{k=1}^{n^\gamma} L_k^{n,o_s} \times \mathbb{B}^{s-n} \cup \bigcup_{s'=1}^{s} C_{\geq s'} \Big)$$

be the set of strings of length s that roughly (i) are not queried and (ii) whose length n prefix is not in any quasi-sparse set. If $F_s \neq \{\}$,

$$\text{let } \dot{x}_{1:s}^s \text{ be the lexicographically first string in } F_s$$

If $F_s = \{\}$, arbitrarily let $\dot{x}_{1:s}^s = 0_{1:s}$. In any case define $q_s(\dot{x}_{1:s}^s) := 1$, and 0 on all other sequences of length s.

Fast good $\tilde{q}^o$ implies $F_s = \{\}$. Let $\tilde{q}^o$ be an online (=TC) estimator with access to oracle $o = (q_s)$ and small regret

$$R_n(\tilde{q}^o || q_n) \;\equiv\; \max_{x_{1:n}} \ln \frac{q_n(x_{1:n})}{\tilde{q}^o(x_{1:n})} \;<\; f(n) \ln 2 \quad \text{for all large } n \tag{18}$$

This implies $\tilde{q}^o(x_{1:s}) > 2^{-f(s)} q_s(x_{1:s}) \; \forall x_{1:s} \forall' s$ and in particular $\tilde{q}^o(\dot{x}_{1:s}^s) > 2^{-f(s)}$. Since $\tilde{q}^o$ is TC, we have $\tilde{q}^o(\dot{x}_{1:n}^s) \geq \tilde{q}^o(\dot{x}_{1:s}^s) > 2^{-f(s(n))}$.

Now let us assume that $\tilde{q}^o \in \text{TIME}^o(g(n))$. Then membership in

$$L^{n,o} \;:=\; \{x_{1:n} : \tilde{q}^o(x_{1:n}) > 2^{-f(s)}\}$$

can be determined in the same (or less) time and since $\tilde{q}^o$ is a probability, $|L^{n,o}| < 2^{f(s)}$. Therefore, there is a k_0 such that $T_{k_0}^o$ computes $L^{n,o} = L_{k_0}^{n,o}$.

Now assume $F_s \neq \{\}$ for some $n^\gamma \geq k_0$. The construction of $\dot{x}_{1:s}^s$ is such that $T_k^{o_s}(z_{1:n}) = T_k^o(z_{1:n})$ for all $k \leq n^\gamma$ and all $z_{1:n}$, since their oracles coincide on all queried strings $y_{1:s'}$: For $y_{1:s'} \neq \dot{x}_{1:s'}^{s'}$ both oracles answer 0. For $s' < s$ both oracles coincide also on $\dot{x}_{1:s'}^{s'}$, and for $s' \geq s$ any queried string is added to the tabu list $C_{\geq s}$ and the choice of $\dot{x}_{1:s'}^{s'}$ outside $\bigcup_{s'=1}^{s} C_{\geq s'}$ ensures it has also not been queried earlier in the construction. So o also returns 0 for $s' \geq s$ on all queried strings.

In particular $T_{k_0}^{o_s}(z_{1:n}) = T_{k_0}^o(z_{1:n})$, hence $L_{k_0}^{n,o} = L_{k_0}^{n,o_s}$. Further, $\dot{x}_{1:s}^s \in F_s$ implies prefix $\dot{x}_{1:n}^s \notin L_{k_0}^{n,o_s}$ by definition of F_s. We conclude $\dot{x}_{1:n}^s \notin L^{n,o}$, which clearly contradicts $\tilde{q}^o(\dot{x}_{1:n}^s) > 2^{-f(s)}$. Therefore, $\tilde{q}^o \in \text{TIME}^o(g(n))$ implies $F_s = \{\}$ for all large s.

$F_s = \{\}$ implies slow good $\tilde{q}^o$. $t(n) := n^\delta g(n)$ upper bounds the running time of T_k^o for $k \leq n^\gamma$. It also bounds the number of oracle calls in T_k^o, since each oracle call costs at least one step. Note that $C_{\geq s'} \subseteq C_{\geq s''}$ if $s' \geq s''$ and $n(s') = n(s'')$, which implies $\bigcup_{s'=1}^{s} C_{\geq s'} = \bigcup_{n'=1}^{n} C_{\geq s(n')}$ due to $s(n(s)) \leq s$. Using

$$|L_k^{n,o_s}| \leq 2^{f(s)} \quad \text{and} \quad |C_{\geq s}| \leq n^\gamma\, 2^n t(n) \quad \Rightarrow \quad \Big| \bigcup_{n'=1}^{n} C_{\geq s(n')} \Big| \leq n^\gamma\, 2^{n+1} t(n)$$

$$\text{implies} \quad |F_s| \;\geq\; 2^s - n^\gamma\, 2^{f(s)} 2^{s-n} - n^\gamma 2^{n+1} t(n)$$

$$\text{hence} \quad F_s = \{\} \quad \text{implies} \quad 2t(n) \geq 2^{s-n}[n^{-\gamma} - 2^{f(s)-n}] \;=:\; h(n)$$

This contradicts the assumption on $g(n)$ in the theorem, hence $F_s \neq \{\}$, hence $\tilde{q}^o \notin \text{TIME}^o(g(n))$ for all $\tilde{q}^o$ with regret (18), whose contrapositive is

$$\forall \tilde{q}^o \in \text{TIME}^o(g(n)) : R_n(\tilde{q}^o||q_n) \geq f(n)\ln 2 \;\forall' n$$

Complexity of (q_s). The construction of q_s requires running $T_k^{o_s}(z_{1:n})$ for all $z_{1:n}$ for all $k \leq n^\gamma$, each requiring $k^{\delta/\gamma} g(n) \leq t(n)$ steps. Hence $q_s \in \text{TIME}^{o_s}(n^\gamma 2^n t(n))$ where $n = n(s)$. We can get rid of the self-reference to oracle o_s by considering the complexity of the iterative construction of $q_{s'}$ and $\dot{x}_{1:s'}^{s'}$ for all $s' \leq s$.

Assume we have constructed and stored $\dot{x}_{1:s'}^{s'}$ for all $s' < s$. We construct $\dot{x}_{1:s}^s$ as follows: First note that oracle o_s in $T_k^{o_s}$ can be eliminated. If queried for $y_{1:s'}$ for $s' < s$ we simply return 1 iff $y_{1:s'} = \dot{x}_{1:s'}^{s'}$, which can be done in time $s' \leq s$, since $\dot{x}_{1:s'}^{s'}$ has been pre-computed and stored. If o_s is queried for $y_{1:s'}$ for $s' \geq s$ the answer was defined to be 0, clearly computable in time s.

To efficiently compute $\dot{x}_{1:s}^s$, we first construct $U_n := \bigcup_{k=1}^{n^\gamma} L_k^{n,o_s}$ in time $O(n^\gamma 2^n t(n) s)$. We now make a list of the lexicographically first $\min\{2^s, n^\gamma 2^{n+1} t(n) + 1\}$ strings of length s whose length n prefix is *not* in U_n. Next we cross out all strings queried in the definition of $C_{\geq s(n')}$ for all $1 \leq n' \leq n$ in time $O(n^\gamma 2^{n+1} t(n) s)$. The lexicographically first string left over can be found in time $O(n^\gamma 2^{n+1} t(n) s)$ and will be $\dot{x}_{1:s}^s$. Since $|\bigcup_{n'=1}^n C_{\geq s(n')}| \leq n^\gamma 2^{n+1} t(n)$ at least one string survived elimination, except $F_s = \{\}$, in which case $\dot{x}_{1:s}^s = 0_{1:s}$. This shows that $q_s \in \text{TIME}(n^\gamma 2^n t(n) s)$ where $n = n(s)$. This construction assumed a random access machine (RAM). For other machines, some extra powers of s may be needed with marginal effect on the results. So in general

$$(q_s) \in \text{TIME}(n(s)^{\gamma+\delta} 2^{n(s)} g(n(s)) s^m) \qquad \blacksquare$$

Proof of Theorem 4. In Theorem 6, weaken $\text{TIME}^o \rightsquigarrow \text{TIME}$ and $\tilde{q}^o \rightsquigarrow \tilde{q}$, and let $s = 2^{(1-\varepsilon)n/r}$ and $f(s) = r\log s = (1-\varepsilon)n$. Then $h(n) = 2^{2^{(1-\varepsilon)n/r} - n}[n^{-\gamma} - 2^{-\varepsilon n}]$, so clearly $g(n) := 2^{cn} < \frac{1}{2} n^{-\delta} h(n)$ for large n. For any $\varepsilon > 0$ and sufficiently large n we have

$$\begin{aligned} n^{\gamma+\delta} 2^n g(n) s^m &= s^m 2^{(c+1)n + (\gamma+\delta)\log n} \\ &\leq s^m 2^{(c+1+\varepsilon)n} = s^m 2^{(c+1+\varepsilon)\frac{r\log s}{1-\varepsilon}} = s^{m + \frac{c+1+\varepsilon}{1-\varepsilon} r} = s^{b+m} \end{aligned}$$

This proves (i). (ii) is just a weaker version of (i) since $\text{TIME}(s^{b+m}) \subset \text{P}$. (iii) follows from the fact that $b := r + \varepsilon'$ implies $c > 0$ for sufficiently small $\varepsilon > 0$, and $\text{E}^c \supset \text{P}$. (iv) follows from (i) and the fact that $R_n(\tilde{q}^{\text{mix}}) \leq 2\ln(n+2) < r\ln n \,\forall' n$ for any $r > 2$. $\blacksquare$

Proof of Theorem 5. In Theorem 6 let $s = (1+\varepsilon)n$ and $f(s) = (1-\varepsilon)s$. Then $h(n) = 2^{\varepsilon n}[n^{-\gamma} - 2^{-\varepsilon^2 n}]$, so clearly $g(n) := 2^{\varepsilon n/2} < \frac{1}{2} n^{-\delta} h(n)$ for large n. For any $\varepsilon > 0$ and sufficiently large n we have $n^{\gamma+\delta} 2^n g(n) s^m = (1+\varepsilon)^m 2^{n + \varepsilon n/2 + (\gamma+\delta+m)\log n} \leq 2^{(1+\varepsilon)n} = 2^s$. $\blacksquare$

7 Open Problems

We now discuss and quantify the problems that we raised earlier and are still open. For some *specific* collection (q_n) of probabilities, does there exist a polynomial-time computable time-consistent $\tilde{q}$ with $R_n(\tilde{q}||q_n) \leq 2\ln(n+2)\,\forall n$? Note that $\tilde{q}^{\text{mix}}$ satisfies the bound, but a direct computation is prohibitive. So one way to a positive answer could be to find an efficient approximation of $\tilde{q}^{\text{mix}}$. If the answer is negative for a specific (q_n) one could try to weaken the requirements on R_n. We have seen that for some, (non-TC) (q_n), namely Ristad's, simple normalization $\tilde{q}^{\text{n1}}$ solves the problem.

A concrete unanswered example are the triple uniform Good-Turing probabilities (q_n). Preliminary experiments indicate that they and therefore $\tilde{q}^{\text{mix}}$ are more robust than current heuristic smoothing techniques, so a tractable approximation of $\tilde{q}^{\text{mix}}$ would be highly desirable. It would be convenient and insightful if such a $\tilde{q}$ had a traditional GT representation but with a smarter smoothing function $m_{()}$.

The nasty (q_n) constructed in the proof of Theorem 6 is very artificial: It assigns extreme probabilities (namely 1) to quasi-random sequences. It is unknown whether there is any offline estimator of practical relevance (such as Good-Turing) for which no fast online estimator can achieve logarithmic regret.

An open problem for general (q_n) is as follows: Does there exist for every (q_n) a polynomial-time algorithm that computes a time-consistent $\tilde{q}$ with $R_n(\tilde{q}||q_n) \leq f(n)\,\forall n$. We have shown that this is not possible for $f(n) = O(\log n)$ and not even for $f(n) = (1-\varepsilon)n\ln 2$ if $\tilde{q}$ has only oracle access to (q_n). This still allows for a positive answer to the following open problem:

Open Problem 7 (Fast online from offline with small extra regret) *Can every polynomial-time offline estimator (q_n) be converted to a polynomial-time online estimator $\tilde{q}$ with small regret $R_n(\tilde{q}||q_n) \leq \sqrt{n}\,\forall' n$? Or weaker: $\forall (q_n) \in P\, \exists \tilde{q} \in P : R_n = o(n)$? Or stronger: $\forall (q_n) \in P\, \exists \tilde{q} \in P : R_n = O(\log n)^2$?*

A positive answer would reduce once and for all the problem of finding good online estimators to the apparently easier problem of finding good offline estimators. We could also weaken our notion of worst-case regret to e.g. expected regret $\mathbb{E}[\ln(q_n/\tilde{q})]$. Expectation could be taken w.r.t. (q_n), but other choices are possible. Other losses than logarithmic also have practical interest, but I do not see how this makes the problem easier.

Ignoring computational considerations, of theoretical interest is whether $O(\log n)$ is the best one can achieve in general, say $\exists q_n \forall \tilde{q} : R_n(\tilde{q}) \geq \ln n$, or whether a constant is achievable.

Devising general techniques to upper bound $R_n(\tilde{q}^{\text{n1}}||q_n)$, especially if small, is of interest too.

Acknowledgements. Thanks to Jan Leike for feedback on earlier drafts.

References

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[AS74] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions.* Dover publications, 1974.

[BC91] A. R. Barron and T. M. Cover. Minimum complexity density estimation. *IEEE Transactions on Information Theory*, 37:1034–1054, 1991.

[CG99] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, 1999.

[Goo53] I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3/4):237–264, 1953.

[Grü07] P. D. Grünwald. *The Minimum Description Length Principle.* The MIT Press, Cambridge, 2007.

[Hut03] M. Hutter. Optimality of universal Bayesian prediction for general loss and alphabet. *Journal of Machine Learning Research*, 4:971–1000, 2003.

[Hut05] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability.* Springer, Berlin, 2005.

[Hut09] M. Hutter. Discrete MDL predicts in total variation. In *Advances in Neural Information Processing Systems 22 (NIPS'09)*, pages 817–825, Cambridge, MA, USA, 2009. Curran Associates.

[LH14] J. Leike and M. Hutter. Indefinitely oscillating martingales. *Technical report: http://www.hutter1.net/publ/martoscx.pdf*, 2014.

[Nad85] A. Nadas. On Turing's formula for word probabilities. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(6):1414–1416, 1985.

[PH05] J. Poland and M. Hutter. Asymptotics of discrete MDL for online prediction. *IEEE Transactions on Information Theory*, 51(11):3780–3795, 2005.

[RH07] D. Ryabko and M. Hutter. On sequence prediction for arbitrary measures. In *Proc. IEEE International Symposium on Information Theory (ISIT'07)*, pages 2346–2350, Nice, France, 2007. IEEE.

[Ris95] E. S. Ristad. A natural law of succession. Technical Report CS-TR-495-95, Princeton University, 1995.

[San06] N. Santhanam. *Probability Estimation and Compression Involving Large Alphabets.* PhD thesis, Univerity of California, San Diego, USA, 2006.

[Sol78] R. J. Solomonoff. Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transactions on Information Theory*, IT-24:422–432, 1978.

A Proof of Theorem 1

For GT we prove $\max_{x_{1:n}}\mathcal{N}_n\to 2$, therefore $\max_{x_{1:n}}\mathcal{N}(x_{1:n})\to 2$ due to $\frac{\text{Part}(n)}{\text{Part}(n+1)}\to 1$ for $n\to\infty$. We can upper bound (14) as

$$\begin{aligned}(n+1)\mathcal{N}_n &= \sum_{r=0,m_r\neq 0}^{n}(r+1)m_{r+1}+\sum_{r=0,m_r\neq 0}^{n} r \;+\sum_{r=0,m_r\neq 0}^{n} 1\\ &\leq \sum_{r'=1}^{n+1} r'm_{r'}+\sum_{r=0}^{n} rm_r+|\{r:m_r\neq 0\}|\\ &= n+n+|\{r:m_r\neq 0\}| \;\leq\; 2n+\sqrt{2n}+1\end{aligned}$$

$|\{r:m_r\neq 0\}|$ under the constraint $\sum_{r=0}^n rm_r=n$ is maximized for $m_0=...=m_k=1$ and $m_{k+1}=...=m_n=0$ for suitable k. We may have to set one $m_r=2$ to meet the constraint. Therefore $n=\sum_{r=0}^n rm_r\geq\sum_{r=0}^k r=\frac{k(k+1)}{2}\geq\frac{1}{2}k^2$, hence $|\{r:m_r\neq 0\}|=k+1\leq\sqrt{2n}+1$.

For the lower bound we construct a sequence that attains the upper bound. For instance, $x_{1:k(k+1)/2}=1223334444\ ...\ k...k$ has $m_1=...=m_k=1$, hence $x_{1:\infty}=1223334444...$ has $m_1\geq 1,...,m_k\geq 1$ for all $n\geq\frac{1}{2}k(k+1)$. Conversely, for any n we have $m_1\geq 1,...,m_k\geq 1$ with $k:=\lfloor\sqrt{2n}\rfloor-1$. For the chosen sequence we therefore have

$$(n+1)\mathcal{N}_n \;\geq\; \sum_{r=0}^{k-1}(r+1)(1+1) \;=\; k(k+1) \;\geq\; 2n-3\sqrt{2n}$$

The upper and lower bounds together imply $\max_{x_{1:n}}\mathcal{N}_n=2\pm O(n^{-1/2})$, therefore $\max_{x_{1:n}}\mathcal{N}(x_{1:n})=2\pm O(n^{-1/2})$ due to $\frac{\text{Part}(n)}{\text{Part}(n+1)}=1-O(n^{-1/2})$ [AS74]. Inserting this into (15) gives $R_n=n\ln 2\pm O(n^{-1/2})$.

The upper bound holds for any d, but the lower bound requires $d=\infty$ or at least $d\geq\sqrt{2n}$. We now show linear growth of R_n even for finite $d\geq 3$. The lower bound is based on the same sequence as used in [San06]: For $x_{1:\infty}=12(132)^\infty$ elementary algebra gives $\mathcal{N}_n=\frac{5}{3}+\frac{7/3}{n+1}$ and $\mathcal{N}_{n+1}=\frac{5}{3}+\frac{5/3}{n+2}$ and $\mathcal{N}_{n+2}=\frac{4}{3}+\frac{1}{n+3}$ for n a multiple of 3, hence $\mathcal{N}_n\mathcal{N}_{n+1}\mathcal{N}_{n+2}\geq\frac{100}{27}$ (except $\mathcal{N}_0\mathcal{N}_1\mathcal{N}_2=\frac{2}{3}$). Together with asymptotics $\ln(\text{Part}(n))\sim\pi\sqrt{2n/3}$ [AS74], this implies that $R_n\geq\frac{n}{3}\ln\frac{100}{27}-O(\sqrt{n})$. ■

B List of Notation

Symbol	**Explanation**
$\equiv$	identical, equal by definition, trivially equal
$\binom{n}{n_1...n_d}$	multinomial
$n\in\mathbb{N}_0$	length of sequence
$t\in\{1,...,n\}$	current "time"
$s\in\mathbb{N}$	any "time"
$\mathcal{X}=\{1,...,d\}$	finite alphabet, $d>1$
$i,x,x_t\in\mathcal{X}$	symbol
$x_{t:n}\in\mathcal{X}^{n-t+1}$	sequence $x_t...x_n$
$x_{<t}\in\mathcal{X}^{t-1}$	sequence of length $t-1$
$\epsilon=x_{1:0}=x_{<1}$	empty string
Q	any measure on $\mathcal{X}^\infty$
$q_n:\mathcal{X}^n\to[0;1]$	offline estimated probability mass function
$\bar{q}_s:\mathcal{X}^*\to[0;1]$	extends q_n to any TC probability on $\mathcal{X}^*$
$\tilde{q}:\mathcal{X}^*\to[0;1]$	online estimator desired to be close to q_n
$\tilde{q}_{\lvert\mathcal{X}^n}$	constrains the domain of $\tilde{q}$ to $\mathcal{X}^n$
log, ln	binary and natural logarithms, respectively
$\text{TIME}^o(g(n))$	algorithms that run in time $O(g(n))$ with access to oracle o
$\text{P}:=\bigcup_{k=1}^\infty\text{TIME}(n^k)$	polynomial time algorithms
$\text{E}^c:=\text{TIME}(2^{cn})$	exponential time algorithms (much smaller than EXP or even E!)
$\mathbb{B}:=\{0,1\}$	binary alphabet
$\forall' n$	for all but finitely many n, short 'for all large n'
quasi	akin to but not necessarily an established definition