Formalisms for Sequence Modeling

Jason Eisner — JHU 601.765

January 28, 2019

This class aims to present many methods in a unified way. To highlight their connections, we will adopt some shared notation and terminology, and stick to them as much as possible. This document serves as a concise introduction and reference to our notation, technical vocabulary, and formalisms. It may be extended or revised over the course of the class. Refer to it whenever you are disoriented. Post any questions or suggestions on Piazza.

In order to remain concise, this document only provides definitions. It does not discuss the properties or applications of the defined objects, give examples, or present algorithms for working with them.

1 Typographic conventions (just so you know)

Technical terms are marked with a blue background when they are introduced. Terms that are introduced in the footnotes are provided there for reference but will generally not be used further.

We consistently use italic font for real-valued variables, across all alphabets:[1] scalars s, λ, vectors $\boldsymbol{s}, \boldsymbol{\lambda}$ in boldface, and matrices $\boldsymbol{S}, \boldsymbol{\Lambda}$ in uppercase boldface. Uppercase scalars such as F, G usually represent aggregate quantities.

Integer indices into sequences, vectors, and matrices are also written as italic lowercase letters, typically i, j, k. In this case the corresponding uppercase letter denotes the upper bound. For example, a vector $\boldsymbol{h} \in \mathbb{R}^I$ and a matrix $\boldsymbol{W} \in \mathbb{R}^{I \times J}$ respectively consist of elements h_i and w_{ij} for $1 \le i \le I, 1 \le j \le J$.

The font of a function symbol is determined by the type of the returned value, e.g., $\boldsymbol{h}(t)$ is a different *vector* for each t, with j^{th} element $h_j(t) \in \mathbb{R}$. Analogously, boldface $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots$ denotes a sequence of *vectors*, with the jth element of vector $\boldsymbol{h}_i$ written as $h_{ij} \in \mathbb{R}$.

We use upright font for variables that range over a discrete, unordered set: y, ω denote symbols (such as letters or words), while $\mathbf{y}, \boldsymbol{\omega}$ denote strings (finite sequences of symbols, such as words or sentences).

For literal symbols, we use typewriter font: `a`, `b`. Strings are written by concatenation: `abc`. For literal symbols with multi-character names, we instead use small caps, and use whitespace to set them off from adjacent symbols in a string: thus BOS `abc` EOS is a string of 5 symbols, while HELLO WORLD is a string of 2 symbols.

The string $\mathbf{y} = \mathrm{y}_1 \cdots \mathrm{y}_J$ for some $J \in \mathbb{N}$.[2] For convenience, for any $0 \le i \le j \le J$, we define $\mathbf{y}_{i:j} \stackrel{\text{def}}{=} \mathrm{y}_{i+1} \cdots \mathrm{y}_j$ (a substring of length $j-i$), $\mathbf{y}_{:j} \stackrel{\text{def}}{=} \mathbf{y}_{0:j} = \mathrm{y}_1 \cdots \mathrm{y}_j$ (the prefix of length j), and $\mathbf{y}_{j:} \stackrel{\text{def}}{=} \mathbf{y}_{j:J} = \mathrm{y}_{j+1} \cdots \mathrm{y}_J$ (the suffix of length $J-j$).

In general we need to declare a variable's set of possible values (its type). When this set is fixed by the problem setting, we use calligraphic uppercase: $\mathrm{y} \in \mathscr{Y}, \mathbf{y} \in \boldsymbol{\mathscr{Y}}$. Sets that are fixed across mathematics are written in blackboard bold (the natural numbers $\mathbb{N}$, the real numbers $\mathbb{R}$), so we might declare $\boldsymbol{s} \in \mathbb{R}^D$

[1] As recommended for physical quantities by international standard ISO 80000-2. To this end, consistent fonts are arranged via the LaTeX `isomath` package.

[2] Note that this convention differs from (say) Python, where the first character of $\mathbf{y}$ is called y_0 rather than y_1, and where $\mathbf{y}_{i:j} \stackrel{\text{def}}{=} \mathrm{y}_i \cdots \mathrm{y}_{j-1}$ rather than $\mathrm{y}_{i+1} \cdots \mathrm{y}_j$.

for a vector. When random variables are made explicit, they are generally written as the corresponding uppercase letter in the same font, e.g., $\mathrm{y}, \mathbf{y}, \boldsymbol{\lambda}$ might denote possible values of random variables $\mathrm{Y}, \mathbf{Y}, \boldsymbol{\Lambda}$.

2 Input and output

This class focuses on conditional probability distributions $p(\mathbf{y} \mid \mathbf{x})$. That is, we are interested in modeling the distribution of the output $\mathbf{y}$ given a fully observed input $\mathbf{x}$.[3]

In general $\mathbf{x}$ and $\mathbf{y}$ will be structured objects. $\mathscr{X}$ denotes the input space of possible $\mathbf{x}$ values. For each input $\mathbf{x}$, the distribution $p(\mathbf{y} \mid \mathbf{x})$ is a function $\mathscr{Y}_{\mathbf{x}} \to \mathbb{R}_{\geq 0}$ where $\mathscr{Y}_{\mathbf{x}}$ is the output space that we will consider computationally.

The methods in the class can be generalized to many structured prediction problems. However, our *canonical problem* takes $\mathbf{x}$ to be a string of length J and allows $\mathbf{y}$ to range over strings of the same length J. Then $\mathbf{x} \in \mathscr{X} \subseteq \mathscr{X}^*$ where $\mathscr{X}$ is a fixed input alphabet, and $\mathbf{y} \in \mathscr{Y}_{\mathbf{x}} \subseteq \mathscr{Y}^J$ where $J = |\mathbf{x}|$ and $\mathscr{Y}$ is a fixed output alphabet.[4]

Note that we did not assume identical input and output alphabets ($\mathscr{X} = \mathscr{Y}$), nor did we assume that the output space includes all length-J strings over the output alphabet ($\mathscr{Y} = \mathscr{Y}^J$), although these are common cases.

Question 2.1 *How would you do language modeling?*

Question 2.2 *How would you do BIO tagging?*

3 Observations

A training or test example is a pair $\langle \mathbf{x}, \mathscr{Y} \rangle$ where $\mathscr{Y} \subseteq \mathscr{Y}_{\mathbf{x}}$ is an observation of the output—that is, a non-empty set that is known to contain the output string.

- A fully observed output has the form $\mathscr{Y} = \{\mathbf{y}\}$ (a singleton set). In this case we may abbreviate the example as $\langle \mathbf{x}, \mathbf{y} \rangle$.
- An unobserved output is $\mathscr{Y} = \mathscr{Y}_{\mathbf{x}}$, which provides no information about $\mathbf{y}$.
- A partially observed output is anything in between.

Note that the input $\mathbf{x}$ is always fully observed. In general,

- training outputs will be fully or partially observed, so that they *contain* information that can be learned from;
- test outputs will be unobserved or partially observed, so that they *omit* information that must be predicted.

We will often assume that examples are presented in the alternative form $\langle \mathbf{x}, \mathbf{o} \rangle$, where the observation $\mathbf{o}$ is now specified as a string of length J. What set $\mathscr{Y}$ does $\mathbf{o}$ describe? Each observed symbol o_j is an observation of the output symbol y_j, meaning that it is compatible with some set of possibilities for y_j, denoted $\mathscr{Y}_{\mathrm{o}_j} \subseteq \mathscr{Y}$. Thus, observing the string $\mathbf{o}$ is equivalent to observing the set $\mathscr{Y}_{\mathbf{x},\mathbf{o}} \stackrel{\text{def}}{=} \mathscr{Y}_{\mathbf{x}} \cap (\mathscr{Y}_{\mathrm{o}_1} \cdots \mathscr{Y}_{\mathrm{o}_J})$.

[3] The use of x for input and y for output has become common in machine learning.

[4] In the literature of formal languages and transducers, the alphabets $\mathscr{X}, \mathscr{Y}$ are conventionally called Σ, Δ.

As a special case, it is often convenient to suppose that each output symbol $\mathrm{y} \in \mathscr{Y}$ is an ordered pair $\langle \mathrm{o}, \mathrm{v} \rangle$, where v is a latent symbol that specifies the unobserved portion of y. Thus o is compatible with $\mathscr{Y}_{\mathrm{o}} = \{\langle \mathrm{o}, \mathrm{v} \rangle : \mathrm{v} \in \mathscr{V}\}$. Then given the observation $\mathbf{o}$, predicting the complete string $\mathbf{y}$ is equivalent to predicting its unobserved portion—the latent string $\mathbf{v}$.[5]

Question 3.1 *If $\mathscr{Y} \subseteq \mathscr{Y}_{\mathbf{x}}$ is a regular language, then how can it be described using $\mathbf{o}$?*

4 Decision theory: Actions and rewards

We are concerned with building agents—systems that make decisions. Given an input $\mathbf{x}$ (and perhaps a *partially* observed output $\mathbf{o}$), an agent may decide how to act based on what it thinks $\mathbf{y}$ is. We will allow the agent to choose a string $\mathbf{a}$ of length J, in some set $\mathscr{A}_{\mathbf{x}} \subseteq \mathscr{A}^J$.

An agent that is designed to predict $\mathbf{y}$ from $\mathbf{x}$ takes $\mathscr{A}_{\mathbf{x}} = \mathscr{Y}_{\mathbf{x}}$ and is trained to choose $\mathbf{a} \approx \mathbf{y}$. Other agents might be designed to predict only $\mathbf{y}$'s latent string $\mathbf{v}$ or $\mathbf{y}$'s observable string $\mathbf{o}$. In all these cases, $\mathbf{a}$ is called a prediction. In general, however, an agent can be designed to choose any real-world response to $\mathbf{x}$: so in general we refer to $\mathbf{a}$ as a plan consisting of the actions $\mathrm{a}_1, \ldots, a_J$.

Some predictions are more accurate than others, and some plans work better than others. The reward function[6] $R(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$ evaluates the quality of the prediction or plan $\mathbf{a}$ given that the true output is $\mathbf{y}$. The reward function is specified externally: it defines what kinds of predictions or other actions are valuable for the applied task.

It is often useful to break down the total reward as a sum $R = r_1 + \ldots + r_J$, where r_j is regarded as the reward of the individual action or predicted symbol a_j (possibly in context).

Question 4.1 *How would you use this framework if the agent simply needs to decide whether $\mathbf{x}$ is "spam"? How would you penalize precision and recall errors?*

5 Probability distributions

We now turn to probabilistic modeling, which is often useful in making decisions.

$p^*(\mathbf{x}, \mathbf{y})$ denotes the true joint distribution over $\langle \mathbf{x}, \mathbf{y} \rangle$ pairs. The test examples on which we will have to choose actions will be drawn from this unknown distribution.

$p^*(\mathbf{y} \mid \mathbf{x})$ or more generally $p^*(\mathbf{y} \mid \mathbf{x}, \mathbf{o})$ is known as the posterior predictive distribution (given evidence $\mathbf{x}$ or $\mathbf{x}, \mathbf{o}$ respectively). It is fully determined by $p^*(\mathbf{x}, \mathbf{y})$.

$p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ or $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}, \mathbf{o})$ is an estimated conditional distribution, which the agent may consult in order to make a decision on a test input $\mathbf{x}$. We will focus on estimating $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ since that fully determines $p^*(\mathbf{y} \mid \mathbf{x}, \mathbf{o})$ for any $\mathbf{o}$.[7]

[5] Generalizing slightly, we can get the same effect without actually requiring $\mathscr{Y}$ to consist of ordered pairs. It suffices to say that for every possible observed symbol o, there is a bijection between the possible complete symbols $\mathscr{Y}_{\mathrm{o}}$ and their possible latent parts $\mathscr{V}_{\mathrm{o}}$. In other words, there exist functions $\mathrm{y}(), \mathrm{v}()$ so that it follows that $\mathrm{y}_j = \mathrm{y}(\mathrm{o}_j, \mathrm{v}_j)$ once we know v_j, and moreover $\mathrm{v}_j = \mathrm{v}(\mathrm{o}_j, \mathrm{y}_j)$ is the only choice for v_j that would yield that y_j. *Remark:* It is important that v_j be uniquely determined by y_j (that is, y_j does not lose any information from $\mathbf{v}_j$). It means that when we sum over the possible outputs $\mathbf{y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}}$ to get $p(\mathbf{o} \mid \mathbf{x})$, we do not have to sum over possible $\mathbf{v}$ values as well. Only one $\mathbf{v}$ value is compatible with each output $\mathbf{y}$.

[6] Loss is the negative of reward. Decision theory and machine learning traditionally specify loss functions, whereas reinforcement learning papers usually specify reward functions. I opted to use reward functions so that we are consistently solving maximization problems—maximizing log-probability while training, and maximizing expected reward when predicting.

[7] $p^*(\mathbf{y} \mid \mathbf{x}, \mathbf{o})$ could be written more precisely using random variables as $p^*(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x}, \mathbf{Y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}})$. It is proportional to $p^*(\mathbf{y} \mid \mathbf{x})$ for $\mathbf{y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}}$ and is 0 for all other $\mathbf{y}$. Proof: $p^*(\mathbf{y} \mid \mathbf{x}, \mathbf{o}) = \frac{p^*(\mathbf{o}, \mathbf{y} \mid \mathbf{x})}{p^*(\mathbf{o} \mid \mathbf{x})} \propto p^*(\mathbf{o}, \mathbf{y} \mid \mathbf{x}) = p^*(\mathbf{y} \mid \mathbf{x}) \cdot p^*(\mathbf{o} \mid \mathbf{x}, \mathbf{y})$, where $p^*(\mathbf{o} \mid \mathbf{x}, \mathbf{y}) = p^*(\mathbf{Y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}} \mid \mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = p^*(\mathbf{y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}} \mid \mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y})$ is 1 or 0 according to whether $\mathbf{y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}}$.

$\mathscr{P} = \{p_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ is our parametric model: the family of conditional distributions $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ that we will consider. After specifying $\mathscr{P}$, we will try to estimate a value of the parameter vector $\boldsymbol{\theta}$ such that the distribution $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ tends to be a useful approximation to $p^*(\mathbf{y} \mid \mathbf{x})$.[8]

$\hat{p}(\mathbf{x}, \mathbf{y})$ refers to the empirical distribution when all training examples are fully observed. That is, $\hat{p}(\mathbf{x}, \mathbf{y}) = \frac{c(\mathbf{x},\mathbf{y})}{n}$ where $c(\mathbf{x}, \mathbf{y})$ denotes the number of occurrences of $(\mathbf{x}, \mathbf{y})$ in the training dataset.

$q_{\boldsymbol{\phi}}(\cdots)$ is our standard name for any distribution—such as a proposal distribution or a variational approximation—that we temporarily construct within an algorithm for computational reasons. We typically select such a distribution $q_{\boldsymbol{\phi}}$ from a variational family $\mathscr{Q} = \{q_{\boldsymbol{\phi}} : \boldsymbol{\phi} \in \boldsymbol{\Phi}\}$ of computationally tractable distributions that we have designed.

Outside this handout, I may abbreviate $p_{\boldsymbol{\theta}}, q_{\boldsymbol{\phi}}$ as p, q.

6 Information theory

Given a probability distribution p over the discrete set $\mathscr{Y}$, the following quantities are useful:

- The surprisal of p on data $\mathbf{y}$: $-\log p(\mathbf{y}) \in [0, \infty]$.
- The support of p: the set $\{\mathbf{y} \in \mathscr{Y} : p(\mathbf{y}) > 0$.
- The entropy $H(p) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{y}\sim p}[-\log p(\mathbf{y})] \in [0, \infty)$. This is the expected surprisal of p on data sampled from p. We can write the expectation explicitly as $\sum_{\mathbf{y}\in\mathscr{Y}:p(\mathbf{y})>0} p(\mathbf{y})(-\log p(\mathbf{y}))$ to avoid summands of the form $0 \log 0$ (which are omitted on the grounds that $z \log z \to 0$ as $z \to 0$).

If p is a model, we may be interested in other distributions q also over $\mathscr{Y}$. For example, q might be the empirical distribution of data, or a simplified model that is more computationally tractable. The following quantities are useful for relating p to q. Note that the expectation is always taken under the distribution that is given first.

- The cross-entropy $H(q, p) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{y}\sim q}[-\log p(\mathbf{y})] \in [0, \infty]$. This is the expected surprisal of p on data sampled from q.
- The exclusive Kullback-Leibler divergence[9] (KL divergence from q to p) is $\mathrm{KL}(q \mid\mid p) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{y}\sim q}[\frac{\log q(\mathbf{y})}{\log p(\mathbf{y})}] = H(q, p) - H(q) \geq 0$. This equals 0 iff $q = p$, and is finite iff $\text{support}(q) \subseteq \text{support}(p)$.
- The inclusive Kullback-Leibler divergence (KL divergence from p to q) is $\mathrm{KL}(p \mid\mid q) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{y}\sim p}[\frac{\log p(\mathbf{y})}{\log q(\mathbf{y})}] = H(p, q) - H(p) \geq 0$. This equals 0 iff $p = q$, and is finite iff $\text{support}(p) \subseteq \text{support}(q)$.

The fact that KL divergences ≥ 0 will be proved in **??**.

The above formulas measure information. The unit of information is bits if log is interpreted as $\log_2$, or nats if log is interpreted as $\log_e$. This is simply a difference in units, with 1 bit ≈ 0.69 nats. It is conventional to use bits in information theory; however, nats will be convenient in §11.

Given a family of distributions $\mathscr{Q}$, we deifne

- The I-projection $q = \min_{q\in\mathscr{Q}} \mathrm{KL}(q \mid\mid p)$, which will have $\text{support}(q) \subseteq \text{support}(p)$ if possible.
- The M-projection $q = \min_{q\in\mathscr{Q}} \mathrm{KL}(p \mid\mid q)$, which will have $\text{support}(p) \subseteq \text{support}(q)$ if possible.

[8] Especially for the $\mathbf{x}$ that are probable according to p^*. That is, we want $p^*(\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ to be a good approximation of $p^*(\mathbf{x}, \mathbf{y})$.

[9] The term "divergence" is used instead of "distance" because it is not symmetric: $\mathrm{KL}(q \mid\mid p) \neq \mathrm{KL}(p \mid\mid q)$ in general.

7 Datasets

$\mathscr{D} = \{\langle \mathbf{x}_1, \mathbf{o}_1 \rangle, \ldots, \langle \mathbf{x}_N, \mathbf{o}_N \rangle\}$ denotes the training dataset. We use $\mathscr{D}$ to help us estimate $\boldsymbol{\theta}$, under

- the independent and identically distributed (IID) assumption that $\mathscr{D}$ is an observation of N pairs $\langle \mathbf{x}_n, \mathbf{y}_n \rangle$ that were drawn independently from the same distribution,
- the in-domain assumption that this distribution was p^*, and
- the missing at random (MAR) assumption that when $\mathscr{D}$ contains only a *partial* observation $\mathbf{o}_n$ of $\mathbf{y}_n$, no additional information about the missing data $\mathbf{v}_n$ is provided by the fact that $\mathbf{v}_n$ is missing.

Together, these assumptions imply that the posterior distribution of $\mathbf{y}_n$ given $\mathscr{D}$ is indeed $p^*(\mathbf{Y} \mid \mathbf{x}_n, \mathbf{o}_n)$.

8 More decision theory: Decision rules and action values

The expectation $\sum_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) f(\mathbf{y})$ can be written informally as $\mathbb{E}[f(\mathbf{y})]$, or with increasing degrees of explictness as $\mathbb{E}_{\mathbf{y}}[f(\mathbf{y})]$ or $\mathbb{E}_{\mathbf{y}|\mathbf{x}}[f(\mathbf{y})]$ or $\mathbb{E}_{\mathbf{y}\sim p}[f(\mathbf{y})]$ or $\mathbb{E}_{\mathbf{y}\sim p(\cdot|\mathbf{x})}[f(\mathbf{y})]$.

A decision rule is a method for choosing a plan $\mathbf{a}$ given input $\mathbf{x}$. A deterministic decision rule can be written as a function $\mathbf{a} = \boldsymbol{\pi}(\mathbf{x})$. More generally, a stochastic decision rule is a distribution $\boldsymbol{\pi}(\mathbf{a} \mid \mathbf{x})$.

Any decision rule may be expressed as a policy, which chooses the plan $\mathbf{a}$ one action a_j at a time. A deterministic policy is a function $\mathrm{a}_j = \pi(\mathbf{x}, \mathbf{a}_{:j-1})$, while a stochastic policy is a distribution $\pi(\mathrm{a}_j \mid \mathbf{x}, \mathbf{a}_{:j-1})$. In general, these may be selected from some fixed parametric family by choosing the parameters.

Given $\mathbf{x}$, the value of a plan $\mathbf{a}$ is defined to be its expected reward under p^*,[10] $Q(\mathbf{a} \mid \mathbf{x}) \stackrel{\text{def}}{=} \sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}} p^*(\mathbf{y} \mid \mathbf{x}) R(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$.[11] Here the expectation is taken under the posterior predictive distribution, that is, $p^*(\mathbf{y} \mid \mathbf{x})$ or more generally $p^*(\mathbf{y} \mid \mathbf{x}, \mathbf{o})$. A Bayes decision rule chooses a plan $\mathbf{a}^*$ that maximizes $Q(\mathbf{a} \mid \mathbf{x})$: that is, $\text{decision}(\mathbf{x}) = \mathbf{a}^* \operatorname{argmax}_{\mathbf{a}} Q(\mathbf{a} \mid \mathbf{x})$.[12] Any other plan $\mathbf{a}$ is said to have regret of $Q(\mathbf{a}^* \mid \mathbf{x}) - Q(\mathbf{a} \mid x\, x) \geq 0$.

In our setting, we do not actually know Q because we do not know the true distribution p^*. However, if we can estimate $p_{\boldsymbol{\theta}} \approx p^*$, we can use $p_{\boldsymbol{\theta}}$ as a drop-in replacement for p^*. That is, we can define the *estimated* value $\hat{Q}(\mathbf{a} \mid \mathbf{x}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{y}\sim p_{\boldsymbol{\theta}}}[R(\mathbf{a} \mid \mathbf{x}, \mathbf{y})]$, and use the rule $\text{decision}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{a}} \hat{Q}(\mathbf{a} \mid \mathbf{x})$.[13] Notice that in this approach, $\boldsymbol{\theta}$ is tuned to make $p_{\boldsymbol{\theta}} \approx p^*$ without considering rewards; the decision rule then follows from $p_{\boldsymbol{\theta}}$ and the reward function.

When $\mathbf{a}$ is intended to be a prediction of $\mathbf{y}$ (or $\mathbf{o}$ or $\mathbf{v}$), a decision rule is sometimes called a prediction rule or a decoder. In this case, the Bayes decision rule or the above approximation to it is commonly known as a minimum Bayes risk (MBR) decoder (see footnote 10), although in our terminology it would better be called a maximum value decoder.

9 Decision processes

Different problem settings arise in machine learning. So far we have considered prediction and decision settings. The setting used for reinforcement learning (RL) requires two changes.

[10] Similarly, the risk of an plan is defined as its expected loss. (Recall from footnote 6 that loss is negated reward.) The specific risk defined here would be called the posterior predictive risk of the plan $\mathbf{a}$, because the expectation is taken under the posterior predictive distribution. By contrast, the Bayes risk of $\mathbf{a}$ averages over $\mathbf{x}$ as well as $\mathbf{y}$: $Q(\mathbf{a}) = \sum_{\mathbf{x} \in \mathscr{X}, \mathbf{y} \in \mathscr{Y}_{\mathbf{x}}} p^*(\mathbf{x}, \mathbf{y}) R(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$.

[11] The traditional Q-function notation in RL is $Q(\mathbf{x}, \mathbf{a})$. But I prefer writing $Q(\mathbf{a} \mid \mathbf{x})$, which should be read "the value of $\mathbf{a}$ *given* $\mathbf{x}$," similar to a conditional probability. Indeed, its definition strongly resembles a valid conditional probability formula.

[12] There may exist multiple Bayes decision rules, but these differ only in how they break ties.

[13] There may also be other ways to construct an estimated value function $\hat{Q}$. Alternatively, we can train a policy so that it tends to prefer high-value plans, without necessarily fitting $\hat{Q}$. An actor-critic method does both: it jointly trains a policy $\boldsymbol{\pi}$ (the actor) and a value estimator $\hat{Q}$ (the critic).

There is still a true probability distribution $p^*(\mathbf{y} \mid \mathbf{x})$, which is called the environment. However, the agent's job is now to choose $\mathbf{x}$! In other words, there is no separate $\mathbf{a}$; we identify $\mathbf{a}$ with $\mathbf{x}$.

The reward function now has the form $R(\mathbf{x}, \mathbf{y})$ rather than $R(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$. The value of the plan $\mathbf{x}$ is now $\mathbb{E}_{\mathbf{y} \sim p_{\boldsymbol{\theta}}}[R(\mathbf{x}, \mathbf{y})]$, so it is still an expected reward where the expectation is over $\mathbf{y}$. The goal is still to choose a high-value action.

The other major change is that the decision rule is replaced with an interactive decision process. The agent and the environment take turns choosing the characters of their strings $\mathbf{x}$ and $\mathbf{y}$, so they can react to one another.[14] At each time step $j = 1, 2, \ldots$, the agent first draws its action x_j (traditionally called a_j) from a policy $\pi(\mathrm{x}_j \mid \mathbf{x}_{:j-1}, \mathbf{y}_{:j-1})$.[15] The environment then responds by drawing its output symbol y_j from the distribution $p^*(\mathrm{y}_j \mid \mathbf{x}_{:j}, \mathbf{y}_{:j-1})$.

In a partially observable decision process, the agent's policy is not allowed to depend on all of the environment's previous output $\mathbf{y}_{:j-1}$, but only on the observables $\mathbf{o}_{:j-1}$ that the agent has actually seen.

It is traditional to define the reward function as $R(\mathbf{x}, \mathbf{y}) = \sum_j r_j(\mathbf{x}_{:j}, \mathbf{y}_{:j})$, so that the reward is accumulated one step at a time: $R = r_1 + r_2 + \cdots$. This decomposition involves no loss of generality.

Notice that in a decision process, $p^*(\mathbf{y} \mid \mathbf{x})$ is factored as $\prod_j p^*(\mathrm{y}_j \mid \mathbf{x}_{:j}, \mathbf{y}_{:j-1})$. This deliberately loses generality: it means that the environment's output y_j cannot depend on the agent's *future* actions $\mathbf{x}_{j:}$.[16]

Conversely, the agent's next action x_{j+1} cannot depend on the environment's future outputs $\mathbf{y}_{j:}$. However, it could still consider how the environment *might* react to a future plan $\mathbf{x}_{j:} = \mathrm{x}_{j+1}\mathrm{x}_{j+2}\cdots$. The (residual) value of this future plan is an *expectation* of $r_{j+1} + r_{j+2} + \ldots$, where the expectation is taken under $p^*(\mathbf{y} \mid \mathbf{x})$. That is, computing the value must guess the environment's unknown future outputs $\mathbf{y}_{j:}$, as well as the unknown past outputs $\mathbf{y}_{:j}$ in the partially observable case. An agent with knowledge of p^* could consider these values in its policy for selecting x_{j+1}.

10 Scoring functions

Given an input $\mathbf{x}$, we will evaluate the "goodness" of a candidate output $\mathbf{y}$ by its score $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$.[17] So the scoring function is $G_{\boldsymbol{\theta}} : \{\langle \mathbf{x}, \mathbf{y} \rangle : \mathbf{x} \in \mathscr{X}, \mathbf{y} \in \mathscr{Y}_{\mathbf{x}}\} \to \mathbb{R} \cup \{-\infty\}$, with parameters $\boldsymbol{\theta}$. As before, I may suppress the $\boldsymbol{\theta}$ subscript for brevity. We will often define

$$G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{J} g_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_{:j}) \tag{1}$$

where the jth summand can be viewed as scoring output character y_j given all previous outputs, much as in most language models. These summands are also called scores (or subscores).

Remark: A simple decision rule for predicting $\mathbf{y}$ is to choose the highest-scoring $\mathbf{y}$. There exist methods (e.g., the structured perceptron, the structured support vector machine, direct loss minimization, and bandit algorithms) for choosing $\boldsymbol{\theta}$ so that this simple decision rule achieves high value (equivalently, low Bayes risk).

[14] If instead the agent must choose $\mathbf{x}$ all at once, then we are in the bandit setting. This is equivalent to the special case of RL in which $J = 1$. The environment in this case is called a multi-armed bandit, i.e., a slot machine whose arms are the possible choices of $\mathbf{x}$. Traditionally in the bandit setting, the agent does not get to observe $\mathbf{y}$ but only the resulting reward R.

[15] I have written this as a stochastic policy; a deterministic policy is a special case.

[16] Environments do not even depend on the agent's *policy* (i.e., its probabilities of future actions). Environments remain fixed while the agent chooses its policy; they are random, not cooperative or adversarial. As we will see in the next paragraph, this is a fundamental asymmetry between agent and environment. An environment that could adapt to the agent's policy would be deemed another agent—e.g., an opponent—rather than an environment. That multi-agent setting is called game playing rather than reinforcement learning.

[17] This is unrelated to the use of "score" in statistics (i.e., the gradient of log-likelihood with respect to a model's parameters).

I will sometimes abbreviate the score as simply G. We will also use the shorthand notation G_j for the prefix score of the prefix $\mathbf{y}_{:j}$, defined by $G_j = G_{j-1} + g_\theta(\mathbf{x}, \mathbf{y}_{:j})$ with base case $G_0 = 0$. Then the total score of $\mathbf{y}$ is $G = G_J$.

Scores fall in $\mathbb{R} \cup \{-\infty\}$. In all of our methods, an output $\mathbf{y}$ with score of $-\infty$ can safely be treated as if it is not even in the output space $\mathscr{Y}_{\mathbf{x}}$. Thus, it is sometimes convenient to define a simple output space that is larger than necessary, such as $\mathscr{Y}_{\mathbf{x}} = \mathscr{Y}^J$, and exclude ill-formed outputs from it using scores. Specifically, any "bad configuration" that makes an output ill-formed should receive a subscore of $-\infty$, so that that output gets a total score of $-\infty$.

11 Exponential probability models (Boltzmann distributions)

This class focuses on probability models. We will ordinarily derive our parametric probability model from a parametric scoring function as follows:[18]

$$p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{Z(\mathbf{x})} \tilde{p}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \tag{2}$$

where the normalizer[19] $Z(\mathbf{x})$ ensures that the distribution sums to 1,

$$Z(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}} \tilde{p}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \tag{3}$$

and the unnormalized conditional distribution $\tilde{p}_{\boldsymbol{\theta}}$ is obtained from the scores as

$$\tilde{p}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \stackrel{\text{def}}{=} \exp G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{J} \underbrace{\exp g_\theta(\mathbf{x}, \mathbf{y}_{:j})}_{\text{call this } \psi_j(\mathbf{x}, \mathbf{y}_{:j})} = \prod_{i=1}^{J} \psi_j(\mathbf{x}, \mathbf{y}_{:j}) \tag{4}$$

Here the exponentiated subscore $\psi_j(\mathbf{x}, \mathbf{y}_{:j}) \geq 0$ is referred to as simply a factor—that is, a factor of the unnormalized probability $\tilde{p}$. The function ψ_j is traditionally called a potential function.[20] Note that a subscore of $-\infty$ yields a 0 factor, which always results in an overall probability of $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) = 0$. These are usually structural zeroes, meaning that their 0 value is imposed by the structure of the model and does not depend on how the parameters $\boldsymbol{\theta}$ are set.

What are the units of G? Since $\exp G$ is a probability, we might say that G returns goodness values in units of nats.[21] More precisely, $\exp G$ is an *unnormalized* probability, so we should say that *differences* in G values are in units of nats: if the difference $G(\mathbf{y}) - G(\mathbf{y}') = g$ nats, then $\exp g$ gives the probability ratio $p(\mathbf{y})/p(\mathbf{y})$. It is really only such differences that have units: the raw G values are meaningless, since adding a constant to all G values (replacing $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ with $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) + c$ or even $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) + c(\mathbf{x})$) would not change the conditional distribution $p(\mathbf{y} \mid \mathbf{x})$.

11.1 Marginal and conditional probabilities

Given a training or test example $\langle \mathbf{x}, \mathbf{o} \rangle$ (see §3), we are often interested in the marginal distribution

$$p(\mathbf{o} \mid \mathbf{x}) = \sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}}} p(\mathbf{y} \mid \mathbf{x}) = \sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}}} \frac{1}{Z(\mathbf{x})} \exp G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x},\mathbf{o}}} \exp G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})}{\sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}} \exp G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})} = \frac{Z(\mathbf{x}, \mathbf{o})}{Z(\mathbf{x})} \tag{5}$$

[18] Often such models are expressed as $\exp -E$ instead of $\exp G$, where $E(\mathbf{x}, \mathbf{y}) = -G(\mathbf{x}, \mathbf{y})$ is called the energy of $\langle \mathbf{x}, \mathbf{y} \rangle$. This leads to the name energy-based model.

[19] Another common name for $Z(\mathbf{x})$ is the partition function. Like "energy," this term comes from statistical physics.

[20] Which appears to be a misnomer. One would expect a potential function to return a potential. However, as I understand the physics analogy, the potential energy contributed by y_j is not actually $\psi_j(\mathbf{x}, \mathbf{y}_{:j})$ but rather $-\log \psi_j(\mathbf{x}, \mathbf{y}_{:j}) = -g_\theta(\mathbf{x}, \mathbf{y}_{:j})$.

[21] If we defined $\tilde{p} = 2^G$ instead of $\exp G$, then G would be denominated in bits rather than nats.

where the last step defines $Z(\mathbf{x}, \mathbf{o})$ to sum over $\mathscr{Y}_{\mathbf{x},\mathbf{o}}$ just as $Z(\mathbf{x})$ sums over $\mathscr{Y}_{\mathbf{x}}$. We are also sometimes interested in the conditional distribution

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{o}) = \frac{p(\mathbf{y} \mid \mathbf{x})}{p(\mathscr{Y} \mid \mathbf{x})} = \frac{\frac{1}{Z(\mathbf{x})} \exp G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})}{\frac{1}{Z(\mathbf{x})} Z(\mathbf{x}, \mathbf{o})} = \frac{1}{Z(\mathbf{x}, \mathbf{o})} \exp G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \tag{6}$$

11.2 Special case: Locally normalized models

In general, the normalizer $Z(\mathbf{x})$ is necessary to ensure that $p_{\boldsymbol{\theta}}$ is a proper probability distribution. Since it is defined by the "global" sum over $\mathscr{Y}_{\mathbf{x}}$, we say that the model is globally normalized.

However, global normalization turns out to be unnecessary if each potential function ψ_j happens to specify a conditional distribution over symbols $\mathrm{y}_j \in \mathscr{Y}$. Formally, for any $\mathbf{x}, \mathbf{y}_{:j-1}$, we happen to have $\sum_{\mathrm{y}_j \in \mathscr{Y}} \psi_j(\mathbf{x}, \mathbf{y}_{:j})$. This can be ensured by using a locally normalized model as follows.

The true conditional distribution can always be factored into a product of local conditional probabilities (as commonly noted in language modeling):

$$p^*(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{J} p^*(\mathrm{y}_j \mid \mathbf{x}, \mathrm{y}_{:j-1}) \tag{7}$$

without any global normalizing constant. We can similarly define our model in a factored form:

$$p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \stackrel{\text{def}}{=} \prod_{i=1}^{J} p_{\boldsymbol{\theta}}(\mathrm{y}_j \mid \mathbf{x}, \mathrm{y}_{:j-1}) \tag{8}$$

where each local conditional distribution $p_{\boldsymbol{\theta}}$ approximates the corresponding p^* factor. Specifically, we can choose to fit all the p^* factors using a single "locally normalized" sub-model of the form

$$p_{\boldsymbol{\theta}}(\mathrm{y}_j \mid \mathbf{x}, \mathrm{y}_{:j-1}) \stackrel{\text{def}}{=} \psi_j(\mathbf{x}, \mathbf{y}_{:j}) = \frac{1}{Z(\mathbf{x}, \mathrm{y}_{:j-1})} \tilde{\psi}_j(\mathbf{x}, \mathbf{y}_{:j}) \tag{9}$$

where $$\tilde{\psi}_j(\mathbf{x}, \mathbf{y}_{:j}) \stackrel{\text{def}}{=} \exp \tilde{g}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_{:j}) \tag{10}$$

and the local normalizer $Z(\mathbf{x}, \mathrm{y}_{:j-1})$ is chosen to make (9) sum to 1.

Clearly (8) is an example of (4), with $g_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_{:j}) \stackrel{\text{def}}{=} \log \psi_j(\mathbf{x}, \mathbf{y}_{:j})$. However, thanks to the special structure of the potential functions ψ_j, the global normalizing constant $Z(\mathbf{x})$ is now guaranteed to be 1 and can be dropped.

Notice that a globally normalized model may be defined by directly specifying the local scoring function $g_{\boldsymbol{\theta}}$, from which we obtain ψ. A locally normalized model is defined by specifying an *unnormalized* local scoring function $\tilde{g}_{\boldsymbol{\theta}}$, from which we obtain $\tilde{\psi}$, ψ, and finally $g_{\boldsymbol{\theta}}$. This may be viewed as an indirect way of specifying $g_{\boldsymbol{\theta}}$ that guarantees the desired condition on ψ.

11.3 Restricting summations to the output space

$p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ is only defined for $\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}$. We always want $\sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}} p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) = 1$. In the globally normalized model, we achieved this by summing over $\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}$ when computing the global normalizer in (3).

In the locally normalized model (9), it is slightly harder to ensure that we do not waste probability mass on illegal strings. We must define

$$Z(\mathbf{x}, \mathbf{y}_{:j-1}) \stackrel{\text{def}}{=} \sum_{\mathrm{y} \in \mathscr{Y}_{\mathbf{x}, \mathbf{y}_{:j-1}}} \tilde{\psi}_j(\mathbf{x}, \mathbf{y}_{:j}) \tag{11}$$

where $\mathscr{Y}_{\mathbf{x},\mathbf{y}_{:j-1}} \stackrel{\text{def}}{=} \{\mathrm{y} : (\exists \mathbf{y}_{j:} \in \mathscr{Y}^{J-j})\, \mathbf{y}_{:j-1}\, \mathrm{y}\, \mathbf{y}_{j:} \in \mathscr{Y}_{\mathbf{x}}\}$ is the set of legal choices for the next character y_j.

Equivalently (as §10 suggested), we could sum over a larger space but force the probability of the illegal choices to 0. Define $\tilde{g}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_{:j}) = -\infty$ whenever $\mathrm{y}_j \notin \mathscr{Y}_{\mathbf{x},\mathbf{y}_{:j-1}}$. Then it is safe to sum over all $\mathrm{y} \in \mathscr{Y}$ in (11). (In the globally normalized model, defining g with the same 0 values would make it safe to sum over all $\mathbf{y} \in \mathscr{Y}^J$.)

11.4 Joint models

A different approach to obtaining a conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ is to first train a *joint* distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \approx p^*(\mathbf{x}, \mathbf{y})$, and then conditionalize it. This means that $\boldsymbol{\theta}$ is being asked to explain the $\mathbf{x}$ values—not just the $\mathbf{y}$ values—that $\mathscr{D}$ has drawn from p^*. Joint training could either help or hurt the fit of $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x})$ to $p^*(\mathbf{y} \mid \mathbf{x})$, depending on the qualities of the joint model $\mathscr{P}$.

In order to write down a joint distribution, we have two formal options:

Change the formulas. The traditional solution is to write down joint versions of the conditional models above, moving $\mathbf{x}$ to the left side of the probability bar.

- For the globally normalized case, define $\tilde{p}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \exp G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, and normalize it via $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \frac{1}{Z} \tilde{p}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$. The global normalizer $Z = \sum_{\mathbf{x} \in \mathscr{X}} \sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}} \tilde{p}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ now requires a double sum.
- For the locally normalized case, define $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \prod_{i=1}^{J} p_{\boldsymbol{\theta}}(\mathrm{x}_j, \mathrm{y}_j \mid \mathbf{x}_{:j-1}, \mathbf{y}_{:j-1})$. The local normalizer now has the form $Z(\mathbf{x}_{:j-1}, \mathbf{y}_{:j-1})$ and requires a double sum over x_j and y_j.

Change the problem mapping. A different solution is to change the problem setup. Recall from §3 that we can take each symbol y to be an ordered pair $\langle \mathrm{o}, \mathrm{v} \rangle$. Thus, we already have enough machinery to handle joint distributions: $p(\mathbf{y} \mid \mathbf{x})$ can be regarded as a joint distribution $p(\mathbf{o}, \mathbf{v} \mid \mathbf{x})$. This is still conditioned on $\mathbf{x}$, which may be useful: we can use $\mathbf{x}$ to specify "background information" such as the domain of the example, so that we get a different joint distribution in each domain. Or we can ignore $\mathbf{x}$ by always taking it to be $-^J$ (a string of J dashes), where J is the length of $\mathbf{o}$.

We can thus regard our problem's $\langle \text{input}, \text{output} \rangle$ pairs as $\langle \mathbf{o}, \mathbf{v} \rangle$ pairs instead of $\langle \mathbf{x}, \mathbf{y} \rangle$ pairs. Fitting the model $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{o}, \mathbf{v} \mid \mathbf{x})$ is now fitting a joint distribution over $\langle \text{input}, \text{output} \rangle$ pairs (conditioned on the optional background information $\mathbf{x}$). When $\mathbf{o}$ is observed at test time, the posterior predictive distribution $p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}, \mathbf{o})$ is tantamount to $p_{\boldsymbol{\theta}}(\mathbf{v} \mid \mathbf{x}, \mathbf{o})$, a conditional distribution of output $\mathbf{v}$ given input $\mathbf{o}$ as desired.

Which option is better? Unfortunately both are useful:

- Changing the formulas is best when we are focusing on a single formalization of the *problem*, so that we have fixed $\langle \mathbf{x}, \mathbf{y} \rangle$ as the names of the $\langle \text{input}, \text{output} \rangle$ strings. We want to compare the behavior of conditional vs. joint models.
- Changing the mapping is best when we are focusing on a single *algorithm*. Although the algorithm assumes a conditional model, we would like to be able to reuse its implementation or analysis for joint models.

11.5 Heated and cooled distributions

For computational reasons, it is sometimes useful to consider related distributions. These distributions are marked with an exponent $\beta \geq 0$, the inverse temperature. We define the unnormalized distribution

$$\tilde{p}_{\boldsymbol{\theta}}^{\beta}(\mathbf{y} \mid \mathbf{x}) = (\tilde{p}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}))^{\beta} = \prod_{i=1}^{J} \psi_j(\mathbf{x}, \mathbf{y}_{:j})^{\beta} = \exp\left(\beta G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})\right) \tag{12}$$

and normalize it as usual:

$$Z^{\beta}(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}} \tilde{p}_{\boldsymbol{\theta}}^{\beta}(\mathbf{y} \mid \mathbf{x}) \tag{13}$$

$$p_{\boldsymbol{\theta}}^{\beta}(\mathbf{y} \mid \mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{Z^{\beta}(\mathbf{x})} \tilde{p}_{\boldsymbol{\theta}}^{\beta}(\mathbf{y} \mid \mathbf{x}) \tag{14}$$

In general, such a normalized distribution is called a Boltzmann distribution or a Gibbs distribution (more properly, a conditional distribution, since it is conditioned on $\mathbf{x}$). When $\beta = 1$, this is our main distribution of interest, as already defined at the start of §11. The heated distributions obtained with $\beta < 1$ are "flatter" (more uniform), whereas the cooled distributions obtained with $\beta > 1$ are "sharper" and place more of the probability mass on the higher-scoring outputs. At the limit $\beta = \infty$, $p_{\boldsymbol{\theta}}^{\beta}$ places all its probability mass on the single highest-scoring output (or outputs, in case of ties).

11.6 Feature vectors

12 Training objectives

The likelihood of a model p is the probability that it assigns to the data.

If q is the empirical distribution of data, then $\operatorname{argmin}_{\boldsymbol{\theta}} H(q, p_{\boldsymbol{\theta}}) = \operatorname{argmin}_{\boldsymbol{\theta}} \mathrm{KL}(q \parallel p_{\theta}) = \operatorname{argmax}$ is called the maximum-likelihood estimate of θ

13 States and belief states

14 Weight semirings

15 Graphical models

16 FSTs and options

17 Trees and grammars

18 Proof systems

19 Neural networks

20 Reinforcement learning

Index

actions, 3
actor-critic method, 5
agents, 3

bandit setting, 6
Bayes decision rule, 5
Bayes risk, 5
bits, 4
blue background, 1
Boltzmann distribution, 10

complete string, 3
conditional distribution, 8
cooled, 10
cross-entropy, 4

decision process, 6
decision rule, 5
decoder, 5
drop-in replacement, 5

energy, 7
energy-based model, 7
entropy, 4
environment, 6
estimated conditional distribution, 3
example, 2
exclusive Kullback-Leibler divergence, 4

factor, 7
family, 4
fully observed, 2

game playing, 6
Gibbs distribution, 10
globally normalized, 8

heated, 10

I-projection, 4
IID, *see* independent and identically distributed
in-domain, 5
inclusive Kullback-Leibler divergence, 4
independent and identically distributed, 5
input, 2
input alphabet, 2
input space, 2
inverse temperature, 10

latent string, 3
latent symbol, 3
likelihood, 10
local conditional distribution, 8
local normalizer, 8
locally normalized, 8
Loss, 3

M-projection, 4
MAR, *see* missing at random
marginal distribution, 7
maximum-likelihood estimate, 10
MBR, *see* minimum Bayes risk
minimum Bayes risk, 5
missing at random, 5
multi-agent, 6
multi-armed bandit, 6

nats, 4
normalizer, 7

observation, 2
observed symbol, 2
output, 2
output alphabet, 2
output space, 2

parameter vector, 4
parametric model, 4
partially observable decision process, 6
partially observed, 2
partition function, 7
plan, 3
policy, 5
posterior predictive distribution, 3
posterior predictive risk, 5
potential function, 7
prediction, 3
prediction rule, 5
prefix, 1
prefix score, 7
problem settings, 5

regret, 5
reinforcement learning, 5

reward function, 3
risk, 5
RL, *see* reinforcement learning

score, 6
scoring function, 6
strings, 1
structural zeroes, 7
subscores, 6
substring, 1
suffix, 1
support, 4
surprisal, 4
symbols, 1

training dataset, 5
true joint distribution, 3

unnormalized conditional distribution, 7
unobserved, 2

value, 5
variational family, 4