Formalisms for Sequence Modeling

Jason Eisner — JHU 601.765

January 29, 2018

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 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 matrix $\boldsymbol{W} \in \mathbb{R}^{I \times J}$ would have elements written like W_{ij} for $1 \leq i \leq I, 1 \leq j \leq J$.

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. Similarly, boldface $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots$ denotes a sequence of *vectors*, with h_{ij} representing the scalar j^{th} element of the vector $\boldsymbol{h}_i$.

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: BOS `abc` EOS, or HELLO WORLD.

The string $\mathbf{y} = \mathrm{y}_1 \cdots \mathrm{y}_J$ for some $J \in \mathbb{N}$. For convenience, for any $0 \leq i \leq j \leq J$, 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: $\mathrm{y} \in \mathcal{Y}, \mathbf{y} \in \mathcal{Y}^*$. 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 $\boldsymbol{s} \in \mathbb{R}^D$ 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 $\mathbf{x}$.[2]

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

In general $\mathbf{x}$ and $\mathbf{y}$ will be structured objects. $\mathcal{X}$ 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 $\mathcal{Y}_{\mathbf{x}} \rightarrow \mathbb{R}_{\geq 0}$ where $\mathcal{Y}_{\mathbf{x}}$ is the output space that we will consider computationally. Outside this handout, I may abbreviate $\mathcal{Y}_{\mathbf{x}}$ as $\mathcal{Y}$ when $\mathbf{x}$ is clear from context.

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 $\mathbf{x} \in \mathcal{X} \subseteq \mathcal{X}^*$ where $\mathcal{X}$ is a fixed input alphabet, and $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}} \subseteq \mathcal{Y}^J$ where $J = |\mathbf{x}|$ and $\mathcal{Y}$ is a fixed output alphabet.[3]

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

QUESTION: How would you do language modeling?

QUESTION: How would you do BIO tagging?

3 Observations

For any training or test example, we are given the full input $\mathbf{x}$ together with an observation of the output—a subset of $\mathcal{Y}_{\mathbf{x}}$ that is known to contain $\mathbf{y}$. An unobserved output is the case where this subset is all of $\mathcal{Y}_{\mathbf{x}}$; a fully observed output is the case where this subset is a singleton set $\{\mathbf{y}\}$; and a partially observed output is anything in between. In general, training examples will be partially or fully observed (so that they *contain* information that can be learned from), while test examples will be unobserved or partially observed (so that they *omit* information that must be predicted).

To simplify this notion without loss of generality, we will assume that the observation is itself presented as a string $\mathbf{o}$ of length J. Each observed symbol o_j is a partial observation of the output symbol y_j, meaning that it is compatible with some set of possibilities for y_j, denoted $\mathcal{Y}_{\mathrm{o}_j} \subseteq \mathcal{Y}$. Thus, when $\mathbf{o}$ is observed with $\mathbf{x}$, the output space can be limited to $\mathcal{Y}_{\mathbf{x},\mathbf{o}} \stackrel{\text{def}}{=} \mathcal{Y}_{\mathbf{x}} \cap (\mathcal{Y}_{\mathrm{o}_1} \cdots \mathcal{Y}_{\mathrm{o}_J})$ (which again I may abbreviate as $\mathcal{Y}$ when clear).

Often it is convenient to suppose that each y_j is an ordered pair $\langle \mathrm{o}_j, \mathrm{v}_j \rangle$, where v_j is a latent symbol that specifies the unobserved portion of y_j. Then predicting $\mathbf{y}$ is equivalent to predicting the latent string $\mathbf{v}$.

Generalizing slightly, we do not actually have to require y_j to be an ordered pair. It suffices to say that for every o, there is a bijection between the possible output symbols $\mathcal{Y}_{\mathrm{o}}$ and their possible latent parts $\mathcal{V}_{\mathrm{o}}$. 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.[4]

QUESTION: why without loss of generality?

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 $\mathcal{A} \subseteq \mathcal{A}^J$.

An agent that is designed to predict $\mathbf{y}$ from $\mathbf{x}$ takes $\mathcal{A} = \mathcal{Y}$ 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, $\mathbf{a}$

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

[4] The fact that v_j is uniquely determined implies that all the latent information from $\mathbf{v}$ is actually preserved in $\mathbf{y}$. Thus, while we may have to sum over possible $\mathbf{y}$ values 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 $\mathbf{y}$.

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 $a_1, \ldots, a_J$.

Some predictions are more accurate than others, and some plans work better than others. The reward function[5] $R(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$ 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: 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 $p^*(\mathbf{y} \mid \mathbf{x}, \mathbf{o})$ 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 $\mathbf{o}$.

$\mathscr{P} = \{p_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ 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 $p^*(\mathbf{y} \mid \mathbf{x})$.[6]

$\hat{p}(\mathbf{x}, \mathbf{y})$ 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 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 $\{\mathbf{x}_n, \mathbf{y}_n\}$ 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 $p^*(\mathbf{Y} \mid \mathbf{x}_n, \mathbf{o}_n)$.

[5] 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.

[6] 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 $p^*(\mathbf{x}, \mathbf{y})$.

7 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 more explicitly as $\mathbb{E}_{\mathbf{y}}[f(\mathbf{y})]$ or $\mathbb{E}_{\mathbf{y} \sim p}[f(\mathbf{y})]$ or $\mathbb{E}_{\mathbf{y} \sim p(\cdot \mid \mathbf{x})}[f(\mathbf{y})]$.

A decision rule is a method for choosing an 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,[7] $Q(\mathbf{x}, \mathbf{a}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{y} \sim p^*}[R(\mathbf{a} \mid \mathbf{x}, \mathbf{y})]$. 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 $Q(\mathbf{x}, \mathbf{a})$.[8]

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, an agent can choose a plan that maximizes the *estimated* value $\hat{Q}(\mathbf{x}, \mathbf{a}) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{y} \sim p_{\boldsymbol{\theta}}}[R(\mathbf{a} \mid \mathbf{x}, \mathbf{y})]$.[9] 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 7), although in our terminology it would better be called a maximum value decoder.

8 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.

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.[10] At each time step $j = 1, 2, \ldots$, the agent first draws its action x_j (traditionally called a_j) from a policy $\pi(\mathrm{x}_j \mid \mathbf{x}_{:j-1}, \mathbf{y}_{:j-1})$.[11] The environment then responds by drawing its output symbol y_j from the distribution $p^*(\mathrm{y}_j \mid \mathbf{x}_{:j}, \mathbf{y}_{:j-1})$.

[7] Similarly, the risk of an plan is defined as its expected loss. (Recall from footnote 5 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. The unconditioned version that takes the expectation under all of $p^*(\mathbf{x}, \mathbf{y})$ is called the Bayes risk of $\mathbf{a}$.

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

[9] 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).

[10] 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.

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

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 $\mathbf{x}_{j:}$.[12]

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}.

9 Scoring functions

Given an input $\mathbf{x}$, we will evaluate the "goodness" of a candidate output $\mathbf{y}$ by its score $G_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y})$.[13] 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 j^{th} 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).

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 $-\infty$.

[12] 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.

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

10 Exponential probability models

This class focuses on probability models. We will ordinarily derive our parametric probability model from a parametric scoring function as follows:[14]

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

where the normalizer[15] $Z(\mathbf{x})$ 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.[16] 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.

10.1 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 $\mathscr{Y}$, 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):

$$p^*(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{J} p^*(\mathrm{y}_j \mid \mathbf{x}, \mathrm{y}_{:j-1}) \tag{5}$$

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

$$p_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \stackrel{\text{def}}{=} \prod_{i=1}^{J} p_{\boldsymbol{\theta}}(\mathrm{y}_j \mid \mathbf{x}, \mathrm{y}_{:j-1}) \tag{6}$$

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

$$p_{\boldsymbol{\theta}}(\mathrm{y}_j \mid \mathbf{x}, \mathrm{y}_{:j-1}) \stackrel{\text{def}}{=} \psi_j(\mathbf{x}, \mathbf{y}_{:j}) = \frac{1}{Z(\mathbf{x}, \mathrm{y}_{:j-1})} \tilde{\psi}_j(\mathbf{x}, \mathbf{y}_{:j}) \tag{7}$$

where

$$\tilde{\psi}_j(\mathbf{x}, \mathbf{y}_{:j}) \stackrel{\text{def}}{=} \exp \tilde{g}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_{:j}) \tag{8}$$

[14] 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.

[16] 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 $-\log \psi_j(\mathbf{x}, \mathbf{y}_{:j}) = -g_\theta(\mathbf{x}, \mathbf{y}_{:j})$.

and the local normalizer $Z(\mathbf{x}, \mathrm{y}_{:j-1})$ is chosen to make (7) sum to 1.

Clearly (6) 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.

10.2 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 (7), it is slightly harder to ensure that we do not waste probability mass on illegal strings. We must define

$$Z(\mathbf{x}, \mathbf{y}_{:j-1}) \stackrel{\text{def}}{=} \sum_{\mathrm{y} \in \mathscr{Y}_{\mathbf{x}, \mathbf{y}_{:j-1}}} \tilde{\psi}_j(\mathbf{x}, \mathbf{y}_{:j}) \tag{9}$$

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 §9 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 (9). (In the globally normalized model, defining g with the same 0 values would make it safe to sum over all $\mathbf{y} \in \mathscr{Y}^J$.)

10.3 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 ⟨input, output⟩ 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 ⟨input, output⟩ 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 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 ⟨input, output⟩ 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.

10.4 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

$$\tilde{p}^{\beta}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) = (\tilde{p}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}))^{\beta} = \prod_{i=1}^{J} \psi_j(\mathbf{x}, \mathbf{y}_{:j})^{\beta} = \exp\left(\beta\, G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})\right) \tag{10}$$

and normalize it as usual:

$$Z^{\beta}(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{\mathbf{y} \in \mathscr{Y}_{\mathbf{x}}} \tilde{p}^{\beta}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \tag{11}$$

$$p^{\beta}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{Z^{\beta}(\mathbf{x})} \tilde{p}^{\beta}_{\boldsymbol{\theta}}(\mathbf{y} \mid \mathbf{x}) \tag{12}$$

When $\beta = 1$, this is our main distribution of interest, as already defined at the start of §10. 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 $\beta = \infty$, $p^{\beta}_{\boldsymbol{\theta}}$ places all its probability mass on the single highest-scoring output (or outputs, in case of ties).

Index