Applied Probabilistic Programming & Bayesian Machine Learning

Glossary & Definitions

MIT IAP 2017 Max W. Shen

1 Fundamentals of Bayesian Statistics

Parameter:

Parameters are quantities that help in defining a probability distribution.

For example, a Bernoulli distribution is parameterized by θ between 0 and 1, and returns either 0 and 1. This can represent a coin flipped with probability θ of landing "1".

A normal distribution is parameterized by a mean μ and standard deviation σ. It's commonly written $Normal(\mu, \sigma)$ or $N(\mu, \sigma)$. Sometimes, μ is called a "location parameter" and σ is called a "scale parameter". These are intuitive names that describe the effect of changing these parameters on the shape of the distribution.

Likelihood Function:

A likelihood function is a function that take as input a set of parameters θ and outputs a probability distribution over some space. A likelihood function can be written as $p(y|\theta)$.

In practice, we use a likelihood function by inputting some values for parameters, and getting out the likelihood of observing certain outcomes. For example, if we have a likelihood function for coin-flips, and we give it an input parameter θ that says that the coin is weighted to land on heads 90% of the time, then it will output that observing mostly tails is very unlikely in a series of flips.

Bayes Rule:

Bayes rule takes the form of:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{1}$$

Given a likelihood function $p(y|\theta)$, Bayes rule instructs us how to update our distribution on model parameters θ after seeing some data y - we update it from $p(\theta) \rightarrow p(\theta|y)$.

Prior Distribution:

A prior ("before") distribution is the distribution on some quantity *before* seeing data, before being updated by Bayes rule. Within the framework of 1, the right-hand side, $p(\theta)$ is the prior distribution of θ (before seeing data).

We start with uncertainty in the true values of the model parameters, which we reflect by using a probability distribution instead of a discrete value to represent θ.

Posterior Distribution:

A posterior ("after") distribution is the distribution on some quantity *after* seeing data, after being updated by Bayes rule. Within the framework of 1, the left-hand side, $p(\theta|y)$ is the posterior distribution of θ (after seeing data).

After seeing data, we typically are more confident on the true value of our parameters θ, but there is still uncertainty. We reflect this by continuing to use a probability distribution to represent our parameters θ.

Inference:

Inference is the act of calculating Bayes rule to update our beliefs about parameters θ after seeing data. At a bare minimum, this involves finding $p(y|\theta)$, $p(\theta)$, and $p(y)$, and combining them according to equation 1. Inference is mathematically tricky for many models of interest due to $p(y)$, and until recently, was performed using pen and paper.

Model:

A model is the combination of our beliefs in our parameters θ denoted by $p(\theta)$, and our chosen likelihood function $p(y|\theta)$.

The word "model" is overloaded with many slightly different meanings, however. In practice, the act of writing a model or building a model is often associated only with choosing a likelihood function since there is a huge range of possibilities. Most complex prior/domain knowledge is expressed through the choice of likelihood function. However, our choices of prior distributions are just as important.

(Machine) Learning:

A Bayesian model "learns" by (1) updating its parameter values on data, and (2) iteratively improving our choice of likelihood function, such that our *combination of the likelihood function and our parameter values* - our model - becomes more predictive of future data, or capture structure in the data.

At each step of training, we retain a distribution over possible parameter values θ to reflect our uncertainty.

Predictions from the model can be made by simulating parameters from $p(\theta)$ and using those to simulate observations from $p(y|\theta)$. Alternatively, we can also make predictions by asserting some values of $p(\theta)$ and only simulating using $p(y|\theta)$.

Marginalization

$$\begin{aligned} p(Y=y) &= \sum_x P(Y=y, X=x) \\ p(Y=y) &= \int P(Y=y, X=x)dx \end{aligned} \tag{2}$$

We can expand $p(Y=y)$ to include the random variable X if we sum or integrate over all possible values that X can take, denoted x. This can be seen by noting that $\sum_x P(X=x) = 1$.

Chain Rule (Probability)

A joint distribution can be factorized into a product of conditional probabilities.

$$\begin{gathered} p(Y = y, X = x) = p(Y = y|X = x)p(X = x) \\ p(Y, X) = p(Y|X)p(X) \end{gathered} \tag{3}$$

Hierarchical Model

In Bayesian modeling, we place distributions over data which we represent as random variables. These distributions have some parameters associated with them. In a hierarchical model, we place distributions over these parameters, with their own "hyperparameters".

Hierarchical models are powerful because many structures in data can be represented hierarchically, and causal processes that generate data are also often hierarchical in nature.

Maximum likelihood estimation (MLE) is a fast and popular method in machine learning that is equivalent to finding the posterior mode given a flat prior. Importantly, MLE/MAP is often not feasible analytically with hierarchical models.

Parametric Bayes

This class falls under the category of parametric Bayesian methods, which works with probability distributions with a fixed number of parameters that does not grow with data.

In contrast, nonparametric Bayesian statistics works with distributions that have a flexible number of parameters that can grow with data. In the extreme, it is theoretically possible to have an infinite number of parameters with infinite data.

A common useful application of nonparametric Bayesian methods is clustering. Often, it's hard to say exactly how many clusters are best representative of data, and we would prefer the data to choose the number of clusters (thus the number of parameters) in our model.

Maximum Likelihood Estimation (MLE)

MLE is equivalent to finding the posterior mode assuming a model with flat priors. This can be solved analytically if conjugate priors are used. In practice, it is common to find the MLE by optimization via gradient descent and other numerical optimization methods.

In contrast to Bayesian methods that find the full posterior distribution, the MLE is a point estimate and cannot reflect uncertainty.

Maximum a Posterior (MAP) Estimation

MLE is equivalent to finding the posterior mode assuming a model with non-flat priors. These non-flat priors are usually thought of as L1 or L2 regularization. The

MAP estimate can be solved analytically if conjugate priors are used. In practice, it is common to find the MLE by optimization via gradient descent and other numerical optimization methods.

In contrast to Bayesian methods that find the full posterior distribution, MAP estimates are a point estimate and cannot reflect uncertainty.

2 Methods for Inference

Analytic Inference

Analytic inference means working on your posterior using Bayes rule using pen and paper, using algebra and calculus.

There are two main problems that arise when performing inference. First, multiplying two arbitrary probability distributions does not always have an analytic expression, which is a problem for an arbitrary likelihood function and prior distribution. Second, the denominator $p(y)$ must be expressed as:

$$\begin{aligned} p(y) &= \int_\Theta p(y, \theta) d\theta \\ &= \int_\Theta p(y|\theta)p(\theta) d\theta \end{aligned} \tag{4}$$

The integral in equation 4 can be intractable if Θ is very large.

Unnormalized Posterior

Since calculating $p(y)$ involves an intractable integral, we prefer to work with the unnormalized posterior:

$$\begin{aligned} p(\theta|y) &= \frac{p(y|\theta)p(\theta)}{p(y)} \\ &\propto p(y|\theta)p(\theta) \end{aligned} \tag{5}$$

Importantly, $p(y)$ does not depend on θ. This means if $p(\theta_1|y) > p(\theta_2|y)$ in the full posterior distribution, the relation also holds in the unnormalized density.

Also importantly, the unnormalized posterior is easy to calculate, even if it does not have a convenient analytical expression, since we can simply calculate the likelihood and the prior separately and multiply the two numbers.

We do not call it a distribution anymore, since it does not necessarily sum to 1.

Markov Chain Monte Carlo (MCMC)

MCMC is a method for approximate inference. It outputs an approximation to the full posterior distribution.

MCMC works by taking advantage of the fact that (1) we can easily calculate the unnormalized posterior at any specific θ, and (2) we can take random steps in the space of Θ.

As an example, we could use these two properties to find some θ^* that has the highest probability in the posterior. Each step, we are at some point θ_{now} and associated value $p(y|\theta_{now})p(\theta_{now})$, and we consider a candidate point θ_{cand} with associated value $p(y|\theta_{cand})p(\theta_{cand})$. We move to the candidate if it's better, otherwise we stay where we are. We will converge to some local or global maximum. However, this approach only gives us a point estimate.

MCMC instead is interested in the full posterior probability distribution. MCMC ensures that we move to a new point $\tilde{\theta}$ at a rate that is *proportional* to the true posterior probability at that point, $p(\tilde{\theta}|y)$. Conveniently, the unnormalized posterior is proportional to the true posterior probability! Then, if we take sufficiently many steps S, we can count how many times we stepped *into* some point θ_i. We call this count s_i. Then,$\frac{s_i}{S} \approx p(\theta_i|y)$. We can do this quickly for all points in Θ, giving us an approximation to the full posterior distribution which becomes exact as our number of steps approaches infinity.

MCMC was originally developed for statistical molecular dynamical simulations.

Another intuitive explanation is offered here:

`http://twiecki.github.io/blog/2015/11/10/mcmc-sampling/`

Hamiltonian Monte Carlo (HMC)

HMC is an state-of-the-art method for approximate inference. Similar to MCMC, it originates in physics where it was originally developed under the name of Hamiltonian dynamics, a reformulation of Newtonian mechanics.

HMC includes a momentum term which reduces correlation between consecutive steps and makes step proposals to more distant territory while still having a decent probability of acceptance. HMC runs faster than MCMC.

Variational Inference (VI)

Variational inference is another method for approximate inference. Compared to HMC, it is less precise since VI requires simplifying assumptions, but VI typically runs faster by orders of magnitude.

Automatic Differentiation

You are already familiar with symbolic differentiation, performed on paper, using the chain rule and other rules you learned in calculus I. Automatic differentiation is an algorithm that can quickly differentiate nearly any function, as long as it can be decomposed into a combination of a certain set of basic operations.

Differentiation is a core pillar of optimization since the maxima and minima of functions must exist at "critical points" where the derivative equals zero. Gradient descent, a ubiquitous optimization method, is a prime example of the success of simple differentiation-based methods in optimization.

For a long time, one major time sink in solving optimization problems was performing calculus by hand on paper, which can be tricky if not sometimes impossible, is prone to errors, and is difficult to debug.

Automatic differentiation is particularly useful with variational inference, since VI casts approximate posterior inference as an optimization problem.

Probabilistic Programming Language

There are many notions meant by ”probabilistic programming”, but the definition used in this class is a programming language that supports automatic, model-agnostic inference for Bayesian models.

Stan is one such language. Stan supports HMC and VI.

3 Class Material: Keywords and Selected Definitions

3.1 Lecture 1

Rhat

Rhat is an estimate of convergence of inference. An Rhat value is provided by Stan for each parameter it attempts to estimate. An Rhat value of 1.0 indicates convergence. If your model has unconverged parameters (rhat $\neq$ 1.0), n_eff can be increased by increasing the number of iterations, increasing the number of chains, or respecifying your model if n_eff is a small fraction of the total number of iterations run.

n_eff

n_eff is the effective iterations for inference on a parameter. Like Rhat, an n_eff value is provided by Stan for each parameter it attempts to estimate. A healthy model will have a high ratio of n_eff to the total number of iterations.

Regression

Identifiability

Regularization

Sparsity

Conjugate Priors

3.2 Lecture 2

Sensitivity Analysis

Posterior Predictions

Posterior Predictive Checks

Continuous Model Expansion

Information Criterion

Effective Parameters

Robust Models

3.3 Lecture 3

Exchangeability

Link Function

Generalized Linear Model

Covariance Matrix

Reparameterization

3.4 Lecture 4

Mixture Models

Nonparametric Bayes

3.5 Lecture 5

Bayesian Variable Selection

Splines

Missing Data Imputation

3.6 Lecture 6

Variational Inference

Edward

4 Bibliography and Additional Resources

This class draws much of its material from McElreath's Statistical Rethinking textbook and Gelman's Bayesian Data Analysis 3 textbook.

The Stan reference is available here: http://mc-stan.org/documentation/