Title: Probabilistic numerics

URL: https://en.wikipedia.org/wiki/Probabilistic_numerics

PageID: 69088046

Categories: Category:Applied mathematics, Category:Applied statistics, Category:Machine learning

---

Probabilistic numerics is an active field of study at the intersection of applied mathematics , statistics , and machine learning centering on the concept of uncertainty in computation . In probabilistic numerics, tasks in numerical analysis such as finding numerical solutions for integration , linear algebra , optimization and simulation and differential equations are seen as problems of statistical, probabilistic, or Bayesian inference . [1] [2] [3] [4] [5]

Introduction

A numerical method is an algorithm that approximates the solution to a mathematical problem (examples below include the solution to a linear system of equations , the value of an integral , the solution of a differential equation , the minimum of a multivariate function). In a probabilistic numerical algorithm, this process of approximation is thought of as a problem of estimation , inference or learning and realised in the framework of probabilistic inference (often, but not always, Bayesian inference). [6]

Formally, this means casting the setup of the computational problem in terms of a prior distribution , formulating the relationship between numbers computed by the computer (e.g. matrix-vector multiplications in linear algebra, gradients in optimization, values of the integrand or the vector field defining a differential equation) and the quantity in question (the solution of the linear problem, the minimum, the integral, the solution curve) in a likelihood function , and returning a posterior distribution as the output. In most cases, numerical algorithms also take internal adaptive decisions about which numbers to compute, which form an active learning problem.

Many of the most popular classic numerical algorithms can be re-interpreted in the probabilistic framework. This includes the method of conjugate gradients , [7] [8] [9] Nordsieck methods , Gaussian quadrature rules, [10] and quasi-Newton methods . [11] In all these cases, the classic method is based on a regularized least-squares estimate that can be associated with the posterior mean arising from a Gaussian prior and likelihood. In such cases, the variance of the Gaussian posterior is then associated with a worst-case estimate for the squared error.

Probabilistic numerical methods promise several conceptual advantages over classic, point-estimate based approximation techniques:

They return structured error estimates (in particular, the ability to return joint posterior samples, i.e. multiple realistic hypotheses for the true unknown solution of the problem)

Hierarchical Bayesian inference can be used to set and control internal hyperparameters in such methods in a generic fashion, rather than having to re-invent novel methods for each parameter

Since they use and allow for an explicit likelihood describing the relationship between computed numbers and target quantity, probabilistic numerical methods can use the results of even highly imprecise, biased and stochastic computations. [12] Conversely, probabilistic numerical methods can also provide a likelihood in computations often considered " likelihood-free " elsewhere [13]

Because all probabilistic numerical methods use essentially the same data type – probability measures – to quantify uncertainty over both inputs and outputs they can be chained together to propagate uncertainty across large-scale, composite computations

Sources from multiple sources of information (e.g. algebraic, mechanistic knowledge about the form of a differential equation, and observations of the trajectory of the system collected in the physical world) can be combined naturally and inside the inner loop of the algorithm, removing otherwise necessary nested loops in computation, e.g. in inverse problems . [14]

These advantages are essentially the equivalent of similar functional advantages that Bayesian methods enjoy over point-estimates in machine learning, applied or transferred to the computational domain.

Numerical tasks

Integration

Probabilistic numerical methods have been developed for the problem of numerical integration , with the most popular method called Bayesian quadrature . [15] [16] [17] [18]

In numerical integration, function evaluations f (x 1) , … , f (x n) {\displaystyle f(x_{1}),\ldots ,f(x_{n})} at a number of points x 1 , … , x n {\displaystyle x_{1},\ldots ,x_{n}} are used to estimate the integral ∫ f (x) ν (d x) {\displaystyle \textstyle \int f(x)\nu (dx)} of a function f {\displaystyle f} against some measure ν {\displaystyle \nu } . Bayesian quadrature consists of specifying a prior distribution over f {\displaystyle f} and conditioning this prior on f (x 1) , … , f (x n) {\displaystyle f(x_{1}),\ldots ,f(x_{n})} to obtain a posterior distribution over f {\displaystyle f} , then computing the implied posterior distribution on ∫ f (x) ν (d x) {\displaystyle \textstyle \int f(x)\nu (dx)} . The most common choice of prior is a Gaussian process as this allows us to obtain a closed-form posterior distribution on the integral which is a univariate Gaussian distribution. Bayesian quadrature is particularly useful when the function f {\displaystyle f} is expensive to evaluate and the dimension of the data is small to moderate.

Optimization

Probabilistic numerics have also been studied for mathematical optimization , which consist of finding the minimum or maximum of some objective function f {\displaystyle f} given (possibly noisy or indirect) evaluations of that function at a set of points.

Perhaps the most notable effort in this direction is Bayesian optimization , [20] a general approach to optimization grounded in Bayesian inference. Bayesian optimization algorithms operate by maintaining a probabilistic belief about f {\displaystyle f} throughout the optimization procedure; this often takes the form of a Gaussian process prior conditioned on observations. This belief then guides the algorithm in obtaining observations that are likely to advance the optimization process. Bayesian optimization policies are usually realized by transforming the objective function posterior into an inexpensive, differentiable acquisition function that is maximized to select each successive observation location. One prominent approach is to model optimization via Bayesian sequential experimental design , seeking to obtain a sequence of observations yielding the most optimization progress as evaluated by an appropriate utility function . A welcome side effect from this approach is that uncertainty in the objective function, as measured by the underlying probabilistic belief, can guide an optimization policy in addressing the classic exploration vs. exploitation tradeoff .

Local optimization

Probabilistic numerical methods have been developed in the context of stochastic optimization for deep learning , in particular to address main issues such as learning rate tuning and line searches , [21] batch-size selection, [22] early stopping , [23] pruning, [24] and first- and second-order search directions. [25] [26]

In this setting, the optimization objective is often an empirical risk of the form L (θ) = 1 N ∑ n = 1 N ■ (y n , f θ (x n)) {\displaystyle \textstyle L(\theta)={\frac {1}{N}}\sum _{n=1}^{N}\ell (y_{n},f_{\theta }(x_{n}))} defined by a dataset D = { (x n , y n) } n = 1 N {\displaystyle \textstyle {\mathcal {D}}=\{(x_{n},y_{n})\}_{n=1}^{N}} , and a loss ■ (y , f θ (x)) {\displaystyle \ell (y,f_{\theta }(x))} that quantifies how well a predictive model f θ (x) {\displaystyle f_{\theta }(x)} parameterized by θ {\displaystyle \theta } performs on predicting the target y {\displaystyle y} from its corresponding input x {\displaystyle x} .

Epistemic uncertainty arises when the dataset size N {\displaystyle N} is large and cannot be processed at once meaning that local quantities (given some θ {\displaystyle \theta }) such as the loss function L (θ) {\displaystyle L(\theta)} itself or its gradient ∇ L (θ) {\displaystyle \nabla L(\theta)} cannot be computed in reasonable time.

Hence, generally mini-batching is used to construct estimators of these quantities on a random subset of the data. Probabilistic numerical methods model this uncertainty explicitly and allow for automated decisions and parameter tuning.

Linear algebra

Probabilistic numerical methods for linear algebra [7] [8] [27] [9] [28] [29] have primarily focused on solving systems of linear equations of the form A x = b {\displaystyle Ax=b} and the computation of determinants | A | {\displaystyle |A|} . [30] [31]

A large class of methods are iterative in nature and collect information about the linear system to be solved via repeated matrix-vector multiplication v ■ A v {\displaystyle v\mapsto Av} with the system matrix A {\displaystyle A} with different vectors v {\displaystyle v} .

Such methods can be roughly split into a solution- [8] [28] and a matrix-based perspective, [7] [9] depending on whether belief is expressed over the solution x {\displaystyle x} of the linear system or the (pseudo-)inverse of the matrix H = A † {\displaystyle H=A^{\dagger }} .

The belief update uses that the inferred object is linked to matrix multiplications y = A v {\displaystyle y=Av} or z = A ■ v {\displaystyle z=A^{\intercal }v} via b ■ z = x ■ v {\displaystyle b^{\intercal }z=x^{\intercal }v} and v = A − 1 y {\displaystyle v=A^{-1}y} .

Methods typically assume a Gaussian distribution, due to its closedness under linear observations of the problem. While conceptually different, these two views are computationally equivalent and inherently connected via the right-hand-side through x = A − 1 b {\displaystyle x=A^{-1}b} . [27]

Probabilistic numerical linear algebra routines have been successfully applied to scale Gaussian processes to large datasets. [31] [32] In particular, they enable exact propagation of the approximation error to a combined Gaussian process posterior, which quantifies the uncertainty arising from both the finite number of data observed and the finite amount of computation expended. [32]

Ordinary differential equations

Probabilistic numerical methods for ordinary differential equations y ■ (t) = f (t , y (t)) {\displaystyle {\dot {y}}(t)=f(t,y(t))} , have been developed for initial and boundary value problems. Many different probabilistic numerical methods designed for ordinary differential equations have been proposed, and these can broadly be grouped into the two following categories:

Randomisation-based methods are defined through random perturbations of standard deterministic numerical methods for ordinary differential equations. For example, this has been achieved by adding Gaussian perturbations on the solution of one-step integrators [33] or by perturbing randomly their time-step. [34] This defines a probability measure on the solution of the differential equation that can be sampled.

Gaussian process regression methods are based on posing the problem of solving the differential equation at hand as a Gaussian process regression problem, interpreting evaluations of the right-hand side as data on the derivative. [35] These techniques resemble to Bayesian cubature, but employ different and often non-linear observation models. [36] [37] In its infancy, this class of methods was based on naive Gaussian process regression. This was later improved (in terms of efficient computation) in favor of Gauss–Markov priors [38] [39] modeled by the stochastic differential equation d x (t) = A x (t) d t + B d v (t) {\displaystyle \mathrm {d} x(t)=Ax(t)\,\mathrm {d} t+B\,\mathrm {d} v(t)} , where x (t) {\displaystyle x(t)} is a ν {\displaystyle \nu } -dimensional vector modeling the first ν {\displaystyle \nu } derivatives of y (t) {\displaystyle y(t)} , and where v (t) {\displaystyle v(t)} is a ν {\displaystyle \nu } -dimensional Brownian motion . Inference can thus be implemented efficiently with Kalman filtering based methods.

The boundary between these two categories is not sharp, indeed a Gaussian process regression approach based on randomised data was developed as well. [40] These methods have been applied to problems in computational Riemannian geometry, [41] inverse problems, latent force models, and to differential equations with a geometric structure such as symplecticity.

Partial differential equations

A number of probabilistic numerical methods have also been proposed for partial differential equations . As with ordinary differential equations, the approaches can broadly be divided into those based on randomisation, generally of some underlying finite-element mesh [33] [42] and those based on Gaussian process regression. [4] [3] [43] [44]

Probabilistic numerical PDE solvers based on Gaussian process regression recover classical methods on linear PDEs for certain priors, in particular methods of mean weighted residuals , which include Galerkin methods , finite element methods , as well as spectral methods . [44]

History and related fields

The interplay between numerical analysis and probability is touched upon by a number of other areas of mathematics, including average-case analysis of numerical methods, information-based complexity , game theory , and statistical decision theory . Precursors to what is now being called "probabilistic numerics" can be found as early as the late 19th and early 20th century.

The origins of probabilistic numerics can be traced to a discussion of probabilistic approaches to polynomial interpolation by Henri Poincaré in his Calcul des Probabilités . [45] In modern terminology, Poincaré considered a Gaussian prior distribution on a function f : R → R {\displaystyle f\colon \mathbb {R} \to \mathbb {R} } , expressed as a formal power series with random coefficients, and asked for "probable values" of f (x) {\displaystyle f(x)} given this prior and n ∈ N {\displaystyle n\in \mathbb {N} } observations f (a i) = B i {\displaystyle f(a_{i})=B_{i}} for i = 1 , … , n {\displaystyle i=1,\dots ,n} .

A later seminal contribution to the interplay of numerical analysis and probability was provided by Albert Suldin in the context of univariate quadrature . [46] The statistical problem considered by Suldin was the approximation of the definite integral ∫ u (t) d t {\displaystyle \textstyle \int u(t)\,\mathrm {d} t} of a function u : [a , b] → R {\displaystyle u\colon [a,b]\to \mathbb {R} } , under a Brownian motion prior on u {\displaystyle u} , given access to pointwise evaluation of u {\displaystyle u} at nodes t 1 , … , t n ∈ [a , b] {\displaystyle t_{1},\dots ,t_{n}\in [a,b]} . Suldin showed that, for given quadrature nodes, the quadrature rule with minimal mean squared error is the trapezoidal rule ; furthermore, this minimal error is proportional to the sum of cubes of the inter-node spacings. As a result, one can see the trapezoidal rule with equally-spaced nodes as statistically optimal in some sense — an early example of the average-case analysis of a numerical method.

Suldin's point of view was later extended by Mike Larkin. [47] Note that Suldin's Brownian motion prior on the integrand u {\displaystyle u} is a Gaussian measure and that the operations of integration and of point wise evaluation of u {\displaystyle u} are both linear maps .

Thus, the definite integral ∫ u (t) d t {\displaystyle \textstyle \int u(t)\,\mathrm {d} t} is a real-valued Gaussian random variable.

In particular, after conditioning on the observed pointwise values of u {\displaystyle u} , it follows a normal distribution with mean equal to the trapezoidal rule and variance equal to 1 12 ∑ i = 2 n (t i − t i − 1) 3 {\displaystyle \textstyle {\frac {1}{12}}\sum _{i=2}^{n}(t_{i}-t_{i-1})^{3}} .

This viewpoint is very close to that of Bayesian quadrature , seeing the output of a quadrature method not just as a point estimate but as a probability distribution in its own right.

As noted by Houman Owhadi and collaborators, [3] [48] interplays between numerical approximation and statistical inference can also be traced back to Palasti and Renyi, [49] Sard, [50] Kimeldorf and Wahba [51] (on the correspondence between Bayesian estimation and spline smoothing/interpolation) and Larkin [47] (on the correspondence between Gaussian process regression and numerical approximation). Although the approach of modelling a perfectly known function as a sample from a random process may seem counterintuitive, a natural framework for understanding it can be found in information-based complexity (IBC), [52] the branch of computational complexity founded on the observation that numerical implementation requires computation with partial information and limited resources. In IBC, the performance of an algorithm operating on incomplete information can be analyzed in the worst-case or the average-case (randomized) setting with respect to the missing information. Moreover, as Packel [53] observed, the average case setting could be interpreted as a mixed strategy in an adversarial game obtained

by lifting a (worst-case) minmax problem to a minmax problem over mixed (randomized) strategies. This observation leads to a natural connection [54] [3] between numerical approximation and Wald's decision theory , evidently influenced by von Neumann's theory of games . To describe this connection consider the optimal recovery setting of Micchelli and Rivlin [55] in which one tries to approximate an unknown function from a finite number of linear measurements on that function. Interpreting this optimal recovery problem as a zero-sum game where Player I selects the unknown function and Player II selects its approximation, and using relative errors in a quadratic norm to define losses, Gaussian priors emerge [3] as optimal mixed strategies for such games, and the covariance operator of the optimal Gaussian prior is determined by the quadratic norm used to define the relative error of the recovery.

Software

ProbNum : Probabilistic Numerics in Python.

ProbNumDiffEq.jl : Probabilistic numerical ODE solvers based on filtering implemented in Julia.

Emukit : Adaptable Python toolbox for decision-making under uncertainty.

BackPACK : Built on top of PyTorch. It efficiently computes quantities other than the gradient.

See also

Average-case analysis

Information-based complexity

Uncertainty quantification

References