Uncertain Context: Uncertainty Quantification in Machine Learning

Brian Jalaian, Michael Lee, Stephen Russell

■ *Machine learning and artificial intelligence will be deeply embedded in the intelligent systems humans use to automate tasking, optimize planning, and support decision-making. However, many of these methods can be challenged by dynamic computational contexts, resulting in uncertainty in prediction errors and overall system outputs. Therefore, it will be increasingly important for uncertainties in underlying learning-related computer models to be quantified and communicated. The goal of this article is to provide an accessible overview of computational context and its relationship to uncertainty quantification for machine learning, as well as to provide general suggestions on how to implement uncertainty quantification when doing statistical learning. Specifically, we will discuss the challenge of quantifying uncertainty in predictions using popular machine learning models. We present several sources of uncertainty and their implications on statistical models and subsequent machine learning predictions.*

The machine learning (ML) research community has given a great deal of attention to developing efficient algorithms that produce accurate predictions, estimations, and classifications — fundamental behaviors for artificial intelligence (AI; Etzion 2015). Although accurate outputs are the ultimate goal of ML algorithms, anything less than 100-percent accuracy is insufficient for more-sensitive or complex decision-making applications. Moreover, in complex applications, the problems are seldom closed domain, and are subject to contextual influences. One might argue that contextual dynamics or complexity are at the root of all ML and subsequently AI errors. In complex decision-making situations that can be impacted by context, quantification of underlying uncertainties in a system's output is necessary to establish trust, determine risk in alternatives, or communicate the potential for error. There are various sources of uncertainty in ML processes, including inherent noise in data, ambiguity or variance in model parameters, appropriateness of model selection, and vagueness due to extrapolation. We will illustrate the concept of uncertainty in ML by using the following example of extrapolation shown in figure 1. Figure 1 shows training observations (blue points), a regression model based on the training observations (blue line), standard error (faded-blue area around the blue line), and data observed after the model was created (red points).

To illustrate, in the context of figure 1's simple regression model, uncertainty occurs when we make predictions on new (red) data that is outside the range of the training data. This simple ML linear regression model was fit to the training (blue) data points. Given the regression model, the system could support a generalization that as *x* gets larger, *y* also gets larger.

Because the regression model was trained on the blue data, it fits those points reasonably well and predictions would be fairly accurate. The standard error provides some insight as to how well the model fits the data, clearly showing that the best fit is in the area of blue points that have the greatest quantity with the least amount of variance. If the system were to observe a new point that fits in the area of the blue points, the prediction would be fairly reliable. Moreover, the standard error in this situation can provide some degree of uncertainty associated with the predicted output. Further, if a prediction and subsequent observation was made anywhere along the right side of and close to the blue line (for example, where $x = 5$), a fairly accurate prediction could be made and reliably characterized with the standard error. This is true with the assumption that all possible observations would follow a linear pattern similar to the distribution of the blue points.

Now, what if the range of possible observations do not follow the expected pattern of the model? In figure 1, this case is illustrated with the red points. When considering the red points, uncertainty related to a model prediction becomes difficult to quantify because the observations make both the model prediction and the standard error grossly inaccurate. In this case, no insight could be provided about the quality of the machine-learned output. More specifically, the quality of the machine-learned output is completely unknown until such a (contextual) time as an observation is obtained that indicates there is a potential issue, which usually occurs at runtime and can cause catastrophic system problems. This type of problem is frequently the case with contemporary ML algorithms and subsequent AI, leading to the rise of adversarial exploitation or disastrous errors or in the best case, greater ambiguity in risk and decision-making (Russell and Moskowitz 2016).

This simple example shows the problems associated with machine learners not intrinsically handling the sources of contextual and other uncertainty in a way that closely aligns with the learners' implementation data architecture. In more complex systems or systems-of-systems, issues of contextual uncertainty can pose greater system liabilities because the ML models are typically part of an ensemble ML model or composite AI (Hyden, Ioup, and Russell 2011). As such, contemporary AI systems and intelligent decision support systems are vulnerable to many unforeseen risks in providing even local decision guidance, due to limitations in training contexts. This vulnerability is particularly true of modern optimization methods and intelligent systems that augment human decision-making.

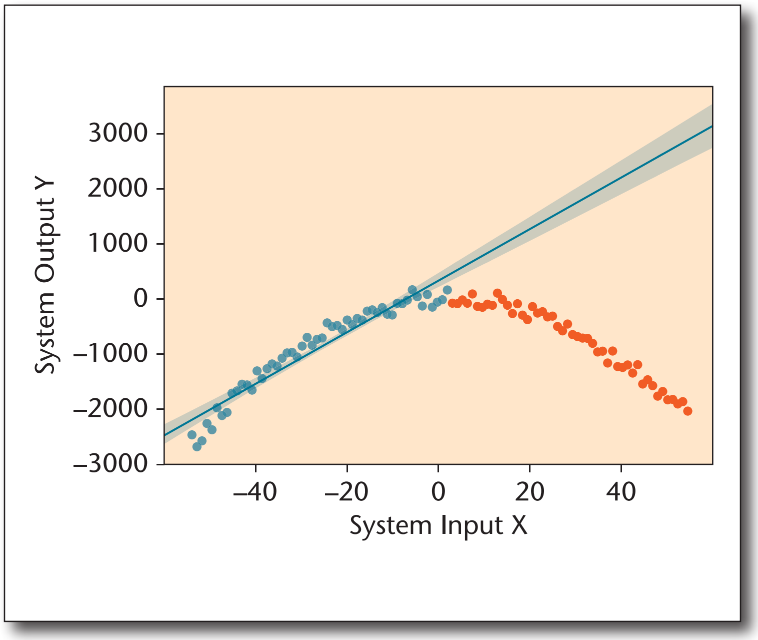


Figure 1. An Illustration of the Concept of Uncertainty in ML.

Initial observations as blue points, model as linear regression, and second set of observation in red points after creating the model based on initial dataset (blue points). The uncertainty estimates in faded blue are very inaccurate due to extrapolation error.

Computational Context and Uncertainty

To better understand uncertainty's relationship to AI and ML, it is important to have a basic understanding of computational context. Context has many definitions ranging from environment to situation. Clark and Carlson (1981) suggest that the term *context* is useful because it is sufficiently vague, general, and can accommodate many different ideas. From a computational perspective, most definitions of context narrow the environment or situational definition to characteristics of an entity's existential or operational domain (Schilit, Adams, and Want 1994). Dey (2001) defines context as any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object under consideration. This definition places appropriate computational emphasis on information, making this definition much more relevant for a computational context. The notion of information that characterizes the environment or situation is highly relevant to any computational model, let alone a machine learner or ensemble of them that form AI. Computational context is one reason there is such emphasis on feature selection in the ML research community. It is noteworthy that the literature for converging context and ML methods in any general sense is surprisingly quite sparse. In Google Scholar searches, at the time of this writing, most of the co-occurrences of context and

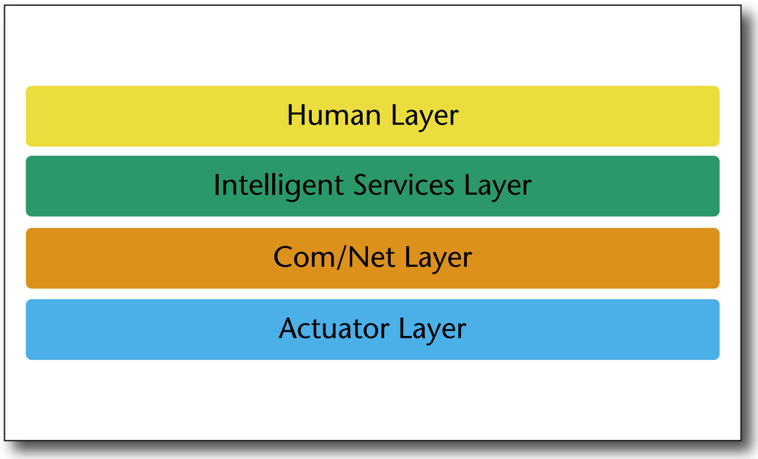


Figure 2. Cross-Layered Autonomous Decision Support Architecture.

ML and AI appear only as bounds on the ML or AI approach or discussion (for example, XXX AI or YYY learning within the context of ABC application area).

The definitional ambiguity referenced by Clark and Carlson (1981) is precisely why context can be associated with complex information, and therefore increase uncertainty in ML outputs. It is in the area of complex information that context becomes a dominant factor in understanding how the information fits within the current situation, the information relationships, the development of the situation in the future, and any related predictions about interdependent effects (Russell, Moskowitz, and Raglin 2017). Ambiguity in computational context can lead to significant errors and when those errors become unexpected or excessive, significant uncertainty. In terms of computational context, uncertainty can take two forms: stochastic uncertainty, which occurs because ML can behave in many different ways due to variable input data; and subjective uncertainty, which arises from a lack of knowledge about features, parameters, or conditions given ML's computational implementation.

Consider the following constraints in designing and implementing a ML algorithm for a decision-making application. It is a safe, but not always accurate, assumption that that a ML model builder had the requisite contextual expertise to validate the ML technique with its associated parameters. It is also possible, but unlikely, that the ML algorithm could have been trained on all the possible data. Further, the system in which the ML was implemented may, but at runtime also may not, be sufficiently stable and inerrant (Etzion 2015). If the optimistic view of these constraints is met, then uncertainty in the output would be consistently deterministic. However, the likelihood of all these constraints being met is low, particularly for nonclosed domain problems. Moreover, the dependence on appropriateness and specificity in training data makes many ML models brittle or limited to overspecified and closed domains, in other words, narrow-context problems that are characteristic of locally isolated or small systems. In more complex applications, these constraints for deterministic uncertainty are lofty objectives to meet and are arguably unrealistic in any generalizable system context. Yet, progress in addressing these issues in a manner applicable to ML has been made in the research domain of optimization.

Optimization under Uncertainty

Contemporary intelligent decision support and AI automation comprises a diverse and vast array of computationally capable sensors, actuators, networks, and information sources, which can provide context. These components are often combined with ML models that have varying degrees of intelligence, capability, and bailiwick. Beyond algorithmic accuracy, the efficacy of the output of these complex system-of-systems are constrained by energy, power, computing, and communication resources (Zhang, Liu, Samani, and Jalaian 2015). The layering diagram shown in figure 2 presents an architecture that describes the constraints in notional system functionality as layers. This diagram abstracts the underlying complexity shown in prior research (Nagaraj and Pasupathy 2017) that studied cross-layer optimization and extended those notions to a system-level model representing the architecture that exists in intelligent decision support systems (Liu et al. 2015; Kaiwartya et al. 2016; Beans 2018). Given a layered architecture such as shown in figure 2, which incorporates computational context in stochastic variables, tractable mathematical models can be developed that optimize the performance of assets and services with respect to decision objectives (Jalaeian, Zhu, Samani, and Motani 2016; Jalaian et al. 2017).

To develop these mathematical models, it is necessary to (1) identify important parameters and decision variables that characterize the interconnections and interdependencies between layers; (2) develop the mathematical constraints that capture the cross-layered interactions and trade-offs between these key variables, that is, express underlying complex system rules, behaviors, and the uncertainty that exists in the system models; and (3) cast the specific service output as equivalent mathematical objectives (Zhu and Azar 2015). We extend this work by including quantified uncertainty in the expression of the underlying system phenomena, such that they can be propagated to downstream models and global considerations. In this manner, the uncertainty propagating in the system can be modeled as a set of deterministic and stochastic mathematical constraints, which express how the value and uncertainty of one parameter or variable impacts other layers' parameters and the overall objective of the system.

Decision choices (system or human) can be considered as utility functions because they must focus on an objective outcome (Kreps 2018). Depending on the utility function, this may mean the objective activity is to maximize or minimize utility. If local system decisions are considered as a utility function, a function $f(x, y, \omega)$ can be defined. In this manner, uncertainty can be incorporated and evaluated. The performance of function f can be evaluated through stochastic simulation for a particular instance of the continuous inputs x, discrete inputs y, and a realization of the random variables in the simulation, the vector ω, which may or may not be a function of x and y. If discrete-event simulation is used to evaluate f, depending on the global decision objectives, the simulation may be partially accessible in algebraic form or may be purely available as an input–output modality (that is, a black box); it may have single or multiple outputs; it may have deterministic or stochastic output(s); it may involve discrete or continuous parameters; and it may or may not involve explicit or implicit or hidden constraints.

If the utility function is formulated in this manner, it becomes possible to model local system characteristics in an abstract closed form that can be exercised and evaluated before implementation. Further, this treatment can be cast as a decision-making under uncertainty optimization problem, where the expected value of the utility function can be minimized or maximized, subject to constraints; for example, those occurring from layer interdependencies and interactions. In this context, the expected value of a vector-valued function g can be considered: $\mathbb{E}_\omega [g(x, y, \omega)] \leq 0$.

The constraints defined by g can also be evaluated with each simulation run over g. In this formulation, expected values for these stochastic functions are used. Similarly, there may be other constraints, as well as bound constraints on the decision variables, which do not involve random variables that would be represented by a function $h(x, y) \leq 0$. Each constraint may be thought of as a representation of additional outputs from the simulation that needs to be taken into consideration. Additionally, in this manner bound constraints may be imposed on the decision variables available or obtained from domain-specific knowledge. Discrete variables may either be binary, integer-ordered, or categorical, and lie in a discrete space. This formulation assumes that f is a real-valued function and g is a real vector-valued function, both of whose expected values may or may not be smooth or continuous functions. In context, this formulation could optimize the expected value of any ML or AI metric that was relevant to the model as its utility function. For instance, an objective function that minimizes the risk of excessive resource consumption might be an alternative function — in which case, it would be important to incorporate a variance measure in the objective function. Like most optimization problems, the inclusion of additional constraints and variables increases the running time of the solver necessary to provide a system implementation for quantitative uncertainty in its ML model(s) (Hutter, Xu, Hoos, and Leyton-Brown 2014). Moreover, runtime is worsened, often nonlinearly, if the system requires an ensemble solution or is part of an interdependent system-of-systems. This situation suggests it is important to consider solver runtimes that can provide uncertainty quantification (UQ) for complex ML and AI systems.

Given the potential complexity of this formulation, stochastic simulation optimization runtimes required to quantify the uncertainty in an intelligent decision support system's learning models may be excessive. However, the high number of runs are necessary to capture the broadest range of computational context, to quantify uncertainty. Although there may be many approaches to reduce runtimes for optimization formulations (Akimoto, Astete-Morales, and Teytaud 2015), such as the one described here, even verifying the feasibility of a potential solution remains a challenge. Nagaraj and Pasupathy (2017) recently proposed a random restart algorithm that repeatedly executes a gradient-based simulation optimization routine on strategically relaxed sample-path problems to return a sequence of local solution estimators at increasing precision. This proposed method is called *cgr-SPLINE*, which implements stochastically constrained simulation optimization on mixed-integer spaces; this algorithm would be a candidate solution for implementing the above formulation of optimizing resource allocation and scheduling with uncertainty incorporated. The cgr-SPLINE method addresses the relatively unexplored problem of simulation optimization in the presence of stochastic constraints, making it an ideal candidate solution solver for the uncertainty formulation herein. However, even given that uncertainty can be quantified and can be ascertained a priori, the issue of stochasticity in statistical learning remains a challenge. Specifically, stochastic optimization approaches are often the correct choice for optimizing the training objective where global optimization of a system's deterministic objective involves the uncertainty in training data (Srebro and Tewari 2010).

Statistical Learning and Stochastic Optimization

The previous discussion illustrates an optimization method to handle variability in computational context manifesting as uncertainty in real environments that involve empirical data. Such an approach handles uncertainty by using methods of probability and decision theory, but would be limited by incomplete data on which to statistically learn. In a pragmatic sense, statistical learning is the ability for a system to extract statistical regularities from observations and then use these statistics to generalize solutions to new problems or unforeseen observations. From an application perspective, *statistical learning* refers to a set of tools for modeling and understanding complex datasets. It is a recently developed area in statistics and blends with parallel developments in computer science and, in particular, ML (James, Witten, Hastie, and Tibshirani 2013).

In statistical learning, data are given in the form of independent and identically distributed samples $\mathbf{x}_n$ of a random variable $\mathbf{x} \in Y$. Based upon $\mathbf{x}_n$, we would like to estimate a target $\mathbf{y}_n$, which is an independent and identically distributed sample random variable $y \in Y$. Here, X is typically called the feature space, and Y is called the target domain. In terms of explicit ML approaches, for example, the target domain may be a discrete set $\{1, \ldots, C\}$ of $\mathbf{y}_n$ in the case of classification (for example, is the set of observations *A* independent or interdependent from observations *B*) or $Y \subset \mathbb{R}$ in the case of regression (for example, see figure 1). Ideally, the selected estimator would be one with a minimal number of mistakes in expectation over all data, also known as the *statistical error rate*.

There are two fundamental issues in minimizing the statistical error that make solving the minimization intractable. One, it is at least as hard as the hardest of problems for a nondeterministic polynomial to optimize over an integer-valued stochastic function. Two, the feasible set, when given a generic function space without any structure, is mathematically impossible to fully optimize over (Murty and Kabadi 1987). Researchers have addressed both of these issues in a variety of ways; the work focused on the latter issue has led to the rich field of supervised ML. There is also work on a unifying method to address both of these fundamental issues. This work applies a convex loss function to the estimator in the statistical error rate. Using Y^X to denote the space of all functions from feature space X to target domain Y, consider a convex loss function $\ell :\in Y^X \times Y \rightarrow \mathbb{R}$, which becomes small when the estimator, typically denoted as $\mathbf{y}\hat{}(\mathbf{x})$ in the literature, is close to $\mathbf{y}$, and large when far apart. Doing this yields the general learning setting of Vapnik (1995), key to drawing better inferences statistically.

The challenge of the general learning setting is that stability is often necessary for learning. Stability has also been suggested as an explicit alternate condition for learnability. Intuitively, stability notions focus on particular algorithms, or learning rules, and measure their sensitivity to perturbations in the training set (Shalev-Shwartz, Shamir, Srebro, and Sridharan 2010). This sensitivity is particularly true in the absence of computational constraints, where the minimizer of a sample average of observed data is commonly referred to as either the empirical risk minimizer or the M-estimator. The empirical risk minimizer is an often-used estimation strategy because of its desirable statistical convergence properties (Frostig, Ge, Kakade, and Sidford 2015). To this end, the standard approach to statistical learning theory is based on assumptions chosen arguably for convenience (for example, independent and identically distributed or stationary ergodic–ergodic systems or data have the same behavior averaged over time and space; for example, a resistor's resistance-value, averaged over time and in different locations, is stable). The notion of independent and identically distributed samples, or highly-inferable probabilities, emboldens a further assumption that achieving the objective is at least possible in that problem domain. Therefore, the literature in approaching the problem of (formalized) learning restricts the focus to those scenarios in which learning is possible. This assumption that learning is possible is clearly an intuitive assumption, but it is one that may not always be valid in a formalized implementation of a theory of learning.

UQ in ML

ML has seen widespread use across countless domains. From election predictions to movie and music recommendations, ML has become an indispensable tool for making inferences based on what has been observed. The basic ML problem is as follows: We have p observed predictor variables $x_1, x_2, \ldots x_p$, which we would like to use to predict the value of some response variable Y (for example, figure 1). However, ML is typically restricted to computational contexts where it is assumed that we have access to sufficient training data and where both the predictor values and response values are known for many examples (for example, for a problem with a known solution). It is the hope that our model can learn from those examples and generalize to unseen data.

For regression problems, Y is a continuous variable (for example, a stock price), and we would like to predict a future Y value based on current observations $x_1, x_2, \ldots, x_p$. For classification, Y is categorical, and we are interested in predicting the correct class label. Two canonical examples of classification are (1) predicting whether an e-mail is spam or not based on its content, and (2) diagnosing a patient given factors such as age, weight, blood pressure, and so forth. For example, if a patient is older, overweight, and has high blood pressure, a well-trained model would likely output a higher probability of heart disease compared with a patient without those characteristics.

Much attention in ML is focused on studying how specific models work (for example, neural nets) so that prediction accuracy is improved. Accuracy is, without question, an important measure of the statistical virtue of a ML approach. Yet, accuracy gives no bounds on the quality of the machine learner, in terms of its contextual generalizability. Although the literature on UQ for traditional statistical models is vast, UQ has yet to become standard in ML practice, likely for several reasons. In many applications, primarily closed-domain problems, it is simply the case that UQ is not a priority. Consider a ML algorithm that recommends music based on a user's listening history. The algorithm's main focus will likely be finding good predictions (songs), rather than trying to quantify how much the user will like a particular song. In other situations, such as those that have contextual complexity, UQ can be a matter of life or death. If a model is used to predict whether a patient has a particular disease, the patient would be very interested in how confident the model is in its output; a simple yes or no would likely be insufficiently informative.

Another reason UQ may not be widespread is that for many popular ML methods, UQ is difficult to implement. For example, the reason there are not simple formulas for confidence intervals of neural network predictions (as there are for, say, linear regression) is because UQ for complex models requires addressing challenges of dynamics in computational context and may have multiple sources of uncertainty.

Four Sources of Uncertainty

In this section, we will examine four sources of uncertainty: noise, parameter uncertainty, uncertainty in model specification, and uncertainty due to extrapolation.

Noise

In almost any modeling problem, there is some sort of inherent noise in the data generating process (Scott, Ingalls, and Kaern 2006). Even when trying to estimate a deterministic physical law, there might be deviations due to measurement error or other contextual dynamics. Figure 3 illustrates a situation where the model choice is correct, the model has been trained well, but there is a lot of noise in the data. Given a value on the *X*-axis, the blue line indicates the model's prediction for the response variable (on the *Y*-axis). Although using this model would yield better predictions than no model at all, the amount of noise present precludes making accurate predictions.

Querying any model for a prediction only yields a single value. For instance, if the model for figure 1 was given an *x*-value of 7.5, it would predict a response value of ~7. Without any UQ, the practitioner has no sense of how confident the model is. Someone who does not have much experience with statistical models may even attribute a high model confidence in the prediction. For this situation, a quantification of the noise would likely be of much use to the practitioner. Instead of simply giving the prediction $y = 7$ when $x = 7.5$, it is helpful to supply the prediction and a margin of error such as 7 ± 4. The interpretation of the interval depends on the context and what kind of interval one constructs, but essentially the purpose is to give some measure of uncertainty in the prediction. Fortunately, several methodologies have been developed to quantify this kind of uncertainty for a wide range of models (Cai and Wang 2011). We will look at two particular examples later.

Parameter Uncertainty

Most ML procedures have parameters that determine their exact form (Murphy 2012). For example, with simple linear regression, the parameters are the slope and intercept of the regression line. For neural networks, the parameters are the weight matrices and bias terms. ML attempts to estimate the true parameters of the model (for example, the true slope and intercept). Of course, with a finite amount of data, we can never know the exact value of the parameters, and this leads to contextual ambiguities. There will always be some error — and thus, uncertainty — with respect to the parameters. Figure 4 illustrates this type of error. In figure 4, the dashed red line indicates the true regression line, that is, the line with the true parameters (slope and intercept). We record 50 observations, the green points, and fit the standard regression line to those observed data. The solid green line represents this fit. As can be seen, the green line is close to the true regression line, but deviates a bit. It is possible to show theoretically that the more data available, the more the closer the blue line would be to the red line, and less uncertainty would exist.

As with uncertainty due to noise, estimating parameter uncertainty has long been a focus for statisticians (Murphy 2012). They have developed numerous ways to quantify this kind of uncertainty for a wide variety of models. It is worth pointing out that there are two major schools of thought on how best to model parameter uncertainty. The two schools are Frequentism and Bayesianism. Roughly, Frequentists assume that model parameters are fixed constants, and that one should only use information from the random sample to draw inferences about the parameters. Bayesians, on the other hand, model the uncertainty about the parameters with probability. They might claim there is a 0.95 probability that the true slope is between 0.5 and 1.3, whereas a Frequentist would not make any probability statements about a parameter (because their model parameters are fixed). Perhaps the main criticism of the Bayesian school is that Bayesians incorporate prior knowledge into their models, which some think is too subjective for rigorous research. Nevertheless, it is hard to dispute the fact that the Bayesian paradigm provides an elegant way of modeling parameter uncertainty in a wide variety of settings. We will explore a specific example that concretely distinguishes these two schools of thought in the next subsection.

Uncertainty in Model Specification

Although the two previous sources of uncertainty have a history of substantive quantitative methodology, the third source of uncertainty is not easily quantified. In practice, to establish context, tools such as visual aids, diagnostics, and domain knowledge can be used to help determine the proper model, but it is impossible to know its true form. For example, in figures 3 and 4, the models fit reasonably well, because the used model — simple linear regression — matches the true model of the data.

Figure 5 gives an example where the data has a quadratic relationship, but a linear model was fitted. The error bands given by the regression model are clearly unreliable. For instance, a Bayesian credible interval might give the conclusion that there is a 0.95 probability that the future value will fall between –220 and –240 given future observed input $x = 220$ even though, in actuality, the probability is nearly zero. This mistake is because the linear regression model assumes the data are linear. If the data are not linear, then any type of UQ will have inaccurate

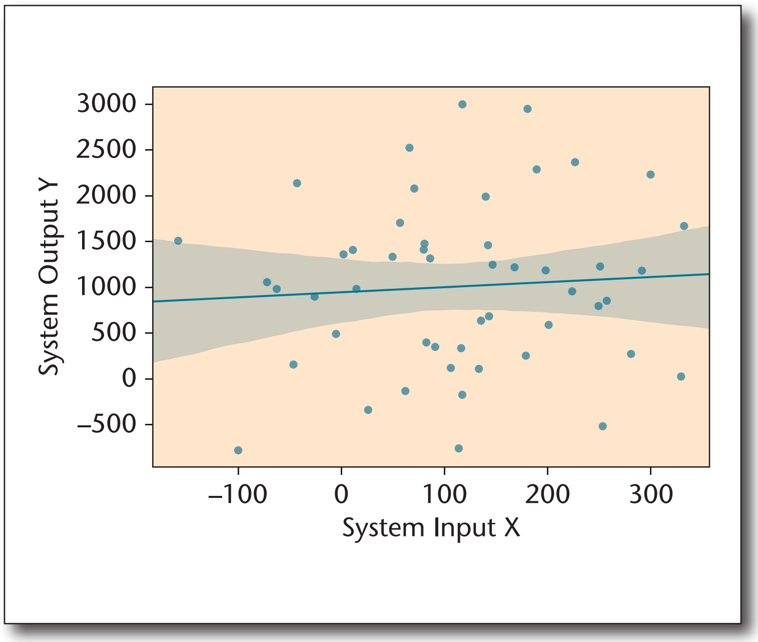


Figure 3. Uncertainty Due to Noise.

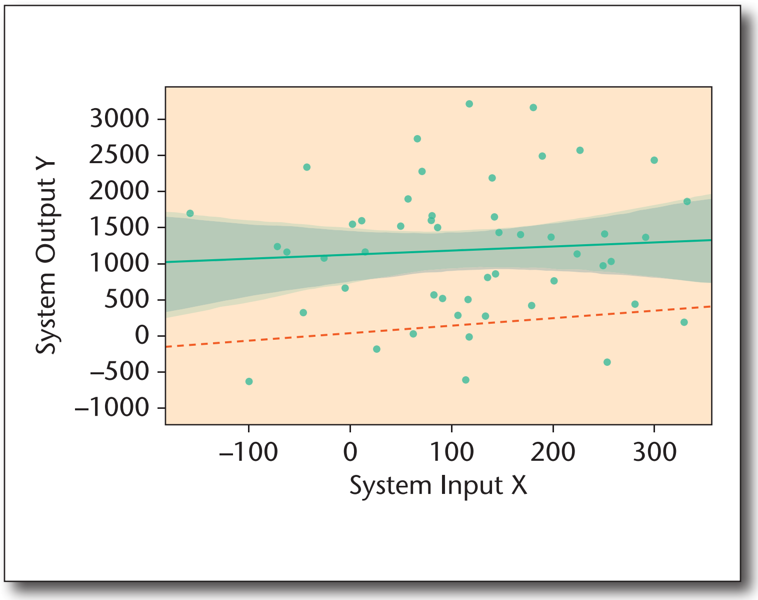


Figure 4. Parameter Uncertainty.

results. Certainly, a trained practitioner would have plotted the data and seen that the relationship is clearly not linear, in which case they would have tried a quadratic model and observed a much better fit. Using models as contextual black boxes can lead to highly inaccurate results if one does not do some exploratory analysis or model assessment, and this is becoming an increasing practice in ML applications (Papernot et al. 2017).

Unfortunately, there is no widely accepted way of measuring how wrong a model choice may be, given an arbitrary decision or learning problem (Livingston et al. 2015) — indeed, such a quantification seems pragmatically impossible. However, there exists more flexible models than those such as illustrated in figure 5. The models can be shown to be robust for a wide variety of computational contexts. Nonparametric models make few assumptions about the data, and can be very flexible. Of course, this flexibility comes at a cost. For large, high-dimensional, or sparse datasets (a common problem in ML), the performance of nonparametric methods becomes very variable. Slight changes in the data can yield wide fluctuations in a model's prediction accuracy. In contrast, simpler models, such as linear regression, can still perform well in these scenarios, as they are much more robust to slight changes in data. The decision of what type of model to use is an art as much as it is a science, and is largely domain-specific (Livingston et al. 2015). Some fields, such as Economics, may rely heavily on linear regression, whereas other contexts with greater variability, such as image recognition, may use neural networks or support vector machines. In any event, one cannot assume that a fitted model will automatically work well in all computational contexts.

Uncertainty Due to Extrapolation

The final source of uncertainty that we highlight is uncertainty due to extrapolation. This uncertainty occurs when predictions are made on new data outside the range of the training data. Figure 6 illustrates this scenario. In figure 6, a Gaussian process regression (GPR) model was fit to the training (red) data points. Because the regression model was trained on the red data, it fits those points reasonably well. If we were to observe a point within the red data, our prediction would be fairly reliable. However, if we were to try and extrapolate — to make a prediction for points outside of the observed data — the prediction would be inaccurate. The GPR model does not have enough data to be confident on its prediction. This fact can be seen at *x* larger than 8 as the faded-blue area is getting larger, which represents high uncertainty.

As with model uncertainty, there is no accepted way to measure uncertainty due to extrapolation without making more model assumptions (Steel 2011). The common way to handle this type of uncertainty in practice is to flag any new data values that are far away from the rest of the data and give them an outlier treatment. A naive way of accomplishing this would be to report any new values that fall outside (the convex hull) of the data on which the model was trained.

GPR model incorporates extrapolation error to reduce uncertainty. By incorporating extrapolation error, GPR can handle a greater degree of contextual variability. GPR is a nonparametric Bayesian method that is flexible and has useful properties (Quiñonero-Candela and Rasmussen 2005). In figure 6, we show a basic example where we have fit a GPR model to some observed data, such as sparse observations of sensor data. These are represented by the six red dots. The blue line represents the model's predictions of *y* based on the values of *x*, whereas the faded-blue region gives upper and lower standard error bounds, which attempt to quantify uncertainty in the model's predictions.

Examining figure 6, the model is clearly flexible in how closely it fits irregular observations. It fits a smooth curve through the observed data in an attempt to guess the underlying function. The model does not assume that the true function is linear, quadratic, or any other strict functional form; it only makes weaker assumptions on how smooth the function is. The model also indicates that for areas where little data has been observed, a higher uncertainty is provided, as evidenced by the wider faded-blue region such as occurs on the plot's far left and right. In this manner, predictions are provided with a degree of uncertainty, and thereby, provide additional insight into differences between the model and variations in the computational context.

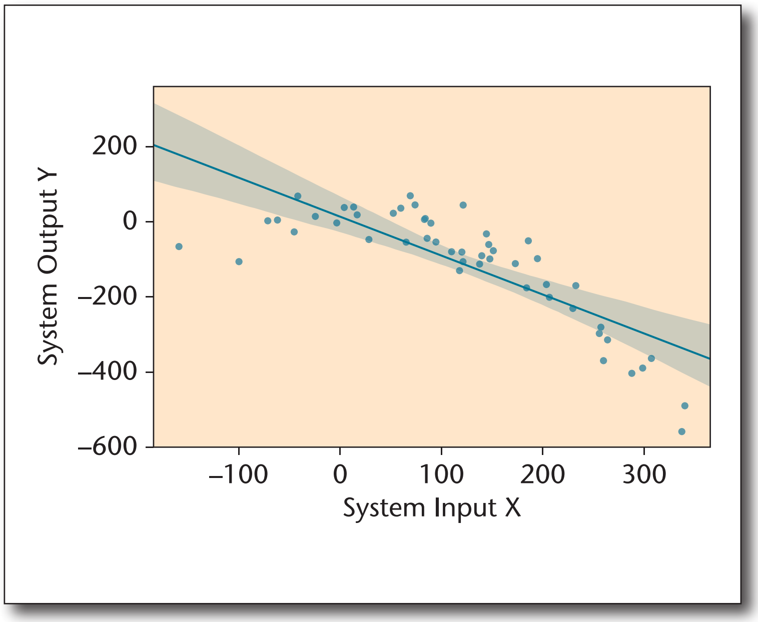


Figure 5. Model Uncertainty.

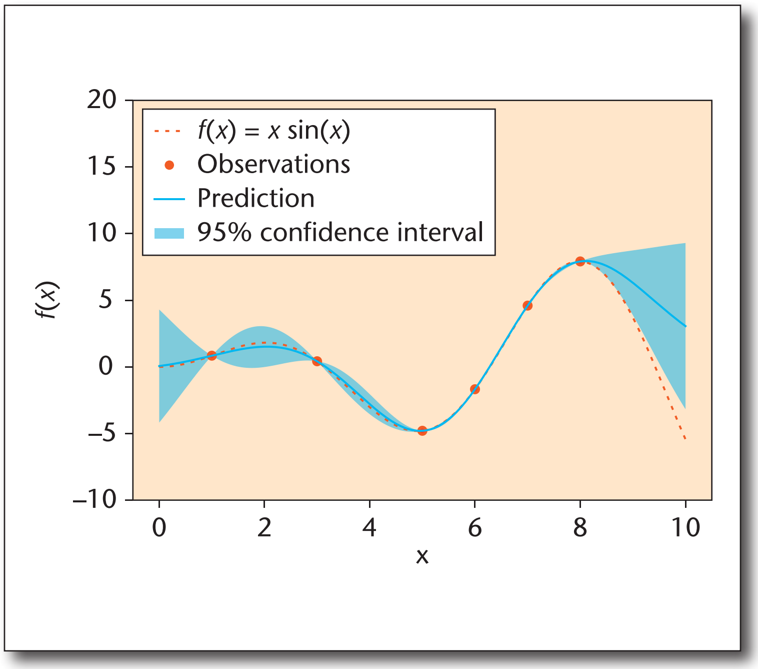


Figure 6. GPR.

Conclusion

The importance of computational context has increased in direct correlation with the complexity of the intelligence decision support systems and the ML and AI that underlie them. Accounting for the variabilities in algorithmic problem domains, underlying training data, model applicability, and pragmatic implementations, all have to be addressed before generalizable learners and AI can be realized. Contextual dynamics and complexity create errors in the systems that cannot adapt to the variability or do not provide insight into the system's own uncertainty about its learning or reasoning mechanics. Although contemporary ML/AI systems may be some distance away from broadly generalizable context-awareness, a degree of UQ can be achieved today.

The topic of UQ is not new to the statistical learning community, but it is often overlooked in the rapidly developing literature of ML. The lack of research attention on UQ for ML might be caused by a growing demand for heuristic ML methods that work well enough for narrow applications where errors in system predictions do not have significant consequences. On the other hand, UQ in ML will be increasingly essential, as ML/AI applications are being used in more-complex and critical problem domains and contexts.

In this article we provided an introduction to uncertainty and discussed its relationship to computational context. We also provided an overview of UQ for ML, as well as providing general suggestions on how to implement UQ in statistical modeling methods. Specifically, the challenge of quantifying the uncertainty in predictions using statistical or ML models was presented. Although not a comprehensive list, several sources of uncertainty and their implications on statistical models and subsequent ML predictions were also presented. As an early contribution to an emergent and necessary field, this review of the literature and discourse shows that numerous applications could benefit from a better understanding of UQ, especially those involving complex information problems or where computational context is particularly dynamic.

References

Akimoto, Y.; Astete-Morales, S.; and Teytaud, O. 2015. Analysis of Runtime of Optimization Algorithms for Noisy Functions over Discrete Codomains. *Theoretical Computer Science* 605: 42–50. doi.org/10.1016/j.tcs.2015.04.008.

Beans, C. 2018. Science and Culture: Sentient Architecture Promises Insight into Our Evolving Relationship with AI. Proceedings of the National Academy of Sciences 115(30): 7638–40. doi.org/10.1073/pnas.1809390115.

Cai, T. T., and Wang, L. 2011. Orthogonal Matching Pursuit for Sparse Signal Recovery with Noise. IEEE Transactions on Information Theory 57(7): 4680–8.

Clark, H. H., and Carlson, T. B. 1981. Context for Comprehension. *Attention and Performance* IX: 313–30.

Dey, A. K. 2001. Understanding and Using Context. *Personal and Ubiquitous Computing* 5(1): 4–7. doi.org/10.1007/s007790170019.

Etzion, O. 2015. When Artificial Intelligence Meets the Internet of Things. In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems*, 246. New York: Association for Computing Machinery. doi.org/10.1145/2675743.2774216.

Frostig, R.; Ge, R.; Kakade, S. M.; and Sidford, A. 2015. Competing with the Empirical Risk Minimizer in a Single Pass. Paper presented at the Conference on Learning Theory, 728–63. arxiv.org/pdf/1412.6606.pdf

Hutter, F.; Xu, L.; Hoos, H. H.; and Leyton-Brown, K. 2014. Algorithm Runtime Prediction: Methods and Evaluation. *Artificial Intelligence* 206: 79–111. doi.org/10.1016/j.artint.2013.10.003.

Hyden, P.; Ioup, E.; and Russell, S. 2011. Communicating Uncertainty Information Across Conceptual Boundaries. In *Proceedings of the 2011 Simulation Winter Conference*, 1096–102. Piscataway, NJ: Institute of Electrical and Electronics Engineers. doi.org/10.1109/WSC.2011.6147832.

Jalaeian, B.; Zhu, R.; Samani, H.; and Motani, M. 2016. An Optimal Cross-Layer Framework for Cognitive Radio Network under Interference Temperature Model. *IEEE Systems Journal* 10(1): 293–301. doi.org/10.1109/JSYST.2014.2342224.

Jalaian, B.; Yuan, X.; Shi, Y.; Hou, Y. T.; Lou, W.; Midkiff, S. F.; and Dasari, V. 2017. On the Integration of SIC and MIMO DoF for Interference Cancellation in Wireless Networks. Wireless Networks 24(7): 2357–74.

James, G.; Witten, D.; Hastie, T.; and Tibshirani, R. 2013. *An Introduction to Statistical Learning*, Vol. 112. Berlin: Springer. doi.org/10.1007/978-1-4614-7138-7.

Kaiwartya, O.; Abdullah, A. H.; Cao, Y.; Altameem, A.; and Mukesh Prasad, C. 2016. Internet of Vehicles: Motivation, Layered Architecture, Network Model, Challenges, and Future Aspects. *IEEE Access: Practical Innovations, Open Solutions* 4: 5356–73. doi.org/10.1109/ACCESS.2016.2603219.

Kreps, D. 2018. *Notes on the Theory of Choice*. London: Routledge. doi.org/10.4324/9780429498619.

Liu, S.; Hudson Smith, M.; Tuck, S.; Pan, J.; Alkuraiji, A.; and Jayawickrama, U. 2015. Where Can Knowledge-Based Decision Support Systems Go in Contemporary Business Management—A New Architecture for the Future. Journal of Economics, *Business and Management* 3(5): 498–504.

Livingston, M. A.; Russell, S.; Decker, J. W.; Leadbetter, E.; and Gilliam, A. 2015. CEDARS: Combined Exploratory Data Analysis Recommender System. In *5th Symposium on Large Data Analysis and Visualization (LDAV) 2015*, 139–40. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Murphy, K. 2012. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: The MIT Press.

Murty, K. G., and Kabadi, S. N. 1987. Some NP-Complete Problems in Quadratic and Nonlinear Programming. *Mathematical Programming* 39(2): 117–29. doi.org/10.1007/BF02592948.

Nagaraj K, Pasupathy R. 2014. Stochastically constrained simulation optimization on integer-ordered spaces: The cgR-SPLINE algorithm. Technical Report. Purdue University, West Lafayette, IN.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical Black-Box Attacks Against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–19. New York: Association for Computing Machinery.

Quiñonero-Candela, J., and Rasmussen, C. E. 2005. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research* 6(Dec): 1939–59.

Russell, S., and Moskowitz, I. S. 2016. Human Information Interaction, Artificial Intelligence, and Errors. In *AI and the Mitigation of Human Error: Anomalies, Team Metrics and Thermodynamics: Papers from the 2016 AAAI Spring Symposium Series*. Technical Report SS-16-01. Palo Alto, CA: Association for the Advancement of Artificial Intelligence Press.

Russell, S.; Moskowitz, I. S.; and Raglin, A. 2017. Human Information Interaction, Artificial Intelligence, and Errors. In *Autonomy and Artificial Intelligence: A Threat or Savior?* 71–101. Berlin: Springer. doi.org/10.1007/978-3-319-59719-5_4.

Schilit, B.; Adams, N.; and Want, R. 1994. Context-Aware Computing Applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications 1994*, 85–90. Piscataway, NJ: Institute of Electrical and Electronics Engineers. doi.org/10.1109/WMCSA.1994.16.

Scott, M.; Ingalls, B.; and Kaern, M. 2006. Estimations of Intrinsic and Extrinsic Noise in Models of Nonlinear Genetic Networks. *Chaos (Woodbury, N.Y.)* 16(2): 026107. doi.org/10.1063/1.2211787.

Shalev-Shwartz, S.; Shamir, O.; Srebro, N.; and Sridharan, K. 2010. Learnability, Stability and Uniform Convergence. *Journal of Machine Learning Research* 11(Oct): 2635–70.

Srebro, N., and Tewari, A. 2010. Stochastic Optimization: ICML 2010 Tutorial. Paper presented at the International Conference on Machine Learning 2010. www.ttic.edu/icml2010stochopt/.

Steel, D. 2011. Extrapolation, Uncertainty Factors, and the Precautionary Principle. *Studies in History and Philosophy of Science Part C Studies in History and Philosophy of Biological and Biomedical Sciences* 42(3): 356–64. doi.org/10.1016/j.shpsc.2011.01.002.

Vapnik, V. N. 1995. *The Nature of Statistical Learning Theory*. Berlin: Springer. doi.org/10.1007/978-1-4757-2440-0.

Zhang, X.; Liu, X.; Samani, H.; and Jalaian, B. 2015. Cooperative Spectrum Sensing in Cognitive Wireless Sensor Networks. *International Journal of Distributed Sensor Networks* 11(8): 170695. doi.org/10.1155/2015/170695.

Zhu, Q., and Azar, A. T., editors. 2015. *Complex System Modelling and Control Through Intelligent Soft Computations*. Berlin: Springer. doi.org/10.1007/978-3-319-12883-2.

Brian Jalaian received his MS degree in electrical engineering and the MS degree in industrial system engineering (operation research) in 2013 and 2014, respectively, and his PhD degree in electrical engineering from Virginia Polytechnic Institute and State University, Blacksburg, Virginia, in 2016. He is currently a research scientist at the US Army Research Laboratory and research lead for the several US Army Research Laboratory research initiatives. He is also an adjunct assistant professor in the Electrical and Computer Engineering Department at Virginia Tech. His research interests include optimization, UQ in ML and AI, adversarial ML, and communication network optimization.

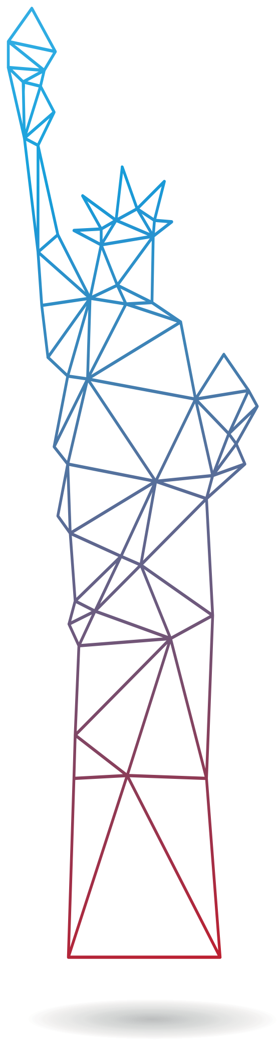
AI in Practice

Colocated with AAAI-20

Hilton New York Midtown

February 12, 2020

aaai.org/Conferences/AAAI-20/ai-in-practice



AI IN PRACTICE will be held at the AAAI-20 conference in New York on February 12. The focus of this year's program will be emerging applications of AI in healthcare. The aim is to offer a venue for exchanging ideas among participants from different disciplines, from general computer science, to AI and ethics, to medicine and public health.

The program will begin at 10:00 AM with opening remarks by AI in Practice cochairs Evgeniy Gabrilovich (Google Health) and Ashish Jha (Harvard School of Public Health), to be immediately followed by the first featured keynote.

Keynotes

The first keynote will be by given by Vivian Lee (Verily Life Sciences). An eminent researcher and healthcare executive, Dr. Vivian S. Lee has served as president, health platforms at Verily Life Sciences, an Alphabet/Google company dedicated to improving health through engaging technology and insightful data analytics. She was elected to the National Academy of Medicine in 2015, and serves as a Director of the Commonwealth Fund. Her research focuses on quantitative functional MRI for the improved understanding of physiology and disease.

The second featured keynote, to be held at 4:45 PM, will be given by Aneesh Chopra (CareJourney). Chopra is the president of CareJourney, an open data intelligence service launched by Hunch Analytics. From 2009–2012, he served as the first U.S. Chief Technology Officer and prior to that, as Virginia's fourth secretary of technology.

Invited Speakers

Invited speakers at AI in Practice will include Isaac Kohane, Harvard Medical School (10:50 AM); Honor Hsin, Kaiser Foundation (2:00 PM); Leo Celi, Massachusetts Institute of Technology (2:25 PM); John Brownstein, Boston Children's Hospital (2:50 PM); and Kira Radinsky, Diagnostic Robotics (3:45 PM).

Panel

The program will also feature an invited panel, to be held at 11:20 AM with panelists Freddy Abnousi (Google), Greg Corrado (Facebook), Jim Weintstein (Microsoft), and moderator Lisa Simpson.

AI in Practice will conclude the day with discussions and networking, from 5:45–6:45 PM.

Michael Lee received his BS degree in computer science from the University of Maryland at College Park in 2000 and his MS degree in information systems from Johns Hopkins University in 2009. He is currently a computer scientist at the US Army Research Laboratory. His research interests include data science and UQ in ML.

Stephen Russell received his BS degree in computer science and his MS and PhD degrees in information systems from the University of Maryland, College Park, Maryland. He is currently the Battlefield Information Processing Branch Chief at the US Army Research Laboratory, Adelphi, Maryland. His published research articles appear in *Expert Systems with Applications*, the *Decision Support Systems Journal*, the *Encyclopedia of Decision Making and Decision Support Technologies*, and *Frontiers in Bioscience*. His primary research interests include the area of decision support systems, ML, systems architectures, and intelligent systems.