On Statistical Thinking in Deep Learning
A Blog Post

Yee Whye Teh
Department of Statistics, University of Oxford
DeepMind

IMS Medallion Lecture
Joint Statistical Meeting
Denver, USA

July 2019

Abstract

In recent years, machine learning, and in particular deep learning, has undergone tremendous growth, much of this driven by advances that are computational in nature, including software and hardware infrastructures supporting increasingly complex models and enabling use of increasingly intense compute power. As a result the field is becoming more computational in its nature. In this talk I would like to highlight the continuing importance of statistical thinking in deep learning, by drawing examples from my research blending probabilistic modelling, Bayesian nonparametrics and deep learning. In particular, I will talk about neural processes, which uses neural networks to parameterise and learn flexible stochastic processes and use them for meta-learning (also known as learning to learning), and about the use of probabilistic symmetries in answering recent questions about neural network architecture choices satisfying certain invariance properties.

1 Introduction

Contrary to what university administrations might think, academic research knows no boundaries, and the boundaries between disciplines, if any, are often fuzzy, ill-defined, porous, and ever-changing. This is most keenly felt among those of us whose work falls into both machine learning and statistics.

Historically, machine learning has its roots in pattern recognition and connectionist systems whose intelligent behaviours are learnt from data or interactions with environments, as opposed to being programmed. In the 90s, led by the visionary works by Mike Jordan, Zoubin Ghahramani, David MacKay,

Radford Neal, among others, the community started realising the widespread connection of ideas with statistics. This led to a period when there was essentially no boundary between the two communities, and statistical approaches flourished and became the dominant framework for both theoretical foundations and methodological developments in machine learning.

In the last decade, with the growing popularity of deep learning, this coming together of machine learning and statistics has started to unravel a bit. The research frontier moved from statistical learning to artificial intelligence and from areas like graphical models, Bayesian nonparametrics and Markov chain Monte Carlo, to autoencoders, neural architecture search, memory, attention, deep reinforcement learning, meta learning, as well as software systems like TensorFlow and PyTorch. Newcomers to the field are much more excited about ConvNets than GLMNets, SGD than MCMC, GANs than PCA, variational autoencoders than latent Dirichlet allocation, transformers than Gaussian processes.

In this dawn of the deep learning era, how much of a role does statistical thinking still have to play in advancing the state-of-the-art in machine learning? Should our graduate students still study The Elements of Statistical Learning (Friedman et al., 2001) as opposed to Deep Learning (Goodfellow et al., 2016)? It is my belief, and that of many others, that statistical thinking continues to play important roles in the current frontiers of machine learning. The deep theoretical roots of statistics and probability have continued to fertilize our theoretical understanding of deep learning phenomena; in unsupervised learning, generative modelling have continued to be the dominant paradigm; and the deep concern for uncertainty and robustness prevalent in statistics is now being increasingly felt as machine learning techniques make their way into the real world.

I will illustrate how statistical thinking played an essential role with two inter-related examples from my own research, one more methodological in nature, on using neural networks to learn stochastic processes, and one more theoretical, on characterising neural architectures satisfying certain symmetry properties.

2 Meta Learning and Neural Processes

The first concerns the idea of meta learning or learning to learn (Thrun and Pratt, 1998; Ravi and Larochelle, 2016; Santoro et al., 2016; Andrychowicz et al., 2016; Finn et al., 2017; Snell et al., 2017). While much of machine learning excels at learning from large datasets, we would like systems that can learn more efficiently, with much less data. For example, in few-shot image classification, a system might be presented with just a few example images of each class, and we would like it to generalise well to classifying other images into the given classes. The idea here behind meta learning is that if our system has seen lots of examples of such few-shot image classification tasks (each with its own small dataset), we might conceivably expect there to be sufficient information spread across tasks for a system to learn about the domain well enough so that it can

perform sensible generalisation from few examples.

We can formalise meta learning as follows. Let $j \in \{1, \ldots, N\}$ index one of N tasks. We assume that these tasks are independent and identically distributed according to some task distribution. As usual, we assume that the data associated with task j can be split into a training set $\mathcal{D}_j^{\mathsf{train}} = \{(x_{ji}^{\mathsf{train}}, y_{ji}^{\mathsf{train}})\}_{i=1}^n$ of n input $x_{ji} \in \mathcal{X}$ and output $y_{ji} \in \mathcal{Y}$ pairs, and a corresponding test set $\mathcal{D}_j^{\mathsf{test}} = \{(x_{ji}^{\mathsf{test}}, y_{ji}^{\mathsf{test}})\}_{i=1}^m$ of size m. We assume the training and test sets are of the same sizes n and m for simplicity. We assume N is large while n and m are small; this distinguishes the meta learning scenario from say multitask or transfer learning, which typically assumes small N and large n and m.

Most current approaches to meta learning are based on the idea that the learning algorithm has at its core an optimiser, and meta learning is about optimising the optimiser (Ravi and Larochelle, 2016; Santoro et al., 2016; Finn et al., 2017; Snell et al., 2017; Andrychowicz et al., 2016). A learning algorithm can be thought of as a procedure that takes $\mathcal{D}_j^{\mathsf{train}}$ and outputs an optimised parameter vector θ_j. We can write this as a function, say $\theta_j = \mathcal{A}_\eta(\mathcal{D}_j^{\mathsf{train}})$, where η are hyperparameters of the learning algorithm. The learnt θ_j parameterises a function (neural network) $f_{\theta_j}(x)$ which maps an input x to a prediction about the corresponding y. In the classification setting, $f_{\theta_j}(x)$ might be a vector of predicted probabilities of x being in each of the classes. Given θ_j, we can evaluate its performance on the test set using a loss function,

$$\mathcal{L}(\theta_j, \mathcal{D}_j^{\mathsf{test}}) = \sum_{i=1}^m \mathcal{L}(f_{\theta_j}(x_{ji}^{\mathsf{test}}), y_{ji}^{\mathsf{test}})$$

The idea of meta learning is now to optimise the hyperparameters η, so that the learning algorithm generalises well from the training set to the test set of each task. If $\mathcal{L}$, f and $\mathcal{A}$ are all differentiable, we get:

$$\frac{d}{d\eta} \sum_{j=1}^N \mathcal{L}(\mathcal{A}_\eta(\mathcal{D}_j^{\mathsf{train}}), \mathcal{D}_j^{\mathsf{test}}) = \sum_{j=1}^N \sum_{i=1}^m \frac{\partial \mathcal{L}}{\partial f}(f_{\theta_j}(x_{ji}^{\mathsf{test}}), y_{ji}^{\mathsf{test}}) \frac{\partial f_{\theta_j}}{\partial \theta}(x_{ji}^{\mathsf{test}}) \frac{\partial \mathcal{A}_\eta}{\partial \eta}(\mathcal{D}_j^{\mathsf{train}})$$

Different approaches to meta learning then involves different architectural choices for $\mathcal{A}$ and f, as well as different choices for the meta optimiser for η.

An interesting alternative to meta learning considers it from the statistical perspectives of hierarchical Bayes and stochastic processes (Garnelo et al., 2018a,b; Kim et al., 2019; Galashov et al., 2019). The idea is that in order to learn effectively from a small amount of data, prior knowledge of the domain is necessary. From a Bayesian perspective, this prior knowledge can be expressed in terms of a prior distribution G. Since we are in a supervised learning setting, for each task j, let $f_j(x)$ be the (unknown) function of interest which gives the prediction for y given x. The hierarchical Bayesian model can then be given by:

$$\begin{aligned} f_j &\sim_{iid} G \\ y_{ji} | f_j, x_{ji} &\sim_{ind} F(f_j(x_{ji})) \end{aligned} \tag{1}$$

where F is an output model and the prior G is a distribution over functions, i.e. a stochastic process.

A Bayesian nonparametric (Hjort et al., 2010) approach typically would assume a particular form for G, for example a Gaussian process (Williams and Rasmussen, 2006) or a Dirichlet process mixture thereof (MacEachern, 1999). Given training data set $\mathcal{D}_j^{\mathsf{train}}$ the posterior over functions f_j is computed and used to make predictions on the test set $\mathcal{D}_j^{\mathsf{test}}$. The problem with such an approach is that the posterior is typically too expensive or intractable to compute. The first implication is that approximations are necessary. The second implication is that this sets off a tension between the complexity of the prior used, and the efficiency with which the posterior can be approximated. In practice this leads to model simplifications made for reasons of computational tractability.

Instead of the typical Bayesian approach which takes as its central operation the computation of the posterior distribution (over functions), the meta learning approach takes as its central operation learning predictive distributions of the test sets given the training sets. More precisely, the predictive distributions $\mathbb{P}(y|x, \mathcal{D}^{\mathsf{train}})$ are those over the test output y given a test input x as well as some training set $\mathcal{D}^{\mathsf{train}} = \{(x_i^{\mathsf{train}}, y_i^{\mathsf{train}})\}_{i=1}^n$, and with the underlying functions f_j marginalised out. These predictive distributions can be learnt using the meta learning setup where each task corresponds to an iid draw from G. In practice, we parameterise a function $f(\cdot|x, \mathcal{D}^{\mathsf{train}})$ using a neural network to learn the predictive distributions.

Interestingly, if both n varies sufficiently over the positive integers and the training sets vary sufficiently over the input and output spaces, for our purposes, the set of predictive distributions completely characterise the stochastic process G, and are arguably all we need. Hence an interpretation of meta learning is as using a neural network to learn to reproduce the stochastic process G (in terms of its predictive distributions). Note that we need not compute the posterior distribution in this setup, we just need to be able to simulate data sets under the hierarchical model (1). We need not have an explicit mathematical construction of the prior G either, so long as we have a family of data sets from which the prior can be learnt (implicitly through its predictive distributions).

We call our approach neural processes since it uses neural networks to parameterise and learn stochastic processes. We have used neural processes for few shot image classification (Garnelo et al., 2018a), image modelling (Garnelo et al., 2018b; Kim et al., 2019), as well as sequential decision making and collaborative filtering (Galashov et al., 2019). Viewing meta learning from a statistical perspective has allowed us to understand the underlying learning phenomena better. This has in turn allowed us to make links with other ideas like Bayesian nonparametrics and Gaussian processes, and motivated new approaches which better handle uncertainty (Garnelo et al., 2018b) and learns more accurate predictions (Kim et al., 2019), as well as new applications of meta learning, for example in Bayesian optimisation and sequential decision making (Galashov et al., 2019).

3 Probabilistic Symmetries and Neural Networks

One interesting question in the previous section is how the function $f(\cdot|x, \mathcal{D}^{\mathsf{train}})$ with $\mathcal{D}^{\mathsf{train}} = \{(x_i^{\mathsf{train}}, y_i^{\mathsf{train}})\}_{i=1}^n$ should be parameterised. In particular, the predictive distribution is invariant with respect to permuting the indices of the training set $\mathcal{D}^{\mathsf{train}}$, so the function should too. In our work we have enforced this permutation invariance explicitly by choosing a specific neural architecture. Specifically, let $g(x, y)$ be a neural network mapping from $\mathcal{X} \times \mathcal{Y}$ to some representation space, say $\mathbb{R}^d$, and $h(x, z)$ be a neural network mapping from $\mathcal{X} \times \mathbb{R}^d$ to a distribution over the output space $\mathcal{Y}$. We call g the encoder and h the decoder. The function f is then given as:

$$f(\cdot|x, \{(x_i^{\mathsf{train}}, y_i^{\mathsf{train}})\}_{i=1}^n) = h\left(x, \sum_{i=1}^n g(x_i^{\mathsf{train}}, y_i^{\mathsf{train}})\right) \tag{2}$$

The summation $\sum_{i=1}^n g(x_i^{\mathsf{train}}, y_i^{\mathsf{train}})$ can be thought of as an implicit representation or encoding of the posterior distribution over the stochastic process given the data set $\mathcal{D}^{\mathsf{train}}$. For this reason g is called an encoder, while h is called a decoder since it decodes the representation of the posterior into a prediction distribution given x.

By construction the function is invariant to permutations of the indexing of the data points in the training set, since addition is commutative. However there are other commutative operators, for example element-wise product, max, or min. This raises the following questions: Which operator is best? Are there other neural architectures or function classes that have this permutation invariance property? Can we characterise all functions with a permutation invariance property? Can we generalise these questions from functions to conditional distributions, and from permutation invariance to other forms of symmetries? In Bloem-Reddy and Teh (2019) we developed a framework to answer these questions using tools from probabilistic symmetries and the study of statistical sufficiency.

For simplicity, consider a function $y = f(x_1, \ldots, x_n)$, which is invariant to permutations of the input sequence $x = (x_1, \ldots, x_n)$. In other words,

$$f(x) := f(x_1, \ldots, x_n) = f(x_{\pi(1)}, \ldots, x_{\pi(n)}) =: f(\pi \cdot x)$$

for all $\pi \in \mathbb{S}_n$ a permutation of the integers $1, \ldots, n$, and where $\pi \cdot x$ is the action of the symmetry group on the input sequence. We can generalise this functional notion of symmetry to a probabilistic notion of symmetry, where a conditional distribution $\mathbb{P}(Y|X_1, \ldots, X_n)$ of a random variable Y is invariant to permutations of the conditioning variables $X = (X_1, \ldots, X_n)$:

$$\mathbb{P}(Y|X) := \mathbb{P}(Y|X_1, \ldots, X_n) = \mathbb{P}(Y|X_{\pi(1)}, \ldots, X_{\pi(n)}) =: \mathbb{P}(Y|\pi \cdot X)$$

In this case, it is reasonable also to consider that the marginal distribution over X should be permutation invariant as well, that is,

$$\mathbb{P}(X) = \mathbb{P}(\pi \cdot X)$$

In the following we will always assume that both the conditional and the marginal distributions are permutation invariant.

The functional and probabilistic notions of symmetry are intimately linked. In one direction, a function $Y = f(X)$ can be thought of as a degenerate conditional distribution of Y with a delta mass at $f(X)$. For the converse direction, under weak conditions, given a conditional distribution $\mathbb{P}(Y|X)$, an idea from probability called noise outsourcing shows that we can write $Y = f(X, \epsilon)$ almost surely, for a deterministic function f and a random variable $\epsilon \sim U[0, 1]$ independent of X. The variable ϵ is called the outsourced noise, as it captures all the randomness in Y conditional on X, leaving the dependence of Y given X captured by the function f. Given this, we see that a conditional distribution $\mathbb{P}(Y|X)$ is permutation invariant if and only if it has a noise outsourced representation, $Y = f(X, \epsilon)$ almost surely, using a function f that is permutation invariant with respect to X.

Now consider a class of joint distributions $\mathbb{P} \in \mathcal{P}$ over X and Y. A statistic $M(X)$ is adequate for Y if it is sufficient for the marginal distribution $\mathbb{P}(X)$ and satisfies $\mathbb{P}(Y|X) = \mathbb{P}(Y|M(X))$ for each $\mathbb{P} \in \mathcal{P}$. It turns out that the noise outsourced function can be written as $Y = h(M(X), \epsilon)$ for another deterministic function h. In other words we can write Y as a function only of the adequate statistics and of the outsourced noise.

The final piece of the puzzle is then, given a class of permutation invariant joint distributions, what is an adequate statistic of X for Y? This turns out to be the empirical measure

$$M(X_1, \ldots, X_n) = \sum_{i=1}^{n} \delta_{X_i}$$

where δ_x is a point mass at x. So that for any permutation invariant conditional distribution $Y|X_1, \ldots, X_n$, we can write

$$Y = h\left(\sum_{i=1}^{n} \delta_{X_i}, \epsilon\right)$$

for some function h and $\epsilon \sim U[0, 1]$ independent of $X_1, \ldots, X_n$. As a corollary, in the deterministic case, any permutation invariant function $Y = f(X)$ can be written as:

$$f(x_1, \ldots, x_n) = h\left(\sum_{i=1}^{n} \delta_{x_i}\right)$$

Note that δ_x can be thought of as a function mapping x into the space of probability measures on $\mathcal{X}$, so plays an analogous role to the encoder g in the beginning of this section. One way to think about this characterisation of permutation invariant functions is that the ordering among the input sequence is unimportant, and, intuitively, the empirical measure exactly captures all information about the input sequence except the ordering. Thus any permutation

invariant function of the input sequence has to be a function of the empirical measure. The implication for neural architecture design is that if we would like a universally rich class of neural networks to model a permutation invariant function, it is sufficient to use functions of the form:

$$f(x_1, \ldots, x_n) = h\left(\sum_{i=1}^{n} g(x_i)\right)$$

In Bloem-Reddy and Teh (2019) we have generalised this result in a few ways. Firstly, we can generalise from invariance with respect to permutations to invariance under the action of some compact group. The results are structurally the same, except that the empirical measure is replaced by an appropriate adequate statistic called a maximal invariant. We have also derived analogous results for a different notion of symmetry called equivariance, where transformations of the input lead to output that is transformed in the same way. Finally, we have also applied our results to characterise the neural and functional architectures associated with invariant and equivariant functions and conditional distributions operating on graph and array structured data, a currently very popular area of deep learning.

4 Final Remarks

Statistical thinking continues to play an important role in machine learning, even as there are exciting and disruptive technological changes brought to the fore by deep learning and the associated software and hardware advances. I have illustrated this with two examples from my own research. Other areas where statistical thinking plays a foundational role include learning theory, deep generative models, causal inference, Bayesian optimisation and sequential decision making, and bias and variance trade-offs in reinforcement learning. As machine learning's impact on society grows in scale and importance, questions of fairness and robustness are becoming more important, and statistical thinking has important roles to play here as well. Finally, as machine learning technologies are applied to solve ever diverse problems in science and society, the applied statistics community also has a lot to contribute, in terms of its deep experiences, wisdom and care applying statistical tools to real world problems.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems (NeurIPS)*, pages 3981–3989.

Bloem-Reddy, B. and Teh, Y. W. (2019). Probabilistic symmetry and invariant neural networks. arXiv:1901.06082.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*. Springer Series in Statistics, New York.

Galashov, A., Schwarz, J., Kim, H., Garnelo, M., Saxton, D., Kohli, P., Eslami, S., and Teh, Y. W. (2019). Meta-learning surrogate models for sequential decision making. In *ICLR Workshop on Structure & Priors in Reinforcement Learning*. arXiv:1903.11907.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. (2018a). Conditional neural processes. In *International Conference on Machine Learning (ICML)*.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. (2018b). Neural processes. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*. arXiv:1807.01622.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Hjort, N. L., Holmes, C., Müller, P., and Walker, S. G. (2010). *Bayesian Nonparametrics*, volume 28. Cambridge University Press.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2019). Attentive neural processes. In *International Conference on Learning Representations (ICLR)*. arXiv:1901.05761.

MacEachern, S. N. (1999). Dependent nonparametric processes. In *ASA Proceedings of the Section on Bayesian Statistical Science*, volume 1, pages 50–55. Alexandria, Virginia. Virginia: American Statistical Association; 1999.

Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning (ICML*, pages 1842–1850.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4077–4087.

Thrun, S. and Pratt, L. (1998). *Learning to learn*. Springer Science & Business Media.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.