Title: Neural network Gaussian process

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

PageID: 63513679

Categories: Category:Artificial neural networks, Category:Bayesian networks, Category:Bayesian statistics, Category:Deep learning, Category:Kernel methods for machine learning

-----

A Neural Network Gaussian Process (NNGP) is a Gaussian process (GP) obtained as the limit of a certain type of sequence of neural networks . Specifically, a wide variety of network architectures converges to a GP in the infinitely wide limit , in the sense of distribution . [1] [2] [3] [4] [5] [6] [7] [8] The concept constitutes an intensional definition , i.e., a NNGP is just a GP, but distinguished by how it is obtained.

Motivation

Bayesian networks are a modeling tool for assigning probabilities to events, and thereby characterizing the uncertainty in a model's predictions. Deep learning and artificial neural networks are approaches used in machine learning to build computational models which learn from training examples. Bayesian neural networks merge these fields. They are a type of neural network whose parameters and predictions are both probabilistic. [9] [10] While standard neural networks often assign high confidence even to incorrect predictions, [11] Bayesian neural networks can more accurately evaluate how likely their predictions are to be correct.

Computation in artificial neural networks is usually organized into sequential layers of artificial neurons . The number of neurons in a layer is called the layer width. When we consider a sequence of Bayesian neural networks with increasingly wide layers (see figure), they converge in distribution to a NNGP. This large width limit is of practical interest, since the networks often improve as layers get wider. [12] [4] [13] And the process may give a closed form way to evaluate networks.

NNGPs also appears in several other contexts: It describes the distribution over predictions made by wide non-Bayesian artificial neural networks after random initialization of their parameters, but before training; it appears as a term in neural tangent kernel prediction equations; it is used in deep information propagation to characterize whether hyperparameters and architectures will be trainable. [14] It is related to other large width limits of neural networks.

Scope

The first correspondence result had been established in the 1995 PhD thesis of Radford M. Neal , [15] then supervised by Geoffrey Hinton at University of Toronto . Neal cites David J. C. MacKay as inspiration, who worked in Bayesian learning .

Today the correspondence is proven for: Single hidden layer Bayesian neural networks; [15] deep [2] [3] fully connected networks as the number of units per layer is taken to infinity; convolutional neural networks as the number of channels is taken to infinity; [4] [5] [6] transformer networks as the number of attention heads is taken to infinity; [16] recurrent networks as the number of units is taken to infinity. [8] In fact, this NNGP correspondence holds for almost any architecture: Generally, if an architecture can be expressed solely via matrix multiplication and coordinatewise nonlinearities (i.e., a tensor program), then it has an infinite-width GP. [8] This in particular includes all feedforward or recurrent neural networks composed of multilayer perceptron, recurrent neural networks (e.g., LSTMs , GRUs), (nD or graph) convolution , pooling, skip connection, attention, batch normalization , and/or layer normalization.

Illustration

Every setting of a neural network's parameters θ {\displaystyle \theta } corresponds to a specific function computed by the neural network. A prior distribution $p(\theta)$ {\displaystyle p(\theta)} over neural network parameters therefore corresponds to a prior distribution over functions computed by the network. As neural networks are made infinitely wide, this distribution over functions converges to a Gaussian process for many architectures.

The notation used in this section is the same as the notation used below to derive the correspondence between NNGPs and fully connected networks, and more details can be found there.

The figure to the right plots the one-dimensional outputs z L (· ; θ) {\displaystyle z^{L}(\cdot ;\theta)} of a neural network for two inputs x {\displaystyle x} and x ∗ {\displaystyle x^{*}} against each other. The black dots show the function computed by the neural network on these inputs for random draws of the parameters from p (θ) {\displaystyle p(\theta)} . The red lines are iso-probability contours for the joint distribution over network outputs z L (x ; θ) {\displaystyle z^{L}(x;\theta)} and z L (x ∗ ; θ) {\displaystyle z^{L}(x^{*};\theta)} induced by p (θ) {\displaystyle p(\theta)} . This is the distribution in function space corresponding to the distribution p (θ) {\displaystyle p(\theta)} in parameter space, and the black dots are samples from this distribution. For infinitely wide neural networks, since the distribution over functions computed by the neural network is a Gaussian process, the joint distribution over network outputs is a multivariate Gaussian for any finite set of network inputs.

Discussion

Infinitely wide fully connected network

This section expands on the correspondence between infinitely wide neural networks and Gaussian processes for the specific case of a fully connected architecture. It provides a proof sketch outlining why the correspondence holds, and introduces the specific functional form of the NNGP for fully connected networks. The proof sketch closely follows the approach by Novak and coauthors. [4]

Network architecture specification

Consider a fully connected artificial neural network with inputs x {\displaystyle x} , parameters θ {\displaystyle \theta } consisting of weights W l {\displaystyle W^{l}} and biases b l {\displaystyle b^{l}} for each layer l {\displaystyle l} in the network, pre-activations (pre-nonlinearity) z l {\displaystyle z^{l}} , activations (post-nonlinearity) y l {\displaystyle y^{l}} , pointwise nonlinearity φ (·) {\displaystyle \phi (\cdot)} , and layer widths n l {\displaystyle n^{l}} . For simplicity, the width n L + 1 {\displaystyle n^{L+1}} of the readout vector z L {\displaystyle z^{L}} is taken to be 1. The parameters of this network have a prior distribution p (θ) {\displaystyle p(\theta)} , which consists of an isotropic Gaussian for each weight and bias, with the variance of the weights scaled inversely with layer width. This network is illustrated in the figure to the right, and described by the following set of equations:

z l | y l {\displaystyle z^{l}|y^{l}} is a Gaussian process

We first observe that the pre-activations z l {\displaystyle z^{l}} are described by a Gaussian process conditioned on the preceding activations y l {\displaystyle y^{l}} . This result holds even at finite width.

Each pre-activation z i l {\displaystyle z_{i}^{l}} is a weighted sum of Gaussian random variables, corresponding to the weights W i j l {\displaystyle W_{ij}^{l}} and biases b i l {\displaystyle b_{i}^{l}} , where the coefficients for each of those Gaussian variables are the preceding activations y j l {\displaystyle y_{j}^{l}} .

Because they are a weighted sum of zero-mean Gaussians, the z i l {\displaystyle z_{i}^{l}} are themselves zero-mean Gaussians (conditioned on the coefficients y j l {\displaystyle y_{j}^{l}}).

Since the z l {\displaystyle z^{l}} are jointly Gaussian for any set of y l {\displaystyle y^{l}} , they are described by a Gaussian process conditioned on the preceding activations y l {\displaystyle y^{l}} .

The covariance or kernel of this Gaussian process depends on the weight and bias variances σ w 2 {\displaystyle \sigma _{w}^{2}} and σ b 2 {\displaystyle \sigma _{b}^{2}} , as well as the second moment matrix K l {\displaystyle K^{l}} of the preceding activations y l {\displaystyle y^{l}} ,

The effect of the weight scale σ w 2 {\displaystyle \sigma _{w}^{2}} is to rescale the contribution to the covariance matrix from K l {\displaystyle K^{l}} , while the bias is shared for all inputs, and so σ b

2 {\displaystyle \sigma _{b}^{2}} makes the z i l {\displaystyle z_{i}^{l}} for different datapoints more similar and makes the covariance matrix more like a constant matrix.

z l | K l {\displaystyle z^{l}|K^{l}} is a Gaussian process

The pre-activations z l {\displaystyle z^{l}} only depend on y l {\displaystyle y^{l}} through its second moment matrix K l {\displaystyle K^{l}} . Because of this, we can say that z l {\displaystyle z^{l}} is a Gaussian process conditioned on K l {\displaystyle K^{l}} , rather than conditioned on y l {\displaystyle y^{l}} ,

As layer width n l → ∞ {\displaystyle n^{l}\rightarrow \infty } , K l ■ K l − 1 {\displaystyle K^{l}\mid K^{l-1}} becomes deterministic

As previously defined, K l {\displaystyle K^{l}} is the second moment matrix of y l {\displaystyle y^{l}} . Since y l {\displaystyle y^{l}} is the activation vector after applying the nonlinearity φ {\displaystyle \phi } , it can be replaced by φ (z l − 1) {\displaystyle \phi \left(z^{l-1}\right)} , resulting in a modified equation expressing K l {\displaystyle K^{l}} for l > 0 {\displaystyle l>0} in terms of z l − 1 {\displaystyle z^{l-1}} ,

We have already determined that z l − 1 | K l − 1 {\displaystyle z^{l-1}|K^{l-1}} is a Gaussian process. This means that the sum defining K l {\displaystyle K^{l}} is an average over n l {\displaystyle n^{l}} samples from a Gaussian process which is a function of K l − 1 {\displaystyle K^{l-1}} ,

{ z i l − 1 (x) , z i l − 1 (x ′) } ∼ G P (0 , σ w 2 K l − 1 + σ b 2) . {\displaystyle {\begin{aligned}\left\{z_{i}^{l-1}(x),z_{i}^{l-1}(x')\right\}&\sim {\mathcal {GP}}\left(0,\sigma _{w}^{2}K^{l-1}+\sigma _{b}^{2}\right).\end{aligned}}}

As the layer width n l {\displaystyle n^{l}} goes to infinity, this average over n l {\displaystyle n^{l}} samples from the Gaussian process can be replaced with an integral over the Gaussian process:

So, in the infinite width limit the second moment matrix K l {\displaystyle K^{l}} for each pair of inputs x {\displaystyle x} and x ′ {\displaystyle x'} can be expressed as an integral over a 2d Gaussian, of the product of φ (z) {\displaystyle \phi (z)} and φ (z ′) {\displaystyle \phi (z')} .

There are a number of situations where this has been solved analytically, such as when φ (⋅) {\displaystyle \phi (\cdot)} is a ReLU , [17] ELU, GELU, [18] or error function [1] nonlinearity.

Even when it can't be solved analytically, since it is a 2d integral it can generally be efficiently computed numerically. [2] This integral is deterministic, so K l | K l − 1 {\displaystyle K^{l}|K^{l-1}} is deterministic.

For shorthand, we define a functional F {\displaystyle F} , which corresponds to computing this 2d integral for all pairs of inputs, and which maps K l − 1 {\displaystyle K^{l-1}} into K l {\displaystyle K^{l}} ,

z L ■ x {\displaystyle z^{L}\mid x} is an NNGP

By recursively applying the observation that K l ■ K l − 1 {\displaystyle K^{l}\mid K^{l-1}} is deterministic as n l → ∞ {\displaystyle n^{l}\rightarrow \infty } , K L {\displaystyle K^{L}} can be written as a deterministic function of K 0 {\displaystyle K^{0}} ,

where F L {\displaystyle F^{L}} indicates applying the functional F {\displaystyle F} sequentially L {\displaystyle L} times.

By combining this expression with the further observations that the input layer second moment matrix K 0 (x , x ′) = 1 n 0 ∑ i x i x i ′ {\displaystyle K^{0}(x,x')={\tfrac {1}{n^{0}}}\sum _{i}x_{i}x'_{i}} is a deterministic function of the input x {\displaystyle x} , and that z L | K L {\displaystyle z^{L}|K^{L}} is a Gaussian process, the output of the neural network can be expressed as a Gaussian process in terms of its input,

Software libraries

Neural Tangents is a free and open-source Python library used for computing and doing inference with the NNGP and neural tangent kernel corresponding to various common ANN architectures. [

19]

References