Midterm report

EE5111 Course Project

Sourav Sahoo, Subhankar Chakraborty, Sumanth Hegde

Objectives

- Obtaining principled uncertainty estimates from deep networks.
- Using Dropout to obtain uncertainty estimates
- How can uncertainty estimates be used for adversarial robustness?

Why should I care about uncertainty?

- Uncertainty information is critical in areas such as the life sciences and autonomous driving, where decisions taken are high-stakes
- Model uncertainty is also important in understanding the shortcomings of the model.
- Current deep learning techniques rely on point estimates for the weights. Softmax scores are not enough to obtain true uncertainty!

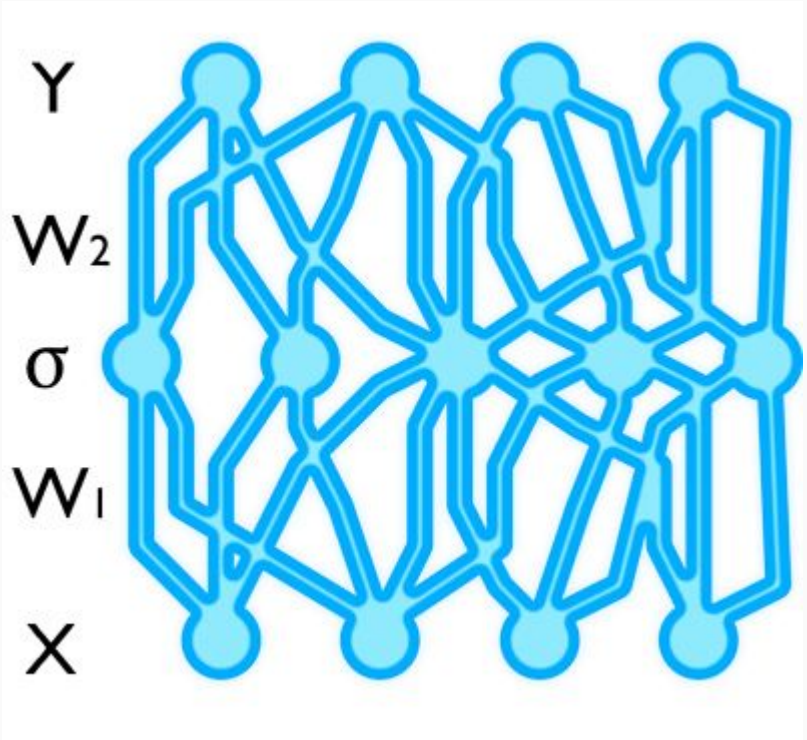
How do you obtain uncertainty information?

- Bayesian methods are traditionally used for this!
- However, these models are typically harder to optimise.
- Gaussian processes - model distributions over possible functions that fit the data
- Neural Networks can be related to a Gaussian process - in the limit of infinite width of the hidden layers.
- Dropout allows us to obtain uncertainty estimates from standard NNs - without changing a thing!

Dropout : A Quick Review

- Consider a neural network with a single hidden layer
- Let W_1 and W_2 be the weights of the network as shown.
- Assume that Input dimensions = Q; Output = D and hidden layer = K
- The output for an input **x** would be

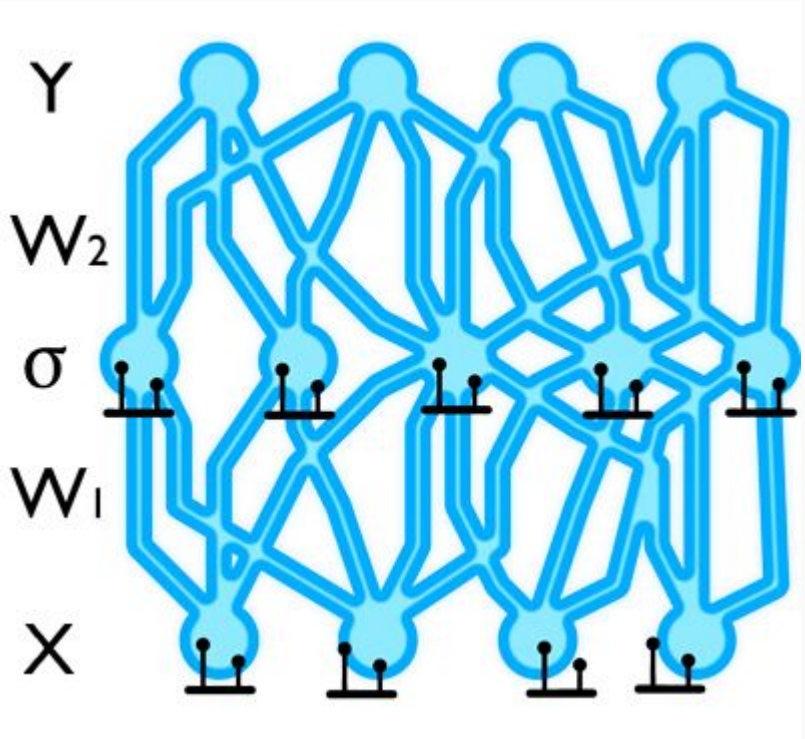
$$\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$$



Dropout : A Quick Review

- Dropout[Hinton *et al*.] is a technique to avoid over-fitting the data.
- Sample two binary vectors $\mathbf{b}_1$, $\mathbf{b}_2$ of dimensions Q and K respectively.
- The elements of $\mathbf{b}_i$ take a value 1 with probability $\mathbf{p}_i$ (i=1,2)
- Essentially, we set a fraction of inputs to each layer to zero.
- The dropout model's output is

$$\hat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{b}_1\mathbf{W}_1 + \mathbf{b})\mathbf{b}_2\mathbf{W}_2.$$



Dropout : A Quick Review

- Suppose we have $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ as N observed outputs for inputs

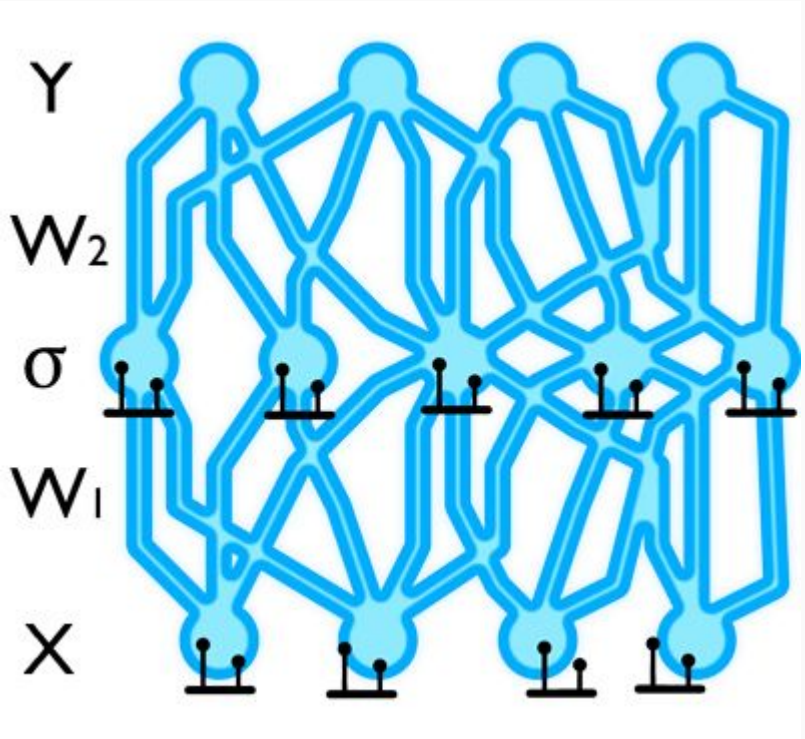
  $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

- For regression, we can use the Euclidean loss

$$E = \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{y}_n - \hat{\mathbf{y}}_n||_2^2$$

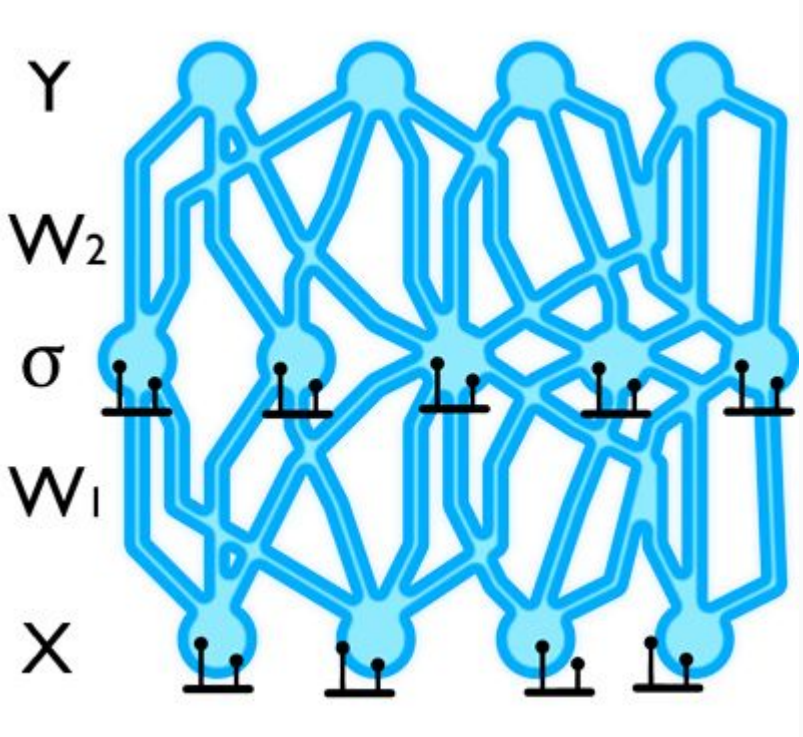
- Typically we also add L_2 regularisation :

$$\mathcal{L}_{\text{dropout}} := E + \lambda_1 ||\mathbf{W}_1||_2^2 + \lambda_2 ||\mathbf{W}_2||_2^2 + \lambda_3 ||\mathbf{b}||_2^2.$$



Dropout : A Quick Review

- During training, we sample random binary vectors $\mathbf{b}_i$ for each input and for each forward pass
- Weights are typically scaled by $1/p_i$ maintain constant output scale
- At test time, we do a single forward pass.



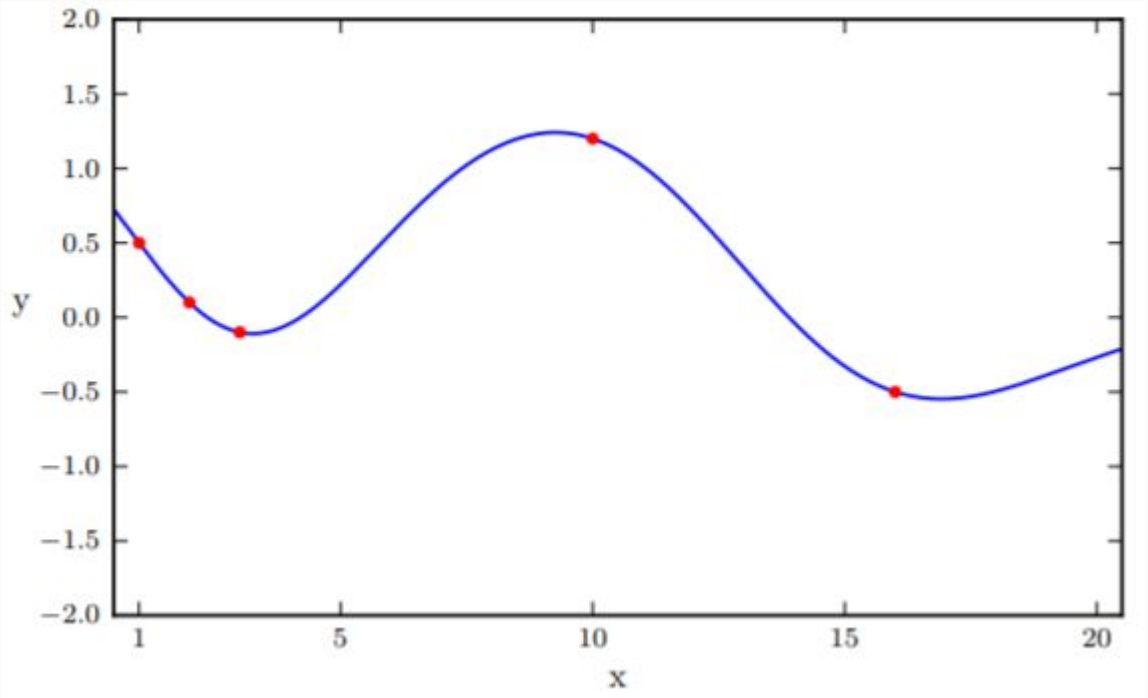
Gaussian Process

Textbook Definition :

A Gaussian process is a collection of random variables, any finite number of which have consistent Gaussian distributions.

But What *is* a Gaussian Process?

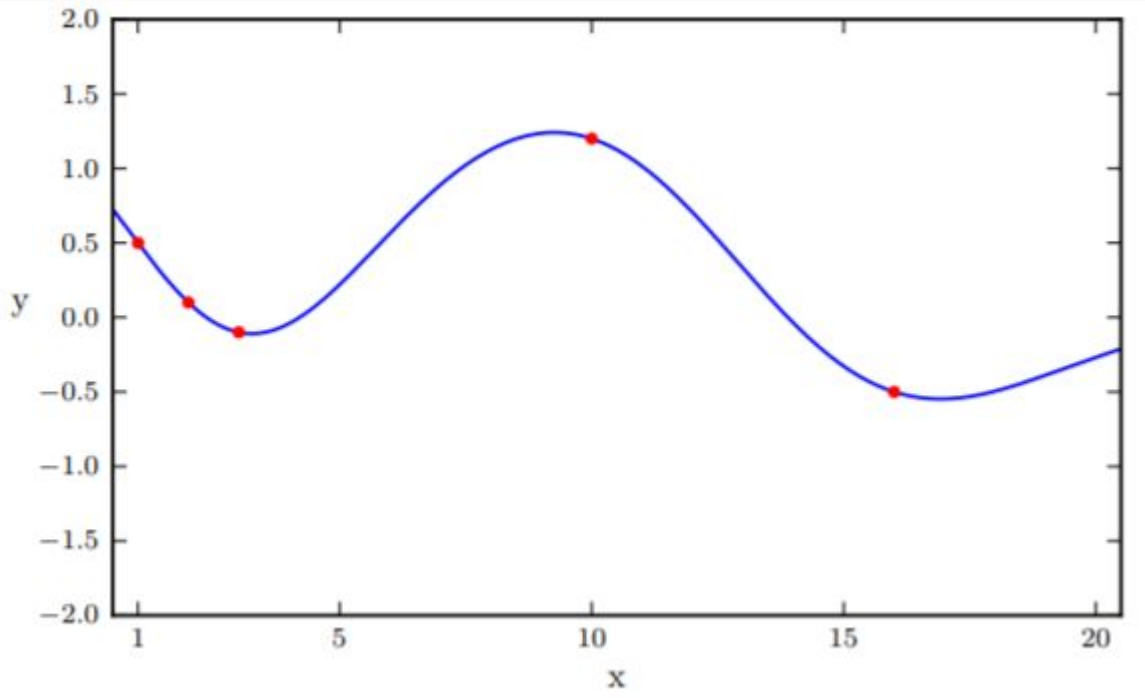
- The problem : What are the possible functions that could have generated our data?



Consider a typical non-linear regression problem. One fit for a given set of points is given here

But What *is* a Gaussian Process?

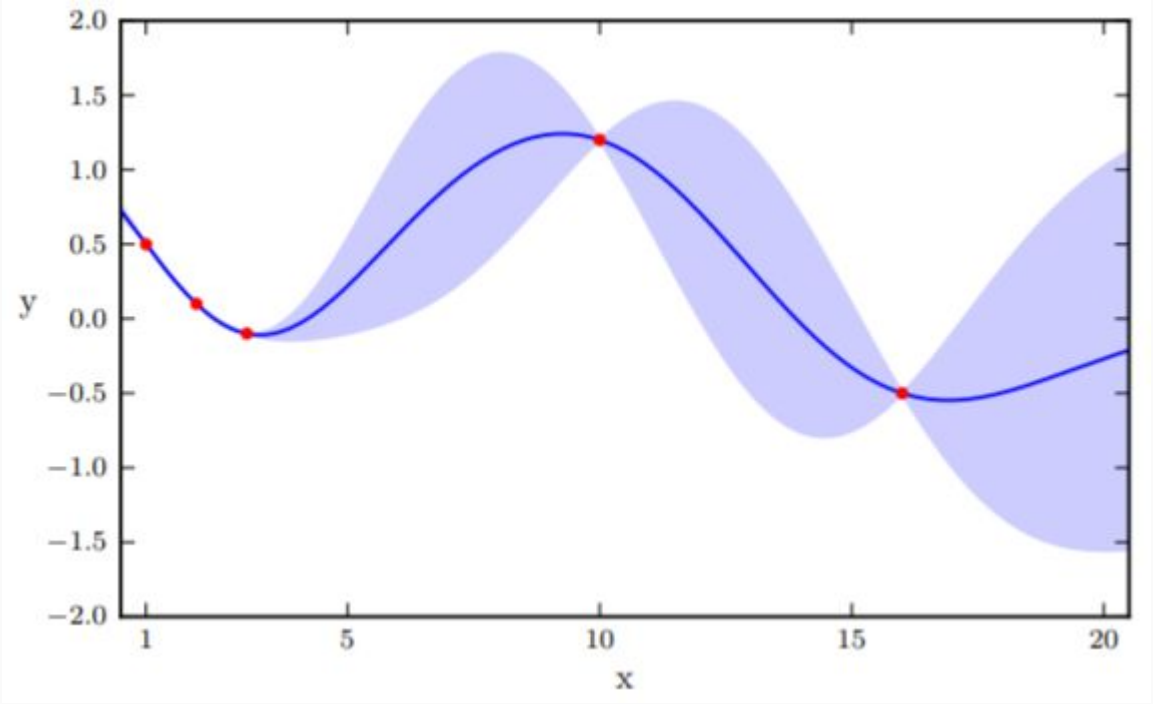
- The problem : What are the possible functions that could have generated our data?



What other curves can explain our data?
With what *certainty* can we predict values for test samples?

But What *is* a Gaussian Process?

- The problem : What are the possible functions that could have generated our data?



Gaussian Processes allow us to model *distributions over functions that fit our data.*

But What *is* a Gaussian Process?

- Let the inputs be denoted by $\mathbf{X} \in \mathbb{R}^{N \times Q}$ and outputs $\mathbf{Y} \in \mathbb{R}^{N \times D}$
- We first place a prior p(**f**) over the space of functions.
- The posterior distribution of the space of functions given the data is

$$p(\mathbf{f}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \mathbf{f})p(\mathbf{f})$$

- In a Gaussian process, we model the posteriors as :

$$\mathbf{F} \mid \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}))$$
$$\mathbf{Y} \mid \mathbf{F} \sim \mathcal{N}(\mathbf{F}, \tau^{-1}\mathbf{I}_N)$$

Where K(. , .) - Covariance function

τ - Precision parameter

Variational Inference

- Inference in a Gaussian process requires computing of inverse of an NxN matrix - Intractable!
- An approximation would be to condition the model on some random variables **w**
- For a new datapoint **x* :** $$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})\, \mathrm{d}\boldsymbol{\omega},$$
- Computing the posterior of **w** is generally intractable.
- Use an approximate *variational* distribution q(**w**)
- We minimize the KL divergence :

$$\mathrm{KL}(q(\boldsymbol{\omega}) \mid p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})),$$

- Approximate predictive distribution : $$q(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}) q(\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}.$$

Variational Inference

- Minimising the KL divergence <---> Maximising the *log evidence lower bound :*

$$\mathcal{L}_{\text{VI}} := \int q(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} - \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$

- Explain the data well + Capture the prior!
- What follows : A variational approximation to the Gaussian Process posterior
- This turns out to be equivalent to use of dropout in standard DNNs!

Dropout as a Bayesian Approximation

3.1 A Gaussian Process Approximation

We begin by defining our covariance function. Let $\sigma(\cdot)$ be some non-linear function such as the rectified linear (ReLU) or the hyperbolic tangent function (TanH). We define $\mathbf{K}(\mathbf{x}, \mathbf{y})$ to be

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{w})p(b)\sigma(\mathbf{w}^T\mathbf{x} + b)\sigma(\mathbf{w}^T\mathbf{y} + b)\mathrm{d}\mathbf{w}\mathrm{d}b$$

with $p(\mathbf{w})$ a standard multivariate normal distribution of dimensionality Q and some distribution $p(b)$. It is trivial to show that this defines a valid covariance function following [Tsuda et al., 2002].

We use Monte Carlo integration with K terms to approximate the integral above. This results in the finite rank covariance function

$$\widehat{\mathbf{K}}(\mathbf{x}, \mathbf{y}) = \frac{1}{K}\sum_{k=1}^{K}\sigma(\mathbf{w}_k^T\mathbf{x} + b_k)\sigma(\mathbf{w}_k^T\mathbf{y} + b_k)$$

with $\mathbf{w}_k \sim p(\mathbf{w})$ and $b_k \sim p(b)$. K will be the number of hidden units in our single hidden layer NN approximation.

Using $\widehat{\mathbf{K}}$ instead of $\mathbf{K}$ as the covariance function of the Gaussian process yields the following generative model:

$$\begin{aligned}
\mathbf{w}_k &\sim p(\mathbf{w}),\ b_k \sim p(b), \\
\mathbf{W}_1 &= [\mathbf{w}_k]_{k=1}^K, \mathbf{b} = [b_k]_{k=1}^K \\
\widehat{\mathbf{K}}(\mathbf{x}, \mathbf{y}) &= \frac{1}{K} \sum_{k=1}^{K} \sigma(\mathbf{w}_k^T \mathbf{x} + b_k) \sigma(\mathbf{w}_k^T \mathbf{y} + b_k) \\
\mathbf{F} \mid \mathbf{X}, \mathbf{W}_1, \mathbf{b} &\sim \mathcal{N}(\mathbf{0}, \widehat{\mathbf{K}}(\mathbf{X}, \mathbf{X})) \\
\mathbf{Y} \mid \mathbf{F} &\sim \mathcal{N}(\mathbf{F}, \tau^{-1} \mathbf{I}_N),
\end{aligned} \tag{8}$$

with $\mathbf{W}_1$ a $Q \times K$ matrix parametrising our covariance function.

Integrating over the covariance function parameters results in the following predictive distribution:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{F}) p(\mathbf{F}|\mathbf{W}_1, \mathbf{b}, \mathbf{X}) p(\mathbf{W}_1) p(\mathbf{b})$$

where the integration is with respect to $\mathbf{F}$, $\mathbf{W}_1$, and $\mathbf{b}$.

Denoting the $1 \times K$ row vector

$$\phi(\mathbf{x}, \mathbf{W}_1, \mathbf{b}) = \sqrt{\frac{1}{K}}\sigma(\mathbf{W}_1^T\mathbf{x} + \mathbf{b})$$

and the $N \times K$ feature matrix $\Phi = [\phi(\mathbf{x}_n, \mathbf{W}_1, \mathbf{b})]_{n=1}^N$, we have $\widehat{\mathbf{K}}(\mathbf{X}, \mathbf{X}) = \Phi\Phi^T$. We rewrite $p(\mathbf{Y}|\mathbf{X})$ as

$$p(\mathbf{Y}|\mathbf{X}) = \int \mathcal{N}(\mathbf{Y}; \mathbf{0}, \Phi\Phi^T + \tau^{-1}\mathbf{I}_N)p(\mathbf{W}_1)p(\mathbf{b})\mathrm{d}\mathbf{W}_1\mathrm{d}\mathbf{b},$$

analytically integrating with respect to $\mathbf{F}$.

The normal distribution of $\mathbf{Y}$ inside the integral above can be written as a joint normal distribution over $\mathbf{y}_d$, the d'th columns of the $N \times D$ matrix $\mathbf{Y}$, for $d = 1, ..., D$. For each term in the joint distribution, following identity [Bishop, 2006, page 93], we introduce a $K \times 1$ auxiliary random variable $\mathbf{w}_d \sim \mathcal{N}(0, \mathbf{I}_K)$,

$$\mathcal{N}(\mathbf{y}_d; 0, \Phi\Phi^T + \tau^{-1}\mathbf{I}_N) = \int \mathcal{N}(\mathbf{y}_d; \Phi\mathbf{w}_d, \tau^{-1}\mathbf{I}_N)\mathcal{N}(\mathbf{w}_d; 0, \mathbf{I}_K)\mathrm{d}\mathbf{w}_d. \tag{9}$$

Writing $\mathbf{W}_2 = [\mathbf{w}_d]_{d=1}^D$ a $K \times D$ matrix, the above is equivalent to[3]

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b})p(\mathbf{W}_1)p(\mathbf{W}_2)p(\mathbf{b})$$

where the integration is with respect to $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$.

3.2 Variational Inference in the Approximate Model

Our sufficient statistics are $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$. To perform variational inference in our approximate model we need to define a variational distribution $q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) := q(\mathbf{W}_1)q(\mathbf{W}_2)q(\mathbf{b})$. We define $q(\mathbf{W}_1)$ to be a Gaussian mixture distribution with two components, factorised over Q:[4]

$$q(\mathbf{W}_1) = \prod_{q=1}^{Q} q(\mathbf{w}_q), \tag{10}$$

$$q(\mathbf{w}_q) = p_1 \mathcal{N}(\mathbf{m}_q, \boldsymbol{\sigma}^2 \mathbf{I}_K) + (1 - p_1)\mathcal{N}(0, \boldsymbol{\sigma}^2 \mathbf{I}_K)$$

with some probability $p_1 \in [0, 1]$, scalar $\boldsymbol{\sigma} > 0$ and $\mathbf{m}_q \in \mathbb{R}^K$. We put a similar approximating distribution over $\mathbf{W}_2$:

$$q(\mathbf{W}_2) = \prod_{k=1}^{K} q(\mathbf{w}_k), \tag{11}$$

$$q(\mathbf{w}_k) = p_2 \mathcal{N}(\mathbf{m}_k, \boldsymbol{\sigma}^2 \mathbf{I}_D) + (1 - p_2)\mathcal{N}(0, \boldsymbol{\sigma}^2 \mathbf{I}_D)$$

with some probability $p_2 \in [0, 1]$.

We put a simple Gaussian approximating distribution over $\mathbf{b}$:

$$q(\mathbf{b}) = \mathcal{N}(\mathbf{m}, \boldsymbol{\sigma}^2 \mathbf{I}_K). \tag{12}$$

3.3 Evaluating the Log Evidence Lower Bound for Regression

We need to evaluate the log evidence lower bound:

$$\mathcal{L}_{\text{GP-VI}} := \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) - \text{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})||p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})), \tag{13}$$

where the integration is with respect to $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$.

For the task of regression we can rewrite the integrand as a sum:

$$\begin{aligned}\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) &= \sum_{d=1}^{D} \log \mathcal{N}(\mathbf{y}_d; \Phi\mathbf{w}_d, \tau^{-1}\mathbf{I}_N) \\ &= -\frac{ND}{2}\log(2\pi) + \frac{ND}{2}\log(\tau) - \sum_{d=1}^{D}\frac{\tau}{2}||\mathbf{y}_d - \Phi\mathbf{w}_d||_2^2,\end{aligned}$$

as the output dimensions of a multi-output Gaussian process are assumed to be independent. Denote $\widehat{\mathbf{Y}} = \Phi\mathbf{W}_2$. We can then sum over the rows instead of the columns of $\widehat{\mathbf{Y}}$ and write

$$\sum_{d=1}^{D}\frac{\tau}{2}||\mathbf{y}_d - \widehat{\mathbf{y}}_d||_2^2 = \sum_{n=1}^{N}\frac{\tau}{2}||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2.$$

Here $\widehat{\mathbf{y}}_n = \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{W}_1, \mathbf{b})\mathbf{W}_2 = \sqrt{\frac{1}{K}}\sigma(\mathbf{x}_n\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$, resulting in the integrand

$$\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = \sum^{N} \log \mathcal{N}(\mathbf{y}_n; \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{W}_1, \mathbf{b})\mathbf{W}_2, \tau^{-1}\mathbf{I}_D).$$

This allows us to write the log evidence lower bound as

$$\sum_{n=1}^{N} \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \log p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) - \mathrm{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) || p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})).$$

We re-parametrise the integrands in the sum to not depend on $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{b}$ directly, but instead on the standard normal distribution and the Bernoulli distribution. Let $q(\boldsymbol{\epsilon}_1) = \mathcal{N}(\mathbf{0}, \mathbf{I}_{Q \times K})$ and $q(\mathbf{z}_{1,q}) = \text{Bernoulli}(p_1)$ for $q = 1, ..., Q$, and $q(\boldsymbol{\epsilon}_2) = \mathcal{N}(\mathbf{0}, \mathbf{I}_{K \times D})$ and $q(\mathbf{z}_{2,k}) = \text{Bernoulli}(p_2)$ for $k = 1, ..., K$. Finally let $q(\boldsymbol{\epsilon}) = \mathcal{N}(0, \mathbf{I}_K)$. We write

$$\begin{aligned} \mathbf{W}_1 &= \mathbf{z}_1(\mathbf{M}_1 + \boldsymbol{\sigma}\boldsymbol{\epsilon}_1) + (1 - \mathbf{z}_1)\boldsymbol{\sigma}\boldsymbol{\epsilon}_1, \\ \mathbf{W}_2 &= \mathbf{z}_2(\mathbf{M}_2 + \boldsymbol{\sigma}\boldsymbol{\epsilon}_2) + (1 - \mathbf{z}_2)\boldsymbol{\sigma}\boldsymbol{\epsilon}_2, \\ \mathbf{b} &= \mathbf{m} + \boldsymbol{\sigma}\boldsymbol{\epsilon}, \end{aligned} \tag{14}$$

allowing us to re-write the sum over the integrals in the above equation as

$$\begin{aligned} &\sum_{n=1}^{N} \int q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \log p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) \mathrm{d}\mathbf{W}_1 \mathrm{d}\mathbf{W}_2 \mathrm{d}\mathbf{b} \\ &= \sum_{n=1}^{N} \int q(\mathbf{z}_1, \boldsymbol{\epsilon}_1, \mathbf{z}_2, \boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}) \log p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}_1(\mathbf{z}_1, \boldsymbol{\epsilon}_1), \mathbf{W}_2(\mathbf{z}_2, \boldsymbol{\epsilon}_2), \mathbf{b}(\boldsymbol{\epsilon})) \end{aligned}$$

where each integration is over $\boldsymbol{\epsilon}_1, \mathbf{z}_1, \boldsymbol{\epsilon}_2, \mathbf{z}_2$, and $\boldsymbol{\epsilon}$.

$$\mathcal{L}_{\text{GP-MC}} := \sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{x}_n, \widehat{\mathbf{W}}_1^n, \widehat{\mathbf{W}}_2^n, \widehat{\mathbf{b}}^n) - \text{KL}(q(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b})||p(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}))$$

with realisations $\widehat{\mathbf{W}}_1^n, \widehat{\mathbf{W}}_2^n, \widehat{\mathbf{b}}^n$ defined following eq. (14) with $\widehat{\epsilon}_1^n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{Q\times K})$, $\widehat{\mathbf{z}}_{1,q}^n \sim$ Bernoulli(p_1), $\widehat{\epsilon}_2^n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{K\times D})$, and $\widehat{\mathbf{z}}_{2,k}^n \sim$ Bernoulli(p_2). Following [Blei et al., 2012; Hoffman et al., 2013; Kingma and Welling, 2013; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014], optimising the *stochastic* objective $\mathcal{L}_{\text{GP-MC}}$ we would converge to the same limit as $\mathcal{L}_{\text{GP-VI}}$.

This is obtained by Monte-Carlo Integration.
For the second term, we can't calculate the KL divergence between a Gaussian mixture and single Gaussian analytically. So, we do a Monte-Carlo approximation of the same.

$$\text{KL}(q(\mathbf{W}_1)||p(\mathbf{W}_1)) \approx QK(\boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - 1) + \frac{p_1}{2}\sum_{q=1}^{Q} \mathbf{m}_q^T \mathbf{m}_q + C$$

$$\text{KL}(q(\mathbf{b})||p(\mathbf{b})) = \frac{1}{2}\left(\mathbf{m}^T\mathbf{m} + K(\boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - 1)\right) + C$$

This is calculated analytically.

3.4 Log Evidence Lower Bound Optimisation

Ignoring the constant terms $\tau, \boldsymbol{\sigma}$ we obtain the maximisation objective

$$\mathcal{L}_{\text{GP-MC}} \propto -\frac{\tau}{2}\sum_{n=1}^{N}||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 - \frac{p_1}{2}||\mathbf{M}_1||_2^2 - \frac{p_2}{2}||\mathbf{M}_2||_2^2 - \frac{1}{2}||\mathbf{m}||_2^2. \tag{15}$$

Note that in the Gaussian processes literature the terms $\tau, \boldsymbol{\sigma}$ will often be optimised as well.

$$\widehat{\mathbf{W}}_1^n \approx \widehat{\mathbf{z}}_1^n\mathbf{M}_1, \quad \widehat{\mathbf{W}}_2^n \approx \widehat{\mathbf{z}}_2^n\mathbf{M}_2, \quad \widehat{\mathbf{b}}^n \approx \mathbf{m}.$$

Note that $\widehat{\mathbf{W}}_1^n$ are not maximum a posteriori (MAP) estimates, but random variable realisations. This gives us

$$\widehat{\mathbf{y}}_n \approx \sqrt{\frac{1}{K}}\sigma(\mathbf{x}_n(\widehat{\mathbf{z}}_1^n\mathbf{M}_1) + \mathbf{m})(\widehat{\mathbf{z}}_2^n\mathbf{M}_2).$$

Scaling the optimisation objective by a positive constant $\frac{1}{\tau N}$ doesn't change the parameter values at its optimum (as long as we don't optimise with respect to τ). We thus scale the objective to get

$$\mathcal{L}_{\text{GP-MC}} \propto -\frac{1}{2N}\sum_{n=1}^{N}||\mathbf{y}_n - \widehat{\mathbf{y}}_n||_2^2 - \frac{p_1}{2\tau N}||\mathbf{M}_1||_2^2 - \frac{p_2}{2\tau N}||\mathbf{M}_2||_2^2 - \frac{1}{2\tau N}||\mathbf{m}||_2^2 \tag{16}$$

Experiments

- Need the model to be uncertain in prediction when it sees a point far away from the training set.
- Can be used for dataset augmentation and is of paramount importance in real life high risk applications.
- Experiments on a subset of the CO_2 dataset collected by Keeling et al. , the solar irradiance dataset described in NASA's website, and, Image classification experiments on the MNIST dataset.

Handling points away from the dataset.

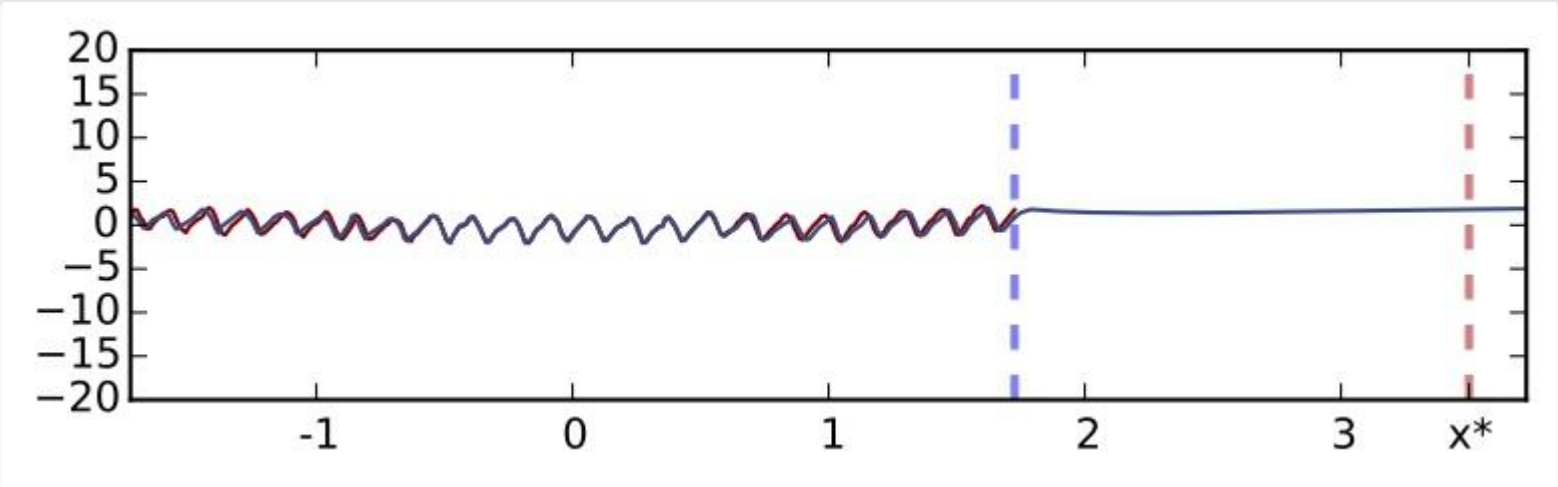
- When asked about a point far away from the dataset, a model must say that it is uncertain in its prediction.
- Standard Neural Networks predict an insensible value with high probability.
- Using softmax probabilities to get uncertainty is not a valid measure (will be shown in an example later)
- Standard dropout networks can be directly used to obtain uncertainty estimates by passing an input sample through it several times and calculating the mean and variance.

$$\mathbb{E}(\mathbf{y}^*) \approx \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}_t^*(\mathbf{x}^*)$$

$$\mathrm{Var}(\mathbf{y}^*) \approx \tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}_t^*(\mathbf{x}^*)^T\widehat{\mathbf{y}}_t^*(\mathbf{x}^*) - \mathbb{E}(\mathbf{y}^*)^T\mathbb{E}(\mathbf{y}^*)$$

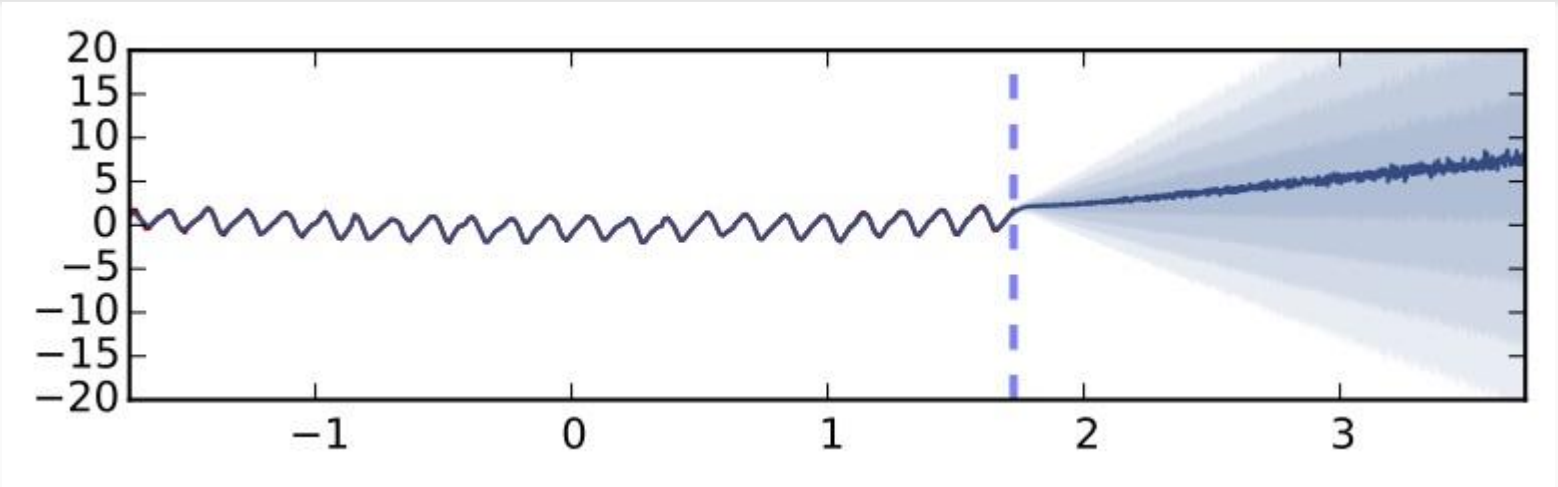
$$\tau = \frac{l^2 p}{2N\lambda}$$

Results without taking care of uncertainty estimates



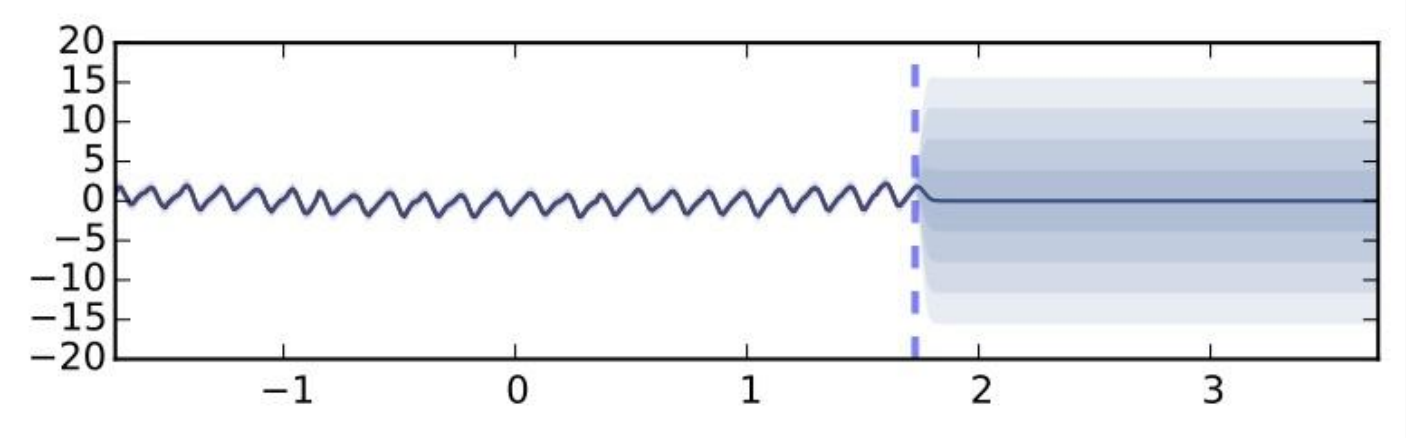
- The training set consists of points only till the blue line. The red line represents a point far from the train set.
- No way for us to tell the model is uncertain or not using a point estimate.

Uncertainty estimates away from the dataset



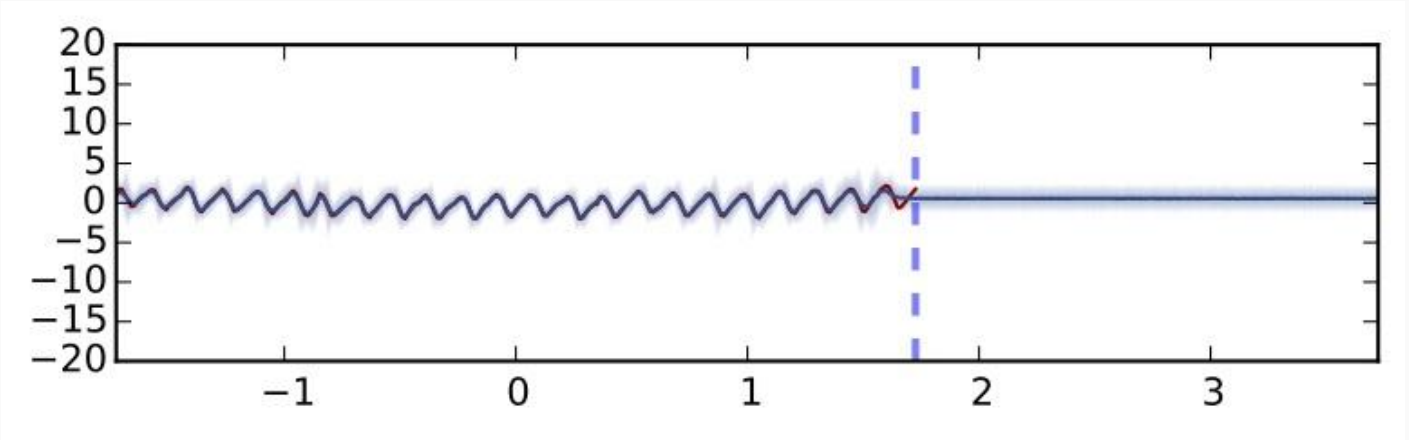
- Exactly the same dropout network performing predictions using uncertainty information and predictive mean (ReLU nonlinearity between layers).
- Clearly the model captures a large amount of uncertain information for the far away test point.
- The uncertainty increases as we go away from the data.

Comparison to a Gaussian Process



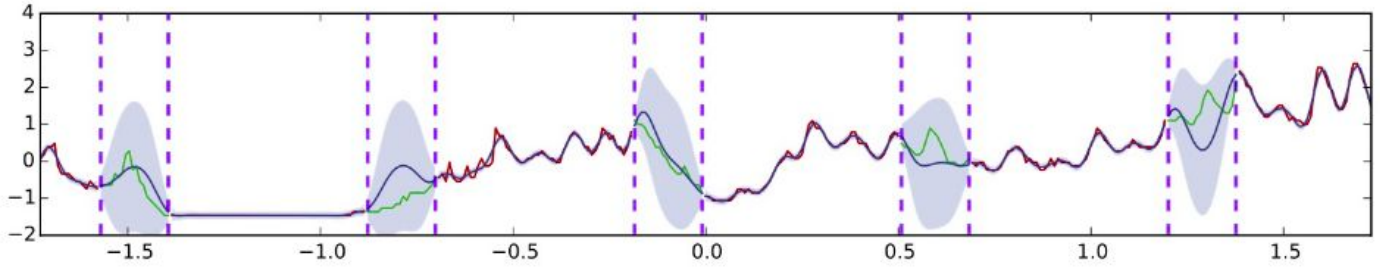
- Uncertainty obtained using a Gaussian Process and a Squared Error(SE) covariance function.
- The uncertainty is clearly higher at points away from the data.
- The estimates look different, however, which should be expected as the ReLU non linearity in the neural network is essentially a different covariance function.

Effect of changing the non-linearity

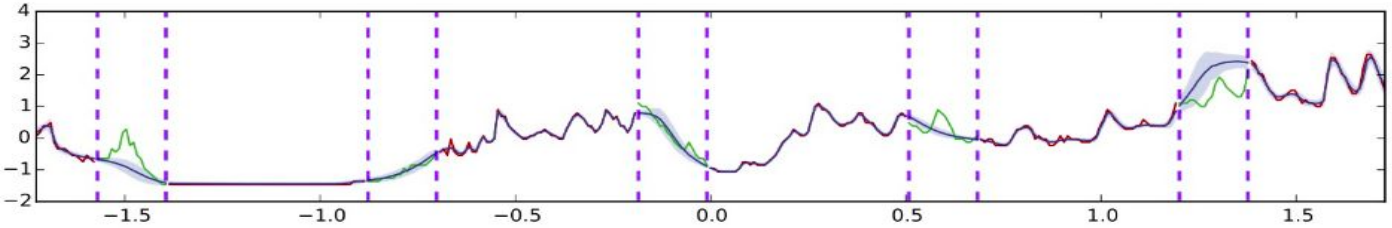


- Uncertainty estimates using a dropout network when the nonlinearity used it Tanh.
- The uncertainty doesn't increase far from the data.
- Might be because TanH saturates whereas ReLU does not.
- Not appropriate for tasks where we uncertainty to increase as we go further away from the data.

Experiment on the Solar Irradiance dataset



Gaussian process with SE covariance function

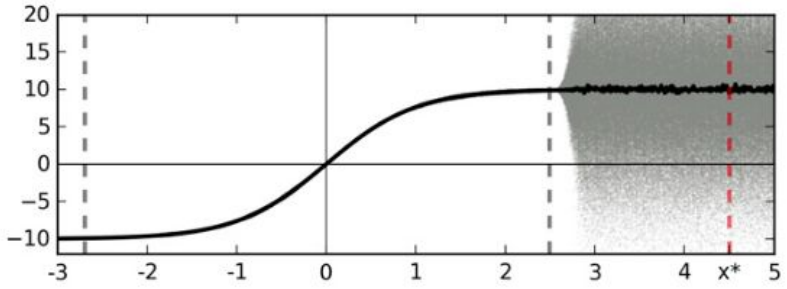


Dropout network using uncertainty information on the same dataset (5 hidden layers, ReLU non-linearity)

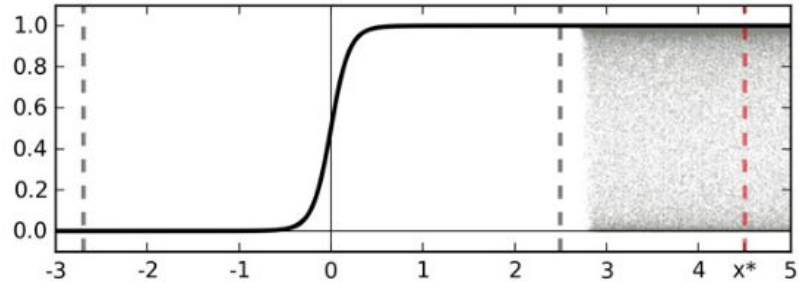
Experiment on the Solar Irradiance dataset

- Considering the case of interpolation of missing data.
- The green and red lines show the actual data samples and the blue line shows the predictive mean.
- The model fits the data very well as well, but with smaller model uncertainty.
- A well known limitation of variational approximations.
- As dropout can be seen as a variational approximation to the Gaussian process, this is not surprising at all.

A binary classification example



Softmax *input* as a function of data $\mathbf{x}$

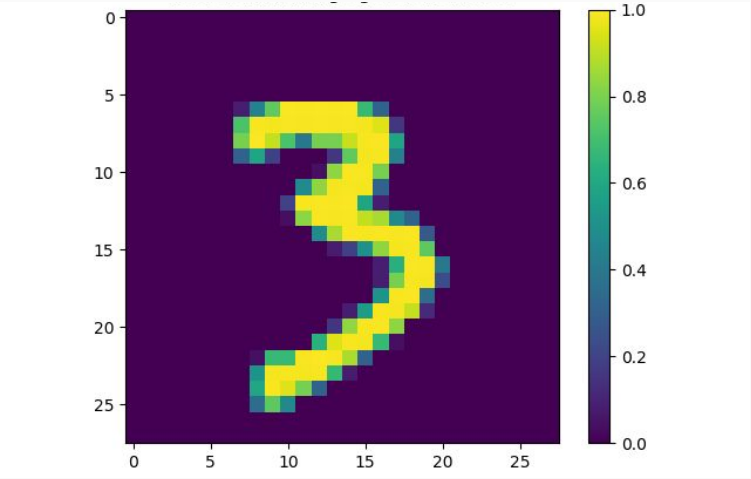


Softmax *output* as a function of data $\mathbf{x}$

A sketch of softmax input and output for an idealised binary classification problem. Training data is given between the dashed grey lines. Function point estimate is shown with a solid line (TanH for simplicity — left). Function uncertainty is shown with a shaded area. Marked with a dashed red line is a point x^* far from the training data. Ignoring function uncertainty, point x^* is classified as class 1 with probability 1.

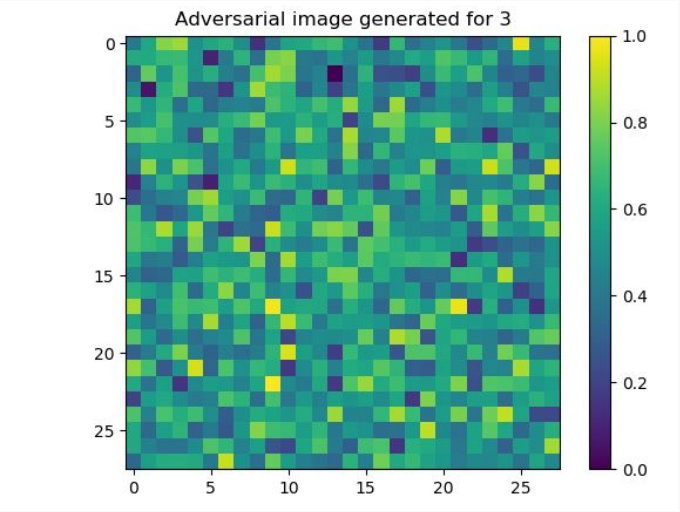
- The softmax probabilities are not enough to say our model is uncertain or not.
- The standard model would pass the predictive mean through the softmax rather than the entire distribution.
- Passing a point estimate of the mean of a function through a softmax results in highly confident extrapolations with x^* (a point far from the training data) classified as class 1 with probability 1.
- Passing the distribution (shaded area on the left) through a softmax (shaded area on the right) reflects classification uncertainty better far from the training data.
- Taking the mean of this distribution passed through the softmax we get class 1 with probability 0.5 — the model's true prediction.

Adversarial example on the MNIST dataset




```
[[0.05855728]
 [0.77182849]
 [0.        ]
 [0.13310686]
 [0.00235378]
 [0.02288536]
 [0.        ]
 [0.00095433]
 [0.0103139 ]
 [0.        ]]
0.036066025677021385
Number to generate : 3
```

```
[[0.]
 [0.]
 [0.]
 [1.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]]
0.0
Number to generate : 3
```



Sourav Sahoo, Subhankar Chakraborty, Sumanth Hegde

Adversarial examples on the MNIST dataset

- Mean probability and variance generated by passing the input through a dropout model 1000 times and taking the mean and variance.
- A clear difference in outputs when a real image was input to the model versus an adversarial example of the digit 3.
- The mean values of the probability clearly tells that the model was uncertain even if we got a high probability.
- The variance, however, is still **not high enough** for us to conclusively speak about the uncertainty estimate.

Future Work

- Tackle adversarial examples using model uncertainty, making the model robust to adversarial attacks.
  - Lewis *et al* show failure cases for MC dropout in detecting adv. Examples.

- Look at Uncertainty Information in other areas of computer vision Kendall et al.

Questions?