Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Yarin Gal | Zoubin Ghahramani

Group 6 | 19/02/2021

E/15/142 - Ganindu
E/15/154 - Chamin
E/15/179 - Anandi
E/15/350 - Pasan

Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Authors

- Yarin Gal
- Zoubin Ghahramani
  [University of Cambridge]

Year

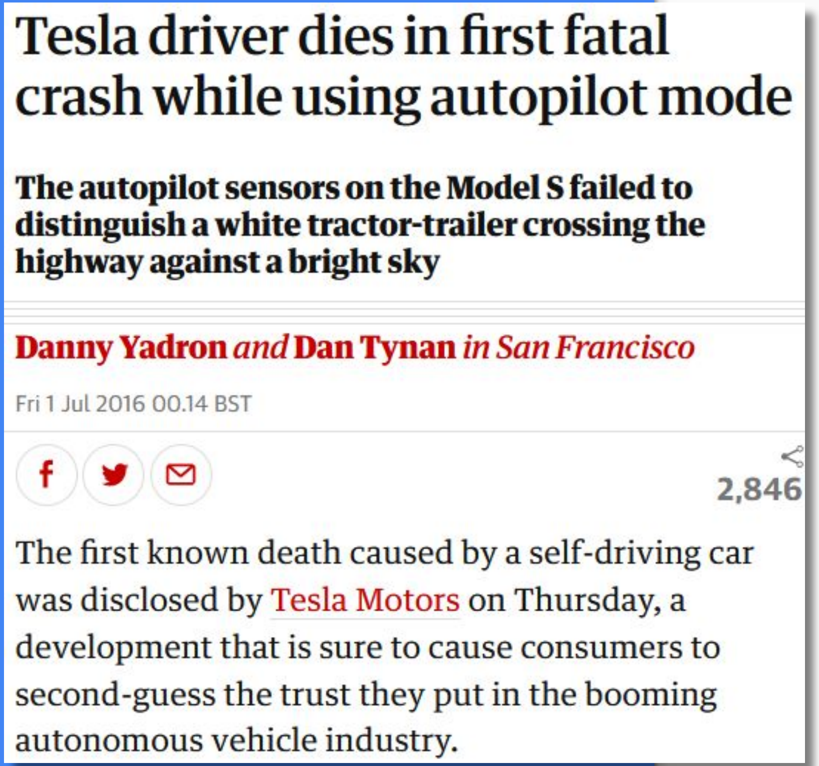
- 2016

Contribution

- A new method to get model uncertainty from existing deep learning models without changing the existing models

Outline

Introduction



Tesla driver dies in first fatal crash while using autopilot mode

The autopilot sensors on the Model S failed to distinguish a white tractor-trailer crossing the highway against a bright sky

Danny Yadron *and* **Dan Tynan** *in San Francisco*

Fri 1 Jul 2016 00.14 BST

2,846

The first known death caused by a self-driving car was disclosed by Tesla Motors on Thursday, a development that is sure to cause consumers to second-guess the trust they put in the booming autonomous vehicle industry.

*"Against a bright spring sky, the car's sensors **system failed to distinguish** a large white 18-wheel truck and trailer crossing the highway, Tesla said."*

Uncertainty

- Given a model trained with several pictures of dog breeds.
- A user asks the model to decide on a dog breed using a photo of a cat.

“We build models for predictions, can we trust them? Are they certain?”

Why should you care about uncertainty?

Sometimes, predictions can be **life threatening!**

Many applications of machine learning depend on good estimation of the uncertainty.

- Medical: automated decision making or recommendation systems
- Automotive: autonomous control of the self driving vehicles
- High frequency trading: ability to affect economic markets on global scale

“A deep learning model should be able to say: *“sorry, I don’t know.”* ”

How to estimate model Uncertainty?

Problem

“Standard deep learning tools for regression and classification do not capture model uncertainty.”

Traditional method: **Bayesian Techniques**

- A mathematical based framework
- Comes with a cost

How to estimate model Uncertainty?

Related Works

- ○ Blei et al., 2012
- ○ Kingma & Welling, 2013
- ○ Rezende et al., 2014
- ○ Titsias & L´azaroGredilla, 2014
- ○ Hoffman et al., 2013

Introduced new techniques
- sampling-based variational inference
- stochastic variational inference

- ○ Blundell et al., 2015

- Obtain new approximations for Bayesian neural networks that perform as well as dropout

- *These models came with a prohibitive computational cost.*
- *They required more time to converge and did not improve on existing techniques.*

How to estimate model Uncertainty?

Solution

A new method to **represent uncertainty using**,

- **Dropout** and
- **Gaussian Processes**

“This mitigates the problem of representing model uncertainty in deep learning without sacrificing either computational complexity or test accuracy.”

Dropout and Gaussian Processes

Dropout

"Dropout is used in many models in deep learning as a way to avoid over-fitting."

E/15/179 Anandi

***What is* Dropout?**

- Ignoring neurons during the training phase
- Neurons are chosen at random
- Each subset of nodes that is not dropped out defines a new network

***Why do we need* Dropout?**

- A regularizer in deep learning to avoid overfitting

Gaussian Processes

"The Gaussian process is a powerful tool in statistics that allows to model distributions over functions."

- Applied in,
  - Both supervised and unsupervised domains
  - And for both regression and classification tasks

The Gaussian process offers

- Uncertainty estimates over the function values
- Robustness to over-fitting
- Methods for hyper-parameter tuning

Why does this work ?

Approximating Bayesian inference in deep Gaussian processes is identical to dropouts in Deep Neural Networks

- Approximate posterior $p(\omega|\mathbf{X},\mathbf{Y})$ with $q_\theta(\omega)$
- KL divergence to minimise:

$$\begin{aligned}&\mathrm{KL}(q_\theta(\omega)\,||\,p(\omega|\mathbf{X},\mathbf{Y}))\\ &\propto \boxed{-\int q_\theta(\omega)\log p(\mathbf{Y}|\mathbf{X},\omega)\mathrm{d}\omega} + \mathrm{KL}(q_\theta(\omega)\,||\,p(\omega))\\ &=: \mathcal{L}(\theta)\end{aligned}$$

- Approximate the integral with MC integration $\widehat{\omega}\sim q_\theta(\omega)$:

$$\widehat{\mathcal{L}}(\theta) := -\log p(\mathbf{Y}|\mathbf{X},\widehat{\omega}) + \mathrm{KL}(q_\theta(\omega)\,||\,p(\omega))$$

- Approximate posterior $p(\omega|\mathbf{X},\mathbf{Y})$ with $q_\theta(\omega)$
- KL divergence to minimise:

$$
\begin{aligned}
&\mathrm{KL}(q_\theta(\omega) \,||\, p(\omega|\mathbf{X},\mathbf{Y})) \\
&\quad \propto \boxed{-\int q_\theta(\omega) \log p(\mathbf{Y}|\mathbf{X},\omega) \mathrm{d}\omega} + \mathrm{KL}(q_\theta(\omega) \,||\, p(\omega)) \\
&\quad =: \mathcal{L}(\theta)
\end{aligned}
$$

- Approximate the integral with MC integration $\widehat{\omega} \sim q_\theta(\omega)$:

$$\widehat{\mathcal{L}}(\theta) := -\log p(\mathbf{Y}|\mathbf{X},\widehat{\omega}) + \mathrm{KL}(q_\theta(\omega) \,||\, p(\omega))$$

- Approximate posterior $p(\omega|\mathbf{X}, \mathbf{Y})$ with $q_\theta(\omega)$
- KL divergence to minimise:

$$\begin{aligned}
&\mathrm{KL}\big(q_\theta(\omega) \;||\; p(\omega|\mathbf{X}, \mathbf{Y})\big) \\
&\quad \propto \boxed{-\int q_\theta(\omega) \log p(\mathbf{Y}|\mathbf{X}, \omega) \mathrm{d}\omega} + \mathrm{KL}\big(q_\theta(\omega) \;||\; p(\omega)\big) \\
&\quad =: \mathcal{L}(\theta)
\end{aligned}$$

- Approximate the integral with MC integration $\widehat{\omega} \sim q_\theta(\omega)$:

$$\widehat{\mathcal{L}}(\theta) := -\log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega}) + \mathrm{KL}\big(q_\theta(\omega) \;||\; p(\omega)\big)$$

- Unbiased estimator:

$$\mathbb{E}_{\widehat{\omega}\sim q_\theta(\omega)}\left(\widehat{\mathcal{L}}(\theta)\right) = \mathcal{L}(\theta)$$

- Converges to the same optima as $\mathcal{L}(\theta)$
- For inference, repeat:
  - Sample $\widehat{\omega} \sim q_\theta(\omega)$
  - And minimise (one step)

$$\widehat{\mathcal{L}}(\theta) = -\log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega}) + \mathrm{KL}\left(q_\theta(\omega) \,||\, p(\omega)\right)$$

w.r.t. θ.

- Unbiased estimator:

$$\mathbb{E}_{\widehat{\omega}\sim q_\theta(\omega)}\left(\widehat{\mathcal{L}}(\theta)\right) = \mathcal{L}(\theta)$$

- Converges to the same optima as $\mathcal{L}(\theta)$
- For inference, repeat:
  - Sample $\widehat{\omega} \sim q_\theta(\omega)$
  - And minimise

$$\widehat{\mathcal{L}}(\theta) = -\log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega}) + \mathrm{KL}(q_\theta(\omega) \mid\mid p(\omega))$$

w.r.t. θ.

- Unbiased estimator:

$$\mathbb{E}_{\widehat{\omega}\sim q_\theta(\omega)}\left(\widehat{\mathcal{L}}(\theta)\right) = \mathcal{L}(\theta)$$

- Converges to the same optima as $\mathcal{L}(\theta)$
- For inference, repeat:
  - Sample $\widehat{\omega} \sim q_\theta(\omega)$
  - And minimise

$$\widehat{\mathcal{L}}(\theta) = -\log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega}) + \mathrm{KL}(q_\theta(\omega) \mid\mid p(\omega))$$

w.r.t. θ.

Specifying $q_\theta(\cdot)$

- Given $\mathbf{z}_{i,j}$ Bernoulli r.v. and variational parameters $\theta = \{\mathbf{M}_i\}_{i=1}^{L}$ (set of matrices):

$$\mathbf{z}_{i,j} \sim \text{Bernoulli}(p_i) \text{ for } i = 1, ..., L,\ j = 1, ..., K_{i-1}$$

$$\mathbf{W}_i = \mathbf{M}_i \cdot \text{diag}([\mathbf{z}_{i,j}]_{j=1}^{K_i})$$

$$q_\theta(\omega) = \prod q_{\mathbf{M}_i}(\mathbf{W}_i)$$

In summary:

Minimise divergence between $q_\theta(\omega)$ and $p(\omega|\mathbf{X}, \mathbf{Y})$:

- Repeat:
  - Sample $\widehat{\mathbf{z}}_{ik} \sim \text{Bernoulli}(p_i)$ and set

$$\widehat{\mathbf{W}}_i = \mathbf{M}_i \cdot \text{diag}([\widehat{\mathbf{z}}_{ik}]_{k=1}^{K})$$
$$\widehat{\omega} = \{\widehat{\mathbf{W}}_i\}_{i=1}^{L}$$

  - Minimise

$$\widehat{\mathcal{L}}(\theta) = -\log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega}) + \text{KL}(q_\theta(\omega) \,||\, p(\omega))$$

w.r.t. $\theta = \{\mathbf{M}_i\}_{i=1}^{L}$ (set of matrices).

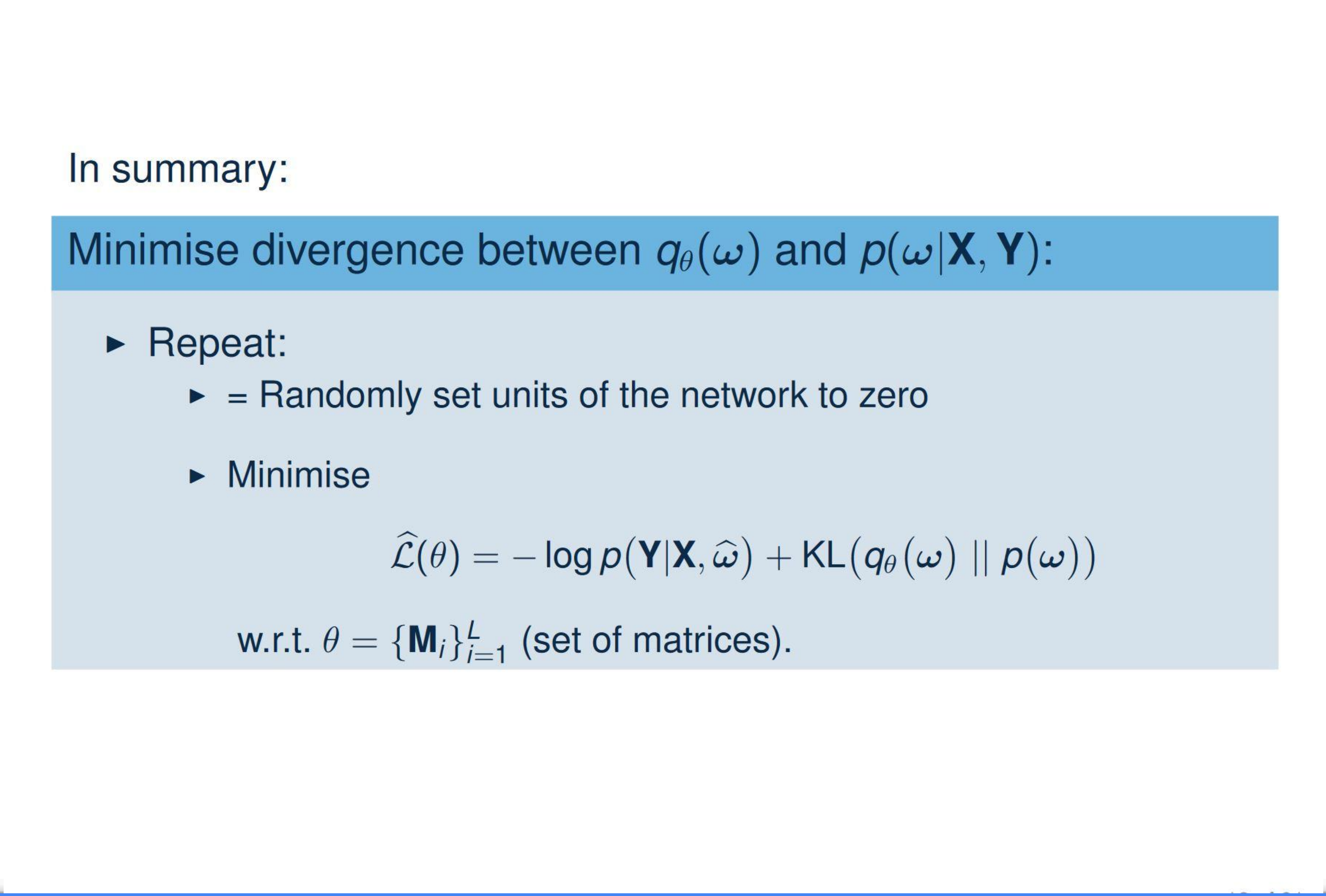
In summary:

Minimise divergence between $q_\theta(\omega)$ and $p(\omega|\mathbf{X},\mathbf{Y})$:

- Repeat:
  - = Randomly set columns of $\mathbf{M}_i$ to zero
  - Minimise

$$\widehat{\mathcal{L}}(\theta) = -\log p(\mathbf{Y}|\mathbf{X},\widehat{\omega}) + \mathrm{KL}(q_\theta(\omega)\ ||\ p(\omega))$$

w.r.t. $\theta = \{\mathbf{M}_i\}_{i=1}^L$ (set of matrices).



In summary:

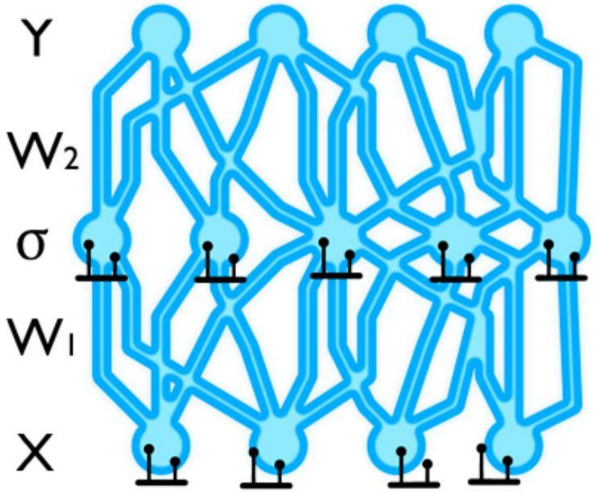
Minimise divergence between $q_\theta(\omega)$ and $p(\omega|\mathbf{X}, \mathbf{Y})$:

- Repeat:
  - = Randomly set units of the network to zero
  - Minimise

$$\widehat{\mathcal{L}}(\theta) = -\log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega}) + \mathrm{KL}(q_\theta(\omega) \,||\, p(\omega))$$

w.r.t. $\theta = \{\mathbf{M}_i\}_{i=1}^{L}$ (set of matrices).

Sounds familiar?



$$\widehat{\mathcal{L}}(\theta) = \overbrace{-\log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega})}^{= \text{loss}} + \overbrace{\mathrm{KL}(q_\theta(\omega) \,||\, p(\omega))}^{= L_2 \text{ reg}}$$

Implementing VI with $q_\theta(\cdot)$ above = implementing dropout in deep network

How to apply to GP?

- GP covariance :

$$\mathbf{K}(\mathbf{x},\mathbf{y}) = \int \mathcal{N}(\mathbf{w}; 0, l^{-2}\mathbf{I}_Q) p(b) \sigma(\mathbf{w}^T\mathbf{x} + b)\sigma(\mathbf{w}^T\mathbf{y} + b) \mathrm{d}\mathbf{w}\mathrm{d}b$$

- Approximate with Monte Carlo integration:

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

- Reparameterized:

$$\mathbf{w}_k \sim \mathcal{N}(0, l^{-2}\mathbf{I}_Q), \quad \mathbf{w}_d \sim \mathcal{N}(0, l^{-2}\mathbf{I}_K), \quad b_k \sim p(b),$$
$$\mathbf{W}_1 = [\mathbf{w}_k]_{k=1}^K, \quad \mathbf{W}_2 = [\mathbf{w}_d]_{d=1}^D, \quad \mathbf{b} = [b_k]_{k=1}^K,$$
$$\omega = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}\}$$
$$p(\mathbf{y}^*|\mathbf{x}^*, \omega) = \mathcal{N}\left(\mathbf{y}^*; \sqrt{\frac{1}{K}}\sigma(\mathbf{x}^*\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2, \tau^{-1}\mathbf{I}_N\right)$$
$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega)p(\omega|\mathbf{X}, \mathbf{Y})\mathrm{d}\omega.$$

- Approximate posterior $p(\omega|\mathbf{X}, \mathbf{Y})$

$$q_\theta(\omega) = q_\theta(\mathbf{W}_1)q_\theta(\mathbf{W}_2)q_\theta(\mathbf{b})$$

$$q_\theta(\mathbf{W}_1) = \prod_{q=1}^{Q} q_\theta(\mathbf{w}_q)$$

$$q_\theta(\mathbf{w}_q) = p_1\mathcal{N}(\mathbf{m}_q, s^2\mathbf{I}_K) + (1 - p_1)\mathcal{N}(0, s^2\mathbf{I}_K)$$

- Monte Carlo integration with $\widehat{\omega} \sim q_\theta(\omega)$:

$$\mathcal{L}_{\text{GP-MC}} \approx \log p(\mathbf{Y}|\mathbf{X}, \widehat{\omega}) - \frac{p_1 l^2}{2}||\mathbf{M}_1||_2^2 - \frac{p_2 l^2}{2}||\mathbf{M}_2||_2^2 - \frac{l^2}{2}||\mathbf{m}||_2^2,$$

Deriving uncertainty

- Imagine a Gaussian $\mathcal{N}(x\ ;\ \mu,\sigma)$
- Variational inference:

  Fit the distribution to posterior

  Optimising over the variational parameters
- Resulting in robustness to over-fitting
- Gaussian distribution captures uncertainty estimate over the weights

Experiments and Results

Procedure

1. Assessment of the properties of uncertainty estimates on the tasks of regression and classification.
2. Comparison of the uncertainty results obtained from different model architectures and non-linearities and show the importance of model uncertainty in classification tasks.
3. Verify using dropout’s uncertainty can result in a considerable improvement in predictive log-likelihood and RMSE compared to existing methods.
4. An example use of the model’s uncertainty in a Bayesian pipeline.

Experiment Setups

1. Regression

Dataset: A subset of the atmospheric **CO_2 concentrations dataset**

(derived from in situ air samples collected at Mauna Loa Observatory, Hawaii, since 1958)

Trained several models using Neural Nets with 4-5 hidden layers and 1024 hidden units

Dropout with probability 0.1 after each weight layer.

Experiment Setups

2. Deep Reinforcement Learning

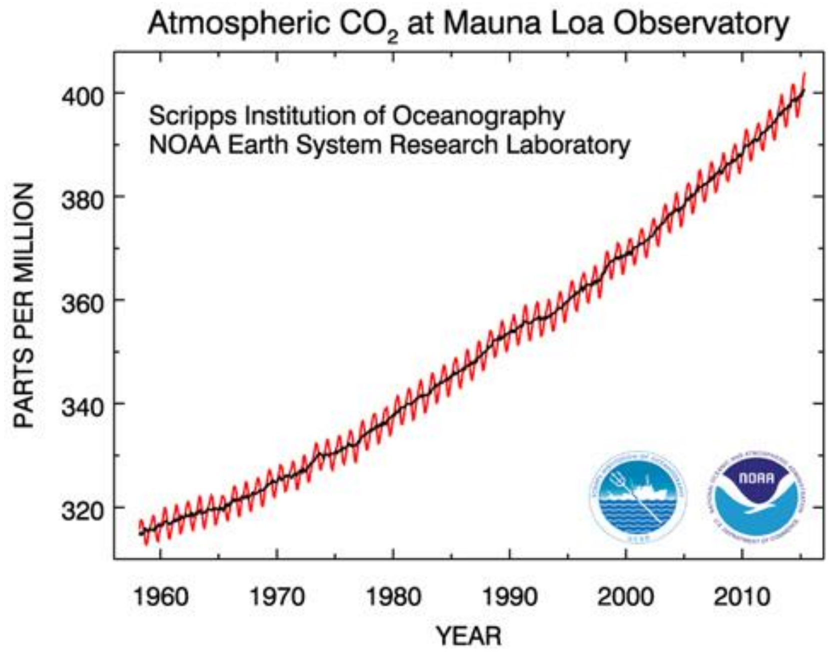
Experiment: A simulated agent environment (using JavaScript) implemented by **Karpathy at el**

Behavioural Policies:

- Epsilon Greedy with a decreasing schedule
- Thompson Sampling

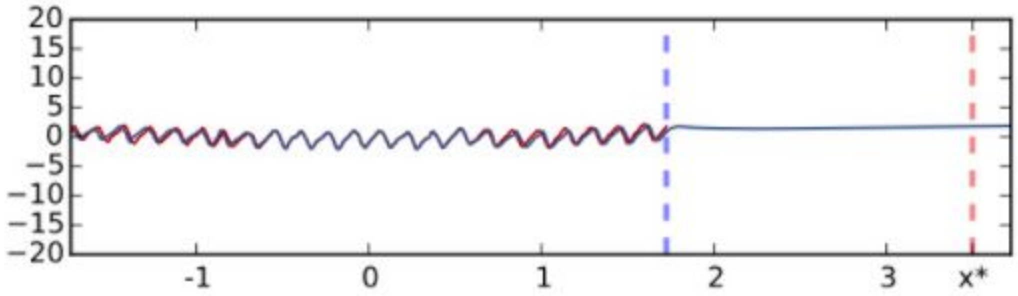
Experiment 1 | Model Uncertainty in Regression Tasks

Raw Data:



Experiment 1 | Model Uncertainty in Regression Tasks

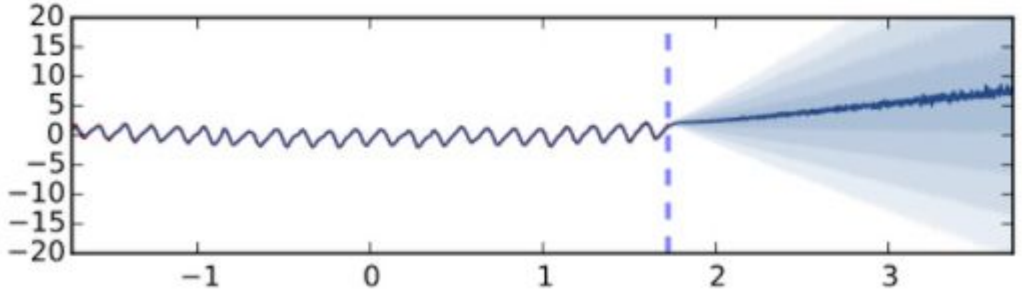
Standard dropout network without using uncertainty information



(5 hidden layers, ReLU non-linearity)

Experiment 1 | Model Uncertainty in Regression Tasks

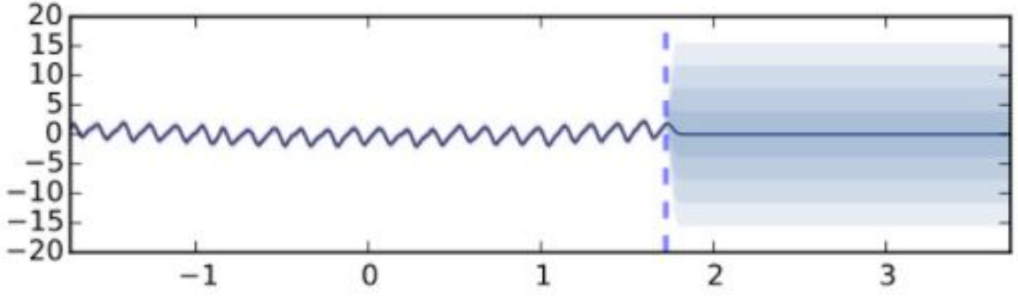
Standard dropout network using uncertainty information and predictive mean



(5 hidden layers, ReLU non-linearity)

Experiment 1 | Model Uncertainty in Regression Tasks

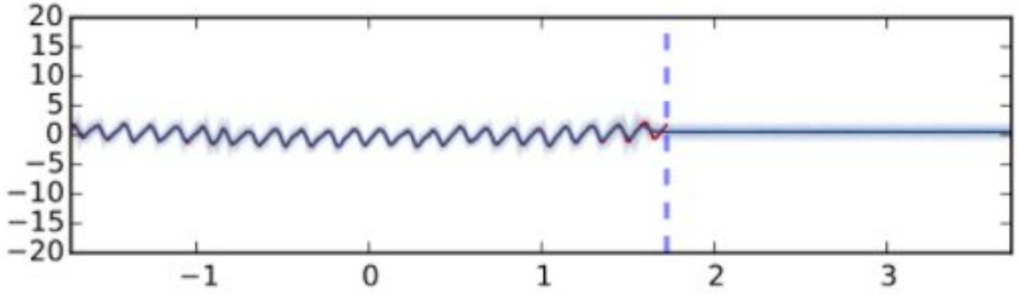
Gaussian process with SE covariance function on the same dataset



(5 hidden layers, ReLU non-linearity)

Experiment 1 | Model Uncertainty in Regression Tasks

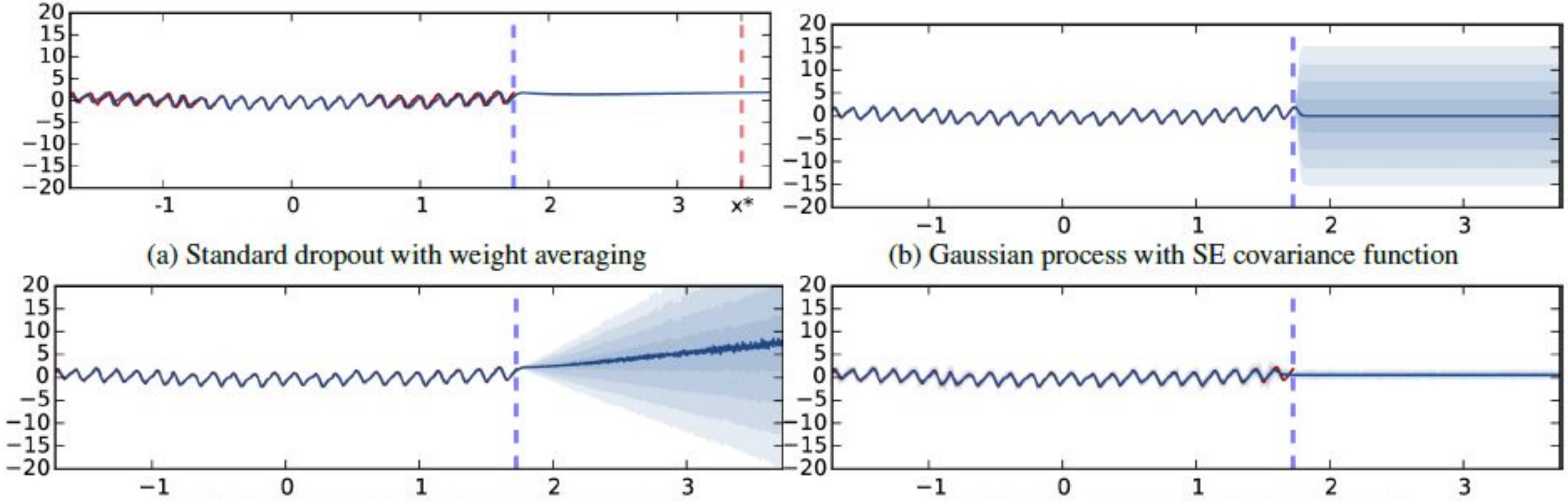
Dropout network using uncertainty information



(5 hidden layers, TanH non-linearity)

Experiment 1 | Model Uncertainty in Regression Tasks

Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning



(a) Standard dropout with weight averaging

(b) Gaussian process with SE covariance function

(c) MC dropout with ReLU non-linearities

(d) MC dropout with TanH non-linearities

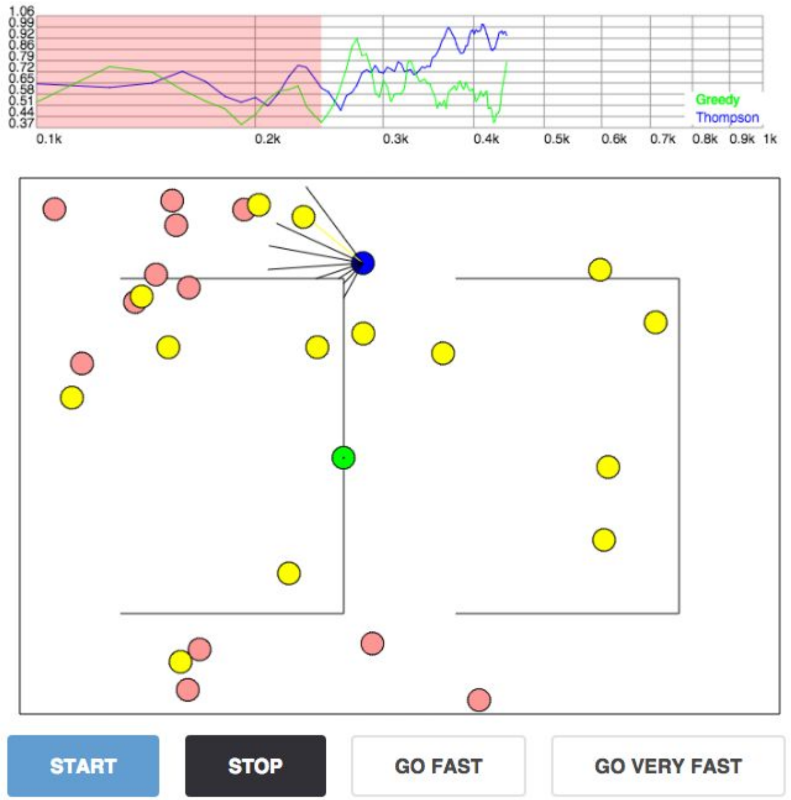
Experiment 2 | Model Uncertainty in Reinforcement Learning

Our simulated setting:

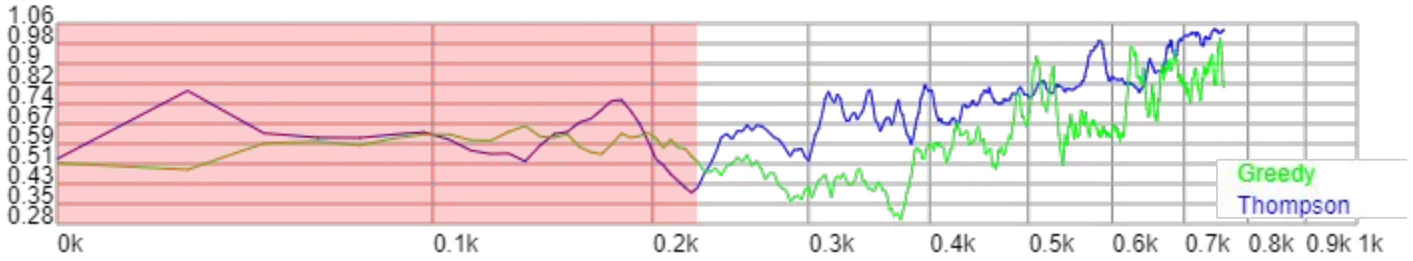
The simulated agent environment implemented by **Karpathy**

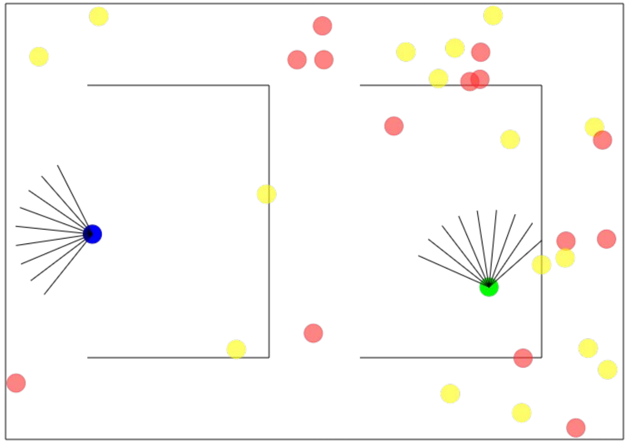
Behavioural settings:

- Epsilon Greedy: no change
- Dropout: single dropout layer with probability of 0.2
- Initial moves: first 3000 are random



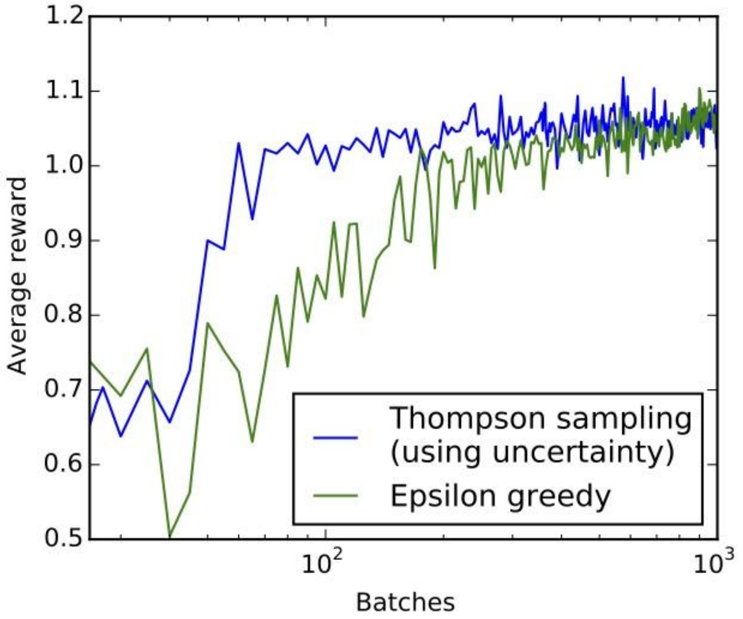
Experiment 2 | Model Uncertainty in Reinforcement Learning





Experiment 2 | Model Uncertainty in Reinforcement Learning

Results: Log plot of average reward



Conclusions

1. A probabilistic interpretation of dropout for obtaining model uncertainty out of existing deep learning models.
2. Demonstration of possible applications, interleaving Bayesian models and deep learning models .
3. Suggest a new interpretation into why dropout works so well as a regularisation technique
4. Further exploration and research work is encouraged and open for discussion.

"This research work would be useful in your research, particularly in data analysis in bioinformatics or image classification in vision systems."

Thank You!