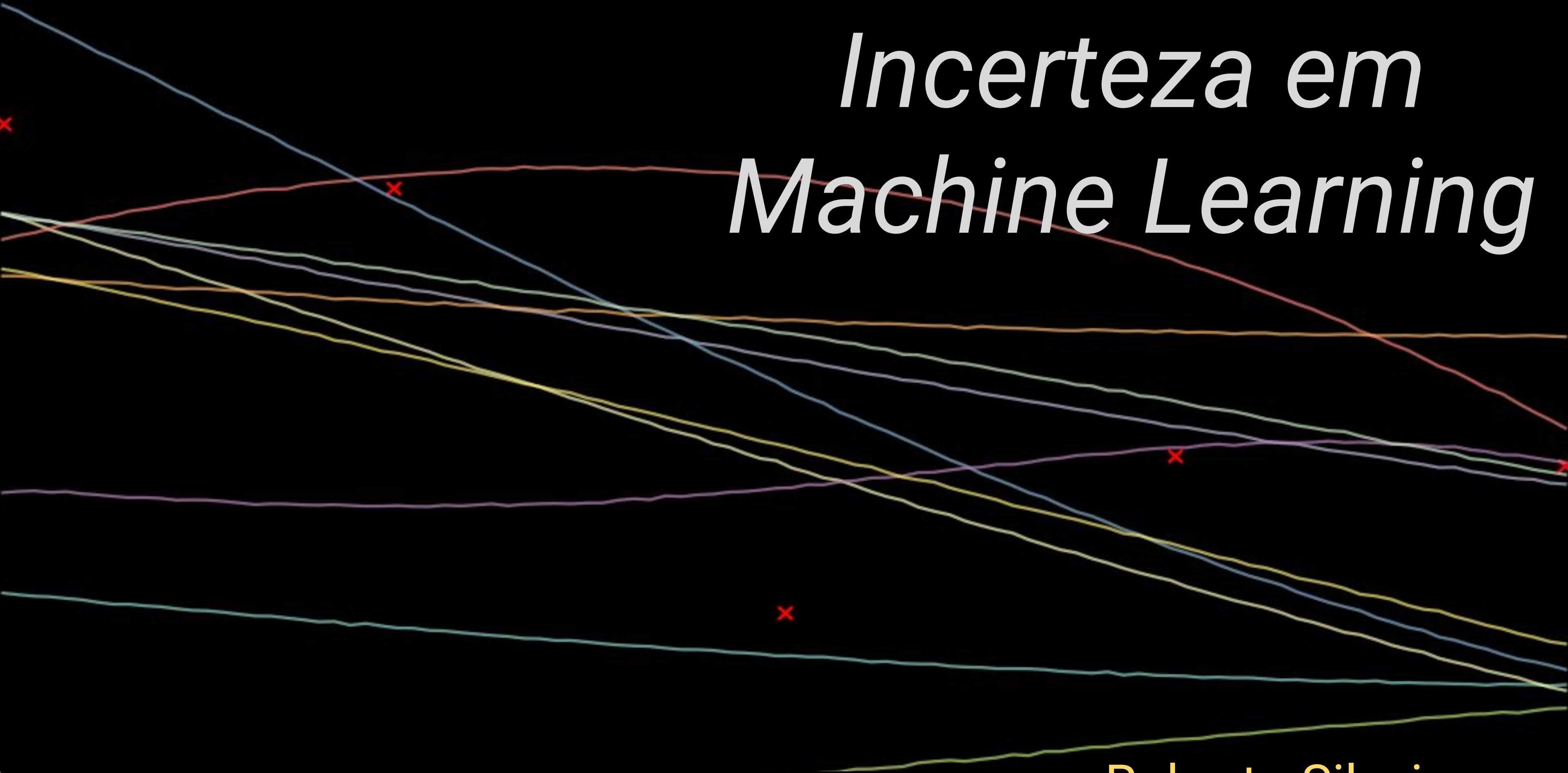
THE DEVELOPER'S CONFERENCE

Trilha – Machine Learning

Roberto Silveira
Engenheiro

Incerteza em Machine Learning



Roberto Silveira

HELLO!

I am Roberto Silveira

EE engineer, ML enthusiast

rsilveira79@gmail.com

@rsilveira79

Why do we care about uncertainty?

Why do we care about uncertainty?

Cat?
Dog?
???

Why do we care about uncertainty?

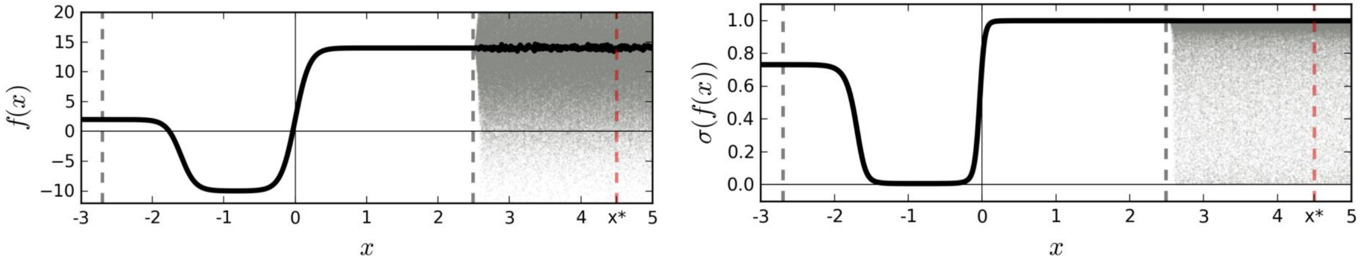
Why do we care about uncertainty?

Why do we care about uncertainty?

A note on
SOFTMAX

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

```
In [73]: import numpy as np

In [79]: samples = np.array([1, 2.0, 0.002, 0.992])

In [80]: def softmax(inp):
             return np.exp(inp)/(np.sum(np.exp(inp)))

In [81]: softmax(samples)

Out[81]: array([0.19689188, 0.53520761, 0.07257748, 0.19532303])

In [82]: samples = np.array([-0.002, -0.1, -0.99, 0.000001])

In [83]: softmax(samples)

Out[83]: array([0.30478768, 0.27633542, 0.11347873, 0.30539817])
```



(a) Arbitrary function $f(\mathbf{x})$ as a function of data $\mathbf{x}$ (softmax *input*)

(b) $\sigma(f(\mathbf{x}))$ as a function of data $\mathbf{x}$ (softmax *output*)

Source: Uncertainty in Deep Learning - PhD Thesis (Gal, 2016)

Demo time

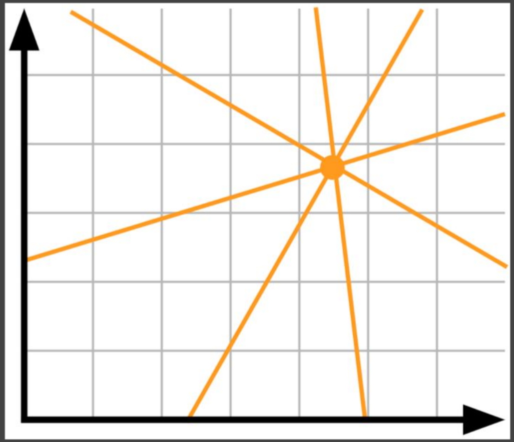
Code:
https://github.com/rsilveira79/intents_uncertainty/blob/master/intent_classifier_uncertainty_mc_dropout.ipynb

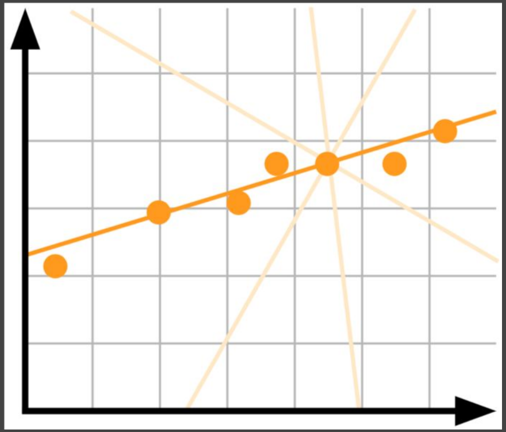
Language of *uncertainty*

Types of Uncertainty

Model uncertainty (epistemic, reducible) = uncertainty in model (either model parameters or model structure) → more data helps

"*epistemic*" → Greeg "*episteme*" = knowledge





Source: Uncertainty in Deep Learning (Yarin Gal, 2016)

Types of Uncertainty

Aleatoric uncertainty (stochastic, irreducible) = uncertainty in data (noise) ⟶ more data doesn't help

"Aleatoric" ⟶ Latin "*aleator*" = "dice player's"

Can be further divided:

- Homoscedastic ⟶ uncertainty is same for all inputs
- Heteroscedastic ⟶ observation (uncertainty) can vary with input **X**

Source: Uncertainty in Deep Learning (Yarin Gal, 2016)

Frequentist vs Bayesian view of the world

$$\hat{p} = \frac{y}{n}$$

$$p(\theta|y)$$

Frequentist vs Bayesian view of the world (2)

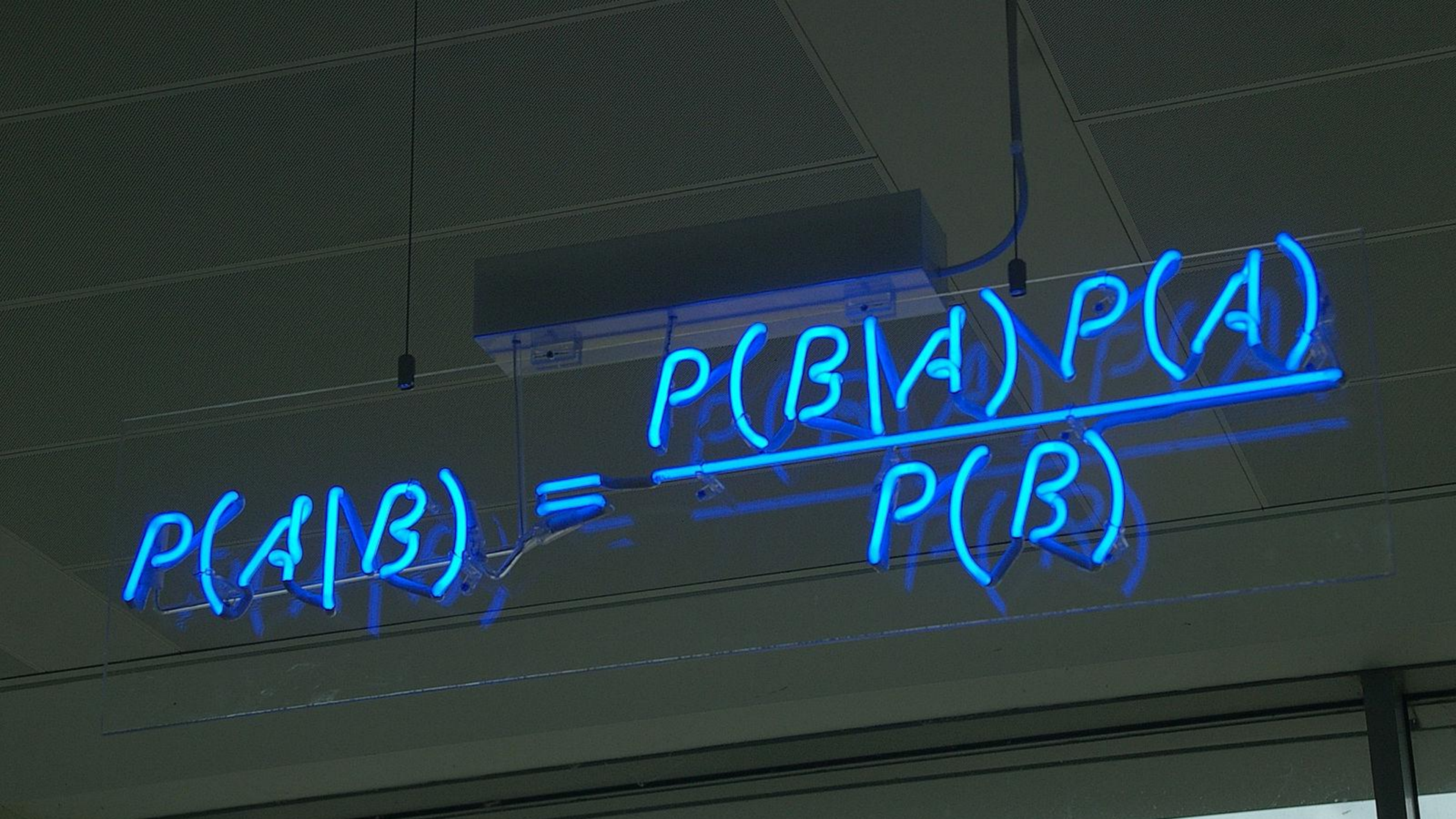
Frequentist

- **Data** is considered **random**
- **Model** parameters are **fixed**
- **Probabilities** are fundamentally related to frequencies of events

Bayesian

- **Data** is considered **fixed**
- **Model** parameters are "*random*" (conditioned to observations - sampled from distribution)
- **Probabilities** are fundamentally related to their own knowledge about an event

Source: https://github.com/fonnesbeck/intro_stat_modeling_2017


$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Bayes Inference 101

$$P(\theta|y) = \frac{P(y|\theta).P(\theta)}{P(y)}$$

Source: https://github.com/fonnesbeck/intro_stat_modeling_2017

Bayes Inference 101

$$\boxed{P(\theta|y)} = \frac{P(y|\theta).P(\theta)}{P(y)}$$

Posterior Probability → probability of model parameters given the data

Source: https://github.com/fonnesbeck/intro_stat_modeling_2017

Bayes Inference 101

Likelihood of observations → information on the observed data ~ proportional to likelihood in frequentist approach

$$P(\theta|y) = \frac{\boxed{P(y|\theta)}.P(\theta)}{P(y)}$$

Source: https://github.com/fonnesbeck/intro_stat_modeling_2017

Bayes Inference 101

Prior Probability → what is known about the model before observing the data

$$P(\theta|y) = \frac{P(y|\theta).\boxed{P(\theta)}}{P(y)}$$

Source:
https://github.com/fonnesbeck/intro_stat_modeling_2017

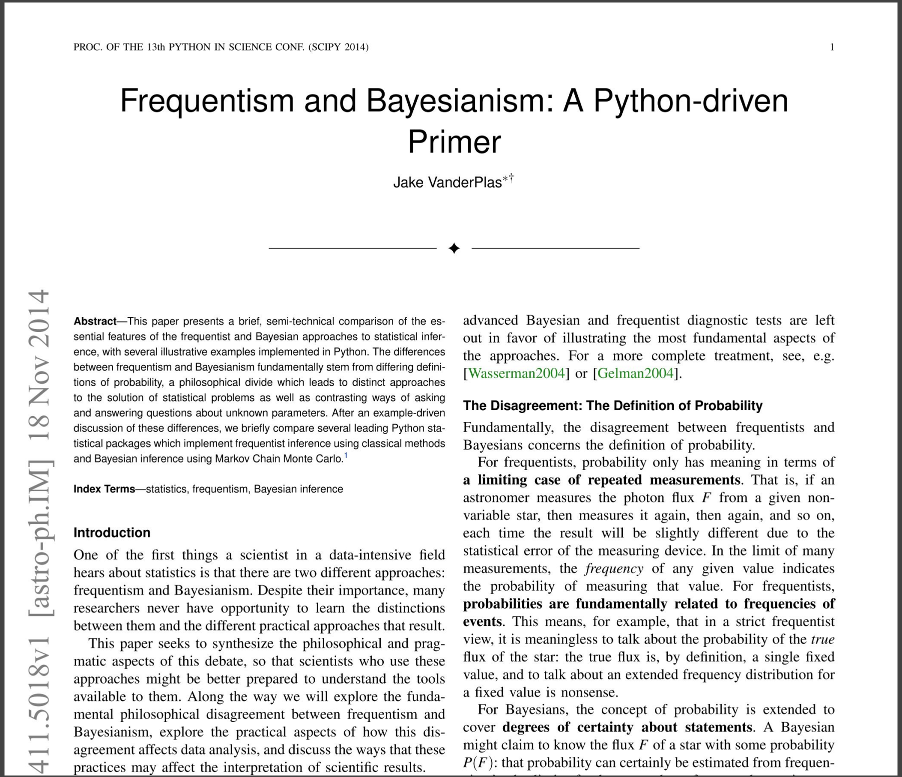
Bayes Inference 101

$$P(\theta|y) = \frac{P(y|\theta).P(\theta)}{\boxed{P(y)}}$$

Normalizing Constant → model evidence, usually not considered in bayesian inference

Source: https://github.com/fonnesbeck/intro_stat_modeling_2017

Further reading



PROC. OF THE 13th PYTHON IN SCIENCE CONF. (SCIPY 2014) 1

arXiv:1411.5018v1 [astro-ph.IM] 18 Nov 2014

Frequentism and Bayesianism: A Python-driven Primer

Jake VanderPlas*†

Abstract—This paper presents a brief, semi-technical comparison of the essential features of the frequentist and Bayesian approaches to statistical inference, with several illustrative examples implemented in Python. The differences between frequentism and Bayesianism fundamentally stem from differing definitions of probability, a philosophical divide which leads to distinct approaches to the solution of statistical problems as well as contrasting ways of asking and answering questions about unknown parameters. After an example-driven discussion of these differences, we briefly compare several leading Python statistical packages which implement frequentist inference using classical methods and Bayesian inference using Markov Chain Monte Carlo.[1]

Index Terms—statistics, frequentism, Bayesian inference

Introduction

One of the first things a scientist in a data-intensive field hears about statistics is that there are two different approaches: frequentism and Bayesianism. Despite their importance, many researchers never have opportunity to learn the distinctions between them and the different practical approaches that result.

This paper seeks to synthesize the philosophical and pragmatic aspects of this debate, so that scientists who use these approaches might be better prepared to understand the tools available to them. Along the way we will explore the fundamental philosophical disagreement between frequentism and Bayesianism, explore the practical aspects of how this disagreement affects data analysis, and discuss the ways that these practices may affect the interpretation of scientific results.

advanced Bayesian and frequentist diagnostic tests are left out in favor of illustrating the most fundamental aspects of the approaches. For a more complete treatment, see, e.g. [Wasserman2004] or [Gelman2004].

The Disagreement: The Definition of Probability

Fundamentally, the disagreement between frequentists and Bayesians concerns the definition of probability.

For frequentists, probability only has meaning in terms of **a limiting case of repeated measurements**. That is, if an astronomer measures the photon flux F from a given non-variable star, then measures it again, then again, and so on, each time the result will be slightly different due to the statistical error of the measuring device. In the limit of many measurements, the *frequency* of any given value indicates the probability of measuring that value. For frequentists, **probabilities are fundamentally related to frequencies of events**. This means, for example, that in a strict frequentist view, it is meaningless to talk about the probability of the *true* flux of the star: the true flux is, by definition, a single fixed value, and to talk about an extended frequency distribution for a fixed value is nonsense.

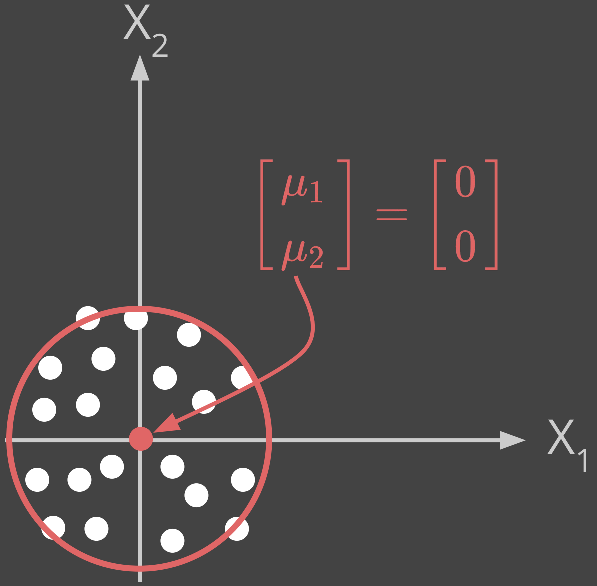
For Bayesians, the concept of probability is extended to cover **degrees of certainty about statements**. A Bayesian might claim to know the flux F of a star with some probability $P(F)$: that probability can certainly be estimated from frequen-

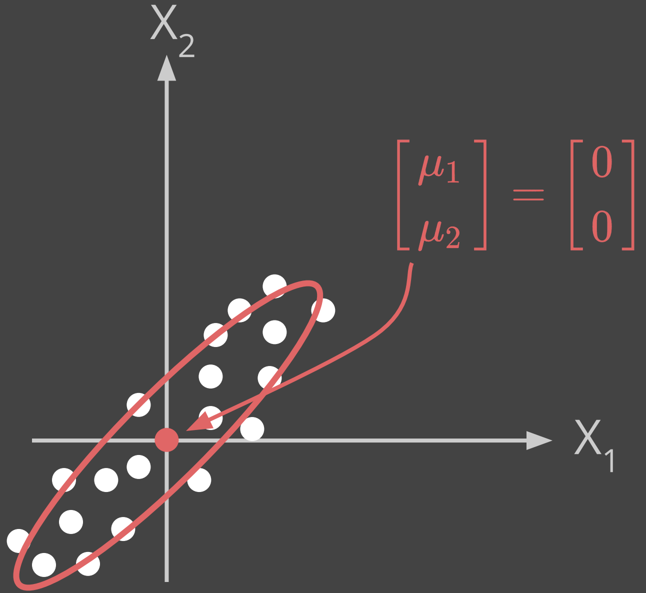
Source: Frequentism and Bayesianism: A Python-driven Primer (Jake VanderPlas, 2014)

Gaussian ***Processes***

Johann Carl Friedrich Gauss
(1777 - 1855)

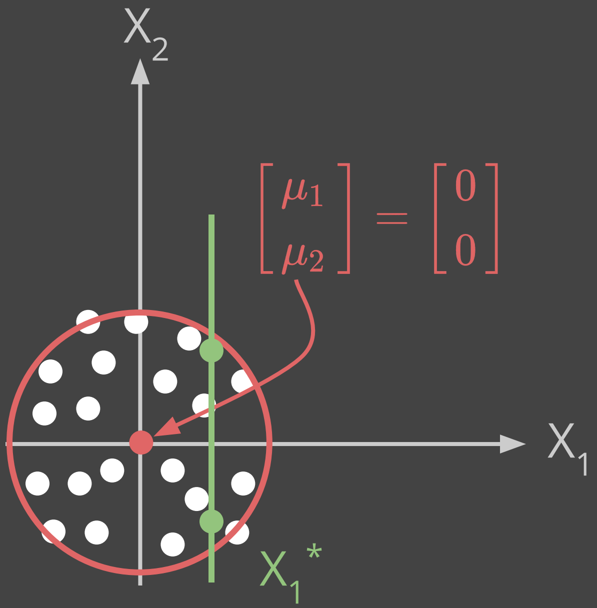
Gaussian Basics (1)



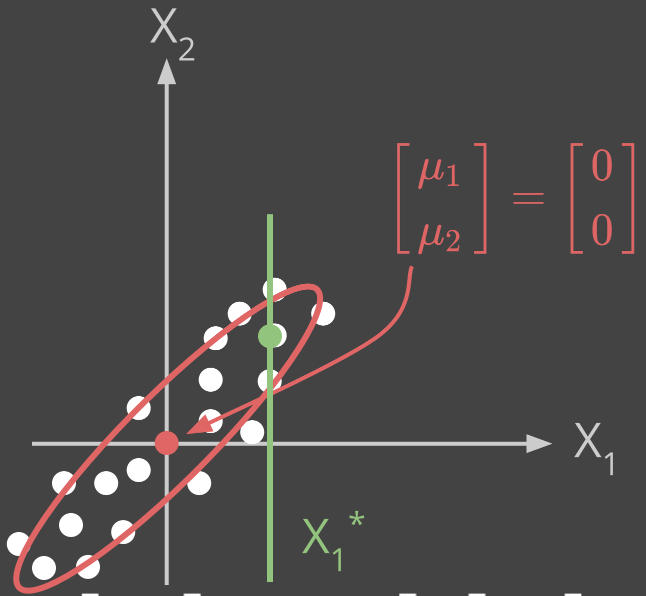


Source: YouTube - Machine learning - Introduction to Gaussian processes (Nando de Freitas, 2013)

Gaussian Basics (2)



$$\begin{bmatrix}x_1\\ x_2\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)$$

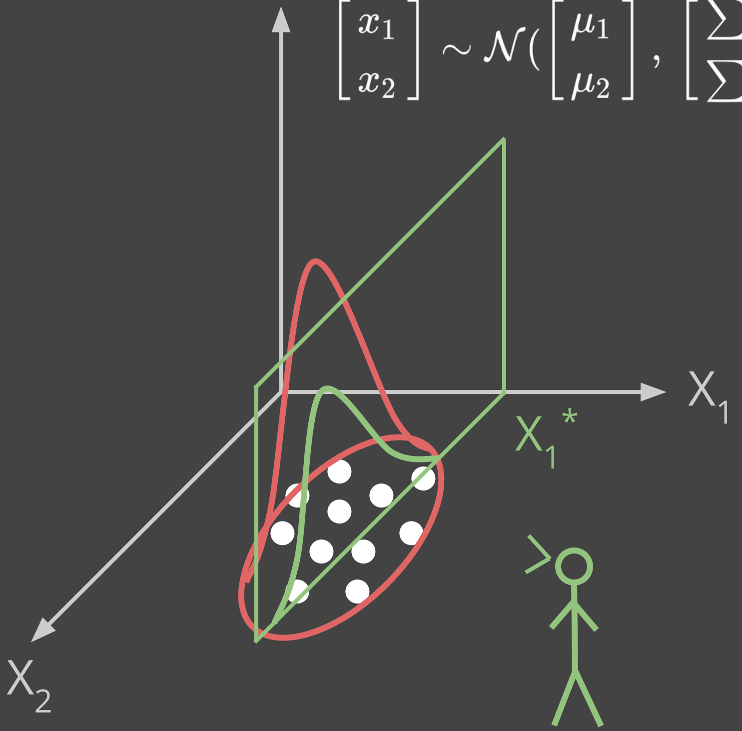


$$\begin{bmatrix}x_1\\ x_2\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}1 & 0.5\\ 0.5 & 1\end{bmatrix}\right)$$

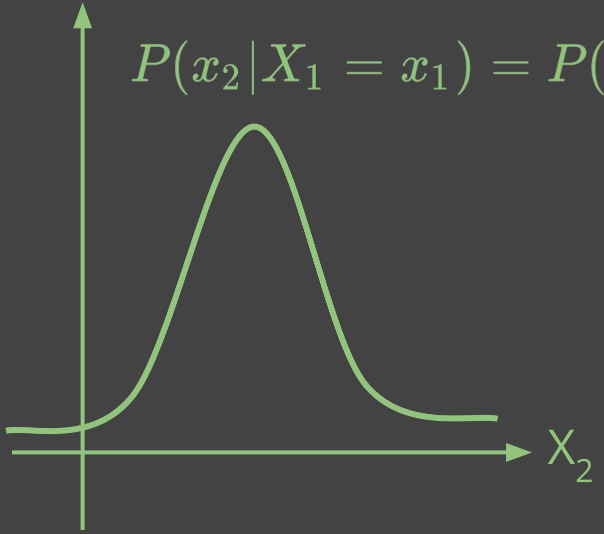
Source: YouTube - Machine learning - Introduction to Gaussian processes (Nando de Freitas, 2013)

Gaussian Basics (3) - from joint to conditional dist.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right)$$

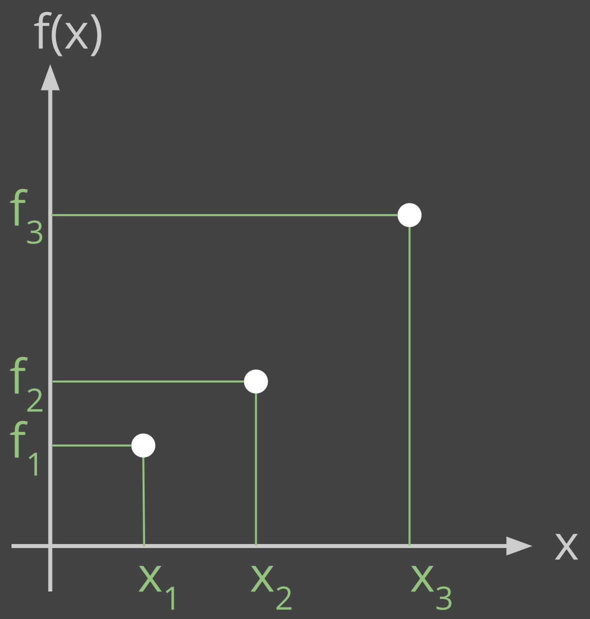


$$P(x_2 | X_1 = x_1) = P(x_2 | x_1)$$



Source: YouTube - Machine learning - Introduction to Gaussian processes (Nando de Freitas, 2013)

Gaussian Basics (4) - Multivariate Gaussian Dist.



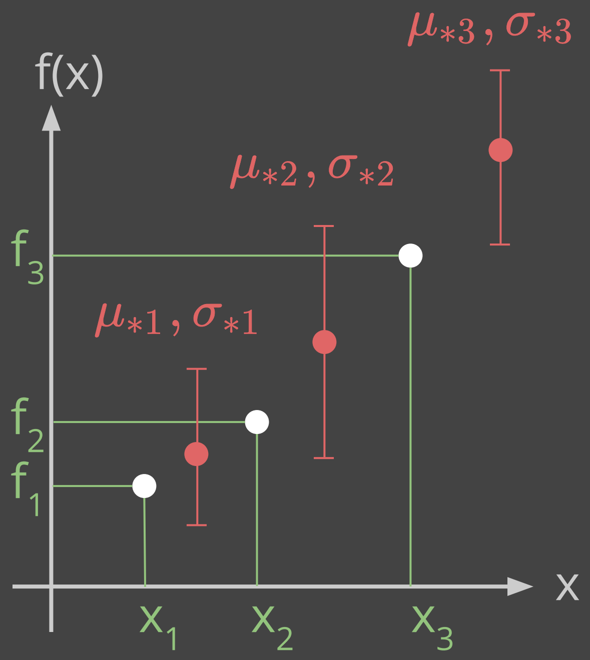
$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \right)$$

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.7 & 0.2 \\ 0.7 & 1 & 0.6 \\ 0.2 & 0.6 & 1 \end{bmatrix} \right)$$

$$K_{ij} = e^{-||X_i - X_j||^2} \begin{cases} 0, -||X_i - X_j||^2 \rightarrow \infty \\ 1, X_i = X_j \end{cases}$$

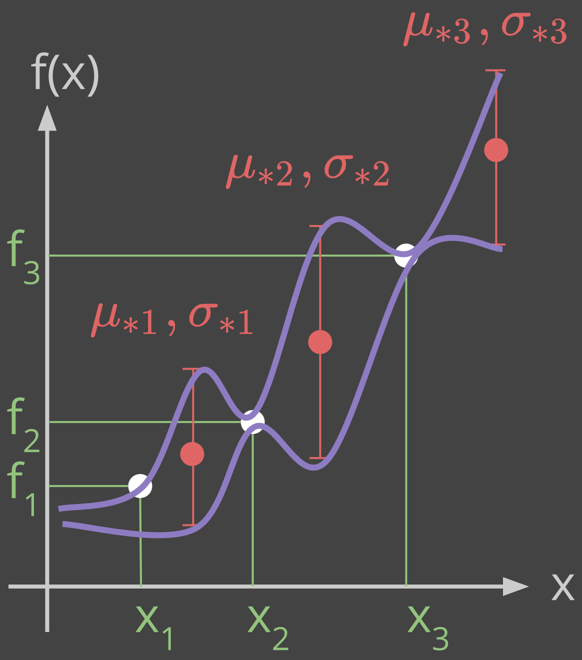
Source: YouTube - Machine learning - Introduction to Gaussian processes (Nando de Freitas, 2013)

Gaussian Basics (4) - Multivariate Gaussian Dist.



Source: YouTube - Machine learning - Introduction to Gaussian processes (Nando de Freitas, 2013)

Gaussian Basics (4) - Multivariate Gaussian Dist.



Where **have** data

+ confidence (lower uncertainty)

Where **don't have** data

- confidence (higher uncertainty)

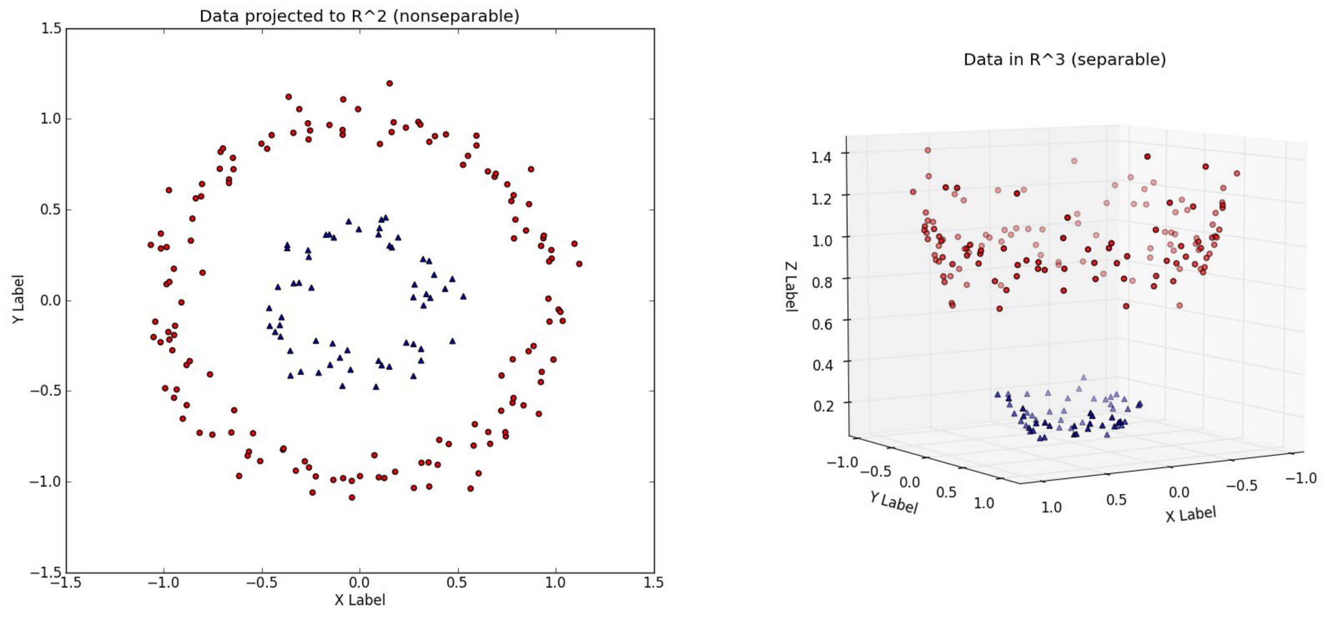
<u>Source</u>: YouTube - Machine learning - Introduction to Gaussian processes (Nando de Freitas, 2013)

Gaussian Processes (1)

- Generalization of multivariate gaussian distribution to infinitely many variables
- BAYESIAN NON-PARAMETRIC = # parameters grows w/ size of dataset = infinitely parametric
- GP are distribution over functions (not point estimates)
- Can be seen as bayesian version of SVM
- Uses kernels as SVM methods

Kernel Trick (recap)

Projects non-linear separable input into high-order linear separable space



Source: https://goo.gl/kxLepp

Gaussian Processes (2)

- GP = distribution over functions (or high dimensional vectors)
- GP is fully specified by:
  - Mean vector $\boldsymbol{\mu}$
  - Covariance matrix $\sum$

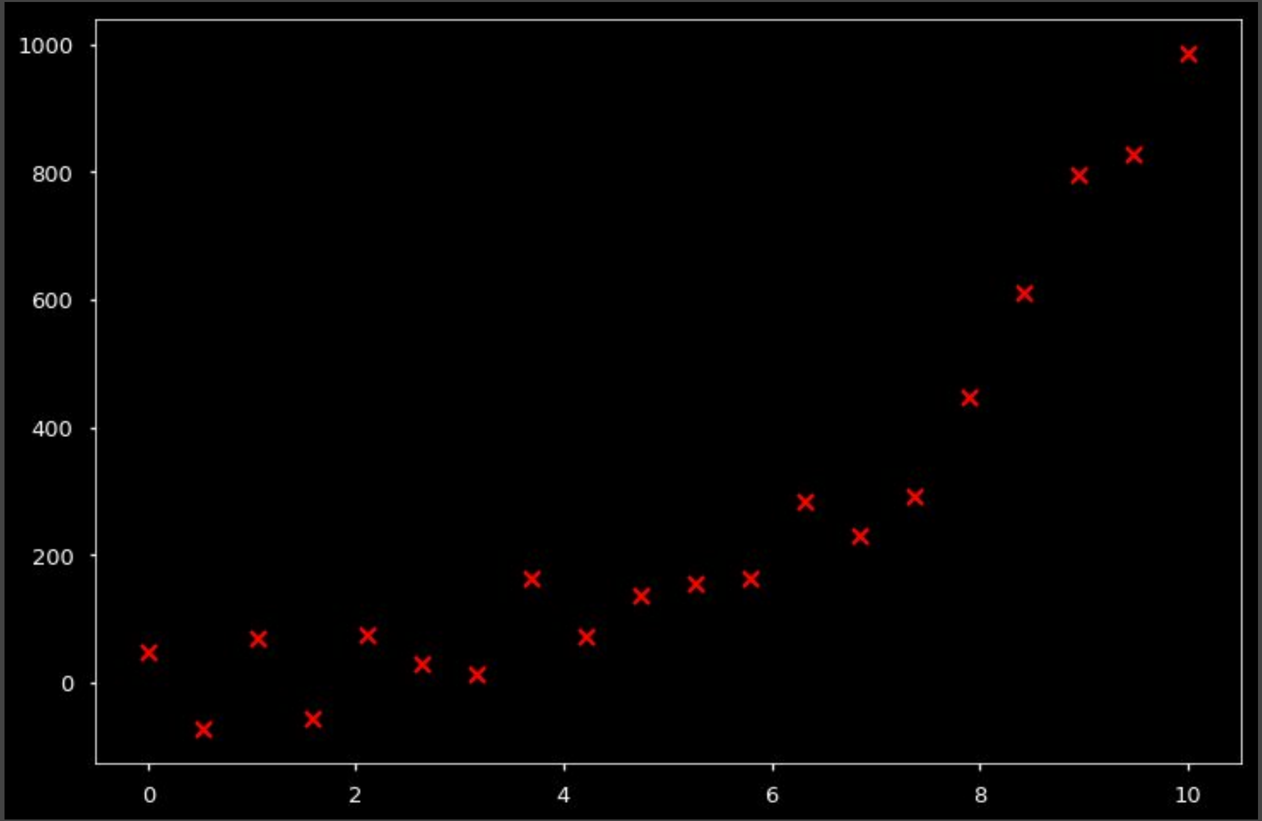
$$f(x) \sim GP(m(x), k(x, x'))$$

$$m(x) = \mathbb{E}[f(x)]$$

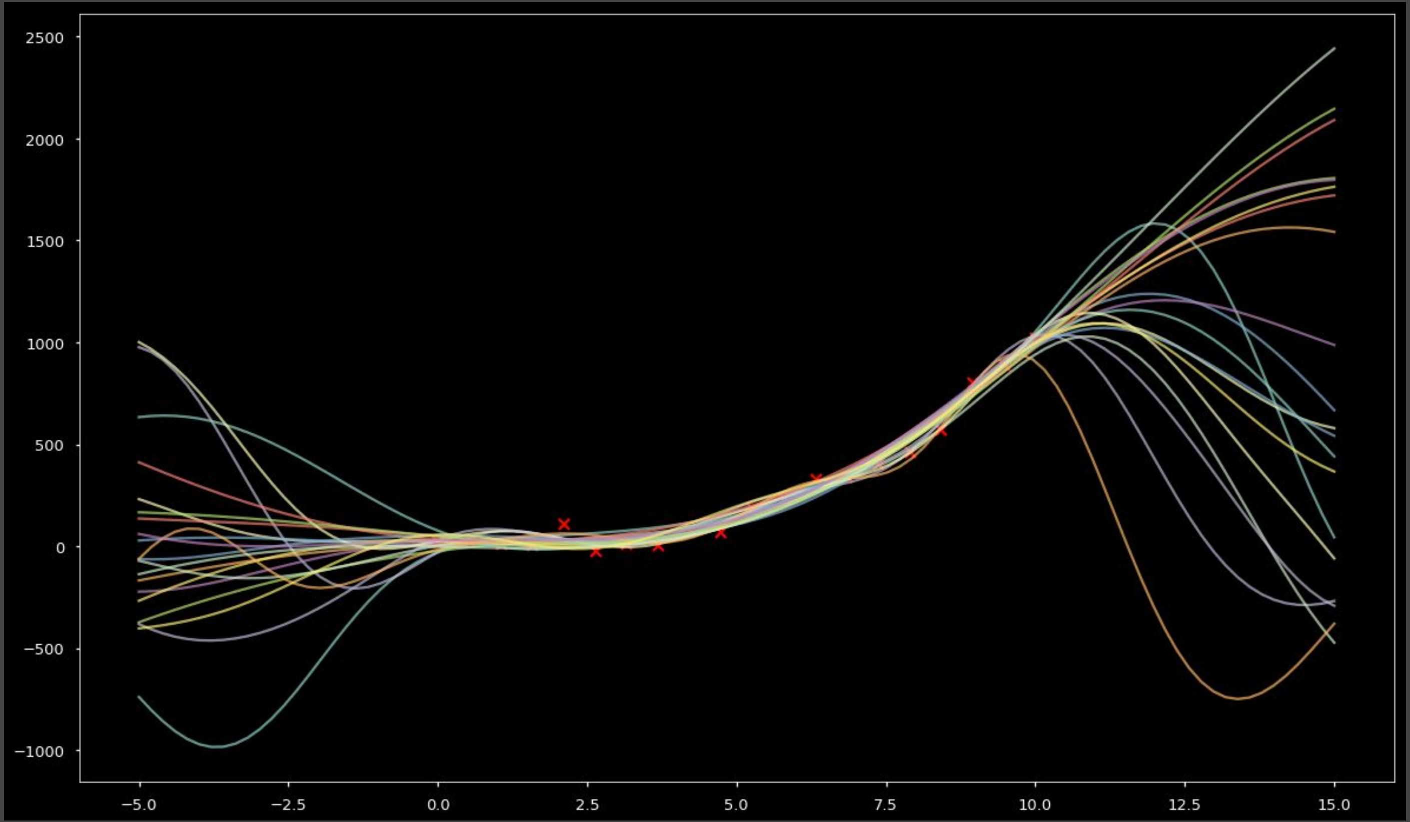
$$k(x, x') = \mathbb{E}[f(x) - m(x))(f(x') - m(x'))^T]$$

- Learning in GP = finding suitable properties for the covariance function

Gaussian Processes (3) - Regression

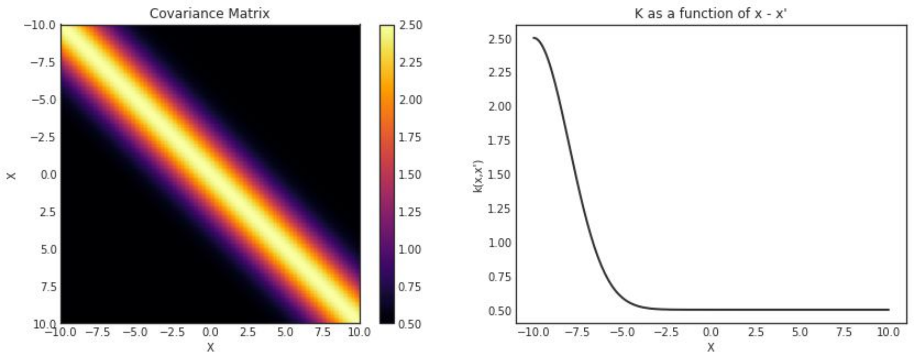


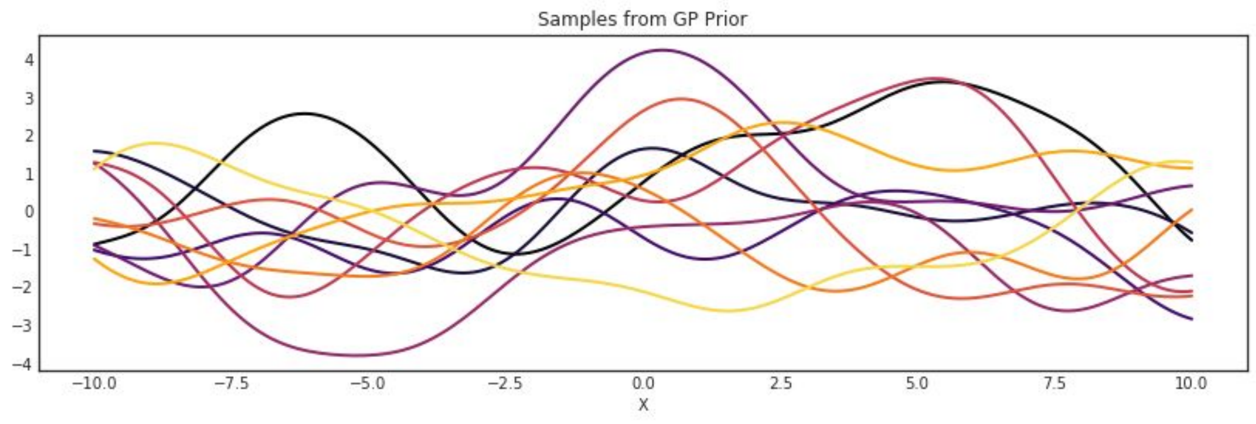
Gaussian Processes (4) - Regression



Covariance Functions - Squared Exponential

$$k(x, x') = \exp\left[-\frac{(x-x')^2}{2\ell^2}\right]$$

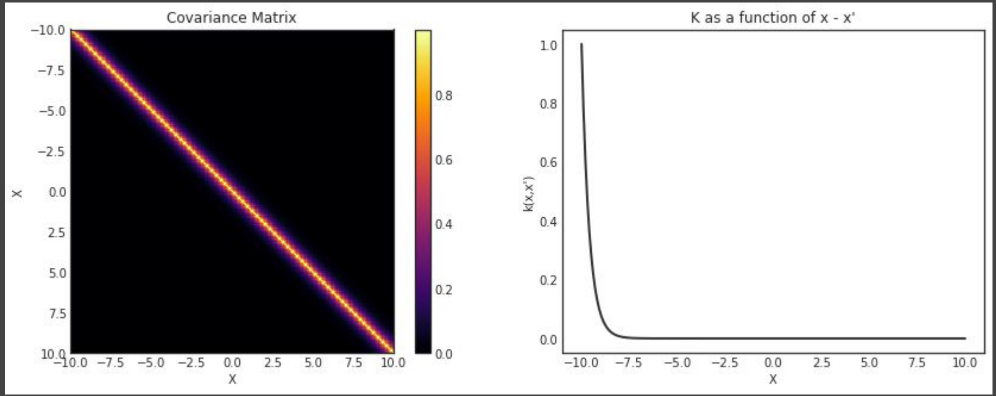


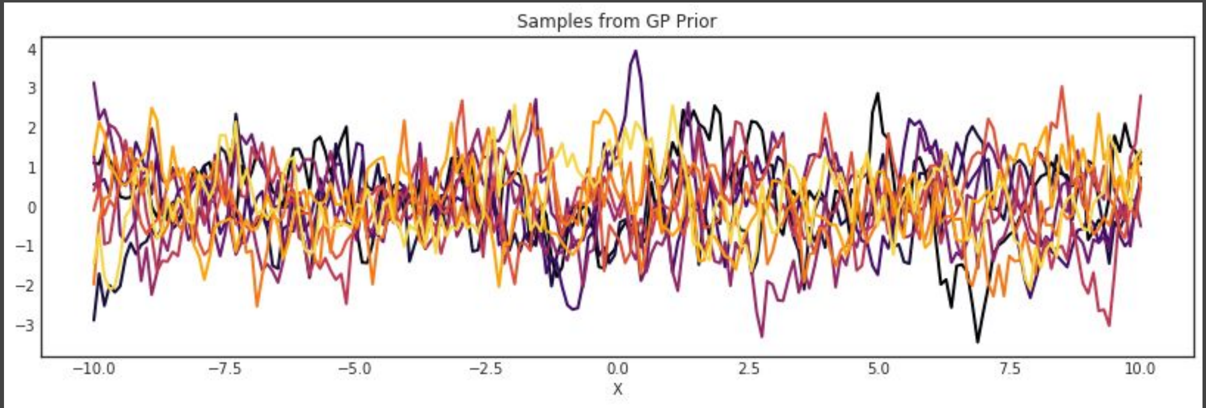


Source: https://docs.pymc.io/

Covariance Functions - Exponential

$$k(x, x') = \exp\left[-\frac{||x - x'||}{2\ell^2}\right]$$

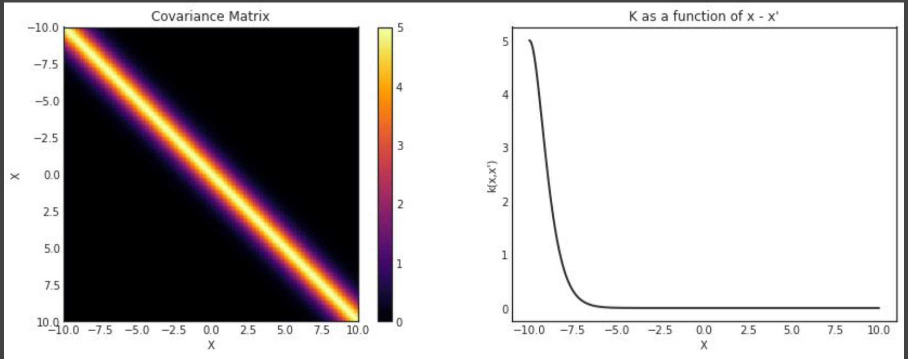


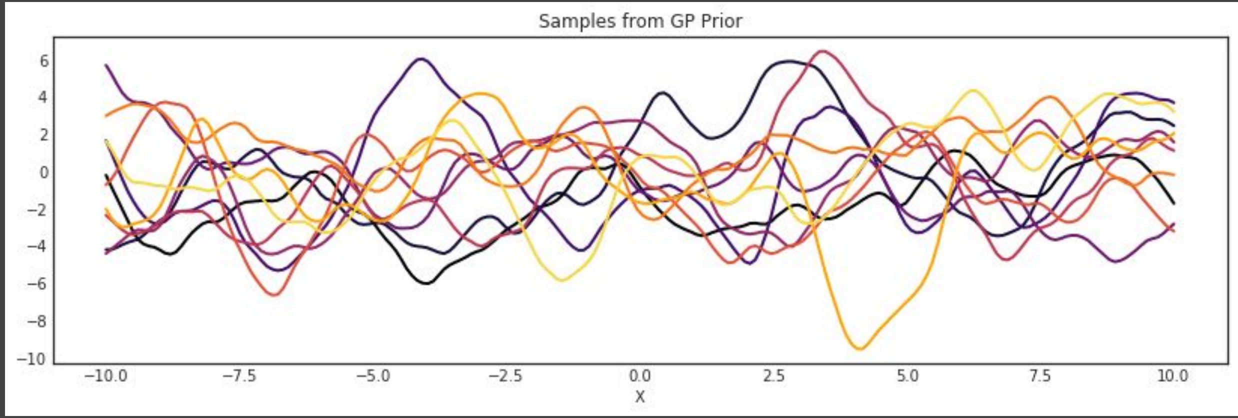


Source: https://docs.pymc.io/

Covariance Functions - Matern 5/2

$$k(x, x') = \left(1 + \frac{\sqrt{5(x-x')^2}}{\ell} + \frac{5(x-x')^2}{3\ell^2}\right) \exp\left[-\frac{\sqrt{5(x-x')^2}}{\ell}\right]$$

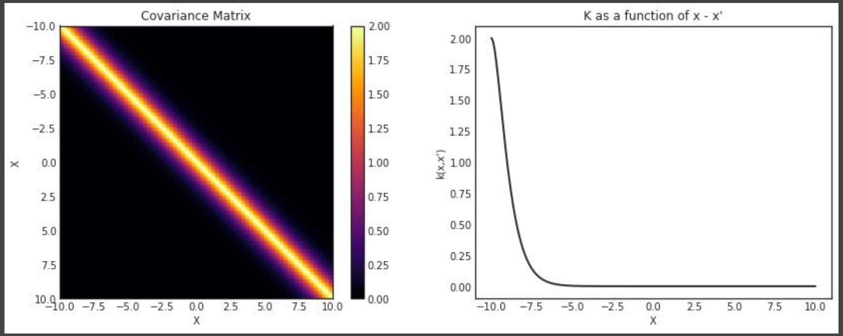


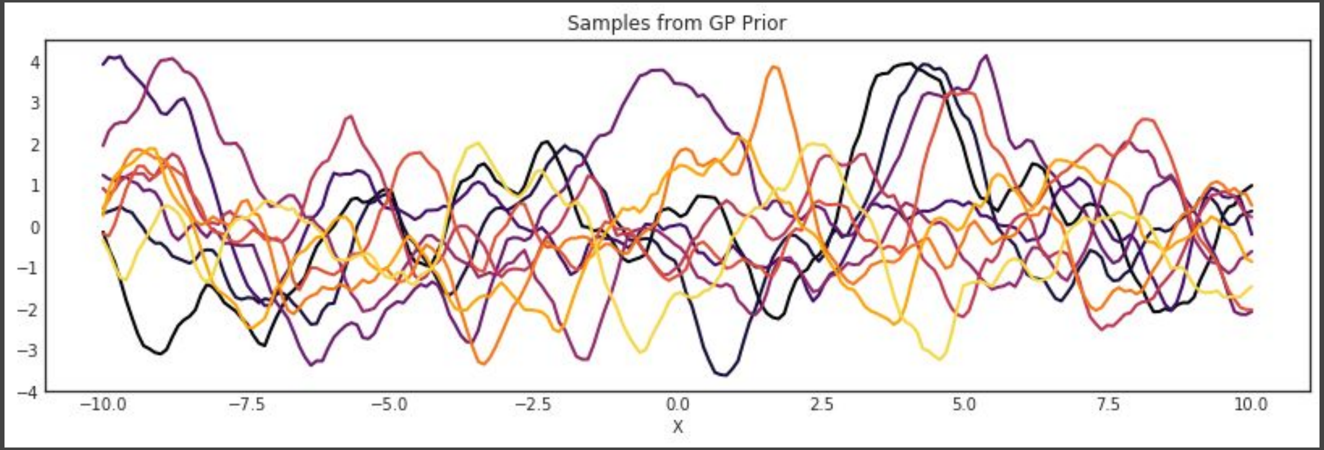


Source: https://docs.pymc.io/

Covariance Functions - Matern 3/2

$$k(x, x') = \left(1 + \frac{\sqrt{3(x-x')^2}}{\ell}\right) \exp\left[-\frac{\sqrt{3(x-x')^2}}{\ell}\right]$$

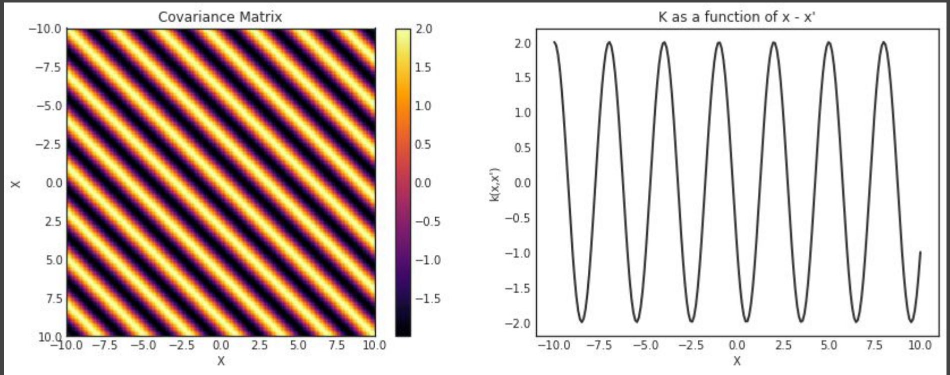


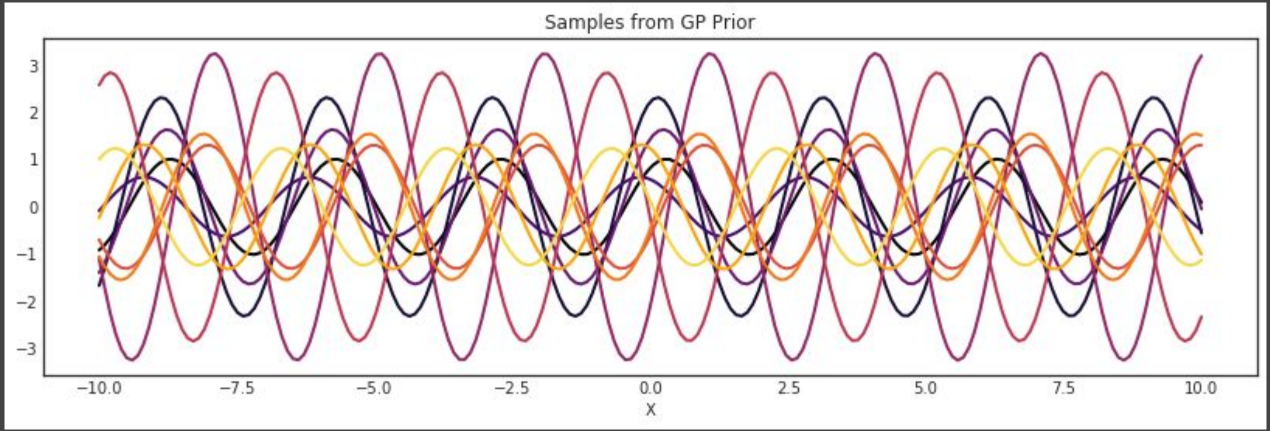


Source: https://docs.pymc.io/

Covariance Functions - Cosine

$$k(x, x') = \cos\left(2\pi \frac{||x - x'||}{\ell^2}\right)$$





Source: https://docs.pymc.io/

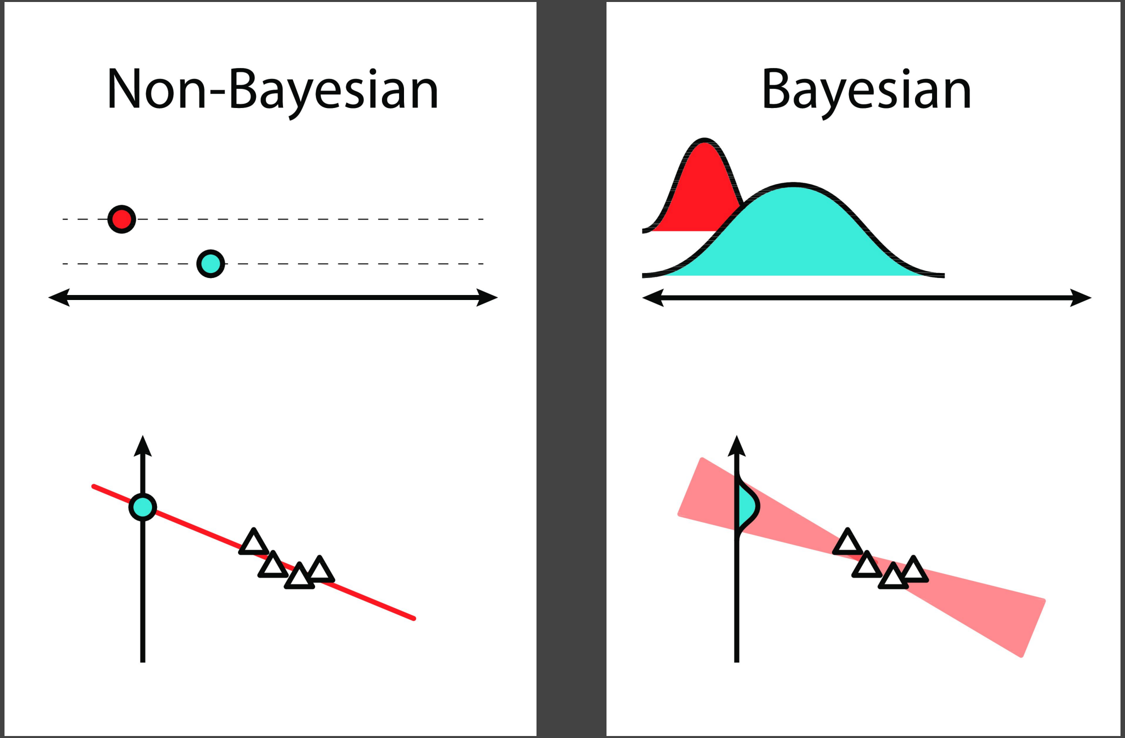
Demo time

Code:
https://github.com/rsilveira79/bayesian_notebooks/blob/master/uncertainty_notebooks/gaussian_process_PyMC3.ipynb

Bayes
meets
Deep
Learning

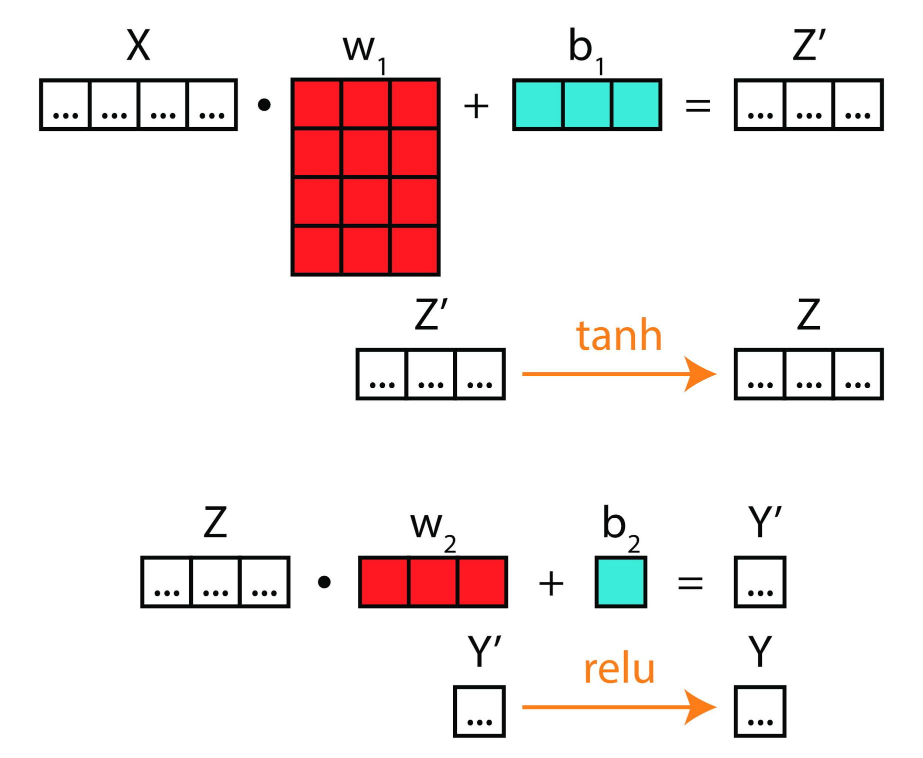
Or more logically, deep learning meets Bayesian inference

Intuition (1)



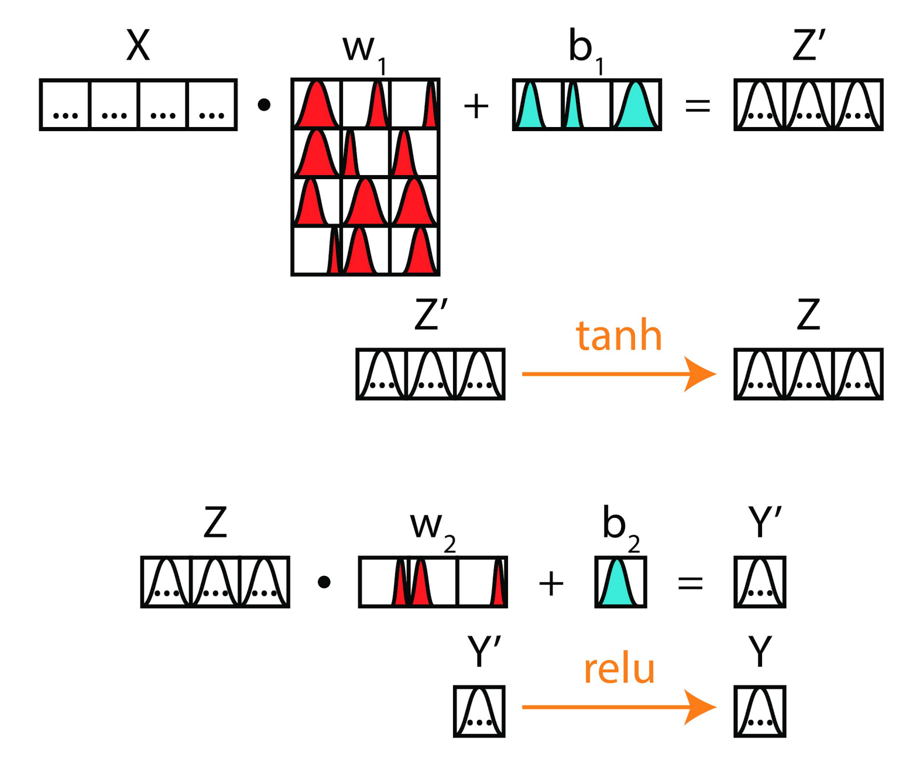
Source: https://ericmjl.github.io/bayesian-deep-learning-demystified/

Intuition (2) - From this ...



Source: https://ericmjl.github.io/bayesian-deep-learning-demystified/

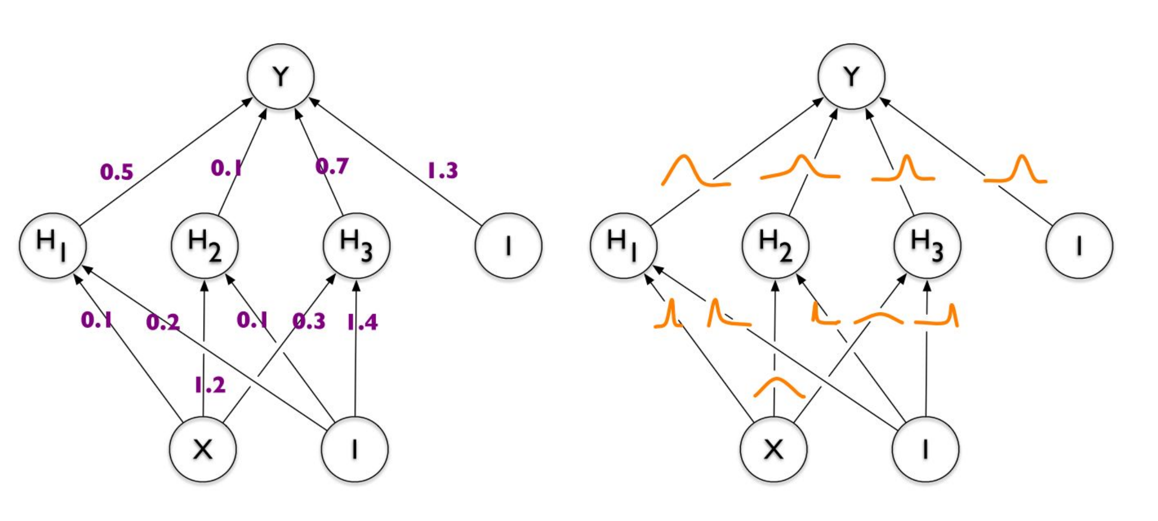
Intuition (3) - ... to this



Source: https://ericmjl.github.io/bayesian-deep-learning-demystified/

Bayes by Backprop (1)

- Each weight ***w*** is a distribution (instead of single point estimate)
- Backpropagation-compatible algorithm to compute distribution over ***w***



Source: Weight Uncertainty in Neural Networks (Blundell et al, 2015)

Bayes by Backprop (2)

$$w^{MAP} = argmax_w \log \boxed{P(w|D)}$$

Bayesian approach: data is fixed, update model beliefs ***(w)***

$$= argmax_w \log(D|w) + \log P(w)$$

mean gradients
(and weights)

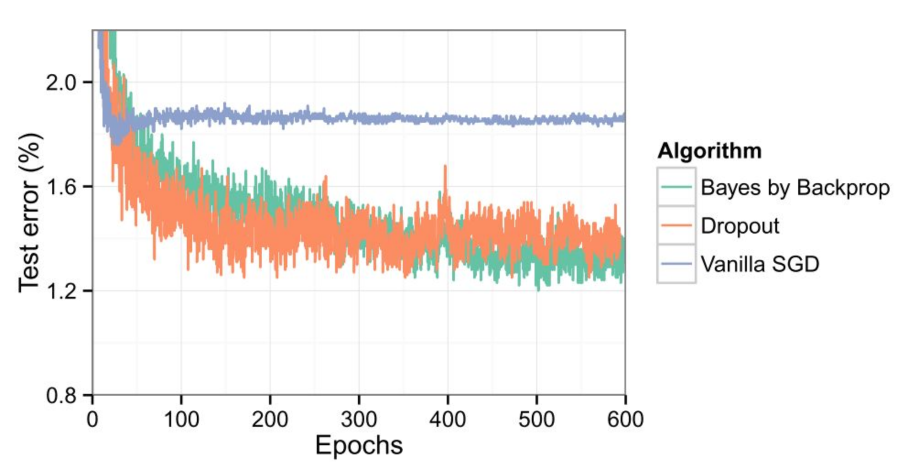
$$\boxed{\begin{aligned}\Delta_\mu &= \frac{\partial f}{\partial w} + \frac{\partial f}{\partial \mu} \\ \mu &\leftarrow \mu - \alpha . \Delta_\mu\end{aligned}}$$

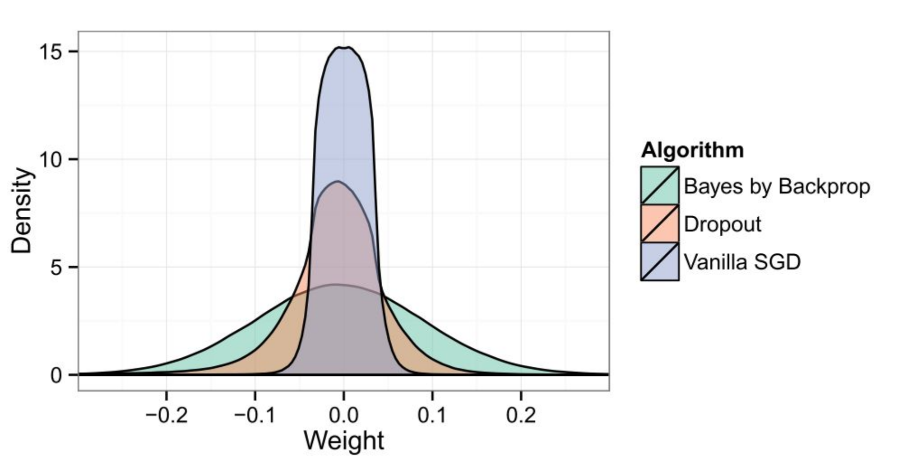
standard deviation gradients
(and weights)

$$\boxed{\begin{aligned}\Delta_\rho &= \frac{\partial f}{\partial w} . \frac{\epsilon}{1+\exp(-\rho)} + \frac{\partial f}{\partial \rho} \\ \rho &\leftarrow \rho - \alpha . \Delta_\rho\end{aligned}}$$

Source: Weight Uncertainty in Neural Networks
(Blundell et al, 2015)

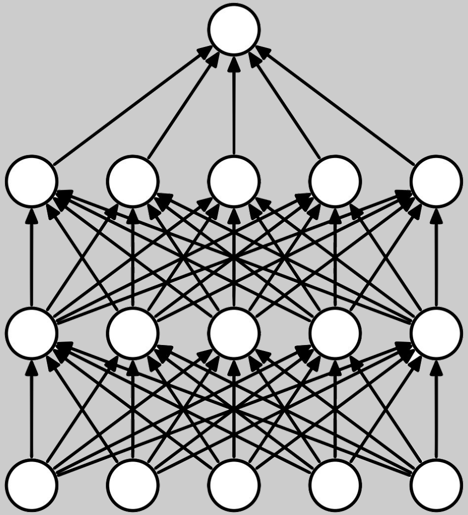
Bayes by Backprop (3) - MNIST



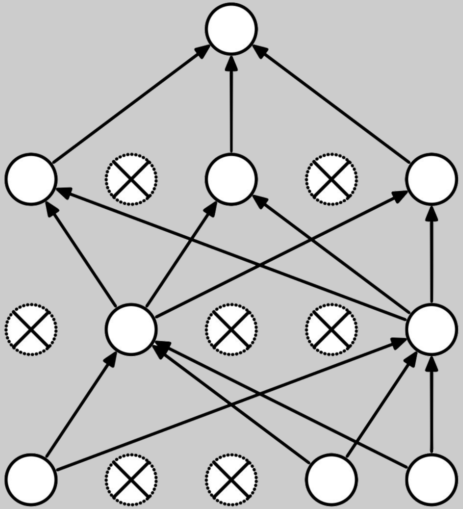


Source: Weight Uncertainty in Neural Networks (Blundell et al, 2015)

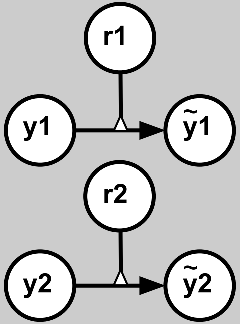
A quick note on *dropout*

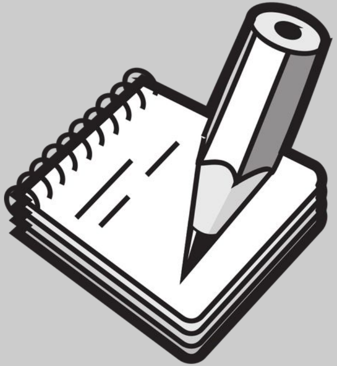


w/o *dropout*

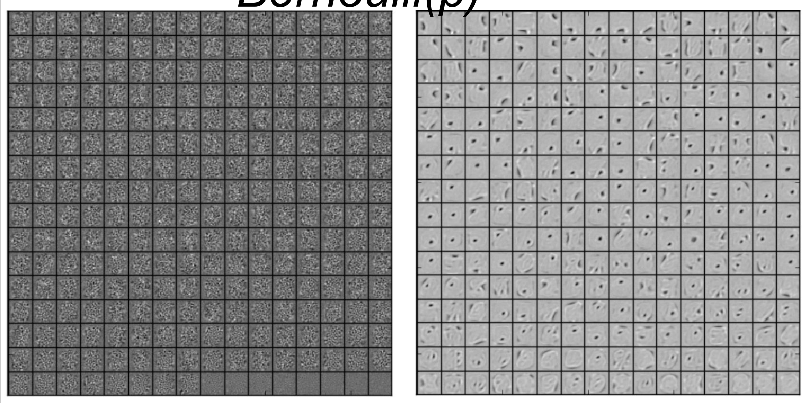


w/ *dropout*





$r_j^{(l)} = \mathit{Bernoulli}(p)$



w/o *dropout* ***w/*** *dropout*

KEY IDEA

1 NN w/ dropout = ensemble of 2^n thinned networks

Source: Dropout: A Simple Way to Prevent Neural Networks from Overfitting (Srivastava et al, 2014)

Monte Carlo Dropout (MC Dropout) (1)

- MC dropout is equivalent to performing T stochastic forward passes through the network and averaging the results (*model averaging*)

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

$$Var(y^*) \approx \tau^{-1} I_D + \frac{1}{T}\sum_{t=1}^{T} \hat{y}_t^*(x^*)^T \hat{y}_t^*(x^*) - \mathbb{E}(y^*)^T \mathbb{E}(y^*)$$

p → probability of units not being dropped

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

Source: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning (Gal et al, 2016), Uncertainty in Deep Learning - PhD Thesis (Gal, 2016)

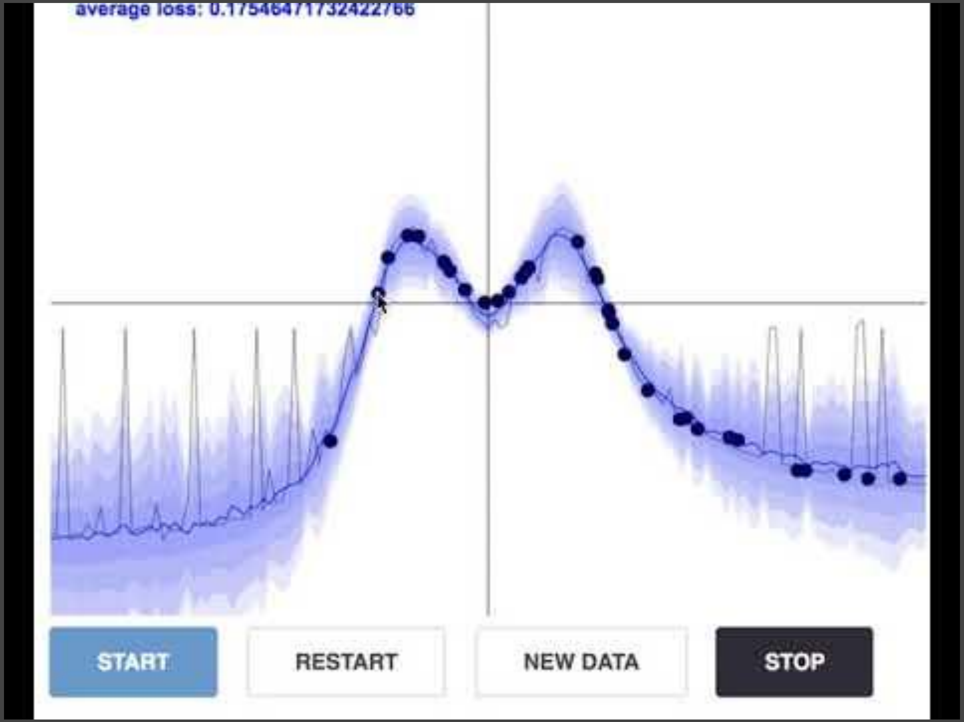
Monte Carlo Dropout (MC Dropout) (2)

- **In practice**, given a point *x*:
  - Drop units at **test time**;
  - Repeat **n** times (e.g. 10, 100, 200);
  - Look at the **mean** and **variance**;

```python
def uncertainity_estimate(X, model, iters, l2=0.005, range_fn=trange):
    outputs = np.hstack([model(X[:, np.newaxis]).data.numpy() for i in range_fn(iters)])
    y_mean = outputs.mean(axis=1)
    y_variance = outputs.var(axis=1)
    tau = l2 * (1-model.dropout_p) / (2*N*model.decay)
    y_variance += (1/tau)
    y_std = np.sqrt(y_variance) #+ (1/tau)
    return y_mean, y_std
```

Source: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning (Gal et al, 2016), Uncertainty in Deep Learning - PhD Thesis (Gal, 2016)

Monte Carlo Dropout (MC Dropout) (3)



Source: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning (Gal et al, 2016), Uncertainty in Deep Learning - PhD Thesis (Gal, 2016)

Demo time

Code:
https://github.com/rsilveira79/bayesian_notebooks/blob/master/uncertainty_notebooks/uncertainty_mc_dropout.ipynb

Deep Ensembles (1)

- Non-bayesian method
- High quality uncertainty predictions
    a. Not over-confident on out-of-distribution examples
- Model don't need to have dropout
- Basic receipt:
    a. Use proper scoring rule (loss function) - NLL for regression
    b. Use adversarial inputs (smooth predictions)
    c. Train *ensemble* of classifiers

Source: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan et al, 2017)

Deep Ensembles (2)

- Scoring Rules (Regression)

mean → no variance in MSE loss

$$MSE = \sum_{n=1}^{N} \left(y_n - \boxed{\mu(x_n)}\right)^2$$

mean

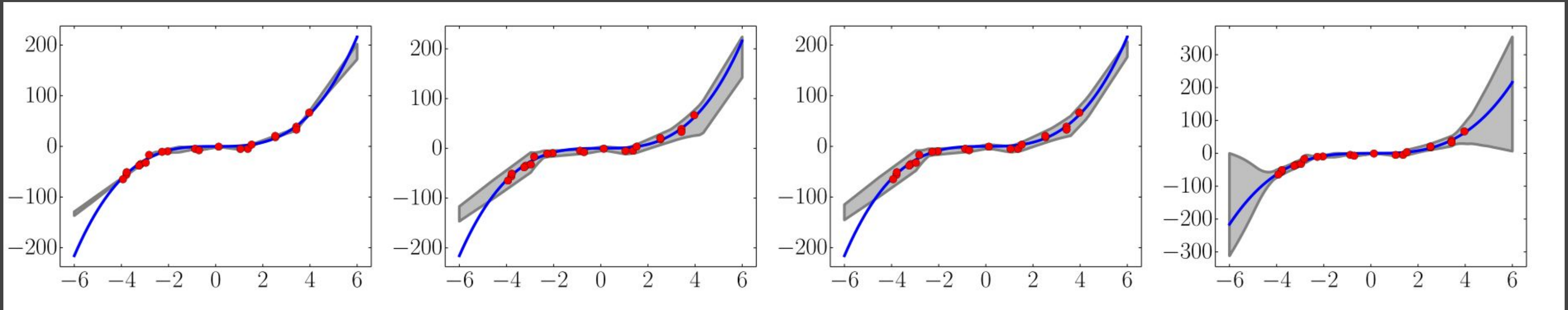
$$NLL = \frac{\log \boxed{\sigma_\theta^2(x)}}{2} + \frac{\left(y - \boxed{\mu_\theta(x)}\right)^2}{2\boxed{\sigma_\theta^2(x)}} + constant$$

variance

Source: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan et al, 2017)

Deep Ensembles (3)

MSE | **NLL** | **Adversarial training** | **Ensemble (M=5)**



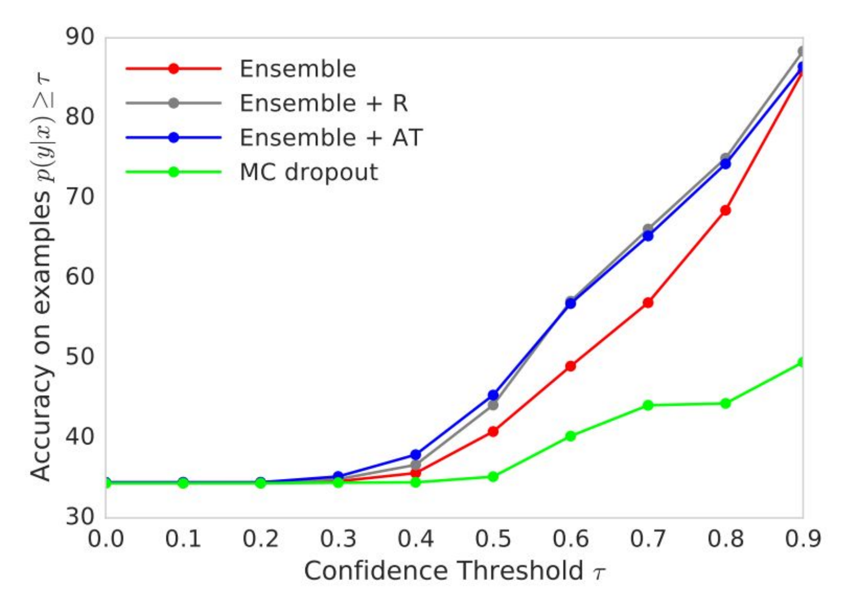
Source: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan et al, 2017)

Deep Ensembles (4)

Datasets	RMSE			NLL		
	PBP	MC-dropout	Deep Ensembles	PBP	MC-dropout	Deep Ensembles
Boston housing	**3.01 ± 0.18**	**2.97 ± 0.85**	**3.28 ± 1.00**	**2.57 ± 0.09**	**2.46 ± 0.25**	**2.41 ± 0.25**
Concrete	**5.67 ± 0.09**	**5.23 ± 0.53**	**6.03 ± 0.58**	**3.16 ± 0.02**	**3.04 ± 0.09**	**3.06 ± 0.18**
Energy	**1.80 ± 0.05**	**1.66 ± 0.19**	**2.09 ± 0.29**	2.04 ± 0.02	1.99 ± 0.09	**1.38 ± 0.22**
Kin8nm	0.10 ± 0.00	0.10 ± 0.00	**0.09 ± 0.00**	-0.90 ± 0.01	-0.95 ± 0.03	**-1.20 ± 0.02**
Naval propulsion plant	0.01 ± 0.00	0.01 ± 0.00	**0.00 ± 0.00**	-3.73 ± 0.01	-3.80 ± 0.05	**-5.63 ± 0.05**
Power plant	**4.12 ± 0.03**	**4.02 ± 0.18**	**4.11 ± 0.17**	2.84 ± 0.01	**2.80 ± 0.05**	**2.79 ± 0.04**
Protein	4.73 ± 0.01	**4.36 ± 0.04**	4.71 ± 0.06	2.97 ± 0.00	2.89 ± 0.01	**2.83 ± 0.02**
Wine	**0.64 ± 0.01**	**0.62 ± 0.04**	**0.64 ± 0.04**	0.97 ± 0.01	**0.93 ± 0.06**	**0.94 ± 0.12**
Yacht	**1.02 ± 0.05**	**1.11 ± 0.38**	**1.58 ± 0.48**	1.63 ± 0.02	1.55 ± 0.12	**1.18 ± 0.21**
Year Prediction MSD	8.88 ± NA	**8.85 ± NA**	8.89 ± NA	3.60 ± NA	3.59 ± NA	**3.35 ± NA**

Source: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan et al, 2017)

Deep Ensembles (5)

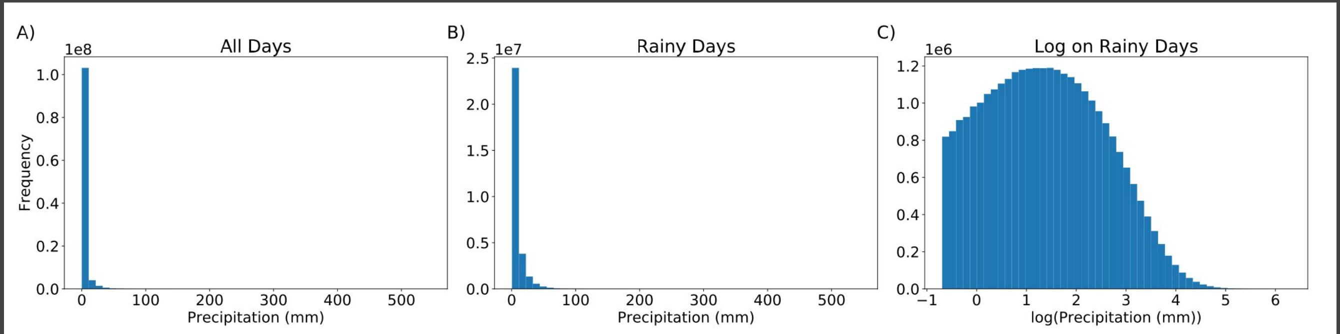


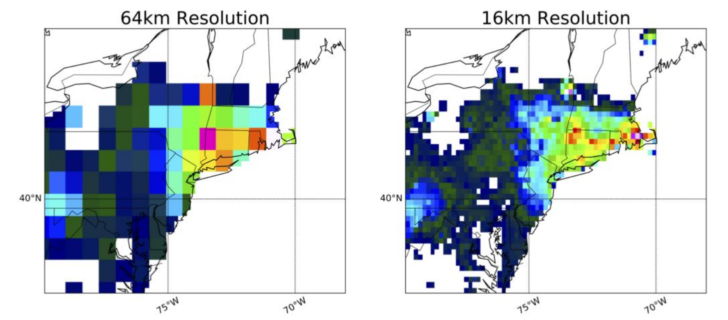
Source: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan et al, 2017)

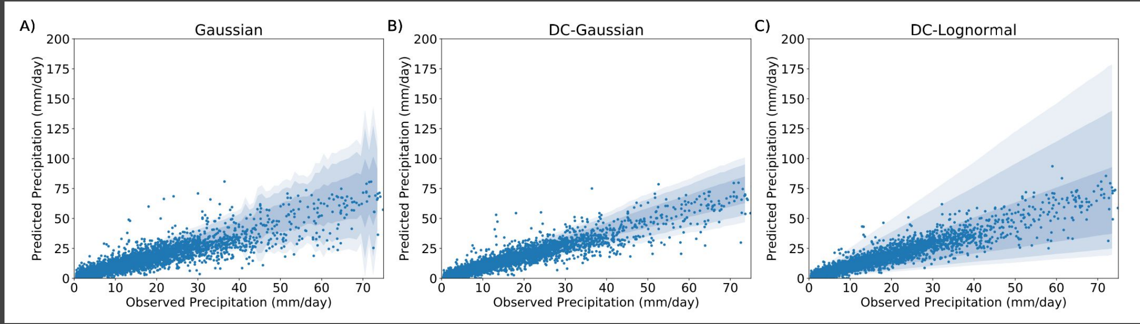
Demo time

Code:
https://github.com/rsilveira79/bayesian_notebooks/blob/master/uncertainty_notebooks/deep_ensemble_uncertainty_Pytorch.ipynb

Uncertainty in Discrete-Continuous Data

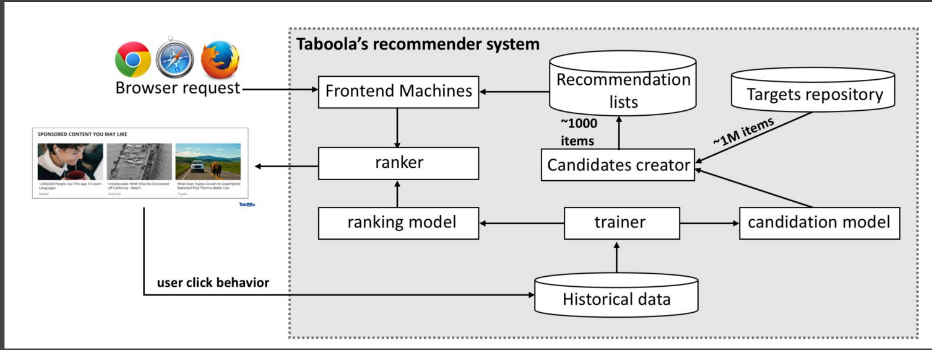


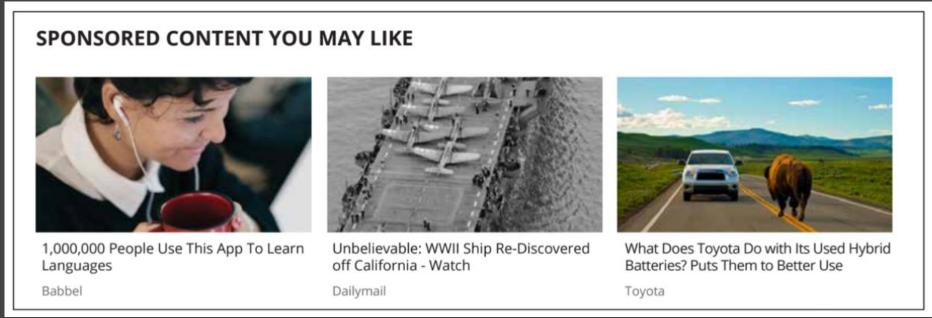




Source: Quantifying Uncertainty in Discrete-Continuous and Skewed Data with Bayesian Deep Learning (Blundell et al, 2015)

Uncertainty in Recommender Systems





Taboola

$$RPM = CTR * CPC * 1000$$

Model	REG	MDN	DDN
RPM lift	0%	1.2%	2.9%

Dataset	MDN	DDN	Improvement
D1	0.2681	0.25368	5.3%
D2	0.25046	0.24367	2.7 %

Source: Deep density networks and uncertainty in recommender systems (Zeldes et al, 2017)

Take home

Take #1

Have free time?

Study Bayesian Stats

(also if doesn't have free time)

Take #2

Probabilistic Programming
(e.g. GP) can be a powerful tool to
master - PyMC3, Pyro.ai, Stan
(specially for small datasets)

Take #3

For Bayesian Deep Learning

- stay tuned w/ latest developments (Cambridge, Deep Mind, Uber)
- always check uncertainty quality
- try different approaches

Probe further

Uncertainty in Deep Learning

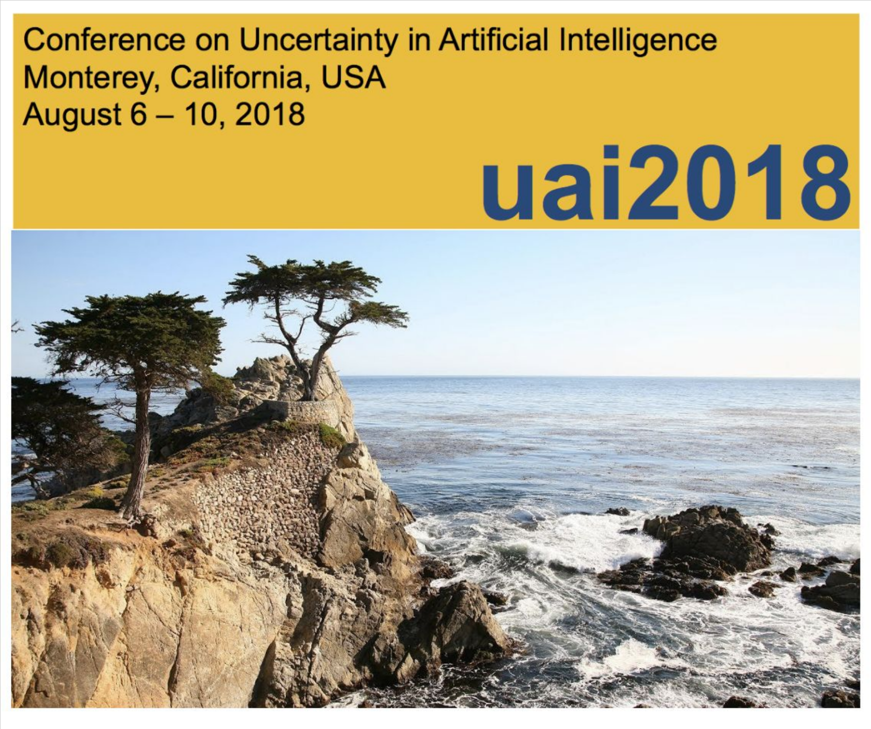
Home Schedule **Accepted Papers**

Accepted Papers

Note: Papers listed here do not constitute as proceedings for this workshop.

1. To Trust Or Not To Trust A Classifier *Heinrich Jiang, Been Kim, Maya Gupta*
2. Ambient Hidden Space of Generative Adversarial Networks *Xinhan Di, Pengqian Yu, Meng Tian*
3. Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion *Jacob Buckman, Danijar Hafner, George Tucker, Eugene*
4. Deep Contextual Multi-armed Bandits *Mark Collier, Hector Urdiales Llorens*
5. Understanding Deep Learning Performance through an Examination of Test Set Difficulty: A Psychometric Case Study *John P. Lalor, Hao W Yu*
6. Approximate Empirical Bayes for Deep Neural Networks *Han Zhao, Yao-Hung Hubert Tsai, Ruslan Salakhutdinov, Geoff Gordon*
7. Deep Matrix-variate Gaussian Processes *Young-Jin Park, Piyush M. Tagade, Han-Lim Choi*
8. Countdown Regression: Sharp and Calibrated Survival Predictions *Anand Avati, Tony Duan, Kenneth Jung, Nigam Shah, Andrew Ng*

Source: https://sites.google.com/view/udl2018/accepted-papers



Golden Sponsor

UBER

Microsoft

Disney Research

Google

BERG

Artificial Intelligence

Bronze Sponsor

The Alan Turing Institute

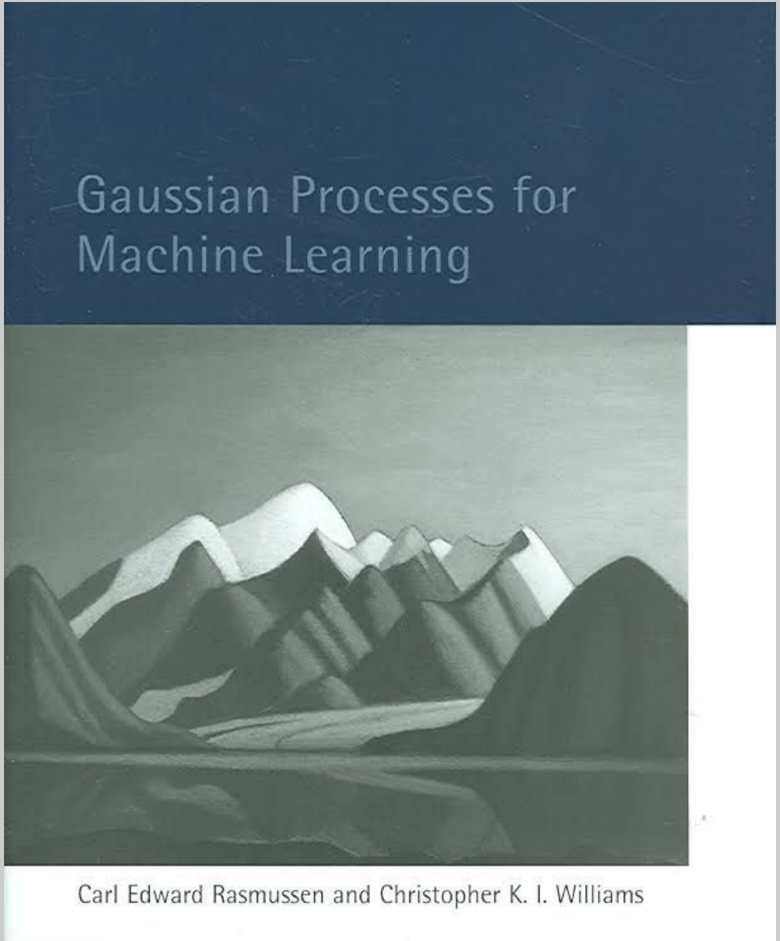
facebook

Startup Sponsor

Noodle.ai

ENTERPRISE ARTIFICIAL INTELLIGENCE

Source: http://auai.org/uai2018/index.php



Source: http://www.gaussianprocess.org/gpml/

UNIVERSITY OF CAMBRIDGE

Uncertainty in Deep Learning

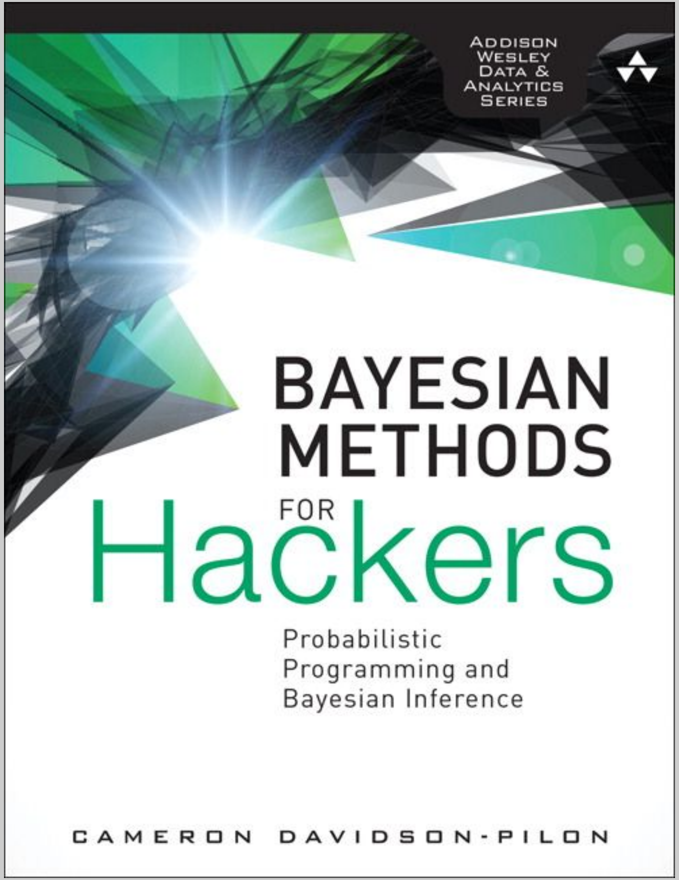
Yarin Gal

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Gonville and Caius College — September 2016

Source: http://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf



Source: https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers

Nice Repos

- https://github.com/fonnesbeck/bayesian_mixer_london_2017
- https://ericmjl.github.io/bayesian-deep-learning-demystified/
- https://github.com/yaringal

Nice blog posts

- https://blog.dominodatalab.com/fitting-gaussian-process-models-python/
- http://katbailey.github.io/post/gaussian-processes-for-dummies/
- http://mlg.eng.cam.ac.uk/yarin/blog_2248.html
- https://www.nitarshan.com/bayes-by-backprop/

THE DEVELOPER’S
CONFERENCE