

# Probabilistic machine learning: foundations and frontiers

**Zoubin Ghahramani**<sup>1,2</sup>

<sup>1</sup> University of Cambridge

<sup>2</sup> Uber AI Labs

`zoubin@eng.cam.ac.uk`

`http://mlg.eng.cam.ac.uk/zoubin/`

`https://www.uber.ai/`

**Cambridge Statistics Discussion Group**

**Cambridge 2017**

An exciting time for Machine Learning!

# APPLICATIONS OF MACHINE LEARNING

Speech and Language Technologies,  
automatic speech recognition,  
machine translation, question-answering,  
dialog systems



Computer Vision:  
Object, Face and Handwriting  
Recognition, Image Captioning



"He's in black when he's playing guitar."



"Sun protection is essential on a boat safety vest is working on boat."



"Some young girls are playing with lego toy."



"He is doing back flip on water slides."

Scientific Data Analysis  
(e.g. Bioinformatics, Astronomy)



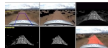
Recommender Systems



Self-driving cars

Autonomous driving

• ALVINN - Drives 70mph on highways



Financial Prediction and  
Automated Trading

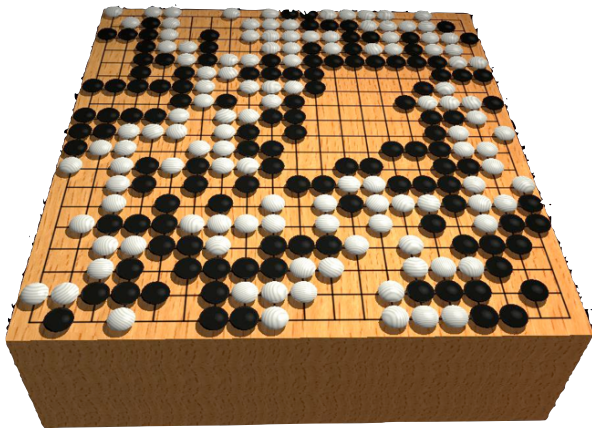


# Computer Games



Figure 1: Screen shots from five Atari 2600 Games: (*Left-to-right*) Pong, Breakout, Space Invaders, Seaquest, Beam Rider

# Computer Games





## BRAINS VS. ARTIFICIAL INTELLIGENCE

Be sure to tweet @WinBigRivers and @SCSatCMU using #BrainsvsAI



JANUARY 11-30 | 11AM-7PM

WE ARE LIPPING THE ANTE!  
120,000 HANDS NO-LIMIT HOLD 'EM

Each hand starts with each player having 200 big blinds.  
One big blind is \$100, and one small blind is \$50.

Hands Dealt: 115,756/120,000

**BRAINS : (\$1,560,189)**

**LIBRATUS : \$1,560,189**

**DONG KIM : (\$84,054)**

**JASON LES : (\$862,347)**

**LIBRATUS : \$84,054**

**LIBRATUS : \$862,347**

**JIMMY CHOU : (\$338,347)**

**DANIEL MCAULAY : (\$275,441)**

**LIBRATUS : \$338,347**

**LIBRATUS : \$275,441**

Parentheses indicate a negative number.



BRAINS VS. AI  
EVENT  
SCHEDULED  
FOR JANUARY 11-30  
11AM-7PM  
AT RIVERS CASINO

Carnegie Mellon University  
School of Computer Science

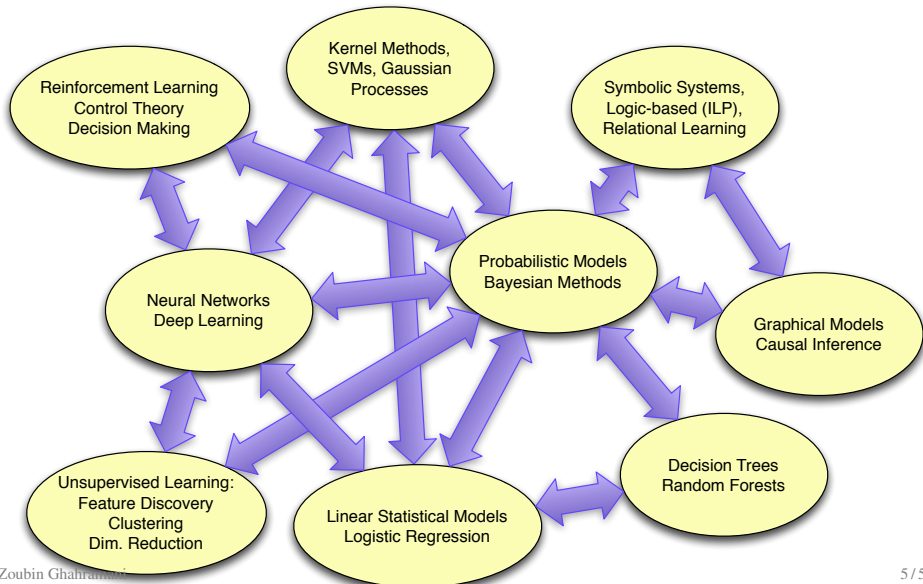


PITTSBURGH  
SUPERCOMPUTING  
CENTER



GREATPOINT TNG  
OPTIMIZED

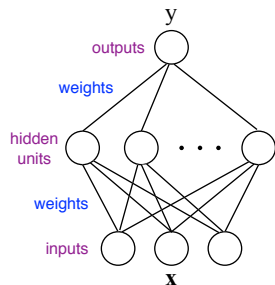
# ML: MANY METHODS WITH MANY LINKS TO OTHER FIELDS



# Neural networks and deep learning



# NEURAL NETWORKS



Neural networks are tunable nonlinear functions with many parameters.

Parameters  $\theta$  are weights of neural net.

Neural nets model  $p(y^{(n)}|\mathbf{x}^{(n)}, \theta)$  as a nonlinear function of  $\theta$  and  $\mathbf{x}$ , e.g.:

$$p(y^{(n)} = 1|\mathbf{x}^{(n)}, \theta) = \sigma\left(\sum_i \theta_i x_i^{(n)}\right)$$

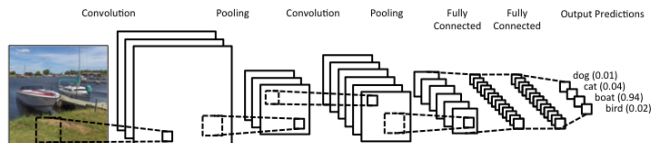
Multilayer neural networks model the overall function as a **composition of functions** (layers), e.g.:

$$y^{(n)} = \sum_j \theta_j^{(2)} \sigma\left(\sum_i \theta_{ji}^{(1)} x_i^{(n)}\right) + \epsilon^{(n)}$$

Usually trained to **maximise likelihood** (or penalised likelihood) using variants of **stochastic gradient descent (SGD)** optimisation.

**NN = nonlinear function + basic stats + basic optimisation**

# DEEP LEARNING



*Deep learning* systems are neural network models similar to those popular in the '80s and '90s, with:

- ▶ some architectural and algorithmic **innovations** (e.g. many layers, ReLUs, dropout, LSTMs)
- ▶ vastly larger **data** sets (web-scale)
- ▶ vastly larger-scale **compute** resources (GPU, cloud)
- ▶ much better **software** tools (Theano, Torch, TensorFlow)
- ▶ vastly increased industry **investment** and **media hype**

figure from <http://www.andreykurenkov.com/>

# LIMITATIONS OF DEEP LEARNING

Neural networks and deep learning systems give amazing performance on many benchmark tasks but they are generally:

- ▶ very **data hungry** (e.g. often millions of examples)
- ▶ very **compute-intensive** to train and deploy (cloud GPU resources)
- ▶ poor at representing **uncertainty**
- ▶ **easily fooled** by adversarial examples
- ▶ **finicky to optimise**: non-convex + choice of architecture, learning procedure, initialisation, etc, require expert knowledge and experimentation
- ▶ uninterpretable **black-boxes**, lacking in transparency, difficult to trust

# Beyond deep learning

# TOOLBOX VS MODELLING VIEWS OF MACHINE LEARNING

- ▶ **Machine Learning is a toolbox of methods for processing data:** feed the data into one of many possible methods; choose methods that have good theoretical or empirical performance; make predictions and decisions
- ▶ **Machine Learning is the science of learning models from data:** define a space of possible models; learn the parameters and structure of the models from data; make predictions and decisions

# MACHINE LEARNING AS PROBABILISTIC MODELLING

- ▶ A *model* describes data that one could observe from a system
- ▶ If we use the mathematics of *probability theory* to express all forms of uncertainty and noise associated with our model...
- ▶ ...then *inverse probability* (i.e. Bayes rule) allows us to infer unknown quantities, adapt our models, make predictions and learn from data.

# BAYES RULE

$$P(\text{hypothesis}|\text{data}) = \frac{P(\text{hypothesis})P(\text{data}|\text{hypothesis})}{\sum_{\mathbf{h}} P(\mathbf{h})P(\text{data}|\mathbf{h})}$$

- ▶ Bayes rule tells us how to do inference about **hypotheses (uncertain quantities)** from **data (measured quantities)**.
- ▶ Learning and prediction can be seen as forms of inference.



Reverend Thomas Bayes (1702-1761)

# ONE SLIDE ON BAYESIAN MACHINE LEARNING

*Everything follows from two simple rules:*

**Sum rule:**  $P(x) = \sum_y P(x, y)$

**Product rule:**  $P(x, y) = P(x)P(y|x)$

## Learning:

$$P(\theta|\mathcal{D}, m) = \frac{P(\mathcal{D}|\theta, m)P(\theta|m)}{P(\mathcal{D}|m)}$$

$P(\mathcal{D}|\theta, m)$  likelihood of parameters  $\theta$  in model  $m$   
 $P(\theta|m)$  prior probability of  $\theta$   
 $P(\theta|\mathcal{D}, m)$  posterior of  $\theta$  given data  $\mathcal{D}$

## Prediction:

$$P(x|\mathcal{D}, m) = \int P(x|\theta, \mathcal{D}, m)P(\theta|\mathcal{D}, m)d\theta$$

## Model Comparison:

$$P(m|\mathcal{D}) = \frac{P(\mathcal{D}|m)P(m)}{P(\mathcal{D})}$$



# WHY SHOULD WE CARE?

**Calibrated model and prediction uncertainty:** getting systems that know when they don't know.

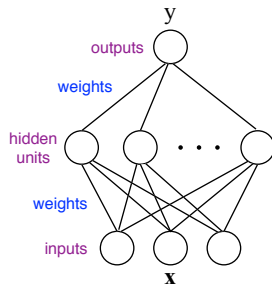
Automatic model **complexity control and structure learning**  
(Bayesian Occam's Razor)

# WHAT DO I MEAN BY BEING BAYESIAN?

Let's return to the example of neural networks / deep learning:

Dealing with all sources of **parameter uncertainty**

Also potentially dealing with **structure uncertainty**



Feedforward neural nets model  $p(y^{(n)} | \mathbf{x}^{(n)}, \theta)$

**Parameters**  $\theta$  are weights of neural net.

**Structure** is the choice of architecture, number of hidden units and layers, choice of activation functions, etc.

# BAYESIAN DEEP LEARNING

Bayesian deep learning can be implemented in many ways:

- ▶ Laplace approximations (MacKay, 1992)
- ▶ variational approximations (Hinton and van Camp, 1993; Graves, 2011)
- ▶ MCMC (Neal, 1993)
- ▶ Stochastic gradient Langevin dynamics (SGLD; Welling and Teh, 2011)
- ▶ Probabilistic back-propagation (Hernandez-Lobato et al, 2015, 2016)
- ▶ Dropout as Bayesian averaging (Gal and Ghahramani, 2015, 2016)

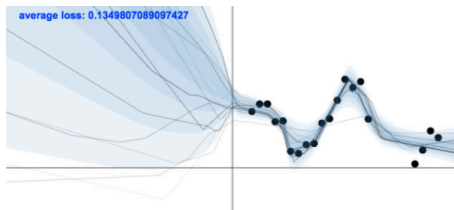


Figure from Yarin Gal's thesis "Uncertainty in Deep Learning" (2016)

→ [NIPS 2016 workshop on Bayesian Deep Learning](#)

When do we need probabilities?

# WHEN IS THE PROBABILISTIC APPROACH ESSENTIAL?

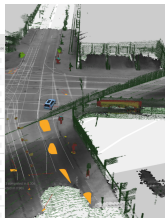
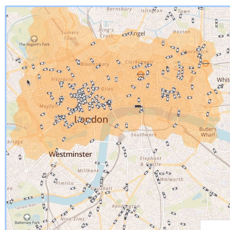
Many aspects of learning and intelligence depend crucially on the careful probabilistic representation of *uncertainty*:

- ▶ Forecasting
- ▶ Decision making
- ▶ Learning from limited, noisy, and missing data
- ▶ Learning complex personalised models
- ▶ Data compression
- ▶ Automating modelling, discovery, and experiment design

# WHY DOES UBER CARE?

Many aspects of learning and intelligence depend crucially on the careful probabilistic representation of *uncertainty*:

- ▶ Forecasting, Decision making, Learning from limited, noisy, and missing data, Learning complex personalised models, Data compression, Automating modelling, discovery, and experiment design



supply (driver) and demand (rider) prediction, ETA prediction, pricing, business intelligence, modelling traffic and cities, mapping, self-driving cars, customer service, UberEATs personalization...





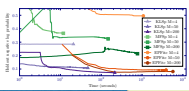
Automatic Statistician

```

stationary = 1, 0, 1, 0 # Stationary parameters.
initial = [Categorical(0.5, 0.5), Categorical(0.5, 0.5)] # Fresh doctor of student?
times = [Categorical(0.5, 0.5, 0.5), Categorical(0.5, 0.5, 0.5)]
Categorical(0.5, 0.5, 0.5) # Fresh doctor for each station.
data = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]

@model begin
  @plate 2 {
    station = Dirac(0.5, length(times))
    @observed{times} ~ Bernoulli(station)
  }
  for k = 1:length(times)
    @observed{times[k]} ~ Bernoulli(station[k], 0.5)
  end
end
  
```

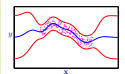
Probabilistic Programming



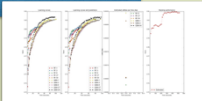
Large-scale Inference

Automating Machine Learning

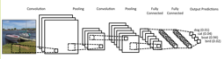
Bayesian Nonparametrics



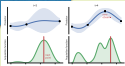
Computational Resource Allocation



Bayesian Deep Learning



Bayesian Optimisation





# Automating Inference: Probabilistic Programming

# PROBABILISTIC PROGRAMMING

**Problem:** Probabilistic model development and the derivation of inference algorithms is time-consuming and error-prone.

# PROBABILISTIC PROGRAMMING

**Problem:** Probabilistic model development and the derivation of inference algorithms is time-consuming and error-prone.

**Solution:**

- ▶ Develop **Probabilistic Programming Languages** for expressing probabilistic models as computer programs that generate data (i.e. simulators).
- ▶ Derive **Universal Inference Engines** for these languages that do inference over program traces given observed data (Bayes rule on computer programs).

# PROBABILISTIC PROGRAMMING

**Problem:** Probabilistic model development and the derivation of inference algorithms is time-consuming and error-prone.

**Solution:**

- ▶ Develop **Probabilistic Programming Languages** for expressing probabilistic models as computer programs that generate data (i.e. simulators).
- ▶ Derive **Universal Inference Engines** for these languages that do inference over program traces given observed data (Bayes rule on computer programs).

**Example languages:** BUGS, Infer.NET, BLOG, STAN, Church, Venture, Anglican, Probabilistic C, Stochastic Python\*, Haskell, **Turing**, WebPPL ...

**Example inference algorithms:** Metropolis-Hastings, variational inference, particle filtering, particle cascade, slice sampling\*, particle MCMC, nested particle inference\*, austerity MCMC\*

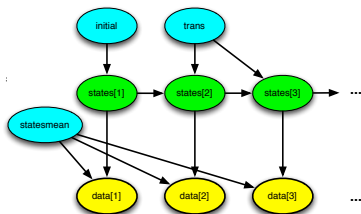
# PROBABILISTIC PROGRAMMING

```
K = 5; N = 201; initial = fill(1.0 / K, K)
means = (collect(1.0:K)*2-K)*2
```

```
@model hmmdemo begin
  states = zeros(Int,N)

  # Uncomment for a Bayesian HMM
  # for i=1:K, T[i,:] ~ Dirichlet(ones(K)./K); end

  states[1] ~ Categorical(initial)
  for i = 2:N
    states[i] ~ Categorical(vec(T[states[i-1],:]))
    obs[i] ~ Normal(means[states[i]], 4)
  end
  return states
end
```



*Probabilistic programming could revolutionise scientific modelling, machine learning, and AI.*

→ NIPS 2015 tutorial by Frank Wood

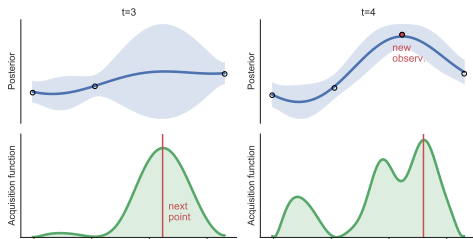
→ Noah Goodman's book

→ Turing: <https://github.com/yebai/Turing.jl>

Show video

# Automating Optimisation: Bayesian optimisation

# BAYESIAN OPTIMISATION

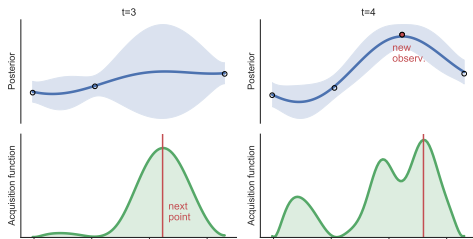


**Problem:** Global optimisation of black-box functions that are *expensive to evaluate*

$$x^* = \arg \max_x f(x)$$



# BAYESIAN OPTIMISATION



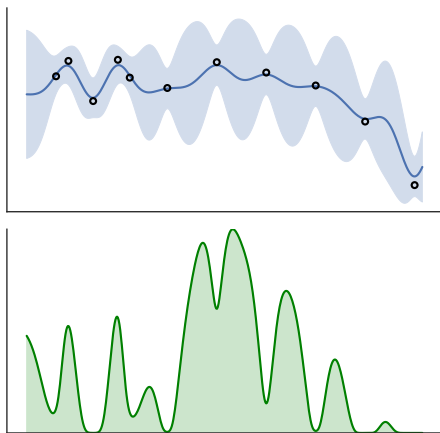
**Problem:** Global optimisation of black-box functions that are *expensive to evaluate*

$$x^* = \arg \max_x f(x)$$

**Solution:** treat as a problem of sequential decision-making and model uncertainty in the function.

*This has myriad applications, from robotics to drug design, to learning neural network hyperparameters.*

# BAYESIAN OPTIMISATION: IN A NUTSHELL



[Moćkus et al., 1978, Jones et al., 1998, Jones, 2001]

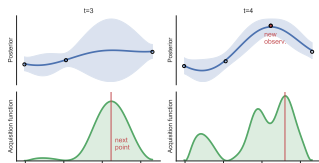
Black-box optimization  
in a nutshell:

- 1 initial sample
- 2 initialize our model
- 3 get the acquisition function  $\alpha(\mathbf{x})$
- 4 optimize it!  
 $\mathbf{x}_{\text{next}} = \arg \max \alpha(\mathbf{x})$
- 5 sample new data;  
update model
- 6 repeat!
- 7 **make recommendation**

(slide thanks to Matthew W. Hoffman)

3/36

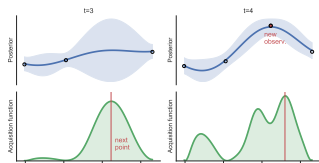
# BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



A good framework for thinking about *any* optimisation problem. It is especially useful if:

- ▶ evaluating the function is expensive

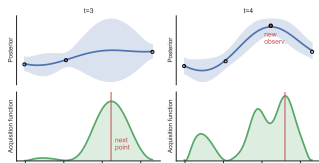
# BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



A good framework for thinking about *any* optimisation problem. It is especially useful if:

- ▶ evaluating the function is expensive
- ▶ evaluating derivatives is hard or impossible

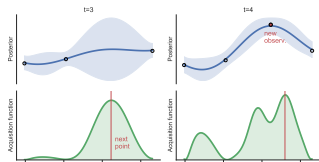
# BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



A good framework for thinking about *any* optimisation problem. It is especially useful if:

- ▶ evaluating the function is expensive
- ▶ evaluating derivatives is hard or impossible
- ▶ there is noise in the function evaluations

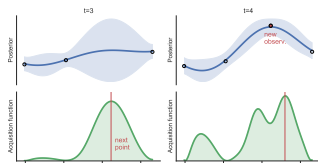
# BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



A good framework for thinking about *any* optimisation problem. It is especially useful if:

- ▶ evaluating the function is expensive
- ▶ evaluating derivatives is hard or impossible
- ▶ there is noise in the function evaluations
- ▶ there may be (possibly noisy) constraints

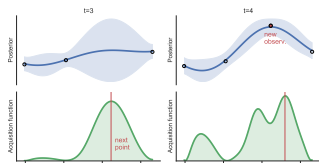
# BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



A good framework for thinking about *any* optimisation problem. It is especially useful if:

- ▶ evaluating the function is expensive
- ▶ evaluating derivatives is hard or impossible
- ▶ there is noise in the function evaluations
- ▶ there may be (possibly noisy) constraints
- ▶ there is prior information about the function

# BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?

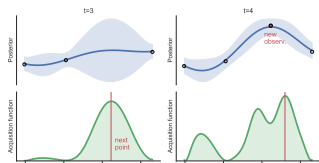


A good framework for thinking about *any* optimisation problem. It is especially useful if:

- ▶ evaluating the function is expensive
- ▶ evaluating derivatives is hard or impossible
- ▶ there is noise in the function evaluations
- ▶ there may be (possibly noisy) constraints
- ▶ there is prior information about the function
- ▶ one needs to optimise many similar functions



# BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?

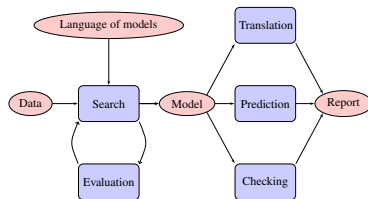


A good framework for thinking about *any* optimisation problem. It is especially useful if:

- ▶ evaluating the function is expensive
- ▶ evaluating derivatives is hard or impossible
- ▶ there is noise in the function evaluations
- ▶ there may be (possibly noisy) constraints
- ▶ there is prior information about the function
- ▶ one needs to optimise many similar functions

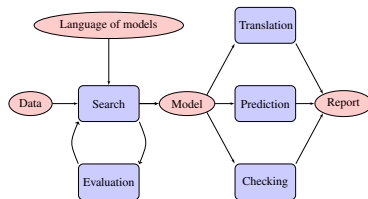
# Automating model discovery: The automatic statistician

# THE AUTOMATIC STATISTICIAN



**Problem:** Data are now ubiquitous; there is great value from understanding this data, building models and making predictions... however, *there aren't enough data scientists, statisticians, and machine learning experts.*

# THE AUTOMATIC STATISTICIAN

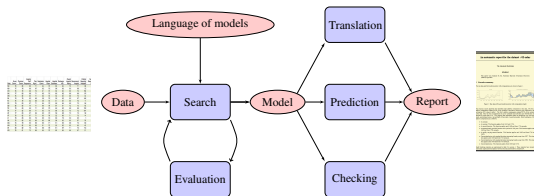


**Problem:** Data are now ubiquitous; there is great value from understanding this data, building models and making predictions... however, *there aren't enough data scientists, statisticians, and machine learning experts.*

**Solution:** Develop a system that automates model discovery from data:

- ▶ processing data, searching over models, discovering a good model, and explaining what has been discovered to the user.

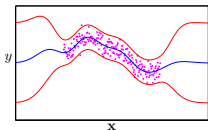
# INGREDIENTS OF AN AUTOMATIC STATISTICIAN



- ▶ **An open-ended language of models**
  - ▶ Expressive enough to capture real-world phenomena...
  - ▶ ...and the techniques used by human statisticians
- ▶ **A search procedure**
  - ▶ To efficiently explore the language of models
- ▶ **A principled method of evaluating models**
  - ▶ Trading off complexity and fit to data
- ▶ **A procedure to automatically explain the models**
  - ▶ Making the assumptions of the models explicit...
  - ▶ ...in a way that is intelligible to non-experts

# BACKGROUND: GAUSSIAN PROCESSES

Consider the problem of **nonlinear regression**: You want to learn a **function  $f$**  with **error bars** from **data  $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$**



A **Gaussian process** defines a distribution over functions  $p(f)$  which can be used for Bayesian regression:

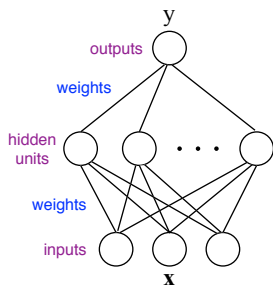
$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

**Definition:**  $p(f)$  is a **Gaussian process** if for *any* finite subset  $\{x_1, \dots, x_n\} \subset \mathcal{X}$ , the marginal distribution over that subset  $p(\mathbf{f})$  is multivariate Gaussian.

GPs can be used for regression, classification, ranking, dim. reduct...

→ **GPflow**: a Gaussian process library using TensorFlow

# BAYESIAN NEURAL NETWORKS AND GAUSSIAN PROCESSES



## Bayesian neural network

Data:  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N = (X, \mathbf{y})$

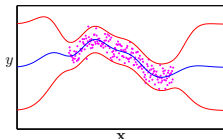
Parameters  $\theta$  are weights of neural net

prior  $p(\theta|\alpha)$

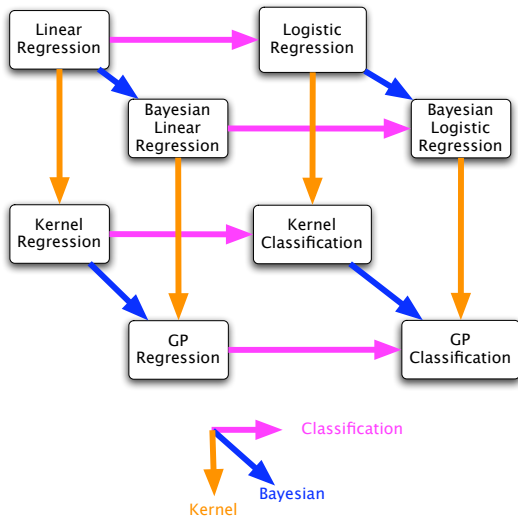
posterior  $p(\theta|\alpha, \mathcal{D}) \propto p(\mathbf{y}|X, \theta)p(\theta|\alpha)$

prediction  $p(y'|\mathcal{D}, \mathbf{x}', \alpha) = \int p(y'|\mathbf{x}', \theta)p(\theta|\mathcal{D}, \alpha) d\theta$

A neural network with one hidden layer, infinitely many hidden units and Gaussian priors on the weights  $\rightarrow$  a GP (Neal, 1994). He also analysed infinitely deep networks.



# A PICTURE: GPs, LINEAR AND LOGISTIC REGRESSION, AND SVMs

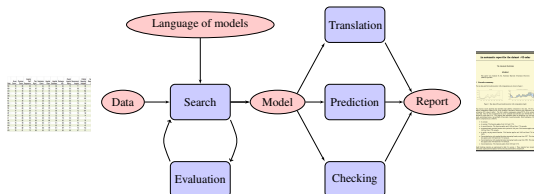


→ GPflow: a Gaussian process library using TensorFlow



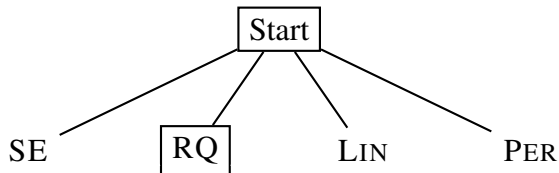
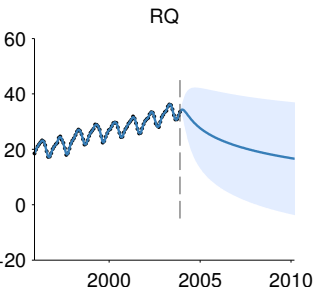
# Automatic Statistician for Regression and Time-Series Models

# INGREDIENTS OF AN AUTOMATIC STATISTICIAN

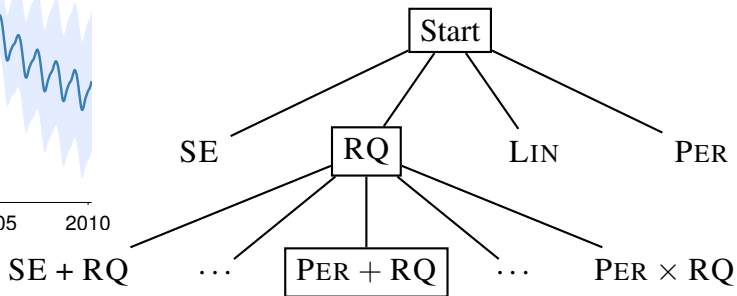
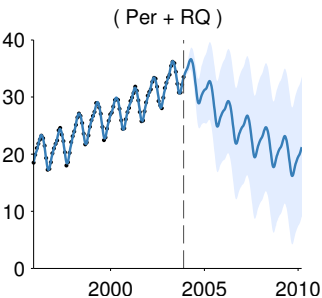


- ▶ **An open-ended language of models**
  - ▶ Expressive enough to capture real-world phenomena...
  - ▶ ...and the techniques used by human statisticians
- ▶ **A search procedure**
  - ▶ To efficiently explore the language of models
- ▶ **A principled method of evaluating models**
  - ▶ Trading off complexity and fit to data
- ▶ **A procedure to automatically explain the models**
  - ▶ Making the assumptions of the models explicit...
  - ▶ ...in a way that is intelligible to non-experts

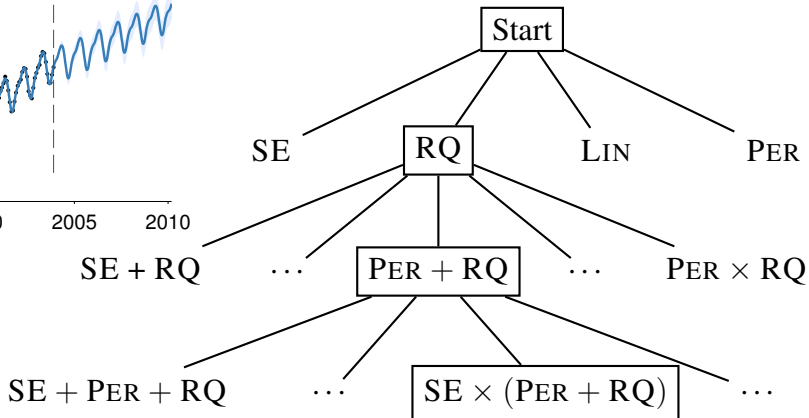
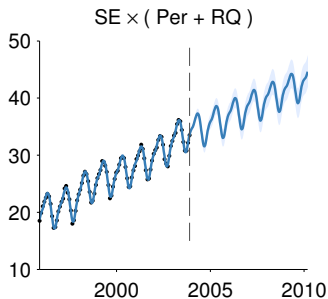
# MODEL SEARCH: MAUNA LOA KEELING CURVE



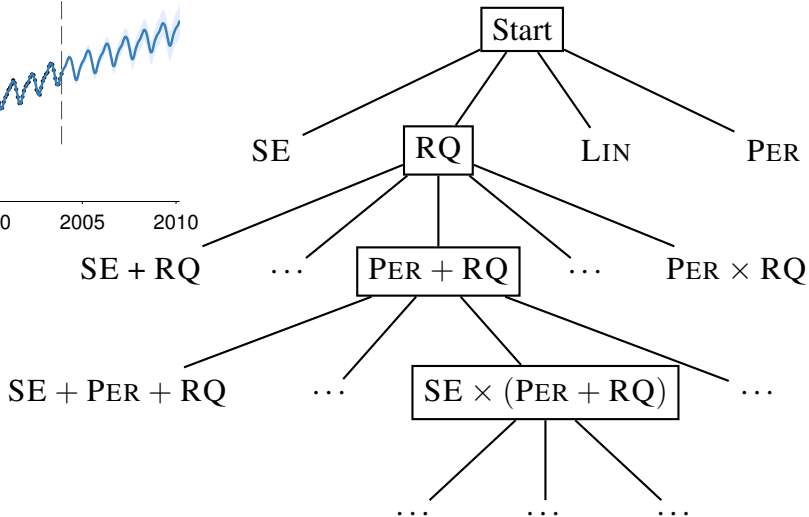
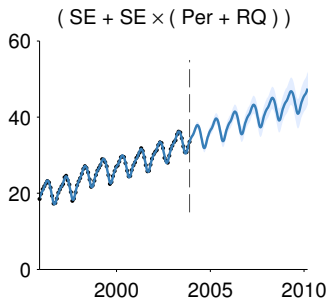
# MODEL SEARCH: MAUNA LOA KEELING CURVE



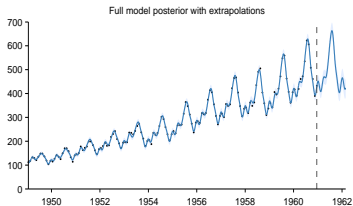
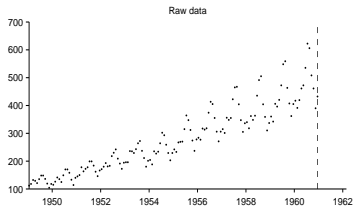
# MODEL SEARCH: MAUNA LOA KEELING CURVE



# MODEL SEARCH: MAUNA LOA KEELING CURVE



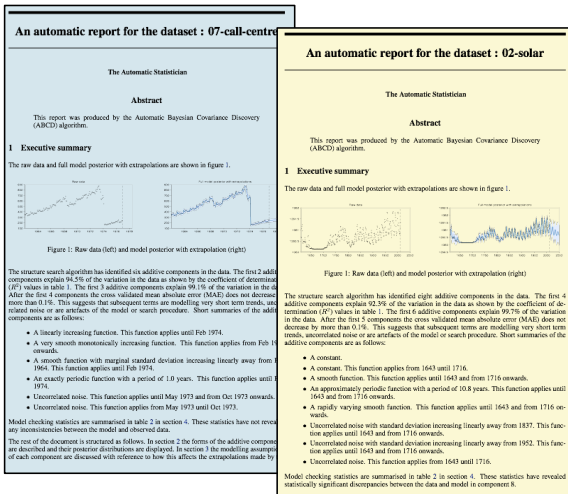
# EXAMPLE: AN ENTIRELY AUTOMATIC ANALYSIS



Four additive components have been identified in the data

- ▶ A linearly increasing function.
- ▶ An approximately periodic function with a period of 1.0 years and with linearly increasing amplitude.
- ▶ A smooth function.
- ▶ Uncorrelated noise with linearly increasing standard deviation.

# EXAMPLE REPORTS

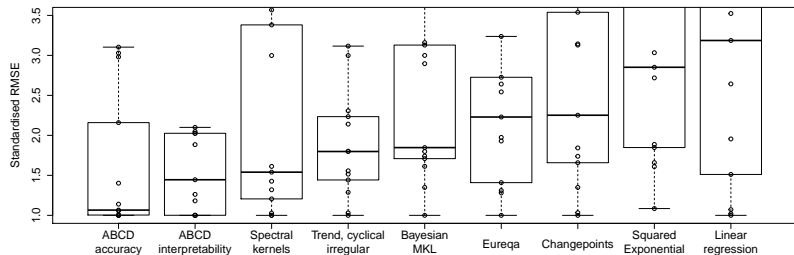


See <http://www.automaticstatistician.com>



# GOOD PREDICTIVE PERFORMANCE AS WELL

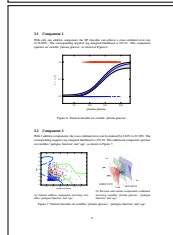
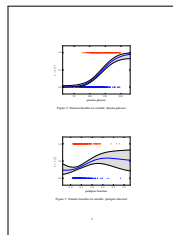
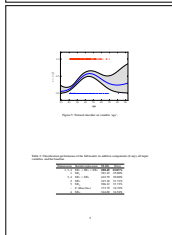
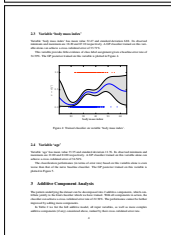
## Standardised RMSE over 13 data sets



- ▶ Tweaks can be made to the algorithm to improve accuracy or interpretability of models produced. . .
- ▶ . . . but both methods are *highly competitive* at extrapolation

# Automatic Statistician for Classification

# An AutoGPC Report on the Pima Diabetes Dataset<sup>2</sup>



<sup>2</sup>Obtained from [Smith et al., 1988]

# AutoGPC Report: Executive Summary

## ► Full model description

..., the model involves an additive combination of variable 'plasma glucose' and a 2-way interaction between variables 'pedigree function' and 'age' ( $SE_1 + SE_3 \times SE_4$ ). This model can achieve a cross-validated error rate of 22.38% and a negative log marginal likelihood of 353.28.

## ► Summary of input variables

Variable	Statistics			Classifier Performance		
	Min	Max	Mean	Kernel	NLML	Error
1 plasma glucose	44.00	199.00	121.88	smooth	381.22	<b>25.00%</b>
2 body mass index	18.20	67.10	32.47	smooth	347.18	33.71%
3 pedigree function	0.08	2.42	0.47	smooth	366.12	33.71%
– <i>Baseline</i>	–	–	–	<i>constant</i>	373.79	34.39%
4 age	21.00	81.00	33.35	smooth	<b>344.69</b>	34.54%

## AutoGPC Report: Individual Variable Analysis

- ▶ Variable 'plasma glucose': moderate evidence, monotonic  
... Compared with the null model (baseline), this variable contains **some evidence** of class label assignment. There is a **positive correlation** between the value of this variable and the likelihood of the sample being classified as positive.
- ▶ Variable 'body mass index': little evidence  
... This variable provides **little evidence** of class label assignment given a baseline error rate of 34.39%.
- ▶ Variable 'age': no evidence  
... The classification performance (in terms of error rate) based on this variable alone is even **worse than** that of the naïve baseline classifier.
- ▶ Other variants: **strong evidence**, etc.

## AutoGPC Report: Additive Component Analysis

- ▶ For example, for the second additive component

[With only one additive component, ...]

With **2 additive components**, the cross-validated error can be **reduced by 2.62% to 22.38%**. The corresponding negative log marginal likelihood is 353.28. The additional component operates on variables ‘pedigree function’ and ‘age’, as shown in Figure ...

- ▶ Summary of all models

Dim	Kernel expression	NLML	Error
1, 3, 4	$SE_1 + SE_3 \times SE_4$	<b>280.45</b>	<b>21.83%</b>
1	$SE_1$	381.22	25.00%
3, 4	$SE_3 \times SE_4$	422.79	30.80%
2	$SE_2$	347.18	33.71%
3	$SE_3$	366.12	33.71%
–	<i>C (Baseline)</i>	373.79	34.39%
4	$SE_4$	344.69	34.54%

# MODEL CHECKING AND CRITICISM

- ▶ Good statistical modelling should include model criticism:
  - ▶ *Does the data match the assumptions of the model?*
- ▶ Our automatic statistician does posterior predictive checks, dependence tests and residual tests
- ▶ We have also been developing more systematic nonparametric approaches to model criticism using kernel two-sample testing:

→ Lloyd, J. R., and Ghahramani, Z. (2015) Statistical Model Criticism using Kernel Two Sample Tests. *NIPS 2015*.

# THE AUTOML COMPETITION

- ▶ New algorithms for building machine learning systems that learn under **strict time, CPU, memory and disk space constraints**, making decisions about where to allocate computational resources so as to maximise statistical performance.

ChaLearn Automatic Machine Learning Challenge (AutoML)

RESULTS								
User	<Rank>	Set 1	Set 2	Set 3	Set 4	Set 5	Duration	Detailed Results
1 backstreet.bayes	1.80 (1)	0.3193 (3)	0.9198 (1)	0.3361 (1)	0.3495 (2)	0.2351 (2)	5954.71 (2)	<a href="#">View</a>
2 aad.freiburg	3.40 (2)	0.3305 (8)	0.8600 (2)	0.3311 (2)	0.3305 (4)	0.2359 (1)	5987.17 (1)	<a href="#">View</a>
3 lukasz.romaszko	3.80 (3)	0.3304 (2)	0.6662 (4)	0.2647 (5)	0.3440 (3)	0.2194 (5)	711.28 (7)	<a href="#">View</a>
4 matthias.vonrohr	4.40 (4)	0.3029 (5)	0.5939 (5)	0.3064 (3)	0.2994 (6)	0.2220 (3)	4964.44 (4)	<a href="#">View</a>
5 marc.bouille	4.40 (4)	0.3533 (1)	0.4561 (7)	0.2130 (7)	0.3692 (1)	0.1434 (6)	4315.09 (6)	<a href="#">View</a>
6 guyon	4.60 (5)	0.3031 (4)	0.5915 (6)	0.2976 (4)	0.3027 (5)	0.2202 (4)	4823.69 (5)	<a href="#">View</a>
7 tadj	6.20 (6)	0.2956 (6)	0.7164 (3)	0.2616 (6)	0.1403 (8)	0.0003 (8)	5439.29 (3)	<a href="#">View</a>
8 Geek	7.40 (7)	0.2550 (7)	0.0583 (8)	0.1052 (8)	0.2194 (7)	0.0068 (7)	679.30 (8)	<a href="#">View</a>

- ▶ Second and First place in the first two rounds of the AutoML classification challenge to “*design machine learning methods capable of performing all model selection and parameter tuning without any human intervention.*”



# CONCLUSIONS

**Probabilistic modelling** offers a framework for building systems that **reason about uncertainty** and **learn from data**, going beyond traditional pattern recognition problems.

I have *briefly* reviewed some of the frontiers of our research, centred around the theme of **automating machine learning**, including:

- ▶ The automatic statistician
- ▶ Probabilistic programming
- ▶ Bayesian optimisation

Ghahramani, Z. (2015) Probabilistic machine learning and artificial intelligence. *Nature* **521**:452–459.

<http://www.nature.com/nature/journal/v521/n7553/full/nature14541.html>

# COLLABORATORS

Ryan P. Adams

Yutian Chen

David Duvenaud

Yarin Gal

Hong Ge

Michael A. Gelbart

Roger Grosse

José Miguel Hernández-Lobato

Matthew W. Hoffman

James R. Lloyd

David J. C. MacKay

Adam Ścibior

Amar Shah

Emma Smith

Christian Steinruecken

Joshua B. Tenenbaum

Andrew G. Wilson

Kai Xu

## General:

Ghahramani, Z. (2013) Bayesian nonparametrics and the probabilistic approach to modelling. *Philosophical Trans. Royal Society A* 371: 20110553.

Ghahramani, Z. (2015) Probabilistic machine learning and artificial intelligence *Nature* **521**:452–459. <http://www.nature.com/nature/journal/v521/n7553/full/nature14541.html>

## Automatic Statistician:

Website: <http://www.automaticstatistician.com>

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B. and Ghahramani, Z. (2013) Structure Discovery in Nonparametric Regression through Compositional Kernel Search. ICML 2013.

Lloyd, J. R., Duvenaud, D., Grosse, R., Tenenbaum, J. B. and Ghahramani, Z. (2014) Automatic Construction and Natural-language Description of Nonparametric Regression Models AAAI 2014. <http://arxiv.org/pdf/1402.4304v2.pdf>

Lloyd, J. R., and Ghahramani, Z. (2015) Statistical Model Criticism using Kernel Two Sample Tests. <http://mlg.eng.cam.ac.uk/Lloyd/papers/kernel-model-checking.pdf>. NIPS 2015.

## Bayesian Optimisation:

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. (2014) Predictive entropy search for efficient global optimization of black-box functions. NIPS 2014

Hernández-Lobato, J.M., Gelbart, M.A., Adams, R.P., Hoffman, M.W., Ghahramani, Z. (2016) A General Framework for Constrained Bayesian Optimization using Information-based Search. *Journal of Machine Learning Research*. **17**(160):1–53.

## **Probabilistic Programming:**

Turing: <https://github.com/yebai/Turing.jl>

Chen, Y., Mansinghka, V., Ghahramani, Z. (2014) *Sublinear-Time Approximate MCMC Transitions for Probabilistic Programs*. arXiv:1411.1690

Ge, Hong, Adam Scibior, and Zoubin Ghahramani (2016) *Turing: rejuvenating probabilistic programming in Julia*. (In preparation).

## **Bayesian neural networks:**

José Miguel Hernández-Lobato and Ryan Adams. *Probabilistic backpropagation for scalable learning of Bayesian neural networks*. ICML, 2015.

Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian approximation: Representing model uncertainty in deep learning*. ICML, 2016.

Yarin Gal and Zoubin Ghahramani. *A theoretically grounded application of dropout in recurrent neural networks*. NIPS, 2016.

José Miguel Hernández-Lobato, Yingzhen Li, Daniel Hernández-Lobato, Thang Bui, and Richard E Turner. *Black-box alpha divergence minimization*. ICML, 2016.

# THE ATOMS OF OUR LANGUAGE OF MODELS

## Five base kernels



Squared  
exp. (SE)



Periodic  
(PER)



Linear  
(LIN)



Constant  
(C)



White  
noise (WN)

## Encoding for the following types of functions



Smooth  
functions



Periodic  
functions



Linear  
functions



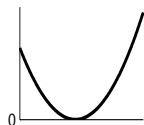
Constant  
functions



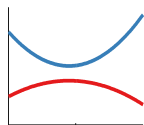
Gaussian  
noise

# THE COMPOSITION RULES OF OUR LANGUAGE

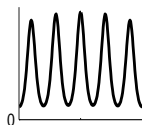
- ▶ Two main operations: addition, multiplication



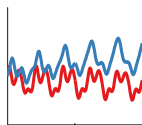
$\text{LIN} \times \text{LIN}$



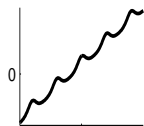
quadratic  
functions



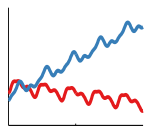
$\text{SE} \times \text{PER}$



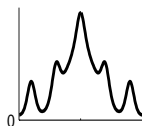
locally  
periodic



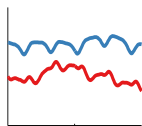
$\text{LIN} + \text{PER}$



periodic plus  
linear trend



$\text{SE} + \text{PER}$



periodic plus  
smooth trend

# BAYESIAN OPTIMISATION

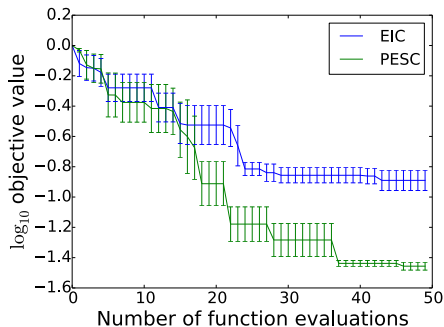


Figure 4. Classification error of a 3-hidden-layer neural network constrained to make predictions in under 2 ms.

(work with J.M. Hernández-Lobato, M.A. Gelbart, M.W. Hoffman, & R.P. Adams) [arXiv:1511.09422](#) [arXiv:1511.07130](#) [arXiv:1406.2541](#)