Probabilistic machine learning: foundations and frontiers

Zoubin Ghahramani[1,2]

[1] University of Cambridge

[2] Uber AI Labs

zoubin@eng.cam.ac.uk

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

https://www.uber.com/info/ailabs/

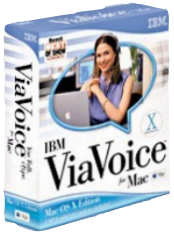
MLSS 2017

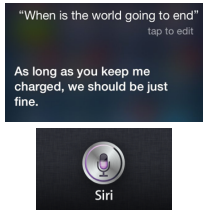
Tübingen

An exciting time for Machine Learning!

Applications of Machine Learning

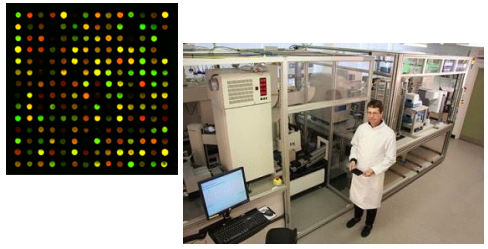
Speech and Language Technologies,
automatic speech recognition,
machine translation, question-answering,
dialog systems





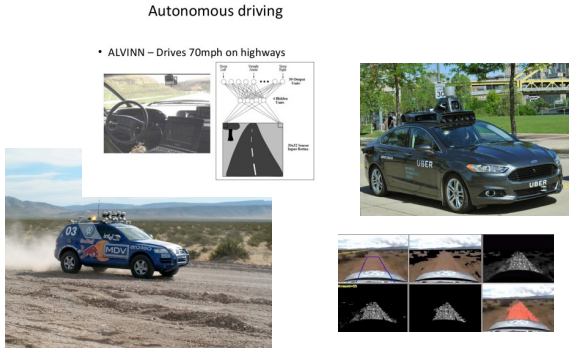
Computer Vision:
Object, Face and Handwriting
Recognition, Image Captioning

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



Recommender Systems

Self-driving cars

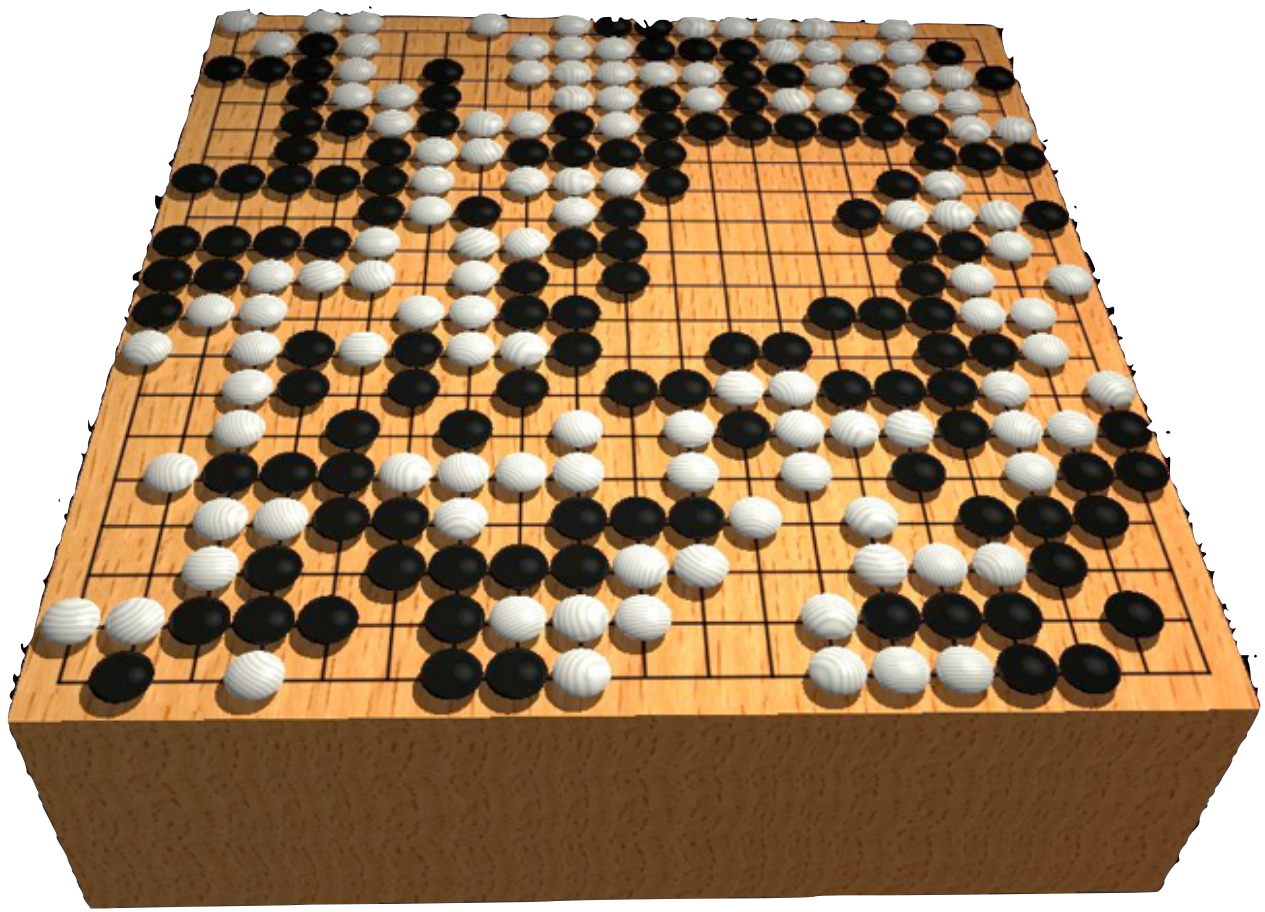


Financial Prediction and Automated Trading

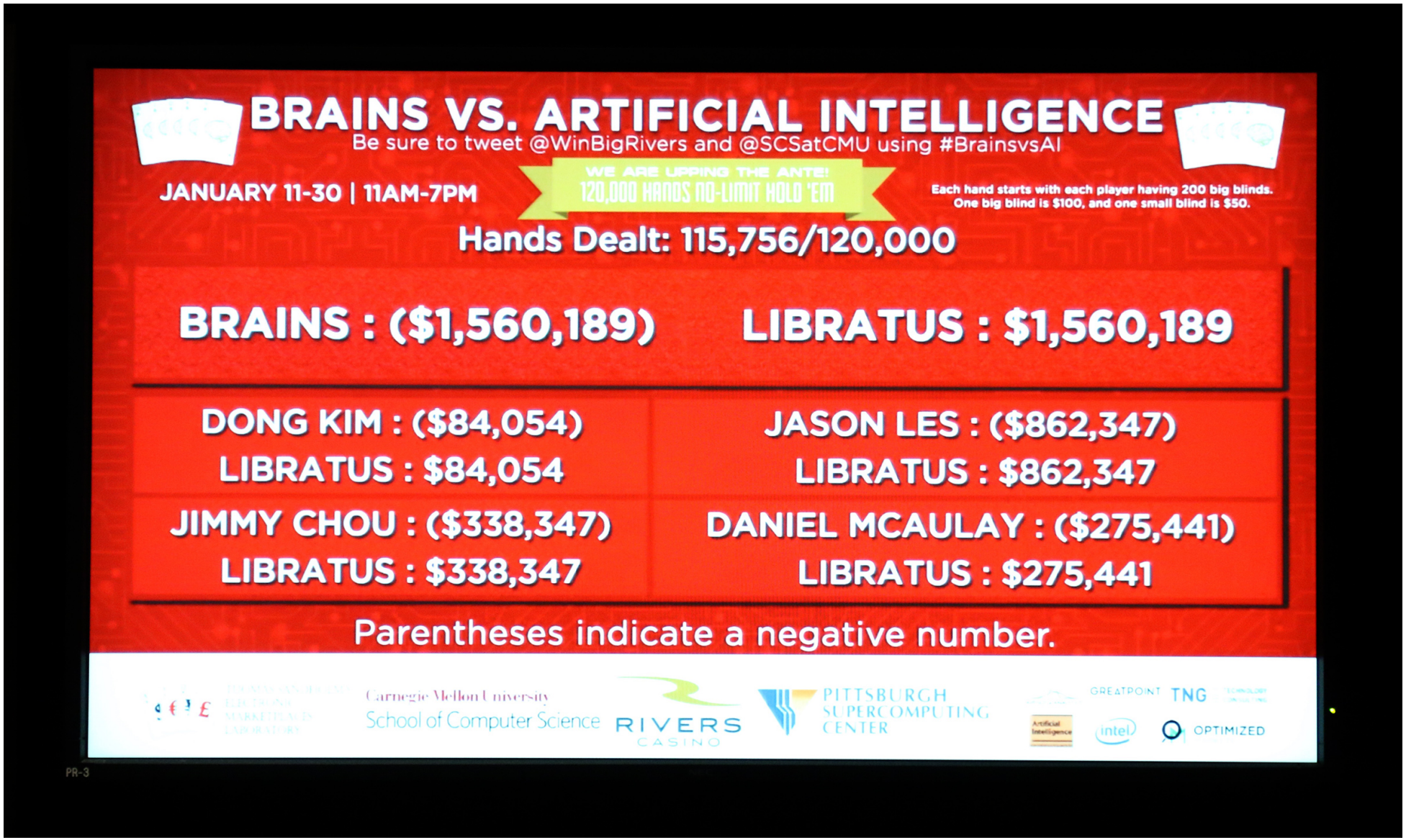
Computer Games

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

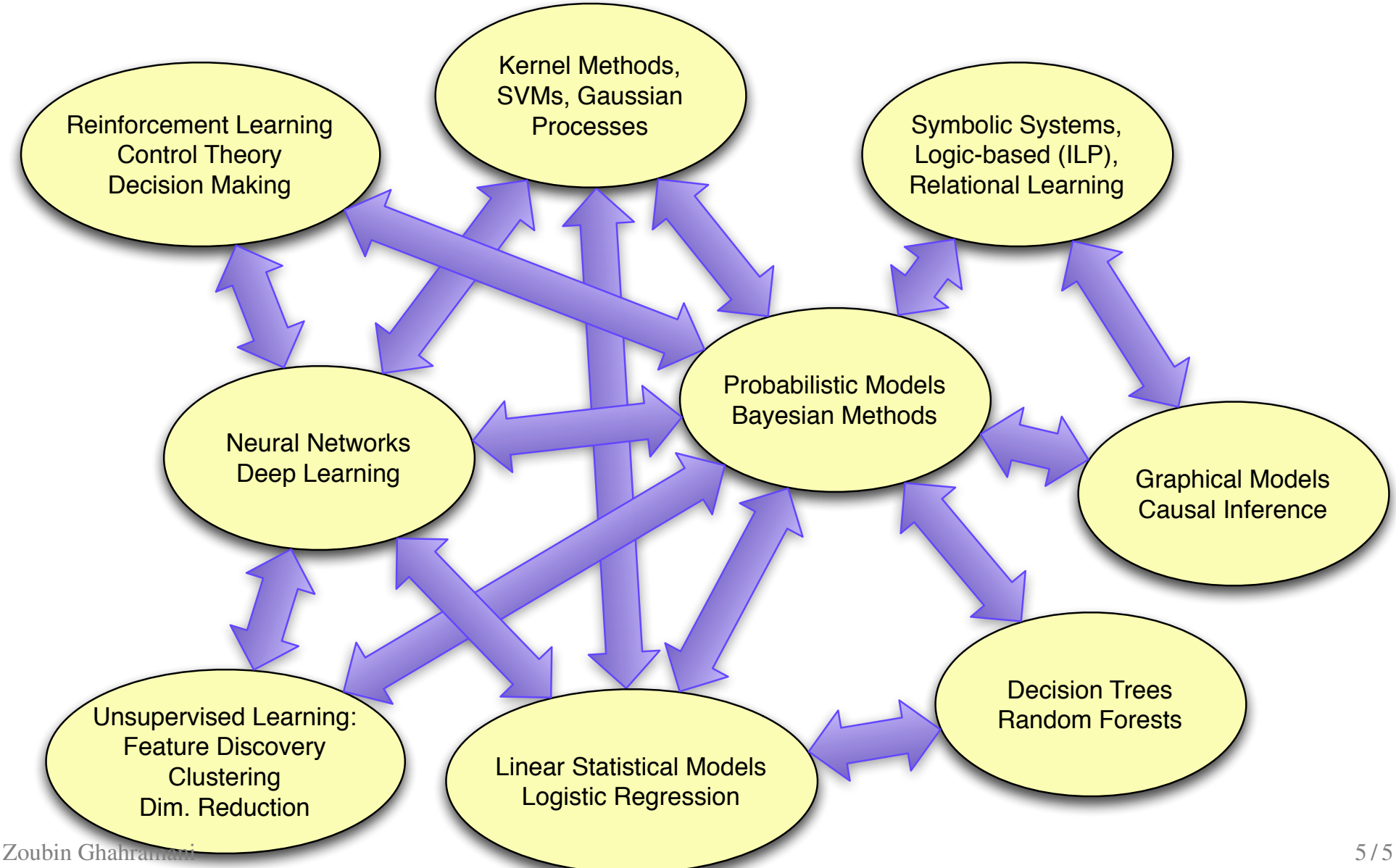
Computer Games



GAMES

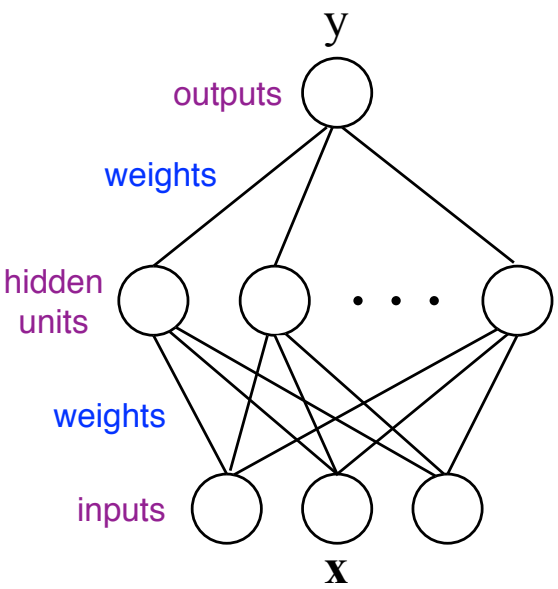

BRAINS VS. ARTIFICIAL INTELLIGENCE
Be sure to tweet @WinBigRivers and @SCSatCMU using #BrainsvsAI
JANUARY 11-30 | 11AM-7PM
WE ARE UPPING THE ANTE!
120,000 HANDS NO-LIMIT HOLD 'EM
Each hand starts with each player having 200 big blinds.
One big blind is $100, and one small blind is $50.
Hands Dealt: 115,756/120,000
BRAINS : ($1,560,189)
LIBRATUS : $1,560,189
DONG KIM : ($84,054)
LIBRATUS : $84,054
JASON LES : ($862,347)
LIBRATUS : $862,347
JIMMY CHOU : ($338,347)
LIBRATUS : $338,347
DANIEL MCAULAY : ($275,441)
LIBRATUS : $275,441
Parentheses indicate a negative number.
Carnegie Mellon University
School of Computer Science
RIVERS
CASINO
PITTSBURGH
SUPERCOMPUTING
CENTER
GREATPOINT
TNG
intel
OPTIMIZED
PR-3

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 $\boldsymbol{\theta}$ are weights of neural net.

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

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

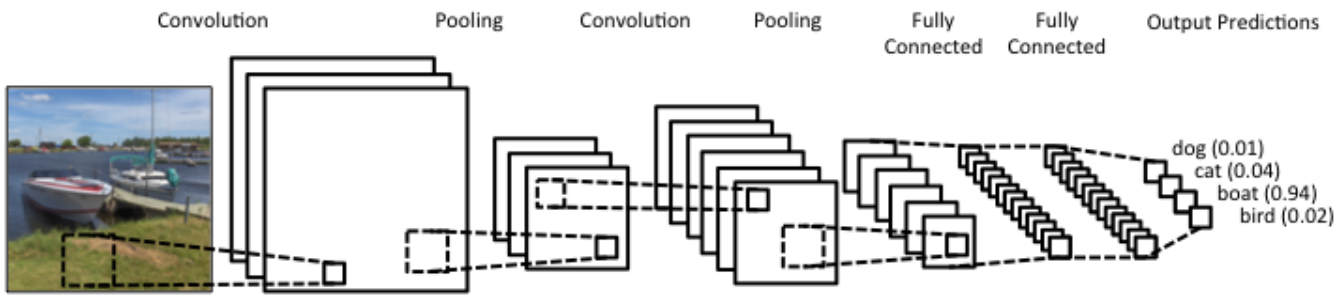
Multilayer neural networks model the overall function as a composition of functions (layers), e.g.:

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

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 trasparency, 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

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

- 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 θ in model m
$P(\theta\|m)$	prior probability of θ
$P(\theta\|\mathcal{D},m)$	posterior of θ 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:

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

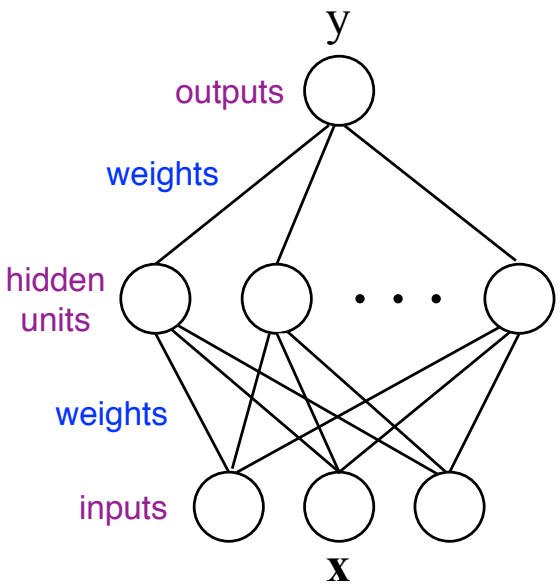
WHY SHOULD WE CARE?

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 $p(y^{(n)}|\mathbf{x}^{(n)}, \boldsymbol{\theta})$

Parameters $\boldsymbol{\theta}$ are weights of neural net.

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

A NOTE ON MODELS VS ALGORITHMS

In early NIPS there was an "Algorithms and Architectures" track

Models:	**Algorithms**
convnets	Stochastic Gradient Descent
LDA	Conjugate-gradients
RNNs	MCMC
HMMs	Variational Bayes and SVI
Boltzmann machines	SGLD
State-space models	Belief propagation, EP
Gaussian processes	...
LSTMs	...

There are algorithms that target finding a parameter optimum, θ^* and algorithms that target inferring the posterior $p(\theta|D)$

Often these are *not so different*

Let's be clear: *"Bayesian" belongs in the Algorithms category, not the Models category*. Any well defined model can be treated in a Bayesian manner.

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)

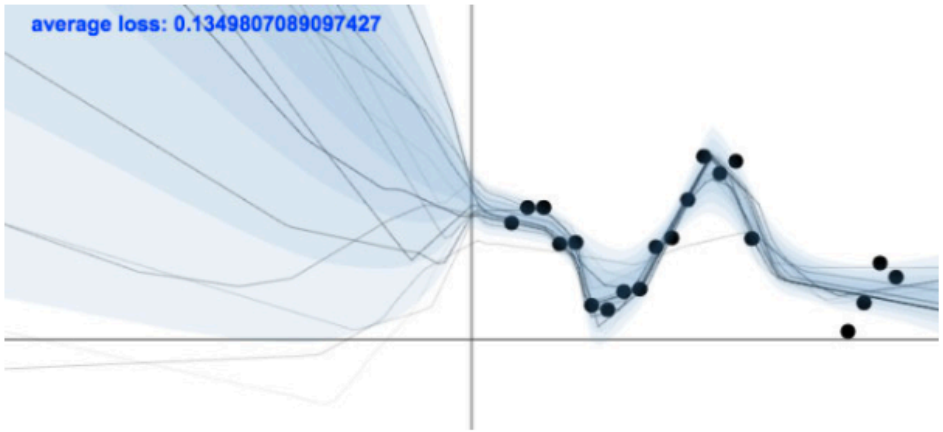


Figure from Yarin Gal's thesis "Uncertainty in Deep Learning" (2016)

→ NIPS 2016 workshop on Bayesian Deep Learning

When do we need probabilities?

When is the probabilistic approach essential?

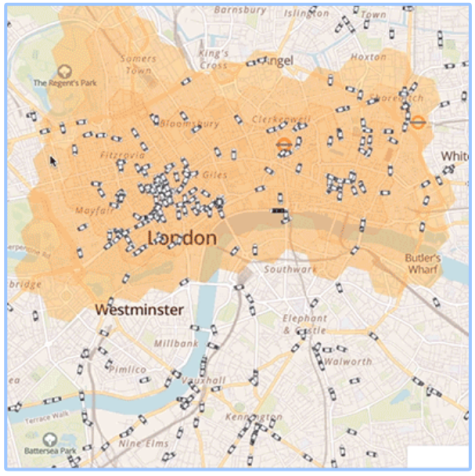
Many aspects of learning and intelligence depend crucially on the careful probabilistic representation of *uncertainty*:

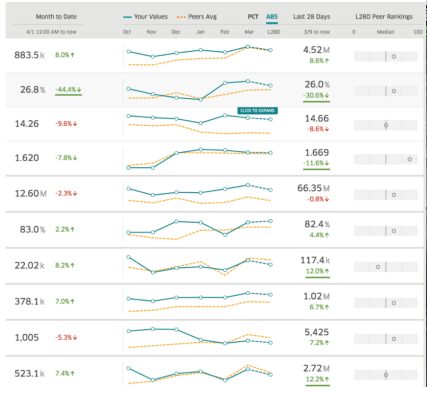
- Forecasting
- Decision making
- Learning from limited, noisy, and missing data
- Learning complex personalised models
- Data compression
- Automating modelling, discovery, and experiment design

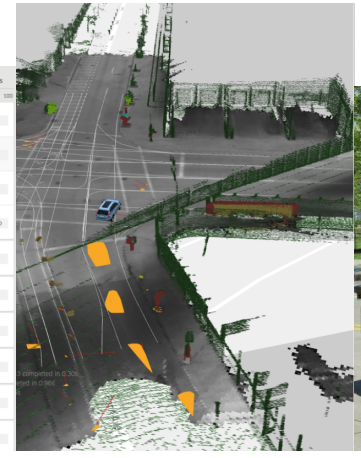
WHY DOES UBER CARE?

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

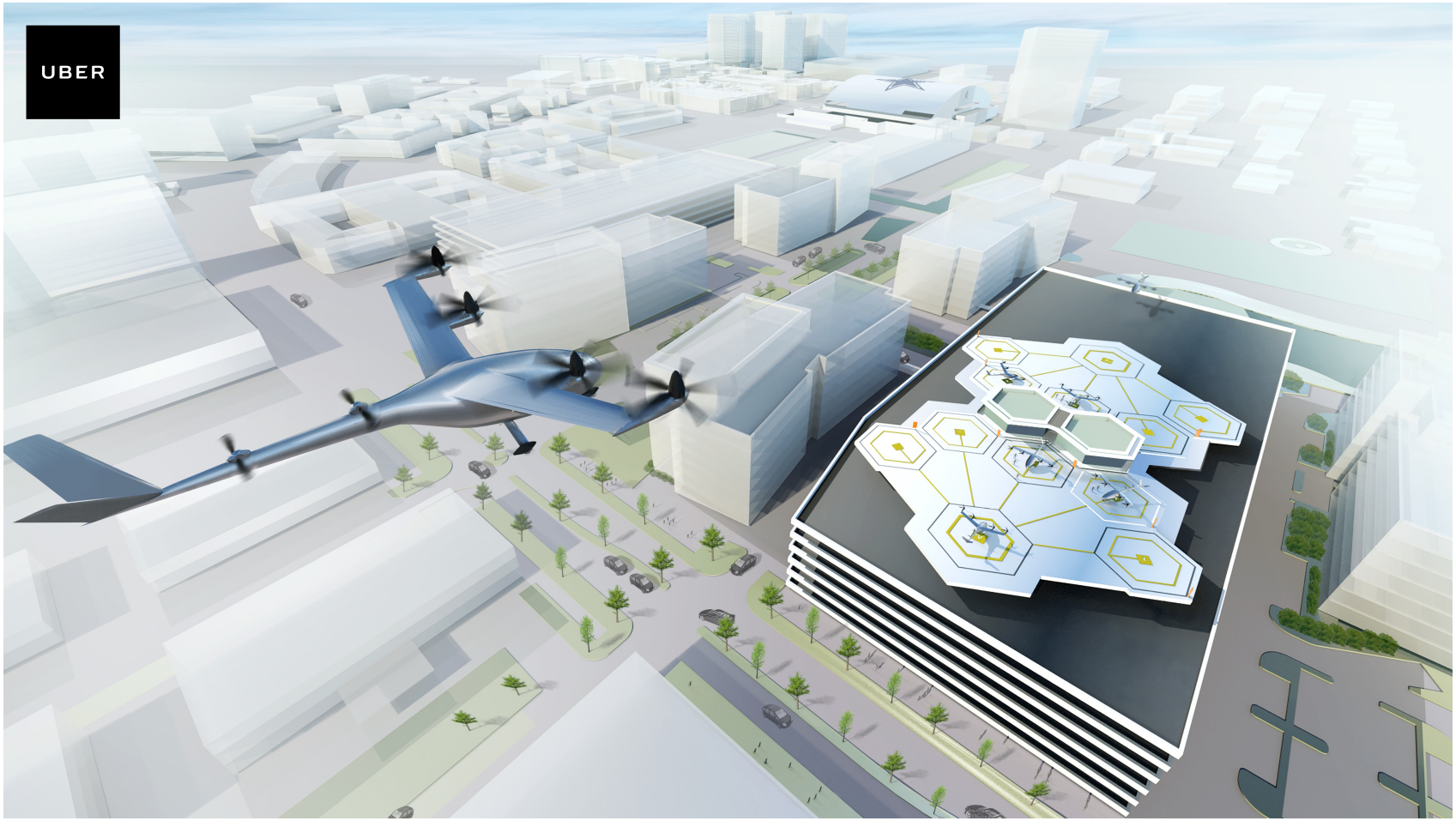
- Forecasting, Decision making, Learning from limited, noisy, and missing data, Learning complex personalised models, Data compression, Automating modelling, discovery, and experiment design

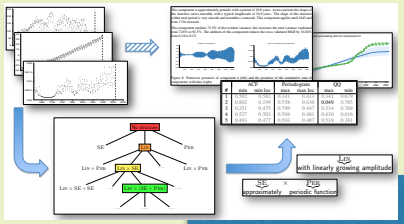






supply (driver) and demand (rider) prediction, ETA prediction, pricing, business intelligence, modelling traffic and cities, mapping, self-driving cars, customer service, UberEATs personalization...


UBER

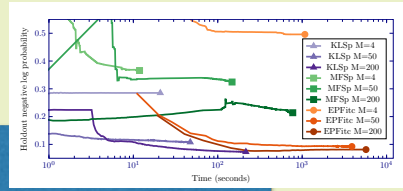


Automatic Statistician

```
statesmean = [-1, 1, 0]  # Emission parameters.
initial    = Categorical([1.0/3, 1.0/3, 1.0/3]) # Prob distr of state[1].
trans      = [Categorical([0.1, 0.5, 0.4]), Categorical([0.2, 0.2, 0.6]),
              Categorical([0.15, 0.15, 0.7])]  # Trans distr for each state.
data       = [Nil, 0.9, 0.8, 0.7, 0, -0.025, -5, -2, -0.1, 0, 0.13]

@model hmm begin # Define a model hmm.
 states = Array(Int, length(data))
 @assume(states[1] ~ initial)
 for i = 2:length(data)
   @assume(states[i] ~ trans[states[i-1]])
   @observe(data[i]  ~ Normal(statesmean[states[i]], 0.4))
 end
 @predict states
end
```

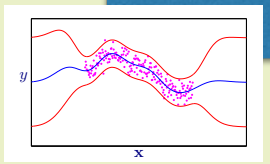
Probabilistic Programming



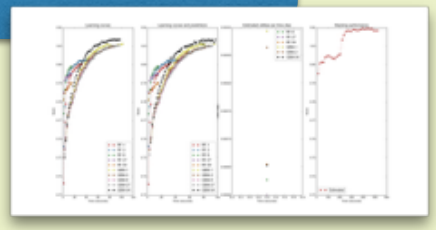
Large-scale Inference

Automating Machine Learning

Bayesian Nonparametrics

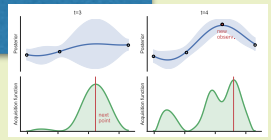


Computational Resource Allocation



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*

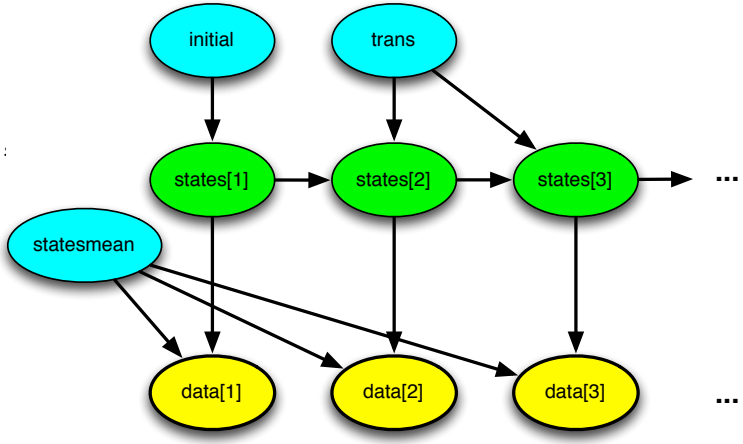
PROBABILISTIC PROGRAMMING

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

@model hmmdemo begin
    states = tzeros(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

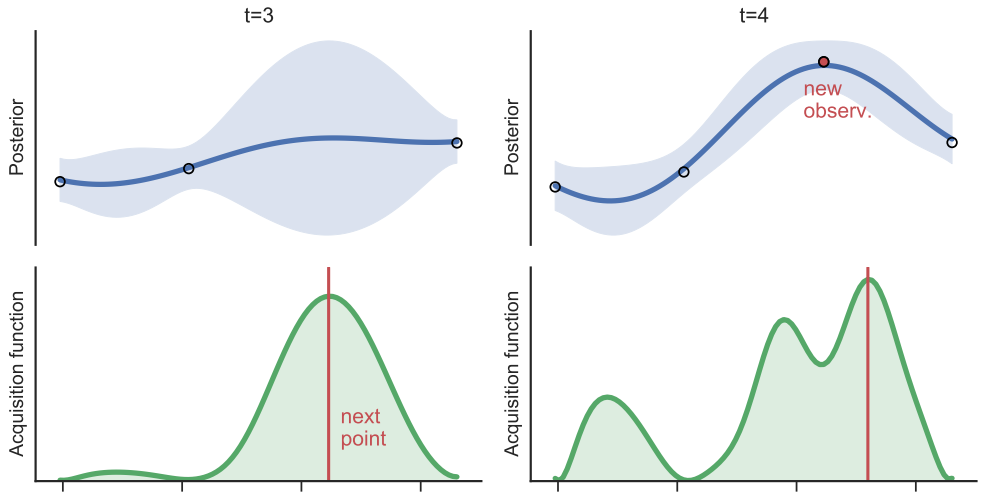
→ Turing: `https://github.com/yebai/Turing.jl`

TURING

Show video

Automating Optimisation: Bayesian optimisation

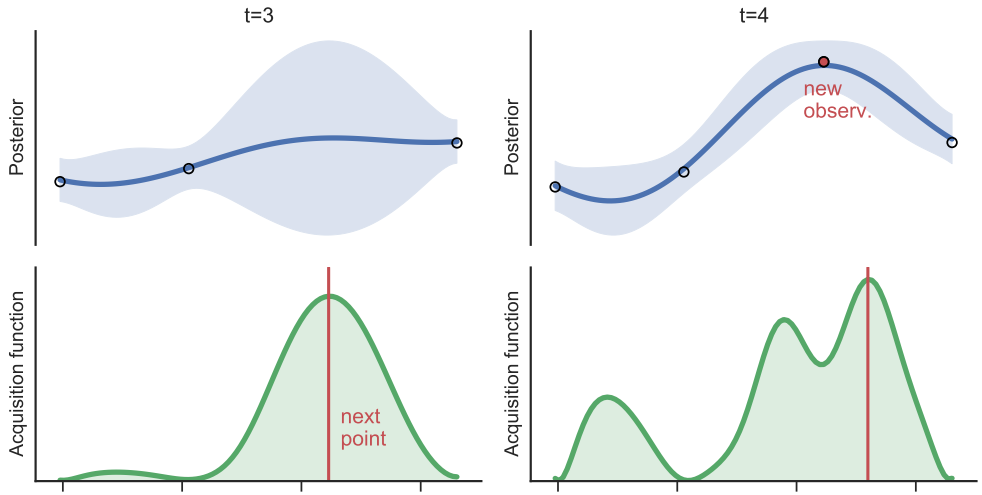
BAYESIAN OPTIMISATION



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

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

BAYESIAN OPTIMISATION



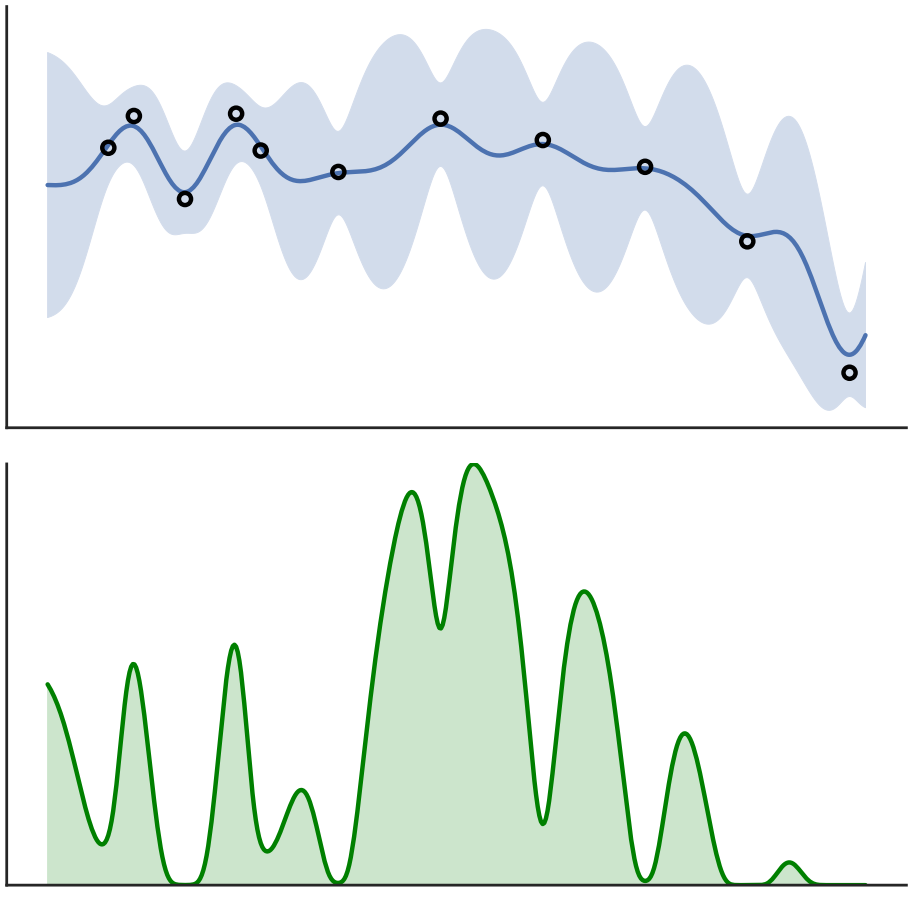
Problem: Global optimisation of black-box functions that are *expensive to evaluate*

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

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



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

Black-box optimization in a nutshell:

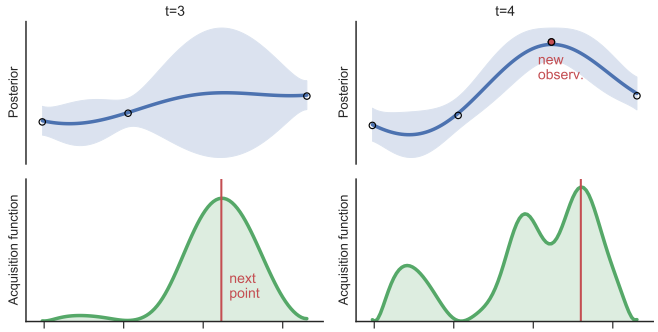
1. initial sample
2. initialize our model
3. get the acquisition function $\alpha(\mathbf{x})$
4. optimize it! $\mathbf{x}_{\text{next}} = \arg\max \alpha(\mathbf{x})$
5. sample new data; update model
6. repeat!

7. **make recommendation**

3/36

(slide thanks to Matthew W. Hoffman)

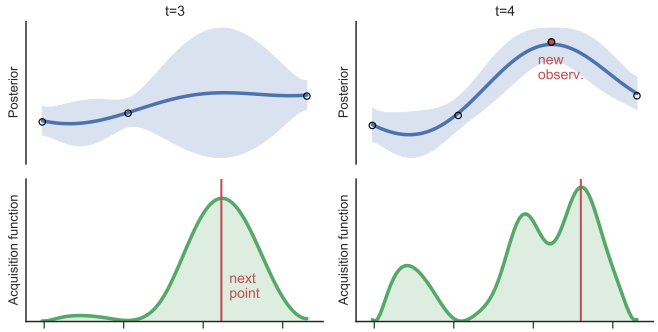
BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



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

- evaluating the function is expensive

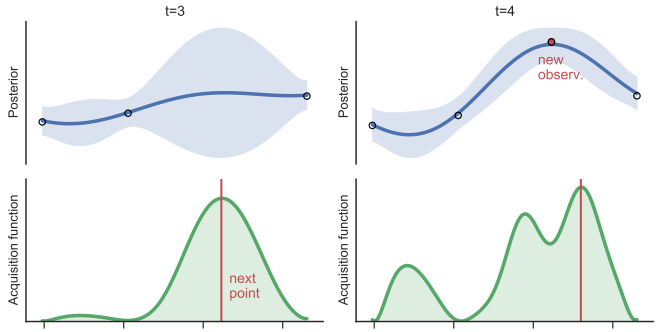
BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



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

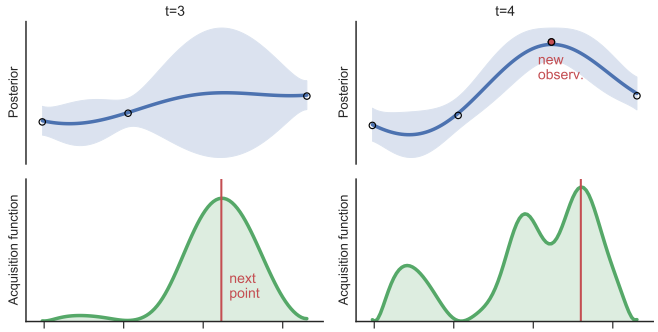
BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



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

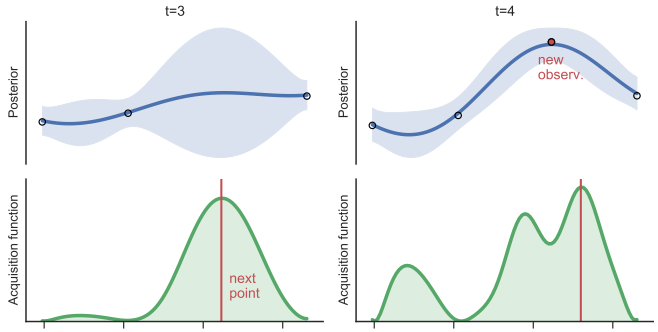
BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



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

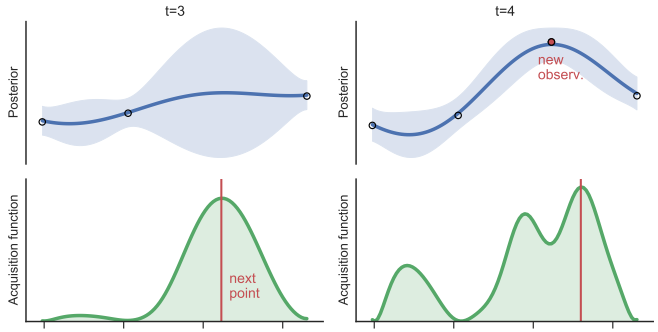
BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



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

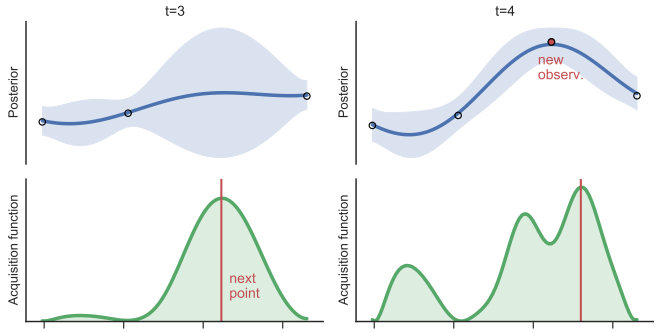
BAYESIAN OPTIMISATION: WHY IS IT IMPORTANT?



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

BAYESIAN OPTIMISATION

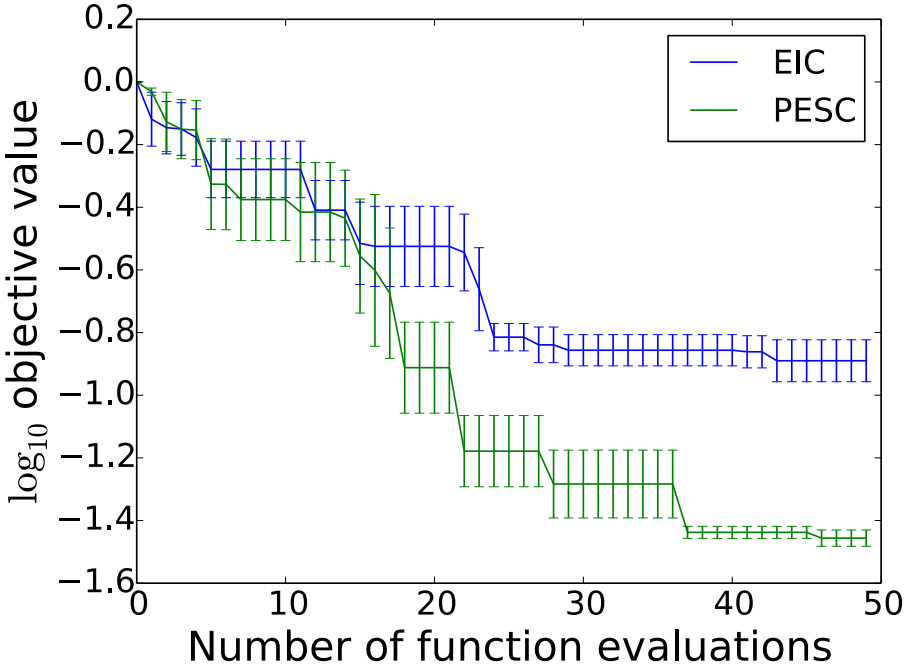
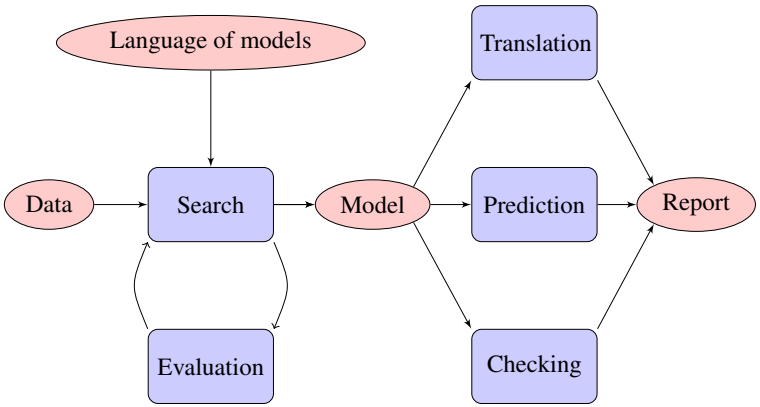


Figure 4. Classification error of a 3-hidden-layer neural network constrained to make predictions in under 2 ms.

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

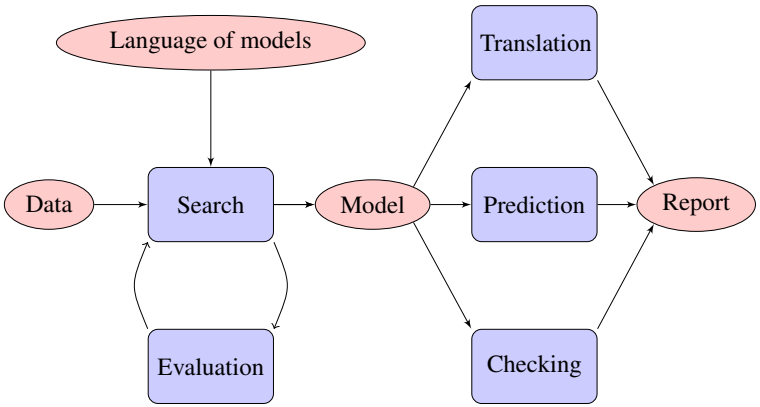
Automating model discovery: The automatic statistician

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

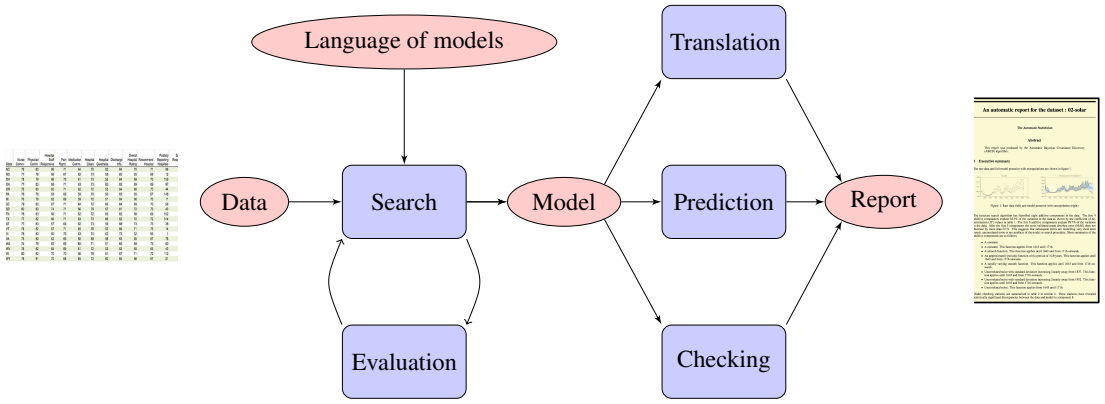


Problem: Data are now ubiquitous; there is great value from understanding this data, building models and making predictions... however, *there aren't enough data scientists, statisticians, and machine learning experts.*

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.

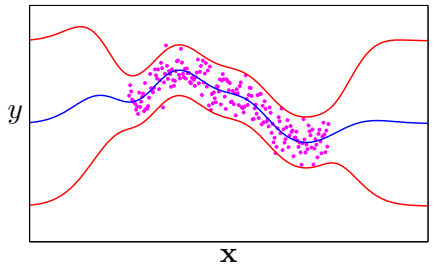
INGREDIENTS OF AN AUTOMATIC STATISTICIAN



- **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:

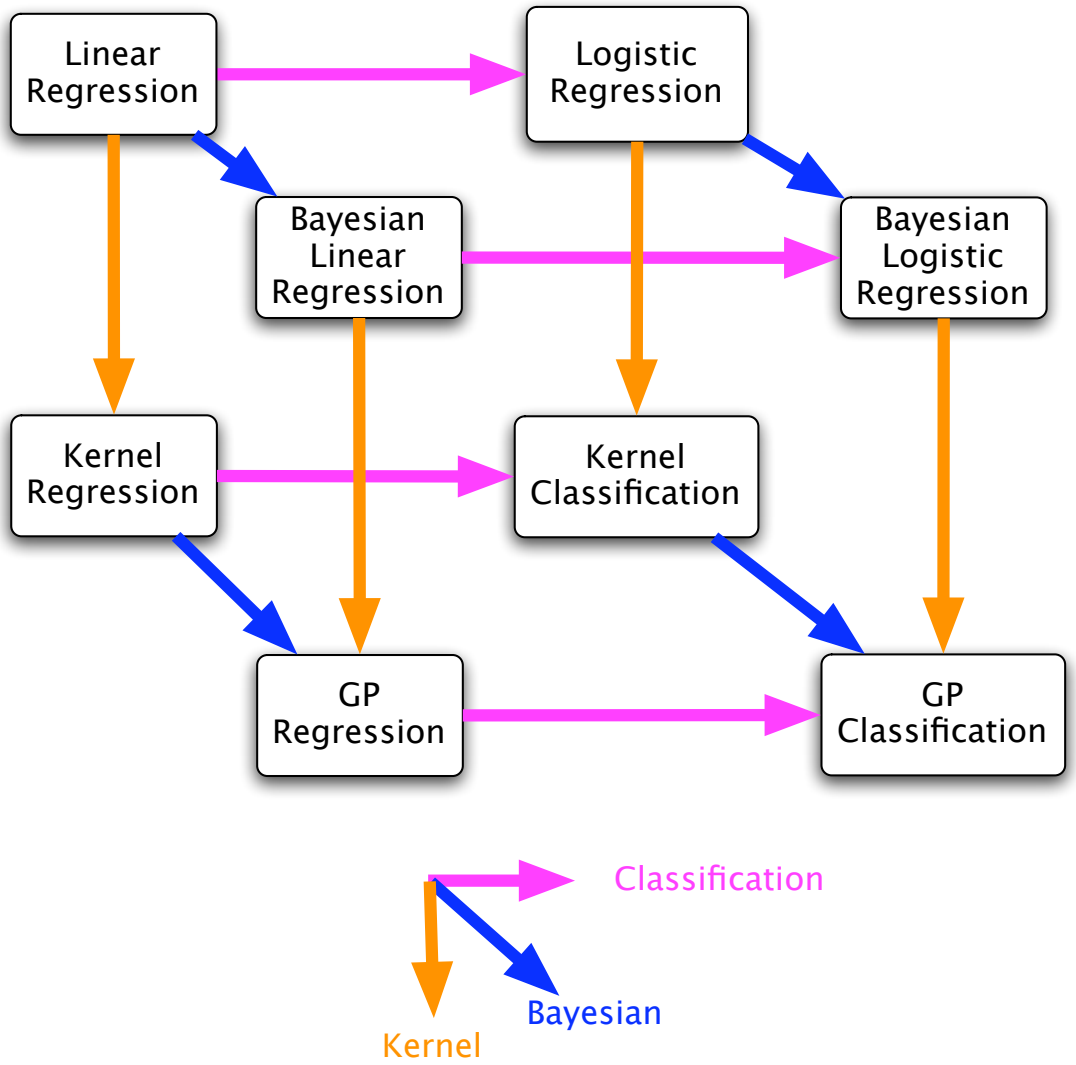
$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

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

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

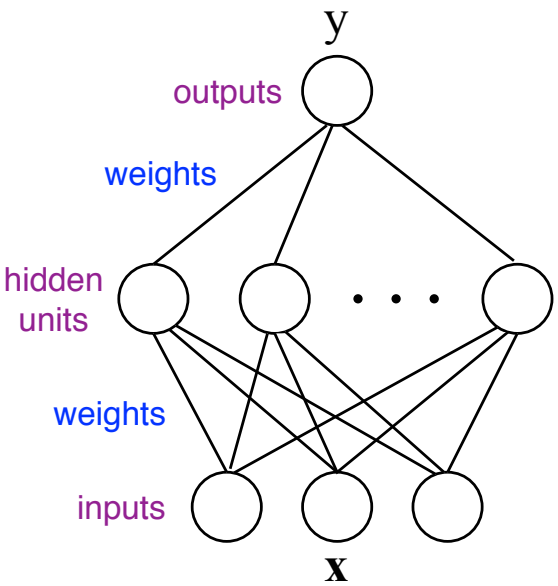
$\rightarrow$ GPflow: a Gaussian process library using TensorFlow

A PICTURE: GPS, LINEAR AND LOGISTIC REGRESSION, AND SVMS



→ GPflow: a Gaussian process library using TensorFlow

BAYESIAN NEURAL NETWORKS AND GAUSSIAN PROCESSES



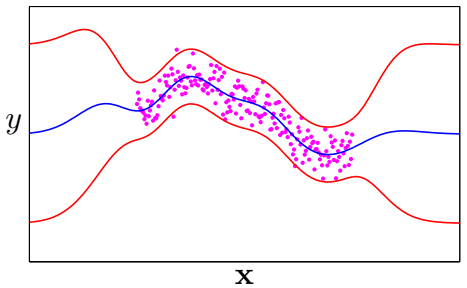
Bayesian neural network

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

Parameters $\boldsymbol{\theta}$ are weights of neural net

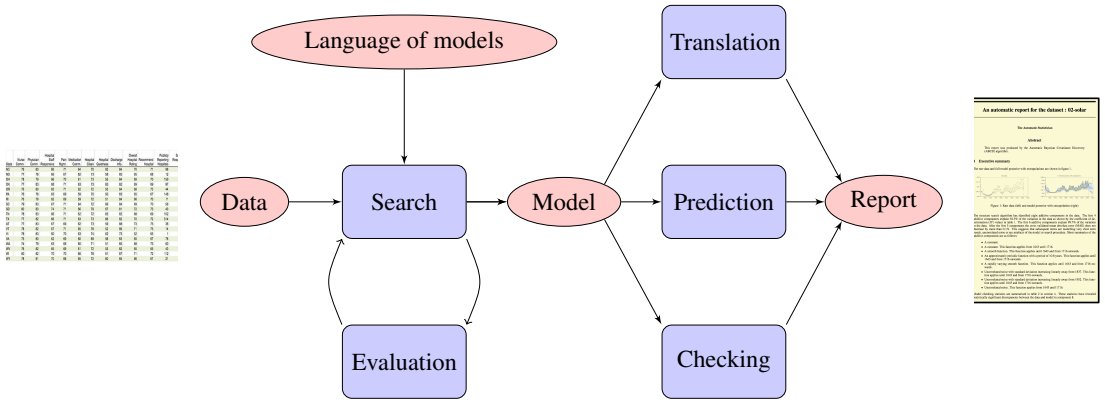
prior	$p(\boldsymbol{\theta}\|\boldsymbol{\alpha})$
posterior	$p(\boldsymbol{\theta}\|\boldsymbol{\alpha}, \mathcal{D}) \propto p(\mathbf{y}\|X, \boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\alpha})$
prediction	$p(y'\|\mathcal{D}, \mathbf{x}', \boldsymbol{\alpha}) = \int p(y'\|\mathbf{x}',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

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.



Automatic Statistician for Regression and Time-Series Models

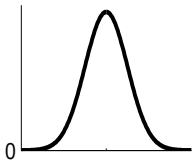
INGREDIENTS OF AN AUTOMATIC STATISTICIAN



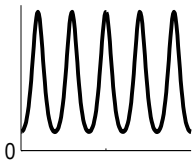
- **An open-ended language of models**
  - Expressive enough to capture real-world phenomena...
  - ...and the techniques used by human statisticians
- **A search procedure**
  - To efficiently explore the language of models
- **A principled method of evaluating models**
  - Trading off complexity and 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

THE ATOMS OF OUR LANGUAGE OF MODELS

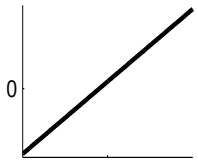
Five base kernels



Squared exp. (SE)

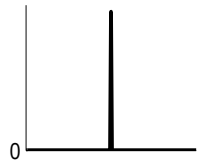


Periodic (PER)



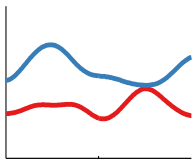
Linear (LIN)

Constant (C)

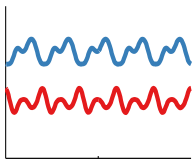


White noise (WN)

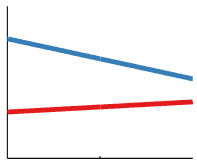
Encoding for the following types of functions



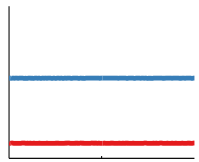
Smooth functions



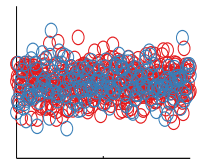
Periodic functions



Linear functions



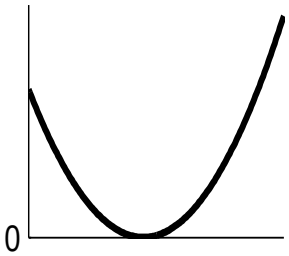
Constant functions



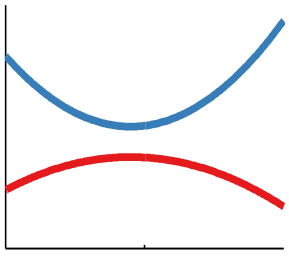
Gaussian noise

THE COMPOSITION RULES OF OUR LANGUAGE

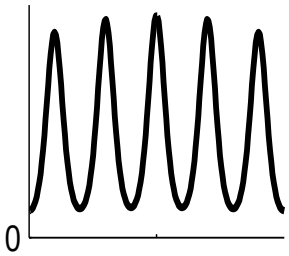
- Two main operations: addition, multiplication



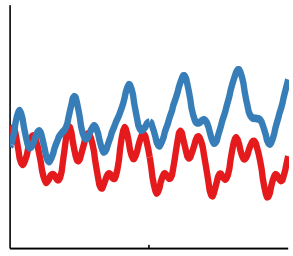
LIN × LIN



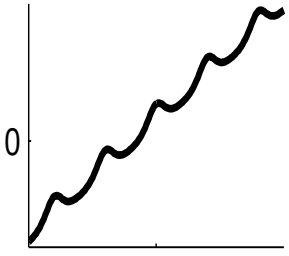
quadratic functions



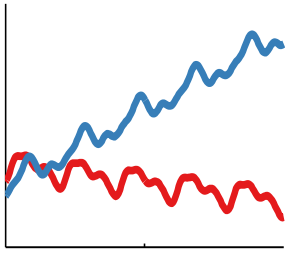
SE × PER



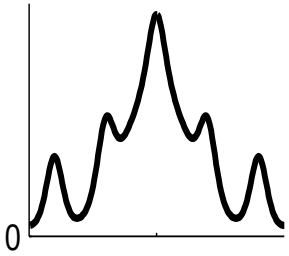
locally periodic



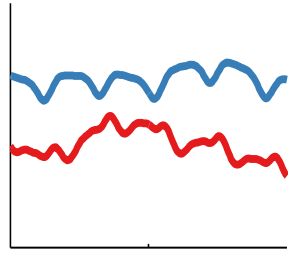
LIN + PER



periodic plus linear trend

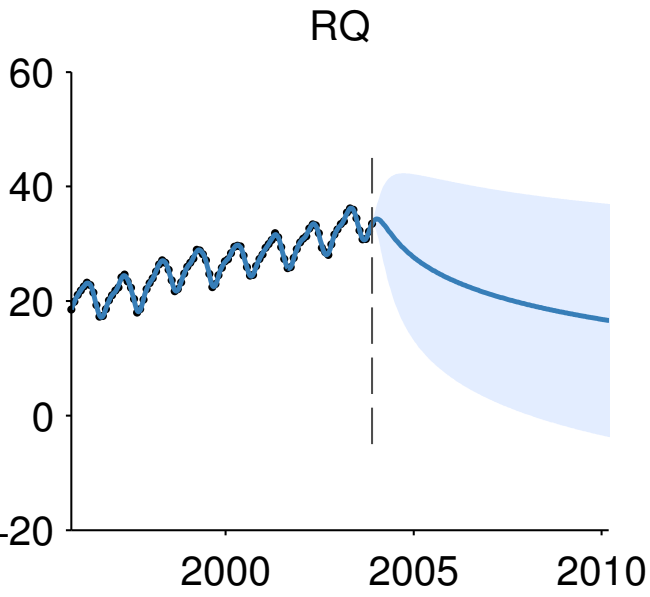


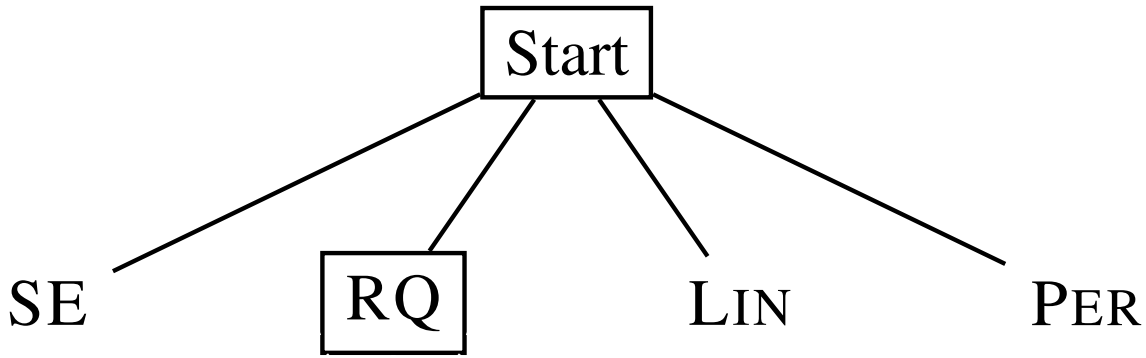
SE + PER



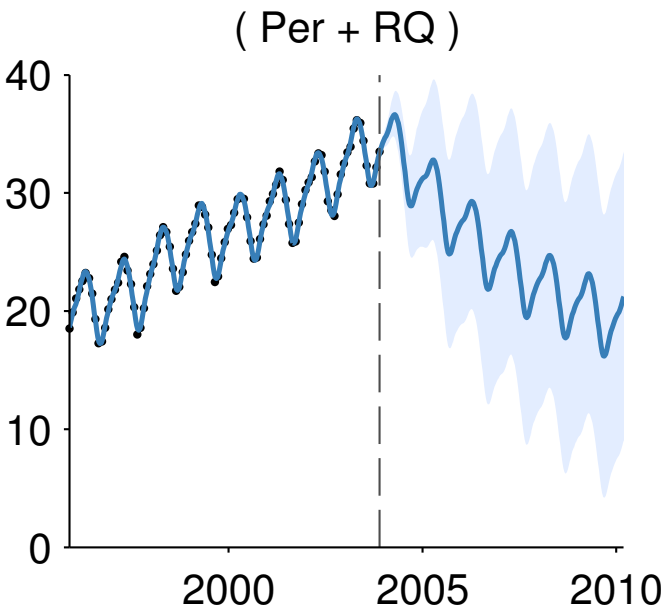
periodic plus smooth trend

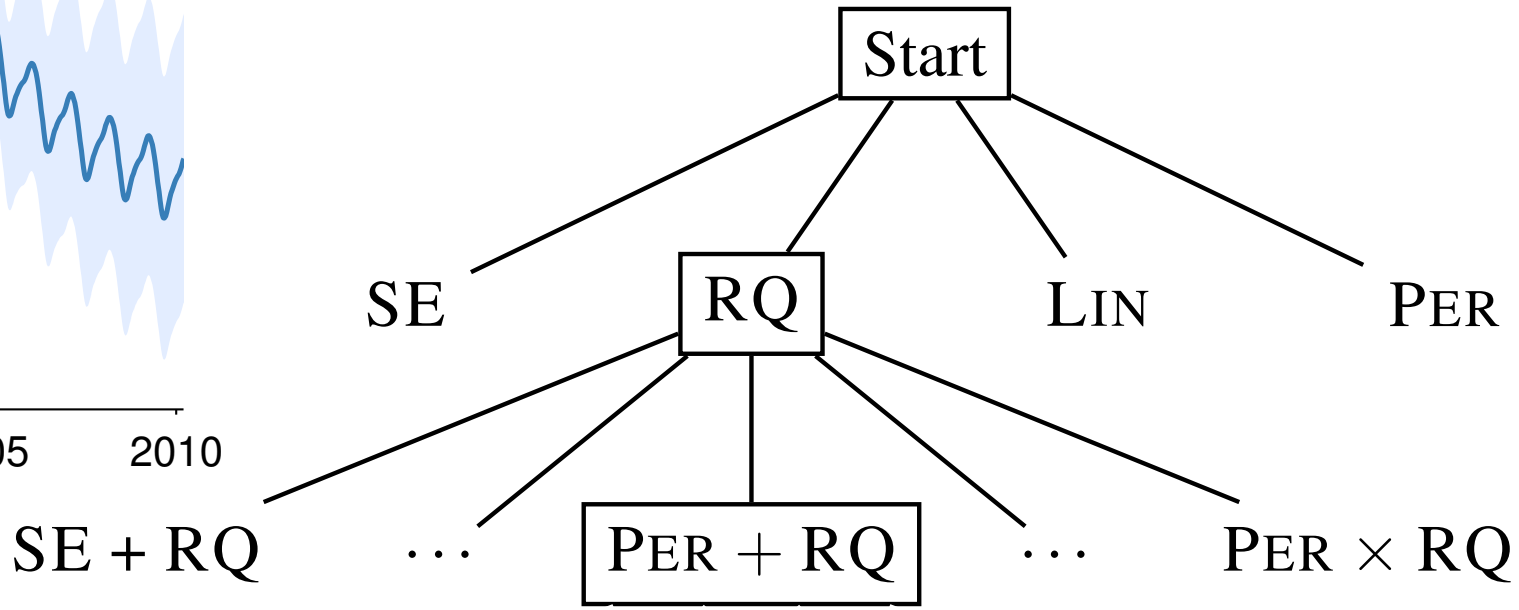
MODEL SEARCH: MAUNA LOA KEELING CURVE



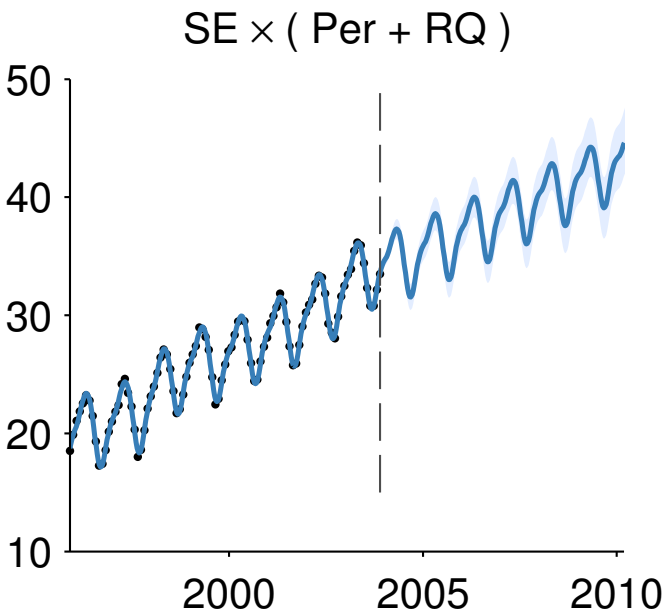


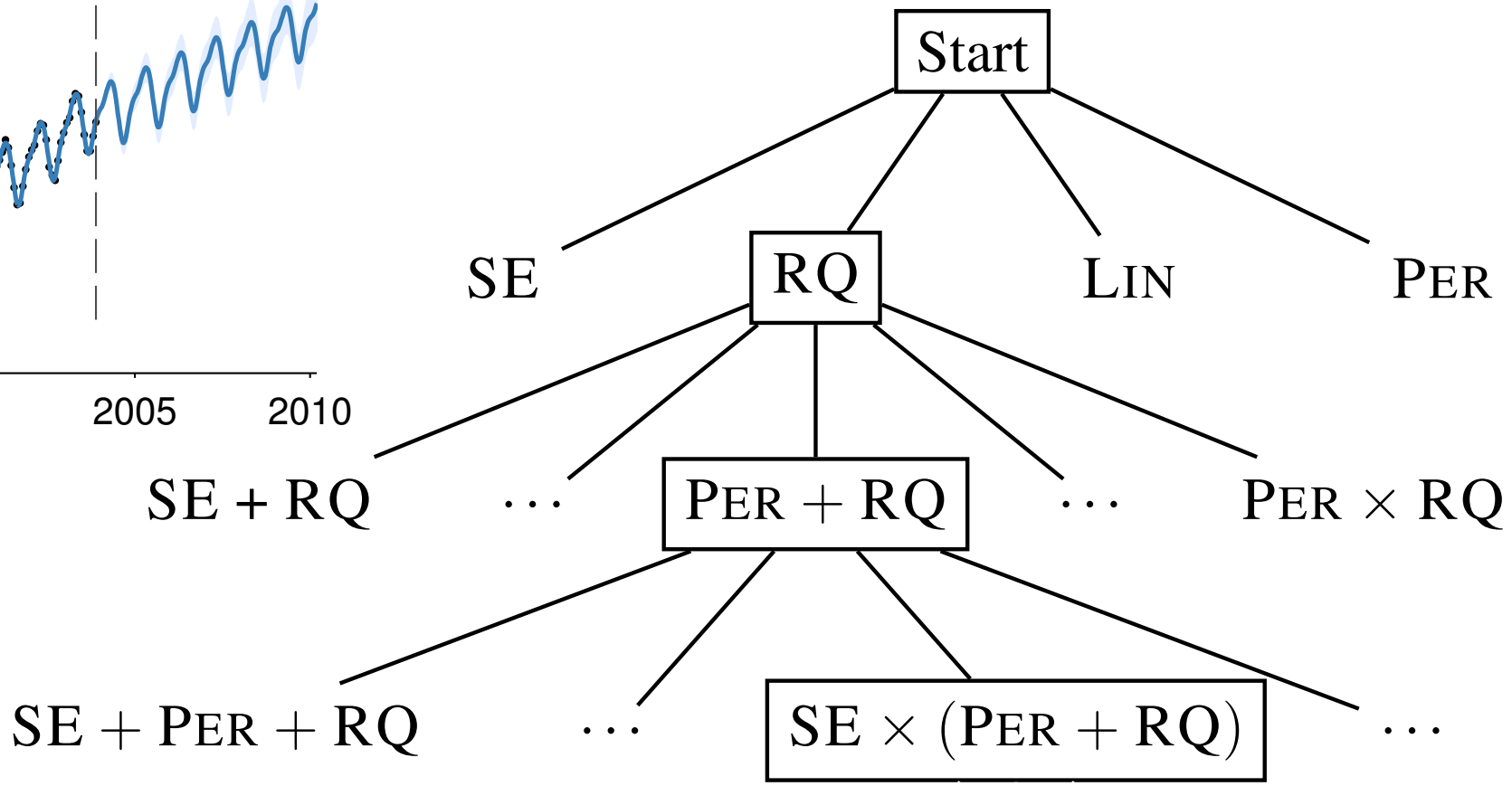
MODEL SEARCH: MAUNA LOA KEELING CURVE



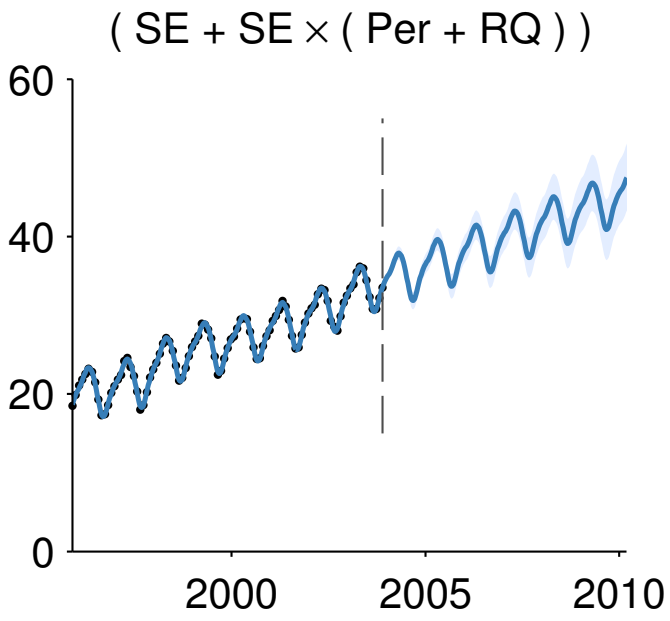


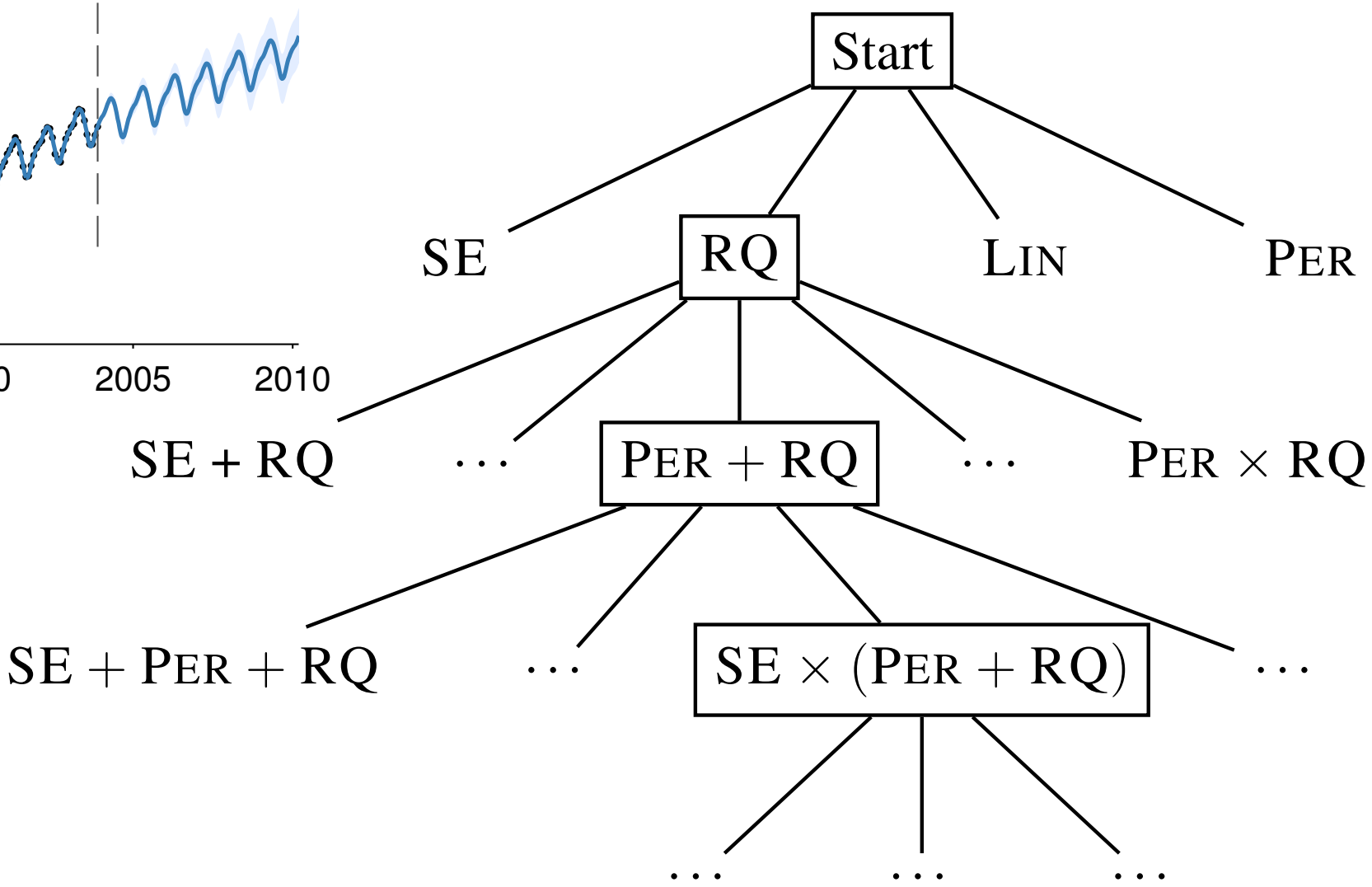
Model Search: Mauna Loa Keeling Curve



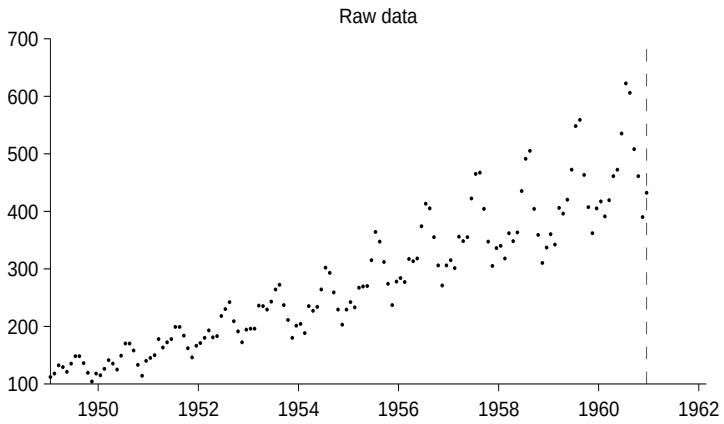


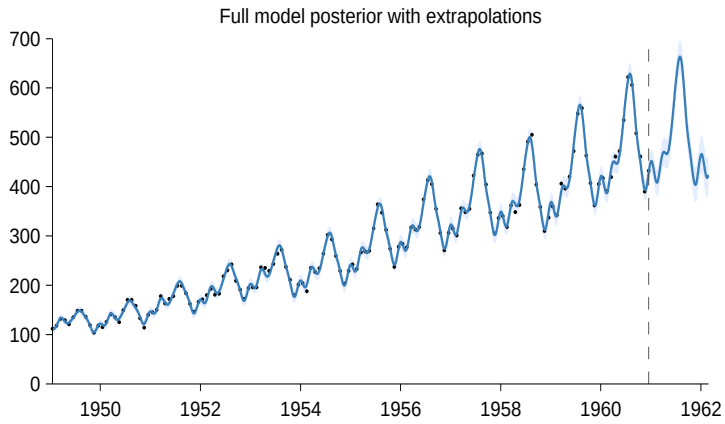
Model Search: Mauna Loa Keeling Curve





EXAMPLE: AN ENTIRELY AUTOMATIC ANALYSIS

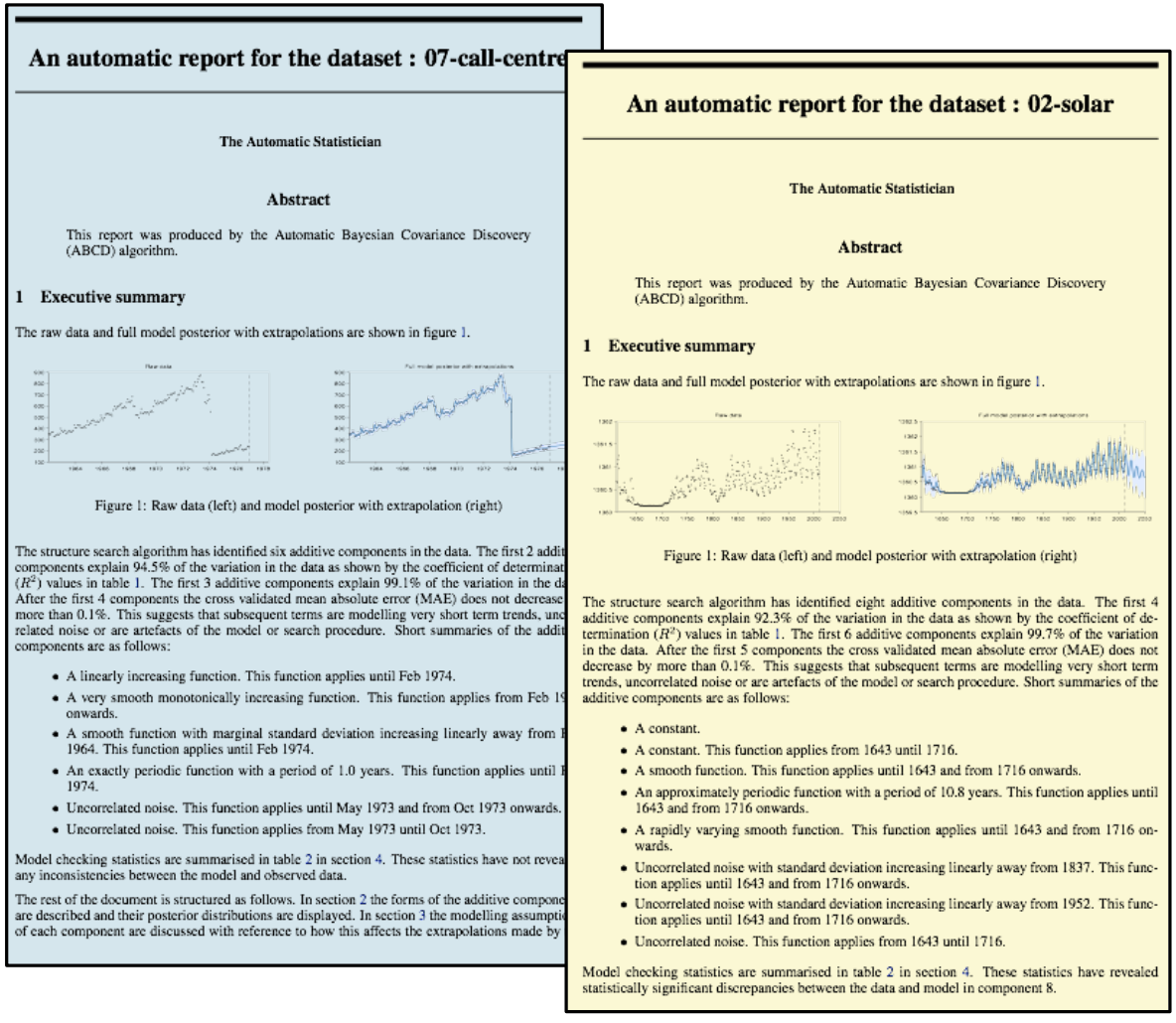




Four additive components have been identified in the data

- 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



An automatic report for the dataset : 07-call-centre

The Automatic Statistician

Abstract

This report was produced by the Automatic Bayesian Covariance Discovery (ABCD) algorithm.

1 Executive summary

The raw data and full model posterior with extrapolations are shown in figure 1.

Figure 1: Raw data (left) and model posterior with extrapolation (right)

The structure search algorithm has identified six additive components in the data. The first 2 addit components explain 94.5% of the variation in the data as shown by the coefficient of determinat (R^2) values in table 1. The first 3 additive components explain 99.1% of the variation in the da After the first 4 components the cross validated mean absolute error (MAE) does not decrease more than 0.1%. This suggests that subsequent terms are modelling very short term trends, unc related noise or are artefacts of the model or search procedure. Short summaries of the addit components are as follows:

- A linearly increasing function. This function applies until Feb 1974.
- A very smooth monotonically increasing function. This function applies from Feb 19 onwards.
- A smooth function with marginal standard deviation increasing linearly away from 1964. This function applies until Feb 1974.
- An exactly periodic function with a period of 1.0 years. This function applies until 1974.
- Uncorrelated noise. This function applies until May 1973 and from Oct 1973 onwards.
- Uncorrelated noise. This function applies from May 1973 until Oct 1973.

Model checking statistics are summarised in table 2 in section 4. These statistics have not revea any inconsistencies between the model and observed data.

The rest of the document is structured as follows. In section 2 the forms of the additive compone are described and their posterior distributions are displayed. In section 3 the modelling assumpti of each component are discussed with reference to how this affects the extrapolations made by

An automatic report for the dataset : 02-solar

The Automatic Statistician

Abstract

This report was produced by the Automatic Bayesian Covariance Discovery (ABCD) algorithm.

1 Executive summary

The raw data and full model posterior with extrapolations are shown in figure 1.

Figure 1: Raw data (left) and model posterior with extrapolation (right)

The structure search algorithm has identified eight additive components in the data. The first 4 additive components explain 92.3% of the variation in the data as shown by the coefficient of determination (R^2) values in table 1. The first 6 additive components explain 99.7% of the variation in the data. After the first 5 components the cross validated mean absolute error (MAE) does not decrease by more than 0.1%. This suggests that subsequent terms are modelling very short term trends, uncorrelated noise or are artefacts of the model or search procedure. Short summaries of the additive components are as follows:

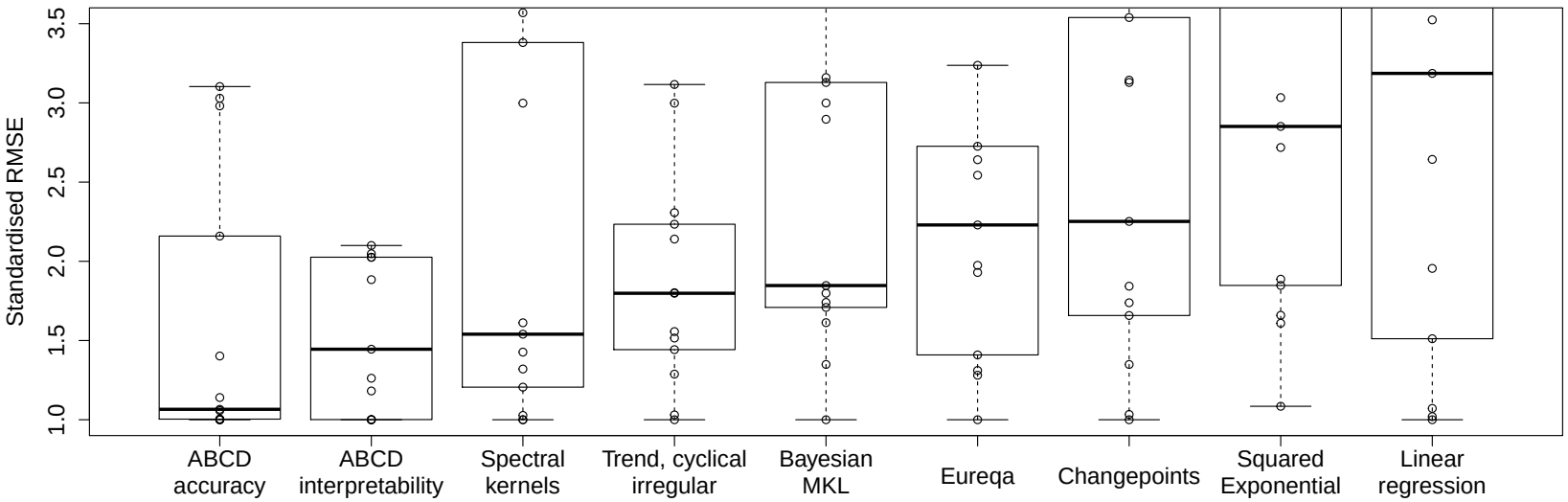
- A constant.
- A constant. This function applies from 1643 until 1716.
- A smooth function. This function applies until 1643 and from 1716 onwards.
- An approximately periodic function with a period of 10.8 years. This function applies until 1643 and from 1716 onwards.
- A rapidly varying smooth function. This function applies until 1643 and from 1716 onwards.
- Uncorrelated noise with standard deviation increasing linearly away from 1837. This function applies until 1643 and from 1716 onwards.
- Uncorrelated noise with standard deviation increasing linearly away from 1952. This function applies until 1643 and from 1716 onwards.
- Uncorrelated noise. This function applies from 1643 until 1716.

Model checking statistics are summarised in table 2 in section 4. These statistics have revealed statistically significant discrepancies between the data and model in component 8.

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[2]


AutoGPC Data Analysis Report on Pima Dataset

The Automatic Statistician

24th May 2016

This report is automatically generated by the AutoGPC. AutoGPC is an automatic Gaussian process (GP) classifier that aims to find the structure underlying a dataset without human input. For more information, please visit http://github.com/thaines2/autogpc.

The training dataset spans 4 input dimensions, which are variables 'plasma glucose', 'body mass index', 'pedigree function' and 'age'. There is one binary output variable y, where $y = 0$ (negative) represents 'not diabetic', and $y = 1$ (positive) represents 'diabetic'.

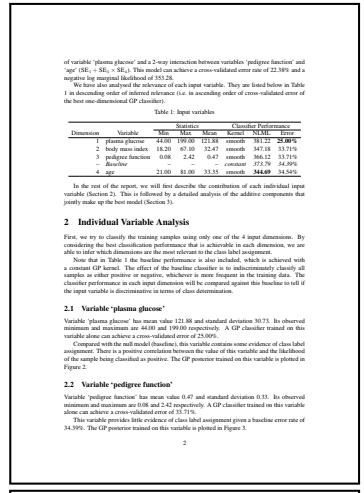
The training dataset contains 724 data points. Among them, 249 (34.39%) have positive class labels, and the other 475 (65.61%) have negative labels. All input dimensions as well as the class label assignments are plotted in Figure 1.

Figure 1: The input dataset. Positive samples ('diabetic') are coloured red, and negative ones ('not diabetic') blue.

1 Executive Summary

The best kernel that we have found relates the class label assignment to input variables 'plasma glucose', 'pedigree function' and 'age'. In specific, the model involves an additive combination

1


of variable 'plasma glucose' and a 2-way interaction between variables 'pedigree function' and 'age' ($SE_2 + SE_3 \times SE_4$). This model can achieve a cross-validated error rate of 22.58% and a negative log marginal likelihood of 353.28.

We have also analysed the relevance of each input variable. They are listed below in Table 1 in descending order of inferred relevance (i.e. in ascending order of cross-validated error of the best one-dimensional GP classifier).

Table 1: Input variables

Dimension	Variable	Statistics Min	Max	Mean	Classifier Performance Kernel	NLML	Error
1	plasma glucose	44.00	199.00	121.88	smooth	381.22	**25.00%**
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%
–	*Baseline*	–	–	–	*constant*	*373.79*	*34.39%*
4	age	21.00	81.00	33.35	smooth	**344.69**	34.54%

In the rest of the report, we will first describe the contribution of each individual input variable (Section 2). This is followed by a detailed analysis of the additive components that jointly make up the best model (Section 3).

2 Individual Variable Analysis

First, we try to classify the training samples using only one of the 4 input dimensions. By considering the best classification performance that is achievable in each dimension, we are able to infer which dimensions are the most relevant to the class label assignment.

Note that in Table 1 the baseline performance is also included, which is achieved with a constant GP kernel. The effect of the baseline classifier is to indiscriminately classify all samples as either positive or negative, whichever is more frequent in the training data. The classifier performance in each input dimension will be compared against this baseline to tell if the input variable is discriminative in terms of class determination.

2.1 Variable 'plasma glucose'

Variable 'plasma glucose' has mean value 121.88 and standard deviation 30.73. Its observed minimum and maximum are 44.00 and 199.00 respectively. A GP classifier trained on this variable alone can achieve a cross-validated error of 25.00%.

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. The GP posterior trained on this variable is plotted in Figure 2.

2.2 Variable 'pedigree function'

Variable 'pedigree function' has mean value 0.47 and standard deviation 0.33. Its observed minimum and maximum are 0.08 and 2.42 respectively. A GP classifier trained on this variable alone can achieve a cross-validated error of 33.71%.

This variable provides little evidence of class label assignment given a baseline error rate of 34.39%. The GP posterior trained on this variable is plotted in Figure 3.

2

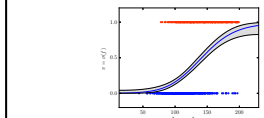


Figure 2: Trained classifier on variable 'plasma glucose'.

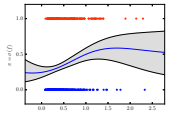


Figure 3: Trained classifier on variable 'pedigree function'.

3

2.3 Variable 'body mass index'

Variable 'body mass index' has mean value 32.47 and standard deviation 6.88. Its observed minimum and maximum are 18.20 and 67.10 respectively. A GP classifier trained on this variable alone can achieve a cross-validated error of 33.71%.

This variable provides little evidence of class label assignment given a baseline error rate of 34.39%. The GP posterior trained on this variable is plotted in Figure 4.

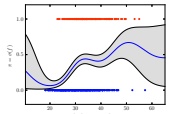


Figure 4: Trained classifier on variable 'body mass index'.

2.4 Variable 'age'

Variable 'age' has mean value 33.35 and standard deviation 11.76. Its observed minimum and maximum are 21.00 and 81.00 respectively. A GP classifier trained on this variable alone can achieve a cross-validated error of 34.54%.

The classification performance (in terms of error rate) based on this variable alone is even worse than that of the naive baseline classifier. The GP posterior trained on this variable is plotted in Figure 5.

3 Additive Component Analysis

The pattern underlying the dataset can be decomposed into 2 additive components, which contribute jointly to the final classifier which we have trained. With all components in action, the classifier can achieve a cross-validated error rate of 22.58%. The performance cannot be further improved by adding more components.

In Table 2 we list the full additive model, all input variables, as well as more complex additive components (if any) considered above, ranked by their cross-validated error rate.

4

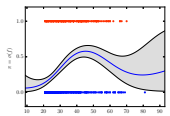


Figure 5: Trained classifier on variable 'age'.

Table 2: Classification performance of the full model, its additive components (if any), all input variables, and the baseline.

Dimensions	Kernel expression	NLML	Error
1, 3, 4	$SE_1 + SE_3 \times SE_4$	**280.45**	**21.83%**
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%
–	*C (Baseline)*	*373.79*	*34.39%*
4	SE_4	344.69	34.54%

5

3.1 Component 1

With only one additive component, the GP classifier can achieve a cross-validated error rate of 25.00%. The corresponding negative log marginal likelihood is 381.22. This component operates on variable 'plasma glucose', as shown in Figure 6.

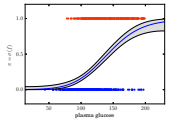
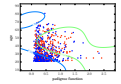


Figure 6: Trained classifier on variable 'plasma glucose'.

3.2 Component 2

With 2 additive components, the cross-validated error can be reduced by 2.62% to 22.58%. The corresponding negative log marginal likelihood is 353.28. The additional component operates on variables 'pedigree function' and 'age', as shown in Figure 7.


(a) Current additive component involving variables 'pedigree function' and 'age'.

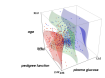

(b) Previous and current components combined, involving variables 'plasma glucose', 'pedigree function' and 'age'.

Figure 7: Trained classifier on variables 'plasma glucose', 'pedigree function' and 'age'.

6

[2] 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

		Statistics			Classifier Performance		
	Variable	Min	Max	Mean	Kernel	NLML	Error
1	plasma glucose	44.00	199.00	121.88	smooth	381.22	**25.00%**
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%
–	*Baseline*	–	–	–	*constant*	*373.79*	*34.39%*
4	age	21.00	81.00	33.35	smooth	**344.69**	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	$\mathrm{SE}_1 + \mathrm{SE}_3 \times \mathrm{SE}_4$	**280.45**	**21.83%**
1	SE_1	381.22	25.00%
3, 4	$\mathrm{SE}_3 \times \mathrm{SE}_4$	422.79	30.80%
2	SE_2	347.18	33.71%
3	SE_3	366.12	33.71%
–	C *(Baseline)*	*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:

$\rightarrow$ 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)	View
2	aad_freiburg	3.40 (2)	[illegible] (8)	[illegible] (2)	[illegible] (2)	[illegible] (4)	[illegible] (1)	5987.17 (1)	View
3	lukasz.romaszko	3.80 (3)	0.3304 (2)	0.6662 (4)	0.2647 (5)	0.3440 (3)	0.2194 (5)	711.28 (7)	View
4	matthias.vonrohr	4.40 (4)	0.3029 (5)	0.5939 (5)	0.3064 (3)	0.2994 (6)	0.2220 (3)	4964.44 (4)	View
5	marc.boulle	4.40 (4)	0.3533 (1)	0.4561 (7)	0.2130 (7)	0.3692 (1)	0.1434 (6)	4315.09 (6)	View
6	guyon	4.60 (5)	0.3031 (4)	0.5915 (6)	0.2976 (4)	0.3027 (5)	0.2202 (4)	4823.69 (5)	View
7	tadej	6.20 (6)	0.2956 (6)	0.7164 (3)	0.2616 (6)	0.1403 (8)	0.0003 (8)	5439.29 (3)	View
8	Geek	7.40 (7)	0.2550 (7)	0.0583 (8)	0.1052 (8)	0.2194 (7)	0.0068 (7)	679.30 (8)	View

- 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

Uber AI Labs is hiring!

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

PAPERS

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.

PAPERS II

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.