Probabilistic Machine Learning: Foundations and Frontiers

Zoubin Ghahramani[1,2,3,4]

[1] University of Cambridge

[2] Alan Turing Institute

[3] Leverhulme Centre for the Future of Intelligence

[4] Uber AI Labs

zoubin@eng.cam.ac.uk

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

Sackler Forum, National Academy of Sciences

Washington DC, 2017

Machine Learning

Many Related Terms

Statistical Modelling

Artificial Intelligence

Neural Networks

Machine Learning

Data Mining

Data Analytics

Deep Learning

Pattern Recognition

Data Science

Many Related Fields

Computer Science

Engineering

Statistics

Machine Learning

Applied Mathematics

Computational Neuroscience

Economics

Cognitive Science

Physics

Many Many Applications

Bioinformatics

Robotics

Computer Vision

Scientific Data Analysis

Natural Language Processing

Machine Learning

Information Retrieval

Speech Recognition

Recommender Systems

Signal Processing

Machine Translation

Medical Informatics

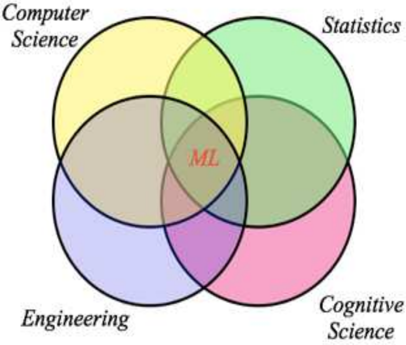
Targeted Advertising

Finance

Data Compression

Machine Learning

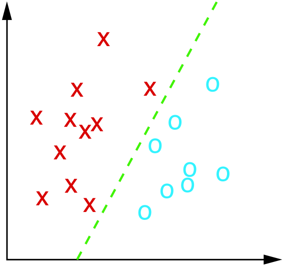
- Machine learning is an interdisciplinary field that develops both the mathematical foundations and practical applications of systems that learn from data.



Main conferences and journals: NIPS, ICML, AISTATS, UAI, KDD, JMLR, IEEE TPAMI

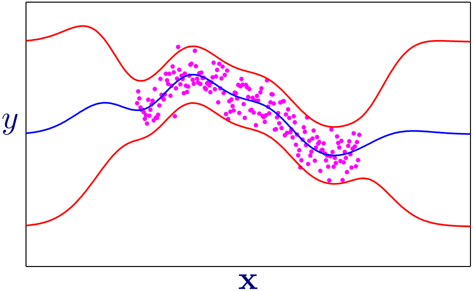
Canonical problems in machine learning

Classification



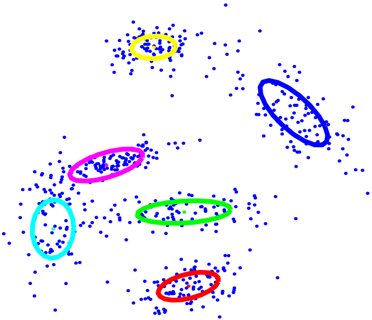
- **Task:** predict discrete class label from input data
- **Applications:** face recognition, image recognition, medical diagnosis…
- **Methods:** Logistic Regression, Support Vector Machines (SVMs), Neural Networks, Random Forests, Gaussian Process Classifiers…

Regression



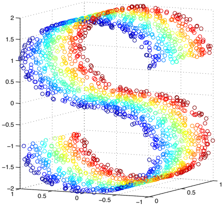
- **Task:** predict continuous quantities from input data
- **Applications:** financial forecasting, click-rate prediction, …
- **Methods:** Linear Regression, Neural Networks, Gaussian Processes, …

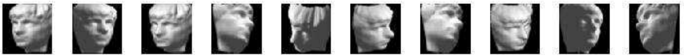
Clustering



- **Task:** group data together so that similar points are in the same group
- **Applications:** bioinformatics, astronomy, document modelling, network modelling, …
- **Methods:** k-means, Gaussian mixtures, Dirichlet process mixtures, …

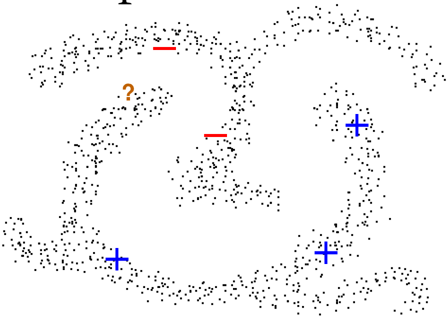
Dimensionality Reduction





- **Task:** map high-dimensional data onto low dimensions while preserving relevant information
- **Applications:** any where the raw data is high-dimensional
- **Methods:** PCA, factor analysis, MDS, LLE, Isomap, GPLVM,…

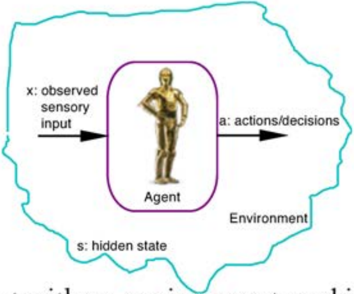
Semi-supervised Learning



- **Task:** learn from both labelled and unlabelled data
- **Applications:** any where labelling data is expensive, e.g. vision,speech…
- **Methods:** probabilistic models, graph-based SSL, transductive SVMs…

Reinforcement Learning,

Adaptive Control, and Sequential Decision Making



- **Task:** learn to interact with an environment, making sequential decisions so as to maximise future rewards
- **Applications:** robotics, control, games, trading, dialogue systems,...
- **Methods:** Q-learning, direct-policy methods, PILCO, etc...

Computer Vision:

Object, Face and Handwriting Recognition, Image Captioning

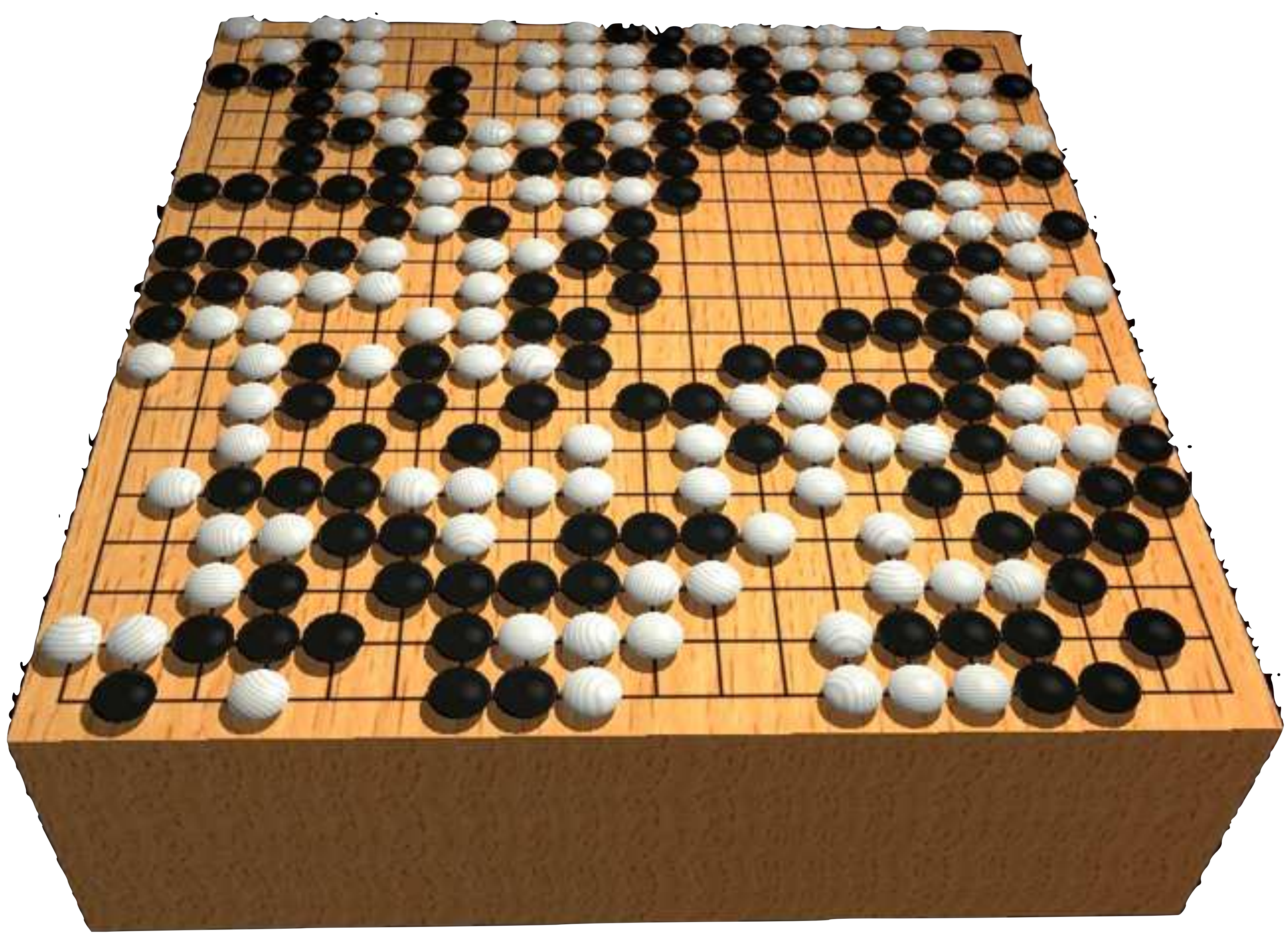
"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with legos toy."

"boy is doing backflip on wakeboard."

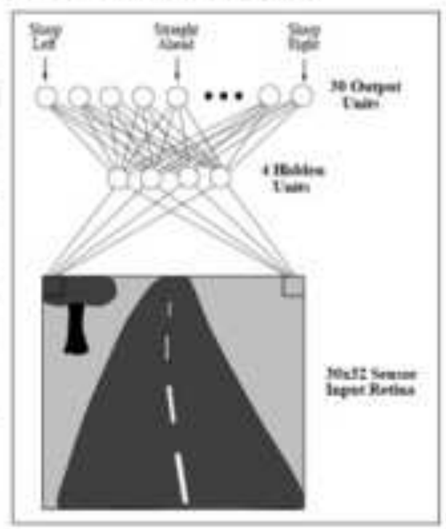
Computer Games

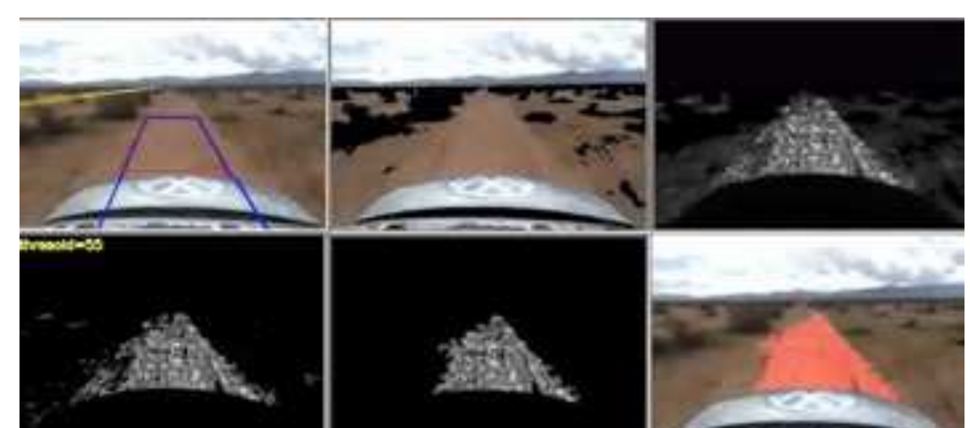


Autonomous Vehicles

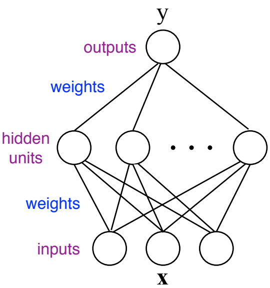
Autonomous driving

- ALVINN – Drives 70mph on highways





Neural networks and deep learning



Neural networks

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

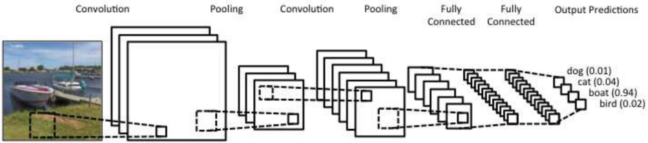
$$p(y^{(n)} = 1|x^{(n)}, \boldsymbol{\theta}) = \sigma(\sum_i \theta_i x_i^{(n)})$$

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.

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

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:

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

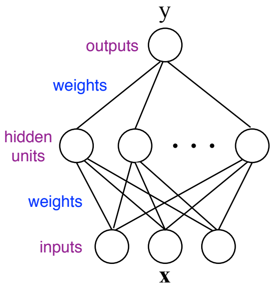
WHY SHOULD WE CARE?

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.

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)

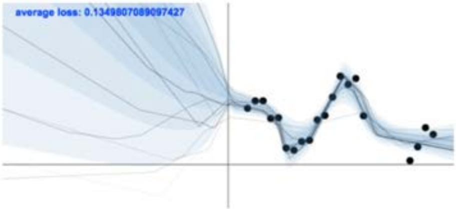


Figure from Yarin Gal's thesis "Uncertainty in Deep Learning" (2016)

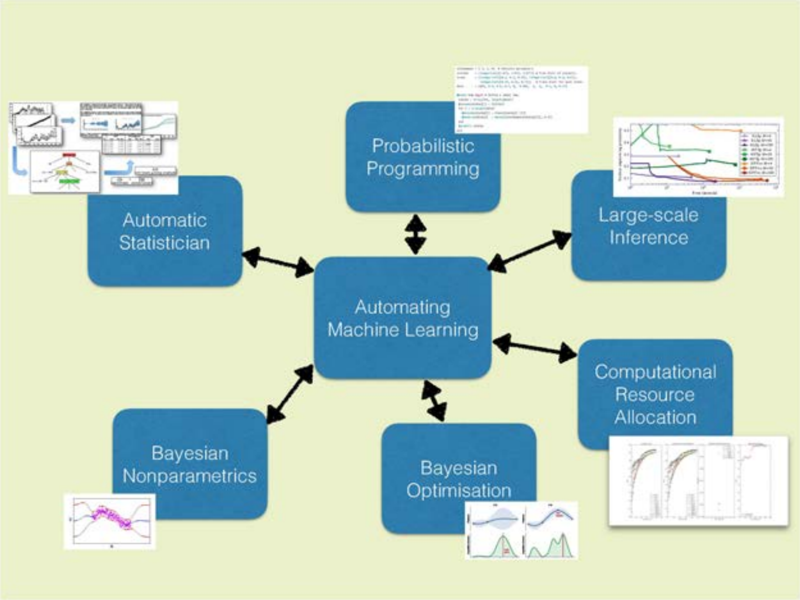
→ NIPS 2016 workshop on Bayesian Deep Learning

When do we need probabilities?

When is the probabilistic approach essential?

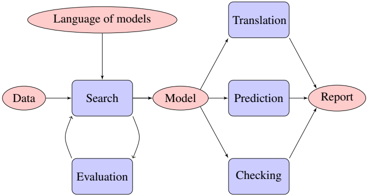
Many aspects of learning and intelligence depend crucially on the careful probabilistic representation of *uncertainty*:

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


Probabilistic Programming
Automatic Statistician
Large-scale Inference
Automating Machine Learning
Computational Resource Allocation
Bayesian Nonparametrics
Bayesian Optimisation

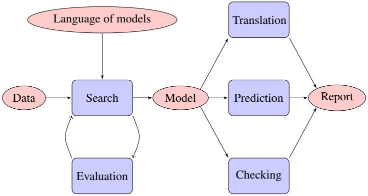
Automating model discovery: The automatic statistician

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

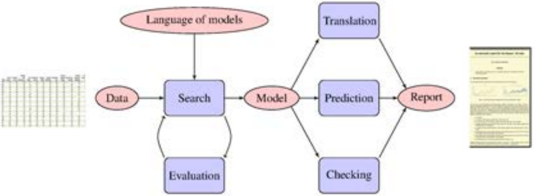


Problem: Data are now ubiquitous; there is great value from understanding this data, building models and making predictions... however, *there aren't enough data scientists, statisticians, and machine learning experts.*

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.

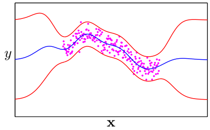
INGREDIENTS OF AN AUTOMATIC STATISTICIAN



- **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 $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$



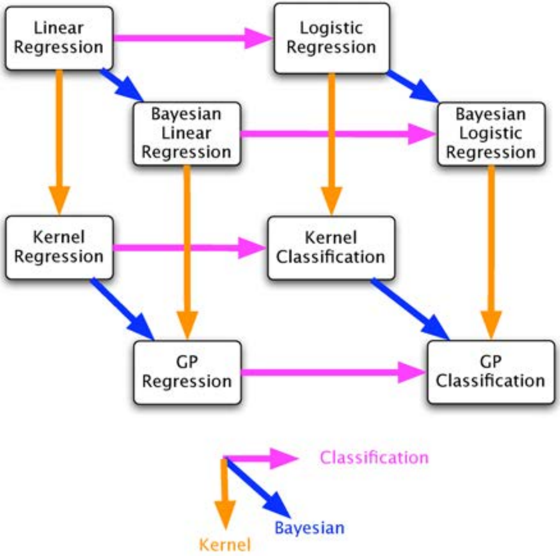
A Gaussian process defines a distribution over functions $p(f)$ which can be used for Bayesian regression:

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

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.

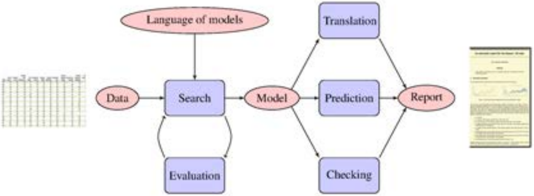
GPs can be used for regression, classification, ranking, dim. reduct...

A picture: GPs, linear and logistic regression, and SVMs



Automatic Statistician for Regression and Time-Series Models

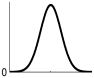
INGREDIENTS OF AN AUTOMATIC STATISTICIAN



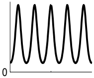
- **An open-ended language of models**
  - Expressive enough to capture real-world phenomena…
  - …and the techniques used by human statisticians
- **A search procedure**
  - To efficiently explore the language of models
- **A principled method of evaluating models**
  - Trading off complexity and fit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

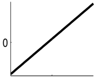
Five base kernels



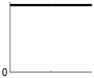
Squared exp. (SE)



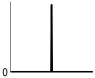
Periodic (Per)



Linear (Lin)

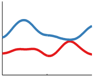


Constant (C)

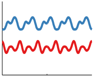


White noise (WN)

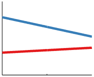
Encoding for the following types of functions



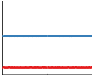
Smooth functions



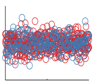
Periodic functions



Linear functions



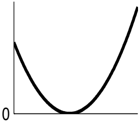
Constant functions



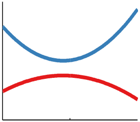
Gaussian noise

The composition rules of our language

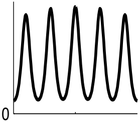
- Two main operations: addition, multiplication



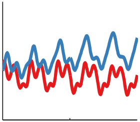
LIN $\times$ LIN



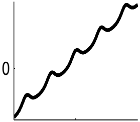
quadratic functions



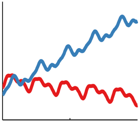
SE $\times$ PER



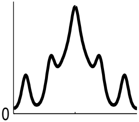
locally periodic



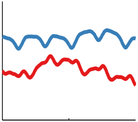
LIN $+$ PER



periodic plus linear trend

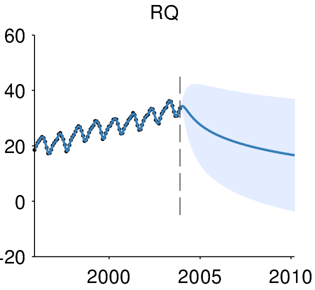


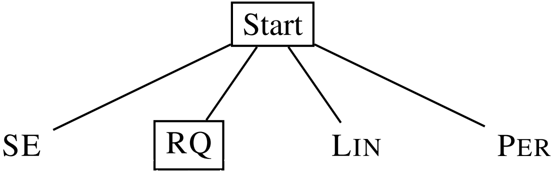
SE $+$ PER



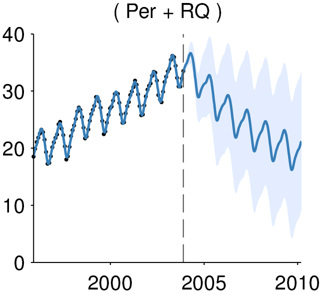
periodic plus smooth trend

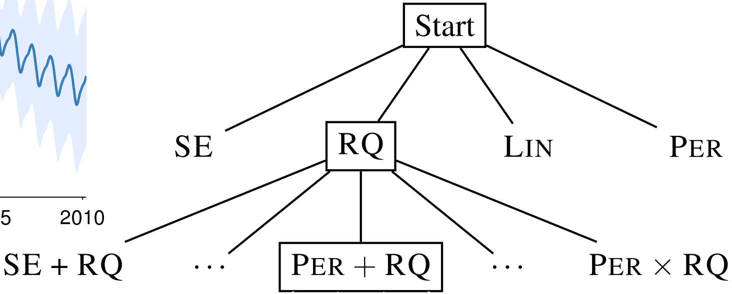
MODEL SEARCH: MAUNA LOA KEELING CURVE



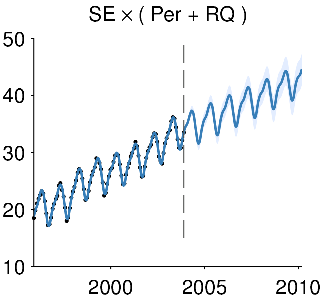


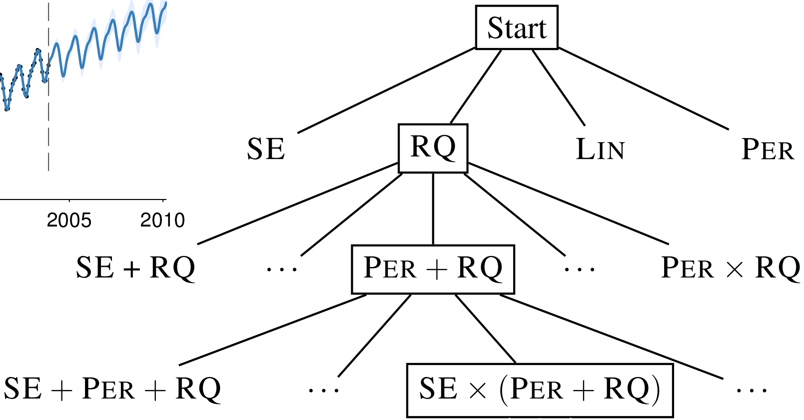
Model Search: Mauna Loa Keeling Curve



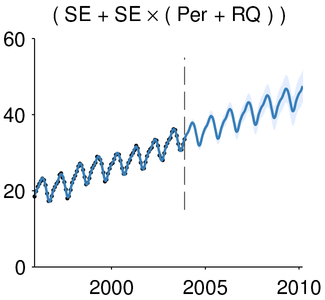


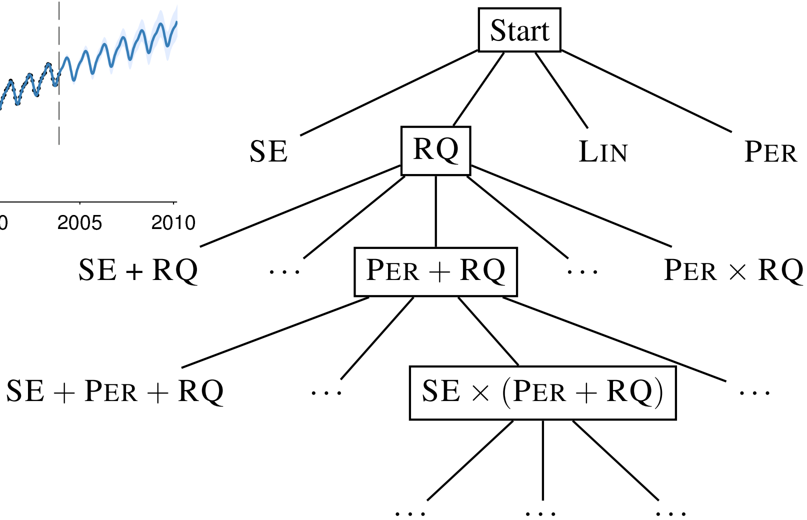
Model Search: Mauna Loa Keeling Curve



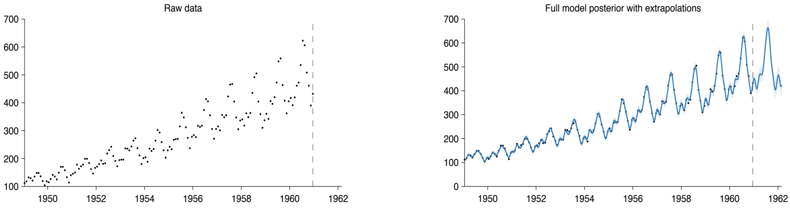


Model Search: Mauna Loa Keeling Curve





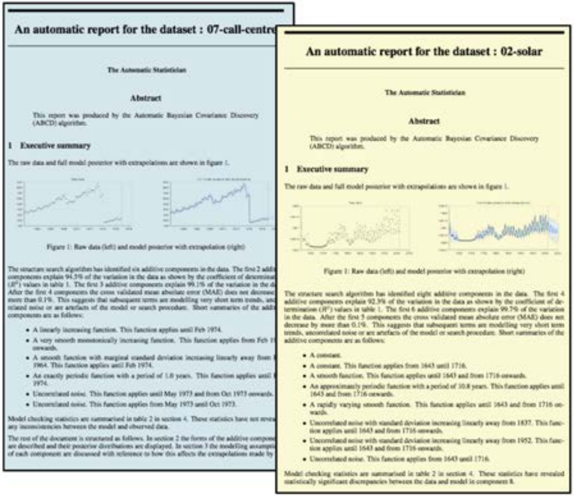
EXAMPLE: AN ENTIRELY AUTOMATIC ANALYSIS



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.

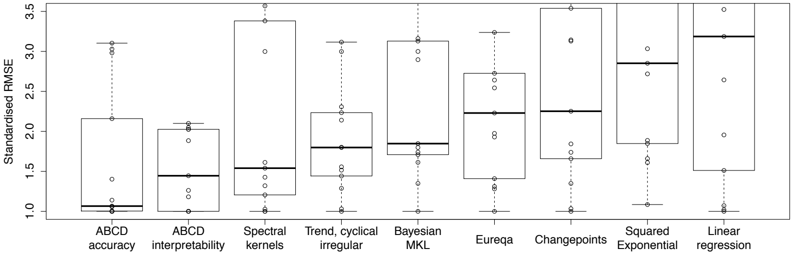
EXAMPLE REPORTS



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

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."*

Automating Inference: 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).

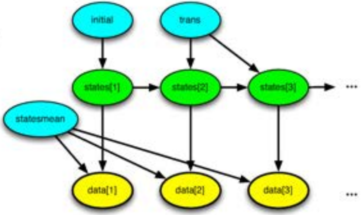
Example languages: BUGS, Infer.NET, BLOG, STAN, Church, Venture, Anglican, Probabilistic C, Stochastic Python*, Haskell*, Turing*, ...

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

PROBABILISTIC PROGRAMMING

```
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
```



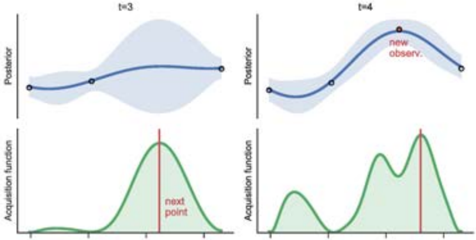
Probabilistic programming could revolutionise scientific modelling, machine learning, and AI.

→ NIPS 2015 tutorial by Frank Wood

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

Automating Optimisation: Bayesian optimisation

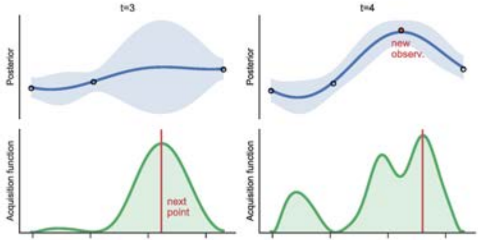
BAYESIAN OPTIMISATION



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.

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

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:

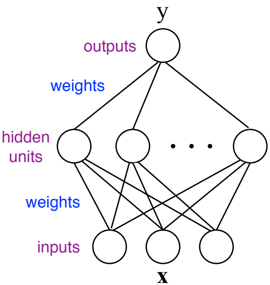
José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. ICML, 2015.

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.

BAYESIAN NEURAL NETWORKS AND GAUSSIAN PROCESSES



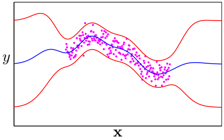
Bayesian neural network

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

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

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

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.



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.*

Bayesian Optimisation

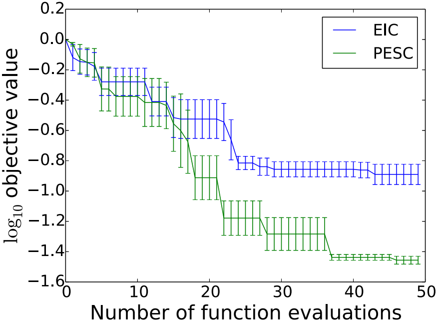


Figure 4. Classification error of a 3-hidden-layer neural network constrained to make predictions in under 2 ms.

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