A history of Bayesian neural networks

Zoubin Ghahramani[*†‡]

[*]University of Cambridge, UK

[†]Alan Turing Institute, London, UK

[‡]Uber AI Labs, USA

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

NIPS 2016 Bayesian Deep Learning

DEDICATION

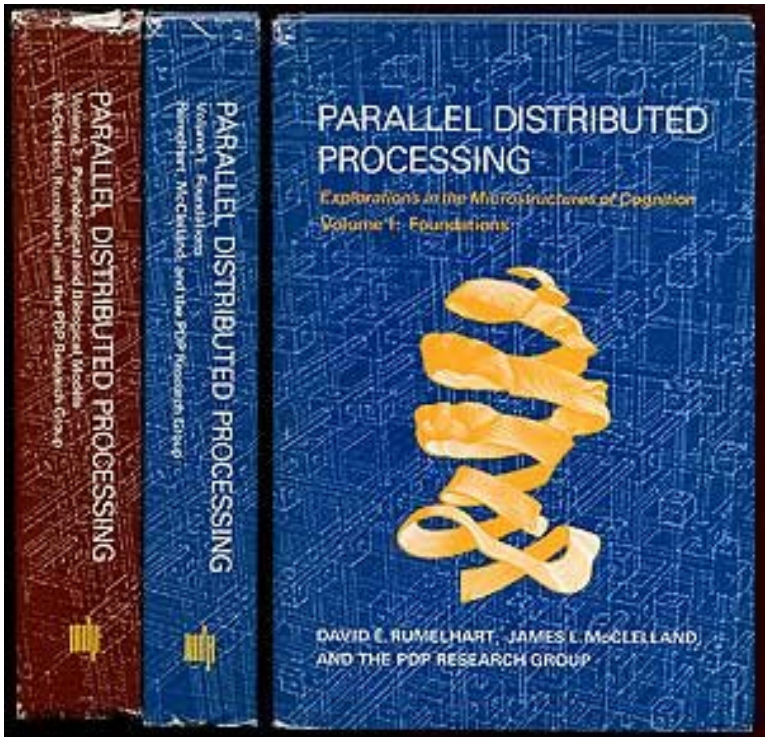
to my friend and colleague David MacKay:

I'm a NIPS old-timer, apparently...

...so now I give talks about history.

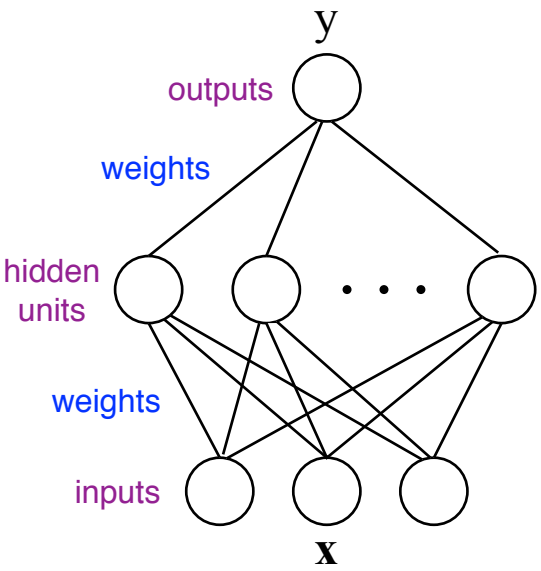
BACK IN THE 1980S

There was a huge wave of excitement when Boltzmann machines were published in 1985, the backprop paper came out in 1986, and the PDP volumes appeared in 1987.



This field also used to be called *Connectionism* and NIPS was its main conference (launched in 1987).

WHAT IS A NEURAL NETWORK?



Neural network is a parameterized function

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

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

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

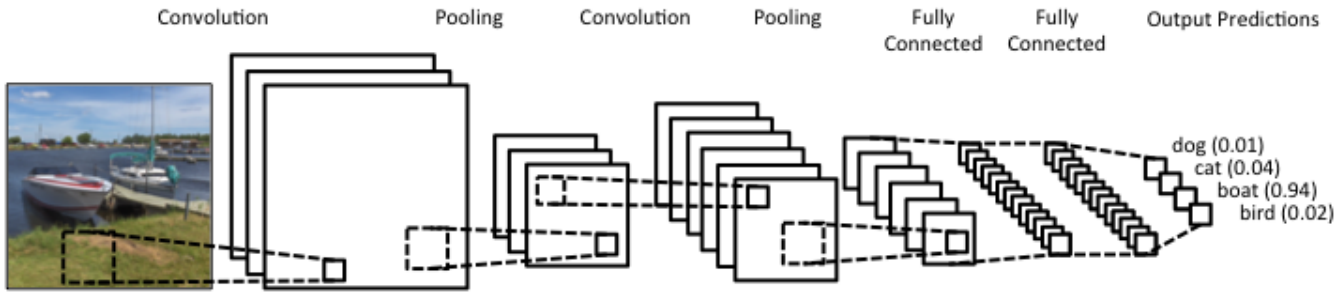
$$p(y^{(n)} = 1|x^{(n)}, \boldsymbol{\theta}) = \sigma(\sum_i \theta_i x_i^{(n)})$$

Multilayer / deep 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, better initialisation and learning rates, 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.andrewkurenkov.com/

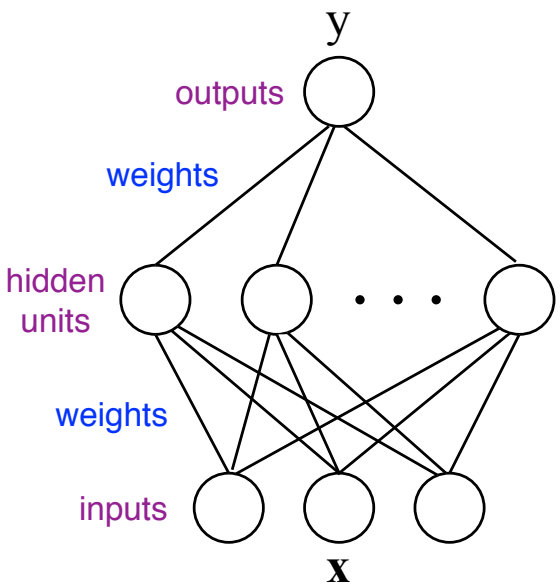
LIMITATIONS OF DEEP LEARNING

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

WHAT DO I MEAN BY BEING BAYESIAN?

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.

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)

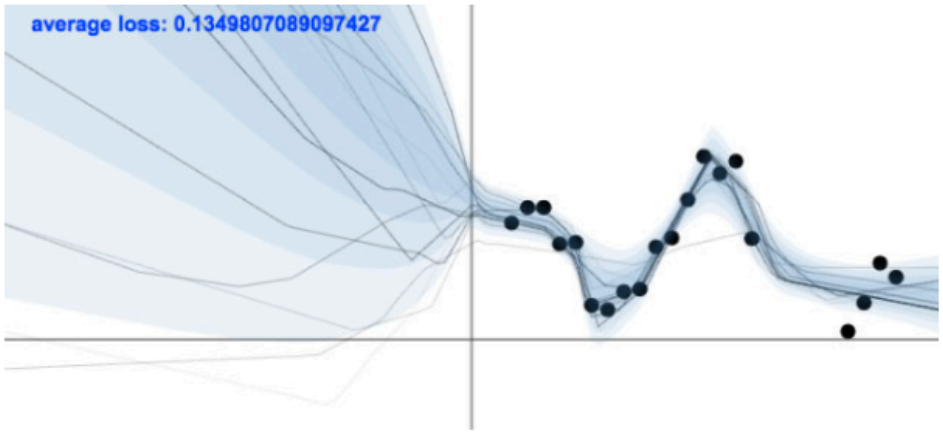


Figure from Yarin Gal's thesis "Uncertainty in Deep Learning" (2016)

A NOTE ON MODELS VS ALGORITHMS

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

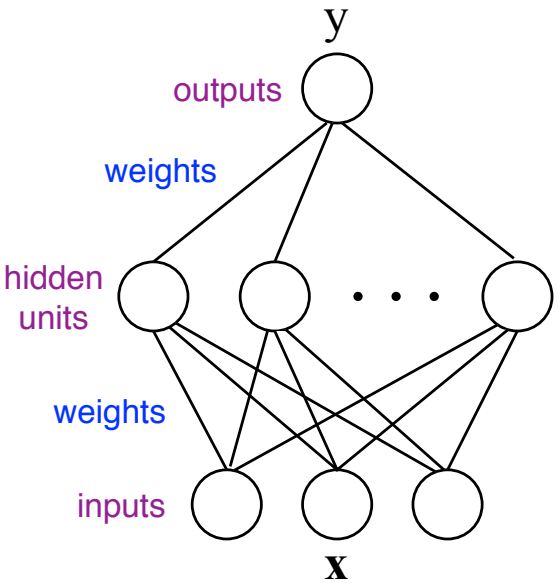
Models:	**Algorithms**
convnets	Stochastic Gradient Descent
LDA	Conjugate-gradients
RNNs	MCMC
HMMs	Variational Bayes and SVI
Boltzmann machines	SGLD
State-space models	Belief propagation, EP
Gaussian processes	...

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 NEURAL NETWORKS



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 $\quad p(\boldsymbol{\theta}|\boldsymbol{\alpha})$

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

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

EARLY HISTORY OF BAYESIAN NEURAL NETWOKS

Complex Systems 1 (1987) 877–922

Large Automatic Learning, Rule Extraction, and Generalization

John Denker
Daniel Schwartz
Ben Wittner
Sara Solla
Richard Howard
Lawrence Jackel
AT&T Bell Laboratories, Holmdel, NJ 07733, USA

John Hopfield
AT&T Bell Laboratories, Murray Hill, NJ 07974, USA
and
California Institute of Technology, Pasadena, CA 91125, USA

We remind the reader that one is *not* allowed to search $\widehat{W}$ space to find the "correct" rule extracting network. That cannot be done without using data from the testing set X, which defeats the purpose, by definition. That would be like betting on the winning horse after the race is over. We are only allowed to play the probabilities in $\widehat{W}$ space.

14.2 Derivation

The task of choosing a probability distribution in W space is a bit tricky. The choice depends on just what method is used for "learning", i.e. for searching W space. Fortunately, the exact form of the distribution is not important for our argument. You could, for instance, use a probability density proportional to $e^{|W|/\omega}$, for some "radius" ω. We will for most purposes use a distribution that is uniform inside a hypercubical volume (ω on a side) and zero elsewhere. We choose ω to be big enough to enclose reasonable weight values, but not too much bigger than that.

We can map weight space onto function space as follows: for each configuration of weights, W, build a network with those weights. Present it all possible binary inputs. Observe the corresponding outputs, and convert to binary. This mapping associates a definite truth table, i.e. a definite Boolean function, with each point in W space. To say it the other way, the inverse image of a function is a region in weight space.

By integrating over weight space, we can assign a probability P_i to each function. If ω is large enough, and if there are enough hidden units ($H \propto 2^N$), there will be non-zero probability assigned to every function, according to the discussion in section 5. On the other hand, we are particularly interested

p. 904 hints at Bayesian integration over network parameters

- John Denker, Daniel Schwartz, Ben Wittner, Sara Solla, Richard Howard, Lawrence Jackel, and John Hopfield. Large automatic learning, rule extraction, and generalization. Complex Systems, 1(5):877-922, 1987.

EARLY HISTORY OF BAYESIAN NEURAL NETWOKS

- Naftali Tishby, Esther Levin, and Sara A Solla. Consistent inference of probabilities in layered networks: Predictions and generalizations. In IJCNN, 1989.

The conditional distribution (7) can now be inverted to induce a distribution on the network configuration space, $\boldsymbol{W}$, given the set of input-output pairs $\mathbf{x}^{(m)}$, using Bayes formula

$$\rho^{(m)}(\omega) \equiv P(\omega \mid \mathbf{x}^{(m)}) = \frac{\rho^{(0)}(\omega)\, P(\mathbf{x}^{(m)} \mid \omega)}{\int_{W} d\omega\, \rho^{(0)}(\omega)\, P(\mathbf{x}^{(m)} \mid \omega)}, \qquad (8)$$

where $\rho^{(0)}$ is a nonsingular *prior* distribution on the configuration space.

4. Example: architecture selection for the contiguity problem

To demonstrate the utility of the average prediction error for determining a sufficient size of the training set, as well as selecting the optimal architecture of the network, we focus on a

EARLY HISTORY OF BAYESIAN NEURAL NETWOKS

- John Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In NIPS 3, 1991.

weight configuration is an exponential function of the loss (Tishby, Levin and Solla, 1989). Therefore the probability can be modelled locally as a multidimensional gaussian centered at $\bar{W}$; to a reasonable (Denker and leCun, 1990) approximation the probability is proportional to:

$$\rho_m(W) = \rho_0(W) \exp[-\beta \sum_i h_{ii}(W_i - \bar{W}_i)^2/2] \quad (1)$$

where h is the second derivative of the loss (the Hessian), β is a scale factor that determines our overall confidence in the training data, and ρ_0 expresses any information we have about prior probabilities. The sums run over the dimensions of parameter space. The width of this gaussian describes the range of networks in the ensemble that are reasonably consistent with the training data.

- Wray L Buntine and Andreas S Weigend. Bayesian back-propagation. Complex Systems, 5(6):603-643, 1991.

GOLDEN ERA OF BAYESIAN NEURAL NETWOKS

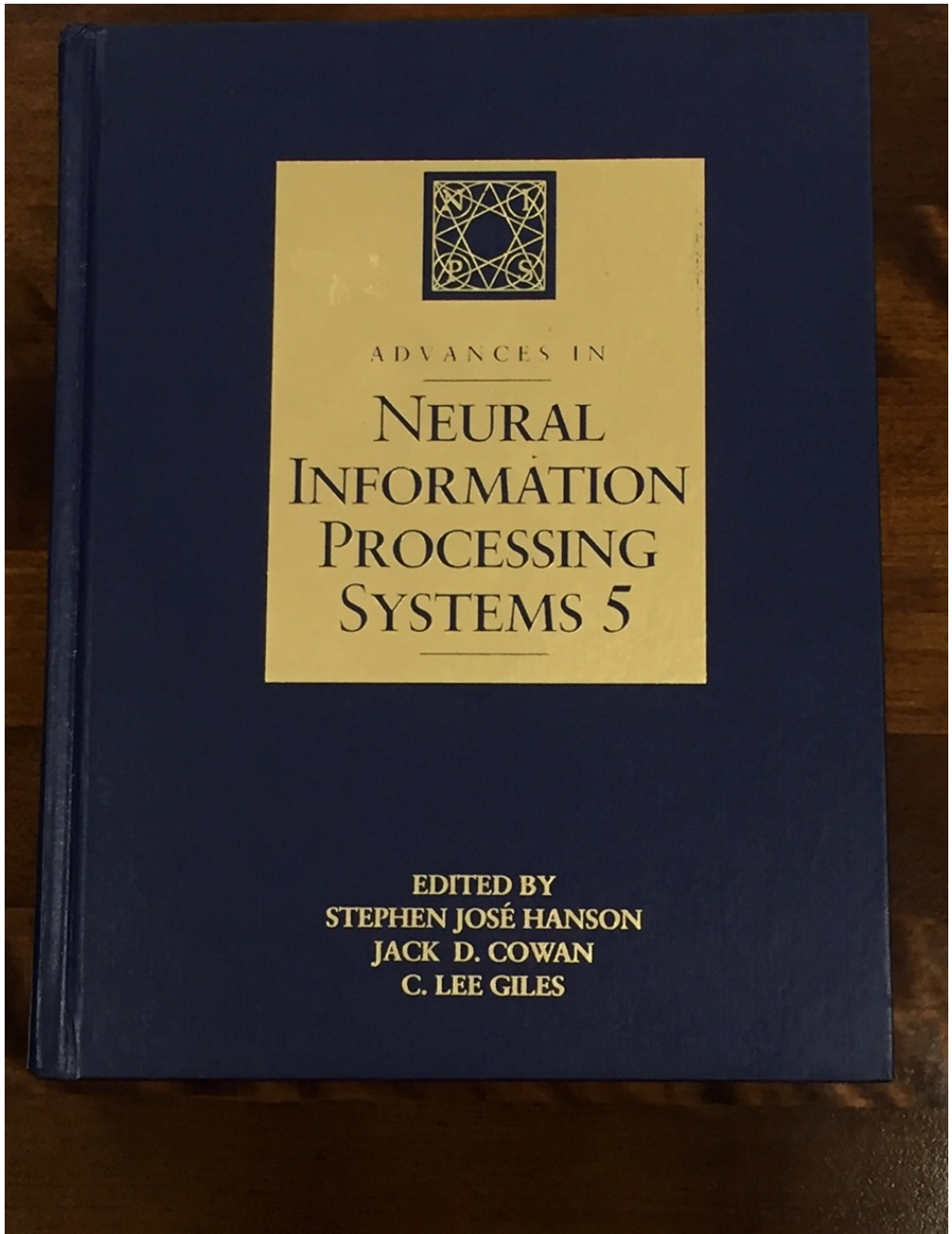
- David JC MacKay. *Neural Computation*, 4(3):448-472, 1992:

A Practical Bayesian Framework for Backpropagation Networks

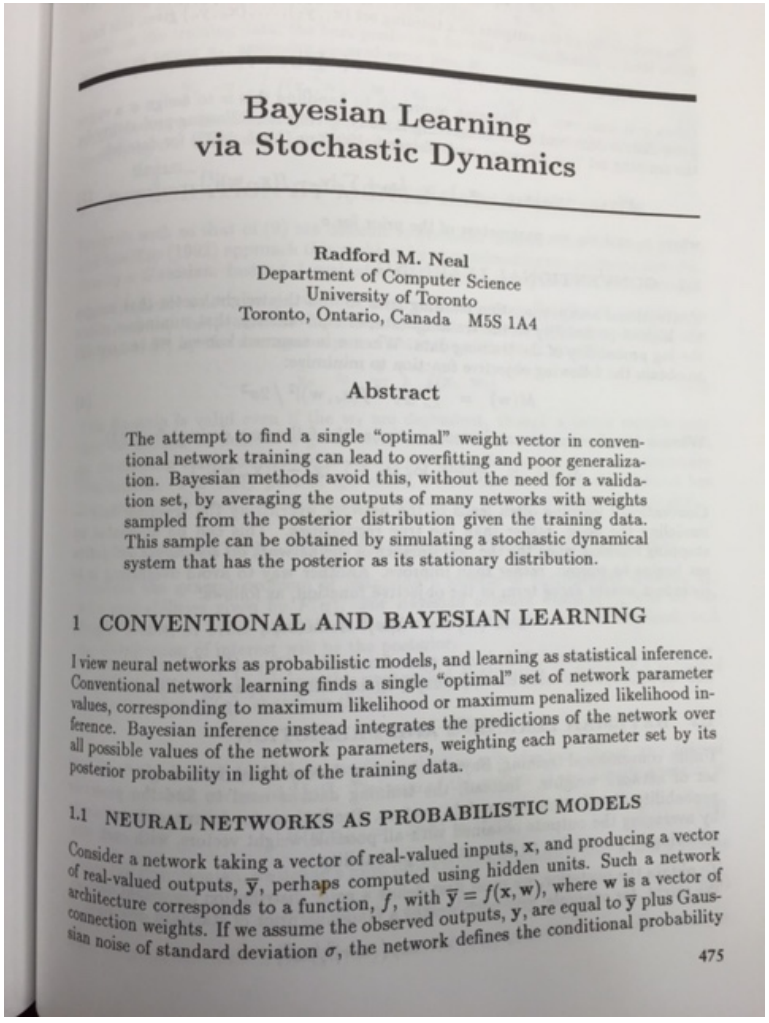
David J. C. MacKay*
Computation and Neural Systems, California Institute of Technology 139-74, Pasadena, CA 91125 USA

A quantitative and practical Bayesian framework is described for learning of mappings in feedforward networks. The framework makes possible (1) objective comparisons between solutions using alternative network architectures, (2) objective stopping rules for network pruning or growing procedures, (3) objective choice of magnitude and type of weight decay terms or additive regularizers (for penalizing large weights, etc.), (4) a measure of the effective number of well-determined parameters in a model, (5) quantified estimates of the error bars on network parameters and on network output, and (6) objective comparisons with alternative learning and interpolation models such as splines and radial basis functions. The Bayesian "evidence" automatically embodies "Occam's razor," penalizing overflexible and overcomplex models. The Bayesian approach helps detect poor underlying assumptions in learning models. For learning models well matched to a problem, a good correlation between generalization ability and the Bayesian evidence is obtained.

GOLDEN ERA OF BAYESIAN NEURAL NETWOKS



GOLDEN ERA OF BAYESIAN NEURAL NETWORKS



Bayesian Learning
via Stochastic Dynamics

Radford M. Neal
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4

Abstract

The attempt to find a single "optimal" weight vector in conventional network training can lead to overfitting and poor generalization. Bayesian methods avoid this, without the need for a validation set, by averaging the outputs of many networks with weights sampled from the posterior distribution given the training data. This sample can be obtained by simulating a stochastic dynamical system that has the posterior as its stationary distribution.

1 CONVENTIONAL AND BAYESIAN LEARNING

I view neural networks as probabilistic models, and learning as statistical inference. Conventional network learning finds a single "optimal" set of network parameter values, corresponding to maximum likelihood or maximum penalized likelihood inference. Bayesian inference instead integrates the predictions of the network over all possible values of the network parameters, weighting each parameter set by its posterior probability in light of the training data.

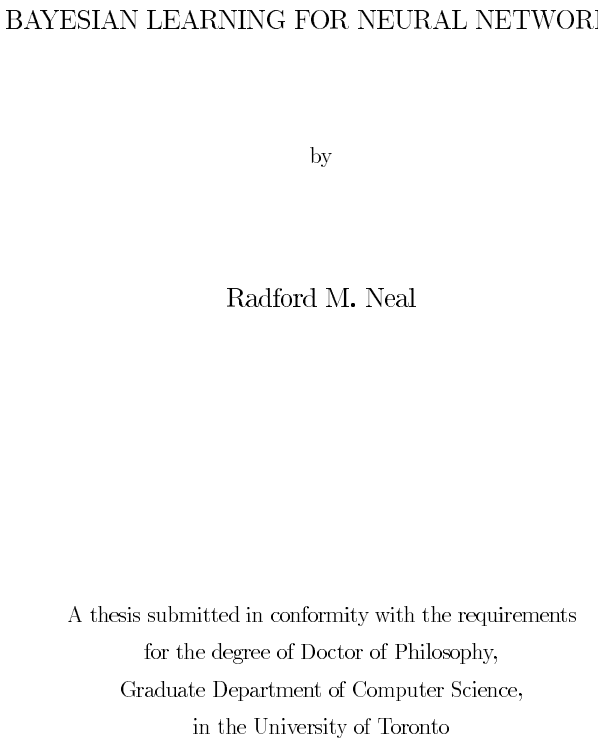
1.1 NEURAL NETWORKS AS PROBABILISTIC MODELS

Consider a network taking a vector of real-valued inputs, $\mathbf{x}$, and producing a vector of real-valued outputs, $\overline{\mathbf{y}}$, perhaps computed using hidden units. Such a network architecture corresponds to a function, f, with $\overline{\mathbf{y}} = f(\mathbf{x}, \mathbf{w})$, where $\mathbf{w}$ is a vector of connection weights. If we assume the observed outputs, $\mathbf{y}$, are equal to $\overline{\mathbf{y}}$ plus Gaussian noise of standard deviation σ, the network defines the conditional probability

475

- Neal, R.M. Bayesian learning via stochastic dynamics. In NIPS 1993.
  First Markov Chain Monte Carlo (MCMC) sampling algorithm for Bayesian neural networks. Uses Hamiltonian Monte Carlo (HMC), a sophisticated MCMC algorithm that makes use of *gradients* to sample efficiently.

Golden Era of Bayesian Neural Networks


BAYESIAN LEARNING FOR NEURAL NETWORKS

by

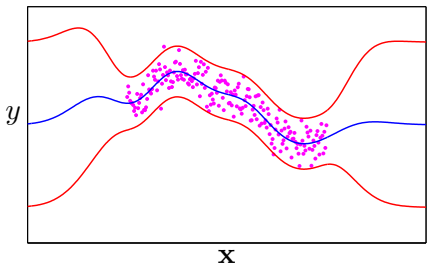
Radford M. Neal

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy,
Graduate Department of Computer Science,
in the University of Toronto

- Neal, R.M. Bayesian learning for neural networks. PhD thesis, University of Toronto, 1995. ... thesis also establishes link between BNNs and Gaussian processes and describes ARD (automatic relevance determination).

GAUSSIAN PROCESSES

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



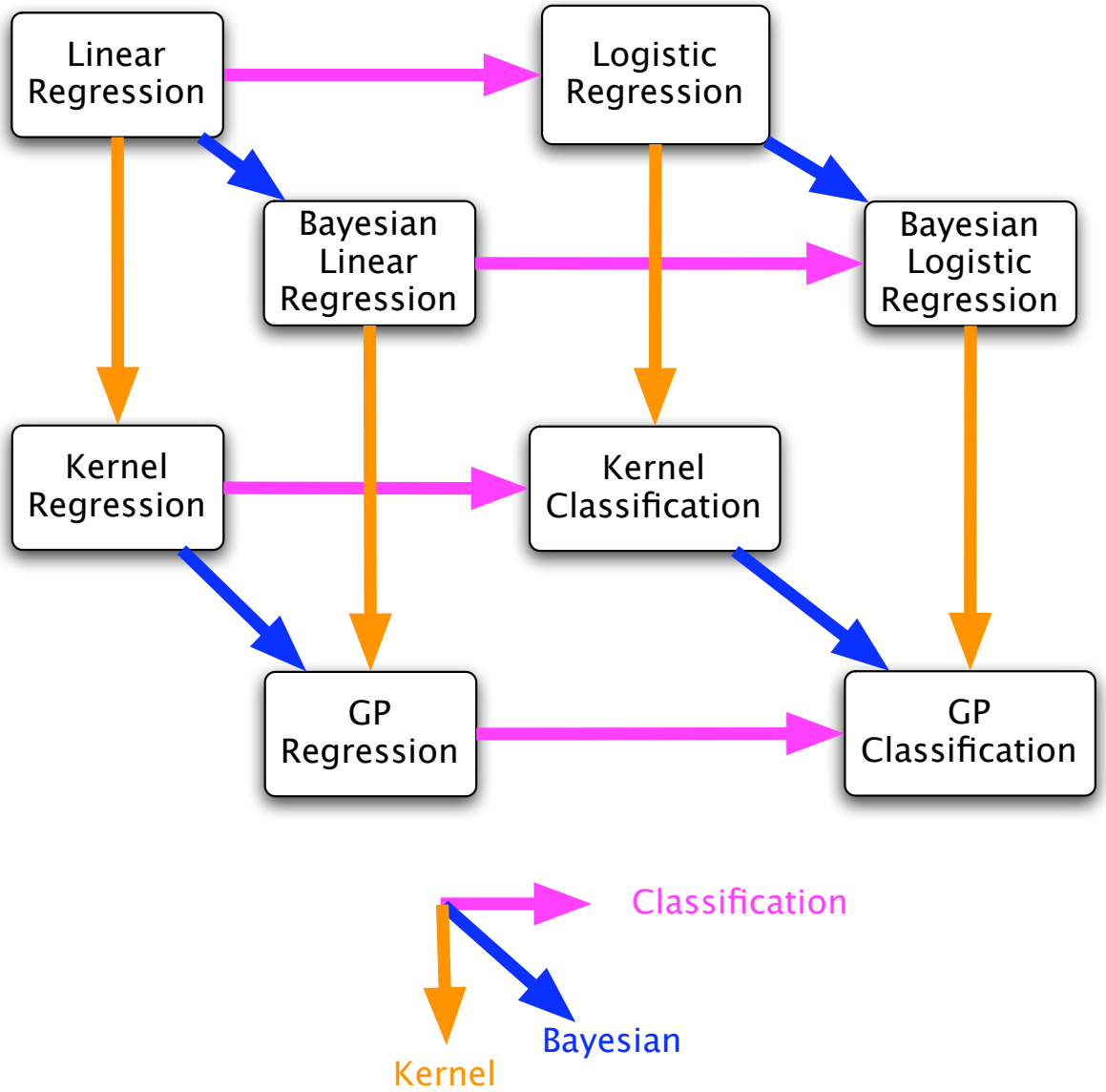
A Gaussian process defines a distribution over functions $p(f)$ which can be used for Bayesian regression:

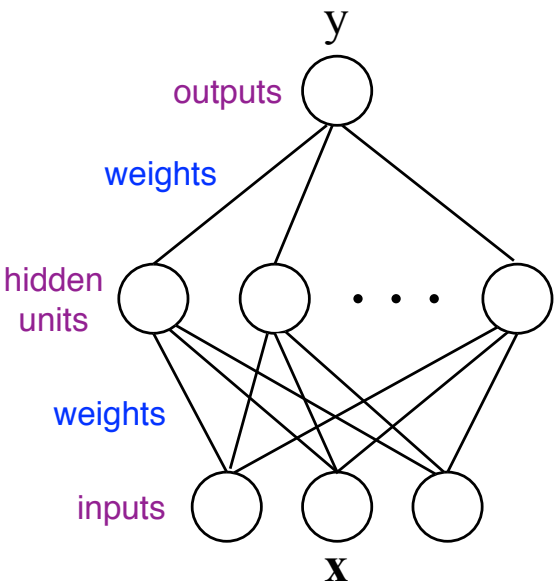
$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

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





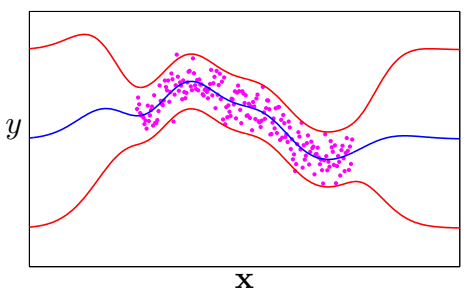
Bayesian neural network

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

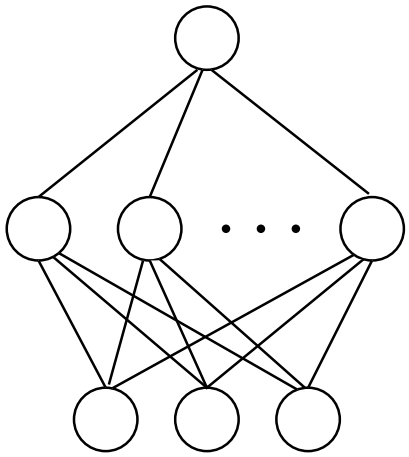
Parameters $\boldsymbol{\theta}$ are weights of neural net

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 Relevance Determination



Bayesian neural network

Data: $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N} = (X, \mathbf{y})$
Parameters (weights): $\boldsymbol{\theta} = \{\{w_{ij}\}, \{v_k\}\}$

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

Automatic Relevance Determination (ARD):

Let the weights from feature x_d have variance α_d^{-1}: $p(w_{dj}|\alpha_d) = \mathcal{N}(0, \alpha_d^{-1})$

Let's think about this:

$\alpha_d \to \infty$	variance $\to 0$	weights $\to 0$	(irrelevant)
$\alpha_d \ll \infty$	finite variance	weight can vary	(relevant)

ARD: Infer relevances $\boldsymbol{\alpha}$ from data. Often we can optimize $\hat{\boldsymbol{\alpha}} = \operatorname{argmax}_{\boldsymbol{\alpha}} p(\mathbf{y}|X, \boldsymbol{\alpha})$.

During optimization some α_d will go to ∞, so the model will discover irrelevant inputs.

Feature and architecture selection, due to MacKay and Neal, now often associated with GPs.

VARIATIONAL LEARNING IN BAYESIAN NEURAL NETWORKS

- Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In COLT, pages 5-13. ACM, 1993. Derives a diagonal Gaussian variational approximation to the Bayesian network weights but couched in a minimum description length information theory language.
- David Barber and Christopher M Bishop. Ensemble learning in Bayesian neural networks. In *Generalization in Neural Networks and Machine Learning* Springer Verlag, 215-238, 1998. Full covariance Gaussian variational approximation to the Bayesian network weights.

VARIATIONAL LEARNING IN BAYESIAN NEURAL NETWORKS

Abstract

Supervised neural networks generalize well if there is much less information in the weights than there is in the output vectors of the training cases. So during learning, it is important to keep the weights simple by penalizing the amount of information they contain. The amount of information in a weight can be controlled by adding Gaussian noise and the noise level can be adapted during learning to optimize the trade-off between the expected squared error of the network and the amount of information in the weights. We describe a method of computing the derivatives of the expected squared error and of the amount of information in the noisy weights in a network that contains a layer of non-linear hidden units. Provided the output units are linear, the exact derivatives can be computed efficiently without time-consuming Monte Carlo simulations. The idea of minimizing the amount of information that is required to communicate the weights of a neural network leads to a number of interesting schemes for encoding the weights.

Target of Bayesian inference:
posterior over weights $p(\theta|\mathcal{D})$.

MCMC:
a chain that samples $\theta_{(t)} \rightarrow \theta_{(t+1)} \rightarrow \theta_{(t+2)} \rightarrow \ldots$ such that the samples converge to the distribution $p(\theta|\mathcal{D})$.

Variational Bayes:
find approximation $q(\theta)$ that is $\arg\min KL(q(\theta)||p(\theta|\mathcal{D}))$.

ASIDE: SIGMOID BELIEF NETWORKS

Connectionist learning of belief networks

Radford M. Neal

Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 1A4

Received January 1991
Revised November 1991

Abstract

Neal, R.M., Connectionist learning of belief networks, Artificial Intelligence 56 (1992) 71–113.

Connectionist learning procedures are presented for "sigmoid" and "noisy-OR" varieties of probabilistic belief networks. These networks have previously been seen primarily as a means of representing knowledge derived from experts. Here it is shown that the "Gibbs sampling" simulation procedure for such networks can support maximum-likelihood learning from empirical data through local gradient ascent. This learning procedure resembles that used for "Boltzmann machines", and like it, allows the use of "hidden" variables to model correlations between visible variables. Due to the directed nature of the connections in a belief network, however, the "negative phase" of Boltzmann machine learning is unnecessary. Experimental results show that, as a result, learning in a sigmoid belief network can be faster than in a Boltzmann machine. These networks have other advantages over Boltzmann machines in pattern classification and decision making applications, are naturally applicable to unsupervised learning problems, and provide a link between work on connectionist learning and work on the representation of expert knowledge.

- Explicit link between feedforward neural networks (aka connectionist networks) and graphical models (aka belief networks).
- Gibbs samples over hidden units.
- A Bayesian nonparametric version of this model which samples over number of hidden units, number of layers, and types of hidden units is given in (Adams, Wallach, and Ghahramani, 2010)

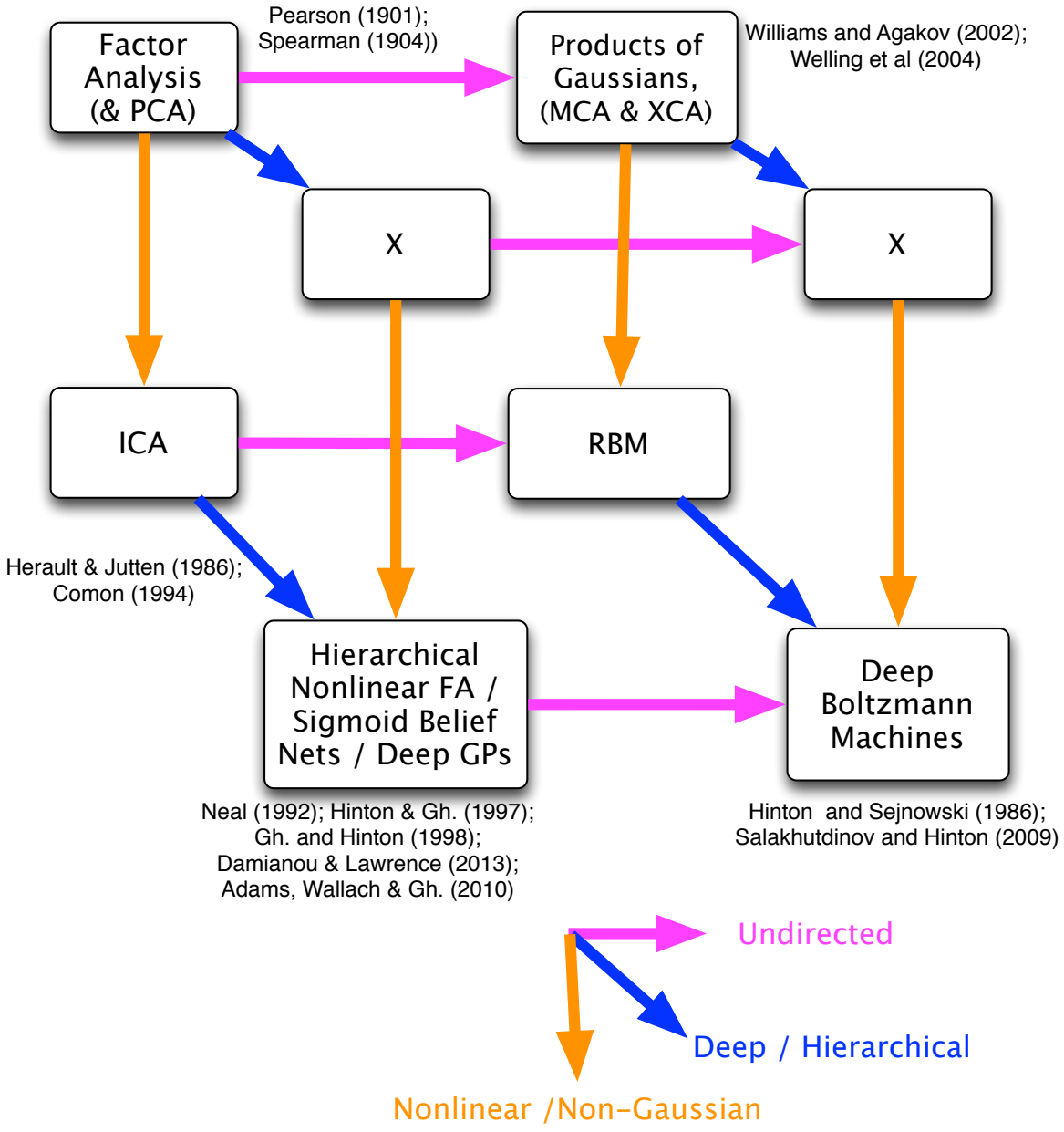
LEARNING THE STRUCTURE OF DEEP SPARSE GRAPHICAL MODELS

By Ryan P. Adams*, Hanna M. Wallach and Zoubin Ghahramani

University of Toronto, University of Massachusetts and University of Cambridge

Deep belief networks are a powerful way to model complex probability distributions. However, learning the structure of a belief network, particularly one with hidden units, is difficult. The Indian buffet process has been used as a nonparametric Bayesian prior on the directed structure of a belief network with a single infinitely wide hidden layer. In this paper, we introduce the cascading Indian buffet process (CIBP), which provides a nonparametric prior on the structure of a layered, directed belief network that is unbounded in both depth and width, yet allows tractable inference. We use the CIBP prior with the nonlinear Gaussian belief network so each unit can additionally vary its behavior between discrete and continuous representations. We provide Markov chain Monte Carlo algorithms for inference in these belief networks and explore the structures learned on several image data sets.

ANOTHER CUBE...



STOCHASTIC GRADIENT LANGEVIN DYNAMICS

- Max Welling and Yee Whye Teh, Bayesian Learning via Stochastic Gradient Langevin Dynamics. ICML 2011.
  Combines SGD with Langevin dynamics (a form of MCMC) to get a highly scalable approximate MCMC algorithm based on minibatch SGD.

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{ti}|\theta_t)\right) + \eta_t$$

$$\eta_t \sim N(0, \epsilon_t) \tag{4}$$

Doing Bayesian inference can be as simple as running noisy SGD

Bayesian Neural Network Revival (some recent papers)

- A. Honkela and H. Valpola. Variational learning and bits-back coding: An information-theoretic view to Bayesian learning. IEEE Transactions on Neural Networks, 15:800-810, 2004.
- Alex Graves. Practical variational inference for neural networks. In NIPS 2011.
- Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In ICML, 2015.
- José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In ICML, 2015.
- José Miguel Hernández-Lobato, Yingzhen Li, Daniel Hernández-Lobato, Thang Bui, and Richard E Turner. Black-box alpha divergence minimization. In Proceedings of The 33rd International Conference on Machine Learning, pages 1511-1520, 2016.
- 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.

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

CONCLUSIONS

Probabilistic modelling offers a general framework for building systems that learn from data

Advantages include better estimates of uncertainty, automatic ways of learning structure and avoiding overfitting, and a principled foundation.

Disadvantages include higher computational cost, depending on the approximate inference algorithm

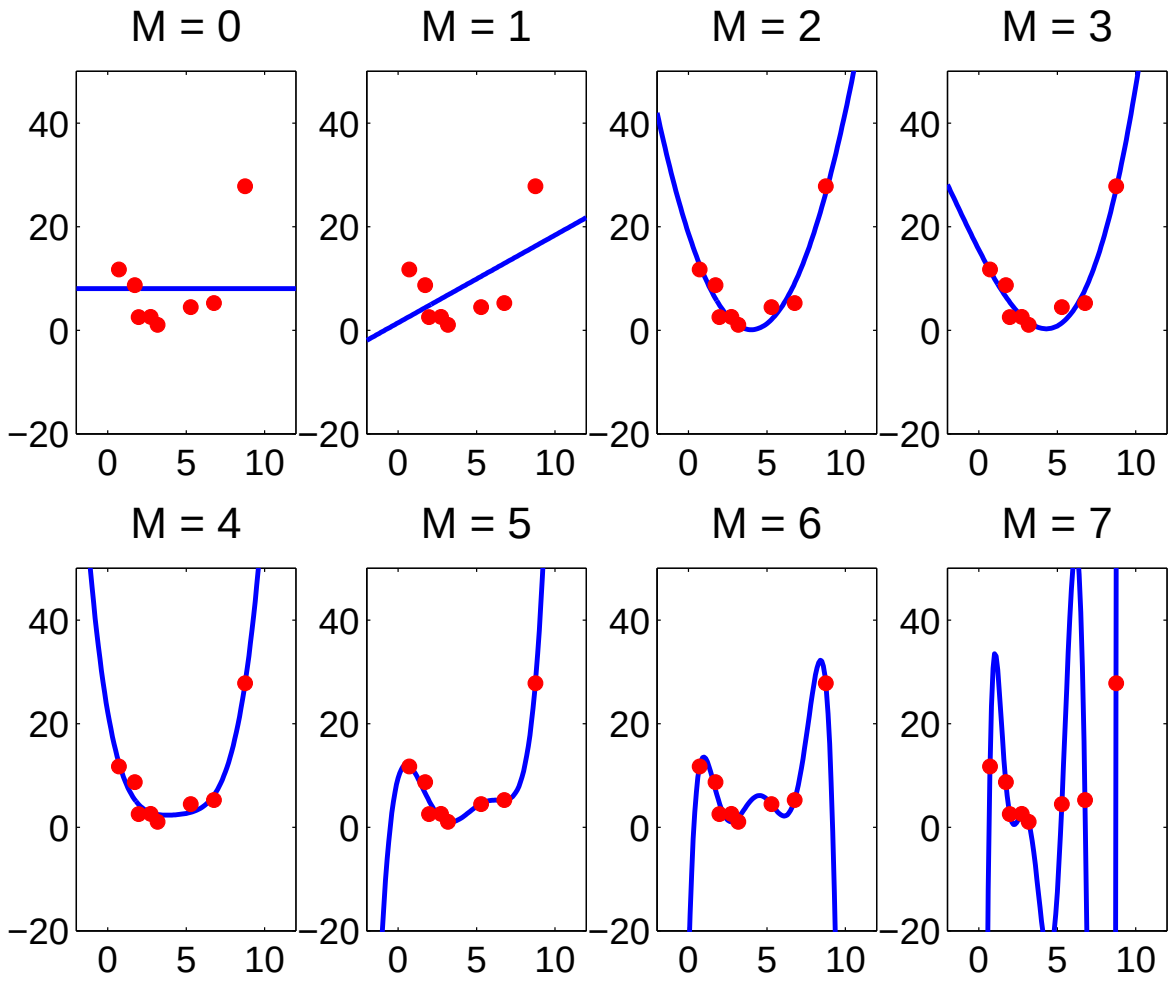
Bayesian neural networks have a long history and are undergoing a tremendous wave of revival.

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

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

APPENDIX

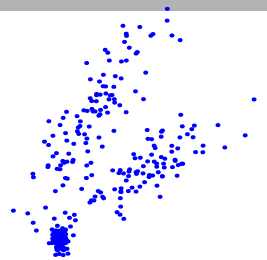
Model Comparison



LEARNING MODEL STRUCTURE

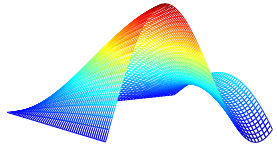
How many clusters in the data?

k-means, mixture models



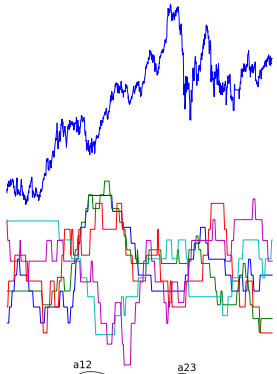
What is the intrinsic data dimensionality ?

PCA, LLE, Isomap, GPLVM



Is this input relevant to predicting that output?

feature / variable selection

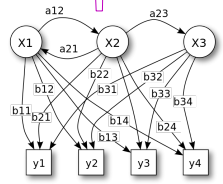


What is the order of a dynamical system?

state-space models, ARMA, GARCH

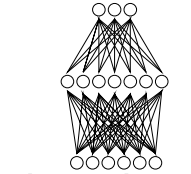
How many states in a hidden Markov model?

HMM

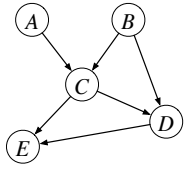


How many units or layers in a neural net?

neural networks, RNNs, ICA



How to learn the structure of a graphical model?



BAYESIAN OCCAM'S RAZOR

Compare model classes, e.g. m and m', using posterior prob. given $\mathcal{D}$:

$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)\, p(m)}{p(\mathcal{D})}, \qquad p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)\; p(\boldsymbol{\theta}|m)\; d\boldsymbol{\theta}$$

BAYESIAN OCCAM'S RAZOR

Compare model classes, e.g. m and m', using posterior prob. given $\mathcal{D}$:

$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)\, p(m)}{p(\mathcal{D})}, \qquad p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)\; p(\boldsymbol{\theta}|m)\; d\boldsymbol{\theta}$$

Interpretations of the Marginal Likelihood ("model evidence"):

- Probability of the data under the model, *averaging* over all possible parameter values.
- The probability that *randomly selected* parameters from the prior would generate $\mathcal{D}$.
- $\log_2 \left(\frac{1}{p(\mathcal{D}|m)} \right)$ is the number of *bits of surprise* at observing data $\mathcal{D}$ under model m.

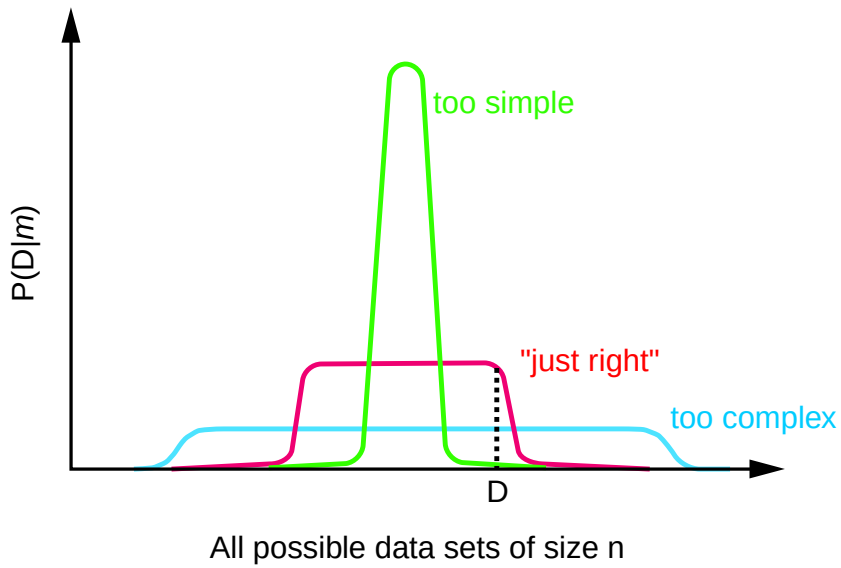
BAYESIAN OCCAM'S RAZOR

Compare model classes, e.g. m and m', using posterior prob. given $\mathcal{D}$:

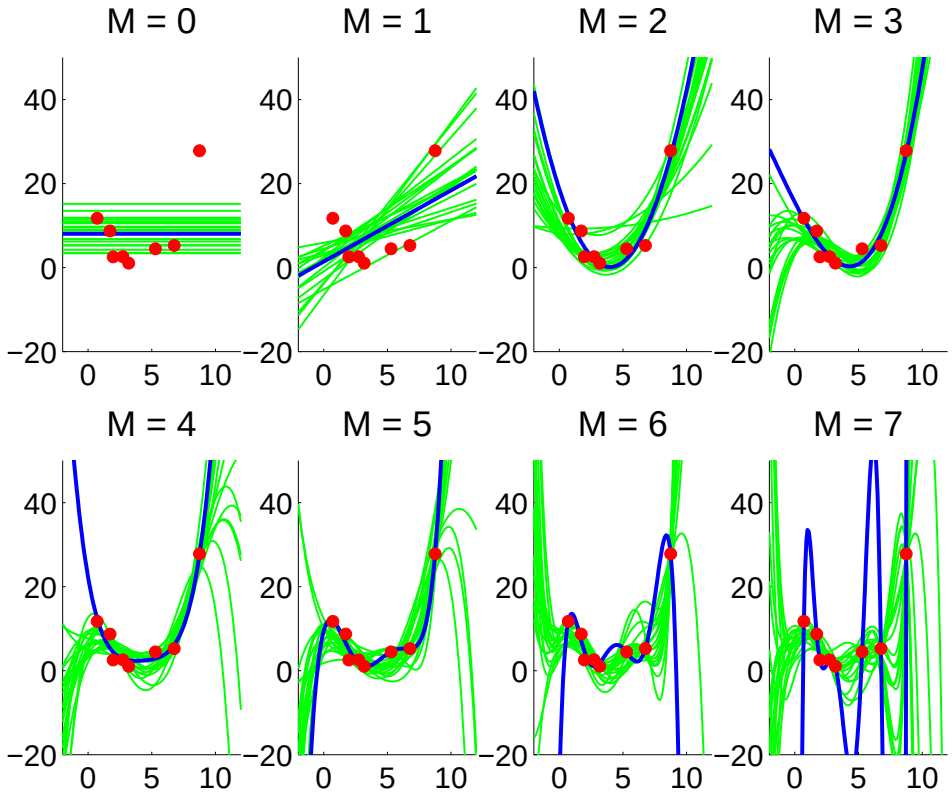
$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)\,p(m)}{p(\mathcal{D})}, \qquad p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)\; p(\boldsymbol{\theta}|m)\; d\boldsymbol{\theta}$$

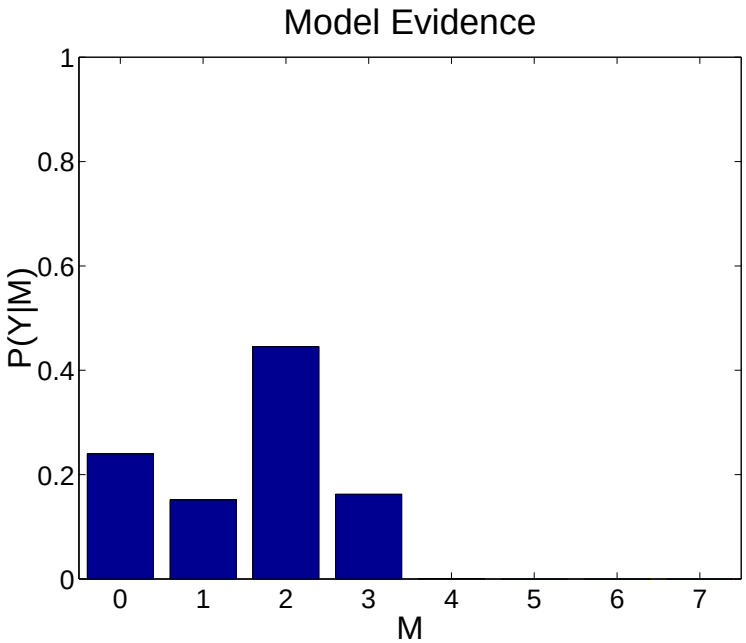
Model classes that are too simple are unlikely to generate the data set.

Model classes that are too complex can generate many possible data sets, so again, they are unlikely to generate that particular data set at random.



MODEL COMPARISON & OCCAM'S RAZOR





For example, for quadratic polynomials ($m = 2$):
$y = a_0 + a_1 x + a_2 x^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and parameters
$\boldsymbol{\theta} = (a_0\ a_1\ a_2\ \sigma)$

`demo: polybayes`

Approximation Methods for Posteriors and Marginal Likelihoods

Observed data $\mathcal{D}$, parameters $\boldsymbol{\theta}$, model class m:

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

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)\; p(\boldsymbol{\theta}|m)\; d\boldsymbol{\theta}$$

Approximation Methods for Posteriors and Marginal Likelihoods

Observed data $\mathcal{D}$, parameters $\boldsymbol{\theta}$, model class m:

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

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)\; p(\boldsymbol{\theta}|m)\; d\boldsymbol{\theta}$$

- Laplace approximation
- Bayesian Information Criterion (BIC)
- Variational approximations
- Expectation Propagation (EP)
- Markov chain Monte Carlo methods (MCMC)
- Sequential Monte Carlo (SMC)
- Exact Sampling
- ...