Nonparametric Bayesian Modelling

Zoubin Ghahramani

Department of Engineering
University of Cambridge, UK

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

ICMLA, Washington 2010

An Information Revolution?

- We are in an era of abundant data:
  - **Society:** the web, social networks, mobile networks, government, digital archives
  - **Science:** large-scale scientific experiments, biomedical data, climate data, scientific literature
  - **Business:** e-commerce, electronic trading, advertising, personalisation

- We need tools for modelling, searching, visualising, and understanding large data sets.

Modelling Tools

Our modelling tools should:

- Faithfully represent **uncertainty** in our model structure and parameters and **noise** in our data
- Be automated and **adaptive**
- Exhibit **robustness**
- **Scale well** to large data sets

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

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

Rev'd Thomas Bayes (1702–1761)

- Bayes rule tells us how to do inference about hypotheses from data.
- Learning and prediction can be seen as forms of inference.

Machine Learning

- *Machine learning is an interdisciplinary field studying both the mathematical foundations and practical applications of systems that learn, reason and act.*

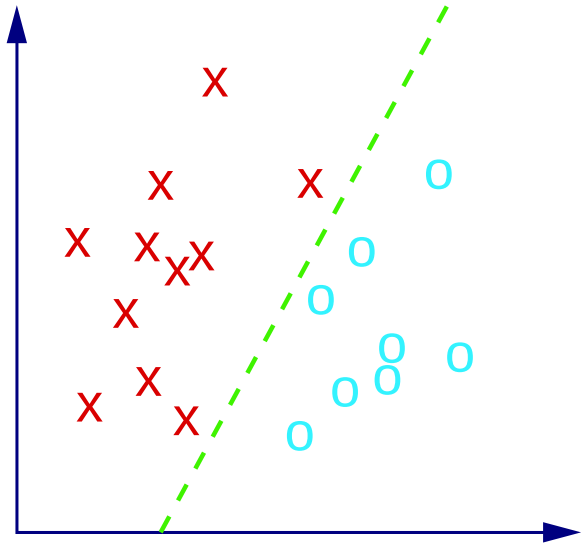
- Other related terms: Pattern Recognition, Neural Networks, Data Mining, Statistical Modelling ...

- Using ideas from: Statistics, Computer Science, Engineering, Applied Mathematics, Cognitive Science, Psychology, Computational Neuroscience, Economics

Linear Classification

Data: $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}$ for $n = 1, \ldots, N$ data points

$$\begin{aligned} \mathbf{x}^{(n)} &\in \Re^D \\ y^{(n)} &\in \{+1, -1\} \end{aligned}$$



Model:

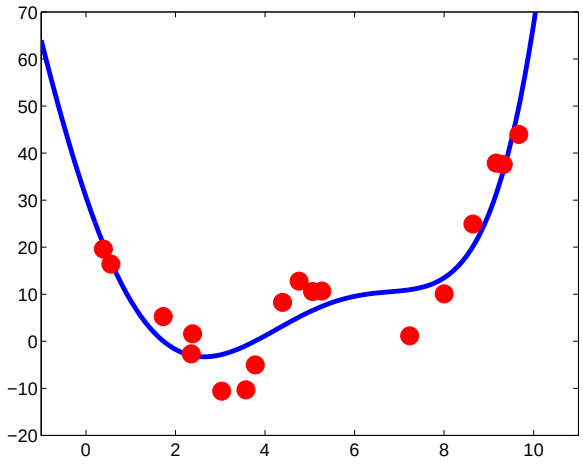
$$P(y^{(n)} = +1 | \boldsymbol{\theta}, \mathbf{x}^{(n)}) = \begin{cases} 1 & \text{if } \sum_{d=1}^{D} \theta_d \, x_d^{(n)} + \theta_0 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Parameters: $\boldsymbol{\theta} \in \Re^{D+1}$

Goal: To infer $\boldsymbol{\theta}$ from the data and to predict future labels $P(y|\mathcal{D}, \mathbf{x})$

Polynomial Regression

Data: $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}$ for $n = 1, \ldots, N$



$$x^{(n)} \in \Re$$
$$y^{(n)} \in \Re$$

Model:

$$y^{(n)} = a_0 + a_1 x^{(n)} + a_2 {x^{(n)}}^2 \ldots + a_m {x^{(n)}}^m + \epsilon$$

where

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

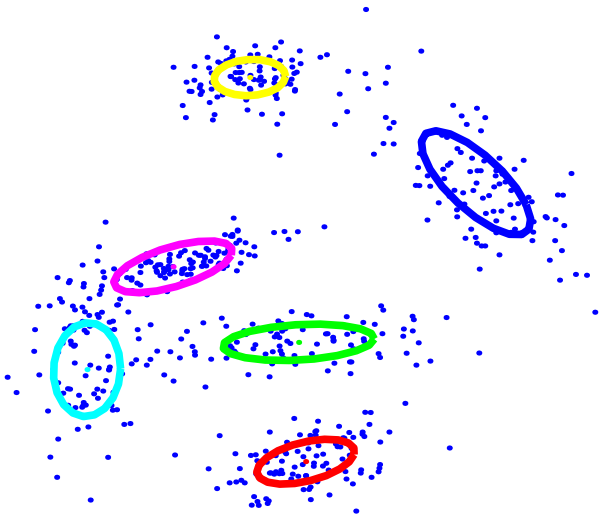
Parameters: $\boldsymbol{\theta} = (a_0, \ldots, a_m, \sigma)$

Goal: To infer $\boldsymbol{\theta}$ from the data and to predict future outputs $P(y|\mathcal{D}, x, m)$

Clustering with Gaussian Mixtures (Density Estimation)

Data: $\mathcal{D} = \{\mathbf{x}^{(n)}\}$ for $n = 1, \ldots, N$

$$\mathbf{x}^{(n)} \in \Re^D$$



Model:

$$\mathbf{x}^{(n)} \sim \sum_{i=1}^{m} \pi_i \, p_i(\mathbf{x}^{(n)})$$

where

$$p_i(\mathbf{x}^{(n)}) = \mathcal{N}(\boldsymbol{\mu}^{(i)}, \Sigma^{(i)})$$

Parameters: $\boldsymbol{\theta} = \left((\boldsymbol{\mu}^{(1)}, \Sigma^{(1)}) \ldots, (\boldsymbol{\mu}^{(m)}, \Sigma^{(m)}), \boldsymbol{\pi}\right)$

Goal: To infer $\boldsymbol{\theta}$ from the data and predict the density $p(\mathbf{x}|\mathcal{D}, m)$ or the probability that two points belong to the same cluster.

Bayesian Machine Learning

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

$P(\mathcal{D}\|\theta)$	likelihood of θ
$P(\theta)$	prior probability of θ
$P(\theta\|\mathcal{D})$	posterior of θ given $\mathcal{D}$

Prediction:

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

Model Comparison:

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

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

Bayesian Occam's Razor and Model Comparison

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

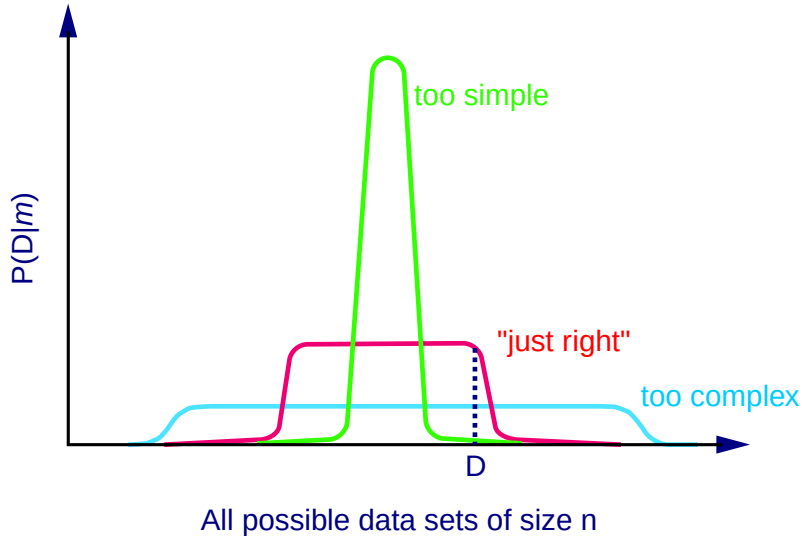
$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)\,p(m)}{p(\mathcal{D})}, \qquad p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta}, m)\; p(\boldsymbol{\theta}|m)\; d\boldsymbol{\theta}$$

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

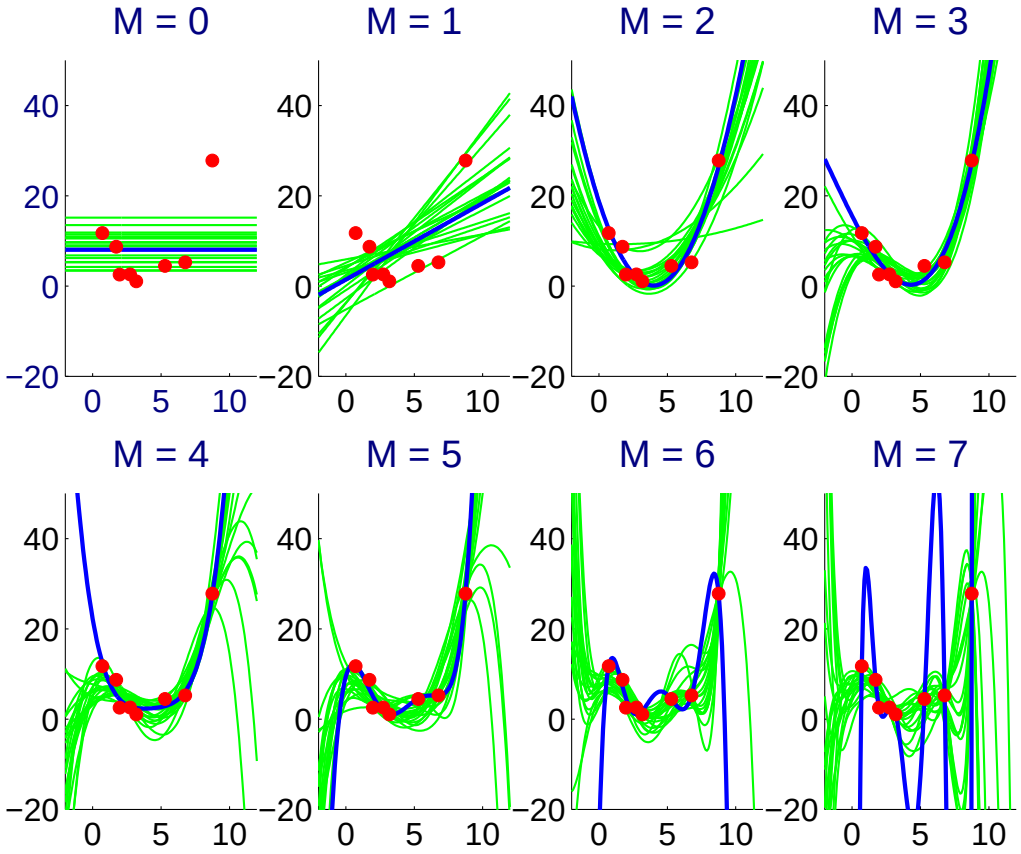
- The probability that *randomly selected* parameters from the prior would generate $\mathcal{D}$.
- Probability of the data under the model, *averaging* over all possible parameter values.
- $\log_2\left(\frac{1}{p(\mathcal{D}|m)}\right)$ is the number of *bits of surprise* at observing data $\mathcal{D}$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

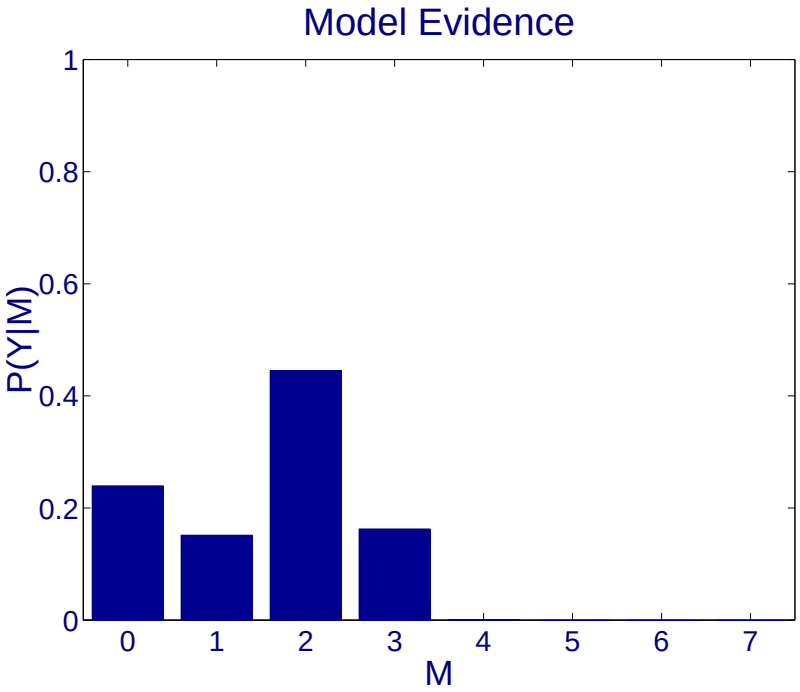
Model classes that are too simple are unlikely to generate the data set.

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.



Bayesian Model Comparison: Occam's Razor at Work





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`

Myths and misconceptions about Bayesian methods

- **Bayesian methods make assumptions where other methods don't**
  All methods make assumptions! Otherwise it's impossible to predict. Bayesian methods are transparent in their assumptions whereas other methods are often opaque.

- **If you don't have the right prior you won't do well**
  Certainly a poor model will predict poorly but there is no such thing as the right prior! Your model (both prior and likelihood) should capture a reasonable range of possibilities. When in doubt you can choose vague priors (cf nonparametrics).

- **Maximum A Posteriori (MAP) is a Bayesian method**
  MAP is similar to regularization and offers no particular Bayesian advantages. The key ingredient in Bayesian methods is to average over your uncertain variables and parameters, rather than to optimize.

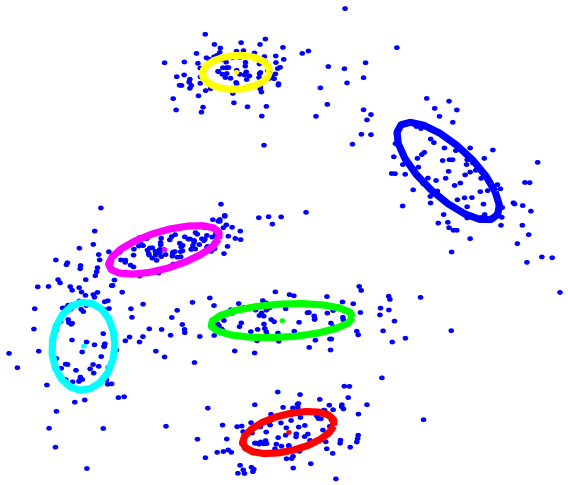
Myths and misconceptions about Bayesian methods

- **Bayesian methods don't have theoretical guarantees**
  One can often apply frequentist style generalization error bounds to Bayesian methods (e.g. PAC-Bayes). Moreover, it is often possible to prove convergence, consistency and rates for Bayesian methods.

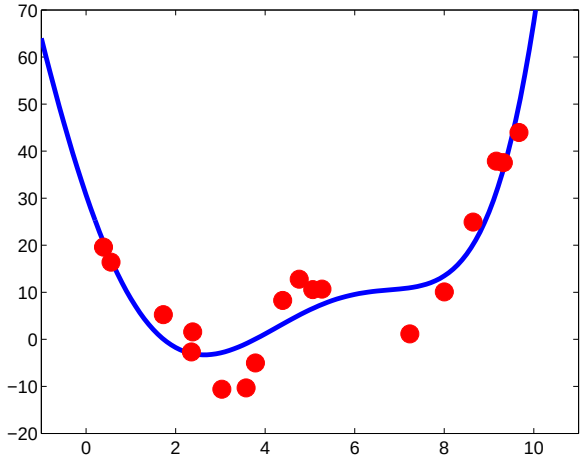
- **Bayesian methods are generative**
  You can use Bayesian approaches for both generative and discriminative learning (e.g. Gaussian process classification).

- **Bayesian methods don't scale well**
  With the right inference methods (variational, MCMC) it is possible to scale to very large datasets (e.g. excellent results for Bayesian Probabilistic Matrix Factorization on the Netflix dataset using MCMC).

Model Comparison: two examples



e.g. selecting m, the number of Gaussians in a mixture model



e.g. selecting m the order of a polynomial in a nonlinear regression model

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

A possible procedure:

1. place a prior on m, $P(m)$
2. given data, use Bayes rule to infer $P(m|\mathcal{D})$ and to make predictions $P(y|\mathcal{D})$

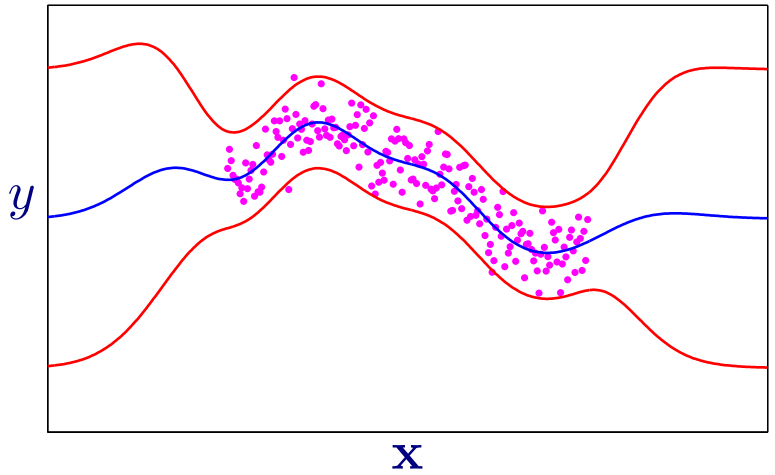
What is the problem with this procedure?

Non-parametric Bayesian Models

- Real data are complicated and we don't really believe a low-order polynomial or mixture of Gaussians can adequately model it.
- Bayesian methods are most powerful when your model adequately captures relevant aspects of your data.
- Inflexible models (e.g. mixture of 5 Gaussians, 4th order polynomial) yield unreasonable inferences.
- Non-parametric models are a way of getting very flexible models.
- Many can be derived by starting with a finite parametric model and taking the limit as number of parameters $\rightarrow \infty$
- The effective complexity of the model grows with more data.
- Nonparametric methods are often faster and conceptually easier to implement since one doesn't have to compare multiple nested models.

Nonlinear regression and 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:

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

Let $\mathbf{f} = (f(x_1), f(x_2), \ldots, f(x_n))$ be an n-dimensional vector of function values evaluated at n points $x_i \in \mathcal{X}$. Note, $\mathbf{f}$ is a random variable.

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.

Gaussian process covariance functions (kernels)

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

Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, or kernel, $K(x, x')$.

$$P(f(x), f(x')) = \mathsf{N}(\boldsymbol{\mu}, \Sigma)$$

where

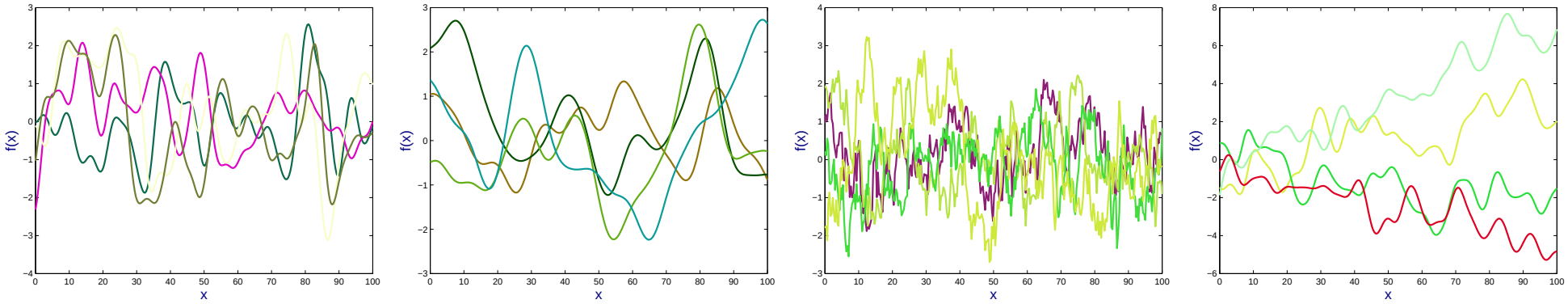
$$\boldsymbol{\mu} = \begin{bmatrix} \mu(x) \\ \mu(x') \end{bmatrix} \quad \Sigma = \begin{bmatrix} K(x,x) & K(x,x') \\ K(x',x) & K(x',x') \end{bmatrix}$$

and similarly for $P(f(x_1), \ldots, f(x_n))$ where now $\boldsymbol{\mu}$ is an $n \times 1$ vector and Σ is an $n \times n$ matrix.

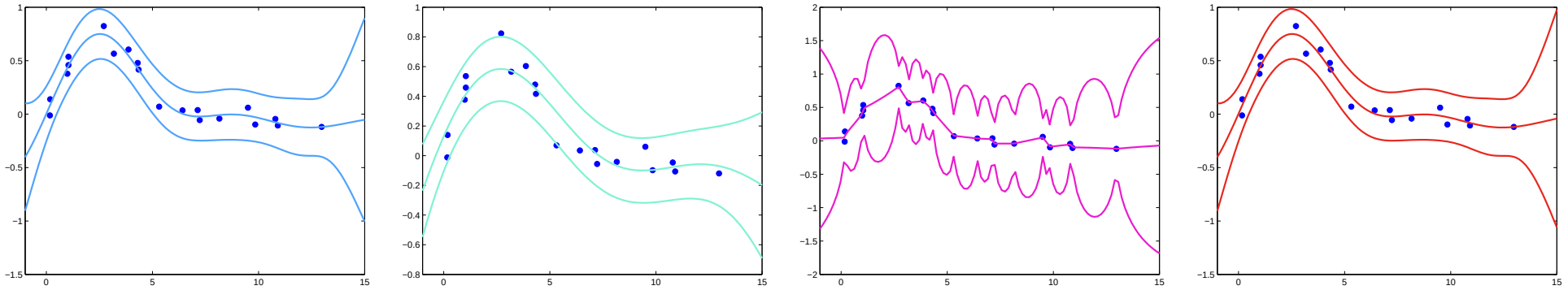
Once the mean and kernel are defined, everything else about GPs follows from the basic rules of probability applied to mutivariate Gaussians.

Prediction using GPs with different $K(x, x')$

Samples from the prior for different covariance functions:



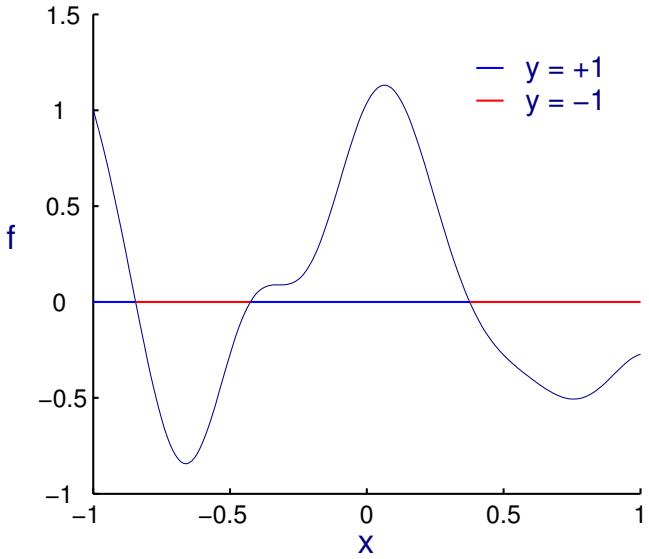
Corresponding predictions, mean with two standard deviations:

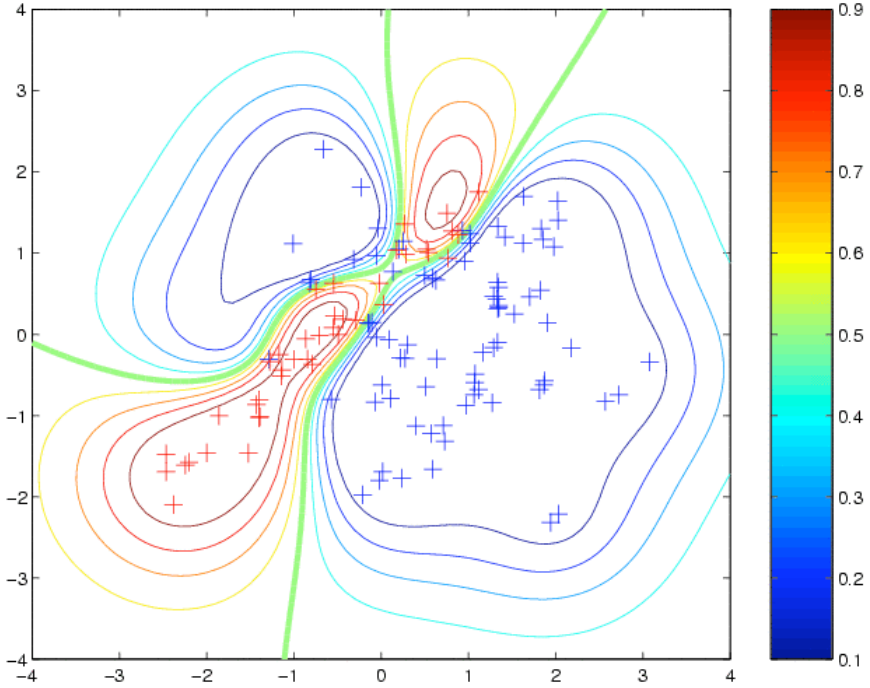


E.g.: $K(x_i, x_j) = v_0 \exp\left\{-\left(\frac{|x_i - x_j|}{\lambda}\right)^{\alpha}\right\} + v_1 + v_2\,\delta_{ij}$ with params $(v_0, v_1, v_2, \lambda, \alpha)$

Using Gaussian Processes for Classification

Binary classification problem: Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with binary class labels $y_i \in \{-1, +1\}$, infer class label probabilities at new points.





There are many ways to relate function values $f_i = f(\mathbf{x}_i)$ to class probabilities:

$$p(y_i|f_i) = \begin{cases} \frac{1}{1+\exp(-y_i f_i)} & \text{sigmoid (logistic)} \\ \Phi(y_i f_i) & \text{cumulative normal (probit)} \\ \boldsymbol{H}(y_i f_i) & \text{threshold} \\ \epsilon + (1-2\epsilon)\boldsymbol{H}(y_i f_i) & \text{robust threshold} \end{cases}$$

Non-Gaussian likelihood, so we need to use approximate inference methods (Laplace, EP, MCMC).

Support Vector Machines and Gaussian Processes Classification

We can write the SVM loss as: $\min_{\mathbf{f}} \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_i (1 - y_i f_i)_+$

We can write the negative log of a GP likelihood as: $\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} - \sum_i \ln p(y_i|f_i) + c$

Equivalent? No.

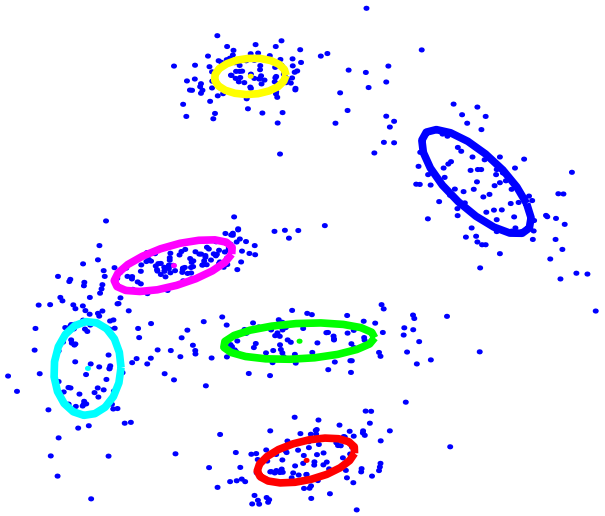
With Gaussian processes we:

- Handle **uncertainty** in unknown function $\mathbf{f}$ by averaging, not minimization.
- Compute $p(y = +1|\mathbf{x}) \neq p(y = +1|\hat{\mathbf{f}}, \mathbf{x})$.
- Can **learn the kernel parameters** automatically from data, no matter how flexible we wish to make the kernel.
- Can **learn the regularization parameter** C without cross-validation.
- Can incorporate **interpretable** noise models and priors over functions, and can sample from prior to get intuitions about the model assumptions.
- We can combine **automatic feature selection** with learning using ARD.

Clustering

Clustering

Basic idea: each data point belongs to a cluster

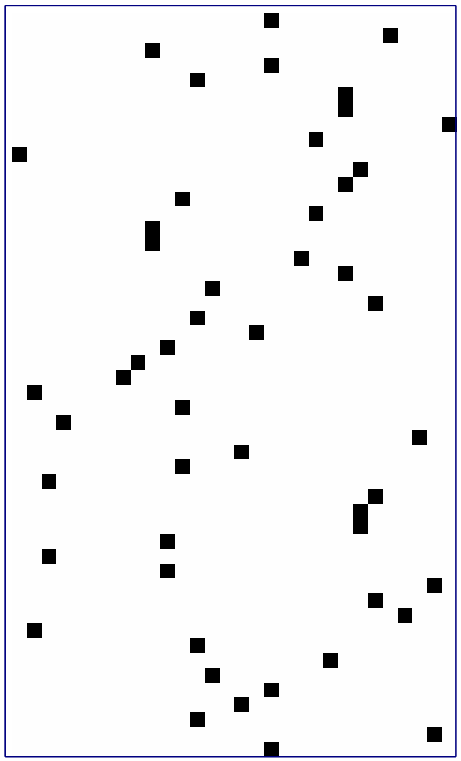


Many clustering methods exist:

- mixture models
- hierarchical clustering
- spectral clustering

Goal: to partition data into groups in an unsupervised manner

A binary matrix representation for clustering

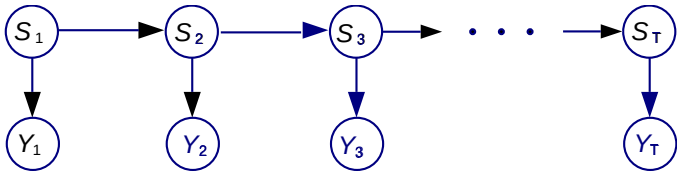


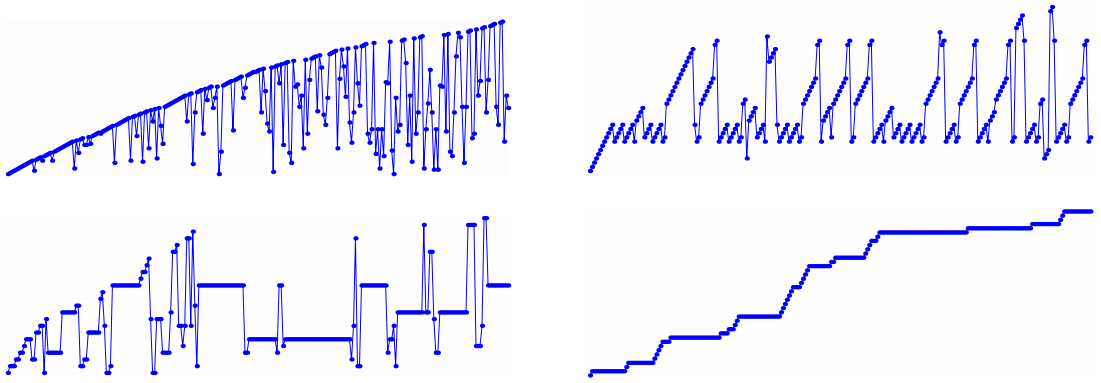
- Rows are data points
- Columns are clusters
- Since each data point is assigned to one and only one cluster, the rows sum to one.
- Finite mixture models: number of columns is finite
- Infinite mixture models (DPMs): number of columns is countably infinite

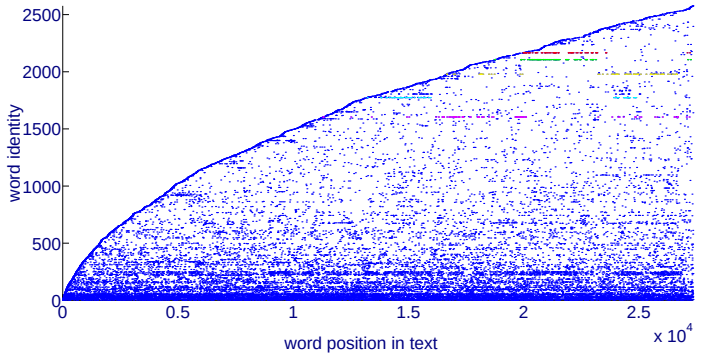
Infinite hidden Markov models (iHMMs)

Hidden Markov models (HMMs) are widely used sequence models for speech recognition, bioinformatics, text modelling, video monitoring, etc. HMMs can be thought of as *time-dependent mixture models*.

In an HMM with K states, the transition matrix has $K \times K$ elements. Let $K \to \infty$.

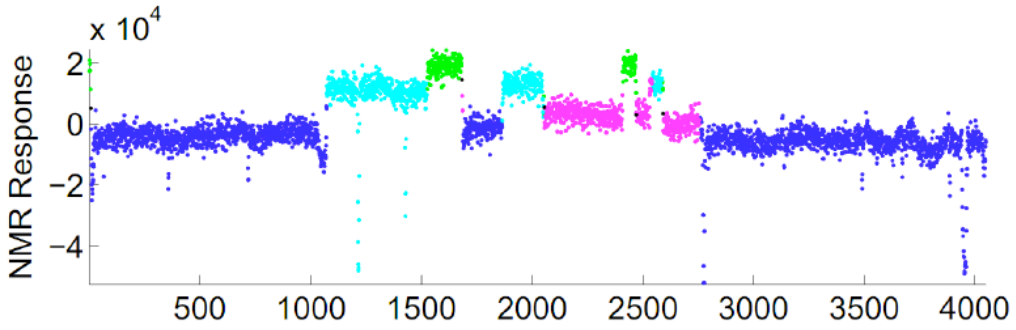


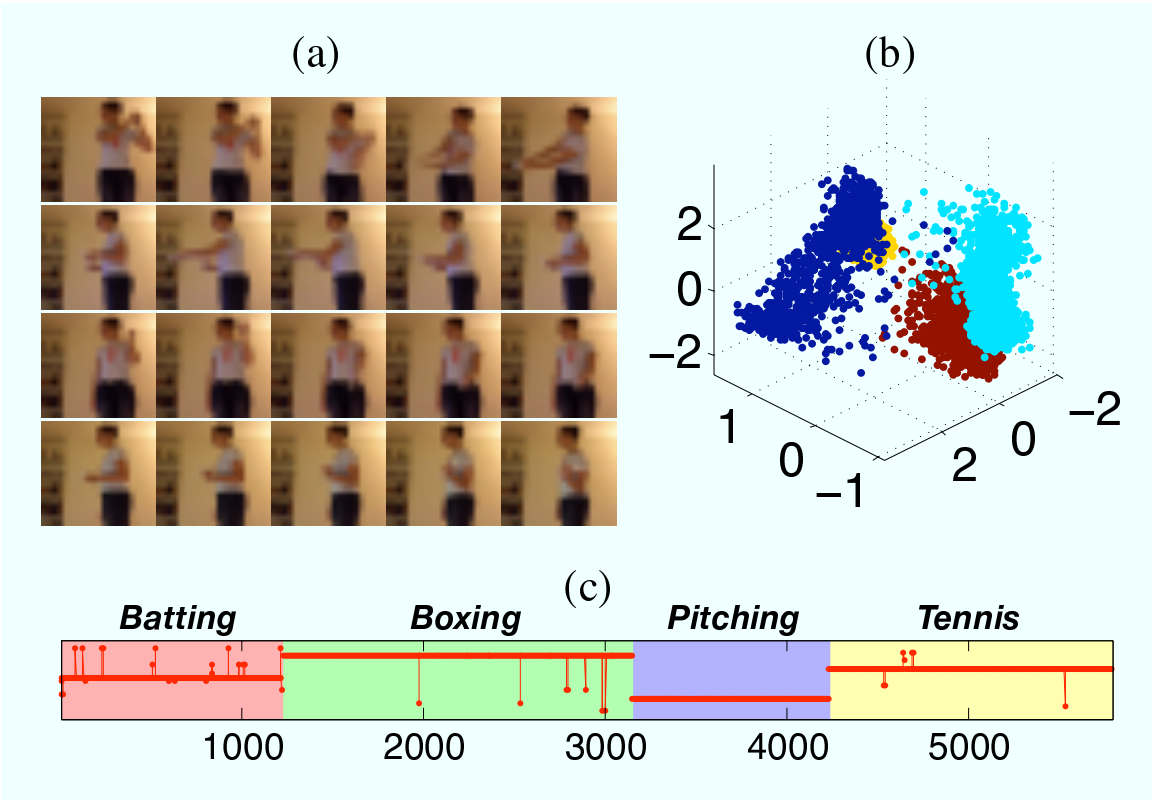




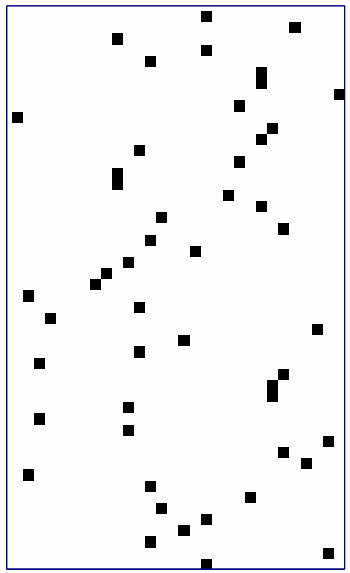
- Introduced in (Beal, Ghahramani and Rasmussen, 2002).
- Teh, Jordan, Beal and Blei (2005) showed that iHMMs can be derived from hierarchical Dirichlet processes, and provided a more efficient Gibbs sampler.
- We have recently derived a much more efficient sampler based on Dynamic Programming (Van Gael, Saatci, Teh, and Ghahramani, 2008).

Infinite HMM: Changepoint detection and video segmentation



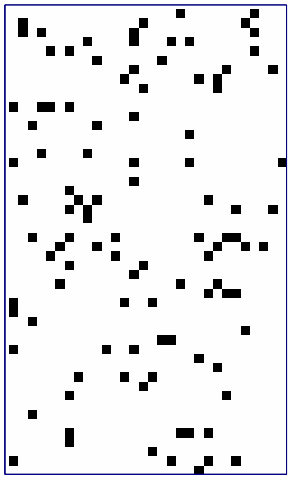


A binary matrix representation for clustering



- Rows are data points
- Columns are clusters
- Since each data point is assigned to one and only one cluster...
- ...the rows sum to one.

More general priors on binary matrices



- Rows are data points
- Columns are latent features
- We can think of **infinite** binary matrices...
  ...where each data point can now have *multiple* features, so...
  ...the rows can sum to more than one.

Another way of thinking about this:

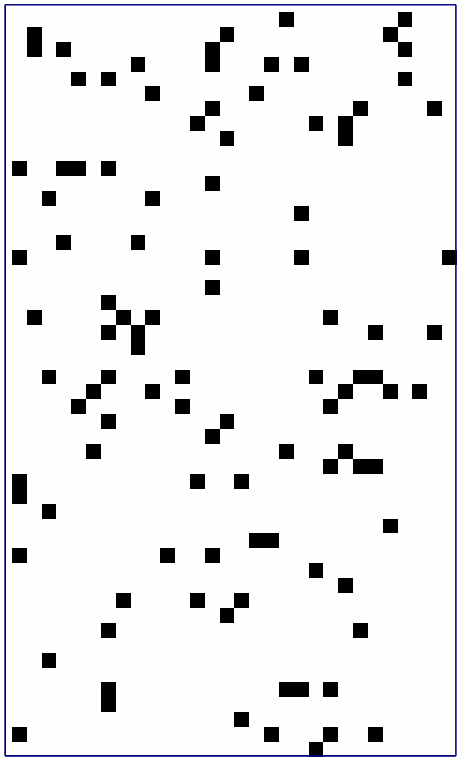
- there are multiple overlapping clusters
- each data point can belong to several clusters simultaneously.

(Griffiths and Ghahramani, 2005)

Why?

- Many statistical models can be thought of as modelling data in terms of hidden or latent variables.
- Clustering algorithms (e.g. using mixture models) represent data in terms of which cluster each data point belongs to.
- But clustering models are restrictive, they do not have distributed representations.
- Consider modelling people's movie preferences (the "Netflix" problem). A movie might be described using features such as "is science fiction", "has Charlton Heston", "was made in the US", "was made in 1970s", "has apes in it"... these features may be unobserved (latent).
- The number of potential latent features for describing a movie (or person, news story, image, gene, speech waveform, etc) is unlimited.

From finite to infinite binary matrices



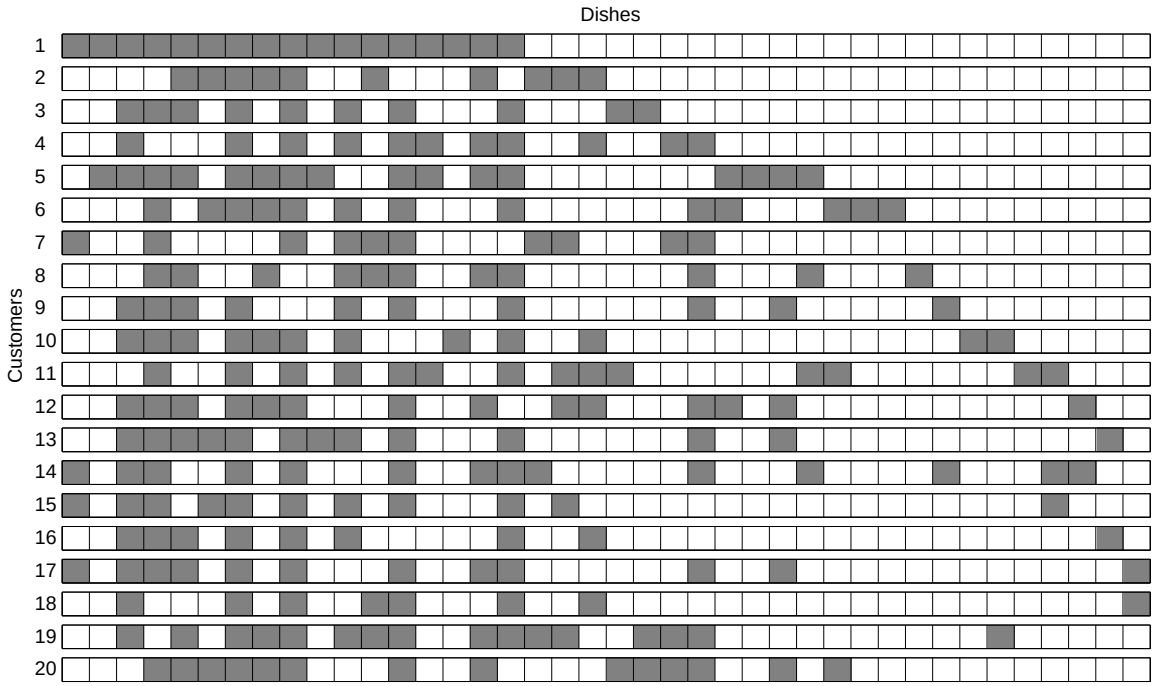
$z_{nk} = 1$ means object n has feature k:

$$z_{nk} \sim \text{Bernoulli}(\theta_k)$$

$$\theta_k \sim \text{Beta}(\alpha/K, 1)$$

- Note that $P(z_{nk} = 1|\alpha) = E(\theta_k) = \frac{\alpha/K}{\alpha/K+1}$, so as K grows larger the matrix gets sparser.
- So if $\mathbf{Z}$ is $N \times K$, the expected number of nonzero entries is $N\alpha/(1 + \alpha/K) < N\alpha$.
- Even in the $K \to \infty$ limit, the matrix is expected to have a finite number of non-zero entries.

Indian buffet process



"Many Indian restaurants in London offer lunchtime buffets with an apparently infinite number of dishes"

- First customer starts at the left of the buffet, and takes a serving from each dish, stopping after a Poisson(α) number of dishes as her plate becomes overburdened.
- The nth customer moves along the buffet, sampling dishes in proportion to their popularity, serving himself with probability m_k/n, and trying a Poisson(α/n) number of new dishes.
- The customer-dish matrix is our feature matrix, $\mathbf{Z}$.

Modelling Data with Indian Buffet Processes

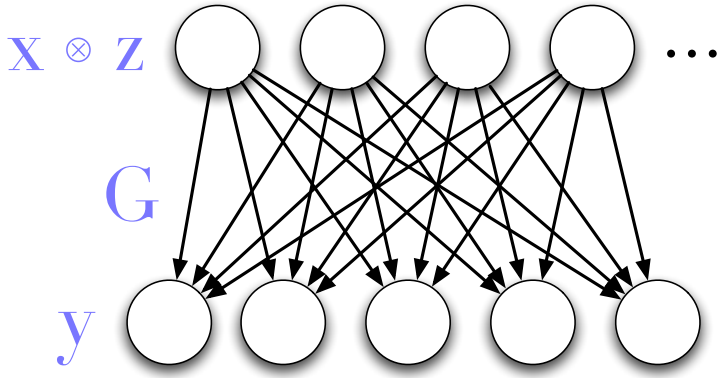
Latent variable model: let $\mathbf{X}$ be the $N \times D$ matrix of observed data, and $\mathbf{Z}$ be the $N \times K$ matrix of binary latent features

$$P(\mathbf{X}, \mathbf{Z}|\alpha) = P(\mathbf{X}|\mathbf{Z})P(\mathbf{Z}|\alpha)$$

By combining the IBP with different likelihood functions we can get different kinds of models:

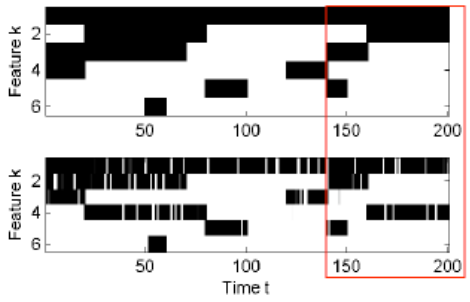
- Models for graph structures (w/ Wood, Griffiths, 2006; w/ Adams and Wallach, 2010)
- Models for protein complexes (w/ Chu, Wild, 2006)
- Models for choice behaviour (Görür & Rasmussen, 2006)
- Models for users in collaborative filtering (w/ Meeds, Roweis, Neal, 2006)
- Sparse latent trait, pPCA and ICA models (w/ Knowles, 2007)
- Models for overlapping clusters (w/ Heller, 2007)

Nonparametric Sparse Latent Factor Models and Infinite Independent Components Analysis

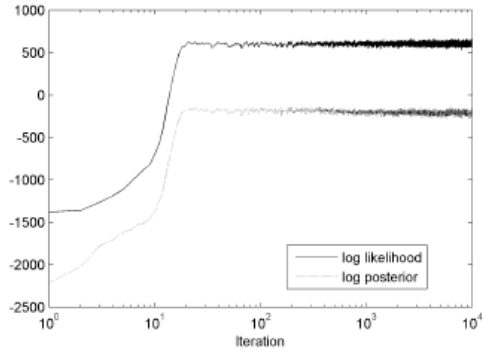


Model: $\mathbf{Y} = \mathbf{G}(\mathbf{Z} \otimes \mathbf{X}) + \mathbf{E}$

where $\mathbf{Y}$ is the data matrix, $\mathbf{G}$ is the mixing matrix $\mathbf{Z} \sim \mathrm{IBP}(\alpha, \beta)$ is a mask matrix, $\mathbf{X}$ is heavy tailed sources and $\mathbf{E}$ is Gaussian noise.



(a) *Top:* True $\mathbf{Z}$. *Bottom:* Inferred $\mathbf{Z}$. Red box denotes test data.



(b) Plot of the log likelihood and posterior for the duration of the iICA_2 run.

Fig. 1. True and inferred $\mathbf{Z}$ and algorithm convergence.

(w/ David Knowles, 2007)

Binary Matrix Factorization

genes $\times$ patients

users $\times$ movies

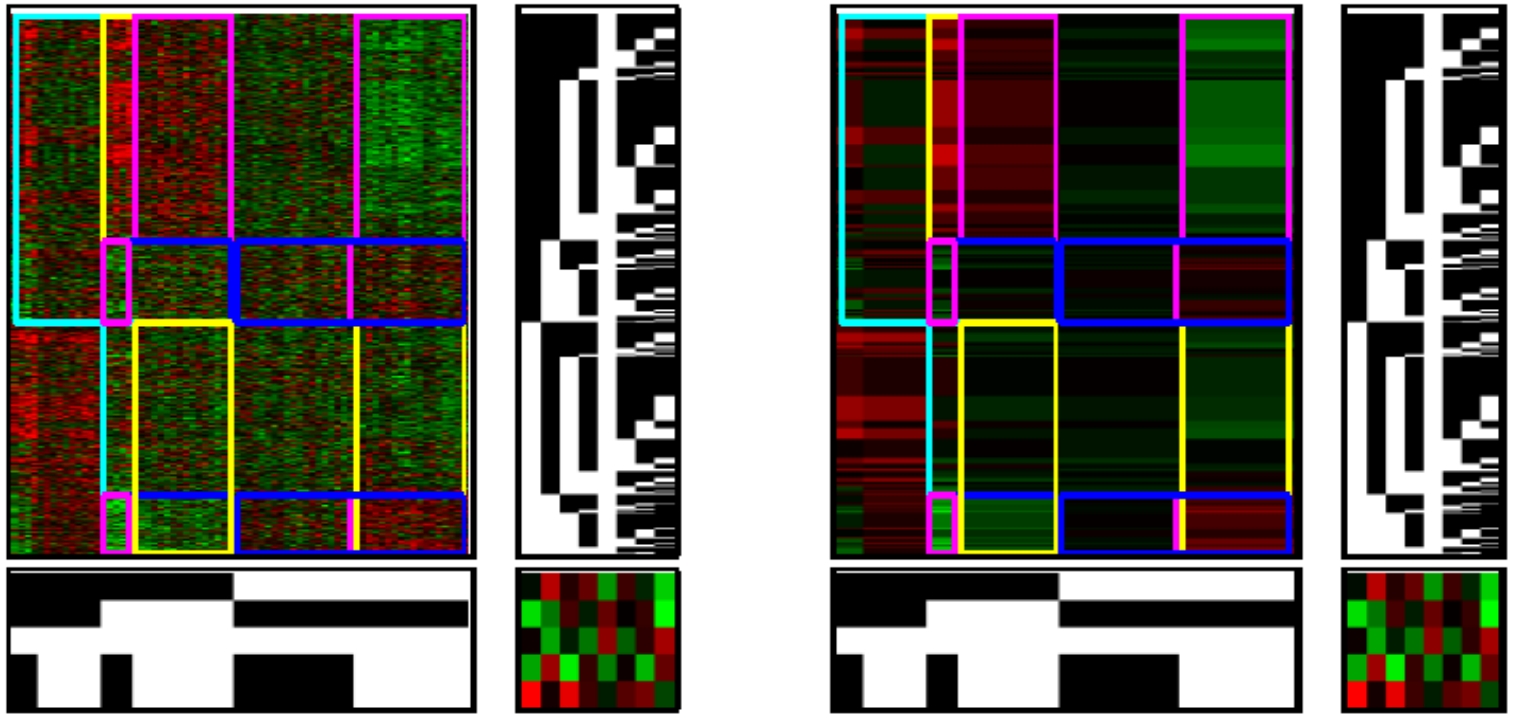
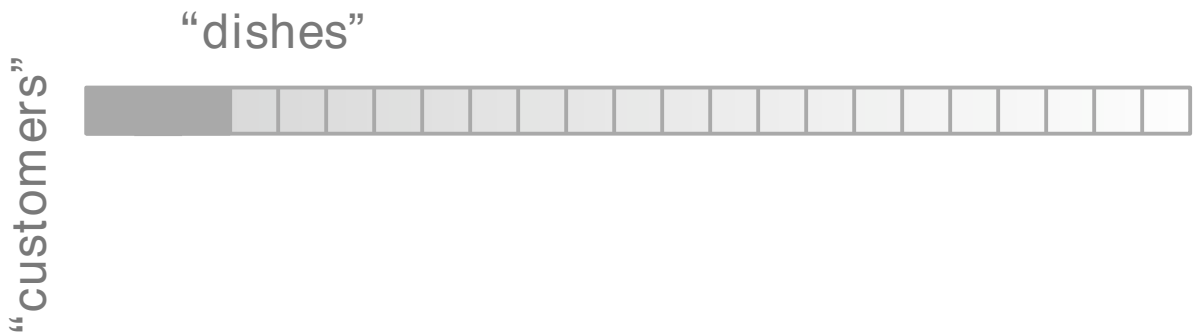


Figure 5: Gene expression results. **(A)** The top-left is $\mathbf{X}$ sorted according to contiguous features in the final $\mathbf{U}$ and $\mathbf{V}$ in the Markov chain. The bottom-left is $\mathbf{V}^\top$ and the top-right is $\mathbf{U}$. The bottom-right is $\mathbf{W}$. **(B)** The same as **(A)**, but the expected value of $\mathbf{X}$, $\hat{\mathbf{X}} = \mathbf{U}\mathbf{W}\mathbf{V}^\top$. We have hilighted regions that have both u_{ik} and v_{jl} on. For clarity, we have only shown the (at most) two largest contiguous regions for each feature pair.

(w/ Meeds, Roweis, Neal, 2006)

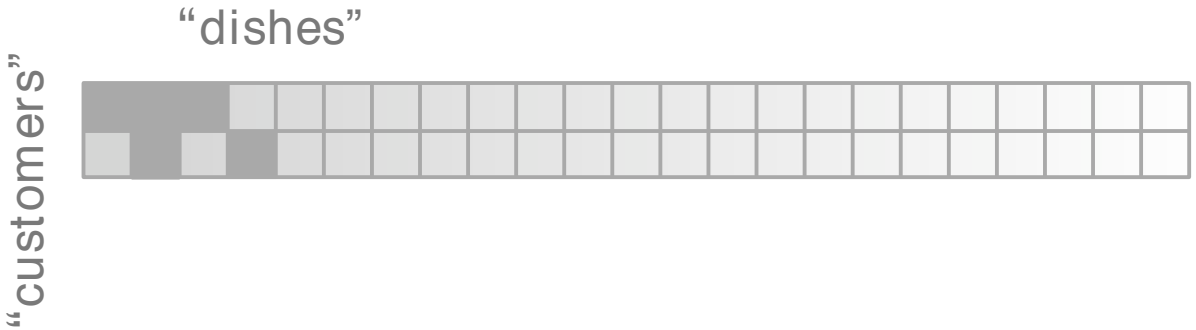
Modelling Graph Structure



Parameters:
α and β

customers = observed units
dishes = hidden units

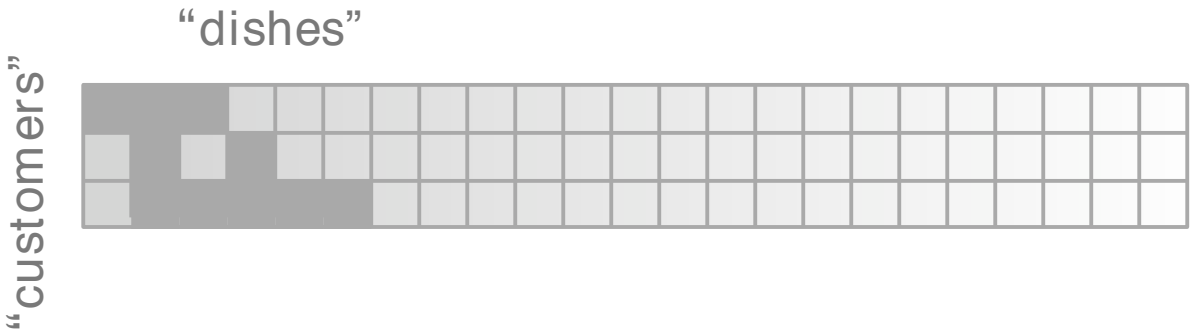
Modelling Graph Structure



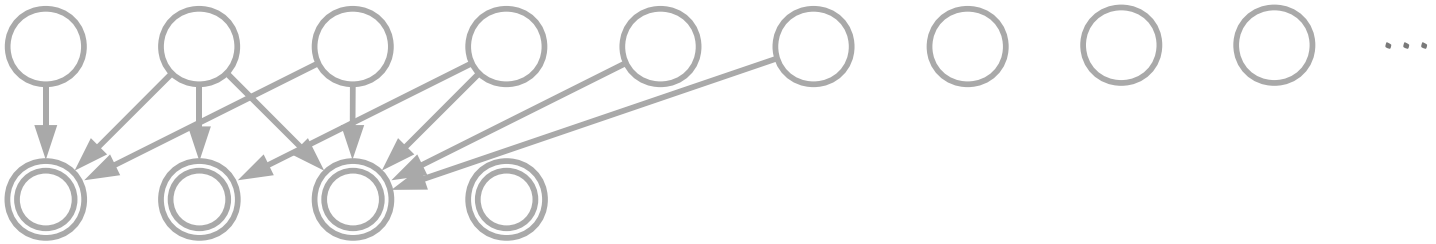
Parameters:
α and β

customers = observed units
dishes = hidden units

Modelling Graph Structure

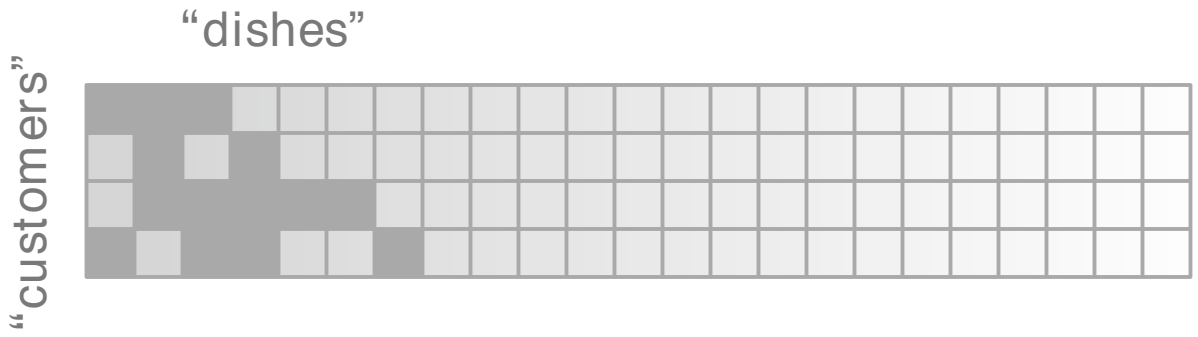


Parameters:
α and β

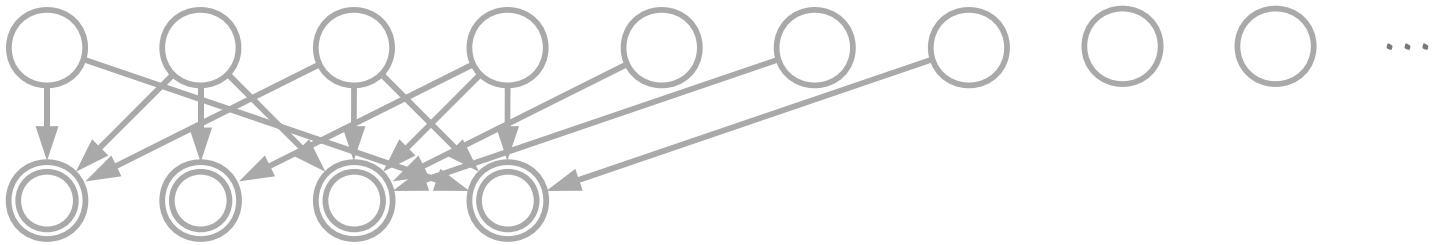


customers = observed units
dishes = hidden units

Modelling Graph Structure



Parameters:
α and β



customers = observed units
dishes = hidden units

Modelling Graph Structure

Inferring stroke localization from patient symptoms:

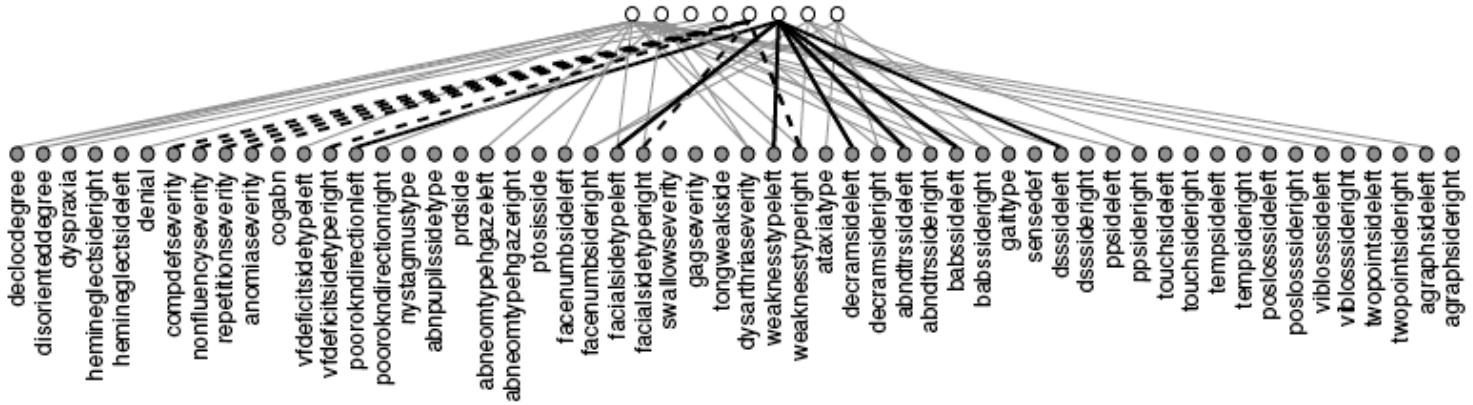


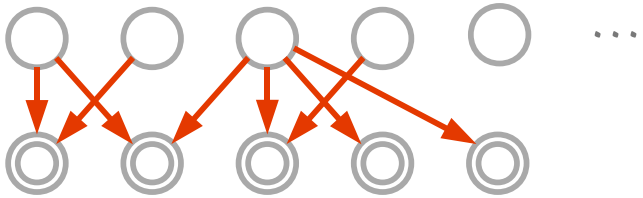
Figure 6: Causal structure with highest posterior probability. Two grouping of signs are highlighted. In solid black, we find a grouping of poor optokinetic nystagmus, lack of facial control, weakness, decreased rapid alternating movements, abnormal deep tendon reflexes, Babinski sign, and double simultaneous stimulation neglect, all on the left side, consistent with a right frontal/parietal infarct. In dashed black, we find a grouping of comprehension deficit, non-fluency, repetition, anomia, visual field deficit, facial weakness, and general weakness, with the latter three on the right side, generally consistent with a left temporal infarct.

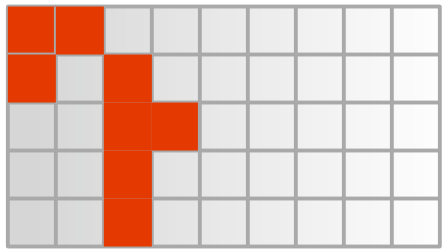
(50 stroke patients, 56 symptoms/signs)

A Non-Parametric Bayesian Method for Inferring Hidden Causes

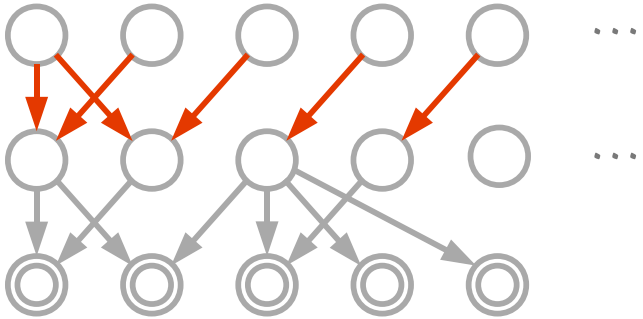
(w/ Frank Wood, Tom Griffiths, 2006)

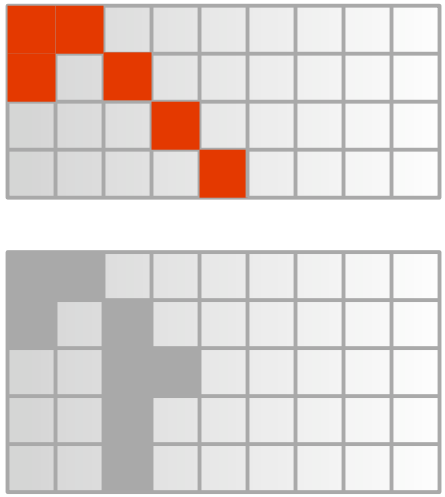
Learning Structure of Deep Sparse Graphical Models



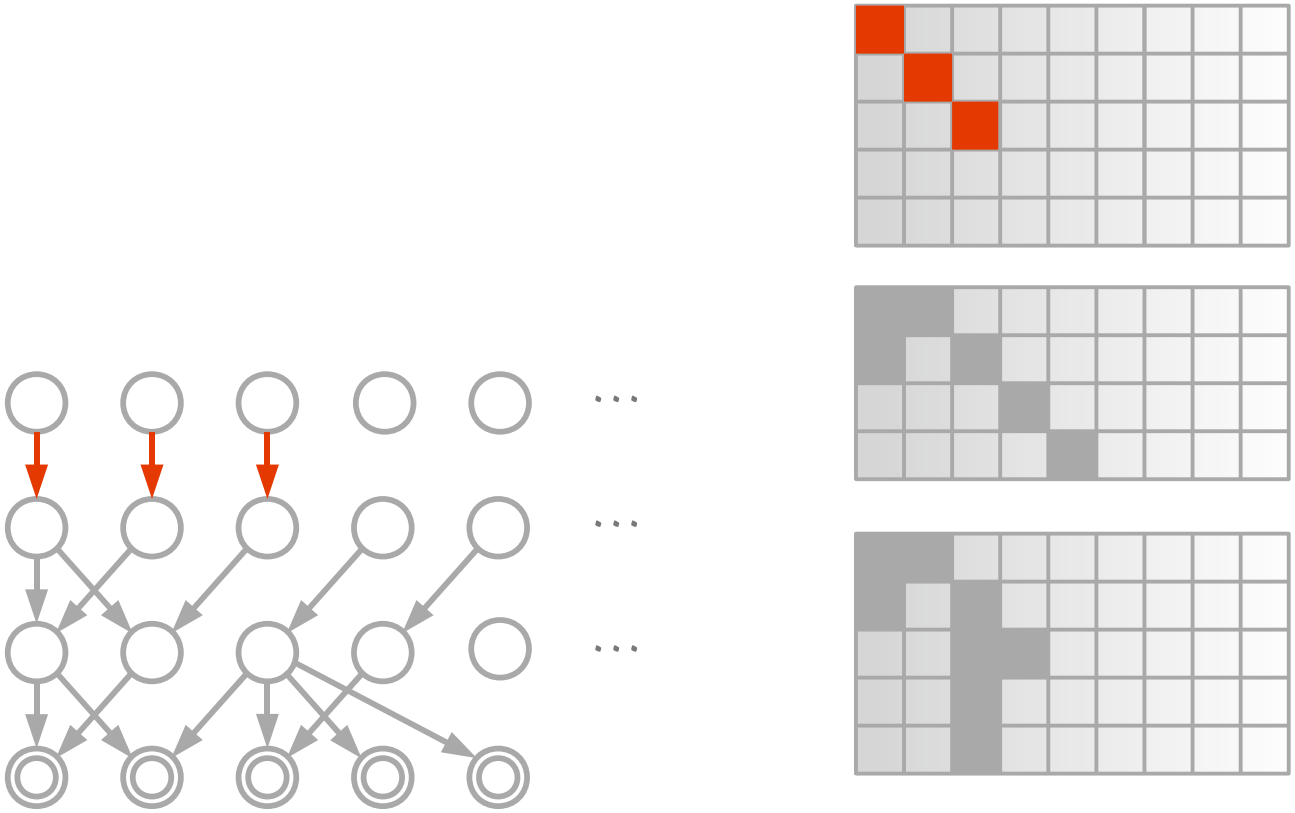


Learning Structure of Deep Sparse Graphical Models

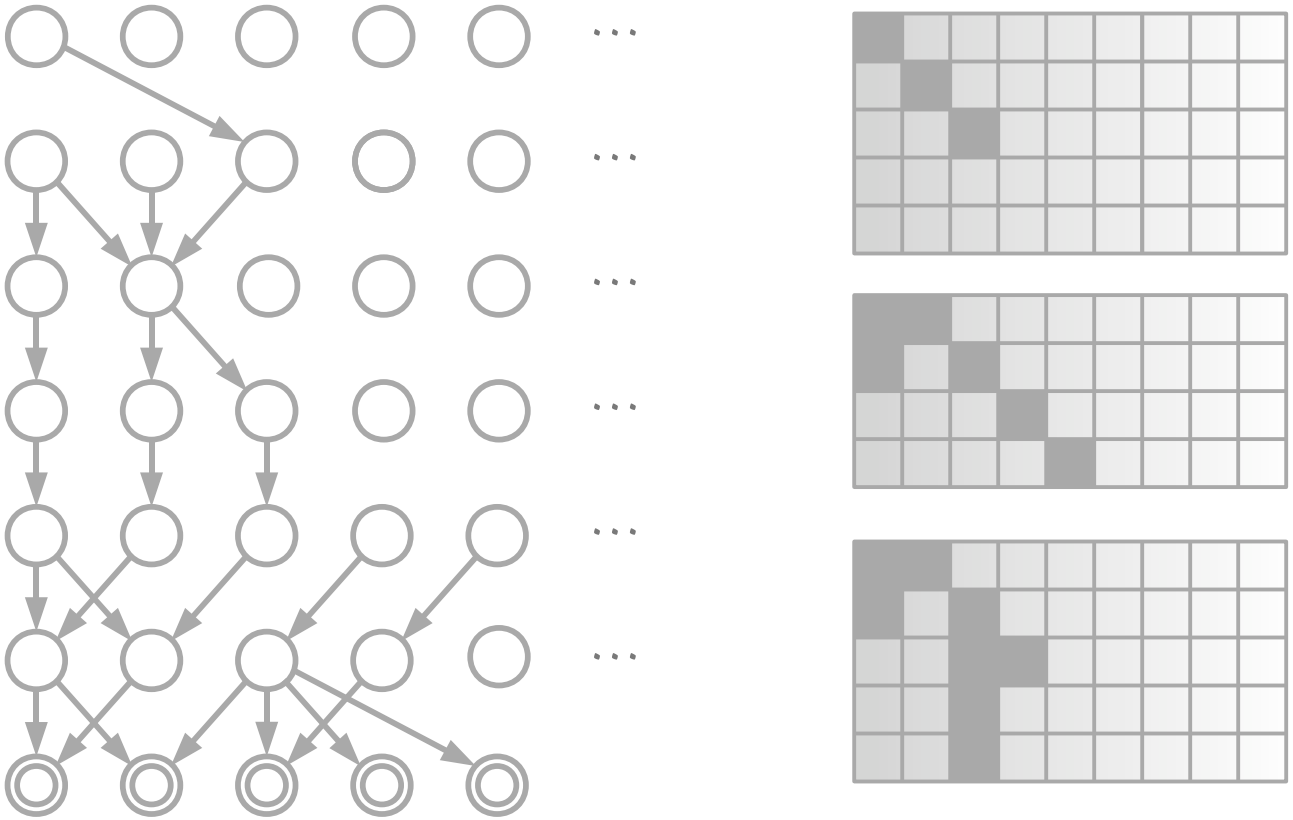




Learning Structure of Deep Sparse Graphical Models



Learning Structure of Deep Sparse Graphical Models

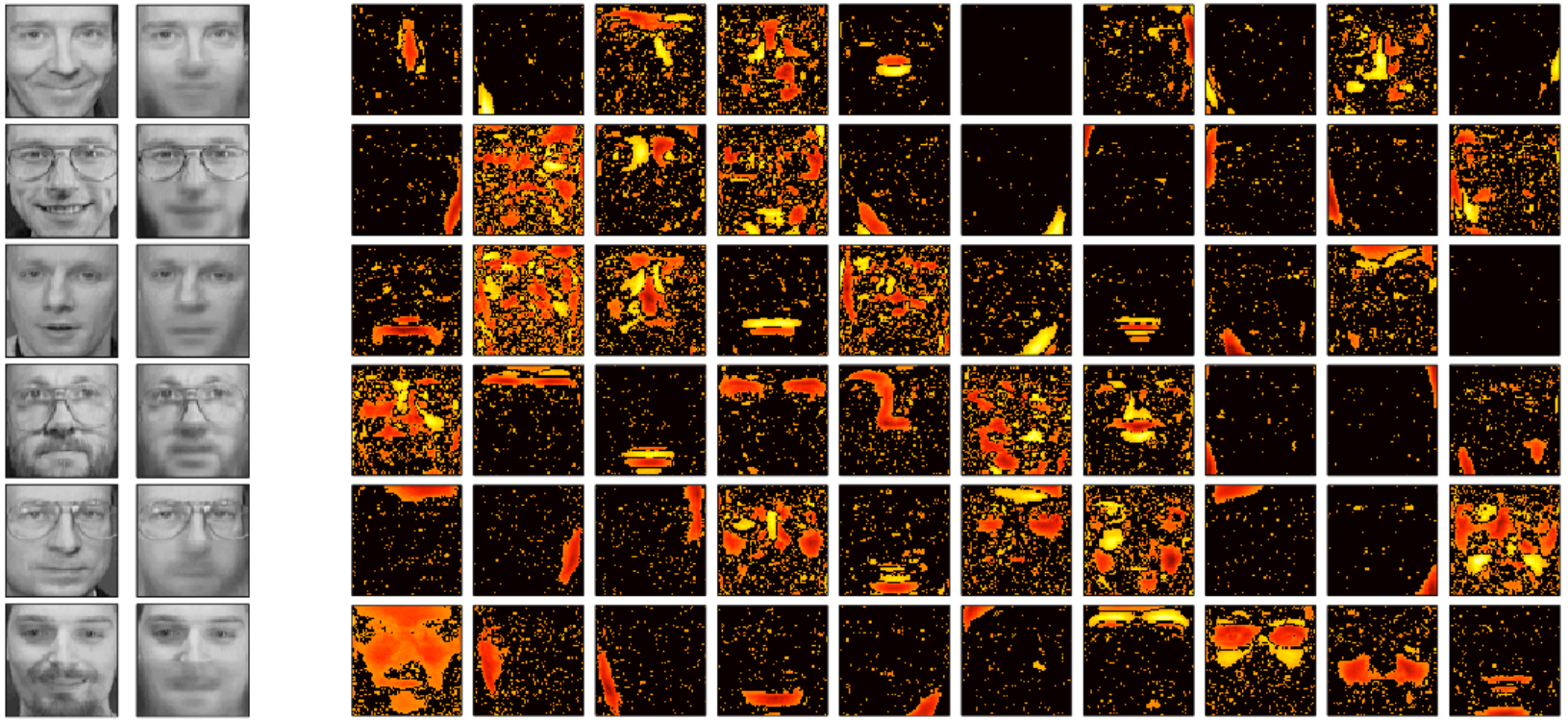


(w/ Ryan P. Adams, Hanna Wallach, 2010)

Learning Structure of Deep Sparse Graphical Models

Olivetti Faces: 350 + 50 images of 40 faces (64×64)
Inferred: 3 hidden layers, 70 units per layer.

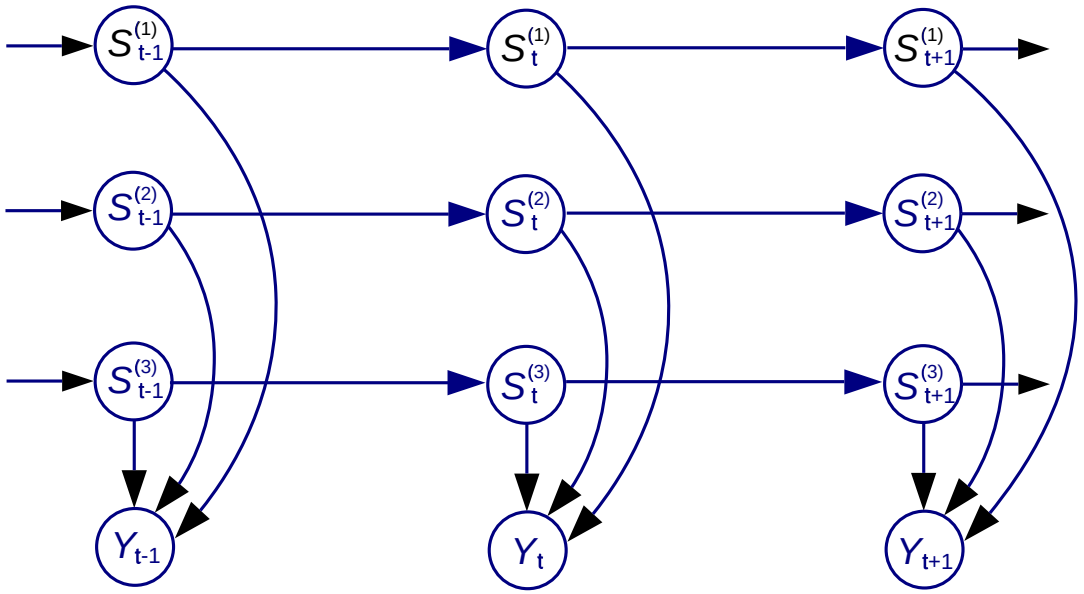
Reconstructions and Features:



Learning Structure of Deep Sparse Graphical Models

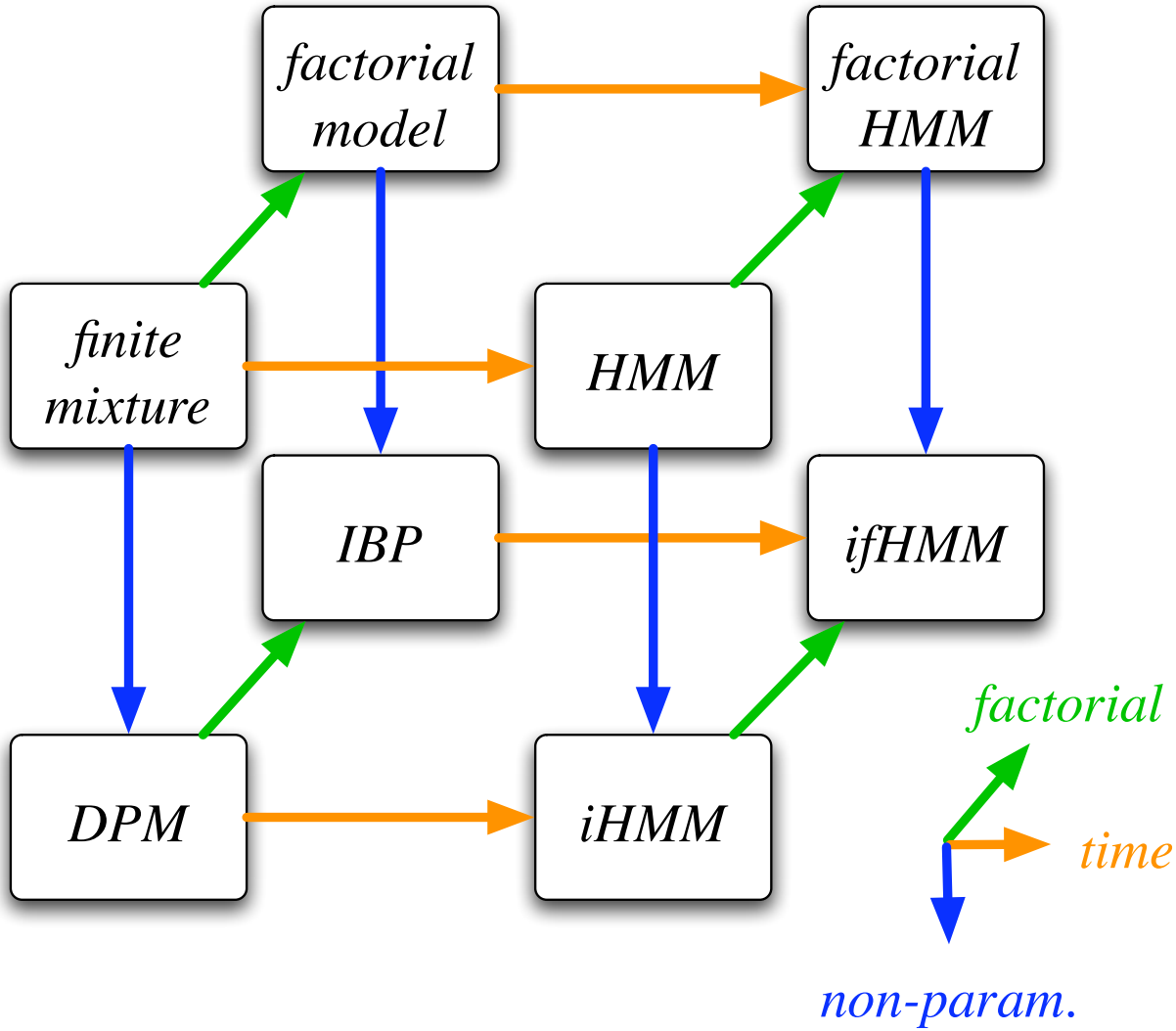
Fantasies and Activations:

Markov Indian Buffet Process and Infinite Factorial Hidden Markov Models



- Hidden Markov models (HMMs) represent the history of a time series using a **single** discrete state variable
- Factorial HMMs (fHMM) are a kind of HMM with a factored state representation (w/ Jordan, 1997)
- We can extend the Indian Buffet Process to time series and use it to define a non-parametric version of the fHMM (w/ van Gael, Teh, 2008)

The Big Picture:
Relations between some models



Summary

- Probabilistic modelling and Bayesian inference are two sides of the same coin
- Bayesian machine learning treats learning as a probabilistic inference problem
- Bayesian methods work well when the models are flexible enough to capture relevant properties of the data
- This motivates non-parametric Bayesian methods, e.g.:
  - Gaussian processes for regression
  - Dirichlet process mixtures for clustering
  - Infinite HMMs for time series modelling
  - Indian buffet processes for sparse matrices and latent feature modelling

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