Planting the Seeds of Probabilistic Thinking

Foundations | Tricks | Algorithms

Shakir Mohamed

Research Scientist, DeepMind

@shakir_za

DeepMind

Planting the Seeds of Probabilistic Thinking: Foundations, Tricks and Algorithms

Probabilistic machine learning approaches task of describing of data, to complex systems or our world using the language and tools of probability. Almost all of machine learning can be viewed in probabilistic terms, making probabilistic thinking fundamental. It is, of course, not the only view. But it is through this view that we can connect what we do in machine learning to every other computational science, whether that be in stochastic optimisation, control theory, operations research, econometrics, information theory, statistical physics or bio-statistics. For this reason alone, mastery of probabilistic thinking is essential.

The aim of this tutorial is to develop flexible and broad tools that will support your probabilistic thinking. Part 1, Foundations looks at the philosophy of machine learning, builds an understanding of the model-inference-algorithm paradigm, and the explores fundamental areas of machine learning - we'll look at deep learning, kernels and reinforcement learning. Part 2 Tricks, will look at 6 individual probabilistic problems and a tricks to solve them, using these tricks to develop flexibility in our thinking. Part 3: Algorithms will look at how the foundations and tricks combine to develop machine learning algorithms, with a specific focus on the area of deep generative models.

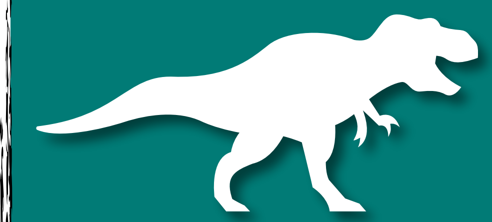
Principles to Products

Applications	Assistive Technology	Advancing Science	Climate and Energy	Healthcare	Fairness and Safety	Autonomous systems
Reasoning	Planning	Explanation	Rapid Learning	World Simulation	Objects and Relations	
Information	Uncertainty	Information Gain	Causality	Prediction		
Principles	Probability Theory	Bayesian Analysis	Hypothesis Testing	Estimation Theory	Asymptotics	

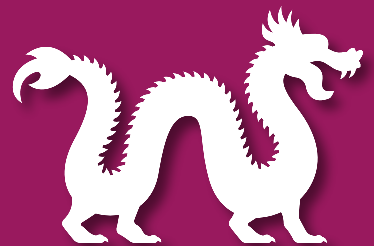
Planting the Seeds of Probabilistic Thinking

Part I: Foundations

Learning Objectives



1. Language to think about the Philosophy of Machine Learning



2. Understand the Model-Inference-Algorithm paradigm

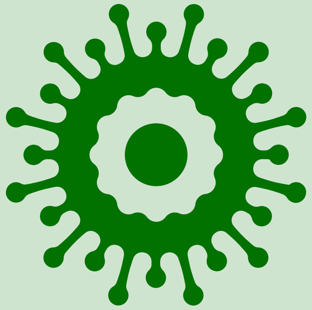
3. Use probabilistic thinking applied to problems in supervised, unsupervised, and reinforcement learning.

Probability

Some Definitions for probability

Statistical Probability
Frequency ratio of items

Logical Probability
Degree of confirmation of a hypothesis based on logical analysis



Probability as Propensity
Probability used for predictions

Subjective Probability
Probability as a degree of belief

Probability is sufficient for the task of reasoning under uncertainty

Probability

Probability as a Degree of Belief

Probability is a measure of the belief in a proposition **given** evidence.
A description of a state of knowledge.

No such thing as **the probability** of an event, since the value depends on the evidence used.

Inherently subjective in that it depends on the believer's information

Different observers with different information will have different beliefs.

Probabilistic Quantities

Probability $p(\mathbf{x}) \qquad p^*(\mathbf{x}) \qquad q(\mathbf{x})$

Conditions $p(\mathbf{x}) \geq 0 \qquad \int p(\mathbf{x}) d\mathbf{x} = 1$

Bayes Rule

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

Parameterisation

$$p_\theta(\mathbf{x}|\mathbf{z}) \equiv p(\mathbf{x}|\mathbf{z}; \boldsymbol{\theta})$$

Expectation

$$\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[f(\mathbf{x}; \phi)] = \int p_\theta(\mathbf{x}|\mathbf{z}) f(\mathbf{x}; \phi) d\mathbf{x}$$

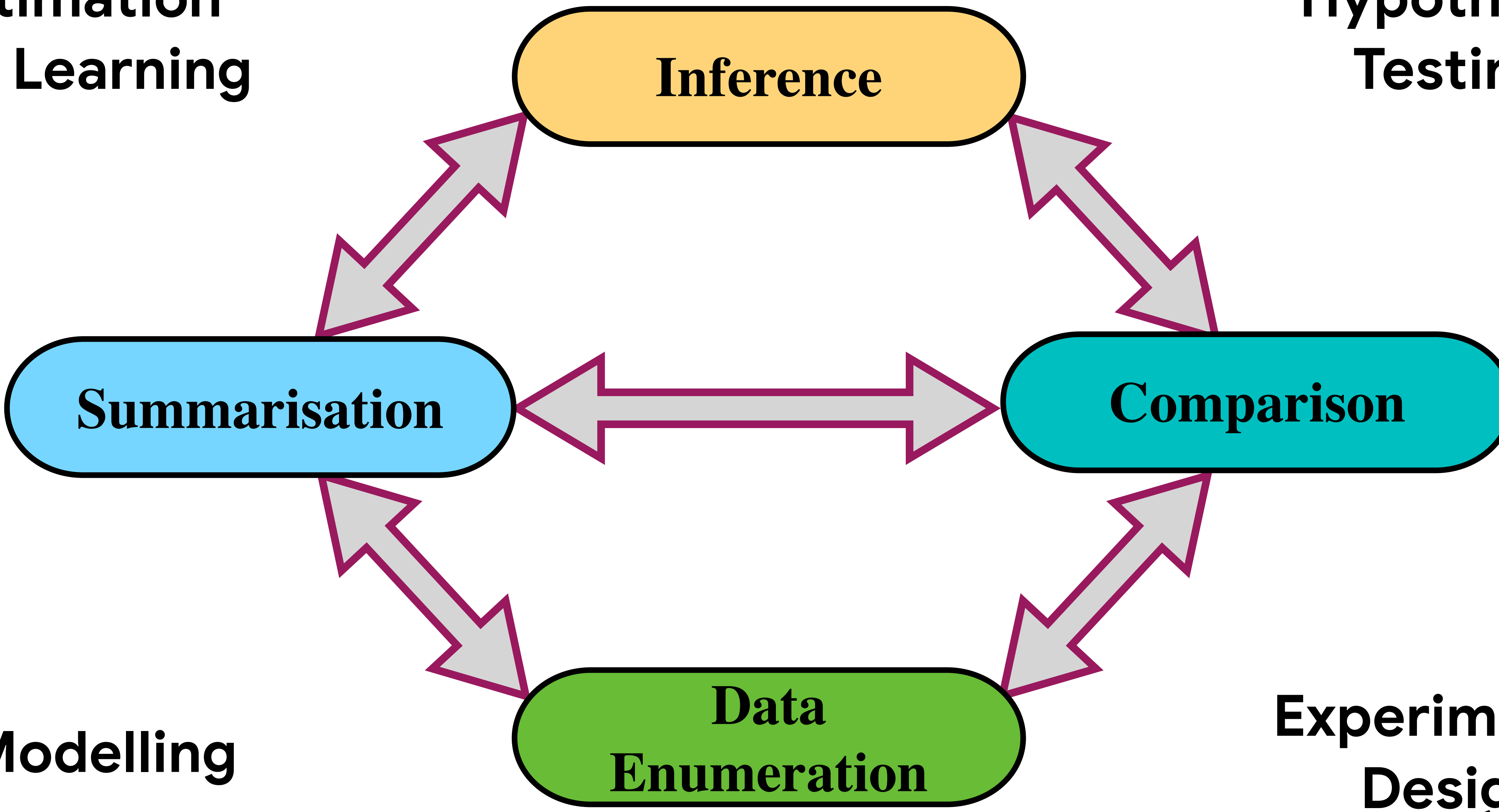
Gradient

$$\nabla_\phi f(\mathbf{x}; \phi) = \frac{\partial f(\mathbf{x}; \phi)}{\partial \phi}$$

Statistical Operations

Estimation and Learning

Hypothesis Testing



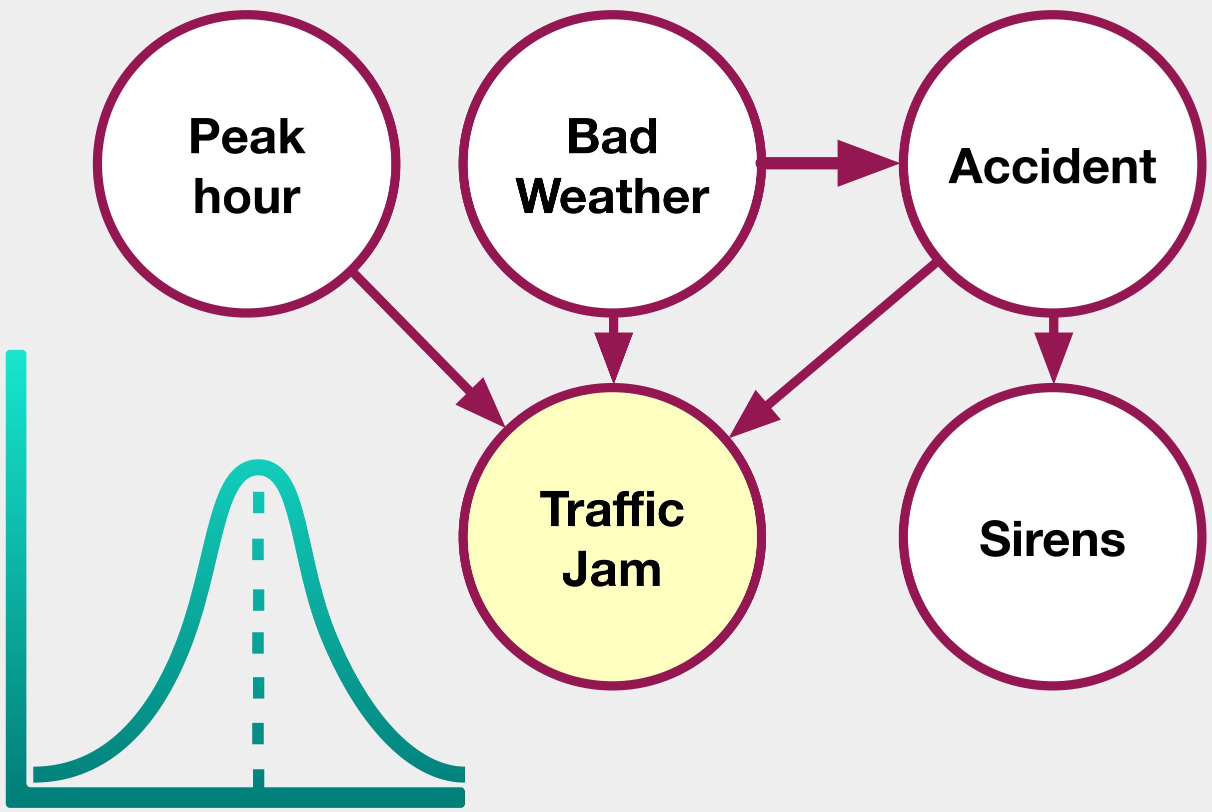
Modelling

Experimental Design

Probabilistic Models

Model: Description of the world, of data, of potential scenarios, of processes.

A probabilistic model writes out these models using the language of probability



prob(traffic Jam)

prob(sirens | Accident)

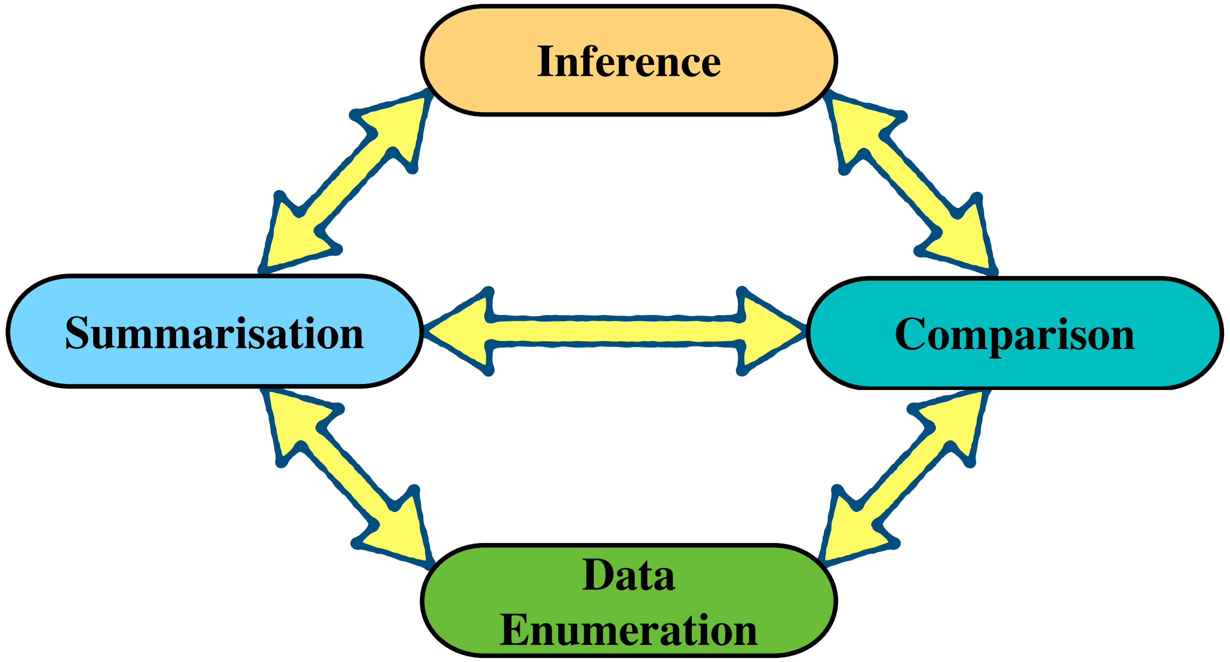
prob(peak hour | Traffic Jam)

Most models in machine learning are probabilistic.

Probabilistic models let you learn probability distributions of data.

You can choose what to learn: Just the mean. Or the entire distribution.

Centrality of Inference



Artificial General Intelligence will be the refined instantiation of these statistical operations.

The core questions of AGI will be those of probabilistic inference

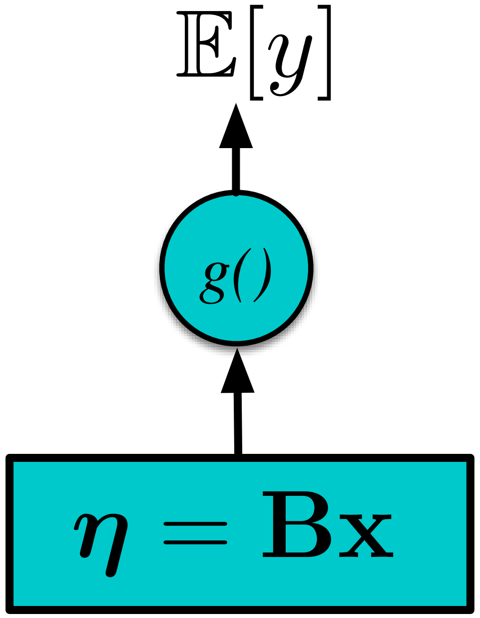
Linear Regression

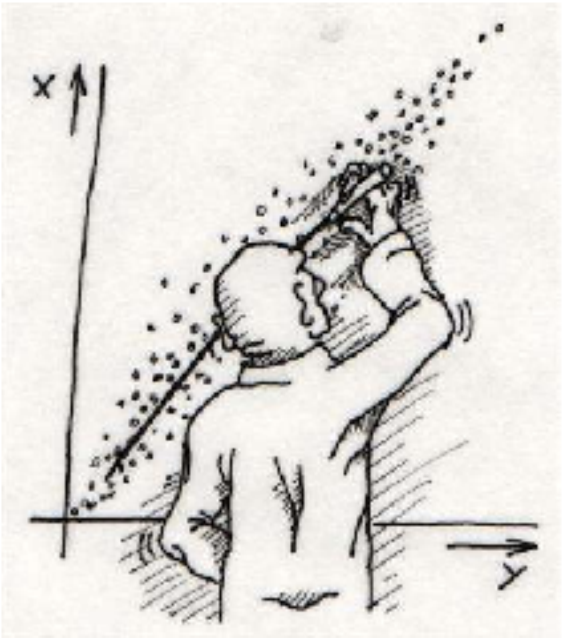
Generalised Linear Regression

$$\eta = \mathbf{w}^\top \mathbf{x} + b$$

$$p(y|\mathbf{x}) = p(y|g(\eta);\theta)$$

- The basic function can be any linear function, e.g., affine, convolution.
- g(.) is an **inverse link function** that we'll refer to as an activation function.





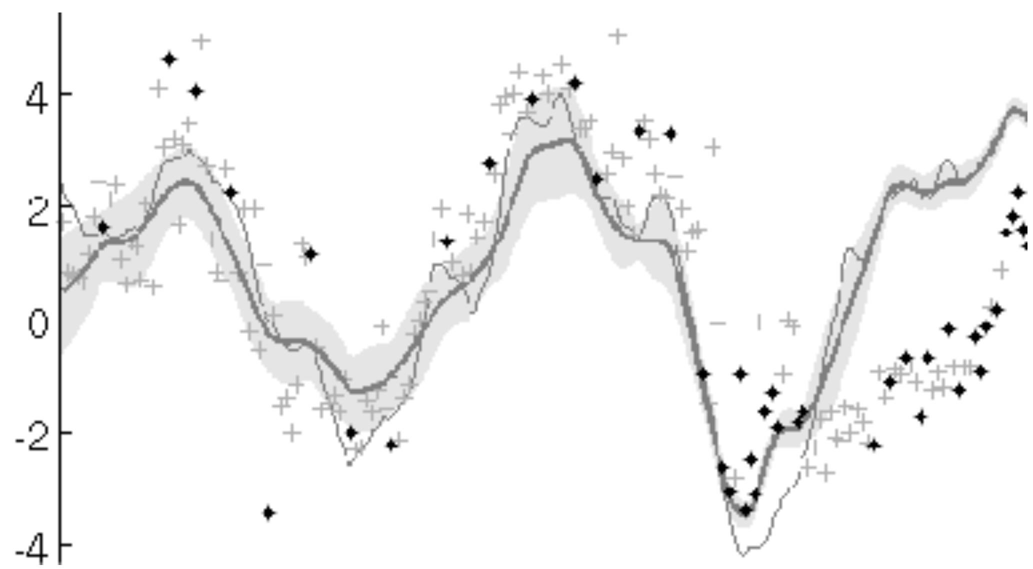
Target	Regression	Link	Inv link	Activation
Real	Linear	Identity	Identity	
Binary	Logistic	Logit $\log \frac{\mu}{1-\mu}$	Sigmoid $\frac{1}{1+\exp(-\eta)}$	Sigmoid
Binary	Probit	Inv Gauss CDF $\Phi^{-1}(\mu)$	Gauss CDF $\Phi(\eta)$	Probit
Binary	Gumbel	Compl. log-log $\log(-\log(\mu))$	Gumbel CDF $e^{-e^{-x}}$	
Binary	Logistic		Hyperbolic Tangent $\tanh(\eta)$	Tanh
Categorical	Multinomial		Multin. Logit $\frac{\eta_i}{\sum_j \eta_j}$	Softmax
Counts	Poisson	$\log(\mu)$	$\exp(\nu)$	
Counts	Poisson	$\sqrt(\mu)$	ν^2	
Non-neg.	Gamma	Reciprocal $\frac{1}{\mu}$	$\frac{1}{\nu}$	
Sparse	Tobit		max $\max(0;\nu)$	ReLU
Ordered	Ordinal		Cum. Logit $\sigma(\phi_k - \eta)$	

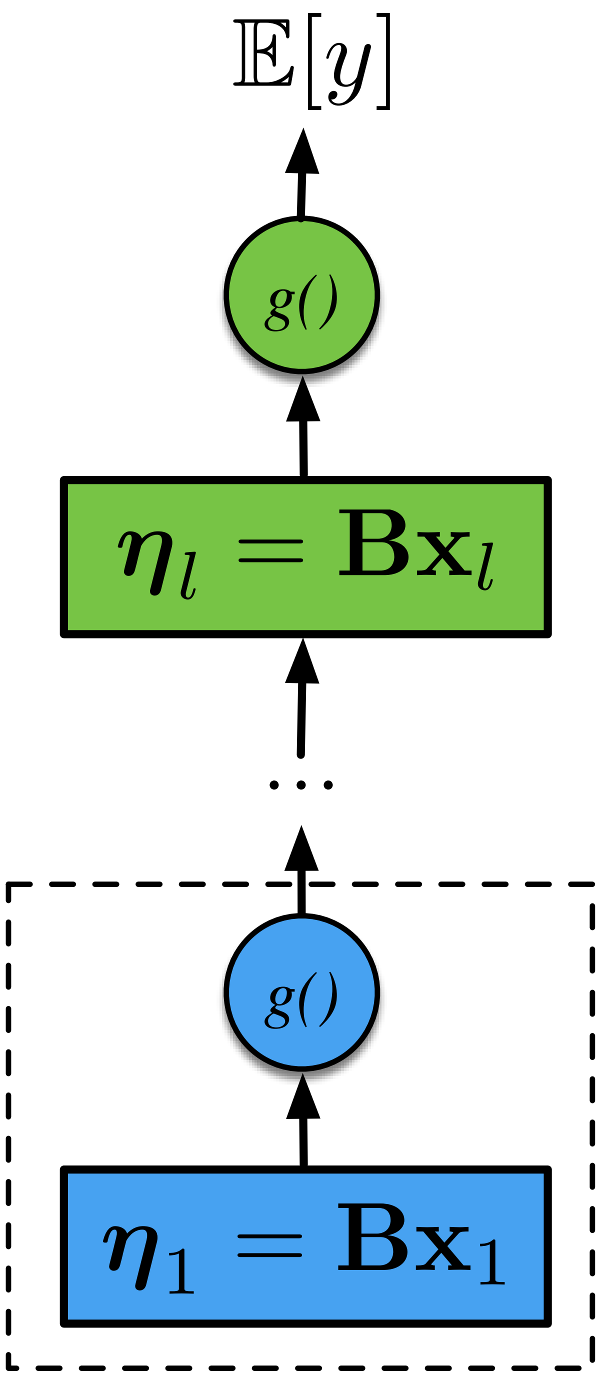
Optimise the negative log-likelihood

$$\mathcal{L} = -\log p(y|g(\eta);\theta)$$

Deep Networks

Recursive Generalised Linear Regression





- Recursively compose the basic linear functions.
- Gives a deep neural network.

$$\mathbb{E}[y] = h_L \circ \ldots \circ h_l \circ h_0(\mathbf{x})$$

A general, flexible framework for building non-linear, parametric models

Likelihood

Probabilistic Model

$$p(y|\mathbf{x}) = p(y|h(\mathbf{x}); \boldsymbol{\theta})$$

Likelihood function

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_n \log p(y_n|\mathbf{x}_n; \boldsymbol{\theta})$$

Likelihood of parameters

Prescribed Likelihoods

Efficient Estimators
- Statistically efficient (Cramer-Rao lower bound)
- Asymptotically unbiased, consistent
- Maximum entropy (principle of indifference)

Tests with Good Power
- Likelihood ratio tests
- Can construct small confidence regions

Widely-applicable
- Handle data that is incompletely observed, distorted, samples with bias
- Can offset or correct these issues.

Pool Information
- Combine different data sources
- Knowledge outside the data can be used, like constraints on domain or prior probabilities.

Misspecification: Inefficient estimates; or confidence intervals/tests can fail completely.

Estimation Theory

Probabilistic Model

$$p(y|\mathbf{x}) = p(y|h(\mathbf{x});\boldsymbol{\theta})$$

Likelihood function

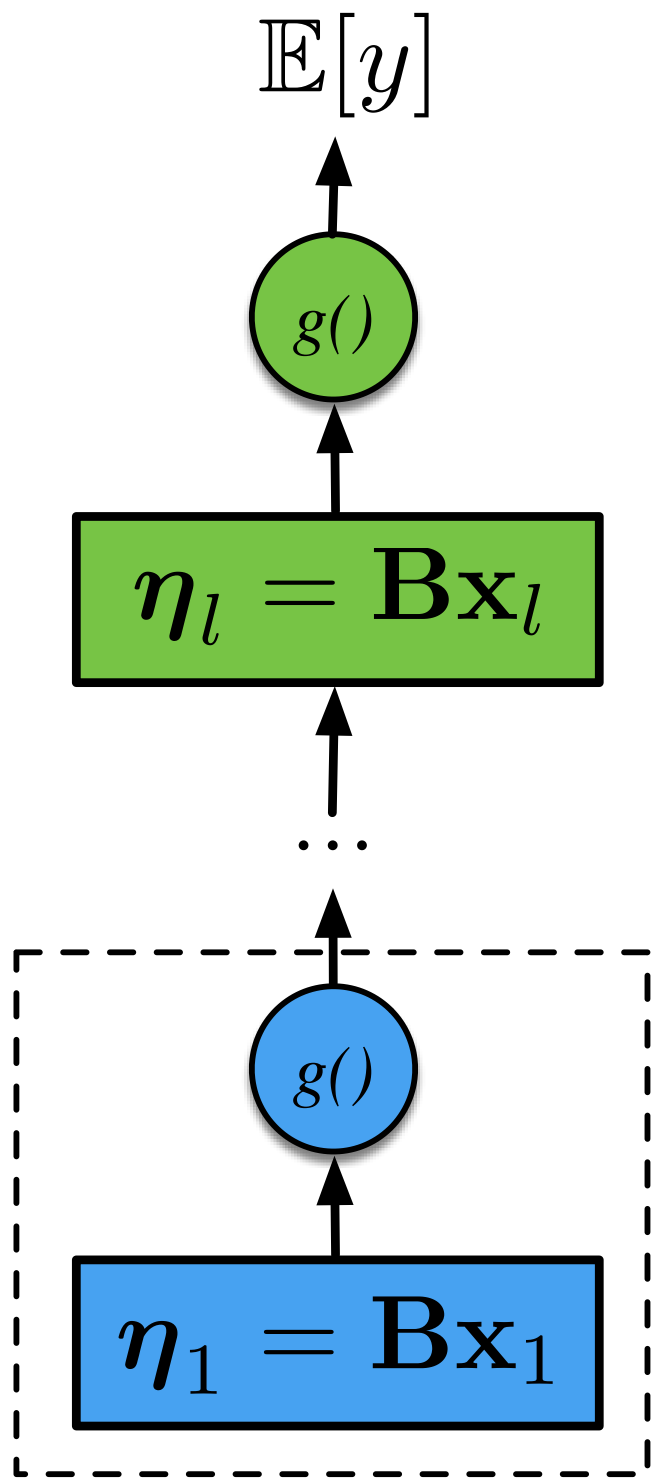
$$\mathcal{L}(\boldsymbol{\theta}) = \sum_n \log p(y_n|\mathbf{x}_n;\boldsymbol{\theta})$$

Maximum Likelihood

Optimisation Objective

$$\arg\max_{\theta} \mathcal{L}(\boldsymbol{\theta})$$

- Straightforward and natural way to learn parameters
- Can be biased in finite sample size, e.g., Gaussian variances with N and N-1.
- Easy to observe **overfitting** of parameters.



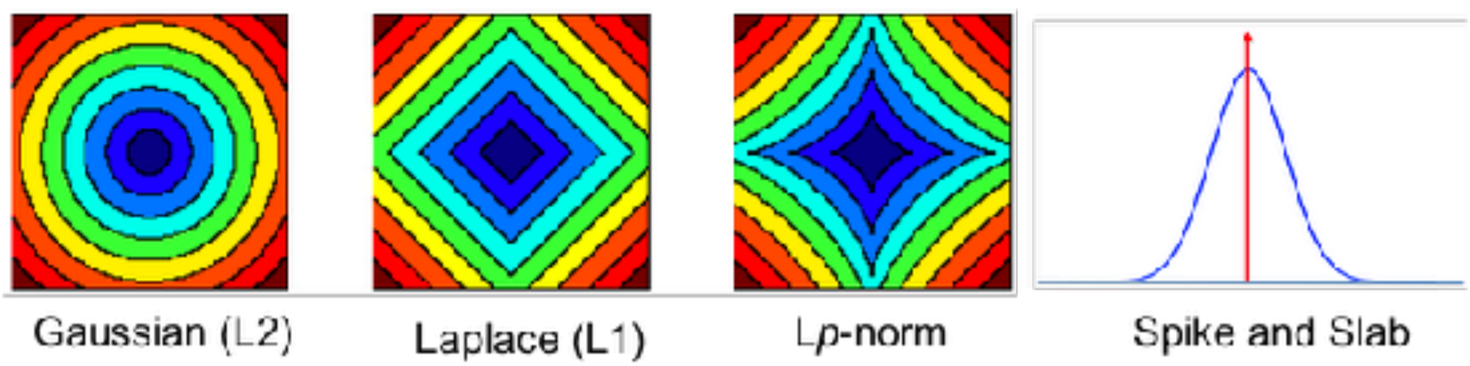
Estimation Theory

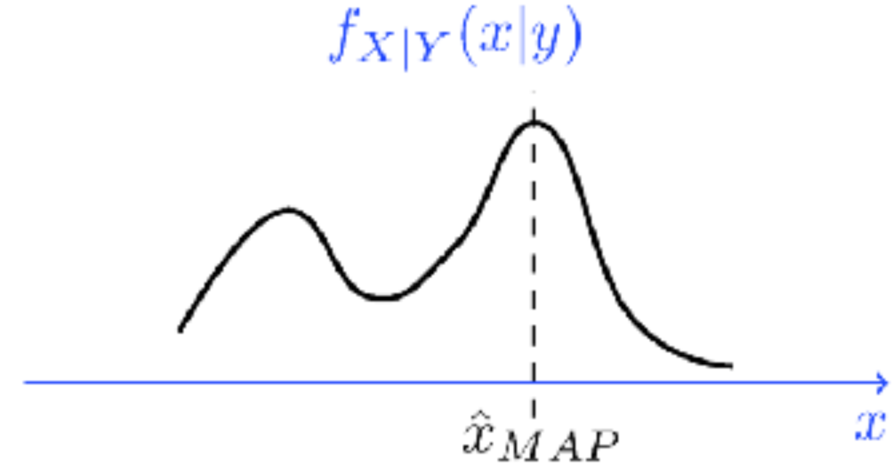
Probabilistic Model

$$p(\boldsymbol{\theta}|y, \mathbf{x}) \propto p(y|h(\mathbf{x}); \boldsymbol{\theta})p(\boldsymbol{\theta})$$

Likelihood function

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_n \log p(y_n|\mathbf{x}_n; \boldsymbol{\theta}) + \tfrac{1}{\lambda}\mathcal{R}(\boldsymbol{\theta}))$$





Maximum a Posteriori (MAP)

Optimisation Objective

$$\arg\max_{\theta} \mathcal{L}(\boldsymbol{\theta})$$

- Generalises the MLE (uniform prior)
- **Shrinkage**: shrink parameters back to initial beliefs.
- Not every regulariser corresponds a valid probability distribution.

Regularisation

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_n \log p(y_n|\mathbf{x}_n; \boldsymbol{\theta}) + \frac{1}{\lambda}\mathcal{R}(\boldsymbol{\theta}))$$

✦ **Regularisation** is essential to overcome the limitations of maximum likelihood estimation.

✦ **Other names:** Regularisation, penalised regression, shrinkage.

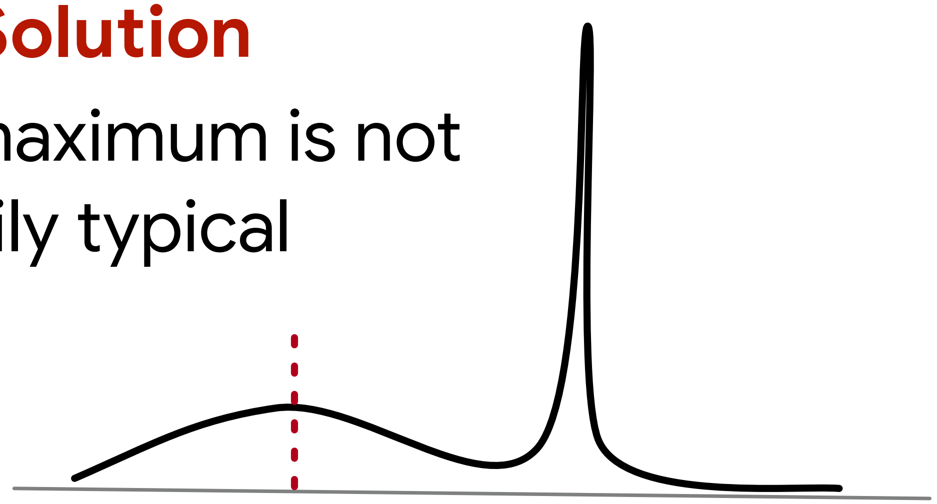
A wide range of available regularisation techniques:

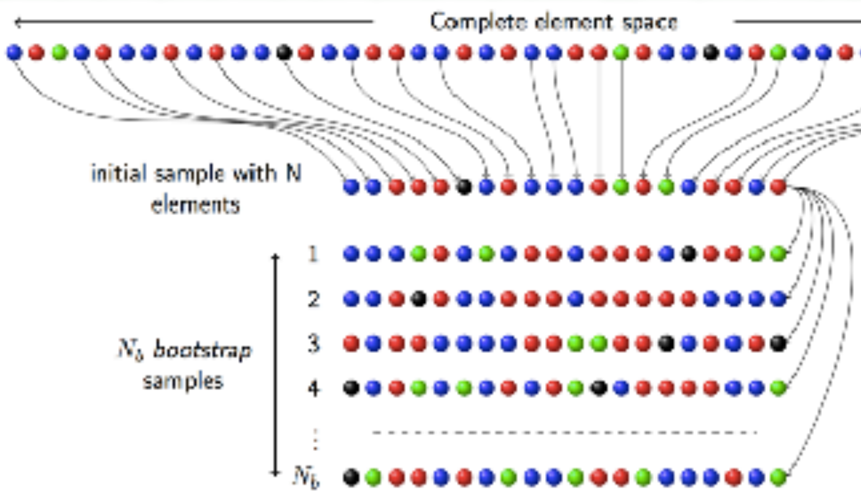
- Large data sets
- Input noise/jittering and data augmentation/expansion.
- L2 /L1 regularisation (Weight decay, Gaussian prior)
- Binary or Gaussian Dropout
- Batch normalisation

MAP Estimation

Type of Solution

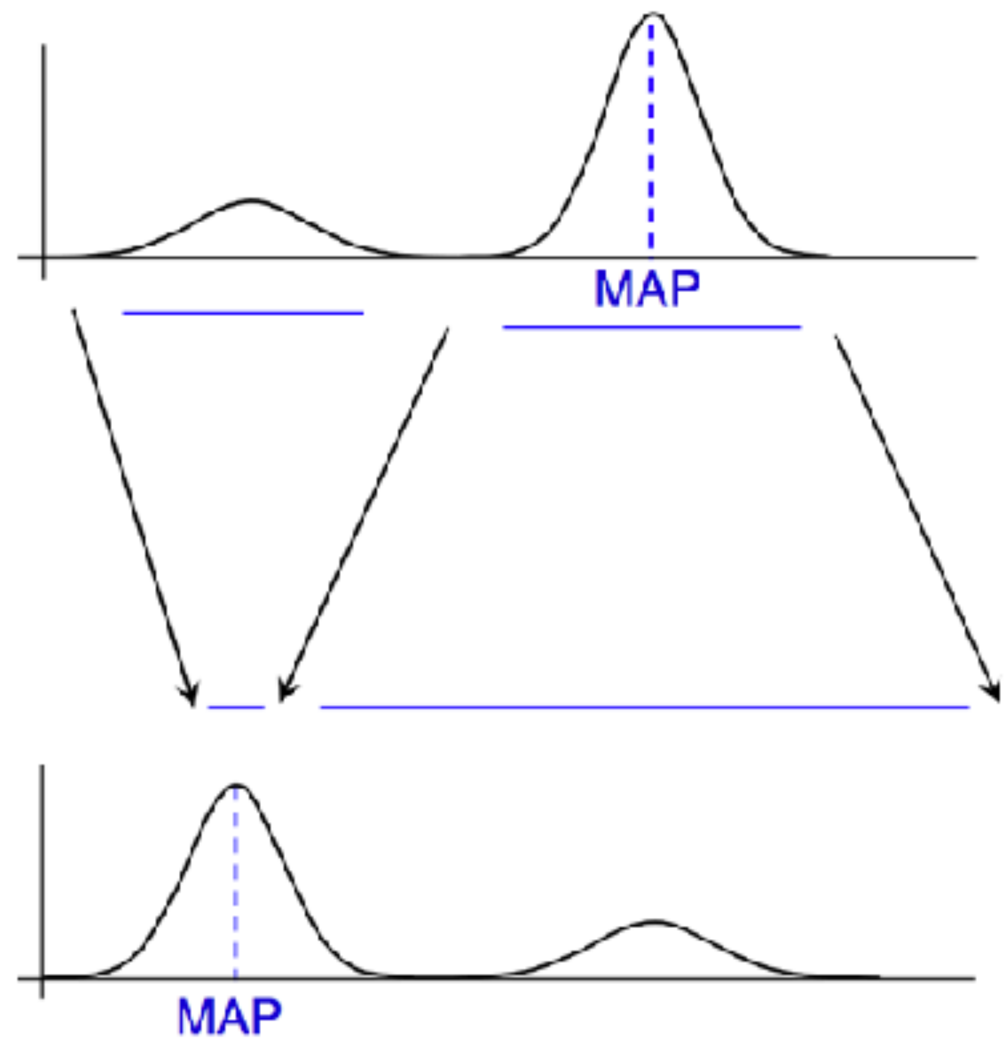
What is maximum is not necessarily typical





Uncertainty

Can be reported using confidence intervals or bootstrap estimates.



Parameterisation sensitive

Location of max will change depending on parameterisation

Invariant MAP

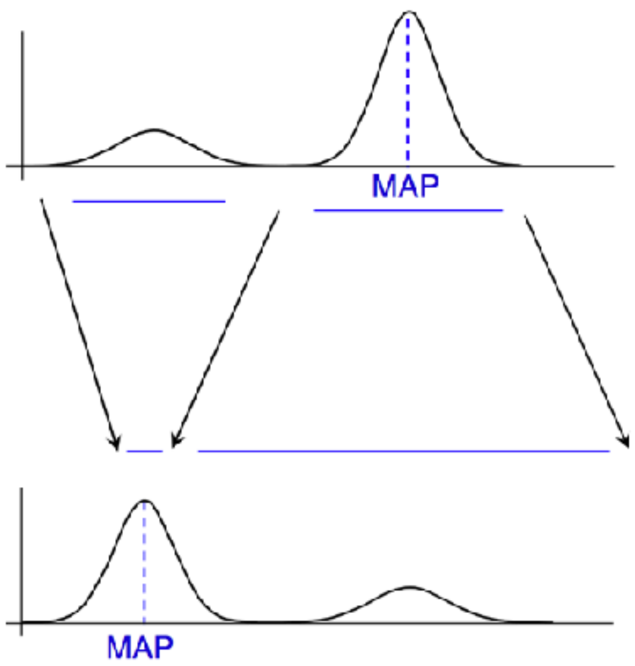
Popular Example

$y \in 0,1; \quad 0 \leq \mu \leq 1$

Change of variables	$p(\phi) = p(\mu)\left\|\frac{d\mu}{d\phi}\right\|$	**Bernoulli**	$p(y=1\|\mu) = \mu$
		Uniform	$p(\mu) = 1$

Mode of the prior $\hat{\phi}_{MAP} = \arg\max_{\phi \in [0,1]} p(\phi)$

	Parameterisation 1	**Parameterisation 2**
Transform	$\mu = \phi^2$	$\mu = 1 - (1-\phi)^2$
New prior	$p(\phi) = 2\phi$	$p(\phi) = 2(1-\phi)$
MAP Est.	$\hat{\phi}_{MAP} = 1$	$\hat{\phi}_{MAP} = 0$



Clear sensitivity: Sensitive to units, affects interpretability, affects gradients, learning stability, design of models.

Invariant MAP

Use a modified probabilistic model that removes sensitivity

Invariant MAP $p(y|h(\mathbf{x});\boldsymbol{\theta})p(\boldsymbol{\theta})|\mathcal{I}(\boldsymbol{\theta})|^{\frac{1}{2}}$

- Use the Fisher information
- Connection to the natural gradients and trust-region optimisation.
- Uninformative priors.

Proposed solutions have not fully dealt with the underlying issues.

Bayesian Analysis

Issues arise as a consequence of:

- Reasoning only about the most likely solution, and
- Not maintaining knowledge of the underlying variability (and averaging over this).

Motivates learning more than the mean.
This is the core of a Bayesian philosophy.

$$p(\boldsymbol{\theta}|y, \mathbf{x}) \propto p(y|h(\mathbf{x}); \boldsymbol{\theta})p(\boldsymbol{\theta})$$

Pragmatic Bayesian Approach for
Probabilistic Reasoning in Deep Networks.
(and all of machine learning)

Bayesian reasoning over some, but not all parts of our models (yet).

Bayesian Analysis

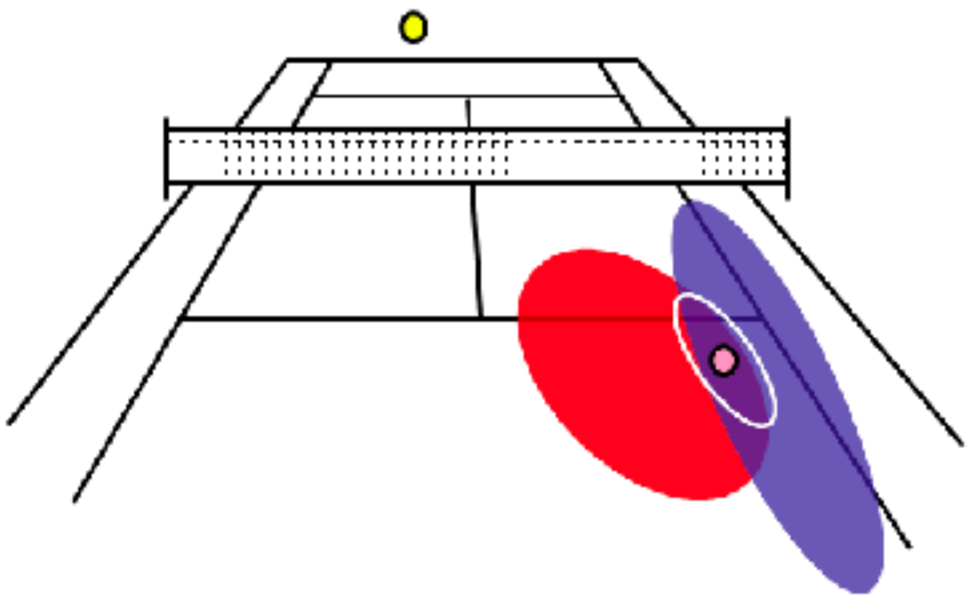
Interested in reasoning about two important quantities

Evidence $$p(y|\mathbf{x}) = \int p(y|h(\mathbf{x});\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$$

Posterior $$p(\boldsymbol{\theta}|y,\mathbf{x}) \propto p(y|h(\mathbf{x});\boldsymbol{\theta})p(\boldsymbol{\theta})$$

- In Bayesian analysis, things that are *not. observed* must be integrated over - averaged out.
- This makes computation difficult.
- Integration is the central operation.

Intractable Integrals: Will often see this phrasing.
- Don't know the integral in closed form
- Very high-dimensional quantities and can't compute (e.g., using quadrature)



Learning and Inference

Statistics, no distinction between learning and inference - only inference (or **estimation**).

Machine learning makes a distinction between **inference and learning**:

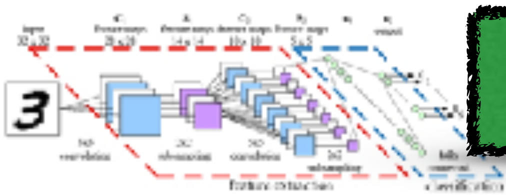
- **Inference**: reason about (and compute) unknown probability distributions.
- (Parameter) **Learning** is finding point estimates of quantities in the model.

Bayesian statistics, all quantities are probability distributions, so there is only the problem of **inference**.

Software engineering, **inference** is the forward evaluation of a trained model (to get predictions).

Decision making and AI, refer to **learning** in general as the means of understanding and acting based on past experience (data).

Two Streams of ML



Deep Learning

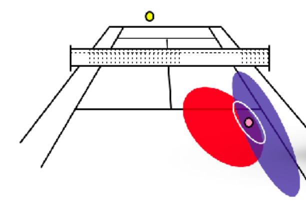
+ Rich non-linear models for classification and sequence prediction.

+ Scalable learning using stochastic approximation and conceptually simple.

+ Easily composable with other gradient-based methods

- Only point estimates

- Hard to score models, do selection and complexity penalisation.



Bayesian Reasoning

- Mainly conjugate and linear models

- Potentially intractable inference, computationally expensive or long simulation time.

+ Unified framework for model building, inference, prediction and decision making

+ Explicit accounting for uncertainty and variability of outcomes

+ Robust to overfitting; tools for model selection and composition.

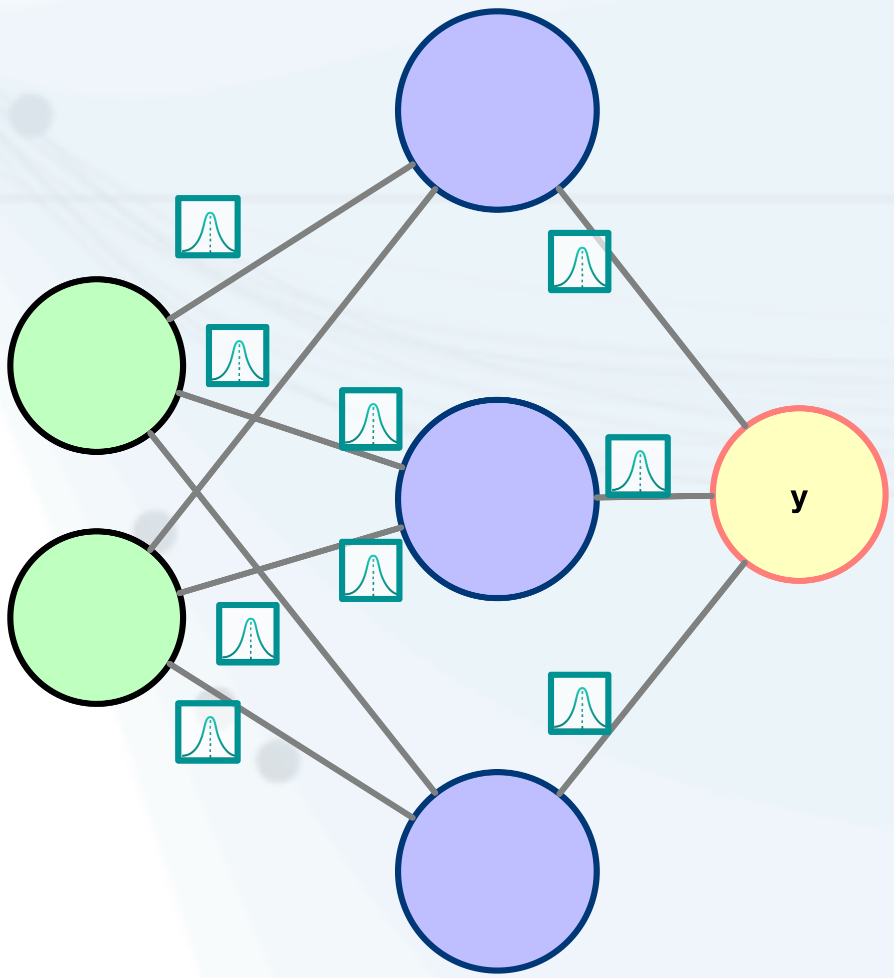
Natural to consider the marriage of these approaches: Bayesian Deep Learning

Bayesian Regression

Probabilistic models over functions

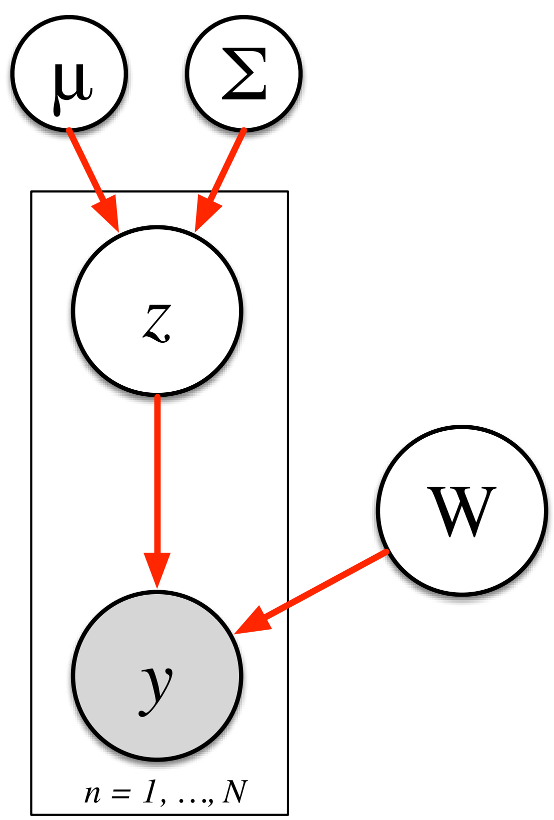
Prior	$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}\|0, \mathbf{I})$
Observation model	$p(y\|\mathbf{x}, \boldsymbol{\theta}) = Categorical(\pi(\mathbf{x}; \boldsymbol{\theta}))$
Posterior	$p(\boldsymbol{\theta}\|y, \mathbf{x})$

- Ways of learning distributions over functions and maintaining uncertainty over functions.
- Difficult in parametric models (like deep networks) because of high-dimensional parameter space.
- Many ways to learn the posterior distribution. Focus of Part III



Density Estimation

Learn probability distributions over the data itself

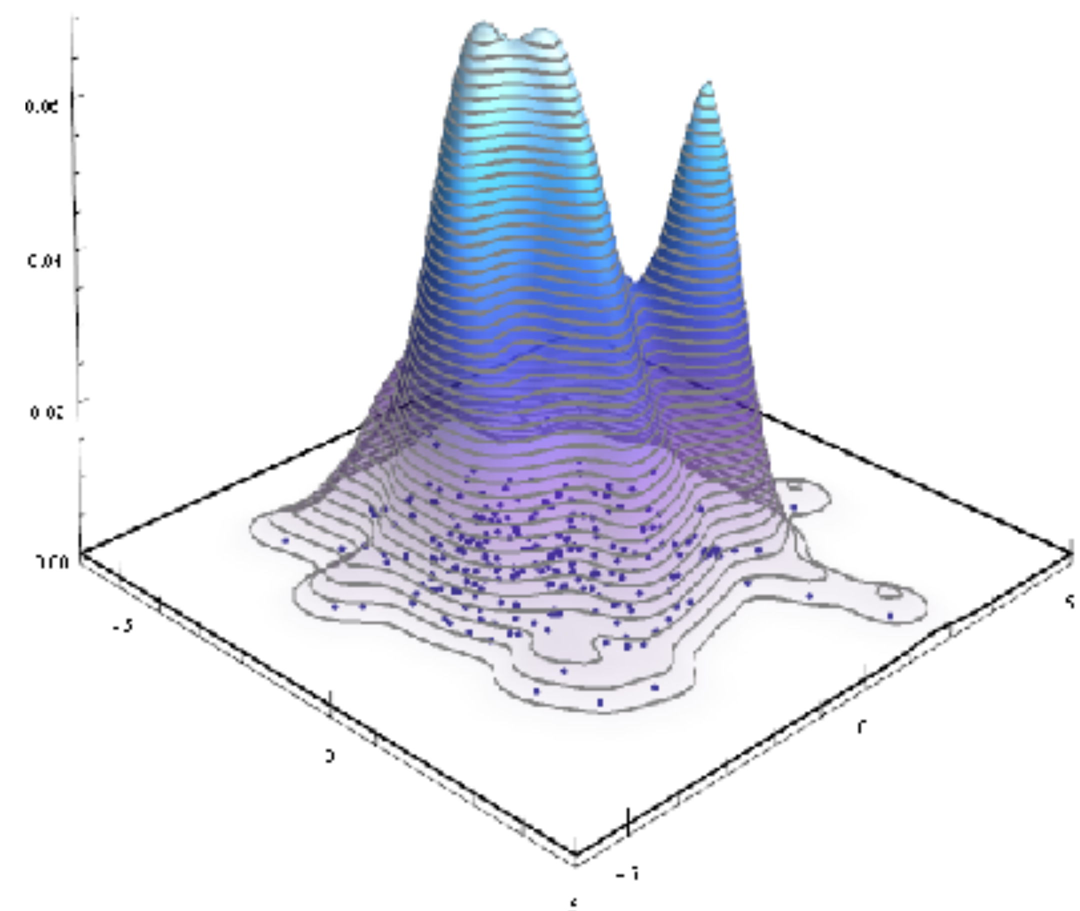


Factor Analysis / PCA

$$z \sim \mathcal{N}(z|\mu, \Sigma)$$

$$y \sim \mathcal{N}(y|Wz, \sigma_y^2 I)$$

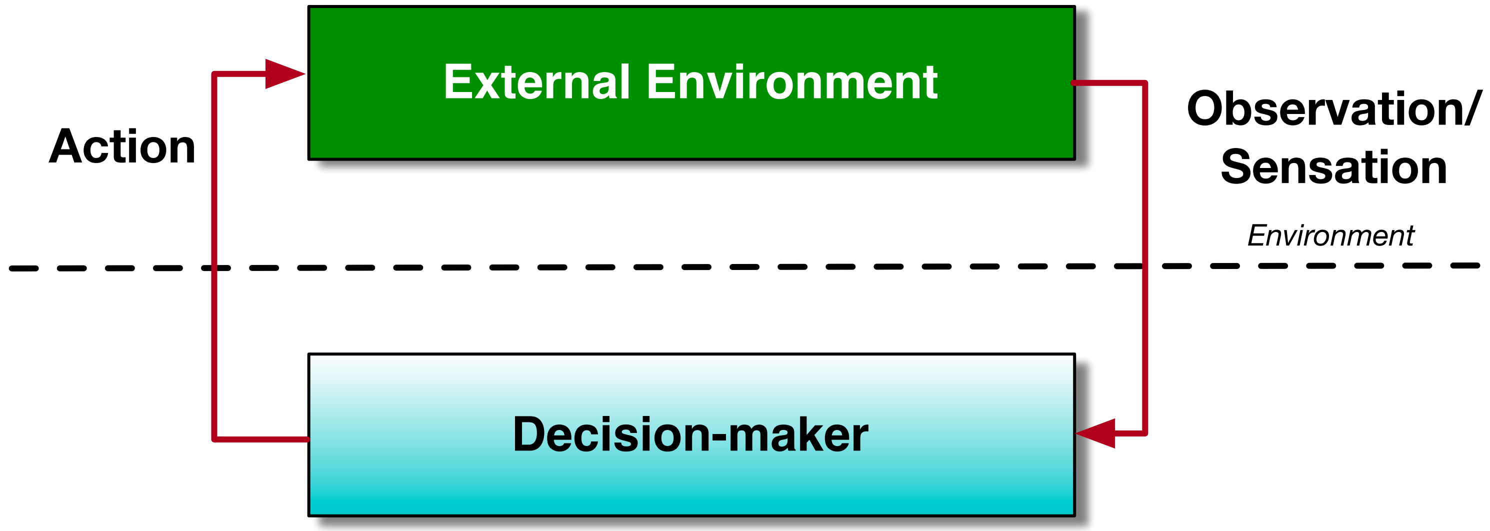
- Can learn distributions of some things and point estimates of others.
- Deep Generative Models and Unsupervised learning - more in Part III



Decision-making

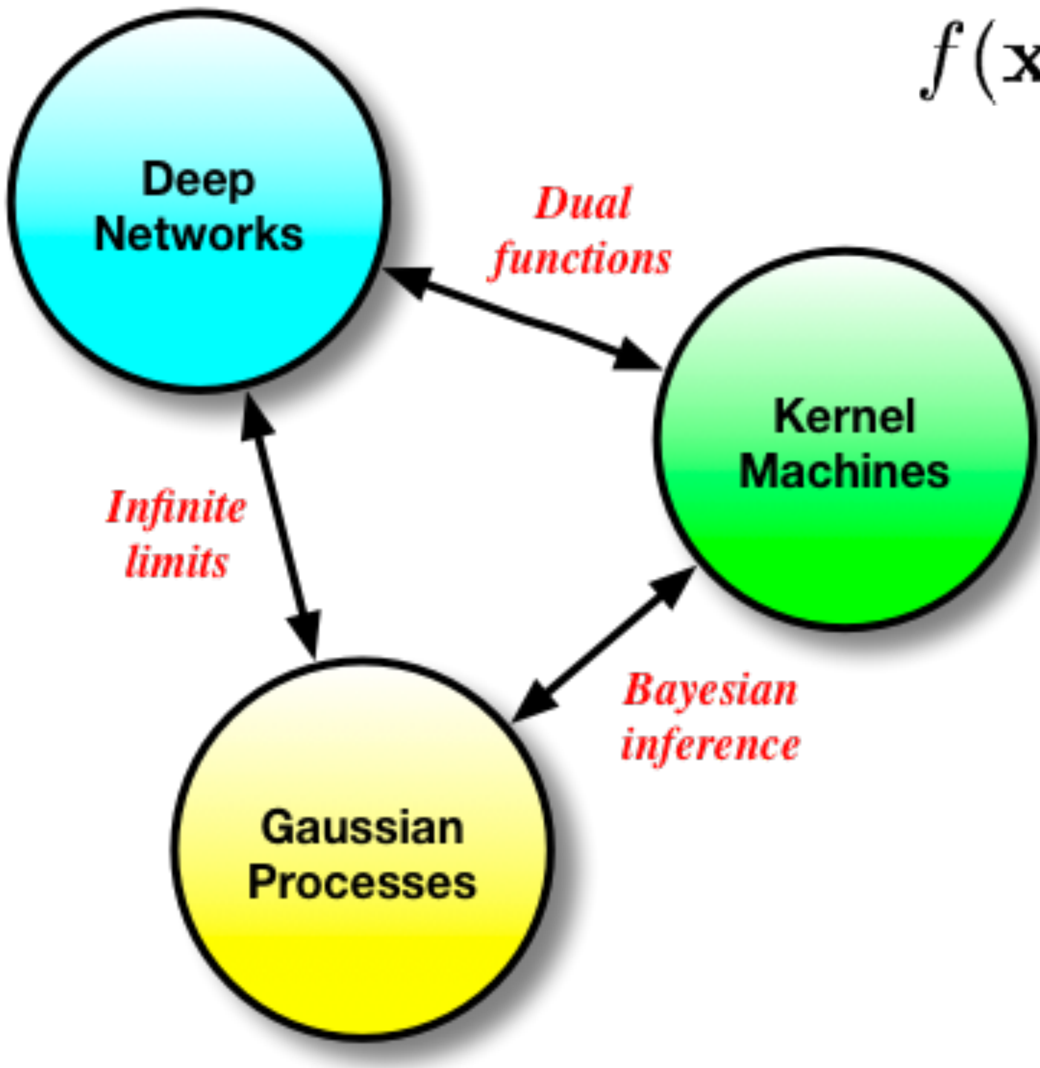
Probabilistic models of environments and actions

Prior over actions	$a \sim p(a)$
Interaction only	$u(s, a) \sim \mathrm{Environment}(a)$
Reward/Utility	$p(R(s)\|a) \propto \exp(u(s, a))$



Setup is common in experimental design, causal learning, reinforcement learning.

Probabilistic Dualities



Basis Function Regression

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}; \boldsymbol{\theta}); \qquad \{\mathbf{w}, \boldsymbol{\theta}\} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

$$y = f(\mathbf{x}) + \epsilon; \qquad \epsilon \sim \mathcal{N}(0, \sigma_y^2)$$

Move from primal variables to dual variables

$$\mathcal{L}(f) = \frac{1}{2} \sum_n (y_n - f(\mathbf{x}))^2 + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}}$$

Kernel trick and methods

Probability distributions over functions

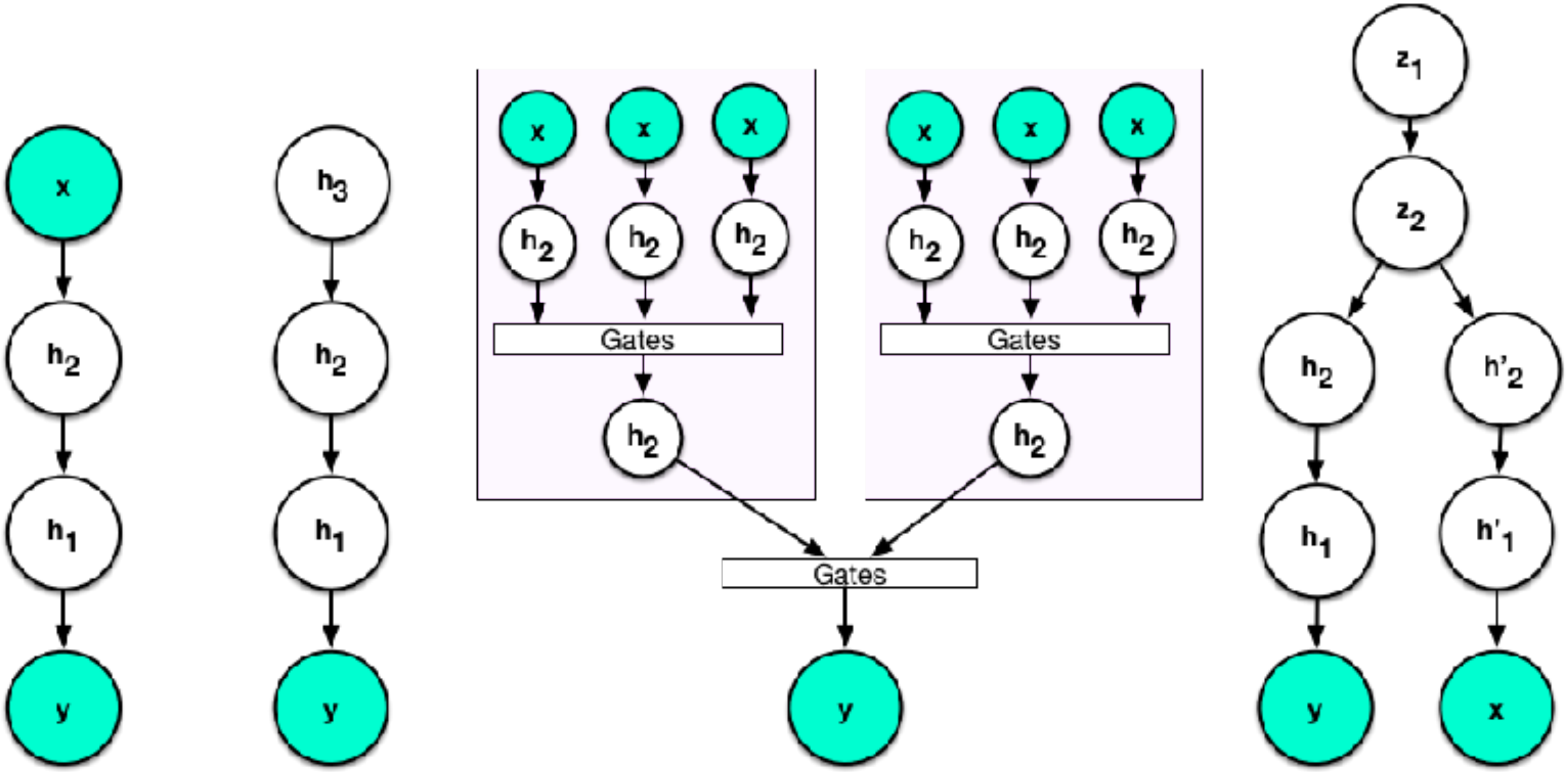
$$p(f) = \mathcal{N}(0, \mathbf{K}) \quad p(y|f) = \mathcal{N}(f, \sigma^2)$$

Gaussian processes

Deep and Hierarchical

Hierarchical Model: models where the (prior) probability distributions can be decomposed into a sequence of conditional distributions

$$p(z) = p(z_1|z_2)p(z_2|z_3)\dots p(z_{L-1}|z_L)p(z_L)$$



Deep feed-forward regression

Deep directed generative model

Hierarchical Mixture of Experts

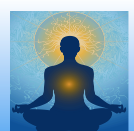
Deep Multi-view model/ Information Bottleneck

Foundations

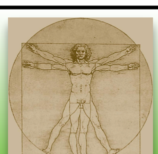
How will you approach your ML research and practice?

In general:
Human-centred,
interdisciplinary approach


Sociological

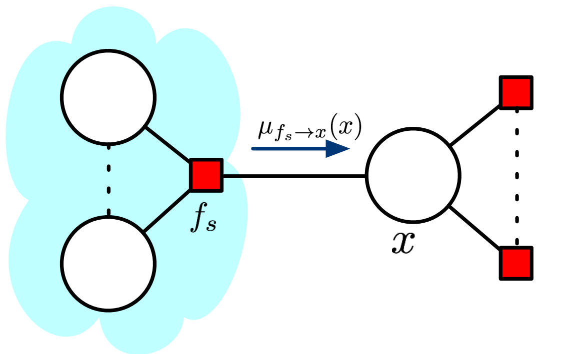

Psychological


Componential


Physiological

Sun's Phenomenological Levels

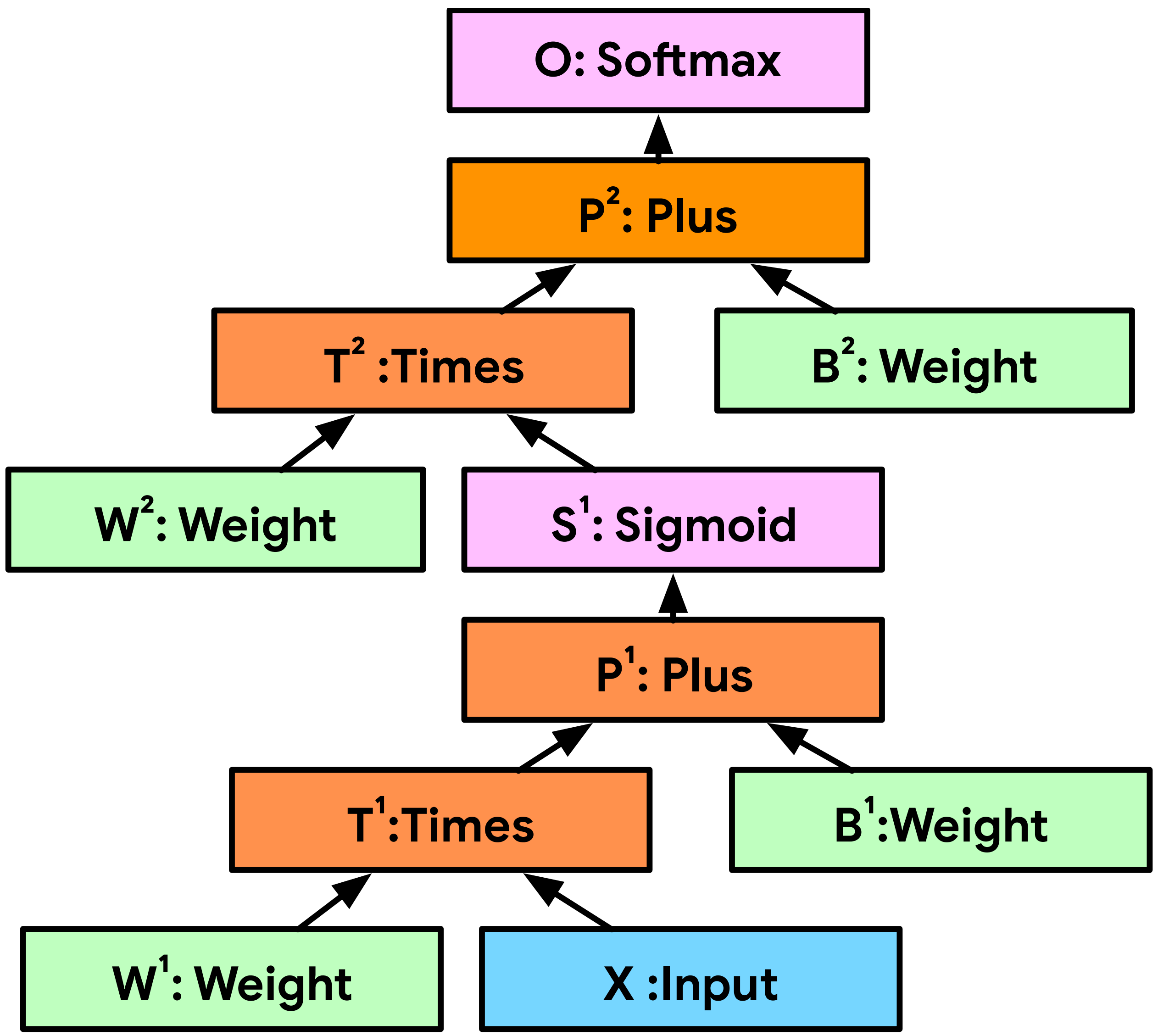
For the ML Core:
Probabilistic and pragmatic in approach

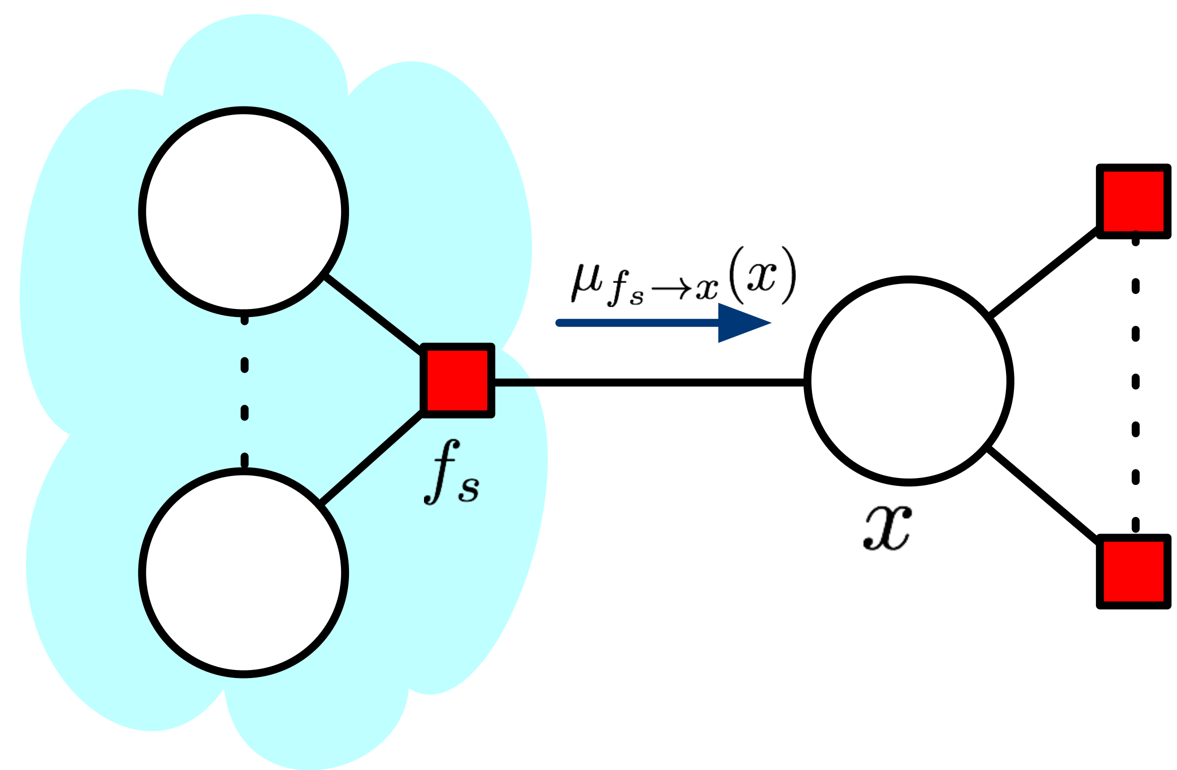


Architecture-Loss

Model-Inference-Algorithm

Architecture-Loss

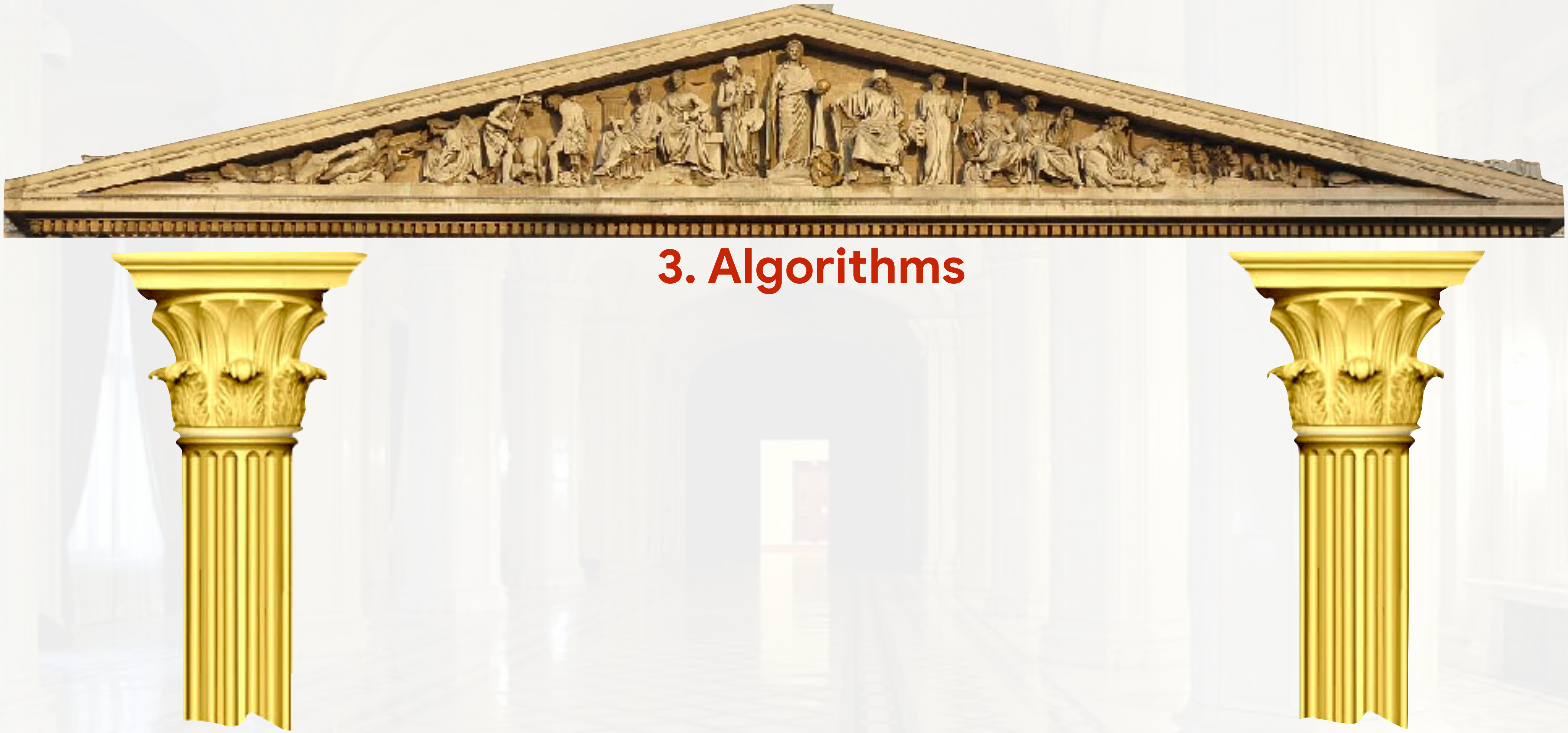




1. Computational Graphs

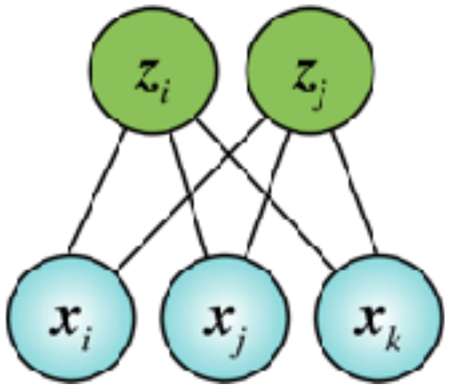
2. Error propagation

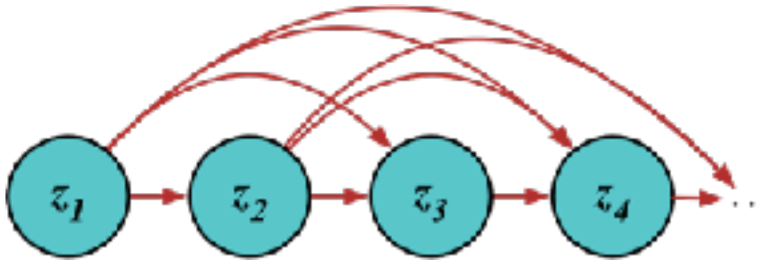
Model-Inference-Algorithm



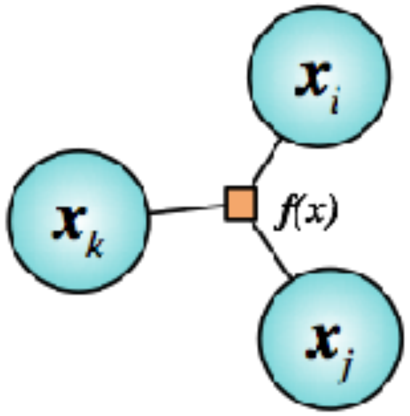
Models

Directed and Undirected

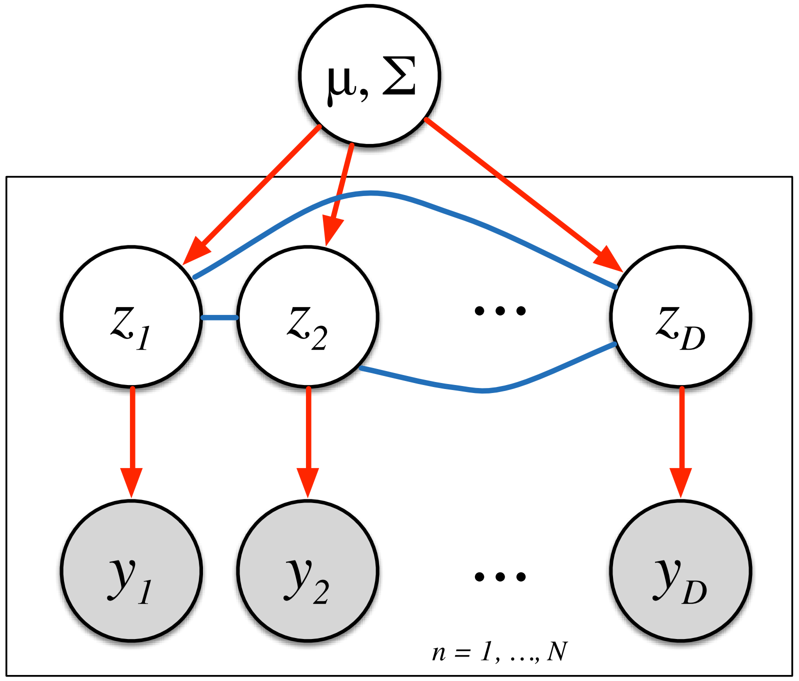




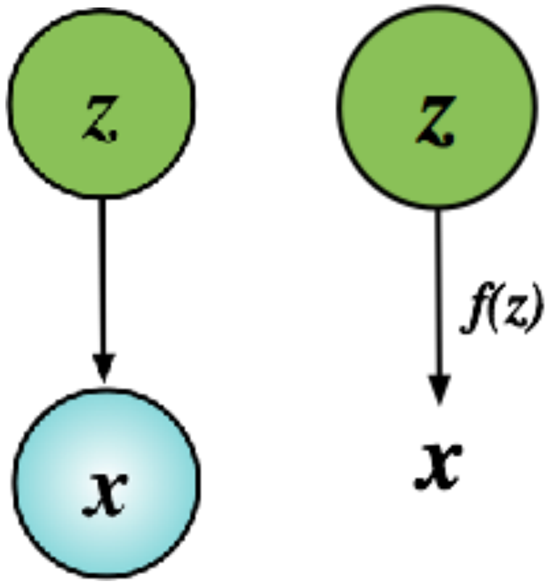
Fully-observed



Parametric, Non-parametric And semi-parametric



Latent Variable



Learning Principles

Statistical Inference

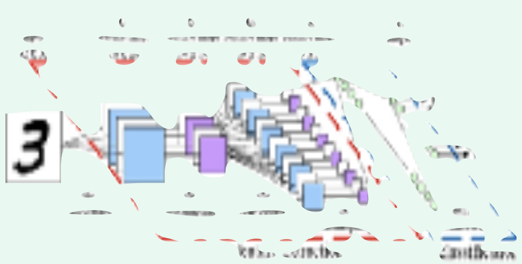
- **Direct**
  - Laplace approximation
  - Maximum Likelihood
  - Maximum a posteriori
  - Variational Inference
  - Cavity Methods
  - Integr. Nested Laplace Approx
  - Expectation Maximisation
  - Markov chain Monte Carlo
  - Noise Contrastive
  - Sequential Monte Carlo
- **Indirect**
  - Two Sample Comparison
  - Method of Moments
  - Approx Bayesian Computation
  - Transportation methods
  - Max Mean Discrepency

Learning Principles

Algorithms

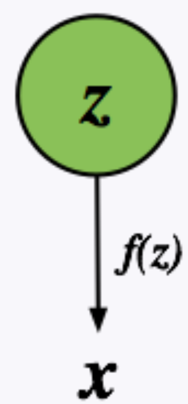
A given model and learning principle can be implemented in many ways.

Convolutional neural network + penalised maximum likelihood



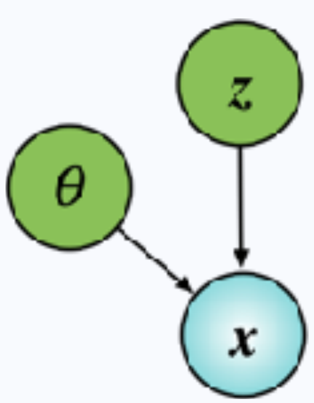
- Optimisation methods (SGD, Adagrad)
- Regularisation (L1, L2, batchnorm, dropout)

Implicit Generative Model + Two-sample testing



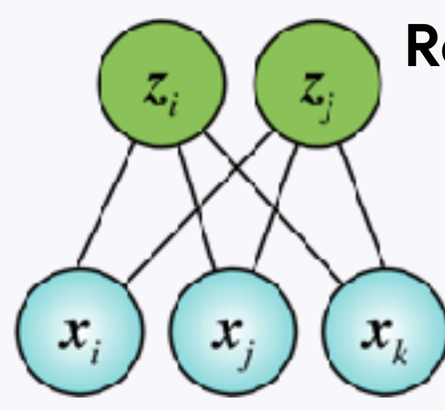
- Unsupervised-as-supervised learning
- Approximate Bayesian Computation (ABC)
- Generative adversarial network (GAN)

Latent variable model + variational inference



- VEM algorithm
- Expectation propagation
- Approximate message passing
- Variational auto-encoders (VAE)

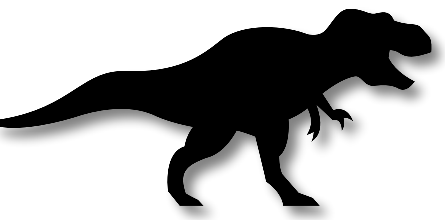
Restricted Boltzmann Machine + maximum likelihood



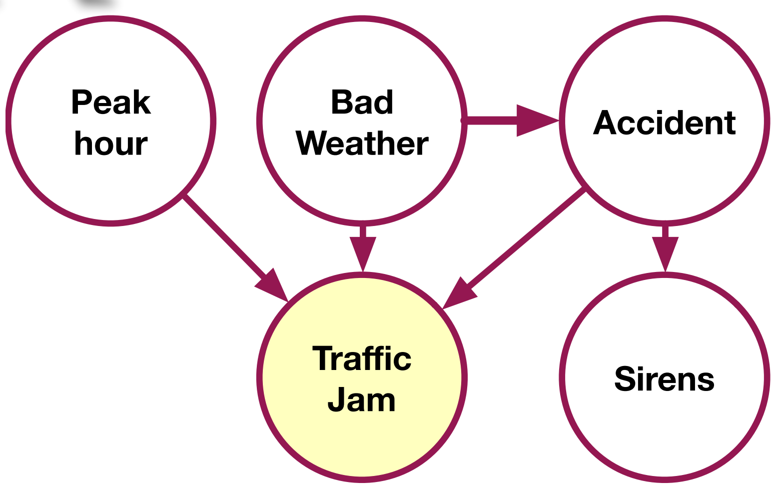
- Contrastive Divergence
- Persistent CD
- Parallel Tempering
- Natural gradients

Final Words

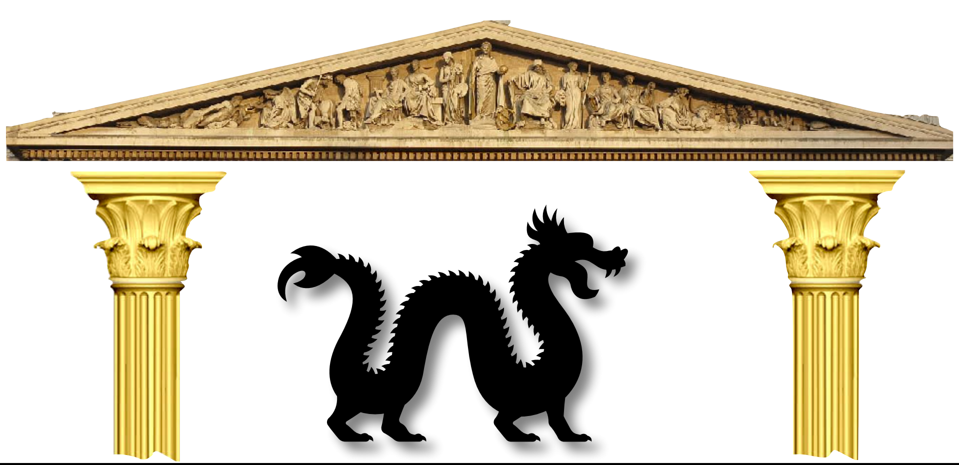
Subjective Probability
Probability as a degree of belief



Probabilistic descriptions of systems and data

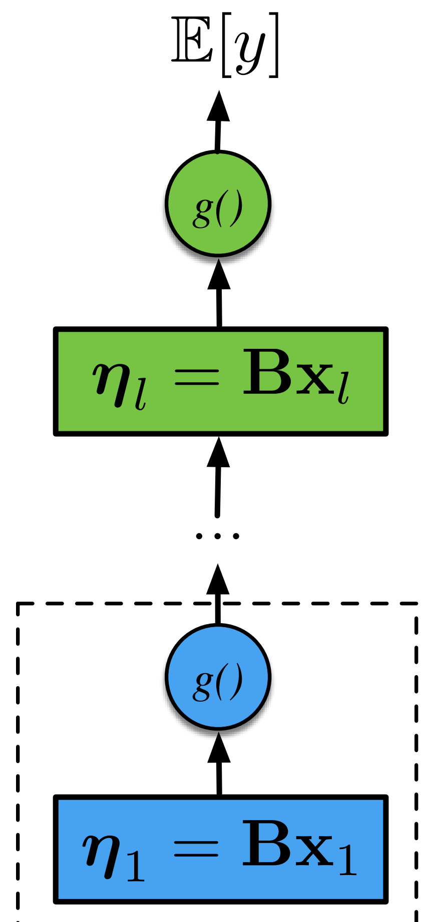


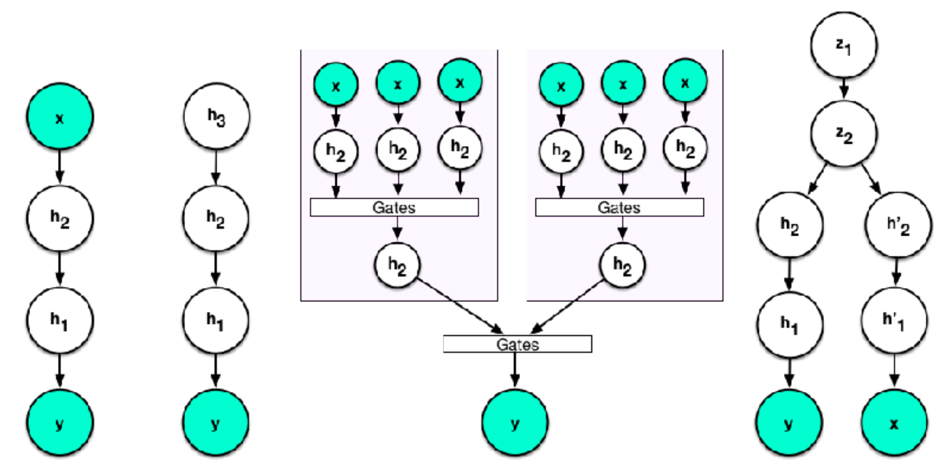
Model-Inference-Algorithm

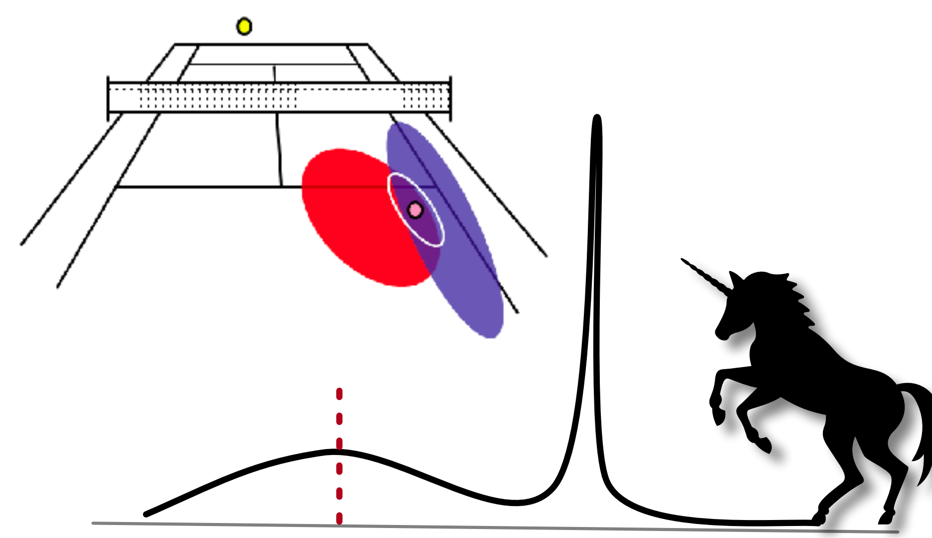


Deep Learning, Estimation theory, hierarchical models, dualities

Probabilistic Model | **Likelihood function**







Planting the Seeds of Probabilistic Thinking

Foundations | Tricks | Algorithms

Shakir Mohamed

Research Scientist, DeepMind

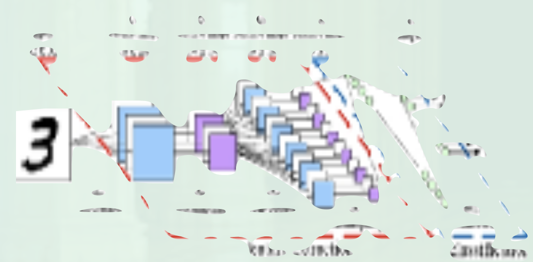
@shakir_za

DeepMind

Last Time …

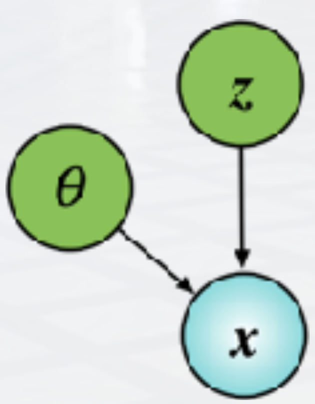
3. Algorithms

Convolutional neural network
+ penalised maximum likelihood



- Optimisation methods (SGD, Adagrad)
- Regularisation (L1, L2, batchnorm, dropout)

Latent variable model
+ variational inference



- VEM algorithm
- Expectation propagation
- Approximate message passing
- Variational auto-encoders (VAE)

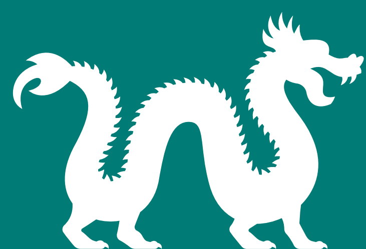
1. Models

2. Learning Principles

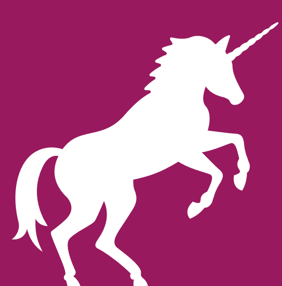
Planting the Seeds of Probabilistic Thinking

Part II: Tricks

Learning Objectives



1. Develop tools to manipulate distributions by studying 6 probability questions.



2. Build connections between concepts in machine learning and those in other computational sciences.

Inferential Questions

Probabilistic dexterity is needed to solve the fundamental problems of machine learning and artificial intelligence.

Evidence Estimation

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) dz$$

Moment Computation

$$\mathbb{E}[f(\mathbf{z})|\mathbf{x}] = \int f(\mathbf{z}) p(\mathbf{z}|\mathbf{x}) dz$$

Parameter Estimation

$$p(\boldsymbol{\theta}|\mathbf{x}_{0:N})$$

Prediction

$$p(\mathbf{x}_{t+1}|\mathbf{x}_{0:t})$$

Planning

$$\mathcal{J} = \mathbb{E}_p\left[\int_0^\infty C(\mathbf{x}_t) dt | \mathbf{x}_0, \mathbf{u}\right]$$

Hypothesis Testing

$$\mathcal{B} = \log p(\mathbf{x}|H_1) - \log p(\mathbf{x}|H_2)$$

Experimental Design

$$\mathcal{IG} = D[p(\mathbf{x}_{t:T}|u) \| p(\mathbf{x}_{0:t})]$$

Identity Trick

Transform an expectation w.r.t. distribution *p*,
into an expectation w.r.t. distribution *q*.

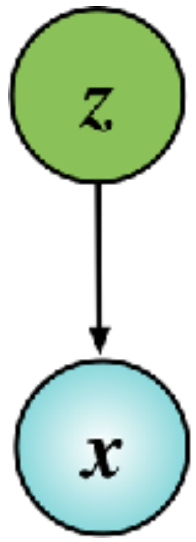
$$\int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$$

$$\updownarrow$$

$$\mathbb{E}_{q(\mathbf{x})}[g(\mathbf{x}; f)] = \int q(\mathbf{x}) g(\mathbf{x}, f) d\mathbf{x}$$

Do this by introducing a **probabilistic one** $\frac{p(\mathbf{x})}{p(\mathbf{x})}$

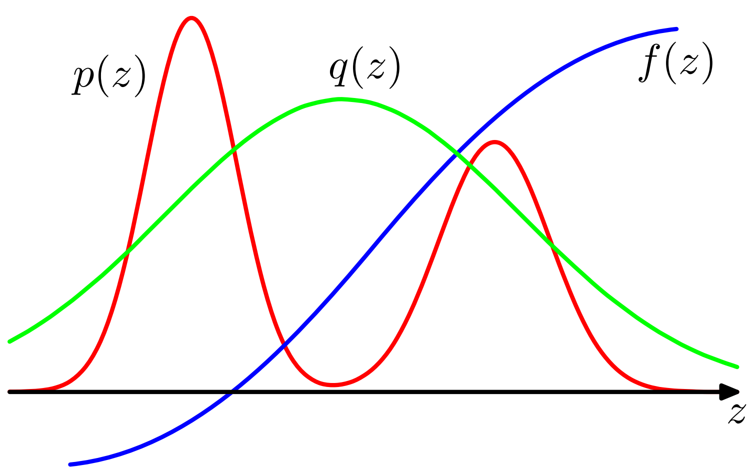
Identity Trick



Integral problem $$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

Probabilistic one $$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\frac{q(\mathbf{z})}{q(\mathbf{z})}d\mathbf{z}$$

Re-group/re-weight $$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})d\mathbf{z}$$



Conditions

- *q(z)*>0, when p(x|*z)p(z)* $\neq 0$.
- *q(z) is known/easy to handle.*

$$p(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z})}\left[p(\mathbf{x}|\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}\right]$$

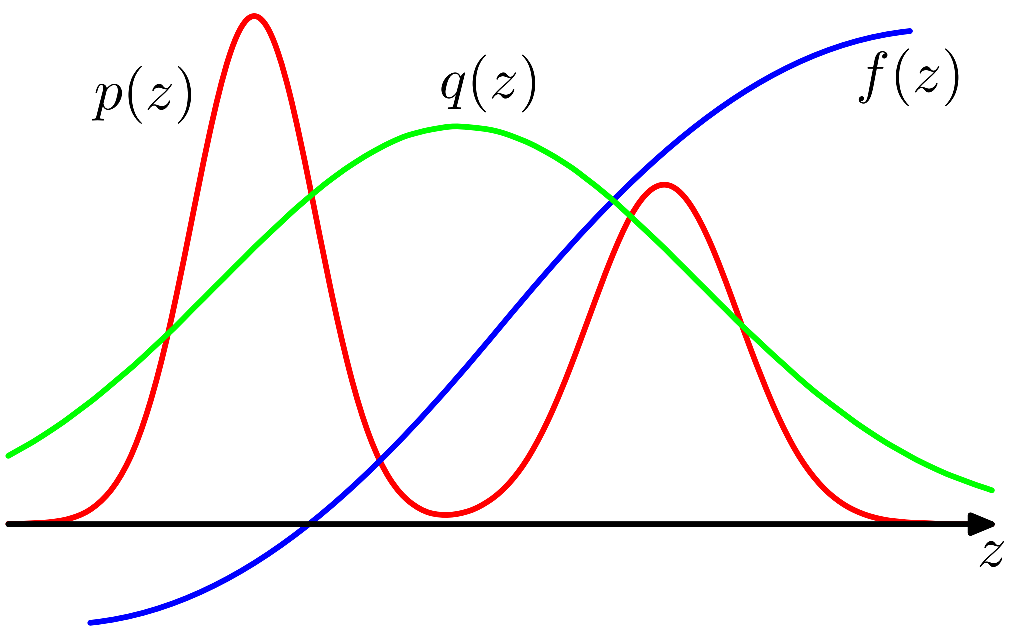
Importance Sampling

$$p(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z})}\left[p(\mathbf{x}|\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}\right]$$

Monte Carlo Estimator

$$p(\mathbf{x}) = \frac{1}{S}\sum_{s} w^{(s)} p(\mathbf{x}|\mathbf{z}^{(s)})$$

$$w^{(s)} = \frac{p(z)}{q(z)} \quad z^{(s)} \sim q(z)$$



Identity Trick Elsewhere

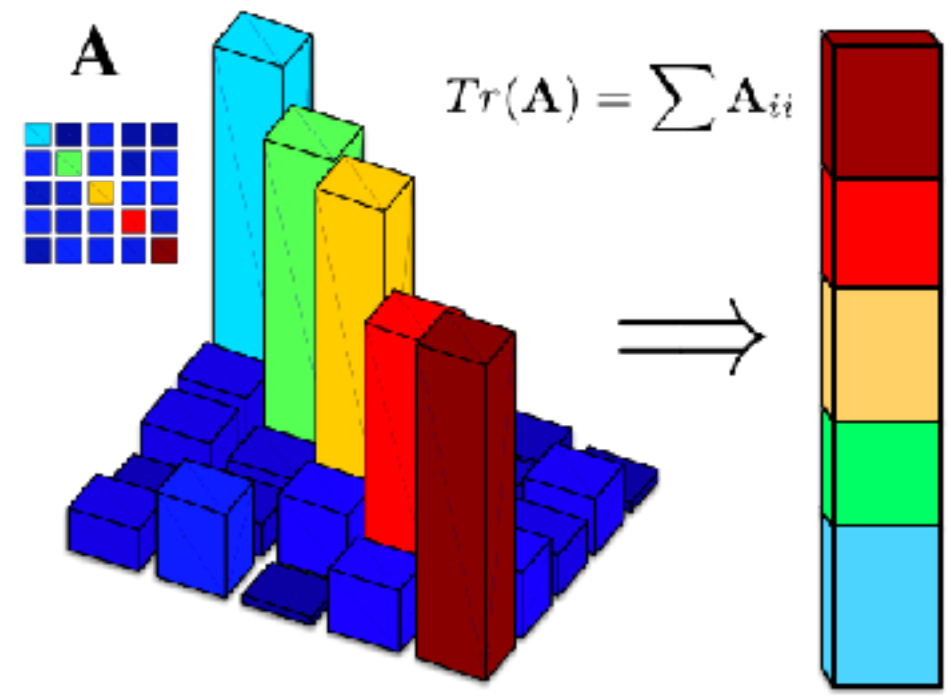
- Manipulate stochastic gradients
- Derive probability bounds
- RL for policy corrections

Hutchinson's Trick

Compute the trace of a matrix:

- KL between two Gaussians.
- Gradient of a log-determinant.

$$\partial(\log\det(\mathbf{X})) = Tr(\mathbf{X}^{-1}\partial X)$$



Trace problem $Tr(\mathbf{A})$

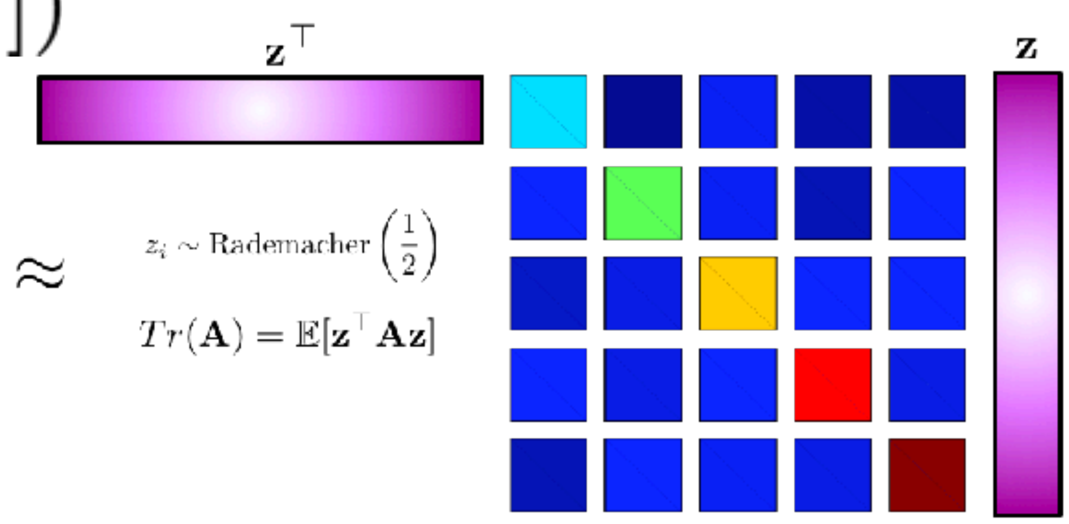
Zero mean unit var $\mathbb{E}[\mathbf{z}\mathbf{z}^\top] = \mathbf{I}$

$(\boldsymbol{\Sigma} + \mathbf{m}\mathbf{m}^\top)$

Identity Trick $Tr(\mathbf{A}\mathbf{I}) = Tr(\mathbf{A}\mathbb{E}[\mathbf{z}\mathbf{z}^\top])$

Linear operations $\mathbb{E}[Tr(\mathbf{A}\mathbf{z}\mathbf{z}^\top)]$

Trace property $\mathbb{E}[\mathbf{z}^\top\mathbf{A}\mathbf{z}]$



Sampling z randomly, compute Trace using linear systems of equations

Probability Flow Tricks

Distribution and sample $$\hat{x} \sim p(x)$$

Transformation $$\hat{y} = g(\hat{x}; \theta)$$

Unconscious Statistician $$\mathbb{E}_{p(x)}[g(x;\theta)]$$

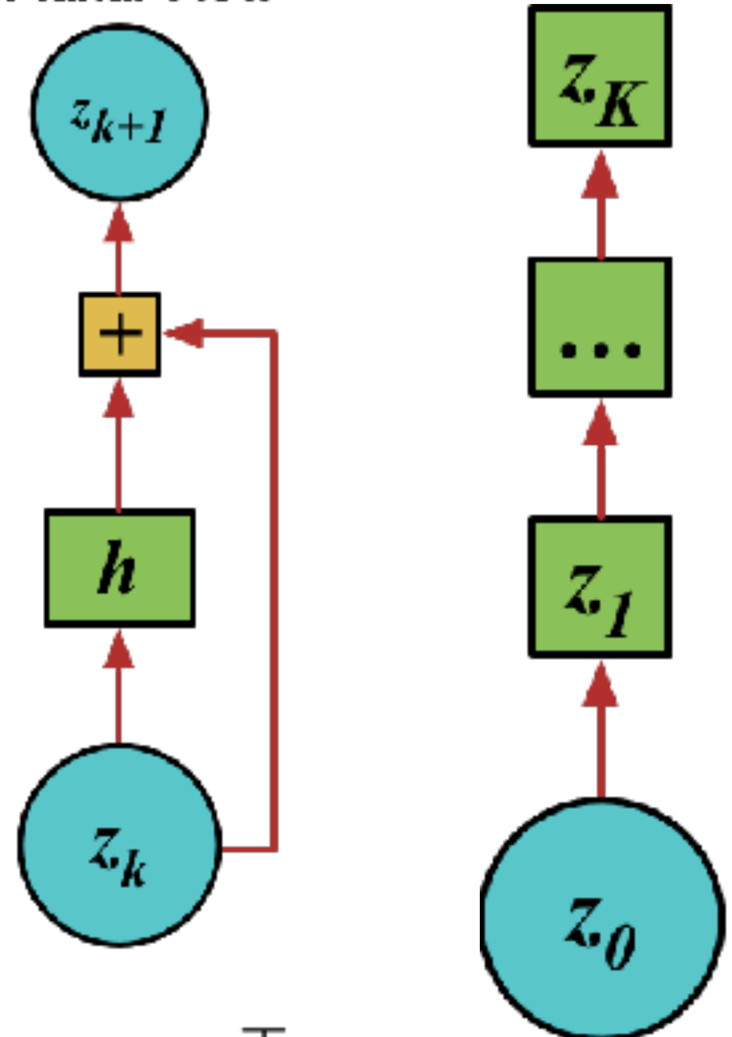
Change of Variables $$p(\mathbf{y}) = p(\mathbf{x}) \left| \frac{dg}{d\mathbf{x}} \right|^{-1}$$

Makes entropy computation and backpropagation easy.

Begin with a diagonal Gaussian and improve by change of variables.

Triangular Jacobians allow for computational efficiency.

Planar Flow



Compute

$$\log\det\left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}}\right)$$

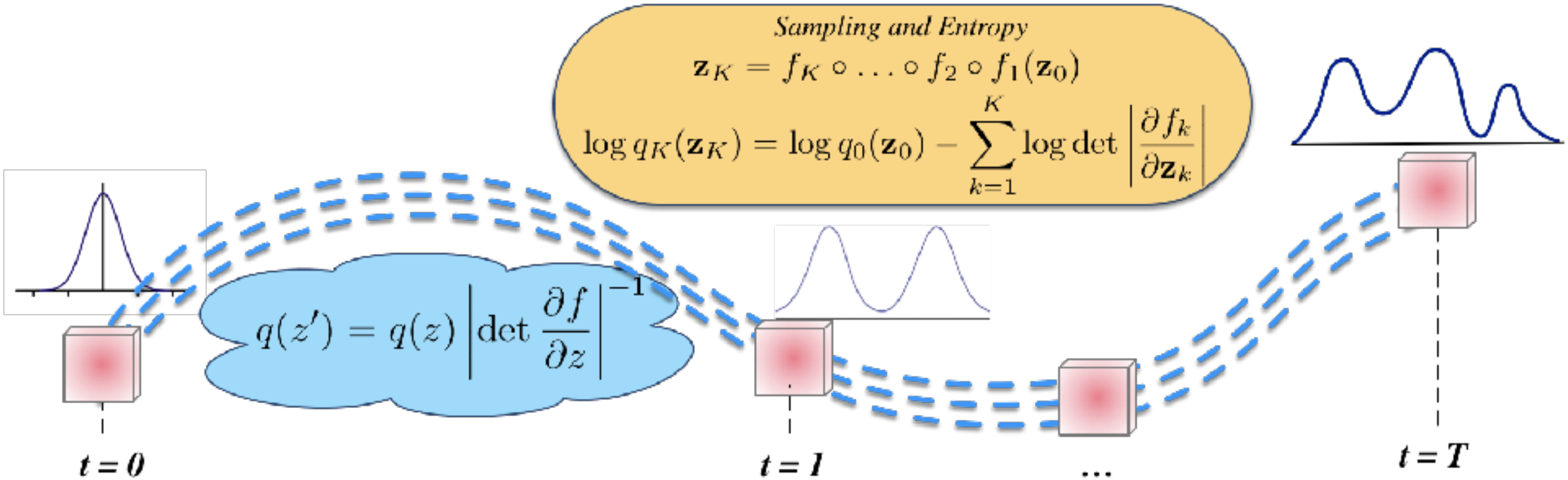
$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^\top \mathbf{z} + b)$$

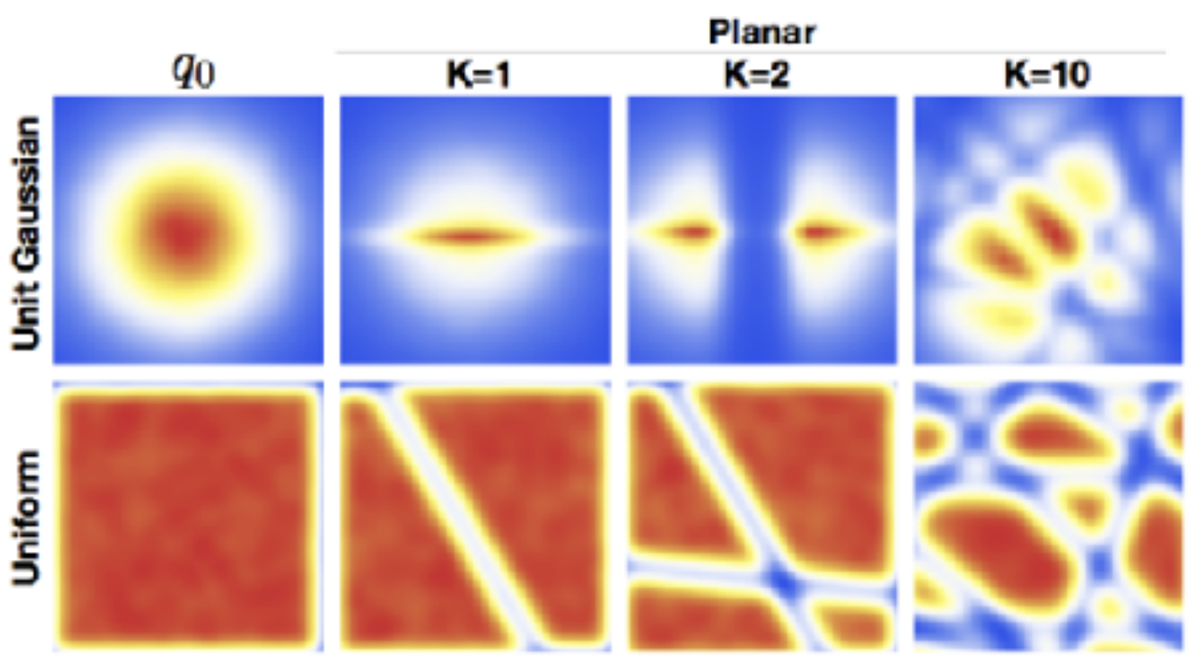
$$\det(I + ab^\top) = 1 + a^\top b$$

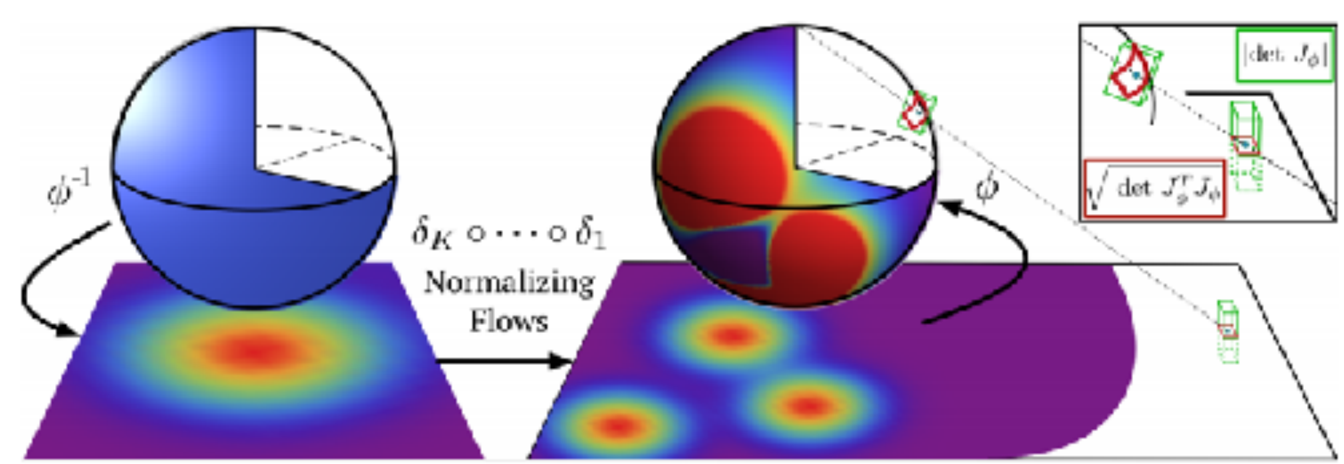
$$\det(\mathbf{I} + \mathbf{u}\mathbf{s}^\top) = (1 + \mathbf{u}^\top \mathbf{s}) \qquad \mathbf{s} = h'\mathbf{w}$$

Linear time computation of the determinant and its gradient.

Normalising Flows







Stochastic Optimisation

Common gradient problem

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int \boxed{q_\phi(\mathbf{z})} \boxed{f_\theta(\mathbf{z})} d\mathbf{z}$$

- Don't know this expectation in general.
- Gradient is of the parameters of the distribution w.r.t. which the expectation is taken.

1. **Pathwise estimator**: Differentiate the function **f(z)**
2. **Score-function estimator**: Differentiate the density **q(z|x)**

Typical problem areas

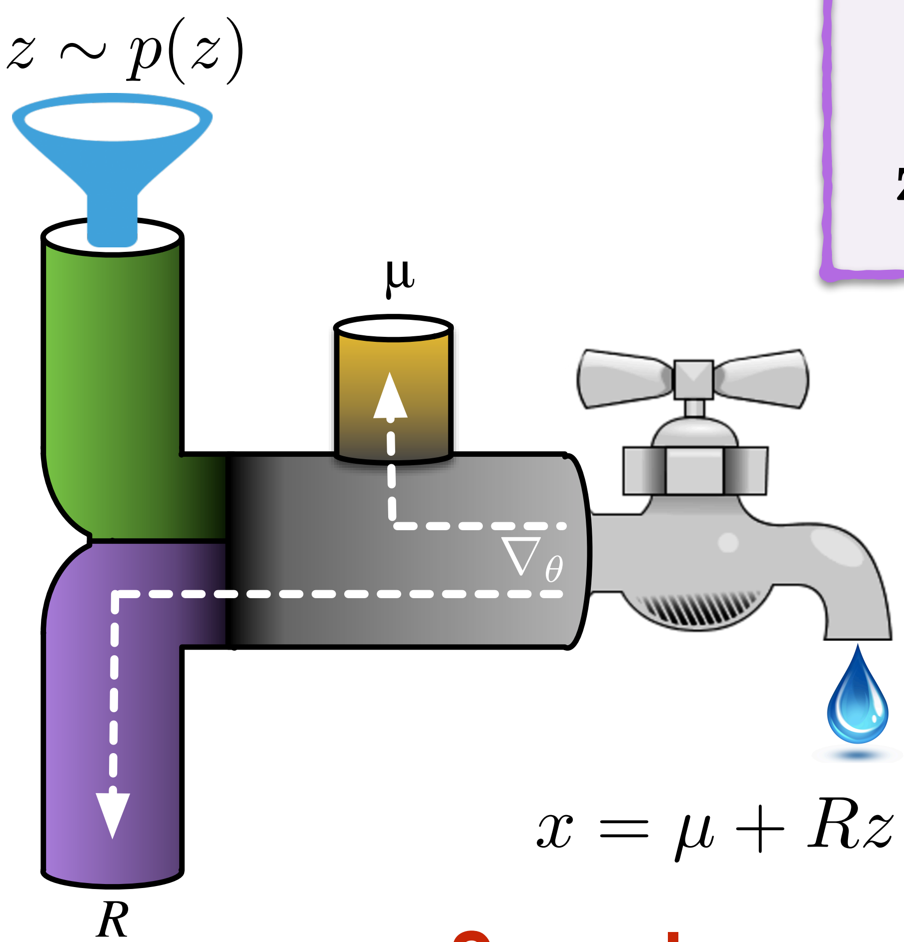
- Sensitivity analysis
- Generative models and inference
- Reinforcement learning and control
- Operations research and inventory control
- Monte Carlo simulation
- Finance and asset pricing

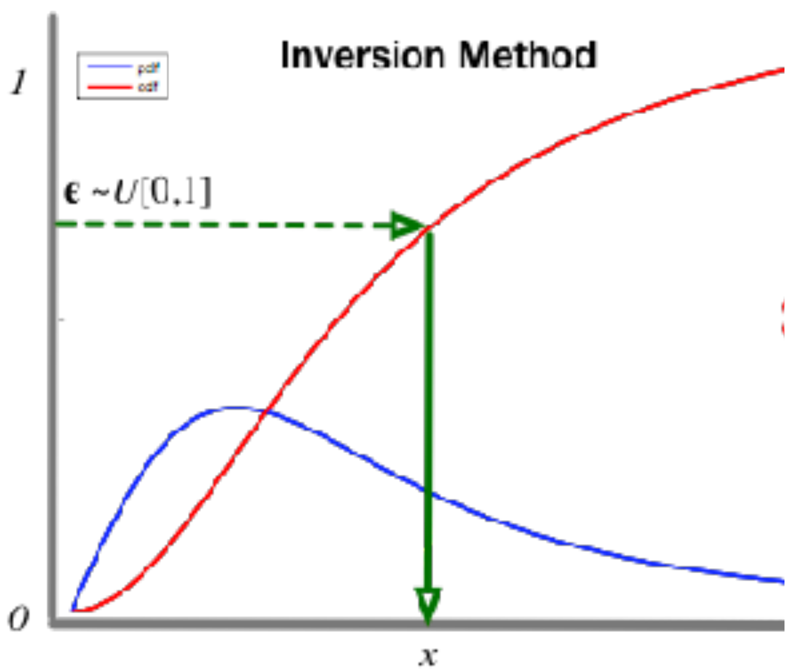
Reparameterisation Tricks

Distributions can be expressed as a transformations of other distributions.

$$z \sim q_\phi(\mathbf{z})$$

$$\mathbf{z} = g(\epsilon, \phi) \quad \epsilon \sim p(\epsilon)$$





Samplers, one-liners and change-of-variables

$$p(z) = \left|\frac{d\epsilon}{dz}\right| p(\epsilon) \implies |p(z)dz| = |p(\epsilon)d\epsilon|$$

Pathwise Estimator

(Non-rigorous) Derivation

$$\nabla_\phi \mathbb{E}_{q(z)}[f(\mathbf{z})] = \nabla_\phi \int q_\phi(\mathbf{z}) \boxed{f(\mathbf{z})} d\mathbf{z}$$

Known transformation

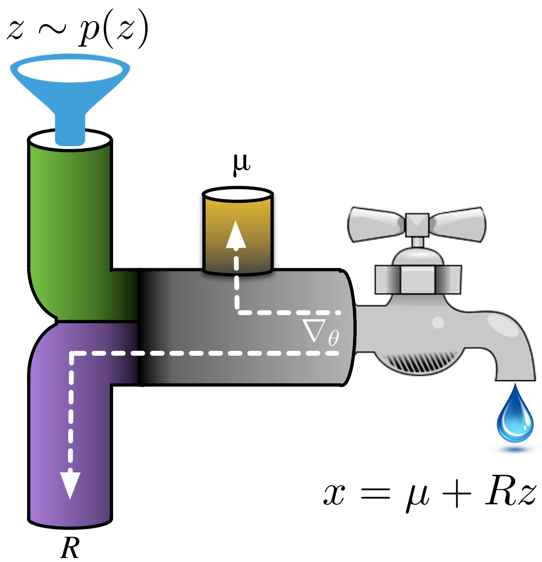
$z = g(\epsilon, \phi); \quad \epsilon \sim p(\epsilon)$

$$= \nabla_\phi \int p(\boldsymbol{\epsilon}) \frac{d\boldsymbol{\epsilon}}{d\mathbf{z}} f(g(\epsilon, \phi)) g'(\epsilon, \phi) d\boldsymbol{\epsilon}$$

Change of variables

$$= \nabla_\phi \mathbb{E}_{p(\epsilon)}[f(g(\phi, \epsilon)] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(g(\phi, \epsilon)]$$

Inv fn Thm



$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f_\theta(g(\epsilon, \phi))]$$

Other names

- Unconscious statistician
- Stochastic backpropagation
- Perturbation analysis
- Reparameterisation trick
- Affine-independent inference

When to use

- Function *f* is differentiable
- Density *q* is known with a suitable transform of a simpler base distribution: inverse CDF, location-scale transform, or other co-ordinate transform.
- Easy to sample from base distribution.

Log-derivative Trick

Score function is the derivative of a log-likelihood function.

$$\nabla_\phi \log q_\phi(\mathbf{z}) = \frac{\nabla_\phi q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})}$$

Several useful properties

Expected score $\mathbb{E}_{q(z)}\left[\nabla_\phi \log q_\phi(\mathbf{z})\right] = 0$ ↞Show this

$$\mathbb{E}_{q(z)}[\nabla_\phi \log q_\phi(\mathbf{z})] = \int q(z) \frac{\nabla_\phi q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})} = \nabla \int q_\phi(\mathbf{z}) = \nabla 1 = 0$$

Fisher Information $\mathbb{V}[\nabla_\theta \log p(\mathbf{x};\theta)] = \mathcal{I}(\theta) = \mathbb{E}_{p(x;\theta)}[\nabla_\theta \log p(\mathbf{x};\theta) \nabla_\theta \log p(\mathbf{x};\theta)^\top]$

Score-function Estimator

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int \boxed{q_\phi(\mathbf{z})} f_\theta(\mathbf{z}) d\mathbf{z}$$

Leibnitz integral rule

$$= \int \frac{q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})} \nabla_\phi q_\phi(\mathbf{z}) f(\mathbf{z}) d\mathbf{z}$$

Identity

$$= \int q_\phi(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z}) f(\mathbf{z}) d\mathbf{z}$$

Log-deriv

$$= \mathbb{E}_{q_\phi(\mathbf{z})} [f(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})]$$

Gradient

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \mathbb{E}_{q_\phi(\mathbf{z})} [(f(\mathbf{z}) - c) \nabla_\phi \log q_\phi(\mathbf{z})]$$

Control Variate

Other names

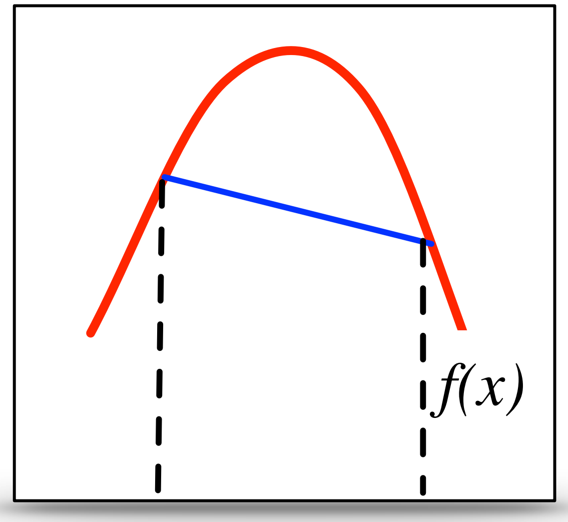
- Likelihood ratio method
- REINFORCE and policy gradients
- Automated & Black-box inference

When to use

- Function is not differentiable, not analytical.
- Distribution *q* is easy to sample from.
- Density *q* is known and differentiable.

Bounding Tricks

An important result from convex analysis lets us move expectations through a function:



For concave functions $f(.)$

$$f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$$

Logarithms are strictly *concave* allowing us to use Jensen's inequality.

$$\log \int p(x) g(x) dx \geq \int p(x) \log g(x) dx$$

Bounding Trick Elsewhere

Optimisation; Variational Inference; Rao-Blackwell Theorem;

Other Bounding Tricks

- Fenchel duality
- Holder's inequality
- Monge-Kantorovich Inequality

Evidence Bounds

Integral problem

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

Proposal

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\frac{q(\mathbf{z})}{q(\mathbf{z})}d\mathbf{z}$$

Importance Weight

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})d\mathbf{z}$$

Jensen's inequality

$$\log \int p(x)g(x)dx \geq \int p(x) \log g(x)dx$$

$$\log p(\mathbf{x}) \geq \int q(\mathbf{z}) \log \left(p(\mathbf{x}|\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}\right) d\mathbf{z}$$

$$= \int q(\mathbf{z}) \log p(\mathbf{x}|\mathbf{z}) - \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})}$$

Lower bound

$$\boxed{\mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})] - KL[q(\mathbf{z})\|p(\mathbf{z})]}$$

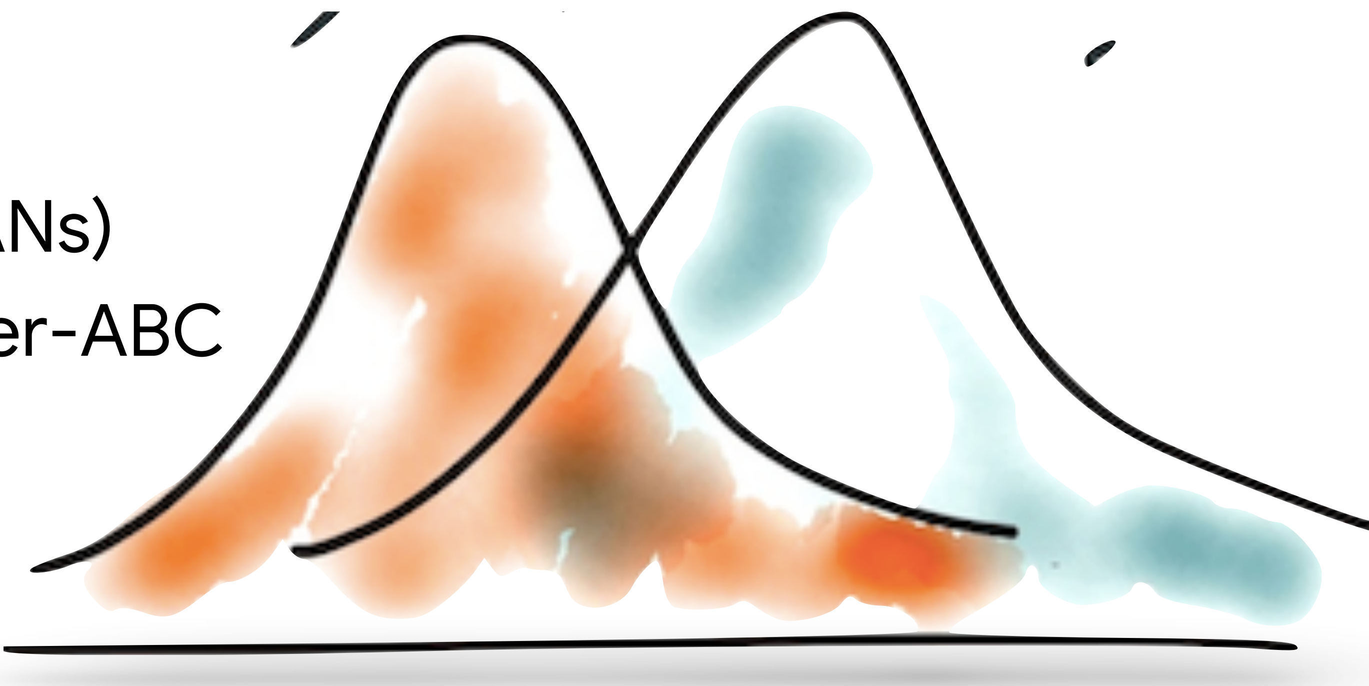
Density Ratio Trick

The ratio of two densities can be computed using a classifier of using samples drawn from the two distributions.

$$\frac{p^*(\mathbf{x})}{q(\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}$$

Density Ratio Trick Elsewhere

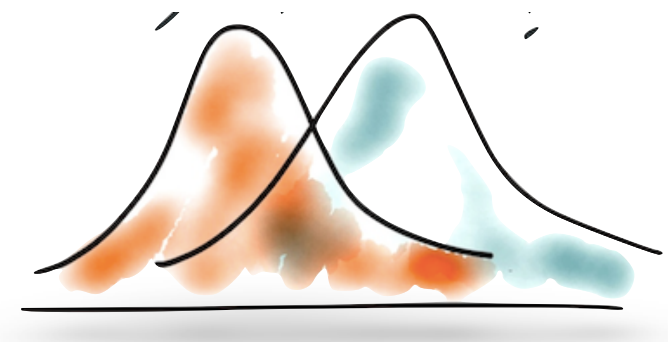
- Generative Adversarial Networks (GANs)
- Noise contrastive estimation, Classifier-ABC
- Two-sample testing
- Covariate-shift, calibration



Density Ratio Estimation

Assign labels $\{y_1, \ldots, y_N\} = \{+1, \ldots, +1, -1, \ldots, -1\}$

Equivalence $p^*(\mathbf{x}) = p(\mathbf{x}|y=1) \qquad q(\mathbf{x}) = p(\mathbf{x}|y=-1)$



Density Ratio $\dfrac{p^*(\mathbf{x})}{q(\mathbf{x})}$

Bayes' Rule $p(\mathbf{x}|y) = \dfrac{p(y|\mathbf{x})p(\mathbf{x})}{p(y)}$

Conditional

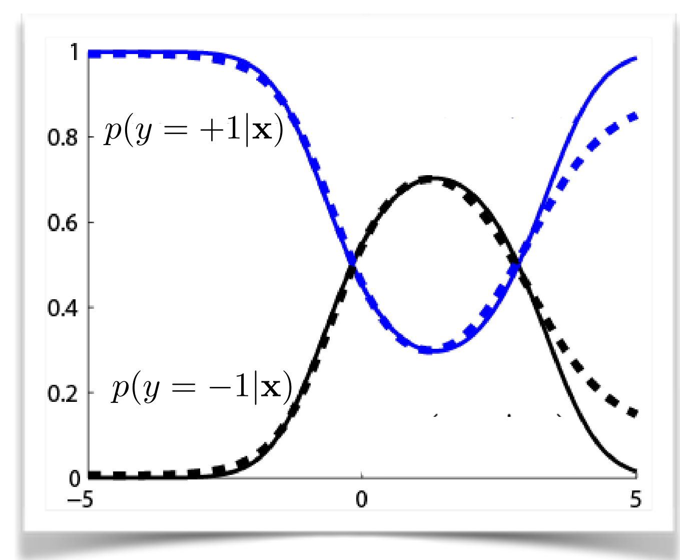
$$\frac{p^*(\mathbf{x})}{q(\mathbf{x})} = \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=-1)}$$

Bayes' Subst.

$$= \frac{p(y=+1|\mathbf{x})p(\mathbf{x})}{p(y=+1)} \Bigg/ \frac{p(y=-1|\mathbf{x})p(\mathbf{x})}{p(y=-1)}$$

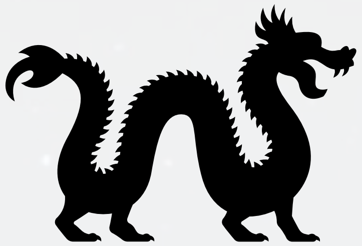
Class probability

$$\frac{p^*(\mathbf{x})}{q(\mathbf{x})} = \boxed{\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}}$$



Computing a density ratio is equivalent to class probability estimation.

Final Words



Strengthen your probabilistic dexterity.

Identity	$\frac{p(\mathbf{x})}{p(\mathbf{x})}$
Hutchinson's	$\mathbb{E}[\mathbf{z}\mathbf{z}^\top] = \mathbf{I}$
Flows	$p(\mathbf{y}) = p(\mathbf{x}) \left\lvert \frac{dg}{d\mathbf{x}} \right\rvert^{-1}$
Log-derivative	$\nabla_\phi \log q_\phi(\mathbf{z}) = \frac{\nabla_\phi q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})}$
Reparameterisation	$\mathbf{z} = g(\epsilon, \phi) \quad \epsilon \sim p(\epsilon)$
Density Ratio	$\frac{p^*(\mathbf{x})}{q(\mathbf{x})} = \frac{p(y=1\lvert\mathbf{x})}{p(y=-1\lvert\mathbf{x})}$

Planting the Seeds of Probabilistic Thinking

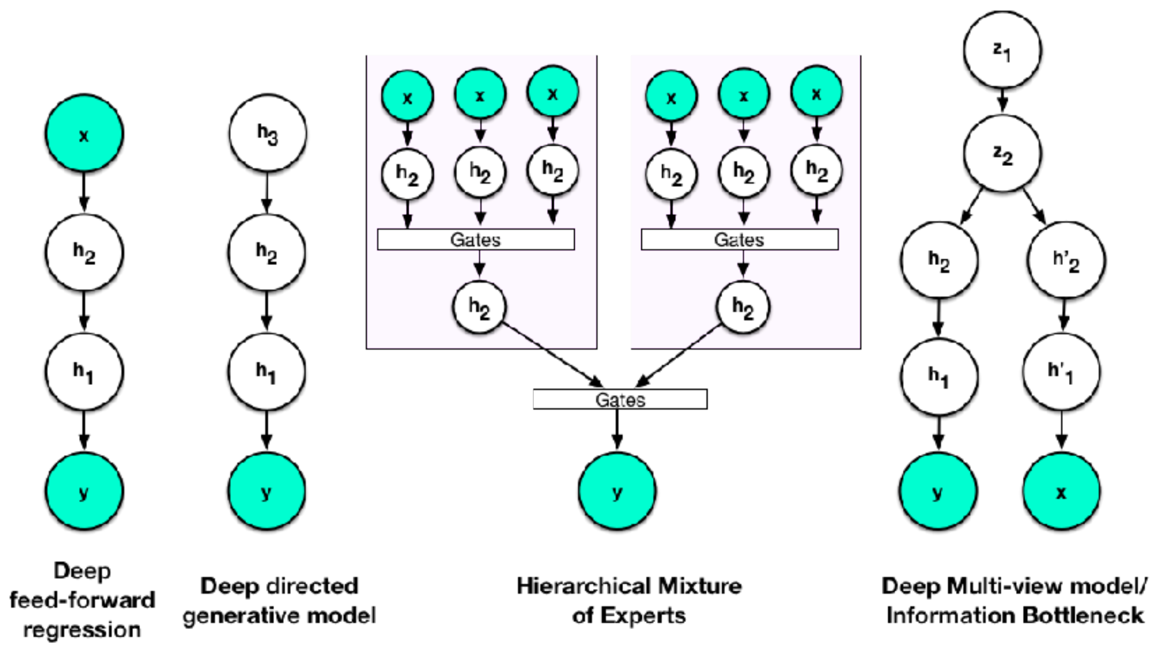
Foundations | Tricks | Algorithms

Shakir Mohamed
Research Scientist, DeepMind

@shakir_za

DeepMind

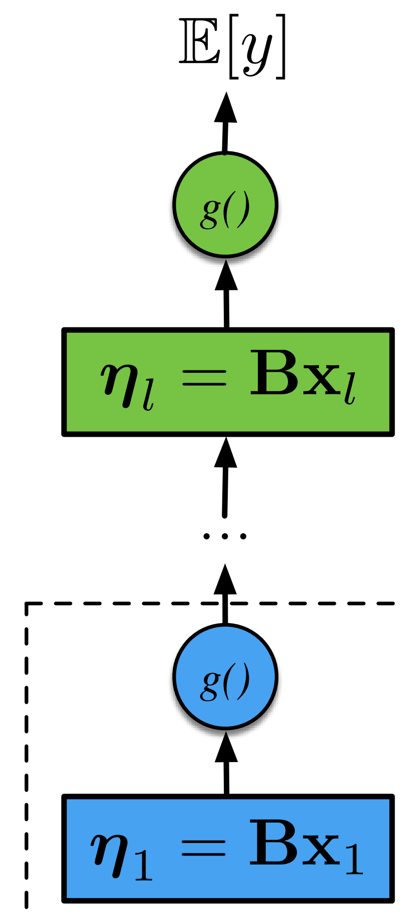
Last Time ...



Subjective Probability

Probability as a degree of belief

Model-Inference-Algorithm



Manipulating Integrals

Evidence Estimation

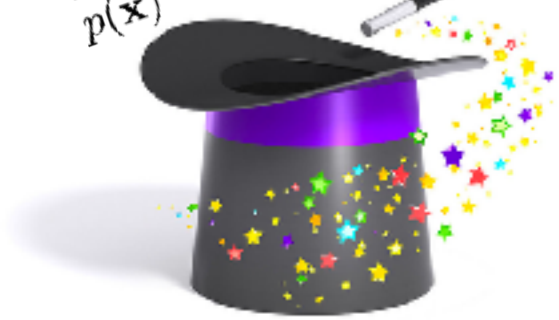
$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Moment Computation

$$\mathbb{E}[f(\mathbf{z})|\mathbf{x}] = \int f(\mathbf{z}) p(\mathbf{z}|\mathbf{x}) d\mathbf{z}$$

Parameter Estimation

$$p(\boldsymbol{\theta}|\mathbf{x}_{0:N})$$



Prediction

$$p(\mathbf{x}_{t+1}|\mathbf{x}_{0:t})$$

Planning

$$\mathcal{J} = \mathbb{E}_p\left[\int_0^\infty C(\mathbf{x}_t) dt | \mathbf{x}_0, \mathbf{u})\right]$$

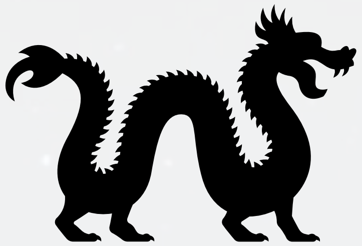
Hypothesis Testing

$$\mathcal{B} = \log p(\mathbf{x}|H_1) - \log p(\mathbf{x}|H_2)$$

Experimental Design

$$\mathcal{IG} = D[p(\mathbf{x}_{t:T}|u)||p(\mathbf{x}_{0:t})]$$

Your Tricks



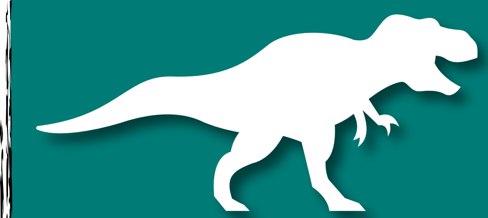
Strengthen your probabilistic dexterity.

Identity	$\frac{p(\mathbf{x})}{p(\mathbf{x})}$
Hutchinson's	$\mathbb{E}[\mathbf{z}\mathbf{z}^\top] = \mathbf{I}$
Flows	$p(\mathbf{y}) = p(\mathbf{x}) \left\| \frac{dg}{d\mathbf{x}} \right\|^{-1}$
Log-derivative	$\nabla_\phi \log q_\phi(\mathbf{z}) = \frac{\nabla_\phi q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})}$
Reparameterisation	$\mathbf{z} = g(\epsilon, \phi) \quad \epsilon \sim p(\epsilon)$
Density Ratio	$\frac{p^*(\mathbf{x})}{q(\mathbf{x})} = \frac{p(y=1\|\mathbf{x})}{p(y=-1\|\mathbf{x})}$

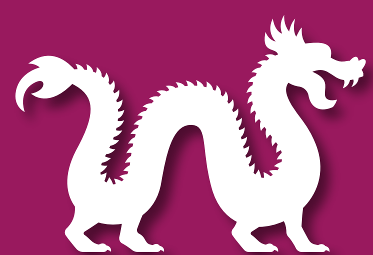
Planting the Seeds of Probabilistic Thinking

Part III: Algorithms

Learning Objectives



1. Have knowledge of different types of probabilistic models for unsupervised learning.



2. Understand generative algorithms (pixelCNN, VAEs, GANs) within the model-inference-algorithm framework.

3. Build awareness of the breadth of applications of generative models.

Beyond Classification

Move beyond associating inputs to outputs

Understand and simulate how the world evolves

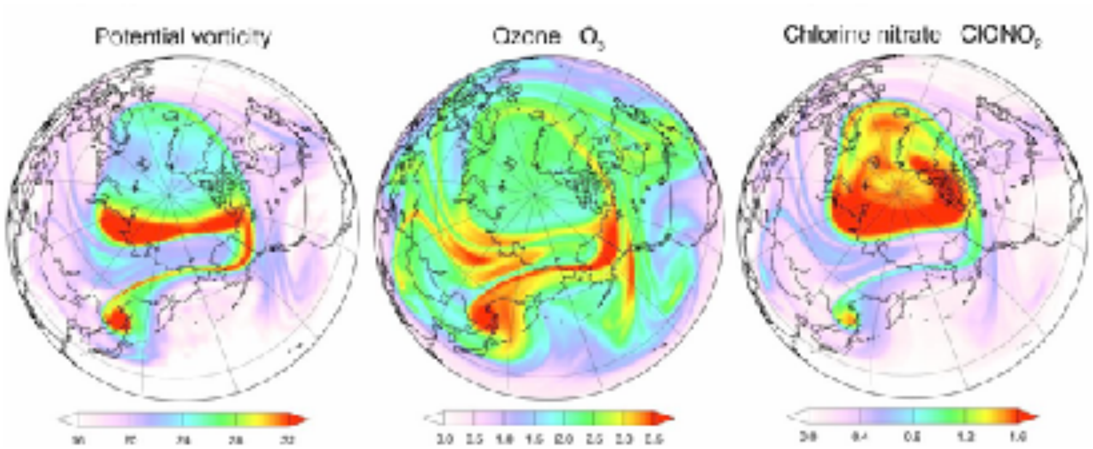
Recognise objects in the world and their factors of variation

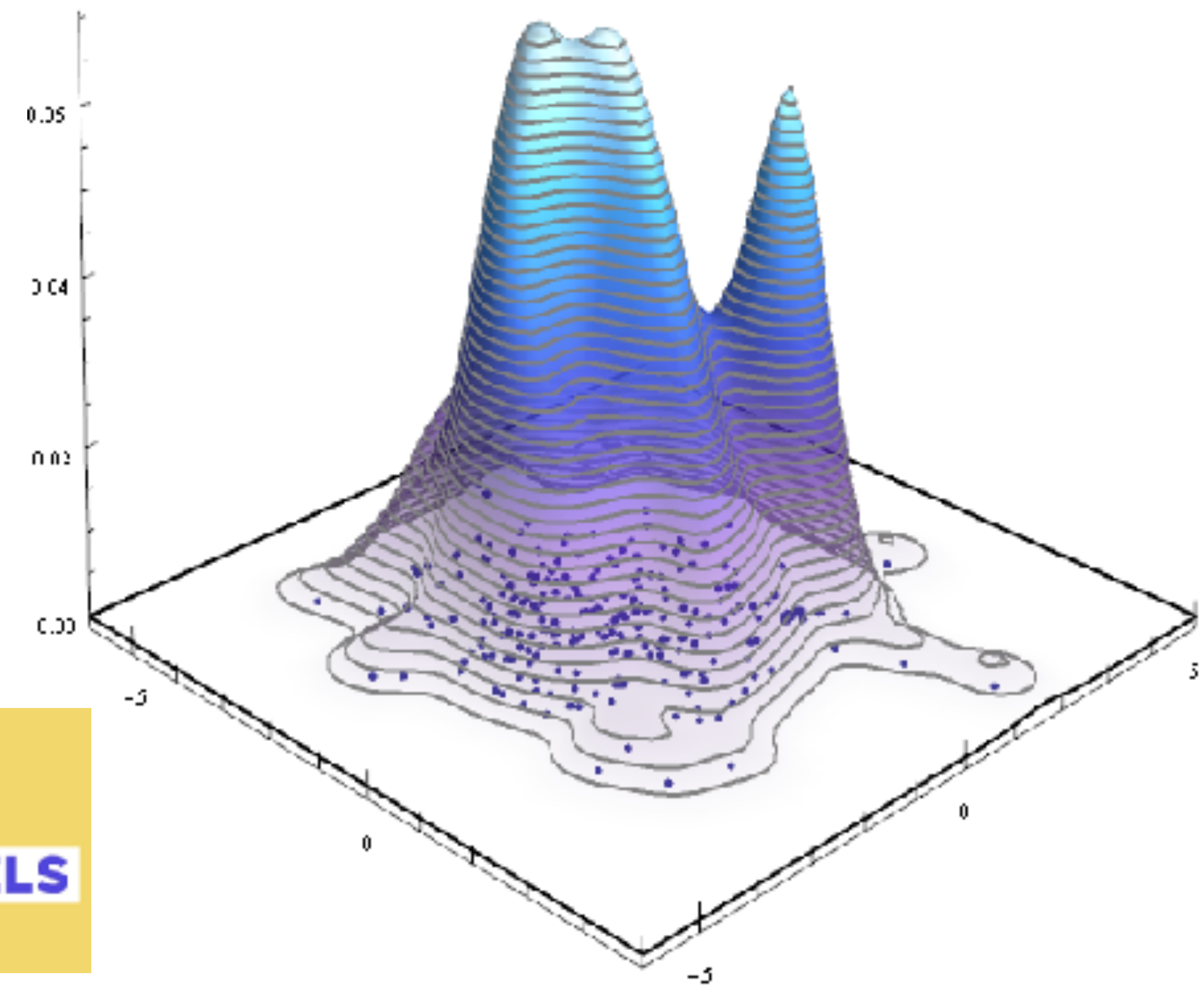
Detect surprising events in the world

Establish concepts as useful for reasoning and decision making

Anticipate and generate rich plans for the future

Generative Models





NO LABELS

A model that allows us to learn a simulator of data

Models that allow for (conditional) density estimation

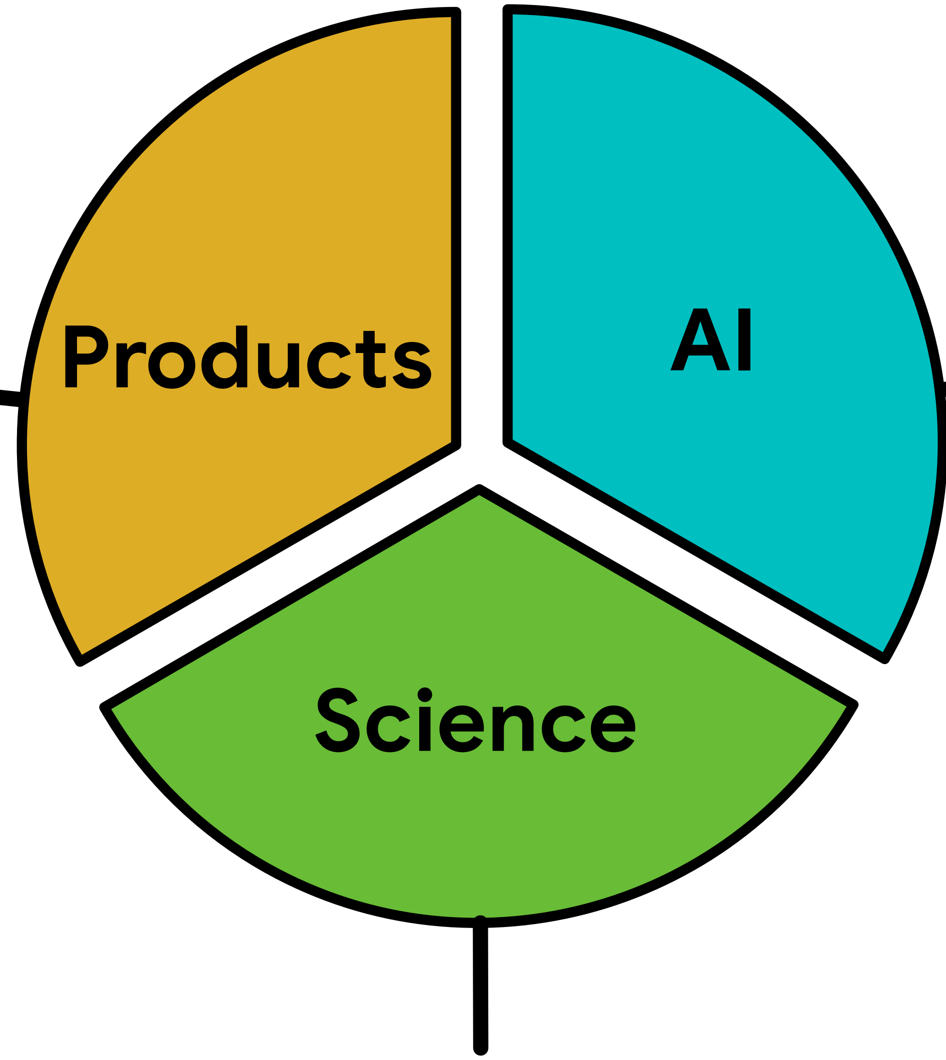
Approaches for unsupervised learning of data

Characteristics are:

- **Probabilistic** models of data that allow for uncertainty to be captured.
- **High-dimensional** data.
- **Data distribution** is targeted.

Applications

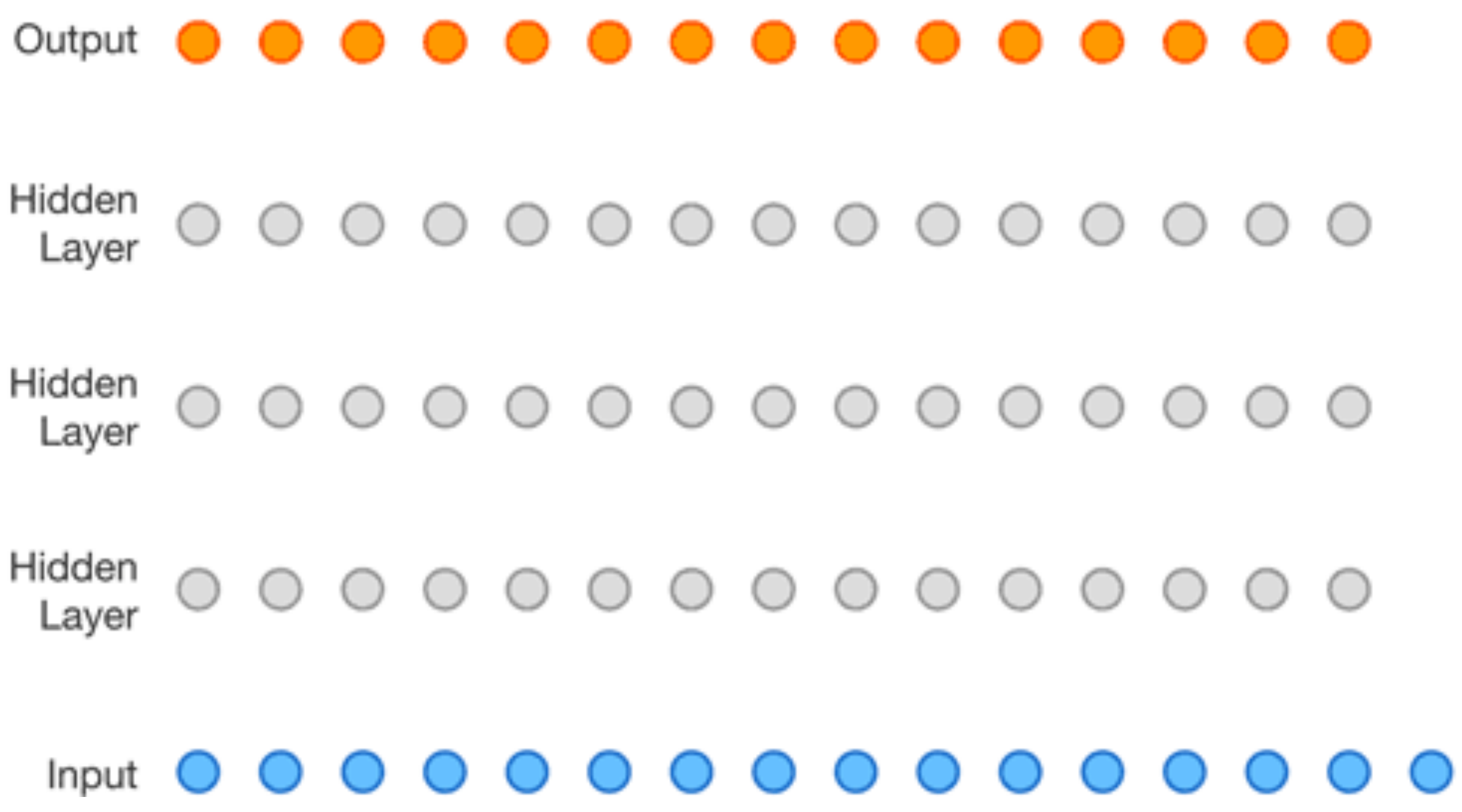
Super-resolution,
Compression,
Text-to-speech

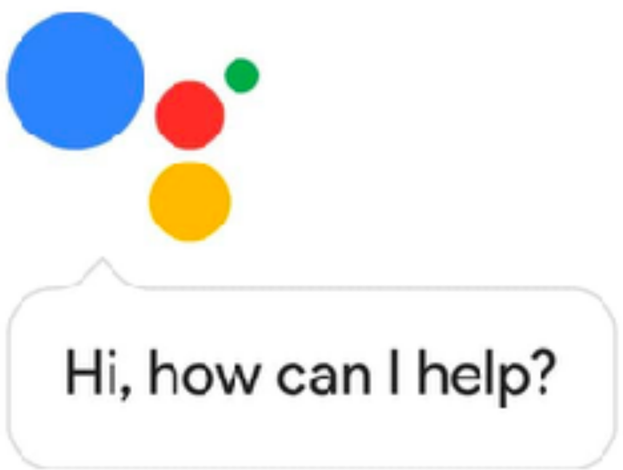


Planning,
Exploration
Intrinsic motivation
Model-based RL

Proteomics,
Drug Discovery,
Astronomy,
High-energy physics

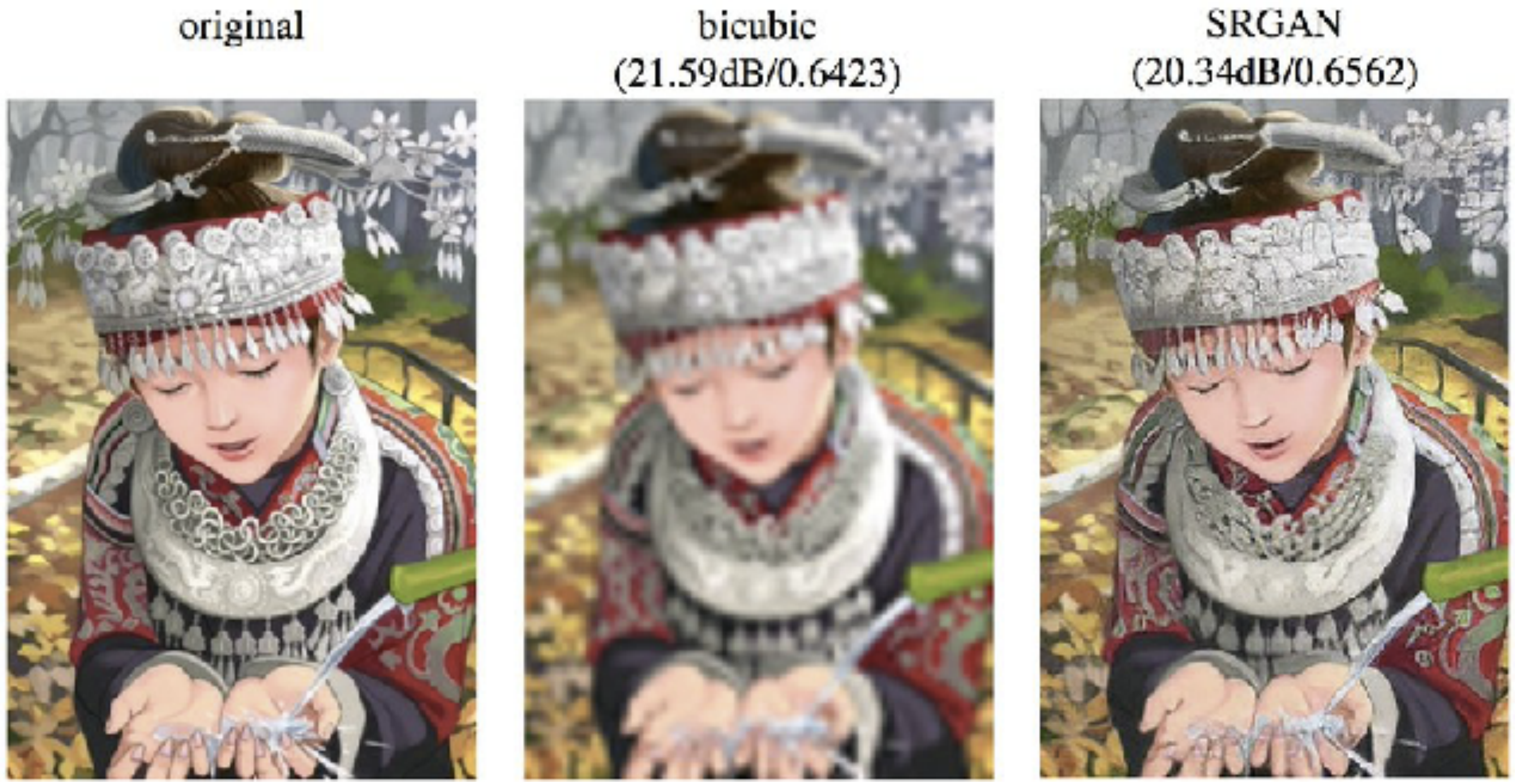
Assistive Technologies

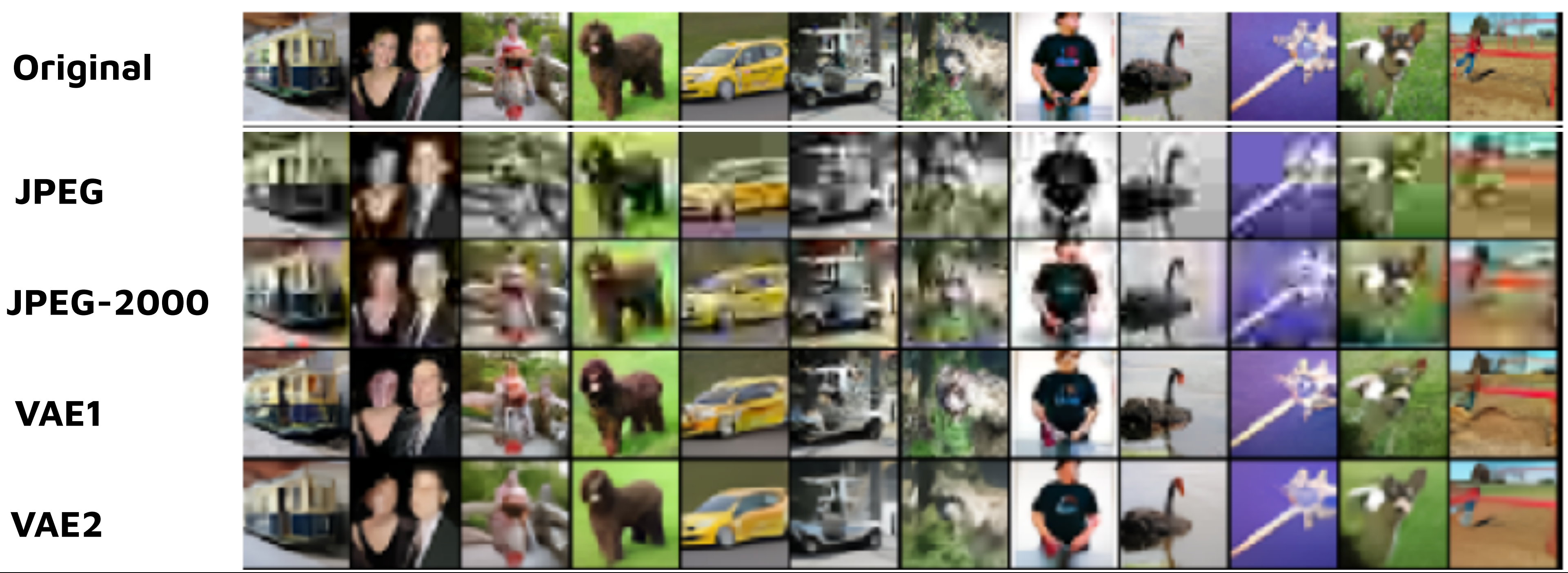




Fully-observed conditional generative model

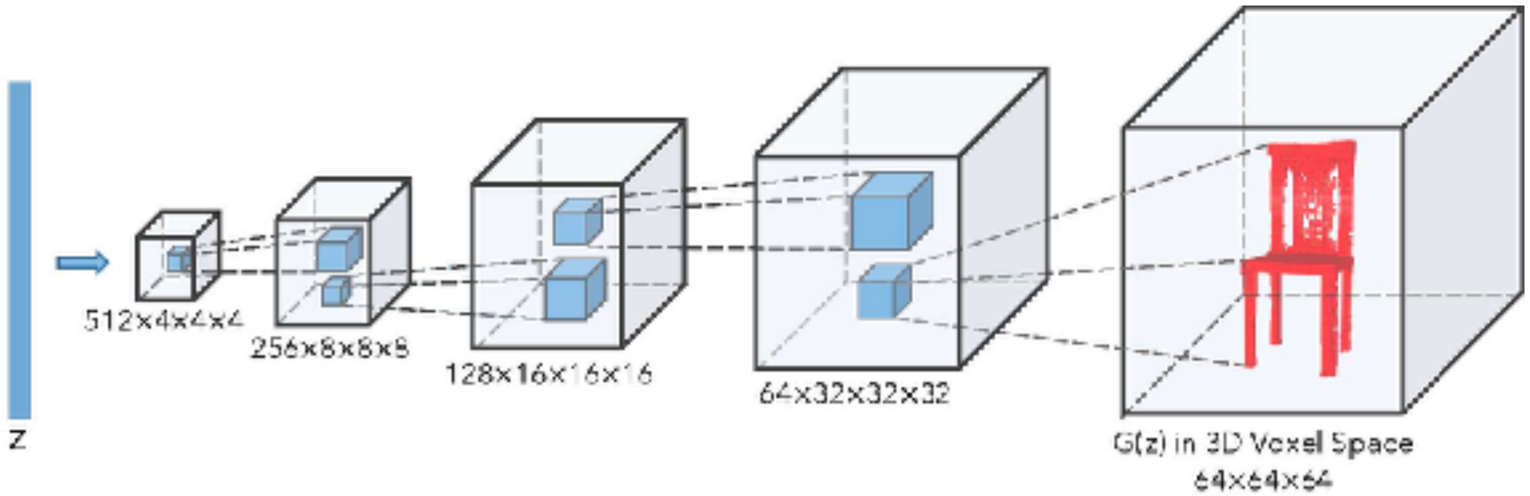
Compression-Communication

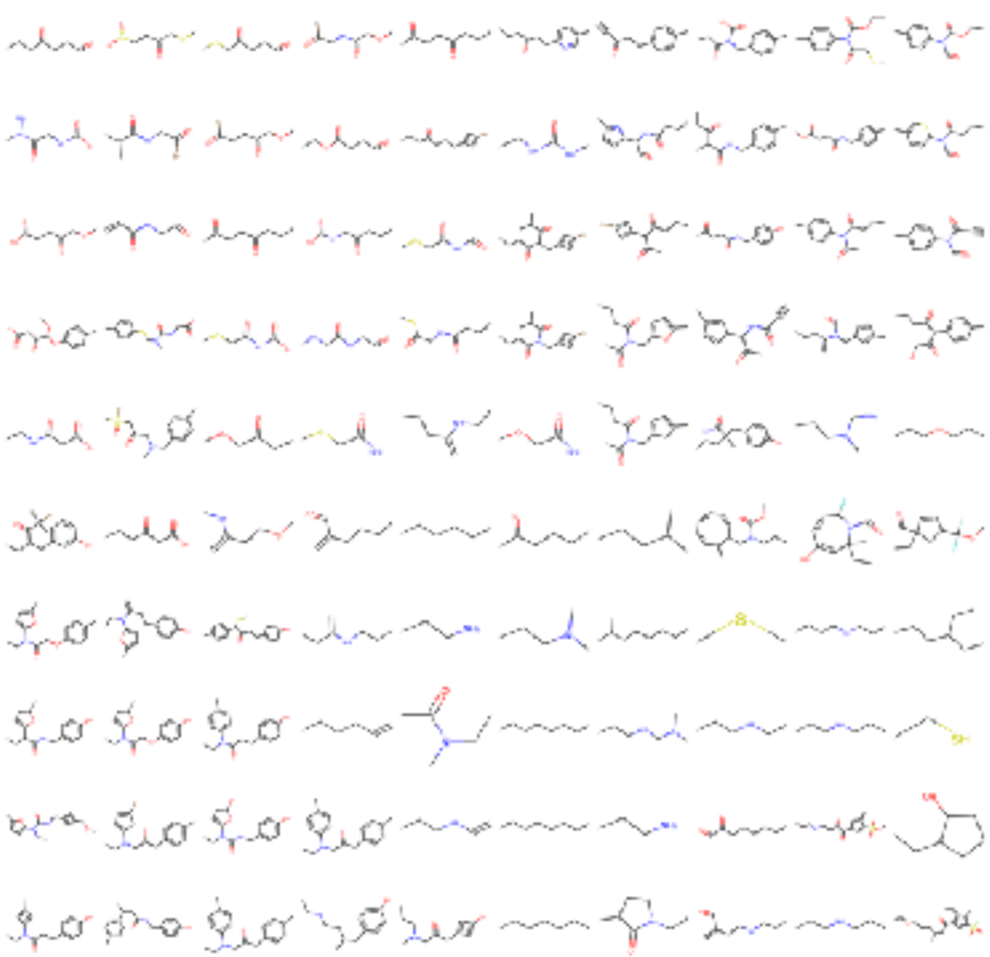


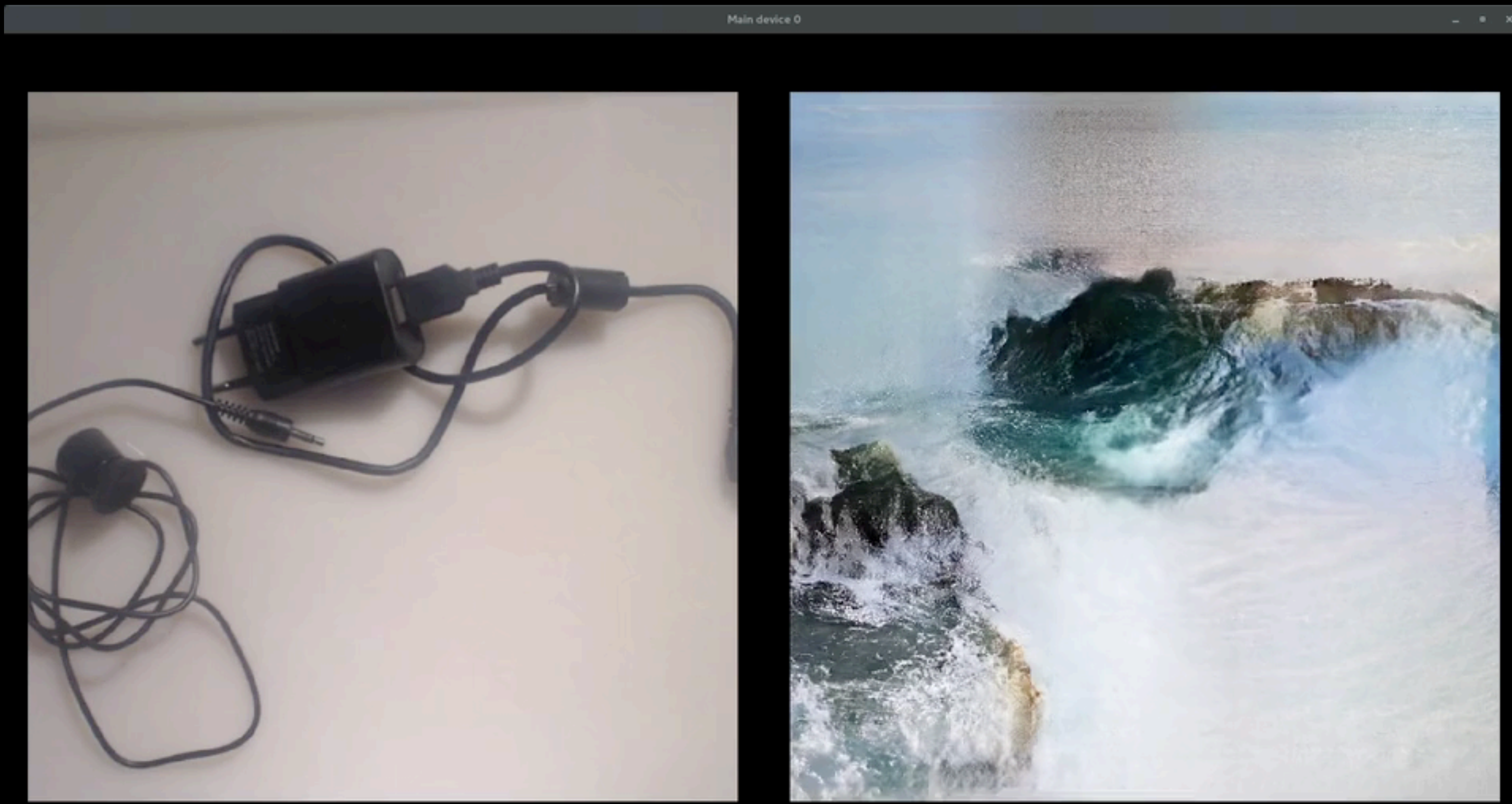


Compression rate:
0.2bits/dimension

Generative Design

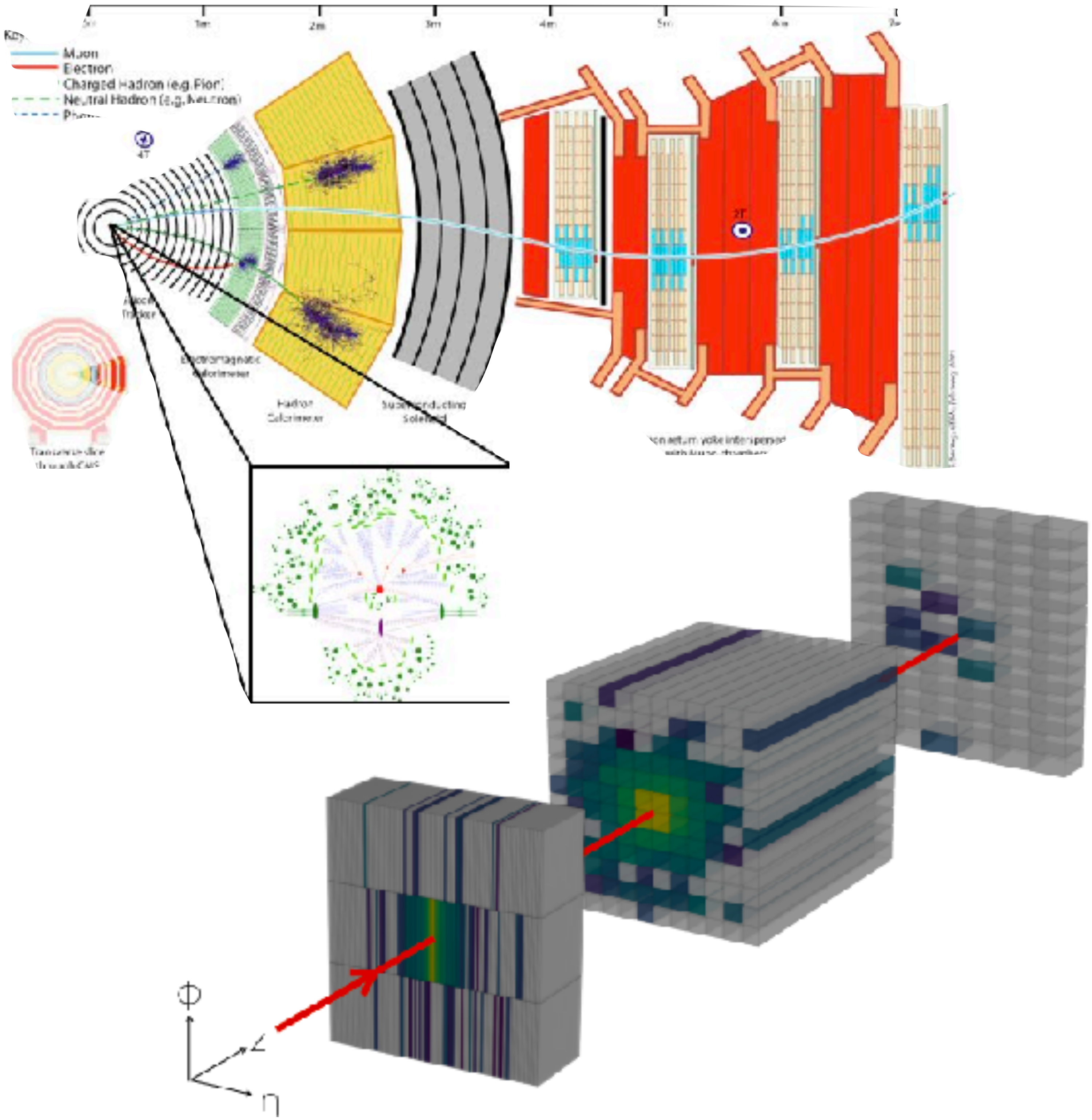


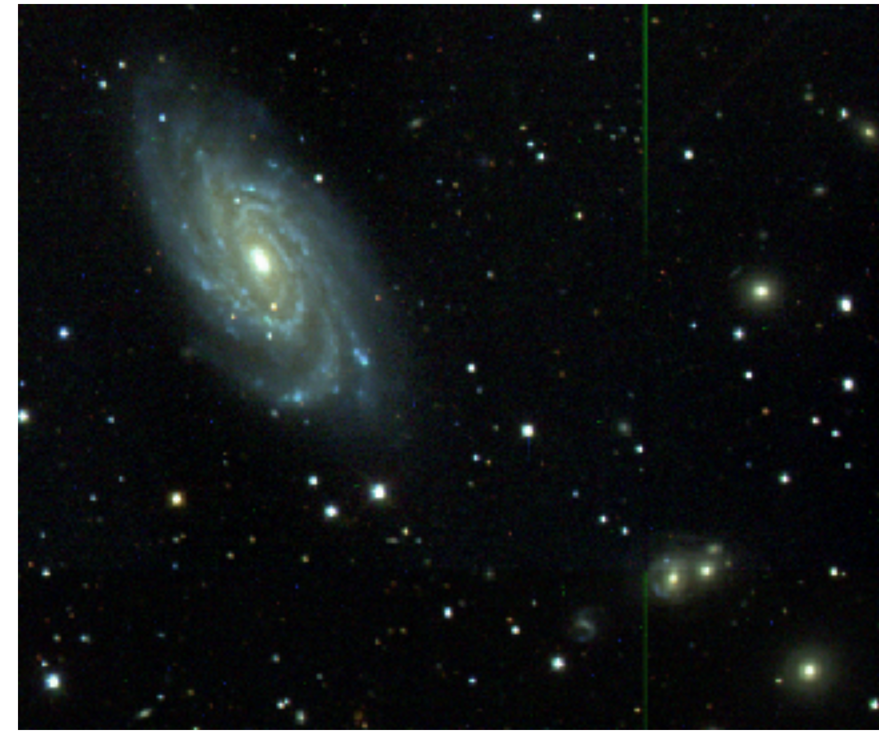


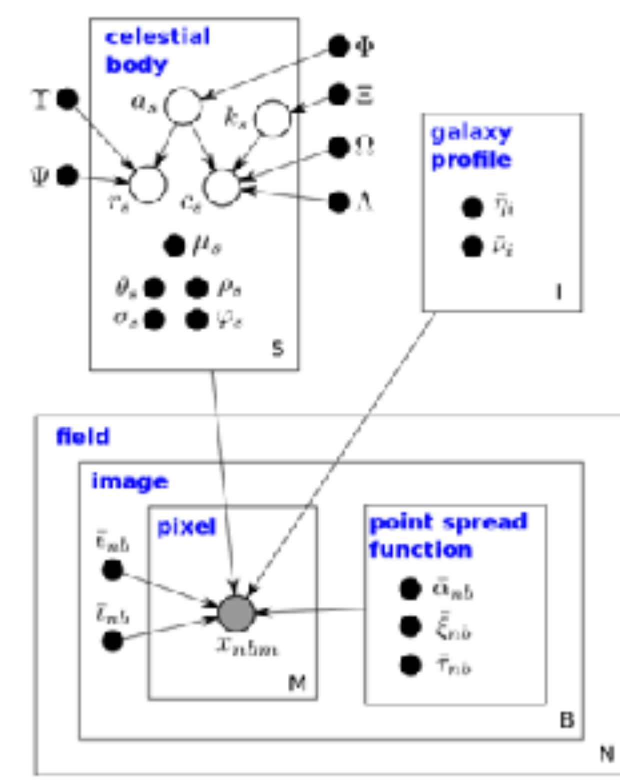


Video from work of Memo Aktem

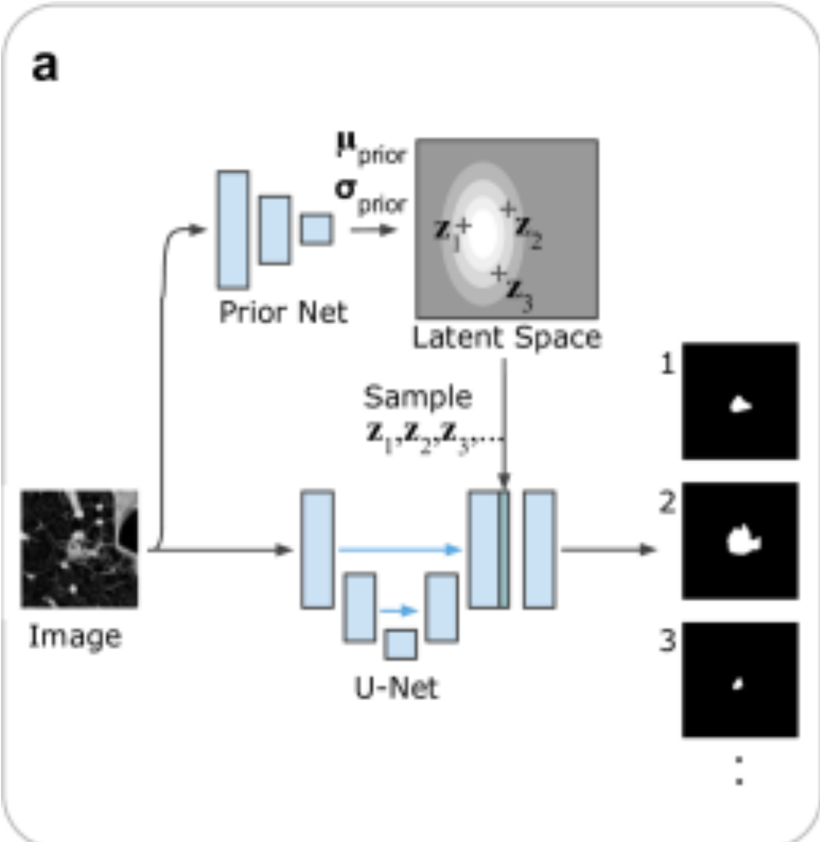
Advancing Science

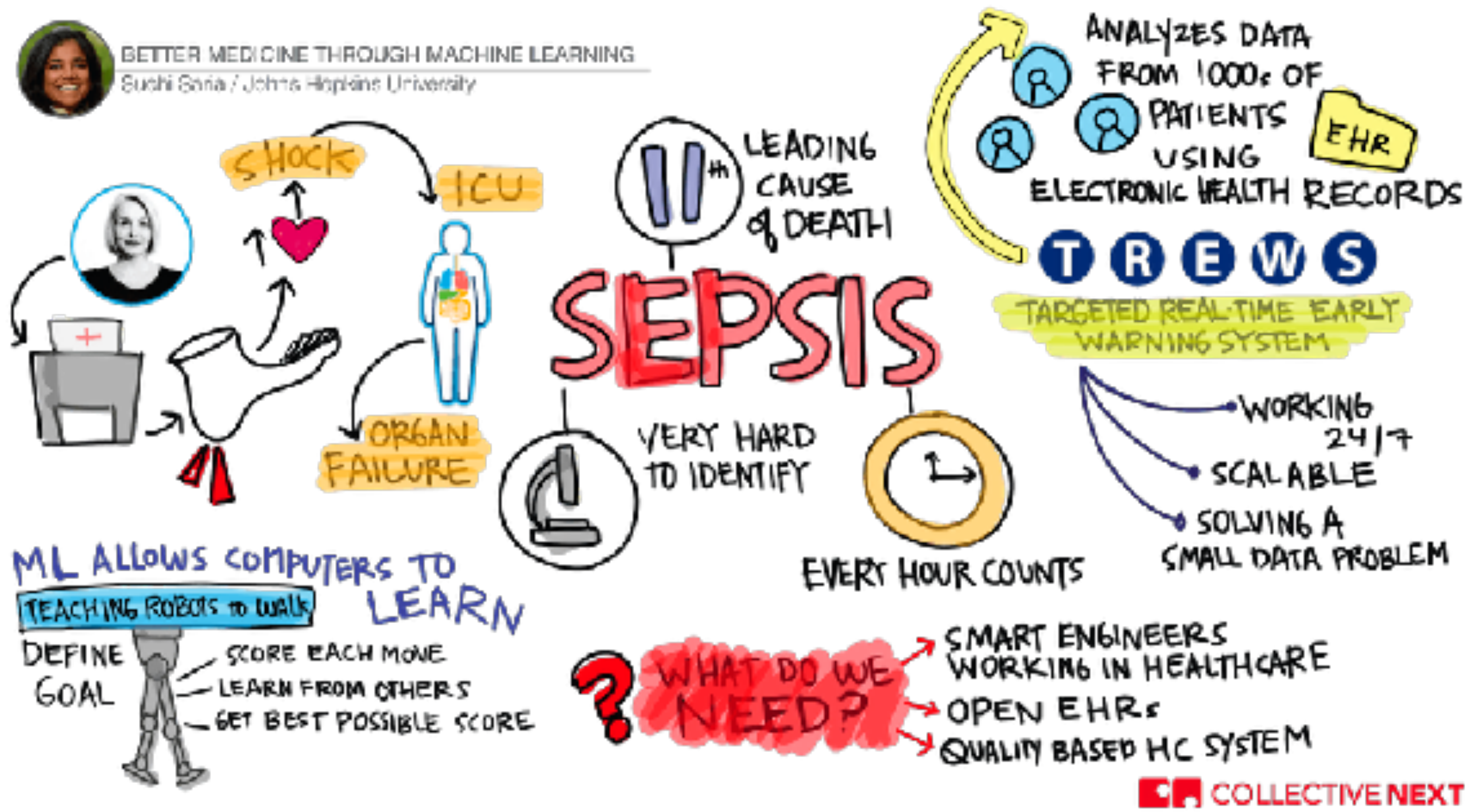






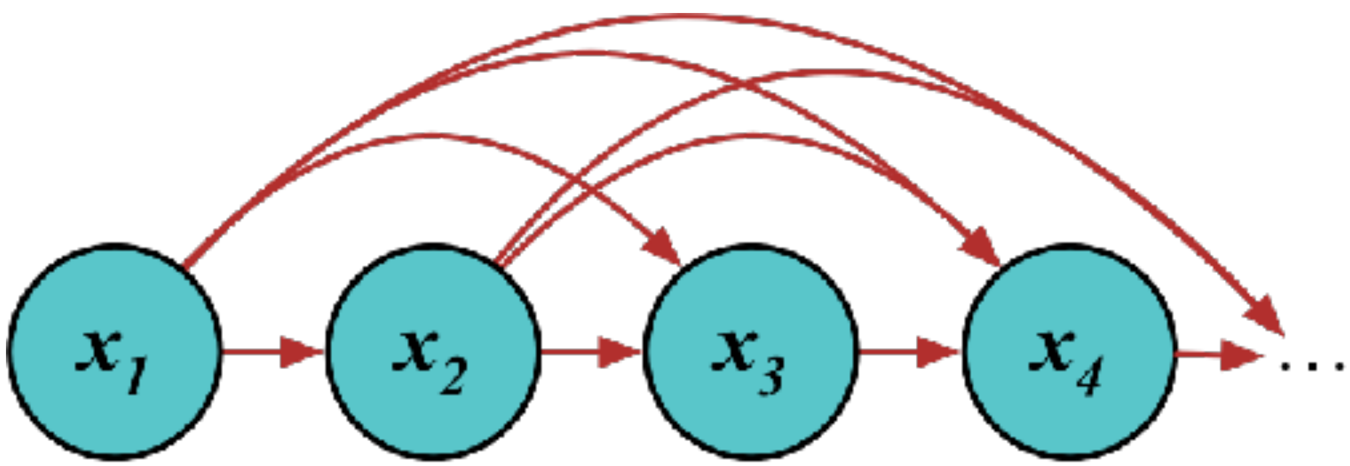
Advancing Healthcare



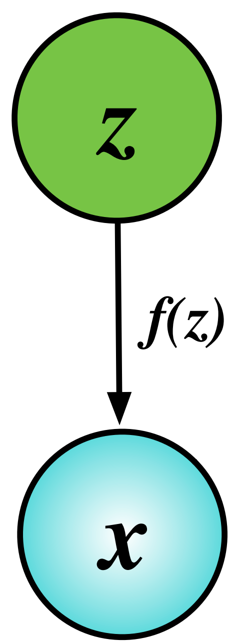


Types of Generative Models

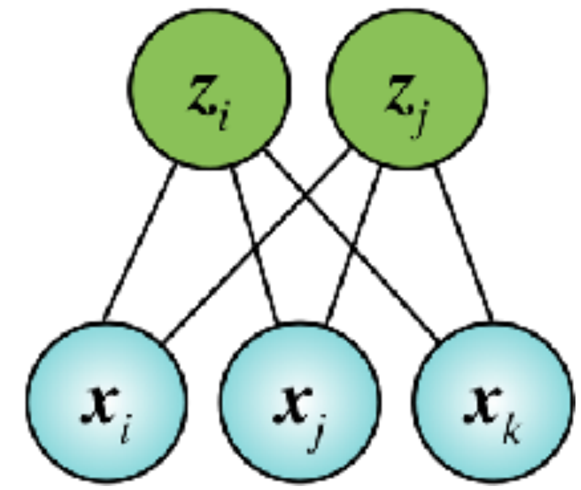
Fully-observed models



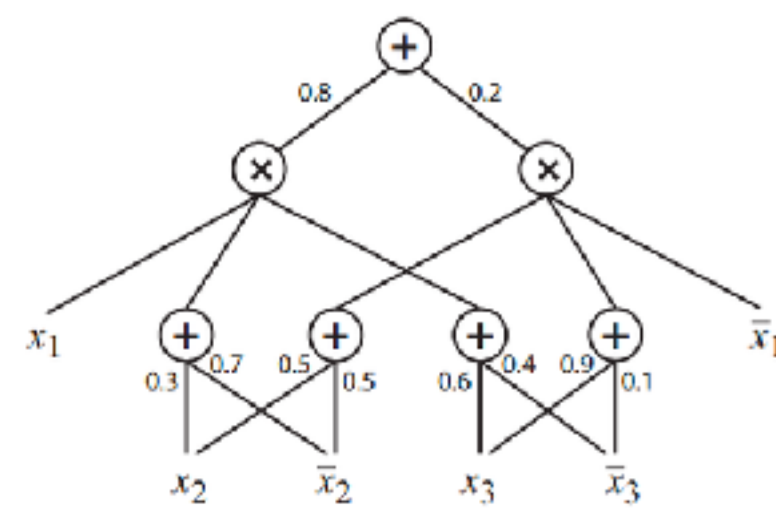
Latent variable models



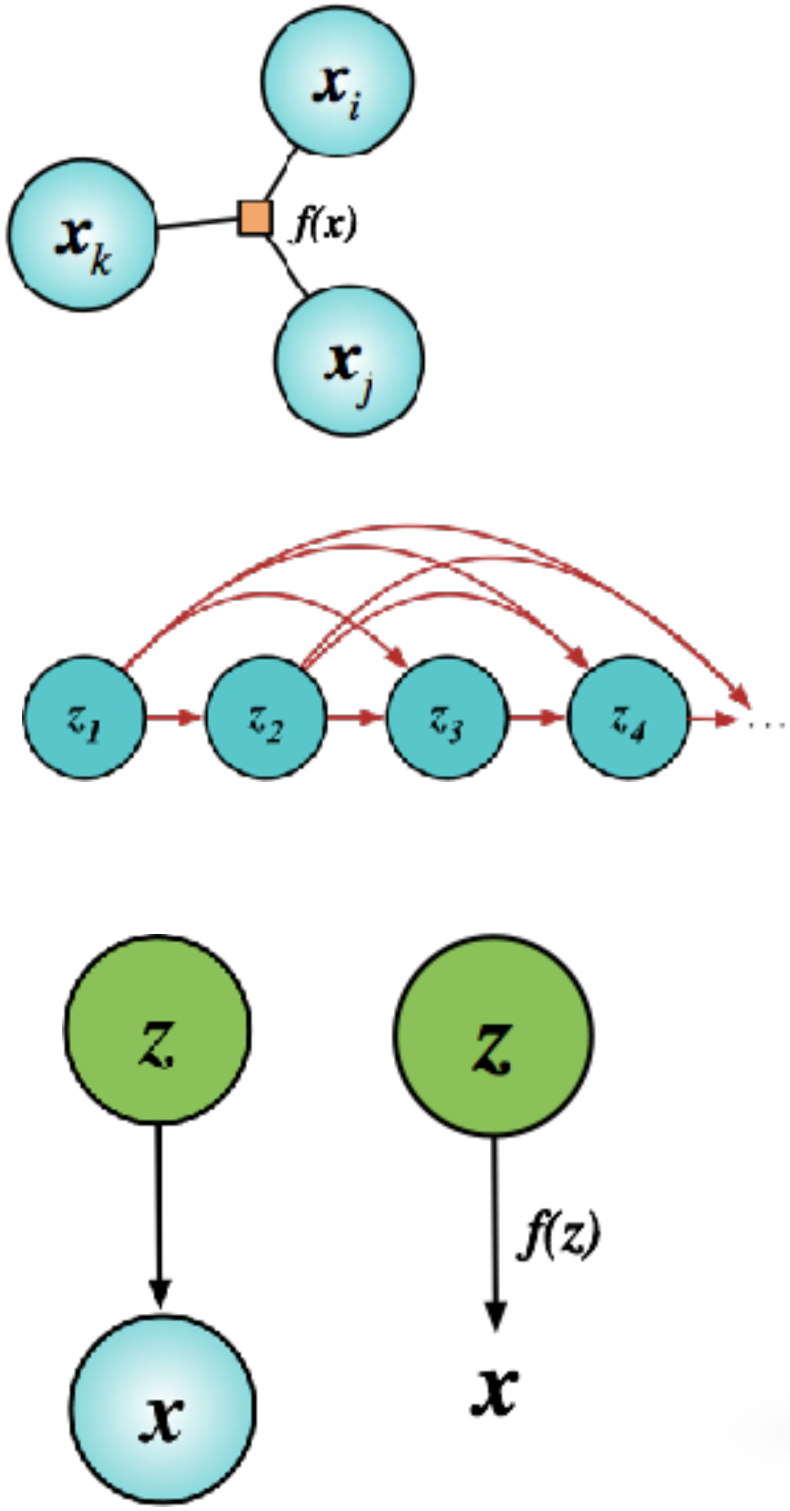
Undirected Models



Sum-Product Networks



Types of Generative Models

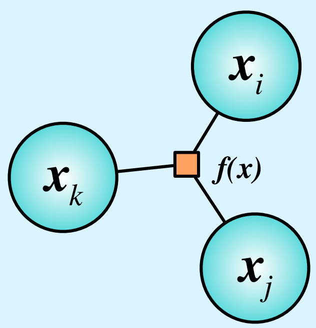


Design Dimensions

- **Data**: binary, real-valued, nominal, strings, images.
- **Dependency**: independent, sequential, temporal, spatial.
- **Representation**: continuous or discrete
- **Dimension**: parametric or non-parametric

- Computational complexity
- Modelling capacity
- Bias, uncertainty, calibration
- Interpretability

Fully-observed Models



Fully-observed models

Model observed data directly **without** introducing any new unobserved **local variables**.

Model Parameters are **global variables.**

Stochastic activations & unobserved random variables are **local variables.**

Markov Models

$$x_1 \sim \mathrm{Cat}(x_1|\pi)$$

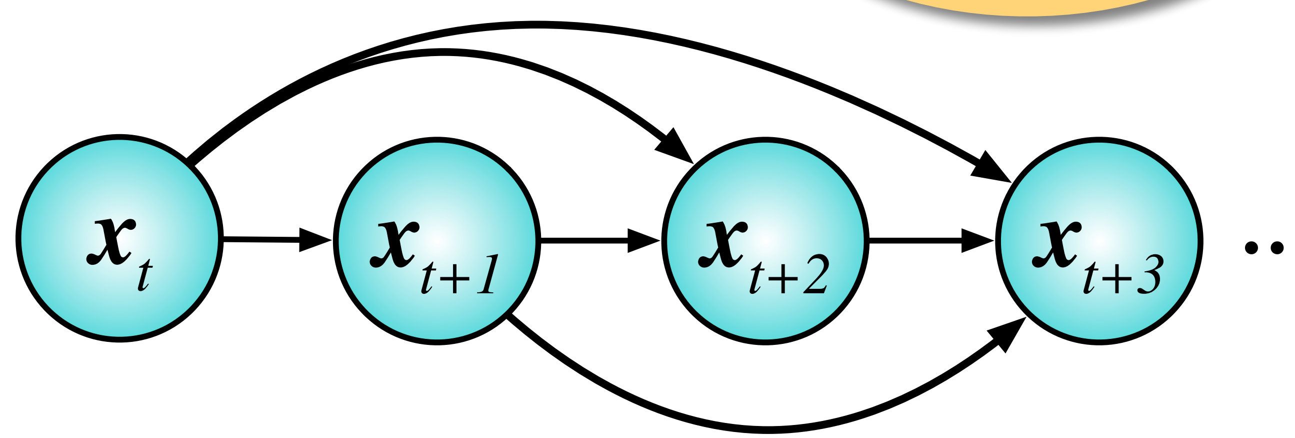
$$x_2 \sim \mathrm{Cat}(x_2|\pi(\mathbf{x}_1))$$

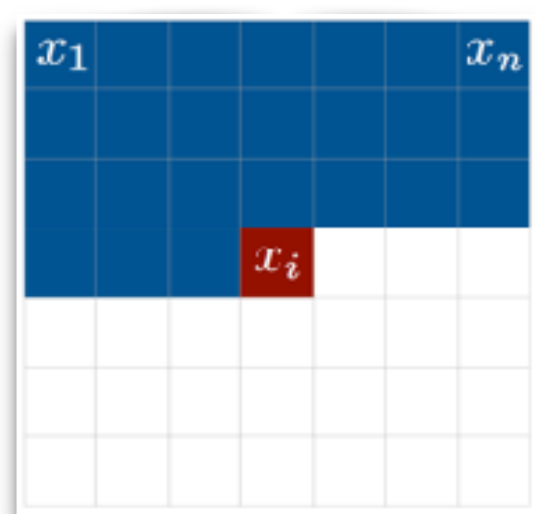
$$\cdots$$

$$x_i \sim \mathrm{Cat}(x_i|\pi(\mathbf{x}_{<n}))$$

$$p(\mathbf{x}) = \prod_i p(x_i|f(\mathbf{x}_{<i};\boldsymbol{\theta}))$$

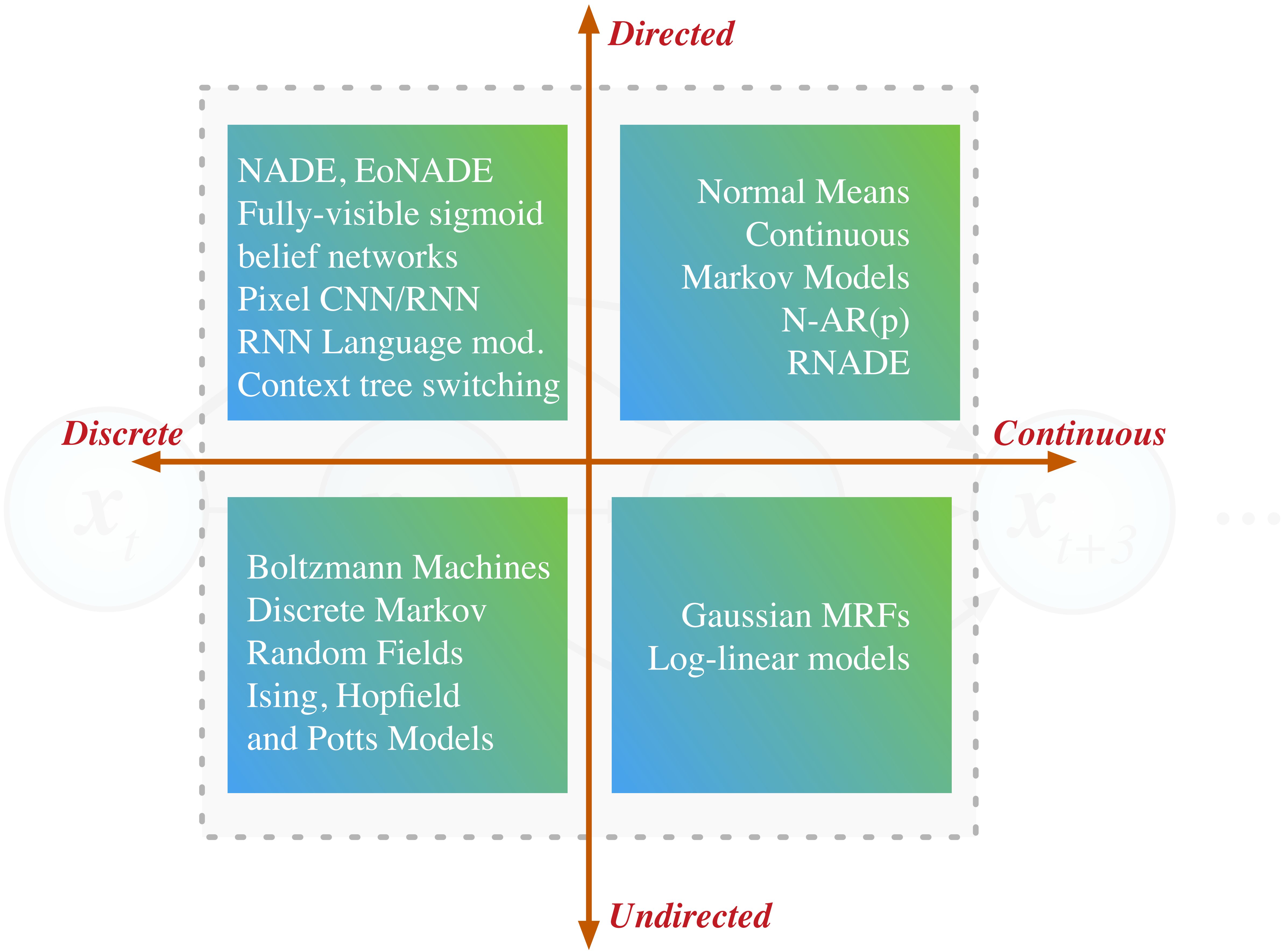
All conditional probabilities described by deep networks.



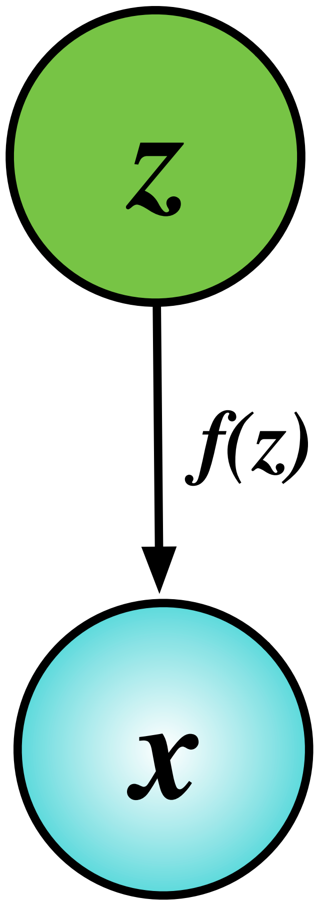


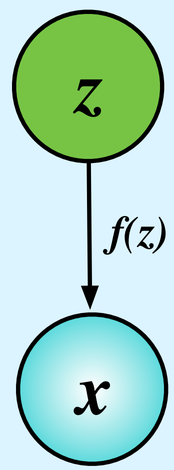
Properties

- + Can directly encode how observed points are related.
- + *Any data* type can be used
- + For directed graphical models:
  - + **Parameter learning simple**: Log-likelihood is directly computable, no approximation needed.
  - + Easy to scale-up to large models, many optimisation tools available.
  - - Order sensitive.
- - For undirected models,
  - - **Parameter learning difficult**: Need to compute normalising constants.
- - **Generation can be slow**: iterate through elements sequentially, or using a Markov chain.


Directed
Undirected
Discrete
Continuous
NADE, EoNADE
Fully-visible sigmoid belief networks
Pixel CNN/RNN
RNN Language mod.
Context tree switching
Normal Means
Continuous Markov Models
N-AR(p)
RNADE
Boltzmann Machines
Discrete Markov Random Fields
Ising, Hopfield and Potts Models
Gaussian MRFs
Log-linear models

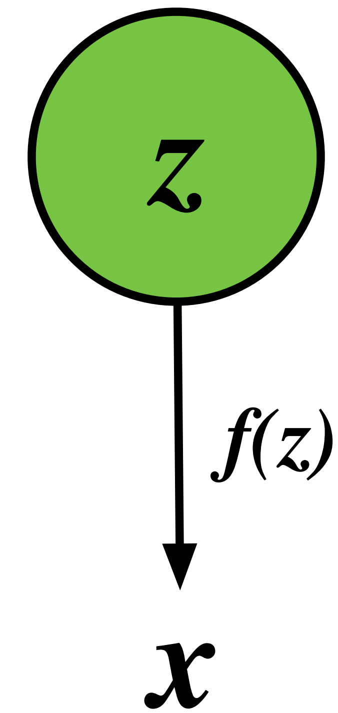
Latent Variable Models





Latent variable models

Introduce an unobserved **local random variables** that represents hidden causes.



Prescribed models

Use observer likelihoods and assume observation noise.

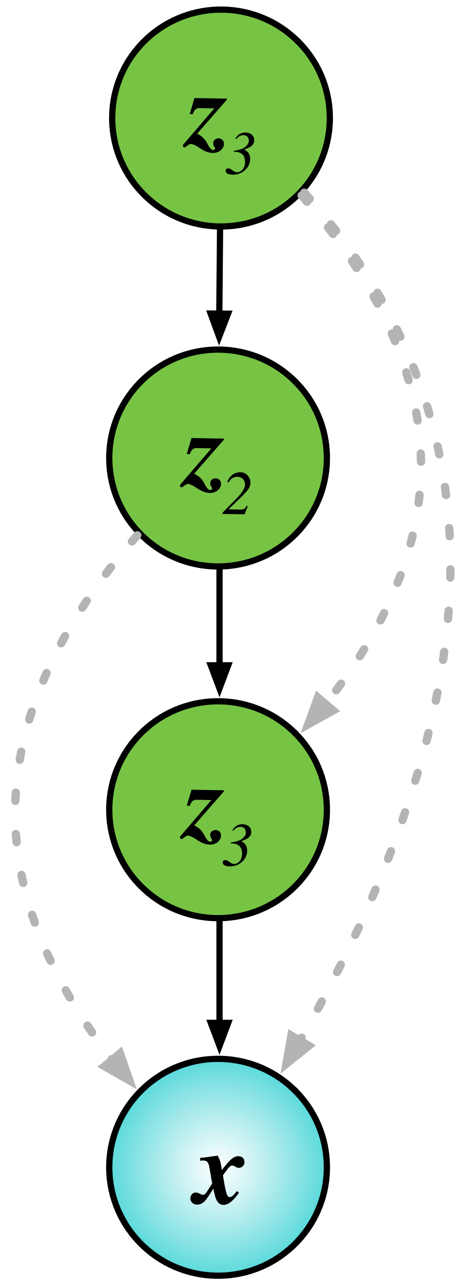
Implicit models

Likelihood-free or simulation-based models.

Diggle and Gratton (1984); Mohamed and Lakshminarayanan (2016)

Prescribed Models

Deep Latent Gaussian Model



$$\mathbf{z}_3 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathbf{z}_2 | \mathbf{z}_3 \sim \mathcal{N}(\mu(\mathbf{z}_3), \Sigma(\mathbf{z}_3))$$

$$\mathbf{z}_1 | \mathbf{z}_2 \sim \mathcal{N}(\mu(\mathbf{z}_2), \Sigma(\mathbf{z}_2))$$

$$\mathbf{x} | \mathbf{z}_1 \sim \mathcal{N}(\mu(\mathbf{z}_1), \Sigma(\mathbf{z}_1))$$

Convolutional DRAW

Properties

- \+ Easy sampling.
- \+ Easy way to include hierarchy and depth.
- \+ Easy to encode structure believed to generate the data
- \+ Avoids order dependency assumptions: marginalisation of latent variables induces dependencies.
- \+ Latents provide compression and representation the data.
- \+ Scoring, model comparison and selection possible using the marginalised likelihood.
- \- Inversion process to determine latents corresponding to a input is difficult in general
- \- Difficult to compute marginalised likelihood requiring approximations.
- \- Not easy to specify rich approximations for latent posterior distribution.

Cascaded Indian Buffet process
Hierarchical Dirichlet process

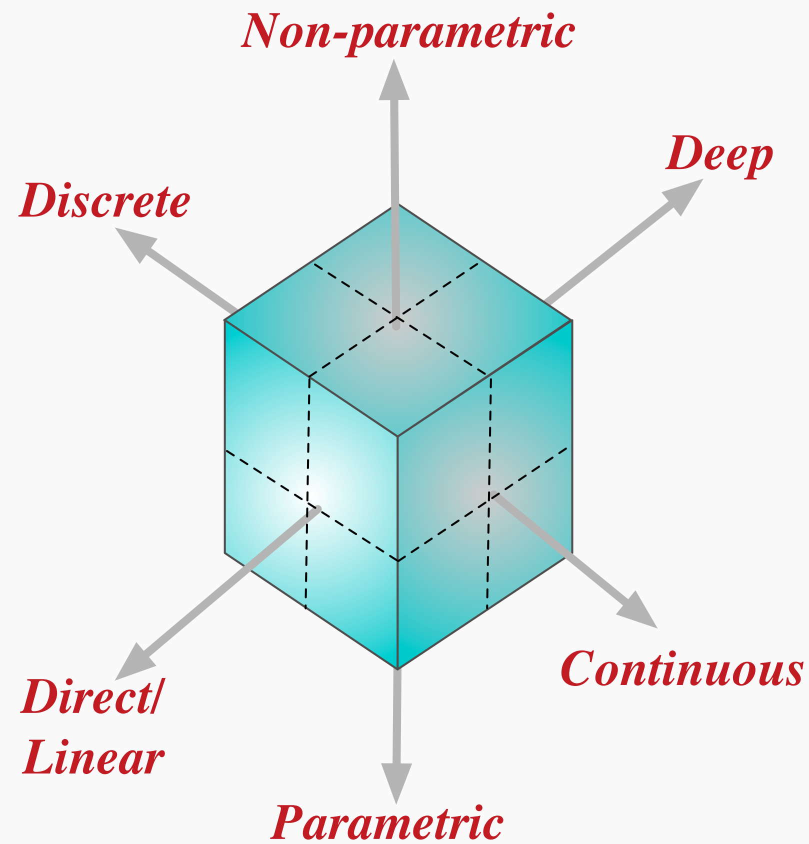
Deep Nonparametric Discrete

Sigmoid Belief Net
Deep auto-regressive networks (DARN)

Deep Parametric Discrete

Indian buffet process
Dirichlet process mixture

Direct Nonparametric Discrete



Deep Gaussian processes
Recurrent Gaussian Process
GP State space model

Deep Nonparametric Continuous

Hidden Markov Model
Discrete LVM
Sparse LVMs

Linear Parametric Discrete

Nonlinear factor analysis
Nonlinear Gaussian belief network
Deep Latent Gaussian (VAE, DRAW)

Deep Parametric Continuous

PCA, factor analysis
Independent components analysis
Gaussian LDS
Latent Gauss Field

Linear Parametric Continuous

Gaussian process LVM

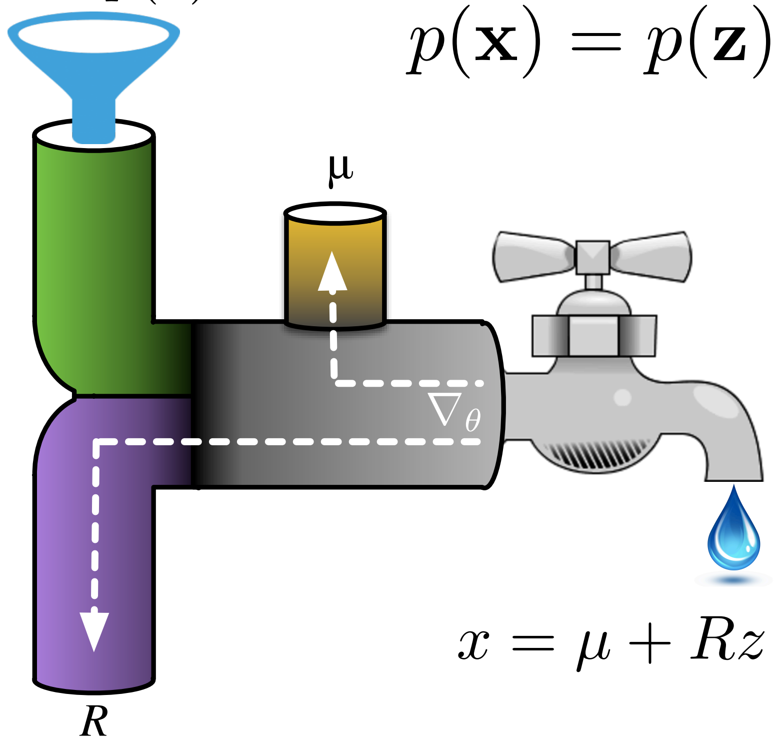
Direct Nonparametric Continuous

Implicit Models

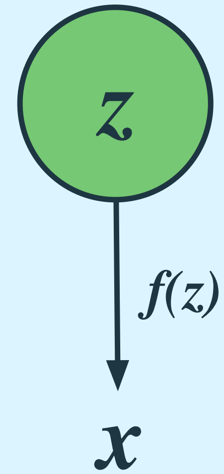
Change of variables for invertible functions

$z \sim p(z)$

$$p(\mathbf{x}) = p(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}$$



$x = \mu + Rz$



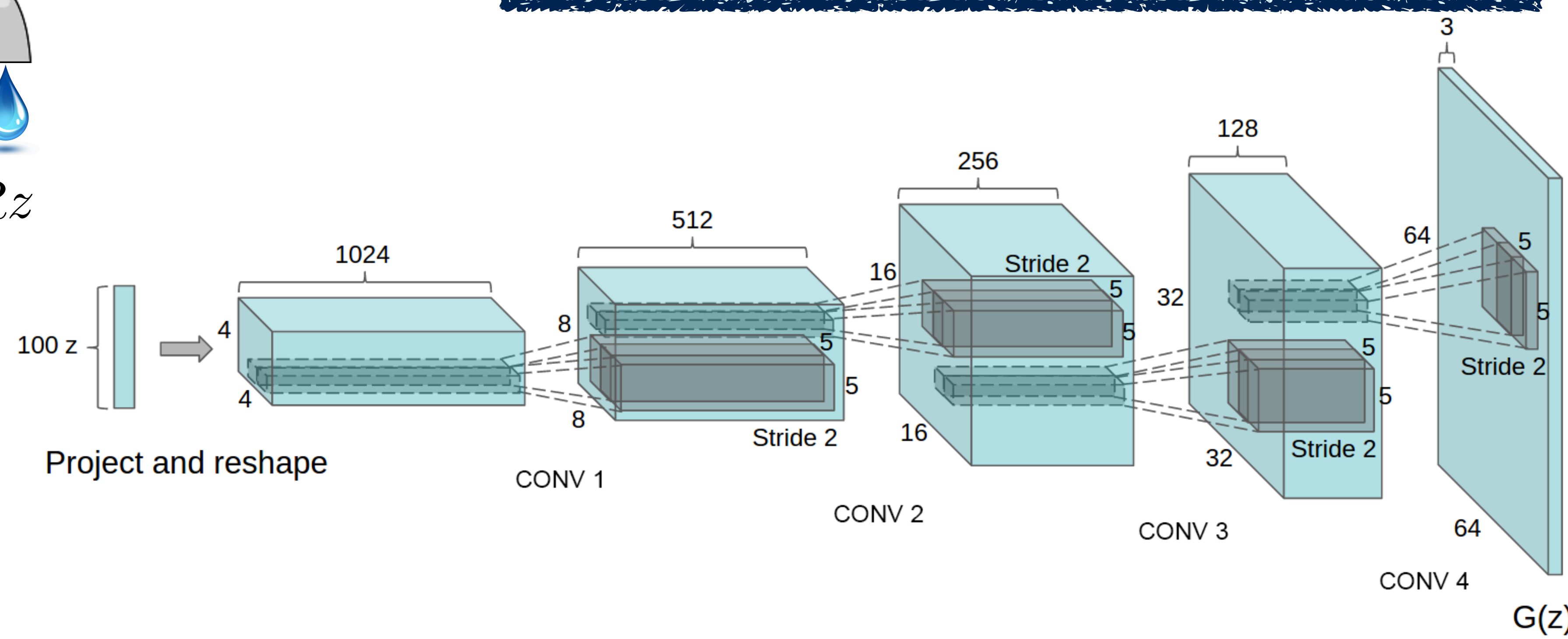
Implicit models

Transform an unobserved **noise source** using a parameterised **function**.

Generator Networks

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathbf{x} = f(\mathbf{z}; \boldsymbol{\theta})$$

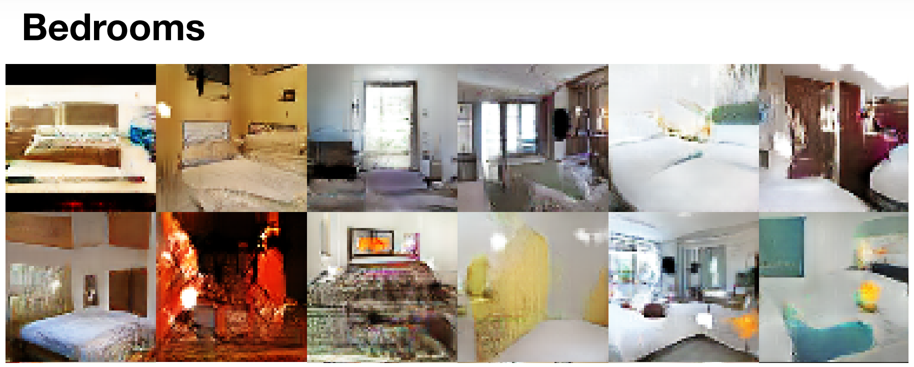


The transformation function is parameterised by a linear or deep network (fully-connected, convolutional or recurrent).

Properties

+ Easy sampling, and natural to specify.
+ Easy to compute expectations without knowing final distribution.
+ Can exploit with large-scale classifiers and convolutional networks.
- ***Difficult to satisfy constraints***: Difficult to maintain invertibility, and challenging optimisation.
- ***Lack of noise model*** (likelihood):
  - Difficult to extend to generic data types
  - Difficult to account for noise in observed data.
  - Hard to compute marginalised likelihood for model scoring, comparison and selection.

Convolutional generative adversarial network



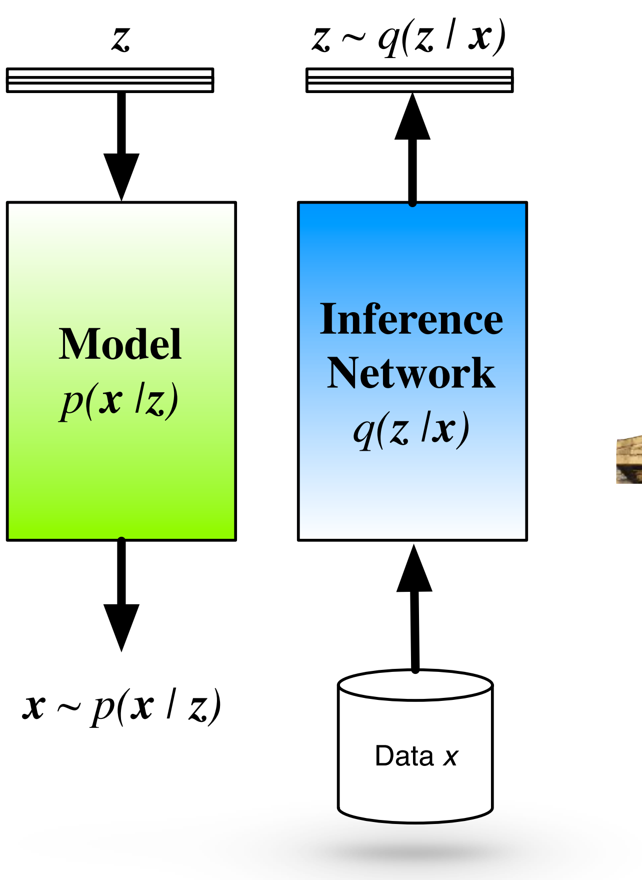
Stochastic Differential Equations
Hamiltonian and Langevin SDE
Diffusion Models
Non- and volume preserving flows

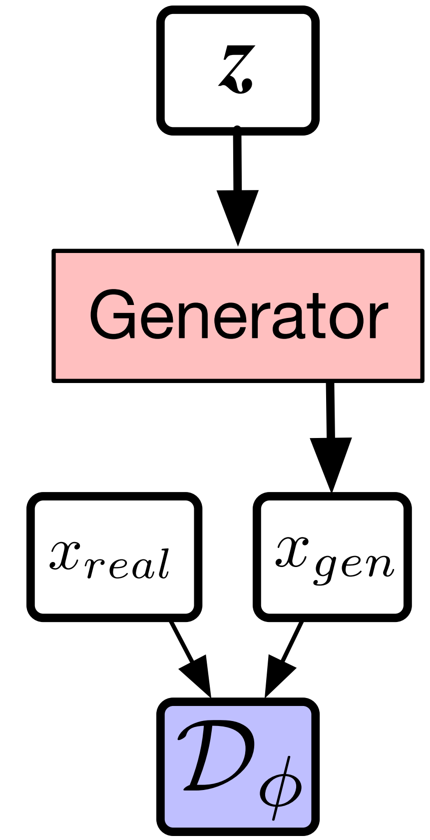
One-liners and inverse sampling
Distrib. warping
Normalising flows
GAN generator nets
Non- and volume preserving transforms

Diffusions
Continuous time

Functions
Discrete time

Model-Inference-Algorithm





Prescribed latent variable models and variational inference

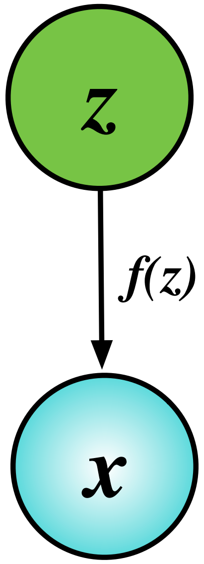
Implicit latent variable models and estimation-by-comparison

Variational Autoencoders (VAEs)

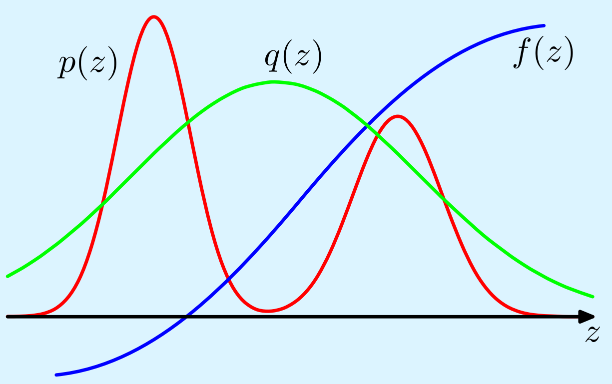
Generative Adversarial Networks (GANs)

Model Evidence

Model evidence (or marginal likelihood, partition function):
Integrating out any global and local variables enables model scoring, comparison, selection, moment estimation, normalisation, posterior computation and prediction.



We take steps to improve the model evidence for given data samples.



Learning principle: Model Evidence

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

Integral is intractable in general and requires approximation.

Basic idea:
Transform the integral into an expectation over a simple, known distribution.

Variational Inference

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

$$\mathcal{F}(\mathbf{x}, q) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})] - KL[q(\mathbf{z})\|p(\mathbf{z})]$$

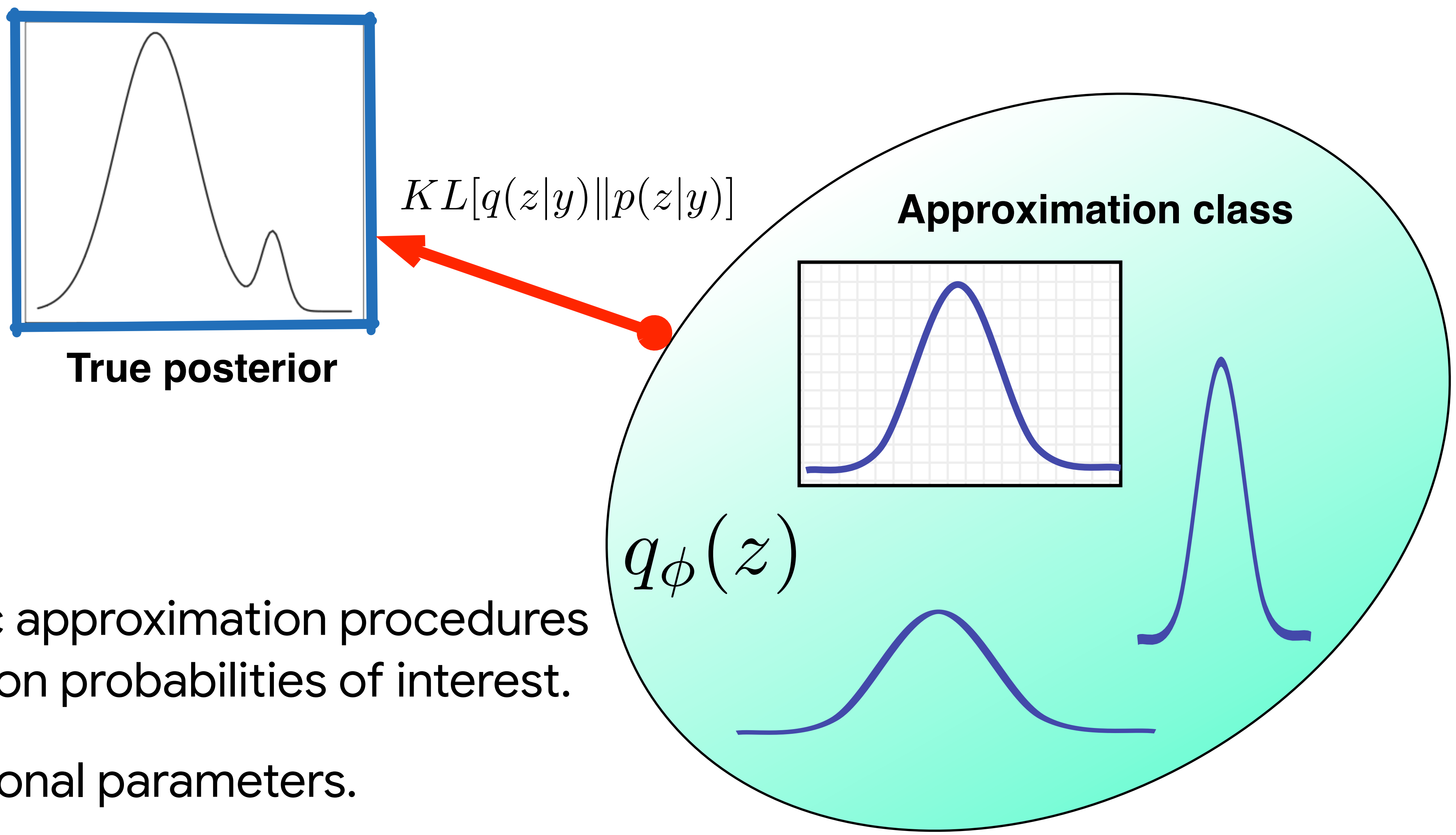
This bound is exactly of the form we are looking for.

- **Variational free energy:** We obtain a functional and are free to choose the distribution *q(z)* that best matches the true posterior.
- **Evidence lower bound (ELBO):** principled bound on the marginal likelihood, or model evidence.
- Certain choices of q(z) makes this quantity easier to compute. Examples to come.

Identity | Bounding

Variational Methods

Variational Principle
General family of methods for approximating complicated densities by a simpler class of densities.



Deterministic approximation procedures with bounds on probabilities of interest.

Fit the variational parameters.

Variational Bound

Interpreting the bound:

$$\mathcal{F}(\mathbf{x}, q) = \boxed{\mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})]} - \boxed{KL[q(\mathbf{z})\|p(\mathbf{z})]}$$

Approx. Posterior

Reconstruction

Penalty

- **Approximate posterior distribution *q(z)*:** Best match to true posterior *p(z|y)*, one of the unknown inferential quantities of interest to us.
- **Reconstruction cost**: The expected log-likelihood measure how well samples from *q(z)* are able to explain the data *y*.
- **Penalty:** Ensures the the explanation of the data *q(z)* doesn't deviate too far from your beliefs *p(z)*. A mechanism for realising Okham's razor.

Variational Bound

$$\mathcal{F}(\mathbf{x}, q) = \underbrace{\mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction}} - \underbrace{KL[q(\mathbf{z})\|p(\mathbf{z})]}_{\text{Penalty}}$$

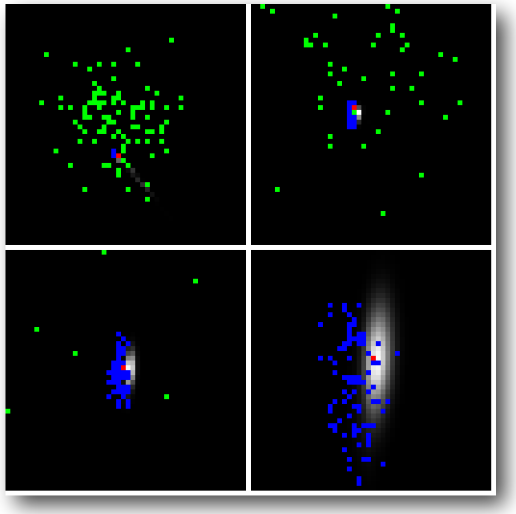
Approx. Posterior

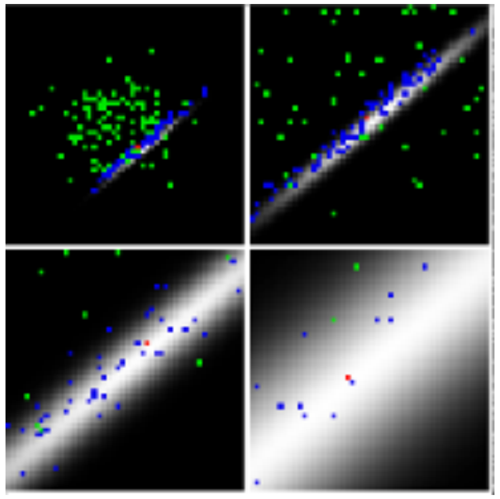
Some comments on *q*:

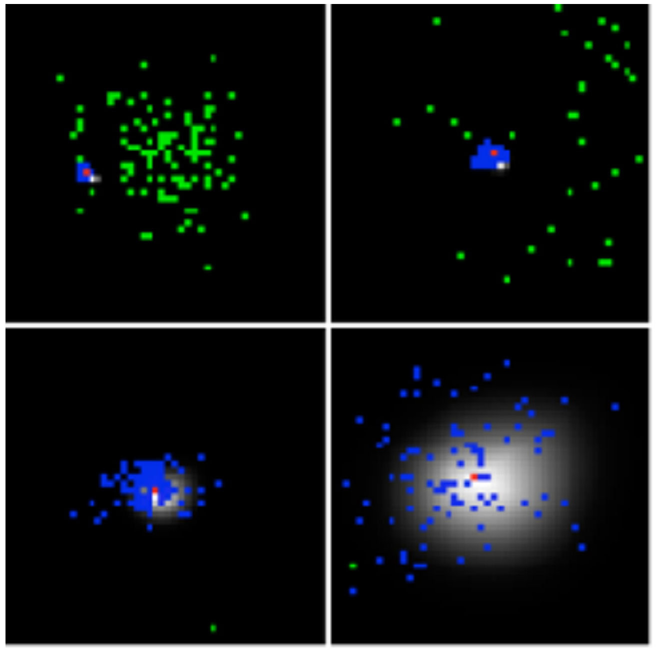
- **Integration is now optimisation**: optimise for *q(z)* directly.
  - I write *q(z)* to simplify the notation, but it depends on the data, *q(z|x)*.
  - *Easy convergence assessment* since we wait until the free energy (loss) reaches convergence.
- **Variational parameters:** parameters of *q(z)*
  - E.g., if a Gaussian, variational parameters are mean and variance.
  - Optimisation allows us to *tighten the bound* and get as close as possible to the true marginal likelihood.

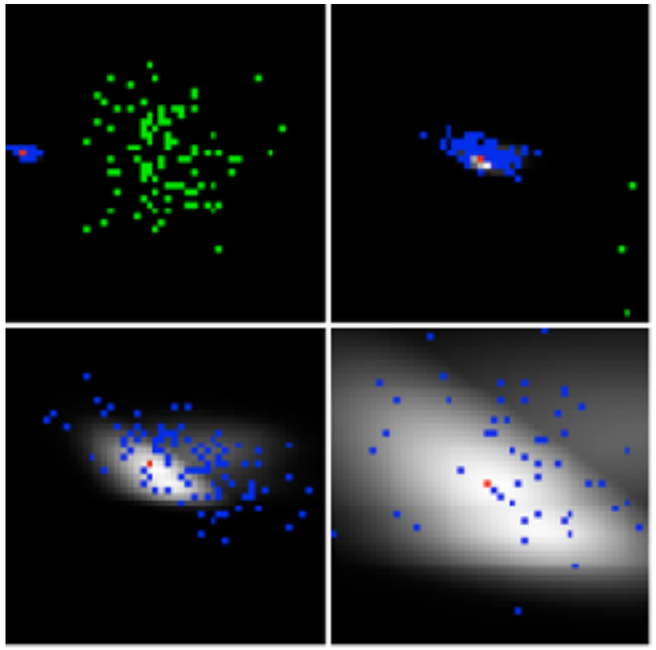
Real Posteriors

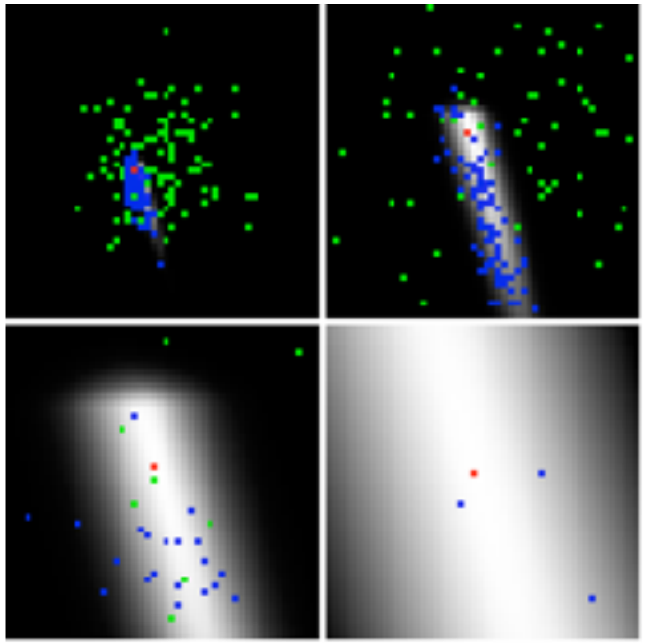
Require flexible approximations for the types of posteriors we are likely to see.







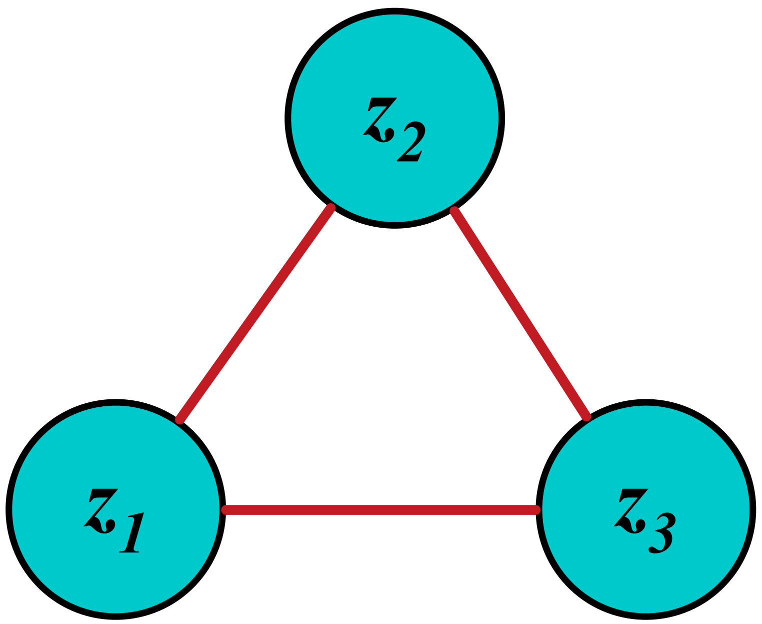


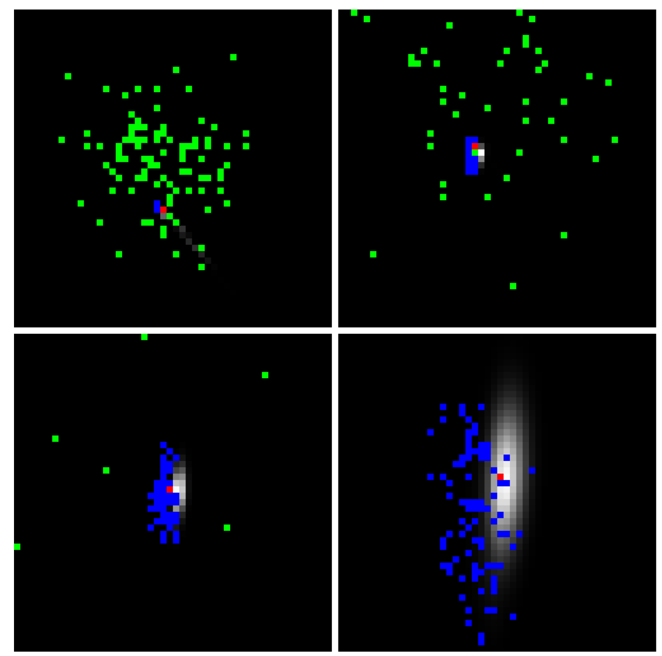


Mean-Fields

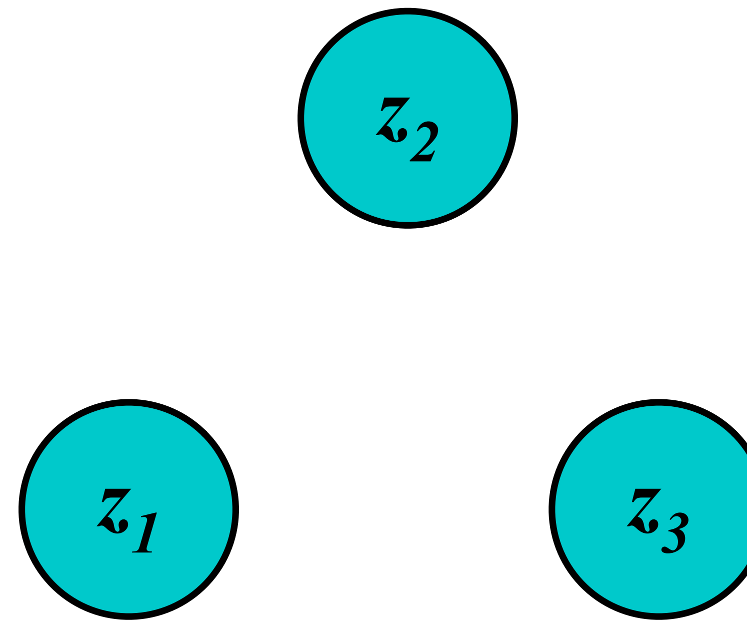
Mean-field methods assume that the distribution is factorised.

True Posterior





Fully-factorised



Most Expressive ⟷ *Least Expressive*

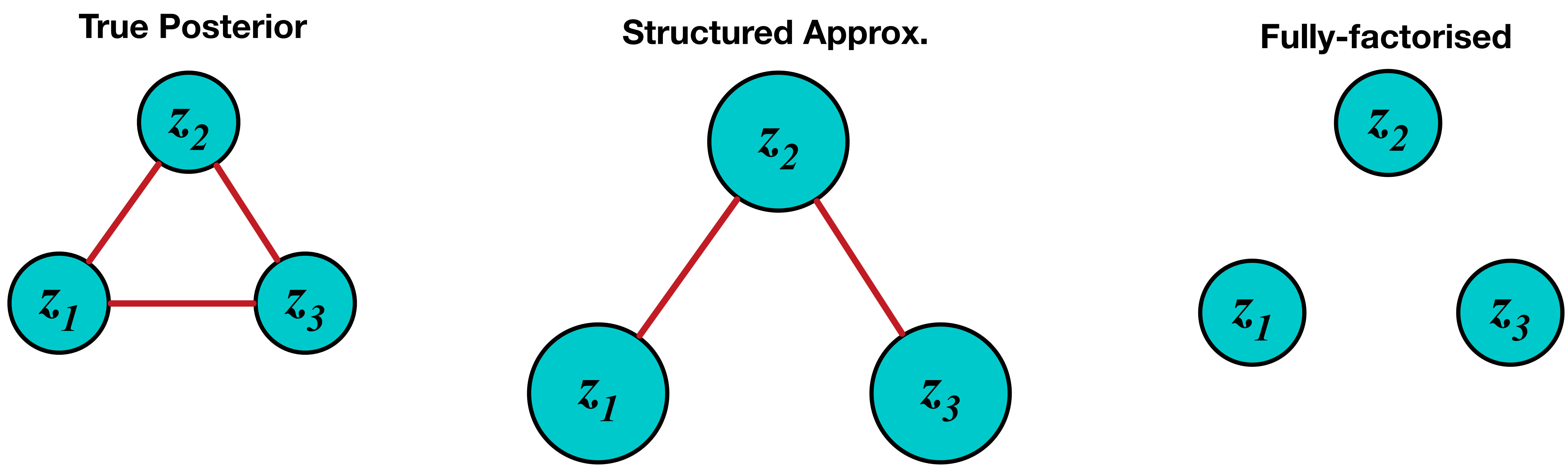
$$q^*(z|x) \propto p(x|z)p(z)$$

$$q_{MF}(z|x) = \prod_k q(z_k)$$

Restricted class of approximations: every dimension (or subset of dimensions) of the posterior is independent.

Structured Mean-field

Structured mean-field: introduce dependencies into our factorisation.



Most Expressive ⟵ ⟶ *Least Expressive*

$$q^*(z|x) \propto p(x|z)p(z) \qquad q(z) = \prod_k q_k(z_k|\{z_j\}_{j\neq k}) \qquad q_{MF}(z|x) = \prod_k q(z_k)$$

Latent Gaussian Models

Examples: GP regression or DLGM.

Probabilistic Model $z \sim \mathcal{N}(z|0,1) \qquad y \sim p(y|f_\theta(z))$

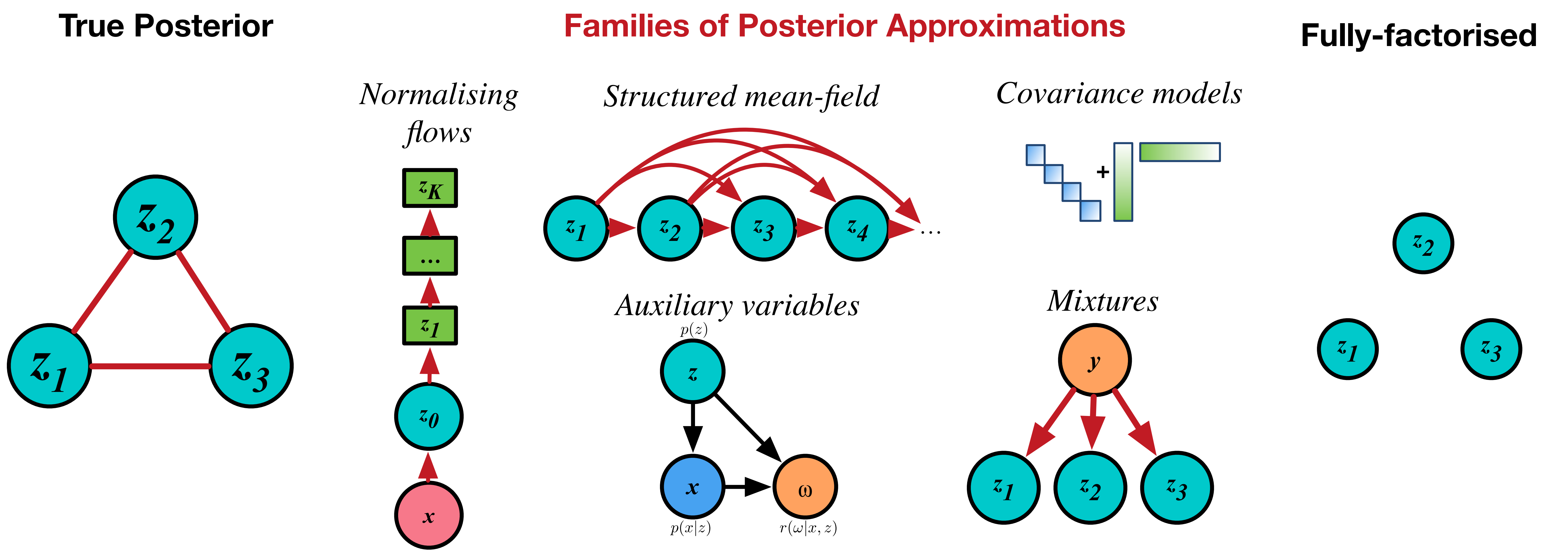
Mean-field approx $q(z) = \prod_i \mathcal{N}(z_i|\mu_i, \sigma_i^2)$

Variational bound
$$\mathcal{F}(y,q) = \mathbb{E}_{q(z)}[\log p(y|z)] - KL[q(z)\|p(z)]$$
$$\mathcal{F}(y,q) = \mathbb{E}_{q(z)}[\log p(y|z)] - \sum_i KL[q(z_i)\|p(z_i)]$$

$$\mathcal{F}(y,q) = \mathbb{E}_{q(z)}[\log p(y|z)] - \sum_i KL[\mathcal{N}(z_i|\mu_i,\sigma_i^2)\|\mathcal{N}(z_i|0,1)]$$

$$\boxed{\mathcal{F}(y,q) = \mathbb{E}_{q(z)}[\log p(y|f_\theta(z))] - \frac{1}{2}\sum_i(\sigma_i^2 + \mu_i^2 - 1 - \ln \sigma_i^2)}$$

Families of Approximations



Most Expressive ←→ ***Least Expressive***

$$q^*(z|x) \propto p(x|z)p(z)$$

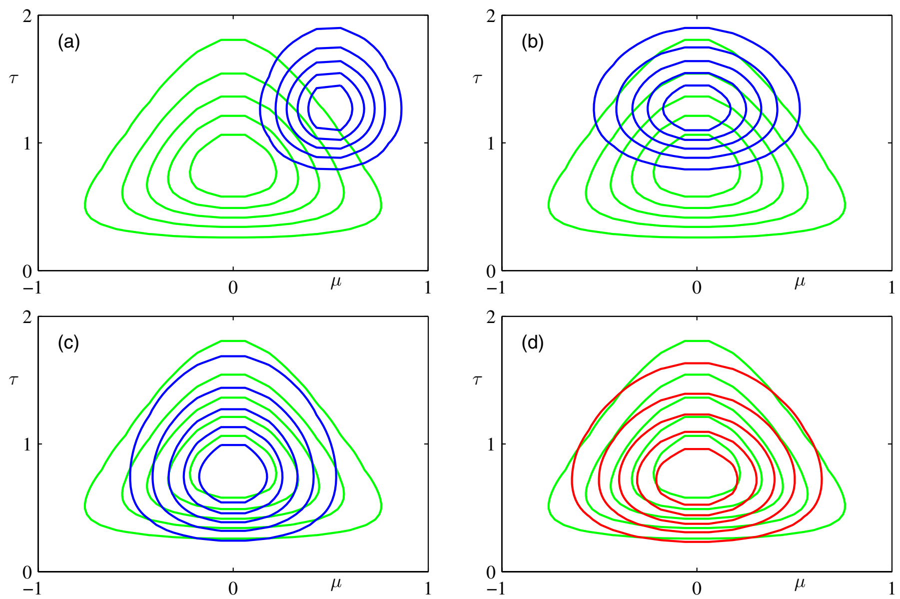
$$q_{MF}(z|x) = \prod_k q(z_k)$$

Variational Optimisation

$$\mathcal{F}(\mathbf{x}, q) = \underbrace{\boxed{\mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})]}}_{\text{Reconstruction}} - \underbrace{\boxed{KL[q(\mathbf{z})\|p(\mathbf{z})]}}_{\text{Penalty}}$$

Approx. Posterior

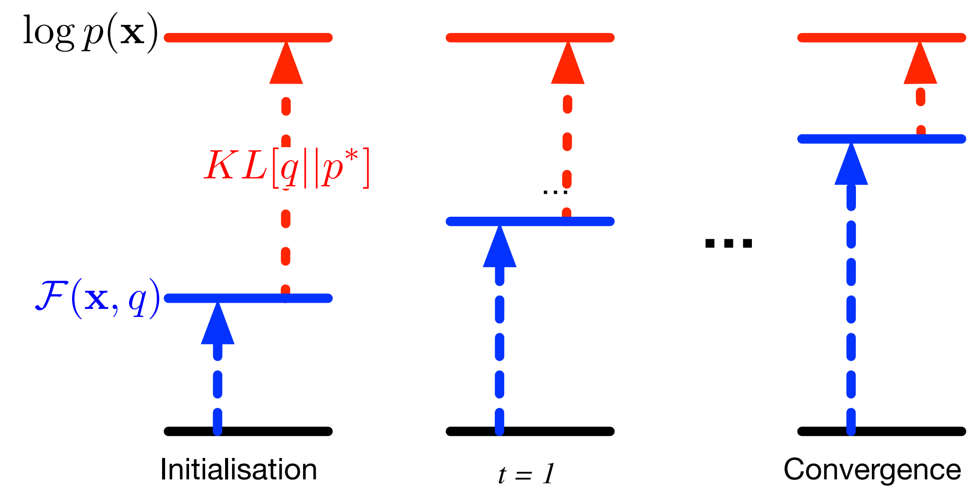
- Variational EM
- Stochastic Variational Inference
- Doubly Stochastic Variational Inference
- Amortised Inference



Variational EM

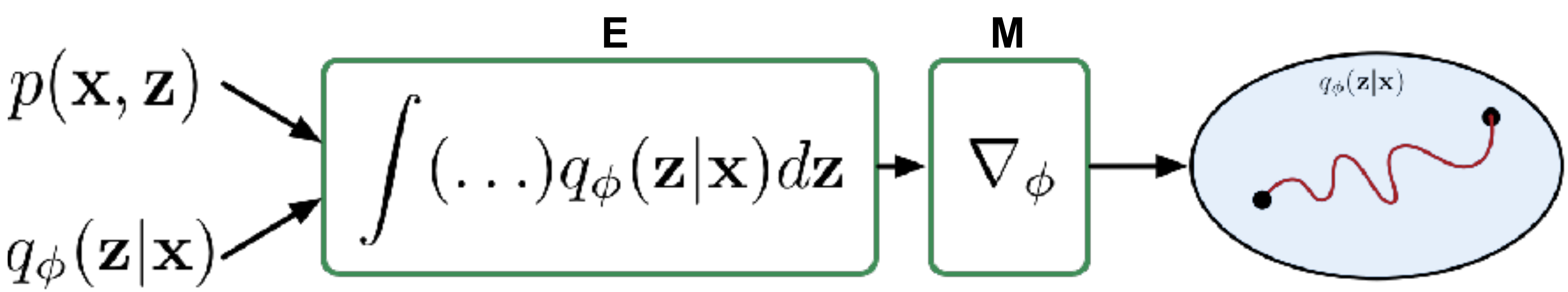
$$\mathcal{F}(\mathbf{x}, q) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})] - KL[q(\mathbf{z})\|p(\mathbf{z})]$$

Alternating optimisation for the variational parameters and then model parameters (VEM).



Repeat:

E-step	$\phi \propto \nabla_\phi \mathcal{F}(\mathbf{x}, q)$	*Var. params*
M-step	$\theta \propto \nabla_\theta \mathcal{F}(\mathbf{x}, q)$	*Model params*



Amortised Inference

Repeat:

E-step (compute q)

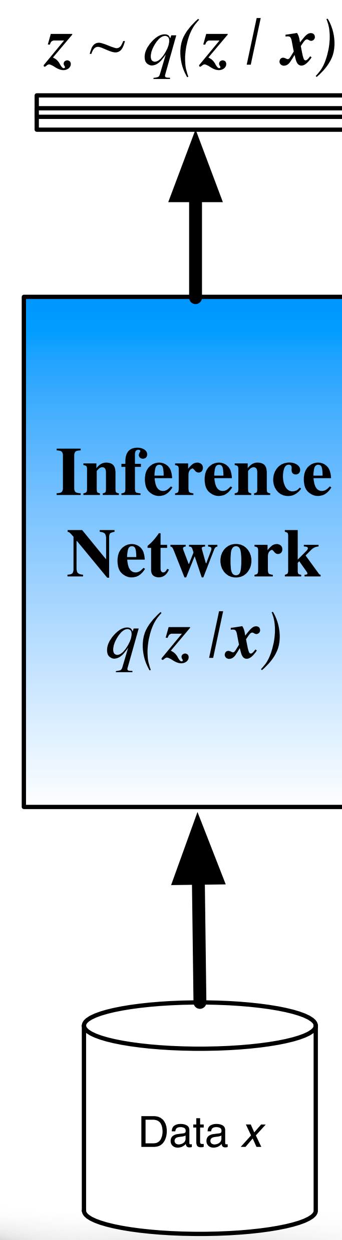
For i = 1, ... N

$$\phi_n \propto \nabla_\phi \mathbb{E}_{q_\phi(z)}[\log p_\theta(\mathbf{x}_n|z_n)] - \nabla_\phi KL[q(z_n)\|p(z)]$$

Instead of solving for every observation, amortise using a model.

M-step

$$\theta \propto \frac{1}{N}\sum_n \mathbb{E}_{q_\phi(z)}[\nabla_\theta \log p_\theta(\mathbf{x}_n|z_n)]$$

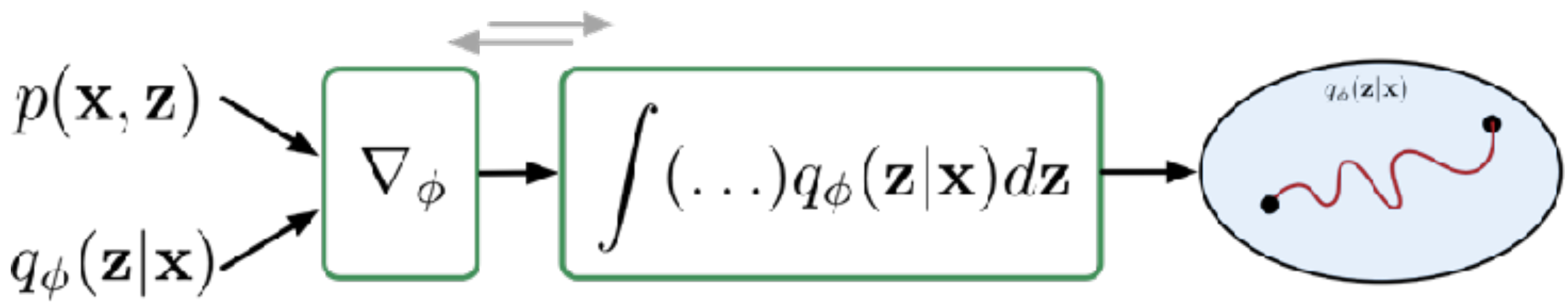


- **Inference network**: q is an ***encoder,*** an ***inverse*** model, ***recognition model***.
- Parameters of q are now a set of *global parameters* used for inference of all data points - test and train.
- **Amortise (spread) the cost of inference over all data.**
- Joint optimisation of variational and model parameters.

Inference networks provide an efficient mechanism for **posterior inference with memory**

Stochastic Gradients

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}[f_\theta(\mathbf{z})] = \nabla \int \boxed{q_\phi(\mathbf{z})}\boxed{f_\theta(\mathbf{z})} d\mathbf{z}$$



Doubly stochastic estimators

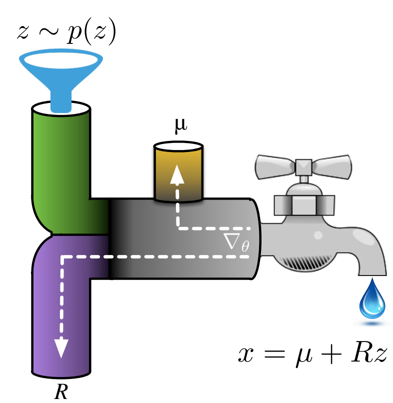
Pathwise Estimator

When easy to use transformation is available and differentiable function *f*.

$$= \mathbb{E}_{p(\epsilon)}[\nabla_\phi f_\theta(g(\epsilon, \phi))]$$

$$z \sim q_\phi(\mathbf{z})$$

$$\mathbf{z} = g(\epsilon, \phi) \quad \epsilon \sim p(\epsilon)$$



Reparameterisation

Score-function estimator

When function *f* non-differentiable and *q(z)* is easy to sample from.

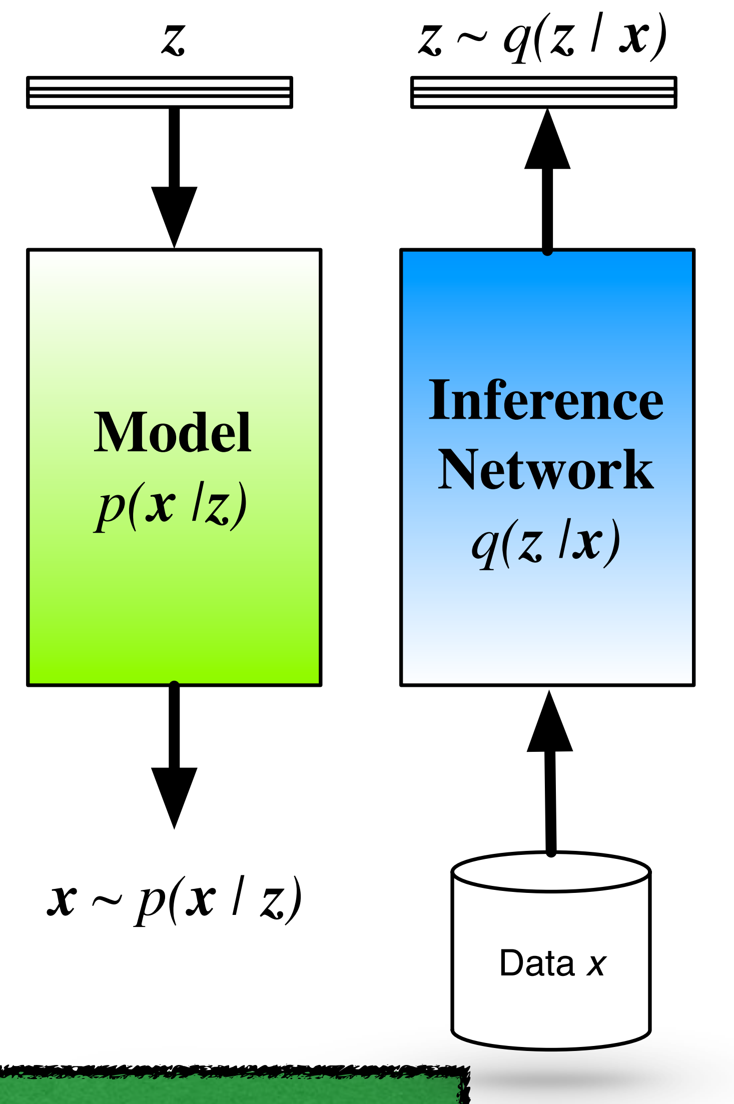
$$= \mathbb{E}_{q(z)}[f_\theta(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z})]$$

Identity **Log-derivative**

Variational Autoencoder

$$\mathcal{F}(\mathbf{x}, q) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})] - KL[q(\mathbf{z})\|p(\mathbf{z})]$$

Approx. Posterior Reconstruction Penalty



Stochastic encoder-decoder system to implement variational inference.

- **Model (Decoder):** likelihood p(**x**|**z**).
- **Inference (Encoder):** variational distribution q(**z**|**x**)
- Transforms an auto-encoder into a generative model

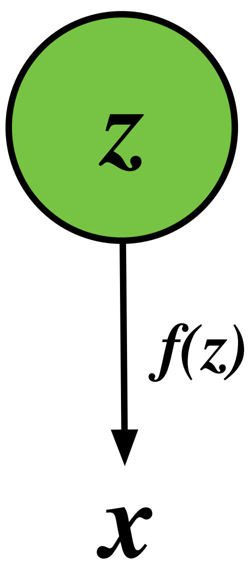
Specific combination of **variational inference** in **latent variable models** using **inference networks**
Variational Auto-encoder

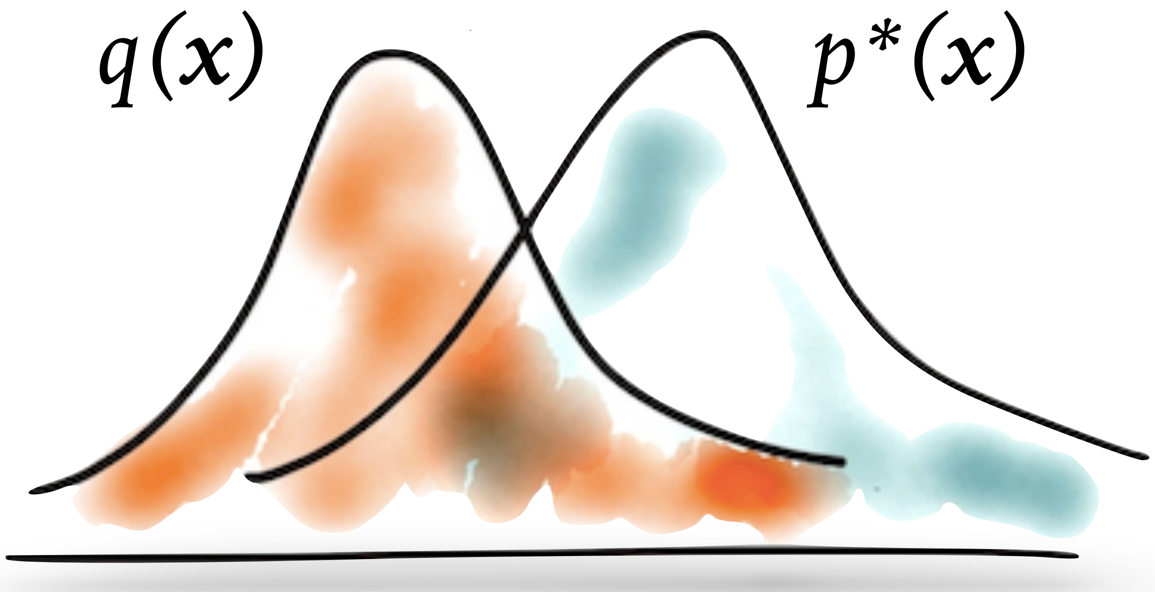
But don't forget what your model is, and what inference you use.

Learning by Comparison

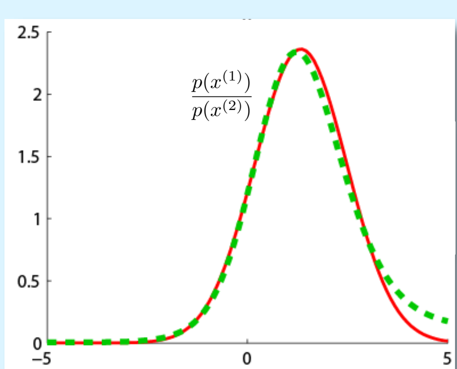
For some models, we only have access to an unnormalised probability or partial knowledge of the distribution.

Basic idea:
Transform into learning a model of the density ratio.





We compare the estimated distribution q(x) to the true distribution p*(x)



Learning principle: Two-sample tests

$$\frac{p^*(\mathbf{x})}{q(\mathbf{x})} = 1 \qquad p^*(\mathbf{x}) = q(\mathbf{x})$$

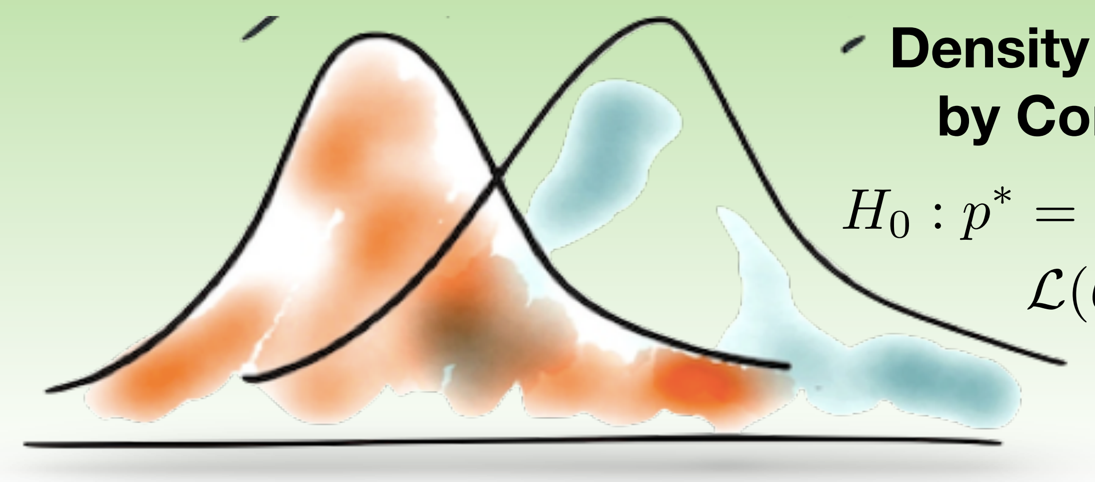
Interest is not in estimating the marginal probabilities, only in how they are related.

Estimation by Comparison

Two steps

1. Use a hypothesis **test or comparison** to obtain some model to tells how data from our model differs from observed data.

2. **Adjust model** to better match the data distribution using the comparison model from step 1.



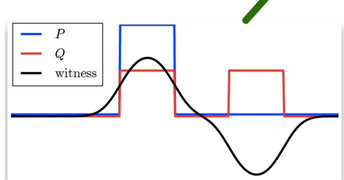
Density Estimation by Comparison

$$H_0 : p^* = q_\theta \text{ vs. } p^* \neq q_\theta$$

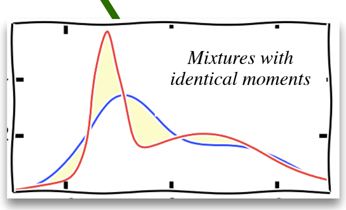
$$\mathcal{L}(\theta, \phi)$$

Density Difference

$$r_\phi = p^* - q_\theta$$



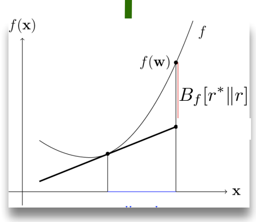
Max Mean Discrepancy



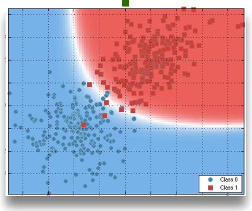
Moment Matching

Density Ratio

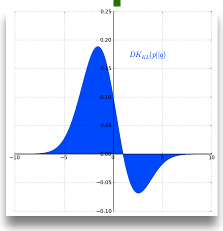
$$r_\phi = \frac{p^*}{q_\theta}$$



Bregman Divergence



Class Probability Estimation



f-Divergence

$$f(u) = u \log u - (u+1)\log(u+1)$$

Adversarial Learning

$$\frac{p^*(\mathbf{x})}{q(\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}$$

$$p(y=+1|\mathbf{x}) = D_\theta(\mathbf{x}) \quad p(y=-1|\mathbf{x}) = 1 - D_\theta(\mathbf{x})$$

Scoring Function

$$\mathcal{F}(\mathbf{x}, \theta, \phi) = \mathbb{E}_{p^*(x)}[\log D_\theta(\mathbf{x})] + \mathbb{E}_{q_\phi(x)}[\log(1 - D_\theta(\mathbf{x})]$$

Bernoulli Loss

Generative Adversarial Networks

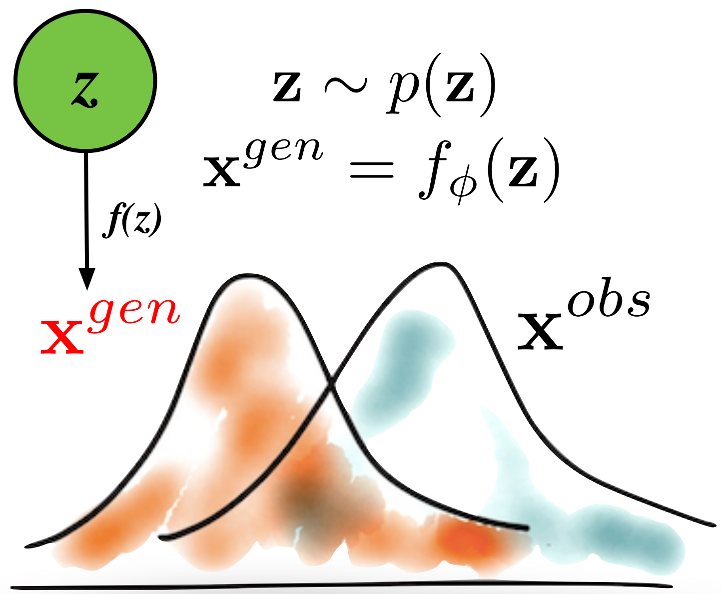
Alternating optimisation $\min_\phi \max_\theta \mathcal{F}(\mathbf{x}, \theta, \phi)$

Comparison loss $\theta \propto \nabla_\theta \mathbb{E}_{p^*(x)}[\log D_\theta(\mathbf{x})] + \nabla_\theta \mathbb{E}_{q_\phi(x)}[\log(1 - D_\theta(\mathbf{x})]$

Generative loss $\phi \propto -\nabla_\phi \mathbb{E}_{q(z)}[\log(1 - D_\theta(f_\phi(\mathbf{z}))]$

Instances of testing and inference:

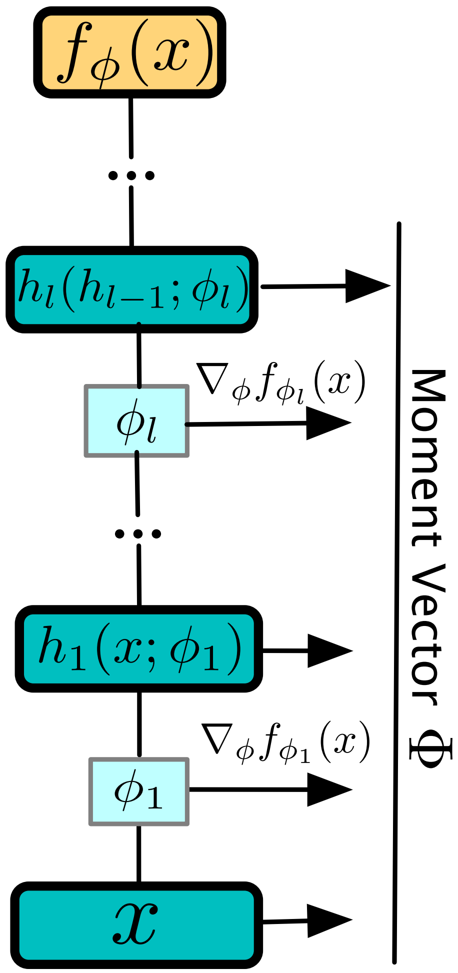
- Unsupervised-as-supervised learning
- Classifier ABC
- Noise-contrastive estimation
- Adversarial learning and GANs



Density-ratio

Reparameterisation

Method of Moments



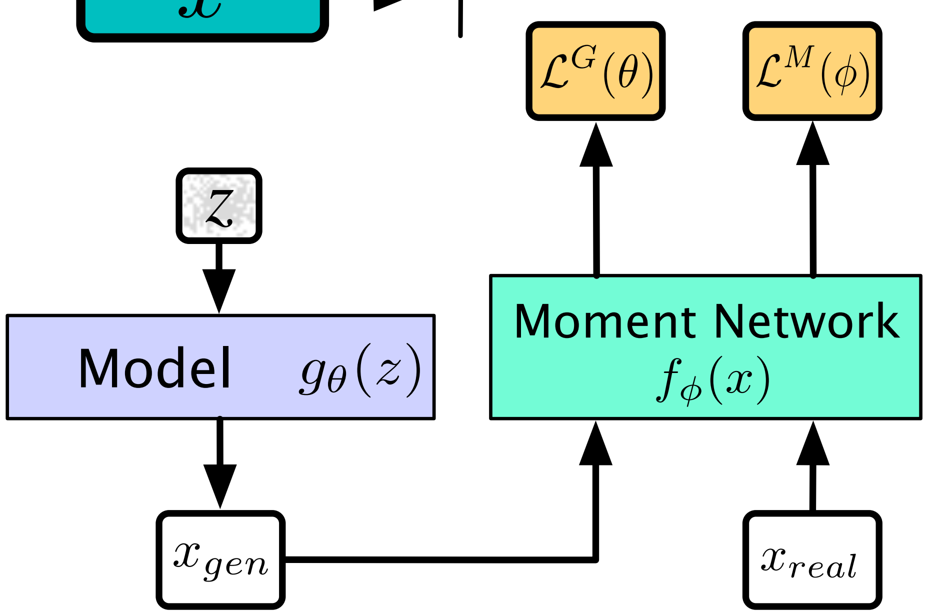
Moment estimator

$$m(\theta) = \mathbb{E}_{p_\theta(s)}[f(s)]$$

Tangent of posterior odds.

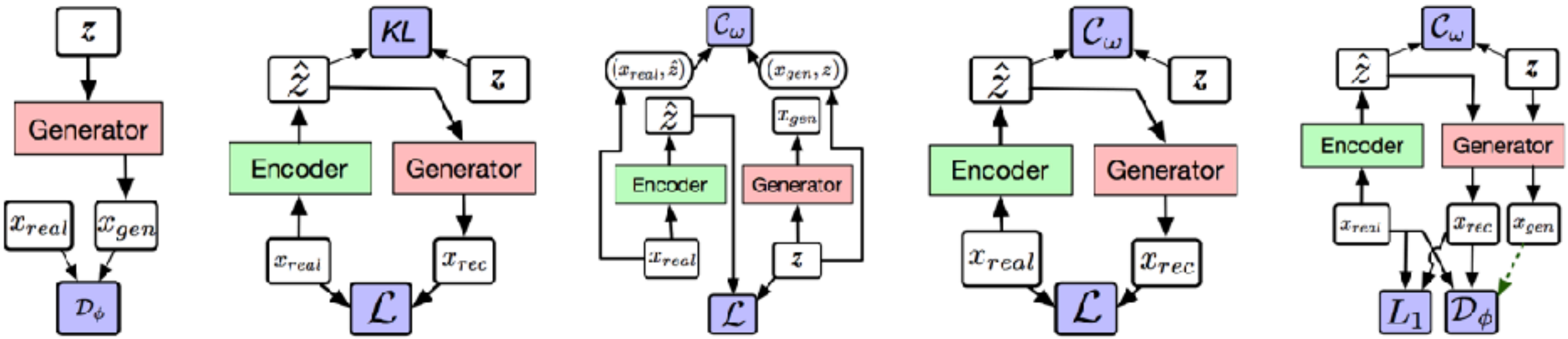
$$\nabla_\phi \log \frac{p(y=+1|x)}{p(y=-1|x)} = \nabla_\phi f_\phi(x)$$

$$\mathcal{F} = \left\|\mathbb{E}_{p^*(x)}[f(x)] - \mathbb{E}_{p(z)}[f(g_\phi(z))]\right\|_2^2$$



- Consistent estimators: the number of moments is greater than the number of model parameters.
- Features should not be not co-linear.
- More stable than adversarial training.
- Does not require frequent updating of the classifier.

Convergent Approaches



(a) DCGAN (b) VAE (c) VEEGAN (d) AAE (e) AAE++/AAE-GAN

$$\mathcal{F}(q, \boldsymbol{\theta}) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x}|)\|p(\mathbf{z})]$$

$$\mathbb{E}_{p^*(\mathbf{x})} KL[q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})] = KL[q(\mathbf{z})\|p(\mathbf{z})] + \mathbb{I}[q(\mathbf{z}|\mathbf{x}), p^*(\mathbf{x})]$$

Marginal KL

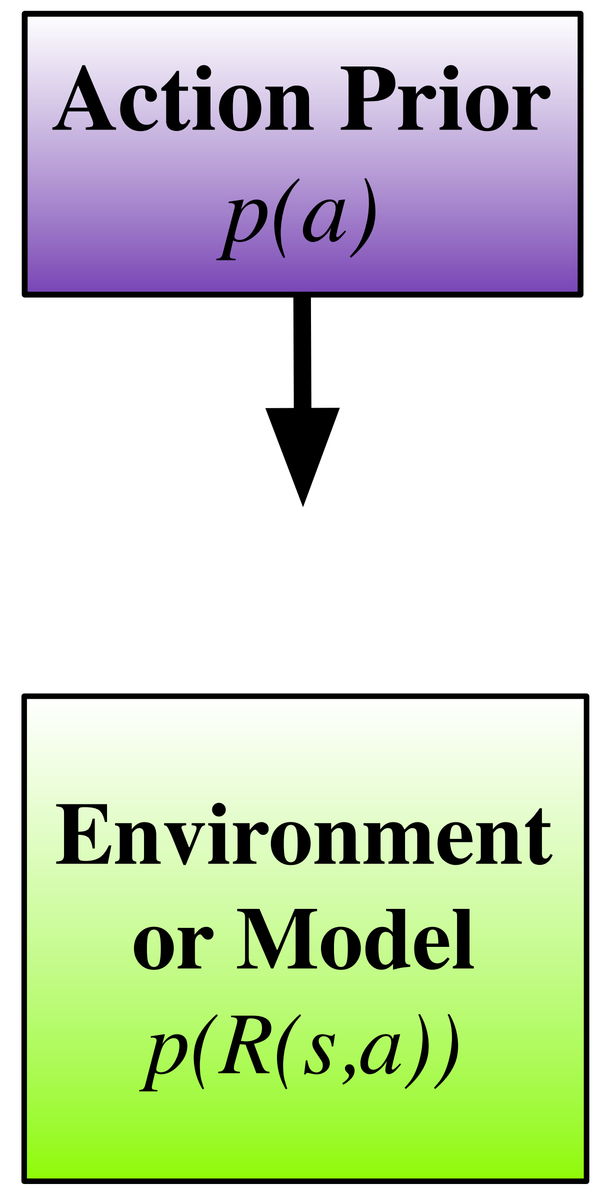
Information

- Replace density ratios by classifiers, replace posteriors with implicit models,
- Views from optimal transport and connections to integral probability metrics.

Environments and Rewards

Environment as a generative process:

- An unknown likelihood;
- Not known analytically;
- Only able to observe its outcomes.



$$a \sim p(a)$$ **Prior over actions**

$$u(s, a) \sim \text{Environment}(a)$$ **Interaction only**

$$p(R(s)|a) \propto \exp(u(s, a))$$ **Long-term reward**

All the key inferential questions can now be asked in this simple framework.

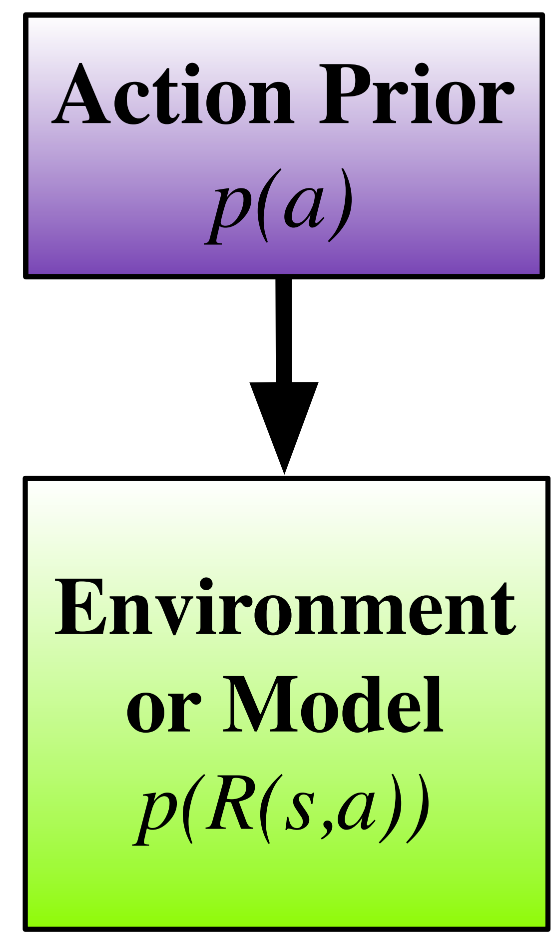
Planning-as-Inference

Simplest question

What is the posterior distribution over actions?
Maximising the probability of the return *log p(R)*.

Variational inference in the hierarchical model

$$\mathcal{F}(\theta) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}[R(s,a)] - \mathrm{KL}[\pi_\theta(\mathbf{a}|s)\|p(\mathbf{a})]$$



Recover policy search methods:

- Uniform prior over distributions
- Continuous policy parameters
- Can evaluate environment, but not differentiate.

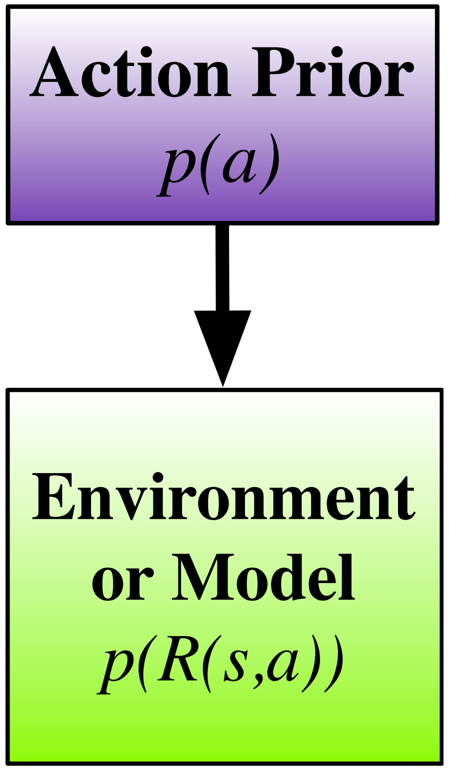
Policy Search

Free Energy $\mathcal{F}(\theta) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}[R(s,a)] - \mathrm{KL}[\pi_\theta(\mathbf{a}|s)\|p(\mathbf{a})]$

Policy gradient using score-function gradient

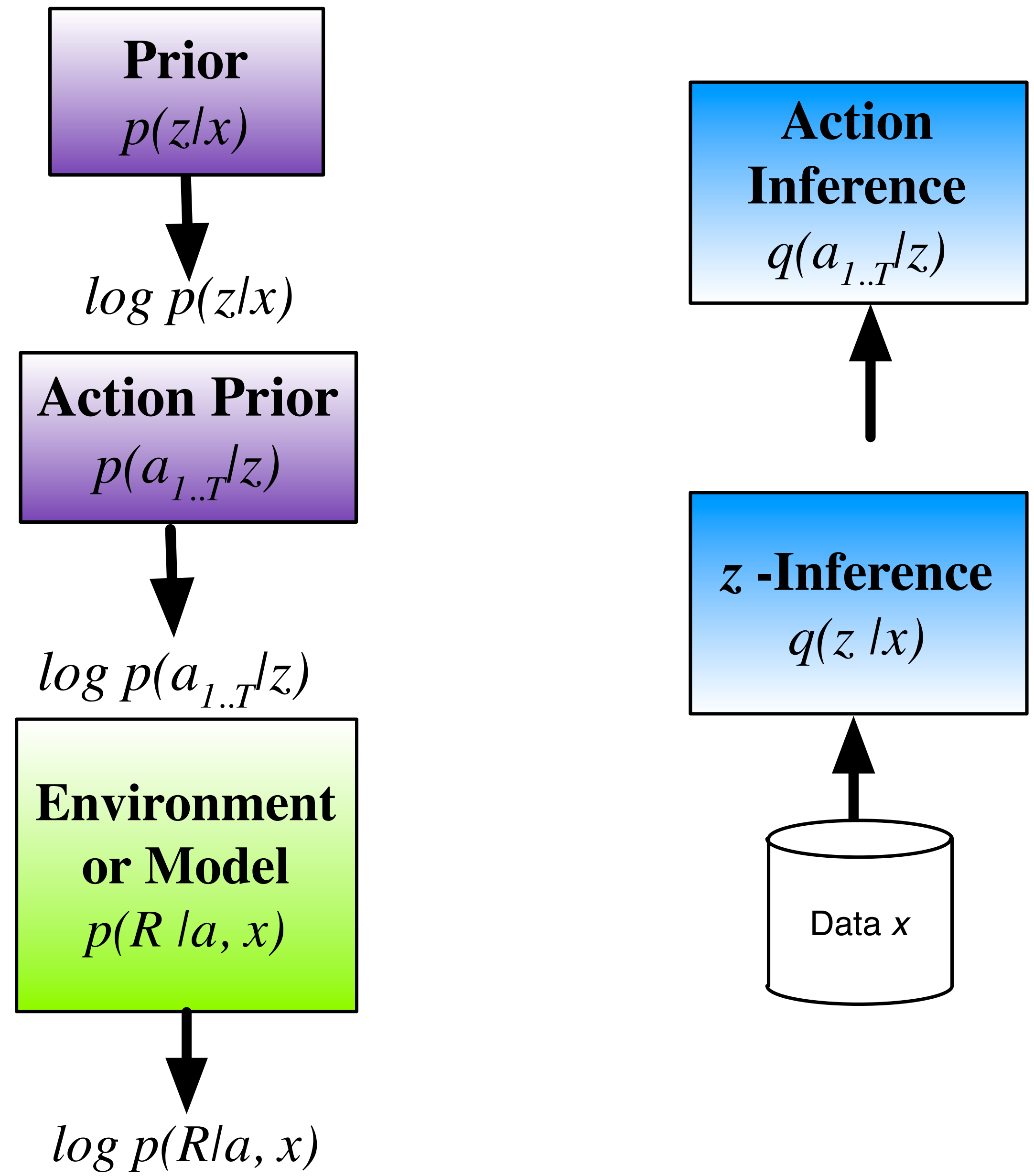
$$\nabla_\theta \mathcal{F}(\theta) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}[(R(s,a) - c)\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s})] + \nabla_\theta \mathbb{H}[\pi_\theta(\mathbf{a}|s)]$$

- Appearance of the entropy penalty is natural and alternative priors easy to consider.
- Can easily incorporate prior knowledge of the action space.
- Use any of the tools of probabilistic inference available.
- Easily handle stochastic and deterministic policies.



Identity | Log-derivative

Hierarchical Planning

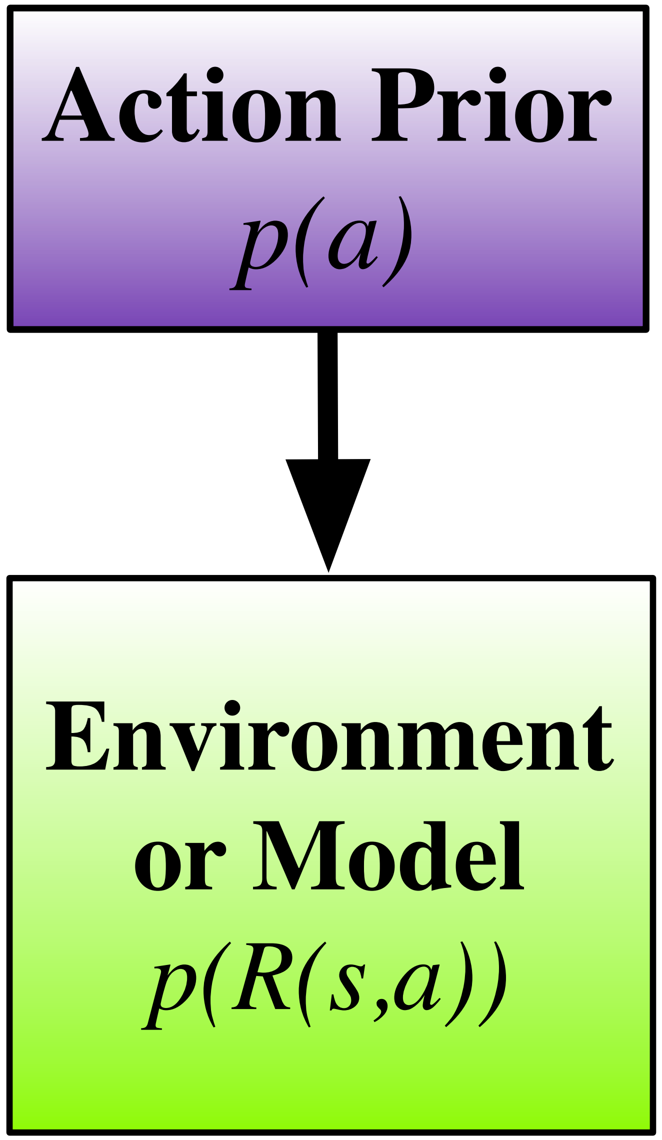


Variational MDP

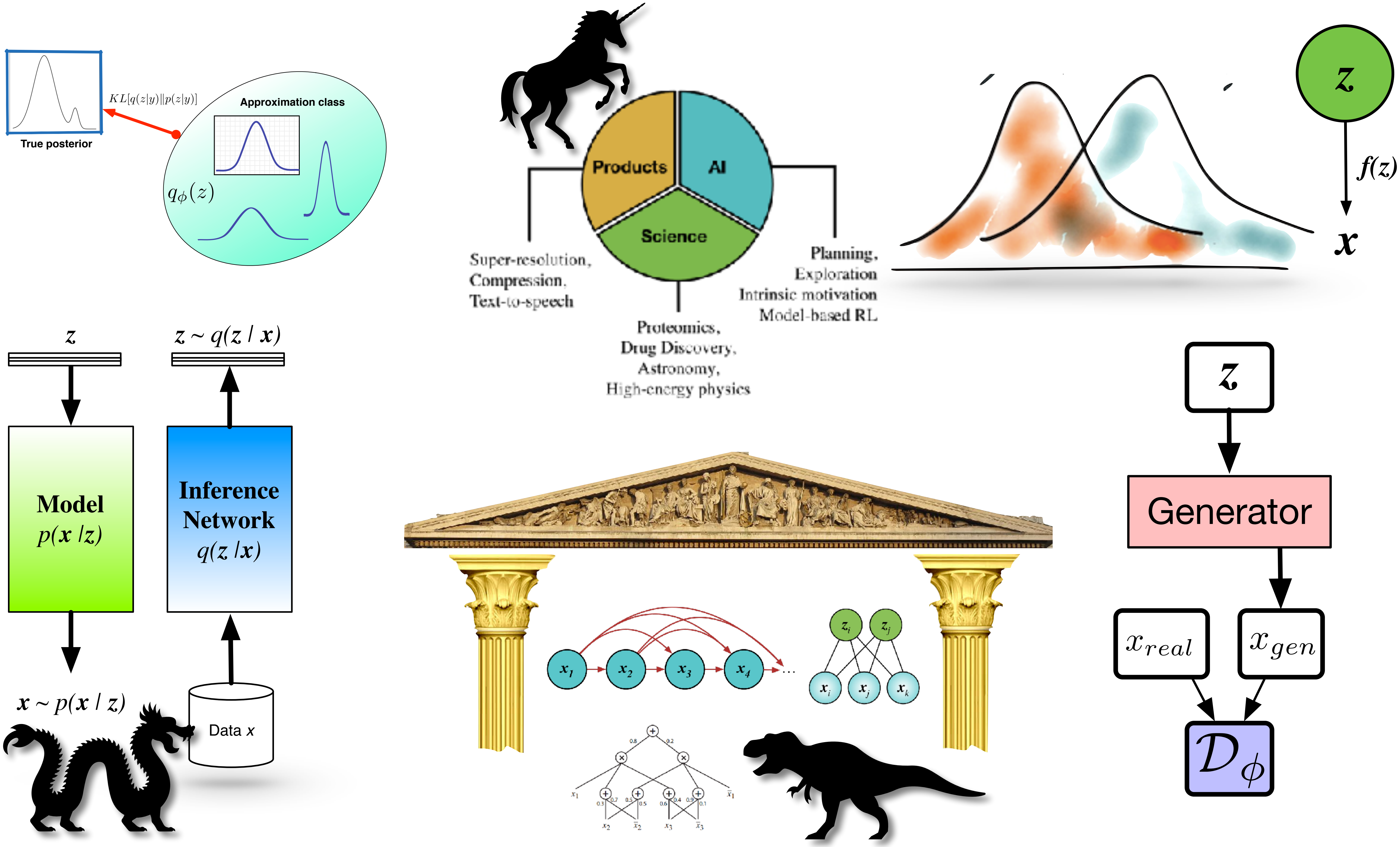
$$\mathcal{F}^{\pi}(\theta) = \mathbb{E}_{q(a,z|x)}[R(a|x)] - \alpha KL[q_{\theta}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}|\mathbf{x})] + \alpha\mathbb{H}[\pi_{\theta}(\mathbf{a}|\mathbf{z})]$$

With a more realistic expansion as graphical model

- Derive Bellman's equation as a different writing of message passing.
- Application of the EM algorithm for policy search becomes possible.
- Easily consider other variational methods, like EP.
- Both model-free and model-based methods emerge.



Final Words



True posterior
$KL[q(z|y)||p(z|y)]$
Approximation class
$q_\phi(z)$
Products
AI
Science
Super-resolution,
Compression,
Text-to-speech
Planning,
Exploration
Intrinsic motivation
Model-based RL
Proteomics,
Drug Discovery,
Astronomy,
High-energy physics
z
f(z)
x
z
z ~ q(z | x)
Model
p(x |z)
Inference
Network
q(z |x)
x ~ p(x | z)
Data x
x_1
x_2
x_3
x_4
z_i
z_j
x_i
x_j
x_k
z
Generator
x_{real}
x_{gen}
$\mathcal{D}_\phi$

Build Communities

DEEP LEARNING INDABA


shakirm.com/feedback
Planting the Seeds of Probabilistic Thinking
Foundations | Tricks | Algorithms
Shakir Mohamed
Research Scientist, DeepMind
@shakir_za
DeepMind

Not exhaustive list, and many references to be updated.

Probabilistic Thinking

Cheeseman, P.C., 1985, August. In Defense of Probability. In *IJCAI* (Vol. 2, pp. 1002-1009).

Murphy, K.P., 2012. Machine Learning: A Probabilistic Perspective.

Efron, B., 1982. Maximum likelihood and decision theory. *The annals of Statistics*, pp.340-356.

Stochastic Optimisation

P L'Ecuyer, Note: On the interchange of derivative and expectation for likelihood ratio derivative estimators, Management Science, 1995

Peter W Glynn, Likelihood ratio gradient estimation for stochastic systems, Communications of the ACM, 1990

Michael C Fu, Gradient estimation, Handbooks in operations research and management science, 2006

Ronald J Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning, 1992

Paul Glasserman, Monte Carlo methods in financial engineering, , 2003

Luc Devroye, Random variate generation in one line of code, Proceedings of the 28th conference on Winter simulation, 1996

L. Devroye, Non-uniform random variate generation, , 1986

Omiros Papaspiliopoulos, Gareth O Roberts, Martin Skold, A general framework for the parametrization of hierarchical models, Statistical Science, 2007

Michael C Fu, Gradient estimation, Handbooks in operations research and management science, 2006

Ranganath, Rajesh, Sean Gerrish, and David M. Blei. "Black Box Variational Inference." In AISTATS, pp. 814-822. 2014.

Mnih, Andriy, and Karol Gregor. "Neural variational inference and learning in belief networks." arXiv preprint arXiv:1402.0030 (2014).

Lázaro-Gredilla, Miguel. "Doubly stochastic variational Bayes for non-conjugate inference." (2014).

Wingate, David, and Theophane Weber. "Automated variational inference in probabilistic programming." arXiv preprint arXiv:1301.1299 (2013).

Paisley, John, David Blei, and Michael Jordan. "Variational Bayesian inference with stochastic search." arXiv preprint arXiv:1206.6430 (2012).

- Applications of Deep Generative Models
- Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models." ICML 2014
- Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." ICLR 2014
- Gregor, Karol, et al. "Towards Conceptual Compression." arXiv preprint arXiv:1604.08772 (2016).
- Eslami, S. M., Heess, N., Weber, T., Tassa, Y., Kavukcuoglu, K., & Hinton, G. E. (2016). Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. arXiv preprint arXiv:1603.08575.
- Oh, Junhyuk, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder Singh. "Action-conditional video prediction using deep networks in atari games." In Advances in Neural Information Processing Systems, pp. 2863-2871. 2015.
- Rezende, Danilo Jimenez, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. "One-Shot Generalization in Deep Generative Models." arXiv preprint arXiv:1603.05106 (2016).
- Rezende, Danilo Jimenez, S. M. Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. "Unsupervised Learning of 3D Structure from Images." arXiv preprint arXiv:1607.00662 (2016).
- Kingma, Diederik P., Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. "Semi-supervised learning with deep generative models." In Advances in Neural Information Processing Systems, pp. 3581-3589. 2014.
- Maaløe, Lars, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. "Auxiliary Deep Generative Models." arXiv preprint arXiv:1602.05473 (2016).
- Odena, Augustus. "Semi-Supervised Learning with Generative Adversarial Networks." arXiv preprint arXiv:1606.01583 (2016).
- Springenberg, Jost Tobias. "Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks." arXiv preprint arXiv:1511.06390 (2015).
- Blundell, Charles, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z. Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. "Model-Free Episodic Control." arXiv preprint arXiv:1606.04460 (2016).
- Higgins, Irina, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. "Early Visual Concept Learning with Unsupervised Deep Learning." arXiv preprint arXiv:1606.05579 (2016).
- Bellemare, Marc G., Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. "Unifying Count-Based Exploration and Intrinsic Motivation." arXiv preprint arXiv:1606.01868 (2016).
- Alexander (Sasha) Vezhnevets, Mnih, Volodymyr, John Agapiou, Simon Osindero, Alex Graves, Oriol Vinyals, and Koray Kavukcuoglu. "Strategic Attentive Writer for Learning Macro-Actions." arXiv preprint arXiv:1606.04695 (2016).
- Gregor, Karol, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. "DRAW: A recurrent neural network for image generation." arXiv preprint arXiv:1502.04623 (2015).

- **Fully-observed Models**
- Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).
- Larochelle, Hugo, and Iain Murray. "The Neural Autoregressive Distribution Estimator." In AISTATS, vol. 1, p. 2. 2011.
- Uria, Benigno, Iain Murray, and Hugo Larochelle. "A Deep and Tractable Density Estimator." In ICML, pp. 467-475. 2014.
- Veness, Joel, Kee Siong Ng, Marcus Hutter, and Michael Bowling. "Context tree switching." In 2012 Data Compression Conference, pp. 327-336. IEEE, 2012.
- Rue, Havard, and Leonhard Held. Gaussian Markov random fields: theory and applications. CRC Press, 2005.
- Wainwright, Martin J., and Michael I. Jordan. "Graphical models, exponential families, and variational inference." Foundations and Trends® in Machine Learning 1, no. 1-2 (2008): 1-305.

- **Implicit Probabilistic Models**
- Tabak, E. G., and Cristina V. Turner. "A family of nonparametric density estimation algorithms." Communications on Pure and Applied Mathematics 66, no. 2 (2013): 145-164.
- Rezende, Danilo Jimenez, and Shakir Mohamed. "Variational inference with normalizing flows." arXiv preprint arXiv:1505.05770 (2015).
- Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In Advances in Neural Information Processing Systems, pp. 2672-2680. 2014.
- Verrelst, Herman, Johan Suykens, Joos Vandewalle, and Bart De Moor. "Bayesian learning and the Fokker-Planck machine." In Proceedings of the International Workshop on Advanced Black-box Techniques for Nonlinear Modeling, Leuven, Belgium, pp. 55-61. 1998.
- Devroye, Luc. "Random variate generation in one line of code." In Proceedings of the 28th conference on Winter simulation, pp. 265-272. IEEE Computer Society, 1996.
- Ravuri, S., Mohamed, S., Rosca, M. and Vinyals, O., 2018. Learning Implicit Generative Models with the Method of Learned Moments. *arXiv preprint arXiv:1806.11006*.

Latent variable models

Dayan, Peter, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. "The helmholtz machine." Neural computation 7, no. 5 (1995): 889-904.

Hyvärinen, A., Karhunen, J., & Oja, E. (2004). Independent component analysis (Vol. 46). John Wiley & Sons.

Gregor, Karol, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. "Deep autoregressive networks." arXiv preprint arXiv:1310.8499 (2013).

Ghahramani, Zoubin, and Thomas L. Griffiths. "Infinite latent feature models and the Indian buffet process." In Advances in neural information processing systems, pp. 475-482. 2005.

Teh, Yee Whye, Michael I. Jordan, Matthew J. Beal, and David M. Blei. "Hierarchical dirichlet processes." Journal of the american statistical association (2012).

Adams, Ryan Prescott, Hanna M. Wallach, and Zoubin Ghahramani. "Learning the Structure of Deep Sparse Graphical Models." In AISTATS, pp. 1-8. 2010.

Lawrence, Neil D. "Gaussian process latent variable models for visualisation of high dimensional data." Advances in neural information processing systems 16.3 (2004): 329-336.

Damianou, Andreas C., and Neil D. Lawrence. "Deep Gaussian Processes." In AISTATS, pp. 207-215. 2013.

Mattos, César Lincoln C., Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A. Barreto, and Neil D. Lawrence. "Recurrent Gaussian Processes." arXiv preprint arXiv:1511.06644 (2015).

Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." In Proceedings of the 24th international conference on Machine learning, pp. 791-798. ACM, 2007.

Saul, Lawrence K., Tommi Jaakkola, and Michael I. Jordan. "Mean field theory for sigmoid belief networks." Journal of artificial intelligence research 4, no. 1 (1996): 61-76.

Frey, Brendan J., and Geoffrey E. Hinton. "Variational learning in nonlinear Gaussian belief networks." Neural Computation 11, no. 1 (1999): 193-213.

Inference and Learning

Jordan, Michael I., Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. "An introduction to variational methods for graphical models." Machine learning 37, no. 2 (1999): 183-233.

Hoffman, Matthew D., David M. Blei, Chong Wang, and John William Paisley. "Stochastic variational inference." Journal of Machine Learning Research 14, no. 1 (2013): 1303-1347.

Honkela, Antti, and Harri Valpola. "Variational learning and bits-back coding: an information-theoretic view to Bayesian learning." IEEE Transactions on Neural Networks 15, no. 4 (2004): 800-810.

Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov. "Importance weighted autoencoders." arXiv preprint arXiv:1509.00519 (2015).

Li, Yingzhen, and Richard E. Turner. "Variational Inference with R\'enyi Divergence." arXiv preprint arXiv:1602.02311 (2016).

Borgwardt, Karsten M., and Zoubin Ghahramani. "Bayesian two-sample tests." arXiv preprint arXiv:0906.4032 (2009).

Gutmann, Michael, and Aapo Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." AISTATS. Vol. 1. No. 2. 2010.

Tsuboi, Yuta, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama. "Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation." Information and Media Technologies 4, no. 2 (2009): 529-546.

Sugiyama, Masashi, Taiji Suzuki, and Takafumi Kanamori. Density ratio estimation in machine learning. Cambridge University Press, 2012.

Amortised Inference

Gershman, Samuel J., and Noah D. Goodman. "Amortized inference in probabilistic reasoning." In Proceedings of the 36th Annual Conference of the Cognitive Science Society. 2014.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models." arXiv preprint arXiv:1401.4082 (2014).

Heess, Nicolas, Daniel Tarlow, and John Winn. "Learning to pass expectation propagation messages." In Advances in Neural Information Processing Systems, pp. 3219-3227. 2013.

Jitkrittum, Wittawat, Arthur Gretton, Nicolas Heess, S. M. Eslami, Balaji Lakshminarayanan, Dino Sejdinovic, and Zoltán Szabó. "Kernel-based just-in-time learning for passing expectation propagation messages." arXiv preprint arXiv:1503.02551 (2015).

Korattikara, Anoop, Vivek Rathod, Kevin Murphy, and Max Welling. "Bayesian dark knowledge." arXiv preprint arXiv:1506.04416 (2015).