Planting the Seeds of Probabilistic Thinking

Foundations | Tricks | Algorithms

Shakir Mohamed
Research Scientist, DeepMind

@shakir_za
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.
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.
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**

\[ p(x) \quad p^*(x) \quad q(x) \]

**Conditions**

\[ p(x) \geq 0 \quad \int p(x) \, dx = 1 \]

**Bayes Rule**

\[ p(z|x) = \frac{p(x|z)p(z)}{p(x)} \]

**Parameterisation**

\[ p_\theta(x|z) \equiv p(x|z; \theta) \]

**Expectation**

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

**Gradient**

\[ \nabla_\phi f(x; \phi) = \frac{\partial f(x; \phi)}{\partial \phi} \]
Statistical Operations

- Estimation and Learning
- Modelling
- Inference
- Data Enumeration
- Comparison
- Experimental Design
- Hypothesis Testing
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

The core questions of AGI will be those of probabilistic 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

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

\[ p(y|x) = p(y|g(\eta); \theta) \]

- 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.

Optimise the negative log-likelihood

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

<table>
<thead>
<tr>
<th>Target</th>
<th>Regression</th>
<th>Link</th>
<th>Inv link</th>
<th>Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Linear</td>
<td>Identity</td>
<td>Identity</td>
<td>Sigmoid</td>
</tr>
<tr>
<td>Binary</td>
<td>Logistic</td>
<td>Logit ( \log \frac{1}{1+\mu} )</td>
<td>Sigmoid</td>
<td>Sigmoid</td>
</tr>
<tr>
<td>Binary</td>
<td>Probit</td>
<td>Inv Gauss CDF ( \Phi^{-1}(\mu) )</td>
<td>Gauss CDF</td>
<td>Probit</td>
</tr>
<tr>
<td>Binary</td>
<td>Gumbel</td>
<td>Compl. log-log ( \log(-\log(\mu)) )</td>
<td>Gumbel CDF</td>
<td>( e^{-e^{-x}} )</td>
</tr>
<tr>
<td>Binary</td>
<td>Logistic</td>
<td></td>
<td>Hyperbolic Tangent ( tanh(\eta) )</td>
<td>Tanh</td>
</tr>
<tr>
<td>Categorical</td>
<td>Multinomial</td>
<td></td>
<td>Multin. Logit ( \sum_i \eta_i )</td>
<td>Softmax</td>
</tr>
<tr>
<td>Counts</td>
<td>Poisson</td>
<td>log(µ) ( \sqrt{\mu} )</td>
<td>exp(\nu)</td>
<td>( \frac{1}{\nu} )</td>
</tr>
<tr>
<td>Counts</td>
<td>Poisson</td>
<td>( \sqrt{\mu} )</td>
<td>( \nu^2 )</td>
<td>( \frac{1}{\nu} )</td>
</tr>
<tr>
<td>Non-neg.</td>
<td>Gamma</td>
<td>Reciprocal ( \frac{1}{\eta} )</td>
<td>max(max(0; \nu))</td>
<td>ReLU</td>
</tr>
<tr>
<td>Sparse</td>
<td>Tobit</td>
<td></td>
<td>Cum. Logit ( \sigma(\phi_k - \eta) )</td>
<td>DeepMind</td>
</tr>
<tr>
<td>Ordered</td>
<td>Ordinal</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Recursive Generalised Linear Regression

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

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

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

**Probabilistic Model**

\[ p(y|x) = p(y|h(x); \theta) \]

**Likelihood function**

\[ L(\theta) = \sum_n \log p(y_n|x_n; \theta) \]

**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.

**Prescribed Likelihoods**

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

**Probabilistic Model**

\[ p(y|x) = p(y|h(x); \theta) \]

**Likelihood function**

\[ \mathcal{L}(\theta) = \sum_n \log p(y_n|x_n; \theta) \]

**Maximum Likelihood**

\[ \arg \max_\theta \mathcal{L}(\theta) \]

- 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.

---

\[ \mathbb{E}[y] \]

\[ g() \]

\[ \eta_l = Bx_l \]

\[ \ldots \]

\[ g() \]

\[ \eta_1 = Bx_1 \]
Estimation Theory

**Probabilistic Model**

\[ p(\theta | y, x) \propto p(y | h(x); \theta)p(\theta) \]

**Likelihood function**

\[ \mathcal{L}(\theta) = \sum_n \log p(y_n | x_n; \theta) + \frac{1}{\lambda} \mathcal{R}(\theta) \]

- **Maximum a Posteriori (MAP)**

\[ \text{arg max}_\theta \mathcal{L}(\theta) \]

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

\[ \mathcal{L}(\theta) = \sum_n \log p(y_n | x_n; \theta) + \frac{1}{\lambda} \mathcal{R}(\theta) \]

- **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

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

Transform

\[ \mu = \phi^2 \]

New prior

\[ p(\phi) = 2\phi \]

MAP Est.

\[ \hat{\phi}_{MAP} = 1 \]

Parameterisation 2

Transform

\[ \mu = 1 - (1 - \phi)^2 \]

New prior

\[ p(\phi) = 2(1 - \phi) \]

MAP Est.

\[ \hat{\phi}_{MAP} = 0 \]

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

Use a modified probabilistic model that removes sensitivity

\[ p(y|h(x); \theta)p(\theta)|\mathcal{I}(\theta)|^{\frac{1}{2}} \]

- 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.

\[ p(\theta | y, x) \propto p(y | h(x); \theta) p(\theta) \]

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

In Bayesian analysis, things that are *not* observed must be integrated over - averaged out.

This makes computation difficult.

Integration is the central operation.

**Evidence**

\[ p(y|x) = \int p(y|h(x); \theta)p(\theta)d\theta \]

**Posterior**

\[ p(\theta|y, x) \propto p(y|h(x); \theta)p(\theta) \]

- 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).

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

**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.

**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: $p(\theta) = \mathcal{N}(\theta|0, I)$
- Observation model: $p(y|x, \theta) = \text{Categorical}(\pi(x; \theta))$
- Posterior: $p(\theta|y, x)$

- 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

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

Factor Analysis / PCA

\[ z \sim \mathcal{N}(z|\mu, \Sigma) \]
\[ y \sim \mathcal{N}(y|Wz, \sigma_y^2 I) \]
Decision-making

Probabilistic models of environments and actions

Prior over actions
\[ a \sim p(a) \]

Interaction only
\[ u(s, a) \sim \text{Environment}(a) \]

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

\[ f(x) = w^T \phi(x; \theta); \quad \{w, \theta\} \sim \mathcal{N}(0, \sigma^2 I) \]

\[ y = f(x) + \epsilon; \quad \epsilon \sim \mathcal{N}(0, \sigma^2_y) \]

Move from primal variables to dual variables

\[ \mathcal{L}(f) = \frac{1}{2} \sum_n (y_n - f(x))^2 + \frac{\lambda}{2} \|f\|^2_H \]

Kernel trick and methods

Probability distributions over functions

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

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

\[ p(z) = p(z_1|z_2)p(z_2|z_3) \ldots p(z_{L-1}|z_L)p(z_L) \]
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
1. Computational Graphs

2. Error propagation
Models

Directed and Undirected

Fully-observed

Parametric, Non-parametric
And semi-parametric

Latent Variable
Learning Principles

Statistical Inference

Direct
- Laplace approximation
- Maximum a posteriori
- Cavity Methods
- Expectation Maximisation
- Noise Contrastive
- Maximum Likelihood
- Variational Inference
- Integr. Nested Laplace Approx
- Markov chain Monte Carlo
- Sequential Monte Carlo

Indirect
- Two Sample Comparison
- Approx Bayesian Computation
- Max Mean Discrepancy
- Method of Moments
- Transportation methods
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)

**Latent variable model + variational inference**
- VEM algorithm
- Expectation propagation
- Approximate message passing
- Variational auto-encoders (VAE)

**Implicit Generative Model + Two-sample testing**
- Unsupervised-as-supervised learning
- Approximate Bayesian Computation (ABC)
- Generative adversarial network (GAN)

**Restricted Boltzmann Machine + maximum likelihood**
- Contrastive Divergence
- Persistent CD
- Parallel Tempering
- Natural gradients
Subjective Probability
Probability as a degree of belief

Deep Learning, Estimation theory, hierarchical models, dualities

Probabilistic descriptions of systems and data

Model-Inference-Algorithm

\[
\eta_l = Bx_l
\]

\[
E[y] = g() \Rightarrow \eta_1 = Bx_1
\]
Planting the Seeds of Probabilistic Thinking

Foundations | Tricks | Algorithms

Shakir Mohamed
Research Scientist, DeepMind

@shakir_za
Last Time …

1. Models
   - Convolutional neural network
   - Latent variable model

2. Learning Principles
   - Optimisation methods (SGD, Adagrad)
   - Regularisation (L1, L2, batchnorm, dropout)
   - VEM algorithm
   - Expectation propagation
   - Approximate message passing
   - Variational auto-encoders (VAE)

3. Algorithms
   - Convolutional neural network + penalised maximum likelihood
   - Latent variable model + variational inference
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.

\[ p(x) = \int p(x, z) dz \]
\[ \mathbb{E}[f(z) | x] = \int f(z) p(z | x) dz \]
\[ p(\theta | x_{0:N}) \]

Evidence Estimation

Moment Computation

Parameter Estimation

Prediction

Planning

Hypothesis Testing

Experimental Design

\[ B = \log p(x | H_1) - \log p(x | H_2) \]
\[ IG = D[p(x_{t:T} | u) || p(x_{0:t})] \]
Identity Trick

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

\[ \int p(x)f(x)dx = \mathbb{E}_{p(x)}[f(x)] \]

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

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

Conditions

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

Integral problem

$$p(x) = \int p(x|z)p(z)dz$$

Probabilistic one

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

Re-group/re-weight

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

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

Figure 11.8

Importance sampling addresses the problem of evaluating the expectation of a function \( f(z) \) with respect to a distribution \( p(z) \) from which it is difficult to draw samples directly. Instead, samples \( \{z^{(l)}\} \) are drawn from a simpler distribution \( q(z) \), and the corresponding terms in the summation are weighted by the ratios \( \frac{p(z^{(l)})}{q(z^{(l)})} \).

Furthermore, the exponential decrease of acceptance rate with dimensionality is a generic feature of rejection sampling. Although rejection can be a useful technique in one or two dimensions it is unsuited to problems of high dimensionality. It can, however, play a role as a subroutine in more sophisticated algorithms for sampling in high dimensional spaces.

11.1.4 Importance sampling

One of the principal reasons for wishing to sample from complicated probability distributions is to be able to evaluate expectations of the form (11.1). The technique of importance sampling provides a framework for approximating expectations directly but does not itself provide a mechanism for drawing samples from distribution \( p(z) \).

The finite sum approximation to the expectation, given by (11.2), depends on being able to draw samples from the distribution \( p(z) \). Suppose, however, that it is impractical to sample directly from \( p(z) \) but that we can evaluate \( p(z) \) easily for any given value of \( z \). One simplistic strategy for evaluating expectations would be to discretize \( z \)-space into a uniform grid and to evaluate the integrand as a sum of the form

\[
E[f] \approx \sum_{l=1}^{L} p(z^{(l)}) f(z^{(l)})
\]

(11.18)

An obvious problem with this approach is that the number of terms in the summation grows exponentially with the dimensionality of \( z \). Furthermore, as we have already noted, the kinds of probability distributions of interest will often have much of their mass confined to relatively small regions of \( z \) space and so uniform sampling will be very inefficient because in high-dimensional problems, only a very small proportion of the samples will make a significant contribution to the sum. We would really like to choose the sample points to fall in regions where \( p(z) \) is large, or ideally where the product \( p(z) f(z) \) is large.

As in the case of rejection sampling, importance sampling is based on the use of a proposal distribution \( q(z) \) from which it is easy to draw samples, as illustrated in Figure 11.8. We can then express the expectation in the form of a finite sum over Monte Carlo Estimator

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

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.

**Trace problem**
\[ Tr(A) \]

**Zero mean unit var**
\[ (\Sigma + mm^T) \]

**Identity Trick**
\[ Tr(\textbf{AI}) = Tr(\textbf{A}E[\textbf{zz}^\top]) \]

**Linear operations**
\[ E[Tr(\textbf{Azz}^\top)] \]

**Trace property**
\[ E[\textbf{z}^\top \textbf{Az}] \]

Sampling \( z \) randomly, compute Trace using linear systems of equations.

\[ \partial(\log \det(\textbf{X})) = Tr(\textbf{X}^{-1} \partial \textbf{X}) \]

\[ Tr(A) = \sum A_{ii} \]
**Probability Flow Tricks**

- **Distribution and sample**
  
  \[ \hat{x} \sim p(x) \]

- **Transformation**
  
  \[ \hat{y} = g(\hat{x}; \theta) \]
  
  \[ \mathbb{E}_{p(x)}[g(x; \theta)] \]

- **Unconscious Statistician**
  
  \[ p(y) = p(x) \left| \frac{dg}{dx} \right|^{-1} \]

- **Change of Variables**
  
  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**
  
  \[ \log \det \left( \frac{\partial f(z)}{\partial z} \right) \]
  \[ f(z) = z + uh(w^T z + b) \]
  \[ \det(I + ab^T) = 1 + a^T b \]
  \[ \det(I + us^T) = (1 + u^T s) \]
  \[ s = h'w \]

Linear time computation of the determinant and its gradient.
Normalising Flows

**Sampling and Entropy**

\[ z_K = f_K \circ \ldots \circ f_2 \circ f_1(z_0) \]

\[ \log q_K(z_K) = \log q_0(z_0) - \sum_{k=1}^{K} \log \det \frac{\partial f_k}{\partial z_k} \]

\[ q(z') = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1} \]

- **Unit Gaussian**
- **Uniform**

Shakir Mohamed

DeepMind
Stochastic Optimisation

Common gradient problem

\[ \nabla \phi \mathbb{E}_{q_\phi(z)}[f_\theta(z)] = \nabla \int q_\phi(z)f_\theta(z)dz \]

- 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.

\[ z \sim q_\phi(z) \]
\[ z = g(\epsilon, \phi) \quad \epsilon \sim p(\epsilon) \]

R \[ x = \mu + Rz \]

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

\[ \nabla_{\phi} \mathbb{E}_{q(z)} \left[ f(z) \right] = \nabla_{\phi} \int q_{\phi}(z) f(z) \, dz \]

\[ = \nabla_{\phi} \int p(\epsilon) \frac{d\epsilon}{dz} f(g(\epsilon, \phi)) g'(\epsilon, \phi) \, d\epsilon \]

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

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.

\[ \nabla_{\phi} \log q_\phi(z) = \frac{\nabla_{\phi} q_\phi(z)}{q_\phi(z)} \]

Several useful properties

**Expected score**

\[ \mathbb{E}_{q(z)} [\nabla_{\phi} \log q_\phi(z)] = 0 \]

**Fisher Information**

\[ \nabla [\nabla_\theta \log p(x; \theta)] = \mathcal{I}(\theta) = \mathbb{E}_{p(x;\theta)} [\nabla_\theta \log p(x; \theta) \nabla_\theta \log p(x; \theta)^\top] \]
Score-function Estimator

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

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

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

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

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

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

\[ p(x) = \int p(x|z)p(z)dz \]

Proposal

\[ p(x) = \int p(x|z)p(z)\frac{q(z)}{q(z)}dz \]

Importance Weight

\[ p(x) = \int p(x|z)\frac{p(z)}{q(z)}q(z)dz \]

Jensen's inequality

\[ \log p(x) \geq \int q(z) \log \left( p(x|z)\frac{p(z)}{q(z)} \right) dz \]
\[ = \int q(z) \log p(x|z) - \int q(z) \log \frac{q(z)}{p(z)} \]

Lower bound

\[ \mathbb{E}_{q(z)}[\log p(x|z)] - KL[q(z)\|p(z)] \]
The ratio of two densities can be computed using a classifier of using samples drawn from the two distributions.

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

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
\[ p^*(x) = p(x \mid y = 1) \quad q(x) = p(x \mid y = -1) \]

Bayes' Rule
\[ p(x \mid y) = \frac{p(y \mid x) p(x)}{p(y)} \]

Conditional
\[ \frac{p^*(x)}{q(x)} = \frac{p(x \mid y = 1)}{p(x \mid y = -1)} \]

Bayes' Subst.
\[ = \frac{p(y = +1 \mid x) p(x)}{p(y = +1)} \bigg/ \frac{p(y = -1 \mid x) p(x)}{p(y = -1)} \]

Class probability
\[ \frac{p^*(x)}{q(x)} = \frac{p(y = 1 \mid x)}{p(y = -1 \mid x)} \]

Computing a density ratio is equivalent to class probability estimation.
**Final Words**

Strengthen your probabilistic dexterity.

<table>
<thead>
<tr>
<th>Identity</th>
<th>$\frac{p(x)}{p(x)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hutchinson’s</td>
<td>$\mathbb{E}[zz^\top] = I$</td>
</tr>
<tr>
<td>Flows</td>
<td>$p(y) = p(x) \left</td>
</tr>
<tr>
<td>Log-derivative</td>
<td>$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$</td>
</tr>
<tr>
<td>Reparameterisation</td>
<td>$z = g(\epsilon, \phi)$  $\epsilon \sim p(\epsilon)$</td>
</tr>
<tr>
<td>Density Ratio</td>
<td>$\frac{p^*(x)}{q(x)} = \frac{p(y = 1</td>
</tr>
</tbody>
</table>
Planting the Seeds of Probabilistic Thinking

Foundations | Tricks | Algorithms

Shakir Mohamed
Research Scientist, DeepMind
Shakir Mohamed

Subjective Probability
Probability as a degree of belief

\[ \mathbb{E}[y] = g() \]

\[ \eta_l = Bx_l \]

\[ \eta_1 = Bx_1 \]
Manipulating Integrals

**Evidence Estimation**

\[ p(x) = \int p(x, z) dz \]

**Moment Computation**

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

**Parameter Estimation**

\[ p(\theta|x_{0:N}) \]

**Prediction**

\[ p(x_{t+1}|x_{0:t}) \]

**Planning**

\[ J = \mathbb{E}_p \left[ \int_0^\infty C(x_t) dt | x_0, u \right] \]

**Hypothesis Testing**

\[ \mathcal{B} = \log p(x|H_1) - \log p(x|H_2) \]

**Experimental Design**

\[ \mathcal{IG} = \mathcal{D}[p(x_{t:T}|u) || p(x_{0:t})] \]
<table>
<thead>
<tr>
<th>Your Tricks</th>
<th>Strengthen your probabilistic dexterity.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identity</strong></td>
<td>$\frac{p(x)}{p(x)}$</td>
</tr>
<tr>
<td><strong>Hutchinson’s</strong></td>
<td>$\mathbb{E}[zz^\top] = I$</td>
</tr>
<tr>
<td><strong>Flows</strong></td>
<td>$p(y) = p(x) \left</td>
</tr>
<tr>
<td><strong>Log-derivative</strong></td>
<td>$\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$</td>
</tr>
<tr>
<td><strong>Reparameterisation</strong></td>
<td>$z = g(\epsilon, \phi) \quad \epsilon \sim p(\epsilon)$</td>
</tr>
<tr>
<td><strong>Density Ratio</strong></td>
<td>$\frac{p^*(x)}{q(x)} = \frac{p(y = 1</td>
</tr>
</tbody>
</table>
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

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

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

Products
- Super-resolution
- Compression
- Text-to-speech

AI
- Planning
- Exploration
- Intrinsic motivation
- Model-based RL

Science
- Proteomics
- Drug Discovery
- Astronomy
- High-energy physics
Fully-observed conditional generative model
Compression-Communication

Original

bicubic (21.59dB/0.6423)

SRGAN (20.34dB/0.6562)

Compression rate: 0.2bits/dimension
Generative Design

Shakir Mohamed
Video from work of Memo Aktem
Advancing Science
Advancing Healthcare

M.L. allows computers to learn
Define goal
Score each move
Learn from others
Get best possible score

Leading cause of death
Sepsis
Very hard to identify
Every hour counts

WHAT DO WE NEED?

Smart engineers working in healthcare
Open EHRs
Quality based HC system

COLLECTIVE NEXT

Shakir Mohamed
Types of Generative Models

**Fully-observed models**

- $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow \ldots$

- $f(z)$

**Latent variable models**

- $z \rightarrow x$

**Undirected Models**

- $z_i \rightarrow z_j$

**Sum-Product Networks**

- $x_1 \times x_2 \times x_3 \times x_4 \times x_5$

Shakir Mohamed
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

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

Model Parameters are global variables.
Stochastic activations & unobserved random variables are local variables.

$X_k \sim \text{Cat}(X_k | \pi)$

$X_2 \sim \text{Cat}(X_2 | \pi(X_1))$

$\ldots$

$X_i \sim \text{Cat}(X_i | \pi(X_{<n}))$

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

Fully-observed models

Markov Models

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

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

Continuous

Boltzmann Machines
Discrete Markov Random Fields
Ising, Hopfield and Potts Models

Gaussian MRFs
Log-linear models

Discrete

Undirected
Latent Variable Models

**Prescribed models**
Use observer likelihoods and assume observation noise.

**Implicit models**
Likelihood-free or simulation-based models.

Latent variable models
Introduce an unobserved local random variables that represents hidden causes.

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

Deep Latent Gaussian Model

\begin{align*}
  z_3 & \sim \mathcal{N}(0, I) \\
  z_2 | z_3 & \sim \mathcal{N}(\mu(z_3), \Sigma(z_3)) \\
  z_1 | z_2 & \sim \mathcal{N}(\mu(z_2), \Sigma(z_2)) \\
  x | z_1 & \sim \mathcal{N}(\mu(z_1), \Sigma(z_1))
\end{align*}

Convolutional DRAW

Figure 10. Generated samples from a network trained on 64 \times 64 ImageNet with input scaling \(\epsilon = 0.4\). Qualitatively asking the model to be less precise seems to lead to visually more appealing samples.
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.
Implicit Models

Change of variables for invertible functions

\[ z \sim p(z) \]

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

\[ x = \mu + Rz \]

\[ z \sim \mathcal{N}(0, I) \]

\[ x = f(z; \theta) \]

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

Implicit models

Transform an unobserved noise source using a parameterised function.

Generator Networks

Project and reshape

CONV 1

Stride 2

16

CONV 2

512

16

CONV 3

256

16

CONV 4

64

16

G(z)
**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*

**Bedrooms**

Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

4.3 **MAGENET**

We use Imagenet-1k (Deng et al., 2009) as a source of natural images for unsupervised training. We train on $32 \times 32$ min-resized center crops. No data augmentation was applied to the images.
Shakir Mohamed

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

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

---

Continuous time

Discrete time
Model-Inference-Algorithm

**Variational Autoencoders (VAEs)**

Prescribed latent variable models and variational inference

**Generative Adversarial Networks (GANs)**

Implicit latent variable models and estimation-by-comparison

\[
\begin{align*}
    z &\sim q(z \mid x) \\
    x &\sim p(x \mid z)
\end{align*}
\]
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

\[ p(x) = \int p(x, z) dz \]

Integral is intractable in general and requires approximation.

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

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

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.
Variational Methods

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

\[ KL[q(z|y) \| p(z|y)] \]

Approximation class

True posterior

Deterministic approximation procedures with bounds on probabilities of interest.

Fit the variational parameters.
**Variational Bound**

Interpreting the bound:

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

- **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.
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-field methods assume that the distribution is factorised.

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

True Posterior

Fully-factorised

Most Expressive

Least Expressive

\[
q^*(z|x) \propto p(x|z)p(z)
\]

\[
q_{MF}(z|x) = \prod_k q(z_k)
\]
Structured mean-field: introduce dependencies into our factorisation.

\[
\begin{align*}
q^*(z|x) &\propto p(x|z)p(z) \\
q(z) &= \prod_k q_k(z_k|\{z_j\}_{j\neq k}) \\
q_{MF}(z|x) &= \prod_k q(z_k)
\end{align*}
\]
Latent Gaussian Models

Examples: GP regression or DLGM.

**Probabilistic Model**
\[ z \sim \mathcal{N}(z|0, 1) \quad y \sim p(y|f_\theta(z)) \]

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

**Variational bound**
\[ \mathcal{F}(y, q) = \mathbb{E}_{q(z)}[\log p(y|z)] - KL[q(z)\|p(z)] \]
\[ \mathcal{F}(y, q) = \mathbb{E}_{q(z)}[\log p(y|z)] - \sum_i KL[q(z_i)\|p(z_i)] \]
\[ \mathcal{F}(y, q) = \mathbb{E}_{q(z)}[\log p(y|z)] - \sum_i KL[\mathcal{N}(z_i|\mu_i, \sigma_i^2)\|\mathcal{N}(z_i|0, 1)] \]
\[ \mathcal{F}(y, q) = \mathbb{E}_{q(z)}[\log p(y|f_\theta(z))] - \frac{1}{2} \sum_i (\sigma_i^2 + \mu_i^2 - 1 - \ln \sigma_i^2) \]
Families of Approximations

**True Posterior**

**Families of Posterior Approximations**

- **Normalising flows**
  - $z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow \cdots \rightarrow z_K$

- **Structured mean-field**
  - $z_1 \xrightarrow{p(z)} z_2 \xrightarrow{p(x|z)} z_3 \rightarrow \cdots$

- **Covariance models**
  - $z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow \cdots \rightarrow z_K$

- **Auxiliary variables**
  - $z_0 \rightarrow z_1 \rightarrow \cdots \rightarrow z_K$

- **Mixtures**
  - $y \rightarrow z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow \cdots$

**Most Expressive**

\[ q^*(z|x) \propto p(x|z)p(z) \]

**Least Expressive**

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

Shakir Mohamed

DeepMind
Variational Optimisation

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

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

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

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

Repeat:

- E-step: \( \phi \propto \nabla \phi F(x, q) \) \textbf{Var. params}
- M-step: \( \theta \propto \nabla \theta F(x, q) \) \textbf{Model params}

Initialisation: \( \log p(x) \)  
\( KL[q||p^*] \)  
\( F(x, q) \)  
\( t = 1 \)  
Convergence
Amortised Inference

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

- **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

\[ \nabla_{\phi} \mathbb{E}_{q_\phi(z)} [f_{\theta}(z)] = \nabla \int q_\phi(z) f_{\theta}(z) dz \]

\[ p(x, z) \quad q_\phi(z | x) \quad \nabla_{\phi} \quad \int (\ldots) q_\phi(z | x) dz \quad q_\phi(z | x) \]

Doubly stochastic estimators

Pathwise Estimator
When easy to use transformation is available and differentiable function \( f \).

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

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

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

Reparameterisation

Identity

Log-derivative

Shakir Mohamed

DeepMind
Variational Autoencoder

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

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.
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.

Learning principle: Two-sample tests

\[
\frac{p^*(x)}{q(x)} = 1 \quad p^*(x) = q(x)
\]

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

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

\[
p(y = +1|x) = D_\theta(x) \quad p(y = -1|x) = 1 - D_\theta(x)
\]

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

Generative Adversarial Networks

Alternating optimisation \[\min_{\phi} \max_{\theta} \mathcal{F}(x, \theta, \phi)\]

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

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

Instances of testing and inference:
- Unsupervised-as-supervised learning
- Classifier ABC
- Noise-contrastive estimation
- Adversarial learning and GANs
**Method of Moments**

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

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

\[ F = \| \mathbb{E}_{p^*}(x)[f(x)] - \mathbb{E}_p(z)[f(g_\phi(z))] \|^2 \]

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

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

\[
\mathcal{F}(q, \theta) = \mathbb{E}_{q_\phi(z|x)} \left[ \log p_\theta(x|z) \right] - \text{KL}[q_\phi(z|x) \parallel p(z)]
\]

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

Marginal KL
Information
Environment as a generative process:
- An unknown likelihood;
- Not known analytically;
- Only able to observe its outcomes.

\[ a \sim p(a) \]

\[ u(s, a) \sim \text{Environment}(a) \]

\[ p(R(s)|a) \propto \exp(u(s, a)) \]

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

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

Recover policy search methods:
- Uniform prior over distributions
- Continuous policy parameters
- Can evaluate environment, but not differentiate.
Policy Search

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

Policy gradient using score-function gradient

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

- 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.
Hierarchical Planning

Variational MDP

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

Indaba 2018
Stellenbosch, South Africa

Indaba 2019
Nairobi, Kenya

DEEP LEARNING INDABA

Shakir Mohamed
Planting the Seeds of Probabilistic Thinking

Foundations | Tricks | Algorithms

Shakir Mohamed
Research Scientist, DeepMind

shakirm.com/feedback
Probabilistic Thinking

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
Michael C Fu, Gradient estimation, Handbooks in operations research and management science, 2006
• Applications of Deep Generative Models

• Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models." ICML 2014


• **Fully-observed Models**

• **Implicit Probabilistic Models**
Latent variable models
Inference and Learning


Amortised Inference


