Normalizing Flows Tutorial

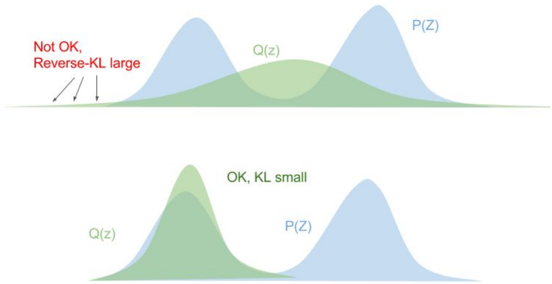
Michael Przystupa

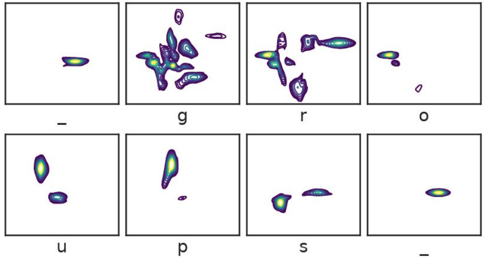
Outline

- Motivation
- Background / Review
  - Determinants, Jacobian, distribution functions
- Change of Variable Theorem
  - Increasing/ Decreasing Function, Actual theorem
- Normalizing Flows
  - Definition, Ways to use, main focus of most research
- Other Stuff for Flows
  - Sampling, log probability, some examples, on flow parameterization

Motivation

- Gaussian distribution is everywhere in machine learning:
  - Variational inference, reinforcement learning, GANs, VAEs, etc.
- It is a pretty nice distribution
  - Analytic KL form, central limit theorem, etc.
- **Problem:** the world isn't Gaussian
  - Multi-modal distributions exist everywhere
- Can we do better in a way that still scales?
  - Background first though

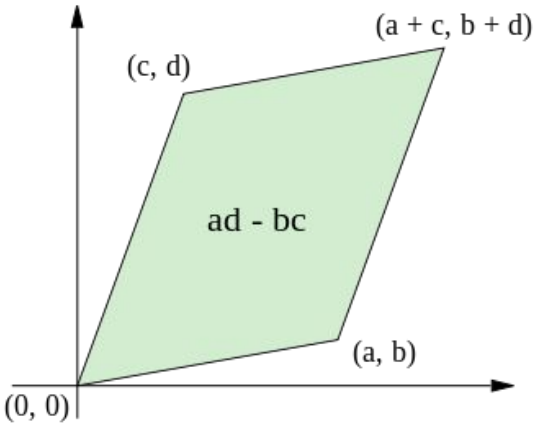




Determinant

- Measures **volume** of n-dim parallelotope with some square matrix **A**
  - det(A) is a scalar value
- Can be +/-
  - Affects orientation in space
- Useful properties for later:
  - $\det(A^{-1}) = 1 / \det(\mathbf{A})$
  - if $\det(A) \neq 0 \Rightarrow A^{-1}$ exists
- E.g.
  - In 2D case determinant can give you area of a square:

$$det \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = 2 * 2 - 1 * 1 = 3$$



Jacobian Matrix

- Matrix **J** of partial derivatives of a vector-valued function:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$f(\mathbf{x}) = [ReLU(x_1), tanh(x_2)] = \begin{bmatrix} \frac{\partial \mathrm{ReLU}}{\partial x_1} & \frac{\partial \mathrm{ReLU}}{\partial x_2} \\ \frac{\partial \mathrm{tanh}}{\partial x_1} & \frac{\partial \mathrm{tanh}}{\partial x_2} \end{bmatrix}$$

Consider the function $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2$, with $(x, y) \mapsto (f_1(x, y), f_2(x, y))$, given by

$$\mathbf{f}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix} = \begin{bmatrix} x^2 y \\ 5x + \sin y \end{bmatrix}.$$

Then we have

$$f_1(x,y) = x^2 y$$

and

$$f_2(x,y) = 5x + \sin y$$

and the Jacobian matrix of $\mathbf{f}$ is

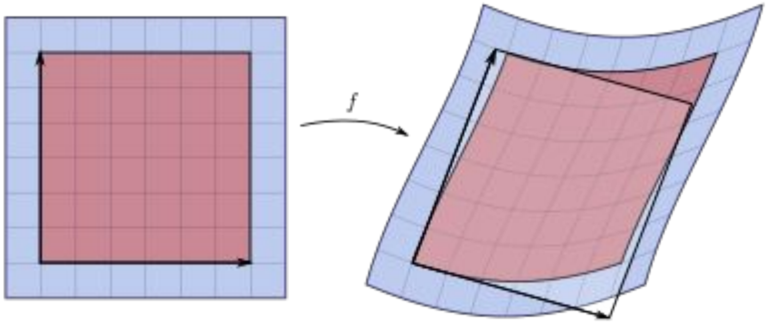
$$\mathbf{J}_{\mathbf{f}}(x,y) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} 2xy & x^2 \\ 5 & \cos y \end{bmatrix}$$

and the Jacobian determinant is

$$\det(\mathbf{J}_{\mathbf{f}}(x,y)) = 2xy\cos y - 5x^2.$$

THE Jacobian

- $f: R^n \Rightarrow R^n$ & differential $\Rightarrow$ Jacobian Matrix J is Square
- We can calculate the determinant: det(J)
- Properties of det(J) at **p** in R^n:
  - If det(J(**p**)) != 0 $\Rightarrow$ f^{-1} exists at **p**
  - det(J) > 0 $\Rightarrow$ orientation preserved
  - det(J) < 0 $\Rightarrow$ orientation reversed
  - |det(J(**p**))| : factor f expand/shrink volume near **p**



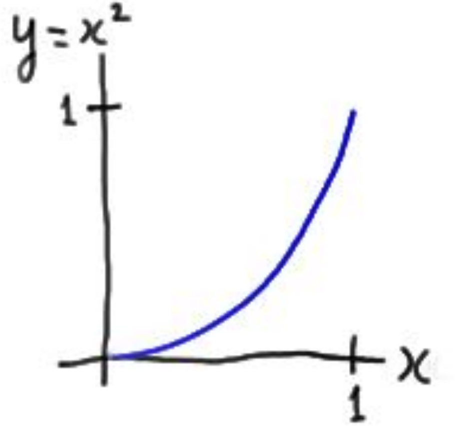
Functions of Probability Distributions

- Given random variables (R.V.) X and Y s.t. $Y = U(X)$
  - i.e. Y is a **function** of X
- How do we find the p.d.f?
  - p.d.f: probability density function
- Two Steps:
  1. Find cumulative distribution function: $F_Y(y) = P(Y <= y)$
  2. Differentiate w.r.t (y): $f_Y(y) = F_Y'(y)$
     - f_Y = p.d.f

Example with Increasing Function

R.V. X with p.d.f X = $3x^2$, support: 0 <= x <= 1

What is p.d.f of $Y = X^2$?



$$F_Y(y) = P(Y \le y) = P(X^2 \le y) = P(X \le y^{1/2}) = F_X(y^{1/2})$$

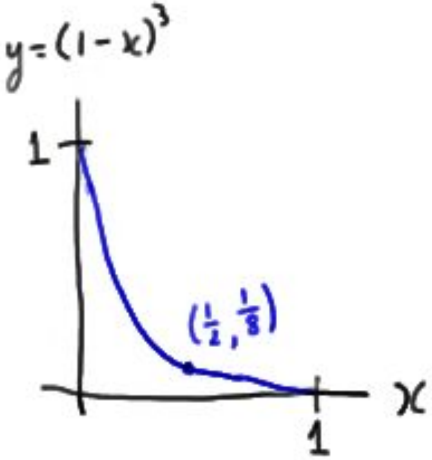
$$= \int_0^{y^{1/2}} 3t^2\,dt = \left[t^3\right]_{t=0}^{t=y^{1/2}} = y^{3/2}, \quad 0 < y < 1$$

Example with Decreasing Function

For X with p.d.f $X = 3(1 - x)^2$, support: $0 <= x <= 1$

What is p.d.f of $Y = (1 - X)^{2}$?

$$F_Y(y) = P(Y \le y) = P((1-X)^3 \le y) = P(1-X \le y^{1/3})$$
$$= P(-X \le -1 + y^{1/3}) = P(X \geq 1 - y^{1/3}) = 1 - F_X(1 - y^{1/3})$$
$$= 1 - \int_0^{1-y^{1/3}} 3(1-t)^2\,dt = 1 + \left[(1-t)^3\right]_{t=0}^{t=1-y^{1/3}}$$
$$= 1 + \left[(1-(1-y^{1/3}))^3 - (1-0)^3\right]$$
$$= 1 + y - 1 = y$$

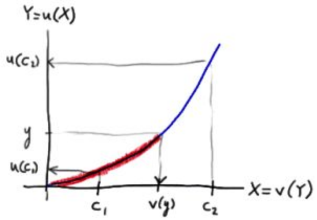


Generalizing Previous Examples

- Y = u(X) , X = v(Y),
  - assumes u is continuous & either increasing or decreasing
  - $v = u^{-1}$, because it is continuous
  - c_i = values from X , $d_i = u(c_i)$ i.e. values from Y
- u(X) is an **increasing** function:

$$F_Y(y) = P(Y \leq y) = P(u(X) \leq y) = P(X \leq v(y)) = \int_{c_1}^{v(y)} f(x)dx$$

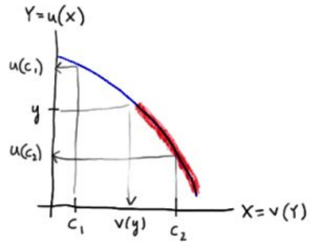
$$f_Y(y) = F_Y'(y) = f_x(v(y)) \cdot v'(y)$$



- u(X) is a **decreasing** function:

$$F_Y(y) = P(Y \leq y) = P(u(X) \leq y) = P(X \geq v(y)) = 1 - P(X \leq v(y)) = 1 - \int_{c_1}^{v(y)} f(x)dx$$

$$f_Y(y) = F_Y'(y) = -f_x(v(y)) \cdot v'(y)$$

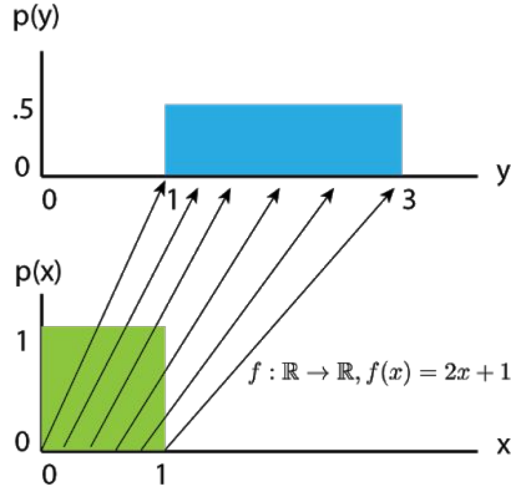


Change of Variable Theorem (Single Variable)

Definition. Let X be a continuous random variable with generic probability density function $f(x)$ defined over the support $c_1 < x < c_2$. And, let $Y = u(X)$ be an *invertible* function of X with inverse function $X = v(Y)$. Then, using the **change-of-variable technique**, the probability density function of Y is:

$$f_Y(y) = f_X(v(y)) \times |v'(y)|$$

defined over the support $u(c_1) < y < u(c_2)$.



Theorem 1.1.2 (Change of Variables) *Let V and W be bounded open sets in $\mathbb{R}^n$. Let $h : V \to W$ be a 1-1 and onto map, given by*

$$h(u_1, \ldots, u_n) = (h_1(u_1, \ldots, u_n), \ldots, h_n(u_1, \ldots, u_n)) .$$

Let $f : W \to \mathbb{R}$ be a continuous, bounded function. Then

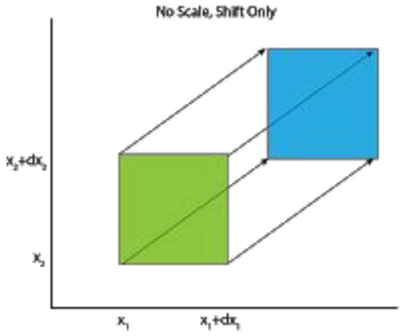
$$\begin{aligned} &\int \cdots \int_W f(x_1, \ldots, x_n) dx_1 \cdots dx_n \\ &= \int \cdots \int_V f\left(h(u_1, \ldots, u_n)\right) J(u_1, \ldots, u_v) du_1 \cdots du_n, \end{aligned}$$

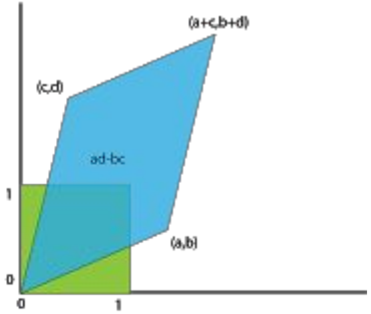
where J is the ***Jacobian***

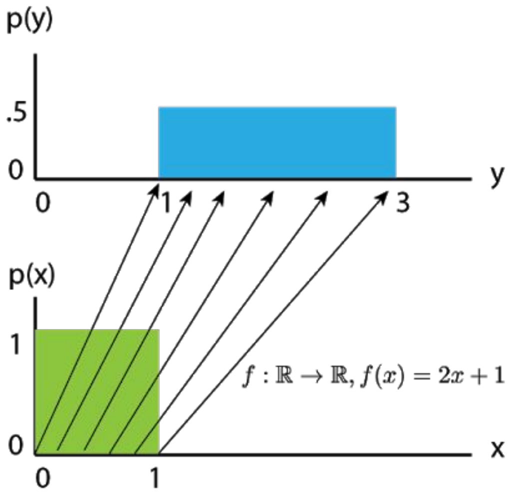
$$J = \begin{vmatrix} \frac{\partial h_1}{\partial u_1} & \cdots & \frac{\partial h_1}{\partial u_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial u_1} & \cdots & \frac{\partial h_n}{\partial u_n} \end{vmatrix} .$$

Recap

- We know that the Jacobian stretches / shrinks functions
  - + some other nice little properties
- Change of variables lets us represent a distribution as a function of another distribution
- Great, but where to normalizing flow come in?







Normalizing Flows

- An application of *Change of Variables* theorem in machine learning
- Given:
  - Base distribution $q(\mathbf{z}_0)$
  - Invertible, differentiable function(s) f
- Allows one to model distributions directly:
  - Reminder for square matrix A: $\det(A^{-1}) = 1 / \det(\mathbf{A})$

$$q_y(\mathbf{y}) = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}$$

Normalizing Flows Research Focus

- Find numerically stable invertible functions that are computationally efficient
- Determinants can be expensive to calculate
  - $O(n^3)$ in worse case
- Most existing flows play with facts about determinant
  - E.g. Square Diagonal matrix determinant is product of diagonal terms

$$y_1 = \mu_1 + \sigma_1 z_1$$

$$y_i = \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i$$

Normalizing Flows Research Focus

- Find numerically stable invertible functions that are computationally efficient
- Determinants can be expensive to calculate
  - $O(n^3)$ in worse case
- Most existing flows play with facts about determinant
  - E.g. Square Diagonal matrix determinant is product of diagonal terms

$$y_1 = \mu_1 + \sigma_1 z_1$$

$$y_i = \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i$$

Normalizing Flow Example

- Example: Consider an autoregressive style flow on random variable **z**

$$y_1 = \mu_1 + \sigma_1 z_1$$

$$y_i = \mu(\mathbf{y}_{1:i-1}) + \sigma(\mathbf{y}_{1:i-1}) z_i$$

Let's quickly do it by hand for **y** = $[y_1\ y_2\ y_3]$

Sampling from a Normalizing Flow

- Just sample from your base distribution and pass through transformations

$$\mathbf{z}_K = f_K \circ \cdots \circ f_1(\mathbf{z}_0), \quad \mathbf{z}_0 \sim q_0(\mathbf{z}_0)$$

$$\mathbf{z}_K \sim q_K(\mathbf{z}_K) = q_0(\mathbf{z}_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|^{-1}$$

Code Snippet Directly from Pytorch Distributions

```
def sample(self, sample_shape=torch.Size()):
    """
    Generates a sample_shape shaped sample or sample_shape shaped batch of
    samples if the distribution parameters are batched. Samples first from
    base distribution and applies `transform()` for every transform in the
    list.
    """
    with torch.no_grad():
        x = self.base_dist.sample(sample_shape)
        for transform in self.transforms:
            x = transform(x)
        return x
```

Ways to Optimize

- Can tack onto other models doing variational inference
  - e.g. Variational Autoencoders

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}^{(0)}|\mathbf{x})}\left[\ln p(\mathbf{x}|\mathbf{z}^{(T)}) + \sum_{t=1}^{T} \ln \left|\det \frac{\partial \mathbf{f}^{(t)}}{\partial \mathbf{z}^{(t-1)}}\right|\right] - \mathrm{KL}\big(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(T)})\big)$$

- Can directly optimize log p(x)
  - By comparison ELBO is a lower bound

$$\log p(y) = \log p(f^{-1}(y)) + \log |\det(J(f^{-1}(y)))|$$

- Probably other ways as well
  - Future research?

Code snippet for calculating log probability

```
def log_prob(self, value):
    """
    Scores the sample by inverting the transform(s) and computing the score
    using the score of the base distribution and the log abs det jacobian.
    """
    event_dim = len(self.event_shape)
    log_prob = 0.0
    y = value
    for transform in reversed(self.transforms):
        x = transform.inv(y)
        log_prob = log_prob - _sum_rightmost(transform.log_abs_det_jacobian(x, y),
                                             event_dim - transform.event_dim)
        y = x

    log_prob = log_prob + _sum_rightmost(self.base_dist.log_prob(y),
                                         event_dim - len(self.base_dist.event_shape))
    return log_prob
```

Planar Flows

- Parameters: $\mathbf{u}, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$
- *h* is some non linearity
- Transformation:

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

- Calculating THE Jacobian:

$$\psi(\mathbf{z}) = h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}$$

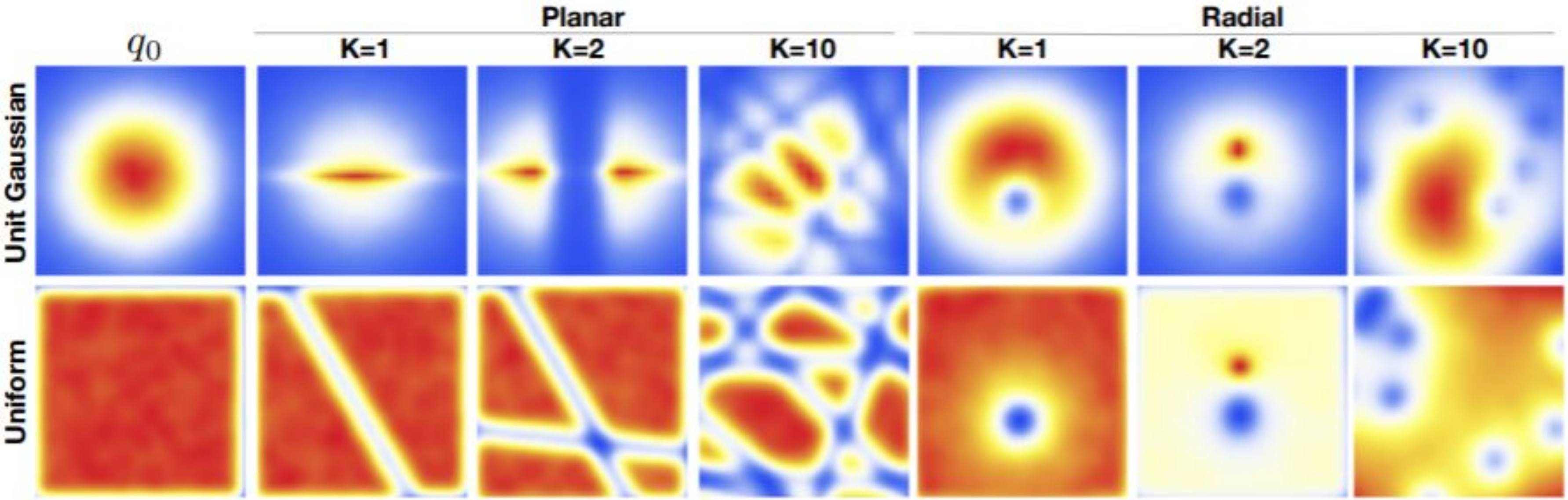
$$\left|\det\frac{\partial f}{\partial \mathbf{z}}\right| = \left|1 + \mathbf{u}^T\psi(\mathbf{z})\right|$$

Radial Flows

- Similar structure to Planar Flows
- Parameters: $\mathbf{z}_0 \in \mathbb{R}^d, \alpha \in \mathbb{R}_+, \beta \in \mathbb{R}$
- Details: $r = \|\mathbf{z} - \mathbf{z}_0\|_2, h(\alpha, r) = \frac{1}{\alpha + r}$
- Function:

$$f(\mathbf{z}) = \mathbf{z} + \beta h(\alpha, r)(\mathbf{z} - \mathbf{z}_0),$$

- Determinant:
    - Same form as planar flows


Planar
Radial
q_0
K=1
K=2
K=10
K=1
K=2
K=10
Unit Gaussian
Uniform

How do you parameterize a flow?

- Directly parameterize the flow
    - Basically like a neural network
    - Less flexible distributions
- Amortize the parameters
    - You train a neural network that outputs the parameters of the flow
    - Allows per datum level flexibility of the flows