Lattice Models

Afzal(180043) and Ramandeep(180589)

February 7, 2022

1 RNVP 2D Python Code

Before we begin with our theoretical discussion on Lattice Models, we shall be skimming through the Python Code for implementing Real Non-Volume Preserving Flows. The code for the same can be accessed by this link.

1.1 Data Generation Function

```
def generate_2d_data(batch_size=1000):
    rng = np.random.RandomState()

    scale = 4.
    centers = [(1, 0), (-1, 0), (0, 1), (0, -1)]
    centers = [(scale * x, scale * y) for x, y in centers]

    dataset = []
    for i in range(batch_size):
        point = rng.randn(2) * 0.5
        idx = rng.randint(4)
        center = centers[idx]
        point[0] += center[0]
        point[1] += center[1]
        dataset.append(point)
    dataset = np.array(dataset, dtype="float32")
    dataset /= 1.414
    return dataset
```

The aforementioned sub-routine essentially chooses a center from the array of `centers` and generates a 2D-normal distribution with its mean as that center. This distribution is a similar to a Gaussian Mixture Model.

1.2 Neural Network Class for Mapping σ & μ

```
class NN(layers.Layer):
    """
    Neural Network Architecture for calculating s and t for Real-NVP

    :param input_shape: shape of the data coming in the layer
    :param hidden_units: Python list-like of non-negative integers, specifying
     the number of units in each hidden layer.
    :param activation: Activation of the hidden units
    """
    def __init__(self, input_shape, n_hidden=[64,64], activation="relu"):
        super(NN, self).__init__(name="nn")
        layer_list = []
        for n in n_hidden:
```

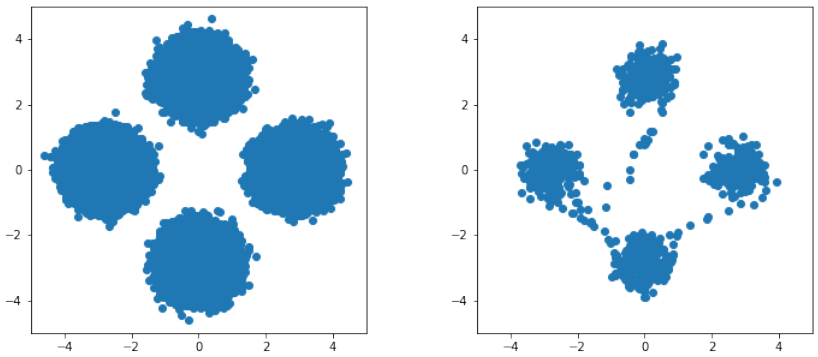
```
            layer_list.append(layers.Dense(n, activation=activation))
        self.layer_list = layer_list
        self.log_s_layer = layers.Dense(input_shape, activation="tanh", name='
    log_s')
        self.t_layer = layers.Dense(input_shape, name='t')

    def call(self, x):
        y = x
        for layer in self.layer_list:
            y = layer(y)
        log_s = self.log_s_layer(y)
        t = self.t_layer(y)
        return log_s, t
```

In the aforementioned Class defn., `s` is the scaling factor & `t` is the shift factor. The variable `layer_list` essentially contains `Dense` layers with `relu` activation. We are finally applying a layer with `tanh` activation for computation of $log(\sigma)$. The `call` function returns $log(\sigma)$ & μ. The $log(\sigma)$ is returned instead of σ to ensure that it remains positive.

For the sake of brevity, we shall be excluding the code of the `RealNVP` Class. It essentially implements Real-NVP Flow in a manner as discussed in our previous lectures.

1.3 Results Comparison



(a) Real Distribution

(b) Approximated Distribution through R-NVP Flow

Figure 1: Plots of Real & Approximated Distribution

As can be seen from the above figure, we have been able to decently approximate the given distribuition.

2 Lattice Models

We shall be considering an example from Statistical Physics to appreciate the application of Normalizing Flows. The model whose distribution we want to sample is called the Ising Model.

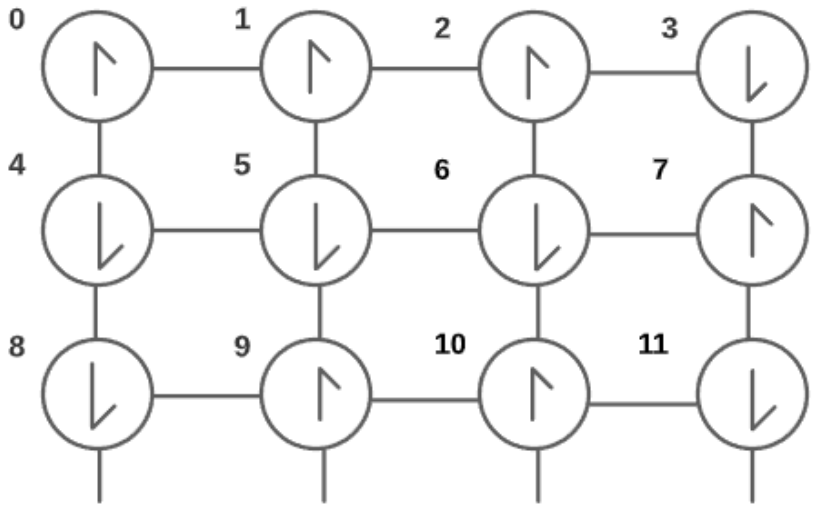


Figure 2: Sample Configuration of an Ising Model on a Lattice

2.1 Mathematical Definition

The realisations that we would sample from this distribution are configurations (as shown in Fig. 2), where every ith node in the lattice takes the value x_i. As per the definition of the given model, $x_i \in \{-1, 1\}$.

The configuration X follows the distribution $p(X)$ defined as:

$$p(X) = \frac{\exp\{-E(X)\}}{Z} \tag{1}$$

wherein $E(X)$ is the energy of interaction of the given configuration & is mathematically defined as:

$$E(X) = -\beta \sum_{<i,j>} x_i x_j \tag{2}$$

Here $\beta \in \mathbb{R}^+$ is a given constant and $< i, j >$ denotes the set containing all the nearest neighbours corresponding to the ith node. *Example:* Consider the node numbered 5 in Fig. 2, plugging in $i = 5$, we get $< i, j >= \{6, 1, 4, 9\}$.

The given distribution mentioned in Eq. (1) prefers configurations with low energy. Before we proceed further, let us observe the following cases.

Case 1: $\beta \approx 0$
$\Rightarrow E \approx 0 \Rightarrow p(X) = \frac{\exp(0)}{Z}$
$\Rightarrow$ All the states have equal energy and therefore equal probability of occurring.

Case 2: $\beta \gg 1$
$\Rightarrow$ The distribution would prefer configurations which have minimum energy $\{E(X)\}$ amongst

all possible configurations. This state with the minimum energy would be if $x_i = 1 \; \forall \; i$ or $x_i = -1 \; \forall \; i$

2.2 Physical Interpretation

The Ising Model is considered as a mathematical model of Ferromagnetism in Statistical Mechanics. Let us say that β is inversely proportional to temperature (T) i.e. $\beta \sim \frac{1}{T}$.

- When $T \to 0$, $E = 0 \Rightarrow X =$ all $\uparrow$ or all $\downarrow$
- When $T \gg 1$, X will be highly random and exhibiting a lot of entropy [$\mathsf{H}(X) = \sum_i p(x_i) \log p(x_i)$]

2.3 Sampling & Estimating pdf

As can be seen from the expression in Eq. (1), the calculation of the normalising constant Z is intractable. Therefore, we can estimate the pdf of this distribution upto a normalising constant i.e. we can calculate the numerator, but not the denominator of the given expression.

The process of sampling new X from this distribution $p_X(x)$ for a given β is not straightforward. It is realistically speaking not viable. One of the possible way around this problem could be by the following methods:

- Normalizing Flows to model $X = f(U)$.
- Ancestral Sampling; seemingly a difficult way, but still doable.
- Mean Field Approximation, which shall be discussed in the following section.

3 Variational Inference

Say that we have a difficult to sample distribution, so in that case what we do is we approximate we a easy to sample distribution.

If we want to approximate our distribution with a normal distribution then our parameters would be μ and σ.

Our approximation: q(x;θ) where θ are the parameters we need to tune.
we could do it in following ways:

$$\theta^* = \arg\min_{\theta} \sum_x \log q(x;\theta) \tag{3}$$

$$\theta^* = \arg\max_{\theta} \; KL\{q(x;\theta)||p^*(x)\} \tag{4}$$

where $p^*(x)$ is our target distribution. We would discuss further about **Mean Field Approximation** in the next lecture.