Lectures Notes from Beg Rohu 2025

Clark Miyamoto (cm6627@nyu.edu)

June 2, 2025

Abstract

Hello....

Contents

I ^^ef^^bf^^bcMallat: Score Based Diffusion & Renormalization Group Flow **2**

1 Overview **2**
1.1 Transport 2
1.2 Outline 3

2 Historical Models: Energy Based Models, GANs, Normalization Flows **3**
2.1 Energy Based Models (EBM) 3
2.1.1 Likelihood 3
2.2 GANs 4
2.2.1 Conditional GANs 4
2.3 Normalizing Flows 5

II Murugan: Leanring w/o Neurons **6**

3 Overview **6**

III Other Things **8**

4 People Presentations **8**

Part I
Mallat: Score Based Diffusion & Renormalization Group Flow

1 Overview

We'll be going from score-diffusion and eventually make our way to the renormalization group.

The setup is you're given a set of input data $\{x_i : x_i \in \mathbb{R}^d\}_{i \leq n}$ which (we believe) to be sampled from a distribution $p(x)$. Your objective is to reconstruct this probability distribution.

Doing this problem in the physics context, the canonical example will be attempting to recreate samples from a Ising / ϕ^4 model. Some more advanced problems are turbulence, cosmology, metrology, and climatology. Apart from physics, you can also generate new image (i.e. conditional generation), and use it to solve inverse problems.

As you start to think more about this problem, it might seem hopeless to learn $p(x)$. This is because the problem suffers from the **curse of dimensionality**. A crude way to calculate this is to first assume $p(x)$ is Lipschifts

Definition 1 (Lipschifts) *That is* $|p(x) - p(x')| \leq K \underbrace{||x - x'||}_{\epsilon \in [0,1]^d}$

This means you'll need $\sim C\epsilon^{-d}$ amount of data... You can see you're screwed.

We can start to tackle this problem by making a key assumption, factorize the distribution

$$p(x) = \prod_k p_k(x) \implies \log p(x) = \sum_k \log p_k(x) \tag{1.1}$$

Two problems now emerge

1. How do we choosing this factorization?
2. How do we prevent it from just simply memorizing a distribution?

The first method is to assume $p(x)$ is a empirical distribution, by this I mean $p(x) = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ (which are get all the observed data, and draw it uniformly).

1.1 Transport

By **transport**, we mean constructing a new function $p_t(x)$ which is a probability distribution $\forall t$, with boundary conditions $p_{t=0}(x) = p(x)$ and $p_{t=T}(x) = p_{gaussian}(x)$. So basically, can we find a new probability distribution which moves from complex things to NOT complex things.

There is a **forward process**, which is some noising operator. And there is a **inverse process**, which undo's the noise (usually involves the ML algorithm). What we'll end up seeing is that these processes ends up being equivalent to RG discovered by Kadanoff & Wilson.

A particular ansatz we can take is

$$p(x) = \underbrace{p(x_T)}_{\text{Gaussian}} \prod_{t=T+1}^{1} p(x_t/x_{t-1}) \tag{1.2}$$

1.2 Outline

1. Energy Based Models, GANs, and Normalizing Flows.
2. Score based diffusion, Fokker Planck, and denoising
3. Generalization and memorization.
4. Renormalization Group
5. Particular models: simple models based on wavelet / scattering transforms.

2 Historical Models: Energy Based Models, GANs, Normalization Flows

2.1 Energy Based Models (EBM)

When we say we have an energy based model, we have a probability distribution over $x = (x_1, ..., x_d)$ which takes the form

$$p_\theta(x) = \frac{e^{-U_\theta(x)}}{Z_\theta}, \quad \text{s.t. } Z_\theta = \int dx\ e^{-U_\theta(x)} \tag{2.1}$$

Example 1 *An example model is a perturbation away from a Gaussian*

$$U_\theta(x) = x^T \Sigma_\theta^{-1} x + \theta \sum_i V(x_i) \tag{2.2}$$

In physics the first one is the kinetic energy term, and the second is the potential energy term. You could start by making U_θ a neural network.

Example 2 *Another example model is the exponential family*

$$U_\theta(x) = \langle \theta, \Phi(x) \rangle \tag{2.3}$$

Notice, this is the Boltzmann distribution in physics, where $\theta \leftrightarrow \beta$ (inverse temperature) and $\Phi(x) \leftrightarrow H(x)$ (Hamiltonian)

2.1.1 Likelihood

Definition 2 (Likelihood) *The negative likelihood $\ell(\theta)$ of probability distribution p*

$$\ell(\theta) = -\mathbb{E}_{x\sim p} \log p_\theta(x) = KL(p||p_\theta) + H[p] \tag{2.4}$$

where KL is the Kullback-Lieber divergence, and H is the entropy.

Our objective will be choose θ s.t. we minimize the negative likelihood

$$\theta^* = argmin_\theta\ \ell(\theta) \tag{2.5}$$

To minimize this, we end up needing to calculate gradients of the likelihood. In the case of energy based models, that would mean we have to calculate the normalization $\nabla_\theta \log Z_\theta$, which in practice is quite hard. However, we can massage the problem into

$$\nabla_\theta \ell(\theta) = \mathbb{E}_{x\sim p}[\nabla_\theta U(x)] + \log Z_\theta \tag{2.6}$$
$$= \mathbb{E}_{x\sim p}[\nabla_\theta U_\theta(x)] - \mathbb{E}_{x\sim p_\theta}[\nabla_\theta U_\theta(x)] \tag{2.7}$$

So now the problem becomes figuring out how to compute $\mathbb{E}_{x\sim p_\theta}$ which involves spending time to construct a Markov Chain Monte Carlo method.

2.2 GANs

GAN stands for Generative Adversarial Network, the original paper is by Goodfellow & Bengio (2014). What you do is you construct two neural networks G_{θ_G} and D_{θ_D} and make their tasks adversarial to one another. G's task is to generate images from noise $z \sim \mathcal{N}(0, 1)$, and D's task is to figure out which image was real vs generated. So to recap, $G : z \to$ Images and $D :$ Images $\to [0, 1]$, where your goal is that

$$D(x) = \begin{cases} 1 & \text{if } x \text{ is a real image} \\ 0 & \text{if } x \text{ is a fake image} \end{cases} \tag{2.8}$$

The loss function is

$$\mathcal{L}(\theta_D, \theta_G) = \mathbb{E}_{x\sim p_{data}}[\log D(x)] + \mathbb{E}_{z\sim \text{Gaussian}}[\log(1 - D(G(z))] \tag{2.9}$$

$$\text{where } D(x) = \begin{cases} 1 & \text{if } x \text{ is a real image} \\ 0 & \text{if } x \text{ is a fake image} \end{cases} \tag{2.10}$$

What ends up happening in practice is that GANs almost always memorize the data instead of learning to gneerate. Another way of saying this is this is a difficult nash-equilibrium to obtain, or you see **mode collapse**. This is why people used them for image generation, and couldn't use them for physics data...

A small aside... If you successfully find the global minimizer/maximizer

Lemma 1 *The optimal discriminator ends up being*

$$D^* = argmax_D\ \mathcal{L}(D, G) \tag{2.11}$$
$$= \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \tag{2.12}$$

Theorem 1

$$\min_G \max_D \mathcal{L}(D, G) \iff p_G = p_{data} \tag{2.13}$$

Both are theoretical results, so they don't consider using finite number of samples.

2.2.1 Conditional GANs

What if you want to condition your learned distribution on some variables (i.e. text to image, or in-painting). Your loss function will be

$$\mathcal{L}(D, G) = \mathbb{E}_{p_{data}}[\log D(x|y)] + \mathbb{E}_{z\sim p_{data}}[\log(1 - D(G(x|y)))] \tag{2.14}$$

2.3 Normalizing Flows

In this problem, we have an easy to sample distribution p_z (which is Gaussian), and difficult to sample distribution p_x (which is the target distribution). The way we'll do transport is to learn a diffeomorphism T_θ, and now we can do change of basis between the two

$$p_x(x) = p_z(T^{-1}(x)) \det \mathcal{J}_{T^{-1}}(x) \tag{2.15}$$

where $\mathcal{J}_{T^{-1}}$ is the Jacobian of T^{-1}.

The likelihood ends up being

$$\ell(\theta) = -\mathbb{E}_{x\sim p}[\log p_z(T_\theta^{-1}(x))] - \mathbb{E}_{x\sim p}[\log|\det -\mathcal{J}_{T_\theta^{-1}}|] \tag{2.16}$$

hi

Part II
Murugan: Leanring w/o Neurons

3 Overview

Since we're talking about learning without neurons, perhaps we should talk about what we mean about making computation. Murugan states there are two types of computation

1. Symbolic: These are analytic expressions which a compute can crunch / a human can set

2. Behavior: Living organism / smart materials which respond and interact with their environment.

The talk will focus on the latter. However computations are used to solve problems, so let's also define some ways of solving problems. Let's work through an example of heating and cooling a room with an AC.

1. Abstraction First:

   An example of this is PI control. So you set up a control flow.

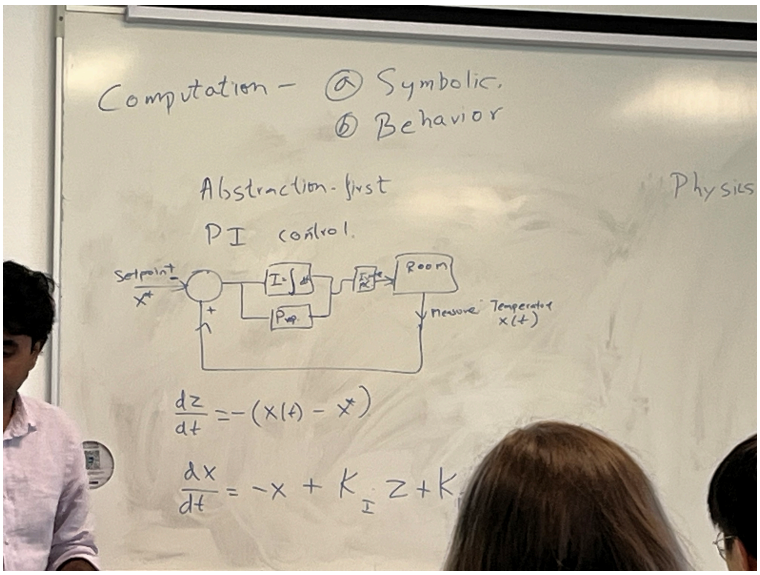

Figure 1: Diagram of control line

   This defines a differentiation equation which defines your control line z due to observation of a process x

   - Pros:
   - General / modular
   - Cons
   - Cannot exploit specific physics
   - The translation between theory vs implementation is non-trivial.

2. Physics First:

   You can set up a thermodynamic state equation, and solve for the phase diagram. Pros and Cons

   - Pros:
   - Fewer parts (measurement, computation), which means its more robust
   - It's passive, meaning it self computes
   - Cons
   - Specific to the physics

This talk will focus on how to move toward the 2nd way of thinking. An example of this is the Liquid-Liquid phase transition of two liquids inside of a cell (see Zechnor 2020). You can think of this as oil and water separating.

There are two communities working on this

1. Energy efficient computations (vs silicon).

   - Example. We currently run computations on a silicon chip, however this is not energy efficient, so much so governments are thinking about building more nuclear reactors to power the next generation of AI). So perhaps we can build biological circuits which run these computations natively on some bio-efficient chip.
   - Example. Encode your optimization problem onto a Ising-like physical system, and anneal that system! Now you can read off optimized parameters.
   - Example. You can use optimal computing to make better matrix multiplication.

2. Biological computations

   - Backprop. That is each organism is reacting and adjusting it's behavior. Much more complex than just physics or chemical behavior.
   - Evolution. No one organism does the computation, but rather it's the ensemble of them which can perform computations.
   - Physics / Chemistry. It's just a reaction based on laws of physics / chemistry.

Now, how does this all relate to machine learning. Let's consider a high dimensional classifier– you could think of the decision boundary as phase boundaries! So your features correspond to temperature, pressure, etc., and your classification is the phase. So the questions become what in the material / organism do I have to tune s.t. you can learn, and this also may find new interesting physical mechanisms for scientists to play with.

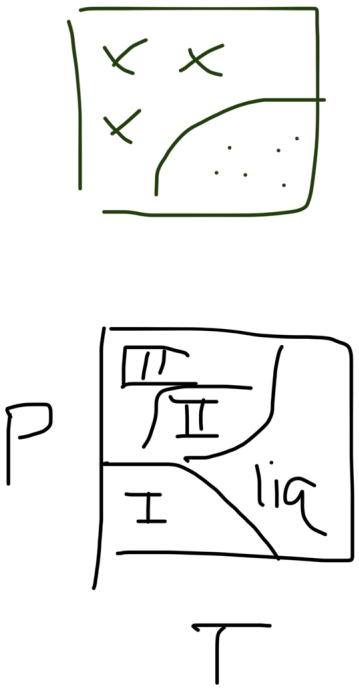

Figure 2: Decision boundary problems can be mapped into a physics system by inspecting their phase transition

Part III
Other Things

4 People Presentations

1. Paolo Balioni, working on feature learning in Bayesian Neural Networks. Also works on MCMC methods.

2. Claudian Merger. Mappping normalizing flows to ϕ^n theory.