Generating data by sampling from EBM

Aapo Hyvärinen

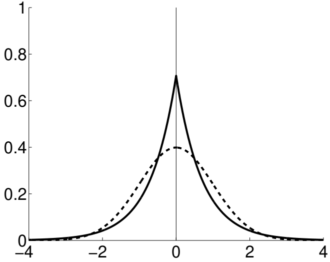
for the NNDL course, 16 April 2025 (second half)

- Definition of sampling as generating new data from a pdf
- Markov Chain Monte Carlo methods
- Combining score matching and MCMC
- Langevin algorithm as an example of MCMC
- Application in Generative AI of images
- Diffusion models

Literature: Bishop's book Chapter 14 and Sections 20.1 and 20.3

The sampling problem formulation

- Given a (possibly unnormalized) pdf p,
  generate a data point $\mathbf{x}$ from that distribution
- We focus here on sampling in $\mathbb{R}^n$
- In general, this is quite difficult
- Consider, e.g. the Laplacian pdf $p(x) = \exp(-\sqrt{2}|x|)/\sqrt{2}$



  How could you sample from it? Not obvious.
- Even more difficult if p is given by a NN and $\dim(\mathbf{x})$ high!
- No simple method efficient in high dimensions exists
- (Terminological point: "sample/ing" has different meanings)

Sampling as the basis for generative AI of images

- If p is the pdf of images, sampling generates new images
  - Basis of modern GenAI of images
- In statistics literature, p given as a mathematical formula, but here, we have to *learn* it
  - Any method for learning energy-based model could be used
- Typical DL sampling methods use the score function

  $\Rightarrow$ Learning naturally by score matching
- Some of the state-of-the-art systems (Dall-e, Stable Diffusion) work approximatively like this:
  - Denoising score matching + sampling by "diffusion model"
  - ... with of course a huge number of technicalities perfected
- Same approach should work on similar data domains:
  - Video, audio
  - Sensor data (e.g. biomedical)
  - Etc.
- Generating text is very different (not $\mathbb{R}^n$), not considered here

Markov Chain Monte Carlo (MCMC) methods

- Most good sampling methods in $\mathbb{R}^n$ for *large* n are MCMC
- *Markov Chain* means here that the method is iterative, producing a sequence of points $\xi_1, \xi_2, \ldots$ so that the distribution approaches p
- "Monte Carlo": Name of a famous gambling casino, refers to the fact that the algorithm is stochastic
- Well-known methods: Metropolis-Hastings algorithm, Hamiltonian Monte Carlo, NUTS, etc.
- Here, we briefly consider some methods typically used in DL:
  - Langevin algorithm: very simple, quite good
  - (Generative) Diffusion Models: more complex, state-of-the-art
- For in-depth information, e.g. course *Computational Statistics*

Langevin algorithm ("flow")

- Perhaps the simplest MCMC method to work well in $\mathbb{R}^n$
- Starting from a random point $\mathbf{x}_0$, compute the sequence

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mu\phi_{\mathbf{x}}(\mathbf{x}_t) + \sqrt{2\mu}\,\mathbf{n}_t \tag{1}$$

  where
  - $\phi_{\mathbf{x}} = \nabla_{\mathbf{x}} \log p(\mathbf{x})$, score function of $\mathbf{x}$ (learned beforehand)
  - μ is a step size (difficult to choose as always)
  - $\mathbf{n}_t$ is Gaussian noise from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (easy to generate by libraries)
- *Theorem:* For infinitesimal μ and in the limit of infinite t, $\mathbf{x}_t$ will be a sample from p.
- Remark: This is the "unadjusted" version valid for infinitesimal μ
  - to give exact sampling even for a non-infinitesimal μ, an "adjusted" method ("MALA") can be used
  - ... but rarely used in DL since it needs p_{un}, score is not enough

An intuitive interpretation of Langevin algorithm

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mu\phi_{\mathbf{x}}(\mathbf{x}_t) + \sqrt{2\mu}\,\mathbf{n}_t \tag{2}$$

- The score function $\phi_{\mathbf{x}}$ works a bit like in a gradient method
  - gives direction of increasing probability
  - $\mathbf{x}_t$ tends to go to areas where there is a lot of probability mass
  - this term alone would make $\mathbf{x}_t$ to go the modes and stay there
- Gaussian noise $\mathbf{n}_t$ introduces randomness
  - the system cannot get stuck anywhere (e.g. in the modes)
  - forces "exploration" of the whole space
  - but this term alone would give just white gaussian noise
- "Seek higher probability but explore at the same time"
- Important to have the right balance between the two terms: sophisticated theory gives us μ vs. $\sqrt{2\mu}$ in above

Illustration of sampling by Langevin

[Show demo langevin.jl]

Image generation by EBM+MCMC: simple variant (DSM+Langevin) summarized

1. Learn density model by denoising score matching:
   Add Gaussian noise of covariance $\sigma^2\mathbf{I}$ to the data to get

$$\tilde{\mathbf{x}}(i) = \mathbf{x}(i) + \mathbf{n}(i) \tag{3}$$

   Minimize the following with respect to $\boldsymbol{\theta}$:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \|\mathbf{x}(i) - [\tilde{\mathbf{x}}(i) + \sigma^2 \phi(\tilde{\mathbf{x}}(i); \boldsymbol{\theta})]\|^2 \tag{4}$$

   where ϕ is NN with parameters $\boldsymbol{\theta}$.
   Denote optimal value by $\boldsymbol{\theta}^*$.
2. Generate data by Langevin algorithm:
   Starting from a random $\mathbf{x}_0$, repeat

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mu\phi(\mathbf{x}_t; \boldsymbol{\theta}^*) + \sqrt{2\mu}\,\mathbf{n}_t \tag{5}$$

   and "after a while", you have a generated image (data point) !

Image generation by EBM+MCMC: some results

1. Learn density model by (denoising) score matching
2. Generate data by Langevin algorithm

- Some examples with dynamics:

Rows: different $\mathbf{x}_0$; Horizontal axis: Iteration t (rescaled).

From (Yang and Ermon, 2019)

Problem of noise in DSM

- But we have a problem:
  - DSM does not estimate the score function of the real image but a noisy version (see slides on SM)
  - ... while basic SM is too heavy to compute in a deep NN due to second derivatives
- Several solutions:
  1. Use such a small noise level that it is not visible
     - but learning in DSM can be unstable for very small noise
  2. Estimate score function for several noise levels, extrapolate to zero noise
     - means more computation but still feasible
  3. SOTA: Use generative diffusion models
     - They are based on scores for noisy data
     - Still needs to estimate score for many noise levels
     - More complex theory
     - Best generated images (Stable Diffusion, Dall-e)
  4. Use something else than DSM
     - E.g. NCE? Active field of research...

Using prompts in generation

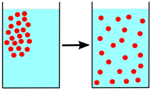
- The method above gives just any random image similar to those training set
- Real systems typically use input, e.g. a textual prompt
- This is formalized as conditional (possibly unnormalized) probability $p(\mathbf{x}|\mathbf{u})$ where $\mathbf{u}$ is some representation of prompt
- Conditional score function defined in obvious way

$$\phi_{\mathbf{x}|\mathbf{u}}(\mathbf{x}|\mathbf{u}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{u}) \tag{6}$$

- The NN computing the score is given the concatenated vector $(\mathbf{x}, \mathbf{u})$ as input
- All the methods above can be easily adapted to use such conditioning
  - Intuitively, think of data in two classes, $\mathbf{u} \in \{1, 2\}$:
    the scores are just learned separately for the two classes, and the score of the desired class is used for generation

Basic idea of generative diffusion models (1)

- "Diffusion is the net movement of anything (for example, atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration" (Wikipedia)



- In probabilistic terms, something like:

  starting from a structured case, go towards randomness
- Diffusion process defined as a stochastic process:
  - starting from a point following initial distribution p
  - add Gaussian noise little by little
  - ending up with just Gaussian noise

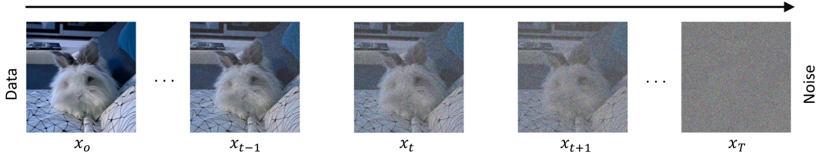
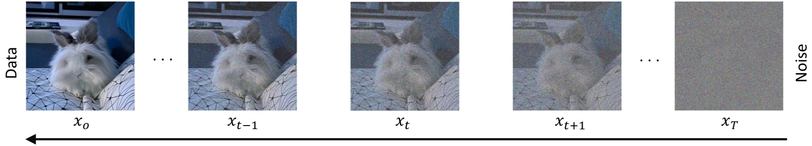


Figure by Croitoru et al (2023)

Basic idea of generative diffusion models (2)

- In generative AI, the idea is to *reverse* the diffusion process
  - starting from an initial Gaussian noise
  - remove Gaussian noise little by little (denoising)
  - ending up with data following "original" distribution p
- More precisely, it is a *reverse* or *denoising* diffusion model



- Denoising is performed by a denoising autoencoder
  - Equivalent to using the score function; the one for noisy data

  $\Rightarrow$ In generative diffusion models,
  estimation of *noisy* score function is not a bug but a feature

Formal definition of generative diffusion model (almost)

- An observed data point (image) is the initial point $\mathbf{x}_0$
- Diffusion created by adding Gaussian noise

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathbf{n}_t \tag{7}$$

  with some hyperparameters β_t
- How to revert this? Try to find $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$.
  - This is "denoising" but reconstructing the whole distribution
  - Learn a NN, $\mathbf{g}_\theta(\mathbf{x}; t)$, to give mean $E\{\mathbf{x}_{t-1}|\mathbf{x}_t\}$
    - Kind of DSM, so Tweedie-Miyasawa shows connection to $\phi_\mathbf{x}$
  - Variance $\mathrm{var}\{\mathbf{x}_{t-1}|\mathbf{x}_t\}$ is a simple function of the β_t
  - This is Gaussian at least with infinitesimal noise
- Must learn denoising function $\mathbf{g}$ for many noise levels t
- For learning $\mathbf{g}$, many variants:
  - Denoising score matching / Denoising autoencoder
  - Variational MLE (kind of VAE)
- Generation by successively sampling from $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$, starting from Gaussian noise $\mathbf{x}_T$ for some large initial T
- For details, check Algorithms 20.1 , 20.2 in Bishop's book

Conclusion

- Sampling from a given pdf is a classic problem in statistics
- Reborn in DL as sampling from a pdf *learned* from the data
  - First learn pdf by energy-based models, then do MCMC
- Basis of state-of-the-art generation in $\mathbb{R}^n$
  - Images, but also potentially audio, video, etc.
  - (Text is very different and often uses other methods)
- Langevin algorithm is a simple fundamental method
- Diffusion models are state-of-the-art at the moment
  - Theory is compatible with Denoising SM
  - Works well even in with real images
- Generative adversarial networks (GAN) are another option (see next lecture)