MCMC in Text Generation

Shlok Mishra

Jaunuary 2024

1 Introduction

- **Language Model**: A pre-trained language model is a computational model that has been previously trained on a large corpus of text data. This training process involves the model learning various aspects of a language, such as its syntax, semantics, and context, without being explicitly programmed with the rules of the language. The goal is to enable the model to understand and generate human-like text, or to perform a variety of language-related tasks.

- **Word Embeddings**: In NLP, this is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.

- **Natural text Generation**: Neural text generation typically involves two stages:

  - modeling a distribution over text sequences
  - using a decoding algorithm to generate sequences with the model

  Let $\boldsymbol{y} = (y_1, \ldots, y_T)$ denote a discrete sequence where each y_t is a token from a vocabulary $\mathcal{V}$. Common neural language models factorize the probability of a sequence into the product of per-token conditionals in left-to-right order, $p_\theta = \prod_{t=1}^{T} p_\theta(y_t|\boldsymbol{y}_{<t})$, with each conditional parameterized by a shared neural network, such as transformers. Popular decoding algorithms produce text sequences $\boldsymbol{y}$ using the model p_θ, often conditioned on prompt $\boldsymbol{x}$

- **Constrained Text Generation**: We view text generation as the problem of finding a sequence that satisfies a collection of constraints. For instance, the scenario above amounts to generating a sequence $\boldsymbol{y} = (y_1, \ldots, y_T)$ subject to a soft constraint that the continuation $\boldsymbol{y}$ should be fluent and logically coherent with the prompt $\boldsymbol{x}$. Other constrained generation problems impose additional constraints, such as text infilling where coherence

constraints move beyond a left-hand prefix, lexically constrained generation in which hard constraints require the output to contain given tokens, and various forms of semantically-constrained generation in which the output is softly constrained to be similar to another sequence. Since common decoding algorithms generate text monotonically, relying on $p_\theta(y_t|\boldsymbol{y}_{<t})$ for determining the next token, it is challenging to enforce these diverse constraints.

2 Literature Review

2.1 Word Embeddings

Word embeddings are a numerical representation technique for words, utilized in the domain of natural language processing. The core concept resembles multi-dimensional scaling, where words are represented as points within a high-dimensional vector space. The spatial arrangement in this space is determined through the analysis of a large text corpus, ensuring that semantic relationships and contextual similarities among words are effectively captured.

Mathematically, if we denote the vocabulary by V and the embedding space by $\mathbb{R}^d$ where d is the dimensionality of the embeddings (commonly in the hundreds), each word $w \in V$ is associated with a vector $\mathbf{v}_w \in \mathbb{R}^d$. The goal of the embedding process is to learn this mapping $f : w \to \mathbf{v}_w$ based on contextual usage patterns in the text.

The essence of word embeddings can also be viewed as a dimensionality reduction technique. It transforms the sparse, high-dimensional space of text data into a denser, continuous vector space. This transformation is not merely based on variance, as in principal component analysis (PCA), but rather optimized to preserve linguistic characteristics.

In terms of mathematical formulation, the similarity between words is quantitatively assessed using distance metrics in the embedding space, such as the cosine similarity. For two words w_1 and w_2, their similarity is given by

$$\text{similarity}(\mathbf{v}_{w_1}, \mathbf{v}_{w_2}) = \frac{\mathbf{v}_{w_1} \cdot \mathbf{v}_{w_2}}{\|\mathbf{v}_{w_1}\|\|\mathbf{v}_{w_2}\|},$$

where closer vectors indicate higher semantic similarity.

The process of learning these embeddings is akin to solving a statistical estimation problem, where the objective is to adjust the vectors such that they can predict the presence of words in a given context. This is often achieved through optimization techniques, minimizing a loss function that captures prediction errors across the corpus.

Applications of word embeddings are vast, extending to tasks such as clustering semantically similar words, solving analogies, or serving as input features for various predictive models. By transforming words into a continuous numerical form, embeddings facilitate the application of statistical analysis and machine learning algorithms to textual data, enabling a deeper understanding of language semantics.

2.2 Energy Based Models (EBMs)

[3, Predicting Structured Data] Energy-based models (EBMs) capture dependencies between variables by associating a scalar energy to each configuration of the variables. Inference consists in clamping the value of observed variables and finding configurations of the remaining variables that minimize the energy. Learning consists in finding an energy function in which observed configurations of the variables are given lower energies than unobserved ones.
Inference, i.e., making a prediction or decision, consists in setting the value of observed variables and finding values of the remaining variables that minimize the energy. *Learning* consists in finding an energy function that associates low energies to correct values of the remaining variables, and higher energies to incorrect values. A *loss functional*, minimized during learning, is used to measure the quality of the available energy functions.

Training an EBM consists in finding an energy function that produces the best Y for any X. The search for the best energy function is performed within a family of energy functions $\mathcal{E}$ indexed by a parameter W

$$\mathcal{E} = \{E(W, Y, X) : W \in \mathcal{W}\}$$

Given an energy function $E(\boldsymbol{y}) \in \mathcal{R}$, an EBM is defined as a Boltzmann distribution $p(\boldsymbol{y}) = exp\{-E(\boldsymbol{y})/Z\}$ where $Z = \sum_{\boldsymbol{y}} exp\{-E(\boldsymbol{y})\}$ is the normalization constant.

2.3 Langevin Dynamics

Langevin dynamics is a stochastic process that is often used in statistical physics and computational chemistry to simulate the behavior of particles in a potential field. It combines deterministic motion under a potential with random motion due to thermal fluctuations. This approach is particularly useful for modeling systems at finite temperatures.

The Langevin equation describes the motion of a particle of mass m under the influence of a potential field and a frictional force, along with a random force representing thermal fluctuations. The equation can be written as:

$$m\frac{d^2\mathbf{x}}{dt^2} = -\nabla U(\mathbf{x}) - \gamma\frac{d\mathbf{x}}{dt} + \sqrt{2\gamma k_B T}\mathbf{R}(t)$$

- $\mathbf{x}$ is the position of the particle.
- $\frac{d^2\mathbf{x}}{dt^2}$ is the acceleration of the particle.
- $U(\mathbf{x})$ is the potential energy as a function of position.
- $\nabla U(\mathbf{x})$ represents the force due to the potential field.
- γ is the friction coefficient.

- $\frac{d\mathbf{x}}{dt}$ is the velocity of the particle.
- k_B is the Boltzmann constant.
- T is the temperature of the system.
- $\mathbf{R}(t)$ is a random force with Gaussian distribution, representing thermal fluctuations.

The Langevin equation is a way to describe the dynamics of particles that are small enough to be influenced by thermal noise, such as molecules in a fluid. It's a cornerstone of molecular dynamics simulations and is used to explore the properties of materials and biological systems at the atomic and molecular levels.

Stochastic Gradient Langevin Dynamics (SGLD) is an algorithm that combines aspects of stochastic gradient descent (SGD) with Langevin dynamics, a method from statistical physics. SGLD is particularly useful in the context of Bayesian learning and deep learning, where it is used for sampling from the posterior distribution of model parameters.

The basic idea behind SGLD is to add a noise term to the gradient updates in SGD, where this noise term is scaled in a way that approximates sampling from the true posterior distribution under certain conditions. This allows SGLD to not only find a point estimate of the model parameters (like standard SGD) but also to explore the parameter space more broadly, providing information about the uncertainty in the model parameters.

The update rule for SGLD can be written as:

$$\theta_{t+1} = \theta_t - \frac{\epsilon_t}{2}\left(\nabla_\theta \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla_\theta \log p(y_i|\theta_t)\right) + \eta_t,$$

where:

- θ_t is the parameter vector at iteration t.
- ϵ_t is the learning rate at iteration t.
- $\nabla_\theta \log p(\theta_t)$ is the gradient of the log prior distribution with respect to the parameters.
- N is the total number of data points.
- n is the number of data points used in the stochastic gradient estimation (batch size).
- $\nabla_\theta \log p(y_i|\theta_t)$ is the gradient of the log likelihood for a data point y_i given the parameters.
- η_t is a noise term sampled from a normal distribution with mean 0 and variance ϵ_t, $\eta_t \sim \mathcal{N}(0, \epsilon_t)$.

This update rule enables SGLD to balance exploration and exploitation in the parameter space, allowing for more robust and informative inference in complex models.

3 Methodology

[1,] Pretrained language models have demonstrated impressive abilities in generating fluent texts by factorizing a string-values distribution $p_{LM}(\boldsymbol{x})(\boldsymbol{w} \in \sum^*)$ over some vocabulory $\sum$ with local normalization:

$$p_{LM}(\boldsymbol{w}) = p_{LM}(EOS|\boldsymbol{w}) \prod_{n=1}^{N} p_{LM}(w_n|\boldsymbol{w}_{<n}) \tag{1}$$

where EOS is a termination symbol, to define a random variable whose value can be a string, i.e., an element of $\sum^*$, or an infinite sequence.

$$\pi(\boldsymbol{w}) = \frac{1}{Z} exp(-U(\boldsymbol{w})) \tag{2}$$

where $U(\boldsymbol{w})$is called the energy function. The flexibility of this framework lies in the fact that one can refine an existing model by coupling its energy function with arbitrary functions that express the desired attributes of the output text.

$$U(\boldsymbol{w}) = U_{LM}(\boldsymbol{w}) + \sum_{i=1}^{I} U_i(\boldsymbol{w}) \tag{3}$$

where $U_{LM}(\boldsymbol{w}) \stackrel{\text{def}}{=} -logp_{LM}(\boldsymbol{w})$ and each $U_i(\boldsymbol{w})$ measures the extent to which the sequence $\boldsymbol{w}$ satisfies the i^th constraint. This energy function yields a distribution that is related to $p_{LM}(\boldsymbol{w})$ but places more probability mass on the sequences that better satisfy the constraints.

3.1 Problems in Sampling

Consider sampling a sequence of N words $\boldsymbol{w} = w_1 \cdots w_N \in \sum^N$ from the EBM defined by equations (2) and (3). The normalization constant Z is then an intractable sum of $|\sum|^N$ terms. Similarly, the locally normalized conditional probabilities needed for left-to-right autoregressive sampling—which are effectively ratios of normalization constants—are also intractable.
For MCMC, in our situation, the combinitorially large underlying state space $\sum^N$ means that naive MCMC algorithms would have near-zero acceptance rate. Gibbs sampling is also impractical since evaluating $\pi(\cdot|\boldsymbol{w}_{\backslash n})$ in a locally normalized LM would again involve a summation of $|\sum|^N$ terms.

3.2 Gradient-based Sampling

U_{LM} obtained from a pretrained neural LM is differnetiable, as well as possibly the constraint functions U_i. This allows us to use gradient-based MCMC algorithms, such as Langevin dynamics, to sample from the EBM.
However, problems arise when gradient-based sampling algorithms only directly apply to continuous distributions. Therefore, to apply such algorithms to sample from discrete distributions, prior works that developed gradient-based sampling for energybased text generation all focus on continuous relaxations of the underlying discrete space

3.3 Faithfulness of Gradient-based Text Samplers

Consider the setting of sampling a sequence of N words $\boldsymbol{w} = w_1 \cdots w_N \in \sum^N$ from an energy-based sequence model. We denote corresponding word embeddings $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N) \in \mathcal{X} \stackrel{\text{def}}{=} \mathcal{V}^N \subset \mathbb{R}^{Nd}$ where $\mathcal{V} \subset \mathbb{R}^d$ is the disrecrete set of word embeddings.

3.3.1 Constrained Decoding with Langevin Dynamics (COLD)

[4, text] This is a decoding approach that treats text generation as sampling from an energy-based distribution, allowing for flexibly composing constraints based on the task at hand. COLD decoding generates text by sampling from an EBM defined over a sequence of "soft" tokens using Langevin dynamics, then maps the continuous sample into discrete, fluent text.
The set of constraints induces a distribution over text, written in an energy-based form as:

$$p(\boldsymbol{y}) = exp\{\sum_i \lambda_i f_i(\boldsymbol{y})\}/Z$$

where $\lambda_i \geq 0$ is the weight of the i^{th} constraint. Generating text under the constraints can then be seen as sampling from the energy-based distribution $\boldsymbol{y} \sim p(\boldsymbol{y})$.

For efficient sampling from $p(\boldsymbol{y})$ we want to use Langevin dynamics, which makes use of the gradient $\nabla_{\boldsymbol{y}} E(\boldsymbol{y})$. However, in our case $\boldsymbol{y}$ is a discrete sequence and the gradient $\nabla_{\boldsymbol{y}} E(\boldsymbol{y})$ is not well-defined. As a result, we perform Langevin dynamics with an energy defined on a sequence of continuous token vectors.

Instead of defining the energy function on discrete tokens, we define the energy function on a sequence of continuous vectors $\tilde{\boldsymbol{y}} = (\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_T)$, which we call a soft sequence. Each position in the soft sequence is a vector $\tilde{\boldsymbol{y}}_t \in \mathbb{R}^V$, where V is the vocabulary size, and each element $\tilde{\boldsymbol{y}}_t(v) \in \mathbb{R}$ corresponds to the logit of word v in the vocabulary.

Example: Consider a vocabulary of three words: $\mathcal{V} = \{\text{cat}, \text{dog}, \text{bird}\}$. Let $\tilde{\boldsymbol{y}}_t = (2.5, -1.0, 0.8)$ represent the soft sequence at position t. Then:

- $\tilde{y}_t(1) = 2.5$, indicating a high preference for "cat".
- $\tilde{y}_t(2) = -1.0$, indicating a low preference for "dog".

- $\tilde{y}_t(3) = 0.8$, indicating a moderate preference for "bird".

Taking the softmax of $\tilde{\boldsymbol{y}}_t$ yields a distribution over the vocabulary for position t, $\tilde{p}_t^\tau = \text{softmax}(\tilde{\boldsymbol{y}}_t/\tau)$. As $\tau \to 0$, $\tilde{p}_t^\tau$ becomes a one-hot vector, indicating a discrete token.

By specifying an energy $E(\tilde{\boldsymbol{y}})$ on the soft sequence $\tilde{\boldsymbol{y}}$, we can use Langevin dynamics to obtain a sample. Specifically, the sampling is done by forming a Markov chain:

$$\tilde{\boldsymbol{y}}^{(n+1)} \leftarrow \tilde{\boldsymbol{y}}^{(n)} - \eta \nabla_{\tilde{\boldsymbol{y}}} E(\tilde{\boldsymbol{y}}^{(n)}) + \epsilon^{(n)}, \tag{4}$$

where $\eta > 0$ is the step size, and $\epsilon^{(n)} \in \mathcal{N}(0, \sigma)$ is the noise at iteration n. By adding the right amount of noise and annealing the step size, the procedure will converge to samples from the true distribution. That is, if we let $p^{(n)}$ be the distribution such that $\tilde{y}^{(n)} \sim p^{(n)}$, then as $n \to \infty$ and $\sigma \to 0$, we have $p^{(n)} \to p(\tilde{y}) := \exp\{-E(\tilde{y})\}/Z$. That is, the procedure ends up generating samples from the distribution induced by the energy function.

3.3.2 Multiple Constraints from LMs using Langevin Dynamics (MuCoLa)

[2,] This work defines a Markov Chain using gradients of the energy function with respect to token embeddings of the output sequence. Stochasticity is introduced into the process to generate diverse samples, achieved by modifying the gradients with additive noise, a process known as Langevin Dynamics. The energy function is operationalized by setting each constraint to be smaller than a threshold, and expressing it as a Lagrangian—with language model likelihood as the primary objective. This approach allows for the combination of any number of constraints of varying scales without the need for tuning their weights. It is demonstrated that low-energy solutions under this definition are true samples from the LM distribution. This algorithm is referred to as **MuCoLa**, an acronym for sampling with multiple constraints from LMs using Langevin Dynamics.

Let $P(\boldsymbol{y}|\boldsymbol{x};\theta)$ model the conditional probability distribution of an output token sequence $\boldsymbol{y} = (y_1, \ldots, y_N)$, given an optional input token sequence $\boldsymbol{x} = (x_1, \ldots, x_M)$ where $x_m, y_n \in V$. We are interested in constrained sampling from P—finding output sequences $\boldsymbol{y}$ that have a high probability under P while minimizing a given set of constraint functions: $\{f_1, \ldots, f_C\}$. We assume that each $f_i : ([\boldsymbol{x}], \boldsymbol{y}) \to \mathbb{R}$ is defined such that a lower value of f_i implies that the output better satisfies the constraint.

Instead of representing each target token y_n as a soft representation over the vocabulary $\tilde{y}_n \in \mathbb{R}^{|V|}$, we represent it as $\tilde{e}_n \in \boldsymbol{E}$, where $\boldsymbol{E}$ denotes the embedding table of the underlying language model containing $|V|$ vectors of size d, with $d \ll |V|$. We denote this sequence of embeddings as $\tilde{\boldsymbol{e}} = \{\tilde{e}_1, \ldots, \tilde{e}_N\}$.

At an update step t, instead of feeding each $\tilde{\boldsymbol{y}}$ to the model(s) (which are then transformed into an embedding to be fed to the first layer), we directly feed $\tilde{\boldsymbol{e}}$ to the first layer to compute the energy function, now defined as a function of

embeddings instead of tokens. In the case of deterministic minimization, these vectors are updated as:

$$\tilde{\boldsymbol{e}}_t = \mathrm{Proj}_{\boldsymbol{E}}\left(\tilde{\boldsymbol{e}}^{t-1} - \eta \nabla_{\tilde{\boldsymbol{e}}} \mathcal{E}(\tilde{\boldsymbol{e}}^{t-1})\right)$$

where $\mathrm{Proj}_{\boldsymbol{E}}(\hat{e}) = \arg\min_{e \in \boldsymbol{E}} \|e - \hat{e}\|^2$ denotes a projection operation on the embedding table $\boldsymbol{E}$. In other words, after every gradient step, we project each updated vector back to a quantized space, that is the embedding table, using Euclidean distance as the metric. This projection is done to prevent adversarial solutions.

3.4 Toy Example: Energy-Based Language Model

Consider a toy energy-based language model (LM) over a sequence of N tokens, with a binary vocabulary and a one-dimensional embedding. Let the vocabulary be $\mathcal{V} = \Sigma = \{-1, +1\}$.

We define an energy function in the form:

$$U(\boldsymbol{x}) = -\beta \left(\frac{1}{2} \boldsymbol{x}^T A \boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} \right)$$

where $\boldsymbol{x} \in \mathcal{V}^N$. The probability distribution of the model is given by:

$$\pi_{toy}(\boldsymbol{x}) \propto \exp(-U(\boldsymbol{x}))$$

Here, A is the adjacency matrix of an N-cycle. Specifically, A is a symmetric $N \times N$ matrix with $A_{i,i+1} = A_{i+1,i} = 1$ for $i = 1, \ldots, N-1$, and $A_{1,N} = A_{N,1} = 1$, with all other elements being zero. This structure models the interaction between adjacent tokens in the sequence. The vector $\boldsymbol{b}$ is set to zero ($\boldsymbol{b} = \boldsymbol{0}$).

The final form of the energy function, considering the specific form of A and $\boldsymbol{b} = \boldsymbol{0}$, is:

$$U(\boldsymbol{x}) = -\frac{\beta}{2} \sum_{i=1}^{N} (x_i x_{i+1} + x_N x_1)$$

where x_i is the i-th component of $\boldsymbol{x}$ and indices are taken modulo N.

This setup corresponds to a linear-chain Ising model with zero magnetic field. The Ising model is a statistical mechanical model used to describe ferromagnetism in statistical physics. In our context, the model describes a sequence of tokens (spins) where each token interacts only with its immediate neighbors. The absence of an external magnetic field (represented by $\boldsymbol{b} = \boldsymbol{0}$) simplifies the model, focusing on the interactions between adjacent tokens.

The energy function $U(\boldsymbol{x})$ is differentiable with respect to β, as it consists of linear and quadratic terms in $\boldsymbol{x}$. This differentiability is crucial for applying optimization algorithms and gradient-based learning methods.

The reasons for choosing this model are:

1. Differentiability of the energy function enables the application of gradient-based algorithms.

2. The model allows for exact computation of the distribution for small N.

3. The binary vocabulary simplifies the computation of the transition matrix of MuCoLa exactly and the stationary distribution as well.

4 Results

5 Discussion

6 Conclusion

References

[1] Li Du, Afra Amini, Lucas Torroba Hennigen, Xinyan Velocity Yu, Jason Eisner, Holden Lee, and Ryan Cotterell. Principled gradient-based markov chain monte carlo for text generation, 2023.

[2] Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. Gradient-based constrained sampling from language models, 2022.

[3] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. Energy-Based Models. In *Predicting Structured Data*. The MIT Press, 07 2007.

[4] Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics, 2022.