Searching for Activation Functions in Langevin Energy Based Models

February 2021

1 Group Name and Members

Group Name: Quantum Robots
Group Members:
Andrew Nader, Student Number: 1007369107
Abhishek Goudar, Student Number: 1004874852

2 TLDR

Energy based models (EBMs) have many attractive features for Machine Learning, but they suffer from heavy training instability. The authors in [DM19] note that Swish, an activation function found by an automated algorithm for discriminative networks, offers much better training stability. We thus propose to evolve activation functions specifically for EBMs and report on potential training advantages.

3 Extended Abstract

3.1 Summary of Technical Aspect

Energy Based Models are much more flexible than probabilistic models, since they do not explicitly deal with probability distributions that integrate to one, but rather output unnormalized negative log probabilities. This means that the problem is essentially a non-linear regression problem, which allows us to take advantage of the stunning advances made in that domain by modern neural networks such as graph neural networks. In addition, Energy Based Models have many other attractive features for generative modelling, such as the fact that they are easily composable, unlike Generative Adversarial Networks. At first glance, EBMs appear to be the ideal method for generative ML, but unfortunately, training them and sampling from them is still a challenge. Recent work such as [SK21] proposes multiple tips to deal with this problem, but the primary paper of interest in our project is the one in [DM19]. The authors combine multiple modern techniques with an iterative refinement process based on Langevin Dynamics, which is a kind of procedure that performs noisy gradient descent to reach low energy configurations. We explain the idea behind Langevin dynamics and the need for it in Section 3.4, but for now, we will just take the fact that Langevin dynamics can allow for better training of EBMs for granted. In this paper, the authors combine Langevin dynamics with other techniques, the most important being spectral normalization and keeping a sample replay buffer. The sample replay buffer $\mathcal{B}$ is simply a memory containing all of the previously generated samples, and its role is to provide a good initial value for Langevin Dynamics. The initial value for Langevin Dynamics is sampled from $\mathcal{B}$ with 95% probability, and from uniform noise with 5% probability. This has the advantage of continuting to refine past samples and increasing sample diversity. Spectral normalization is used to bound the Lipschitz constant of the network f, where the Lipschitz constant of f is defined as being the smallest constant M such that $\forall x_1, x_2$, $\frac{||f(x_1)-f(x_2)||}{||x_1-x_2||} \leq m$, with the norm being the L_2 norm. This is useful for stabilizing training, and it can be achieved by dividing the weight matrices by their spectral norm and then rescaling to get the desired Lipschitz constant.

The authors found that these techniques allowed for much better training of EBMs and achieved good results, but a particularly interesting note was given at the end of their paper: the Swish activation function [RZL17] achieves much better training stability than more traditional functions like the ReLU. Swish was found by an automated machine learning algorithm, but the focus of the Swish paper was on discriminative networks, not EBMs. This raises a natural question: is it possible to search for activation functions that are specifically designed to improve the performance of EBMs trained by Langevin Dynamics? We believe that this is a nice project to look into, as it tackles an unanswered question, it specifically deals with models that are of high importance to this course, and it is feasible to tackle within the short time-frame that we have. The actual search algorithm is not that important, since the focus of our work is on improving EBMs and not designing a Neural Architecture Search algorithm, but we will use Genetic Programming as tree based structures seem to be a natural representation for activation functions. However, we do not make any claim that GP is the optimal method: Bayesian Optimization, Random Search or any other method could be replaced without affecting the thesis of the paper, since, again, we are focusing on improving the performance of EBMs and the GP part is just an arbitrary tool that helps us achieve that. Genetic Programming is conceptually very simple: it keeps a population of candidate solutions represented as trees (in our case, the solutions are activation functions), and it iteratively generates new candidate solutions by a *crossover* operation, which "mixes" solutions in some way, such as exchanging sub-trees, and a *mutation* operation, which randomly perturbs a candidate solution in some way such as deleting a node.

We give an example of the crossover operation in figures 1 and 2. We first show the plots for the ReLU and two arbitrary unnamed functions $f_1(x)$ and $g_1(x)$ in Figure 1. These functions are combined in a genetic programming tree to define the activation functions $f(x) = \max(\text{relu}(x), f_1(x))$ and $g(x) = \min(f_1(x), g_1(x))$ in figures 2a and 2b, respectively. $f(x)$ and $g(x)$ then exchange their left sub-trees rooted at the middle level to get two new functions $c_1(x)$ and $c_2(x)$, where $c_1(x)$ is shown in Figure 2c. This is the crossover operation, and any mutation operator is similarly easy to visualize. The probability of a tree being chosen to undergo crossover and mutation in order to generate new candidate solutions is proportional to its performance, and thus the hope is that good candidate solutions can be

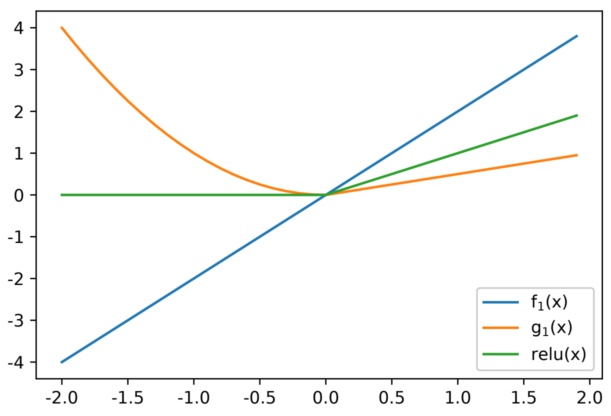
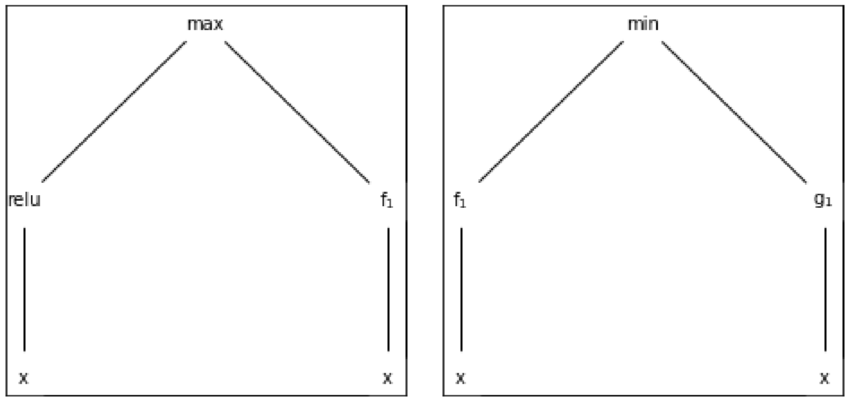
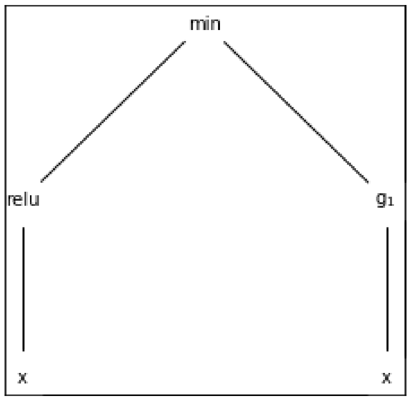


Figure 1: Plot of f_1, g_1 and ReLU.



(a) Activation Function Tree f(x). (b) Activation Function Tree g(x).



(c) Activation Function Tree c_1(x).

Figure 2: Activation Function Trees f(x), g(x) and c_1(x). c_1(x) is one of the trees resulting from crossover between f(x) and g(x).

combined or perturbed to generate even better candidate solutions.

3.2 Project Deliverables

First of all, we will implement the GP algorithm using the Distributed Evolutionary Algorithms in Python (DEAP) library [For+12]. Then, we will combine this algorithm with the source code given by the authors of the paper in [DM19] to try to evolve activation functions that improve the performance of energy based models that make use of the techniques described in the paper. The improvement is defined as achieving a better Frechet Inception Distance on image generation tasks, or better area under the ROC scores on out of distribution generalization tasks. Due to time constraints, we may have to use a subset of the full datasets used in [DM19] (such as the CIFAR-10, SVHN, and textures datasets) and so our project can be seen as preliminary work towards improving EBMs through the use of better activation functions rather than a full fledged research endeavor. We will start our report by reviewing EBMs and the various training methods proposed in the literature with a focus on explaining the advantages of using Langevin Dynamics, along with any potential pitfalls. We will then move on to explaining GP and our particular implementation of the GP algorithm for activation functions. Following that, we will move on the important part of the project and produce a full analysis comparing our evolved EBMs to EBMs with the Swish and ReLU activation functions as a baseline, along with the subsets of datasets used and a table of results for each dataset. In addition, we will analyze the shape of the functions evolved, and see if they differ from run to run or if there are any similarities, which might indicate that a particular shape of activation function is beneficial to EBMs. Finally, we will make our code and results publicly available on Github for reproducibility, and we will also provide a folder full of the data points generated by our EBMs.

3.3 Nice-to-haves

If we find that we have enough time, we would try evolving activation functions for EBMs trained with other methods and other MCMC variants. If we find that we are not able to evolve better functions, then we would try to conduct an analysis into what makes the Swish or ReLU variants a good choice for EBMs.

3.4 Related work

As alluded to in Section 3.1, EBMs are much more flexible than probabilistic models in terms of the functional forms allowed. As such, there has been extensive research on the application of EBMs, and a good cursory introduction to EBMs can be found in [LeC+06]. Given the generality of EBMs, any nonlinear function can be used as an energy function (as long as it is integrable), and in practice, this energy function is usually implemented as a neural network. One usually wishes to perform Maximum Likelihood training on the EBM, but the problem comes from the fact that the density function associated with an EBM, which is called the Boltzmann distribution, contains an intractable term. The Boltzmann distribution associated with the energy function $E_\theta(\mathbf{x})$, parameterized by θ, is defined as:

$$p(\theta) = \frac{e^{-E_\theta(\mathbf{x})}}{Z_\theta}, \tag{1}$$

where Z_θ is the *partition* function given by:

$$Z_\theta = \int e^{-E_\theta(\mathbf{x})}\mathbf{dx}. \tag{2}$$

The gradient of the expected log-likelihood $\log p_\theta(\mathbf{x})$ w.r.t θ is:

$$\mathbb{E}_{x\sim p_\theta(\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x})] = \mathbb{E}_{x\sim p_\theta(\mathbf{x})}[\nabla_\theta E_\theta(\mathbf{x})] + \nabla_\theta \log Z_\theta.$$

While $\nabla_\theta E_\theta(\mathbf{x})$ can be found using Automatic Differentiation, calculation of $\nabla_\theta \log Z_\theta$ is difficult because the evaluation and differentiation of Z_θ is generally intractable. Remedies involve using Markov chain Monte Carlo (MCMC) techniques to approximate the gradient by sampling from $p_\theta(x)$:

$$\nabla_\theta \log Z_\theta = \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[-\nabla_\theta E_\theta(\mathbf{x})] \tag{3}$$

This is easier said than done, and in addition, MCMC methods are known to suffer from long mixing times, so much of the literature has focused on devising better MCMC methods. One way of doing that is by relying on Langevin Dynamics [SK21; DM19], which defines the following iterative procedure:

$$x^k = x^{k-1} - \frac{\lambda}{2}\nabla_x \mathbb{E}_\theta(x^{k-1}) + \omega^k, w^k \sim \mathcal{N}(0, \lambda) \tag{4}$$

and this iterative procedure defines a distribution q_θ in the sense that $x^k \sim q_\theta$. Moreover,as $K \to \infty$ and $\lambda \to 0$, q_θ converges to the Boltzmann distribution defined by the energy function, which means that the energy function is generating samples *implicitly* here.

Langevin Dynamics is used to compute an approximation of the gradient of the negative log likelihood of the data $\nabla_\theta \mathcal{L}_{ML}$, through the formula

$$\nabla_\theta \mathcal{L}_{ML} \approx \mathbb{E}_{x^+\sim p_D}[\nabla_\theta \mathbb{E}_\theta(x^+)] - \mathbb{E}_{x^-\sim q_\theta}[\nabla_\theta \mathbb{E}_\theta(x^-)] \tag{5}$$

with x^+ and x^- being the positive and negative energy samples, respectively. Other MCMC variants such as Contrastive Divergence (CD) or Score Matching techniques can be used [SK21]. However, CD uses truncated MCMC to reduce mixing times, resulting in biased gradients and hurting the learning dynamics. In this project, we focus mainly on training based on Langevin Dynamics.

References

[LeC+06] Yann LeCun et al. “A tutorial on energy-based learning”. In: *PREDICTING STRUCTURED DATA*. MIT Press, 2006.

[For+12] Félix-Antoine Fortin et al. “DEAP: Evolutionary algorithms made easy”. In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 2171–2175.

[RZL17] Prajit Ramachandran, Barret Zoph, and Quoc V Le. “Searching for activation functions”. In: *arXiv preprint arXiv:1710.05941* (2017).

[DM19] Yilun Du and Igor Mordatch. "Implicit generation and generalization in energy-based models". In: *arXiv preprint arXiv:1903.08689* (2019).

[SK21] Yang Song and Diederik P Kingma. "How to Train Your Energy-Based Models". In: *arXiv preprint arXiv:2101.03288* (2021).