# Bridging the Gap between Stochastic Gradient MCMC and Stochastic Optimization

**Changyou Chen**[†] **David Carlson**[‡] **Zhe Gan**[†] **Chunyuan Li**[†] **Lawrence Carin**[†]
[†]Department of Electrical and Computer Engineering, Duke University
[‡]Department of Statistics and Grossman Center for Statistics of Mind, Columbia University

## Abstract

Stochastic gradient Markov chain Monte Carlo (SG-MCMC) methods are Bayesian analogs to popular stochastic optimization methods; however, this connection is not well studied. We explore this relationship by applying simulated annealing to an SG-MCMC algorithm. Furthermore, we extend recent SG-MCMC methods with two key components: *i*) adaptive preconditioners (as in ADAgrad or RMSprop), and *ii*) adaptive element-wise momentum weights. The zero-temperature limit gives a novel stochastic optimization method with *adaptive element-wise* momentum weights, while conventional optimization methods only have a shared, static momentum weight. Under certain assumptions, our theoretical analysis suggests the proposed simulated annealing approach converges close to the global optima. Experiments on several deep neural network models show state-of-the-art results compared to related stochastic optimization algorithms.

## 1 Introduction

Machine learning has made significant recent strides due to large-scale learning applied to "big data". Large-scale learning is typically performed with stochastic optimization, and the most common method is stochastic gradient descent (SGD) (Bottou, 2010). Stochastic optimization methods are devoted to obtaining a (local) optima of an objective function. Alternatively, Bayesian methods aim to compute the expectation of a test function over the posterior distribution. At first glance, these methods appear to be distinct, independent approaches to learning. However, even the celebrated Gibbs sampler was first introduced to statistics as a simulated annealing method for *maximum a posteriori* estimation (*i.e.,* finding an optima) (Geman and Geman, 1984).

Recent work on large-scale Bayesian learning has focused on incorporating the speed and low-memory costs from stochastic optimization. These approaches are referred to as stochastic gradient Markov chain Monte Carlo (SG-MCMC) methods. Well-known SG-MCMC methods include stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011), stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014), and stochastic gradient thermostats (SGNHT) (Ding et al., 2014). SG-MCMC has become increasingly popular in the literature due to practical successes, ease of implementation, and theoretical convergence properties (Teh et al., 2014; Vollmer et al., 2015; Chen et al., 2015).

There are obvious structural similarities between SG-MCMC algorithms and stochastic optimization methods. For example, SGLD resembles SGD with additive Gaussian noise. SGHMC resembles SGD with momentum (Rumelhart et al., 1986), adding additive Gaussian noise when updating the momentum terms (Chen et al., 2014). These similarities are detailed in Section 2. Despite these structural similarities, the theory is unclear on how additive Gaussian noise differentiates a Bayesian algorithm from its optimization analog.

Just as classical sampling methods were originally used for optimization (Geman and Geman, 1984), we directly address using SG-MCMC algorithms for optimization. A major benefit of adapting these schemes is that Bayesian learning is (in theory) able to fully explore the parameter space. Thus it may find a better local optima, if not the global optima, for a non-convex objective function.

Specifically, in this work we first extend the recently proposed multivariate stochastic gradient thermostat

Appearing in Proceedings of the $19^{th}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 41. Copyright 2016 by the authors.

algorithm (Gan et al., 2015) with Riemannian information geometry, which results in an adaptive preconditioning and momentum scheme with analogs to Adam (Kingma and Ba, 2015) and RMSprop (Tieleman and Hinton, 2012). We propose an annealing scheme on the system temperature to move from a Bayesian method to a stochastic optimization method. We call the proposed algorithm Stochastic AnNealing Thermostats with Adaptive momentum (Santa). We show that in the temperature limit, Santa recovers the SGD with momentum algorithm except that: *i*) adaptive preconditioners are used when updating both model and momentum parameters; *ii*) each parameter has an individual, *learned* momentum parameter. Adaptive preconditioners and momentums are desirable in practice because of their ability to deal with uneven, dynamic curvature (Dauphin et al., 2015). For completeness, we first review related algorithms in Section 2, and present our novel algorithm in Section 3.

We develop theory to analyze convergence properties of our algorithm, suggesting that Santa is able to find a solution for an (non-convex) objective function close to its global optima, shown in Section 4. The theory is based on the analysis from stochastic differential equations (Teh et al., 2014; Chen et al., 2015), and presents results on bias and variance of the annealed Markov chain. This is a fundamentally different approach from the traditional convergence explored in stochastic optimization, or the regret bounds used in online optimization. We note we can adapt the regret bound of Adam (Kingma and Ba, 2015) for our zero-temperature algorithm (with a few trivial modifications) for a convex problem, as shown in Supplementary Section F. However, this neither addresses non-convexity nor the annealing scheme that our analysis does.

In addition to theory, we demonstrate effective empirical performance on a variety of deep neural networks (DNNs), achieving the best performance compared to all competing algorithms for the same model size. This is shown in Section 5. The code is publicly available at https://github.com/cchangyou/Santa.

## 2 Preliminaries

Throughout this paper, we denote vectors as bold, lower-case letters, and matrices as bold, upper-case letters. We use $\odot$ for element-wise multiplication, and $\oslash$ as element-wise division; $\sqrt{\cdot}$ denotes the element-wise square root when applied to vectors or matricies. We reserve $(\cdot)^{1/2}$ for the standard matrix square root. $\mathbf{I}_p$ is the $p \times p$ identity matrix, $\mathbf{1}$ is an all-ones vector.

The goal of an optimization algorithm is to minimize an objective function $U(\boldsymbol{\theta})$ that corresponds to a (non-convex) model of interest. In a Bayesian model, this corresponds to the potential energy defined as the negative log-posterior, $U(\boldsymbol{\theta}) \triangleq -\log p(\boldsymbol{\theta}) - \sum_{n=1}^{N} \log p(\mathbf{x}_n \,|\boldsymbol{\theta})$. Here $\boldsymbol{\theta} \in \mathbf{R}^p$ are the model parameters, and $\{\mathbf{x}_n\}_{n=1,\dots,N}$ are the $d$-dimensional observed data; $p(\boldsymbol{\theta})$ corresponds to the prior and $p(\mathbf{x}_n \,|\boldsymbol{\theta})$ is a likelihood term for the $n^{th}$ observation. In optimization, $-\sum_{n=1}^{N} \log p(\mathbf{x}_n \,|\boldsymbol{\theta})$ is typically referred to as the loss function, and $-\log p(\boldsymbol{\theta})$ as a regularizer.

In large-scale learning, $N$ is prohibitively large. This motivates the use of stochastic approximations. We denote the stochastic approximation $\tilde{U}_t(\boldsymbol{\theta}) \triangleq -\log p(\boldsymbol{\theta}) - \frac{N}{m}\sum_{j=1}^{m} \log p(\mathbf{x}_{i_j} \,|\boldsymbol{\theta})$, where $(i_1, \cdots, i_m)$ is a random subset of the set $\{1, 2, \cdots, N\}$. The gradient on this minibatch is denoted as $\tilde{\mathbf{f}}_t(\boldsymbol{\theta}) = \nabla \tilde{U}_t(\boldsymbol{\theta})$, which is an unbiased estimate of the true gradient.

A standard approach to learning is SGD, where parameter updates are given by $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \tilde{\mathbf{f}}_{t-1}(\boldsymbol{\theta})$ with $\eta_t$ the learning rate. This is guaranteed to converge to a local minima under mild conditions (Bottou, 2010). The SG-MCMC analog to this is SGLD, with updates $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \tilde{\mathbf{f}}_{t-1}(\boldsymbol{\theta}) + \sqrt{2\eta_t}\boldsymbol{\zeta}_t$. The additional term is a standard normal random vector, $\boldsymbol{\zeta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$ (Welling and Teh, 2011). The SGLD method draws approximate posterior samples instead of obtaining a local minima.

Using momentum in stochastic optimization is important in learning deep models (Sutskever et al., 2013). This motivates SG-MCMC algorithms with momentum. The standard SGD with momentum (SGD-M) approach introduces an auxiliary variable $\boldsymbol{u}_t \in \mathbf{R}^p$ to represent the momentum. Given a momentum weight $\alpha$, the updates are $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \eta_t \boldsymbol{u}_t$ and $\boldsymbol{u}_t = (1-\alpha)\boldsymbol{u}_{t-1} - \tilde{\mathbf{f}}_{t-1}(\boldsymbol{\theta})$. A Bayesian analog is SGHMC (Chen et al., 2014) or multivariate SGNHT (mSGNHT) (Gan et al., 2015). In mSGNHT, each parameter has a unique momentum weight $\boldsymbol{\alpha}_t \in \mathbf{R}^p$ that is learned during the sampling sequence. The momentum weights are updated to maintain the system temperature $1/\beta$. An inverse temperature of $\beta = 1$ corresponds to the posterior. This algorithm has updates $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \eta_t \boldsymbol{u}_t$, $\boldsymbol{u}_t = (\mathbf{1} - \eta_t \boldsymbol{\alpha}_{t-1}) \odot \boldsymbol{u}_{t-1} - \eta_t \tilde{\mathbf{f}}_{t-1}(\boldsymbol{\theta}) + \sqrt{2\eta_t/\beta}\boldsymbol{\zeta}_t$. The main difference is the additive Gaussian noise and step-size dependent momentum update. The weights have updates $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \eta_t((\boldsymbol{u}_t \odot \boldsymbol{u}_t) - \mathbf{1}/\beta)$, which matches the kinetic energy to the system temperature.

A recent idea in stochastic optimization is to use an adaptive preconditioner, also known as a variable metric, to improve convergence rates. Both ADAgrad (Duchi et al., 2011) and Adam (Kingma and Ba, 2015) adapt to the local geometry with a regret bound of $\mathcal{O}(\sqrt{N})$. Adam adds momentum as well through moment smoothing. RMSprop (Tieleman and Hinton, 2012), Adadelta (Zeiler, 2012), and RMSspectral (**?**)

**Algorithm 1:** Santa with the Euler scheme

**Input**: $\eta_t$ (learning rate), $\sigma$, $\lambda$, *burnin*, $\beta = \{\beta_1, \beta_2, \cdots\} \to \infty$, $\{\boldsymbol{\zeta}_t \in \mathbf{R}^p\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$.
Initialize $\boldsymbol{\theta}_0$, $\mathbf{u}_0 = \sqrt{\eta} \times \mathcal{N}(0, I)$, $\boldsymbol{\alpha}_0 = \sqrt{\eta} C$, $\mathbf{v}_0 = 0$ ;
**for** $t = 1, 2, \ldots$ **do**
  Evaluate $\tilde{\mathbf{f}}_t \triangleq \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{t-1})$ on the $t^{\text{th}}$ mini-batch;
  $\mathbf{v}_t = \sigma\, \mathbf{v}_{t-1} + \frac{1-\sigma}{m^2} \tilde{\mathbf{f}}_t \odot \tilde{\mathbf{f}}_t$ ;
  $\mathbf{g}_t = 1 \oslash \sqrt{\lambda + \sqrt{\mathbf{v}_t}}$ ;
  **if** $t <$ *burnin* **then**
    /* *exploration* */
    $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + (\mathbf{u}_{t-1} \odot \mathbf{u}_{t-1} - \eta/\beta_t)$;
    $\mathbf{u}_t = \frac{\eta}{\beta_t}\left(1 - \mathbf{g}_{t-1} \oslash \mathbf{g}_t\right) \oslash \mathbf{u}_{t-1} + \sqrt{\frac{2\eta}{\beta_t}\, \mathbf{g}_{t-1}} \odot \boldsymbol{\zeta}_t$
  **else**
    /* *refinement* */
    $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}$;  $\boldsymbol{u}_t = \mathbf{0}$;
  **end**
  $\mathbf{u}_t = \mathbf{u}_t + (1 - \boldsymbol{\alpha}_t) \odot \mathbf{u}_{t-1} - \eta\, \mathbf{g}_t \odot \tilde{\mathbf{f}}_t$;
  $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{g}_t \odot \mathbf{u}_t$;
**end**

are similar methods with preconditioners. Our method introduces adaptive momentum and preconditioners to the SG-MCMC. This differs from stochastic optimization in implementation and theory, and is novel in SG-MCMC.

Simulated annealing (Kirkpatrick et al., 1983; Černý, 1985) is well-established as a way of acquiring a local mode by moving from a high-temperature, flat surface to a low-temperature, peaky surface. It has been explored in the context of MCMC, including reversible jump MCMC (Andrieu et al., 2000), annealed important sampling (Neal, 2001) and parallel tempering (Li et al., 2009). Traditional algorithms are based on Metropolis–Hastings sampling, which require computationally expensive accept-reject steps. Recent work has applied simulated annealing to large-scale learning through mini-batch based annealing (van de Meent et al., 2014; Obermeyer et al., 2014). Our approach incorporates annealing into SG-MCMC with its inherent speed and mini-batch nature.

## 3 The Santa Algorithm

Santa extends the mSGNHT algorithm with preconditioners and a simulated annealing scheme. A simple pseudocode is shown in Algorithm 1, or a more complex, but higher accuracy version, is shown in Algorithm 2, and we detail the steps below.

The first extension we consider is the use of adaptive preconditioners. Preconditioning has been proven critical for fast convergence in both stochastic optimization (Dauphin et al., 2015) and SG-MCMC algorithms (Patterson and Teh, 2013). In the MCMC literature, preconditioning is alternatively referred to as *Riemannian information geometry* (Patterson and Teh, 2013). We denote the preconditioner as $\{\mathbf{G}_t \in \mathbf{R}^{p \times p}\}$. A popular choice in SG-MCMC is the Fisher information matrix (Girolami and Calderhead, 2011). Unfortunately, this approach is computationally prohibitive for many models of interest. To avoid this problem, we adopt the preconditioner from RMSprop and Adam, which uses a vector $\{\mathbf{g}_t \in \mathbf{R}^p\}$ to approximate the diagonal of the Fisher information matrixes (**?**). The construction sequentially updates the preconditioner based on current and historical gradients with a smoothing parameter $\sigma$, and is shown as part of Algorithm 1. While this approach will not capture the Riemannian geometry as effectively as the Fisher information matrix, it is computationally efficient.

Santa also introduces an annealing scheme on system temperatures. As discussed in Section 2, mSGNHT naturally accounts for a varying temperature by matching the particle momentum to the system temperature. We introduce $\beta = \{\beta_1, \beta_2, \cdots\}$, a sequence of inverse temperature variables with $\beta_i < \beta_j$ for $i < j$ and $\lim_{i \to \infty} \beta_i = \infty$. The infinite case corresponds to the zero-temperature limit, where SG-MCMCs become deterministic optimization methods.

The annealing scheme leads to two stages: the *exploration* and the *refinement* stages. The *exploration* stage updates all parameters based on an annealed sequence of stochastic dynamic systems (see Section 4 for more details). This stage is able to explore the parameter space efficiently, escape poor local modes, and finally converge close to the global mode. The *refinement* stage corresponds to the temperature limit, *i.e.*, $\beta_n \to \infty$. In the temperature limit, the momentum weight updates vanish and it becomes a stochastic optimization algorithm.

We propose two update schemes to solve the corresponding stochastic differential equations: the Euler scheme and the symmetric splitting scheme (SSS). The Euler scheme has simpler updates, as detailed in Algorithm 1; while SSS endows increased accuracy (Chen et al., 2015) with a slight increase in overhead computation, as shown in Algorithm 2. Section 4.1 elaborates on the details of these two schemes. We recommend the use of SSS, but the Euler scheme is simpler to implement and compare to known algorithms.

**Practical considerations** According to Section 4, the *exploration* stage helps the algorithm traverse the parameter space following the posterior curve as accurate as possible. For optimization, slightly biased samples do not affect the final solution. As a result,

**Algorithm 2:** Santa with SSS

**Input**: $\eta_t$ (learning rate), $\sigma$, $\lambda$, *burnin*, $\beta = \{\beta_1, \beta_2, \cdots\} \to \infty$, $\{\boldsymbol{\zeta}_t \in \mathbf{R}^p\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$.
Initialize $\boldsymbol{\theta}_0$, $\mathbf{u}_0 = \sqrt{\eta} \times \mathcal{N}(0, I)$, $\boldsymbol{\alpha}_0 = \sqrt{\eta}C$, $\mathbf{v}_0 = 0$ ;
**for** $t = 1, 2, \ldots$ **do**
  Evaluate $\tilde{\mathbf{f}}_t \triangleq \nabla_{\boldsymbol{\theta}} \tilde{U}(\boldsymbol{\theta}_{t-1})$ on the $t^{\text{th}}$ mini-batch;
  $\mathbf{v}_t = \sigma\, \mathbf{v}_{t-1} + \frac{1-\sigma}{m^2} \tilde{\mathbf{f}}_t \odot \tilde{\mathbf{f}}_t$ ;
  $\mathbf{g}_t = 1 \oslash \sqrt{\lambda + \sqrt{\mathbf{v}_t}}$ ;
  $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{g}_t \odot \mathbf{u}_{t-1}\, /2$;
  **if** $t <$ *burnin* **then**
    /* *exploration* */
    $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + (\mathbf{u}_{t-1} \odot \mathbf{u}_{t-1} - \eta/\beta_t)\, /2$;
    $\mathbf{u}_t = \exp(-\boldsymbol{\alpha}_t/2) \odot \mathbf{u}_{t-1}$;
    $\mathbf{u}_t = \mathbf{u}_t - \mathbf{g}_t \odot \tilde{\mathbf{f}}_t \eta + \sqrt{2\, \mathbf{g}_{t-1}\, \eta/\beta_t} \odot \boldsymbol{\zeta}_t + \eta/\beta_t \left(1 - \mathbf{g}_{t-1} \oslash \mathbf{g}_t\right) \oslash \mathbf{u}_{t-1}$;
    $\mathbf{u}_t = \exp(-\boldsymbol{\alpha}_t/2) \odot \mathbf{u}_t$;
    $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_t + (\mathbf{u}_t \odot \mathbf{u}_t - \eta/\beta_t)\, /2$;
  **else**
    /* *refinement* */
    $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1}$; $\mathbf{u}_t = \exp(-\boldsymbol{\alpha}_t/2) \odot \mathbf{u}_{t-1}$;
    $\mathbf{u}_t = \mathbf{u}_t - \mathbf{g}_t \odot \tilde{\mathbf{f}}_t \eta$; $\mathbf{u}_t = \exp(-\boldsymbol{\alpha}_t/2) \odot \mathbf{u}_t$;
  **end**
  $\boldsymbol{\theta}_t = \boldsymbol{\theta}_t + \mathbf{g}_t \odot \mathbf{u}_t\, /2$;
**end**

the term consisting of $(1 - \mathbf{g}_{t-1} \oslash \mathbf{g}_t)$ in the algorithm (which is an approximation term, see Section 4.1) is ignored. We found no decreasing performance in our experiments. Furthermore, the term $\mathbf{g}_{t-1}$ associated with the Gaussian noise could be replaced with a fixed constant without affecting the algorithm.

## 4 Theoretical Foundation

In this section we present the stochastic differential equations (SDEs) that correspond to the Santa algorithm. We first introduce the general SDE framework, then describe the *exploration* stage in Section 4.1 and the *refinement* stage in Section 4.2. We give the convergence properties of the numerical scheme in Section 4.3. This theory uses tools from the SDE literature and extends the mSGNHT theory (Ding et al., 2014; Gan et al., 2015).

The SDEs are presented with re-parameterized $\mathbf{p} = \mathbf{u}/\eta^{1/2}$, $\boldsymbol{\Xi} = \text{diag}(\boldsymbol{\alpha})/\eta^{1/2}$, as in Ding et al. (2014). The SDEs describe the motion of a particle in a system where $\boldsymbol{\theta}$ is the location and $\mathbf{p}$ is the momentum.

In mSGNHT, the particle is driven by a force $-\nabla_{\boldsymbol{\theta}} \tilde{U}_t(\boldsymbol{\theta})$ at time $t$. The stationary distribution of $\boldsymbol{\theta}$ corresponds to the model posterior (Gan et al., 2015). Our critical extension is the use of Riemannian information geometry, important for fast convergence (Patterson and Teh, 2013). Given an inverse temperature $\beta$, the system is described by the following SDEs[1]:

$$\begin{cases} \mathrm{d}\boldsymbol{\theta} &= G_1(\boldsymbol{\theta})\, \mathbf{p}\, \mathrm{d}t \\ \mathrm{d}\,\mathbf{p} &= \Big(-G_1(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) - \boldsymbol{\Xi}\, \mathbf{p} + \frac{1}{\beta} \nabla_{\boldsymbol{\theta}} G_1(\boldsymbol{\theta}) \\ &\quad + G_1(\boldsymbol{\theta})(\boldsymbol{\Xi} - G_2(\boldsymbol{\theta})) \nabla_{\boldsymbol{\theta}} G_2(\boldsymbol{\theta})\Big)\, \mathrm{d}t + \big(\tfrac{2}{\beta} G_2(\boldsymbol{\theta})\big)^{\frac{1}{2}}\, \mathrm{d}\boldsymbol{w} \\ \mathrm{d}\boldsymbol{\Xi} &= \Big(\mathbf{Q} - \tfrac{1}{\beta} I\Big)\, \mathrm{d}t\ , \end{cases} \tag{1}$$

where $\mathbf{Q} = \text{diag}(\mathbf{p} \odot \mathbf{p})$, $\boldsymbol{w}$ is standard Brownian motion, $\mathbf{G}_1(\boldsymbol{\theta})$ encodes geometric information of the potential energy $U(\boldsymbol{\theta})$, and $\mathbf{G}_2(\boldsymbol{\theta})$ characterizes the manifold geometry of the Brownian motion. Note $\mathbf{G}_2(\boldsymbol{\theta})$ may be the same as $\mathbf{G}_1(\boldsymbol{\theta})$ for the same Riemannian manifold. We call $\mathbf{G}_1(\boldsymbol{\theta})$ and $\mathbf{G}_2(\boldsymbol{\theta})$ Riemannian metrics, which are commonly defined by the Fisher information matrix (Girolami and Calderhead, 2011). We use the RMSprop preconditioner (with updates from Algorithm 1) for computational feasibility. Using the Fokker-Plank equation (Risken, 1989), we show that the marginal stationary distribution of (1) corresponds to the posterior distribution.

**Lemma 1.** *Denote* $\mathbf{A} : \mathbf{B} \triangleq tr\Big\{\mathbf{A}^T\, \mathbf{B}\Big\}$. *The stationary distribution of* (1) *is:* $p_\beta(\boldsymbol{\theta}, \mathbf{p}, \boldsymbol{\Xi}) \propto$

$$e^{-\beta U(\boldsymbol{\theta}) - \frac{\beta}{2}\, \mathbf{p}^T\, \mathbf{p} - \frac{\beta}{2} (\boldsymbol{\Xi} - G_2(\boldsymbol{\theta})) : (\boldsymbol{\Xi} - G_2(\boldsymbol{\theta}))}\ . \tag{2}$$

An inverse temperature $\beta = 1$ corresponds to the standard Bayesian posterior.

We note that $\mathbf{p}$ in (1) has additional dependencies on $\mathbf{G}_1$ and $\mathbf{G}_2$ compared to Gan et al. (2015) that must be accounted for. $\boldsymbol{\Xi}\, \mathbf{p}$ introduces friction into the system so that the particle does not move too far away by the random force; the terms $\nabla_{\boldsymbol{\theta}} G_1(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} G_2(\boldsymbol{\theta})$ penalize the influences of the Riemannian metrics so that the stationary distribution remains invariant.

### 4.1 Exploration

The first stage of Santa, *exploration*, explores the parameter space to obtain parameters near the global mode of an objective function[2]. This approach applies ideas from simulated annealing (Kirkpatrick et al., 1983). Specifically, the inverse temperature $\beta$ is slowly annealed to temperature zero to freeze the particles at the global mode.

Minimizing $U(\boldsymbol{\theta})$ is equivalent to sampling from the zero-temperature limit $p_\beta(\boldsymbol{\theta}) \triangleq \frac{1}{Z_\beta} e^{-\beta U(\boldsymbol{\theta})}$ (propor-

[1] We abuse notation for conciseness. Here, $\nabla_{\boldsymbol{\theta}}\, \mathbf{G}(\boldsymbol{\theta})$ is a vector with the $i$-th element being $\sum_j \nabla_{\boldsymbol{\theta}_j}\, \mathbf{G}_{ij}(\boldsymbol{\theta})$.

[2] This requires an ergodic algorithm. While ergodicity is not straightforward to check, we follow most MCMC work and assume it holds in our algorithm.

tional to (2)), with $Z_\beta$ being the normalization constant such that $p_\beta(\boldsymbol{\theta})$ is a valid distribution. We construct a Markov chain that sequentially transits from high temperatures to low temperatures. At the state equilibrium, the chain reaches the temperature limit with marginal stationary distribution $\rho_0(\boldsymbol{\theta}) \triangleq \lim_{\beta\to\infty} e^{-\beta U(\boldsymbol{\theta})}$, a point mass[3] located at the global mode of $U(\boldsymbol{\theta})$. Specifically, we first define a sequence of inverse temperatures, $(\beta_1, \beta_2, \cdots, \beta_L)$, such that $\beta_L$ is large enough[4]. For each time $t$, we generate a sample according to the SDE system (1) with temperature $\frac{1}{\beta_t}$, conditioned on the sample from the previous temperature, $\frac{1}{\beta_{t-1}}$. We call this procedure annealing thermostats to denote the analog to simulated annealing.

**Generating approximate samples** Generating exact samples from (1) is infeasible for general models. One well-known numerical approach is the Euler scheme in Algorithm 1. The Euler scheme is a 1st-order method with relatively high approximation error (Chen et al., 2015). We increase accuracy by implementing the symmetric splitting scheme (SSS) (Chen et al., 2015; **?**). The idea of SSS is to split an infeasible SDE into several sub-SDEs, where each sub-SDE is analytically solvable; approximate samples are generated by sequentially evolving parameters via these sub-SDEs. Specifically, in Santa, we split (1) into the following three sub-SDEs:

$$A: \begin{cases} \mathrm{d}\boldsymbol{\theta} &= \mathbf{G}_1(\boldsymbol{\theta})\,\mathbf{p}\,\mathrm{d}t \\ \mathrm{d}\,\mathbf{p} &= 0 \\ \mathrm{d}\Xi &= \left(\mathbf{Q} - \frac{1}{\beta} I\right)\mathrm{d}t \end{cases}, B: \begin{cases} \mathrm{d}\boldsymbol{\theta} &= 0 \\ \mathrm{d}\,\mathbf{p} &= -\boldsymbol{\Xi}\,\mathbf{p}\,\mathrm{d}t \\ \mathrm{d}\boldsymbol{\Xi} &= 0 \end{cases},$$

$$O: \begin{cases} \mathrm{d}\boldsymbol{\theta} &= 0 \\ \mathrm{d}\,\mathbf{p} &= \Big(-G_1(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) + \frac{1}{\beta}\nabla_{\boldsymbol{\theta}} G_1(\boldsymbol{\theta}) \\ &+ G_1(\boldsymbol{\theta})(\boldsymbol{\Xi} - G_2(\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}} G_2(\boldsymbol{\theta})\Big)\,\mathrm{d}t + \left(\frac{2}{\beta} G_2(\boldsymbol{\theta})\right)^{\frac{1}{2}} \mathrm{d}\boldsymbol{w} \\ \mathrm{d}\boldsymbol{\Xi} &= 0 \end{cases}$$

We then update the sub-SDEs in order *A-B-O-B-A* to generative approximate samples (Chen et al., 2015). This uses half-steps $h/2$ on the $A$ and $B$ updates[5], and full steps $h$ in the $O$ update. This is analogous to the leapfrog steps in Hamiltonian Monte Carlo (Neal, 2011). Update equations are given in the Supplementary Section A. The resulting parameters then serve as an approximate sample from the posterior distribution with the inverse temperature of $\beta$. Replacing $\mathbf{G}_1$ and $\mathbf{G}_2$ with the RMSprop preconditioners gives Algorithm 2. These updates require approximations to $\nabla_{\boldsymbol{\theta}} G_1(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} G_2(\boldsymbol{\theta})$, addressed below.

[3]The sampler samples a uniform distribution over global modes, or a point mass if the mode is unique. We assume uniqueness and say point mass for clarity henceforth.

[4]Due to numerical issues, it is impossible to set $\beta_L$ to infinity; we thus assign a large enough value for it and handle the infinity case in the *refinement* stage.

[5]As in Ding et al. (2014), we define $h = \sqrt{\eta}$.

**Approximate calculation for $\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_1(\boldsymbol{\theta})$** We propose a computationally efficient approximation for calculating the derivative vector $\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_1(\boldsymbol{\theta})$ based on the definition. Specifically, for the $i$-th element of $\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_1(\boldsymbol{\theta})$ at the $t$-th iteration, denoted as $(\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_1^t(\boldsymbol{\theta}))_i$, it is approximated as:

$$(\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_1^t(\boldsymbol{\theta}))_i \overset{A_1}{\approx} \sum_j \frac{(\mathbf{G}_1^t(\boldsymbol{\theta}))_{ij} - (\mathbf{G}_1^{t-1}(\boldsymbol{\theta}))_{ij}}{\boldsymbol{\theta}_{tj} - \boldsymbol{\theta}_{(t-1)j}}$$
$$\overset{A_2}{=} \sum_j \frac{(\Delta\,\mathbf{G}_1^t)_{ij}}{(\mathbf{G}_1^t(\boldsymbol{\theta})\,\mathbf{p}_{t-1})_j h} = \sum_j \frac{(\Delta\,\mathbf{G}_1^t)_{ij}}{(\mathbf{G}_1^t(\boldsymbol{\theta})\,\mathbf{u}_{t-1})_j}$$

where $\Delta\,\mathbf{G}_1^t \triangleq \mathbf{G}_1^t - \mathbf{G}_1^{t-1}$. Step $A_1$ follows by the definition of a derivative, and $A_2$ by using the update equation for $\boldsymbol{\theta}_t$, *i.e.*, $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{G}_1^t(\boldsymbol{\theta})\,\mathbf{p}_{t-1}\,h$. According to Taylor's theory, the approximation error for the $\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_1(\boldsymbol{\theta})$ is $O(h)$, *e.g.*,

$$\sum_i \left| \sum_j \frac{(\Delta\,\mathbf{G}_1^t)_{ij}}{(\mathbf{G}_1^t(\boldsymbol{\theta})\,\mathbf{u}_{t-1})_j} - (\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_1^t(\boldsymbol{\theta}))_i \right| \le \mathcal{B}_t h\ , \qquad (3)$$

for some positive constant $\mathcal{B}_t$. The approximation error is negligible in term of convergence behaviors because it can be absorbed into the stochastic gradients error. Formal theoretical analysis on convergence behaviors with this approximation is given in later sections. Using similar methods, $\nabla_{\boldsymbol{\theta}} G_2^t(\boldsymbol{\theta})$ is also approximately calculated.

### 4.2 Refinement

The *refinement* stage corresponds to the zero-temperature limit of the *exploration* stage, where $\boldsymbol{\Xi}$ is learned. We show that in the limit Santa gives significantly simplified updates, leading to an stochastic optimization algorithm similar to Adam or SGD-M.

We assume that the Markov chain has reached its equilibrium after the *exploration* stage. In the zero-temperature limit, some terms in the SDE (1) vanish. First, as $\beta \to \infty$, the term $\frac{1}{\beta}\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_1(\boldsymbol{\theta})$ and the variance term for the Brownian motion approach 0. As well, the thermostat variable $\boldsymbol{\Xi}$ approaches $\mathbf{G}_2(\boldsymbol{\theta})$, so the term $\mathbf{G}_1(\boldsymbol{\theta})(\boldsymbol{\Xi} - \mathbf{G}_2(\boldsymbol{\theta}))\nabla_{\boldsymbol{\theta}}\,\mathbf{G}_2(\boldsymbol{\theta})$ vanishes. The stationary distribution in (2) implies $\mathbb{E}\,\mathbf{Q}_{ii} \triangleq \mathbb{E}\,\mathbf{p}_i^2 \to 0$, which makes the SDE for $\boldsymbol{\Xi}$ in (1) vanish. As a result, in the *refinement* stage, only $\boldsymbol{\theta}$ and $\mathbf{p}$ need to be updated. The Euler scheme for this is shown in Algorithm 1, and the symmetric splitting scheme is shown in Algorithm 2.

**Relation to stochastic optimization algorithms** In the *refinement* stage Santa is a stochastic optimization algorithm. This relation is easier seen with the Euler scheme in Algorithm 1. Compared with SGD-M (Rumelhart et al., 1986), Santa has both adaptive gradient and adaptive momentum updates. Unlike Adagrad (Duchi et al., 2011) and RMSprop (Tieleman and

Hinton, 2012), *refinement* Santa is a momentum based algorithm.

The recently proposed Adam algorithm (Kingma and Ba, 2015) incorporates momentum and preconditioning in what is denoted as "adaptive moments." We show in Supplementary Section F that a constant step size combined with a change of variables nearly recovers the Adam algorithm with element-wise momentum weights. For these reasons, Santa serves as a more general stochastic optimization algorithm that extends all current algorithms. As well, for a convex problem and a few trivial algorithmic changes, the regret bound of Adam holds for *refinement* Santa, which is $\mathcal{O}(\sqrt{T})$, as detailed in Supplementary Section F. However, our analysis is focused on non-convex problems that do not fit in the regret bound formulation.

### 4.3 Convergence properties

Our convergence properties are based on the framework of Chen et al. (2015). The proofs for all theorems are given in the Supplementary Material. We focus on the *exploration* stage of the algorithm. Using the Monotone Convergence argument (Schechter, 1997), the *refinement* stage convergence is obtained by taking the temperature limit from the results of the *exploration* stage. We emphasize that our approach differs from conventional stochastic optimization or online optimization approaches. Our convergence rate is weaker than many stochastic optimization methods, including SGD; however, our analysis applies to *non-convex* problems, whereas traditionally convergence rates only apply to convex problems.

The goal of Santa is to obtain $\boldsymbol{\theta}^*$ such that $\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} U(\boldsymbol{\theta})$. Let $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_L\}$ be a sequence of parameters collected from the algorithm. Define $\hat{U} \triangleq \frac{1}{L}\sum_{t=1}^{L} U(\boldsymbol{\theta}_t)$ as the sample average, $\bar{U} \triangleq U(\boldsymbol{\theta}^*)$ the global optima of $U(\boldsymbol{\theta})$.

As in Chen et al. (2015), we require certain assumptions on the potential energy $U$. To show these assumptions, we first define a functional $\psi_t$ for each $t$ that solves the following Poisson equation:

$$\mathcal{L}_t \psi_t(\boldsymbol{\theta}_t) = U(\boldsymbol{\theta}_t) - \bar{U} \ , \tag{4}$$

$\mathcal{L}_t$ is the generator of the SDE system (1) in the $t$-th iteration, defined $\mathcal{L}_t f(\mathbf{x}_t) \triangleq \lim_{h\to 0^+} \frac{\mathbb{E}[f(\mathbf{x}_{t+h})] - f(\mathbf{x}_t)}{h}$ where $\mathbf{x}_t \triangleq (\boldsymbol{\theta}_t, \mathbf{p}_t, \boldsymbol{\Xi}_t)$, $f : \mathbf{R}^{3p} \to \mathbf{R}$ is a compactly supported twice differentiable function. The solution functional $\psi_t(\boldsymbol{\theta}_t)$ characterizes the difference between $U(\boldsymbol{\theta}_t)$ and the global optima $\bar{U}$ for every $\boldsymbol{\theta}_t$. As shown in Mattingly et al. (2010), (4) typically possesses a unique solution, which is at least as smooth as $U$ under the elliptic or hypoelliptic settings. We assume $\psi_t$ is bounded and smooth, as described below.

**Assumption 1.** *$\psi_t$ and its up to 3rd-order derivatives, $\mathcal{D}^k\psi_t$, are bounded by a function $\mathcal{V}(\boldsymbol{\theta}, \mathbf{p}, \boldsymbol{\Xi})$, i.e., $\|\mathcal{D}^k\psi\| \le C_k\mathcal{V}^{r_k}$ for $k = (0, 1, 2, 3)$, $C_k, r_k > 0$. Furthermore, the expectation of $\mathcal{V}$ is bounded: $\sup_t \mathbb{E}\mathcal{V}^r(\boldsymbol{\theta}, \mathbf{p}, \boldsymbol{\Xi}) < \infty$, and $\mathcal{V}$ is smooth such that $\sup_{s\in(0,1)} \mathcal{V}^r\left(s\boldsymbol{x} + (1-s)\boldsymbol{y}\right) \le C\left(\mathcal{V}^r(\boldsymbol{x}) + \mathcal{V}^r(\boldsymbol{y})\right)$, $\forall \boldsymbol{x} \in \mathbf{R}^{3p}, \boldsymbol{y} \in \mathbf{R}^{3p}, r \le \max\{2r_k\}$ for some $C > 0$.*

Let $\Delta U(\boldsymbol{\theta}) \triangleq U(\boldsymbol{\theta}) - U(\boldsymbol{\theta}^*)$. Further define an operator $\Delta V_t = \left(G_1(\boldsymbol{\theta})(\nabla_{\boldsymbol{\theta}}\tilde{U}_t - \nabla_{\boldsymbol{\theta}}U) + \frac{\mathcal{B}_t}{\beta_t}\right) \cdot \nabla_{\mathbf{p}}$ for each $t$, where $\mathcal{B}_t$ is from (3). Theorem 2 depicts the closeness of $\hat{U}$ to the global optima $\bar{U}$ in term of *bias* and *mean square error* (MSE) defined below.

**Theorem 2.** *Let $\|\cdot\|$ be the operator norm. Under Assumption 1, the bias and MSE of the* exploration *stage in Santa with respect to the global optima for $L$ steps with stepsize $h$ is bounded, for some constants $C > 0$ and $D > 0$, with:*

$$\textit{Bias: } \left|\mathbb{E}\hat{U} - \bar{U}\right| \le Ce^{-U(\boldsymbol{\theta}^*)}\left(\frac{1}{L}\sum_{t=1}^{L}\int e^{-\beta_t\Delta U(\boldsymbol{\theta})}\mathrm{d}\boldsymbol{\theta}\right) + D\left(\frac{1}{Lh} + \frac{\sum_t \|\mathbb{E}\Delta V_t\|}{L} + h^2\right) .$$

$$\textit{MSE: } \mathbb{E}\left(\hat{U} - \bar{U}\right)^2 \le C^2e^{-2U(\boldsymbol{\theta}^*)}\left(\frac{1}{L}\sum_{t=1}^{L}\int e^{-\beta_t\Delta U(\boldsymbol{\theta})}\mathrm{d}\boldsymbol{\theta}\right)^2 + D^2\left(\frac{\frac{1}{L}\sum_t \mathbb{E}\|\Delta V_t\|^2}{L} + \frac{1}{Lh} + h^4\right) .$$

Both bounds for the bias and MSE have two parts. The first part contains integration terms, which characterizes the distance between the global optima, $e^{-U(\boldsymbol{\theta}^*)}$, and the unnormalized annealing distributions, $e^{-\beta_t U(\boldsymbol{\theta})}$, decreasing to zero exponentially fast with increasing $\beta$; the remaining part characterizes the distance between the sample average and the annealing posterior average. This shares a similar form as in general SG-MCMC algorithms (Chen et al., 2015), and can be controlled to converge. Furthermore, the term $\frac{\sum_t \|\mathbb{E}\Delta V_t\|}{L}$ in the bias vanishes as long as the sum of the annealing sequence $\{\beta_t\}$ is finite[6], indicating that the gradient approximation for $\nabla_{\boldsymbol{\theta}}G_1(\boldsymbol{\theta})$ in Section 4.1 does not affect the bias of the algorithm. Similar arguments apply for the MSE bound.

To get convergence results right before the *refinement* stage, let a sequence of functions $\{g_m\}$ be defined as $g_m \triangleq -\frac{1}{L}\sum_{l=m}^{L+m-1} e^{-\beta_l\hat{U}(\boldsymbol{\theta})}$; it is easy to see that $\{g_m\}$ satisfies $g_{m_1} < g_{m_2}$ for $m_1 < m_2$, and $\lim_{m\to\infty} g_m = 0$. According to the Monotone Convergence Theorem (Schechter, 1997), the bias and MSE in the limit exists, leading to Corollary 3.

**Corollary 3.** *Under Assumptions 1, the bias and MSE of the* refinement *stage in Santa with respect*

[6] In practice we might not need to care about this constraint because a small bias in the *exploration* stage does not affect convergence of the *refinement* stage.

*to the global optima for $L$ steps with stepsize $h$ are bounded, for some constants $D_1 > 0$, $D_2 > 0$, as*

$$\textit{Bias: } \left|\mathbb{E}\hat{U} - \bar{U}\right| \leq D_1 \left(\frac{1}{Lh} + \frac{\sum_t \|\mathbb{E}\Delta V_t\|}{L} + h^2\right)$$

$$\textit{MSE: } \mathbb{E}\left(\hat{U} - \bar{U}\right)^2 \leq D_2 \left(\frac{\frac{1}{L}\sum_t \mathbb{E}\|\Delta V_t\|^2}{L} + \frac{1}{Lh} + h^4\right)$$

Corollary 3 implies that in the *refinement* stage, the discrepancy between annealing distributions and the global optima vanishes, leaving only errors from discretized simulations of the SDEs, similar to the result of general SG-MCMC (Chen et al., 2015). We note that after *exploration*, Santa becomes a pure stochastic optimization algorithm, thus convergence results in term of regret bounds can also be derived; refer to Supplementary Section F for more details.

# 5 Experiments

## 5.1 Illustration

In order to demonstrate that Santa is able to achieve the global mode of an objective function, we consider the double-well potential (Ding et al., 2014),

$$U(\theta) = (\theta + 4)(\theta + 1)(\theta - 1)(\theta - 3)/14 + 0.5 \ .$$

As shown in Figure 1 (left), the double-well potential has two modes, located at $\theta = -3$ and $\theta = 2$, with the global optima at $\theta = -3$. We use a decreasing learning rate $h_t = t^{-0.3}/10$, and the annealing sequence is set to $\beta_t = t^2$. To make the optimization more challenging, we initialize the parameter at $\theta_0 = 4$, close to the local mode. The evolution of $\theta$ with respect to iterations is shown in Figure 1(right). As can be seen, $\theta$ first moves to the local mode but quickly jumps out and moves to the global mode in the *exploration* stage (first half iterations); in the *refinement* stage, $\theta$ quickly converges to the global mode and sticks to it afterwards. In contrast, RMSprop is trapped on the local optima, and convergences slower than Santa at the beginning.

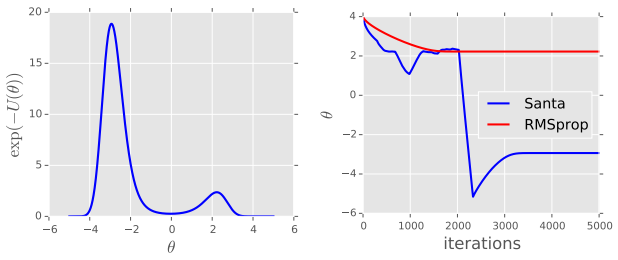


Figure 1: (Left) Double-well potential. (Right) The evolution of $\theta$ using Santa and RMSprop algorithms.

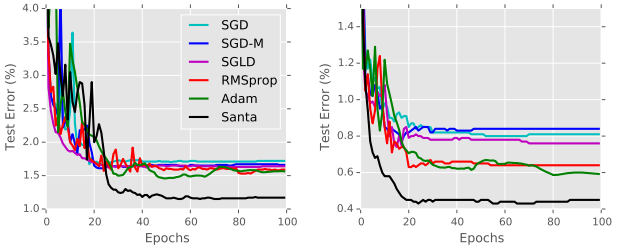


Figure 2: Learning curves of different algorithms on MNIST. (Left) FNN with size of 400. (Right) CNN.

## 5.2 Feedforward neural networks

We first test Santa on the Feedforward Neural Network (FNN) with rectified linear units (ReLU). We test two-layer models with network sizes 784-X-X-10, where X is the number of hidden units for each layer; 100 epochs are used. For variants of Santa, we denote Santa-E as Santa with a Euler scheme illustrated in Algorithm 1, Santa-r as Santa running only on the *refinement* stage, but with updates on $\boldsymbol{\alpha}$ as in the *exploration* stage. We compare Santa with SGD, SGD-M, RMSprop, Adam, SGD with dropout, SGLD and Bayes by Backprop (Blundell et al., 2015). We use a grid search to obtain good learning rates for each algorithm, resulting in $4 \times 10^{-6}$ for Santa, $5 \times 10^{-4}$ for RMSprop, $10^{-3}$ for Adam, and $5 \times 10^{-1}$ for SGD, SGD-M and SGLD. We choose an annealing schedule of $\beta_t = At^{\gamma}$ with $A = 1$ and $\gamma$ selected from 0.1 to 1 with an interval of 0.1. For simplicity, the *exploration* is set to take half of total iterations.

We test the algorithms on the standard MNIST dataset, which contains $28 \times 28$ handwritten digital images from 10 classes with $60,000$ training samples and $10,000$ test samples. The network size (X-X) is set to 400-400 and 800-800, and test classification errors are shown in Table 1. Santa show improved state-of-the-art performance amongst all algorithms. The Euler scheme shows a slight decrease in performance, due to the integration error when solving the SDE. Santa without *exploration* (*i.e.*, Santa-r) still performs relatively well. Learning curves are plotted in Figure 2, showing that Santa converges as fast as other algorithms but to a better local optima[7].

## 5.3 Convolution neural networks

We next test Santa on the Convolution Neural Network (CNN). Following Jarrett et al. (2009), a standard network configuration with 2 convolutional layers followed by 2 fully-connected layers is adopted. Both convolutional layers use $5 \times 5$ filter size with 32 and 64

[7]Learning curves of FNN with size of 800 are provided in Supplementary Section G.

| Algorithms | FNN-400 | FNN-800 | CNN |
|---|---|---|---|
| Santa | **1.21%** | **1.16%** | **0.47%** |
| Santa-E | 1.41% | 1.27% | 0.58% |
| Santa-r | 1.45% | 1.40% | 0.49% |
| Adam | 1.53% | 1.47% | 0.59% |
| RMSprop | 1.59% | 1.43% | 0.64% |
| SGD-M | 1.66% | 1.72% | 0.77% |
| SGD | 1.72% | 1.47% | 0.81% |
| SGLD | 1.64% | 1.41% | 0.71% |
| BPB$^\diamond$ | 1.32% | 1.34% | – |
| SGD, Dropout$^\diamond$ | 1.51% | 1.33% | – |
| Stoc. Pooling$^\triangleright$ | – | – | 0.47% |
| NIN, Dropout$^\circ$ | – | – | 0.47% |
| Maxout, Dropout$^\star$ | – | – | 0.45% |

Table 1: Test error on MNIST classification using FNN and CNN. ($^\diamond$) taken from Blundell et al. (2015). ($^\triangleright$) taken from Zeiler and Fergus (2013). ($^\circ$) taken from Lin et al. (2014). ($^\star$) taken from Goodfellow et al. (2013).

channels, respectively; $2 \times 2$ max pooling is used after each convolutional layer. The fully-connected layers have 200-200 hidden nodes with ReLU activation. The same parameter setting and dataset as in the FNN are used. The test errors are shown in Table 1, and the corresponding learning curves are shown in Figure 2. Similar trends as in FNN are obtained. Santa significantly outperforms other algorithms with an error of 0.45%. This result is comparable or even better than some recent state-of-the-art CNN-based systems, which have much more complex architectures.

### 5.4 Recurrent neural networks

We test Santa on the Recurrent Neural Network (RNN) for sequence modeling, where a model is trained to minimize the negative log-likelihood of training sequences:

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} -\log p(\mathbf{x}_t^n \,|\, \mathbf{x}_1^n, \ldots, \mathbf{x}_{t-1}^n; \boldsymbol{\theta}) \quad (5)$$

where $\boldsymbol{\theta}$ is a set of model parameters, $\{\mathbf{x}_t^n\}$ is the observed data. The conditional distributions in (5) are modeled by the RNN. The hidden units are set to gated recurrent units (Cho et al., 2014).

We consider the task of sequence modeling on four different polyphonic music sequences of piano, *i.e.*, Piano-midi.de (Piano), Nottingham (Nott), MuseData (Muse) and JSB chorales (JSB). Each of these datasets are represented as a collection of 88-dimensional binary sequences, that span the whole range of piano from A0 to C8.

The number of hidden units is set to 200. Each model is trained for at most 100 epochs. According to the experiments and their results on the validation set, we use a learning rate of 0.001 for all the algorithms. For Santa, we consider an additional experiment using a learning rate of 0.0002, denoted Santa-s. The annealing coefficient $\gamma$ is set to 0.5. Gradients are clipped if the norm of the parameter vector exceeds 5. We do not perform any dataset-specific tuning other than early stopping on validation sets. Each update is done using a minibatch of one sequence.

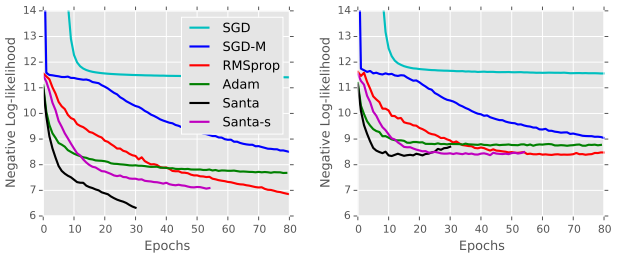


Figure 3: Learning curves of different algorithms on Piano using RNN. (Left) training set. (Right) validation set.

| Algorithms | Piano. | Nott. | Muse. | JSB. |
|---|---|---|---|---|
| Santa | **7.60** | **3.39** | **7.20** | **8.46** |
| Adam | 8.00 | 3.70 | 7.56 | 8.51 |
| RMSprop | 7.70 | 3.48 | 7.22 | 8.52 |
| SGD-M | 8.32 | 3.60 | 7.69 | 8.59 |
| SGD | 11.13 | 5.26 | 10.08 | 10.81 |
| HF$^\diamond$ | 7.66 | 3.89 | **7.19** | 8.58 |
| SGD-M$^\diamond$ | 8.37 | 4.46 | 8.13 | 8.71 |

Table 2: Test negative log-likelihood results on polyphonic music datasets using RNN. ($^\diamond$) taken from Boulanger-Lewandowski et al. (2012).

The best log-likelihood results on the test set are achieved by using Santa, shown in Table 2. Learning curves on the Piano dataset are plotted in Figure 3. We observe that Santa achieves fast convergence, but is overfitting. This is straightforwardly addressed through early stopping. The learning curves for all the other datasets are provided in Supplementary Section G.

## 6 Conclusions

We propose Santa, an annealed SG-MCMC method for stochastic optimization. Santa is able to explore the parameter space efficiently and locate close to the global optima by annealing. At the zero-temperature limit, Santa gives a novel stochastic optimization algorithm where both model parameters and momentum are updated element-wise and adaptively. We provide theory on the convergence of Santa to the global optima for an (non-convex) objective function. Experiments show best results on several deep models compared to related stochastic optimization algorithms.

## Acknowledgements

This research was supported in part by ARO, DARPA, DOE, NGA, ONR and NSF.

## References

C. Andrieu, N. de Freitas, and A. Doucet. Reversible jump mcmc simulated annealing for neural networks. In *UAI*, 2000.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *ICML*, 2015.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. COMPSTAT*, 2010.

N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML*, 2012.

C. Chen, N. Ding, and L. Carin. On the convergence of stochastic gradient mcmc algorithms with high-order integrators. In *NIPS*, 2015.

T. Chen, E. B. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *ICML*, 2014.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *arXiv:1406.1078*, 2014.

Y. N. Dauphin, H. de Vries, and Y. Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *NIPS*, 2015.

N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven. Bayesian sampling using stochastic gradient thermostats. In *NIPS*, 2014.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *JMLR*, 2011.

Z. Gan, C. Chen, R. Henao, D. Carlson, and L. Carin. Scalable deep Poisson factor analysis for topic modeling. In *ICML*, 2015.

S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *PAMI*, 1984.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. In *JRSS*, 2011.

I. Goodfellow, D. Warde-farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.

K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

S. Kirkpatrick, C. D. G. Jr, and M. P. Vecchi. Optimization by simulated annealing. In *Science*, 1983.

Y. Li, V. A. Protopopescu, N. Arnold, X. Zhang, and A. Gorin. Hybrid parallel tempering and simulated annealing method. In *Applied Mathematics and Computation*, 2009.

M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014.

J. C. Mattingly, A. M. Stuart, and M. V. Tretyakov. Construction of numerical time-average and stationary measures via Poisson equations. In *SIAM J. NUMER. ANAL.*, 2010.

R. M. Neal. Annealed importance sampling. In *Statistics and Computing*, 2001.

R. M. Neal. Mcmc using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, 2011.

F. Obermeyer, J. Glidden, and E. Jonas. Scaling nonparametric bayesian inference via subsample-annealing. In *AISTATS*, 2014.

S. Patterson and Y. W. Teh. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *NIPS*, 2013.

H. Risken. *The Fokker-Planck equation*. Springer-Verlag, New York, 1989.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. In *Nature*, 1986.

E. Schechter. *Handbook of Analysis and Its Foundations*. Elsevier, 1997.

I. Sutskever, J. Martens, G. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Y. W. Teh, A. H. Thiery, and S. J. Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. In *arXiv:1409.0578*, 2014.

T. Tieleman and G. E. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *Coursera: Neural Networks for Machine Learning*, 2012.

J. W. van de Meent, B. Paige, and F. Wood. Tempering by subsampling. In *arXiv:1401.7145*, 2014.

V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. In *J. Optimization Theory and Applications*, 1985.

S. J. Vollmer, K. C. Zygalakis, and Y. W. Teh. (Non-)asymptotic properties of stochastic gradient Langevin dynamics. In *arXiv:1501.00438*, 2015.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, 2011.

M. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.

M. D. Zeiler. Adadelta: An adaptive learning rate method. In *arXiv:1212.5701*, 2012.