Embedding Conjugate Gradient in Learning Random Walks for Landscape Connectivity Modeling in Conservation

Pramith Devulapalli[1] , **Bistra Dilkina**[2] and **Yexiang Xue**[1]
[1]Purdue University
[2]University of Southern California
pdevulap@purdue.edu, dilkina@usc.edu, yexiang@purdue.edu

Abstract

Models capturing parameterized random walks on graphs have been widely adopted in wildlife conservation to study species dispersal as a function of landscape features. Learning the probabilistic model empowers ecologists to understand animal responses to conservation strategies. By exploiting the connection between random walks and simple electric networks, we show that learning a random walk model can be reduced to finding the optimal graph Laplacian for a circuit. We propose a moment matching strategy that correlates the model's hitting and commuting times with those observed empirically. To find the best Laplacian, we propose a neural network capable of back-propagating gradients through the matrix inverse in an end-to-end fashion. We developed a scalable method called CGInv which back-propagates the gradients through a neural network encoding each layer as a conjugate gradient iteration. To demonstrate its effectiveness, we apply our computational framework to applications in landscape connectivity modeling. Our experiments successfully demonstrate that our framework effectively and efficiently recovers the ground-truth configurations.

1 Introduction

With many animal species experiencing a population decline due to habitat loss and fragmentation, efforts to conserve wildlife and to protect biodiversity are forefront challenges for computational sustainability [Gomes, 2009]. The difficulty of the task intensifies with the necessity to accurately predict the effects of broad-scale ecological processes from different actions implemented by conservation efforts [McRae and Beier, 2007].

Towards these set of challenges, artificial intelligence has been employed to understand the relationship between environmental variables and animal movement behavior. Early approaches in this domain were based on maximizing graph-theoretic notions such as the least-cost paths, multi-commodity flows, steiner trees, etc. [Sheldon *et al.*, 2010; Bras *et al.*, 2013; Dilkina *et al.*, 2013] but fall short in capturing the stochasticity of animal decision-making.

A newer model called the Circuitscape [McRae *et al.*, 2008] incorporates circuit theory and random walks to model animals as individual agents who traverse to neighboring land parcels with probabilities proportional to resistance values, i.e., the cost of moving. However, existing methods indirectly learn random walk parameters; these parameters are derived from separate statistical models that estimate the landscape cost [Inman *et al.*, 2013; McClure *et al.*, 2016], which can lead to inaccurate random walk models that may not resemble true movement dynamics. Moreover, ecologists have stated a dire need to develop automated procedures to weigh different environmental factors in constructing Circuitscape models [Dickson *et al.*, 2019].

We develop a ***principled computational framework*** that learns random walk models directly from data. Our core idea is a *moment matching strategy*, which correlates the model's statistical properties, such as the hitting and commuting times, with those observed empirically. By exploiting the connection of random walks with circuit theory, we show that learning the probabilistic process can be reduced to finding the optimal graph Laplacian matching empirical observations. The graph Laplacian is the central notion that connects statistical properties of random walks with concepts of voltages, currents, and resistances. Following a gradient descent strategy, our forward pass to compute the hitting and commuting times requires inverting the graph Laplacian. The backward pass, therefore, requires an efficient computation scheme to flow the gradients through the matrix inverse.

Computationally, we propose ***a neural network capable of back-propagating gradients through the matrix inverse in an end-to-end fashion***. Our first idea, DirectInv, was developed through the implicit function theorem, motivated by the work of [Amos and Kolter, 2017; Ling *et al.*, 2019]. Nevertheless, the direct application of the implicit function theorem leads to a Kronecker computation, which is quartic in the size of the graph Laplacian. We then developed CGInv, which ***back-propagates the gradients through a neural network encoding each layer as one conjugate gradient iteration***. While the time complexity of full CGInv is lower than DirectInv, CGInv can be further expedited by initializing the current execution with the root from the previous one, which is relatively close to the correct solution. This

decreases the number of iterations to convergence and further speeds up the forward and backward computations.

To showcase our computational framework, we designed two sets of experiments to learn random walk models from animal movement data. In the first experiment, we simulated data from Dilkina *et al.* (2013) and learnt optimal models from empirical observations. Our second experiment learned landscape-resistance values by correlating them with genetic similarities among animal individuals in different locations via the Mantel test. In both applications, our approach succeeds in recovering the ground-truth animal movement patterns. Our contributions can be summarized in the following way:

1. We formulated an end-to-end computational framework using a moment matching strategy to learn random walks from data.
2. We developed an efficient and powerful technique called CGInv to back-propagate gradients through the matrix inverse, via embedding each conjugate gradient step as a neural network layer.
3. We successfully demonstrated the effectiveness of our framework in recovering the ground-truth animal movement patterns from realistic datasets.

2 Random Walks and Circuit Theory

We describe the behavior of a single animal individual as a random walker traversing over a predefined landscape. The landscape is modeled as an undirected graph $G = (V, E)$ comprised of a node set V representing various locations and an edge set E representing direct, physical connections between two locations. The animal individual is fitted with unique transition probabilities which are species-specific. The transition probability $p_{u,v}$ from node u to node v is encoded in edge $(u, v) \in E$ through an edge weight.

Circuit Theory. The Circuitscape model builds an electrical circuit described by the graph G. The edge weights feature resistance values, $r_{u,v}$, that reflect the cost to travel along a given edge (u, v) from u to v. The reciprocal of resistance, $1/r_{u,v}$, yields a conductance value denoted as $c_{u,v}$. We connect the Circuitscape model with the random walk model by letting the transition probability of a random walker at node u traveling to node v be $c_{u,v}/C_u$ where $C_u = \sum_{v \in N(u)} c_{u,v}$ represents the sum of edge conductance values from all the neighboring nodes $v \in N(u)$ of node u.

Ohm's Law. The Ohm's law relates voltages, currents, and resistances in the Circuitscape model through a set of linear equations. Let C_G be the conductance matrix, where its u, v-th entry $c_{u,v}$ represents the conductance value of edge (u, v). Let D_G be a diagonal matrix, where the u-th diagonal entry is C_u, the sum of the conductance values of the edges incident to node u. The *graph Laplacian* of G is defined as $L_G = D_G - C_G$. Denote $i = (i_1, \ldots, i_n)^T$ as the vector of *currents*, in which i_k is the amount of current that has been injected to node k[1]. Denote $v = (v_1, \ldots, v_n)^T$ as the vector of *voltages*, in which v_k is the voltage value of node k[2]. Then, Ohm's Law can be written in the following matrix form:

$$L_G \, v = i. \tag{1}$$

An important extension of this principle is used in defining *effective resistance*, $r^{eff}_{u,v}$, between two nodes u and v. The $r^{eff}_{u,v}$ measures the resistance between u and v by injecting 1 ampere of current into node u, extracting 1 ampere from node v, and measuring $V(u) - V(v)$. Mathematically, let v^{st} be the voltage vector corresponding to the current vector $i^{st} = 1_s - 1_t$ with only 1 at the s and t-th entries and 0 elsewhere. Applying Ohm's law , the effective resistance $r^{eff}_{s,t}$ is exactly the voltage difference between the s-th and the t-th entry of v^{st}, i.e.,

$$r^{eff}_{s,t} = (1_s - 1_t)^T L_G^{-1} (1_s - 1_t). \tag{2}$$

Here L_G^{-1} represents the pseudo-inverse. In this paper, we assume that graph G is always connected. Therefore L_G has exactly one zero eigenvalue, corresponding to the eigenvector $(1, \ldots, 1)^T$ implying L_G to be highly ill-conditioned. When we perform matrix inversions in this paper, we remove the singularity of L_G by grounding a single node in the voltage vector to 0. Then, we remove the row and column in L_G that corresponds to the grounded node. This reduction yields in a well-conditioned, invertible matrix that can be handled.

Hitting and Commuting Times. An important notion in random walks is the hitting time. The hitting time H_{uv} from u to v represents the expected number of steps for a random walker on G to reach v starting from u. From the hitting time, the commuting time J_{uv} is defined as $J_{uv} = H_{uv} + H_{vu}$, which represents the expected number of steps to start from u, travel to v, and come back at u. The following two theorems connect effective resistance and Ohm's Law to the commuting and hitting times respectively.

Theorem 1. *[Chandra* et al.*, 1996] Let* $F = \sum_{x,y} 1/r_{x,y} = \sum_{x,y} c_{x,y}$. *The commuting time for a random walk between node s and t is equal to* $F \; r^{eff}_{s,t}$.

Theorem 2. *[Chandra* et al.*, 1996] Suppose we inject* $\sum_{y \in N(x)} \frac{1}{r_{xy}} = \sum_{y \in N(x)} c_{xy}$ *amount of current to each node* $x \in V$ *and extract* $\sum_{x \in V} \sum_{y \in N(x)} \frac{1}{r_{xy}}$ *amount of current from node v, then the voltage difference between node u and node v,* Φ_{uv} *is exactly the hitting time between u, v,* H_{uv}. *In mathematical notation,* $H_{uv} = (1_u - 1_v)^T L_G^{-1} i_{uv}$, *where the v-th entry of* i_{uv} *is* $\sum_{x \neq v} \sum_{y \in N(x)} c_{xy}$, *and the x-th entry of* i_{uv} $(x \neq v)$ *is* $\sum_{y \in N(x)} c_{xy}$.

3 Learning Random Walks

Learning the migration of wild-life animals is crucial towards designing effective conservation strategies. Maintaining landscape connectivity allows animals to roam freely, promoting the dispersal of their gene flows and increasing the robustness of the species against environmental changes.

[1] Current that flows out of a node is denoted by negative values

[2] All voltages should be referenced with respect to a given node, for example, node 0. In other words, we assume that node 0 is grounded.

To understand landscape connectivity, the Circuitscape model was developed to use circuit theory and the concept of effective resistance to accurately represent landscape-to-resistance costs [McRae, 2006; McRae *et al.*, 2008]. While circuit-theory based models are powerful, many approaches in inferring landscape connectivity parameters rely on expert designed values or subjective methods in evaluating the contribution of different environmental factors.

To combat these challenges, we developed a principled framework whose foundation lies in the *method of moments* technique to precisely estimate landscape-to-resistance values from data. We use method of moments approach to build the appropriate random walk model by finding resistance values that best mimic animal movement patterns.

We first learn a Circuitscape model representing animals' dispersal patterns based on partially observed trajectories. As a second application, we learn landscape features by modeling gene flow as done in previous work [Cushman *et al.*, 2006; McRae and Beier, 2007]. We correlate the genetic distances of animal individuals observed across a landscape with their migratory patterns using the Circuitscape model. We show that both applications can be solved with a single computational framework.

3.1 Learning Random Walks from Partial Trajectories

As animals disperse across their habitats, they arrive at different observable locations at different times. Since the path taken by the animal between two observed locations is unknown, the time series data represents partial trajectory data as a sequence of time-location pairs:

$$\{(t_1,l_1),(t_2,l_2),(t_3,l_3),\dots,(t_k,l_k)\}$$

Here, a single animal individual visits location l_1 at time t_1, then visits location l_2 at time t_2, and repeats this process until time t_k. The behavior learning problem is: *how can we identify the probabilistic model of the random walk given partial trajectory data listed as above?*

We adopt the method of moments technique to learn the true probability distribution by matching the empirically observed hitting times with those generated from the Circuitscape model. From data, we can observe *instances* of the hitting time between any two nodes u and v. An instance of the hitting time represents the number of steps an animal individual was observed to travel starting from u and arriving at v for the first time. The following is a more rigorous definition:

$$h_{l_u,l_v} = t_u - t_v$$

The time t_u represents the animal's timestamp at location l_u and the time t_v represents the animal's timestamp at location l_v. As the number of observations increase, the true expected hitting time should be well approximated by the empirically computed hitting times.

Therefore, the core idea is to match the expected hitting times computed by the Circuitscape model to the N instances of the hitting times observed empirically:

$$L_G^* = \arg\min_{L_G} \sum_{i=1}^{N} \left(h_{l_i,l_{i+1}} - H_{l_i,l_{i+1}}\right)^2 \tag{3}$$

This optimization equation translates into finding the optimal L_G^* that minimizes the sum of the squared differences for all N instances of the hitting time observed in the data.

3.2 Mantel Coefficient Optimization

A fundamental challenge within ecology and conservation biology is to use genetic approaches to understand the dispersal of animal species over long periods of time [Cushman *et al.*, 2006; McRae and Beier, 2007]. One approach employed by Cushman *et al.* (2006) is to manually find landscape-resistance costs through Mantel tests among the genetic distances between pairs of animal individuals and their least-cost paths. Nevertheless, they used a brute-force approach to consider all combinations of environmental features. We extend the study by Cushman *et al.* (2006) by providing a principled way to find the optimal combination of environmental features via correlation of the genetic distances with commuting times inferred from the resistance surface.

Here, we assume as input a genetic distance matrix that describes the genetic similarity between all pairs of animal individuals across a landscape. Similar to Cushman *et al.* (2006), we use the Bray-Curtis percentage dissimilarity measure of genetic distance between individuals where 0 indicates identical composition and 1 indicates completely different individuals. Then, we compute the commuting times between all pairs of individuals using the Circuitscape model and find the L_G that maximizes the Mantel correlation coefficient with the genetic distance matrix [Legendre and Legendre, 2012]. This translates to finding the graph Laplacian L_G which maximizes the following objective:

$$L_G^* = \arg\max_{L_G} \frac{1}{d-1} \sum_{k=1}^{n-1} \sum_{l=k+1}^{n} \left(\frac{G_{kl}-\bar{G}}{s_G}\right)\left(\frac{J_{kl}-\bar{J}}{s_J}\right) \tag{4}$$

Here, G_{kl} refers to specific indexed locations within the genetic distance matrix G. Similarly, J_{kl} refers to specific matrix elements within the commuting time matrix J. The mean and standard deviation of the genetic matrix are $\bar{G}$ and s_G respectively. The mean and standard deviation of the commuting time are $\bar{J}$ and s_J respectively. Since G and J are symmetric, Equation (4) only operates on the upper-triangular section of both matrices where $d = n(n-1)/2$ is the number of items in the sum.

3.3 Training via Stochastic Gradient Descent

Analytically solving the optimization problems in Equations (3) and (4) is infeasible due to over-constrained linear systems. Gradient-based learning offers a lucrative alternative that yields a straightforward methodology. In both scenarios, the gradients from the optimization equations need to propagate back to the physical resistance values to perform stochastic gradient descent. We can see that the hitting times $H_{l_i,l_{i+1}}$ from Equation (3) and commuting time terms J_{kl}, $\bar{J}$, and s_J from Equation (4) both rely on the L_G^{-1} from Theorems (2) and (1) respectively. We must propagate one step further to the L_G to reach the conductance values, which are reciprocals of resistance. By finding the gradients through L_G, we can then implement a stochastic gradient descent framework to perform updates on the resistance values.

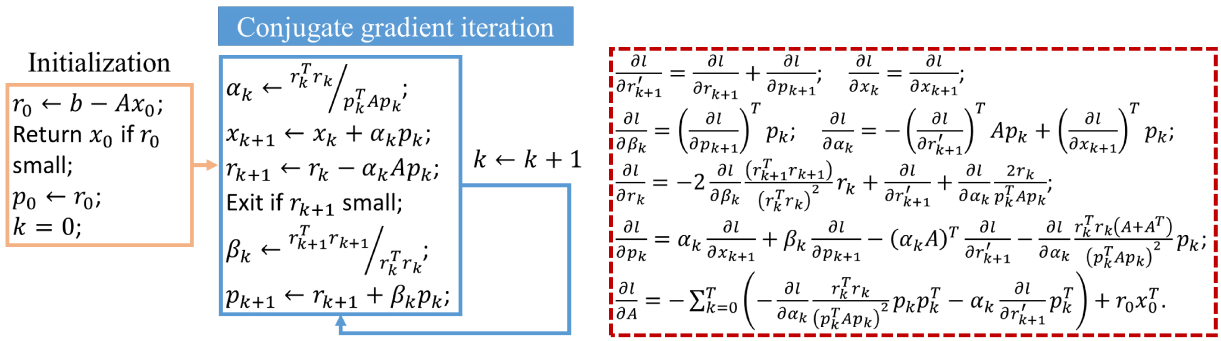


Figure 1: (Left, orange and blue box) The conjugate gradient iteration to find the root for $Ax = b$. Every operation in one iteration is fully differentiable. One conjugate gradient iteration takes the input of r_k, p_k and x_k, and computes r_{k+1}, p_{k+1} and x_{k+1} as the output for the next iteration. After n iterations, x_{n+1} is guaranteed to be the root of $Ax = b$. (Right, red dashed box) Back-propagate gradient through one conjugate gradient iteration. Here, we assume having access to the partial derivative of the loss function w.r.t. the output r_{k+1}, p_{k+1} and x_{k+1}, $\frac{\partial l}{\partial r_{k+1}}$, $\frac{\partial l}{\partial p_{k+1}}$ and $\frac{\partial l}{\partial x_{k+1}}$. We compute $\frac{\partial l}{\partial r_k}$, $\frac{\partial l}{\partial p_k}$ and $\frac{\partial l}{\partial x_k}$ w.r.t. the input. The computation is dominated by matrix vector multiplication, which scales $O(n^2)$. The last line shows $\frac{\partial l}{\partial A}$ can be computed via $\frac{\partial l}{\partial r_k}$ and $\frac{\partial l}{\partial \alpha_k}$.

4 Propagating Gradients through the Matrix Inverse Efficiently

Since hitting and commuting times rely on inverting L_G in Ohm's Law in Equation (1), a key ingredient in the stochastic gradient descent framework is to compute the following partial derivative:

$$\frac{\partial L_G^{-1}}{\partial L_G}$$

In our machine learning framework, successfully computing this quantity dictates the efficiency of the gradient descent methodology. We attacked this problem from two different angles. We initially started our approach, DirectInv, by using the implicit function theorem to derive a gradient rule. We iterated upon this idea to develop our second approach, CGInv, which inverts the process of conjugate gradient. The following two sections detail both approaches.

4.1 DirectInv using Implicit Function Theorem

To compute the $\frac{\partial A^{-1}}{\partial A}$, we reformulate the matrix A and A^{-1} into vectors:

$$vec(A) = (k_{1,1}, ..., k_{1,n}, k_{2,1}, ..., k_{2,n}, ..., k_{n,1}, ..., k_{n,n})^T,$$
$$vec(A^{-1}) = (l_{1,1}, ..., l_{1,n}, l_{2,1}, ..., l_{2,n}, ..., l_{n,1}, ...l_{n,n})^T.$$

Here, $k_{i,j}$ is the i,j-th entry of the matrix A and $l_{i,j}$ is the i,j-th entry of A^{-1}. We define $\partial A^{-1}/\partial A$ as a n^2-by-n^2 matrix, in which the (r,s)-th entry is $\frac{\partial \text{vec}(A^{-1})_r}{\partial \text{vec}(A)_s}$. In other words, the $((i-1)n+j, (u-1)n+v)$-th entry of the matrix $\partial A^{-1}/\partial A$ is $\partial {A^{-1}}_{u,v}/\partial A_{i,j}$. We can derive $\partial A^{-1}/\partial A$ in a closed form, reflected by the following theorem:

Theorem 3. *Suppose matrix A is full rank, then*

$$\frac{\partial A^{-1}}{\partial A} = -A^{-1} \otimes (A^{-1})^T. \tag{5}$$

Here, $\otimes$ means the Kronecker product. Our derivation is based on the following implicit function theorem:

Theorem 4. *Let $f : \mathbf{R}^{n+m} \to \mathbf{R}^m$ be a continuously differentiable function, and let $\mathbf{R}^{n+m}$ have coordinates (x,y). Fix a point $(a,b) = (a_1, \ldots, a_n, b_1, \ldots, b_m)$ with $f(a,b) = 0$, where $0 \in \mathbf{R}^m$ is the zero vector. If the Jacobian matrix $J_{f,y}(a,b)$ is invertible, then there exists an open set U of $\mathbf{R}^n$ containing a such that there exists a unique continuously differentiable function $g : U \to \mathbf{R}^m$ such that*

$$g(a) = b,$$

and

$$f(x, g(x)) = 0 \quad \text{for all } x \in U.$$

Moreover, the partial derivatives of g in U are given by

$$\frac{\partial g(x)}{\partial x} = -J_{f,y}^{-1}(x,y)\, J_{f,x}(x,y).$$

Here, $\frac{\partial g(x)}{\partial x}$ is a matrix in which the (i,j)-th entry is $\frac{\partial g_i(x)}{\partial x_j}$.

Through this derivation, we can see that the $\partial L_G^{-1}/\partial L_G = -L_G^{-1} \otimes \left(L_G^{-1}\right)^T$ can be computed through the Kronecker product. DirectInv produces a gradient computation that requires $O(n^4)$ multiplication operations to compute the Kronecker product. Due to its large time complexity, DirectInv is limited to application on small-scale instances.

4.2 CGInv: Inverting Conjugate Gradient

Our second approach, CGInv, relies on back-propagating through a neural network representing conjugate gradient, motivated from the properties of L_G. The structure of L_G is symmetric, diagonally-dominant, and positive semi-definite. Additionally, L_G is quite sparse due to the limited number of edges incident on each node. Therefore, a matrix inversion technique needs to only work with non-zero elements of the matrix. By removing the row and column corresponding to the node with grounded voltage, L_G is a well-conditioned, positive-definite matrix suitable for conjugate gradient. An

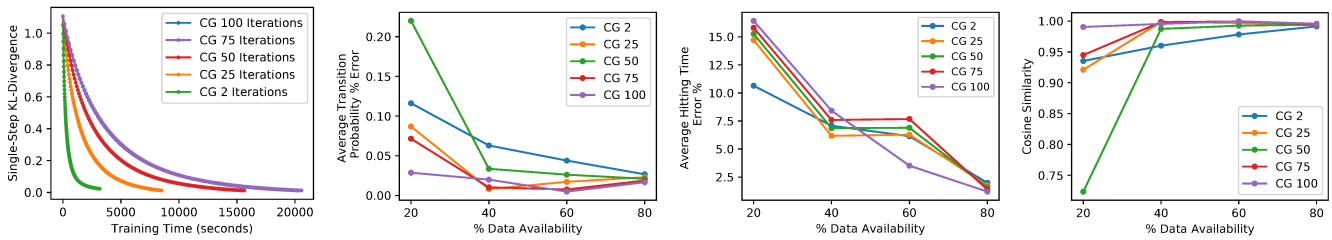


Figure 2: An overview of the results for learning random walks from partial trajectories. The first graph on the left represents an example of an experiment with 40 % data availability. We tested the performance of CGInv under different levels of maximum allowed conjugate gradient iterations. The second, third, and fourth graphs from the left represent average % error between the true and learnt transition probability matrix, average % error between the true and learnt hitting times, and the cosine similarity between the true and learnt weight vectors respectively. We can see that our models are able to recover the ground-truth random walk models, achieving almost zero in KL divergence, average transition probability error, and small errors in average hitting time and the cosine similarity of the learnt weight vectors.

illustration of conjugate gradient is shown in Figure 1. The method of conjugate gradients is initialized with a vector x_0 that serves as a guess. Following the iterative procedure as shown in the blue box of Figure 1, the k-th conjugate gradient iteration uses the current approximation of the root x_k and refines the solution to x_{k+1}. Essentially, conjugate gradient modifies the solution vector by taking correct length steps in the computed search directions, p_{k+1}, from the residual vectors r_{k+1} [Shewchuk, 1994]. Hence, the method converges in at most n iterations to find the true solution. The time complexity of each conjugate gradient iteration scales $O(n^2)$ and yields a worst-case running time of $O(n^3)$ if all n iterations are used.

Differentiating Conjugate Gradient. The depth of conjugate gradient is dependent on how far away the initial guess is from the true root, but this variability never alters the routine in each iteration. Therefore, we can treat conjugate gradient as a chain of identical processes that can be unrolled one iteration at a time. This description can naturally be extended as treating each conjugate gradient iteration as a *single layer* in a neural network and connecting them sequentially. Since matrix-vector operations dominate each iteration, we can easily back-propagate through the neural network to compute the gradient $\frac{\partial L_G^{-1}}{\partial L_G}$. The red dashed box of Figure 1 represents the partial derivatives of each quantity in a single conjugate gradient iteration with respect to the loss function. We assume access to the partial derivatives $\frac{\partial l}{\partial r_{k+1}}$, $\frac{\partial l}{\partial p_{k+1}}$, and $\frac{\partial l}{\partial x_{k+1}}$ to compute derivatives for $\frac{\partial l}{\partial r_k}$, $\frac{\partial l}{\partial p_k}$, and $\frac{\partial l}{\partial x_k}$ in the previous iteration. These derivatives are linked together to compute the partial derivative of the loss function with respect to matrix A, $\frac{\partial l}{\partial A}$, where A represents L_G. The computation of derivatives between two consecutive layers scales $O(n^2)$. Since the gradient through the matrix inverse is neatly represented as a feed-forward neural network, we simply embed the gradient computation into the stochastic gradient descent framework.

5 Experimental Results

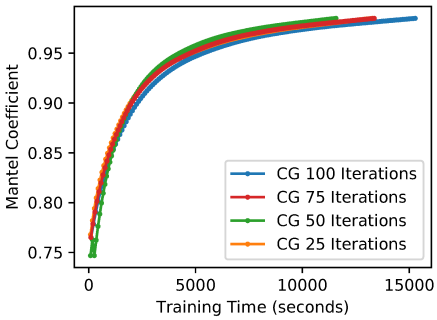
In ecological research, landscape-resistance costs are usually combinations of different factors such as roads, slope, elevation, land cover, and range limitations [Cushman *et al.*, 2006; McRae and Beier, 2007]. We generated multiple structured 10×10 resistance grids from Dilkina *et al.* (2013) to mimic these costs. Then, we computed a single resistance surface by adopting a linear parametrization assigning a specific weight to each factor. These resistance surfaces served as our ground-truth configurations.

5.1 Results for Learning Random Walks from Partial Trajectories

To comprehensively test our model performance in learning random walks from partial trajectories, we simulated an animal individual for a 1000 time-steps based on a unique linear parametrization of three structured instances. To emulate partial trajectories, we experimented with 4 different scenarios that consisted of 80%, 60%, 40%, and 20% data availability. For each scenario, we executed 5 different variations of CGInv to test the effectiveness of restricting the maximum number of conjugate gradient iterations. We also ran DirectInv, but the runtime was much slower than CGInv, so we omitted graphing the results.

Figure 2 represents the results of different experiments. To observe that training converges for all 5 variations of CGInv, the graph on the left plots the 40% data availability experiment as a representative example of the single-step KL divergence plateauing near 0. We measured the KL-divergence between the learnt and true probabilities of moving from one-step in every location and summed all the individual values into an aggregate measure called the single-step KL divergence. As expected, all measures between the probability transition matrices, hitting time matrices, and weight vectors improve as data availability scales upward. Through the graphs in Figure 2, we verified that our computational framework fitted with CGInv variations can effectively learn the ground-truth values in three different aspects: transition probabilities, hitting times, and the weight vector.

From a computational efficiency standpoint, we noticed that running CGInv with only 2 conjugate gradient iterations produces massive benefits in total run-time while maintaining similar accuracy to other training variations. The benefits of running smaller number of conjugate gradient iterations lie



	Single-Step KL-Divergence	Average Commuting Time Error %	Cosine Similarity	Runtime (s)
CG 2	0.302	7.48	0.883	>16000
CG 25	0.002	0.79	0.994	13273
CG 50	0.087	4.28	0.847	11571
CG 75	0.084	4.35	0.849	13362
CG 100	0.084	4.35	0.849	15278

Figure 3: The experimental results of Mantel coefficient optimization shows our models recover the ground-truth random walk models. The graph shows the training curves of CGInv under variations of maximum number of conjugate gradient iterations. CG 2 results were omitted because training failed to converge on the ground-truth value. The table on the right represents the KL-divergence of the final transition probability matrix, error between true and learnt commuting times, cosine similarity between true and learnt weight vectors, and the total runtime. Values with ">" failed to converge within a time-limit.

in the fact that one can compute many more gradient updates and the added stochasticity of the gradient updates can help in optimization. Additionally, optimization algorithms are observed to converge faster in terms of total computation if the computed gradients are estimates of the true gradient [Goodfellow *et al.*, 2016]. From this perspective, variations of CGInv with less number of conjugate gradient iterations have the potential to be more efficient and effective in training.

5.2 Mantel Coefficient Optimization Results

To execute Mantel optimization experiments, we first simulated tagging 100 animal individuals randomly initialized across 100 locations expressed as a 10×10 grid. The ground-truth resistance surface was a linear combination of 10 different structured instances. To compute the genetic distance matrix, we programmed a simple genetics simulator that generates Bray-Curtis distances between two animal individuals based on their true commuting time added with Gaussian noise. The magnitude of the Bray-Curtis distances were linearly proportional to the length of the commuting time. Since the Mantel coefficient detects a linear relationship between the two ecological distance matrices, we created the genetic distance matrix based on linear methods so our framework has an opportunity to recover the true resistance grid. To test the efficiency of CGInv, we initialized 5 variations of the training algorithm from CG 2 to CG 100 to recover the original set of weights.

The results of the Mantel optimization experiments are shown in Figure 3. We omitted the results of CG 2 from the graph because it failed to converge to the ground-truth value. However, our computational framework is successful in maximizing the Mantel coefficient for other variations of CGInv as training progresses. An interesting result in the table of Figure 3 is CG 25 obtains a remarkably accurate solution vector remarkably even though the training curve is masked by CG 75 and nearly identical to other CGInv variations. Compared to CG 100, 75, and 50, the output of CG 25 contains noisier solutions due to fewer allowed conjugate gradient iterations. Back-propagating through such solutions might yield more stochastic gradients that allow the training algorithm to converge on more optimal extremum.

Surprisingly, we observed that reducing the number of conjugate gradient iterations seems to have no substantial effect on the efficiency of CGInv. We performed subsequent experiments to measure the average number of conjugate gradient iterations on the initial epochs for CG 100, which is going to be made available in the full version of the paper. The average number of conjugate gradient iterations started at 50 and started dropping by 1 or 2 iterations every couple epochs or so. Due to this observation, we realized that the training progress across different variations of CGInv produced minimal improvements in training time efficiency.

6 Conclusion

Ecological models such as the Circuitscape provide novel and intuitive approaches in evaluating the role of landscape features in genetic differentiation and animal movement. However, the true capacity of these models can be unlocked only when we master the ability to fully utilize ecological data, computational power, and statistical learning. Our computational framework is one such example integrating core concepts from multiple disciplines.

Through the method of moment matching, we showed that our framework successfully learns random walks in two different ecological setups. Wrapping conjugate gradient as layers in a neural network allowed us to scale the efficiency of our approach as evidenced in experiments. Additionally, our computational framework can be readily deployed in real-world ecological studies. As future work, we look forward to embedding the learned random walk model in landscape optimization studies to fuel conservation efforts.

Acknowledgements

Devulapalli acknowledges the support from the Purdue Doctoral Fellowship. This research was supported by NSF grants IIS-1850243, CCF-1918327, CMMI-1763108, CMMI-1935451, as well as by Microsoft AI for Earth grants.

References

[Amos and Kolter, 2017] Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017.

[Bras *et al.*, 2013] Ronan Le Bras, Bistra N. Dilkina, Yexiang Xue, Carla P. Gomes, Kevin S. McKelvey, Michael K. Schwartz, and Claire A. Montgomery. Robust network design for multispecies conservation. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, 2013.

[Chandra *et al.*, 1996] Ashok K Chandra, Prabhakar Raghavan, Walter L Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity*, 6(4):312–340, 1996.

[Cushman *et al.*, 2006] Samuel A Cushman, Kevin S McKelvey, Jim Hayden, and Michael K Schwartz. Gene flow in complex landscapes: testing multiple hypotheses with causal modeling. *The American Naturalist*, 168(4):486–499, 2006.

[Dickson *et al.*, 2019] Brett G Dickson, Christine M Albano, Ranjan Anantharaman, Paul Beier, Joe Fargione, Tabitha A Graves, Miranda E Gray, Kimberly R Hall, Josh J Lawler, Paul B Leonard, et al. Circuit-theory applications to connectivity science and conservation. *Conservation Biology*, 33(2):239–249, 2019.

[Dilkina *et al.*, 2013] Bistra N. Dilkina, Katherine J. Lai, Ronan LeBras, Yexiang Xue, Carla P. Gomes, Ashish Sabharwal, Jordan Suter, Kevin S. McKelvey, Michael K. Schwartz, and Claire A. Montgomery. Large landscape conservation - synthetic and real-world datasets. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, (AAAI)*, 2013.

[Gomes, 2009] Carla P. Gomes. Computational Sustainability: Computational Methods for a Sustainable Environment, Economy, and Society. *The Bridge*, 39(4):5–13, 2009.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Inman *et al.*, 2013] Robert M. Inman, Brent L. Brock, Kristine H. Inman, Shawn S. Sartorius, Bryan C. Aber, Brian Giddings, Steven L. Cain, Mark L. Orme, Jay A. Fredrick, Bob J. Oakleaf, Kurt Alt, Eric A. Odell, and Guillaume Chapron. Developing priorities for metapopulation conservation at the landscape scale: Wolverines in the western united states. 2013.

[Legendre and Legendre, 2012] Pierre Legendre and Loic FJ Legendre. *Numerical ecology*, volume 24. Elsevier, 2012.

[Ling *et al.*, 2019] Chun Kai Ling, Fei Fang, and J. Zico Kolter. Large scale learning of agent rationality in two-player zero-sum games. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*, 2019.

[McClure *et al.*, 2016] Meredith L McClure, Andrew J. Hansen, and Robert M. Inman. Connecting models to movements: testing connectivity model predictions against empirical migration and dispersal data. *Landscape Ecology*, 31:1419–1432, 2016.

[McRae and Beier, 2007] Brad H McRae and Paul Beier. Circuit theory predicts gene flow in plant and animal populations. *Proceedings of the National Academy of Sciences*, 104(50):19885–19890, 2007.

[McRae *et al.*, 2008] Brad H. McRae, Brett G. Dickson, Timothy H. Keitt, and Viral B. Shah. Using circuit theory to model connectivity in ecology, evolution, and conservation. *Ecology*, 2008.

[McRae, 2006] Brad H McRae. Isolation by resistance. *Evolution*, 60(8):1551–1561, 2006.

[Sheldon *et al.*, 2010] Daniel Sheldon, Bistra Dilkina, Adam Elmachtoub, Ryan Finseth, Ashish Sabharwal, Jon Conrad, Carla Gomes, David Shmoys, William Allen, Ole Amundsen, and William Vaughan. Maximizing the spread of cascades using network design. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 517–526, 2010.

[Shewchuk, 1994] others Shewchuk, Jonathan Richard. An introduction to the conjugate gradient method without the agonizing pain, 1994.