Toroidal Grid Optimization via Gradient Descent

Jason Ken A

February 25, 2020

1 Abstract

Gradient descent is an algorithm to iteratively optimize (by either minimizing or maximizing) a continuous function. It does so by shifting the value of the paramaters in the direction of its gradient with respect to the function. In the context of minimization, it is defined as follows

$$\theta' = \theta - \alpha \frac{d}{d\theta} L(\theta)$$

where θ is the paramater to optimize to minimize the value of the function L, and α is a scaling factor. This equation can be generalized to optimize many paramaters by utilizing partial derivatives instead. Anyhow, it is obvious why a continuous function is required: gradients are only defined with continuous functions.

The main goal of this essay is to test whether a discrete loss function and its corresponding discrete paramaters can be generalized to a continuous loss function and continuous paramaters, with the purpose of finding optimum discrete variables. But what fun is it, without applications? So to make it more interesting, I chose to apply it to a programming contest by Al Zimmermann,[1] which had rendered analytical solutions to the problem impossible. The word "programming" within the contest name indicates that the problem was meant to be approached with algorithms that do not utilize gradients, such as hill climbing and simulated annealing[1]. But I wanted to do it with gradients! Because that's what my calculus classes are for.

The objective of Al Zimmermann's programming contest, is to rearrange discrete tokens within a discrete grid, to minimize the the a loss function which depends on the distances between these tokens on a toroidal surface. The intricacies of this problem are explained within the essay.

There are many techniques used within this essay to accomplish this goal. Among them are

1. Generalization of Euclidean distances to accomodate toroidal surfaces
2. Matric permutations to remove duplicate entries
3. "Superposition" to model discrete tokens being within multiple positions, with their respective "probabilities".[2]
4. Enforcement of probabilities (that they should sum to one), using the Sinkhorn-Knopp algorithm.[2]
5. Use of Jacobian matrices for gradient descent

[1]Which are beyond the scope of this essay

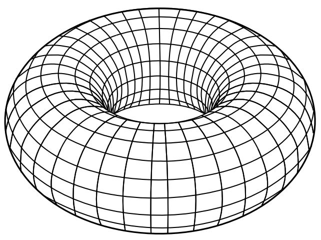
[2]Probabilities here does not refer to the chance of a random event occuring, but rather the "confidence" in which a token is in its position, or the extent in which the position affects the loss function

Contents

2 Problem Statement

2.1 Toroidal Graph

According to the problem description on AZsPCs,[1] the initial toroidal grid O is defined as an $N \times N$ grid of unique tokens which "wrap around" (the implications are addressed in 2.2.1). The token can be arbitrarily defined, as long as it is unique, but for the sake of simplicity (for reasons explained in subsubsection 2.2.1), in this essay, a token is of the form by $\boldsymbol{IJ}$, where $\boldsymbol{I}$ and $\boldsymbol{J}$ represent the alphabetic indices of the rows and columns respectively, eg. AC corresponds to the token in row 1, column 3, DF corresponds to row 4, column 6, and so on. For $N = 4$, the grid is shown in Figure 1b. The tokens outside of the square grid represent tokens which wrap around the edges, resembling a toroidal surface as shown in Figure 1a.



(a) A Simple Toroid by Yassine Mrabet[3]

	DA	DB	DC	DD	
AD	AA	AB	AC	AD	AA
BD	BA	BB	BC	BD	BA
CD	CA	CB	CC	CD	CA
DD	DA	DB	DC	DD	DA
	DA	DB	DC	DD	

(b) A 4×4 toroidal grid

Figure 1: Representations of a toroidal grid

2.2 Evaluation Function

The goal of the challenge (and hence this essay) is to create a new grid X — with tokens from the initial grid O to be rearranged arbitrarily — which minimizes a loss function (which quantifies error) which is computed with the following procedure:

1. For each unique pair of tokens (eg. AA, BA is equivalent to BA, AA, so BA, AA is not included), calculate the squared distance between them in the new grid,
2. Multiply it with the squared distance between the pair of tokens within the original grid
3. The loss is the sum of all of these products

2.2.1 Distance Metric

To evaluate the loss function, a distance metric between two tokens must be established. And to do that, we must define the position of a token in a coordinate system. This is where the methodology we used to define the unique tokens

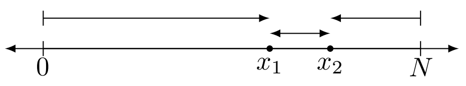


Figure 2: A one-dimensional diagram of toroidal distance, with the two arrows representing two possible distances

comes in handy; the coordinates of the tokens are simply the indices of the token *within the particular grid X, and not merely in O*. To avoid confusion between whether the rows or columns correspond to the x-coordinate or y-coordinate, a token within a grid, with indices (i, j) simply has the coordinate (i, j).[3]

With this definition, we can now define the distance metric. If two two-dimensional coordinates are defined as $s_1 = (x_1, y_1)$ and $s_2 = (x_2, y_2)$, the Euclidean distance d_{euclid} is defined as

$$d_{euclid}(s_1, s_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = \sqrt{(\Delta_{euclid}x)^2 + (\Delta_{euclid}y)^2} \quad (1)$$

and the squared Euclidean distance d^2 evaluates to

$$d^2_{euclid}(s_1, s_2) = (\Delta_{euclid}x)^2 + (\Delta_{euclid}y)^2 \quad (2)$$

simplifying much of the computation.

But to generalize Euclidean distance to the distance on a toroidal surface, we must consider several implications. First of all, Δx or Δy can have 2 possible values. Let us first consider the one-dimensional case. A toroidal surface of length N is illustrated in Figure 2. It can be shown that the corresponding possible values for Δx are as follows:

$$\begin{aligned} \Delta_1(x) &= x_2 - x_1 \\ \Delta_2(x) &= (x_1 - 0) + (N - x_2) = x_1 + N - x_2 \end{aligned}$$

To obtain a general equation which works with both $x_1 > x_2$ and $x_1 < x_2$ (as we cannot assume $x_1 < x_2$), we can write

$$\begin{aligned} \Delta_1(x) &= |x_2 - x_1| \\ \Delta_2(x) &= \min(x_1, x_2) + N - \max(x_1, x_2) \end{aligned}$$

where $|a|$ is the absolute value of a, $\min(a, b)$ and $\max(a, b)$ are defined as the minimum and maximum between the values of a and b respectively. They can be defined with piece-wise functions as follows:

$$\min(a, b) = \begin{cases} a & \text{if } a \leq b \\ b & \text{if } a > b \end{cases}$$

$$\max(a, b) = \begin{cases} a & \text{if } a \geq b \\ b & \text{if } a < b \end{cases}$$

min and max are used to determine the “left-most” and the “right-most” numbers.

[3]Note that this means we are not adhering to Cartesian coordinates, where the index (i, j) would have the coordinate (j, i).

Since, the distance can only have one value, it is defined as the minimum of the two possible values:

$$\Delta x = \min\left(\Delta_1(x), \Delta_2(x)\right)$$

Generalizing it to 2 dimensions, we get

$$\begin{aligned} d(s_1, s_2) &= \sqrt{(\Delta x)^2 + (\Delta y)^2} \\ d^2(s_1, s_2) &= (\Delta x)^2 + (\Delta y)^2 \end{aligned}$$

where d^2 is the distance metric to be used in our calculations.

2.2.2 Token Comparisons

Within this essay, a comparison is defined as the toroidal distance between two tokens within a toroidal grid. To be able to perform comparisons for every possible pair of tokens, a grid of comparisons on a toroidal grid (denoted by X) is required. Let $C(X)$ be the comparison grid of X. A viable approach would be to represent the comparison in a 4-dimensional grid (or tensor), where every token in the grid, is compared to every token. An example of the structure of the comparison for a 2×2 grid is shown in Figure 3a. The white boxes denote duplicate comparisons (eg. $\substack{BA \\ AA}$ is identical to $\substack{AA \\ BA}$), because the distance is invariant to the order of the tokens.

But since removing the irregularly shaped duplicate entries is not a trivial task with a 4-dimensional representation — especially if we extend beyond a $N = 2$ grid — we can simplify the problem by reducing the dimensionality of the grid: from an $N \times N \times N \times N$ grid to a $N^2 \times N^2$ grid. The elements within the 4-dimensional matrix are in the form $C_{i,j,k,l}$, and the 2-dimensional grid in the form $C_{ij,kl}$, but both represent the same thing: a the toroidal distance between the token $\boldsymbol{IJ}$ and $\boldsymbol{KL}$. The two-dimensional representation of Figure 3a is shown in Figure 3b. By doing so, we can easily remove the duplicate entries by multiplying it with an upper triangular matrix, further explained in subsubsection 2.2.3. Let $C(X)$ denote a function mapping from the input toroidal grid X and returning the 2d-comparison grid C.

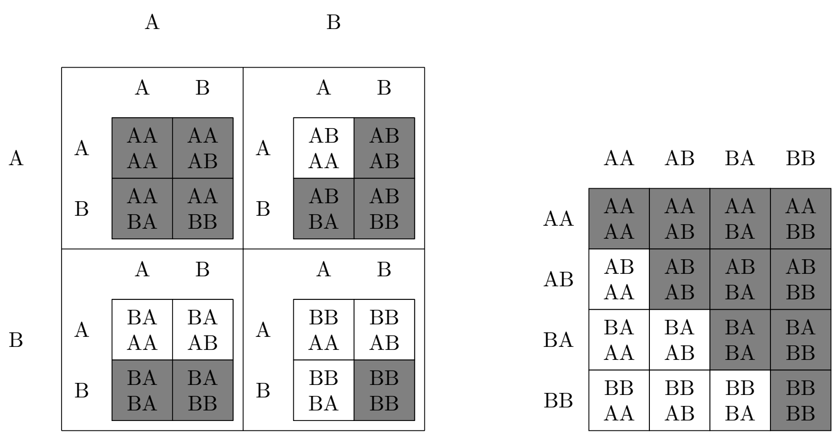
2.2.3 Loss Function as Matrix Multiplications

Let $\odot$ denote the element-wise matrix multiplication (eg. $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \odot \left(\begin{smallmatrix} e & f \\ g & h \end{smallmatrix}\right) = \left(\begin{smallmatrix} a\cdot e & b\cdot f \\ c\cdot g & d\cdot h \end{smallmatrix}\right)$), and O denotes the original grid. The loss function for toroidal grid X is equal to

$$\begin{aligned} L(x) &= \sum_{i}^{N^2} \sum_{j}^{N^2} C(X)_{ij} C(O)_{ij} U_{ij} \\ &= \sum C(X) \odot C(O) \odot U \end{aligned}$$

where $C(X), C(O),$ and U are matrices of the form $N^2 \times N^2$, and U_{ij} is defined with the piece-wise defined function:

$$U_{ij} = \begin{cases} 1, & j \geq i \\ 0, & j < i \end{cases}$$



(a) A 4-dimensional comparison with elements in the form $A_{i,j,k,l}$ (b) A 2-dimensional comparison with elements in the form $A_{ij,kl}$

Figure 3: Tensors $C(X)$ representing comparison grids of a 2×2 toroidal grid, where every element represents the toroidal distance between X_{ij} and X_{kl}. Shaded cells denote unique comparisons.

an example the matrix U with shape 4×4 is as follows:

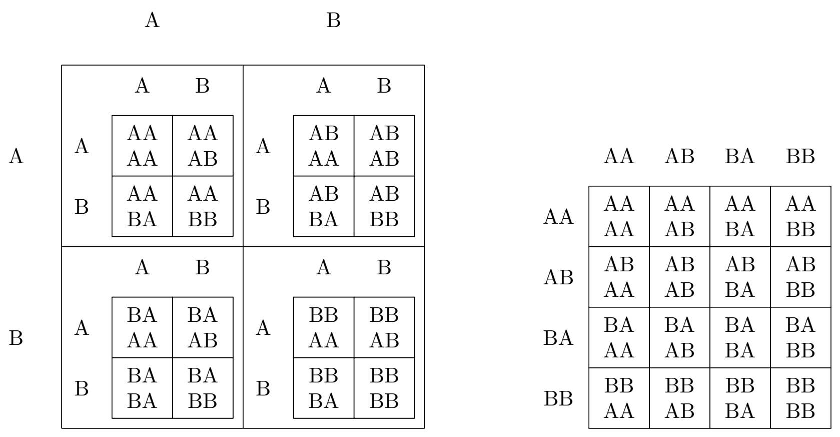
$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying

3 Superposition

Since the entries into the toroidal grid are discrete (eg. AA resolves to discrete coordinates within the grid), it is not possible to optimize the loss function. Therefore, relaxing the constraints to enable superposition – here defined as having a token being in multiple positions at once each with its own "probabilities" – is essential. In this essay, "probability" will not refer to the likeliness of a random event, but rather, the confidence of a token in its posiiton.

A simple method of allowing superposition, is by allowing any position to have any token value, which again, can be visualized as a 4-dimensional matrix, and 2-dimensional matrix, as seen in Figure 4. But this time, the elements of the grid do not represent the distances between tokens. But rather, they represent the probability of a token being placed in a certain position. Further constraints to limit the total probability to 1 will be discussed in a later section. Note that now the $y-$axis represents various positions (the same token in various places), and the $x-$axis represents the possible token values (all of which lie on the same toroid grid cell).



		A		B	
		A	B	A	B
A	A	AA AA	AA AB	AB AA	AB AB
	B	AA BA	AA BB	AB BA	AB BB
B	A	BA AA	BA AB	BB AA	BB AB
	B	BA BA	BA BB	BB BA	BB BB

(a) A_{ijkl}, a 4-dimensional superposition

	AA	AB	BA	BB
AA	AA AA	AA AB	AA BA	AA BB
AB	AB AA	AB AB	AB BA	AB BB
BA	BA AA	BA AB	BA BA	BA BB
BB	BB AA	BB AB	BB BA	BB BB

(b) $A_{ij,kl}$ a 2-dimensional superposition

Figure 4: Tensors S representing superpositions of a 2×2 toroidal grid, where every element S_{ijkl} represents the probability of the token KL being in the of position IJ in the original grid

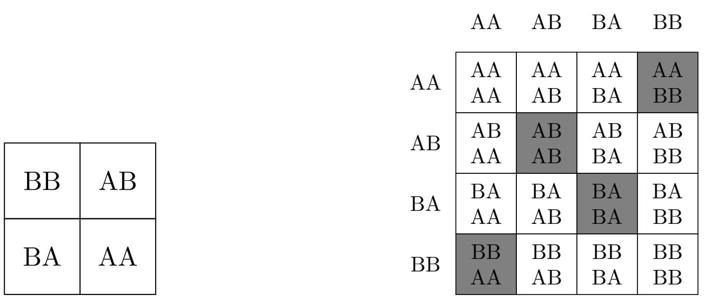
An example is shown in Figure 5

3.0.1 Generalization of the Loss Function

Defining the loss function for this formulation requires us to compare every value, in every position, to every value in every position. And we must do so, while taking both of their confidence's into account (ie. the distance metric should be scaled by their confidence). Therefore, the distance should be scaled with

$$S_{ab}S_{cd} \tag{3}$$

The distance between these two probabilities is defined as $C_{a,c}$, because only



(a) A_{ijkl}, an example toroidal grid

(b) $A_{ij,kl}$ superposition of the grid on the left

Figure 5: Shaded cells represent cells with probability 1, the rest have probability 0

the indices a and c correspond to positions on the grid (thus being important in calculating distance), whereas b and d correspond to only the token itself.

Alike with the situation in subsubsection 2.2.2, we must prevent duplicate comparisons between values (not position), for example: each of $[\genfrac{}{}{0pt}{}{AA}{AA}, \genfrac{}{}{0pt}{}{AB}{AA}, \genfrac{}{}{0pt}{}{BA}{AA}, \genfrac{}{}{0pt}{}{BB}{AA}]$ should be compared to $[\genfrac{}{}{0pt}{}{AA}{AB}, \genfrac{}{}{0pt}{}{AB}{AB}, \genfrac{}{}{0pt}{}{BA}{AB}, \genfrac{}{}{0pt}{}{BB}{AB}]$, but not vice versa, because the values (not the positions) will have been compared already. Again, to do so, we must construct an upper triangular matrix U, of the shape $N^2 \times N^2$.

Therefore, the loss function can be written as

$$L(X) = \sum_{a}^{N^2} \sum_{b}^{N^2} \sum_{c}^{N^2} \sum_{d}^{N^2} S_{a,b} S_{c,d} C_{a,c} C_{b,d} U_{a,c} \tag{4}$$

$\sum_a^{N^2}$	Represents an iteration over Source Positions
$\sum_b^{N^2}$	Represents an iteration over Source Values
$\sum_c^{N^2}$	Represents an iteration over Target Positions
$\sum_d^{N^2}$	Represents an iteration over Target Values
$S_{a,b}$	Represents the Probability of the Source Position and Value
$S_{c,d}$	Represents the Probability of the Target Position and Value
$C_{a,c}$	Represents the Distance between Source and Target Position
$C_{b,d}$	Represents the Distance between the Source and Target Values in the original grid
$U_{a,c}$	Upper triangular matrix to remove redundant value comparisons

4 Optimization

4.1 Constraints

To be able to properly model probability, the sum of the probabilities of each token in all of its positions must be equal to one. Similarly, the sum of the probabilities of each position, in all of its values must be equal to one. Therefore, the sums of each of the rows and each of the columns must be equal to one, this is called a Doubly Stochastic Matrix. To be able to enforce this constraint, the Sinkhorn-Knopp algorithm was developed.

4.1.1 Sinkhorn-Knopp Algorithm

Given K an $n \times n$ non-negative matrix, by repetitively normalizing the rows and columns, a doubly stochastic matrix can be obtained. A single iteration can be seen through the following equation

$$K' = \left(\sum_{i}^{N} K_{ij} \right)^{-1} K \left(\sum_{j}^{N} K_{ij} \right)^{-1} \tag{5}$$

4.1.2 Non-negative Matrices

Since Sinkhorn-Knopp requires a non-negative matrix, and because negative probabilities make no sense, at every iteration of Sinkhorn-Knopp and Gradient Descent, the values have to be maintained above 0.

$$K'_{ij} = \max(K_{ij}, 0) \tag{6}$$

4.2 Gradient Descent

Gradient Descent is an algorithm to iteratively optimize a convex function, with knowledge of the derivative of the function with respect to all of the function parameters θ. In the case of optimizing a loss function, steps must be taken in the direction opposite to the gradient, therefore, the equation is as follows

$$\theta' = \theta - \alpha \frac{\delta}{\delta\theta} L(\theta) \tag{7}$$

where α is a parameter determining how big of a change there is between iterations of gradient descent.

4.2.1 Derivative of Loss Function coming Soon

5 Evaluation

5.1 Graphs of Loss over Time

5.2 Comparison to State of the Art Lower Bounds

5.3 Limitations of Computational Complexity

5.4 Limitations of How to Discretive Superpositions

5.5 Summary

References

[1] Al Zimmermann. *Reversing Nearness.* URL: `http://azspcs.com/Contest/Nearness` (visited on 02/25/2020).

[2] Richard Sinkhorn and Paul Knopp. “Concerning nonnegative matrices and doubly stochastic matrices”. In: *Pacific Journal of Mathematics* 21.2 (1967), pp. 343–348.

[3] Yassine Mrabet. *A simple Torus.* 2007. URL: `https://upload.wikimedia.org/wikipedia/commons/c/c6/Simple_Torus.svg` (visited on 02/25/2020). License: Creative Commons BY-SA.