Sinkhorn Optimal Transport Algorithm

Class Definition

The class `SinkhornOptimalTransport` is designed to solve the optimal transport problem using entropy regularization.

Properties

- **NumPoints**: The number of points, which equals the length of the supply array.

$$\text{NumPoints} = \text{len}(\text{Supply})$$

- **CostMatrix**: A matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$, where each entry C_{ij} represents the cost of transporting one unit of supply from source i to destination j.

- **Supply**: A vector $\mathbf{a} \in \mathbb{R}^n$, representing the total supply available at each source.

- **Demand**: A vector $\mathbf{b} \in \mathbb{R}^n$, representing the total demand required at each destination.

- **TransportMatrix**: The resulting transport plan, denoted as $\mathbf{T} \in \mathbb{R}^{n \times n}$, where each entry T_{ij} is the amount transported from source i to destination j.

Constructor

The constructor initializes the following:

$$\text{NumPoints} = \text{len}(\text{Supply}), \quad \mathbf{T} \in \mathbb{R}^{n \times n} \text{ initialized as zeros.}$$

Sinkhorn Algorithm

Inputs

- ϵ: Entropy regularization parameter.

- maxIter: Maximum number of iterations.

- tol: Tolerance for convergence.

Steps

1. Initialization

- Initialize scaling vectors:

$$u_i = 1.0, \quad v_j = 1.0 \quad \forall i, j$$

- Compute the kernel matrix $\mathbf{K}$:

$$K_{ij} = \begin{cases} 0, & \text{if } C_{ij} = \frac{\text{MAX\_VALUE}}{2} \\ \exp\left(-\frac{C_{ij}}{\epsilon}\right), & \text{otherwise.} \end{cases}$$

2. Iterative Updates The algorithm alternates between updating the scaling vectors u and v.

- Update u:

$$u_i = \begin{cases} \frac{a_i}{\sum_j K_{ij} v_j}, & \text{if } \sum_j K_{ij} v_j > 0 \\ 0, & \text{otherwise.} \end{cases}$$

- Update v:

$$v_j = \begin{cases} \frac{b_j}{\sum_i K_{ij} u_i}, & \text{if } \sum_i K_{ij} u_i > 0 \\ 0, & \text{otherwise.} \end{cases}$$

These updates continue for a maximum of maxIter iterations or until a convergence criterion (not implemented in the code) is met.

3. Compute Transport Matrix Once the iterative updates are complete, compute the transport plan:

$$T_{ij} = u_i \cdot K_{ij} \cdot v_j \quad \forall i, j$$

Key Points

- **Entropy Regularization**: The parameter ϵ controls the smoothness of the transport plan. Smaller ϵ values result in a more accurate transport plan but can lead to numerical instability.
- **Efficiency**: The Sinkhorn algorithm is computationally efficient compared to traditional linear programming methods for optimal transport.
- **Applications**:
  - Comparing probability distributions (e.g., Wasserstein distance).
  - Image processing.
  - Supply chain logistics.

Potential Improvements

- **Convergence Check**: Implement a convergence criterion to stop the iteration early if the changes in u and v fall below a threshold.
- **Edge Case Handling**: Handle cases where supply or demand is zero more robustly.
- **Scalability**: Optimize the implementation for large-scale problems using specialized matrix libraries or GPU acceleration.