Scaling Up Ordinal Embedding: A Landmark Approach
ICML 2019

Jesse Anderton
Northeastern University
jesse@ccs.neu.edu
Spotify
janderton@spotify.com

Javed Aslam
Northeastern University
jaa@ccs.neu.edu
Suppose we want to perform image search by learning a pairwise distance between pixel vectors, with smaller distances between images with more similar labels.

Image a: architecture, building

Image b: escalator, architecture

Image c: flower, plant
Embedding with Features and Triplets: Metric/Kernel Learning

• We can define the pixel vector for image $i$ as $X_i$

• We can induce similarity triplets like $(a, b, c)$ from labels to indicate that image $a$ should be closer to image $b$ than to image $c$

• We can then learn a metric $\phi$ defined on $X$ which preserves this ordering

Given $m$-dimensional features for $n$ objects $X \in \mathbb{R}^{n \times m}$ and similarity triplets $T \subset [n]^3$, find metric $\phi : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ s.t. $(a, b, c) \in T \Rightarrow \phi(X_a, X_b) < \phi(X_a, X_c)$
Assumptions of Metric Learning

\[(a, b, c) \in T \Rightarrow \phi(X_a, X_b) < \phi(X_a, X_c)\]

• Implicitly assumes that \(T\) derives from an unknown metric space \((Y, \sigma)\).

\[\exists Y \in \mathbb{R}^{n \times d}, \sigma : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R} \text{ s.t. } (a, b, c) \in T \Rightarrow \sigma(Y_a, Y_b) < \sigma(Y_a, Y_c)\]

• Critically, assumes \(Y\) is a transformation of the observable features \(X\), so we only need to recover the metric.

• What if image labels include side information not observable from pixels, e.g. copyright license, photographer, date/time, event being photographed, information about people in photo, …?

• No \(\phi\) can approximate \(\sigma\) well when \(Y\) contains a lot of information missing from \(X\).
Embedding with Only Triplets: Ordinal Embedding

• In Metric Learning, we fix the \textbf{representation} and learn a \textbf{metric} to satisfy triplets.

• In Ordinal Embedding, we fix the \textbf{metric} (Euclidean distance) and learn the \textbf{representation} that satisfies triplets.

Given target dimension $d$ and similarity triplets $T \subset [n]^3$, find positions $X \in \mathbb{R}^{n \times d}$ s.t. $(a, b, c) \in T \Rightarrow \|X_a - X_b\| < \|X_a - X_c\|$
Embedding with Only Triplets: Ordinal Embedding

Given target dimension $d$ and similarity triplets $T \subset [n]^3$,
find positions $X \in \mathbb{R}^{n \times d}$ s.t. $(a, b, c) \in T \Rightarrow \|X_a - X_b\| < \|X_a - X_c\|

**Uniqueness Theorem [Kleindessner and von Luxburg, 2014; Arias-Castro 2015]:** Under certain conditions, with enough points, any $n \times d$ matrix $X$ which satisfies $T$ must recover the true latent representation $Y$ up to similarity transformations and bounded perturbation ($\varepsilon \to 0$ as $n \to \infty$).
## Metric Learning vs. Ordinal Embedding

### Metric Learning:
- Triplets used to constrain metric.
- Assumes features adequate to compute metric; poor performance otherwise.
- Rich models to transform features; large literature on possible approaches.
- Generalizes easily to new instances.
- Scales well to many objects in high dimension.

### Ordinal Embedding:
- Triplets used to infer latent representation.
- Recovers adequate features for Euclidean metric of fixed dimension, if possible.
- No explicit features to transform; relatively few optimization objectives.
- Does not generalize without new triplets.
- **Prior methods do not scale** past tens of thousands of objects.
Scalability Problems

- Poor scalability has limited the usefulness of Ordinal Embedding.
- Many existing methods are $\Omega(n^2)$.
- All known $O(|T|)$ objectives fail to find global optima starting around $n$ in the 10,000’s.
- For larger problems, embedding takes days or weeks and finds bad local minima.
- Goal: Embed large datasets accurately with $O(n)$ operations.

Representative Result Sizes in the Literature

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$n$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNM-MDS (JMLR 2007)</td>
<td>55</td>
<td>2</td>
</tr>
<tr>
<td>Crowd Kernel (ICML 2011)</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>t-STE (MLSP 2012)</td>
<td>1,000</td>
<td>2</td>
</tr>
<tr>
<td>SOE / LOE (ICML 2014)</td>
<td>5,000</td>
<td>2</td>
</tr>
<tr>
<td>ASAP LOE (MLSP 2015)</td>
<td>50,000</td>
<td>2</td>
</tr>
</tbody>
</table>
A Landmark Approach

Idea: Accurately embed a small subset, providing fixed reference distances to use to embed remaining points.

1. Phase one (L-SOE Phase, first $m$ points)
   - Goal is to produce highly accurate small-to-medium scale ordinal embedding.

2. Phase two (LLOE Phase, remaining $n - m$ points)
   - Goal is to embed remaining points in $O(n)$ time, with accuracy depending on accuracy of L-SOE phase.
A Landmark Approach

1. **Phase one** (L-SOE Phase, first \(m\) points):

   Pick random \(m\) points from \(|n|\).
   Pick \(L\) of \(m\) points as landmarks.
   Sort \(m\) points by distance to each \(L\) point.
   Sort \(L\) points by distance to each \(m\) point.
   Embed resulting triplets with SOE.

Contribution: Show empirically that small-to-medium scale ordinal embedding is solved with novel combination of existing methods.

Given accurate positions for \(l_1, l_2, l_3, a,\) and \(c,\) \(b\) (not in subset) will be tightly constrained.
Uniform Sample from Ball in $\mathbb{R}^{30}$

GMM in $\mathbb{R}^{30}$

Phase One Performance in $\mathbb{R}^{30}$

Times on 2013 MacBook Pro, 2 GHz Core i7.
A Landmark Approach

2. **Phase two** (LLOE Phase, remaining $n - m$ points, independently and in parallel):

Pick $2(d+1)$ subset points as landmarks by FFT
Insert $b$ into landmark orderings of subset
Embed $b$ into shell intersection:

$$\mathcal{L}(X_b; l, r, m) = \sum_{i=1}^{2(d+1)} \max \left( 0, (\|X_b - X_l\| - r_i)^2 - m_i^2 \right)$$

Contribution: Novel, efficient approach for adding points to an existing ordinal embedding.

Each landmark $l_i$ has corresponding shell radius $r_i$ and width $m_i$. 
Phase two: LLOE embedding for point $b$

\[
\delta(l_1,a) < \delta(l_1,b) < \delta(l_1,c), \quad \delta(l_2,c) < \delta(l_2,b) < \delta(l_2,a), \quad \delta(l_3,a) < \delta(l_3,b) < \delta(l_3,c)
\]
A Landmark Approach

2. **Phase two** (LLOE Phase, remaining $n - m$ points, independently and in parallel):

   Pick $2(d+1)$ points as landmarks by FFT
   Insert $b$ into landmark orderings of subset
   Embed $b$ into shell intersection

**Theorem [Embedding Quality]:** Let $X \subset \mathbb{R}^d$ be $n$ i.i.d. draws from a Lipschitz-smooth measure over a bounded, connected subspace of $\mathbb{R}^d$. Let $S \subset X$ be a uniformly-sampled subset of size $m \gg d$ with known positions, and let $A \subset S$ be a set of at least $d+1$ anchors chosen by farthest-first traversal. For any $x \in X$, let $x' \in \mathbb{R}^d$ be any point satisfying the distance constraints to the members of $A$ imposed by the order of $S \cup \{x\}$. Then there is a constant $c \in \mathbb{R}$ such that for $\delta \in (0,1)$, with probability at least $1 - \delta$,

$$||x - x'|| \leq \frac{cd}{m} \ln \frac{m}{\delta}$$
Uniform Sample from Ball in $\mathbb{R}^{30}$

Phase Two Performance in $\mathbb{R}^{30}$

Used L-SOE with $m = 1,000$, $L = 100$
Comparison to the Literature

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$n$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNM-MDS (JMLR 2007)</td>
<td>55</td>
<td>2</td>
</tr>
<tr>
<td>Crowd Kernel (ICML 2011)</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>t-STE (MLSP 2012)</td>
<td>1,000</td>
<td>2</td>
</tr>
<tr>
<td>SOE / LOE (ICML 2014)</td>
<td>5,000</td>
<td>2</td>
</tr>
<tr>
<td>ASAP LOE (MLSP 2015)</td>
<td>50,000</td>
<td>2</td>
</tr>
<tr>
<td>Phase One (L-SOE)</td>
<td>8,000</td>
<td>30</td>
</tr>
<tr>
<td>Phase Two (LLOE)</td>
<td>1,000,000</td>
<td>30</td>
</tr>
</tbody>
</table>
Phase Two Performance in $\mathbb{R}^{30}$

Used L-SOE with $m = 1,000$, $L = 100$

---

**MNIST Digits in $\mathbb{R}^{30}$**

- Plot 1: Median CPU Time (seconds) vs $n$ (thousands)
- Plot 2: Error Probability vs $n$ (thousands)

**20 Newsgroups in $\mathbb{R}^{30}$**

- Plot 3: Median CPU Time (seconds) vs $n$ (thousands)
- Plot 4: Error Probability vs $n$ (thousands)
Thank You!

Implementation at:  
https://github.com/jesand/lloe

Find me at my poster:  
Pacific Ballroom #227