Wasserstein Wormhole: Scalable Optimal Transport Distance with Transformers.

Machine Learning in Practice Reading Group

Duke B&B

February 28, 2025

Presented by Aditya Parekh

ChatGPT Art

"Draw me a Wasserstein Wormhole"

Introduction: Optimal Transport

- Given probability measure **P**, transport to probability measure **Q**. If the transport is optimal, you have achieved Optimal Transport. (yay!)
- More formally, define two probability measures $P_x = \sum_{i=1}^{n} \mu \delta_{x_i}$ and $P_y = \sum_{j=1}^{m} \nu_j \delta_{y_j}$. Let

$$U(\nu, \mu) = \left\{ P \in \mathbf{R}_+^{n,m} : \sum_i P_{i,j} = \mu_j, \sum_j P_{i,j} = \nu_i \right\}$$

be the set of all valid mappings between P_x and P_y. Let $c(x_i, y_j)$ be a cost function for mapping the point x_i to y_j. The Wasserstein Distance between P_x and P_y is defined as

$$W(P_x, P_y) = \min_{P \in U(\nu,\mu)} \sum_{i,j} c(x_i, y_j) P_{i,j}$$

Background: Optimal Transport

Optimization Problem:

$$\min_{P \in U(\nu,\mu)} \sum_{i,j} c(x_i, y_j) P_{i,j}.$$

Very difficult to solve.

Instead, one can minimize the entropically-regularized objective function

$$\min_{P \in U(\nu,\mu)} \sum_{i,j} c(x_i, y_j) P_{i,j} - \varepsilon H(P),$$

where $H(P) = -\sum_{i.j} P_{i,j} (\log P_{i,j} - 1)$.

Background: Optimal Transport: Exact Solution

Define

$$W_\varepsilon(P_x, P_y) = \min_{P \in U(\nu,\mu)} \sum_{i,j} c(x_i, y_j) P_{i,j} - \varepsilon H(P).$$

To find $W_\varepsilon(P_x, P_y)$, use Sinkhorn algorithm (not described). One iteration of the algorithm takes runtime $O(N^2 n^2)$ for N distributions with sample space of size n.

That is, takes quadratic time, not feasible for any sizable dataset. **Need efficient but accurate approximation**.

Background: Optimal Transport: Approximate Solution

Previous work on approximate optimal transport:

- DWE [1] - Siamese network: train a network to predict output of the other network and vice versa, metric learning approach
- DiffusionEMD [2] - Diffusion model based approach to approximating Wasserstein distance; prove approximation bounds in the case where the two probability distributions (i.e., the data) lie on a Riemannian Manifold.
- Neural Optimal Transport [3] - Monte Carlo based approximation of optimal transport plan (presented by Mengying last term)

Good general background on Optimal Transport: [4].

Here the authors take an approach inspired by **multidimensional scaling**.

Background: Multidimensional Scaling

Multidimensional Scaling (MDS) is a dimensionality reduction method for representing higher-dimensional Euclidean spaces in lower dimensions.

Idea is to preserve pairwise distances as faithfully as possible.

Definition: An N-by-N distance matrix D is *Euclidean* if and only if it is hollow ($D_{ii} = 0, \forall i$) and the criterion matrix

$$F = -JDJ$$

is positive semidefinite. Here, J is the centering matrix; $J = I_N - \frac{\mathbf{1}_N \mathbf{1}_N^T}{N}$.

Background: Multidimensional Scaling

Given a distance matrix D (for points in arbitrary dimensions), MDS computes embeddings in $\alpha_1, \ldots, \alpha_N \in \mathbf{R}^d$ by minimizing the *cumulative stress*:

$$\mathcal{L} = \min_{\alpha} \sum_{i,j} (||\alpha_i - \alpha_j||_2^2 - D_{i,j})^2.$$

If D is Euclidean, then $\mathcal{L}$ decreases as d increases. In fact, if D is Euclidean, then $\exists \alpha_i \in \mathbf{R}^d$ s.t. $\mathcal{L} = 0$ for all $d \geq N$.

However, if D is not Euclidean, then $\mathcal{L}$ can *increase* with d. In general, Wasserstein distances are **not Euclidean**.

The authors extend classical MDS theory to prove upper and lower bounds on their approximation error.

Methods

Consider a sample space of N point clouds $\Omega = \{X_1, X_2, \ldots, X_N\}$, where X_i contains n_i points in $\mathbf{R}^d$.

Goal: train an encoder $T(X)$ such that for two point clouds X_i, X_j,

$$||T(X_i) - T(X_j)||_2^2 \approx OT(X_i, X_j),$$

where $OT(X_i, X_j)$ is the optimal transport (Wasserstein) distance between the two point clouds X_i and X_j.

Note that in this problem formulation, the encoder T must be able to take an input of variable size.

Methods

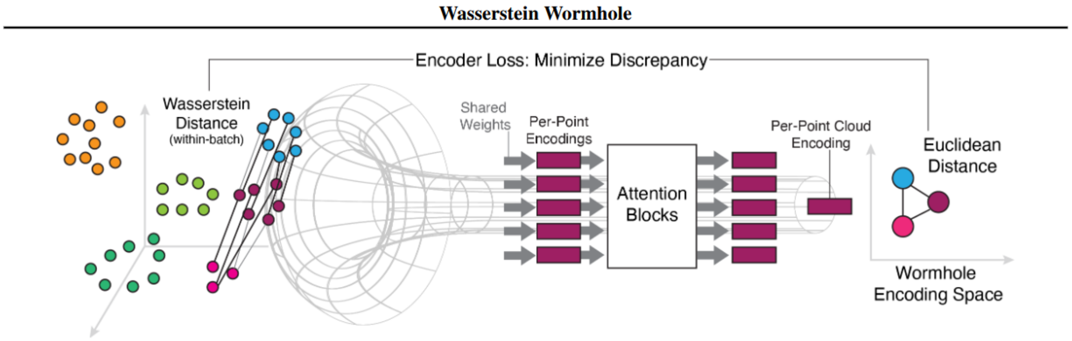
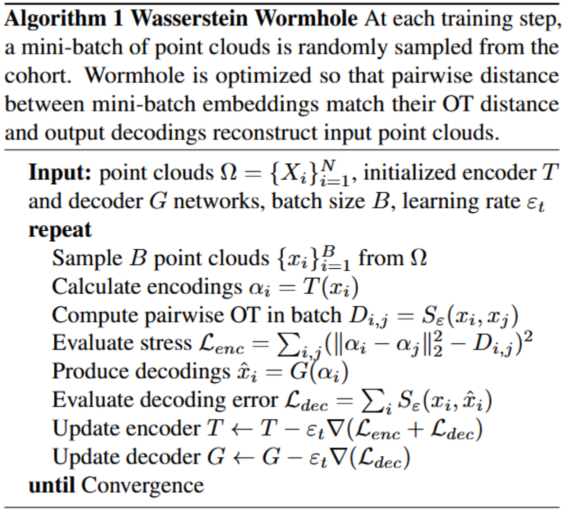


Figure 1. **Schematic of Wasserstein Wormhole**. Empirical distributions (point clouds) are passed through a transformer to produce per point-cloud vector embeddings such that the Euclidean distance between embeddings match the pairwise Wasserstein distance between point clouds. Since computation of OT distances is laborious, Wormhole is optimized by mini-batches to minimize the discrepancy between the embedding pairwise distances and the pairwise Wasserstein distances of the batch point clouds. The Wormhole decoder (not shown) is a second transformer trained to reproduce the input point clouds from the embedding by minimizing the OT distance between input and output.

Methods



Algorithm 1 Wasserstein Wormhole At each training step, a mini-batch of point clouds is randomly sampled from the cohort. Wormhole is optimized so that pairwise distance between mini-batch embeddings match their OT distance and output decodings reconstruct input point clouds.

```
Input: point clouds $\Omega = \{X_i\}_{i=1}^{N}$, initialized encoder $T$
and decoder $G$ networks, batch size $B$, learning rate $\varepsilon_t$
repeat
    Sample $B$ point clouds $\{x_i\}_{i=1}^{B}$ from $\Omega$
    Calculate encodings $\alpha_i = T(x_i)$
    Compute pairwise OT in batch $D_{i,j} = S_\varepsilon(x_i, x_j)$
    Evaluate stress $\mathcal{L}_{enc} = \sum_{i,j}(\|\alpha_i - \alpha_j\|_2^2 - D_{i,j})^2$
    Produce decodings $\hat{x}_i = G(\alpha_i)$
    Evaluate decoding error $\mathcal{L}_{dec} = \sum_i S_\varepsilon(x_i, \hat{x}_i)$
    Update encoder $T \leftarrow T - \varepsilon_t \nabla(\mathcal{L}_{enc} + \mathcal{L}_{dec})$
    Update decoder $G \leftarrow G - \varepsilon_t \nabla(\mathcal{L}_{dec})$
until Convergence
```

Methods

Why use transformers?

- Must be able to take in variable input size.
- Must be permutation equivariant. That is, the order of the input points used to compute the empirical distribution should not change the computed distance.

Both properties are satisfied by transformers.

Methods

Permutation equivariance of transformers: For a given block X, we must have $A(X) = A(PX)$ for any permutation matrix P. Here, $A(X)$ is the attention matrix; i.e. $A_{i,j}$ is the attention between X_i and X_j.

Proof of permutation equivariance:

Note that $A(X) = \text{softmax}\left(XW_qW_k^TX^T/\sqrt{d}\right)XW_V$. Let P be any permutation matrix.

$$\begin{aligned}
A(PX) &= \text{softmax}\left(PXW_qW_k^TX^TP^T/\sqrt{d}\right)PXW_V \\
&= P\text{softmax}\left(XW_qW_k^TX^T/\sqrt{d}\right)P^TPXW_V \\
&= P\text{softmax}\left(XW_qW_k^TX^T/\sqrt{d}\right)XW_V \\
&= PA(X).
\end{aligned}$$

Hence any invariant function of $A(X)$, such as average-pooling, will be invariant under permutation of the input data.

Methods

The authors also give upper and lower approximation bounds for MDS-approximation of Wasserstein distances.

Optimization problem:

$$\mathcal{L} = \min_{\alpha} \sum_{i,j} (||\alpha_i - \alpha_j||_2^2 - D_{i,j})^2.$$

The problem as stated is non-convex. However, can be transformed into a convex optimization problem by constraining the solution to a convex set:

$$\mathcal{L} = \min_{\hat{D}} \sum_{i,j} (\hat{D}_{i,j} - D_{i,j})^2$$

s.t.

- C1: $\hat{D}$ is positive definite
- C2: $\hat{D}$ is hollow
- C3: $\hat{D}_{i,j} \geq 0$

The optimal $\hat{D}$ can be found using a projected gradient descent algorithm (Algorithm 2 in the paper, not described here).

Methods

For this convex optimization problem, the authors derive the following lower and upper bounds (Section 5). First, let $F = -JDJ$ be the criterion matrix of the target distance matrix, with eigendecomposition $\{\lambda_i, v_i\}_{i=1}^{N}$. Let $\mathcal{L}^*$ be the solution to the convex optimization problem in the previous slide. Then $\mathbf{L} \leq \mathcal{L}^* \leq \mathbf{U}$, where

$$\mathbf{L} = \sum_{i:\lambda_i<0} \lambda_i^2$$

and

$$\mathbf{U} = \sum_{i,j} (\Delta g_i - \Delta g_j)^2 + \sum_{i:\lambda_i<0} \lambda_i^2$$

where

$$\Delta g_i = \sum_{j:\lambda j<0} \lambda_j \cdot (v_{ij})^2.$$

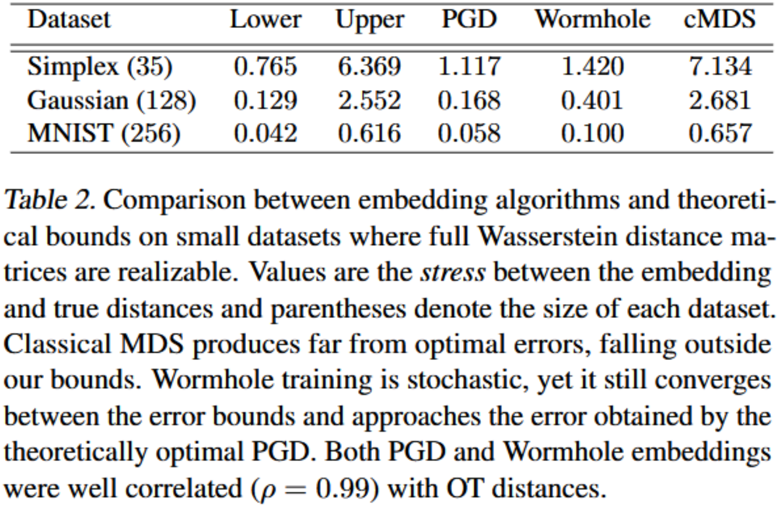
Section 6: Experimental Results

Many Datasets:

- MNIST
- Fashion-Mnist
- ModelNet40 (synthetic CAD reps of various items)
- ShapeNet
- Single-cell dataset (not covered here)
- Various small datasets where OT distances can be recovered

Section 6: Experimental Results

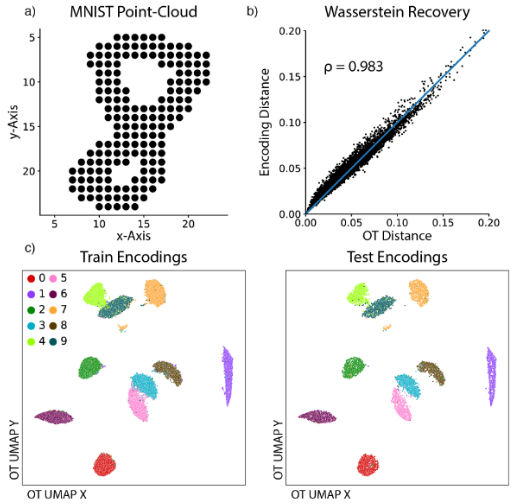
Recovery of optimal transport distances



Dataset	Lower	Upper	PGD	Wormhole	cMDS
Simplex (35)	0.765	6.369	1.117	1.420	7.134
Gaussian (128)	0.129	2.552	0.168	0.401	2.681
MNIST (256)	0.042	0.616	0.058	0.100	0.657

Table 2. Comparison between embedding algorithms and theoretical bounds on small datasets where full Wasserstein distance matrices are realizable. Values are the *stress* between the embedding and true distances and parentheses denote the size of each dataset. Classical MDS produces far from optimal errors, falling outside our bounds. Wormhole training is stochastic, yet it still converges between the error bounds and approaches the error obtained by the theoretically optimal PGD. Both PGD and Wormhole embeddings were well correlated ($\rho = 0.99$) with OT distances.

Section 6: MNIST Results



Section 6: Fashion-MNIST Results, OOD Performance

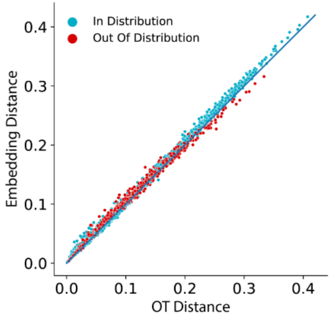


Figure 4. **Generalization to OOD on Fashion-MNIST**. During training, we held out out point clouds labeled 'Bag'. Performance was slightly lower compared to observed classes, especially for larger Wasserstein distances, as the MSE for OOD samples was $4.38 \cdot 10^{-5}$ as opposed to $3.37 \cdot 10^{-5}$ for observed samples. While not trained on any 'Bag' point cloud, their encodings still largely agree with true OT distance, producing a Pearson correlation of $\rho = 0.995$ and label accuracy of 0.97.

Section 6: Variational Wasserstein Problems (1)

Given probability measures $\mu_1, \ldots, \mu_n$, the **Wasserstein barycenter** is the measure

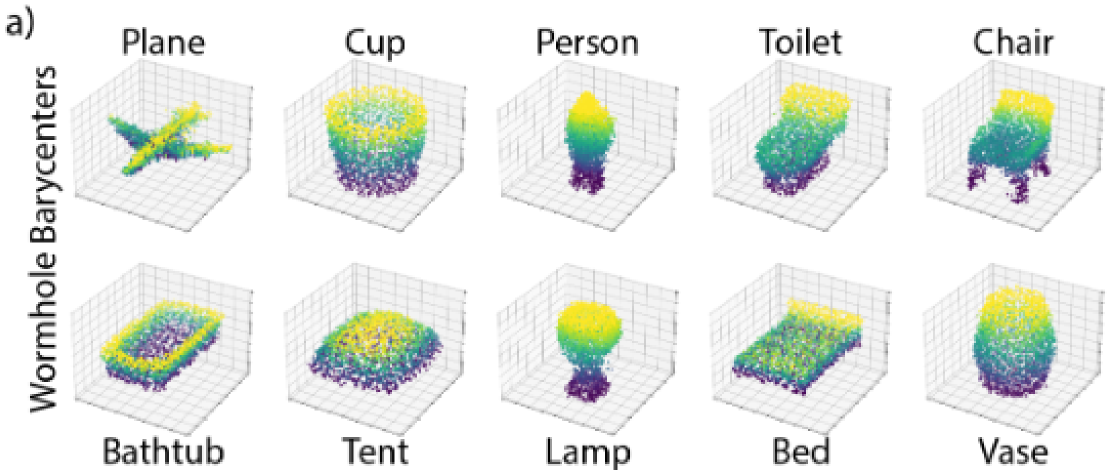
$$\nu^* = \arg\min_{\nu} \frac{1}{n} \sum_{i=1}^{n} W(\nu, \mu_i)^2.$$

Kind of like asking, "Here are a bunch of different airplanes. Can you draw me the average airplane?" Several methods for obtaining (approximating) ν^*. Here, the authors take a simple approach: Given point clouds $X_1, \ldots, X_N$, let

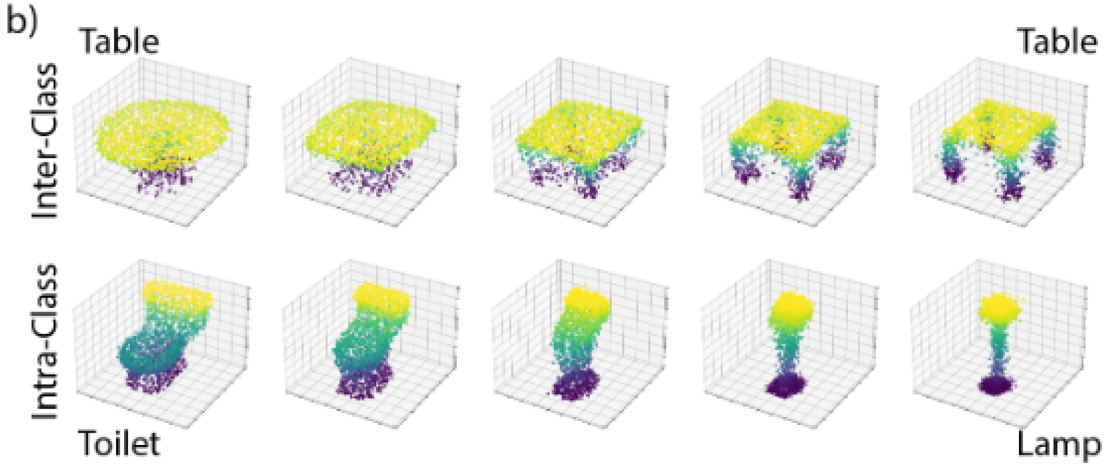
$$\bar{T}(X) = \frac{1}{n} \sum_{i=1}^{n} T(X_i).$$

The Wasserstein barycenter is then obtained by passing $\bar{T}(X)$ into their trained decoder.

Section 6: Variational Wasserstein Problems (1)



Section 6: Variational Wasserstein Problems (2)



Section 6: Generalization to Gromow-Wasserstein distances

The Gromow-Wasserstein Problem is an extension of OT: given two point clouds X, Y

$$GW(X, Y) = \min_{P \in U(\nu,\mu)} \sum_{i,j,j,j'} ||d^X_{i,i'} - d^Y_{j,j'}||^2 P_{i,j} P_{i',j'}.$$

Intuition: we want the most isometric map between X and Y. This allows for a distance metric that is invariant to rotations and reflections applied to X and Y.

Significantly more expensive to compute than even the Wasserstein distance.

The authors tested the ability of Wasserstein Wormhole to generalize to the Gromow-Wasserstein setting by randomly applying rotations to images in the ShapeNet database, and training a "Gromow-Wasserstein Wormhole" (replace regularized wasserstein distance in loss function with regularized GW distance).

Section 6: Generalization to Gromow-Wasserstein distances

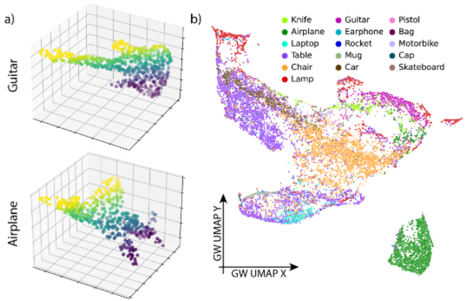
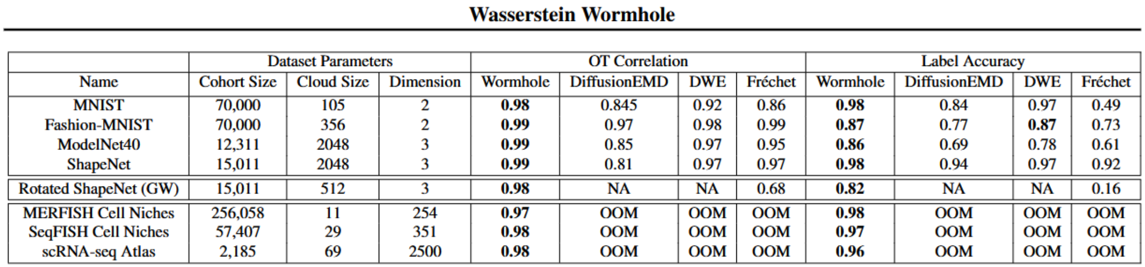


Figure 6. **Gromov-Wasserstein Wormhole. a.** Examples of point clouds from the ShapeNet dataset randomly rotated in 3D to reflect common GW applications. **b.** Embeddings learned by GW Wormhole, visualized in 2D with UMAP and colored by object class. Despite their random orientations, Wormhole was able to encode the ShapeNet point clouds and represent their class, as well as recover GW distances (Table 1).

Section 6: Summary of Results



Wasserstein Wormhole

	Dataset Parameters			OT Correlation				Label Accuracy			
Name	Cohort Size	Cloud Size	Dimension	Wormhole	DiffusionEMD	DWE	Fréchet	Wormhole	DiffusionEMD	DWE	Fréchet
MNIST	70,000	105	2	**0.98**	0.845	0.92	0.86	**0.98**	0.84	0.97	0.49
Fashion-MNIST	70,000	356	2	**0.99**	0.97	0.98	0.99	**0.87**	0.77	**0.87**	0.73
ModelNet40	12,311	2048	3	**0.99**	0.85	0.97	0.95	**0.86**	0.69	0.78	0.61
ShapeNet	15,011	2048	3	**0.99**	0.81	0.97	0.97	**0.98**	0.94	0.97	0.92
Rotated ShapeNet (GW)	15,011	512	3	**0.98**	NA	NA	0.68	**0.82**	NA	NA	0.16
MERFISH Cell Niches	256,058	11	254	**0.97**	OOM	OOM	OOM	**0.98**	OOM	OOM	OOM
SeqFISH Cell Niches	57,407	29	351	**0.98**	OOM	OOM	OOM	**0.97**	OOM	OOM	OOM
scRNA-seq Atlas	2,185	69	2500	**0.98**	OOM	OOM	OOM	**0.96**	OOM	OOM	OOM

Section 7: Recommendations

Conclusions

- Very nice paper, work seems meaningfully useful in any application where OT would be useful.

Recommendations

- Worth reading? Yes, paper is easy to read, not overly technical, and presents the authors' thinking quite nicely.
- Plenty of proofs (relatively well written from my brief review) in the supplement.
- Give good references for prior work.
- Worth implementing? Yes. Training algorithm does not seem overly complicated. Should be fairly easy to implement and train using pytorch.

References

[1] Nicolas Courty, Rémi Flamary, and Mélanie Ducoffe. "Learning wasserstein embeddings". In: *arXiv preprint arXiv:1710.07457* (2017).

[2] Alexander Y Tong et al. "Diffusion earth mover's distance and distribution embeddings". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10336–10346.

[3] Alexander Korotin, Daniil Selikhanovych, and Evgeny Burnaev. "Neural optimal transport". In: *arXiv preprint arXiv:2201.12220* (2022).

[4] Gabriel Peyré, Marco Cuturi, et al. "Computational optimal transport: With applications to data science". In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607.