Representation Learning with Multisets

Vasco Portilheiro
Department of Computer Science
Stanford University
Stanford, CA 94305
vascop@stanford.edu

Abstract

We study the problem of learning permutation invariant representations that can capture containment relations. With motivation from fuzzy set theory, we propose training a model on a novel task: predicting the size of the symmetric difference between pairs of multisets. Grounding our model in fuzzy set theory allows it to perform well on this task in comparison to baselines. Furthermore, the model learns meaningful representations, mapping objects of different classes to different standard basis vectors.

1 Introduction

Certain tasks, such as multiple-instance learning and point-cloud classification, demand representing unordered collections of inputs — in other words, *sets*. Recent work on set-based models has studied permutation invariant representations, leading to state-of-the-art performance on such tasks [3, 9, 15]. Sets have a natural partial order: the containment order. This partial order can be extended to the more general *multisets*, which may contain multiple copies of the same object. Learning to represent multisets in a way that respects this partial order is an interesting problem. For example, when comparing document keywords, we may wish to be able to predict that $\{\texttt{currency}, \texttt{equilibrium}\}$ is contained in $\{\texttt{money}, \texttt{balance}, \texttt{economics}\}$. This example makes clear an additional point: we want to learn representations of the multisets' *elements* which induce the desired multiset relations. Here in particular, we would want $\texttt{money} \approx \texttt{currency}$ and $\texttt{balance} \approx \texttt{equilibrium}$.

Learning order-respecting multiset representations is a topical problem. Learning representations that respect hierarchies or orderings is an area with exciting recent advances [2, 7, 11, 13]. However, this prior work focuses on modeling order relations between pairs of single objects. Here, we want to model relations between multisets of objects via the objects' learned representations. Furthermore, we can in fact have richer information about the relationship between two multisets than just containment — for example, the size of their intersection.

In this paper, we present a method for learning representations of multisets and their elements, given the relationships between pairs of multisets — in particular, we choose to use the sizes of their symmetric differences. We learn these representations with the goal of predicting the relationships between unseen pairs of multisets (whose elements may themselves have been unseen during training), by learning to map objects to standard basis vectors, and representing multisets as the sums of these vectors. We not only provide a theoretical basis for our method rooted in fuzzy set theory, but we also demonstrate empirically that our approach outperforms the naïve application of existing methods.

2 Related work

Set representation Qi et al. [9] and Zaheer et al. [15] both explore learning functions on sets. Importantly, they arrive at very similar theoretical statements about the approximation of such

Sets & Partitions Workshop at 33rd Conference on Neural Information Processing Systems (NeurIPS 2019).

functions, which rely on permutation invariant pooling functions. In particular, Zaheer et al. [15] show that any set function $f(A)$ can be approximated by a model of the form $\rho\left(\sum_{a\in A}\phi(a)\right)$ for some learned ρ and ϕ, which they call DeepSets. They note that the sum can be replaced by a max-pool (which is essentially the formulation of Qi et al. [9]), and observe empirically that this leads to better performance.[1] More recently, there has been some very interesting work on leveraging the relationship between sets. Probst [8] proposes a set autoencoder, while Skianis et al. [10] learn set representations with a network that compares the input set to trainable "hidden sets." However, both these approaches require solving computationally expensive matching problems at each iteration.

Orders and hierarchies Vendrov et al. [13] and Ganea et al. [2] seek to model partial orders on objects via geometric relationships between their embeddings — namely, using cones in Euclidean space and hyperbolic space, respectively. Nickel and Kiela [7] use a similar idea to embed hierarchical network structures in hyperbolic space, simply using the hyperbolic distance between embeddings. These approaches are unified under the framework of "disk embeddings" by Suzuki et al. [11]. The idea is to map each object to the product space $X \times \mathbb{R}$, where X is a metric space. This mapping can be expressed as $A \mapsto (f(A), r(A))$, and it is trained with the objective that $A \preceq B$ if and only if $d_X(f(A), f(B)) \leq r(B) - r(A)$. An equivalent statement can be made for multisets (Section 3).

Fuzzy- and multi- sets The theory of fuzzy sets can be traced back to Lotfi A. Zadeh in 1965 [14]. A fuzzy set A of objects from a universe $\mathcal{U}$ is defined via its *membership function* $\mu_A : \mathcal{U} \to [0, 1]$. Fuzzy set operations — such as intersection — are then defined in terms of this function. In modern fuzzy set theory, intersection is usually defined via a *t-norm*, which is a function $T : [0, 1]^2 \to [0, 1]$ satisfying certain properties. The intersection of two fuzzy sets A and B is defined via the membership function $\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x))$. (For more in-depth background, including the defining properties of t-norms, see Appendix A.) There is also more recent literature on extending fuzzy set theory to multisets [1, 6], using a membership function of the form $\mu_A : \mathcal{U} \times [0, 1] \to \mathbb{N}$, where $\mu_A(x, \alpha)$ is the number of appearances in A of an object x with membership α.

3 Representation of multisets

Let $\mathcal{U} = \{x_1, \ldots, x_n\}$ be the universe of possible objects. We will denote by $\mathcal{U}^*$ the set of all multisets with elements from $\mathcal{U}$. We are interested in formulating a learnable function $\Psi : \mathcal{U}^* \to \mathbb{R}^d$ mapping such multisets to vectors. Our contributions are: (1) formulating fuzzy multiset operations in a way that is amenable to machine learning, and (2) proposing to learn representations by predicting the sizes of the symmetric differences between pairs of multisets.

It is useful to begin by noting a canonical such function for sets: $\Psi(A) = [\mathbf{1}_{x_1\in A}, \ldots, \mathbf{1}_{x_n\in A}]$. Each ith coordinate of the vector is 1 if x_i is in A, 0 otherwise. This can easily be extended to multisets by letting $\Psi(A) = [m_A(x_1), \ldots, m_A(x_n)]$, where $m_A(x_i) \in \mathbb{N}$ is the multiplicity of x in A. This function m_A should immediately remind us of fuzzy sets. In fact, if we simply let m_A map to any non-negative real number — that is, $m_A(x_i) \in \mathbb{R}_+$ — we obtain a representation of fuzzy multisets. Note that this is not the same formulation of "fuzzy multisets" usually given in literature [1, 6]. However, this formulation is much more easily amenable to the machine-learning setting. Intuitively, the benefit of "fuzzifying" our multisets is the ability to optimize via backpropagation.

With this definition of fuzzy multisets, we now define fuzzy multiset operations. Furthermore, we require that these operations be compatible both with their equivalents for non-fuzzy multisets and plain old sets (i.e. when $m_A(x_i) \in \mathbb{N}$ and $m_A(x_i) \in \{0, 1\}$, respectively). This will in fact force us to adopt the following definitions (see Appendix B):

- Size: $|A| = \sum_{i=1}^{n} m_A(x_i)$
- Intersection: $m_{A\cap B}(x_i) = \min\{m_A(x_i), m_B(x_i)\}$[2]

[1] We believe there is an interesting theoretical distinction worth noting here, which may help explain this observation. Namely, max-pooling is *idempotent*, meaning that repeatedly pooling a representation with itself does not change the result. On the other hand, summation does not have this property, and so repeated copies of an element are reflected in the result. In this way, DeepSets (with the sum rather than max-pool) is in fact modeling *multisets* rather than sets, which depending on the application may be undesirable.

[2] Note that $\min$ is in fact a t-norm used for defining fuzzy set intersection since Zadeh [14]. However, as noted in Appendix B, it is the only t-norm we can use that is compatible with multiset intersection.

- Union: $m_{A\cup B}(x_i) = \max\{m_A(x_i), m_B(x_i)\}$
- Multiset addition: $m_{A+B}(x_i) = m_A(x_i) + m_B(x_i)$
- Multiset difference: $m_{A\setminus B}(x_i) = \max\{m_A(x_i) - m_B(x_i), 0\}$
- Symmetric difference: $m_{A\triangle B}(x_i) = |m_A(x_i) - m_B(x_i)|$

We can now justify our choice of the size of the symmetric difference to capture the relationship between two multisets. First, note that given $|A|$ and $|B|$, and any one of $|A \cap B|$, $|A \setminus B|$, or $|A \triangle B|$, we can tell what fraction of elements of A are in B and vice-versa. In particular, we can also tell whether $A \subseteq B$ and whether $B \subseteq A$. In fact, we can formulate this relationship between containment and symmetric difference as a disk embedding inequality, where the metric space is that induced on $\mathcal{U}^*$ by the counting measure: $A \subseteq B$ if and only if $|A \triangle B| \leq |B| - |A|$ (see Appendix C for proof). This is by itself an appealing reason to use symmetric difference. In the context of backpropagation, however, our expression for fuzzy symmetric difference also has the advantage of having derivatives depending both on $m_A(x_i)$ and $m_B(x_i)$ for each i, unless $m_A(x_i) = m_B(x_i)$.

We are now ready to formulate our learning model. To do so, we must formulate Ψ and our loss function. Our loss function will penalize error in predicting $|A \triangle B|$ for input multisets A and B. We will use $\Delta(A, B)$ to denote our predicted symmetric difference size. Given a training distribution D of pairs of multisets, our loss is $\mathcal{L} = \mathbb{E}_{A,B\sim D}\left[(\Delta(A, B) - |A \triangle B|)^2\right]$.

We formulate Ψ with the DeepSets expression in mind: $\Psi(A) = \rho\left(\sum_{a\in A} \phi(a)\right)$, where $\phi : \mathcal{U} \to \mathbb{R}^d$ is some given object-featurization function. To make a perfect analogy to the fuzzy multiset representation above, we would require that each $\phi(a)$ be a standard basis vector. We want to be a little more general however. For example, we may want to allow $\phi(a) \in \mathbb{R}^d$ with $d < n$. More importantly, we need to be able to learn ϕ. We do this by relaxing from standard basis vectors, allowing $\phi(a)$ to be any non-negative vector such that $||\phi(a)||_1 = 1$. That is, $\phi(a)$ can be a weighted combination of each of the standard bases with non-negative weights summing to 1 (which for $d = n$ can be interpreted as a weighted combination each of the elements of $\mathcal{U}$). The normalization guarantees that $\phi(a)$ has the same scale as any of the basis elements. Importantly, if the norm $||\cdot||_1$ is invariant under ρ, then we are guaranteed that $||\Psi(A)||_1 = |A|$. In practice, we guarantee the restrictions above by replacing $\phi(a)$ with $\frac{f(\phi(a))}{||f(\phi(a))||_1}$, where $f : \mathbb{R} \to \mathbb{R}_+$ is a function applied element-wise. For differentiability, we choose the softplus function $f(x) = \log(1 + e^x)$. Letting ρ be the identity, we thus obtain: $\Psi(A) = \sum_{a\in A} \frac{f(\phi(a))}{||f(\phi(a))||_1}$, and $\Delta(A, B) = ||\Psi(A) - \Psi(B)||_1$.

4 Experiments

We compare our model above with two reasonable alternatives. For the first, we simply relax the restrictions on object representations — which by abuse of notation we will call the restriction of ϕ (to the standard simplex) — letting $\Psi(A) = \sum_{a\in A} \phi(a)$, and Δ as before. The second alternative is to simply replace both Ψ and Δ (which should itself permutation invariant) with DeepSets functions, letting $\Psi(A) = \rho_1\left(\sum_{a\in A} \phi(a)\right)$ and $\Delta(A, B) = \rho_2\left(\Psi(A) + \Psi(B)\right)$. We compare these models both on the task of predicting symmetric difference size, and in terms of their learned representations.

We use MNIST [5] as our dataset. For training, we sample a fixed number of pairs of multisets from the training set, whose sizes are uniformly distributed, either on $[2, 5]$ or $[2, 10]$.[3] The symmetric difference to be predicted is calculated directly from the image labels. Evaluation is performed similarly, with the addition of multiset sizes uniform on $[2, 20]$, and with images sampled from the test set. Importantly, this means that the none of the objects which are members of the multisets seen during training appear during evaluation. Further details on the training and evaluation procedures (including network architecture) are included in Appendix D.

Symmetric difference size prediction Our results on MNIST are presented in Table 1. We observe that including the restriction on ϕ greatly improves performance when the symmetric difference is predicted as $\Delta(A, B) = ||\Psi(A) - \Psi(B)||_1$. Furthermore, in the restricted case, training on larger multisets leads to slightly better performance. It is unclear, however, whether training on larger multisets helps the multiset models "generalize" to multisets with sizes larger than those seen in

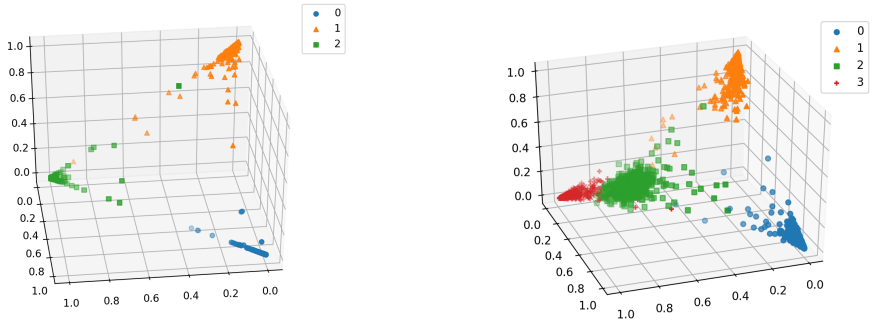
[3] We exclude singleton sets to ensure that the models aren't just learning from comparing pairs of singletons.

Table 1: Mean absolute errors in symmetric difference size prediction on MNIST

	sizes $\in [2, 5]$	sizes $\in [2, 10]$	sizes $\in [2, 20]$
Multisets (restr. ϕ), training sizes $\in [2, 5]$	0.0722	0.1264	0.2061
Multisets (restr. ϕ), training sizes $\in [2, 10]$	0.0595	0.1059	0.1756
Multisets (unrestr. ϕ), training sizes $\in [2, 5]$	0.5614	0.7693	1.1813
Multisets (unrestr. ϕ), training sizes $\in [2, 10]$	0.6349	0.7740	0.9859
DeepSets, training sizes $\in [2, 5]$	1.2152	1.9386	5.2831
DeepSets, training sizes $\in [2, 10]$	1.2227	1.5358	3.0340

training. Finally, we see that the using the fuzzy symmetric multiset difference was also important to learning, as the pure DeepSets model struggles to predict the correct value. (We note that rounding the models' predictions to whole numbers only slightly improved performance, if at all.)

Examining learned representations With $\phi(a) \in \mathbb{R}^n$ restricted to the standard $(n-1)$-simplex, we find empirically that learned representations of objects are approximately the standard basis vectors (as shown in Figure 1a for $n = 3$). This makes sense intuitively, since this is exactly the fuzzy multiset representation on which we base our symmetric difference and size operations. This is an interesting property; we note that, given the mapping from coordinate to class, 98.9% classification accuracy is achieved just by picking the maximum-valued coordinate of the representation of each object. We also examine the case $d < n$, when the dimension of our representations is smaller than the number of possible objects. Here, the "pinched" nature of the restricted representations may be undesirable (Figure 1b). This problem, of course, gets worse with the discrepancy between number of objects and dimension (Figure 2). On the other hand, the unrestricted multiset model is able to learn more balanced-looking clusters. However, the clusters for $d = n$ appear slightly less well-separated (Figures 3 and 4). The DeepSets model didn't learn interpretable representations (Figure 5).



(a) Model trained on zeros, ones and twos.

(b) Model trained on zeros, ones, twos, and threes.

Figure 1: Three-dimensinoal representations of test-set MNIST images generated by the restricted multiset model trained on multisets of sizes $\in [2, 5]$.

5 Conclusion

We have developed a model that given only the sizes of symmetric differences between pairs of multisets, learns representations of such multisets and their elements. Our model learns to map each type of object to an orthogonal basis vector, thus essentially performing semi-supervised clustering. One interesting area for future theoretical work is understanding a related problem: clustering n objects given multiset difference sizes. As a first step, we show in Appendix F that $n - 1$ specific multiset comparisons are sufficient to recover the clusters. We would also be curious to see if our model can learn meaningful representations even if the given labels are not exactly the sizes of symmetric differences — for example, human judgements of how different two bags-of-words are.

References

[1] Jaume Casasnovas and Gaspar Mayor. Discrete t-norms and operations on extended multisets. *Fuzzy Sets and Systems*, 159:1165–1177, 2008.

[2] Octavian Ganea, Gary Becigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[3] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[5] Yann LeCun. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

[6] Sadaaki Miyamoto. Fuzzy multisets and their generalizations. In *Proceedings of the Workshop on Multiset Processing: Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View, Workshop on Membrane Computing (WMP)*, 2000.

[7] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

[8] Malte Probst. The set autoencoder: Unsupervised representation learning for sets. *OpenReview*, 2018.

[9] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. Rep the set: Neural networks for learning set representations. *ArXiv*, 2019.

[11] Ryota Suzuki, Ryusuke Takahama, and Shun Onoda. Hyperbolic disk embeddings for directed acyclic graphs. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[12] Laurens van der Maaten and Geoffrey Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 2008.

[13] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *International Conference on Learning Representations (ICLR)*, 2015.

[14] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

[15] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.

A T-norms and Fuzzy Sets

Here, we provide a brief look into fuzzy set theory.

A *t-norm* is a function $T : [0, 1]^2 \to [0, 1]$, satisfying the following properties:

- Commutativity: $T(a, b) = T(b, a)$
- Monotonicity: If $a \leq c$ and $b \leq d$, then $T(a, b) \leq T(c, d)$
- Associativity: $T(a, T(b, c)) = T(T(a, b), c)$
- 1 is the identity: $T(a, 1) = a$

T-norms generalize the notion of conjunction. Note that the above conditions imply that for any a, $T(0, a) = 0$. Additionally, $T(1, 1) = 1$. These two observations show that t-norms are "compatible" with classical, non-fuzzy logic — where we identify 0 with "false" and 1 with "true." The standard t-norm is $T(a, b) = \min\{a, b\}$.

A *strong negator* — the generalization of negation — is a strictly monotonic, decreasing function $n : [0, 1] \to [0, 1]$ such that $n(0) = 1$, $n(1) = 0$ and $n(n(x)) = x$. The standard strong negator is $n(x) = 1 - x$.

S-norms (also called *t-conorms*) are functions with the same properties as t-norms, except that the identity element is 0. They generalize disjunction. For every t-norm (and a given negator), we can define a *complementary s-norm*: $S(a, b) = n(T(n(a), n(b)))$. This is a generalization of De Morgan's laws. The standard s-norm, complementary to the $\min$ t-norm, is $S(a, b) = \max\{a, b\}$.

Recall that a fuzzy set A taking elements from a universe $\mathcal{U}$ is defined through its *membership function* $\mu_A : \mathcal{U} \to [0, 1]$. Intuitively, μ_A maps each $x \in \mathcal{U}$ to "how much of a member" x is of A, on a scale from 0 to 1. It is therefore natural to define the membership function for the intersection of two fuzzy sets A and B as $\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x))$ for a t-norm T. Similarly, the complement of a fuzzy set is given by $\mu_{\overline{A}}(x) = n(\mu_A(x))$ for a strong negator n, and the union of two fuzzy sets is given by $\mu_{A\cup B}(x) = S(\mu_A(x), \mu_B(x))$ for an s-norm S. Usually, we want T and S to be complementary with respect to n. Then, we can define all the usual set operations by combining the three basic operations above.

B Fuzzy Multiset Operations

Here, we will provide some intuition for the fuzzy multiset operations we list in Section 3, and show that they are the only reasonable choices we can make. Let us start with non-fuzzy multisets. That is, suppose we are dealing with multisets with membership functions mapping elements of $\mathcal{U}$ to the natural numbers. Let us take two multisets, $A = \{1, 1, 1, 2, 2\}$ and $B = \{1, 1, 2, 3\}$. Their intersection should contain all their elements in common: $A \cap B = \{1, 1, 2\}$. That is, we take the minimum count of each element appearing in either A or B, and that is the number of time the element appears in $A \cap B$. This is straightforward, and gives us $m_{A\cap B}(x) = \min\{m_A(x), m_B(x)\}$. This is interesting, as $\min$ is a t-norm, used to define fuzzy set intersection. However, it is clear that we in fact *must* use the $\min$ t-norm, in order to be compatible with our notion of multiset intersection. (Note that for *fuzzy* multisets, nothing is stopping us from using a function that is the same as $\min$ only on integer inputs, although this would introduce unnecessary complexity with no clear gain.)

Following similar reasoning, we can convince ourselves that multiset union must be defined as $m_{A\cup B}(x) = \max\{m_A(x), m_B(x)\}$. It is important to differentiate this from "multiset addition," which simply combines two multisets directly: $A + B = \{1, 1, 1, 1, 1, 2, 2, 2, 3\}$ for our example above, and in general $m_{A+B} = m_A(x) + m_B(x)$.

Multiset difference is a little bit trickier. One interesting problem is that we cannot rely on a notion of "complement" for multisets. Instead, let us again try to reason by example. For our example multisets above, we would expect the result to be $A \setminus B = \{1, 2\}$. What did we do here? We removed from A copies of elements which appeared in B, ignoring any elements which did not appear in A. We can also imagine that if B had more of a certain element than A, that element would not appear in the final result. In other words, we are performing a subtraction of counts which is "glued" to a minimum value of zero. So, $m_{A\setminus B}(x) = \max\{m_A(x) - m_B(x), 0\}$. We can further convince ourselves of the correctness of this expression by noting that we recover the identity $A \setminus (A \setminus B) = A \cap B$ by plugging in the functions above.

Finally, symmetric multiset difference can be defined using our expression for multiset difference above, combined with either multiset addition or union. In particular, note that $A \triangle B = (A \setminus B) + (B \setminus A) = (A \setminus B) \cup (B \setminus A)$ — addition and union both work because $(A \setminus B)$ and $(B \setminus A)$ are necessarily disjoint. This gives us:

$$m_{A\triangle B}(x) = \max\{m_A(x) - m_B(x), 0\} + \max\{m_B(x) - m_A(x), 0\} = |m_A(x) - m_B(x)|.$$

(And again, note that the equation still holds if we replace the addition with a maximum.)

C A Disk Embedding Inequality for (Fuzzy) Multisets

We will prove that for any two (possibly fuzzy) multisets A and B, $A \subseteq B$ if an only if $|A \triangle B| \leq |B| - |A|$. (It is worth clarifying that in the context of multisets, $A \subseteq B$ is equivalent to the statement that for any $x \in \mathcal{U}$, $m_A(x) \leq m_B(x)$.) In fact, we will show that the inequality can be replaced with an equality!

Suppose that $A \subseteq B$. Then, for all $x \in \mathcal{U}$, $m_B(x) \geq m_A(x)$, and the desired result follows:

$$|A \triangle B| = \sum_{x \in \mathcal{U}} m_{A \triangle B}(x) = \sum_{x \in \mathcal{U}} |m_B(x) - m_A(x)| = \sum_{x \in \mathcal{U}} m_B(x) - m_A(x) = |B| - |A|.$$

Suppose on the other hand that $|A \triangle B| \leq |B| - |A|$. We first note that this in fact means that $|A \triangle B| = |B| - |A|$, since we are given

$$\sum_{x \in \mathcal{U}} |m_B(x) - m_A(x)| \leq \sum_{x \in \mathcal{U}} m_B(x) - m_A(x),$$

but it must be that

$$\sum_{x \in \mathcal{U}} |m_B(x) - m_A(x)| \geq \sum_{x \in \mathcal{U}} m_B(x) - m_A(x).$$

Now, suppose for the sake of contradiction that for some $x^* \in \mathcal{U}$, it is the case that $m_A(x^*) > m_B(x^*)$. Then, $m_B(x^*) - m_A(x^*) < 0 \leq |m_B(x^*) - m_A(x^*)|$. But this implies that

$$\sum_{x \in \mathcal{U}} |m_B(x) - m_A(x)| > \sum_{x \in \mathcal{U}} m_B(x) - m_A(x),$$

which is a contradiction.

D Training and Evaluation Procedures

We train all the models on 3×10^5 training pairs of multisets. Both of the multisets in each pair are generated randomly at each iteration, as follows. First a size is uniformly sampled in the chosen range (for us, either $[2, 5]$ or $[2, 10]$). That many images are then chosen uniformly at random (with replacement) from the training set. All models are optimized using Adam [4] with the default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a learning rate of 5×10^{-5}. The learning rate was chosen by logarithmic grid search from 1 down to 5×10^{-6}, training on up to 10^4 pairs during the search. (All models performed best with the chosen learning rate — or at least no worse than any of the other learning rates.) Given this learning rate, we chose to train the models for 3×10^5 iterations, finding that almost all of the models converged by this point. The multiset model with restricted ϕ appeared as though further training could still slightly improve performance, but this model already outperformed the others.

Evaluation is performed similarly to training, expect that the pairs of multisets are generated using only images from the test datasets. For symmetric difference size prediction, each model is evaluated on 3×10^4 such multiset pairs.

For the object featurizing function ϕ, we use a variant of the LeNet-5 neural network [5]. Specifically, we adopt the same architecture as used by Ilse et al. [3]. Given input images with c channels, and an output dimension d, the network consists of:

1. A two-dimensional convolution layer with c input channels, 20 output channels, kernel size 5, and stride 1 (no padding)
2. A ReLU activation
3. A two-dimensional max-pooling layer with kernel size 2 and stride 2
4. A two-dimensional convolution layer with 20 input channels, 50 output channels, kernel size 5, and stride 1 (no padding)
5. A ReLU activation
6. A two-dimensional max-pooling layer with kernel size 2 and stride 2

7. A fully-connected linear layer with output size d (the input size is determined by c)

For the DeepSets model, we used for ρ_1 the architecture:

1. A fully-connected linear layer with input size d and output size 100
2. A hyperbolic tangent activation
3. A fully-connected linear layer with input and output size 100

For ρ_2 we used:

1. A fully-connected linear layer with input and output size 100
2. A hyperbolic tangent activation
3. A fully-connected linear layer with input size 100 and output size 1

E Additional Figures

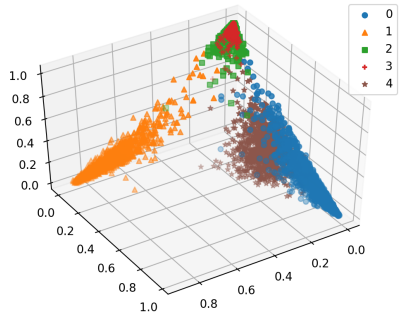
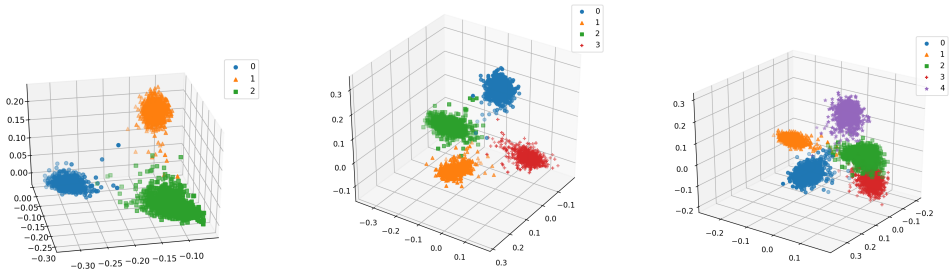
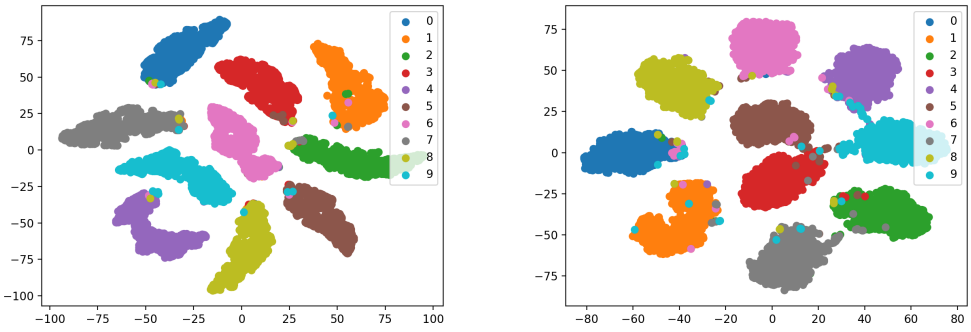


Figure 2: Three-dimensinoal representations of test-set MNIST images generated by the restricted multiset model trained on multisets of sizes $\in [2, 5]$; the model is trained on images of zeros, ones, twos, threes, and fours. The representations of twos and threes are essentially inseparable.



(a) (b) (c)

Figure 3: Three-dimensinoal representations of test-set MNIST images generated by the unrestricted multiset model trained on multisets of sizes $\in [2, 5]$ in (a), the model is trained on images of zeros, ones and twos; in (b), the model is trained on zeros, ones, twos, and threes; in (c), the model is trained on zeros, ones, twos, threes, and fours. Note that in (c), the clusters essentially form a tetrahedron, with one of the vertices being the combination of twos and threes.



(a) (b)

Figure 4: Two dimensional TSNE [12] "projections" of the ten-dimensional representations of test-set MNIST images generated by the multiset models; the models were trained on multisets of sizes $\in [2, 5]$; in (a), the model has object representations ϕ restricted to the standard simplex; in (b), ϕ is unrestricted.

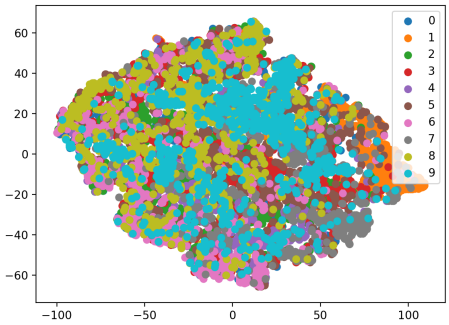


Figure 5: Two dimensional TSNE [12] "projections" of the ten-dimensional representations of test-set MNIST images generated by the plain DeepSets model trained on multisets of sizes $\in [2, 5]$.

F Clustering n Objects Given $n-1$ Symmetric Set Difference Sizes

We are interested in the following problem. Suppose we have a set of n objects $\mathcal{U}$, each of which belongs to one of k clusters, $C_1, \ldots, C_k$. Let $M : 2^{\mathcal{U}} \to \{1, \ldots, k\}^*$ be the function which takes any subset of $\mathcal{U}$, and gives the multiset of cluster labels represented in that subset. We are given oracle access to the function $\Delta : 2^{\mathcal{U}} \times 2^{\mathcal{U}} \to \mathbb{N}$ which gives the size of the symmetric set difference between the cluster-label multisets: $\Delta(A, B) = |M(A) \triangle M(B)|$. How many queries are required to determine the clusters $C_1, \ldots, C_k$ (up to permutation)?

We show that the clusters can be determined with $n-1$ specific queries. (Another way to think of this is as a training data problem, rather than an oracle querying problem; we show $n-1$ training examples can be sufficient.) We do this in two steps. The step lets us identify k disjoint subsets of $\mathcal{U}$, such that no two of these subsets contain objects from the same cluster. The second step confirms that these subsets are in fact the clusters $C_1, \ldots, C_k$.

The first step consists of logarithmically "splitting" $\mathcal{U}$. The very first query in this step is $\Delta\left(\bigcup_{i=1}^{\lceil k/2 \rceil} C_i, \bigcup_{i=\lceil k/2 \rceil}^{k} C_i\right)$, which tells us that $\bigcup_{i=1}^{\lceil k/2 \rceil} C_i$ and $\bigcup_{i=\lceil k/2 \rceil}^{k} C_i$ are disjoint in terms of represented clusters. We proceed recursively, each query "splitting" the sets in half (in terms of which clusters they contain). The number of such steps required is $k-1$ (which is the number of internal nodes in a balanced binary search tree for k objects). We'll call the resulting disjoint sets $\tilde{C}_1, \ldots, \tilde{C}_k$ (since we technically don't yet know they correspond to the true clusters).

For the second step, we must verify that the objects in each of our sets resulting from step one all belong to the same cluster. This can be done by ordering the objects within each set, and comparing each consecutive pair as singletons. For each of our sets $\tilde{C}_i$, we thus make $|\tilde{C}_i| - 1$ such queries. Across all such sets, we thus make $\sum_{i=1}^{k} |\tilde{C}_i| - 1 = n - k$ queries.

So, the total number of queries made is $(k-1) + (n-k) = n - 1$.