Although recent work in non-linear dimensionality reduction investigates multiple choices of divergence measure during optimization (Yang et al., 2013; Bunte et al., 2012), little work discusses the direct effects that divergence measures have on visualization. We study this relationship, theoretically and through an empirical analysis over 10 datasets. Our work shows how the $\alpha$ and $\beta$ parameters of the generalized alpha-beta divergence can be chosen to discover hidden macro-structures (categories, e.g. birds) or micro-structures (fine-grained classes, e.g. toucans). Our method, which generalizes t-SNE (van der Maaten, 2008), allows us to discover such structure without extensive grid searches over $(\alpha, \beta)$ due to our theoretical analysis: such structure is apparent with particular choices of $(\alpha, \beta)$ that generalize across datasets. We also discuss efficient parallel CPU and GPU schemes which are non-trivial due to the tree-structures employed in optimization and the large datasets that do not fully fit into GPU memory. Our method runs 20x faster than the fastest published code (Vladymyrov & Carreira-Perpinán, 2014). We conclude with detailed case studies on the following very large datasets: ILSVRC 2012, a standard computer vision dataset with 1.2M images; SUSY, a particle physics dataset with 5M instances; and HIGGS, another particle physics dataset with 11M instances. This represents the largest published visualization attained by SNE methods. We have open-sourced our visualization code: http://rll.berkeley.edu/absne/.

1. Introduction and Related Work

Data visualization techniques aim to generate a low-dimensional representation of a dataset which data scientists and researchers can inspect to gain insight into the structure and complexity of the data. Vital to data-driven decision making, visualizations graphically represent latent structure and meaning in the dataset. Many recent approaches produce low-dimensional embeddings of data

---

**Figure 1.** Discovering micro-structures using the settings $\alpha < 1, \lambda = 1$, prescribed by the intuition in (Section 4). Using Caffe fc7 features (Jia et al., 2014), we discover structures in the ILSVRC 2012 test set, e.g., classes of birds, and visual transitions, e.g., perched birds transitioning smoothly to swinging monkeys.
instances, which we can view via scatter plots (Carreira-Perpín, 2010; Hinton & Roweis, 2003; van der Maaten, 2008; Lawrence, 2011; Bunte et al., 2012; Vladymyrov & Carreira-Perpíná, 2014; Yang et al., 2014). Particularly, recent variants on stochastic neighborhood embedding (SNE) have become popular (Hinton & Roweis, 2003); the recently published t-SNE (t-distributed stochastic neighbor embedding) is particularly popular (van der Maaten, 2008). At a high level, SNEs (i) capture neighborhood information from pairs of points in the original dataset with \( m \) data instances into an \( m \times m \) data similarity matrix, and (ii) learn a low-dimensional embedding of those points whose similarity matrix closely matches the original. Computing an embedding which matches the structure of the original data involves minimizing a divergence measure, e.g., KL-divergence, between the data and embedding similarity matrices. Indeed, most methods employ the KL-divergence (Hinton & Roweis, 2003; van der Maaten, 2008). Past work primarily explores minimizing purely a single divergence in the SNE framework. We discover that minimizing the generalized alpha-beta divergence (a.k.a. AB-divergence) (Cichocki et al., 2011), which blends the \( \alpha \)- and \( \beta \)-divergences, is crucial to discovering important micro and macro structures in the data (Section 4).

2. Background: t-distributed Stochastic Neighborhood Embedding

Given a dataset \( D = \{x_1, x_2, \ldots, x_n\} \), with data instances \( x_i \in \mathbb{R}^d \), t-distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten, 2008) aims to learn an embedding \( \mathcal{E} = \{y_1, y_2, \ldots, y_m\} \), where \( y_i \in \mathbb{R}^d \) (usually, \( d = 2 \) or 3). To achieve this goal, t-SNE defines

\[
P_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_k exp(-||x_i - x_k||^2/2\sigma_i^2)}, \quad (1)
\]

\[
P_{ij} = (P_{ij} + P_{ji})/2m, \quad (2)
\]

\[
Q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k\neq l}(1 + ||y_k - y_l||^2)^{-1}}, \quad (3)
\]

where additionally, \( P_{ii} = Q_{ii} = 0 \). Determining the individual variances \( \sigma_i^2 \) involves running a binary search such that the perplexity (2 raised to the entropy) of the conditional \( P_{ij} \) equals \( k \), a free parameter (van der Maaten, 2013). The embedding employs a Student-t kernel rather than a Gaussian to prevent embedding points from crowding near the center of the visualization map without clear clustering, a.k.a. the “crowding problem.” t-SNE prescribes learning the embedding vectors \( y_i \) by running gradient descent to minimize the resulting non-convex KL-divergence, \( J(\mathcal{E}) = \sum_{i \neq j} P_{ij} \log P_{ij}/Q_{ij} \):

\[
\frac{\partial J}{\partial y_i} = 4 \sum_{i \neq j} Z(y_i - y_j)(P_{ij}Q_{ij} - Q_{ij}^2) \quad (4)
\]

\[
= \sum_{i \neq j} 4P_{ij}Q_{ij}Z(y_i - y_j) - \sum_{i \neq j} 4Q_{ij}^2Z(y_i - y_j) \quad \text{Force 1}
\]

\[
= \sum_{i \neq j} 4P_{ij}Q_{ij}Z(y_i - y_j) - \sum_{i \neq j} 4Q_{ij}^2Z(y_i - y_j) \quad \text{Force 2}
\]

where \( Z = \sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1} \). Naively computing this gradient takes \( O(m^2) \) time, making visualizations of
greater than 50K data vectors prohibitively expensive.

Barnes-Hut-SNE (BHSNE) (van der Maaten, 2013) approximates the t-SNE’s gradient in sub-quadratic time, yielding nearly identical visualizations to t-SNE while allowing for visualizations of millions of data vectors in a few hours. To this end, BHSNE only retains \( P_{ij} \) where data vector \( x_j \) is one of \( x_i \)’s closest \( 3k \) neighbors, and sets the rest to 0. In practice, BHSNE constructs a vantage point tree to execute all nearest neighbor searches in \( O(kmn \log m) \) time (van der Maaten, 2013; Yianilos, 1993). BHSNE computes Force 1 in \( O(km) \) time by adding only terms involving positive \( P_{ij} \) and computing each \( Q_{ij} \).

\[
Barnes-Hut-SNE \text{ computes Force 1 in } O(km) \text{ time by adding only terms involving positive } P_{ij} \text{ and computing each } Q_{ij}.
\]

\[
\ln \alpha(j) \text{ in computing Force 1 for a point } y_j \text{ will be roughly equal. The Barnes-Hut algorithm exploits this in computing the sum of all contributions to an embedding vector } y_j \text{ by (i) constructing a quadtree over } \{y_i\}, \text{ (ii) traversing the quadtree via a depth-first-search, and (3) at every quadtree node, deciding whether the corresponding cell can summarize the gradient contributions for all points in that cell. In computing Force 2 for a point } y_j, \text{ if a cell is sufficiently small and far away from } y_j, \text{ then } Q_{ij}^2 \text{ will be similar for all points } y_j \text{ in that cell. As such, BHSNE approximates the total contribution as } N_{cell} \cdot Q_{ij}^2 (y_j - y_j) \text{ where } N_{cell} \text{ denotes the total number of points in the cell.}\]

\[
\text{To compute } Z \text{ efficiently, BHSNE runs a separate Barnes-Hut procedure to compute a } z_i = \sum_j K_q(\|y_i - y_j\|^2) \text{ for each embedding point } i \text{ and (ii) sums over the } z_i \text{'s to yield } Z. \text{ BHSNE then uses this value of } Z \text{ in the Barnes-Hut procedure to compute Force 2. BHSNE decides whether a cell can summarize the points that it contains by checking whether } ||y_i - y_{cell}||^2/r_{cell} < \theta, \text{ where } r_{cell} \text{ is the length of the cell diagonal, } y_{cell} \text{ is the cell’s center of mass, and } \theta \text{ is a threshold that trades off speed and accuracy (larger values lead to poorer approximations).}
\]

3. Alpha-Beta Stochastic Neighborhood Embedding

Alpha-Beta Stochastic Neighborhood Embedding (AB-SNE), our proposed method, differs from t-SNE in that it minimizes the alpha-beta divergence (AB-divergence). We minimize the cost \( J_{\text{AB-SNE}}[E; \alpha, \beta] = D_{\text{AB}}^{\alpha\beta}(P||Q), \) computed as

\[
\frac{1}{\alpha \beta} \sum_{i \neq j} \left( -P_{ij}^\alpha Q_{ij}^\beta + \frac{\alpha}{\alpha + \beta} P_{ij}^{\alpha + \beta} + \frac{\beta}{\alpha + \beta} Q_{ij}^{\alpha + \beta} \right),
\]

\[(6)\]

\[
\text{where } \lambda = \alpha + \beta. \text{ We inspect the updates taken during learning:}
\]

\[
\Delta y_i = -\frac{\partial D_{\text{AB}}^{\alpha\beta}(P||Q)}{\partial y_i} = \sum_j \frac{\partial Q_{ij}}{\partial y_i} Q_{ij}^{\lambda - 1} \ln_{1 - \alpha}(P_{ij}/Q_{ij}) \]

\[(7)\]

\[
= \sum_j \frac{\partial Q_{ij}}{\partial y_i} Q_{ij}^{\lambda - 1} \ln_{1 - \alpha}(r_{ij}) \]

\[(8)\]

\[
\text{where } \ln_q(x) \text{ is the Tsallis deformed } q\text{-logarithm, and } r_{ij} = P_{ij}/Q_{ij}. \text{ Our theoretical analysis considers how}
\]

\[\ln_q(x) \equiv x^{1-q} - 1/(1-q) \text{ if } q \neq 1, \text{ else } \ln_q(x) = \ln x.\]
each force \( f_{ij} = \frac{\partial Q_{ij}}{\partial y_j} \cdot Q_{ij}^{\lambda-1} \ln_{1-\alpha}(r_{ij}) \) affects \( \Delta y_i \). The term \( \frac{\partial Q_{ij}}{\partial y_j} \) is a vector parallel to the ray \( k(y_i - y_j) \). Since the t-distribution monotonically decreases for positive arguments, we must have \( k < 0 \), i.e., \( f_{ij} \) points towards \( y_j \), implying that \( f_{ij} \) attracts \( y_i \) towards \( y_j \) if \( Q_{ij}^{\lambda-1} \ln_{1-\alpha}(r_{ij}) > 0 \Rightarrow r_{ij} > 1 \) (see Figure 2) and repulses \( y_i \) from \( y_j \) if \( r_{ij} < 1 \). Specifically, non-neighboring point pairs in the original dataset \( D \) with \( P_{ij} = 0 \Rightarrow r_{ij} = 0 < 1 \) repel each other.

We interleave theoretical intuition with empirical verification on two datasets (more results presented in Section 6): MNIST (LeCun & Cortes, 1998), a dataset of 70K 28 × 28 grayscale images depicting handwritten digits 0-9 and CIFAR-10 (Krizhevsky & Hinton, 2009), a dataset of 32 × 32 color images depicting 10 distinct object classes. We use the raw pixel data as MNIST’s feature representation. For CIFAR-10, we train a 3-layer convolutional neural network with the CUDACONVNET (Krizhevsky, 2011) architecture using Caffe (Jia et al., 2014) and employ only third layer convolutional features; we visualize the test set to avoid having training labels directly influence the embedding. Applying PCA to center and reduce each dataset to 100 dimensions, we ran ABSNE under various \((\alpha, \lambda)\) for both datasets with perplexity 30 (Figure 3). We initialize all \( y_i \) in experiments with the same random seed and run exactly 1000 iterations of gradient descent per configuration (see Section 5 for optimization details), so structural details across a row should be comparable; the first column \((\alpha = 1.0, \lambda = 1.0)\) denotes vanilla t-SNE.

**Intuition Behind \( \alpha \).** To study \( \alpha \), let us fix \( \lambda = 1 \). Consider a cluster of embedding points consisting of a few sub-clusters. After convergence, the sub-clusters will be placed together in such a way that the attractive and repulsive forces are balanced. According to Figure 2a., decreasing \( \alpha \) below 1 emphasizes the magnitude of (repulsive) forces with \( r_{ij} < 1 \) relative to (attractive) forces with \( r_{ij} > 1 \). The emphasized repulsive forces and diminished attractive forces should cause sub-clusters to be placed further apart, implying that ABSNE should tend to produce lots of small, fine-grained clusters for \( \alpha < 1 \).

**Intuition Behind \( \lambda \).** To study \( \lambda \), let us fix \( \alpha = 1 \). According to Figure 2b., setting \( \lambda < 1 \) emphasizes \( f_{ij} \) with low \( Q_{ij} \) (distant \( y_i, y_j \)), over \( f_{ij} \) with high \( Q_{ij} \) (near \( y_i, y_j \)). So, changing \( \lambda \) should primarily affect global over local structure. Thus, \( \lambda < 1 \) increases the magnitudes of forces on distant, repulsive point pairs, exaggerating repulsion of non-neighboring point pairs in the original dataset, which
we conjecture leads to greater cluster separation while setting $\lambda > 1$ leads to lower separation. Inspecting column 2, setting $\lambda < 1$ yields the anticipated effect of greater cluster separation: examining MNIST for $\lambda = 0.95 < 1$, ABSNE places each cluster of points further away from the others. Similarly, the purple and brown clusters are more separated from the other clusters with CIFAR-10. In column 3, setting $\lambda = 1.05 > 1$ yields a single large glob of points containing smaller globs corresponding to the same class, as expected. This setting could be useful if the user wishes to inspect “boundary” cases between embedding points with known classes.

The Importance of Blending $\alpha$ and $\beta$. While past work has individually applied the $\alpha$- and $\beta$- divergences to the SNE problem (Yang et al., 2014; Bunte et al., 2012; Yang et al., 2013), the heavy dependence of the theory on $\lambda = \alpha + \beta$ shows that incorporating both divergences in the objective is crucial to discovering important micro and macro structures in data.

5. Parallel CPU and GPU Implementations

While many optimization methods exist for embeddings, e.g. spectral descent (Memisevic & Hinton, 2005), partial Hessian strategies (Vladymyrov & Carreira-Perpinan, 2012), fast multipole methods with L-BGFFS (Vladymyrov & Carreira-Perpinan, 2014), we found that warm-started gradient descent (van der Maaten, 2013) obtained strong results: we (1) initialize all $y_i$ from a 2D isotropic Gaussian with variance $10^{-4}$ and (2) update each $y_i$ via gradient descent (GD) with momentum (step size 200). For the first 250 descent iterations, we use momentum 0.5 and multiply all $P_{ij}$ values by a user-defined constant $\alpha = 12$. For the last 750 iterations, we use momentum 0.8. We use a per-parameter adaptive learning rate scheme to speed up GD convergence (Jacobs, 1988), otherwise known to be very slow and sensitive to local optima in practice (Vladymyrov & Carreira-Perpinan, 2012). Concrete considerations behind these choices can be found in (van der Maaten, 2008; 2013). We now detail how to compute gradients and update the $y_i$ on the GPU, particularly using the NVIDIA compute unified device architecture (CUDA).

5.1. Parallel GPU Gradients

We store the $y_i$ in two arrays on the GPU, one array per dimension, to take advantage of cache locality, as done in (Burtscher & Pingali, 2011). We store the (sparse) affinity matrix $P$ as a list of triplets. We take advantage of $P$’s symmetry to minimize the number of memory reads by storing only the upper half of the matrix. Our implementation consists of 13 kernels, which we now discuss.

Kernels 1 – 5: Partially Computing Force 2. Recall from Section 3 that the ABSNE gradient consists of two types of forces: Force 1 and Force 2. We first compute Force 2, since it will yield structures useful in computing Force 1.

Discussed in Section 2, computing Force 2 involves building a quadtree over the $y_i$. To do this, we (Kernel 1) construct a bounding box over the $y_i$, (Kernel 2) hierarchically subdivide the bounding box until each cell contains at most a single $y_i$, and (Kernel 3) compute the center of mass and cumulative mass per cell. Next, we (Kernel 4) perform an in-order traversal of the quadtree, which approximately places nearby cells next to each other in the traversal; this is crucial in accelerating Kernel 5, which actually computes the forces on the individual $y_i$.

To understand why the in-order traversal is necessary, recall that in CUDA, all threads within a single warp will execute in lockstep on entering a conditional statement only if the conditional evaluation is identical for all threads; otherwise, threads within the warp belonging to different branches will execute serially, often severely affecting performance. The in-order traversal substantially reduces such within-warp thread divergence in Kernel 5, leading to an order of magnitude savings in run-time. For more details, see (Burtscher & Pingali, 2011), which we follow closely in implementing these five kernels.

Jointly, this set of kernels computes and stores $g_1 = \sum_j 4ZQ^2_{ij}(y_i - y_j)$ and $g_2 = \sum_j 4ZQ^2_{ij}(y_i - y_j)Q_{ij}^{\alpha+\beta-1}$ in GPU memory. We incorporate the contributions of $J_1$ and $J_2$ in later kernels.

Kernel 6, 7: Computing $J_2$ and $Z$. We evaluate $J_2 = \sum_{k\neq i} Q_{ki}^{\alpha+\beta}$ by (1) computing a $J_2^{(i)} = \sum_{k\neq i} Q_{ki}^{\alpha+\beta}$ per $y_i$, and (2) executing a reduction to compute $J_2 = \sum_i J_2^{(i)}$. We efficiently perform (1) by computing auxiliary $J_2^{(i)}$ variables per $y_i$ in executing Kernel 5. We perform the reduction through the open-source Thrust library (Bell & Hogerock, 2011). We compute $Z$ similarly.

Kernel 8, 9: Computing $J_1$, Partially Computing Force 1. Rather than computing Force 1 by iterating over each $y_i$, we iterate through the positive entries of $P$, whose contributions we add to the prescribed $y_i$, (Kernel 8). To parallelize computation, we split the list of triplets $(i, j, P_{ij})$ across enough blocks to allow for 1024 threads per block; each thread processes a single triplet and updates points $i$ and $j$. One caveat of this approach is that these updates must be made atomically; e.g., parallel updates for tuples $(2, 3)$ and $(3, 5)$ without atomic updates would yield a race condition for $y_3$. Kernel 8 computes and stores $g_3 = \sum_j 4ZQ^2_{ij}(y_i - y_j)P_{ij}^{\alpha}Q_{ij}^{\beta-1}$ in GPU memory.

Computing $J_1$ entails iterating through the positive entries of $P$, computing the associated summands, and executing a reduce operation (Kernel 9). With large datasets, the GPU cannot fully store $P$ in memory; so, we swap batches of $P$ between CPU and GPU memory and apply Kernels 8 and
We now explore ABSNE for use in such goal-driven exploration. We strongly encourage readers to use a computer with access to a GPU for this purpose.

6. Case Studies

Evaluating visualization quality is a difficult problem; in previous work, researchers have used supervised labels in tandem with a k-nearest neighbors metric to quantitatively assess performance (van der Maaten, 2008; 2013; Yang et al., 2009). However, such scores may not directly translate to “better visualization”: For example, a data-miner aspiring to explore micro, within-class clusters in a dataset without supervised labels may assign a low score to an embedding that perfectly separates high-level macro clusters.

We now explore ABSNE for use in such goal-driven visualization. We strongly encourage readers to use a computer in tandem with digital zoom set to around 400% in looking at the large visualization in this section.
**Figure 5.** Please use digital zoom at high levels to view details. (a - c) depict ILSVRC 2012’s micro (specific animal species) in the purple, blue panels and macro (classes under the animal kingdom) structures in the red, green panels. As expected, macro clusters are more widely separated in (c) than (a) and in the red, green panels than purple, blue panels. (h - j) depict analogous structures in the HIGGS and SUSY datasets; we are searching for what physical meaning these structures have in the Standard Model of particle physics.
6.1. ILSVRC 2012: Kingdoms to Species

A subset of ImageNet, ILSVRC 2012 (Deng et al., 2009) features a training set of 1.28M images of varying size, validation set of 50K images, and 1000 object categories. We employ fc7 features yielded by Alexnet (Krizhevsky et al., 2012) implemented in Caffe (Jia et al., 2014), trained on the 1.28M images. We show a t-SNE plot of the validation set in the top-middle of Figure 5; we also present plots and 2 zoom views for $(\alpha = 0.8, \lambda = 1)$ and $(\alpha = 0.95, \lambda = 0.98)$. Zoom views present a random subset of images from the plot; displayed images were not hand-picked. As Section 4 predicts, the top scatter plots show tight clusters and greater class separation in $(\alpha = 0.95, \lambda = 0.98)$ compared with the other settings, particularly clusters corresponding to birds, mammals, and dogs. The maroon oval in the purple panel shows a transition between perched birds to monkeys in trees, while the blue and orange boxes in the red panel show well-separated classes. As expected, we found stronger intra-class clustering for $(\alpha = 0.8, \lambda = 1)$; the purple panel shows separate clusters for hummingbirds (green), macaw parrots (yellow), toucans (orange), flamingos (blue), chickens (red), ducks (purple), and birds in the sky (gray). Similarly, the blue panel separately clusters dalmanets (green), Bernese mountain dogs (blue), black and brown dogs, e.g. Doberman Pinschers and German Rottweilers (orange), and black dogs (red). In comparison, the red panel associated with $(\alpha = 0.95, \lambda = 0.98)$ shows fewer, vague clusters with all birds (orange): chickens (red) and swans/ducks (purple). The green panel shows fewer classes: Bernese mountain dogs (blue) and black and brown dogs (orange).

6.2. HIGGS: Discovering Higgs Bosons

We return to the HIGGS dataset (Baldi et al., 2014), whose training set consists of 28 features: the first 21 describe kinematic properties measured by detectors in the accelerator, while the last 7 are high-level functions of the first 21. Again, we first run t-SNE on the HIGGS dataset (bottom left of Figure 5). For HIGGS and SUSY, we first initialize and optimize with 0.5M random instances. Upon convergence, we place another 0.5M new random instances near their nearest neighbor in the original dataset with isotropic Gaussian noise with variance 0.1. Repeating until all points have been embedded, this produces final objectives with lower value.

Interestingly, while clustering occurs, the clusters don’t correspond to the desired classes. Hoping that tighter global clustering will lead to more intuitive results, we set $\alpha = 0.98$ (bottom middle plot). As predicted in Section 4, straggling points align with existing clusters to yield a cleaner plot. While some clusters appear to have slightly denser concentrations of positive signals, there still is not any concrete class separation. While we are not aware of what the resulting clusters mean, we believe that the clusters could yield further insights.

6.3. SUSY: Discovering Supersymmetric Particles

Discovering evidence of supersymmetry (SUSY) constitutes a major goal in the Large Hadron Collider’s central mission; one ramification of the theory includes the discovery of dark matter candidate particles (Baldi et al., 2014). We explore the SUSY dataset (Baldi et al., 2014), which similarly tries to distinguish between processes which do and do not produce supersymmetric particles. There is currently a vigorous effort to improve performance in this classification task (Cheng & Han, 2008; Barr et al., 2003; Rogan, 2010; Buckley et al., 2013). As with HIGGS, we found that setting $\alpha = 0.98, \lambda = 1$ yields a cleaner plot with greater separation than that of t-SNE (not shown due to lack of space). While we do not observe perfect cluster separation, there is a distinct purple region corresponding to observed SUSY particles. Perhaps, replicating experimental conditions leading to particles in this region would have a higher chance of yielding SUSY particles.

7. Discussion

Although several papers have explored varying divergences in SNE methods, this is the first paper to theoretically attribute and empirically verify how divergence parameters qualitatively affect visualization. Our analysis reveals that parameter variation in $(\alpha, \beta)$ for the AB-divergence discovers micro and macro structures within data in a dataset-agnostic fashion. Our well-optimized GPU implementation yields speedups of more than 20x on datasets with 1 – 10M instances over the state of the art implementation (Vladymyrov & Carreira-Perpinán, 2014). This yields the largest published SNE visualization of a dataset (11M instances).

8. Acknowledgements

We thank Eriz Tzeng, Animesh Garg, and Teodor Moldovan for useful discussions and NVIDIA for donating 2x Tesla K40 GPUs. KN was funded by an NSF Graduate Fellowship (Summer 2014) and by an NDSEG Fellowship (Fall 2014 - 2015). AP was partially funded by NSERC. This research was also funded in part by ONR through a Young Investigator Program award.
References


Supplementary Materials: Alpha-Beta Divergences Discover Micro and Macro Structures in Data

We first discuss details regarding continuity in Alpha-Beta SNE which allow us to make settings, e.g., \((\alpha, \alpha + \beta) = (1, 1)\) used in t-SNE, while bypassing issues arising from singularity. We then display a smattering of ABSNE plots across a large selection of datasets.

1. Continuity and Gradients for the Alpha-Beta Objective

The form of the original Alpha-Beta divergence used in the paper, defined \(\mathcal{F}(\mathcal{E}; \alpha, \beta) = D^{\alpha\beta}_{AB}(P||Q)\), is computed as:

\[
\frac{1}{\alpha\beta} \sum_{i \neq j} \left( -P^{\alpha}_{ij} Q^{\beta}_{ij} + \frac{\alpha}{\alpha + \beta} P^{\alpha+\beta}_{ij} + \frac{\beta}{\alpha + \beta} Q^{\alpha+\beta}_{ij} \right),
\]

where \(\alpha \in \mathbb{R} \setminus \{0\}, \beta \in \mathbb{R}\) are hyperparameters. According to (Cichocki et al., 2011), to account for cases such as \(\beta = 0\) and \(\alpha + \beta = 0\) in this objective, we can re-define:

\[
D^{\alpha\beta}_{AB}(P||Q) = \sum_{ij} d^{(\alpha,\beta)}_{AB}(P_{ij}, Q_{ij}),
\]

where:

\[
d^{(\alpha,\beta)}_{AB}(P_{ij}, Q_{ij}) = \begin{cases} 
-\frac{1}{\alpha\beta} \left( P^{\alpha}_{ij} Q^{\beta}_{ij} - \frac{\alpha}{\alpha + \beta} P^{\alpha+\beta}_{ij} - \frac{\alpha}{\alpha + \beta} Q^{\alpha+\beta}_{ij} \right), & \alpha, \beta, \alpha + \beta \neq 0 \\
\frac{1}{\alpha^2} \left( P^{\alpha}_{ij} \ln \frac{P^{\alpha}_{ij}}{Q^{\alpha}_{ij}} - P^{\alpha}_{ij} + Q^{\alpha}_{ij} \right), & \alpha \neq 0, \beta = 0 \\
\frac{1}{\alpha^2} \left( \ln \frac{Q^{\alpha}_{ij}}{P^{\alpha}_{ij}} + P^{\alpha}_{ij} - 1 \right), & \alpha = -\beta \neq 0 \\
\frac{1}{\beta^2} \left( Q^{\beta}_{ij} \ln \frac{Q^{\beta}_{ij}}{P^{\beta}_{ij}} + P^{\beta}_{ij} \right), & \alpha = 0, \beta \neq 0 \\
\frac{1}{2} \left( \ln P_{ij} - \ln Q_{ij} \right)^2, & \alpha = \beta = 0
\end{cases}
\]

As described in the paper, note that we obtain the KL-divergence by setting \(\alpha = 1, \beta = 0\) and the Itakura-Saito divergence by setting \(\alpha = 1, \beta = -1\), since \(P\) and \(Q\) are probability distributions. Given these various definitions, we would like to ensure that the gradient described in the paper is the same particularly for the first three cases (namely where \(\alpha \neq 0\)), as these are the ones that arise in our reductions. Specifically, we need to ensure that the gradients \(\partial D^{(\alpha,\beta)}_{AB}(P||Q)/\partial y_i\) all match.
Case 1: $\alpha, \beta \neq 0$. We have that:
\[
\frac{\partial D_{AB}^{(\alpha, \beta)}(P \parallel Q)}{\partial y_i} = \frac{1}{\alpha \beta} \left[ \frac{1}{\alpha \beta} \left( P_{ij}^{\alpha} Q_{ij}^{\beta} \right) - \frac{\alpha}{\alpha + \beta} P_{ij}^{\alpha + \beta} - \frac{\beta}{\alpha + \beta} Q_{ij}^{\alpha + \beta} \right] \quad (4)
\]
\[
= \frac{\partial}{\partial y_i} \left[ \frac{1}{\alpha \beta} P_{ij}^{\alpha} Q_{ij}^{\beta} + \frac{1}{\alpha (\alpha + \beta)} Q_{ij}^{\alpha + \beta} \right] \quad (5)
\]
\[
= \frac{\partial Q_{ij}}{\partial y_i} \cdot \left[ -\frac{1}{\alpha} P_{ij}^{\alpha} \cdot Q_{ij}^{\beta - 1} + \frac{1}{\alpha (\alpha + \beta)} \cdot (\alpha + \beta) \cdot Q_{ij}^{\alpha + \beta - 1} \right] \quad (6)
\]
\[
= \frac{\partial Q_{ij}}{\partial y_i} \cdot \left[ -\frac{1}{\alpha} P_{ij}^{\alpha} \cdot Q_{ij}^{\beta - 1} + \frac{1}{\alpha} \cdot Q_{ij}^{\alpha + \beta - 1} \right] \quad (7)
\]
\[
= -\frac{1}{\alpha} \cdot \frac{\partial Q_{ij}}{\partial y_i} \cdot \left[ P_{ij}^{\alpha} Q_{ij}^{\beta - 1} - Q_{ij}^{\alpha + \beta - 1} \right] \quad (8)
\]
In the remainder of the cases, we set $\beta$ to the appropriate value to demonstrate that the gradients match.

Case 2: $\alpha \neq 0$, $\beta = 0$. We have that:
\[
\frac{\partial D_{AB}^{(\alpha, \beta)}(P \parallel Q)}{\partial y_i} = \frac{1}{\alpha} \left[ \frac{1}{\alpha} \ln \frac{P_{ij}^{\alpha} Q_{ij}^{\alpha}}{Q_{ij}^{\alpha + 1}} \right] \quad (9)
\]
\[
= \frac{\partial}{\partial y_i} \left[ -\frac{1}{\alpha} \ln Q_{ij}^{\alpha} + \frac{1}{\alpha^2} Q_{ij}^{\alpha} \right] \quad (10)
\]
\[
= \frac{\partial Q_{ij}}{\partial y_i} \cdot \left[ -\frac{1}{\alpha} P_{ij}^{\alpha} \cdot Q_{ij}^{\alpha - 1} \cdot \frac{1}{\alpha} Q_{ij}^{\alpha - 1} \right] \quad (11)
\]
\[
= \frac{\partial Q_{ij}}{\partial y_i} \cdot \left[ -\frac{1}{\alpha} P_{ij}^{\alpha} \cdot Q_{ij}^{\alpha - 1} \right] \quad (12)
\]
\[
= -\frac{1}{\alpha} \cdot \frac{\partial Q_{ij}}{\partial y_i} \cdot \left[ P_{ij}^{\alpha} Q_{ij}^{\alpha - 1} - Q_{ij}^{\alpha - 1} \right] \quad (13)
\]
Setting $\beta = 0$ in Equation (8), we observe that the gradients match, as desired.

Case 3: $\alpha = -\beta \neq 0$. We have that:
\[
\frac{\partial D_{AB}^{(\alpha, \beta)}(P \parallel Q)}{\partial y_i} = \frac{1}{\alpha^2} \left( \ln Q_{ij}^{\alpha} + P_{ij}^{\alpha} - 1 \right) \quad (14)
\]
\[
= \frac{\partial}{\partial y_i} \left[ \frac{1}{\alpha^2} \ln Q_{ij}^{\alpha} + \frac{1}{\alpha^2} P_{ij}^{\alpha} \right] \quad (15)
\]
\[
= \frac{\partial Q_{ij}}{\partial y_i} \cdot \left[ \frac{1}{\alpha^2} \cdot \alpha Q_{ij}^{\alpha - 1} \cdot \frac{1}{\alpha} \cdot P_{ij}^{\alpha} - \frac{1}{\alpha} \cdot \frac{1}{Q_{ij}^{\alpha + 1}} \right] \quad (16)
\]
\[
= -\frac{1}{\alpha} \cdot \frac{\partial Q_{ij}}{\partial y_i} \cdot \left[ P_{ij}^{\alpha} Q_{ij}^{\alpha - 1} - 1 \right] \quad (17)
\]
Setting $\beta = -\alpha$ in Equation (8), we again observe that the gradients match, as desired. It follows that for all cases in the paper that we explore, we can simply use the gradient in Equation (8).

2. Embedding Plots of Various Datasets

In Figures 1, 2 and 3 we showcase visualizations of datasets on which we presented some results in the main paper, but did not include plots due to lack of space.

These plots aim to show the impact of $\alpha$ and $\beta$ on qualitative properties of the embedding. As discussed in the paper: changing $\lambda$ should primarily affect global over local structure, where (i) $\lambda < 1$ should lead to greater cluster separation
while (ii) $\lambda > 1$ should lead to low separation. Further, ABSNE should tend to produce lots of small, fine-grained clusters for $\alpha < 1$ with few global changes in visualization while $\alpha > 1$ should lead to fewer, larger clusters with more global visualization changes.

We directly use pixels for the ORL and COIL-20 vision datasets, while we compute fc7 features yielded by Caffe’s ImageNet model (Jia et al., 2014) for Caltech256. We use raw features for the other datasets. For all datasets with over 100 dimensions, we first apply 100 dimensional PCA before computing neighborhood scores.

References


Alpha-Beta Divergences Discover Micro and Macro Structures in Data

Figure 1. ABSNE visualizations for (rows from the top) ATT-Faces, Caltech256, COIL20, Segmentation Datasets. The left column corresponds to t-SNE. The center and right column contain visualizations with $\alpha$ and $\lambda$ set manually to emphasize local or global clustering. These two parameters can be used by a data scientist for goal-driven exploratory data visualization.
Figure 2. ABSNE visualizations for (rows from the top) CIFAR10, MNIST, ILSVRC2012 (validation) Datasets. The left column corresponds to t-SNE. The center and right column contain visualizations with $\alpha$ and $\lambda$ set manually to emphasize local or global clustering. These two parameters can be used by a data scientist for goal-driven exploratory data visualization.
Figure 3. BSNE visualizations for (rows from the top) WDBC, WINE, IRIS Datasets. The left column corresponds to t-SNE. The center and right column contain visualizations with α and λ set manually to emphasize local or global clustering. WINE and IRIS are very small datasets with clear clusters, and ABSNE plots do not differ widely from t-SNE.