Autism Classification Using Topological Features and Deep Learning: A Cautionary Tale
Supplementary Material

Archit Rathore[1], Sourabh Palande[1], Jeffrey Anderson[1], Brandon A. Zielinski[1], P. Thomas Fletcher[2], and Bei Wang[1]

[1] University of Utah, Salt Lake City, UT 84112, USA
{archit, sourabh, beiwang}@sci.utah.edu, andersonjeffs@gmail.com, brandon.zielinski@utah.edu
[2] University of Virginia, Charlottesville, VA 22904-4259, USA
ptf8v@virginia.edu

1 Mathematical Formulations

We experiment with four different kernels for persistence diagrams to train SVM classifiers. In this section, we give mathematical formulations of these kernels. They are computed using implementations provided in the Python package **sklearn_tda**, see [2] for implementation details. We also provide details of the projection layer defined by Hofer et al. [5] in their neural network architecture. For an introduction to persistent homology, see [4].

1.1 Kernels for Persistence Diagrams

An (ordinary) i-dimensional persistence diagram A is the disjoint union of a multi-set of off-diagonal points $\{(b,d) \mid b \neq d, b, d \in \mathbb{R}_{\geq 0}\}$ on the Euclidean plane $\mathbb{R}^2$ (where $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$) and the diagonal $\Delta = \{(b,b) \mid b \in \mathbb{R}_{\geq 0}\}$ counted with infinite multiplicity.

Persistence scale-space kernel [8]. For two i-dimensional persistence diagrams A and B, the persistence scale-space kernel K_S is defined as:

$$K_S(A, B, \sigma) = \frac{1}{8\pi\sigma} \sum_{p \in A, q \in B} e^{\frac{\|p-q\|}{8\sigma}} - e^{\frac{\|p-\overline{q}\|}{8\sigma}}, \tag{1}$$

where $\forall\ q = (b,d) \in B_i$, $\overline{q} = (d,b)$. In our experiments, we only consider 0- and 1-dimensional topological features. Correspondingly, we compute scale-space kernels for 0- and 1-dimensional persistence diagrams which can then be combined using a weighted average.

Persistence weighted Gaussian kernel [6]. For two i-dimensional persistence diagrams A and B, the persistence weighted Gaussian kernel K_G is defined as:

$$K_G(A, B, \sigma) = \sum_{p \in A, q \in B} w(p) w(q) e^{-\frac{\|p-q\|^2}{2\sigma^2}}, \tag{2}$$

where $w(p)$ is the weight assigned to the point p. Kusano et al. [6] suggest $w(p) = \arctan(C(d-b)^t)$ as the weight for $p = (b, d)$, where C is a positive constant for practical purposes and t is assumed to be greater than the dimension of the underlying space. In our experiments, we use the default weight function implemented in the **sklearn_tda** package which assigns a unit weight to each point in the persistence diagram.

Sliced Wasserstein kernel [3]. Let A and B be i-dimensional persistence diagrams. Given a unit vector θ in $\mathbb{R}^2$, let $L(\theta) = \{\lambda\theta \mid \lambda \in \mathbb{R}\}$ denote the line and $\pi(\theta, p)$ denote the orthogonal projection of point p on the line $L(\theta)$. To compute the sliced Wasserstein kernel, we first augment persistence diagram A with the orthogonal projection π_Δ of points in B and vice-versa to obtain two new sets A^* and B^*. The sliced Wasserstein distance between these two sets is approximated as:

$$SW(A^*, B^*, M) = \frac{1}{\pi} \sum_{j=1}^{M} \|V(A^*, \theta_j) - V(B^*, \theta_j)\|_1,$$

where $\theta_j = j\pi/M - \pi/2$ and $V(A^*, \theta_j)$ is the vector of dot products $< p, \theta_j >$ of all points $p \in A^*$. The sliced Wasserstein kernel is then computed as:

$$K_W(A^*, B^*, M) = e^{\frac{-SW(A^*, B^*, M)}{2\sigma^2}}, \tag{3}$$

In our experiments, we fix the number of directions $M = 10$. The bandwidth σ is determined using grid search.

Persistence Fisher kernel [7]. Given an i-dimensional persistence diagram A and a bandwidth $\sigma > 0$, one can define a smooth, normalized measure

$$\rho_A = \left[\frac{1}{Z} \sum_{u \in A} N(x; u, \sigma I) \right]_{x \in \Theta}$$

over a given set Θ, where I is the identity matrix, N is a Gaussian function and $Z = \int_\Theta \sum_{u \in A} N(x; u, \sigma I) \mathrm{d}x$. Note that if Θ is the entire Euclidean space $\mathbb{R}^2$ then ρ_A is a probability distribution similar to the case of persistence images [1]. Given two i-dimensional persistence diagrams A and B, we obtain two new sets A^* and B^* by augmenting A with the orthogonal projection of points of B on the diagonal and vice-versa. For these two sets, the persistence Fisher kernel is defined as:

$$K_F(A^*, B^*) = e^{-t d_F(A^*, B^*)} \tag{4}$$

where $t > 0$ is a scalar parameter and d_F is the Fisher information metric defined as follows:

$$d_F(A^*, B^*) = \arccos\left(\int \sqrt{\rho_{A^*}(x)\ \rho_{B^*}(x)} \mathrm{d}x \right)$$

In our experiments, we fix the bandwidth $\sigma = 1.0$ needed to compute the measure ρ. The kernel parameter t is tuned using grid search.

1.2 Neural Networks with Topological Features

To train a neural network with persistence diagrams, we use the approach proposed by Hofer et al. [5], where a projection layer is defined as a set of nodes in the neural network that takes a persistence diagram as input and outputs an n-dimensional vector.

Denote an i-dimensional persistence diagram as A and a point in the diagram $(b, d) \in A$. A change of coordinates $(x, y) = (b + d, d - b)$ is applied to each point of A. Let $\boldsymbol{\mu} = [\mu_x, \mu_y]$ and $\boldsymbol{\sigma} = [\sigma_x, \sigma_y]$ denote the location and scale of a structure element. Define $s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}$ as follows:

$$s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}(x, y) = \begin{cases} e^{-\sigma_x^2(x-\mu_x)^2 - \sigma_y^2(y-\mu_y)^2} & y \in [\nu, \infty) \\ e^{-\sigma_x^2(x-\mu_x)^2 - \sigma_y^2(\ln \frac{y}{\nu} + \nu - \mu_y)^2} & y \in (0, \nu) \\ 0, & y = 0 \end{cases}$$

where ν is a cutoff parameter for handling points with persistence (difference between birth and death) close to zero. The projection of an i-dimensional persistence diagram A w.r.t to $s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}$ is

$$S_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu} = \sum_{(x,y)\in\rho(A)} s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}(x, y).$$

Note that $S_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}$ maps A to a scalar value. Now, suppose $\theta = \{(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)\}_{i=1}^n$ is a set of location and scale parameters for n structure elements. The projection layer $\mathcal{S}_{\theta,\nu}$ for A is composed of n nodes, where each node corresponds to one structure element $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ and outputs the projection $S_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}$ of A. Each of the n nodes output one scalar value and these scalar values are concatenated to form the output vector. Note that projection layers are defined independently for each dimension of the persistence diagram. Hofer et al. [5] show that the function $s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}(x, y)$ is stable with respect to the Wasserstein and bottleneck distances and differentiable with respect to parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ so that gradient descent can be applied to train the neural network.

2 Implementation Details

2.1 Neural Networks

We implement the neural networks using the **Pytorch** Python library. The layer sizes for the networks with correlation features are as follows:

- 3 Layers: n-1,000-100-10-2
- 5 Layers: n-10,000-5,000-1,000-100-10-2
- 7 Layers: n-10,000-5,000-1,000-500-100-50-10-2

For CC200 $n = 19,900$ and for CC400 $n = 79,800$. For the hybrid neural network, the layer sizes after the concatenation layer are identical to the above. We use 50 structure elements for both 0-dimensional and 1-dimensional features for CC200 and 100 structure elements for CC400. This makes the output size of concatenation layer to be 20,000 (50+50+19,900) for CC200 and 80,000 (100+100+79,800) for CC400.

Site	SVM_{corr}	$NN3_{corr}$	$NN3_{PD+corr}$
CALTECH	61.23	66.18	67.78
CMU	61.74	65.28	68.43
KKI	61.73	67.19	67.24
LEUVEN	60.95	65.64	67.83
MAX_MUN	60.55	66.32	67.75
NYU	62.39	64.88	68.48
OHSU	61.40	65.58	68.46
OLIN	63.23	65.42	67.02
PITT	59.41	64.60	67.98
SBL	60.99	65.66	66.82
SDSU	62.22	65.55	67.19
STANFORD	61.76	65.75	67.69
TRINITY	63.57	65.56	67.56
UCLA	61.45	66.12	67.40
UM	62.02	65.70	67.34
USM	63.20	65.94	67.94
YALE	61.90	66.29	67.63
Mean	61.74	65.74	67.68

Table 1: Accuracies for various models using a leave-one-site-out cross validation scheme.

2.2 Parameter Tuning via Grid Search

We tune parameters for the baselines models using grid search. We split the dataset into 60% training, 20% validation, and 20% testing sets. We first specify the possible set of values for each of the tunable parameters of the model. We then train the model by using each point in the Cartesian product of the parameter sets using the training set and store the parameters with best performance on the validation set. The model is then trained using the best parameters on data obtained by merging the training and validation sets and performance is evaluated on the test data set. This process is exponential in the cardinality of parameters. Continuous parameters are discretized by two strategies. For unbounded parameters, the parameter set is created by taking a reasonable range and taking values that differ by orders of magnitudes. For bounded parameters, a fixed step size is used to create the parameter set.

3 Additional Results

3.1 Cross-Site Variation of Accuracy

We also perform leave-one-site-out cross validation for SVM_{corr}, NN_{corr} and $NN3_{PD+corr}$ models. In this scheme, one site is used as test data while the model is trained on all other sites. Due to site-wise differences in equipments and processing methods, prediction is expected to be more challenging for data from previously unseen sites. This is reflected in our results in Table 1 where

the mean accuracies of the models is lower than models trained using the 5-fold cross validation scheme.

3.2 Dimensionality Reduction on Correlation Features

We evaluate the SVM_{corr} model on lower dimensional projection of the CC200 and CC400 datasets using principal component analysis (PCA) . We keep the components corresponding to top k eigenvalues that explain 99% variance in each dataset. This leads to a value of $k = 45$ for CC200 and $k = 67$ for CC400 and yields classification accuracy of 65.12% and 65.24% respectively.

References

1. Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., Ziegelmeier, L.: Persistence images: A stable vector representation of persistent homology. Journal of Machine Learning Research **18**(1), 218–252 (2017)
2. Carriére, M.: sklearn-tda: a scikit-learn compatible python package for machine learning and tda. https://github.com/MathieuCarriere/sklearn-tda
3. Carrière, M., Cuturi, M., Oudot, S.: Sliced Wasserstein kernel for persistence diagrams. Proceedings of the 34th International Conference on Machine Learning **70**, 664–673 (2017)
4. Edelsbrunner, H., Harer, J.: Persistent homology–a survey. Contemporary mathematics **453**, 257–282 (2008)
5. Hofer, C., Kwitt, R., Niethammer, M., Uhl, A.: Deep learning with topological signatures. Advances in Neural Information Processing Systems pp. 1634–1644 (2017)
6. Kusano, G., Fukumizu, K., Hiraoka, Y.: Kernel method for persistence diagrams via kernel embedding and weight factor. Journal of Machine Learning Research **18**(1), 6947–6987 (2017)
7. Le, T., Yamada, M.: Persistence fisher kernel: A Riemannian manifold kernel for persistence diagrams. Advances in Neural Information Processing Systems **31**, 10028–10039 (2018)
8. Reininghaus, J., Huber, S., Bauer, U., Kwitt, R.: A stable multi-scale kernel for topological machine learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp. 4741–4748 (2015)