Electronic supplementary materials *Mendeleev Commun.*, 2024, **34**, 786–787

Contrastive representation learning for spectroscopy data analysis

Artem P. Vorozhtsov and Polina V. Kitina

Benchmark dataset

The dataset contains synthetic spectra for 500 different classes, with each class represented by 50 objects in the training set, 10 objects in the validation set, and 9 objects in the test set. In our work, the model was trained on a reduced training set (20 or 5 objects for each of the 500 classes), the validation and test sets were used in this paper without changing the number of objects. Each synthetic spectrum is a one-dimensional array of 5000 elements. Figure S1 shows an example spectrum from the original dataset. The horizontal axis corresponds to the position of the signal and the vertical axis corresponds to its intensity. Objects of the same class differ from each other by shifting the position of peaks, changing their intensities, as well as the width of peaks, which corresponds to the errors in recording real spectra.

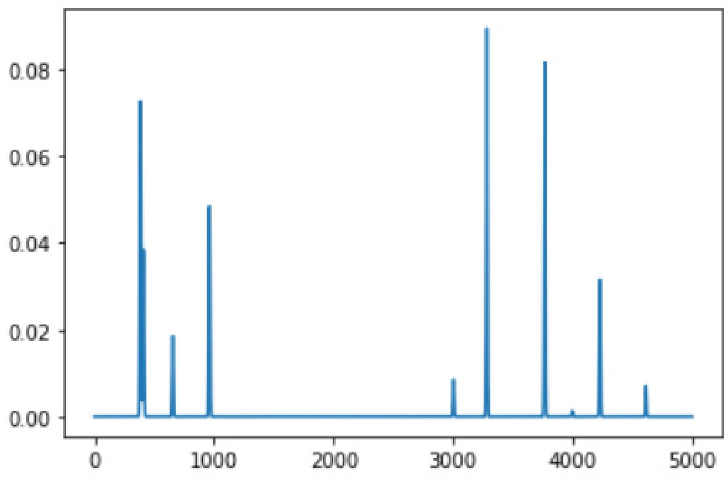

Figure S1 Sample data for training set

Data preprocessing consists of normalizing each vector of length 5000 by the maximum element of the vector: in this way data in the range [0; 1] are obtained. During training, normally distributed noise is added to spectra as data augmentation. An example of the data after processing is shown in Fig. S2.

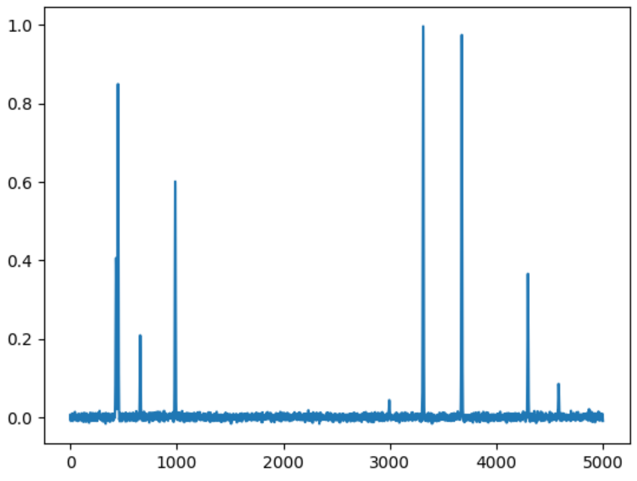

Figure S2 Data after preprocessing

Neural network architecture and its training process

A convolutional neural network (CNN) with 1D convolutions is used to extract high-level features because there are correlations between neighboring points in the spectrum. In addition, convolutional neural networks have translational invariance and therefore can respond adequately to peak position shifts. The architecture of the neural network is shown in Fig. S3.

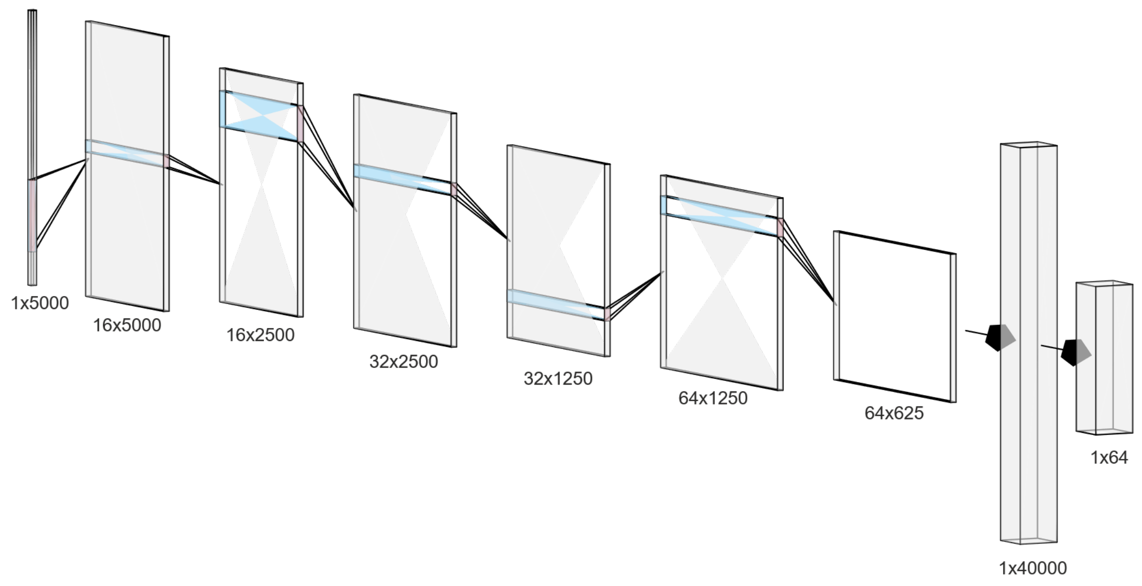

Figure S3 CNN architecture

The model consists of three convolution layers with specified values of the number of filters (16-32-64) and convolution kernel size (21-11-5), each convolution layer is followed by a pooling and a batch-normalization layer, the activation function between convolutional blocks of the neural network is LeakyReLU. After the convolutional layers, there is a flatten layer and a linear layer that translates the vector into the *latent_dim* dimension space. The model outputs are normalized. The result is the points of the latent representation space distributed on a unit sphere in this space. The dimensionality of the latent space was 64, this value was determined using the Optuna hyperparameter fitting library[S1]. It should be noted that increasing number of convolutional or linear layers in this architecture leads to fast overfitting.

A detailed description of all used layers of the neural network architecture is available in the PyTorch[S2] framework documentation.

To train the neural network, the preprocessed data are divided into triples: the anchor element is the spectrum of an object from the training set; the positive element is another example of the spectrum of an object of the same class as the anchor element; the negative element is the spectrum of an object of a class different from the anchor element. The spectra for the positive and negative element are chosen randomly.

TripletMarginLoss with cosine distance was used as the loss function:

$$TripletMarginLoss = \max(0, cosine\_dist(positive) - cosine\_dist(negative) + 1)$$

$$cosine\_dist(positive) = 1 - \cos(anchor, positive)$$

$$cosine\_dist(negative) = 1 - \cos(anchor, negative)$$

where anchor is the latent representation of the anchor element, positive and negative are the latent representations of positive and negative objects, respectively.

Hard example mining was used in the training. For this purpose, we optimized the loss function averaged only over "positive triplets", which included hard triplets and part of semi-hard triplets (fig. S4).

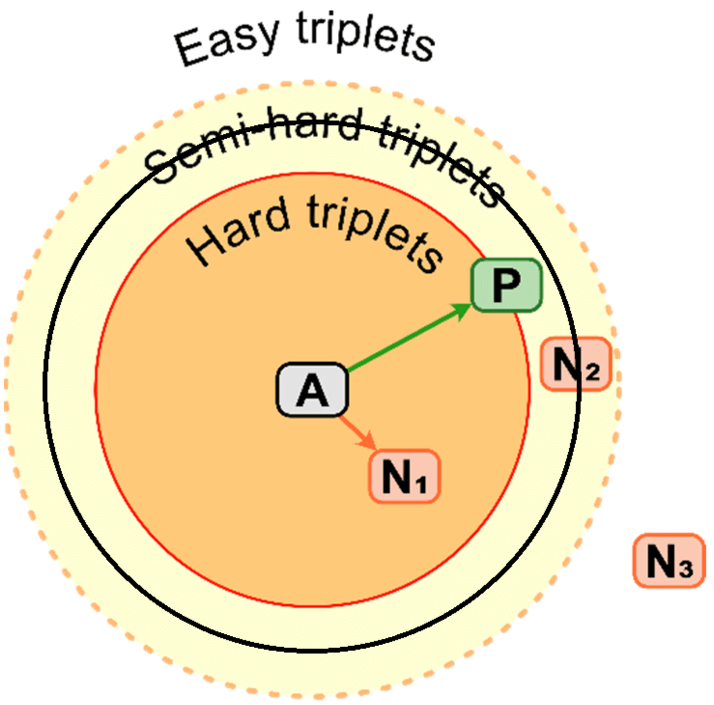

Figure S4 Hard example mining. A is anchor, P is positive element, N_i – negative elements

The AdamW optimizer was used to optimize the weights of the neural network, the initial value of the learning rate (*lr*) was $5 \cdot 10^{-3}$, every 4 epochs of training *lr* was reduced by a factor of 2. Training was performed for 20 epochs. During training, there was a decrease in the proportion of complex examples (Fig. S5 (a)) and in the value of the loss function on these complex examples (Fig. S5 (b)). Model training was performed on a single NVIDIA Tesla T4 in Google Colab.

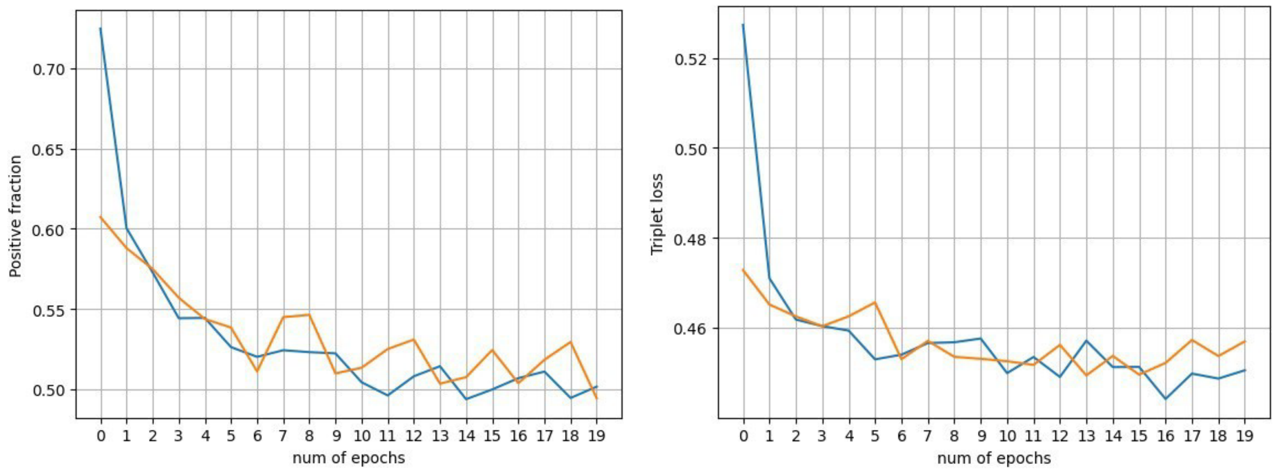

Figure S5 Training curve for fraction of positive triplets and loss function

After training the neural network, the optimal value of the number of nearest neighbors (*k*) in the k-NN algorithm was selected. For this purpose, latent representations of the validation set objects were obtained, for which the class was predicted using the sklearn.neighbors.KNeighborsClassifier algorithm[S3] with different values of the number of nearest neighbors. The optimal value of k was taken as the one that gives the maximum classification accuracy on the validation dataset. This value of k was used to classify the test set.

Training, validation and testing of the algorithm were performed 5 times with different initializations of random parameters, then the prediction accuracy on the test set was averaged.

Challenging dataset

The challenging dataset consists of spectra from 27 different classes. Each class is represented by 50 examples in the training sample, 10 examples in the validation sample, and 6 examples in the test sample. All classes are grouped into three groups: (1) classes that are characterized by peaks with low intensity, (2) classes in which the position of the peaks is slightly different from each other, (3) classes in which the position of the peaks is the same but their intensity is different. Training, validation and testing of the algorithm were performed similarly to the above procedure.

To identify patterns in the data that cause the algorithm to have the highest error, it was tested separately on each group of classes. Figure S6 shows the projections of the latent representation space onto the two-dimensional plane obtained by principal component analysis. The more complete separation of the classes of the second group can be clearly seen compared to the first and third groups, which is also consistent with the prediction accuracy of the algorithm on these groups separately.

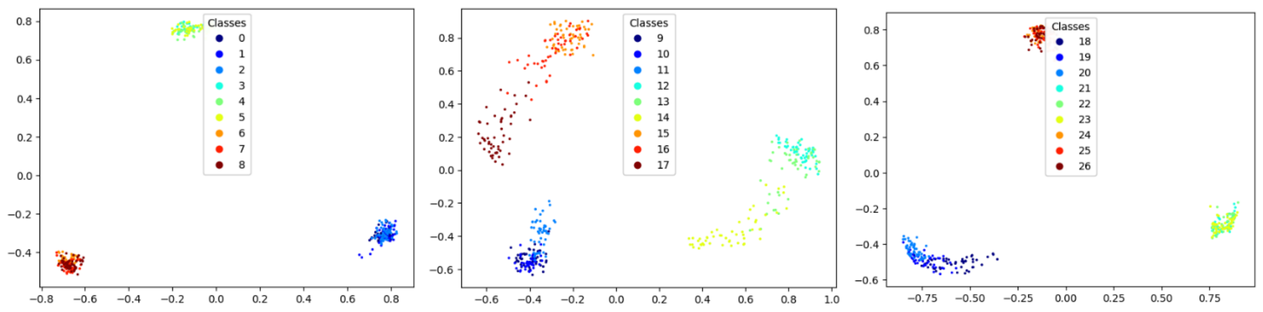

Figure S6 PCA analysis of latent space for challenging dataset

Processing mixtures of objects of different classes

To train the neural network, the preprocessed data are divided into triples: the anchor element is the spectrum of an object from the training set; the positive element is the normalized sum of another object of the same class as the anchor element and an element of another class; the negative element is the normalized sum of spectra of objects of two classes other than the anchor class. The spectra for the positive and negative element are chosen randomly.

To test the model, the following steps of the algorithm are performed:

1) Embeddings of spectra for each individual object class are generated using a pretrained neural network;
2) Within a class, the embeddings are averaged to obtain a vector characterizing this class of objects "on average". After averaging the embeddings are normalized;
3) Embeddings of different classes are stacked with each other to create all possible unique embeddings of mixtures with a given number of elements. After stacking, the embeddings are normalized. The array of these embeddings is passed for training to the k-NN algorithm with $k = 1$;
4) Embeddings are formed for the spectra of mixtures from the test sample.
5) Using the k-NN algorithm trained in step 3, the composition of the mixtures is determined from the embeddings obtained in step 4.

Metrics

The quality of model's predictions was measured by the value of accuracy, that defined as the fraction of correctly classified samples in test set. When the model was tested on individual objects, we used the built-in KNeighborsClassifier.score method. To test the model on mixtures, we wrote a function that determines the number of matching elements in the array of predicted classes included in the mixture and the array of classes actually included in the mixture. This function returned an array of accuracy values, with each value being the accuracy of identifying 1, 2, 3, or all 4 compounds in the mixture.

Code availability

All code and data are available on github: https://github.com/ArtemVorozhtsov/Contrastive-representation-learning-for-spectroscopy-data-analysis

References

S1 T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, *25th ACM SIGKDD international conference on knowledge discovery & data mining*, Anchorage, 2019, 2623; https://doi.org/10.1145/3292500.3330701.

S2 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, *Advances in Neural Information Processing Systems 32*, Vancouver, 2019, 8024; https://doi.org/10.48550/arXiv.1912.01703.

S3 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, *J. Mach. Learn. Res.*, 2011, **12**, 2825; https://doi.org/10.48550/arXiv.1201.0490.