Advanced Deep Learning and Kernel Methods Challenge 1

Gabriele Pintus
gabrielegavino.pintus@studenti.units.it

ABSTRACT

In this notebook we explore the efficacy of kernel methods in treating data before solving both unsupervised and supervised tasks. We work on the **Fashion MNIST** dataset which contains 70,000 28x28 grayscale images of fashion products from 10 categories from a dataset of Zalando article images, with 7,000 images per category. All the code is written in Python and it can be found in the GitHub repository at the following link: Challenge 1

1 UNDERSTANDING DATA GEOMETRY

We are given a dataset $D = \{(x_i, y_i) \quad i = 1, 2, \dots, n\}$ with x_i being the flattened image and y_i the label taking values in $\{0, 1, \dots, 9\}$.

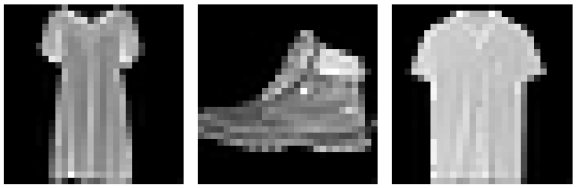

Figure 1: Three samples from the dataset corresponding to the categories: dress, ankle boot and T-shirt.

1.1 Data preparation

Before performing PCA, it's important to center the data. In fact, it's considered best practice to not only center the data but also standardize it.

$$X \leftarrow \frac{X - \mu_X}{\sigma_X} \tag{1}$$

1.2 Understanding PCA and KPCA

The **Principal Component Analysis** is a linear transformation of the data that allows us to project it in a new space in which the covariance matrix is diagonal and its eigenvalues are sorted from the largest to the smallest. Selecting the first k components allows us to reduce the dimensionality of the data minimizing the information loss.

However, in most cases, the orignal space in which our data lies does not make it easy to disentagle it. Often, to better disentagle our data it is necessary to project it onto a bigger space, here is where **Kernel PCA** comes in.

The covariance matrix can be seen as a matrix of similarities in which the inner product employed is the covariance, and since we are working in a centered space, the dot product. To compute similarities in a wider space that allow us to introduce non-linearities we can use kernels. The kernel matrix is defined as

$$K = k(x_i, x_j) \quad \forall i, j \tag{2}$$

where k is the kernel function. We then want to solve the same problem as before

$$Kv = \lambda v \tag{3}$$

where v are the eigenvectors and λ the eigenvalues. The projection is then computed as

$$y_k = \sum_{k=1}^{n} \alpha_{i,k} \cdot k(\mathbf{x}_i, \mathbf{x}) \tag{4}$$

where $\alpha_{i,k}$ are the coefficients of the k-th eigenvector.

1.3 Testing multiple kernels

Every kernel indirectly defines a Hilbert space of functions with some properties. The tested kernels are the following:

$$\begin{aligned}
\mathcal{H}_{\text{gaussian}} &= \left\{ f : \int |\hat{f}(w)|^2 e^{\frac{\sigma^2 w^2}{2}} dw < \infty \right\} \\
\mathcal{H}_{\text{poly2}} &= \{\} \\
\mathcal{H}_{\text{sigmoid}} &= \{\} \\
\mathcal{H}_{\text{cs}} &= \{\} \\
\mathcal{H}_{\text{laplace}} &= \left\{ f : \int |\hat{f}(w)|^2 \frac{\gamma^2 + w^2}{\gamma} dw < \infty \right\}
\end{aligned}$$

In addition to this we enunciate the following theorem.

Theorem 1.1. *Forall $i = 1, 2, \dots, M$, let $\phi_i : \mathcal{X} \to \mathcal{H}_i$ be a feature map such that*

$$K_i(x, y) = \langle \phi_i(x), \phi_i(y) \rangle_{\mathcal{H}_i}$$

Then the kernel K defined by

$$K(x, y) = \sum_{i=1}^{M} K_i(x, y)$$

can be expressed as

$$K(x, y) = \langle \phi_S(x), \phi_S(y) \rangle_{\mathcal{H}_S}$$

where

$$\phi_S : \mathcal{X} \to \mathcal{H}_S = \mathcal{H}_1 \bigoplus \mathcal{H}_2 \bigoplus \cdots \bigoplus \mathcal{H}_M$$

*is the **concatenation** of the feature maps ϕ_i:*

$$\phi_S(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_M(x) \end{bmatrix}$$

Moreover, the other following theorem allows us to understand what is happing from a functional point of view.

Theorem 1.2. *The solution $f^\star \in \mathcal{H}_{K_S}$ when we learn with $K_S = \sum_{i=1}^{M} K_i$ is equal to*

$$f^\star = \sum_{i=1}^{M} f_i^\star$$

where $(f_1^\star, f_2^\star, \dots, f_M^\star) \in \mathcal{H}_1 \times \mathcal{H}_2 \times \cdots \times \mathcal{H}_M$ is the solution of

$$\min_{f_1, f_2, \dots, f_M} R\left(\sum_{i=1}^{M} f_i^n\right) + \lambda \sum_{i=1}^{M} ||f_i||^2_{\mathcal{H}_{K_i}}$$

Both the theorems are taken from the slides of the course "Machine Learning with Kernel Methods" by J. Mairal, et al. [2].

In practice, this means we can define a new kernel as the sum of others and obtain a linear separator in the space made by the concatenation of the feature maps of the original kernels. Moreover, each kernel contribute with its regularization term to the final solution. The sum can be generalized to a weighted sum to obtain:

$$K(x, y) = \sum_{i=1}^{M} \eta_i K_i(x, y)$$

$$\min_{f_1, f_2, \dots, f_M} \mathrm{R}\left(\sum_{i=1}^{M} f_i^n\right) + \lambda \sum_{i=1}^{M} \frac{1}{\eta_i} ||f_i||^2_{\mathcal{H}_{K_i}}$$

Notice that the regularization term is now *weighted by the inverse* of the coefficients $\eta_i \geq 0$. This means that the solution is not invariant to a scaling of the coefficients vector η, and smaller values lead to a stronger regularization.

In the Multiple Kernel Learning framework proposed by Lanckriet, et al. [?], the coefficients η are learned by solving the following optimization problem:

We define our custom kernel as:

$$\begin{aligned} K_{\text{custom}}(x, y) &= K_{\text{gaussian}}(x, y) + K_{\text{cs}}(x, y) \\ &= e^{\gamma_1 ||x-y||^2} + \tanh\left(\gamma_2 \left(1 + \frac{\langle x, y \rangle}{||x|| ||y||}\right)\right) \end{aligned}$$

where K_{cs} is the kernel of the cosine similarity rescaled in the interval $[0, 2]$.

Notice that we are combining a kernel invariant to rigid motions with a kernel invariant to rotations (with respect to the origin) and scaling. The aim is to obtain a kernel which is invariant to all these transformations.

The hyperparameters of the kernels are the following:

- Gaussian: $\gamma = 0.1$
- Poly2: $\gamma = 0.1$
- Sigmoid: $\gamma = 0.1$, $\beta = 0$
- CS: –
- Custom: $\gamma_1 = 0.5$, $\gamma_2 = 1.6$

which have been chosen by manual tuning to maximize the explained variance. We then compute the explained variance ratio of the kernels and we plot it in the following figure.

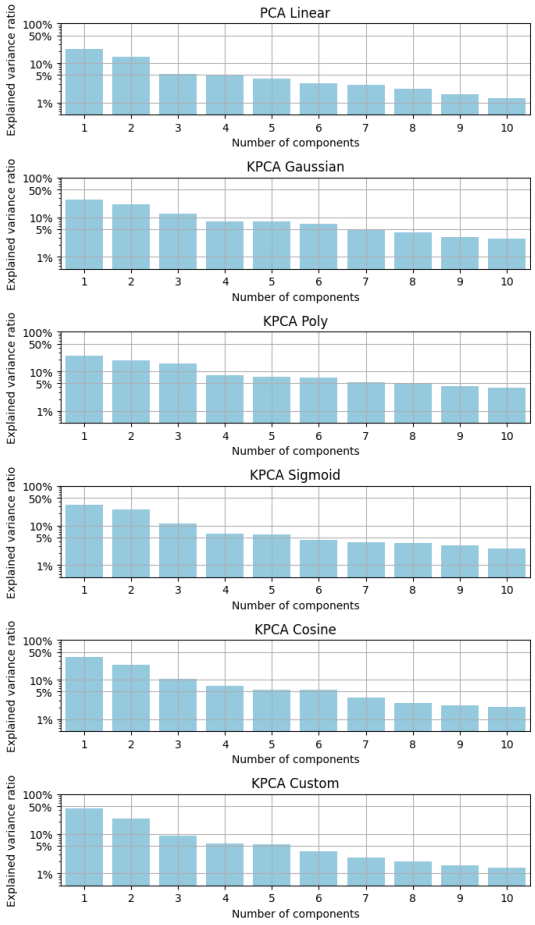

Figure 2: Explained variance ratio of the kernels.

The best kernel is the one with the highest decay of the explained variance. We proceed by computing the decay rate for each kernel as:

$$\text{decay}_i = \frac{\text{expvar}_{i-1} - \text{expvar}_i}{\text{expvar}_{i-1}}$$

Then we want to compute the average decay rate of the kernels, but since we prefer a decay rate which is higher in the first components, we weight this average by the harmonic series.

$$\text{HWDR} = \frac{\sum_{i=1}^{n} \frac{1}{i} \cdot \text{decay}_i}{\sum_{i=1}^{n} \frac{1}{i}}$$

The results are displayed in the following table.

Kernel	HWDR
Linear	31.3
Gaussian	25.1
Poly2	20.9
Sigmoid	28.9
CS	32.8
Custom	**38.5**

The custom kernel is the one with the highest HWDR, and it is the one we will use for the rest of the analysis. We can visualize the 3D projection of the data in the following plot.

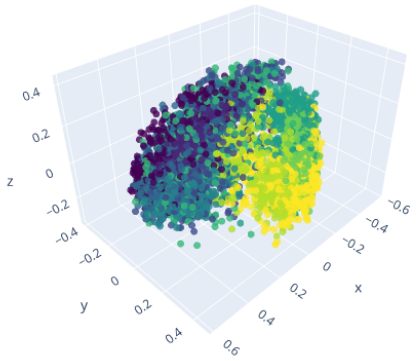

Figure 3: 3D projection of the data using the custom kernel.

2 BRIDGING UNSUPERVISED AND SUPERVISED

It is not always easy to retrieve labeled data, and in these cases we can use unsupervised learning to cluster the data and then assign the labels to the clusters.

2.1 Clustering

In this section we try to reconstruct the labels by solving a clustering task and finding the best assignment of the labels to the clusters. In order to do so we test the following clustering algorithms:

- KMeans
- Agglomerative Clustering
- Gaussian Mixture Model
- Spectral Clustering

The choide of these algorithms is motivated by the fact that we now a priori the number of clusters, which is the number of classes in the dataset. We then train each algorithm on the data and compute multiple metrics, such as the Adjusted Rand Index, the Normalized Mutual Information and the Silhouette Score. The results are displayed in the following table.

Algorithm	ARI	NMI	Silhouette
KMeans	0.318	0.464	**0.227**
Agglomerative	0.371	**0.541**	0.201
Gaussian Mixture	0.329	0.501	0.170
Spectral	**0.374**	0.505	0.202

From which we can see that the Gaussian Mixture Model is the best performing algorithm. To map the clusters to the labels we perform a manual inspection of the clusters and the labels by visualizing the ten points closest to the centroid of each cluster. We then assign the label to the cluster by majority voting. Because it is not clear how to assign the couples (pullover, shirt) and (dress, coat) we test each of the four possible assignments and we choose the one with the highest adjusted rand score. In the end we obtain the following mapping:

Cluster	Label
0	T-shirt/top
1	Sandal
2	Trouser
3	Pullover
4	Ankle boot
5	Bag
6	Shirt
7	Sneaker
8	Dress
9	Coat

2.2 Classification

We can now use the labels obtained from the clustering to train multiple classifiers and compare their performance. We consider Support Vector Machines with different kernels, a Multi Layer Perceptron and a Convolutional Neural Network. For both the Neural Networks a batch size of 64 was used and the training was performed for a maximum of 150 epochs with the early stopping mechanism of patience 10 epochs. The optimizer is Adam with the following hyperparameters:

- Learning rate: 10^{-4}
- β_1: 0.9
- β_2: 0.999
- ε: 10^{-8}

The "Reduce lr on plateau" learning scheduler is used with a factor of 0.5 and a patience of 5 epochs. The loss function is the cross entropy loss and the activation function is always the ReLU function, except for the output layer where the softmax function is used.

The MLP has 4 hidden layers, each with 128 neurons.

The CNN has 3 convolutional blocks composed by:

- 2D Convolutional Layer with kernel size 2, stride 1 and padding 1
- Batch Normalization Layer
- Activation Layer
- Max Pooling Layer with kernel size 2 and stride 2

The filters are 16, 32 and 64 respectively. In the end there are 2 fully connected layers with 64 neurons each.

The hyperparameters have been chosen manually following the guidelines described in *Deep Learning* by Goodfellow, et al. [1].

The results are displayed in the following table.

Model	Training Accuracy	Validation Accuracy
SVM (Linear)	16.3	**10.0**
SVM (Poly)	18.8	7.3
SVM (Gaussian)	**20.4**	6.5
SVM (Sigmoid)	10.4	8.6
MLP	**20.4**	5.9
CNN	16.3	**10.0**

Where the training accuracy is computed on the labels obtained from the clustering and the validation accuracy is computed on the original labels of the validation set.

3 FULLY SUPERVISED LEARNING

In this section we avoid learning the labels from the clustering and we directly train the classifiers on the original labels. We test them on both the KPCA reduced dataset and the original dataset.

3.1 KPCA reduced dataset

We train the same classifiers as before on the KPCA reduced dataset and we compare the results with the ones obtained in the previous section. The results are displayed in the following table.

Model	Training Accuracy	Validation Accuracy
SVM (Linear)	78.0	76.4
SVM (Poly)	80.1	77.1
SVM (Gaussian)	83.6	81.1
SVM (Sigmoid)	65.2	64.5
MLP	**87.1**	**82.5**
CNN	85.4	79.1

As we can see, the performance of the classifiers definitely improved when trained on the original labels.

3.2 Original dataset

We repeat the same experiment on the original dataset and obtain the following results.

Model	Training Accuracy	Validation Accuracy
SVM (Linear)	**100.0**	80.4
SVM (Poly)	89.6	83.2
SVM (Gaussian)	90.8	84.6
SVM (Sigmoid)	79.0	77.3
MLP	93.9	85.0
CNN	96.9	**86.4**

We can see that the performance of the classifiers improves even more when trained on the original dataset. We should also note that no tuning of the hyperparameters was performed in this section and we just used the same hyperparameters as in the previous section. By tuning them for this dataset we could obtain even better results.

4 CONCLUSIONS

In this report we have seen how Kernel PCA can be used to reduce the dimensionality of the data and how it can outperform PCA when the data lies in a non-linear manifold. We have also seen how unsupervised learning can be used to retrieve labels from the data and how these labels can be used to train classifiers. Finally, we have seen how the performance of the classifiers can improve when trained on the original labels.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, Cambridge, MA, 2016.

[2] J. Mairal, M. Arbel, and J.-P. Vert (virtually), "Machine Learning with Kernel Methods", slides from MSc Mathematics, Vision, and Machine Learning (MVA) and MSc Mathematics, Machine Learning, and the Humanities (MASH), Université Paris-Saclay, Spring 2022. [Online]. Available: slides.pdf Course website: Machine learning with kernel methods