SVM-Kernel PCA and LDA-Clustering

Kaltsidis Michail

Aristotle University of Thessaloniki

January 21, 2023

Summary

Introduction

Mnist and **cifar10** data set processing using algorithms for classification, dimension reduction and clustering according to the classes of the dataset

Support Vector Machine

Support Vector Classification

Classification PRIMAL problem

The primal problem that this method is going to solve by find minimum of

$$\mathcal{L} = \frac{1}{2}w^T w - \sum_{i=1}^{M} \alpha_k \{y_i(w^T x_i + b) - 1\}$$

Algorithm results

The results have to be +1 or -1 from the:

$$y(x) = sign(w^T x + b)$$

or

$$y(x) = sign(w^T \phi(x) + b)$$

if Kernel trick is going to be used.

Support Vector Classification

Classification DUAL problem

The equivalent dual problem that this method is going to solve by find maximum of

$$\mathcal{DL} = \sum_{k=1}^{N} \alpha_k - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j K(x_i, x_j) \alpha_i \alpha_j, 0 \leq \alpha_i \leq \forall i, \sum_i \alpha_i y_i = 0$$

Algorithm results

The results have to be +1 or -1 from the:

$$y(x) = sign(\sum_i \alpha_i y_i x + b) \textbf{ or } y(x) = sign(\sum_i \alpha_i y_i K(x_i, x) + b)$$

if Kernel trick is going to be used.

Support Vector Regression

Looking for line-hyperplane $\mathbf{y} = w^T x_i + b$ so that $|y_i - w^T x_i - b| \leq \epsilon, \forall i$

Regression problem

Finding the minimum of the function

$$\frac{1}{2} w^T w + c \sum_i (\xi_i + \xi_i^*),$$

$$y_i - w^T \phi(x_i) - b \leq \epsilon + \xi_i, w^T \phi(x_i) + b - y_i \leq \epsilon + \xi_i^*, \xi_i, \xi_i^* \geq 0$$

Parameters in python's function **SVC()** from the **sklearn** library

- *kernel*
- *C*
- *gamma*
- *degree*
- *decision_function_shape*

and more...

metrics that measured

1. Accuracy: It is the metric that calculates the number of correct predictions of the algorithm to the total number of test data.
2. Recall: Is the division of TP to TP+FN.
3. Precision: Is the division of TP to TP+FP.
4. f1: Is the division (2*precision * recall) / (precision + recall).

MNIST IMAGE DATASET

MNIST data results after after implementing the SVC python function

After taking a **sample of 12000** from 60000 training data and **2000** from 10000 test data and after a Principal Component Analysis/**PCA** to the dataset and projected the data to 100 dimensions the results using S.V. Classification will be:

SVM Algorithm(SVC-non linear)	time of training (seconds)	success rates in test stages	success rates in the training stages	f1	acccuracy	recall	precision
kernel="linear", C=100, gamma= 0.01	72	0.98	0.9065	0.906	0.906	0.906	0.906
kernel="linear", C=1, gamma= 0.01	5	0.967	0.9175	0.917	0.917	0.916	0.918
kernel="linear", C=0.01, gamma= 0.0001	2.5	0.93275	0.93	0.93	0.928	0.93	0.929
kernel="linear", C=0.01, gamma= 1	3	same	same	same	same	same	same
kernel="rbf", C=100, gamma=0.01	3.5	0.97	1	0.968	0.969	0.967	0.969
kernel="rbf", C=1, gamma=0.01	4	0.961	0.976	0.96	0.961	0.96	0.96
kernel="rbf", C=0.01, gamma=0.01	99	0.11	0.15	0.11	0.12	0.11	0.11
kernel="rbf", C=200, gamma=0.00001	2	0.93	0.95	0.935	0.934	0.935	9.034
kernel="rbf", C=30, gamma=0.01	2	0.96	1	0.97	0.98	0.97	0.978
kernel="sigmoid", C=30, gamma=0.0001	9	0.905	0.905	0.905	0.904	0.9054	0.905
kernel="sigmoid", C=1, gamma=0.1	7	0.47	0.46	0.47	0.47	0.47	0.49
kernel="sigmoid", C=100, gamma=0.5	9	0.31	0.31	bad	bad	bad	bad
kernel="sigmoid", C=1000, gamma=0.00001	4	0.93	0.94	0.938	0.938	0.937	0.938
kernel="poly", C=10, gamma=0.01	5.2	0.96	0.99	0.966	0.966	0.966	0.966
kernel="poly", C=100, gamma=0.01	7.1	0.97	1	0.978	0.977	0.98	0.977
kernel="poly", C=1, gamma=0.001,degree=5	23	0.11	0.11	bad	bad	bad	bad
kernel="poly", C=1000, gamma=0.001	10.2	0.93	0.96	0.94	0.94	0.94	0.94

Figure: MNIST after PCA

Here some results after using all the dataset (**60000 training images** and **10000 test images**)

	SVM Algorithm(SVC-non linear)	time of training (seconds)	success rates in the training stages	success rates in test stages	f1	acccuracy	recall	precision	# of wrong predictions
1	SVC(kernel="rbf",C=10,gamma=0.1)	359	1	0.9762	0.9762461	0.9762	0.9760223	0.9768359	240
2	SVC(kernel="rbf",C=100,gamma=0.01)	45	1	0.984	0.98391997	0.984	0.983861346	0.983992	160
3	SVC(kernel="poly",C=100,gamma=0.01,degree=3)	75	0.99	0.9837	0.9835612	0.9837	0.98354	0.9836047	163
4	SVC(kernel="poly",C=5,gamma=1,degree=10)	897	1	0.895	0.903268	0.895	0.89381	0.93165	1050
5	SVC(kernel="poly",degree=10,C=100,gamma=0.01)	851	0.543	0.5037	0.5431433	0.5037	0.49332	0.9137	4963
6	SVC(kernel="poly",degree=3,C=1000,gamma=0.001)	195	0.97573	0.9714	0.9712483	0.9714	0.97138	0.971275	286
7	SVC(kernel="poly",degree=2,C=1,gamma=0.01	102	0.981133	0.9773	0.9772614	0.9773	0.9773	0.9772535	227
8	SVC(kernel="poly",degree=2,C=0.5,gamma=0.1)	50	0.999	0.9833	0.983216	0.9833	0.983132	0.9833262	167
9	SVC(kernel="sigmoid",C=100,gamma=0.01)	52	0.876	0.8815	0.88	0.8815	0.88	0.88	1185
10	SVC(kernel="poly",C=10000,gamma=0.01,degree=7)	323	0.999	0.9628	0.96352	0.9628	0.962461	0.9677	372
11	SVC(kernel="rbf",C=1)-default	71	0.9927	0.9844	0.984347	0.9844	0.98433	0.98438	156

Figure: MNIST after PCA

Confusion Matrix

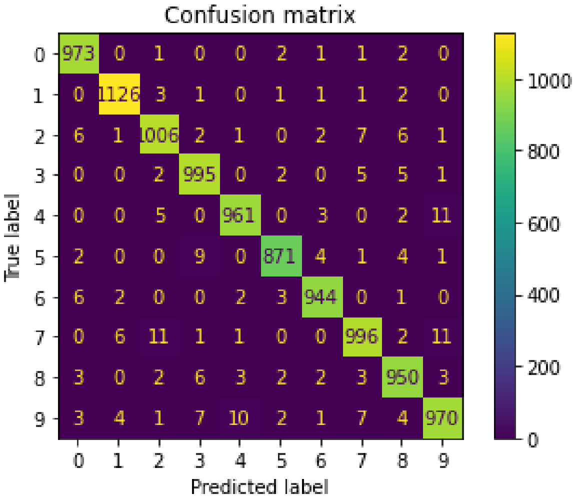


Figure: SVC(C=10,gamma=0.1)

Example

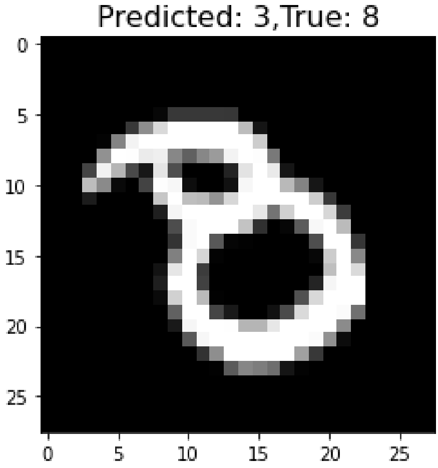


Figure: Example of non correct prediction

CIFAR10 IMAGE DATASET

cifar10

SVM Algorithm(SVC-non linear)	time of training (seconds)	success rates in test stages	success rates in the training stages	f1	acccuracy	recall	precision
decision_function_shape="ovo",C=0.01	24	0.22	0.24	0.14	0.22	0.22	0.18
decision_function_shape="ovr",C=100,kernel="rbf",gamma=0.01	19	0.47	1	0.47	0.47	0.47	0.47
decision_function_shape="ovo",C=100,kernel="rbf", gamma=0.01	19	0.47	1	0.47	0.47	0.47	0.47
decision_function_shape="ovo",C=1,kernel="rbf",gamma=0.0001	16	0.34	0.35	0.33	0.34	0.34	0.33
decision_function_shape="ovo",C=100,kernel="sigmoid",gamma=0.0001	12	0.39	0.44	0.38	0.39	0.39	0.39
decision_function_shape="ovo",C=1000,kernel="sigmoid",gamma=0.1	12	0.15	0.14	bad	bad	bad	bad
decision_function_shape="ovo",C=100,kernel="poly",gamma=0.01	21	0.38	0.99	0.35	0.36	0.35	0.35
decision_function_shape="ovo",C=0.01,kernel="rbf", gamma=0.01	20	0.18	0.18	0.11	0.18	0.18	0.12
C=1,gamma=0.01	11	0.47	0.70	0.47	0.47	0.47	0.47
decision_function_shape="ovo",C=0.01,kernel="linear"	15	0.39	0.43	0.38	0.39	0.39	0.38

Figure: cifar10 after PCA

cifar10 results

Here some results after using all the dataset (**60000 training images** and **10000 test images**)

SVM Algorithm(SVC-non linear)	time of training (seconds)	success rates in the training stages	success rates in test stages	f1	acccuracy	recall	precision	# of wrong predictions
SVC(decision_function_shape="ovo",C=0.1)	319	0.4768	0.46	0.457	0.46	0.46	0.46	5393
SVC(kernel="rbf",C=100,gamma=0.01)	908	0.99	0.554	0.555	0.554	0.554	0.558	4459
SVC(decision_function_shape="ovo")	224	0.6546	0.541	0.54	0.541	0.5409	0.54	4591
SVC(kernel="sigmoid",C=100,gamma=0.01)	321	0.1953	0.1986	0.1785	0.1986	0.1986	0.24	8014
SVC(kernel="sigmoid",decision_function_shape="ovo")	204	0.224	0.2243	0.21	0.2243	0.2243	0.257	7757
SVC()	576	0.66202	0.54	0.5383	0.54	0.54	0.539	4601

Figure: cifar10 after PCA

cifar10 with **LinearSVC()**

SVM Algorithm(SVC-non linear)	time of training (seconds)	success rates in the training stages	success rates in test stages	f1	acccuracy	recall	precision
LinearSVC(dual=False,C=0,1)	34	0.40	0.40	0.38	0.40	0.40	0.385
LinearSVC(dual=False,C=0.1,penalty="l1"	28	0.40	0.40	0.39	0.40	0.389	0.356
LinearSVC(C=1)	508	0.4012	0.3891	0.3773	0.3891	0.3891	0.38
LinearSVC(C=1000)	523	0.2455	0.2324	0.2161	0.2324	0.2324	0.2265

Figure: cifar10

k-Nearest Neighbors and Nearest Class Centroid

k-NN and NCC for MNIST

k-NN Algorithm	time of training (seconds)	success rates in test stages	success rates in the training stages	f1	acccuracy	recall	precision
KNeighborsClassifier(n_neighbors=8)	0.06	0.95	0.96	0.95	0.95	0.96	0.95
KNeighborsClassifier(n_neighbors=20)	0.03	0.95	0.94	0.95	0.95	0.94	0.95
KNeighborsClassifier(n_neighbors=2)	0.03	0.95	0.97	0.95	0.95	0.96	0.95
KNeighborsClassifier(n_neighbors=15,p=3)	480	0.95	0.95	0.95	0.95	0.95	0.95
NearestCentroid()	0.002	0.80	0.81	0.80	0.815	0.812	0.81
NearestCentroid(metric='minkowski')	0.01	0.80	0.80	0.80	0.80	0.80	0.80

Figure: k-NN and Centroid

k-NN and NCC for cifar10

k-NN Algorithm	time of training (seconds)	success rates in test stages	success rates in the training stages	f1	acccuracy	recall	precision
KNeighborsClassifier(n_neighbors=90)	0.06	0.31	0.31	0.31	0.31	0.31	0.31
KNeighborsClassifier(n_neighbors=20)	0.03	0.33	0.37	0.34	0.33	0.33	0.34
KNeighborsClassifier(n_neighbors=2)	0.03	0.30	0.64	0.31	0.31	0.30	0.31
KNeighborsClassifier(n_neighbors=15,p=2)	0.1	0.33	0.4	0.33	0.34	0.33	0.34
NearestCentroid()	0.002	0.29	0.27	0.29	0.29	0.30	0.30
NearestCentroid(metric='minkowski')	0.01	0.29	0.26	0.30	0.30	0.29	0.30

Figure: k-NN and Centroid

KERNEL PCA AND LDA

Kernel Principal Component Analysis

- **Data matrix** : $\mathbf{X} = [\mathbf{x_1}, \ldots, \mathbf{x_N}]$ (X is an D×N),
- The Kernel matrix of the X is $\mathbf{K} = \mathbf{\Phi^T \Phi}$, where $\Phi(X)$ are the data in the Hilbert space and $\Phi(X)$ is a N×N matrix (N are the number of features of data matrix X).
- $\mathbf{K} = \mathbf{W \Lambda W^T}$, where W is the matrix of the eigenvectors of K and Λ is the diagonal matrix of the eigenvalues of K.
- In the new low dimension the data will be given by $\mathbf{Y} = \mathbf{\Lambda^{\frac{1}{2}} W^T}$
- After finding the **mean vector** of $\mathbf{\Phi}$ and transforming this vector to a equivalent vector with mean value 0. This new $\bar{\mathbf{\Phi}}$ have it's Kernel matrix
- Calculating $\bar{\mathbf{K}}(x_i, x_j) = \cdots = (I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)K(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T) = \mathbf{J}K\mathbf{J}^T$

Linear Discriminant Analysis

- Project the data on the direction of vector v, and get the mean of every class of the data. Let it be $\mu_1 = v^T m_1$ and $\mu_2 = v^T m_2$ for two classes.
- The variances of these two classes given as $s_1^2 = \sum_{x_i \in C_1} (\alpha_i - \mu_1)^2$ and $s_2^2 = \sum_{x_i \in C_2} (\alpha_i - \mu_2)^2$
- The function which maximum have to be found is

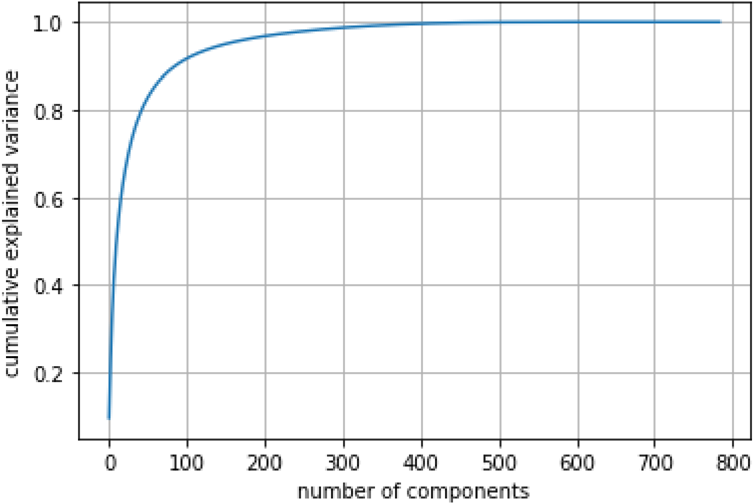
$$\max_{v:||v||=1} \frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2}$$

- The equivalent function that have to maximized is

$$\max_{v:||v||=1} \frac{v^T S_B v}{v^T S_W v}$$

$S_b = \sum_i N_i (\bar{u}_i - \bar{u})(\bar{u}_i - \bar{u})^T$ and $S_W = \sum_i \sum_j (x_i - m_j)(x_i - m_j)^T$

How to select number of components in **KPCA/PCA**



k-NN and Centroid after Kernel PCA and LDA

Για X_train=6000 και X_test=5000

KPCA+LDA+kNN (n=17)	time of training (seconds)	f1	acccuracy	recall	precision
kernel="rbf",gamma=0.00001	37	0.896	0.898	0.896	0.8967
kernel="rbf",gamma=0.1	40	0.7596	0.764	0.7615	0.7972
kernel="linear",gamma=0.00001	35	0.8933	0.8962	0.8935	0.8942
kernel="linear",gamma=0.1	35	0.9022	0.9042	0.9022	0.9032
kernel="sigmoid",gamma=0.00001	37	0.901	0.903	0.901	0.902

KPCA+LDA+Nearest Centroid	time of training (seconds)	f1	acccuracy	recall	precision
kernel="rbf",gamma=0.00001	26	0.8721	0.8736	0.872	0.8741
kernel="poly",gamma=0.001,degree=4	36	0.87	0.87	0.87	0.87

Για X_train=12000 και X_test=5000

KPCA+LDA+kNN (n=17)	time of training (seconds)	f1	acccuracy	recall	precision
kernel="rbf",gamma=0.00001	193	0.9	0.9	0.9	0.9

KPCA+LDA+Nearest Centroid	time of training (seconds)	f1	acccuracy	recall	precision
kernel="rbf",gamma=0.00001	193	0.87	0.87	0.87	0.87
kernel="poly",gamma=0.001,degree=4	190	0.87	0.87	0.87	0.87

Figure: k-NN and Nearest Centroid

k-Nearest Neighbors

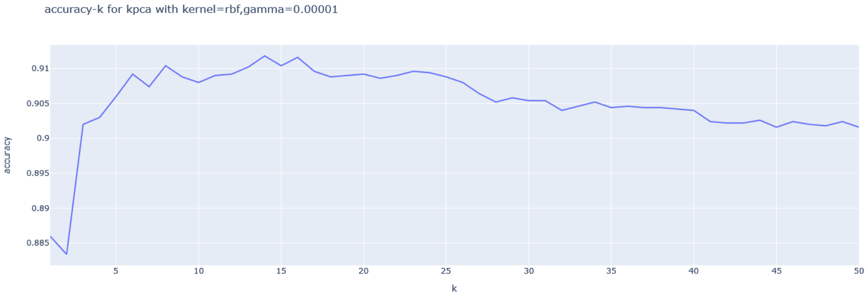


Figure: Graph of accuracy as a function of number of neighbors

PCA+SVM vs KPCA+LDA+SVM (1)

Is there any difference between SVM with prior reduction with **PCA** and SVM with prior reduction with **Kernel PCA** combined with **LDA**?

PCA+SVM vs KPCA+LDA+SVM (2)

LinearDiscriminantAnalysis(n_components=9)

SVM results	SVM without Kernel PCA+LDA	*KPCA(kernel="linear",gamma=0.01)* SVM with Kernel PCA + LDA	*KPCA(kernel="linear",gamma=10)* SVM with Kernel PCA + LDA	*KPCA(kernel="poly",gamma=0.001,degree=2)* SVM with Kernel PCA + LDA	*KPCA(kernel="rbf",gamma=0.001)* SVM with Kernel PCA + LDA
kernel="linear",C=0.00000001,gamma=1	0.1	0.117	0.1125	0.11	0.11
kernel="linear",C=100,gamma=0.01	0.906	0.8885	0.885	0.89	0.91
kernel="linear",C=1,gamma=0.01	0.917	0.8885	0.885	0.89	0.9055
kernel="linear",C=0.01,gamma=0.0001	0.928	0.887	0.8815	0.89	0.903
kernel="linear",C=0.01,gamma=1	0.928	0.887	0.8815	0.888	0.903
kernel="rbf",C=100,gamma=0.01	0.969	0.92	0.9065	0.913	0.9165
kernel="rbf",C=1,gamma=0.01	0.961	0.90	0.89	0.90	0.91
kernel="rbf",C=0.01,gamma=0.01	0.12	0.8725	0.8735	0.8755	0.8875
kernel="rbf",C=200,gamma=0.00001	0.934	0.888	0.8825	0.89	0.903
kernel="rbf",C=30,gamma=0.01	0.98	0.906	0.9035	0.912	0.912
kernel="sigmoid",C=30,gamma=0.0001	0.904	0.8865	0.8825	0.889	0.91
kernel="sigmoid",C=1,gamma=0.1	0.47	0.62	0.642	0.6035	0.6255
kernel="sigmoid",C=100,gamma=0.5	0.30	0.48	0.18	0.19	0.1915
kernel="sigmoid",C=1000,gamma=0.00001	0.938	0.8875	0.8815	0.89	0.903
kernel="poly",C=10,gamma=0.01	0.966	0.872	0.8695	0.879	0.8895
kernel="poly",C=10000,gamma=0.01	0.977	0.92	0.893	0.90	0.8985
kernel="poly",C=1,gamma=0.001,degree=5	0.11	0.11	0.11	0.11	0.11
kernel="poly",C=1000,gamma=0.001	0.94	0.794	0.794	0.792	0.8255

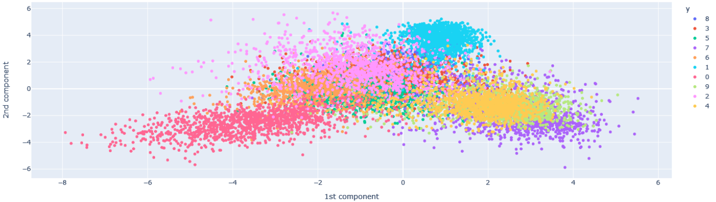
Visualization MNIST and cifar10 data after Kernel PCA and LDA

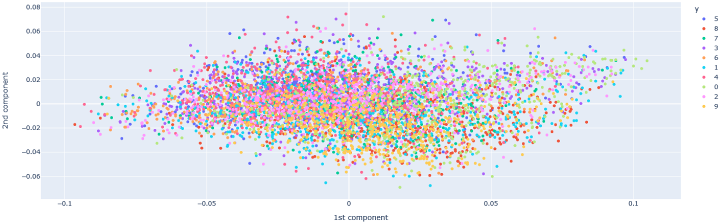
After Kernel Principal Component Analysis the dimension will be 100

After the Linear Discriminant Analysis the dimension of data will be 10-1=9

Scatter Plots

kpca: kernel=rbf,gamma=0.00001 for 12000 train and 5000 test





t-SNE + SPECTRAL CLUSTERING

t-SNE

About t-SNE technique:

- Dimension reduction method
- Used for linearly non-separable data
- Usually the preferred dimensions to transform the data are 2 or 3
- Used for visualize the data

The parameters that are used in function *TSNE()* are:

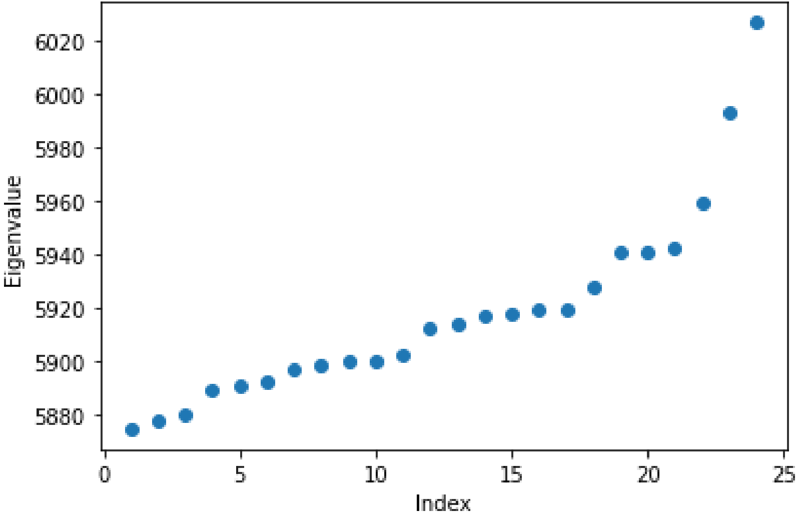
1. n_components
2. perplexity
3. learning_rate

Spectral clustering

After the data has been reduced to 2 dimensions, clustering of the data set is applied using:

- SpectralClustering (scikit learn)
  1. n_clusters
  2. n_components
  3. affinity
  4. assign_labels
- Uses **k-means** algorithm to cluster the data
- Number of clusters from eigenvalues of Laplacian matrix

How to choose the number of clusters to create?

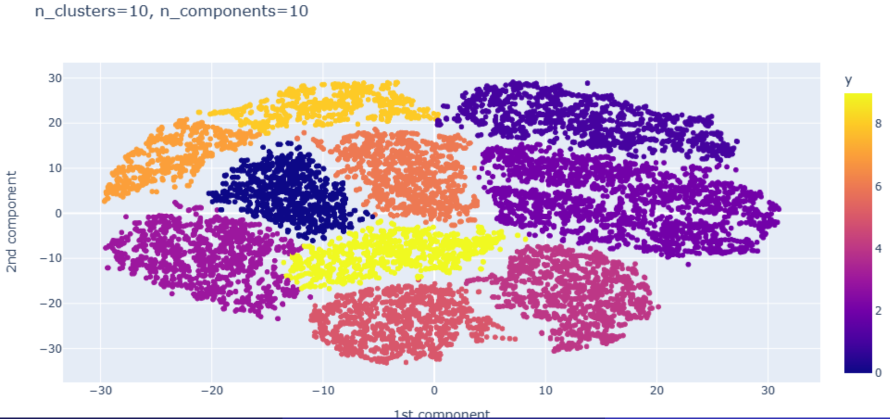


Sihlouette Metric

- k= 6 average silhouette score is : 0.41882777
- k = 7 average silhouette score is : 0.44772887
- k = 8 average silhouette score is : 0.4472915
- k = 9 average silhouette score is : 0.45699435
- k = 10 average silhouette score is : 0.447488
- k = 11 average silhouette score is : 0.4379061
- k = 12 average silhouette score is : 0.44494075
- k = 13 average silhouette score is : 0.43937606

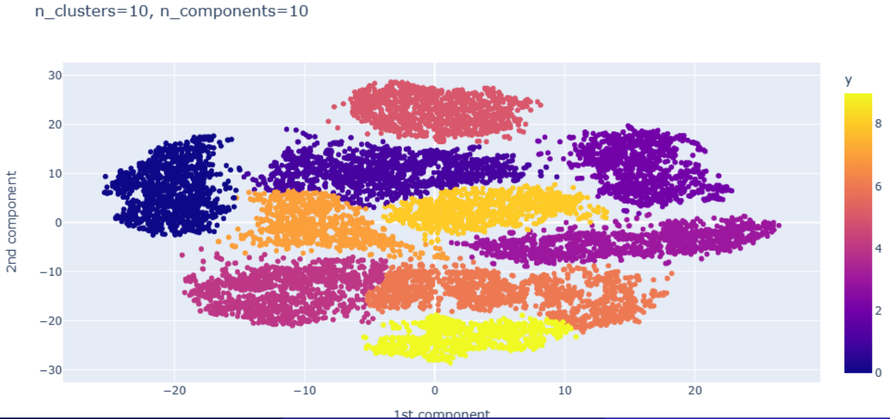
Clustering on **MNIST** data using rbf kernel for computing distances (1)

-TSNE(n_components=2, perplexity=15,learning_rate=100,n_iter=500)
-SpectralClustering(n_clusters=10, n_components=10)



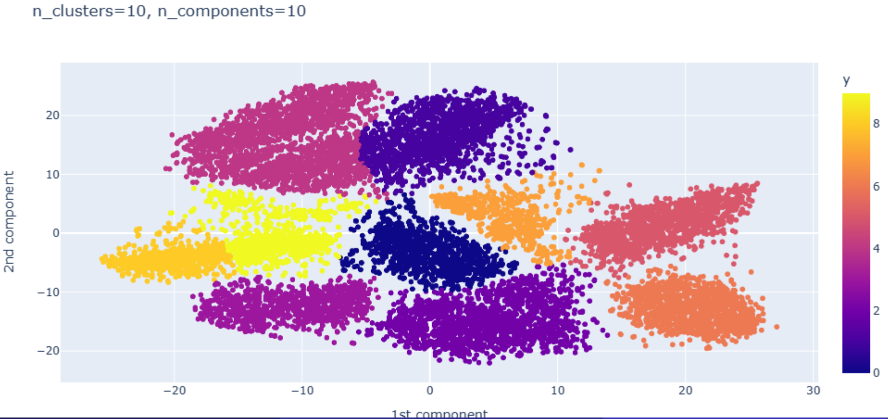
Clustering on **MNIST** data using rbf kernel for computing distances (2)

-TSNE(n_components=2, perplexity=50,learning_rate=100,n_iter=500)
-SpectralClustering(n_clusters=10, n_components=10)



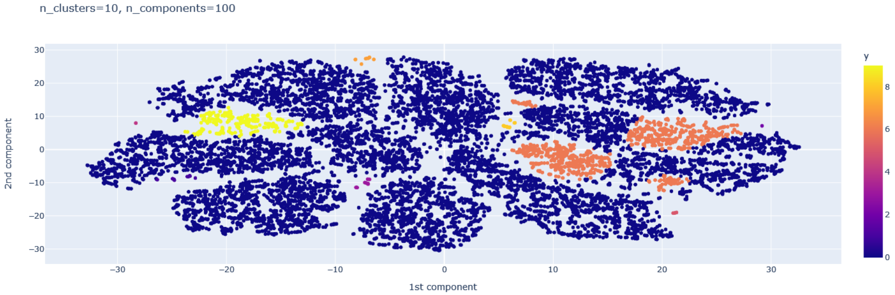
Clustering on **MNIST** data using rbf kernel for computing distances (3)

-TSNE(n_components=2, perplexity=300,learning_rate=100,n_iter=500)
-SpectralClustering(n_clusters=10, n_components=10)



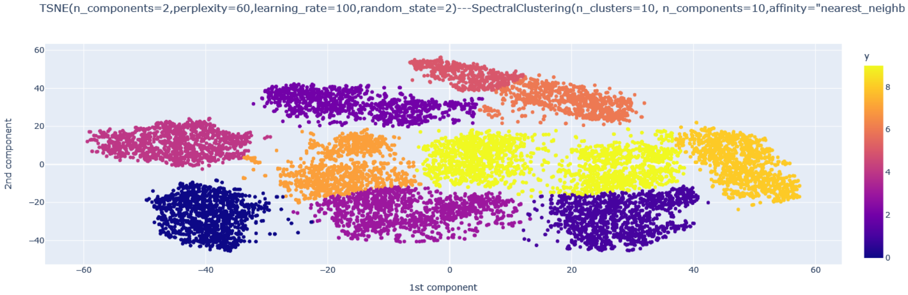
Clustering on **MNIST** data using rbf kernel for computing distances (4)

-TSNE(n_components=2, perplexity=15,learning_rate=100,n_iter=500)
-SpectralClustering(n_clusters=10, n_components=100)



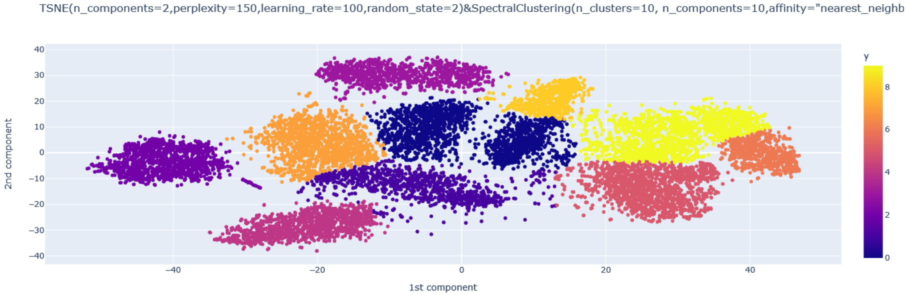
Clustering on **MNIST** data by computing a graph of nearest neighbors (1)

-TSNE(n_components=2, perplexity=60,learning_rate=100,random_state=2)
-SpectralClustering(n_clusters=10, n_components=10,affinity='nearest_neighbors')



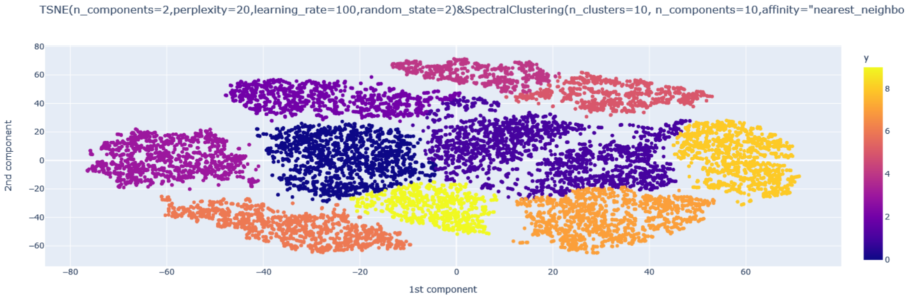
Clustering on **MNIST** data by computing a graph of nearest neighbors (2)

-TSNE(n_components=2, perplexity=150,learning_rate=100,random_state=2)
-SpectralClustering(n_clusters=10, n_components=10,affinity='nearest_neighbors')



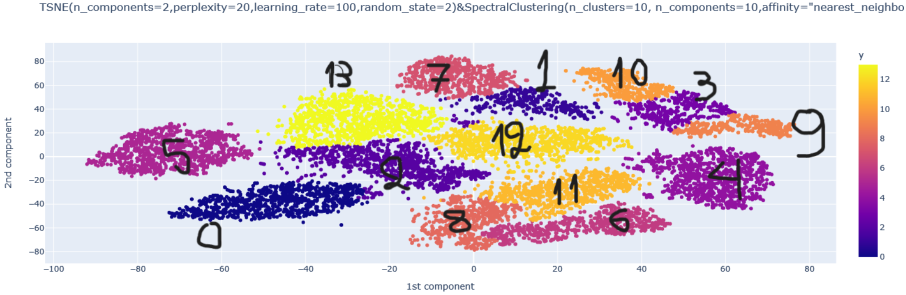
Clustering on **MNIST** data by computing a graph of nearest neighbors (3)

-TSNE(n_components=2, perplexity=20,learning_rate=100,random_state=2)
-SpectralClustering(n_clusters=10, n_components=10,affinity='nearest_neighbors' ,n_neighbors=15)



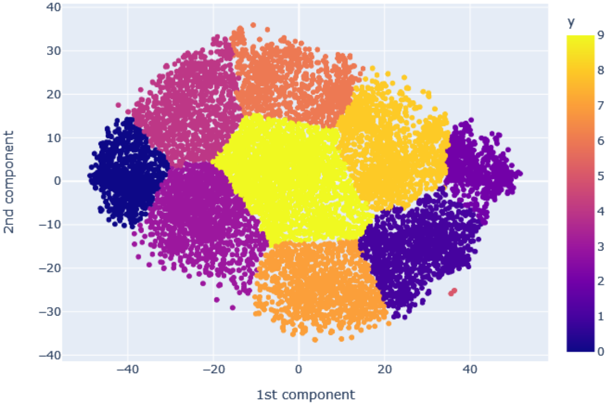
Clustering in 14>10 clusters

-TSNE(n_components=2,random_state=2)
-SpectralClustering(n_clusters=14, n_components=10,affinity='nearest_neighbors')



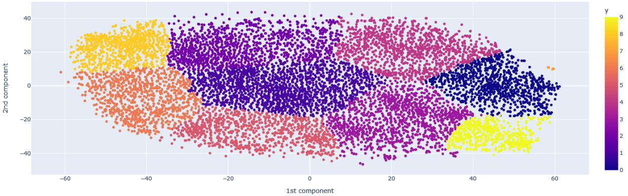
Clustering on **cifar10** data

-TSNE(n_components=2, perplexity=30,learning_rate=100, random_state=2)
-SpectralClustering(n_clusters=10, n_components=10, affinity='nearest_neighbors')



Clustering on **cifar10** data

-TSNE(n_components=2, perplexity=30,learning_rate=800, random_state=2)
-SpectralClustering(n_clusters=10, n_components=10, affinity='nearest_neighbors')



Thank you very much for your attention!