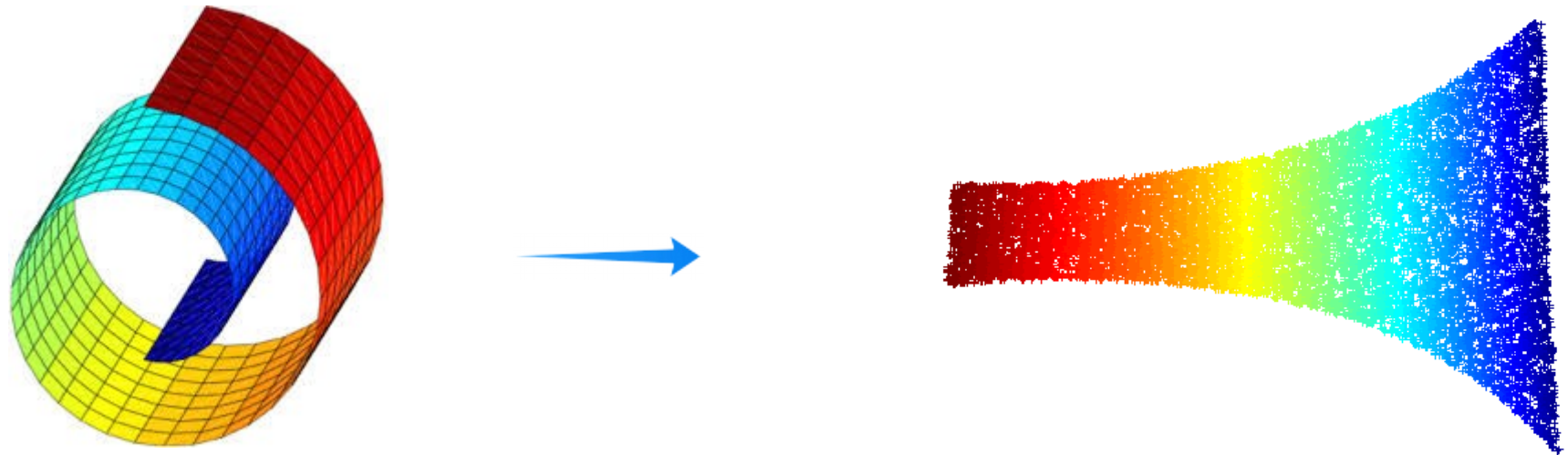


Locally Linear Embeddings in Pattern Recognition



- Elaboration: Katsileros Petros
- Supervision: Nikolaos Pitsianis

Introduction

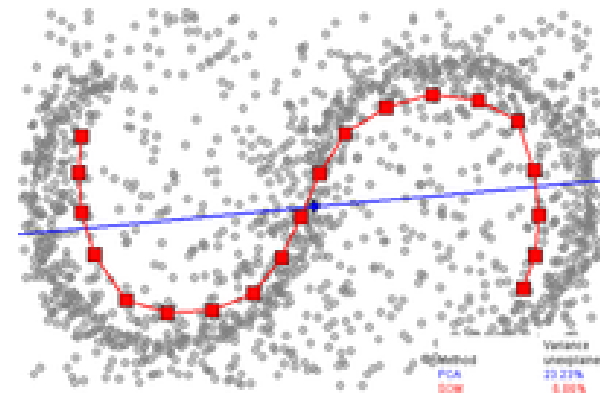
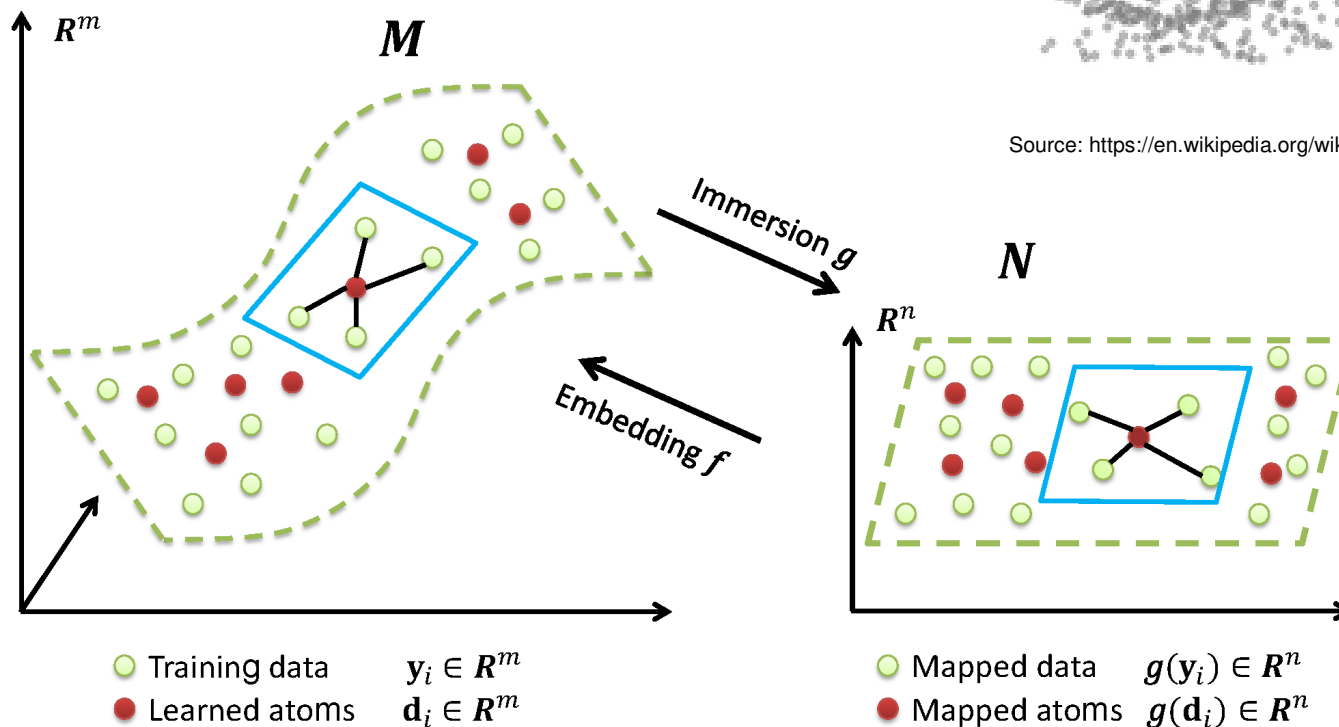
- Why we need information compression
- Dimensionality Reduction Algorithms (PCA, SVD, ISOMAP, Laplassian Eigenmaps, LLE)
- The Locally Linear Embeddings Algorithm (LLE)
- **Surpass LLE limitations with two new variations of LLE algorithm**
- Experiments using native LLE and the two new methods
 - Datasets: MNIST, SVHN, Arcene
- **Results analysis**
- Future work
- Conclusion

Dimensionality Reduction Algorithms

- Linear:
 - **Principal Component Analysis (PCA)**
 - Metric Multi-dimensional Scaling (MDS)
 - **Singular Value Decomposition (SVD)**

Dimensionality Reduction Algorithms

- Non-Linear:
 - Isometric Mapping (ISOMAP)
 - Laplacian-Eigenmaps
 - **Locally Linear Embeddings**



Source: <https://www.eecis.udel.edu/~zhou/Research.html>

Locally Linear Embeddings

- Step-1: Find K-Nearest Neighbors for each data (Adjacency matrix)
 - With CUDA
- Step-2: Linear prediction for every point using it's neighbors. Find Weight matrix (W).
 - Minimize cost function: $\arg \min E_w = \sum_{i=1}^N \|X_i - \sum_{j=1}^N W(i, j) X_j\|^2$
 - Weights W must follow these two restrictions:
 - $W(i, j) = 0$, if j and i are not neighbors
 - $\sum_{j=1}^N W(i, j) = 1$
 - Following these restrictions we have Rotation, Scaling and Translation independence

Locally Linear Embeddings

- Step-3: Find the embedded coordinates, using matrix W

- Minimize cost function: $\arg \min E_y = \sum_{i=1}^N \|Y_i - \sum_{j=1}^N W(i, j) Y_j\|^2$

- $E_y = \|(I - W)Y\|^2 = Y^T M Y$

- Find eigen-values of the square [NxN] **sparse** matrix

- $M = (I - W)^T (I - W)$

- Step-4: Keep the final embedded coordinates

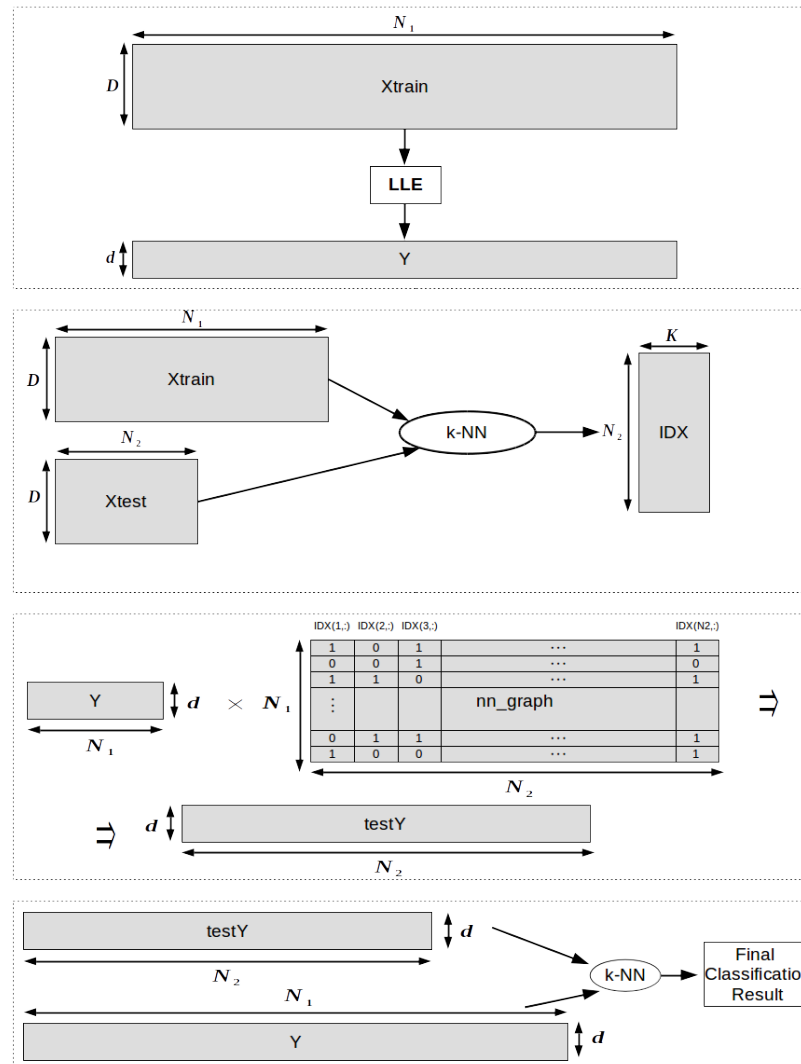
- Discard λ_0 eigenvalues (equal to zero)

- Keep the rest d eigenvectors. These are the final d embedded coordinates

LLE Limitations

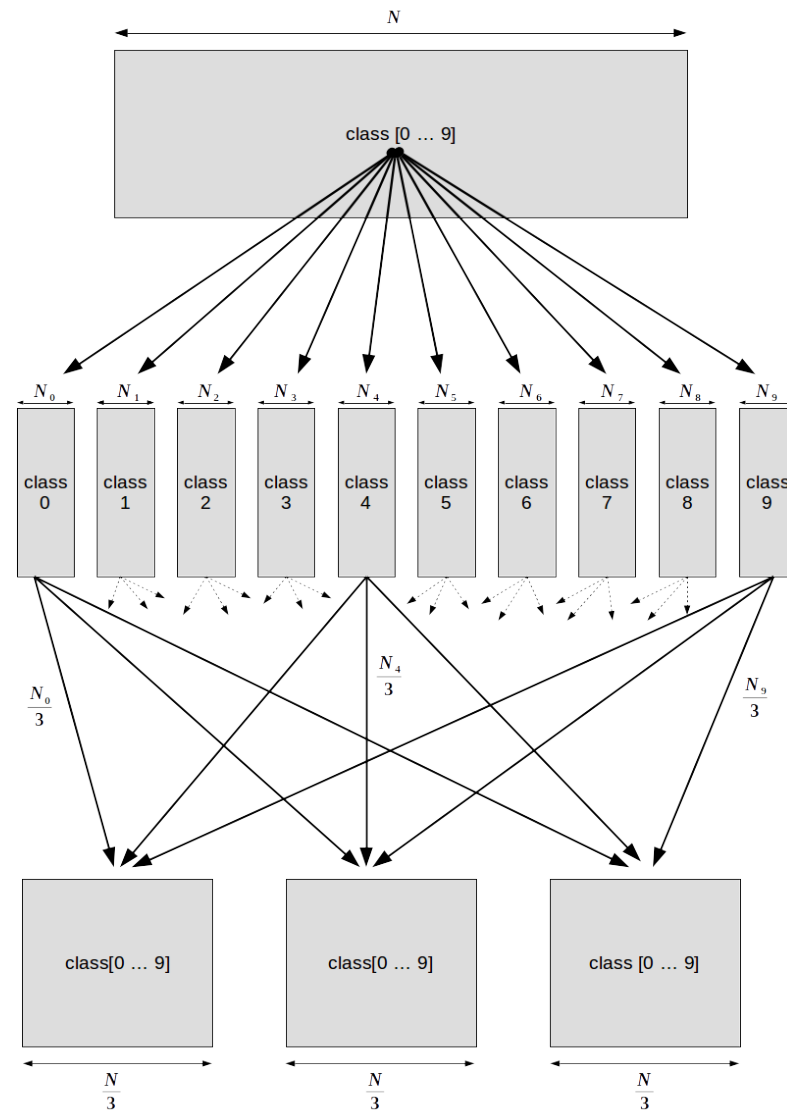
- Eigen-value decomposition step has complexity $d \cdot O(N^2)$
- We must run LLE algorithm with both train and test datasets as input data
- **Solutions:**
 - Method-1 will solve 2nd limitation
 - Execute LLE on Train+Test data
 - Method-2 will solve 1st limitation
 - Runtime algorithm's complexity

Method-1: LLE with Test-Projection



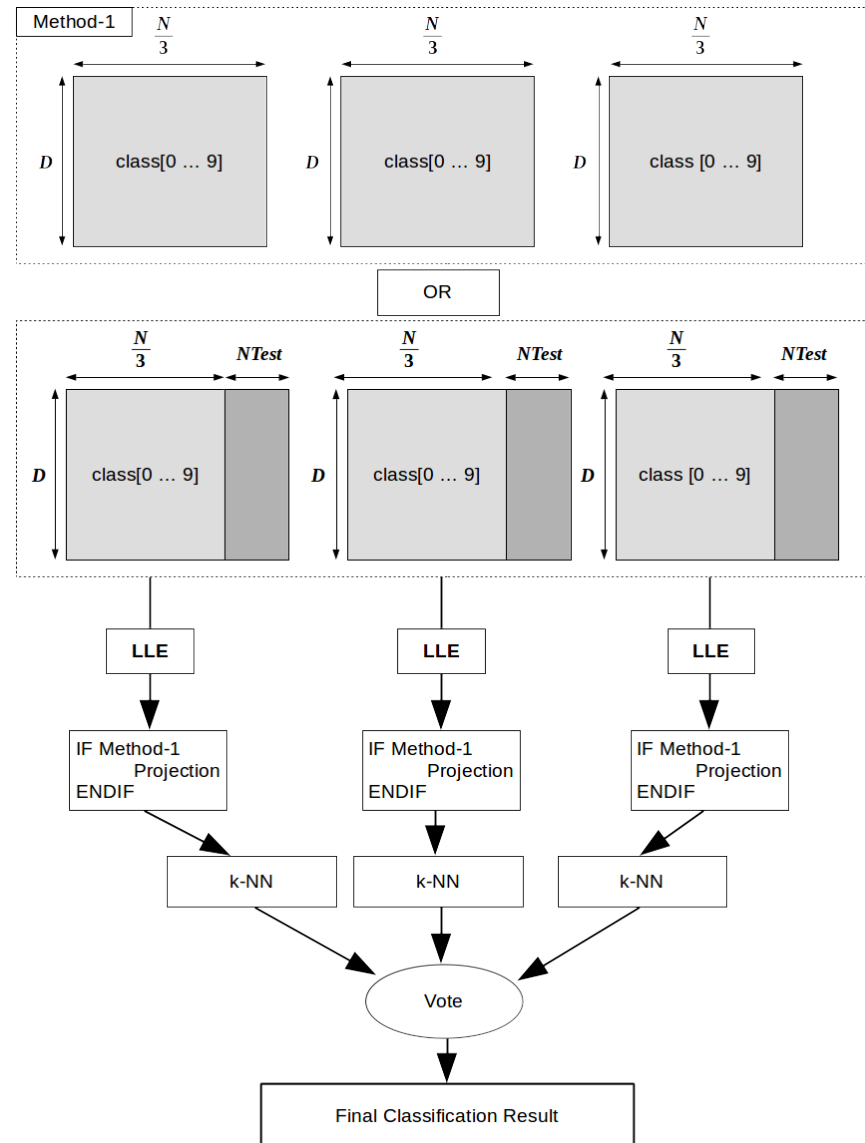
Method-2: Sub-set Majority Voting

LLE



Method-2: Sub-set Majority Voting

LLE



Experiments

- Datasets: MNIST, SVHN, ARCENE
 - MNIST: Gray scale images of handwritten digits
 - 60K Train data, 10K Test data, $D=[28 \times 28]$
 - SVHN: Google street view house numbers RGB images
 - 73257 Train data, 26032 Test data, $D=[32 \times 32]$
 - Arcene: Cancer dataset with a large number of predefined features for each patient.
 - 200 Train data, 700 Test data, $D = 10K$



Source: <http://pavel.surmenok.com/2014/07/06/neural-networks-training-using-encog/>

- Classification algorithm: k-NearestNeighbors
- Classification metric: Mean average % error
 - Sum Accuracies from every class / number of classes



Source: <http://ufldl.stanford.edu/housenumbers/>

MNIST Experiments

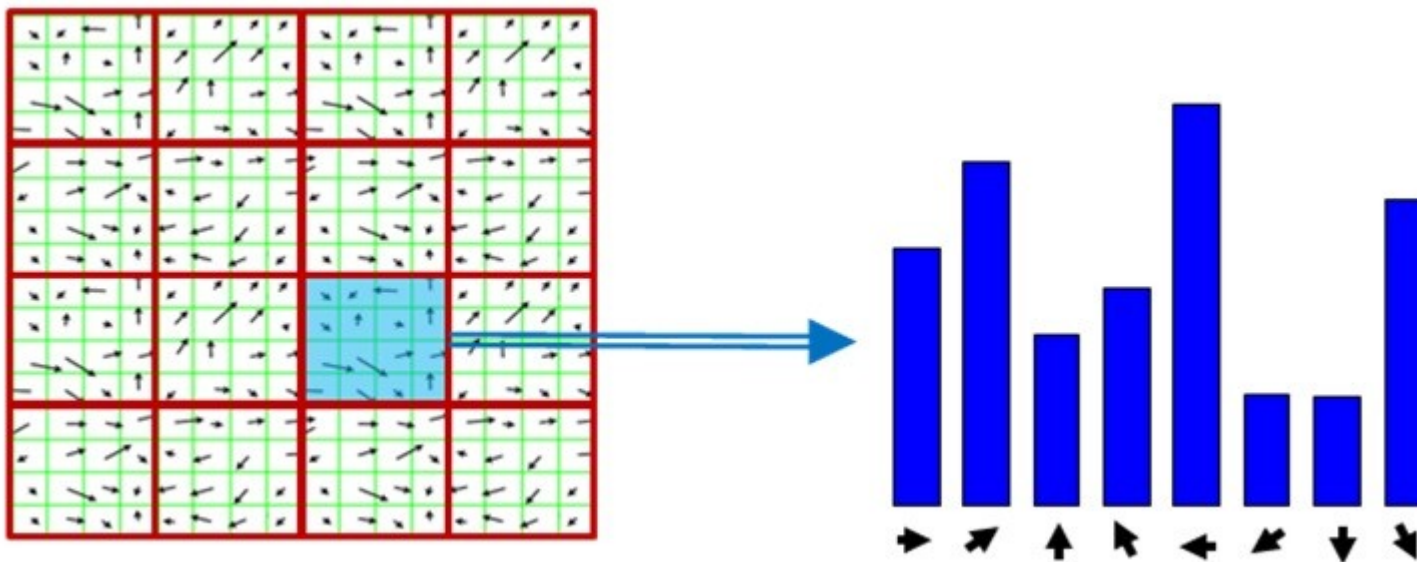
- Invest how LLE parameters (k, d) affect the classification process. Also find out if sub-sampling (Method-2) can lead to acceptable results
- $K = [6, 7, 8, 9, 10, 12, 16, 20, 24, 32, 64]$, $d = [10, 16, 20, 24, 32, 40, 52, 64, 96, 128, 256]$, subSet_size=[60000,20000,10000] of Train dataset
 - Classification error using k-NN (k=2) without dimensionality reduction (D=784): **3.5%**
 - Classification error using k-NN (k=2): **K=12, d=128, subSet_size=60K**, equal to **3.06%**
 - Classification error using k-NN (k=2): **K=8, d=10, subSet_size=60K** equal to **3.31%**
- LLE algorithm can be used as a feature extraction process with a great data compression ability

MNIST Experiments

- Method-2 results:
 - K=16, d=256, batch_size=20K. Best classification error equal to **3.27%**
 - K=10, d=128, batch_size=10K. Best classification error equal to **3.31%**
 - **Huge reduction both in time and space.** (Step-3 of LLE has $O(N^2)$ complexity)
- Method-1: Use of LLE dimensionality reduction in “Real time” applications
 - K=8, d=256, batch_size=60K. Best classification error equal to 3.85%

SVHN Experiments

- Extract HoG features due to noise and light distortions
 - Split image into sub sections and calculate the gradient of pixel intensity



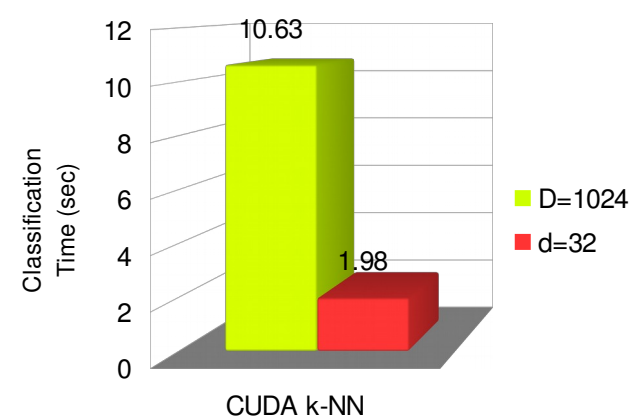
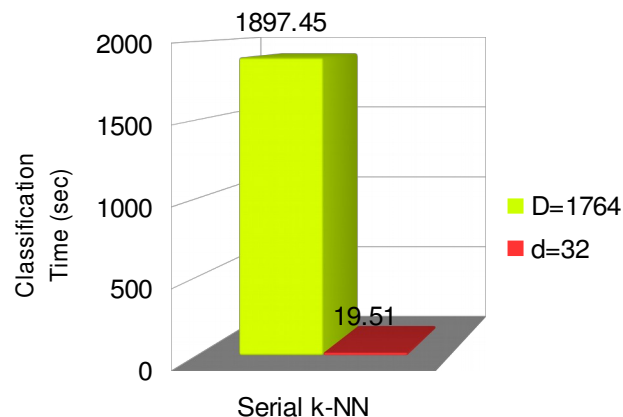
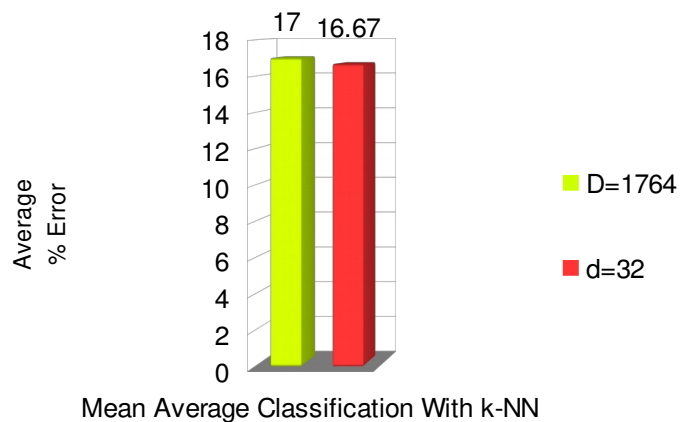
Source: <https://www.quora.com/What-is-a-histogram-of-gradient-directions-in-computer-vision>

SVHN Experiments

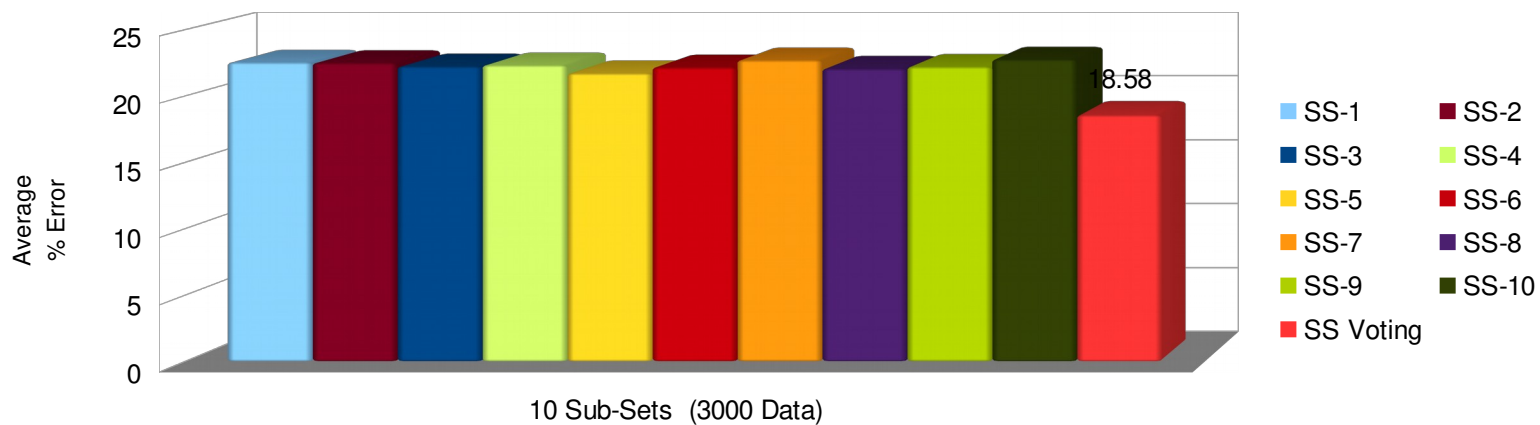
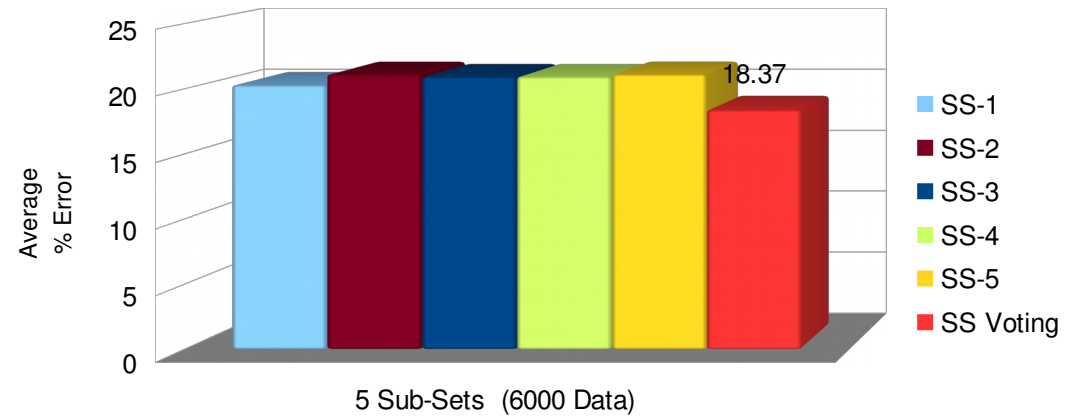
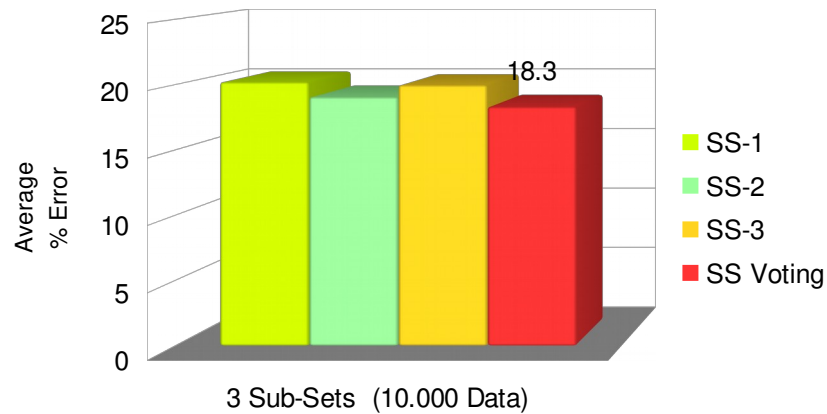
- **1st-Exp: Investigate LLE behaviour to this dataset. Also find out the best HoG kernel size**
 - $K = [8, 10, 12]$, $d = [16, 20, 32, 64, 96, 128, 164, 196, 256]$, 30K of SVHN Train data-set
 - HoG Kernel size: $[2 \times 2]$, $[4 \times 4]$, $[8 \times 8]$ produce features of length: 8100, 1764, 324
 - **Best parameters for SVHN dataset: $K=12$, $d=32$, kernel= $[4 \times 4]$**
- **2nd -Exp Method-1: Find out if Method-1 can lead to acceptable results**
 - $K=12$, $d=32$, kernel= $[4 \times 4]$. 42K of SVHN Train dataset
 - Classification error with LLE dimensionality reduction on Train data and projection for Test data is equal to **18.34%**
 - Classification error without dimensionality reduction is equal to **18.07%**

SVHN Full Dataset Experiment

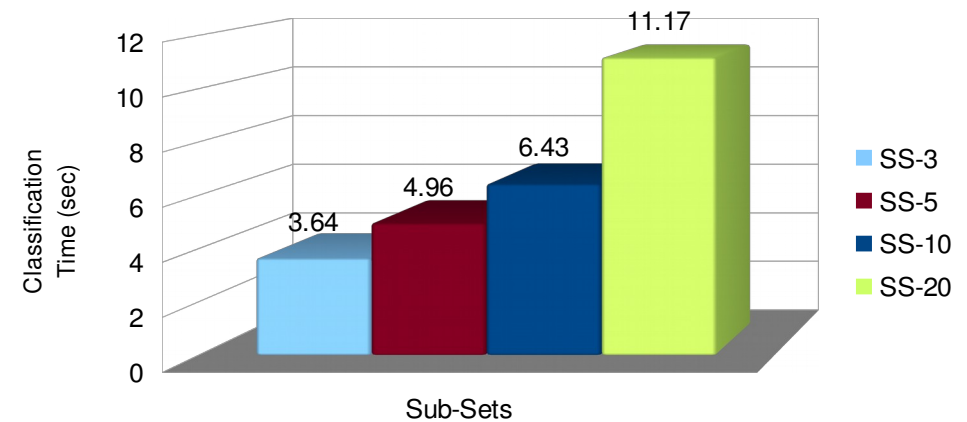
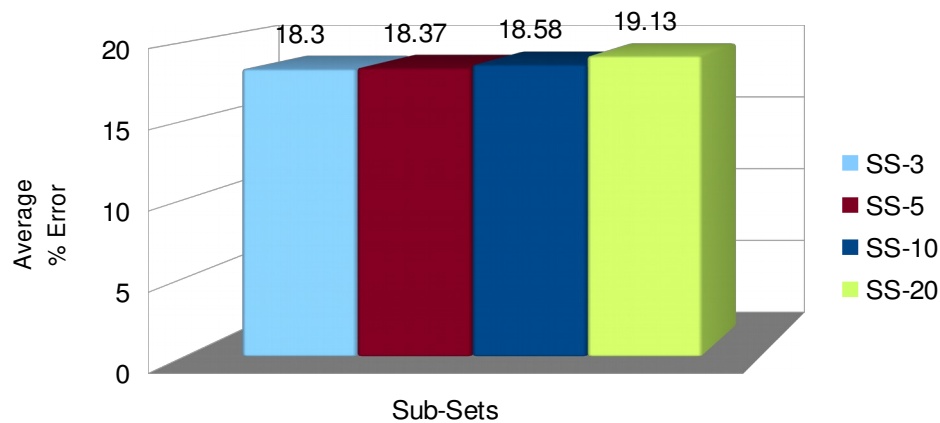
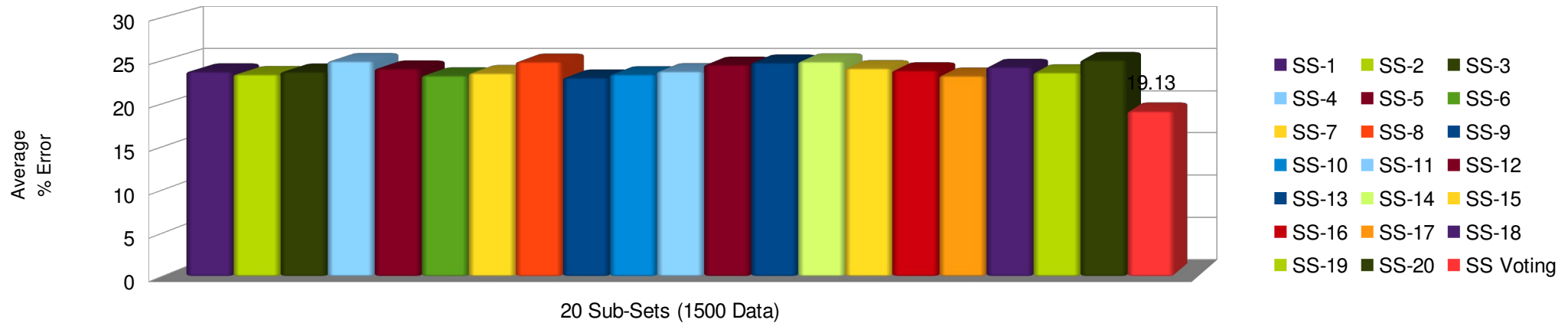
- **3rd-Exp: Classification result with vs without LLE dimensionality reduction**
 - K=12, **d=32**, kernel=[4x4]. Full SVHN dataset (73257 Train data, 26032 Test data)



SVHN Method-2 Experiments



SVHN Method-2 Experiments



ARCENE Experiments

- Investigate LLE dimensionality reduction as feature selection/extraction algorithm.

	K=10	K=12	K=16	K=20	K=24	K=32	K=64	Classification Time/Sample
d=10	10	12	18	18	20	18	18	2.4 ms
d=16	14	22	16	22	18	18	18	2.4 ms
d=20	16	18	16	16	22	16	16	2.5 ms
d=24	14	14	18	20	18	18	16	2.7 ms
d=32	14	20	24	18	20	18	18	2.7 ms
d=40	16	14	14	16	16	22	18	2.7 ms
d=52	10	16	14	14	16	16	14	2.7 ms
d=64	14	16	22	20	22	18	14	2.8 ms
d=96	22	22	22	18	12	16	22	2.9 ms
d=128	24	10	26	14	28	28	26	3.0 ms
D=10.000	24							51.7 ms

Classification Accuracy Error using k-NN Algorithm

Future Work

- Parallelization: Even though k-NN algorithm both in LLE and in Classification process is executed in CUDA, further parallelization can be achieved especially in Method-2
- Extend to large datasets and invest the behaviour of subspace average voting
- Find the best fit between size of Train dataset subsets and Test data

Conclusion

- LLE produces very accurate results after huge dimensionality reduction
- LLE algorithm can be used both as feature extraction and feature selection algorithm
- LLE has the ability to remove noise-data, producing very accurate classification results
- Huge space and time savings at Final Classification Step ($d \ll D$)
- Method-1 can lead to real time Classification results. Using LLE on Train Data, Classification process can be executed for low dimensional data
- Method-2 achieves huge reduction both in space and time, without or with minimal information loss
- Method-1 and Method-2 make the execution of LLE algorithm available for normal PCs

Thank You

katsiler@auth.gr