ITC05F Machine Learning Project 1

Group G-02
Team members:

1. Hoang Tuan Linh	m5232108	(*)
2. Tran Thi Thoa	s1242006	(**)

Datasets: Iris flower, Handwritten digits

Algorithms:

Nearest neighbor classifier (*)

Learning vector quantization (*)

Maximum likelihood + Gaussian distribution (**)

Maximum a posterior + Gaussian Kernel-based PDF estimation (**)

Contents

- Iris and handwritten digits datasets
- Algorithms:
  - Nearest neighbor classifier (1NN)
  - Learning vector quantization (LVQ)
  - Maximum likelihood + Gaussian distribution (ML)
  - MAP + Gaussian Kernel-based PDF estimation (MAP)
- Numerical results/Python3
  - Accuracy (Mean + Variance)
  - Training + Testing time
  - Confusion matrix
- Conclusions

Iris dataset

(from UCI Machine Learning Repository**)**

Iris Versicolor **Iris Setosa** **Iris Virginica**

Image credit: pngkey.com

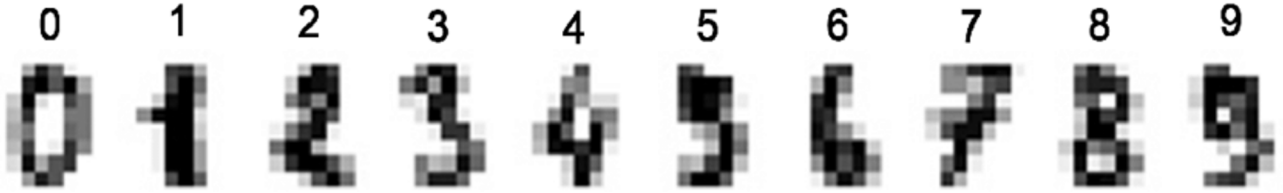
- No. of classes: **3**
- No. of datapoints: **150** # 50 for each of three classes
- No. of attributes: **4** # real numbers, in cm

		Min	Max	Mean	SD	Class Correlation
1	sepal length:	4.3	7.9	5.84	0.83	0.7826
2	sepal width:	2.0	4.4	3.05	0.43	-0.4194
3	petal length:	1.0	6.9	3.76	1.76	0.9490
4	petal width:	0.1	2.5	1.20	0.76	0.9565

- For testing: **train size = 75, test size = 75 (randomly)**

Handwritten digits dataset

(from UCI Machine Learning Repository**)**



- Labels: **[0 1 2 3 4 5 6 7 8 9]**
- Train size: **3823** , test size: **1797**
  - ~380 examples for each digit in training set
  - ~180 examples for each digit in testing set
- No. of attributes:
  - **64** input attributes (integers in the range 0->16)
  - 1 class attribute (integers in the range 0->9)
- 32x32 bitmaps divided into nonoverlapping blocks of 4x4 -> input matrix of 8x8 -> input array of 64 attributes

Nearest neighbor classifier (1NN)

K-nearest neighbor classifier (KNN):

- If k=3 (solid line circle): the green dot is assigned to the red triangles
- If k=5 (dashed line circle): the green dot is assigned to the blue squares

For k =1 : Nearest neighbor classifier

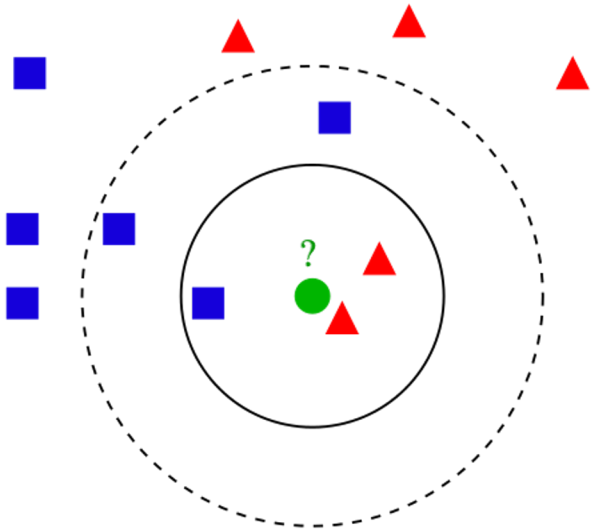
label (x) = label (p) if

$$p = \arg\min_{q \in \Omega} ||x - q||_2$$

x: a new datapoint

p: a datapoint in the training set (Ω)

$||.||$: Euclidean distance (2-norm)

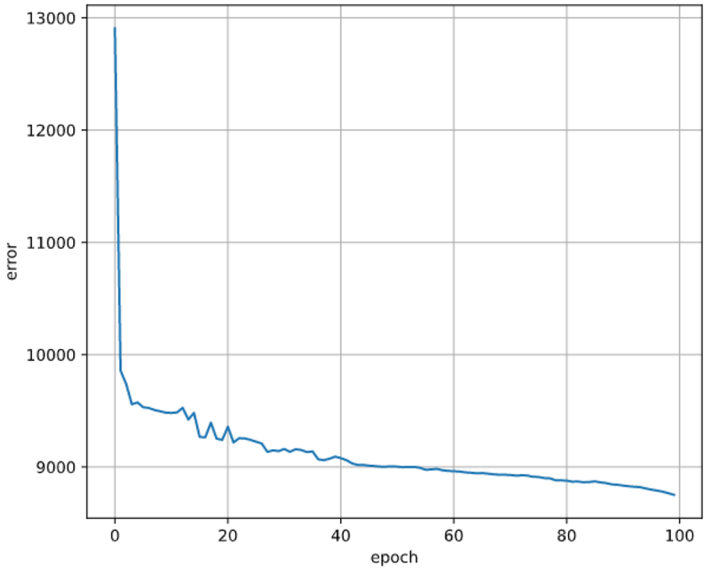


Example of *k*-NN classification, from Wikipedia

Learning vector quantization

Pseudo-code

```
# Initialize the prototype set
>> Create n_prototype prototypes by randomness
# Train the prototype set
>> Choose n_epoch          # no. of training cycles
>> Choose lrate_init       # initial learning rate
>> For i_epoch run from 1 to n_epoch :
       lrate = lrate_init * (1-i_epoch/n_epoch)
       sum_err = 0
       For each datapoint x in the training set :
           Find the nearest neighbor p of x from
           the training set
           sum_err += alpha*||x-p||^2
           If label(x) == label(p):
               Set p += lrate * (x-p)
           Elseif label(x) != label(p):
               Set p -= lrate * (x-p)
               Re-adjust p to make sure p in an
               appropriate range (if needed)
>> Use 1NN classifier on the prototype set to
   label new data points in the testing set
```



Error (sum_err) when training the prototype set with Digits dataset
n_prototype = 100

	Iris dataset	Digits dataset
n_prototype	15 (3 classes)	100 (10 classes, 10 prototypes for each)
n_epoch	30	100
lrate_init	0.5	0.3
alpha	1	1/16

Maximum likelihood + Gaussian distribution

- Maximum likelihood: given a new datapoint x, we classify it to the i-th class if

$$i = \arg\max_j p(x|C_j)$$

C_j : set of parameters that belong to the j-th class

- Assume that the probability $p(x|C_j)$ follows the Gaussian distribution
- Then, C_j can be defined as the tuple of the mean value (μ_i) and the standard deviation (σ_i) of all training datapoints in the i-th class.

$$p(x|C_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma_j}\right)^2}$$

Maximum a posterior + Gaussian Kernel-based PDF estimation

- MAP classifier: given a new datapoint x, we classify it to the i-th class if

$$\boxed{i = \arg\max_j p(C_j|x) = \arg\max_j p(x|C_j)p(C_j)}$$

$$p(C) = \frac{\#\text{ of training datapoints belong to j}-\text{th class}}{\text{total }\#\text{ of training datapoint}}$$

$p(x|C_j)$: calculated based on Gaussian kernel-based PDF estimation

$$p(x|C_j) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right)$$

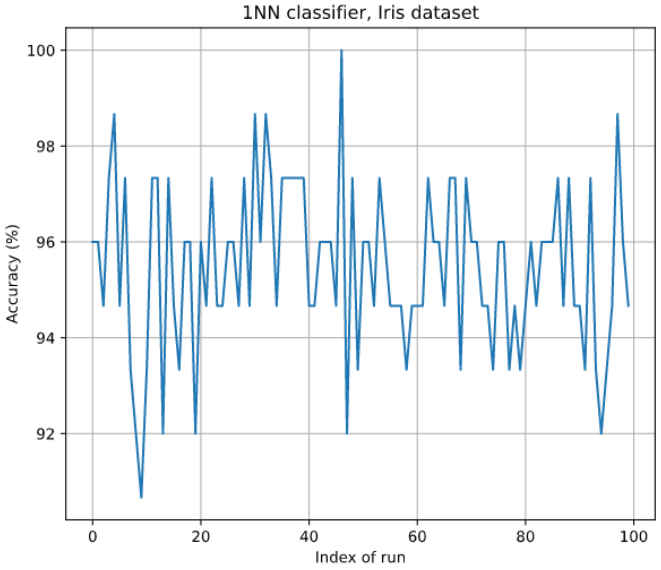
h : the bandwidth ($h > 0$)

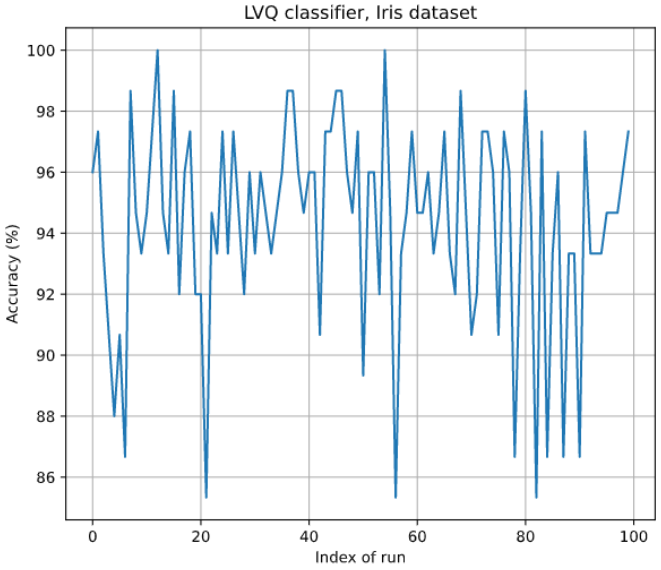
$K(.)$: the Gaussian kernel function

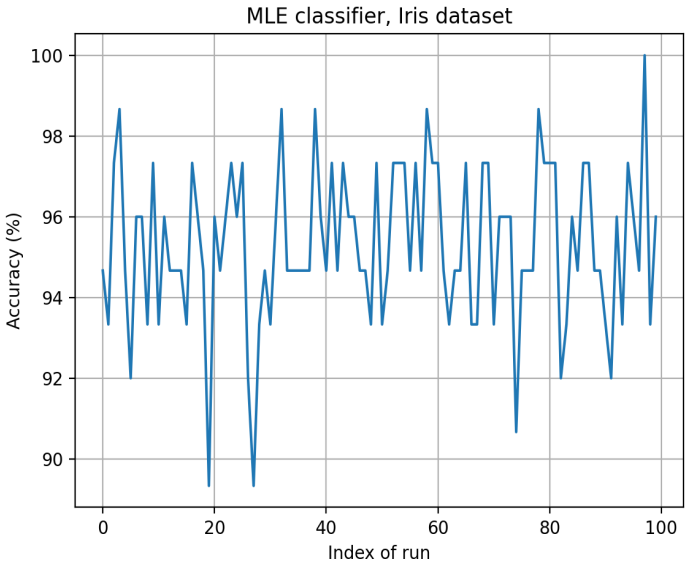
$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$$

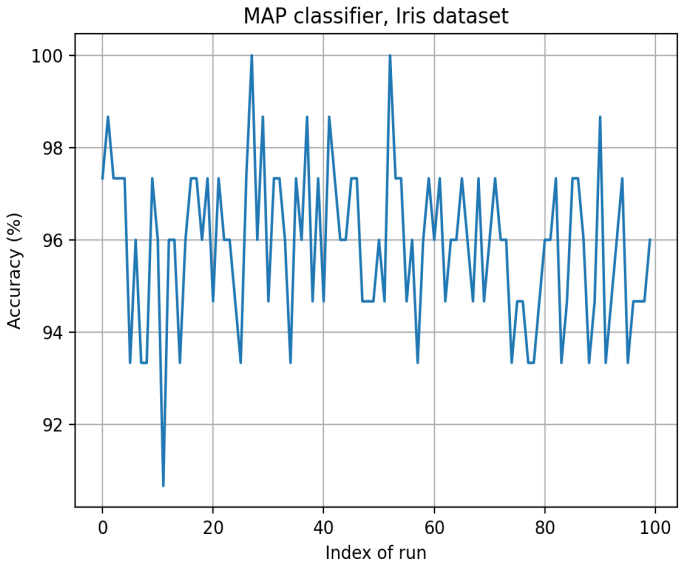
In simulation: $\mathbf{h = 0.337}$ for Iris dataset and $\boldsymbol{h = 3}$ for Digits dataset.

Numerical results/Iris dataset

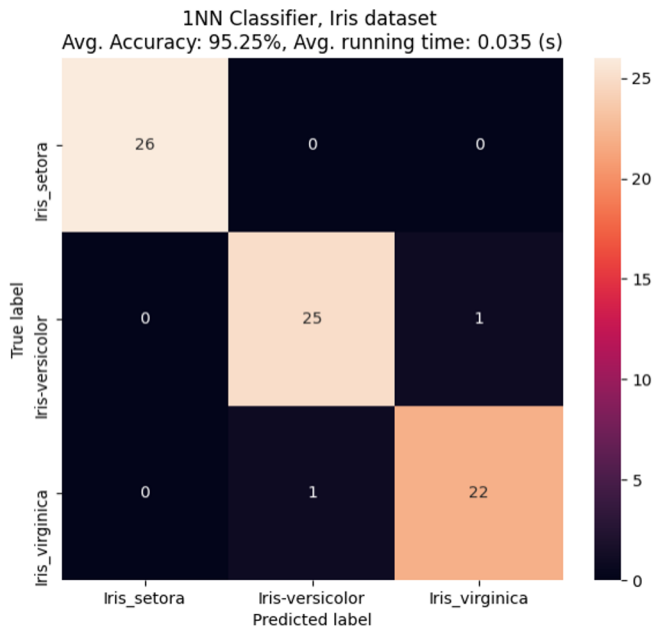


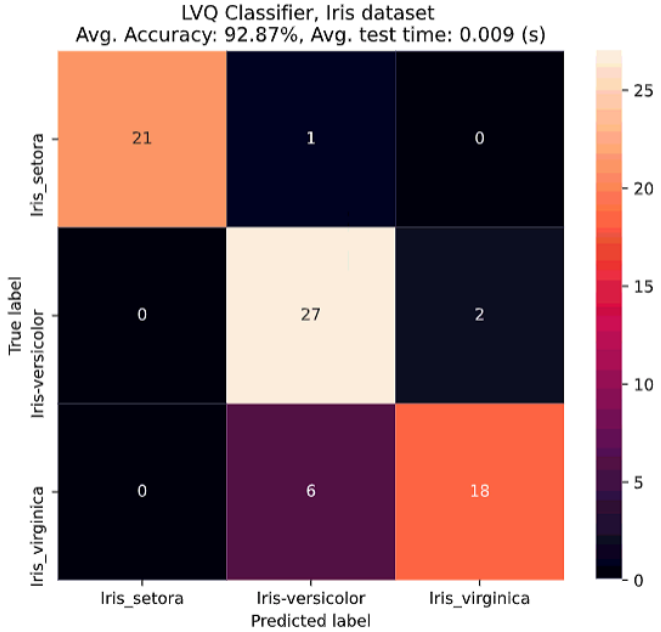


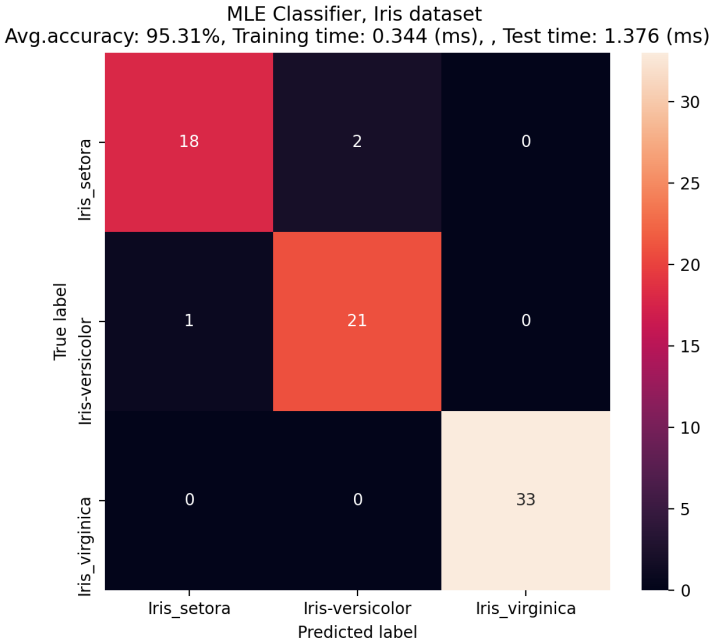


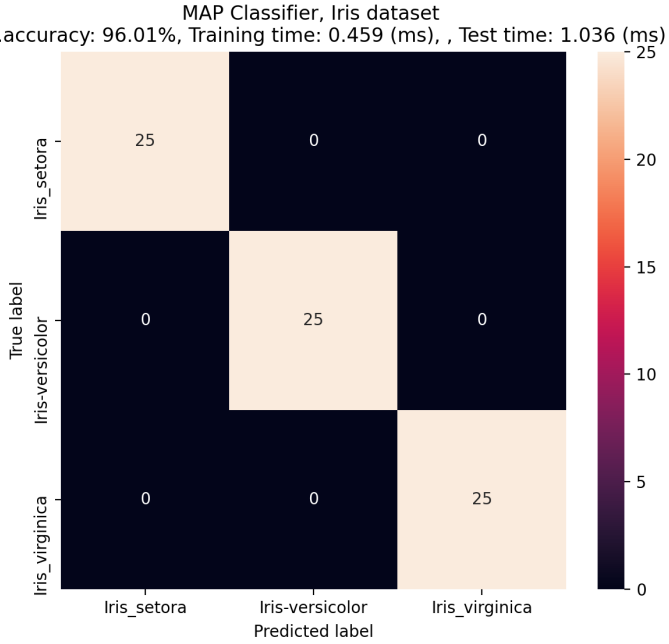


Numerical results/Iris dataset









Numerical results/Iris dataset

	1NN	LVQ	ML	MAP
Avg. accuracy	95.25%	92.87%	95.31%	96.01%
Accuracy variance	3.02%	10.47%	3.26%	3.77%
Avg. train time	--	378.25 (ms)	0.344 (ms)	0.495 (ms)
Avg. test time	34.623 (ms)	8.74 (ms)	1.376 (ms)	1.036 (ms)

Note:

- Train size = 75, test size = 75 (randomly split from the whole dataset in each run).
- All statistics are averaged over 100 runs.
- LVQ: `n_prototype = 15 (3 classes), n_epoch = 30 lrate_init = 0.5`
- MAP: `bandwidth = 0.337`

Comments:

- All 4 methods produce good average accuracy.
- 1NN is the most stable method, it does not require training phase but requires long time for testing since the entire training set is used for prediction.
- LVQ is the most unstable method and requires long time for training to produce a good accuracy.
- ML and MAP take much less time for training and testing than 1NN and LVQ.

Further experimets for LVQ classifier/Iris dataset (1)

Statistics of LVQ with response to some values of n_prototypes:

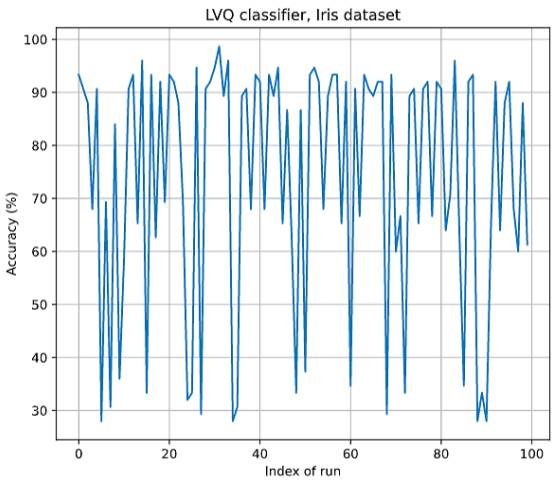


Fig. 1

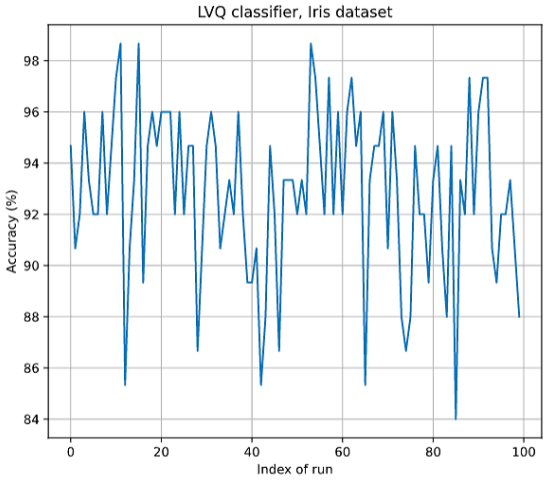


Fig. 2

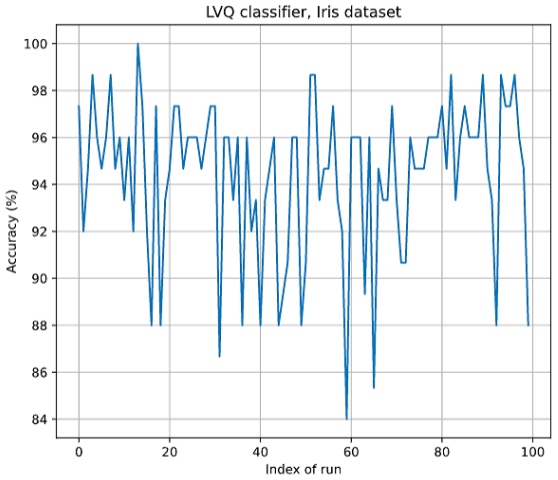


Fig. 3

	Fig.1	Fig. 2	Fig. 3
No. of prototypes for each class	1	5	10
Average accuracy	74.09%	92.87%	94.39%
Accuracy variance	521.22%	10.47%	10.89%
Average train time	117.76 ms	378.25 ms	698.39 ms
Average test time	1.92 ms	8.74 ms	17.27ms

`Note: n_epoch = 30, lrate_init = 0.5`

Further experimets for LVQ classifier/Iris dataset (1)

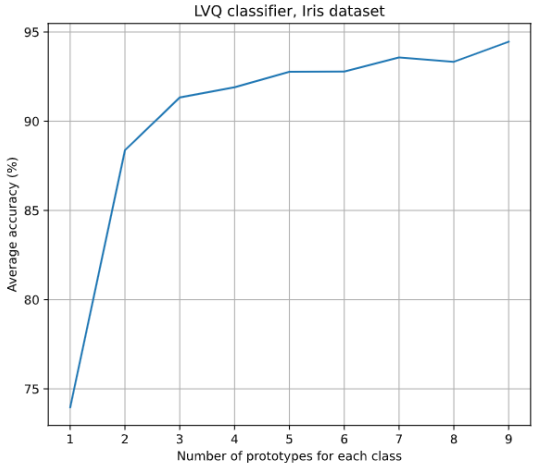


Fig. 1

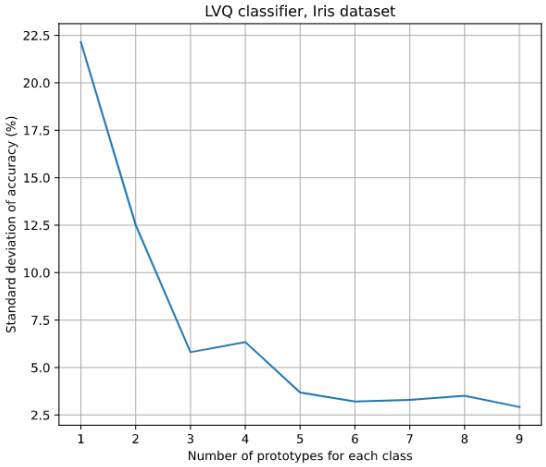


Fig. 2

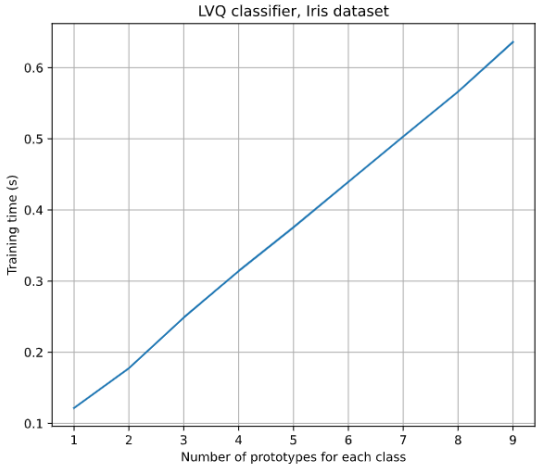


Fig. 3

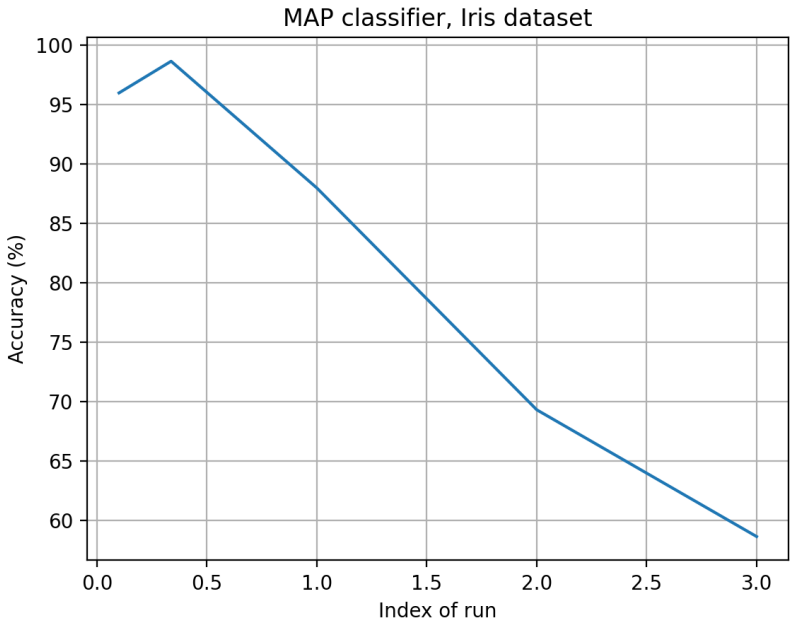
Comments:

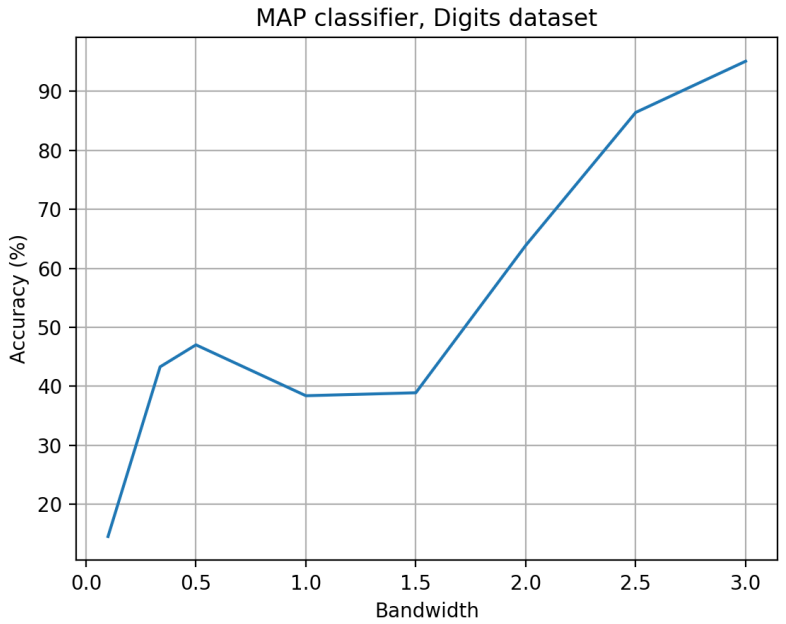
Increasing no. of prototypes for each class helps:

- Improve the average accuracy (Fig. 1)
- Improve the stability (Fig. 2)
- At the cost of lengthening both training and testing time (Fig. 3) (since more prototypes are used)

-> The no. of prototypes is a critical parameter for LVQ, which should be selected at the trade-off between (accuracy + stability) and (training + testing) time.

Further experimets for MAP classifier



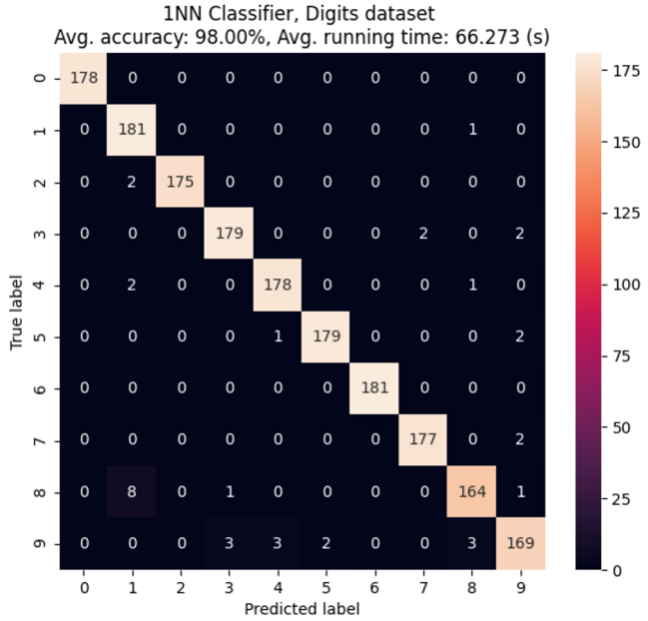


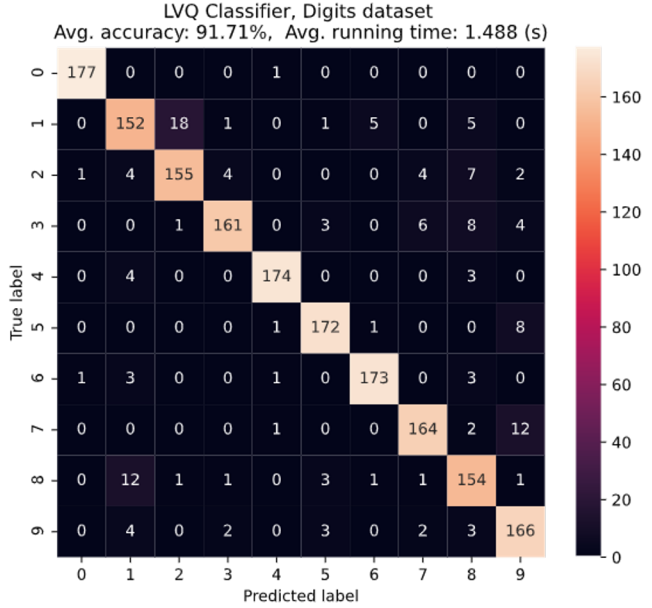
Comments:

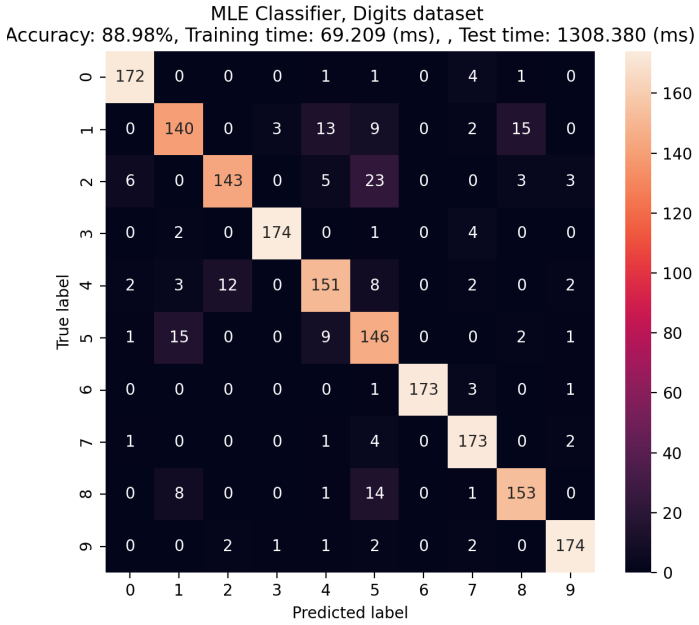
The bandwidth (h) is a critical parameter for MAP classifier with kernel-based PDF estimation:

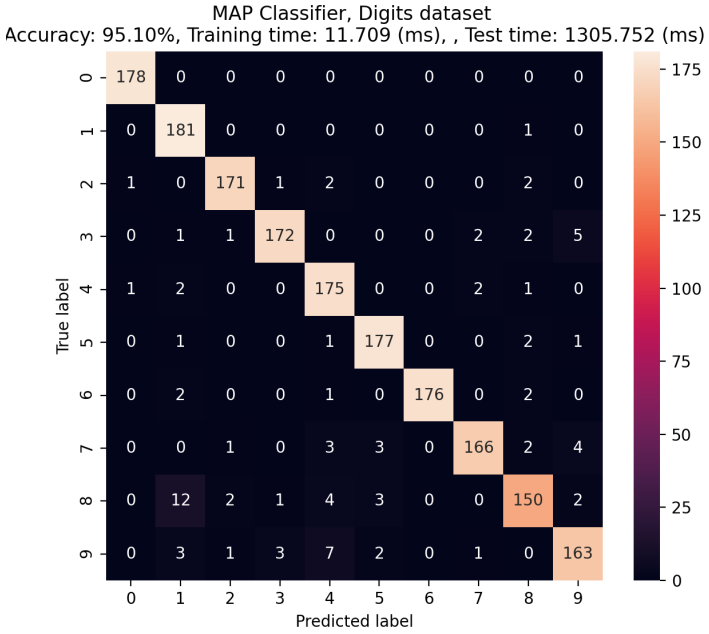
- For Iris dataset: increasing h results in worse accuracy, the best value (producing best accuracy) is h = 0.377.
- For Digits dataset: h=3 produces best accuracy.

Numerical results/Digits dataset









Numerical results/Digits dataset

	1NN	LVQ	ML	MAP
Accuracy	98.00 %	91.71%	88.98%	95.10%
Train time	--	670.443 (s)	69.2 (ms)	11.7 (ms)
Test time	66.273 (s)	1.488 (s)	1.308 (s)	1.305 (s)

Note:

Train size = 3823, test size = 1797 (train set and test set are fixed).

1NN classifier does not require training phase.

LVQ statistics are averaged over 10 runs

LVQ: `n_prototype = 100 (10 classes), n_epoch = 100, lrate_init = 0.3`

MAP: `bandwidth = 3`

Comments:

- 1NN produces good accuracy and does not require training phase, but requires long time for testing since it use the entire training set in prediction.
- LVQ is the most unstable method although its average accuracy is quite good. LVQ requires long time for training to have a good accuracy.
- Since train set and test set are fixed, the accuracy of 1NN, ML and MAP do not change over running times.

Conclusions

- 1NN: - does not require training phase.
  - produces good accuracy for both two considered datasets.
- LVQ: - the no. of prototypes is a critical parameter.
  - may require long traing time to produce a good prototype set.
  - in general, the most unstable method among 4 counterparts.
- 1NN vs. LVQ: LVQ generally requires less validation time since the size of prototype set (used in LVQ) is much smaller than the size of training set (used in 1NN).
- MAP with kernel-based estimation: the bandwidth is an important parameter.
- MLE vs. MAP: MAP produces better accuracy since the kernel-based PDF estimation is used. By adjusting the parameter bandwidth at appropriate values, we can have a good estimation of an unknown PDF.

References

KNN

- J. Brownlee, “Develop k-Nearest Neighbors in Python From Scratch,” *Machine Learning Mastery*, Oct. 23, 2019. https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/ (accessed Jul. 29, 2020).
- T. Vu, “K-nearest neighbors,” *Tiep Vu’s blog*, Jan. 08, 2017. https://machinelearningcoban.com/2017/01/08/knn/ (accessed Jul. 29, 2020).

LVQ

- J. Brownlee, “How To Implement Learning Vector Quantization (LVQ) From Scratch With Python,” *Machine Learning Mastery*, Nov. 03, 2016. https://machinelearningmastery.com/implement-learning-vector-quantization-scratch-python/ (accessed Jul. 29, 2020).

MLE

- Lecture note on Maximum Likelihood Estimation [Online], Available: http://statweb.stanford.edu/~susan/courses/s200/lectures/lect11.pdf

MAP

- T. Vu, “Maximum Likelihood và Maximum A Posteriori estimation,” *Tiep Vu’s blog*, Jul. 17, 2017. https://machinelearningcoban.com/2017/07/17/mlemap/ (accessed Jul. 30, 2020).

- Thank you for your attention
- Q&A

- Please read the file “readme.txt” for more details on the source code.