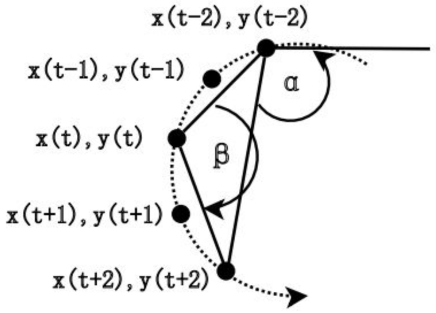
HMM-BASED RECOGNITION OF ONLINE HANDWRITTEN DIGITS

Anna Pederzani Samarati
Thomas Verardo

HMM-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature

Lei Hu, Richard Zanibbi - Rochester Institute of Technology

- Recognition system based on HMM for isolated online handwritten mathematical symbols.
- Design of a continuous left to right HMM for each symbol class.
- Segmental K-means to get initialization of the Gaussian Mixture Models’ parameters.
- Features :
    - Pen-up/down
    - Normalized distance to stroke edges
    - Normalized y-coordinate
    - Vicinity slope
    - Curvature



Goal of the analysis:

The goal of our analysis is to classify online handwritten digits by using a Hidden Markov Model.

The HMM classifier tries to find the probability that a specific class is the most likely to occur given a sequence of observations.

The problem can be formulated as:

$$\underset{i}{argmax}\, P(y_i | x_1, \ldots, x_t)$$

Where y_i is the i 'th digit class.

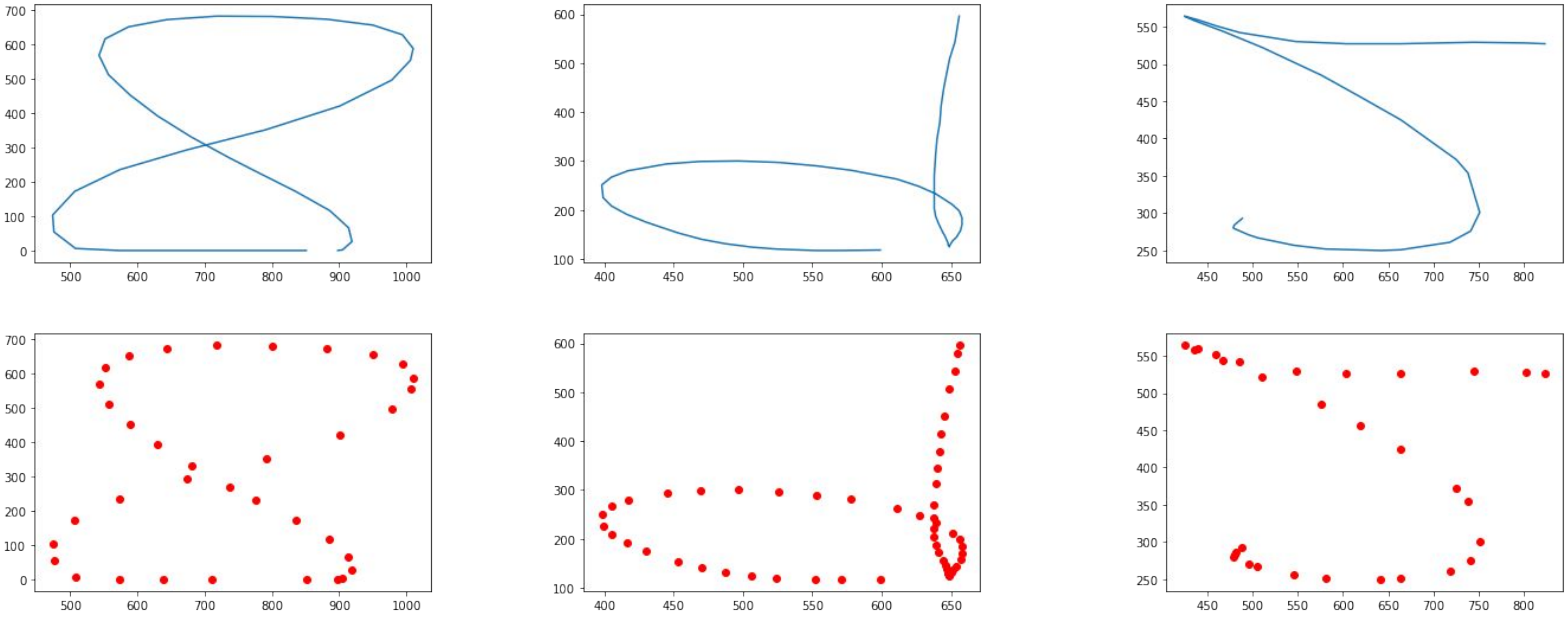
HMM

- HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states.
- It is specified by the parameter set:
    - Transition probability matrix
    - Initial probability distribution
    - Emission probabilities
- Baum-Welch algorithm is a type of expectation-maximization method used in HMM to estimates the parameters.

Dataset

- The dataset contains handwritten digits that have been drawn on digital devices.
- It was composed by 1288 observations. Each observation contains an ordered set of points drawn by the pen.
- Features:
  - **Label** : represent the digit (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
  - **X**: ordered sequence of x-coordinates of the points
  - **Y**: ordered sequence of y-coordinates of the points
  - **Strokes**: number of continuous strokes
  - **Speed:** total time to draw each stroke

Dataset



0 100 200 300 400 500 600 700
500 600 700 800 900 1000
100 200 300 400 500 600
400 450 500 550 600 650
250 300 350 400 450 500 550
450 500 550 600 650 700 750 800
0 100 200 300 400 500 600 700
500 600 700 800 900 1000
100 200 300 400 500 600
400 450 500 550 600 650
250 300 350 400 450 500 550
450 500 550 600 650 700 750 800

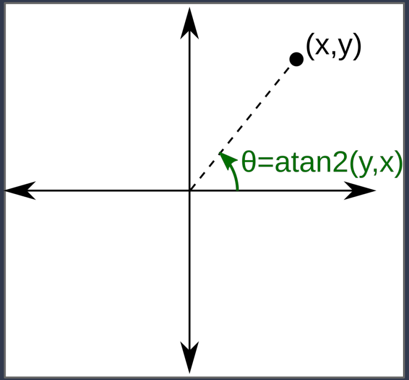
Preprocessing

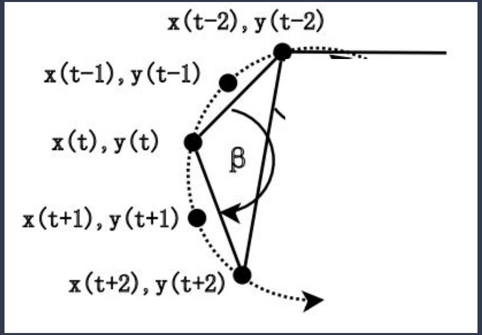
- Duplicate points filtering.
- Remove the observations with less than 15 points.
- Remove the feature speed.
- Smoothing of the features X and Y and creating related features.

Number of observations for each class after preprocessing

```
len of class  0 :  100
len of class  1 :  75
len of class  2 :  105
len of class  3 :  100
len of class  4 :  101
len of class  5 :  110
len of class  6 :  103
len of class  7 :  94
len of class  8 :  108
len of class  9 :  95
```

Added features





Sequences of:

- **Arctan2_1**: angle between the x axis and the ray from the origin to a point.
- **Arctan2_2**: angle between the x axis and the ray from the origin to a point calculated as the difference of two consecutive points.
- **Curvature_1**: curvature with respect to the next and the previous points.
- **Curvature_2**: curvature with respect to the points +2 and -2.
- **Curvature_4**: curvature with respect to the points +4 and -4.

Models

Training:

- We trained a HMM model for each class of our dataset by running the Baum-Welch algorithm by using **hmmlearn**.
- The parameter set is estimated by the algorithm.
- The HMM models have converged.

Testing:

- For each observation in the test set:
  - Calculate the score for each model obtained in the training part.
  - Find the label of the highest score and assigninging it as the predicted class.
- Calculate the accuracy and the confusion matrix of the classification over the test set.

Models

To perform our analysis and find a good HMM model to classify our data, we have tried:

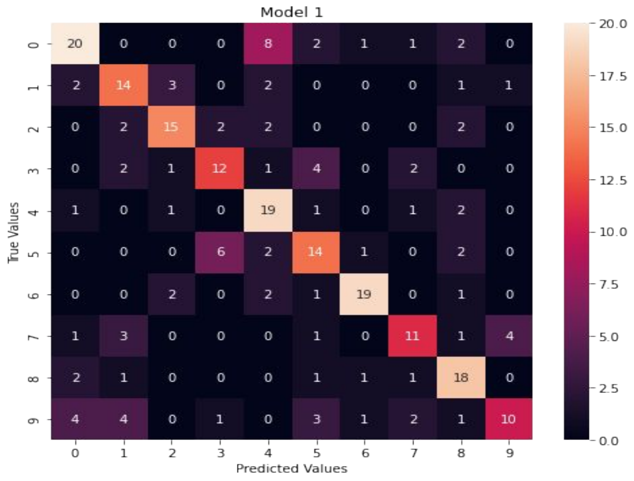
- Two different emission probability distributions:
    - **Gaussian** emission distribution.
    - **Gaussian mixture** emission distribution.

- Models considering different features and different combinations of features.
- Different **parameters** such as the number of hidden states.
    - With GMM models we have also set different number of Gaussian distributions.

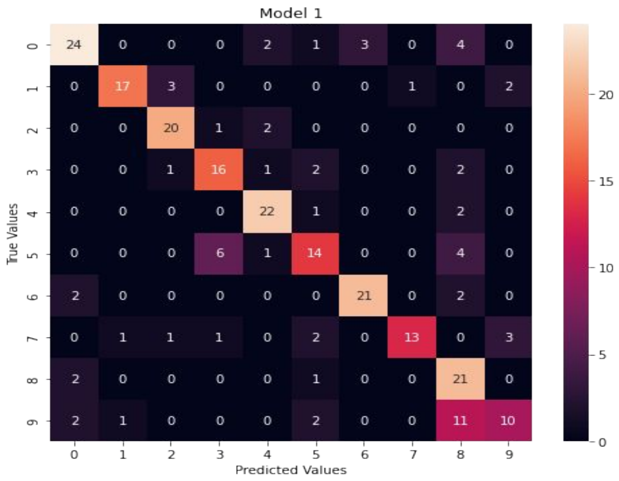
Model 1:

X and Y Gaussian

- Hidden states = 6
  - Accuracy Top-1: **0.61**
  - Accuracy Top-3: **0.82**
- Hidden states = 20
  - Accuracy Top-1: **0.72**
  - Accuracy Top-3: **0.91**



6 hidden states

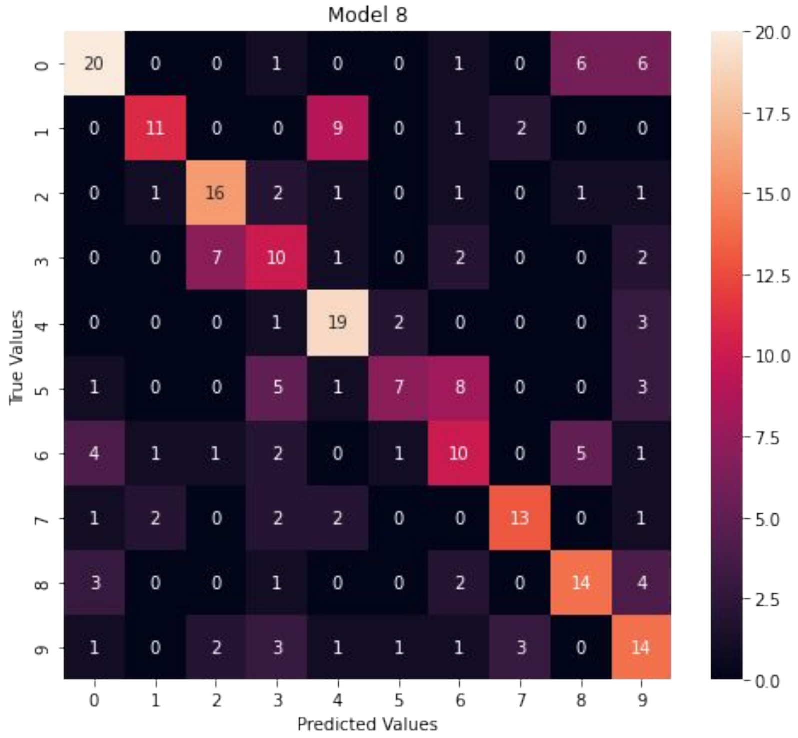


20 hidden states

Model 2:

X and Y
GMM

- Hidden states = 20
- Number of Gaussians = 2
  - Accuracy Top-1: **0.69**
  - Accuracy Top-3: **0.90**

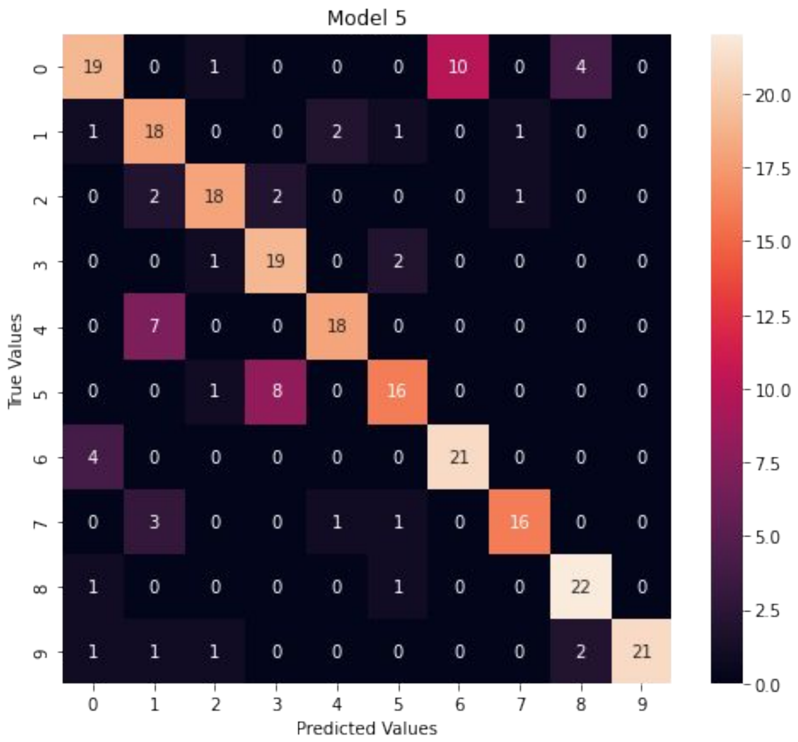


Model 5:

Arctan2_2 Gaussian

- Hidden states = 6
  - Accuracy Top-1: **0.78**
  - Accuracy Top-3: **0.98**

- Hidden states = 20
  - Accuracy Top-1: **0.78**
  - Accuracy Top-3: **0.96**

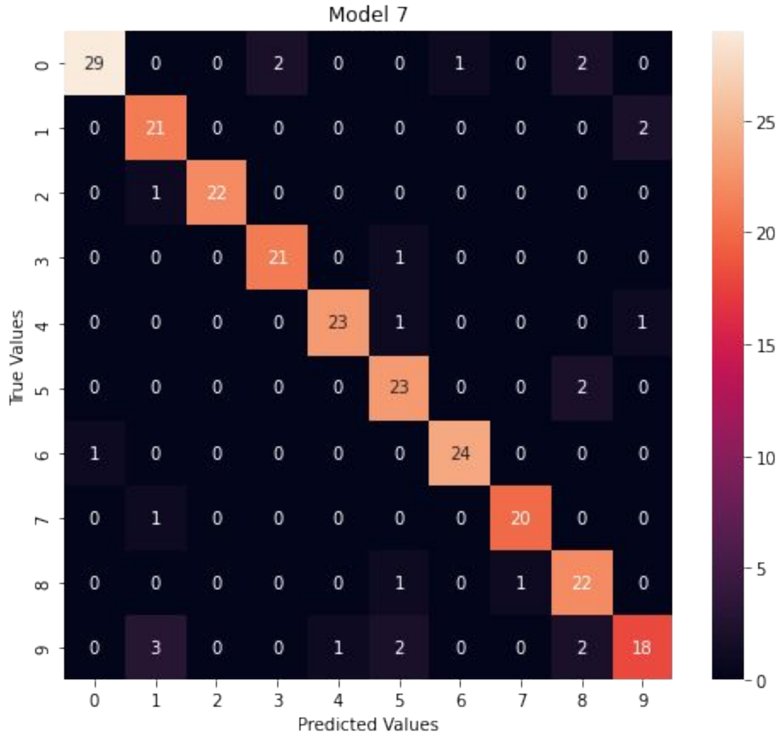


Model 7:

Arctan2_2, X and Y Gaussian

- Hidden states = 6
  - Accuracy Top-1: **0.90**
  - Accuracy Top-3: **0.96**

- Hidden states = 20
  - Accuracy Top-1: **0.89**
  - Accuracy Top-3: **0.96**



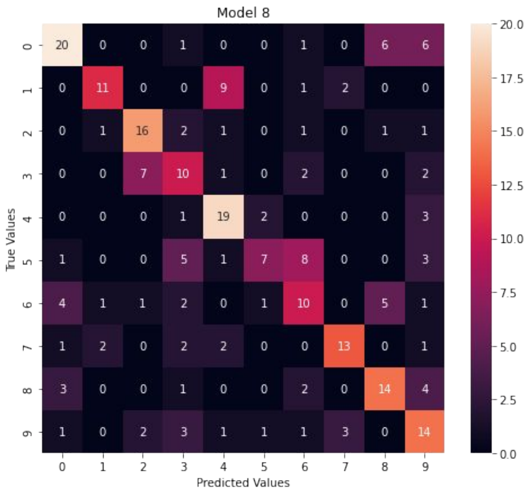
	0	1	2	3	4	5	6	7	8	9
0	29	0	0	2	0	0	1	0	2	0
1	0	21	0	0	0	0	0	0	0	2
2	0	1	22	0	0	0	0	0	0	0
3	0	0	0	21	0	1	0	0	0	0
4	0	0	0	0	23	1	0	0	0	1
5	0	0	0	0	0	23	0	0	2	0
6	1	0	0	0	0	0	24	0	0	0
7	0	1	0	0	0	0	0	20	0	0
8	0	0	0	0	0	1	0	1	22	0
9	0	3	0	0	1	2	0	0	2	18

Model 8:

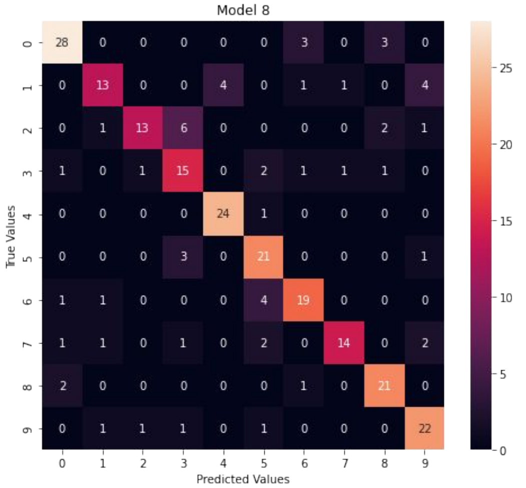
Arctan2_1 and curvature_1 Gaussian

- Hidden states = 6
  - Accuracy Top-1: **0.54**
  - Accuracy Top-3: **0.86**

- Hidden states = 20
  - Accuracy Top-1: **0.77**
  - Accuracy Top-3: **0.91**



6 hidden states



20 hidden states

To conclude

- Using the **smoothed X** and **Y**, the accuracy of the models do not change significantly.
- The feature **curvature** alone is not able to explain the observations.
- **GMM** models do not work well as the Gaussian models.
- Considering the time needed to train the models and the accuracy we have found that a good tradeoff for the number of **hidden states is 6**.

- The model with the best performance is **model 7** (with features: **Arctan2_2**, **X**, **Y**).

- Further improvements: implement more features, perform other preprocessing operations, apply these methods also to other handwritten letters and sings.