Pattern Recognition
Spring 2019

Final report

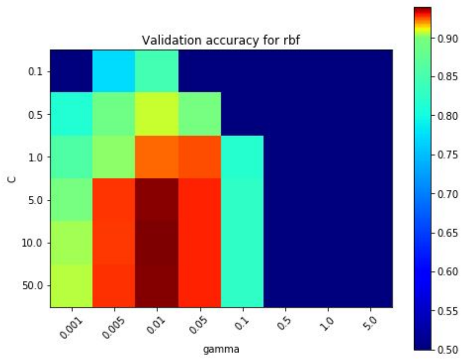
Authors : Nicolas Fuchs, Jérôme Vonlanthen, Thomas Schaller and Sylvain Julmy

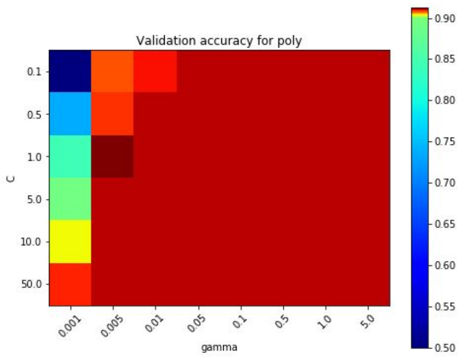
Group Organization

Tasks

SVM

The SVM task was achieved by using the sklearn[1] library, which is very simple to use and provides a lot of examples for a better understanding.
In order to find the best parameter, we used a grid search approach with various parameters and display the heatmaps of the results for each tested kernel (see below).

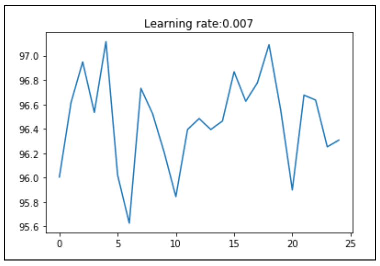




Using this, we were able to obtain a 97.9% accuracy.

MLP and CNN

The MLP and CNN tasks where both achieved by using the pytorch[2] library. We also performed a gridsearch like approach to find the best parameters for each of them.
For the MLP, we finally got a model with 512 hidden neuron and a learning rate of 0.001, other learning rate where not stable at all for this task.
We also go a learning rate of 0.001 for the CNN model since higher one (like 0.002 or 0.007) showed rollercoaster like curve :



Finally, we achieved a accuracy of 97.0% for the MLP and 97.8% for the CNN.

[1]`https://scikit-learn.org/stable/modules/svm.html`
[2]`https://pytorch.org/`

Keyword spotting

The features we are using for this task are the suggested one : lower contour, upper contour, black/white transitions, fraction of black pixels in the window, fraction of black pixels between the upper and lower contour and the gradient.
The evaluation part was kind of hard to understand and to set up, since we didn't get immediatly how to measure the quality of our solution.
We implemented our own DTW function but it was really slow and that's messed our work too, since we had to wait between evaluation and run in order to work on the task. The implementation is shown on the right.

```python
def __dtw(self, x, y, dist):
  len_x, len_y = len(x), len(y)
  window = [(i, j) for i in range(len_x) for j in range(len_y)]
  window = ((i + 1, j + 1) for i, j in window)
  D = defaultdict(lambda: (float('inf'),))
  D[0, 0] = (0, 0, 0)
  for i, j in window:
    dt = dist(x[i - 1], y[j - 1])
    D[i, j] = min((D[i - 1, j][0] + dt, i - 1, j),(D[i, j - 1][0] + dt, i, j -1),
    (D[i - 1, j - 1][0] + dt, i - 1, j - 1),key = lambda a: a[0])

  path = []
  i, j = len_x, len_y
  while not (i == j == 0):
    path.append((i - 1, j - 1))
    i, j = D[i, j][1], D[i, j][2]
  path.reverse()
  return D[len_x, len_y][0], path
```

Signature Verification

We use the same DTW function to solve this task as the previous one, so it was slow as well.
For the features, we computed the velocity for x and y as well as the pressure, and normalize them with respect to the user.

General thoughts and feedback

The two differents kind of exercices : implementing the algorithms by ourselves and using existing solutions is a good approach of the project. Whit that, we can understand the algorithms and discover various libraries to solve specific tasks (for example, the gridsearch part or when ploting heatmaps).
As a personal feedback (Sylvain Julmy), it was hard to use the Python programming language with the other members of the group, since I almost never used it. Therefore reading, understanding and using the implementation of my comrades lost me a lot of time during the tasks.