Machine Learning (HWS21)
Assignment 1: Naive Bayes

The archive provided to you contains this assignment description, a dataset in a binary format, as well as Python code fragments for you to complete. Comments and documentation in the code provide further information.

It suffices to fill out the "holes" that are marked in the code fragments provided to you, but feel free to modify the code to your liking. You need to stick with Python though.

Provide a single ZIP archive with name `ml21-<assignment number>-<your ILIAS login>.zip`. The archive needs to contain:

- A **single PDF report** that contains answers to the tasks specified in the assignment. Do not simply convert your Jupyter notebook to a PDF! Write a separate document, stay focused and brief. **Use at most 10 single-column pages.**
- All the **code** that you created and used in its original format.
- Optional: A PDF document that renders your Jupyter notebook with all figures. (If you don't use Jupyter, then you obviously do not need to provide this.)

You need to adhere to the above guidelines in your submission, otherwise we may grade your solution as a FAIL.

Generally, your report should

- include a high-level description of your approach and helpful figures,
- be self-explanatory (i.e., refer to code *only* for implementation-only tasks),
- follow standard scientific practice,
- include appropriate references if you used additional sources or material,
- not include any hand-written notes,
- label all figures/tables and refer to figures/tables via their labels,
- use one section per task and one subsection per subtask, each numbered with the (sub)task numbers from the assignment sheet.

Your report will be downgraded if you do not follow these points (e.g., you can't get EXCELLENT).

Hand-in your solution via ILIAS until the date specified there. This is a hard deadline.

MNIST Dataset

We will use a preprocessed variant of the MNIST digits dataset in this assignment. The task is to classify hand-written digits. There is one class for each digit (i.e., classes $0, 1, 2, ..., 9$). The features represent a scanned image (28×28 pixels, values in $\{ 0, 1, \ldots, 255 \}$). The dataset contains both training data ($\approx$ 6000 images per class) and test data ($\approx$ 1000 images per class). See `http://yann.lecun.com/exdb/mnist/` for more information.

1 Training

Provide a function `nb_train` that trains a Naive Bayes classifier for categorical data using a symmetric Dirichlet prior and MAP parameter estimates. A description of the parameters and expected result can be found in the Python file. For example, you may assume that all features take values in $\{ 0, 1, \ldots, K-1 \}$, where K is the number of possible values.

Hints. First, try out your function with a uniform Dirichlet prior ($\alpha = 1$) on the small example provided by us. Then try it with add-one smoothing ($\alpha = 2$). If both results are correct, train your model on the MNIST digits dataset. You may need to work with a sample of the training data (the source code to do this is provided). Can you get your implementation to train the model on the full dataset in a reasonable time of, say, a few seconds? If not, don't worry and continue with a sample of the data.

2 Prediction

Provide a function `nb_predict` that takes your model and a set of examples, and outputs the most likely label for each example as well as the log probability (confidence) of that label. A description of the parameters and expected result can be found in the Python file.

Hints. Again, start with our small example, check that your implementation is correct, then try it on the digits dataset.

3 Experiments on MNIST digits data

a) Train your model with $\alpha = 2$ on the MNIST training dataset, then predict the labels of the MNIST test data using your model. What is accuracy ($= 1 -$ misclassification rate) of your model?

b) Plot some test digits for each predicted class label (code provided). Can you spot errors? Then plot some misclassified test digits for each predicted class label (code provided). Finally, compute the confusion matrix (`https://en.wikipedia.org/wiki/Confusion_matrix`, code provided). Discuss the errors the model makes.

4 Model selection (optional)

Use cross-validation to find a suitable value of the hyperparameter α (of the symmetric Dirichlet prior). Also plot the accuracy (as estimated via cross-validation) as a function of α. Discuss.

Note. Some hints to get you started with running cross-validation are given in the Python file.

5 Generating data

a) Implement a function `nb_generate` that generates digits for a given class label. A description of the parameters and expected result can be found in the Python file.

b) Generate some digits of each class for your trained model and plot (code provided). Interpret the result. Repeat data generation for different models by varying the hyperparameter α. How does α influence the results? Discuss.