The objective of this lab is for you to program in Matlab a discrete Markov process, train it on sequences of characters, and sample sequences from it. The TA will first demonstrate the results of this on several datasets of sequences, and then you will program them, replicate those results, and further explore the datasets. You can use the textbook, lecture notes and your own notes.

I Datasets

Use sequences of characters defined on N states, say $N = 5$ with states $\{1, 2, 3, 4, 5\}$, and sequences such as `32334`, `5432544551`, etc. Once this works, use English sentences as training sequences, where the states correspond to the English letters and punctuation, e.g. `a`–`z`,`.`,`;`, etc. You can also consider as states whole words rather than individual characters, and have the discrete Markov process generate sequences of words.

II Discrete Markov processes

What you have to do Write Matlab code to implement the following:

- *The learning problem*: given a dataset of training sequences over a set of N states, estimate the parameters:
  - The transition matrix $\mathbf{A}$ of $N \times N$.
  - The initial distribution $\boldsymbol{\pi}$ of $N \times 1$.
- *The sampling problem*: given parameters $(\mathbf{A}, \boldsymbol{\pi})$, generate a sequence of length T by sampling from the discrete Markov process.

Questions to consider:

- Given the parameters $(\mathbf{A}, \boldsymbol{\pi})$ learnt from a set on English sequences:
  - Plot them as an image.
  - Inspect their values and try to guess what kind of sequences will be generated with high probability.
  - How will sequences sampled from these parameters $(\mathbf{A}, \boldsymbol{\pi})$ differ from:
    - True English text sequences.
    - Random sequences of English characters. (We call "random sequences" sequences generated randomly, i.e., picking each element independently from each other at random from the states.)
  - If you train a discrete Markov process on a set of random sequences, how will $(\mathbf{A}, \boldsymbol{\pi})$ look like?

The following Matlab functions will be useful (among others): `randi rand cumsum`.

The objective of this lab is for you to program in Matlab ensembles of learners (based on bagging, both for regression and classification), apply them to some datasets and observe their behavior. The TA will first demonstrate the results of the algorithms on several datasets, and then you will program them, replicate those results, and further explore the datasets with the algorithms. You can use the textbook, lecture notes and your own notes.

I Datasets

Construct your own toy datasets to visualize the result easily and be able to get the algorithm right. For regression, take the input instances $\{\mathbf{x}_n\}_{n=1}^N$ in $\mathbb{R}$ and the labels $\{y_n\}_{n=1}^N$ in $\mathbb{R}$. Generate a noisy sample from a known function, e.g. $y_n = f(x_n) + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ and $f(x) = ax + b$ or $f(x) = \sin x$. For binary classification, take the input instances $\{\mathbf{x}_n\}_{n=1}^N$ in $\mathbb{R}$ or $\mathbb{R}^2$ and the labels $\{y_n\}_{n=1}^N$ in $\{-1, +1\}$.

II Ensemble learning using bagging

In bagging, we generate L bootstrap samples of the training set $\{(x_n, y_n)\}_{n=1}^N$, train a learner in each, and combine their outputs by averaging them (for regression) or by majority voting (for classification). So programming this in Matlab just requires a loop over the $l = 1, \dots, L$ learners. You can use any learner you want, we suggest the following:

- Polynomial regression (degree 1 gives a linear regression).
- Decision trees, grown to 1 or 2 levels (decision stumps) or more deeply, both for regression and classification.
- k-nearest-neighbor classifier.
- Support vector machine classifier.

III What you have to do

Program the ensemble (learning it and applying it to test data). Note: you can use Matlab's code for the individual learners, e.g. for decision trees you can use the function `fitctree`; make sure you understand how to use it properly, in particular how to set any (default) parameters in may require.

Regression Consider an ensemble of L polynomials all of the same degree k (e.g., $k = 1$ corresponds to an ensemble of linear regressors). To visualize the results, create the following plots:

- Plot the dataset (y_n vs x_n), the true function $f(x)$ from which you generated the data, the L learners' functions and the ensemble function.
- Plot the validation error as a function of the degree k of the polynomials, and of the number of learners L.

Questions to consider:

- How does the ensemble regressor look, compared to the individual regressors?
- How does the validation error of the ensemble compare with the validation errors of the individual learners?
- How does the validation error depend on the degree k (= complexity) of the polynomials?
- How does the validation error of the ensemble behave as L increases?

Classification Repeat as for regression. Define ensembles where the members are of the same type, e.g. k-nearest-neighbor classifiers with fixed $k = 1$.

Questions to consider:

- Similar questions as for regression, suitably modified for your learners (e.g. effect of k on the k-nearest-neighbor classifier, effect of the tree depth on decision trees).
- How do the following ensembles (of fixed size L) compare with each other: decision stumps; deep decision trees; k-nearest-neighbor classifiers; linear SVMs.