The objective of this lab is for you to explore the behavior in Matlab of ensembles of learners (based on bagging, for regression or classification) and apply them to some datasets. The TA will first demonstrate the results of the algorithms on several datasets. Then, you will replicate those results, and further explore the datasets with the algorithms.

We provide you with the following:

- The script `lab06_regr.m` sets up a regression problem (on a toy dataset) and plots various figures. It uses polynomials of different degrees as base learner, and bagging as ensembling mechanism.
- `polytrain.m` trains a polynomial regression of a given degree.

I Datasets

Construct your own toy datasets to visualize the result easily. For regression, take the input instances $\{x_n\}_{n=1}^N$ in $\mathbb{R}$ and the output values $\{y_n\}_{n=1}^N$ 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:

- Regression: polynomials (degree 1 gives a linear regression); Gaussian RBF networks; decision trees.
- Classification: logistic regression; k-nearest-neighbor classifier; support vector machine; decision trees.

Use the provided script `lab06_regr.m` as driver for your experiments. It creates a dataset (training and validation) for regression, trains L learners (specifically, polynomials of different degree, using the function `polytrain.m` from another lab), and plots the results. See more details below.

Instead of polynomials, other types of learner may be used; just call the corresponding function to train it. The point of this lab is to learn about the combination of learners in an ensemble (rather than about the specific learners themselves, which do in other labs). We suggest you try functions from other labs, or any function from Matlab or from the web, for regression or for classification. In particular, decision trees are among the most interesting learners to use in an ensemble. You can use the Matlab function `fitctree` to train them. Make sure you understand how to use it properly, in particular how to set any (default) parameters it may require.

Exploration: regression We discuss polynomials as an example. Consider an ensemble of L polynomials of the same degree k ($k = 1$ corresponds to linear regressors). To visualize the results, we plot the following:

- The dataset (y_n vs x_n), the true function $f(x)$ from which the data was generated, the L learners' functions and the ensemble function.
- The training and 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?

Exploration: classification Define ensembles where the learners are of the same type, e.g. k-nearest-neighbor classifiers with fixed $k = 1$, and proceed as with regression. 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.