EE-613: Machine Learning for Engineers
Practical session 12
(Boosting)

1 Download and test

```
wget http://www.idiap.ch/~fleuret/files/EE613/EE613-pw12.tgz
tar zxvf EE613-pw12.tgz
cd EE613/pw12
./do.sh example
```

after a few seconds you should obtain the following printout in the terminal

```
Loading the train data ... done.
round 0 train_error 0.0788462 test_error 0.0816327
round 1 train_error 0.0754352 test_error 0.134615
round 2 train_error 0.0813492 test_error 0.0769231
round 3 train_error 0.0766284 test_error 0.106383
round 4 train_error 0.0728346 test_error 0.147541
round 5 train_error 0.0754352 test_error 0.134615
round 6 train_error 0.0789981 test_error 0.08
round 7 train_error 0.0793037 test_error 0.0576923
round 8 train_error 0.0810811 test_error 0.0784314
round 9 train_error 0.072549 test_error 0.118644
```

2 Data set

The data-set is the Wisconsin Breast Cancer data set. Each sample corresponds to a cell. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The class corresponds to malignant (1) or benign (0).

Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter^2 / area - 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension ("coastline approximation" - 1)

3 Programming

3.1 Compilation

It is suggested to implement each question in the corresponding function

```
void computation_question1(DataSet *train_set, DataSet *test_set) {
  ...
}
```

and to test it with

```
./do.sh [--nb-rounds xx] question1
```

The `--nb-rounds` option will run the computation several times with different random initializations.

Note that if you re-compile the code in debug mode with

```
make clean && make -k DEBUG=yes
```

before running `do.dh`, there will be additional checks for out-of-bound errors at run time.

3.2 Classes

A **DataSet** stands for a set of samples from $\mathbb{R}^d$ with binary labels.

```
class DataSet {
public:
  virtual int nb_features() = 0;
  virtual int nb_samples() = 0;
  virtual scalar_t feature(int s, int f) = 0;
  virtual int label(int s) = 0;
};
```

and **ConcreteDataSet** inherits from **DataSet** and stands for an object which actually contains data loaded from a file. The `split` method splits the content into two subsets, putting each sample in the training set with the provided probability.

```
class ConcreteDataSet : public DataSet {
public:
  ConcreteDataSet();
  virtual ~ConcreteDataSet();
  int nb_features();
  int nb_samples();
  scalar_t feature(int s, int f);
  int label(int s);
  void allocate(int nb_samples, int nb_features);
  void read(const char *name);
  void write(const char *name);
  void split(ConcreteDataSet *train, ConcreteDataSet *test,
             scalar_t proba_train);
};
```

A **Classifier** is a virtual object standing for a classifier which can be trained from a sample set and compute a response on a sample. Both `train` and `response` have to be implemented in any sub-class to be instantiated.

```
class Classifier {
public:
  virtual void train(DataSet *data) = 0;
  virtual scalar_t response(DataSet *data, int s) = 0;
  scalar_t error_rate(DataSet *data);
};

```

The class **Stump** inherits from **Classifier** and implements a very simple classifier which compares a certain feature to a threshold and responds $+1$ or -1.

It has an additional training method which accepts weighted samples and returns the weighted error.

```
class Stump : public Classifier {
  int _feature_index;
  scalar_t _threshold;
  scalar_t _polarity;
public:
  // Trains and return the empirical error rate
```

```
7     scalar_t train(DataSet *data, scalar_t *weights);
8     void train(DataSet *data);
9     scalar_t response(DataSet *data, int s);
10  };
```

4 Questions

Question 1: Bagging

Write a subclass **BaggedClassifier** of **Classifier** which implements a simple voting procedure with stumps. Each of them will be trained by bootstrapping samples from the complete training set.

Help 1: The definition of the **BaggedClassifier** class will be

```
class BaggedClassifier : public Classifier {
  int _nb_stumps;
  Stump *_stumps;
public:
  BaggedClassifier(int ns);
  virtual ~BaggedClassifier();
  void train(DataSet *data);
  scalar_t response(DataSet *data, int s);
};
```

Help 2: You can implement the bootstrapping with weighting: The weight of a sample will be proportional to the number of times it has been re-sampled (hence will be zero if it was never picked).

Question 2: AdaBoost

Implement a subclass **BoostedClassifier** of **Classifier** which implements a linear combination of **Stump**s obtained with the standard AdaBoost procedure.

Help 1: The definition of the **BoostedClassifier** class will be

```
class BoostedClassifier : public Classifier {
  int _nb_stumps;
  Stump *_stumps;
  scalar_t *_weights;
public:
  BoostedClassifier(int ns);
  virtual ~BoostedClassifier();
  void train(DataSet *data);
  scalar_t response(DataSet *data, int s);
};
```

Help 2: For AdaBoost, at step k, given that the strong classifier built so far is f_k, and the class y_n is binary (0 or 1), we have:

- the weight of sample x_n at that step is

$$\nu_{k,n} = \frac{1}{Z} \exp\left(-\left(2y_n - 1\right) f_k\left(x_n\right)\right)$$

  where Z is such that $\sum_n \nu_{k,n} = 1$.

- the weight of the chosen weak learner h_{k+1} is

$$\alpha_{k+1} = \frac{1}{2} \log\left(\frac{1 - e_{k+1}}{e_{k+1}}\right)$$

  where $e_{k+1} \in [0, 1]$ is the weighted error of h_{k+1}.

Question 3: Dimensional blow-up

Write a subclass **QuadraticDataSet** of **DataSet** which stands for an artificial data set with both the original features, and one additional for every pair of features, equal to their product. Hence, if the original data set had D features, this new one has $D + \frac{(D+1)D}{2}$.

Test the classifiers with it.

Help: Such an object will have a pointer to a **DataSet** and every method will dynamically calls the ones of the original data set.