Assignment 1

Data Mining and Machine Learning (2IIG0)

Stiven Dias, Sibylle Hess and Ghada Sokar

This is your first assignment of the Data Mining and Machine learning course. The assignment is composed of six questions, five of them are auto-graded. The last question is open-ended. Multiple-choice questions have four possible choices; among these, only one answer is correct. The number of points for each question is written at the beginning of it. The maximum achievable score is 100 points. In all the questions except the open-ended one, you have to implement the algorithm required in the question yourself from scratch using Python, analyze the results you get, and carefully choose one of the choices. Only in the last question, we are going to use scikit-learn.

Submit the answers for the auto-graded questions in the **HW1** quiz in Canvas. You have to submit this quiz individually. That is, every group member has to fill out their own quiz, even when you are encouraged to work on the exercises together. You can submit the quiz multiple times, the latest one will be graded. Please fill this quiz out carefully.

The open question can be submitted on a group level by means of the **Open Question HW1** assignment in Canvas. That is, one of your group members should upload:

- A PDF document containing the answer to the open question. Include a paragraph about the workload distribution as well. We assume that you can decide yourself how to organize the work in your group. But if someone didn't contribute at all, or only marginally, then you should note this in the PDF.

- A Python file/ notebook containing the implementations you used to answer the implementation-based questions. This notebook is your insurance, it shows what you did and which results you got. If you want to argue for a higher grade/get partial points then you can back it up with the results stated in the uploaded notebook. However, if we see that you couldn't provide a working implementation, but just guessed the correct answer in the auto-graded format, then we will also subtract the corresponding points from your grade.

1. (20 points) **Gradient Descent**. Consider the function

$$f(\mathbf{x}) = x_1^4 + 4x_1x_2 + 2x_2 + \frac{1}{2}x_2^2$$

We try to find the minimum of f with the gradient descent algorithm. In particular, we evaluate various strategies to define a step-size policy. Therefore:

- Implement a function that takes a point $\mathbf{x} = (x_1, x_2)$ and returns the the gradient of f at this point.
- Implement a function

  `gradient_descent(f, grad_f, eta, x_0, max_iter=100)`

  that performs `max_iter` gradient descent steps:

  `x_tplus1` $\leftarrow$ `x_t-eta(t)grad_f(x_t)`

  where `f` is the function to be minimized, `grad_f` returns the gradient, `eta`: $\mathbb{N} \to \mathbb{R}$ returns the step-size at iteration t and `x_0` is the starting point (initialization).
- Implement a function `eta_const(t,c=0.1)` that returns for each iteration `t` the constant `c` as stepsize.
- Implement a function `eta_sqrt(t,c=0.1)` that returns for iteration `t` the step size $c/\sqrt{t+1}$.
- Implement a function `eta_multistep(t, milestones, c=0.1, eta_init=0.1)` that returns a step size that is initially set to `eta_init`, but is decayed at each milestone by multiplying it with factor `c`. For example:

$$\texttt{eta\_multistep(t, [20,50], c=0.1, eta\_init=1)} = \begin{cases} 1, & \texttt{t} < 20 \\ 0.1 & 20 \leq \texttt{t} < 50 \\ 0.01 & 50 \leq \texttt{t} \end{cases}$$

Perform 100 iterations, starting at `x_0`=(1,1) and return the function value of x_{100} for the following step size policies:

A. What is $f(x_{100})$ when using the constant step size policy `eta_const(t,c=0.01)`?

B. What is $f(x_{100})$ when using the step size policy `eta_sqrt(t,c=0.1)`?

C. What is $f(x_{100})$ when using the constant step size policy `eta_multistep(t,[10,60,90],c=0.5, eta_init=0.1)`?

2. (10 points) **Coordinate Descent.** Consider the function

$$f(\mathbf{x}) = \frac{1}{2}x_1^4 - x_1x_2 + x_2^2 + x_2x_3 + x_3^2$$

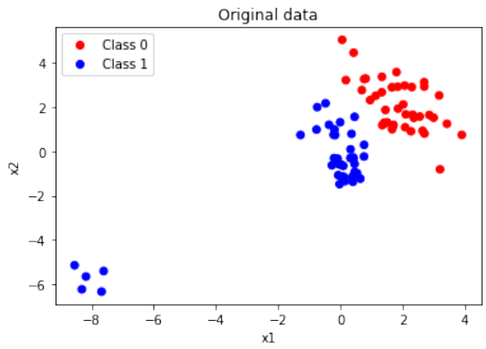
We try to find the minimum of f with coordinate descent. Therefore:

- Implement for each coordinate x_i ($i \in \{1, 2, 3\}$) a function `argmin_xi(x)` that returns $\arg\min_{x_i} f(x)$.
- Implement a function `coordinate_descent(f, argmin, x_0, max_iter=100)` that performs `max_iter` coordinate descent steps. So, at iteration `t`, we have to go through all the coordinates (indexed by `i`, going from the first to the last coordinate index) and update each coordinate with the following update rule:

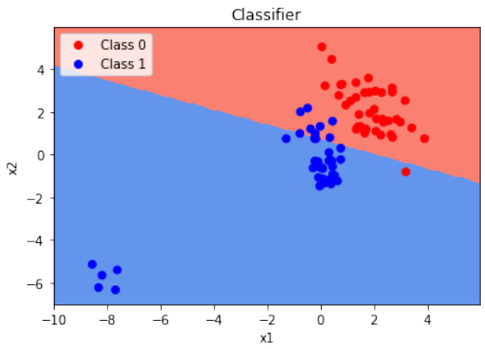
  `x_t[i]` $\leftarrow$ `argmin[i](x_t)`, (1)

  where `f` is the function to be minimized, `argmin` is an array of the `argmin_xi` functions for each coordinate, and `x_0` is the starting point (initialization).

Starting at `x_0`=$(5, 10, 5)$, what is the first coordinate `x_t[0]` of the minimizer coordinate descent converges to? You can round the coordinate to one decimal after the point.



(a)



(b)

3. (10 points) Consider the two-classes dataset shown in the left Figure (a). The data has two features (x_1, x_2). Suppose that we train a certain linear classifier and get the following the decision boundary: $y = -0.078x_1 - 0.227x_2 + 0.165$

   The decision boundary and predictions of the training data are also shown in the right Figure (b) as background colors.

   We would like to evaluate the performance of that classifier on the following validation data. Therefore:

   - Calculate the confusion matrix for the given classifier on the following validation data.

ID	x_1	x_2	y
1	3	1	0
2	-2	2	1
3	2	2	0
4	-1.5	2	1
5	1.5	3.5	0
6	-2.5	2.5	1
7	-2	0	1
8	2	2.5	0

   From the figures above and your evaluation, which of the following statements is **correct**?

   A. The training data is not linearly separable.

   B. This classifier results from training linear SVM on the training data.

   C. All the samples of class 0 in the validation data is correctly classified.

   D. 50% of the samples of class 1 in the validation data is misclassified as class 0.

   In the following two exercises, you are going two work with *Heart Disease* dataset. You can find this dataset in the Assignment .zip file split into two *.csv files*: *data/heart_train_data.csv* and *data/heart_validate_data.csv*. This dataset is taken from the UCI Machine Learning repository. However, this is a modified version in which a subset of the original features are only considered. We consider three features. The description of the features and the target is as follows:

   1. cp (Chest Pain Type): [0: Typical Angina, 1: Atypical Angina, 2: Non-Anginal Pain, 3: Asymptomatic]
   2. exang (Exercise Induced Angina): [1 = yes, 0 = no]
   3. thal (Thallium heart scan): [1 = normal, 2 = fixed defect, 3 = reversible defect]
   4. target: [0 = disease, 1 = no disease]

4. (20 points) We would like to build a Naive Bayes classifier to predict whether a new patient has a heart disease or not. To this end:

   - Import the data in Python and explore it a bit.

- Estimate the class probability $P(y) \triangleq Pr\{Y = y\}$.
- Estimate the conditional probability $P(x_d \mid y) \triangleq Pr\{X_d = x_d \mid Y = y\}$ which is the probability of feature x_d given class y.
- Implement a function that takes an input and return the corresponding prediction based on the estimations calculated in the previous two steps.

Based on your implementation, which of the following statements is **correct**?

A. The probability of having a fixed defect thallium heart scan given that the patient has heart disease equals to 0.4.

B. The Naive Bayes reaches 85% accuracy on the validation set.

C. 60% of the people in the training data suffer from heart disease.

D. A patient that has atypical angina, exercise induced angina, and a fixed defect thallium heart scan will be diagnosed as not having a heart disease by the Naive Bayes classifier.

5. (20 points) Now using the same dataset, we would like to diagnose the patients using decision tree classifier. Therefore:

- Use the CART algorithm explained in the lecture to implement the decision tree.
- For the cost: use entropy as impurity measure.
- Stopping criterion: stop at tree depth 3 (8 leaves).
- During the CART algorithm, consider all One-vs-All decisions when learning decision nodes.

Based on your implementation, which of the following statements is **correct**?

A. The decision tree reach 75% accuracy on the validation set.

B. All leaves nodes of the tree have impurity equals zero.

C. The decision of the root of the tree is based on whether chest pain type is typical angina or not ($cp == 0$).

D. A patient that has atypical angina, exercise induced angina, and a fixed defect thallium heart scan will be diagnosed as having a heart disease by the decision tree classifier.

6. (20 points) **Open Question**. In this exercise, we will explore SVM kernels on a toy dataset using scikit-learn. In Assignment .zip file, you can find *data/toy_dataset.csv* and Python notebook (SVM.ipynb) with a startup code. We would like to study the behavior of each kernel and check underfitting and overfitting. Therefore, import the data in Python and explore it. Then:

(A) (10 points) Compare different kernels

(i) Fit SVM with linear, polynomial and RBF kernels with default parameter values. Use *SVC* function in the svm module of scikit-learn.

(ii) Plot the decision boundary for each kernel with the helper function given in the notebook.

(iii) Interpret the plots and compare the behavior of the three kernels.

(B) (10 points) Now, we will optimize the two hyperparameters $'C'$ and gamma $'\gamma'$ of the $'rbf'$ kernel using with *GridSearchCV* from the model_selection module of scikit-learn. Check the documentation of *GridSearchCV*.

(i) Create a grid with the following values:
$'gamma'$: [1e-4, 1e-3, 1e-2, 1e-1, 1, 2],
$'C'$: [1e-2, 1e-1, 1, 2, 5, 10]

(ii) Use *GridSearchCV* with *SVC*(kernel='rbf') as classifier, and 3-fold-cross-validation (cv) [1].

[1] In k-fold-cross validation, the data is split into k parts (folds). We perform the training procedure k times. Each time, we take one fold as the validation set and the remaining $k-1$ folds as the training set. We fit the model on the training set and compute the validation score on the chosen fold (part). After computing k validation scores, we take the average.

(iii) Plot a *heatmap* of the results (average validation scores) using the provided helper function.

(iv) Interpret the *heatmap*. Analyze the effect of different values of hyperparameters. Does any combination of C and γ leads to underfitting or overfitting?

(v) Report the accuracy of the best model you get. State the hyperparameters used.