جامعة محمد بن زايد
للذكاء الاصطناعي
MOHAMED BIN ZAYED UNIVERSITY
OF ARTIFICIAL INTELLIGENCE

AI701: Foundations of Artificial Intelligence
Fall 2024

Assignment-01

Instructions:

- Group Assignment. Maximum number of students per group: 3.
- This assignment has 5 sections and carries 15 marks.
- The deadline to submit the assignment is by the end of October 15, 2024 (23:59 UAE time).
- Assignment deliverables: four completed Jupyter notebooks and a report. All the required material should be zipped in one folder. Only one submission per group is required.
- For the report, you can include the following items: Answers to questions explicitly mentioned in the Jupyter notebook, output in the form of tables/graphs/plots, evaluation results, training time, and interpretation of the result. You can use any format of your choice. The report is expected to be within 4 pages.

---

1 Linear Regression (3 points)

1.1 Mean Squared Error (1 pts)

Task 1: In the provided notebook, write a function to compute the mean squared error of a given line for the Iris dataset.

1.2 Computing the analytic solution (1 pts)

The best coefficients a and b that minimise the mean squared error (MSE) can be found analytically, using a bit of calculus.
In particular, we set the gradient of the MSE loss function to 0 in order to obtain the least squares estimate for a and b. For MSE denoted by $\mathcal{L}(a, b)$, setting $\frac{\partial \mathcal{L}}{\partial a} := 0$ and $\frac{\partial \mathcal{L}}{\partial b} := 0$, we obtain the least squares estimate

$$a = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

$$b = \bar{y} - a\bar{x}$$

where $\bar{x}$ is the mean value of $[x_0, \ldots, x_{N-1}]$ and so on.
Task 2: Use this solution to compute the least squares estimate for the Iris dataset directly.
Task 3: Predict petal widths for new flowers

1.3 Higher dimensional input features (1 pts)

Task 4: Fit a linear regression model to a dataset with higher dimensional input features. We choose to model the housing price as a function of the property's location and its size, using a synthetic dataset. Is the plane a good fit to the data? If not, what is the reason?

2 Logistic Regression (5 points)

2.1 Decision boundary (2 pts)

* The decision boundary is a line which separates out the data points from different clusters.
* If we have a data point $\mathbf{x}$ that we want to classify, where $\mathbf{x} = \{x_1, x_2\}$ – a 2D feature vector. Then the predicted class depends on which side of the line $\mathbf{x}$ lies.
* We represent the decision boundary of the form: $f(\mathbf{x}) = ax_1 + bx_2 + c = 0$.
* This allows us to easily compute which side of the line the point lies from looking at the sign of $f(\mathbf{x}) = ax_1 + bx_2 + c$.
* A positive sign indicates that the data point $\mathbf{x}$ belongs to the blue class, and a negative sign indicates red.
Task 1: Implement the Sigmoid function
Task 2: Implement a function which gives you the predictive probability $p(\mathbf{x})$ of 2D data points $\mathbf{x} = (x_1, x_2)$

2.2 Learning Objective: Binary Cross-Entropy Loss (1 pts)

* We want to find a way to automatically infer the decision boundary.
* We use the Binary Cross Entropy loss to work out the decision boundary.

$$\mathcal{L} = \sum_{i=1}^{N} -y_i \log(p(\mathbf{x}_i)) - (1 - y_i)\log(1 - p(\mathbf{x}_i))$$

Task 3: Implement the Binary cross entropy (BCE) loss.

2.3 Optimization with Gradient Descent (GD) (2 pts)

* Gradient descent is a way to minimize (or maximize) a function, similar to walking down a hill.
* If you want to walk down a hill from a random point, you'd choose a direction which points down, and then take a step.
* That's what we do a lot in machine learning (literally, this is what everyone uses all the time), but rather than take a physical step, we just move the parameters by a certain amount which we call a step.
* For GD to be useful, we need to calculate the gradient of our loss function.
* The gradients for BCE are:

$$\frac{\partial \mathcal{L}}{\partial a} = \sum_{i=1}^{N} (\sigma(ax_1^i + bx_2^i + c) - y^i)x_1^i$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{N} (\sigma(ax_1^i + bx_2^i + c) - y^i)x_2^i$$

$$\frac{\partial \mathcal{L}}{\partial c} = \sum_{i=1}^{N} (\sigma(ax_1^i + bx_2^i + c) - y^i)$$

Note that $(\mathbf{x}^i, y^i)$ refer to the i-th data points in our observations.
Task 4: Compute the gradient.
Task 5: Implement the gradient descent algorithm.
Task 6: Predict output on test set and evaluate.

3 Support Vector Machines (2 points)

The goal is to determine whether a tweet was written by a Democratic or Republican politician, using just the text of the tweet.
Working with text data The features for an SVM can't be words or whole tweets. We need a numerical representation for the words in the texts. One method is to transform the text into TF-IDF vectors.
It will take the tweets, tokenise them into words (using a special tokeniser that knows how best to split up tweets), remove stop words then it will create a sparse matrix representation of all the tweets.
Task 1: Train a linear kernel SVM
Task 2: Use grid search to find the best model

4 Classification (3 points)

The target of this task is to predict the risk of credit default. The dataset is provided in ‘default.xls’. It contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan. Information on all the included variables is available here. No reference notebook is provided for this task.
Task 1.1: Create a feature matrix X and a target vector y from data. The target to be predicted is given in the last column.
In the following, you are required to evaluate your models with a cross-validation score. Make sure your splits are fixed across the experiments.
Task 1.2: Fit a Decision Tree classifier and evaluate.
Task 1.3: Transform categorical columns into one-hot encoded features. Fit and test a decision tree classifier with the updated features.
Task 1.4: Use grid search to find best random forest classifier parameters min_samples_split and max_depth.
Can you suggest other feature pre-processing methods that would further improve the result?

5 Gaussian Naive Bayes Classifier (2 points)

Assume that Y is Boolean and $X = \{X_1, \ldots, X_n\}$ is a vector of continuous variables. Taking into consideration the assumptions of the Gaussian Naive Bayes classifier, show that $P(Y|X)$ is given by:

$$P(Y = 1|X) = \frac{1}{1 + \exp\left(w_0 + \sum_{i=1}^{n} w_i X_i\right)}$$

and

$$P(Y = 0|X) = \frac{\exp\left(w_0 + \sum_{i=1}^{n} w_i X_i\right)}{1 + \exp\left(w_0 + \sum_{i=1}^{n} w_i X_i\right)}$$