GRA4150
AI - Technologies and Applications
EXERCISE 1

Silvia Lavagnini

January 24, 2023

1 Exercise

Let

$$A = \begin{bmatrix} 1 & -2 \\ 3 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix}, \quad \text{and} \quad v = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

and calculate the following expression, first by hand and then in Python:

1. $A + B$
2. $5A$
3. $(A + B)v$
4. $A^\top$
5. $v^\top v$
6. $vv^\top$

2 Exercise

1. Find the line that passes through the points $(3, 3)$ and $(2, 1)$;
2. Draw it, first by hand and then in Python.

3 Exercise

We have seen the perceptron model.

1. Write down which function we want to learn when we use the perceptron model rule;
2. Which advantage/disadvantage does this function have?
3. List which are the model parameters and which are the hyperparameters for the perceptron model;
4. What is the specific role/meaning of the model parameters for the perceptron model? Specify for each of them.
5. What about hyperparameters: what changes when we tune them? Specify for each of them.

4 Exercise (optional)

When we use gradient descent in the Adaline learning rule (next lecture), we want to minimize a loss function. In particular, the Mean Squared Error loss function:

$$L(\mathbf{w}, b) = \frac{1}{2n} \sum_{i=1}^{n} \left(y^{(i)} - \sigma(z^{(i)}) \right)^2$$

where $\sigma(z^{(i)}) = \hat{y}^{(i)}$ is the i-th prediction with the Adaline learning rule.

1. Describe in words what is the role of the loss function and what is the idea behind gradient descent.

2. Assume for a moment that $n = 1$, then we get

$$L_1(\mathbf{w}, b) = \frac{1}{2} \left(y - \sigma(z) \right)^2 .$$

   Compute the following derivatives:

$$\frac{\partial L_1(\mathbf{w}, b)}{\partial w_j} \quad \text{and} \quad \frac{\partial L_1(\mathbf{w}, b)}{\partial b}.$$

3. Go now back to the general formulation for L and compute the following derivatives:

$$\frac{\partial L(\mathbf{w}, b)}{\partial w_j} \quad \text{and} \quad \frac{\partial L(\mathbf{w}, b)}{\partial b}.$$

5 Exercise

Consider the colors dataset that we used in class.

1. Rewrite the function `update_fn` so that:

   (a) All the operations are in vectortorized form, hence the function works for a generic $\mathbf{w}$, namely the vector $\mathbf{w}$ can have more than 2 components;

   (b) You remove the if-else statement and replace it with the function `np.where` (check the documentation!);

   (c) You take track of the number of misclassified element at each iteration.

2. Try to play a bit around with different starting configurations (`w_new`, `b_new`). Do you always reach convergence? Do you need to increase the number of iterations to reach convergence?

3. Try to play a bit around with different values for the learning rate `eta`. Do you always reach convergence? Do you need to increase the number of iterations to reach convergence?

4. Define a function that computes the perceptron learning rule found above (you can call it `sigma`): it takes as parameters `w_new` and `b_new` found above, and takes `x` as input (use the lambda format). Now add 4 new rows to the dataframe `col_df`: two rows containing observations from Class 1, and 2 rows containing observations from Class 0. Use the function `sigma` to predict the class of these four new observations.

5. (Optional) Repeat the steps above with the "extended" dataset, where we also consider the white pigment.

6 Exercise

Consider now the Iris dataset. This contains the measurements of 150 Iris flowers from three different species (setosa, versicolor and virginica). However, since the perceptron model is a binary classifier, we restrict our attention to two flower classes, setosa and versicolor. Moreover, we consider only two features, namely sepal length (first column) and petal length (third column).

1. Initialize the model parameters and plot the initial boundary;
2. Initialize the hyperparameters and run the optimization;
3. Plot the final boundary.
4. Repeat the points above by adding a third feature to the dataset (for example, the petal width – fourth column);
5. (Optional) Apply the perceptron rule to the Iris dataset with all the four features available (include also the one that you left out at the previous point). Do you need to change to code to do that? What about plotting th results? What ca we do if we can't make a plot with the observations and the boundary?

7 Exercise (Optional)

Compare your code with the object-oriented perceptron code given during Lecture 2.

1. Are there differences in how the perceptron learning rule is applied?
2. What are the advantages (if any) of such formulation?
3. Select as in Exercise 6.4 two classes and three features from the Iris dataset. Call the class `Perceptron` three times with three different configurations of hyperparameters (hence you create three objects). Train the three perceptrons and compare the results, in particular the behavior of the "number of updates" in the three different cases.