PW-08

{dorian.magnin, noemien.kocher}@master.hes-so.ch

1. The Perceptron and the Delta rule

1_activation_function.ipynb

- Observe the shape of the different activation functions proposed.

- Observe the effects of modifying the weight. How the shape of the function changes? How the first derivative changes?

Pour la fonction linéaire, le poids change la valeur de la pente.
Pour la sigmoïde, le poids “aplatit les fonctions jusqu’à les “renverser” (symétrie horizontale)
Pour la tangente hyperbolique, le poids produit un effet similaire à la sigmoïde
Pour le reste, nous sommes passé à Python 2.7 et le widget javascript ne fonctionne plus...

- Implement the activation function of a rectified Linear Unit (ReLU)

```
def redLu(neta):
    '''Rectified Linear Unit function'''
    output = 0
    d_output = 0
    if(neta >= 0):
        output = neta
        d_output = 1
    return (output, d_output)
```

Impossible d’afficher le graph!

2_perceptron.ipynb

- Use the sliders to change the weights of the perceptron and observe the effects on its output
- Select different activation functions and observe the output of the perceptron for different weight configurations

3_MLP.ipynb

- Use the sliders to change the values of the connection weights and biases, and observe the resulting changes in the MLP output
- Change the activation function and observe the changes

4_delta_rule.ipynb

You are free to modify the learning rate (ALPHA) and the number of iterations (NUMBER_OF_EPOCHS).
Try different 2D classification problems and observe the behaviour of the algorithm in terms of:

- Learning rate needed
- Convergence speed
- Oscillations

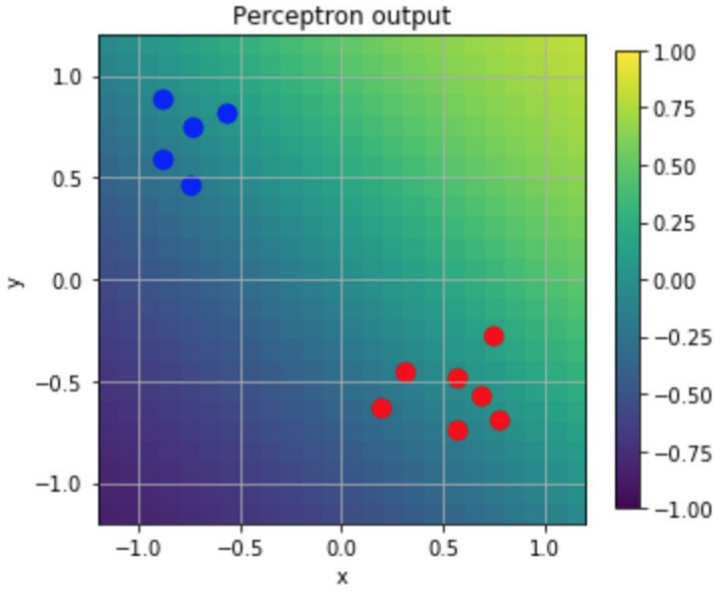
Bare in mind that, in the current implementation, the parameters (weights and bias) are initialized randomly every time you launch the cell.

Create some datasets as it is shown, and perform the following tests:

1. What happens if the boundaries between both classes are well defined?

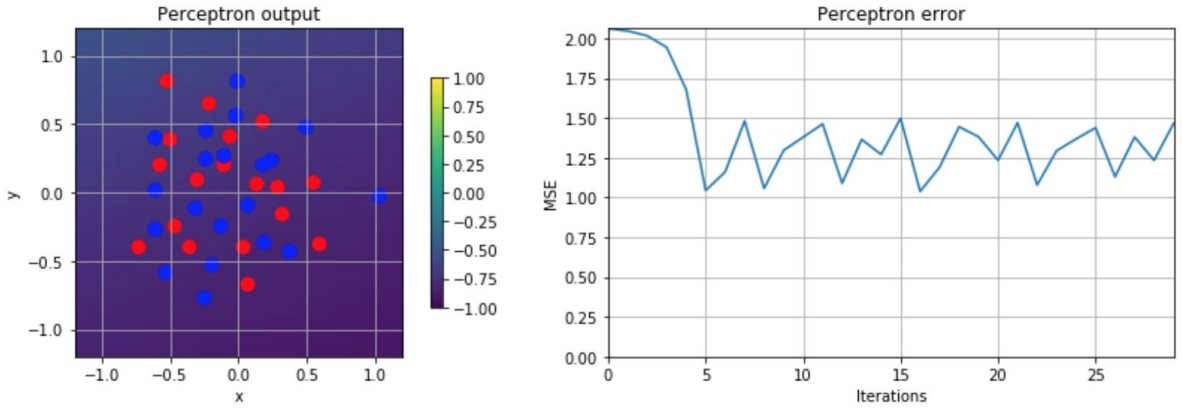
La séparation se fait bien, la frontière est claire. L'erreur diminue rapidement.

```
TypeError: unsupported operand type(s) for *: 'NoneType' and 'float'
```



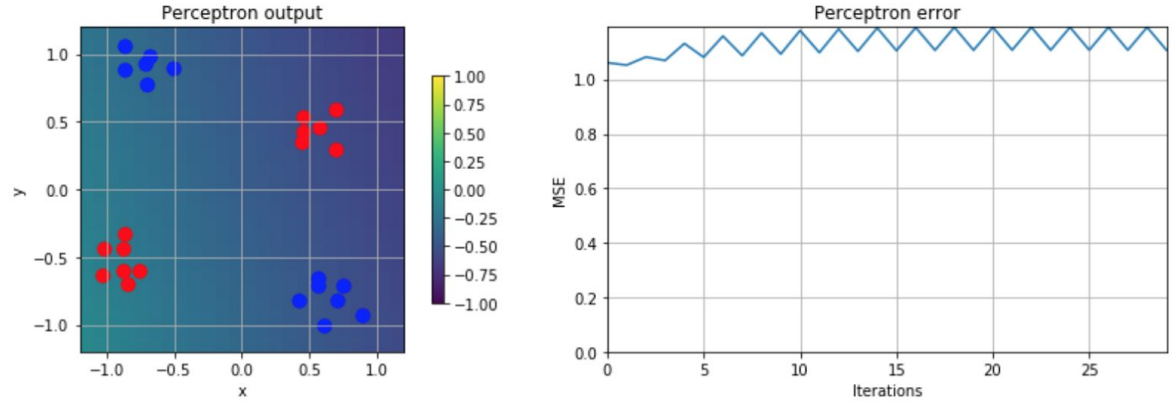
2. What happens if the classes overlap? What could you say about oscillations in the error signal?

La séparation n'est pas bonne et le signal d'erreur oscille sans diminuer.



3. What happens if it is not possible to separate the classes with a single line? What could you say about local minima?

La fonction essaie tant bien que mal de trouver une séparation mais celle-ci change constamment. L'erreur ne diminue pas mais remonte.



2. Backpropagation

You are free to modify the learning rate (ALPHA) and the number of iterations (NUMBER_OF_EPOCHS).

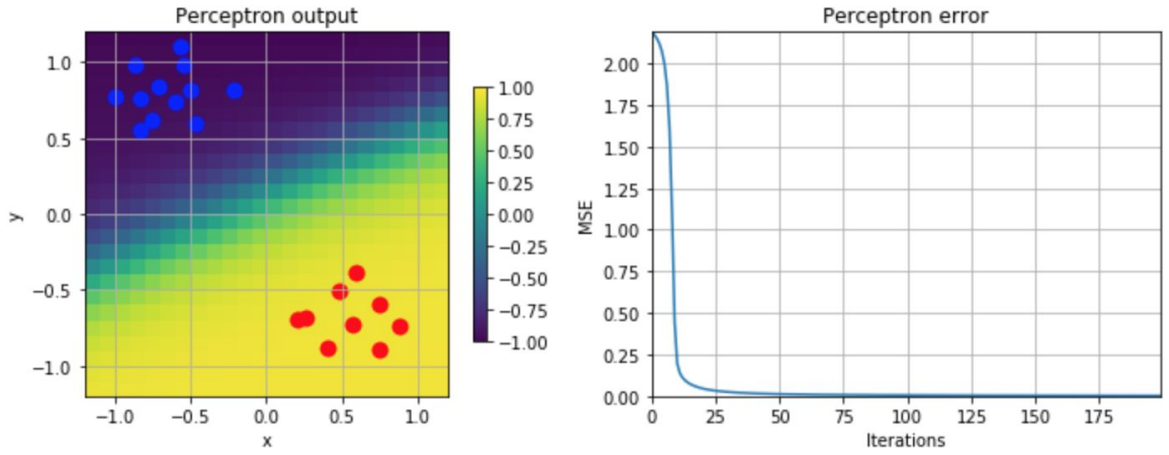
Try different 2D classification problems and observe the behaviour of the algorithm in terms of:

- Learning rate needed
- Convergence speed
- Oscillations

Bare in mind that, in the current implementation, the parameters (weights and bias) are initialized randomly every time you launch the cell

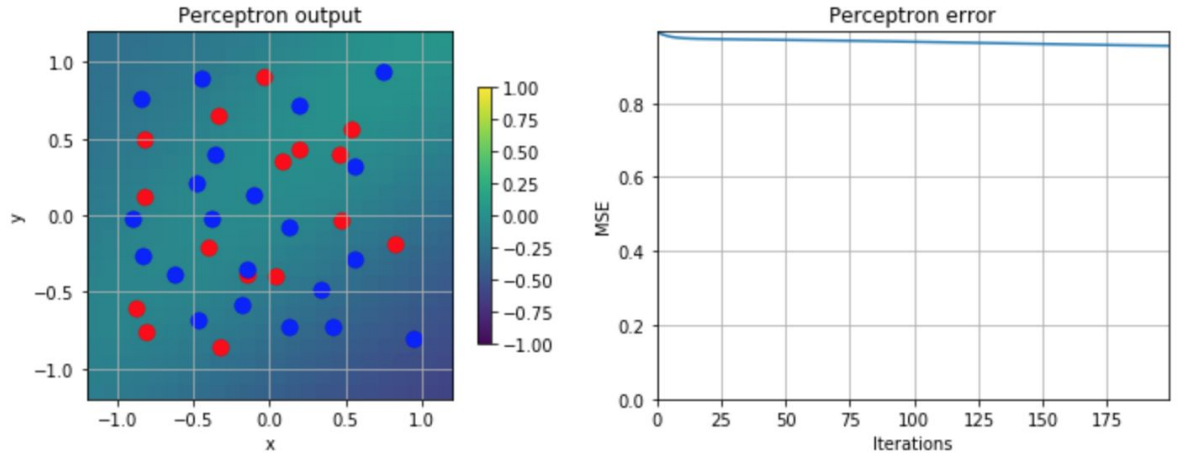
1. What happens if the boundaries between both classes are well defined?

La séparation se fait bien, la frontière est claire. L'erreur diminue rapidement.



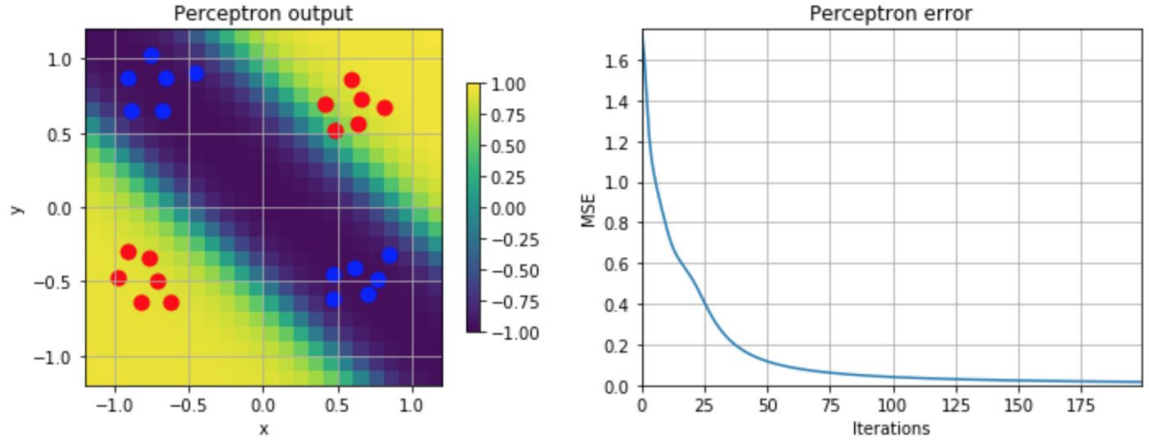
2. What happens if the classes overlap? What could you say about oscillations in the error signal?

La séparation n'est pas bonne mais le signal d'erreur n'oscille pas et a tendance à diminuer puis se stabiliser.



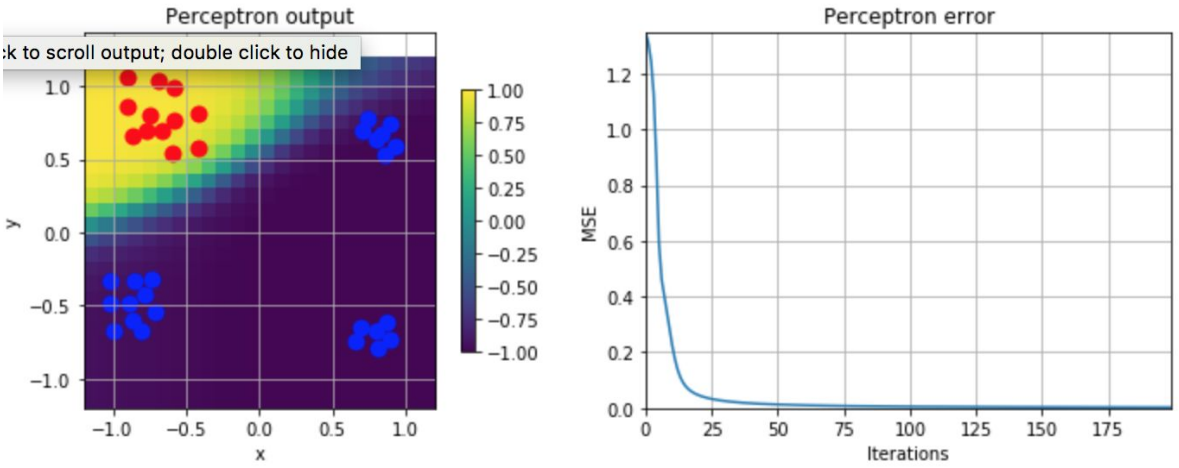
3. What happens if it is not possible to separate the classes with a single line? What could you say about local minima?

Le système a réussi à classifier correctement les groupes. L'erreur n'oscille pas et diminue jusqu'à se stabiliser.



4. What happens if the points of one of the classes are separated in subgroups (blobs)?

Le système classifie correctement tous les groupes et l'erreur tombe à 0.



Source code with momentum

```
def fit(self, X, y, learning_rate=0.1, momentum=0.5, epochs=100):
        """
        Online learning.
        :param X: Input data or features
        :param y: Input targets
        :param learning_rate: parameters defining the speed of learning
        :param epochs: number of times the dataset is presented to the network for learning
        """
        X = np.atleast_2d(X)
        temp = np.ones([X.shape[0], X.shape[1]+1])
        temp[:, 0:-1] = X  # adding the bias unit to the input layer
        X = temp
        y = np.array(y)
        to_return = np.zeros(epochs)

        for k in range(epochs):
            temp = np.zeros(X.shape[0])
            old_deltas = [0, 0]
            for it in range(X.shape[0]):
                i = np.random.randint(X.shape[0])
                a = [X[i]]

                for l in range(len(self.weights)):
                    a.append(self.activation(np.dot(a[l], self.weights[l])))
                error = y[i] - a[-1]
                temp[it] = error ** 2
                deltas = [error * self.activation_deriv(a[-1])]
```

```
            for l in range(len(a) - 2, 0, -1):
                deltas.append(deltas[-1].dot(self.weights[l].T) *
                                                    self.activation_deriv(a[l]))
            deltas.reverse()

            for i in range(len(self.weights)):
                layer = np.atleast_2d(a[i])
                delta = np.atleast_2d(deltas[i])
                old_delta = np.atleast_2d(old_deltas[i])
                self.weights[i] += learning_rate * layer.T.dot(delta) +
                                            momentum * layer.T.dot(old_delta)

            old_deltas = deltas
        to_return[k] = np.mean(temp)
    return to_return
```

- Run notebooks 7 and 8, provide the final plots MSE vs spread and comment the difference between results

Nous n'avons pas pu lancer les notebook à cause des erreurs suivantes:
(notebook n°7)

```
Testing dataset with variance: 0.4
.
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-8-25b36b01ea19> in <module>()
     13             sys.stdout.write('.')
     14             nn = mlp.MLP([2,N_NEURONS,1], 'tanh')
---> 15             input_data = dataset_train[:,0:nn.n_inputs]
     16             output_data = dataset_train[:,nn.n_inputs:(nn.n_inputs+nn.n_outputs)]
     17             input_data_test = dataset_test[:,0:nn.n_inputs]

AttributeError: MLP instance has no attribute 'n_inputs'
```

(notebook n°8)

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-9-0f0e27a52d0f> in <module>()
     16                                                learning_rate=LEARNING_RATE,
     17                                                momentum=MOMENTUM,
---> 18                                                epochs=EPOCHS)
     19         MSE_train[p,d] = temp1
     20         MSE_test[p,d] = temp2

<ipython-input-7-30992d31fec2> in k_fold_cross_validation(mlp, dataset, K, learning_rate, mom
entum, epochs)
     13         dataset_test = parts[k]
     14
---> 15         input_data = dataset_train[:,0:nn.n_inputs]
     16         output_data = dataset_train[:,nn.n_inputs:(nn.n_inputs+nn.n_outputs)]
     17         input_data_test = dataset_test[:,0:nn.n_inputs]

AttributeError: MLP instance has no attribute 'n_inputs'
```

- Run notebook 9 for three different spread values (e.g., 0.3, 0.5 and 0.7), describe the final model chosen and justify your selection (e.g., based on the plots of MSE vs parameters)

Nous n'avons pas pu lancer le notebook à cause de l'erreur suivante:

```
Testing 2 neurons...

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-220aa792defd> in <module>()
      9         MSE[i_h, i, :] = nn.fit(dataset[:,0:2], dataset[:,2:4],
     10                                 learning_rate=LEARNING_RATE,
---> 11                                 momentum=MOMENTUM, epochs=EPOCHS)

/Users/nkcr/Documents/HES-SO MSE/2017-2018/T-MachLe/TPs/08/data/mlp_backprop_momentum.pyc in
fit(self, X, y, learning_rate, momentum, epochs)
     64                     a.append(self.activation(np.dot(a[l], self.weights[l])))
     65                 error = y[i] - a[-1]
---> 66                 temp[it] = error ** 2
     67                 deltas = [error * self.activation_deriv(a[-1])]
     68

ValueError: setting an array element with a sequence.
```

Nous avons rencontré plein d'erreurs tout au long des exercices. Malgré notre bonne volonté, nous n'avons pas pu toute les corriger et répondre à toutes les questions.