

Start from a feature matrix and for each row you obtain a vector

You compute the product between the input and the weight. You apply threshold function (FI finciton) that have a range. Threshold function introduce non-linearity because from continuous values you obtain a discrete value.

The output y^ is compare with the actual (true) label and if the actual and if they are the same you don’t update anything because the prediction is correct (no need to adjust skills) the update is done when there is an error (predicted values differs from the actual value)

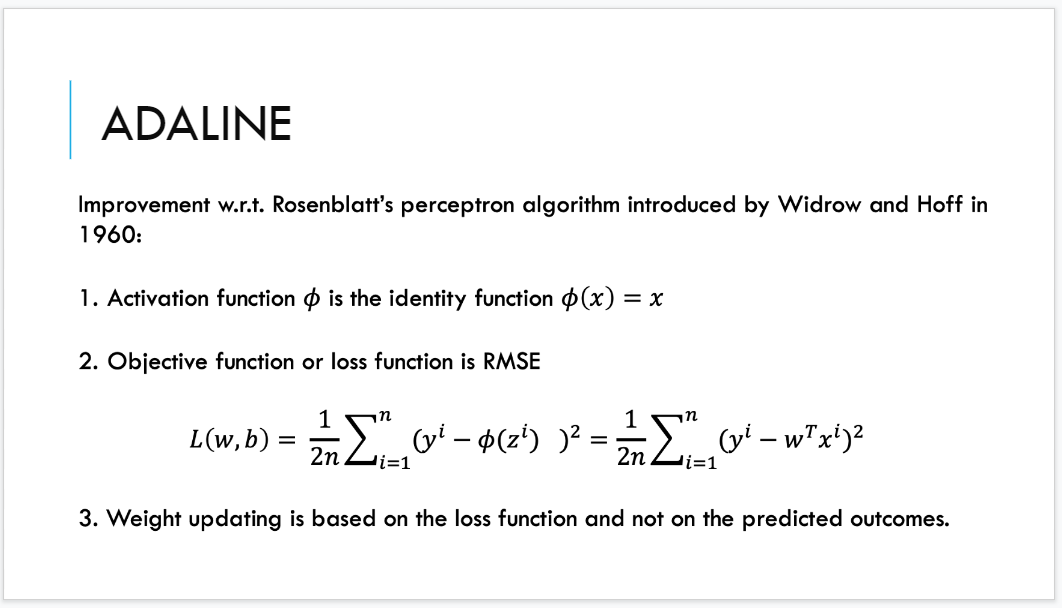
(check update rule) is a pull/push on the line moving the line to the right or left or up or down by an index eta (η) and the same for bias.

The compisiton between the weight and bias factor is the same (?)

**Epoch**: hyperparameters that corresponds to the number of passes (how many time I see the entire data)

Hyperparameter: you have to choose before running fit for each xi means you scan all the rows of the data

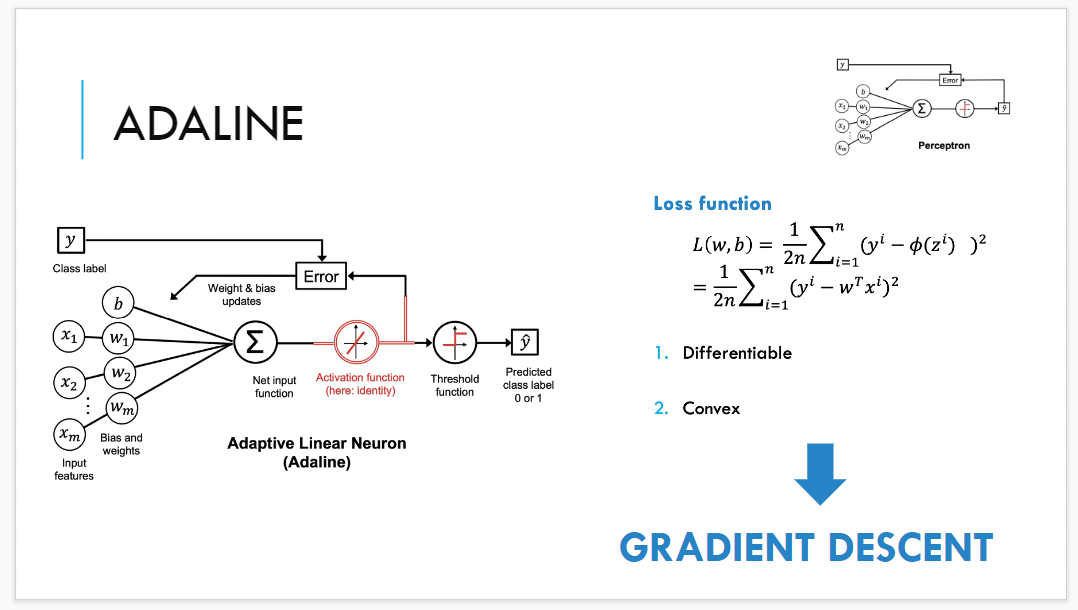
two paratmehrs η and the number of epoch (number of interaction)



Instead of using a threshold function given by the step function we use the activiaiton function the same of identity function. We use the net input function the output of the algorithm.

Second idea: don't use the loss function but use instead use the root mean square error.

acting on the learning algorithm and the architutecture of the model



The first part is the same. Then I put an activation function (identifity)

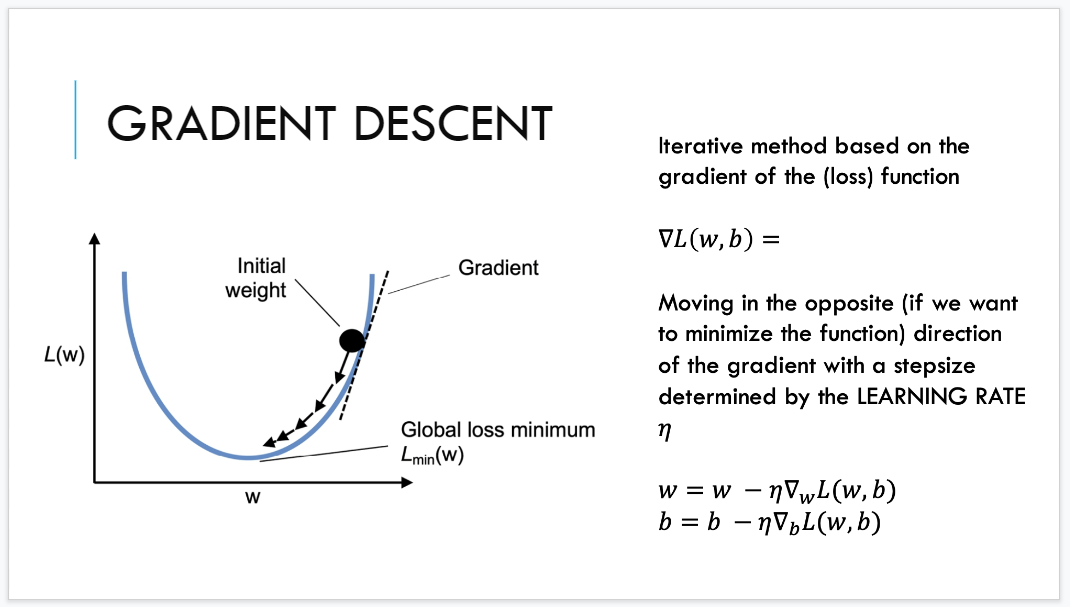
take the net input function as my prediction and to learn we don’t want to use the discrete valute but instead the discrete value

In the perceptron you use discrete values to update thew weight. In adaline you use the continuous value to update the weight.

Gradient descent optimization technique.

The threshold function don’t disappear but used at the end of the architecture becuase use to predict the class. To take the activation function

If I remove the threshold function I get regression, same approach of the example of GDP and LS (life satisfaction)

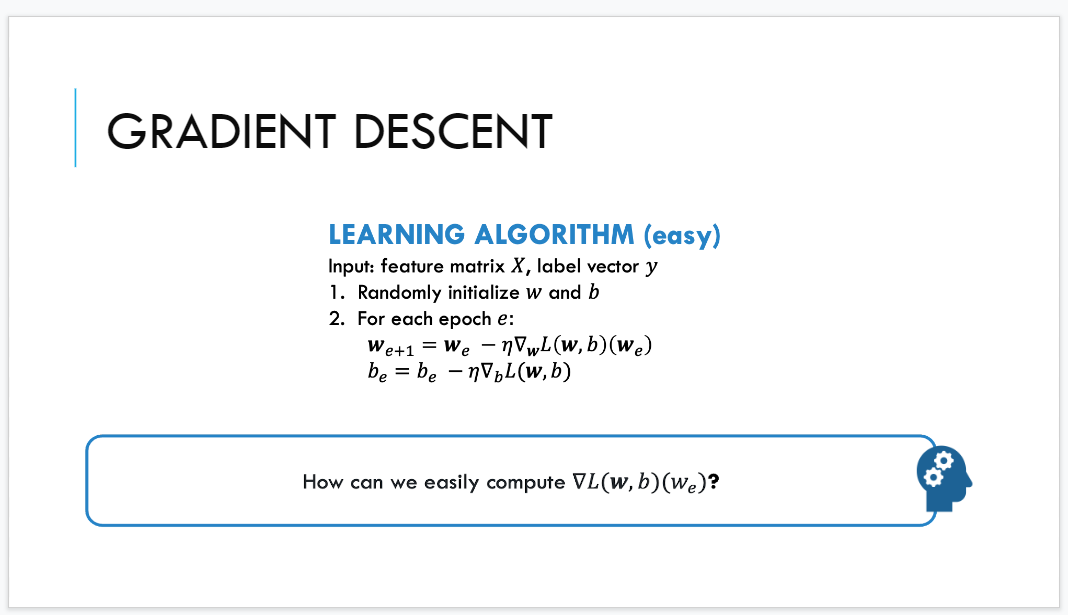


Reshaping of the things. Once you a differenetinal function you can compute the partial derivative for each direction.

The gradient show the best direction to maximise a function.

Gradient says the direction give is to maximise. If I follow it I go to the best direction to maximise the function. But if you turn and move opposite direction you move to the minimum.

- (negative) sign for moving in opposite direction of the gradient respect to w and we move using a discount given by the learning rate η

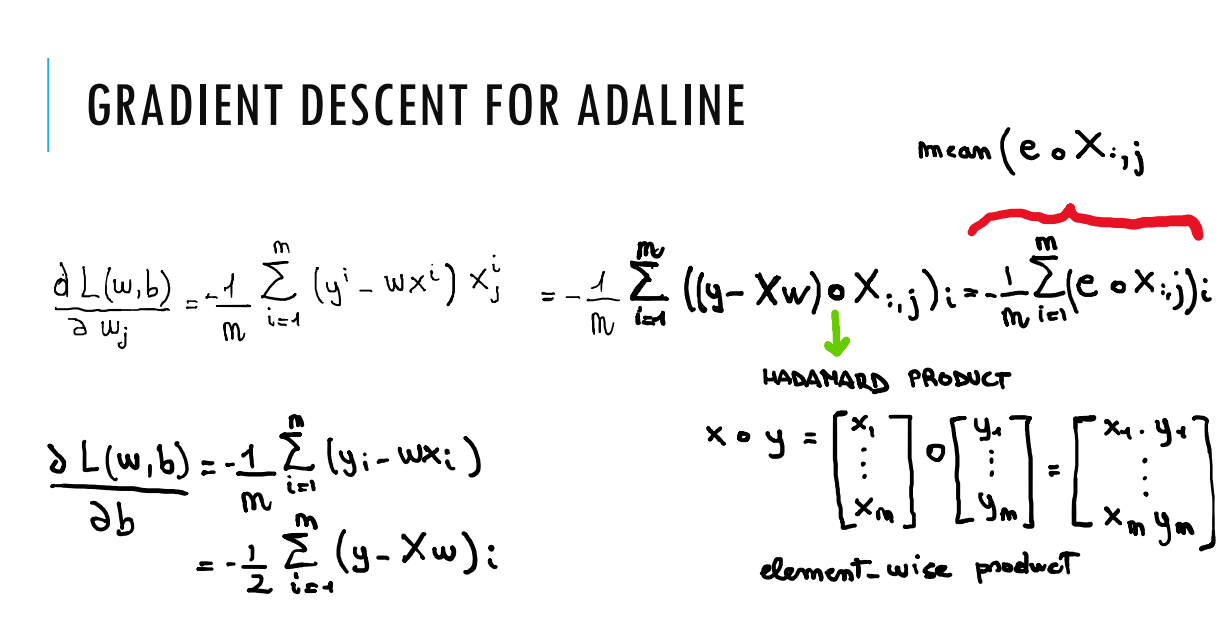


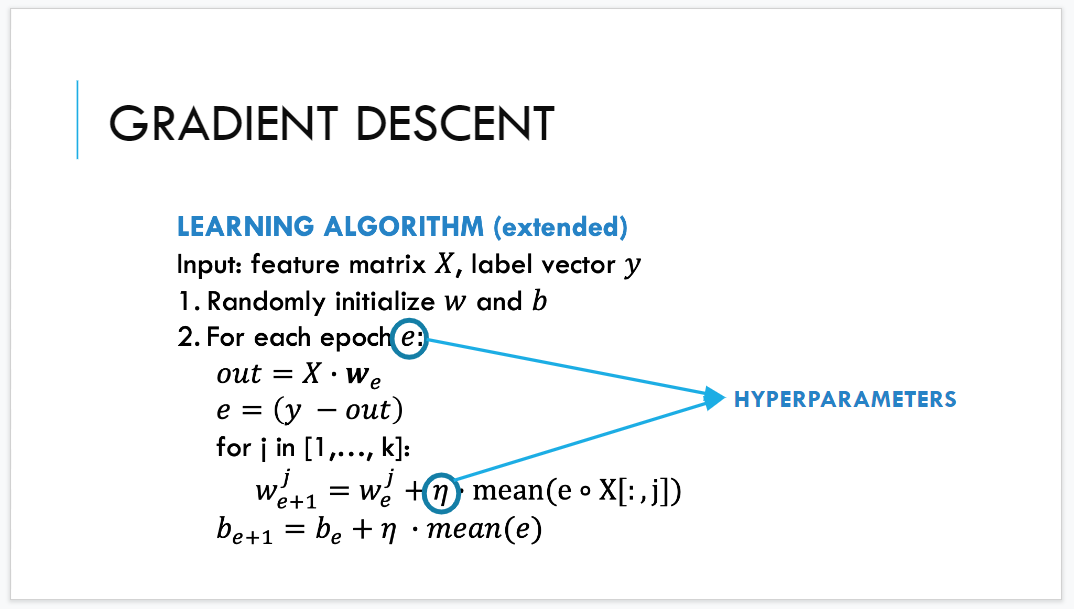
random initialize the weight and balance factor. For each epoch compute +compute

We have only one for that is used for each epoch. (no for each xi) this is the difference between full batch and online.

We are using all the training set to update the weight. take the entire training set and update. while in perception we take one row and update the weight.

How can we easily compute the gradient? make the depends of the computation of teh gradient explicin to the whole dataset.





Out placehold to say X \* we (ector of the weights) w. the error is the true lavel minus the oupot of my model

for each index in the gradient vector I update the element in position j. the origins weight + (because we have a minus before)

(the k is the value m)

hyperparsmters are the same as in perceptron. numer of epch and learning rate is the same.