PERCEPTRON / ARTIFICIAL NEURON

3 inputs : (1) learning rate
(2) n_iter (epoch)
(3) random state - for initialisatn.

fit method

2 inputs : (1) X - can be multiple dimensionl
(2) y - label

algorithm :
1. Make a random number generator using np.random.RandomState
2. Initialise w, b randomly.
   ↳ w is initialised using normal distribution w/ mean 0 and s.d 0.01
   ↳ b is initialised as 0
3. Iterate through epoch, take note of error each epoch
4. Iterate through training data
5. Update w, b
6. Increment error / epoch
7. Append epoch's error into a list of errors attribute of perceptron
8. return the model (self)

net_input & predict

net_input takes in X and returns the dot product output

predict takes in X and return 1/0 based on output of net_input.

np.where (condition, value_if_true, value_if_false) ternary operator!

some notes

- since b is used, the treshold is always 0!
- some impt eqn

$$z = w_1x_1 + w_2x_2 + \dots + w_mx_m + b = \underline{w}^T\underline{x} + b$$

$$\sigma(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

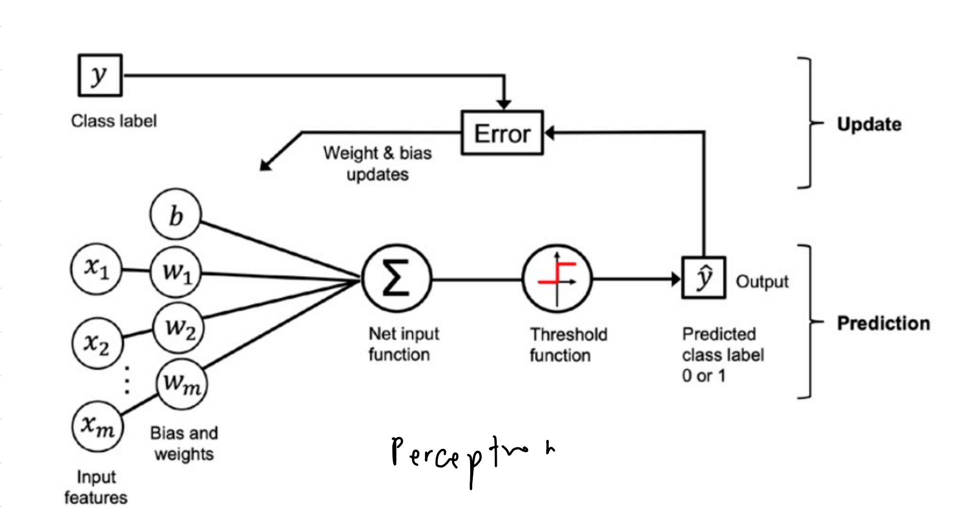
note that $x_j^{(i)}$ means
i^{th} example
j^{th} dimension / feature

$$\Delta w_1 = \alpha\,(y^{(i)} - \hat{y}^{(i)})\,x_1^{(i)}$$

$$\Delta w_2 = \alpha\,(y^{(i)} - \hat{y}^{(i)})\,x_2^{(i)}$$

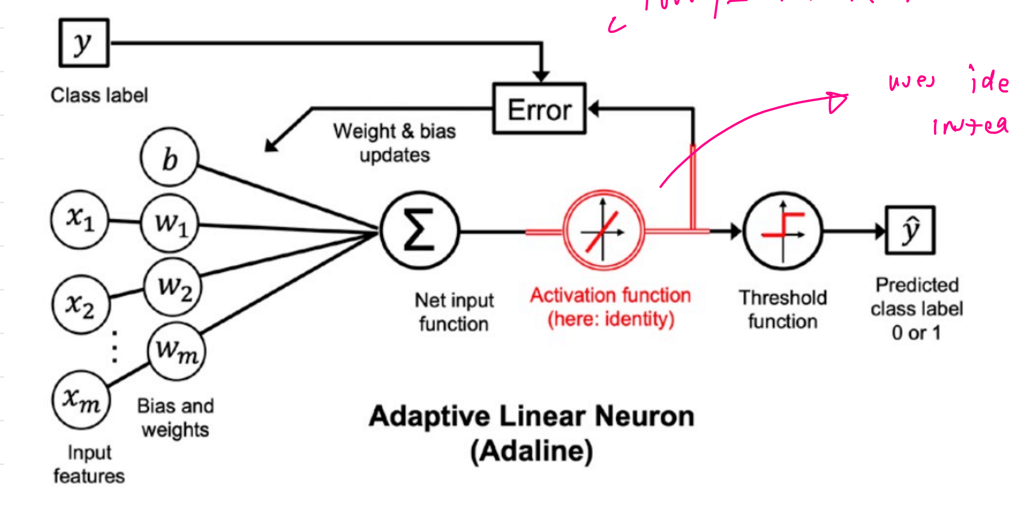
$$\Delta b = \alpha\,(y^{(i)} - \hat{y}^{(i)})$$

PERCEPTRON vs ADALINE



loss fn will be more complicated

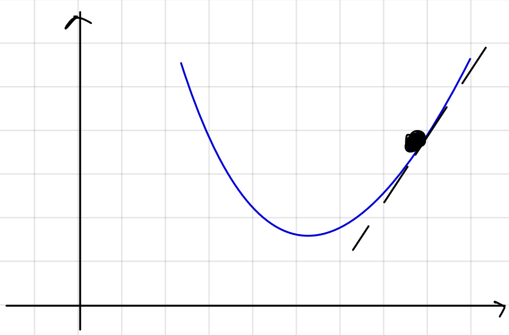
uses identity fn instead of threshold!



minimizing loss fn w/ gradient descent

In Adaline, the loss fn is the mean squared error (MSE)

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



if the $\frac{d}{dx} > 0$, you want to go down ↓

$$\Delta w = -\alpha \nabla_w L(w,b)$$

$$\Delta b = -\alpha \nabla_b L(w,b)$$

$$\frac{\partial L}{\partial w} = -\frac{2}{n}\sum\left(y^{(i)} - \sigma(z^{(i)})\right)x_j^{(i)}$$

$$\frac{\partial L}{\partial b} = -\frac{2}{n}\sum\left(y^{(i)} - \sigma(z^{(i)})\right)$$

∇_w is the same as $\frac{\partial L}{\partial w}$!