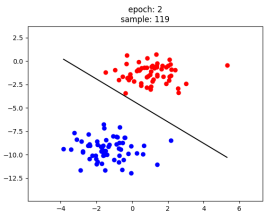
Perceptron Worked Examples

Reading:
Perceptron Materials
Patrick Loeber video / code

Perceptron models can be used for binary classification tasks (classify between 2 classes). They try to separate the two classes.

During training, the model is shown training examples, one at a time, along with the correct (gold) answer. Unlike the models we've seen so far, perceptron models don't count anything (e.g. tokens or occurrences of one token following another). Perceptron models calculate a **weight** for each **feature** in the training data, and a **bias**. Features and gold values are numeric values. Training samples are assigned a gold value of 0 or 1, representing the class membership. Next semester we will learn how to represent text numerically, but for now we'll use simple numeric data as our features.

The plot shows a dataset with 2 features. The first feature is plotted on the x-axis, and the second feature is plotted on the y-axis. Class membership is represented with color (blue points belong to one class, red points the other). The two classes can be separated by a line. If we had 3 features, we could visualize the data in 3 dimensions with a plane separating the classes. (Datasets can have many more features, but it is difficult to visualize anything beyond 3 dimensions.) The weights determine the slope (gradient) of the line, and the bias is the offset of the line. The goal is to find the weights and bias such that the line separates the classes.



Perceptron models need to see the training examples multiple times to get the weights and bias right. In each **epoch** of training, each sample in the training data is seen once. For each sample, the model makes a prediction based on the current weights and bias, and the **activation function**. Then the update value is calculated based on the **learning rate** and whether the prediction was right or not (that is, the difference between the predicted value **y_pred** and the gold value **y**). Then the weights and bias are updated using the update value. These new weights and bias are then used when looking at the next training sample, where they will again be updated. And so on...

When are we done? We're done when the weights and bias do not change for an entire epoch. That is, each training sample was seen, but it was not necessary to update the weights and/or bias for any of them. Sometimes this never happens (for example, when the data cannot be cleanly separated and the model alternates between several not-quite-perfect options). Typically, a fixed number of epochs is used, but if the weights and bias don't change during an earlier epoch, we stop.

In these worked examples, we look at how weights and bias are adjusted during training, and at how predictions are made with a trained model. We start with the prediction task, then look at how the weights and bias are updated during training.

1 Prediction

A prediction is made by appying the activation function to the linear output. The activation function determines if the "neuron" gets "fired" (output is 1), or not (output is 0).

$$\text{linear\_output} = \boldsymbol{X} \cdot \boldsymbol{w} + b$$

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

$$\text{y\_pred} = g(\text{linear\_output})$$

Perceptron Worked Examples

1.1 Prediction Exercise 1

A perceptron model was trained on a 3-feature dataset. The final weights and bias after training are:

$$\boldsymbol{w} = \begin{bmatrix} -4 \\ 2 \\ 3 \end{bmatrix} \qquad bias = 2$$

Calculate the predicted value of the following test data, which contains one sample.

$\boldsymbol{X_{test}} = \begin{bmatrix} 3 & 1 & -5 \end{bmatrix}$

Solution: Apply the activation function (the step function) to the linear output:

$$\begin{aligned} linear\_output &= \boldsymbol{X_{test}} \cdot \boldsymbol{w} + b \\ &= \begin{bmatrix} 3 & 1 & -5 \end{bmatrix} \begin{bmatrix} -4 \\ 2 \\ 3 \end{bmatrix} + 2 \\ &= (3 * -4) + (1 * 2) + (-5 * 3) + 2 \\ &= (-12 + 2 + -15) + 2 \\ &= -25 + 2 \\ &= -23 \end{aligned}$$

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

y_pred $= g(-23) = 0$

1.2 Prediction Exercise 2

Using the same weights and bias as in Exercise 1, calculate the predicted value of the following test data, which contains two samples.

$$\boldsymbol{X_{test}} = \begin{bmatrix} -2 & 2 & 1 \\ 1 & 5 & -1 \end{bmatrix}$$

Solution: Apply the activation function (the step function) to the linear output:

$$\begin{aligned} linear\_output &= \boldsymbol{X_{test}} \cdot \boldsymbol{w} + b \\ &= \begin{bmatrix} -2 & 2 & 1 \\ 1 & 5 & -1 \end{bmatrix} \begin{bmatrix} -4 \\ 2 \\ 3 \end{bmatrix} + 2 \\ &= \begin{bmatrix} (-2 * -4) + (2 * 2) + (1 * 3) \\ (1 * -4) + (5 * 2) + (-1 * 3) \end{bmatrix} + 2 \\ &= \begin{bmatrix} (8 + 4 + 3) \\ (-4 + 10 + -3) \end{bmatrix} + 2 \\ &= \begin{bmatrix} 15 \\ 3 \end{bmatrix} + 2 \\ &= \begin{bmatrix} 17 \\ 5 \end{bmatrix} \end{aligned}$$

y_pred $= g(\begin{bmatrix} 17 \\ 5 \end{bmatrix}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ The model predicts 1 for both samples.

2 Training

During training, we look at each sample in the training data and predict the outcome based on the current weights and bias. Then, if the prediction was wrong, the weights and bias are updated by nudging them in the right direction. How big the update is depends on the **learning rate**.

2.1 Training Exercise

Suppose you are training a perceptron model on a 3-feature dataset. The current weights and bias are:

$$\boldsymbol{w} = \begin{bmatrix} 3 \\ -1 \\ 4 \end{bmatrix} \qquad bias = -0.3$$

Calculate the weights and bias after processing the following training sample **x**. Note that **x** is one row vector in the training data, and y is the gold value for sample **x**. Use a learning rate of .01.

$$\boldsymbol{x} = \begin{bmatrix} 1 & 0 & -2 \end{bmatrix} \qquad y = 1$$

Solution: Apply the inner for-loop in fit() once (see Patrick Loeber video and code)

$$\begin{aligned} linear\_output &= \boldsymbol{x} \cdot \boldsymbol{w} + b \\ &= \begin{bmatrix} 1 & 0 & -2 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ -1 \\ 4 \end{bmatrix} + -.3 \\ &= (1 * 3) + (0 * -1) + (-2 * 4) - .3 \\ &= -5.3 \end{aligned}$$

$$y\_pred = g(-5.3) = 0$$

$$\begin{aligned} update &= lr * (y - y\_pred) \\ &= .01 * (1 - 0) \\ &= .01 \\ weights &= weights + (update * \boldsymbol{x}) \\ &= \begin{bmatrix} 3 \\ -1 \\ 4 \end{bmatrix} + .01 \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix} \\ &= \begin{bmatrix} 3 \\ -1 \\ 4 \end{bmatrix} + \begin{bmatrix} .01 \\ 0 \\ -.02 \end{bmatrix} \\ &= \begin{bmatrix} 3.01 \\ -1 \\ 3.98 \end{bmatrix} \\ bias &= bias + update \\ &= -.3 + .01 \\ &= -.29 \end{aligned}$$

Note that in this exercise the prediction (y_pred) is not correct, but when the prediction is correct (y == y_pred), the update value = 0, and the weights and bias remain unchanged.