Comparison between Classification (CL) and Linear Regression (LR)

Both classification (CL) and linear regression (LR) algorithms are supervised machine learning techniques, meaning that they both use labels (known, true y outputs) to train their algorithms to predict outputs of new, unknown samples. The difference is that the labels in LR are continuous real numbers while in CL they are categorical labels (i.e. 0/1) that carry no numerical values (e.g. 0 here does not mean that it is less than 1).

Assuming a linear relationship between x and y in LR and assuming that the data points are linearly separable in CL, both algorithms are trained to construct an optimal linear model which gives the least amount of loss/error. In LR, the line itself represents and contains the predicted output values y, while the line in CL is just a separator between the two different classes of data points; it is not the output y itself.

Since both models are linear, both algorithms use the same linear function $f(w,x) = w^T x$. The LR model is $f(w,x) = w^T x$ itself (the function output/prediction is any real number $\mathbb{R}$), but the CL model is $f(w,x) = sign(w^T x)$ (the function output is either $+1$, 0 or -1, which gives the prediction of the datapoint belonging to the first class ($+1$), the second class (-1) or if the sample is unclassified (0).

Iterations, updating weights and algorithm termination:

In LR, you update the weights vector only once per iteration, but in CL you update it several times in each iteration (as many time as how many misclassified samples you have), this is shown in the algorithm steps summary below. As per the lessons we have covered, you terminate the LR algorithm when it converges, by comparing it to a given tolerance value ε. In CL, you terminate when you reach the maximum number of iterations R or when you have no misclassified samples anymore.

Algorithm steps summary:

For each LR iteration:

1- Calculate total loss $F(w)$ by summing up each individual sample's loss.
2- Find the gradient $F_w(w)$ (using $F(w)$ from the previous step) by summing up each sample's gradient.
3- Update weights w_{n+1} (using $F_w(w)$ from the previous step).
4- Perform convergence test ΔF using ε (if not converged, go to the next iteration and repeat steps, otherwise exit algorithm with $w^* = w_{n+1}$).

For each CL iteration:

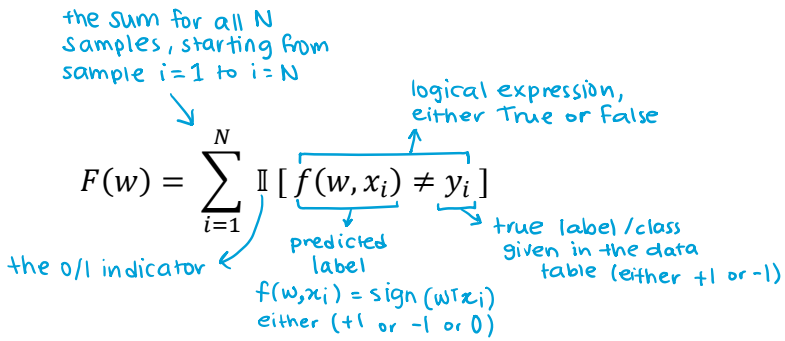
For each sample:

1- Check if the sample is misclassified using $sign(w^T x)$.
2- If misclassified, update weights w_{n+1} and move on to next sample. If correctly classified keep current weights and move on to the next sample.

Repeat these two steps until you cover all samples (only check each sample once). Unlike in LR, in order to update weights in CL you don't need to calculate the loss $F(w)$ or the gradient $F_w(w)$ beforehand. You can just use the $w_{n+1} = w_n + \alpha y_i x_i$ equation. Once you've covered all samples, move on to the next iteration and check the list of samples one by one again using the same two steps above. Exit algorithm when you reach the given maximum number of iterations R.

CL loss functions:

There are two different loss functions for CL. The first shows the **misclassification error** and the second is the **perceptron loss**. The misclassification error is expressed as:



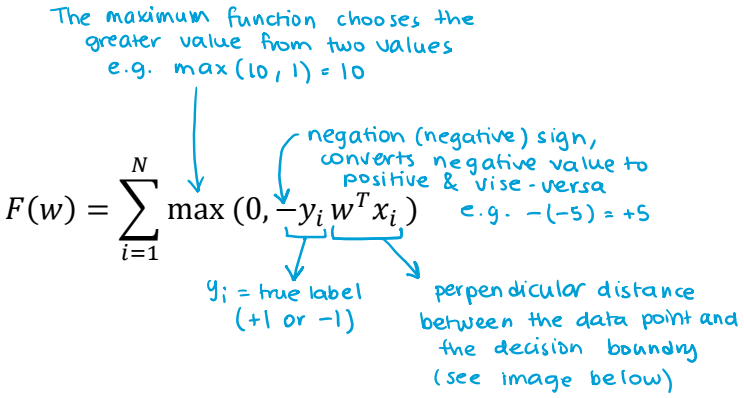
How does this function work?

This 0/1 loss function looks a bit complicated but it is basically a simple *counter* telling you exactly how many samples (datapoints) are misclassified (e.g. if you have 5 misclassified points then $F(w) = 5$). $\mathbb{I}$ is called an indicator where $\mathbb{I}\,[true] = 1$ and $\mathbb{I}\,[false] = 0$.

For example, if the predicted class from your classification model was $f(w, x_i) = sign(w^T x) = +1,$ but the true label from your data was $y_i = -1$ then it is misclassified and the term $f(w, x_i) \neq y_i$ is "$true$" which will add one count to the counter. So if you sum up all instances of misclassification you get the total count (e.g. if you have 9 samples and 5 are misclassified then $F(w) = 1 + 0 + 0 + 1 + 1 + 0 + 1 + 0 + 1 = 5$)

The **downside** of this type of loss function is that is only tells you if your sample is misclassified or not (0 or 1), it doesn't give a measure on how far the datapoint is from the line i.e. how bad the misclassification really is (see image on last page). Another downside is that due to the nature of this 0/1 function, you sometimes cannot calculate it's gradient $F_w(w)$ (i.e. $F(w)$ is not differentiable at all points).

The perceptron loss function is expressed as:

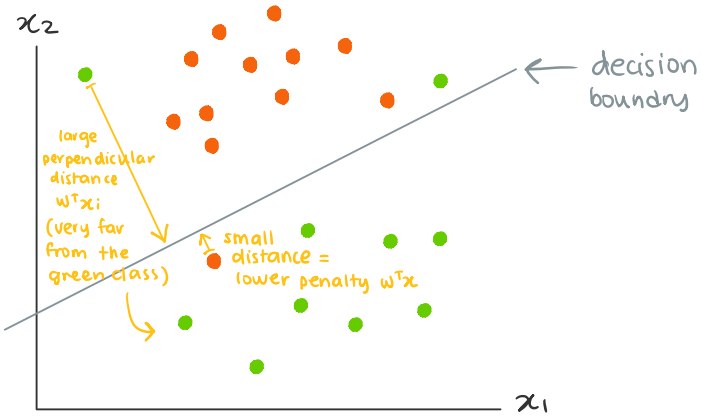


How does this perceptron function work?

If your sample is correctly classified (meaning $y_i = sign(w^T x_i)$ whether it is $(+1 = +1)$ or $(-1 = -1)$), multiplying two values with the same sign will give a positive value. In the perceptron loss function, this positive value is negated (converted to a negative value) so that the $\max()$ function can choose 0 since there is no misclassification error. Like in the 0/1 loss function above, correct classifications return 0 since we don't want to add any misclassification penalty to the final total loss/error.

Otherwise, if your sample is misclassified (meaning $y_i \neq sign(w^T x_i)$ whether it is $(+1 \neq -1)$ or $(-1 \neq +1)$), multiplying two values with opposite signs will give a negative value. When this value is negated it turns into a positive value and will thus be returned by the $\max()$ function as a misclassification penalty that will be added to the final loss sum $F(w)$. Here, badly misclassified samples will have a large $y_i\, w^T x_i$ penalty value while the "slightly" misclassified samples (i.e. those that are relatively closer to the decision boundary) will have a lower $y_i\, w^T x_i$ penalty contribution.

The **advantages** of the perceptron loss compared to the indicator loss is that it is a differentiable function (you can calculate its gradient $F_w(w)$ at most points) and it gives you a measure (that is equal to $y_i\, w^T x_i$ as we saw in the explanation above) of how far the data point is from the decision boundary line, which is why we use the perceptron loss instead of the 0/1 loss function in our classification calculations.



$w^T x_i$ is the perpendicular (90° angle) distance between the data point & the decision boundry.