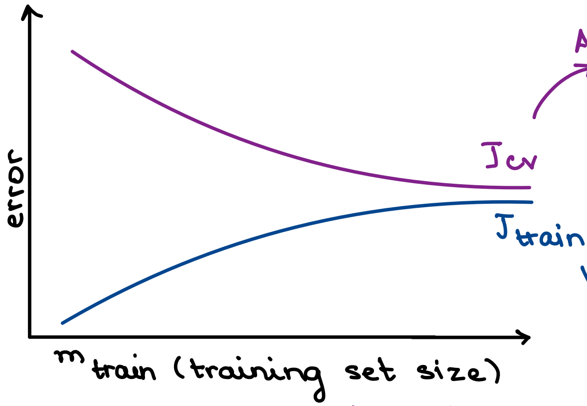
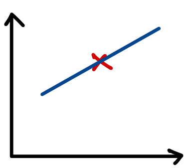
Learning curves are a great way to tell how an algorithm is doing based on the no. of training examples it has.



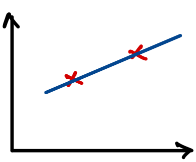
As training set size increases J_{cv} gets lower because more the data, better the algorithm

As training set size increases J_{train} error increases because from training set perspective it's easier to perfectly fit less data

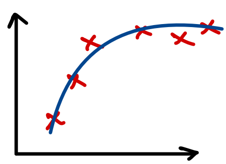
J_{cv} always greater than J_{train} because we have fit our parameters according to the training set so we expect to do better on that



perfect fit



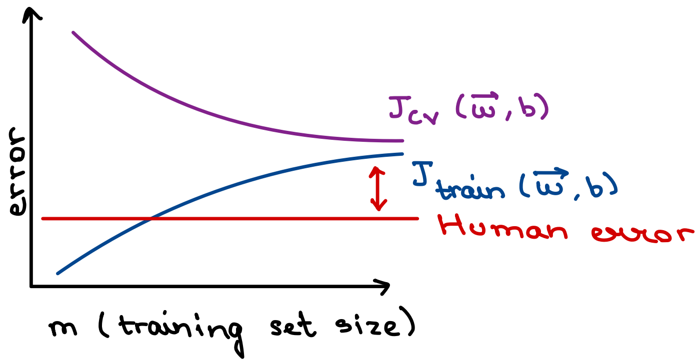
perfect fit



not a perfect fit

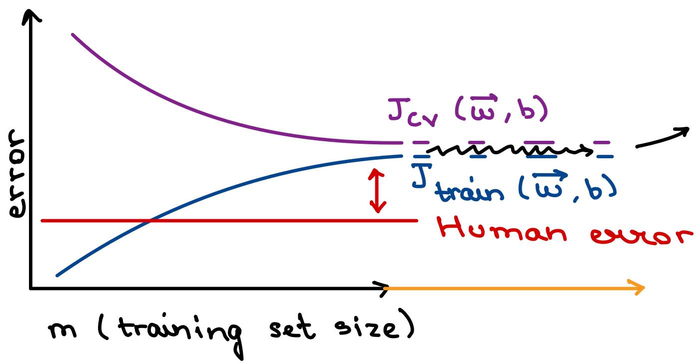
small data = easier to fit

Case of High bias



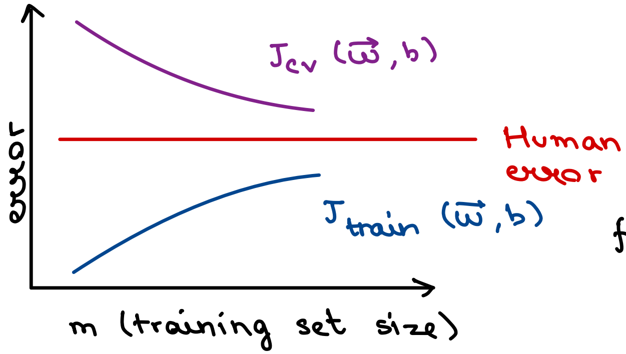
mostly happens in linear function $f_{\vec{w},b}(x) = w.x + b$

increasing training size won't help because J_{train} and J_{cv} tend to be flat (plateau) and don't change much in case of linear functions.



We shouldn't add more training data in case of high bias, instead we should look for other methods.

Case of high variance



mostly happens in case of a high order polynomial

$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + w_4x^4$$

As we've discussed before, increasing training data size is likely to help with the problem of overfitting.