# Machine Learning

Suhas Jain
19CS30048

Shashvat Gupta
19CS30042

5th March 2021

## Validation versus regularisation

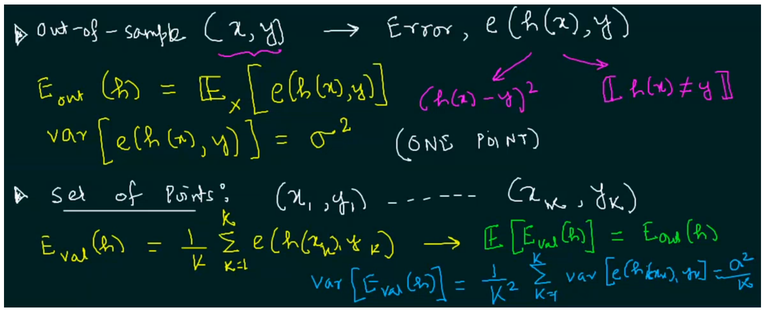
Our goal while training our machine learning model is to track our out of sample error and bring it as close as we can to our in sample error. It still can't be tracked exactly so we allow some amount of penalty and say that the absolute difference between the in sample and out of sample error is upper bounded by this penalty. This penalty can be present due to variety of different reasons like overfitting, wrong model selection, noise in the training data etc.

$$\underbrace{E_{out}(h)}_{validation} = E_{in}(h) + \underbrace{Penalty}_{regularization}$$

Here, as indicated in the expression, regularisation helps us estimate the value of the penalty, whereas on the other hand validation helps us estimate the value of out of sample error of our model.

## How does validation work?

In validation we use certain data points not used while training our model initially, and then calculate error for these data points between the predicted and actual value. Then we use these error calculations to estimate the out of sample error of our model.



▷ Out-of-sample $\underbrace{(x, y)}$ $\longrightarrow$ Error, $e(h(x), y)$

$(h(x) - y)^2$ $\quad$ $[\![ h(x) \neq y ]\!]$

$E_{out}(h) = \mathbb{E}_x\left[e(h(x), y)\right]$

$\text{var}\left[e(h(x), y)\right] = \sigma^2$ (ONE POINT)

▷ Set of Points: $(x_1, y_1)$ ------ $(x_K, y_K)$

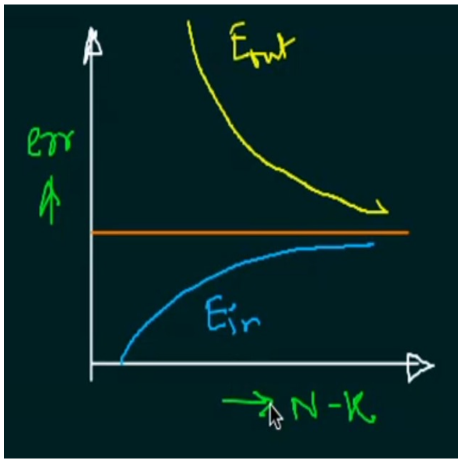
$E_{val}(h) = \frac{1}{K}\sum_{k=1}^{K} e(h(x_k), y_k) \longrightarrow \mathbb{E}\left[E_{val}(h)\right] = E_{out}(h)$

$\text{Var}\left[E_{val}(h)\right] = \frac{1}{K^2}\sum_{k=1}^{K} \text{var}\left[e(h(x_k), y_k)\right] = \frac{\sigma^2}{K}$

As we can see in the figure above how using the error e(h(x),y) for a point we didn't use in the training can help us track the out of sample error of our model. Similarly it can be extended to multiple data points and we can estimate out out of sample error using them.
One important metric to look here is the variance of the errors we are calculating for these points, as this tells us how accurately we are estimating the out of sample error, which is our ultimate goal. As we can see the k (no. of validation data points) is in the denominator of the equation for calculating variance, which brings us to the conclusion that the more number of validation data points we have the better we'll be able to estimate our out of sample error.

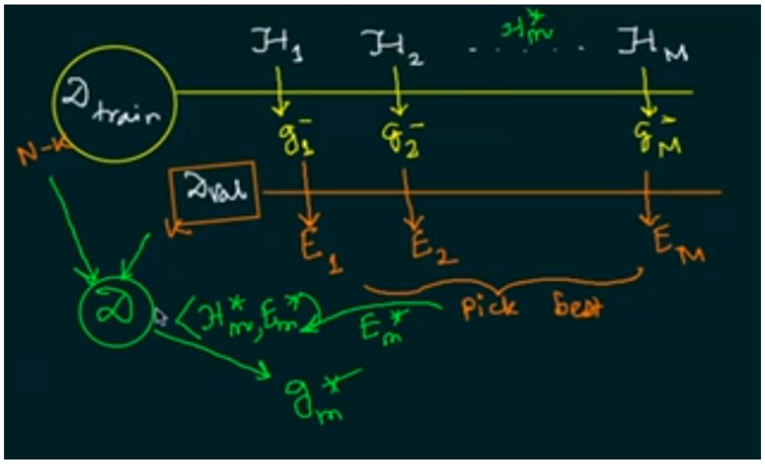
## The validation-accuracy trade off

Suppose we have N training data points and we use k of these points for our validation purposes. This leaves us with N-k data points to train our model. Now when we have a close look at the graph between our errors and the number of data points we use for training, we get some interesting observations. Here

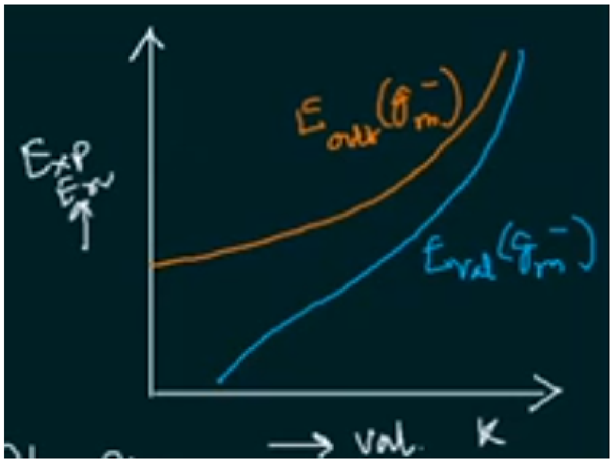


we observe that as we increase the k (and hence decrease N-k), the less accurate our model becomes worse. We estimate our out of sample error very well but we have a much less accurate model. Similarly for a small k, we train a better model but our estimation of out of sample error is bad.
The possible solution to this problem is that we use N-k data points for training, k data points for validation, and finally supply a model which was trained using all the N data points while using the out of sample error estimation of our previous model as the proxy for this model. As we can intuitively understand how with increasing k this proxy will become less and less accurate but practical experience has found out the values of k around $\frac{N}{6}$ to $\frac{N}{5}$ gives us good results.

## Understanding Validation



To get a deeper understanding of how validation works, let's assume we have $M$ models, namely $H_1$, $H_2$, ...., $H_M$. On training these models on the training dataset $D_{train}$ which contains $N - k$ points, we get $M$ hypothesis, which can be called as $g_1^-$, $g_2^-$, ...., $g_M^-$. Now, we calculate the error of each of these hypothesis with the validation dataset $D_{val}$ which contains $k$ points. These gives us validation set errors of each hypothesis $E_1$, $E_2$, ....., $E_M$. Out of these we pick up the model with the least validation set error $(E_m^*)$ and train the model with the complete dataset $(D)$ containing all $N$ points to give the best hypothesis $(g_m^*)$.



This plot arises *two questions* in our minds:

1. *Why does the $E_{out}(g_m^-)$ curve grow like this?*

2. *Why do the two curves meet?*

**Explanation**: The X-axis of the graph denotes $k$ which is number of points in validation set, which is inversely proportional to number of points in training set. Figure on page 2 shows the $E_{out}$ with respect to number of points in training set, which is the mirror image of the plot we have in this graph. Hence, the curve grows in this manner. As for the second question, as the $k$ increases, there is a better chance that my validation set estimates $E_{out}$ correctly.
By *Hoeffding inequality*, we can say:

$$E_{out}(g^-_{m^*}) \leq E_{val}(g^-_{m^*}) + \underbrace{O(\sqrt{\frac{ln(M)}{k}})}$$

where this is:

$$\sqrt{\frac{1}{2k}ln(\frac{2M}{\delta})}\ , with\ (1-\delta)\ probability\ or\ confidence$$

From this inequality, we can see that $k$ is inversely proportional to difference of Validation error and $E_{out}$.
**The Dilemma**: Our target is,

$$E_{out}(g)\ \approx\ E_{out}(g^-)\ \approx\ E_{val}(g^-)$$

Now, to satisfy the second inequality, the error is inversely proportional to $k$, hence a larger $k$ is good for the second approximation. Whereas, the first approximation requires a smaller value of $k$ as $E_{out}(g)$ is with respect to final hypothesis and $E_{out}(g^-)$ is with respect to the training data. There is no solution for this dilemma
*Rule of Thumb:* Take $k$ as $\frac{N}{5}$

## Cross Validation

### General Cross Validation

Let we have $D_n\ =\ (x_1, y_1),\ .....,\ (x_N, y_N)$,
Out of this we keep 1 point for validation and the rest $N-1$ points for training. Hence the Validation error for hypothesis $g^-_n$ will be:

$$e_n\ =\ E_{val}(g^-_n)\ =\ e(g^-_n(x_n), y_n)$$

**Cross Validation Error**: This is defined as the mean of validation errors with respect to all the points in the dataset i.e. mean of all $e_n$. Hence, mathematically

$$E_{CV}\ =\ \frac{1}{N}\sum_{n=1}^{N} e_n$$

That means that we perform this experiment, $N$ times for all points taking a different point as validation set each time and calculating the validation error with respect to each point. The mean of all these validation errors becomes the cross validation error.
**Explaining with an example**:

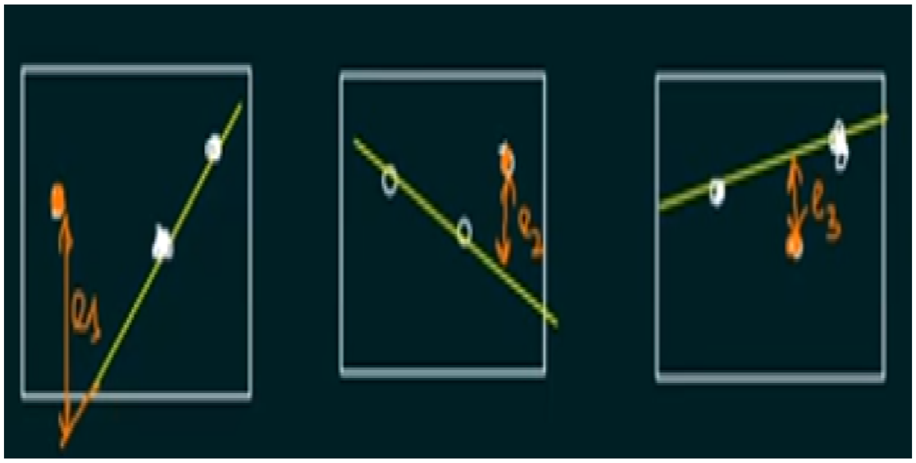


Figure 1: Linear Hypothesis Fitting

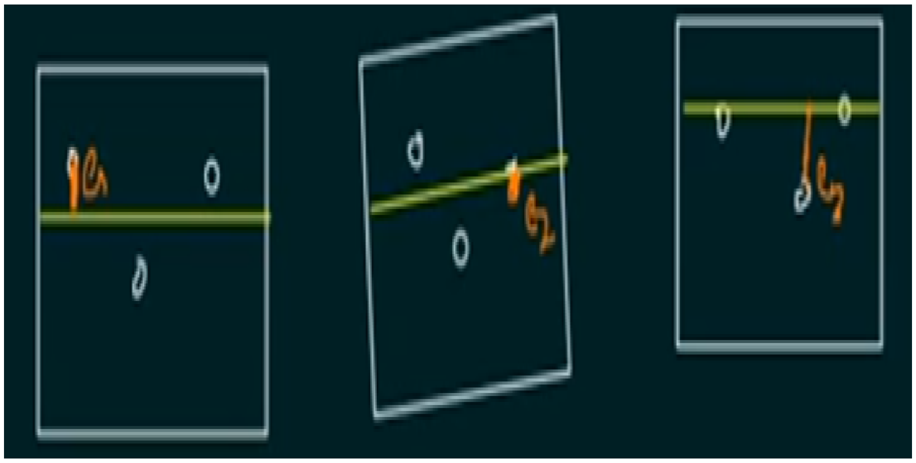


Figure 2: Constant Hypothesis Fitting

Let we want to fit a *linear* hypothesis for these three points. The cross validation experiment suggests that we leave one point out, train a model for the other two and find the corresponding validation error in the manner shown in the above plots. Final Cross Validation Error will be:

$$E_{CV} = \frac{e_1 + e_2 + e_3}{3}$$

Now we fit a *constant* hypothesis for these three points. The cross validation experiment suggests that we leave one point out, train a model for the other two and find the corresponding validation error in the manner shown in the above plots.
Depending on the Cross Validation error of both the hypothesis we may decide which hypothesis fits the dataset better.

## K-Fold Cross Validation

We know that datasets can be huge and in such cases in cross validation where we have to train and compute error $N$ number of times, Cross Validation gets computationally expensive. Hence, we use another method called the **K-Fold Cross Validation** method to calculate Cross Validation Error.

**Method**: We segregate the data into folds of equal number of points i.e. $k$ number of points. Let the folds be $D_1$, $D_2$, .., $D_k$, ..., $D_n$.
Now, we take out a fold (let say $D_k$) and train the model on the rest of the folds i.e. we train the model on $\{D_i\} - D_k$ folds. We validate the hypothesis on only $D_k$ and we repeat this procedure for all folds one by one. Thus, we train the model $n(= \frac{N}{k})$ times on $N - k$ points each time. Finally we take mean of all validation errors obtained from validation with respect to each fold.

## Stratified K-fold Cross Validation

In K-Fold Cross Validation, we don't take care about the distribution in each fold which can lead to adverse results in some cases. Hence, we use **Stratified K-Fold Cross Validation**. In this variant, we create different *strata* of points i.e. we create strata of same output points and then use the strata to create the folds with uniform distribution of points. The rest of the procedure is same as *K-Fold Cross Validation* to obtain the final Cross Validation error of the dataset with respect to a hypothesis.