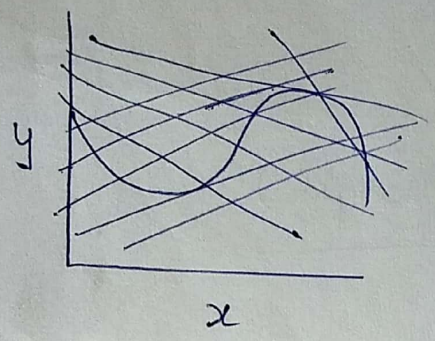
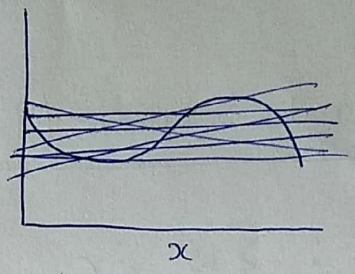
Regularization

Regularization is the cure for overfitting. Regularization is used in every machine learning application you will encounter.



without regularization



with regularization

Controling the lines by restricting them in terms of offset ~~that~~ they can have & the slope they can have.

without regularization: bias = 0.21 ; var = 1.69

with regularization: bias = 0.23 ; var = 0.33

Side note: We can use Legendre polynomials to approximate target function

Unconstrained Solution to minimize MSE

Let's assume we have a vector $z = \begin{bmatrix} 1 \\ 2^{nd} \text{ order term, } x^2 \\ 3^{rd} \text{ order term, } x^3 \\ \vdots \\ q \text{ order term, } x^q \end{bmatrix}$

For this set up, y is defined as

$y = \sum_{q} w^T z$, where w is the vector of weights

Minimize $E_{in}(w) = \frac{1}{N}\sum_{N}(w^T z_n - y_n)^2$

$$= \frac{1}{N}(Zw - y)^T(Zw - y)$$

$$w_{lin} = (Z^T Z)^{-1} Z^T y$$

Constrained Solution (Constraining the weights)

Constraint : $\sum_{q} w_q^2 \leq C$ $\quad w_1^2 + w_2^2 + \cdots + w_q^2 \leq C$

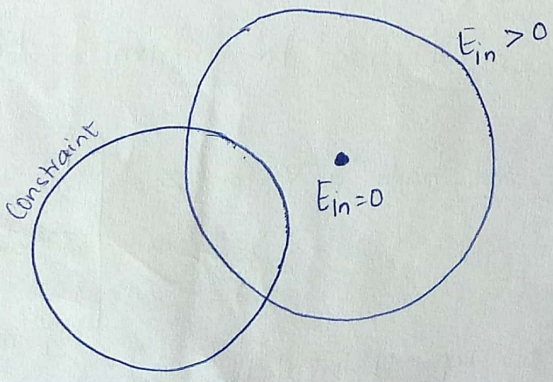
(Ball) (Sphere formula)

Minimize $\frac{1}{N}(Zw - y)^T(Zw - y)$

Subject to : $w^T w \leq C$

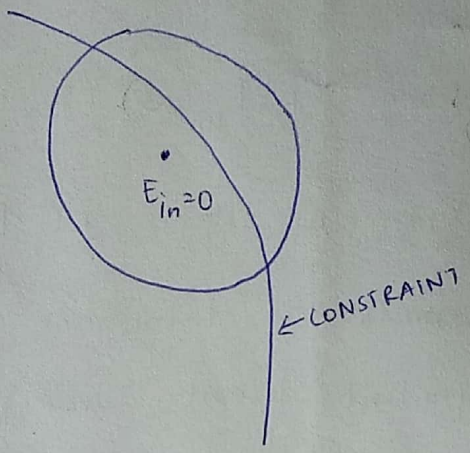
Solving graphically

Let's draw a contour line for E_{in} (MSE) & for the constraint



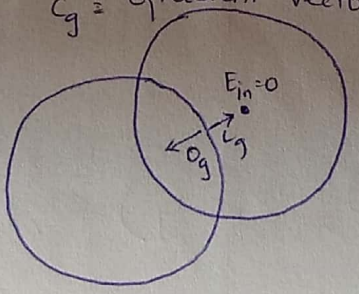
Best value of E_{in} is actually at $w^T w = C$

If C is big enough that includes $E_{in}=0$, the solution is w_{lin}



The best possible 'w' can be found when the gradient vectors of both the constraint function & the objective function are opposite to one another

O_g = Gradient Vector of Objective function
C_g = Gradient Vector of Constraint function



Because of Lagrangian, we get

$$L(w) = E_{in}(w) + \lambda w w^T \quad \text{---} (1)$$

as $L(x,y,\lambda) = f(x,y) - \lambda(g(x,y) - b)$

It's +ve $+\lambda w w^T$ & not −ve because both the gradients are opposite to one another i.e $\nabla f = -\lambda \Delta g$.

Now applying the gradient for (1) & equating to 0, we get,

$\nabla E_{in}(w) + 2\alpha w = 0$, ~~where~~ Similar

$$\nabla E_{in}(w_{reg}) + \frac{2\alpha}{N} w_{reg} = 0 \quad \text{---} \quad (2)$$

Here α is the rate of change of $E_{in}(w)$ for unit change of constant C. In the above case it is $C \uparrow \alpha \downarrow$

Here in this particular scenario as α increases C decreases as we are talking about gradient descent. In gradient descent case, the $\nabla E_{in}(w)$ is opposite to $\nabla(ww^T - C)$ & so are inversely proportional to one another.

If C is huge, α is zero.

If we expand (1) & ~~solve~~, we get

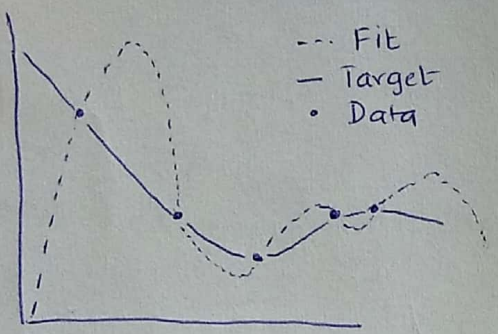
$$\frac{1}{N}\left((Zw - y)^T(Zw - y) + \alpha w^T w\right)$$

& Solve for $\nabla E_{aug}(w) = 0$, where aug stands for augmenting $\alpha w^T w$ to $E_{in}(w)$, we get

$$w_{reg} = (Z^T Z + \alpha I)^{-1} Z^T y, \text{ with regularization}$$

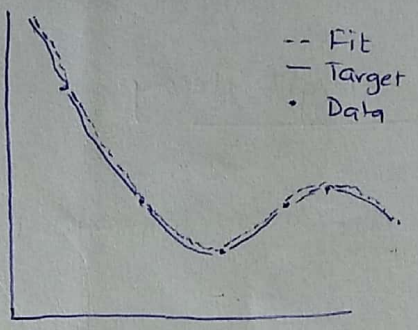
$$w_{in} = (Z^T Z)^{-1} Z^T y, \text{ without regularization.}$$

Example



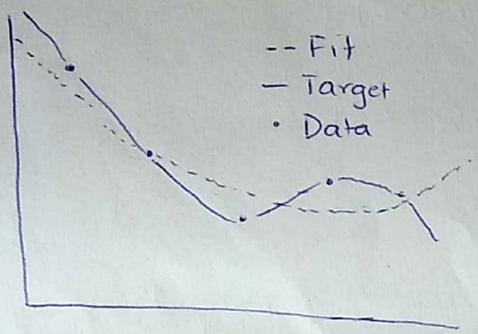
$\lambda = 0$

(Overfit)



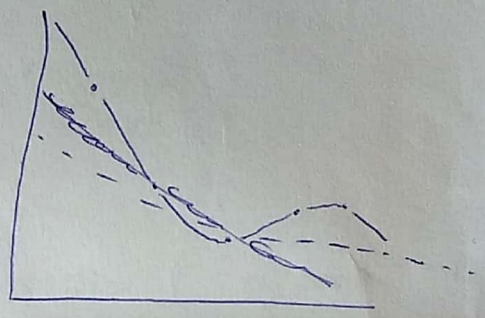
$\lambda = 0.0001$

(Good Fit)



$\lambda = 0.01$

(under fit)



$\lambda = 1$

(underfit)

The regulizer explained above is called weight decay.

~~if this term applies to neural network etc~~

In gradient descent) $w_{new} = w_{current} - \nabla E_{in}(w_{current})$ (without reg)

$w_{new} = w_{current} * (1 - \frac{2\lambda}{N}\eta)_{reg} - \nabla E_{in}(w_{current})$ [with reg]

If you note in the above equation, in every iteration to get w_{new} you decay $w_{current}$ by the factor of (reg) & then apply gradient descent. Hence this type of regularizer is termed "weight decay".

"Weight decay" term is more applicable in neural network.

Variations of weight decay

Emphasis on weights:

Some weights are more important than others

$$\sum_{q=0}^{Q} \gamma_q w_q^2 \leq C$$

Examples $\gamma_q = 2^q \Rightarrow$ low-order fit

$\gamma_q = 2^{-q} \Rightarrow$ high-order fit

because smaller γ can have larger weights & vice versa

Tikhonov Regularizer is a special form of weight decay with the form $w^T \Gamma^T \Gamma w$. If we put this in matrix form is a general quadratic form. It has

- γ for diagonal guys $(w_1 w_1, w_2 w_2, w_3 w_3, \cdots)$ & it has
- γ for off diagonal guys also $(w_1 w_2, w_1 w_3, w_3 w_4, \cdots w_5 w_6, \cdots)$ for corelated weights.

With proper choices of γ you can get various forms like low order, high order etc.

Regularizer harms fitting the noise more than it harms fitting the signal, It harms overfitting more than it harms the fitting.

Soft weight elimination is another form of regularizer. where some of the weights are eliminated (not completely, but close to zero) This is applied in neural networks & is given by

$$\Omega(w) = \sum \frac{(w_{ij}^{(l)})^2}{\beta^2 + (w_{ij}^{(l)})^2}$$

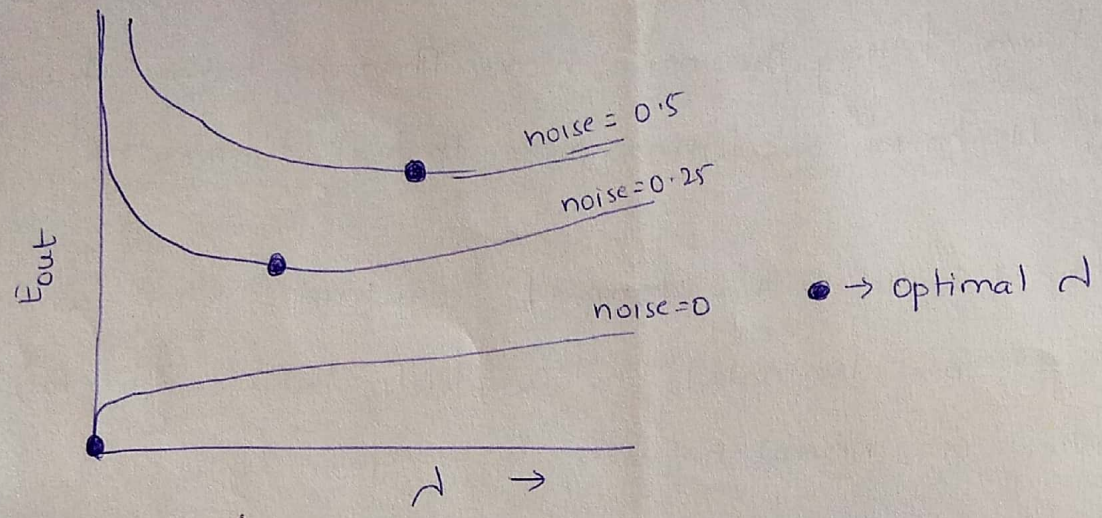
The intution is ~~small~~ Fewer weights ⇒ Smaller VC dimension.

Guiding principle for choosing the perfect regularizer is to choose in the direction of smoother or simpler.

If we choose bad regularizer then Validation comes rescue.

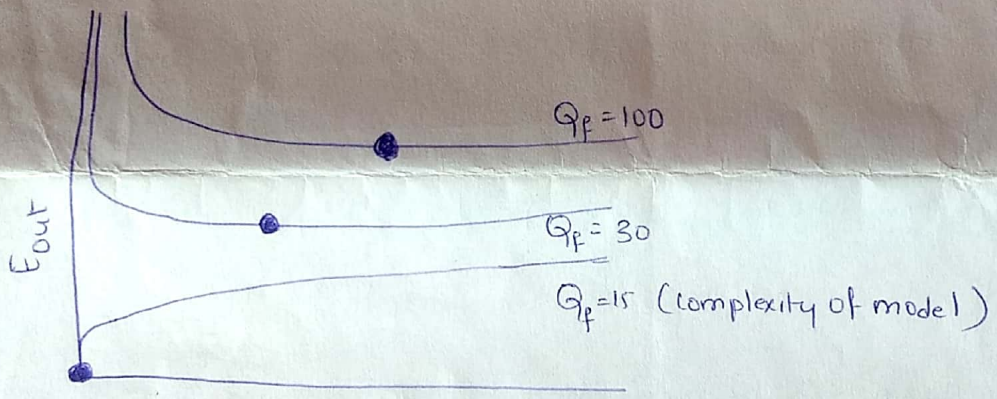
Early stopping as a regularizer. Stop the epoch early.

Optimal λ



(Stochastic Noise)

λ should increase as noise increases



(Deterministic Noise)

λ should increase as complexity of the model increases.

Finally, Choosing a regulizer is an experimental activity, more than a completely principled activity.

Bigger VC dimensions Vs Choosing regularizer

is to choose higher VC dimension & work backwards into the lower side by adding regularizer.

Validation

Regularization vs Validation

Generalization Error = Empirical Error + overfit penalty

↓ Validation estimates this quantity

↓ Regularization estimates this quantity

Let's explore the composition / ~~Statistical~~ parameters of generalization error

$$\text{Mean}(\mu) = E[e(h(x), y)] = \text{Generalization Error}$$

$$\text{Variance} = \text{var}[e(h(x), y)] = \sigma^2$$

Here $e(h(x), y)$ is the error function either squared error or binary error. It is a random variable.

Now let's consider a set of input datapoints

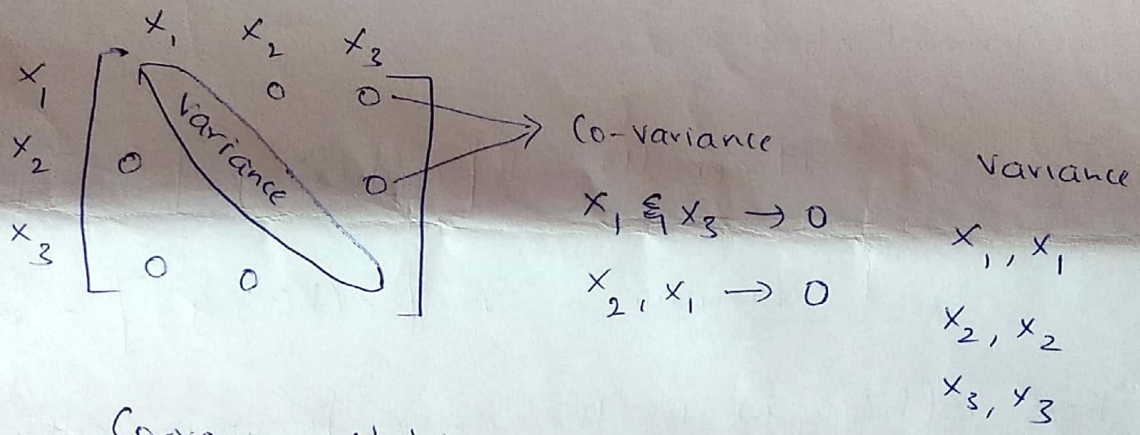
$(x_1, y_1), \ldots (x_k, y_k) \rightarrow$ Validation Set.

$$\text{Mean}(\mu) = \frac{1}{K} \sum_{k=1}^{K} E[e(h(x), y)] = \text{Generalization error}$$

$$\text{Variance} = \left(\frac{1}{k^2}\right) \sum_{k=1}^{k} \underbrace{\text{var}\left[e(h(x_k), y_k)\right]}_{\text{Variance (Not Co-variance)}} = \frac{\sigma^2}{k}$$

Number of elements in the Co-variance Matrix

Co-variance is zero because the data points picked are random & hence we end up with only the diagonal elements which are Variance



Co-variance Matrix

Generalization Error = Validation Error + $O\left(\frac{1}{\sqrt{k}}\right)$

This suggests validation error is the good approximation to Generalization error & is proportional to $\frac{1}{\sqrt{k}}$ (Standard deviation)

where k is the number of data points chosen for validation set.

~~Divisio~~

~~let's assume~~

Division of the data set

Given the data set of size N $\{(x_1, y_1), \ldots, (x_N, y_N)\}$

Training set → N − K

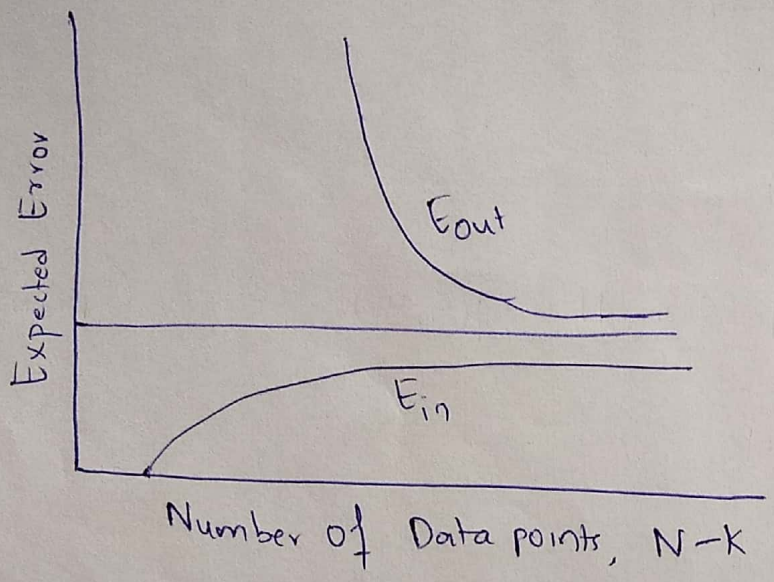
Validation set → K

The strength of Generalization Error estimate is directly proportional to $\left(\frac{1}{\sqrt{K}}\right)$

Small K → Less confidence on the estimate

Large K → Good confidence on the estimate ?

It's a question mark because we cannot afford to have large K as the size of K increases, ~~the~~ the training size decreases & reliability on the hypothesis will ~~co~~ be costly as shown in the graph below.



Another technique is to put back k datapoints & train the model again which will be given by

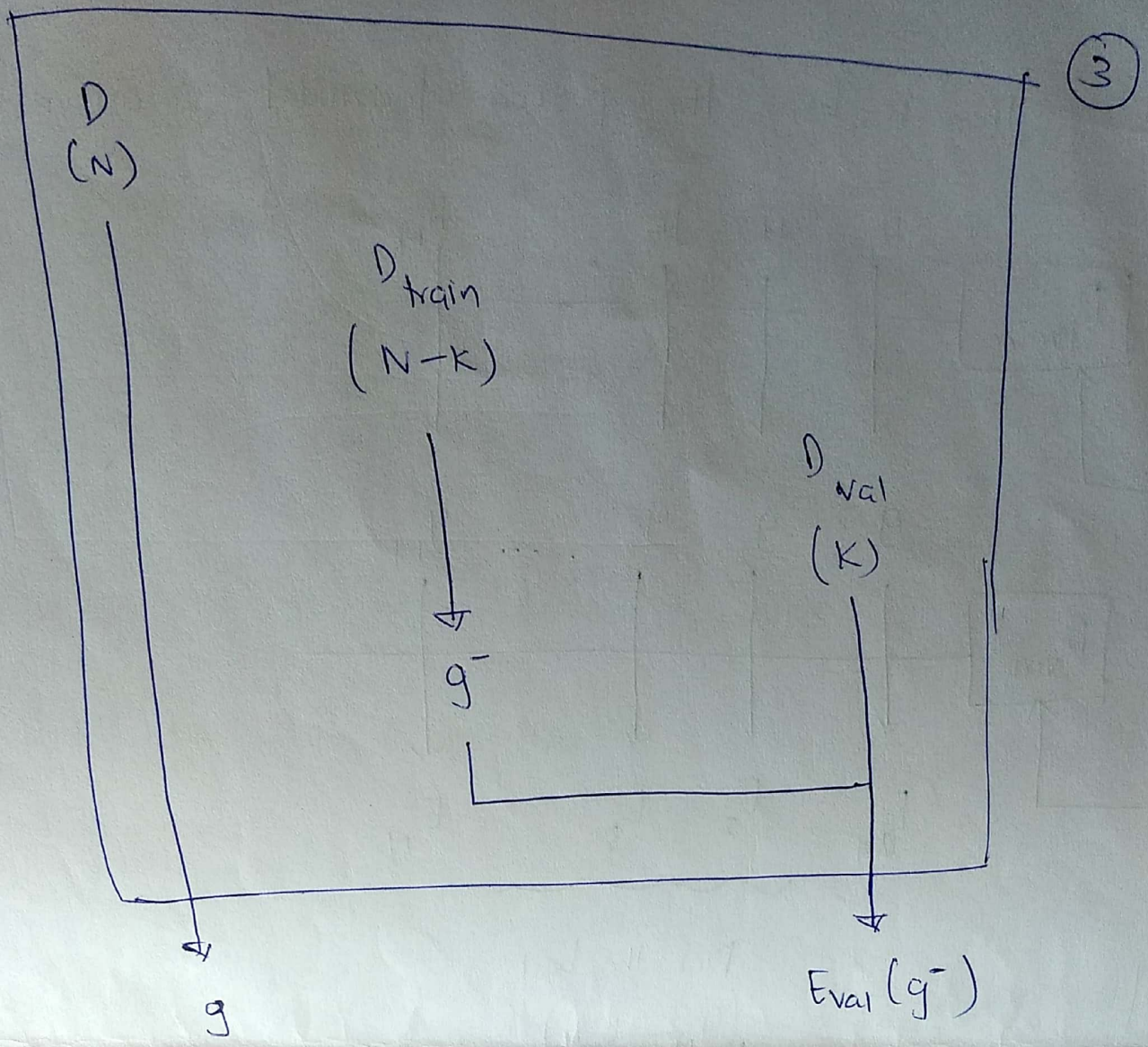
$$E_{out}(g) = E_{val}(g^-) \pm O\left(\frac{1}{\sqrt{k}}\right)$$

Notice g & g^- (selected final hypothesis on the reduced set, $N-k$)

↓

(selected final hypothesis on the complete set, N)

Again, it is not reliable as we are estimating on $N-k$ but reporting on N. As shown below

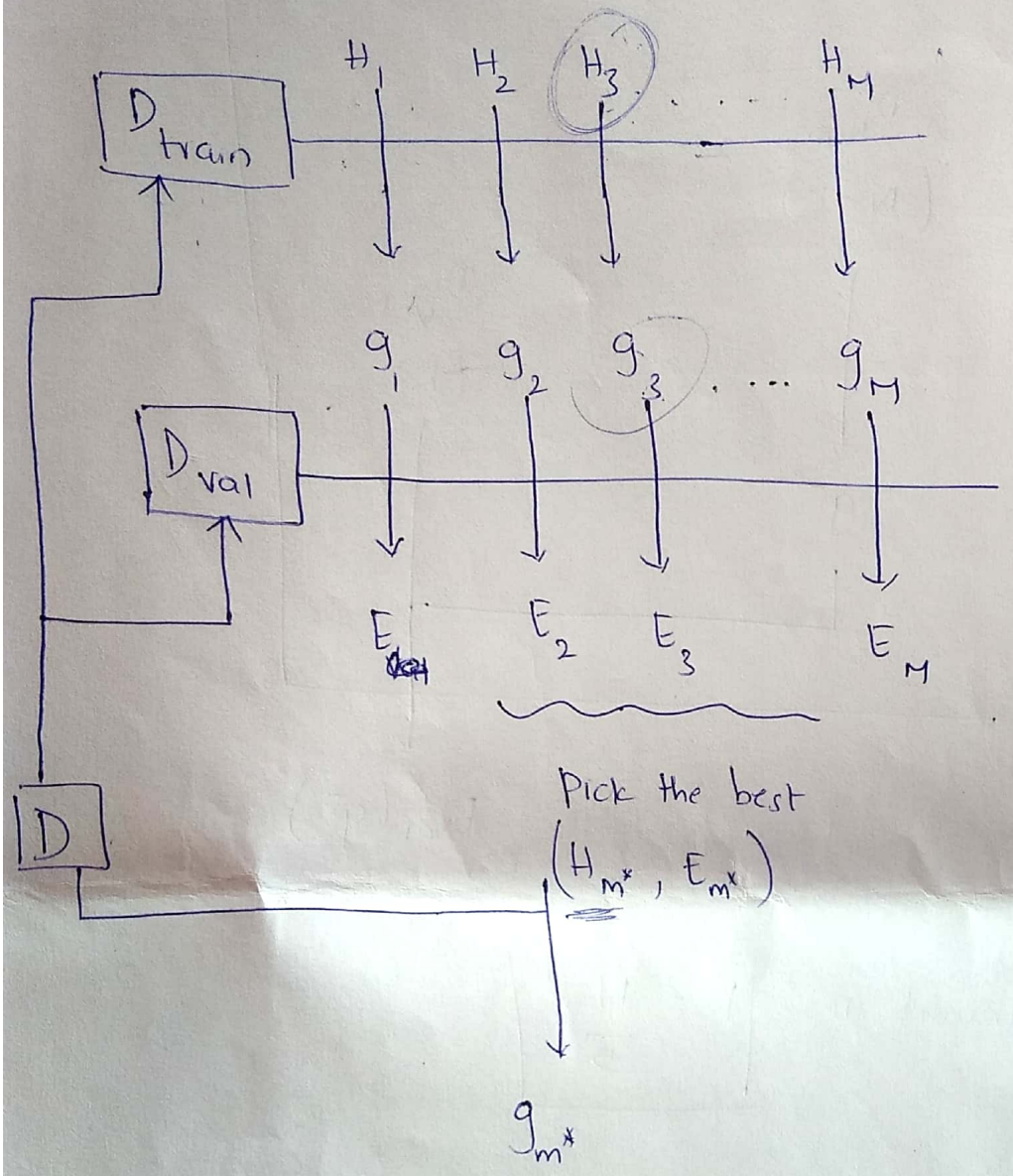


So, the rule of thumb is $\boxed{K = \frac{N}{5}}$

Why is validation set used for ?

Validation set is used for picking the model or choosing the right model & hence it is also called as Optimistic bias model. We are biased towards the model which reports small validation error. But this bias will not hurt us in generalization & usually ignorable.

This is how the process of model selection happens



Pick D_{train} & train 'M' models & ~~evaluate~~ evaluate them using D_{val} & pick the best model. Now train the best model using D (complete) & report the final hypothesis.

The dilemma about k

final hypothesis we report

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

(small k) (large k)

Can we have k both small & large ?

It turns out to be possible.

Leave one out

N-1 points for training & 1 point for validation

$$D_n = (x_1, y_1), \cdots (x_{n-1}, y_{n-1}), \cancel{(x_n, y_n)}, (x_{n+1}, y_{n+1}), \cdots$$

Final hypothesis learned from D_n is g_n^-

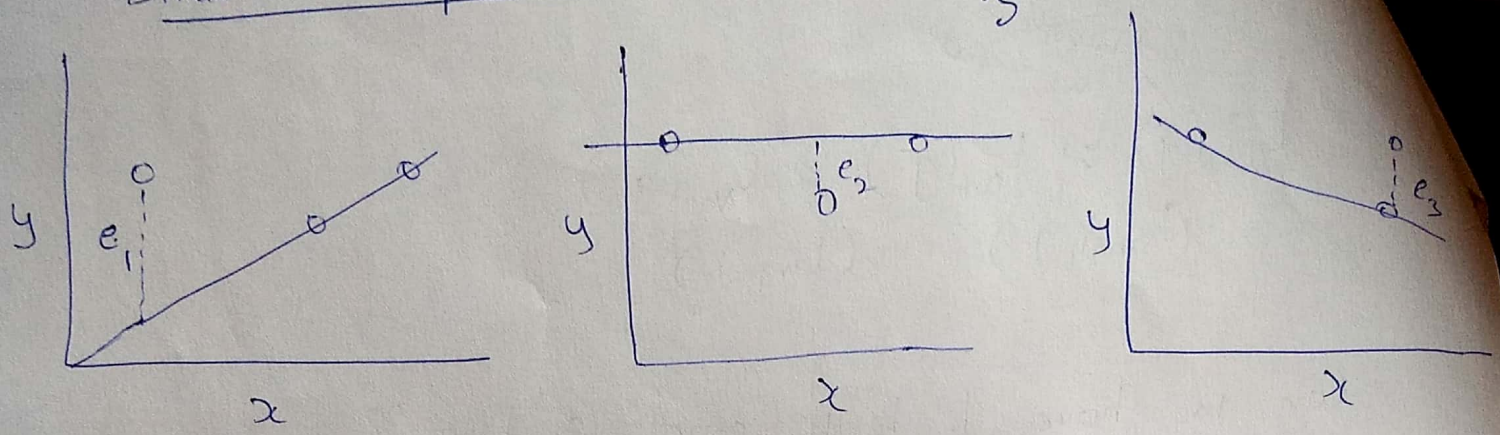
They are all different hypothesis but they are the realization of almost similar data set (training examples)

$$e_n = E_{val}(g_n^-) = e(g_n^-(x_n), y_n)$$

Cross validation error : $E_{cv} = \frac{1}{N} \sum_{n=1}^{N} e_n$

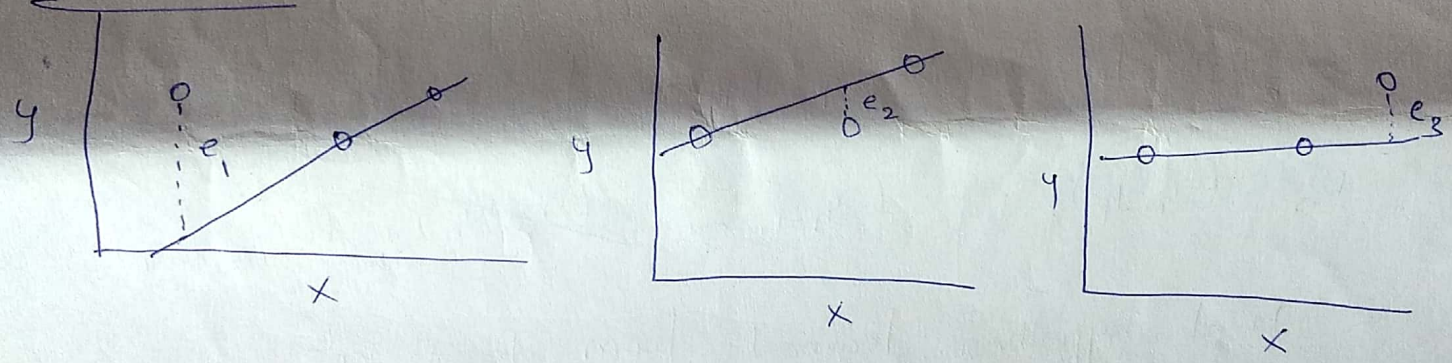
Illustration of cross validation

3-data points example.

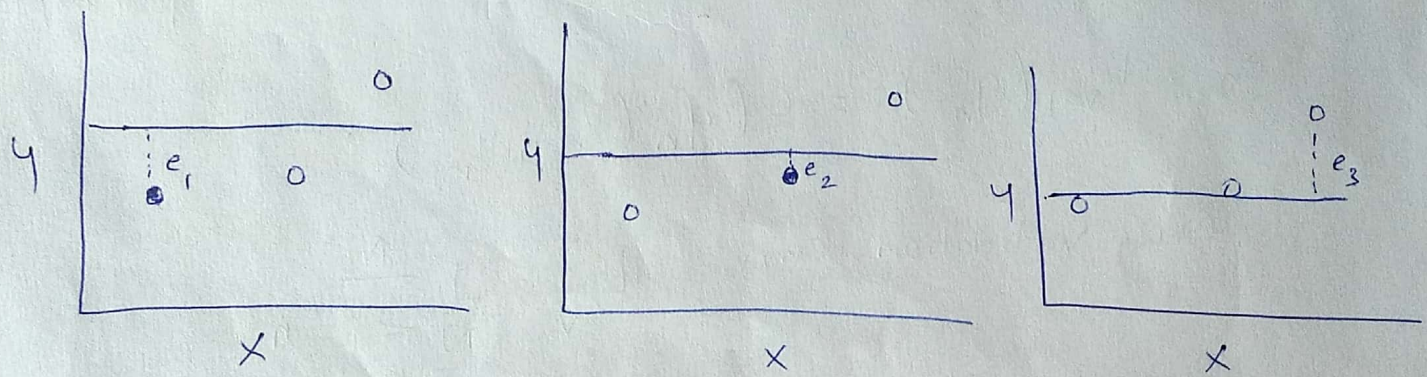


$$E_{CV} = \frac{1}{3}(e_1 + e_2 + e_3)$$

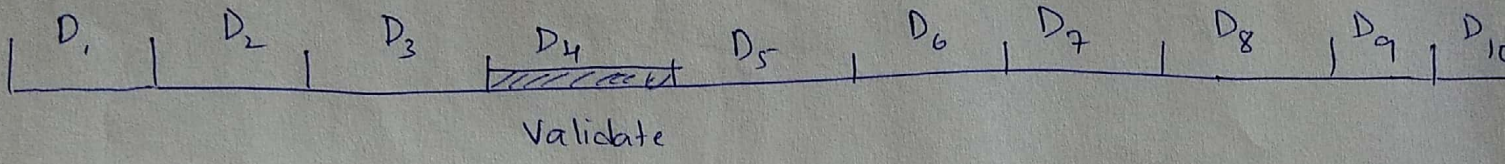
Linear Model:



Constant Model:



Leave more than one Out



$\frac{N}{k}$ training sessions on $N-k$ points each

10-fold cross validation : $k = \frac{N}{10}$

$$P\left[\left|E_{out} - E_{in}\right| > \epsilon\right] \leq \underbrace{4 m_H(2N) e^{-1/8 \epsilon^2 N}}_{\delta}$$

Get ϵ in terms of δ

$$\delta = 4 m_H(2N) e^{-1/8 \epsilon^2 N}$$

$$\Rightarrow \epsilon = \underbrace{\sqrt{\frac{8}{N} \ln \frac{4 m_H(2N)}{\delta}}}_{\Omega}$$

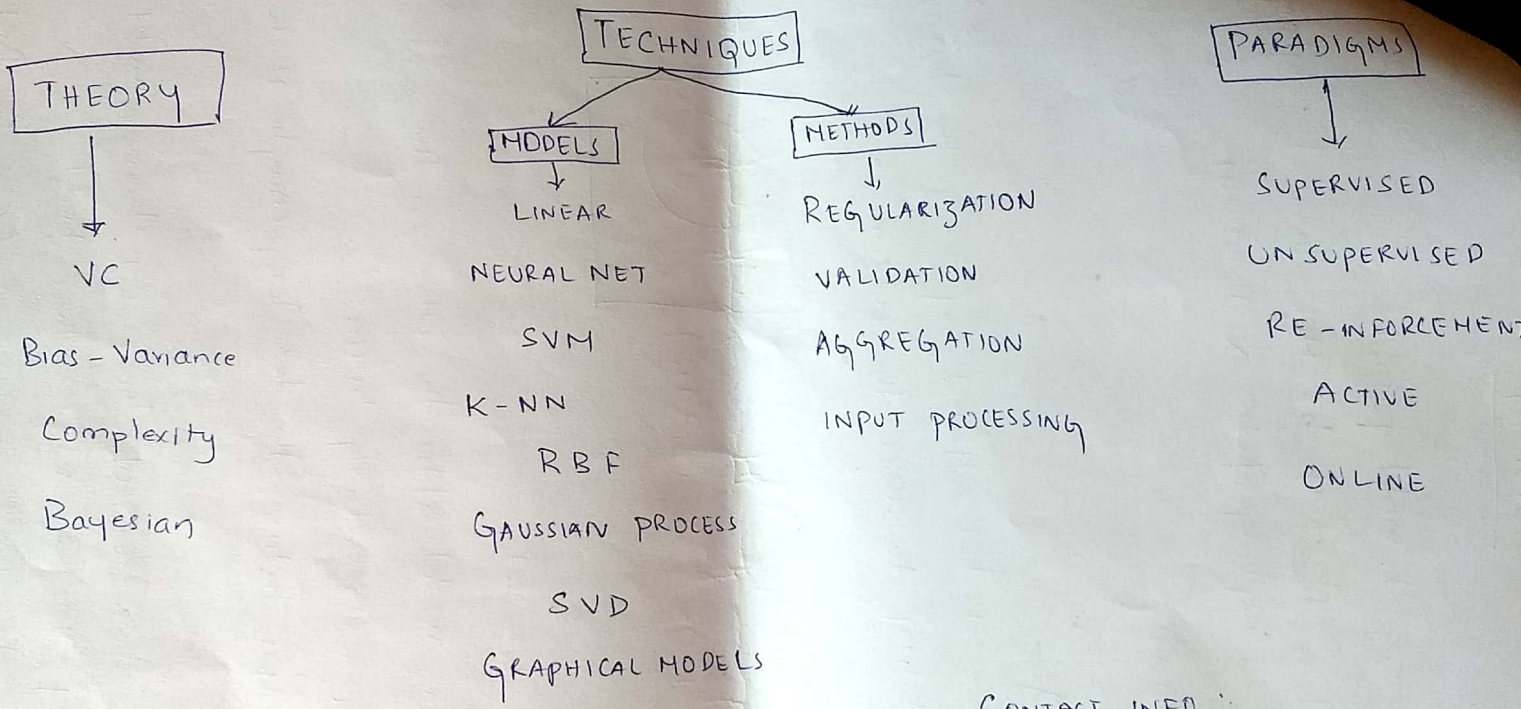
with probability $\geq 1 - \delta$ we get

$$\underbrace{\left|E_{out} - E_{in}\right| \leq \Omega(N, H, \delta)}$$

$\downarrow$

The Generalization bound.

The MAP



CONTACT INFO :

EMAIL : vinuth.tulasi@gmail.com

GITHUB : www.github.com/vinutht/Deep Learni

Summary

① Occam's Razor

Choose the simplest model possible

~~Simplicity~~ Complexity of h: MDL, order of polynomial

Complexity of H: VC Dimension

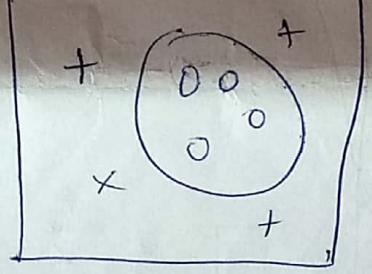
② Sampling Bias

③ Data Snooping

If a data set has affected any set in the learning process, its ability to assess the outcome has been compromised

$z = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$

or $z = (1, x_1^2, x_2^2)$ or $(1, x_1^2 + x_2^2)$



Reuse of a data set

Trying one model after the other on the same data set, you will eventually 'succeed'

If you torture the data long enough, it will confess

VC dimension of the ~~total~~ learning model

May include what others tried!

④ Use of prior knowledge is necessary (Remember Bait Shyness)