Ily we can add dummy features to get complex boundary.

But overfitting, F!

Regularisation: "I'm gonna save you"

$$\text{cost}' = \text{cost} + \lambda \sum f(m_i)$$

~~λ=0~~ → high λ → underfitting

$$L1 = \text{cost} + \lambda \sum |m_i| \quad \text{lasso}$$

$$L2 = \text{cost} + \lambda \sum m_i^2 \quad \text{Ridge}$$

* No regularisation on intercept.

Performance measures in classification

(1) Confusion matrix

sklearn.metrics → confusion matrix

Predicted \ Actual	T	F
T	TP	FP
F	FN	TN

$$TPR = \frac{TP}{TP+FN}$$

$$FPR = \frac{FP}{FP+TN}$$

~~Prec~~ For a class (+)

$$\text{Accuracy} = \frac{TP + TN}{TP+FN+TN+FP}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN} = TPR = \text{Sensitivity}$$

$$\text{Sensitivity} = \frac{TP}{TP+FN} = TPR \quad \text{(ability to detect +ve class)}$$

$$\text{Specivity} = \frac{TN}{TN+FP} = 1 - FPR.$$

Bayes Theorem : Naive Bayes :

Bayes Theorm : $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$

$y = a_1, a_2, \ldots a_k$ (output classes)

then $Y = \max_{i=1}^{k} \left(P(y = a_i \mid x = x) \right)$

$= \max_{i=1}^{k} \left(\frac{P(x = x \mid y = a_i)\; P(y = a_i)}{P(x = x)} \right)$

$$Y = \max_{i=1}^{k} \left(P(X = x \mid y = a_i)\; P(y = a_i) \right)$$

Y → output class

$P(y = a_i) = \frac{|a_i|}{|y|} = \frac{|a_i|}{|a_1| + |a_2| \ldots |a_k|}$

$P(X = x \mid y = a_i)$

Independent Events : $P(A \cap B) = P(A) * P(B)$

Given input $x \rightarrow \{x_1, x_2 \ldots x_k\}$ where x_i is value corresponding to feature f_i.

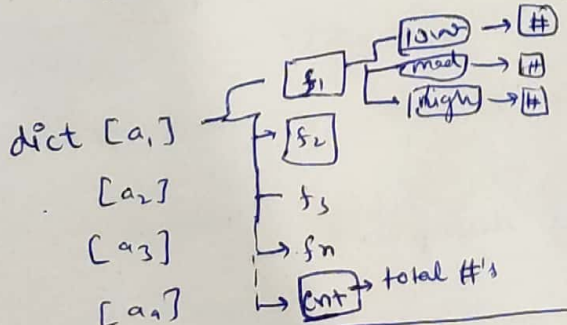
$P(X = x \mid y = a_i) = P((x_1, x_2, x_3 \ldots x_k) \mid y = a_i)$ ($f_1\ f_2\ f_3\ \ldots\ f_k$)

* Assumption → All features are independent.

$= P(f_1 = x_1 \mid y = a_i) * P(f_2 = x_2 \mid y = a_i) * \ldots P(f_k = x_k \mid y = a_i)$

$$P(X = x \mid y = a_i) = \prod_{j=1}^{k} P(f_j = x_j \mid y = a_i)$$

$$P(f_j = x_j \mid y = a_i) = \frac{\#(f_j = x_j \;\&\; y = a_i)}{\#(y = a_i)}$$



$P(f_j = x_j \mid y = a_i) = \frac{dict[a_i][f_j][x_j]}{dict[a_i][cnt]}$

Given data X_train Y_train

f_1	f_2	f_3	...	f_k	label
low					a_1
med					a_2
hig					a_1
					a_5
					a_n

Laplace correction: $P(\ )$ can be 0 if there are no value (x_j) present in feature (F_j) given a_i

So we modify formula

$$= \frac{dict[a_j][F_j][x_j] + 1}{dict[a_j][cnt] + |F_j|}$$

↳ values F_j can take

Continuous data ?

$$P(f_j = x_j \mid y = a_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(x_j - \mu)^2}{2\sigma^2}\right)}$$

Naive_bayes can be used as text collection

KNN → k-Nearest Neighbour	Do feature scaling before KNN.

You know what It is!

Cross validation:

Low value of k leads to overfitting

High value of k leads to underfitting

Make sure we have relevant data and independent.

→ Assign weights to feature (w_i for f_i)

$$\sum_{i=1}^{n} w_i (x_1^i - x_2^i)^2$$

→ Feature selection (Backward elimination)

Handling labeled data:

→ use 0, 1 in binary data. (2 values).

→ we can't assign 0, 1, 2, 3 .. to discrete data bcz distance would be effected.

	⇒	Red	Blue	Green
Red		0	0	1
Blue		0	0	1
Green		1	0	0
		0	1	0

→ **Other KNNs**

→ Brute force → compare with all datapoints

→ KD Trees something like BST

→ Ball Trees

Pros of KNN → Easy to understand and code (lol)

(2) Works for multiclassification

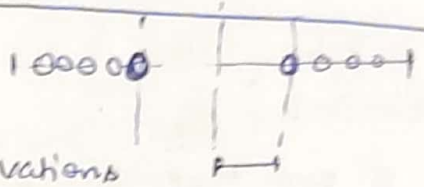
Cons of KNN:

(1) Testing time is huge.

(2) It can be biased, if training data is biased.

(3) Curse of dimensionality.

SVM



The shortest distance b/w the observations and the threshold is called margin

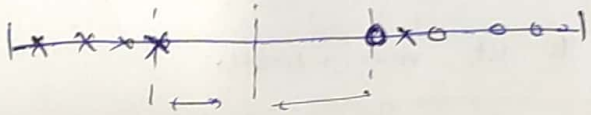
Maximal margin classifier (MMC): threshold that gives us the largest margin.

MMC is not optimal (think of outliers)

We do cross validation to find the best soft margin aka Support vector classifier.

The suppor vector classifier is a hyperplane in n-dimension.

So SVC can handle outliers nicely using cross validation



But what about

so many overlaps...!

Support Vector machine: We add extra dimensions to points and then try to fit hyperplane, SVC.

How to transform the data? SVM uses kernels function

We can find good value of (d) dimension via cross validation

Other kernal is RBF (finds SVM in infinite direction).

Kernels just calculate the r/n b/w points in high dimension without actually trasforming them.

let $r = \frac{1}{2}$ $d = 2$

Polynomial kernel $= (a \times b + r)^d$

$= ab + a^2b^2 + \frac{1}{4}$

$= (a, a^2, \frac{1}{2}), (b, b^2, \frac{1}{2})$

$\downarrow x \quad \downarrow y \quad \downarrow z$

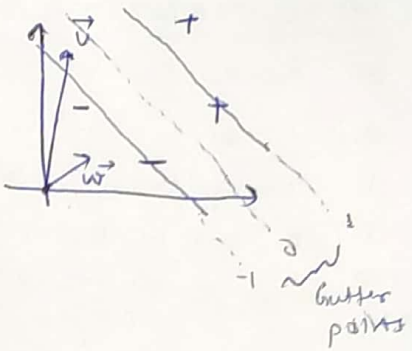
$\vec{w}$ is $\perp$ to median

$\vec{u}$ be unknown sample point

$\vec{w} \cdot \vec{u} \geq c$

i.e $\vec{w} \cdot \vec{u} + b \geq 0$ for + sample

but we still dont know $\vec{w}$ and b.



$\vec{w} \cdot \vec{x}_+ + b \geq 1$ ... ①

$\vec{w} \cdot \vec{x}_- + b \leq -1$ .. ②

introduce y_i such that $y_i = 1$ +ve sample

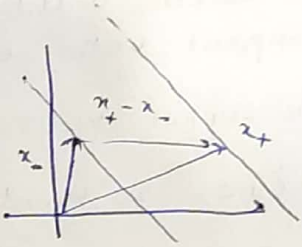
-1 -ve sample

eq 1 and 2 become

$$y_i (\vec{x}_i \cdot \vec{w} + b) \geq 1$$

$$y_i (x_i \vec{w} + b) - 1 \geq 0$$

$$y_i (x_i \vec{w} + b) - 1 = 0 \rightarrow \text{For Gutter Points.}$$



width $= (x_+ - x_-) * \frac{\vec{w}}{\|w\|}$

$= \frac{x_+ \cdot \vec{w}}{\|w\|} - \frac{x_- \vec{w}}{\|w\|} = \frac{1-b + 1+b}{\|w\|} = \frac{2}{\|w\|}$

we are maximising $\frac{1}{\|w\|}$ to get max width.

minimise $\|w\| \sim$ minimise $\frac{1}{2}\|w\|^2$ Primal Form.

we have to use Langrange multipliers.

$$L = \frac{1}{2}\|w\|^2 - \sum \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1] \quad -③$$

$$\frac{\partial L}{\partial w} = \vec{w} - \sum \alpha_i y_i x_i = 0 \Rightarrow \vec{w} = \sum_i \alpha_i y_i \vec{x}_i$$

$$\frac{\partial L}{\partial b} = \sum \alpha_i y_i = 0$$

substitute in eq 3.

$$L \Rightarrow \frac{1}{2} \left(\sum_i \alpha_i y_i \vec{x}_i\right) \cdot \left(\sum_j \alpha_j y_j \vec{x}_j\right) - \sum \alpha_i [y_i$$

$$L \Rightarrow \frac{1}{2} \left(\sum_i \alpha_i y_i \vec{x}_i\right)\left(\sum_j \alpha_j y_j \vec{x}_j\right) - \left(\sum \alpha_i y_i x_i\right)\left(\sum \alpha_j y_j \vec{x}_j\right) + \sum \alpha_i y_i b + \sum \alpha_i$$

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) \rightarrow \text{Max of this.} \quad ④$$

depend on dot product.

$\sum \alpha_i y_i \underbrace{\vec{x_i} \cdot \vec{u}}_{\text{dot product}} + b \geq 0$ +ve samble.

Won't work with non linear decision boundary.

transformation : $\phi(\vec{x}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$ to max

$\phi(x_i) \cdot \phi(u)$

$K(x_i, x_j) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$

↳ kernel function. we actually don't have to do ~~dot~~ transform

so ~~$\phi(\vec{x})$~~ , all depends on dot product.

some kernels : (1) $(\vec{u} \cdot \vec{v} + r)^d$ Polynomial kernel.

(2) $e^{-\frac{\|x_i - x_j\|}{2}}$ Radial basis kernel

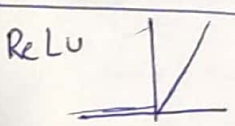
Neural Network

Activation Function → sigmoid f^n , ReLU, Soft Plus.

We get x-axis coordinate from the weight & and use activation func to get Y.

Weights come from Activation Function.

Act. (weight + biases) → new perception.



$f(x) = x \quad x \geq 0$

$= 0 \quad x < 0$

$f(x) = \max(0, x)$.

Maam PPT's after mid sem:

KNN → pattern recognition what k to chose? $k < \sqrt{n}$

→ While scaling we can add weight to the important features. w_k we can learn w_k using cross-validation.

Ez:)

Kmeans Algo: (Unsupervised learning) Jalpur PPTs

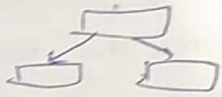
(1) Select a k (clusters to identify) (k=3)
(2) Randomly select k distinct dp. (~~Centroids~~)
(3) Measure distance of remaining points to clusters and classify them to nearest cluster.
(4) Calculate mean of each cluster ← (centroid)
(5) Repeat. (Measure distance along the mean this time)
(6) ~~Read~~ When it does draw its result and calculate the variance.

centroid can be real or imaginary dp

Repeat whole with different k and diff. starting points.
The best is where we have a gradual drop in ~~mean~~ variance.

AdaBoost → contains stumps

resource: Stat Quest PPTs



The tree with just one node and two leaves is stump.

~~Multiple~~ Forest of stumps.

A tree uses are features to make decision, but a stump would take one feature to make decision.
So 'stumps' are weak learners.

In RF, each tree has equal vote to classification
In Adaboost, each stump has different vote to classification.

In RF, each tree is independent.
In Adaboost, order is important, The error of first stump influences how second stump is made and soon.

Steps

~~(1) Make stumps of~~
(1) Assign sample weight (this will tell which dp is important) (initially for all = 1/no. of samples)
(2) Make stumps using each feature and calculate gini index. The one with lowest would be the first stump.

(4) Now we find the ~~weight~~ vote of this stump (by calculating error).

Amount of vote (say) $= \frac{1}{2} \log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$ [Σ # of incorrectly classified with their sample weight.]

(5) Now we modify sample weight.

we increase the sample weight of incorrectly classified weight of that feature, so that next time we try to classify it correctly and decrease all other sample weight.

↑ Net sample weight = sample weight × $e^{\text{amount of vote}}$

↓ sa. wt × $e^{-\text{amount of vote}}$ (also we normalize later)

and calculate new sample weight. (also we normalize later)

and find rest of stumps.

Ensemble Technique → Combining multiple models.

Need to add from Main

→ Bagging : Random Forest
(Bootstrap aggregation)

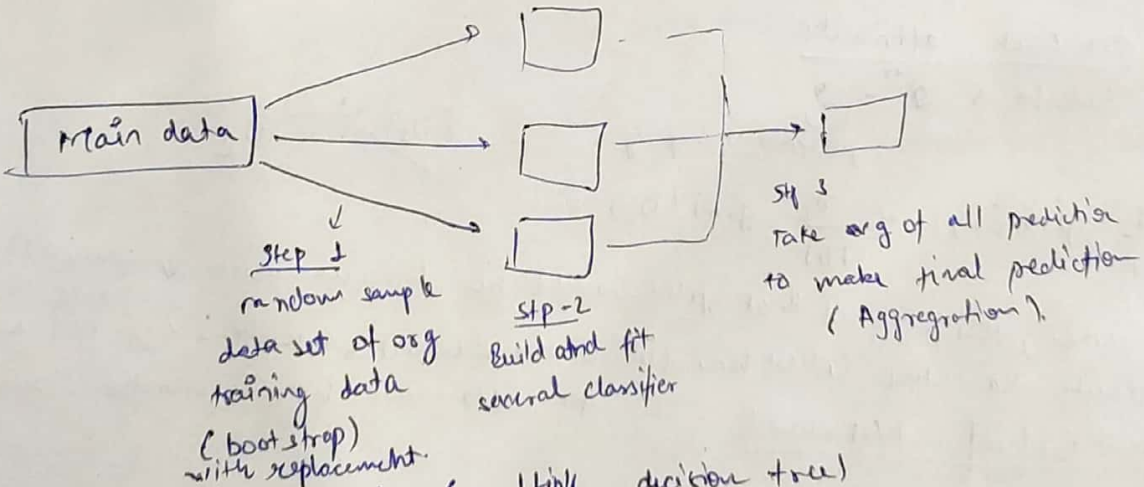
→ Boosting : ADA BOOST, GRADIENT BOOSTING, XG Boost.

Error → Bias + variance

using Bagging we can decrease variance
using Boosting ——— bias

simplilearn
~~skillcorn~~
PPT

Bagging :→ / bootstrap aggregation reduces variance of an estimate by taking mean of multiple estimates.



eg Random forest (multiple decision tree)

Boosting reduces bias by training weak learners sequentially, each trying to correct predecessor.

eg: → ADA BOOST (multiple stumps)
→ Gradient Boosting (GBM)
→ XG BOOST

{2, 1, 5 : 3} | confidence if these rules can be formed or not. {5} → {1}

SVM

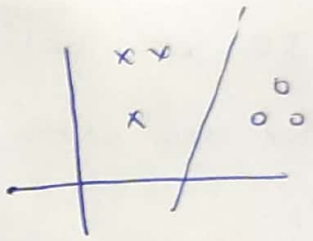
Hyperplane = $w^T \phi(n) + b = 0$

↳ error (bias)

Distance of a point (x_0) from hyperplane = $\frac{|w^T \phi(x_0) + b|}{||w||_2} = d_H \phi(x_n)$

We aim to find hyperplane that has maximum distance from the closest point - (maximise the minimum distance)

i.e $w^* = \arg\max_w [\min_n d_H \phi(x_n)]$



For x → $w^T \phi(n) + b \geq 0$

o → $w^T \phi(n) + b < 0$

→ $y_n [w^T \phi(n) + b] \geq 0$ {correct}

< 0 {incorrect}

$w^* = \arg\max_w [\min_n \frac{w^T \phi(x_0) + b|}{||w||_2}]$ ∵ perfect separation

$w^* = \arg\max \frac{1}{||w||^2}$ ~~Dual~~ Primal form

$\min \frac{1}{2} ||w||_2$

Case 2 Non-perfect separation.

New Primal form of SVM $\xi_n \geq 0$

$$\min \frac{1}{2}\|w\|_2 + C\sum_n \xi_n$$

↑ penalty for incorrect classification

C=0 : Less complex boundary

C=Inf : More complex boundary.

We want to get rid of $\phi(n)$.

We use kernels to get dual form.

Method of langrang multipliers

Steps

(1) Obtain new primal form = $\min \frac{1}{2}\|w\|_2 + C\sum \xi_n$

(2) Determine Lagrange.

(3) $\frac{dL}{dw} = 0$.

(4) Express in form of duals

(5) Back substitution.

(6) Depends to dot product.

Dual form

↳ in terms of multiplication of 2 $\phi(n)$.

which is simplified by kernels

Hyperparameters in kernelisation:

(1) Margin

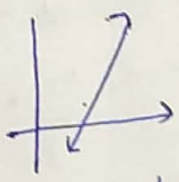
(2) Kernel

(3) Regularisation (C) [how much you want to avoid mis classification]

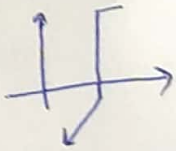
(4) Gamma.

↓ low gamma ↳ Hyperplane is influenced by far distant point

↳ High gamma ↳ Hyperplane is influenced by nearby points only.

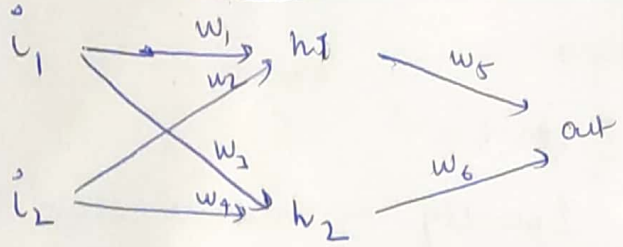


Low regul.



High regular.

Feed Forward Neural Network



$h_1 = i_1 w_1 + i_2 w_2$

$h_2 = i_1 w_3 + i_2 w_4$

predicted = $w_5 h_1 + w_6 h_2 = (i_1 w_1 + i_2 w_2) w_5 + (i_1 w_3 + i_2 w_4) w_6$

Error = $\frac{1}{2}(\text{out} - \text{predicted})^2$

$w_x^{new} = w_x - \alpha \frac{d\,\text{Error}}{d w_x}$

$$w_6' = w_6 - \alpha \frac{d\ \frac{1}{2}(\text{out} - \text{pre})^2}{d\,\text{pre}} \times \frac{d\,\text{pre}}{d w_6}$$

$$= w_6 - \alpha\,(\text{out} - \text{pred}) \times (i_1 w_3 + i_2 w_4)$$

$$= w_6 - \alpha\,\Delta \times h_2$$

$w_6' = w_6 - \alpha \Delta h_2$

$w_5' = w_5 - \alpha \Delta h_1$

$$w_1' = w_1 - \alpha \frac{d\,\text{Error}}{d w_1}$$

$$= \alpha \frac{d}{d\,\text{pre}} \frac{1}{2}(\text{out} - \text{pre})^2 \times \frac{d(w_5 h_1 + w_6 h_2)}{d h_1} \times \frac{d(i_1 w_1 + i_2 w_3)}{d w_1}$$

$$= \alpha \Delta w_6 i_1$$

$w_1' = w_1 - \alpha\, i_1 \Delta w_5$

$w_2' = w_2 - \alpha\, i_2 \Delta w_5$

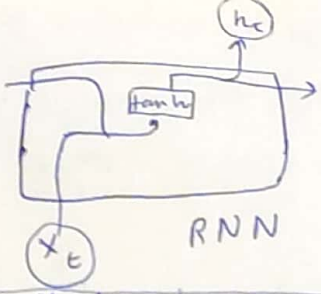
$w_3' = w_3 - \alpha\, i_1 \Delta w_6$

$w_4' = w_4 - \alpha\, i_2 \Delta w_6$

No memory in Feed Forward Network.

→ RNN models sequence data
→ short term value.

long short term memories.



→ CNN is a feed forward neural network that is generally used to analyse visual image by preprocessing data with grid like topology:

→ Every image is represented in form of arrays of pixel values.

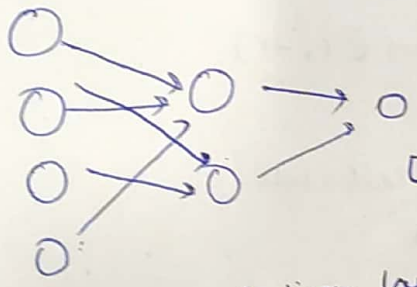
→ Layers in CNN

input → filters → feature map

① Convolution layer → has no of filters that perform convolution operation (can be optional)
② ReLU Layer → Move convolute value to ReLU (Rectified feature map)
③ Pooling layer → Pooling is a down sampling operation that reduces dimensionality of feature map. (~~2 dimension Arrays~~) (Pooled feature map)
④ Fully connected layer. Then we do flattening them into 1D.

↓

The flattened matrix from the pooling is fed as input to Fully connected layer. (Forward Fed Neural Network)



We get multiple feature map as we slide filters over the image.

There are multiple features.

[We can see this is not fully connected graph] [~~it ends~~]

(1) So in convolution layer we have multiple filters slides over image so we get multiple feature matrix
(2) Pass them through ReLU to elimate -ve values.
(3) Pooling ~~layers~~, take a window of some size n and reduce dimensionality of a ~~fe~~ rectified feature map (we take max among all values in window)

Now we stack up all the feature matrix,

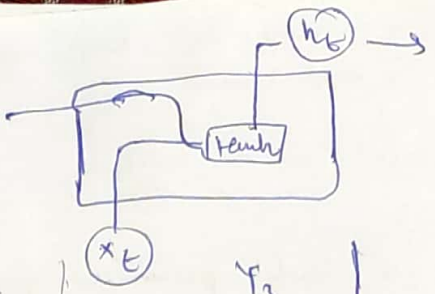
take each matrix and shrink to 1D matrix and append.

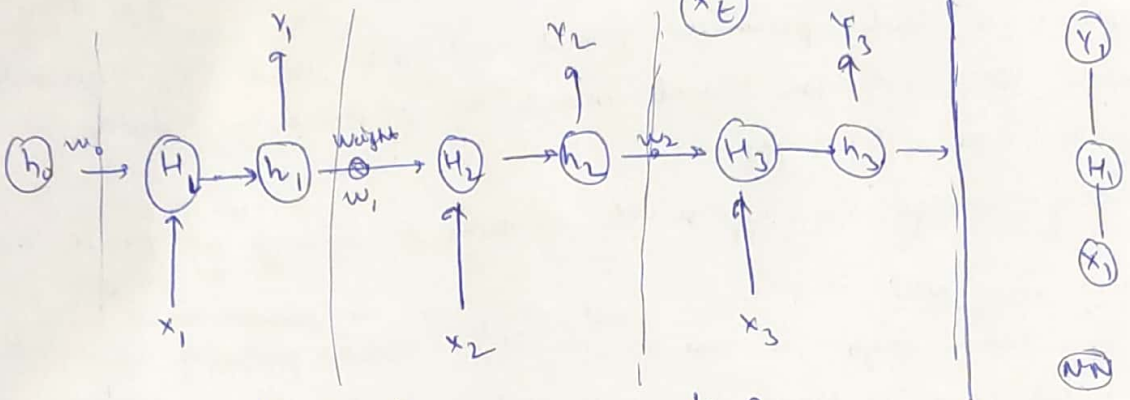
and fed it to Fully connected layer (Forward feed network.

Recurrent Neural Network

→ model for Sequential data

→ has memory. (short)





time series 1 ts 2 ts 3

taylor series

$h_t = f(h_{t-1}, x_t)$

the function is tanh

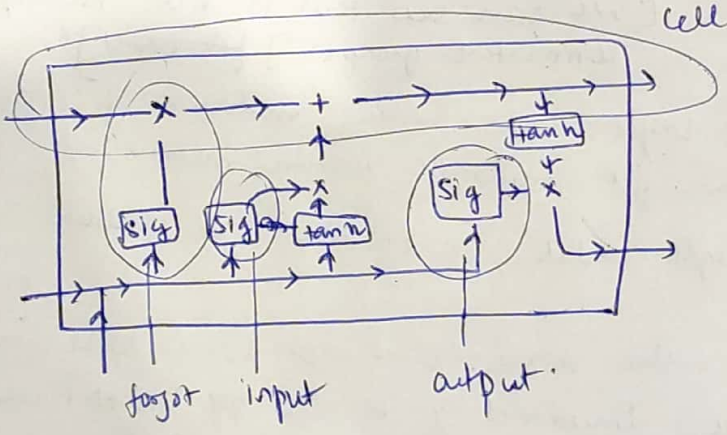
Y_t = (some output weight) × h_t

Previous states stores vanishing due to backpropagation so we use LSTM (vanishing gradient)

LSTM (long story term Memory)

→ use gate to store info

tanh makes sure value are b/w [1, -1] (h_t)



3 sigmoid
2 tanh

forgot state

sigmoid → { 1 ← retained
0 ← rejected

used in speech recognition.