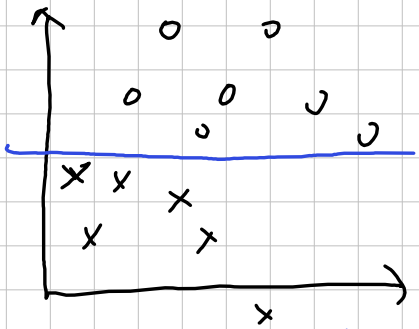
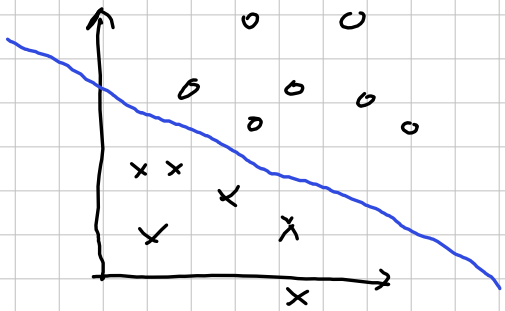
Machine Learning Notes

SVMs (Support Vectors Machine)



correct ✓

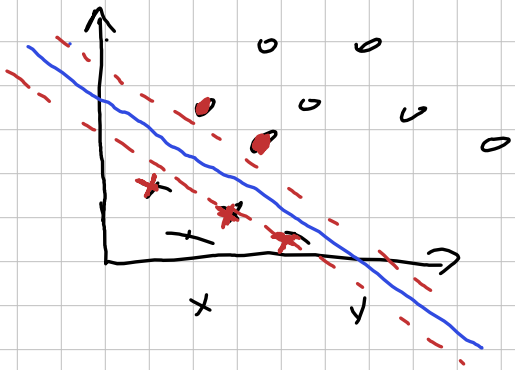
Best ✗



Correct ✓

Best ✓

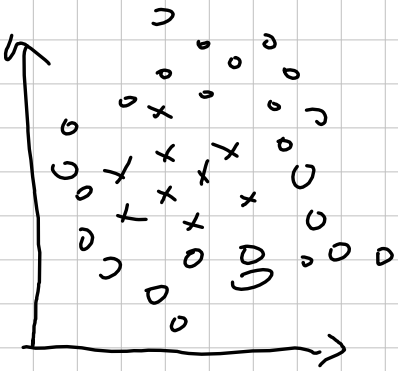
- Support vectors Machine tries to maximise the margin between Classes.
  - ↳ The closest points to the hyperplane are the ones responsible of supporting it



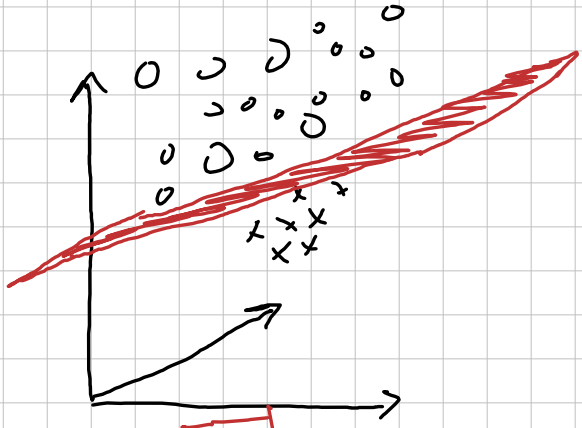
SVMs / Kernal Trick

Limitation: most raw data can not be separated by just a hyperplane

ex:



svm $(x, y) = ??$



SVM $(\boxed{f(x)}, y)$ = Flat hyperplan

Non linear Transformation of x

↳ problems:
1_ Select the correct Transformation
2_ Sofisticated boundaries
↳ increase dimentional output of the transformation
↳ Increase Computational requirements.

↳ Solution: The Kernal Trick

⇓

The idea: instead of caring about $x \to f(x) = x'$

we care about the relation between x, x' comparing them with each other

⇒ in Math it's done using the dot product

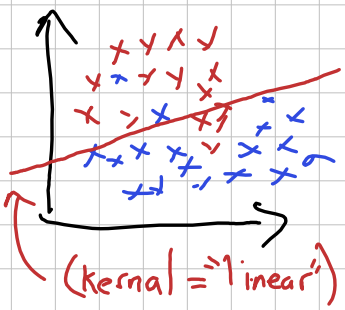
SVMs / Kernal Trick

⇒ in Math it's done using the dot product
also called inner product

$$f(x)^T f(x')$$

⟺ The kernal function $k(x, x')$

Simple Example: The identity Tranformation
(we do nothing about x)

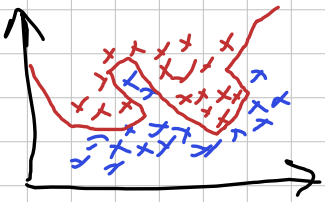


⟺ $f(x) = x$

⟺ $k(x, x') = x^T x'$ (linear kernal)

[SKL SVC(kernel="linear").fit(X,y)]

Complexe Example: The polynomial Tranformation



⟺ $f(x) = (x_1, x_2, x_1 x_2, x_1^2, x_2^2)$

⟺ $k(x, x') = (1 + x^T x')^2$ (polynomial kernal)

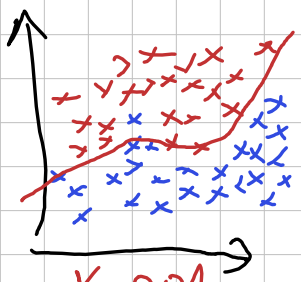
[SKL SVC(kernal="poly").fit(X,y)]

∞ Complexe example: ???? T

$f(x) =$ infinite dimentional!!

⟺ $k(x, x') = e^{-\gamma \|x - x'\|^2}$

(Radial Basis Function Kernal)



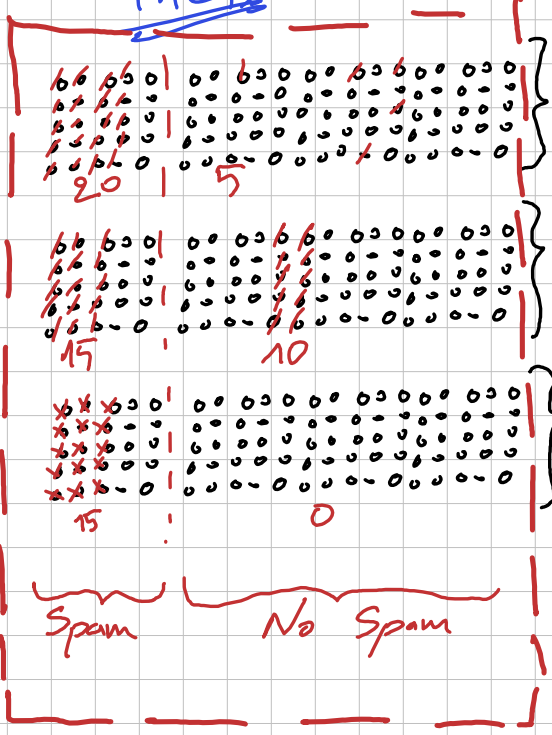
[SKL SVC(kernal="rbf", gamma=0.01).fit(X,y)]

Naive Bayes

↳ Do a naive assumption(s) about features in the dataset to make it easier to predict/classify

Example:

FACTS



Selected Dataset

(100 email)

Spam probability if:

"Buy" word ⇒
- 100 % probability = 25
- No spam probability ⇒ $\frac{5}{25}$ = 20 %
- Spam = $\frac{20}{25}$ = 80 %

"Cheap" word ⇒
- No spam = $\frac{10}{25}$ = 40 %
- Spam = $\frac{15}{25}$ = 60 %

B∩C

"Buy" & "Cheap" together ⇒
- No spam = $\frac{0}{15}$ = 0 %
- Spam = $\frac{15}{15}$ = 100 %

Probability

→ based on the dataset the model gives 100 % which is not realistic for cases outside the dataset

Solution ?
- ↳ Use bigger dataset ?
- ↳ Guess the number

Naïve assumption that "Buy" and "Cheap" are independent

We can multiply $P_{spam}(Buy) \times P_{spam}(Cheap)$

↳ $\frac{20}{25} \times \frac{15}{25} = \frac{4}{5} \times \frac{3}{5} = \frac{12}{25}$

↳ Actual number of email that are spam and contain buy & cheap

$\frac{12}{25} \times 25 = \boxed{12 \text{ email}}$

We can multiply $P_{nos}(Buy) \times P_{nos}(Cheap)$

↳ $\frac{5}{75} \times \frac{10}{75} = \frac{1}{15} \times \frac{2}{15} = \frac{2}{225}$

↳ Actual number of emails that are not spam and contains buy & cheap

$\frac{2}{225} \times 75 = \boxed{\frac{2}{3} \text{ Email}}$

New Dataset based on Naive assumptions

12 | 2/3

Spam | No spam

↳ Spam Probability "Buy" & "Cheap"

- Spam = $\frac{12}{12 + 2/3} = \frac{36}{38} = \boxed{94{,}737\ \%}$
- No spam = $\frac{2/3}{12 + 2/3} = \frac{2}{38} = \boxed{5{,}263\ \%}$

Bayes theorem

$$P(S|B) = \frac{P(B|S)P(S)}{P(B|S)P(S) + P(B|NS)P(NS)}$$

$$= \frac{20/25 \times 25/100}{20/25 \times 25/100 + \frac{5}{75} \times \frac{75}{100}} = 80\ \%$$

S: Spam
NS: Not spam
B: Buy
C: Cheap

Naïve Bayes

$$P(B \cap C) \underset{\text{Naive}}{=} P(B) \times P(C)$$

$$\Rightarrow P(S|B \cap C) = \frac{P(B|S)P(C|S)P(S)}{P(B|S)P(C|S)P(S) + P(B|NS)P(C|NS)P(NS)}$$

$$= \frac{20/25 \times 15/25 \times 25/100}{20/25 \times 15/25 \times 25/100 + 5/25 \times 10/25 \times 75/100}$$

$$= \boxed{94{,}737\ \%}$$

Decision Trees

Main idea:

Example: Existing dataset

Gender	Age	App
F	15	TT
F	25	YT
M	32	FB
F	35	YT
M	12	TT
M	14	TT

- Recomended App for F, 16: TT
- " " " F, 30: YT
- " " " M, 35: FB

Feature 1 — Feature 2

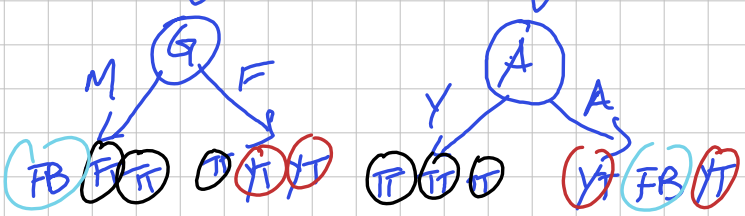
numbers → categories

simplify: <20 <

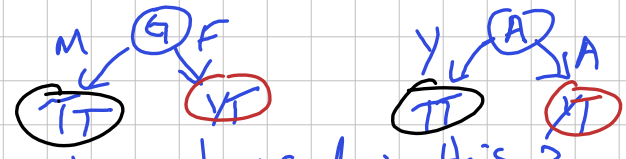
Gende	App	Age	App
F	TT	Y	TT
F	YT	A	YT
M	FB	A	FB
F	YT	A	YT
M	TT	Y	TT
M	TT	Y	TT

Gender decision stump

Age decision stump



⇒ what would we recommend based on this ??



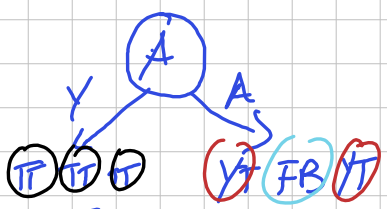
⇒ How much correct is this ?

2/3 2/3 3/3 2/3 are correct

4/6 < 5/6 Age decision stump wins!

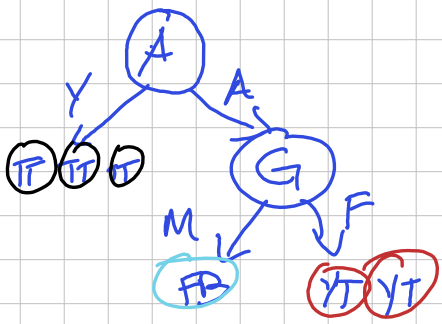
↑ Accuracy ↑

⇒



100% 2/3 accurate

Could it be improved? ⇔

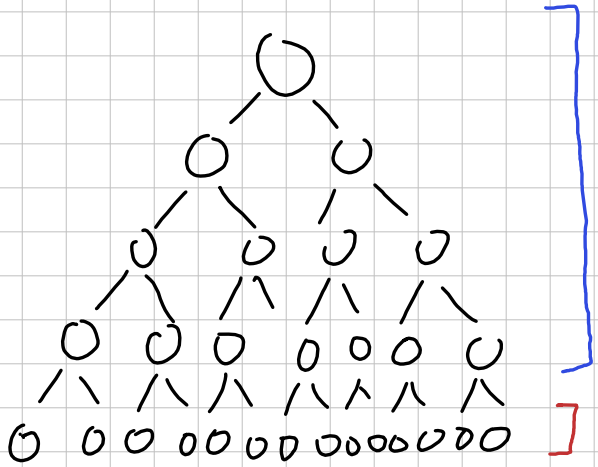


Decision Trees

Example 2: Multiple features

Gender	Age	Country	Likes	History	Subscriptions	App
✓	✓	✓	✓	✗	✓	✓
✗	✗	✗	✗	✗	✗	✗
✗	✗	✓	✗	✗	✓	✗
✓	✓	✓	✓	✗	✓	✓

Deep Tree
=
High chance
of overfitting



In example 1 Precision was used to build the decion tree

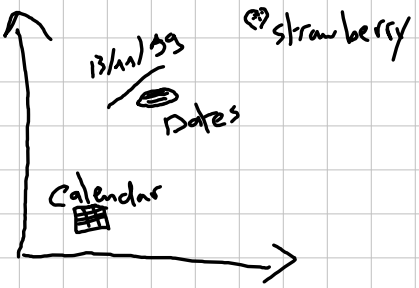
How to decide what is the best decision stump?

- ↳ Precision
- ↳ Gini Impurity Index
- ↳ Information Gain & Shannon Entropy

Avg Gini

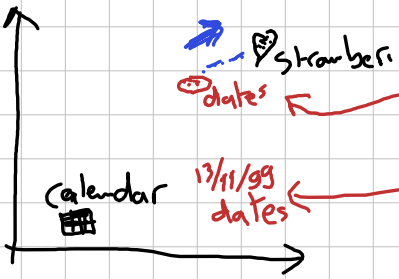
Attention Mechanism

The Challenge



- I bought strawberries & dates
- Metallica announced new dates for their world tour

The solution



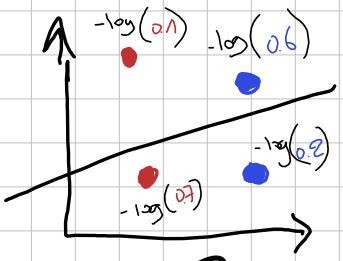
similar

I bought Strawberries & dates.

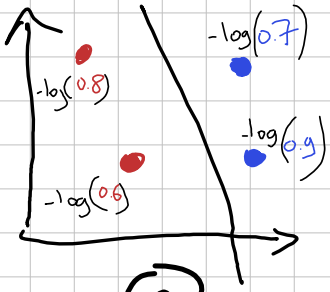
Metallica announced new dates for their world tour.

Neural Networks NN

→ Principal: minimise error (logistic Regression)



1



2 ✓

1

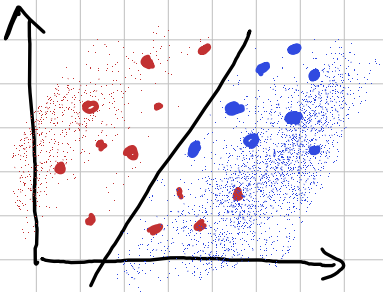
$-\log(\hat{y}_0) - \log..(\hat{y}_4) = 4.8$

2

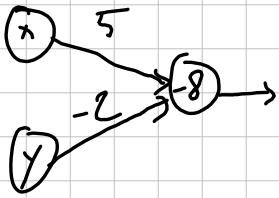
$-\log(\hat{y}_0) - \log(\hat{y}_4) = \underline{\underline{1.2}}$

→ Neurons

3x

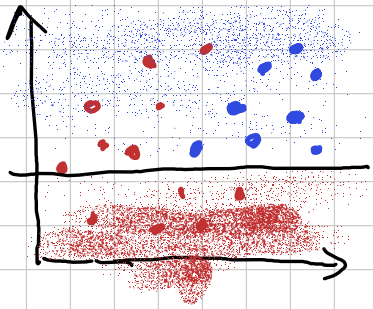


$5x - 2y = -8$

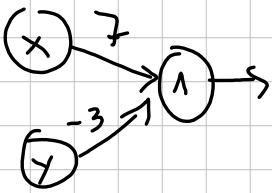


+

1x

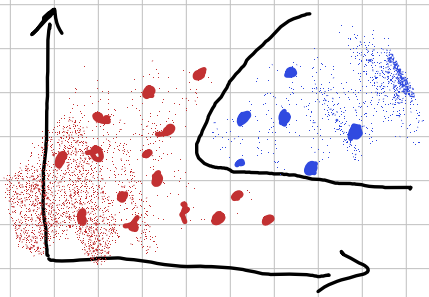


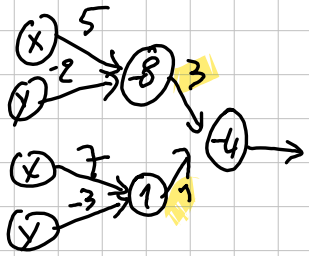
$7x - 3y = 1$

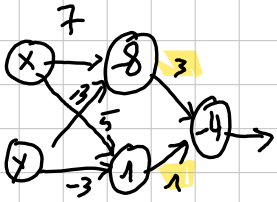


=

-4

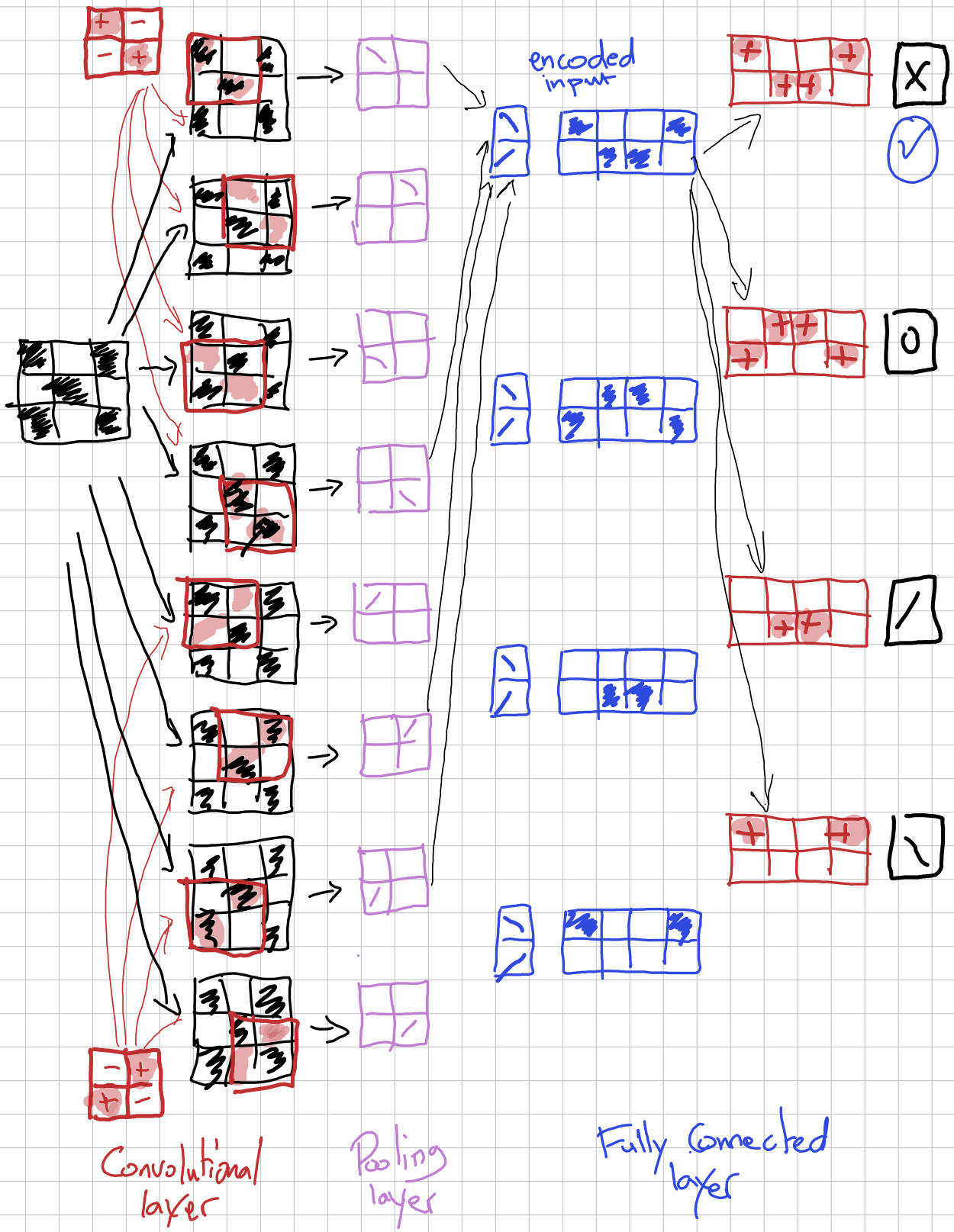




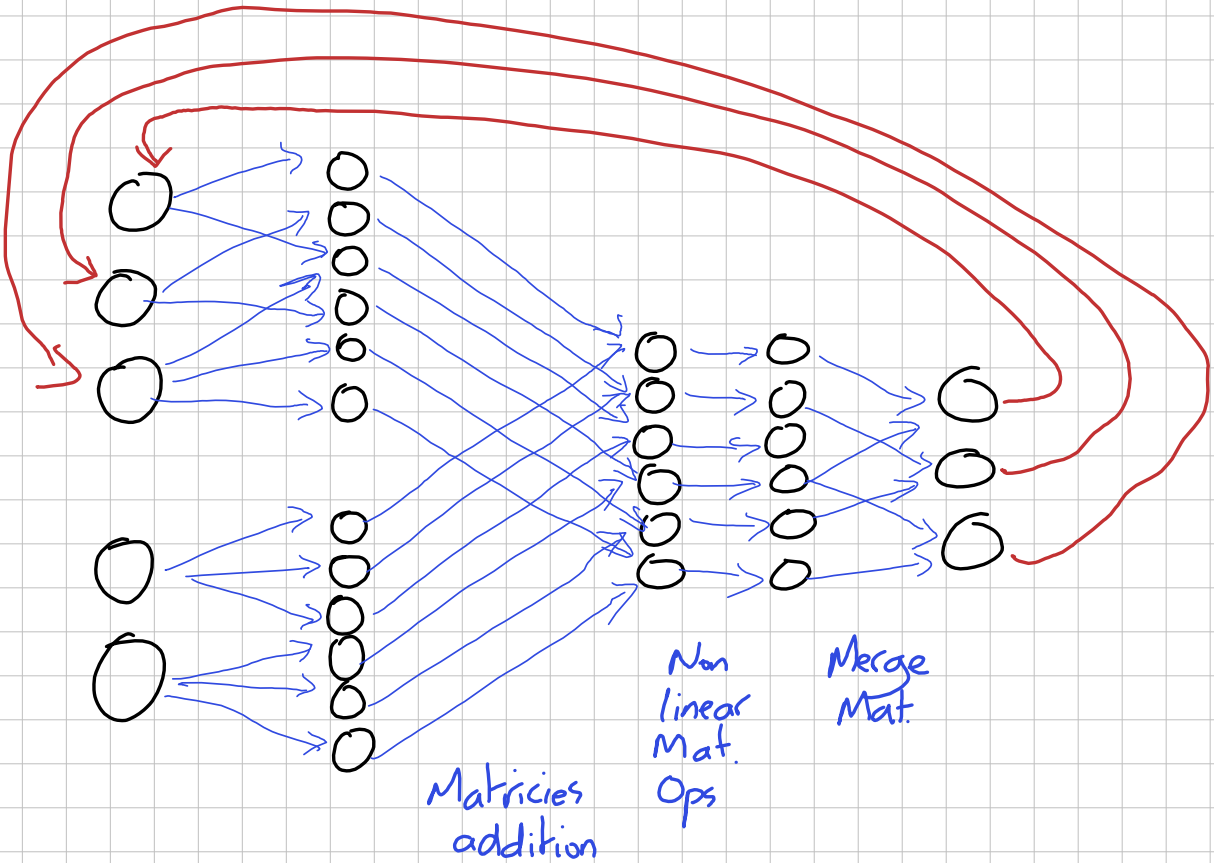


input layer | hidden layer | output layer

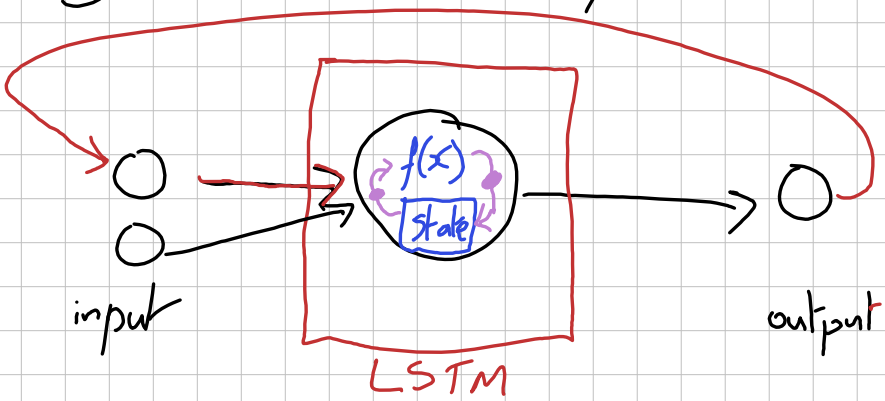
Convolutional Network



Recurrent Neural Networks



Long Short-term Memory Networks

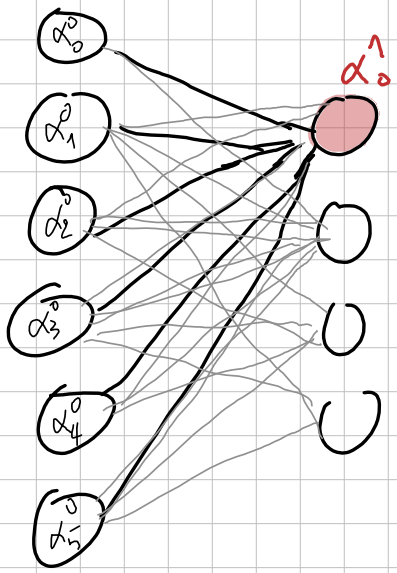


good in learning sequences of data

Activation functions

- Sigmoid (classic)

$$\sigma = \frac{1}{1+e^{-x}}$$



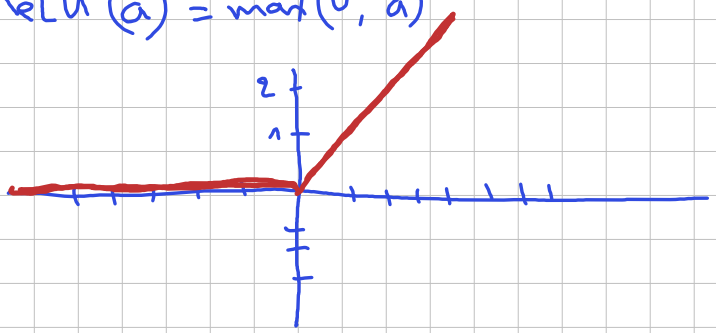
$$\alpha_0^1 = \sigma\left(w_{0,0}\alpha_0^0 + w_{0,1}\alpha_1^0 + \dots + w_{0,n}\alpha_n^0 + b_0\right)$$

$$\sigma\left(\underbrace{\begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & w_{0,n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} \alpha_0^0 \\ \alpha_1^0 \\ \alpha_2^0 \\ \alpha_n^0 \end{bmatrix}}_{a^0} + \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ \\ b_n \end{bmatrix}}_{b}\right)$$

$$\alpha_0^1 = \sigma\left(W a^0 + b\right)$$

- ReLU (Rectified Linear Unit)

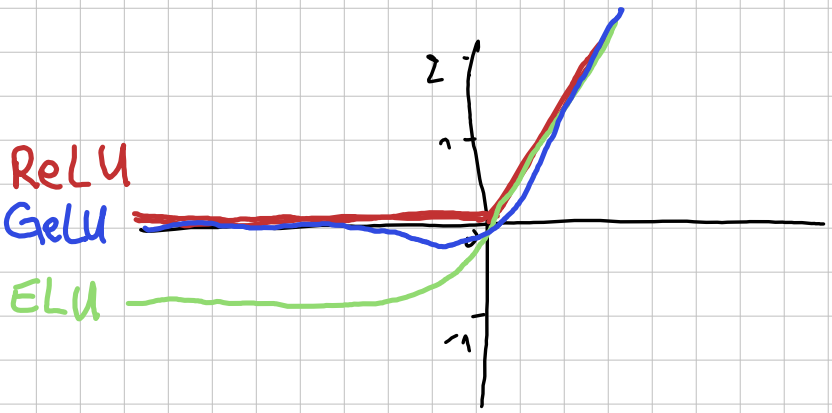
$$\mathrm{ReLU}(a) = \max(0, a)$$



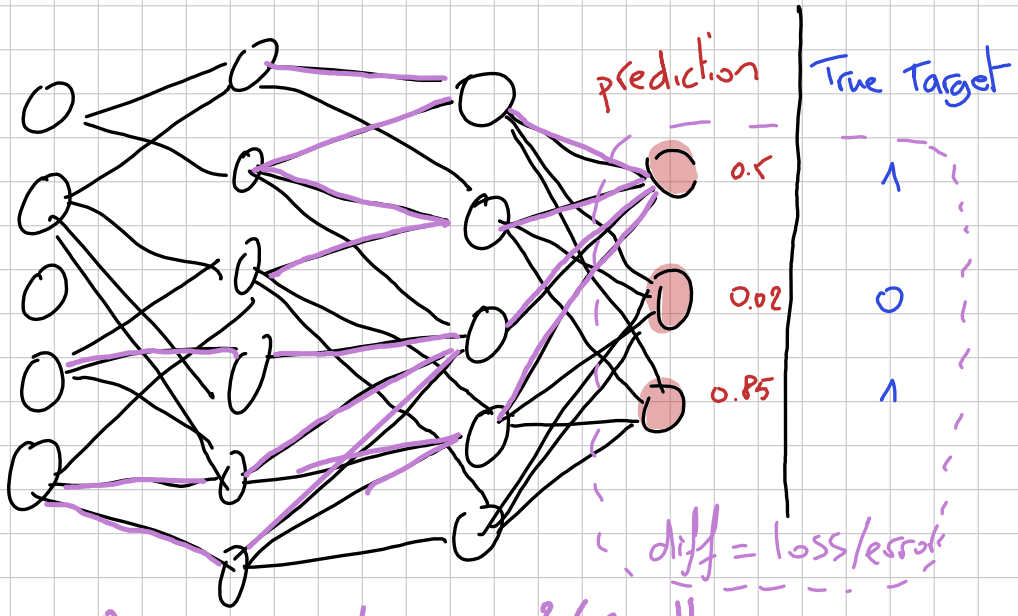
Motivated by biological analogy

↳ ReLU well for very deep N.N.

- GeLU (Gaussian Error Linear Unit)



Backpropagation



$w^0 b^0$, w^1 / b^1 , w^2 / b^2 ⇐ change

Challenges:

- Exploding Gradients
  - ↳ due to accumulation from deep network
  - ↳ Solution : gradient clipping.
- Vanishing Gradients
  - ↳ due to very small weights in earlier layers
  - ↳ solution : residual connections (add input layer to its output) + layer normalisation.
- Space / time Complexity
  - ↳ Sochastic Gradient (Batches of loss function)

Tranformer Architecture