Machine Learning with python

MNIT, Jaipur

Falak Shah
Lead Research Scientist
Infocusp Innovations Pvt. Ltd.

Contents

Machine Learning

- Machine learning is the science of getting computers to act without being explicitly programmed.

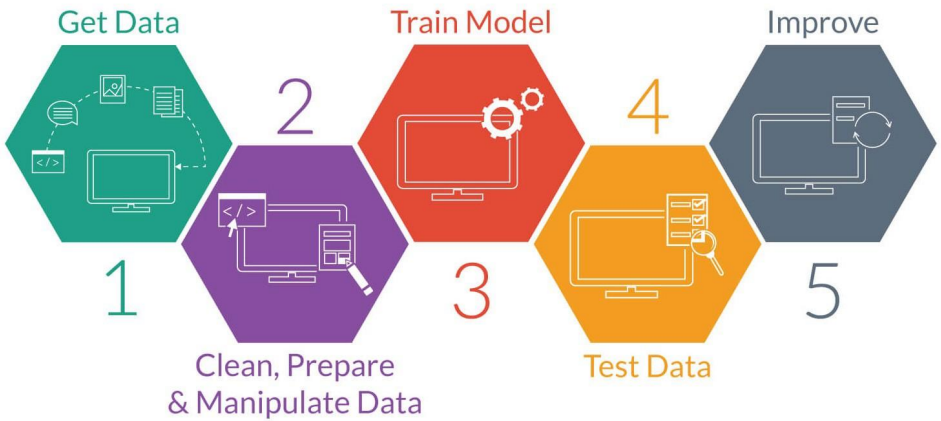


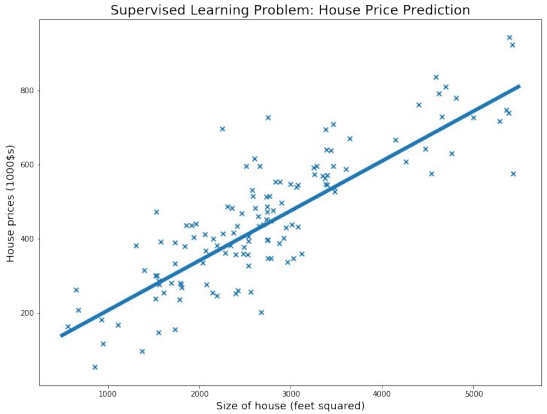
Image Courtesy: https://upxacademy.com/artificial-intelligence-foundation/

Supervised Learning

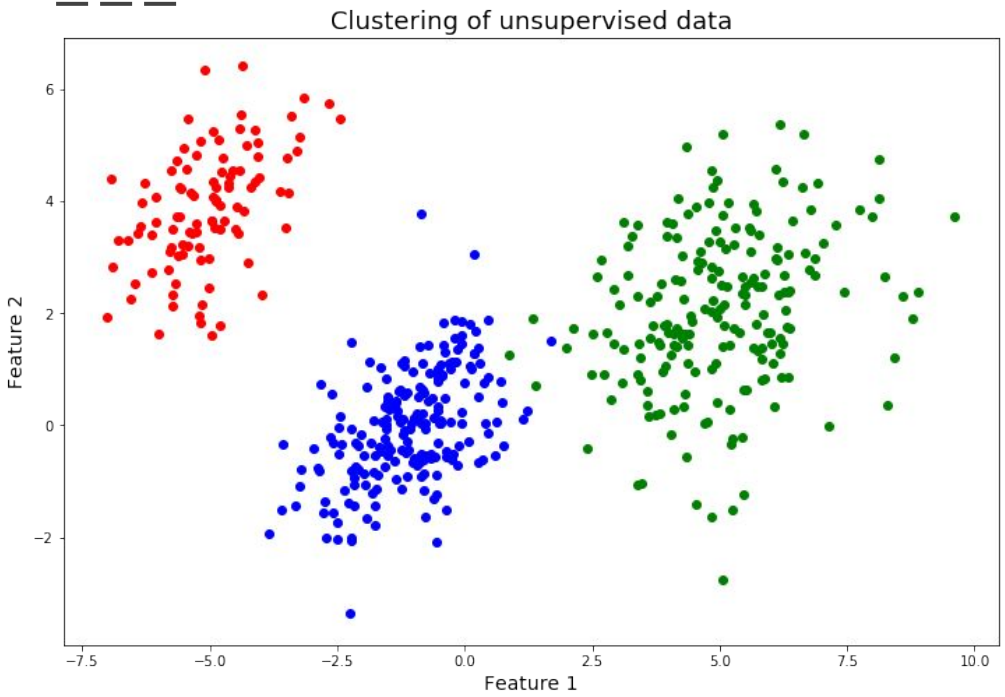
Input data with correct output labels which are used by machine in learning process.

Example: Housing prices

Area ($feet^2$)	Price (1000$s)
1210	234
1450	266
750	172
1200	232
1380	257
840	184



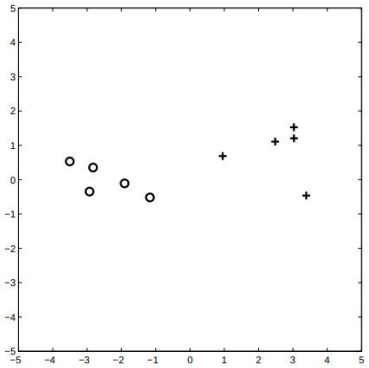
Unsupervised Learning



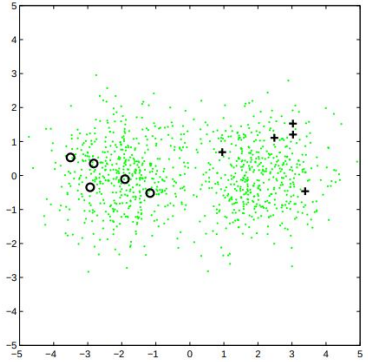
Actual labels are not provided in the data. Patterns and clusters are determined from the given data.

Example - Clustering news stories into categories

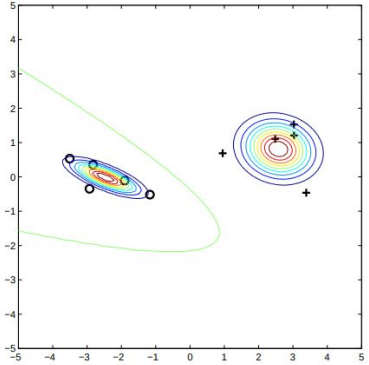
Semi-supervised Learning



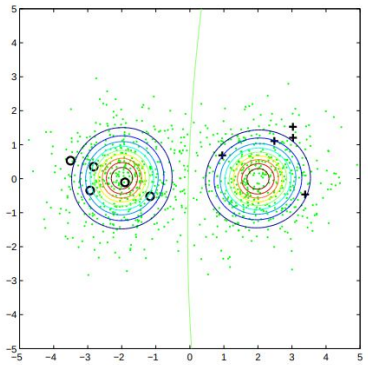
(a) labeled data



(b) labeled and unlabeled data (small dots)



(c) model learned from labeled data



(d) model learned from labeled and unlabeled data

Unsupervised Learning can be interpreted as:

- Supervised Learning + Additional Unlabelled Data
- Unsupervised Learning + Additional Labelled Data

Here, we are trying to predict Gaussian distributions that generated the given data.

Interested folks can read up more on Expectation Maximization (EM)

Reinforcement Learning

———

The "cause and effect" idea can be translated into the following steps for an RL agent:

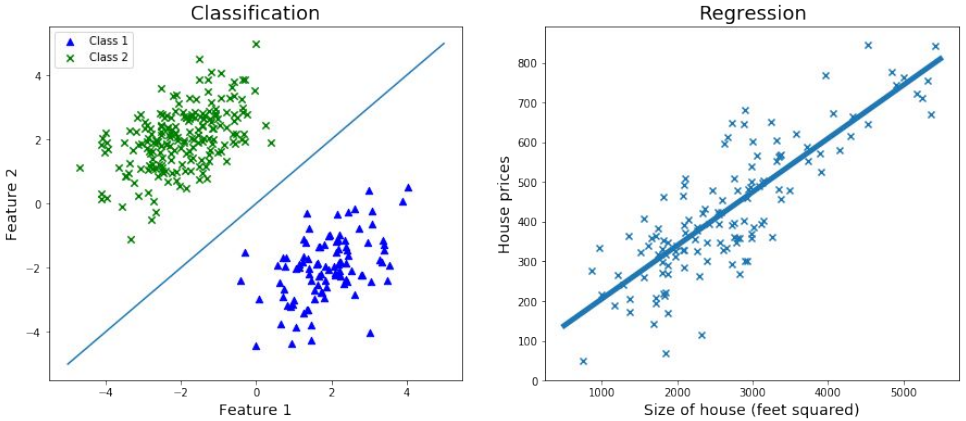
1. The agent observes an input state
2. An action is determined by a decision making function (policy)
3. The action is performed
4. The agent receives a scalar reward or reinforcement from the environment
5. Information about the reward given for that state / action pair is recorded

Supervised Learning

Given: Pair of input and output values.

Goal: Define a loss function and minimize that to predict accurate output values.

Regression and Classification problems



Examples:

1. Classify images into different classes.
2. Housing price prediction based on features of house.

Linear Regression

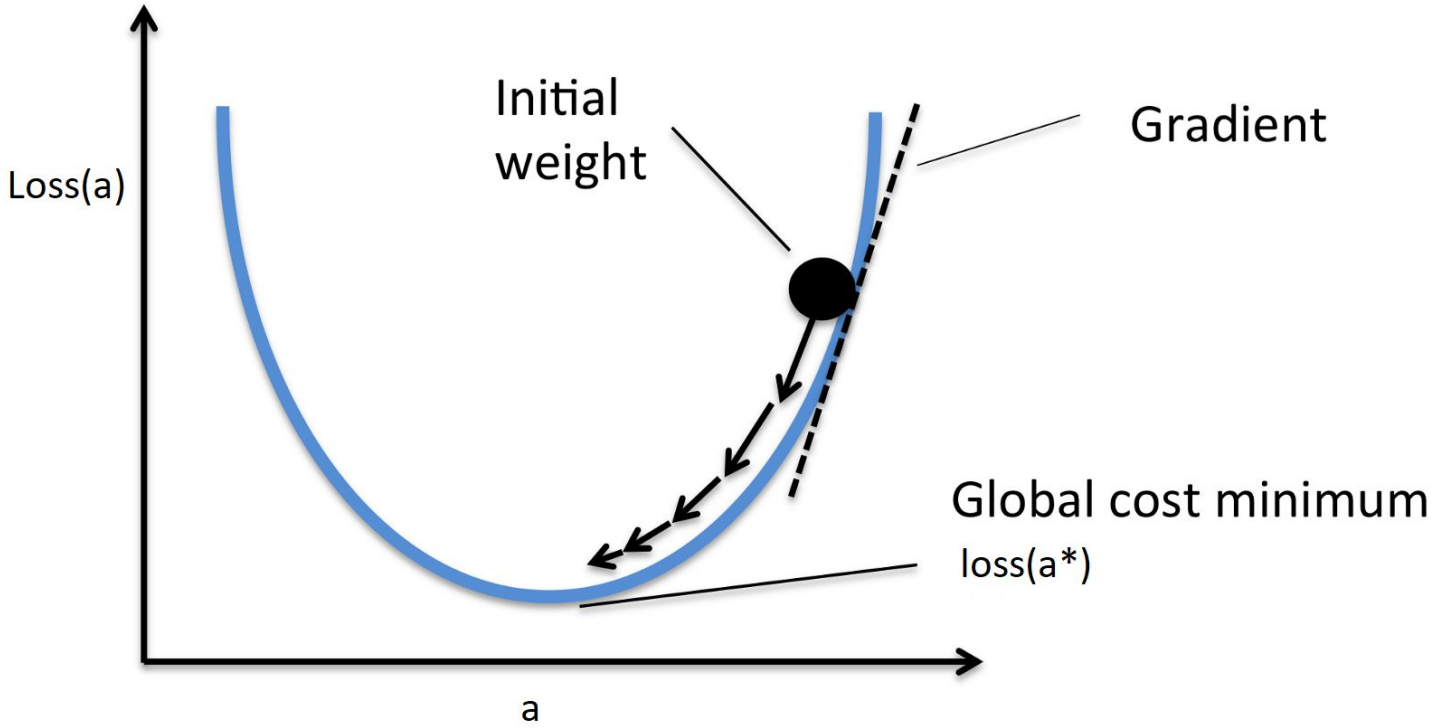
One of the simplest ML algorithms
Trying to find a linear mapping from input to output
Example

$$y = ax + b$$

$$J = \frac{1}{2}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

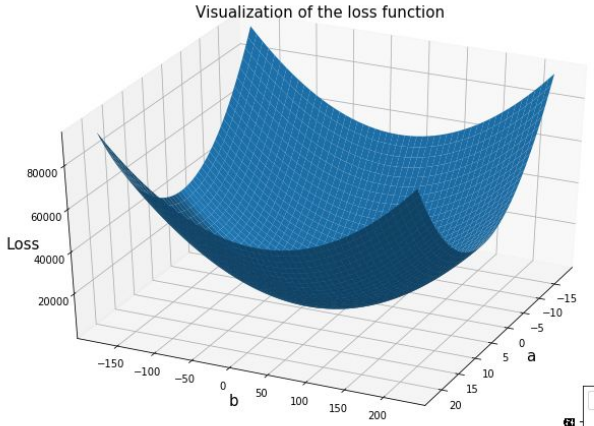
$$(a_{opt}, b_{opt}) = \underset{a,b}{\operatorname{argmin}}(J)$$

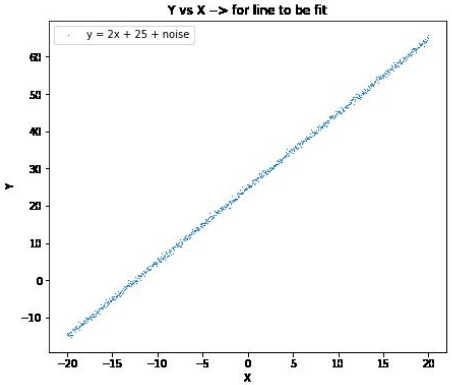
Gradient Descent Intuition



https://github.com/falaktheoptimist/gradient_descent_optimizers/blob/master/Optimizers%20Linear%20Regression%20example.ipynb

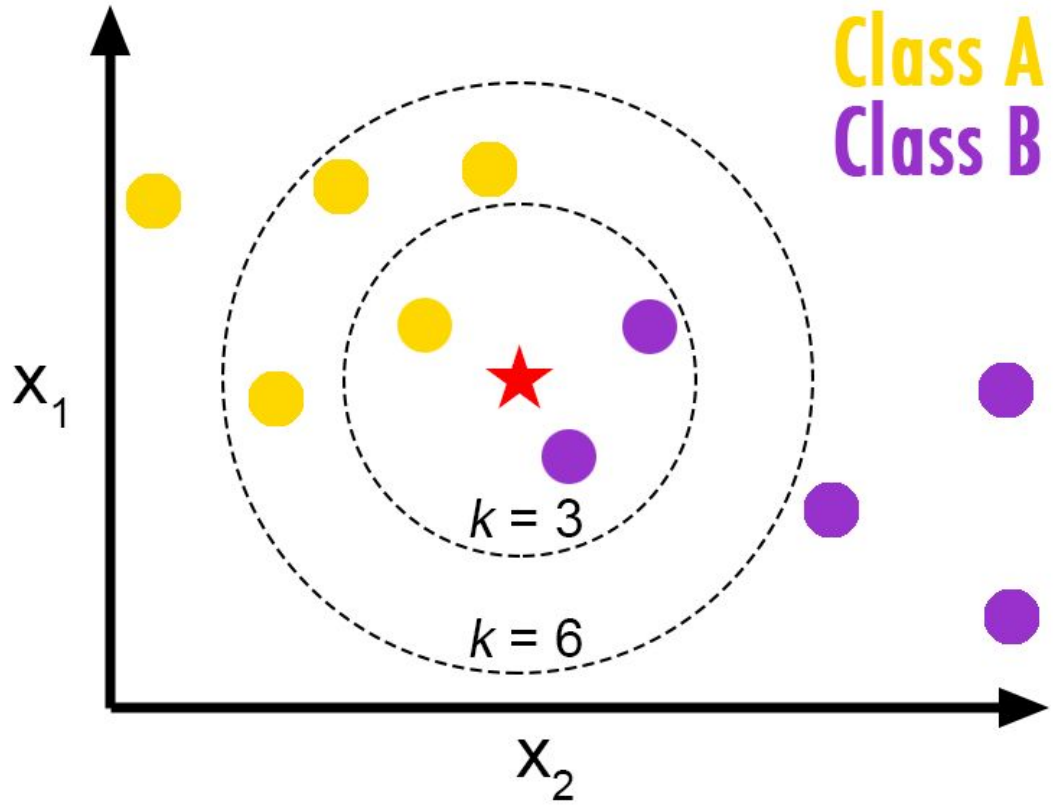
Data and loss visualization





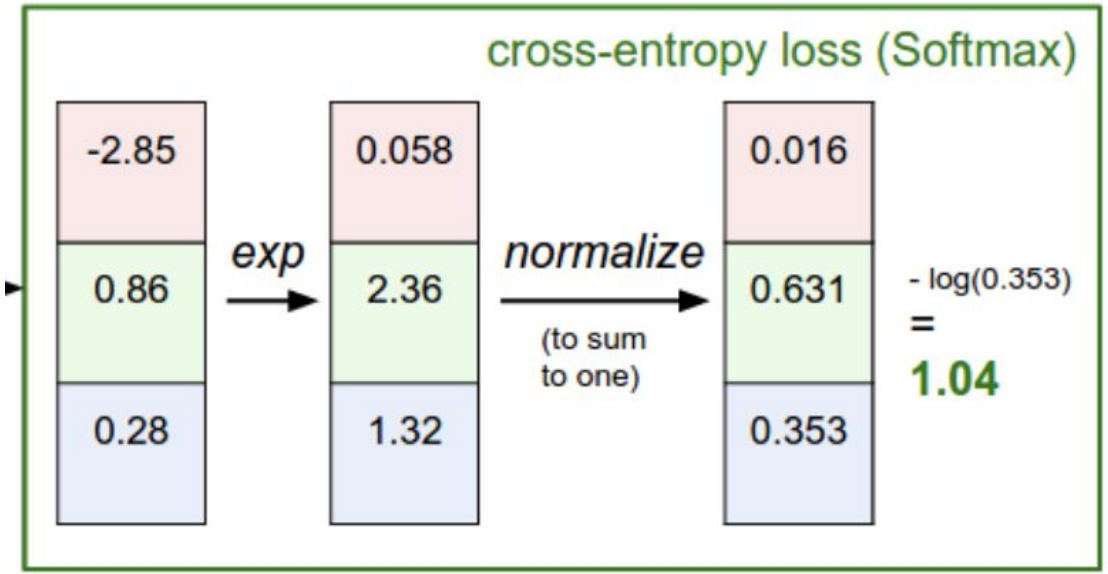
Classification: Simple approach

- Find k nearest neighbours
- This is why ML is intuitive
- No training needed
- Vectorized approach possible (50 seconds with for loops to 0.5 seconds with matrices)
- Important to learn linear algebra
- CS231n - Stanford - Begins with this
- How to decide k? K folds holdout accuracy


Class A
Class B
x1
x2
k = 3
k = 6

Classification problem

Softmax Loss

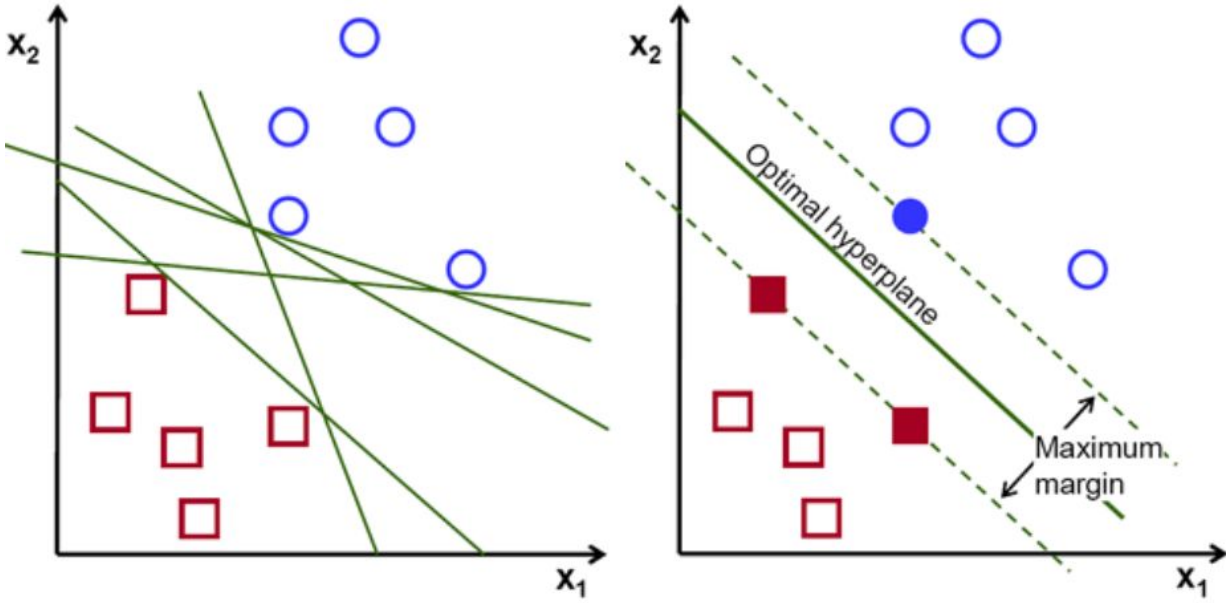


$$t = \begin{bmatrix} t1 \\ t2 \\ t3 \end{bmatrix}$$

$$loss = -\sum_{i=1}^{3} y_i log(P_i)$$

$$\text{where } P_i = \frac{e^{ti}}{\sum_{j=1}^{3} e^{tj}}$$

Support Vector Machines



Kernels: to handle non linearly separable data

Take advantage of computing just the distance metric in higher dimension

Can map upto infinite dimensional hyperspace (RBF kernel)

Learning from Data by Prof Yaser of Caltech

Excellent explanations on the kernel trick

Depending on the complexity of function you fit

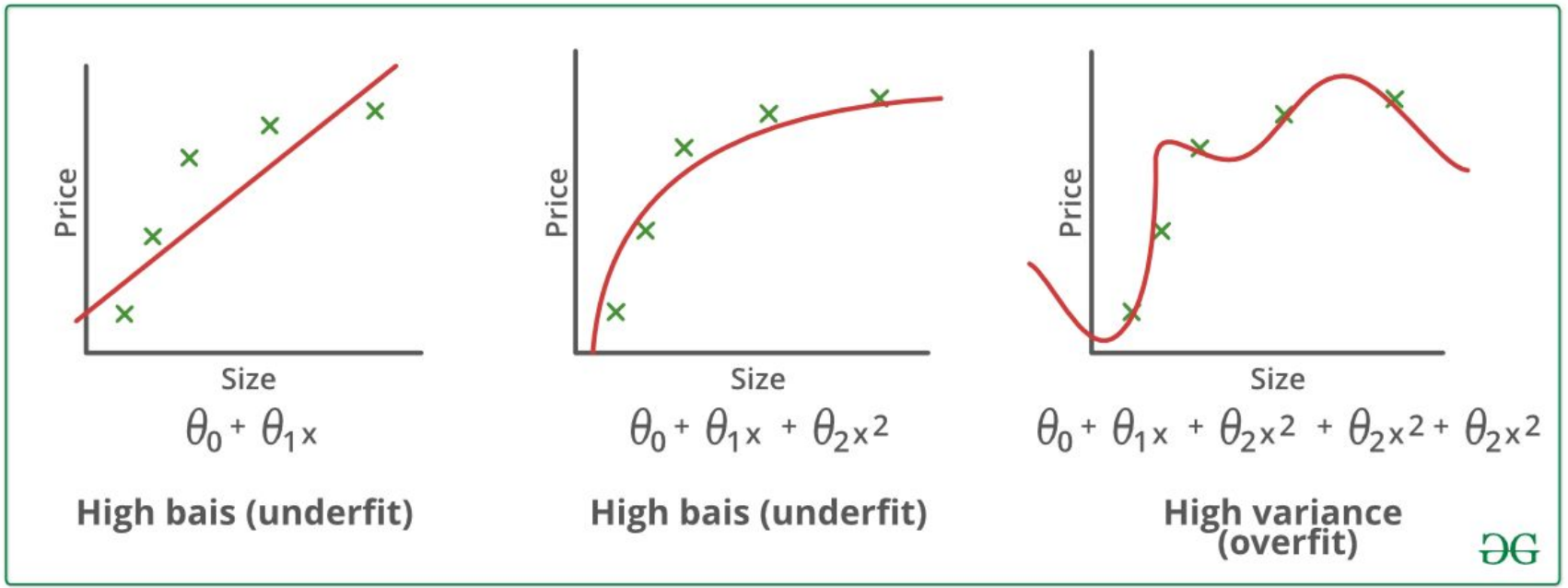


Image courtesy: Geeksforgeeks

Let’s try some hands on

– – –

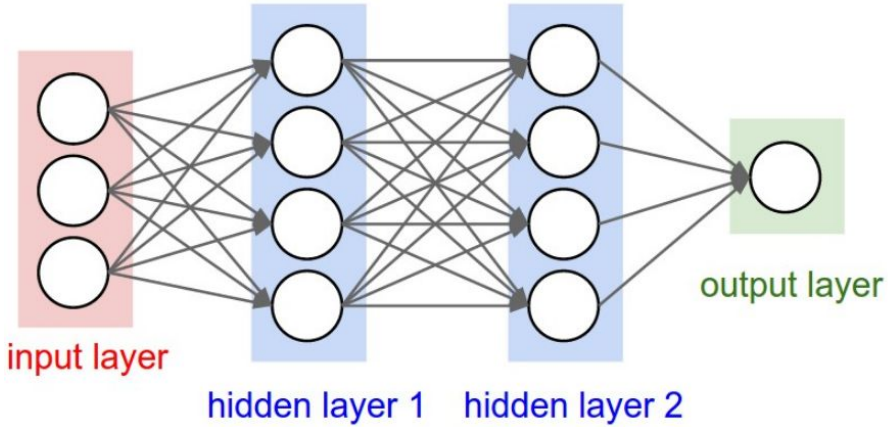
http://localhost:8889/notebooks/tutorial.ipynb

https://colab.research.google.com/drive/1buDdba1UmgXYsXnE02ZZJSVJSjripGqF

Break?
Back to deep learning

Artificial Neural Networks

- ANNs are used for modelling Non-linear hypothesis



H = activation(WX + b)
Y_pred = activation(WH + b)

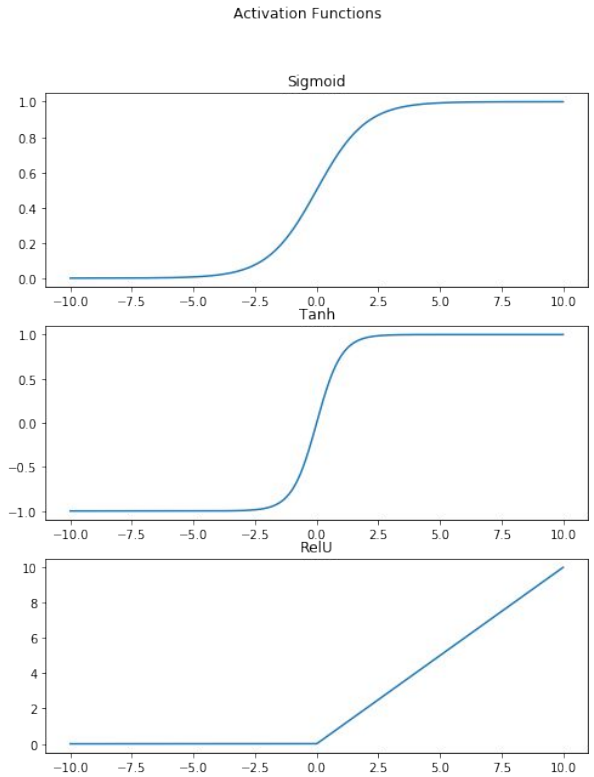
Loss = f(y, y_pred)

Softmax loss = $-\log(p_{y\_true})$

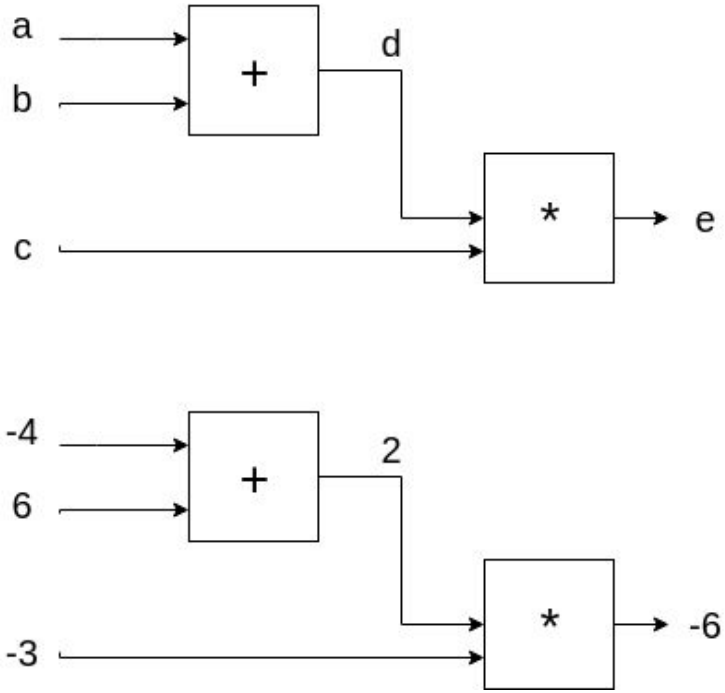
Image Courtesy: http://cs231n.github.io/neural-networks-1/

Activation Functions

1. Sigmoid/ Logistic
2. Tanh - Hyperbolic tangent
3. ReLU - Rectified Linear Units



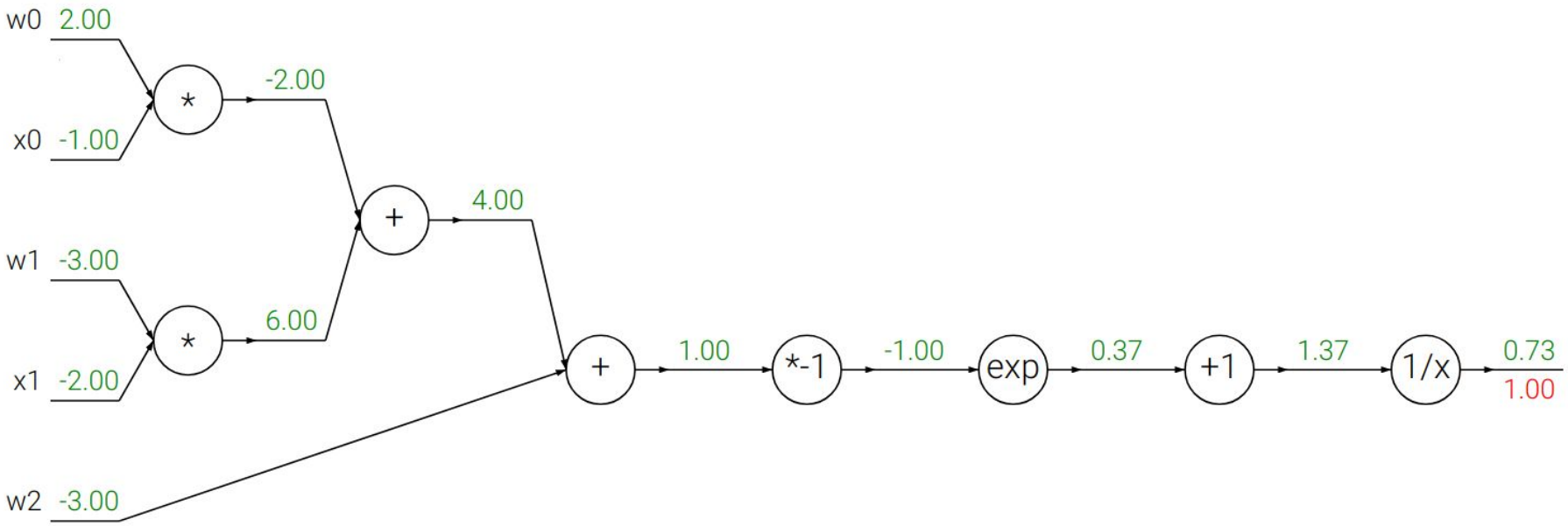
Backpropagation Intuition



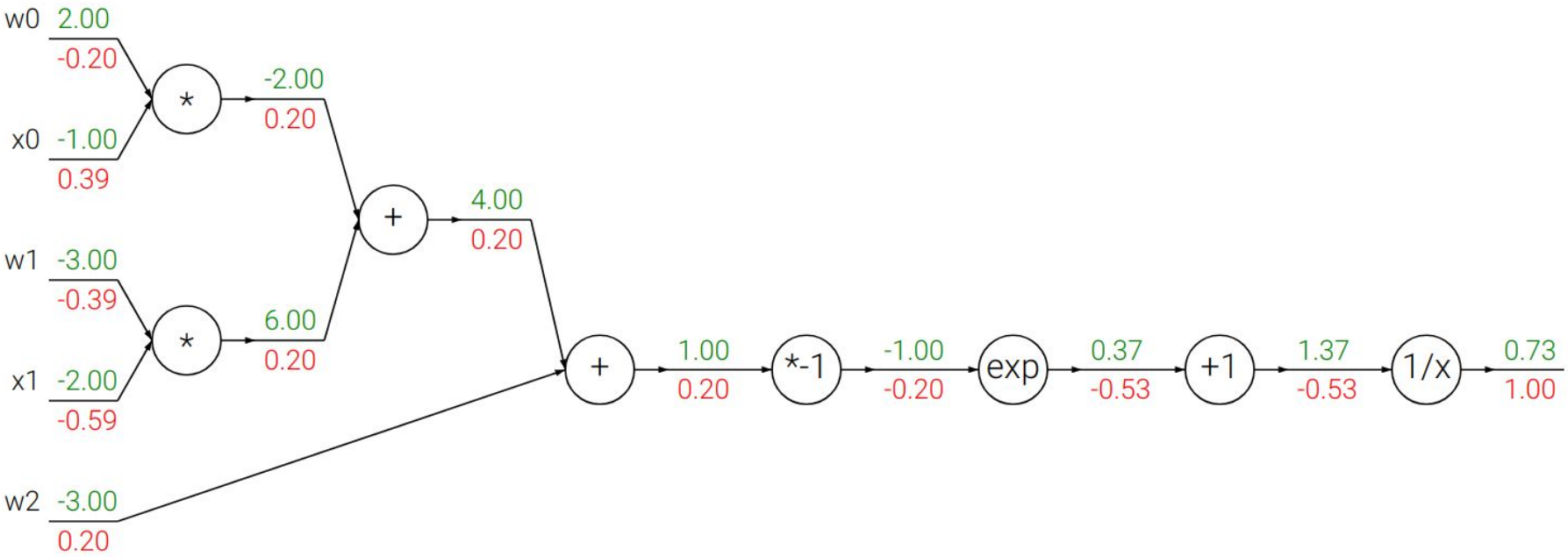
Once we have the loss at output layer, we want to back propagate that till the input layer such at we penalise the appropriate cells according to their contribution in the loss.

$$d(a,b) = a + b$$
$$e(a,b,c) = (a+b)*c$$

Back-prop Example



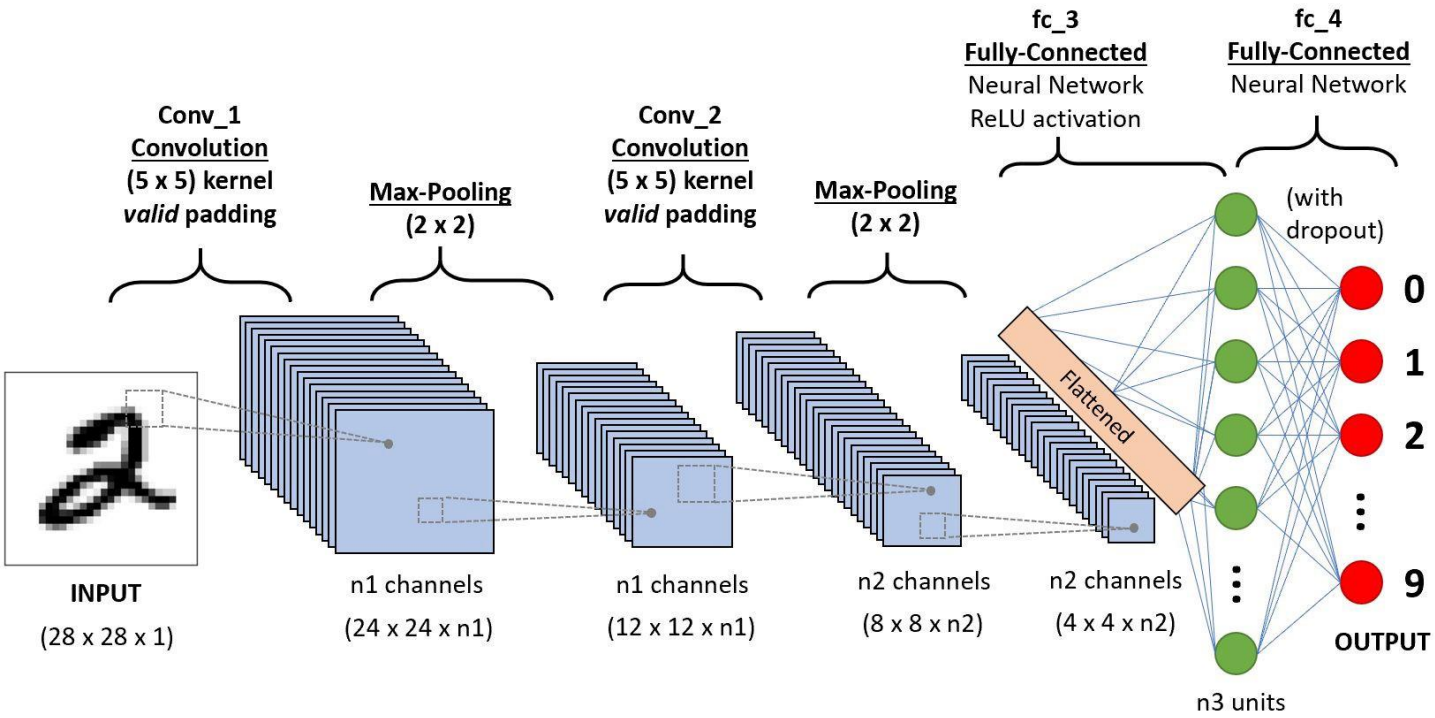
Back-prop Example



Optimizers

- Different methods of taking steps when descending to the minimum of loss
- Hands on

Convolutional Neural Network

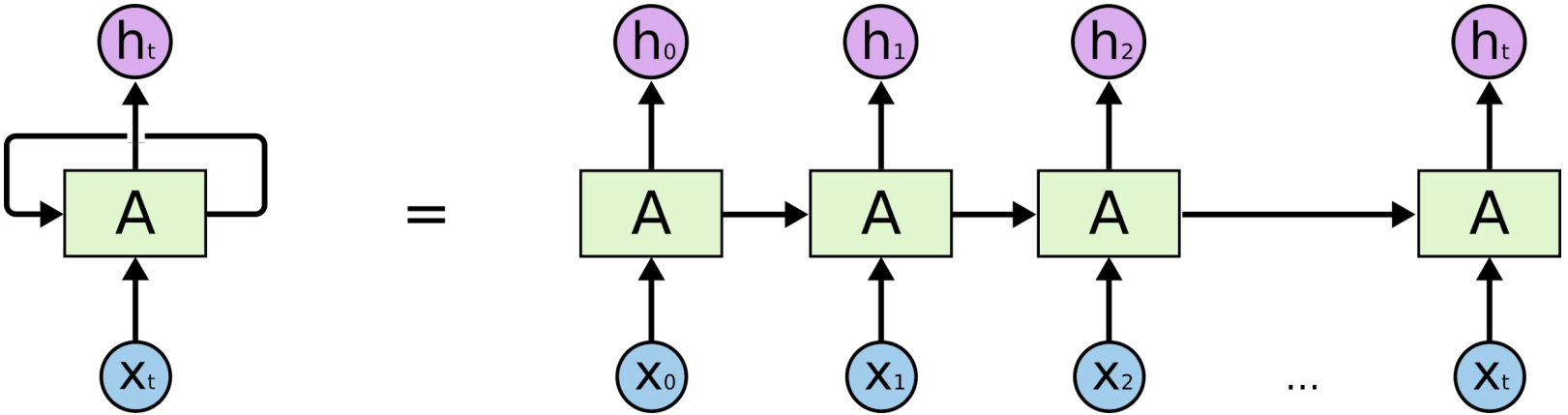


CNN used in

- Image classification
- Image segmentation (Bunch of contests on kaggle)
- Extracting features for caption generation
- Demo : caption generation
- Train CNN on MNIST in under 2 minutes: Colab
- Colabs and GPUs freely available - so are kaggle kernels
- Object detection from images : fast RCNN

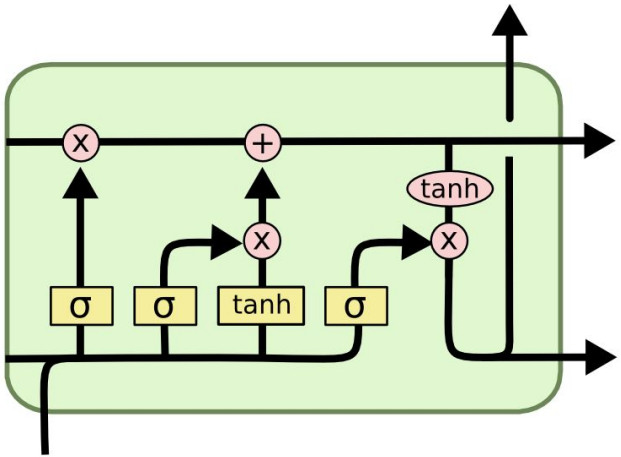
Recurrent Neural networks/ LSTMs

- Variable length sequences with dependencies across time
- Cannot be captured by static ANN/ CNN
- Try out on a small problem for understanding and then see its application:Link1 and Link 2



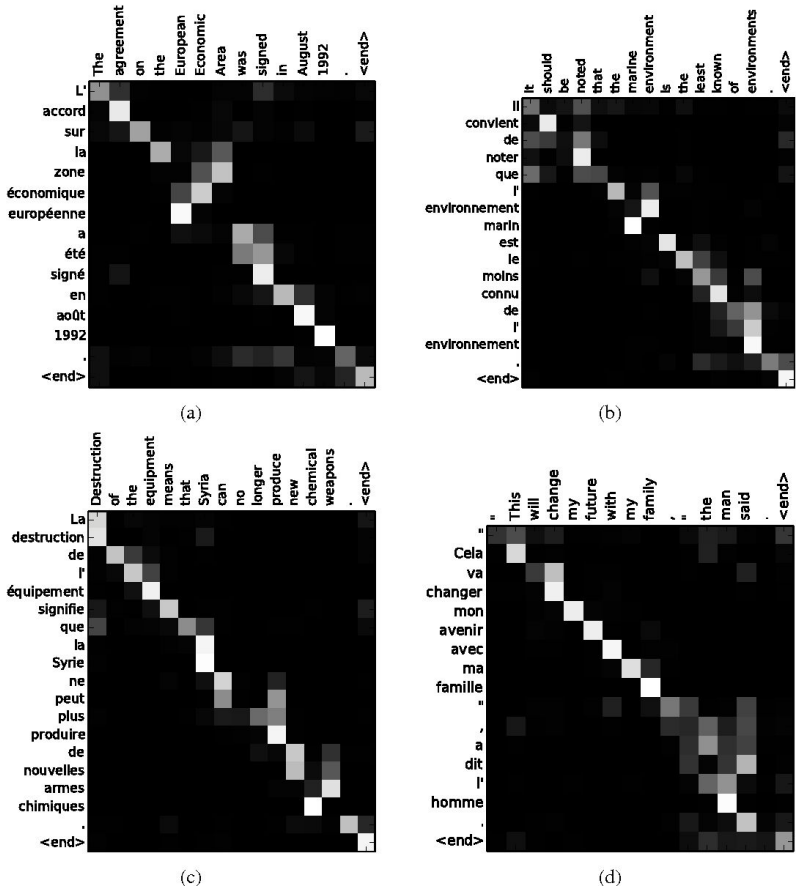
Some interesting LSTM applications

- Handwriting generation demo (Uses attention mechanism)
- Generated music using LSTMs: Link
- Stock trends prediction
- Copying Shakespeare



Attention based networks

Has proved effective for machine translation tasks



Tips for data scientists contd.- ML specific

Validation

- Check, double-check, triple-check if the model is learning generic concepts, and not just regurgitating the information contained in the training data set.
- Use advanced cross-validation techniques suitable for your data set and domain
- Keep various test data sets, untouched/unrelated from training process. Ideally mimicking the real-life situation where you want the models to be used.
- Using a pool of models, and selecting ones that have good results on validation set is often a good ploy
- External validation is a must.

Be algorithm agnostic

- Don’t make a particular algorithm (or a particular class of algorithm) your favourite.
- Best algorithm is the one that solves your problem most effectively.

Employ good programming practices

- Many ML practitioners write sloppy code. Because their work is exploratory. (Roughly <5 out of 10/15/100 models they create go into final production).
- Good code means better re-using of code!
- Use of comments
- Good code architecting.
- Generic bits at generic places. Specific at specific places
- Object oriented when possible
- Refactoring when possible
- Module / Package structuring
- All this even if you are 1-person team and only user of your code!

Enhance knowledge of ML systems

Specifically

- Databases
- Distributed systems
- Cloud computing

Keep an eye on stability

- For your problem, you often don't know how quickly the data will grow!
- Some algorithms / approaches are prohibitively slow if the data becomes big. Make sure you know that - and keep some breathing space.

My pathway

Machine Learning course by Andrew NG

Some bits from Learning from data by Caltech

Statistical Learning Theory by Tibshirani (Stanford)

CS231n

7 MOOCs

Kaggle contests

Most importantly - learn linear algebra/ probability/ stats

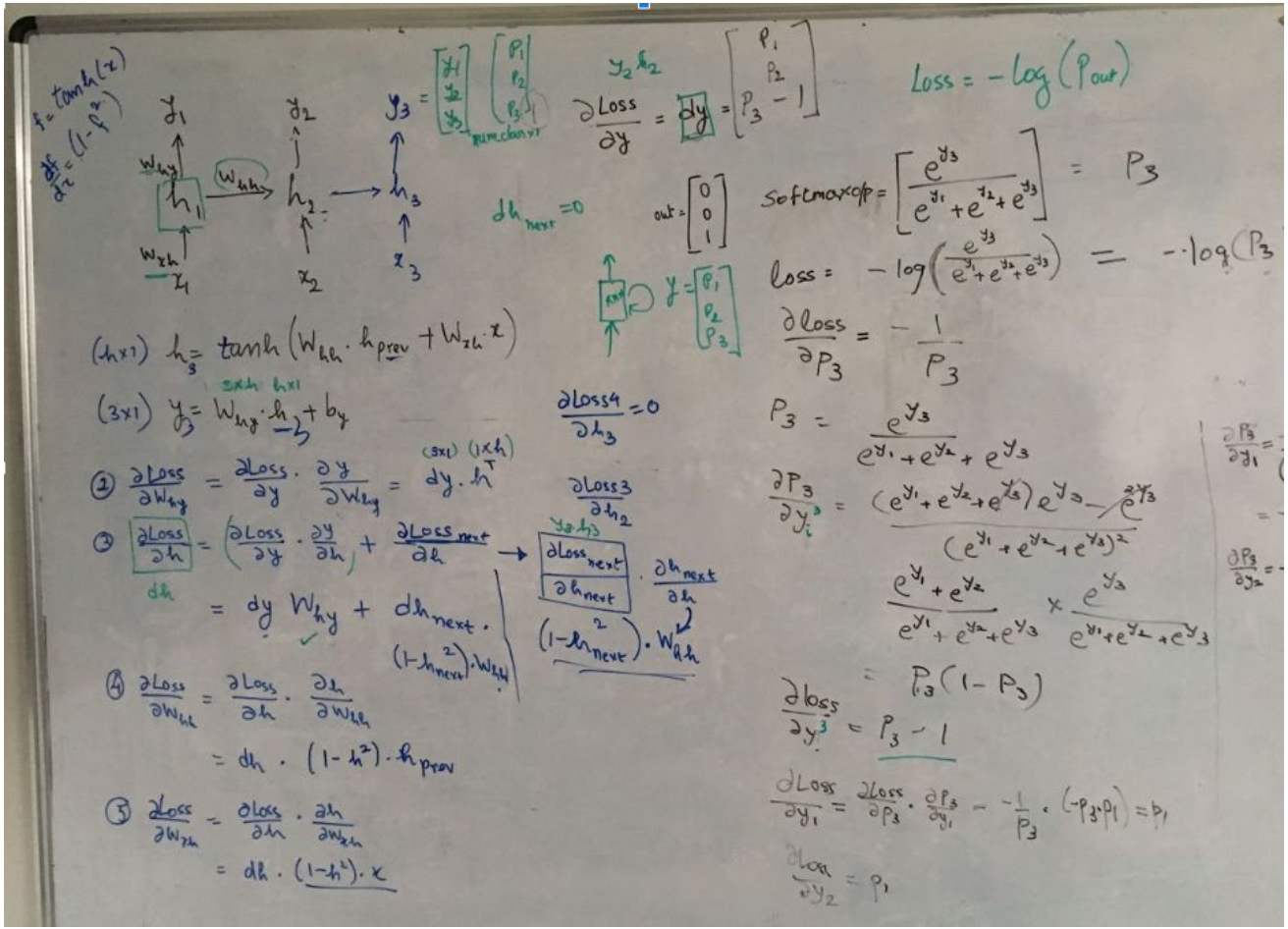
Some good practices for ML/ DL projects

- **Make sure it can be replicated**
- Inputs/ outputs for all steps in the pipeline must be clear
- Make the pretrained models available online
- Share the project work on open source whenever possible
- Have full knowledge of the black boxes you use

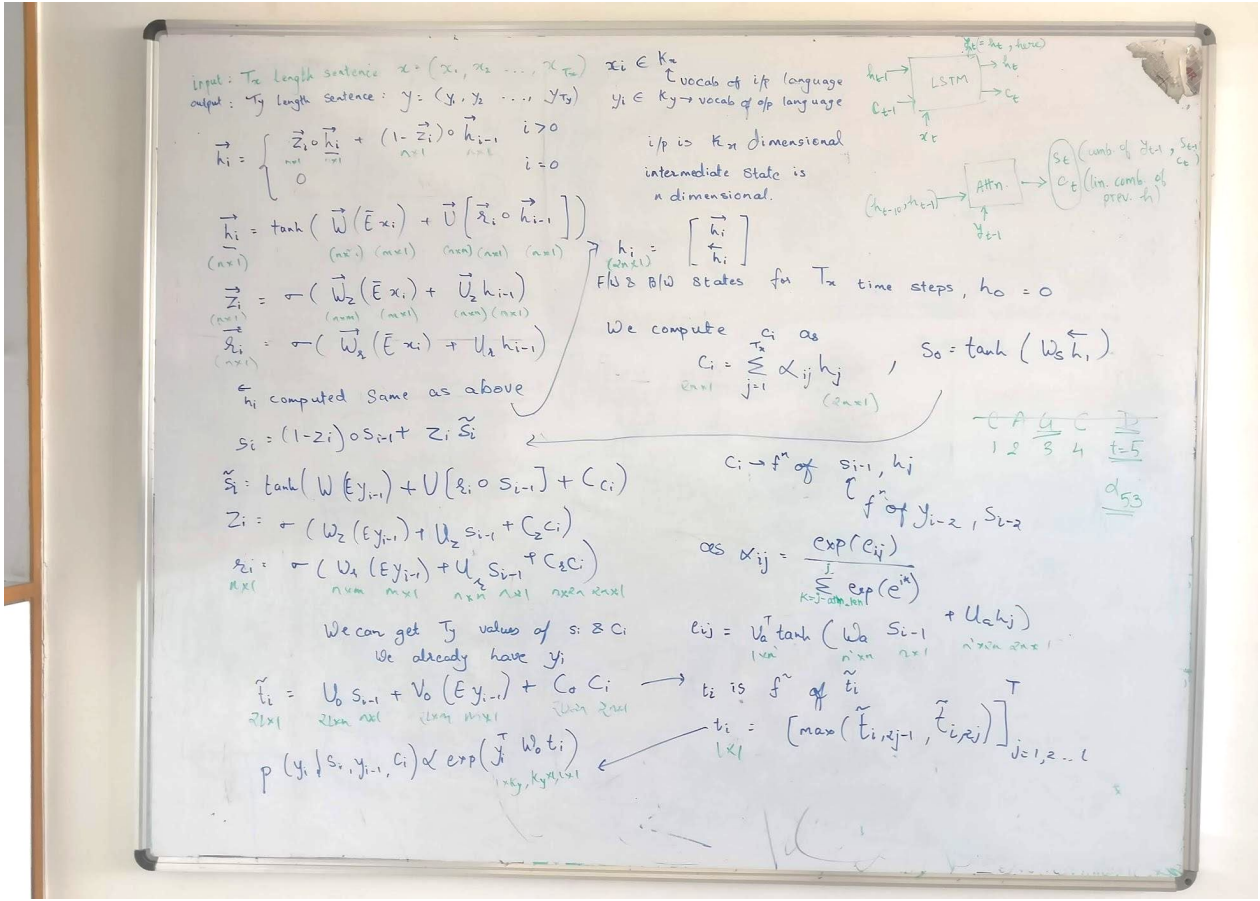
Keeping up to pace with a growing field

- Read papers- required if you want to stay up to date
- Discuss - reading groups help keep up the pace
- Start simple and build up (almost upto 120 papers)
- We just whiteboard it and then dive into the code as inputs outputs are clearer there
- ICLR, ICML, ICCV, CVPR, NeurIPS, PAMI - some good conferences out there

RNN backprop



Attention models



Conclusion

- ML \ DM landscape changes rapidly, with (smarter, yay!) people getting more and more familiar\comfortable with the techniques.
- Avoid pitfalls of overfitting.
- To use ML in industrial settings: just make sure you keep in mind the advisory nuggets (to certain extent). Special mention to
  - Know your data
  - Know your settings (business, customers, products etc.)
  - Know your maths
  - Know your tools
- The problem is not always well-defined. You have to define\design\transform a messy real-life problem into an ML problem.
- ML is possible in ***any*** field! - given that we have adequate data.