Soliton

Vision for a Better World

Computer Vision with Machine Learning

PSG Tech - 2018

The team

Dhivakar

Computer Vision and
Machine Learning,
Soliton Technologies

Senthil

Computer Vision and
Machine Learning,
Soliton Technologies

Soliton
Vision for a Better World

Day 2

Had Colorful Dreams..?

Agenda

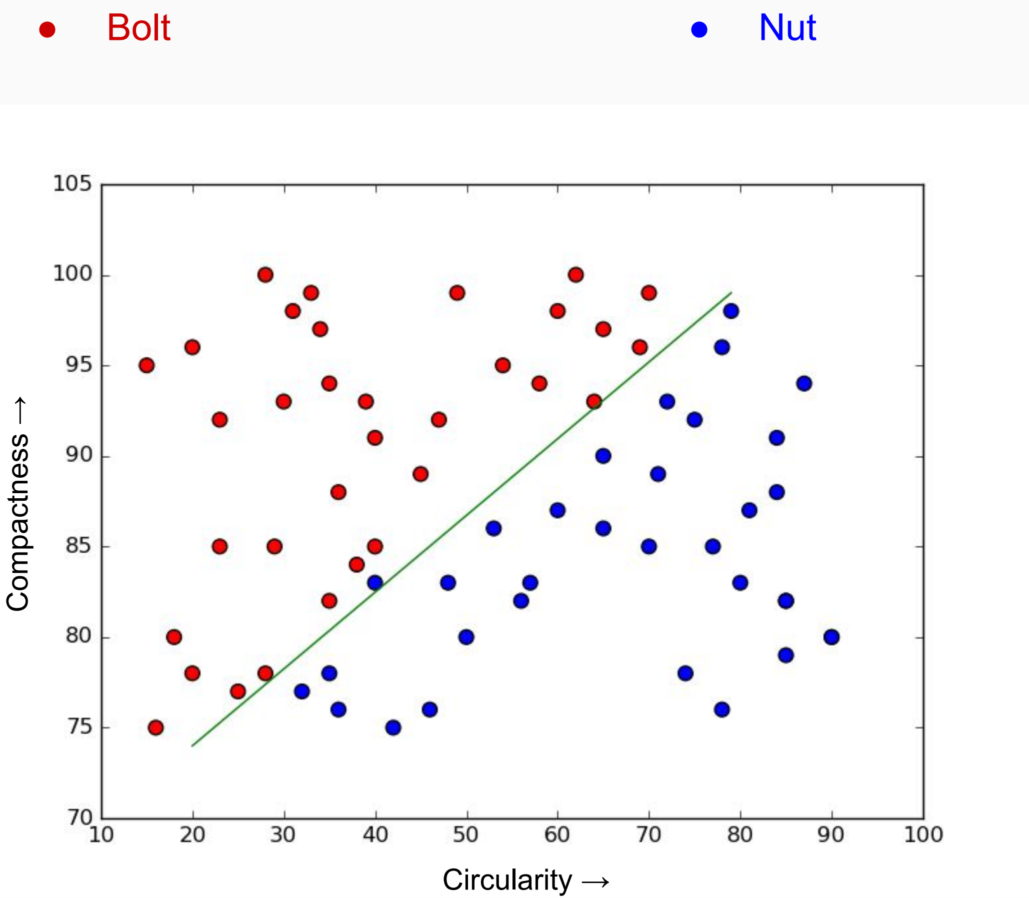
Session I

- Linear Regression
- Logistic Regression
- Over / Under Fitting
- SVM
- Other ML Techniques

Linear Classifier

A line should classify the data

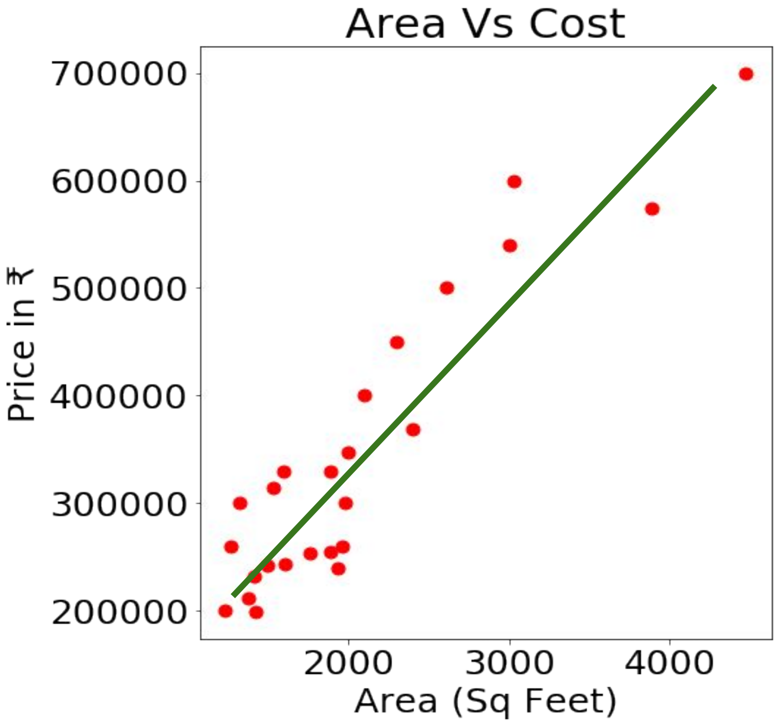
But, How do I come up with a Line which separates both the classes of data optimally?



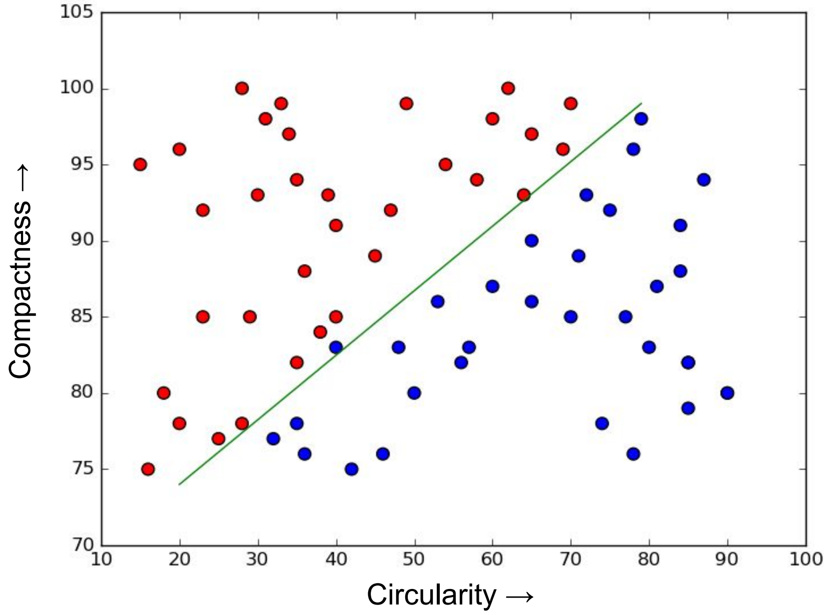
Soliton
Vision for a Better World

2 ML Problems

Regression

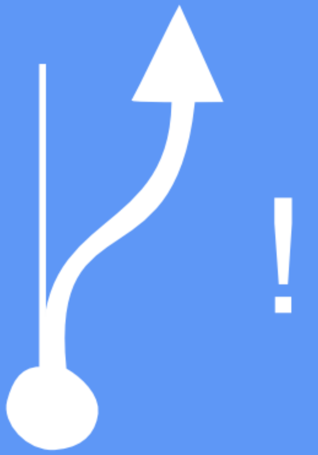


Classification



Soliton
Vision for a Better World

Let's Digress into Regression



Gradient Descent

- The magnitude of the update is proportional to the error term (y(i) − h θ (x (i)))
- If a training example on which our prediction nearly matches the actual y (i), there is little need to change the parameters

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

Batch Gradient Descent

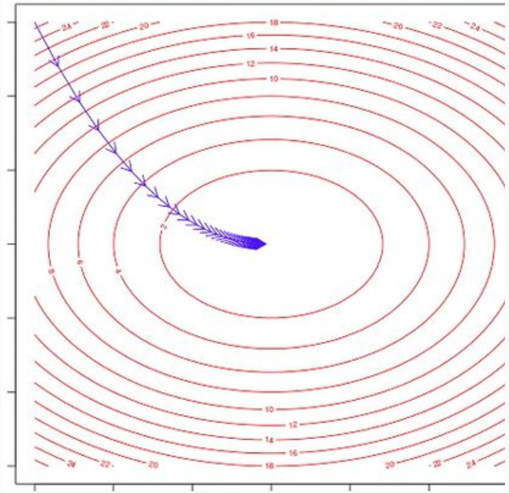
for i=1 to m, {

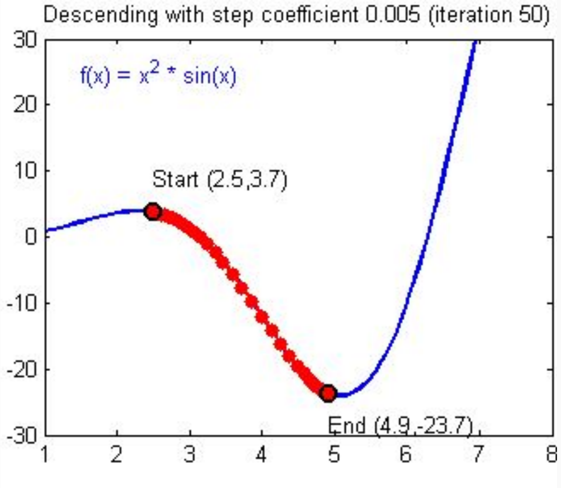
$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

}

Stochastic Gradient Descent (SGD)



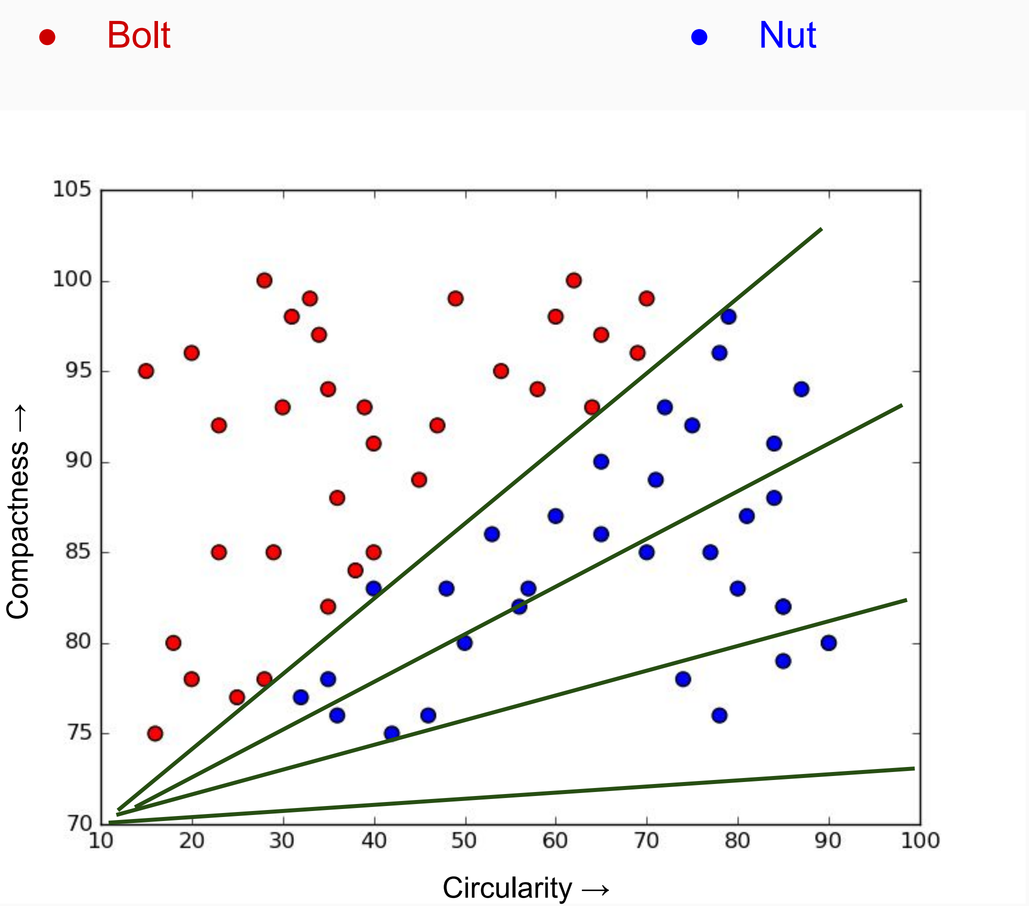


Soliton
Vision for a Better World

Linear Classifier

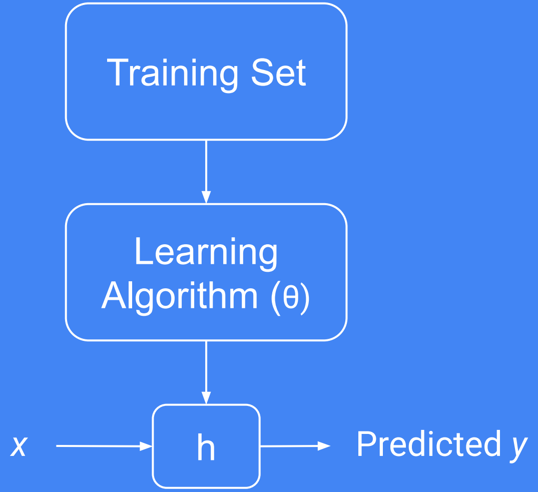
A line should classify the data

- Randomly Initialize a Line
- Calculate Error
- Move in a Random Direction
- Recalculate Error
- If Error reduced, move in same direction
- Else move in opposite direction



Soliton
Vision for a Better World

Parts of an ML Algo: ERM



Inference

- Hypothesis space, the set of possible hypotheses it can come up with in order to model the unknown target function by formulating the final hypothesis

- An example (linear function) $h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$

Learning

X - Training data   Y - Labels   h(θ) = Predicted label

- Cost function => J(θ) = h(θ) - y   [Predicted - Actual]

- Update Rule $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$

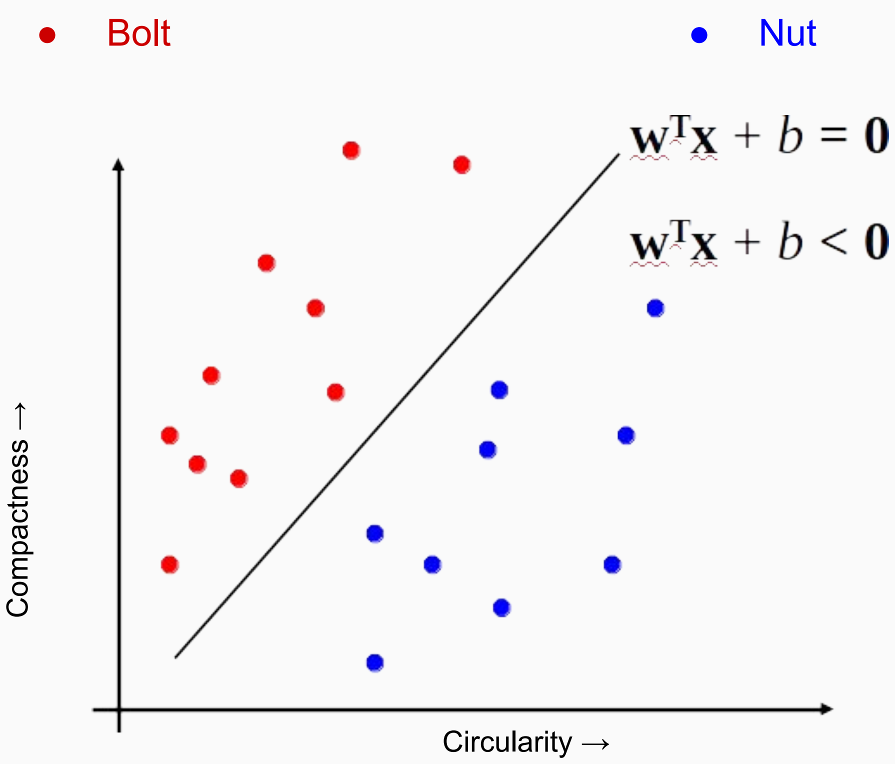
Soliton
Vision for a Better World

Logistic Regression

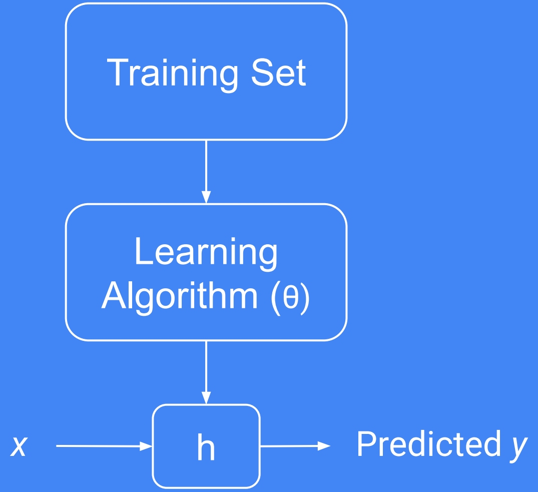
The Line can be drawn from the learned parameters

$$\theta_0 + \theta_1 X_1 + \theta_2 X_2 = 0$$

$$X_1 = \theta_0/\theta_1 + \theta_2 X_2 /\theta_1$$



Logistic Regression



Learning

X - Training data Y - Labels h(θ) = Predicted label

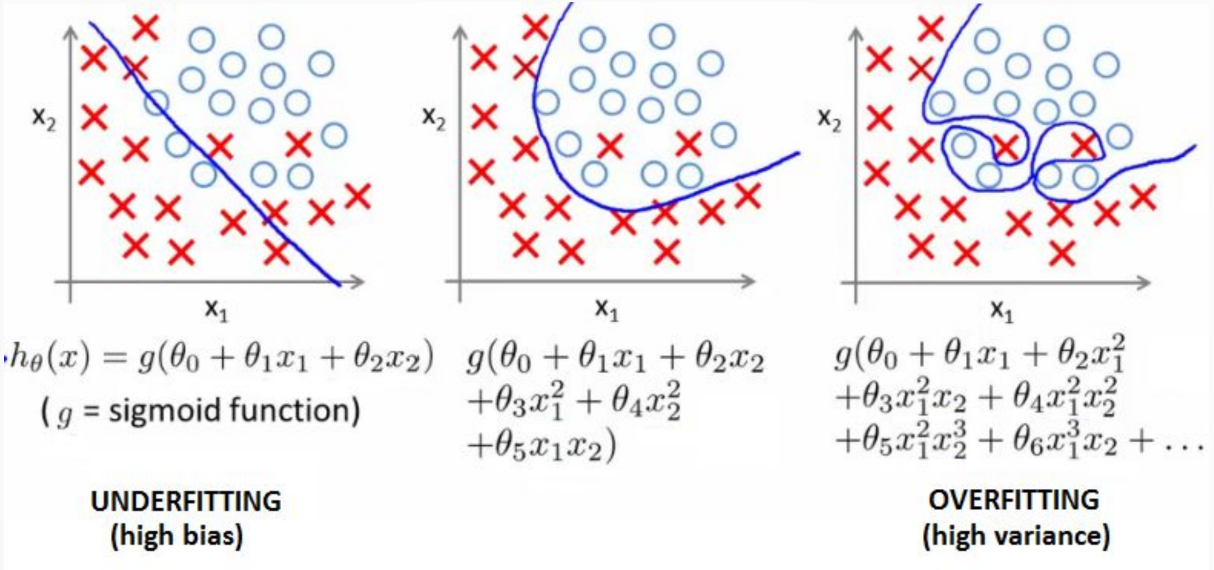
- This can be simplified as $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$
- Cost function $J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$
- Update Rule $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$

Gradient Descent: Starts with some initial value of θ and repeatedly performs update

Soliton
Vision for a Better World

Overfitting & Underfitting

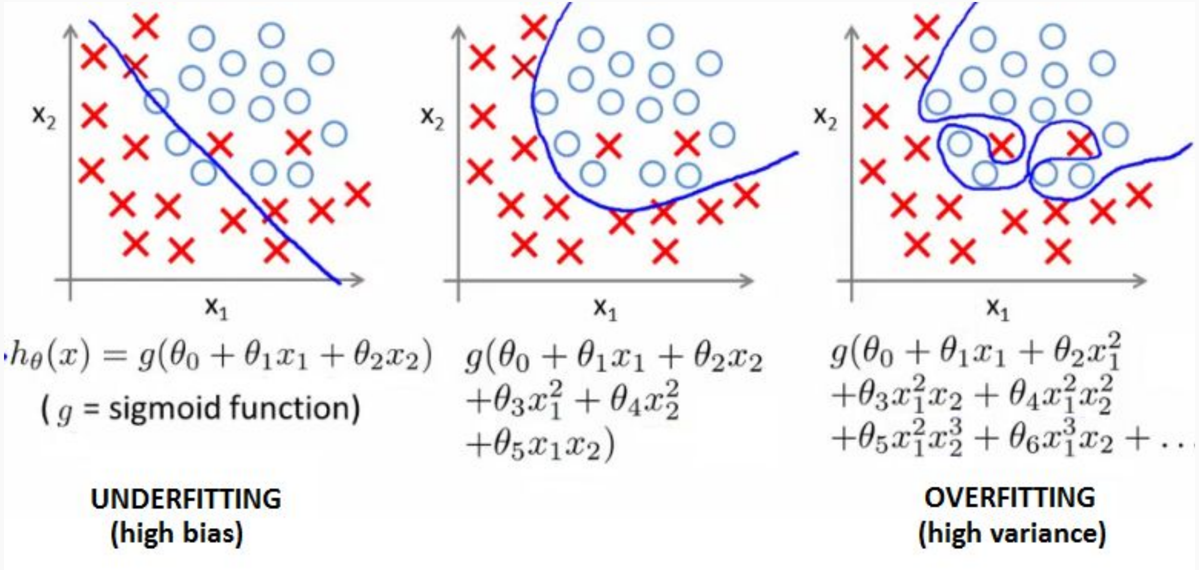
1.



Overfitting & Underfitting

How to Fix

1. Cross Validation
2. Feature selection
3. Regularization



Regularization

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

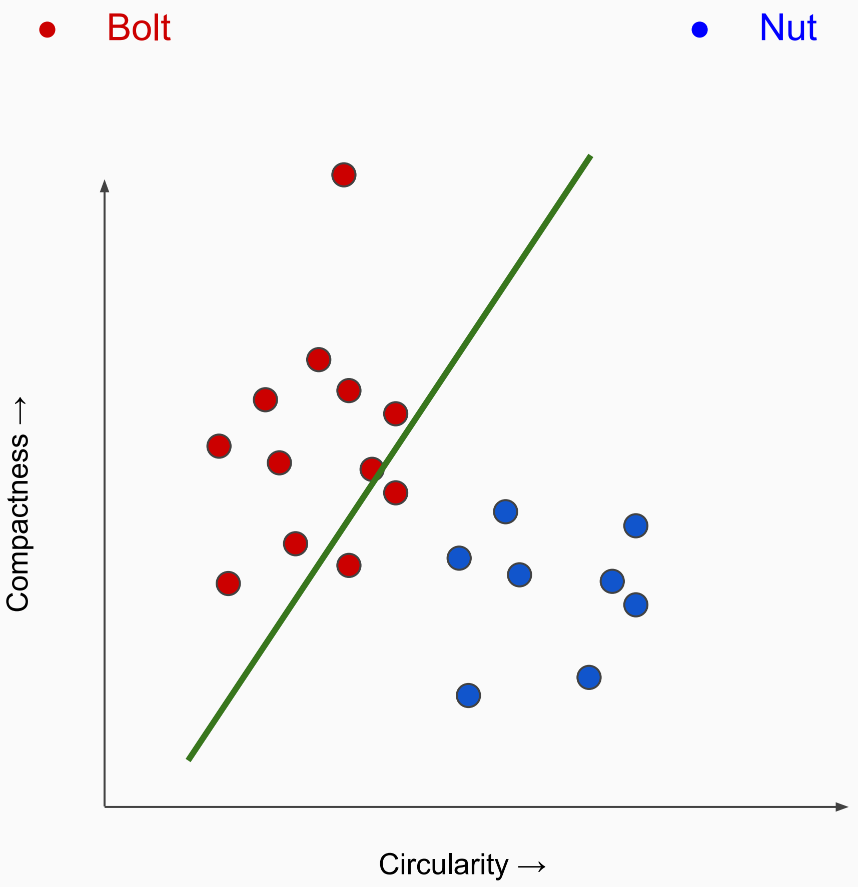
Regularization parameter which penalizes higher order θ

All (Wo)Men must Code

Basic ML Notebook

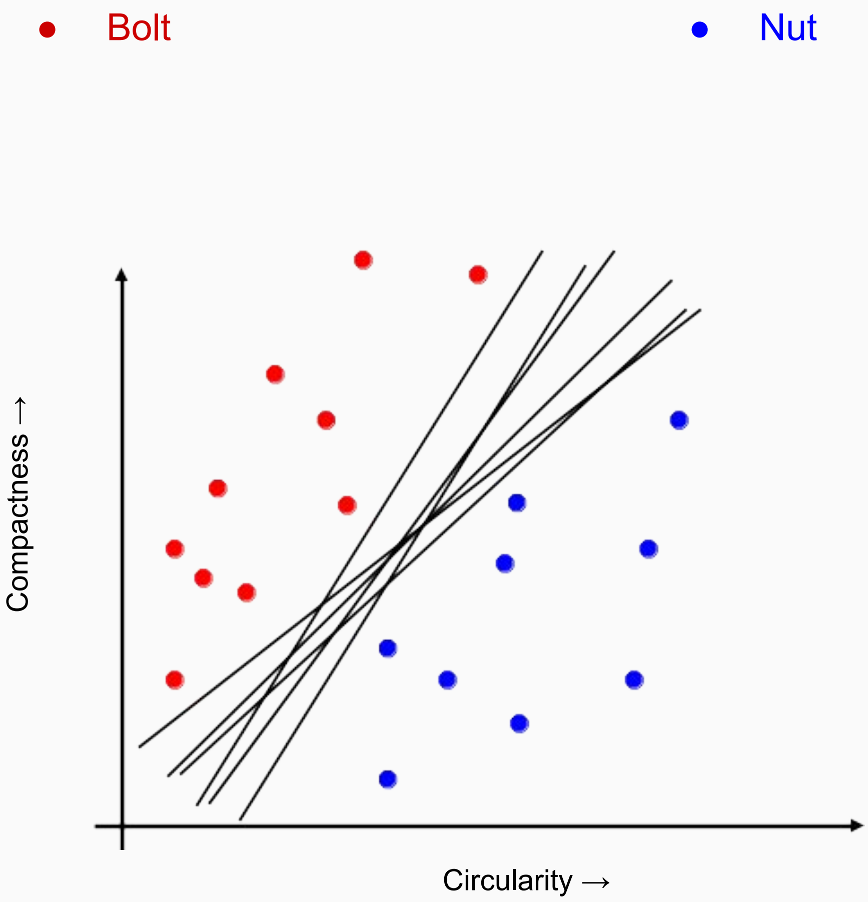
Problems

Why Logistic Regression does not work?



More intelligent classifier

Which of the linear separators is optimal?



Soliton
Vision for a Better World

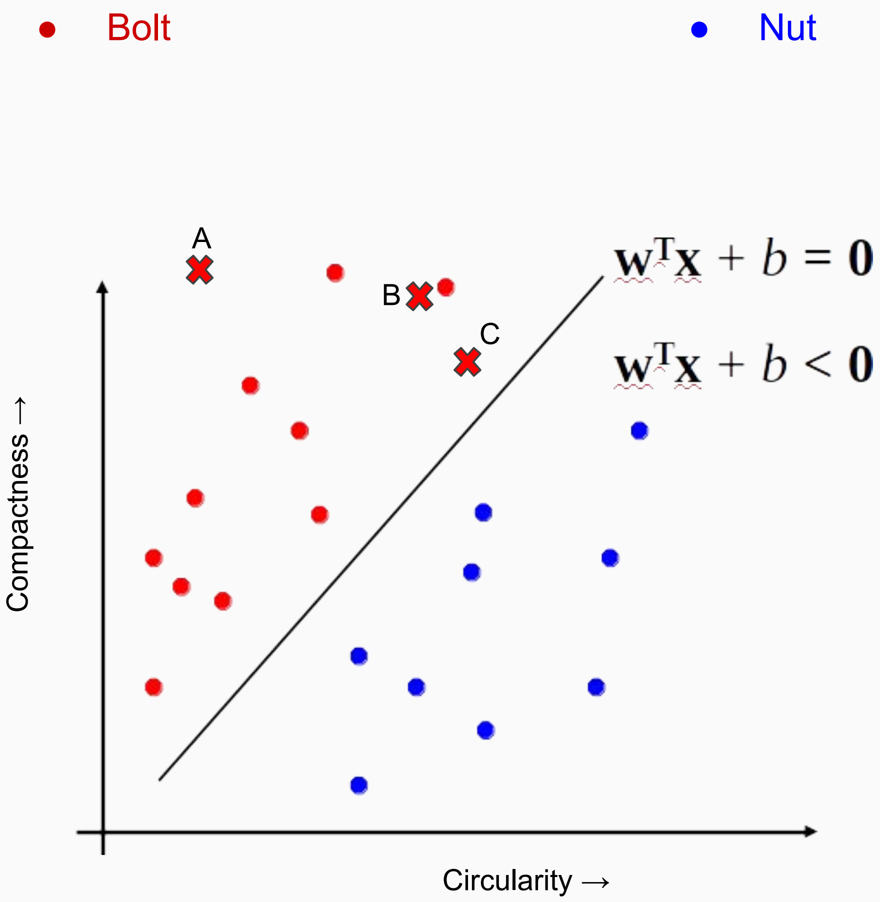
Margins

Let's take the Logistic regression example

Among A, B and C which data point would you confidently classify as Bolt?

If the point is far from the separating hyperplane, we may be significantly more confident in our predictions

Find a decision boundary that allows us to make all CORRECT and CONFIDENT predictions

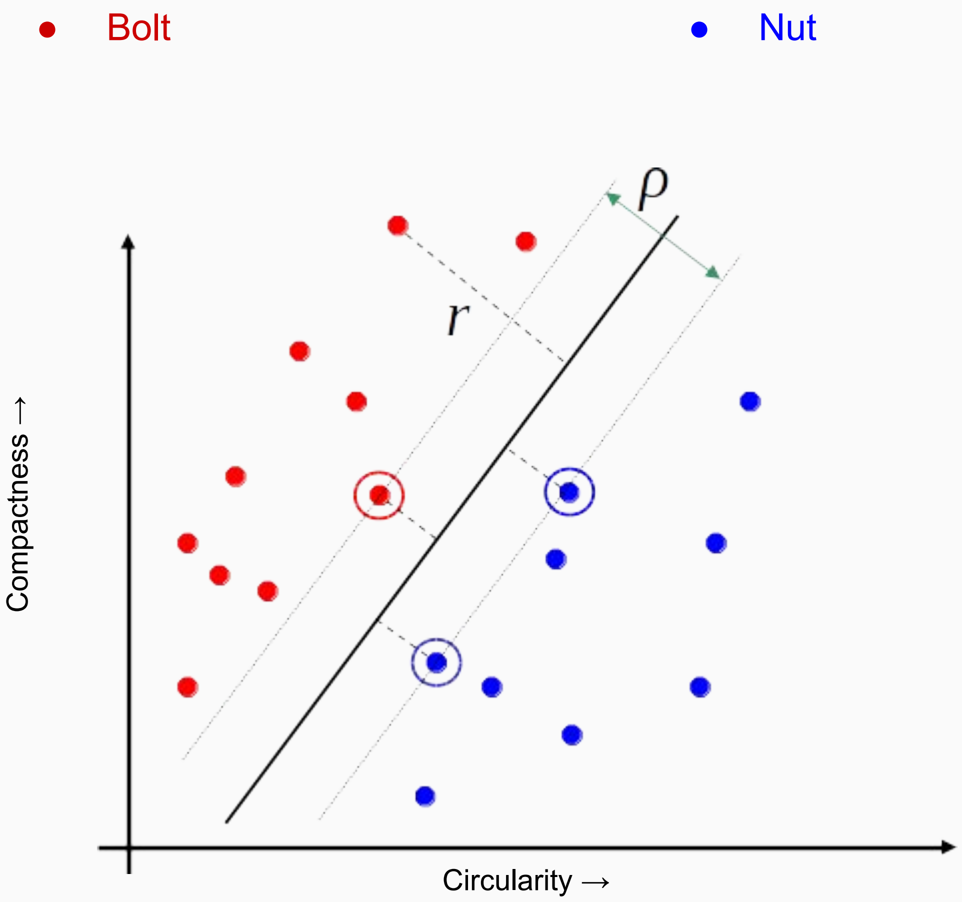


Soliton
Vision for a Better World

SVM

Optimal Margin Classifier

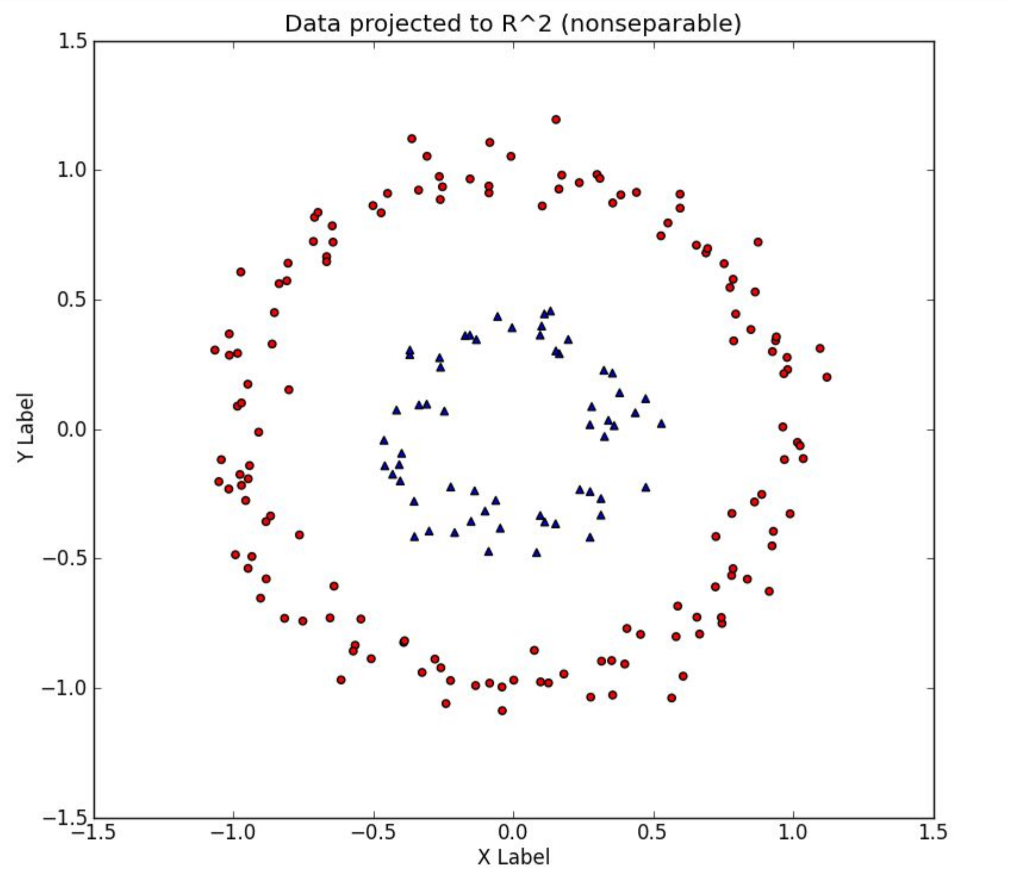
- Examples closest to the separator are support vectors.
- Margin **ρ** of the separator is the distance between support vectors
  $r = (W^TX + b) / ||W||$
- Functional Margin
  $\hat{\gamma}^{(i)} = y^{(i)} (w^T x + b)$



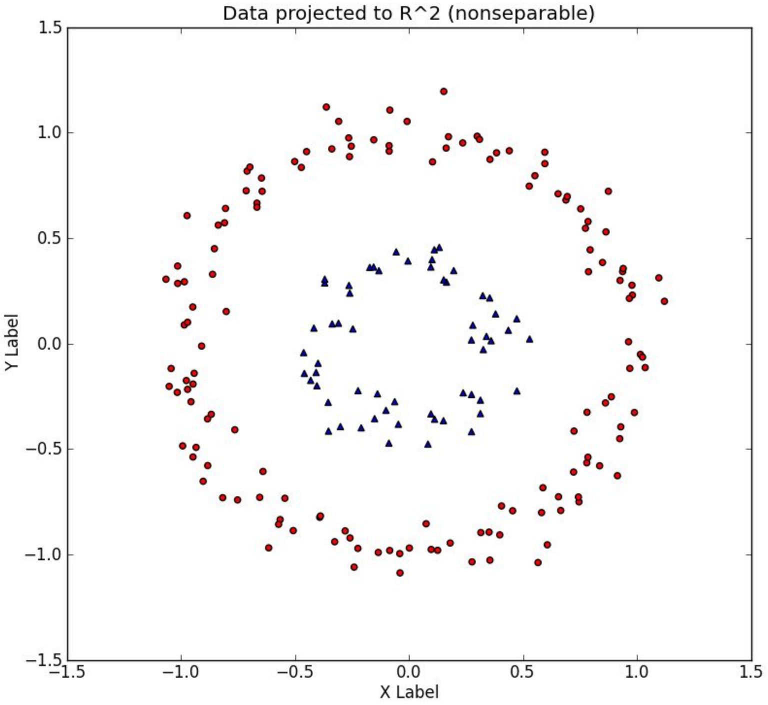
Nonseparable Data

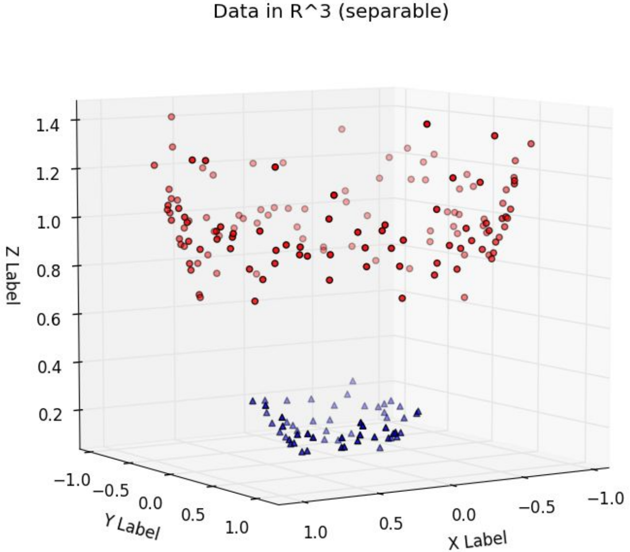
We learned that SVM is a Linear Classifier

Can SVM classify the given data?



Project data to Higher Dimension

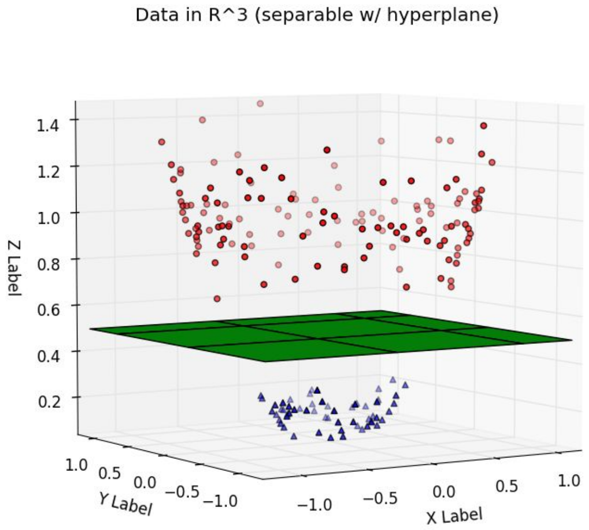


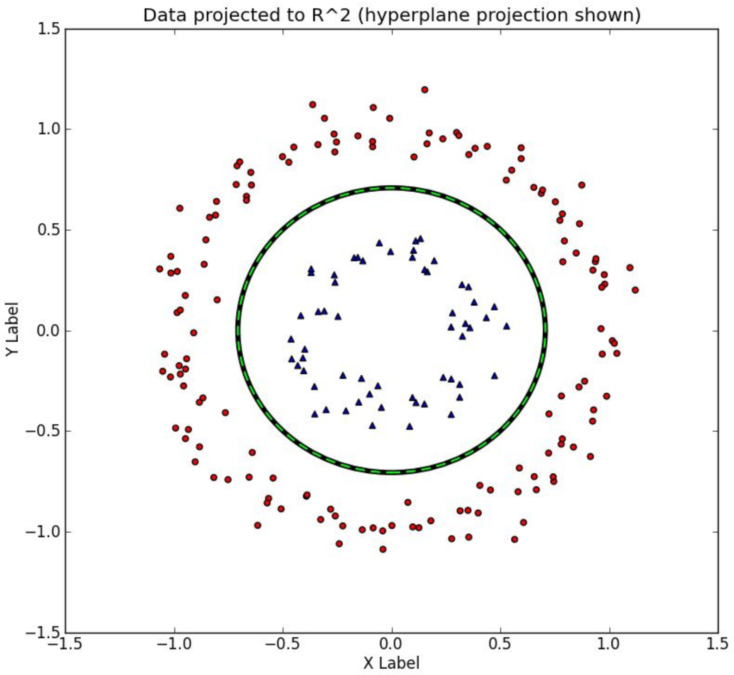


$$[x_1, x_2] = [x_1, x_2, x_1^2+x_2^2]$$

Soliton
Vision for a Better World

Hyperplane re-projection in 2D



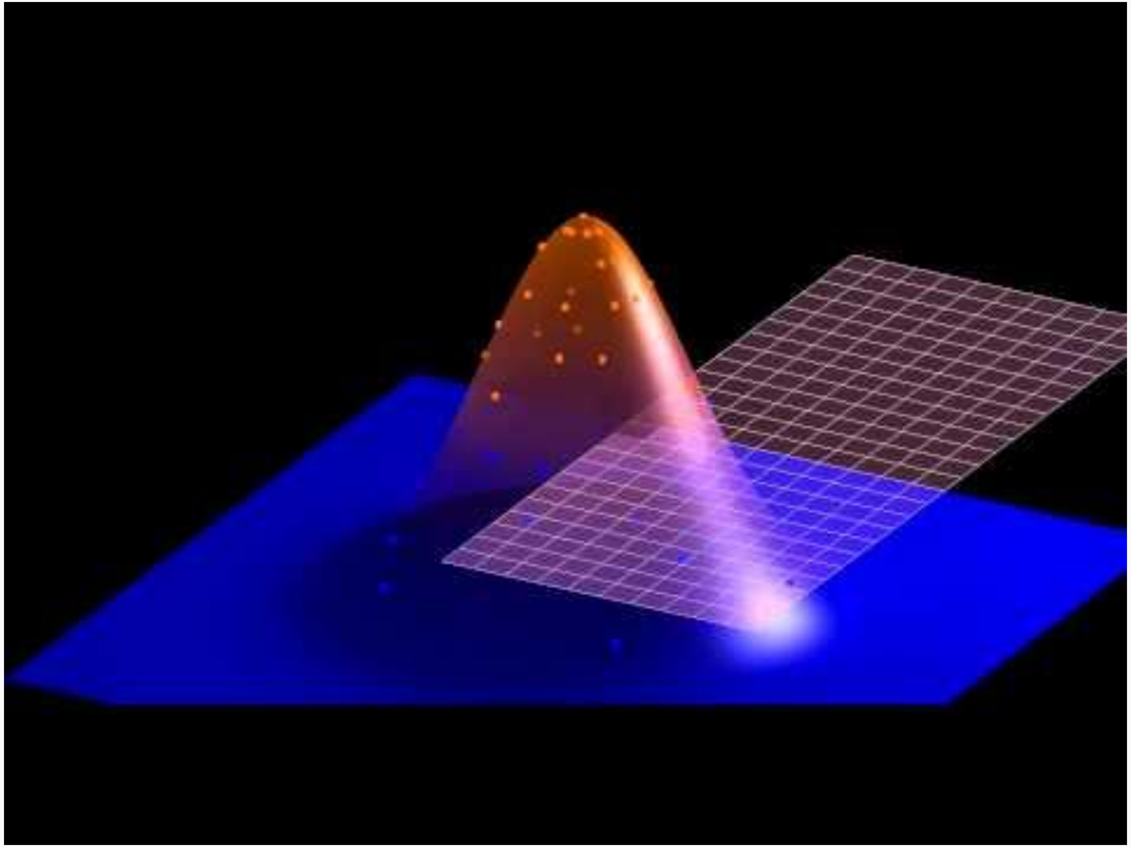


Soliton
Vision for a Better World

Demo

Just,

- Project the data into higher Dimension
- Find Hyperplane
- Re-project the Hyperplane back to original dimension



Soliton
Vision for a Better World

Kernel Trick

It is so simple. Is it True?

Higher Dimensional Projection is Expensive

- Impractical for Large Dimensions
- Huge memory and Computation are required
- Transformation from N dimension to M Dimension is **O(N²)** expensive

We only need Dot Products..! Not the High Dimension Data

$$K(x, z) = \varphi(x)^T \varphi(z)$$

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}.$$

$O(N^2)$

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_i z_i\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \end{aligned}$$

$O(N)$

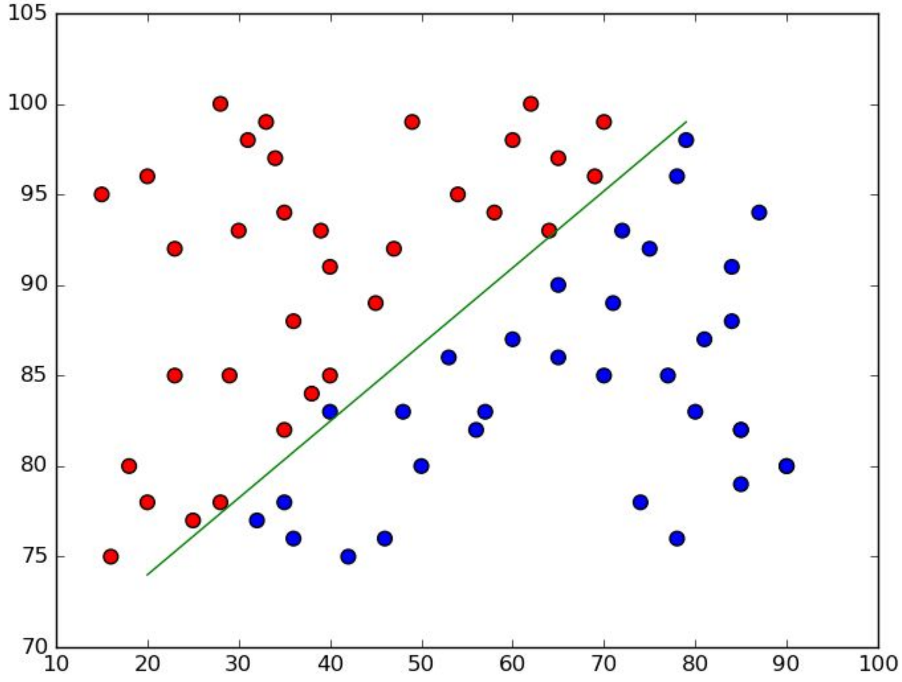
Computationally Faster and No extra memory needed

Soliton
Vision for a Better World

SVM

Find the Hyperplane which optimizes the Functional and Geometric margin iteratively

● Bolt ● Nut



Soliton
Vision for a Better World

Hands-on is Coming..!

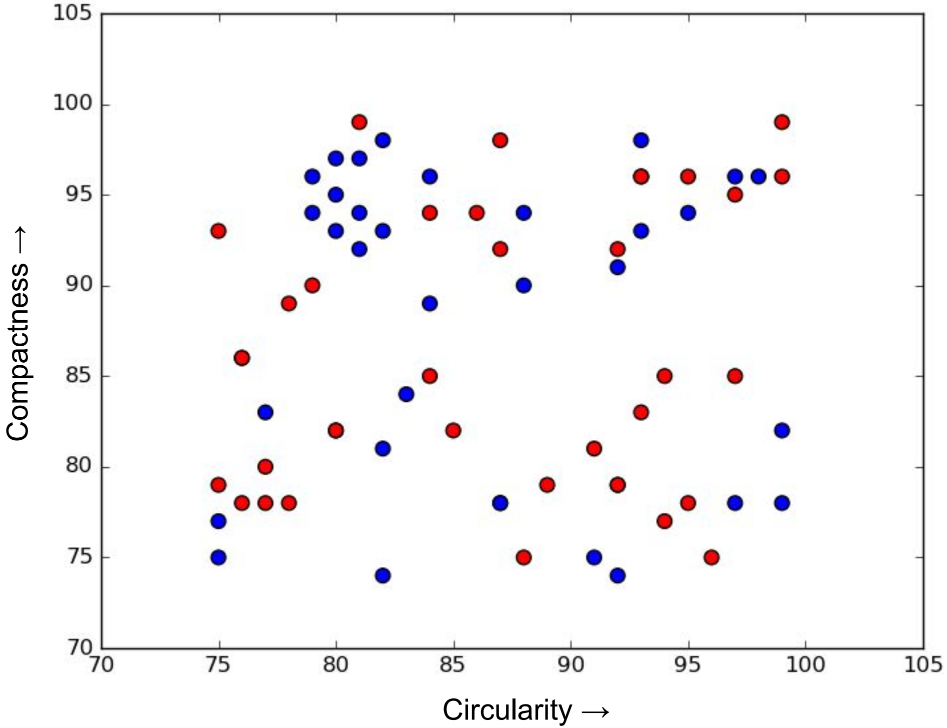
IPython Notebook

Real World Data

In the real world the data may not be Linearly separable

How do we classify the data now?

● Bolt ● Nut

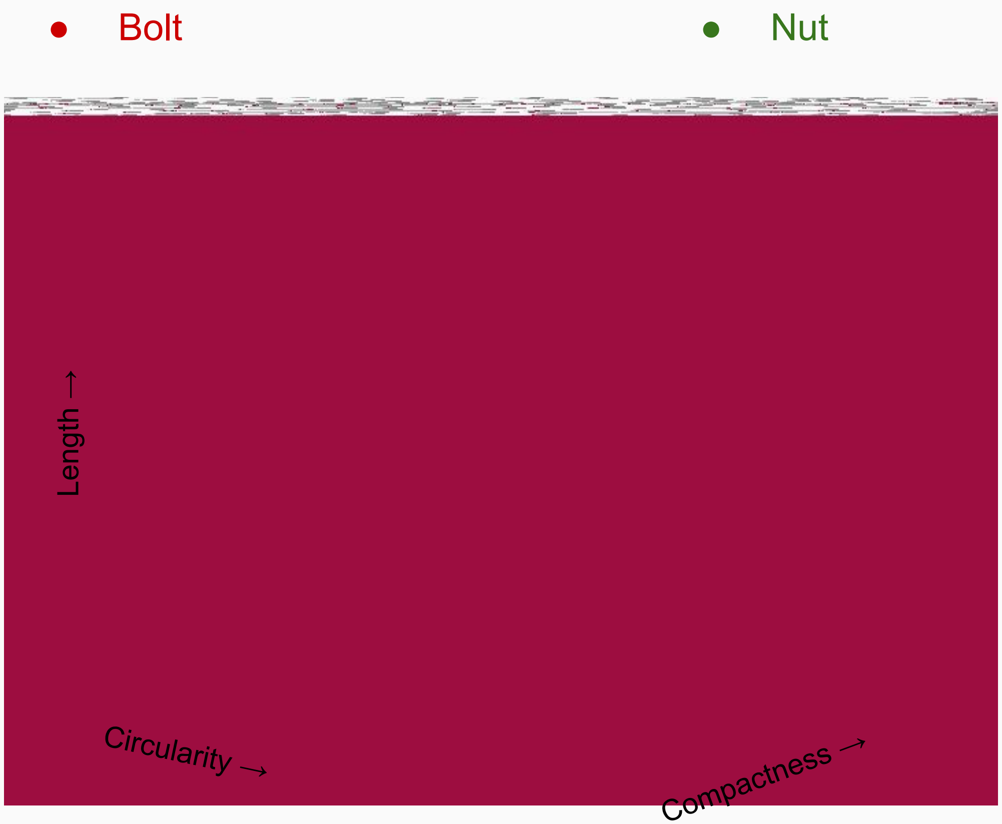


Soliton
Vision for a Better World

More Features

If I project the same data into 3D / 4D / 5D, etc, can I separate the data Linearly?

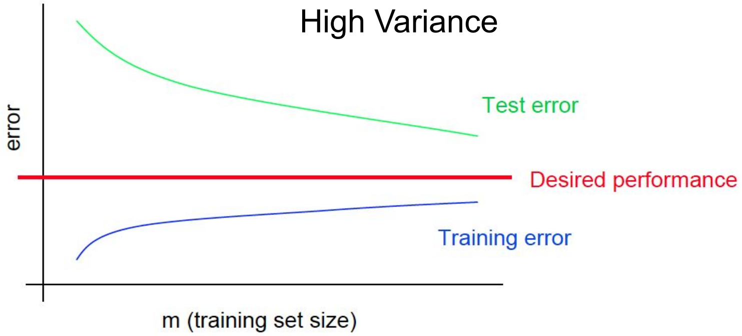
Introduce more features for more accurate classification

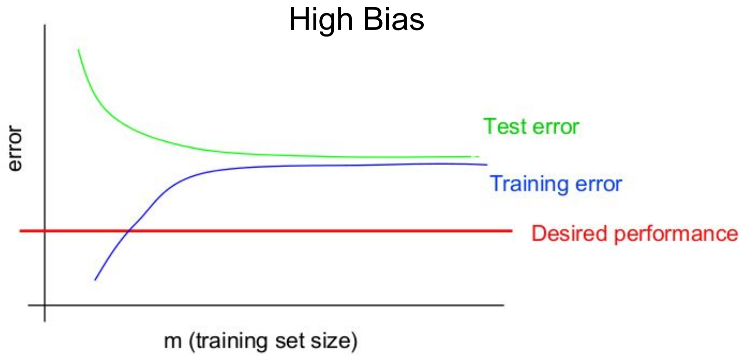


Soliton
Vision for a Better World

ML Advice - Diagnostics

Try getting more training examples	Fixes high Variance
Try a smaller set of features	Fixes high Variance
Try a larger set of features	Fixes high Bias
Try changing features (e.g, email header features)	Fixes high Bias
Run gradient descent for more iterations	Fixes optimization algorithm
Try Newton's method	Fixes optimization algorithm
Use a different value for λ	Fixes optimization objective
Try using an SVM	Fixes optimization objective





Credits: cs229, Andrew Y. Ng

ML Advice

Rule #1: Plot the Data

Questions to ask:

Is the Algorithm converging?

Are you optimizing the right function?

Is the value for λ is correct?

Is the value for C is correct?

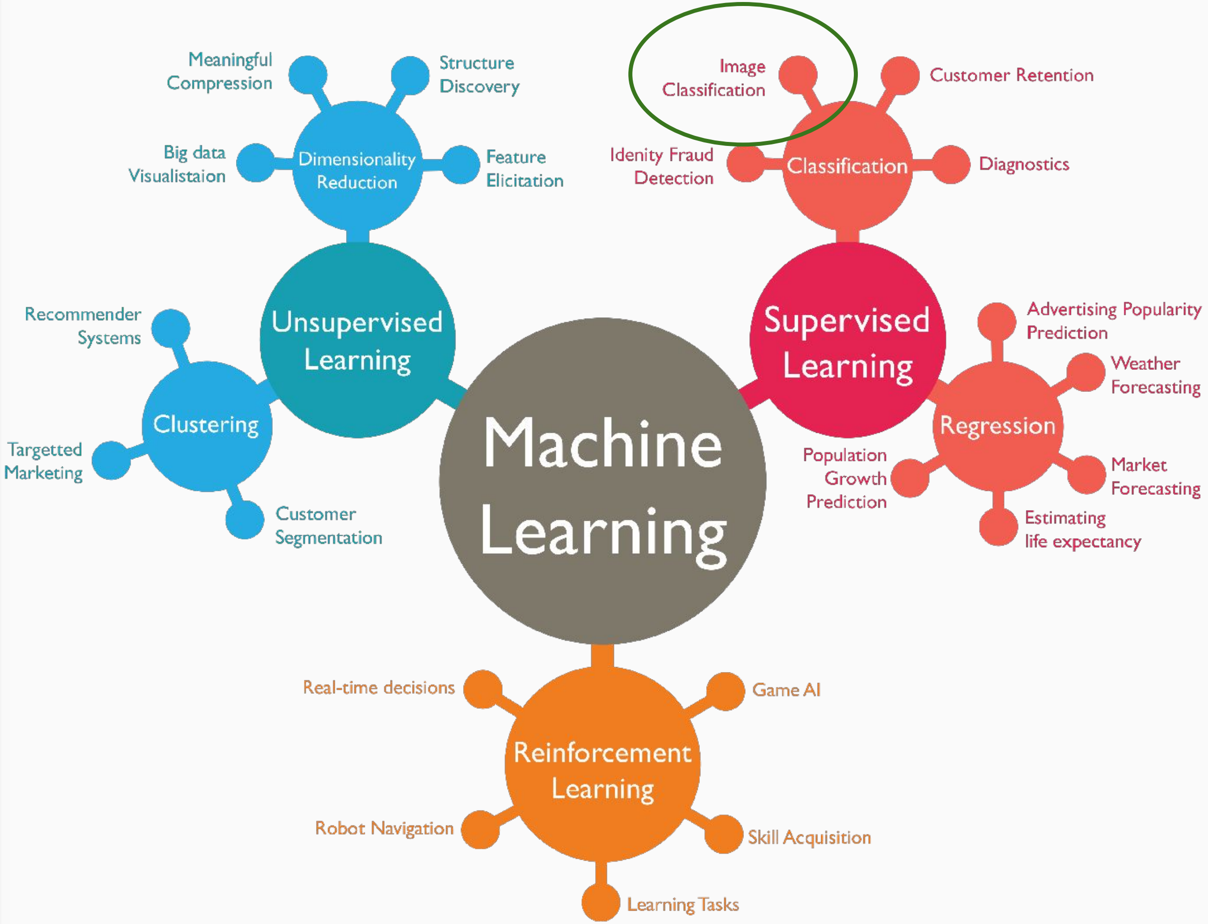
Are initial parameters correct?

Error Analysis	Ablative Analysis
• Helps to understand how much error is attributable to each component? • Helps to identify Poor components by which we can improve performance • List down accuracy **DROP** after introducing each component. • Plug in ground-truth for each component, and see how accuracy changes	• Helps to understand how each component in the system helps to achieve final better accuracy • Helps to identify the less contributing component so they can be removed • List down what is the accuracy **IMPROVEMENT** after each level starting from the basic model • Remove one component at a time and see how accuracy drops

Types of ML

Classification

- **Logistic Regression**
- **Decision Tree**
- AdaBoost
- Naive Bayes Classification
- **SVM (Support Vector Machine)**



Soliton
Vision for a Better World

Types of Learning

Supervised: Learning by Labelled Ex

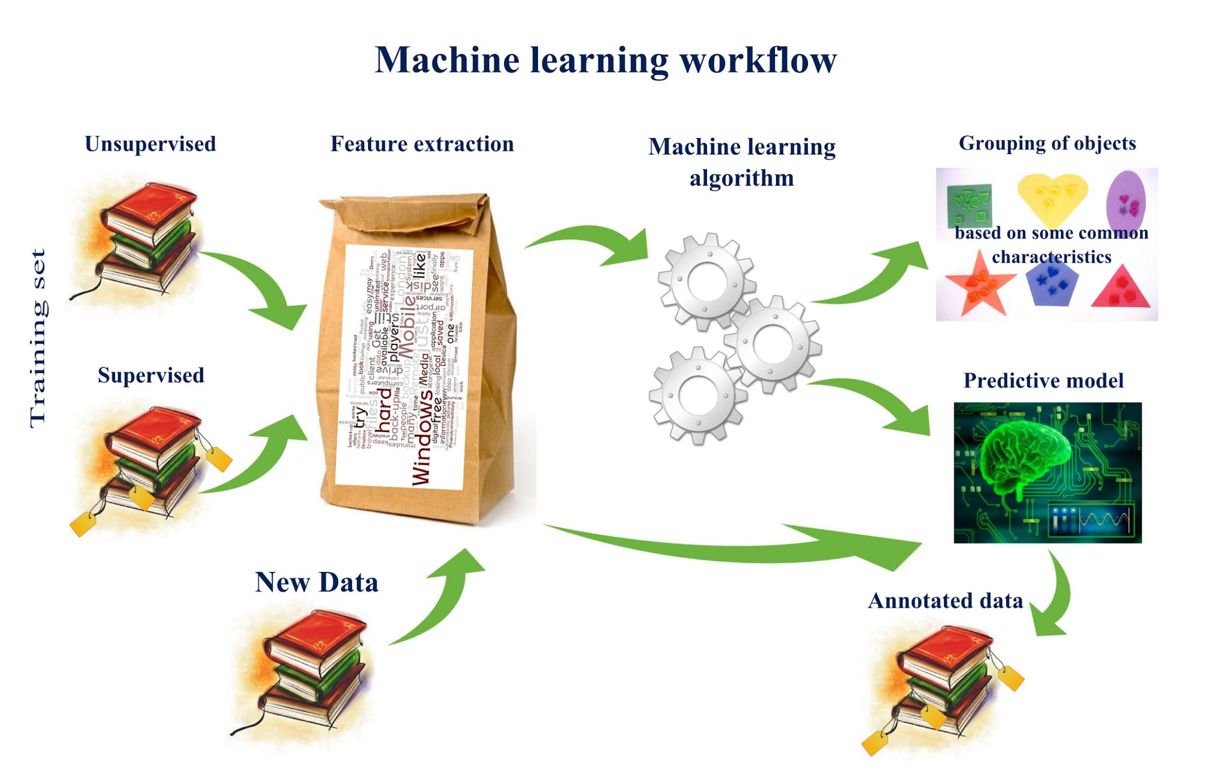
- Eg. Face Recognition
- Amazingly effective if you have labelled examples

Unsupervised: Discovering Patterns

- Eg. Google News - Data Clustering
- Useful if you lack labelled data

Reinforcement: Feedback right/wrong

- Eg. Playing chess by winning or losing
- Works well in some domains, becoming more important

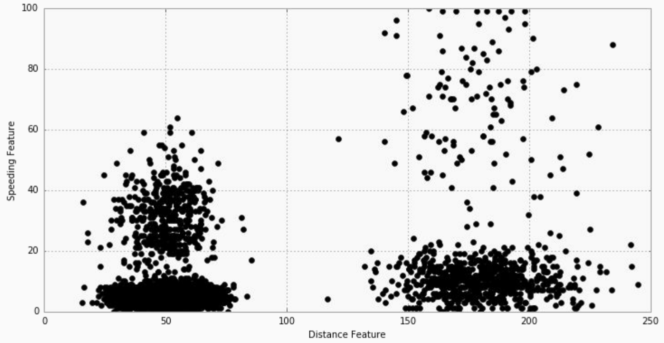


Soliton
Vision for a Better World

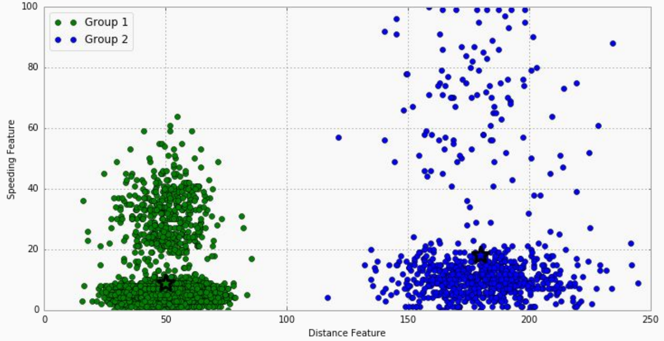
Unsupervised Learning

K-means Clustering

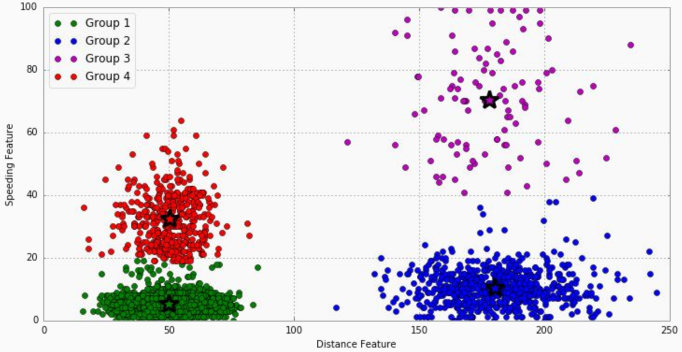
Input Data



K = 2



K = 4



Soliton
Vision for a Better World

https://www.datascience.com/blog/k-means-clustering

Unsupervised Learning

- Clustering

Soliton
Vision for a Better World

https://www.datascience.com/blog/k-means-clustering

Reinforcement Learning

How does a Human learn?

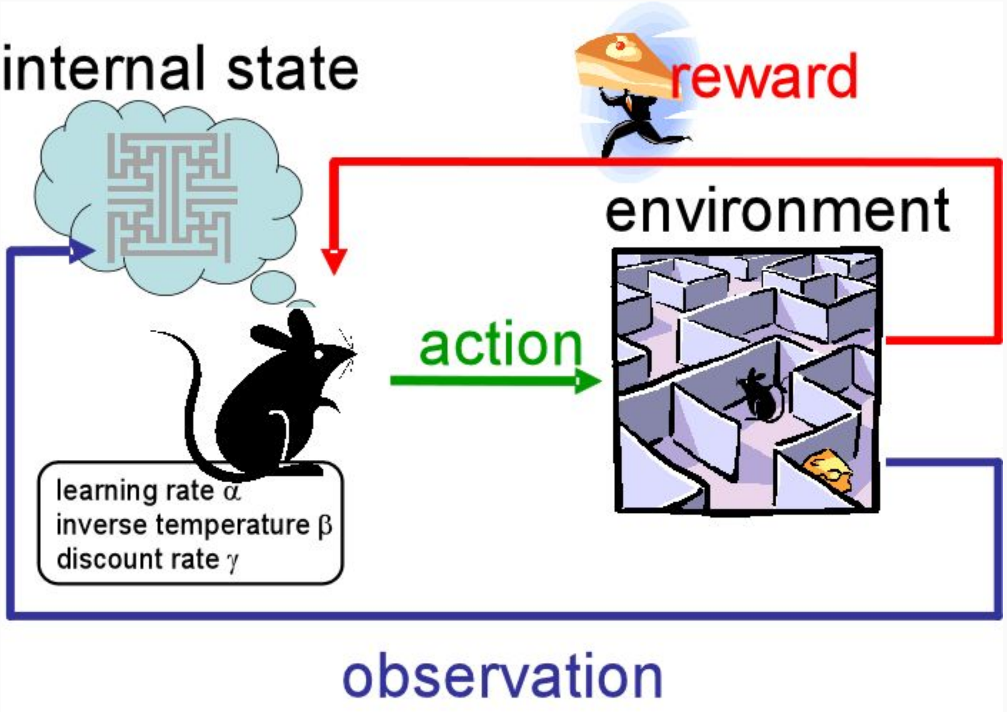
- Tilt the bike in 0 deg: Goes out of Fence
- Tilt it in 90 deg: Loses balance and falls down
- Getting these feedback / reward, learn how to ride i.e, learn the angle

Soliton
Vision for a Better World

https://www.datascience.com/blog/k-means-clustering

Reinforcement Learning

-



Soliton
Vision for a Better World

https://www.datascience.com/blog/k-means-clustering

Reinforcement Learning

Soliton
Vision for a Better World

https://www.datascience.com/blog/k-means-clustering

MOOC for ML

cs229 is good place to start

Do a lot of assignments

Work on pet projects

Contribute to ML Open source libraries

Courses

- ML: cs229 by Andrew Y. Ng
- RL: David Silver
-

Books:

Blogs & Github:

- Scikit-Learn examples
- ML Playground

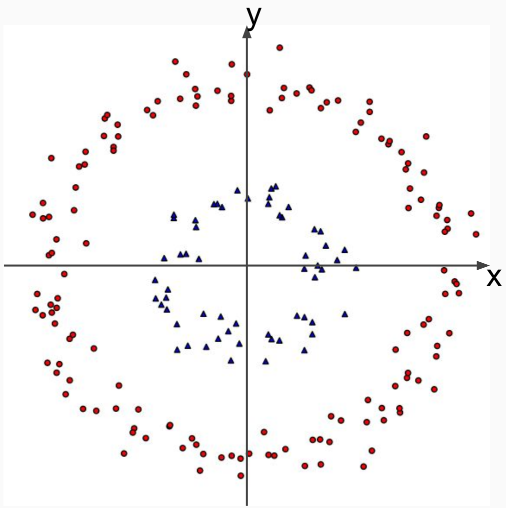
Soliton
Vision for a Better World

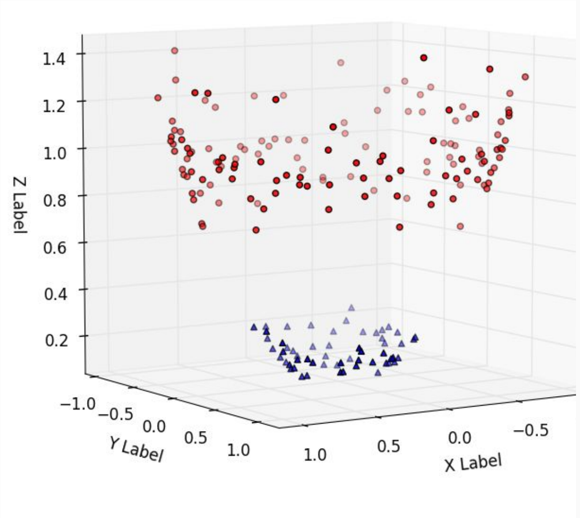
How to Solve this?

We MAY be able to solve this by introducing one more feature which MAY separate them linearly.

Or I will let SVM project it to higher dimension and find hyperplane and reproject it back

Do you see any pattern? Any mathematical solution?





Clue..!

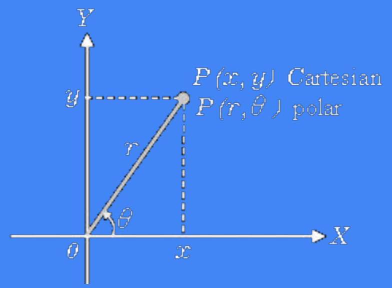
Think in MATHEMATICS..!

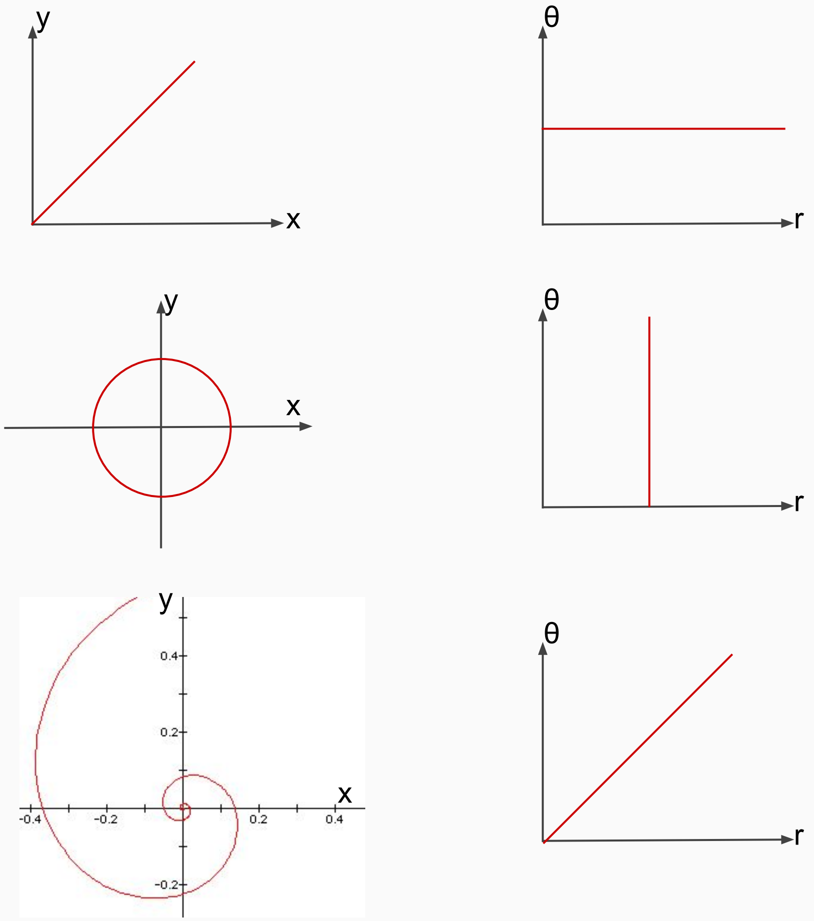
Soliton
Vision for a Better World

Cartesian -> Polar

$r = \sqrt{\,(x2 + y2)}$

$\theta = \text{tan-1}\,(y / x)$

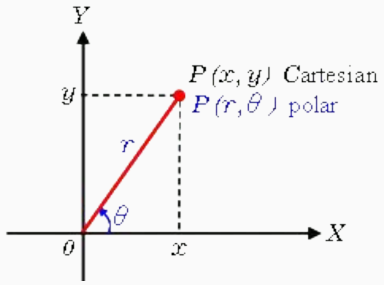


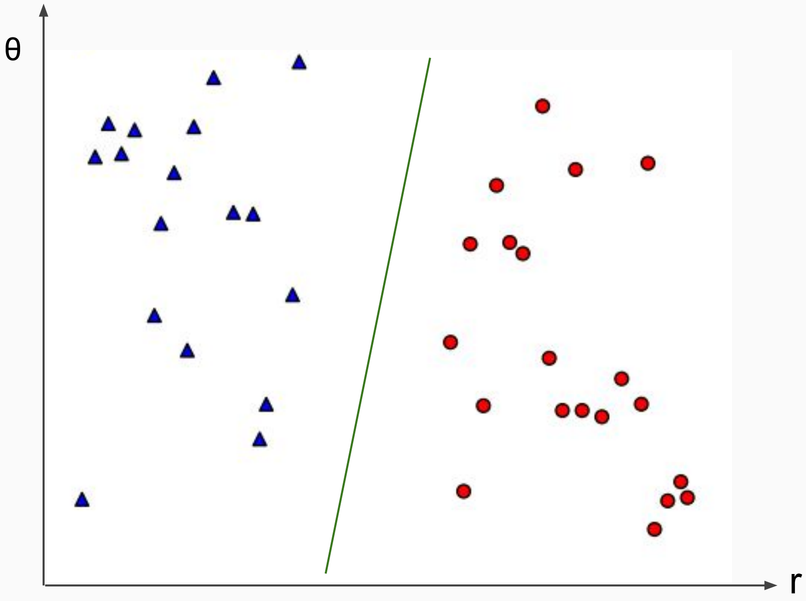


Cartesian -> Polar

r = √ (x2 + y2)

θ = tan-1 (y / x)





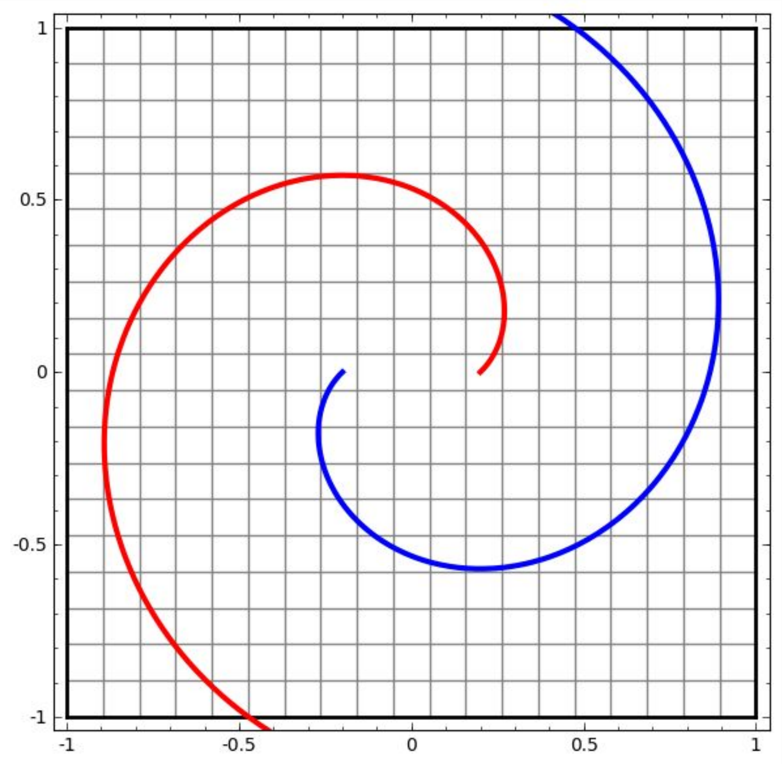
What is Next?

Let the Algorithm Learn these

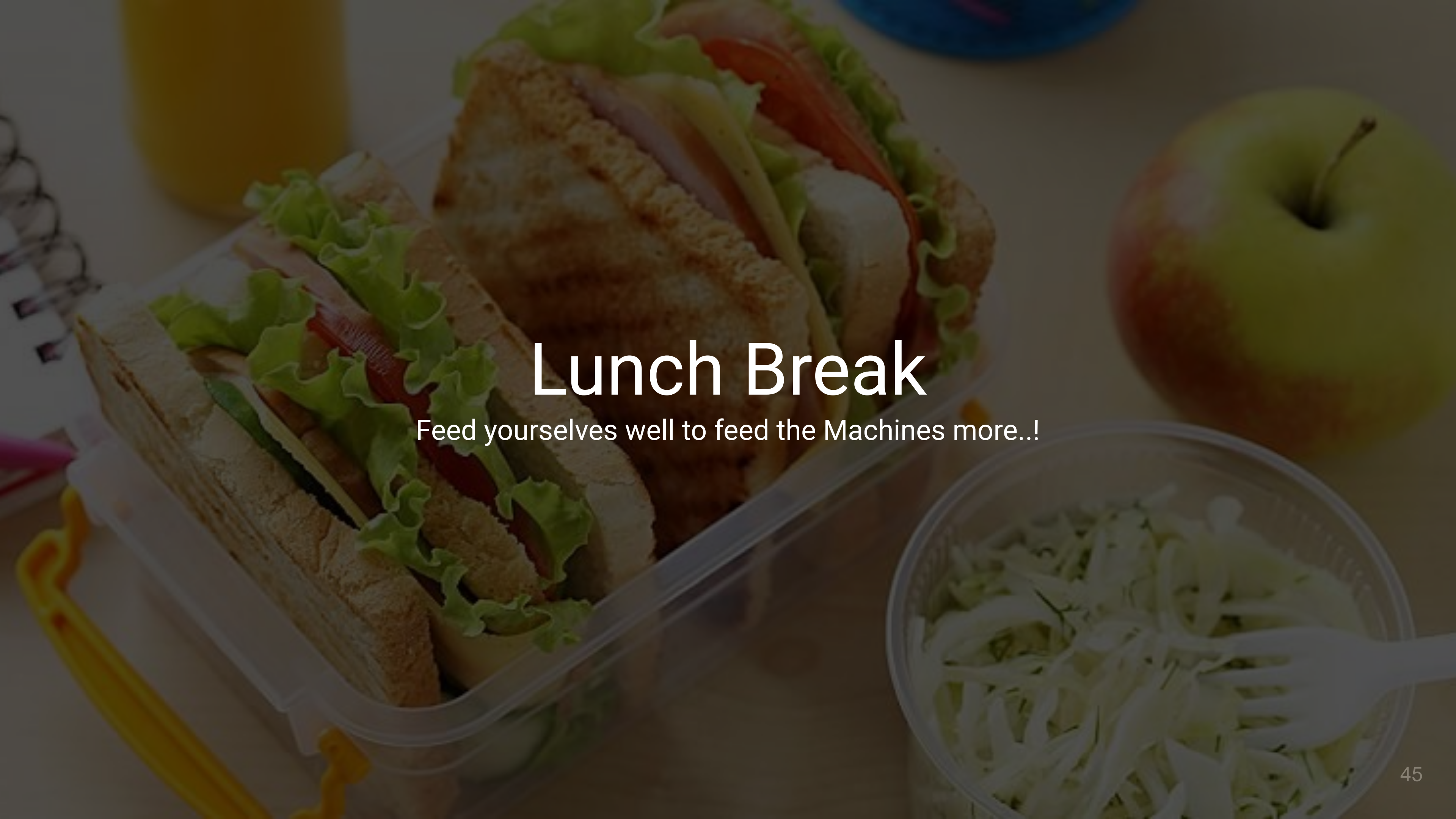
- Functions
- Features

ON ITS OWN…!

Deep Learning



Soliton
Vision for a Better World



Lunch Break

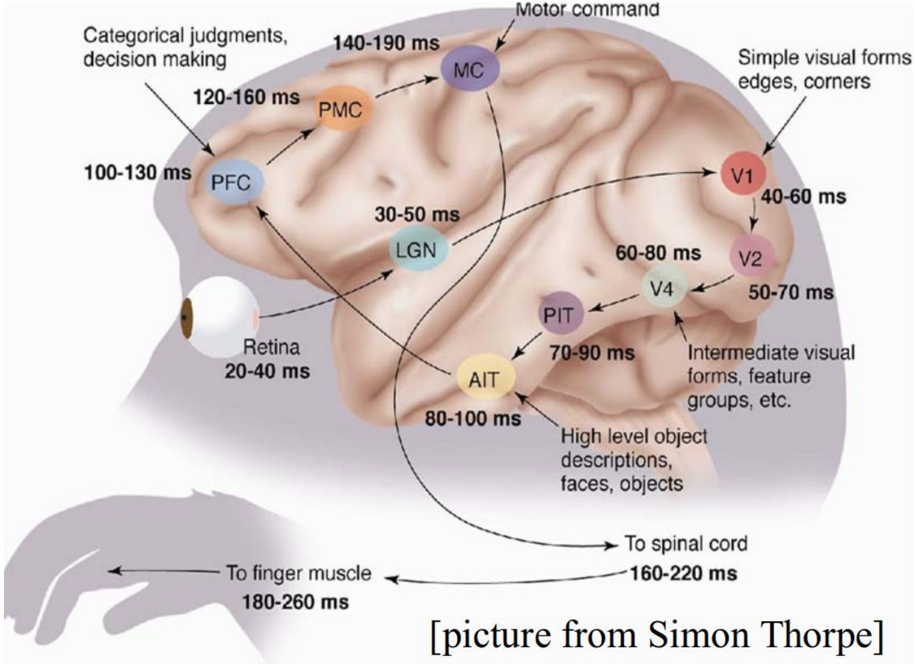
Feed yourselves well to feed the Machines more..!

Day 2 - AN 1 (DL)

What is the limitation of simple Image Processing and why we need intelligent systems?

What we know from Neuroscience

- Layer wise processing
- Hierarchical
- Simple Cells -> Complex Cells
- Closest Analog: IT
- Fovea
- Fusion of other sensory i/p
- Generative Model of world
- Even V1 gets feedback from
- Feedforward IT ~ CNN
- (role of feedback)



[picture from Simon Thorpe]

http://timdettmers.com/2015/07/27/brain-vs-deep-learning-singularity/

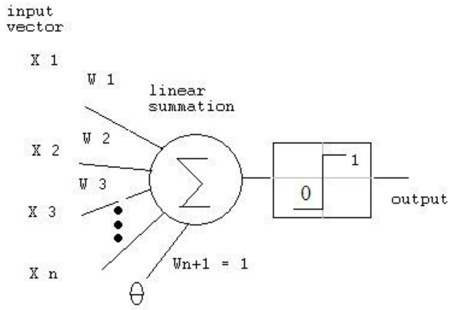
Limitation of Traditional ML

- Need to hand engineer features which can take a lot of time
- The limitations in its ability to represent complex features (Requires a lot of diligence and intelligence)
- Models developed for one problem cannot be easily be utilised for a similar problem

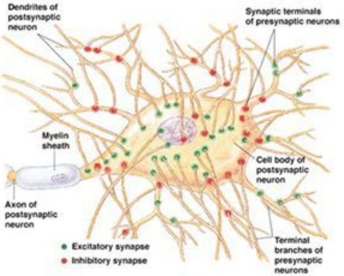
Simplest Models

Perceptron vs. the point neuron

- Incoming signals from synapses are summed up at the soma
- $\sum$, the biological “inner product”
- On crossing a threshold, the cell “fires” generating an action potential in the axon hillock



The McCulloch and Pitt’s neuron

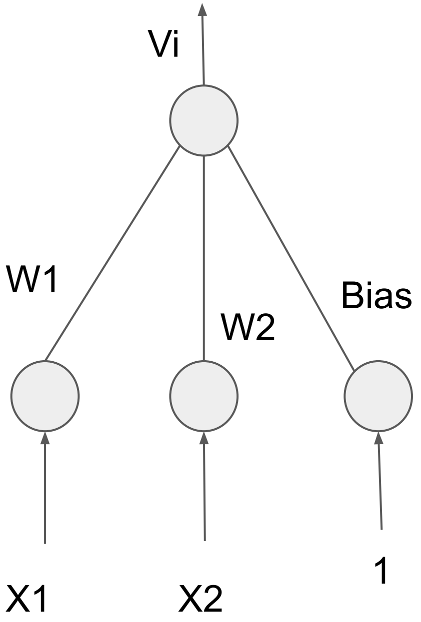


Synaptic inputs: Artist’s conception

Simplest Models

- **Perceptron training**

Include bias term as the third weight(w3) with its input always set to 1
Step 1: Initialization: $w_i = 0$, i = 1 to n
For each of the training sample do steps 2 -4
Step 2: Compute output by weighted linear combination of inputs
(Vi = w1 *x1 + w2 * x2 + 1 * bias)
Step 3: Find the error (Error = Anticipated output - predicted output)
Step 4: Update each weight based on the following
ΔWi = Error * э * Xi
Wi = Wi + ΔWi
Where э is the learning rate and its range is
0 <= э < 1
Step 5: Repeat the procedure until no error results



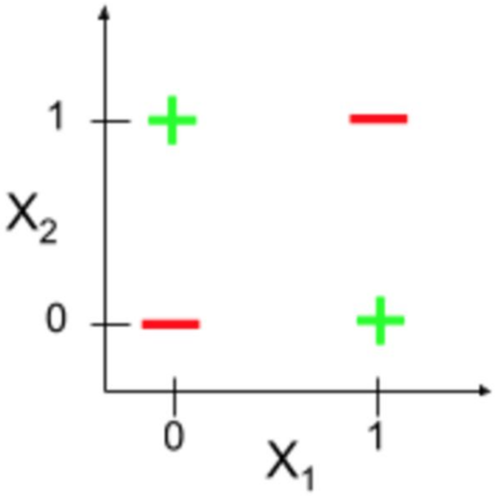
Enough Talk..! Let’s Code!!

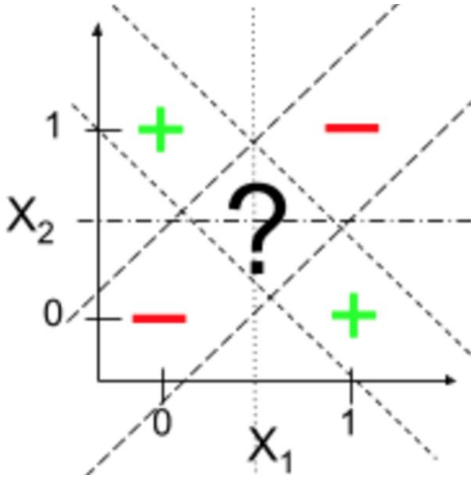
[IPython Notebook]

Perceptron Limitations - Linear separability

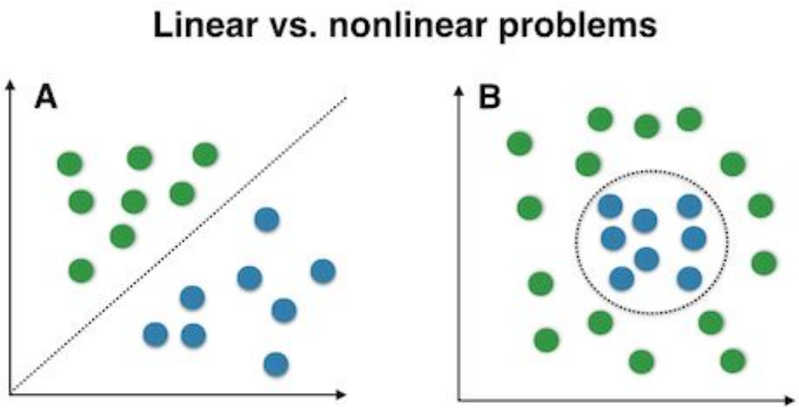
- A simple perceptron cannot learn a classifier for a XOR gate
- How to draw a decision boundary in case of a XOR gate?

x_1	x_2	x_1 **XOR** x_2
0	0	0
0	1	1
1	0	1
1	1	0





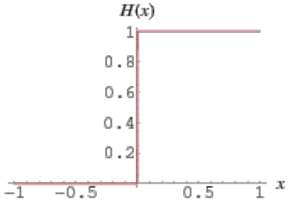
Problem of Linear separability



- Can this kind of perceptron provide solutions to all kinds of data patterns we might encounter in practice? Let's find out
- This is because we don't have non-linear elements in our network. Hence, this kind of network can only learn linear functions of inputs.
- How can we improve the network to learn non-linear functions?
- Key observation - Cannot directly classify data. Convert the data to a new feature space to classify
- Nonlinearities should be included in the network

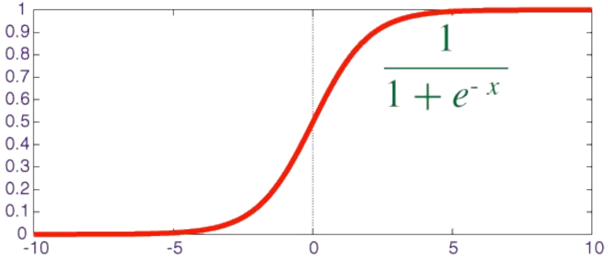
Sigmoid activation function

- We can include non-linear functions in our to improve the representational power of the network.
- We know how a initial models of how neuron fires.



- Can we represent this kind of activation as continuous and smooth function

- The sigmoid function is shown as follows

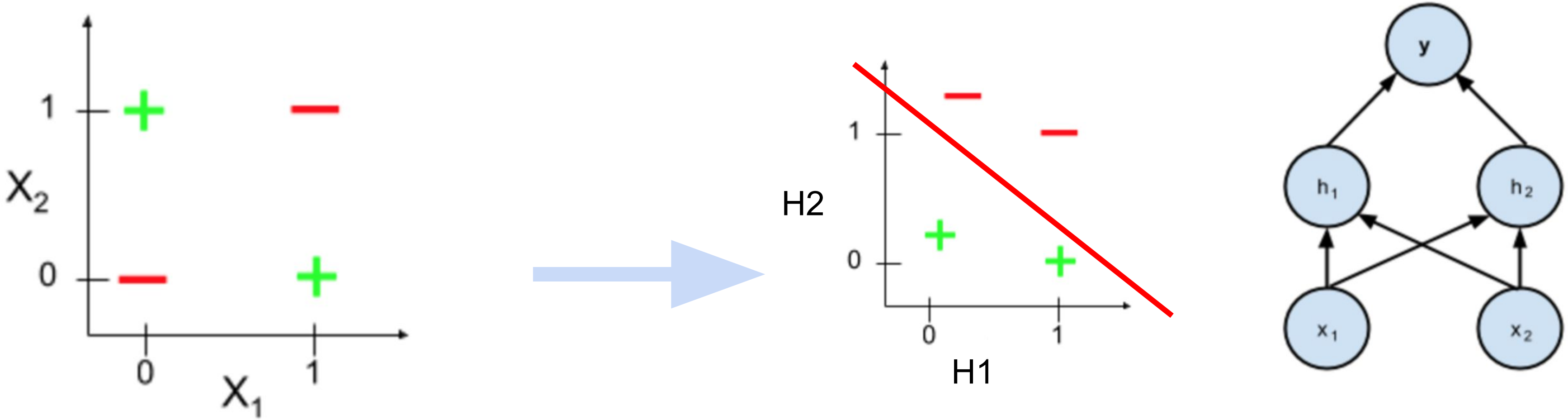


- Here x is weighted combination of the neuron inputs.
- The neuron fires when x >> 0 and does not fire when x << 0. The neuron lies in a transition state when when x ≈ 0.
- The function is smooth and differentiable

Case of two layers with non-linearities - MLP

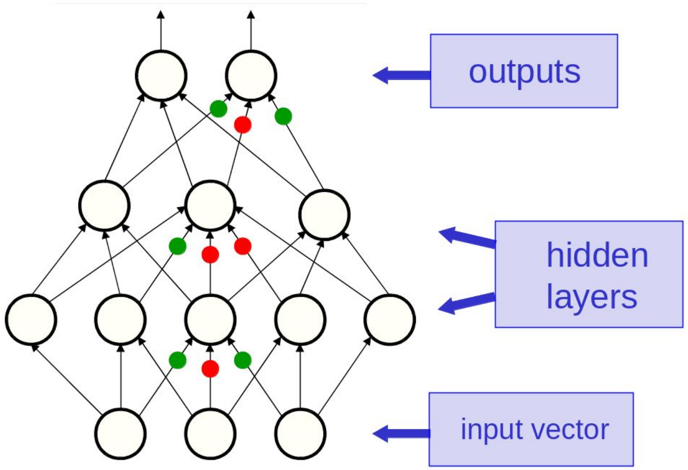
- The perceptron revenge

- Convert the inputs into a new feature space where the data points are linearly separable
- This necessitates the need at least two neurons with non-linearities. This kind of architecture is called Multilayer Perceptron (MLP)
- The first layer is called input layer, the layer at the last is called output layer. The layer / layers in between are called hidden layers



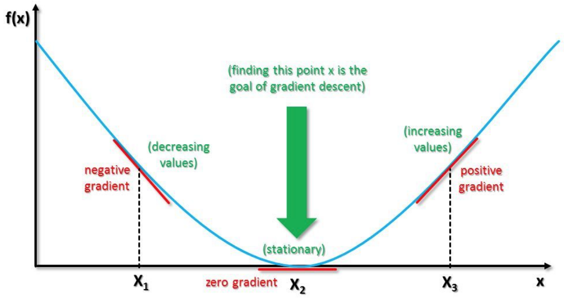
MLP

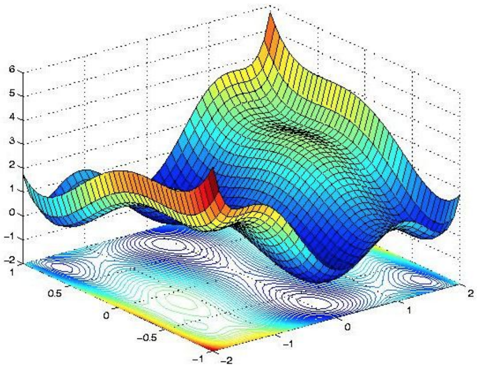
- A simple Multilayer network will consist of an input layer, output layer and one or more hidden layer in between
- It is not necessary that each hidden layer should contain same number of neuron. Each hidden layer usually contains a non-linear activation function



But what about training?

- In a single layer perceptron, there was direct interaction between input and output, hence we were able to update their weights directly based on output and inputs.
- Training becomes a little harder in MLPs, since we there are multiple layers of weight.
- We will measure the error committed by the network through an objective function (Just measure the squared difference between the anticipated output and predicted output - MSE)
- We will then try to find the minima of the objective function through gradient descent
- But how do we update weights of the network based on direction to move in gradient descent?





MLP - Backpropagation

- Backpropagation provides a way to compute the gradient of the error function with respect to each of the weights in the network.
- This provides a way to update update the weights of the network based on the error function.

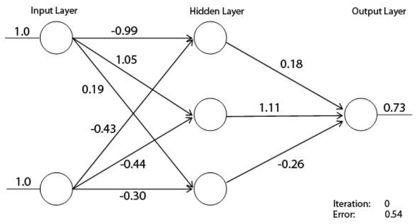
Terminologies:-
C - cost function
a_j^L- j^{th} neuro
δ_j^L - error of the j^{th} neuron in the L^{th} layer
b_j^l - bias of the j^{th} neuron in the L^{th} layer
w_{jk}^l- weight connecting the j^{th} neuron in the L^{th} layer to k^{th} neuron in the $L-1^{th}$ layer

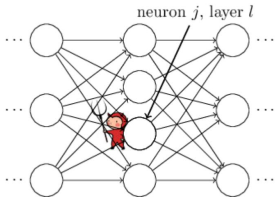
$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L).$$

$$\delta^l = ((w^{l+1})^T\delta^{l+1}) \odot \sigma'(z^l),$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l.$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l.$$





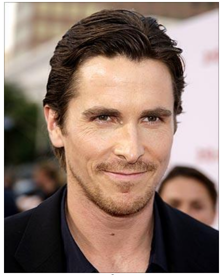
Representational power of MLP

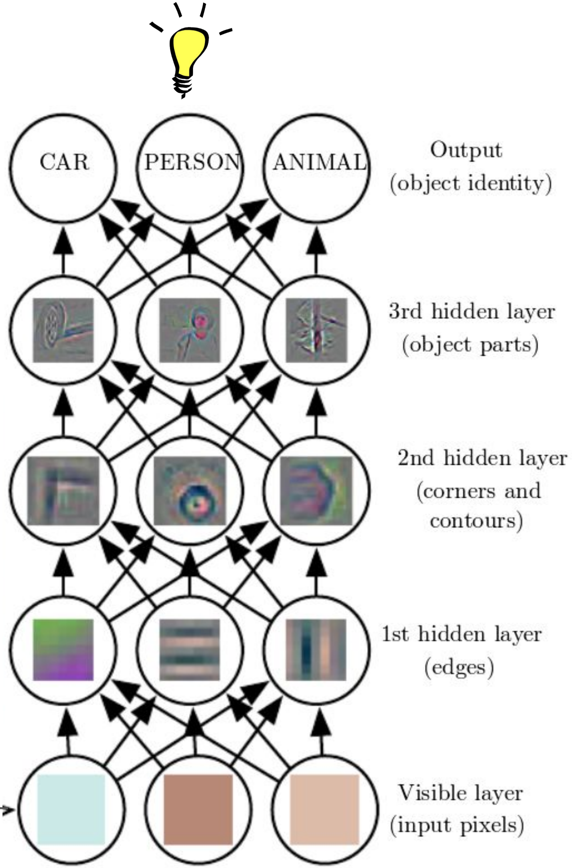
- Can this kind of MLP learn any functions?
- It turns out that this kind of neural network with one hidden layer is a universal approximator i.e., these neural networks can model any continuous function.
- Then why are many hidden layers required?
- It is practically difficult to learn the exact values of the parameters in such networks. Hence multiple layers make it practically possible to exploit the representational power of a neural network.

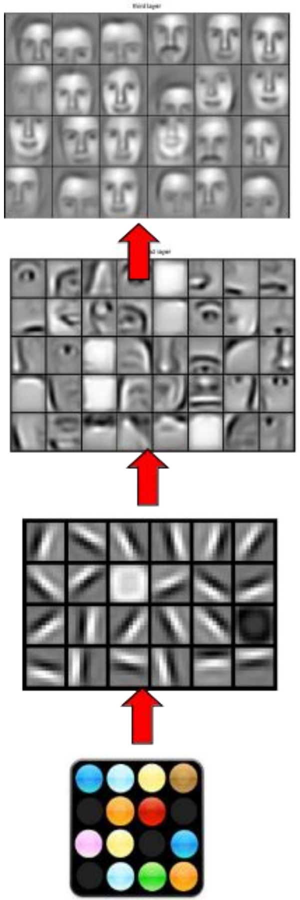
MLP - Limitations

- Very expensive training process (Too many parameters to learn)
- Not scalable to a larger architecture. The number of neurons increases rapidly with the number of neurons in the network.
- Does not converge

What do Neural Networks Learn?

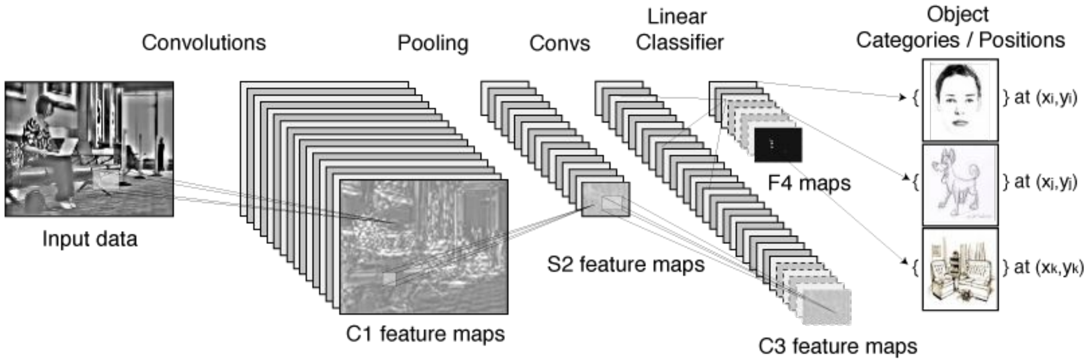






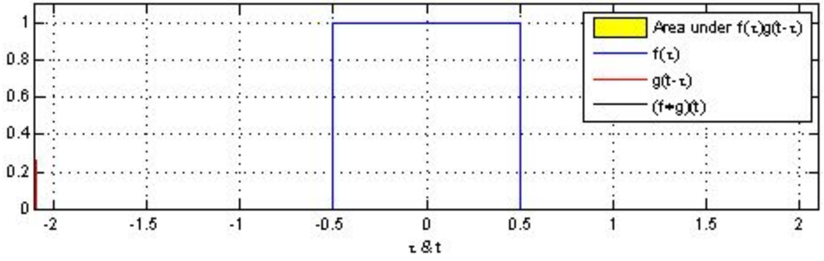
CNN

- Neurons are arranged in a 3D layer, unlike a MLP, where it is arranged in a 2D layer
- Each neuron views only a specific portion of the input and shares its parameters with many neurons in the same layer
- Encodes properties that are more desirable for images.
- Convolutional neural networks calculate the output from the input by repeated applying convolution operation.

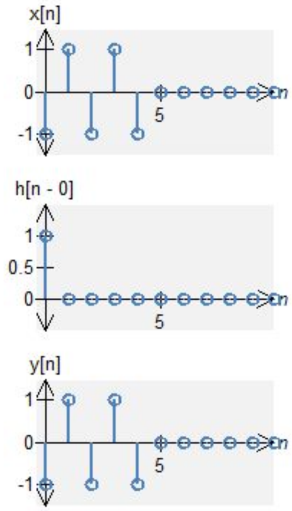


Convolution - signals motivation

Continuous case



Discrete case

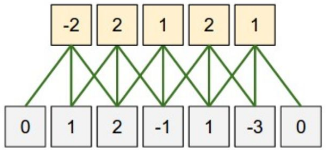


convolution is a mathematical operation on two functions to produce a third function giving the summation of the pointwise multiplication of the two functions as one of the functions is translated throughout

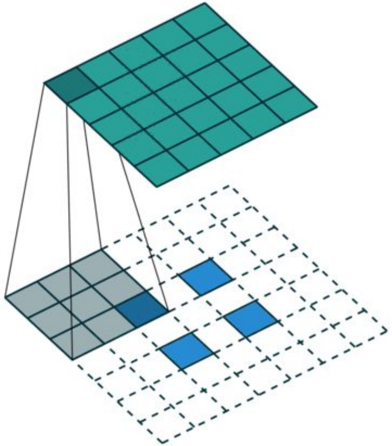
$$(f*g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$
$$= \sum_{m=-\infty}^{\infty} f[n-m]g[m].$$

Convolution

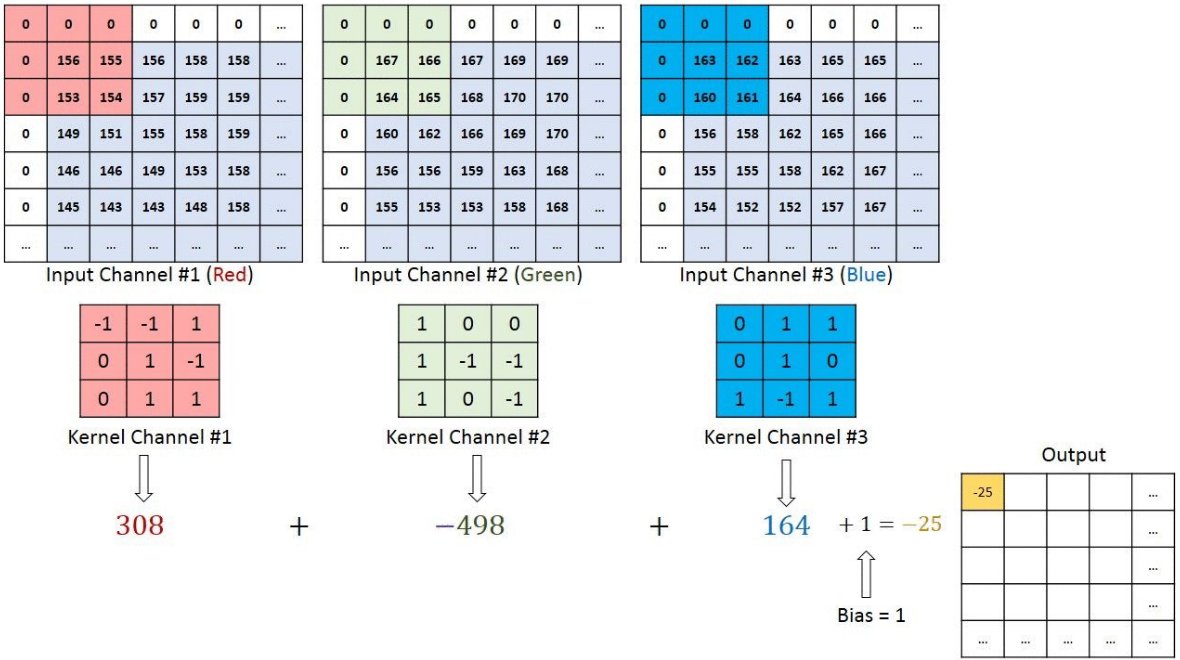
1D



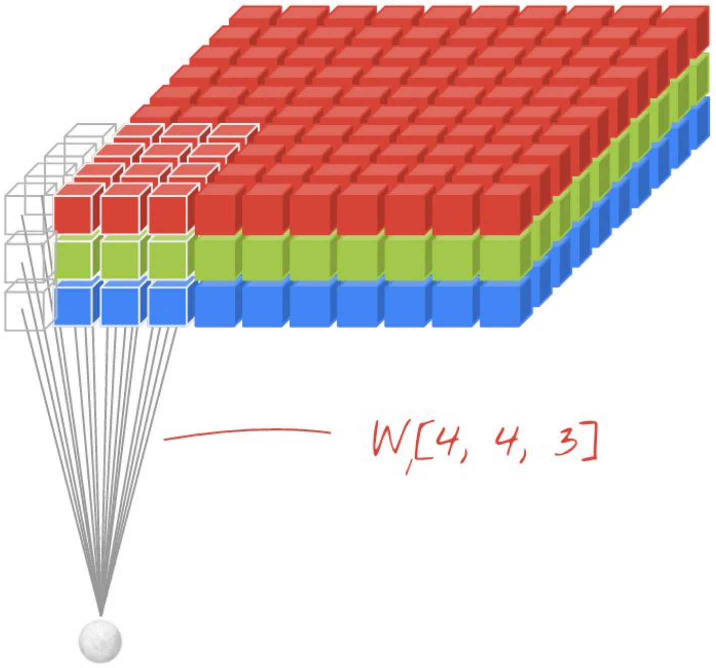
2D



3D

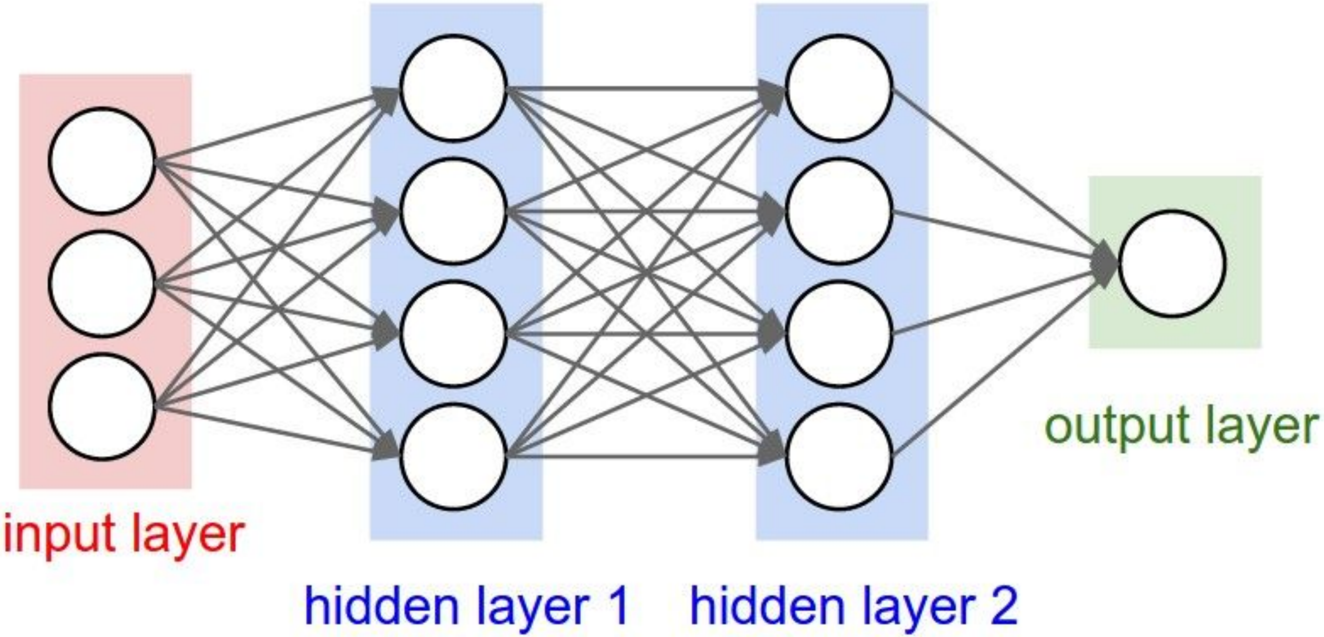


Convolution



Demo

Convolution by Neurons

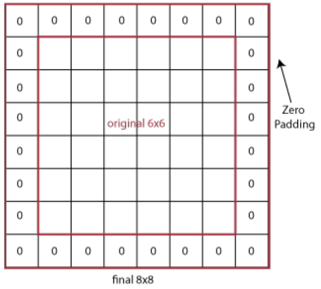


Convolution

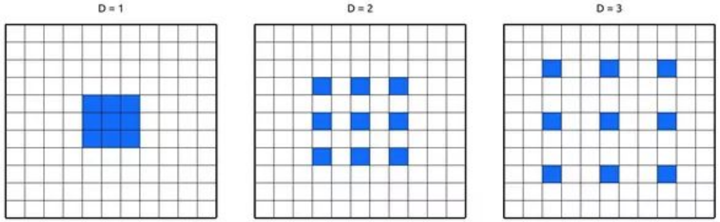
ZERO PADDING - zeros can be added to the feature map to increase the size of the feature map. The idea is to keep the size of the feature map the same throughout

STRIDES - Number elements that should be skipped in the feature map while doing convolution

Zero Padding



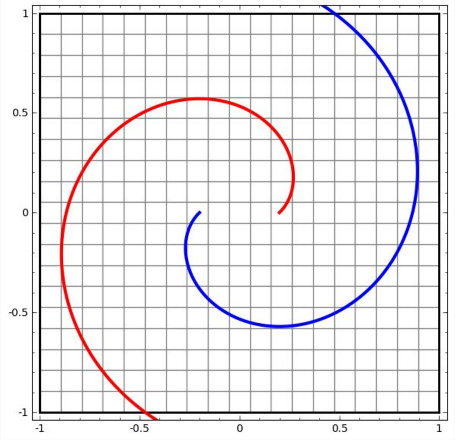
Convolution on a feature map with three different strides

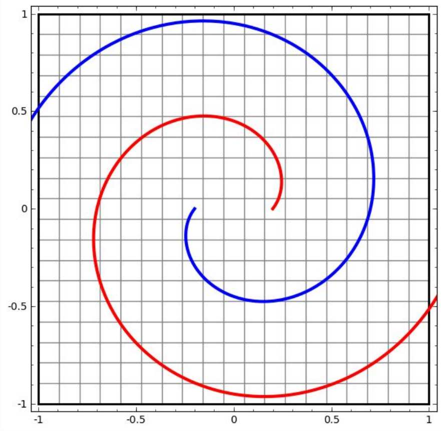


What does deep learning learn?

- MLPs without bias term model linear transformation
- MLPs with bias term model affine transformation
- The activation functions introduces further non-linearities
- Deep learning learns a transformation of a feature space that becomes linearly separable in a different topological space
- Link - [convnetjs]

Deep Learning

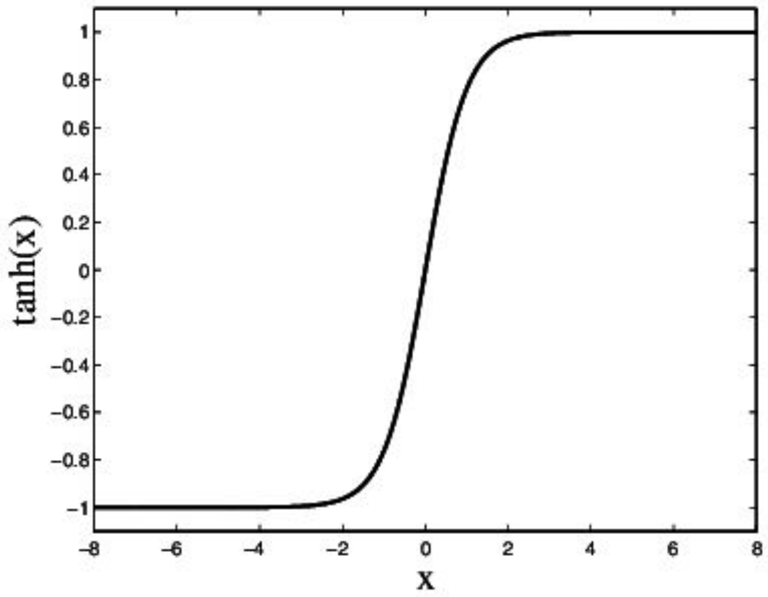




Soliton
Vision for a Better World

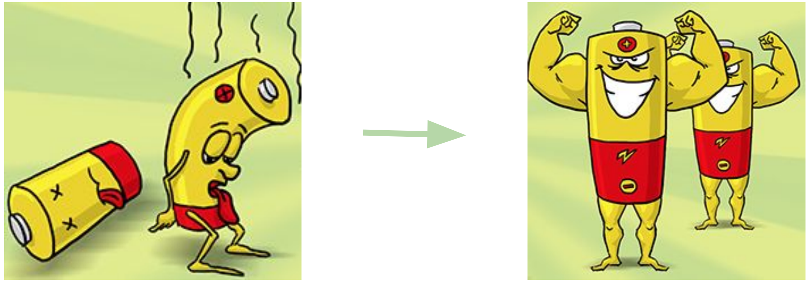
Problem with tanh layer /sigmoid

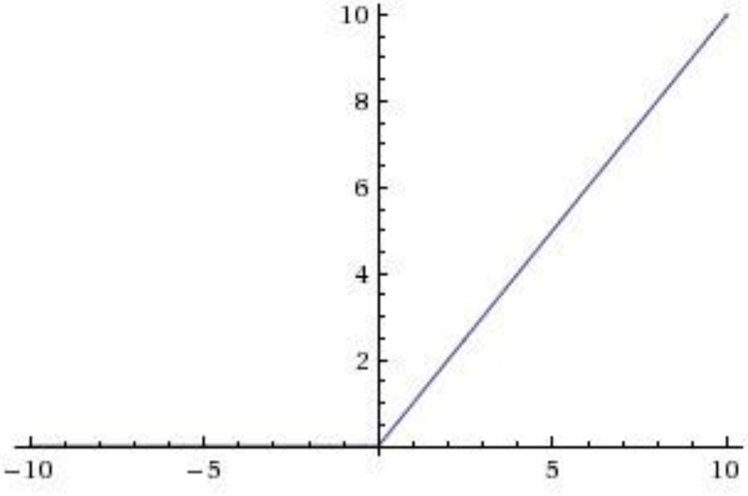
- The tanh/ sigmoid activation function squashes a real number in between zero and one
- It involves expensive operations, hence slows down the training process
- Its output saturates at both ends, hence produces “vanishing gradient problem” i.e., the gradient becomes zero at these saturation points and the network cannot learn weights based on backpropagation



ReLU

- Does not saturate at extremes, hence allow gradients to propagate through larger networks
- Sparse activations - A characteristic property of biological neurons
- ReLUs are computationally inexpensive.
- Though Relu's gradient is undefined at x=0, it is not practically a huge concern

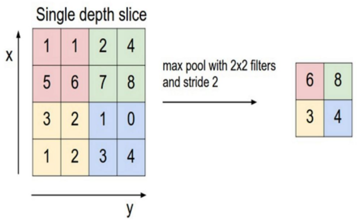




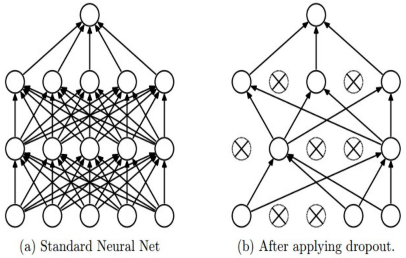
h=max(0,a)

Other layers

Pooling layer - Down samples the size of feature map. Promotes translation invariance

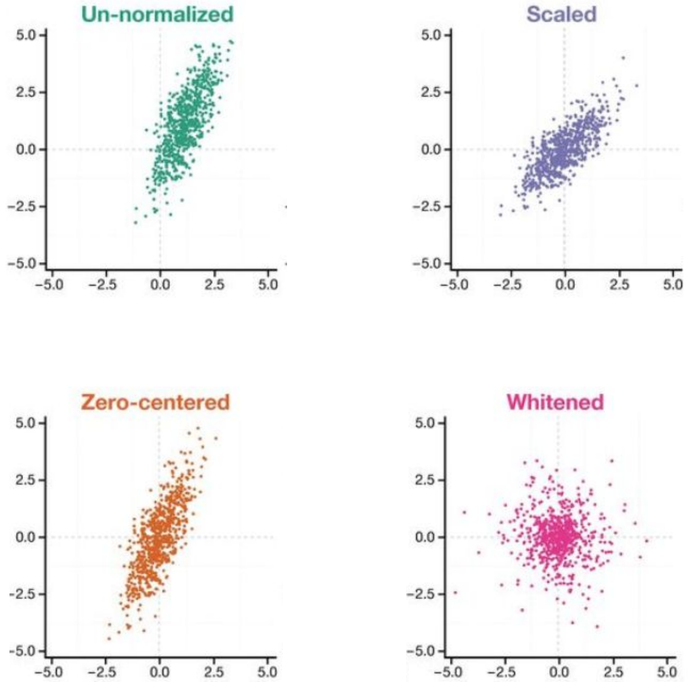


Dropout layer - Randomly removes a few neurons in the connection. Reduces overfitting

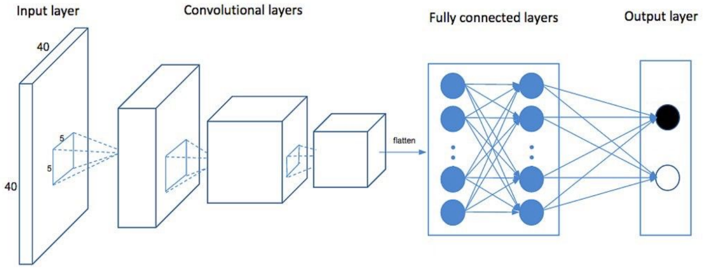


Data Preprocessing

Does not improve accuracy generally. But improves the speed of training



CNN

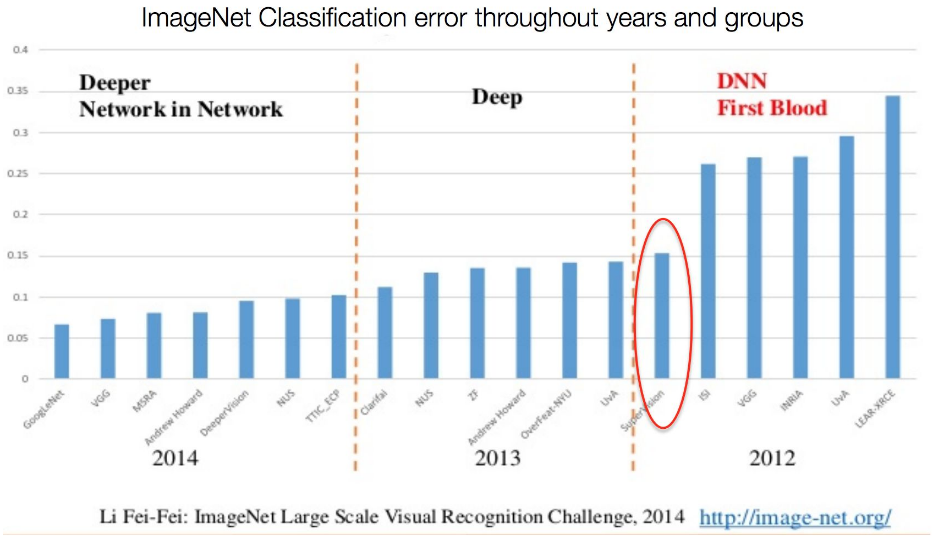


FC / Fully connected layer:-

- The neurons in the normal layers in CNNs are not connected to all neurons of the previous layer. But towards the end of the network, all neurons in a layer are connected to all neurons in the previous layers.
- This layer increases the representational power of the network but involves more parameters than a convolutional layers
- This layer is very useful for making classification decision based on the learnt feature maps. Fully connected layer gives the power to a neuron to mix all features of the previous layer, which is not possible with convolution.
- The last fully connected layer does the function of a SVM or softmax

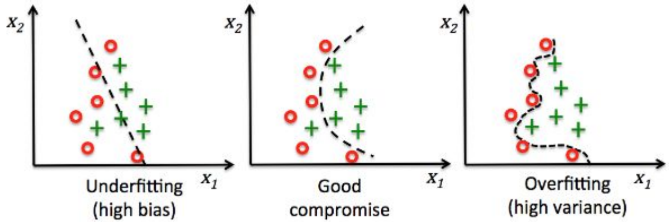
CNN: What Changed

ILSVRC

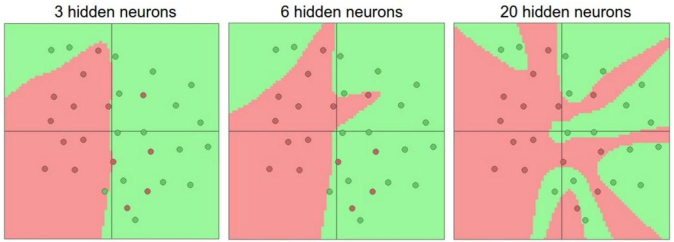


- ReLU
- Shared Weights
- Specialized Layers: CP
- GPGPU
- Availability of OSS Libraries/Datasets

Overfitting

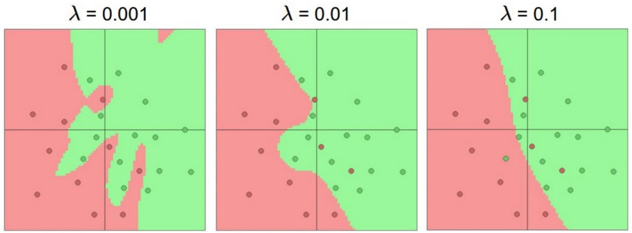


Noise in the data is also fit by the model leading to overfitting



Overfitting increases with number of neurons. But it is not a good practice to decrease the number of neurons but to use regularisation

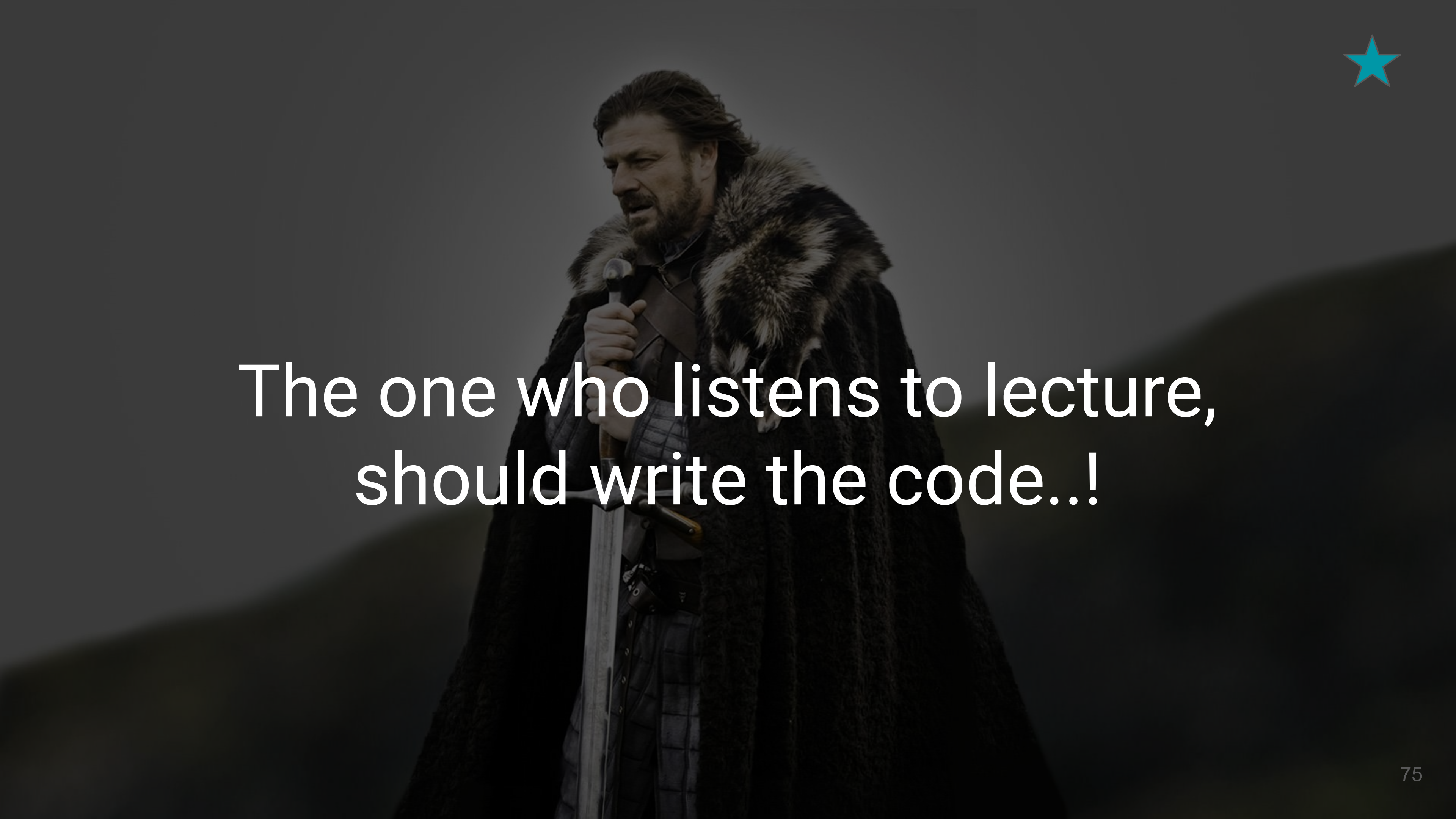
Regularisation



Regularisation helps to generalise to the given data while maintaining the representational power of the network

Types of regularisation:-

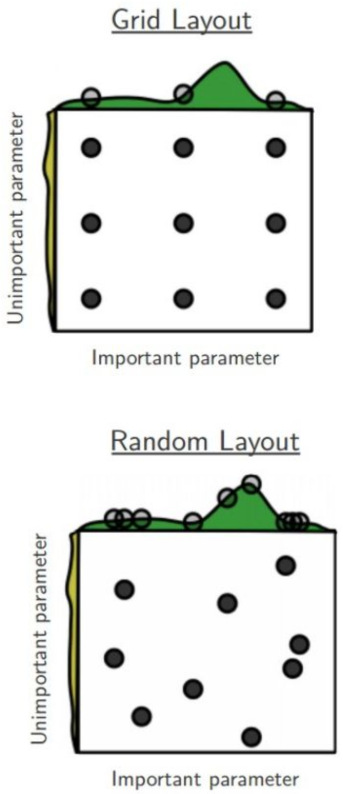
- L1 regularisation - Prefers sparse distribution
- L2 regularisation - heavily penalizes peaky weight vectors and prefers more diffuse weight vectors.
- Combination - Combination of both



The one who listens to lecture,
should write the code..!

HYPERPARAMETER STRATEGIES

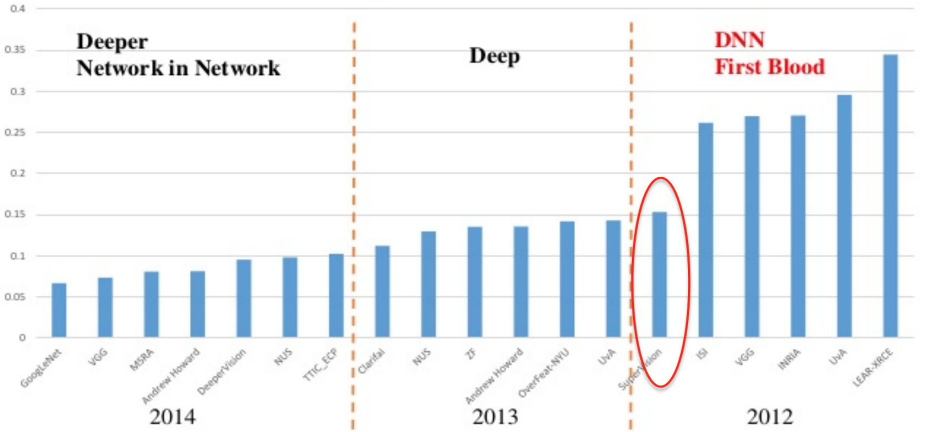
- Hyperparameters are parameters which are not learned by the algorithm but should be manually fixed empirically. Deep learning involves several hyperparameters like learning rate, batch size, size of convolution, stride etc..
- First do a coarse search with small epochs and fine search with larger epochs
- If the best value for a hyperparameters occurs in the border of an interval, do a double check by trying values beyond the boundary so that so you don't miss out on the optimal hyperparameter
- Don't do grid search, always prefer random search



Impact of deep in ILSVRC

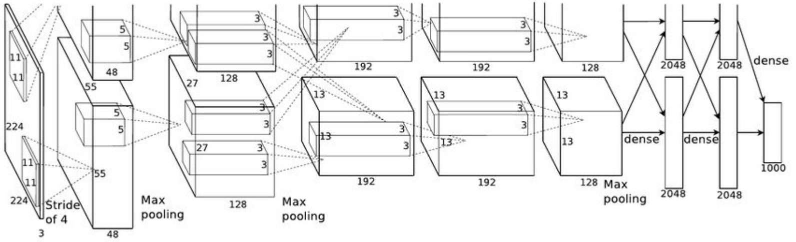
ILSVRC

ImageNet Classification error throughout years and groups



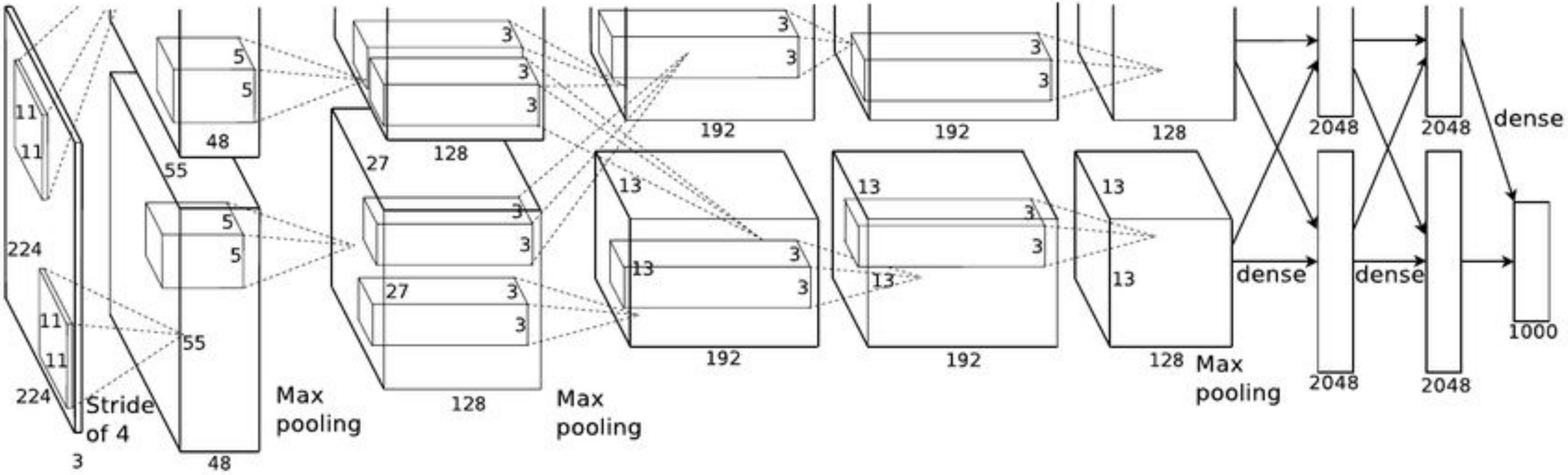
Li Fei-Fei: ImageNet Large Scale Visual Recognition Challenge, 2014 http://image-net.org/

ALEXNET - The gamechanger



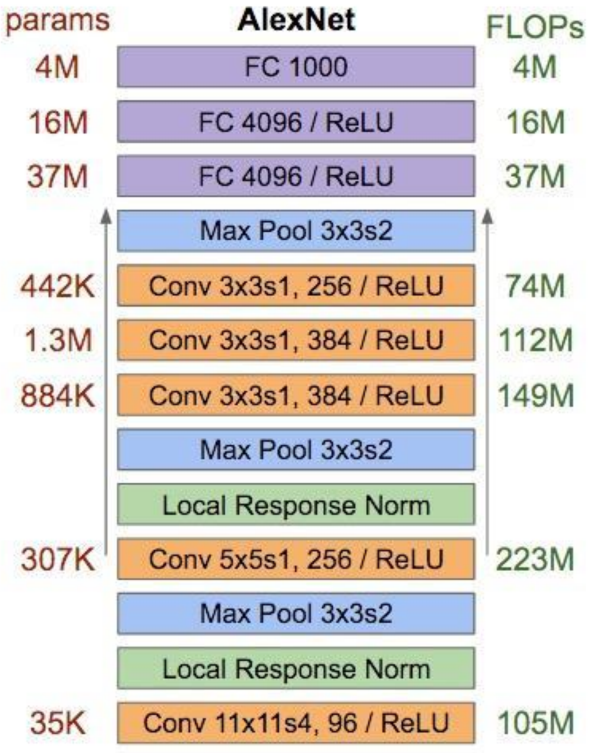
Impact of deep in ILSVRC

ALEXNET - The gamechanger



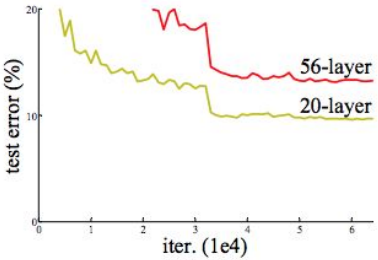
Alexnet architecture details

It consists of 8 layers - 5 convolutional layers and 3 fully connected layers.

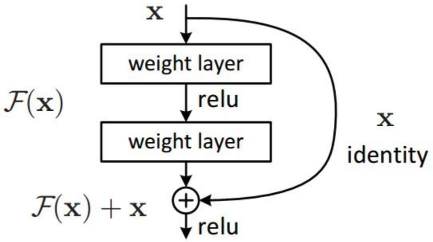


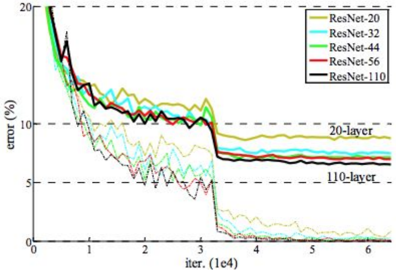
Residual Networks

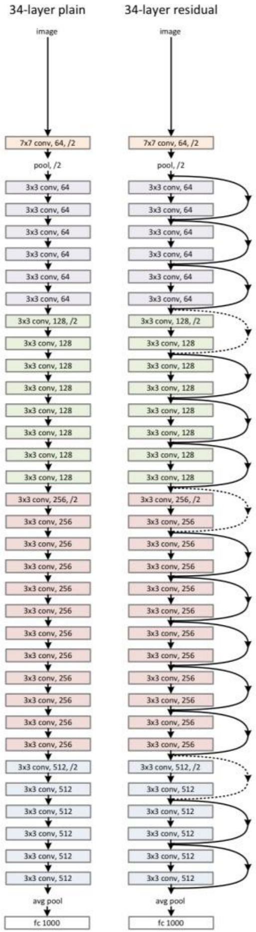
Counterintuitively, error increases with increase in depth beyond a certain point



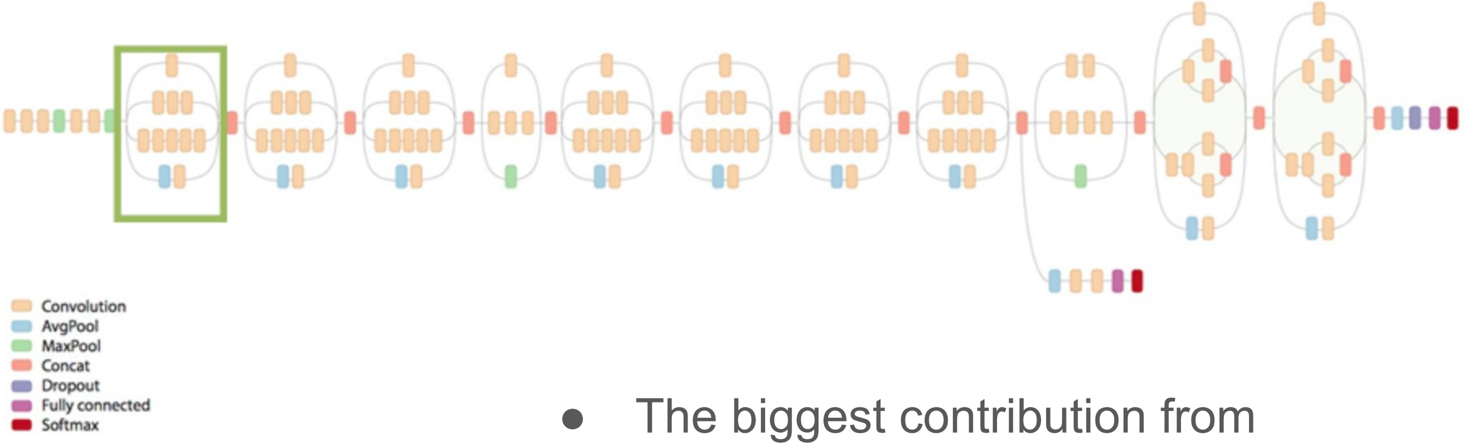
The idea of skip connection suggests that it is easier to learn minor modifications to identity functions

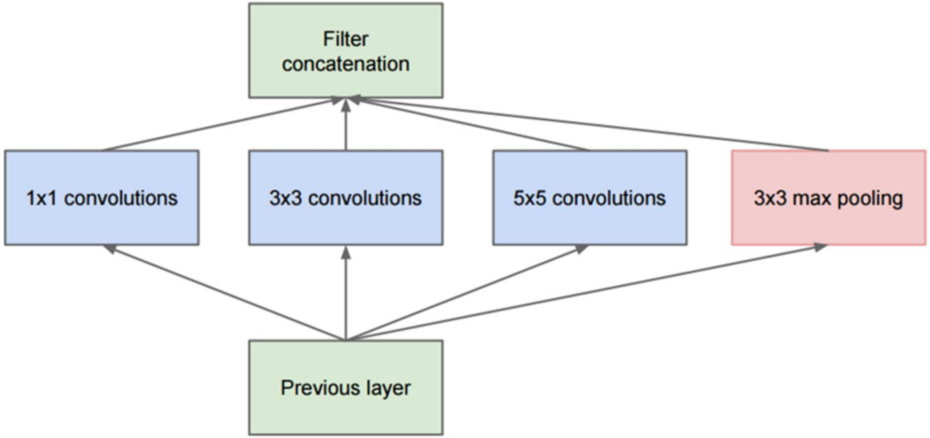






Google net

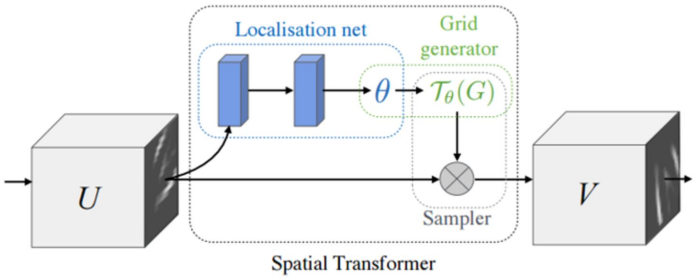


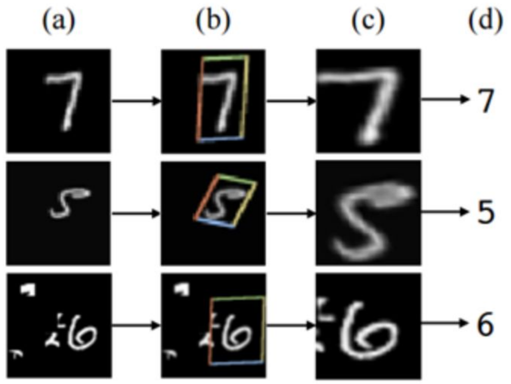


- The biggest contribution from google net was the introduction of inception module - A module where features from multiple layers can be mixed together
- Inception provides a way for combining local and global features

Spatial Transfer Networks (STN)

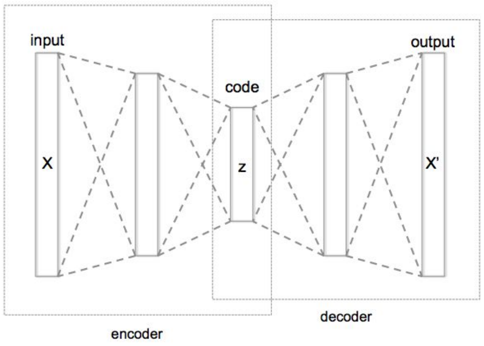
- Introduces a network to make images invariant to rotations and translations
- It consists of a localisation network that computes the spatial transformation, creation of sampling grid through a grid generator and a sampler which warps the input based on the generated grid.
- Link for [STN]

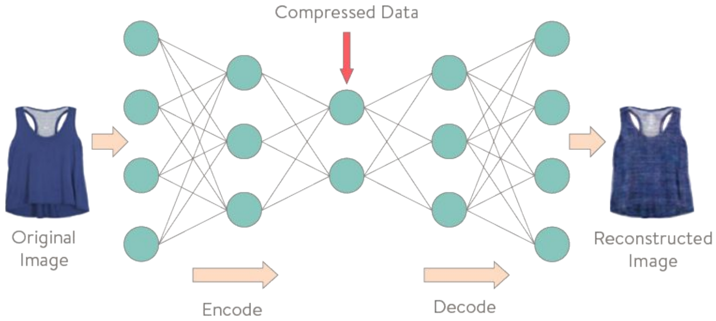




Autoencoders

- Autoencoder is a network which is typically used to learn a fixed number of features that can best represent the data
- This type of networks can be used for data compression



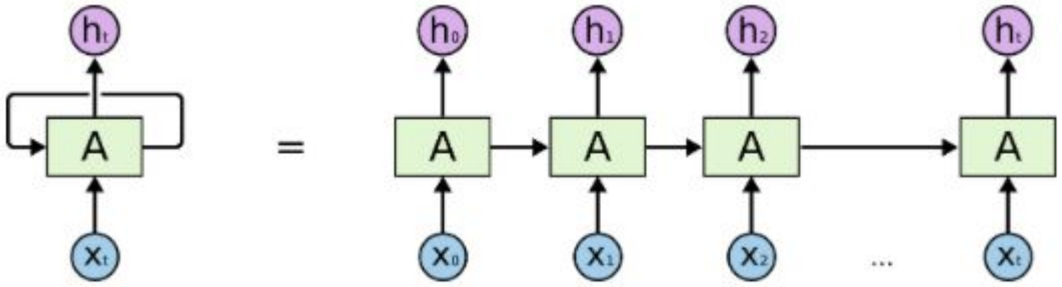


Recurrent Neural Network

Unlike feedforward neural network, RNN can have loops.

Useful for Sequential inputs.

Useful in applications where memory about the past output can play a role in predicting the current output



Visualization examples

Techniques:-

1. Layer activations
    a. Visualise the activations of the network. When we use ReLU, the activations starts out relatively blobby but spreads out during learning.
    b. Some activations may be all zero indicating high learning rate.
2. Visualise weights
    a. Weights are the most interpretable on the first layer, which is looking at the input pixels directly.
    b. Weights from other layers can be visualised too. They will usually form some smooth patterns. Noisy patterns indicate that the network has not probably learnt well and needs to trained longer.
3. Retrieving images that maximally excite a neuron.
    a. A large dataset of images is taken. The images which fired maximally for some neuron are recorded. Hence this will give us a good insight into what the neuron has learnt.
    b. One problem with this is that each Relu neuron might not learn something sematic. It is the combination of several Relu neurons that learn something semantic.

Visualization examples

4. Low dimensional embedding :-

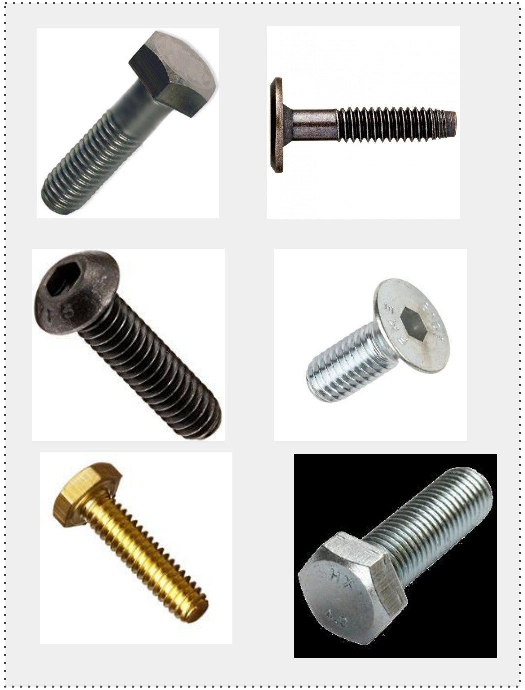
Several visualisation techniques have been proposed which convert the image vectors in high dimensional space to a 2-D space, preserving the pairwise distance between any two points. The best known technique is tsne- Embedding

5.Occluding parts of image (for classification):-

We can set a patch of the image to be all zeros. We can iterate position of the patch throughout the image and record the probability of correct class label as a function of position. A 2-dimensional heat map can be produced through such procedure. The probability should reduce considerably at position where the actual object is placed in the image.

Be careful with DATA

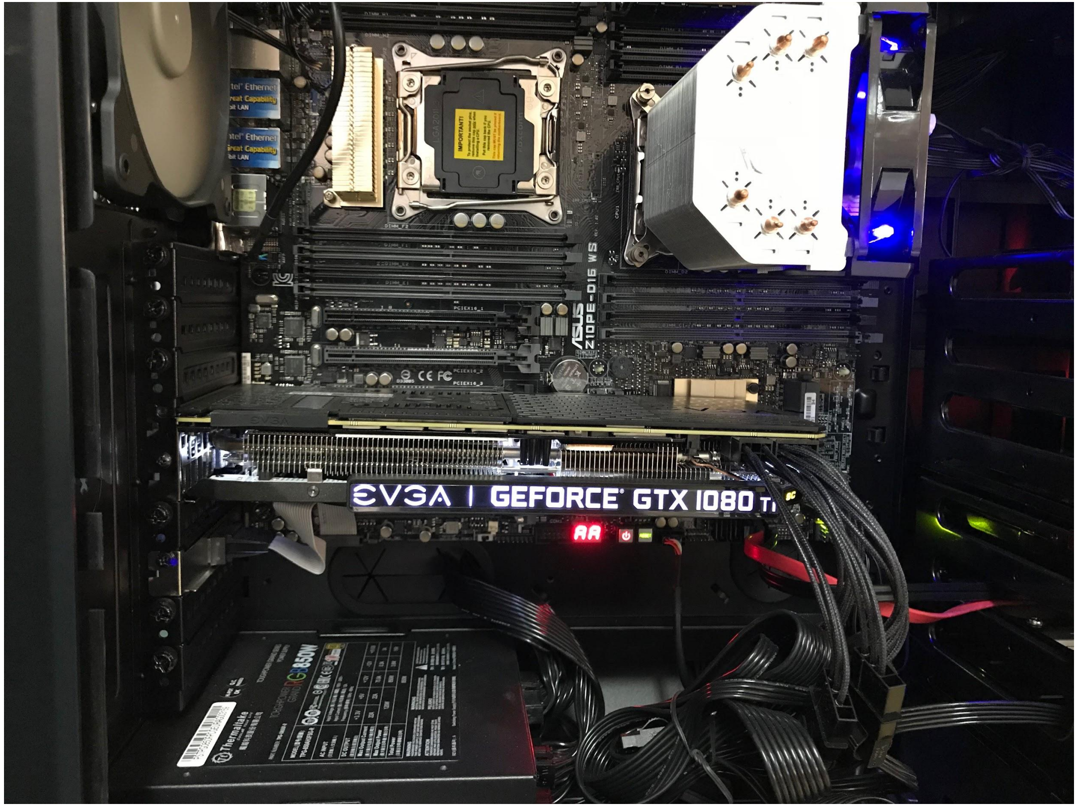
Nuts



Bolts

?

Challenges (GPU for Training)




GAME OVER