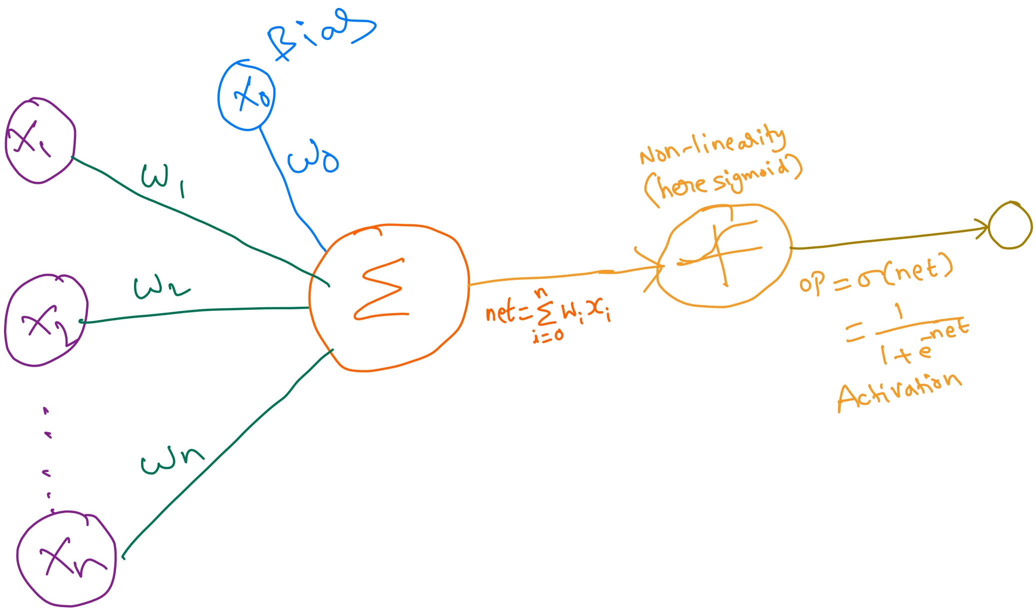
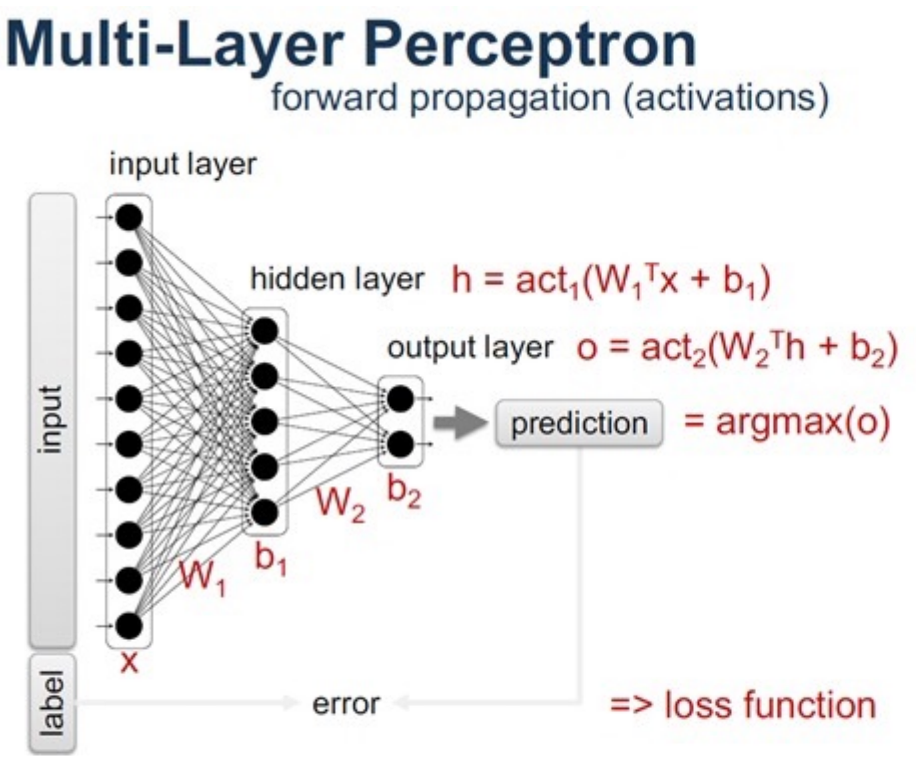
Q What is Perceptron.

It contains Input Layer and output layer, Connected with Params.



Q Explain MLP

MLP- Multi Layer perceptron. Feed forward fully connected network with multiple layers.
Feed Forward: Information propagated in forward direction starting from input, through hidden layers to output layer.



1. How does gradient descent work?

Bonus: What are the differences between stochastic, batch and mini-batch gradient descent?

The idea is we want to reduce cost function, So we will optimize the params.

How ?

By Computing the gradients. How ?

By differentiating total Error w.r.t the parameters.

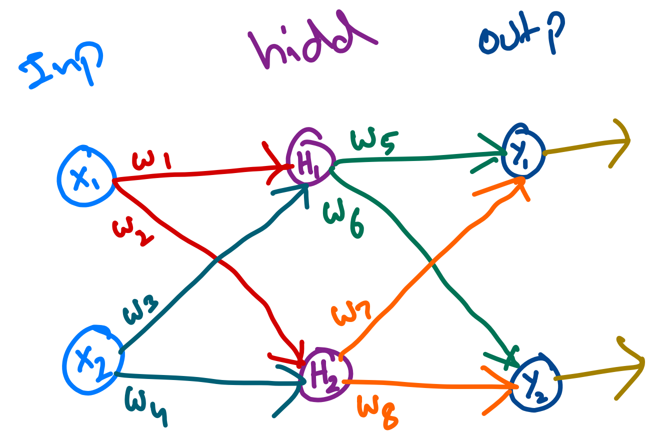
Stochastic: Only one training sample will be processed and also the gradient will be calculated per sample.

Batch: the Entire train Samples (very long)

Mini-Batch: Trade-off between Stochastic & Batch

2. Explain the back-propagation algorithm and its importance for neural network training!

Consider the following Network



$w_1, w_2 \ldots\ldots w_8$ are params.

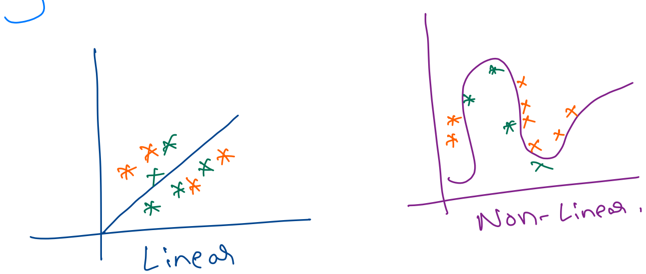
* First we do forward-passing.
* Next at the end we compute loss
* If there is no loss then no-need to optimize params. Else we should optimize the params to reduce the error. How?
* Imagine our loss func f(l) we want to update the params. In-order to update params we should back-propagate

For e.g.: $\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial y_1} \times \frac{\partial y_1}{\partial w_1}$

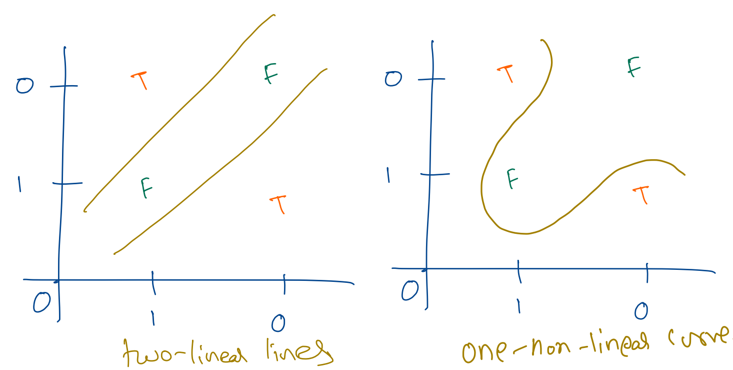
3. Why do we need non-linearities?

Linear models can only represent linear functions, so the model cannot fully understand the interaction between two input variables. Multiple linear layers can be reduced to a single linear transform.
Ex: Inability of represent XOR with just linear functions

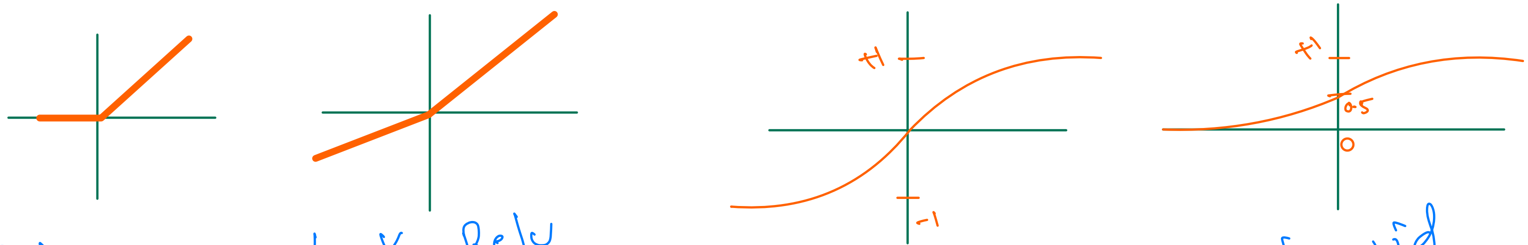
* Many real world problems can't be solved with linear activations.



* Better Example - XOR Problem.



4. Which activation functions do you know? Sketch them and discuss their pros and cons!



Relu
$\max(0, x)$

Leaky-Relu
$\max(\alpha x, x)$
$(0 < \alpha < 1)$

Tanh
$[-1, 1]$

Sigmoid
$[0, 1]$

* Relu can't handle negativity & Pro is that it is extremely sharp. convergence is faster
* L-Relu it can handle negativity. If it is too low then it will saturated towards 0.
* If the mode is too deep in Sigmoid then exp & vanishing Problem. Saturation problem.

Function	Uses
ReLU	Called as Rectified Linear Unit. **Formulae**: max {0, input} → It is not differentiable at 0, but very rarely it happens to be this case. Used in hidden layers. **Advantage**: Almost like linear, so easy to optimize with gradient descent. **Disadvantage**: Gradient is 0 for input less than 0, so no learning. Have other variants like leaky ReLU that have gradients when input is negative.
Sigmoid	**Formulae**: 1/1+e^(-x) Used in both hidden and output layers. **Advantage**: Values bound between 0,1 so can be used to represent probabilities. **Disadvantage**: sigmoidal units saturate to a high value when x is very positive, saturate to a low value when x is very negative, and are only strongly sensitive to their input when x is near 0. The widespread saturation of sigmoidal units can make gradient-based learning very difficult.
Tanh	**Formulae**: tanh(x) Used in hidden layers. Bound between (-1,+1) **Advantage**: Resembles an identity function near 0. so, for network with very less activations, the tanh hidden layer will have effect same as being linear layer. **Disadvantage**: tanh units saturate to a high value when x is very positive, saturate to a low value when x is very negative, and are only strongly sensitive to their input when x is near 0. The widespread saturation of tanh units can make gradient-based learning very difficult.
Softmax	**Formulae**: exp(z)/ Sum all exp(z) Used in output layer. **Advantage**: Used in most classification models. Output bound between (0,1) so can be used to represent probabilities. Works well with log optimization functions such as negative log likelihood. Predicts class with highest probability, winner takes all principle. **Disadvantage**: objective functions that do not use a log to undo the exp of the softmax, fail to learn when the argument to the exp becomes very negative, causing the gradient to vanish.
Linear	**Formulae**: cx Used in both hidden and output layer. **Advantage**: Easy to optimize. **Disadvantage**: Not much of representation power. Multiple linear layers can be reduced to a single linear transform.

6) Explain Cost function?

Function that is used to figure out, how big the error is.

Ex:

Maximum likelihood: Pick the class, that maximized the probability of your data.

Instead of multiplying probabilities over several values, we maximize log over probability. In ML we always try to minimize, so we minimize negative log likelihood. This is also called as Cross-entropy.

Multi-class classification/ Binary Classification: Multi class Cross entropy, binary Cross entropy.

Regression: RMSE, MSE

Loss: The error measured by loss function is called loss.

Loss function: Defined for 1 observation.

Cost function: Defined for entire training set.

What is the winner-take-all principle and how does it relate to SoftMax?

SoftMax can be thought of creating some form of competition between units. The SoftMax outputs always sum to 1 so an increase in the value of one unit necessarily corresponds to a decrease in the value of others.

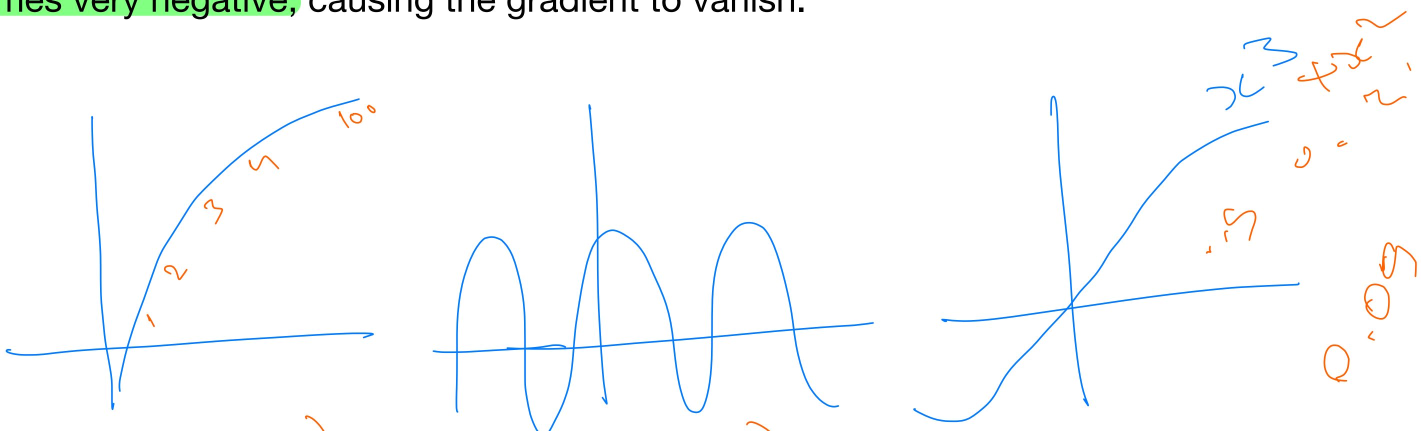
At the extreme (when the difference between the maximal ai and the others is large in magnitude) it becomes a form of winner-take-all (one of the outputs is nearly 1, and the others are nearly 0).

Apart from computing the derivative of the cost function, what are other applications of gradient descent in the context of deep learning?

- optimize input -> analyze the learned model (introspection) what about MSE

5. Why is the negative log-likelihood a popular choice as loss function?

Main requirement of NN: Gradient of the cost function should be big enough for the learning algorithm.
Problem: Most of the activation functions have exp in them, and this make the output saturate when the input is too high or too low.
Having a log in the cost function, cancels the exp in the activation functions.
MSE and MEA perform poorly when used with gradient based optimization because it doesn't have a log in its function. Objective functions that do not use a log to undo the exp of the softmax fail to learn when the argument to the exp becomes very negative, causing the gradient to vanish.



$Log(x^3+x^2)$ $\qquad$ $sin(x^3+x^2)$

* Consistent output
* It will shrink higher values to low.
* It minimizes the loss & maximizes the probability.

6. How do the choice of output units and the loss function relate?

* It depends on the choice of task

for e.g.: If it is Multi-class problem.

output units can have — Softmax — Cross Entropy

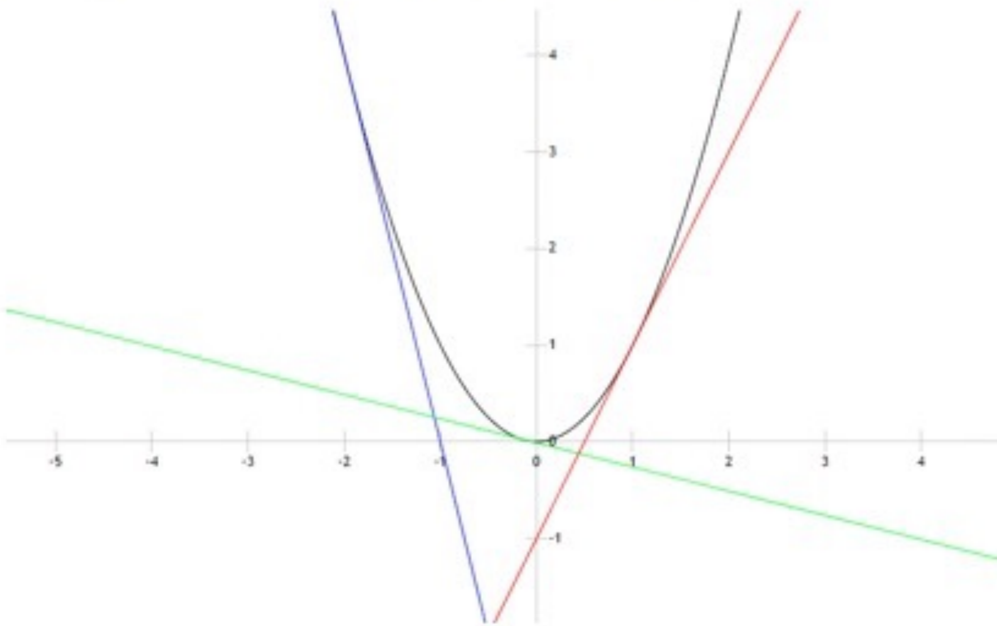
Binary — output units — Sigmoid — Binary Cross Entropy

Loss function help us to update params

and not help in classification.

7. What is Maxout? Sketch a layer with Maxout activation!

- piecewise linear => learnable act. function
- each piece computed by separate neuron



8. Why use a linear hidden layer?

Assume we have 1000 inputs, 1000 neurons and it's a Fully-connected layer

then we need

1000 X 1000 = 1000000 weights

Lets Keep a linear hidden layer of 100 neurons

1000 X 100 + 100 X 1000 = 200000

We reduced Params to 20%

Dimensionality Reduction

9. What does the universal approximation theorem (practically) mean?

You can learn everything mostly with a single hidden layer with "N" neurons.

It doesn't tell us how many "N" size

how many neurons are needed.

10. Sketch and explain the computation graph for the cross-entropy loss of an MLP with 1 hidden layer!