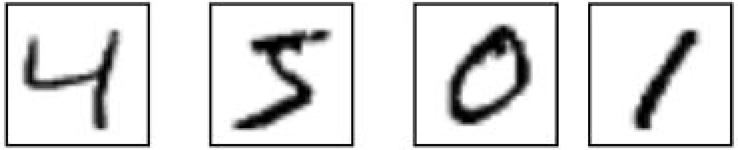
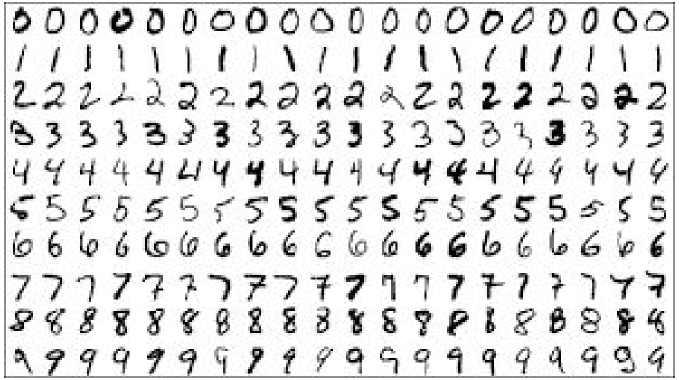
Intro to Machine Learning

Girish Varma
IIIT Hyderabad
http://bit.ly/2tzcXHu

A Machine Learning Problem



Given a image of a handwritten digit, find the digit.



No well defined function from input to output.

Programming vs Machine Learning

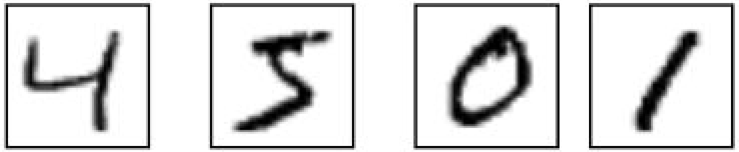
Programming:

Find the shortest path in an input graph **G**.

- Implement Dijkstra's algorithm for shortest path in a programming language.

Machine Learning:

Find the handwritten digit in an image.



- Collect (image, digit) pairs (**dataset**).
- **Train** a machine learning **model** to **fit** the dataset.
- Given a new image, apply the model to get the digit (**testing** or **inference**).

Dataset

- Consist of **(x,y)** pairs, **x** is the input and **y** is called the label.
- Examples
  - MNIST: x is a 28x28 b/w image of a handwritten digit, y is a digit in 0 to 9.
  - CIFAR10: x is a 32x32 color image, y is a label in {aeroplane, automobile, bird, car ..}. Y is given as a number in 0 to 9, and there is a mapping between the numbers and the correct label.
- Divided into train, test and validation.

Tensors

All data, intermediate outputs, learnable parameters are represented by a **tensor**.

A machine learning model transforms an input tensor to an output tensor.

Tensors have a shape.

1. Tensor T with shape [10,10] is equivalent to a 10x10 matrix. It can be indexed by 2 numbers. T[i,j] is a real number.
2. Tensor can be 3D. T with shape [5, 10, 15] can be indexed by 3 numbers i, j, k (i <= 5, j <=10, k <= 15).
3. Tensor can have arbitrary shape. T with shape [100, 32, 32, 3] can represent 100 color images each 32x32 in size.

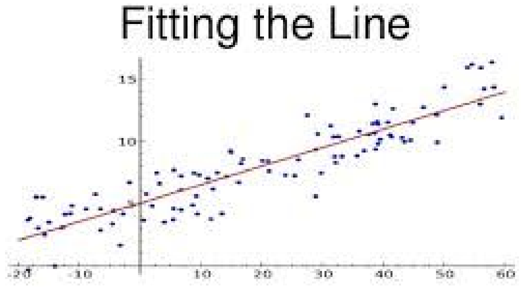
Model

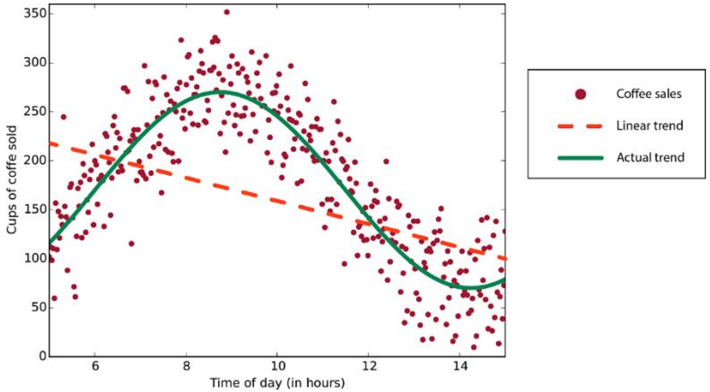
The function that maps the input to the output.

$$y = f_\theta(x)$$

A model has learnable parameters, $\boldsymbol{\theta}$.

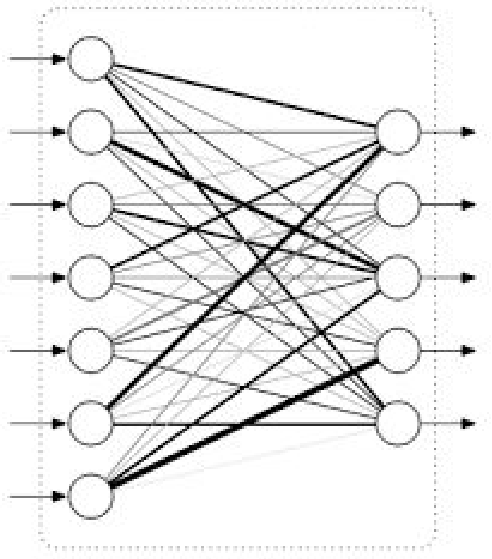
1. Fit a line to a set of points.
    - Slope and offset are learnable parameters.
2. Fit a degree 4 polynomial.
    - Coefficients are learnable parameters.
3. Fit a Multilayered perceptron.
    - Weights and biases are learnable parameters





The Neural Network Model

- Neuron or Perceptron
  - Input X is n dimensional, Y is 1 dimensional.
  - Has learnable parameters $W = (W_1, W_2, \ldots, W_n)$ (weights) and b (bias).
  - $Y = \boldsymbol{\sigma}(\sum W_i X_i + b)$
  - $\boldsymbol{\sigma}$ is a non linear activation function.



- Fully Connected or Linear
  - Y is also multidimensional (dimension m).
  - Has learnable parameters $W = (W_{ij})$ and $b = (b_j)$ where $i \le n$, $j \le m$
  - $Y = \boldsymbol{\sigma}(WX + b)$

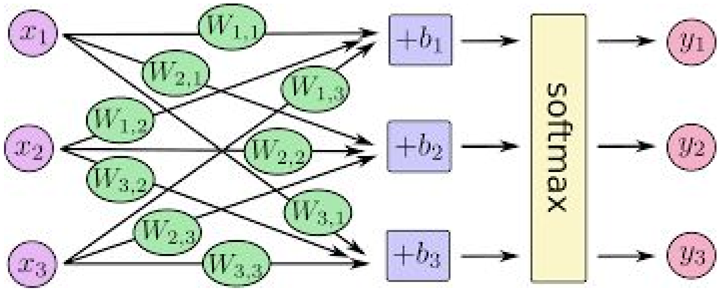
MNIST Classification

Input : x is a [28,28] shaped tensor, giving pixel values of the image

Output : y is a [10] shaped tensor, giving the probabilities of being 0 to 9.

If the dataset gives y as a digit, convert it to probability vector by one hot encoding.

Use Softmax function for converting real valued output to probabilities.



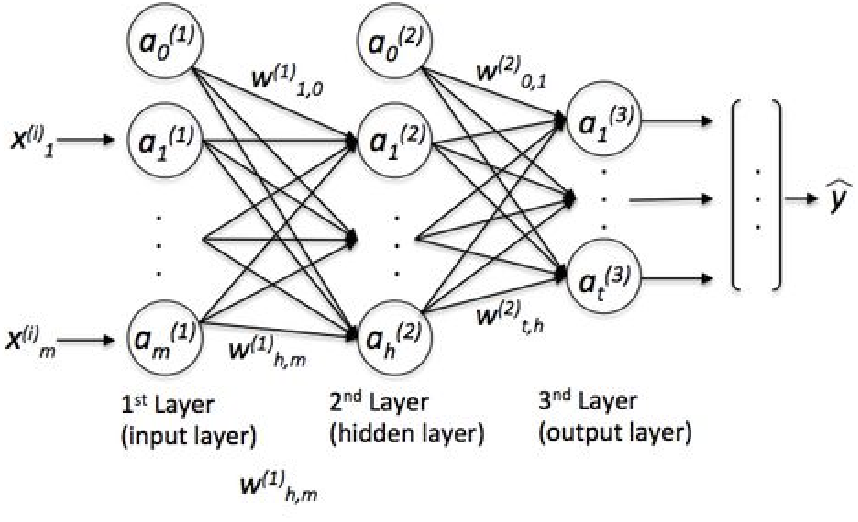
Multilayered Network

Complex data fits only more complex models.

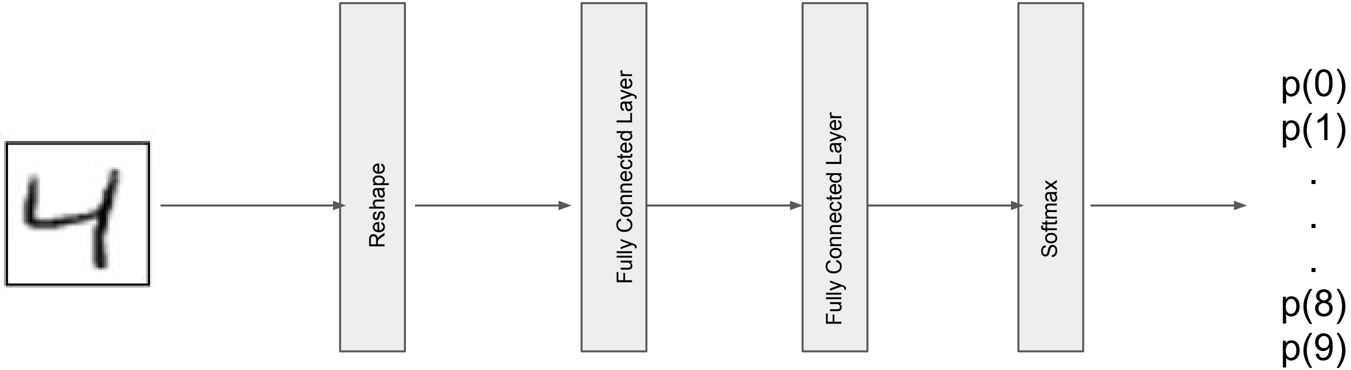
Obtain complex models by layering multiple linear layers.

Multilayered Perceptron (MLP)

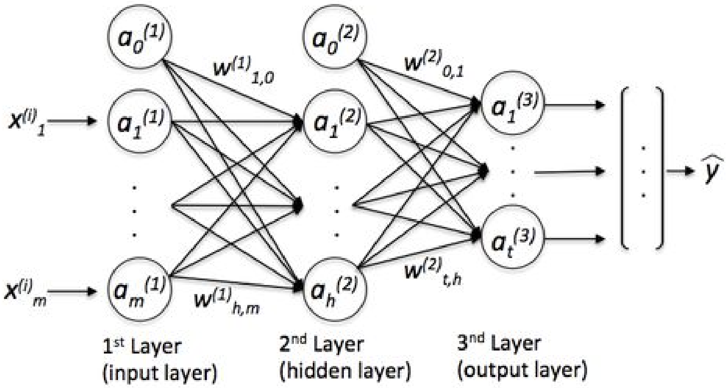
- Multiple Linear layers
  one following the other.
- $Y = \boldsymbol{\sigma}(V\, \boldsymbol{\sigma}(WX + b) + c)$
- Intermediate outputs are called hidden units.



A MLP model for MNIST



Predicted probabilities for different digits



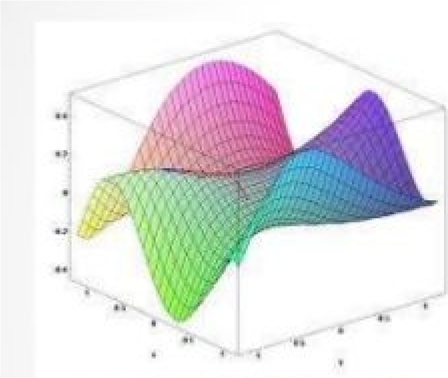
$w^{(1)}_{h,m}$

Training a Model

The process of finding the right parameters for the model.

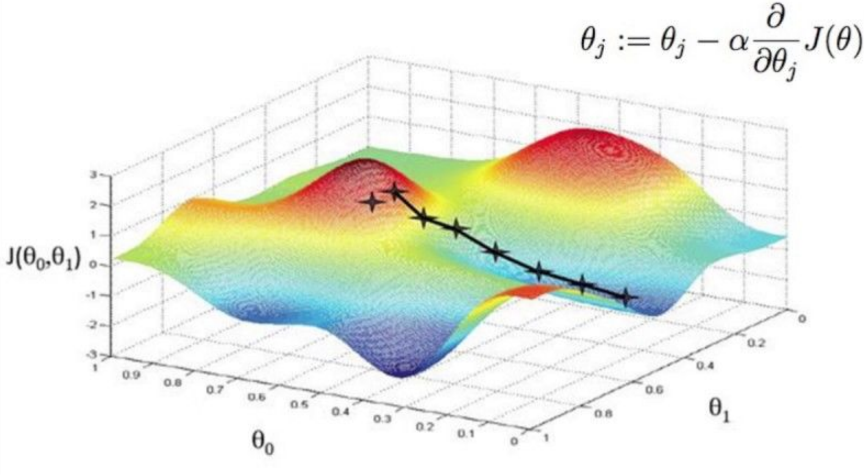
Loss Function

- Loss Function : A function that computes the difference between the predicted output and the correct output.
  - Eg: Mean Squared Error $(f(x) - y_{correct})^2$. $y_{correct}$ is also called the ground truth.
  - Eg: Cross Entropy Loss $\sum_i y_{correct}(i) \log y_{pred}(i)$



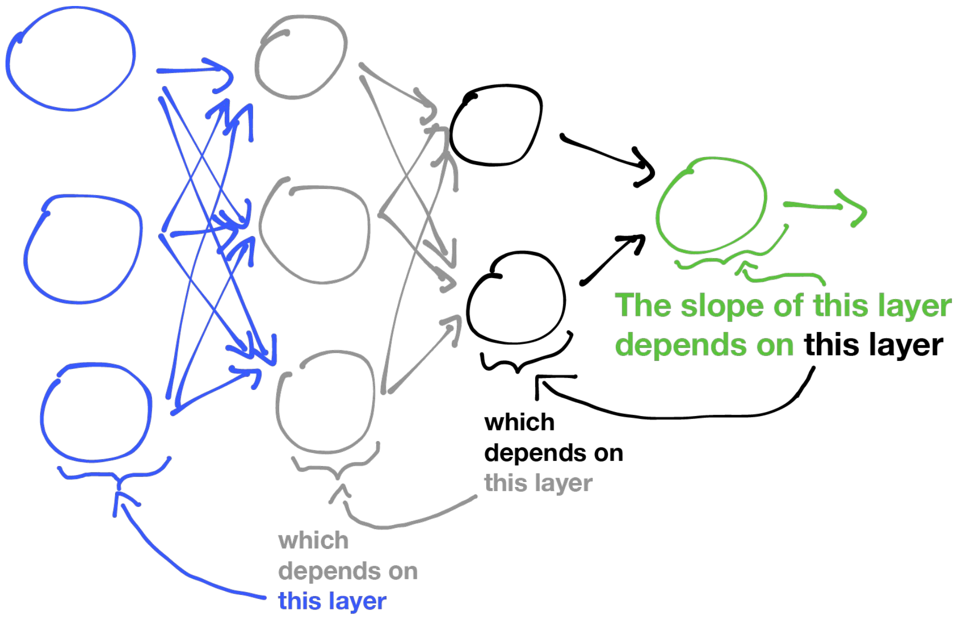
Gradient Descent

Gradient Descent : Change the parameters θ slightly such that the loss function decreases. Gradients are the partial derivatives of the loss function wrt. the parameters.



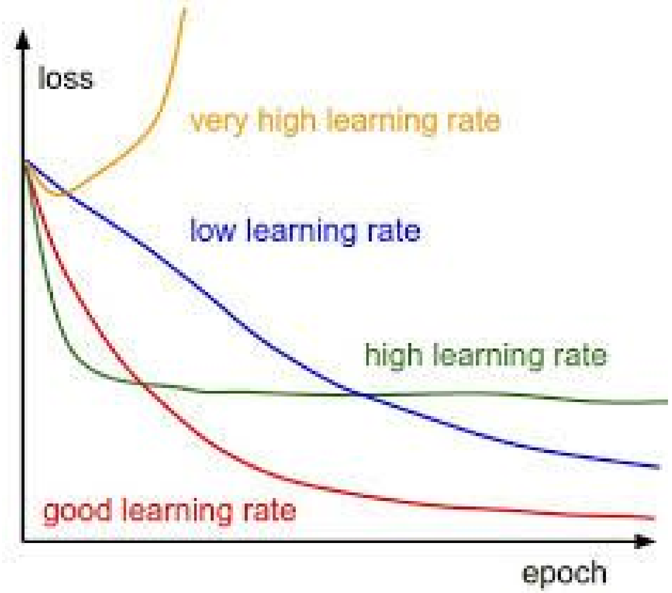
Backpropagation

Backpropagation : The process of finding the gradients of parameters in a multilayered network.

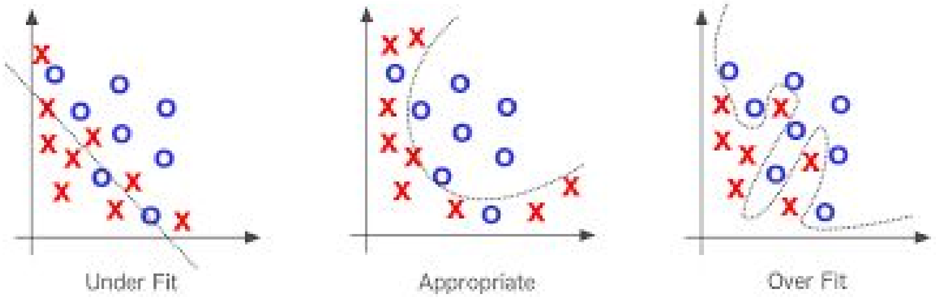


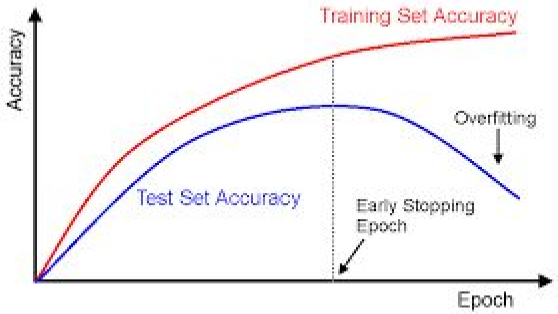
Training Algorithm

1. Initialize model with random parameters.
2. Repeat
    a. Take a small random subset of the dataset that will fit in memory (minibatch).
    b. Forward Pass : pass the subset through the model and obtain predictions
    c. Compute the mean loss function for the subset
    d. Backward Pass: compute the gradients of the parameters, last layer to the first.
    e. Update the gradients using learning rate



Overfitting





Testing or Inference

Some References

http://bit.ly/2tzcXHu [This presentation]

https://ml.berkeley.edu/blog/2016/11/06/tutorial-1/

https://ml.berkeley.edu/blog/2016/12/24/tutorial-2/

https://ml.berkeley.edu/blog/2017/02/04/tutorial-3/