Deep Learning Theory I

AI, ML and DL

Artificial Intelligence

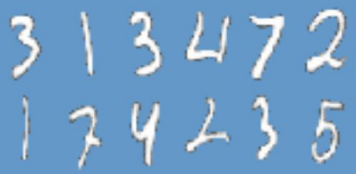
Any technique that enables computers to mimic human behavior

Machine Learning

Ability to learn without explicitly being programmed

Deep Learning

Learn underlying features in data using neural networks



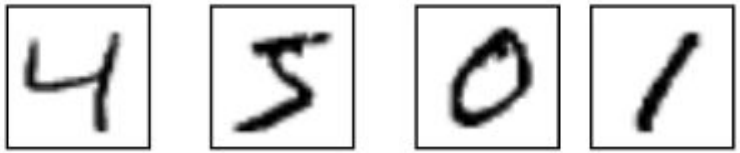
A Machine Learning Problem

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

No well defined function from input to output.

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.

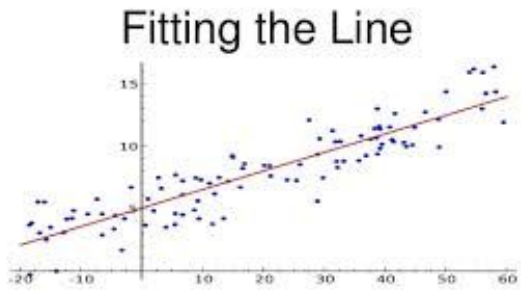
Model

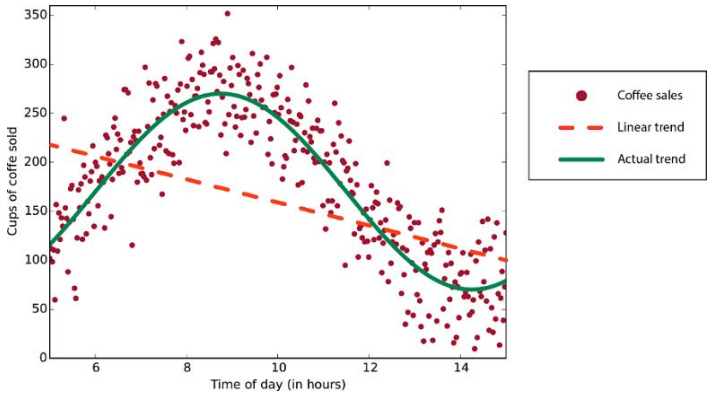
The function that maps the input to the output.

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

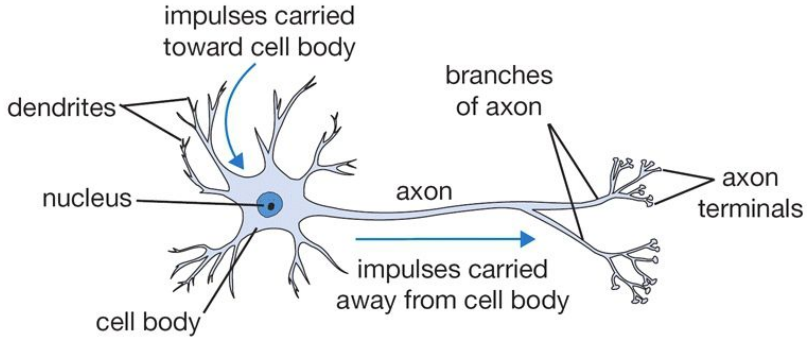
A model has learnable parameters, θ.

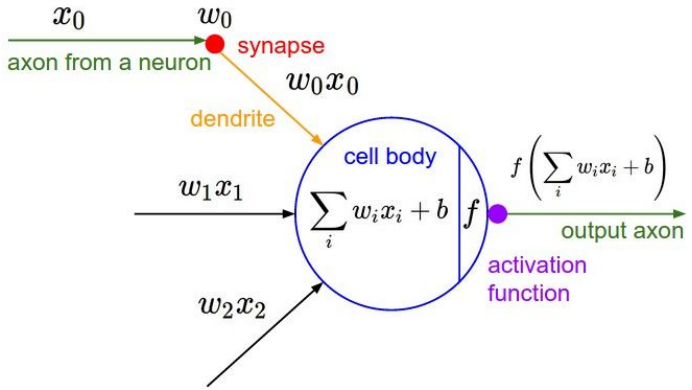
1. Fit a line to a set of points.
   - Slope and offset are learnable parameters.
2. Fit a degree 4 polynomial.
   - Coefficients are learnable parameters.
3. Fit a SVM (Support Vector Machine)





The Neural Network Model for Binary Classification



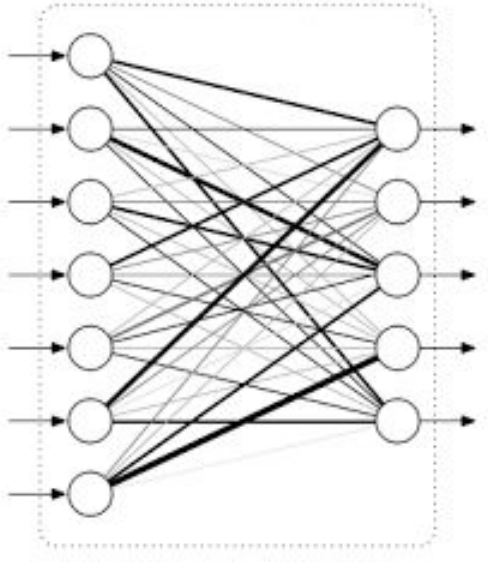


- Neuron or Perceptron
  - Input X is n dimensional, Y is 1 dimensional.
  - Has learnable parameters W = (W_1,W_2,..., W_n) (weights) and b (bias).
  - Y = **σ**(∑$W_i X_i$+ b)
  - **σ** is a non linear activation function.

Neural Network Model for Multi-Class

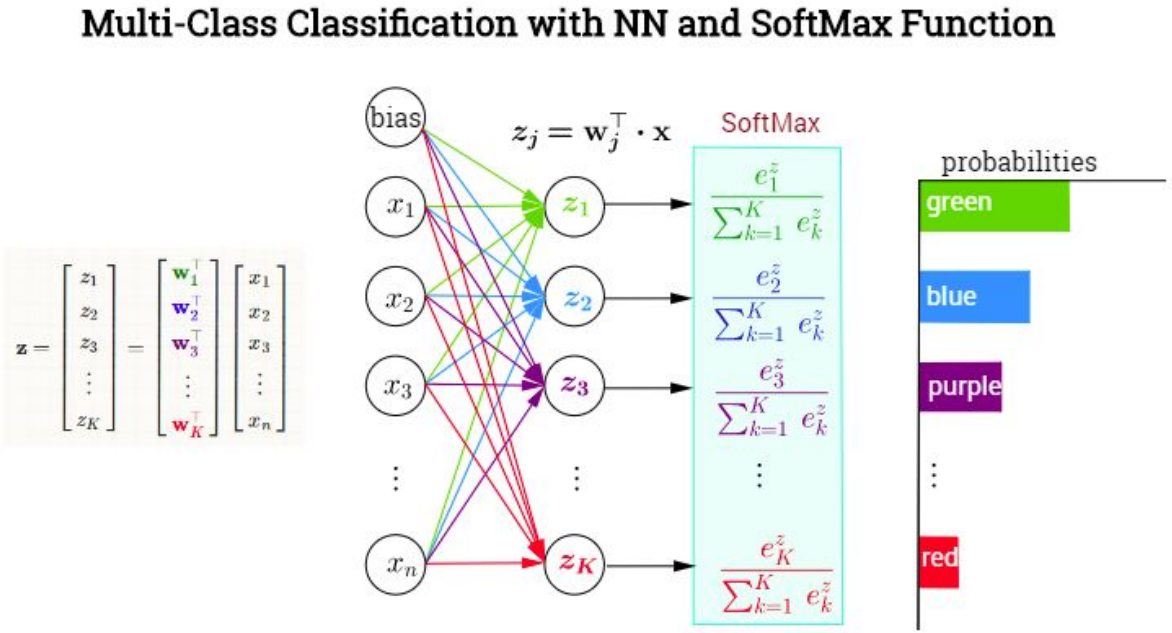
- Fully Connected or Linear
  - Y is also multidimensional (dimension m).
  - Has learnable parameters W = (W_{ij}) and b = (b_j)

    where i <= n, j <= m
  - $Y = \sigma(WX + b)$
- The class predicted for input X is the i with maximum value of Y_i.



Softmax Function

Converts output to prediction probabilities.



Convert vector to positive valued:

$$Z_i \Rightarrow \exp(Z_i)$$

Normalize to get probability distribution

$$\frac{\exp(Z_i)}{\Sigma_j \exp(Z_j)}$$

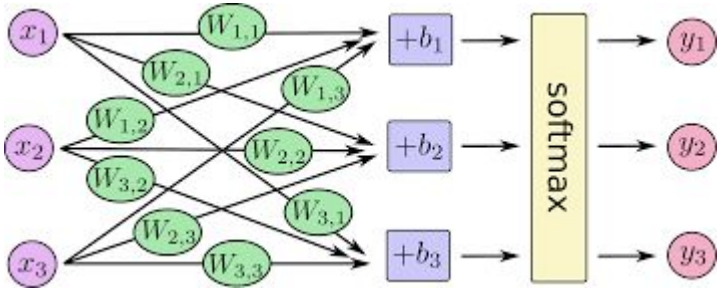
MNIST Classification

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.



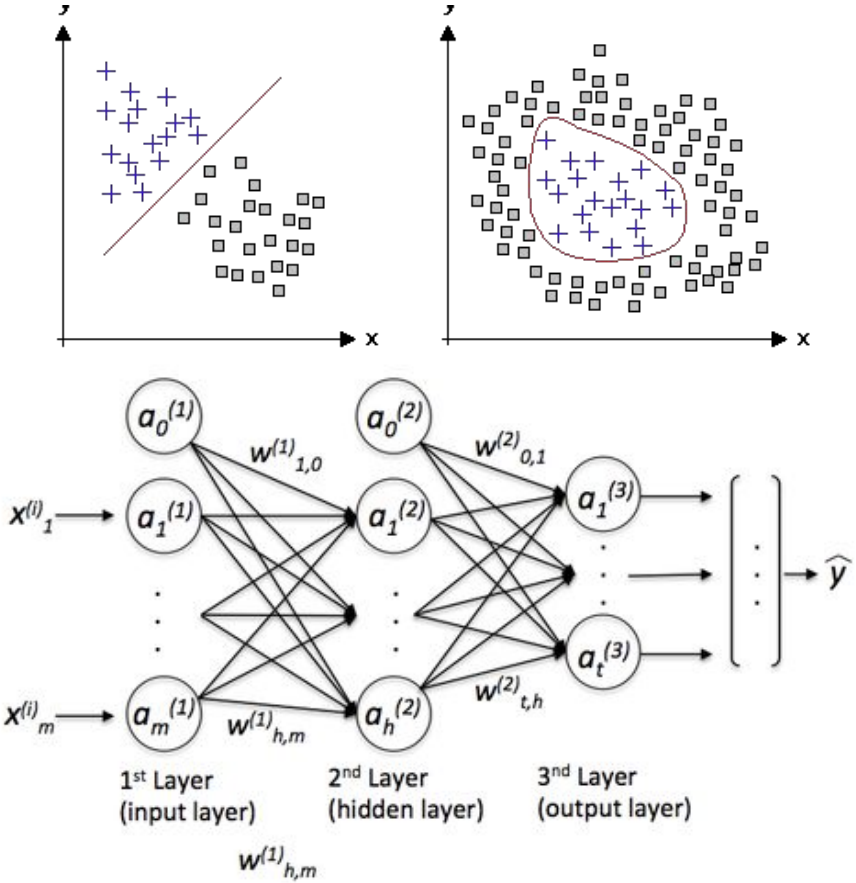
Multilayered Network

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 = **σ**(V **σ**(WX+ b) + c)
- Intermediate outputs are called hidden units.



A MLP model for MNIST

Reshape

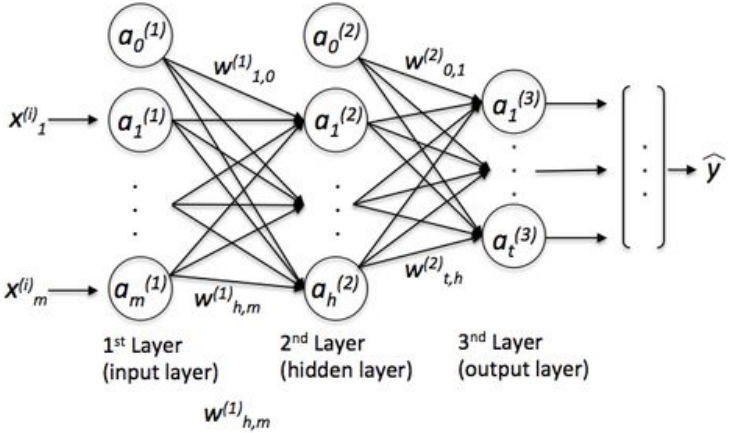
Fully Connected Layer

Fully Connected Layer

Softmax

p(0)
p(1)
.
.
.
p(8)
p(9)

Predicted probabilities for different digits

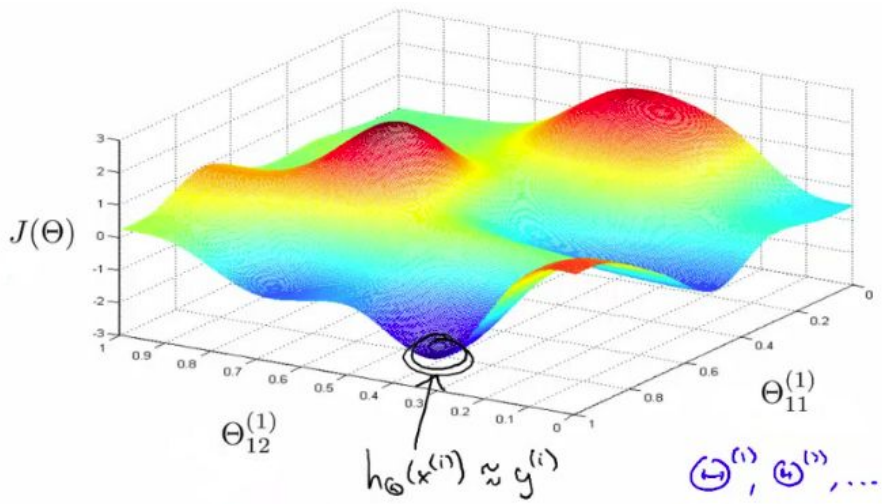


Training a Model

The process of finding the right parameters for the model.

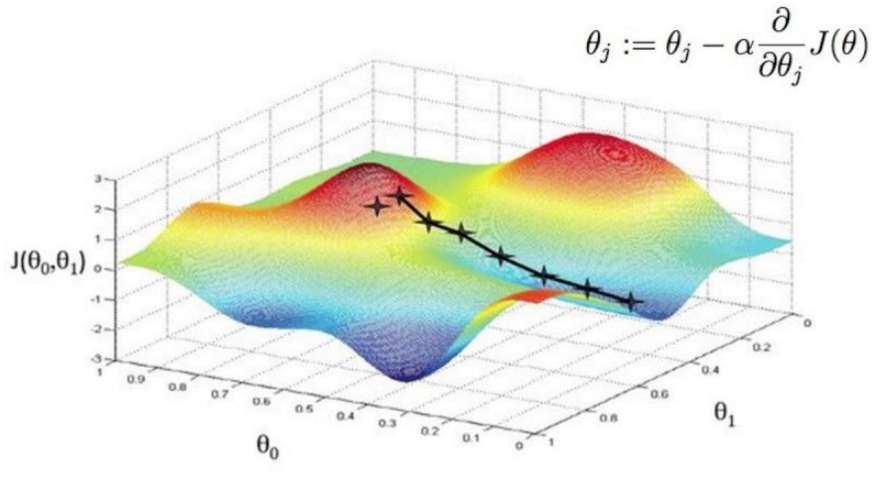
Loss Function

- 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.



Mathematically

Compute Loss (Forward pass):

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

Compute partial derivatives (Backward pass)

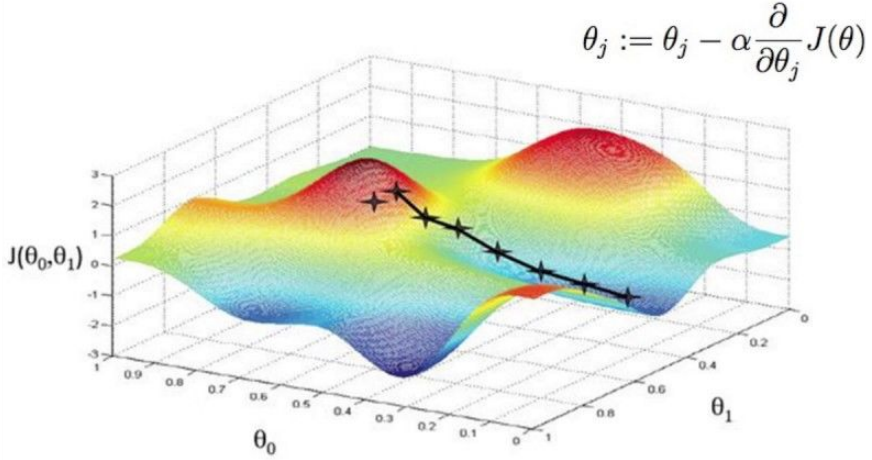
$$\frac{\partial J}{\partial m} = \frac{2}{n}\sum_{i} x_i \cdot (y_i - (mx_i + b))$$

$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_{i} (y_i - (mx_i + b))$$

Update Learnable parameters

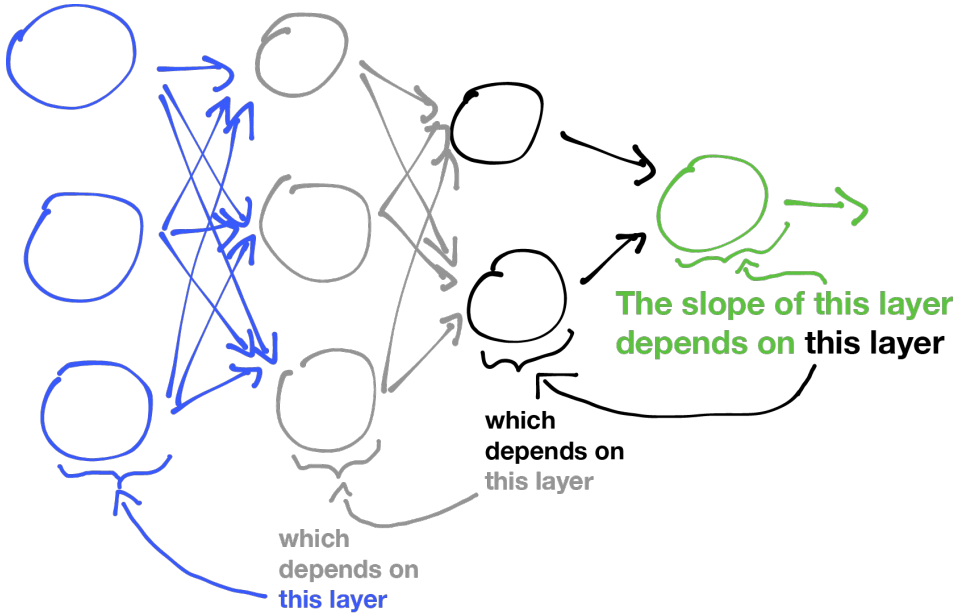
$$m := m - \alpha \cdot \frac{\partial J}{\partial m}$$

$$b := b - \alpha \cdot \frac{\partial J}{\partial b}$$



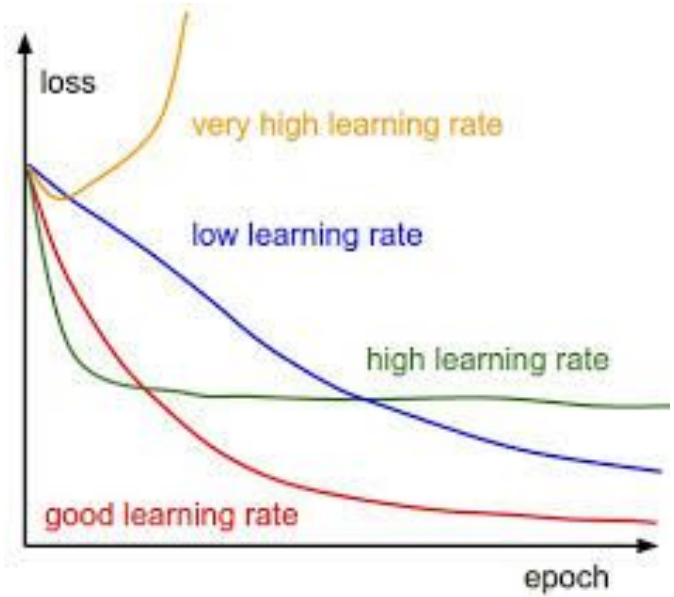
Backpropagation for Multilayered Networks

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

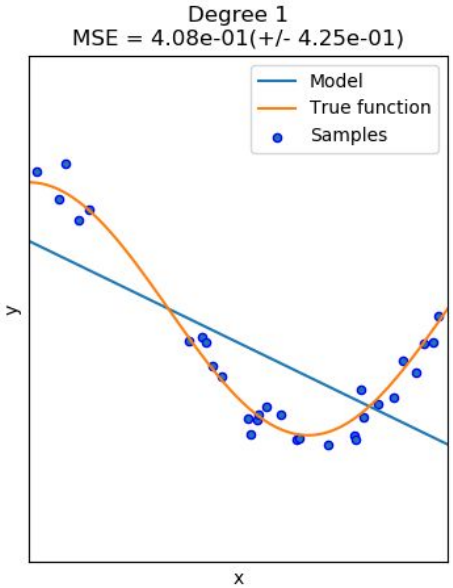


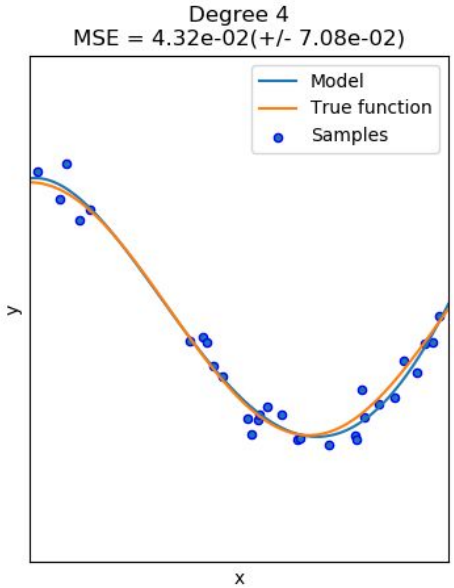
Training Algorithm

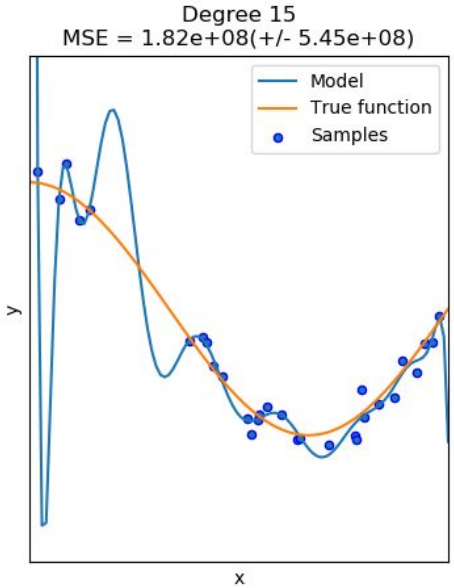
1. Initialize model with random parameters.
2. Repeat
    a. Take a small random subset of the dataset that will fit in memory (minibatch).
    b. Forward Pass : pass the subset through the model and obtain predictions
    c. Compute the mean loss function for the subset
    d. Backward Pass: compute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



Overfitting





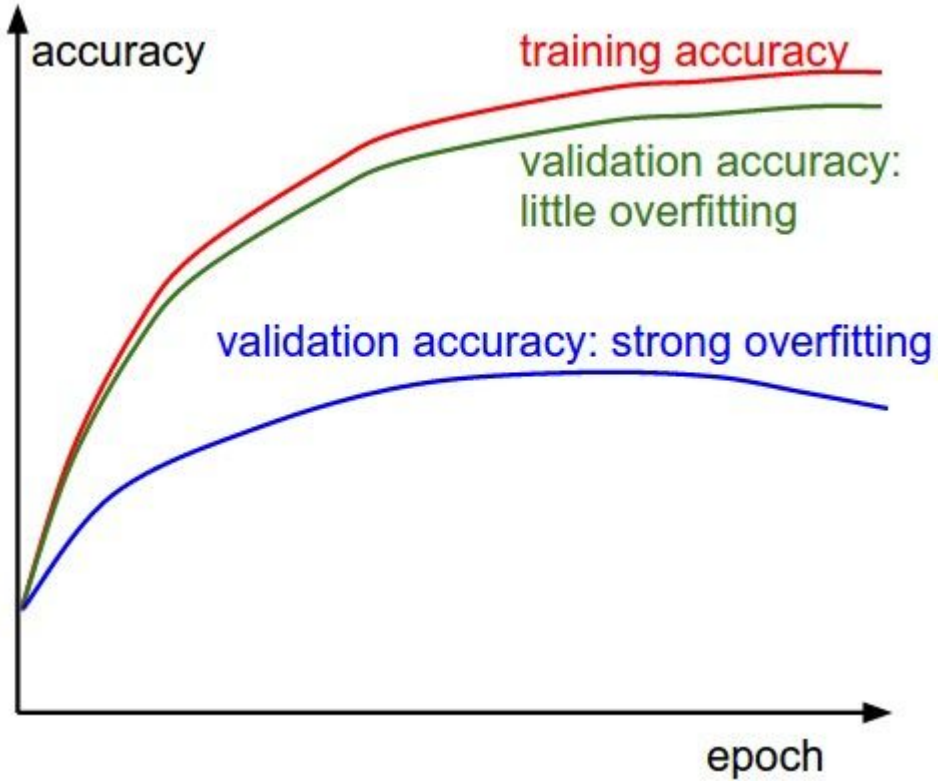


Overfitting during Training

To observe overfitting, we have test data not used for gradient descent.

Accuracy is computed periodically on test data.

If we observe training accuracy and test accuracy diverging, then there is overfitting.



Some References

Must watch video (30 mins)
https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

Blog (short read)
http://colah.github.io/posts/2015-08-Backprop/
http://www.wildml.com/2015/09/implementing-a-neural-network-from-scratch/
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/

Course Notes: (will require lots of time)
http://cs231n.github.io/ ,
https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv