AI BOOTCAMP - LECTURE 1.

The field of pattern recognition is concerned with automatic discoveries of regularities in the data through the use of computer learning algorithms and with the use of these regularities, it takes actions such as classification of the data into different categories.

MNIST dataset is one of the first test bed for Deep learning algorithms.

Machine learning approach often outperforms hand-coded-rule-based algorithms.

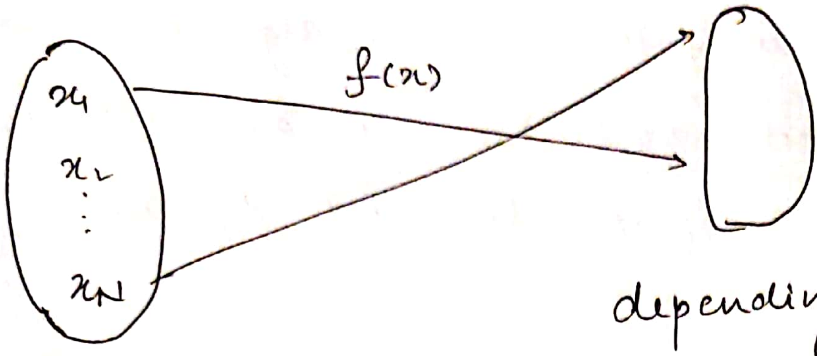
We usually pass a large set of training data, to a Machine learning "model". These dataset tunes various parameters of the models (adaptive).

Taking the example of MNIST dataset, we have 'N' images labelled with their categories.
The categories of the training data is represented as a target vector.

Basically, a learning algorithm is a mapping from an image to a category.

Image — Learning algorithm $f(x)$ — y

x matrix — target vector.



depending on the design of learning algorithm the target vector can vary.

Finding the right algorithm by iteratively tuning the parameters of an adaptive model is known as training (or learning).

Once the model is learned (or trained) then we test the model on the images which the model has not seen in the training phase, this dataset is called Test set.

The ability of a model to categorize the new samples correctly is called generalization.

Talk about pre-processing here.

—— x ——

Usually training methodologies can be represented as follows.

- Supervised learning
  - if discrete → classification
  - if continuous → regression.
- semi-supervised
- Unsupervised.
  - clustering
  - density estimation
  - etc.
- reinforced.

Although all these learning paradigms are different there are a lot of common foundations between all of them.

Scanned by CamScanner

Polynomial Curve Fitting

data generated synthetically $\sin(2\pi x)$

$x \equiv (x_1, \ldots, x_N)^T$

corresponding values

$t = (t_1, \ldots, t_N)^T$

for now we will stick to $N = 10$ data points.

$x_n \in [0, 1]$ and t can be obtained by computing $\sin(2\pi x)$ and then we add a small level of noise having Gaussian Distribution.

By generating the data in this way, we are capturing a property of many real world datasets. i.e, There is an underlying regularity but individual observations are carrupted by random noise.

The goal of the learning algorithm is to identify the underlying $\sin(2\pi x)$.

This is intrinsically a difficult problem because we want to "generalize" from a finite amount of data.

For now, lets proceed with simple curve fitting.

$$y(x, w) = w_0 + w_1 x + w_2 x^2 + \dots w_M x^M$$

$$= \sum_{j=0}^{M} w_j x^j$$

where M is the order of polynomial.

The polynomial coefficients $w_0, \dots w_M$ are collectively ~~known~~ ~~as~~ denoted as w.

Note - $y(x, w)$ is not linear for x, but linear for w.

We would also need a mechanism to measure how far our prediction is from ground truth. This can be done via Error function or loss function.

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2$$

The '$\frac{1}{2}$' is just included for mathematical convinience.

Now, the loss function will be zero iff $y(x, w)$ passes exactly thro' the training data point.

Hence, we need the value of 'w' for which $E(w)$ can be closest to zero. (as small as possible).

The error function is in quadratic form. Hence the minimization of $E(w)$ has a unique solution. which we will denote by w^*.

i.e we are searching for $y(x, w^*)$

But we also need to choose the order 'M' of the polynomial. (model selection).

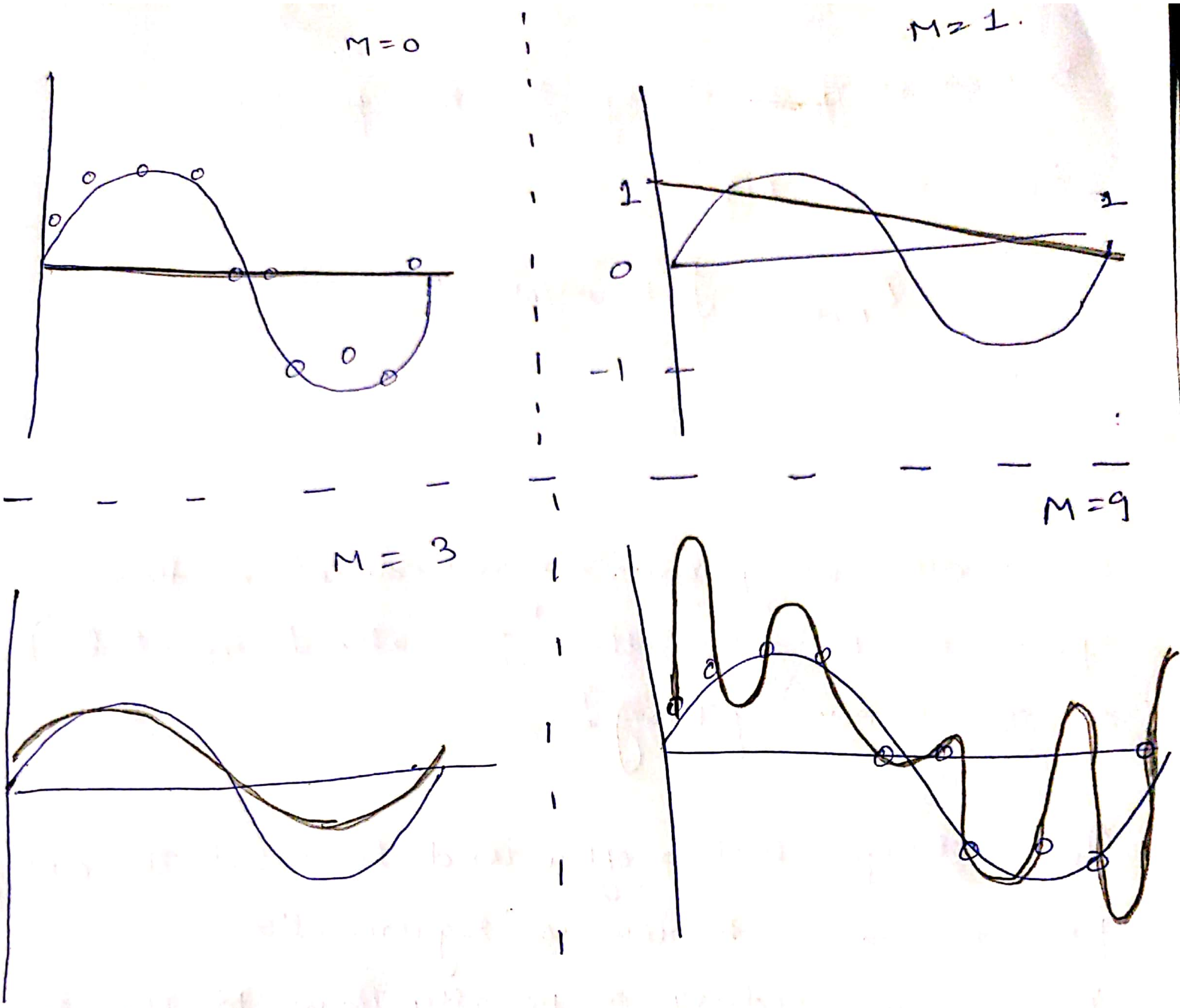
For this problem we choose $M = 0, 1, 3, 9$

$M = 0$ is constant

$M = 1$ is first order polynomial.

$M = 3$ is third order

$M = 9$ is much higher ordered polynomial.



Plots of various orders of M.

poly

which order of polynomial is best fit?

But what if we change the loss function?

How about RMS?

$$E_{RMS} = \sqrt{2E(w^*)/N}$$

---

what other way can we introduce in this approach to get better generalized model? or avoid over-fitting?

One technique that is often used to control the over-fitting phenomenon is known as "regularization".
It involves adding a penalty term to the loss function

$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\|w\|^2$$

——x——