CMSC 491/691: Computer Vision **Fall 2023**

Lecture 7: 2024-02-028 Machine Learning in Computer Vision I

Lecturer: Tejas Gokhale *Scribe: Kyle Manspeaker*

The notes below are an example of what I am expecting. They were taken from a random graduate class. They illustrate some uses of various LaTeX macros. Take a look at this and imitate.

7.1 Machine Learning Basics

In the last lecture, we introduced the use of Machine Learning in computer vision. Beginning with a general overview of ML by observing a linear expression example which could comfortably model the points with a standard line of best fit described with the equation:

$$f_\theta(x) = \theta_0 + \theta_1 x \tag{7.1}$$

We observed how a line of this nature could be acceptable in some circumstances, however; many scenarios require a more complex shape. A line containing one-to-many curve like features, also known as a parabola.

With the main goals of model training being to construct a series of weights such that error of the best fit line is minimized. Increasing the number of weight used can help achieve this. For example, if we increase the degrees of the polynomial, we are able to fit the model to represent each point in the training data, enabling the error to reach zero. Using the simple summation below, increasing the value of k enables the model to better fit the training data.

$$f_\theta(x) = \sum_{k=0}^{N} \theta_k x^k \tag{7.2}$$

Although reducing the error of the training data is part of the model creation process, we observed the phenomenon known as over fitting. Over fitting leads to a high testing error, where the model can perfectly represent the training data, but in turn reduces its performance when introduced to the testing data. Combating over fitting is a fine dance of tune weights to work for both the training and the test data. The balance between under fitting (high training error) and over fitting cannot be derived from an exact algorithm, rather careful consideration when making and tuning a model are necessary.

The model is only as good as the data. When working with a limited selection of data available, one needs to alleviate potential biases in the training and test selection. Introducing the concept of IID or Independent Identically Distributed training and test data. The sets used to train and test the model need to be different while maintaining similarities in type and distribution.

Incomplete data sets is expected as not all scenarios can be represented in one collection. There will always be new/ unseen data that could be passed into a model, hence the model should makes the best out of the data is available

7.2 Machine Learning with Images

For the most part, all of the previous points are general concepts behind Machine Learning, where in this section, we will explore how it relates into Computer Vision.

Digital images have additional layers of complexity when compared to the previous example of points on a 2-D coordinate plane. In theory, the concepts behind ML on images works the exact same, however in practice, the processes to operate on images is more complex to setup.

Using an example of an object detection/ image classifier, lets look at how an image will be interpreted and used as inputs to ML function. In class, we used a 32X32 pixel image with 3 color channels.

$$32 * 32 * 3 = 3072 \tag{7.3}$$

Matrix Representation:

$$\begin{bmatrix} R_{1,1} & G_{1,1} & B_{1,1} & ... & R_{1,32} & G_{1,32} & B_{1,32} \\ R_{2,1} & G_{2,1} & B_{2,1} & ... & R_{2,32} & G_{2,32} & B_{2,32} \\ ... & & & & & & \\ R_{32,1} & G_{32,1} & B_{32,1} & ... & R_{32,32} & G_{32,32} & B_{32,32} \end{bmatrix} \tag{7.4}$$

Say we wanted a classifier that outputs a *10x1* matrix that indicates they type of animal displayed in a image. At a simple level, a model can be represented by the equation $y = wx + b$. As it pertains to our example, each variable is its own matrix.

- y is the output classification represented by the *10x1* matrix
- w is the set of weights that is calculated when creating the model and is a *10x3072* matrix
- x is the images itself, stretched into a 1-D matrix of size *3072x1*
- b is the bias, or additional tuner variable that can be used to adjust the final value of y

When constructing the model, the weights and biases are tuned to minimize the sum of the errors between the classification and expected output, represented in the equation below.

$$\sum_{x*y} (f(x) - y)^2 \tag{7.5}$$

(The same process works with larger matrices as well)

7.3 Linear Classifier Limitations to Neural Networks

In class we looked at the problem with Linear Classifiers using the example of the XOR operations ($\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$), with value 1 resulting from only one of the inputs being 1, otherwise 0. When attempting to draw a single line to separate the $0's$ and $1's$ of the possible outputs, we begin to see the problem. A single line (linear or parabolic) cannot be used to truly separate them in to different classes. One options (more visible when drawn out) takes a asymptotic approach, however for these notes I am unable to neatly provide this example.

Smooth Transition for lack of written notes here

Computation behind Neural Networks. Series of weights for each value in the output matrix. For each value in x there exists a weight that will be used when computing the value for the corresponding value in y.

$$y_j = \sum_{i=0}^{n} w_{ij} x_i + b \tag{7.6}$$

- y is the output matrix
- n is the size of the output matrix y
- w_j is the set of weights that corresponds to the jth output in y
- x is the input value (1-Dimensional)
- b is the bias, or additional tuner variable that can be used to tune that specific y_j

Adding Layers to the NN requires an additional activation function that is non-linear in nature. Due to the properties of linear computations, if all the layers were computed one after the other, the concept of layers no longer exists as all weights/ values can be wrapped into a single layer. Hence the introduction of non-linearity which enables each layer to make distinctive decisions from the previous ones based only on its inputs. As an example the following function is a popular non-linear activation function:

$$g(y) = \sigma(y) = \frac{1}{1 + e^{-y}} \tag{7.7}$$

Choosing to take the bullet list approach for this last sections as my notes from class began to lack some in this department when the graphs and video were displayed.

- Touched upon the sin wave of enthusiasm pertaining to Perceptrons and the stepping stones leading to ML as we know it today (specifically relating to neural networks). We saw as new methods/ ideas were being introduces, feasibility of computational power and potential uses decreased the overall excitement in the world of ML, which each new advancement helping to oscillate the enthusiasm graph in the positive direction. Main advancements include
perceptrons $->$ Convolutional Neural Networks $->$ ImageNet $->$ Alex Net.
- Highlighted Neural Networks as being a series of layers which consist of linear function of combinations of weights followed by non-linear activation functions.
- Connectivity Patterns
  - Fully Connected/ all weights are greater than zero
  - Sparse/ Locally Connected / some weights may be zero