CMSC 491/691: Computer Vision **Spring 2024**

Lecture 1: 2024-03-04 ML for CV (NN)

Lecturer: Tejas Gokhale *Scribe: Faisal Rasheed Khan*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

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.

1.1 Recap of Last Lecture

In the last lecture, we have seen the field of Machine Learning for Computer Vision. The motivation to the need of Machine Learning in the Computer Vision is due to the Image Classification. The image is classified based on the features.
Challenges to the image classification are viewpoint variations, illumination, background clutter, occlusion, pose and deformation, inter-class variation, illusions. The task of image classification is handled by Machine Learning by training the model with lots of data. ImageNet contains all the images data.
This is how Machine Learning works:

i. Lots of image data

ii. Train the model with these data

iii. Evaluate the model on the unseen images

For the linearly distributed data, the equation which fits the line is

$$f_\theta(x) = \theta_1 \cdot x + \theta_0 \text{ where } \theta_1 \text{ and } \theta_0 \text{ are the learning parameters.} \tag{1.1}$$

The loss function is defined by:

$$L(\hat{y}, y) = (\hat{y} - y)^2; \text{ where } \hat{y} = f_\theta(x) \tag{1.2}$$

The objective is to minimize the learning problem with respect to θ and is given by:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{n} \left(f_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{1.3}$$

To tackle a vision based model need to have the questions like how to represent inputs and outputs, whats the objective, hypothesis space, how to optimize, whats the training data. For the image classification task, the inputs labels are represented with numbers for the same category of images or one-hot representations. The loss 0-1 or cross-entropy loss functions can be used. The probabilistic model is used for the image classification tasks. The softmax regression activation function can be used where the class labels are selected based on the highest probability of the class.

1.2 Neural Networks

In the last lecture we have seen the linearly distributed data, the solution which fits the data is a line equation eq(1.1). When the data is not linearly distributed, we need to have more complexity in the equation which is polynomial which is given by:

$$f_\theta(x) = \sum_{k=0}^{K} \theta_k \cdot x^k; \text{ where } \theta_k \text{ are the learning parameters.} \tag{1.4}$$

The higher the value of k, the model will result in fitting every point which results in overfitting of the model. The low value of k for the polynomial function will underfit the model.

1.2.1 Parametric Approach: Linear Classifier

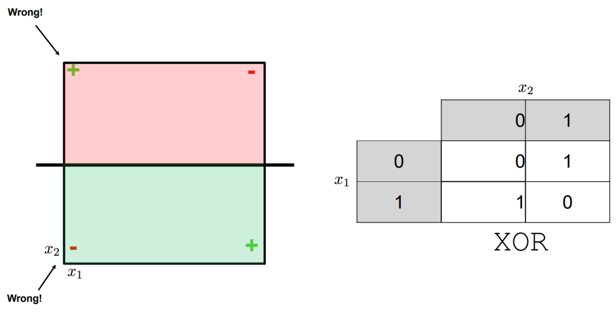
The parametric equation for a Linear Classifier is given by:

$$f(x, W) = W \cdot x + b; \text{ where } W \text{ and } b \text{ are the learning parameters.} x \text{ -Input image with 3 channels} \tag{1.5}$$

x - size of the image w x h x 3
W - Weight matrix rows are number of class labels and columns are size of image
b - rows are number of class labels and 1 column
f(x,W) - rows are number of class labels and 1 column

1.2.2 Limitations to Linear Classifier

If the decision boundary is nicely separable then linear classifier separates the data. But if the decision boundary is not linearly separable, we need to have non-linear decision boundary. The example for the non-linear boundary is the XOR function where linear classifier fails.



		0	1
x_1	0	0	1
	1	1	0

Figure 1.1: Limitations of Non-Linear Classifier

1.2.3 Improvements over the years

The basic model is perceptron which uses the parametric approach for a linear classifier where there is single layer with activation function to the output. There can be more number of layers which is a neural network and based on these there are LeCun's Convolutional Neural Network. LeCun's Neural Network was used to classify the digit images. After that ImageNet dataset has been released which contains millions of images of 1000 categories. With the release of ImageNet, AlexNet architecture released trained on ImageNet data to classify the images and its a 8 layer architecture. Today, we have 100+ layers architecture

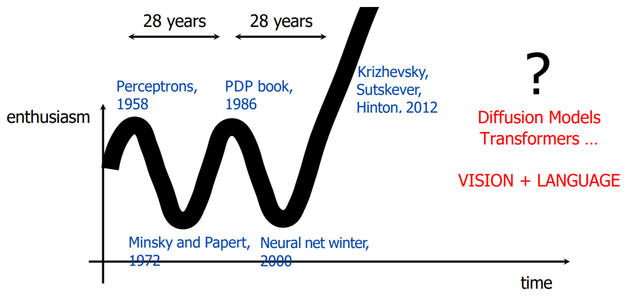
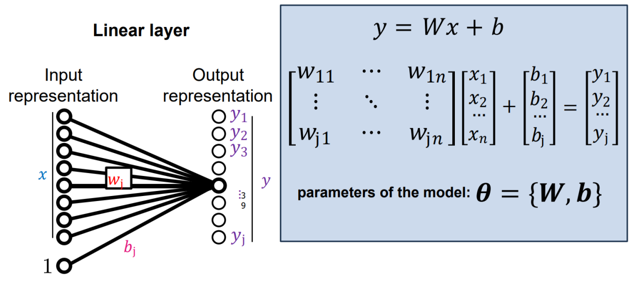


Figure 1.2: Advances over the years

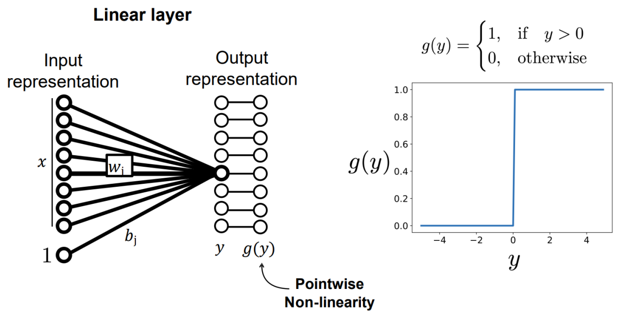
1.2.4 Neural Linear Layer Perceptron



$$\begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{j1} & \cdots & w_{jn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_j \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_j \end{bmatrix}$$

Figure 1.3: Perceptron Linear Layer with parameter representation

We can now transform our input representation vector into some output representation vector using a bunch of linear combinations of the input and then those outputs can be used as inputs and so on to solve the problem. But the above approach cant solve the XOR problem, we can solve it with a single linear layer but with a non-linear activation function. There will be many non-linearity functions where gradients are



$$g(y) = \begin{cases} 1, & \text{if} \quad y > 0 \\ 0, & \text{otherwise} \end{cases}$$

Figure 1.4: Perceptron Linear Layer with Non-Linear Activation functions

not defined, need to handle that. The other non-linear activation functions are Sigmoid, ReLU, Leaky RelU. Sigmoid sets the function between [0,1] range.
In ReLU, the derivative is not defined at exactly 0, but it is handled using subderivative. When the input is exactly 0, the gradient is considered to be 0 for simplicity. Therefore, the function increases linearly for positive values and stays at 0 for negative values.
The ReLU part(subderivative) is handled in the Leaky ReLU and is completely differentiable at any point.

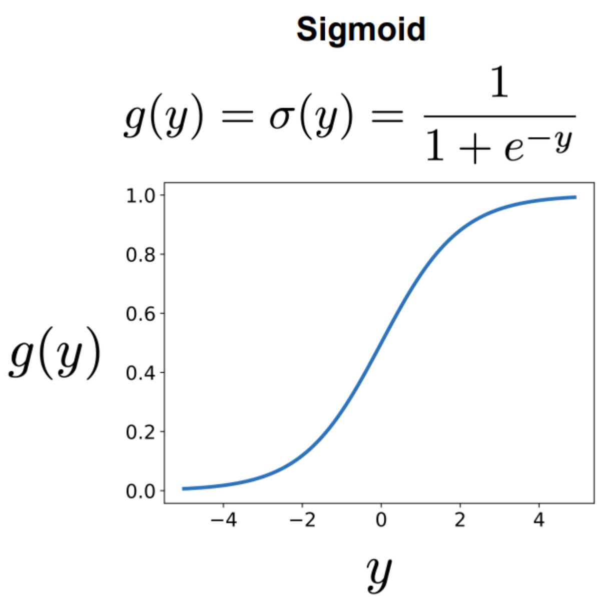


Figure 1.5: Non-Linear Sigmoid Activation function

1.2.5 Deep Nets

Stacking many layers in between input and output layer with a non-linear activation function attached with a linear layer is the deep nets. The single layer approach is applied to the multiple layer individually and every layer has parameters with a non-linear activation function.

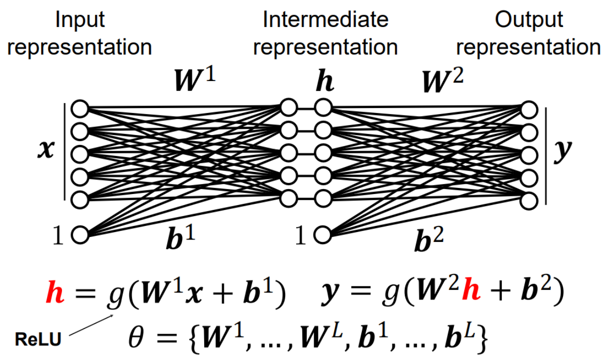

Input representation
Intermediate representation
Output representation
$\mathbf{W}^1$
$\mathbf{h}$
$\mathbf{W}^2$
$\mathbf{x}$
$\mathbf{y}$
1
$\mathbf{b}^1$
1
$\mathbf{b}^2$
$\mathbf{h} = g(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1)$
$\mathbf{y} = g(\mathbf{W}^2\mathbf{h} + \mathbf{b}^2)$
ReLU
$\theta = \{\mathbf{W}^1, \dots, \mathbf{W}^L, \mathbf{b}^1, \dots, \mathbf{b}^L\}$

Figure 1.6: Stacking Layers

The reason because of the more layers are due to the individual features learning at each step. Consider image classification of the dog. There will be different features of dog: low level features like edges, texture etc in one layer, mid level features like shape etc. in another layer and high level features like paw, fur etc in another layer. All the above layers constitute as deep nets to classify image as a dog.
The parameter learnings will be at each step because each layer has parameters associated to that feature and we compute loss at those layers and update the model.

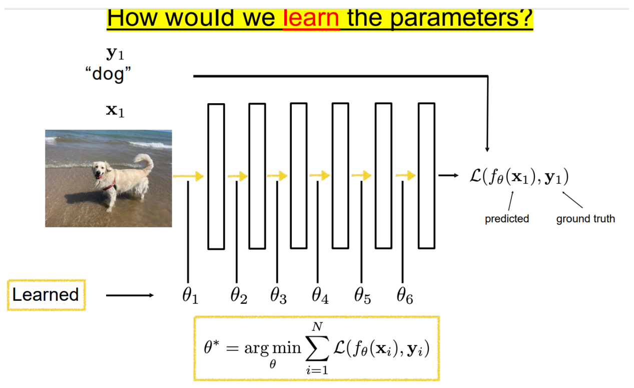

How would we learn the parameters?
$\mathbf{y}_1$
“dog”
$\mathbf{x}_1$
$\mathcal{L}(f_\theta(\mathbf{x}_1), \mathbf{y}_1)$
predicted
ground truth
Learned
θ_1 θ_2 θ_3 θ_4 θ_5 θ_6
$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)$

Figure 1.7: Parameter Learning for Deep Nets