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

Lecture 9: 2024-03-4 Neural Network Optimization for Computer Vision III

Lecturer: Tejas Gokhale *Scribe: Amanjot Singh*

9.1 Recap of Last Lecture

In the last lecture, we studied about neural networks for computer vision.

What is linear regression? Linear regression analysis is used to predict the value of a variable based on the value of another variable.

What is polynomial regression? In polynomial regression, an nth-degree polynomial is used to represent the relationship between the independent variable (or variables) and the dependent variable. Unlike linear regression, it captures between non-linear relationships between the input features and the output.

What is parametric approach?

A linear function with a fixed set of parameters is used to model the relationship between the input features and the class labels in a parametric approach to linear classification.

What is perceptron?

A neural network with only one layer that can classify things in binary is called a perceptron. It generates a single binary output from multiple binary inputs (0 or 1), each with a corresponding weight.

After then we studied about LeCun convolutional neural networks, imageNet and AlexNet.

We studied about compuatation in neural network it is combining the input features with matching weights and biases. Below is the equation-

$$y_j = \sum_i w_{ij} x_i + b_j \tag{9.1}$$

There was a limitation for the linear model that because of its linear decision boundary limitation, a single-layer linear neural network is unable to solve the XOR (exclusive OR) problem. So the activation function introduced non-linearty in the neural network. We studied about the following activation function-

Step function

Sigmoid function

ReLu

Leaky ReLu

What are stacking Layers?

In neural networks, stacking layers refers to the sequential arrangement of several layers of neurons. Each layer uses bias, activation functions, and weighted connections to process inputs.

What are DeepNet?

Deep Nets are multi-layered, complex architectures that facilitate learning from data and hierarchical feature extraction

9.2 Online Learning

We watched the video of the Professor Shree Nayar who is faculty in the Computer Science Department, School of Engineering and Applied Sciences, Columbia University. We learnt about gradient descent is a popular optimization technique for changing the weights and biases of neural networks in order to minimize the cost function is gradient descent. In order to minimize costs, it iteratively updates parameters such as weights and biases. The Taylor series is used to measure the change in function and approximate the change in cost related to parameter changes. Gradient descent guarantees descent to the minimum by orienting the parameter change in the opposite direction of the gradient of the cost function. The rate of descent is determined by the learning rate, low rates cause slower convergence, while high rates increase the chance of overshooting. The idea behind gradient descent is similar to navigating a hilly terrain in the direction of its lowest point. Gradient descent is still useful even for high-dimensional problems, even though the computational complexity rises with dimensionality.

9.3 Neural Networks for Computer Vision

9.3.1 Gradient descent

Gradient descent is an optimization algorithm that uses iterative model parameter adjustments to minimize a machine learning model's loss function. It operates by moving in the direction of the loss function's steepest descent in relation to the parameters. Until the algorithm converges to a minimum, this process is repeated.

$$w = w - \Delta w \tag{9.2}$$

w represents the current value of the model parameter.
Δw represents the change in the parameter that needs to be applied.
$w - \Delta w$ updates the parameter w by subtracting the change Δw from its current value.

9.3.2 Backpropogation

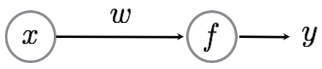
An essential algorithm for neural network training in machine learning is backpropagation. In order to produce predictions, the network processes incoming data in a forward pass. Then, using a loss function, the error between these predictions and the actual outputs is computed. This error propagates back through the network in the backward pass, calculating the gradients of the loss function with respect to the weights. With the goal of minimizing the loss, these gradients are used to update the weights using an optimization algorithm such as gradient descent. Until the model converges to an ideal solution, this process is repeated. Neural networks can learn from data by modifying their weights to increase prediction accuracy through backpropagation.

9.3.3 Perceptron

The most basic type of neural network, a perceptron functions as a foundational model for tasks involving binary classification. It consists of activation function, weights assigned to each input, and input nodes. The fundamental function of a perceptron can be expressed as follows:

$$\hat{y} = wx \tag{9.3}$$

where w is the weight connected to the input x and $\hat{y}$ denotes the anticipated result. This formula represents the linear decision-making process of the perceptron: the weighted sum is obtained by multiplying the input, x, by its weight, w. We also need to modify w such that $\hat{y}$ equals y.



Predict: We took random sample (x_i, y_i)

Forward Pass: Using the weighted sum of inputs as input, apply the activation function of the perceptron to determine the expected outputs.

$$\hat{y} = wx_i \tag{9.4}$$

Compute Loss: By contrasting the expected outputs with the actual target values, compute the loss function, which is usually mean squared error (MSE).

$$L_i = \frac{1}{2}(y_i - \hat{y}_i)^2 \tag{9.5}$$

Compute Gradient: Find each parameter's gradient in relation to the loss function.

$$\frac{\partial L_i}{\partial w} = -(y_i - \hat{y}_i) \cdot x_i \tag{9.6}$$

Gradient Descent Update: Using gradient descent, update the parameters by scaling them according to the learning rate and adjusting them in the opposite direction of the gradient.

$$w = w - \Delta w \tag{9.7}$$

9.3.4 Multi-layer perceptron

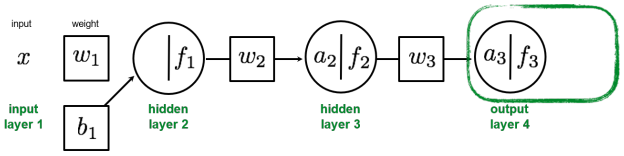
An artificial neural network consisting of several layers of nodes (neurons), each fully connected to the next, is called a multilayer perceptron (MLP). Every node in the next layer is connected to every other layer's node, but connections don't exist within layers. Below is the equation-

$$f(x) = f_3\left(W_3 \cdot f_2\left(W_2 \cdot f_1\left(W_1 \cdot x + b_1\right)\right)\right) \tag{9.8}$$

where:

- **x** is the input vector.

- $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ are weights for the connections between the input and the first hidden layer, the first and the second hidden layer, and the second hidden layer and the output layer.
- $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$ represent the bias vectors for the first, second, and output layers, respectively.
- f_1, f_2 and f_3 denote activation functions applied element-wise to the weighted sums in the first, second, and output layers, respectively.



Observing the model, many values are unknown so we need to calculate using gradient descent algorithm

Sample: Using gradient descent, every sample in the batch is processed. (x_i, y_i)

Forward Pass: The network processes the incoming data and computes the output of each layer as well as the ultimate prediction.

$$\hat{y}_i = f_{\mathrm{MLP}}(x_i; \theta) \tag{9.9}$$

Compute Loss: To measure the difference, the loss function is assessed using the actual target values and the predicted output.

$$L_i = \frac{1}{2}(y_i - \hat{y}_i)^2 \tag{9.10}$$

Back Propagation: Backpropagation is used to compute the gradients of the loss function with respect to each parameter.

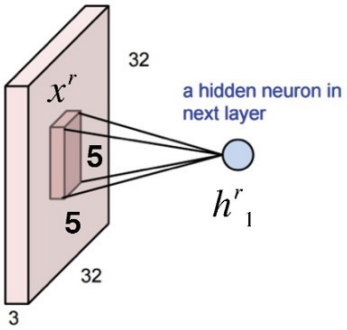
$$\frac{\partial L_i}{\partial \theta} \tag{9.11}$$

Gradient Update: The computed gradients are used to update the parameters. A learning rate is used to scale the update in the direction that is opposite to the gradient.

$$w = w - \Delta w \tag{9.12}$$

9.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is a kind of deep learning architecture that is best suited for visual data like images. They mimic human visual system and automatically extract meaningful patterns and features from images. The CNNs are made up of layers of interconnected neurons, which are responsible for processing local regions of the input image. This happens through convolutional operations, which comprise moving small filters across the input. Following convolutional layers, pooling layers bring about dimensional reduction of the feature spaces but keep the useful information. These layers allow the network to learn hierarchical representation of the input data, and this enables it to capture features at different levels of details. 3D activation typically refers to the activation maps produced by convolutional layers in Convolutional Neural Networks (CNNs) when processing three-dimensional data, such as volumetric images or videos. These activation maps represent the output of the convolutional layer after applying filters to the



input volume. 3D convolutional layers are advantageous over 1D layers when dealing with volumetric data because they capture spatial and temporal information simultaneously.

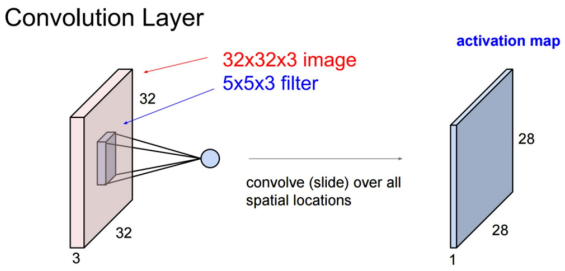
The above image is a 3*32*32 image meaning (depth=3, height=32, width=32 and having RGB channels)

Consider the region of input x^r with output neuron h^r. Then, the output is:

$$h^r = \sum_{i,j,k} x^r_{ijk} \cdot W_{ijk} + b \tag{9.13}$$

In this equation:

- h^r represents the output.
- $\sum_{i,j,k}$ denotes the sum over the 3 axes i, j, and k.
- x^r_{ijk} represents the input.
- W_{ijk} represents the filter weights at position i, j, k.
- b represents the bias term.



The above image is a 3*32*32 image meaning (depth=3, height=32, width=32 and having RGB channels) and has a 5*5*3 filter when convolved over all the spatial locations we got the output with dimension is 28*28*1.

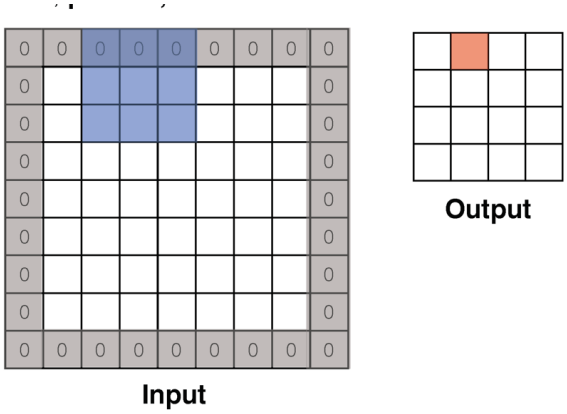


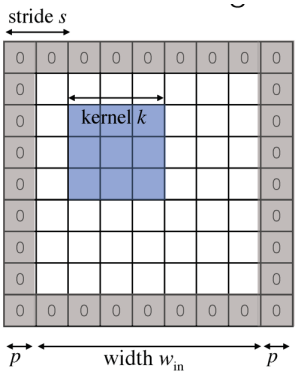
Figure 9.1: This figure is showing padding = 1 and stride= 2

9.4.1 Striding

It moves the filter or kernel across the input data by a predetermined number of pixels or units at a time. It enables adjustment of the output feature maps' spatial dimensions. In the figure 9.1 we can show the input has stride =2

9.4.2 Padding

The padding technique to add extra pixels or values to the input data's border. This ensures that information at the input's edges is sufficiently captured and processed by maintaining the spatial dimensions. In the figure 9.1 we can show the input has zero padding (pad =1).



$$w_{\text{out}} = \frac{w_{\text{in}} + 2p - k}{s} + 1 \tag{9.14}$$

where:

- w_{out} is the output width,

- w_{in} is the input width,
- p is the padding i.e 1,
- s is the stride i.e 2,
- k is the kernel width i.e 3.

9.4.3 Pooling

Pooling is a downsampling method that convolutional neural networks frequently employ to minimize the spatial dimensions of feature maps

9.4.4 Maxpooling

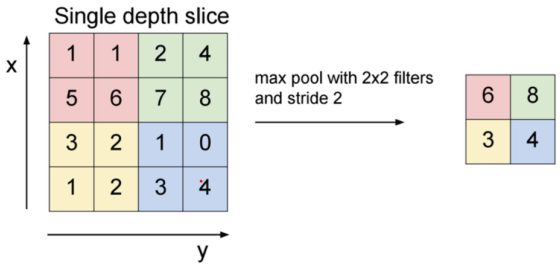


Figure 9.2: Max Pooling

CNNs employ the downsampling method known as max pooling to minimize the dimensionality of feature maps while maintaining crucial data. It saves the index of the highest value in each pooling window during the forward pass. Only the neuron with the maximum value receives gradients backpropagated in the backward pass; all other neurons within the window receive zero gradient. In addition to facilitating effective computation and preventing overfitting, this guarantees the preservation of important features. Consequently, by preserving important spatial information, max pooling aids in the learning of hierarchical representations.