Indian Institute of Technology Tirupati
Deep Learning (CS5223): Mid Semester

Date: 7th October, 2021 (10 - 11:30AM)
Maximum Marks: 40
Instructions: Answer all the questions.

---

1. Consider a Neural network applied to an industrial data with 7 channels and 16×16 spatial dimensions. Compute the number of parameters and products in the first layer for each of the following configurations. (You may leave the expression as a product without finding the actual number, e.g. $5 \cdot 4^2 \cdot 3^2$)[2+2=4 Marks]

   (a) Use a convolution layer with 8 feature maps and filters of size 5×5 (use default values for stride and padding)

   (b) Use a fully-connected layer where the number of input and output units are kept same as part (a).

   Solution
   (a) Feature map size after the convolution is $D \times (W - w + 1) \times (H - h + 1)$, where D is the number of filters, W, H are width and height of input, and w, h are width and height of filters.
   $\rightarrow$ Number of units in the layer (or, size of the output) $= 8 \times (16 - 5 + 1 = 12) \times (16 - 5 + 1 = 12) = 8 \times 12 \times 12$.
   Number of parameters $= 8 \cdot (7 \cdot 5^2 + 1)$
   Number of products $= 8 \cdot 12^2 \cdot 7 \times 5^2$

   (b) Size of the input $= 7 \times 16 \times 16 = 7 \cdot 16^2$
   Number of units in the layer (or, size of the first layer's output) $= 8 \times 12 \times 12 = 8 \times 12^2$ [computed in part (a)]
   If we use a fully connected layer, number of parameters would be $8 \cdot 12^2 \times (7 \cdot 16^2 + 1)$
   Number of products$= 8 \cdot 12^2 \times 7 \cdot 16^2$

2. What are different types of pooling layers that are commonly used in CNNs. Describe how the pooling operation happens. State one advantage and one disadvantage of using pooling layer in CNNs. [1+1+2=4 Marks]

   Solution
   General pooling operations performed in CNNs are max pooling and average (or mean) pooling. Different other pooling operations are also possible (such as min pooling, weighted average pooling, etc.), but or not very common.

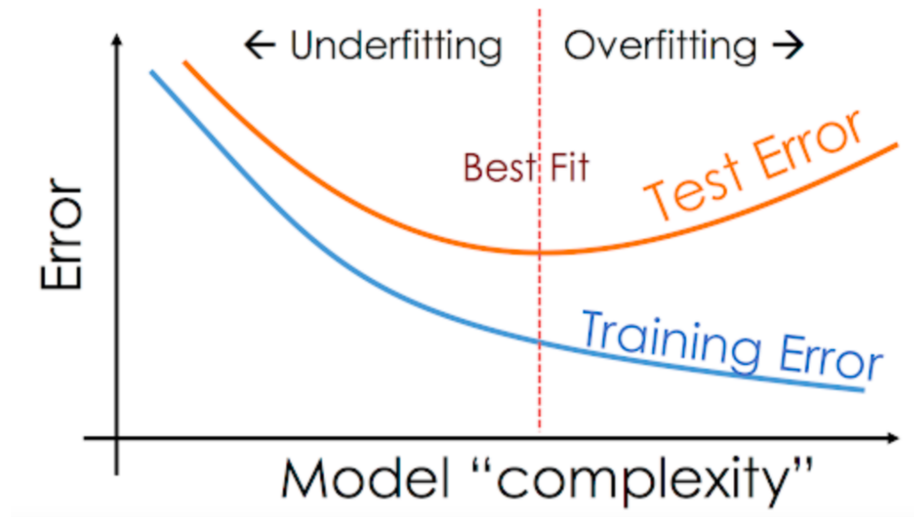
   Pooling operation is performed by operating the kernel over the input from top left to bottom right, generally using a non-overlapping stride. Important difference from convolution operation is that pooling happens within each channel without interacting across channels. This is because the aim of pooling is to reduce the spatial extent of the feature maps.

   Advantage of using pooling is that the spatial dimensions of the input is reduced progressively so that the number of required parameters for processing is reduced, thereby making the model less prone to overfitting.

   Disadvantage of performing pooling is that information is lost during the operation. For instance, during the maxpooling with $2 times 2$ kernel, 3 values from the receptive field are lost but only the maximum value will be retained.

3. Plot the learning curves where you depict the (average) training and validation losses against the model complexity. Indicate the overfitting scenario with clear labeling in the figure. [2 Marks]

Solution



4. Write pseudo code for the following: [2+2+2=6 Marks]
   (a) Stochastic Gradient Descent for parameter update
   (b) Full batch Gradient Descent for parameter update
   (c) Mini-batch Gradient Descent for parameter update

   Solution
   (a) Stochastic Gradient Descent

```
for i in range(nb_epochs):
→np.random.shuffle(data)
→for example in data:
→→params_grad = evaluate_gradient(loss_function, example, params)
→→params = params - learning_rate * params_grad
```

   (b) Full-batch Gradient Descent

```
for i in range(nb_epochs):
→params_grad = evaluate_gradient(loss_function, data, params)
→params = params - learning_rate * params_grad
```

   (c) Mini-batch Gradient Descent

```
for i in range(nb_epochs):
→ np.random.shuffle(data)
→ for batch in get_batches(data, batch_size = 50):
→→ params_grad = evaluate_gradient(loss_function, batch, params)
→→ params = params - learning_rate * params_grad
```

5. Mention at least two ways for minimizing overfitting in neural networks. Describe clearly why they work. [2+2=4 Marks]

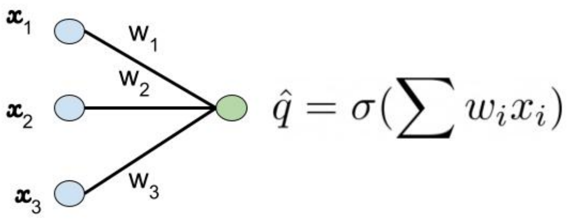
   Solution

   - More training data can reduce the overfitting effect. Since more data can help improve (i.e., reduce) the variance component in the Bias-Variance decomposition. It is suggested to collect more data if possible before we start training the neural network. In some cases it may not be possible to collect more data, then we have to resort to other techniques.

- Data preprocessing or feature design is a way to reduce the model capacity without affecting (or improving) the bias. In other words, we can use a lower capacity model with the processed features so that the model is less prone to overfitting.
- Reducing the number of feasible models (or size of hypothesis set) is another way to reduce the overfitting. This is often known as regularizing.

6. Consider a one-layer neural network with 3 input units and 1 output unit. All the units have the sigmoid nonlinearity ($\sigma(x) = \frac{1}{1+e^{-x}}$). Assume bias terms are not there for simplicity. Loss used for optimizing the network is the cross-entropy loss. Derive the equations for a single step of weight update. Clearly identify all the symbols used in the derivation. [6 marks]

   Solution
   Since there are 3 i/p units, input (x) is $3D$ vector. Also, since there is one o/p unit, it is a binary classification problem. Note that it is a one-layer neural network, now that means there is only one neuron in the whole network (because i/p layer is a dummy layer and only accepts the input). We can assume that classes are $\{0, 1\}$ and the predicted probability ($\hat{q}$) is for class 1, hence the other class probability can be obtained by subtracting it from 1.



$$\begin{aligned} \sigma(x) &= \frac{1}{1+e^{-x}} \\ \frac{\partial \sigma(x)}{\partial x} &= \sigma(x)(1-\sigma(x)) \\ \frac{\partial log(\sigma(x))}{\partial x} &= \sigma(-x) \end{aligned}$$

loss $l = p \cdot log(\hat{q}) + (1-p) \cdot log(1-\hat{q})$, where p is the groundtruth probability for class 1.

$$\begin{aligned} \frac{\partial l}{\partial \hat{q}} &= \frac{p}{\hat{q}} - \frac{(1-p)}{(1-\hat{q})} \\ \frac{\partial l}{\partial w_i} &= \frac{\partial l}{\partial \hat{q}} \frac{\partial \hat{q}}{\partial w_i} \\ \frac{\partial \hat{q}}{\partial w_i} &= \sigma(\sum w_i x_i)[1-\sigma(\sum w_i x_i)]x_i \\ w_i^{t+1} &= w_i^t - \eta \frac{\partial l}{\partial \hat{q}} \frac{\partial \hat{q}}{\partial w_i} \end{aligned}$$

7. Consider the binary classification data from Table 1. Perform perceptron training on this data clearly mentioning all the required symbols and steps (until a maximum of 5 epochs). [4 Marks]
   Solution

   First step is to add an extra dimension to input samples (x_i) for absorbing the bias (b) into

sample (x_i)	label (y_i)
(1, 1, 1)	+1
(1, 1, 0)	-1
(1, 0, 1)	-1
(0, 1, 1)	-1

Table 1: Sample binary classification dataset.

the weight vector (w). Hence, x_i and w becomes $4D$ vectors.
Initialize $w_0 = (0, 0, 0, 0)$
First epoch:

t	x_i	y_i	${w_t}^T x_i$	$y_i({w_t}^T x_i)$	$w_{t+1} = w_t + y_i x_i$
1	(1, 1, 1, 1)	+1	0	0	(1, 1, 1, 1)
2	(1, 1, 0, 1)	-1	3	-3	(0, 0, 1, 0)
3	(1, 0, 1, 1)	-1	1	-1	(-1, 0, 0, -1)
4	(0, 1, 1, 1)	-1	-1	1	(-1, 0, 0, -1)

Total number of weight updates = 3 (shown in red).
Epoch 2: $w_4 = (-1, 0, 0, -1)$

t	x_i	y_i	${w_t}^T x_i$	$y_i({w_t}^T x_i)$	$w_{t+1} = w_t + y_i x_i$
5	(1, 1, 1, 1)	+1	-2	-2	(0, 1, 1, 0)
6	(1, 1, 0, 1)	-1	1	-1	(-1, 0, 1, -1)
7	(1, 0, 1, 1)	-1	-1	+1	(-1, 0, 1, -1)
8	(0, 1, 1, 1)	-1	0	0	(-1, -1, 0, -2)

Total number of weight updates = 3.
Epoch 3:$w_8 = (-1, -1, 0, -2)$ Total number of wight updates = 2.

t	x_i	y_i	${w_t}^T x_i$	$y_i({w_t}^T x_i)$	$w_{t+1} = w_t + y_i x_i$
9	(1, 1, 1, 1)	+1	-4	-4	(0, 0, 1, -1)
10	(1, 1, 0, 1)	-1	-1	+1	(0, 0, 1, -1)
11	(1, 0, 1, 1)	-1	0	0	(-1, 0, 0, -2)
12	(0, 1, 1, 1)	-1	-2	+2	(-1, 0, 0, -2)

Epoch:4 $w_{12} = (-1, 0, 0, -2)$ Total number of weight updates = 2.

t	x_i	y_i	${w_t}^T x_i$	$y_i({w_t}^T x_i)$	$w_{t+1} = w_t + y_i x_i$
13	(1, 1, 1, 1)	+1	-3	-3	(0, 1, 1, -1)
14	(1, 1, 0, 1)	-1	0	0	(-1, 0, 1, -2)
15	(1, 0, 1, 1)	-1	-2	2	(-1, 0, 1, -2)
16	(0, 1, 1, 1)	-1	-1	1	(-1, 0, 1, -2)

Epoch:5 $w_{16} = (-1, 0, 1, -2)$

t	x_i	y_i	${w_t}^T x_i$	$y_i({w_t}^T x_i)$	$w_{t+1} = w_t + y_i x_i$
17	(1, 1, 1, 1)	+1	-2	-2	(0, 1, 2, -1)
18	(1, 1, 0, 1)	-1	0	0	(-1, 0, 2, -2)
19	(1, 0, 1, 1)	-1	-1	1	(-1, 0, 2, -2)
20	(0, 1, 1, 1)	-1	0	0	(-1, -1, 1, -3)

8. What are hyper-parameters in the context of neural networks. Give at least 5 examples. How are the values for these assigned? [1+1+1=3 Marks]

Solution:
During the design and training of neural networks some properties of the network under construction need to be chosen by the individual developer. For instance, the depth and width of the neural network need to be chosen. Also, in case of CNNs the kernel width, stride, and padding need to be set before training process. Further, during the training also, parameters such as learning rate (η) value, the procedure for reducing the learning rate, etc. need to be appropriately set by the developer. These parameters are known as hyper-parameters. Note that values for these parameters are not learned, instead are chosen heuristically and based on the experience to suit the problem at hand.

9. Contrast convolution layer with fully connected layer. Mention advantages and disadvantages along with the suitable use cases where they are preferred. [1.5+1.5=3 Marks]
Solution

Convolution layer	Fully Connected layer
Shares(or, same) weights for all the neurons in a given layer	Different weights for neurons in the same layer
Less memory footprint for the parameters	More memory footprint for the parameters
Computationally more expensive	Computationally less expensive (one matrix vector product)
preferred when preserving the structure in the data is necessary	preferred when features are available for the task (e.g. classification)

10. Write the important portion of a neural network training code implemented in PyTorch using the autograd machineray. (Give pseudo code with two nested iterations for epochs and iterations; no need to mention the exact loss function, model architecture, optimizer, etc.) [2 Marks]

Solution

```
for i in range(nb_epochs):
→ np.random.shuffle(data)
→ for minibatch in get_batches(data, batch_size = 50):
→→ forward pass the minibatch
→→ Compute the loss
→→ optimizer.zero_grad()
→→ loss.backward()
→→ optimizer.step()
```

11. Autoencoder is a specific neural network architecture configuration. It accepts input, through a series of layers it reduces the input dimension, and from there through another series of layers it reconstructs the input. From the understanding of Machine Learning and neural network concepts, answer the following: [1+1=2 Marks]
(a) Autoencoders are supervised or unsupervised technique? Explain your answer.
(b) Autoencoders fall into which ML task (classification, regression, or density estimation)? Justify your answer.
Solution (a) Since there is no specific labeling on the data (similar to image of a digit and the digit category) autoencoders are generally treated as unsupervised technique. However, the input (or a slightly modified version of it) acts as the target. Also, autoencoders are generally trained using supervised learning techniques. Therefore they are referred to as self-supervised.

(b) Since there is no label and the task is not to predict any target label, autoencoders do not fall into classification. Input (or a slightly modified version) is the target for the learning, therefore it can be seen as a special case of regression. Usually autoencoders are restricted in ways: reconstruct only approximately, and to be able to do that only for input that resembles the training data, etc. Modern autoencoders have generalized the idea of encoding and decoding to stochastic mappings. Therefore sometimes they are treated as a density estimation/understanding task.