Deep Learning-IV

(Hyper-Parameter Tuning; Underfitting and Overfitting in Deep Neural Networks)

DR. JASMEET SINGH,
ASSISTANT PROFESSOR,
CSED, TIET

Parameters in Deep Neural Networks

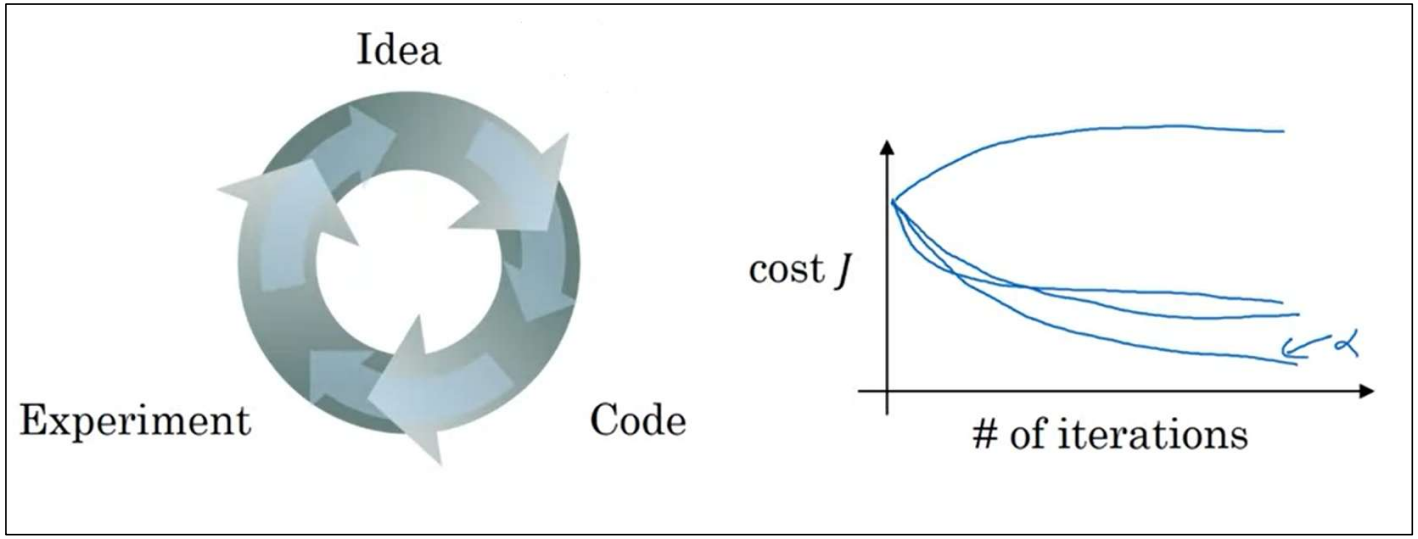
- Parameters in a deep neural network are the variables that needs to be optimized during the training of a neural network.
- Specifically, there are two parameters in neural network:
  - Weight matrix at each layer L i.e. W_L
  - Bias matrix at each layer L i.e. B_L

Hyperparameters in Deep Neural Networks

- Hyperparameters are the parameters that controls the optimization of the parameters (W_L and b_L) in deep neural network.
- The number of hyperparameters are more in deep neural networks are large as compared to machine learning models.
- The hyperparameters in deep neural networks are:
  - Learning Rate (alpha)
  - Number of iterations/epochs
  - Number of hidden layers i.e. L
  - Number of neurons (units) at each layer i.e. n_L
  - Choice of activation function in each layer i.e. $g_L(z_L)$

Hyperparameter Tuning

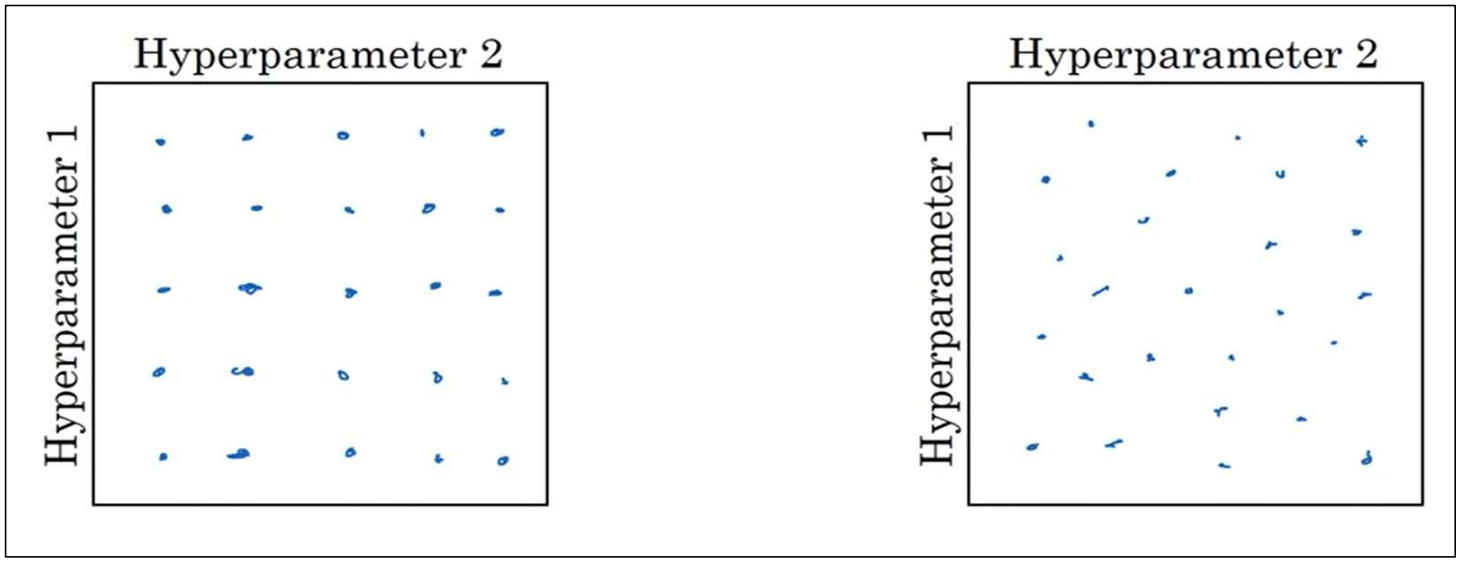
- Hyperparameter Tuning is an empirical process i.e. we need to experiment with different values of hyperparameters and then see which of them performs well.
- For example, in order to decide learning rate, we can plot the graph of the cost with number of iterations, and can choose the learning rate which converges in lesser number of iterations.



Hyperparameter Tuning Process

- **Try Random Process rather than grid of values:**

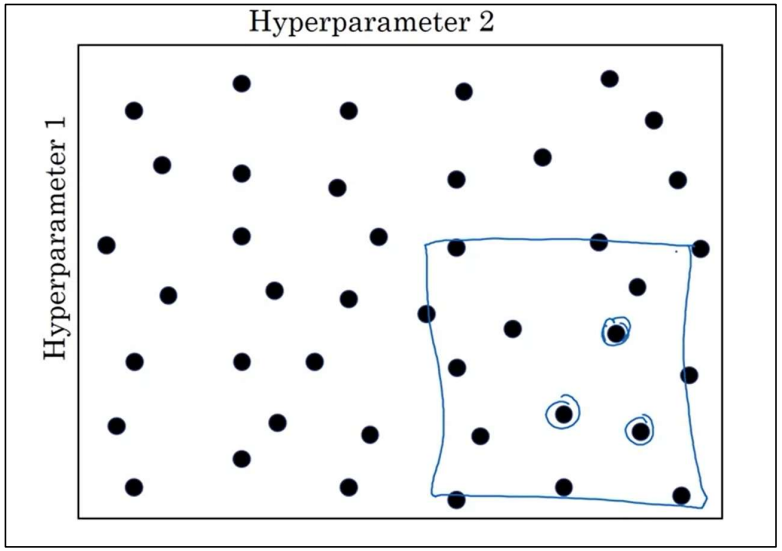
It is always suggested that when the number of hyperparameters are large in number, we must pick a random set of points, rather than a grid.



Hyperparameter Tuning Process (Contd..)

- **Apply coarse to fine search process**

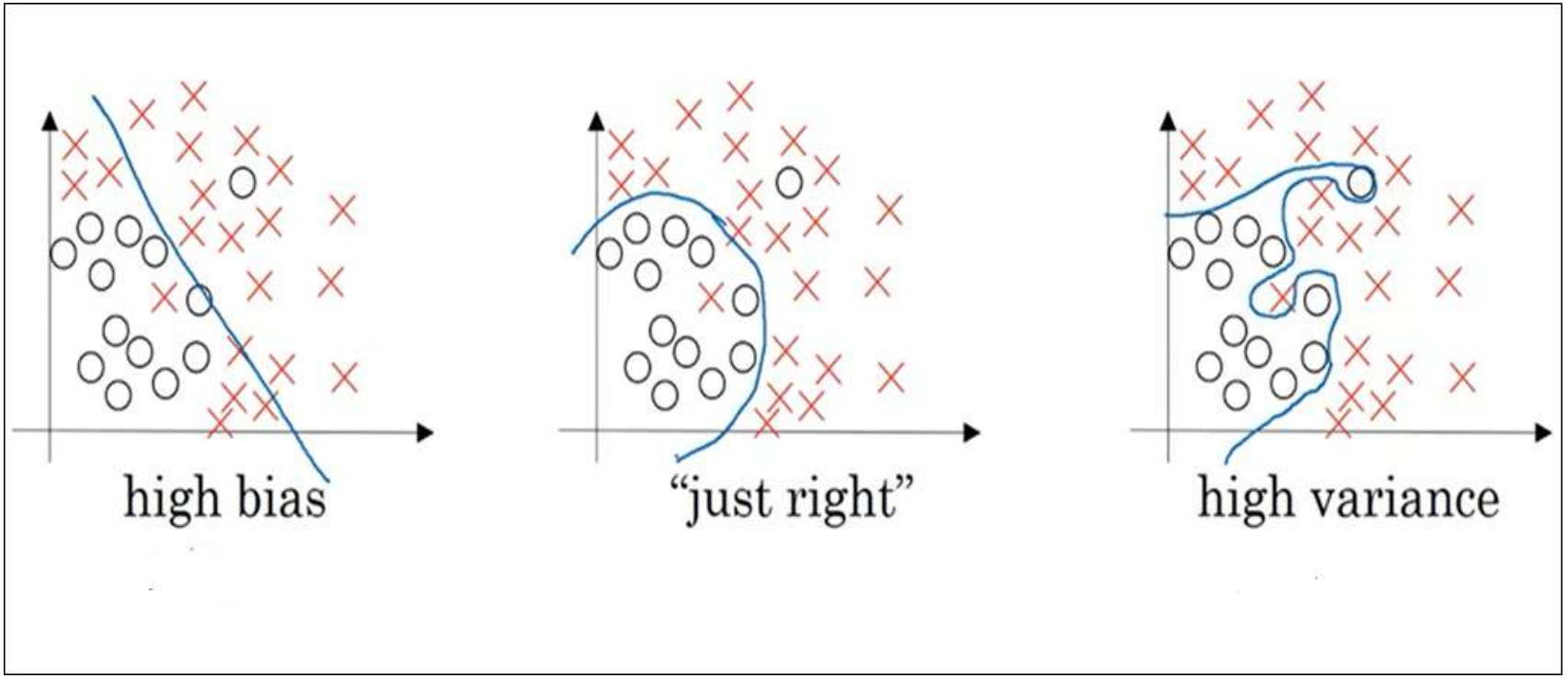
Zoom in to a smaller region of hyperparameter space (which is giving better performance) and then sample more densely in that region.



Hyperparameter Tuning Process (Contd..)

- **Using an appropriate scale to pick hyperparameters**
  - For parameters like number of hidden layers/ number of units in each hidden layer, choose appropriate scale of parameters and randomly choose values from the scale.
  - But for parameters like learning rate, choose parameters on log scale.
  - For exponentially weighted averages parameters, choose parameters on log scale for 1-parameter.
- **Divide your data into training/validation/test set**
  - Train the model on the training data for different values of hyperparameters, and evaluate its performance on the validation set.
  - The best values of hyperparameters (for which validation accuracy is best), must be chosen and finally evaluated on the test set.
  - In deep learning applications working on larger datasets, the ratio of training to validation to test set should not be as high as 60:20:20. It can be set to 98:1:1.
  - Not having a test set might be okay.

Bias and Variance

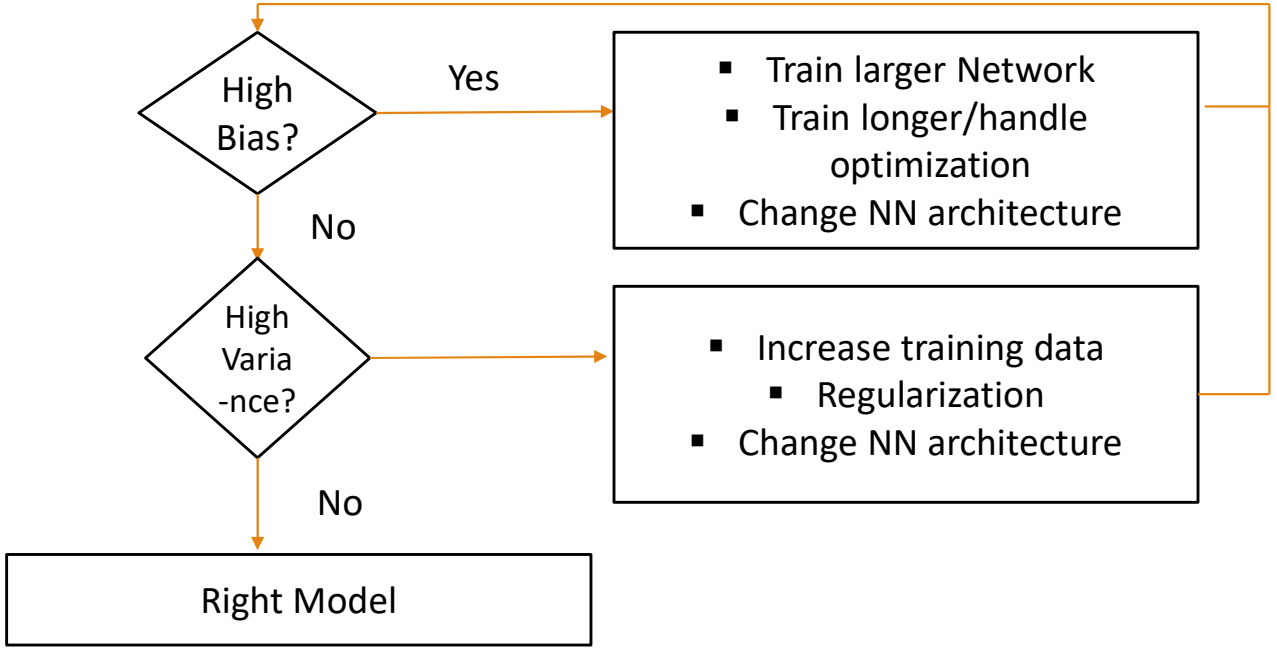


Bias and Variance

- In the deep learning era, we do not talk about a trade-off between bias and variance.
- The train-set error and the validation set-error gives you idea about the bias and variance of the model.
- The training set error actual tells about bias i.e. how well the model is fitting the training data; and the validation-set error tells about the variance (as shown in the table below)

Classifier Performance (Assuming Human Error/Optimal Bias Error ~ 0%)			
Training Set Error	1%	15%	0.5%
Validation Set Error	15%	16%	1%
Type of Model	High Variance (Overfitting)	High Bias (Underfitting)	Low Bias and Variance

Bias and Variance (Contd……)



Regularization in Neural Networks

- Like machine learning algorithms, regularization helps to handle overfitting in the model.
- Regularization is a technique used for tuning the function by adding an additional penalty term in the error function that reduces the magnitude of parameters.
- The cost function for the neural network is modified as:

$$J(W_1, b_1, W_2, b_2, \dots \dots . W_L, b_L) = \frac{1}{n}\sum_{i=1}^{n} L(a_L^i, y^i) + \frac{\lambda}{2n}\sum_{l=1}^{L} \|W^l\|_F^2$$

- The factor $\|W^L\|_F^2$ is called Frobenius normalization and is computed as:

$$\|W^l\|_F^2 = \sum_{i=1}^{n_l}\sum_{j=1}^{n_{l-1}} (W_{ij}^l)^2$$

Regularization in Neural Networks (Contd…)

- Gradient Descent Optimization with Regularization:

$$\frac{\partial J}{\partial W_L} = \frac{1}{n} dZ_L . A_{L-1}^T + \frac{\lambda}{n} W_L$$

Where $dZ_L = dAL * g_L'(Z_L)$ and are obtained through backpropagation phase of neural network.

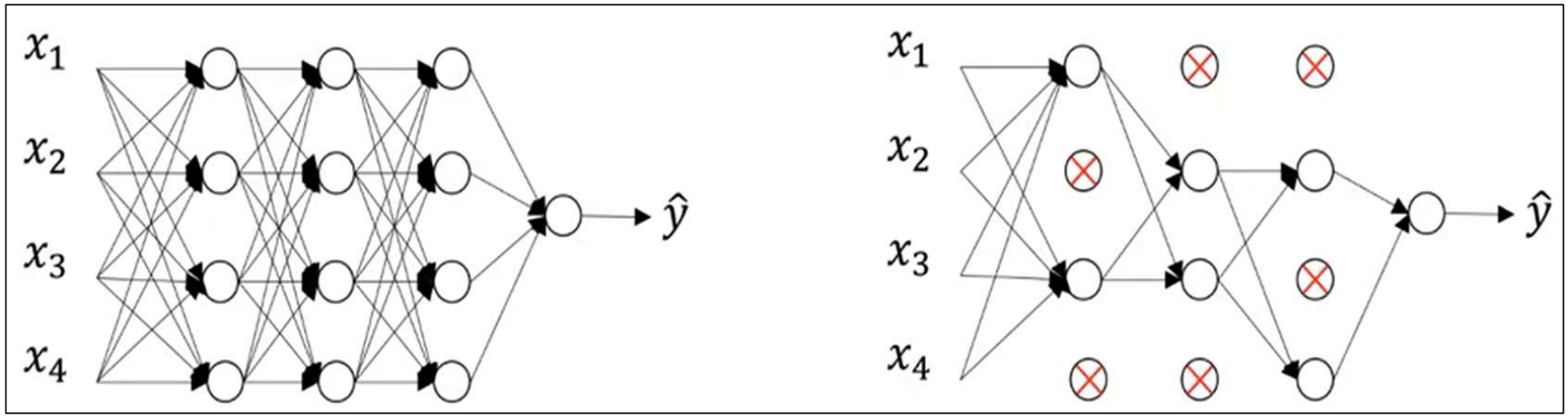
- The Weights are then updated as:

$$W_L = W_L - \alpha \left(\frac{1}{n} dZ_L . A_{L-1}^T + \frac{\lambda}{n} W_L \right)$$

$$W_L = W_L \left(1 - \frac{\alpha\lambda}{n} \right) - \left(\frac{\alpha}{n} dZ_L . A_{L-1}^T \right)$$

Dropout Regularization

- Dropout is a regularization method that approximates training a large number of neural networks with different architectures.
- During training, some number of layer outputs are randomly ignored or "*dropped out*."
- By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections.



Implementing Dropout Regularization

- The most common method to implement dropout regularization is **inverted dropout**.
- In inverted dropout, we define dropout rate and a drop out layer before the activation layer. The dropout layer is computed as:

$$D_L = np.ranodm.rand\big((A_L.shape[0], A_L.shape[1])\big) < dropout\_rate$$

- The activation layer is then computed as:

$$Z_L = W_L A_{L-1} + b_L$$
$$A_L = g_L(Z_L)$$
$$A_L = A_L * D_L = np.multiply(A_L, D_L)$$

(where * denote element wise multiplication)

$$A_L = \frac{A_L}{dropout\ rate}$$

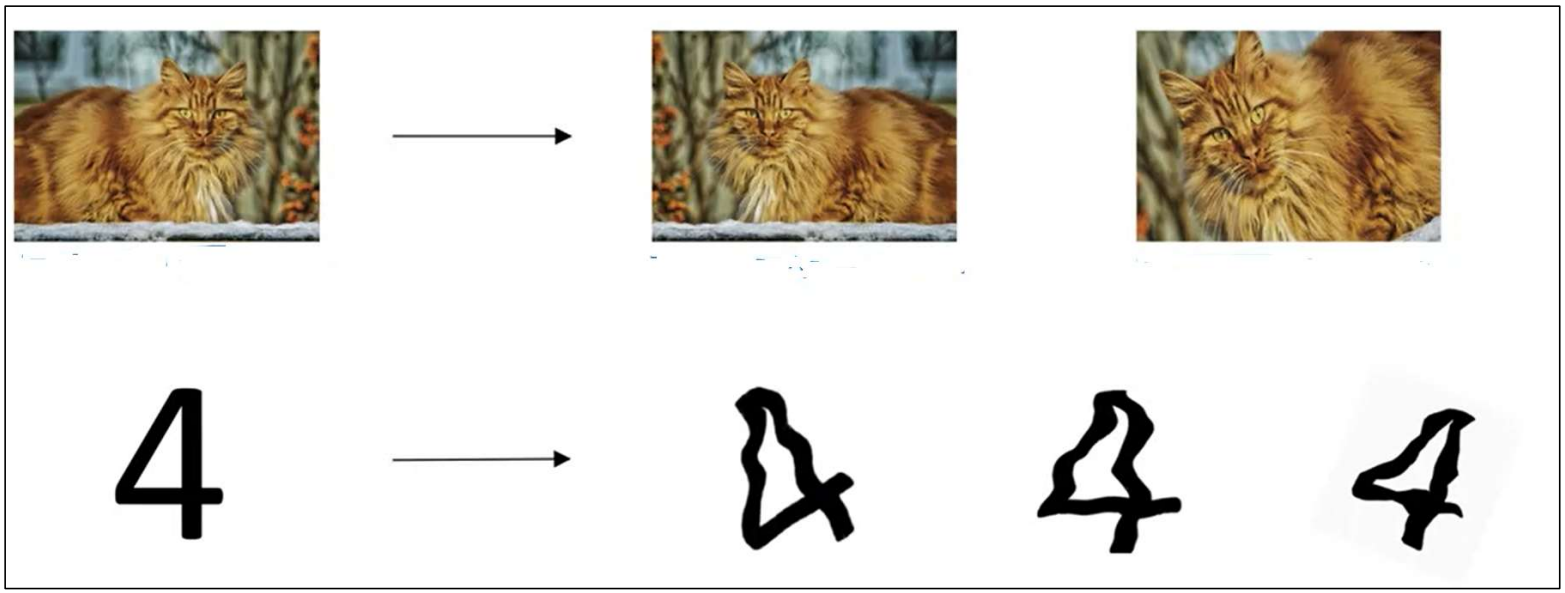
Due to last step, it is called inverted dropout. It is done to scale up the values of activation layer, so that next layers are activations are not effected.

Understanding Dropout

- The main intuition behind dropout is that do not rely on any one feature, so spread weights.
- The spreading of these weights, results in shrinkage of the weights. Hence dropout regularization is equivalent to L2-normalization.
- The dropout rate can be set different for different layers. For example, for the layers with more number of neurons, we can set dropout rate to be low, so that more number of neurons are dropped out and the layer do not overfit.

Other Regularization methods

- **Data Augmentation**: Enlarge the dataset



Other Regularization methods (Contd…)

- **Early Stopping**: is a regularization technique for deep neural networks that stops training when parameter updates no longer begin to yield improves on a validation set.