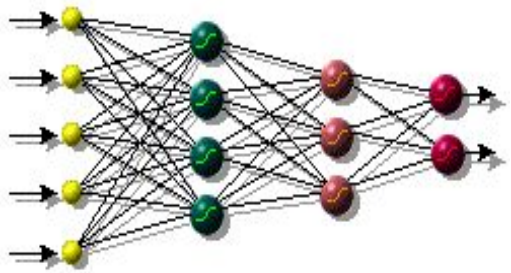
Regularization of Neural Networks



Dinesh K. Vishwakarma, Ph.D.

ASSOCIATE PROFESSOR, DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY, DELHI.

Webpage: http://www.dtu.ac.in/Web/Departments/InformationTechnology/faculty/dkvishwakarma.php

Regularization

- To improve the performance of the NN, regularization is done.
- An NN performs incredibly well on the training set, but not nearly as good on the test set.
- NN has a very high variance and it cannot generalize well to data it has not been trained on.
- These are the **sign of overfitting.**

Solution of Overfitting

- **Get more data**
- **Use regularization**

✔ Getting more data is sometimes impossible, and other times very expensive.

✔ Therefore, regularization is a common method to reduce overfitting and consequently improve the model's performance.

Solution of Overfitting…

- Two most common approach used as Regularization for NN:
  - **L2 regularization**
  - **Dropout**.

L2 regularization

- The cost function can be defined as $\mathbf{J}(\boldsymbol{w^1}, \boldsymbol{b^1} \ldots\ldots \boldsymbol{w^L}, \boldsymbol{b^L}) = \frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \boldsymbol{L}(\hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}_{\boldsymbol{i}})$.
- Where L can be a loss function such as cross entropy loss function.
- L2 regularization, a component is added that penalizes large weights.

$$J\left(w^{[1]}, b^{[1]}, \ldots, w^{[L]}, b^{[L]}\right) = \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) + \frac{\lambda}{2m} \sum_{l=1}^{L} \left\| w^{[L]} \right\|_F^2$$

L2 regularization…

- *Lambda:* **regularization parameter**. The addition of the Frobenius norm, denoted by the subscript *F*.
- *lambda* is a parameter that can be tuned.

✔ Larger weight values will be more penalized if the value of *lambda* is large.

✔ Similarly, for a smaller value of *lambda*, the regularization effect is smaller.

- This makes sense, because the cost function must be minimized.
- By adding the squared norm of the weight matrix and multiplying it by the regularization parameters, large weights will be driven down in order to minimize the cost function.

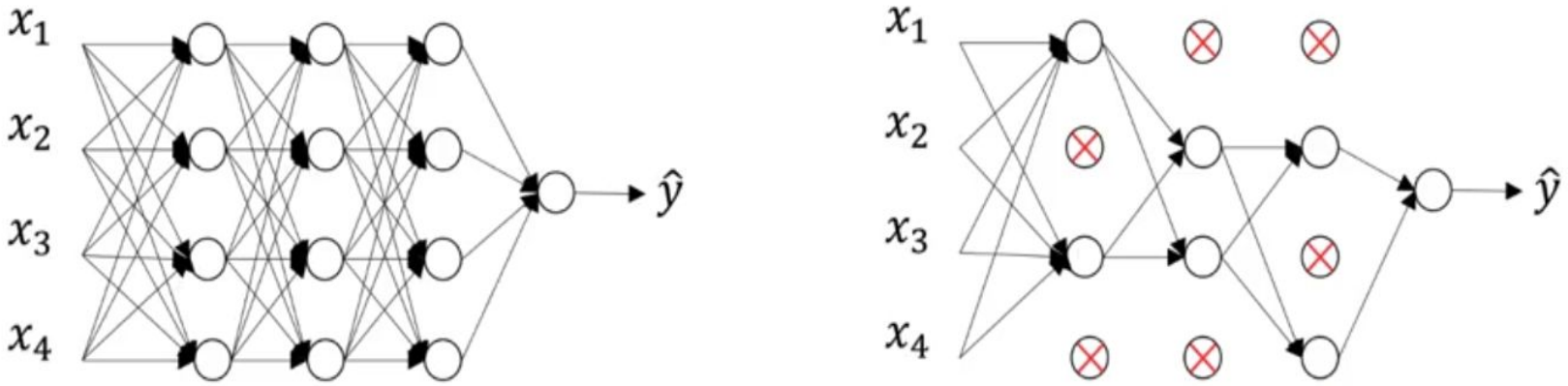
How Regularization Works?

- Adding the regularization component will drive the values of the weight matrix down. This will effectively de-correlate the NN.
- Recall, we feed the activation function with the following weighted sum: $\boldsymbol{z = w^T x + b}$.
- By reducing the values in the weight matrix, $\boldsymbol{z}$ will also be reduced, which in turns decreases the effect of the activation function.
- Therefore, a less complex function will be fit to the data, effectively reducing overfitting.

Dropout Regularization

- Dropout involves going over all the layers in a neural network and setting probability of keeping a certain nodes or not.
- The input layer and the output layer are kept the same.
- The probability of keeping each node is set at random. Only threshold is decided: a value that will determine if the node is kept or not.
- For example, if you set the threshold to 0.8, then there is a probability of 20% that a node will be removed from the network.
- Therefore, this will result in a much smaller and simpler neural network.

Dropout Regularization…

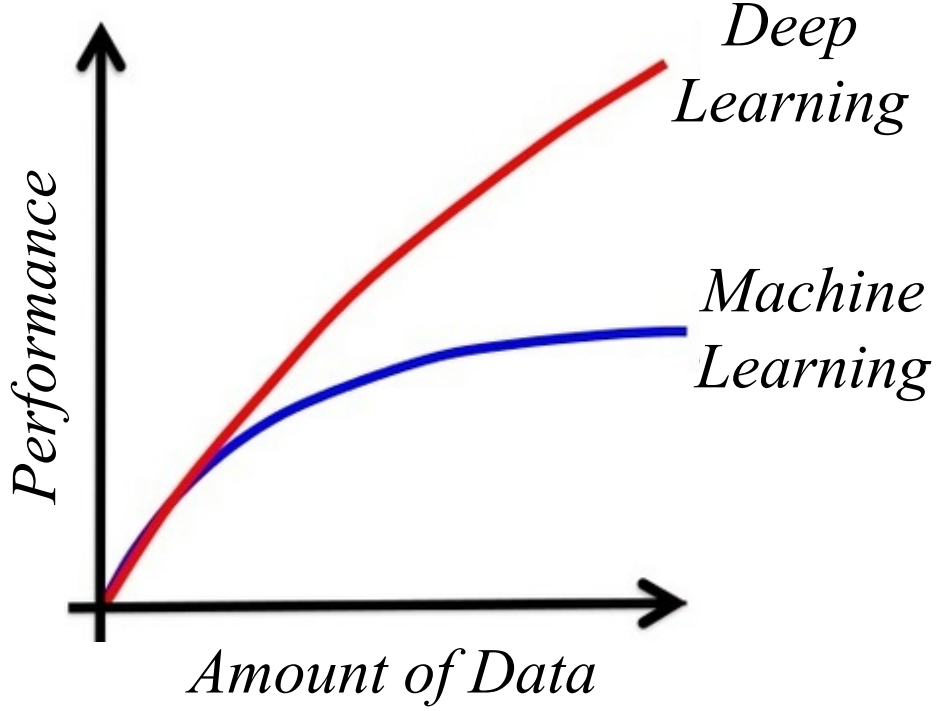


- Dropout means that the NN cannot rely on any input node, since each have a random probability of being removed. Therefore, the NN will be reluctant to give high weights to certain features, because they might disappear.
- Consequently, the weights are spread across all features, making them smaller. This effectively shrinks the model and regularizes it.

- https://towardsdatascience.com/how-to-improve-a-neural-network-with-regularization-8a18ecda9fe3

When to use Deep Learning?

- Data size is large
- High end infrastructure
- Lack of domain understanding
- Complex problem such as image classification, speech recognition etc.



Fuel of deep learning is the big data by Andrew Ng

Limitations of Deep Learning

- Very slow to train
- Models are very complex, with lot of parameters to optimize:

✔ Initialization of weights

✔ Layer-wise training algorithm

✔ Neural architecture

- Number of layers
- Size of layers
- Type – regular, pooling, max pooling, soft max

✔ Fine-tuning of weights using back propagation

Thank you!

dinesh@dtu.ac.in