18AIC301J: DEEP LEARNING TECHNIQUES

B. Tech in ARTIFICIAL INTELLIGENCE, 5th semester

Faculty: **Dr. Athira Nambiar**

Section: A, slot:D

Venue: TP 804

Academic Year: 2022-22

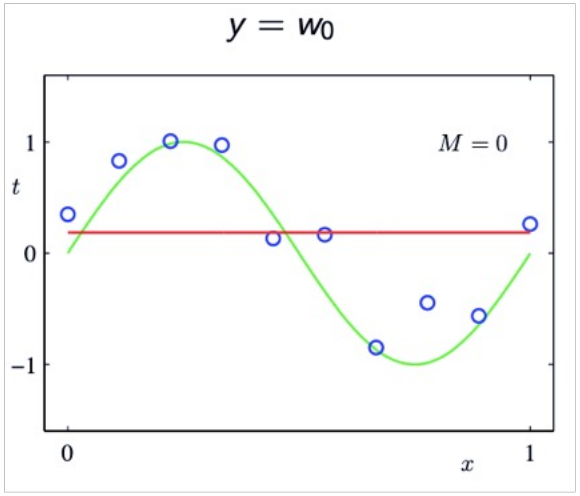
UNIT-2

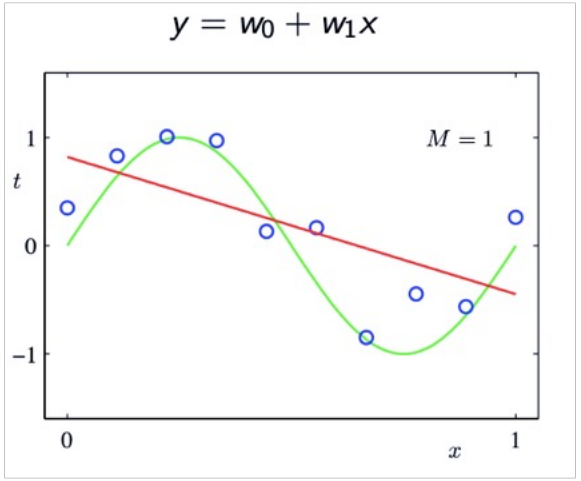
Limitations of gradient descent learning algorithm, Contour maps,
Momentum based gradient descent, Nesterov accelerated gradient Descent, AdaGrad, RMSProp, Adam learning Algorithm, Stochastic gradient descent
Implement linear regression with stochastic gradient descent
Mini-batch gradient descent, Bias Variance tradeoff, Overfitting in deep neural networks,
Hyperparameter tuning, Regularization: L2 regularization, Dataset Augmentation and Early stopping
Implement linear regression with stochastic mini-batch gradient descent and compare the results with previous exercise.
Dimensionality reduction, Principal Component Analysis, Singular value decomposition
Autoencoders, Relation between PCA and Autoencoders, Regularization in Autoencoders
Optimizing neural networks using L2 regularization, Dropout, data augmentation and early stopping.

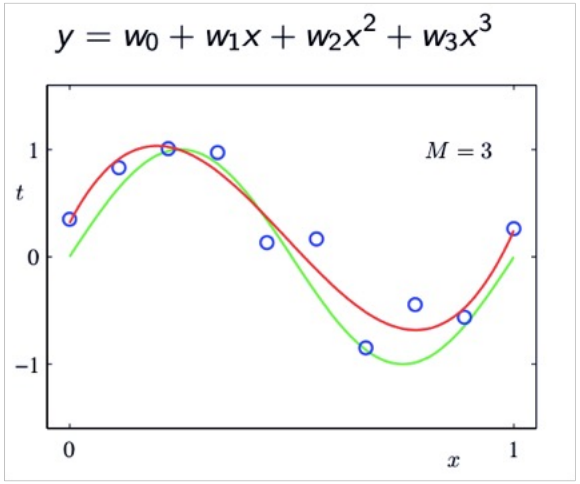
What is OVERFITTING?
GENERALIZATION?

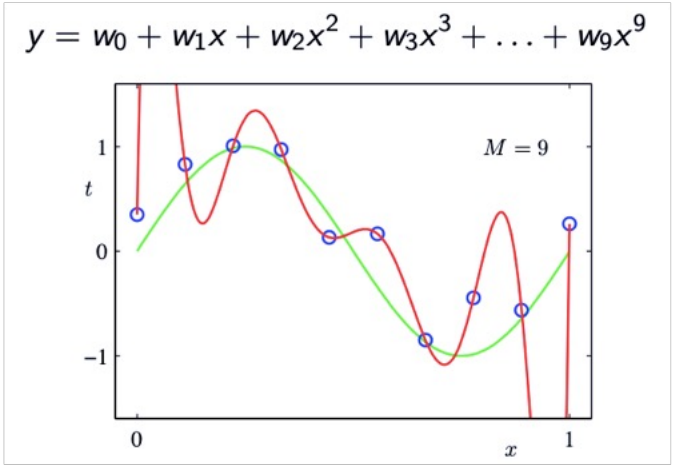
Fitting polynomials

Plots of polynomials having various orders M, shown as red curves,





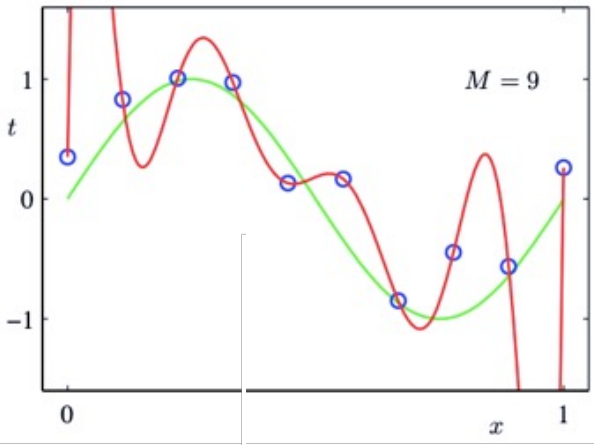




Pattern Recognition and Machine Learning, Christopher Bishop.

Overfitting

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_9 x^9$$



Overfitting the Training Data

The model is too complex - fits perfectly, does not generalize.

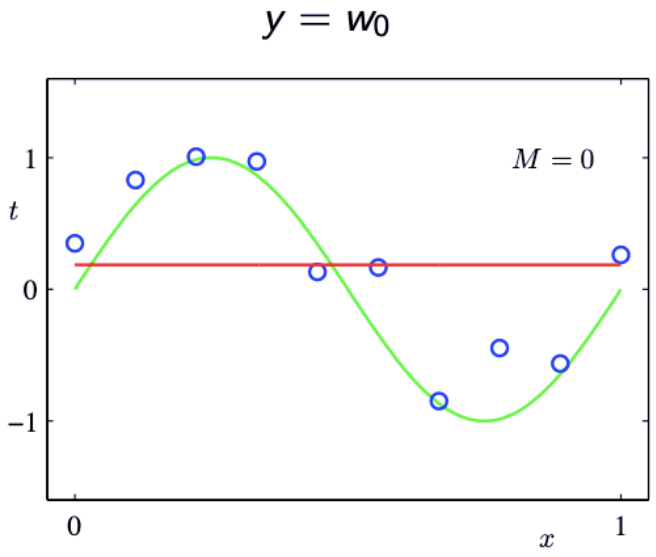
Need a "complex" model with higher order polynomials to fit the data set?

NO, it may be possible that such a model gives very wrong predictions on Test data.

Though it fits well on training data, it may fail to estimate the real relationship among variables beyond the training set. This is known as **"Over-fitting"**

Underfitting

- *underfitting* is the opposite of overfitting
- it occurs when the model is too simple to learn the underlying structure of the data.



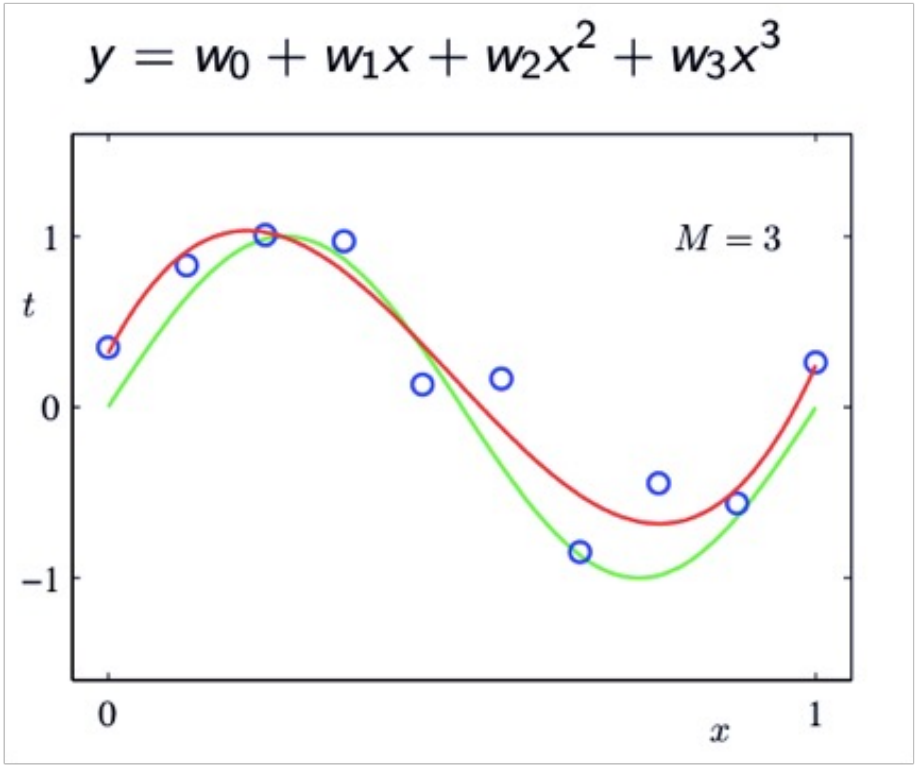
- The model neither fits the training data nor generalizes on the new data.

The main options to fix this problem are:

- Selecting a more powerful model, with more parameters
- Feeding better features to the learning algorithm (feature engineering)
  • Reducing the constraints on the model (e.g., reducing the regularization hyper- parameter)

Generalization

- The models should **generalize** to data they haven't seen before



Overfitting: solutions?

Overfitting happens when the model is too complex relative to the amount and noisiness of the training data.

The possible solutions are:

- To simplify the model by selecting one with fewer parameters by reducing the number of attributes in the training data or by constraining the model
- To gather more training data
- To reduce the noise in the training data (e.g., fix data errors and remove outliers)
- Regularization

Bias-Variance tradeoff

An important theoretical result of statistics and Machine Learning is the fact that a model's generalization error can be expressed as the sum of three very different errors:

Bias

This part of the generalization error is due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic. A high-bias model is most likely to underfit the training data

Variance:

This part is due to the model's excessive sensitivity to small variations in the training data. A model with many degrees of freedom (such as a high-degree polynomial model) is likely to have high variance, and thus to overfit the training data.

Bias-Variance tradeoff

Irreducible error

This part is due to the noisiness of the data itself. The only way to reduce this part of the error is to clean up the data (e.g., fix the data sources, such as broken sensors, or detect and remove outliers).

The formula that connects test MSE to bias,variance and irreducible error:

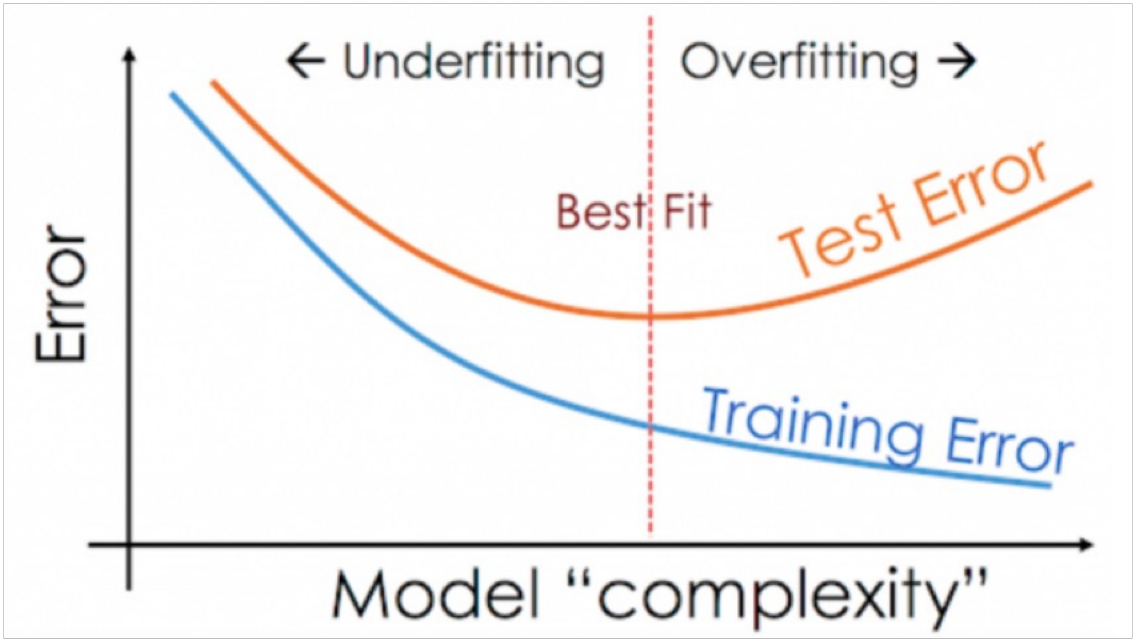
The formula that connects test MSE to bias,variance and irreducible error:

$$E\left[\left(y - \hat{f}(x)\right)^2\right] = \left(\text{Bias}\left[\hat{f}(x)\right]\right)^2 + \text{Var}\left[\hat{f}(x)\right] + \sigma^2$$

- Total error = Bias^2 + Variance + Irreducible error

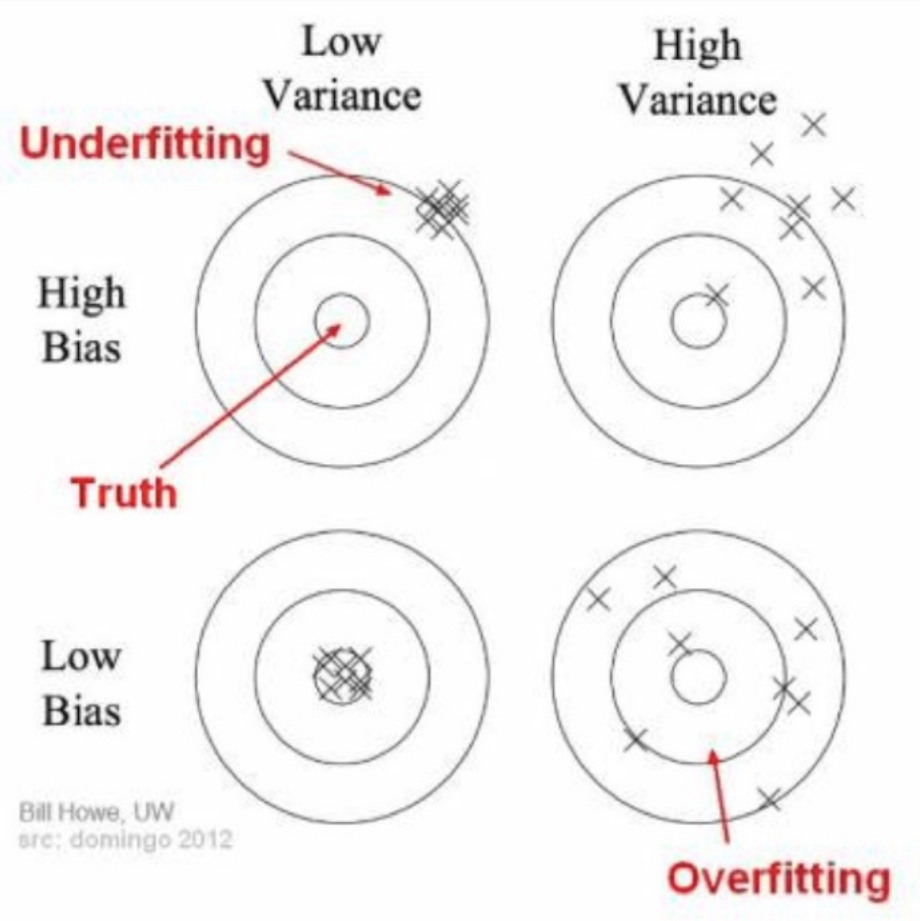
Model error vs. Complexity

HIGH BIAS **HIGH VARIANCE**



Overfitting and Underfitting represented using model error vs complexity

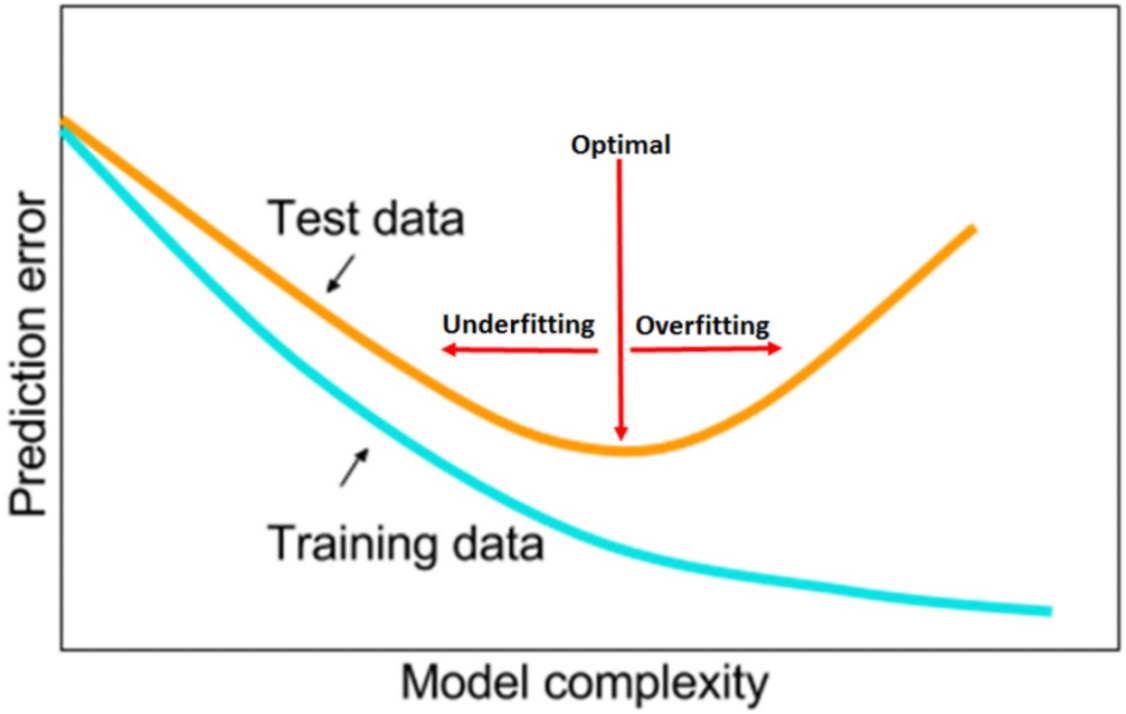
Bias and Variance



https://en.wikipedia.org/wiki/Bias–variance_tradeoff

Approach

1. Observe and understand the clues available during training by **monitoring validation/test loss** early in the training, tune your architecture and hyper-parameters with short runs of a few epochs.
2. Signs of ***underfitting*** or ***overfitting*** of the test or validation loss early in the training process are useful for tuning the hyper-parameters.



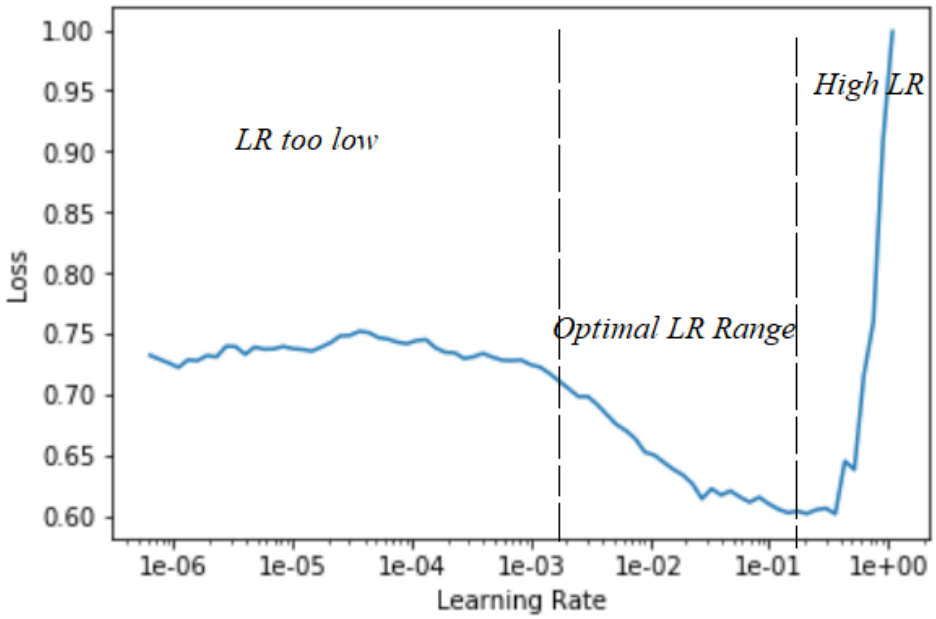
Model complexity refers to the capacity of the machine learning model. The figure shows the optimal capacity that falls between underfitting and overfitting.

Hyperparameter tuning

Finding Optimal Hyper-parameters

(1) Learning Rate (LR)

If the learning rate (LR) is too small, overfitting can occur. Large learning rates help to regularize the training but if the learning rate is too large, the training will diverge.



Hyperparameter tuning

(1) Learning Rate (LR)

- Perform a learning rate range test to identify a “large” learning rate.

- Using the 1-cycle LR policy with a maximum learning rate determined from an LR range test, set a minimum learning rate as a tenth of the maximum.

LR range test: Run your model for several epochs while letting the learning rate increase linearly between low and high LR values. This test is enormously valuable whenever you are facing a new architecture or dataset.

Hyperparameter tuning

(2) Batch Size

- Batch size must be examined in conjunction with the execution time of the training

- The batch size is limited by your hardware's memory, while the learning rate is not.

- Use as large batch size as possible to fit your memory then you compare performance of different batch sizes.

- Small batch sizes add regularization while large batch sizes add less, so utilize this while balancing the proper amount of regularization.

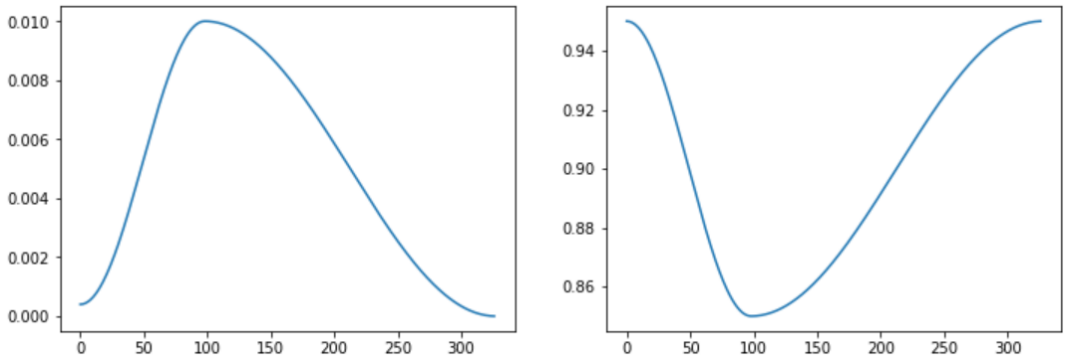
- It is often better to use a larger batch size so a larger learning rate can be used.

Hyperparameter tuning

(3) Momentum

•Test with short runs of momentum values 0.99, 0.97, 0.95, and 0.9 to get the best value for momentum.

•If using the 1-cycle learning rate schedule, it is better to use a cyclical momentum (CM) that starts at this maximum momentum value and decreases with increasing learning rate to a value of 0.8 or 0.85.



left: learning rate one cycle, right:momentum for one cycle

Hyperparameter tuning

(4) Weight decay

- **Weight decay** is defined as multiplying each weight in the gradient descent at each epoch by a factor λ [0<λ<1].

•Weight Decay, is a regularization technique applied to the weights of a neural network. We minimize a loss function compromising both the primary loss function and a penalty on the norm of the weights

$$L_{new}(w) = L_{original}(w) + \lambda w^T w$$

λ is a value determining the strength of the penalty (encouraging smaller weights).

https://nanonets.com/blog/hyperparameter-optimization/

Regularization: L2 regularization

L2 & L1 regularization:

- L1 and L2 are the most common types of regularization.
- These update the general cost function by adding another term known as the regularization term.

Cost function = Loss (say, binary cross entropy) + Regularization term

- Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

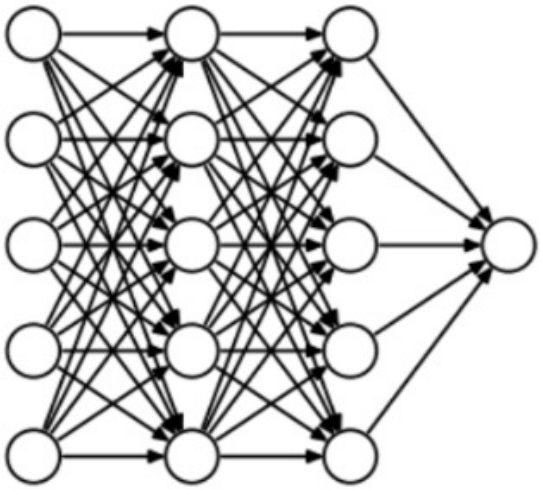
- In L2, we have:

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum ||w||^2$$

(**lambda** is the regularization parameter)

Dropout

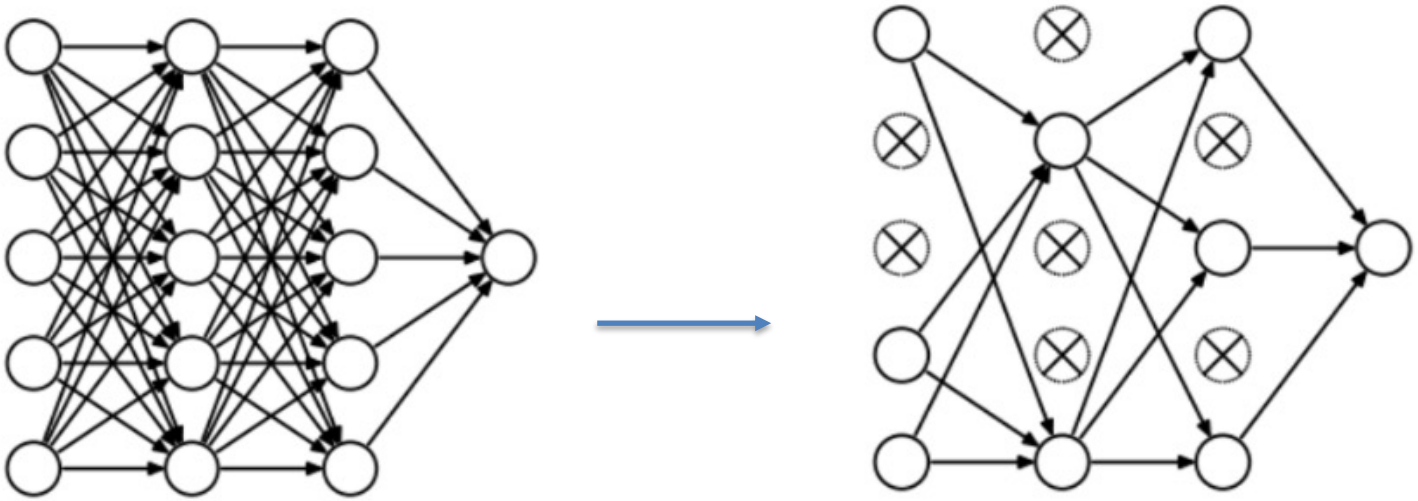
- This is the one of the most interesting types of regularization techniques.
- It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.
- To understand dropout, let's say our neural network structure:



Dropout

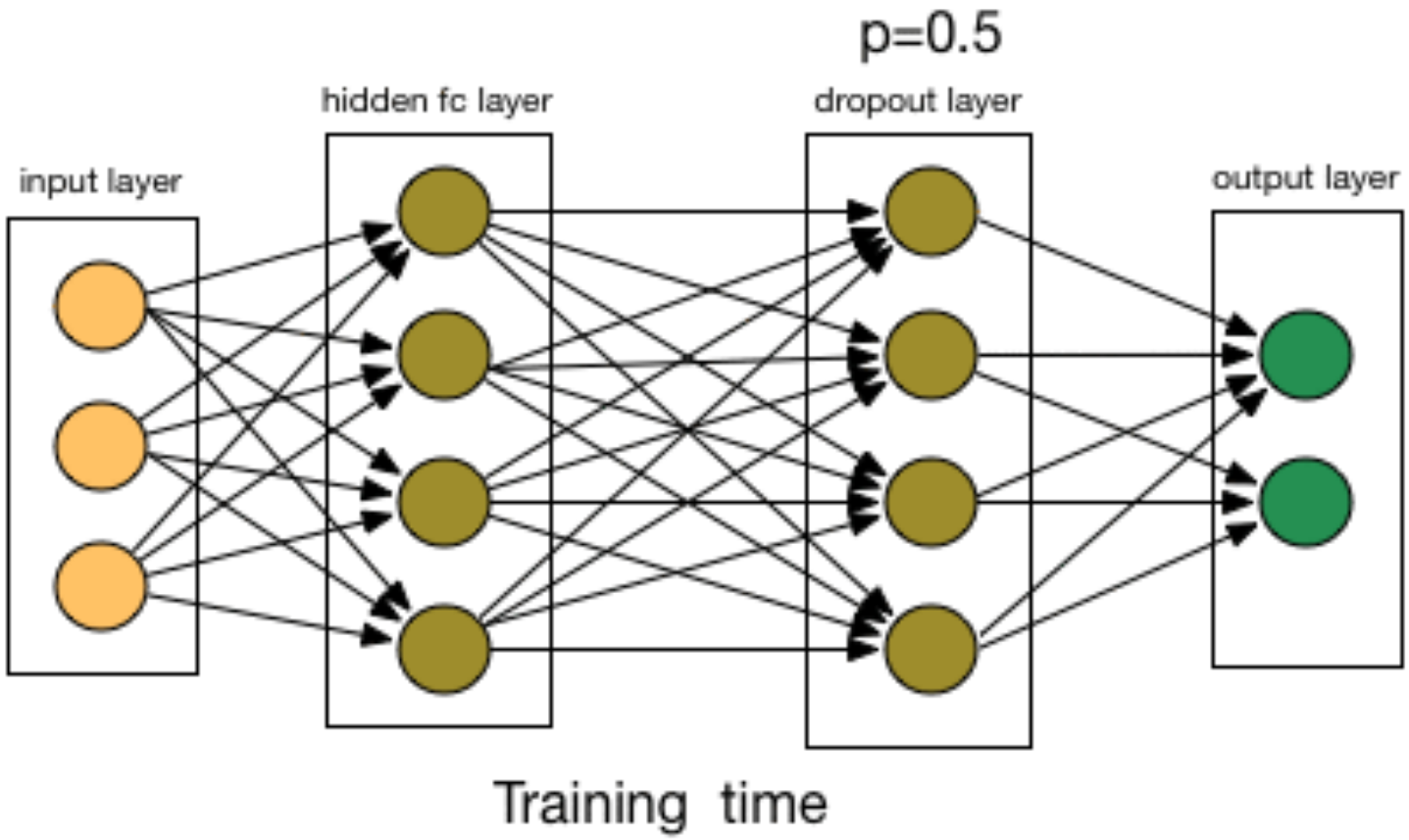
What does dropout do?

At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as shown below.



So each iteration has a different set of nodes and this results in a different set of outputs. **It can also be thought of as an ensemble technique in machine learning.**

Dropout

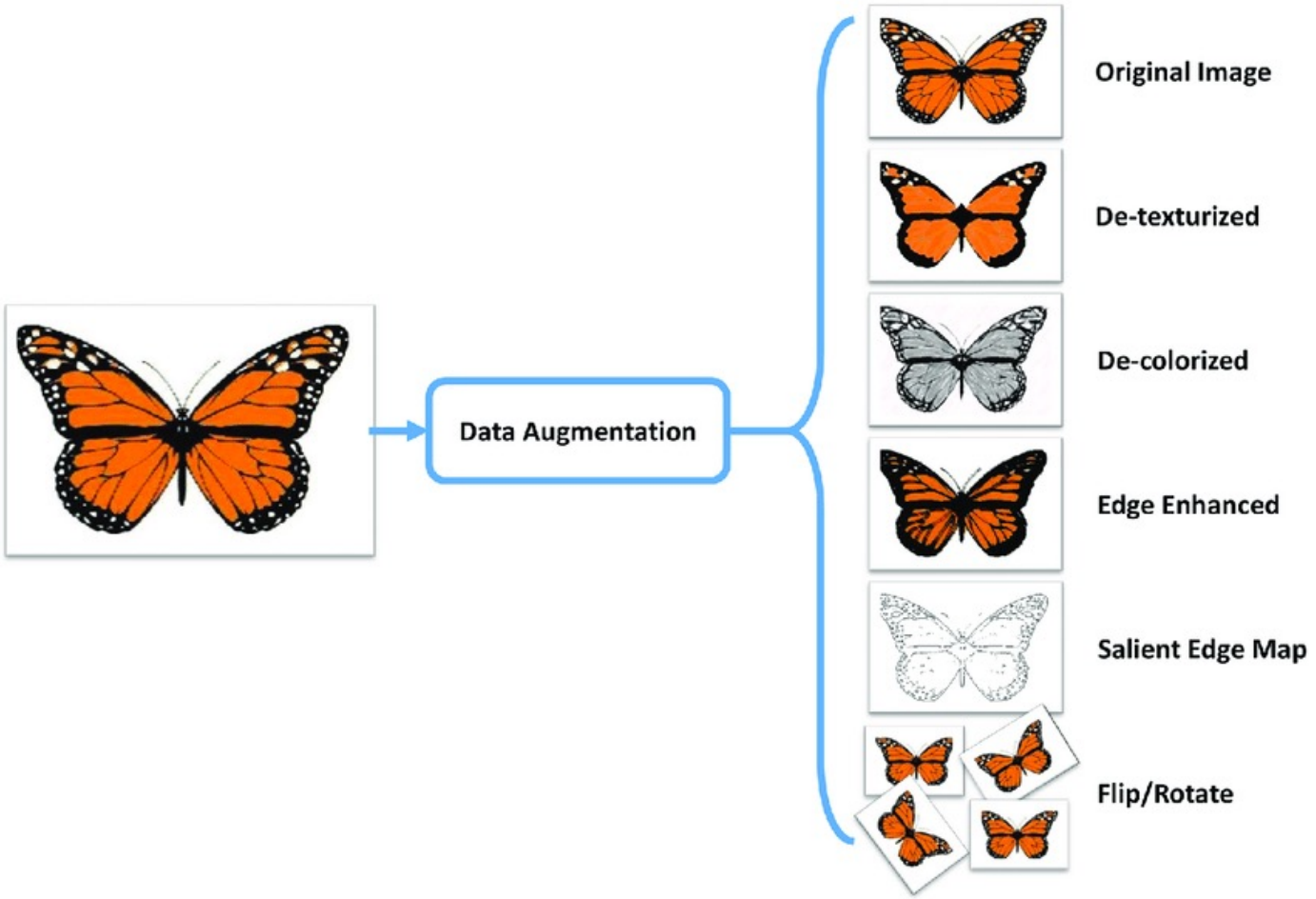


This probability of choosing how many nodes should be dropped is the hyperparameter of the dropout function. As seen in the image above, dropout can be applied to both the hidden layers as well as the input layers.

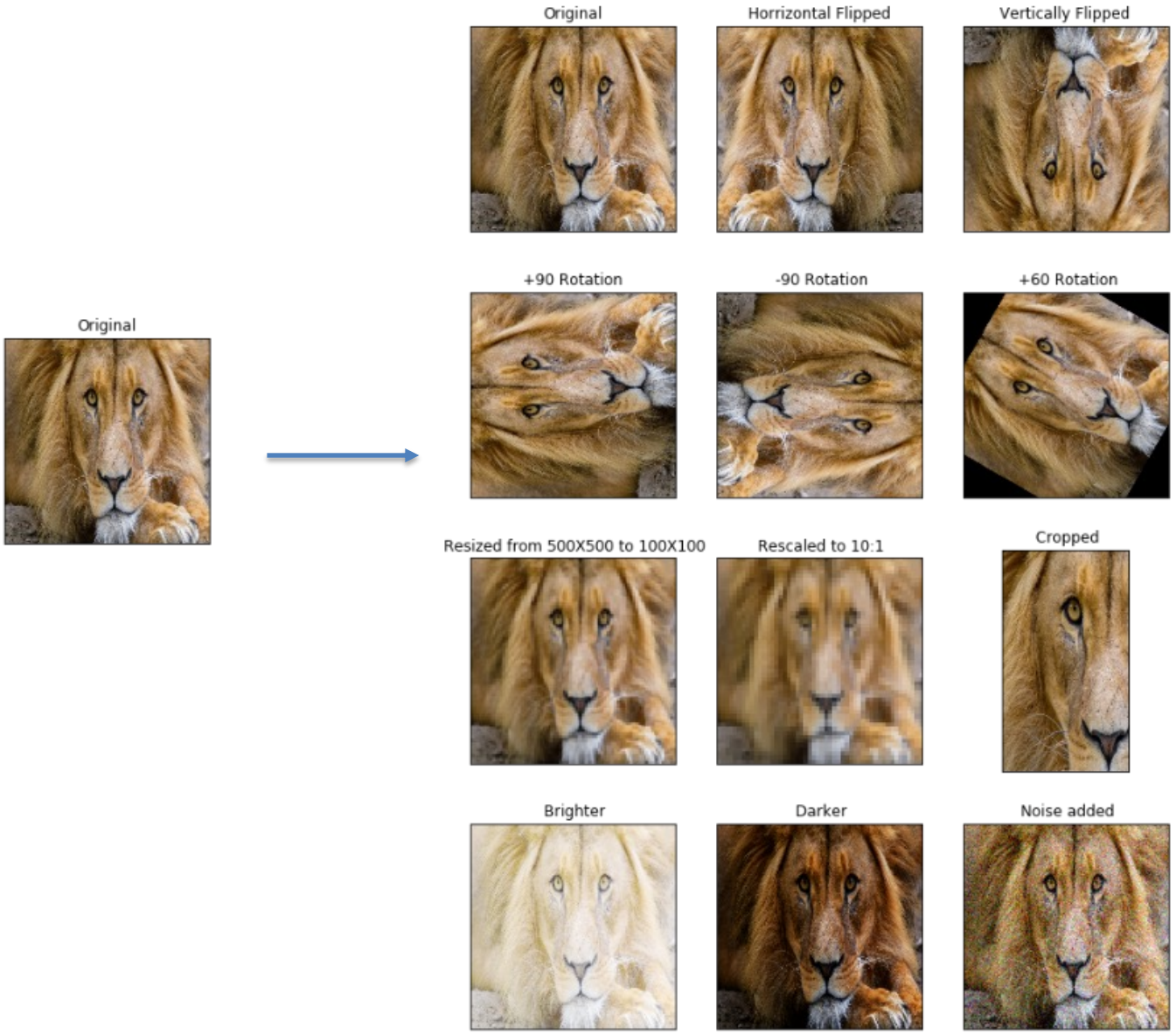
Dataset Augmentation

- The simplest way to reduce overfitting is to increase the size of the training data.
- In machine learning, we were not able to increase the size of training data as the labeled data was too costly.
- There are a few ways of increasing the size of the training data – rotating the image, flipping, scaling, shifting, etc.
- Some transformation has been done on the handwritten digits dataset.

Dataset Augmentation

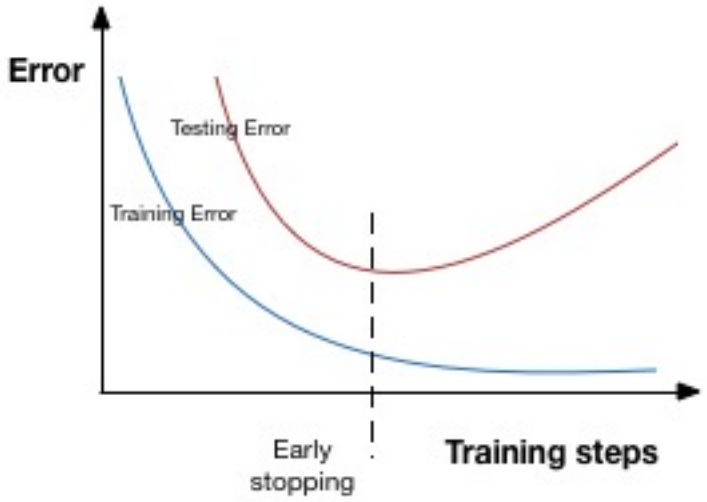


Dataset Augmentation



Early stopping

- Early stopping is a kind of cross-validation strategy where we keep one part of the training set as the validation set.
- When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.
  - Stop training at the dotted line since after that our model will start overfitting on the training data.



Learning Resources

- Charu C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018.
- Eugene Charniak, Introduction to Deep Learning, MIT Press, 2018.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
- Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.
- Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.
- https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

Thank you