Project-2, Week-01

1. Original Figure:

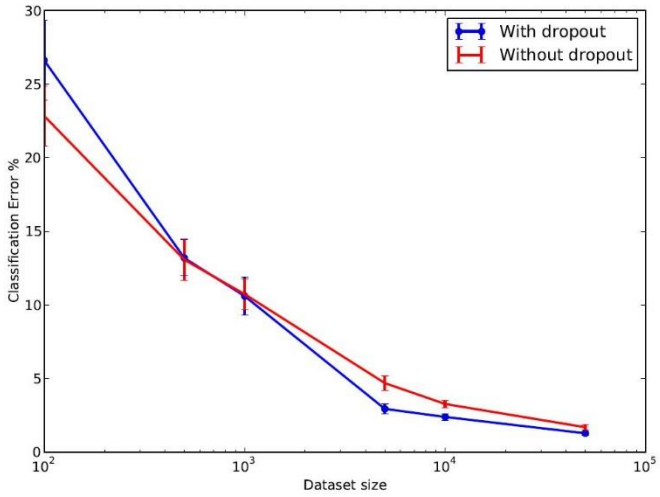

Fig: Effect of varying dataset size on dropout

Citation: Nitish, et al. Dropout: A simple way to prevent neural networks from overfitting, Figure 10. (http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf)

Problem Setting:

Input: 28x28 pixel grayscale image

Output: Ten classes (Each for 0-9 digits)

Function: The network will learn a function that predicts a value to represent the estimated probability that an image x has digit class k, for k=1, 2, 3, …., 9.

In this classification project, I will implement dropout method as a regularization term in the neural network. The size of the data set would be varied and the corresponding classification error (using dropout and without using dropout) would be measured. By this way the above figure would be generated.

Data Source:

Real-World Problem: Classification of Handwritten Digits

Dataset link: http://yann.lecun.com/exdb/mnist/

Training Observation: 60000, Test Observation: 10000
Each observation contains a 28x28 (784 pixel) input feature.
Each observation contains an output id (0-9) of a digit.

Pseudocode:

1. Input: 28x28 Grayscale image (784 pixel)
Output: Class Label (0-9)
2. Set the parameters of the neural networks (such as weights, number of hidden layers, neuron connections, loss function, etc.)
3. Implement dropout method in the network
4.Set the training data number
5. Train the network with training dataset
6. Test the network with test dataset
7. Calculate the test Error

In this project, I will use PyTorch/keras library of Python. In that library, I can set the parameters for each of the neural networks. To generate the figure, I will trace the test error with respect to the data set size. In this project, I will learn about the neural network architecture and get hands on experience on the popular python library “PyTorch” and learn how to implement dropout regularization method.

2. Original Figure:

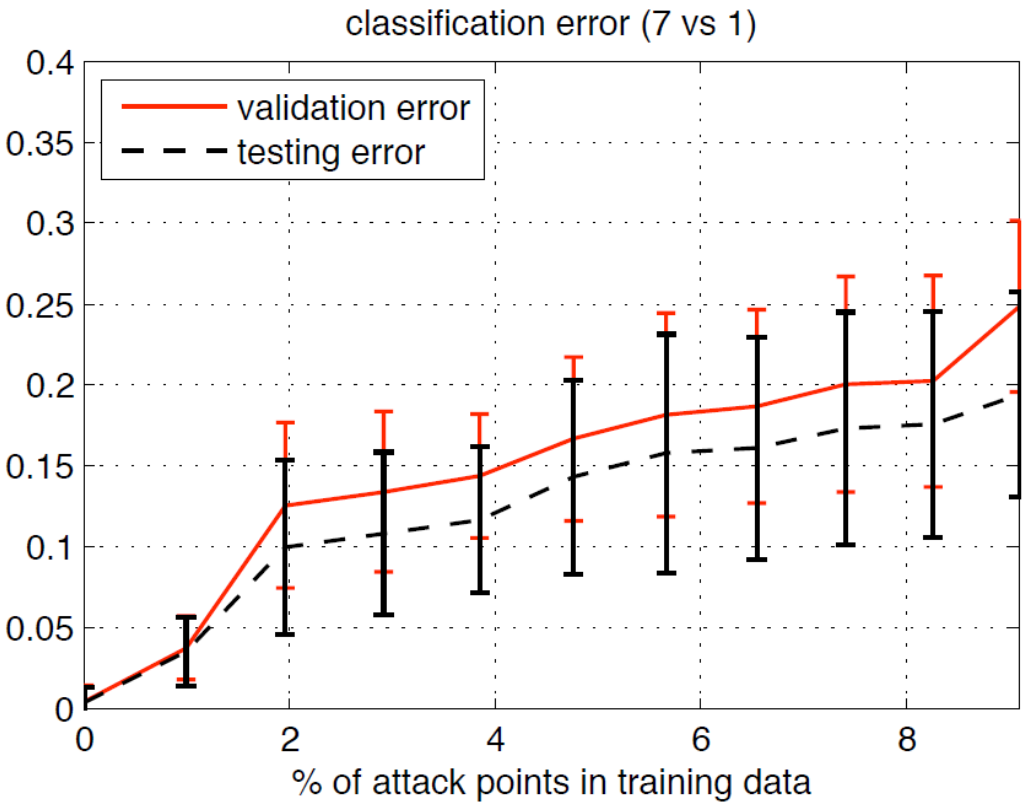

Figure-2: Results of the multi-point, multi-run experiments on the MNIST data set. In each plot, we show the classification errors due to poisoning as a function of the percentage of training contamination for both the validation (red solid line) and testing sets (black dashed line).

Citation: Battista et al. Poisoning Attacks against Support Vector Machines, Figure 3 (https://dl.acm.org/citation.cfm?id=3042761)

Problem Setting:

Input: 28x28 pixel grayscale image

Output: Two classes (Each for 7 and 1 digits)

Function: The network will learn a function that predicts a value to represent the estimated probability that an image x has digit class k, for k=1, 2, 3, …., 9.

In this project, poisonous attack would be introduced to the training set and the corresponding classification error would be measured. The network would implement the SVM algorithm. Based on the variation of the percent of attack on the training set, the classification error would be calculated and plotted.

Data Source:

Real-World Problem: Classification of Handwritten Digits

Dataset link: http://yann.lecun.com/exdb/mnist/

Training Observation: 60000, Test Observation: 10000
Each observation contains a 28x28 (784 pixel) input feature.
Each observation contains an output id of a digit.

Pseudocode:

1. Input: 28x28 (Grayscale image)

Output: Class label (7 or 1)

2. Introduce poisonous attack to the training set

3. Learn the parameters of the SVM algorithm

4. Calculate the test error

In this project, I will code SVM from the scratch and also introduce attack to the training set. I will learn the SVM algorithm and also the algorithm for introducing adversarial attacks in the data set.