Support Vector Machine and Multilayer Perceptron for Image Reconstruction

Smriti Shyamal, Dr. Simon Haykin, *McMaster University.*

***Abstract*—This work focuses on implementation of Support Vector Machine (SVM) and Multilayer Perceptron (MLP) for image reconstruction of the front cover of the Neural Networks and Learning Book authored by Dr. Simon Haykin. In this work, we focused on the spider-web part of the original image. The three steps involved were generating the data, bringing in the learning algorithm (for SVM and MLP) and subsequent algorithmic computation. For SVM, we use support vector regression using Radial Basis Function (RBF) kernel. We employed three hidden rectified linear units (ReLU) layers for MLP and back-propagation algorithm was used. In this paper, we first present the implementation details with sufficient background. Our machine learning results demonstrate that SVM and MLP are very efficient in reconstructing complex images. The constructed book cover image is very close to the original image.**

***Keywords*—Machine Learning, Support Vector Machines, Neural Networks.**

I. Introduction

Reconstructing image is a challenging learning task and requires state-of-the-art learning machines to solve the problem. Support Vector Machines (SVM) is widely considered as highly mathematically sound and elegant machine learning algorithm, and have been employed to tackle tough learning problems. The algorithm attempts to find a hyperplane that maximizes the margin of separation between negative and positive samples [1]. A subset of the data points is then found out as support vectors. Multilayer Perceptron (MLP) on the other hand refers to a neural net structure consisting of multiple hidden layers. The weights for the network are learnt using Back-propagation algorithm. Depending on the complexity of the network abstract features for the input data can be extracted [2]. In this work, we implement both SVM and MLP for a image reconstruction problem.

The image in consideration is the book cover image of the Neural Networks and Learning Book authored by Dr. Simon Haykin. We are focusing the spider web part of the figure particularly. The cover image being very complex, we have carried out sophisticated implementations for both SVM and MLP. Both SVM and MLP is described in details to provide sufficient background. We first generated the input data from the image through image pre-processing. The input data is then provided to SVM and MLP. For SVM, we have chosen support vector regression (SVR) with Radial Basis Function (RBF) kernel. Instead of using sigmoid layers we have used three hidden rectified linear units (ReLU) layers for MLP. In this work, the output we are trying to achieve is the original greyscale image. MLP is trained using high performance Graphical Processing Units (GPUs) for faster learning and large number of epochs are considered. SVM is trained on CPU with sufficiently large Random Access Memory (RAM) for quicker final result. The detailed simulation results are presented in the paper with a detailed discussion on the performances of SVM and MLP.

Dr. Haykin is with the Department of Electrical and Computer Engineering, McMaster University, Canada.

II. Theoretical Background

The capability to extract useful patterns from raw input data is known as machine learning [2]. In this work we have employed two machine learning techniques viz. Support Vector Machines (MLP) and Multilayer Perceptron (MLP) for image reconstruction. The corresponding algorithms are described in the subsequent subsections.

A. Support Vector Machines

SVM is a popular machine learning approach for classification and regression. Linear SVM regression attempts to find the linear function

$$f(x) = x'w + b, \tag{1}$$

where x_N is the set of N observations and y_n are the responses. Since, we want $f(x)$ to be as flat as possible we need to find the minimal nominal value of $w'w$. We solve the following convex optimization problem

$$\min \quad J(w) = \frac{1}{2}w'w \quad s.t. \forall n : |y_n - (x_n'w + b)| \leq \epsilon \tag{2}$$

where ϵ is the lower bound for the residuals. Since, it is possible that no such function $f(x)$ exits which satisfy the constraints. This leads to introducing slack variables ξ and ξ^* for each point and solving a *soft margin* problem (primal) instead

$$\min \quad J(w) = \frac{1}{2}w'w + C\sum_{n=1}^{N}\left(\xi + \xi^*\right), \tag{3}$$

subject to

$$\forall n : y_n - (x_n'w + b) \leq \epsilon + \xi_n \tag{4}$$

$$\forall n : (x_n'w + b) - y_n \leq \epsilon + \xi_n^* \tag{5}$$

$$\forall n : \xi_n^*, \xi_n \geq 0 \tag{6}$$

where C is the box constraint which helps with regularization. It is straightforward to solve the dual counterpart of the above

problem given as

$$\min\ L(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)x_i' x_j + \epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) \quad (7)$$

subject to

$$\sum_{n=1}^{N}(\alpha_n - \alpha_n^*) = 0 \quad (8)$$

$$\forall n : 0 \le \alpha_n \le C \quad (9)$$

$$\forall n : 0 \le \alpha_n^* \le C \quad (10)$$

where α_n and α_n^* are non-negative multipliers for each observation x_n. The above dual Lagrange formulation can be extended to nonlinear functions as well. In nonlinear SVM regression, the dot product $x_i' x_j$ is replaced with a nonlinear kernel function $G(x_1, x_2) =< \phi(x_1), \phi(x_2) >$, where $\phi(x)$ is a transformation that maps x to a high-dimensional space. Radial basis kernel function used in this work is given as

$$G(x_j, x_k) = \exp\left(-\frac{1}{2\sigma^2}||x_j - x_k||^2\right) \quad (11)$$

where σ is specified by user *a priori*. The *Gram matrix* is an n-by-n matrix that contains elements $g_{i,j} = G(x_i, x_j)$. Each element $g_{i,j}$ is equal to the inner product of the predictors as transformed by ϕ. However, we do not need to know ϕ, because we can use the kernel function to generate Gram matrix directly. Using this method, nonlinear SVM finds the optimal function $f(x)$ in the transformed predictor space.The dual formula for nonlinear SVM regression replaces the inner product of the predictors $x_i x_j$ with the corresponding element of the Gram matrix $(g_{i,j})$.

Nonlinear SVM regression finds the coefficients that minimize

$$\min\ L(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)G(x_i, x_j) + \epsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) - \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) \quad (12)$$

subject to

$$\sum_{n=1}^{N}(\alpha_n - \alpha_n^*) = 0 \quad (13)$$

$$\forall n : 0 \le \alpha_n \le C \quad (14)$$

$$\forall n : 0 \le \alpha_n^* \le C \quad (15)$$

. The function used to predict new values is equal to

$$f(x) = \sum_{n=1}^{N}(\alpha_n - \alpha_n^*)G(x_n, x) + b. \quad (16)$$

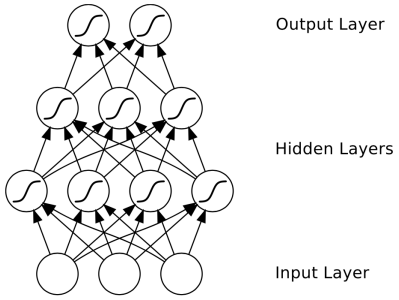

Figure 1. A multilayer perceptron. The S-shaped curves in the hidden and output layers indicate the application of 'sigmoidal nonlinear activation functions. [3].

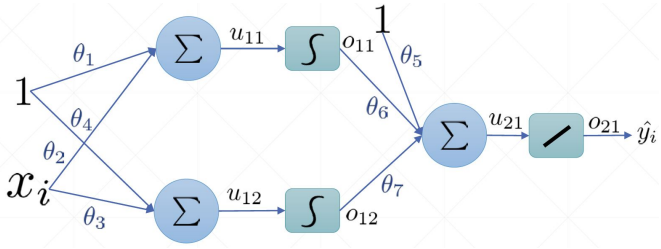

Figure 2. A neural network on 1 dimensional input data set.

The w parameter can be completely described as a linear combination of the training observations using the equation

$$w = \sum_{n=1}^{N}(\alpha_n - \alpha_n^*)x_n. \quad (17)$$

The Karush-Kuhn-Tucker (KKT) complementarity conditions are optimization constraints required to obtain optimal solutions. If either α_n or α_n^* is not zero, then the corresponding observation is called a *support vector*.

B. Multilayer Perceptron

Deep feedforward networks, also called feedforward neural networks, or multilayer perceptrons (MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f^*. For example, for a classifier, $y = f(x)$ maps an input x to a category y. A feedforward network defines a mapping $y = f(x;\theta)$ and learns the value of the parameters θ that result in the best function approximation [2]. The basic structure of an MLP is a network of small processing units, or nodes, joined to each other by weighted connections. The units in an MLP are arranged in layers, with connections feeding forward from one layer to the next [3] (see Fig. 1). In the next subsection we describe a simple neural network and how forward and backward pass works for the same.

a) Neural Network regression: The neural networks can be expressed as set of operations as shown in Fig. 2.

From Fig. 2, first the linear regression model which is a weighted combination of inputs and weights are used. Followed by a sigmoid activation function ($\sigma(x) = \frac{1}{1+e^{-x}}$), which

specifies if the neuron has to be activated or not (1 if activated). Other common activation function used with neural network are, Sigmoid, $\sigma(x) = 1/(1+e^{-x})$, ReLU, $\sigma(x) = max\{0, x\}$, and tanh $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

The θ_1 and θ_2 are weights or parameters here. Ultimately, the final layer has linear activation because we are interested in using neural networks for regression.

In Fig. 2, the output $\hat{y_i}$ is given by,

$$\hat{y_i} = \theta_5 + \theta_6\sigma(\theta_1 + \theta_2 x_i) + \theta_7\sigma(\theta_4 + \theta_3 x_i) \tag{18}$$

Where,

- $\theta' s$ are the parameters.
- x_i is the input of the i^{th} sample
- Activation function (sigma represented in green circular rectangle) is the sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$)
- u_{11}, u_{12} are the outputs of the neurons before activation
- o_{11}, o_{12} are the outputs of the neurons after activation in first layer
- o_{21} is the output of the neuron after activation in the second layer
- Since linear activation is used in final layer, $\hat{y_i} = o_{21} = u_{21}$

Now our goal is to learn the parameters θ of the neural network. If g is the loss function, the parameters are computed using the gradient descent update given as,

$$\theta^{t+1} = \theta^t - \alpha \times \frac{\partial g(\theta)}{\partial \theta} \tag{19}$$

The gradient of loss, g with respect to the parameter is given as,

$$\frac{\partial g(\theta)}{\partial \theta_j} = -2(y_i - \hat{y_i}(x_i, \theta))\frac{\partial \hat{y_i}(x_i, \theta)}{\partial \theta_j} \tag{20}$$

The value of $\hat{y_i}(x_i, \theta)$ can be computed while doing a forward pass (discussed below) and the derivative of $\hat{y_i}$ with respect to the parameter is computed during the back propagation step discussed below.

b) Forward Pass: In forward pass, each element of θ has a known value. Then from Fig. 2, the values of u, o and $\hat{y_i}$ are given as,

- $u_{11} = \theta_1 \times 1 + \theta_2 \times x_i$
- $u_{12} = \theta_4 \times 1 + \theta_3 \times x_i$
- $o_{11} = \frac{1}{1+e^{-u_{11}}}$
- $o_{12} = \frac{1}{1+e^{-u_{12}}}$
- $u_{21} = \theta_5 \times 1 + \theta_6 \times o_{11} + \theta_7 \times o_{12}$
- $\hat{y_i} = o_{21} = u_{21}$

c) Backpropagation: The next step is to compute the partial derivatives of $\hat{y_i}$ with respect to the parameter θ. Each $\hat{y_i}$ can be represented as a function of previous layer's weights as,

$$\hat{y_i} = \theta_5 \times 1 + \theta_6 \times o_{11} + \theta_7 \times o_{12} \tag{21}$$

From the above equation the gradient of $\hat{y_i}$ with respect to $\theta_5, \theta_6, \theta_7$ is given as,

- $\frac{\partial \hat{y_i}}{\partial \theta_5} = 1$
- $\frac{\partial \hat{y_i}}{\partial \theta_6} = o_{11}$
- $\frac{\partial \hat{y_i}}{\partial \theta_7} = o_{12}$

Figure 3. Image reconstruction problem: Original RGB image (image size 1030 X 763).

Similarly, $u_{12} = \theta_2 \times x_i + \theta_3 \times 1$, now the derivative of $\hat{y_i}$ with respect to θ_3 is given in equation (22). It is essential to note that θ_3 only influences $\hat{y_i}$ through u_{12}.

$$\frac{\partial \hat{y_i}}{\partial \theta_3} = \frac{\partial \hat{y_i}}{\partial o_{12}}\frac{\partial o_{12}}{\partial u_{12}}\frac{\partial u_{12}}{\partial \theta_3} \tag{22}$$

The $o' s$ in the equation 22 are received during the forward pass. Similarly the gradient with respect to $\theta_1, \theta_2, \theta_4$ are computed.

Once the $\hat{y_i}$ and the derivatives of $\hat{y_i}$ with respect to the θ are computed, the gradient descent as given in 19 is used to update the parameters θ and the forward pass and update step continues until the optimum value of parameter (or minimal loss) is attained.

III. Project Description

A. Problem Statement

The image reconstruction problem involved reconstructing the book cover image of Neural Networks and Learning Machines book by Dr. Haykin. The original color image was first cropped and converted from RGB (see Fig. 3) to gray-scale. Also, the texts were removed from the image. The image

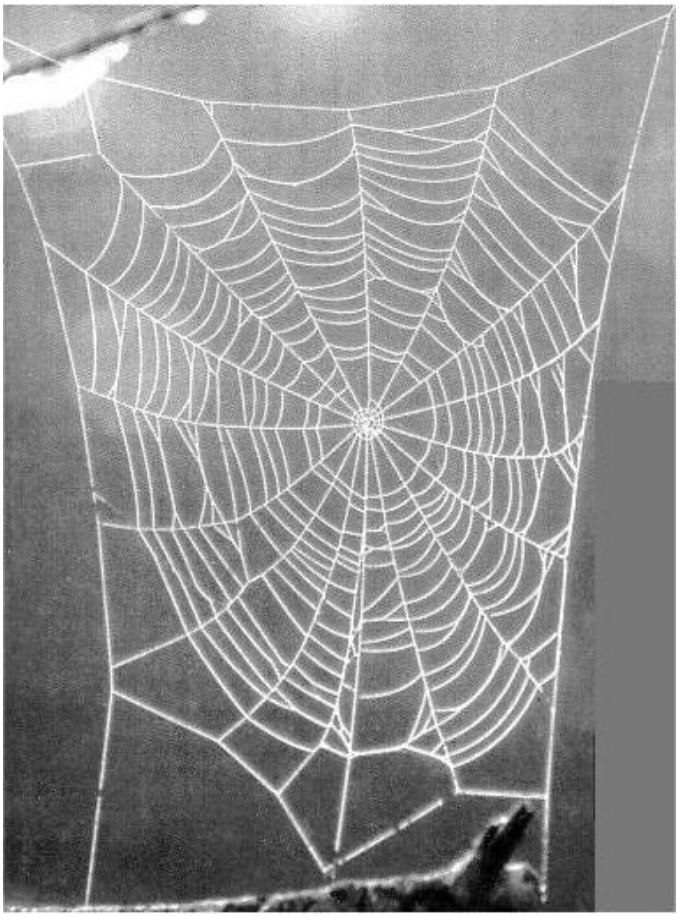

Figure 4. Image reconstruction problem: Output image we are trying to reconstruct (image size 1030 X 763 with gray-scale value ranging from 0 to 255).

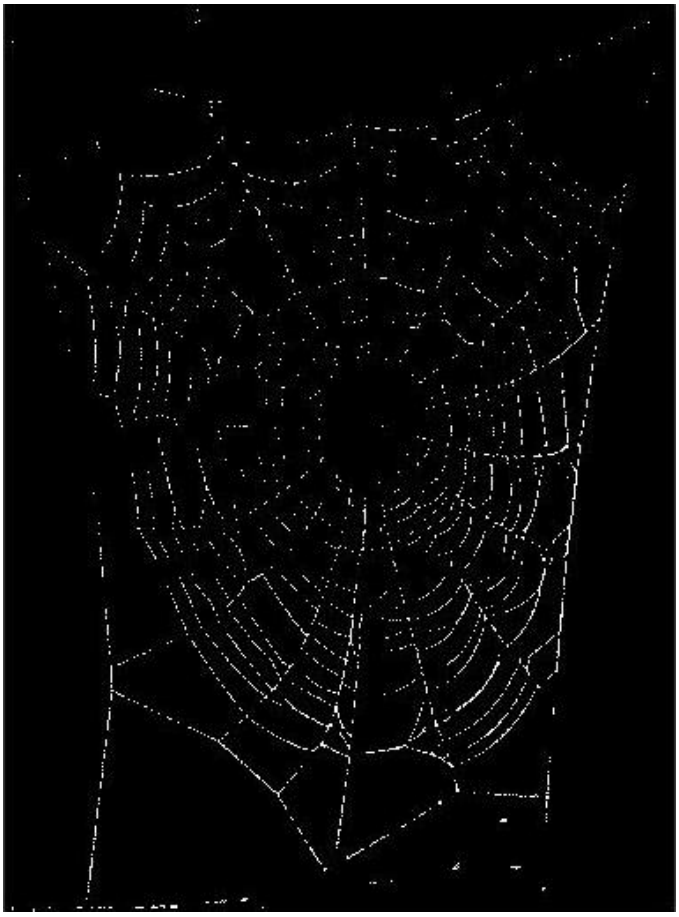

Figure 5. Image reconstruction problem: Input data we are using for the project (image size 1030 X 763 with binary value 0 or 1).

processing was carried out in MATLAB. Thus, the final gray-scale image that we are trying to reconstruct is shown in fig 4. The image matrix is of size 1030 X 763 with gray-scale value ranging from 0 to 255 (only integer values). We flatten the image matrix and thus the output matrix is of dimension 785890 X 1.

B. Input Data Generation

To generate the input data we started with the original RGB image as a starting point for image processing in MATLAB (Image processing toolbox). The default image segmentation algorithm in MATLAB was used to first convert a cropped RGB image to gray-scale and then background and parts of spider web structure was removed. Then the whole image was converted to a binary image with size 1030 X 763 with values in the image matrix be either 0 or 1. The input image is show in 5. We can notice the input data is very challenging. The input data consists of three values,

- X-coordinate
- Y-coordinate
- Value (0 or 1).

The input data matrix dimension is thus 785890 X 3. We know that more precise the data point, better the end result will be. So, we need a robust machine learning algorithm to tackle this problem.

C. Machine Learning Algorithms

1) Support Vector Regression: The support vector regression is implemented in Python 2.7 using the *Scikit-learn* library. The SVR function was employed with input parameters as C=1, kernel = 'rbf', shrinking = False, max_iter=1000000. We have used RBF kernel for this problem. The training time was 7 hours. Finally, to obtain integer value we used the ceiling function which maps x to the least integer, ceiling$(x) = \lceil x \rceil$ that is greater than or equal to x. The final learnt output values are thus integers which is subsequently used to obtain the final reconstructed image.The final reconstructed image is shown in Fig. 6. It can be noticed that SVM is able to reconstruct the image structure very well. We used an Intel Core i7 processor with 4 CPU cores running MacOS at 2.8 GHz with 16 GB RAM to perform the numerical calculations.

2) Multilayer Perceptron: The MLP structure employed for this work is shown below:

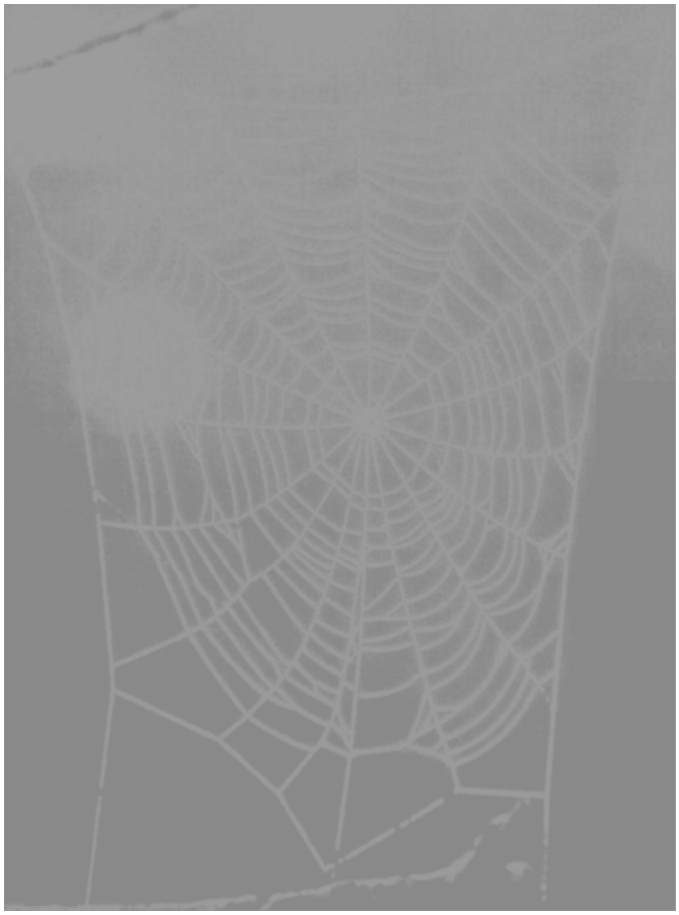

Figure 6. SVM output using RBF kernel (max_iter=1000000).

Figure 7. Multilayer perceptron output (3 hidden layers with 200, 400 and 500 ReLU units, 1 linear unit as output, epochs=25000).

- **Input layer**: 3 units
- **Hidden layer 1**: 200 ReLU units
- **Hidden layer 2**: 400 ReLU units
- **Hidden layer 3**: 500 ReLU units
- **Output layer**: 1 linear unit

ReLU units were used because of its good properties for handling vanishing gradient. The MLP was implemented in Python 2.7 using Keras 2 (with TensorFlow backend and CUDA 8). The back-propagation is coded in TensorFlow library. The input parameters for the model training and prediction are given as:

1) Optimizer: Adam
2) Loss: Mean squared error
3) Metrics: Accuracy
4) Epochs: 25000
5) Batch size: 1000
6) Prediction batch size: 128.

1st epoch loss was 4867.2906 with accuracy 0.0061. 25000th epoch loss was 417.6152 with accuracy as 0.0334. The losses for last 10 epochs were: 418.72, 417.43, 423.30, 426.08, 415.29, 410.28, 416.61, 423.66, 417.52 and 417.61. So, we can observe that we have trained the model well enough. The training time was 40.3 hours using 4 GB GPU. Finally, to obtain integer output values we used the ceiling function. The final learnt output values are thus integers which is subsequently used to obtain the final reconstructed image. The final reconstructed image is shown in Fig. 7. We used an Amazon AWS server g2.2xlarge (8 virtual CPU, 15 GB RAM) with NVIDIA GPU to perform the numerical calculations.

IV. Conclusion

We have applied both SVM and MLP for an image reconstruction problem. Both the learning algorithms performed very well. Even though the input data was very challenging, the reconstructed image resembles very closely to the expected output image.

V. Future Work

Immediate future work involves tuning the SVM learning parameters to further optimize its performance. Additionally, the more complex MLP structures can be explored. It would interesting to see how these algorithms would perform when the RGB image is to be reconstructed instead of the gray-scale one.

REFERENCES

[1] Haykin, Simon S. *Neural networks and learning machines*, Vol. 3. Upper Saddle River, NJ, USA:: Pearson, 2009.

[2] Goodfellow, I., Bengio, Y. and Courville, A. *Deep learning*. MIT press, 2016.

[3] Graves, Alex. *Supervised sequence labelling with recurrent neural networks*, Vol. 385. Heidelberg: Springer, 2012.