CSE 574: Introduction to Machine Learning (Fall 2018)
Project 3: Classification

Objective
This project is to implement machine learning methods for the task of classification. The classification task will be that of recognizing a 28_28 grayscale handwritten digit image and identify it as a digit among 0, 1, 2, ... , 9. You are required to train the following four classifiers using MNIST digit images.

1. Logistic regression, which you implement yourself using back-propagation and tune hyperparameters.
2. A publicly available multilayer perceptron neural network, train it on the MNIST digit images and tune hyperparameters.
3. A publicly available Random Forest package, train it on the MNIST digit images and tune hyperparameters.
4. A publicly available SVM package, train it on the MNIST digit images and tune hyperparameters.

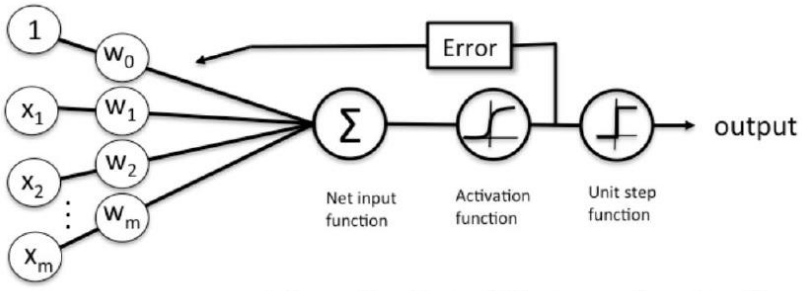
Tasks performed for project implementation:
1. Data Processing

MNIST: The original black and white (bilevel) images from MNIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

USPS: The images are resized to 28x28 like MNIST digits and fed into the trained model.

2. Algorithms implemented:

Logistic Regression:



Schematic of a logistic regression classifier.

Logistic regression is implemented as a one layer neural network which does not include any hidden layers in between. Simple implementation of logistic regression involves multiplying the inputs with weights and passing the result through Activation function, preferably sigmoid to obtain the output. Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. Logistic regression predictions are discrete

Arooshi Avasthy arooshia@buffalo.edu

values unlike Linear regression which gives out continuous values. To squash the predicted value between 0 and 1, we use the sigmoid function.

$$z = \theta_0 + \theta_1 \cdot x_1 + \theta \cdot x_2 + \cdots \qquad g(x) = \frac{1}{1+e^{-x}}$$

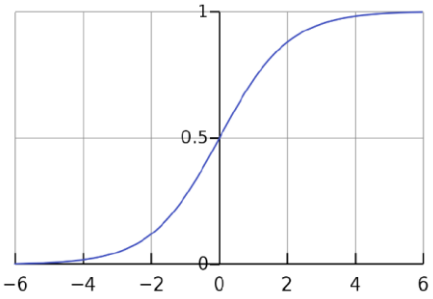
Linear Equation and Sigmoid Function

We take the output(z) of the linear equation and give to the function g(x) which returns a squashed value h, the value h will lie in the range of 0 to 1.

$$h = g(z) = \frac{1}{1+e^{-z}}$$

Squashed output-h

Graph of



Sigmoid Function graph

The graph shows that the sigmoid function becomes asymptote to y=1 for positive values of x and becomes asymptote to y=0 for negative values of x.

Cost Function

We do not use ERMS for calculating cost like incase of linear regression because we are trying to predict class values. Therefore, we use a logarithmic loss function to calculate the cost for misclassifying.

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

The above cost function can be rewritten as below since calculating gradients from the above equation is difficult.

$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Calculating Gradients

Take partial derivatives of the cost function with respect to each parameter(theta_0, theta_1, …) to obtain the gradients. Using these gradients, update the values of theta_0, theta_1, …

Arooshi Avasthy arooshia@buffalo.edu

$$J = \frac{-1}{m} \cdot [\sum_{i=1}^{m} y_i \cdot \log h_i + (1 - y_i) \cdot \log 1 - h_i]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot [\sum_{i=1}^{m} \frac{y_i}{h_i} \cdot h_i^2 \cdot x_n \cdot \frac{1 - h_i}{h_i} + \frac{1 - y_i}{1 - h_i} \cdot -h_i^2 \cdot x_n \cdot \frac{1 - h_i}{h_i}]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot [\sum_{i=1}^{m} x_n \cdot (1 - h_i) \cdot y_i - x_n \cdot h_i \cdot (1 - y_i)]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{1}{m} \cdot x_i \cdot [\sum_{i=1}^{m} h_i - y_i]$$

Gradients

Experiments with the model:

Experiments:

Learning Rate= 0.05

Number of Epochs = 1000

Batch Size = 128

MNIST Precision: 89.11
USPS Precision: 24.28

```
Confusion Matrix MNIST
[[ 926    0    3    9    1   22   18    2    7    3]
 [   0 1014    2   11    0    4    3    6   22    2]
 [   5   24  746   70    6   14   32   22   69    2]
 [   1    1    6  952    1   32    3    5   19   10]
 [   5   15    5    3  847    3   29   13    7   56]
 [   9    2    4   46    4  802   22    5   20    1]
 [   5    2    2    0    3   18  933    0    4    0]
 [   6    4    1   17    4    2    0 1024    3   29]
 [   0   13    2   46    1   69   12   14  833   19]
 [   4    6    0   12   15   23    2   59    6  834]]
```

```
Confusion Matrix USPS
[[317   6 367 320 108 261 169 318  21 113]
 [ 49 118 552 184 159 228 103 367 161  79]
 [114  78 848 160  24 365 237  72  69  32]
 [ 39  69 396 506  12 720  45  87  86  40]
 [ 57  56 211  57 569 249 119 429 158  95]
 [ 59  42 552 146  29 935 104  87  33  13]
 [144  30 615 107  24 510 466  66  24  14]
 [104  76 100 455  95 136  22 697 249  66]
 [155  51 170 474 145 510 110 153 182  50]
 [ 38  69 103 404 127 173  27 685 156 218]]
```

Here the high values in the diagonal matrix for MNIST indicates that the true value and predicted value match is high. Whereas, incase of USPS all other elements apart from the diagonal also have high values, this shows the inaccuracy that exists.

Neural Network:

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information.

An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process.

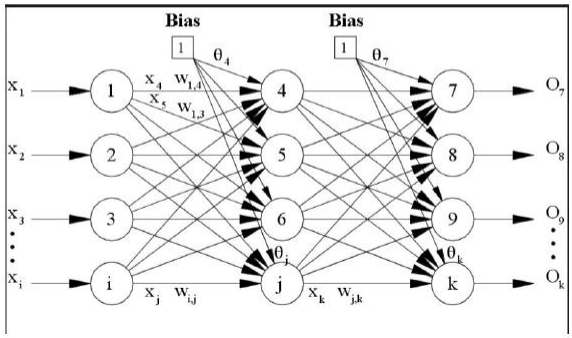
When a signal(value) arrives, it gets multiplied by a weight value. If a neuron has 4 inputs, it has 4 weight values which can be adjusted during training time.

Genesis Equation:

$$z = x_1 * w_1 + x_2 * w_2 + \ldots\ldots + x_n * w_n + b * 1$$

$$\hat{y} = a_{out} = sigmoid(z)$$

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$



Forward propagation is a process of feeding input values to the neural network and getting an output which we call predicted value. When input values are fed to the neural network's first layer, it goes without any operations. Second layer takes values from first layer and applies multiplication, addition and activation operations and passes this value to the next layer. Same process repeats for subsequent layers and finally we get an output value from the last layer.

After forward propagation, the predicted value is obtained. To calculate error the predicted value is compared with the actual output value. Loss function is used to calculate the error value. Then the derivative of the error value is calculated with respect to each and every weight in the neural network. Back-Propagation uses chain rule of Differential Calculus. In chain rule first the derivatives of error value is calculated with respect to the weight values of the last layer. These derivatives, gradients and use these gradient values to calculate the gradients of the second last layer. This process is repeated until gradients for each and every weight is obtained. This gradient value is then subtracted from the weight value to reduce the error value. This way Local Minima(means minimum loss) is obtained.

Experiments:

Epochs: 2000

Activation Function: Softmax

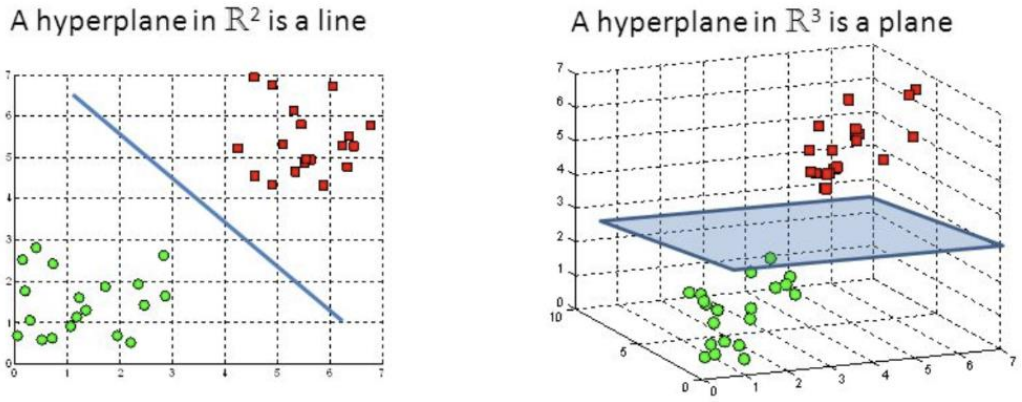
MNIST Dataset: accuracy: 0.7553 loss: 0.632573749542

USPS Dataset: accuracy: 0.325466273305 loss: 2.4758614878

Batch Size: 250

SVM:

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane.



Hyperplanes in 2D and 3D feature space

In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases} \qquad c(x, y, f(x)) = (1 - y * f(x))$$

After adding the regularization parameter, the cost functions looks as below.

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)$$

Loss function for SVM

Take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

Arooshi Avasthy arooshia@buffalo.edu

$$\frac{\delta}{\delta w_k}\lambda \parallel w \parallel^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k}\left(1 - y_i\langle x_i, w\rangle\right)_+ = \begin{cases} 0, & \text{if } y_i\langle x_i, w\rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Gradients

Experiments with the model:

SVC(kernel='poly', C=3, gamma =0.05)

SVC(kernel='linear', C=3);

kernel : *string, optional (default='rbf')*
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable

C : *float, optional (default=1.0)*
Penalty parameter C of the error term.

gamma : *float, optional (default='auto')*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

MNIST Precision: 0.98
USPS Precision: 0.01

Confusion Matrix USPS:

```
Confusion Matrix USPS
[[ 113  154   56  101  269   12  103 1143    0   49]
 [   3  244   23   20   14    0   17 1594    0   85]
 [ 116  595   40   20   30   19   49  974    2  154]
 [ 145  453   15   77  201   45   27  772    7  258]
 [  10  269   33   72  405  125   10  982    3   91]
 [ 145  381   19  128  305   46   78  842    1   55]
 [ 165  420   36  204  136   40   55  865    1   78]
 [  52  246   42   26   51    8   44 1489    0   42]
 [ 100  444   70  191  282   66   79  705    3   60]
 [  22  266   16   65  327   29   48 1151    0   76]]
```

Confusion Matrix MNIST:

```
Confusion Matrix MNIST
[[ 966    0    4    3    1    2    3    0    9    3]
 [   0 1042    6    4    0    1    3    4    3    1]
 [  11    4  929   11    6    3    4   13    7    2]
 [   7    7   23  940    3   25    1    7   11    6]
 [   6    5   11    2  922    4    5    2    8   18]
 [  19    5   10   44    6  799   13    0   12    7]
 [   6    7    4    2   11    8  923    0    5    1]
 [   3    9    9   11    9    0    2 1033    2   12]
 [  14   13   22   36   13   25    4    6  861   15]
 [  10    4    6   17   37   18    1   12   11  845]]
```

Arooshi Avasthy arooshia@buffalo.edu

Random forest:

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Decision Trees
Decision trees are simple but intuitive models that utilize a top-down approach in which the root node creates binary splits until a certain criteria is met. This binary splitting of nodes provides a predicted value based on the interior nodes leading to the terminal (final) nodes. In a classification context, a decision tree will output a predicted target class for each terminal node produced.

Bootstrap Aggregating Trees
Through a process known as bootstrap aggregating (or bagging), it's possible to create an ensemble (forest) of trees where multiple training sets are generated with replacement
This approach helps reduce variance by averaging the ensemble's results, creating a majority-votes model.

Experiments:

```
classifier1 = RandomForestClassifier(class_weight=None, criterion='gini',
        max_depth=2, max_features='auto', max_leaf_nodes=None,n_estimators=3);

classifier1 = RandomForestClassifier(n_estimators=5);
```

estimators: ***list of DecisionTreeClassifier***
The collection of fitted sub-estimators.
criterion : ***string, optional (default="gini")***
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

MNIST Precision: 0.93
USPS Precision: 0.12

```
Confusion Matrix USPS
[[ 113  154   56  101  269   12  103 1143    0   49]
 [   3  244   23   20   14    0   17 1594    0   85]
 [ 116  595   40   20   30   19   49  974    2  154]
 [ 145  453   15   77  201   45   27  772    7  258]
 [  10  269   33   72  405  125   10  982    3   91]
 [ 145  381   19  128  305   46   78  842    1   55]
 [ 165  420   36  204  136   40   55  865    1   78]
 [  52  246   42   26   51    8   44 1489    0   42]
 [ 100  444   70  191  282   66   79  705    3   60]
 [  22  266   16   65  327   29   48 1151    0   76]]
```

```
Confusion Matrix MNIST
[[ 966    0    4    3    1    2    3    0    9    3]
 [   0 1042    6    4    0    1    3    4    3    1]
 [  11    4  929   11    6    3    4   13    7    2]
 [   7    7   23  940    3   25    1    7   11    6]
 [   6    5   11    2  922    4    5    2    8   18]
 [  19    5   10   44    6  799   13    0   12    7]
 [   6    7    4    2   11    8  923    0    5    1]
 [   3    9    9   11    9    0    2 1033    2   12]
 [  14   13   22   36   13   25    4    6  861   15]
 [  10    4    6   17   37   18    1   12   11  845]]
```

Arooshi Avasthy arooshia@buffalo.edu

Ensemble Learning – Majority Voting

A collection of several models working together on a single set is called an Ensemble. The method is called Ensemble Learning.

Trained the model using diverse algorithms such Logistic regression, SVM, random forest and, neural network and then ensembled them to predict the final output. All models are pitted against each other and selected upon best performance by voting.

Hard voting is where a model is selected from an ensemble to make the final prediction by a simple majority vote for accuracy.

Soft Voting can only be done when all your classifiers can calculate probabilities for the outcomes. Soft voting arrives at the best result by averaging out the probabilities calculated by individual algorithms.

Analysis of the models

1. We test the MNIST trained models on two different test sets: the test set from MNIST and a test set from the USPS data set. Do your results support the "No Free Lunch" theorem?

Ans) The "No Free Lunch" theorem states that there is no one model that works best for every problem. The assumptions of a great model for one problem may not hold for another problem, so it is common in machine learning to try multiple models and find one that works best for a particular problem. This is especially true in supervised learning; validation or cross-validation is commonly used to assess the predictive accuracies of multiple models of varying complexity to find the best model.

Experiments conducted on MNIST and USPS datasets support No free Lunch theorem. It shows how different models trained on MNIST dataset give high training and testing accuracy with MNIST dataset, however the accuracy is really low when model which is trained on MNIST dataset performs testing on USPS dataset.

2. Observe the confusion matrix of each classifier and describe the relative strengths/weaknesses of each classifier. Which classifier has the overall best performance?

Ans) Models implemented and their strengths/weaknesses:

- Logistic regression

  Strengths: Outputs have a nice probabilistic interpretation, and the algorithm can be regularized to avoid overfitting. Logistic models can be updated easily with new data using stochastic gradient descent.
  Weaknesses: Logistic regression tends to underperform when there are multiple or non-linear decision boundaries. They are not flexible enough to naturally capture more complex relationships.

- SVM
  Strengths: SVM's can model non-linear decision boundaries, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space.

Weaknesses: However, SVM's are memory intensive, trickier to tune due to the importance of picking the right kernel, and don't scale well to larger datasets.

- Random forest
  Strengths: Decision trees can learn non-linear relationships, and are fairly robust to outliers. Ensembles perform very well in practice, winning many classical (i.e. non-deep-learning) machine learning competitions.
  Weaknesses: Unconstrained, individual trees are prone to overfitting because they can keep branching until they memorize the training data.

- Neural Network
  Strengths: Corruption of one or more cells of ANN does not prevent it from generating output. This feature makes the networks fault tolerant. The loss of performance here depends on the importance of the missing information.
  Weakness: There is no specific rule for determining the structure of artificial neural networks. Appropriate network structure is achieved through experience and trial and error.

According to the experiments performed, SVM gives the best accuracy of 0.98. But we cannot consider SVM to best among the given models because accuracy for each model depends on the size of dataset and selected parameters for each model and tuning of those parameters.

3. Combine the results of the individual classifiers using a classifier combination method such as majority voting. Is the overall combined performance better than that of any individual classifier?

Ans) The max voting method is used for classification problems. Multiple models- logistic regression, neural network, SVM and random forests are used to make predictions for each data point. The predictions by each model are considered as a 'vote'. The predictions with majority votes of the models are used as the final prediction. The experiment conducted included multiple models and the result accuracy is 0.21 for USPS dataset which is higher than accuracy for few models individually.