IE7860: Intelligent Engineering Systems

Assignment 4 : Support Vector Machines

# Non-Separable Classification – Pattern Recognition in Process Control System

## Overview

Then problem is to check if a process is out of statistical control, and for what reason. The data set had 10 inputs and 7 outputs being the type of error. There were 2500 data points for the training file and 250 data points for the test data. 1000 patterns representing a process in control and 250 patterns representing each of the six rule violations for training and 100 patterns representing a process in control and 25 patterns representing each of the six rule violations for test data.

A picture containing screenshot

Description automatically generated

To solve such a problem we implement a SVM(Support Vector Machines) such a model a can be built on python using Keras and TensorFlow as backend. Both of these libraries are the industry standards for deep learning along with Pytorch.

## Data Preprocessing

Then dataset is clean but require certain level of pre-processing. We Start by loading the data from the text file into the np arrays and then start to normalize and flatten the image. Before which we shape the data based on the input and output parameters. This is done so as to shape the data for the model to process. We can observe that the training data has 2500 rows or examples. While the input data has 10 elements of these 2500 samples, the output has 7 elements of these same 2500 samples. 20% of the training dataset is split into validation dataset.

The validation set is used during the model fitting to evaluate the loss and metrics. However, the model is not fit with this data. The test set is completely unused during the training phase and is only used at the end to evaluate how well the model generalizes to new data. This is especially important with imbalanced datasets where overfitting is a significant concern from the lack of training data.

One key observation was regarding the removal of validation actually lead to a very strong model. Upon experimenting to find the reason, it was discovered that the validation set was reducing the number of data points for certain classes and hence not able to train those classes.

A close up of a piece of paper

Description automatically generatedA close up of a map

Description automatically generated

## Creating the Model

The model definition is just a few lines of code but has all the parameters required. The backend and all the complications are handled by Keras and Tensorflow. The weights of artificial neural networks must be initialized to small random numbers. This is because this is an expectation of the stochastic optimization algorithm used to train the model, called stochastic gradient descent. The algorithm uses randomness in order to find a good enough set of weights for the specific mapping function from inputs to outputs in your data that is being learned. It means that your specific network on your specific training data will fit a different network with a different model skill each time the training algorithm is run. The activation function is ReLU in input and and hif=dden nodes, while Softmax is used for the output node. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0. Softmax normalizes your data (outputs a proper probability distribution) and is differentiable hence is a very good choice for output nodes. Categorical crossentropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only. The block before the Target block must use the activation function ​Softmax. Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. The model gave relatively strong results after tuning the hyper parameters manually. For 10, 512, 256 and 7 nodes in input, hidden layer 1, hidden layer 2 and output layer respectively. It was later optimized by evaluating the model.

Since most of these features can be defined directly using SVC method in “from sklearn.svm”. Hence the model passed is simply as SVC() for Grid Search to get the new parameters. Predict() function is used to run the model on training data. The object is the model, and x and y are our training data in matrix and numeric vector forms, respectively. We set different epochs to control the number training cycles. Typically, we want epochs to be reasonably large, which is important in visualizing the training history. The training process should complete in certain time usually in few seconds based on the number of epochs defined. Training occurs over epochs; an epoch is one pass through all of the rows in the training dataset

## Evaluating the Model

Starting by getting the confusion matrix for the entire data actual values and predicted values. The initial confusion matrix was showing low correlation between actual and predicted parameters since initially the model has unoptimized parameters. The confusion matrix for the best output is given below. The model was then evaluated by implementing Stratified Shuffle and F1 tests. Since Stratified Shuffle has the exclusivity rule. All the subsets are exclusive

A close up of a piece of paper

Description automatically generated

Using the Classification Report, we can analyze the training and at stages. Based in which we can conclude that the accuracy achieved is 92%. Which compared to 89% on MLP and the change in parameters to {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}. This gave me New Training Score = 0.9132 and New Testing Score = 0.916.

A screenshot of a cell phone

Description automatically generated

# Non-Separable Classification – Multi Font Character Recognition

## Overview

The network will be trained on upper-case English letters in selected fonts. The data that will be used for this assignment, consisting of 6 fonts (Courier, New York, Chicago, Geneva, Times, and Venice).

1. The image of the letter is normalized to an 18 x 18 character matrix, where the line thickness is one and the image is represented by 0's (background) and 1's (foreground).

2. Fourteen properties similar to those proposed by Fuji and Morita (see reference) were extracted from each image. Each property is a 3 x 3 matrix, thus, for each image, a 14 x 9 matrix is generated. This is the X matrix for that image. A property recognition matrix, Y, is constructed for each image and is also a 14 x 9 matrix. It is chosen arbitrarily and is as simple as possible. W is a 9 x 9 filter matrix which maps X to Y and can be found from: W = X\* Y, where X\* the pseudo-inverse of X.

3. A 3 x 3 window is moved from upper left to lower right over the character image. The 9 elements in the window are multiplied by the matrix W. If the output matches a row of Y, say row k, the kth place in the count matrix is incremented by the weighting factor of that property.

Thus, the count matrix for a character contains the number of exact template matches, weighted by position. The result is 156 (26 x 6) 14-element vectors.

## Data Preprocessing

Then dataset is clean but require certain level of pre-processing. We Start by loading the data from the text file into the np arrays and then start to normalize and flatten the image. Before which we shape the data based on the input and output parameters. This is done so as to shape the data for the model to process. We can observe that the training data has rows or examples. While the input data has 14 elements of these 78 samples, the output has 26 elements of these same 78 samples. 20% of the training dataset is split into validation dataset.

The validation set is used during the model fitting to evaluate the loss and metrics. However, the model is not fit with this data. The test set is completely unused during the training phase and is only used at the end to evaluate how well the model generalizes to new data. This is especially important with imbalanced datasets where overfitting is a significant concern from the lack of training data.

The dataset is very small hence the earlier mentioned problem of reduction of the accuracy and metrics due to the addition of validation in Shuffle. Since Stratified Shuffle has the exclusivity rule for each class to increase diversity and variation inter subsets.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA close up of a white wall

Description automatically generated

## Creating the Model

The model definition is just a few lines of code but has all the parameters required. The backend and all the complications are handled by Keras and Tensorflow. The weights of artificial neural networks must be initialized to small random numbers. This is because this is an expectation of the stochastic optimization algorithm used to train the model, called stochastic gradient descent. The algorithm uses randomness in order to find a good enough set of weights for the specific mapping function from inputs to outputs in your data that is being learned. It means that your specific network on your specific training data will fit a different network with a different model skill each time the training algorithm is run. The activation function is ReLU in input and and hif=dden nodes, while Softmax is used for the output node. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0. Softmax normalizes your data (outputs a proper probability distribution) and is differentiable hence is a very good choice for output nodes. Categorical crossentropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only. The block before the Target block must use the activation function ​Softmax. Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. The model gave relatively strong results after tuning the hyper parameters manually. For 10, 512, 256 and 7 nodes in input, hidden layer 1, hidden layer 2 and output layer respectively. It was later optimized by evaluating the model.

Since most of these features can be defined directly using SVC method in “from sklearn.svm”. Hence the model passed is simply as SVC() for Grid Search to get the new parameters. Predict() function is used to run the model on training data. The object is the model, and x and y are our training data in matrix and numeric vector forms, respectively. We set different epochs to control the number training cycles. Typically, we want epochs to be reasonably large, which is important in visualizing the training history. The training process should complete in certain time usually in few seconds based on the number of epochs defined. Training occurs over epochs; an epoch is one pass through all of the rows in the training dataset

A close up of a device

Description automatically generatedA close up of a device

Description automatically generatedA close up of a mans face

Description automatically generatedA close up of a mans face

Description automatically generated

## Evaluating the Model

Evaluation of the Model begins with the confusion matrix and ends within the f1 test and subsets to optimize the hyperparameters is used. The confusion matrix given above is for the best solution and shows high correlation among the actual and predicted values. Except for class 0,1 and 14. This is due to lack of training data for the classes. Especially since applying Shuffle will reduce the size even further. So as to include all classes. Stratified Shuffle ensures that in the backend.

We first check the loss and accuracy for training, validation and test dataset. We then produce plots of model's accuracy and loss on the training and validation set. They are useful to check for overfitting. By the following result we can conclude that the model is relatively strong and there is no overtraining.

A screenshot of a cell phone

Description automatically generated

Using the Classification Report, we can analyze the training and at stages. Based in which we can conclude that the accuracy achieved is 82%. Which compared to MLP, it is relatively high due to the change in parameters to {'C': 10, 'gamma': 1, 'kernel': 'rbf'}. This gave me New Training Score = 0.9103 and New Testing Score = 0.8334.

A close up of a piece of paper

Description automatically generated

# Process Modeling – Reactive ion Plasma Etching Process

## Overview

Of the nearly dozen different factors that have been shown to influence plasma etch behavior in the literature, the study focused on the following parameters regarded as most critical: chamber pressure, RF power, electrode spacing, and the gas flow rate of CCl4. The primary etchant gas is CCl4, but He and O2 are added to the mixture to enhance uniformity and reduce polymer deposition in the process chamber, respectively.

A picture containing object

Description automatically generated

The experiments were conducted by May, Huang, and Spanos in two phases at the Berkeley Microfabrication Laboratory. In the first phase a 26-1 fractional factorial design requiring 32 runs was performed to reduce the experimental budget. Experimental runs were performed in two blocks of 16 trials each in such a way that no main effects or first order interactions were confounded. Three center points were also added to check the model for nonlinearity. Analysis of the first stage of the experiment revealed significant nonlinearity, and showed that all six factors are significant

## Data Preprocessing

The data here is very linear and not a classification problem, it contains 6 input parameters and 4 output parameters. But they are all continuous values and quantitative.

A close up of a logo

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedThe following table is basic statistics of the entire data and includes values like mean and standard deviation along with quartiles. This is used to check if the data is normal or has high variance and intensities.

A screenshot of a cell phone

Description automatically generatedThe data is complete and normal, since there were more features to explore in pandas for the deep learning I choose to do this dataset with pandas instead of np array and had the converted and along with shaping and defining the structure of the data for further labelling of the graphs and plots.

## Creating the Model

The number of neurons in a hidden layer must identify efficiently to get a better performance in the model. Therefore, the network trained for different number of neurons in hidden layer. The best performance was identified when use 10 number of neurons in a single hidden layer.

The weights of artificial neural networks must be initialized to small random numbers. This is because this is an expectation of the stochastic optimization algorithm used to train the model, called stochastic gradient descent. The algorithm uses randomness in order to find a good enough set of weights for the specific mapping function from inputs to outputs in your data that is being learned. It means that your specific network on your specific training data will fit a different network with a different model skill each time the training algorithm is run. The activation function is ReLU in input and and hidden nodes, while Softmax is used for the output node. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0. Softmax normalizes your data (outputs a proper probability distribution) and is differentiable hence is a very good choice for output nodes. Categorical crossentropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only. The block before the Target block must use the activation function ​Softmax. Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. The model gave relatively strong results after tuning the hyper parameters manually.

A close up of a logo

Description automatically generatedA close up of a map

Description automatically generatedA screenshot of a cell phone

Description automatically generated

The above Plots Highlight the normality and the distribution of the data. The third image shows the shape of the data and the later image shows the correlation of the actual and predicted parameters.

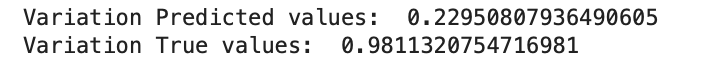
A screenshot of a cell phone

Description automatically generated

Looking at the correlation matrix we can understand the that the correlation between the actual and predicted parameters Is high for the current model and hence is very good for the data set. The above correlation is for the chosen best solution.

## Evaluating the Model

Evaluation of the Model begins with the confusion matrix and ends within the f1 test and folds to optimize the hyperparameters is used. The confusion matrix given above is for the best solution and shows high correlation among the actual and predicted values. All the data has high correlation in the confusion matrix. Due to the nature of data being linear and classification evaluation I smore important towards the variance of the original y to the predicted y. Hence I separated each output variable and calculated the respective variance, comparing the original and predicted values for each output variable. The overall Variance from the model is given below, where we can see that the variance of the predicted values I slower compared to the true values. The second image highlights the variance of each of the output variables which is pretty high individually, till me merge them all together.



A screenshot of a cell phone

Description automatically generated