Landsat Satellite Dataset Analysis

By:-
Aditya Priya (2005004)
Tushar kant Behera(2005626)
Yash Anand (20051769)

ABSTRACT:

Multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood

1. INTRODUCTION:

Attributes extracted from Landsat program has been a critical tool for monitoring and understanding changes in the Earth's surface over time, and it continues to provide valuable data for a wide range of applications and research areas.

Real world use-case :

One real-world use case of the Landsat Program is its use in monitoring deforestation in the Amazon rainforest. Deforestation is a major environmental issue in the Amazon, and Landsat data has been used to track changes in forest cover over time and to identify areas of deforestation.

For example, researchers have used Landsat data to analyze deforestation patterns in the Brazilian Amazon from 2000 to 2019, finding that deforestation rates have increased in recent years due to factors such as agricultural expansion, mining, and logging. The data has

also been used to identify areas of illegal deforestation and to support conservation efforts in the region.

In addition to monitoring deforestation, Landsat data has been used for a range of other environmental applications in the Amazon and other regions, including monitoring forest fires, tracking changes in water resources, and supporting ecosystem management and conservation efforts. Overall, the Landsat program has been an important tool for understanding and addressing environmental challenges around the world.

METHODOLOGY:

The database consists of the multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood. The aim is to predict this classification, given the multi-spectral values. In the sample database, the class of a pixel is coded as a number.

The Landsat satellite data is one of the many sources of information available for a scene. The interpretation of a scene by integrating spatial data of diverse types and resolutions including multispectral and radar data, maps indicating topography, land use etc. is expected to assume significant importance with the onset of an era characterised by integrative approaches to remote sensing (for example, NASA's Earth Observing System commencing this decade). Existing statistical methods are ill-equipped for handling such diverse data types. Note that this is not true for Landsat MSS data considered in isolation (as in this sample database). This data satisfies the important requirements of being numerical and at a single resolution, and standard maximum-likelihood classification performs very well. Consequently, for this data, it should be interesting to compare the performance

of other methods against the statistical approach.

One frame of Landsat MSS imagery consists of four digital images of the same scene in different spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infra-red. Each pixel is a 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution of a pixel is about 80m x 80m. Each image contains 2340 x 3380 such pixels.

The database is a (tiny) sub-area of a scene, consisting of 82 x 100 pixels. Each line of data corresponds to a 3x3 square neighbourhood of pixels completely contained within the 82x100 sub-area. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighbourhood and a number indicating the classification label of the central pixel. The number is a code for the following classes:

Number Class:
1 red soil
2 cotton crop
3 grey soil
4 damp grey soil
5 soil with vegetation stubble
6 mixture class (all types present)
7 very damp grey soil

SUPPORT VECTOR MACHINE(SVM)

SVM (Support Vector Machine) module provides an implementation of the SVM algorithm for classification and regression tasks. SVM is a powerful and versatile machine learning algorithm that can be used for both linear and non-linear classification and regression tasks.
Here's a brief overview of how SVM works:

- In SVM, the goal is to find a hyperplane that separates the data points into different classes in the best possible way. The hyperplane is defined as the line that maximizes the margin (i.e., the distance between the hyperplane and the closest data points).
- If the data cannot be linearly separated by a hyperplane, SVM uses a technique called kernel trick to map the data into a higher-dimensional space where a hyperplane can be found. This allows SVM to handle non-linear classification tasks.
- SVM can also handle multi-class classification tasks by using techniques such as one-vs-one or one-vs-all.

Scikit-learn's SVM module provides a variety of options and parameters for customizing the SVM algorithm, including the choice of kernel function (e.g., linear, polynomial, radial basis function), the value of the regularization parameter, and the type of SVM (e.g., C-SVC, nu-SVC, SVR, nu-SVR).
To use SVM in scikit-learn, you typically start by creating an SVM object (e.g., svm.SVC() for classification or svm.SVR() for regression) and then fitting the model to your training data using the fit() method. Once the model has been trained, you can use it to make predictions on new data using the predict() method. Scikit-learn also provides a variety of evaluation metrics (e.g., accuracy, precision, recall, F1 score) for assessing the performance of the SVM model.

CONVOLUTIONAL NEURAL NETWORK (CNN)

Traditional machine learning methods often entail processes of manually defining features, which may not necessarily be optimal representations for problems at hand. The tasks tackled using the deep CNN approach .However, deep neural networks for image recognition, including CNN, can assume

minimally processed input and find optimal network configuration through a training procedure.
It is widely used for face recognition. Convolutional Neural Networks are very similar to ordinary neural networks but are made up of neurons that have learnable weights and biases. ConvNets perform better than the other deep neural network architectures because of their unique process. Instead of looking at the image one pixel at a time, ConvNets group several pixels together so they can understand a temporal pattern.
Over fitting is a common problem while using machine learning based methods on small image collections [S. Chen, C. Zhang, M. Dong, J. Le and M. Rao, "Using ranking-cnn for age estimation"]. This problem is intensified when considering deep convolutional neural networks due to large parameters. So we have to be very careful while using such methods.

It is a kind of artificial neural network but it passes the image through various stages before sending it to hidden layers.
So what are these new stages before hidden layers?and what do they do?They are there to reduce the no of computations that are done in a ANN without sacrificing the accuracy of the model.First the images undergoes convolutional operation with a filter or kernel of size
3 X 3. These operations helps us to find the horizontal and vertical lines or edges in the image.Combining the results from both the filters will result in all of the lines in an image being highlighted..The values in the kernel of size 3 X 3 are determined by back propagation which is done while training the model.

Then we pass the output of this convolutional layer to a max pooling layer which chooses the maximum pixel from the image

using a 2 X 2 matrix. These 2 layers makes one convolutional layer which is then passed to another 3 convolutional layers.The no of filters increases in each layer from 32 to 64 to 128 to 256.
At the end of these 4 layers we get a multi dimensional array which is to be flattened so that it can be used in the hidden layers or dense layers for further computations.

The architectures for age classification and gender classification differ in the fact that they have 3 & 2 blocks with 256 filters respectively (in convolutional layer) and the output layer has 5 and 2 neurons respectively with softmax activation function (being classification tasks).We assign an activation function to each layer.The purpose of introducing activation function is to give neural network nonlinear expression ability, so that it can better fit the results, so as to improve the accuracy.The activation function used in hidden layers is typically chosen based on the type of neural network architecture.So, in the case of CNN the activation function is the Relu activation function .

The next layer is the dropout layer.There are two dropout layers in our model.The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others.Dropout layers are important in training CNNs because they prevent over fitting on the training data. If they aren't present, the first batch of training samples influences the learning in a disproportionately high manner. This, in turn, would prevent the learning of features that appear only in later samples or batches.

There are two output layers, one for gender and the other for age, whose input is the output of the dropout layer.For gender

the activation function is Sigmoid and For age the activation function is relu.
We use two loss functions for our model.For gender we use binary-cross entropy and for age we use mean absolute error.

We are using the ‘Adam’ optimizer function.Instead of this, we can also use the ‘stochastic gradient descent’ optimizer function. Below is the diagram of our model.

RANDOM FOREST CLASSIFIER

Random Forest Classifier is a popular ensemble learning method for classification tasks in machine learning. The algorithm combines multiple decision trees, where each tree is trained on a random subset of the training data and features, to make predictions.
Here's how its work:

- Randomly select a subset of the training data.
- Randomly select a subset of the features.
- Build a decision tree based on the selected data and features.
- Repeat steps 1-3 multiple times to create a forest of decision trees.
- To make a prediction for a new data point, each decision tree in the forest predicts the class of the data point, and the class with the most votes becomes the final prediction.

The key advantage of Random Forest Classifier is that it reduces the variance of the model by averaging the predictions of multiple trees, which reduces the risk of overfitting. Additionally, the use of randomly selected subsets of features and data points also helps to prevent overfitting and improve the generalization ability of the model.
Scikit-learn's Random Forest Classifier provides a variety of hyperparameters that can be used to customize the model, including the number of trees in the forest, the maximum

depth of each tree, and the number of features to consider when making each split. The optimal values of these hyperparameters can be found using techniques such as grid search or random search.
To use the Random Forest Classifier in scikit-learn, you typically create a Random Forest object (e.g., RandomForestClassifier()) and then fit the model to your training data using the fit() method. Once the model has been trained, you can use it to make predictions on new data using the predict() method. Scikit-learn also provides a variety of evaluation metrics (e.g., accuracy, precision, recall, F1 score) for assessing the performance of the Random Forest Classifier.

GAUSSIAN NAIVE BAYES

Gaussian Naive Bayes is a probabilistic algorithm that is commonly used for classification problems. It is based on Bayes' theorem, which states that the probability of a hypothesis (in this case, a class label) given the observed evidence (input features) is proportional to the probability of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis.
The "Naive" in Gaussian Naive Bayes refers to the assumption that the input features are independent of each other, given the class label. This means that the probability of each feature can be calculated separately, and then multiplied together to obtain the probability of the entire set of features.
In Gaussian Naive Bayes, it is assumed that the input features follow a Gaussian (normal) distribution. This means that the probability of each feature is calculated using the mean and standard deviation of the feature values for each class label.
To classify a new instance using Gaussian Naive Bayes, the algorithm calculates the probability of each class label given the input features, using Bayes' theorem. The class label with the

highest probability is then assigned as the predicted label for the instance.
Gaussian Naive Bayes is a simple and efficient algorithm, which can perform well on certain types of classification problems. It is particularly useful when there are a large number of input features, as it can handle high-dimensional datasets well. However, it is important to note that the assumption of independence between the input features may not hold in all cases, which can lead to decreased accuracy.

Training & Testing.

For better analysis and testing we apply K-Fold cross-validation. K-Fold cross-validation is a commonly used technique in machine learning to evaluate the performance of a model on a dataset. It is used to split a dataset into k folds, where k is a user-specified parameter, and then trains and evaluates the model k times. In each iteration, k-1 folds are used for training, and the remaining fold is used for testing.
Here's how K-Fold cross-validation works:

1. The dataset is randomly split into k equally sized folds.
2. The model is trained on k-1 folds and tested on the remaining fold.
3. Steps 2 is repeated k times, with each of the k folds used once as the testing data.
4. The evaluation metrics (e.g., accuracy, precision, recall, F1 score) for each iteration are averaged to obtain a single score that represents the model's performance on the dataset.

The advantage of K-Fold cross-validation is that it provides a more reliable estimate of the model's performance than a

single train/test split, as it uses all the available data for training and testing. It also helps to reduce the bias that can arise from the specific choice of training and testing data in a single split. Scikit-learn's KFold module provides an easy way to implement K-Fold cross-validation. You can specify the number of folds (k) and the shuffle option to randomly shuffle the data before splitting. You can then use the split() method to obtain the indices of the training and testing data for each iteration, and train and evaluate the model accordingly.

CONCLUSION:

In conclusion, the analysis of Landsat satellite dataset using different machine learning algorithms provides valuable insights into the application of these methods in remote sensing and land cover classification tasks.
Support Vector Machines (SVM), K-Nearest Neighbors (KNN) Classifier,Random Forest classifier,Gaussian Naive Bayes and Neural Networks have been used to classify land cover types using Landsat imagery. SVM is a powerful and widely used algorithm for classification, as it can handle high-dimensional datasets and non-linear decision boundaries. KNN is a simple and intuitive algorithm that can perform well in certain scenarios, especially when there is a small number of classes. Neural Networks, on the other hand, can learn complex relationships between the input features and output classes, and are particularly useful for large datasets.

K-Fold cross-validation has been used to evaluate the performance of the models, which involves splitting the data into k subsets and performing k iterations of training and testing. This technique provides a more reliable estimate of the model's performance and helps to reduce overfitting.
Overall, the analysis of Landsat satellite dataset using SVM, KNN, Neural Networks, and K-Fold cross-validation demonstrates the effectiveness of these machine learning techniques in land cover classification and remote sensing applications.
As we have observed the SVM algorithm gives us a better prediction i.e of 0.8797 .