**Satellite Image Classification**

This demo illustrates using a neural network as a classifier to identify the different classes in a satellite image from the RGB values of pixel of the image.

**Contents**

* The Problem: Classification of Satellite Image
* [Why Neural Networks?](file:///D:\Program%20Files\MATLAB\R2010a\toolbox\nnet\nndemos\html\crabclassify.html#2)
* [Preparing the Data](file:///D:\Program%20Files\MATLAB\R2010a\toolbox\nnet\nndemos\html\crabclassify.html#3)
* [Building the Neural Network Classifier](file:///D:\Program%20Files\MATLAB\R2010a\toolbox\nnet\nndemos\html\crabclassify.html#6)
* [Testing the Classifier](file:///D:\Program%20Files\MATLAB\R2010a\toolbox\nnet\nndemos\html\crabclassify.html#9)

**The Problem: Classification of** **Satellite Image**

In this demo we attempt to build a classifier that can identify land, water, marsh, urban areas, etc based on the decision profile matrix of the pixels. Three characteristics of a pixel are considered: red band, blue band and green band. The problem on hand is to identify which pixel belong to which class given the extracted values for each of these three bands from the image.

**Why Neural Networks?**

Neural networks have proven themselves as proficient classifiers and are particularly well suited for addressing non-linear problems. Given the non-linear nature of real world phenomena, like satellite image classification, neural networks is certainly a good candidate for solving the problem.

The three characteristics will act as inputs to a neural network and the belonging of pixel to a class will be target. Given an input, which constitutes the three RGB values for a pixel location, the neural network is expected to identify belonging to one of the five classes (land, urban, water, mangrove and forest).

This is achieved by presenting previously recorded inputs to a neural network and then tuning it to produce the desired target outputs. This process is called neural network training.

**Preparing the Data**

Data for classification problems can very often have textual or non-numeric information. In our case, color of the pixel. Neural networks however cannot be trained with non-numeric data. Hence we need to translate the textual data into a numeric form.

There are several ways to translate textual or symbolic data into numeric data. Some of the common symbol translation techniques used are unary encoding, binary encoding and numbering classes. We are going to use unary encoding in this demo to perform symbol translation.

fid = fopen(fullFilename);

C = textscan(fid,'%f%f%f%s','delimiter',','); % Import data

fclose(fid);

The first 3 columns of data represent the RGB values of a pixel. The 4th column represents the class to which it belongs.

The next step is to preprocess the data into a form that can be used with a neural network.

The neural network object in the toolbox expects the samples along columns and its features along rows. Our dataset has its samples along rows and its features along columns. Hence the matrices have to be transposed.

**Building the Neural Network Classifier**

The next step is to create a neural network that will learn to identify the class of pixels.

Since the neural network starts with random initial weights, the results of this demo will differ slightly every time it is run. The random seed is set to avoid this randomness.

rand('seed', 491218382)

A 1-hidden layer feed forward network is created with 20 neurons in the hidden layer.

Now the network is ready to be trained. The samples are automatically divided into training, validation and test sets. The training set is used to teach the network. Training continues as long as the network continues improving on the validation set. The test set provides a completely independent measure of network accuracy.

[net,tr] = train(net,physchars,class);

**Testing the Classifier**

The trained neural network can now be tested with the testing samples. This will give us a sense of how well the network will do when applied to data from the real world.

out = sim(net,testInputs); % Get response from trained network

The network response can now be compared against the desired target response to build the classification matrix which will provides a comprehensive picture of a classifiers performance.

The classification matrix provides a comprehensive picture of the classification performance of the classifier. The ideal classification matrix is the one in which the sum of the diagonal is equal to the number of samples.

It can also be understood in terms of percentages. The following matrix provides the same information as above but in terms of percentages.

cm\_p = (cm ./ N) .\* 100 % classification matrix in percentages

fprintf('Percentage Correct classification : %f%%\n', 100\*(cm(1,1)+cm(2,2))/N);

fprintf('Percentage Incorrect classification : %f%%\n', 100\*(cm(1,2)+cm(2,1))/N);

cm\_p =

45.0000 0

2.5000 52.5000

Percentage Correct classification : 97.500000%

Percentage Incorrect classification : 2.500000%