Itasca Consulting Group

Mortenson ML Project

GitHub: https://github.com/ssinghai6/Itasca_Mortenson_ML

Problem Statement

The present problems consist of a domain with a strip loading. Based on the loading there is the development of velocity fields. The aim is to predict the velocity field and depth of failure based on the input data.

Variables

n_sim: no. of simulation
n_x : discretization along x- direction
n_y : discretization along y - direction
velo : velocity at failure
coh_data: cohesion
fric_data : friction
poly_data: poly

Input

- 4ft_cohesion.numpy: consists of a variation of cohesion along the z-direction. [Dimension - (25,n_sim)]
- 4ft_friction.numpy: consists of a variation of friction along the z-direction. [Dimension - (25,n_sim)]
- 4ft_poly.numpy : consists of a variation of poly along the z-direction. [Dimension - (25,n_sim)]
- 4ft_water_table.numpy : consist of a variation of cohesion along the z-direction. [Dimension - (1,n_sim)]

Output

- 4ft_failure_depths.npy : failure depth [Dimension - (1,n_sim)]
- 4ft_velocity_plots.npy : velocity field [Dimension - (n_x,n_y,n_sim)]

Approach 1: Using ANN for predicting velocity plot

- The input variable will be (cohesion, friction and poly) and the output as the velocity field.
- Each training example will be considered with respect to the node(For Example if there are 825 nodes in the domain and total simulations are n). The number of training examples will be (825*n)
- The dense layer has a RelU activation function since the model is for the prediction. Therefore, the last layer's activation is taken as linear.
- The obtained results show the overfitting with low mean square error (the fine-tuning is performed by considering a different number of hidden layers and some additional dropout layers to avoid overfitting.

Outcome

- The ANN is based on the assumption of linear regression that each training sample will be independent of the others. Therefore, it is difficult to capture the spatial relationship using the ANN approach.
- The problem of overfitting is leading to the loss function (Mean Square Error) reach to a very small-Value. This can be overcome by using batch normalization (Presently the normalizations are just performed for the input and dense layer)
- The variation of input features is only in the Y-direction making it difficult for a model to learn efficiently.

Approach 2: Using CNN for predicting velocity plot

- The input variable will be (cohesion, friction and poly) in the form of the image having each feature in the form of channels. The input will be like (n_sim, n_y, n_x, 3) and the output as the velocity field (n_sim,825).
- The training example will be sample images with 3 channels and sizes (25,33).
- Normalization and batch normalization is performed to get all the variables in the range (0-1)
- The convolution operation is performed, and feature mapping is done using the different kernels followed by max polling and layer flattening.
- Each image is provided a label in the form of column matrix.
- The aim is to predict the whole column (825,1) corresponding to given input in the form of images.

Outcome

- The proposed architecture works fine. The problem is with the model optimization as during the training after some steps the loss function reaches the NaN.
- The loss function and the learning rate is varied to fix the NaN.
- The input images are in the form of stripes as the properties are just varied along the z-direction making it difficult for feature mapping.
- The present work can be extended, and each pixel can be assigned as a label and Segmentation can be performed.

Approach 3: Using CNN for predicting the depth of failure

- The input, in this case, is velocity plots and the output is the depth of failure.
- Normalization is performed to get all the variables in the range (0-1)
- The convolution operation is performed, and feature mapping is done using the different kernels followed by max polling and layer flattening.
- Each image is provided with a label in the form of depth of failure.

Outcome

- The model works fine in predicting the depth of failure.
- It can be made more accurate by adding more kernels and batch normalization.