Mechanical Engineering Department — ME249, Fall 2022
UC Berkeley — Prof. V. Carey

**Project 3. Solar PV Power**  **Due date: November 17, 2022**

You may team up with a partner for this project. Do not share information or results with other groups.

Files to be used:

Part 1
DS3.1.1LowfluxF22
P3pcaExampleF22
P3pcaPlot1F22
DS3.1.2HifluxF22
CodeP3.1.2F22

Part 2
CodeP3.2.1F22
DS3.2.1multiModePerfF22
CodeP3.2.2F22

**Part 2.**
**Introduction**
In part two of this project you will consider a solar PV system comprised of 4 solar panels of the type described in Part 1. They will be installed in a close-spaced rectangular array, but will be wired with switches so it is possible to connected the four in parallel (mode 0), 2x2 in series/parallel (mode 1), or with the four in series (mode 2). As shown in Fig. 2, these changes combine the *V-I* characteristics of the individual modules to produce three very different overall system *V-I* characteristics.

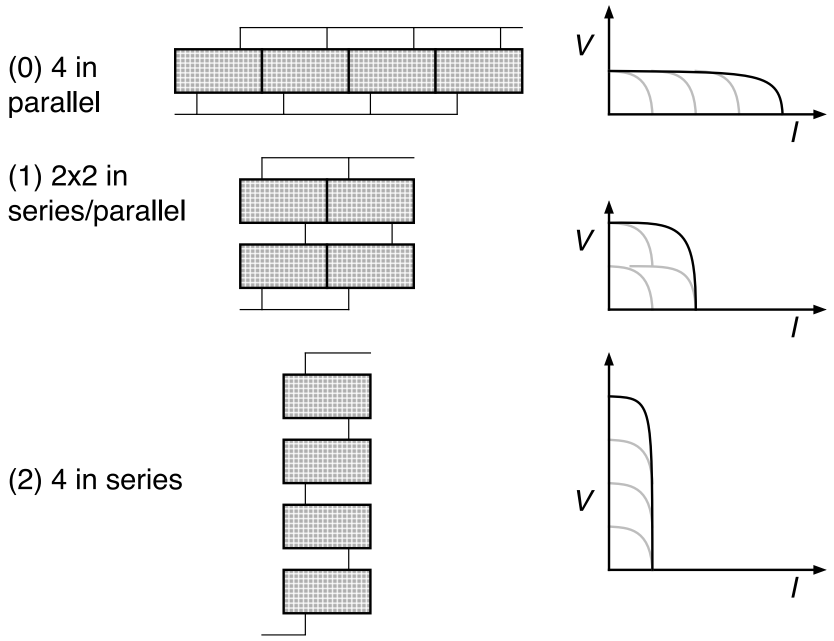

**Figure 2.** Four PV panel system in different modes.

Performance data for the system can be obtained at specified values of the following operating parameters:

Outside air temperature, $T_{air}$ (°C)
Incident direct normal solar radiation intensity, $I_D$ (W/m$^2$)
Load resistance, $R_L$ (Ohms)

The system can be tested in all three modes to determine which mode (0, 1 or 2) provides the highest power output. The performance data outputs from such a test could be:

The mode number $M_{max}$ (0, 1 or 2) providing maximum power
System power output for the maximum mode $\dot{W}_{\max}$ (W)

For this system, the goal is to develop and evaluate a machine-learning based model that can predict which mode will produce the most power for a specified set of operating conditions. Specifically, the objective is to use the model that predicts the mode number $M_{max}$ for maximum performance and output power $\dot{W}_{\max}$ of the solar PV panel for that mode at a given set of operating conditions ($T_{air}$, $I_D$, $R_L$) for model-based control of the PV system. The details of how to set up this type of model, and a second model to assess its performance are described in the two tasks below.

**Task 2.1**
**(a)** Start with the skeleton code CodeP3.2.1F22. This code is similar to CodeP3.1.2F22, but here, it directs you to use data set DS3.2.1multiModePerfF22, which contains the arrays for the input data [$M$, $T_{air}$, $I_D$, $R_L$] (note this includes the mode number $M$), and the output parameters are the load voltage and power output [$V_L$, $\dot{W}$] for the 4 panel system depicted in Fig. 2. Except for the mode numbers, determine the median value for each parameter and normalize the data by dividing each parameter value by its median value. Do not normalize the mode numbers.

**(b)** Take the normalized DS3.2.2multiModePerfF22 data and separate it randomly into two data sets: a training set with 3/4ths of the data and a second validation set with 1/4 of the data.

**(c)** Substitute the normalized training data into the skeleton code CodeP3.2.1F22 and convert the code to a neural network model that can be trained using the training data set. For this model, $M$ with normalized $T_{air}$, $I_D$, and $R_L$ should be the inputs, and the model should be trained to match normalized data values of [$V_L$, $\dot{W}_{\max}$]. Here, use a sequential network and make appropriate choices for the number of inputs, the number of hidden layers, and the number of neurons in each layer (including the output layer). Base your choices on your experience in constructing previous models, and make the network complex enough to accurately fit the data, and avoid making it so complex that convergence takes an extreme number of iterations and/or the model overfits the data. Be sure to clearly document all your network design choices in your final report.

**(d)** Train the neural network model constructed in part (c) using the training data. Try to get the mean absolute error below 0.020 if possible.

**(e)** Compare the trained model predictions to the training data set, report the mean absolute error for the fit, and create separate log-log plots of predicted power output vs. data value power output, for each set of data point operating conditions.

**(f)** Repeat the steps of part (e), comparing the model predictions, this time to the normalized validation data. Report the mean absolute error and include the log-log plots specified in (e) in the summary report.

**Task 2.2**
For this task, the goal is to create and train a neural network that can predict the mode number (categorization) that maximizes the power output for a given set of input conditions. This

prediction can be provided to a control system which can set switches so the system has the most efficient mode. Use skeleton code CodeP3.2.2F22 as a starting point and follow the steps below to construct and train a neural network model to predict the maximum power mode.

**(a)** Note that in the first cell, arrays are installed so they provide normalized air temperature (deg C), solar input (W/sqm), and load resistance (ohms) as inputs and the output array contains the mode number providing maximum power output, $M_{maxint}$. As a first step you must determine appropriate median values for the input variables, and replace the dummy values for Tamed, IDmed, and RLmed in the skeleton program. Do not normalize the mode numbers.

**(b)** As a next step you must replace the ydata array with an array that represents the categories 0, 1, 2 with one-hot encoding. Specifically, you replace:

[0.] with [1, 0, 0]
[1.] with [0, 1, 0]
[2.] with [0, 0, 1]

and rename the output array 'ydataCatOHE'. Note that the input and output arrays are printed, so check the printed values to make sure your changes are functioning correctly. Note also that you must convert xdata and ydataCatOHE to numpy arrays xdataarray and ydataCatOHEarray.

**(c)** Take the data in the combined xdataarray and ydataCatOHEarray arrays and separate it randomly into two data sets: a training set with 3/4ths of the data and a second validation set with 1/4 of the data. Give the arrays the names:

**train_X** ~ the numpy array containing input data for the training set
**train_label** ~ the numpy array containing one hot encoding category data for the training set
**valid_X** ~ the numpy array containing input data for the validation set
**valid_label** ~ the numpy array containing one hot encoding category data for the training set

Instead of doing this manually, you can use scikitlearn tools to do it. To follow this path, add the code indicated below to the first cell just after the comment that says

```
#ADD CODE TO PARTITION THE DATA HERE - MANUALLY OR WITH THE CODE IN THE PROJECT DESCRIPTION:
```

Here is the code to add:

```
#make sure scikit-learn (vers 1.0.2) is installed
#in your python 3.7 environment
from sklearn.model_selection import train_test_split

train_X,valid_X,train_label,valid_label = train_test_split(xarray,
ydataCatOHEarray, test_size=0.25, random_state=13)

# print to check the shape of training and validation set
print(train_X.shape,valid_X.shape,train_label.shape,valid_label.shape)

#you can add these lines of code at the end to see how the labels are #divided
between the sets:
print('training OHE labels:')
print(train_label)
print('validation OHE labels:')
print(valid_label)
```

After the code for partitioning the data, the skeleton code sets the number of classes/categories:

```
num_classes = 3
```

**(d)** Cell 2 defines the neural network model. Use the initializer as specified in the skeleton code:

```
#initialize weights with values between -0.2 and 0.5
initializer = keras.initializers.RandomUniform(minval= -0.2, maxval=0.5)
```

The code specifying the neural network in cell 2 is:

```
model = keras.Sequential(
    keras.layers.Dense(16, activation=K.elu, input_shape=[3], kernel_initializer=initializer)
    keras.layers.Dense(32, activation=K.elu, kernel_initializer=initializer)
    keras.layers.Dense(16, activation=K.elu, kernel_initializer=initializer),
  ])
```

The layer list is incomplete because it needs a proper specification for an output layer for categorization. Here the number of classes/categorizations is 3 (num_classes = 3). In the sequential model layer list, add the following output layer specification at the bottom of the list:

```
keras.layers.Dense(num_classes, activation='softmax')
```

Note this specifies the softmax activation function.

**(e)** Cell 3 compiles the model. Complete the compile statement so it reads:

```
model.compile(loss=keras.losses.categorical_crossentropy,
       optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
```

Note that this specifies crossentropy as the loss function, which make the output of each of the output layer neurons a probability. It also specifies the Adam optimizer, which is an adaptive learning rate optimizer. Cell 3 also prints a summary of the model features.

**(f)** Cell 4 trains the model. It includes the callback to save the result for the lowest loss epoch used in the previous project. The objective is to get a trained model for which the training data set and validation set both have very low loss values.

Complete the model.fit statement in this cell so it reads:

```
model_train = model.fit(train_X, train_label, epochs=2000,verbose=1,
             validation_data=(valid_X, valid_label)
```

**(g)** Cell 5 contains code that can specify a test set of the input parameters and determine the corresponding maximum power output mode $M_{maxint}$ using the trained model. It also includes code that evaluates the trained model against the validations set and reports the validation loss and the validation accuracy. Note that to generate a prediction of the highest power output mode $M_{maxint}$ using the trained model, you load the input variables $T_{air}, I_D, R_L$ into the test array. For just one input test point, the code delivers the output to an array outpt[0]. When printed, outpt[0] looks something like:

```
[9.9955148e-01 4.4846025e-04 8.5744372e-09]
```

The three numbers are the outputs resulting from the one hot encoding and the crossentropy loss function. Note that this indicates the probability of $M_{max}$ being 0 is 0.9955, being 1 is 4.48e-04 and being 2 is 8.57e-09. To get the most probable category as an integer output we use:

```
Mmaxint = np.argmax(np.round(outpt[0]))
```

The round function rounds each number in the array to the nearest integer, and the argmax function returns the index of the largest value (either 0, 1, or 2).

So using

```
test = [] #specifies a test input data set
outpt=[]  #output of model for test input
test = [[ T_air , I_D , R_L ]]
testarray = np.array(test)
outpt = model.predict(testarray)
Mmaxint = np.argmax(np.round(outpt[0]))
```

determines the mode `Mmaxint` predicted by the trained network that provides maximum power output for specified values of $T_{air}, I_D, R_L$.

Adapt the code in cell 5 as necessary to evaluate the trained model and answer the questions in the following subsections.

**(h)** After completing the steps above to set up the model, run the fit routine multiple times to see if you can get the loss below 0.020. Use code in Cell 5 to compare the trained model predictions to the training data set, and report the loss and accuracy for the trained model for these data. Also compare the model predictions to the normalized validation data. Determine the loss and accuracy of the trained model for the validation data. Summarize the loss and accuracy values for the training and validation data in a table in your report.

**(i)** Based on your results, is there evidence of overfitting in this model? Quantitatively justify your conclusions

**(j)** As discussed in class, the tendency to overfit can be reduced by adding dropout layers to the network. To explore this add a dropout layer after each of the hidden layers (excluding the input and output layers). Set the dropout probability to 0.25. Your model definition should then look like:

```
model = keras.Sequential([
    keras.layers.Dense(16, activation=K.elu, input_shape=[3],kernel_initializer=initializer),
    keras.layers.Dense(32, activation=K.elu,  kernel_initializer=initializer),
    keras.layers.Dropout(0.25),
    keras.layers.Dense(16, activation=K.elu, kernel_initializer=initializer),
    keras.layers.Dropout(0.25),
    keras.layers.Dense(num_classes, activation='softmax')
  ])
```

With this change, run the fit routine multiple times to see if you can get the loss below 0.025. If you don't get much of an effect, you can try increasing the dropout probability a bit. Compare the trained model predictions to the training data set and report the loss and accuracy for the trained model for these data. Also compare the model predictions to the normalized validation data. Determine the loss and accuracy of the trained model for the validation data. Summarize the loss and accuracy values for the training and validation data in a table in your report. Based on your results, is there evidence that the addition of dropout reduces overfitting in this model, or does it not make much difference? Quantitatively justify your conclusions.

**(k)** The final element of this task is to compare predictions of the two models you have created in Part 2 of this project to examine how well the Task2.2 model predicts the mode number $M_{max}$ for maximum power output. To do this:

**[k.1]** Use the Part 2, Task 2 model to predict the mode number for maximum power output

| $T_{air}$ (deg, C) | $I_D$ (W/m$^2$) | $R_L$ (Ohms) | $M_{max,int}$ predicted by Task 2.2 model | $\dot{W}$ (W) predicted by Task 2.1 model for $M = 0$ | $\dot{W}$ (W) predicted by Task 2.1 model for $M = 1$ | $\dot{W}$ (W) predicted by Task 2.1 model for $M = 2$ |
|---|---|---|---|---|---|---|
| 10.0 | 200 | 50. | | | | |
| 20.0 | 200 | 130. | | | | |
| 10.0 | 500 | 40. | | | | |
| 20.0 | 500 | 80. | | | | |
| 20.0 | 700 | 30. | | | | |
| 20. | 700 | 55. | | | | |
| 10.0 | 1000 | 12. | | | | |
| 20.0 | 1000 | 25. | | | | |
| 20.0 | 1000 | 39. | | | | |

($M_{max,int}$) for the combinations of operating conditions $T_{air}, I_D, R_L$ in the table below:

In the table, document the values of $M_{max,int}$ predicted by the Task2.2 model neural network model for these conditions.

**[k.2]** Then, for $M$ = 0, 1, and 2, input each combination of the variables $T_{air}, I_D$, and $R_L$ in the table into the neural network model developed in Task2.1, and determine its predicted power output $\dot{W}$. With the results generated this way, fill in the empty columns and determine the accuracy (ratio of correct values to total number of values) for the collection of $M_{max,int}$ values in the table.

Based on your results from the two tasks in Part 2, in your report, summarize your assessment of whether the best Task2.2 neural network model (with or without dropout) can accurately control the switch setting for optimal performance in the multi-mode 4 PV panel system described in Fig. 2.

## Overall Project 3 Tasks to be divided between coworkers:

(1) Data prep and program modifications for Part 1
(2) Training process and computations for comparisons
(3) Plotting and interpretations of results for Part 1
(4) Data prep for Part 2
(3) Program modifications for neural network modeling in Part 2
(5) Plotting and analysis of the results for Part 2
(6) Write-up of the results and conclusions

## Deliverables:

Written final report should include:

(1) Written summary of how the work was divided between coworkers.

(2) Assessment of the results and comparisons for the two different neural network designs considered in Part 1.

(3) Plots and tables requested in Parts 1 and 2. (should not be in the Appendx, be sure to label axes with units)

(4) An assessment of viability of the first neural network design considered for system control in Part 2.

(5) Your assessments and conclusions should be clearly written with quantitative information to justify them.

(6) A copy of your programs should be attached to the report as an appendix.

## Grade will be based on:

(1) thoroughness of documentation of your analysis , especially the logic behind design choices for neural network
(2) accuracy and clarity of interpretation
(3) thoroughness and the documentation of the reasons for your assessments of results.

Summary report due: Thursday **November 17, 2022**