ALBANY STATE UNIVERSITY, GA

WEB DESIGN AND DEVELOPMENT USING AI

CHAPTER FOUR – TRAIN MODEL ONE

BY

Wanjun Hu

Department Of Math, Cs & Physics

Albany State University

504 College Dr, Albany Ga

Wanjun.Hu@asurams.edu

https://www.backupspirit.com/camp

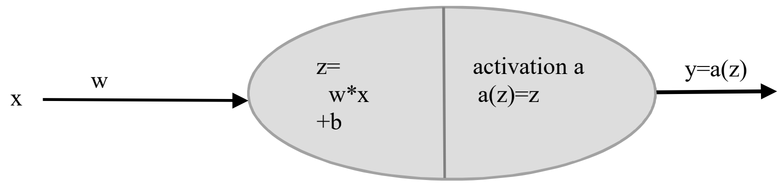
Table of Contents

CHAPTER FOUR — TRAIN MODELS

We will train two models using our web browsers. They are small machine learning models and the dataset is also small. So we do not need GPT or TPU. Further, we will use JavaScript as underlying data processing language, other than Python.

Section 4.1 The linear regression model

In Chapter 3, section 3.6, we discuss a sequential model for linear regression model y=wx+b.



The dataset

The dataset for the training comes from Google, https://storage.googleapis.com/tfjs-tutorials/carsData.json. You visit the link to take a look at it. It is in the JSON format. It is an array of JavaScript object. Each object looks like the following.

```
{
    "Name": "chevrolet chevelle malibu",
    "Miles_per_Gallon": 18,
    "Cylinders": 8,
    "Displacement": 307,
    "Horsepower": 130,
    "Weight_in_lbs": 3504,
    "Acceleration": 12,
    "Year": "1970-01-01",
    "Origin": "USA"
},
```

It consists of **key: value** pairs, e.g., “Name”: "chevrolet chevelle malibu", or "Miles_per_Gallon": 18, etc. There are 9 key : value pairs.

We will only use these two:

Miles_per_gallon: value,

Horsepower: value.

So the dataset becomes a collection of the following javascript objects.

```
{
    "Miles_per_Gallon": 18,
    "Horsepower": 130,
},
```

Section 4.2 Training process of model one

Step 1: load the javascript libraries

We will need two javascript libraries from TensorFlow, one is the "tf.min.js" for defining the machine learning model, and the other is "tfjs-vis.umd.min.js" for browser visualization of training process. The links are

https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js, and

https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis@1.0.2/dist/tfjs-vis.umd.min.js.

Step 2: Fetch the dataset and extract the targeting key:value pairs

We read the dataset from google, then extract above mentioned two key:value pairs from the dataset. The dataset is from Google. https://storage.googleapis.com/tfjs-tutorials/carsData.json'. We can display the first 10 data values from the dataset.

Step 3: Extract Miles_per_Gallon and Horsepower from the dataset

For our training, we only need Miles_per_Gallon and Horsepower. So, we need to extract from the original dataset.

Step 4: Visualize the extracted dataset

We use TFVS_VIS library to display a plot of Miles_per_gallon vs Horsepower.

Step 5: Define the machine learning model

We will use a sequential model. There is only one input variable x. The output layer has only one unit, y.

Step 6: Show a summary of the model

After create the model, use the tfjs.vis library to show a summary of the model to the web browser.

Step 7: prepare the dataset

We will shuffle the dataset for randomization. In addition, we will normalize the values to the range of [0,1]. This step is useful when your values are in different scales.

We show another plot of the prepared dataset.

Step 8: train the model

Now, we can train the model using the prepared dataset. The cost function to use is the MSE (mean square error). In addition, we will use the "adam" optimizer.

Training set size: The total number of (Miles_per_Gallon, Horsepower) pairs, here 406.

Sample: Each number pair (Miles_per_Gallon, Horsepower) is considered as one sample.

Batch: A batch contains several samples. The cost function is calculated as the sum of individual loss of each sample in a batch.

Batch size: The number of samples in a batch is called *batch size*. The batch size can be any number between 1 and total number of images.

- **a.** When batch size=1, we will calculate cost function and apply gradient descent 406 times, basically on pair. That is called ***Stochastic Gradient Descent***.
- **b.** When batch size=406 (the total number of training samples), we will calculate cost function only once and apply gradient descent once. That is called ***Batch Gradient Descent***. The example discussed in Chapter 3 of house sales is such a batch gradient descent.
- **c.** When batch size is a number between 2 and 406-1, say 32, we will calculate cost function and apply gradient descent 406/32=12 times. That is called ***Mini-Batch Gradient Descent***.

Epoch: By one epoch, we mean the training go over the whole training dataset once. Sometimes, we need to go over the whole training dataset more than once in order to reach a better result.

For this model, we set

Batch size =32

Epochs = 50

When running the model, also visualize the training performance to web browser.

<u>Step 8: Generate a sample uniform inputs between 0 and 1</u>

To have a 100 uniformly distributed inputs, we can show the line for y=wx+b.

<u>Step 9: make predictions</u>

Use TFJS_VIS to show a plot that display the extracted data, and the generated data with their predictions.

Section 4.3 AI prompts for model one

Use the following sequence of prompts.

1. Create a webpage that contains a banner showing “Summer CAM WDD Using AI”.

2. Make the background color to silver.

3. Set the webpage to use bootstrap framework.

4. inside <body>, add a <main> block.

5. inside the main block, add 20 <div>'s with id to be "div1", "div2", ...,"div20".

6. inside the <div id="div1">, add headline 2, with content "Section 1. What is a linear regression?". Under the headline 2, add an unordered list. The first list item shows "It tries to find a linear function y=wx+b". The second list item shows "The linear function tries to fit a collection of number pairs, such as house sales, (sqft, sale price). The third list item shows "There is no linear function that can accurately fit all number pairs. So, we we have to find the 'best' one." The fourth item shows "By being best, we design a so-called cost function, such as mean sqaure error invented by the great Carl Gauss. The fifth list item says "This is what machine learning is doing." Inside the fifth list item, shows the picture "linearRegression1.jpg" on a new line.

7. inside <div id="div2">, add headline 2 “Section 2. The javaScript libraries to use”. Then add a paragraph "We will use TensorFlow.js" to train the model. Then, add another paragraph, "Click the 'Run' button below to import two javaScript libraries".

8. Inside <div id="div3">, add a paragraph shows "import https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js for defining and training models". Add another paragraph shows "import https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis@1.0.2/dist/tfjs-vis.umd.min.js for web browser visualization".

9. Inside <div id="div3", add a button with content "Run". When clicking on the button, it will load the two libraries "https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js", and "https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis@1.0.2/dist/tfjs-vis.umd.min.js". Display the console output in <div id="div4">

10. add borders to all <div>'s inside the main block.

11. resize the image "linearRegression1.jpg" to have width =400 and height = 300.

12. Resize the image "linearRegression1.jpg" inside <div id="div1"> to 400 by 300.

13. Inside <div id="div5">, add a headline 2 “Section 3. Using the dataset from Google”. Then add a paragraph that says "We will use a dataset from Google, https://storage.googleapis.com/tfjs-tutorials/carsData.json". Add a button "Load the dataset". When clicking the "Load the dataset" button, fetch the dataset from https://storage.googleapis.com/tfjs-tutorials/carsData.json. Display console output to <div id="div6">.

14. in the loadDataset function, save the dataset to a global variable so that I can use it later.

15. In the function loadDataset(), when display console output in <div id="div6">, only display the first 10 data values.

16. Add another javascript function that extract the "Miles_per_gallon: value" and "Horsepower: value" from the globalDataset. Save the extracted dataset to another global variable.

17. Inside <div id="div7">, add a headline 2 “Section 4. Extract the Miles_per_Gallon and Horsepower”. Then add a paragraph showing "We will use only Miles_per_Gallon:value and Horsepower: value for our training." Then, add a button "Extract training dataset". When clicking "Extract training dataset", call the extractData() method, save the extracted dataset to extractData, and display the first 10 data values of the extractData.

18. inside the extractData() function, change "Miles_per_gallon" to "Miles_per_Gallon".

19. inside <div id="div8">, add a headline 2 “Section 5. Visualize the training dataset”. Then add a button "Visualize the training dataset". When clicking "Visualiza the training dataset" button, display a plot based on extractedData. The title is "Horsepower vs Miles_per_Gallon". The horizontal axis is "Horsepower" starting from 0 to 240 with an increment of 20. The vertical axis is "MPG" starting from 0 to 50 with an increment of 10.

20. Inside <div id="div9">, change the background color to black. Then add a headline 2 “Section 6. Define the model”. Then, add a paragraph showing "We will define a TensorFlow sequential model with one input x and one output y". Then add javascript coding to define a TensorFlow sequential model. Add one TensorFlow layer as input with shape 1, units=1. Then add a layer as output with 1 unit.

21. Inside <div id="div10">, add a button "Show summary of the model". When clicking the bottom, display summary of the model below the button.

22. make the variable model global.

23. change the font color in <div id="div9"> to red.

24. Add a button “Define the model” inside <div id="div9">. When clicking, it defines a TensorFlow sequential model. The input layer has shape [1] and 1 unit. The output layer has one unit.

25. inside the showModelSummary function, use the TensorFlow tfjs_vis library to show model summary

26. Inside the showModelSummary function, use tfvis to show model summary inside <div id="div10">

27. Inside <div id="div11">, add a headline 2 “Section 7. Prepare the training set”. Then, add a paragraph saying "We will prepare the extracted data." Then add an unordered list. The first list item says "Convert the extracted data to Tensor". The second item says "Shuffle the data". The third item says "Normalize to the range between 0 and 1 for fast computation".

28. Inside <div id="div11">, add a button "Prepare the dataset". When clicking, it will convert extractedData to a TensorFlow tensor. Let that tensor to be global. Then do shuffling of that tensor and then normalizing the values to the range of 0 to 1.

29. Inside the function prepareDataset, first extract horsepower as inputs, and mpg as labels. Then, convert both inputs and labels to TensorFlow tensors. Finally, normalize both tensors. Return the object {inputs: normalized tensor, labels: normalized tensor, input Max, input Min, label Max, label Min}.

30. Inside prepareDataset function, the inputTesnor and labelTensor are 2 dimensional.

31. Inside the function preparedDataset, save the return value to the global variable preparedData.

32. Inside <div id="div12", add a headline “Section 8. Visualize the inputs and labels for training”. add a button "Visualize the inputs and labels". When clicking, use the tfjs_vis library to visualize a plot of the inputs and labels from the preparedData. The title of the plot "Inputs vs Labels". The x-axis is "Inputs" starting from 0 to 1 with an increment of 0.1. The y-axis is "Labels" starting from 0 to 1 with an increment of 0.1. Make the plot inside <div id="div12">.

33. Inside the prepareDataste function, the tf.tensor2d should have a shape of [*, 1].

34. inside <div id="div13">, add a headline 2 “Section 9. Train the model”. Then add a button "Train the model using prepared data". When clicking, it starts train the model. The loss is "mean squared error". The optimizer is "adam". The metrics is "mse". The batch size is 32 and the epochs is 50. For callback, Use tfvs_vis to show the training performance. The title is "training performance. The los s is "mse". The height is 200.

35. Inside the trainModel function, display the tfvis callbacks inside <div id="div13">.

36. Inside <body>, add a <footer> tag. In the <footer> tag, add a paragraph "Copyright@2024, Albany State University Summer Camp 'Web Design and Development Using AI'". The padding top is 30px and the padding bottom is 20 px.

37. In the <footer>, position the content to the center.

38. In <div class="banner">, position the content to the center.

39. Inside <div id="div14">, add a headline 2 “Section 10. Generate the dataset for testing”. Then add a button "Generate a test data values". When clicking, it will generate a uniform range of numbers between 0 and 1 for inputs. Then, it will use the trained model to predict the results. After, calculating the predictions, it will un-normalize the

data by doing the inverse of the min-max scaling that we did earlier. Save the un-normalized inputs and un-normalized predictions to a global variable.

40. inside generateTestData function, there is an error, which says "numFeatures" not defined. Please correct it.

41. Inside generateTestData, when calling model.predict, there is an error, "Uncaught Error: Error when checking : expected dense_Dense1_input to have shape [null,1] but got array with shape [10,5].". Please correct it.

42. Inside generateTestData function, set numFeatures =1

43. Inside the generateTestData, inputMax, inputMin should come from preparedData. outputMax, outputMin should be labelMax, labelMin in preparedData.

44. Inside <div id="div15">, add a headline 2 “Section 11. Visualize the predictions”. Then, add a button "Visualize the predictions". When clicking, use the tsjs_vis to display a plot that include the extractedData and testDataInputs with testDataPredictions. Display the plot inside <div id="div15">.

45. Neither extractedData nor testDataInputs display as expected. Please correct it.

46. AI got stucked here. Please this version of “VisualizePredictions()” function.

```
function visualizePredictions() {

     /* not from AI --- start*/
   const predictedPoints = Array.from(testDataInputs).map((val, i) => {
          return {x: val, y: testDataPredictions[i]}
   });

   const originalPoints = extractedData.map(d => ({
          x: d.Horsepower, y: d.Miles_per_Gallon,
   }));
   /* not from AI --- start*/

   const data = {
        values: [originalPoints, predictedPoints],
        series: ['Extracted Data', 'Test Data Predictions'],
   };
   const surface = tfvis.render.scatterplot(
                div15, data, {xLabel: 'X', yLabel: 'Y', height: 300,
                                series: ['Extracted Data', 'Test Data Predictions']
                  }
            );
        }
```

Section 4.4 Link to the final product

The final webpage can be located https://www.backupspirit.com/camp/model1.html.