[01. Neural Network Regression with TensorFlow]

Regression Problems = Predicting a number

// -- -- -- -- -- -- -- -- -- -- -- -- // -- -- -- -- -- -- -- -- -- -- -- -- // -- -- -- -- -- -- -- -- -- -- -- -- //

Steps to solve a regression problem using TensorFlow:

* Input shapes and output shapes
* Architecture of a regression model
  + X (inputs): features/data
  + y (outputs): labels
* Creating custom data to view and fit
* Steps in modeling
  + Creating a model
  + Compiling a model
    - Defining a loss function
    - Setting up an optimizer
    - Creating evaluation metrics
  + Fitting a model (getting it to find patterns in our data)
* Evaluating a model
  + Visualizing the model ("visualize, visualize, visualize")
  + Looking at training curves
  + Compare predictions to ground truth (using our evaluation metrics)
* Saving a model (so we can use it later)
* Loading a model

| ****Hyperparameter**** | ****Typical value**** |
| --- | --- |
| Input layer shape | Same shape as number of features (e.g. 3 for # bedrooms, # bathrooms, # car spaces in housing price prediction) |
| Hidden layer(s) | Problem specific, minimum = 1, maximum = unlimited |
| Neurons per hidden layer | Problem specific, generally 10 to 100 |
| Output layer shape | Same shape as desired prediction shape (e.g. 1 for house price) |
| Hidden activation | Usually [ReLU](https://www.google.com/url?q=https%3A%2F%2Fwww.kaggle.com%2Fdansbecker%2Frectified-linear-units-relu-in-deep-learning) (rectified linear unit) |
| Output activation | None, ReLU, logistic/tanh |
| Loss function | [MSE](https://www.google.com/url?q=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FMean_squared_error) (mean square error) or [MAE](https://www.google.com/url?q=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FMean_absolute_error) (mean absolute error)/Huber (combination of MAE/MSE) if outliers |
| Optimizer | [SGD](https://www.google.com/url?q=https%3A%2F%2Fwww.tensorflow.org%2Fapi_docs%2Fpython%2Ftf%2Fkeras%2Foptimizers%2FSGD) (stochastic gradient descent), [Adam](https://www.google.com/url?q=https%3A%2F%2Fwww.tensorflow.org%2Fapi_docs%2Fpython%2Ftf%2Fkeras%2Foptimizers%2FAdam) |

**Note:** A **hyperparameter** in machine learning is something a data analyst or developer can set themselves, where as a **parameter** usually describes something a model learns on its own (a value not explicitly set by an analyst).

// -- -- -- -- -- -- -- -- -- -- -- -- // -- -- -- -- -- -- -- -- -- -- -- -- // -- -- -- -- -- -- -- -- -- -- -- -- //

Neural networks accept numbers and output numbers. These numbers are typically represented as tensors (or arrays).

|  |  |
| --- | --- |
| Using np to create arrays | Using tf to create tensors |
| import numpy as np  import matplotlib.pyplot as plt  X = np.array([-7.0, -4.0, -1.0, 2.0, 5.0, 8.0, 11.0, 14.0])  y = np.array([3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0, 24.0])  plt.scatter(X, y) | import numpy as np  import matplotlib.pyplot as plt  # Create features (using tensors)  X = tf.constant([-7.0, -4.0, -1.0, 2.0, 5.0, 8.0, 11.0, 14.0])  # Create labels (using tensors)  y = tf.constant([3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0, 24.0])  # Visualize it by creating scatter plots  plt.scatter(X, y); |

In TensorFlow, you can create tensors in several ways:

|  |  |
| --- | --- |
| 1. Using tf.constant():  This method creates a tensor with constant values that cannot be changed.  import tensorflow as tf  tensor = tf.constant([1, 2, 3]) | 2. Using tf.Variable():  This method creates a tensor with initial values that can be modified during computation.  import tensorflow as tf  tensor = tf.Variable([1, 2, 3]) |
| 3. From NumPy arrays:  You can convert a NumPy array to a TensorFlow tensor using tf.convert\_to\_tensor().  import tensorflow as tf  import numpy as np  array = np.array([[1, 2], [3, 4]])  tensor = tf.convert\_to\_tensor(array) |  |
| 4. Using tf.zeros(), tf.ones(), tf.fill():  These methods create tensors filled with zeros, ones, or a specified value.  import tensorflow as tf  zeros\_tensor = tf.zeros([2, 3])  ones\_tensor = tf.ones([2, 3])  filled\_tensor = tf.fill([2, 3], 5) | 5. Using random number generators:  You can create tensors with random values using functions like tf.random.normal() or tf.random.uniform().  import tensorflow as tf  random\_tensor = tf.random.normal([2, 3]) |

Note: Two of the things you'll spend the most time on when you work with neural networks: making sure your input and outputs are in the correct shape.

// -- -- -- -- -- -- -- -- -- -- -- -- // -- -- -- -- -- -- -- -- -- -- -- -- // -- -- -- -- -- -- -- -- -- -- -- -- //

Improving a model

|  |
| --- |
|  |

increasing the number of layers (making the network deeper)

increasing the number of hidden units (making the network wider)

Because the above values are all human-changeable, they're referred to as [*hyperparameters*](https://www.google.com/url?q=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FHyperparameter_(machine_learning)) and the practice of trying to find the best hyperparameters is referred to as [*hyperparameter tuning*](https://www.google.com/url?q=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FHyperparameter_optimization).

// -- -- -- -- -- -- -- -- -- -- -- -- // -- -- -- -- -- -- -- -- -- -- -- -- // -- -- -- -- -- -- -- -- -- -- -- -- //