Aqsa Noreen

Professor Christopher Brooks

Intro to AI

December 1, 2023

Task 2: TensorFlow

What is a MaxPooling2D layer? What's it do?

MaxPooling2D is like a zoom-out feature that we are familiar with that takes a bigger image and makes it smaller while still maintaining the most important parts, two spatial dimensions, height and breadth are mainly flattened. This reduces the workload for the neural network and focuses on the key features.

What's Adam?

Swapping out the traditional stochastic gradient descent method for repeatedly updating network weights using training data is possible using Adam, which is an optimization algorithm. It functions similarly to an intelligent assistant to train neural networks. It is crucial for optimal learning since Adam helps control the network's learning rate to make sure it doesn't go too fast or too slow.

What's the softmax function do?

It is a decision-maker at the end of the network that takes the network's output and then turns them into probabilities and shows which category is most likely the correct one.

In a machine learning model's picture guessing, Softmax is the smart function that serves as the game's ultimate judge. The picture of a sunflower is a good example to use from the article on image classification. The model begins by attempting to identify the sunflower in the image by assigning numerical values to each of the words it is familiar with, such as "sunflower," "rose," etc. The ratings represent its method of expressing, "I believe it's this likely to be a sunflower, this much for a rose," and similarly distributed. However, these ratings are unprocessed and not very user-friendly for immediate comprehension. Softmax is useful in this situation. After verifying that all scores are positive, it modifies them so that they sum to 100%. This is the first of two primary steps in the scoring process. Like turning these numbers into a language of absolute certainty. It becomes crystal obvious, "I'm 70% sure it's a sunflower, 20% a rose," once softmax is applied and the scores become possibilities.

What is CategoricalCrossEntropy? What do we use it for?

It's a method for evaluating the current state of the network. In tasks with several categories to pick from, it acts as a scorekeeper by telling how distant the network's predictions are from the actual responses. In multi-class classification issues, it is often used as a loss function. A classification model's performance is evaluated using this metric, which yields a probability value between 0 and 1. By comparing the expected and actual distributions of the one-hot encoded labels, categorical cross-entropy finds the most accurate distribution.

In the CNN example, what does the Flatten layer do?

One function of the Flatten layer in the TensorFlow Core CNN example is to flatten the three-dimensional output of the convolutional base into a one-dimensional array. The subsequent Dense layers need this transition because of their requirement for 1D input. For the convolutional base's output tensor, which is (4, 4, 64) in this case, the Flatten layer simplifies it into a single vector of shape (1024).

In the CNN example, what does the Dense layer do?

Classification is the function of the model's Dense layers. The model incorporates two Dense layers after the flattening of the output from the convolutional base. There are 64 nodes in the first Dense layer, which employs the'relu' activation function. The last Dense layer, which corresponds to the 10 classes in the CIFAR dataset, has 10 outputs. The categorization results are output by this layer.

In the CNN example, why does the height and width get smaller for each convolutional layer?

The usage of MaxPooling2D layers causes the convolutional layers to become thinner and taller as the network depth increases. In order to concentrate on the input's higher-level characteristics, these layers decrease the input volume's spatial dimensions (height and width) after each convolutional layer. This helps to decrease computation and the number of parameters.

What does it mean to normalize the data? Where else have we seen normalization?

Data normalization is the process of making all numerical columns have the same scale or range of values. When talking about feature scaling and preprocessing in ML models like neural networks, we have encountered normalization before. Both the convergence and performance of the model are enhanced by normalization.

Why is it a problem that the Titanic data has different types and ranges? Why did we not have to worry about this with the decision tree?

The different data types and ranges in the Titanic dataset make it very hard to simply stack the features into a NumPy array and pass them to a model like tf.keras.Sequential. Each column needs to be handled individually. This wasn't an issue with decision trees which can directly handle categorical and numeric features without any data normalization or preprocessing.

What is a one-hot vector?

A one-hot vector is a way to represent categorical data. It uses binary encoding, where each category value is represented by a vector with one high (1) bit at each index. A numeric vector that may be used as model input is created by encoding each category value in this way:

```
one_hot = layers.CategoryEncoding(num_tokens=lookup.vocabulary_size())
x = lookup(input)
x = one_hot(x)
```

The example that shows how to manually slice the feature dictionary uses yield instead of return. Why is this? What's the difference between them, and why would you want to use yield?

Yield is used instead of return here because it allows the function to generate multiple values over the iterations instead of just one value. This avoids having to build a huge list in memory. For example:

```
def slices(features):
    for i in itertools.count():
        example = {name:values[i] for name, values in features.items()}
    yield example
```

So yield makes it like a generator, allowing us to iterate over the examples and generate them on the fly. When dealing with massive datasets, yield is useful since it generates things as required instead of storing them all in memory, which helps with memory management.

As we know, encoding is a particularly important part of working with neural networks. Explain how text is encoded for an RNN.

The TextVectorization layer is used to encode the raw text into integer token indices. Its adapt() method sets the vocabulary based on the input text data. An embedding layer then converts these integer sequences into sequences of dense vectors. The vectors are trainable words with similar meanings often end up having similar vectors after training on data that is just enough. This is more efficient than passing one-hot encoded vectors through a Dense layer.

What does the Bidirectional layer do? What are the advantages and disadvantages of this approach? How does it compare to the way we processed sequence data with an HMM?

In order to combine the outputs of the RNN layer, the bidirectional wrapper sends the input sequence in both directions. One advantage is that the signal from the beginning of the input doesn't need to propagate all the way through to affect the output and one disadvantage is that we can't efficiently stream predictions as new words are added since the full sequence needs to be processed in both directions. In an HMM we only process sequence data in one direction, so streaming predictions as new observations come in is more feasible. However, the bidirectional approach allows the full context on both sides of an input timestep to inform the output.

What is masking? Why do we need to use it in this example?

Due to the batching of text sequences of varying lengths, masking is used. To ensure that they are uniform in length, padding tokens are inserted. Layers are instructed to disregard these padding tokens by the mask.

```
model = tf.keras.Sequential([
    encoder,
    tf.keras.layers.Embedding(
        input_dim=len(encoder.get_vocabulary()),
        output_dim=64,
        # Use masking to handle the variable sequence lengths
        mask_zero=True),
    ...
])
```

The Embedding layer is configured with mask_zero=True to ignore the padding tokens.