nn3

March 16, 2023

1 Neural Networks

1.1 Introduction

popular

1.1.1 Setup

neurons w/ weights w (+ biases b) and nonlinearity/activation ϕ

$\phi(\sum_i x_i w_i + b_i)$

In layers w/ weights $W \in \mathbb{R}^{n_l \times n_{l+1}}$ and biases $b_l \in \mathbb{R}^{n_k}$ w/ n_l neurons in layer l:

$$\phi(W_l x_l + b_l)$$

(abuse of notation w/ ϕ)

(if input points are $x \in \mathbb{R}^d$, then $n_l = d$)

Do this for all layers to get some output values in your final layer (*forward pass*)

set initial weights W_l randomly

Tons of different shapes/types of NNs

split data into train and test (80/20ish is good)

1.1.2 Backpropagation

Loss $L(y)$ is a function of the output y and the target t, e.g.:

$$L(y) = (t - j)^2$$

Calculate derivative wrt each weight $D_n = \frac{\partial L(y)}{\partial w_n}$ and use gradient descent to update weights:

$$w_n \leftarrow w_n - \eta D_n$$

for learning rate η

[ ]:
```
from sklearn.datasets import load_iris
X, y = load_iris(as_frame = True, return_X_y=True)
```

[ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[ ]:
```
import tensorflow as tf
train = tf.data.Dataset.from_tensor_slices((X_train, y_train))
test = tf.data.Dataset.from_tensor_slices((X_test, y_test))
```

```
2023-03-16 15:23:23.935251: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 AVX_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2023-03-16 15:23:24.029077: I tensorflow/core/util/port.cc:104] oneDNN custom
operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn them
off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2023-03-16 15:23:24.031748: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot
open shared object file: No such file or directory
2023-03-16 15:23:24.031760: I
tensorflow/compiler/xla/stream_executor/cuda/cudart_stub.cc:29] Ignore above
cudart dlerror if you do not have a GPU set up on your machine.
2023-03-16 15:23:24.432335: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot
open shared object file: No such file or directory
2023-03-16 15:23:24.432374: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libnvinfer_plugin.so.7'; dlerror:
libnvinfer_plugin.so.7: cannot open shared object file: No such file or
directory
2023-03-16 15:23:24.432378: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with
TensorRT, please make sure the missing libraries mentioned above are installed
properly.
2023-03-16 15:23:25.776869: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-03-16 15:23:25.777135: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot
open shared object file: No such file or directory
2023-03-16 15:23:25.777174: W
```

```
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcublas.so.11'; dlerror: libcublas.so.11: cannot
open shared object file: No such file or directory
2023-03-16 15:23:25.777196: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcublasLt.so.11'; dlerror: libcublasLt.so.11: cannot
open shared object file: No such file or directory
2023-03-16 15:23:25.777217: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcufft.so.10'; dlerror: libcufft.so.10: cannot open
shared object file: No such file or directory
2023-03-16 15:23:25.777237: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcurand.so.10'; dlerror: libcurand.so.10: cannot
open shared object file: No such file or directory
2023-03-16 15:23:25.777257: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcusolver.so.11'; dlerror: libcusolver.so.11: cannot
open shared object file: No such file or directory
2023-03-16 15:23:25.777278: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcusparse.so.11'; dlerror: libcusparse.so.11: cannot
open shared object file: No such file or directory
2023-03-16 15:23:25.777297: W
tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could
not load dynamic library 'libcudnn.so.8'; dlerror: libcudnn.so.8: cannot open
shared object file: No such file or directory
2023-03-16 15:23:25.777301: W
tensorflow/core/common_runtime/gpu/gpu_device.cc:1934] Cannot dlopen some GPU
libraries. Please make sure the missing libraries mentioned above are installed
properly if you would like to use GPU. Follow the guide at
https://www.tensorflow.org/install/gpu for how to download and setup the
required libraries for your platform.
Skipping registering GPU devices…
2023-03-16 15:23:25.777582: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 AVX_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

[ ]:
```
train = train.repeat(20).shuffle(1000).batch(32)
test = test.batch(1)
```

[ ]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(10, activation=tf.nn.relu),   # hidden layer
    tf.keras.layers.Dense(10, activation=tf.nn.relu),   # hidden layer
```

```
    tf.keras.layers.Dense(3, activation=tf.nn.softmax)  # output layer
])

model.compile(
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
)

model.fit(
    train,
    validation_data=test,
    epochs=10,
)
```

```
Epoch 1/10
75/75 [==============================] - 0s 2ms/step - loss: 0.9898 - accuracy:
0.4992 - val_loss: 0.8580 - val_accuracy: 0.7000
Epoch 2/10
75/75 [==============================] - 0s 1ms/step - loss: 0.7437 - accuracy:
0.8879 - val_loss: 0.6605 - val_accuracy: 0.8667
Epoch 3/10
75/75 [==============================] - 0s 1ms/step - loss: 0.5791 - accuracy:
0.9571 - val_loss: 0.5170 - val_accuracy: 0.9667
Epoch 4/10
75/75 [==============================] - 0s 831us/step - loss: 0.4592 -
accuracy: 0.9692 - val_loss: 0.4134 - val_accuracy: 0.9667
Epoch 5/10
75/75 [==============================] - 0s 2ms/step - loss: 0.3777 - accuracy:
0.9708 - val_loss: 0.3459 - val_accuracy: 0.9667
Epoch 6/10
75/75 [==============================] - 0s 1ms/step - loss: 0.3220 - accuracy:
0.9750 - val_loss: 0.2993 - val_accuracy: 0.9667
Epoch 7/10
75/75 [==============================] - 0s 3ms/step - loss: 0.2798 - accuracy:
0.9762 - val_loss: 0.2630 - val_accuracy: 0.9667
Epoch 8/10
75/75 [==============================] - 0s 2ms/step - loss: 0.2474 - accuracy:
0.9733 - val_loss: 0.2340 - val_accuracy: 0.9667
Epoch 9/10
75/75 [==============================] - 0s 2ms/step - loss: 0.2190 - accuracy:
0.9762 - val_loss: 0.2100 - val_accuracy: 0.9667
Epoch 10/10
75/75 [==============================] - 0s 820us/step - loss: 0.1931 -
accuracy: 0.9746 - val_loss: 0.1880 - val_accuracy: 0.9667
```

[ ]: <keras.callbacks.History at 0x7f00a0705090>

[ ]:
```
predict_X = [
  [5.1, 3.3, 1.7, 0.5],
  [5.9, 3.0, 4.2, 1.5],
  [6.9, 3.1, 5.4, 2.1],
]

predictions = model.predict(predict_X)

print(predictions[0])
```

```
1/1 [==============================] - 0s 27ms/step
[9.9778795e-01 2.0219113e-03 1.9010153e-04]
1/1 [==============================] - 0s 27ms/step
[9.9778795e-01 2.0219113e-03 1.9010153e-04]
```

[ ]:
```
print(predictions[0].argmax())
```

[ ]:
```
0
```

[ ]:
```
for pred_dict, expected in zip(predictions, ["setosa", "versicolor","␣
↪"virginica"]):
    predicted_index = pred_dict.argmax()
    predicted = load_iris().target_names[predicted_index]
    probability = pred_dict.max()
    tick_cross = " " if predicted == expected else " "
    print(f"{tick_cross} Prediction is '{predicted}' ({100 * probability:.
↪1f}%), expected '{expected}'")
```

```
 Prediction is 'setosa' (99.8%), expected 'setosa'
 Prediction is 'versicolor' (80.1%), expected 'versicolor'
 Prediction is 'virginica' (72.6%), expected 'virginica'
```