nn4

May 25, 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 $\eta > 0$

[ ]:
```
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))
```

```
2023-05-25 10:32:46.830399: I tensorflow/core/util/port.cc:110] 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`.
2023-05-25 10:32:46.831816: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not
find cuda drivers on your machine, GPU will not be used.
2023-05-25 10:32:46.856799: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not
find cuda drivers on your machine, GPU will not be used.
2023-05-25 10:32:46.857396: I tensorflow/core/platform/cpu_feature_guard.cc:182]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations,
rebuild TensorFlow with the appropriate compiler flags.
2023-05-25 10:32:47.343245: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not
find TensorRT
2023-05-25 10:32:48.761339: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:996]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-05-25 10:32:48.761768: W
tensorflow/core/common_runtime/gpu/gpu_device.cc:1956] 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…
```

[ ]:
```
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.Dropout(0.2),
    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
```

```
2023-05-25 10:32:55.174651: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_1' with dtype int64 and shape [120]
         [[{{node Placeholder/_1}}]]
2023-05-25 10:32:55.174817: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_1' with dtype int64 and shape [120]
         [[{{node Placeholder/_1}}]]
```

```
75/75 [==============================] - 0s 2ms/step - loss: 1.0545 - accuracy:
0.5008 - val_loss: 0.7285 - val_accuracy: 0.7000
Epoch 2/10
75/75 [==============================] - 0s 864us/step - loss: 0.8121 -
accuracy: 0.5783 - val_loss: 0.6143 - val_accuracy: 0.7333
Epoch 3/10
 1/75 […] - ETA: 0s - loss: 0.7435 - accuracy:
0.5625
```

```
2023-05-25 10:32:55.503983: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_1' with dtype int64 and shape [30]
         [[{{node Placeholder/_1}}]]
```

```
75/75 [==============================] - 0s 795us/step - loss: 0.6647 -
accuracy: 0.6425 - val_loss: 0.5250 - val_accuracy: 0.8333
Epoch 4/10
75/75 [==============================] - 0s 884us/step - loss: 0.5678 -
accuracy: 0.7437 - val_loss: 0.4611 - val_accuracy: 0.8000
Epoch 5/10
75/75 [==============================] - 0s 1ms/step - loss: 0.5177 - accuracy:
0.7975 - val_loss: 0.4182 - val_accuracy: 0.9000
Epoch 6/10
75/75 [==============================] - 0s 1ms/step - loss: 0.4831 - accuracy:
0.8271 - val_loss: 0.3814 - val_accuracy: 0.9333
```

```
Epoch 7/10
75/75 [==============================] - 0s 1ms/step - loss: 0.4406 - accuracy:
0.8496 - val_loss: 0.3469 - val_accuracy: 0.9000
Epoch 8/10
75/75 [==============================] - 0s 830us/step - loss: 0.3950 -
accuracy: 0.8679 - val_loss: 0.3244 - val_accuracy: 0.9667
Epoch 9/10
75/75 [==============================] - 0s 835us/step - loss: 0.3635 -
accuracy: 0.8958 - val_loss: 0.3008 - val_accuracy: 1.0000
Epoch 10/10
75/75 [==============================] - 0s 895us/step - loss: 0.3407 -
accuracy: 0.8933 - val_loss: 0.2770 - val_accuracy: 1.0000
```

[ ]:
```
<keras.callbacks.History at 0x7f93c3baab60>
```

[ ]:
```
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 43ms/step
[0.92734677 0.07142308 0.0012302 ]
1/1 [==============================] - 0s 43ms/step
[0.92734677 0.07142308 0.0012302 ]
```

[ ]:
```
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' (92.7%), expected 'setosa'
 Prediction is 'versicolor' (70.1%), expected 'versicolor'
 Prediction is 'virginica' (62.7%), expected 'virginica'
```

1.2 Convolutional Neural Networks (CNNs)

1.2.1 Image Kernel Convolutions

images are matrices of pixel values use kernel to convolve over image to get new image (using padding at edges maybe so output image is same size as input image—e.g. zero padding (add a border of zeros) or mirror padding (add a border of identical pixels to the edge pixels))

e.g. for kernel w and image w/ pixel coords $f(x, y)$ we get pixel value $g(x, y)$ where:

$$g(x, y) = w * f(x, y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} w(dx, dy) f(x - dx, y - dy)$$

e.g. for a 3x3 kernel:

$$w = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

(^^directional edge detection kernel (I think?))

1.2.2 CNNs

Convolutional Layers We'll make an NN learn the convolution kernels for us! (i.e. learn the weights $w_{x,y}(dx, dy)$—i.e. the weights of the kernel depend on the pixels being convolved over.) And we can stack these layers to get more complex kernels.

Pooling Layers We can also use pooling layers to reduce the size of the image (e.g. max pooling). These just take a window of pixels and output the max value (or average value or something), meaning we can reduce the size of the image without losing too much information (downsampling).

Fully Connected Layers (FC/Dense Layers) Fully connected layers are just like the ones we've seen before (i.e. in non-convolution-land), but we flatten the image first (i.e. we take the image and turn it into a vector of pixel values).

[ ]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(
        filters=16,
        kernel_size=5,
        padding="same",
        activation=tf.nn.relu
    ),
    tf.keras.layers.MaxPool2D((2, 2), (2, 2), padding="same"),
    tf.keras.layers.Conv2D(
        filters=32,
        kernel_size=5,
        padding="same",
        activation=tf.nn.relu
    ),
    tf.keras.layers.MaxPool2D((2, 2), (2, 2), padding="same"),
    tf.keras.layers.Conv2D(
        filters=64,
        kernel_size=5,
        padding="same",
        activation=tf.nn.relu
    ),
    tf.keras.layers.MaxPool2D((2, 2), (2, 2), padding="same"),
```

```
    tf.keras.layers.Conv2D(
        filters=128,
        kernel_size=5,
        padding="same",
        activation=tf.nn.relu
    ),
    tf.keras.layers.MaxPool2D((2, 2), (2, 2), padding="same"),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(10, activation="softmax")
])

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

[ ]:
```
import tensorflow_datasets as tfds

ds_train, ds_test = tfds.load(
    "mnist",
    split=["train", "test"],
    as_supervised=True,
)
```

```
/home/dg22309/Documents/Compass First
Year/Compass/SC2/.conda/lib/python3.10/site-packages/tqdm/auto.py:21:
TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

[ ]:
```
ds_train.element_spec
```

[ ]:
```
(TensorSpec(shape=(28, 28, 1), dtype=tf.uint8, name=None),
 TensorSpec(shape=(), dtype=tf.int64, name=None))
```

[ ]:
```
def normalize_img(image, label):
    return tf.cast(image, tf.float32) / 255., label

ds_train = ds_train.map(normalize_img)

ds_train = ds_train.shuffle(1000)
ds_train = ds_train.batch(128)

ds_test = ds_test.map(normalize_img)
ds_test = ds_test.batch(128)
```

[ ]:
```
model.fit(
    ds_train,
    validation_data=ds_test,
    epochs=3,
)
```

```
Epoch 1/20

2023-05-25 10:47:29.029591: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_1' with dtype string and shape [1]
         [[{{node Placeholder/_1}}]]
2023-05-25 10:47:29.029965: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_0' with dtype string and shape [1]
         [[{{node Placeholder/_0}}]]

468/469 [============================>.] - ETA: 0s - loss: 0.2923 - accuracy:
0.9053

2023-05-25 10:47:40.700244: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_0' with dtype string and shape [1]
         [[{{node Placeholder/_0}}]]
2023-05-25 10:47:40.700526: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_2' with dtype string and shape [1]
         [[{{node Placeholder/_2}}]]

469/469 [==============================] - 13s 26ms/step - loss: 0.2919 -
accuracy: 0.9054 - val_loss: 0.0500 - val_accuracy: 0.9852
Epoch 2/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0593 -
accuracy: 0.9831 - val_loss: 0.0447 - val_accuracy: 0.9865
Epoch 3/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0378 -
accuracy: 0.9894 - val_loss: 0.0271 - val_accuracy: 0.9908
Epoch 4/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0267 -
accuracy: 0.9922 - val_loss: 0.0265 - val_accuracy: 0.9926
Epoch 5/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0193 -
accuracy: 0.9943 - val_loss: 0.0255 - val_accuracy: 0.9928
Epoch 6/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0154 -
accuracy: 0.9956 - val_loss: 0.0291 - val_accuracy: 0.9919
```

```
Epoch 7/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0127 -
accuracy: 0.9963 - val_loss: 0.0275 - val_accuracy: 0.9937
Epoch 8/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0100 -
accuracy: 0.9973 - val_loss: 0.0363 - val_accuracy: 0.9923
Epoch 9/20
469/469 [==============================] - 12s 27ms/step - loss: 0.0088 -
accuracy: 0.9976 - val_loss: 0.0298 - val_accuracy: 0.9916
Epoch 10/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0073 -
accuracy: 0.9978 - val_loss: 0.0559 - val_accuracy: 0.9884
Epoch 11/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0070 -
accuracy: 0.9980 - val_loss: 0.0327 - val_accuracy: 0.9927
Epoch 12/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0059 -
accuracy: 0.9982 - val_loss: 0.0431 - val_accuracy: 0.9910
Epoch 13/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0055 -
accuracy: 0.9985 - val_loss: 0.0418 - val_accuracy: 0.9923
Epoch 14/20
469/469 [==============================] - 13s 28ms/step - loss: 0.0047 -
accuracy: 0.9985 - val_loss: 0.0430 - val_accuracy: 0.9908
Epoch 15/20
469/469 [==============================] - 13s 28ms/step - loss: 0.0049 -
accuracy: 0.9987 - val_loss: 0.0420 - val_accuracy: 0.9923
Epoch 16/20
469/469 [==============================] - 14s 29ms/step - loss: 0.0045 -
accuracy: 0.9987 - val_loss: 0.0384 - val_accuracy: 0.9935
Epoch 17/20
469/469 [==============================] - 15s 31ms/step - loss: 0.0031 -
accuracy: 0.9991 - val_loss: 0.0372 - val_accuracy: 0.9931
Epoch 18/20
469/469 [==============================] - 14s 29ms/step - loss: 0.0030 -
accuracy: 0.9991 - val_loss: 0.0484 - val_accuracy: 0.9928
Epoch 19/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0033 -
accuracy: 0.9991 - val_loss: 0.0454 - val_accuracy: 0.9931
Epoch 20/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0030 -
accuracy: 0.9991 - val_loss: 0.0470 - val_accuracy: 0.9921
```

[ ]:
```
<keras.callbacks.History at 0x7f934c448640>
```

[ ]:
```
from urllib.request import urlretrieve
```

```
for i in list(range(1,10)) + ["dog"]:
    urlretrieve(f"https://github.com/milliams/intro_deep_learning/raw/master/
 ↪{i}.png", f"{i}.png")
```

[ ]:
```
import numpy as np
from skimage.io import imread

images = []
for i in list(range(1,10)) + ["dog"]:
    images.append(np.array(imread(f"{i}.png")/255.0, dtype="float32"))
images = np.array(images)[:,:,:,np.newaxis]
images.shape
```

[ ]:
```
(10, 28, 28, 1)
```

[ ]:
```
probabilities = model.predict(images)
```

```
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 42ms/step
```

[ ]:
```
truths = list(range(1, 10)) + ["dog"]

table = []
for truth, probs in zip(truths, probabilities):
    prediction = probs.argmax()
    if truth == 'dog':
        print(f"{truth}. CNN thinks it's a {prediction} ({probs[prediction]*100:
 ↪.1f}%)")
    else:
        print(f"{truth} at {probs[truth]*100:4.1f}%. CNN thinks it's a␣
 ↪{prediction} ({probs[prediction]*100:4.1f}%)")
    table.append((truth, probs))
```

```
1 at 45.1%. CNN thinks it's a 1 (45.1%)
2 at  8.9%. CNN thinks it's a 3 (34.6%)
3 at 26.7%. CNN thinks it's a 3 (26.7%)
4 at  0.0%. CNN thinks it's a 5 (59.8%)
5 at 100.0%. CNN thinks it's a 5 (100.0%)
6 at  1.0%. CNN thinks it's a 5 (37.4%)
7 at 24.0%. CNN thinks it's a 5 (58.7%)
8 at  2.8%. CNN thinks it's a 3 (42.9%)
9 at 22.5%. CNN thinks it's a 3 (29.3%)
dog. CNN thinks it's a 3 (18.6%)
```

1.2.3 Data Augmentation

add inverted images to training data to make the NN more robust to different images (could also do rotated images, &c.)

[ ]:
```
ds_train, ds_test = tfds.load(
    "mnist",
    split=["train", "test"],
    as_supervised=True,
)

def invert_img(image, label):
    return 1.-image, label

ds_train = ds_train.map(normalize_img)
ds_train = ds_train.concatenate(ds_train.map(invert_img))  # new line
ds_train = ds_train.shuffle(1000)
ds_train = ds_train.batch(128)

ds_test = ds_test.map(normalize_img)
ds_test = ds_test.concatenate(ds_test.map(invert_img))  # new line
ds_test = ds_test.batch(128)

model.fit(
    ds_train,
    validation_data=ds_test,
    epochs=3,
)
```

```
Epoch 1/5
  3/938 […] - ETA: 24s - loss: 0.0028 - accuracy:
0.9974

2023-05-25 10:56:51.891885: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_28' with dtype int64 and shape [1]
         [[{{node Placeholder/_28}}]]
2023-05-25 10:56:51.892199: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_0' with dtype string and shape [1]
         [[{{node Placeholder/_0}}]]

937/938 [============================>.] - ETA: 0s - loss: 0.0587 - accuracy:
0.9830

2023-05-25 10:57:14.820890: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_27' with dtype int64 and shape [1]
         [[{{node Placeholder/_27}}]]
2023-05-25 10:57:14.821536: I tensorflow/core/common_runtime/executor.cc:1197]
[/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an
```

```
error and you can ignore this message): INVALID_ARGUMENT: You must feed a value
for placeholder tensor 'Placeholder/_3' with dtype int64 and shape [1]
         [[{{node Placeholder/_3}}]]
```

```
938/938 [==============================] - 25s 26ms/step - loss: 0.0587 -
accuracy: 0.9830 - val_loss: 0.0577 - val_accuracy: 0.9836
Epoch 2/5
938/938 [==============================] - 24s 26ms/step - loss: 0.0281 -
accuracy: 0.9921 - val_loss: 0.0454 - val_accuracy: 0.9880
Epoch 3/5
938/938 [==============================] - 24s 26ms/step - loss: 0.0201 -
accuracy: 0.9942 - val_loss: 0.0531 - val_accuracy: 0.9876
Epoch 4/5
938/938 [==============================] - 24s 26ms/step - loss: 0.0164 -
accuracy: 0.9954 - val_loss: 0.0457 - val_accuracy: 0.9901
Epoch 5/5
938/938 [==============================] - 25s 27ms/step - loss: 0.0133 -
accuracy: 0.9964 - val_loss: 0.0409 - val_accuracy: 0.9914
```

[ ]:
```
<keras.callbacks.History at 0x7f93b0278fa0>
```

[ ]:
```
probabilities = model.predict(images)
```

```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
```

[ ]:
```
truths = list(range(1, 10)) + ["dog"]

table = []
for truth, probs in zip(truths, probabilities):
    prediction = probs.argmax()
    if truth == 'dog':
        print(f"{truth}. CNN thinks it's a {prediction} ({probs[prediction]*100:
 ↪.1f}%)")
    else:
        print(f"{truth} at {probs[truth]*100:4.1f}%. CNN thinks it's a␣
 ↪{prediction} ({probs[prediction]*100:4.1f}%)")
    table.append((truth, probs))
```

```
1 at 67.2%. CNN thinks it's a 1 (67.2%)
2 at 100.0%. CNN thinks it's a 2 (100.0%)
3 at 99.7%. CNN thinks it's a 3 (99.7%)
4 at 100.0%. CNN thinks it's a 4 (100.0%)
5 at 100.0%. CNN thinks it's a 5 (100.0%)
6 at 100.0%. CNN thinks it's a 6 (100.0%)
7 at 99.8%. CNN thinks it's a 7 (99.8%)
8 at 100.0%. CNN thinks it's a 8 (100.0%)
9 at 12.4%. CNN thinks it's a 0 (51.8%)
dog. CNN thinks it's a 8 (32.5%)
```