Neural Networks - intro

Part 1 - XOR

1. Using the XOR dataset below, train (400 epochs) a neural network (NN) using 1, 2, 3, 4, and 5 hidden layers (where each layer has only 2 neurons). For each n layers, store the resulting loss score along with n. Plot the results to find what the optimal number of layers is.
2. Repeat the above with 3 neurons in each Hidden layers. How do these results compare to the 2 neuron layers?
3. Repeat the above with 4 neurons in each Hidden layers. How do these results compare to the 2 and 3 neuron layers?
4. Using the most optimal configuraion (n-layers, k-neurons per layer), compare how `tanh`, `sigmoid`, `softplus` and `relu` effect the loss after 400 epochs. Try other Activation functions as well (https://keras.io/activations/)
5. Again with the most optimal setup, try other optimizers (instead of `SGD`) and report on the loss score. (https://keras.io/optimizers/)

Part 2 - BYOD (Bring your own Dataset)

Using your own dataset, experiment and find the best Neural Network configuration. You may use any resource to improve results, just reference it.

While you may use any dataset, I'd prefer you didn't use the diabetes dataset used in the lesson.

https://stackoverflow.com/questions/34673164/how-to-train-and-tune-an-artificial-multilayer-perceptron-neural-network-using-k

https://keras.io/

In [1]:
```
!pip3 install tensorflow keras
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: tensorflow in /Users/caseymasamitsu/Library/Pyt
hon/3.10/lib/python/site-packages (2.8.0)
Requirement already satisfied: keras in /Users/caseymasamitsu/Library/Python/
3.10/lib/python/site-packages (2.8.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /Users/caseymasamit
su/Library/Python/3.10/lib/python/site-packages (from tensorflow) (4.2.0)
Requirement already satisfied: h5py>=2.9.0 in /Users/caseymasamitsu/Library/Py
thon/3.10/lib/python/site-packages (from tensorflow) (3.6.0)
Requirement already satisfied: tf-estimator-nightly==2.8.0.dev2021122109 in /U
sers/caseymasamitsu/Library/Python/3.10/lib/python/site-packages (from tensorf
low) (2.8.0.dev2021122109)
Requirement already satisfied: google-pasta>=0.1.1 in /Users/caseymasamitsu/Li
brary/Python/3.10/lib/python/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: termcolor>=1.1.0 in /Users/caseymasamitsu/Libra
ry/Python/3.10/lib/python/site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: six>=1.12.0 in /Users/caseymasamitsu/Library/Py
thon/3.10/lib/python/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: gast>=0.2.1 in /Users/caseymasamitsu/Library/Py
thon/3.10/lib/python/site-packages (from tensorflow) (0.5.3)
Requirement already satisfied: opt-einsum>=2.3.2 in /Users/caseymasamitsu/Libr
ary/Python/3.10/lib/python/site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: flatbuffers>=1.12 in /Users/caseymasamitsu/Libr
ary/Python/3.10/lib/python/site-packages (from tensorflow) (2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /Users/caseymasamitsu/Li
brary/Python/3.10/lib/python/site-packages (from tensorflow) (1.44.0)
Requirement already satisfied: protobuf>=3.9.2 in /Users/caseymasamitsu/Librar
y/Python/3.10/lib/python/site-packages (from tensorflow) (3.20.0)
Requirement already satisfied: setuptools in /Library/Frameworks/Python.framew
ork/Versions/3.10/lib/python3.10/site-packages (from tensorflow) (58.1.0)
Requirement already satisfied: libclang>=9.0.1 in /Users/caseymasamitsu/Librar
y/Python/3.10/lib/python/site-packages (from tensorflow) (13.0.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /Users/
caseymasamitsu/Library/Python/3.10/lib/python/site-packages (from tensorflow)
(0.24.0)
Requirement already satisfied: tensorboard<2.9,>=2.8 in /Users/caseymasamitsu/
Library/Python/3.10/lib/python/site-packages (from tensorflow) (2.8.0)
Requirement already satisfied: absl-py>=0.4.0 in /Users/caseymasamitsu/Librar
y/Python/3.10/lib/python/site-packages (from tensorflow) (1.0.0)
Requirement already satisfied: astunparse>=1.6.0 in /Users/caseymasamitsu/Libr
ary/Python/3.10/lib/python/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: numpy>=1.20 in /Users/caseymasamitsu/Library/Py
thon/3.10/lib/python/site-packages (from tensorflow) (1.22.3)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /Users/caseymasam
itsu/Library/Python/3.10/lib/python/site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: wrapt>=1.11.0 in /Users/caseymasamitsu/Library/
Python/3.10/lib/python/site-packages (from tensorflow) (1.14.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /Users/caseymasamitsu/Lib
rary/Python/3.10/lib/python/site-packages (from astunparse>=1.6.0->tensorflow)
(0.37.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /User
s/caseymasamitsu/Library/Python/3.10/lib/python/site-packages (from tensorboar
d<2.9,>=2.8->tensorflow) (0.6.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /Users/case
ymasamitsu/Library/Python/3.10/lib/python/site-packages (from tensorboard<2.9,
>=2.8->tensorflow) (0.4.6)
Requirement already satisfied: requests<3,>=2.21.0 in /Users/caseymasamitsu/Li
brary/Python/3.10/lib/python/site-packages (from tensorboard<2.9,>=2.8->tensor
flow) (2.27.1)
Requirement already satisfied: werkzeug>=0.11.15 in /Users/caseymasamitsu/Libr
```

```
ary/Python/3.10/lib/python/site-packages (from tensorboard<2.9,>=2.8->tensorfl
ow) (2.1.1)
Requirement already satisfied: markdown>=2.6.8 in /Users/caseymasamitsu/Librar
y/Python/3.10/lib/python/site-packages (from tensorboard<2.9,>=2.8->tensorflo
w) (3.3.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in /Users/caseymasamitsu/
Library/Python/3.10/lib/python/site-packages (from tensorboard<2.9,>=2.8->tens
orflow) (2.6.5)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /Users/caseyma
samitsu/Library/Python/3.10/lib/python/site-packages (from tensorboard<2.9,>=
2.8->tensorflow) (1.8.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /Users/caseymasamitsu/
Library/Python/3.10/lib/python/site-packages (from google-auth<3,>=1.6.3->tens
orboard<2.9,>=2.8->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /Users/caseymasamitsu/Library/
Python/3.10/lib/python/site-packages (from google-auth<3,>=1.6.3->tensorboard<
2.9,>=2.8->tensorflow) (4.8)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /Users/caseymasamits
u/Library/Python/3.10/lib/python/site-packages (from google-auth<3,>=1.6.3->te
nsorboard<2.9,>=2.8->tensorflow) (5.0.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /Users/caseymasamit
su/Library/Python/3.10/lib/python/site-packages (from google-auth-oauthlib<0.
5,>=0.4.1->tensorboard<2.9,>=2.8->tensorflow) (1.3.1)
Requirement already satisfied: certifi>=2017.4.17 in /Library/Frameworks/Pytho
n.framework/Versions/3.10/lib/python3.10/site-packages (from requests<3,>=2.2
1.0->tensorboard<2.9,>=2.8->tensorflow) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /Users/caseymasamitsu/Library/P
ython/3.10/lib/python/site-packages (from requests<3,>=2.21.0->tensorboard<2.
9,>=2.8->tensorflow) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /Users/caseymasami
tsu/Library/Python/3.10/lib/python/site-packages (from requests<3,>=2.21.0->te
nsorboard<2.9,>=2.8->tensorflow) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/caseymasamitsu/
Library/Python/3.10/lib/python/site-packages (from requests<3,>=2.21.0->tensor
board<2.9,>=2.8->tensorflow) (1.26.9)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /Users/caseymasamitsu/L
ibrary/Python/3.10/lib/python/site-packages (from pyasn1-modules>=0.2.1->googl
e-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /Users/caseymasamitsu/Librar
y/Python/3.10/lib/python/site-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8->tensorflow) (3.2.0)
```

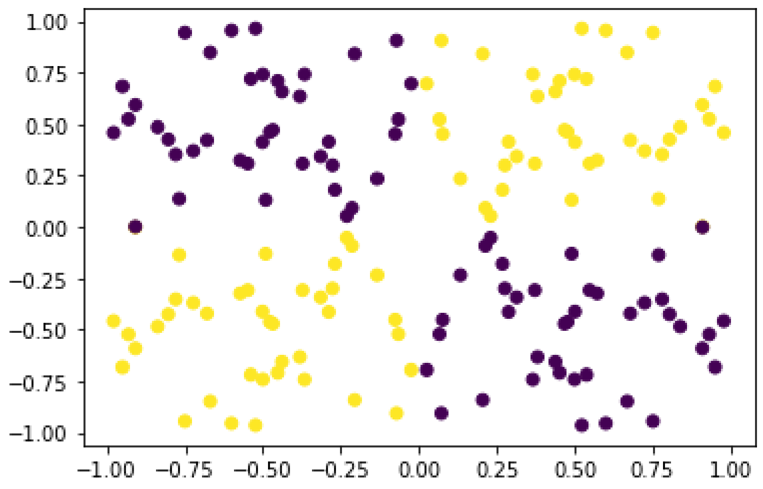
In [3]:
```python
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import SGD  #Stochastic Gradient Descent

import numpy as np
# fix random seed for reproducibility
np.random.seed(7)

import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```python
n = 40
xx = np.random.random((n,1))
yy = np.random.random((n,1))
```

In [5]:
```python
X = np.array([np.array([xx,-xx,-xx,xx]),np.array([yy,-yy,yy,-yy])]).reshape(2,4
y = np.array([np.ones([2*n]),np.zeros([2*n])]).reshape(4*n)
```

In [6]:
```
plt.scatter(*zip(*X), c=y)
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x29aa5a640>
```



In [7]:
```
num_layers = [1,2,3,4,5]
scores = []
for num_layer in num_layers:
    if num_layer == 1:
        model = Sequential()
        model.add(Dense(2, input_dim = 2, activation = 'tanh'))
        model.add(Dense(1, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=
        model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
    if num_layer == 2:
        model = Sequential()
        model.add(Dense(2, input_dim = 2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(1, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=
        model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
    if num_layer == 3:
        model = Sequential()
        model.add(Dense(2, input_dim = 2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(1, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=
        model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
    if num_layer == 4:
        model = Sequential()
        model.add(Dense(2, input_dim = 2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(1, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=
        model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
    if num_layer == 5:
        model = Sequential()
        model.add(Dense(2, input_dim = 2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(2, activation = 'tanh'))
        model.add(Dense(1, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=
        model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)

    score = model.evaluate(X, y)
    scores.append(score)
```

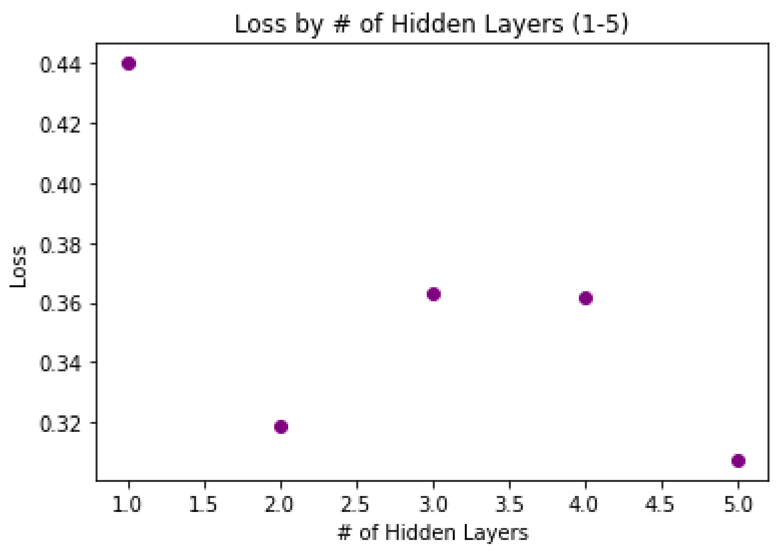
```
Metal device set to: Apple M1 Pro
2022-04-18 15:12:17.727905: I tensorflow/core/common_runtime/pluggable_device/
pluggable_device_factory.cc:305] Could not identify NUMA node of platform GPU
ID 0, defaulting to 0. Your kernel may not have been built with NUMA support.
2022-04-18 15:12:17.728214: I tensorflow/core/common_runtime/pluggable_device/
pluggable_device_factory.cc:271] Created TensorFlow device (/job:localhost/rep
lica:0/task:0/device:GPU:0 with 0 MB memory) -> physical PluggableDevice (devi
ce: 0, name: METAL, pci bus id: <undefined>)
2022-04-18 15:12:17.822187: W tensorflow/core/platform/profile_utils/cpu_util
s.cc:128] Failed to get CPU frequency: 0 Hz
2022-04-18 15:12:17.952926: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 4ms/step - loss: 0.4399 - accuracy:
0.6125
2022-04-18 15:13:33.681170: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-18 15:13:33.880658: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 3ms/step - loss: 0.3190 - accuracy:
0.8813
2022-04-18 15:14:59.645035: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-18 15:14:59.844993: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 4ms/step - loss: 0.3630 - accuracy:
0.6313
2022-04-18 15:16:30.153969: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-18 15:16:30.371859: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 4ms/step - loss: 0.3617 - accuracy:
0.7125
2022-04-18 15:18:07.273458: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2022-04-18 15:18:07.505004: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 4ms/step - loss: 0.3075 - accuracy:
0.8625
2022-04-18 15:19:51.706471: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [8]:

```
columns = list(zip(*scores))
loss = columns[0]
```

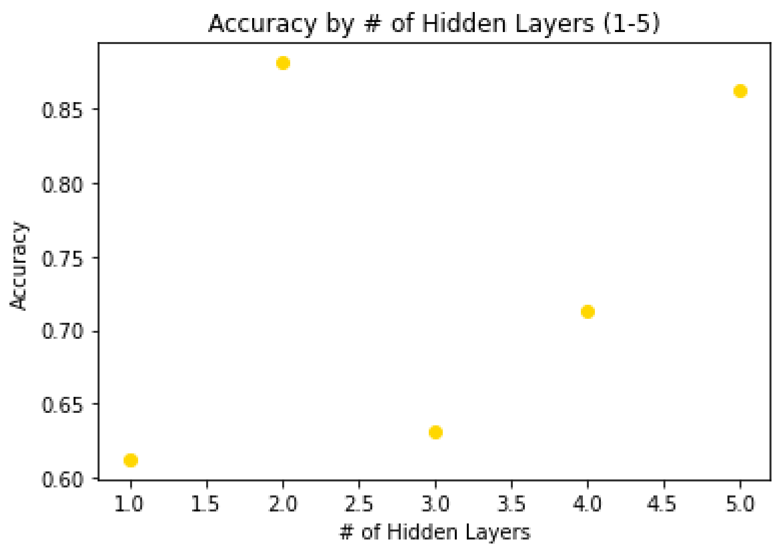
```
plt.scatter(num_layers, loss, c = "purple")
plt.title('Loss by # of Hidden Layers (1-5)')
plt.xlabel("# of Hidden Layers")
plt.ylabel('Loss')
```

Out[8]:
```
Text(0, 0.5, 'Loss')
```



In [9]:
```
accuracy = columns[1]

plt.scatter(num_layers, accuracy, c = "gold")
plt.title('Accuracy by # of Hidden Layers (1-5)')
plt.xlabel("# of Hidden Layers")
plt.ylabel('Accuracy')
```

Out[9]:
```
Text(0, 0.5, 'Accuracy')
```



In both loss and accuracy, 3 hidden layers with 2 neurons is the optimal fit.

Optimizers

Using the most optimal configuraion (n-layers, k-neurons per layer), compare how tanh, sigmoid,softplus and relu effect the loss after 400 epochs.

In [10]:
```
# Tanh
model = Sequential()
model.add(Dense(2, input_dim = 2, activation = 'tanh'))
model.add(Dense(2, activation = 'tanh'))
model.add(Dense(2, activation = 'tanh'))
model.add(Dense(1, activation='tanh'))
sgd = SGD(learning_rate = 0.1)
model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=['accura
model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
tanh = model.evaluate(X, y)
```

```
2022-04-18 15:20:01.904793: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 5ms/step - loss: 3.8662 - accuracy:
0.7438
2022-04-18 15:21:33.816127: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [13]:
```
# Sigmoid
model = Sequential()
model.add(Dense(2, input_dim = 2, activation = 'sigmoid'))
model.add(Dense(2, activation = 'sigmoid'))
model.add(Dense(2, activation = 'sigmoid'))
model.add(Dense(1, activation='sigmoid'))
sgd = SGD(learning_rate = 0.1)
model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=['accura
model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
sigmoid = model.evaluate(X, y)
```

```
2022-04-18 15:23:23.991514: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy:
0.5688
2022-04-18 15:24:52.088996: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [14]:
```
# Softplus
model = Sequential()
model.add(Dense(2, input_dim = 2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(1, activation='softplus'))
sgd = SGD(learning_rate = 0.1)
model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=['accura
model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
softplus = model.evaluate(X, y)
```

```
2022-04-18 15:24:57.238863: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 7ms/step - loss: 0.2754 - accuracy:
0.8875
2022-04-18 15:28:00.595534: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [11]:
```
# Relu
model = Sequential()
model.add(Dense(2, input_dim = 2, activation = 'relu'))
model.add(Dense(2, activation = 'relu'))
model.add(Dense(2, activation = 'relu'))
model.add(Dense(1, activation='relu'))
sgd = SGD(learning_rate = 0.1)
model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=['accura
model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
relu = model.evaluate(X, y)
```

```
2022-04-18 15:21:44.067677: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 3ms/step - loss: 7.7125 - accuracy:
0.5000
2022-04-18 15:23:14.186965: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [15]:
```
print(tanh, sigmoid, softplus, relu)
```

```
[3.86621356010437, 0.7437500357627869] [0.6930978894233704, 0.568750023841857
9] [0.2753864824771881, 0.887499988079071] [7.712474346160889, 0.5]
```

Of the four, softplus has the lowest loss and highest accuracy.

Try other Activation functions as well (https://keras.io/activations/) Again with the most optimal setup, try other optimizers (instead of SGD) and report on the loss score. (https://keras.io/optimizers/)

Optimizing Softplus

In [16]:
```
# Softplus with SGD:
model = Sequential()
model.add(Dense(2, input_dim = 2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(1, activation='softplus'))
sgd = SGD(learning_rate = 0.1)
model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics=['accura
model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
sgd = model.evaluate(X, y)
```

```
2022-04-18 15:28:09.149143: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 7ms/step - loss: 0.2778 - accuracy:
0.8688
2022-04-18 15:31:08.734330: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [17]:
```
# Softplus with RMSprop:
model = Sequential()
model.add(Dense(2, input_dim = 2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(1, activation='softplus'))
sgd = SGD(learning_rate = 0.1)
model.compile(loss = 'binary_crossentropy', optimizer = 'RMSprop', metrics=['ac
model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
RMSprop = model.evaluate(X, y)
2022-04-18 15:31:12.638707: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 7ms/step - loss: 0.3095 - accuracy:
0.8813
2022-04-18 15:34:51.250138: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [18]:

```
# Softplus with Adam:
model = Sequential()
model.add(Dense(2, input_dim = 2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(1, activation='softplus'))
sgd = SGD(learning_rate = 0.1)
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accur
model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
adam = model.evaluate(X, y)
```

```
2022-04-18 15:34:56.631691: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 7ms/step - loss: 0.3017 - accuracy:
0.8813
2022-04-18 15:38:16.705073: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [19]:

```
# Softplus with Adagrad:
model = Sequential()
model.add(Dense(2, input_dim = 2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(2, activation = 'softplus'))
model.add(Dense(1, activation='softplus'))
sgd = SGD(learning_rate = 0.1)
model.compile(loss = 'binary_crossentropy', optimizer = 'adagrad', metrics=['ac
model.fit(X, y, batch_size = 2, epochs = 400, verbose = 0)
adagrad = model.evaluate(X, y)
```

```
2022-04-18 15:38:28.476550: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5/5 [==============================] - 0s 7ms/step - loss: 0.6904 - accuracy:
0.5563
2022-04-18 15:41:30.154040: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

In [20]:

```
print(sgd, RMSprop, adam, adagrad)
```

```
<keras.optimizer_v2.gradient_descent.SGD object at 0x29f6fcbe0> [0.30945071578
02582, 0.8812500238418579] [0.30166906118392944, 0.8812500238418579] [0.690419
1970825195, 0.5562500357627869]
```

It is interesting, sometimes softplus "breaks" between 325-375 epochs and the loss score jumps from ~.50 to 7+. However, if I reduce the epochs to around 300, softplus is consistently the best. Seems there is a lot to play with here.

Using Diabetes data

http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

In [21]:
```
dataset = np.loadtxt("../data/pima-indians-diabetes.data", delimiter=",")
dataset
```

Out[21]:
```
array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
       [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
       [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
       [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
       [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

In [23]:
```
import pandas as pd
df = pd.DataFrame(dataset)
df
```

Out[23]:

	0	1	2	3	4	5	6	7	8
0	6.0	148.0	72.0	35.0	0.0	33.6	0.627	50.0	1.0
1	1.0	85.0	66.0	29.0	0.0	26.6	0.351	31.0	0.0
2	8.0	183.0	64.0	0.0	0.0	23.3	0.672	32.0	1.0
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21.0	0.0
4	0.0	137.0	40.0	35.0	168.0	43.1	2.288	33.0	1.0
...	...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.0	180.0	32.9	0.171	63.0	0.0
764	2.0	122.0	70.0	27.0	0.0	36.8	0.340	27.0	0.0
765	5.0	121.0	72.0	23.0	112.0	26.2	0.245	30.0	0.0
766	1.0	126.0	60.0	0.0	0.0	30.1	0.349	47.0	1.0
767	1.0	93.0	70.0	31.0	0.0	30.4	0.315	23.0	0.0

768 rows × 9 columns

In [24]:
```
X = dataset[:,0:8]
print(X.shape)
X
```

```
(768, 8)
```

```
Out[24]: array([[  6.   , 148.   ,  72.   , ...,  33.6  ,   0.627,  50.   ],
                [  1.   ,  85.   ,  66.   , ...,  26.6  ,   0.351,  31.   ],
                [  8.   , 183.   ,  64.   , ...,  23.3  ,   0.672,  32.   ],
                ...,
                [  5.   , 121.   ,  72.   , ...,  26.2  ,   0.245,  30.   ],
                [  1.   , 126.   ,  60.   , ...,  30.1  ,   0.349,  47.   ],
                [  1.   ,  93.   ,  70.   , ...,  30.4  ,   0.315,  23.   ]])
```

```
In [25]: Y = dataset[:,8]
         print(Y.shape)
         Y
```

```
         (768,)
Out[25]: array([1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 1., 1.,
                1., 0., 1., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 0.,
                0., 0., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0.,
                0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0.,
                0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1.,
                0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0.,
                0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 1., 1., 1., 0., 0.,
                0., 1., 0., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0.,
                0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
                0., 1., 1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0.,
                1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1.,
                1., 1., 1., 0., 0., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 0.,
                0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1.,
                1., 0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 1., 1., 1.,
                1., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
                1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0.,
                0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0.,
                0., 0., 1., 1., 1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 0.,
                1., 0., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 1., 1.,
                1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 1.,
                0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 1.,
                1., 0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0.,
                0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0.,
                1., 0., 0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 0., 1., 0., 1., 0.,
                1., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 1., 0., 0., 0., 0., 1.,
                1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 0.,
                0., 1., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1.,
                0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
                1., 0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0.,
                1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0.,
                1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
                0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 1., 1., 0.,
                0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
                1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1.,
                0., 1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 0., 1., 0., 0.,
                1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 1.,
                1., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0.,
                1., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 1., 1.,
                1., 1., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1., 0.,
                0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1.,
                0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1., 1., 0., 0., 1., 0.,
                0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1.,
                1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 1., 0.,
                1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 0., 1., 0., 1., 0., 0., 0.,
                0., 1., 0.])
```

In [26]:
```
# create model
model = Sequential()
model.add(Dense(16, input_dim=8, activation='tanh'))
model.add(Dense(16, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'
# Fit the model
model.fit(X, Y, epochs=1000, batch_size=10, verbose = 0)
# evaluate the model
scores = model.evaluate(X, Y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
2022-04-18 15:41:55.459088: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
24/24 [==============================] - 0s 3ms/step - loss: 0.4384 - accurac
y: 0.7917

accuracy: 79.17%
2022-04-18 15:47:15.337316: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

Part 2: BYOD - Wine Dataset

In [27]:
```
import pandas as pd
wine = pd.read_csv('../data/WineQT.csv', names=['fixedacid', 'volatileacid','ci
wine = wine.iloc[1: , :]
wine
```

Out[27]:

	fixedacid	volatileacid	citricacid	residualsugar	chlorides	freesulfurdio	totalsulfurdio
1	7.4	0.7	0	1.9	0.076	11	34
2	7.8	0.88	0	2.6	0.098	25	67
3	7.8	0.76	0.04	2.3	0.092	15	54
4	11.2	0.28	0.56	1.9	0.075	17	60
5	7.4	0.7	0	1.9	0.076	11	34
...	...	...	...	...	...	...	...
1138	5.4	0.74	0.09	1.7	0.089	16	26
1139	6.3	0.51	0.13	2.3	0.076	29	40
1140	6.8	0.62	0.08	1.9	0.068	28	38
1141	6.2	0.6	0.08	2	0.09	32	44
1142	5.9	0.55	0.1	2.2	0.062	39	51

1142 rows × 12 columns

In [28]:
```
wine["fixedacid"] = pd.to_numeric(wine.fixedacid, errors='coerce')
wine["volatileacid"] = pd.to_numeric(wine.volatileacid, errors='coerce')
wine["citricacid"] = pd.to_numeric(wine.citricacid, errors='coerce')
wine["residualsugar"] = pd.to_numeric(wine.residualsugar, errors='coerce')
wine["chlorides"] = pd.to_numeric(wine.chlorides, errors='coerce')
```

```
wine["freesulfurdio"] = pd.to_numeric(wine.freesulfurdio, errors='coerce')
wine["totalsulfurdio"] = pd.to_numeric(wine.totalsulfurdio, errors='coerce')
wine["density"] = pd.to_numeric(wine.density, errors='coerce')
wine["pH"] = pd.to_numeric(wine.pH, errors='coerce')
wine["sulphates"] = pd.to_numeric(wine.sulphates, errors='coerce')
wine["alcohol"] = pd.to_numeric(wine.alcohol, errors='coerce')
wine["quality"] = pd.to_numeric(wine.quality, errors='coerce')
```

In [29]:
```
wine.head()
print(wine.dtypes)
```

```
fixedacid          float64
volatileacid       float64
citricacid         float64
residualsugar      float64
chlorides          float64
freesulfurdio      float64
totalsulfurdio     float64
density            float64
pH                 float64
sulphates          float64
alcohol            float64
quality              int64
dtype: object
```

In [30]:
```
wine_array = wine.to_numpy()
wine_array
```

Out[30]:
```
array([[ 7.4 ,  0.7 ,  0.  , ...,  0.56,  9.4 ,  5.  ],
       [ 7.8 ,  0.88,  0.  , ...,  0.68,  9.8 ,  5.  ],
       [ 7.8 ,  0.76,  0.04, ...,  0.65,  9.8 ,  5.  ],
       ...,
       [ 6.8 ,  0.62,  0.08, ...,  0.82,  9.5 ,  6.  ],
       [ 6.2 ,  0.6 ,  0.08, ...,  0.58, 10.5 ,  5.  ],
       [ 5.9 ,  0.55,  0.1 , ...,  0.76, 11.2 ,  6.  ]])
```

In [31]:
```
X = wine_array[:,0:12]
print(X.shape)
X
```

```
(1142, 12)
```

Out[31]:
```
array([[ 7.4 ,  0.7 ,  0.  , ...,  0.56,  9.4 ,  5.  ],
       [ 7.8 ,  0.88,  0.  , ...,  0.68,  9.8 ,  5.  ],
       [ 7.8 ,  0.76,  0.04, ...,  0.65,  9.8 ,  5.  ],
       ...,
       [ 6.8 ,  0.62,  0.08, ...,  0.82,  9.5 ,  6.  ],
       [ 6.2 ,  0.6 ,  0.08, ...,  0.58, 10.5 ,  5.  ],
       [ 5.9 ,  0.55,  0.1 , ...,  0.76, 11.2 ,  6.  ]])
```

In [32]:
```
quality = pd.get_dummies(wine["quality"])
```

In [33]:
```
Y = pd.get_dummies(wine["quality"])
print(Y.shape)
Y
```

```
(1142, 6)
```

Out[33]:

	3	4	5	6	7	8
1	0	0	1	0	0	0
2	0	0	1	0	0	0
3	0	0	1	0	0	0
4	0	0	0	1	0	0
5	0	0	1	0	0	0
...	...	...	...	...	...	...
1138	0	0	0	1	0	0
1139	0	0	0	1	0	0
1140	0	0	0	1	0	0
1141	0	0	1	0	0	0
1142	0	0	0	1	0	0

1142 rows × 6 columns

In [34]:
```
# Create model
model = Sequential()
model.add(Dense(20, input_dim = 12, activation='tanh'))
model.add(Dense(20, input_dim = 12, activation='tanh'))
model.add(Dense(20, input_dim = 12, activation='tanh'))
model.add(Dense(6, activation='sigmoid'))

# Compile model
sgd = SGD(learning_rate = 0.1)
model.compile(loss='CategoricalCrossentropy', optimizer='sgd', metrics=['accura

# Fit the model
model.fit(X, Y, epochs = 500, batch_size = 10, verbose = 0)

# Evaluate the model
scores = model.evaluate(X, Y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
2022-04-18 15:47:26.330155: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
29/36 [======================>......] - ETA: 0s - loss: 0.5612 - accuracy: 0.
7823
2022-04-18 15:52:58.437220: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
36/36 [==============================] - 0s 6ms/step - loss: 0.5576 - accurac
y: 0.7890

accuracy: 78.90%
```

After a while playing around with activations, optimizers, and neurons above, I found the best results with 20 neurons, tanh/sigmoid, and sgd as an optimizer. Below, I ran the same code as above to test it out with 1-5 layers.

In [35]:
```
num_layers = [1,2,3,4,5]
```

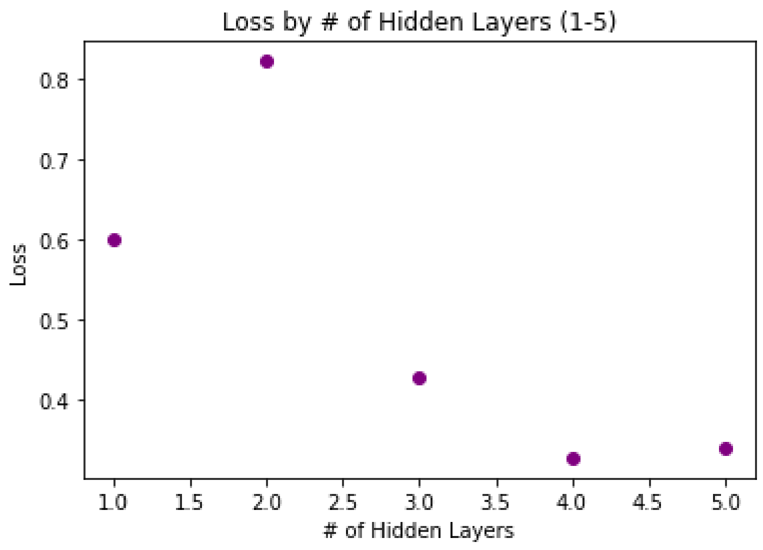
```
scores = []
for num_layer in num_layers:
    if num_layer == 1:
        model = Sequential()
        model.add(Dense(20, input_dim = 12, activation = 'tanh'))
        model.add(Dense(6, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'CategoricalCrossentropy', optimizer = 'sgd', metr
        model.fit(X, Y, batch_size = 10, epochs = 300, verbose = 0)
    if num_layer == 2:
        model = Sequential()
        model.add(Dense(20, input_dim = 12, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(6, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'CategoricalCrossentropy', optimizer = 'sgd', metr
        model.fit(X, Y, batch_size = 10, epochs = 300, verbose = 0)
    if num_layer == 3:
        model = Sequential()
        model.add(Dense(20, input_dim = 12, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(6, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'CategoricalCrossentropy', optimizer = 'sgd', metr
        model.fit(X, Y, batch_size = 10, epochs = 300, verbose = 0)
    if num_layer == 4:
        model = Sequential()
        model.add(Dense(20, input_dim = 12, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(6, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'CategoricalCrossentropy', optimizer = 'sgd', metr
        model.fit(X, Y, batch_size = 10, epochs = 300, verbose = 0)
    if num_layer == 5:
        model = Sequential()
        model.add(Dense(20, input_dim = 12, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(20, activation = 'tanh'))
        model.add(Dense(6, activation='sigmoid'))
        sgd = SGD(learning_rate = 0.1)
        model.compile(loss = 'CategoricalCrossentropy', optimizer = 'sgd', metr
        model.fit(X, Y, batch_size = 10, epochs = 300, verbose = 0)

    score = model.evaluate(X, Y)
    scores.append(score)
```

```
2022-04-18 15:53:48.443395: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
30/36 [========================>.....] - ETA: 0s - loss: 0.6000 - accuracy: 0.
8063
2022-04-18 15:56:47.320419: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
```

```
36/36 [==============================] - 0s 6ms/step - loss: 0.5990 - accurac
y: 0.8135
2022-04-18 15:56:47.681272: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
29/36 [=======================>......] - ETA: 0s - loss: 0.8295 - accuracy: 0.
6584
2022-04-18 15:59:53.576048: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
36/36 [==============================] - 0s 6ms/step - loss: 0.8212 - accurac
y: 0.6602
2022-04-18 15:59:53.963050: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
30/36 [========================>.....] - ETA: 0s - loss: 0.4454 - accuracy: 0.
8635
2022-04-18 16:03:11.149469: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
36/36 [==============================] - 0s 6ms/step - loss: 0.4272 - accurac
y: 0.8687
2022-04-18 16:03:11.544112: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
29/36 [=======================>......] - ETA: 0s - loss: 0.3286 - accuracy: 0.
9106
2022-04-18 16:06:45.120456: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
36/36 [==============================] - 0s 6ms/step - loss: 0.3287 - accurac
y: 0.9116
2022-04-18 16:06:45.540390: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
26/36 [====================>.........] - ETA: 0s - loss: 0.3522 - accuracy: 0.
8702
2022-04-18 16:10:39.376238: I tensorflow/core/grappler/optimizers/custom_graph
_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
36/36 [==============================] - 0s 6ms/step - loss: 0.3406 - accurac
y: 0.8783
```

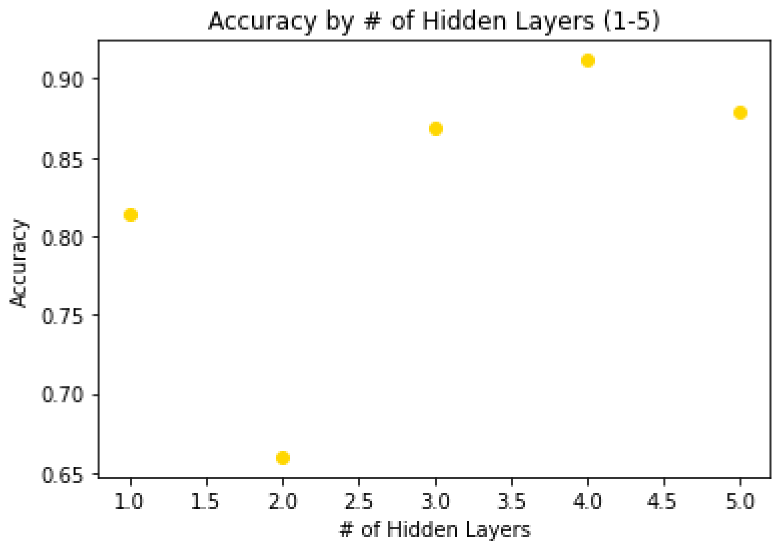
In [36]:
```
columns = list(zip(*scores))
loss = columns[0]

plt.scatter(num_layers, loss, c = "purple")
plt.title('Loss by # of Hidden Layers (1-5)')
plt.xlabel("# of Hidden Layers")
plt.ylabel('Loss')
```

Out[36]:
```
Text(0, 0.5, 'Loss')
```



```
In [37]: accuracy = columns[1]

         plt.scatter(num_layers, accuracy, c = "gold")
         plt.title('Accuracy by # of Hidden Layers (1-5)')
         plt.xlabel("# of Hidden Layers")
         plt.ylabel('Accuracy')
```

```
Out[37]: Text(0, 0.5, 'Accuracy')
```



Four hidden layers with 20 neurons each had the highest accuracy and lowest loss.

```
In [ ]:
```