Deep%2Blearning%2Bbatch%2Bsize%2B32_new

December 10, 2017

1 Table of Contents

2 Installation of tensorflow and keras

```
In [1]: import pip
        pip.main(['install', '-U', 'keras'])
        pip.main(['install', '-U', 'tensorflow'])
```

```
Collecting keras
  Downloading Keras-2.1.2-py2.py3-none-any.whl (304kB)
Requirement already up-to-date: pyyaml in /opt/conda/lib/python3.6/site-packages (from keras)
Collecting numpy>=1.9.1 (from keras)
  Downloading numpy-1.13.3-cp36-cp36m-manylinux1_x86_64.whl (17.0MB)
Collecting scipy>=0.14 (from keras)
  Downloading scipy-1.0.0-cp36-cp36m-manylinux1_x86_64.whl (50.0MB)
Collecting six>=1.9.0 (from keras)
  Downloading six-1.11.0-py2.py3-none-any.whl
Installing collected packages: numpy, scipy, six, keras
  Found existing installation: numpy 1.13.1
    Uninstalling numpy-1.13.1:
      Successfully uninstalled numpy-1.13.1
  Found existing installation: scipy 0.19.1
    Uninstalling scipy-0.19.1:
      Successfully uninstalled scipy-0.19.1
  Found existing installation: six 1.10.0
```

```
    Uninstalling six-1.10.0:
      Successfully uninstalled six-1.10.0
Successfully installed keras-2.1.2 numpy-1.13.3 scipy-1.0.0 six-1.11.0
Collecting tensorflow
  Downloading tensorflow-1.4.1-cp36-cp36m-manylinux1_x86_64.whl (41.2MB)
Collecting numpy>=1.12.1 (from tensorflow)
  Using cached numpy-1.13.3-cp36-cp36m-manylinux1_x86_64.whl
Collecting wheel>=0.26 (from tensorflow)
  Downloading wheel-0.30.0-py2.py3-none-any.whl (49kB)
Collecting six>=1.10.0 (from tensorflow)
  Using cached six-1.11.0-py2.py3-none-any.whl
Collecting protobuf>=3.3.0 (from tensorflow)
  Downloading protobuf-3.5.0.post1-cp36-cp36m-manylinux1_x86_64.whl (6.4MB)
Collecting enum34>=1.1.6 (from tensorflow)
  Downloading enum34-1.1.6-py3-none-any.whl
Collecting tensorflow-tensorboard<0.5.0,>=0.4.0rc1 (from tensorflow)
  Downloading tensorflow_tensorboard-0.4.0rc3-py3-none-any.whl (1.7MB)
Collecting setuptools (from protobuf>=3.3.0->tensorflow)
  Downloading setuptools-38.2.4-py2.py3-none-any.whl (489kB)
Requirement already up-to-date: bleach==1.5.0 in /opt/conda/lib/python3.6/site-packages (from te
Collecting markdown>=2.6.8 (from tensorflow-tensorboard<0.5.0,>=0.4.0rc1->tensorflow)
  Downloading Markdown-2.6.10.zip (414kB)
Collecting werkzeug>=0.11.10 (from tensorflow-tensorboard<0.5.0,>=0.4.0rc1->tensorflow)
  Downloading Werkzeug-0.13-py2.py3-none-any.whl (311kB)
Requirement already up-to-date: html5lib==0.9999999 in /opt/conda/lib/python3.6/site-packages (f
Building wheels for collected packages: markdown
  Running setup.py bdist_wheel for markdown: started
  Running setup.py bdist_wheel for markdown: finished with status 'done'
  Stored in directory: /home/jovyan/.cache/pip/wheels/1e/5a/55/a80b200d12e234d575ad68c1528593d1c
Successfully built markdown
Installing collected packages: numpy, wheel, six, setuptools, protobuf, enum34, markdown, werkze
  Found existing installation: numpy 1.13.1
    Can't uninstall 'numpy'. No files were found to uninstall.
  Found existing installation: wheel 0.29.0
    Uninstalling wheel-0.29.0:
      Successfully uninstalled wheel-0.29.0
  Found existing installation: six 1.10.0
    Can't uninstall 'six'. No files were found to uninstall.
  Found existing installation: setuptools 36.2.2.post20170724
    Uninstalling setuptools-36.2.2.post20170724:
      Successfully uninstalled setuptools-36.2.2.post20170724
Successfully installed enum34-1.1.6 markdown-2.6.10 numpy-1.13.3 protobuf-3.5.0.post1 setuptools
```

Out[1]: 0

3 Importing standard deep learning libraries:

In [2]:
```
import numpy as np
import tensorflow as tf
import scipy
import keras as kr
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization
from matplotlib import pyplot as plt
from keras.layers import Convolution2D, MaxPooling2D
from keras import optimizers
from keras.datasets import mnist
import time
```

```
/opt/conda/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning: compiletime version 3.5 of
  return f(*args, **kwds)
Using TensorFlow backend.
```

In [3]:
```
np.random.seed(123)
# This makes sure that the same set of random variables is generated in each run
#Repeatability is maintained
```

4 Stochastic gradient descent optimizer:

In [4]:
```
t_SGD = time.time()
(X_train_SGD, y_train_SGD), (X_test_SGD, y_test_SGD) = mnist.load_data()

X_train_SGD = X_train_SGD.reshape(X_train_SGD.shape[0], 28, 28,1)
X_test_SGD = X_test_SGD.reshape(X_test_SGD.shape[0], 28, 28,1)
X_train_SGD = X_train_SGD.astype('float32')
X_test_SGD = X_test_SGD.astype('float32')
X_train_SGD /= 255
X_test_SGD /= 255

Y_train_SGD = np_utils.to_categorical(y_train_SGD, 10)
Y_test_SGD = np_utils.to_categorical(y_test_SGD, 10)

network_SGD = Sequential() #defining the type of neural network

#Adding layers to the neural network:
network_SGD.add(Convolution2D(32, 5, 5, activation='relu', input_shape=(28,28,1)))
network_SGD.add(BatchNormalization())
network_SGD.add(MaxPooling2D(pool_size=(2,2)))
network_SGD.add(Convolution2D(32, 5, 5, activation='relu', input_shape=(28,28,1)))
network_SGD.add(BatchNormalization())
network_SGD.add(MaxPooling2D(pool_size=(2,2)))
```

```
        network_SGD.add(BatchNormalization())

        #Post processing of layers:
        network_SGD.add(Dropout(0.25))
        network_SGD.add(Flatten())
        network_SGD.add(Dense(128, activation='relu')) #activation function used
        network_SGD.add(Dropout(0.5))
        network_SGD.add(Dense(10, activation='softmax')) #activation function used

        network_SGD.compile(loss='categorical_crossentropy',optimizer='sgd',metrics=['accuracy']

        results_SGD = network_SGD.fit(X_train_SGD, Y_train_SGD, batch_size=32, nb_epoch=20, verb
        elapsed_SGD = time.time() - t_SGD
        print("Neural network training time is: {0:1.4f}".format(elapsed_SGD),"seconds")
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 3s 0us/step


/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:17: UserWarning: Update your `Conv2
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:20: UserWarning: Update your `Conv2
/opt/conda/lib/python3.6/site-packages/keras/models.py:939: UserWarning: The `nb_epoch` argument
  warnings.warn('The `nb_epoch` argument in `fit` '


Epoch 1/20
60000/60000 [==============================] - 126s 2ms/step - loss: 0.3287 - acc: 0.8962
Epoch 2/20
60000/60000 [==============================] - 123s 2ms/step - loss: 0.1269 - acc: 0.9615
Epoch 3/20
60000/60000 [==============================] - 119s 2ms/step - loss: 0.0990 - acc: 0.9705
Epoch 4/20
60000/60000 [==============================] - 134s 2ms/step - loss: 0.0804 - acc: 0.9757
Epoch 5/20
60000/60000 [==============================] - 130s 2ms/step - loss: 0.0690 - acc: 0.9793
Epoch 6/20
60000/60000 [==============================] - 131s 2ms/step - loss: 0.0649 - acc: 0.9804
Epoch 7/20
60000/60000 [==============================] - 113s 2ms/step - loss: 0.0584 - acc: 0.9816
Epoch 8/20
60000/60000 [==============================] - 129s 2ms/step - loss: 0.0547 - acc: 0.9834
Epoch 9/20
60000/60000 [==============================] - 130s 2ms/step - loss: 0.0489 - acc: 0.9851
Epoch 10/20
60000/60000 [==============================] - 132s 2ms/step - loss: 0.0479 - acc: 0.9853
Epoch 11/20
60000/60000 [==============================] - 130s 2ms/step - loss: 0.0451 - acc: 0.9859
Epoch 12/20
```

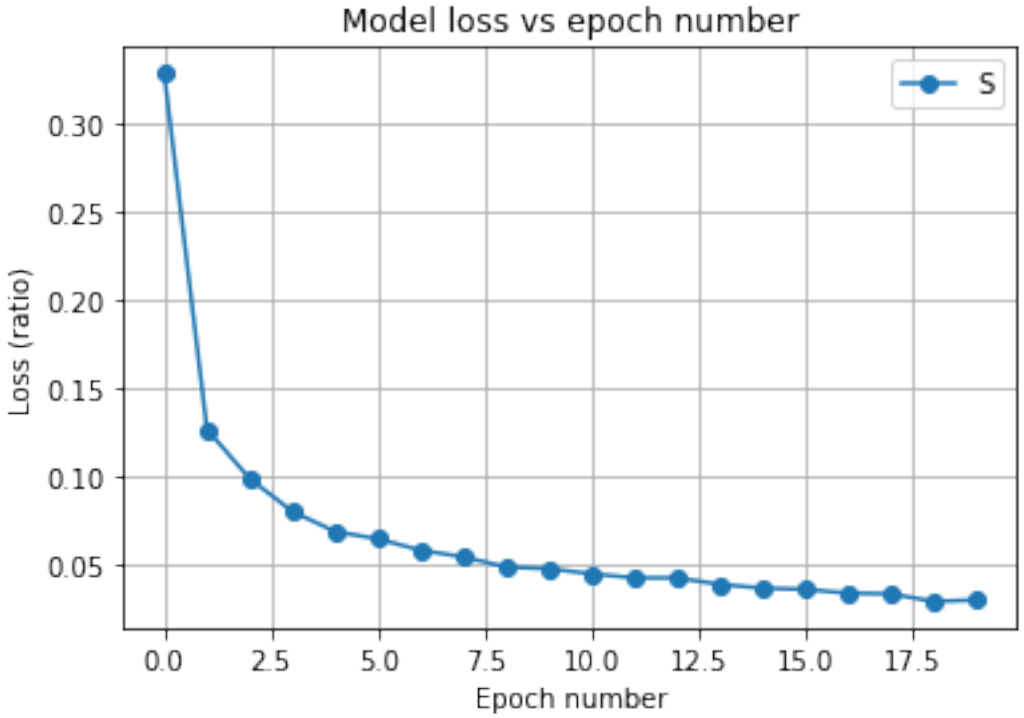
```
60000/60000 [==============================] - 125s 2ms/step - loss: 0.0428 - acc: 0.9871
Epoch 13/20
60000/60000 [==============================] - 132s 2ms/step - loss: 0.0428 - acc: 0.9875
Epoch 14/20
60000/60000 [==============================] - 115s 2ms/step - loss: 0.0391 - acc: 0.9880
Epoch 15/20
60000/60000 [==============================] - 105s 2ms/step - loss: 0.0370 - acc: 0.9884
Epoch 16/20
60000/60000 [==============================] - 107s 2ms/step - loss: 0.0363 - acc: 0.9889
Epoch 17/20
60000/60000 [==============================] - 106s 2ms/step - loss: 0.0341 - acc: 0.9894
Epoch 18/20
60000/60000 [==============================] - 106s 2ms/step - loss: 0.0338 - acc: 0.9894
Epoch 19/20
60000/60000 [==============================] - 106s 2ms/step - loss: 0.0294 - acc: 0.9904
Epoch 20/20
60000/60000 [==============================] - 110s 2ms/step - loss: 0.0303 - acc: 0.9906
Neural network training time is: 2412.8643 seconds
```

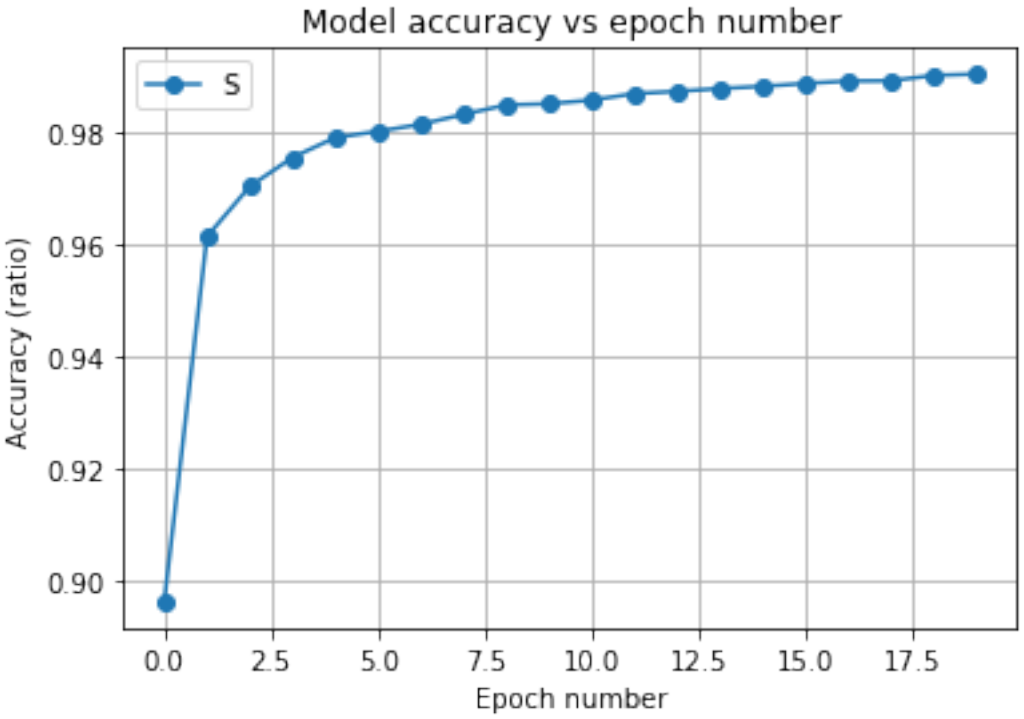
5 Testing trained data (SGD)

```
In [5]: t_test_SGD = time.time()
        score_SGD = network_SGD.evaluate(X_test_SGD, Y_test_SGD, verbose=0)
        print("When evaluated on the MNIST test dataset, the loss is: {0:1.4f}".format(score_SGD
        print("When evaluated on the MNIST test dataset, the acccuracy is: {0:1.4f}".format(scor
        #Loss versus epochs
        plt.plot(results_SGD.history['loss'],'-o')
        plt.title("Model loss vs epoch number")
        plt.xlabel("Epoch number")
        plt.ylabel("Loss (ratio)")
        plt.legend("SGD")
        plt.grid()
        plt.show()

        #Accuracy versus epochs
        plt.plot(results_SGD.history['acc'],'-o')
        plt.xlabel("Epoch number")
        plt.ylabel("Accuracy (ratio)")
        plt.title("Model accuracy vs epoch number")
        plt.legend("SGD")
        plt.grid()
        plt.show()

        elapsed_test_SGD = time.time() - t_test_SGD
        print("Evaluation and plotting runtime is: {0:1.4f}".format(elapsed_test_SGD),"seconds")
```

```
When evaluated on the MNIST test dataset, the loss is: 0.0218
```

```
When evaluated on the MNIST test dataset, the acccuracy is: 0.9929
```





```
Evaluation and plotting runtime is: 8.2898 seconds
```

6 SGD with momentum optimizer:

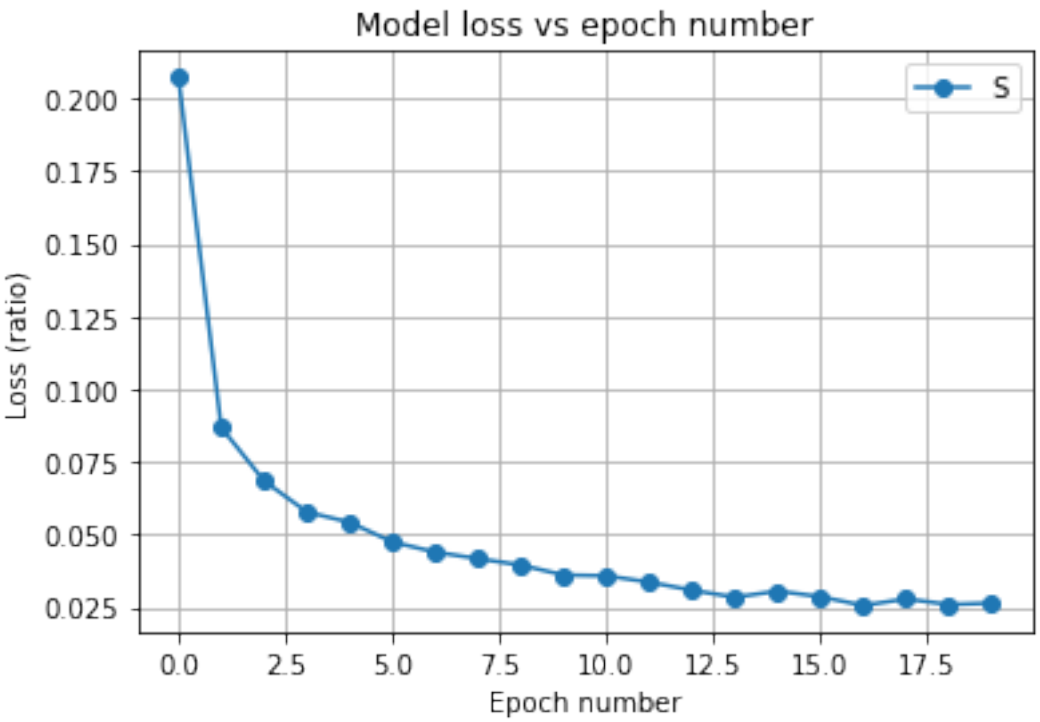
```
In [6]: t_SGD_mom = time.time()
        (X_train_SGD_mom, y_train_SGD_mom), (X_test_SGD_mom, y_test_SGD_mom) = mnist.load_data()

        X_train_SGD_mom = X_train_SGD_mom.reshape(X_train_SGD_mom.shape[0], 28, 28,1)
        X_test_SGD_mom = X_test_SGD_mom.reshape(X_test_SGD_mom.shape[0], 28, 28,1)
        X_train_SGD_mom = X_train_SGD_mom.astype('float32')
        X_test_SGD_mom = X_test_SGD_mom.astype('float32')
        X_train_SGD_mom /= 255
        X_test_SGD_mom /= 255

        Y_train_SGD_mom = np_utils.to_categorical(y_train_SGD_mom, 10)
        Y_test_SGD_mom = np_utils.to_categorical(y_test_SGD_mom, 10)

        network_SGD_mom = Sequential() #defining the type of neural network

        #Adding layers to the neural network:
        network_SGD_mom.add(Convolution2D(32, 5, 5, activation='relu', input_shape=(28,28,1)))
        network_SGD_mom.add(BatchNormalization())
        network_SGD_mom.add(MaxPooling2D(pool_size=(2,2)))
        network_SGD_mom.add(Convolution2D(32, 5, 5, activation='relu', input_shape=(28,28,1)))
        network_SGD_mom.add(BatchNormalization())
        network_SGD_mom.add(MaxPooling2D(pool_size=(2,2)))
        network_SGD_mom.add(BatchNormalization())

        #Post processing of layers:
        network_SGD_mom.add(Dropout(0.25))
        network_SGD_mom.add(Flatten())
        network_SGD_mom.add(Dense(128, activation='relu')) #activation function used
        network_SGD_mom.add(Dropout(0.5))
        network_SGD_mom.add(Dense(10, activation='softmax')) #activation function used

        opt = optimizers.SGD(lr=0.01, momentum=0.95, decay=1e-6, nesterov=False)

        network_SGD_mom.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy

        results_SGD_mom = network_SGD_mom.fit(X_train_SGD_mom, Y_train_SGD_mom, batch_size=32, n
        elapsed_SGD_mom = time.time() - t_SGD_mom
        print("Neural network training time is: {0:1.4f}".format(elapsed_SGD_mom),"seconds")
        # see what happens when you use optimize.minimize
```

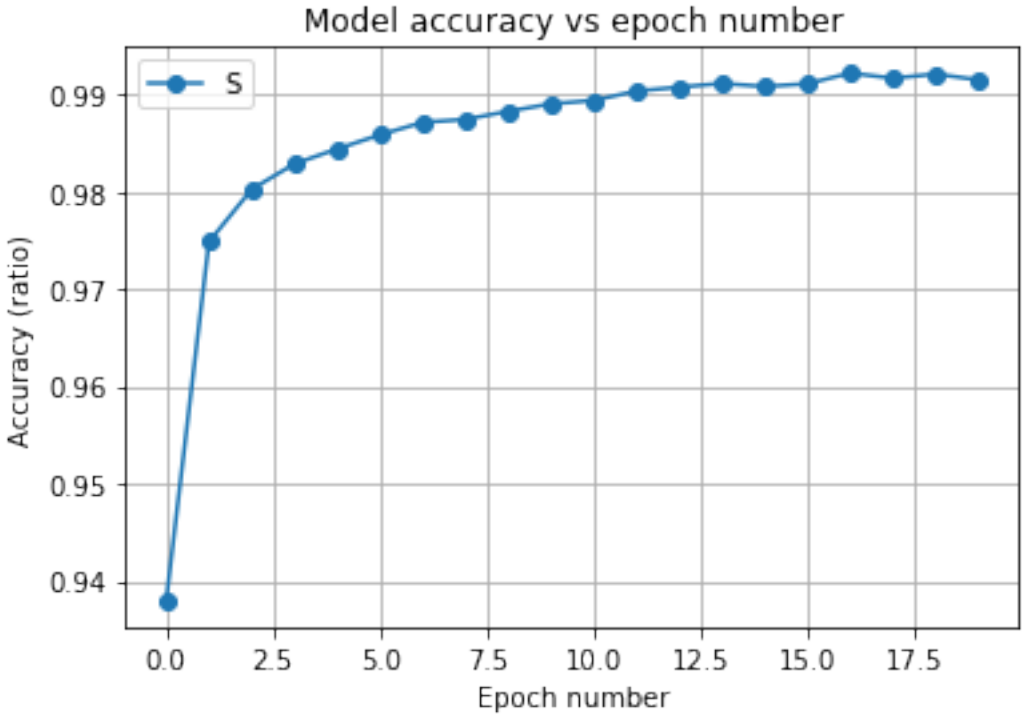
```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:17: UserWarning: Update your `Conv2
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:20: UserWarning: Update your `Conv2
/opt/conda/lib/python3.6/site-packages/keras/models.py:939: UserWarning: The `nb_epoch` argument
  warnings.warn('The `nb_epoch` argument in `fit` '


Epoch 1/20
60000/60000 [==============================] - 115s 2ms/step - loss: 0.2072 - acc: 0.9380
Epoch 2/20
60000/60000 [==============================] - 109s 2ms/step - loss: 0.0868 - acc: 0.9749
Epoch 3/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0685 - acc: 0.9803
Epoch 4/20
60000/60000 [==============================] - 114s 2ms/step - loss: 0.0578 - acc: 0.9829
Epoch 5/20
60000/60000 [==============================] - 107s 2ms/step - loss: 0.0542 - acc: 0.9844
Epoch 6/20
60000/60000 [==============================] - 105s 2ms/step - loss: 0.0475 - acc: 0.9859
Epoch 7/20
60000/60000 [==============================] - 105s 2ms/step - loss: 0.0440 - acc: 0.9871
Epoch 8/20
60000/60000 [==============================] - 106s 2ms/step - loss: 0.0417 - acc: 0.9875
Epoch 9/20
60000/60000 [==============================] - 129s 2ms/step - loss: 0.0395 - acc: 0.9883
Epoch 10/20
60000/60000 [==============================] - 125s 2ms/step - loss: 0.0361 - acc: 0.9890
Epoch 11/20
60000/60000 [==============================] - 121s 2ms/step - loss: 0.0358 - acc: 0.9894
Epoch 12/20
60000/60000 [==============================] - 111s 2ms/step - loss: 0.0337 - acc: 0.9904
Epoch 13/20
60000/60000 [==============================] - 109s 2ms/step - loss: 0.0309 - acc: 0.9908
Epoch 14/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0285 - acc: 0.9912
Epoch 15/20
60000/60000 [==============================] - 113s 2ms/step - loss: 0.0306 - acc: 0.9908
Epoch 16/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0286 - acc: 0.9911
Epoch 17/20
60000/60000 [==============================] - 124s 2ms/step - loss: 0.0256 - acc: 0.9922
Epoch 18/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0279 - acc: 0.9917
Epoch 19/20
60000/60000 [==============================] - 106s 2ms/step - loss: 0.0259 - acc: 0.9921
Epoch 20/20
60000/60000 [==============================] - 107s 2ms/step - loss: 0.0265 - acc: 0.9915
Neural network training time is: 2238.8152 seconds
```

7 Testing trained data (SGD w/ momentum)

In [7]:
```
t_test_SGD_mom = time.time()
score_SGD_mom = network_SGD_mom.evaluate(X_test_SGD_mom, Y_test_SGD_mom, verbose=0)
print("When evaluated on the MNIST test dataset, the loss is: {0:1.4f}".format(score_SGD
print("When evaluated on the MNIST test dataset, the acccuracy is: {0:1.4f}".format(scor
#Loss versus epochs
plt.plot(results_SGD_mom.history['loss'],'-o')
plt.title("Model loss vs epoch number")
plt.xlabel("Epoch number")
plt.ylabel("Loss (ratio)")
plt.legend("SGD with Momentum")
plt.grid()
plt.show()

#Accuracy versus epochs
plt.plot(results_SGD_mom.history['acc'],'-o')
plt.xlabel("Epoch number")
plt.ylabel("Accuracy (ratio)")
plt.title("Model accuracy vs epoch number")
plt.legend("SGD with Momentum")
plt.grid()
plt.show()

elapsed_test_SGD_mom = time.time() - t_test_SGD_mom
print("Evaluation and plotting runtime is: {0:1.4f}".format(elapsed_test_SGD_mom),"secon
```

```
When evaluated on the MNIST test dataset, the loss is: 0.0323
When evaluated on the MNIST test dataset, the acccuracy is: 0.9923
```


Model loss vs epoch number
S
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
Loss (ratio)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch number


Model accuracy vs epoch number
S
0.99
0.98
0.97
0.96
0.95
0.94
Accuracy (ratio)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch number

```
Evaluation and plotting runtime is: 8.1990 seconds
```

8 Adam optimizer (Neural network training):

```
In [8]: t_adam = time.time()
        (X_train_adam, y_train_adam), (X_test_adam, y_test_adam) = mnist.load_data()

        X_train_adam = X_train_adam.reshape(X_train_adam.shape[0], 28, 28,1)
        X_test_adam = X_test_adam.reshape(X_test_adam.shape[0], 28, 28,1)
        X_train_adam = X_train_adam.astype('float32')
        X_test_adam = X_test_adam.astype('float32')
        X_train_adam /= 255
        X_test_adam /= 255

        Y_train_adam = np_utils.to_categorical(y_train_adam, 10)
        Y_test_adam = np_utils.to_categorical(y_test_adam, 10)

        network = Sequential() #defining the type of neural network

        #Adding layers to the neural network:
        network.add(Convolution2D(32, 5, 5, activation='relu', input_shape=(28,28,1)))
        network.add(BatchNormalization())
        network.add(MaxPooling2D(pool_size=(2,2)))
        network.add(Convolution2D(32, 5, 5, activation='relu', input_shape=(28,28,1)))
        network.add(BatchNormalization())
        network.add(MaxPooling2D(pool_size=(2,2)))
        network.add(BatchNormalization())

        #Post processing of layers:
        network.add(Dropout(0.25))
        network.add(Flatten())
        network.add(Dense(128, activation='relu')) #activation function used
        network.add(Dropout(0.5))
        network.add(Dense(10, activation='softmax')) #activation function used

        network.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

        results_adam = network.fit(X_train_adam, Y_train_adam, batch_size=32, nb_epoch=20, verbo
        elapsed_adam = time.time() - t_adam
        print("Neural network training time is: {0:1.4f}".format(elapsed_adam),"seconds")

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:17: UserWarning: Update your `Conv2
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:20: UserWarning: Update your `Conv2
/opt/conda/lib/python3.6/site-packages/keras/models.py:939: UserWarning: The `nb_epoch` argument
  warnings.warn('The `nb_epoch` argument in `fit` '
```

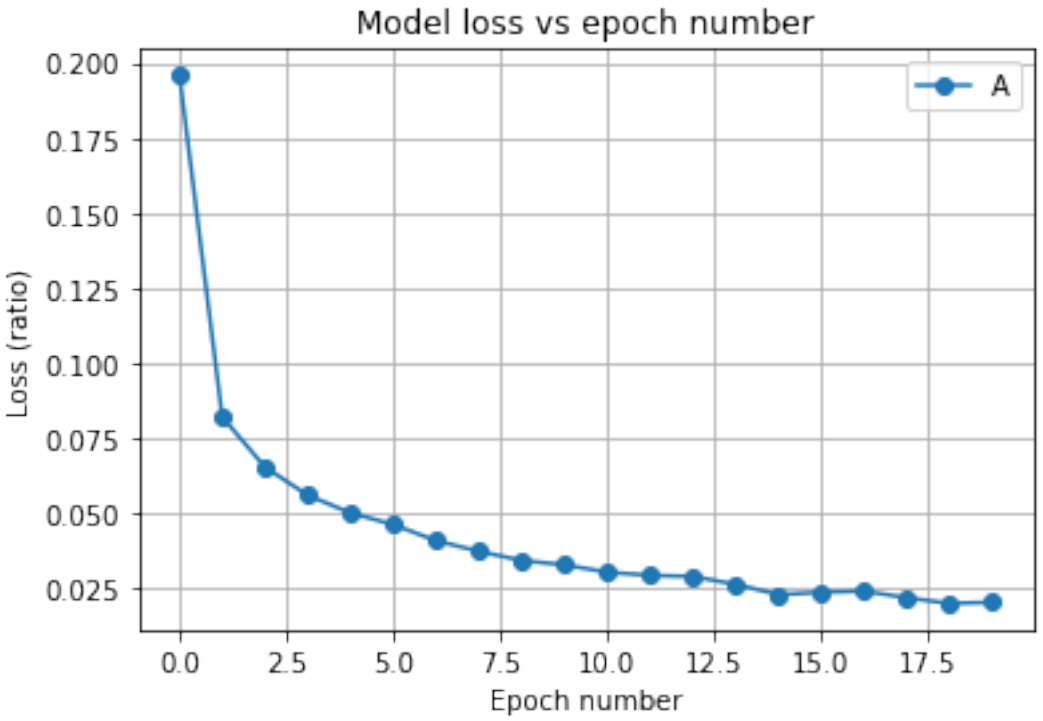
```
Epoch 1/20
60000/60000 [==============================] - 110s 2ms/step - loss: 0.1959 - acc: 0.9401
Epoch 2/20
60000/60000 [==============================] - 109s 2ms/step - loss: 0.0822 - acc: 0.9756
Epoch 3/20
60000/60000 [==============================] - 115s 2ms/step - loss: 0.0655 - acc: 0.9809
Epoch 4/20
60000/60000 [==============================] - 145s 2ms/step - loss: 0.0561 - acc: 0.9829
Epoch 5/20
60000/60000 [==============================] - 119s 2ms/step - loss: 0.0503 - acc: 0.9854
Epoch 6/20
60000/60000 [==============================] - 111s 2ms/step - loss: 0.0464 - acc: 0.9861
Epoch 7/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0410 - acc: 0.9881
Epoch 8/20
60000/60000 [==============================] - 109s 2ms/step - loss: 0.0375 - acc: 0.9887
Epoch 9/20
60000/60000 [==============================] - 114s 2ms/step - loss: 0.0344 - acc: 0.9897
Epoch 10/20
60000/60000 [==============================] - 110s 2ms/step - loss: 0.0330 - acc: 0.9901
Epoch 11/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0306 - acc: 0.9903
Epoch 12/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0295 - acc: 0.9908
Epoch 13/20
60000/60000 [==============================] - 111s 2ms/step - loss: 0.0291 - acc: 0.9912
Epoch 14/20
60000/60000 [==============================] - 115s 2ms/step - loss: 0.0265 - acc: 0.9920
Epoch 15/20
60000/60000 [==============================] - 120s 2ms/step - loss: 0.0231 - acc: 0.9929
Epoch 16/20
60000/60000 [==============================] - 124s 2ms/step - loss: 0.0239 - acc: 0.9923
Epoch 17/20
60000/60000 [==============================] - 126s 2ms/step - loss: 0.0243 - acc: 0.9926
Epoch 18/20
60000/60000 [==============================] - 122s 2ms/step - loss: 0.0221 - acc: 0.9934
Epoch 19/20
60000/60000 [==============================] - 122s 2ms/step - loss: 0.0202 - acc: 0.9937
Epoch 20/20
60000/60000 [==============================] - 121s 2ms/step - loss: 0.0206 - acc: 0.9935
Neural network training time is: 2327.2067 seconds
```

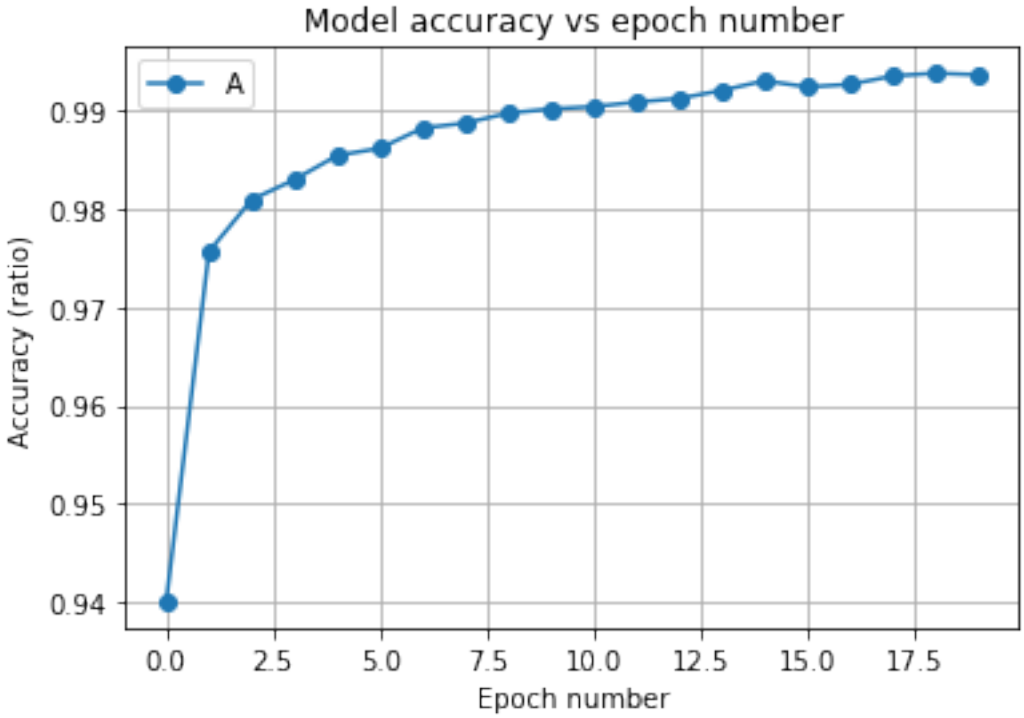
9 Testing trained data (adam)

```
In [9]: t_test_adam = time.time()
        score_adam = network.evaluate(X_test_adam, Y_test_adam, verbose=0)
```

```
print("When evaluated on the MNIST test dataset, the loss is: {0:1.4f}".format(score_ada
print("When evaluated on the MNIST test dataset, the acccuracy is: {0:1.4f}".format(scor
#Loss versus epochs
plt.plot(results_adam.history['loss'],'-o')
plt.title("Model loss vs epoch number")
plt.xlabel("Epoch number")
plt.ylabel("Loss (ratio)")
plt.legend("Adam")
plt.grid()
plt.show()

#Accuracy versus epochs
plt.plot(results_adam.history['acc'],'-o')
plt.xlabel("Epoch number")
plt.ylabel("Accuracy (ratio)")
plt.title("Model accuracy vs epoch number")
plt.legend("Adam")
plt.grid()
plt.show()

elapsed_test_adam = time.time() - t_test_adam
print("Evaluation and plotting runtime is: {0:1.4f}".format(elapsed_test_adam),"seconds"
```

```
When evaluated on the MNIST test dataset, the loss is: 0.0244
When evaluated on the MNIST test dataset, the acccuracy is: 0.9932
```






```
Evaluation and plotting runtime is: 8.6509 seconds
```

10 Adadelta optimizer

```
In [10]: t_adadelta = time.time()
         (X_train_adadelta, y_train_adadelta), (X_test_adadelta, y_test_adadelta) = mnist.load_d

         X_train_adadelta = X_train_adadelta.reshape(X_train_adadelta.shape[0], 28, 28,1)
         X_test_adadelta = X_test_adadelta.reshape(X_test_adadelta.shape[0], 28, 28,1)
         X_train_adadelta = X_train_adadelta.astype('float32')
         X_test_adadelta = X_test_adadelta.astype('float32')
         X_train_adadelta /= 255
         X_test_adadelta /= 255

         Y_train_adadelta = np_utils.to_categorical(y_train_adadelta, 10)
         Y_test_adadelta = np_utils.to_categorical(y_test_adadelta, 10)

         network_adadelta = Sequential() #defining the type of neural network
```

```
        #Adding layers to the neural network:
        network_adadelta.add(Convolution2D(32, 5, 5, activation='relu', input_shape=(28,28,1)))
        network_adadelta.add(BatchNormalization())
        network_adadelta.add(MaxPooling2D(pool_size=(2,2)))
        network_adadelta.add(Convolution2D(32, 5, 5, activation='relu', input_shape=(28,28,1)))
        network_adadelta.add(BatchNormalization())
        network_adadelta.add(MaxPooling2D(pool_size=(2,2)))
        network_adadelta.add(BatchNormalization())

        #Post processing of layers:
        network_adadelta.add(Dropout(0.25))
        network_adadelta.add(Flatten())
        network_adadelta.add(Dense(128, activation='relu')) #activation function used
        network_adadelta.add(Dropout(0.5))
        network_adadelta.add(Dense(10, activation='softmax')) #activation function used

        opt = optimizers.SGD(lr=0.01, momentum=0.95, decay=1e-6, nesterov=False)

        network_adadelta.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accura

        results_adadelta = network_adadelta.fit(X_train_adadelta, Y_train_adadelta, batch_size=
        elapsed_adadelta = time.time() - t_adadelta
        print("Neural network training time is: {0:1.4f}".format(elapsed_adadelta),"seconds")
        # see what happens when you use optimize.minimize
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:17: UserWarning: Update your `Conv2
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:20: UserWarning: Update your `Conv2
/opt/conda/lib/python3.6/site-packages/keras/models.py:939: UserWarning: The `nb_epoch` argument
  warnings.warn('The `nb_epoch` argument in `fit` '


Epoch 1/20
60000/60000 [==============================] - 119s 2ms/step - loss: 0.1980 - acc: 0.9405
Epoch 2/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0870 - acc: 0.9748
Epoch 3/20
60000/60000 [==============================] - 109s 2ms/step - loss: 0.0698 - acc: 0.9800
Epoch 4/20
60000/60000 [==============================] - 114s 2ms/step - loss: 0.0591 - acc: 0.9826
Epoch 5/20
60000/60000 [==============================] - 104s 2ms/step - loss: 0.0528 - acc: 0.9844
Epoch 6/20
60000/60000 [==============================] - 109s 2ms/step - loss: 0.0474 - acc: 0.9863
Epoch 7/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0447 - acc: 0.9871
Epoch 8/20
60000/60000 [==============================] - 118s 2ms/step - loss: 0.0414 - acc: 0.9877
Epoch 9/20
```

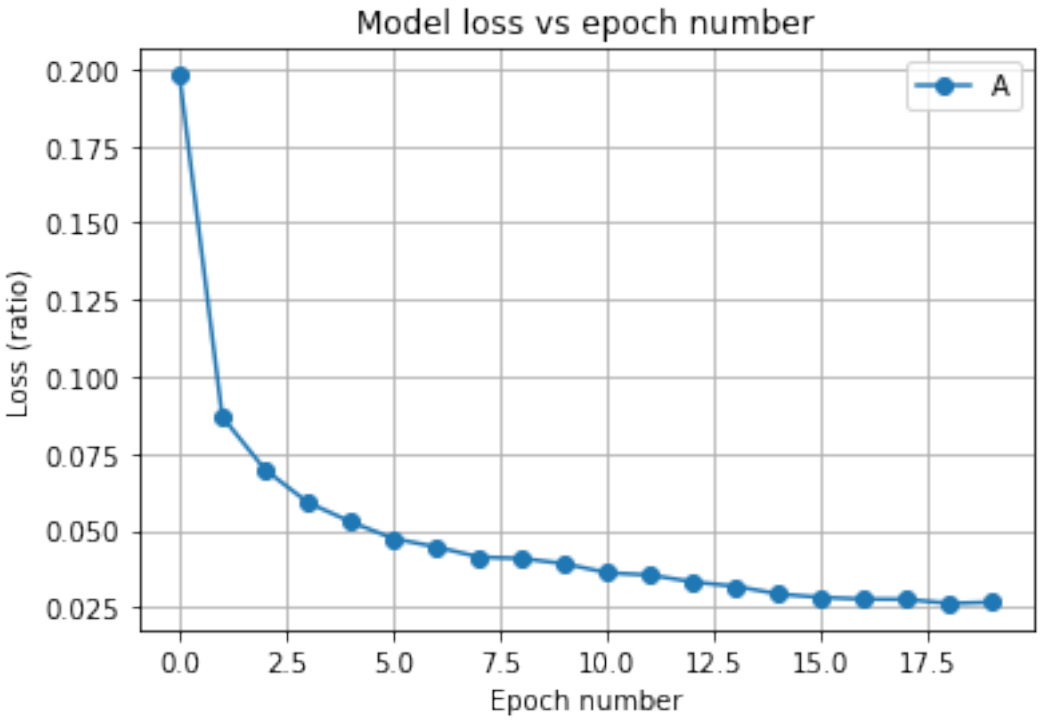
```
60000/60000 [==============================] - 112s 2ms/step - loss: 0.0409 - acc: 0.9878
Epoch 10/20
60000/60000 [==============================] - 107s 2ms/step - loss: 0.0392 - acc: 0.9878
Epoch 11/20
60000/60000 [==============================] - 107s 2ms/step - loss: 0.0363 - acc: 0.9890
Epoch 12/20
60000/60000 [==============================] - 107s 2ms/step - loss: 0.0356 - acc: 0.9891
Epoch 13/20
60000/60000 [==============================] - 106s 2ms/step - loss: 0.0333 - acc: 0.9899
Epoch 14/20
60000/60000 [==============================] - 107s 2ms/step - loss: 0.0319 - acc: 0.9899
Epoch 15/20
60000/60000 [==============================] - 104s 2ms/step - loss: 0.0294 - acc: 0.9909
Epoch 16/20
60000/60000 [==============================] - 106s 2ms/step - loss: 0.0283 - acc: 0.9914
Epoch 17/20
60000/60000 [==============================] - 107s 2ms/step - loss: 0.0277 - acc: 0.9919
Epoch 18/20
60000/60000 [==============================] - 105s 2ms/step - loss: 0.0277 - acc: 0.9919
Epoch 19/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0263 - acc: 0.9917
Epoch 20/20
60000/60000 [==============================] - 108s 2ms/step - loss: 0.0267 - acc: 0.9919
Neural network training time is: 2175.5490 seconds
```

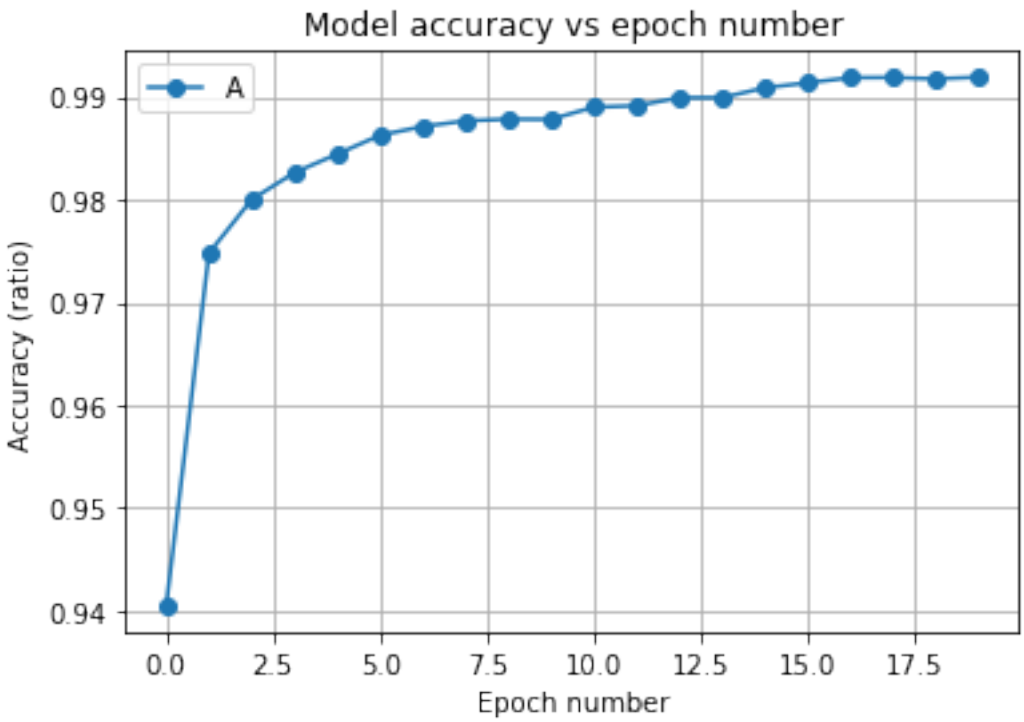
11 Testing trained data (Adadelta)

```
In [11]: t_test_adadelta = time.time()
         score_adadelta = network_adadelta.evaluate(X_test_adadelta, Y_test_adadelta, verbose=0)
         print("When evaluated on the MNIST test dataset, the loss is: {0:1.4f}".format(score_ad
         print("When evaluated on the MNIST test dataset, the acccuracy is: {0:1.4f}".format(sco
         #Loss versus epochs
         plt.plot(results_adadelta.history['loss'],'-o')
         plt.title("Model loss vs epoch number")
         plt.xlabel("Epoch number")
         plt.ylabel("Loss (ratio)")
         plt.legend("Adadelta")
         plt.grid()
         plt.show()

         #Accuracy versus epochs
         plt.plot(results_adadelta.history['acc'],'-o')
         plt.xlabel("Epoch number")
         plt.ylabel("Accuracy (ratio)")
         plt.title("Model accuracy vs epoch number")
         plt.legend("Adadelta")
```

```
        plt.grid()
        plt.show()

        elapsed_test_adadelta = time.time() - t_test_adadelta
        print("Evaluation and plotting runtime is: {0:1.4f}".format(elapsed_test_adadelta),"sec
```

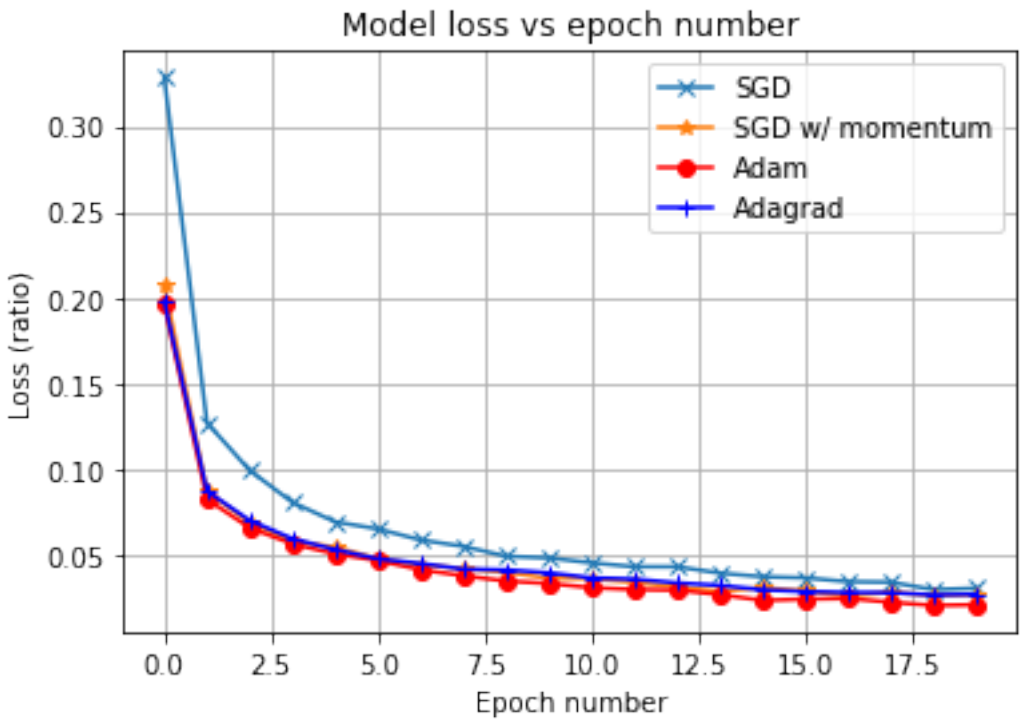
```
When evaluated on the MNIST test dataset, the loss is: 0.0243
When evaluated on the MNIST test dataset, the acccuracy is: 0.9928
```



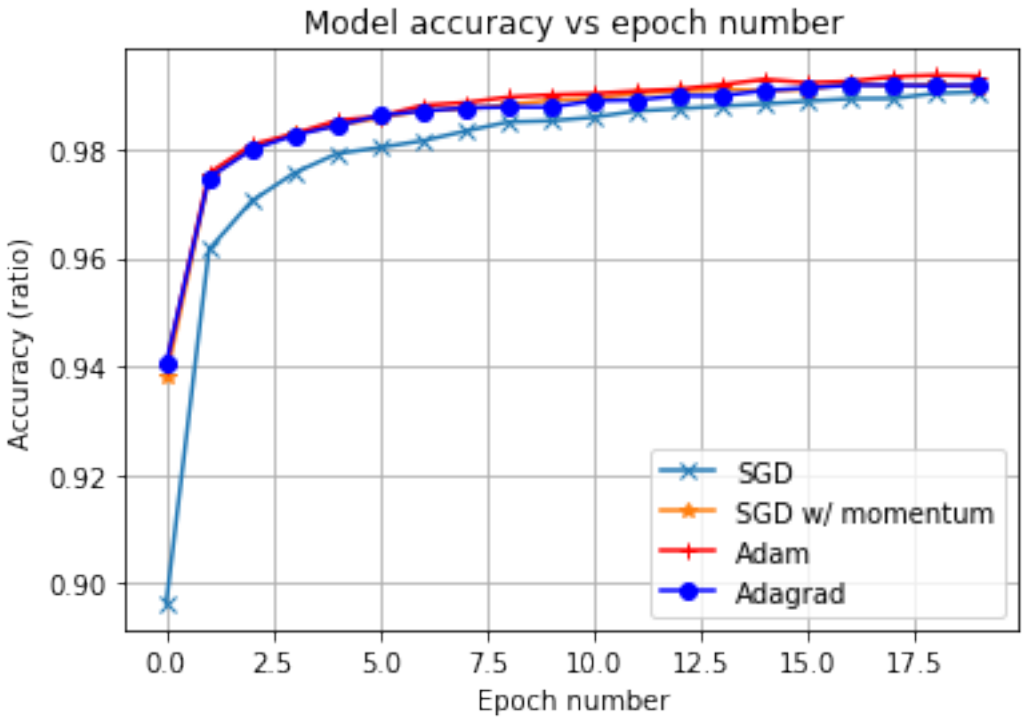



```
Evaluation and plotting runtime is: 8.2756 seconds
```

```
In [12]: plt.plot(results_SGD.history['loss'],'-x',results_SGD_mom.history['loss'],'-*',results_
         plt.title("Model loss vs epoch number")
         plt.xlabel("Epoch number")
         plt.ylabel("Loss (ratio)")
         plt.legend(['SGD','SGD w/ momentum','Adam','Adagrad'])
         plt.grid()
         plt.show()
```



In [13]:
```
plt.plot(results_SGD.history['acc'],'-x',results_SGD_mom.history['acc'],'-*',results_ad
plt.xlabel("Epoch number")
plt.ylabel("Accuracy (ratio)")
plt.title("Model accuracy vs epoch number")
plt.legend(['SGD','SGD w/ momentum','Adam','Adagrad'])
plt.grid()
plt.show()
```


Model accuracy vs epoch number
Accuracy (ratio)
0.98
0.96
0.94
0.92
0.90
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch number
SGD
SGD w/ momentum
Adam
Adagrad