Lecture 25 - Classification

Part 1 - Classification with TensorFlow - https://youtu.be/_Md2wy6Uzx8
Part 2 - Classification with TensorFlow - https://youtu.be/iDedralOejw

From Amazon S3 - for download (same as youtube videos)

Part 1 - Classification with TensorFlow
Part 2 - Classification with TensorFlow

When will we have self driving cars / Self driving trucks?

Example of Clasification

Show video for 90 seconds.

Trucks First

According to BLS 5.1 million full time truck drivers in the US making on average $37,770.00 a year. Multiply: $192,627,000,000.00 -- That's a bunch. Now burden the cost. The employer has to pay health insurance, taxes on labor, workmen's comp etc. 1.42 is a fairly good estimate. That gives us: $273,530,340,000.00 (274 Billion US Dollars) - that is in the US alone. 327 Million - Europe is over 450 million.

I hypothesize that if you can build a self-driving truck for the US you can do it for France, etc.

Suppose that you only manage long haul trucking - This is 87.2% of trucking.

So just to fix this chunk: $238,518,456,480.00 (239 Billion).

"A major factor for businesses in choosing self-driving trucks is greater fuel efficiency, which cuts fuel costs by at least 15%…" "There's no question that autonomous trucks will be ready before autonomous cars."

From: CNBC

Right now fresh vegetables are being delivered from southern California to Texas using self driving on I-10. UPS is testing in Arizona. JB Hunt is testing in Virginia etc…

Will truck driving jobs disappear?

Yes and No: Some will - look at the vinyl LP industry - in 1980 150,000 jobs, today less than 5,000. Solving long-haul trucking will still leave local. What will happen is the cost of shipping will drop and more shipping will take place. There will be more local drivers that work with loading and un-loading.

It won't happen all at once - but it will happen fast enough to be scary. The 2008 recession saw 2.6 million job lost in 2008. 50% of 5.1 million truck drives is around 2.6 million!

Cars - Taxies

Between Ford, Uber, Alphabet (Waymo), General Maters, Intel, Daimler Chrysler (Mercedes Benz), Toyota and Hyundai have put a combined $24 billion into self driving cars.

Announcements from Ford say 2021, from GM and Toyota say 2023 for full self driving. Google bought 140,000 cars for Phoenix as self driving taxies that are now full level 3 - self driving right now. No driver at all.

What is the "real" cost of ownership of a car - and can you live with just "transpiration" instead. Average payments on a car are $682.00 --- according to Motor Trends. Add in $100 for maintenance, $100 for insurance, $200 for parking - this leaves out licensing - and you get around $1100.00 a month. This is for around 12 hours a week of driving time - so $90.00 per hour. For example one LA person reports that they saved $423 a month by getting rid of 1 of 2 family cars and just using Uber 5 days a week to get to work. Considering that the average American family reports a "disposable" income of $92.01 a month this is a "huge" raise.

An average Uber trip (from the annual report) is $49.44. So most people if they can pay for an average Uber would be better off to Uber into and back from work.

Also Uber reports that 75% of costs are for drivers. So if you are a self-driving company you can eliminate most of those costs.

So let's look at the potential: Uber reports $49 a ride for 10 rides a day - $490.00 a day with 75% profit. That's $362 a day in profit. Say 20 days a month for 5 day a week use - you may get less or more on the weekends.

That's $7350.00 a month in profit per car. This leaves lots of time for maintenance - refiling etc. Waymo's purchase of 140000 cars works out to a little over $10 billion a year in profit for 1 US city for just taxies - think of all the commuters.

If you think this is a lot - then use the $1.00 to $2.00 a mile that Uber charges and you get a monthly profit of more like $14,000.000 a month.

As an individual purchasing a car that is used for a Uber-like service would have a quick payback - and - potentially a large profit for anybody that had a self-driving car that they could rent out.

The entire concept of “car ownership” will change. Cars set 92% idle. Parking is expensive etc. If you own a self-driving car you may “lease” it out when you are not using it as a Taxi - Tesla has in the ownership/license agreement that they get a “cut” of the rental if you do this.

Remember that big business that are super-successful, like Kodak, Blockbuster, Radio Shack are all “gone”. So yea… This is going to have a huge impact. GM and Ford are the 11th and 12th largest companies in the US. If ½ of the market for cars goes away what happens?

Classification is the single most important algorithm

All self driving is based on using “classification” to identify objects.

You may need to install

```
$ pip install -q pyyaml h5py
```

So that you can save in h5 format for the model.

train-model.py

```
# From: https://www.tensorflow.org/tutorials/images/classification

# mport packages
# ----------------------
# start by importing the required packages. The os package is used to read files
# and directory structure, NumPy is used to convert python list to numpy array and
# to perform required matrix operations and matplotlib.pyplot to plot the graph
# and display images in the training and validation data.

from __future__ import absolute_import, division, print_function, unicode_literals

# Import Tensorflow and the Keras classes needed to construct our model.

from datetime import datetime
from packaging import version

import tensorflow as tf

print("TensorFlow version: ", tf.__version__)
assert version.parse(tf.__version__).release[0] >= 2, "This program TensorFlow 2.0 or a

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
import os
import numpy as np
import matplotlib.pyplot as plt

# Load data
# ----------------------
# Begin by downloading the dataset. This tutorial uses a filtered version of Dogs
# vs Cats dataset from Kaggle. Download the archive version of the dataset and
# store it in the "/tmp/" directory.

_URL = 'https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip'

path_to_zip = tf.keras.utils.get_file('cats_and_dogs.zip', origin=_URL, extract=True)

PATH = os.path.join(os.path.dirname(path_to_zip), 'cats_and_dogs_filtered')

# The dataset has the following directory structure:
#
#cats_and_dogs_filtered
#|__ train
#    |______ cats: [cat.0.jpg, cat.1.jpg, cat.2.jpg ....]
#    |______ dogs: [dog.0.jpg, dog.1.jpg, dog.2.jpg ...]
#|__ validation
#    |______ cats: [cat.2000.jpg, cat.2001.jpg, cat.2002.jpg ....]
#    |______ dogs: [dog.2000.jpg, dog.2001.jpg, dog.2002.jpg ...]
# After extracting its contents, assign variables with the proper file path for
# the training and validation set.


train_dir = os.path.join(PATH, 'train')
validation_dir = os.path.join(PATH, 'validation')

train_cats_dir = os.path.join(train_dir, 'cats')  # our training cat pictures
train_dogs_dir = os.path.join(train_dir, 'dogs')  # our training dog pictures
validation_cats_dir = os.path.join(validation_dir, 'cats')  # our validation cat pictur
validation_dogs_dir = os.path.join(validation_dir, 'dogs')  # our validation dog pictur

# Understand the data
# ----------------------
# Let's look at how many cats and dogs images are in the training and
# validation directory:

num_cats_tr = len(os.listdir(train_cats_dir))
num_dogs_tr = len(os.listdir(train_dogs_dir))

num_cats_val = len(os.listdir(validation_cats_dir))
num_dogs_val = len(os.listdir(validation_dogs_dir))

total_train = num_cats_tr + num_dogs_tr
total_val = num_cats_val + num_dogs_val

print('total training cat images:', num_cats_tr)
print('total training dog images:', num_dogs_tr)

print('total validation cat images:', num_cats_val)
print('total validation dog images:', num_dogs_val)
print("--")
print("Total training images:", total_train)
print("Total validation images:", total_val)

# For convenience, set up variables to use while pre-processing the dataset and
# training the network.

batch_size = 128
epochs = 15
IMG_HEIGHT = 150
IMG_WIDTH = 150



# Data preparation
# ----------------------
# Format the images into appropriately pre-processed floating point tensors
# before feeding to the network

# Read images from the disk.
# Decode contents of these images and convert it into proper grid format as
# per their RGB content.
#

# Convert them into floating point tensors.
# Rescale the tensors from values between 0 and 255 to values between 0 and 1, as
# neural networks prefer to deal with small input values. Fortunately, all these
# tasks can be done with the ImageDataGenerator class provided by tf.keras. It can
# read images from disk and preprocess them into proper tensors. It will also set
# up generators that convert these images into batches of tensors—helpful when
# training the network.

train_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our training
validation_image_generator = ImageDataGenerator(rescale=1./255) # Generator for our val

# After defining the generators for training and validation images, the
# flow_from_directory method load images from the disk, applies rescaling, and
# resizes the images into the required dimensions.

train_data_gen = train_image_generator.flow_from_directory(batch_size=batch_size,
     directory=train_dir,
     shuffle=True,
     target_size=(IMG_HEIGHT, IMG_WIDTH),
     class_mode='binary')

val_data_gen = validation_image_generator.flow_from_directory(batch_size=batch_size,
     directory=validation_dir,
     target_size=(IMG_HEIGHT, IMG_WIDTH),
     class_mode='binary')

# Visualize training images
# ----------------------
# Visualize the training images by extracting a batch of images from the
# training generator—which is 32 images in this example—then plot five of
# them with matplotlib.

sample_training_images, _ = next(train_data_gen)

# The next function returns a batch from the dataset. The return value of next
# function is in form of (x_train, y_train) where x_train is training features
# and y_train, its labels. Discard the labels to only visualize the training
# images.

# This function will plot images in the form of a grid with 1 row and 5 columns
# where images are placed in each column.
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()

plotImages(sample_training_images[:5])



# Revised with image manipulation.
# ----------------------------

# Apply all the previous augmentations. Here, you applied rescale, 45 degree
# rotation, width shift, height shift, horizontal flip and zoom augmentation to
# the training images.

image_gen_train = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=45,
                    width_shift_range=.15,
                    height_shift_range=.15,
                    horizontal_flip=True,
                    zoom_range=0.5
                    )

train_data_gen = image_gen_train.flow_from_directory(batch_size=batch_size,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_HEIGHT, IMG_WIDTH
                                                     class_mode='binary')
```

```
# Visualize how a single image would look five different times when passing these augme

augmented_images = [train_data_gen[0][0][0] for i in range(5)]
plotImages(augmented_images)

# Create validation data generator
# Generally, only apply data augmentation to the training examples. In this case, only

image_gen_val = ImageDataGenerator(rescale=1./255)

val_data_gen = image_gen_val.flow_from_directory(batch_size=batch_size,
                                                 directory=validation_dir,
                                                 target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                 class_mode='binary')




# Create the model
# -----------------------------
# The model consists of three convolution blocks with a max pool layer in each
# of them. There's a fully connected layer with 512 units on top of it that is
# activated by a relu activation function. The model outputs class
# probabilities based on binary classification by the sigmoid activation
# function.

model = Sequential([

    Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])



# Compile the model
# -----------------------------
# For this tutorial, choose the ADAM optimizer and binary cross entropy loss
# function. To view training and validation accuracy for each training epoch,
# pass the metrics argument.

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Model summary
# -----------------------------
# View all the layers of the network using the model's summary method:

model.summary()

# Train the model
# ----------------------------
# Use the fit_generator method of the ImageDataGenerator class to train the network.

history = model.fit_generator(
    train_data_gen,
    steps_per_epoch=total_train // batch_size,
    epochs=epochs,
    validation_data=val_data_gen,
    validation_steps=total_val // batch_size
)


# Visualize training results
# ----------------------------
# Now visualize the results after training the network.

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']


epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()


# Save Model
# ----------------------------

model.save('cats-and-dogs.h5')
```

predict.py

```python
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import tensorflow as tf
import numpy as np
from tensorflow import keras
from tensorflow.keras.preprocessing import image

model = keras.models.load_model ('cats-and-dogs.h5')

IMG_HEIGHT = 150
IMG_WIDTH = 150




img = image.load_img('cat3.jpg', target_size = (IMG_WIDTH, IMG_HEIGHT))
img1 = image.img_to_array(img)
img2 = np.expand_dims(img1, axis = 0)

pv = model.predict(img2)

print ( 'prediction on cat3.jpg', pv )





img = image.load_img('dog2.jpg', target_size = (IMG_WIDTH, IMG_HEIGHT))
img1 = image.img_to_array(img)
img2 = np.expand_dims(img1, axis = 0)

pv = model.predict(img2)

print ( 'prediction on dog2.jpg', pv )
```

What is “productivity” in the economy.

Don’t fall for silly arguments that we are all going to be out of jobs.