TensorFlow 2.0 Cheatsheet

TENSORFLOW INTRODUCTION

TensorFlow is a high-performance computing library developed by Google Brain team for building machine learning and deep learning models.

Installing TensorFlow – *pip install tensorflow*

Importing TensorFlow – *import tensorflow as tf*

Tensors– n-dimensional arrays of base datatypes.

Shape – *tensor_var.shape*

Represents the dimension of data.

Common Data Types – float32, int32, string etc.

Axis – particular dimension of a tensor

Rank/Degree – *tf.rank(tensor_var)*

Number of tensor dimension. A scalar: rank 0, vector: rank 1, 2D matrix: rank 2.

Creating Tensors –
```
n = tf.Variable(324, tf.int16)
s= tf.Variable(" hi! ", tf.string)
```

Slicing– *tf.slice(t1, begin=[1], size=[3]))*

Types of Tensors – Variable, Constant, Placeholder, SparseTensor

TENSORFLOW CORE LEARNING ALGORITHMS

Linear Regression

If data points are related linearly, we can generate a line of best fit for these points and use it to predict future values. It is used for predicting numeric values.

Line of best fit – y=mx+b

Refers to a line through a scatter plot of data that best expresses the relationship between those points.

Setup and Imports – *pip install -q sklearn*

Training vs Testing Data

Training data is fed to the model so it can develop and learn. **Testing data** is used to evaluate the model and determine its performance

Feature Columns – Categorical or Numeric

Convert categorical data into numeric data by encoding category with an integer.

tf.feature_column.categorical_column_with_vocabulary_list(key, vocabulary_list, dtype=None, default_value=-1, num_oov_buckets=0)

Training Process

data streamed into small batches, fed to model multiple times according to epochs

Epoch– One stream of entire dataset.

Creating Model–linear estimator is used to utilize linear regression algorithm

linear_est = tf.estimator.LinearClassifier(feature_columns=feature_columns)

Training Model – *linear_est.train(train_input_fn)*

Classification

Classification is used to separate data points into classes of different labels.

Setup and Imports

from __future__ import division, print_function, unicode_literals

Feature Columns
```
for key in train.keys():
    my_feature_columns.append(tf.feature_column.numeric_column(key=key))
```

Building Model – models can be -> DNNClassifier, LinearClassifier

DNN is best choice if there's no linear correspondence in data. The no. of neurons is an arbitrary number and many experiments are done to determine best choice.

Training Model

classifier.train(input_fn=lambda: input_fn(train, train_y, training=True),steps=5000)

Predictions – *classifier.predict(input_fn=lambda: input_fn(predict))*
```
def input_fn(features, batch_size=256):
    return tf.data.Dataset.from_tensor_slices(dict(features)).batch(batch_size)
```

Clustering

Clustering is a ML technique that involves grouping of data points. Data points in the same group should have similar properties and/or features.

KMeans Algorithm

Used to find groups in the data, number of groups represented by variable K. It works iteratively to assign each data point to one of K groups based on the features. It is used when you have unlabeled data. As a result we get centroids of the K clusters, which are used to label new data

Hidden Markov Models

It's a finite set of states, each of which is associated with a probability distribution to predict future states.

States – These states are "hidden" within the model.

Observations – Each state has an observation associated with it based on a probability distribution.

Transition–State has probability of transitioning to a different state

Creating Model–

tfd.HiddenMarkovModel(initial_distribution=initial_distribution, transition_distribution=transition_distribution,observation_distribution= observation_distribution, num_steps=7)

NEURAL NETWORKS INTRODUCTION

A deep neural network is a layered representation of data. It represents data in different ways and dimensions by applying specific operations to transform data at each layer.

Keras

Sub module of TensorFlow, its a high-level NN API, written in Python

When to use Keras

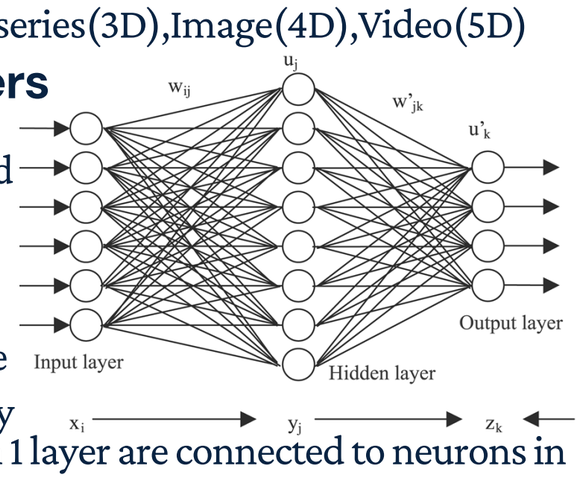
If needed a deep learning library that is easy and fast prototyping, supports CNN,RNN and runs on CPU and GPU

How Neural Networks work

Every layer is made up of neurons and the neurons of each layer are connected to the next. These connections are defined by weight and bias Data enters the input layer and is transformed through layers.Calculate weighted sum at each neuron of the network. Add bias to this sum $Y=(\sum_{i=0}^{n} w_i x_i)+b$ where **w** is weight, **x** is value of the connected neuron, **b** is bias, **n** is number of connections, **y** is output

Data in a NN– Vector(2D),Timeseries(3D),Image(4D),Video(5D)

Layers



Input layer

First layer where initial data is passed

Output layer

Last layer that retrieves results

Hidden layer(s)

At least 1 hidden layer but can include many layers acc. to model complexity

Connected Layers–Neurons in 1 layer are connected to neurons in the next layer in a variety of different ways.

Dense Layer

Each neuron in a layer is connected to every neuron in the next layer.

Neurons–Each layer is made up of neurons,responsible for 1 numeric

Weight

Each pair of connected nodes will have 1 weight to denote the strength.

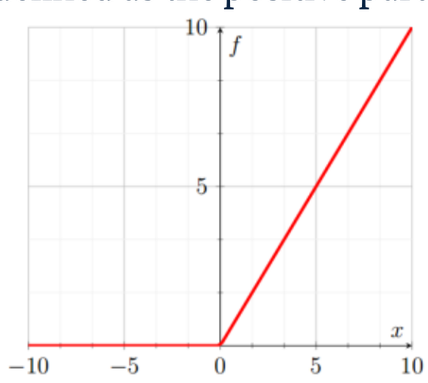
Bias– constant value associated with each layer, helps to shift by const

Activation Function

Function that is applied to the weighted sum of a neuron to add dimensionality to the network.Starts with predefined activation func.

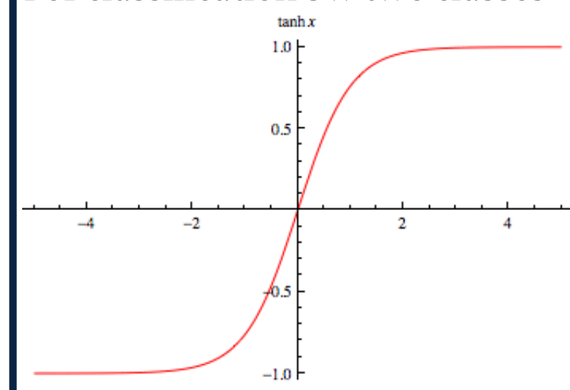
Relu (Rectified Linear Unit)

defined as the positive part of its arg.



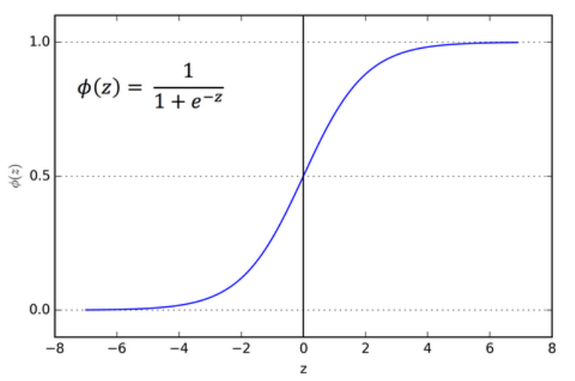
Tanh

For classification bw two classes



Sigmoid

Input to the function is transformed into a value between 0.0 and 1.0



Backpropagation

It changes the weights and biases of the network

Loss/Cost Function – determines the performance of the network
Pass it the output and the expected output, and it returns loss of the network

Loss function types– Mean Squared Error,Mean Absolute Error, Hinge Loss

Gradient Descent– optimization algorithm to find the weights and biases backpropagation calculates the gradient that is used in the gradient descent step.

Optimizer– The function that implements the backpropagation algorithm

Types of Optimizer– Gradient Descent, Stochastic Gradient Descent, Mini-batch Gradient Descent, Momentum, Nesterov Accelerated Gradient

Building sequential model – feed-forward neural network with 3 layers

```
model = keras.Sequential([keras.layers.Flatten(input_shape=(28, 28)),
keras.layers.Dense(128, activation='relu'),keras.layers.Dense(10, activation='softmax')])
```

Flatten– reshapes the array into a vector so each pixel will be with 1 neuron.

Softmax– Calculates a probability distribution for each class

Compile Model–
```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Training Model–
```
model.fit(train_images, train_labels, epochs=10)
```

DEEP COMPUTER VISION

Convolutional neural networks classify and detect images or objects from image.

Image Data– made up of 3 dimensions ie. image height, width,color channels

Convolutional Neural Network

Each CNN is made up of convolutional layers which to find patterns from within images that can be used to classify the image. **Dense layers detect patterns globally while convolutional layers detect patterns locally.**

Feature Maps– 3D tensor with 2 axes - width and height and one depth axis used as input and returns the presence of specific filters called response maps.

Layer Parameters– 2 key parameters ie. Filters and Sample size

Filters– number of patterns each layer is looking for

Padding– addition of the appropriate number of rows and/or columns to input data such that each pixel can be centered by the filter

Pooling– downsample feature maps and reduce their dimensions

CNN Architecture

Basic architecture for a CNN is a stack of Conv2D and MaxPooling2D layers followed by a few densely connected layers to extract the features from the image which are flattened and fed to densly connected layers that determine the image class

```
model = models.Sequential(),model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3))),model.add(layers.MaxPooling2D((2,2)))
,model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten()),model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

Data Augmentation– To avoid overfitting and create a larger dataset from a smaller one by performing random transformations

Freezing the Base– disabling the training property of a layer

Data Preprocessing– convert all images to same size

Training– train and compile the model using the TensorFlow hyperparameters.

```
model.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentropy
(from_logits=True),metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=4,validation_data=(img, test_labels))
```

NATURAL LANGUAGE PROCESSING

NLP deals with the communication between natural and computer languages.

Sequence Data– text, weather, videos, or anything where the notion of time is relevant needs to be processed in a special way.

Encoding Text– encode textual data to numeric values for feeding to model

Bag of Words– Each word in a sentence is encoded with an integer, without keeping the order of the words but keeping track of frequency.

Integer Encoding

represent each word/character as a unique integer and keep the order of words.

Word Embeddings– add embedding layer to the beginning of the model
It encodes the frequency, order, and meaning of those words in the sentence.

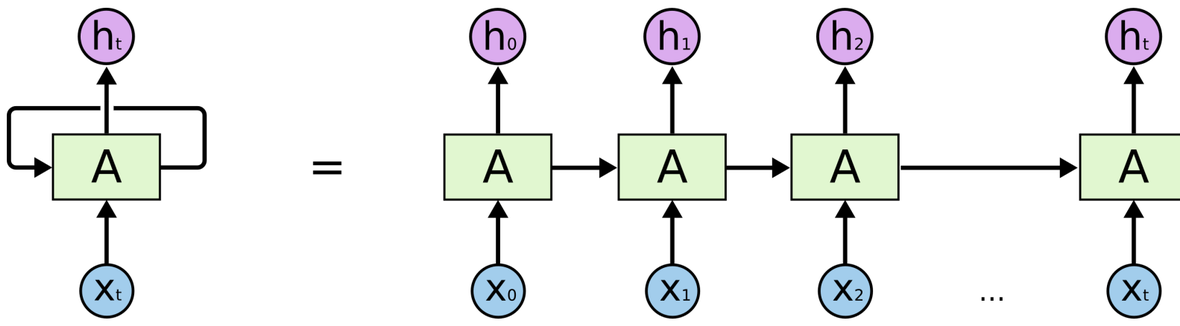
Recurrent Neural Networks

RNN will process one word at a time while maintaining an internal memory of what it's already seen. It helps to treat words differently based on their order.

Simple RNN layer – Effective at processing shorter sequences of text for simple problems

LSTM– Long Short-Term Memory layer adds a way to access inputs from any time step in the past. This allows it to discover more useful relationships between inputs

Sentiment Analysis–
The process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral.



REINFORCEMENT LEARNING

Rather than feeding the model millions of examples, let the model come up with its own examples by exploring an environment.

Environment

There's a notion of the environment. So the agent can explore.

Agent– The entity that is exploring the environment. The agent will interact and take different actions within the environment.

State– The state simply tells about the status of the agent.

Action– Interaction between the agent and environment would be considered an action.

Reward
Action by agent results in a reward of some magnitude (positive or negative). The goal of an agent is to maximize its reward in an environment.

Q–Learning – Involves learning a matrix of action-reward values. The Q-learning algorithm defines the way we update the values in the matrix and decide what action to take at each state. After successful training of Q-Table, we can determine the action an agent should take in any state by looking at that states row in the matrix and taking the maximum value column as the action.

Learning the Q–Table

Q-Table starts off with all 0 values. Agent decides action by:

1. Randomly picking a valid action
2. Using the current Q-Table to find the best action.

After each new action, agent records new state and the reward received from taking that action. These values will be used to update the Q-Table. The agent will stop taking new actions once a certain time limit is reached or it has achieved the goal or comes at end of the environment.

Updating Q–Values–

Q[state,action]=Q[state,action]+α∗(reward+γ∗max(Q[newState, :])−Q[state,action]) **α** is **Learning Rate**, **γ** is the **Discount Factor**

Learning Rate α

It's a numeric constant that defines how much change is permitted on each QTable update. A high learning rate means that each update will introduce a large change to the current state-action value.

Discount Factor γ

Used to balance amount of focus to put on the current and future reward. A high discount factor means that future rewards will be considered more heavily.

Open AI Gym

Developed so programmers could practice machine learning using unique environments.

Constants

Used to update our Q-Table and tell our agent when to stop training.

https://www.tensorflow.org/api_docs
https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56.