Whizlabs - ML Specialty Exam Course - Algorithms - part 3

Image Analysis Algorithm

AWS Machine Learning - Image Analysis Algorithms

Image Analysis Algorithm - Defined

- Supervised learning algorithms
- Take images as input and either labels the images or identifies objects in the image
- Two image analysis algorithms in the SageMaker built-in algorithms

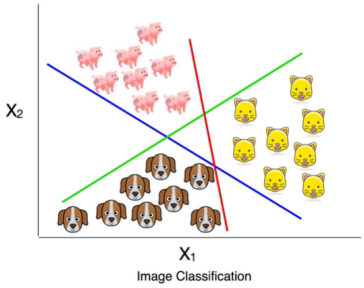


Image Classification


Object Detection

Example Image Analysis Algorithm Use Cases

- Facial recognition
- Airport baggage scanning
- Analyze social media images for missing persons
- Real-time vehicle damage assessment
- Medical image analysis
- Building entrance security
- Product line analysis

SageMaker Image Analysis Algorithms - Image Classification

- Image Classification
  - Supervised learning algorithm that supports multi-label classification
  - Takes an image as input and outputs one or more labels assigned to that image
  - Uses a convolutional neural network (ResNet) that can be trained from scratch or trained using transfer learning when a large number of training images are not available
  - Can also seed the training of a new model with the artifacts from a model that you trained previously, called incremental training
  - Recommended input format is RecordIO, but can also use raw images in .jpg or .png format.
  - Example use case: classify image of person as on building entry security list

SageMaker Image Analysis Algorithms - Object Detection

- Object Detection
  - Supervised learning algorithm
  - Detects and classifies objects in images using a deep neural network
  - Takes images as input and identifies all instances of objects within the image
  - Object is categorized into one of the classes in a collection you specify, with a confidence score assigned to the class
  - Location and scale of the object in the image are noted by bounding box
  - Can be trained from scratch, or trained with models that have been pre-trained on the ImageNet dataset
  - Use input format of RecordIO, but can also use raw images in .jpg or .png format.
  - Example use case: detect objects in baggage at airport scan

SageMaker Image Analysis Algorithms - Important Hyperparameters

- Image Classification
  - num_classes: number of output classes
  - num_training_samples: number of training examples in the input dataset
  - early_stopping: define a threshold at which to stop training
- Object Detection
  - num_classes: number of output classes
  - num_training_samples: number of training examples in the input dataset
  - use_pretrained_model: use a pre-trained model for training

AWS Machine Learning - Text Analysis Algorithms

Text Analysis Algorithm - Defined

- Both supervised and unsupervised learning algorithms
- Take text or documents as input and either categorize, sequence, or classify the text or documents
- Used as preprocessing for many downstream natural language processing (NLP) tasks, such as sentiment analysis, named entity recognition, machine translation
- Text classification used for applications that perform web searches, ranking, and document classification

Example Text Analysis Algorithm Use Cases

- Sentiment analysis for social media streams
- Categorize documents by topic for law firms
- Language translation
- Speech-to-text
- Summarizing longer documents
- Conversational user interfaces
- Text generation
- Word pronunciation app

SageMaker Text Analysis Algorithms - Blazing Text

- Blazing Text
  - Implements the Word2vec and text classification algorithms
  - Can use pre-trained vector representations that improve the generalizability of other models that are later trained on a more limited amount of data
  - Can easily scale for large text datasets
  - Can train a model on more than a billion words very quickly, in minutes, using a large multi-core CPU or a GPU
  - Words that are semantically similar correspond to vectors that are close together, resulting that word embeddings capture the semantic relationships between words.
  - Example use case: market research using sentiment analysis

SageMaker Text Analysis Algorithms - Latent Dirichlet Allocation (LDA)

- Latent Dirichlet Allocation (LDA)
  - Unsupervised learning algorithm that organizes a set of text observations into distinct categories
  - Frequently used to discover a number of topics shared across documents within a collection of texts, or a corpus
  - In an LDA algorithm based model, each observation is a document and each feature is a count of a word in the documents
  - Topics are not specified in advance
  - Each document is described as a mixture of topics
  - Example use case: find common topics in call center transcripts

SageMaker Text Analysis Algorithms - Neural Topic Model (NTM)

- Neural Topic Model (NTM)
  - Unsupervised learning algorithm that organizes a corpus of documents into topics containing word groupings, based on the statistical distribution of the word groupings
  - Frequently used to classify or summarize documents based on topics detected
  - Also used to retrieve information or recommend content based on topic similarities
  - Topics are inferred from observed word distributions in the corpus
  - Used to visualize the contents of a large set of documents in terms of the learned topics
  - Similar to LDA, but will produce different outcomes
  - Example use case: find the topics of newsgroup message posts

SageMaker Text Analysis Algorithms - Object2Vec

- Object2Vec
  - General purpose neural embedding algorithm that finds related clusters of words (words that are semantically similar)
  - Embeddings can be used to find nearest neighbors of objects, and can also visualize clusters of related objects
  - Besides word embeddings, Object2Vec can also learn the embeddings of other objects such as sentences, customers, products, etc.
  - Frequently used for information retrieval, product search, item matching, customer profiling, etc. based on related topics
  - Supports embeddings of paired tokens, paired sequences, and paired token to sequence
  - Example use case: recommendation engine based on collaborative filtering

SageMaker Text Analysis Algorithms - Sequence-to-Sequence (seq2seq)

- Sequence-to-Sequence (seq2seq)
  - Supervised learning algorithm with input of a sequence of tokens (audio, text, radar data) and output of another sequence of tokens
  - Can be used for translation from one language to another, text summarization, speech-to-text
  - Uses Recurrent Neural Networks (RNNs) and Convolutional Neural Network (CNN) models
  - Uses state-of-the-art encoder-decoder architecture
  - Uses input of sequence data in recordio-protobuf format and JSON vocabulary mapping files
  - Example use case: word pronunciation dictionary, a sequence of text as input and a sequence of audio as output

SageMaker Text Analysis Algorithms - Important Hyperparameters

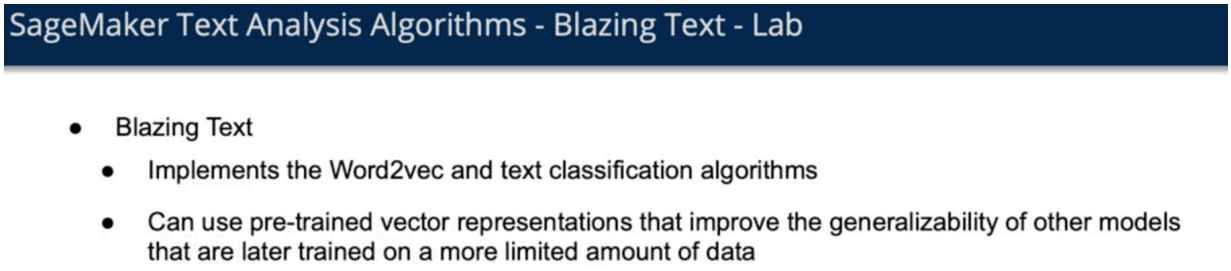
- Blazing Text
  - mode: architecture used for training
- Latent Dirichlet Allocation (LDA)
  - num_topics: number of topics to find in the data
  - feature_dim: size of the vocabulary of the input document corpus
  - min_batch_size: total number of documents in the input document corpus
- Neural Topic Model (NTM)
  - feature_dim: vocabulary size of the dataset
  - num_topics: number of required topics

SageMaker Text Analysis Algorithms - Important Hyperparameters

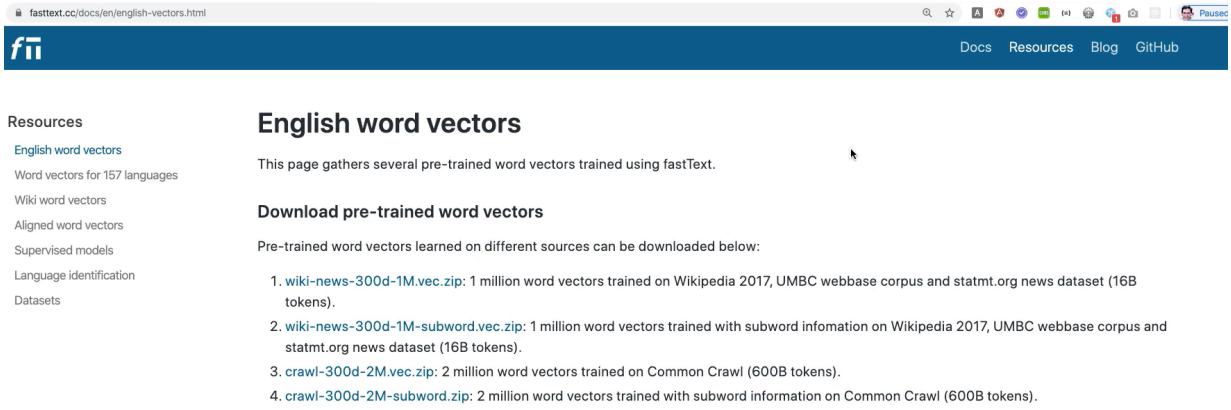
- Object2Vec
  - enc0_max_seq_len: maximum sequence length for the enc0 encoder
  - enc0_vocab_size: vocabulary size of enc0 tokens
- Sequence-to-Sequence (seq2seq)
  - Has no required hyperparameters

Text Analysis - Lab

Using BlazingText with an accelerated GPU version and a pre-trained model from FastText



FastText for pertained models:



Get the dataset from fasttext , tar it and save it to S3

In [1]:
```
import sagemaker
from sagemaker import get_execution_role
import boto3
import json

my_session = sagemaker.Session()

role = get_execution_role()

s3 = boto3.resource('s3')
bucket_name = 'machine-learning-exam'       # Change to your bucket
prefix = 'language_identification/fasttext' # and prefix
```

In [2]:
```
my_region_name = boto3.Session().region_name
```

In [3]:
```
blazing_text_container = sagemaker.amazon.amazon_estimator.get_image_uri(my_region_name, "blazingtext", "latest")
print('Using SageMaker BlazingText container: {} ({})'.format(blazing_text_container, my_region_name))
```

```
Using SageMaker BlazingText container: 811284229777.dkr.ecr.us-east-1.amazonaws.com/blazingtext:latest (us-east-1)
```

In [4]:
```
!wget -O model.bin https://dl.fbaipublicfiles.com/fasttext/supervised-models/lid.176.bin
```

```
--2020-02-26 00:59:24--  https://dl.fbaipublicfiles.com/fasttext/supervised-models/lid.176.bin
Resolving dl.fbaipublicfiles.com (dl.fbaipublicfiles.com)... 104.20.6.166, 104.20.22.166, 2606:4700:10::6814:16a6,
...
Connecting to dl.fbaipublicfiles.com (dl.fbaipublicfiles.com)|104.20.6.166|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 131266198 (125M) [application/octet-stream]
Saving to: 'model.bin'

model.bin           100%[===================>] 125.18M  26.0MB/s    in 5.5s

2020-02-26 00:59:30 (22.6 MB/s) - 'model.bin' saved [131266198/131266198]
```

In [5]:
```
!tar -czvf langid.tar.gz model.bin
blazing_text_model_location = my_session.upload_data("langid.tar.gz", bucket=bucket_name, key_prefix=prefix)
!rm langid.tar.gz model.bin
```

```
model.bin
```

Now set our inference endpoint

In [6]:
```
language_identifier = sagemaker.Model(model_data=blazing_text_model_location,
                                      image=blazing_text_container,
                                      role=role,
                                      sagemaker_session=my_session)

language_identifier.deploy(initial_instance_count = 1,
                           instance_type = 'ml.m4.xlarge')

language_identifier_predictor = sagemaker.RealTimePredictor(endpoint=language_identifier.endpoint_name,
                                    sagemaker_session=my_session,
                                    serializer=json.dumps,
                                    deserializer=sagemaker.predictor.json_deserializer)
```

```
-------------------!
```

We can now detect the language

In [7]:
```
some_language_examples = ["À quoi sert l'intelligence artificielle",
             "Was ist der Zweck der künstlichen Intelligenz?",
             "Wat is die doel van kunsmatige intelligensie",
             "ما هو الغرض من الذكاء الاصطناعي",
             "Süni intellektin məqsədi nədir",
             "Hvad er formålet med kunstig intelligens"]
prediction_input = {"instances" : some_language_examples}
```

In [9]:
```
language_predictions = language_identifier_predictor.predict(prediction_input)
print(language_predictions)
```

```
[{'prob': [0.8571586608886719], 'label': ['__label__fr']}, {'prob': [0.9994584321975708], 'label': ['__label__de']},
{'prob': [0.465190052986145], 'label': ['__label__af']}, {'prob': [0.9983780980110168], 'label': ['__label__ar']},
{'prob': [0.9949907064437866], 'label': ['__label__az']}, {'prob': [0.864094614982605], 'label': ['__label__da']}]
```

In [10]:
```
# Remove the '__label__' before each language identifier in the prediction output
# and change the label and prob to more readable values
for output in language_predictions:
    output['label'] = output['label'][0][9:].upper() # remove __label__ preceding the language identifier
    output['language'] = output.pop('label')         # make the labels
    output['probability'] = output.pop('prob')       # readable

print(language_predictions)
```

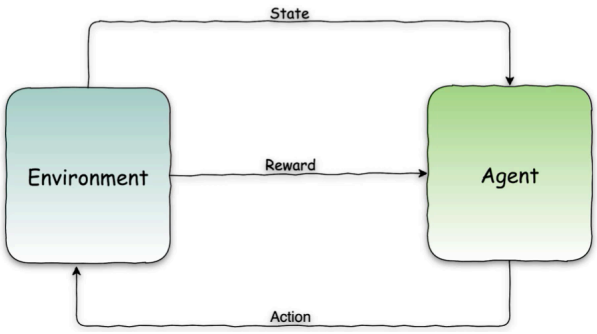
```
[{'language': 'FR', 'probability': [0.8571586608886719]}, {'language': 'DE', 'probability': [0.9994584321975708]},
{'language': 'AF', 'probability': [0.465190052986145]}, {'language': 'AR', 'probability': [0.9983780980110168]}, {'la
nguage': 'AZ', 'probability': [0.9949907064437866]}, {'language': 'DA', 'probability': [0.864094614982605]}]
```

Reinforcement Learning

AWS Machine Learning - Reinforcement Learning Algorithms

Reinforcement Learning Algorithms - Defined

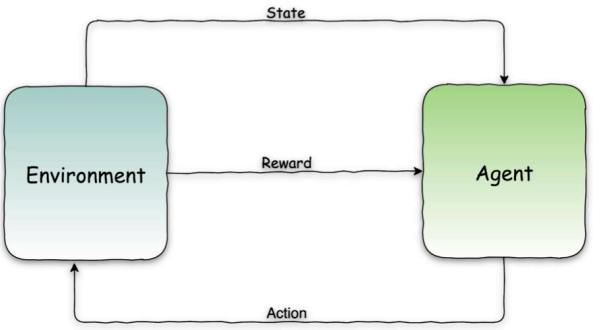
- Machine learning technique which learns a strategy, called a policy, that optimizes an objective for an agent acting in an environment
- Agent performs an action, checks the state of the environment, and gets rewarded based on the resulting state of the environment
- Attempts to achieve the goal of maximizing the long-term reward that the agent receives resulting from its actions
- Good for solving problems where you want your model to make independent decisions



Agent takes action within environment, check the state and get potentially reward, then try similar actions if reward

Reinforcement Learning Algorithms - Markov Decision Process (MDP)

- Based on Markov Decision Processes (MDPs) models
- Works through a sequence of time steps with each step containing the following
  - Environment: operating space of the RL model
  - State: complete information describing the environment and relevant past steps
  - Action: agent's activity
  - Reward: number that reflects the state resulting from the last action
  - Observation: data about the state of the environment available at each step



Example Reinforcement Learning Algorithm Use Cases

- Robotics
- Traffic light control
- Predictive auto scaling
- Tuning parameters of a web system
- Optimizing chemical reactions
- Personalized recommendations
- Gaming
- Deep learning: "see an environment" and learn how to interact with it

SageMaker Reinforcement Learning Algorithms - Deep Learning Framework

- Reinforcement Learning in TensorFlow and Apache MXNet
  - Uses a Reinforcement Learning toolkit
    - Manages the agents interaction with the environment
    - SageMaker supports Intel Coach and Ray RLlib toolkits
  - Uses an environment
    - Custom environment
    - Open source environments: EnergyPlus, RoboSchool
    - Commercial environments: MATLAB, Simulink

SageMaker Reinforcement Learning Algorithms - Important Hyperparameters

- Reinforcement Learning
  - learning_rate: how fast the model learns
  - discount_factor: take action with short-term or long-term rewards
  - entropy: degree of uncertainty; exploit what's already known versus thorough exploration of the environment

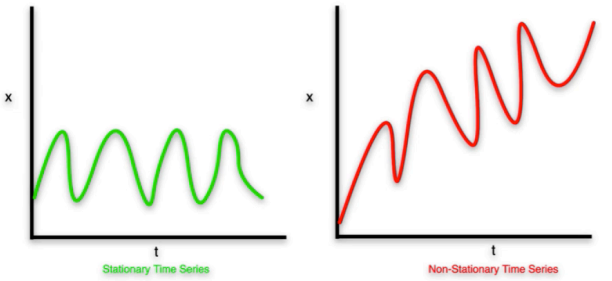
discount_factor is a value between 0 and 1
- 0: agent cares for his first reward only
- 1: agent cares for all future rewards

Forecasting Algorithms

AWS Machine Learning - Forecasting Algorithms

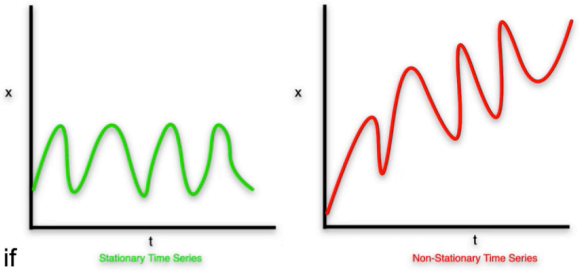
Forecasting Algorithms - Defined

- DeepAR algorithm
  - Supervised learning algorithm that forecasts one-dimensional (scalar) time series using recurrent neural networks (RNN)
  - Trains a single model jointly over all of the similar time series in your dataset
  - Use your trained model to create forecasts for new time series that are similar to the ones on which it was trained
  - Training input is one or more target time series that have been generated by the same process or similar processes
  - Uses an approximation of the process(es) and predicts how the target time series evolves



Forecasting Algorithms - DeepAR Algorithm

- How it works
  - Uses a training dataset and an optional test dataset
  - Use a trained model to predict forecasts for the future of the time series in the training set, or for other time series
  - Automatically derives time series based on the frequency of the target series
    - For example: a day series generates day-of-week, day-of-month, day-of-year
  - Can predict "what if?" scenarios, such as, what if I change/broaden the distribution of my product?
  - Builds a single model for all time-series and tries to identify similarities across them



Example Forecasting Algorithm Use Cases

- Forecasting global food demand
- Predicting global temperature
- Predicting fast-food queues
- Predicting rainfall
- Predicting stock prices
- Options pricing
- Disease progression
- Supply chain management

SageMaker Forecasting Algorithms - Important Hyperparameters

- DeepAR algorithm
  - contexct_length: the number of time-points that the model gets to see before making the prediction
  - epochs: the maximum number of passes over the training data
  - prediction_length: the number of time-steps that the model is trained to predict, also called the forecast horizon
  - time_freq: the granularity of the time series in the dataset