AWS Maching Learning Notes

CALVIN WRIGHT

2022-0227

Data Storage & Pre-processing

Data Stores

	Redshift	RDS / Aurora	Dynamo DB	S3	ElasticSearch	ElastiCache
Definition	- Data Warehousing for **Online Analytical Processing** (OLAP)- SQL - Columnar Based - Redshift Spectrum can query data directly from S3	- Relational data store for **Online Transaction Processing** (OTP)- SQL - Must provision servers in advance	- No-SQL data store - Serverless (provision read/write capacity)	- Object Storage - Serveless / infinite - SageMaker	- Indexing of data	- Caching Mechanism - Not used for ML
Use Cases	-Huge scale analytics - **ML based predictions** - Big data processing	- Accounting - Product information - Basic queries - E-commerce	- Pair wise data - User interactions	- **SageMaker training data** - Image data	- Clickstream analytics	- Retrieve hot data

- **FSx for Lustre** - When your training data is in S3 and you plan to run training jobs several times using different algorithms and parameters. After first training job the data is saved in SageMaker and speeds up further retrieval.
- **Lake Formation** – Data Lake solution to hold structured and unstructured data in one place

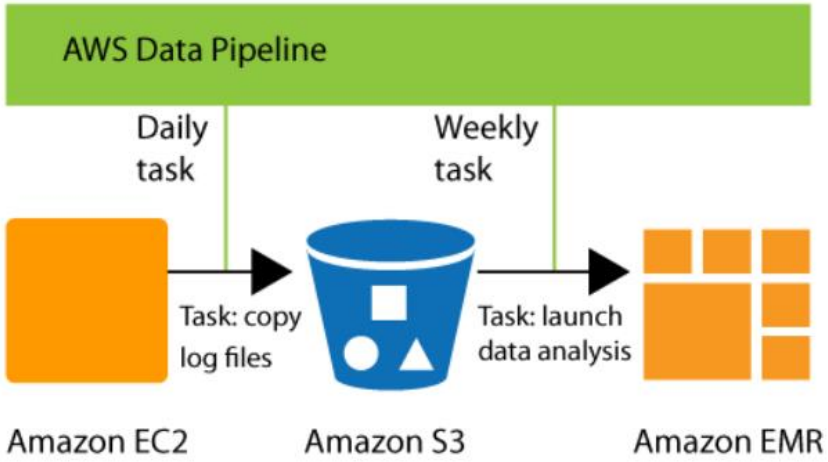
Kinesis Family

- Amazon Kinesis makes it easy to collect, process, and analyze real-time, **streaming data**
- The Kinesis Data Streams APIs help you manage many aspects of Kinesis Data Streams, including creating streams, resharding, and putting and getting records.
- Kinesis Consumer Library (**KCL**) helps you consume and process data from a Kinesis data stream by taking care of many of the complex tasks associated with distributed computing - focus your efforts on writing your custom record-processing logic.
- The Kinesis Producer Library (**KPL**) simplifies producer application development, allowing developers to achieve high write throughput to a Kinesis data stream.

	Data Streams	Data Firehose	Data Analytics	Video Streams
Definition	Scalable and durable **real-time** data streaming service.	Capture, **transform**, and deliver streaming data into data lakes, data stores, and analytics services. **Near real time** **Fully managed!**	Transform and analyze streaming data in real time with Apache Flink.	Stream video from connected devices to AWS for analytics, machine learning, playback, and other processing.
Data sources	Any data source (servers, mobile devices, IoT devices, etc) that can call the Kinesis API to send data.	Any data source (servers, mobile devices, IoT devices, etc) that can call the Kinesis API to send data.	Amazon MSK, Amazon **Kinesis Data Streams/firehose**, servers, mobile devices, IoT devices	Any streaming device that supports Kinesis Video Streams SDK.
Data consumers	Kinesis Data Analytics, Amazon EMR, Amazon EC2, AWS Lambda	Kinesis Data Analytics, Kinesis Data Streams Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, **HTTP endpoints partners**(Datadog, New Relic, MongoDB…) and Splunk	Analysis results can be sent to another Kinesis stream, a Kinesis Data Firehose delivery stream, or a **Lambda function**	Amazon Rekognition, Amazon SageMaker, MxNet, TensorFlow, HLS-based media playback, custom media processing application
Use cases	– Log and event data collection – Real-time analytics – Mobile data capture – Gaming data feed	– IoT Analytics – Clickstream Analytics – Log Analytics – Security monitoring	– Streaming ETL – Real-time analytics – Stateful event processing	– Smart technologies – Video-related AI/ML – CCTV

Data Pipeline

- **ETL** service – **orchestrates** the movement of data from 1 location to another
- Destinations: S3, RDS, DyanamoDB, Redshift and EMR
- Manages task dependencies and retries on failures
- Can be used to move data from **on-prem**
- More control than AWS Glue (uses Apache Spark / Scala or python code and manages resources). Pipeline allows you to control EC2 or EMR instances.



Use Cases

- Copy RDS or DynamoDB tables to S3, transform data structure, run analytics using SQL queries and load it to Redshift.
- Analyze unstructured data like clickstream logs using **Hive or Pig on EMR**, combine it with structured data from RDS and upload it to Redshift for easy querying.
- Load log files such as from the AWS billing logs, or AWS CloudTrail, Amazon CloudFront, and Amazon CloudWatch logs, from Amazon S3 to Redshift.

AWS Glue

AWS Glue is a **serverless** data integration service that makes it easy to **discover, prepare, and combine data** for analytics, machine learning, and application development.

- Visually create, run, and monitor ETL workflows with a few clicks in AWS **Glue Studio**.
- Use **AWS Glue DataBrew** to visually enrich, **clean**, and normalize data without writing code.
- With **AWS Glue Elastic Views,** developers can use (**SQL**) to combine and replicate data across multiple data stores. Live updates.
- **Glue Crawlers** search data sources (S3,RDS,Redshift) and extract schemas for your data and store them in **Glue Data Catalog**
- **Glue ETL –** Automatically use dropfields, filter, join, map, convert (CSV->Parquet), **FindMatches ML** (Find duplicates), Apache Spark transformations (example:K-means)
- Integrates with **Athena** to know your schemas.

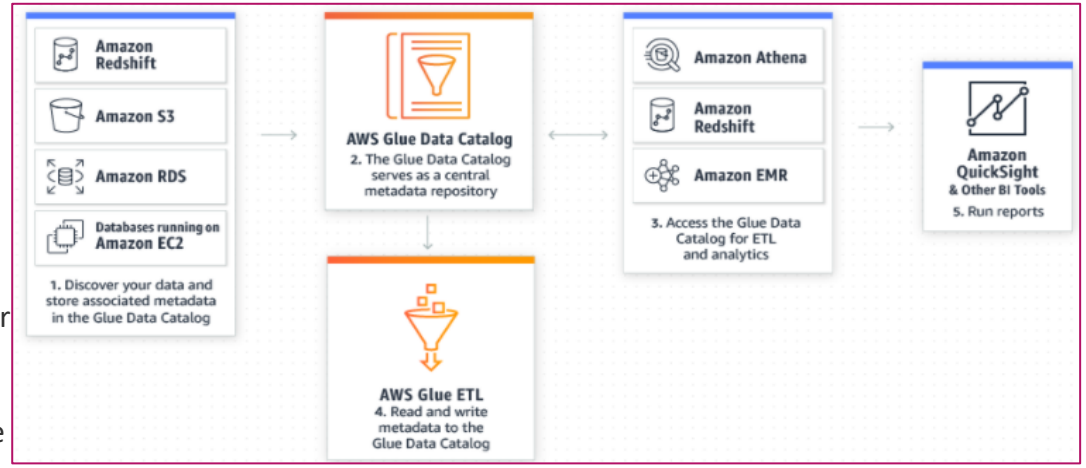
Use Caes

1. Build event-driven ETL (extract, transform, and load) pipelines

   AWS Glue can run your ETL jobs as new data arrives. For example, you can use an AWS Lambda function to trigger your ETL jobs to run as soon as new data becomes available in Amazon S3. You can also register this new dataset in the AWS Glue Data Catalog as part of your ETL jobs.

2. Create a unified catalog to find data across multiple data stores

   AWS Glue Studio makes it easy to visually create, run, and monitor AWS Glue ETL jobs



Additional Services

AWS Batch

- Run **any jobs** as **Docker images**
- Dynamic provisioning of EC2 (optimized on volume and requirements)
- Fully **serverless**
- Schedule jobs using Cloudwatch events
- Orchestrate jobs using Step Functions
- Example: Clean-up process of your S3 buckets which are triggered each week.
- Batch orientated non-ETL work

Data Migration Service

- Quickly **migrate DB to AWS**.
- Source DB remains available at all times
- Homogenous – Oracle > Oracle
- Heterogenous – Oracle > Aurora
- Continuous Data replication using change data capture.
- Must create EC2 to replicate
- **No data transformation**. Use Glue ETL to do this after a migration if necessary.

AWS Step Functions

- Use to design workflow
- Advanced error handling, Prepare Data for waiting and retry logic.
- Audit log available
- Max exec time = 1 year
- Human or automated steps available

Example

Prepare Data for Machine Learning (ML)

To enable machine learning, source data must be collected, processed, and normalized so that ML modelling systems like Amazon SageMaker can train on that data. Step Functions makes it easier to sequence the steps it takes to automate your ML pipeline.

Amazon Elastic Map Reduce (EMR)

Managed Hadoop framework on a cluster of EC2 instances

- EMR Notebooks provide a managed environment, based on Jupyter Notebooks, to help users prepare and visualize data
- Hbase, Presto, Flink, Hive, Pig…
- Quickly provision as much capacity as you need, and automatically or manually add and remove capacity.
- **Transient** – Shuts down once all tasks are complete
- **Long Running** – Want to experiment with data and shut down manually when done.
- Store input/output data on S3
- Use Pipelines to start/stop EMR

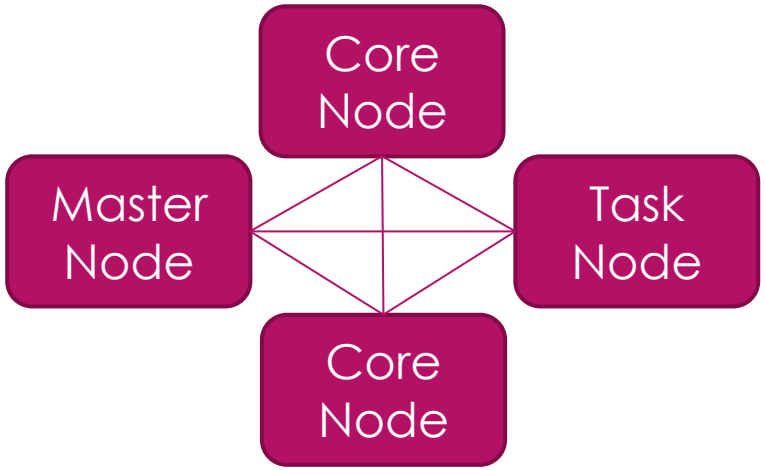
EMR Storage

- Default storage is HDFS – ephemeral storage so good for performance but in mem only
- EMRFS in S3 as if it was HDFS. Still pretty fast

- **Spark** being used instead of MapReduce these days. Interacts with YARN+HDFS
  YARN NodeManager Port=8042
  YARN ResourceManager Port=8088
- Spark **MLLib** allows you to use ML and helps with **parallelization** which they can't on their own
- **Spark Streaming**: integrates with Kinesis Data Stream. Use KCL
- **Zeppelin** notebooks with Spark.
  Port=8890
- EMR Notebook – Similar to Zeppelin with more AWS integration
  - Provision cluster from a notebook
  - Hosted in a VPC, backed up to S3
- Security: IAM policies, Kerberos, SSH, IAM roles

EMR Cluster

- **Master Node**: Manages the cluster. Cluster must contain 1. Distributes tasks and manages cluster
- **Core Node**: Hosts HDFS data, run tasks, can be scales up or down.
- **Task Node**: Run tasks, does not host data. Good for **Spot Instances**.



Amazon QuickSight

Fast, **Serverless** and easy business analytics service

- Data Sources: Redshift, Aurora/RDS, Athena, EC2, S3 or on-Prem
- SPICE – Super-fast Parallel, In-memory Calculation Engine. Uses columnar storage, accelerates interactive queries on large datasets. Each user get 10GB Spice

Use Cases

- Perform **ad-hoc exploration** and display results across browsers and mobile
- Stories – guided tours through specific views of an analysis. Evolution of data analysis
- Anomaly detection (RCF), Forecasting and Auto-narratives
- Not for full canned reports or ETL

Security

- MFA and VPC connectivity through security groups, Elastics Network Interface and AWS Direct Connect
- Row level security on data

Graph Types

- AutoGraph – QuickSight chooses best graph from analyzing the data.
- Bar Charts – For comparison and distribution
- Line Graphs – Changes over time (time series)
- Scatter Plots & Heat Maps – For correlation
- Pie Graphs & Tree Maps – Aggregation, % summation
- Pivot Tables – Tabular data & aggregation analytics

ML Knowledge

Feature Engineering

- Curse of Dimensionality – Too many features causing compute resources to be too high. Can be reduced with unsupervised reduction techniques like **PCA**, **t-SNE** and **K-means**.
- Imputing missing data:
  - Replace with mean – fast and easy, not losing any data. But not very accurate, no categorical
  - Dropping – only okay if you need cheap and easy solution, and it's a small % of data
  - **Use ML** – KNN (most similar), Deep Learning (V. Good, complicated), Regression (**MICE**)
  - Get more data!
- Unbalanced Data: (like fraud detection)
  - Duplicate the minority case
  - Remove majority case (only an option if CPU is an issue)
  - **SMOTE** – Artificially generate new samples of minority class using KNN
  - Adjust **threshold** of identifying a positive case
- Outliers (Outside X standard deviations):
  - Throw them away if skewing results – your judgement
  - AWS Random Cut Forest good for detecting them
- Binning: Grouping items to get more samples (grouping age). Unlikely to help predictions, can help visualize. If you want all bins to be equal in size, use Quantile Binning

Feature Engineering

- **Bagging** – Train multiple decision trees on different data, joint predictions. Stops overfitting and train in parallel
- **Boosting** – Observations are weighted. Sequential training and weights updated. Great accuracy (XGBoost)
- **Encoding** – transforming categorical data to numerical:
  - One Hot: Each value gets its own column and then binary values to show vals. Great for Neural Networks
  - Binary – Similar to One Hot but requires less columns. Get binary value of each column to understand value.
  - Label Encoding – Map each category value to a number
  - Ordinal Encoding – Same as label encoding but order preserved for simplicity
- Normalization – Min-Max or MaxAbs Scaling values to between [0;1] – Sensitive to outliers
- Standardization – Similar to above but between [-1;1] and variance of 1 – Bad if data is NOT normal
- Robust Standardization – Uses quartiles to stop outliers from affecting the standardization
- Shuffling – Mix up the data at each training phase to prevent over fitting or not sampling certain populations.

NLP Prepareation

- Tokenisation - split up sentence into vector of words
- Padding – Adding null values so that all sentences have same shape
- Embeddings – Convert words to a numerical feature space where similar words are close together.
- Stemming – Converting different forms/tenses of word to 1 distinct word. (play,playing,playful -> play)
- Stop Words – Words that don't add value like "a", "an", "the"

Evaluation metrics

Accuracy: Percentage of correct inferences.
Recall: How well you get all cases of interest. When False Negative has a large cost (Fraud, tumor detection). % of positive identified.
Precision: The number of correctly classified, divided by the total number of classifications. High precision means that the classifier returned substantially more relevant results than irrelevant ones. Recommendation systems
Specifity: True negative rate - Guess negatives correctly drug testing
F1 Score: When you care about recall and precision
RMSE: This is just a regression evaluation metric. Sum of squared errors – penalize outliers.
ROC Curve: Plot true positive rate (recall) vs false negative rate at various thresholds. More bent toward top left corner the better. Then the AUC is a common metric for comparing classifiers.
Micro X : Overall score of all X scores added together.
Hamming Loss: Fraction of wrong labels compared to the total number of labels.

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \text{ Or } \frac{\text{Total Correct}}{\text{Total}}$$

$$\textbf{Recall} = \frac{TP}{TP+FN} \text{ Or } \frac{\text{Correct Positive}}{\text{Total Positive}}$$

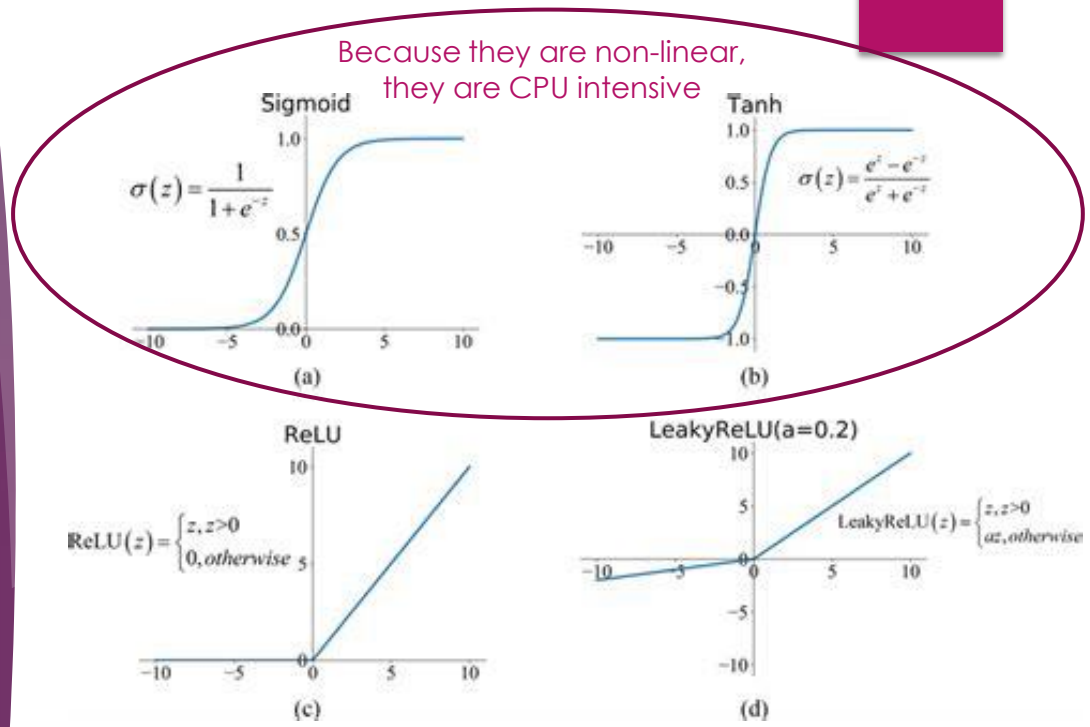
$$\textbf{Precision} = \frac{TP}{TP+FP} \text{ Or } \frac{\text{Correct Positive}}{\text{Guessed Positive}}$$

$$\textbf{F1 Score} = \frac{2*TP}{2*TP+(FP+FN)} \text{ Or } \frac{2*(\text{Precision}*\text{Recall})}{\text{Precision} + \text{Recall}}$$

Activation Functions

- **Sigmoid**: 0 (as more negative a number gets) or 1 (as more positive a number gets). This activation function is more desirable for binary classification problems
- **Tanh**: Like the Sigmoid, but it's centered around 0 instead of 0.5. Tanh often preferable as it converges faster to a minimum
- **ReLU**: Convert inputs into the range of (0, positive input value). Fast and accurate but has issue of 0->neg numbers being linear (Dying ReLU)
- **Leaky ReLu**: Solves dying ReLU issue with slight negative slope. **Parametric RelU** is advanced version that uses back propagation. CPU intensive
- **Softmax** : converts a vector of real numbers into a vector of a probability distribution whose sum is equal to 1. Good for **final layer**
- **Swish**: From Google, good for V. deep networks
- **Maxout**: Outputs max of the inputs

Because they are non-linear, they are CPU intensive

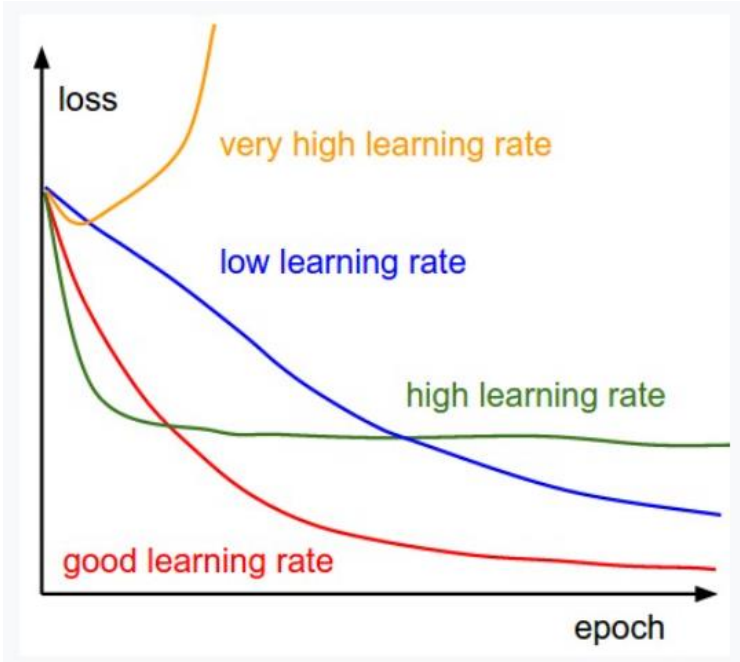


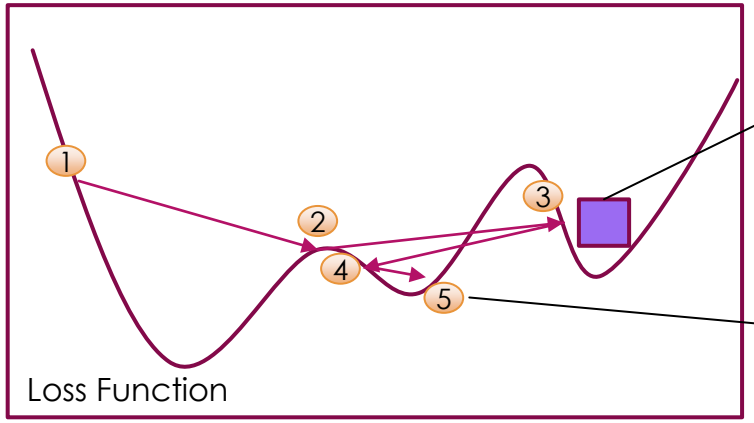
Recommendations

- For multiple classifications final layer = softmax
- RNNs do well with Tanh
- Others: Start with ReLU, if you need better try going to Leaky ReLU, PReLU or Maxout

Tuning Neural Networks

- Small batch sizes don't get stuck in local minima
- Large batch sizes can converge on wrong solution at random
- Large learning rates can overshoot the correct solution
- Small learning rates increase training time





Large batch sizes get stuck. Think of them as being too heavy to escape local minima

High Learning rate overshoots the best solution

Optimizers

Optimizers tie together the loss function and model parameters by updating the model in response to the output of the loss function. In simpler terms, optimizers shape and mold your model into its most accurate possible form by futzing with the weights.

- **Adagrad** - Adaptively sets the learning rate according to a parameter.
- **Adadelta** - Adadelta is a more robust extension of Adagrad that adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients.
- **Adam** - Combines ideas from both RMSProp and Momentum. It computes adaptive learning rates for each parameter.
- **Conjugate Gradients**
- **BFGS**
- **Momentum** - takes into account past gradients to smooth out the update. This results in minimizing oscillations and faster convergence.
- Nesterov **Momentum**
- **Newton's Method**
- **RMSProp** - Another adaptive learning rate optimization algorithm
- **SGD** – A few samples are selected randomly instead of the whole data set for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy or less random manner, but the problem arises when our datasets get really huge.

Sage Maker Models

SageMaker Models

Model	Description	Data	Tuning	Hardware
Linear Learner	**Linear regression** – fit line to data. Can handle **numeric** and **classification**. Simple/Cheap	RecordIO-wrapped protobuf CSV File or pipe mode for both	-Balance_multiclass_weights -Learning rate -Mini_batch_size -L1 (Lasso) & -L2 (ridge reg. / weight decay) regression	Multi CPU Single GPU
XGBoost	Boosted group of decision trees Classification and regression	CSV or libsvm - original RecordIO-protobuf & Parquet - new	-Subsample (prevents overfitting) -Eta (prevents overfitting) -Gamma -Scale_pos_weight (unbalanced data) Eval_metric (AUC/error/RMSE..) -Max_depth -Alpha (L1) -Lambda (L2)	Multi CPU Single GPU (if gpu then set gpu_hist param)
Seq2Seq	Input and output = **sequence of tokens** Translation, text summary, speech to text. Uses RNNs CNNs	RecordIO-Protobuf (tokens must be ints) Tokenized text -> ints with text file	-Batch size -Optomizer (adam/sgd/rmspop) -Learning rate -Num_layers_encoder Accuracy/**BLEU** (compares multiple translations)/ Perplexity	Only GPU Only single instance Multiple GPUs on single instance
DeepAR	Forecasting one-dimensional time series data. Uses RNNs Find **frequency** / **seasonality**	JSON lines (Gzip or Parquet) Each record: timestamp + value Additional info like features/category can also be present alongside value	-Context_length – number of time points model sees -Epochs -Mini_batch_size -Learning_rate -Num_cells	CPU or GPU Single or multi machine. CPU only for inference
Object Detect	Identify all objs in image Bounding box, classification and a score. From scratch or ImageNet	RecordIO or Raw Image (jpg / png) With image format needs a JSON file for annotation of objects and locations	Base: VGG-16 or ResNet-50 Transfer / incremental training -Mini_batch_size -Learning_rate -Optimizer (sgd / adam / rmsprop / adadelta)	GPU for taining Multi GPU and multi machine CPU or GPU for inference

Model	Description	Data	Tuning		Hardware
Blazing Text	**Text Classification** – predict labels for a sentence. Web searches, information retrieval. Supervised. **Word2Vec** – Semantically similar words put close together. Useful for NLPs, but not an NLP itself. Individual words, not sentences or docs.	Supervised – one sentence per line. First word is "_label_" followed by label. Augmented manifest text format Word2Vec just wants a text file with one training sentence per line **Modes** - Cbow (continuous bag of words) - Skip gram - Batch skip gram (distributed over multiple CPUs)	**Text Classifications** - Epochs - Learning_rate - Word_ngrams - Vector_dim **Word2Vec**: - Mode - Learning Rate - Window Size - Vector_dim - Negative Samples		Only need a GPU if doing large training on text classification. Otherwise use CPU.
Object2Vec	Compute nearest neighbor of objects (similar ones) Visualize clusters **Genre prediction** Recommendations	Tokenized ints Training data = pairs/sequence of tokens {"label":1. "in0":[2 3 14 90 101…-}	Usual Deep learning:Dropout, early stopping, epochs, learning_rate, batch size, layers, activation function, optimizer, weight decay Enc1_network, enc2_network = choose your encoder type (CNN, LSTM or Average-pooled embedding layer)		Single Machine CPU or GPU Multi GPU on 1 machine Use **Inference preferred mode** when setting up
Image Classification	Assign 1 or more **labels** to image. Doesn't say where image is.	Apache MXNet recordIO (Not Protobuf!) Or raw jpg/png .lst file to map image to labels Augmented manifest img -pipe mode	ResNet CNN under the hood. + transfer learning -Batch size -Learning rate -Optimizer	-Weight decay -Beta ½ -Eps -Gamma	GPU training Multi GPU Multi machine CPU/GPU inference
Semantic Segmentation	**Pixel level** classification Useful for self-driving cars, robots Produces segmentation mask We can refer to **Instance Segmentation** as a combination of semantic segmentation and object detection	Raw Image (JPG/PNG) training/validation Label maps to describe annotations Augmented manifest img -pipe mode	MXNet Gluon Gluon CV FCN, PSP, DeeplabV3 Resnet50, ResNet101 - Algorithm/backbone^	-Epochs -Learning rate -Batch size -Optomizer	Only GPU – training Single machine CPU/GPU inference

Model	Description	Data	Tuning	Hardware
Random cut forest	**Anomaly Detection** Unsupervised Detect spikes in TS data. Assign anomaly score **AWS developed** product Used in **Kinesis Analytics**	RecordIO-protobuf or CSV Can use File or Pipe Mode Optional test channel for computing accuracy, precision, recall and F1 on labeled data	- **Num_trees** (increasing reduces noise) - Num_samples_per_tree (should be 1/X = ratio of anomalous/normal data)	CPU only for training and inference
Neural Topic Model	**Organize docs into topics** Uses deep learning Unsupervised – labels have no meaning	4 data channels (train=mandatory, validate, test, auxiliary) recordIO-protofbuf or CSV Words must be **tokenized** as ints File or Pipe Mode	Choose number of topics – **Num_topics** - mini_batch_size - Learning_rate Lowering these 2^ can reduce validation loss, but increase training time. **Deep Learning tuning** too!	CPU and GPU GPU training
Latent Dirichlet Allocation	**Organize docs into topics** No deep-learning (Unsupervised) Not always for docs – cluster customers, music classification Similar to NTM but cheaper	Train channel (mandatory), test channel RecordIO-protobuf (Pipe Mode) or CSV	Choose number of topics – **Num_topics** - Alpha0 (concentration param) Smaller vals = sparse topic mix Large vals (>1.0) = uniform mix/more topics	Single-instance CPU
KNN	**Classification** – find k closest points and choose most frequent **Regression** – Find k closest points and return avg val	RecordIO-protobuf or CSV training File or Pipe mode for either Dimensionality reduction stage in SageMaker.	- **K** (experiment until greater values don't help) - Sample_size	CPU/GPU training Inference: CPU – low latency GPU – high throughput on large batches
K-Means	Unsupervised – find K groups **K-means++** **Web-scale K-Means clustering**	Train chanel (mandatory), test chanel RecordIO-protobuf or CSV File or Pipe Mode for either	- K (elbow method, optimize tight clusters) - Mini_batch_size - Extra_center_factor - Init_method	CPU (recommend)/GPU Multi CPU, single GPU **Shard data by S3 key** for multi instance taining.

Model	Description	Data	Tuning	Hardware
PCA	Used for dimensionality reduction. - Unsupervised Returns list of **components** from most to least impactful	RecordIO-protobuf or CSV File or Pipe Mode for both	Algorithm mode: - **Regular mode** – sparse data - **Randomized** – Large # features/observations - Subtract_mean (unbiased data)	CPU or GPU
Factorization Machine	Sparse data like **click prediction**, recommend item. Supervised – classification or regression. Limited to **pair-wise** interactions (user->item)	**Only** RecordIO-protobuf with float32 Creates recommender matrix	Initialization methods for bias, factors, linear terms - Predictor_Type : Regression or Classification - Mini_batch_size - Num_factors	CPU or GPU **CPU recommended**
IP Insights	Unsupervised IP address usage patterns **Detect suspicious behavior**	CSV Only Training channel optional – could create an AUC score. **SM will automatically generate negative samples by randomly paring entities and IPs**	Num_entity_vectors: - Hash size - Set to 2x the #unique entity identifiers (users) -Epochs -learning rate -num_ip_encode_layers - rand_neg_sample_rate Vector_dim - Size embedding vector - Scales mode size - Too large =overfitting -batch_size -weight_decay	CPU or GPU (Better) Multi GPU
Reinforcement learning	Agent that explores space **Autonomous vehicles**, Dialog systems, HVAC systems Q-learning / MDP	Must design environment Optional envs offered: - Intel Coach - Ray Rllib - MATLAB - Simulink - EnergyPlus - Roboschool - PyBullet - Sumerian	Value for epsilon = the probability that we do not choose the optional path Anything else is custom Can use Hyperparameter tuning in SM	CPU or GPU (Better) Multiple GPU across multiple machines

SM Data & Resource Overview

- Deep Learning algos use GPU (P2/P3) for training.
- Inference less demanding so use CPU (C4/C5). Only use GPU if speed is absolutely necessary.
- **Managed Spot Training**: If you need to save money and time isn't a big factor. Make sure to set up S3 checkpoints.
- **Elastic Inference**: Add alongside a CPU instance to increase performance without cost of GPU. Only available with Deep Learning and built in Image Classification / Detection.
- Automatic Scaling: Scaling policy that works with CloudWatch
- SM will automatically distribute resources across endpoints

ContentTypes for Built-in Algorithms

ContentType	Algorithm
application/x-image	Object Detection Algorithm, Semantic Segmentation
application/x-recordio	Object Detection Algorithm
application/x-recordio-protobuf	Factorization Machines, K-Means, k-NN, Latent Dirichlet Allocation, Linear Learner, NTM, PCA, RCF, Sequence-to-Sequence
application/jsonlines	BlazingText, DeepAR
image/jpeg	Object Detection Algorithm, Semantic Segmentation
image/png	Object Detection Algorithm, Semantic Segmentation
text/csv	IP Insights, K-Means, k-NN, Latent Dirichlet Allocation, Linear Learner, NTM, PCA, RCF, XGBoost
text/libsvm	XGBoost

Sage Maker Features

SageMaker Features

Hyperparameter Tuning

- Define the params you care about and ranges
- Can deploy best set of params
- **Learns as it goes** (try not to do too much parallelization)
- Don't optimize too many params at once!
- Use reasonable ranges and logarithmic scales when possible.

Apache Spark

- **Pre-process data** with Spark – generate dataframe
- Call fit on SM Estimator (**Kmeans**, **PCA**, **XGBoost**) -> then have a SM Model
- Call transform on SM Model for inferences
- Connect notebook to remote EMR cluster running Spark

SageMaker Features

SageMaker Studio

- Visual IDE for ML – create/share Jupyter notebooks
- SM Experiments – organize, capture and compare Models

SageMaker Debugger

- Saves model state at intervals
- Rules: detect unwanted conditions – if triggered fires CloudWatch alarm
  Monitor system bottlenecks, framework operations, model params
  StopTraining(), Email(), SMS()
- Auto-generate training reports
- TensorFlow, PyTorch, MXNet, XGBoost
- Insights Dashboard – UI

SageMaker Features

Model Monitor

- Get alerts on quality deviations via Cloudwatch
- Vizualize data drift, model quality and feature drift
- Detect anomalies/outliers and new features
- Integrates with **Clarify** – explains model behavior and see bias

JumpStart = 1 click models from model zoos

Data Wrangle = Import/transform/analyze data (like GLUE, but in SM)

Feature Store = Find features and organize into groups

Edge Manager = Software agent for edge devices. Works with Neo. Helps collect data from edge devices for future training/eval.

SM Neo = Optimize for specific devices. Consists of a compiler and runtime. Then use **IoT Greengrass** to deploy your model to edge devices.

SageMaker Notebooks

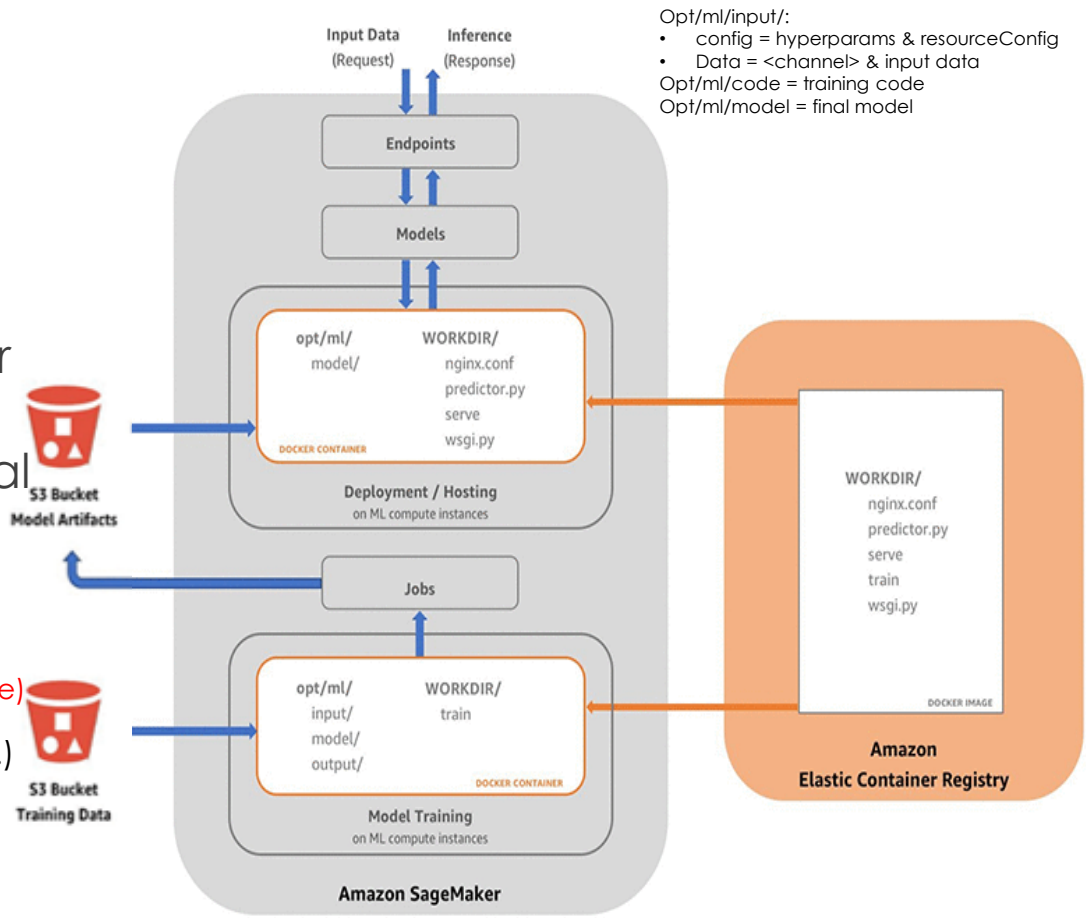
SageMaker notebook instance is an ML compute instance running the Jupyter Notebook App

- Can use **an Elastic Inference** to accelerate requests – Good option if no GPU wanted
- Use **KMS** to encrypt your notebook
- Choose hardware when training, then another when creating your inference endpoint.
- **Lifecycle configuration**: Shell scripts that run when create/start a notebook. (**install packages**)

SageMaker +Docker

Every model is hosted in Docker container and registered in ECR

- Pre-built models available for DeepLearning, Spark, MXNet, TensorFlow, PyTorch
  (**Distributed training** not automatic – must use **Horovod** or **Parameter Servers)**
- Can create multiple model endpoints, use **model weights** to do A/B tests.
- **Inference Pipelines**: Any combo of pre-trained, built in or your algos in Docker containers. Combine pre-processing, predictions, post-processing. Can handle real time or batch transforms.
- Environment Variables:
  - **SAGEMAKER_PROGRAM**: Defines the script that will run the training (/opt/ml/code)
  - TRAINING_MODULE / SERVICE_MODULE: Load modules (TensorFlow, MXNet etc.)
  - SM_MODEL_DIR: Checkpoints saved and pushed to S3
  - SM_CHANNELS / SM_HPS / SM_USER_ARGS….



Opt/ml/input/:
- config = hyperparams & resourceConfig
- Data = <channel> & input data

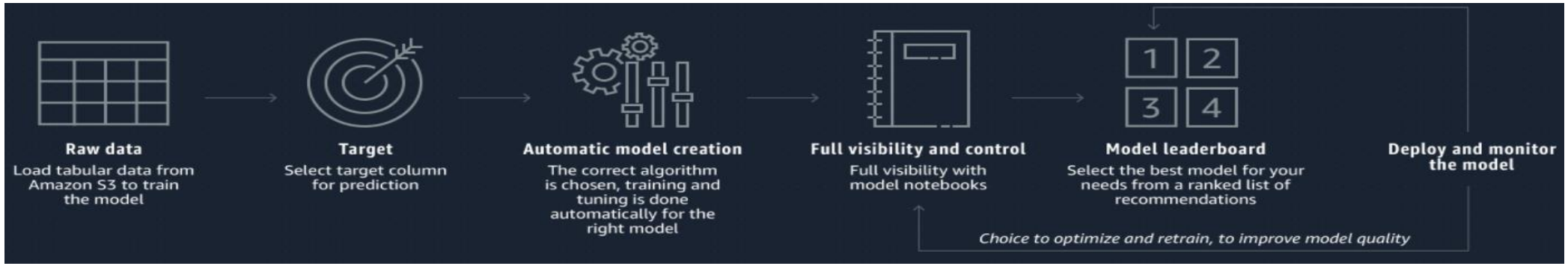
Opt/ml/code = training code
Opt/ml/model = final model

SageMaker Security

- Use IAM – only give users necessary permissions. Give roles to SageMaker itself.
- Use MFA, SSL/TLS when connecting, CloudTrail for API usage log
- Use KMS for notebooks and all jobs (training, transforms, endpoints…)
- S3 can encrypt buckets for training data and hosting models. S3 can also use KMS
- Inter node traffic encryption can slow down training but might be necessary
- If using a VPC use a S3 VPC endpoints. Custom endpoint policies and S3 bucket policies
- Notebooks **are internet-enabled by default** – If disabled your VPC needs an interface endpoint (PrivateLink) or NAT Gateway and allow outbound connections.
- CloutWatch can:
  - log Invocations and latency of endpoints
  - Health of instance nodes (CPU, RAM etc.)
  - Ground Truth – (active workers, how much they're doing)
- CloudTrail records actions from users

SageMaker Autopilot

- Automates algorithm selection, preprocessing, tuning, infrastructure
- Load data from **S3** -> choose target column -> automation -> leaderboard
- Problem Types: **Binary/multiclass classification, regression** (only 3 options)
- Algo types: Linear Learner, XGBoost, Deep Learning MLPs
- Data: Must be CSV
- Autopilot Explainability with **SM Clarify** for transparency
  Shows what are most important features



Sage Maker
High Level Services

High Level Services

Amazon Comprehend

- NLP and Text Analytics for extracting entities, language, key phrases and sentiment analysis.
- Use cases**: Social Media Analysis**, Automatic Feedback Analytics.

Amazon Translate

- Deep learning from translation that also supports custom terminology via CSV or TMX.
- Use cases: **Language translation**

Amazon Transcribe

- Speech to text for English, French and Spanish. Also includes speaker identification and custom vocab.
- Use Cases: Add subtitles, **Record Customer Calls** (specific version for medical and PPI)

Amazon Polly

- Text to speech. Has the option of using Lexicons (acronyms), SSML (Speech tones, pronunciation), Speech Marks (Lip-synching)
- Use Cases: **Create Lectures**, Assistance to Visually Impaired

Amazon Forecast

- Fully managed time series prediction.
- Use Cases: **Financial Planning**, Trend Analysis

High Level Services

Amazon Rekognition

- Computer vision for object, facial, text detection and video analysis.
- Video must come from Kinesis Video Streams.
- Integrates with lambda functions
- Use Cases: Moderate Images, **Intruder Alarm**, Facial Recognition

Amazon Lex

- Inner workings of Alex – chatbot takes Utterance("I want to order a pizza"), builds Intents (order_pizza), fills slots (pizza_toppings)
- Use Cases: Chat Bot, **Alexa**

Amazon Personalize

- Fully managed recommender engine that uses historical and real time data. Can weight on time, filter certain users/robots – data not shared across accounts
- Uses Recipes (possible algorithms):
  - USER_PERSONALIZATION - personal recommendations
  - PERSONALIZED_RANKING - hierarchical recurrent neural network (HRNN) for providing a list of best recommendations
  - RELATED_ITEMS – predicts item similar to a given item
- Use Cases: **Recommend Items** (Can use relevance +price for example to max profits), **Search Engine Results**

High Level Services – Less Important?

Amazon Textract

- Optical Character Recognition – used for extracting data from forms/tables/values – Banking application

DeepLens

- Deep learning enabled camera. Integrates with recognition, SageMaker, Polly, TensorFlow, MXNet, Caffe

Amazon Lookout

- Equipment (Detects abnormalities from sensor data), Metrics (monitors anything), Vision (detect defects in equipment etc.)

Amazon Monitron

- End to end system for monitoring industrial equipment and predictive maintenance. Specific AWS Sensor (piece of equipment + ML application.

TorchServe

- Model Serving framework for PyTorch. Open source project with Meta

AWS Neuron

- SDK for ML inference specifically on AWS inferential chips (ECS Inf1). Integrated with SageMake or other ML libs.

AWS Panorama

- Computer vision at the edge. Brings computer vision to your existing IP cameras. **Good for Security Cameras**

Amazon DeepComposer

- AI powered keyboard – more for educational purposes

Amazon Fraud Detector

- Upload your own fraud data. Build a custom model from a template.

Amazon Cloud Guru

- Automated code reviews. Resource leaks, race conditions, profiler

Contact Lens for Amazon Connect

- For customer call centers. Ingest audio and do sentiment analysis
- Find utterances that correlate with success. Categorize, discover emerging issues

Amazon Kendra

- Enterprise search with natural language.
- Combines multiple file systems and uses ML from user interaction to refine searches

Amazon Augmented AI (A2I)

- Human review of ML predictions. Uses Amazon mechanical Turk
- Integrated with Amazon Textract and Rekognition
- Very similar to ground Truth, but only for low confidence predictions.