Google Cloud Professional Data Engineer Exam Study Materials

By Leverege Engineers

Machine Learning

ML Terms

- **RMSE vs. Cross Entropy**: both used to train neural networks but cross entropy is preferred for **classification** whereas RMSE is better for **regression**. Cross entropy is a value between 0 and 1 to measure the probability value where a perfect model would have a log loss of 0.
- **Accuracy vs. Precision vs. Recall vs. F1 (Confusion Matrix):**
    - **Accuracy:**
        - Ratio of correctly predicted observation to the total observations. Only good if the dataset is symmetric
        - (true positive + true negative) / (TP + FP + FN + TN)
    - **Precision:**
        - Ratio of correctly predicted true positives to total predicted positive outcomes. Good to measure false positive measures (e.g. spam)
        - true positive / (false positive + true positive)
    - **Recall:**
        - Ratio of correctly predicted positive to all the actually positive. Good to measure false negatives (e.g. bank fraud)
        - true positive / (true positive + false negative)
    - **F1-value:**
        - Weighted average of precision and recall
        - 2 * (precision * recall) / (precision + recall)
    - Accuracy is a good measure in symmetric data sets. Precision and Recall measure the degree of skew or asymmetry of the dataset.
- **Wide vs Deep models**
    - Wide: memorization
    - Deep: generalization
    - Used for recommendation, search, and ranking problems
- **Feature Engineering**
    - Feature Cross: synthetic feature to encode nonlinearity in the feature space (e.g. one-hot encoding). Otherwise known as activation functions.
    - Bucketing: transforming numerical features into categorical features
    - Hyperparameter tuning: tuning the variables that govern the training process itself (number of hidden layers, how many nodes, learning rate, etc)
    - Embedding: mapping from discret objects (e.g. words) to vector of numbers
    - Sigmoid: useful for mapping numbers to probability in logistic regression
    - Softmax: like a sigmoid, but for multiple inputs
    - Logits: very big or very small numbers that get mapped to probabilities

    - Regularization: approaches to reduce overfitting (dropping layers, adding weights)
    - Optimizer: operation that change the weights and biases to reduce log (adagrad or adam)
    - Bias vs. Variance:
        - Bias: diff between the avg prediction and the correct value → leads to underfitting
        - Variance: variability of model prediction between multiple runs → leads to overfitting
        - We want low bias, low variance
- Ensemble learning: training multiple models with different parameters to solve the same problem
- Numerical instability: issues with very large or small values due to limits of floating point numbers. It can lead to gradient explosion.

Cloud ML Engine

Supports TF, scikit-learn, and XGBoost

- Cloud ML Engine Terms:
    - Replica: running job on a given node
    - Master: one replica is designated the master, signals the overall job status and manages the task of the distribution
    - Worker: 1+ replicas may be designated as workers to run the job configurations
    - Parameter servers: 1+ replicas designated, coordinate shared model state between the workers
- Tiers
    - Basic: master
    - Standard: 1 master + 4 workers + 3 PS
    - Premium: 1 master, 19 workers
    - Custom Tier: allow you to specify the number of workers and parameter servers
- Online vs. Batch
    - Online: minimize latency of predictions and return in the response message
    - Batch: handle high volume and complex models, return in GCS
- Training Process
    - Local training
    - Upload to GCS (must be regional): BQ works as a data source, but GCS is the preferred method and gcloud only works with GCS
    - Run training process on ML engine → outputs a model on GCS
    - Deploy model on ML engine (batch or online)

    - Serve requests
- APIs
    - ML APIs:
        - Speech-to-text
        - Text-to-speech
        - Translation
        - Vision
        - NLP: sentiment, intent, entity, syntax
        - Video intelligence
    -
- Things to still review
    - Basic TF architecture
    - Control APIs: Python API
    - Custom ML models
    - Low level APIs: C++ engine and API
    - Hardware types
    - Estimator API for Linear/DNN
    - Tips:
        - If the set of possible values are unknown in advance, use categorical_column_with_hash_bucket
        - Train locally with gcloud ml-engine local train
    - Tensorboard

Pub/Sub

- Benefits:
    - Availability: buffer during outage
    - Low latency & high throughput
    - Consistency
- Deduplication is possible by using Pub/Sub's message_id when received by the subscriber
- IAM: on topics and subscribers

DataLab

- Supported languages: Python, SQL & JS for BQ
- Can run locally, on Docker via GCE (ssh access), or Docker + Gateway
- Can analyze data in BQ, GCE, and GCS
- Datalab notebooks are per-user only. Need to sync with Cloud Source Repository to share work.
- Data Studio IAM and connectors (see Qwiklabs)

- Works well with DataProc as well for PySpark

DataStore

- Potential Cassandra replacement for <10TB of data (documentDB)
- Fast read, but slow write (must update indexes for every write)
- Can support multiple indexes
- Allow ACID transactions and SQL-like queries to keys and property/field level, but can't support complex joins with inequality filters
- Use case: structured data from AppEngine (html, XML, JSON)
- Error handling:
    - Unavailable, deadline exceeded: exponential back off
    - Internal: don't retry this request more than once
    - Other: fix before retrying

DataFlow

- Guarantees exactly once processing. During update, it uses drain to maintain in-flight data for streaming pipelines.
- IAM:
    - Developer: enable developer interacting with the job
    - Worker: permissions for SA to work on the pipeline
- Terminology
    - PCollections: abstraction that represents a distributed, multi-element data set
    - Transformation: data processing operation / step in the pipeline
    - DirectPipelineRunner: local version of the pipeline
    - Watermark: notion of when all data in a certain window can be expected to have arrived
    - SideInput: additional inputs to a transform
- Triggers:
    - Determines when a window's content should be processed
    - Default: trigger first when the watermark passes the end of the window and then trigger again every time there is arriving late data
    - Time-based triggers:
        - Event time: timestamp on the data element (default)
        - Processing time: time when data element is processed at any given stage in the pipeline. Can add timestamps with ParDo and DoFn
    - Data-driven triggers: fire when data meets a certain property in a window (when it reaches a certain number of data elements)
    - Composite triggers: combine multiple trigger types
- Windowing: divides a PCollection according to timestamps

    - Fixed time
    - Sliding time
    - Session window: specify minimum gap time
    - Single global window (default)
- Inputs: PubSub, DataStore, Avro, Kafka
- Outputs: BQ, Cloud ML, BigTable
- Aggregations:
    - GroupByKey: similar to the Shuffle phase, collection of key/values pairs
    - Combine: combines collections based on logic
    - CoGroupByKey: relational join
- Accumulation vs. Discarding: either to keep previous data every time a new trigger fires

BigTable

- Can update number of clusters/replication settings and dev->prod without downtime
- Can't switch from SSD to HDD without exporting data first
- IAM: control at the project and instance level (clusters, nodes). Tables belong to instances and not clusters/nodes.
- **Size limits:**
    - Row key: 4KB
    - Value in cell: 100MB (recommended 10MB)
    - All values in single row: 256 MB (recommended 100MB)
    - Max 1000 tables in each instance
    - Max 100 column families
- BigTable uses **Sorted String Tables** to organize data. These are immutable pre-sorted key, value pairs.
- Data Model:
    - Row key: unique identifier for an entity
    - Column: contains individual values
    - Column family and column qualifier identifies a column
- BT periodically compacts the data and reorganize
- Optimization:
    - Group columns of data you are likely to query together
    - Use short column names
    - Organize into column families
    - Use tall and narrow tables (append new rows -> tall, collapse flags into a single column -> narrow)

    - BT selects the most recent value that matches the key → when deleting/updating, we write a new row and compaction removes the deleted row later → deleting/updating temporarily increase the disk usage
- Performance Test:
    - Run a heavy pre-test for several minutes and test for 10+ mins
    - BT learns about your access patterns and will adjust the metadata to balance your workloads
    - Stay below the recommended storage utilization per node
    - Performance test may take mins to hours and requires at least 300GB of data
- Design:
    - If query is most recent data, use a reverse timestamp at the end of the key
    - Tables are sparse: empty columns don't take up space
    - Avoid hotspotting
        - Field promotion (preferred): move fields from the column data into the row key to make writes non-continguous
        - Salting: add an additional calculated element to the row key
    - For time-series data, use tall/narrow tables
    - Denormalize: use multiple tall and narrow tables
- Features:
    - For 1TB+ data
    - No transaction and relational support
    - **ACID only at the row level → avoid schema designs that require atomicity across rows**
    - Single key lookup (no property search)

Cloud SQL & Spanner

- Both have managed backups & automatic replication
- Spanner: can optionally define relationships between tables to physically co-locate their rows for efficient retrieval

GCS

- Nearline (~1 month) vs coldline (~1 year)
- Store data for BQ and Dataproc
- Encryption
    - Default: Google managed
    - Customer-supplied: create and manage your own for server-side
    - Customer-managed: generate and manage your keys via KMS
    - Client-side: encrypt before sending data to GCS

Dataproc

- Managed Hadoop and Spark
- Prefer ephemeral model with GCS to store data and support multiple, temporary processing clusters instead of monolithic persistent HDFS cluster
- **Hadoop Environment**
    - **Hadoop**: open source MapReduce framework
        - HDFS: Hadoop Filesystem
        - YARN: yet another resource negotiator (job scheduling and coordination)
        - MapReduce: parallel processing paradigm
    - **Pig:** scripting language that compiles to MapReduce jobs that supports Avro. It pulls unstructured data and performs ETL into data warehouses (DataFlow)
    - **Hive**: SQL like data warehousing system and query language. Hive abstracts away underlying MapReduce jobs and returns HDFS in the form of tables. (BQ)
    - **Spark**: fast, interactive framework for SQL, streaming, ML. Uses fast in-memory approach to solve Hadoop questions
    - **RDD**: resilient distributed datasets which are immutable, lazily evaluated, and tracks lineage
    - Sqoop: bi-directional transfer of data between Hadoop and structured datastores. You can run Sqoop on a Dataproc Hadoop cluster to built-in GCS connector. (Use Sqoop to import data directly into GCS)
    - Oozie: workflow scheduler to manage Hadoop jobs
    - Cassandra: wide-column data store based on BT and Datastore. No ACID compliance, but can handle high throughput of writes
    - MongoDB: schema-less document store (Datastore)
    - **HBase**: non-relational, NoSQL, scanning hugh, 2D, join-less tables that are useful for search, analyze log (BigTable)
    - Flink/Kafka: stream processing framework
    - Beam: programming model to define and execute data processing pipelines. Beam's backend can be Apex, Flink, SPark, or Dataflow. Modeled as a Direct Acyclic Graph.
    - Redis: in-memory search for cache or pubsub kind of system
    - Avro: Data serialization framework that supports parallel read even when it's compressed. No encoding issues.
- Configuration options
- Customize clusters with initialization scripts: set initialization actions, modify config files using cluster properties, or log into master node
- IAM: need Dataproc Worker role and need RW access to GCS
- Access: SOCKS proxy, SSH into the master node

MISC Topics

Cloud Interconnect: option to extend on-prem network to GCP VPC.

- Dedicated Interconnect: 10+ Gbps
- Partner Interconnect (via service provider): 50+ Mbps or if Google colocation facility option is not available
- Cloud VPN: public internet network

Transfer Services:

- Transfer Appliance: hardware appliance used to securely migrate 100+TB to 1 PB of data
- Cloud Storage Transfer: import online data into Cloud Storage from S3, HTTP location, or GCS
- BigQuery Data Transfer: schedule and automate data transfers from Google tools to BQ

Compliance:

- GDPR: in EU regulates how businesses can collect, use, and store personal data

Other Services:

- Cloud Deployment Manager: Terraform equivalent
- Cloudshell: Temp VM, recycled every hour

Data Studio

- Prefetch cache is only active for data sources with owner credentials
- Don't use cache if you want frequently changes or if BQ cost is high for caching all the time

BigQuery

Know syntax for Standard vs Legacy SQL

Data Loading

- Load from GCS, Google Ads, streaming insert, Dataflow, local data source, DML (data manipulation language) for bulk inserts
- Format: CSV, JSON (newline), Avro, Parquet, ORC, Datastore and Firestore on GCS only
- Encoding is UTF-8 by default

- Use nested and repeated fields to load data denormalized

Best Practices

- Use WHERE or EXCEPT to select subset of the columns
- Avoid self-joins and cross-join (generate more output than inputs)
- Avoid skew (unequally sized partitions)
- Use query preview and price estimator
- Stage queries to save cost (re-running the entire query costs money)
- Use expiration dates on default table to expire staging queries
- Use ORDER BY in the outermost query
- Avoid repeated data transformation via SQL queries

Misc Facts

- BQ Data Transfer service periodically imports Campaign Manager, Google Ads, and Youtube reports to BQ.
- You can add labels to datasets, tables, or jobs. A label with an empty value is a tag.
- You can use KMS to manage own encryption keys
- Exports: JSON/CSV/Avro and GZIP. To export more than 1GB, use * in the filename
- Cache: 24 hours and on by default unless destination table is specified or if referenced tables/logical views have changed, querying streaming or external data source or or using a wildcard
- BQ Colors:
    - Yellow: Wait
    - Purple: Read
    - Orange: Compute
    - Blue: Write
- **Billing:** storage (active or long-term - no change for 90 days), querying (on-demand or flat-rate), streaming insert
- IAM: dataset and project level. Can set up authorized views to share results without giving access to underlying data
- **Bucketing**: equal size partitions/splits based on a hash function of a column. Each bucket is a separate file.
- **Windowing**: window functions analyze partitions (windows) of a dataset
- Partitions: cannot change existing table into partitioned table
- Batch queries: can queue a batch of queries but only counts for daily and switches to interactive if idle for 24 hours
- View: virtual table defined by a SQL query
- **Limits:**
    - Each partitioned table can have up to 2500 partitions
    - Daily limit is 2000 partition updates per table

    - Rate limit is 50 partition updates/10 sec

External Data Sources

- Works with BigTable, GCS, and Drive
- Possible use cases:
    - Loading and cleaning data in one pass by querying the data from external data source then writing the cleaned result to BQ storage.
    - Having a small amount of frequently changing data that you join with other tables
- Limitations:
    - Slower query performance
    - No data consistency
    - Can't use wildcard table query
    - Can't directly export data from an external data source.
    - Cannot query data stored in Parquet or ORC format
    - Results are not cached
    - Must be in the same region
- Moving BQ between locations
    - Export to same region GCS bucket
    - Copy data to destination region GCS bucket
    - Import data to BQ

Google Cloud Dataprep

- Import data from GCS, BQ, or custom
- Jobs are stored as CSV on GCS by default
- Dataprep used for quick data transformation (deduplicating, removing nulls, unions, etc)