Notes for machine learning

Ashok Kumar
https://github.com/meetashok

June 30, 2020

Contents

1. Linear Algebra

References

1. Linear Algebra Review and Reference, Stanford

1.1 Introduction

A vector has both a magnitude and a direction. Let's say we want to represent velocity of an object in a two-dimensional space, then it could be written as $\vec{v} = (3, 4)$. This means that the object is moving at a speed of 3 in the x-direction and at a speed of 4 in the y-direction. Vectors passing through origin to the coordinate are known as position vectors.

Vectors can be used to represent lines. Let's say we have a position vector, $\vec{a} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, then the set $L = \left\{ t\vec{a} \mid t \in \mathbb{R} \right\}$ represents the set of lines passing through origin with the slope $\vec{a}$. If we want a general definition of lines passing through two position vectors, then we can write, $L = \left\{ \vec{a} + t(\vec{a} - \vec{b}) \mid t \in \mathbb{R} \right\}$. This definition of representing lines is extensible to higher dimensions. This gives us a parametric definition of line. In higher dimensions, we cannot define a line with just x, y, z and one equation.

Let's take two vectors, $\vec{v}_1 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, $\vec{v}_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$. The linear combination of the vectors is defined as $c_1\vec{v}_1 + c_2\vec{v}_2$. It can be shown that for these values of $\vec{v}_1, \vec{v}_2$, any vector in the 2-dimensional space can be derived via the linear combination. So, we can say that $\text{Span}(\vec{v}_1, \vec{v}_2) = \mathbb{R}^2$. Not all vectors in 2-d space will have a span of $\mathbb{R}^2$, e.g., $\vec{v}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $\vec{v}_2 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$. For 2-d space, the vectors $\vec{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\vec{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ are known as the basis vectors.

Trace of a square matrix is defined as the sum of its diagonal elements.

1.2 Eigenvalues and eigenvectors

Let's take a 2×2 matrix, $\mathbf{A}$. When we multiply this matrix with a vector, $\mathbf{v}$, the vector gets transformed. The transformation could be of two types, scaling and rotation. For a given transformation, there could be some vectors that only get scaled but there is no rotation. Such vectors are known as eigenvectors. The factor by which they get scaled are known as eigenvalues.ed

This can be represented as $\mathbf{Av} = \lambda\mathbf{v}$. To solve for lambda, we can do the following -

$$\begin{aligned} \mathbf{Av} &= \lambda\mathbf{v} \\ (\mathbf{A} - \lambda\mathbf{I})\mathbf{v} &= 0 \\ \det(\mathbf{A} - \lambda\mathbf{I}) &= 0 \end{aligned} \tag{1.1}$$

The solutions for λ from the above equation will give us the eigenvalues which could be used to solve for eigenvectors. An eigenvector multiplied by a scalar is also an eigenvector. So, typically, eigenvectors are reported with normalized norm.

Summation of the eigenvalues of a matrix is equal to the trace of the matrix. Product of the eigenvalues is equal to the determinant.

$$\begin{aligned} \mathrm{tr}(\mathbf{A}) &= \sum_{i=1}^{n} \lambda_i \\ \det(\mathbf{A}) &= \prod_{i=1}^{n} \lambda_i \end{aligned} \tag{1.2}$$

Geometrically, determinant of a square represents the volume inside the parellelopipe formed by the constrained (weights are in the range of $[0,1]$) linear combination of column vectors. If all columns are linearly independent then the volume will be non-zero and the matrix is invertible. But if any of the columns is linearly dependent then the volume will be a flat-surface and hence it will be zero. In this case, the matrix is singular or non-invertible.

1.3 Matrix operations

For matrix multiplication, multiplying two matrices of size $n \times n$, will take $O(n^3)$ time complexity as each element requires multiplication of two vectors of size n. Multiplying two matrices of sizes $n \times p$ and $p \times m$ will have the time complexity of $O(npm)$.

Inversion of matrices can be done by two methods, a) Gauss-Jordan elimination by constructing the double wide matrix and using matrix-row operations, b) by calculating the adjugcate matrix and dividing by determinant. Gauss-Jordan method is $O(n^3)$, determinant operation is $O(n!)$.

2. Programming

2.1 Data Structures

2.1.1 Array

Arrays are collections of items. The items could be of any type, either primitive data types or user-defined data types. When arrays are initialized, user has to specify the type and size of the array. Advantage of arrays is that elements of arrays can be access in constant time. But the disadvantage is that the size of the array needs to be declared at the time of initialization.

Python `numpy` arrays are implemented as arrays. Python lists are implemented as dynamic arrays. But the elements of the list are not the object themselves, but references to objects. If the list needs to increase in size, existing array is destroyed and a new array of double the size is created. The reference to objects are maintained.

2.1.2 Hashmap

3. Statistics

3.1 Concepts

If X is a random variable with probability distribution function, $f_X(x)$. Then the expected value of the variable is given by $\mathbb{E}(X) = \int_{-\infty}^{\infty} x f_X(x) dx$. Variance is given by $\text{var}(X) = \int_{-\infty}^{\infty} (x-\mu)^2 f_X(x) dx$.

$$\text{Univariate normal distribution:} f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{\left(\frac{x-\mu}{\sigma^2}\right)} \tag{3.1}$$

3.1.1 Type I & Type II errors, null-hypothesis

The null hypothesis is the default position that there is no relationship between the variables that are being measures. For example, if we are trying to assess if a person has a disease based on the symptoms, then the null hypothesis could be that the person is healthy.

1. Rejecting null-hypothesis: Person has a disease
2. Accepting null-hypothesis: Person is healthy

If the null-hypothesis is falsely rejected, that is the person is said to have a disease, then that's a false positive (Type I error). If the null-hypothesis is falsely accepted, that is the person is healthy, then it is a false negative (Type II error).

3.1.2 Central Limit Theorem

Suppose we have a distribution with population mean, μ and population variance, σ^2 but we do not know the distribution of the population. If we take samples of size n from the population, and let's say that the sample mean is μ_s. The central limit theorem states that the sample mean is normally distributed. This means that if we take multiple samples from any distribution, then the means of the samples will be normally distributed. Further, the mean of the sampling distribution will be equal to the population mean, μ, and standard deviation will be $\frac{\sigma}{\sqrt{n}}$. Hence, the variance of sampling distribution decreases as the sample size is increased.

3.1.3 Frequentist v/s Bayesian inference

The main different between frequentist and bayesian approach arises from how they think about probability. Frequentists think of probability based on the number of times a result is repeated in a series of measurements. Hence, for frequentists probability has a meaning only in a limiting case of repeated measurements. Let's say we are trying to measure the flux of a star, and make n observations. Then using the frequentist approach we can assume a distribution for flux, and then calculate the true flux by using the maximum likelihood approach (MLE). In this case, it will be the simple mean of n observations. In strict frequentist view, it is meaningless to talk about the distribution of flux, as flux is a fixed quantity. Bayesian think of probability as related to their knowledge of an event. As such, bayesian can talk about their estimate of flux as their is uncertainty in their knowledge.

Further reading: Stackexchange

4. Introduction to Statistical Learning

4.1 Introduction

4.1.1 CRISP-DM

CRISP-DM is an industry wide process for managing data mining projects. The main steps of the process are:

1. Business understanding
2. Data understanding
3. Data preparation
4. Modeling
5. Model evaluation
6. Model deployment

4.1.2 Predictions versus Inference

There are two key problems in machine learning: prediction and inference. For prediction, we can treat the model as black box. We only care about the accuracy of the predictions. For inference, we are interested in knowing how the features impact the response. For example, in case of housing price modeling, we may be interested in knowing the impact of adding additional rooms to the price of the house.

In machine learning, we are trying to build a model between features, $x \in \mathbb{R}^p$ and the response variable, y.

$$y = f(x) + \epsilon \tag{4.1}$$

Our goal is to find a function $\hat{y} = \hat{f}(x)$ such that it approximates f. The expected value of the squared error is given as,

$$\mathbb{E}\left[(y - \hat{y})^2\right] = \left(f(x) - \hat{f}(x)\right)^2 + \sigma^2 \tag{4.2}$$

Here σ is the variance of the error term ϵ. The First part of equation 4.2 is termed as the reducible error and the second part is the irreducible error. Variance of the error term, ϵ places an upper-bound on the accuracy that a model can achieve, but is usually unknowable.

4.1.3 Parametric versus non-parametric

When building a machine learning model, we have two main choices - parametric model or non-parametric model. In parametric model, we assume a fixed form for f. For example, in linear regression, we assume, $f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$. Once a functional form is assumed, the problem reduces to estimating the parameters. In non-parametric approaches, no explicit assumption is made about the functional form of f. The goal is to get a smooth estimate of f. Parametric models are generally easier to interpret. Non-parametric models require much more data to fit. Also, because non-parametric models are much more flexible, they are more prone to over-fitting. Hence, simpler parametric models may sometimes be more accurate than non-parametric approaches. Examples:

1. Parametric model: Linear regression, Lasso
2. Non-parametric model: Splines

4.1.4 Supervised versus unsupervised methods

We can use a supervised algorithm if we have both the features and corresponding response available to us. But if only the feature observations are available, then the choices of analysis we can do is limited. One such analysis could be to cluster the observation into group such that within-group variation is lower compared to inter-group variation between the data points.

There is another class of methods called semi-supervised learning. Suppose we have n data points, out of which m ($m < n$) data points have a response variable and the rest do not. If we want to build a supervised learning model, we can only use subset with labels but the unlabeled data set also contains valuable information about the underlying structure of data. So, we should try to use the unlabeled data as well. One way of doing so is via pseudo-labeling.

In pseudo-labeling, we start by building a model on the data with labels. Then we predict the labels for unlabeled data. We then retrain the model by concatenating the labeled and pseudo-labeled data. We may want to limit the pseudo-labels where the label confidence is high. This approach works well in Kaggle competitions when we have a relatively small training data but a large test data. This approach is useful when it is difficult or expensive to acquire labeled data. Figure 4.1 shows how used of unlabeled data could lead to a more accurate decision boundary (source).

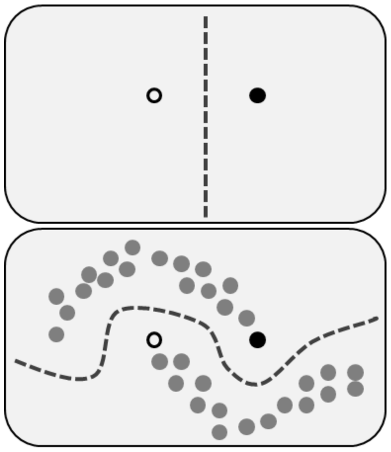
Figure 4.1: Only using black and white points, would get us the first decision boundary. Knowledge of more unlabeled data could lead to a more accurate boundary - clustering then labeling clusters with labeled data

Another example of semi-supervised approach is training CNN using adversarial example. We know that CNN models are highly susceptible to minor noise in the data. Adversarial training creates new training data by adding minor noise to images which keeping the labels same.

Links: (1) Medium post on semi-supervised learning, (2) pseudo-label paper

4.1.5 Bias variance trade-off

For a regression problem, the expected value of squared error can be decomposed into two terms, variance and square of bias (and irreducible error term). This means that to build an accurate model, we must have low bias and low variance. Difference between training error and Bayes's error is called avoidable bias. Error analysis is another important technique to understand the source of avoidable bias. Variance refers to how much the error changes because of change in training data. If the model is highly flexible, then the predictions could change a lot with changes in training data. Bias refers to the error arising from using too simple a model to depict the real association between response and predictors. This decomposition is depicted in equation 4.3.

$$\begin{aligned}\mathbb{E}\left[(y-\hat{y})^2\right] &= \mathbb{E}\left[\left(f(x)+\epsilon-\hat{f}(x)\right)^2\right] \\ &= \underbrace{\left[\left(f-\mathbb{E}(\hat{f})\right)\right]^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}\left[\left(\hat{f}-\mathbb{E}(\hat{f})\right)^2\right]}_{\text{Variance}} + \sigma^2\end{aligned} \tag{4.3}$$

Bias-variance trade-off isn't a concern any more as we now have many tools that can specifically tackle the problem. If there is a large difference between the training error and the irreducible error (Bayes's error for classification), then the model has a bias problem, i.e., too simple a model is being fit. If there is a significant difference between the validation and training error, then there is a variance (or over-fitting issue).
Solutions for bias problem:

1. Fit a more complex model (more layers in neutral network)
2. Train longer
3. User better optimization algorithms
4. Hyper-parameter search
5. Different neural architecture
6. Get more training data

Solutions for variance problem:

1. Regularize the models (Dropout, batch-normalization, penalty)
2. Get more data
3. Data augmentation
4. Different architecture or hyper-parameter search

It is critical that the validation data and the test data come from the same distribution. For example, if we are building a cat-classifier end-users, it is possible that the images from the web are of higher quality than images uploaded by users. But having images from web in training data could be useful for training. The validation and test data should not have images from web if they are deemed to have a different distribution.

4.2 Naive Bayes Classification

Naive Bayes is a classification technique. To understand it, let's start with the definition of bayes' rule.

$$\mathbf{P}(A|B) = \frac{\mathbf{P}(B|A) \cdot \mathbf{P}(A)}{\mathbf{P}(B)} \tag{4.4}$$

Let's say we want to build a classification model to detect spam v/s ham. For a new email message with words $w_1, w_2, \ldots, w_n$ we can calculate,

$$\mathbf{P}(\text{spam}|w_1, w_2, \ldots, w_n) = \frac{\mathbf{P}(w_1, w_2, \ldots, w_n|\text{spam}) \cdot \mathbf{P}(\text{spam})}{\mathbf{P}(w_1, w_2, \ldots, w_n)} \tag{4.5}$$

Naive bayes algorithm assumes that the conditional probabilities of each word is independent, hence the term naive. So, equation 4.6 can be expanded as,

$$\begin{aligned}\mathbf{P}(\text{spam}|w_1, w_2, \ldots, w_n) &= \frac{\mathbf{P}(w_1|\text{spam}) \cdot \mathbf{P}(w_2|\text{spam}) \ldots \mathbf{P}(w_n|\text{spam}) \cdot \mathbf{P}(\text{spam})}{\mathbf{P}(w_1, w_2, \ldots, w_n)} \\ \mathbf{P}(\text{spam}|w_1, w_2, \ldots, w_n) &\propto \mathbf{P}(w_1|\text{spam}) \cdot \mathbf{P}(w_2|\text{spam}) \ldots \mathbf{P}(w_n|\text{spam}) \cdot \mathbf{P}(\text{spam})\end{aligned} \tag{4.6}$$

$\mathbf{P}$(spam) is known as the prior probability and can be estimated as the count of spam messages in training set divided by total message. Similarly, conditional probability $\mathbb{P}(w|\text{spam})$ the number of times the word w occurs in spam messages divided by total words in spam messages. We usually add 1 to the frequency of all words, so that none of the conditional probabilities are zero, otherwise the posterior probability will be zero. If numerical features are involve, then we assume normal distribution with sample mean and variance, and then calculate conditional likelihood.

4.3 Linear Regression

Assumptions for linear regression:

1. The features should be normally distributed

4.4 Generalized linear models (GLM)

GLM is a generalization of linear regression. In linear regression, it is assumed that the actual value, y_i has a normal distribution with an expected value of $\hat{y}_i$ and variance ϵ^2.

$$\begin{aligned} \hat{y}_i &= \beta_0 + \beta_1 x_i \\ y_i &= \mathcal{N}(\hat{y}_i, \epsilon) \end{aligned} \tag{4.7}$$

GLM allows for two relaxations, a) transforming the linear predictor with a link function, b) in linear regression, we assumed that the variance of y_i is the same for all values of x_i. But GLM allows for the variance to vary with $\hat{y}_i$. For example, if we want to model Poisson data (discrete non-negative response, and variance increased with x), then we could use a log link function with Poisson distribution. Using logit link function and binomial distribution gives us the logistic regression form.

4.5 Support vector machine

To understand SVM, let's take the example of a classification problem with two linearly separable classes. To separate the two classes, we could draw infinitely many decision boundaries. Intuitively it makes sense to choose the decision boundary which provides maximum separation between the two classes, i.e., all points in both classes should be at least at a distance of C from the decision boundary, and our optimization problem should maximize C. Let's say the two classes are $y = 1$ and $y = -1$. Then, we can write our constraints as,

$$\begin{aligned} \text{if } y_i = +1, w^T x_i + b &\geq C \\ \text{if } y_i = -1, w^T x_i + b &\leq -C \\ \text{Or, } y_i(w^T x_i + b) &\geq C \end{aligned} \tag{4.8}$$

The distance between the two lines $w^T x + b = C$ and $w^T x + b = -C$ is given by $d = \frac{2C}{\|w\|}^2$. So, our optimization problem can be written as,

$$\begin{aligned} &\max_{w,b} \frac{2C}{\|w\|^2} \\ \text{s.t. } & y_i(w^T x_i + b) \geq C \end{aligned} \tag{4.9}$$

Not all constraints are relevant. The ones that are relevant are called support vectors. Note that the optimization problem will not change if we put $C = 1$, as it can be part of the variables w, b. Also, the maximization can be changed to minimization as follows

$$\begin{aligned} &\min_{w,b} \|w\|^2 \\ \text{s.t. } & y_i(w^T x_i + b) \geq 1 \end{aligned} \tag{4.10}$$

4.6 Decision Trees

Decision trees is a supervised learning algorithm. One of the key aspects of decision trees is that they are highly interpretable, and can be highly flexible but due to this they are also prone to over-fitting (high variance). The name derives from the fact that like a tree, decision trees partition the feature space into two partitions at each node. At each node in the decision tree, we take a feature and divide the training data into two partitions. If the feature is numerical, then the two partitions could be $X_f \leq X_0$ and $X_f > X_0$. If the feature is categorical with values, $X_f = \{a, b, c\}$, then the two partitions can be $X_f = a$ and $X_f \neq a$. There are two concerns that we have while building a decision tree: a) which feature to use for partitioning, b) how should the feature be partitioned.

4.6.1 Regression trees

For regression problems, we choose the feature and location that minimizes the error. For example, let's say we partition the data with feature X_f at point $X_f = x_0$. Also, let's assume that there are m_l points in the left partition, and m_r points in the right partition. Then the squared error is given by,

$$SSE = \sum_{i=0}^{m_l} \left(y_i - \sum_{i=0}^{m_l} y_i \right)^2 + \sum_{j=0}^{m_r} \left(y_j - \sum_{j=0}^{m_r} y_j \right)^2 \tag{4.11}$$

If the number of features isn't large, the choice of feature and point of partition can be found quickly. We could define a stopping criteria for the decision tree, say, number of points in the leaf node is less than some threshold. But decision trees can grow a lot and hence overfit the training data. One strategy could be stop the tree growth is the SSE is lower than a threshold. But this could be short-sighted because a split with not much change in SSE could be followed by one with a large change in SSE. So, a better strategy is to let the tree grow and then prune it. One way to prune is the add a penalty to equation 4.11. Penalty term $= \alpha|T|$. Here, α is a tuning parameter and $|T|$ is the number of terminal nodes. For each sub-tree, it can be check if the split lowers the RSS after accounting for the penalty term.

4.6.2 Classification trees

4.6.3 Advantages and disadvantages of decision trees

1. Easy to explain. Even easier than regression. Mirrors human thinking
2. Can be displayed graphically
3. Decision trees can handle qualitative variables without the need to create dummy variables
4. Decision trees suffer from high variance
5. Lower predictive power compared to other algorithms

4.7 Random forests

As decision trees are prone to over-fitting (low-bias and high-variance), if we combine multiple decision trees and we can reduce the variance by averaging. In random forests, we used only a subset of the rows and columns. The first step is to build a bootstrapped dataset. If the original training dataset contains n points, then we build a sample of size n by random selection with replacement. Then at each node split, we use a subset of features to select from. One of the advantages with random forests is that they can be trained in parallel. With bootstrapping, on

4.8 Adaptive Boosting (adaboost)

Adaptive boosting algorithm has similarities with the random forest algorithm but differs in a few key aspects. In adaptive boosting, only weak learners (stumps) are built based on a single feature, i.e., one node and two leafs. Further, in adaboost, the stumps are built successively and the split is dependent on the error in the previous stump.

We start by giving an equal weight to all samples. If there are n samples, then each sample is given a weight of $w_t(i) = \frac{1}{n}$. Then, we choose the feature that results in the least classification error or best weighted gini index. The error in this step is $e_t = \sum_{i=1}^{n} w_t(i) * I\left(y(i) = \hat{h}_t(i)\right)$. The samples are then re-weighted for next step based on e_t. For the next step, $w_{t+1}(i) = w_t(i)e^{\alpha_t}$ if $y(i) \neq \hat{h}_t(i)$ else $w_{t+1}(i) = w_t(i)e^{-\alpha_t}$. The weights also need to be normalized before proceeding with the next step. Here, $\alpha_t = \frac{1}{2}\log\left(\frac{1-e_t}{e_t}\right)$. Using this form means that if a week learner results in a large error, then α_t is a large negative value (lower say in the final classification). If the error is small then the value is a large positive value (higher say in the final classification). The final classification is based on the weighted average of all weak learners, where weight is α_t.

References: StatQuest, Computational data analytics slides

4.9 Gradient boosting

Gradient boosting is a supervised learning technique that can solve both classification and regression problems. To understand it, let's take the example of a regression problem. Let's say, we have n samples, then in the absence of any features, the best guess for the response will be the average of samples. So, we start with the average, and calculate the pseudo residuals, $y_i - \bar{y}$. Then using the features, we we build a tree to predict the pseudo residuals. Let's say the predictions from this tree are y_i^t. Then the overall prediction is $\bar{y} + \lambda y_i^t$, where λ is the learning rate.

5. Hadoop

5.1 Hadoop architecture

A Hadoop cluster has a master and worker architecture. The master machine is called the namenode and the worker machines are called data nodes. The namenode maintains information on the directory tree of all files and the location of data blocks. This information is maintained in two files - `fsimage` and `editlog`. `fsimage` image is persisted on the disk and contains the state of the cluster from when the namenode was started. The `editlog` contains information on all the edits and is stored in memory. In production systems, the namenodes are rarely restarted and hence the `editlog` file could grow a lot. So, it needs to be merged with `fsimage`. The merging process is called checkpointing and it runs each hour. During checkpointing, the `editlog` is copied to the secondary namenode, and is merged with the `fsimage` file maintained on secondary name node. The merged `fsimage` file then replaces the copy on the namenode.

As the information on secondary namenode and namenode isn't in sync at all times, the secondary name node in itself is not a back-up solution for namenode failure. But copy of `editlog` could be maintained on another machine or NFS in real-time which can be used for running secondary namenode as primary namenode in case of failure.

5.1.1 Block size

On a hard disk, the lowest storage unit is called a sector which typically stores 512 bytes of information. Sectors can be grouped into blocks. An operating system assigns an address to each block. This limits the amount of disk size that an operating system can support. Initial versions of DOS OS and Mac could have only $2^{16} = 65$k addresses (allocation blocks). With a block size of 512 bytes this limited the amount of disk volume to 32MB (source). To support a 1GB hard disk, the block size will need to be 16KB. So, even if a file contained on 1 byte of data, it would occupy 16KB of space. Operating systems now support 2^{32} allocation blocks. Block size on Mac can be found using the command `$ diskutil info / | grep -i block`.

HDFS also has its own block size which is 64MB by default. If a file is 130MB, then it will be split into three blocks of 64MB, 64MB and 2MB. The last block does not occupy the full 64MB of space. Choosing a large block size helps Hadoop in minimizing the impact of seek time. Typically, seek time is 10ms and transfer rate is 100MB/s. To keep the seek time to 1% of the transfer time, the block size will need to be 100MB. Also, if the block size is too small, it will increase the size of metadata and managing this metadata will cause an overhead. If we choose a block size too large then we may not get the advantage of parallel processing with map tasks as too few of them may get created.

5.1.2 Write and read mechanism

When a client node wants to write a file to HDFS, the client node will split the file into blocks. Then it will ask namenode for the IP addresses of the data nodes where the blocks need to be written. The client node will then contact the first data node and check if it's available and ready. Then the first data node will do the same with second data node and so on. If a data node is unreachable, client node will ask for another data node from the namenode. Once all data nodes are ready, the block will be written to the first data node and then the second data node will copy it from the first data node. All blocks are written in parallel. Read process is simpler. The client node will ask for the file and name node will point to the IP addresses from where the blocks are to be read.

5.2 Map-reduce

The map-reduce is a framework for processing data in a distributed manner. The framework is composed of three parts: map, shuffle/sort and reduce. The map program takes a key-value pair as input and outputs a key-value pair using a map function. The shuffle-sort program shuffles the data across nodes, and groups the key-value pairs by keys. Data for each key is sent to one reducer as an iterable. The reduce program reduces the values for each key into a single value using the map function.

We can also have a combiner functions which acts like a reduce function for each mapper and can reduce the work required by reducer. But not all types of functions can have a combiner. If a and b are iterables, only reduce functions of the form $f\left(f(a), f(b)\right) = f(a, b)$ can have a combiner. For example, we can have a combiner for a max task but not for an average task.

Let's say we have a file size of 1TB, and the block-size is 128MB, then the number of mappers needed for maximum parallelization will be 8000. If a node has a 16 core CPU, then we can run about 10 mappers per node. Similarly, we can calculate the number of reducers. Number of reduces = factor $\times$ number of nodes $\times$ number of containers per node. Factor $= 0.95, 1.75$. If factor $= 0.95$, all reducers get launched when map tasks finish. With factor $= 1.75$, few reduces may finish early and take up the remaining reduce tasks.

5.3 YARN

YARN stands for yet-another-resource-manager. The core component of the YARN framework is called the Resource Manager which runs on the namenode. It is responsible for resource management and job scheduling. When a job is submitted by a client node, it is accepted by Resource Manager which generates an application ID. It then communicates with the Node Manager which reside on the data nodes. The Node Manager launches an App Master that requests for resources from the Resource Manager. Once resources are allocated, the App Master creates a container where the map-reduce job is executed.

5.4 Sqoop

Sqoop is used to transfer data between relational databases and Hadoop HDFS. The data can be imported from relational databases and stores as either text data or in binary format like Avro. Key sqoop commands are,

1. `sqoop import`: For importing a particular table
2. `sqoop import-all`: Importing all tables in a database. Each table must have a primary key
3. `sqoop export`: Exporting a table that has already been created on destination
4. `sqoop eval`: For evaluating an expression
5. `sqoop list-databases`: Listing all databases
6. `sqoop list-tables`: Listing all tables in a database

Reference for Sqoop commands: TutorialsPoint

6. Special topics

6.1 A/B testing

AB testing guide