Design and Development Application Review of the Corporate Standards (DDARCS)

Yiqiao Yin
Ph.D. Student

Abstract

Abstract for this Assignment. This assignment proposes the DDARCS document which is short for the Design and Development Application Review of the Corporate Standards, which is a signature document for this chapter of the course. The assignment is built on certain premises as this is left open from the assignment instruction. The company this document is proposed to serve is in healthcare sector specifically in radiography where there are machine learning scientists to assist radiography doctors such as pathologists or histologists to make a prognostic decision on detecting cancerous cells for patients. The company is assumed to be a global leading research and hospital group and facility teams in both hospitals as well as machine learning research. These premises are designed to set up certain environment for the document to serve better needs, because the instruction mentioned for ERP or EHR implementations. Without these premises, the document can be presented in vain and without stronger motivations.

The body of the document is composed of (1) Abstract, (2) Introduction, (3) Goal Statement, (4) Comparisons of the main features of different level of programming languages, (5) Recommendation for data management, (6) Coding standards and examples, and (7) Conclusions.

An important premise is that the company is backed by machine learning and neural network based algorithms that need to be deployed to the cloud and the mobile platform. Hence, this document introduces the basics of these machine learning topics in the Goal Statement to serve as a purpose. The comparisons of Go, Swift, and Julia (the main programming languages for desk, cloud, mobile applications) would need to be designed to serve for machine learning purposes.

Contents

Abstract of DDARCS

This document conducts an in-depth analysis of the current programming languages and development environments for desktop, cloud, mobile applications. The purpose is to create appropriate corporate standards for database and application design and development. The document starts with an Section 1: Introduction which is to cover the basic ideas of this document. The presmises in the cover page of this assignment stated the purpose of using machine learning backed modules to support frontend development. Hence, the next part Section 2: Goal Statement breaks apart down the argument and describe some basic machine learning topics, matrix form of the data frame, and the mathematical operation required for the support of the backend development. Section 3: Comparisons briefly described the syntax highlighting the Fibonacci series as an introductory example. Section 4 Recommendation for Data Management Platform summarizes the final recommendation of this document. As a bonus before the conclusion, Section 5: Coding Standards takes the basics from Section 3 and introduce code standards for all contenders from programming, data structure, function declarations, and mathematical operation perspectives. Section 6: Conclusions finishes the document and provides a quick summary of this document.

1 Introduction

The company provides integrated solutions to secure connectivity between discrepancies amongst different systems. The solutions at an enterprise level should be able to minimize or eliminate the manual or paper-driven processes amongst different departments. The health data and records need to be secured and transferred without friction or loss of information as efficiently as they possibly can. The results should provide our employees, patients, and clients the thorough electronic transfer of data from laboratories as well as clinical practices. The format and type of data should be flexible upon request and can be delivered on time without loss.

For advanced laboratories or central laboratories, the enterprise solutions need to aim for reduction of data entry cost into client's system. The health record and data management platform needs to increase efficiency throughout paperless centric approach. The entire database management system needs to be easily accessible for all approved personnel only and efficiently.

To deliver this idea, this investigative report lands on a proposal of some upgrades of the current engineering and database management system in regards to mobile application and data transfer. The document investigates a selective few famous computer programming languages and their pros and cons to reach a conclusion of which upgrade would be the most optimal in regards to handle the workflow and database management of the healthcare system. (Hoerbst and Ammenwerth, 2010; Evans, 2016; Cowie et al., 2017)

2 Goal Statement

To deliver a better system, online mobile service, and database management platform remotely, an optimal choice of a programming language. In this document, the investigation of three popular programming languages are conducted to understand the performance in regards of strengths and weaknesses.

First, the language Go was initially created at Google specifically for the desire to develop scalable models. The prior environment developers have been using stacks of languages composed of C++, Python, and Java applications which can raise a series of issues for QAQC (Quality Assessment and Quality Control). Often times the developers must collaborate in an enclosed environment where the system is tuned by one senior level developer and the same system is duplicated for the others. This way the collaboration can exist temporarily for the lifetime of the how long the team exists. However, in regards to all future references, it is questionable whether a new lead engineer or junior level engineer joining team can still follow up with the workflow. This design raises a series of questions and the stops developers from writing sophisticated on-going software products.

The language Swift is the second contender for the competition in this document. Unlike Go, Swift is backed by Apple. It is actually a successor to some of the conventional languages such as C and Objective-C and it even includes some of the low-level development primitives such as types, operators, arrays, and flow control. In addition, to properly operate the environment and allow any newcomers joining the team to be comfortable with the system, Swift has been known to be one of the simplest language to learn. It can even be taught using iPad (by downloading Sift Playgrounds) and then be professionally designed using Xcode.

A third language that is worth competing with Go and Swift is Julia. Scientific computing has been traditionally been well established and it is constantly being developed. In this sense, modern day computer programming requires the design and compiler to be made possible to be able to handle some of the day-to-day trade-off and to be able to debug in an unique environment such that the initial protocol can be designed efficiently to deploy some of the performance-intensive applications. This is an area where Julia programming shines and fits the profiles.

2.1 Machine Learning and Artificial Neural Networks Background

In the premise statement in the beginning of the DDARCS document, it is stated that the healthcare company that this CEO manages has pipelines heavily backed by machine learning problems. To understand the application field of machine learning component used in the backend of the cloud and mobile systems, it is important to introduce the fundamentals of the machine learning problems.

2.1.1 Overview

In the past, traditional machine learning strategies such as logistic regression, Support Vector Machine (SVM), Random Forests (RF) were restricted by their abilities to process natural data in their raw form. For decades, constructing a pattern recognition or machine system required careful engineering and considerable domain expertise to design a feature extractor that transformed the data LeCun et al. (2015).

Representation learning refers to a family of machine learning methodologies that allows a machine to be fed with raw data to automatically construct the representations needed for making predictions or classifications. Deep learning is a family of learning methods that fall under representations learning. Specifically, deep learning methods are obtained by consisting simple but non-linear modules that each transform the representation of the data form at a unique level into a global but higher and more abstract level. With the construction of enough such transformations, many complex functionalities can be discovered and learned LeCun et al. (2015). For the machine learning tasks that are aiming to make good prediction results, high layers of representation enhanced the way the input features can be processed and hence important rules can be constructed for discrimination and suppress irrelevant information. For example, images are defined in the form of an array of pixel values and the important features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image LeCun et al. (2015). The second layer typically detects higher level of abstractions such as edges, positioning, and so on. In some occasions, we even have a third layer which assemble all the patterns from the second layer into a much bigger combinations that are functions of familiar objects presented locally in the original raw image data. The key of deep learning of building these layers of features is that we need the machine to be able to learn the important information with the help of designed layers but not human engineers – the machines are learning from the data fed in and are using a generalized purpose in pattern recognition and making predictions.

Advancement of Deep Learning. Deep Learning is revolutionizing the modern day Artificial Intelligence and it has been for many years. The family of the entire deep learning based methods to make classifications has turned out to be extremely suitable for learning and representing the intricate structures in high-dimensional data and therefore is adapted in many domains of science, economics, biology, and so on. Famous applications can be found in image recognition (Krizhevsky et al., 2012; Farabet et al., 2012; Szegedy et al., 2015), speech recognition (Mikolov et al., 2011; Hinton et al., 2012; Sainath et al., 2013) and so on.

Supervised Learning. The most common form of machine learning task is supervised learning. This is the type of learning task that we have well-defined target variable. Whether if it is regression problem or classification, we know the variable we want to predict before we start designing the lab procedure and fitting the machine. This type of problems can have wide range of applications such as image classification, sentiment analysis, user recommendation system, and so on. In regression problem, the target variable we aim to predict is continuous. These variables can be temperature, stock price, sales number, next year's GDP per capita, and so on. In classification problem, the target variable we want to predict is discrete. These variables can be binary or can be in multiple different levels. Each level may represent a particular class. In a binary image classification task, a goal can be to classify cat images from dog images. In this case, the target variable would be dichotomous. We compute the objective function that measures error (or distance) between the ground truth and the predicted scores or values. We then need to modify the parameters (or weights) so that we can reduce the error computed by the objective function. The objective function can be seen as mapping the landscape of high-dimensional space that is computed based on the ground truth and the predicted values. This gives us an optimization problem. The argument in the optimization problem is to minimize the loss from the objective function by changing the parameters (or weights) in the model. A famous optimization technique is called gradient descent. The algorithm can have many iterations. At each iteration, the algorithm updates the weights of the pre-defined model according to the gradient (partial derivatives with respect to the parameters) of the loss. At present day, a deep learning model that completes these tasks can sometimes have hundreds of millions of trainable weights.

2.1.2 Linear Regression

In a supervised machine learning problem, the first type of task is regression problem. In a regression problem, we assume that the target variable $y \in \mathbb{R}$ has the following likelihood function

$$p(y|X) = \mathcal{N}(y|f(X, \sigma^2) \tag{1}$$

while $X \in \mathbb{R}^D$ are input variables or explanatory variables and $y \in \mathbb{R}$ is the target variable or the response variable. Another way to state the equation 1 is

$$y = f(X) + \epsilon, \tag{2}$$

while $\epsilon \sim \mathcal{N}(0, \sigma^2$ is iid Gaussian noise with mean 0 and variance σ^2. The goal is to search for a set of parameters (or weights) that generate the data well. To further dissect the equation 2, we can make the linearity assumption, which means that we rewrite the linear regression model as the following

$$p(y|X, \theta) = \mathcal{N}(y|X^T\Theta, \sigma^2) \Leftrightarrow y = X^T\Theta + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{3}$$

With the model set up as in equation 3, we can discuss in detail what kind of parameter set works "well" in this model. For now, we assume that the noise variance σ^2 is known.

Consider a model that takes the form of equation 2 and assume in training set we are given $\{(X_{1,\cdot}, y_1), ..., (X_{N,\cdot}, y_n)\}$. Notice that y_i and y_j are conditionally independent given their respective inputs $X_{i,\cdot}, X_{j,\cdot}$ so that the likelihood factorizes according to

$$\begin{array}{rcl} p(Y|X, \Theta) & = & p(y_1, ..., y_N|X_1, ..., X_N, \Theta) \\ & = & \prod_{i=1}^N p(y_i|X_i, \Theta) \\ & = & \prod_{i=1}^N \mathcal{N}(y_i|X_i^T\Theta, \sigma^2) \end{array} \tag{4}$$

where we define $X = \{X_1, ..., X_N\}$ and $Y = \{y_1, ..., y_N\}$ as the sets of training inputs and corresponding targets, respectively.

A widely used procedure to find the desired parameters $\hat{\Theta}_{\text{ML}}$ is maximum likelihood estimation, where we find the optimal parameters $\hat{\Theta}_{\text{ML}}$ that maximize the likelihood which is written in the last line of equation 4. In other words, we want to maximize the predictive distribution of the training data given the model parameters, which means we obtain the parameters as

$$\hat{\Theta}_{\text{ML}} = \arg\min_{\Theta} p(Y|X, \Theta) \tag{5}$$

To find the desired parameters , we typically perform gradient ascent (or gradient descent on the negative likelihood). In this case, this linear model has a closed-form solution which makes iterative gradient descent unnecessary. Mathematically, it is much more efficient to use log-likelihood function. Hence, to find the optimal parameters $\hat{\Theta}_{\text{ML}}$, we minimize the negative log-likelihood

$$-\log p(Y|X, \Theta) = -\log \prod_{i=1}^N p(y_i|X_i, \Theta) = -\sum_{i=1}^N \log p(y_i|X_i, \Theta) \tag{6}$$

In the linear regression model 3, the likelihood is Gaussian, so we arrive at

$$\log p(y_i|X_i, \Theta) = -\frac{1}{2\sigma^2}(y_i - X_i^T\Theta)^2 + \text{constant} \tag{7}$$

where the constant term includes all terms independent of the parameter Θ. Thus, we obtain

$$\begin{array}{rcl} \mathcal{L}(\Theta) & = & \frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - X_i^T\Theta)^2 \\ & = & \frac{1}{2\sigma^2}(Y - X\Theta)^T(Y - X\Theta) \\ & = & \frac{1}{2\sigma^2}||Y - X\Theta||^2 \end{array} \tag{8}$$

where X is the design matrix and $X \in \mathbb{R}^{N \times D}$ and $Y \in \mathbb{R}^N$. In practice, we can think of the data matrix X to have N samples in training set and D parameters so it is a matrix with dimension $N \times D$ while Y is the response vector so Y is a vector of length N. The loss function in equation 1 is also known as sum of squared between the training set response variable Y and the prediction $X^T\Theta$. The final step is to compute the gradient of $\mathcal{L}$ with respect to the parameters

$$\begin{array}{rcl} \frac{\partial}{\partial\theta}\mathcal{N} & = & \frac{\partial}{\partial\theta}(\frac{1}{2\sigma^2}(Y - X\Theta)^T(Y - X\Theta)) \\ & = & \frac{1}{2\sigma^2}\frac{\partial}{\partial\theta}(Y^TY - 2Y^TX\Theta + \Theta^TX^TX\Theta) \\ & = & \frac{1}{\sigma^2}(-Y^TX + \Theta^TX^TX) \end{array} \tag{9}$$

which then we can set to zero and solve for

$$\begin{array}{rcl} \text{set: } \frac{\partial}{\partial\Theta}\mathcal{L} = \vec{0} & \Leftrightarrow & \hat{\Theta}^TX^TX = Y^TX \\ & \Leftrightarrow & \hat{\Theta}^T = Y^TX(X^TX)^{-1} \\ & \Leftrightarrow & \hat{\Theta} = (X^TX)^{-1}X^TY \end{array} \tag{10}$$

which is the closed-form solution for a linear regression model.

2.1.3 Logistic Regression

The logistic regression model is another crucial stepping stone in statistics and machine learning when the objective is to investigate the posterior probabilities of the object with K classes by implementing linear functions in x while x refers to the explanatory variables assuming the probabilities of all K classes sum to one and remain in $[0, 1]$. The model has the following form

$$\begin{array}{rcl} \log\frac{\mathbb{P}(G=1|X=x)}{\mathbb{P}(G=K|X=x)} & = & \beta_{10} + \beta_1^Tx \\ \log\frac{\mathbb{P}(G=2|X=x)}{\mathbb{P}(G=K|X=x)} & = & \beta_{20} + \beta_2^Tx \\ & \vdots & \\ \log\frac{\mathbb{P}(G=K-1|X=x)}{\mathbb{P}(G=K|X=x)} & = & \beta_{(K-1)0} + \beta_{K-1}^Tx \end{array} \tag{11}$$

The model describes the $k-1$ log-odds or logit transformations providing the trainable weights (or parameters) $\{\beta_{\cdot,0}, \beta_{\cdot}\}$. For each k, we can simply write

$$\begin{array}{rcl} \mathbb{P}(G = k|X = x) & = & \frac{\exp(\beta_{k0}+\beta_k^Tx)}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^Tx)}, k = 1, ..., K-1 \\ \mathbb{P}(G = k|X = x) & = & \frac{1}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^Tx)} \end{array} \tag{12}$$

and they clearly sum to one. To denote the entire parameter set $\theta = \{\beta_{10}, \beta_1^T, ..., \beta_{(K-1)0}, \beta_{K-1}^T\}$, we denote the probabilities $\mathbb{P}(G = k|X = x) = p_k(x;\theta)$.

In the following, it is important to discuss when $K = 2$. In this case, the model is simple and straightforward for interpretation. Since there are only a single linear function, it is widely accepted to use applications in bio-statistics where we face two classes or binary responses. Suppose we have an experiment where there are two possible outcomes: either success or failure, either diseased or healthy, etc.. Suppose success happens with probability p. Then failure happens with probability $1-p$. A random variable that takes value 1 in case of success and value 0 in case of failure is called a Bernoulli random variable. Formally, a random variable Y is a Bernoulli random variable if it has support $R_Y = \{0, 1\}$ and with a probability p it has probability mass function to be

$$\mathbb{P}_Y(y) = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{if } y = 0 \\ 0 & \text{elsewhere} \end{cases} \tag{13}$$

Based on the above probability mass function, we can find the expectation of Y to be

$$\begin{array}{rcl} \mathbb{E}(Y) & = & \sum_{y \in R_Y} y\mathbb{P}_Y(y) \\ & = & 1 \cdot \mathbb{P}_Y(1) + 0 \cdot \mathbb{P}_Y(0) \\ & = & 1 \cdot p + 0 \cdot (1-p) \\ & = & p \end{array} \tag{14}$$

and in order to find the variance of Y, we first find

$$\begin{array}{rcl} \mathbb{E}(Y^2) & = & \sum_{y \in R_Y} y^2\mathbb{P}_Y(y) \\ & = & 1^2 \cdot \mathbb{P}_Y(1) + 0^2 \cdot \mathbb{P}_Y(0) \\ & = & 1 \cdot p + 0 \cdot (1-p) \\ & = & p \\ \text{which gives us } \Rightarrow \text{var}(Y) & = & \mathbb{E}(Y^2) - \mathbb{E}(Y)^2 \\ & = & p - p^2 \\ & = & p(1-p) \end{array} \tag{15}$$

In the next component of this section, we discuss the procedure of searching for the optimal parameters for $\beta = \{\beta_{10}, \beta_1\}$ in the equation 11 when $K = 2$ (because we assume we are treating a binary-class classification problem). In order to motivate the search process, we review the procedure of finding the maximum likelihood estimator for Bernoulli random variables. Suppose we have $Y_1, ..., Y_n \sim_{\text{iid}}$ Bernoulli(p). The goal is to find the maximum likelihood estimator (MLE) for p. We first find the log-likelihood function.. Then we set the first order partial derivatives to zero and solve for the optimal parameter. It is essential to start by writing down the likelihood function

$$L(p) = \prod_{i=1}^{n} p^{y_i}(1-p)^{(1-y_i)} \tag{16}$$

and to make the math more efficient we take logarithm, so we write

$$l(p) = \log p \sum_{i=1}^{n} y_i + \log(1-p) \sum_{i=1}^{n}(1-y_i) \tag{17}$$

To find the most optimal weight, we take the first order partial derivatives (with respective to p) and we set the equation to zero.

$$\begin{array}{rcl} \frac{\partial}{\partial p} l(p) & = & \frac{1}{p}\sum_{i=1}^{n} y_i - \frac{1}{1-p}\sum_{i=1}^{n}(1-y_i) \overset{\text{set}}{=} 0 \\ \Rightarrow \sum_{i=1}^{n} y_i - p\sum_{i=1}^{n} y_i & = & p\sum_{i=1}^{n}(1-y_i) \\ p & = & \frac{1}{n}\sum_{i=1}^{n} y_i \end{array} \tag{18}$$

Moreover, we can always check the second order partial derivative to ensure that the optimal solution is a global optimal.

$$\frac{\partial^2}{\partial p^2} l(p) = \frac{-1}{p^2}\sum_{i=1}^{n} y_i - \frac{1}{(1-p)^2}\sum_{i=1}^{n}(1-y_i) \tag{19}$$

In the logistic regression models, it is common procedure to search for the best fit using maximum likelihood. Since $\mathbb{P}(G|X)$ completely specifies the conditional distribution, the multinomial distribution is appropriate in general scenarios for the equation 11. In other words, we may write the log-likelihood for N observations to be

$$l(\theta) = \sum_{i=1}^{N} \log \mathbb{P}_{g_i}(x_i; \theta) \tag{20}$$

where $\mathbb{P}(k(x_i; \theta) = \mathbb{P}(G = k|X = x_i; \theta)$.

For simplicity, denote two-class g_i with the response y_i which means that $y_i = 1$ when $g_i = 1$ and $y_i = 0$ when $g_i = 2$. Since the probabilities for these two classes are complete of each other, we further assume $\mathbb{P}(G = 1|X = x_i; \theta) = 1 - \mathbb{P}(G = 2|X = x_i; \theta)$. With the motivation in equations 16 17 18, we can write out the log-likelihood

$$\begin{aligned} l(\beta) &= \textstyle\sum_{i=1}^{N} \{y_i \log \mathbb{P}(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta))\} \\ &= \textstyle\sum_{i=1}^{N} \{y_i \beta^T x_i - \log(1 + e^{\beta^T x_i})\} \end{aligned}$$

and here we let $\beta = \{\beta_{10}, \beta_1\}$ following the notation in equation 11 while we assume that the vector of input variables x_i to include the constant term to accommodate the implementation of intercept (or bias term) in the model. In information theory, the equation 2.1.3 can also be referred to as the cross-entropy error. In two-class classification problems, we also call the equation 2.1.3 the binary-cross-entropy. To maximize the log-likelihood, we set the partial derivatives to zero. This becomes

$$\frac{\partial}{\partial \beta} l(\beta) = \sum_{i=1}^{N} x_i (y_i - p(x_i; \beta)) = 0, \tag{21}$$

and we can use the Newton-Raphson algorithm, which requires the second-derivative (or Hessian matrix)

$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} = - \sum_{i=1}^{N} x_i x_i^T p(x_I; \beta)(1 - p(x_i; \beta)). \tag{22}$$

Starting with β^{old}, a single Newton update is

$$\beta^{\text{new}} = \beta^{\text{old}} - \big(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T}\big)^{-1} \frac{\partial l(\beta)}{\partial \beta} \tag{23}$$

where the derivatives are evaluated at β^{old}. To make our notations simpler to use, write in matrix notation. Denote **y** the vector of y_i values, **X** the $N \times (p+1)$ matrix of x_i values, **p** the vector of fitted probabilities with the ith element $p(x_i; \beta^{\text{old}})$ and **W** a $N \times N$ diagonal matrix of weights with ith diagonal element $p(x_i; \beta^{\text{old}})(1 - p(x_i; \beta^{\text{old}}))$. Then we have

$$\begin{aligned} \tfrac{\partial l(\beta)}{\partial \beta} &= \mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\ \tfrac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} &= -\mathbf{X}^T \mathbf{W} \mathbf{X} \end{aligned} \tag{24}$$

The Newton step is thurs

$$\begin{aligned} \beta^{\textbf{new}} &= \beta^{\text{old}} + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{p}) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}(\mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z} \end{aligned} \tag{25}$$

In the above set of equations, notice that we rewrite the Newton step as weighted least squares, the response is

$$\mathbf{z} = \mathbf{X}\beta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p}), \tag{26}$$

which is also known as the adjusted response. These equations can be solved iteratively using an algorithm called iteratively reweighted least squares (or IRLS), with each iteration being defined as

$$\beta^{\text{new}} \leftarrow \arg\min_{\beta} (\mathbf{z} - \mathbf{X}\beta)^T \mathbf{W} (\mathbf{z} - \mathbf{X}\beta) \tag{27}$$

To initialize this algorithm, it is common practice to start with $\beta = 0$. However, the convergence cannot be guaranteed. Due to the nature of concavity of the log-likelihood function, the algorithm should converge theoretically. But practice with real world application is more of a state of art.

2.1.4 Neural Networks

The term neural network has evolved to encompass a large class of models and learning methods. In this paper, we describe the most widely used "vanilla" neural network. This can also be called the single hidden layer back-propagation network, or single layer perceptron.

A neural network is a two-stage regression or classification model. We present the most basic design of a neural network in Figure 1. In Figure 1, there is a diagram on the left consists of input features such as $\{x_1, x_2, ...\}$. This is the architecture of a basic neuron. The diagram on the right consists of many of the neurons in a hidden layer of which we can use to build many deep neural networks (or sometimes called deep Artificial Neural Networks). This network architecture applies to both regression or classification. For regression, there is one output and we need to amend the output to just O instead of $\{O_1, O_2\}$ as shown in Figure 1. For classification such as a two-class classification problem, we can directly use the architecture presented in Figure 1. We can even produce pictures at the output layer (this leads to variational-autoencoder which will cover in later chapters).

To generalize this architecture, we denote the input features of each neuron to be $\{x_1, ..., x_p\}$ as illustrated on the left diagram of Figure 1. These input features are covariates (or explanatory variables) when the neuron is in the first hidden layer. These input features are output from an activation function if these input features are fed into a neuron in the middle of the deep learning architecture (this happens if there are more than one hidden layer). In each neuron, we have the symbol $\Sigma|\sigma$ to illustrate that it has a linear combination and a non-linear operation.

Now we formally present the mathematical formulation to construct one single neuron. Suppose we have input features $\{x_1, ..., x_p\}$ and each of the input feature carries a weight (also known as parameter) $\{w_1, ..., w_p\}$. In addition, we have a one vector that represents the bias term with weight w_0. This bias term can be set to zero if desired. The linear combination takes the form as the following

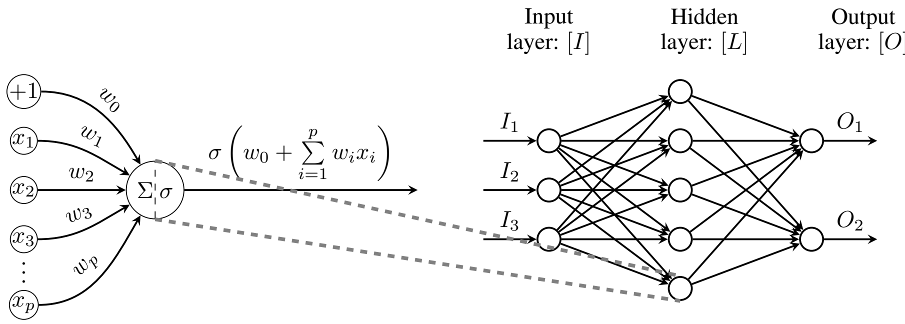
$$\text{linear combination: } w_0 + \sum_{j=1}^{p} w_j x_j \tag{28}$$

and the result of this linear combination is then fed into a non-linear function. This non-linear function is also called an activation function. The reason is because the common choice of activation function can be ReLU, i.e. $\text{ReLU}(z) = \max(z, 0)$. The nature of the ReLU activation function acts as a switch to turn the information on and off. Mathematically, it is a link function to assist the output to generate positive numerical value for the purpose of classifications. In other words, the choice of activation function is dependent on data set. Next, we feed the result of the linear combination into a non-linear function. Two famous activation functions are ReLU and sigmoid. The ReLU activation function is common used in between layers while the sigmoid activation function is commonly used in the end of a neural network architecture. We formally write them below

$$\text{output for ReLU activation} = \text{ReLU}(w_0 + \sum_{j=1}^{p} w_j x_j, 0) \tag{29}$$

$$\text{output for sigmoid activation} = \sigma(w_0 + \sum_{j=1}^{p} w_j x_j) = \frac{1}{1 + \exp(-(w_0 + \sum_{j=1}^{p} w_j x_j))} \tag{30}$$

Figure 1: **Basic Structure of Artificial Neural Network.** This figure represents the internal structure of a neuron in an Artificial Neural Network (ANN) with one hidden layer.



3 Comparisons

To handle large-scale programming projects, object-oriented programming is the most optimal approach though not the only one. For the purpose of this document, it is worth to compare Go, Swift, and Julia in how the codes are stacked and compiled. Go attempts to reconsider the whole pipeline by Duck typing the syntax language. Swift uses object-oriented approach as a base and it updates its pipeline using other interface and class extensions. Julia has its own mind and uses multiple dispatch (a programming feature that the code or functions can be dynamically dispatched based on the dynamic type or run-time).

3.1 Go

In Go language, the duck typing is, without a doubt, one of the basic ways to solve programming problems. It is simple and very straightforward. In Go, there is no specific requirements of stating the specific element implementations with an interface. The language automatically does depending on the listed methods. To develop ERP or EHR system, Go can definitely be one of the go-to languages. There are many existing libraries available to use that can allow users and developers to speed up the environment. A famous library is Doppler that allows to develop secure data storage system across different platforms, environments, servers, and across different teams. Another famous library is WorkOS, which allows the developers to use just a few lines of code to start selling products and services to enterprise customers.

3.2 Swift

The second contender, Swift, has commonly been known for the easier choice of developing online database management system. It is also a trending language for newcomers due to its easy syntax and many existing languages. From today's demand, users tend to prefer to have mobile applications integrated with the latest technologies and techniques. Swift is a relatively young language comparing with C++ and Java, however, there is definitely a persuasive argument to use Swift. Its newborn status could actually provide key features for Swift to be easily integrated with other libraries required to handle large-scale programming needs. Due to its clean syntax, Swift actually allows users, developers, and newcomers clean environment to write easy syntax which makes it easier to read, write, edit, and collaborate with other code. Swift is also backed by

Apple, which has the largest amount of IOS apps available across the board and has the largest amount of online users. This allows the modern day healthcare system to be able to develop mobile applications that can easily be reviewed or accessed by the patients. The patients, with a move of a finger, can log in to their profile or accounts and edit account information, check-in to a doctor's appointments, pay the co-pay, review X-ray scans and do many other things that are desirable to make the their lives easier.

3.3 Julia

Julia acts as the third contender and a compiled languages using interpreters like that of Python and R. The processing of the script of Julia can be less intuitive and slow. However, the language, if truly well written, can function as efficient as C. Another perspective is that Julia programming using a lot of syntax from Python, Ruby, Perl, and so on, which makes the programming language familiar for some newcomers. The base library of Julia is actually written by Julia itself, which includes most of the primitive compositions. The richness of the language is tested to be constructive and easy to elaborate with different objects that can usually be used to declare different components of the programming project.

3.4 Examples: Fibonacci Series

The famous Fibonacci numbers have been the most intriguing phenomenon for all mathematicians and it is a good building block to learn the fundamentals of a programming language.

Formally, the Fibonacci series is referring to a series of numbers, i.e. S_n, where n is the running index of this number. The sequence starts from 0 is connected with 1. From there, every number afterwards is always the sum of the previous two numbers. In other words, we can formally write the following

$$F_n = F_{n-1} + F_{n-2} \tag{31}$$

where $n \geq 2$. The document starts by introducing the basic syntax of Go, Swift, and Julia using the famous Fibonacci Series.

First, the Fibonacci Series written in Go is presented in the following. Then we present the same algorithm written in Swift. Last, the Julia version is presented for comparison. The time complexity is $\mathcal{O}(2^n)$ and the space complexity is $\mathcal{O}(2^n)$.

```
func this_particular_fibo(n int) int {
    if n <= 1 {
        return n
    }
    return this_particular_fibo(n-1) + this_particular_fibo(n-2)
}
```

```
func this_particular_fibo(_ n: Int) -> Int {
    guard n > 1 else { return n }
    return this_particular_fibo(n-1) + this_particular_fibo(n-2)
}
```

```
function this_particular_fibo(n::Int)
            n<0 && error("n must be non negative")
            n==0 && return 0
            n==1 && return 1
```

```
            this_particular_fibo(n-1) + this_particular_fibo(n-2)
            end
```

4 Recommendation for Data Management Platforms

This document has covered some major differences amongst Go, Swift, and Julia as well as introducing some of the basic syntax. The document also has supporting topics of machine learning and basic Artificial Intelligence. The section we combine all reasoning together to make a finalized recommendation of programming language based on all the above in-depth analysis of programming languages. The final recommendation is Swift.

5 Coding Standards

5.1 Examples

This portion of the document starts with a simple task in each of the three programming languages: Go, Swift, and Julia. Then we show examples of in-house development that can directly be used in ERP and EHR solutions.

5.1.1 Go

In Go, it is easier practice to start with a working module. This approach allows each building block to be modular and it is simpler to put the packages together in discrete and useful way. In a command prompt, a programmer should first start with setting the desired directory:

```
cd SOME_PATH
```

In this directory (which is supposed to be empty because it is a new project), clean directory is recommended so a starter directory is created as a subfolder.

```
mkdir starting
cd starting
```

Next, in order to start a module, the command "go mod init" is recommended. Hence, the followning code can be a good starter.

```
go mod init some_examples.com/starting
```

This assumes that the software is written and the code is packed into a file (one can open this in a VSCode window)

```
package greetings

import "fmt"

// Hello returns a greeting for the named person.
func Hello(name string) string {
    // Return a greeting that embeds the name in a message.
    message := fmt.Sprintf("Hi, %v. Welcome!", name)
    return message
}
```

In the above code, it is important to understand the the structure of the syntax instead of the syntax itself. The function takes a name parameter. The name parameter is a string and it serves as an argument. Then a return type is specified which also happens to be a string.

To a simple reverse function, a ".Reverse" function can be applied and the sample code can be seen below.

```
package main

import (
    "fmt"

    "golang.org/x/example/stringutil"
)

func main() {
    fmt.Println(stringutil.Reverse("Hello"))
}
```

To set up database system to ensure client-side can have access of the database without issue, a module needs to be developed and the common language to handle database platform is SQL or MySQL. At the command line, it is important to enter

```
mysql -u root -p
Enter password: ENTER_YOUR_PASSWORD
mysql>
```

and this is when the user password is required. Next, a database can be created

```
mysql> create database recordings;
```

and inside the file the following sample SQl code is provided as an example

```
DROP TABLE IF EXISTS viproom;
CREATE TABLE viproom (
  id INT AUTO_INCREMENT NOT NULL,
  title VARCHAR(128) NOT NULL,
  age DECIMAL(5,2) NOT NULL,
  PRIMARY KEY (‘id‘)
);

INSERT INTO viproom
  (title, artist, age)
VALUES
  ('Doctor', 'John Coltrane', 56),
  ('Nurse', 'Jenny Smith', 63),
  ('Surgeon', 'Gerry Neil', 65),
  ('Patient', 'Autumn Vaughan', 34);
```

which does a series of actions. The code first delete the table called the "viproom". The code then create this table with three primary keys called: id, title, and age.

Once database is setup, Golang offers libraries to support the development of machine learning based backend environment. A famous library is GoLearn and the environment functions similar to that of Sci-kit Learn, a common python based machine learning model. The following is taken from the sample github repo from here

```
package main

import (
        "fmt"

        "github.com/sjwhitworth/golearn/base"
        "github.com/sjwhitworth/golearn/evaluation"
        "github.com/sjwhitworth/golearn/knn"
)

func main() {
    // load in the desired data frame with columns
    // columns need to be stored
    // consider the instance and ensure the data frame
    // come from a proper structure from ``df'' in R or ``pandas''
        ↪  in Python
        rawData, err := base.ParseCSVToInstances("datasets/iris.
            ↪ csv", true)
        if err != nil {
                panic(err)
        }

        // Print a pleasant summary of your data.
        fmt.Println(rawData)

        //Initialises a new KNN classifier
        cls := knn.NewKnnClassifier("euclidean", "linear", 2)

        //Do a training-test split
        trainData, testData := base.InstancesTrainTestSplit(
            ↪ rawData, 0.50)
        cls.Fit(trainData)

        //Calculates the Euclidean distance and returns the most
            ↪ popular label
        predictions, err := cls.Predict(testData)
        if err != nil {
                panic(err)
        }

        // Prints precision/recall metrics
        confusionMat, err := evaluation.GetConfusionMatrix(
            ↪ testData, predictions)
        if err != nil {
                panic(fmt.Sprintf("Unable to get confusion matrix:
                    ↪ %s", err.Error()))
        }
        fmt.Println(evaluation.GetSummary(confusionMat))
}
```

5.1.2 Swift

Next, Swift programming language also offers similar contained environment as Go for online database management and machine learning backend pipelines. The Swift programming language is highly optimized to handle large-scale machine learning tasks. In fact, it is one of the most important task at the birth of the language. Swift works with tensorflow and one of its significant achievements was to include language-based differentiable programming. This allows us to have optimized condition to run highly complex algorithms such as that is discussed earlier in this document, see equation 10. In this subsection, sample tutorials and code standards for Swift is presented. One can starts with a simple "hello world".

```
print("Hello, world!")
// Prints "Hello, world!"
```

As discussed above, since Swift comes from objective-C, it is not surprising that the syntax looks like that of the C. Unlike C, this is a complete program in Swift and there is no need to define global variable and set the value type as string to be able to define a variable and process the data. In case of handling larger scale project and variables do need to be defined, one can follow the syntax below

```
let some_label = "Yiqiao Yin went to college in "
let some_width = 2010
let this_output = some_label + String(some_width)
# Yiqiao Yin went to college in 2010
```

Loops are common operation in favor if the desired task is dynamically related. Similar to programming languages such as Python or R, Swift code also supports for loop. The example is below

```
let examGrades = [75, 43, 103, 87, 12]
var this_grade_ = 0
for score in examGrades {
    if score > 50 {
        this_grade_ += 3
    } else {
        this_grade_ += 1
    }
}
print(this_grade_)
// Prints "11"
```

The most important modular component is the capabilities to design large-scale projects in functions. The functions of Swift is very easy to understand. There is some sample syntax below

```
func computeThisNumber(scores: [Int]) -> (min: Int, max: Int, sum:
    ↪ Int) {
    var min = scores[0]
    var max = scores[0]
    var sum = 0

    for score in scores {
        if score > max {
            max = score
```

```
        } else if score < min {
            min = score
        }
        sum += score
    }

    return (min, max, sum)
}
let statistics = computeThisNumber(scores: [5, 3, 100, 3, 9])
print(statistics.sum)
// Prints "120"
print(statistics.2)
// Prints "120"
```

In this function, the data types need to be specified inside the parenthesis. The scores need to be an integer and that is an input argument. The output are three integers and they also need to be specified. The variables are locally defined globally inside of a function. This sounds mouthful. The function itself is a local component of a large-scale project. The function is defined locally. Inside of a function, there is a for loop. In case of the needs for building machine learning backed algorithms to support the system for clients and patients, functions are desired to facilitate this setup.

5.1.3 Julia

For high performance multiplaform supercomputing, Julia is another preferred language. Though Julia might not be as optimal as Go and Swift when it comes to handle large-scale online or mobile platform specifically when it comes to designing a full scale of ERP or EHR system for a healthcare company, it is still worth the role in this document for junior level developers who are not directly writing code into production but are trying to write small projects for concept arts.

Julia is dynamically typed and, just like Python, it has modules and functions. Though Julia is dynamic, it does not lack of the advantage of static typing. As a matter of fact, many Julia scripts without specific methods attached to them. Let us start with some basic declaration of variables.

```
my_name = "Yiqiao"
my_age = 30
my_city = "New York"
```

Thus, the declarations are basic commands that are very much like that of Python other object-oriented language. The declared variables can be printed out.

```
print(my_name)
# Yiqiao
```

The next component is the function declaration. As a comparison to Go and Swift, the function component is quite straightforward.

```
function my_addition(x)
    # perform an addition operation
    return x + 5
end
```

Unlike Python, the declaration of Julia needs end on "end". Correct indentation also needs to be specified correctly like Python, otherwise interpreter will fail to execute the code. To handle and design online and mobile service to assist ERP or EHR platforms, it is important to introduce how Julia handles data structures. The following code presents a small task to handle some toy data and conduct some basic mathematical operations on them. A library needs to be called the syntax "using". The data x, y, xx, yy are declared. The data types can even be higher dimensional. Hence, we design the following experiment to showcase how data is handled and used in a mathematical operation.

```
using JLD

x = collect(-10:0.1:10)
y = collect(-10:0.1:10)
xx = reshape([xi for xi in x for yj in y], length(y), length(x))
yy = reshape([yj for xi in x for yj in y], length(y), length(x))

z = cos.(xx.+ yy.^2)
some_output = Dict("x" => x, "y" => y, "z" => z)
save("some_output.jld", some_output)
```

6 Conclusion

Design and Development Application Review of the Corporate Standards (DDARCS) is a document that provides in-depth analysis of three popular programming languages for developing cloud and mobile system for ERP and EHR system. We recommend to use Swift for the development and sample codes are provided to demonstrate showcase of the usage of the proposed programming language.

References

Cowie, M. R., Blomster, J. I., Curtis, L. H., Duclaux, S., Ford, I., Fritz, F., Goldman, S., Janmohamed, S., Kreuzer, J., Leenay, M., et al. (2017). Electronic health records to facilitate clinical research. *Clinical Research in Cardiology*, 106(1):1–9.

Evans, R. S. (2016). Electronic health records: then, now, and in the future. *Yearbook of medical informatics*, 25(S 01):S48–S61.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2012). Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

Hoerbst, A. and Ammenwerth, E. (2010). Electronic health records. *Methods of information in medicine*, 49(04):320–336.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Mikolov, T., Deoras, A., Povey, D., Burget, L., and Cernocky, J. (2011). Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 196–201. IEEE.

Sainath, T., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. (2013). Acoustics, speech and signal processing (icassp). In *2013 IEEE International Conference on. IEEE*, pages 8614–8618.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.