Spam and Phishing

CSIT375 AI for Cybersecurity

Dr Wei Zong

SCIT University of Wollongong

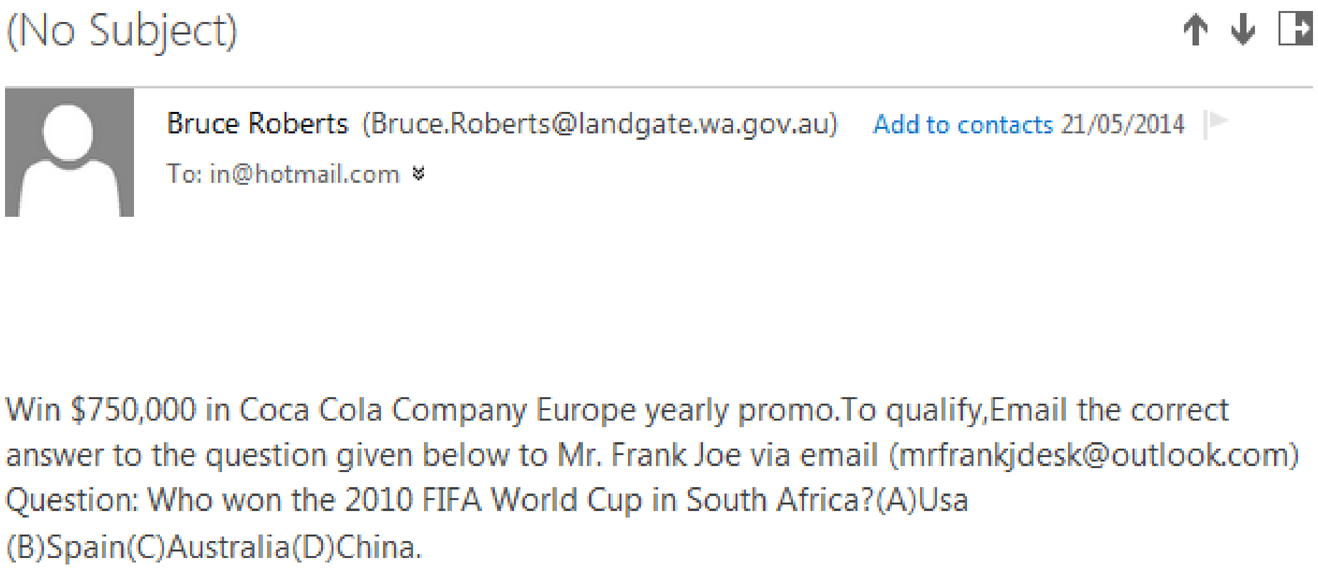
Disclaimer: The presentation materials come from various sources. For further information, check the references section

Outline

- What is Spam?
- Examples of Email Fraud
- Spam Filtering Techniques
- Pre-processing Text in Email Messages
- Spam detection with neural networks
  - Model representation

What is Spam

- Spam: Unsolicited bulk email or message
- Emails that involves sending identical or nearly identical messages to thousands (or millions) of recipients



What are the problems?

- With a tiny investment, a spammer can send over 100,000 bulk emails per hour
- Waste users' time, may cause financial loss
  - Study shows: an average person spends 28% of the workday (≈ 670 hrs/year) reading and responding to emails
  - only 38% of the emails on an average are relevant and important
- Junk mails waste storage and transmission bandwidth
- Might include malware as executable files
- Spam is a problem because the cost is forced onto the recipient

Statistics

- Spam accounts for 45% of all emails sent
- About 14.5 billion spam emails are sent every single day
- Spam costs $20.5 billion yearly (reduced network bandwidth, storage capacity)
- Spammers receive on average 1 click for every 12 million emails sent
  - Even with this response, spammers earn millions of dollars yearly
- 80% of all spam is sent by the same 100 spammers

Image credit: https://www.propellercrm.com/blog/email-spam-statistics

Phishing

- **Phishing:** attacks where a victim is lured to a fake web, and is deceived into disclosing personal data or credentials
- **Phishing URLs** seem like legitimate URLs, and redirect the users to phishing web pages, which mimic the look and feel of their target websites
  - ***URL (Uniform Resources Locator)*** is a web address that specifies the location of the webpage on a computer network
  - A typical URL http://www.example.com/index.html consists of several components:
    - Protocol type = http
    - Domain name = www.example.com
    - File name = index.html

Phishing

- Phishing emails are a more serious threat than spam emails, because they aim to steal users' private information, such as bank accounts, passwords, SSNs

Dropbox Phishing

- Attackers create fake sign-in pages for Dropbox as a part of credential harvesting
- They then use the stolen credentials to log in to legitimate sites and steal user data

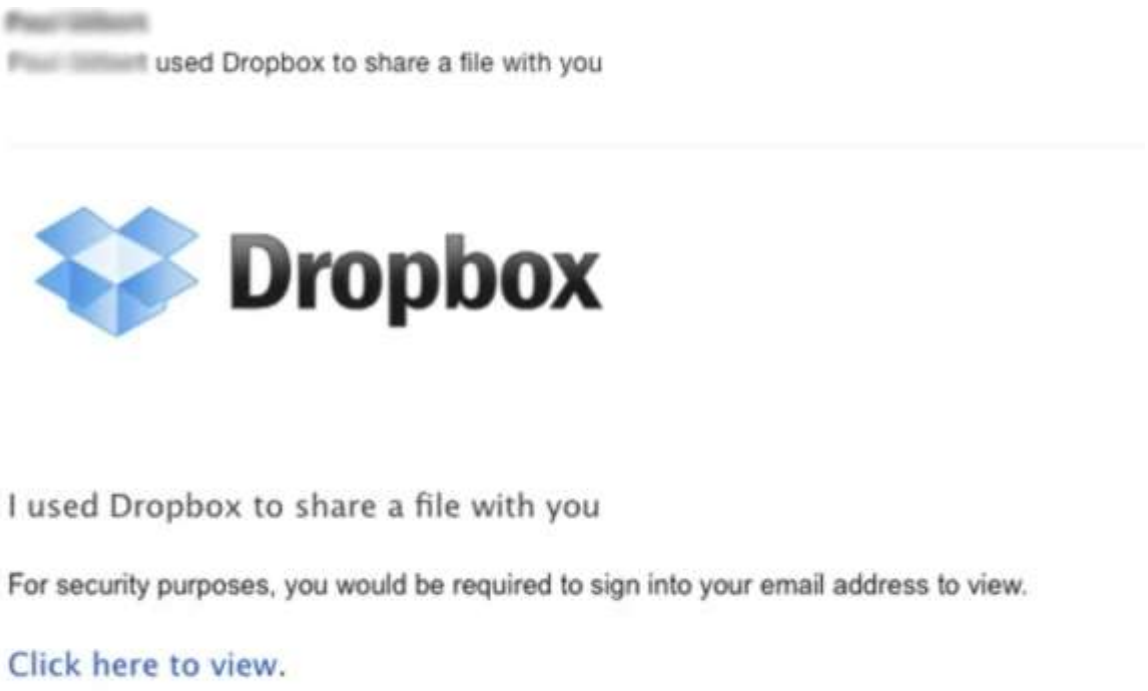
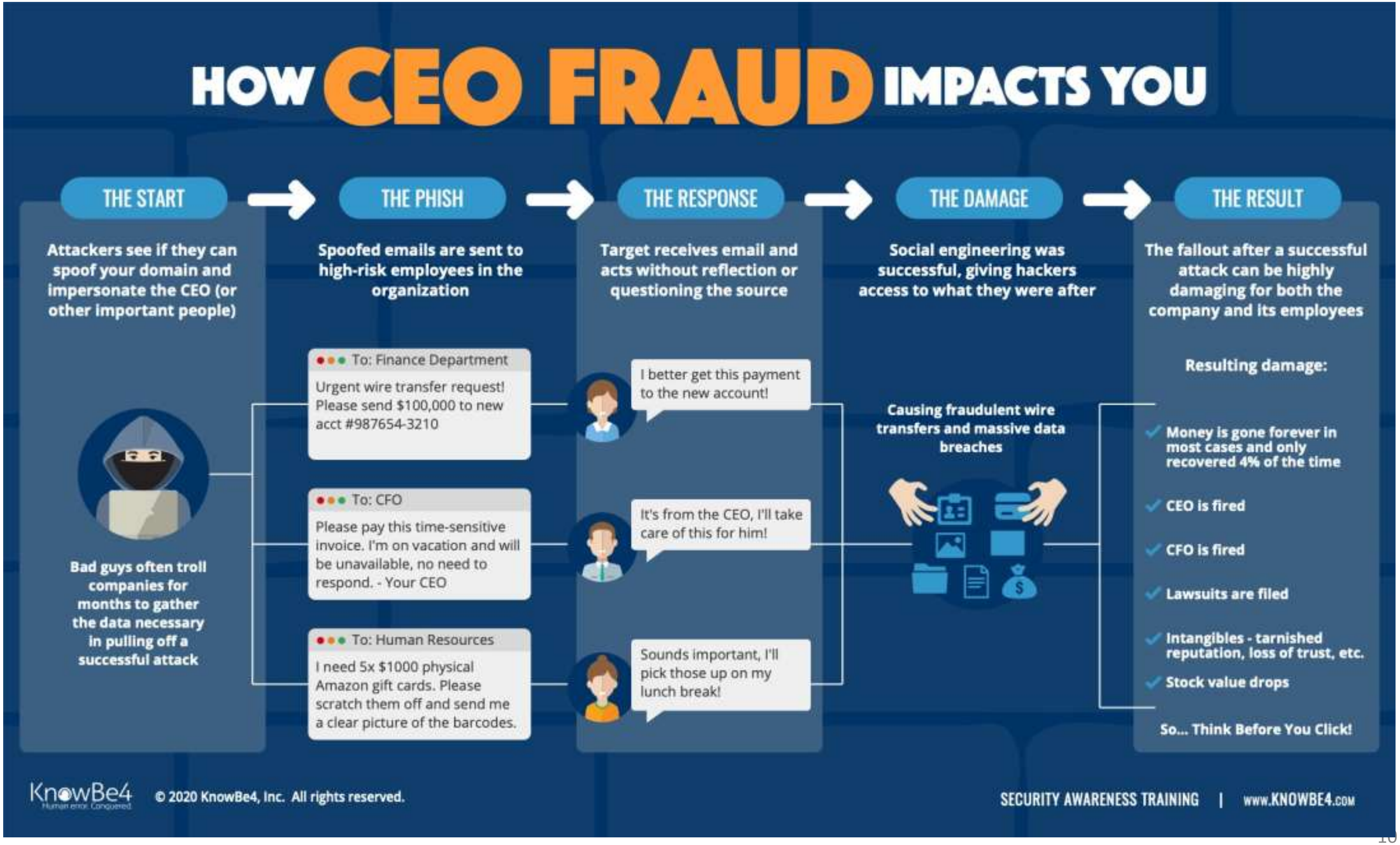


image credit: [mailguard]

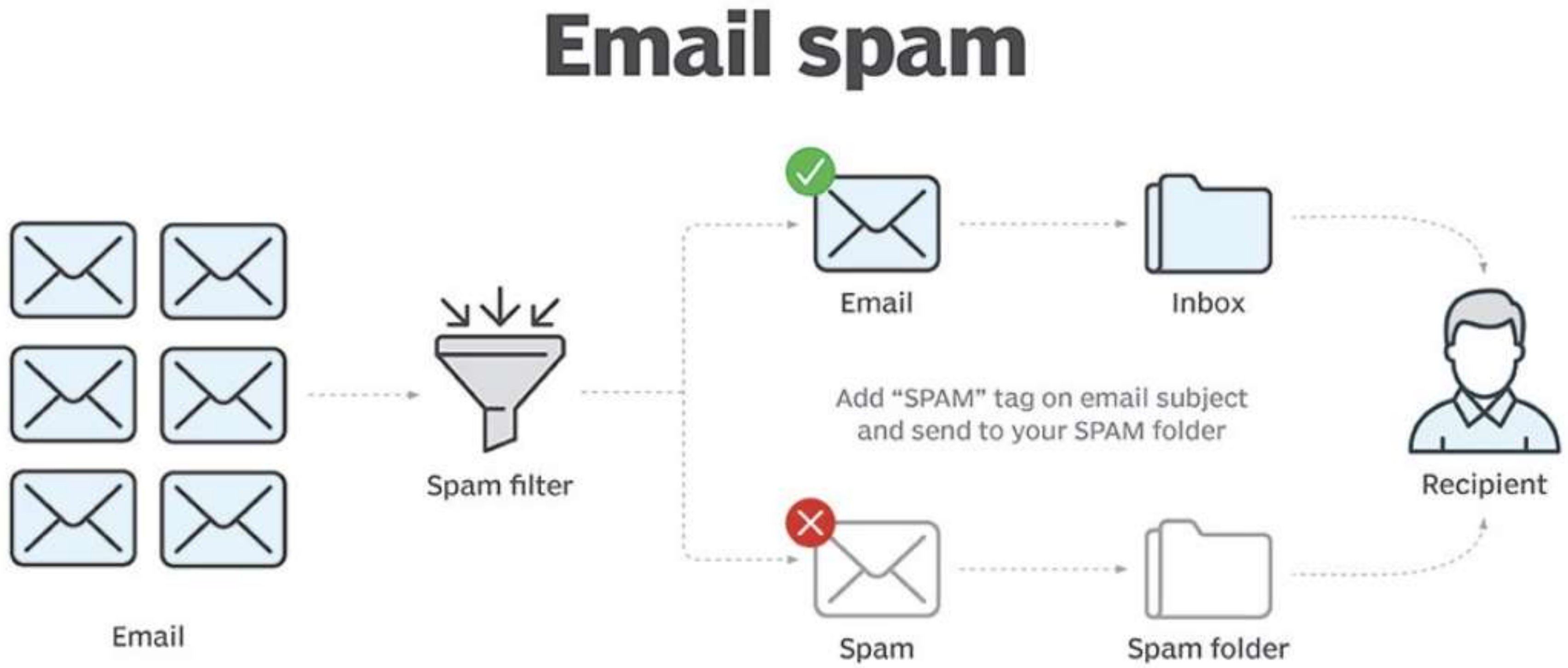
CEO Fraud

- **CEO fraud**: (also called whaling attack)
  - target: top executives of an organization
  - suffer from account takeovers due to stolen login credentials
- The scammer takes over CEO Account
- CEO’s credentials are used to login
- Scam emails sent to all


HOW CEO FRAUD IMPACTS YOU
THE START
Attackers see if they can spoof your domain and impersonate the CEO (or other important people)
Bad guys often troll companies for months to gather the data necessary in pulling off a successful attack
THE PHISH
Spoofed emails are sent to high-risk employees in the organization
To: Finance Department
Urgent wire transfer request! Please send $100,000 to new acct #987654-3210
To: CFO
Please pay this time-sensitive invoice. I'm on vacation and will be unavailable, no need to respond. - Your CEO
To: Human Resources
I need 5x $1000 physical Amazon gift cards. Please scratch them off and send me a clear picture of the barcodes.
THE RESPONSE
Target receives email and acts without reflection or questioning the source
I better get this payment to the new account!
It's from the CEO, I'll take care of this for him!
Sounds important, I'll pick those up on my lunch break!
THE DAMAGE
Social engineering was successful, giving hackers access to what they were after
Causing fraudulent wire transfers and massive data breaches
THE RESULT
The fallout after a successful attack can be highly damaging for both the company and its employees
Resulting damage:
Money is gone forever in most cases and only recovered 4% of the time
CEO is fired
CFO is fired
Lawsuits are filed
Intangibles - tarnished reputation, loss of trust, etc.
Stock value drops
So... Think Before You Click!
KnowBe4
© 2020 KnowBe4, Inc. All rights reserved.
SECURITY AWARENESS TRAINING | www.KNOWBE4.com

Spam Filtering

- Spam filter: Goal is to determine whether an incoming message is legitimate (i.e., non-spam, ham) or unsolicited (i.e., spam)



Spam Filtering

- A typical data flow in spam filtering
- How input emails are processed and sent to the model
- To give data as input to the model, we'd need to transform them to some numerical format

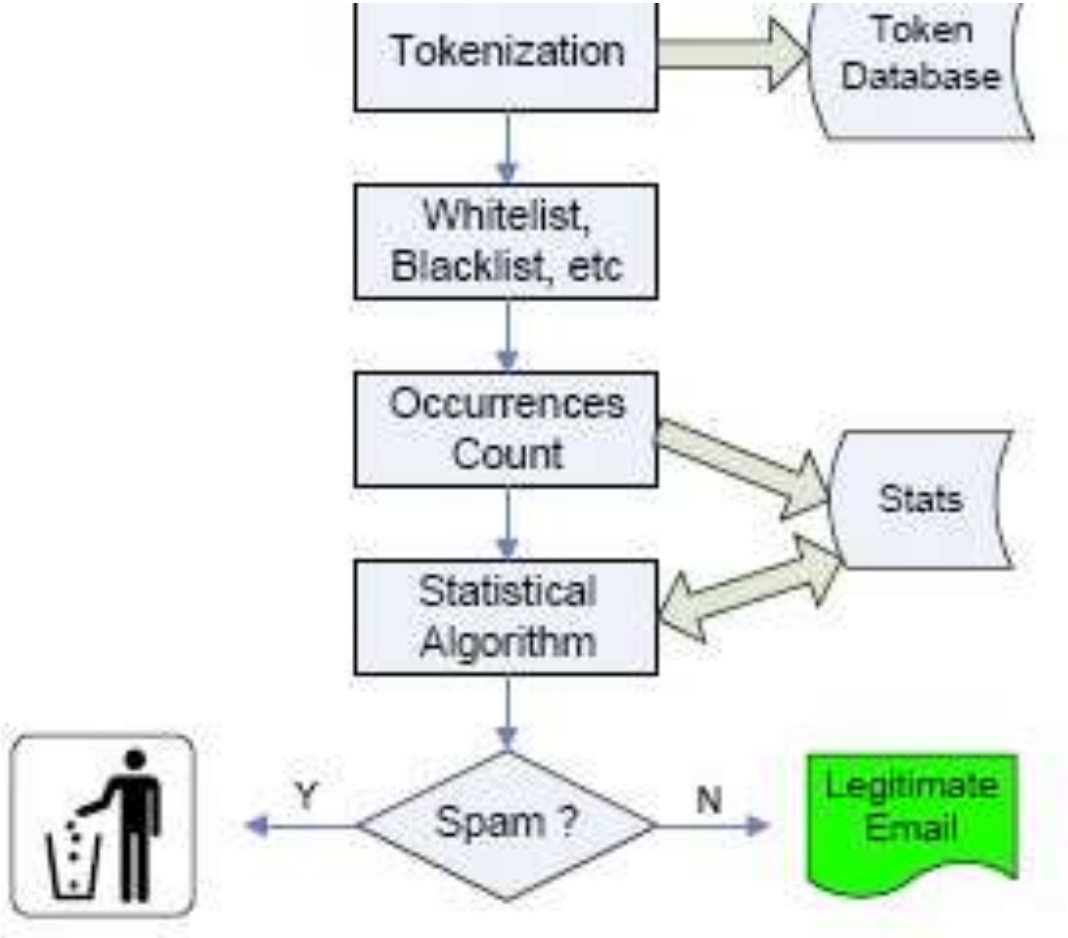


Image credit: Figure from: Yan (2010) – Workload Characterization of Spam Email Filtering Systems

Spam Filtering

- ML-based spam classifiers were among the first applications of machine learning in the cyber security domain
  - Subsequently, they were among the first to be attacked
  - Attackers' goal is to modify spam emails (without changing the nature of the message) to bypass spam filters
- Recent spam filters increasingly rely on machine learning and neural networks approaches for email classification
  - These approaches are being extensively used by email service providers like Gmail, Outlook, or Yahoo

Filtering Techniques

- Challenge-Response Filtering
- Blacklists and Whitelists
- Rule based filters
- Content based filters

Challenge-Response Filtering

- When a sender sends an email message, a challenge (e.g., CAPTHCA) is sent to the sender. A legitimate user can solve the challenge easily
- But, if a spammer forges the return address of a spam email or sends a large number of spam messages, it is difficult for the spammer to solve the challenge
- As a result, the challenge-response spam filtering system can differentiate legitimate emails from spam emails easily

- Limitation
  - Sometimes senders don’t or forget to reply to the challenge

Blacklists and Whitelists

- **Blacklists** of misbehaving servers or known spammers that are collected by several sites
- Sender id in the email is compared with the blacklist
- **Whitelists** are complementary to blacklists, and contain addresses of trusted contacts
- Use blacklists and whitelists for the first level filtering (before applying content checks) and not used as the only tool for making decision

- Limitation
  - Prone to wrong configurations with legitimate servers unable to exit from a list where they had been incorrectly inserted

Rule based Filtering

- **Rule-based filtering techniques**
  - apply static rules to discover similar patterns in a large number of spam and non-spam emails
  - Scores are assigned to each rule, and the scores are weighted based on the importance of the rule
  - Repeating patterns in a message increase the total score of being a spam
  - If the total score > a predefined threshold, the message is labeled as spam
- Rules could be created based on:
  - Words and phrases, lots of uppercase characters, exclamation points, unusual Subject lines, special characters, web links, HTML messages, background colors, etc
- Limitation
  - requires constant updating of the rules
  - continually adapting strategies by spammers

Content based Filtering

- Content based filtering techniques
  - The filter scans the content of incoming emails, looking for trigger keywords
    - E.g., keywords frequently used in spam emails, such as free, buy, application, mortgage
  - The content of the body and header of emails are scanned
- The frequency of occurrence and distribution of trigger words and phrases in the content of emails are used as features for training ML approaches, and afterward, for classifying new emails
  - Naïve Bayes classifiers are one of the early successful ML models for spam filtering
  - Other conventional ML approaches have been successfully applied, such as SVMs, k-nearest neighbors, decision trees, …
  - NNs and deep learning are commonly used nowadays for spam classification

Content based Filtering

- Scanning the body of emails explores the what in the email
  - Scanning the header of emails explores the who sent the email
- Email headers display important information, such as:
  - Message ID – an identifier generated by the sender’s email service
    - There can be no two identical message IDs, hence, it helps to detect forged email headers
  - Sender address – is used to consult blacklists to check sender’s domain reputation
  - DNS records – the DNS (Domain Name System) records of the sender allows to check the sender’s SPF, DKIM, and DMARC policies regarding email authentication
    - SPF (Sender Policy Framework), DKIM (Domain Keys Identified Mail), DMARC (Domain-based Message Authentication, Reporting and Conformance)

An example of a Gmail header

Original Message

Message ID	<000000000000d7d44705c2854da2@google.com>
Created at:	Mon, May 17, 2021 at 2:57 PM (Delivered after 12 seconds)
From:	christian@gmail.com
To:	folderly@gmail.com
Subject:	Request to connect
SPF:	PASS with IP 209.85.220.69 Learn more
DKIM:	'PASS' with domain belkins.io Learn more
DMARC:	'PASS' Learn more

Spam filters in action - a Motivating Example

- How does an anti-spam algorithm behave in the classification of emails? - based on suspicious keywords, e.g. buy, shop
- Task: classify the email messages within a table, showing the number of occurrences of the individual keywords identified within the text of the emails, indicating the messages as spam or ham:

Email	Buy	Shop	Spam or Ham?
1	1	0	H
2	0	1	H
3	0	0	H
4	1	1	S

Spam filters in action - a Motivating Example

- Need to assign a score to every single email message
  - This score will be calculated using a scoring function that takes into account the number of occurrences of suspicious keywords contained within the text
- A simple scoring function with weights $y = 2B + 3S$

Email	B	S	2B + 3S	Spam or Ham?
1	1	0	2	H
2	0	1	3	H
3	0	0	0	H
4	1	1	5	S

- Threshold value to separate spam from ham: e.g. 4

Pre-processing Text in Email Messages

- Before we can process documents (emails), it's important to convert these documents into numerical representations

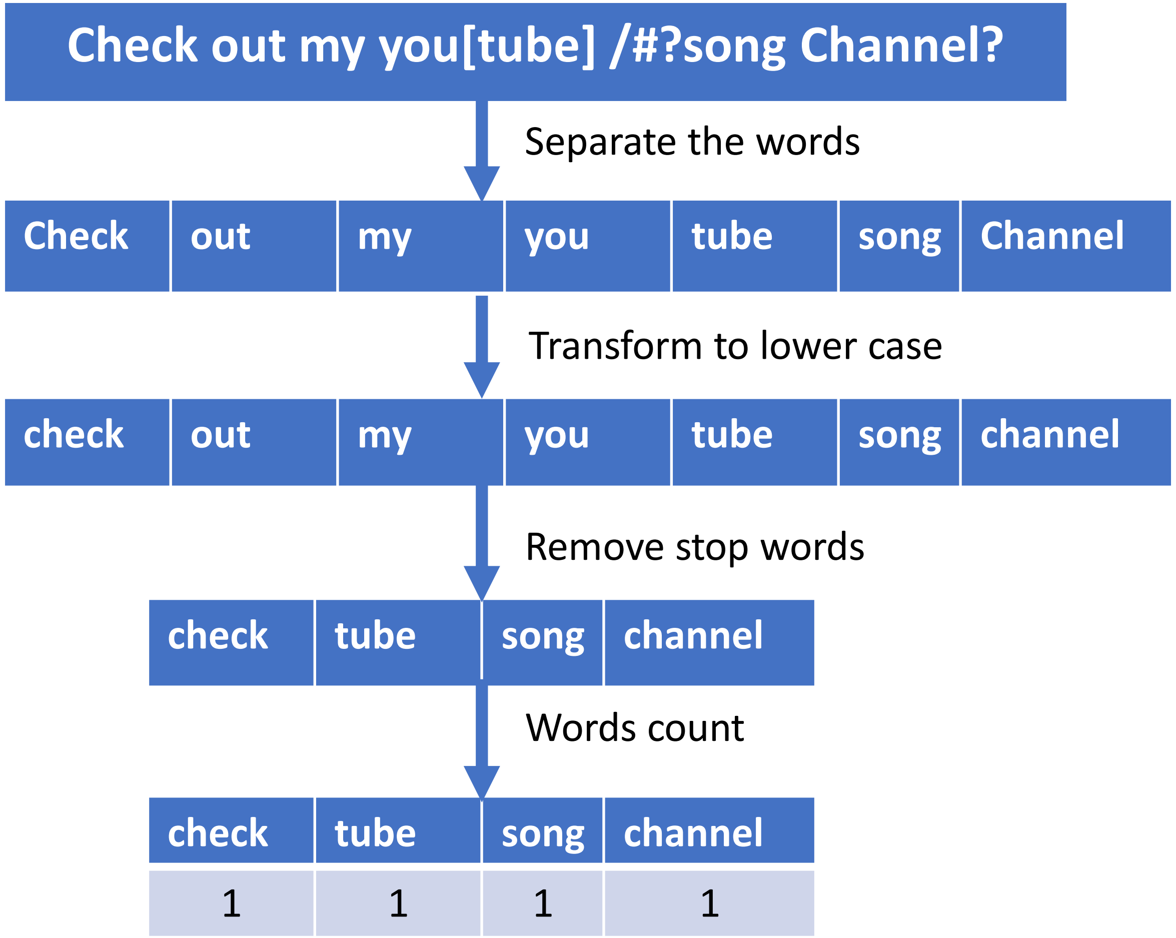
- Preprocessing text data in emails typically involves
  - Tokenization
    - Refers to separating the words in a text
    - Tokenization transforms an email into a sequence of representative symbols (tokens)
  - Vectorization
    - Transform to numerical format – 'vectors'
    - Text in Email Messages is preprocessed into numeric representation for use by ML models

Tokenization

- Tokenization usually includes:
- Remove punctuation signs (comma, period) or non-alphabetic characters (@, #, {,])
- Remove stop words, such as for, the, is, to, some
  - These words appear in both spam and non-spam emails, and are not relevant for filtering
- Correct spelling errors or abbreviations
- Change all words to lower-case letters
  - I.e., the model should consider Text and text as the same word
- Stemming - transforming words to their base form
  - E.g., the words buy-bought or grill-grilled have a common root

Tokenization

- Example of tokenization



Vectorization

- Then we need to transform the tokens (terms) to numerical format

```
tokenized_messages: {
    'A': ['hello', 'mr', 'bear'],
    'B': ['hello', 'hello', 'gunter'],
    'C': ['goodbye', 'mr', 'gunter']
}

# Bag-of-words feature vector column labels:
# ['hello', 'mr', 'doggy', 'bear', 'gunter', 'goodbye']
vectorized_messages: {
    'A': [1,1,0,1,0,0],
    'B': [2,0,0,0,1,0],
    'C': [0,1,0,0,1,1]
}
```

- Some methods:
  - Bag of words
  - n-grams
  - TF-IDF

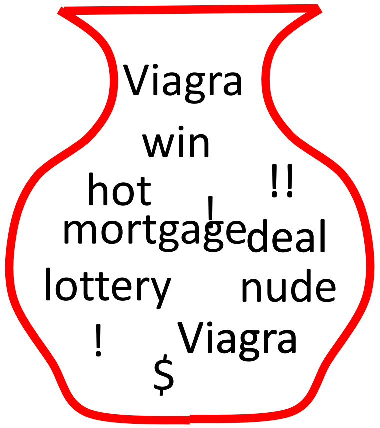
Bag-of-Words

- ***Bag-of-words model***
  - The tokenized words in emails are represented as a bag (i.e., set) of words
- The term *bag* implies that the order of the words and the structure of the text is lost
- Each word is a token having a numeric feature representation
- Typically, the frequency of occurrence of each word is used as a feature for training a classifier
- Example
  - Text: John likes to watch movies. Mary likes movies too.
  - Bag-of-words listing the words and the frequency of each word:

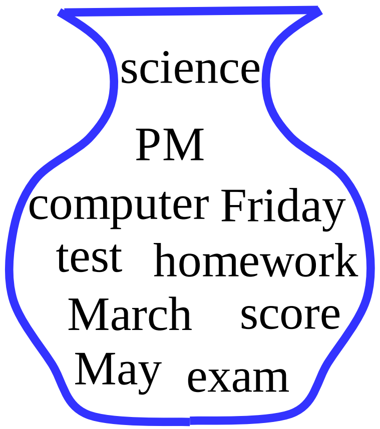
    {"John":1,"likes":2,"to":1, "watch":1, "movies":2,"Mary":1,"too":1}

Bag-of-Words

- Represent an email by the occurrence counts of each word
- What information is lost with this representation?
- **Ordering** of words is lost
  - Alice is quicker than Bob and Bob is quicker than Alice have the same vector representation
  - => n-grams
- **Term importance** is also lost
  - => TF-IDF



spam



legit

n-Grams

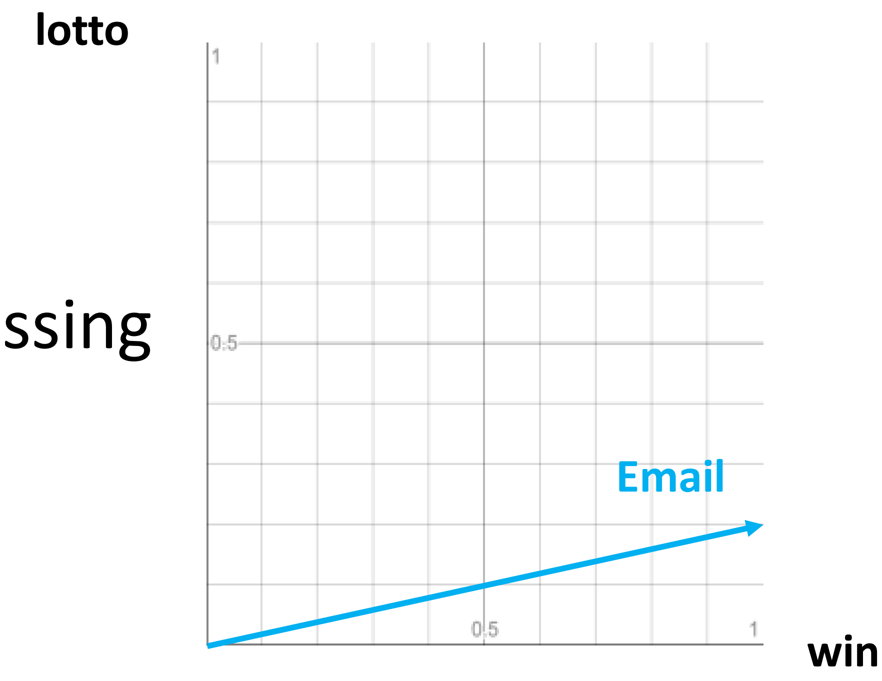
- Instead of using single words as tokens, it is also possible to use *n* consecutive words, referred to as ***n-grams***
  - Combining several consecutive words together creates more specialized tokens
  - E.g., the word *play* is considered a neutral word, but the two-words phrase *play lotto* is less neutral
    - Such *n*-grams consisting of adjacent pairs of words are called bigrams
    - *n*-grams consisting of single words are called unigrams
- The *n*-grams model preserves the words order, potentially capture more information than the bag-of-words model

TF-IDF

- **Vector-Space Model**
- Assume ***t*** distinct tokens remain after preprocessing
  - call them terms or the vocabulary
- These "orthogonal" tokens form a vector space

  Dimension = $\boldsymbol{t}$ = |vocabulary|
- Each token ***i***, in a document ***j***, is given a real-valued weight , $\boldsymbol{w_{ij}}$
- Documents are expressed as ***t***-dimensional feature vectors

  $$\boldsymbol{d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})}$$



Document Collection

- A collection of *n* documents can be represented in the vector space model by a term-document matrix
- An entry in the matrix corresponds to the "weight" of a term in the document
    - zero means the term has no significance in the document or it simply doesn't exist in the document
- How to compute w_{ij}
    - **TF-IDF (or TF.IDF)**
    - Term frequency–inverse document frequency

$$
\begin{array}{c|cccc}
 & d_1 & d_2 & \dots & d_n \\
\hline
t_1 & w_{11} & w_{12} & \dots & w_{1n} \\
t_2 & w_{21} & w_{22} & \dots & w_{2n} \\
\vdots & \vdots & \vdots & & \vdots \\
\vdots & \vdots & \vdots & & \vdots \\
t_t & w_{t1} & w_{t2} & \dots & w_{tn}
\end{array}
$$

TF-IDF: Term Frequency

- To compute the *term frequency*, we can use a term-document ***count matrix***

Documents

Terms

	Email 1	Email 2	Email 3	Email 4	Email 5	Email 6
lotto	**4**	0	0	**3**	0	0
mr	1	0	0	1	0	0
bear	**2**	0	0	0	**4**	0
gunter	0	**5**	0	0	0	0
doggy	0	0	0	0	0	**5**
win	0	**2**	**3**	0	0	0

Log-Frequency Weighting

- The ***raw term frequency***, $tf_{t,d}$, of term ***t*** in document ***d*** is defined as the number of times that ***t*** occurs in ***d***
  - However, relevance does not increase proportionally with raw term frequency
  - To this end, we can use *log-frequency weighting*

- The ***log-frequency weight***, $\boldsymbol{w_{t,d}}$, of term ***t*** in document ***d*** is:

$$\boldsymbol{w_{t,d}} = \begin{cases} \boldsymbol{1 + log\left(tf_{t,d}\right)\ if\ tf_{t,d} > 0} \\ \boldsymbol{0\ otherwise} \end{cases}$$

Log-Frequency Weighting: Example

- Here is our previous term-document ***count matrix***

Documents

	Email 1	Email 2	Email 3	Email 4	Email 5	Email 6
lotto	**4**	0	0	**3**	0	0
mr	1	0	0	1	0	0
bear	**2**	0	0	0	**4**	0
gunter	0	**5**	0	0	0	0
doggy	0	0	0	0	0	**5**
win	0	**2**	**3**	0	0	0

$tf_{win,\ Email\ 3} = 3 > 0$

$tf_{win\ \ Email\ 5} = 0$

Log-Frequency Weighting: Example

- Here is the corresponding term-document ***log frequency matrix***

Documents

	Email 1	Email 2	Email 3	Email 4	Email 5	Email 6
lotto	1.602	0	0	1.477	0	0
mr	1	0	0	1	0	0
bear	1.301	0	0	0	1.602	0
gunter	0	0	0	0	1	0
doggy	0	1.698	0	0	0	1.698
win	0	1.301	1.477	0	0	0

$$w_{win,\, Email\, 3} = 1 + log(tf_{win,\, Email\, 3}) = 1 + log(3) = 1.477$$

$$w_{win,\, Email\, 5} = 0$$

Document frequency

- Rare terms are more informative than frequent terms
  - Recall stop words: “a”, “the”, “to”, “of”, etc., are frequent but not very informative

- Want higher weights for rare terms than for more frequent terms

- Use document frequency to capture this

Document frequency

- **document frequency** $\boldsymbol{df_t}$, of term $\boldsymbol{t}$
  - the number of documents in the given collection that contain $\boldsymbol{t}$
- Thus, $\boldsymbol{df_t \leq N}$, where $\boldsymbol{N}$ is the number of documents in the collection

- $\boldsymbol{df_t}$ is an inverse measure of the informativeness of $\boldsymbol{t}$
  - *the smaller the number of documents that contain* ***t*** *the more informative* ***t*** *is*
- **Inverse document frequency**, $\boldsymbol{idf_t}$, is defined as:

$$\boldsymbol{idf_t = log\left(\frac{N}{df_t}\right)}$$

If $\boldsymbol{df_t}$ = 1, $\boldsymbol{idf_t = log(N)}$
If $\boldsymbol{df_t}$ = N, $\boldsymbol{idf_t = 0}$

Document frequency

- **document frequency** $\boldsymbol{df_t}$, of term $\boldsymbol{t}$
  - the number of documents in the given collection that contain $\boldsymbol{t}$
- Thus, $\boldsymbol{df_t \leq N}$, where $\boldsymbol{N}$ is the number of documents in the collection
- $\boldsymbol{df_t}$ is an inverse measure of the informativeness of $\boldsymbol{t}$
  - *the smaller the number of documents that contain* ***t*** *the more informative* ***t*** *is*
- **Inverse document frequency**, $\boldsymbol{idf_t}$, is defined as:

$$\boldsymbol{idf_t = \log\left(\frac{N}{df_t}\right)}$$

Note: There is only one value of $\boldsymbol{idf_t}$ for each $\boldsymbol{t}$ in the collection

TF-IDF Weighting

- A typical combined term importance indicator is *TF.IDF weighting*:

$$\boldsymbol{w_{t,d} = TF.IDF = (1 + log}(tf_{t,d})) \times \boldsymbol{log\left(\frac{N}{df_t}\right)}$$

- A term occurring frequently in the document but rarely in the rest of the collection is given high weight
- **TF.IDF** increases with:
  - the number of occurrences of term ***t*** in document ***d***
  - the rarity of ***t*** in the collection

TF-IDF: Example

- here is the respective term-document ***TF-IDF matrix***

Documents

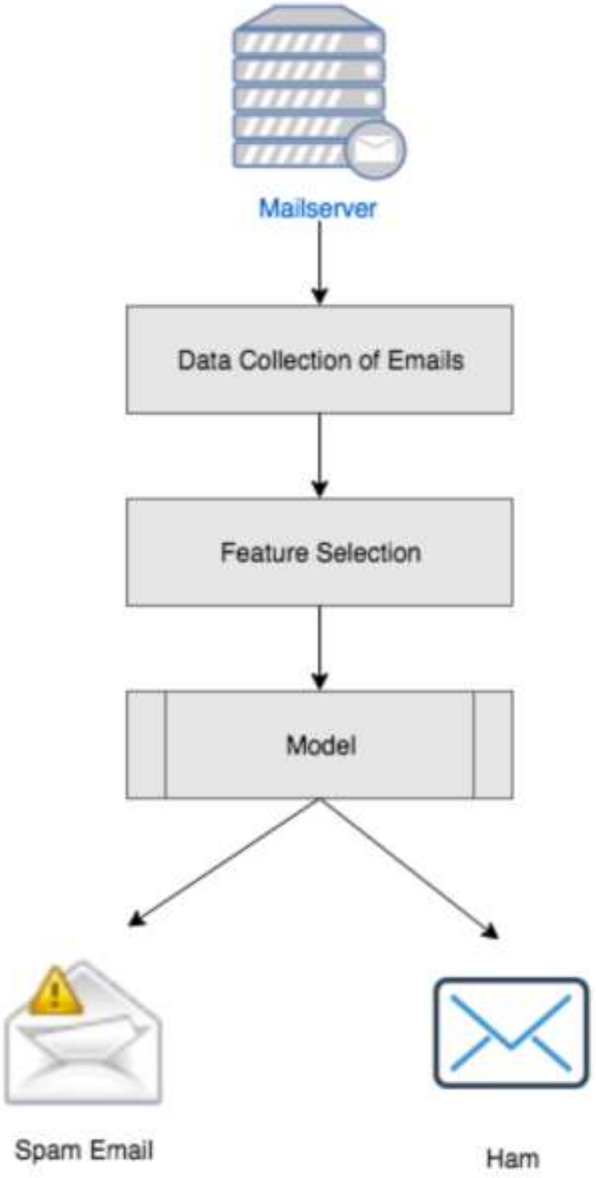
	Email 1	**Email 2**	**Email 3**	**Email 4**	**Email 5**	**Email 6**
lotto	0.764	0	0	0.704	0	0
mr	0.477	0	0	0.477	0	0
bear	0.620	0	0	0	0.764	0
gunter	0	0	0	0	0.778	0
doggy	0	0.810	0	0	0	0.810
win	0	0.620	0.704	0	0	0

$$w_{win,\, Email\, 3} = [1 + log(tf_{win,\, Email\, 3})] \times log\left(\frac{N}{df_{win}}\right) = 1.477 \times log\left(\frac{6}{2}\right) = 0.704$$

Spam Detection

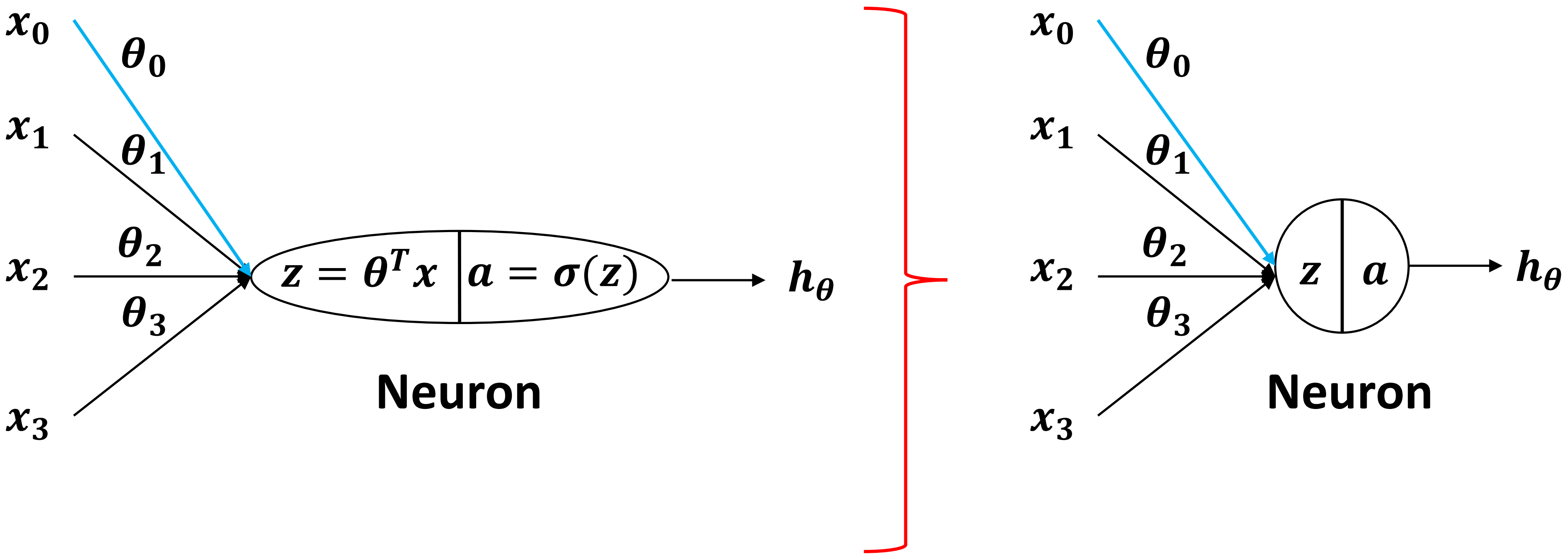
- Task: separate spam emails from a set of non-spam emails
  - Given a large collection of example emails, each labeled “spam” or “ham”
  - Learn to predict labels of new, future emails

- How: use neural networks to distinguish between spam and ham emails



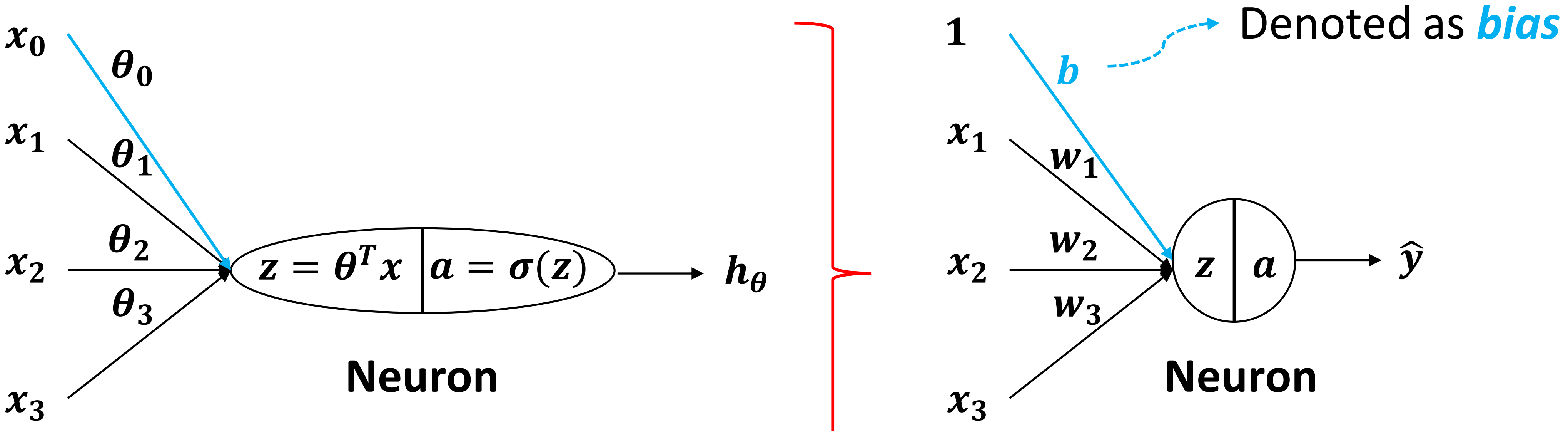
Towards Neural Networks

- Technically, logistic regression is a **neural network** with only 1 neuron



Towards Neural Networks

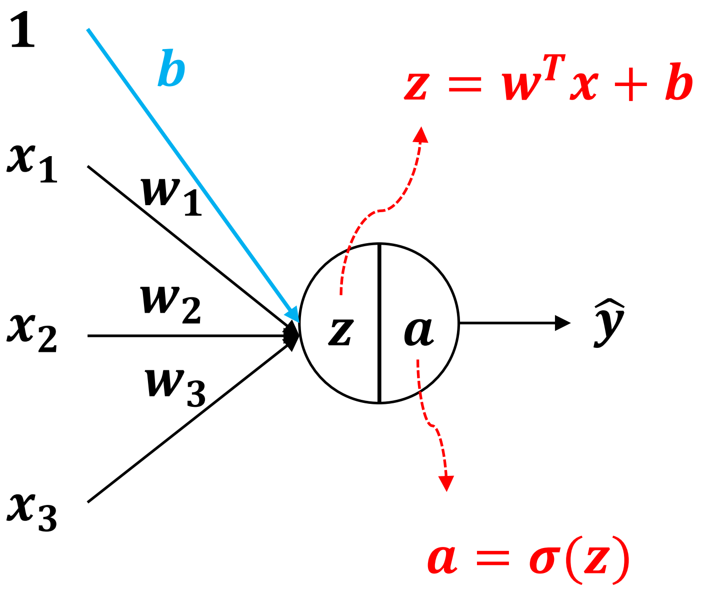
- Technically, logistic regression is a **neural network** with only 1 neuron



Using the notations in the neural network literature, where $\boldsymbol{\theta} = \boldsymbol{w} = [\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}]$ ($\boldsymbol{w_0}$ is not part of this vector here), $\boldsymbol{h_\theta} = \boldsymbol{\hat{y}}$, and $\boldsymbol{\theta_0} = \boldsymbol{w_0} = \boldsymbol{b}$

Towards Neural Networks

- Technically, logistic regression is a **neural network** with only 1 neuron



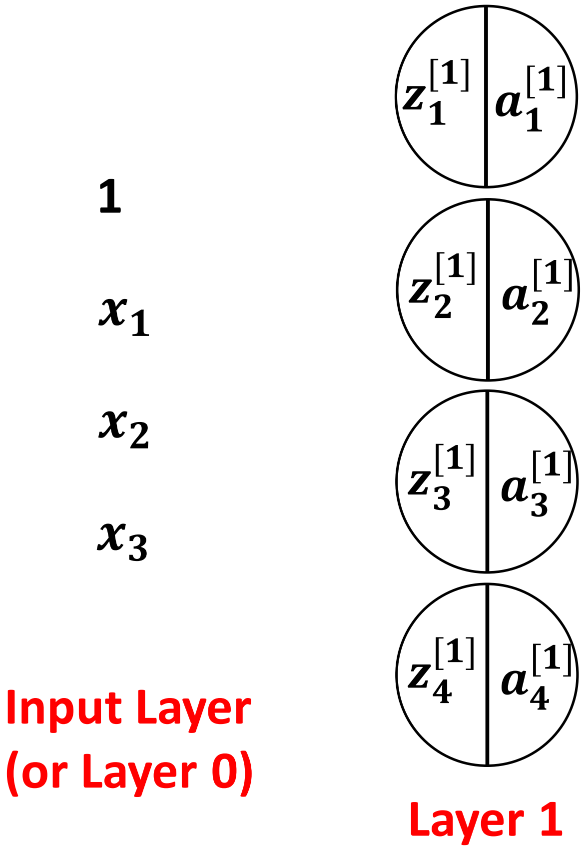
$$\hat{y} = a = \sigma(z) = \sigma\left(w^T x + b\right) = \sigma\left([w_1\ w_2\ w_3]\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b\right)$$

$$= \sigma(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$$

$$= \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)}}$$

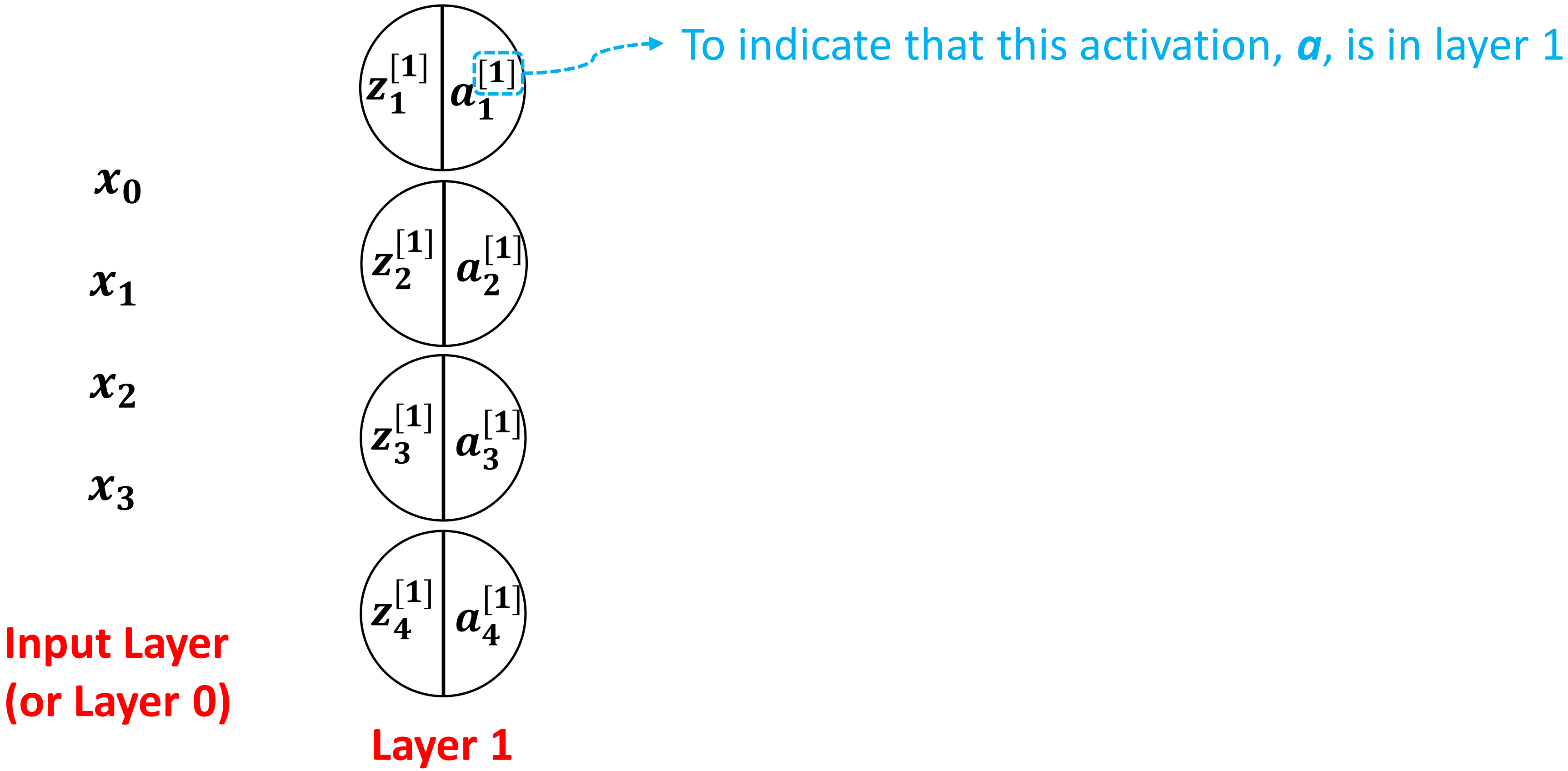
Neural Networks

- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



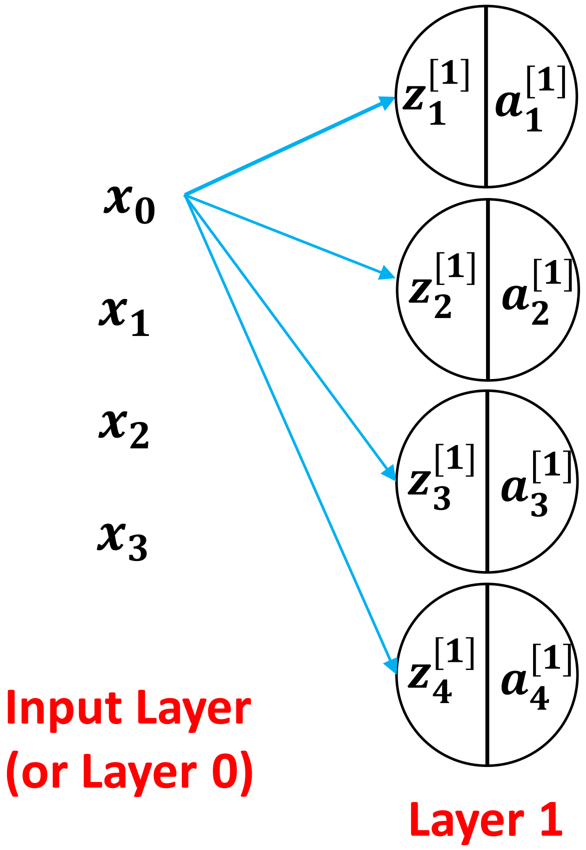
Neural Networks

- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



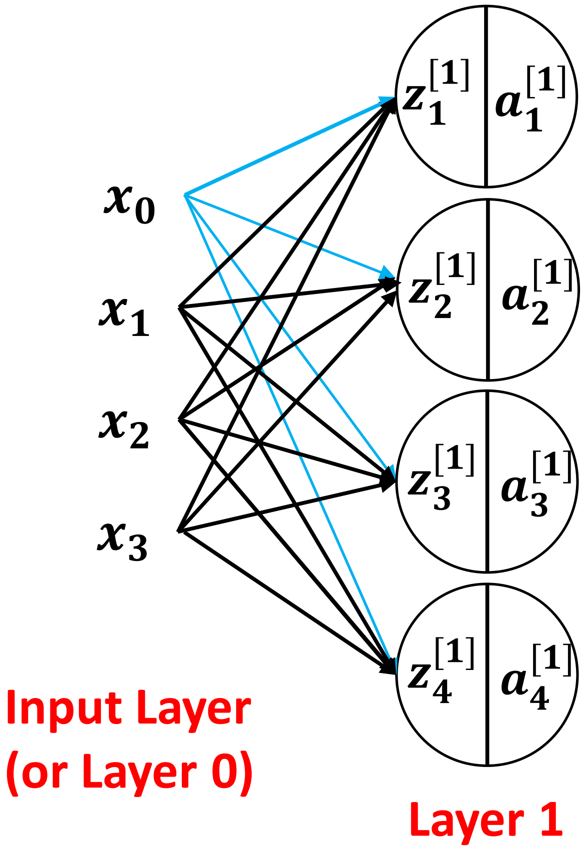
Neural Networks

- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



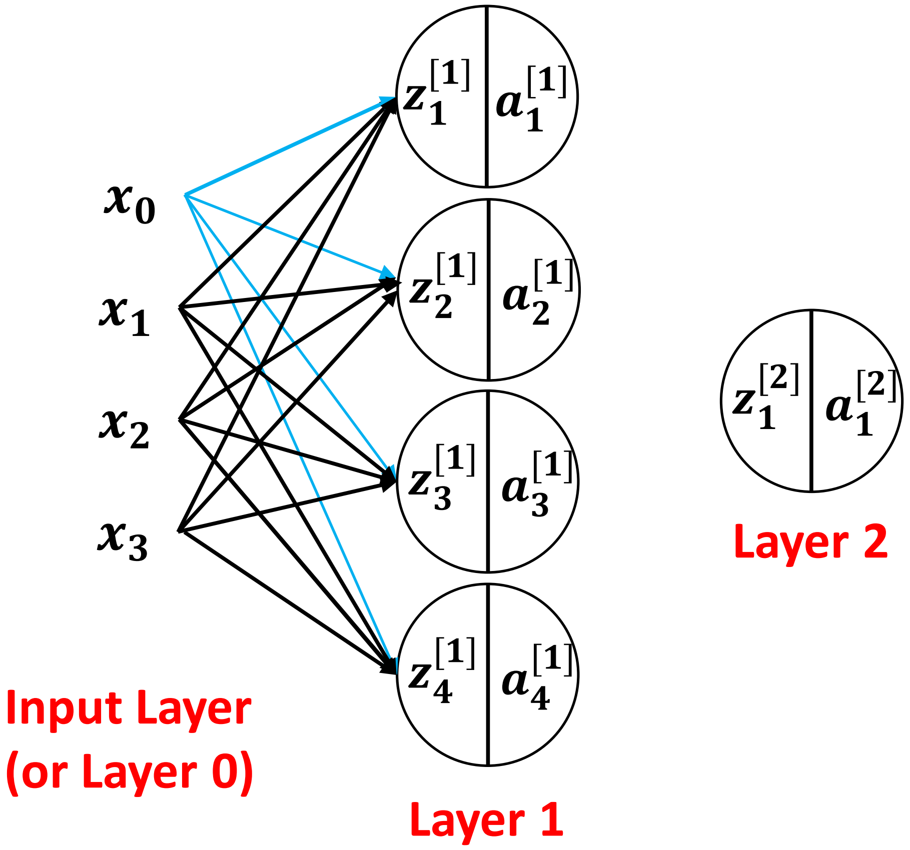
Neural Networks

- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



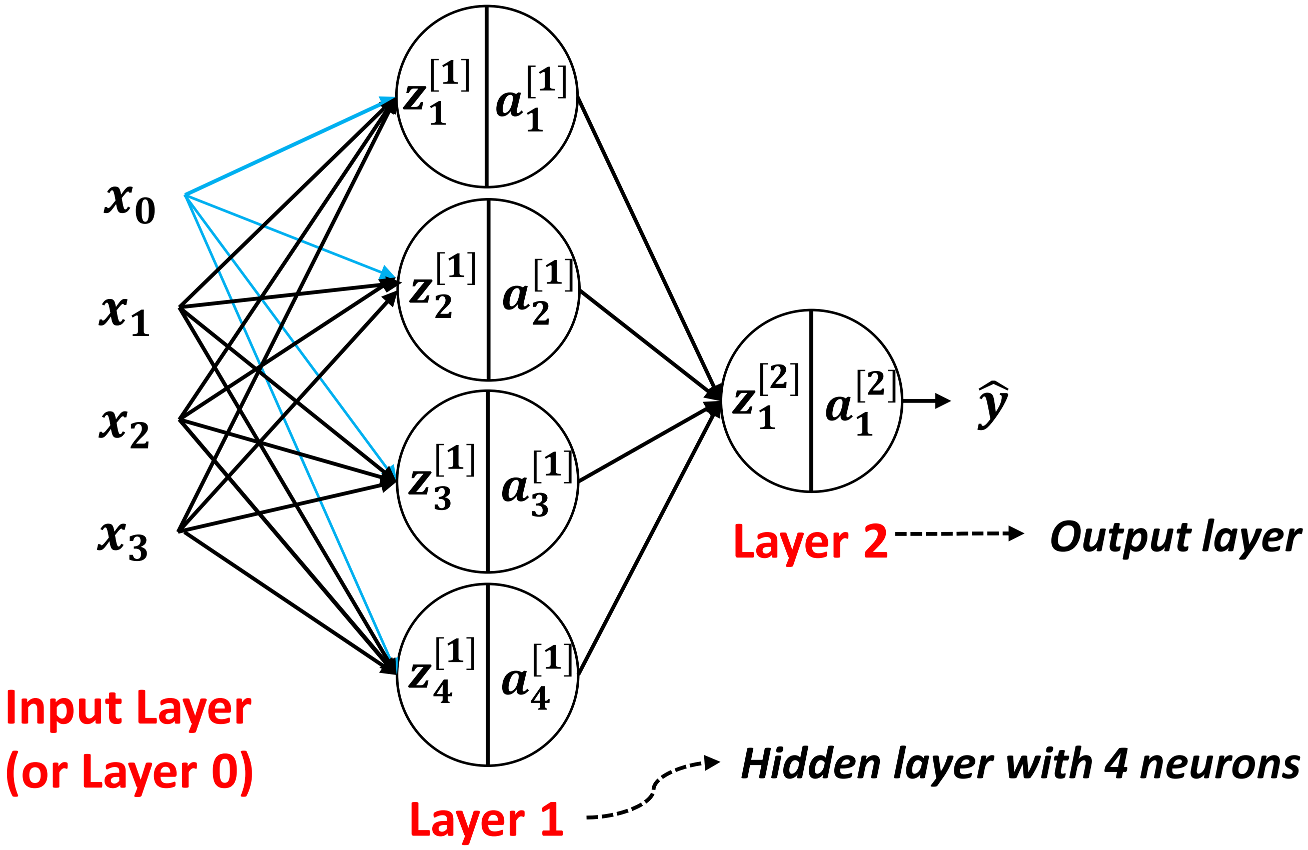
Neural Networks

- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



Neural Networks

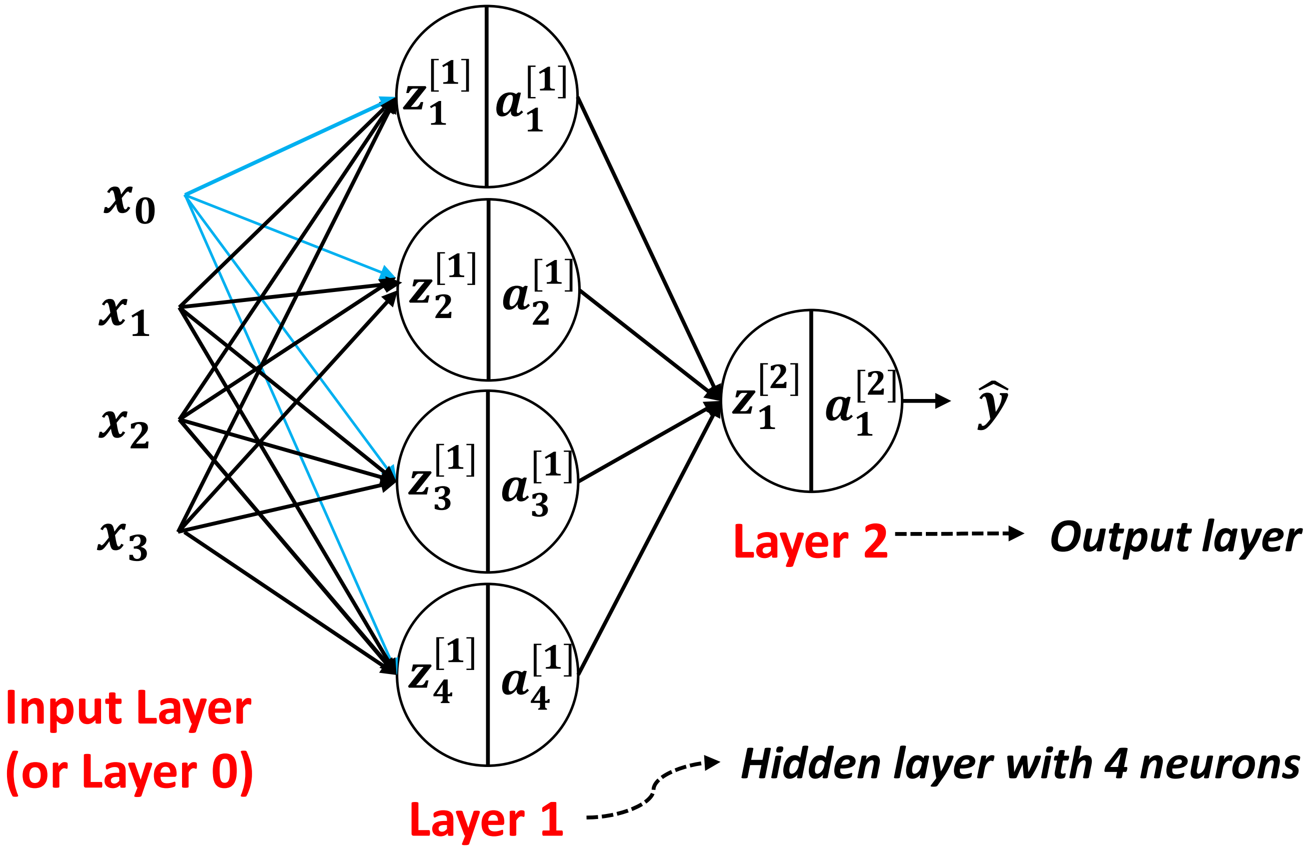
- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



By convention, this neural network is said to have 2 layers (and not 3) since the input layer is typically not counted!

Neural Networks

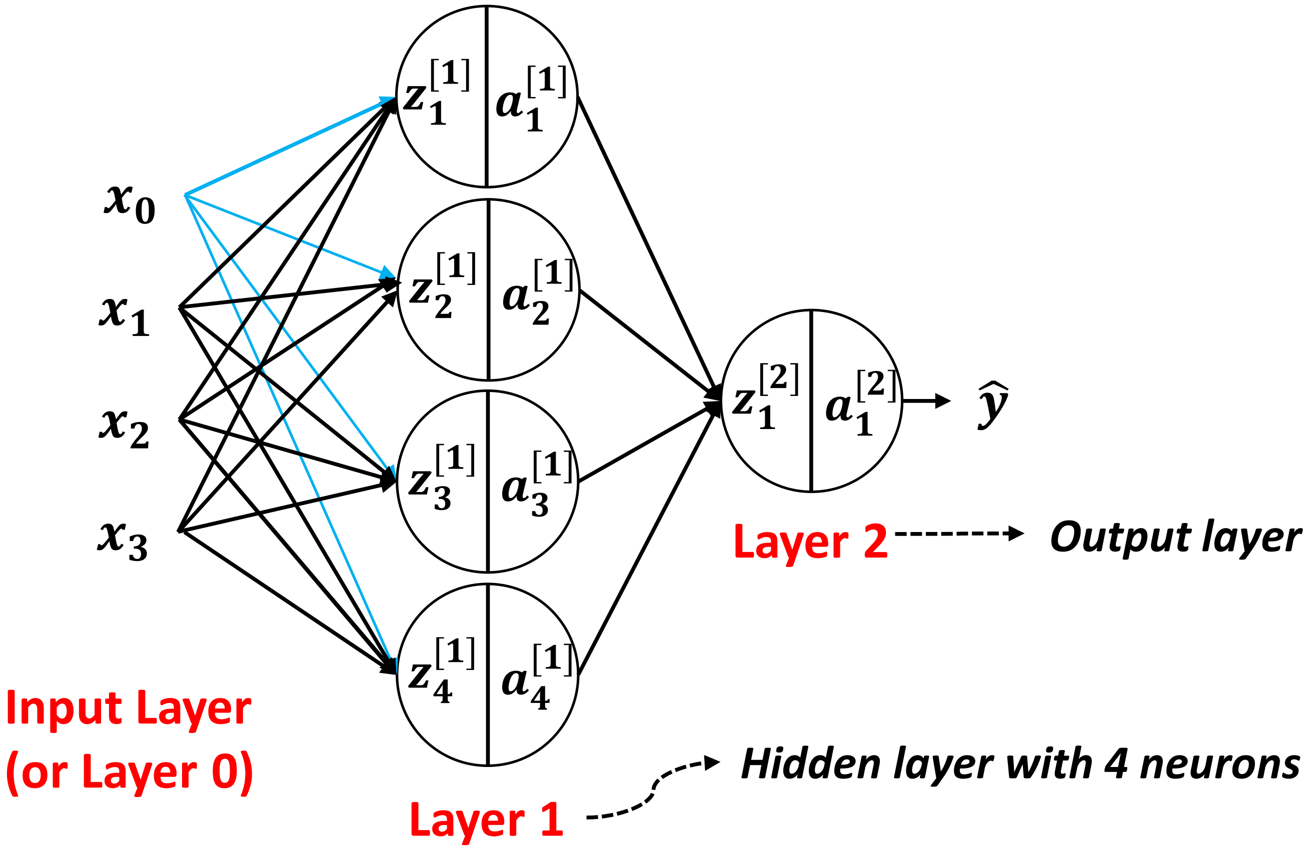
- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



Also, the more layers we add, the ***deeper*** the neural network becomes, giving rise to the concept of ***deep learning***!

Neural Networks

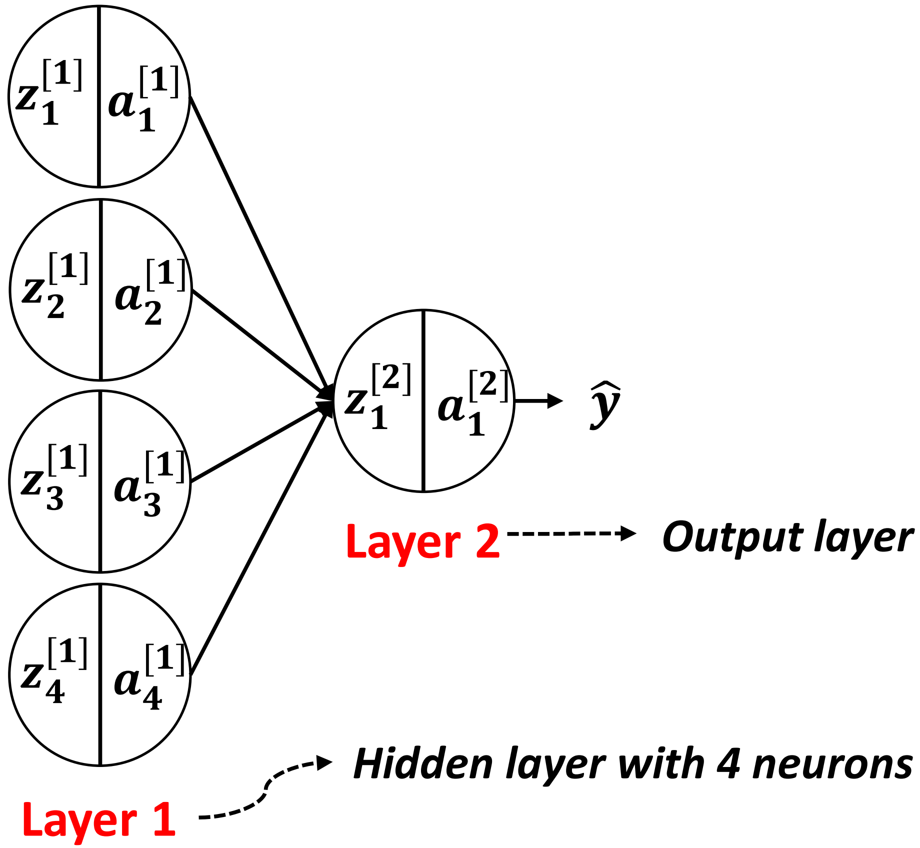
- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



Interestingly, neural networks *learn* their own features!

Neural Networks

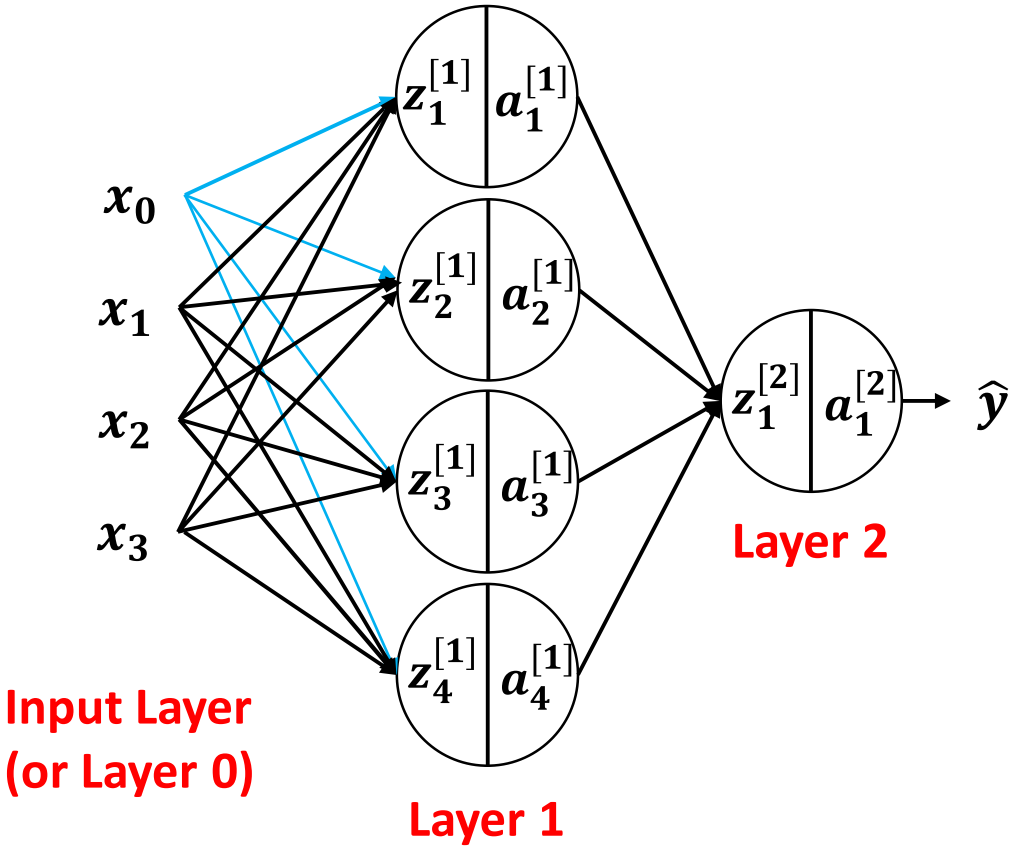
- We can construct a network of neurons (i.e., a neural network) with as many *layers,* and neurons in any layer, as needed



This looks like logistic regression, but with ***features that were learnt*** (i.e., $a_1^{[1]}$, $a_2^{[1]}$, $a_3^{[1]}$, $a_4^{[1]}$) ***and NOT engineered by us*** (i.e., x_1, x_2, and x_3)

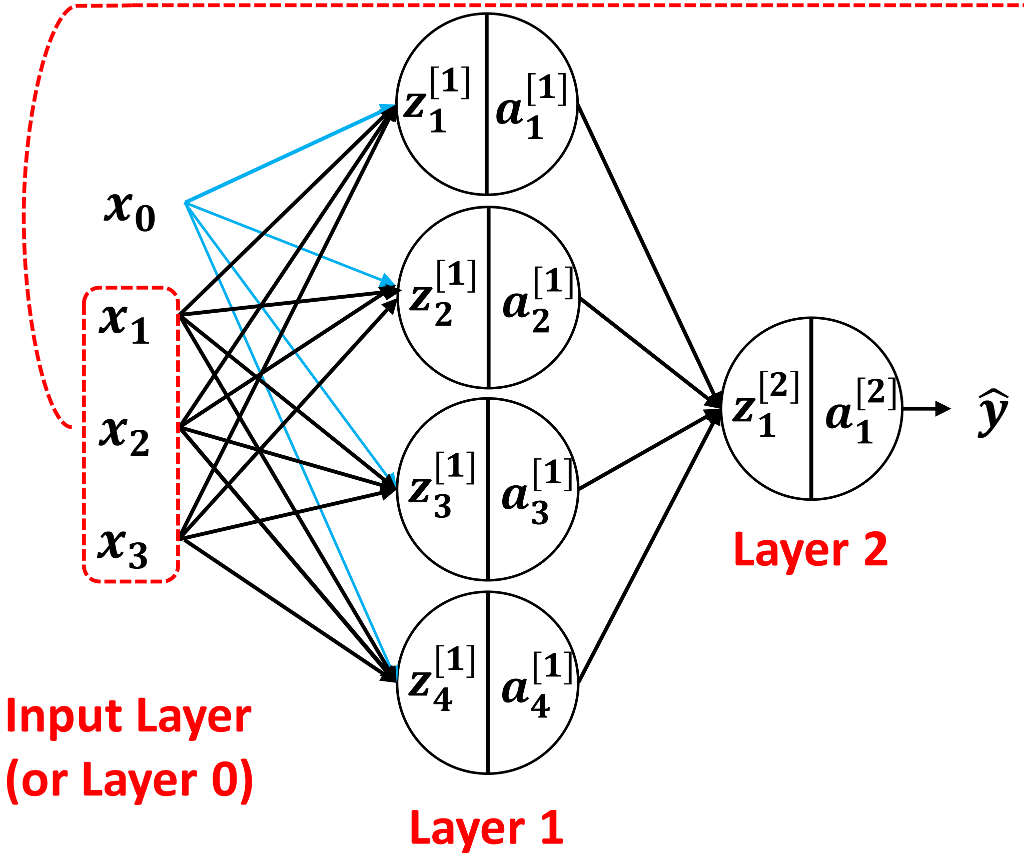
Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



Vectorizing Input and All Variables

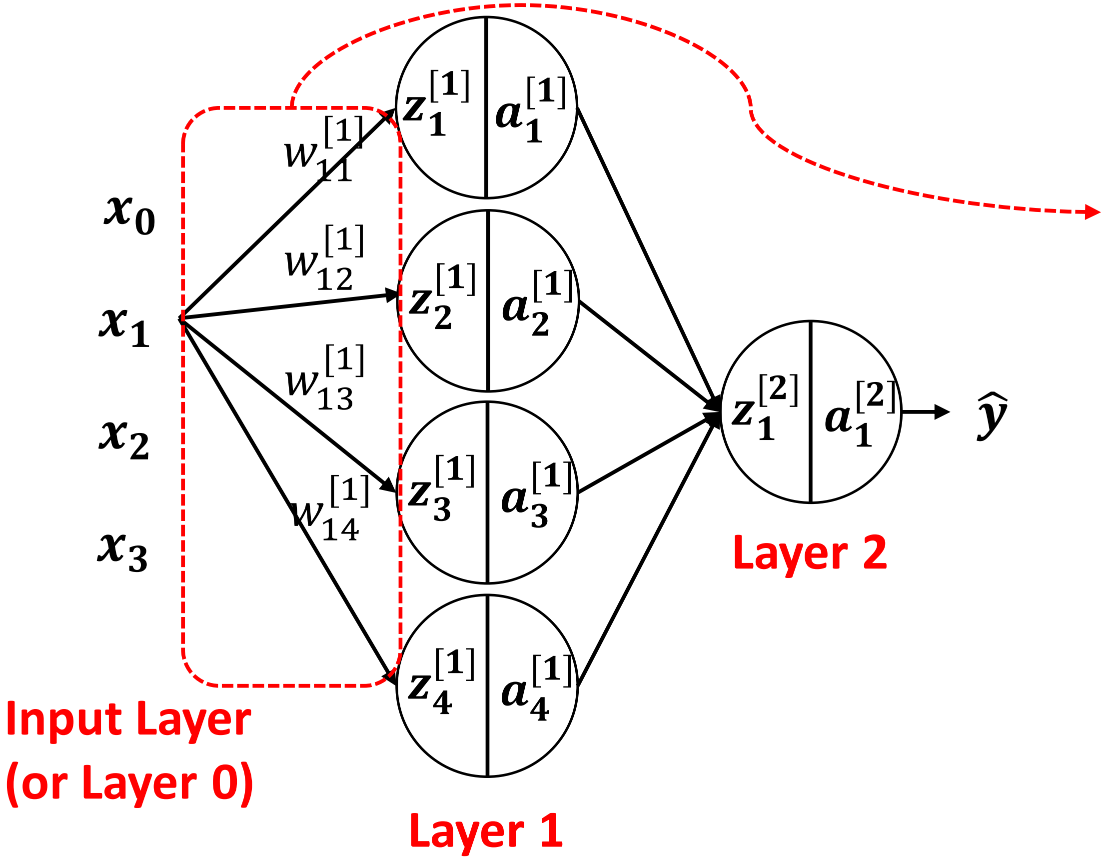
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Vectorizing Input and All Variables

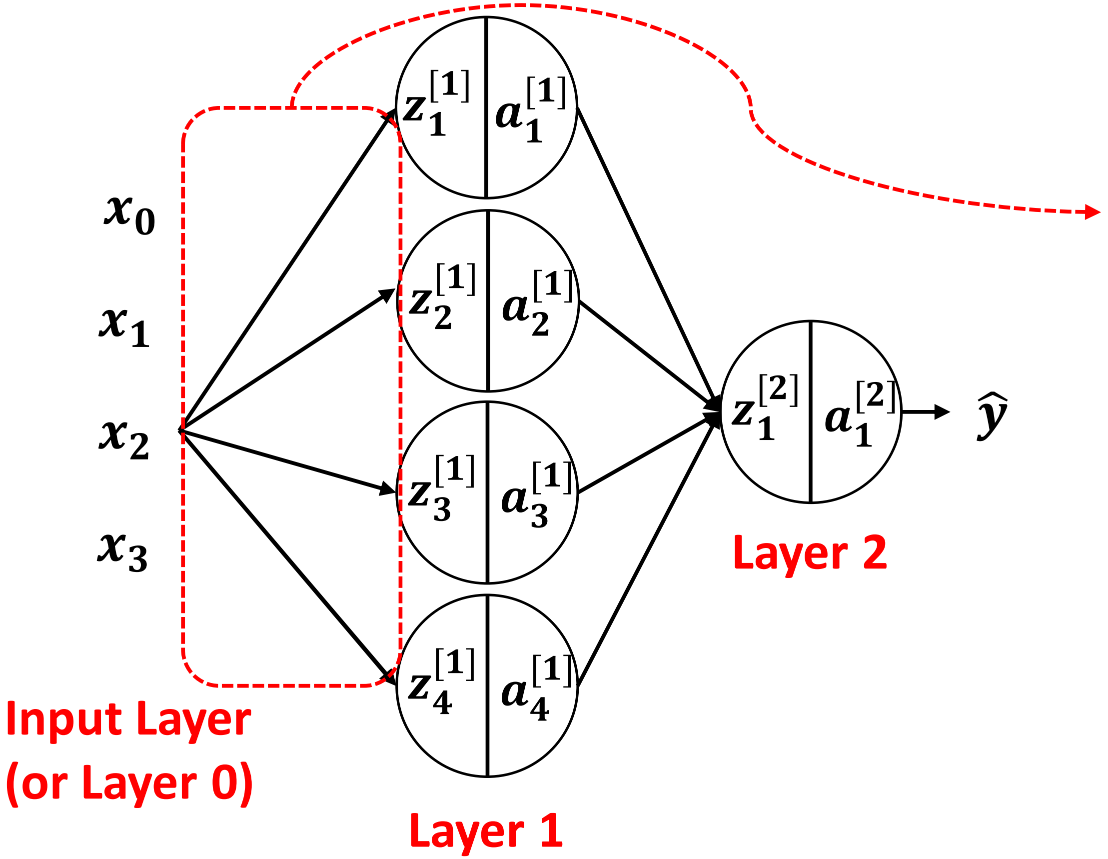
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{w}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} & w_{14}^{[1]} \\ & & & \\ & & & \end{bmatrix}$$

Vectorizing Input and All Variables

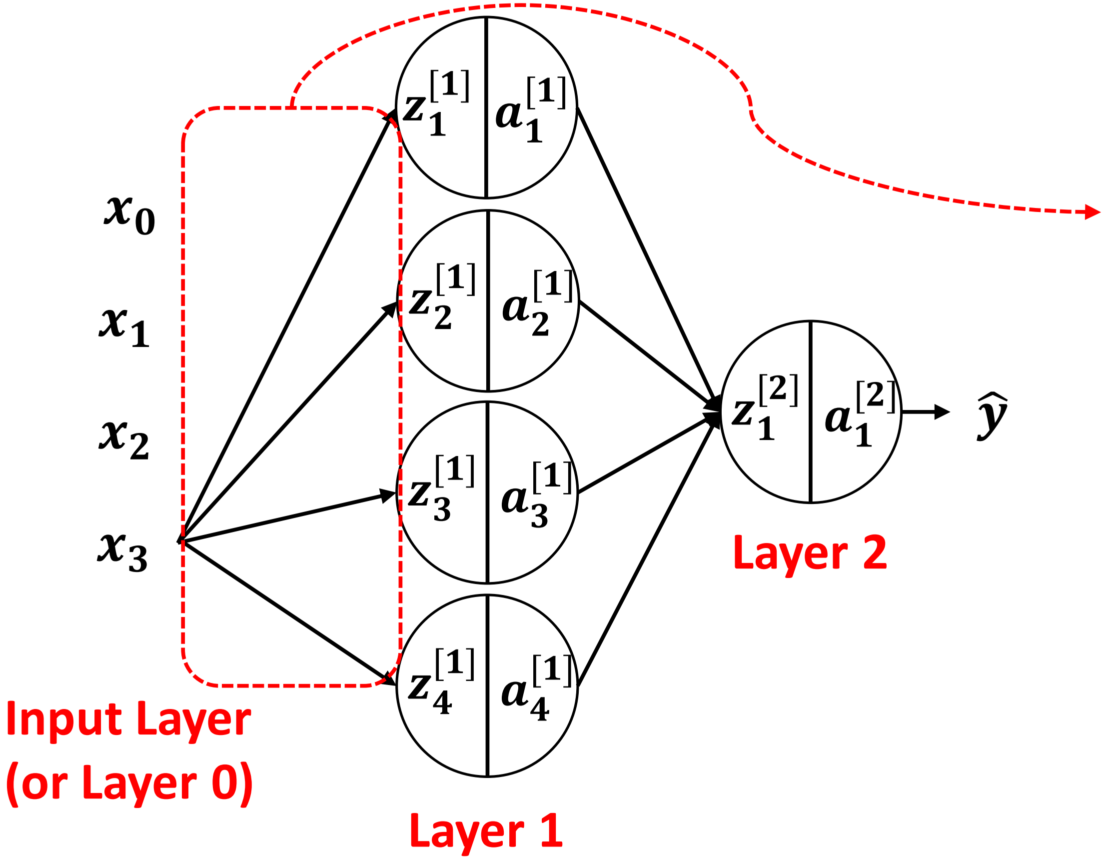
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{w}^{[1]} = \begin{bmatrix} w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} & w_{24}^{[1]} \end{bmatrix}$$

Vectorizing Input and All Variables

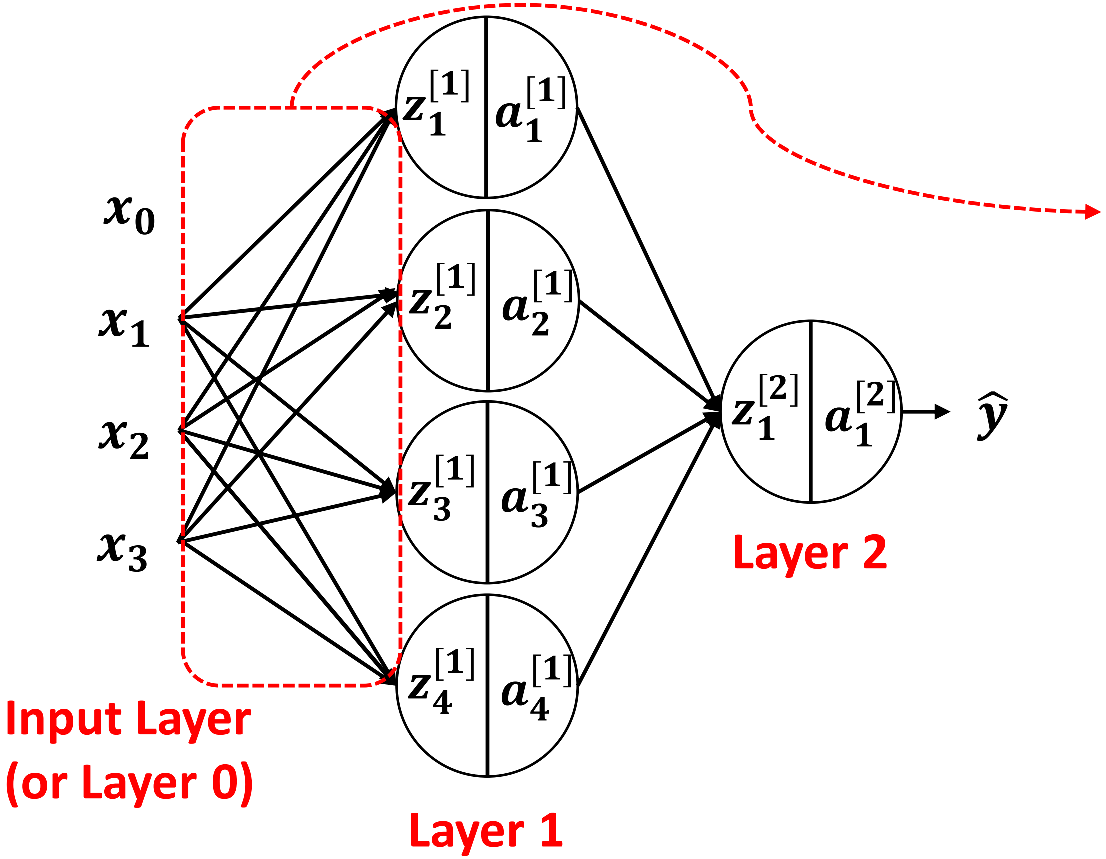
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$w^{[1]} = \begin{bmatrix} & & & \\ & & & \\ w_{31}^{[1]} & w_{32}^{[1]} & w_{33}^{[1]} & w_{34}^{[1]} \end{bmatrix}$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

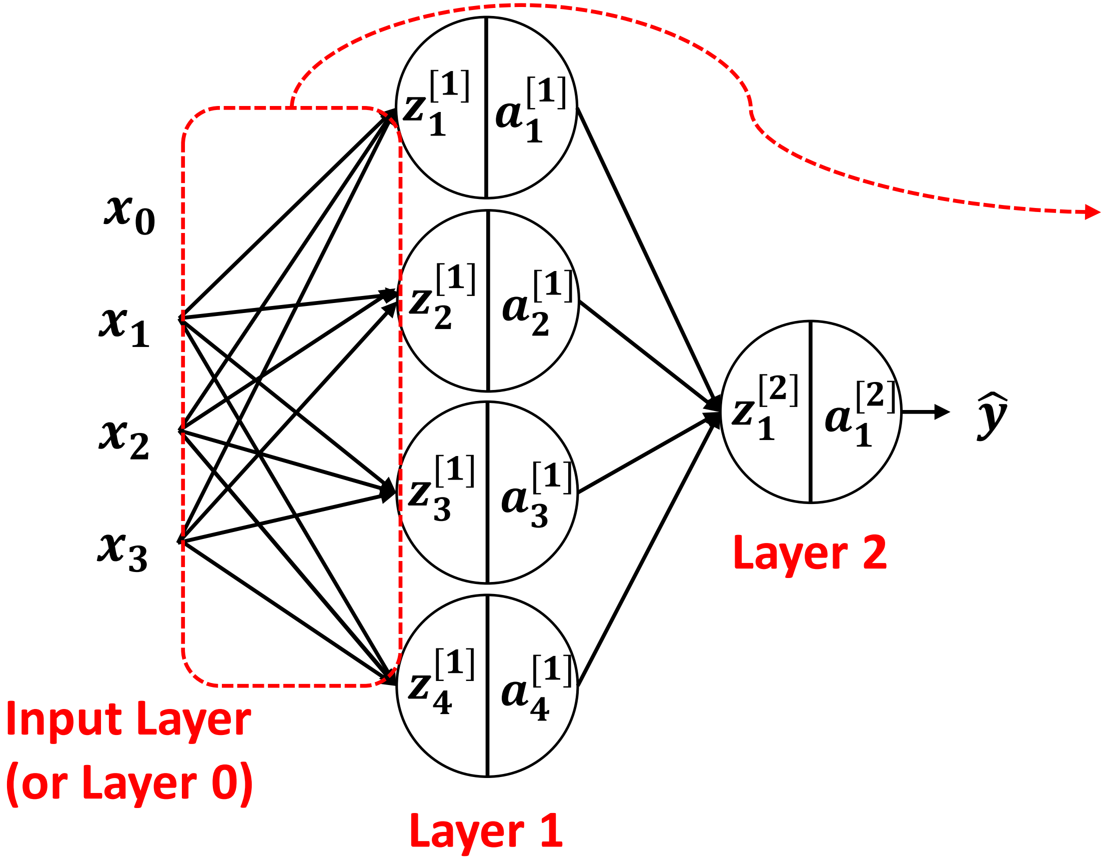


$$\boldsymbol{w}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} & w_{14}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} & w_{24}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} & w_{33}^{[1]} & w_{34}^{[1]} \end{bmatrix}$$

Dimension of $w^{[1]}$ = (3, 4)

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

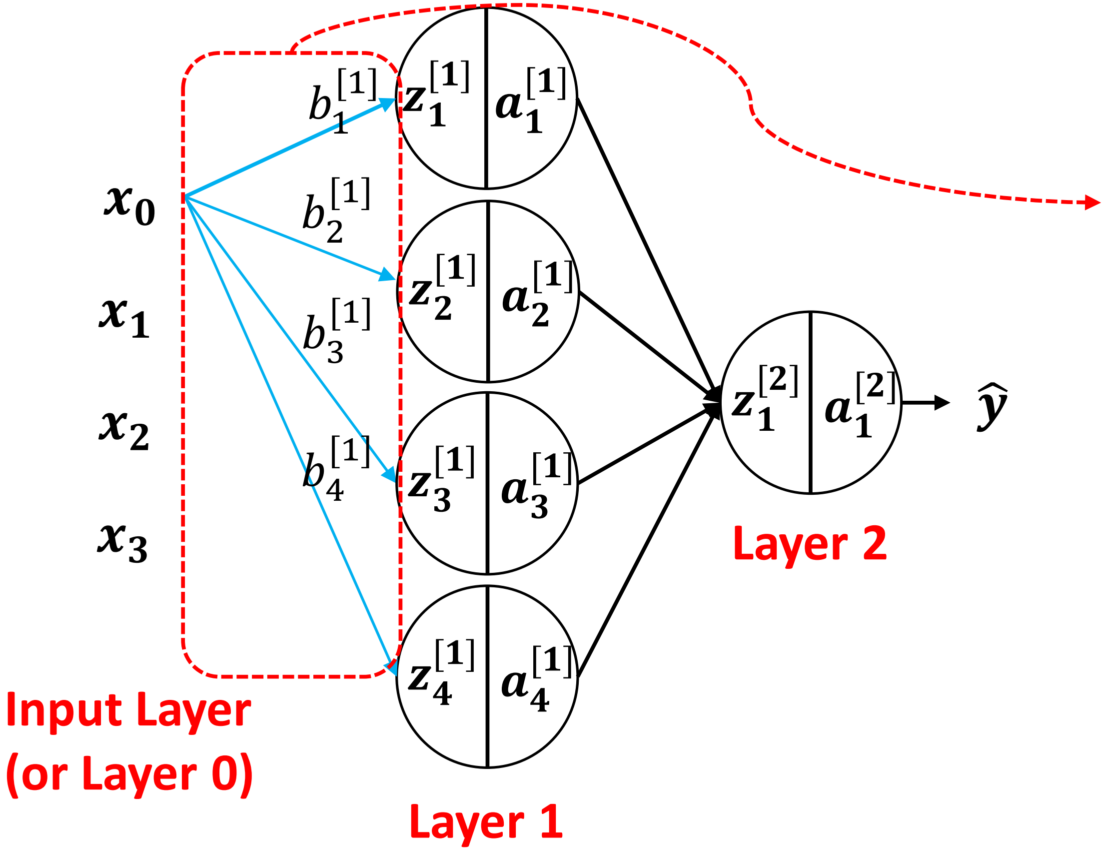


$$w^{[1]^T} = \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} & w_{31}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} & w_{32}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} & w_{33}^{[1]} \\ w_{14}^{[1]} & w_{24}^{[1]} & w_{34}^{[1]} \end{bmatrix}$$

Dimension of $w^{[1]^T}$ = (4, 3)

Vectorizing Input and All Variables

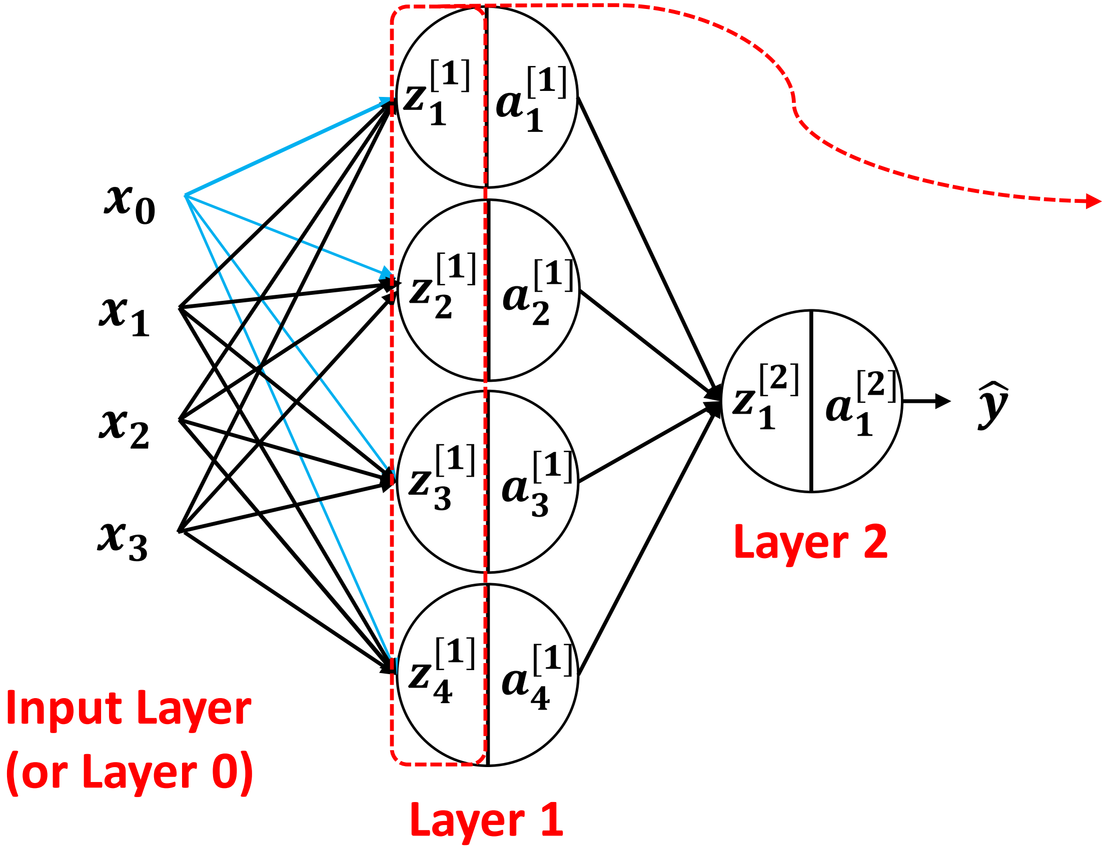
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{b}^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}$$

Vectorizing Input and All Variables

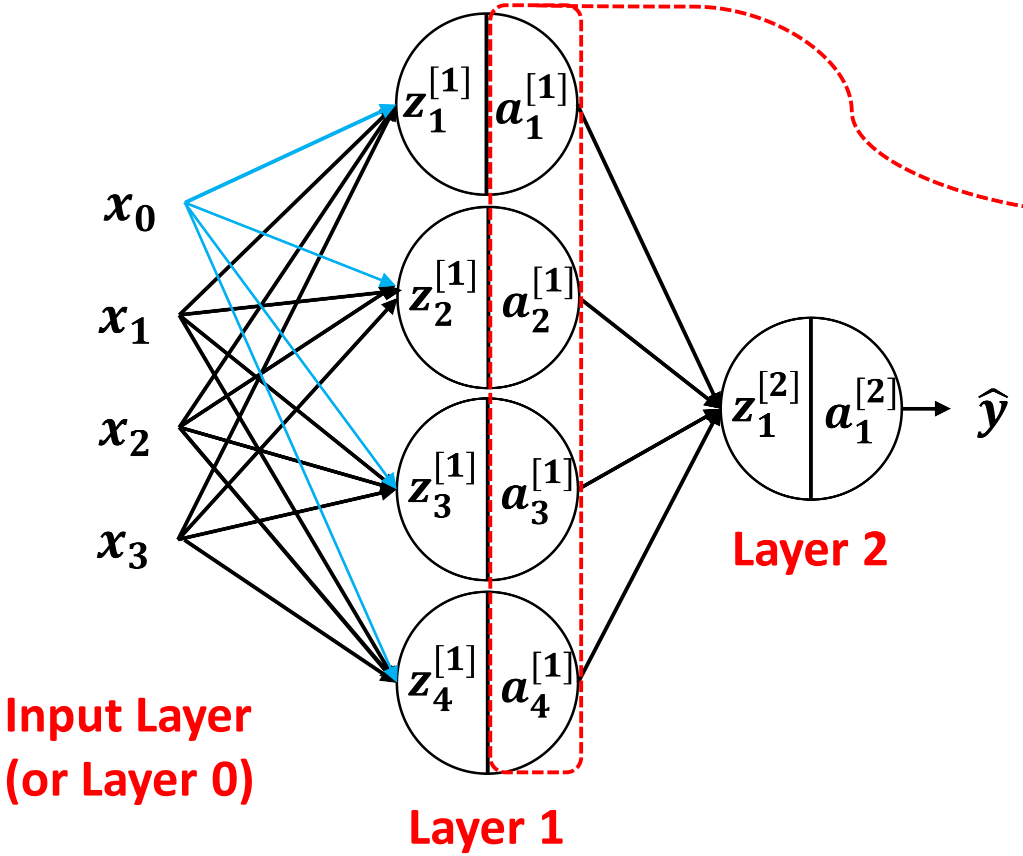
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

Vectorizing Input and All Variables

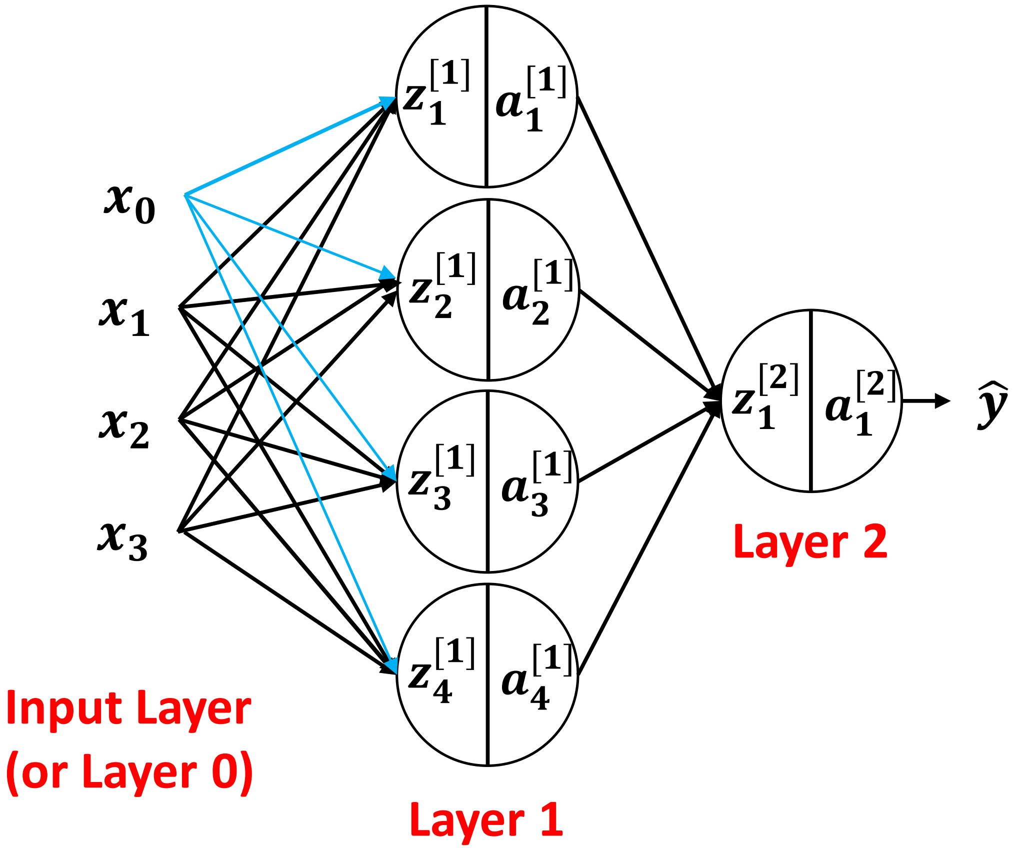
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{a}^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

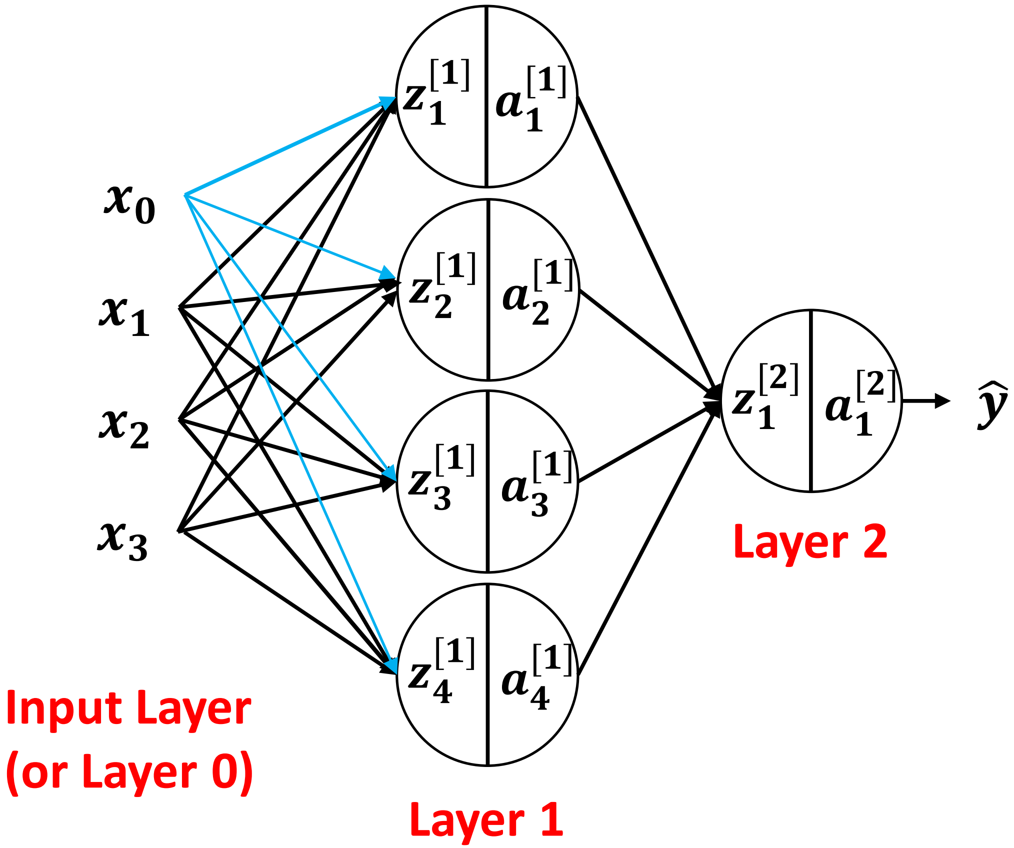


$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

$$= \begin{bmatrix} w_{11}^{[1]} & w_{21}^{[1]} & w_{31}^{[1]} \\ w_{12}^{[1]} & w_{22}^{[1]} & w_{32}^{[1]} \\ w_{13}^{[1]} & w_{23}^{[1]} & w_{33}^{[1]} \\ w_{14}^{[1]} & w_{24}^{[1]} & w_{34}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_1^{[1]} \\ \boldsymbol{b}_2^{[1]} \\ \boldsymbol{b}_3^{[1]} \\ \boldsymbol{b}_4^{[1]} \end{bmatrix}$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

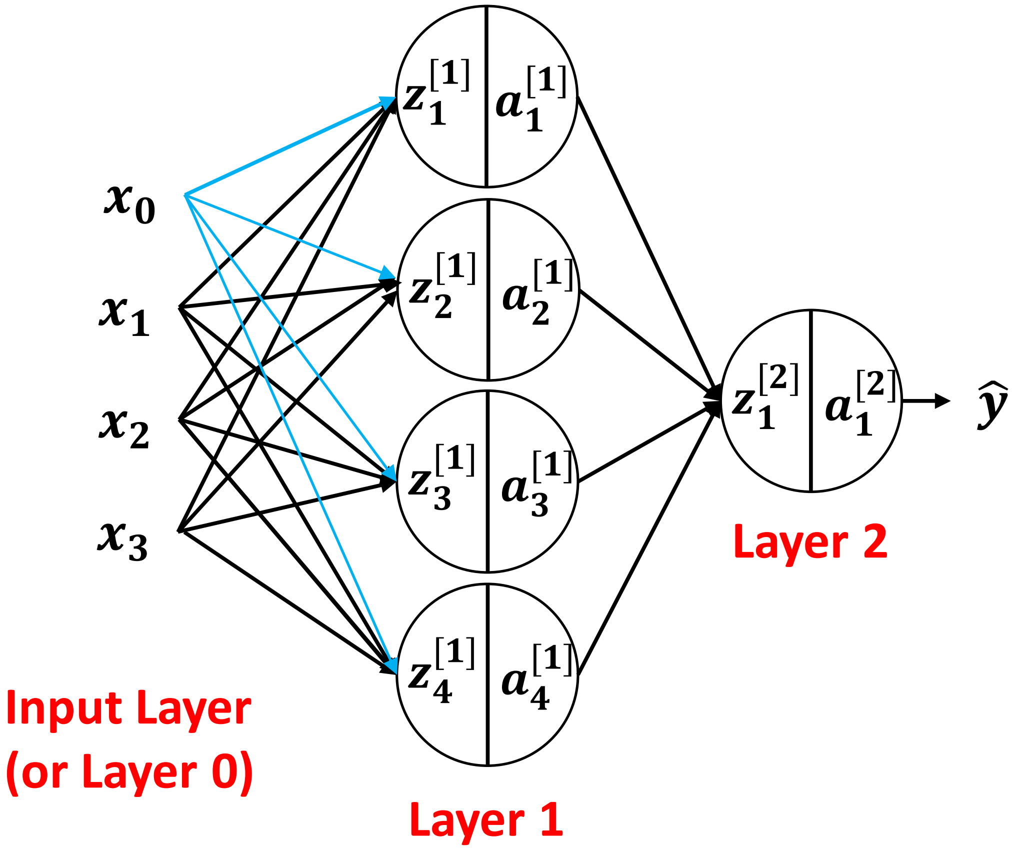


$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



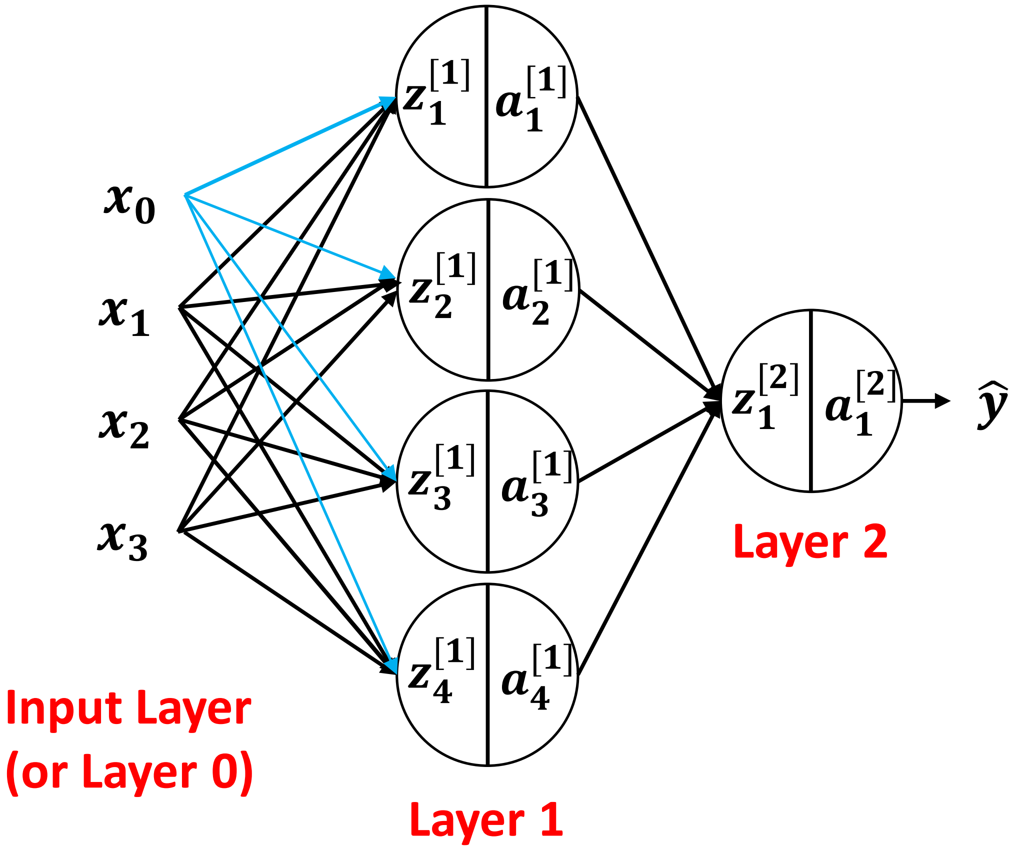
$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]} \end{bmatrix}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

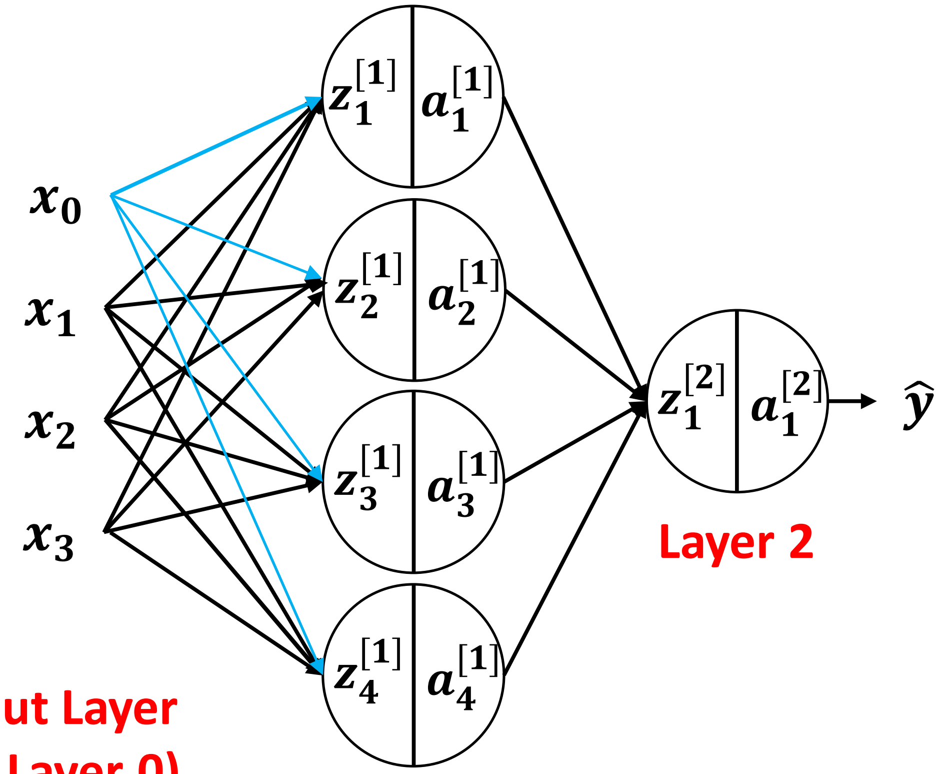
$$= \begin{bmatrix} \boxed{w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]}} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]} \end{bmatrix}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

$\boldsymbol{z}_1^{[1]}$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

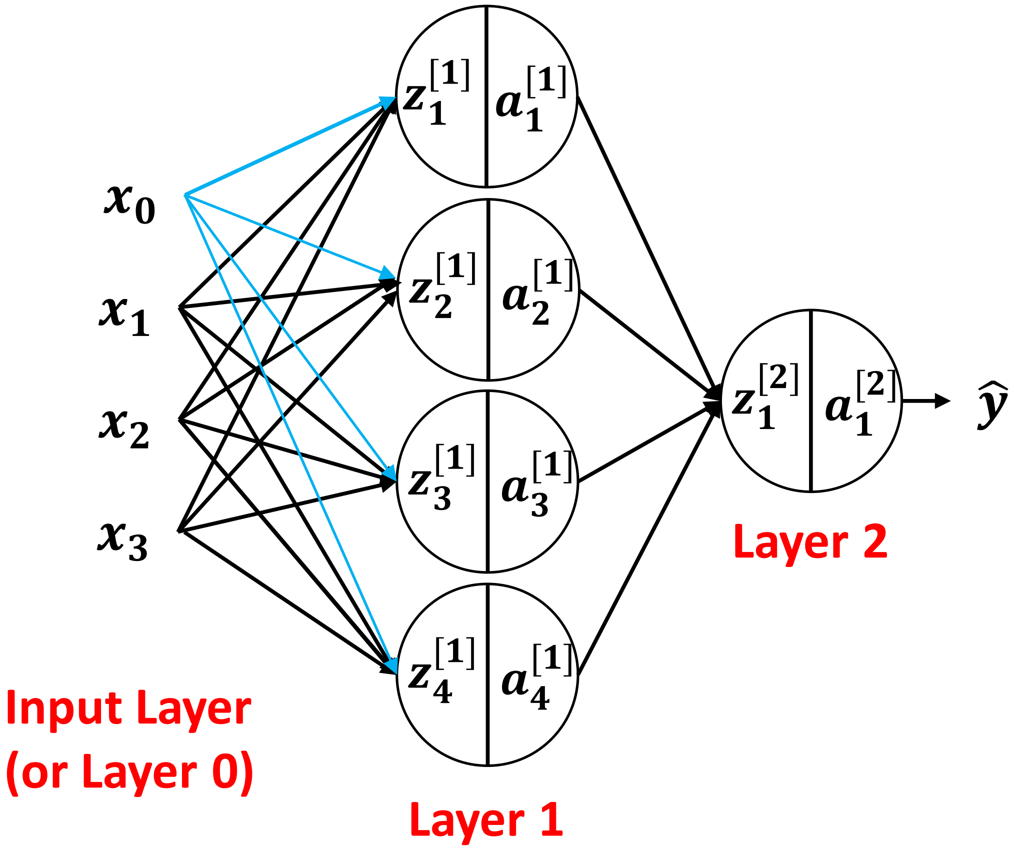
$$= \begin{bmatrix} \boxed{w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]}} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]} \end{bmatrix}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

$$\boldsymbol{a}_1^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}_1^{[1]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

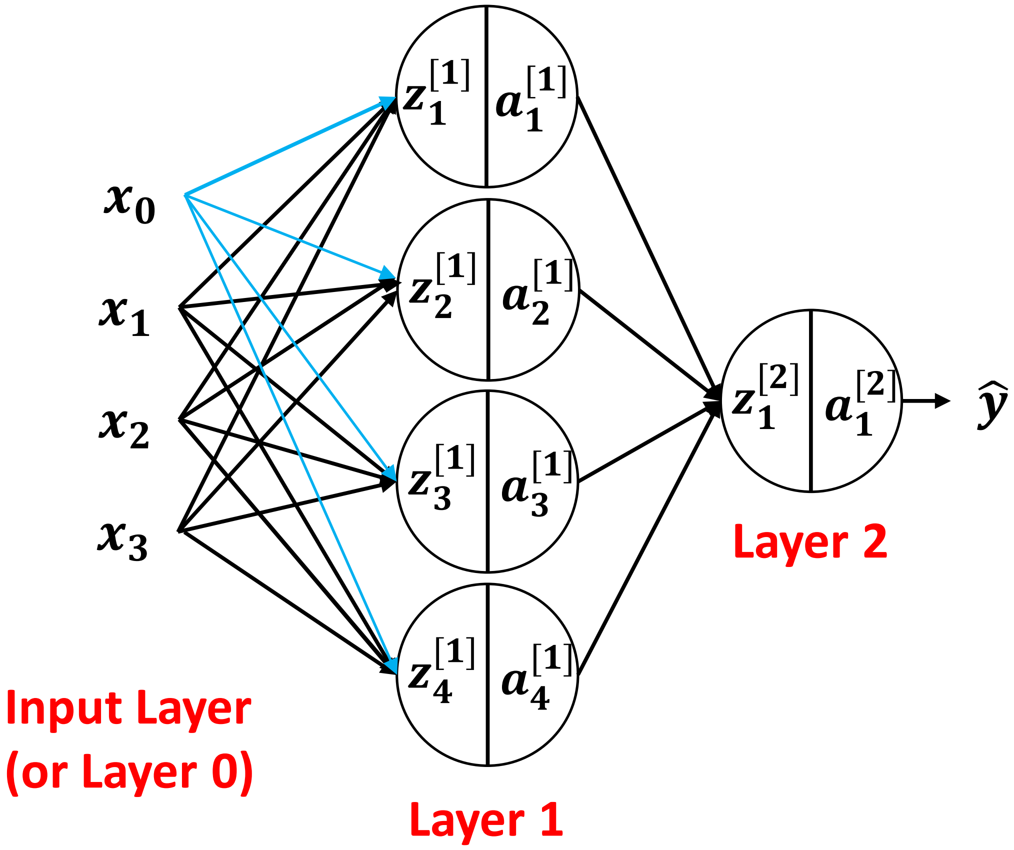
$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} \\ \boxed{w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]}} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]} \end{bmatrix}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

$\boldsymbol{z}_2^{[1]}$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

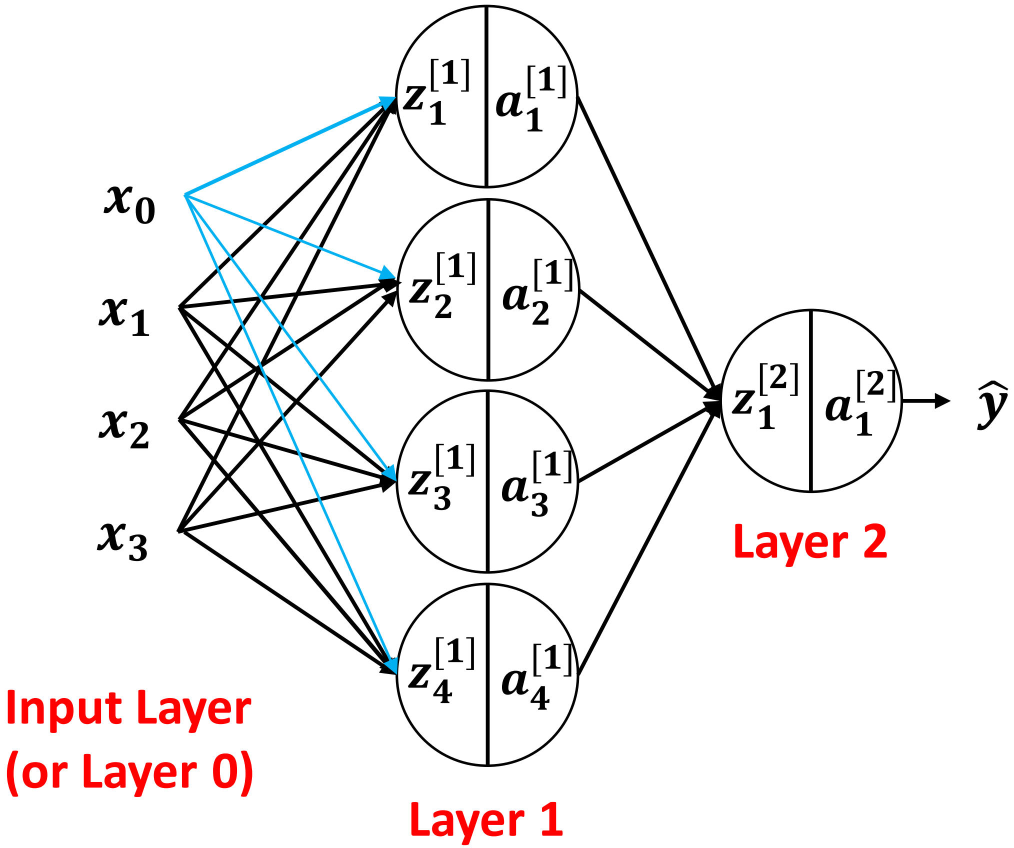
$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} \\ \boxed{w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]}} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]} \end{bmatrix}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

$$\boldsymbol{a}_2^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}_2^{[1]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

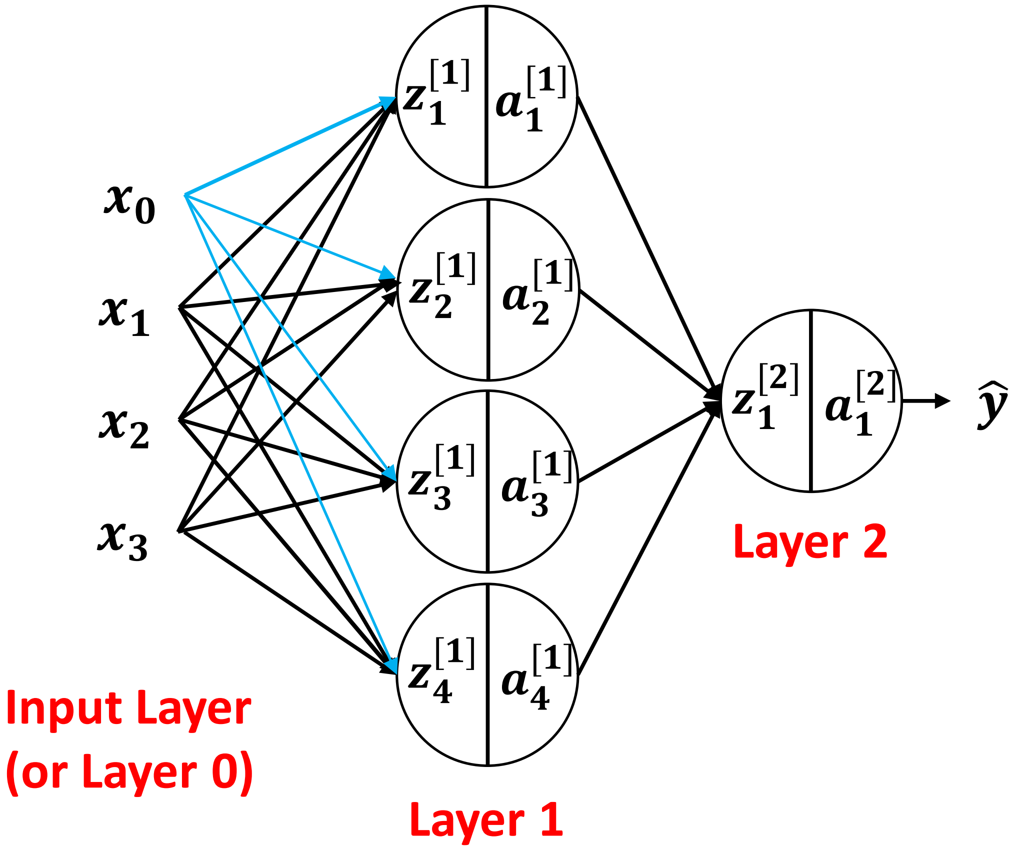
$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]} \end{bmatrix}$$

$$\boldsymbol{z}_3^{[1]}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

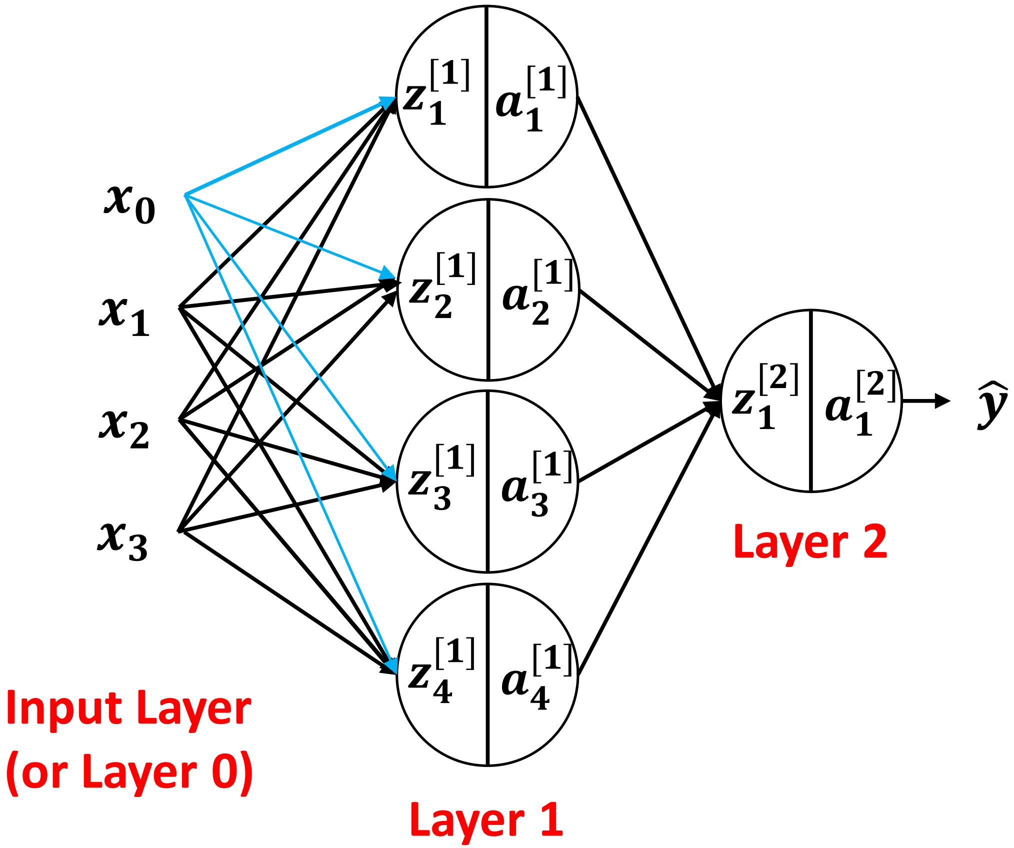
$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]} \end{bmatrix}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

$$\boldsymbol{a}_3^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}_3^{[1]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

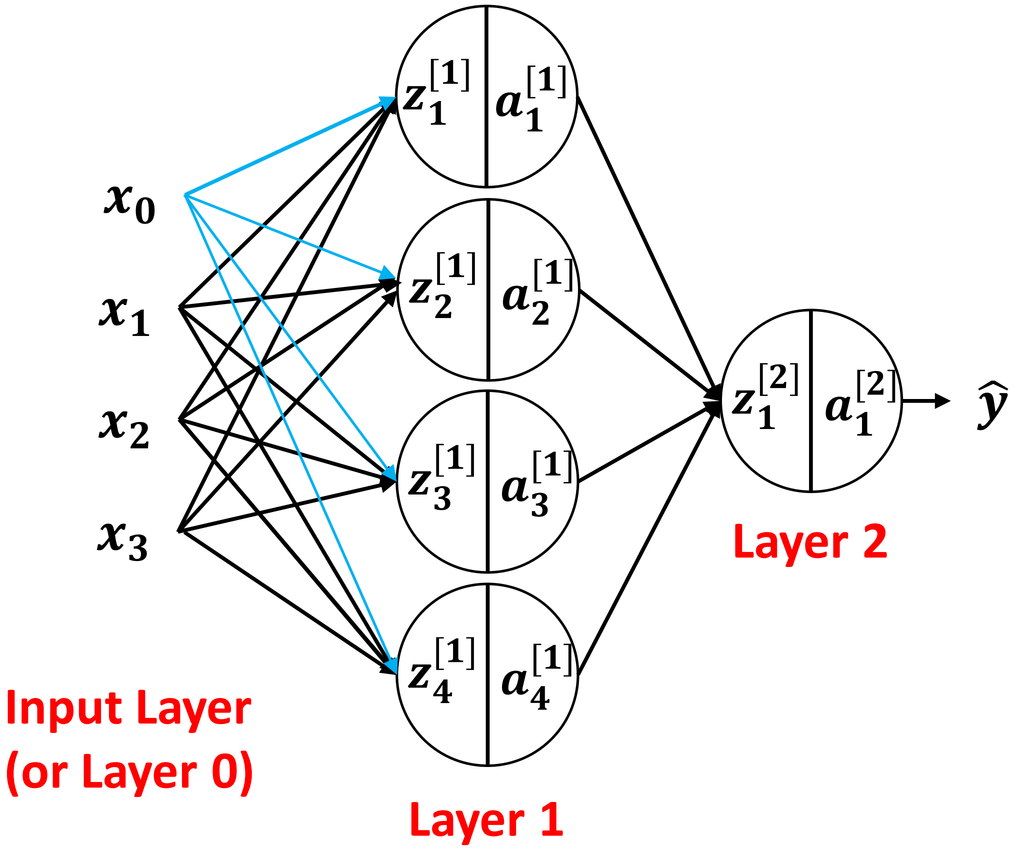
$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ \boxed{w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]}} \end{bmatrix}$$

$$\boldsymbol{z}_4^{[1]}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

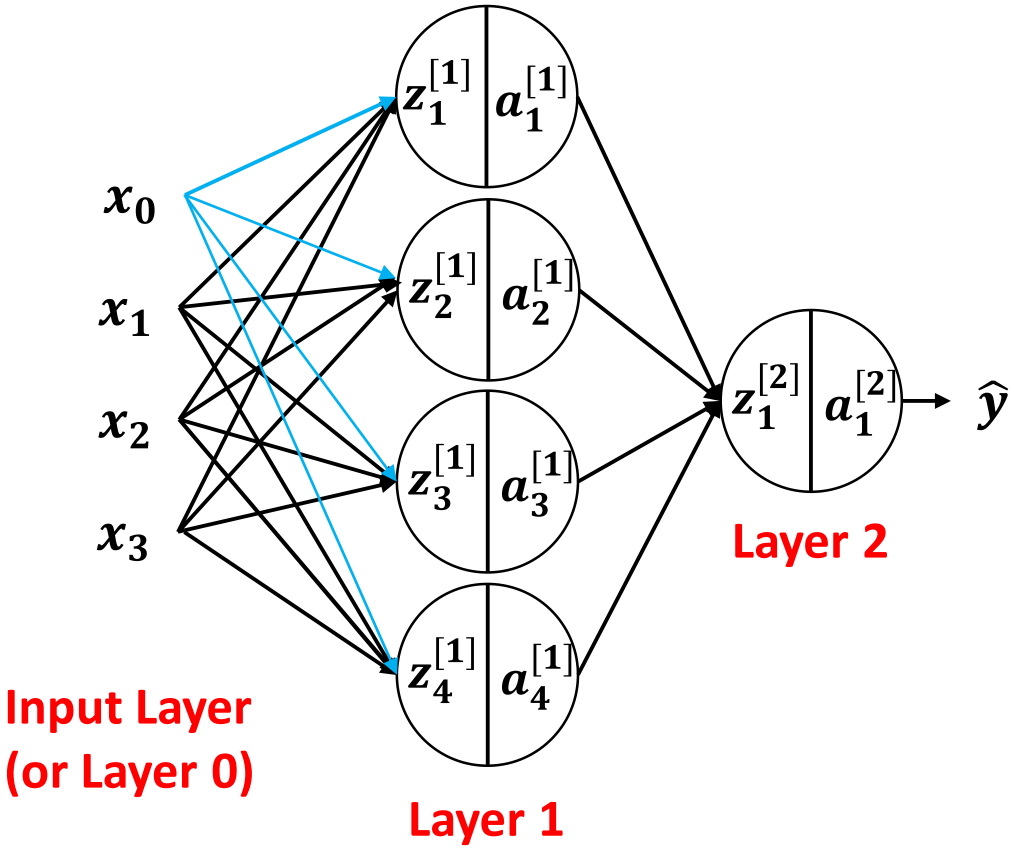
$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + b_1^{[1]} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + b_3^{[1]} \\ \boxed{w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + b_4^{[1]}} \end{bmatrix}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

$$\boldsymbol{a}_4^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}_4^{[1]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

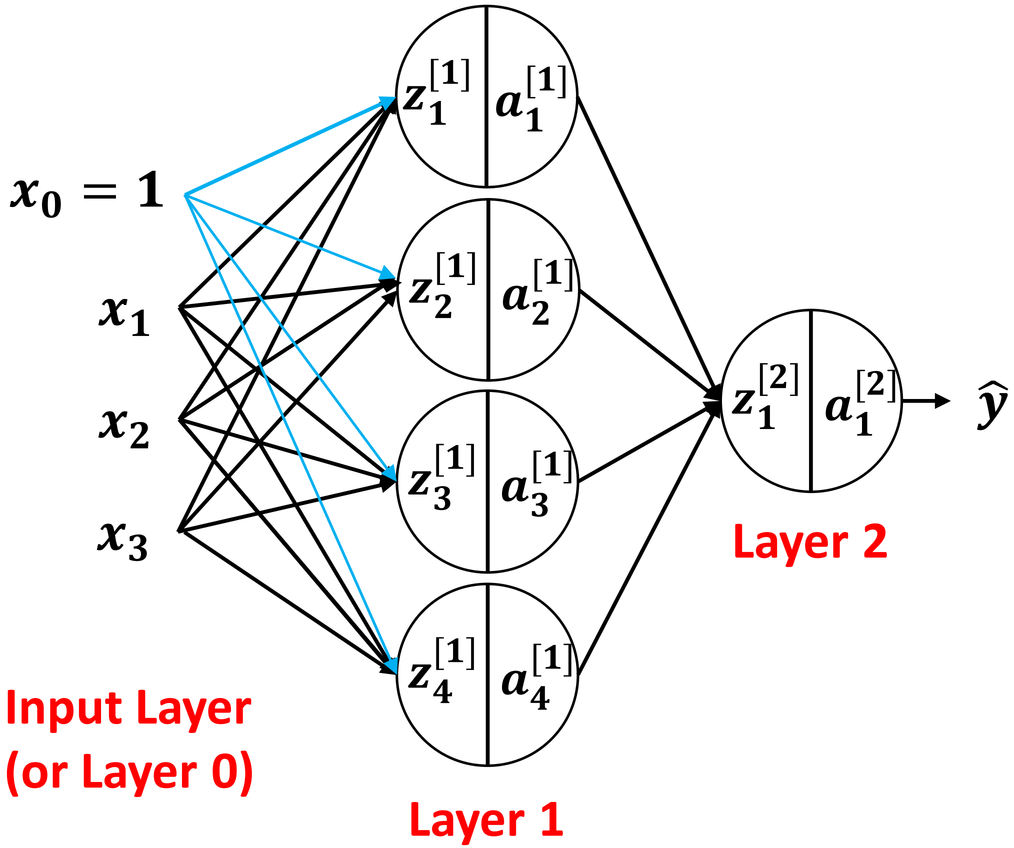


$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + 1 \cdot b_1^{[1]} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + 1 \cdot b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + 1 \cdot b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + 1 \cdot b_4^{[1]} \end{bmatrix}$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



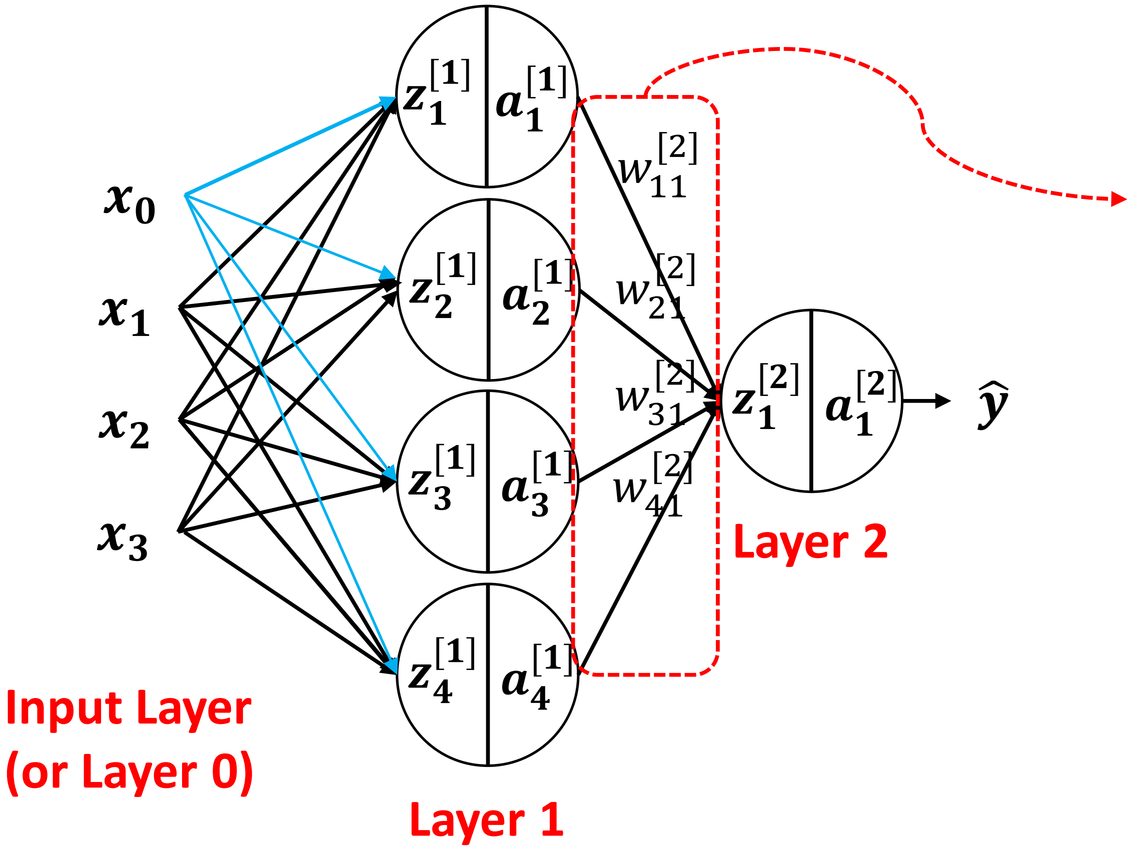
$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

$$= \begin{bmatrix} w_{11}^{[1]}x_1 + w_{21}^{[1]}x_2 + w_{31}^{[1]}x_3 + 1 \cdot b_1^{[1]} \\ w_{12}^{[1]}x_1 + w_{22}^{[1]}x_2 + w_{32}^{[1]}x_3 + 1 \cdot b_2^{[1]} \\ w_{13}^{[1]}x_1 + w_{23}^{[1]}x_2 + w_{33}^{[1]}x_3 + 1 \cdot b_3^{[1]} \\ w_{14}^{[1]}x_1 + w_{24}^{[1]}x_2 + w_{34}^{[1]}x_3 + 1 \cdot b_4^{[1]} \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{b_1^{[1]}} & w_{11}^{[1]} & w_{21}^{[1]} & w_{31}^{[1]} \\ \boldsymbol{b_2^{[1]}} & w_{12}^{[1]} & w_{22}^{[1]} & w_{32}^{[1]} \\ \boldsymbol{b_3^{[1]}} & w_{13}^{[1]} & w_{23}^{[1]} & w_{33}^{[1]} \\ \boldsymbol{b_4^{[1]}} & w_{14}^{[1]} & w_{24}^{[1]} & w_{34}^{[1]} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

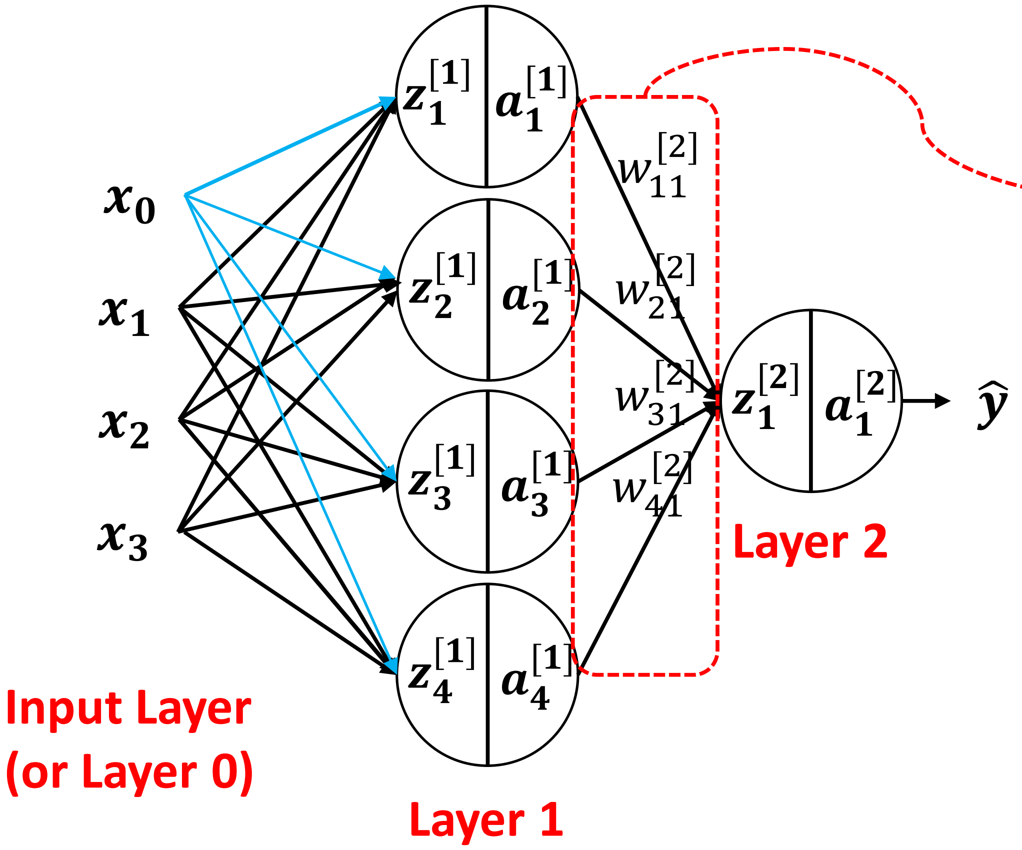


$$\boldsymbol{w}^{[2]} = \begin{bmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \\ w_{31}^{[2]} \\ w_{41}^{[2]} \end{bmatrix}$$

Dimension of $\boldsymbol{w}^{[2]}$= (4, 1)

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

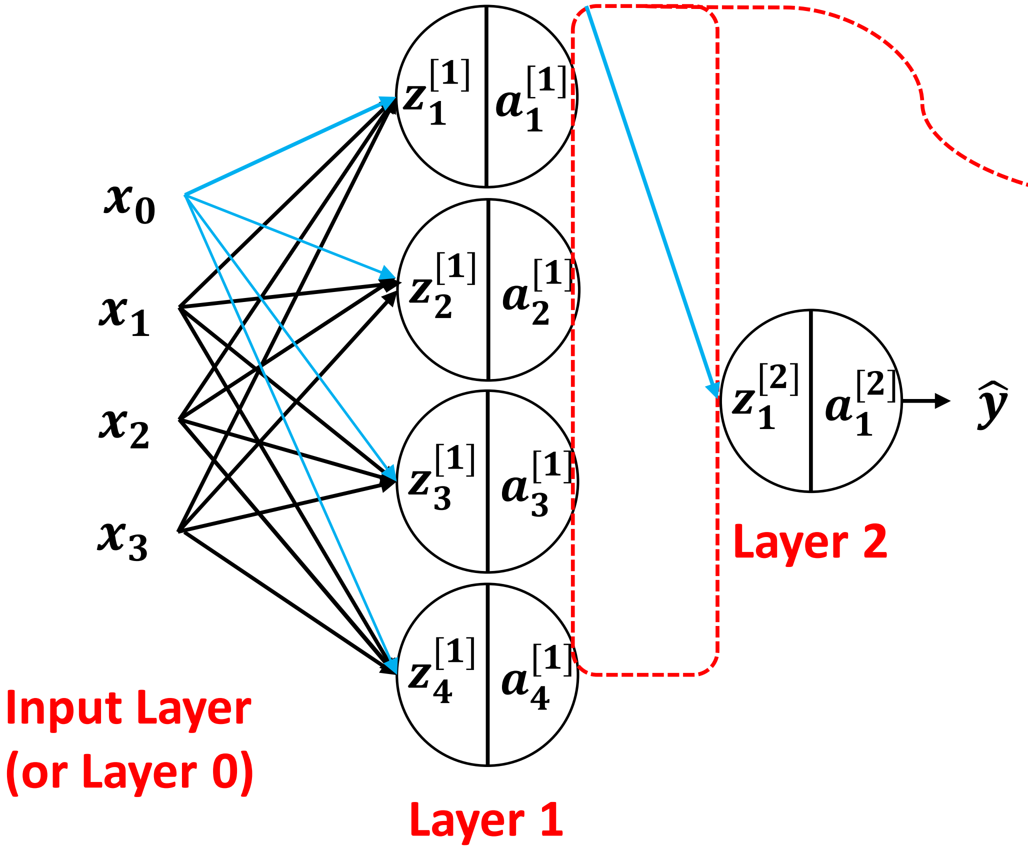


$$\boldsymbol{w}^{[2]^T} = \left[w_{11}^{[2]} \; w_{21}^{[2]} \; w_{31}^{[2]} \; w_{41}^{[2]} \right]$$

Dimension of $\boldsymbol{w}^{[2]^T}$ = (1, 4)

Vectorizing Input and All Variables

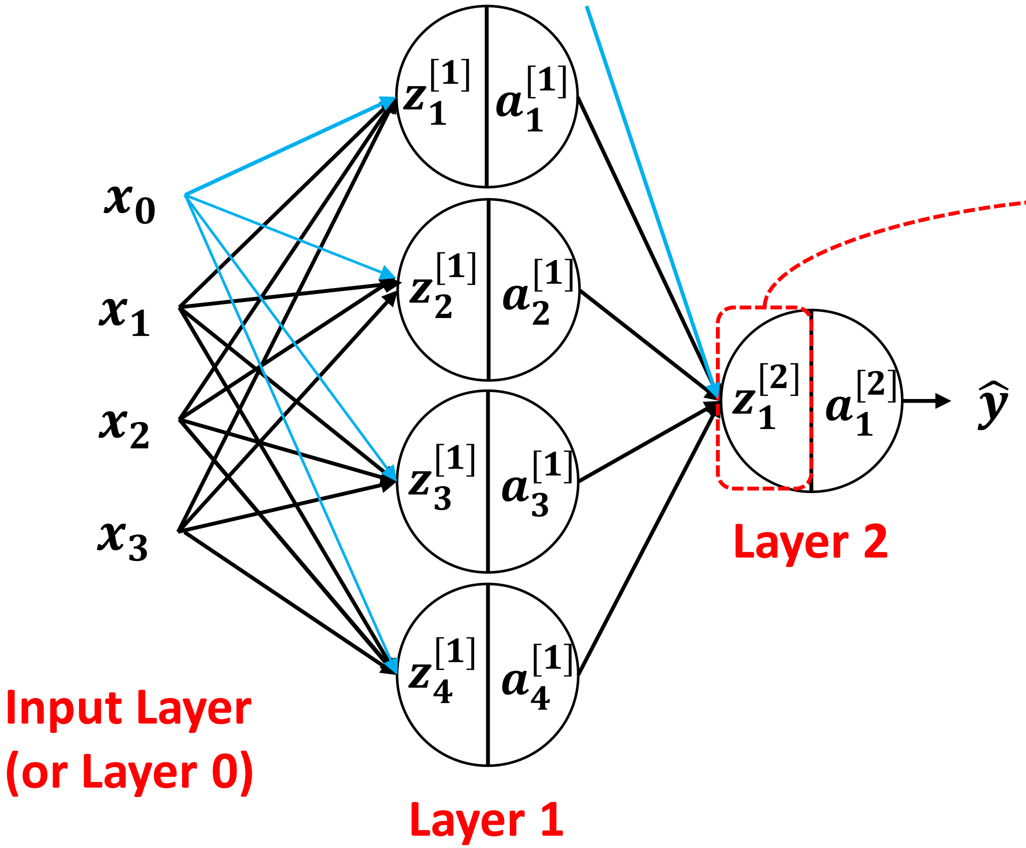
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{b}^{[2]} = \left[\boldsymbol{b}_1^{[2]}\right]$$

Vectorizing Input and All Variables

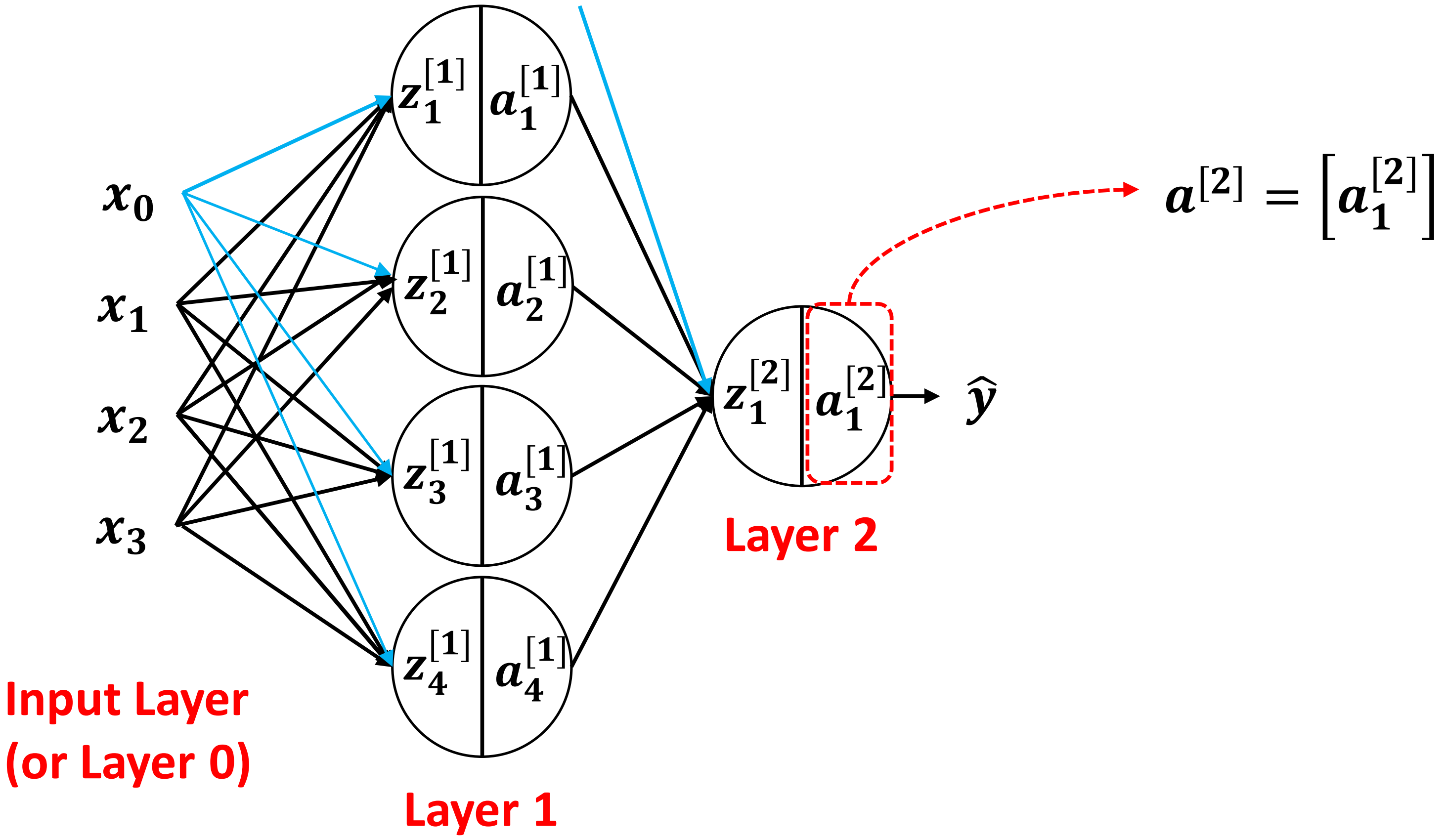
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[2]} = \left[\boldsymbol{z}_1^{[2]} \right]$$

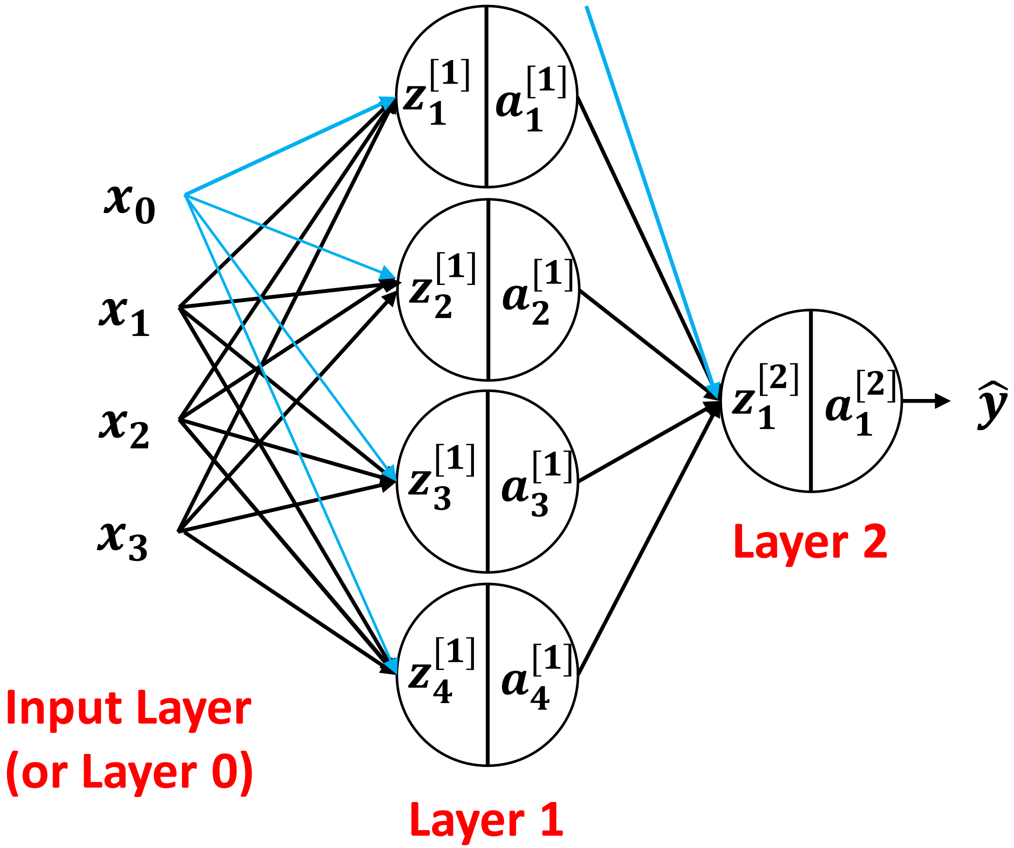
Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



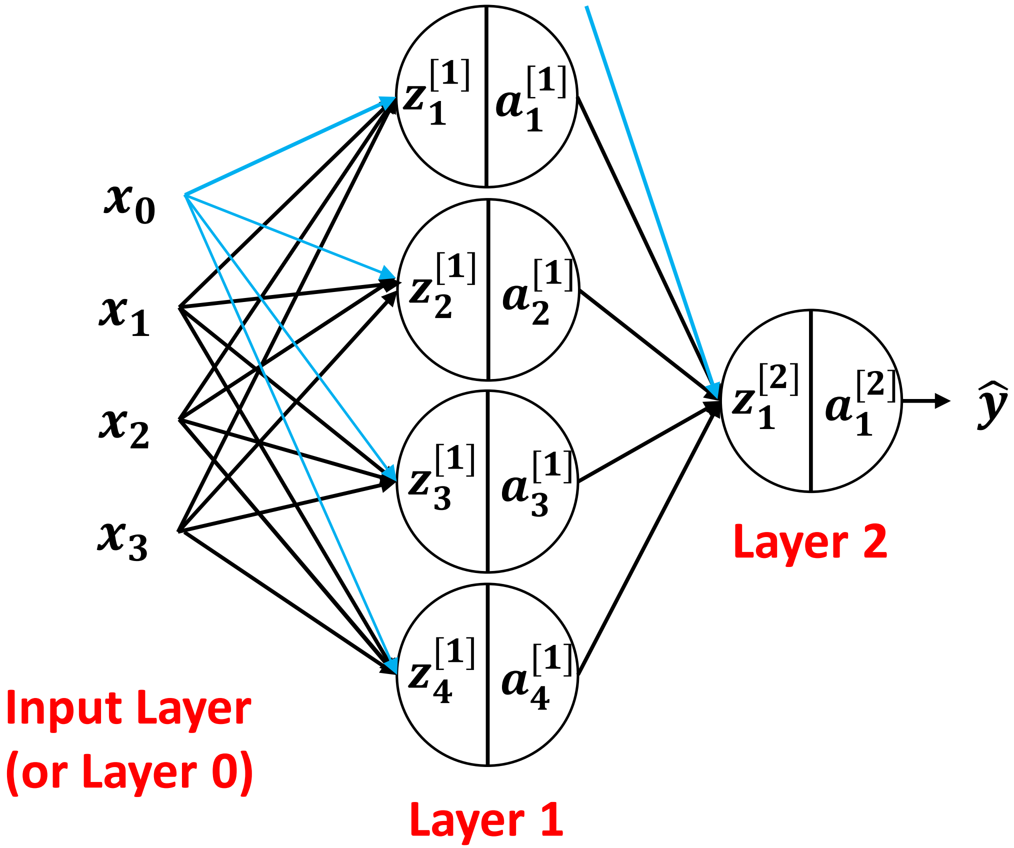
$$\boldsymbol{z}^{[2]} = \boldsymbol{w}^{[2]^T}\boldsymbol{x} + \boldsymbol{b}^{[2]}$$

$$= \boldsymbol{w}^{[2]^T}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$= \begin{bmatrix} w_{11}^{[2]} & w_{21}^{[2]} & w_{31}^{[2]} & w_{41}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \end{bmatrix}$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



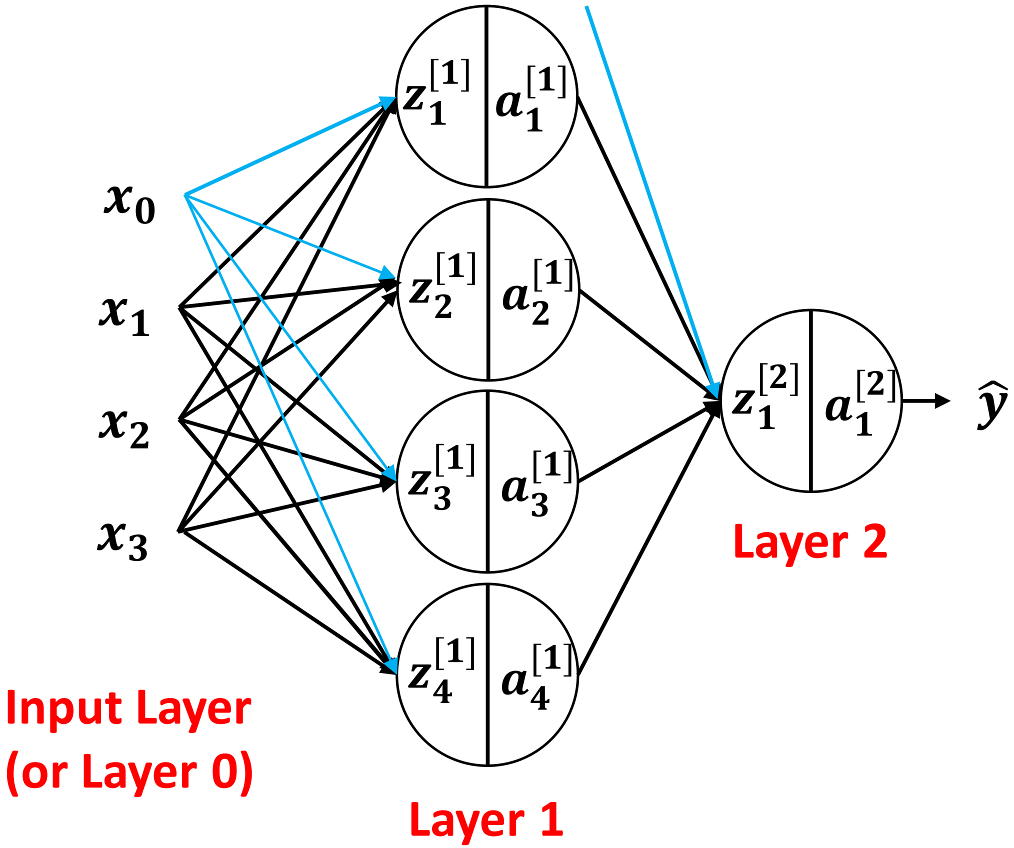
$$\boldsymbol{z}^{[2]} = \boldsymbol{w}^{[2]^T}\boldsymbol{x} + \boldsymbol{b}^{[2]}$$

$$= \boldsymbol{w}^{[2]^T}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$= \left[w_{11}^{[2]}a_1^{[1]} + w_{21}^{[2]}a_2^{[1]} + w_{31}^{[2]}a_3^{[1]} + w_{41}^{[2]}a_4^{[1]}\right] + \left[b_1^{[2]}\right]$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[2]} = \boldsymbol{w}^{[2]^T}\boldsymbol{x} + \boldsymbol{b}^{[2]}$$

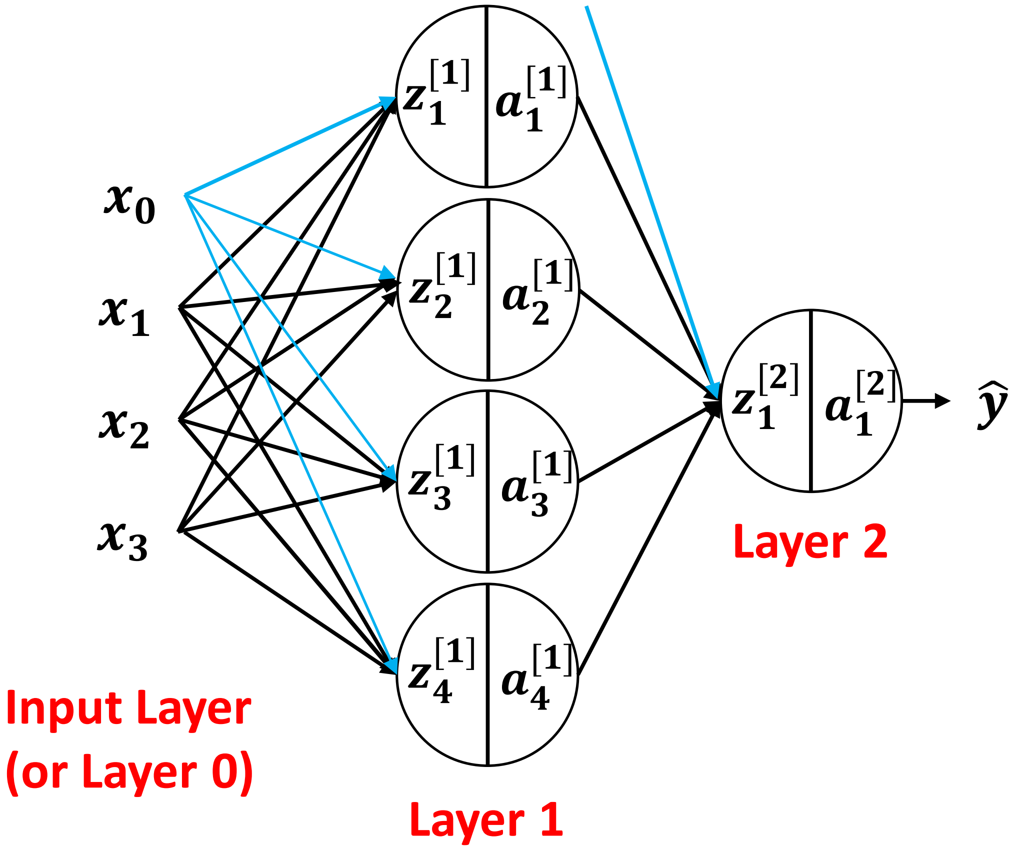
$$= \boldsymbol{w}^{[2]^T}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$= \left[w_{11}^{[2]}a_1^{[1]} + w_{21}^{[2]}a_2^{[1]} + w_{31}^{[2]}a_3^{[1]} + w_{41}^{[2]}a_4^{[1]} + b_1^{[2]}\right]$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[2]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[2]} = \boldsymbol{w}^{[2]^T}\boldsymbol{x} + \boldsymbol{b}^{[2]}$$

$$= \boldsymbol{w}^{[2]^T}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

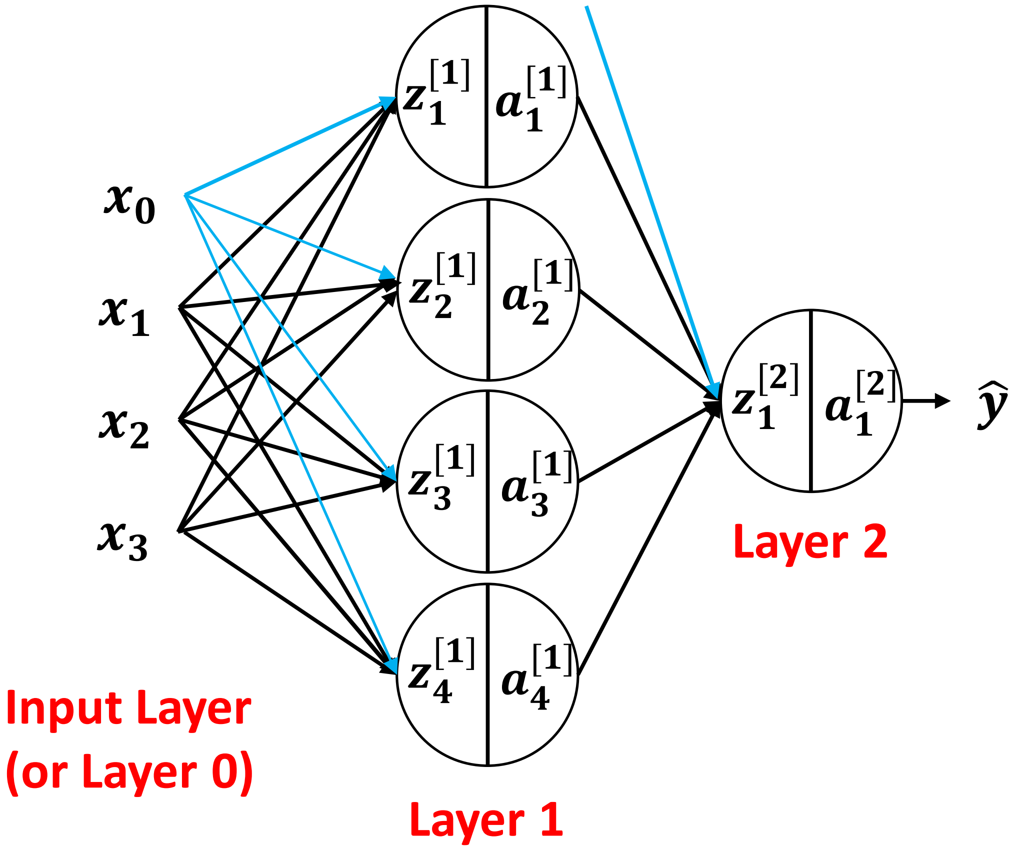
$$= \left[\boxed{w_{11}^{[2]}a_1^{[1]} + w_{21}^{[2]}a_2^{[1]} + w_{31}^{[2]}a_3^{[1]} + w_{41}^{[2]}a_4^{[1]} + b_1^{[2]}}\right]$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[2]})$$

$$\boldsymbol{z}_1^{[2]}$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[2]} = \boldsymbol{w}^{[2]^T}\boldsymbol{x} + \boldsymbol{b}^{[2]}$$

$$= \boldsymbol{w}^{[2]^T}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

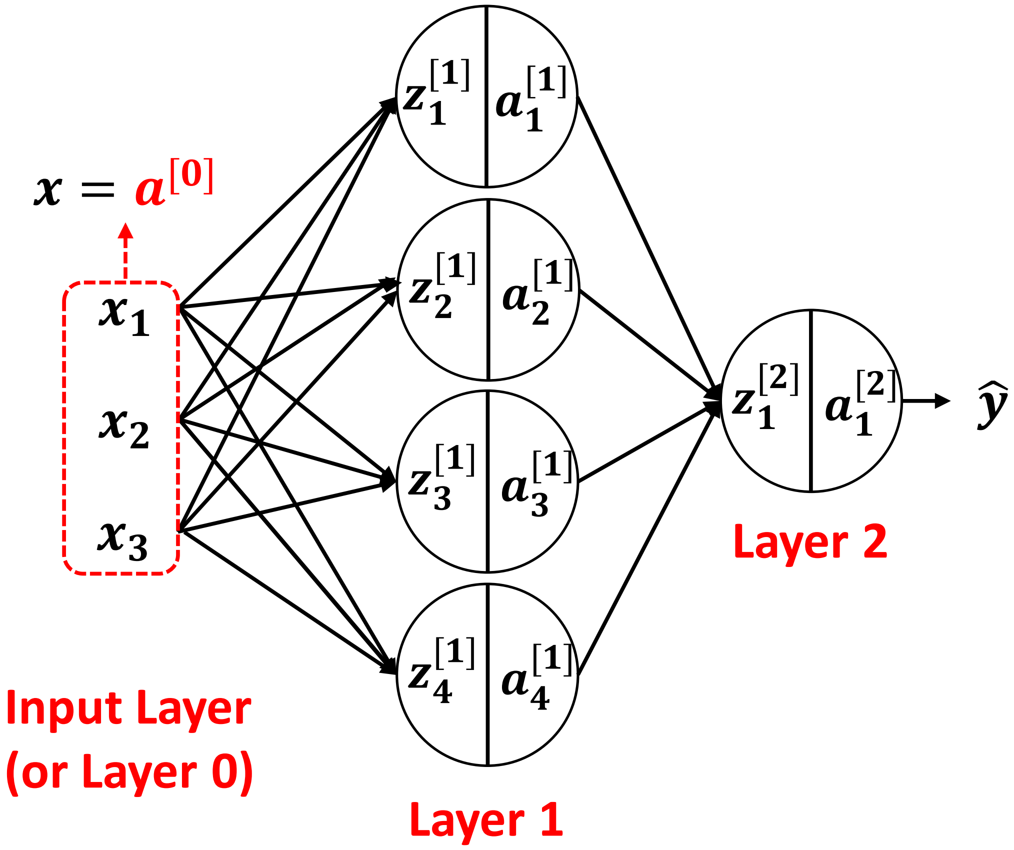
$$= \left[\boxed{w_{11}^{[2]}a_1^{[1]} + w_{21}^{[2]}a_2^{[1]} + w_{31}^{[2]}a_3^{[1]} + w_{41}^{[2]}a_4^{[1]} + b_1^{[2]}}\right]$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[2]})$$

$$\boldsymbol{a}_1^{[2]} = \boldsymbol{\sigma}(\boldsymbol{z}_1^{[2]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{x} + \boldsymbol{b}^{[1]}$$

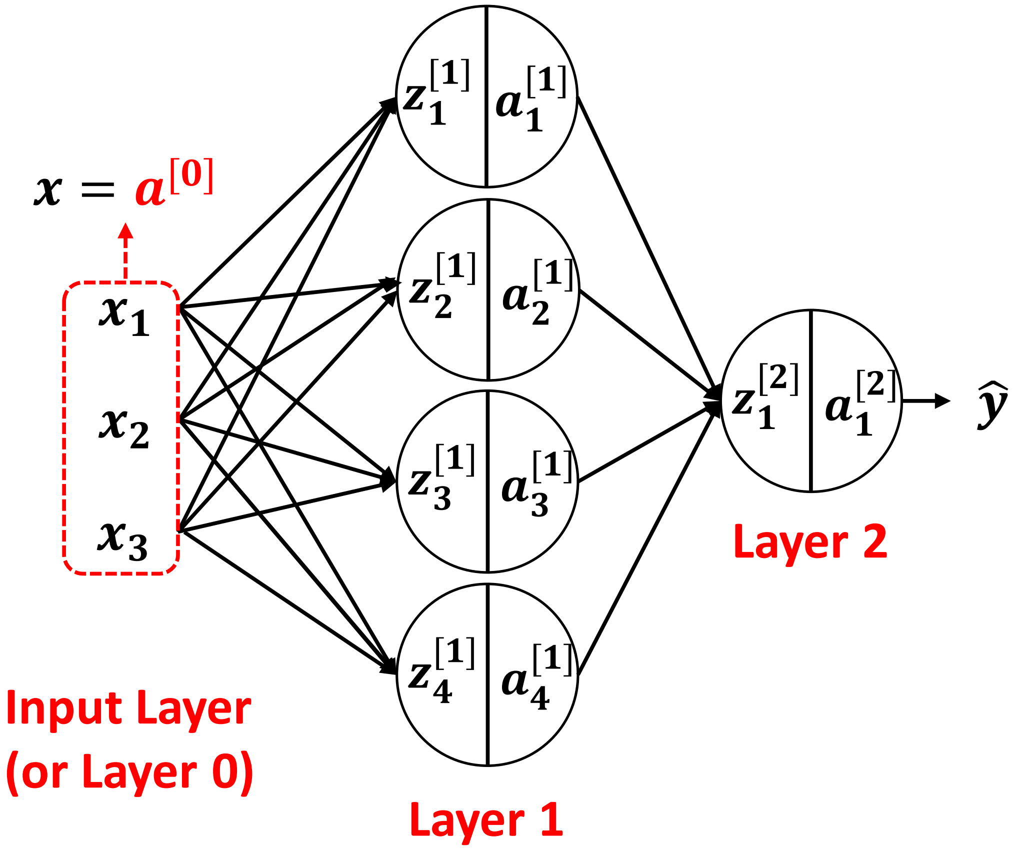
$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

$$\boldsymbol{z}^{[2]} = \boldsymbol{w}^{[2]^T}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[2]})$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{z}^{[1]} = \boldsymbol{w}^{[1]^T}\boldsymbol{a}^{[0]} + \boldsymbol{b}^{[1]}$$

$$\boldsymbol{a}^{[1]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[1]})$$

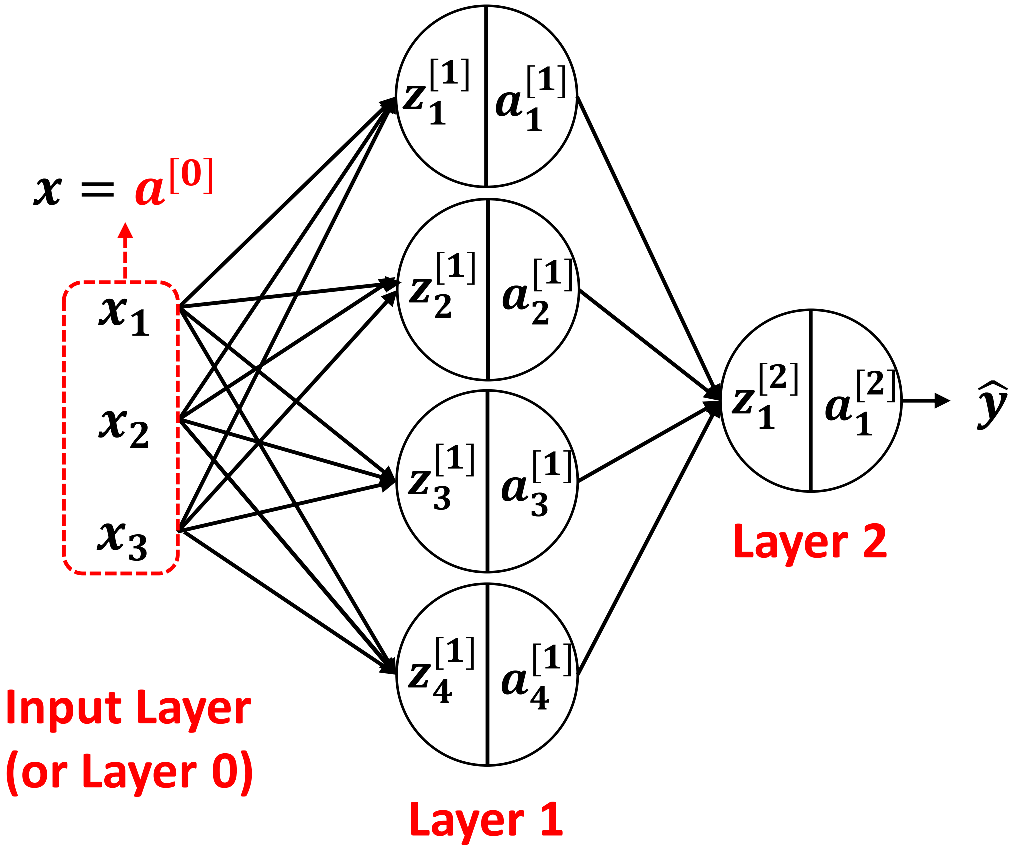
$$\boldsymbol{z}^{[2]} = \boldsymbol{w}^{[2]^T}\boldsymbol{a}^{[1]} + \boldsymbol{b}^{[2]}$$

$$\boldsymbol{a}^{[2]} = \boldsymbol{\sigma}(\boldsymbol{z}^{[2]})$$

But, this assumes only 1 training example!

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$z^{[1]} = w^{[1]^T} a^{[0]} + b^{[1]}$$

$$a^{[1]} = \sigma(z^{[1]})$$

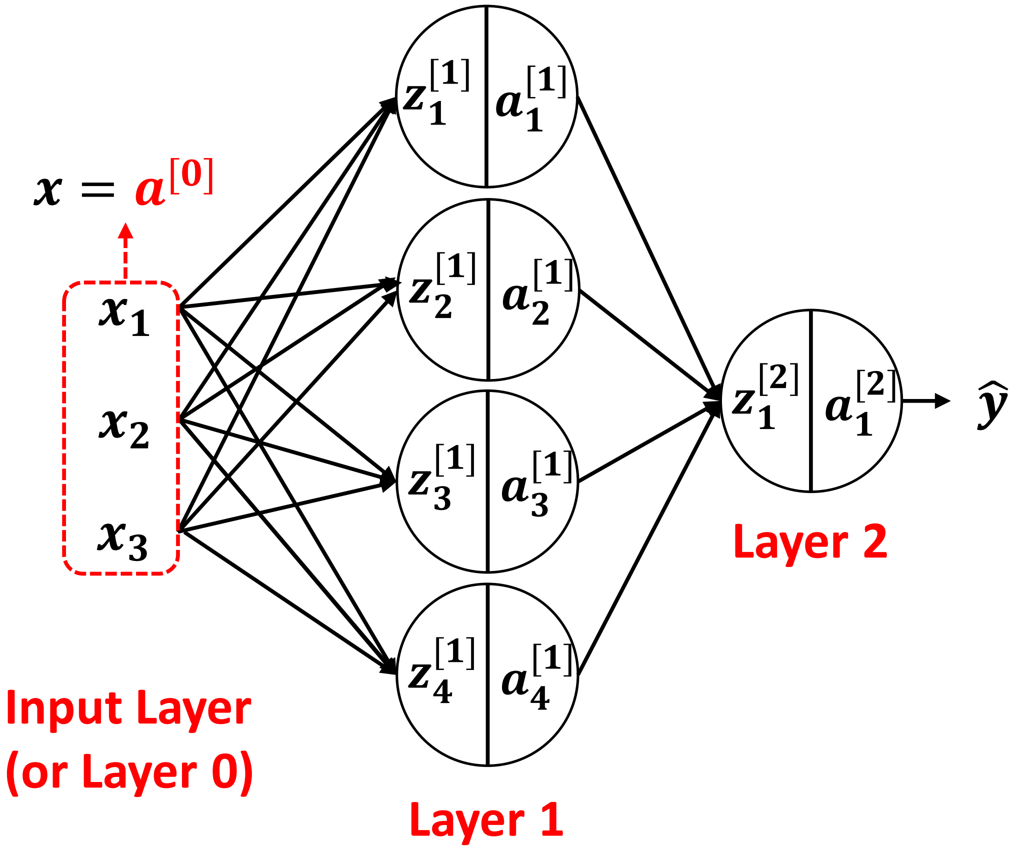
$$z^{[2]} = w^{[2]^T} a^{[1]} + b^{[2]}$$

$$a^{[2]} = \sigma(z^{[2]})$$

How can we account for all the training examples?

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$\boldsymbol{for\ i = 1\ to\ n}$: Assuming $\boldsymbol{n}$ examples

$$z^{[1](i)} = w^{[1]^T(i)} a^{[0](i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma(z^{[1](i)})$$

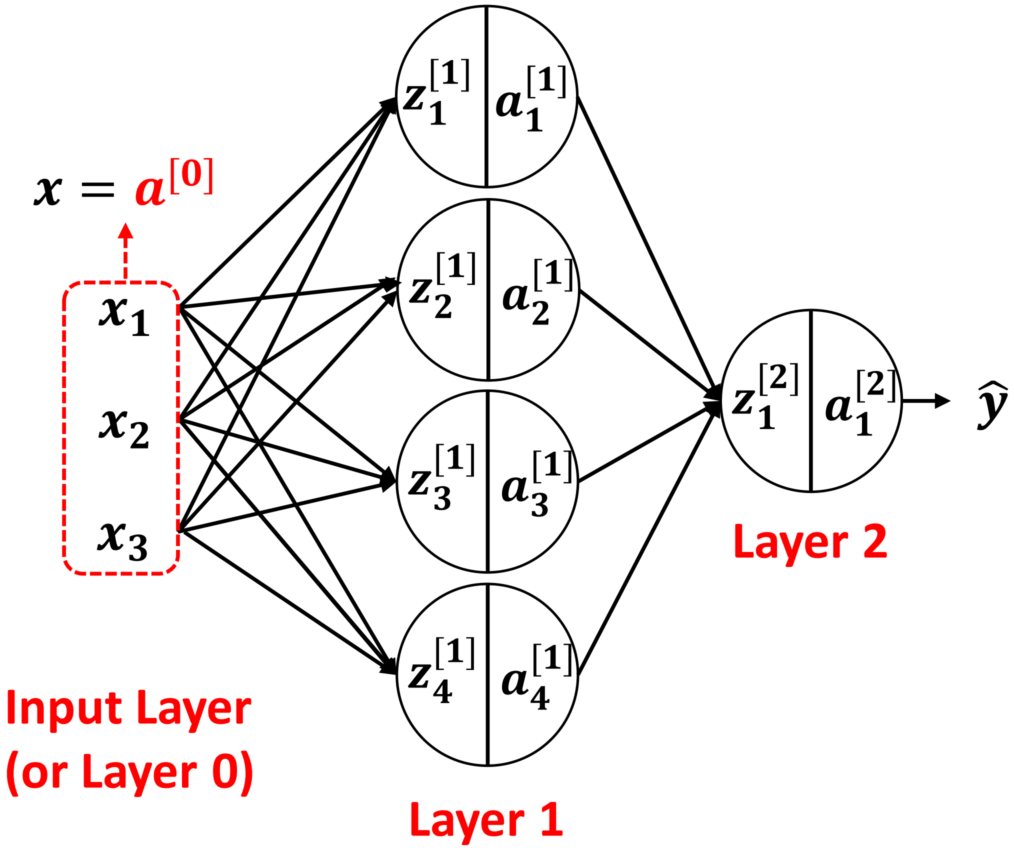
$$z^{[2](i)} = w^{[2]^T(i)} a^{[1](i)} + b^{[2]}$$

$$a^{[2]\boxed{(i)}} = \sigma(z^{[2](i)})$$

Refers to the i^{th} example in the training dataset

Vectorizing Input and All Variables

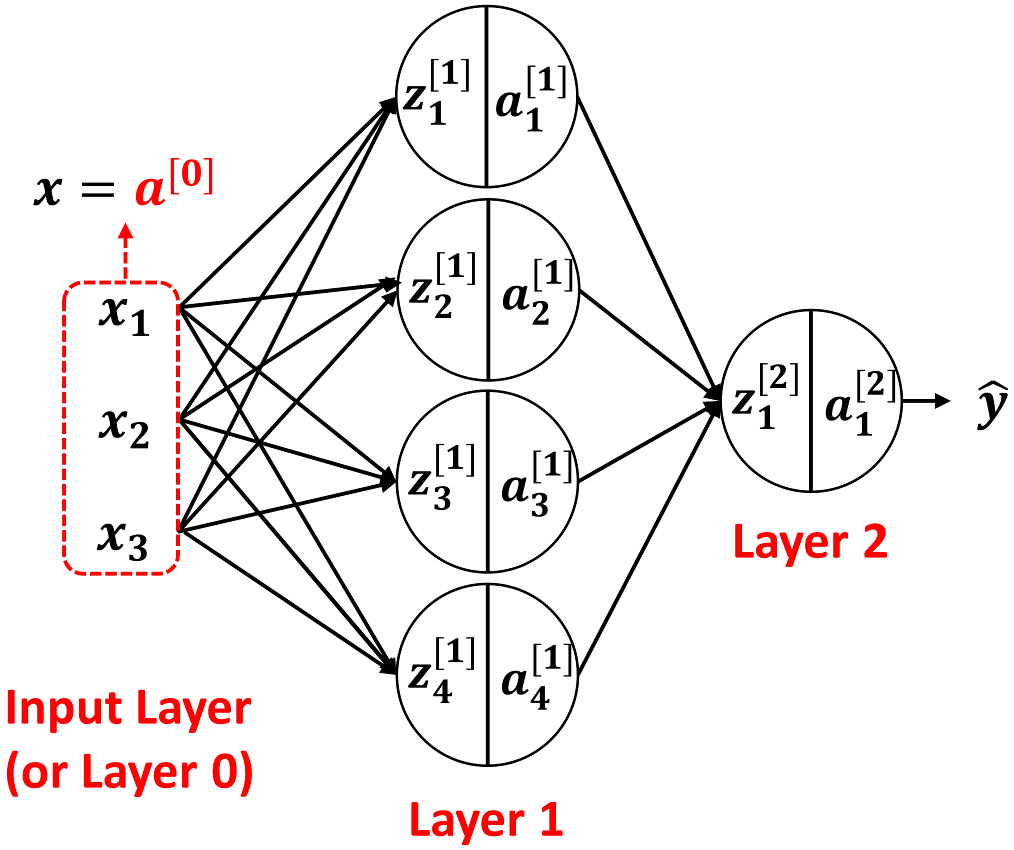
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



But, loops in general slow down programs; hence, it is better to further ***vectorize*** the implementation in order to avoid any loop, whenever possible

Vectorizing Input and All Variables

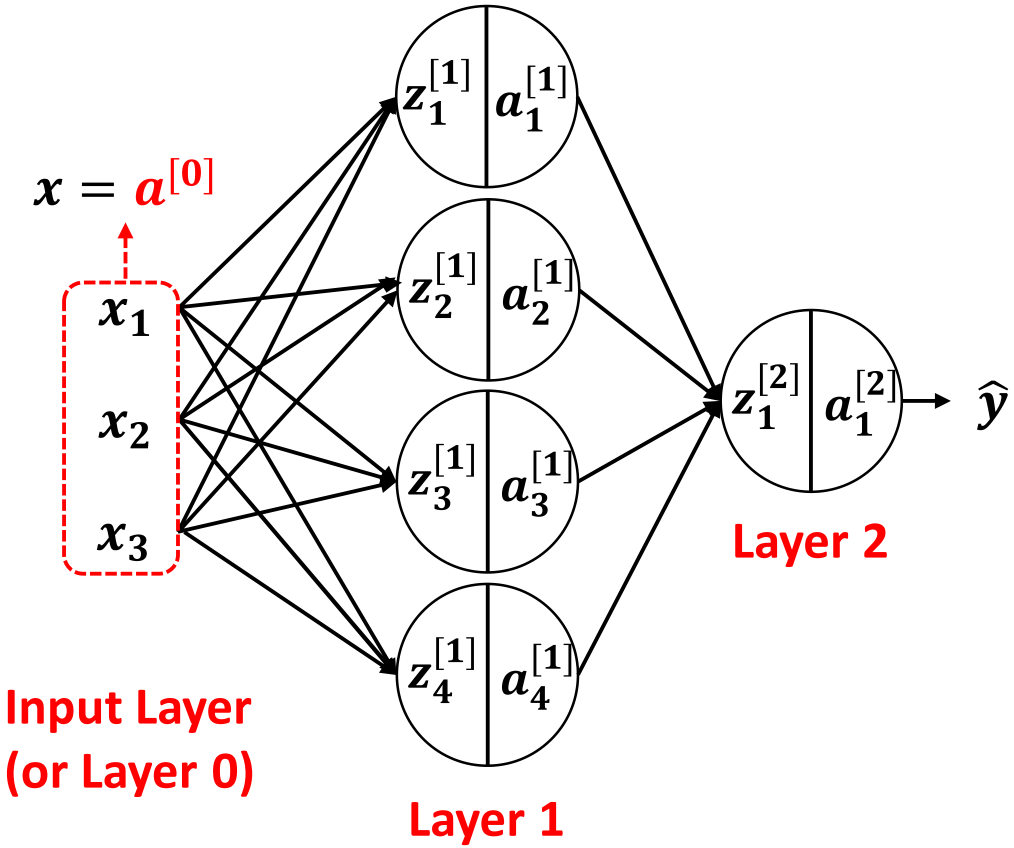
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



To this end, we can simply stack all $\boldsymbol{x}$ vectors (or $\boldsymbol{a}^{[0]}$ vectors), $\boldsymbol{z}$ vectors, and $\boldsymbol{a}$ vectors in different matrices of every layer!

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



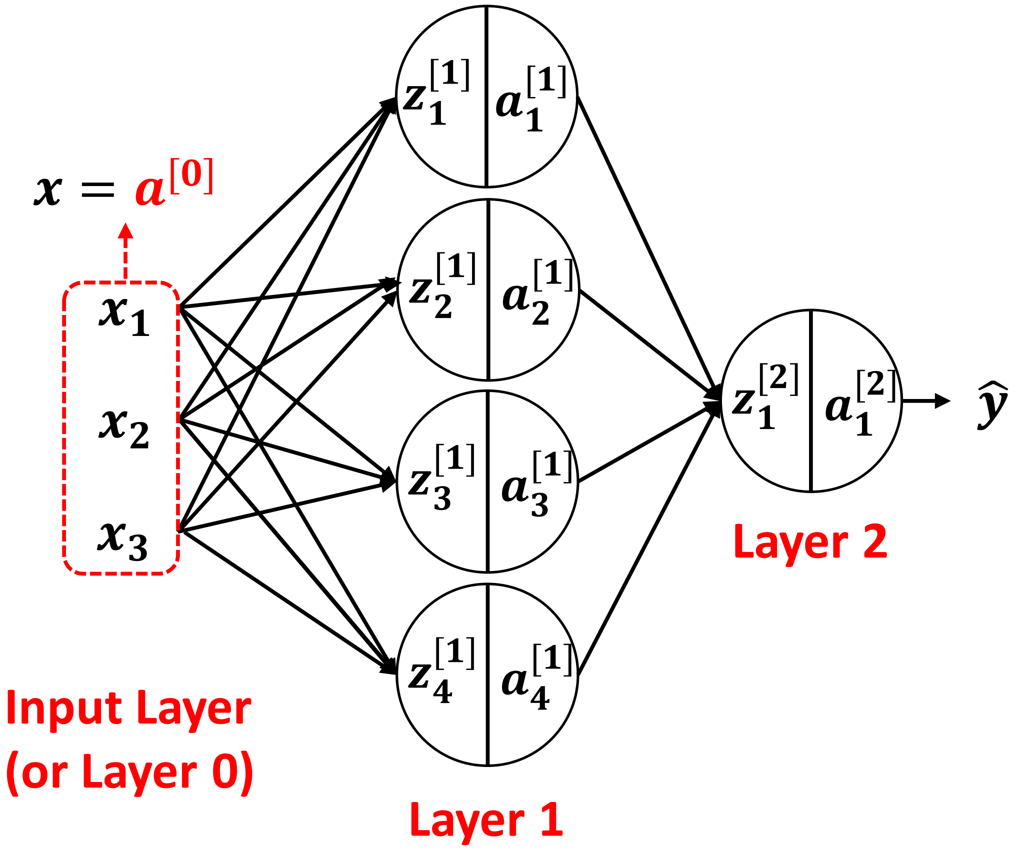
$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & & x_1^{(n)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(n)} \\ x_3^{(1)} & x_3^{(2)} & & x_3^{(n)} \end{bmatrix}$$

Assuming n examples

This vector represents the *first* example in the training dataset

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



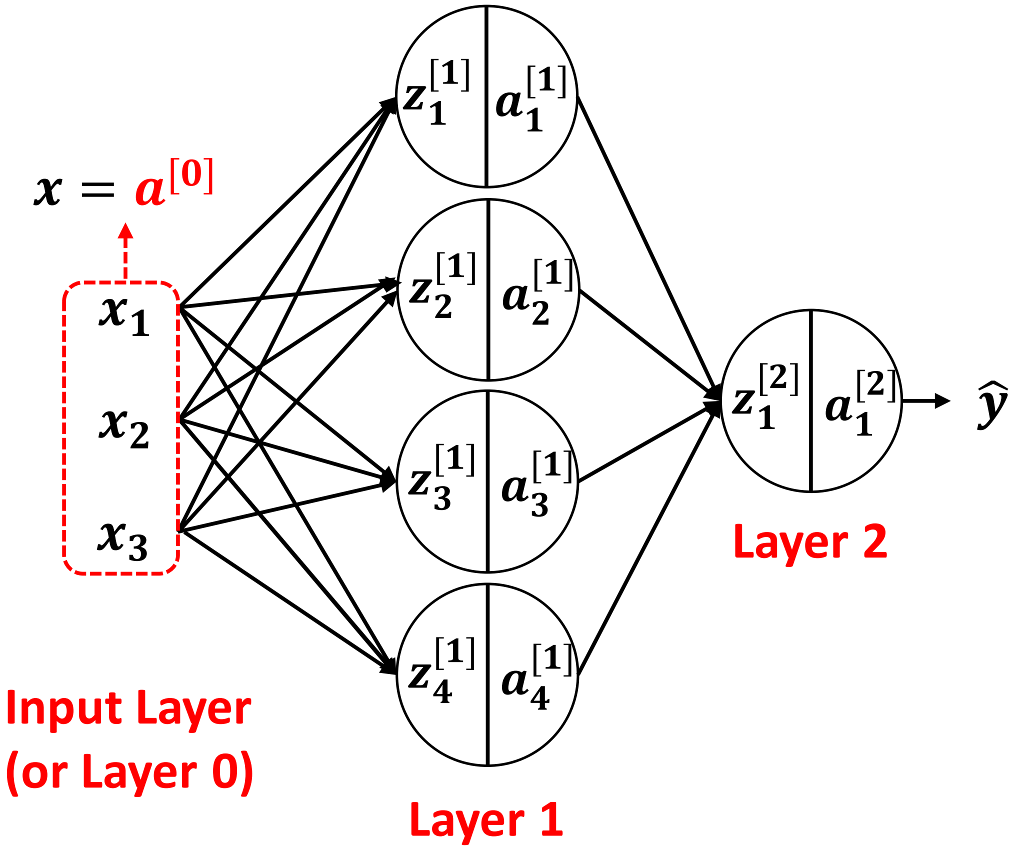
$$X = \begin{bmatrix} x_1^{(1)} & \boxed{x_1^{(2)}} & & x_1^{(n)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(n)} \\ x_3^{(1)} & x_3^{(2)} & & x_3^{(n)} \end{bmatrix}$$

Assuming n examples

This vector represents the *second* example in the training dataset

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



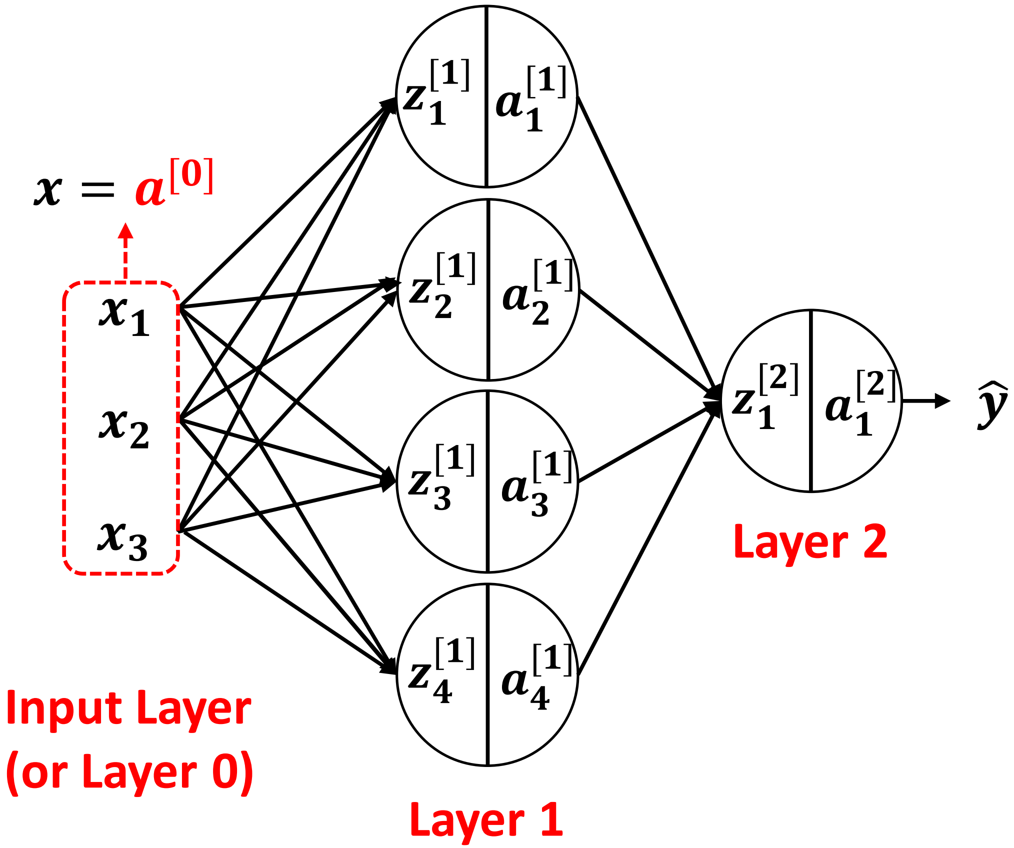
$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & & x_1^{(n)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(n)} \\ x_3^{(1)} & x_3^{(2)} & & x_3^{(n)} \end{bmatrix}$$

Assuming n examples

This vector represents the n^{th} example in the training dataset

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

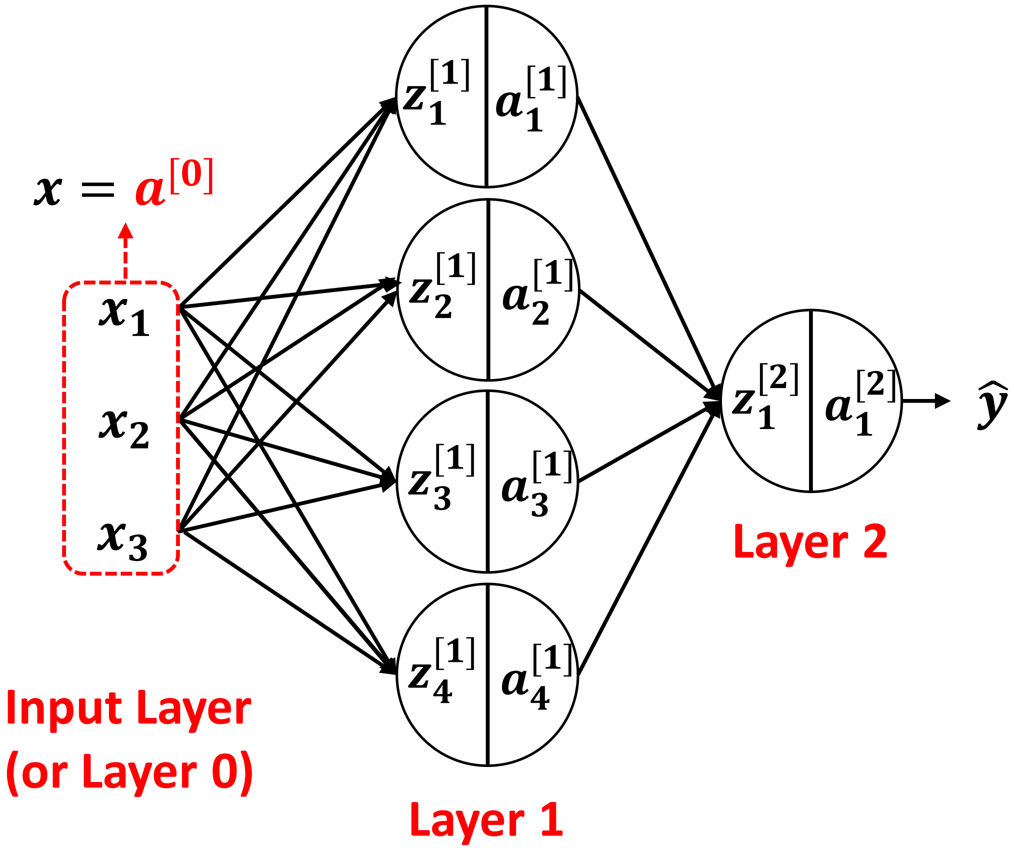


$$X = \underbrace{\begin{bmatrix} \boldsymbol{a}_1^{[0](1)} & \boldsymbol{a}_1^{[0](2)} & & \boldsymbol{a}_1^{[0](n)} \\ \boldsymbol{a}_2^{[0](1)} & \boldsymbol{a}_2^{[0](2)} & \dots & \boldsymbol{a}_2^{[0](n)} \\ \boldsymbol{a}_3^{[0](1)} & \boldsymbol{a}_3^{[0](2)} & & \boldsymbol{a}_3^{[0](n)} \end{bmatrix}}$$

If we denote $\boldsymbol{x}$ as $\boldsymbol{a}^{[0]}$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved

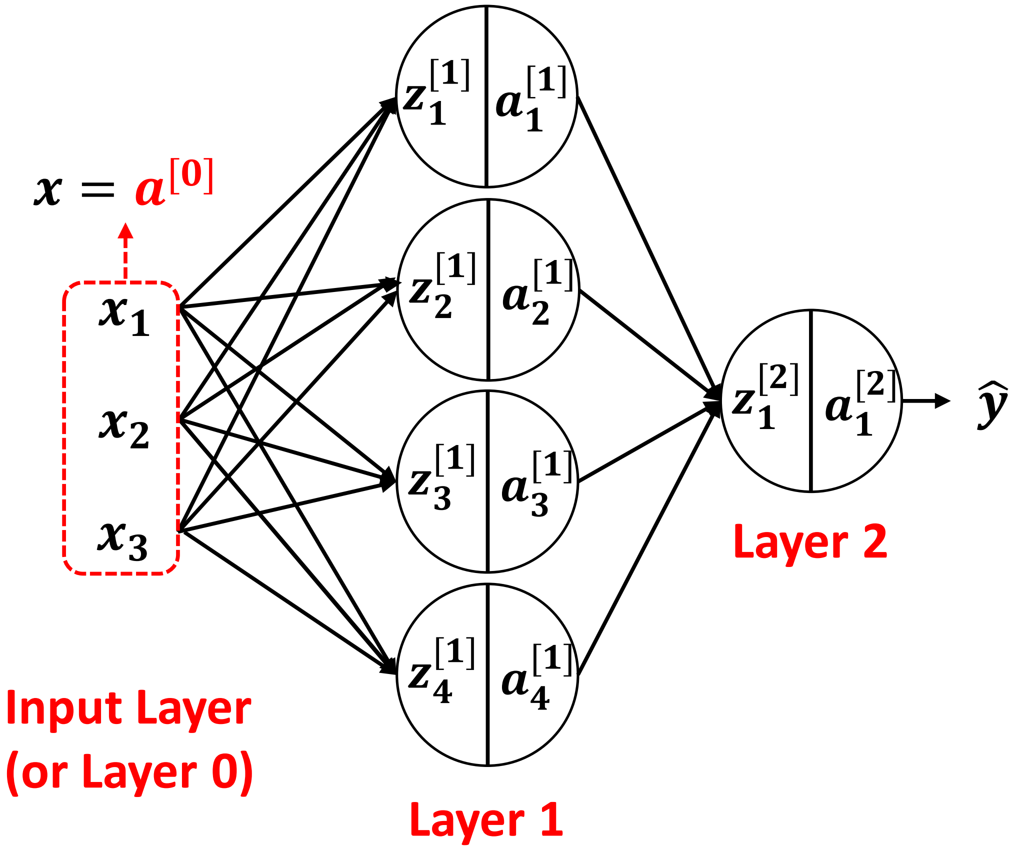


$$X = A^{[0]} = \underbrace{\begin{bmatrix} \boldsymbol{a}_1^{[0](1)} & \boldsymbol{a}_1^{[0](2)} & & \boldsymbol{a}_1^{[0](n)} \\ \boldsymbol{a}_2^{[0](1)} & \boldsymbol{a}_2^{[0](2)} & \cdots & \boldsymbol{a}_2^{[0](n)} \\ \boldsymbol{a}_3^{[0](1)} & \boldsymbol{a}_3^{[0](2)} & & \boldsymbol{a}_3^{[0](n)} \end{bmatrix}}$$

If we denote $\boldsymbol{x}$ as $\boldsymbol{a}^{[0]}$

Vectorizing Input and All Variables

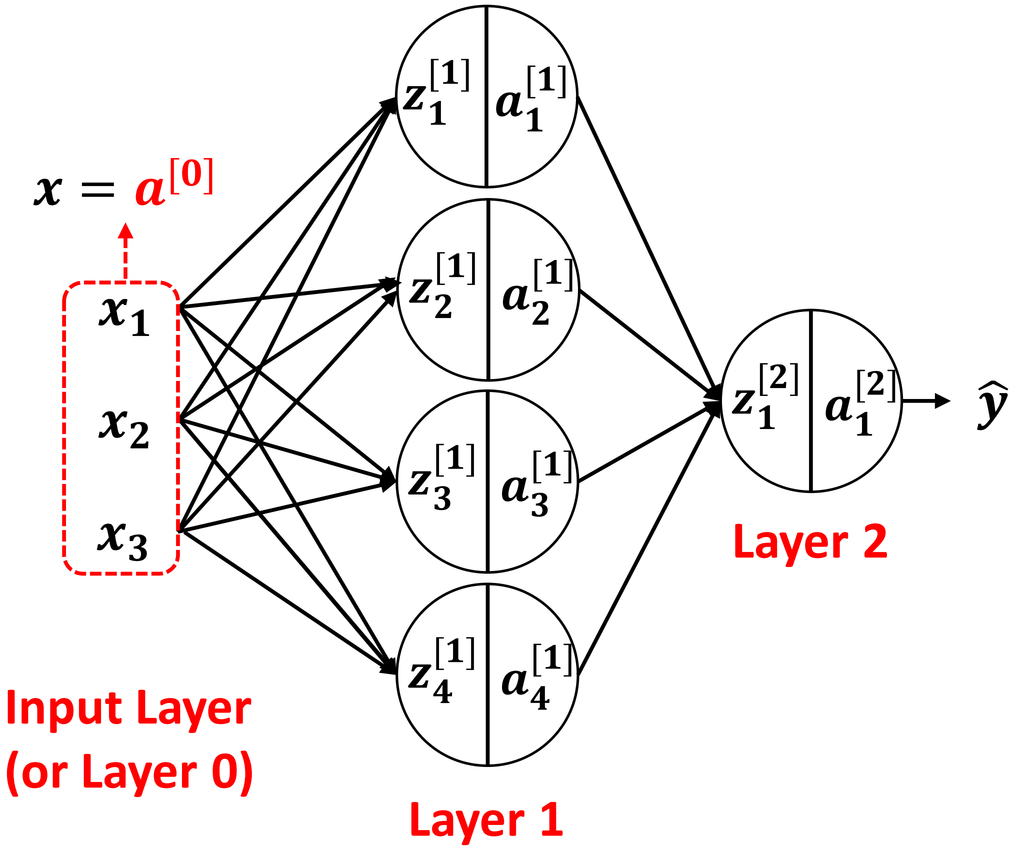
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$Z^{[1]} = \begin{bmatrix} z_1^{[1](1)} & z_1^{[1](2)} & & z_1^{[1](n)} \\ z_2^{[1](1)} & z_2^{[1](2)} & & z_2^{[1](n)} \\ z_3^{[1](1)} & z_3^{[1](2)} & \cdots & z_3^{[1](n)} \\ z_4^{[1](1)} & z_4^{[1](2)} & & z_4^{[1](n)} \end{bmatrix}$$

Vectorizing Input and All Variables

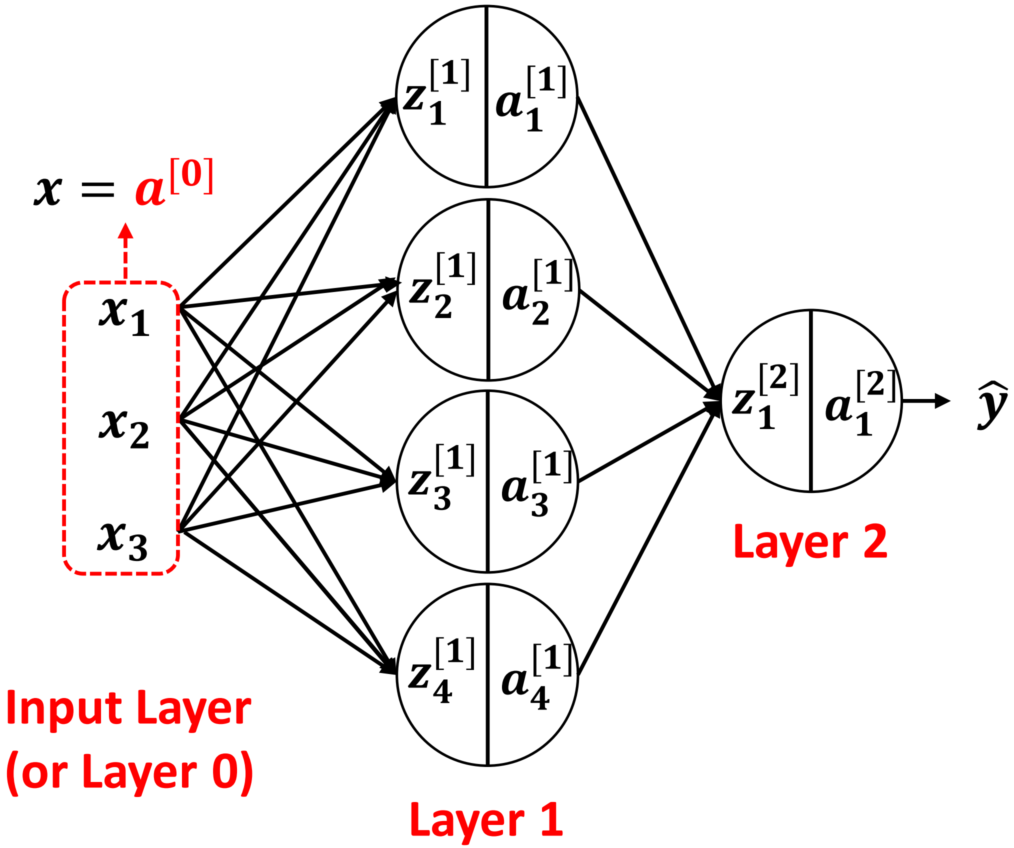
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$A^{[1]} = \begin{bmatrix} a_1^{[1](1)} & a_1^{[1](2)} & & a_1^{[1](n)} \\ a_2^{[1](1)} & a_2^{[1](2)} & & a_2^{[1](n)} \\ a_3^{[1](1)} & a_3^{[1](2)} & \cdots & a_3^{[1](n)} \\ a_4^{[1](1)} & a_4^{[1](2)} & & a_4^{[1](n)} \end{bmatrix}$$

Vectorizing Input and All Variables

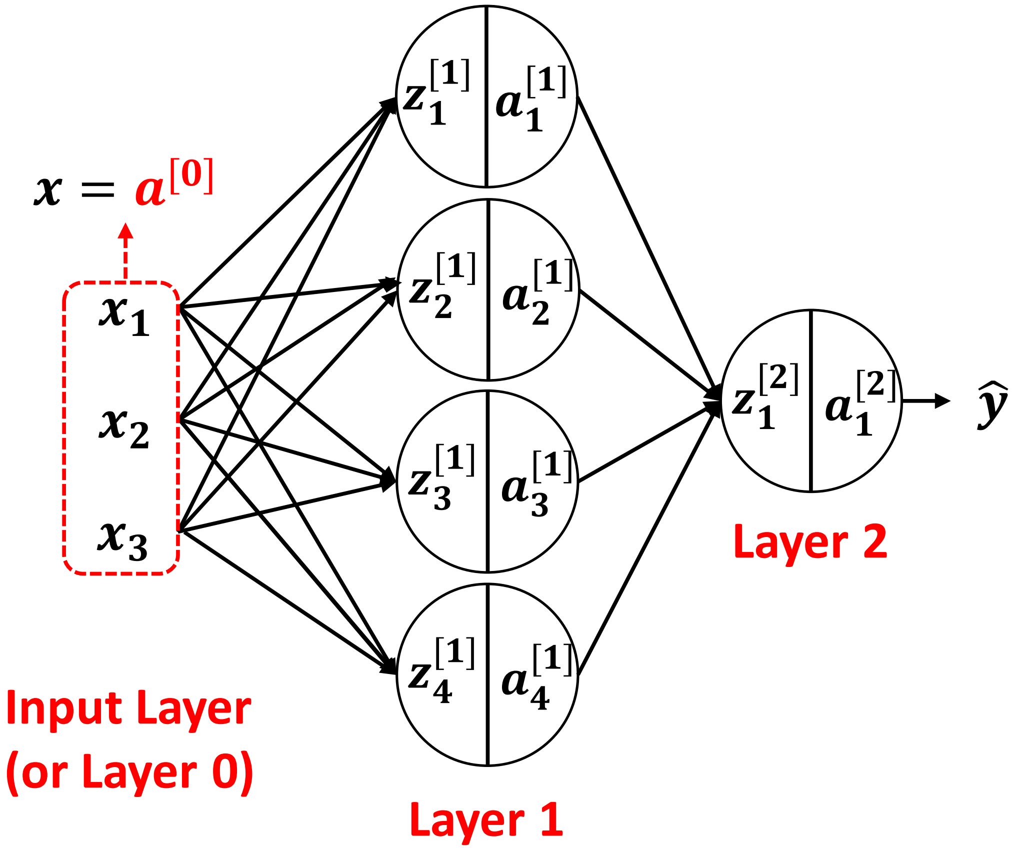
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$Z^{[2]} = \left[z_1^{[2](1)} \quad z_1^{[2](2)} \dots \quad z_1^{[2](n)} \right]$$

Vectorizing Input and All Variables

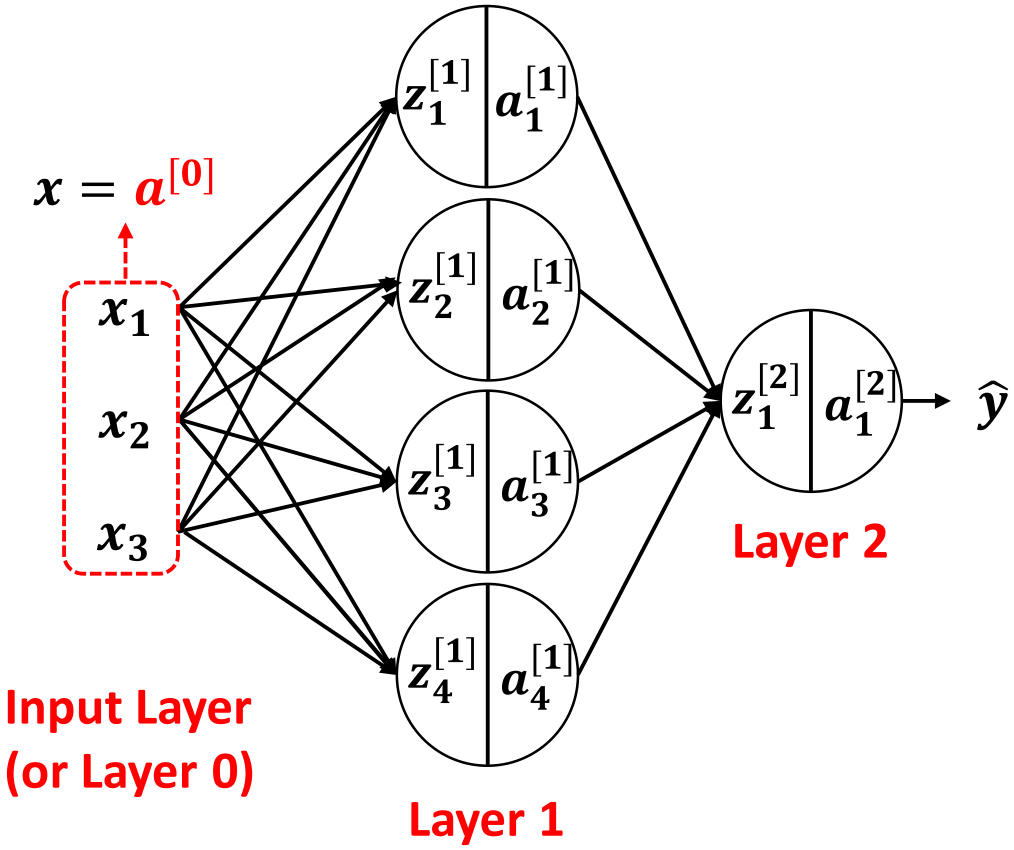
- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$A^{[2]} = \left[a_1^{[2](1)} \quad a_1^{[2](2)} \dots \quad a_1^{[2](n)} \right]$$

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



$$\boldsymbol{for\ i = 1\ to\ n:}$$

$$\boldsymbol{z^{[1](i)} = w^{[1]^{T}(i)} a^{[0](i)} + b^{[1]}}$$

$$\boldsymbol{a^{[1](i)} = \sigma(z^{[1](i)})}$$

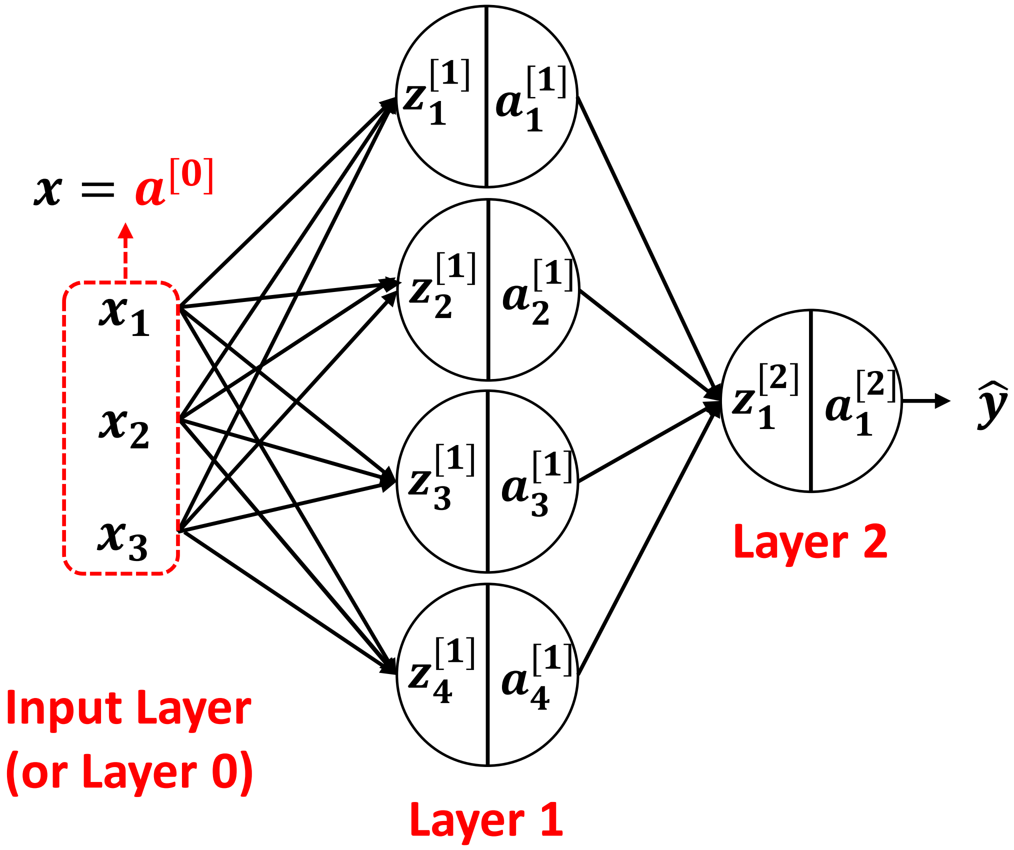
$$\boldsymbol{z^{[2](i)} = w^{[2]^{T(i)}} a^{[1](i)} + b^{[2]}}$$

$$\boldsymbol{a^{[2](i)} = \sigma(z^{[2](i)})}$$

***Before* Vectorization**

Vectorizing Input and All Variables

- To help develop an efficient learning algorithm, let us ***vectorize*** (represent in vectors & matrices) the input and the variables involved



No Explicit Loop!

$$Z^{[1]} = w^{[1]^T} A^{[0]} + b^{[1]}$$

$$A^{[1]} = \sigma(Z^{[1]})$$

$$Z^{[2]} = w^{[2]^T} A^{[1]} + b^{[2]}$$

$$A^{[2]} = \sigma(Z^{[2]})$$

After **Vectorization**

Summary

- Spam and phishing
- Examples of Email Fraud
- Spam Filtering Techniques
- Pre-processing Text in Email Messages
  - bag-of-words
  - n-gram
  - TF-IDF
- Spam detection with neural networks
  - Model representation

References

- *Email Spam Filtering, Computer Security Seminar, N.Muthiyalu Jothir*
- *Special Topics: Adversarial Machine Learning, Dr. Alex Vakanski, University of Idaho*
- *Introduction to Information Retrieval, Christopher D. Manning et al, Cambridge University Press. 2008.*
- *Machine Learning and Security Protecting Systems with Data and Algorithms, Clarence Chio, David Freeman*
- *Deep learning, Mohammad Hammoud, CMU Qatar*