# Exam and Revision

CSIT375 AI for Cybersecurity
Dr Wei Zong
SCIT University of Wollongong

# Outline

- Things to know for the final exam
- Subject Revision
- Q&A

# Final Exam – Online invigilated Moodle Quiz

- 3 Hours
- Close-book exam
  - No calculations.
    - Calculations have already been done in Quiz 2 and the assignment.
  - Knowledge and intuition are the key.

# Final Exam – question types

- 21 - 31 questions (50 marks), including:
  - 7 multiple choice (14 marks in total)
  - 4 True or False (6 marks in total)
  - 10 - 20 short answer (30 marks in total)
    - Convince me that you understand the ideas.
    - Make sure your handwriting is clear enough.
- A minimum of 40% of the final exam marks
  - Get at least 20 marks to avoid TF.

# Final Exam

- For all multiple-choice questions, one or multiple answers can be correct
- For each multiple-choice question, you’ll receive
  - full marks if all options are checked correctly (i.e. all correct answers are checked, wrong answers are not checked)
  - partial marks if part of options are checked correctly (i.e. part of correct answers are checked, wrong answers are not checked)
  - zero otherwise
- **No code or calculations** in the final exam

# How to prepare

- Scope: Lecture slides and lab notes
- Review the lectures and examples
- Review the lab notes
- Review the assignments
- Attempt the quiz “Sample Exam for CSIT375” (available on Moodle)

# During the Exam

- Answer **ALL** questions
  - Read questions carefully
  - You can attempt the questions in any order, arrange your time wisely
- Short answer questions
  - **BRIEFLY** answer each question with important and the most relevant points.
  - There is no need to include explanation if the question does not require it.

# Revision

# Introduction

- AI, Machine learning, Deep learning
  - ML: classification/regression/clustering, supervised/unsupervised learning
- Cybersecurity: C-I-A triad
  - **C**onfidentiality, Integrity, Availability
- Common types of cybersecurity threats
  - Phishing, ransomware, malware, social engineering
- Applications of AI in Cyber security
  - Monitoring (intrusion, privacy violation, exfiltration, ... )
  - Analysis (causation, consequence, correlation, summarization)
- Limitations of AI in Security
  - False negatives/false positives
  - False positives + inexplicable results → Signal fatigue

# Linear regression

- How to learn a linear regression model $\boldsymbol{h_\theta(x)} = \boldsymbol{\theta_0} + \boldsymbol{\theta_1 x}$ given a labelled dataset ($\boldsymbol{\theta}_0$ = b, $\boldsymbol{\theta}_1$= m, and $\boldsymbol{h_\theta(x)}$= y' when using y' = mx + b)?
  - By minimizing ***mean squared error***. That is:

This problem is referred to as an ***optimization problem*** with the objective of:

$$\underset{\theta_0,\theta_1}{\text{minimizing}}\, \boldsymbol{J(\theta_0,\theta_1)}$$

$$\text{minimize}\frac{1}{2n}\sum_{i=1}^{n}\left(y'^{(i)} - y^{(i)}\right)^2$$

$$\equiv$$

$$\underset{\theta_0,\theta_1}{\text{minimize}}\boxed{\frac{1}{2n}\sum_{i=1}^{n}\left((\theta_0+\theta_1 x)^{(i)} - y^{(i)}\right)^2}$$

Known as ***cost function*** $\boldsymbol{J(\theta_0,\theta_1)}$ **MSE**

$$\equiv$$

$$\underset{\theta_0,\theta_1}{\text{minimize}}\,\boldsymbol{J(\theta_0,\theta_1)}$$

# Linear regression

- Gradient Descent Steps
  - Have some cost function $\boldsymbol{J(\theta_0, \theta_1)}$ (here we are using MSE)
  - Start with some guesses for $\theta_0, \theta_1$
    - a common choice is to set them both initially to zero or random values.
  - Repeat until convergence{

$$temp_0 = \theta_0 - \alpha \boxed{\frac{\partial \boldsymbol{J(\theta_0, \theta_1)}}{\partial \theta_0}} \quad \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta(x)^{(i)} - y^{(i)}\right)$$

Partial derivatives of MSE with respect to parameters

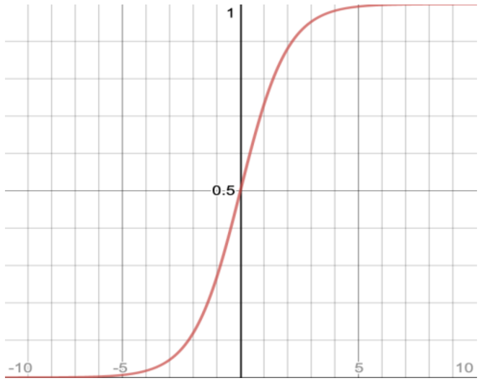
$$temp_1 = \theta_1 - \alpha \boxed{\frac{\partial \boldsymbol{J(\theta_0, \theta_1)}}{\partial \theta_1}} \quad \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta(x)^{(i)} - y^{(i)}\right) .x^{(i)}$$

$$\theta_0 = temp_0$$
$$\theta_1 = temp_1$$

}

# Logistic regression

- The Logistic Regression Model




$$\boldsymbol{h_\theta(x)} = \boldsymbol{g(\theta^T x)} = \frac{1}{1 + e^{-\theta^T x}}$$

  - $\boldsymbol{x}$ is the ***feature vector***: $\boldsymbol{x} = [\boldsymbol{x_0}, \boldsymbol{x_1}, \dots, \boldsymbol{x_m}]^T$
  - $\boldsymbol{\theta}$ is the ***parameter vector***: $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \boldsymbol{\theta_1}, \dots, \boldsymbol{\theta_m}]^T$
  - $\boldsymbol{g(z)}$ is the sigmoid function: $g(z) = \frac{1}{1+e^{-z}} \in [0,1]$
  - $h_\theta(x)$: the probability of input belonging to the class labeled with 1
- ***After*** learning the model, we can then apply thresholding to predict the binary output as follows:

$$if\ h_\theta(x) < 0.5 \text{ predict } 0$$
$$if\ h_\theta(x) \geq 0.5 \text{ predict } 1$$

# Logistic regression

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x)} = \boldsymbol{g(\theta^T x)}$, where $\boldsymbol{\theta}$ = $[\boldsymbol{\theta_0}, \dots, \boldsymbol{\theta_m}]^T$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \dots, \boldsymbol{x_m}]^T$?
  - By minimizing the following cost function:

$$\underset{\theta}{\text{minimize}} \boxed{\frac{1}{n} \sum_{i=1}^{n} -y^{(i)} \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}}\right) - (1-y^{(i)}) \log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}}\right)}$$

***Cross entropy loss***

- where n is the total number of examples, the sum is over all training inputs, $y^{(i)}$ true label, log(predicted probability observation)
- cross entropy loss is also used in classification task using NN (lab task)

$$J = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \log\left(a^{(i)}\right) + (1-y^{(i)}) \log\left(1-a^{(i)}\right)\right)$$

# Model evaluation

- Accuracy = $\frac{\textit{number of correctly predicted data points}}{\textit{total number of data points}}$
- Different types of errors
  - False positive, false negative
- Precision and Recall
  - Recall: TP/(TP + FN)
  - Precision: TP/(TP + FP)
- Confusion matrix

| | | | |
|---|---|---|---|
| | | **Predicted condition** | |
| | Total population = P + N | **Positive (PP)** | **Negative (PN)** |
| **Actual condition** | **Positive (P)** | **True positive (TP)** | **False negative (FN)** |
| | **Negative (N)** | **False positive (FP)** | **True negative (TN)** |

- F-score: A measure that combines ***P***recision and ***R***ecall
  - $F_\beta, F_1$

# Spam detection

- Spam and phishing
- Spam Filtering Techniques
  - Challenge-Response Filtering, Blacklists and Whitelists, Rule based filters, Content based filters
- Pre-processing Text in Email Messages
  - Tokenization, Vectorization
  - Bag-of-words, n-grams, Vector Space Model,
  - TF-IDF
    - raw term frequency, log-frequency weight
    - document frequency, Inverse document frequency
    - TF-IDF Weighting: $= (\mathbf{1} + \boldsymbol{log}(tf_{t,d})) \times \boldsymbol{log}\left(\frac{\boldsymbol{N}}{\boldsymbol{df}_t}\right)$

# Neural network

## How to train a Neural Network: Summary (More details in the lab task)

1. Define the neural network structure (model representation)
2. Initialize the model's parameters

    initialize the weights matrices with random values

    initialize the bias vectors as zeros

3. Loop:

    - Implement forward propagation
    - Compute loss (cross-entropy loss as shown in the lab notes)
    - Implement backward propagation to get the gradients
    - Update parameters (gradient descent)

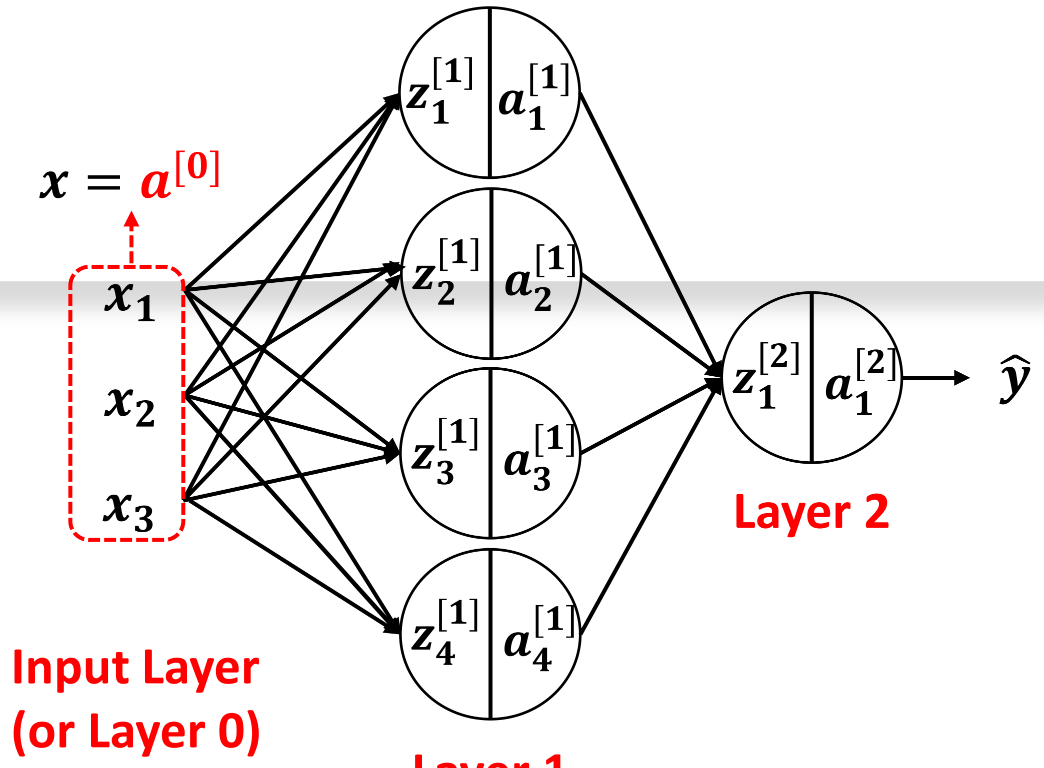
## How to make predictions?

Use forward propagation to predict results

**Reminder**: predictions = $y_{prediction} = \mathbb{1}\{\text{activation} > 0.5\} = \begin{cases} 1 & \text{if } activation > 0.5 \\ 0 & \text{otherwise} \end{cases}$

# Neural network

- Forward propagation




Assume 1 training example:

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

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

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

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

Assuming $n$ examples

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

$$\underbrace{\begin{aligned} Z^{[1]} &= w^{[1]^T} A^{[0]} + b^{[1]} \\ A^{[1]} &= \sigma(Z^{[1]}) \\ Z^{[2]} &= w^{[2]^T} A^{[1]} + b^{[2]} \\ A^{[2]} &= \sigma(Z^{[2]}) \end{aligned}}_{\textit{After Vectorization}}$$

# Anomaly detection

- Anomalies (outliers)
  - data object that deviates significantly from the rest of the objects
- Types of anomalies
  - Global anomalies, Contextual anomalies, Collective anomalies
- Applications of anomaly detection
  - Fraud detection, Industrial damage detection, Cyber Intrusion detection
- Intrusion detection system (IDS)
  - Network-based, Host-based, signature-based, anomaly-based
  - Feature engineering for HIDS (IoCs), for NIDS (DPI)
- Anomaly detection techniques
  - With labels – train classifiers, confidence score, softmax function
  - Without labels – statistical methods, parametric (Gaussian), nonparametric (histogram)

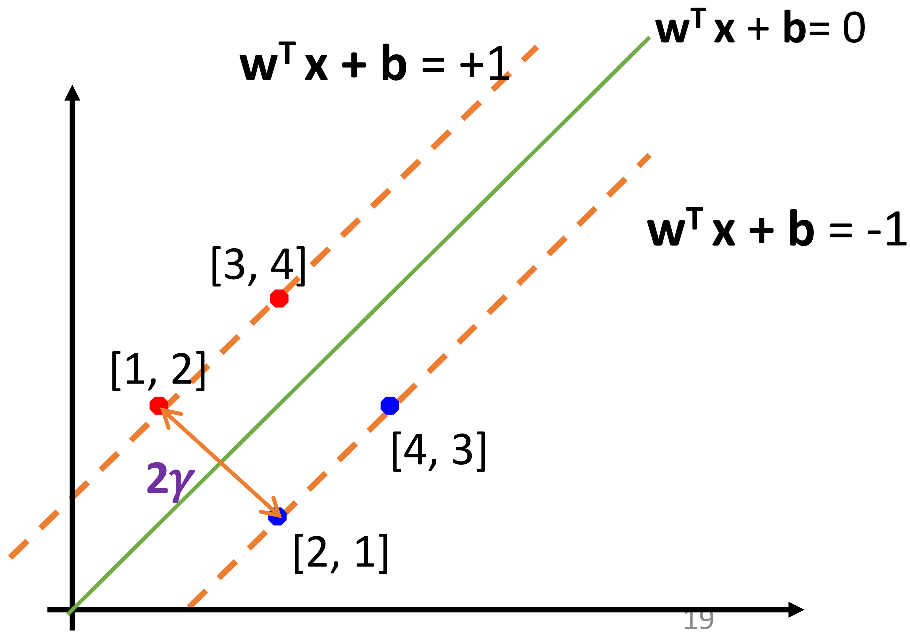
# SVM

- limitations of Perceptron model
- Linear SVM: Maximum margin linear classifier
  - Objective: select a hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ that maximizes the distance between the hyperplane and examples in the training set
  - Constraints that all data follows:

    $\mathbf{w}^T\mathbf{x}^i + b \geq 1$ if $y^i = +1$

    $\mathbf{w}^T\mathbf{x}^i + b \leq -1$ if $y^i = -1$
  - For support vectors, the inequality becomes equality
  - The margin is: $2\gamma = \frac{2}{\|\boldsymbol{w}\|}$

| x | y |
|---|---|
| [1, 2] | +1 |
| [2, 1] | -1 |
| [3, 4] | +1 |
| [4, 3] | -1 |



# Decision Tree

- Malware: malicious software, steal data, damage computer systems
- Common types of Malware
  - Virus, trojan, botnet, downloader, rootkit, ransomware…
- Malware analysis methodology
  - Static (code) analysis
  - Dynamic (behavioral) analysis
- Decision tree
  - Entropy: uncertainty
  - Information Gain: goodness of a split
  - Build a Decision Tree: ID3 algorithm

# Entropy and Information gain

- **Entropy $H(Y)$ of a random variable $Y$ with $n$ different possible values:**

$$H(Y) = -\sum_{i=1}^{n} P(y_i) log_2 P(y_i)$$

- **Conditional entropy $H(Y|X = x_j)$ of $Y$ given $X = x_j$:**

$$H(Y|X = x_j) = -\sum_{i=1}^{n} P(y_i|X = x_j)\, log_2 P(y_i|\, X = x_j)$$

- The above equation: split the data according to the value of $X$, the entropy of random variable $Y$ with $X = x_j$.

- **Expected entropy $H(Y|X)$ after split:**

$$H(Y|X) = \sum_{j=1}^{k} P(x_j)\, H(Y|X = x_j)$$

- The above equation: $X$ has $k$ possible values, $P(x_j)$ is the probability that $X = x_j$.

- **Information gain $I(Y, X)$** : expected reduction in entropy of target variable $Y$ after split over variable $X$

$$I(Y, X) = H(Y) - H(Y|X)$$

# Case study

- Explore and prepare the data
  - Data distribution, class imbalance
  - Feature scaling: standardization/normalization, consistent transformations to both training & test sets
- Multiclass classification: split into multiple binary classification problems
  - one-versus-all: train one classifier per class
  - one-versus-one: train one classifier per class pair
- Supervised learning: DT, Random Forest, SVM
- PCA

# Adversarial Machine Learning

- Adversarial examples
  - White-box
  - Black-box
  - Physical
  - Detection
  - Adversarial training
- Backdoor attack
  - Visible triggers
  - Invisible triggers
  - Clean-label attack.

# Deepfake detection

- GAN
- Detecting artifacts
- Watermarks

# Good Luck!

Questions?