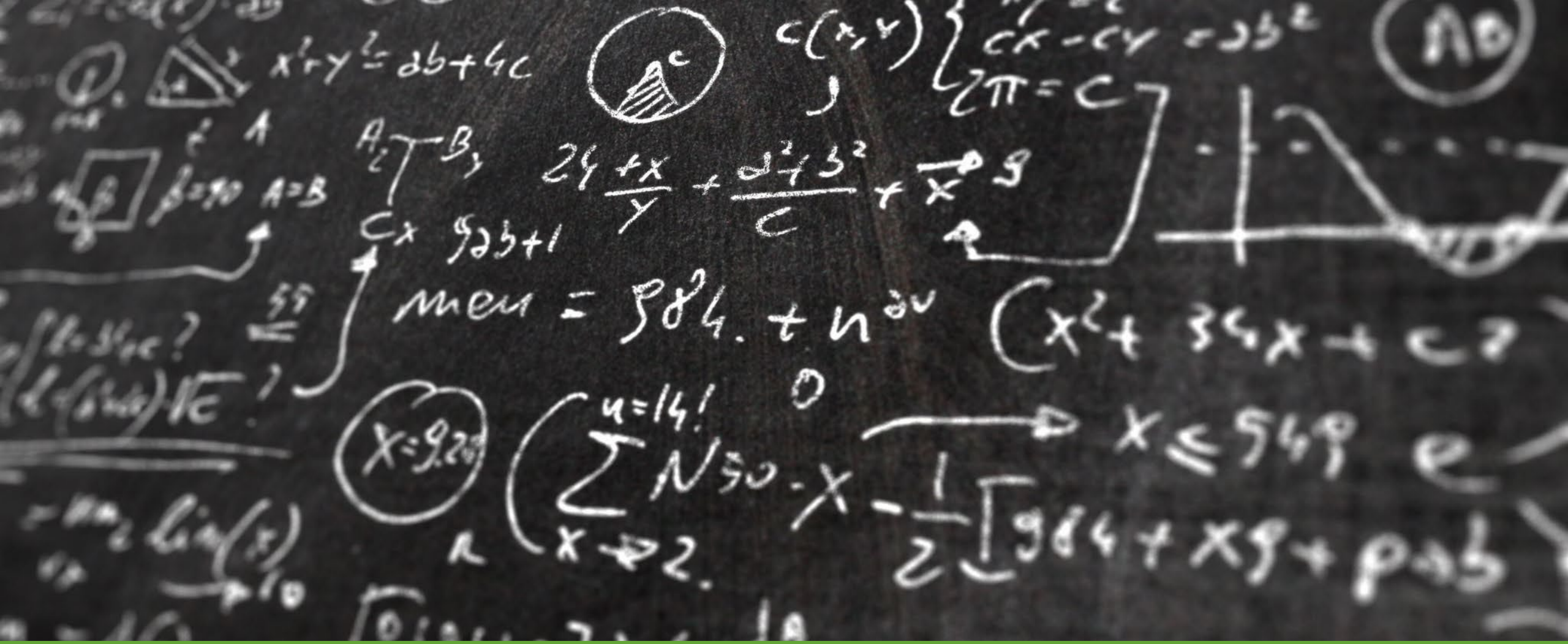


OTHER NOTABLE ALGORITHMS

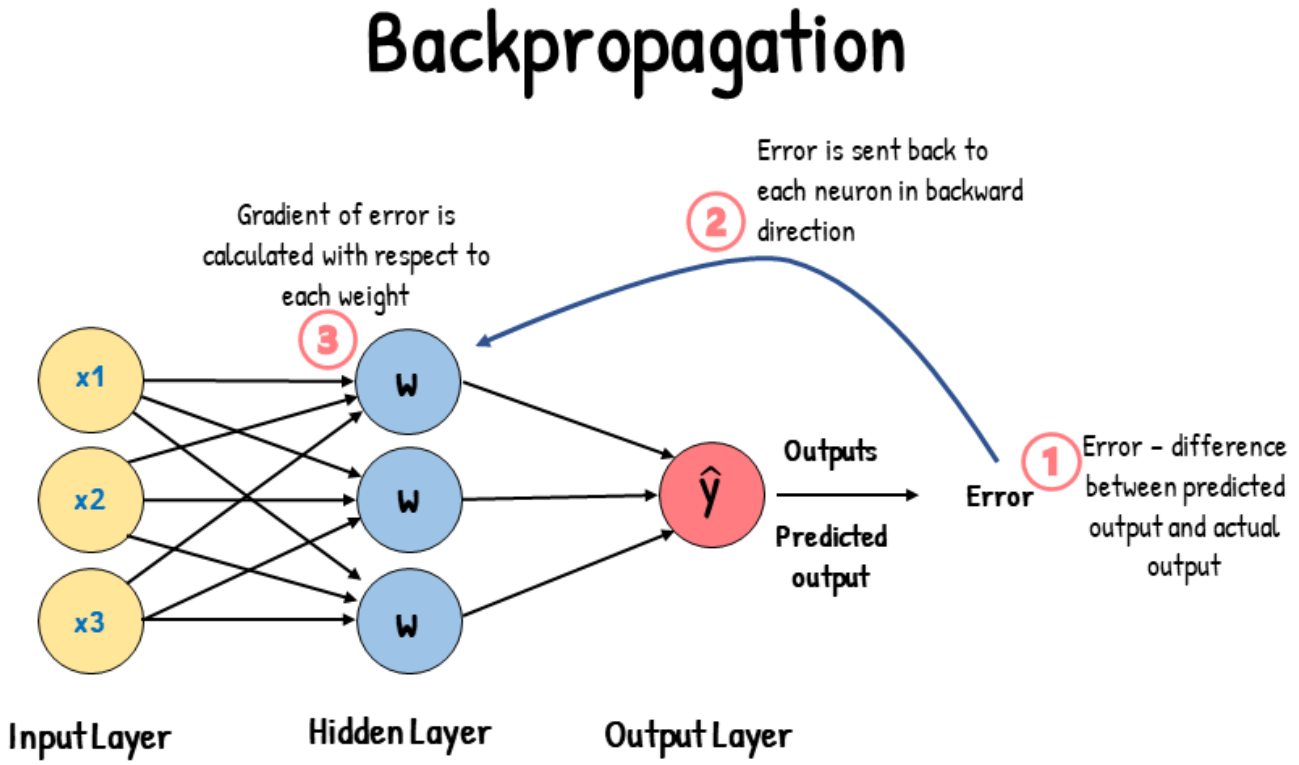
Lecturer: Mark Gillan

A REMINDER: BACKPROPAGATION

Short for 'backward propagation of errors.

An algorithm commonly used for supervised learning of artificial neural networks.

It is a 'gradient estimation' method – used to train neural network models.



BRAIN TRAIN: BACKPROPAGATION ALGORITHM

The Journey of a Prediction:

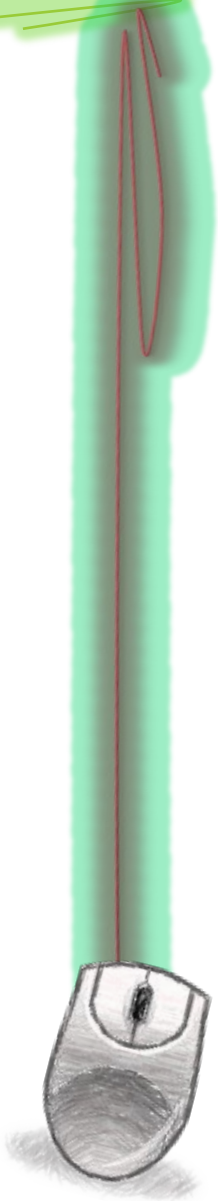
- **Forward Pass:** Information flows through the network, layer by layer, transforming inputs into predictions.
- **Comparing Output:** We compare the network's prediction to the actual answer, calculating the error (frustration points).

The Learning Begins:

- **Backward Pass:** Backpropagation starts at the "frustration points" and travels back through the network, layer by layer.
- **Adjusting Weights:** The error is "distributed" among the connections (weights) between neurons, like assigning blame for the wrong answer.
- **Smaller Steps:** We then update the weights slightly based on the blame, pushing the network closer to the right answer in the future.

Remember:

- Backpropagation is an iterative process – repeat forward passes, calculate errors, and adjust weights until the network gets good at its task.
- It's like training a child – reward successes, gently correct mistakes, and watch the network learn and grow!



A BUOYANT BREAKDOWN: BUBBLESORT ALGORITHM

- *Imagine a tray of rising bubbles. The biggest and lightest ones effortlessly float to the top, while the smaller and denser ones slowly make their way up. That's kind of like how Bubble Sort works!*
- Iterate through a list: Think of each element in the list as a bubble.
- Compare adjacent bubbles: If the right bubble is smaller than the left bubble, swap them! Just like lighter bubbles rise above heavier ones.
- Repeat the process: Keep iterating through the list, comparing and swapping bubbles until no more swaps are needed. Now, all the "biggest" elements (i.e., the ones in the correct order) have risen to the top!
- Here's an analogy to help visualise it:
- Imagine sorting books on a shelf by height. You start by comparing the heights of neighbouring books. If the book on the right is shorter than the one on the left, swap them! Keep going through the shelf, swapping shorter books over taller ones, until all the books are in ascending order by height.
- That's Bubble Sort in action!

DECODING THE WEB'S POPULARITY: PAGERANK

- *Imagine surfing the vast ocean of the internet, where websites are islands and links are ships. Some islands might be bustling metropolises, brimming with connections, while others are quiet villages with few visitors. PageRank is like a sophisticated compass, helping navigate this digital sea by ranking websites based on their online "importance."*

- Votes on the web: Imagine each link to a website is like a vote of confidence. The more links a website has, the more "popular" it's considered.

- Quality matters: But not all votes are equal! Links from important or relevant websites carry more weight. Think of a respected scholar praising a research paper versus a random comment on social media.

- The democratic dance: PageRank iteratively analyses these votes, distributing and redistributing "link juice" (think voting power) between websites. Websites with many high-quality links gain authority, influencing the ranking of others they link to.

- Reaching equilibrium: This iterative process eventually converges, reaching a stable ranking where the most important websites (according to the web's "votes") hold the top spots.

- Here's an analogy to help visualise it:

- Imagine trying to find the best restaurants in town. You might ask friends, family, and trusted online reviews for recommendations. These recommendations are like links, and the most frequently mentioned restaurants gain traction in your mind

QUICK EXPLANATION: K-NEAREST NEIGHBORS

- *Imagine you have a bunch of labelled data points, life fruits represented by their size and colour.*
- *Now, you encounter a new, unlabelled fruit. K-Nearest Neighbors helps you predict its label (type of fruit) by looking at its 'k' closest neighbour in the existing data.*
- *The majority vote among these k neighbours determines the predicted label for the new fruit.*
- k is a crucial parameter in KNN. Choosing the right k value significantly impacts the algorithm's accuracy.
- Low k values (e.g., k=1) can be sensitive to noise and lead to overfitting, where the model memorizes the training data but fails to generalize to unseen examples.
- High k values (e.g., k=10) can make the decision boundary smoother but may lead to underfitting, where the model fails to capture the underlying patterns in the data.

Applications of KNN:

- Image recognition: Classifying images based on their content (e.g., identifying objects in a picture).
- Recommendation systems: Recommending products or services to users based on their past preferences and similar users' behaviour.
- Fraud detection: Identifying fraudulent transactions by comparing them to known fraudulent cases.

UNLEASHING EVOLUTION: GENETIC ALGORITHMS

- *Imagine solving problems like a living organism adapting to its environment. Genetic Algorithms (GAs) mimic natural selection and evolution to optimize solutions.*
- *We start with a population of solutions (chromosomes) represented by strings of genes.*
- *Each generation, these solutions "breed" and "mutate," creating new offspring with potentially better traits..*
- Selection: The fittest solutions (those with higher fitness scores based on the problem) are chosen to reproduce.
- Crossover: Parent solutions combine their genes, leading to offspring with potentially superior features.
- Mutation: Random changes occur in genes, introducing diversity and preventing stagnation..

Applications of Genetic Algorithms:

- Optimization: Scheduling, logistics, resource allocation, financial trading.
- Machine Learning: Feature selection, neural network design, evolutionary robotics.
- Design and Engineering: Material design, drug discovery, image processing.

BRANCHING OUT INTO AI: DECISION TREES

- *Imagine navigating a dense forest, making choices at each fork until you reach your destination. Decision trees work similarly, guiding computers through data to make predictions or classifications.*
- *Decision trees are structured like, well, trees! They start with a root node representing the entire dataset.*
- *Branches split the data based on decision rules, asking questions about features (e.g., "Is the object red?").*
- *Each branch leads to a leaf node, which holds the final prediction or classification (e.g., "This is an apple").*
- Overfitting: Can lead to overly complex trees that memorize the training data but lack generalizability.
- Prone to bias: Splitting decisions based on irrelevant features can skew the results.
- May not perform well with complex relationships: Simple rules might not capture intricate patterns in the data.

Applications of Decision Trees:

- Fraud detection: Analysing financial transactions to identify suspicious activity.
- Medical diagnosis: Supporting doctors in making diagnoses based on patient symptoms and tests.
- Marketing campaigns: Targeting specific customer segments for personalized marketing efforts.