**Machine Learning**: field of study that gives computers the ability to learn without being explicitly programmed

**Machine learning algorithms:**

* \*Supervised learning – used most in real world application, rapid advancements
* \*Unsupervised learning
* \* Recommender systems
* Reinforcement learning
* \* = most used

**Supervised learning**: algorithms that learn x to y (input to output mappings)

* Give learning algorithm examples that includes the correct label y to given input x (right answers)
* Learning algorithm learns to take the input alone, then gives a reason accurate guess of the output
* Includes the input features and output targets. To train the model, you feed the model these 2 features to the learning algorithm. Then it produces a function f. The job of function f is to take an input x and produce a prediction (y-hat). X is called the input feature and the output is the prediction called y-hat. Y-hat is the estimated value. Y variable is the true value.
* Examples ( input (X) -> output(Y) | Application)
  + Email -> spam(0|1) | spam filtering
  + Audio -> text transcripts | speech recognition
  + English - > Spanish | machine translation
  + Ad, user info -> click(0|1) | online advertising – lucrative form
  + Image, radar info -> position of other cars | self-driving cars
  + Image of phone -> defect (0|1) | visual inspecting
  + First train the model with examples of input x, then the right answers. After the model learned from the x and y pairs, then they take an new input, something they never seen before to produce a accurate output (appropriate corresponding y)
* Type 1 of Supervised Learning: **Regression** – predicts a number, infinitely many possible outputs . For instance, predicting the housing prices.
* Type 2 of Supervised Learning: **Classification** – trying to predict a small number of possible outputs or categories. They predict categories (don’t have to be numbers). For instance, if a picture is a cat or a dog. When referring to output, class and category is the same thing. Categories can’t be all possible numbers such as 1.7 or 0.5. can also have two or more inputs. For instance, if a lump is dangerous or not dangerous.

**Unsupervised Learning:** data only comes with inputs x, but not output labels y

* not given any data labels
* algorithm has to find structure in the data
* find something interesting in unlabeled data
* **Clustering Algorithm –** takes data without labels and tries to automatically group them into clusters.
* **Anomaly detection:** find unusual data points or events. Important in fraud detection such as a bank application
* **Dimensionality reduction:** lets you take a big data and compress this big data into a small dataset without taking away important information

**Linear Regression with One Variable:**

* Fitting a straight line to your data
* Can be visually seen using a linear graph and a data table.
* A single output or a single feature such as the size of the house
* Another name is univariate linear regression
* **Univariate:** one variable
* You want a efficient algorithm to automatically find w and b so you can minimize J(w,b)
* To do this, there is called gradient descent and variations. s

**Terminology or standard notation to describe a training model**

* **Training set**: Data used to train the model
* **x**: “input” variable feature
* **y**: “output” variable or “target” variable
* **m**: number of training examples
* **(x, y)**: single training example
* **(x^(i), y^(i))**: ith training example – ith refers to the index in the training set but not the exponent

**Cost Function**: tells us how well the model is doing so we can make this model better

* Parameters, coefficients, weights of the model are variables you can adjust during training in order to improve the model
* How to measure how well a line fits the training data, do cost function
* (“y-hat” – y ) = error, how far off it is from the target
* J(w,b) = 1/2m Sum of I to m ((“y-hat^(i)” – y ^(i))^2
* The above is called squared error cost function which is common for linear regression
* Can use different types of cost function
* M in this case is the number of training examples

**Cost Function:** obtained by subtracting the predicted values from actual values. Our cost function will be the minimum of these error values

A picture containing text, receipt, font, screenshot

Description automatically generated

**Need for Cost Function:**

* After training our model, we need to see our how badly our model is performing. While accuracy function tell us how well our model is performing, they do not provide us with insight on how to better improve them
* Cost Function is used to measure how wrong the model is in finding a relation between the input and output. It tells us how badly our model is behaving or predicting.

**Cost Function Linear Regression:**

* As we optimize the values of our model, for some variables, we will get the perfect fit
* Will never have more than one local minimum
* Convex function: has only one global minimum. Always converge to global mimimum

**What exactly is cost function really computing ?**

* finds the best parameters so it fits the training data
* for each value of w, you can calculate the cost of J of w
* choose w to minimize J(w)

**Training Linear Regression, Gradient Descent:**

* **Gradient Descent:** an algorithm to minimize any general function
* Gradient decent helps you go down hill
* Batch gradient descent: “batch” means each step of gradient desnecent uses all the training examples

**Implementing Gradient descent**

* Learning rate is usually small. How big you take a step downhill
* When a learning rate is always a positive number, you end up with a new value for W that is smaller

A picture containing text, screenshot, font, line

Description automatically generated

**Learning Rate:**

* If learning rate is too small, you do end up decreasing cost J but incredibly slowly. In other words, gradient descent may be slow since it will take a lot of steps to get to the minimum
* If learning rate is too large, then cost can get worst. Gradient descent may overshoot, and never reach minimum. It may also diverge
* If already reached local minimum, then does not change parameters.
* As we get a near a local minimum, gradient descent takes smaller steps. Since derivative becomes smaller and update steps become smaller. Can reach minimum without decreasing learning rate.

A picture containing text, font, handwriting, screenshot

Description automatically generated

**Prompt Engineering:**

* LLM – large language models
* 2 types of LLMs: Base LLM and Instruction Tuned LLM
* Base LLM predicts next word, based on text training data. Predicts the next serval words
* Instruction Tuned LLM tries to follow instructions. Fine-tune instructions and good attempts at following those instructions. They start of by Base LLM, huge text data, and further fine tune with input and output. Further refined by RLHF, Reinforcement learning with human feedback. This LLM is helpful, Honest, and Harmless.
* Practical applications use Instruction Tuned LLM
* **Prompt Guidelines Overview:**
  + Be clear and specific
  + Analyze why result does not give desired output
  + Refine the data and the prompt
  + Repeat
* **Guidelines for Prompting:** How to prompt Engineering effectively?
  + **Principle 1:** Write clear and specific instructions
  + **Principle 2:** Give model time to think
* Principle 1:
  + Tactic 1: Delimiters
    - Triple Quotes: “””
    - Tripe backticks: ```
    - Triple Dashes: ---
    - Angle Brackets: <>
    - XML tags: <tag> </tag>
    - Delimiters avoid prompt ejections.
  + Tactic 2: Ask for structured output
    - HTML
    - JSON
  + Tactic 3: Check whether conditions are satisfied. Check assumptions required to do the task
  + Tactic 4: few shot prompting – give successful examples of completing tasks. Then ask model to perform tasks.
* Principle 2:
  + Tactic 1: specific the steps to complete a task
    - Step 1: …
    - Step 2: …
    - …
    - Step N: …
  + Tactic 2: Instruct the model to work out its solution before rushing to a conclusion
* Model Limitations:
  + Hallucinations: Make things that sound possible but are not true
  + You can use the above techniques to reduce these hallucinations
  + Other techniques are: first find relevant information, then answer the question based on the relevant information
* **Iterative Prompt Development**:
  + Idea -> Implement (code/data) -> Experimental Results -> Error Analysis
  + Iterate the above over and over until an efficient machine learning model
  + Limit the number of words
  + Ask it to focus on the aspects that are relevant to the intended audience.
  + Ask it to extract information and organize it in a table
  + Summary: Prompt Development is an iterative process: try something, analyze where the results does not give what you want, clarify instructions, and give more time to think
  + To be effective, have a good process to develop prompts for your application
  + Refine prompts with a batch of examples
* **Summarize Text:**
  + Summarize with a word/character/sentence limit
  + Summarize with focus on shipping and delivery
  + Summarize with price and value
  + You can also extra specific information
* **Inferring**
  + extracts the sentiment like positive or negative
  + Information extraction is part of NLP, Natural language processing
  + Zero shot learning algorithm – no training data or labels.
* **Transforming** 
  + Large Language Models for text transformation tasks such as language translation, spelling and grammar checking, tone adjustment, and format conversion.
* **Expanding**
  + Expand a shorter text to a longer text
  + Temperature: degree of expiration or randomness of the model
  + At lower temperature like 0: most likely words
  + At higher temperature: lowers likely words to be chosen (random or more creative)