Machine Learning

1. Định nghĩa
   * Định nghĩa 1: **the field of study that gives computers the ability to learn without being explicitly programmed.**
   * Định nghĩa 2: **A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.**
     + Ví dụ: chơi cờ checkers
       - E: kinh nghiệm chơi cờ checkers
       - T: nhiệm vụ chơi cờ checkers
       - P: xác suất máy tính thắng
2. Phân loại:
   * Thông thường thì mọi vấn đề được chia làm 2 loại:
     + Supervised Learning
     + Unsupervised Learning
3. Supervised Learning
   * Ở vấn đề này, chúng ta được biết trước **data set** và **correct output** với ý tưởng rằng có 1 mối quan hệ nào đó giữa **input** và **output**
   * Supervised Learning chia làm 2 loại:
     + Regression:
       - Dự đoán **continuous output**
       - Tức là gán cho đầu vào một vài **hàm số liên tục**
     + Classification:
       - Dự đoán **discrete output**
       - Tức là gán cho đầu vào những **giá trị rời rạc**
4. Unsupervised Learning
   * Ở vấn đề này, chúng ta **không biết được biết trước output** sẽ như thế nào
   * Mục đích của Unsupervised Learning là đưa ra các nhóm dữ liệu **theo 1 structure** (cấu trúc, kiểu nhóm) nào đó dựa trên **mối quan hệ giữa các thông tin** của dữ liệu
5. Model Representation
   * Kí hiệu thường dùng (Notation):
     + : input
     + : ouput
     + : số lượng **training example** trong **training set**
     + : training example (dòng thứ trong bảng training set, )
     + : tập xác định của
     + : tập xác định của
   * Quá trình xây dựng Model:  
     Diagram

     Description automatically generated
     + Quá trình này miêu tả Supervised Learning
     + Cho 1 Training Set để học 1 hàm số
     + : hàm số dự đoán đoán “đủ tốt” cho giá trị ( viết tắt của *hypthesis*)
6. Cost Function (aka Squared Error Function or Mean Squared Error):
   * Kí hiệu:
   * Dùng để đo độ chính xác của hàm
   * Xây dựng :  
     Chart, line chart

     Description automatically generated
     + Ý tưởng: chọn , sao cho gần với nhất có thể (dựa trên các training example )
     + Ta có:
     + Khoảng cách giữa “giá trị dự đoán” và “giá trị chênh lệch” là:
     + Tổng các khoảng cách đó: (tại sao lại có bình phương)
     + Trung bình của khoảng cách là:
     + Ta nhân thêm vào tổng trên để sau này khi đạo hàm xuống sẽ dễ dàng tính hơn, trong khi giá trị và cần tìm không thay đổi:
     + Đặt tên tổng trên là
   * Tóm lại ta cần đi tìm và sao cho hàm – Cost Function – có giá trị nhỏ nhất
7. Gradient Descent (Suy giảm độ dốc)
   * Là thuật toán để tìm giá trị nhỏ nhất của hàm số
   * Ý tưởng của thuật toán, chọn 1 điểm bất kí trên hàm số, tại đó, nhìn xung quanh, xem đâu là hướng dốc nhất, rồi đi theo hướng đó, lặp đi lặp lại hành động đó cho đến khi tới được điểm thấp nhất.
   * Hình ảnh minh hoạ:  
     Chart, surface chart

     Description automatically generated
   * Tuỳ vào vị trí điểm đầu (được chọn), thuật toán có thể cho các kết quả khác nhau (như hình)\
   * Thuật toán có thể dùng cho , nhưng đơn giản hoá thì thuật toán sau chỉ dùng cho 2 đầu vào và :
     + Lặp lại cho đến khi hội tụ: với
     + : **learning rate**, quyết định độ dài bước đi của thuật toán
       - Nếu như quá nhỏ thì thuật toán sẽ bị chậm
       - Nếu như quá to thì thuật toán có thể sẽ chạy qua điểm nhỏ nhất, không thể hội tụ và có thể phân kì
       - Với cố định, nhưng do độ dốc của hàm thay đổi nên bước đi của thuật toán sẽ ngày càng nhỏ, vậy nên không cần phải thay đổi trong quá trình thực hiện
     + và phải được cập nhật đồng thời  
       A picture containing text

       Description automatically generated
8. Áp dụng Gradient Descent vào Linear Regression
   * Đầu tiên, ta xử lý đạo hàm từng phần của hàm
     + Đạo hàm theo :
     + Đạo hàm theo :
   * Thay đạo hàm vào thuật toán Gradient Descent:
     + Lặp lại cho đến khi hội tụ:
9. Multiple features (variables)
   * Linear regression với nhiều biến (variable) được gọi là **Multivaritate Linear Regression**
   * Notation:
     + : đầu vào thứ trong các training example
     + : giá trị thứ trong training example thứ
     + : số lượng training example
     + : số lượng biến
   * **Hypothesis Function** từ trở thành
   * Viết lại hàm dưới dạng ma trận:
     + Ta thêm 1 biến
     + Gọi:
     + Từ đó, ta viết được lại là:
   * **Cost Function** từ thành  
      nhưng 2 ở đây khác nhau, 1 là của 2 đầu vào, 1 là của đầu vào
     + Thay vì viết , thì ta có thể viết thành (với là )
   * Áp dụng thuật toán **Gradient Descent**:
     + Lặp lại cho đến khi hội tụ: với
     + Ta có:
     + **Lưu ý**:
10. Feature Scaling
    * Mục đích: tăng tốc độ cho thuật toán Gradient Descent
    * Lý do cần: giả sử biến nằm trong khoảng 0-2000, còn nằm trong khoảng 1-5, thì đồ thị của hàm này có dạng như dưới đây, và rất mất thời gian chạy.  
      Diagram

      Description automatically generated
    * Ý tưởng: đưa các giá trị về nằm trong khoảng từ đến (hoặc gần với , như hay
    * Bằng cách:
      + : giá trị trung bình của
      + : range của ,
11. Cách chọn learning rate ()
    * Có 2 cách chọn:
      + Debugging Gradient Descent
      + Automatic Convergence Test
    * Automatic Convergence Test
      + Chọn 1 giá trị đủ nhỏ (như ), nếu giá trị giữa 2 lần chạy thuật toán, tao thấy giảm bằng , thì ta nói hội tụ
      + **Chọn giá trị không dễ**
    * Debugging Gradient Descent:
      + Vẽ đồ thị với *số lần chạy* thuật toán Gradiant Descent (trục ) và *giá trị* của (trục )
      + Nếu quá lớn, đồ thị tăng giảm
      + Nếu quá nhỏ, đồ thị giảm chậm, hội tụ chậm tăng
      + **Nếu đủ nhỏ, đồ thị giảm với mỗi lần chạy**
12. Features và Polynominal Regression
    * Mục đích: cải thiện hàm hypothesis
    * Bằng cách:
      + Kết hợp biến (từ 2 biến chiều dài và chiều sâu thành diện tích chiều dài *chiều rộng*)
      + Đổi hàm (sang hàm bậc 2, hay bậc 3, hay căn)
      + Ví dụ:
        - Ta có:
        - Tao thêm feature dựa trên để có được hàm hoặc
        - Từ đó ta đặt (và ) để có
        - Khi đó, ta phải để ý đến feature scaling của các biến
          * Giả sử: trong khoảng thì sẽ trong khoảng và sẽ trong khoảng từ
13. Normal Function
    * Là giải pháp thay thế cho thuật toán Gradient Descent trong việc tìm để đạt giá trị nhỏ nhất.
    * Tại sao lại dùng Normal Function thay cho Gradient Descend?
      + Không phải đi tìm
      + Không phải lặp đi lặp lại thuật toán nhiều lần như Gradient Desent
      + Nhanh hơn Gradient Descent nếu như (features) nhỏ hơn khoảng
      + **Tuy nhiên** cũng có lúc Gradient Descent sẽ nhanh hơn (khi )
        - Gradient Descent chạy với thời gian là
        - Normal Function tốn thời gian chỉ riêng cho việc tính
    * Ý tưởng của Normal Fucntion:
      + Từ việc đạo hàm, cho đạo hàm bằng 0, giải ra để tìm cực trị:
        - Tìm điểm cực trị của hàm thì ta giải , tìm ra , thì tại đó hàm số đạt nhỏ nhất hoặc lớn nhất trong vùng lân cận
      + Áp dụng điều đó, ta muốn:
        - Lấy đạo hàm từng phần rồi đặt :
        - Rồi giải hệ tìm ra tìm vecto (vecto chiều)
      + Ta có công thức (không đi chứng minh):
        - Với ma trận và vecto được đưa ra giống như ví dụ dưới đây:  
          Diagram, schematic

          Description automatically generated with medium confidence
      + Ta **không cần** để ý tới features scaling với Normal Function
    * Trong trường hợp không thể tính :
      + Lý do 1: thừa feature, tồn tại feature có thể bị lược bỏ
        - Ví dụ: , , thì ta thấy , tức là điều này có thể khiến không tính được (kiến thức Linear Algebra)
      + Lý do 2: quá ít training examples , giải quyết:
        - Xoá bớt feature
        - Sử dụng **Regurazation** (sau sẽ học)