**K-nearest Neighbor**

1. **Giới thiệu.**

* K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning.
* Một số tên gọi khác của phương pháp học dựa trên các láng giềng gần nhất (Nearest neighbor learning)
  + Instance-based learning
  + Lazy learning
  + Memory-based learning
* Khi training, thuật toán này *không học* một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning)), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là [Classification](https://machinelearningcoban.com/2016/12/27/categories/" \l "classification-phan-loai) và [Regression](https://machinelearningcoban.com/2016/12/27/categories/#regression-hoi-quy).
  + Với KNN, trong bài toán Classification, label của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra label.
  + Trong bài toán Regresssion, đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết gần nhất (trong trường hợp K=1), hoặc là trung bình có trọng số của đầu ra của những điểm gần nhất, hoặc bằng một mối quan hệ dựa trên khoảng cách tới các điểm gần nhất đó.
* Một cách ngắn gọn, KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách *chỉ* dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), *không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu*.

1. **Một số khái niệm**
   1. Chuẩn hóa dữ liệu

* Khi có một thuộc tính trong dữ liệu (hay phần tử trong vector) lớn hơn các thuộc tính khác rất nhiều (ví dụ thay vì đo bằng cm thì một kết quả lại tính bằng mm), khoảng cách giữa các điểm sẽ phụ thuộc vào thuộc tính này rất nhiều. Để có được kết quả chính xác hơn, một kỹ thuật thường được dùng là *Data Normalization* (chuẩn hóa dữ liệu) để đưa các thuộc tính có đơn vị đo khác nhau về cùng một khoảng giá trị, thường là từ 0 đến 1, trước khi thực hiện KNN.



* Chuẩn hóa min-max:

A math equation with white text

Description automatically generated

1. K-Nearest Neighbor
   1. Ý tưởng

* Với một tập các ví dụ học
  + Lưu lại các ví dụ học
  + Chưa xây dựng một mô hình (mô tả) rõ ràng và tổng quát của hàm mục tiêu cần học
* Đối với một ví dụ cần phân loại/hồi quy
  + Xét quan hệ giữa ví dụ đó với các ví dụ học để gán giá trị của hàm mục tiêu (một nhãn lớp, hoặc một giá trị thực)
  1. Bài toán
* Biểu diễn đầu vào của bài toán
  + Mỗi ví dụ x được biểu diễn là một vectơ n chiều trong không gian các vecto X∈
  + x = () , trong đó ∈ R là một số thực
* Có thể áp dụng với 2 bài toán Phân loại và Hồi quy (Classification/ Regression):
  + Bài toán phân lớp (classification)
    - Hàm mục tiêu có giá trị rời rạc (a discrete-valued target function)
    - Đầu ra của hệ thống là một trong số các giá trị rời rạc đã xác định trước (một trong các nhãn lớp)
  + Bài toán hồi quy (regression)
    - Hàm mục tiêu có giá trị liên tục (a continuous-valued target function)
    - Đầu ra của hệ thống là một giá trị số thực
* Ví dụ

A diagram of a diagram of a red circle with a blue dot

Description automatically generated with medium confidence

* Xét 1 láng giềng gần nhất ⋄

Gán z vào lớp c2

* Xét 3 láng giềng gần nhất
  + - Gán z vào lớp c1
* Xét 5 láng giềng gần nhất
  + - Gán z vào lớp c1
  + Giải thuật Phân lớp k-NN
* Mỗi ví dụ học x được biểu diễn bởi 2 thành phần:
  + x = () , trong đó ∈ **R** là một số thực
  + Nhãn lớp: c ∈ **C**, với **C** là tập các nhãn lớp được xác định trước
* Giai đoạn học
  + Đơn giản là lưu lại các ví dụ học trong tập học: **D** = { x }
* Giai đoạn phân lớp: Để phân lớp cho một ví dụ (mới) z
  + Với mỗi ví dụ học x ∈ **D**, tính khoảng cách giữa x và z
  + Xác định tập **NB**(z) - các láng giềng gần nhất của z
    - Gồm k ví dụ học trong **D** gần nhất với z tính theo một hàm khoảng cách d
  + Phân z vào lớp chiếm số đông (the majority class) trong số các lớp của các ví dụ học trong **NB**(z)
  + Giải thuật hồi quy k-NN
* Mỗi ví dụ học x được biểu diễn bởi 2 thành phần:
  + x = () , trong đó ∈ **R** là một số thực
  + Giá trị đầu ra mong muốn:∈ **R** là một số thực
* Giai đoạn học
  + Đơn giản là lưu lại các ví dụ học trong tập học: **D** = { x }
* Giai đoạn phân lớp: Để phân lớp cho một ví dụ (mới) z
  + Với mỗi ví dụ học x ∈ **D**, tính khoảng cách giữa x và z
  + Xác định tập **NB**(z) - các láng giềng gần nhất của z
    - Gồm k ví dụ học trong **D** gần nhất với z tính theo một hàm khoảng cách d
  + Dự đoán giá trị đầu ra đối với z :
  + Một hay nhiều **nearest neighbor**?
* Việc phân lớp (hay dự đoán) chỉ dựa trên duy nhất một láng giềng gần nhất (là ví dụ học gần nhất với ví dụ cần phân lớp/dự đoán) thường không chính xác
* Nếu ví dụ học này là một ví dụ bất thường, không điển hình (an outlier) rất khác so với các ví dụ khác
* Nếu ví dụ học này có nhãn lớp (giá trị đầu ra) sai– do lỗi trong quá trình thu thập (xây dựng) tập dữ liệu
* Thường xét k > 1 các ví dụ học (các láng giềng) gần nhất với ví dụ cần phân lớp/dự đoán
* Đối với bài toán phân lớp có 2 lớp, k thường được chọn là một số lẻ, để tránh cân bằng về tỷ lệ các ví dụ giữa 2 lớp
  + Ví dụ: k = 3,5,7,...
  1. Hàm tính khoảng cách d
* Hàm tính khoảng cách d
  + Đóng vai trò rất quan trọng trong phương pháp học dựa trên các láng giềng gần nhất
  + Thường được xác định trước, và không thay đổi trong suốt quá trình học và phân loại/dự đoán
* Lựa chọn hàm khoảng cách d
  + Các hàm khoảng cách hình học: Dành cho các bài toán có các thuộc tính đầu vào là kiểu số thực ∈ **R**
  + Hàm khoảng cách Hamming: Dành cho các bài toán có các thuộc tính đầu vào là kiểu nhị phân ∈ {0,1}
  + Hàm tính độ tương tự Cosine: Dành cho các bài toán phân lớp văn bản ( là giá trị trọng số TF/IDF của từ khóa thứ i)

1. Các hàm tính khoảng cách hình học (Geometry distance functions)

A math equations with black text

Description automatically generated with medium confidence

1. Hàm khoảng cách
   * Hamming

A mathematical equation with numbers and symbols

Description automatically generated

* + - Đối với các thuộc tính đầu vào là kiểu nhị phân {0,1}
    - Ví dụ: x = (0,1,0,1,1)

1. Hàm tính độ tương tự
   * Cosine

A math equations with numbers

Description automatically generated with medium confidence

* + Đối với đầu vào là một vectơ các giá trị trọng số (TF/IDF) của các từ khóa
  1. Trọng số các thuộc tính
  + Làm sao để xác định các giá trị trọng số của các thuộc tính?
    - Dựa trên các tri thức cụ thể của bài toán (vd: được chỉ định bởi các chuyên gia trong lĩnh vực cra bài toán đang xét)
    - Bằng một quá trình tối ưu hóa các giá trị trọng số (vd: sử dụng một tập học để học một bộ các giá trị trọng số tối ưu)
* Xét tập **NB**(z) - gồm k ví dụ học gần nhất với ví dụ cần phân lớp/dự đoán z
  + Mỗi ví dụ (láng giềng gần nhất) này có khoảng cách khác nhau đến z
  + Các láng giềng này có cảnh hưởng khác nhau đối với việc phân lớp/dự đoán cho z
* Gọi v là hàm xác định trọng số theo khoảng cách
  + Đối với một giá trị d(x,z) - khoảng cách giữa x và z
  + v(x,z) tỷ lệ nghịch với d(x,z)
* Đối với bài toán phân lớp:

A math equations and formulas

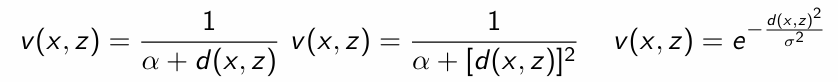
Description automatically generated with medium confidence

* Đối với bài toán dự đoán (hồi quy):

A black and white math symbols

Description automatically generated with medium confidence

* Lựa chọn một hàm xác định trọng số theo khoảng cách:



* 1. Ưu nhược điểm
* Ưu điểm
  + Độ phức tạp tính toán của quá trình training là bằng 0.
  + Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
  + Không cần giả sử gì về phân phối của các class.
* Nhược điểm của KNN
  + KNN rất nhạy cảm với nhiễu khi K nhỏ.
  + Như đã nói, KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới *từng* điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ liệu.
    - Với K càng lớn thì độ phức tạp cũng sẽ tăng lên.
  + Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN.
  1. Sử dụng khi nào?
  + Các ví dụ được biểu diễn trong không gian vectơ
  + Số lượng các thuộc tính (để biểu diễn ví dụ) là không nhiều
  + Một tập học có kích thước lớn
  1. Tăng tốc cho KNN
* Ngoài việc tính toán khoảng cách từ một điểm test data đến tất cả các điểm trong traing set (Brute Force), có một số thuật toán khác giúp tăng tốc việc tìm kiếm này. Bạn đọc có thẻ tìm kiếm thêm với hai từ khóa: [K-D Tree](http://pointclouds.org/documentation/tutorials/kdtree_search.php) và [Ball Tree](https://en.wikipedia.org/wiki/Ball_tree).