K-nearest: 1 thuật toán trong ML được gọi là K-nearest neighbor (hay KNN), một thuật toán được xếp vào loại lazy (machine) learning (máy lười học)

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. 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), 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 và Regression. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

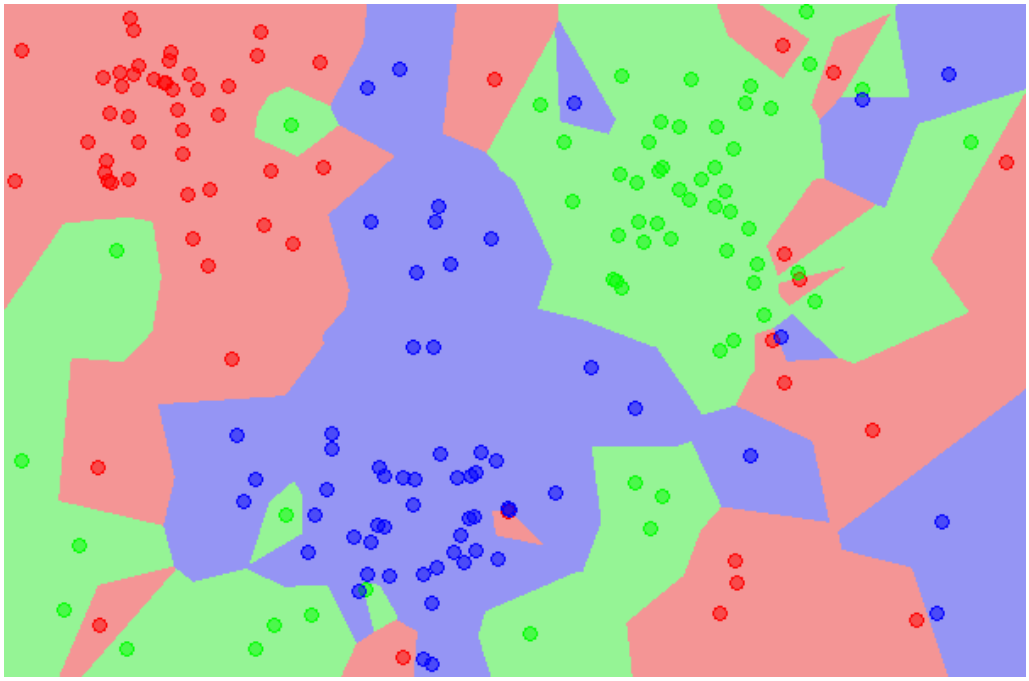
Ngôn ngữ người	**Ngôn ngữ Máy Học**	**in Machine Learning**
Câu hỏi	Điểm dữ liệu	Data point
Đáp án	Đầu ra, nhãn	Output, Label
Ôn thi	Huấn luyện	Training
Tập tài liệu mang vào phòng thi	Tập dữ liệu tập huấn	Training set
Đề thi	Tập dữ liệu kiểm thử	Test set
Câu hỏi trong dề thi	Dữ liệu kiểm thử	Test data point
Câu hỏi có đáp án sai	Nhiễu	Noise, Outlier
Câu hỏi gần giống	Điểm dữ liệu gần nhất	Nearest Neighbor

Với KNN, trong bài toán Classification, label của một điểm dữ liệu mới (hay kết quả của câu hỏi trong bài thi) 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.

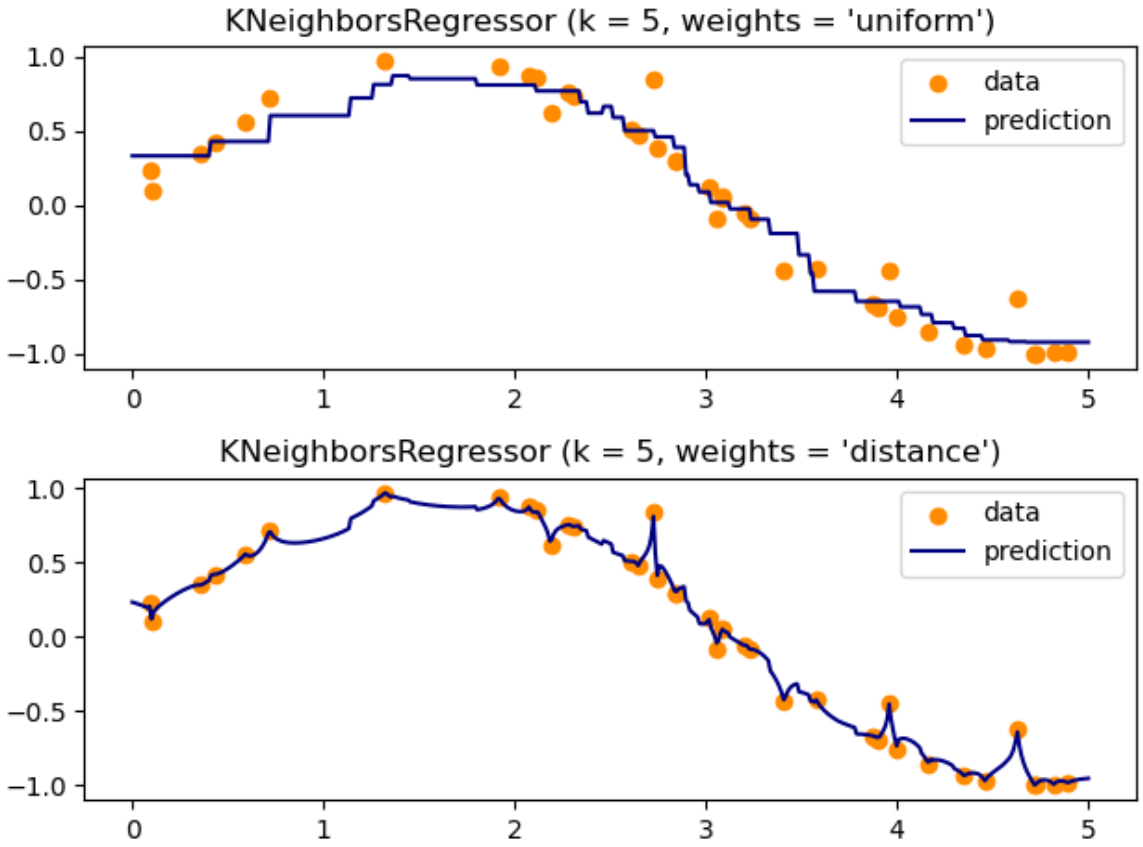
Một cách ngắn gọn, KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách *chỉ* dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), *không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu*. Hình dưới đây là một ví dụ về KNN trong classification với K = 1.



Ví dụ trên đây là bài toán Classification với 3 classes: Đỏ, Lam, Lục. Mỗi điểm dữ liệu mới (test data point) sẽ được gán label theo màu của điểm mà nó thuộc về. Trong hình này, có một vài vùng nhỏ xem lẫn vào các vùng lớn hơn khác màu. Điểm này rất có thể là nhiễu. Dẫn đến nếu dữ liệu test rơi vào vùng này sẽ có nhiều khả năng cho kết quả không chính xác

- => Nó chỉ dựa trên dữ liệu của training set để tiến hành làm output mà k quan tâm đến dữ liệu này có bị nhiễu hay các vấn đề khác hay không.
- KNN phải *nhớ* tất cả các điểm dữ liệu training, việc này không được lợi về cả bộ nhớ và thời gian tính toán - giống như khi cậu bạn của chúng ta không tìm được câu trả lời cho câu hỏi cuối cùng.

Với bài toán Regression, chúng ta cũng hoàn toàn có thể sử dụng phương pháp tương tự: ước lượng đầu ra dựa trên đầu ra và khoảng cách của các điểm trong K-lân cận. Việc ước lượng như thế nào các bạn có thể tự định nghĩa tùy vào từng bài toán.



- Đây là KNN cho bài toán Regression

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.

Có nhiều kỹ thuật chuẩn hóa khác nhau, Các kỹ thuật chuẩn hóa được áp dụng với không chỉ KNN mà còn với hầu hết các thuật toán khác.

Sử dụng các phép đo khoảng cách khác nhau

Một ví dụ đơn giản là đếm số lượng thuộc tính khác nhau giữa hai điểm dữ liệu. Số này càng nhỏ thì hai điểm càng gần nhau

Ưu điểm của KNN

1. Độ phức tạp tính toán của quá trình training là bằng 0.
2. Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
3. Không cần giả sử gì về phân phối của các class.

Nhược điểm của KNN

1. KNN rất nhạy cảm với nhiễu khi K nhỏ.
2. 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.

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 thể tìm kiếm thêm với hai từ khóa: K-D Tree và Ball Tree. Tôi xin dành phần này cho độc giả tự tìm hiểu, và sẽ quay lại nếu có dịp. Chúng ta vẫn còn những thuật toán quan trọng hơn khác cần nhiều sự quan tâm hơn.

Iris flower dataset là một bộ dữ liệu nhỏ (nhỏ hơn rất nhiều so với MNIST. Bộ dữ liệu này bao gồm thông tin của ba loại hoa Iris (một loài hoa lan) khác nhau: Iris setosa, Iris virginica và Iris versicolor.

Mỗi loại có 50 bông hoa được đo với dữ liệu là 4 thông tin: chiều dài, chiều rộng đài hoa (sepal), và chiều dài, chiều rộng cánh hoa (petal). Dưới đây là ví dụ về hình ảnh của ba loại hoa. (Chú ý, đây không phải là bộ cơ sở dữ liệu ảnh như MNIST, mỗi điểm dữ liệu trong tập này chỉ là một vector 4 chiều).

Iris setosa Iris versicolor Iris virginica

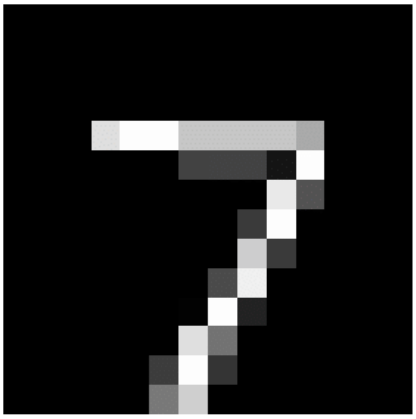
- => Bộ dữ liệu nhỏ này thường được sử dụng trong nhiều thuật toán Machine Learning trong các lớp học.

Tìm hiểu thêm về MNIST

Bộ cơ sở dữ liệu MNIST là bộ cơ sở dữ liệu lớn nhất về chữ số viết tay và được sử dụng trong hầu hết các thuật toán nhận dạng hình ảnh (Image Classification).

MNIST bao gồm hai tập con: tập dữ liệu huấn luyện (training set) có tổng cộng 60k ví dụ khác nhau về chữ số viết tay từ 0 đến 9, tập dữ liệu kiểm tra (test set) có 10k ví dụ khác nhau. Tất cả đều đã được gán nhãn.

Mỗi bức ảnh là một ảnh đen trắng (có 1 channel), có kích thước 28x28 pixel (tổng cộng 784 pixels). Mỗi pixel mang một giá trị là một số tự nhiên từ 0 đến 255. Các pixel màu đen có giá trị bằng 0, các pixel càng trắng thì có giá trị càng cao (nhưng không quá 255). Dưới đây là một ví dụ về chữ số 7 và giá trị các pixel của nó. (*Vì mục đích hiển thị ma trận pixel ở bên phải, tôi đã resize bức ảnh về 14x14*)



=

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0 222 254 254 198 198 198 198 170   0   0   0]
 [  0   0   0   0   0   0  66  67  67  21 254   0   0   0]
 [  0   0   0   0   0   0   0   0   0 233  83   0   0   0]
 [  0   0   0   0   0   0   0   0  59 254   0   0   0   0]
 [  0   0   0   0   0   0   0   0 205  58   0   0   0   0]
 [  0   0   0   0   0   0   0  75 240   0   0   0   0   0]
 [  0   0   0   0   0   0   3 254  35   0   0   0   0   0]
 [  0   0   0   0   0   0 224 115   0   0   0   0   0   0]
 [  0   0   0   0   0  61 254  52   0   0   0   0   0   0]
 [  0   0   0   0   0 121 207   0   0   0   0   0   0   0]]
```

Tham khảo: web: https://machinelearningcoban.com/2017/01/08/knn/