1. K-nearest neighbors

1.1 Ý tưởng

K-nearest neighbors (KNN) la một trong những thuật toán supervised learning (học giám sát) đơn giản. Khi huấn luyện, thuật toán này không học từ dữ liệu huấn luyện mà nhớ lại toàn bộ dữ liệu đó và tính toán được thực hiện khi nó cần dự đoán một đầu ra mới (lazy learning)

Áp dụng KNN với bài toán phân lớp, nhãn của một điểm dữ liêu mới có thể suy ra từ nhãn của các điểm dữ liệu gần nó nhất theo cách bầu chọn đa số, hoặc có thể suy ra bằng cách đánh trọng số cho các điểm gần nó (điểm càng gần, trọng số càng cao và ngược lại) rồi suy ra kết quả. Trong regression, đầu ra của một điểm dữ liệu mới sẽ bằng đầu ra của điểm dữ liệu gần nó nhất hoặc trung bình đầu ra của các điểm gần nó (có đánh trọng số).

Như vậy, ***KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách dựa vào thông tin của K điểm dữ liệu gần nó nhất***.

1.2 Phân tích toán học

Vì mọi tính toán của thuật toán này được thực hiện ở bước kiểm thử nên có 2 vấn đề cần lưu ý

- Mỗi điểm dữ liệu được thể hiện bằng một vector đặc trưng, khoảng cách giữa 2 điểm dữ liệu là khoảng cách giữa 2 vector đó (dung nỏm để tính, phổ biến nhất là norm 2 – khoảng cách Euclid)
- Với các bài toán có tập huấn luyện lớn và vector đặc trung có kích thước lớn thì khối lượng tính toán cực kỳ lớn cho mỗi điểm dữ liệu mới. Cần phải tìm cách tính toán tối ưu cho thuật toán này

Khoảng cách từ một điểm tới từng điểm trong mọt tập hợp

Khoảng cách Euclid từ một điểm $\mathbf{z}$ tới một điểm $\mathbf{x}_i$ trong tập huấn luyện là $||\mathbf{z} - \mathbf{x}_i||_2$. Để đơn giản hơn, ta tính $(||\mathbf{z} - \mathbf{x}_i||_2)^2$:

$$(||\mathbf{z} - \mathbf{x}_i||_2)^2 = (\mathbf{z} - \mathbf{x}_i)^T(\mathbf{z} - \mathbf{x}_i) = (||\mathbf{z}||_2)^2 + (||\mathbf{x}_i||_2)^2 + 2\mathbf{x}_i^T\mathbf{z} \quad (1.0)$$

Vì chỉ cần tìm ra khoảng cách gần nhất nên ta chỉ quan tâm tới các biến số thay đổi là $(||\mathbf{x}_i||_2)^2$ và $2\mathbf{x}_i^T\mathbf{z}$.

Khoảng cách giữa từng cặp điểm trong hai tập hợp

Trong các bài toán thực tế, ta không chỉ tính khoảng cách từ một điểm $\mathbf{z}$ đến một tập hợp $\mathbf{X}$ mà phải tính khoảng cách từ các điểm của một tập hợp $\mathbf{Z}$ đến tập hợp $\mathbf{X}$**.** Khối lượng tính toán khi tính toán thông thường với mỗi tập hợp gồm 1000 phần tử đã là 1 triệu. Do đó, cần phải tìm ra cách tính toán tối ưu. Một trong các cách là dùng biểu thức (1.0) cho X và Z

2. K-means clustering

2.1 Ý tưởng

K-means clustering (phân nhóm K-means) là thuật toán phân các điểm dữ liệu không biết nhãn thành các cụm khác nhau sao cho dữ liệu trong một cụm có những tính chất giống nhau. Với mỗi điểm dữ liệu là một vector đặc trưng thì bài toán trở thành chia tập hợp ban đầu thành các cụm có vector đặc trưng gần nhau. Thông thưởng, khoảng cách được đo bằng norm (phổ biến nhất là norm 2 – khoảng cách Euclid)

2.2Phân tích toán học

Mục đích của thuật toán K-means clustering là từ dữ liệu đầu vào và số nhóm cần chia, xác định centroid của mỗi nhóm và phân các điểm dữ liệu vào các nhóm tương ứng.
Giả sử, ban đầu có N điểm dữ liệu chưa có nhãn (chưa biết nhóm) được ghép lại thành X = [x1, x2,…xN] và số nhóm K < N. Ta cần tìm các centroid m1, m2, …, mK của K nhóm được đánh số từ 1 đến K và nhãn (nhóm) của từng điểm dữ liệu.

Để biểu diễn label của một điểm dữ liệu x_i ta thường dùng kỹ thuật one-hot coding. Dùng một vector hàng y_i (1xK) để biểu diễn, trong đó các phần tư đều bằng 0 trừ phần tử thứ k bằng 1 có nghĩa là nhãn của x_i là nhóm thứ k. Với X = [x1, x2,…xN], ta được Y = [y1; y2;…;yN]

$$y_{ij} \in \{0, 1\}, \ \forall i, j; \quad \sum_{j=1}^{K} y_{ij} = 1, \ \forall i$$

(y_{ij} là phần tử ở hàng thứ I, cột j của Y)

Hàm mất mát và bài toán tối ưu

Nếu gọi m_k là centroid của mỗi cluster và thay thế tất cả các điểm x_I thuộc cluster thứ k bằng m_k thì với mỗi điểm dữ liệu x_i sai số sẽ là ($x_i – m_k$). Ta mong muốn sai số này gần về vector không, tức x_i gần với m_k, khi đó ta dùng (bình phương) khoảng cách Euclid để tính khoảng cách giữa 2 vector:

$$\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = y_{ik}\|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

(y_{ik} = 1, y_{ij} = 0 với j != k)
Suy ra sai số trung bình (hàm mất mát) sẽ là:

$$\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Trong đó M = [m1, m2, …, mK] là ma trận tạo bởi K centroid. Bài toán tối ưu là:

$$\mathbf{Y}, \mathbf{M} = \underset{\mathbf{Y},\mathbf{M}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Thuật toán tối ưu hàm mất mát

Tròn hàm mất mát của bài toán có 2 biến Y và M, nên để tìm được nghiệm gần đúng ta có thể dùng một kỹ thuật đơn giản là xem kẽ tìm Y va M khi biến còn lại cố định cho tới khi hàm mất mát hội tụ.

Cố định M, tìm Y.

Giả sử đã tìm đc các centroid, tìm các label vector để hàm mất mát đạt giá trị nhỏ nhất. Điều này có nghĩa là tìm cluster cho mỗi điểm dữ liệu. Đối với một ddiierm dữ liệu x_i, ta có:

$$\mathbf{y}_i = \underset{\mathbf{y}_i}{\operatorname{argmin}} \frac{1}{N} \sum_{j=1}^{K} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Để y_i đat GTNN thì khoảng cách giữa x_i và m_j phải là nhỏ nhất, nghĩa là ***x_i thuộc về cluster có centroid gần nó nhất.***

Cố định Y, tìm M

Giả sử đã tìm được cluster cho từng điểm, tìm centroid mới cho mỗi cluster để hàm mất mát đạt GTNN.

$$\mathbf{m}_j = \underset{\mathbf{m}_j}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2.$$

$$\nabla_{\mathbf{m}_j} l(\mathbf{m}_j) = \frac{2}{N} \sum_{i=1}^{N} y_{ij}(\mathbf{m}_j - \mathbf{x}_i) = \mathbf{0} \Leftrightarrow \mathbf{m}_j \sum_{i=1}^{N} y_{ij} = \sum_{i=1}^{N} y_{ij}\mathbf{x}_i \Leftrightarrow \mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} y_{ij}}$$

Từ đây suy ra, ***m_j chính là trung bình công của các điểm thuộc về cluster đó***.

3. Naïve Bayes Classifier

3.1 Ý tưởng

Xét bài toán phân lớp với C class khác nhau, thay vì tìm chính xác label của một điểm dữ liệu mới x, ta có thể đi tìm sác suất để đầu ra đó rơi vào mỗi class: p(y = c|x) viết gọn là p(c|x) (xác suất để đầu ra là c biết vector đầu vào là x). Với biểu thức này, ta có thể xác định label của một điểm dữ liệu bằng cách chọn class có xác suất cao nhất:

$$c = \underset{c}{\operatorname{argmax}}\, p(c|\mathbf{x}) = \underset{c}{\operatorname{argmax}} \frac{p(\mathbf{x}|c)p(c)}{p(\mathbf{x})} = \underset{c}{\operatorname{argmax}}\, p(\mathbf{x}|c)p(c)$$

p(c) là xác suất để 1 điểm bất kỳ rơi vào class c. Có thể được xác định bằng Maximum Likelihood Estimation hoặc Maximum A Posteriori tùy vào kích thước của training set.
p(x|c) là phân phối các điểm dữ liệu trong class c. Để dễ dàng tính toán, ta giả sử các điểm dữ liệu độc lập với nhau, khi đó:

$$p(\mathbf{x}|c) = p(x_1, x_2, \ldots, x_d|c) = \prod_{i=1}^{d} p(x_i|c)$$

Ở bước huấn luyện, p(c) và p(x|c) được tính dựa vào dữ liệu trong tập training set dựa vào MLE hoặc MAP

Ở bước kiểm thử, label của một điểm dữ liệu x mới được xác định:

$$c = \underset{c \in \{1,\ldots,C\}}{\operatorname{argmax}}\, p(c) \prod_{i=1}^{d} p(x_i|c)$$

Có thể biến đổi thành:

$$c = \underset{c \in \{1,\ldots,C\}}{\operatorname{argmax}} \left(\log(p(c)) + \sum_{i=1}^{d} \log(p(x_i|c)) \right)$$

3.2 Các phân phối thường dùng trong NBC

- **Gaussian naïve Bayes**

  Mô hình này được sử dụng trong loại dữ liệu có thành phần là các biến liên tục. Với mỗi chiều i của class c, x_i tuân theo phân phối chuẩn có kỳ vọng μci và phương sai σci 2

  $$p(x_i|c) = p(x_i|\mu_{ci}, \sigma_{ci}^2) = \frac{1}{\sqrt{2\pi\sigma_{ci}^2}} \exp\left(-\frac{(x_i - \mu_{ci})^2}{2\sigma_{ci}^2}\right)$$

  Trong đó bộ tham số $\theta = \{\mu ci; \sigma ci$ 2 $\}$ được xác định bằng MLE dựa trên dữ li ệu của training set

- **Multinomial naïve Bayes**

  Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà vector đặc trưng được xây dựng trên mô hình bag of word (BoW: mỗi văn bản được biểu diễn bởi 1 vector có độ dài d là số từ xuất hiện trong từ điển, giá trị của thành phần thứ i chính là số làn xuất hiện của từ thứ i trong văn bản). Khi đó p(x_i|c) tỉ lệ với tần suất xuất hiện của thừ thứ i trong các văn bản của class c

  $$\lambda_{ci} = p(x_i|c) = \frac{N_{ci}}{N_c}$$

  Trong đó:

  N_{ci} là tổng số lần từ thứ I xuất hiện trong các văn bản của class c

  N_c là tổng số từ (kể cả lặp) xuất hiện trong class c

  Nhưng nếu gặp một từ chưa bao giờ xuất hiện thì xác suất của từ đó là 0 trong class c, dẫn đến vế phải bằng 0. Điều này là không hợp lý, có một kỹ thuật giúp ta giải quyết vấn đề này gọi là Laplace smoothing:

  $$\hat{\lambda}_{ci} = \frac{N_{ci} + \alpha}{N_c + d\alpha}$$

  Với a là một số dương, thường bằng 1 để tránh trường hợp tử số bằng 0. Mẫu số công vào da để đảm bảo tổng xác suất bằng 1.

- **Bernoulli Naïve Bayes**
  Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị nhị phân (0 or 1). Ví dụ, cùng với loại văn bản nhưng thay vì đếm, ta chỉ quan tâm từ đó có xuất hiện hay không. $p(x_i|c)$ được tính bằng

$$p(x_i|c) = p(i|c)x_i + (1 - p(i|c))(1 - x_i)$$

  Với p(i|c) là xác suất từ thứ i xuất hiện trong các văn bản của class c.