Lecture 3: Locally Weighted & Logistic Regression

Locally Weighted Regression

- Parametric learning algorithm: khớp một tập các tham số cố định với dữ liệu
- Non-parametric learning algorithm: lượng tham số mà máy lưu trữ tùy thuộc vào số lượng dữ liệu trong tập dữ liệu, lượng tham số này có thể tăng một cách tuyến tính theo lượng dữ liệu
- Đối với Locally Weighted Regression, bài toán có thể được phát biểu như sau:

Khớp tập tham số θ với một hàm

$$\sum_{i=1}^{m} w^i (y^i - \theta^T x^i)^2$$

trong đó w^i là hàm trọng số với

$$w^i = exp(-\frac{(x^i - x)^2}{2})$$

- Với công thức trên, điểm dữ liệu huấn luyện càng gần điểm dữ liệu kiểm tra thì trọng số nó càng cao và ngược lại.
- Để quyết định đường cong của mô hình, một tham số τ được sử dụng trong khi tính trọng số như sau:

$$w^i = exp(-\frac{(x^i - x)^2}{2\tau^2})$$

- Giá trị τ quá lớn có thể dẫn đến underfitting và giá trị nhỏ quá có thể dẫn đến overfitting.

Probabilistic Interpretation: Why Squared Error

Sai số bình phương phát sinh một cách tự nhiên khi giả định các lỗi phân phối chuẩn trong một mô hình hồi quy. Bằng cách tối đa hóa khả năng (hoặc tương đương với việc tối thiểu hóa log-likelihood âm), chúng ta kết thúc bằng việc tối thiểu hóa tổng các sai số bình phương. Ý nghĩa xác suất này lý giải cho việc sử dụng sai số bình phương như một hàm mất mát trong nhiều bối cảnh hồi quy.

Logistic Regression

- Đôi khi chúng ta muốn hàm hypothesis h có thể trả về các kết quả nằm trong khoảng [0, 1]
- Xác định công thức sau:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Đây là hàm sigmoid, hay logistic function được sử dụng để chuyển đổi giá trị trong bất kỳ phạm vi nào về khoảng từ 0 đến 1.

- Khi đó hàm xác suất có thể được viết như sau
-

$$P(y = 1|x;\theta) = h_\theta(x)$$

$$P(y = 0|x;\theta) = 1 - h_\theta(x)$$

Đây là cách chúng ta mô hình hóa xác suất của y tùy thuộc vào x; cho y = 1 và y = 0.

Hay:

$$P(y|x;\theta) = (h_\theta(x))^y(1 - h_\theta(x))^{1-y}$$

Khi đó likelihood của tham số là

$$L(\theta) = \prod_{i=1}^{m} P(y^i|x^i;\theta) = \prod_{i=1}^{m}(h_\theta(x^i))^{y^i}(1 - h_\theta(x^i))^{1-y^i}$$

Và log-likelihood sẽ là:

$$l(\theta) = \sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i))$$

Hàm likelihood đo lường khả năng của một tập tham số nhất định trong mô hình thống kê dựa trên dữ liệu mẫu đã quan sát được. Sử dụng log-likelihood thay vì likelihood giúp việc tối ưu hóa mô hình (tìm ra bộ tham số tốt nhất) trở nên dễ dàng hơn, đặc biệt khi làm việc với các mô hình phức tạp.

- Để tìm ra bộ tham số θ cho mô hình và minimize giá trị $L(\theta)$ chúng ta sẽ sử dụng thuật toán Batch Gradient Ascent theo công thức

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i$$

Trong đó, α là tốc độ học, và cập nhật này được lặp đi lặp lại cho đến khi hội tụ.

Newton method

- Đối với Gradient Ascent, thông thường cần rất nhiều vòng lặp để đạt được hội tụ, do đó có một phương pháp là Newton method nhằm giảm số vòng lặp xuống
- Phương pháp Newton sử dụng đạo hàm thứ hai (đạo hàm của gradient) để nhanh chóng tìm ra điểm tối ưu. Đây là một bước nhảy vọt trong việc giảm thời gian tính toán và nâng cao hiệu suất của mô hình.

Để hiểu rõ hơn về phương pháp Newton, chúng ta cần biết rằng trong thuật toán này, chúng ta sẽ thay đổi giá trị của θ theo công thức:

$$\theta := \theta - H^{-1} \nabla_\theta l(\theta)$$

Trong đó:

- H là ma trận Hessian, là đạo hàm bậc hai của hàm mất mát tại θ
- $\nabla_\theta l(\theta)$ là đạo hàm bậc nhất của hàm mất mát tại θ

Như vậy, thay vì chỉ sử dụng đạo hàm bậc nhất (gradient), phương pháp Newton còn sử dụng thêm đạo hàm bậc hai (Hessian), giúp cho việc tìm điểm tối ưu trở nên nhanh chóng hơn rất nhiều.

Tuy nhiên, việc tính toán ma trận Hessian có thể khá phức tạp và tốn kém về mặt tính toán, đặc biệt khi số lượng tham số lớn. Do đó, phương pháp Newton thường chỉ được sử dụng khi số lượng tham số không quá lớn.