Buổi học MaSSP số 2

July 2019

Nguyen Dinh Hieu

1 Unified Framework

Với bộ dữ liệu đầu vào X và bộ dữ liệu đầu ra Y , cần xây dựng một phương thức giúp máy tính có thể nhận vào 1 dữ liệu x bất kì và đưa ra kết quả chính xác như ta mong muốn. Đầu tiên, ta dùng phương pháp PCA để giảm số chiều dữ liệu cần phải làm việc (nhưng vẫn giữ lại độ chính xác cao) và thu được các vector coordinates dựa trên những basis functions/features cho sẵn. Sau đó, dùng các thuật toán khác nhau về regression hoặc classification để thu được đầu ra dưới dạng mong muốn.

2 PCA

Biểu diễn 1 dữ liệu đầu vào X theo dạng

$$X = X_1 y_1 + X_2 y_2 + ... + X_n y_n$$

với $X_1, X_2, ...$ là các basis functions cho trước, còn $y_1, y_2, ...$ là các coordinates tương ứng. Đồng thời, ta có thể giảm số components, giúp giảm độ phức tạp tính toán (do chỉ cần quan tâm đến các chi tiết liên quan tới basis functions).
Trong PCA, mục tiêu của ta là tìm ra các eigenvectors và eigenvalues cho bộ dữ liệu nhập vào (ví dụ, có n features muốn được xem xét và có k mẫu cho trước $x_1, x_2, ..., x_k$ thì bộ dữ liệu đầu vào là ma trận k*n). Để chạy hàm PCA, đầu tiên với mỗi dimensio/feature phải trừ tât cả các giá trị cho mean của chúng.

Sau đó, ta tính co-variance matrix/ ma trận khoảng cách giữa các dimensions; với n dimensions sẽ thu được ma trận n*n. Vì đây là ma trận vuông nên dễ dàng thu được các eigenvectors và eigenvalues tương ứng cho ma trận này. Khi giảm số components/dimension ta giữ lại những eigenvectors có eigenvalues tương ứng lớn nhất/ gây ảnh hưởng nhiều nhất tới sự phân bố của dữ liệu.

Kết quả thu được: Vẫn là bộ dữ liệu đầu vào (X) nhưng được biểu diễn bởi ít dimension/features hơn. Tuy vậy, do trong thuật toán ta đã chọn những thành phần quan trọng nhất nên chỉ với những features này ta vẫn có thể khôi phục lại dữ liệu ban đầu với độ chính xác cao (Sai số ở đây được gọi là Transformation Error).

Thư viện pca của scikit-learn giúp người dùng chạy thuật toán PCA. Nó cũng đồng thời giúp người dùng thu được các coordinates của bộ dữ liệu X ứng với các basis functions được đưa ra, dưới dạng vector tọa độ. Vector này sau đó có thể dùng ứng dụng vào nhiều thuật toán machine learning khác nhau.

3 Linear regression

Thuật toán linear regression (tìm hệ số thực cho hàm số tuyến tính)giúp ta gán các hệ số $w_0, w_1, ...$ cho hàm số cho trước. Với bộ dữ liệu/experience có sẵn, ta dùng linear regression tìm các hế số để chúng cho ra kết quả y-head gần với kết quả chính xác là y.Nói cách khác, hàm số mất măt (Loss Function) của y-hat và y được cực tiểu hóa. Để tìm cực trị hàm số(cực trị xảy ra khi đạo hàm hàm số bằng 0), thường dùng Gradient Descent nếu không tính được đạo hàm trực tiếp. TUy nhiên, ta tính được trực tiếp các vector hệ số W trong trường hợp này. Công thức:

$$\mathbf{w} = \mathbf{A}^{\dagger}\mathbf{b} = (\bar{\mathbf{X}}^T\bar{\mathbf{X}})^{\dagger}\bar{\mathbf{X}}^T\mathbf{y}$$

Lắp hệ số vào hàm y=f(x) và ta có kết quả.

4 Logistic regression

Logistic regression. Khi ta làm việc với các bài toán về xác suất (thường rơi vào loại classification hơn là regression: xác định xem 1 khối hộp có màu gì trong các màu cho sẵn, dự đoán 1 học sinh rơi vào nhóm đỗ hay nhóm trượt môn dựa theo dữ liệu cho trước...) Linear regression và PLA không phù hợp/ không áp dụng được. Với những bài toán như vậy, ta cần dùng những hàm số không tuyến tính (non-linear), trả về giá trị thực trong khoảng $[0, 1]$, có đạo hàm đơn giản, là hàm liên tục. Hàm sigmoid thỏa mãn các tính chất đó, và xuất hiện nhiều trong tự nhiên dưới dạng learning curves. Do đó nó được sử dụng rộng rãi.

Xây dựng hàm: có công thức sau cho biết xác suất xảy ra sự kiên y_i biết x_i và w : $P(y_i|\mathbf{x}_i; \mathbf{w}) = z_i^{y_i}(1-z_i)^{1-y_i}$ với $z_i = f(\mathbf{w}^T\mathbf{x}_i)$
Do các điểm dữ liệu độc lập với nhau nên xác suất tất cả các sự kiện xảy ra bằng tích từng xác suất riêng lẻ. Ta lấy log để chuyển dấu nhân thành dấu cộng giúp tránh số to và sai số không cần thiết.

$$J(\mathbf{w}) = -\log P(\mathbf{y}|\mathbf{X}; \mathbf{w}) \tag{1}$$

$$= -\sum_{i=1}^{N}(y_i \log z_i + (1-y_i)\log(1-z_i)) \tag{2}$$

tới đây dùng Stochastic Gradient Descent. Thay vì GD bình thường khi ta tính tổng đạo hàm của từng điểm dữ liệu riêng biệt, SGD lấy ngẫu nhiên 1 điểm rồi tính đạo hàm chỉ ở điểm đó. Trong thực tế SGD chạy nhanh hơn GD rất nhiều nhưng vẫn có độ chính xác cao.

Cồng thức cập nhật giá trị của logistic regression:

$$\mathbf{w} = \mathbf{w} + \eta(y_i - z_i)\mathbf{x}_i$$

5 Softmax

Là quá trình renormalization (bình thường hóa) cho các vector đầu vào để biến chúng thành probability vector (vector xác suất) với dữ liệu trong khoảng [0..1] và tổng các phần tử là 1. Lấy dữ liệu Z nhân với ma trận trọng số W ta được W_z. Ta dùng softmax để classify thông qua probability.

Công thức softmax:

$$P_i = softmaxx_i = \frac{\sigma(y_i)}{\sum_{j=1}^{d} \sigma(y_i)}$$

6 PLA

PLA : Thuật toán classification, giúp tạo các đường linear boundary làm ranh giới giữa các class khác nhau.

Cách làm:

1.Chọn ngẫu nhiên một vector hệ số w với các phần tử gần 0.

2.Duyệt ngẫu nhiên qua từng điểm dữ liệu x_i. Nếu x_i phân lớp đúng thì giữ nguyên, không thì cập nhật boundary sao cho x_i được phân lớp đúng:

$$\mathbf{w} = \mathbf{w} + y_i \mathbf{x}_i$$

3. Lặp lại bước 2 đến khi mọi điểm của bộ dữ liệu đều được phân lớp đúng, ta thu được vector w là kết quả.