Xử Lí Đa Chiều - Programming Exercise - 04

T-sne (t-distributed stochastic neighbor embedding)

Sơ qua về SNE

SNE bắt đầu bằng việc chuyển các khoảng cách Euclidean nhiều chiều giữa các điểm dữ liệu thành xác suất có điều kiện diễn tả tính giống nhau (similarties).

Tính giống nhau của điểm dữ liệu x_j với điểm dữ liệu x_i là xác suất có điều kiện $p_{j|i}$ mà x_i sẽ chọn x_j là lân cận khi các lân cận được chọn tỉ lệ với mật độ xác suất của chúng dưới một Gaussian có tâm tại x_i:

$$p_{j|i} = \frac{exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_j\|^2/2\sigma_i^2)}$$

Với phần ở chiều thấp hơn y_i và y_j của các điểm dữ liệu có nhiều chiều x_i, x_j, ta có thể tính xác suất có điều kiện về tính giống nhau, ký hiệu là $q_{j|i}$:

$$q_{j|i} = \frac{exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} exp(-\|y_i - y_j\|^2}$$

Nếu các điểm chiếu y_i, y_j thể hiện đúng tính giống nhau giữa 2 điểm dữ liệu nhiều chiều x_i, x_j, xác suất có điều kiện $p_{j|i}, q_{j|i}$ sẽ bằng nhau. Do đó, SNE sẽ tìm biểu diễn dữ liệu ít chiều mà làm minimize sự sai khác giữa $p_{j|i}, q_{j|i}$. SNE sẽ minimize tổng của Kullback - Leibler tới tất cả các điểm dữ liệu sử dụng gradient descent.

Hàm mất mát (cost function) của SNE tập trung vào việc giữ cấu trúc địa phương (local structure) của các dữ liệu trên map.

SNE thực hiện 1 phép tìm kiếm nhị phân cho giá trị của σ_i mà cho P_i với perplexity cố định. Perplexity được định nghĩa là:

$$Perp(P_i) = 2^{H(P_i)}$$

trong đó, $H(P_i)$ là shannon entropy của P_i được đo bằng bits:

$$H(P_i) = -\sum_j p_{j|i} log_2(p_{j|i})$$

Perplexity có thể diễn giải là 1 độ đo ổn định về số lân cận hiệu quả.

Để minimize hàm cost của sne, ta sử dụng gradient descent:

$$\frac{\partial C}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

Để tăng tốc độ tối ưu và tránh điểm cực tiểu lân cận không tốt, ta có thể thêm 1 cụm tương đối lớn momentum.

$$Y^{(t)} = Y^{(t-1)} + \beta\frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$$

trong đó, $Y^{(t)}$ là solution ở bước thứ t, β là learning rate và $\alpha(t)$ cho biết momentum ở bước t.

t-sne

t-sne khác với sne ở 2 điểm:

- t-sne sử dụng phiên bản đối xứng của hàm mất mát của sne để có gradient đơn giản hơn.
- Nó sử dụng phân phối student thay vì Gaussian để tính sự giống nhau giữa 2 điểm ở không gian ít chiều.

Sử dụng phân phối student, ta có xác suất hợp q_{ij} là:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

Gradient lúc này là:

$$\frac{\partial C}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

Thuật toán t-sne:

- Tính $p_{j|i}$ với perplexity perp
- đặt $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
- Sample solution ban đầu $Y^{(0)} = \{y_1, y_2, ..., y_n\}$ từ $N(0, 10^{-4}I)$
- với t=1 chạy tới T:

  tính q_{ij} đã cho công thức ở trên của t-sne

  tính gradient $\frac{\partial C}{\partial y_i}$ theo công thức ở trên của t-sne

  đặt $Y^{(t)} = Y^{(t-1)} + \beta\frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$

Có thể sử dụng 1 cụm momentum để giảm số bước cần thiết.

Có thể tăng tốc bằng sử dụng learning rate tùy chỉnh, tăng khi learning rate ở hướng mà gradient ổn định.