Dự đoán thuê bao ngủ đông trong mạng viễn thông

Trình bày: Nguyễn Văn Sơn

Ha Noi
01/08/2018

Mục lục

1. Mô tả bài toán

- Dữ liệu bao gồm 500.000 users, với các thông số tiêu dùng trong 9 tuần.

- Các thông số này được tính theo từng giờ, mỗi giờ bao gồm: data (số MB lưu lượng), voice (số phút gọi), sms (số tin nhắn).

1. Mô tả bài toán

- Một user tại một ngày được gọi là inactive các dữ liệu tiêu dùng nhỏ hơn các ngưỡng cho trước…

- Một user trong một tuần được gọi là inactive nếu số ngày inactive trong tuần đó là >3

1. Mô tả bài toán

- Mỗi user_i sẽ có 9 sample tương ứng với 9 tuần, mỗi sample với 3x24x7=504 chiều, có dạng:

$$x_i^{(t)} = (x_{i0}^{(t)}, x_{i1}^{(t)}, \dots, x_{i503}^{(t)})$$

  - Trong đó chẳng hạn $\{x_{i0}^{(t)}, x_{i1}^{(t)}, x_{i2}^{(t)}\}$ tương ứng là dữ liệu tiêu dùng data(MB), voice(minute), sms(#) của user_i tại giờ đầu tiên của ngày thứ nhất của tuần t
  - Các bộ ba của các giờ tiếp theo, các ngày tiếp theo sẽ được concatenate liên tiếp thành 1 vecto 504 chiều như trên

1. Mô tả bài toán

- Mỗi sample $x_i^{(t)}$ sẽ tương ứng với một nhãn $y_i^{(t)} = 0$ hoặc 1 tương ứng tuần $t + 1$ thì user_i là inactive/active

- Bài toán đặt ra: giả sử tại tuần T, quan sát được dữ liệu tiêu dùng của user_i là $x_i^{(T)}$ khi đó cần dự đoán vào tuần T+1, user_i sẽ active (tương ứng $y_i^{(T)} = 1$) hay inactive (tương ứng $y_i^{(T)} = 0$)

2. Xây dựng mô hình

- Ý tưởng: từ các cặp $(x_i^{(t)}, y_i^{(t)})$, xây dựng một bộ phân lớp gồm hai nhãn là active/inactive
- Chia dữ liệu trong 9 tuần thành hai phần: một phần gồm 7 tuần từ tuần 1 đến tuần 7 để training, và phần còn lại là tuần 8 và 9 để testing
- Sau khi training model, dữ liệu tuần thứ T được đưa vào để đưa ra dự đoán cho tuần T+1. sau 1 vài tuần dự đoán, có thể ta phải re-train model hoặc có thể dùng các phương pháp học streaming để update model theo thời gian.

3. Một số thuật toán thực hiện

- **Bỏ yếu tố thời gian:**
  - Với dữ liệu trong 7 tuần để training, có 7x500000=3500000 train samples, tương ứng 2 tuần để testing, có 2x500000=1000000 test samples → bỏ đi yếu tố thời gian
  - Thực hiện các thuật toán với thư viện sklearn: Logistic Regression, SVM, Random Forest
  - Ưu điểm: Đơn giản, dễ thực hiện
  - Nhược điểm: Thời gian tính toán lâu, phải re-train model, chưa phản ánh được chuỗi thời gian

3. Một số thuật toán thực hiện

- **Sử dụng yếu tố thời gian:**

Tham khảo paper: *Predicting User Activity Level in Social Networks*

Xây dựng loss function có dạng:

$$J = \sum_{i=1}^{N} \sum_{t=1}^{T} e^{\alpha(t-T)} l\left(y_i^{(t)}, (w_0 + w_i)^T x_i^{(t)}\right) + \gamma_0 \|w_0\|_2^2 + \gamma \sum_{j=1}^{N} \|w_j\|_2^2$$

- w_0 nắm bắt được những thông tin/đặc tính chung giữa các người dùng
- w_i đặc trưng cho từng người dùng
- Thành phần time_decay $e^{\alpha(t-T)}$ điều khiển sử ảnh hưởng của dữ liệu theo thời gian

3. Một số thuật toán thực hiện

- **Sử dụng yếu tố thời gian:**

$$J = \sum_{i=1}^{N} \sum_{t=1}^{T} e^{\alpha(t-T)} l\left(y_i^{(t)}, \left(w_0 + w_i\right)^T x_i^{(t)}\right) + \gamma_0 \left\|w_0\right\|_2^2 + \gamma \sum_{j=1}^{N} \left\|w_j\right\|_2^2$$

  - Chọn l là hàm cross-entropy:

$$l\left(y_i^{(t)}, \left(w_0 + w_i\right)^T x_i^{(t)}\right) = -y_i^{(t)} \log \widehat{y_i^{(t)}} - \left(1 - y_i^{(t)}\right) \log\left(1 - \widehat{y_i^{(t)}}\right)$$

với $$\widehat{y_i^{(t)}} = \frac{1}{1 + e^{-\left(w_0 + w_i\right)^T x_i^{(t)}}}$$

3. Một số thuật toán thực hiện

- **Sử dụng yếu tố thời gian:**

$$J = \sum_{i=1}^{N} \sum_{t=1}^{T} e^{\alpha(t-T)} l\left(y_i^{(t)}, (w_0 + w_i)^T x_i^{(t)}\right) + \gamma_0 \|w_0\|_2^2 + \gamma \sum_{j=1}^{N} \|w_j\|_2^2$$

  - Chọn l là hàm hinge loss:

$$l\left(y_i^{(t)}, (w_0 + w_i)^T x_i^{(t)}\right) = \max\left(0, 1 - y_i^{(t)}\left[(w_0 + w_i)^T x_i^{(t)} + b\right]\right)$$

  - Chọn l là hàm square-hinge loss

$$l\left(y_i^{(t)}, (w_0 + w_i)^T x_i^{(t)}\right) = \max\left(0, 1 - y_i^{(t)}\left[(w_0 + w_i)^T x_i^{(t)} + b\right]\right)^2$$

3. Một số thuật toán thực hiện

- **Sử dụng yếu tố thời gian:**
  - Chọn l là hàm (square) hinge loss:

$$l\left(y_i^{(t)}, (w_0 + w_i)^T x_i^{(t)}\right) = \max\left(0, 1 - y_i^{(t)}\left[(w_0 + w_i)^T x_i^{(t)} + b\right]\right)$$

$$l\left(y_i^{(t)}, (w_0 + w_i)^T x_i^{(t)}\right) = \max\left(0, 1 - y_i^{(t)}\left[(w_0 + w_i)^T x_i^{(t)} + b\right]\right)^2$$

  - Hàm mục tiêu là một hàm convex, tuy nhiên có rất nhiều vùng flat (bằng phẳng), gradient tại những điểm thuộc vùng này đều bằng 0 → stuck → không update w theo gradient descent được (kết quả thực nghiệm rất kém)
  - Sử dụng ***Perturbed Gradient Descent***: cộng thêm một nhiễu Gaussian vào mỗi bước cập nhật khi $|gradient(w)| \approx 0$:

$$w_i^{t+1} = w_i^t - \eta * gradient\left(w_i^t\right) + Gauss(\mu, \sigma^2)$$

3. Một số thuật toán thực hiện

- **Sử dụng yếu tố thời gian với ý tưởng ensemble method**

Chia dữ liệu training thành từng tuần: $\left(X^{(t)}, y^{(t)}\right)_{t=1,2,\ldots,T}$

Trong đó: $X^{(t)} = \begin{pmatrix} x_1^{(t)} \\ x_2^{(t)} \\ \vdots \\ x_N^{(t)} \end{pmatrix}_{N \times d}$ là ma trận inputs Nx504 tại tuần thứ t

tương ứng $y^{(t)} = \left(y_1^{(t)}, y_2^{(t)}, \ldots, y_N^{(t)}\right)$ à vecto nhãn tại tuần thứ t+1.

3. Một số thuật toán thực hiện

- **Sử dụng yếu tố thời gian với ý tưởng ensemble method**

  - Sử dụng các thuật toán Logistic regression, Linear SVM của thư viện sklearn để training trên từng bộ $\left(X^{(t)}, y^{(t)}\right)$ → thu được T model dưới dạng (w_i, b_i)
  - Xây dựng ma trận trọng số model:

$$W = \left(w_1\ w_2\ \ldots\ w_T\right)$$
$$b = \left(b_1\ b_2\ \ldots\ b_T\right)$$

3. Một số thuật toán thực hiện

- **Sử dụng yếu tố thời gian với ý tưởng ensemble method**

Với bộ dữ liệu test: (X_{test}, y_{test}), sử dụng 2 cách để dự đoán nhãn:

- Tính ma trận score:

$$X_{test}W + b = (score)_{\#test \times T}$$

Mỗi phần tử (i, t) là score của của sample_i với model_t

Nhãn của sample_i = $sign(\sum_{t=1}^{T} e^{\alpha(t-T)} score(i,t))$

- Với mỗi sample_i sử dụng T model để test → thu được T label $a_k \in \{0, 1\}$.

Nhãn của sample_i = $sign\left(\frac{\sum_{k=1}^{T} k a_k}{1+2+\ldots+T} - \frac{1}{2}\right)$

3. Một số thuật toán thực hiện

- **Sử dụng yếu tố thời gian với ý tưởng ensemble method**

  - Ưu điểm: Thời gian thực hiện nhanh, có thể sử dụng theo thời gian thực, không phải re-train model

  - Nhược điểm: Trong một số trường hợp, có thể thiếu tính tổng quát, không hiệu quả nếu dữ liệu tương lai quá biến động…

4. Kết quả thực hiện

- Tiêu chí đánh giá có dạng:

Class	Precision	Recall	F1-score	Fbeta-score	Support
Inactive					
Active					
Avg/Total					

$$F1-score = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \qquad Fbeta-score = \frac{5 * \text{Precision} * \text{Recall}}{4 * \text{Precision} + \text{Recall}}$$

- Cần quan tâm 3 tiêu chí quan trọng:
  - Recall của inactive: tỷ lệ phát hiện đúng inactive = $\frac{\#inactive\ users\ \text{được dự đoán đúng}}{\#inactive\ users\ trên\ thực\ tế}$
  - Recall của avg/total: độ chính xác (Accuracy) = $\frac{\#users\ dự\ đoán\ chính\ xác}{tổng\ số\ users}$
  - Fbeta-score của inactive: ưu tiên recall hơn, mong muốn tìm được càng nhiều inactive users trên thực tế càng tốt

4. Kết quả

PHƯƠNG PHÁP	THUẬT TOÁN		TIÊU CHÍ ĐÁNH GIÁ			GHI CHÚ
			Recall inactive	Fbeta-score inactive	Recall avg	
Sử dụng lib sklearn	Linear SVM		0.79 **0.90**	0.66 **0.71**	0.62 0.5	Thời gian chạy rất lâu (2.5 ngày), phải re-train, phụ thuộc vào điểm khởi tạo
	Logistic Regression		0.84	0.69	0.63	- Thời gian chạy rất nhanh (40 phút), phải re-train, nên có thể hạn chế khi dữ liệu lớn theo thời gian - Perceptron phụ thuộc vào điểm khởi tạo
	AdaBoost	Perceptron	0.81	0.63	0.51	
		Logistic Regression	0.85	0.62	0.45	
	Kernel SVM		Chưa implement được do thời gian chạy quá lâu			
	Random Forest					
Sử dụng ý tưởng paper	Sử dụng hàm Cross-entropy	Sử dụng w0	0.76	0.59	0.51	Thuật toán chạy nhanh, học minibatch, phù hợp với dữ liệu lớn và capture được tính thời gian
		Bỏ w0	0.73	0.60	0.61	
	Sử dụng hàm Hinge loss và Square-hinge loss	Sử dụng w0	0.46	0.41	0.56	
		Bỏ w0	0.54	0.45	0.48	
Sử dụng ý tưởng Ensemble method	Linear SVM		0.52	0.51	**0.70**	Thời gian chạy khá nhanh (2h), không phải re-train, nhưng phụ thuộc điểm khởi tạo
	Gauss SVM		Kết quả chạy thấp			
	Logistic Regression		**0.84**	**0.70**	**0.63**	Thời gian chạy rất nhanh (40 phút), không phải re-train
Khác	Gauss SVM sử dụng Sequential Minimal Optimization		Chưa implement được			

5. Nhận xét

- Các thuật toán đã thực hiện đều là các bộ phân lớp tuyến tính với 2 đại diện là Linear SVM và Logistic Regression.

- Với data của bài toán, Linear SVM không hiệu quả bằng LR

- Do Linear SVM tối ưu theo các hàm hinge loss, square-hinge loss, và Logistic Regression tối ưu theo hàm cross entropy nên khi sử dụng ý tưởng của paper thì $l = hinge\ loss$ và $square - hinge\ loss$ đều kém hiệu quả hơn so với $l = cross\ entropy$

- Linear SVM và AdaBoost Perceptron cùng có một kiểu tối ưu theo hàm hinge loss và đều phụ thuộc vào điểm khởi tạo

6. Công việc sắp tới

- Sử dụng ý tưởng trong thuật toán AdaBoostClassifier để thiết kế bộ phân lớp không cần re-train, capture được tính thời gian
- Thử nghiệm Neural Network: RNN với LSTM