Online Outlier Detection for Time Series

Tingyi Zhu

September 14, 2016

Outline

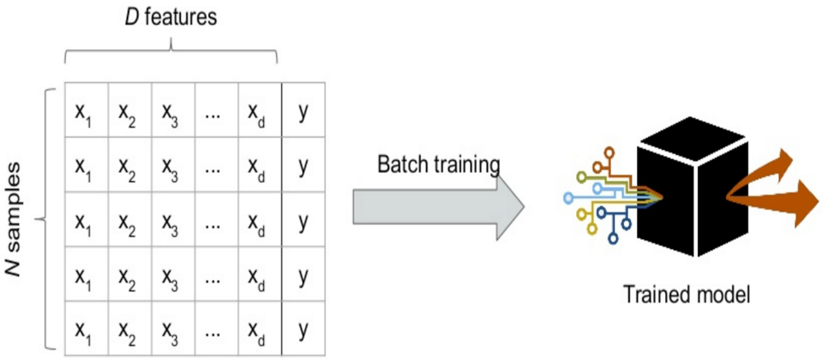
- Overview of Online Machine Learning
- Online Time Series Outlier Detection
- Demo

Overview of Online Machine Learning

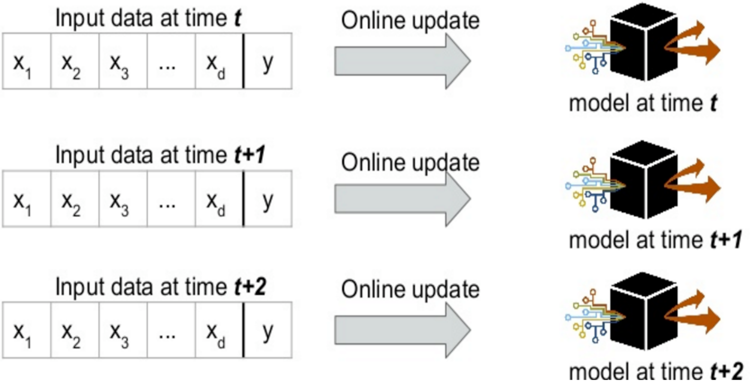
Introduction

- Online learning
  - a method of machine learning in which data is accessed in a sequential order
  - train the data in consecutive steps
  - update the predictor at each step
- Batch (offline) learning
  - learn on the entire training data set
  - generate the best predictor at once

Batch Learning



Online Learning



Two scenarios online algorithms are useful :

- When dealing with large dataset, computationally infeasible to train over the entire dataset
- When new data is constantly being generated, and prediction is needed instantly, cases include
  - In financial market, stock price prediction, portfolio selection
  - Real-time Recommendation
  - Online ad placement
  - Anomaly (outlier) detection

Statistical Learning Model

- X: space of inputs

  Y: space of outputs
- To learn the function $f\colon X \to Y$
- Loss function: $V\colon Y \times Y \to \mathbb{R}$
- $V(f(x), y)$ measures the difference between the predictive value $f(x)$ and the true value y
- Goal: Minimize the expected risk

$$\min_f E[V(f(x), y)]$$

- Given the training set

$$(x_1, y_1), \ldots, (x_n, y_n)$$

- Then minimize the empirical loss

$$\min_f \sum_{i=1}^{n} V(f(x_i), y_i)$$

Online Model

- Pure online learning model:
  - At time $t+1$, learning of f_{t+1} depends on the new input (x_{n+1}, y_{n+1}) and the previous best predictor f_t
  - Need extra stored information, usually independent of training data size
- Hybrid online learning model
  - Learning of f_{t+1} depends on f_t, and all previous data $(x_1, y_1), \ldots, (x_{t+1}, y_{t+1})$
  - The extra storing space requirement no longer constant, depends on training data size
  - Take less time to compute, compared to batch learning

Example: Linear Least Square

- $x_j \in \mathbb{R}^d$, $y_j \in \mathbb{R}$
- Learning a linear function: $f(x) = \langle w, x \rangle$

  Parameters to learn: $w \in \mathbb{R}^d$
- Square loss: $V(f(x), y) = (f(x) - y)^2$
- Minimize the empirical loss:

$$Q(w) = \sum_{j=1}^{n} Q_j(w) = \sum_{j=1}^{n} V(\langle w, x_j \rangle, y_j) = \sum_{j=1}^{n} (x_j^T w - y_j)^2$$

Batching Learning

- Let X be the $i \times d$ data matrix
- Y be the i matrix of target values after the arrival of the first i data points
- The covariance matrix $\Sigma_i = X^T X$
- The best solution of w is

$$\hat{w} = (X^T X)^{-1} X^T Y = \Sigma_i^{-1} \sum_{j=1}^{i} x_j y_j$$

- Recompute the solution after the arrival of every datapoint

Complexity of Batching Learning

After the arrival of the ith datapoint,

- Calculating Σ_i takes time $O(id^2)$
- Inverting the $d \times d$ matrix Σ_i^{-1} takes time $O(d^3)$
- Computation time at ith step: $O(id^2 + d^3)$

Total time with n total data points in the dataset:

$$\sum_i O(id^2 + d^3) = O(n^2 d^2 + nd^3)$$

Improving on Batch Learning

- When storing the matrix Σ_i
- Updating Σ at each step:

$$\Sigma_{i+1} = \Sigma_i + x_{i+1}x_{i+1}^T$$

  which only takes $O(d^2)$ time
- Total time reduces to $\sum_i O(d^2 + d^3) = O(nd^3)$
- Additional storage space of $O(d^2)$ needed to store Σ_i

Online Learning: Recursive Least Square

- Initializing: $w_0 = 0 \in \mathbb{R}^d$, $\Gamma_0 = I \in \mathbb{R}^{d\times d}$
- Solution of LS can be computed by the following iteration

$$\Gamma_i = \Gamma_{i-1} - \frac{\Gamma_{i-1} x_i x_i^T \Gamma_{i-1}}{1 + x_i^T \Gamma_{i-1} x_i},$$

$$w_i = w_{i-1} - \Gamma_i x_i (x_i^T w_{i-1} - y_i)$$

- w computed above is the same as the solution of batch learning, which can be proved by induction on i
- Complexity: $O(nd^2)$ for n steps

  require storage of matrix Γ, which is constant $O(d^2)$

- For the case when Σ_i is not invertible, consider the regularized version of loss function

$$Q(w) = \sum_{j=1}^{n} (x_j^T w - y_j)^2 + \lambda ||w||_2^2.$$

The same recursive algorithm works with the minor change on initialization:

$$\Gamma_0 = (I + \lambda I)^{-1}$$

Stochastic Gradient Descent

- Recall the standard gradient descent (also called batch gradient descent)], dealing with the problem of minimizing an objective function $Q(w)$
- The solution w is found through the iteration

$$w := w - \eta \nabla Q(w)$$

- When the objective has the form of a sum: $Q(w) = \sum_{j=1}^{n} Q_j(w)$

  the standard gradient descent performs the iteration:

$$w := w - \eta \sum_{j=1}^{n} \nabla Q_j(w)$$

- Each summand function Q_j is typically associated with the i-th observation in the data set
- When sample size n is large, standard GD requires expensive evaluations of the gradients from all summand functions, which is redundant because it computes gradients of very similar functions at each update
- SGD: update solution by computing the gradient of a single Q_j

$$w := w - \eta \nabla Q_j(w)$$

SGD for Online Learning

- Linear least square:

$$\min_w Q(w) = \sum_{j=1}^{n} Q_j(w) = \sum_{j=1}^{n} V(\langle w, x_j \rangle, y_j) = \sum_{j=1}^{n} (x_j^T w - y_j)^2$$

- After the arrival of the ith datapoint (x_i, y_i), update the solution by

$$w_i = w_{i-1} - \gamma_i \nabla V(\langle w, x_i \rangle, y_i) = w_{i-1} - \gamma_i x_i (x_i^T w_{i-1} - y_i)$$

which is similar to the updating procedure in recursive least square

$$w_i = w_{i-1} - \Gamma_i x_i (x_i^T w_{i-1} - y_i)$$

- Complexity of SGD algorithm for n steps reduces to $O(nd)$
- The storage requirement is constant at $O(d)$
- Consistency of SGD:

  When the learning rate γ decreases with an appropriate rate, SGD shows the same convergence behavior as batch gradient descent
- Averaging SGD: Choose $\gamma_i = 1/\sqrt{i}$, keep track of

$$\bar{w}_n = \frac{1}{n} \sum_{i=1}^{n} w_i$$

  which is consistent.

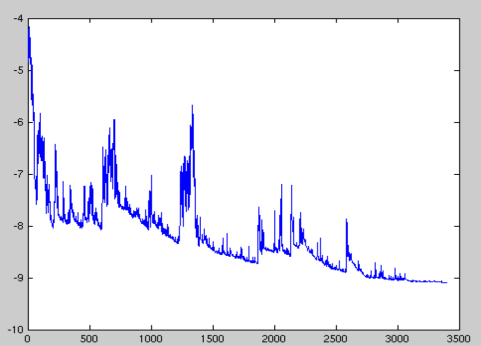


Figure: SGD fluctuation (Source: Wikipedia)

- SGD performs frequent updates, is faster
- With high variance, cause heavy fluctuation
- Complicates convergence to the exact minimum

Mini-Batch Gradient Descent

- Use b datapoints in each iteration
- Generalization and compromise between batch GD and SGD
- Get b datapoints: $(x_{i+1}, y_{i+1}), \ldots, (x_{i+b}, y_{i+b})$,

$$w := w - \gamma \cdot \frac{1}{b} \sum_{k=i+1}^{i+b} x_k (x_k^T w - y_k)$$

- b ranges from 2 to 100
- For online learning, b much smaller than n

Other topics of Online Learning

- Incremental (Online) PCA
- Incremental (Online) SVD
- Adversarial Model

etc.

Online Time Series Outlier Detection

The Problem

- Given: A time series $x_1, \dots, x_t$, new data point keep coming in as time went on
- Goal: Upon arrival of the new data, determine whether it's outlier
- Applications:
  - Financial market: abrupt change of stock price
  - Health data: blood pressure, heart beats
  - Intrusion detection: sudden increase of login attempts

The Framework

- Initial series $x_1, \ldots, x_n$ as training sample
- Use prediction algorithm, which can be adapted to online setting, to predict x_{n+1}
- Set criteria of identifying outliers or outlier events (subsequences), based on the prediction $\hat{x}_{n+1}$ and the new data point coming in x_{n+1}
- Update the training series to $x_1, \ldots, x_n, x_{n+1}$, and predict x_{n+2}

Chanllenge:

- Size of training sample grows as time went on, leading to more training time, fail to detect outliers on time

Solutions:

- Use slide window, drop the earliest data in the training sample while adding the new data, keep the size of training sample
- Use online learning techniques to update the model, avoiding retraining the whole sample.

Classical Online Time Series Prediction Model

1. Given a time series $\{x(t), \quad t = 1, 2, 3, \ldots\}$ and prediction origin O, construct a set of training samples, $\mathbf{A}_{O,B}$, from the segment of time series $\{x(t), \quad t = 1, \ldots, O\}$ as $\mathbf{A}_{O,B} = \{(\mathbf{X}(t), y(t)), t = B, \ldots, 0 - 1\}$, where $\mathbf{X}(t) = [x(t), \ldots, x(t - B + 1)]^T, y(t) = x(t + 1)$, and B is the embedding dimension of the training set $\mathbf{A}_{O,B}$.
2. Train a predictor $P(\mathbf{A}_{O,B}; \mathbf{X})$ from the training set $\mathbf{A}_{O,B}$.
3. Predict $x(O + 1)$ using $\hat{x}(O + 1) = P(\mathbf{A}_{O,B}; \mathbf{X}(O))$.
4. When $x(0 + 1)$ becomes available, update the prediction origin: $O = O + 1$. Then go to step 1 and repeat the procedure.

Training of $A_{O,B}$ —- A machine learning problem

- Linear model: methods in previous section
- Nonlinear model: Support vector regression, online SVR [Ma and Perkins, 2003]

Support Vector Regression

- Training set: $\{(x_i, y_i), i = 1, \ldots, l\}$, where $x_i \in \mathbb{R}^B$, $y_i \in \mathbb{R}$
- Construct the regression function

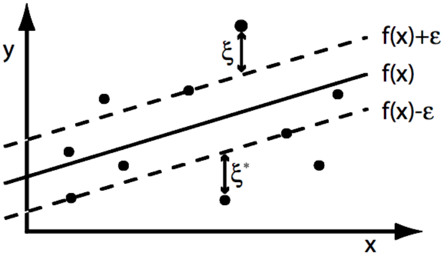
$$f(x) = W^T \Phi(x) + b$$

where $\Phi(x)$ maps the input x to vector in feature space

- W and b obtained by

$$\min_{\mathbf{W},b} \; P = \frac{1}{2}\mathbf{W}^T\mathbf{W} + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

$$\begin{aligned} s.t. \;\; & y_i - (\mathbf{W}^T\Phi(\mathbf{x}) + b) \leq \varepsilon + \xi_i \\ & (\mathbf{W}^T\Phi(\mathbf{x}) + b) - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \; i = 1 \cdots l. \end{aligned}$$



- Width of the band (between dashed lines) is $\frac{2\varepsilon}{||W||}$, so minimize $||W||^2$
- Also penalizes data points whose y-value differ from $f(x)$ by more than ε

- The dual optimization problem:

$$\min_{\alpha,\alpha^*} \; D = \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} Q_{ij}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) + \varepsilon \sum_{i=1}^{l}(\alpha_i + \alpha_i^*)$$

$$- \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*)$$

$$s.t. \;\; 0 \le \alpha_i, \alpha_i^* \le C \quad i = 1, \ldots, l,$$

$$\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0,$$

where $Q_{ij} = \Phi(x_i)^T\Phi(x_j) = K(x_i, x_j)$.

- $K(\cdot,\cdot)$ is the predetermined nonlinear kernel function and the solution of SVR can be written as

$$f(x) = \sum_{i=1}^{l}(\alpha - \alpha^*)K(x_i, x) + b$$

Online SVR: [Ma, Theiler and Perkins, 2003]

- The trained SVR function

$$f(x) = \sum_{i=1}^{l} (\alpha - \alpha^*) K(x_i, x) + b$$

- Update $(\alpha - \alpha^*)$ and b whenever a new data is added
- Faster than batch SVR algorithm

Outlier Detection Criteria

An example [Ma and Perkins, 2003] :

- Occurrence at t: $O(t) = \mathbb{1}\{|\hat{x}_t - x_t| \notin (-\varepsilon, \varepsilon)\}$
  - $\mathbb{1}\{\cdot\}$: indicator function
  - $\hat{x}$: prediction of x_t
  - ε: tolerance width
- A surprise is observed at t if $O(t) = 1$
- Event at time t: $E_n(t) = [O(t), O(t+1), \cdots, O(t+n-1)]^T$
  - n: event duration
- If $|E_n(t)| > h$, then $E_n(t)$ is defined as a novel event (outlier)
  - $|\cdot|$: 1-norm
  - h: a fixed lover bound

Demo