VỀ PHƯƠNG PHÁP GRADIENT DESCENT GIẢI BÀI TOÁN TỐI ƯU TRONG HỌC MÁY

Bùi Thị Thanh Xuân*, Dương Thị Nhung, Hà Thị Thanh
Trường Đại học Công nghệ thông tin và Truyền thông – ĐH Thái Nguyên

TÓM TẮT

Trong học máy, khi đưa về mô hình toán học, rất nhiều bài toán được đưa về dưới dạng một bài toán tối ưu phức tạp. Do đó đòi hỏi phải có phương pháp hiệu quả để giải chúng. Lý thuyết tối ưu đã đưa ra rất nhiều phương pháp giải, tuy nhiên với mỗi phương pháp đòi hỏi quá nhiều ràng buộc và điều kiện, hơn nữa chưa tính đến các thuật toán phải đảm bảo hiệu quả đối với dữ liệu lớn. Phương pháp gradient descent là một phương pháp đủ đơn giản và hiệu quả để giải một số bài toán tối ưu xuất hiện trong học máy. Khắc phục hạn chế của phương pháp gradient descent cổ điển, phương pháp Stochastic Gradient Descent chỉ cần tới đạo hàm bậc nhất kết hợp với việc lấy mẫu ngẫu nhiên đã làm tăng hiệu quả và độ tin cậy của thuật toán so với các cách tiếp cận trước khi phải đối mặt với dữ liệu lớn.

Từ khóa: *học máy, phương pháp giảm gradient ngẫu nhiên, phương pháp độ dốc lớn nhất, phương pháp giảm gradient, tối ưu, tối ưu phi tuyến*

TỔNG QUAN

Trong học máy (Machine Learning – ML) đề cập đến rất nhiều mô hình máy học với các thuật toán huấn luyện được phát biểu hình thức dưới dạng một bài toán tối ưu toán học, ví dụ như các mô hình linear/logistic regression, ANN, SVM, K-means... Ý tưởng chung của các bài toán này là mỗi mô hình đều có một tập tham số θ nào đó cần phải được xác định giá trị thông qua quá trình huấn luyện với dữ liệu. Trước đây khi ngành học máy chưa phát triển, trong lĩnh vực nghiên cứu thống kê gọi quá trình này là ước lượng tham số (Parameter Estimation) trong đó giá trị của các tham số được chọn sao cho bài toán tối ưu theo một tiêu chuẩn nào đó do người xây dựng mô hình quyết định. Các tiêu chuẩn thường được dùng là phân phối xác suất likelihood, posterior hay là Mean Square Error...tương ứng với các phương pháp là Maximum - Likelihood Estimation (MLE), MAP/EAP và Minimum Mean Square Error (MMSE). Ý tưởng để giải các bài toán tối ưu ở đây là ta cần chọn ra một tiêu chuẩn, rồi chọn giá trị của tham số sao cho tối ưu tiêu chuẩn đó. Tối ưu là ngành cung cấp cơ sở lí thuyết cho bài toán này một cách vững vàng nhất, do đó áp dụng tối ưu để học các mô hình thống kê trở thành tư tưởng hết sức tự nhiên ([2],[3]).

Hiện nay trong học máy ngoài việc nghiên cứu các mô hình tham số (parametric model) còn nghiên cứu các mô hình phi tham số (nonparametric model). Tuy nhiên các mô hình tham số vẫn đóng vai trò hết sức quan trọng, nên tìm hiểu nguyên lí chung của các mô hình này là hữu ích. Bài báo này trình bày về các phương pháp ước lượng tham số sử dụng kĩ thuật tối ưu dựa trên kỹ thuật giảm gradient. Cụ thể trong bài báo này sẽ trình bày về các phương pháp Gradient Descent, Minibatch Gradient Descent, Stochastic Gradient Descent cùng kĩ thuật hiệu chỉnh ([2],[3]).

Một cách tổng quát, thông thường muốn cực tiểu một hàm chi phí (cost function) $C(\theta)$ ánh xạ vector tham số θ thành một giá trị vô hướng. Hàm này được tính trên toàn tập huấn luyện. Trong học máy, hàm chi phí (cost function) thường được biểu diễn thành trung bình cộng hoặc kì vọng của một hàm lỗi (loss function) nào đó áp dụng cho mỗi mẫu trong tập huấn luyện:

$$C(\theta) = \frac{1}{n}\sum_{i=1}^{n} L\left(f_0, z_i\right) \qquad (1)$$

(gọi là training loss) hoặc

* *Tel. 0902 001581, Email thanhxuan1581@gmail.com*

$$C(\theta) = \int L(f_0, z) P(z) dz \quad (2)$$

(gọi là generalization loss)

trong đó θ là véc tơ tham số của mô hình, z_i là một mẫu (sample) trong tập huấn luyện và f_θ là đầu ra (output) của mô hình, ứng với tham số θ. Ý tưởng chung là muốn tìm tham số θ sao cho cực tiểu hàm chi phí này.

Trong ngữ cảnh học có giám sát (supervised learning) $z_i = (x_i, y_i)$ với x_i là sample đầu vào và y_i là nhãn tương ứng, còn $f_0(x_i)$ là kết quả dự đoán y_i do mô hình đưa ra. Cụ thể trong một bài toán phân lớp (classification), muốn xây dựng hàm dự đoán

$f : \square^n \to \{0,1,..,L-1\}$

với giả sử có L lớp trong bài toán đang xét.

Một trong các hàm cost function có thể dùng là hàm zero-one loss:

$$C(\theta) \equiv l_{0,1} = \sum_{i=1}^{|D|} I\left(f_0(x_i) \neq y_i\right)$$

trong đó $D \equiv D_{tram}$ là tập huấn luyện hoặc $D \cap D_{tram} = \varnothing$ để đảm bảo tính công bằng khi đánh giá thuật toán trên tập validation hoặc tập test và $I(A)$ là hàm indicating function: $I(A) = 1$ nếu và chỉ nếu A đúng. Tuy nhiên một khó khăn của hàm zero-one loss là không khả vi nên không thể áp dụng các thuật toán tối ưu. Khi đó ta sẽ cực đại hàm log-likelihood trên toàn tập huấn luyện:

$$L(\theta, D) = \log \prod_{i=1}^{|D|} P\left(Y = y_i \middle| x_i, \theta\right)$$

$$= \prod_{i=1}^{|D|} \log P\left(Y = y_i \middle| x_i, \theta\right)$$

Hàm zero-one loss và log-likelihood là hai tiêu chuẩn khác nhau, cũng như ta có thể định nghĩa các tiêu chuẩn khác như xác suất hậu nghiệm (posterior) hoặc Mean Squared Error... Thông thường zero-one loss và log-likelihood là tương quan (correlated) trên tập kiểm tra (validation), tuy nhiên điều này không phải lúc nào cũng đúng. Do chúng ta đang muốn cực tiểu hóa một hàm cost function, nên ta sẽ cực tiểu hàm negative log-likelihood (NLL):

$$C(\theta) \equiv NLL(\theta, D)$$

$$= -\prod_{i=1}^{|D|} \log P\left(Y = y_i \middle| x_i, \theta\right) \quad (3)$$

Yêu cầu đặt ra là tìm θ để cực tiểu hóa hàm giá trị $C(\theta)$. Mặc dù trong lí thuyết tối ưu có rất nhiều phương pháp khác nhau để giải bài toán này, tuy nhiên trong học máy đòi hỏi các phương pháp phải hiệu quả khi làm việc với dữ liệu lớn, phức tạp thì các phương pháp giảm gradient tỏ ra hiệu quả, đặc biệt là Stochastic GD.

PHƯƠNG PHÁP ĐỘ DỐC LỚN NHẤT - GRADIENT DESCENT (GD)

Theo lý thuyết tối ưu, để tìm cực trị của $C(\theta)$ ta có thể giải phương trình:

$$\frac{\partial C(\theta)}{\partial \theta} = 0$$

Nghiệm của phương trình cho giá trị tối ưu của θ. Cách này chỉ làm được khi ta có thể tính toán chính xác đạo hàm bậc nhất của $C(\theta)$ ([1],[4],[5]).

Trong học máy, bằng cách sử dụng hàm likelihood, lấy đạo hàm để có công thức tường minh của các tham số rồi sử dụng thuật toán EM (Expectation Maximization) là cách làm tỏ ra thành công với các mô hình như K-means, GMM (phương pháp moment tổng quát- Generalized Least Square) hay HMM (mô hình Markov ẩn - Hidden Markov Model). Tuy nhiên việc lấy đạo hàm và giải phương trình trên không phải lúc nào cũng thực hiện được, do đó cần sử dụng các phương pháp tối ưu.

Một phương pháp đủ đơn giản mà trong học máy có sử dụng chính là phương pháp gradient descent (trong tối ưu còn gọi là Steepest descent – phương pháp độ dốc lớn nhất). Cách đơn giản nhất để hiểu gradient descent là từ vị trí hiện tại, ta đi theo chiều

giảm của đạo hàm bậc nhất cho đến khi không thể giảm được nữa. Khi đó ta đã ở một điểm tối ưu cục bộ ([2][4][5]). Công thức cập nhật cho gradient descent là:

$$\theta_{k+1} = \theta_k - \alpha_k \frac{\partial}{\partial \theta} C(\theta_k) \quad (4)$$

trong đó θ_k là tham số tại lần lặp thứ k và α_k gọi là tốc độ học (learning rate), có thể được cố định hoặc thay đổi thích nghi trong suốt quá trình huấn luyện.

```
% Gradient Descent
 while True:
loss = f(theta)
d_loss_theta=...%tính gradient
theta- = learning_rate* d_loss_theta
if <stopping condition is met>:
return theta
```

PHƯƠNG PHÁP MINIBATCH GRADIENT DESCENT (MGD)

Khắc phục hạn chế của phương pháp gradient descent là dùng toàn bộ tập huấn luyện cho mỗi bước, ta có phương pháp Minibatch gradient descent, trong đó sử dụng B sample cho mỗi bước cập nhật của θ ([2], [3]).

```
% Minibatch Gradient Descent
For (x_batch,y_batch) in train_batches·
    loss = f(theta, x_batch, y_batch)
    d_loss_theta= ... % tính gradient
    theta - = learning_rate* d_loss_theta
    if <stopping condition is met>:
        return theta
```

Phương pháp Minibatch có lợi thế là có thể giúp thuật toán hội tụ nhanh hơn vì thay vì tính B phép nhân vector và ma trận, ta chỉ cần tính một phép nhân ma trận với ma trận, trong đó ma trận thứ nhất có B hàng. Phép nhân này có thể được cài đặt rất hiệu quả trong Matlab hay các thư viện tính toán khác. Tuy nhiên nếu B lớn thì thuật toán sẽ tốn chi phí cho việc tính hướng giảm tại mỗi bước. Do đó B thực sự là tham số trade-off về mặt hiệu quả tính toán. Giá trị tối ưu cho B không chỉ phụ thuộc vào mô hình mà còn phụ thuộc vào dataset và kiến trúc máy tính. Nếu điều kiện dừng của thuật toán là số lần lặp (epoch) thì B trở nên vô cùng quan trọng vì nó quyết định số lần cập nhật của tham số. Huấn luyện mô hình với số epoch = 10 và B = 1 có thể cho kết quả hoàn toàn khác với epoch = 10 và B = 20. Do đó khi sử dụng minibatch gradient descent nên thận trọng khi thay đổi B, vì có thể ta sẽ phải tune lại toàn bộ các tham số khác với mỗi giá trị mới của B.

PHƯƠNG PHÁP MOMENTUM

Tương tự như Minibatch GD nhưng ý tưởng của phương pháp momentum là tính toán mức độ thay đổi của tham số tại mỗi bước dựa vào bước trước đó. Như vậy tại mỗi bước tham số sẽ thay đổi một cách "thích nghi" dần dần với các lần lặp trước. Cụ thể:

$$\Delta\theta_{k+1} = \varepsilon\Delta\theta_k + (1-\varepsilon)\frac{\partial}{\partial\theta} L(\theta_k, z) \quad (5)$$

với ε là hyper-parameter điều khiển mức độ ảnh hưởng của gradient vào mức giảm tại mỗi bước.

PHƯƠNG PHÁP STOCHASTIC GRADIENT DESCENT (SGD)

Nhận thấy hàm chi phí C(θ) là trung bình cộng, và thông thường tập huấn luyện là độc lập và có cùng phân phối (i.i.d – independently and identically distributed) nên tại mỗi bước ta có thể cập nhật tham số với mỗi sample trong tập huấn luyện:

$$\theta_{k+1} = \theta_k - \alpha_k \frac{\partial}{\partial\theta} L(\theta_k, z) \quad (6)$$

với z là sample tiếp theo trong tập huấn luyện, hoặc trong các ngữ cảnh online khi dữ liệu huấn luyện được đưa đến từng mẫu một (có thể vô hạn) và ta không có trọn vẹn cả tập huấn luyện ngay từ đầu. Một cách để hiểu về SGD là hướng cập nhật cho tham số là một biến ngẫu nhiên mà kì vọng của nó là hướng cập nhật tính bởi Gradient descent.

SGD thông thường nhanh hơn gradient descent vì ta cập nhật các tham số nhiều hơn hẳn ([3]). Điều này đặc biệt đúng khi có tập huấn luyện lớn hoặc không có toàn bộ tập

huấn luyện ngay từ đầu. Trong học máy chỉ dùng GD khi hàm chi phí (cost function) không thể viết dưới dạng trung bình như trên.

```
% Stochastic Gradient Descent
for (x_i,y_i) in training_set:
    loss = f(theta, x_i, y_i)
    d_loss_theta =...% tính gradient
    theta - =learning_rate* d_loss_theta
            if <stopping condition is met>:
                return theta
```

CÁCH CHỌN LEARNING RATE

Nếu learning rate quá lớn, chẳng hạn lớn gấp đôi trị riêng lớn nhất của ma trận Hessian của $C(\theta)$ thì các bước sẽ đi lên (ascent) thay vì đi xuống (descent). Ngược lại nếu learning rate quá nhỏ thì thuật toán sẽ hội tụ chậm. Lý thuyết tối ưu đã giải quyết vấn đề này bằng thuật toán line search ([4], [5]), tuy nhiên trong học máy thì thường chọn α^k theo các cách sau:

+ Hằng số $\alpha_k = \alpha_0$: đây là lựa chọn phổ biến nhất và phù hợp khi phân phối của $C(\theta)$ không cố định. Đây là cách làm robust nhưng giá trị của $C(\theta)$ có thể không giảm nữa sau một số lần lặp, trong khi nếu dùng learning rate nhỏ hơn thì có thể giảm tiếp (tiến gần hơn 1 chút đến điểm tối ưu).

+ Phụ thuộc vào số bước lặp k: $\alpha_k = \alpha_0 \frac{\tau}{\tau + k}$. Chiến lược này đảm bảo đến được điểm tối ưu khi $k \to \infty$ vì nó thỏa $\sum_{k=1}^{\infty} \alpha_k = \infty$ và $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ ([3]). Điều này đúng với mọi τ nhưng α_0 phải đủ nhỏ để tránh đi lên thay vì đi xuống. Tuy nhiên cần phải cân nhắc vì khi đó lại có thêm một hyper-parameter cần phải tune. Chọn giá trị không tốt cho τ có thể làm chậm quá trình hội tụ của thuật toán.

HIỆU CHỈNH

Đây là vấn đề của học máy không phải của lý thuyết tối ưu. Khi tối ưu $C(\theta)$ chúng ta không muốn "tối ưu quá", vì như thế mô hình sẽ bị quá khớp (overfit) vào dữ liệu huấn luyện và sẽ không đủ tổng quát để mô hình hóa cả dữ liệu mới. Hiệu chỉnh (regularization) là cách để cân bằng hiện tượng này ([2],[3]). Có nhiều cách để hiệu chỉnh mô hình học máy, ở đây chúng ta sẽ xem xét đến L1/L2 regularization:

Thêm vào sau $C(\theta)$ một hàm của θ để tránh hiện tượng overfitting. Cụ thể hàm regularized loss function sẽ có dạng:

$$E(\theta, D) = C(\theta) + \lambda R(\theta)$$

với $R(\theta)$ là hàm regularization. Trong trường hợp L1/L2 regularization thì:

$$E(\theta, D) = C(\theta) + \lambda \|\theta\|_p^p$$

với $\|\theta\| p = \left(\sum_{j=1}^{|\theta|} |\theta_j|^p \right)^{\frac{1}{p}}$ là chuẩn L_p của vector θ, p có thể bằng 1 hoặc 2, do đó gọi tên là L1/L2 regularization. λ là hyper - parameter thể hiện mức độ trade-off giữa $C(\theta)$ và $R(\theta)$. Tức là, có $C(\theta)$ thể hiện mức độ phù hợp (khớp- fit) của mô hình vào dữ liệu (nhất là khi sử dụng xác suất likelihood) còn $R(\theta)$ thể hiện mức độ regularization mà ta muốn thực hiện. Tối ưu đồng thời cả 2 yếu tố này, ta hi vọng đạt được tình huống nào đó cân bằng cả hai. Mức độ trade-off giữa hai bên được thể hiện bằng λ.

Trong huấn luyện ANN hay SVM, L1/L2 regularization giúp cân bằng các trọng số, tránh hiện tượng chênh lệch giá trị giữa các tham số và làm cho mô hình "đơn giản" hơn, kết quả tối ưu sẽ giúp chúng ta có kết quả đơn giản nhất có thể (theo tiêu chí đã chọn) mà vẫn phù hợp được dữ liệu. Tất nhiên decision boundary "mượt" không nhất thiết đồng nghĩa với khả năng tổng quát hóa tốt hơn.

KẾT LUẬN

Bài báo này trình bày về các chiến lược gradient descent thường dùng trong các bài toán tối ưu trong học máy, đặc biệt là phương

pháp SGD với cách chọn tốc độ học (learning rate) và kĩ thuật hiệu chỉnh (regularization). Đây là một hướng tiếp cận đủ đơn giản nhưng không giảm tính hiệu quả khi đối mặt với dữ liệu lớn - khó khăn của các bài toán học máy và khai phá dữ liệu hiện nay đang giải quyết.

TÀI LIỆU THAM KHẢO

1. S. Boyd and L.Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
2. Mark Schmidt, *Convex Optimization for Big Data*, Asian Conference on Machine Learning, November 2014.
3. Leon Bottou, *Online Algorithms and Stochastic Approximations*, Online Learning and Neural Networks, Cambridge University Press, 1998.
4. Nguyễn Thị Bạch Kim, *Các phương pháp tối ưu: Lý thuyết và thuật toán*, Nxb Bách Khoa Hà Nội, 2014.
5. Bùi Minh Trí, *Quy hoạch toán học*, Nxb Khoa học kỹ thuật, 2001.

SUMMARY

GRADIENT DESCENT METHOD SOLVING OPTIMIZATION PROBLEM IN MACHINE LEARNING

Bui Thi Thanh Xuan*, Duong Thi Nhung, Ha Thi Thanh
College of Information & Communication Technology – TNU

The gradient descent method was originally consider to be direct method for linear equations, but its favorable properties as an iterative method was soon realized, and it was later generalized to more general optimization problems. It provides a very effective way to optimize large, deterministic systems by gradient descent. Stochastic gradient descent (SGD) is a gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions. The convergence of stochastic gradient descent has been analyzed using the theories of convex minimization and of stochastic approximation. Briefly, when the learning rates decrease with an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to a global minimum when the objective function is convex or pseudoconvex, and otherwise converges almost surely to a local minimum. Stochastic gradient descent is a popular algorithm for training a wide range of models in machine learning, including (linear) support vector machines, logistic regression and graphical models.

Keywords: *machine learning, stochastic gradient descent, gradient descent, minibatch gradient descent, nonlinear optimization, nonconvex optimization*

Ngày nhận bài:03/10/2015; Ngày phản biện:23/11/2015; Ngày duyệt đăng: 15/3/2016
Phản biện khoa học: *TS. Trương Hà Hải – Trường Đại học Công nghệ Thông tin & Truyền thông - ĐHTN*

* *Tel: 0902 001581, Email thanhxuan1581@gmail.com*