**Cost efficient gradient boosting**

1. **Introduction**

Nhiều ứng dụng cần Classifier hoặc Regression, nó không chỉ chính xác mà còn rẻ để đánh giá. Chi phí dự đoán thường bao gồm hai thành phần khác nhau: Việc mua lại hoặc tính toán của các tính năng được sử dụng để dự đoán đầu ra và của chính predictor. Một cách tiếp cận phổ biến để xây dựng một dự đoán chính xác với chi phí đánh giá thấp là sửa đổi mục tiêu giảm thiểu rủi ro

Trong công việc này, chúng tôi cũng làm theo cách tiếp cận chung này, và phát triển một chiến lược nhận thức ngân sách dựa trên các cây hồi quy tăng cường sâu. Mặc dù sự tái xuất hiện gần đây và sự phổ biến của các mạng thần kinh, sự lựa chọn của chúng ta về cây hồi quy được thúc đẩy được thúc đẩy bởi ba quan sát:

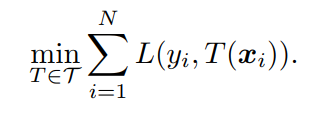
+ Với nhiều dữ liệu đào tạo và tài nguyên tính toán, mạng nơron sâu thường cho kết quả chính xác nhất. Tuy nhiên, các kiến trúc chuyển tiếp tiêu chuẩn chuyển tiếp một thành phần đầu vào đơn (ví dụ, một hệ số đơn lẻ trong trường hợp đầu vào vectơ) thông qua hầu hết các đơn vị mạng. Trong khi chi phí tính toán có thể được giảm thiểu bằng cách nén mạng hoặc lượng tử hóa, trong trường hợp cực đoan chỉ kích hoạt nhị phân, thì đồ thị tính toán về cơ bản là dày đặc. Trong một cây quyết định tiêu chuẩn, mặt khác, mỗi mẫu được định tuyến dọc theo một con đường duy nhất từ gốc đến một lá. Do đó, Hội nghị lần thứ 31 về các hệ thống xử lý thông tin thần kinh (NIPS 2017), Long Beach, CA, USA.truy cập thường chỉ là một tập con nhỏ của tất cả các nút tách, "đơn vị" của cây quyết định. Trong trường hợp cực đoan của cây nhị phân cân bằng, mỗi lượt truy cập mẫu chỉ ghi nhật ký (N) trong tổng số N nút.

+ Decision Tree và Ensemble , Random Forest và Gradient Boosting vẫn là một trong những phương pháp hữu ích và có tính cạnh tranh cao nhất trong học máy, đặc biệt là trong chế độ dữ liệu huấn luyện giới hạn, ít thời gian huấn luyện và ít chuyên môn để điều chỉnh thông số

+ Khi các tính năng và / hoặc các quyết định có mức phí cao, thuận tiện nhưng lãng phí khi giả định rằng tất cả các cá thể trong tập hợp dữ liệu được tạo bằng nhau (ngay cả khi được giả định là i.i.d.). Một số trường hợp có thể dễ dàng phân loại dựa trên việc đọc một phép đo / tính năng duy nhất, trong khi những trường hợp khác có thể yêu cầu kiểm tra đầy đủ pin trước khi có thể đạt được quyết định. Decision Tree cho mình thiết lập "thiết kế thử nghiệm tuần tự": sau khi sử dụng các tính năng giá rẻ để tách tất cả các cá thể thành các tập con, các quyết định tiếp theo có thể dựa trên các tính năng đắt tiền hơn, tuy nhiên, chỉ được gợi ra nếu thật sự cần thiết. Quan trọng hơn, các tính năng đắt tiền hơn được yêu cầu có điều kiện trên các giá trị của các tính năng được sử dụng trước đó trong cây

Trong công việc này, chúng tôi giải quyết vấn đề xây dựng một quần thể cây vừa chính xác vừa rẻ để đánh giá. Đầu tiên chúng tôi mô tả vấn đề thiết lập trong Phần 2, và thảo luận về công việc liên quan trong Phần 3. Đóng góp quan trọng của chúng tôi sẽ xuất hiện trong Phần 4, nơi chúng tôi đề xuất mở rộng gradient.Trái ngược với cách tiếp cận trước đây để học tập với hình phạt chi phí, phương pháp của chúng tôi có thể phát triển cây rất sâu mà trung bình vẫn còn rẻ để tính toán. Thuật toán của chúng tôi dễ thực hiện và thời gian học của nó có thể so sánh với thuật toán của việc tăng cường độ dốc ban đầu. Như minh họa trong Phần 5, trên một số bộ dữ liệu, phương pháp của chúng tôi hoạt động tốt hơn trạng thái hiện tại bằng một lề lớn

1. **Problem setup**

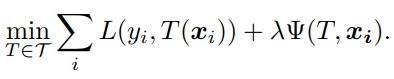
Xem xét một vấn đề hồi quy trong đó đáp ứng và mỗi trường hợp X được biểu diễn bằng các tính năng của M, , cho L là một hàm mất, và T là một tập hợp các hàm được chấp nhận. Trong học có giám sát, được tập huấn về các cặp N (xi; yi) lấy mẫu i.i.d. từ (X; Y), một cách tiếp cận cổ điển để tìm hiểu người dự đoán T 2 T là giảm thiểu tổn thất thực nghiệm L trên tập huấn luyện,

Trong bài báo này, chúng tôi hạn chế bản thân với bộ T bao gồm một tập hợp các cây, cụ thể là các yếu tố dự báo của biểu mẫu

Mỗi cây quyết định có thể được biểu diễn như một tập hợp các nút lá với các đáp ứng tương ứng và hàm

mã hóa cấu trúc cây và ánh xạ đầu vào tới chỉ mục lá cuối tương ứng của nó. Đầu ra của cây là

Học ngay cả một cây duy nhất mà giảm thiểu chính xác chức năng trong phương trình là NP-hard theo một số khía cạnh của sự tối ưu. Tuy nhiên, cây duy nhất và quần thể cây là một số yếu tố dự đoán thành công nhất trong học máy và có nhiều phương pháp dựa trên tham lam để xây dựng các cụm cây mà xấp xỉ giải phương trình.

Tuy nhiên, trong nhiều ứng dụng thực tế, điều quan trọng là người dự đoán T không chỉ chính xác mà còn nhanh chóng tính toán. Đưa ra một hàm chi phí dự đoán một cách tiếp cận tiêu chuẩn là thêm một hình phạt để giảm thiểu rủi ro thực nghiệm ở trên

Tham số λ kiểm soát sự cân bằng giữa độ chính xác và chi phí dự đoán.

Thông thường, hàm chi phí dự đoán Ψ bao gồm hai thành phần. Đầu tiên là chi phí mua hoặc tính toán các tính năng đầu vào có liên quan. Ví dụ, suy nghĩ của một bệnh nhân tại phòng cấp cứu, nơi lấy nhiệt độ và mức độ oxy trong máu của bạn là giá rẻ, nhưng CT-scan là tốn kém. Thành phần thứ hai là chi phí đánh giá hàm T, mà trong trường hợp của chúng ta là tổng chi phí đánh giá các cây K riêng lẻ

Cụ thể hơn, thành phần đầu tiên của chi phí tính toán cũng có thể phụ thuộc vào vấn đề dự đoán cụ thể. Trong một số trường hợp, các cá thể kiểm thử độc lập với nhau và các tính năng có thể được tính toán cho mỗi cá thể đầu vào theo yêu cầu. Nhưng cũng có những người khác. Ví dụ, trong xử lý hình ảnh, nơi đầu vào là một hình ảnh chứa nhiều pixel và nhiệm vụ là dự đoán một số chức năng ở tất cả các pixel. Trong những trường hợp như vậy, mặc dù các tính năng cụ thể có thể được tính toán cho từng pixel độc lập, nhưng có thể rẻ hơn hoặc hiệu quả hơn để tính toán cùng một tính năng, chẳng hạn như bộ lọc chập tách, tại tất cả các pixel cùng một lúc [1, 13]. Hàm chi phí Ψ có thể bị chi phối trong các trường hợp này bởi thành phần thứ hai - thời gian cần để đánh giá cây

Sau khi thảo luận về công việc liên quan trong Phần 3, trong Phần 4 chúng tôi trình bày một sự thích nghi chung của việc tăng cường gradient để giảm thiểu tối đa phương trình, có tính đến cả hai thành phần chi phí dự đoán.

1. **Related work**

Vấn đề học tập với hình phạt dự đoán chi phí đã được nghiên cứu rộng rãi. Một trường hợp cụ thể là sự mất cân bằng lớp học, trong đó một lớp cực kỳ hiếm và điều quan trọng là phải chú thích chính xác nó. Ví dụ, thác Viola-Jones nổi tiếng sử dụng các tính năng giá rẻ để loại bỏ các ví dụ thuộc về lớp phủ định. Các giai đoạn sau đòi hỏi các tính năng đắt tiền chỉ được sử dụng cho lớp tích cực nghi ngờ hiếm hoi. Trong khi một cách tiếp cận như vậy là rất thành công, do chiến lược xuất cảnh sớm của nó, nó không thể sử dụng các tính năng đắt tiền cho các đầu vào khác nhau

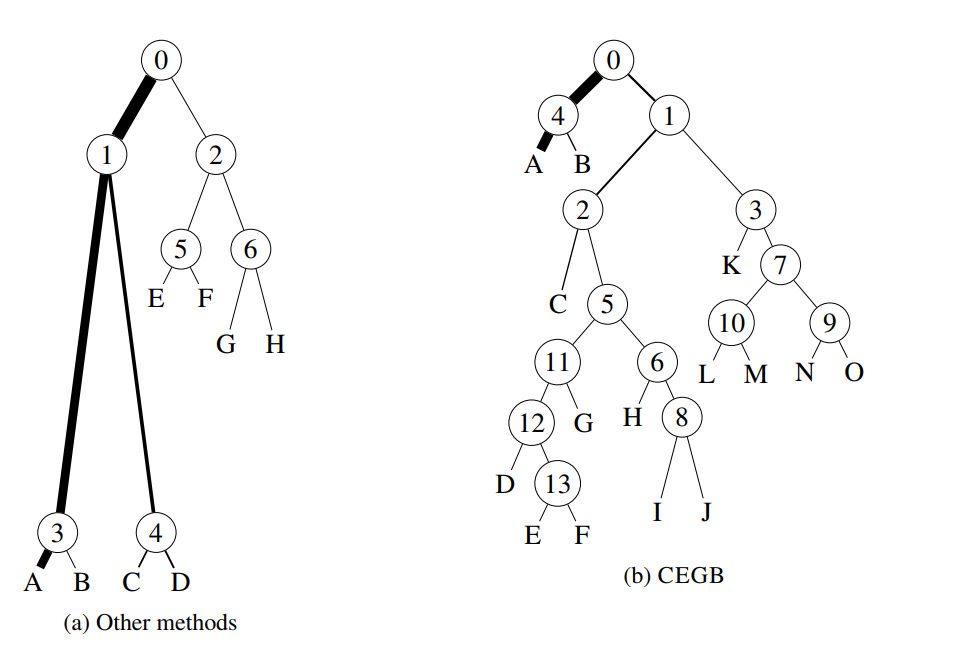
Để khắc phục các hạn chế được áp dụng bởi các chiến lược thoát hiểm sớm, các phương pháp khác nhau đã đề xuất các công trình cây đơn lẻ nhưng với các quyết định phức tạp hơn ở các nút tách riêng lẻ. Cây được học đầu tiên mà không tính đến chi phí dự đoán, sau đó là bước tối ưu hóa bao gồm chi phí này. Thật không may, trong thực tế, các phương pháp đơn cây này thấp hơn các thuật toán hiện đại xây dựng các cụm cây

BUDGETRF dựa trên Random Forest và sửa đổi chức năng tạp chất quyết định sự phân chia nào cần thực hiện, để tính đến chi phí tính năng. BUDGETRF có một số hạn chế: Thứ nhất, giả định rằng chi phí đánh giá cây là không đáng kể so với tính năng mua lại, và do đó không phù hợp với các vấn đề mà tính năng rẻ để tính toán và chi phí dự đoán bị chi phối bởi đánh giá dự đoán hoặc cả hai thành phần đóng góp như nhau. Thứ hai, trong giai đoạn đào tạo của mình, mỗi lần sử dụng tính năng phát sinh chi phí mua lại của nó để sử dụng tính năng lặp lại không được mô hình hóa và xác suất để đạt đến nút không được tính đến. Tại thời điểm thử nghiệm, ngược lại, họ cho phép "miễn phí" tái sử dụng các tính năng đắt tiền và tính toán chi phí chính xác của việc tiếp cận các nhánh cây khác nhau. BUDGETRF do đó thường không mang lại các chi nhánh sâu nhưng đắt tiền mà chỉ hiếm khi đạt được.

BUDGETPRUNE là một lược đồ tỉa cành cho cây quyết định. Nó nhằm mục đích giảm thiểu sự bắt chước của BUDGETRF bằng cách cắt tỉa các chi nhánh đắt tiền từ các cây riêng lẻ. Một chương trình tuyến tính tích phân được xây dựng và giải quyết hiệu quả để sử dụng tính năng lặp lại và xác suất để đạt được các chi nhánh khác nhau. Phương pháp này dẫn đến một sự cân bằng tốt hơn nhưng vẫn không thể tạo ra các chi nhánh sâu và đắt tiền mà chỉ hiếm khi đạt được nếu chúng không có trong bộ gốc. Phương pháp này được coi là trạng thái của nghệ thuật khi chi phí dự đoán bị chi phối bởi chi phí mua lại tính năng. Chúng tôi trình bày trong Phần 5 rằng việc xây dựng các cây sâu hơn với các phương pháp của chúng tôi sẽ mang lại hiệu suất tốt hơn đáng kể.

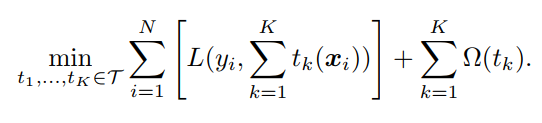
GREEDYMISER, tương tự nhất với công việc của chúng tôi, là một thuật toán kiểu tăng cường gradient theo giai đoạn nhằm mục đích giảm thiểu tối đa phương trình. sử dụng một tập hợp các cây hồi quy. Khi cả hai thành phần chi phí dự đoán được giả định là không kém, GREEDYMISER được coi là trạng thái của nghệ thuật. Tuy nhiên, GREEDYMISER cũng có một số hạn chế: Thứ nhất, tất cả các cây đều được giả định có cùng chi phí dự đoán cho tất cả các yếu tố đầu vào. Thứ hai, bằng cách thiết kế nó xây dựng các cây nông tất cả đều có độ sâu tương tự. Thay vào đó, chúng tôi xem xét chi phí cá nhân cho mỗi chiếc lá và do đó cho phép xây dựng các cây sâu hơn. Các thí nghiệm của chúng tôi trong Phần 5 gợi ý rằng việc xây dựng các cây sâu hơn với phương pháp đề xuất của chúng tôi tốt hơn đáng kể so với GREEDYMISER.

1. **Gradient boosting with cost penalties**

Chúng tôi xây dựng trên khuôn khổ tăng cường gradient và điều chỉnh nó để cho phép tối ưu hóa với các hình phạt chi phí. Trước tiên, chúng tôi xem xét ngắn gọn thuật toán gốc. Sau đó chúng tôi trình bày chi phí của chúng tôi trong Mục 4.1, bước tối ưu hóa trong 4.2 và cuối cùng là thuật toán trồng cây của chúng tôi xây dựng cây với các nhánh sâu nhưng độ sâu dự kiến thấp và chi phí tính năng trong Phần 4.3 (như một hình 1b và so sánh với một cây nông đắt hơn và kém chính xác hơn trong Hình 1a)

**Hình 1**: Minh họa các cây được tạo bởi các phương thức khác nhau: Các nút phân chia được đánh số theo thứ tự chúng đã được tạo ra, các lá được biểu diễn bằng các chữ cái. Vị trí thẳng đứng của các nút tương ứng với chi phí tính năng cần thiết cho mỗi mẫu và độ dày của cạnh biểu thị số lượng mẫu di chuyển dọc theo cạnh này. Một cây được xây dựng bởi GreedyMiser được thể hiện trong (a): Phần lớn các mẫu đi dọc theo một con đường đòi hỏi một tính năng rất tốn kém. BudgetPrune chỉ có thể tỉa bỏ lá E, F, G và H mà không tương ứng với việc giảm chi phí lớn. Tuy nhiên, CEGB chỉ sử dụng hai phần tách rất rẻ cho hầu hết các mẫu (lá A và B) và xây dựng một cây con phức tạp cho người thiểu số khó phân loại. Cây được xây dựng thể hiện trong (b) là sâu nhưng vẫn rẻ để đánh giá trung bình.

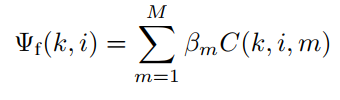
Gradient boosting cố gắng giảm thiểu rủi ro thực nghiệm của phương trình\, bằng cách xây dựng một sự kết hợp tuyến tính của K yếu tố dự báo yếu

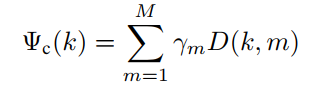
từ một tập F của các hàm được chấp nhận (không nhất thiết là cây quyết định). Bắt đầu với T0 (x) = 0 mỗi lần lặp k> 0 tạo ra một hàm yếu mới tk nhằm giảm tổn thất hiện tại. Các cập nhật tăng cường này có thể được hiểu là xấp xỉ của hướng gốc dốc trong không gian chức năng. Chúng tôi thực hiện theo ký hiệu, người sử dụng gradient tăng cường với các yếu tố dự báo yếu tk từ tập hợp các cây hồi quy T để giảm thiểu rủi ro thực nghiệm được chuẩn hóa

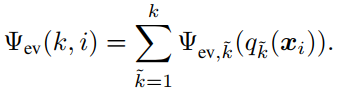
Thuật ngữ chuẩn hóa Ω (tk) phạt tính phức tạp của các hàm hồi quy của cây. Họ giả định rằng Ω (tk) chỉ phụ thuộc vào số lượng lá Lk và phản ứng lá wk và lấy được một thuật toán đơn giản để trực tiếp tìm hiểu chúng. Thay vào đó, chúng tôi sử dụng hình phạt dự đoán phức tạp hơn Ψ và sử dụng thuật toán xây dựng cây khác nhau cho phép tối ưu hóa với hình phạt chi phí.

1. **Prediction cost penalty**

Nhớ lại rằng đối với mỗi cây riêng lẻ, giá phạt dự đoán Ψ bao gồm hai thành phần: (i) chi phí mua lại tính năng Ψf và (ii) chi phí đánh giá cây Ψev. Tuy nhiên, chi phí dự đoán này cho cây thứ k, được gắn với phần còn lại của tất cả các lần lặp trước, phụ thuộc vào các cây trước đó. Cụ thể, đối với bất kỳ đầu vào x nào; các tính năng được sử dụng trong các cây của các lần lặp trước không đóng góp vào chi phí phạt một lần nữa. Do đó, chúng tôi sử dụng chức năng chỉ báo với

nếu và chỉ khi tính năng m được sử dụng để dự đoán xi bởi bất kỳ cây nào được xây dựng trước và bao gồm cả phép lặp k. Hơn nữa βm ≥ 0 là chi phí cho tính toán hoặc tính năng thu được m cho một đầu vào đơn x. Sau đó, đóng góp chi phí tính năng cho cây k đầu tiên được tính như:

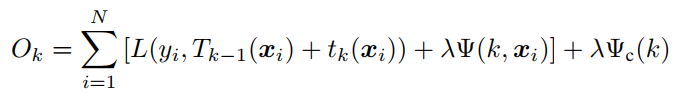
Các tính năng được tính cho tất cả các đầu vào cùng một lúc (ví dụ: bộ lọc chập tách) đóng góp cho hình phạt độc lập với cá thể x đang được đánh giá. Đối với những người chúng tôi sử dụng γm làm tổng chi phí tính toán và xác định hàm chỉ báo nếu và chỉ khi tính năng m được sử dụng cho bất kỳ đầu vào x nào trong bất kỳ cây nào được xây dựng trước và bao gồm cả phép lặp k. Sau đó

Chi phí đánh giá cho một đầu vào x duy nhất đi qua một cây là số lượng các nút phân tách giữa nút gốc và đầu cuối của lá đầu vào qk (x); nhân với một hằng số phù hợp α ≥ 0 để ghi lại chi phí để đánh giá một lần tách. Tổng chi phí cho cây k đầu tiên là tổng chi phí của mỗi cây

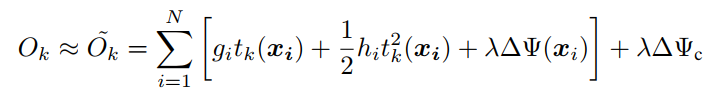
1. **Tree Boosting with Prediction Cost**

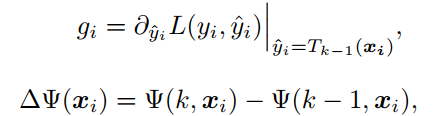
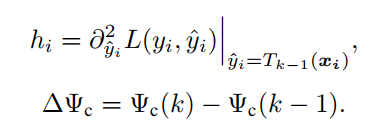
Bây giờ chúng ta đã định nghĩa tất cả các thành phần của phương trình. (2). Tối ưu hóa đồng thời tất cả các cây tk là không thể tránh khỏi. Thay vào đó, như trong việc tăng cường gradient, chúng ta thu nhỏ mục tiêu bằng cách bắt đầu với T0 (x) = 0 và lặp lại thêm một cây mới tại mỗi lần lặp.

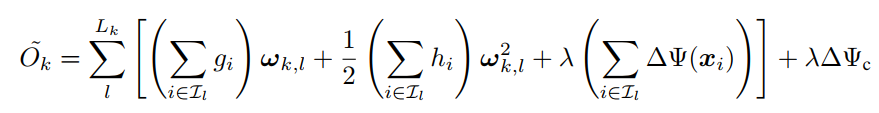
Tại bước lặp k chúng ta xây dựng cây hồi quy k thứ k bằng cách giảm thiểu mục tiêu sau

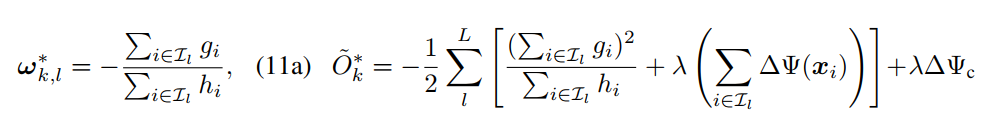


Với Lưu ý rằng hình phạt cho các đối tượng địa lý được tính cho tất cả đầu vào cùng một lúc, Ψc (k) không phụ thuộc vào x mà chỉ phụ thuộc vào cấu trúc của cây hiện tại và cây trước đó.

Trực tiếp tối ưu hóa mục tiêu Ok w.r.t. cây tk là khó khăn vì đối số tk xuất hiện bên trong hàm mất. Sau [8] chúng ta sử dụng lệnh thứ hai Taylor mở rộng sự mất mát xung quanh Tk − 1 (xi). Loại bỏ các thuật ngữ không đổi từ các lần lặp trước đó, hàm mục tiêu có thể được xấp xỉ bằng

Khi mà

Đối với cây quyết định tk (x) =! K; qk (x) với cấu trúc cố định qk

với tập hợp Il = fijqk (xi) = lg chứa đầu vào trong lá l. Đối với cấu trúc cố định này, trọng số tối ưu và mức giảm mục tiêu tốt nhất tương ứng có thể được tính toán rõ ràng: