**31.** **Interpreting learning curves: Other cases** ( Giải thích đường cong học tập : những trường hợp khác )

Xem xét đường cong học tập này:

A diagram of training error

Description automatically generated

Biểu đồ này cho thấy có độ lệch cao, phương sai cao hay cả hai?

Đường cong lỗi huấn luyện màu xanh dương tương đối thấp, trong khi đường cong lỗi trên tập phát triển (dev) màu đỏ lại cao hơn nhiều so với lỗi huấn luyện. Điều này cho thấy rằng độ lệch (bias) nhỏ, nhưng phương sai (variance) thì lớn. Việc thêm nhiều dữ liệu huấn luyện có thể giúp thu hẹp khoảng cách giữa lỗi phát triển và lỗi huấn luyện.

Giờ, hãy xem xét điều này:

A diagram of training and training

Description automatically generated

Lần này, lỗi huấn luyện lớn, vì nó cao hơn nhiều so với mức hiệu suất mong muốn. Lỗi trên tập phát triển (dev) cũng lớn hơn đáng kể so với lỗi huấn luyện. Do đó, bạn đang gặp phải vấn đề về cả độ lệch (bias) và phương sai (variance) đáng kể. Bạn sẽ cần tìm cách giảm cả hai yếu tố này trong thuật toán của mình.

**32. Plotting learning curves** ( Vẽ đường cong học tập ).

Giả sử bạn có một tập huấn luyện rất nhỏ gồm 100 ví dụ. Bạn huấn luyện thuật toán của mình bằng cách sử dụng một tập con được chọn ngẫu nhiên gồm 10 ví dụ, sau đó là 20 ví dụ, rồi 30, cho đến 100, tăng dần số lượng ví dụ theo khoảng 10. Sau đó, bạn sử dụng 10 điểm dữ liệu này để vẽ đường cong học tập của mình. Bạn có thể thấy rằng đường cong trông hơi nhiễu(lệch ) (có nghĩa là các giá trị cao hơn/thấp hơn dự kiến) ở tập huấn luyện có kích thước nhỏ.

Khi tập huấn luyện trên chỉ 10 ví dụ được chọn ngẫu nhiên, bạn có thể không may mắn và có một tập huấn luyện đặc biệt “xấu”, chẳng hạn như một tập huấn luyện có nhiều ví dụ mơ hồ/được dán nhãn sai. Hoặc, bạn có thể may mắn và có được một tập huấn luyện đặc biệt “tốt”. Việc có một tập huấn luyện nhỏ có nghĩa là lỗi phát triển và lỗi đào tạo có thể dao động ngẫu nhiên.

Nếu ứng dụng học máy của bạn bị lệch nặng về một lớp (chẳng hạn như một nhiệm vụ phân loại mèo mà phần trăm các ví dụ âm tính lớn hơn nhiều so với các ví dụ dương tính), hoặc nếu nó có một số lượng lớn các lớp (chẳng hạn như nhận dạng 100 loài động vật khác nhau), thì khả năng chọn một tập huấn luyện đặc biệt “không đại diện” hoặc xấu cũng lớn hơn. Ví dụ, nếu 80% các ví dụ của bạn là các ví dụ âm tính (y=0) và chỉ 20% là các ví dụ dương tính (y=1), thì có khả năng một tập huấn luyện gồm 10 ví dụ chỉ chứa các ví dụ âm tính, khiến cho thuật toán rất khó học được điều gì đó có ý nghĩa.

Nếu nhiễu trong đường cong học tập khiến khó nhìn thấy xu hướng thực sự, đây là hai giải pháp:

* Thay vì chỉ huấn luyện một mô hình trên 10 ví dụ, hãy chọn một số (giả sử 3-10) tập huấn luyện được chọn ngẫu nhiên khác nhau gồm 10 ví dụ bằng cách lấy mẫu có hoàn trả 10 từ tập gốc gồm 100 ví dụ của bạn. Huấn luyện một mô hình khác trên mỗi tập huấn luyện này và tính toán lỗi tập huấn luyện và lỗi tập phát triển của mỗi mô hình thu được. Tính toán và vẽ đồ thị lỗi đào tạo trung bình và lỗi phát triển trung bình.
* Nếu tập huấn luyện của bạn bị lệch về một lớp hoặc nếu nó có nhiều lớp, hãy chọn một tập con “cân bằng” thay vì 10 ví dụ huấn luyện ngẫu nhiên từ tập 100 ví dụ. Ví dụ, bạn có thể đảm bảo rằng 2/10 ví dụ là ví dụ dương tính và 8/10 là ví dụ âm tính. Nói chung, bạn có thể đảm bảo rằng tỷ lệ ví dụ từ mỗi lớp càng gần càng tốt với tỷ lệ tổng thể trong tập huấn luyện gốc.

Không nên sử dụng các kỹ thuật này trừ khi đã thử vẽ đường cong học tập và kết luận rằng các đường cong quá nhiễu để nhìn thấy các xu hướng cơ bản. Nếu tập huấn luyện lớn—giả sử hơn 10.000 ví dụ—và phân bố lớp không bị lệch nhiều, có thể sẽ không cần các kỹ thuật này.

Việc vẽ đường cong học tập có thể tốn nhiều tài nguyên tính toán: Ví dụ, có thể phải huấn luyện mười mô hình với 1.000, rồi 2.000, tất cả lên đến 10.000 ví dụ. Huấn luyện các mô hình với tập dữ liệu nhỏ nhanh hơn nhiều so với huấn luyện các mô hình với tập dữ liệu lớn. Do đó, thay vì cách đều khoảng cách giữa các kích thước tập huấn luyện trên thang tuyến tính, có thể huấn luyện các mô hình với 1.000, 2.000, 4.000, 6.000 và 10.000 ví dụ. Điều này vẫn sẽ cung cấp một ý tưởng rõ ràng về các xu hướng trong đường cong học tập. Tất nhiên, kỹ thuật này chỉ có ý nghĩa nếu chi phí tính toán để huấn luyện tất cả các mô hình bổ sung là đáng kể.

**Comparing to human-level performance** (So sánh với hiệu suất của con người)

**33. Why we compare to human-level performance ?** (Tại sao chúng ta so sánh với hiệu suất của con người?)

Nhiều hệ thống học máy nhằm tự động hóa những việc mà con người làm tốt. Các ví dụ bao gồm nhận dạng hình ảnh, nhận dạng giọng nói và phân loại thư rác email. Các thuật toán học tập cũng đã được cải thiện rất nhiều đến mức chúng ta đang vượt xa hiệu suất của con người ở ngày càng nhiều nhiệm vụ này.

**Hơn nữa, có một số lý do khiến việc xây dựng hệ thống ML trở nên dễ dàng hơn nếu bạn đang cố gắng thực hiện một nhiệm vụ mà con người có thể làm tốt:**

1. **Dễ dàng thu thập dữ liệu từ người dán nhãn:** Ví dụ, vì con người nhận ra hình ảnh mèo tốt nên rất đơn giản cho con người cung cấp nhãn có độ chính xác cao cho thuật toán học của bạn.
2. **Phân tích lỗi có thể dựa vào trực giác của con người:** Giả sử một thuật toán nhận dạng giọng nói đang hoạt động kém hơn so với nhận dạng của con người. Giả sử nó sao chép không chính xác một đoạn âm thanh thành “Công thức này yêu cầu một quả lê táo,” nhầm lẫn “đôi” thành “lê”. Bạn có thể dựa vào trực giác của con người và cố gắng hiểu thông tin mà một người sử dụng để có được bản sao chép chính xác và sử dụng kiến ​​thức này để sửa đổi thuật toán học.
3. **Sử dụng hiệu suất của con người để ước tính tỷ lệ lỗi tối ưu và cũng đặt một “tỷ lệ lỗi mong muốn”.** Giả sử thuật toán của bạn đạt được lỗi 10% trong một nhiệm vụ, nhưng một người đạt được lỗi 2%. Sau đó, chúng ta biết rằng tỷ lệ lỗi tối ưu là 2% hoặc thấp hơn và độ lệch tránh được là ít nhất 8%. Do đó, bạn nên thử các kỹ thuật giảm độ lệch.

Mặc dù mục #3 có vẻ không quan trọng, tôi thấy rằng việc có một tỷ lệ lỗi mục tiêu hợp lý và có thể đạt được giúp tăng tốc tiến độ của một nhóm. Biết thuật toán của bạn có độ lệch tránh được cao là vô cùng giá trị và mở ra một loạt tùy chọn để thử.

Có một số nhiệm vụ mà ngay cả con người cũng không giỏi. Ví dụ, chọn một cuốn sách để giới thiệu cho bạn; hoặc chọn một quảng cáo để hiển thị cho người dùng trên một trang web; hoặc dự đoán thị trường chứng khoán. Máy tính đã vượt qua hiệu suất của hầu hết mọi người trong các nhiệm vụ này. Với các ứng dụng này, chúng ta gặp phải các vấn đề sau:

* **Khó khăn trong việc thu thập nhãn:** Ví dụ, khó cho người dán nhãn để chú thích cơ sở dữ liệu của người dùng với đề xuất sách “tối ưu”. Nếu bạn vận hành một trang web hoặc ứng dụng bán sách, bạn có thể thu thập dữ liệu bằng cách hiển thị sách cho người dùng và xem họ mua gì. Nếu bạn không vận hành một trang web như vậy, bạn cần tìm cách sáng tạo hơn để lấy dữ liệu.
* • **Trực giác của con người khó tin tưởng vào hơn:** Ví dụ, hầu như không ai có thể dự đoán được thị trường chứng khoán. Vì vậy, nếu thuật toán dự đoán cổ phiếu của chúng ta không tốt hơn so với đoán ngẫu nhiên, thì rất khó để tìm ra cách cải thiện nó.
* • **Khó biết tỷ lệ lỗi tối ưu và tỷ lệ lỗi mong muốn hợp lý là bao nhiêu:** Giả sử bạn đã có một hệ thống đề xuất sách đang hoạt động khá tốt. Làm thế nào bạn biết nó có thể cải thiện bao nhiêu nữa mà không có một cơ sở so sánh từ con người?

**34. How to define human-level performance**( cách xác định hiệu xuất của con người )

Giả sử bạn đang làm việc trên một ứng dụng hình ảnh y tế tự động đưa ra chẩn đoán từ hình ảnh X-quang. Một người điển hình không có nền tảng y tế trước đó ngoài một số đào tạo cơ bản đạt được 15% lỗi trong nhiệm vụ này. Một bác sĩ trẻ đạt được 10% lỗi. Một bác sĩ giàu kinh nghiệm đạt được 5% lỗi. Và một nhóm nhỏ các bác sĩ thảo luận và tranh luận về từng hình ảnh đạt được 2% lỗi. Tỷ lệ lỗi nào trong số này xác định “hiệu suất của con người”?

Trong trường hợp này, tôi sẽ sử dụng 2% làm thước đo hiệu suất của con người cho tỷ lệ lỗi tối ưu của chúng tôi. Bạn cũng có thể đặt 2% làm mức hiệu suất mong muốn vì tất cả ba lý do từ chương trước để so sánh với hiệu suất của con người đều áp dụng:

* **Dễ dàng thu thập dữ liệu từ người dán nhãn:** Các bác sĩ có thể cung cấp nhãn chính xác cho thuật toán học.
* **Phân tích lỗi dựa trên trực giác của con người:** Các bác sĩ có thể sử dụng trực giác của họ để hiểu sai sót của thuật toán và cải thiện nó.
* **Đặt mục tiêu hiệu suất:**  **Sử dụng hiệu suất cấp độ con người để ước tính tỷ lệ lỗi tối ưu và cũng đặt "tỷ lệ lỗi mong muốn" có thể đạt được**. Trong trường hợp này, có thể sử dụng 2% làm ước tính tỷ lệ lỗi tối ưu. Mặc dù tỷ lệ lỗi tối ưu có thể thấp hơn 2%, nhưng không thể cao hơn, vì một nhóm bác sĩ có thể đạt được 2% lỗi. Ngược lại, việc sử dụng 5% hoặc 10% làm ước tính tỷ lệ lỗi tối ưu là không hợp lý, vì chúng ta biết những ước tính này chắc chắn quá cao.

Nếu hệ thống hiện tại có tỷ lệ lỗi cao, việc sử dụng một bác sĩ trẻ để dán nhãn dữ liệu và cung cấp trực giác có thể tiết kiệm thời gian và chi phí hơn so với sử dụng một nhóm bác sĩ giàu kinh nghiệm. Tuy nhiên, khi hệ thống đã đạt được tỷ lệ lỗi thấp hơn, việc sử dụng một nhóm bác sĩ giàu kinh nghiệm có thể giúp cải thiện hiệu suất hơn.

Nếu hệ thống của bạn hiện đang ở mức lỗi 40%, thì việc bạn sử dụng bác sĩ cơ sở (lỗi 10%) hay bác sĩ có kinh nghiệm (lỗi 5%) để dán nhãn dữ liệu và cung cấp trực giác không quan trọng. Nhưng nếu hệ thống của bạn đã ở mức lỗi 10%, thì việc xác định tham chiếu cấp độ con người là 2% sẽ cung cấp cho bạn các công cụ tốt hơn để tiếp tục cải thiện hệ thống của mình.

**35. Superpassing human-level performance** ( vượt qua hiệu suất con người )

Giả sử bạn đang phát triển một hệ thống nhận dạng giọng nói và có một tập dữ liệu các đoạn âm thanh. Nếu tập dữ liệu chứa nhiều đoạn âm thanh nhiễu đến mức ngay cả con người cũng mắc lỗi 10%, và hệ thống của bạn đã đạt được 8% lỗi, liệu bạn có thể sử dụng các kỹ thuật trong Chương 33 để tiếp tục cải thiện hiệu suất không?

Nếu bạn xác định được một tập con dữ liệu mà con người vượt trội đáng kể so với hệ thống của bạn, bạn vẫn có thể áp dụng các kỹ thuật đó để thúc đẩy tiến bộ nhanh chóng. Ví dụ, nếu hệ thống của bạn tốt hơn nhiều so với con người trong việc nhận dạng giọng nói trong âm thanh ồn ào, nhưng con người vẫn giỏi hơn trong việc phiên âm lời nói được nói rất nhanh, bạn có thể tập trung vào cải thiện hệ thống trong lĩnh vực này.

**Đối với tập con dữ liệu với lời nói được nói nhanh:**

1. Bạn vẫn có thể thu thập bản sao chép từ con người có chất lượng cao hơn so với đầu ra của thuật toán của bạn.
2. Bạn có thể dựa vào trực giác của con người để hiểu lý do tại sao họ nghe chính xác một lời nói nhanh trong khi hệ thống của bạn thì không.
3. Bạn có thể sử dụng hiệu suất ở cấp độ con người đối với giọng nói nhanh như một mục tiêu hiệu suất mong muốn.

Nói chung, miễn là có những ví dụ trong tập phát triển (dev set) mà con người đúng và thuật toán của bạn sai, thì nhiều kỹ thuật đã được mô tả trước đây sẽ vẫn áp dụng. Điều này đúng ngay cả khi, trung bình trên toàn bộ tập phát triển/kiểm thử, hiệu suất của bạn đã vượt qua hiệu suất của con người.

Có nhiều ứng dụng học máy quan trọng mà máy móc vượt qua hiệu suất của con người. Ví dụ, máy móc giỏi hơn trong việc dự đoán xếp hạng phim, dự đoán thời gian mà một xe giao hàng cần để đến một địa điểm, hoặc quyết định có nên phê duyệt đơn vay vốn hay không. Chỉ một phần các kỹ thuật được áp dụng khi con người gặp khó khăn trong việc xác định các ví dụ mà thuật toán rõ ràng sai. Do đó, tiến bộ thường chậm hơn đối với các vấn đề mà máy móc đã vượt qua hiệu suất của con người, trong khi tiến bộ thường nhanh hơn khi máy móc vẫn đang cố gắng theo kịp con người

Training and testing on different distributions( Đào tạo và thử nghiêm trên các bản phân phối khác nhau )

**36. Khi nào bạn nên đào tạo và thử nghiện trên các tập phân phối khác nhau** (When you should train and test on different distributions)

Người dùng của ứng dụng hình ảnh mèo của bạn đã tải lên 10.000 hình ảnh, và bạn đã gán nhãn thủ công cho các hình ảnh đó, xác định có chứa mèo hay không. Bạn cũng có một tập dữ liệu lớn hơn gồm 200.000 hình ảnh mà bạn tải về từ internet. Vậy bạn nên định nghĩa tập huấn luyện, tập phát triển (dev), và tập kiểm tra (test) như thế nào?

Vì 10.000 hình ảnh từ người dùng phản ánh khá chính xác phân phối xác suất của dữ liệu mà bạn muốn đạt kết quả tốt, bạn có thể sử dụng tập dữ liệu đó cho các tập dev và test. Nếu bạn đang huấn luyện một thuật toán học sâu (deep learning) cần nhiều dữ liệu, bạn có thể sử dụng thêm 200.000 hình ảnh từ internet cho việc huấn luyện. Do đó, các tập huấn luyện và dev/test của bạn sẽ đến từ các phân phối xác suất khác nhau. Điều này sẽ ảnh hưởng như thế nào đến công việc của bạn?

Thay vì chia dữ liệu thành các tập huấn luyện, phát triển (dev) và kiểm tra (test), chúng ta có thể lấy tất cả 210.000 hình ảnh có sẵn và xáo trộn ngẫu nhiên chúng thành các tập train/dev/test. Trong trường hợp này, tất cả dữ liệu đều đến từ cùng một phân phối. Nhưng tôi khuyên không nên sử dụng phương pháp này, vì khoảng 205.000/210.000 ≈ 97,6% dữ liệu dev/test của bạn sẽ đến từ các hình ảnh trên internet, mà không phản ánh chính xác phân phối dữ liệu bạn muốn đạt kết quả tốt. Hãy nhớ lời khuyên của chúng tôi về việc chọn các tập dev/test:

Hãy chọn các tập dev và test sao cho phản ánh dữ liệu mà bạn dự đoán sẽ nhận được trong tương lai và muốn đạt kết quả tốt trên đó.

Phần lớn tài liệu học thuật về học máy giả định rằng tập huấn luyện, tập phát triển (dev) và tập kiểm tra (test) đều đến từ cùng một phân phối. Trong những ngày đầu của học máy, dữ liệu khan hiếm. Chúng ta thường chỉ có một tập dữ liệu được lấy từ một phân phối xác suất nào đó. Vì vậy, chúng ta sẽ chia ngẫu nhiên dữ liệu đó thành các tập huấn luyện, dev và test, và giả định rằng tất cả dữ liệu đều đến từ cùng một nguồn thường được đáp ứng.

Nhưng trong kỷ nguyên của dữ liệu lớn, chúng ta giờ đây có quyền truy cập vào những tập huấn luyện khổng lồ, chẳng hạn như các hình ảnh mèo trên internet. Ngay cả khi tập huấn luyện đến từ một phân phối khác với tập dev/test, chúng ta vẫn muốn sử dụng nó để học hỏi vì nó có thể cung cấp rất nhiều thông tin.

Trong ví dụ về bộ phát hiện mèo, thay vì đưa toàn bộ 10.000 hình ảnh do người dùng tải lên vào các tập dev/test, chúng ta có thể chỉ đưa 5.000 hình ảnh vào các tập dev/test. Chúng ta có thể đưa 5.000 ví dụ còn lại do người dùng tải lên vào tập huấn luyện. Bằng cách này, tập huấn luyện của bạn gồm 205.000 ví dụ sẽ chứa một số dữ liệu đến từ phân phối của các tập dev/test cùng với 200.000 hình ảnh từ internet. Chúng ta sẽ thảo luận trong chương sau tại sao phương pháp này lại hữu ích.

Hãy xem xét một ví dụ thứ hai. Giả sử bạn đang xây dựng một hệ thống nhận dạng giọng nói để phiên âm địa chỉ đường phố cho một ứng dụng bản đồ/dẫn đường di động điều khiển bằng giọng nói. Bạn có 20.000 ví dụ về người dùng nói địa chỉ đường phố. Nhưng bạn cũng có 500.000 ví dụ khác là các đoạn âm thanh khác với người dùng nói về các chủ đề khác. Bạn có thể lấy 10.000 ví dụ về địa chỉ đường phố cho các tập dev/test, và sử dụng 10.000 ví dụ còn lại cùng với 500.000 ví dụ khác để huấn luyện.

Chúng ta sẽ tiếp tục giả định rằng dữ liệu dev của bạn và dữ liệu test đến từ cùng một phân phối. Tuy nhiên, điều quan trọng cần hiểu là các phân phối khác nhau giữa tập huấn luyện và tập dev/test mang lại một số thách thức đặc biệt.

**37. how to decide whether to use all your data.** ( làm sao để quyết định có sử dụng tất cả dữ liệu của bạn hay không ).

Giả sử tập huấn luyện của bộ phát hiện mèo của bạn bao gồm 10.000 hình ảnh do người dùng tải lên. Dữ liệu này đến từ cùng một phân phối với tập dev/test riêng biệt và đại diện cho phân phối mà bạn muốn đạt kết quả tốt. Bạn cũng có thêm 20.000 hình ảnh tải từ internet. Vậy bạn có nên cung cấp tất cả 20.000 + 10.000 = 30.000 hình ảnh cho thuật toán học của mình như một tập huấn luyện, hay loại bỏ 20.000 hình ảnh từ internet vì sợ rằng nó sẽ gây lệch lạc cho thuật toán của bạn?

Khi sử dụng các thế hệ thuật toán học trước đây (chẳng hạn như các đặc trưng thị giác máy tính do con người thiết kế, tiếp theo là bộ phân loại tuyến tính đơn giản), có một rủi ro thực sự rằng việc kết hợp cả hai loại dữ liệu sẽ khiến bạn đạt kết quả tệ hơn. Do đó, một số kỹ sư có thể cảnh báo bạn không nên bao gồm 20.000 hình ảnh từ internet.

Nhưng trong kỷ nguyên hiện đại của các thuật toán học mạnh mẽ và linh hoạt—chẳng hạn như các mạng nơ-ron lớn—rủi ro này đã giảm đáng kể. Nếu bạn có đủ khả năng để xây dựng một mạng nơ-ron với số lượng đơn vị/lớp ẩn đủ lớn, bạn có thể an toàn thêm 20.000 hình ảnh vào tập huấn luyện của mình. Việc thêm các hình ảnh này nhiều khả năng sẽ cải thiện hiệu suất của bạn.

Nhận định này dựa trên thực tế rằng có một ánh xạ x —> y nào đó hoạt động tốt cho cả hai loại dữ liệu. Nói cách khác, tồn tại một hệ thống có thể nhập vào một hình ảnh từ internet hoặc một hình ảnh từ ứng dụng di động và dự đoán nhãn một cách đáng tin cậy, ngay cả khi không biết nguồn gốc của hình ảnh.

Việc thêm 20.000 hình ảnh bổ sung sẽ có những tác động sau:

1.Nó cung cấp cho mạng nơ-ron của bạn nhiều ví dụ hơn về cách mèo trông như thế nào hoặc không trông như thế nào. Điều này hữu ích vì hình ảnh trên internet và hình ảnh do người dùng tải lên từ ứng dụng di động có một số điểm tương đồng. Mạng nơ-ron của bạn có thể áp dụng một phần kiến thức thu được từ hình ảnh trên internet vào hình ảnh từ ứng dụng di động.

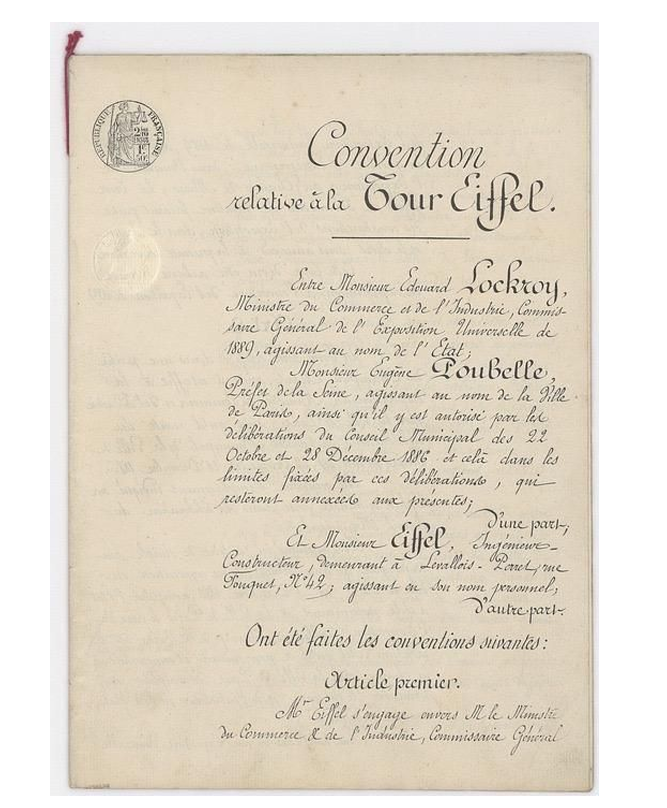
2.Nó buộc mạng nơ-ron phải sử dụng một phần năng lực của mình để học về các đặc điểm riêng biệt của hình ảnh trên internet (chẳng hạn như độ phân giải cao hơn, các phân phối khác nhau về cách bố trí khung hình, v.v.). Nếu những đặc điểm này khác biệt nhiều so với hình ảnh trên ứng dụng di động, nó sẽ 'sử dụng' một phần khả năng biểu diễn của mạng nơ-ron. Do đó, sẽ có ít khả năng hơn để nhận diện dữ liệu từ phân phối của hình ảnh trên ứng dụng di động, thứ mà bạn thực sự quan tâm. Về mặt lý thuyết, điều này có thể ảnh hưởng đến hiệu suất của thuật toán.

Để mô tả tác động thứ hai bằng các thuật ngữ khác, chúng ta có thể quay trở lại với nhân vật hư cấu Sherlock Holmes, người đã nói rằng não của bạn giống như một gác mái; nó chỉ có một lượng không gian hữu hạn. Ông nói rằng 'mỗi khi có thêm kiến thức, bạn sẽ quên đi điều gì đó mà bạn đã biết trước đó. Do đó, điều quan trọng nhất là không để những sự thật vô dụng chen lấn ra ngoài những điều hữu ích.'

May mắn thay, nếu bạn có đủ khả năng tính toán cần thiết để xây dựng một mạng nơ-ron lớn đủ—tức là, một gác mái đủ lớn—thì đây không phải là một mối quan tâm nghiêm trọng. Bạn có đủ khả năng để học từ cả hình ảnh trên internet và hình ảnh từ ứng dụng di động, mà không phải lo lắng về việc hai loại dữ liệu này cạnh tranh cho không gian. 'Cái não' của thuật toán của bạn đủ lớn để bạn không phải lo lắng về việc hết không gian gác mái.

Nhưng nếu bạn không có một mạng nơ-ron đủ lớn (hoặc một thuật toán học linh hoạt khác), thì bạn nên chú ý nhiều hơn đến việc dữ liệu huấn luyện của bạn phải khớp với phân phối của bộ dữ liệu dev/test.

Nếu bạn nghĩ rằng bạn có dữ liệu mà không mang lại lợi ích gì, bạn nên bỏ qua dữ liệu đó vì lý do tính toán. Ví dụ, giả sử bộ dữ liệu dev/test của bạn chủ yếu chứa hình ảnh thông thường về con người, địa điểm, danh lam thắng cảnh, động vật. Giả sử bạn cũng có một bộ sưu tập lớn các tài liệu lịch sử được quét:



Những tài liệu này không chứa bất cứ điều gì giống như một con mèo. Chúng cũng hoàn toàn không giống với phân phối trong bộ dữ liệu dev/test của bạn. Không có lý do gì để bao gồm dữ liệu này như là những ví dụ tiêu cực, vì lợi ích từ tác động đầu tiên ở trên là không đáng kể—gần như không có gì mà mạng nơ-ron của bạn có thể học được từ dữ liệu này để áp dụng cho phân phối bộ dữ liệu dev/test của bạn. Việc bao gồm chúng sẽ lãng phí tài nguyên tính toán và khả năng biểu diễn của mạng nơ-ron.

**38. how to decide whether to include inconsistent data** ( cách quyết định có bao gồm dữ liệu nhất quán không )

Giả sử bạn muốn học cách dự đoán giá nhà ở Thành phố New York. Dựa vào kích thước của một ngôi nhà (đặc trưng đầu vào x), bạn muốn dự đoán giá (nhãn mục tiêu y).

Giá nhà ở Thành phố New York rất cao. Giả sử bạn có một bộ dữ liệu thứ hai về giá nhà ở Detroit, Michigan, nơi mà giá nhà thấp hơn nhiều. Bạn có nên bao gồm dữ liệu này vào bộ huấn luyện của mình không?

Khi kích thước x giống nhau, giá của một ngôi nhà y rất khác nhau tùy thuộc vào việc nó ở Thành phố New York hay ở Detroit. Nếu bạn chỉ quan tâm đến việc dự đoán giá nhà ở Thành phố New York, việc kết hợp hai bộ dữ liệu này sẽ làm giảm hiệu suất của bạn. Trong trường hợp này, tốt hơn là nên bỏ qua dữ liệu không nhất quán từ Detroit.

Làm thế nào ví dụ về Thành phố New York và Detroit khác với ví dụ về hình ảnh mèo trên ứng dụng di động và trên internet?

Ví dụ về hình ảnh mèo khác biệt vì, với một bức tranh đầu vào x, ta có thể dự đoán một cách đáng tin cậy nhãn y cho biết liệu có mèo hay không, ngay cả khi không biết bức tranh đó có phải là hình ảnh từ internet hay từ ứng dụng di động. Nghĩa là, có một hàm f(x) mà ánh xạ một cách đáng tin cậy từ đầu vào x đến đầu ra mục tiêu y, ngay cả khi không biết nguồn gốc của x. Do đó, nhiệm vụ nhận diện từ hình ảnh internet là "nhất quán" với nhiệm vụ nhận diện từ hình ảnh ứng dụng di động. Điều này có nghĩa là việc bao gồm tất cả dữ liệu không có nhiều nhược điểm (ngoài chi phí tính toán), và có thể có một số lợi ích đáng kể. Ngược lại, dữ liệu của Thành phố New York và Detroit thì không nhất quán. Khi kích thước x (kích thước ngôi nhà) giống nhau, giá rất khác nhau tùy thuộc vào nơi mà ngôi nhà đó ở.

*( Có một cách để giải quyết vấn đề dữ liệu từ Detroit không nhất quán với dữ liệu từ Thành phố New York, đó là thêm một đặc trưng bổ sung vào mỗi ví dụ huấn luyện để chỉ định thành phố. Với một đầu vào x—bây giờ chỉ rõ thành phố—giá trị mục tiêu y bây giờ trở nên rõ ràng. Tuy nhiên, trong thực tế, tôi không thấy điều này thường xuyên được thực hiện.)*

**39. Weighting data** ( trọng số dữ liệu )

39. Cân nhắc trọng số dữ liệu

Giả sử bạn có 200.000 hình ảnh từ internet và 5.000 hình ảnh từ người dùng ứng dụng di động của bạn. Có tỷ lệ 40:1 giữa kích thước của các tập dữ liệu này. Về lý thuyết, miễn là bạn xây dựng một mạng nơ-ron lớn và đào tạo nó đủ lâu trên tất cả 205.000 hình ảnh, không có gì sai khi cố gắng làm cho thuật toán hoạt động tốt trên cả hình ảnh từ internet và hình ảnh từ di động.

Nhưng trên thực tế, việc có số lượng hình ảnh từ internet lớn hơn 40 lần so với hình ảnh ứng dụng di động có thể có nghĩa là bạn cần tiêu tốn tài nguyên tính toán gấp 40 lần (hoặc nhiều hơn) để mô hình hóa cả hai loại dữ liệu, so với chỉ việc đào tạo trên 5.000 hình ảnh.

Nếu bạn không có tài nguyên tính toán lớn, bạn có thể giảm trọng số cho các hình ảnh từ internet như một sự thỏa hiệp.

Ví dụ, giả sử mục tiêu tối ưu hóa của bạn là sai số bình phương (đây không phải là sự lựa chọn tốt cho một bài toán phân loại, nhưng nó sẽ đơn giản hóa lời giải thích của chúng tôi). Do đó, thuật toán học của chúng tôi cố gắng tối ưu hóa:

Tổng đầu tiên ở trên là qua 5.000 hình ảnh di động, và tổng thứ hai là qua 200.000 hình ảnh từ internet. Bạn có thể tối ưu hóa với một tham số bổ sung \(\beta\):

\[

\min\_{\theta} \sum\_{(x,y) \in \text{MobileImg}} \left( h\_\theta(x) - y \right)^2 + \beta \sum\_{(x,y) \in \text{InternetImg}} \left( h\_\theta(x) - y \right)^2

\]

Nếu bạn đặt \(\beta = 1/40\), thuật toán sẽ gán trọng số tương đương cho 5.000 hình ảnh di động và 200.000 hình ảnh từ internet. Bạn cũng có thể đặt tham số \(\beta\) thành các giá trị khác, có thể bằng cách điều chỉnh dựa trên tập dev.

Bằng cách giảm trọng số cho các hình ảnh từ internet, bạn sẽ không phải xây dựng một mạng nơ-ron quá lớn để đảm bảo rằng thuật toán hoạt động tốt trên cả hai loại nhiệm vụ. Kiểu điều chỉnh trọng số này chỉ cần thiết khi bạn nghi ngờ rằng dữ liệu bổ sung (hình ảnh từ internet) có sự phân phối rất khác so với tập dev/test, hoặc nếu dữ liệu bổ sung lớn hơn nhiều so với dữ liệu đến từ cùng phân phối như tập dev/test (hình ảnh di động).

**40. Generalizing from the training set to the dev set** ( khát quat từ tập đào tạo đến tập phát triển dev)

Giả sử bạn đang áp dụng Machine Learning (ML) trong một bối cảnh mà phân phối của tập huấn luyện và tập phát triển/test là khác nhau. Ví dụ, tập huấn luyện bao gồm hình ảnh từ Internet + hình ảnh di động, trong khi các tập phát triển/test chỉ chứa hình ảnh di động. Tuy nhiên, thuật toán không hoạt động tốt: nó có lỗi cao hơn nhiều trên tập phát triển/test so với mong muốn của bạn. Dưới đây là một số khả năng về những gì có thể sai:

1. **Không hoạt động tốt trên tập huấn luyện.** Đây là vấn đề của độ thiên lệch cao (có thể tránh được) trên phân phối của tập huấn luyện.
2. **Hoạt động tốt trên tập huấn luyện, nhưng không tổng quát tốt trên dữ liệu chưa thấy trước đó** lấy từ cùng một phân phối như tập huấn luyện. Đây là độ phương sai cao.
3. **Tổng quát tốt trên dữ liệu mới lấy từ cùng một phân phối như tập huấn luyện, nhưng không tổng quát tốt trên dữ liệu lấy từ phân phối tập phát triển/test.** Chúng tôi gọi vấn đề này là sự không tương thích dữ liệu, vì nó xuất phát từ việc dữ liệu trong tập huấn luyện không phù hợp tốt với dữ liệu trong tập phát triển/test.

**Ví dụ**, giả sử rằng con người đạt được hiệu suất gần như hoàn hảo trong nhiệm vụ nhận diện mèo. Thuật toán của bạn đạt được như sau:

* **1% lỗi** trên tập huấn luyện
* **1.5% lỗi** trên dữ liệu được lấy từ cùng một phân phối như tập huấn luyện mà thuật toán chưa thấy
* **10% lỗi** trên tập phát triển

Trong trường hợp này, bạn rõ ràng có vấn đề về sự không tương thích dữ liệu. Để giải quyết điều này, bạn có thể cố gắng làm cho dữ liệu huấn luyện giống hơn với dữ liệu phát triển/test. Chúng tôi sẽ thảo luận về một số kỹ thuật cho điều này sau.

Để chẩn đoán mức độ mà một thuật toán gặp phải từng vấn đề 1-3 ở trên, sẽ rất hữu ích nếu có một tập dữ liệu khác. Cụ thể, thay vì cung cấp cho thuật toán toàn bộ dữ liệu huấn luyện có sẵn, bạn có thể chia nó thành hai tập con: Tập huấn luyện thực tế mà thuật toán sẽ được huấn luyện trên, và một tập riêng biệt, mà chúng tôi sẽ gọi là "tập phát triển huấn luyện," mà chúng tôi sẽ không huấn luyện trên đó.

Bạn hiện có bốn tập con dữ liệu:

* **Tập huấn luyện**: Đây là dữ liệu mà thuật toán sẽ học từ (ví dụ: hình ảnh từ Internet + hình ảnh từ ứng dụng di động). Tập này không nhất thiết phải được lấy từ cùng một phân phối như những gì chúng ta thực sự quan tâm (phân phối tập phát triển/test).
* **Tập phát triển huấn luyện**: Dữ liệu này được lấy từ cùng một phân phối với tập huấn luyện (ví dụ: hình ảnh từ Internet + hình ảnh từ ứng dụng di động). Tập này thường nhỏ hơn tập huấn luyện; nó chỉ cần đủ lớn để đánh giá và theo dõi tiến trình của thuật toán học của chúng ta.
* **Tập phát triển**: Tập này được lấy từ cùng một phân phối với tập kiểm tra, và phản ánh phân phối dữ liệu mà chúng ta cuối cùng muốn đạt được hiệu suất tốt (ví dụ: hình ảnh từ ứng dụng di động).
* **Tập kiểm tra**: Tập này được lấy từ cùng một phân phối với tập phát triển (ví dụ: hình ảnh từ ứng dụng di động).

Với bốn tập dữ liệu này, bạn có thể đánh giá:

* **Lỗi huấn luyện**, bằng cách đánh giá trên tập huấn luyện.
* **Khả năng tổng quát của thuật toán** với dữ liệu mới được lấy từ phân phối tập huấn luyện, bằng cách đánh giá trên tập phát triển huấn luyện.
* **Hiệu suất của thuật toán** trên nhiệm vụ mà bạn quan tâm, bằng cách đánh giá trên các tập phát triển và/hoặc kiểm tra.

Hầu hết các hướng dẫn trong các Chương 5-7 về việc chọn kích thước cho tập phát triển cũng áp dụng cho tập phát triển huấn luyện.