**Overfitting**

Overfitting là hiện tượng mô hình tìm được quá khớp với dữ liệu training. Việc quá khớp này có thể dẫn đến việc dự đoán nhầm nhiễu, và chất lượng mô hình không còn tốt trên dữ liệu test nữa. Dữ liệu test được giả sử là không được biết trước, và không được sử dụng để xây dựng các mô hình Machine Learning.

Ví dụ:

Có 50 điểm dữ liệu được tạo bằng một đa thức bậc ba cộng thêm nhiễu. Tập dữ liệu này được chia làm hai, 30 điểm dữ liệu màu đỏ cho training data, 20 điểm dữ liệu màu vàng cho test data. Đồ thị của đa thức bậc ba này được cho bởi đường màu xanh lá cây. Bài toán của chúng ta là giả sử ta không biết mô hình ban đầu mà chỉ biết các điểm dữ liệu, hãy tìm một mô hình “tốt” để mô tả dữ liệu đã cho.



*Hình 1: Underfitting và Overfitting với Polynominal Regression*

Bài toán này hoàn toàn có thể được giải quyết bằng Linear Regression với dữ liệu mở rộng cho một cặp điểm là với cho đa thức bậc . Điều quan trọng là chúng ta cần tìm bậc của đa thức cần tìm.

Trong ví dụ trên đây, độ phức tạp của mô hình có thể được coi là bậc của đa thức cần tìm. Rõ ràng là một đa thức bậc không vượt quá 29 có thể fit được hoàn toàn với 30 điểm trong training data. Chúng ta cùng xét vài giá trị . Với , mô hình không thực sự tốt vì mô hình dự đoán quá khác so với mô hình thực. Trong trường hợp này, ta nói mô hình bị ***underfitting***. Với, với các điểm dữ liệu trong khoảng của training data, mô hình dự đoán và mô hình thực là khá giống nhau. Tuy nhiên, về phía phải, đa thức bậc 8 cho kết quả hoàn toàn ngược với xu hướng của dữ liệu. Điều tương tự xảy ra trong trường hợp .Đa thức bậc 16 này quá fit dữ liệu trong khoảng đang xét, và quá fit, tức không được mượt trong khoảng dữ liệu training. Việc quá fit trong trường hợp bậc 16 không tốt vì mô hình đang cố gắng mô tả nhiễu hơn là dữ liệu. Hai trường hợp đa thức bậc cao này được gọi là Overfitting.

Về cơ bản, overfitting xảy ra khi mô hình quá phức tạp để mô phỏng training data. Điều này đặc biệt xảy ra khi lượng dữ liệu training quá nhỏ trong khi độ phức tạp của mô hình quá cao.

Một mô hình được coi là tốt **(*fit*)** nếu cả train error và test error đều thấp. Nếu train error thấp nhưng test error cao, ta nói mô hình bị ***overfitting***. Nếu train error cao và test error cao, ta nói mô hình bị ***underfitting***. Nếu train error cao nhưng test error thấp hiện tượng này phải rất may mắn mới xảy ra, hoặc có chỉ khi tập dữ liệu test quá nhỏ.

**Một số phương pháp tránh overfitting**

**Validation/Hold-out**

Phương pháp đơn giản nhất là trích từ tập training data ra một tập con nhỏ và thực hiện việc đánh giá mô hình trên tập con nhỏ này. Tập con nhỏ được trích ra từ training set này được gọi là ***validation set***. Lúc này, training set là phần còn lại của training set ban đầu. Train error được tính trên training set mới này, và có một khái niệm nữa được định nghĩa tương tự như trên validation error, tức error được tính trên tập validation.

Với khái niệm mới này, ta tìm mô hình sao cho cả train eror và validation error đều nhỏ, qua đó có thể dự đoán được rằng test error cũng nhỏ. Phương pháp thường được sử dụng là sử dụng nhiều mô hình khác nhau. Mô hình nào cho validation error nhỏ nhất sẽ là mô hình tốt.

Thông thường, ta bắt đầu từ mô hình đơn giản, sau đó tăng dần độ phức tạp của mô hình. Tới khi nào validation error có chiều hướng tăng lên thì chọn mô hình ngay trước đó. Chú ý rằng mô hình càng phức tạp, train error có xu hướng càng nhỏ đi.

Hình dưới đây mô tả ví dụ phía trên với bậc của đa thức tăng từ 1 đến 8. Tập validation bao gồm 10 điểm được lấy ra từ tập training ban đầu.



*Hình 2: bậc của đa thức*

Ta hãy tạm chỉ xét hai đường màu lam và đỏ, tương ứng với train error và validation error. Khi bậc của đa thức tăng lên, train error có xu hướng giảm. Điều này dễ hiểu vì đa thức bậc càng cao, dữ liệu càng được fit. Quan sát đường màu đỏ, khi bậc của đa thức là 3 hoặc 4 thì validation error thấp, sau đó tăng dần lên. Dựa vào validation error, ta có thể xác định được bậc cần chọn là 3 hoặc 4. Quan sát tiếp đường màu lục, tương ứng với test error, thật là trùng hợp, với bậc bằng 3 hoặc 4, test error cũng đạt giá trị nhỏ nhất, sau đó tăng dần lên. Vậy cách làm này ở đây đã tỏ ra hiệu quả.

Việc không sử dụng test data khi lựa chọn mô hình ở trên nhưng vẫn có được kết quả khả quan vì ta giả sử rằng validation data và test data có chung một đặc điểm nào đó. Và khi cả hai đều là unseen data, error trên hai tập này sẽ tương đối giống nhau.

**Cross-validation**

Trong nhiều trường hợp, chúng ta có rất hạn chế số lượng dữ liệu để xây dựng mô hình. Nếu lấy quá nhiều dữ liệu trong tập training ra làm dữ liệu validation, phần dữ liệu còn lại của tập training là không đủ để xây dựng mô hình. Lúc này, tập validation phải thật nhỏ để giữ được lượng dữ liệu cho training đủ lớn. Tuy nhiên, một vấn đề khác nảy sinh. Khi tập validation quá nhỏ, hiện tượng overfitting lại có thể xảy ra với tập training còn lại. Lúc này ta sẽ sử dụng ***cross-validation*** (kiểm định chéo).

Cross validation là một cải tiến của validation với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác nhau. Một cách thường đường sử dụng là chia tập training ra tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử, được gọi là run, một trong số tập con được lấy ra làm validata set. Mô hình sẽ được xây dựng dựa vào hợp của tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi ***là k-fold cross validation***.

**Regularization**

Một nhược điểm lớn của cross-validation là số lượng training runs tỉ lệ thuận với . Điều đáng nói là mô hình polynomial như trên chỉ có một tham số cần xác định là bậc của đa thức. Trong các bài toán Machine Learning, lượng tham số cần xác định thường lớn hơn nhiều, và khoảng giá trị của mỗi tham số cũng rộng hơn nhiều, chưa kể đến việc có những tham số có thể là số thực. Như vậy, việc chỉ xây dựng một mô hình thôi cũng là đã rất phức tạp rồi. Có một cách giúp số mô hình cần huấn luyện giảm đi nhiều, thậm chí chỉ một mô hình. Cách này có tên gọi chung là ***regularization***.

Regularization, một cách cơ bản, là thay đổi mô hình một chút để tránh overfitting trong khi vẫn giữ được tính tổng quát của nó (tính tổng quát là tính mô tả được nhiều dữ liệu, trong cả tập training và test). Một cách cụ thể hơn, ta sẽ tìm cách di chuyển nghiệm của bài toán tối ưu hàm mất mát tới một điểm gần nó. Hướng di chuyển sẽ là hướng làm cho mô hình ít phức tạp hơn mặc dù giá trị của hàm mất mát có tăng lên một chút. Một kĩ thuật đơn giản là: **early stopping**.

**Early Stopping**

Trong nhiều bài toán Machine Learning, chúng ta cần sử dụng các thuật toán lặp để tìm ra nghiệm, ví dụ như Gradient Descent. Nhìn chung, hàm mất mát giảm dần khi số vòng lặp tăng lên. Early stopping tức dừng thuật toán trước khi hàm mất mát đạt giá trị quá nhỏ, giúp tránh overfitting. Một kỹ thuật thường được sử dụng là tách từ training set ra một tập validation set như trên. Sau một (hoặc một số, ví dụ 50) vòng lặp, ta tính cả train error và validation error, đến khi validation error có chiều hướng tăng lên thì dừng lại, và quay lại sử dụng mô hình tương ứng với điểm và validation error đạt giá trị nhỏ.



*Hình 3: Mô hình được xác định tại vòng lặp mà validation error đạt giá trị nhỏ nhất*

**Thêm số hạng vào hàm mất mát**

Kỹ thuật regularization phổ biến nhất là thêm vào hàm mất mát một số hạng nữa. Số hạng này thường dùng để đánh giá độ phức tạp của mô hình. Số hạng này càng lớn, thì mô hình càng phức tạp. Hàm mất mát mới này thường được gọi là regularized loss function, thường được định nghĩa như sau:

Trong đó: được dùng để ký hiệu các biến trong mô hình, ký hiệu cho hàm mất mát (loss function) và là số hạng regularization. thường là một số dương để cân bằng giữa hai đại lượng ở vế phải.

**regularization**

**(t**ức norm 2 của hệ số)

Norm 2 là căn của tổng bình phương các phần tử.

**Regularizers for sparsity ()**

Norm 1 là tổng các trị tuyệt đối của tất cả các phần tử.