ITF: **Information Technology Faculty**
Thuyloi University

Quá khớp (Overfitting)

Trình bày: PGS.TS Nguyễn Hữu Quỳnh

Overfitting không phải là một thuật toán trong Machine Learning. Nó là một hiện tượng không mong muốn thường gặp, người xây dựng mô hình Machine Learning cần nắm được các kỹ thuật để tránh hiện tượng này.

Bài giảng được dựa trên giáo trình machine learning cơ bản

Giới thiệu

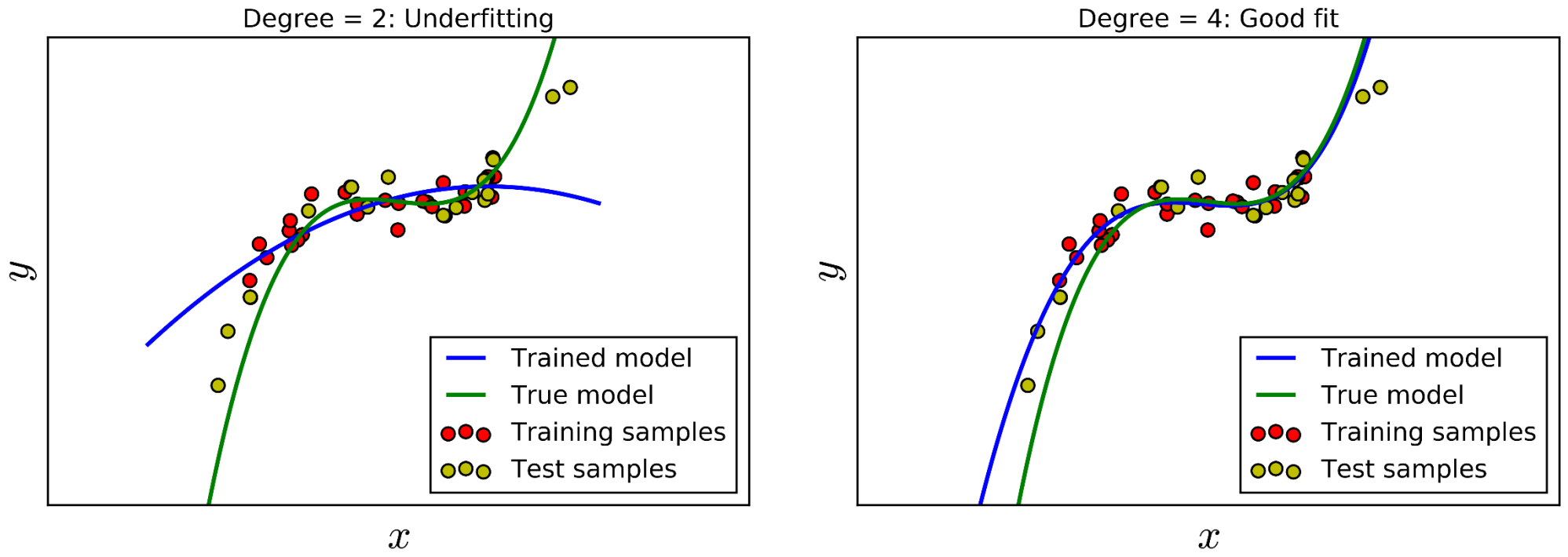
- Khi nói đến hồi qui tuyến tính, chúng ta sẽ muốn biết mục đích của nó để làm gì?
- Câu trả lời phổ biến là: về cơ bản, từ dữ liệu cho trước, chúng ta cần tìm một hàm số để biến các các điểm đầu vào thành các điểm đầu ra tương ứng, không cần chính xác, chỉ cần xấp xỉ.
- Nếu học toán tốt, chúng ta sẽ nghĩ: Đa thức Nội suy Lagrange có thể làm được điều đó, miễn là các điểm đầu vào khác nhau đôi một!
- Tuy nhiên: "những gì ta biết chỉ là nhỏ xíu so với những gì ta chưa biết", bài này sẽ lý giải câu nói này.

Giới thiệu

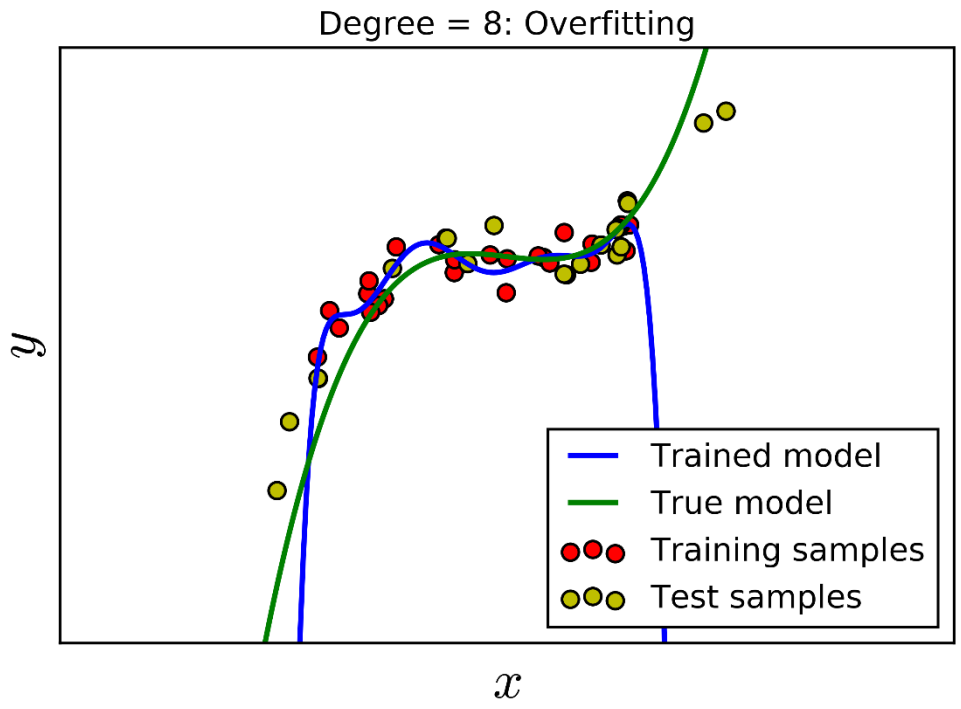
- Về Đa thức nội suy Lagrange:
  - Với N cặp điểm dữ liệu $(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)$ với các x_i kháu nhau đôi một, luôn tìm được một đa thức P(.) bậc không vượt quá N−1 sao cho $P(x_i)=y_i$, $\forall i=1,2,\ldots,N$.
  - Nghe có vẻ như điều này giống với việc ta đi tìm một mô hình phù hợp (fit) với dữ liệu trong bài toán Supervised Learning, thậm chí điều này còn tốt hơn vì trong Supervised Learning ta chỉ cần xấp xỉ.
- Tuy nhiên, nếu một mô hình *quá fit* với dữ liệu thì nó sẽ gây phản tác dụng. Hiện tượng *quá fit* này trong Machine Learning được gọi là *overfitting*, là điều mà khi xây dựng mô hình, chúng ta luôn cần tránh:

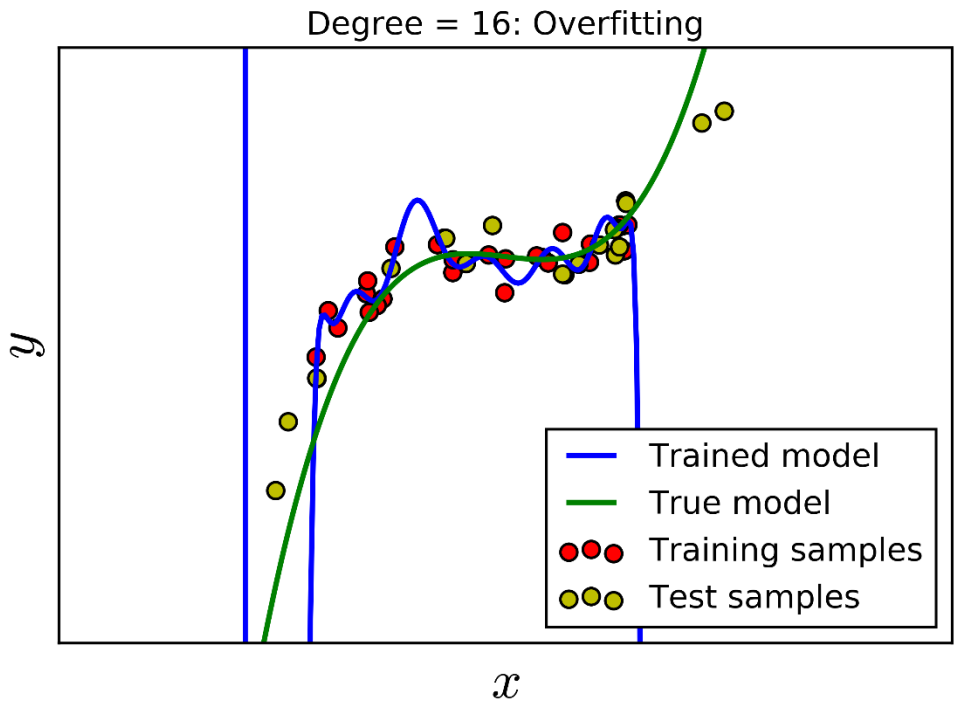
Giới thiệu

- Để có cái nhìn đầu tiên về overfitting, chúng ta cùng xem Hình dưới đây.



Giới thiệu


Degree = 8: Overfitting
y
x
Trained model
True model
Training samples
Test samples


Degree = 16: Overfitting
y
x
Trained model
True model
Training samples
Test samples

Giới thiệu

- Rõ ràng là một đa thức bậc không vượt quá 29 có thể *fit* được hoàn toàn với 30 điểm trong training data.
- Bài toán này hoàn toàn có thể được giải quyết bằng Linear Regression với dữ liệu mở rộng cho một cặp điểm (x,y) là (x,y) với $x=[1,x,x^2,x^3,\ldots,x^d]^T$ cho đa thức bậc d.
  - Điều quan trọng là chúng ta cần tìm bậc d của đa thức cần tìm.

Giới thiệu

- 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.

Giới thiệu

Đại lượng để đánh giá chất lượng của mô hình trên training data và test data.

Với Regression, đại lượng này thường được định nghĩa:

- **Train error:**

$$\text{train error} = \frac{1}{N_{\text{train}}} \sum_{\text{training set}} \|\mathbf{y} - \hat{\mathbf{y}}\|_p^2$$

- **Test error:**

$$\text{test error} = \frac{1}{N_{\text{test}}} \sum_{\text{test set}} \|\mathbf{y} - \hat{\mathbf{y}}\|_p^2$$

Giới thiệu

- 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, hiếm khi xảy ra.

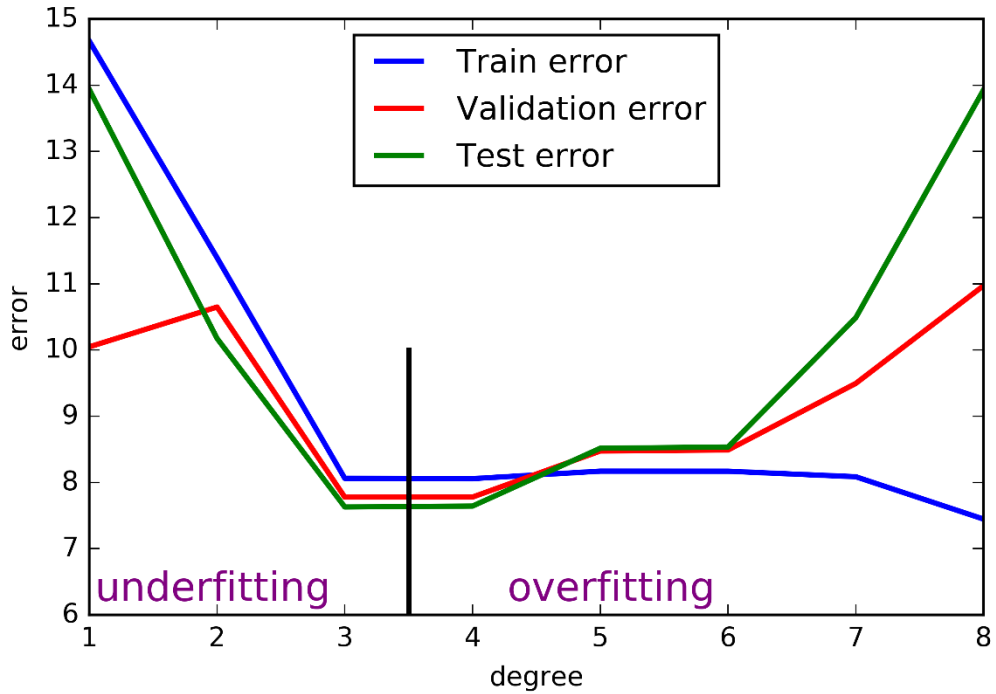
Validation

Validation

- Chúng ta thường chia tập dữ liệu ra thành hai tập nhỏ: training data và test data.
- Khi xây dựng mô hình, ta không được sử dụng test data. Vậy làm cách nào để biết được chất lượng của mô hình với unseen data (tức dữ liệu chưa nhìn thấy bao giờ)?
- Phương pháp đơn giản nhất là trích từ tập training data ra một tập con nhỏ (validation set) và thực hiện việc đánh giá mô hình trên tập con nhỏ này.
- Training set là phần còn lại của training set ban đầu.
  - Train error được tính trên training set mới này
  - validation error được tính trên tập validation.

Validation

- Ta tìm mô hình sao cho cả *train error* và *validation error* đều nhỏ, qua đó có thể dự đoán được rằng *test error* cũng nhỏ.
- Phương pháp thường được sử dụng là xét nhiều mô hình khác nhau. Mô hình nào cho *validation error* nhỏ nhất sẽ là mô hình tốt.



Validation

- 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.

Validation

Cross-validation

- Ta thường có ít 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 sẽ không đủ để xây dựng mô hình.
  - Dẫn đến tập validation phải thật nhỏ để giữ được lượng dữ liệu cho training đủ lớn.
  - Tuy nhiên, khi tập validation quá nhỏ, hiện tượng overfitting lại có thể xảy ra với tập training còn lại.

Validation

- Có giải pháp nào cho tình huống này: Câu trả lời là cross-validation:
  - chia tập training ra k tập con không có phần tử chung, có kích thước gần bằng nhau.
  - Tại mỗi lần kiểm thử, một trong số k tập con được lấy ra làm validata set. Mô hình sẽ được xây dựng dựa vào hợp của k−1 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

- Mô hình polynomial như trên chỉ có một tham số cần xác định là bậc của đa thức. Do đó, ta có thể thực hiện được.
- Các bài toán Machine Learning, lượng tham số cần xác định thường lớn, và khoảng giá trị của mỗi tham số cũng rộng. Như vậy, 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

- *Regularization*: 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).

- Cụ thể, 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.

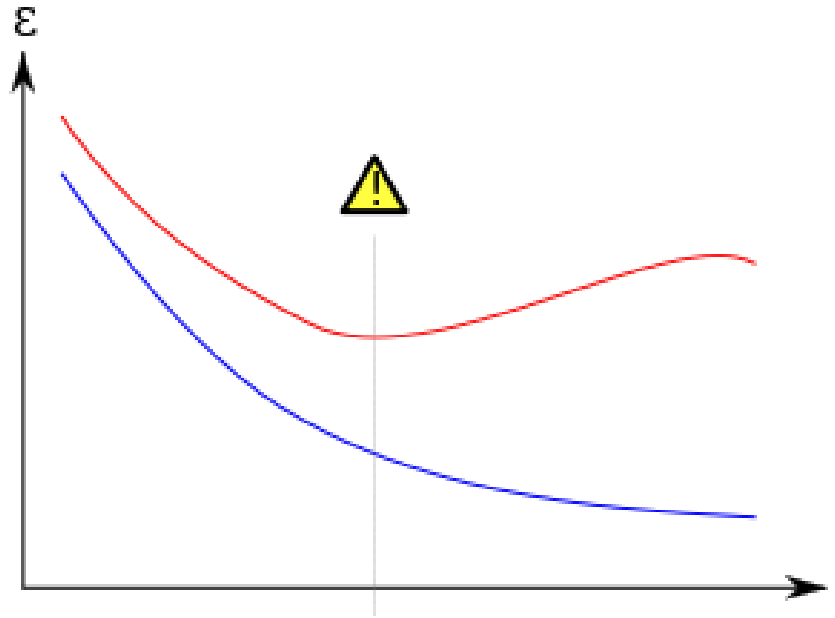
Regularization

Kỹ thuật **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 dừng thuật toán trước khi hàm mất mát đạt giá trị quá nhỏ, giúp tránh overfitting.

Regularization

- Vậy dừng khi nào: Một kỹ thuật thường được sử dụng là:
  - Tách từ training set ra một tập validation set.
  - Sau một số vòng lặp, ta tính cả *train error* và *validation error*, đến khi *validation error* có chiều hướng tăng lên thì dừng lại,
  - Quay lại sử dụng mô hình tương ứng với điểm và *validation error* đạt giá trị nhỏ.



Regularization

Thêm số hạng vào hàm mất mát

- Kỹ thuật regularization 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**:

$$J_{\text{reg}}(\theta) = J(\theta) + \lambda R(\theta)$$

Regularization

- Việc tối thiểu *regularized loss function* đồng nghĩa với việc tối thiểu cả *loss function* và số hạng *regularization*.

$$J_{\text{reg}}(\theta) = J(\theta) + \lambda R(\theta)$$

Tối thiểu Tối thiểu

- Nghiệm của bài toán tối ưu *loss function* và **regularized loss function** là khác nhau:
  - Ta mong sự khác nhau này là nhỏ, vì vậy tham số regularization (*regularizaton parameter*) λ thường được chọn là một số nhỏ để biểu thức regularization không làm giảm quá nhiều chất lượng của nghiệm.

Regularization

l_2 regularization

- Trong kỹ thuật này:

$$R(\mathbf{w}) = \|\mathbf{w}\|_2^2$$

- Trong Xác suất thống kê, Linear Regression với l_2 regularization được gọi là **Ridge Regression**. Hàm mất mát của *Ridge Regression* có dạng:

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

Regularization

Regularizers for sparsity

- Trong nhiều trường hợp, ta muốn các hệ số *thực sự* bằng 0 chứ không phải là *nhỏ gần 0* như l_2 regularization đã làm phía trên. Lúc đó, có một regularization khác được sử dụng, đó là l_0 regularization:

$$R(\mathbf{W}) = \|\mathbf{w}\|_0$$

- Norm 0 không phải là một norm thực sự mà là giả norm.

- Norm 0 của một vector là số các phần tử khác không của vector đó. Khi norm 0 nhỏ, tức rất nhiều phần tử trong vector đó bằng 0, ta nói vector đó là *sparse*.

Regularization

- Việc giải bài toán tối thiểu norm 0 nhìn chung là khó vì hàm số này không *convex*, không liên tục. Thay vào đó, norm 1 thường được sử dụng:

$$R(\mathbf{W}) = \|\mathbf{w}\|_1 = \sum_{i=0}^{d} |w_i|$$

- Norm 1 là tổng các trị tuyệt đối của tất cả các phần tử.
- Người ta đã chứng minh được rằng tối thiểu norm 1 sẽ dẫn tới nghiệm có nhiều phần tử bằng 0.
- Ngoài ra, vì norm 1 là một *norm thực sự* (proper norm) nên hàm số này là *convex*, và hiển nhiên là liên tục, việc giải bài toán này dễ hơn việc giải bài toán tối thiểu norm 0.

Regularization

- Trong Thống Kê, việc sử dụng l_1 regularization còn được gọi là LASSO (Least Absolute Shrinkage and Selection Operator).