PRML Chapter.1

Introduction

2017010698
수학과 오서영

목차

0. Prolog

패턴인식

: 데이터에 내재하는 패턴을 자동으로 추출하는 것
-> 패턴들을 이용해 **Classification, Regression)**
등의 구체적인 작업을 수행

- 간단한 Heuristic 방법론이나 rule 기반 방식들이
더 잘 동작할 것이라 생각하지만, 사실은 예외 사항만 끊임없이
만들어낼 뿐 보통 결과는 더 좋지 않다.

-> 학습 데이터를 이용하여 선택한 모델의 파라미터를
튜닝하는 **기계학습** 방식이 더 좋은 효과를 낸다.

0. Prolog

Generalization 의 문제

: 우리가 관찰한 데이터 : **출현 가능한** 데이터의 **일부**

-> 일부 데이터로부터 전체 데이터를 나타내는 패턴을 발견해야 한다.

-> 패턴 인식의 목표

- **Pre-process or Feature extraction**

: 문제 범위를 축소하기 위해 입력 데이터를 사전에 정제하는 작업

기계 학습의 종류

- **Supervised Learning**

Classification, Regression

- **Unsupervised Learning**

Clustering, Visualization

- **Reinforcement Learning**

보상(reward)을 최대로 만드는 액션을 찾는 문제

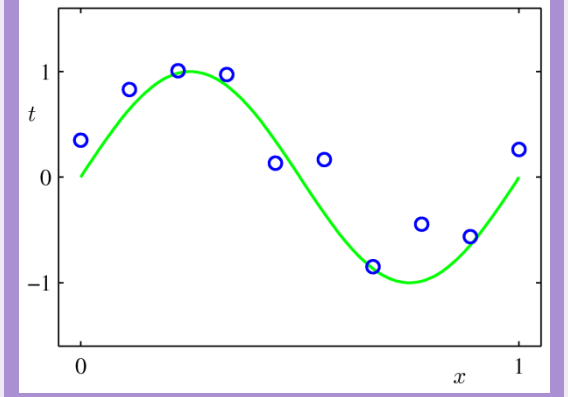
-> Supervised Learning 과 다르게 Target 값 X.

1. Example : Polynomial Curve Fitting

목표

: 입력 변수 x 를 바탕으로 타겟(target) t 의 값을 예측
- 예제로 사용할 모델을 가정하는 것으로부터 시작

- target t 가 함수 sin(2πx) 로부터 발현되는 실수 값이라고 가정 -> 발현된 값은 noise를 포함
-noise는 주어진 문제가 본질적으로 확률에 의한 프로세스이기 때문이거나, 관찰되지 못한 다른 원인으로 관측 값에 포함됨



녹색 선 : sin(2πx)
파란점 : 가우시안 랜덤 분포에 의해 샘플

1. Example : Polynomial Curve Fitting

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

1. 다항식(polynomial) 모델 선택

모델링을 해야 할 함수를 아직 알지 못하는 함수라고 가정하면
이를 근사할 수 있는 근사 식이 필요하다.
일반적으로 테일러 급수나 퓨리에 변환 식 등으로
함수 근사를 많이 한다.
-> 특정 위치에서의 함수를 근사하기 위한 수단으로
테일러 급수를 많이 사용

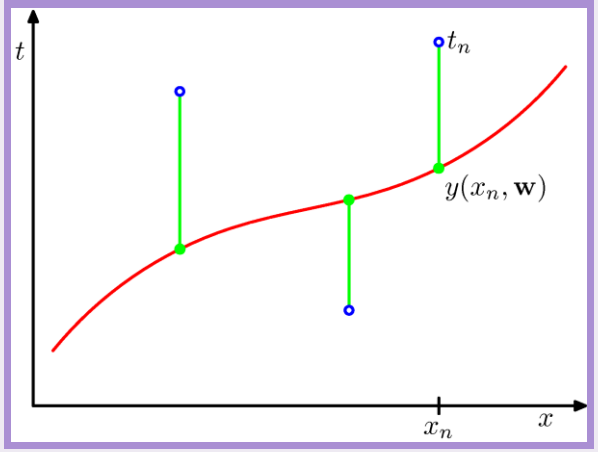
2. 근사하기 위해 최적의 계수 w 를 구하기

-> **Error function**를 도입
-> 모델로부터 도출된 y(x,w) 함수와 실제 타겟 값 t 의 차이를
최소로 하는 방식을 통해 w 값을 결정

1. Example : Polynomial Curve Fitting

보통 제곱합(sum-of-squares) Error function 사용
-> 변위 **제곱의 합**을 사용하는 이유?
-> y 함수가 convex 를 만족하는 경우, 에러 함수도 convex 를 만족하는 함수가 되고 미분 가능하게 되어, 에러 함수를 w 에 대해 단순 미분하면 최소화 문제에서 유일한 해를 가지게 된다.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$



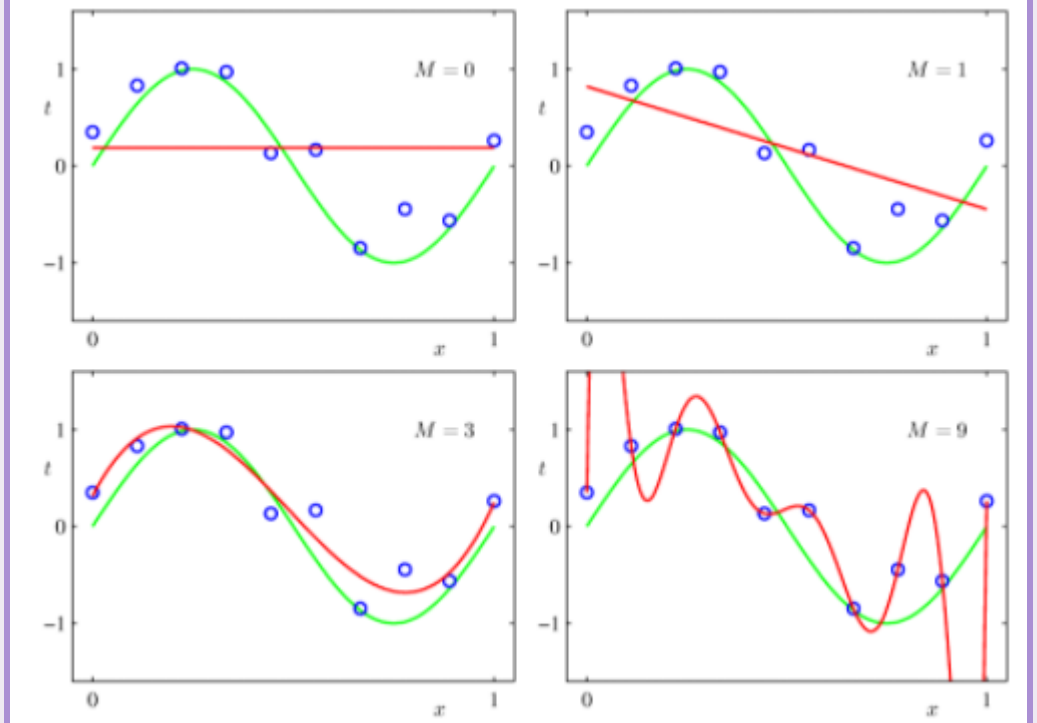
E(w) 값을 **최소**로 만드는
w 를 구하는 문제
-> Error function가 w 에 대해
이차형식(quadratic)의 함수이므로
이를 최소화 하는 값은
유일 해를 가지게 됨을 보장

1. Example : Polynomial Curve Fitting

3. 다항 함수 (polynomial)의 x 의 차수 선택

-> **model comparison or model selection**

M = 1 : 단순한 직선 회귀식 -> **Underfitting**
M = 9 : 노이즈에 대한 학습 결과도 모델 자체에 포함
-> 주어진 데이터에 과도하게 적합된 결과 -> **Overfitting**

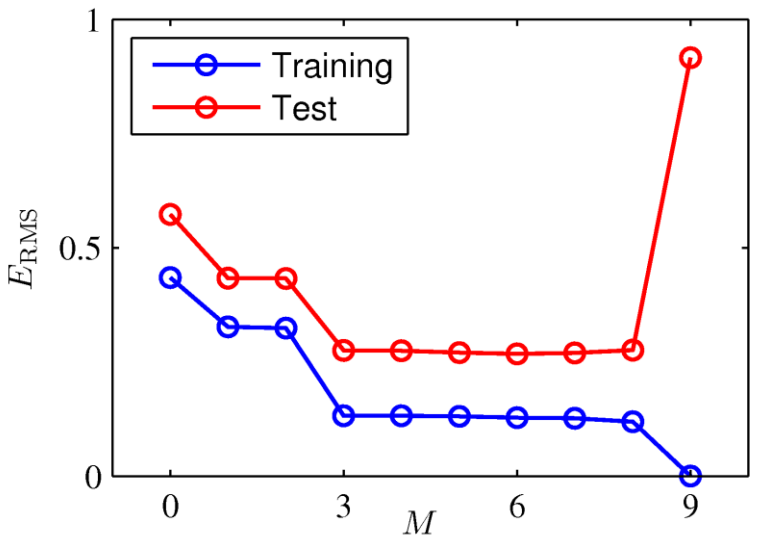


1. Example : Polynomial Curve Fitting

4. 각 모델에 대한 평가
- **RMS** 에러

$$E_{RMS} = \sqrt{\frac{2E(\mathbf{w}^*)}{N}}$$

RMS 값이 작을수록 더 적합한 모델
N : **정규화**(normalization)



M = 9인 값부터 테스트 데이터의 Error 값이 급격하게 커짐
-> Overfitting 발생 지점

1. Example : Polynomial Curve Fitting

5. Intuition

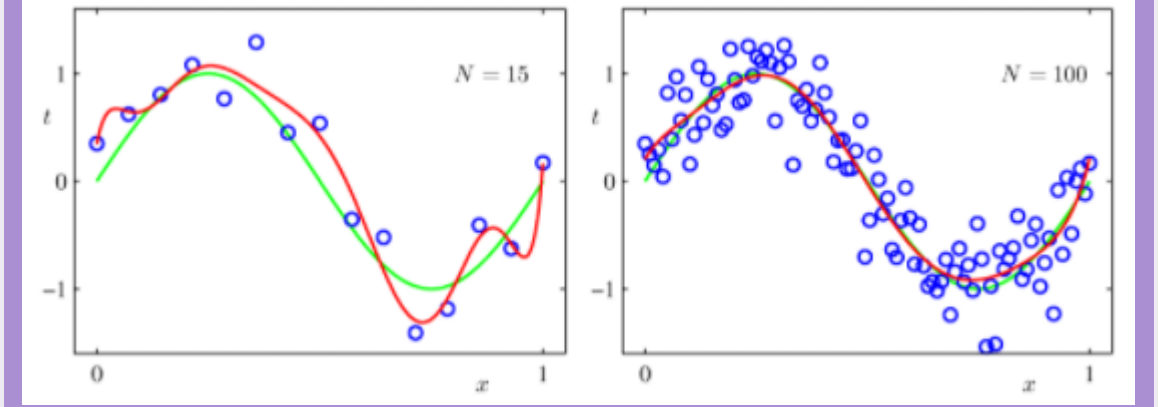
- M = 9 인 상태에서 각각의 w 값들이 매우 크거나 작다
-> 최대한 발현된 데이터 주변으로 모델 함수를
최대한 가깝게 이동시키기 위해
각각의 w 값을 극단적으로 보정한 것
-> Overfitting될 때의 모수 값들은 서로 편차가 크다

1. Example : Polynomial Curve Fitting

6. 데이터의 크기에 따른 모델의 결과 (M=9 고정)

- 적은 수의 데이터를 사용하는 경우 M=9 에서 **Overfitting**
- 많은 데이터를 사용하는 경우 M=9 임에도 **Overfitting X**

-> 데이터의 크기가 클수록 오버피팅 현상을 막을 수 있다.



1. Example : Polynomial Curve Fitting

7. 적은 데이터로 높은 차수의 파라미터를 가진 모델을 만들방법?

: 모델의 복잡도는 올리고 오버피팅을 줄이는 방법

-> **정칙화(regularization)**

- w 가 취할 수 있는 값의 범위를 제한

-> 큰 범위의 값을 가지는 w 가 등장하지 못하도록 막는 것

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

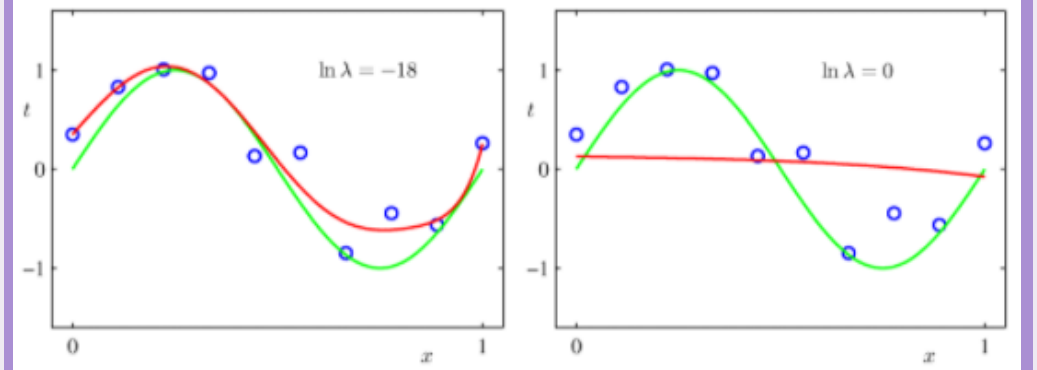
$$\|\mathbf{w}\|^2 \equiv \mathbf{w}^T\mathbf{w} \equiv w_0^2 + w_1^2 + \ldots + w_M^2$$

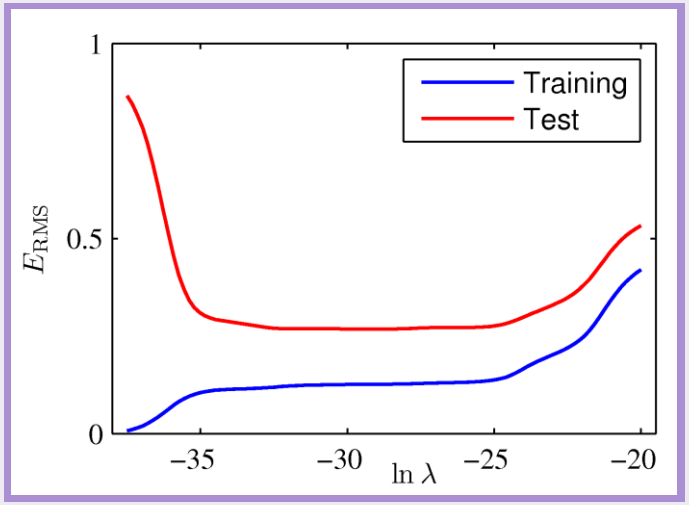
λ : regularization coefficient

-> 정칙 기능이 추가되었더라도 식 자체는 w 에 대해 닫힌 구조(closed form)

1. Example : Polynomial Curve Fitting

- lnλ = −18 : Overfitting이 없어지고 원래 근사하고자 했던 식 sin(2πx) 와 매우 근접한 식
- lnλ = 0 : 지나치게 큰 λ 값은 -> Underfitting





RMS 결과

: 학습 데이터와
테스트 데이터 모두에서
좋은 성능