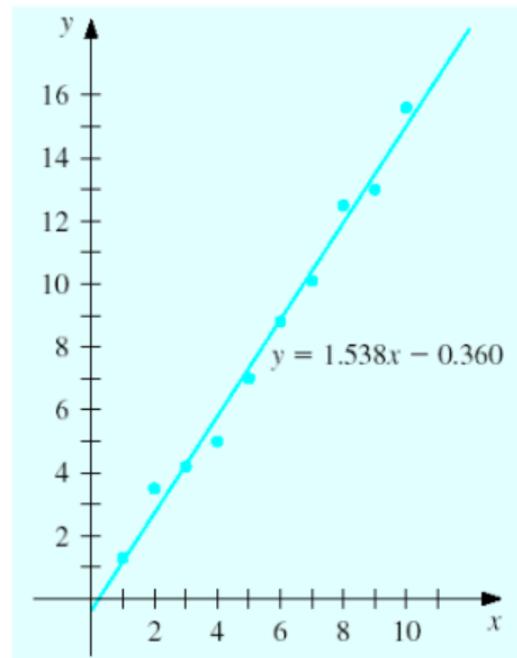
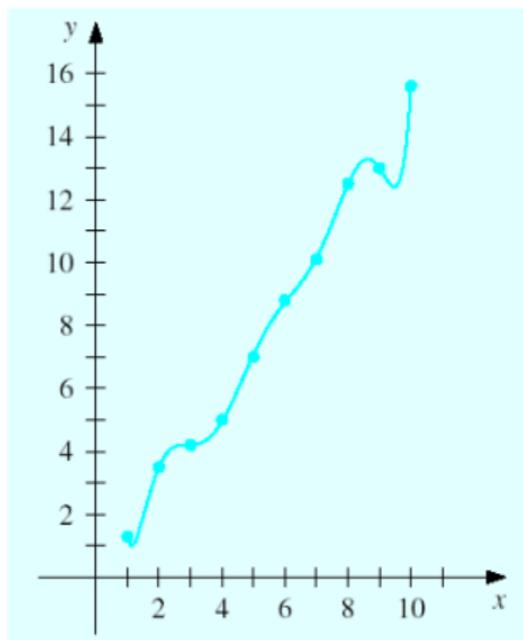


[수치해석] Data fitting

Appoximation fittint에 해당하는 부분이다.

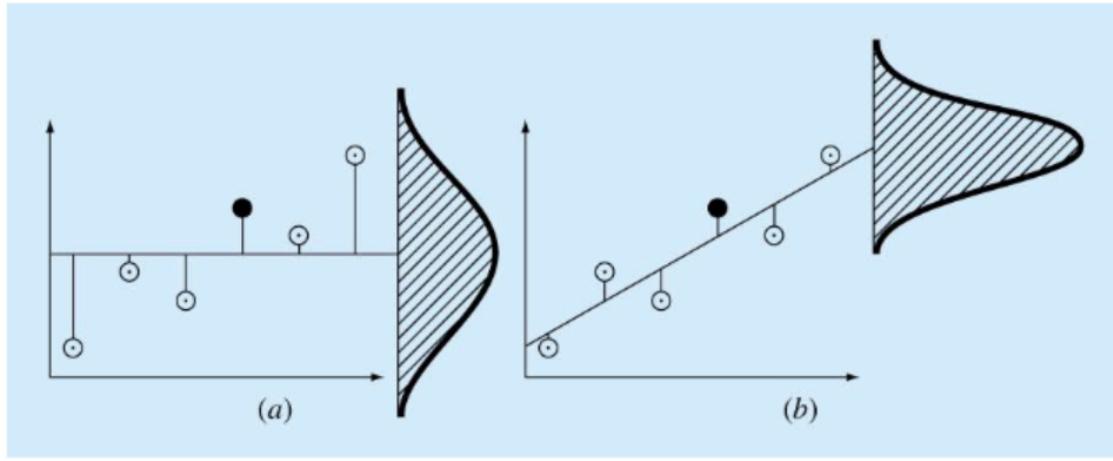
- 어느 정도의 에러를 허용한다.
- 실제 데이터는 Noise가 있기 때문에 데이터를 정확히 맞추는 것이 더 안 좋을 수 있다.
 - Exact fitting처럼 모든 데이터를 통과하면 Noise까지 학습하여 Overfitting이 발생한다.
- 어떤 Noise 모델을 가정하는 지에 따라서 결과가 달라진다.
- Generalization 측면에서 좋다.

Interpolation과 **Appoximation fitting**을 비교하면 다음과 같다.



Regression에서 모델의 구조에 따른 **Error**를 분석해보자.

- **Regression:** 입력 x 가 주어졌을 때 연속적인 출력 y 를 예측하는 방법



Zero-order model

1st-order model

Zero-order model은 $y = c$ (c 는 상수)인 모델이다.

- Residual의 분포가 넓고 낮은 것을 확인할 수 있다.

1st-order model은 $y = ax + b$ 인 모델이다.

- Residual의 분포가 좁고 높은 것을 확인할 수 있다.
 - 비교적 간단해진다.
- 분포가 대칭적/정규분포 형태로 가까워진다.

즉, 모델이 복잡해질수록 Residual이 단순해지는 것을 확인할 수 있다.

- Noise는 실제 데이터에 포함되어져 있다.
 - $y = f(x) + \epsilon$
 - $\epsilon = y - f(x)$
- 하지만 실제 데이터의 Noise를 모르고 대응되는 모델 $f(x)$ 도 모르기 때문에 Residual만 계산한다.
 - **Residual** = $y - \hat{y} = (f(x) + \epsilon) - \hat{f}(x) = y - \hat{f}(x)$
- 모델이 복잡해질수록 $\hat{f}(x)$ 와 $f(x)$ 가 비슷해져 Residual이 실제 Noise와 비슷해지고 Noise가 단순해 보이게 된다.

- 따라서 **Residual**을 **Noise**의 **Modeling**이라고 할 수 있다.
- 실제 **Noise**를 모르기 때문에 **Residual**을 **Noise**의 근사치로 활용한다.
 - Regression model이 충분히 좋다면, **Noise** \sim **Residual**이기 때문이다.
- 따라서 모델에 따라 **Residual**이 달라져 보인다.
 - 사용자가 선택한 모델 구조에 따라 Residual은 자동으로 결정된다.
 - 우리는 Noise를 직접 관측하지 못하여 Residual을 이용하여 근사하는데, **Residual**이 모델에 의해 변화하기 때문이다.

Noise model: 데이터에 섞여 있는 Noise가 어떤 확률 분포를 따르는지에 대한 모형

- 실제 **Noise**가 이런 분포를 가질 것이라고 가정
- **Residual**의 분포와 가정한 **Noise model**이 다르다면 모델 성능이 좋지 않다.

결국 **Residual**을 최소화하는 **Parameter**를 찾아 **Model**을 구성하는 것이 **Approximation fitting**의 목표이다.

이때, 가정한 **Noise model**에 따라 그에 맞는 형태의 **Loss**를 사용한다.

- $y = f(x) + \epsilon$ 에서 어떤 Noise model을 정의 하는지에 따라 확률 $P(Y|X)$ 의 분포 형태가 달라진다.
 - 평균이 $f(x)$ 인 것만 동일하다.
- Loss를 최소화한다는 것은 Maximum likelihood를 하는 것과 동일하고 NLL을 사용할 수 있다.
 - Loss의 형태는 확률 분포의 영향을 받을 수 밖에 없다.
- 이때 $P(Y|X)$ 형태가 다르기 때문에 Noise model이 결과를 결정할 수 있는 것이다.

그 Loss를 최소화하면 **Residual**이 가정한 **Noise model**과 최대한 가까워지도록 모델이 학습된다.

Data fitting 방법들에 대해 알아보자.

Least-Square Data fitting

| Loss function으로 오차의 제곱을 사용하는 방법

Data: $\sum_{i=1}^N (X_i, Y_i)$, Parameter 개수: M , Parameter = $[a_1, a_2, \dots, a_N]$

Least square: $S = \sum_{i=1}^N [(Y_i - y(X_i; a))]^2$ 을 최소화하는 Parameter를 찾는 것

- y 가 모델
- Polynomial, $y = ax + b$ 등 다양한 모델을 사용할 수 있다.

Error를 $e_i = Y_i - y(X_i; a)$ 라고 할 때, 모든 데이터가 서로 독립이고 **Noise (error)**가 Gaussian이라면 Least square는 Maximum Likelihood와 동일하다.

- Residual (Noise)가 Gaussian이라면 $p(y_i|x_i) = N(y(x_i; a), \sigma)$ 이다.
- i.i.d 가정으로 인해 Likelihood $P(Y|X) = \prod_{i=1}^N (P(Y_i|X_i))$ 이다.
- Maximum likelihood를 위해 Negative log likelihood를 하면 Least square의 식과 동일해진다.

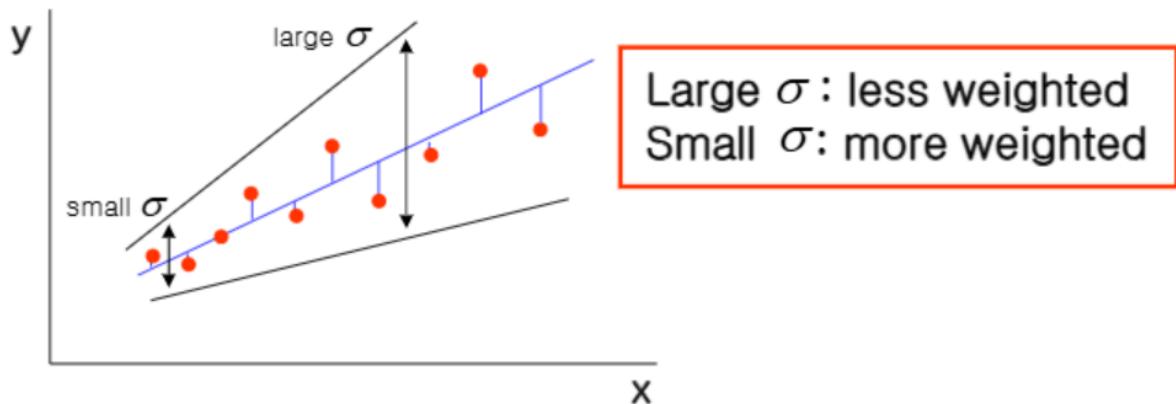
Chi-Square Data fitting

| 각 Data point의 불확실성을 반영하는 Least square

$$S = \sum_{i=1}^N \left[\frac{Y_i - y(X_i; a)}{\sigma_i} \right]^2$$

- 각 Data의 불확실성이 σ 를 통해 표현된다.

Called the “chi-square”



- 불확실성 (σ)가 큰 데이터에 가중치를 덜 주는 방식이다.
- 확실한 Data로 Training하고자 하는 것이다.

Generalization of Linear Least Square

Model: $y = \sum_{i=1}^M c_i f_i(x_i)$

- c_i : 계수, f_i : Basis function
- c 에 대해 Linear해야 한다.

Loss는 다음과 같다.

$$S = \sum_{i=1}^N [Y_i - \sum_{i=1}^M c_i f_i(x_i)]^2$$

이를 미분하여 c 를 구하도록 하면 $J^T J c = J^T y$ 가 나온다.

$$\mathbf{J}^T = \begin{bmatrix} \frac{\partial e_1}{\partial c_1} & \frac{\partial e_1}{\partial c_1} & \cdots & \frac{\partial e_N}{\partial c_1} \\ \frac{\partial e_1}{\partial c_2} & \ddots & & \frac{\partial e_N}{\partial c_2} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial e_1}{\partial c_M} & \frac{\partial e_1}{\partial c_M} & \cdots & \frac{\partial e_N}{\partial c_M} \end{bmatrix} = \begin{bmatrix} f_1(x_1) & f_1(x_2) & \cdots & f_1(x_N) \\ f_2(x_1) & \ddots & & f_2(x_N) \\ \vdots & & \ddots & \vdots \\ f_M(x_1) & f_M(x_2) & \cdots & f_M(x_N) \end{bmatrix}$$

$J^T J c = J^T y$ 식으로 C를 찾는 것은 **1D Least square**뿐만 아니라, **Multi-Dimension Least square**에서도 동일하다.

- f_i 의 입력이 스칼라인지 벡터인지 차이밖에 없다.

즉, 모델이 파라미터에 대해 선형인 경우, **Linear Equation**으로 파라미터를 구할 수 있다.

그렇다면, **Parameter**에 대해 **Non-linear**한 모델은 어떻게 다를 수 있을까?

$$y(\mathbf{x}) = \underbrace{f(\mathbf{x}, \mathbf{a})}_{\text{nonlinear fcn. of } \mathbf{a}} \quad \xrightarrow{\text{parameters}}$$

$$y = f(x_1, x_2, a_1, a_2, a_3, a_4)$$

$$= x_1 + a_3 x_2 - \frac{a_2}{a_4} x_1^2 + \frac{a_1}{a_4} x_1 x_2 - a_2 x_4$$

Non-linear한 경우에는 Linear equation을 통해 파라미터를 구할 수 없다.

1. Easy case

쉽게 Linear한 형태로 변환하여 Least square에서 얻은 Linear equation을 통해 파라미터를 구할 수 있는 경우

$y = \alpha e^{\beta x}$ 라는 Non-linear model을 생각해보자.

양변에 로그를 취하면 다음과 같다.

- $\ln y = \ln \alpha + \beta x$
- **Linearization**

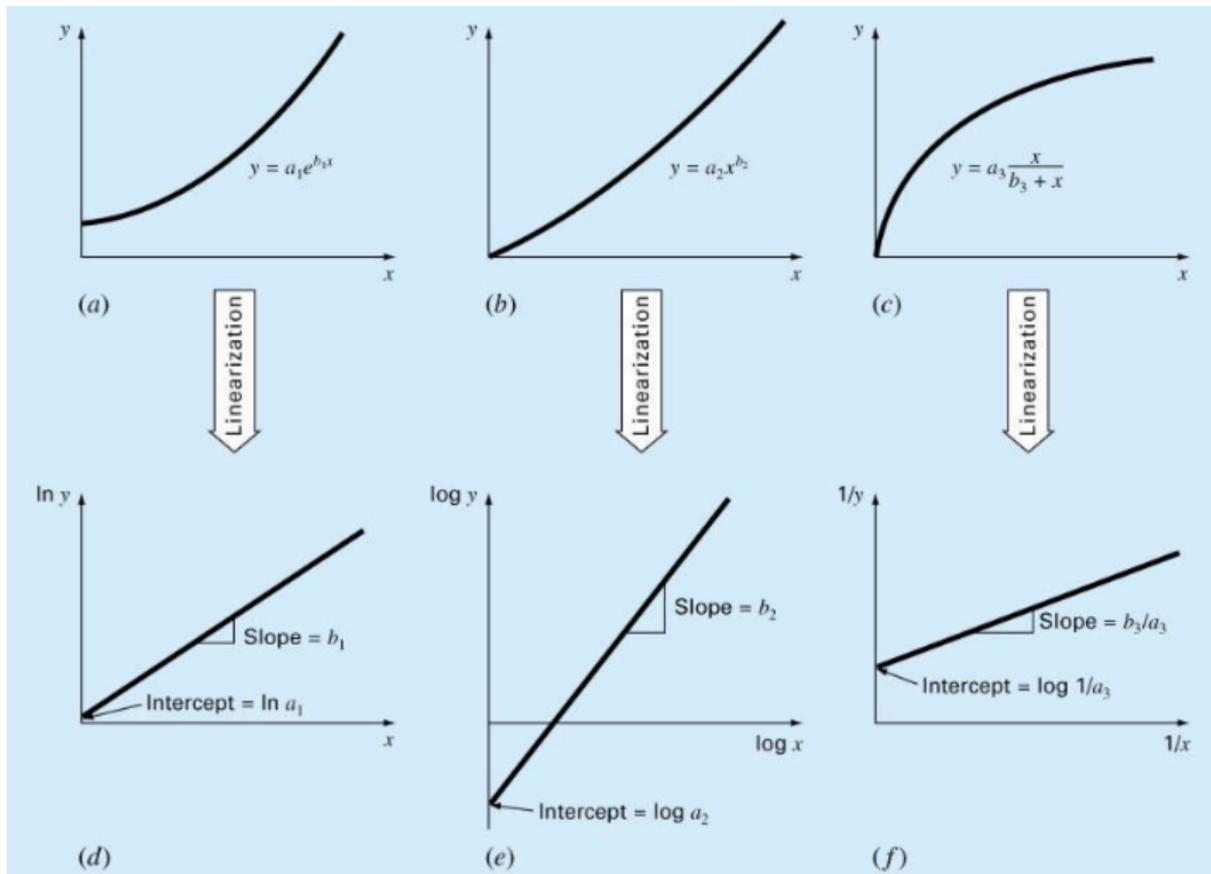
여기서 $y' = \ln y, \alpha' = \ln \alpha$ 이라고 하면 다음과 같은 Linear model이 된다.

- $y' = \alpha' + \beta x$

Linear한 모델을 통해 얻은 Linear equation으로 Parameter를 구할 수 있다.

그러나 이 방법은 Log를 취함으로써 Noise model 구조 자체를 변화시키기 때문에, Optimum Solution을 찾지 못 한다.

여러 Linearization 방법이 있다.



2. Hard case

Model이 $y = f(x, a)$ 일 때, 아래 **Cost function**이 최소가 되도록 하는 Parameter a 를 **Iterative method**로 찾아야 한다.

$$x^2(a) = \sum_{i=1}^N \left[\frac{y_i - f(x_i, a)}{\sigma_i} \right]^2$$

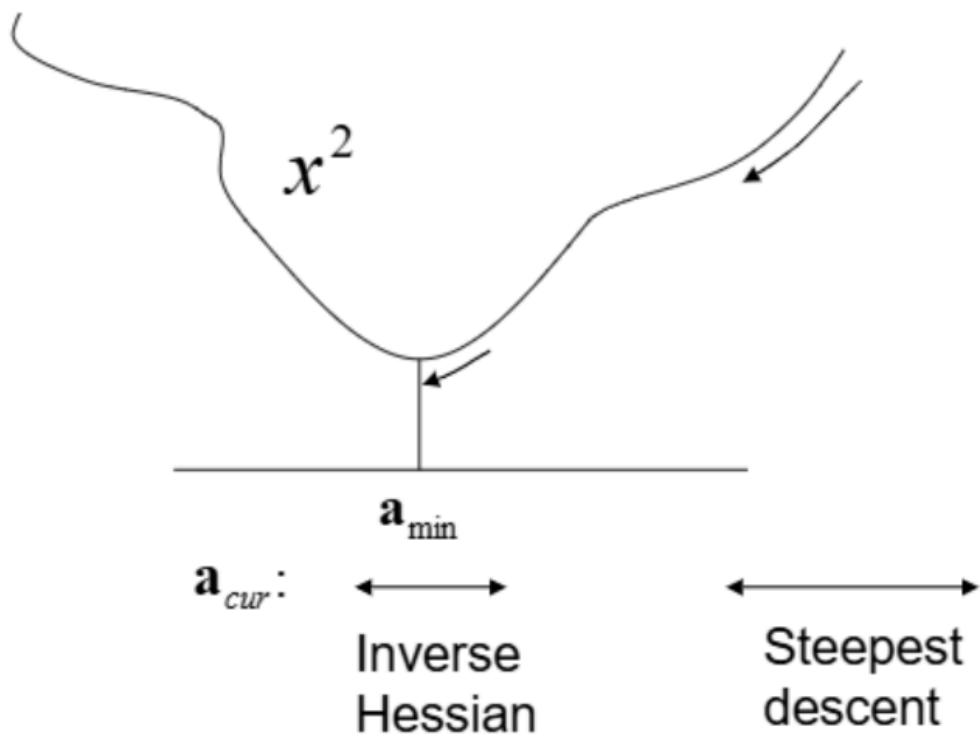
a^* 을 찾아야 한다.

$$a^* = \operatorname{argmin}_a x^2(a)$$

Parameter a 를 찾는 **Iterative method**에 대해 알아보자.

Levenberg–Marquardt Method

최적점에서 멀 때는 Steepest descent를 이용하여 가까울 때는 Inverse Hessian을 이용하는 방법



- a_{cur} : 현재 a 의 위치

Steepest descent

$$\Delta a = -const \cdot \nabla x^2$$

Inverse Hessian

cost function $x^2(a)$ 를 Taylor 전개로 근사하면 다음과 같다.

$$x^2(a) \approx x^2(a_{cur}) + \nabla x^2(a_{cur})^T \Delta a + \frac{1}{2} \Delta a^T H \Delta a$$

최솟값에서의 기울기는 0이기 때문에, $\nabla x^2(a_{min}) = 0$ 이다.

$$\nabla x^2(a_{min}) = \nabla x^2(a_{cur}) + H \Delta a = 0$$

위 식을 정리하면 아래와 같다.

$$H \Delta a = -\nabla x^2(a_{cur})$$

따라서 $\Delta a = H^{-1} \nabla x^2(a_{cur})$ 이고, Δa 만큼 Update하는 방법이다.

과정

1. 초기값 (a_{cur})을 정한다.

2. Cost function ($x^2(a_{cur})$)과 Gradient를 계산한다.

3. λ 를 정한다.

4. $(H + \lambda I) \Delta a = -\nabla x^2(a_{cur})$ 의 Linear equation을 계산하여 Δa 를 얻는다.

- λ 가 클수록 Steepest descent처럼 동작
- λ 가 작을수록 Inverse Hessian method처럼 동작

5. (4)번에서 Update 결과 $x^2(a_{cur} + \Delta a) > x^2(a_{cur})$ 라면 λ 를 10배 증가하여 더 Steepest descent처럼 동작하도록 만든다. $x^2(a_{cur} + \Delta a) < x^2(a_{cur})$ 라면 λ 를 10배 감소하여 더 Inverse Hessian method처럼 동작하도록 만든다.

(4)번에서 λ 가 매우 커지면 **Diagonal dominant**해지고 $H + \lambda I \sim \lambda I$ 가 된다.

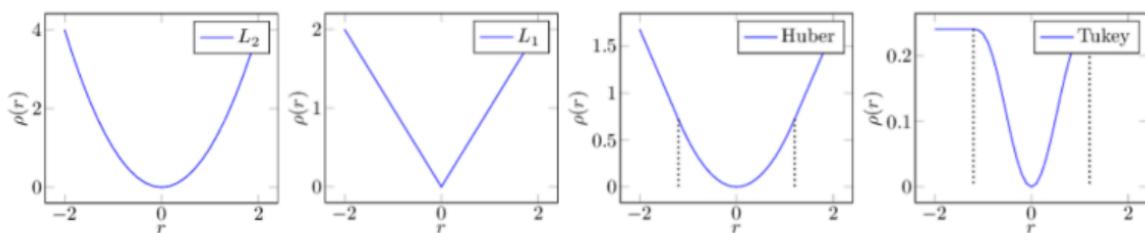
이때, $\Delta a = -\lambda^{-1} \nabla x^2(a_{cur})$ 가 되는데, λ^{-1} 을 **Learning rate**라고 생각하면 **Steepest descent**와 동일하다는 것을 확인할 수 있다.

Robust data fitting

| Least square가 Outlier에 약하기 때문에 사용하는 방법

Least square는 에러의 제곱을 모두 합하기 때문에 Outlier가 있다면 Loss가 커지게 된다.

1. Robust Measure (Loss function) 사용

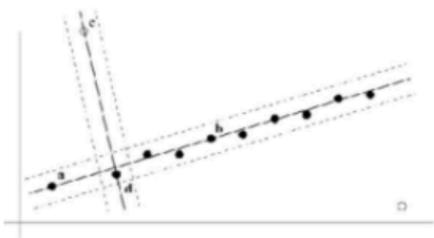


L2 (가장 왼쪽에 비해) 나머지 3가지 Loss는 Robust하다.

r : Residual일 때, 각 Loss의 계산 방법을 알아보자.

1. **L2:** r^2
2. **L1:** $|r|$
3. **Huber:** 작은 r 에는 L2, 큰 r 에는 L2를 적용
4. **Tukey:** $\rho(r) = flat \text{ when } |r| > c$
 - Outlier의 loss는 상수

2. Random sampling 사용



RANdom SAmple Consensus:

RANSAC determines the consistency of a hypothesis by counting the number of points within a threshold. RANSAC determines the consistency of a hypothesis by counting the number of points within a threshold distance (given by the dashed line).

특정 모델을 정하고, 각 데이터와 Model이 예측한 값의 Residual이 임계값 이하면 Inlier, 초과하면 Outlier라고 판단한다.

Inlier가 많을수록 좋은 모델이라고 판단하는 방법이다.