사물인터넷 (Internet of Things)

김태운

목차

- Sensing Layer: 센서 및 센싱/분석기술
  - 데이터 정제

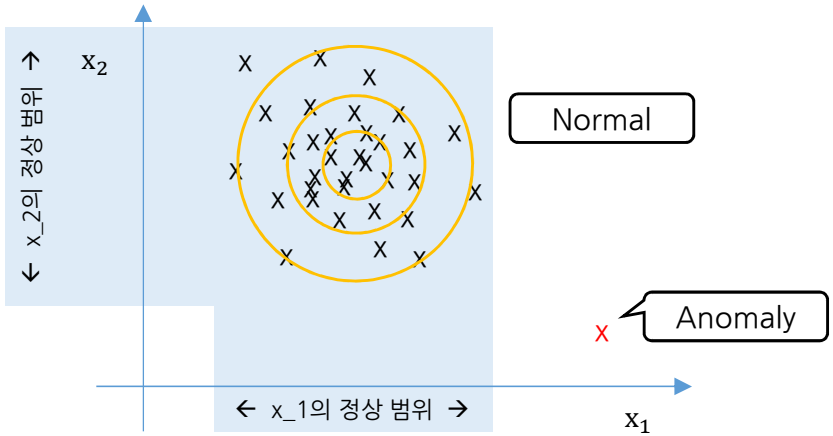
Sensing Layer: 데이터 정제

Sensing Layer: 데이터 정제

지난 수업 내용

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)

    - 이상 탐지 알고리즘 : 노란색 원의 중심에 가까울 수록 정상으로, 멀어질 수록 비정상으로 판단



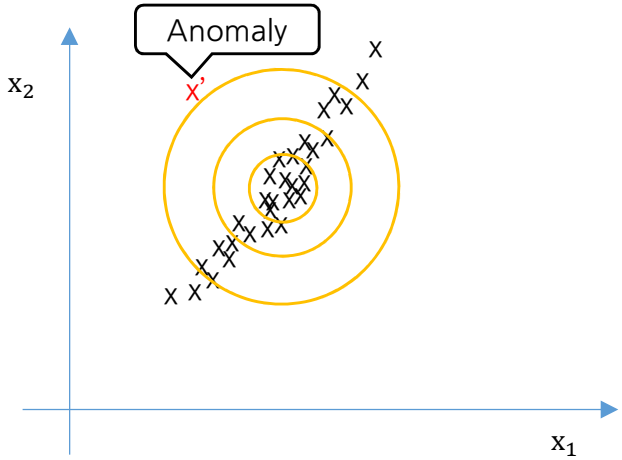
Uni-Variate Anomaly Detection 알고리즘
- x_1 feature 만을 고려하여 모델 적합도 평가
- x_2 feature 만을 고려하여 모델 적합도 평가
- 위의 두 결과를 곱하여 최종 적합도 평가

Sensing Layer: 데이터 정제

지난 수업 내용

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)

    - 그런데,



- x' 샘플의 경우,
- x_1 만을 고려했을 때, 정상 범위에서 크게 벗어나지 않음
- x_2 만을 고려했을 때, 정상 범위에서 크게 벗어나지 않음
- 하지만, 전체적인 데이터의 분포를 고려하면(즉, x_1과 x_2간의 상관관계를 고려하면…), 정상 범위에서 벗어남

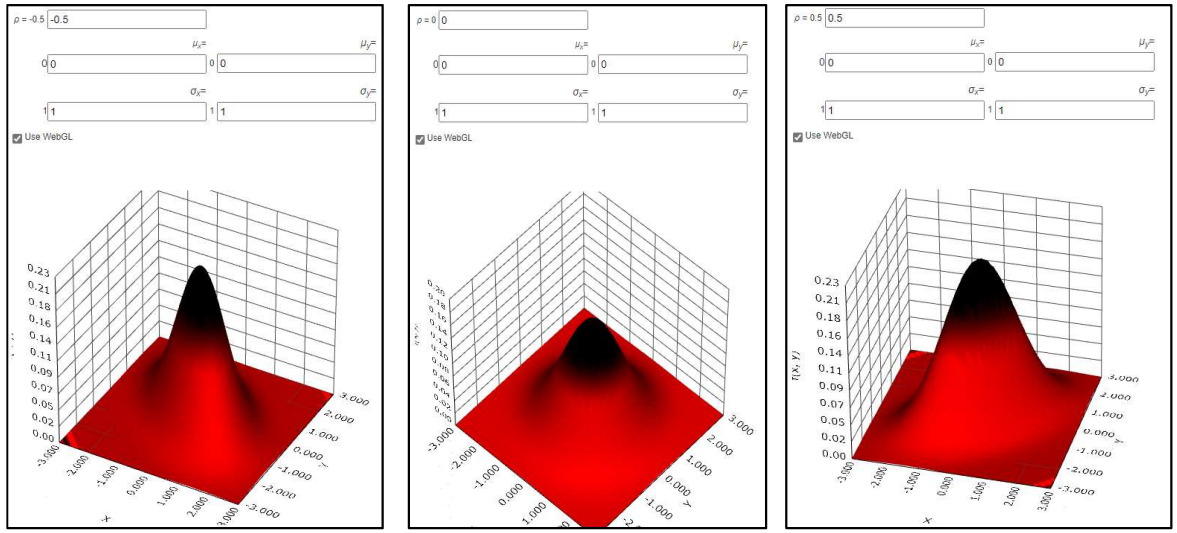
    - 각 데이터 샘플에 다수의 feature 가 포함된 상황에서, feature 들 간의 상관관계까지 동시에 고려할 수 있는 방법은?

Multi-Variate Gaussian Distribution을 모델로 사용하여 이상 탐지를 수행!

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - 각 feature 에 대한 Gaussian 모델을 따로 만드는 것이 아니라, 모든 feature 를 동시에 반영하는 multi-variate Gaussian 모델을 사용하면 feature 값 간의 상관관계를 반영할 수 있음
    - Multi-variate Gaussian 모델을 정의하는 두 파라미터
      - 평균: $\mu \in \mathbb{R}^n$
      - 공분산 행렬(covariance matrix): $\Sigma \in \mathbb{R}^{n \times n}$
    - 예: feature 가 2개(x, y) 일 때 (bi-variate Gaussian distribution)
      - $\mu = [\mu_x, \mu_y]$
      - $\Sigma = \begin{bmatrix} \sigma_x & \rho \\ \rho & \sigma_y \end{bmatrix}$, 공분산 행렬은 전치(transpose) 결과가 동일함
      - μ_x, μ_y는 x, y축의 평균을 결정하고, σ_x, σ_y는 x, y 축으로 퍼진 정도를 결정하며, ρ 는 x, y간 상관관계를 결정함

Feature 값들 간의 상관관계를 반영함



Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - 샘플의 수 m, feature 의 수 n 일 때, Gaussian 모델을 정의하는 파라미터를 학습하는 방법

    - 각 feature 를 별도로 고려하는 uni-variate Gaussian 모델의 파라미터 학습 방법 // model fitting
      - 각 x에 대해서 feature j 별로 μ_j, σ_j 계산
      - $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$ 를 $j = 1, \dots, n$에 대해서 반복해서 각 feature에 대한 mean을 계산
      - $\sigma_j = \frac{1}{m}\sum_{i=1}^{m} \left(x_j^{(i)} - \mu_j\right)^2$ 를 $j = 1, \dots, n$에 대해서 반복해서 각 featur에 대한 std_dev을 계산
        - 정확하게는 m-1로 나눠야 unbiased estimation 이지만, m이 충분히 크다는 가정하에 m-1 대신 m을 사용

    - Cross-Validation sample을 사용한 임계치 계산 (threshold, ϵ)

    - 새로 획득한 샘플에 대해서 모델 적합도 p(x)계산: uni-variable Gaussian distribution 의 pdf (확률밀도함수)
      - $p(x) = \prod_{j=1}^{n} p\left(x_j; \mu_j, \sigma_j^2\right) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}\right)$
      - 새로 획득한 샘플에 대해서
      - IF $p(x) < \epsilon$: 비정상(abnormal)
      - ELSE : 정상(normal)
      - 로 판단

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - 샘플의 수 m, feature 의 수 n 일 때, Gaussian 모델을 정의하는 파라미터를 학습하는 방법

    - Multi-variate Gaussian 모델의 파라미터 학습 방법
      - 모든 feature 를 동시에 반영하여 $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ 를 계산
      - $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$
      - $\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$

    - Cross-Validation sample을 사용한 임계치 계산 (threshold, ϵ)

    - 새로 획득한 샘플에 대해서 모델 적합도 p(x)계산: Multi-variable Gaussian distribution 의 pdf (확률밀도함수)
      - $p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$
      - 새로 획득한 샘플 x에 대해서:
      - IF $p(x = x';\mu,\Sigma) < \epsilon$: 비정상(abnormal)
      - ELSE : 정상(normal)
      - 라고 판단

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - 각 feature 를 별도로 고려하는 Gaussian 모델(Uni-Variate Gaussian)을 사용하는 기법과 Multi-variate Gaussian 모델을 사용한 기법 비교

Uni-Variate Gaussian 모델	Multi-Variate Gaussian 모델
계산 복잡도가 낮음	(역행렬을 계산해야 하므로) 계산 복잡도가 높음
각 feature 들 간의 상관관계를 반영하지 못함	각 feature 들 간의 상관관계를 반영할 수 있음
m(샘플의 수)이 작은 상황에서도 동작	$m > n$ 을 만족하는 상황에서만 동작함(즉, 샘플이 충분한 상황). 그렇지 않으면 Σ 행렬의 역행렬을 계산할 수 없음

Sensing Layer: 데이터 정제

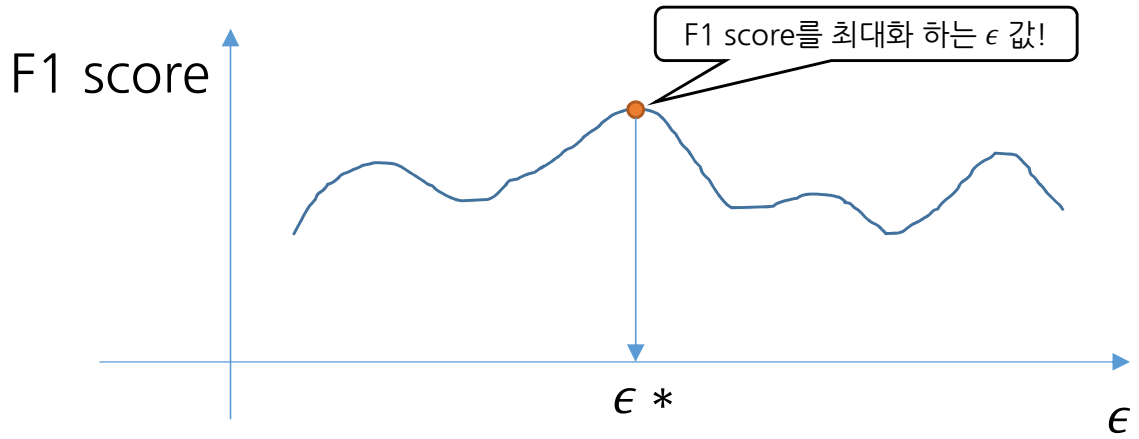
- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - 정상/비정상 판단을 위한 임계치 설정
      - Uni/Multi-Variate Gaussian 모델을 사용하는 경우, 샘플 x에 대해서 아래의 기준을 통해 정상/비정상 여부를 판단함

        IF $p(x = x'; \mu, \Sigma) < \epsilon$: 비정상(abnormal)

        ELSE : 정상(normal)

      - 이 때, ϵ 을 설정하는 방법?

        F1 score를 최대화 하는 ϵ 찾기



Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - F1 score
      - 기계학습의 분류 성능을 평가하는 지표로 사용
      - Precision과 Recall의 조화평균으로 정의하고, 주로 분류 클래스 간 데이터가 심각한 불균형을 이루는 경우에 사용
      - 예: normal 인 샘플의 수에 비해 abnormal인 샘플 수가 현저하게 적은 경우
      - F1 score는 기존 F score에서 beta 값이 1로 설정된 경우에 해당함(즉, precision과 recall에 동일한 가중치를 두는 경우)
      - F score:

$$F = \frac{(\beta^2+1)*Precision*Recall}{\beta^2*Precision+Recall}$$

      - F1 score:

$$F_1 = 2 * \frac{Precision*Recall}{Precision+Recall}$$

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - F1 score
      - 일반적 분류기에 대한 confusion matrix는 아래와 같이 정의함

	실제로 Positive	실제로 Negative
Positive라고 예측	True Positive (TP)	False Positive (FP)
Negative라고 예측	False Negative (FN)	True Negative (TN)

      - Anomaly Detection의 경우, 이상/abnormal 경우를 Positive로, 정상/normal을 negative로 함

    - Accuracy (정확도)
      - 모든 분류 결과 중에서 TP 및 TN 의 비율, $\frac{TP+TN}{TP+FP+TN+FN}$
      - 즉, 모든 샘플 데이터에 대해 positive 및 negative를 정확하게 분류한 비율

    - Precision (정밀도)
      - 분류기가 Positive로 분류한 결과 중에서 실제 Positive의 비율, $\frac{TP}{TP+FP}$

    - Recall (재현율)
      - 실제 Positive 중에서 분류기가 Positive로 분류한 비율, $\frac{TP}{TP+FN}$

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - F1 score
      - Accuracy (정확도): Positive 및 Negative 정답을 맞춘 비율
      - Precision (정밀도): 분류기가 Positive로 분류한 결과 중에서 실제 Positive의 비율
      - Recall (재현율): 실제 Positive 중에서 분류기가 Positive로 분류한 비율
      - Accuracy 가 가장 중요한 성능 지표인데, 왜 정밀도, 재현율을 고려하는지...?
      - 예: MRI 사진으로 암을 판별하는 분류기
        - 데이터 수: 양성 10건, 음성 90건
        - 학습한 분류기 성능

	실제 양성	실제 음성
예측 양성	True Positive = 5	False Positive = 3
예측 음성	False Negative = 5	True Negative = 87

        - 정확도: 양성/음성을 정확히 판단한 비율 = (5+87) / (5+3+5+87) = 92% // 상당한 수준의 정확도를 보이고 있음
        - 하지만, 재현율 및 정밀도를 측정해 본 결과, 분류기의 성능이 좋지 않음을 알 수 있음
        - 재현율: 실제 Positive 중에서 분류기가 Positive로 분류한 비율 = 5 / (5+5) = 50%
        - 정밀도: 실제 Positive 중에서 분류기가 Positive로 분류한 비율 = 5 / (5+3) = 0.625
        - 극단적인 경우, 모든 샘플을 음성이라고 판단하기만 해도 90%의 정확도를 얻을 수 있음
      - 따라서, 분류 클래스 간 데이터가 심각한 불균형을 이루는 경우에는 F1 score를 사용해서 분류기의 성능을 분석하는 편이 더 정확함
        - 참고로, F1 score의 최대값은 1.0이고(최고의 분류기), 최소값은 0.0(최악의 분류기)임

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지)
    - F1 score를 활용한 정상/비정상 판단을 위한 임계치 설정 알고리즘
      - 목표: F1 score를 최대화 하는 ϵ 찾기
      - ϵ 값은 Gaussian distribution의 pdf 값이므로 [0.0, 1.0] 사이의 값을 가짐
    - <Exhaustive Search 알고리즘>

Line	알고리즘
1	`stepsize = 매우 작은 수 (예: 1/1000)`
2	`currEpsilon = 0, bestF1 = 0, bestEpsilon = 0`
3	`While (currEpsilon ≤ 1.0)`
4	`    currEpsilon 값으로 이상 탐지를 수행한 후, F1 score 값을 계산하여 currF1 변수에 저장`
5	`    IF( currF1 > bestF1 )`
6	`        bestF1 = currF1`
7	`        bestEpsilon = currEpsilon`
8	`    currEpsilon += stepsize`
9	`print("Best Epsilon : ", bestEpsilon)`

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - 사전 작업: 필수 라이브러리 import

필수 라이브러리 import

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from scipy.io import loadmat
```

데이터 셋 연동을 위해
구글 드라이브 마운트

```
[ ] # 구글 드라이브 마운트
from google.colab import drive
drive.mount('/content/drive')
```

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - 데이터 셋 불러오기

```
"""
데이터 셋에는 X, Xval, Yval 이렇게 3개의 데이터가 들어있는데
- Xval 및 Yval은 예전에 측정한 feature 값 및 각각에 대한 anomaly 여부에 해당
  => cross-validation data set 이라고 생각하면 됨
  => 여기에서 threshold를 찾고, 이를 이용해서 X에서의 anomaly를 탐지
- X는 최근에 측정한 feature 값이며, 이 값에 대해서 anomaly 여부를 탐지해야 함
- X 및 Xval에 포함된 각 데이터 샘플은 2개의 feature 값을 가짐
- yval에 포함된 탐지 결과는 binary: 0(normal), 1(abnormal)
"""

# mat 형식(matlab)으로 저장된 데이터 불러오기
data1_path = '/content/drive/MyDrive/Colab Notebooks/AnomalyDetection/data1.mat'
data1 = loadmat(data1_path)
# 불러온 데이터의 column title 출력하기
print('Data set 1: ', data1.keys())

#data2_path = '/content/drive/MyDrive/Colab Notebooks/AnomalyDetection/data2.mat'
#data2 = loadmat(data2_path)
#print(data2.keys())
```

```
Data set 1:  dict_keys(['__header__', '__version__', '__globals__', 'X', 'Xval', 'yval'])
```

실습에 필요한 데이터 값

Sensing Layer: 데이터 정제

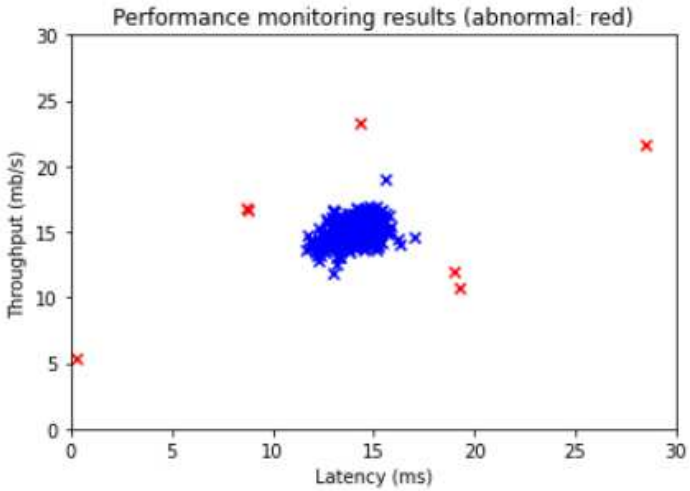
- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지) : 구현
    - 데이터 설명
      - 운영자가 관리하는 시스템의 성능을 측정한 데이터를 모은 데이터 셋
      - Latency (ms) 및 throughout (Mb/s) 을 측정한 값

Column	설명
X	새롭게 획득한 Latency, Throughput 데이터 셋 • X.shape = (307, 2) • 즉, (Latency, Throughput) 쌍으로 구성된 데이터가 총 307개
Xval yval	Xval: 기존에 획득한 Latency, Throughput 데이터 셋 (Cross-Validation 데이터 셋) yval: 기존에 획득한 데이터 각각에 대해 정상/비정상 여부를 표시한 값 • yval[i]=1: i번째 데이터는 비정상(anomaly) • yval[i]=0: i번째 데이터는 정상(normal)
공통	X와 Xval 모두 동일한 시스템을 대상으로 측정한 값이기 때문에, 동일한 특성을 가짐 (즉, mean, co-variance matrix가 동일하다고 가정함)

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - (사전에 분류된) Cross-Validation 데이터 셋에서, 정상/비정상 데이터 샘플을 Blue/Red 컬러로 plotting 하기

```
# Cross-Validation 데이터 셋(labelled) 불러오기
# 사전에 측정한 데이터 샘플이며, 각 샘플별로 정상/비정상 여부가 저장되어 있음
Xval = data1['Xval']
print('Xval: ', Xval.shape)

Yval = data1['yval']
print('Yval: ', Yval.shape)

# 그래프로 출력하기(정상: blue, 비정상: red)
for i in range( Xval.shape[0]):
  if Yval[i] == 0:  # 정상
    plt.scatter(Xval[i,0], Xval[i,1], marker='x', c='b')
  else:  # 비정상 (abnormal)
    plt.scatter(Xval[i,0], Xval[i,1], marker='x', c='r')

plt.xlabel('Latency (ms)')
plt.ylabel('Throughput (mb/s)')
plt.xlim(0,30)
plt.ylim(0,30)
plt.title('Performance monitoring results (abnormal: red)')
```

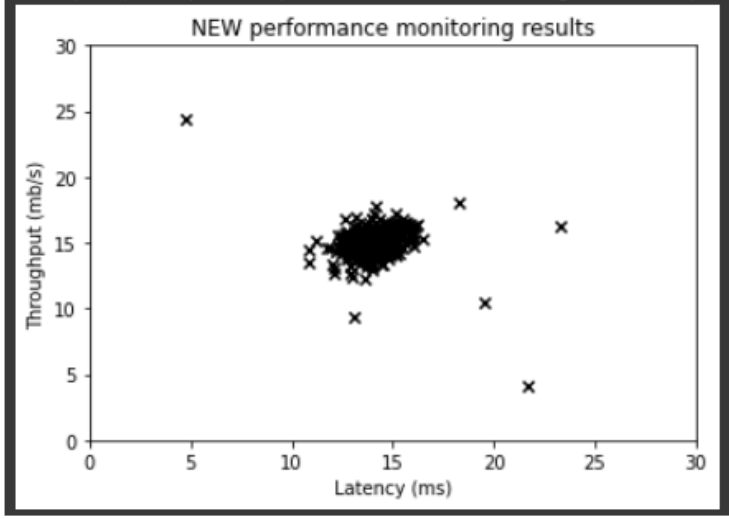


Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - 새로 획득한 (정상/비정상을 분류해야 할) 데이터 셋을 plotting 하기

```
# 새로 획득한 샘플(Unlabelled) 불러오기(최근에 획득한 데이터 샘플)
Xnew = data1['X']
print('X: ', Xnew.shape)

# 시각화하기
plt.scatter(Xnew[:,0], Xnew[:,1], marker='x', c='k')
plt.xlabel('Latency (ms)')
plt.ylabel('Throughput (mb/s)')
plt.xlim(0,30)
plt.ylim(0,30)
plt.title('NEW performance monitoring results')
```

```
X:  (307, 2)
Text(0.5, 1.0, 'NEW performance monitoring results')
```



Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - 이후 구현 코드 흐름 정리

```
"""
Anomaly Detection 알고리즘
- 정상/비정상을 판단할, 최근에 획득한 데이터 셋: Xnew
- Cross-Validation 데이터 셋: (Xval, Yval)
1. Cross-Validation 셋으로 부터 multi-variate Gaussian parameter fitting
   : mean (mu), covariance matrix (var) 계산
2. mu, var로 정의되는 Multi-Variate Gaussian dist.을 이용해서 Xval의 density인 pval (확률값) 얻기
3. pval, Yval를 이용해서 최적의 threshold 찾기(F1 score 활용)
4. Xnew 값을 Gaussian(mu, var)에 대입하여, density 값 p 얻기
5. p, epsilon 값을 사용해서 Xnew에서 비정상 샘플 찾기
"""
```

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - Cross-Validation 셋으로 부터 multi-variate Gaussian parameter fitting

```
"""
1. Cross-Validation 셋으로 부터 multi-variate Gaussian parameter fitting
  : mean (mu), covariance matrix (var) 계산

* Diagonal element 뿐 아니라, non-diagonal element 까지도 계산이 가능해서
  feature간의 correlation까지 모델링 할 수 있음
"""

Xval = np.array(Xval)
print('Shape: ', Xval.shape)
data_len = Xval.shape[0]
num_feature = Xval.shape[1]

mu = np.zeros(2)
for i in range(data_len):
  mu = mu + Xval[i,:]

mu = mu / data_len
print('Mean : ', mu)

var = np.zeros(4).reshape((2,2))
for i in range(data_len):
  var = np.add(var, np.dot((Xval[i,:]-mu).reshape((2,1)),
                           (Xval[i,:]-mu).reshape((1,2))))

var = var / data_len
print('Sigma : ', var)
```

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - mu, var로 정의되는 Multi-Variate Gaussian dist.을 이용해서 Xval의 density인 pval (확률값) 얻기

```
"""
2. mu, var로 정의되는 Multi-Variate Gaussian dist.을 이용해서
   Xval의 density인 pval (확률값) 얻기

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.multivariate_normal.html
"""


from scipy.stats import multivariate_normal
pval = multivariate_normal.pdf(Xval, mean=mu, cov=var);
```

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - pval, Yval를 이용해서 최적의 threshold인 epsilon 찾기(F1 score 활용)

```
"""
3. pval, Yval를 이용해서 최적의 threshold인 epsilon 찾기(F1 score 활용)
"""
import sklearn.metrics

bestEpsilon = 0
bestF1 = 0
F1 = 0

stepsize = (max(pval) - min(pval)) / 1000.0
eps = min(pval)

while eps < max(pval):
  prediction = np.zeros(data_len)
  for i in range(data_len):
    if pval[i] < eps:  # 오류로 판단되는 경우에, 예측값을 1로 표기
      prediction[i] = 1

  F1 = sklearn.metrics.f1_score(Yval.reshape(-1), prediction)
  #print(F1, eps)
  if F1 > bestF1:  # 더 좋은 F1 score를 찾았다면, update
    bestF1 = F1
    bestEpsilon = eps

  #elif F1 == bestF1 and eps < bestEpsilon:  # 더 작은 eps 값이 있다면, 그걸 선택
  #  bestEpsilon = eps

  eps = eps + stepsize

print('Best F1: ', bestF1)
print('Best Eps: ', bestEpsilon)
```

Sensing Layer: 데이터 정제

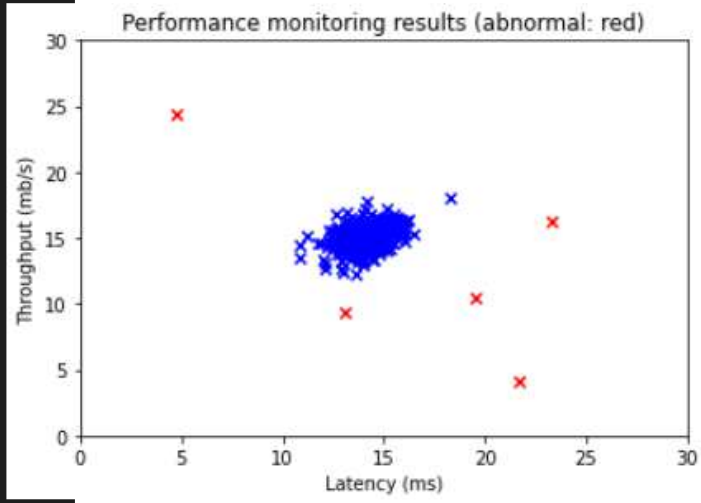
- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - Xnew 값을 Gaussian(mu, var)에 대입하여, density 값 p 얻기

```
"""
4. Xnew 값을 Gaussian(mu, var)에 대입하여, density 값 p 얻기
"""
p = multivariate_normal.pdf(Xnew, mean=mu, cov=var);
```

Sensing Layer: 데이터 정제

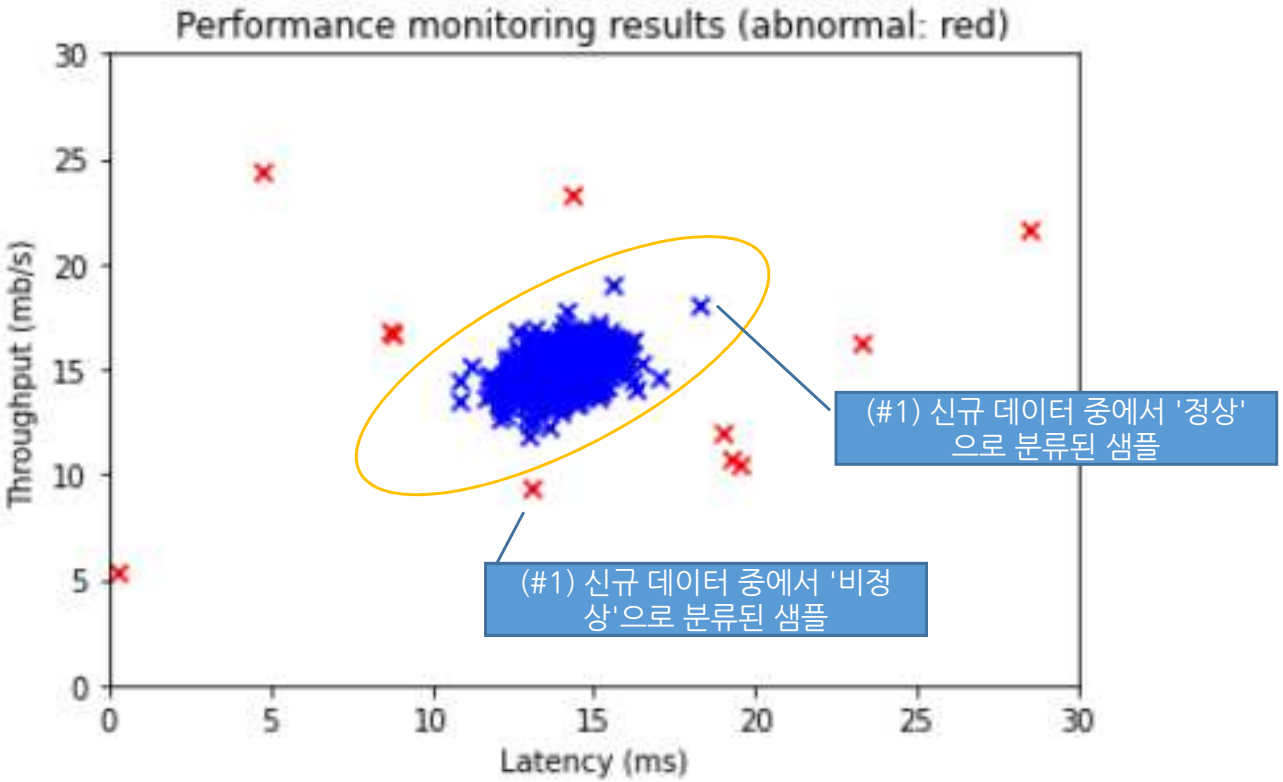
- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - p, epsilon 값을 사용해서 Xnew에서 비정상 샘플 찾기

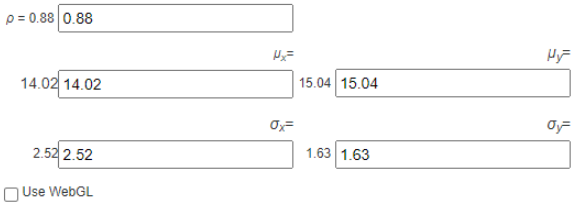
```
"""
5. p, epsilon 값을 사용해서 Xnew에서 비정상 샘플 찾기
"""
from scipy.stats import multivariate_normal

y_pred = np.zeros(data_len)

for i in range(data_len):
  if p[i] < bestEpsilon:
    y_pred[i] = 1

for i in range(Xnew.shape[0]):
  if y_pred[i] == 0:
    plt.scatter(Xnew[i,0], Xnew[i,1], marker='x', c='b')
  else:
    plt.scatter(Xnew[i,0], Xnew[i,1], marker='x', c='r')

plt.xlabel('Latency (ms)')
plt.ylabel('Throughput (mb/s)')
plt.xlim(0,30)
plt.ylim(0,30)
plt.title('Performance monitoring results (abnormal: red)')
```

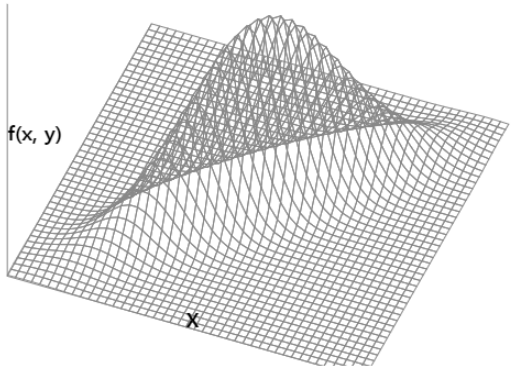


Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing)
  - 4. Anomaly detection (이상 탐지): 구현
    - 기존 데이터와 신규 데이터를 통합해서 plotting 하기



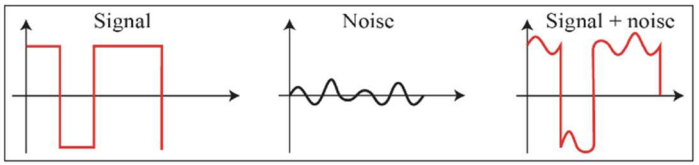




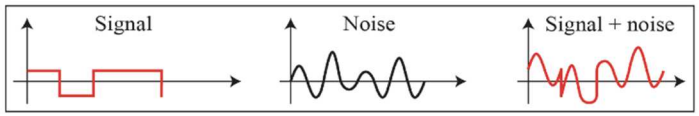
Multi-Variate Gaussian distribution이 위와 같이 기울어진 타원형으로 형성 되었고, 이 때문에 (#1)은 정상으로, (#2)는 비정상으로 분류됨

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈/잡음
    - 계측기에서 검출되는 불필요한 신호.
    - 열, 빛, 소리 등 다양한 원천으로부터 발생하며, 대부분 의미 없는 값으로 표현되고 원하는 신호의 정확한 감지를 방해함
    - 예: 통신에서의 잡음의 영향

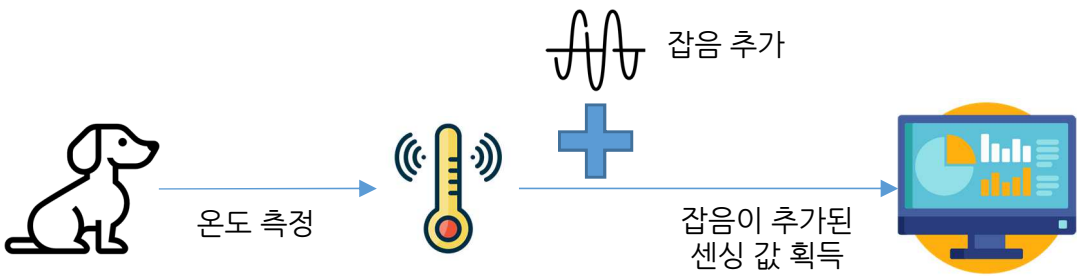


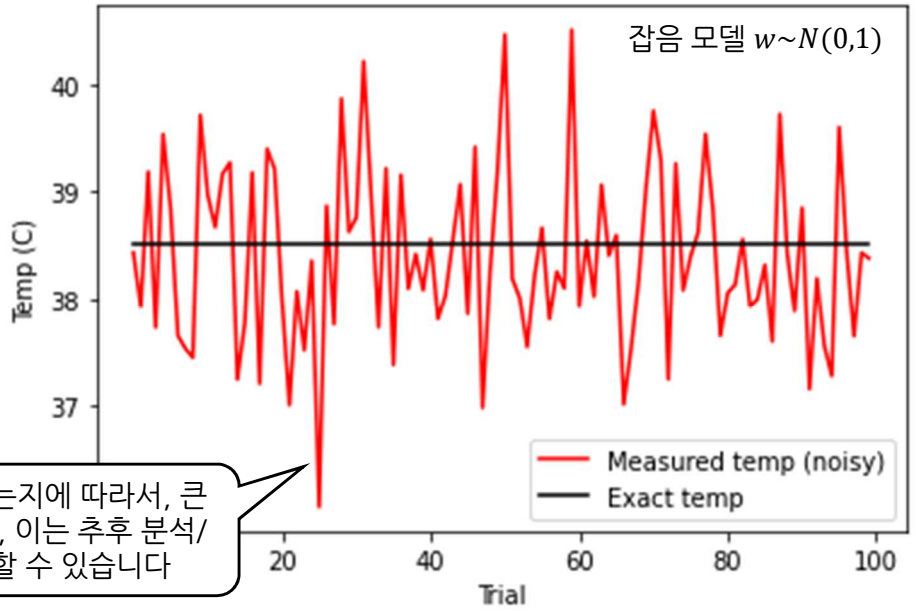
a. High SNR



b. Low SNR

    - 예: 강아지의 체온 센서에서의 잡음의 영향(실제 값: 38.5도)





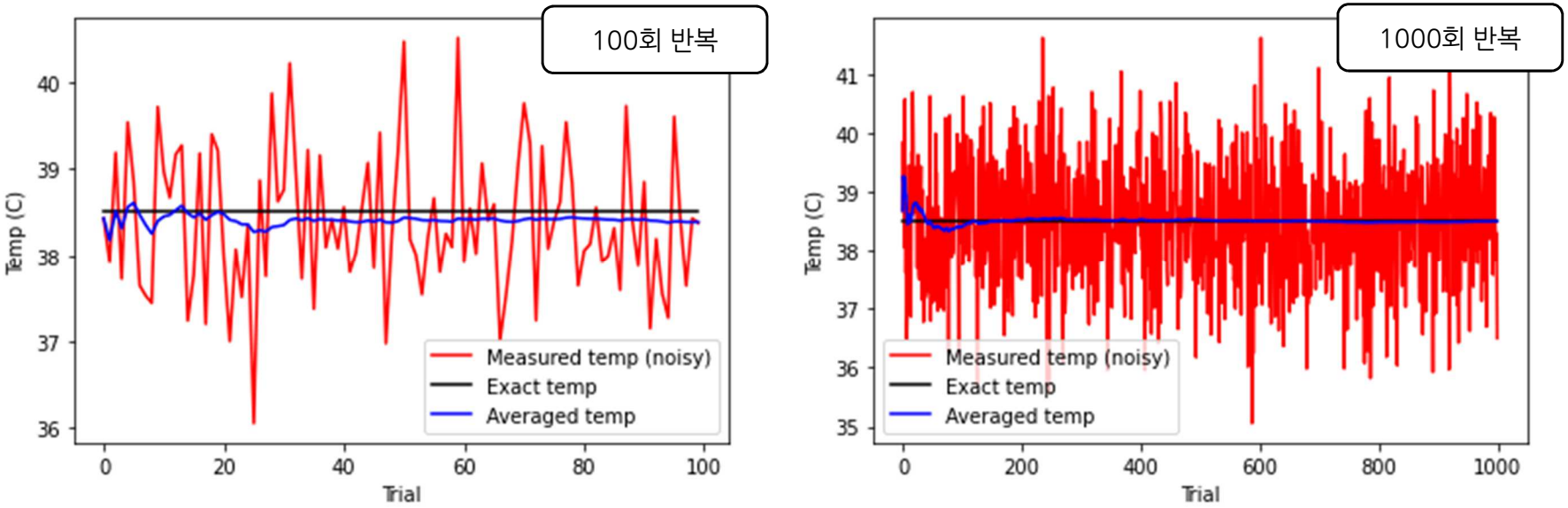
어느 시점에서 센서 값을 획득하는지에 따라서, 큰 오차를 가진 값을 획득할 수 있고, 이는 추후 분석/예측 과정에서의 오차를 유발할 수 있습니다

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 측정 데이터에 포함된 노이즈를 처리하는 방법
    - ~~고가의 고정밀 센서를 사용 => 제품 단가 상승 => 시장 경쟁력 하락~~
    - 노이즈가 포함된 측정값을 그대로 사용하되, 노이즈의 영향을 상쇄시킬 수 있는 기법을 활용
    - 먼저 노이즈를 (최대한) 제거하고, 노이즈가 제거된 데이터를 사용

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 평균 필터
    - 평균: 데이터의 총 합을 데이터 개수로 나눈 것
    - K개의 데이터가 있을 때, 산술 평균(arithmetic mean)은: $\sum_{i=1}^{k} x_i / k$
    - 평균 필터의 효과: 예) 38.5도 체온을 가진 강아지의 체온 측정 => 반복 횟수가 증가할 수록 평균 값이 실제 값에 근사



    - 평균 필터의 이론적 근거: Law of Large Numbers (다음 페이지)

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 평균 필터
    - Law of Large Numbers (LLN, 큰 수의 법칙): 표본의 크기 n이 증가함에 따라 표본 평균($\bar{X}_n$, 샘플을 통해서 얻은 평균값) 이 모평균(μ, 실제 기대 값)에 수렴하는 현상을 말함

    - Week LLN

THEOREM 3.1. *Let $X_1, X_2, ..., X_n$ be a sequence of i.i.d. random variables, each with mean $E(X_i) = \mu$ and standard deviation σ, we define $\bar{X}_n = \frac{X_1+X_2+...+X_n}{n}$. The Weak Law of Large Numbers (WLLN) states for all $\epsilon > 0$ then,*

$$\lim_{n\to\infty} P(\mid \bar{X}_n - \mu \mid > \epsilon) = 0.$$

    - Strong LLN

THEOREM 3.5. *(The Strong Law of Large Numbers) Let $X_1, X_2, ..., X_n$ be a sequence of i.i.d. random variables, each with mean $E(X_i) = \mu$ and standard deviation σ then,*

$$P(\lim_{n\to\infty} \bar{X}_n = \mu) = 1.$$

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 평균 필터
    - 배치식 계산 방법: k개의 데이터가 있을 때, 모든 데이터에 대한 평균은 $\sum_{i=1}^{k} x_i \,/k = (x_1 + \cdots + x_n)/k$
    - 배치식 계산 방법의 단점?
      - k개 데이터의 평균을 계산하기 위해서는 k개의 데이터를 저장하고 있어야 함 // 공간 복잡도!
      - $k+1$번째 데이터를 획득한 후, 전체 평균을 다시 계산해야 하며, 기존에 계산한 결과를 전혀 사용할 수 없음 // 시간 복잡도!
    - 재귀식 계산 방법
      - 배치식 계산 방법의 비효율적인 공간 복잡도와 시간 복잡도를 개선한 방식
      - 직전에 계산한 결과를 활용하므로 계산 효율이 좋음(즉, 시간 복잡도가 낮음)
      - 또한, 메모리에 저장하고 있어야 하는 정보의 양이 배치식 계산 방법에 비해 매우 작음(즉, 공간 복잡도가 낮음)
      - (다음 페이지…)

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 평균 필터
    - 재귀식 계산식 유도하기: $\bar{x}_k = \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k$

Step-1	K개의 데이터 평균	$\bar{x}_k = \frac{x_1 + x_2 + \cdots x_k}{k}$	식 (1.1)
Step-2	K-1개의 데이터 평균	$\bar{x}_{k-1} = \frac{x_1 + x_2 + \cdots x_{k-1}}{k-1}$	식 (1.2)
Step-3	(1.1)식의 양변에 k를 곱하기	$k\bar{x}_k = x_1 + x_2 + \cdots x_k$	
Step-4	위의 식의 양변을 k-1로 나누기	$\frac{k}{k-1}\bar{x}_k = \frac{x_1 + x_2 + \cdots + x_k}{k-1} = \frac{x_1 + x_2 + \cdots x_{k-1}}{k-1} + \frac{x_k}{k-1}$	
Step-5	위의 식에서 붉은색으로 표시된 부분을 $\bar{x}_{k-1}$로 치환	$\frac{k}{k-1}\bar{x}_k = \bar{x}_{k-1} + \frac{x_k}{k-1}$	
Step-6	위의 식에서 양변을 $\frac{k}{k-1}$로 나누기	$\bar{x}_k = \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k$	식 (1.3)

    - 재귀식을 사용해서 평균을 계산하면, 직전 평균값($\bar{x}_{k-1}$), 데이터 개수(k), 추가로 획득한 측정값(x_k)만 알고 있으면 되므로 메모리 사용량을 절약할 수 있고, 연산의 복잡도(=연산량)도 감소함

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 평균 필터
    - 재귀식: $\bar{x}_k = \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k$
    - $\alpha \equiv \frac{k-1}{k}$ 라고 정의하면, $\alpha \equiv \frac{k-1}{k} = 1 - \frac{1}{k}$가 되고, $\frac{1}{k} = 1 - \alpha$ 관계가 성립함
    - 재귀식을 다시 정리하면 $\bar{x}_k = \alpha\bar{x}_{k-1} + (1-\alpha)x_k$ 가 되고, 이러한 형태를 평균 필터(average filter)라고 함
    - 평균 필터는 평균값을 계산하는 방식이지만, 이를 통해 잡음을 제거하는 효과를 얻을 수 있기 때문에 (잡음을 제거한다는 의미를 가지는) “필터” 라는 이름이 붙음
    - 평균 필터는 실시간으로 처리가 필요하고(낮은 시간 복잡도) 처리해야하는 데이터 양이 큰 경우에(낮은 공간 복잡도) 효과적으로 사용할 수 있는 노이즈 필터임

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 이동 평균 필터(moving average filter)
    - 평균 필터의 경우 수집한 모든 데이터의 평균을 계산하는 방식으로 노이즈의 영향을 상쇄함
    - 고정된 값을 반복적인 측정을 통해 얻는 경우에 유용함
      - 예: 오차가 있는 체중계를 반복적으로 사용하여 실제 몸무게를 측정
      - 예: 오차가 있는 길이 측정 센서를 반복적으로 사용하여 건물의 길이를 측정
    - 하지만, 측정하려는 물리량이 시간에 따라 변하는 경우에는 적합하지 않음
      - 예: 가까운 미래의 주식 가격을 예측하기 위해 주식 상장 초기부터 지금까지의 가격을 모두 사용해서 평균 필터를 적용하는 경우, 배치로 계산한 평균값은 큰 의미가 없음
    - (측정하려는 물리량의 동적인 변화를 고려하여) 현재의 물리량을 측정하되 잡음의 영향을 상쇄하려는 경우, 평균 필터 대신 “이동 평균 필터”가 적합함
      - 예: 주식에서 5일, 20일, 60일 이평선(이동평균선)은 최근 5일, 20일, 60일간의 주가를 평균 낸 것으로 최근의 주식 값의 평균치를 계산함
    - 이동 평균(moving average)
      - 모든 측정 데이터가 아니라, 지정된 개수의 최근 측정값만으로 계산한 평균
      - 새로운 측정 데이터가 들어오면 가장 오래된 데이터를 버리는 방식으로, 계산에 사용할 데이터 개수를 일정하게 유지하면서 평균을 계산
      - k개 값을 저장할 수 있는 선입선출 큐(FIFO Queue)를 사용하고, 큐에 저장된 모든 값의 평균을 구하는 것과 같은 원리

Sensing Layer: 데이터 정제

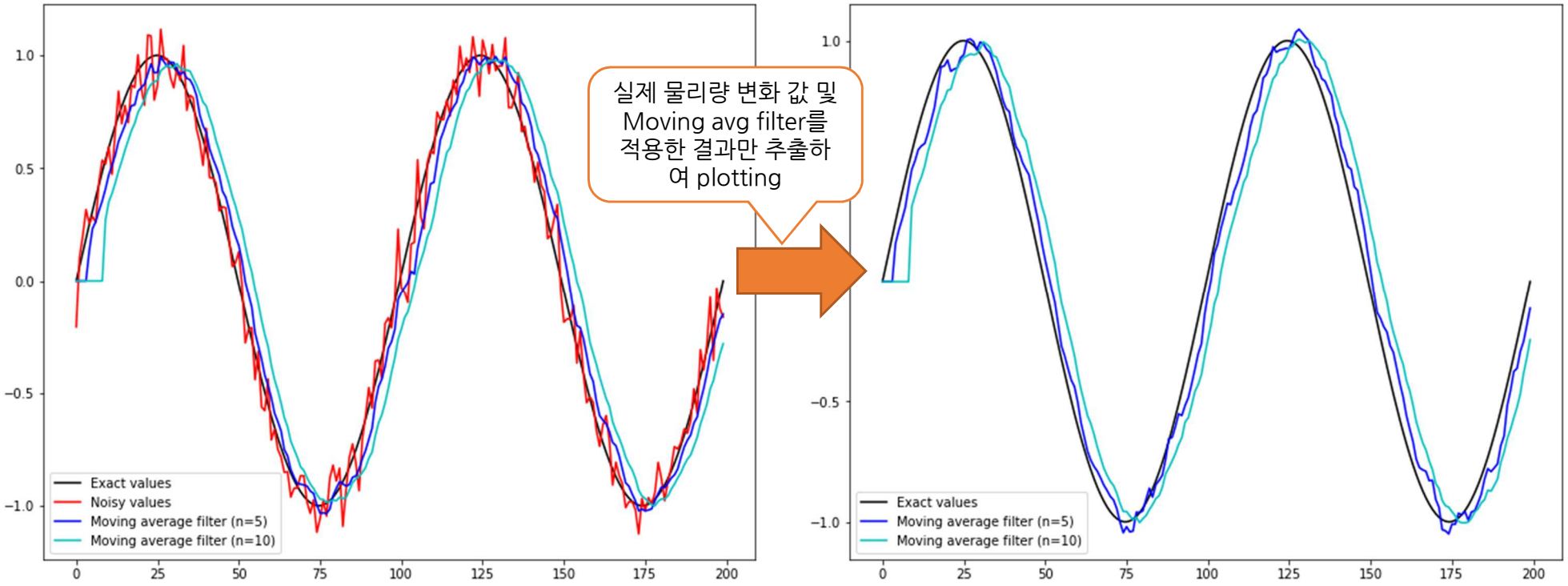
- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 이동 평균 필터(moving average filter)

Step-1	K번째 획득한 샘플을 기준으로, 최근 n개 데이터의 이동 평균	$\bar{x}_k = \frac{x_{k-n+1} + x_{k-n+2} + \cdots x_k}{n}$	식 (2.1)
Step-2	K-1번째 획득한 샘플 기준, 최근 n개의 데이터 평균	$\bar{x}_{k-1} = \frac{x_{k-n} + x_{k-n+1} + \cdots x_{k-1}}{n}$	식 (2.2)
Step-3	(2.1) 식에서 (2.2) 식을 빼기	$\bar{x}_k - \bar{x}_{k-1} = \frac{x_k - x_{k-n}}{n}$	
Step-4	위의 식을 $\bar{x}_k$에 대해서 정리	$\bar{x}_k = \bar{x}_{k-1} + \frac{x_k - x_{k-n}}{n}$	식 (2.3)

  - 구현
    - 배치식: $\bar{x}_k = (x_{k-n+1} + x_{k-n+2} + \cdots x_k)/\mathrm{n}$
    - 재귀식: $\bar{x}_k = \bar{x}_{k-1} + (x_k - x_{k-n})/\mathrm{n}$
    - 재귀식이 효율적인 것 처럼 보이지만, 실제로는 배치식에 비해 큰 장점이 없음
      - 평균 필터는 누적된 모든 데이터를 사용하지만, 배치식 이동 평균 필터는 최근 n개 데이터만 사용하므로 n이 아주 큰 경우가 아니라면 배치식을 사용해도 연산 복잡도가 크지 않음
      - 재귀식의 경우 가장 오래된 데이터인 x_{k-n} 값을 사용하는데, k+1번째 데이터가 들어오게 되면 그 다음으로 오래된 데이터인 $x_{k-n}+1$이 필요함. 즉, 최근 n개 만큼의 데이터를 보관하고 있어야 하며, 이는 배치식의 메모리 사용량과 유사한 수준의 메모리양을 요함
    - n=5인 이동 평균 필터의 경우, 처음에 5개의 샘플이 모이기 전까지는 평균값을 계산할 수 없음

Sensing Layer: 데이터 정제

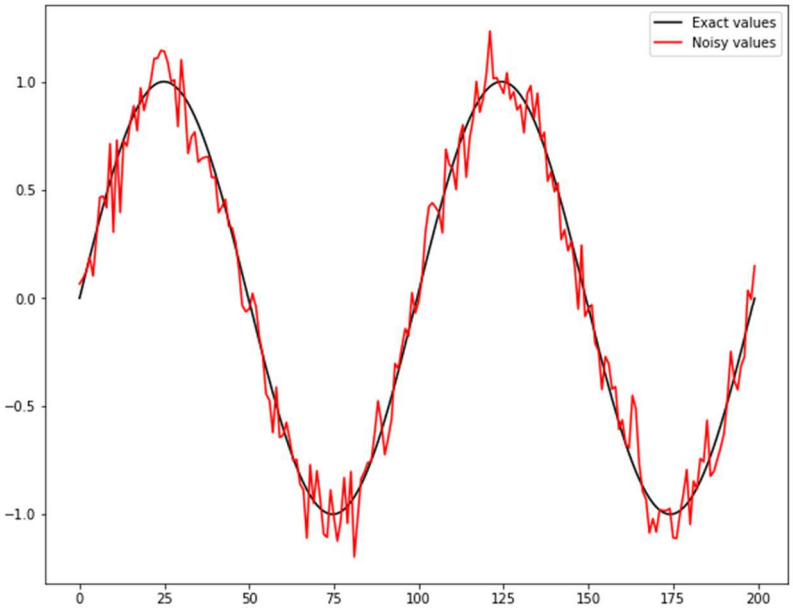
- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 이동 평균 필터(moving average filter)
    - 예: 노이즈($w \sim N(\mu = 0, \sigma = 0.1)$) 가 섞인 sine 파형에 이동 평균 필터 적용



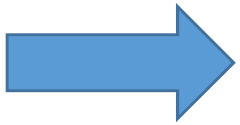
  - 잡음 제거와 변화 민감성을 기준으로 평가하여, 상황에 맞는 n 값을 선정해야 함
    - 잡음 제거: 효과적으로 잡음을 제거하는지?
    - 변화 민감성: 실제 물리량의 변화를 신속하게 반영하는지? (즉, 실제 물리량의 변화가 즉각 반영되는지?)

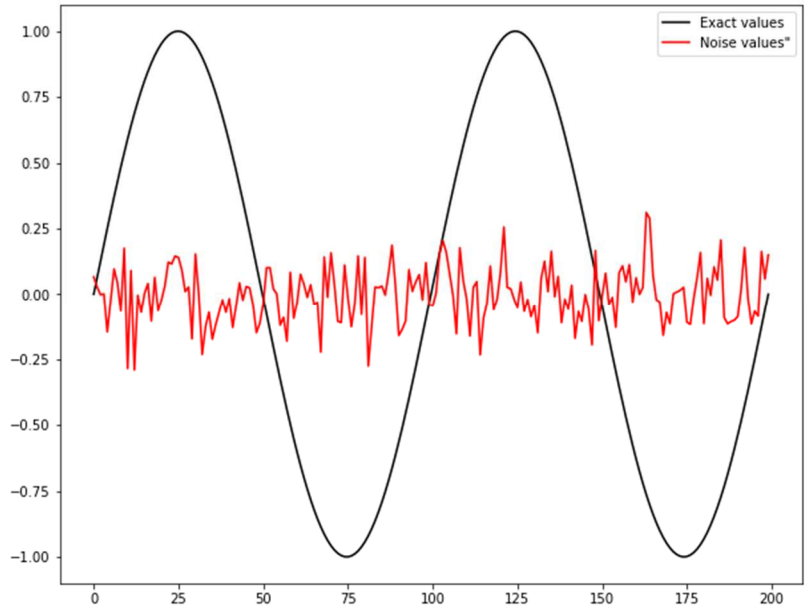
Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 저주파 통과 필터(Low-Pass Filter, LPF)
    - 저주파 신호는 통과시키고 고주파 신호는 걸러내는 필터
    - 잡음 제거용으로 많이 사용하는데, 대개 측정하려는 신호는 저주파이고(상대적으로 느리게 변화) 잡음은 고주파 성분으로(상대적으로 빠르게 변화) 되어 있기 때문



잡음이 더해진 신호를 plotting





측정하려는 신호와 잡음을 분리하여 plotting

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 저주파 통과 필터(Low-Pass Filter, LPF)
    - 이동 평균 필터의 한계
    - 이동 평균 필터: $\bar{x}_k = \frac{x_{k-n+1}+x_{k-n+2}+\cdots x_k}{n} = \frac{1}{n}x_{k-n+1} + \frac{1}{n}x_{k-n+2} + \cdots + \frac{1}{n}x_k$
    - 이동 평균 필터는 모든 데이터 값에 동일한 가중치($1/n$)를 부여함. 즉, 가장 오래된 데이터와 가장 최근의 데이터가 동일한 비중으로 평균에 반영됨
    - 최근의 데이터 일수록 높은 가중치를 부여하면, 최신의 변화를 빠르게 반영하는 값을 얻을 수 있지 않을까?

이것이 저주파 통과 필터의 기본 아이디어!

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 1차 저주파 통과 필터(Low-Pass Filter, LPF)
    - 최근 측정값에는 높은 가중치를, 오래된 값에는 낮은 가중치를 주는 필터 (+ 노이즈 감소)

식 (3.1)

- $\bar{x}_k = \alpha\bar{x}_{k-1} + (1-\alpha)x_k$, 이 때 α 는 $0 < \alpha < 1$ 범위의 임의의 상수
- 위 수식은 평균 필터의 재귀식과 형태가 동일함. 단, 평균 필터에서는 α 값을 임의로 정하지 않고 데이터 개수에 따라 값이 계산되었음
- 또한, LPF에서 $\bar{x}_k$은 평균값과 전혀 관련이 없으며, $\bar{x}_k$를 평균값이 아닌 추정값(estimated value)이라고 부름

식 (3.2)

- 식 (3.1)을 이용해, 직전 추정값을 계산하면: $\bar{x}_{k-1} = \alpha\bar{x}_{k-2} + (1-\alpha)x_{k-1}$
- 식 (3.2)를 식 (3.1)에 대입해서 정리하면:

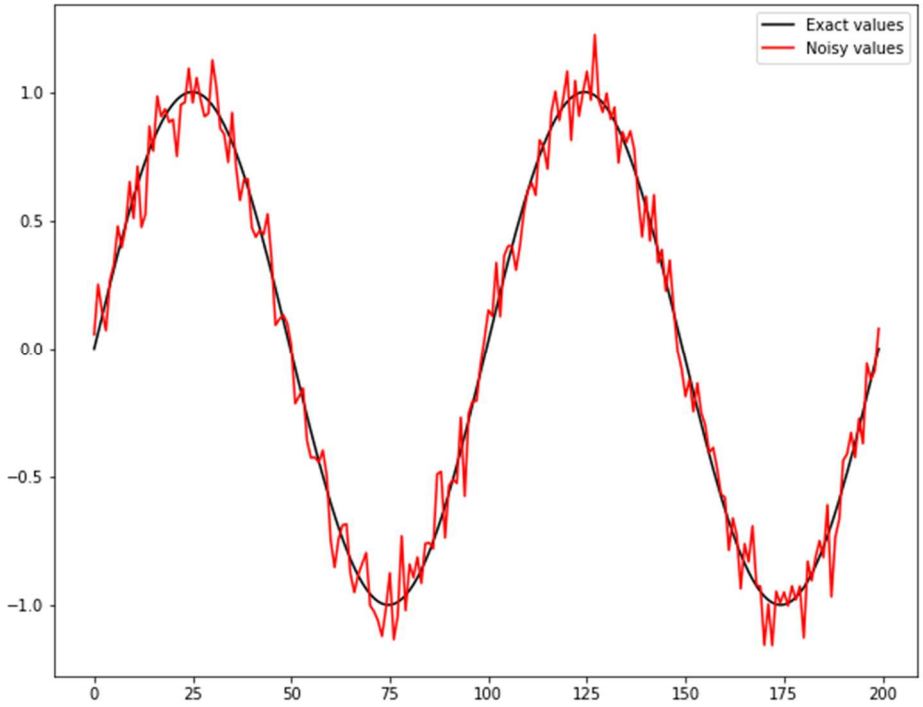
식 (3.3)

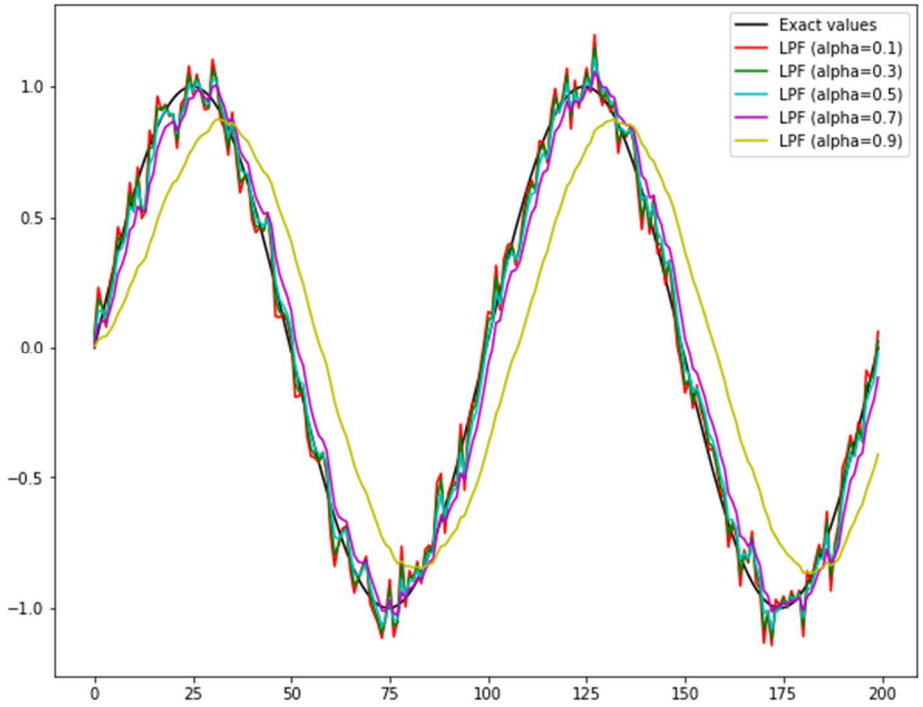
$$\begin{aligned}\bar{x}_k &= \alpha\bar{x}_{k-1} + (1-\alpha)x_k \\ &= \alpha\{\alpha\bar{x}_{k-2} + (1-\alpha)x_{k-1}\} + (1-\alpha)x_k \\ &= \alpha^2\bar{x}_{k-2} + \alpha(1-\alpha)x_{k-1} + (1-\alpha)x_k\end{aligned}$$

- α 는 $0 < \alpha < 1$ 범위의 상수임을 고려할 때, $\alpha(1-\alpha) < 1-\alpha$ 임을 알 수 있음 => 즉, 최근 측정값(x_k)이 이전 측정값(x_{k-1}) 보다 더 큰 가중치를 부여 받아 추정 값 계산에 반영되고 있음
- 같은 방식으로, $\bar{x}_{k-2}$를 식 (3.3)에 대입해서 정리하면: $\bar{x}_k = \alpha^3\bar{x}_{k-3} + \alpha^2(1-\alpha)\bar{x}_{k-2} + \alpha(1-\alpha)x_{k-1} + (1-\alpha)x_k$
- α 는 $0 < \alpha < 1$ 범위의 상수임을 고려할 때,$\alpha^2(1-\alpha) < \alpha(1-\alpha) < 1-\alpha$ 임을 알 수 있음 => 즉, 최근 측정값이 이전 측정값 보다 더 큰 가중치를 부여 받아 추정 값 계산에 반영되고 있음
- 이러한 특성을 통해, LPF는 잡음 제거와 변화 민감성이라는 상충되는 요구를 이동 평균 필터보다 더 잘 충족시킴
- 1차 저주파 통과 필터는 “지수 가중 이동 평균 필터(exponentially weighted moving average filter)”라고도 불리며, 이는 오래된 데이터일수록 기하급수적으로 가중치를 낮게 부여하여 이동평균을 계산하기 때문

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)
  - 노이즈 영향 상쇄 기법: 1차 저주파 통과 필터(Low-Pass Filter, LPF)





- α 값이 작을수록, 변화 민감도는 커지지만 잡음 제거 효과는 감소
- α 값이 클수록, 변화 민감도는 작아지지만 잡음 제거 효과는 증가
- 즉, 상황에 맞는 최적의 α값을 찾아서 사용해야 함

Sensing Layer: 데이터 정제

- 데이터 정제(Data cleaning/cleansing/scrubbing): 5. Noise reduction (잡음 제거)

  - 측정 데이터에 포함된 노이즈를 처리하는 방법
    - ~~고가의 고정밀 센서를 사용 => 제품 단가 상승 => 시장 경쟁력 하락~~

    - 노이즈가 포함된 측정값을 그대로 사용하되, 노이즈의 영향을 상쇄시킬 수 있는 기법을 활용 => 지금까지 학습한 내용

    - 먼저 노이즈를 (최대한) 제거하고, 노이즈가 제거된 데이터를 사용
      - 온도, 습도, 무게 등 저차원(예: 스칼라 값)의 측정값의 경우에는 LPF 등을 사용해서 손쉽게 잡음의 영향을 제거할 수 있지만, 2D 이미지 또는 다수의 오디오 소스가 중첩된 음성신호와 같은 고차원의 데이터에서는 LPF 적용이 어려움
      - 따라서, 측정 데이터(이미지, 오디오 등)에서 노이즈를 제거할 수 있는 별도의 기법이 필요 => 다음시간에…

끝.