Nonlinear Models

실습 코드 : https://bit.ly/3DC80BA

2022.09.16.

한국에너지기술연구원 계산과학연구실

이제현

머신 러닝 강좌

- base : Scikit-learn MOOC @inria
  - 머신 러닝 위주. 딥 러닝은 skip (기회가 되면 한번쯤은 다룰 수도..?)
  - 소스 코드 포함 강의 자료 : 원내 게시판 공개
  - 강의 영상 : KIER-Tube & Youtube 공개

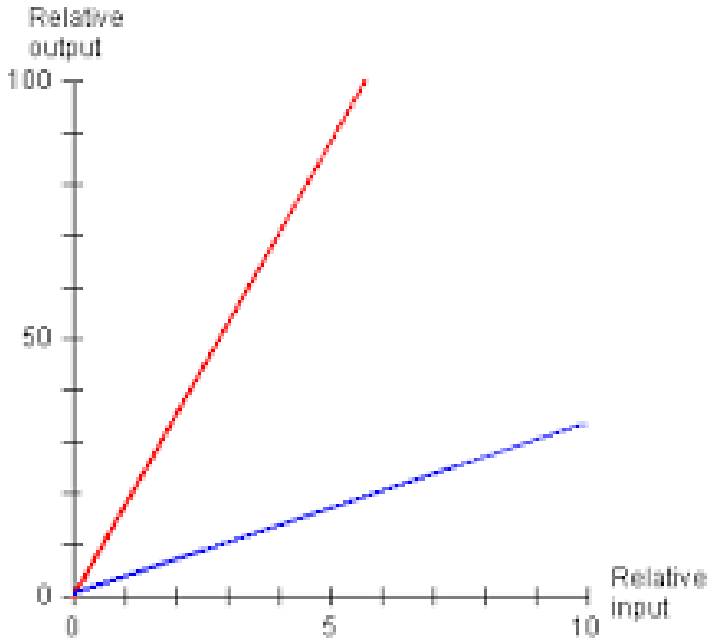
1차 모임 (4월) 머신러닝 기본 개념			
2차 모임 (5월) Modeling pipeline	3차 모임 (6월) Best Model?	4차 모임 (7월) Tree Models	5차 모임 (8월) Hyperparameter
6차 모임 (8월) Nonlinear Models	7차 모임 (9월) Ensemble Models	8차 모임 (10월) Neural Network (?)	9차 모임 (11월) 마무리

Linearity

- 정의 (Wikipedia)
  - Property of a mathematical relationship(function) that can be graphically represented as a straight line. Closely related to proportionality.
  - 직선처럼 똑바른 도형, 또는 그와 비슷한 성질을 갖는 대상. 이러한 성질을 갖고 있는 변환.

가산성 (additivity) : $f(a+b) = f(a) + f(b)$

동차성 (homogeneity) : $f(ka) = kf(a)$

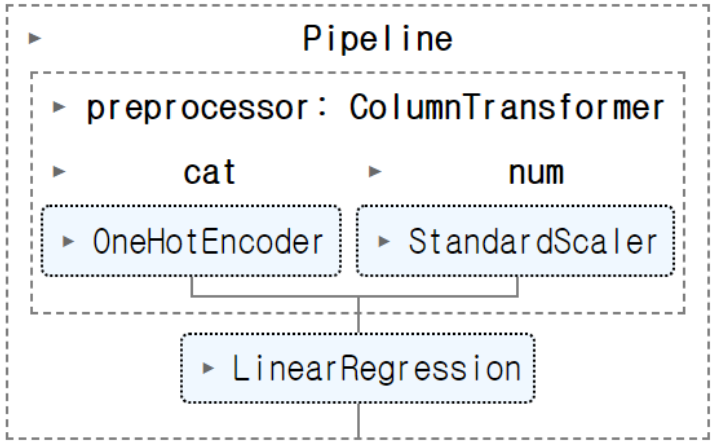


Examples of linearity

https://en.wikipedia.org/wiki/Linearity,
https://ko.wikipedia.org/wiki/%EC%84%A0%ED%98%95%EC%84%B1
https://www.techtarget.com/whatis/definition/linearity

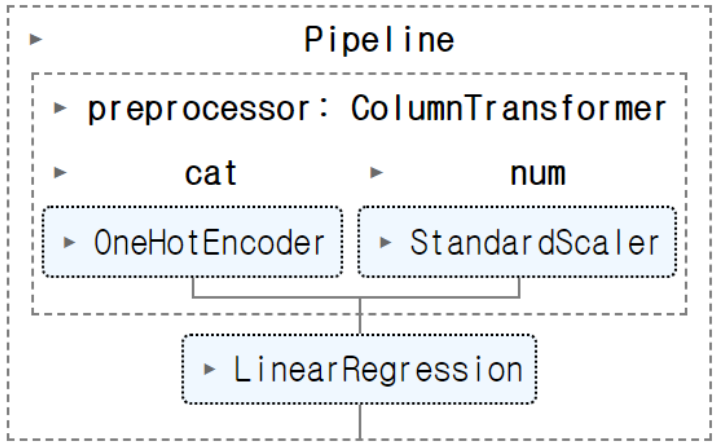
Pipeline + Linear Regression

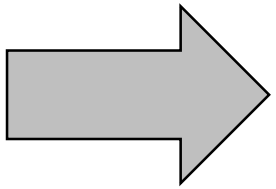
```python
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

from sklearn.ensemble import RandomForestRegressor
from lightgbm import LGBMRegressor
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression

def get_model(method="rf",
              cat_features=["species", "island", "sex"],
              num_features=["bill_length_mm", "bill_depth_mm", "flipper_length_mm"],
              **kwargs):
    # 1-1.categorical feature에 one-hot encoding 적용
    cat_transformer = OneHotEncoder()

    # 1-2.numerical feature는 standard scaler 적용
    num_transformer = StandardScaler()

    # 2. 인자 종류별 전처리 적용
    preprocessor = ColumnTransformer([("cat", cat_transformer, cat_features),
                                      ("num", num_transformer, num_features)])

    # 3. 전처리 후 입력된 방법론 적용
    if method == "rf":
        ml = ("ml", RandomForestRegressor(**kwargs))
    elif method == "lgbm":
        ml = ("ml", LGBMRegressor(**kwargs))
    elif method == "xgb":
        ml = ("ml", XGBRegressor(**kwargs))
    elif method == "lr":
        ml = ("lr", LinearRegression(**kwargs))

    pipeline = Pipeline(steps=[("preprocessor", preprocessor),
                               ml])

    return pipeline
```

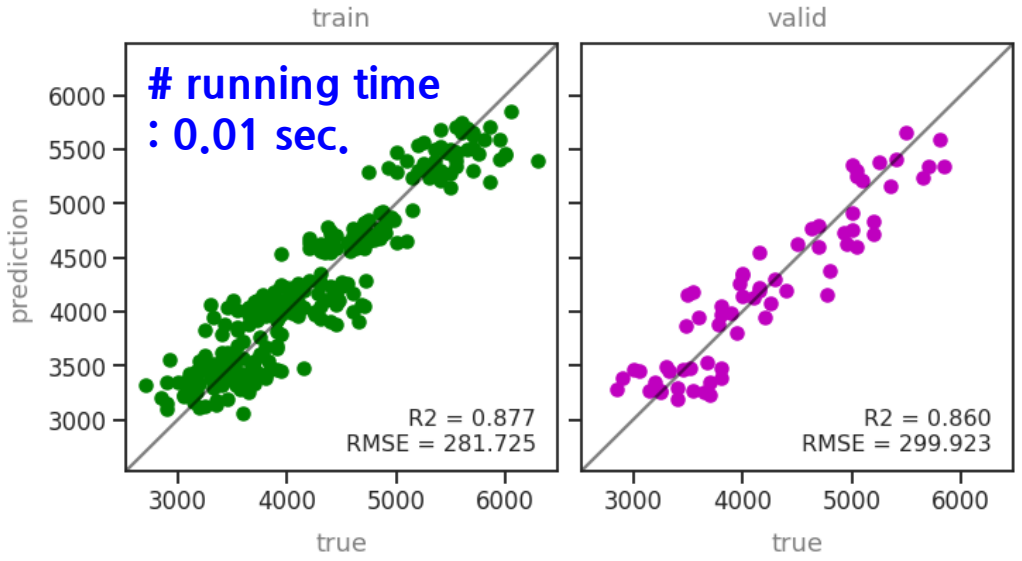


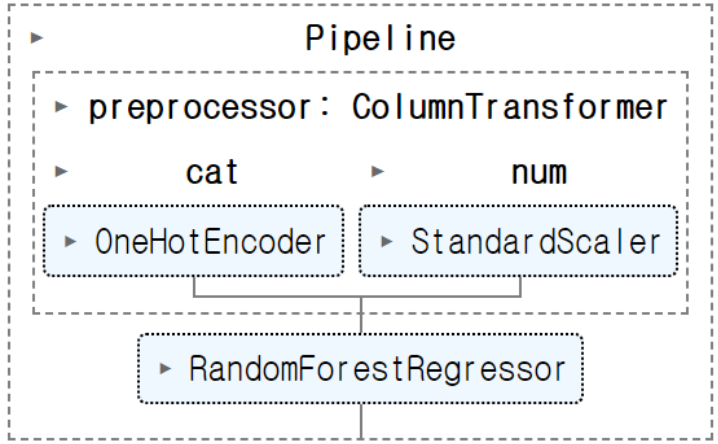
Linear Regression

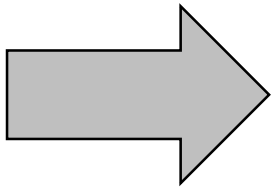
$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

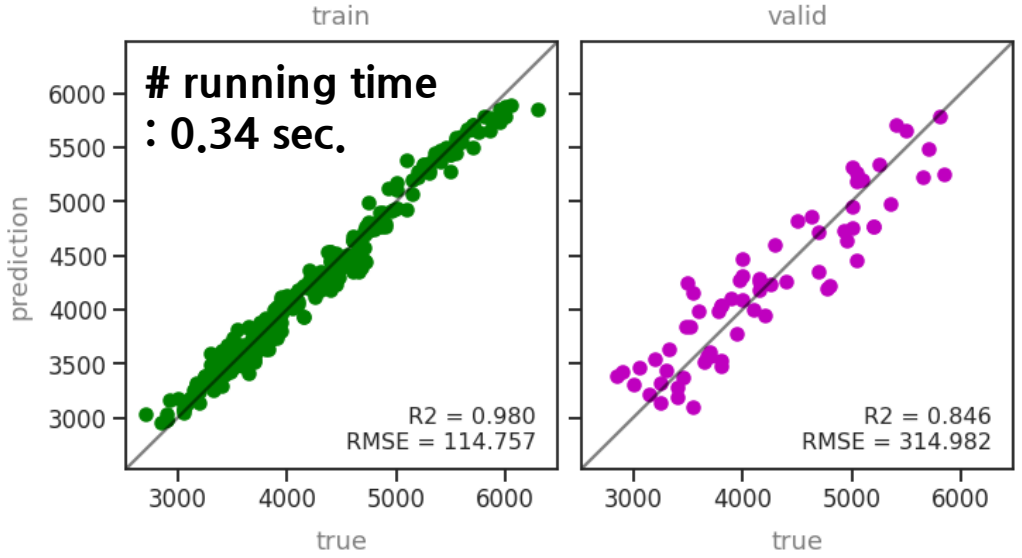






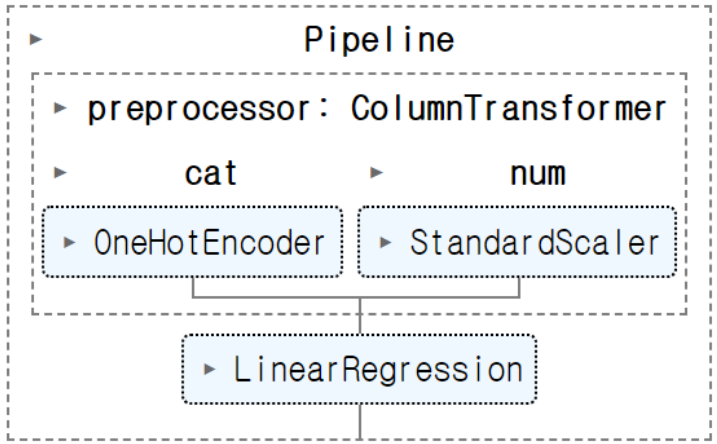






Linear Regression

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$




```
1 X_train.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	sex
48	Adelie	Biscoe	34.5	18.1	187.0	Female
185	Chinstrap	Dream	53.5	19.9	205.0	Male
57	Adelie	Biscoe	41.1	18.2	192.0	Male
168	Chinstrap	Dream	43.2	16.6	187.0	Female
172	Chinstrap	Dream	50.5	18.4	200.0	Female


```
1 model_default["lr"].coef_
```

```
array([-180.35214186, -510.09340248,  690.44554435,  -10.41872756,
         31.37329498,  -20.95456742, -206.01442752,  206.01442752,
        107.00881397,  102.54158725,  259.834919  ])
```

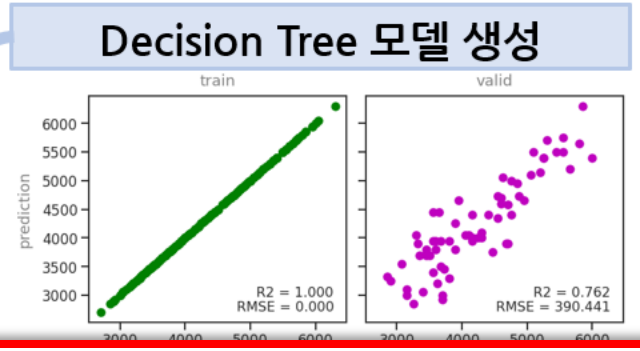
x_i 6개 → 11개 w_i 11개

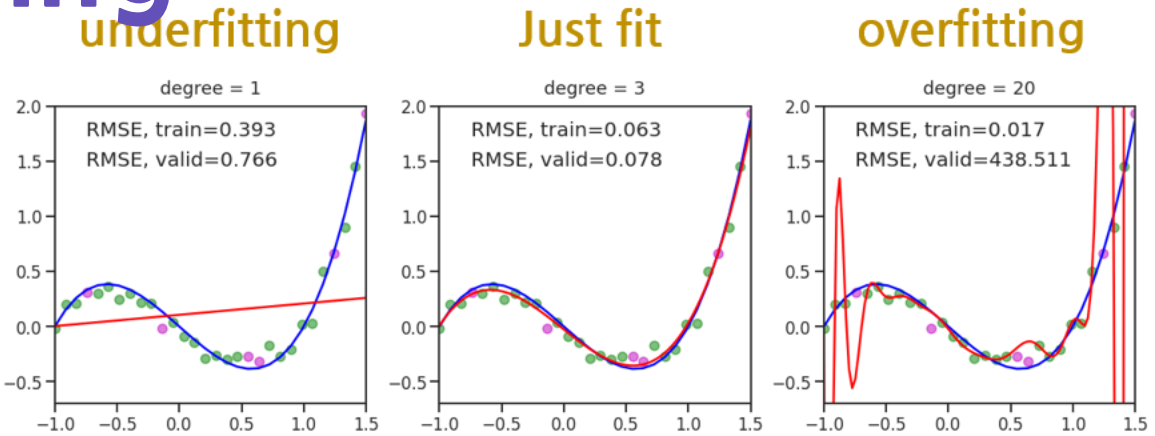
```
1 model_default["lr"].intercept_
```

```
4137.528113385805
```

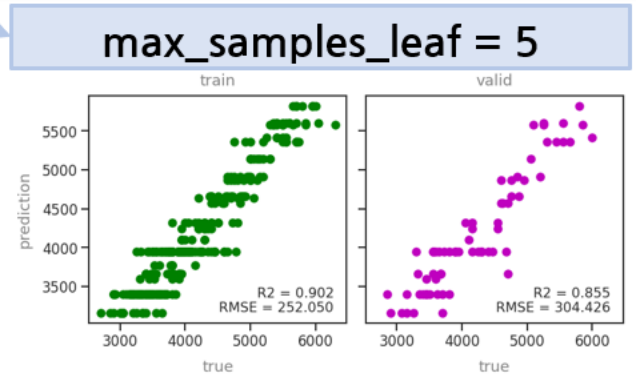
w_0

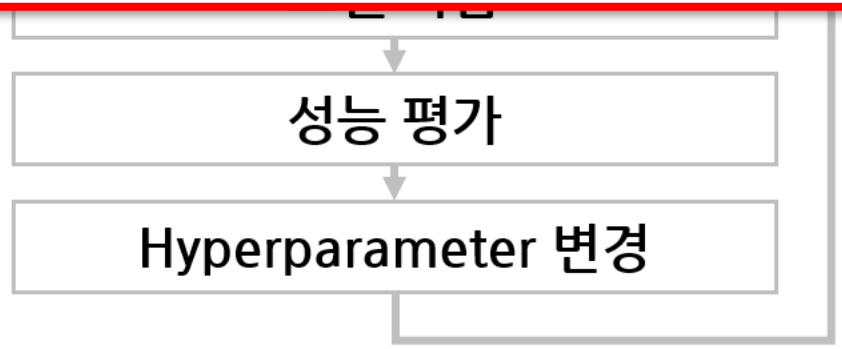
Hyperparameter tuning





NEEDLESS



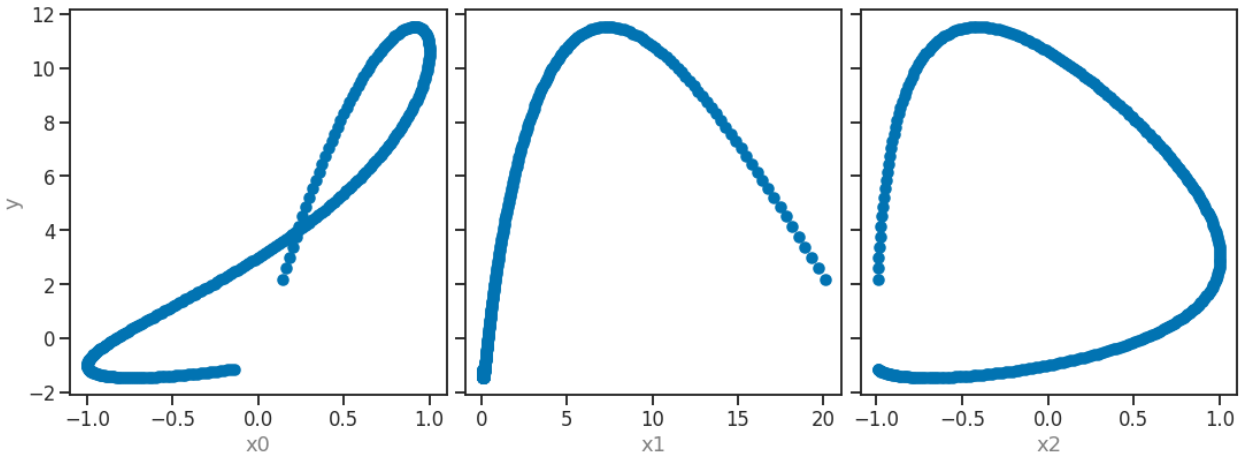


Linear Regression : Difficulty

- 새로운 데이터셋

	x0	x1	x2	y
0	-0.141120	0.049787	-0.989992	-1.137640
1	-0.160890	0.050793	-0.986972	-1.155373
2	-0.180596	0.051819	-0.983557	-1.172660
3	-0.200230	0.052866	-0.979749	-1.189494
4	-0.219784	0.053934	-0.975549	-1.205867
...	...	...	...	...
296	0.219784	18.541287	-0.975549	3.772666
297	0.200230	18.915846	-0.979749	3.391067
298	0.180596	19.297972	-0.983557	2.999491
299	0.160890	19.687817	-0.986972	2.597984
300	0.141120	20.085537	-0.989992	2.186605

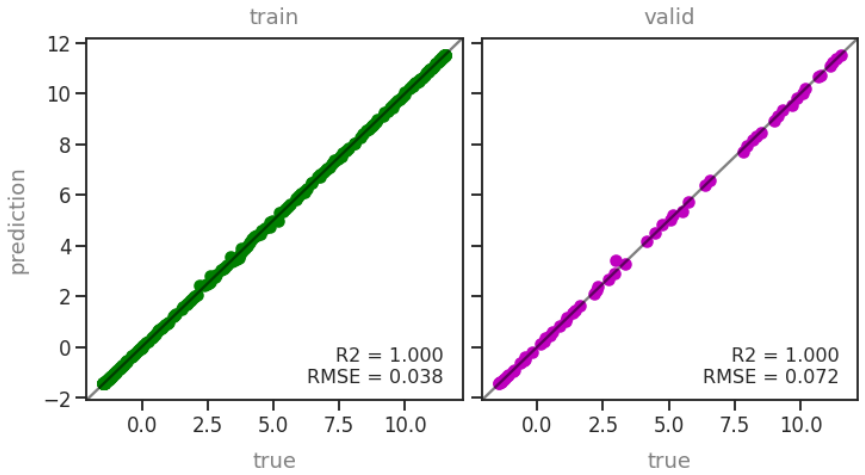
301 rows × 4 columns



train : valid = 8 : 2

Linear Regression :

Random Forest :



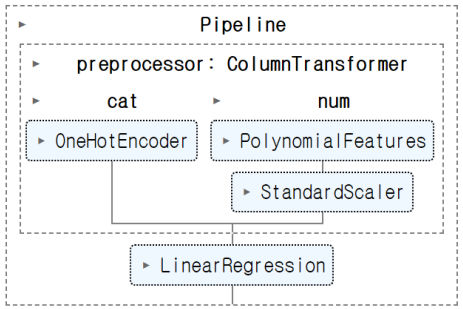
Extending Linear Model : Polynomial

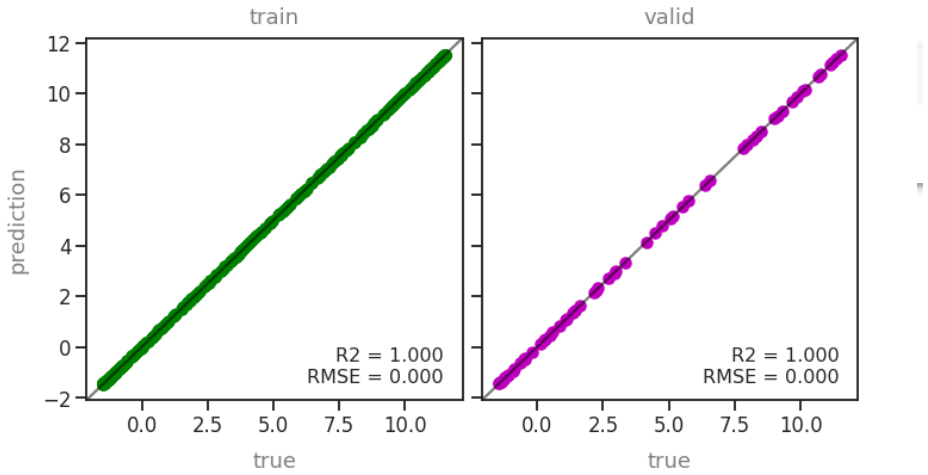
$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2$$

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2$$

```python
from sklearn.preprocessing import PolynomialFeatures

def get_model(method="rf",
              cat_features=["species", "island", "sex"],
              num_features=["bill_length_mm", "bill_depth_mm", "flipper_length_mm"],
              degree=2, **kwargs):
    # 1-1.categorical feature에 one-hot encoding 적용
    cat_transformer = OneHotEncoder()

    # 1-2.numerical feature는 polynomial feature transformer, standard scaler 적용
    num_transformer = Pipeline(steps=[("polynomial", PolynomialFeatures(degree=degree)),
                                      ("scaler", StandardScaler())
                                     ])

    # 2. 인자 종류별 전처리 적용
    preprocessor = ColumnTransformer([("cat", cat_transformer, cat_features),
                                      ("num", num_transformer, num_features)])

    # 3. 전처리 후 입력된 방법론 적용
    if method == "rf":
        ml = ("ml", RandomForestRegressor(**kwargs))
    elif method == "lgbm":
        ml = ("ml", LGBMRegressor(**kwargs))
    elif method == "xgb":
        ml = ("ml", XGBRegressor(**kwargs))
    elif method == "lr":
        ml = ("lr", LinearRegression(**kwargs))

    pipeline = Pipeline(steps=[("preprocessor", preprocessor),
                               ml])

    return pipeline
```

```python
m = get_model(method="lr", degree=2, cat_features=[], num_features=["x0", "x1", "x2"])
m.fit(Xn_train, yn_train)
```





Nonlinearity

- 정의 (Wikipedia)
  - Property of a mathematical relationship(function) that can be graphically represented as a straight line. Closely related to proportionality.
  - 직선처럼 똑바른 도형, 또는 그와 비슷한 성질을 갖는 대상. 이러한 성질을 갖고 있는 변환.

가산성 (additivity) : $f(a+b) = f(a) + f(b)$

동차성 (homogeneity) : $f(ka) = kf(a)$

$f(x) = x^2$

$(a+b)^2 \neq a^2 + b^2$

$(ka)^2 \neq k(a)^2$

- Polynomial Regression
  - Linear Regression + Nonlinearity

scikit-learn website

1. Supervised learning

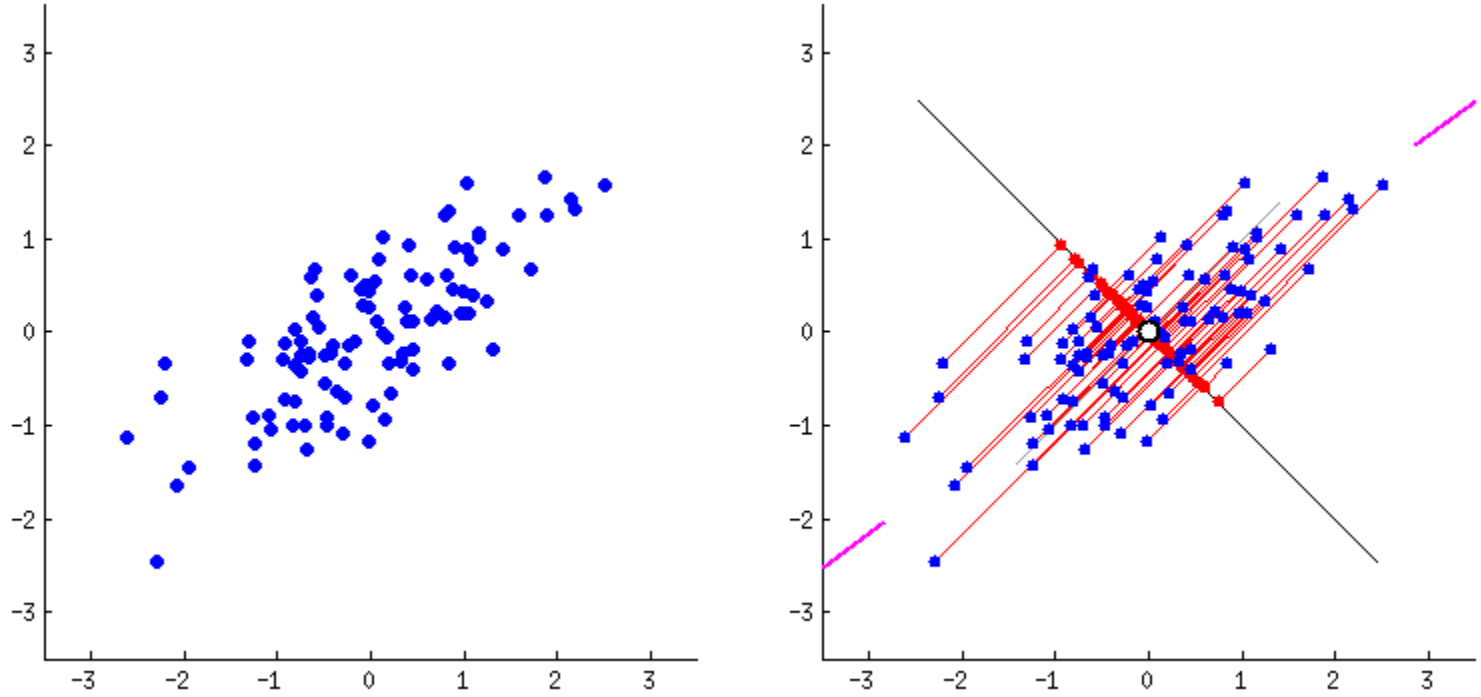
- 1.1. Linear Models
- 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- 1.4. Support Vector Machines
- 1.5. Stochastic Gradient Descent
- 1.6. Nearest Neighbors
- 1.7. Gaussian Processes
- 1.8. Cross decomposition
- 1.9. Naive Bayes
- 1.10. Decision Trees
- 1.11. Ensemble methods
- 1.12. Multiclass and multioutput algorithms
- 1.13. Feature selection
- 1.14. Semi-supervised learning
- 1.15. Isotonic regression
- 1.16. Probability calibration
- 1.17. Neural network models (supervised)

1.1. Linear Models

- 1.1.1. Ordinary Least Squares
- 1.1.2. Ridge regression and classification
- 1.1.3. Lasso
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic-Net
- 1.1.6. Multi-task Elastic-Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
- 1.1.11. Logistic regression
- 1.1.12. Generalized Linear Regression
- 1.1.13. Stochastic Gradient Descent - SGD
- 1.1.14. Perceptron
- 1.1.15. Passive Aggressive Algorithms
- 1.1.16. Robustness regression: outliers and modeling errors
- 1.1.17. Quantile Regression
- 1.1.18. Polynomial regression: extending linear models with basis functions

Linear Transformation

- 주성분 분석 (Principle Component Analysis)
  - 자세한 설명은 다른 교재 참고 (ex. https://bit.ly/3BjEaiE)
  - 주성분 : 데이터가 주로 분포하고 있는 vector

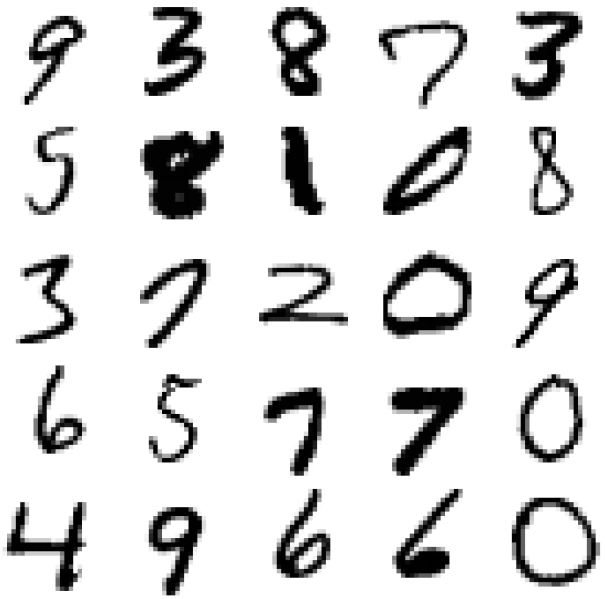


https://jehyunlee.github.io/2020/07/21/Python-DS-21-PCA/

Linear Transformation

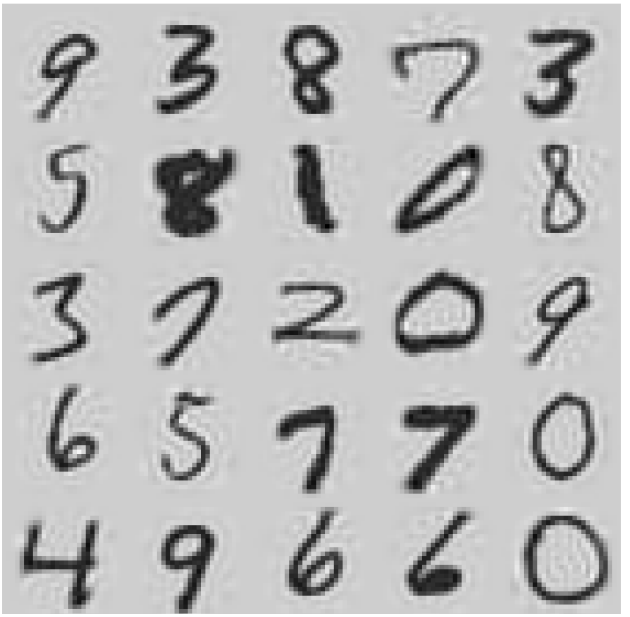
- 주성분 분석 (Principle Component Analysis)
  - 자세한 설명은 다른 교재 참고 (ex. https://bit.ly/3BjEaiE)
  - 주성분 : 데이터가 주로 분포하고 있는 vector
  - 차원 축소 : 데이터의 주요한 내용은 남기면서 데이터를 이루는 feature 수 감소
    - Explained variance ratio : 변수 설명력

Original



784차원 → 154차원
설명력 > 95%
용량 < 20%

Compressed



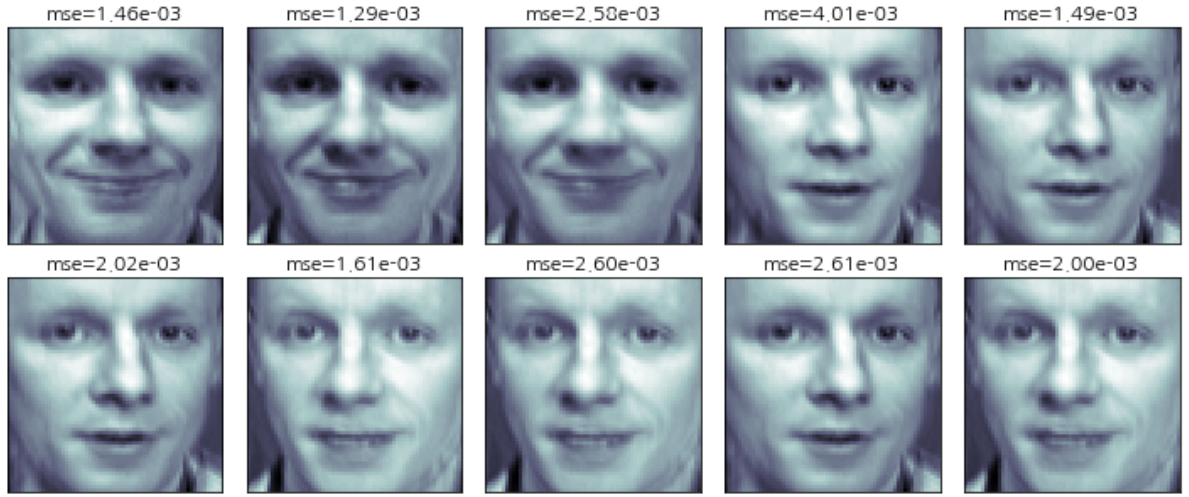
https://jehyunlee.github.io/2020/07/21/Python-DS-21-PCA/

Linear Transformation

- 주성분 분석 (Principle Component Analysis)
  - 자세한 설명은 다른 교재 참고 (ex. https://bit.ly/3BjEaiE)
  - 주성분 : 데이터가 주로 분포하고 있는 vector
  - 데이터 생성 : 주성분을 새롭게 조합하여 새로운 데이터 생성
    - Eigenface

Olivetti face dataset

PC1, PC2만 사용해 역변환

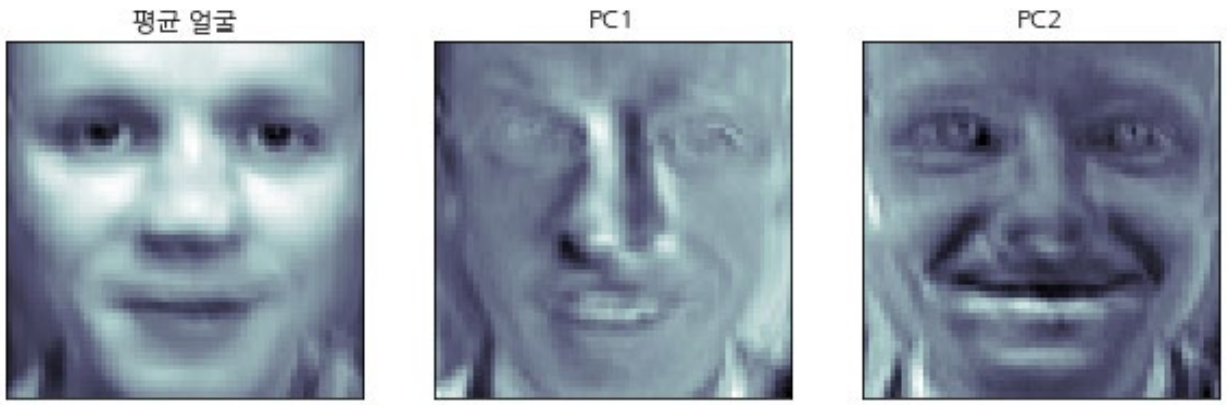


https://jehyunlee.github.io/2020/07/21/Python-DS-21-PCA/

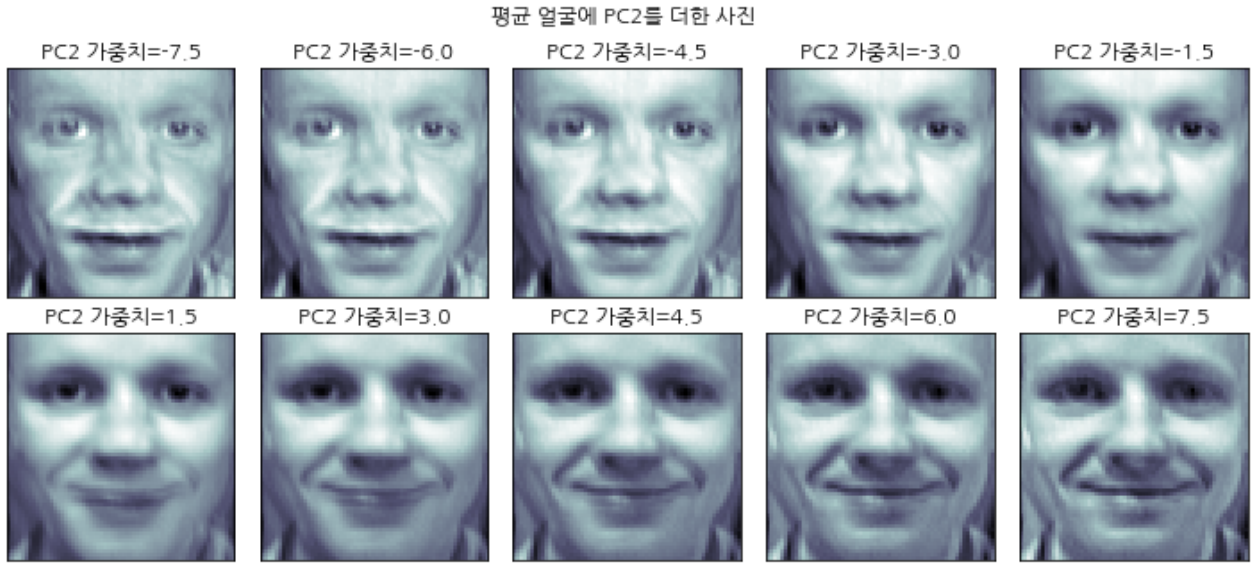
Linear Transformation

- 주성분 분석 (Principle Component Analysis)
  - 자세한 설명은 다른 교재 참고 (ex. https://bit.ly/3BjEaiE)
  - 주성분 : 데이터가 주로 분포하고 있는 vector
  - 데이터 생성 : 주성분을 새롭게 조합하여 새로운 데이터 생성
    - Eigenface

PC1과 PC2 확인

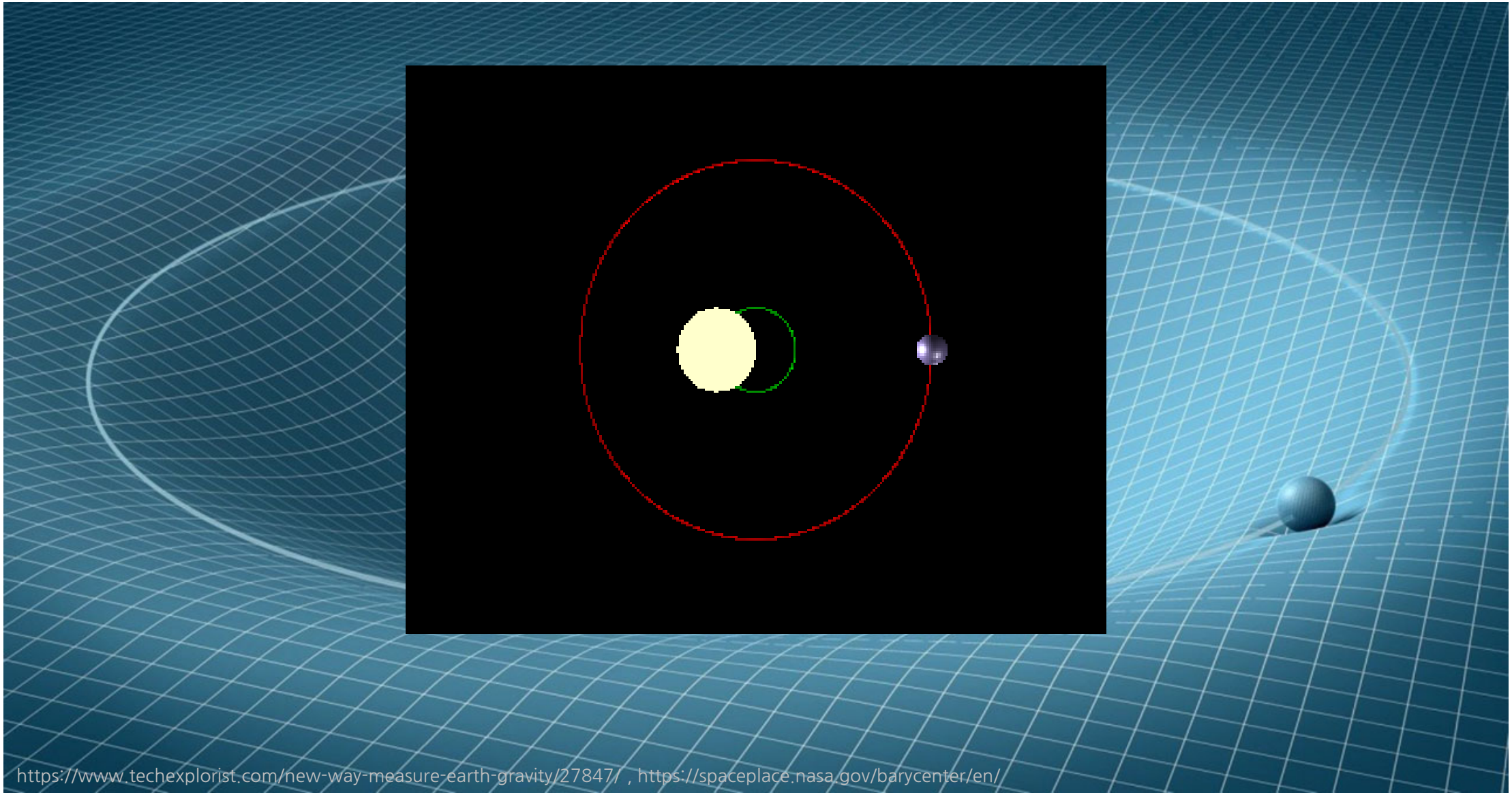


PC2 가중치 변화: 표정 변화



https://jehyunlee.github.io/2020/07/21/Python-DS-21-PCA/

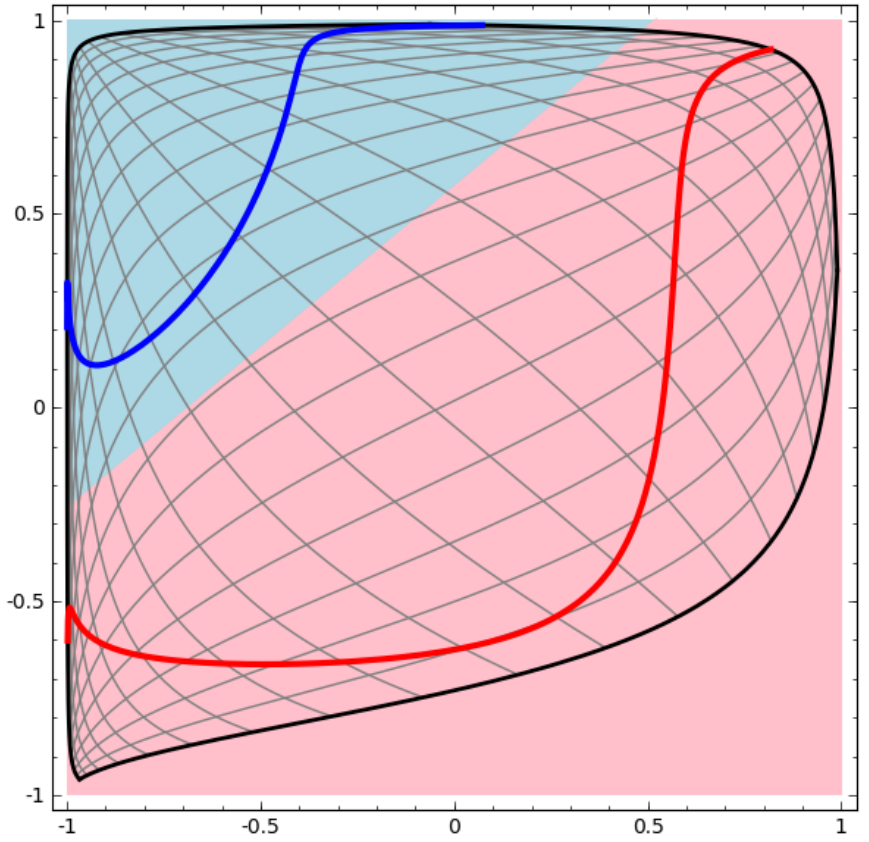
Nonlinear transformation : Kernel Trick

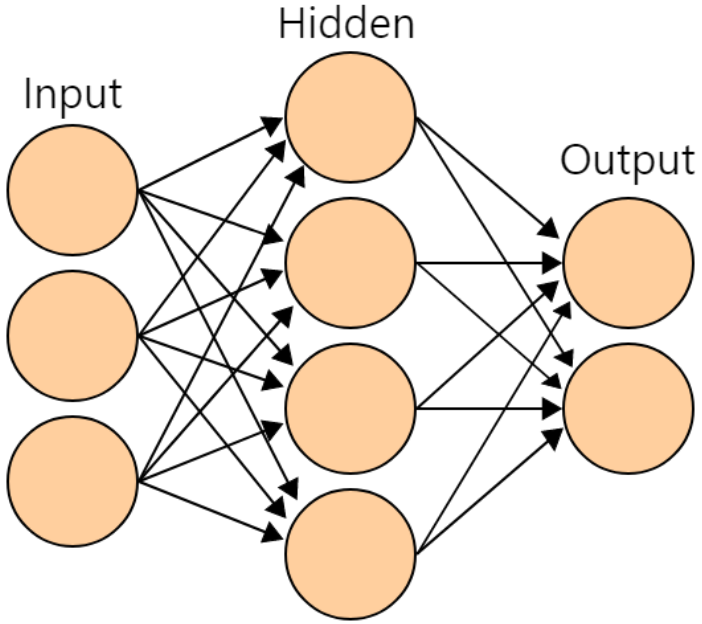


https://www.techexplorist.com/new-way-measure-earth-gravity/27847/ , https://spaceplace.nasa.gov/barycenter/en/

Nonlinear transformation : Kernel Trick

- (비선형) 데이터의 패턴을 쉽게 찾기 위한 데이터 공간 왜곡

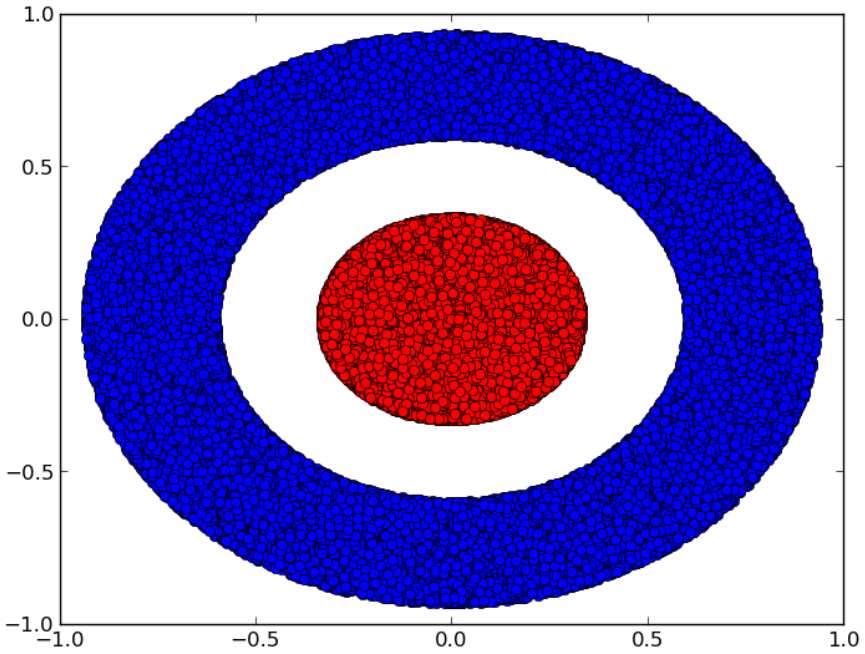


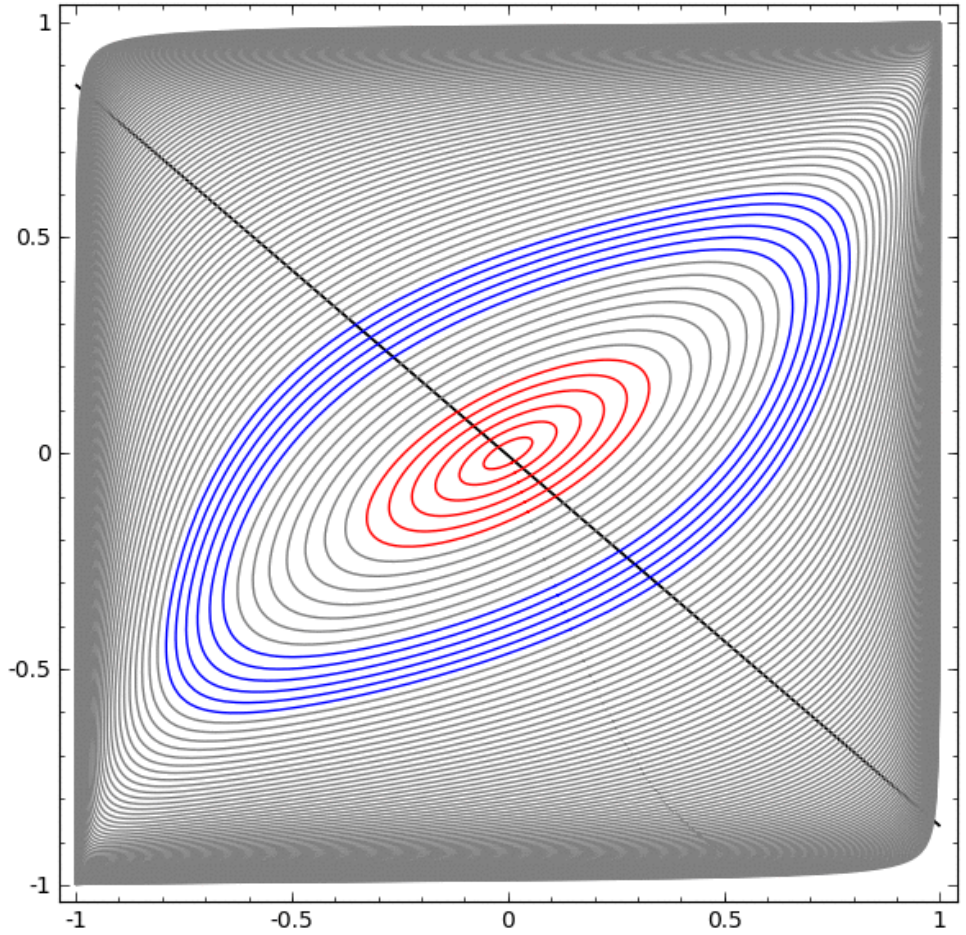


http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Nonlinear transformation : Kernel Trick

- (비선형) 데이터의 패턴을 쉽게 찾기 위한 데이터 공간 왜곡

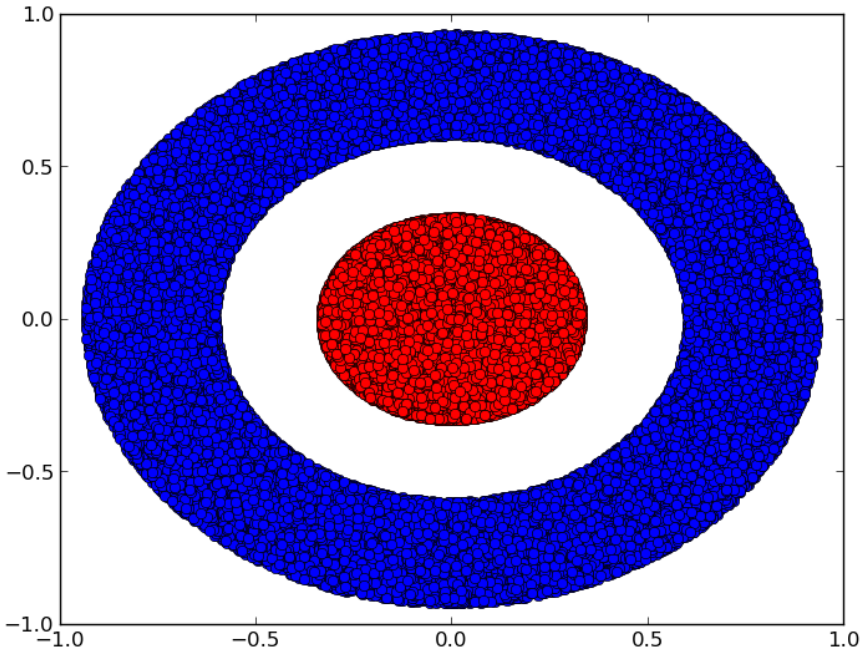


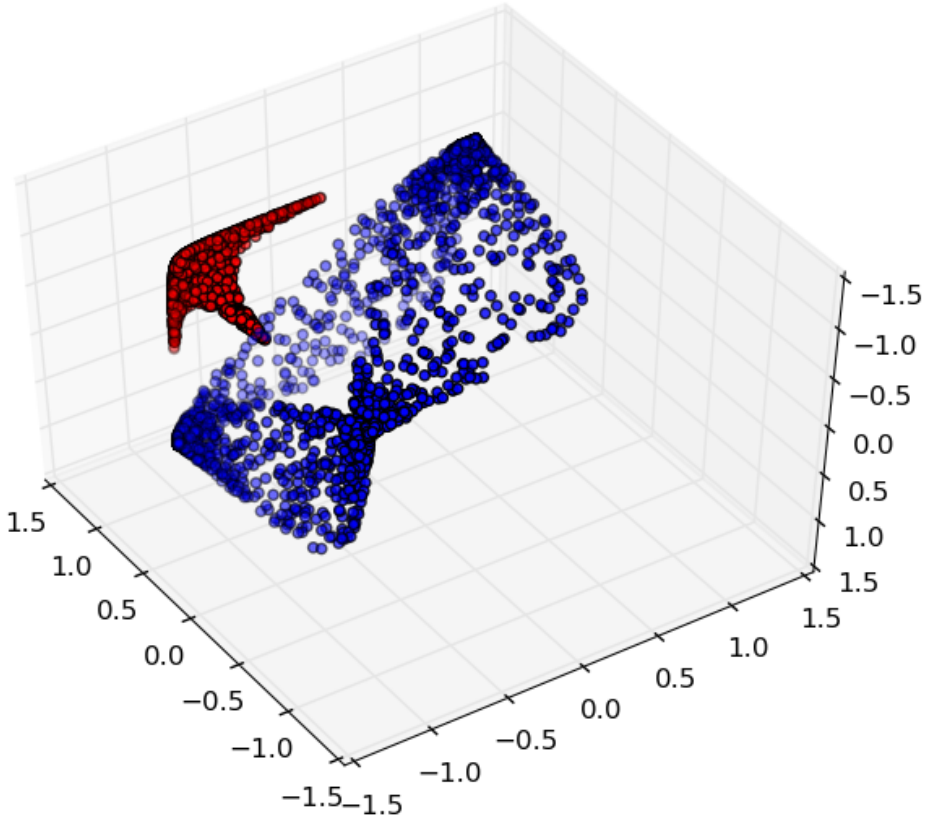


http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Nonlinear transformation : Kernel Trick

- (비선형) 데이터의 패턴을 쉽게 찾기 위한 데이터 공간 왜곡
  - RBF: Radial Basis Function

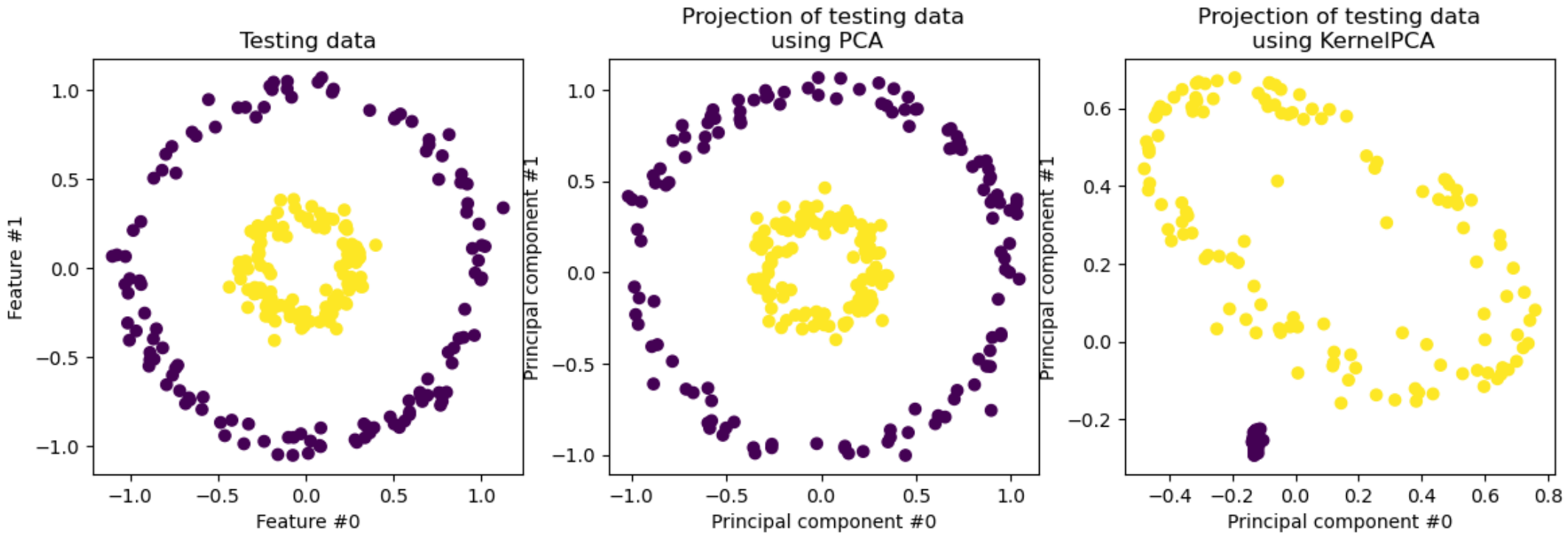




http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Kernel PCA

- (비선형) 데이터의 패턴을 쉽게 찾기 위한 데이터 공간 왜곡
  - RBF: Radial Basis Function

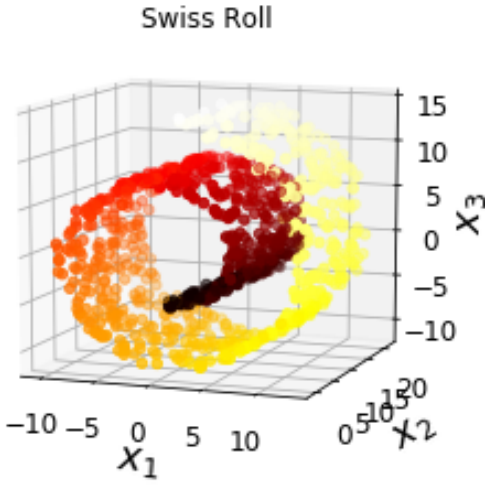


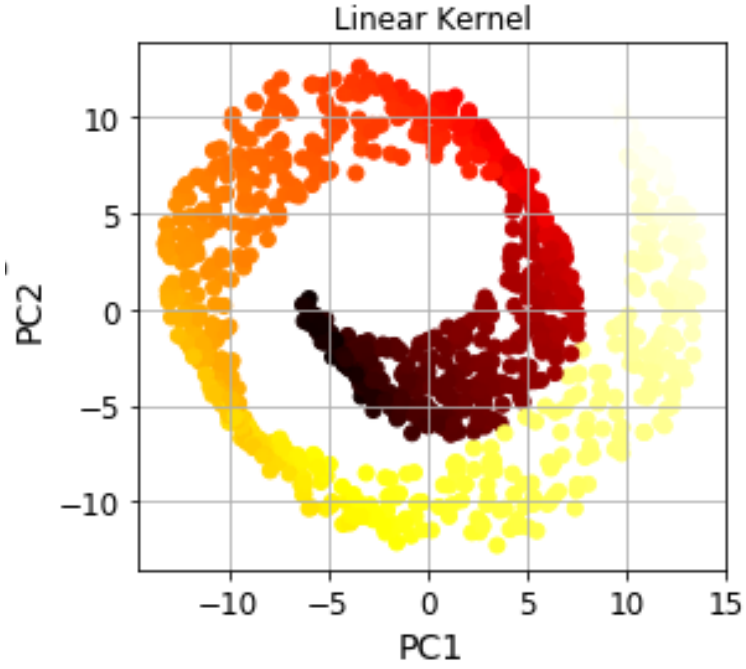
https://scikit-learn.org/stable/auto_examples/decomposition/plot_kernel_pca.html?highlight=linear+kernel

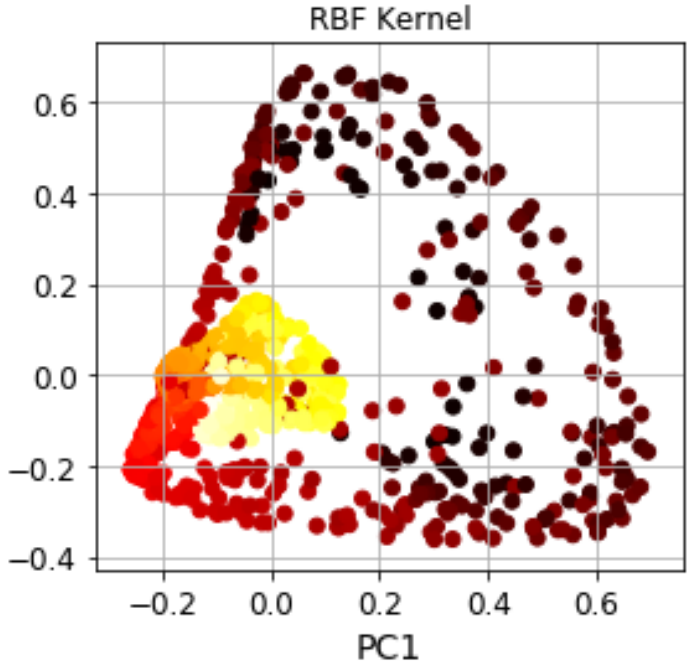
Kernel PCA

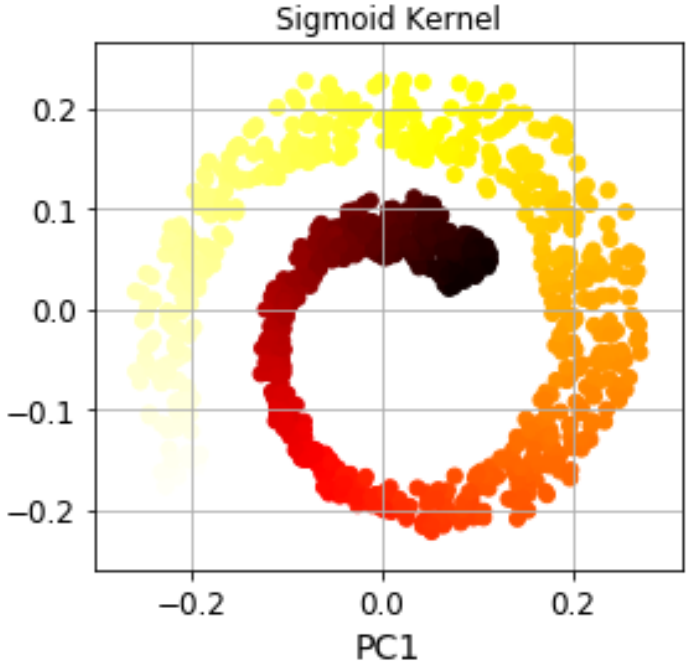
- Regression : Swiss roll

$$y = f(x_1, x_2, x_3)$$



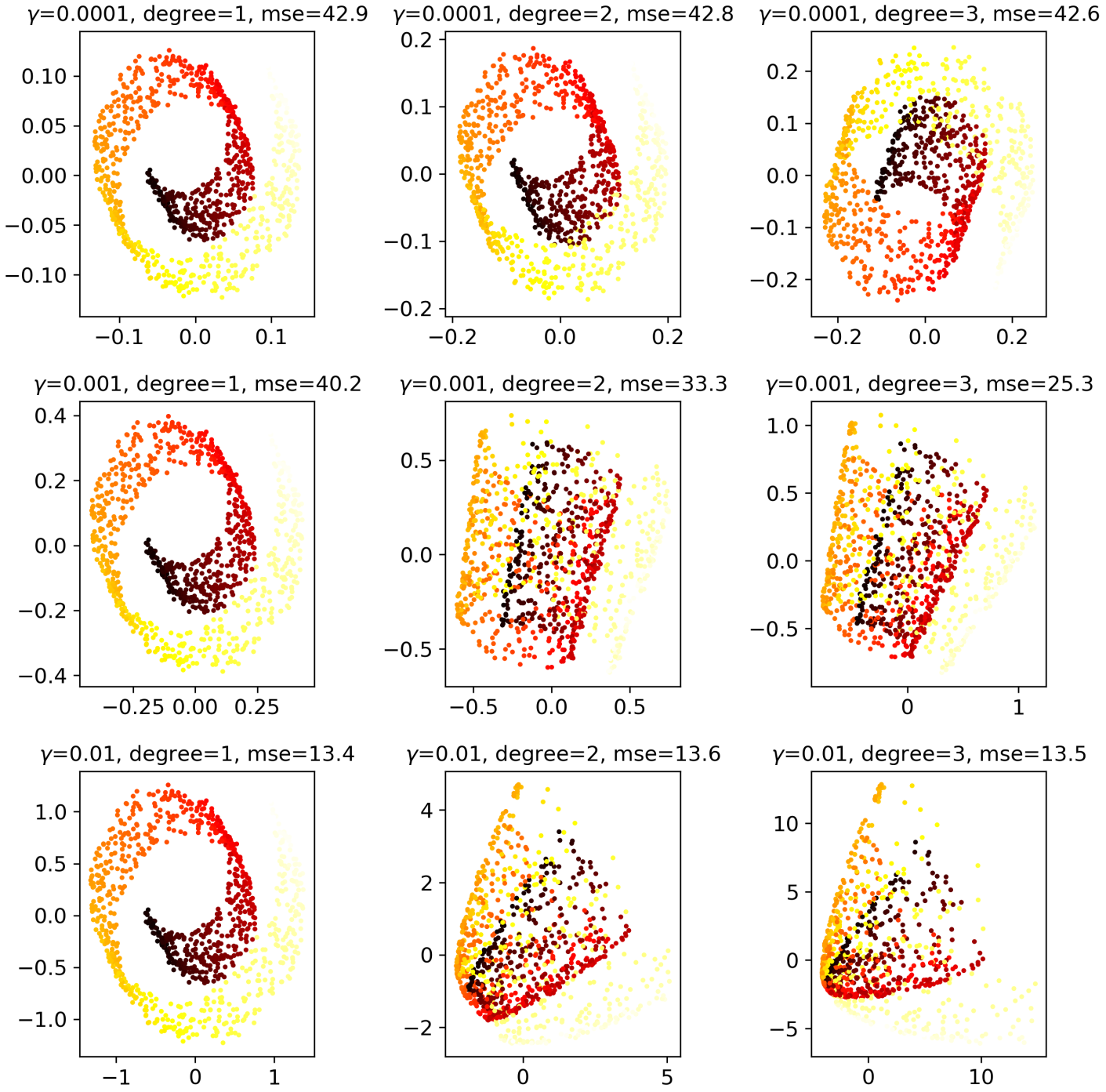






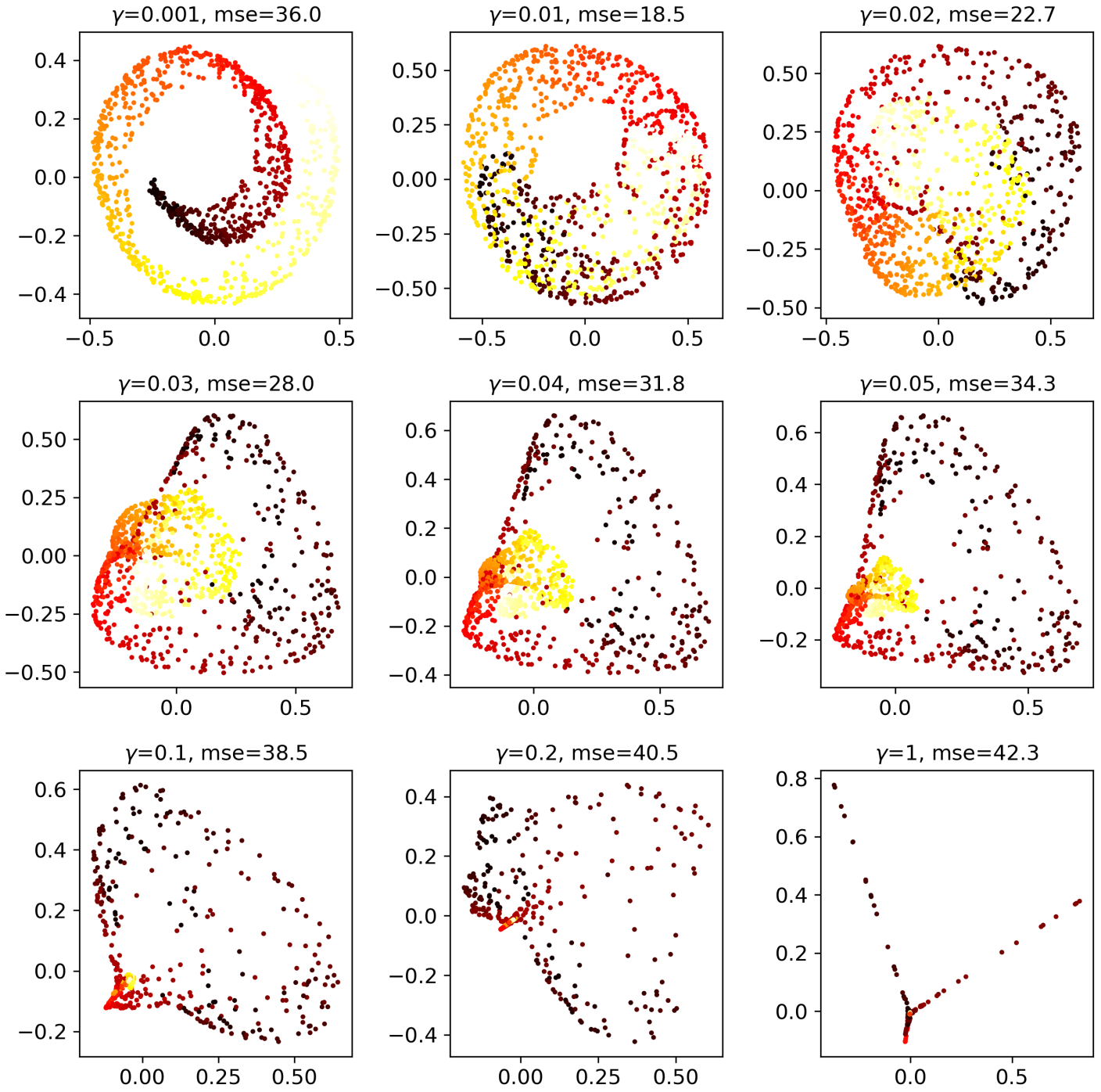
Kernel PCA

- Polynomial Kernel



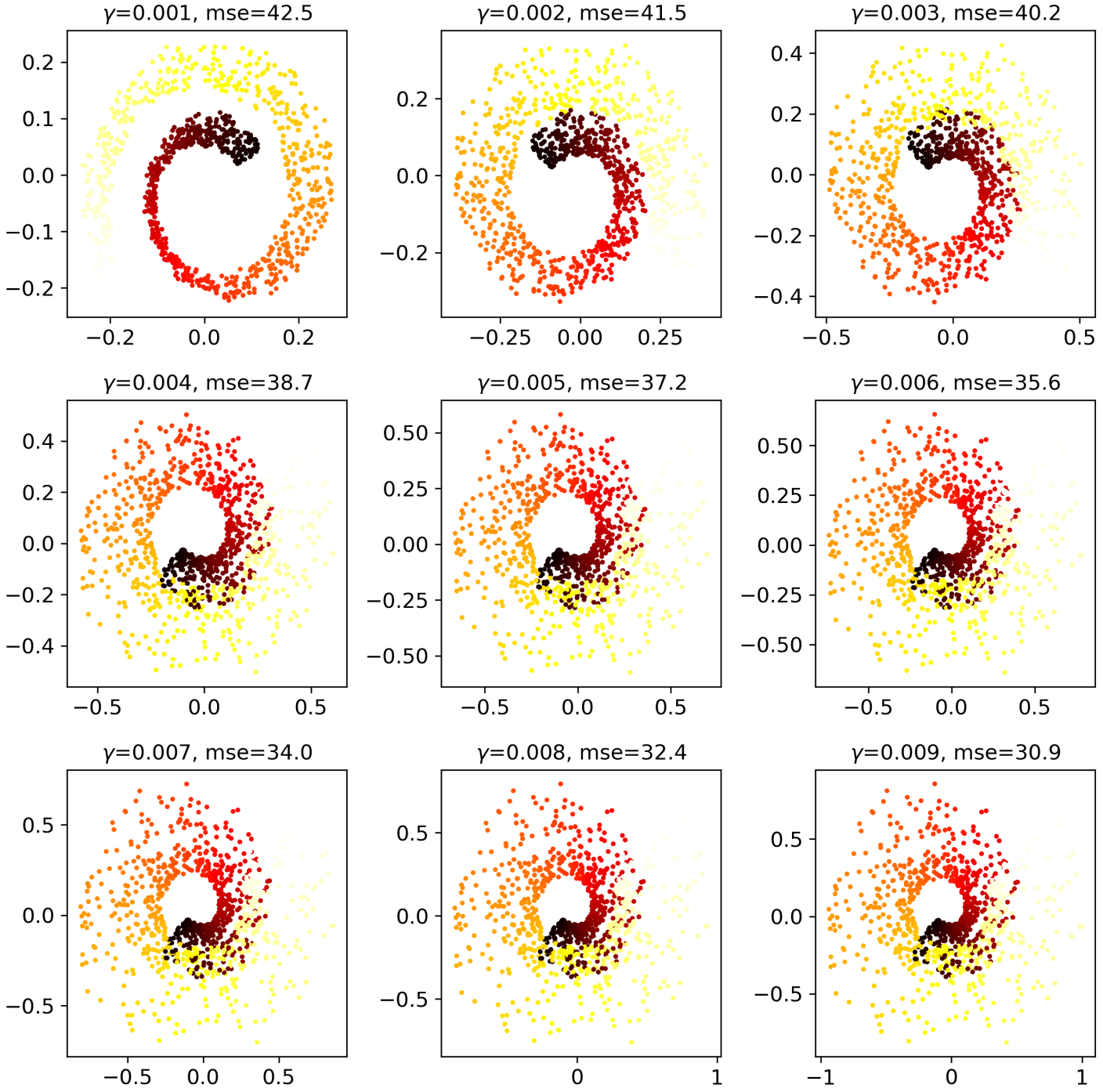
Kernel PCA

- RBF Kernel



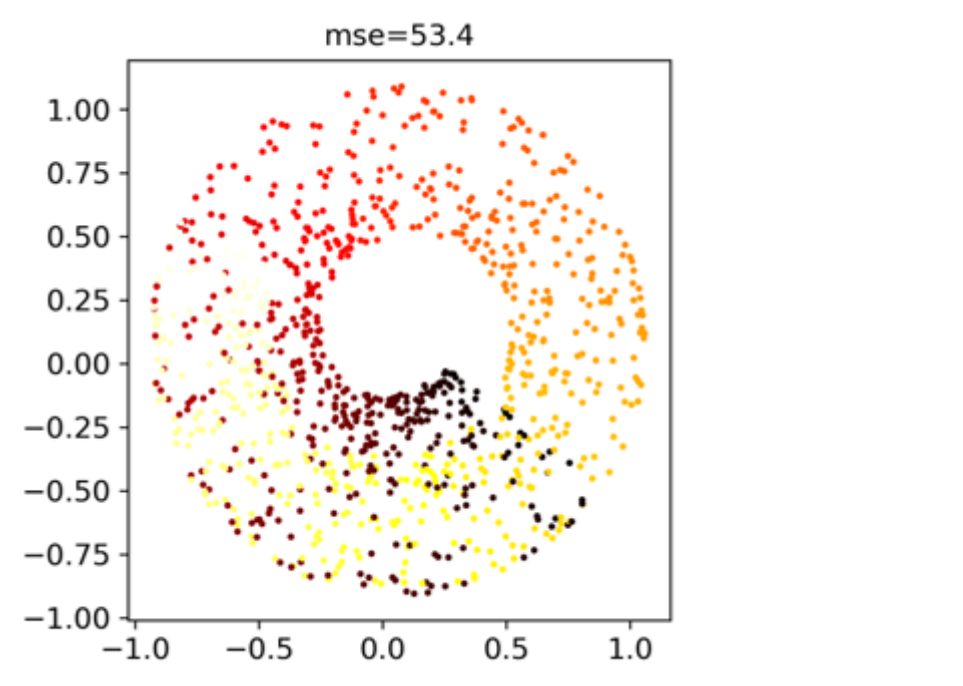
Kernel PCA

- sigmoid Kernel



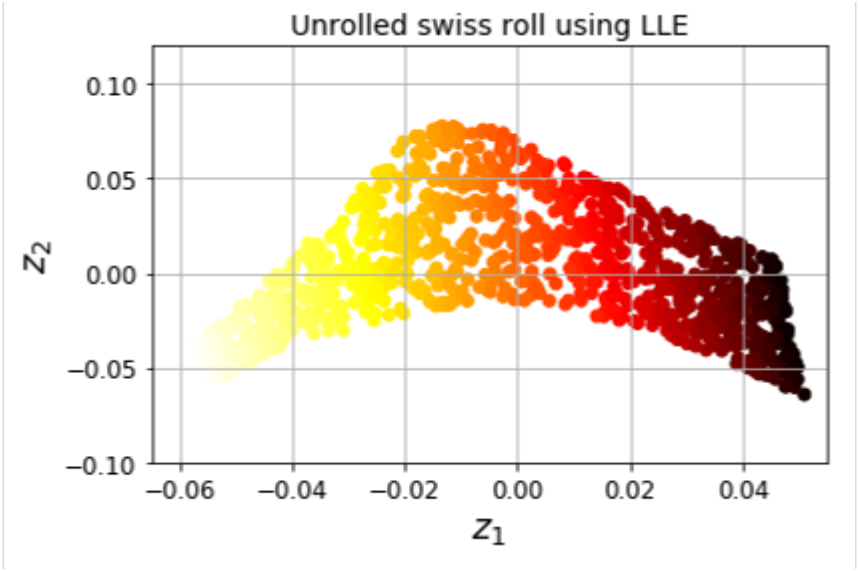
Kernel PCA

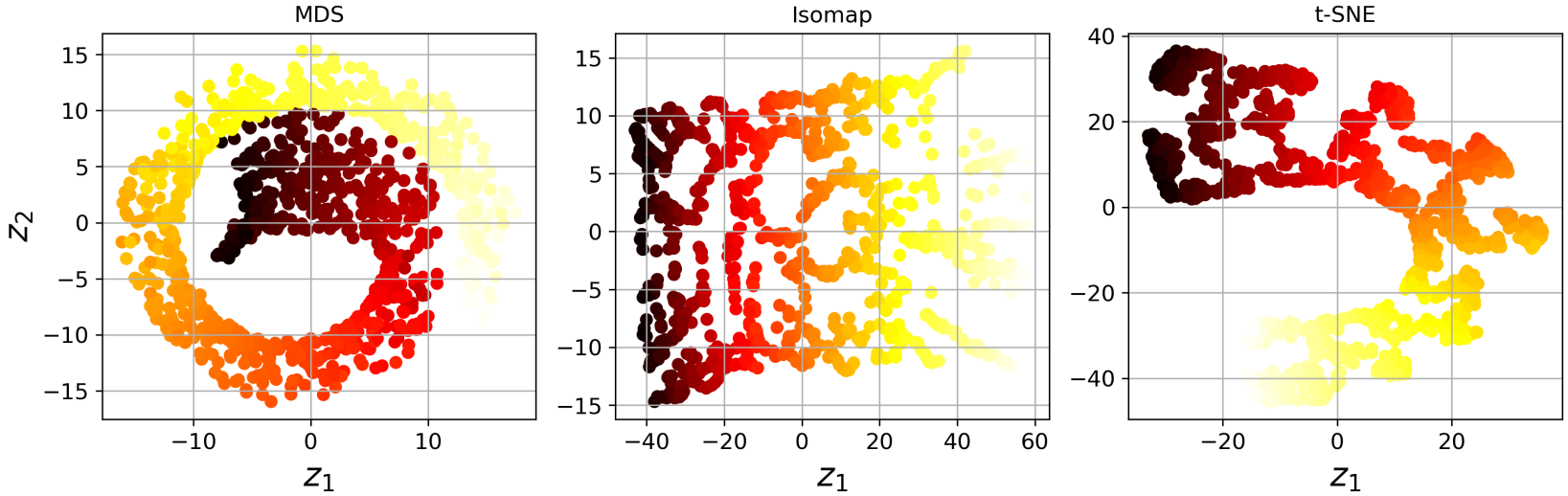
- cosine Kernel



Manifold

- Locally Linear Embedding (2000)
- Multi-dimensional Scaling (1964)
- Isomap (2000)
- t-SNE (2008)





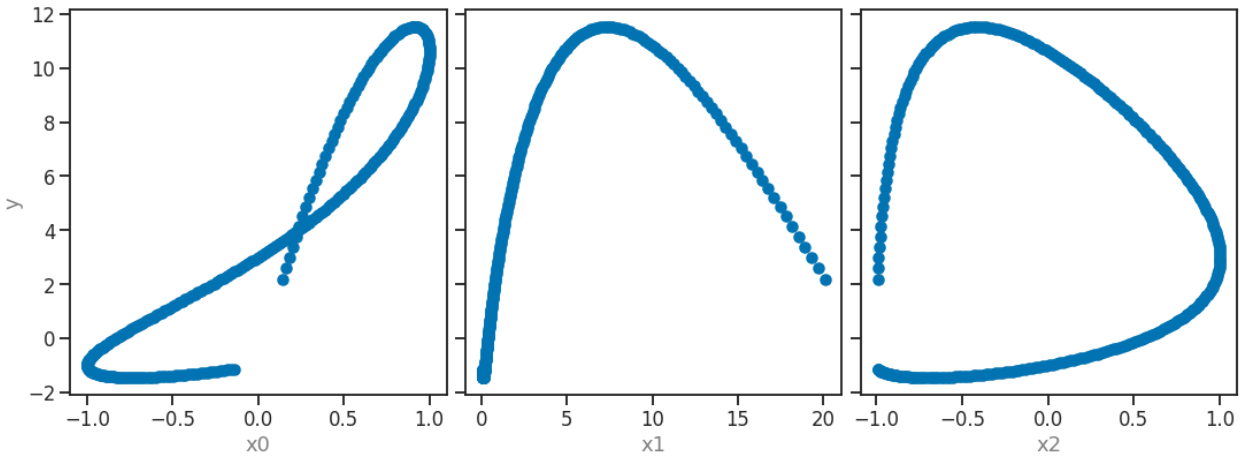
https://lovit.github.io/nlp/representation/2018/09/28/mds_isomap_lle/

아까 잘 안맞던 Linear Regression

- 새로운 데이터셋

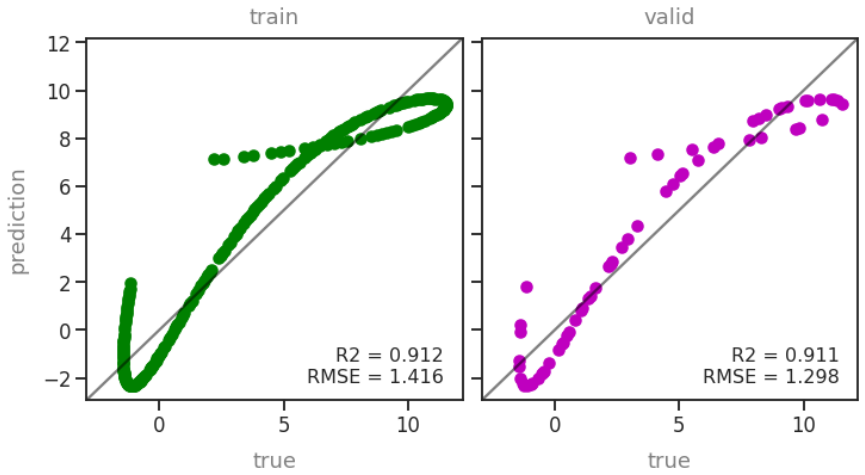
	x0	x1	x2	y
0	-0.141120	0.049787	-0.989992	-1.137640
1	-0.160890	0.050793	-0.986972	-1.155373
2	-0.180596	0.051819	-0.983557	-1.172660
3	-0.200230	0.052866	-0.979749	-1.189494
4	-0.219784	0.053934	-0.975549	-1.205867
...	...	...	...	...
296	0.219784	18.541287	-0.975549	3.772666
297	0.200230	18.915846	-0.979749	3.391067
298	0.180596	19.297972	-0.983557	2.999491
299	0.160890	19.687817	-0.986972	2.597984
300	0.141120	20.085537	-0.989992	2.186605

301 rows × 4 columns



train : valid = 8 : 2

Linear Regression :



Kernel Ridge Regression

Regression is a weighted sum of "kernels" 3

$$d(\underline{x}) = \underline{\phi}^T(\underline{x})\,\underline{w} = \underline{\phi}^T(\underline{x})\left(\underline{\Phi}^T\underline{\Phi} + \lambda \underline{I}\right)^{-1}\underline{\Phi}^T\underline{d}$$

PxP

Matrix identity: $(\underline{\Phi}^T\underline{\Phi}+\lambda\underline{I})^{-1}\underline{\Phi}^T = \underline{\Phi}^T(\underline{\Phi}\,\underline{\Phi}^T+\lambda\underline{I})^{-1}$ (activity)

Thus $d(\underline{x}) = \underline{\phi}^T(\underline{x})\,\underline{\Phi}^T(\underline{\Phi}\,\underline{\Phi}^T+\lambda\underline{I})^{-1}\underline{d}$

N x N

Note: $\left[\underline{\Phi}\,\underline{\Phi}^T\right]_{i,j} = \underline{\phi}^T(\underline{x}^i)\,\underline{\phi}(\underline{x}^j)$

$\left[\underline{\phi}^T(\underline{x})\,\underline{\Phi}^T\right]_j = \underline{\phi}^T(\underline{x})\,\underline{\phi}(\underline{x}^j)$

Define "Kernel" $K(\underline{u},\underline{v}) = \underline{\phi}^T(\underline{u})\,\underline{\phi}(\underline{v})$

Let $\underline{\alpha} = [\alpha_1 \cdots \alpha_N]^T = (\underline{\Phi}\,\underline{\Phi}^T+\lambda\underline{I})^{-1}\underline{d}$

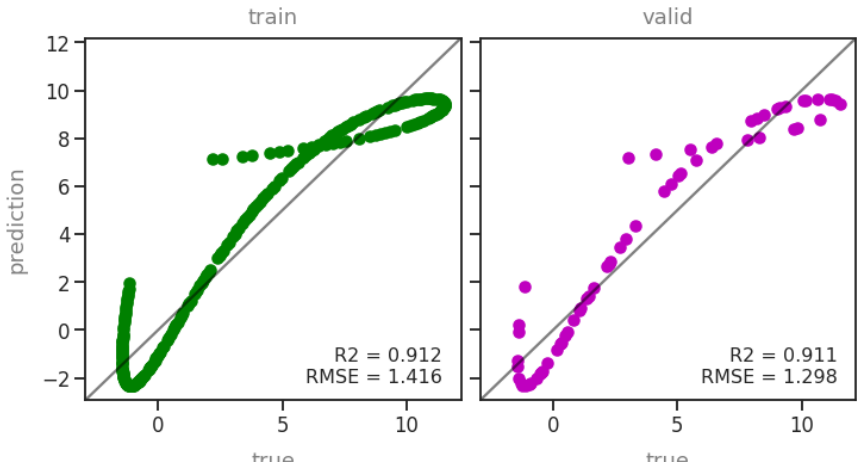
$$d(\underline{x}) = \sum_{i=1}^{N} \alpha_i\, \underline{\phi}^T(\underline{x})\,\underline{\phi}(\underline{x}^i) = \sum_{i=1}^{N} \alpha_i K(\underline{x},\underline{x}^i)$$

Kernel Ridge + HP optimization

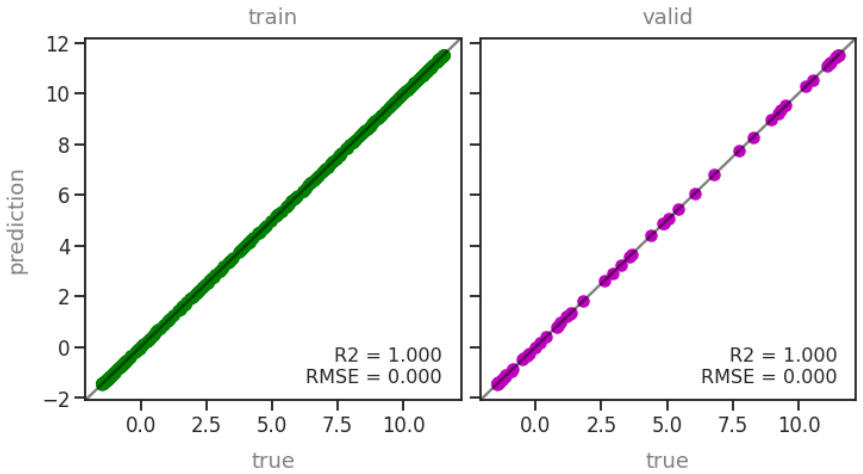
- Ridge Regression : 과적합 해소를 위해 계수의 제곱 합을 손실함수에 포함
- Kernel Ridge : 커널을 통해 변환한 데이터를 Ridge Regression에 적용

```python
from sklearn.kernel_ridge import KernelRidge

def get_model(method="rf",
              cat_features=["species", "island", "sex"],
              num_features=["bill_length_mm", "bill_depth_mm", "flipper_length_mm"],
              degree=2, **kwargs):
    # 1-1.categorical feature에 one-hot encoding 적용
    cat_transformer = OneHotEncoder()

    # 1-2.numerical feature는 polynomial feature transformer, standard scaler 적용
    num_transformer = Pipeline(steps=[("polynomial", PolynomialFeatures(degree=degree)),
                                      #   ("scaler", StandardScaler())
                                       ])

    # 2. 인자 종류별 전처리 적용
    preprocessor = ColumnTransformer([("cat", cat_transformer, cat_features),
                                      ("num", num_transformer, num_features)])

    # 3. 전처리 후 입력된 방법론 적용
    if method == "rf":
        ml = ("ml", RandomForestRegressor(**kwargs))
    elif method == "lgbm":
        ml = ("ml", LGBMRegressor(**kwargs))
    elif method == "xgb":
        ml = ("ml", XGBRegressor(**kwargs))
    elif method == "lr":
        ml = ("ml", LinearRegression(**kwargs))
    elif method == "kr":
        ml = ("ml", KernelRidge(**kwargs))

    pipeline = Pipeline(steps=[("preprocessor", preprocessor),
                               ml])

    return pipeline
```

Linear Regression :



Kernel Ridge :



https://omicro03.medium.com/linear-regression-63164cd2e51

Homework

- 핸즈온 머신러닝 8. 차원 축소 : https://bit.ly/3qJkKyS
- MNIST에 PCA 적용, t-SNE 시각화