CSC 461: Machine Learning
Fall 2024

Feature transformation

Prof. Marco Alvarez, Computer Science
University of Rhode Island

Data transformation

‣ Definition

- converting or mapping data from its raw form into a format that better suits the learning algorithms

- crucial step in the machine learning pipeline that can significantly improve model performance

‣ Why?

- meet algorithm assumptions

- handle non-linear relationships

- scale features appropriately

- handle different data types

Scaling

```
[[ -91   25  -85  -36]
 [  13   23   56   57]
 [  13  -92  -27 -100]
 [ -60   64   15  -84]
 [   0   39  -46  -12]]
```

‣ **Min-max scaling**

$$\tilde{x} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
MinMaxScaler()
[[0.         0.75       0.         0.40764331]
 [1.         0.73717949 1.         1.        ]
 [1.         0.         0.41134752 0.        ]
 [0.29807692 1.         0.70921986 0.10191083]
 [0.875      0.83974359 0.27659574 0.56050955]]
```

‣ **Standardization**

$$\tilde{x} = \frac{x - \mu}{\sigma}$$

```
StandardScaler()
[[-1.54757414  0.24479055 -1.38466458 -0.01789141]
 [ 0.89102753  0.20770107  1.50346717  1.64600941]
 [ 0.89102753 -1.92494389 -0.19663876 -1.16294143]
 [-0.82068325  0.96803537  0.66365581 -0.87667893]
 [ 0.58620232  0.5044169  -0.58581963  0.41150235]]
```

Categorical transformations

‣ One-hot encoding

- converts categorical variables into binary vectors

- each category becomes a column, only one column contains 1, the rest are 0

```
[['red']           [[0. 0. 1.]
 ['green']          [0. 1. 0.]
 ['blue']           [1. 0. 0.]
 ['red']            [0. 0. 1.]
 ['green']          [0. 1. 0.]
 ['blue']           [1. 0. 0.]
 ['red']            [0. 0. 1.]
 ['green']          [0. 1. 0.]
 ['blue']]          [1. 0. 0.]]
```
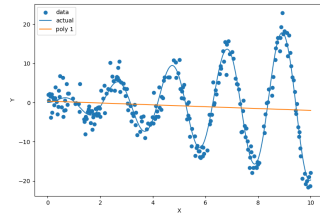
‣ Label encoding

- converts categorical values into numerical by assigning a unique integer to each category

```
[['red']           [[2]
 ['green']          [1]
 ['blue']           [0]
 ['red']            [2]
 ['green']          [1]
 ['blue']           [0]
 ['red']            [2]
 ['green']          [1]
 ['blue']]          [0]]
```

Non-linear transformations

‣ Motivation

- data is not always "linear"

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \quad \xrightarrow{\mathbf{\Phi}} \quad \mathbf{z} = \begin{bmatrix} \Phi_1(\mathbf{x}) \\ \vdots \\ \Phi_{\tilde{d}}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} z_1 \\ \vdots \\ z_{\tilde{d}} \end{bmatrix}$$

input space $\mathcal{X} = \mathbb{R}^d$ feature space $\mathcal{Z} = \mathbb{R}^{\tilde{d}}$
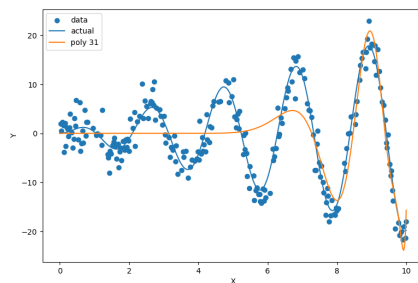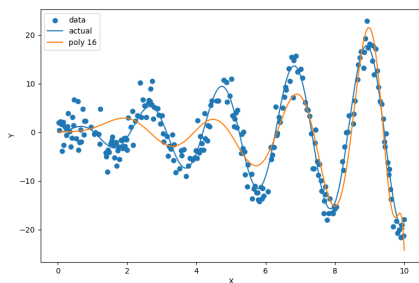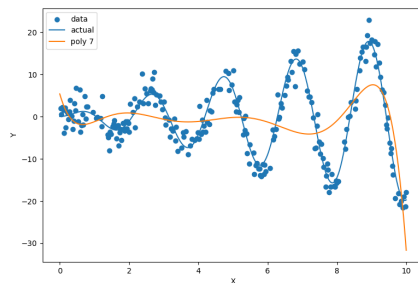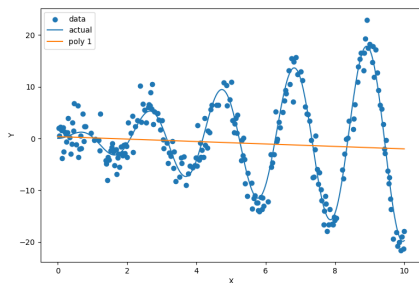

Non-linear transformations

‣ k-th order polynomial features on one variable
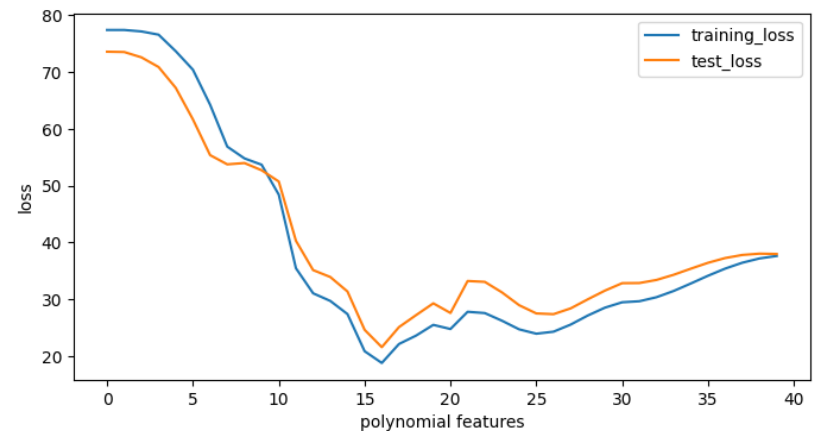
$$\mathbf{x} = [x] \qquad \mathbf{z} = \begin{bmatrix} x^0 \\ x^1 \\ \vdots \\ x^k \end{bmatrix}$$

```
PolynomialFeatures(degree=4)
[[   7]      [[   1.       7.      49.       343.      2401.]
 [  22]       [   1.      22.     484.     10648.    234256.]
 [ -38]       [   1.     -38.    1444.    -54872.   2085136.]
 [ -67]       [   1.     -67.    4489.   -300763.  20151121.]
 [   0]]      [   1.       0.       0.         0.         0.]]
```

Evaluating different degrees

Non-linear transformations

▸ k-th order polynomial features on two variables

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix} \quad \text{degree k=2}$$

```
            PolynomialFeatures(degree=2)
[[ 77  79]       [[   1.    77.    79.  5929.  6083.  6241.]
 [−46 −23]        [   1.   −46.   −23.  2116.  1058.   529.]
 [ 97 −87]        [   1.    97.   −87.  9409. −8439.  7569.]
 [ 53  26]        [   1.    53.    26.  2809.  1378.   676.]
 [ 41  −8]]       [   1.    41.    −8.  1681.  −328.    64.]]
```

Non-linear transformations

▸ k-th order polynomial features on d variables

- "all polynomial combinations of the features with degree less than or equal to the specified degree"

▸ Issues:

- be aware of computational cost

- be aware of overfitting

▸ Alternatives?

- use other non-linear functions: logarithmic, power, etc.

Data transformation

▸ Best Practices

- transform after splitting data

- fit transformers on **training data only**

- handle missing values before transformation and use appropriate imputation

- use pipelines to ensure reproducibility and to prevent data leakage

Embeddings

▸ Vector representations

- embeddings transform discrete data (words, images, audio code) into continuous vector spaces, capturing semantic and syntactic meaning

- key properties of embedding spaces:
  - similar items cluster together
  - meaningful vector arithmetic (e.g., king - man + woman ≈ queen)
  - fixed dimensionality (typically 256-4096 dimensions)

▸ Foundation models

- large models trained on massive datasets

- can produce high-quality embeddings that capture complex semantic and syntactic relationships

▸ Applications

- similarity search, classification, generation, machine translation