Feature engineering

Taehoon Ko (taehoonko@snu.ac.kr)

Terminology: 변수

- **현상들을 설명/표현하는 요소**
- **Variable, Feature, Attribute, Factor, Field, Column, …**

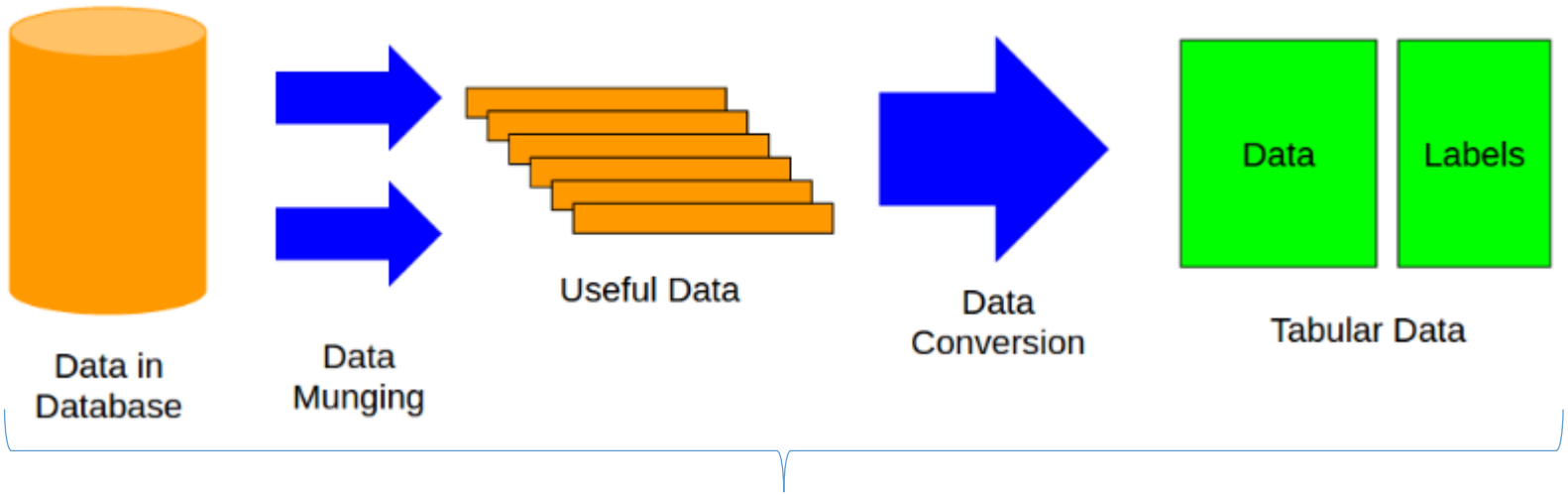
- Predictor variables (예측변수)
- Input variables (입력변수)
- Independent variables (독립변수)

- Target variables (타겟변수)
- Output variables (출력변수)
- Dependent variables (종속변수)

id	X_1	X_2	…	X_p	Y
1	x_{11}	x_{12}	…	$x_{1,p}$	y_1
2	x_{21}	x_{22}	…	$x_{2,p}$	y_2
…	…	…	…	…	…
n	$x_{n,1}$	$x_{n,2}$	…	$x_{n,p}$	y_n

Converting the data to an analyzable form

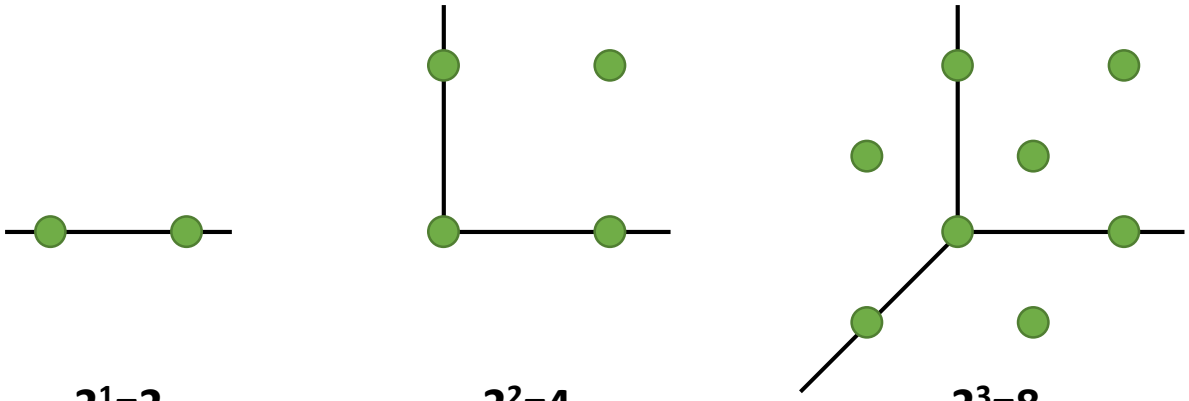
- Before applying the machine learning models, the data must be converted to a tabular form. This whole process is the most time consuming and difficult process and is depicted in the figure below.
- Tabular data is most common way of representing data in machine learning or data mining.



http://blog.kaggle.com/2016/07/21/approaching-almost-any-machine-learning-problem-abhishek-thakur/

Curse of dimensionality

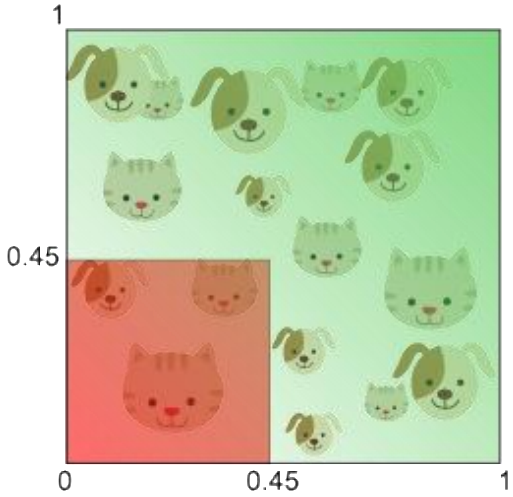
- **The number of records increases exponentially to sustain the same explain ability as the number of variables increases.**
- ***"If there are various logical ways to explain a certain phenomenon, the simplest is the best."* - Occam's Razor**

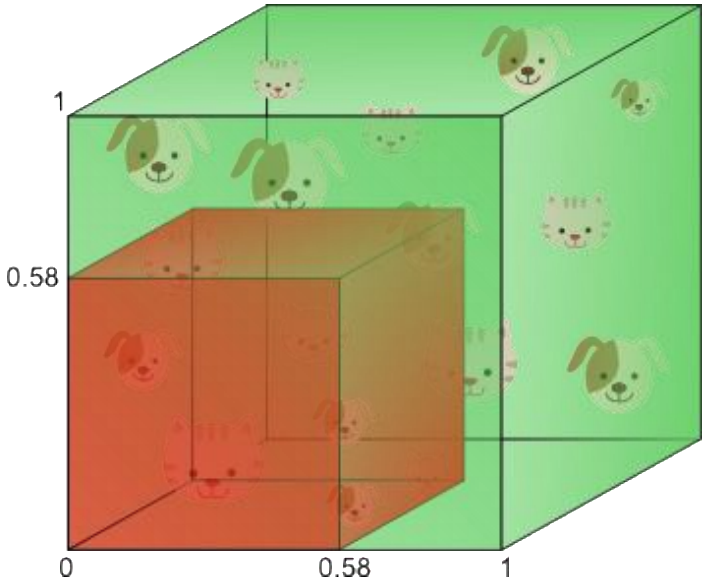


$2^1=2$ **$2^2=4$** **$2^3=8$**

Curse of dimensionality

- **The amount of training data needed to cover 20% of the feature range grows exponentially with the number of dimensions.**

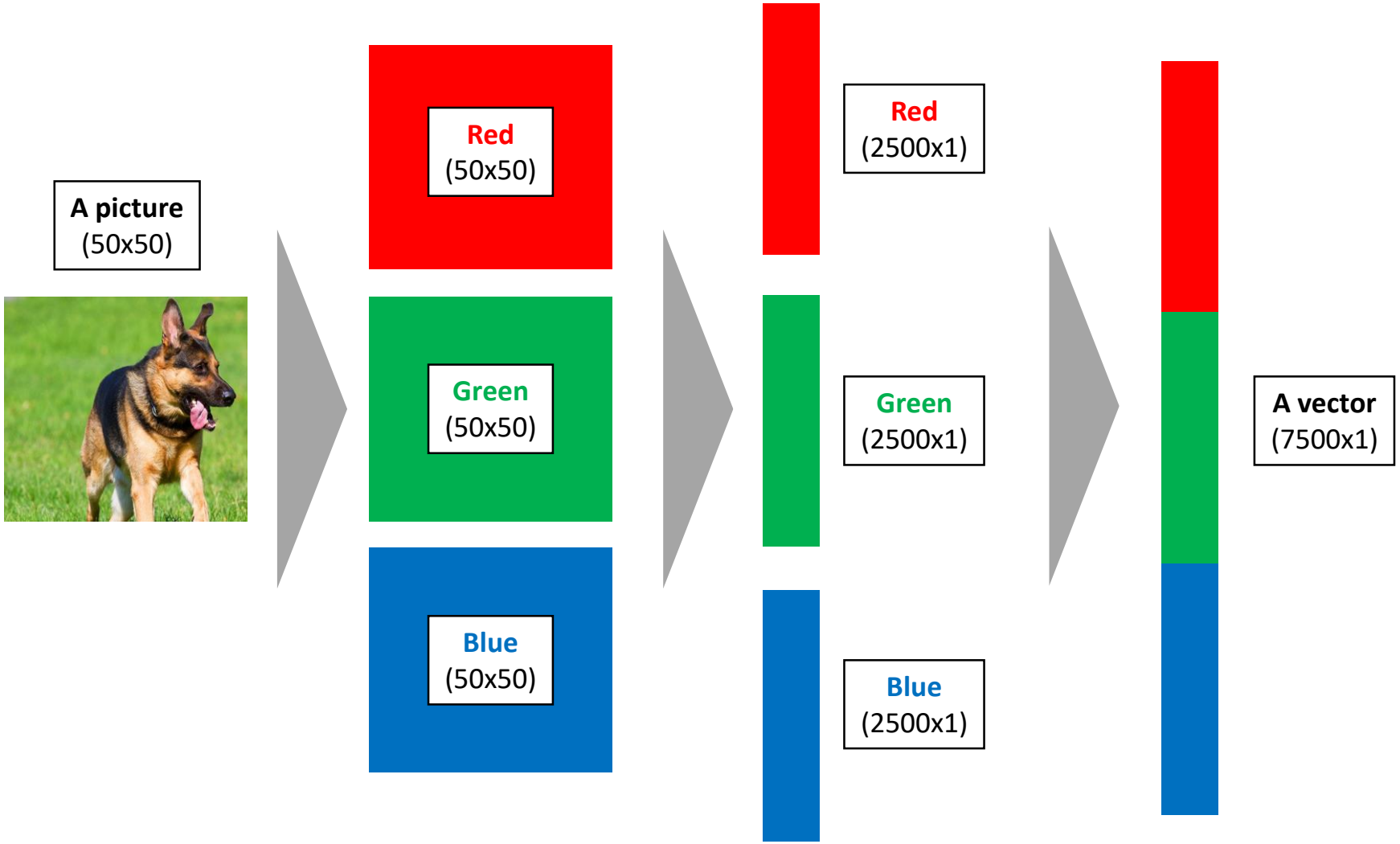




http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

Features = Data representation

- **Example: Representing pictures**
  - A picture → 7500-dimensional vector (with minimum value 0 and maximum value 255)



Features = Data representation

- **Example: Representing documents (bag-of-words)**
  - "I have a pen. You have a pen, too." → (1, 2, 2, 2, 1, 1, 0, 0, 0)
  - "I had lunch. Did you have lunch?" → (1, 1, 0, 0, 1, 0, 1, 2, 1)

Document 1: I have a pen. You have a pen, too.
Document 2: I had lunch. Did you have lunch?

'term frequency'

ID	I	have	a	pen	you	too	had	lunch	did
1	1	2	2	2	1	1	0	0	0
2	1	1	0	0	1	0	1	2	1

Features = Data representation

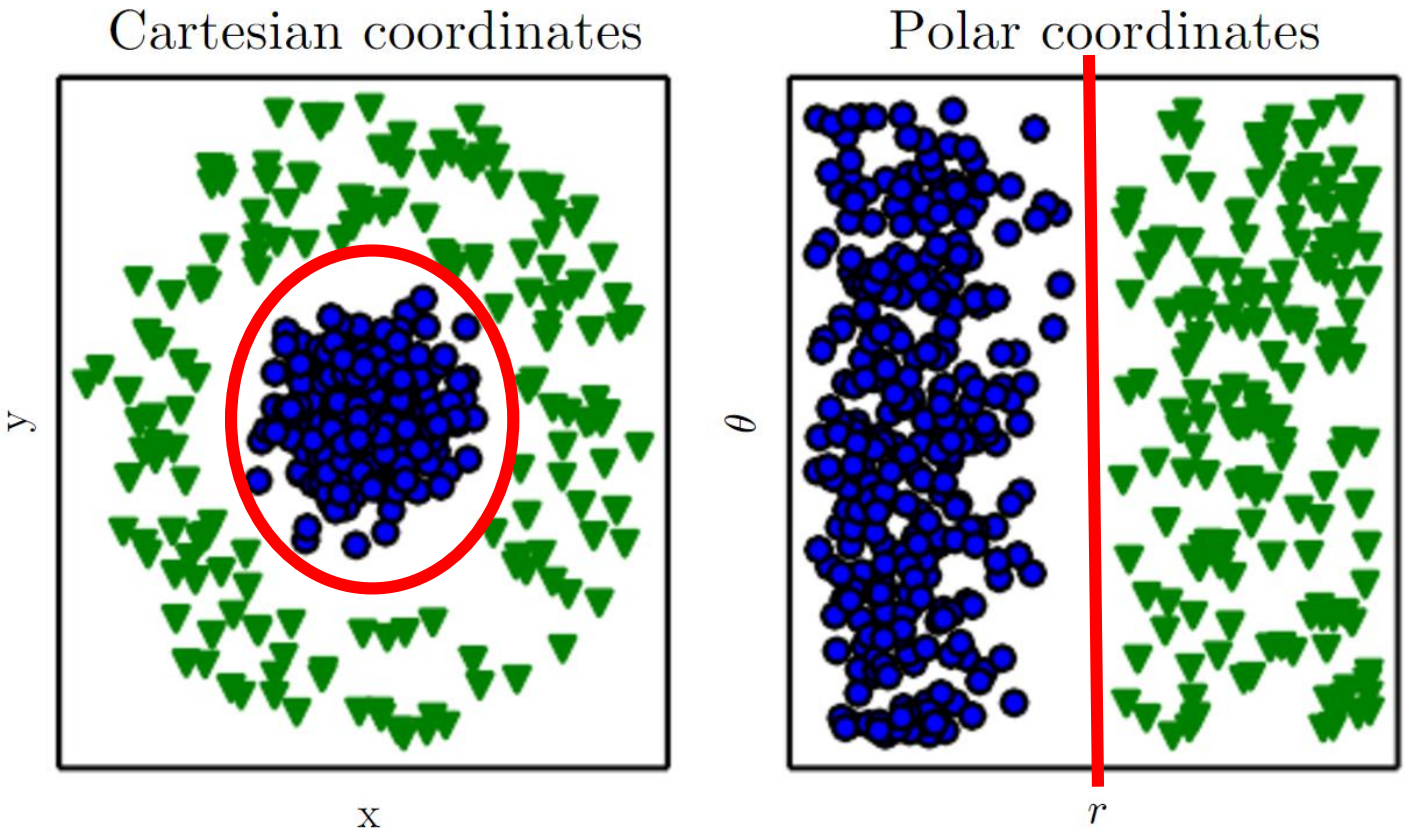
- **Example: One-hot encoding for the categorical variable 'season'**
  - spring → (1, 0, 0, 0)
  - summer → (0, 1, 0, 0)
  - fall → (0, 0, 1, 0)
  - winter → (0, 0, 0, 1)

ID	Season
1	fall
2	summer
3	spring
4	winter
5	summer
…	

ID	spring?	summer?	fall?	winter?
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0
4	0	0	0	1
5	0	1	0	0
…	…	…	…	…

Features = Data representation

- **Same data, but different representations**
  - $(x, y) \rightarrow (r, \theta)$



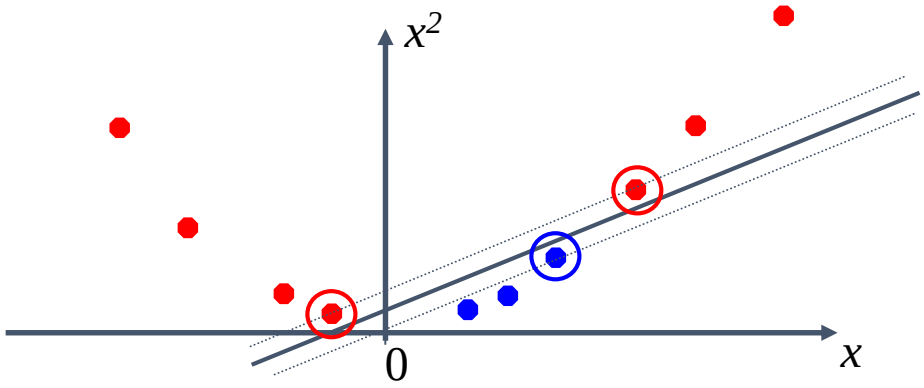
Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, 2016. (http://www.deeplearningbook.org/)

Features = Data representation

- **Same data, but different representations**
  - $(x) \rightarrow (x, x^2)$

Not linearly separable

Linearly separable



Feature engineering

- **Feature engineering**
  - Raw data 혹은 그 이후 단계의 data를 더 나은 표현으로 바꾸기 위한 변수를 생성/추출/변환하는 과정
  - 데이터의 표현, 즉 변수들이 어떻게 구성되느냐에 따라 머신러닝 모델 성능에 엄청난 영향을 미치므로, feature engineering은 매우 중요

- **Types**
  - Feature transformation (변수 변환) and generation (생성)
  - Feature (subset) selection (변수 선택)
  - Feature extraction (변수 추출)

Feature transformation and generation

- **Feature scaling**
- **One-hot encoding (binary dummy variables)**
- **Derived features**
- **Polynomial features**

Feature scaling

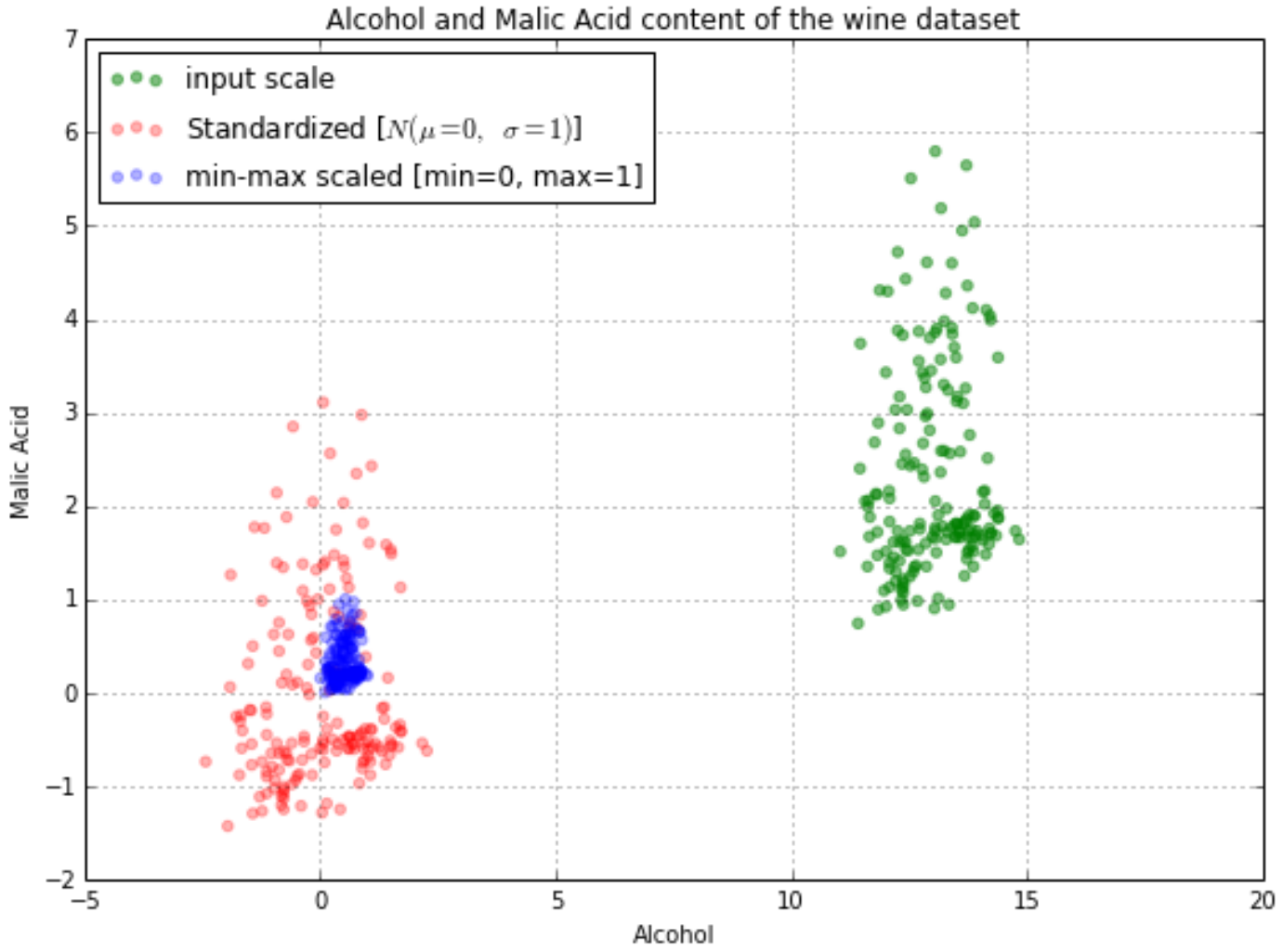
- **centering**
  - Subtract the mean per feature → Making **zero-centered** feature
  - $\boldsymbol{X'} = \boldsymbol{X} - \mathbf{1} \cdot \mu_{\boldsymbol{X}}$
- **standardization**
  - Divide the zero-centered feature by its standard deviation
  - $\boldsymbol{X'} = \frac{(\boldsymbol{X} - \mathbf{1} \cdot \mu_{\boldsymbol{X}})}{\sigma_{\boldsymbol{X}}}$
- **min-max scaling**
  - Rescale the feature to a fixed range [*minimum value*, *maximum value*]
  - [0, 1] scaling: $\boldsymbol{X'} = \frac{1}{\max(\boldsymbol{X}) - \min(\boldsymbol{X})} (\boldsymbol{X} - \mathbf{1} \cdot \min(\boldsymbol{X}))$
  - [-1, 1] scaling: $\boldsymbol{X'} = \frac{2}{\max(\boldsymbol{X}) - \min(\boldsymbol{X})} (\boldsymbol{X} - \mathbf{1} \cdot \min(\boldsymbol{X})) - \mathbf{1}$

Feature scaling

- **centering**
  - sklearn.preprocessing.StandardScaler(with_mean=True, with_std=False)
- **standardization**
  - sklearn.preprocessing.StandardScaler()
  - Default values for parameters '*with_mean*' and '*with_std*': *True*
- **min-max scaling**
  - sklearn.preprocessing.MinMaxScaler(feature_range=(*min*, *max*))
  - You can set arbitrary minimum value and maximum value.

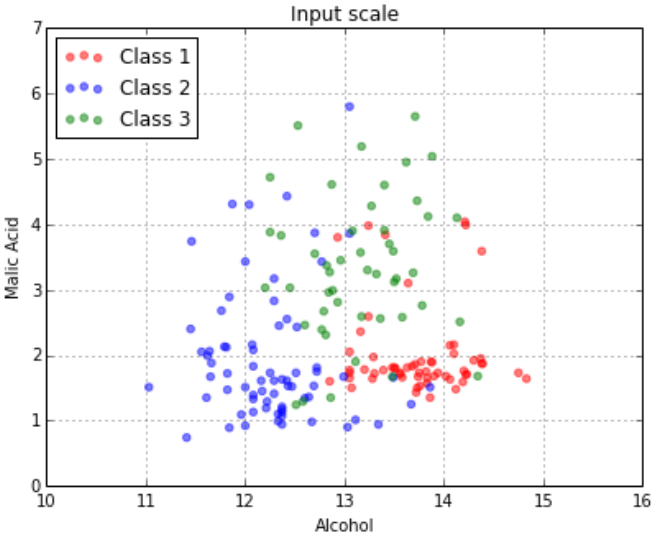
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

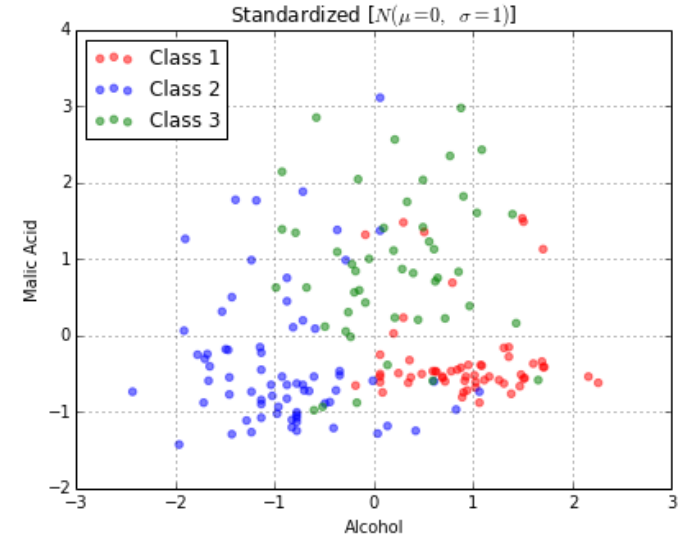
Feature scaling

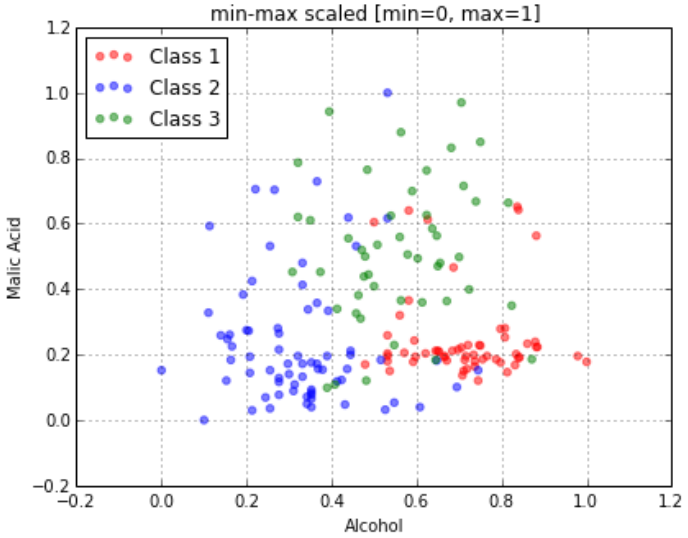


http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Feature scaling







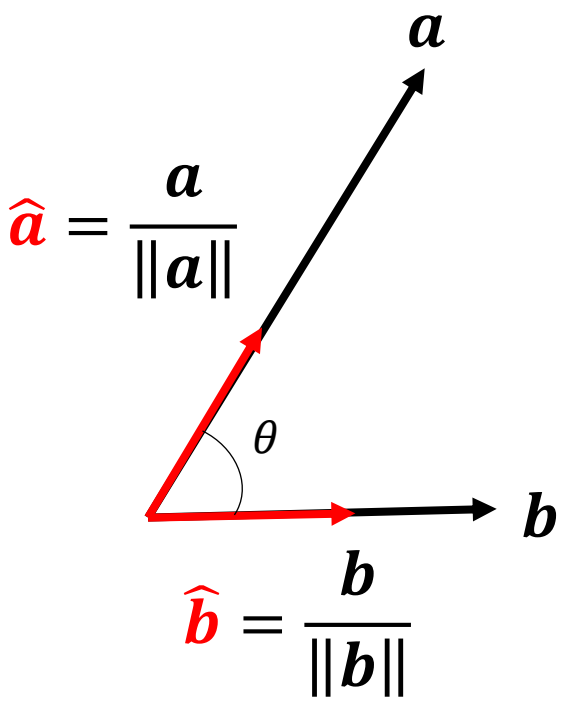
http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

Feature scaling

- **데이터 포인트 간의 거리/유사도를 기반으로 작동하는 알고리즘**
  - ex) k-means clustering, hierarchical clustering, k-nearest neighbors, support vector machine (SVM) with radial basis function (RBF) kernel
  - 스케일이 큰 입력변수가 큰 영향을 미치는 distance/similarity metrics를 사용하는 경우 feature scaling이 필요할 수 있음: Euclidean distance, RBF kernel, etc.
- **Gradient descent/ascent 기반의 최적화를 통해 모델을 학습하는 알고리즘**
  - ex) Logistic regression, neural network, etc.
  - Gradient가 대부분 입력변수의 스케일에 비례하므로 feature scaling을 통해 학습 속도를 빠르게 할 수 있음 (fast converge)
- **Zero-centered data라는 가정을 기반으로 하는 알고리즘**
  - ex) Principal component analysis (PCA), SVM with radial basis function (RBF) kernel
  - 위의 학습 방법은 기본 가정이 zero-centered data (즉, 각 변수의 평균이 0) 에서 출발

Side note: Sample normalization

- **In some cases, normalizing sample vectors (or point vectors) is efficient.**
  - The unit vector can be obtained by vector normalization.
  - The cosine similarity of two original vectors is equal to the inner product of corresponding two unit vectors.



$$cos\theta = \frac{a \cdot b}{\|a\|\|b\|} = \frac{a}{\|a\|} \cdot \frac{b}{\|b\|} = \hat{a} \cdot \hat{b} = \hat{a}^T \hat{b}$$

One-hot encoding (or 1-of-C coding)

- **Example: One-hot encoding for the categorical variable 'season'**
  - spring → (1, 0, 0, 0)
  - summer → (0, 1, 0, 0)
  - fall → (0, 0, 1, 0)
  - winter → (0, 0, 0, 1)

ID	Season
1	fall
2	summer
3	spring
4	winter
5	summer
…	

binary dummy variables

ID	spring?	summer?	fall?	winter?
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0
4	0	0	0	1
5	0	1	0	0
…	…	…	…	…

One-hot encoding (or 1-of-C coding)

- **You must consider**
  - the variable's type: *nominal* or *ordinal*
  - Example: survey response

nominal variable
(명목형 변수)

ID	Season
1	fall
2	summer
3	spring
4	winter
5	summer
…	

ordinal variable
(순서형 변수)

ID	Preference
1	so so
2	good
3	very good
4	bad
5	very bad
…	…

ordinal → interval (구간형)

- very good → 5
- good → 4
- so so → 3
- bad → 2
- very bad → 1

ID	Preference
1	3
2	2
3	1
4	4
5	5
…	…

One-hot encoding (or 1-of-C coding)

- **You must consider**
  - the number of unique values
  - Example: 195 countries
    - Each country is encoded as a one-hot vector of 195 dimensions.
      - Too sparse ➔ difficult to learn this information.
    - How about considering their
      - continents?
      - GDP level?
      - political system?
      - size of land?
      - population?
      - …

attempts to reduce the number of unique values

http://www.worldometers.info/geography/how-many-countries-are-there-in-the-world/

Derived features

- **Derived features (or variables): 파생변수 / 유도변수**
  - 데이터가 생성되는 도메인에 의해 새롭게 만들어지는 변수
  - 컴퓨터가 자동으로 할 수 없는 인간 지능의 영역

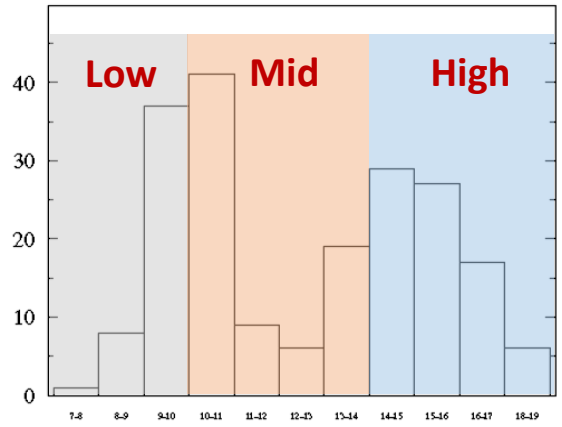
  - Example) Kaggle: Predict bicycle demands
    - 자전거 대여 시간정보를 이용한 유도변수?
    - 요일정보를 활용한 유도변수?
    - 새로운 날씨 정보?
    - 그 외에?

Polynomial features

- **Polynomial features**
  - Generating new features consisting of all polynomial combinations of the features
  - Example: Generating polynomial features with degree=2
    - $(X_1, X_2, X_3) \rightarrow (X_1,\ X_2,\ X_3,\ X_1^2,\ X_2^2,\ X_3^2, X_1X_2,\ X_2X_3,\ X_3X_1)$
    - X_1X_2 represent the interaction of two features X_1 and X_2.
  - Too many features can be created.
    - After generating polynomial variables, you should consider dimension reduction or regularization.

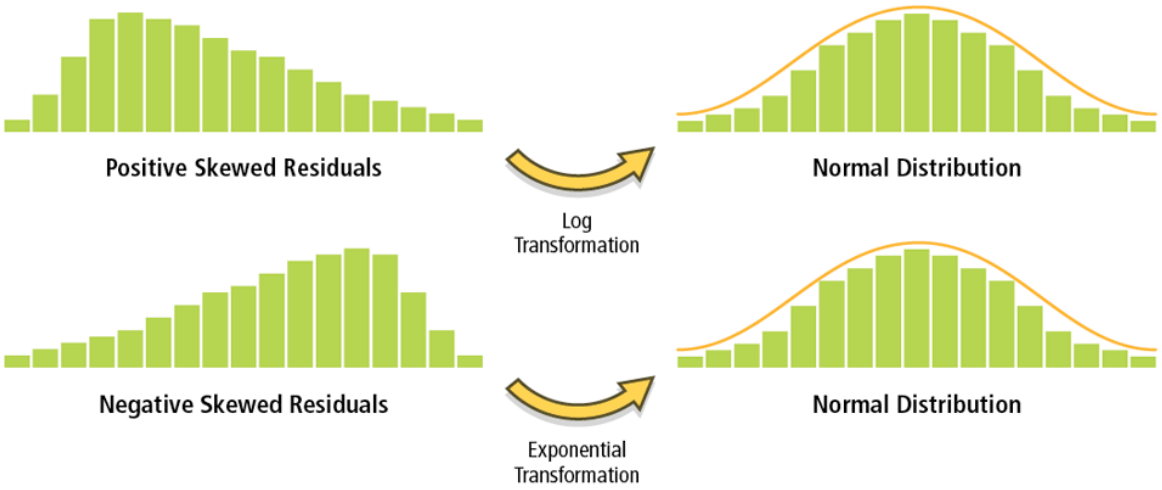
Other methods

- **Binning**
  - continuous → ordinal
  - Note that if you discretize the continuous target variable, your problem turns into a '*classification*' from '*regression*'.



Age
0~6 → 'baby'
7~19 → 'child'
20~ → 'adult'

- **Log-transformation & exponential transformation**
  - Trying to change skewed distribution to symmetric distribution



http://www.esri.com/news/arcuser/0111/findmodel.html

Feature (subset) selection

- **Filter method**
- **Wrapper method**
- **Embedded method**

Filter method

- **Filter method**
  - Select some features regardless of machine learning algorithms.
  - How?
    - Using domain knowledge
    - Finding redundant features among input features based on correlation measure
    - Finding highly correlated input features to target feature

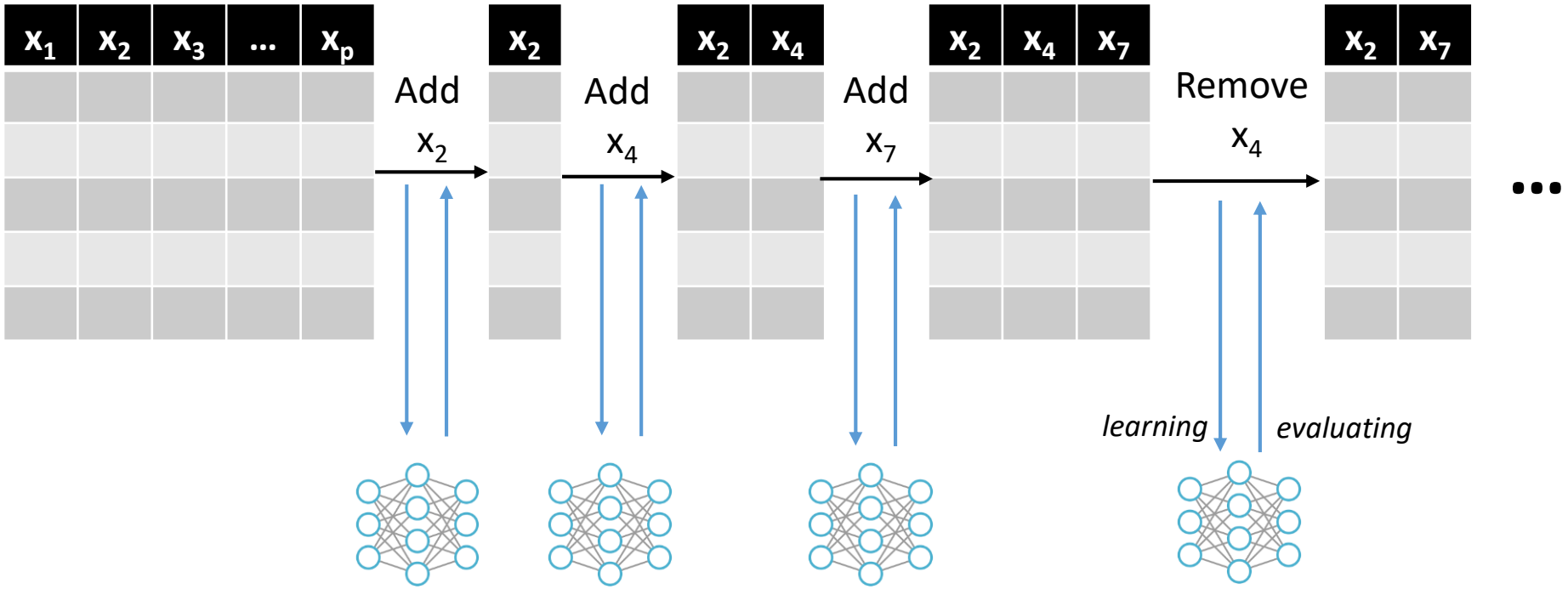
$$Pearson\ correlation(A,B) = \frac{Cov(A,B)}{std(A)std(B)} = \frac{\sum_{i=1}^{p}(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^{p}(a_i - \bar{a})^2}\sqrt{\sum_{i=1}^{p}(b_i - \bar{b})^2}}$$

  - Weak points
    - They cannot consider interactions among features.
    - Are variables with low scores really not influential?

There are some statistical methods, including correlation measures, for calculating feature scores.
Please see the section "*Details of the Feature Selection Methods*" in the below page:
https://msdn.microsoft.com/en-us/library/azure/dn905854.aspx

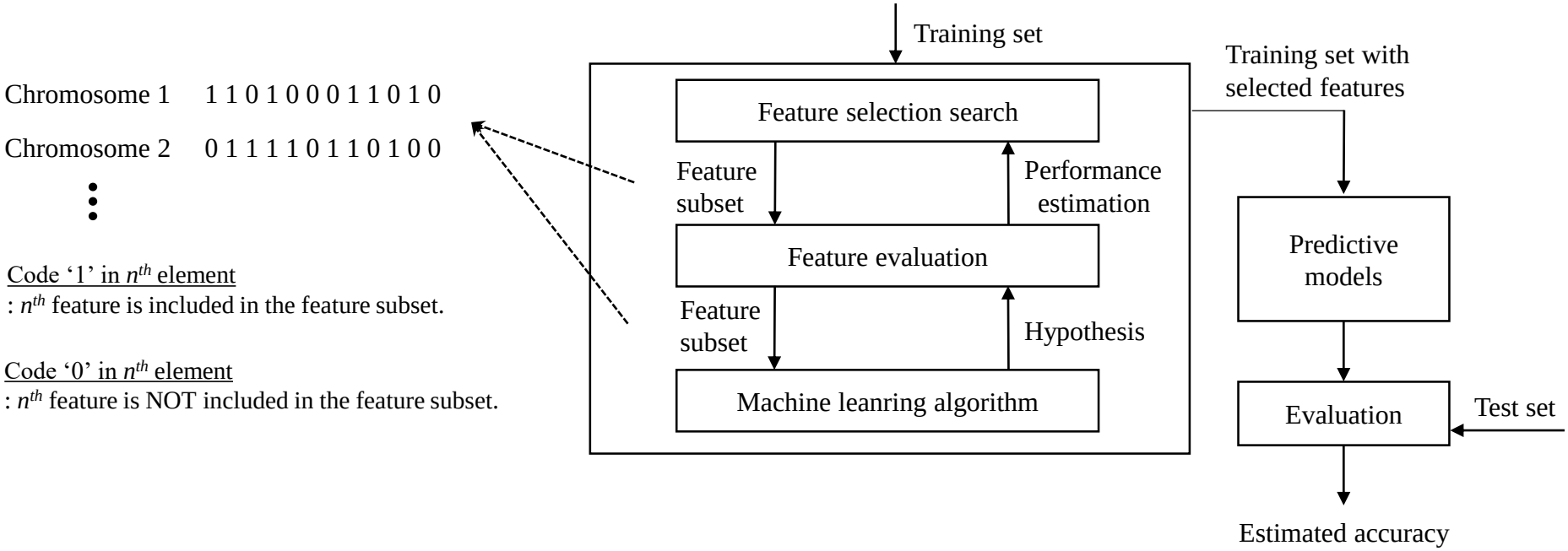
Wrapper method

- **Wrapper method**
  - Score feature subsets and find the best subset by learning and evaluating machine learning models iteratively.
  - Example: Stepwise feature selection



Wrapper method

- **Evolutionary algorithms-based wrapper method**
  - Evolutionary algorithms: Genetic algorithm, Ant-colony algorithm, etc.
  - Example: binary genetic algorithm-based feature subset selection



Wrapper method

- **Wrapper method**
  - Weak points
    - Computation cost: The larger the number of variables, the longer it takes to find the best variable subset.
    - Model Generalizability: There is a high risk of overfitting.

Embedded method

- **Embedded method**
  - Some machine learning algorithms/models can do generating models and selecting and evaluating features at the same time.
    - L1-regularized models: LASSO regression, $l1$-logistic regression
    - Tree-based
      - Single decision tree
      - Random Forest
      - Gradient-boosting tree, especially XGBoost

Feature extraction

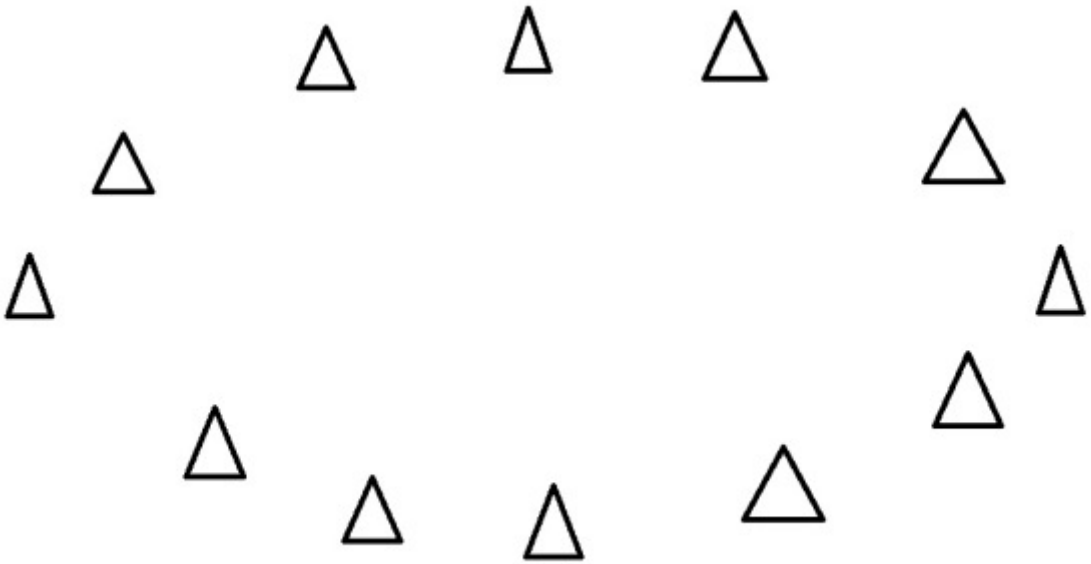
- **Principal component analysis (PCA)**
- **Singular value decomposition (SVD)**
- ~~**Unsupervised neural network: Autoencoders, Restricted Boltzmann machine (RBM)**~~

Principal component analysis (PCA)

- **PCA (주성분분석)**
  - 데이터에 내재된 분산을 잘 설명하는 주성분 (principal component: PC) 들을 찾아내는 방법
  - 수학적 의미
    - 변수들의 공분산 행렬 (covariance matrix) 혹은 상관계수 행렬 (correlation matrix) 을 도출
    - 각 주성분은 공분산 행렬의 고유벡터 (eigenvector) 에 해당되며, 대응되는 고유값 (eigenvalue) 이 클 수록 데이터에 내재된 분산을 많이 설명함.
  - 주성분의 수 = 변수의 수
  - 활용 방법
    - 분산을 많이 설명하는 상위 주성분을 선택하고, 이들을 새로운 변수로 사용*
    - 시각화

Principal component analysis (PCA)

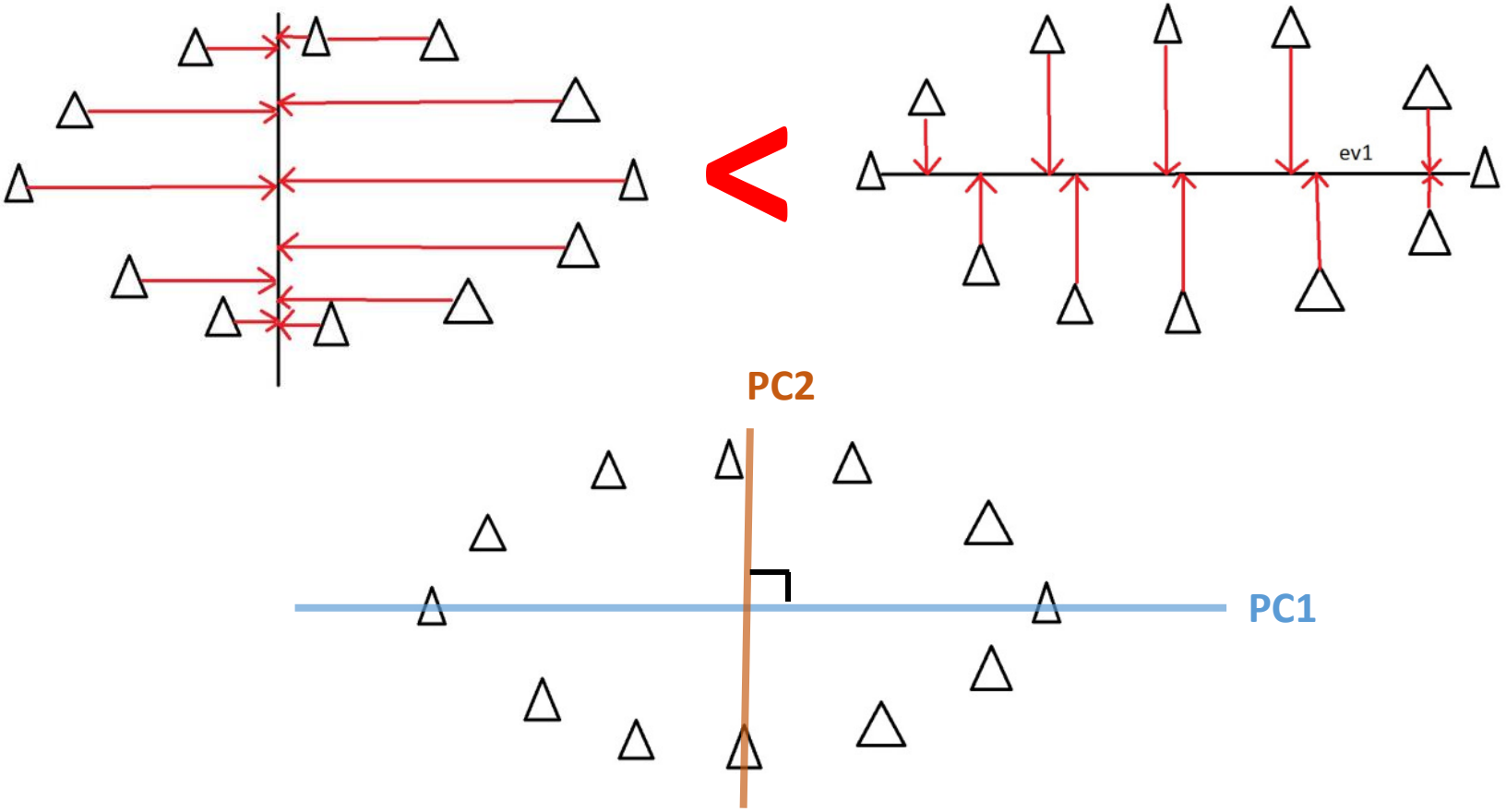
- **There are some triangles(data points) in the shape of an oval.**
- **We want to find the direction where there is most variance.**



https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-reduction/

Principal component analysis (PCA)

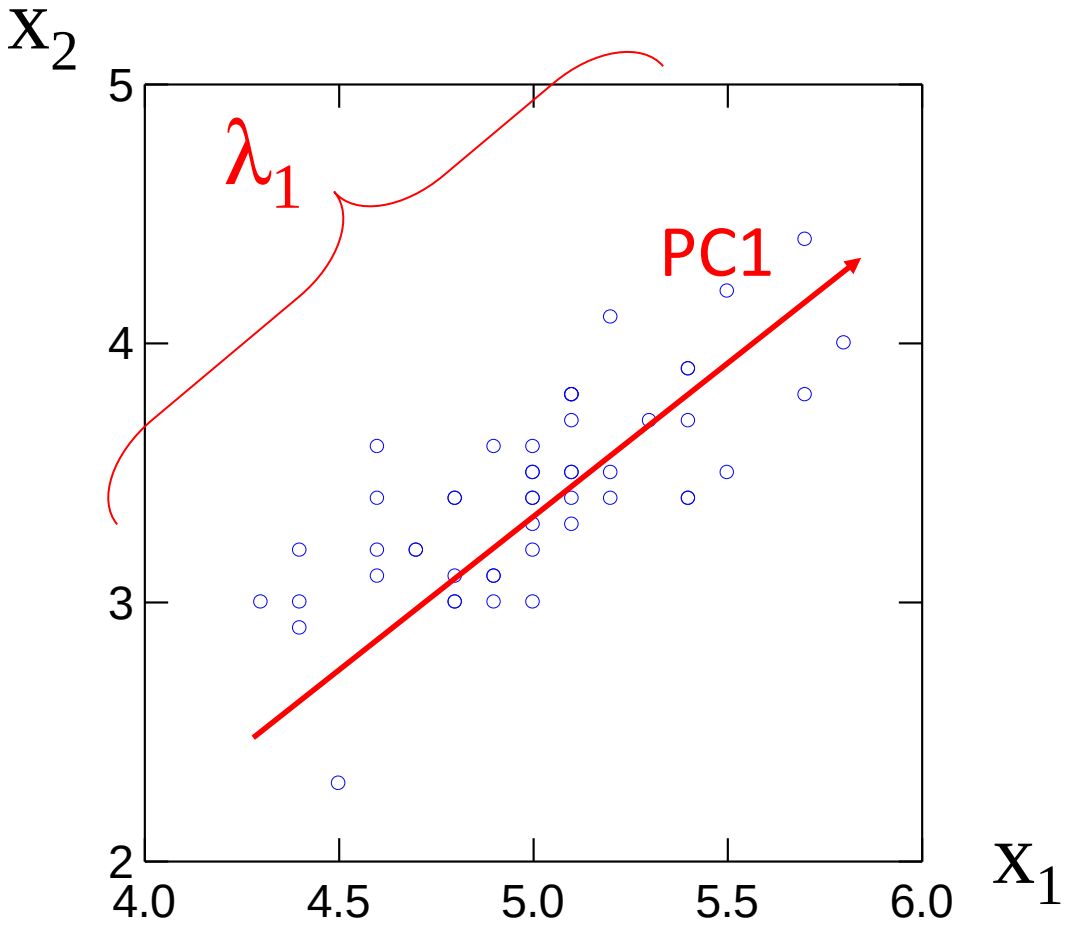
- **Find the straight line where the data is most spread out when projected onto it. ➔ And then find the next one which is *orthogonal* to previous one.**



https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-reduction/

Principal component analysis (PCA)

- **Each PC is the linear combination of original input features.**
- **Each PC explains part of original total variance.**



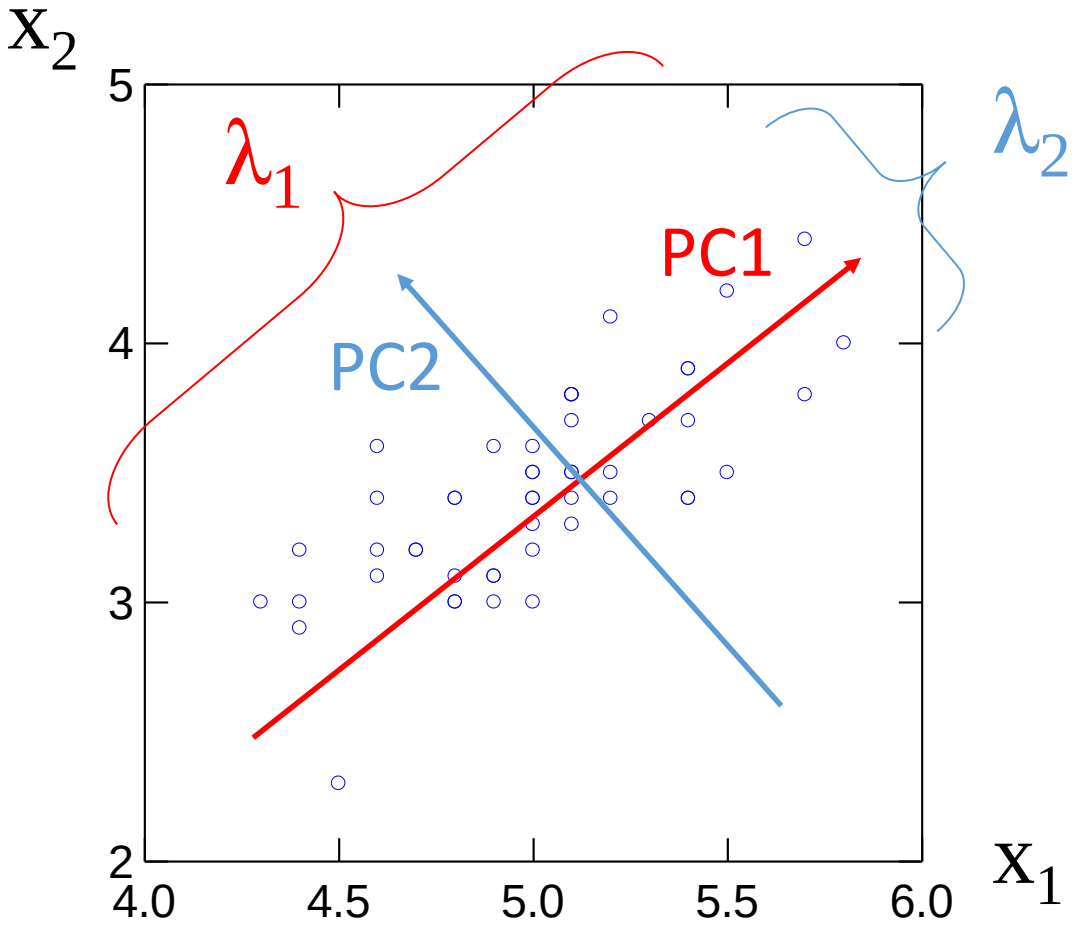
$$PC1 = aX_1 + bX_2$$

Total variance = 100

PC1's variance = 90

Principal component analysis (PCA)

- **Each PC is the linear combination of original input features.**
- **Each PC explains part of original total variance.**



$$PC1 = aX_1 + bX_2$$

$$PC2 = cX_1 + dX_2$$

Total variance = 100

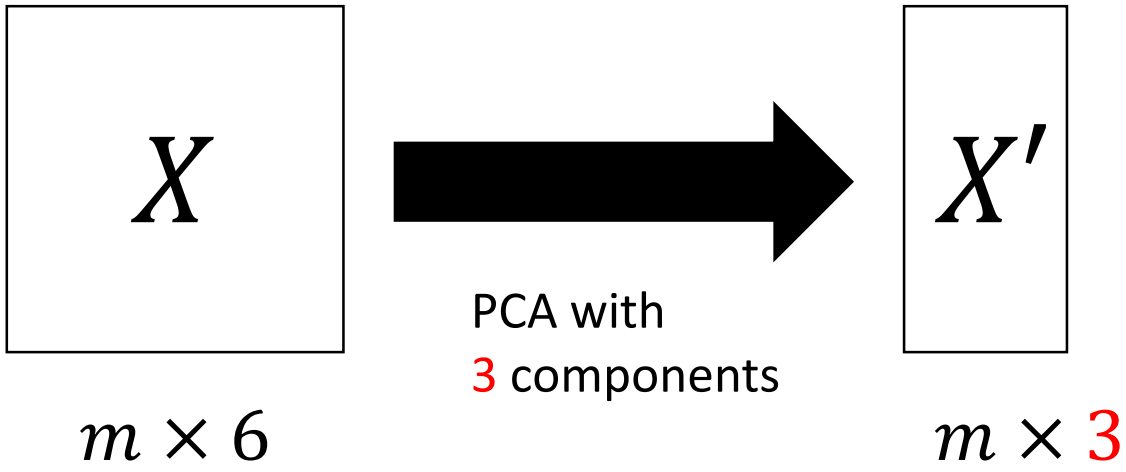
PC1's variance = 90

PC2's variance = 10

Principal component analysis (PCA)

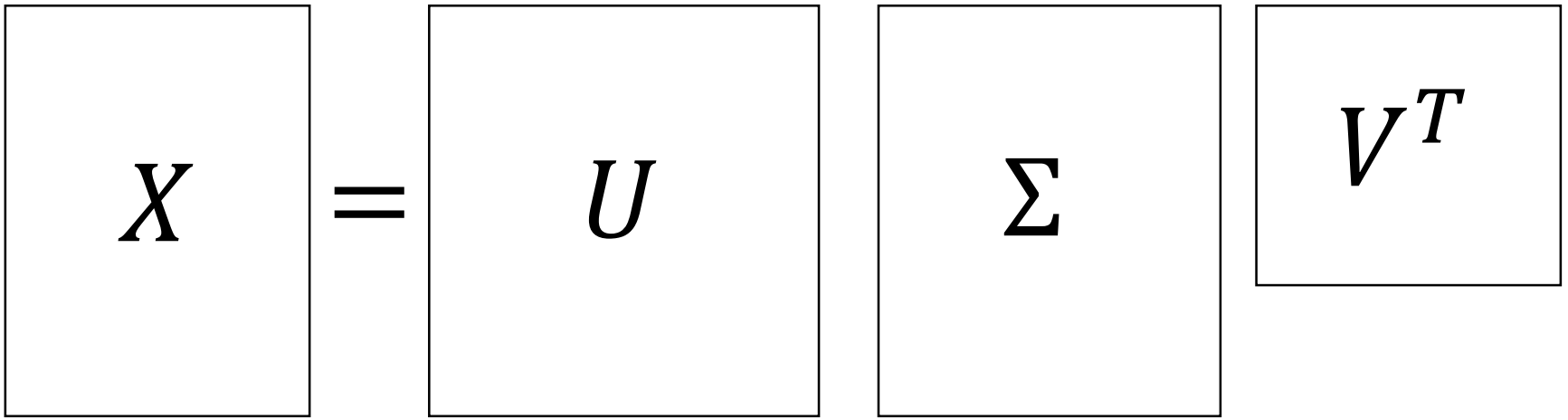
- **How the PCA reduces the dimensionality of data**
  - If you have *n* features, the PCA makes *n* PCs.
  - Select top-*k* PCs.
  - Example: Data with 6 features
    - Select top-3 PCs.

PC	1st	2nd	3rd	4th	5th	6th	Total
var	40	25	15	10	6	4	100%
var (cumulative)	40	65	80	90	96	100	100%



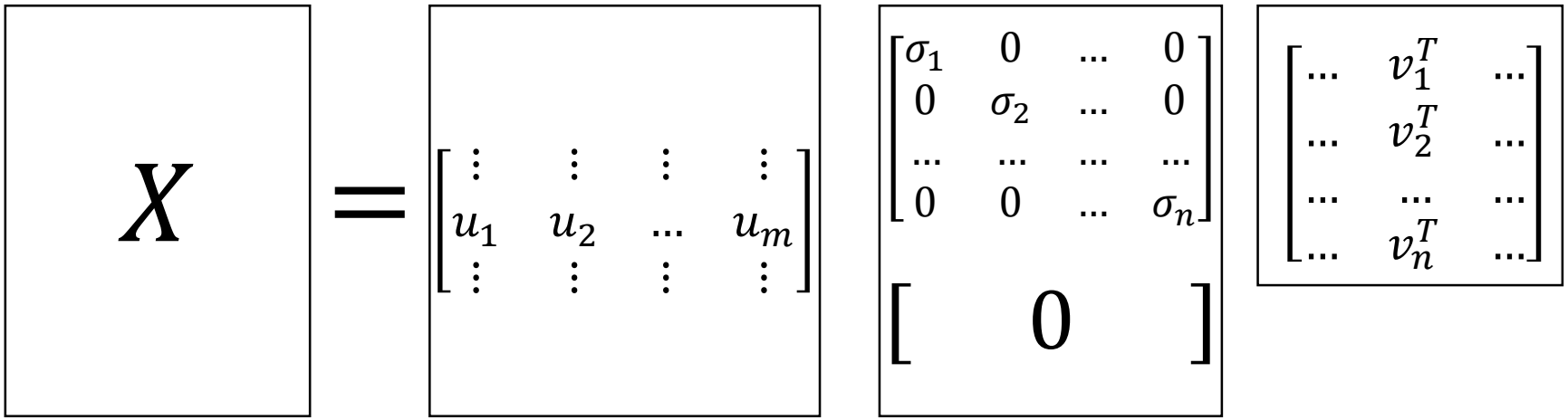
Singular value decomposition (SVD)

- **SVD (특이값 분해)**
  - 실수 행렬 X를 세 개의 행렬로 분해(decomposition): $X = U\Sigma V^T$
    - X: $(m \times n)$ real matrix (실수 행렬)
    - U: $(m \times m)$ orthogonal matrix (직교 행렬)
    - V: $(n \times n)$ orthogonal matrix (직교 행렬)
    - Σ: $(m \times n)$ rectangular diagonal matrix (직사각 대각행렬)



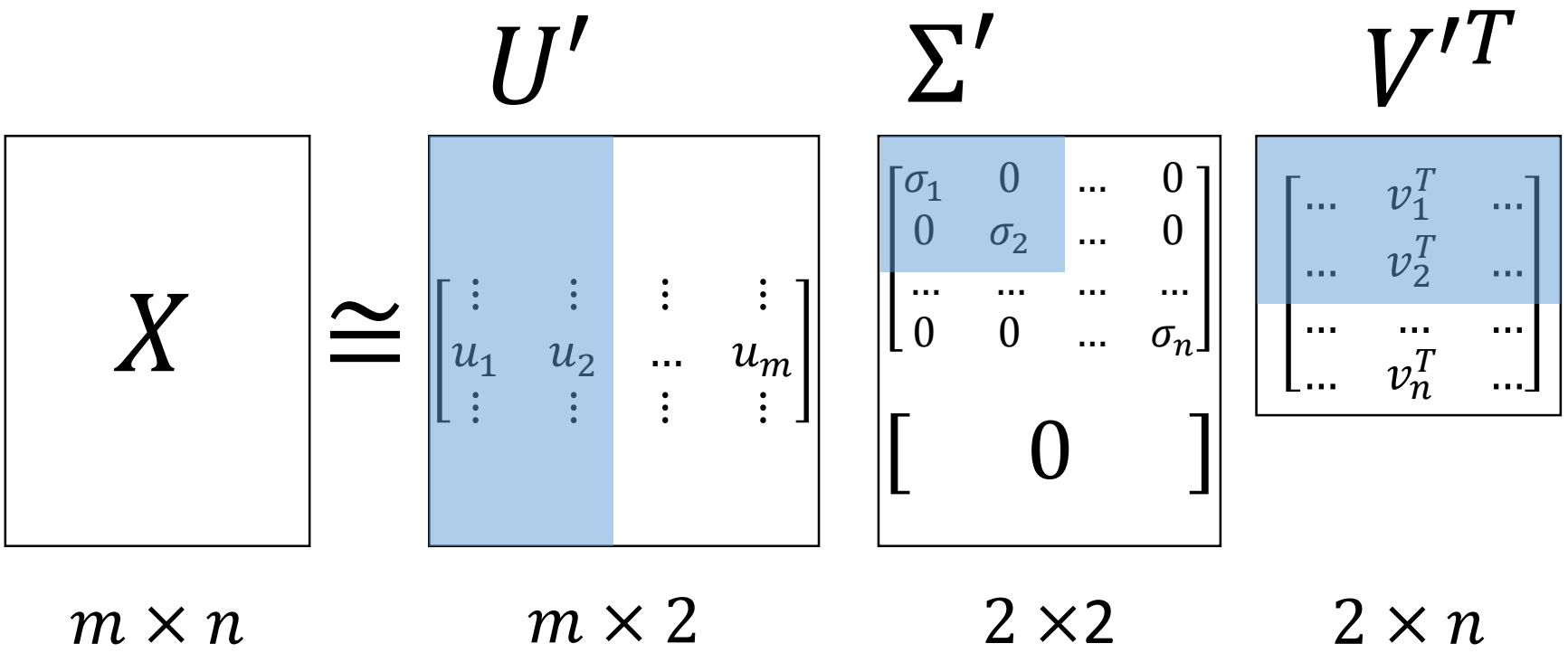
Singular value decomposition (SVD)

- **SVD (특이값 분해)**
  - 실수 행렬 X를 세 개의 행렬로 분해(decomposition): $X = U\Sigma V^T$
    - $u_i (i = 1, 2, \dots, m)$: left-singular vectors
    - $v_j (j = 1, 2, \dots, n)$: right-singular vectors
    - $\sigma_k (k = 1, 2, \dots, n)$: singular values



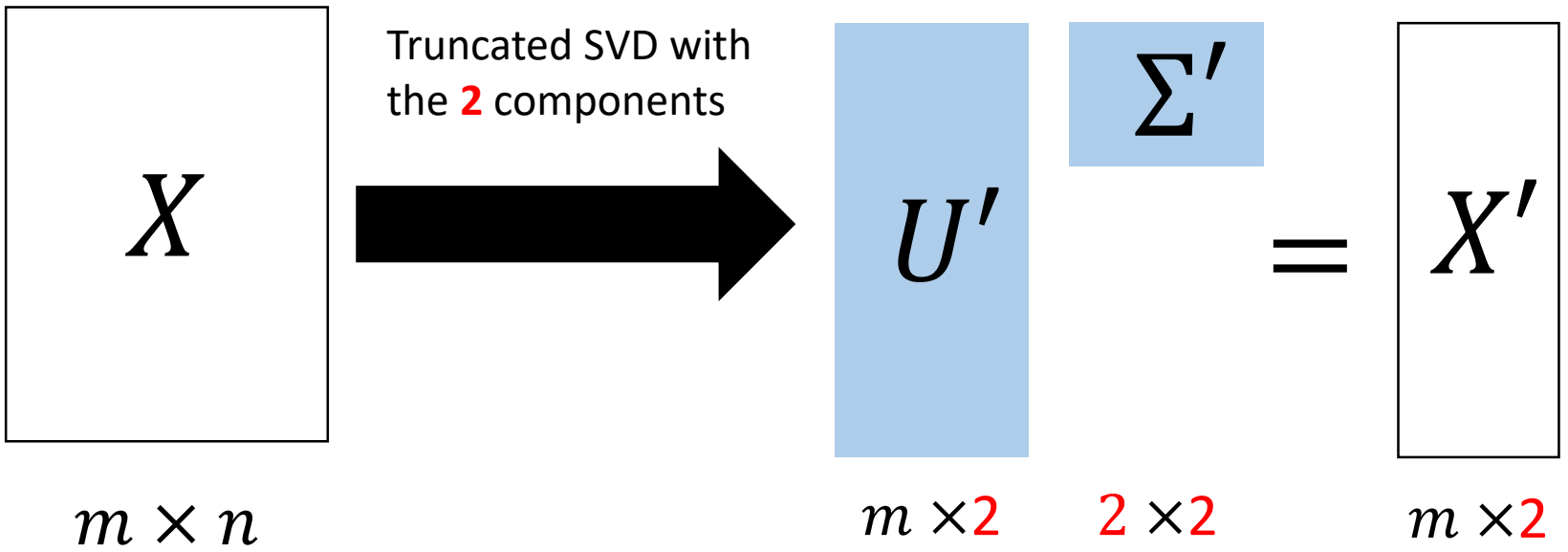
Truncated SVD

- **Truncated SVD**
  - Top-k 개의 특이값과 해당 특이벡터들을 이용하여 행렬을 근사적으로 분해
  - 만약 2개의 특이벡터를 사용하는 경우, 다음 파란색 영역으로 X를 근사적으로 분해



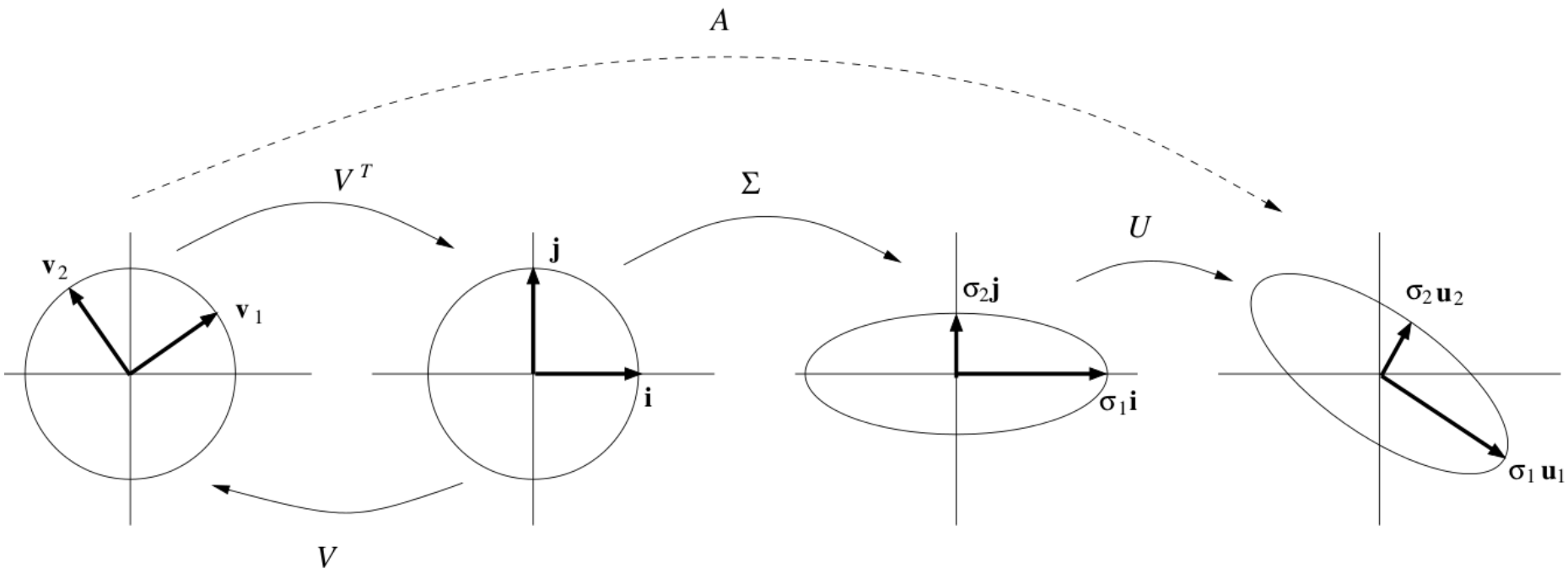
Truncated SVD

- **How to reduce the dimensionality of data**
  - Truncated SVD 수행 후, $X' = U'\Sigma'$를 변환된 데이터로 이용
  - n차원의 데이터를 2차원으로 축소



Singular value decomposition (SVD)

- **SVD (특이값 분해)**
  - U와 V는 데이터 영역을 회전하고,
  - Σ는 데이터 스케일을 변화하는 효과



SVD 참고

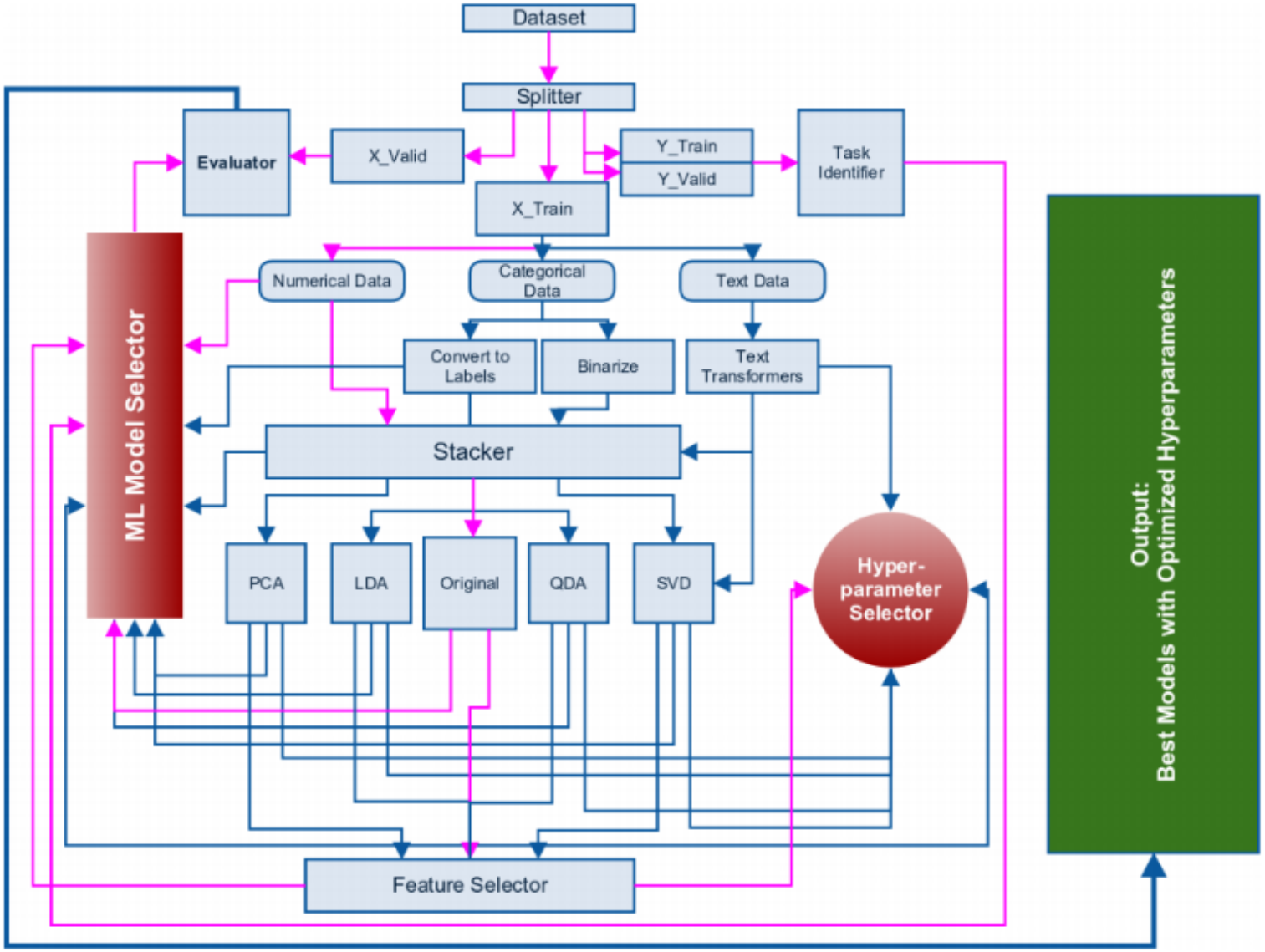
- https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/04/06/pcasvdlsa/
- http://darkpgmr.tistory.com/106

Feature engineering에 대한 제언

- **변수는 일단 많이 확보하자.**
  - 데이터가 생성되는 도메인 배경을 표현할 수 있는 변수를 가능한 한 많이 모으는 것이 좋다.
  - 다양한 방법을 통해 변수는 많이 생성시킬 수 있다.
    - Feature generation, extraction, etc.

- **많은 변수들을 컨트롤할 수 있는 방법을 같이 사용하자.**
  - 변수가 많아도 차원 축소를 위한 feature selection, feature extraction 방법들을 통해 차원의 저주를 어느 정도 피할 수 있다.
  - Regularization이 매우 중요하다.

A framework for machine learning competitions



http://blog.kaggle.com/2016/07/21/approaching-almost-any-machine-learning-problem-abhishek-thakur/