KSBi-BIML 2021

Bioinformatics & Machine Learning (BIML)
Workshop for Life Scientists

생물정보학 & 머쉰러닝 워크샵(온라인)

SVM (Support Vector Machine)의 개념과 활용

이상근

한국생명정보학회 Korean Society for Bioinformatics
서울대학교 생물정보연구소 SNU BIOINFORMATICS INSTITUTE
Gene·2·us

Bioinformatics & Machine Learning for Life Scientists

BIML-2021

안녕하십니까?

한국생명정보학회의 동계 워크샵인 BIML-2021을 2월 15부터 2월 19일까지 개최합니다. 생명정보학 분야의 융합이론 보급과 실무역량 강화를 위해 도입한 전문 교육 프로그램인 BIML 워크샵은 2015년에 시작하였으며 올해로 7차를 맞이하게 되었습니다. 유례가 없는 코로나 대유행으로 인해 올해의 BIML 워크숍은 온라인으로 준비했습니다. 생생한 현장 강의에서만 느낄 수 있는 강의자와 수강생 사이의 상호교감을 가질 수 없다는 단점이 있지만, 온라인 강의의 여러 장점을 살려서 최근 생명정보학에서 주목받고 있는 거의 모든 분야를 망라한 강의를 준비했습니다. 또한 온라인 강의의 한계를 극복하기 위해서 실시간 Q&A 세션 또한 마련했습니다.

BIML 워크샵은 전통적으로 크게 생명정보학과 AI, 두 개의 분야로 구성되어오고 있으며 올해 역시 유사한 방식을 채택했습니다. AI 분야는 Probabilistic Modeling, Dimensionality Reduction, SVM 등과 같은 전통적인 Machine Learning부터 Deep Learning을 이용한 신약개발 및 유전체 연구까지 다양한 내용을 다루고 있습니다. 생명정보학 분야로는, Proteomics, Chemoinformatics, Single Cell Genomics, Cancer Genomics, Network Biology, 3D Epigenomics, RNA Biology, Microbiome 등 거의 모든 분야가 포함되어 있습니다. 연사들은 각 분야 최고의 전문가들이라 자부합니다.

이번 BIML-2021을 준비하기까지 너무나 많은 수고를 해주신 BIML-2021 운영위원회의 김태민 교수님, 류성호 교수님, 남진우 교수님, 백대현 교수님께 커다란 감사를 드립니다. 또한 재정적 도움을 주신, 김선 교수님 (AI-based Drug Discovery), 류성호 교수님, 남진우 교수님께 감사를 표시하고 싶습니다. 마지막으로 부족한 시간에도 불구하고 강의 부탁을 흔쾌히 허락하시고 훌륭한 강의자료를 만드는데 노력하셨을 뿐만 아니라 실시간 온라인 Q&A 세션까지 참여해 수고해 주시는 모든 연사분들께 깊이 감사드립니다.

2021년 2월

한국생명정보학회장 김동섭

강의개요

SVM (Support Vector Machine)의 개념과 활용

SVM은 분류 문제에 해결함에 있어 딥신경망, 그래디언트 부스팅과 함께 가장 좋은 성능을 보이는 감독 학습 기법으로, 기계학습의 아버지라 불리는 Vladmir Vapnik 교수님의 약 30년간의 기계학습 연구 끝에 탄생한 역작이다. SVM은 기본적으로 큰 마진 (large margin)을 갖는 선형 분류기를 학습하며, 커널 (kernel)이라는 매커니즘을 이용하여 원래의 데이터 공간 대신 소위 피처 공간 (feature space)에서 예측에 효과적인 선형 분류기를 학습할 수 있다. 이와 같이, 커널을 이용하는 SVM은 딥신경망과 마찬가지로 데이터의 새로운 표현형을 학습할 수 있기 때문에 비선형 결정 경계를 갖는 분류 문제 해결에 효과적이다.

SVM은 벡터 형 데이터 뿐만 아니라 시퀀스, 그래프 등의 구조를 갖는 데이터에 대해서도 커널을 정의함으로써 적용 가능하며, 다른 기계학습 기법에 비해 과적합 우려가 적으며, 특히 딥신경망 학습과 비교해서는 학습 문제가 볼록 최적화로 귀결되기 때문에 최적의 학습 파라미터를 찾기가 훨씬 용이하다는 장점을 지닌다. 특히, 후자의 사실은 분류기를 구성하는 학습 파라미터의 기여도를 통해 중요 인자를 추출함에 있어 큰 장점으로 작용할 수 있다.

본 강의에서는 SVM의 개념과 학습 최적화 문제, 듀얼 최적화 문제, 커널 SVM에 대해 이해하고, 이를 바탕으로 바이오 데이터를 활용한 문제 해결에 적용할 수 있는 역량을 함양한다.

*교육생준비물:
노트북 (메모리 8GB 이상, 디스크 여유공간 30GB 이상)
구글 계정 생성
Anaconda 다운로드 및 설치 (https://www.anaconda.com/)

* 강의: 이상근 교수 (고려대학교 정보보호학과)

Curriculum Vitae

Speaker Name: Sangkyun Lee, Ph.D.

▶Personal Info

Name	Sangkyun Lee
Title	Assistant professor
Affiliation	Korea University

▶Contact Information

Address 45, Anam-ro, Seongbuk-gu, Seoul, Republic of Korea

Email sangkyun@korea.ac.kr

Phone Number 02-3290-4890

Research interest : Machine Learning, Secure AI, Collaborative Learning

Educational Experience

2003	B.S., Seoul National University
2005	M.S., Seoul National University
2011	Ph.D., University of Wisconsin-Madison, USA

Professional Experience

2011-2014	Post-doc researcher, Collaborative Research Center (SFB876), TU Dortmund University, Germany
2015-2017	Principal Investigator, Collaborative Research Center (SFB876), TU Dortmund University, Germany
2017-2019	Assistant Professor, Department of Computer Science, Hanyang University ERICA
2020-Current	Assistant Professor, Department of Cybersecurity, Korea University

Selected Publications (5 maximum)

1. Data Quality Measures and Efficient Evaluation Algorithms for Large-Scale High-Dimensional Data, Hyeongmin Cho and Sangkyun Lee, Applied Sciences, 2021

2. Sparse Portfolio Selection via the sorted ℓ1 - Norm, Philipp J. Kremer*, Sangkyun Lee*, Małgorzata Bogdan, and Sandra Paterlini, Journal of Banking & Finance, 2020

3. Structure Learning of Gaussian Markov Random Fields with False Discovery Rate Control, Sangkyun Lee, Piotr Sobczyk and Malgorzata Bogdan, Symmetry, 2019

4. Compressed Learning of Deep Neural Networks for OpenCL-Capable Embedded Systems, Sangkyun Lee and Jeonghyun Lee, Applied Sciences, 2019

5. The mutational landscape of MYCN, Lin28b and ALKF1174L driven murine neuroblastoma mimics human disease, Bram De Wilde, Anneleen Beckers, Sven Lindner, Althoff Kristina, Katleen De Preter, Pauline Depuydt, Pieter Mestdagh, Tom Sante, Steve Lefever, Falk Hertwig, Zhiyu Peng, Le-ming Shi, Sangkyun Lee, Elien Vandermarliere, Lennart Martens, Björn Menten, Alexander Schramm, Matthias Fischer, Johannes Schulte, Jo Vandesompele and Frank Speleman, Oncotarget, 2017

KSBi-BIML 2021

SVM (Support Vector Machine)의 개념과 활용

고려대학교 정보보호대학원 이상근

Agenda

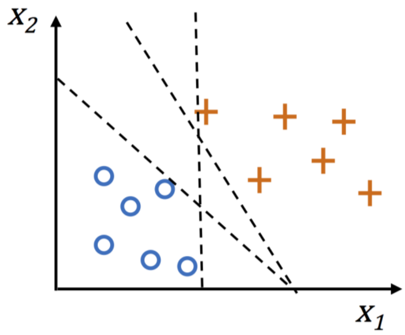
1. SVM의 개념
    - Two learning goals
        - Large margin classification
        - Good prediction: hinge loss
    - SVM Training
        - Optimization
    - Feature selection
    - Kernel의 활용

2. Python (Scikit-Learn)에서의 SVM

SVM (Support Vector Machine)

- A linear classifier, invented by V. Vapnik
  - V. Vapnik: the father of machine learning
  - Cortes, Corinna; Vapnik, Vladimir N., Support-vector net works, Machine Learning, 20 (3): 273–297 (1995)
  - The nature of statistical Learning Theory, Springer, 1999

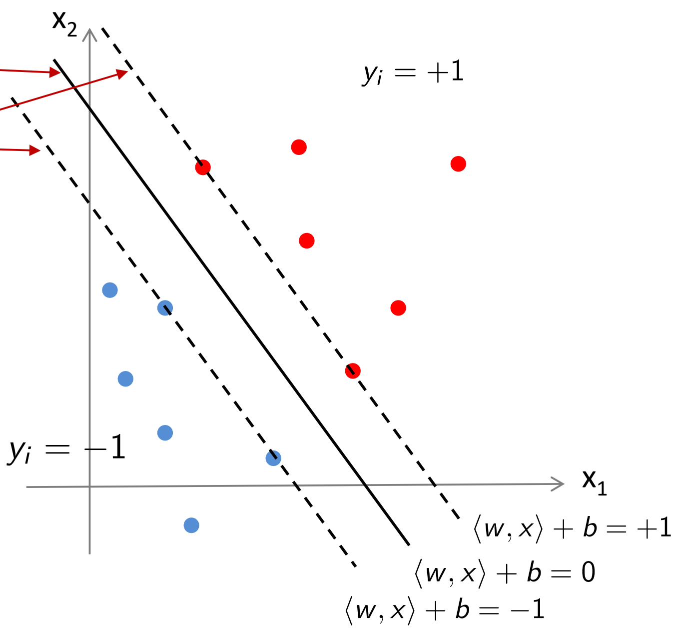
- ‘Margin’ between two class point clouds
  - **Minimize prediction error**
  - **+ maximize the margin** at the same time



- SVM hardly overfits, under regular conditions
- SVM can use **kernels**

SVM

- Data
  - Input: $x \in \mathbb{R}^p$ ($x \in \mathcal{X}$ in general)
  - Label: $y \in \{-1, +1\}$

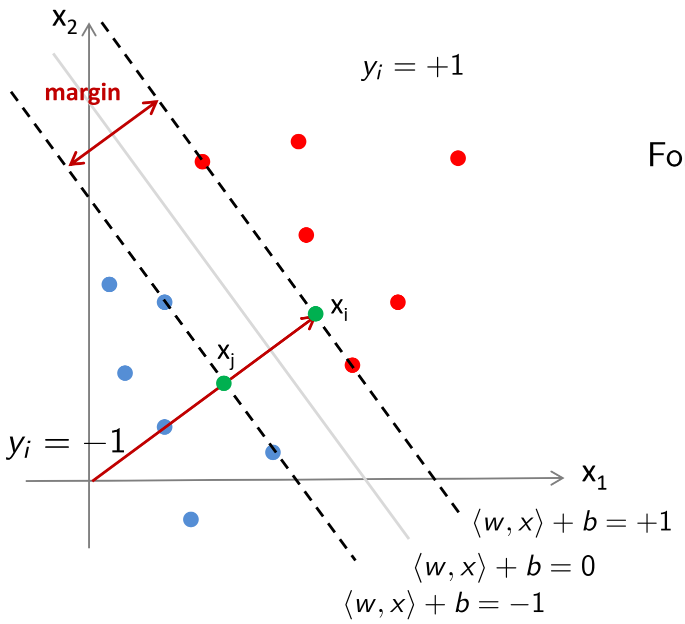


- Decision

$$f_w(x) = \mathrm{sgn}(\langle w, x \rangle + b)$$

$$\langle w, x \rangle = w^T x = \sum_{i=1}^{p} w_i x_i$$

Margin of SVM



$$\langle w, x_i\rangle + b = +1$$
$$\langle w, x_j\rangle + b = -1$$
$$\Rightarrow \langle w, x_i - x_j\rangle = 2$$

For $x_i - x_j = t\dfrac{w}{\|w\|_2},\ t \in \mathbb{R},$

$$t = \frac{2}{\|w\|_2}$$

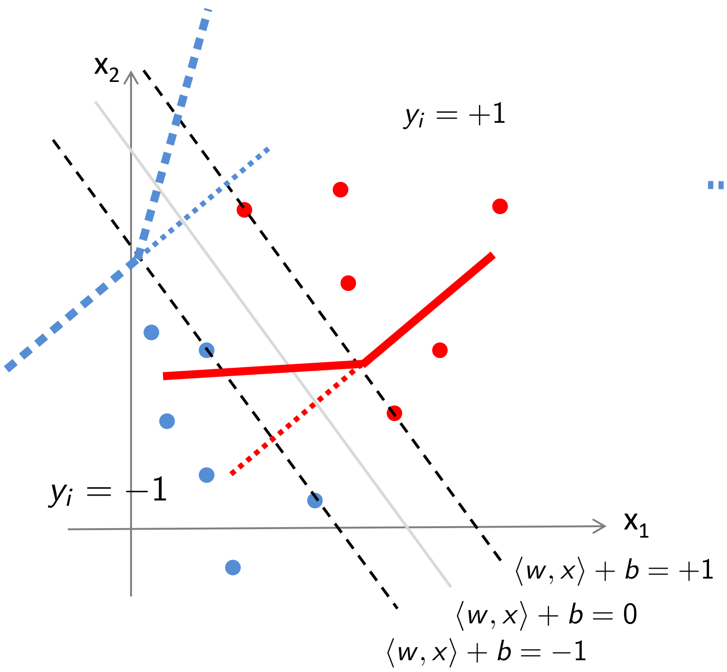
Learning Goal 1 (max. margin)

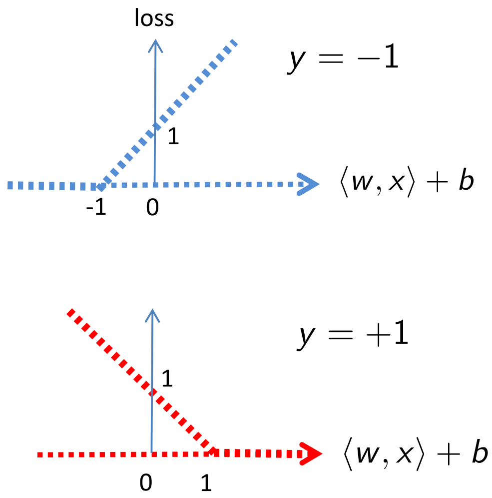
$$\min_w \ \frac{1}{2}\|w\|_2^2$$

Learning Goal 2 (Min. Prediction Error)

- Avoid having data points within the margin (as much as possible)

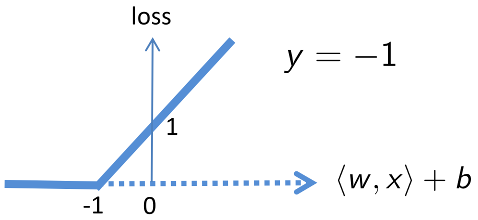
➔ Design a loss function

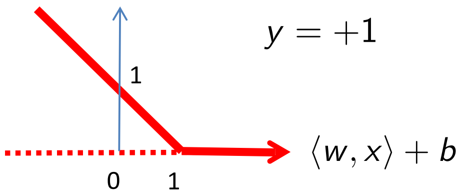


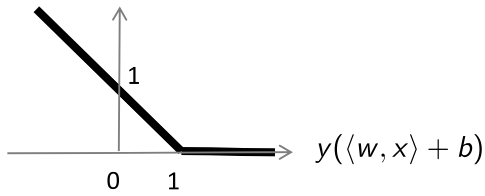


Learning Goal 2 (Min. Prediction Error)

- Hinge loss function







$$\max\{1 - y(\langle w, x \rangle + b), 0\}$$

Learning Goal 2 (min. pred error)

$$\min_{w,b} \sum_{i=1}^{n} \max\{1 - y_i(\langle w, x_i \rangle, b), 0\}$$

SVM Training Problem

$$\min_{w \in \mathbb{R}^p, b} \frac{1}{2}\|w\|_2^2 + C \sum_{i=1}^{n} \max\{1 - y_i(\langle w, x_i \rangle, b), 0\}$$

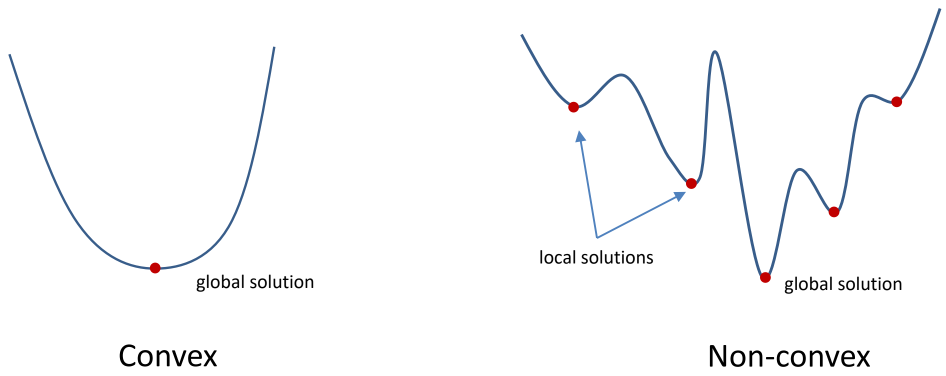
- n: number of observations (patients, etc)
- p: number of input features (genes, etc.)
- C > 0: a hyper-parameter for balancing the two learning goals

- This can be solved using SGD-type algorithms
  - Useful when n is huge
  - Pegasos solver [Shai Shalev-Shwartz et al., 2010]

- However, it is convex optimization, where better alternatives exist
  - When n is in a modest range, non-SGD type algorithms will be better (n < 100,000)
    - Finds the global minimizer (important for feature selection)
  - Special solvers are available when p is large
  - When data points x is NOT in a vector form ➔ kernel trick is available

Convex vs. Non-convex Optimization

- ML methods
  - Based on convex optimization
    - Logistic regression, SVM, …
  - Based on non-convex optimization
    - Deep neural nets with nonlinear activations (DNN, CNN, RNN, …)
    - NMF
- Shape of loss functions



SVM Smooth Formulation

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^n \max\{1 - y_i(\langle w, x_i\rangle, b), 0\}$$

➔

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}, \xi\in\mathbb{R}^n} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^n \xi_i$$

$$\text{subject to } \xi_i = \max\{1 - y_i(\langle w, x_i\rangle, b), 0\}, \; i = 1, \ldots, n$$

➔

$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}, \xi\in\mathbb{R}^n} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^n \xi_i$$

$$\text{subject to } \xi_i \geq 1 - y_i(\langle w, x_i\rangle, b), \; i = 1, \ldots, n$$

$$\xi_i \geq 0, \; i = 1, \ldots, n$$

"The C-SVM"

C-SVM

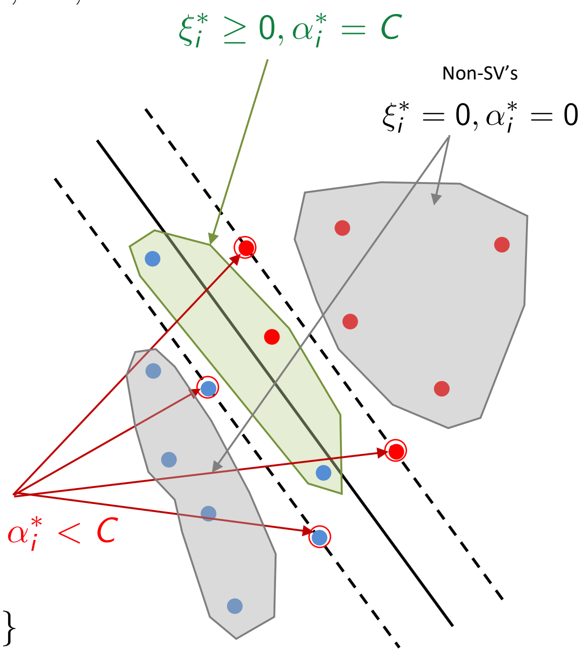
$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}, \xi\in\mathbb{R}^n} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } \xi_i \geq 1 - y_i(\langle w, x_i\rangle, b),\ i = 1, \dots, n$$
$$\xi_i \geq 0,\ i = 1, \dots, n$$

KKT (Karush-Kuhn-Tucker) conditions:
the conditions satisfied by the
solution

Insights from the KKT conditions:

$$w^* = \sum_{i=1}^{n} \alpha_i^* x_i \qquad \underset{\text{dual variable}}{0 \leq \alpha_i^* \leq C}$$



Support vector set: $\{i \in \{1, \dots, n\} : \alpha_i^* > 0\}$

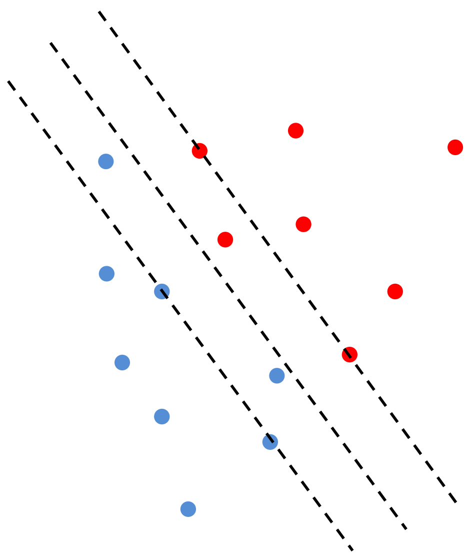
C-SVM

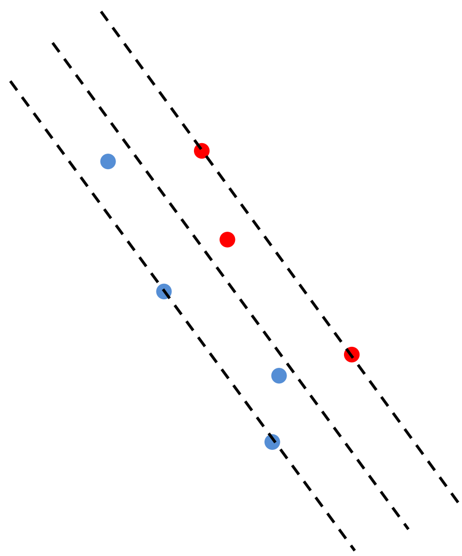
$$\min_{w\in\mathbb{R}^p, b\in\mathbb{R}, \xi\in\mathbb{R}^n} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } \xi_i \geq 1 - y_i(\langle w, x_i\rangle, b),\ i = 1, \dots, n$$
$$\xi_i \geq 0,\ i = 1, \dots, n$$

$$w^* = \sum_{i\in SV} \alpha_i^* x_i$$

b* can be computed using only SV's

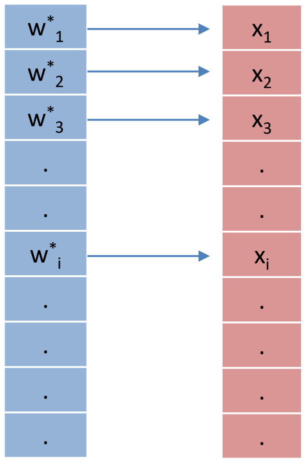


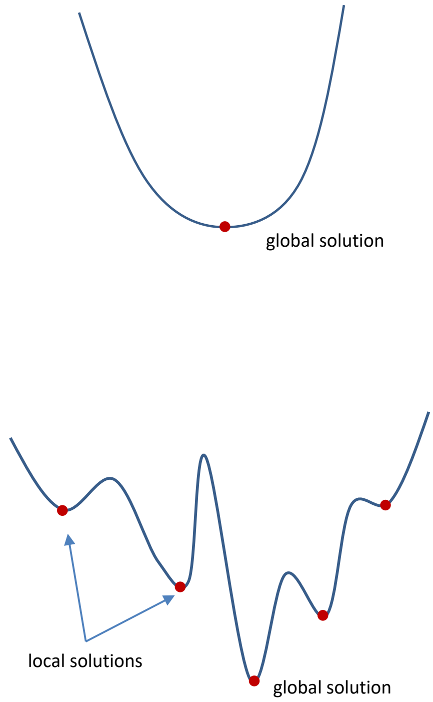


The same classifier will be obtained!

Feature Selection

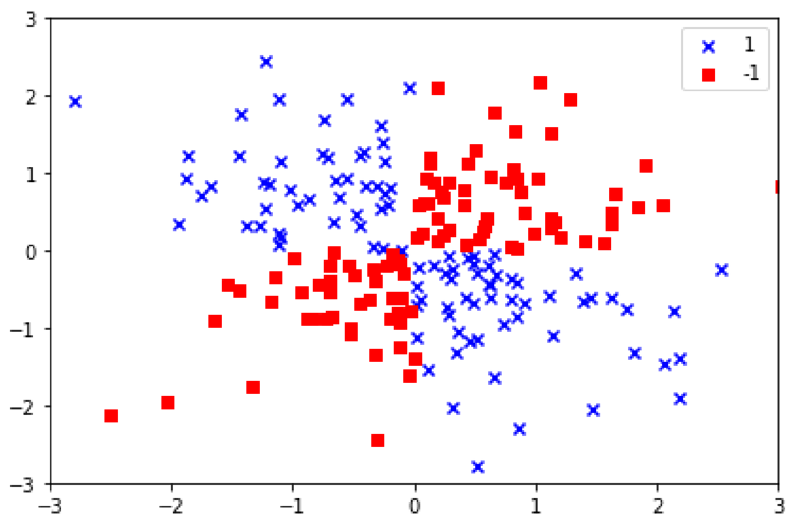
- SVM decision function
  - $f_w(x) = \text{sgn}(\langle w^*, x \rangle + b^*)$
  - Each weight (coefficient) represents the contribution of each input feature to th e decision



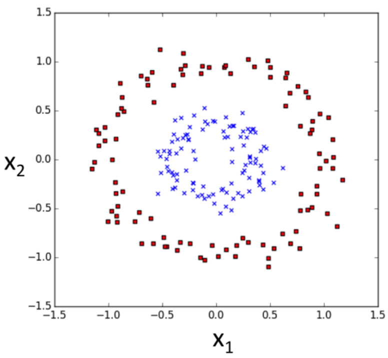


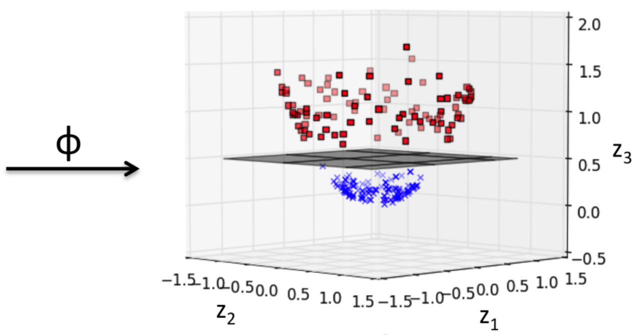
Nonlinear Classification Problems

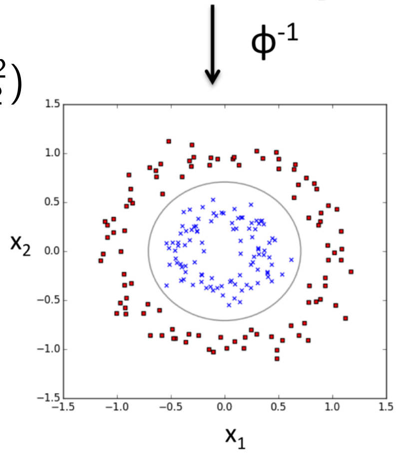
- Data with XOR class pattern



Feature Map





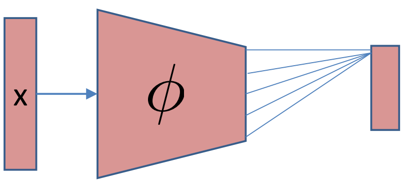


$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$$

We change the representation of data

Representation Learning

Deep neural nets:

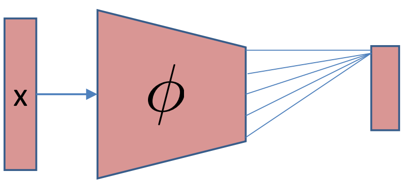


Network

$$\sigma\left(\sum_i w_i h_i + b\right)$$

Decision is a linear function of the final hidden outputs

SVM:



Feature map

$$\text{sgn}(\langle w_i, \phi(x)\rangle + b)$$

Decision is a linear function of the transformed input

Kernel Trick

- It is hard to design the feature map directly

- Instead, define the kernel:

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

$$K \in \mathbb{R}^{n \times n}$$

Kernel (aka Gram) matrix

- Popular choices
  - Linear kernel $k(x_i, x_j) = \langle x_i, x_j \rangle$
  - Polynomial kernel $k(x_i, x_j) = (\langle x_i, x_j \rangle + 1)^d$
  - Gaussian (RBF: radial basis function) kernel $k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|_2^2)$
  - Sigmoid kernel $k(x_i, x_j) = \tanh(b \langle x_i, x_j \rangle + a)$

Supplementary Notes

https://github.com/sanglee/BIML2021

BIML21_SVM.ipynb

File Edit View Insert Runtime Tools Help Last saved at 4:12 PM

+ Code + Text

BIML 2021 (Introduction to SVM)

@author: Sangkyun Lee (sangkyun@korea.ac.kr)

Supplementary Notes

```
# A utility function to plot decision boundaries in 2D

from matplotlib.colors import ListedColormap


def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    for idx, cl in enumerate(np.unique(y)):
```

Loading the IRIS data

```
import pandas as pd

df = pd.read_csv('https://archive.ics.uci.edu/ml/'
        'machine-learning-databases/iris/iris.data', names= ['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width', 'Label'])
```

```
df.head()
```

	Sepal Length	Sepal Width	Petal Length	Petal Width	Label
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
[ ] # Prepare data matrix and labels

    import numpy as np
    from sklearn.preprocessing import LabelEncoder

    feat_idx = [2,3]
    feat_labels = df.columns[feat_idx]
    X = df.iloc[:, feat_idx].values
    y = df.iloc[:, 4].values
    enc_y = LabelEncoder()
    enc_y.fit(y)
    y = enc_y.transform(y)
    feature_names = df.columns.values[[2,3]]

    from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=0)
    X_train, X_val, y_train, y_val = train_test_split(
        X_train, y_train, test_size=0.3, random_state=0)

    from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    sc.fit(X_train)
    X_train_std = sc.transform(X_train)
    X_val_std = sc.transform(X_val)
    X_test_std = sc.transform(X_test)
```

Support Vector Machine (SVM)

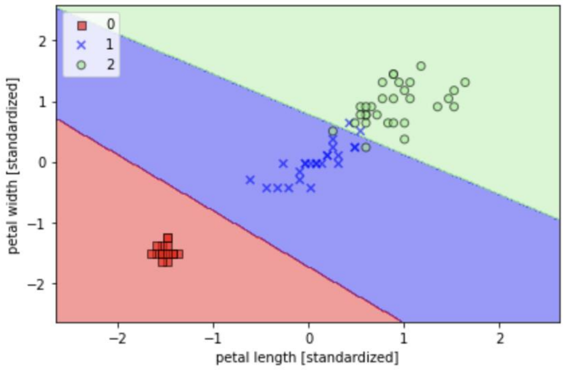
Multi-class Classification

```
[ ] from sklearn.svm import SVC
    X = np.random.randn(4, 1)
    y = [1,2,3,4]

    svm = SVC(decision_function_shape='ovr', kernel='linear', C=10.0)
    svm.fit(X, y)

    dec = svm.decision_function([X[0,:]])
    print("No. of classifiers in OVR:", dec.shape[1])

    svm = SVC(decision_function_shape='ovo', kernel='linear', C=10.0)
    svm.fit(X, y)

    dec = svm.decision_function([X[0,:]])
    print("No. of classifiers in OVO:", dec.shape[1])
```

```
No. of classifiers in OVR: 4
No. of classifiers in OVO: 6
```

Linear SVM on IRIS dataset

```
[ ] from sklearn.svm import SVC
    import matplotlib.pyplot as plt
    from sklearn.metrics import accuracy_score

    svm = SVC(decision_function_shape='ovr', kernel='linear', C=10.0)

    svm.fit(X_train_std, y_train)          # standardization is recommended for numerical reasons here

    plot_decision_regions(X_train_std,
                          y_train,
                          classifier=svm)
    plt.xlabel('petal length [standardized]')
    plt.ylabel('petal width [standardized]')
    plt.legend(loc='upper left')
    plt.tight_layout()
    plt.show()

    print('Train accuracy: %f' % accuracy_score(svm.predict(X_train_std), y_train))
    print('Test accuracy: %f' % accuracy_score(svm.predict(X_test_std), y_test))
```




```
Train accuracy: 0.945205
Test accuracy: 0.977778
```

Feature Importance

- Linear models (logistic regression, linear SVM, perceptron, ...)
  - Prediction for an input $x^{(i)}$ is produced by $\hat{y}^{(i)} = f(w^T x^{(i)})$, where f is some function, e.g., to convert linear output to binary outcome.
  - $w^T x^{(i)} = \sum_{j=1}^{p} w_j x_j^{(i)}$
  - We call $w_1, w_2, \ldots, w_p$ as **coefficients**, **weights**, or **learning parameters**
  - In scikit-learn, `<model>.coef_` contains the coefficients
- Non-linear models (decision trees, random forest, nonlinear SVM, ...)
  - This type of models provides other ways to measure feature importance
    - DT, RF: `<model>.feature_importances_`
  - In some models, it is not easy to extract feature importance
    - Nonlinear SVM

```
[ ] print(svm.coef_)

    imporant_feat = pd.DataFrame(np.abs(svm.coef_))
    imporant_feat.columns = feat_labels
    imporant_feat
```

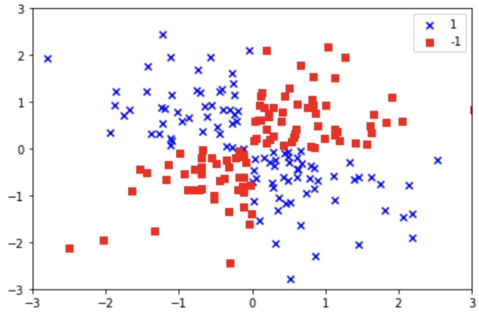
```
[[-1.06139348 -1.1489091 ]
 [-0.57330686 -0.57625103]
 [-2.65965384 -4.01574174]]
```

	Petal Length	Petal Width
0	1.061393	1.148909
1	0.573307	0.576251
2	2.659654	4.015742

Nonlinear Classification

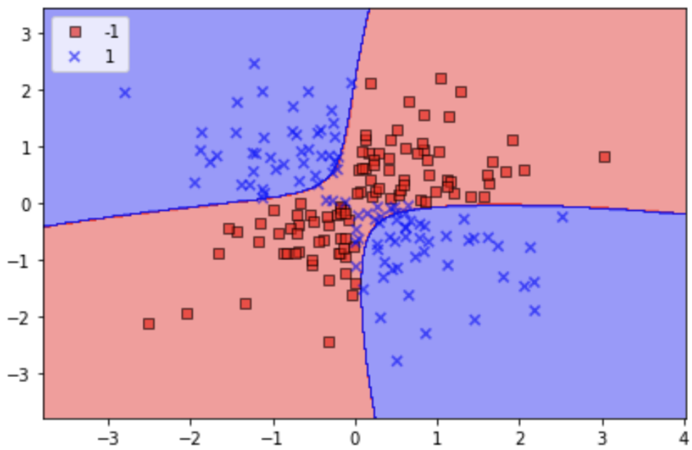
XOR Toy Data

```
[ ] import matplotlib.pyplot as plt
    import numpy as np

    np.random.seed(1)
    X_xor = np.random.randn(200, 2)
    y_xor = np.logical_xor(X_xor[:, 0] > 0,
                           X_xor[:, 1] > 0)
    y_xor = np.where(y_xor, 1, -1)

    plt.scatter(X_xor[y_xor == 1, 0],
                X_xor[y_xor == 1, 1],
                c='b', marker='x',
                label='1')
    plt.scatter(X_xor[y_xor == -1, 0],
                X_xor[y_xor == -1, 1],
                c='r',
                marker='s',
                label='-1')

    plt.xlim([-3, 3])
    plt.ylim([-3, 3])
    plt.legend(loc='best')
    plt.tight_layout()
    plt.show()
```



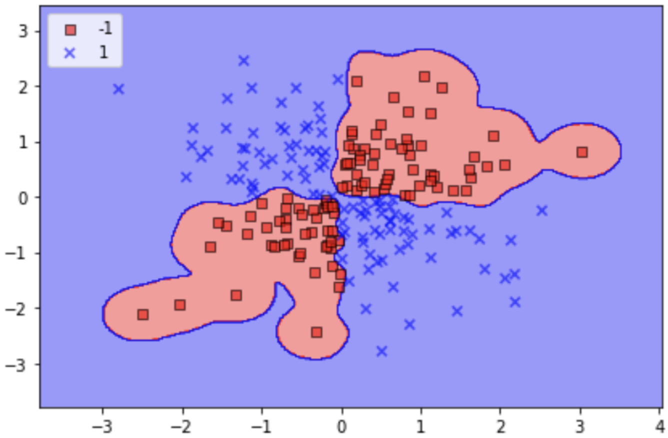
SVM on XOR data

```
#
# SVM on XOR data
#
svm = SVC(kernel='rbf', random_state=1, gamma=.1, C=10)
svm.fit(X_xor, y_xor)
plot_decision_regions(X_xor, y_xor,
                      classifier=svm)

plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



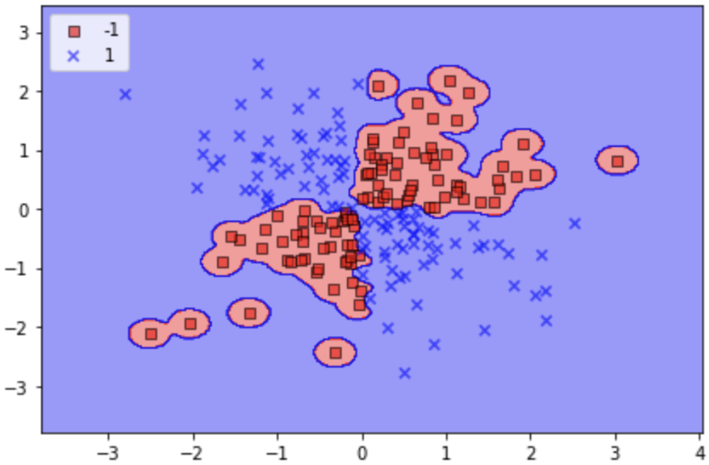
SVM on XOR data

```
[9]  #
     # SVM on XOR data
     #
     svm = SVC(kernel='rbf', random_state=1, gamma=10, C=10)
     svm.fit(X_xor, y_xor)
     plot_decision_regions(X_xor, y_xor,
                           classifier=svm)

     plt.legend(loc='upper left')
     plt.tight_layout()
     plt.show()
```



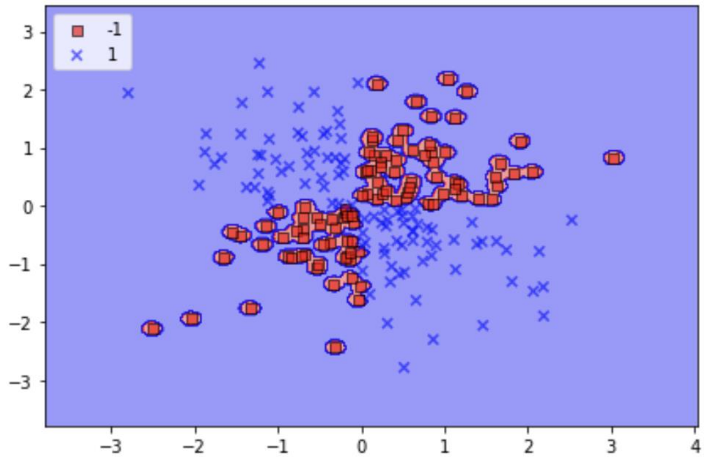
SVM on XOR data

```
[10] #
     # SVM on XOR data
     #
     svm = SVC(kernel='rbf', random_state=1, gamma=50, C=10)
     svm.fit(X_xor, y_xor)
     plot_decision_regions(X_xor, y_xor,
                           classifier=svm)

     plt.legend(loc='upper left')
     plt.tight_layout()
     plt.show()
```



SVM on XOR data

```
[13] #
     # SVM on XOR data
     #
     svm = SVC(kernel='rbf', random_state=1, gamma=250, C=10)
     svm.fit(X_xor, y_xor)
     plot_decision_regions(X_xor, y_xor,
                           classifier=svm)

     plt.legend(loc='upper left')
     plt.tight_layout()
     plt.show()
```



Support Vectors

```
[ ] print(svm.n_support_)
    svm.support_
```

```
[46 46]
array([  6,   8,   9,  13,  17,  20,  23,  24,  36,  40,  41,  42,  44,
        45,  47,  58,  60,  65,  69,  70,  72,  73,  78,  92,  93, 109,
       111, 113, 114, 115, 117, 120, 121, 125, 132, 133, 141, 142, 143,
       145, 147, 160, 166, 173, 190, 193,   4,  14,  18,  22,  25,  27,
        31,  32,  38,  43,  48,  52,  54,  77,  79,  81,  82,  86,  87,
        89,  94, 100, 108, 123, 127, 129, 130, 131, 134, 139, 144, 146,
       153, 158, 161, 164, 171, 172, 176, 178, 182, 185, 186, 192, 196,
       199], dtype=int32)
```

```
[ ] #svm.support_vectors_
```

```
[ ] # shape: n_classes-1, n_SV
    svm.dual_coef_.shape
```

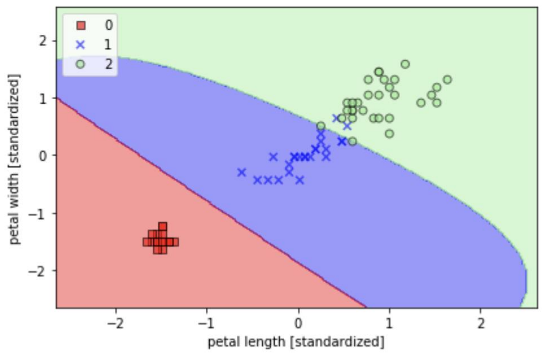
```
(1, 92)
```

```
[ ] svm.dual_coef_
```

```
array([[-10.        , -10.        , -10.        , -10.        ,
        -10.        , -10.        ,  -7.99986544, -10.        ,
        -10.        , -10.        , -10.        , -10.        ,
        -10.        ,  -6.20496694, -10.        , -10.        ,
        -10.        , -10.        , -10.        , -10.        ,
        -10.        , -10.        , -10.        , -10.        ,
        -10.        , -10.        , -10.        , -10.        ,
        -10.        , -10.        , -10.        ,  -1.56513305,
        -10.        , -10.        , -10.        , -10.        ,
        -10.        , -10.        , -10.        , -10.        ,
        -10.        , -10.        , -10.        , -10.        ,
        -10.        , -10.        ,  10.        ,  10.        ,
         10.        ,  10.        ,  10.        ,  10.        ,
         10.        ,  10.        ,  10.        ,  10.        ,
         10.        ,  10.        ,  10.        ,  10.        ,
         10.        ,  10.        ,  10.        ,   1.23159626,
         10.        ,  10.        ,  10.        ,  10.        ,
         10.        ,  10.        ,   9.95761928,  10.        ,
         10.        ,  10.        ,  10.        ,  10.        ,
         10.        ,  10.        ,  10.        ,  10.        ,
         10.        ,  10.        ,  10.        ,  10.        ,
         10.        ,  10.        ,  10.        ,  10.        ,
          4.58074989,  10.        ,  10.        ,  10.        ]])
```

SVM on IRIS data

```
[ ]
from sklearn.svm import SVC

svm = SVC(kernel='rbf', gamma=0.2, C=1.0)  # gamma=100.0, C=1.0
svm.fit(X_train_std, y_train)

plot_decision_regions(X_train_std, y_train, classifier=svm)
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()

print('Train accuracy: %f' % accuracy_score(svm.predict(X_train_std), y_train))
print('Test accuracy: %f' % accuracy_score(svm.predict(X_test_std), y_test))
```




```
Train accuracy: 0.945205
Test accuracy: 0.977778
```

Support Vectors

```
print(svm.n_support_)
svm.support_
```

```
[13 24 23]
array([ 3, 12, 21, 24, 29, 30, 33, 35, 36, 48, 50, 54, 60,  4,  6,  8, 15,
       19, 20, 22, 23, 34, 37, 41, 42, 44, 47, 49, 55, 61, 62, 63, 64, 65,
       66, 69, 70,  1,  2, 10, 11, 13, 14, 16, 17, 18, 26, 27, 31, 32, 38,
       39, 43, 51, 56, 59, 67, 68, 71, 72], dtype=int32)
```

```
svm.support_vectors_
```

```
array([[-1.36607872, -1.50714415],
       [-1.42392312, -1.50714415],
       [-1.53961192, -1.37297156],
       [-1.48176752, -1.23879897],
       [-1.48176752, -1.50714415],
       [-1.53961192, -1.50714415],
       [-1.42392312, -1.50714415],
       [-1.53961192, -1.50714415],
       [-1.42392312, -1.50714415],
       [-1.48176752, -1.23879897],
       [-1.48176752, -1.37297156],
       [-1.59745632, -1.37297156],
       [-1.48176752, -1.50714415],
       [ 0.08003129, -0.03124567],
       [-0.20919071, -0.43376344],
       [ 0.3114089 , -0.03124567],
       [ 0.25356449,  0.3712721 ],
       [-0.09350191, -0.29959085],
       [ 0.4270977 ,  0.63961727],
       [-0.32487951, -0.43376344],
       [ 0.25356449,  0.23709951],
       [-0.03565751, -0.03124567],
```

```
# shape: n_classes-1, n_SV
svm.dual_coef_.shape
```

```
(2, 60)
```

```
svm.dual_coef_
```

```
array([[ 1.        ,  1.        ,  1.        ,  1.        ,  1.        ,
         1.        ,  1.        ,  1.        ,  1.        ,  1.        ,
         1.        ,  1.        ,  1.        , -1.        , -1.        ,
        -0.        , -0.        , -1.        , -0.        , -1.        ,
        -0.        , -1.        , -1.        , -1.        , -0.        ,
        -1.        , -1.        , -0.        , -1.        , -0.        ,
        -1.        , -0.        , -1.        , -0.        , -1.        ,
        -0.        , -0.        , -0.        , -0.38609855, -0.        ,
        -0.        , -0.        , -0.        , -0.        , -0.        ,
        -0.        , -0.        , -1.        , -1.        , -1.        ,
        -1.        , -0.        , -0.        , -0.        , -0.        ,
        -1.        , -1.        , -0.        , -0.        , -0.        ],
       [ 1.        ,  0.        ,  1.        ,  1.        ,  0.        ,
         0.        ,  0.38609855,  0.        ,  1.        ,  1.        ,
         1.        ,  0.        ,  0.        ,  1.        ,  1.        ,
         1.        ,  1.        ,  1.        ,  1.        ,  1.        ,
         1.        ,  1.        ,  0.        ,  1.        ,  1.        ,
         1.        ,  1.        ,  1.        ,  1.        ,  1.        ,
         0.20659426,  1.        ,  1.        ,  1.        ,  1.        ,
         1.        ,  1.        , -1.        , -1.        , -1.        ,
        -0.20659426, -1.        , -1.        , -1.        , -1.        ,
        -1.        , -1.        , -1.        , -1.        , -1.        ,
        -1.        , -1.        , -1.        , -1.        , -1.        ,
        -1.        , -1.        , -1.        , -1.        , -1.        ]])
```

K-Fold Cross Valiation (CV)

K-Fold Cross Valiation (CV)

```
[ ] from sklearn.model_selection import cross_val_score

    C_grid = [0.1, 1, 10]
    cv_err = []

    for c in C_grid:
        scores = cross_val_score(estimator=SVC(C=c),
                                 X=X_train,
                                 y=y_train,
                                 cv=5,
                                 scoring='accuracy',
                                 n_jobs=-1)  # no. cpu cores to use. -1 all cores
        print('C = %f' % c)
        print('CV accuracy scores:\n %s' % scores)
        print('CV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
        print('---------------------------------------------------------------------')
        cv_err.append(np.mean(scores))

    best_idx = np.argmax(cv_err)
    print(best_idx)
    best_C = C_grid[best_idx]

    print('Best C due to CV: %f' % best_C)
```

```
C = 0.100000
CV accuracy scores:
 [1.         0.93333333 0.93333333 0.86666667 1.        ]
CV accuracy: 0.947 +/- 0.050
---------------------------------------------------------------------
C = 1.000000
CV accuracy scores:
 [1.         0.93333333 0.93333333 0.86666667 1.        ]
CV accuracy: 0.947 +/- 0.050
---------------------------------------------------------------------
C = 10.000000
CV accuracy scores:
 [1.         0.93333333 0.86666667 0.86666667 1.        ]
CV accuracy: 0.933 +/- 0.060
---------------------------------------------------------------------
0
Best C due to CV: 0.100000
```

본 강의 자료는 한국생명정보학회가 주관하는 KSBi-BIML 2021 워크샵 온라인 수업을 목적으로 제작된것으로 해당 목적 이외의 다른 용도로 사용할 수 없음을 분명하게 알립니다. 수업 목적으로 배포 및 전송 받은 경우에도 이를 다른 사람과 공유하거나 복제, 배포, 전송할 수 없습니다.

만약 이러한 사항을 위반할 경우 발생하는 모든 법적 책임은 전적으로 불법 행위자 본인에게 있음을 경고합니다.