Classification (Advanced)

Dr. Saerom Park
Statistical Learning and Computational Finance Lab.
Department of Industrial Engineering
psr6275@snu.ac.kr
http://slcf.snu.ac.kr

목차

Reference

- **Reading:** [Raschka. (2017), chapter 3], [Müller. (2016), chapter 2], [GÉRON. (2017), chapter 3 & 5 & 6]

분류 문제

- 지도 학습
  - 지도 학습은 입력과 출력 데이터가 있고 주어진 입력으로부터 출력을 예측하기 위한 학습 방법임
    - 분류 문제: 미리 정의된 여러 클래스 레이블 중 하나를 예측하는 것
    - 회귀 문제: 연속적인 숫자 혹인 실수를 예측하는 것
  - 지도학습에서는 훈련데이터로 학습한 모델이 훈련 데이터와 특성이 같다면 처음 보는 새로운 데이터가 주어져도 정확히 예측할 것이라 기대
    - 일반화 (generalization) / 과대적합 (overfitting) vs. 과소적합 (underfitting)
- No Free Lunch
  - 모든 시나리오에 대해 최고 성능인 단일 분류기(classifier)는 없음
  - 여러 알고리즘들의 성능을 비교: confusion matrix, precision, recall
- 분류 문제와 scikit-learn
  - 지도 학습 중 **분류 문제**의 경우에 예측 값을 다음의 두 함수 중 하나로 제공
    - decision_function: 예측 값으로 분류된 클래스 라벨을 제공
    - predict_proba: 예측 값으로 각 클래스 라벨일 확률을 제공

분류 알고리즘 선택

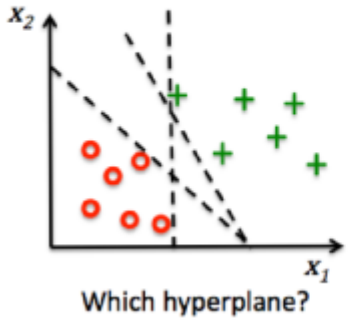
분류 알고리즘 선택

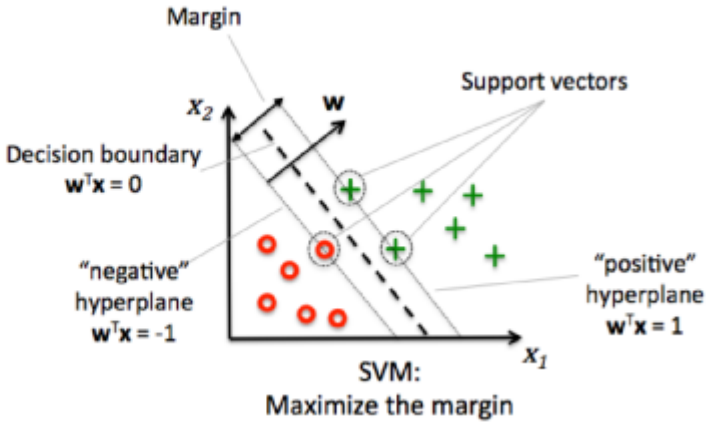
- 알고리즘 learning process
  - Feature 선택
  - Metric 선택
  - **분류 알고리즘 선택**
  - 최적화 알고리즘 선택
  - 모델 성능 평가
  - 알고리즘 튜닝
- 분류 알고리즘
  - 퍼셉트론(Perceptron)
  - Adaline
  - 로지스틱 회귀(Logistic Regression)
  - 서포트 벡터 머신(Support Vector Machine)
  - 의사결정 나무(Decision Tree)
  - KNN(k nearest neighbors)

SUPPORT VECTOR MACHINE

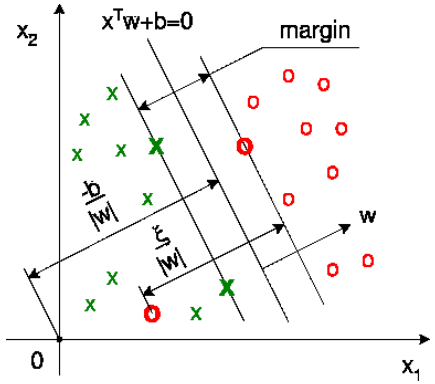
서포트 벡터 머신(Support Vector Machine)

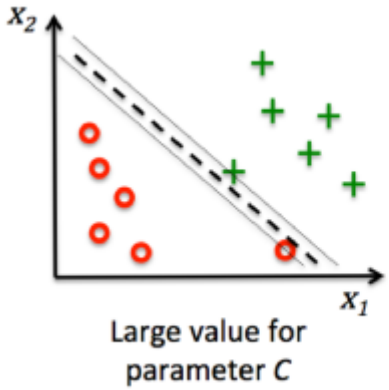
- 마진 최대화

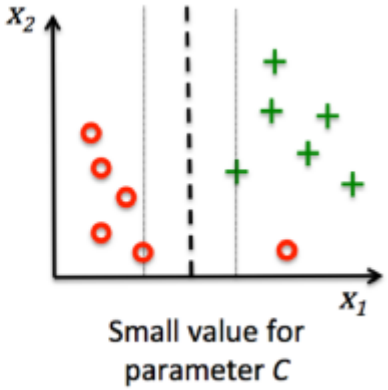




- 여유 변수(Slack variable)







서포트 벡터 머신(Support Vector Machine)

- SVM 모델 학습 (선형 분리 가능)
  - 예측 모델: $f(x) = w^T x + b$
  - SVM은 예측 모델이 데이터와의 마진이 최대가 되도록 학습
    - $margin = \frac{(x_+ - x_-)^T w}{2\|w\|} = \frac{1}{\|w\|}$
  - 다음의 최적화 문제를 통해 마진을 최대화 하는 모델 파라미터를 구함
    - $w^* = argmax_{w,b} \frac{1}{\|w\|}\ subject\ to\ y_j(x_j^T w + b) \geq 1$
    - $w^* = argmin_{w,b} \frac{1}{2}\|w\|^2\ subject\ to\ y_j(x_j^T w + b) \geq 1$
  - 최적화 문제를 풀기 위한 Lagrangian function은 다음과 같음
    - $L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i (y_i(x_i^T w + b) - 1)$
      - $\partial_w L = 0 \rightarrow w = \sum_i \alpha_i y_i x_i$
      - $\partial_b L = 0 \rightarrow \sum_i \alpha_i y_i = 0$
  - 라그랑지안을 통한 쌍대 문제:
    - $\alpha^* = argmin_\alpha -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i\ \ subject\ to\ \sum_i \alpha_i y_i = 0, \alpha_i \geq 0$ (Quadratic problem)

서포트 벡터 머신(Support Vector Machine)

- SVM 모델 학습 (선형 분리 불가능)
  - 앞의 문제에 여유 변수(Slack variable)를 도입함으로써 해결
    - $w^* = argmin_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \ subject\ to\ y_j\left(x_j^T w + b\right) \geq 1 - \xi_i \ and\ \xi_i \geq 0$
  - 새로운 문제의 라그랑지안은 다음과 같음
    - $L(w,b,\alpha) = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\left(y_i\left(x_i^T w + b\right) + \xi_i - 1\right) - \sum_i \eta_i \xi_i$ (Lagrangian)
      - $\partial_w L = 0 \ \rightarrow w = \sum_i \alpha_i y_i x_i$
      - $\partial_b L = 0 \ \rightarrow \sum_i \alpha_i y_i = 0$
      - $\partial_{\xi_i} L = 0 \ \rightarrow C - \alpha_i - \eta_i = 0$
  - 라그랑지안을 통한 쌍대 문제:
    - $\alpha^* = argmin_\alpha -\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \ subject\ to\ \sum_i \alpha_i y_i = 0, 0 \leq \alpha_i \leq C$

서포트 벡터 머신(Support Vector Machine)

- scikit-learn 이용 서포트 벡터 머신(Support Vector Machine)
  - `from sklearn.svm import SVC`
  - `SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)`

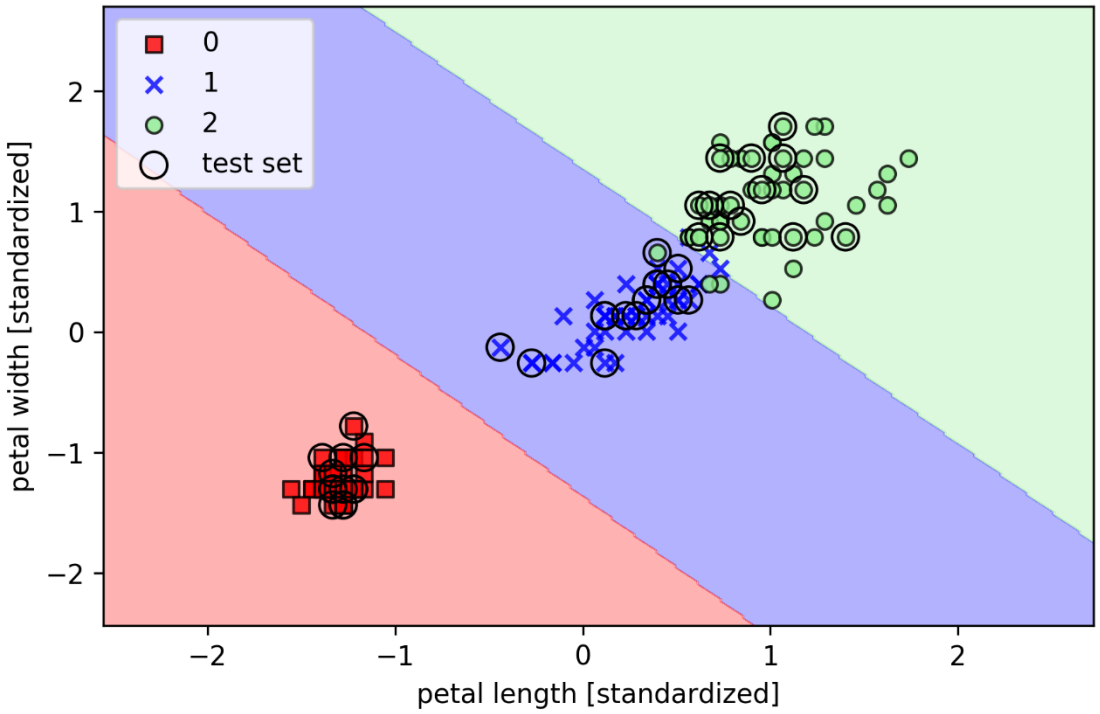
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Parameters	
kernel	알고리즘에 사용하는 kernel('linear', 'poly', 'rbf')
tol	Tolerence
C	여유변수의 Penalty parameter
max_iter	최대의 iteration 횟수
decision_function_shape	Multi-class classification 방법

Attributes	
coef_	Weight (linear kernel의 경우)
support_vectors_	Support vectors

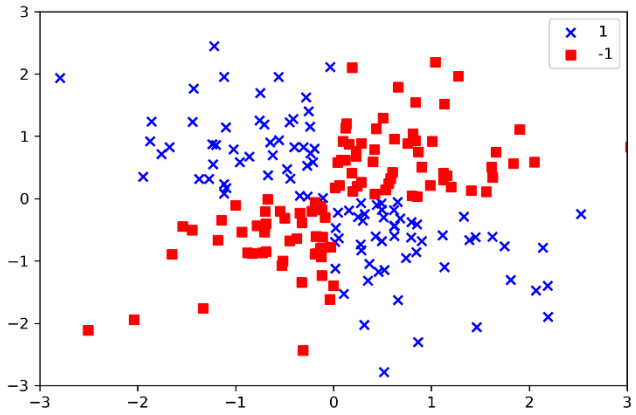
서포트 벡터 머신(Support Vector Machine)

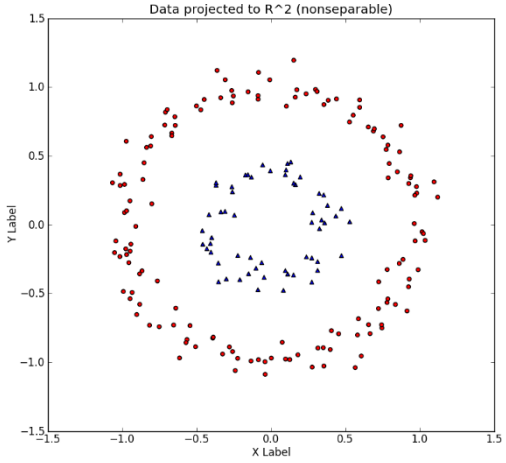
- scikit-learn 이용 서포트 벡터 머신(Support Vector Machine)
  - `from sklearn.svm import SVC`
  - `SVC(kernel='linear')`



커널 SVM

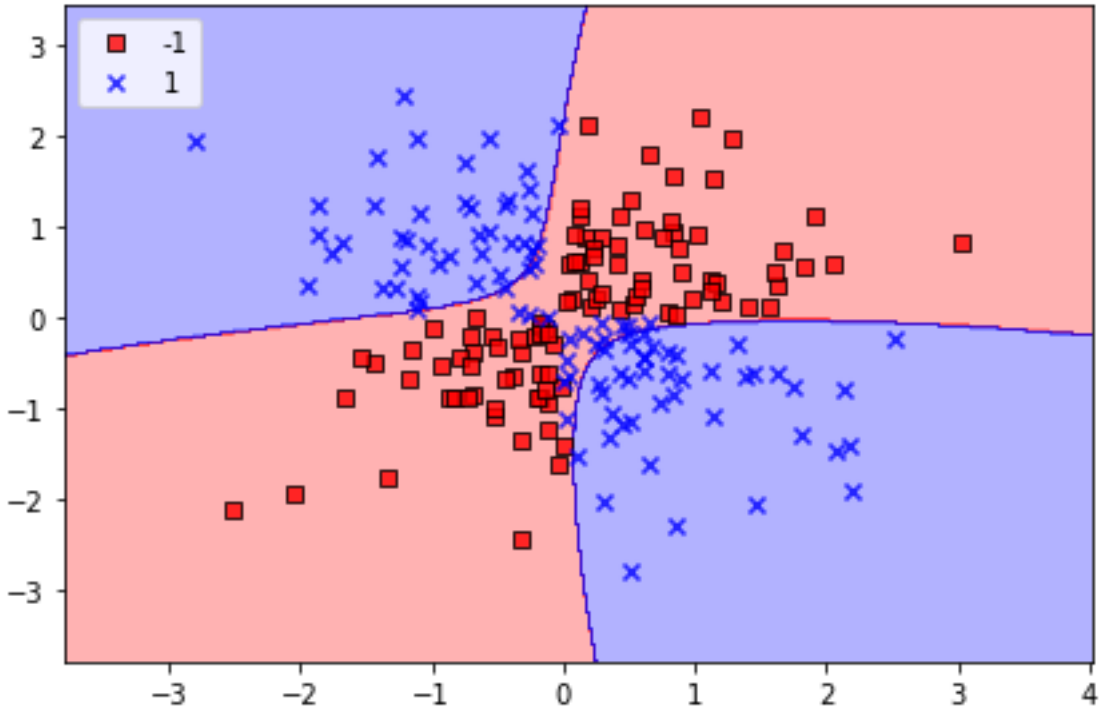
- 커널 SVM
  - 선형으로 분리되지 않는 데이터
  - 선(평면)이 아닌 곡선(곡면)의 분류 경계를 사용
  - 일반적으로 사용되는 커널 함수
    - Linear: $k(a,b) = a^T b$
    - Polynomial: $\mathrm{k(a,b)} = (\gamma a^T b + r)^d$
    - Gaussian RBF: $k(a,b) = \exp\left(-\gamma \left|\left|a-b\right|\right|^2\right)$
    - Sigmoid: $k(a,b) = \tanh(\gamma a^T b + r)$





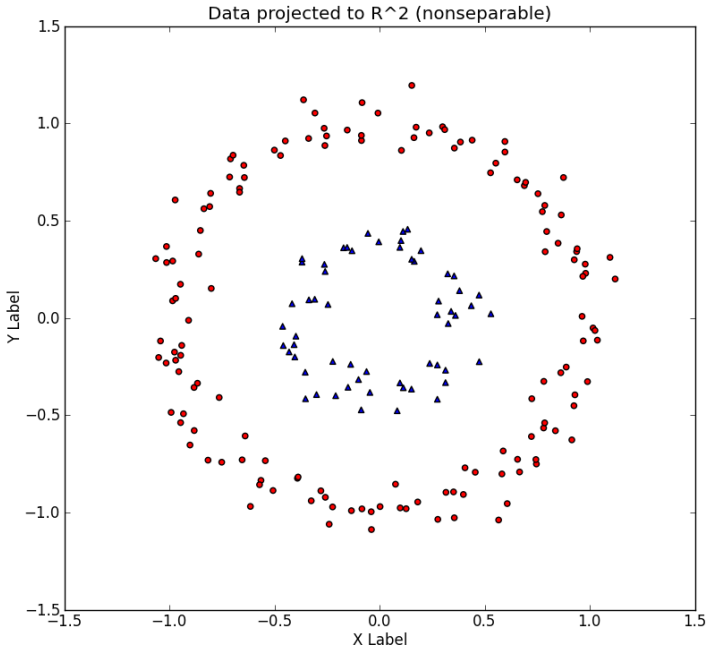
커널 SVM

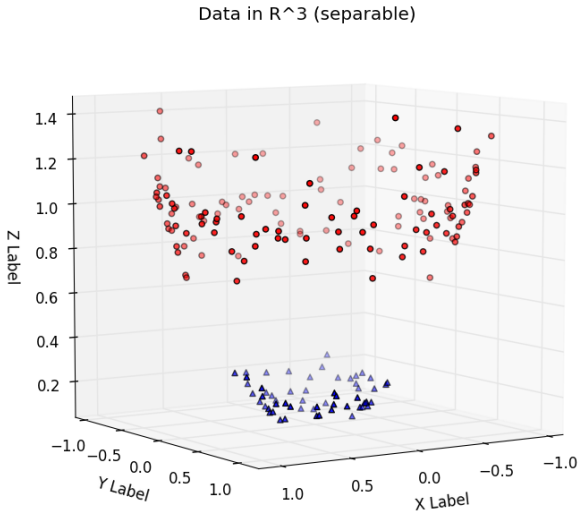
- scikit-learn 이용 서포트 벡터 머신(Support Vector Machine)
  - `from sklearn.svm import SVC`
  - `SVC(kernel='rbf')`



커널 SVM

- scikit-learn 이용 서포트 벡터 머신(Support Vector Machine)
  - `from sklearn.svm import SVC`
  - `SVC(kernel='rbf')`





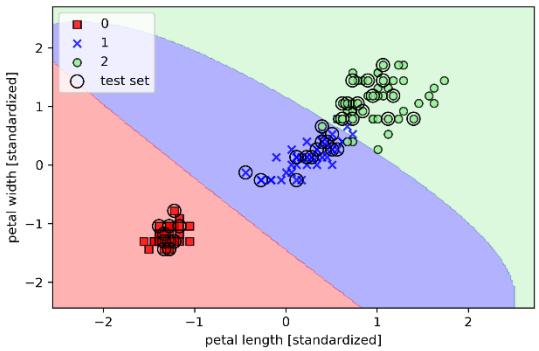
커널 SVM

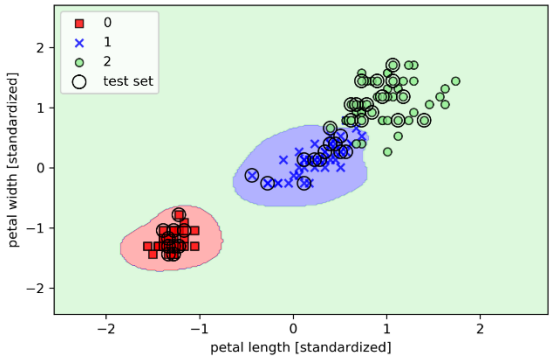
- 커널 함수(Kernel functions)
  - Similarity function
  - 자주 쓰이는 커널 중 하나는 **Radial Basis Function(RBF)** = Gaussian kernel:

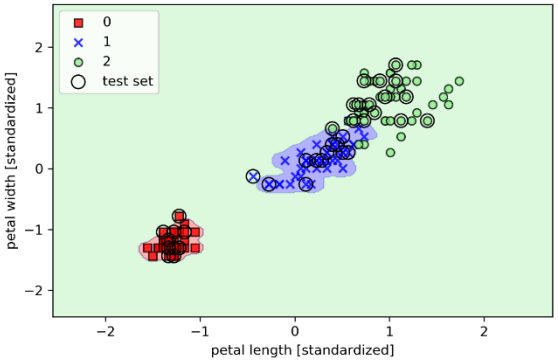
$$k\left(x^{(i)}, x^{(j)}\right) = \exp\left(-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2\sigma^2}\right)$$

$$\gamma = \frac{1}{2\sigma},\ then\ \ k\left(x^{(i)}, x^{(j)}\right) = \exp\left(-\gamma\left\|x^{(i)} - x^{(j)}\right\|^2\right)$$

  - SVC(**kernel = 'rbf'**, gamma)
  - Gamma가 클수록 kernel값이 조금만 떨어진 점에 대해서도 다르게 평가



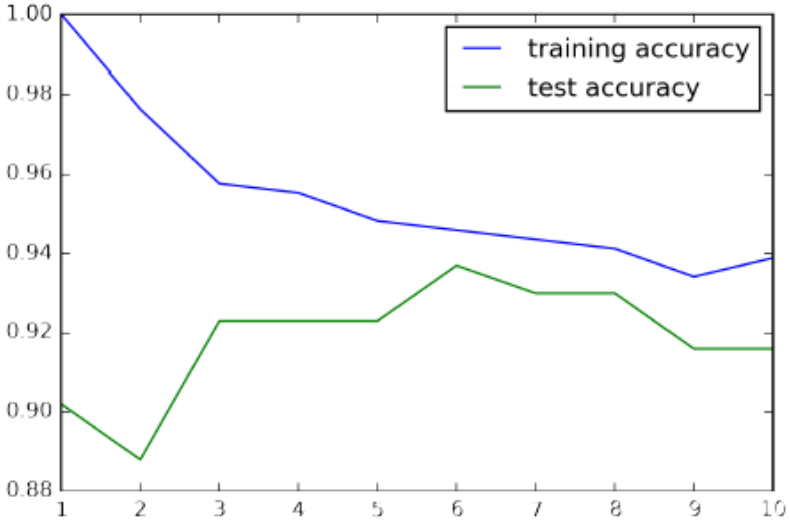




K-NEAREST NEIGHBORS

KNN(k-nearest neighbors)

- KNN 분류기
  - Lazy learning = 학습하는 것이 아니라 데이터를 외움
  - 주어진 metric을 이용하여 k개의 가장 '가까운' 점(training data)들이 어떻게 분류 되었는지를 통해 다수결로 분류
    - 민코우스키(Minkowski) 거리
      - $d(x,y) = \left(\sum_j (x_j - y_j)^p\right)^{\frac{1}{p}}$
    - 맨해튼 거리(시티 블록거리)
      - $d(x,y) = \sum_{j=1}^{m} |x_j - y_j|$
    - 최대좌표거리
      - $d(x,y) = max_{\ j=1,2,...m} |x_j - y_j|$
    - 통계적 거리(마할라노비스 거리)
      - $d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$
      - S는 공분산행렬



  - 학습 데이터의 경우에도 근접 이웃들에 따라서 클래스 라벨이 결정
    - K=1 일때 학습 정확도가 가장 높음 (모델 복잡도도 가장 높음)
    - 너무 큰 K일 경우에는 모델이 너무 단순해져 성능 감소

KNN(k nearest neighbors)

- scikit-learn 이용 KNN
  - `from sklearn.neighbors import KNeighborsClassifier`
  - `KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs)`

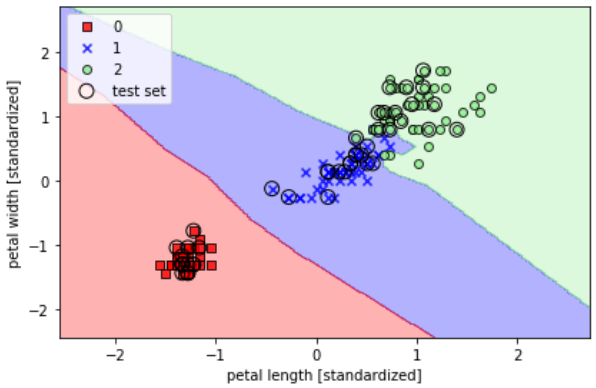
http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

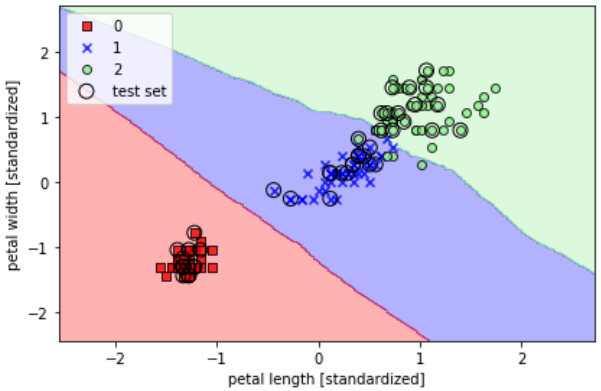
Parameters	
n_neighbors	k에 해당하는 neighbors 수
metric	알고리즘에 사용하는 metric('minkowski')
p	minkowski에 사용하는 power
n_jobs	병렬처리 방법

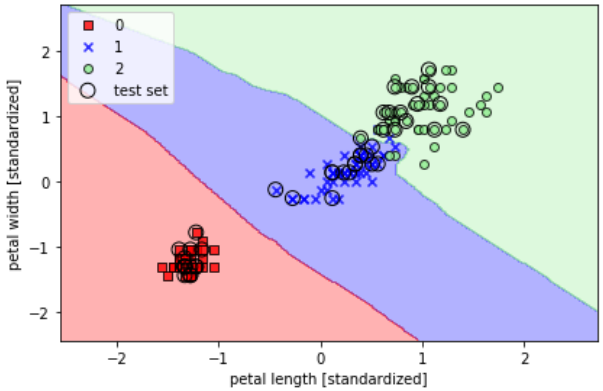
KNN(k nearest neighbors)

- scikit-learn 이용 KNN
  - `from sklearn.neighbors import KNeighborsClassifier`
  - `KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs)`

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html







Multiclass classifier

- One-versus-all(OvA) or One-versus-rest(OvR)
  - 분류할 class 수만큼의 binary classifier를 사용한다.
  - 각각 classifier는 하나의 class에 대응하며 각 class의 원소인지 아닌지를 판별한다.
  - Ex) digits classifier의 경우 0~9의 digits을 판별하는데
    10개의 classifier를 이용하여 각각이 0-detector, 1-detector, …의 역할을 한다.
  - 필요한 classifier 수가 적기 때문에 대부분 classifier의 default 값이다.
- One-versus-one(OvO)
  - 분류할 class 수가 N이라고 하면 N(N-1)/2개의 binary classifier를 사용한다.
  - 각각 classifier는 두개의 class에 대응하며 두 class 중 어느 class로 배정할지를 결정한다.
  - Ex) digit classifier의 경우
    10*9/2=45개의 classifier를 이용하여 각각이 '0,1을 비교', '0,2를 비교', …, '8,9를 비교 ' 하는 classifier의 역할을 한다.
  - 각 classifier에 대해 필요한 training set이 작기 때문에 SVM에서 종종 쓰임

Reference

- [Raschka. (2017)] Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning*. Packt Publishing Ltd, 2017.
- [Müller. (2016)] Müller, Andreas C., and Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. 2016.
- [GÉRON. (2017)] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* Sebastopol, CA: O, 2017.