Day 14

OpenCV

Python Regular Expression

- A sequence of characters that forms a search pattern. 검색패턴을 구성하는 글자의 시퀀스
- Used to check if a string contains the specified search pattern. 글자에 특정 검색패턴이 있는지 체크하는 사용

```
import re
txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)
```

RegExp Metacharacters

Example	Description
"[a-m]"	A set of characters 그중에 하나
"\d"	Signals a special sequence or used to escape special characters 특정시퀀스의 시작
"he..o"	Any character (except newline character) 아무 글자나
"^hello"	Starts with 그 글로 시작
"world$"	Ends with 그 글로 끝
"aix*"	Zero or more occurrences 0개이상
"aix+"	One or more occurrences 1개이상
"al{2}"	Exactly the specified number of occurrences 정확히 그 숫자만큼
"falls|stays"	Either or 둘중에 하나
()	Capture and group

Special Sequences

Example	Description
"\AThe"	Returns a match if the specified characters are at the beginning of the string 그글자가 시작이면 반환
r"\bain" r"ain\b"	Returns a match where the specified characters are at the beginning or at the end of a word 시작이거나 끝이면 반환
r**"**\Bain" r"ain\B"	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word 시작과 끝 상관없이 그냥 있으면 반환
"\\d"	Returns a match where the string contains digits (numbers from 0-9) 문자열안에 숫자를 반환
"\\D"	Returns a match where the string DOES NOT contain digits 문자열안에 숫자가 아닌것을 반환
"\\s"	Returns a match where the string contains a white space character 공간문자를 반환
"\\S"	Returns a match where the string DOES NOT contain a white space character 공간문자가 없는것을 반환
"\\w"	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character) 글자가 있으면 반환
"\\W"	Returns a match where the string DOES NOT contain any word characters 글자가 없으면 반환
"Spain\Z"	Returns a match if the specified characters are at the end of the string 특정 글자가 마지막에 있으면 반환

Sets

Set	Description
[arn]	Returns a match where one of the specified characters (a, r, or n) are present **a 또는 r 또는 n이 있으면 반환**
[a-n]	Returns a match for any lower case character, alphabetically between a and n **a와 n사이의 소문자 알파벳이 있으면 반환**
[^arn]	Returns a match for any character EXCEPT a, r, and n **a,r,n을 제외한 글자가 있으면 반환**
[0123]	Returns a match where any of the specified digits (0, 1, 2, or 3) are present **0,1,2,3중 숫자가 있으면 반환**
[0-9]	Returns a match for any digit between 0 and 9 **0-9사이의 숫자면 반환**
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59 **0에서 59사이의 2글자 숫자면 반환**
[a-zA-Z]	Returns a match for any character alphabetically between a and z, lower case OR upper case **소문자 대문자 알파벳 반환**
[+]	In sets, +, *, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string **특정 심볼이 있으면 반환**

OpenCV (Open Source Computer Vision)

- A library of programming functions mainly aimed at real-time computer vision. 실시간 컴퓨터비젼을 구현하기 위한 라이브러리
- Used for Image Processing. Can do all the operation related to Images. 이미지 프로세싱에 사용되며 이미지와 관련된 모든 오퍼레이션이 가능함

Installing OpenCV

- Window installation

```
pip install
opencv-contrib-python
```

- Test your installation

```
import cv2
cv2.__version__
```

- Linux installation
  - Download latest OpenCV release from sourceforge site.

```
conda install -c
conda-forge opencv for
anaconda
```

```
pip install
opencv_python-X.X-cp36-cp36
m-winX.whl
```

imread() function

- A function for reading a particular image. 특정 이미지를 읽는 함수
- Supports various image formats like PNG, JPEG, JPG, TIFF, etc. 다양한 포맷을 지원함

```
import cv2
image = cv2.imread('flower.jpg')
```

imshow() function

- A function for showing an image in a window. 윈도우안에 이미지를 보여주는 함수
- The window automatically fits to the image size. 자동으로 이미지사이즈를 조정함

```
cv2.imshow('flower',image);cv2.waitKey(0)
```

  - waitKey(0) - see a still image until you actually press any key 이미지를 윈도우를 클로즈할때까지 계속 보여줌
  - waitKey(1) - show a frame for 1 ms only. 1밀리세컨드동안 보여줌

imwrite() function

- A function for writing an image. 이미지를 저장하는 함수
- You can write the same image into the other format 같은 이미지를 다른 포맷으로 저장할수 있음

```
cv2.imwrite('flower.png',image)
```

Color Space Conversion

- OpenCV stored images in the reverse order i.e. in the BGR order. **오픈cv는 이미지를 반대의 순서로 저장**
- The cvtColor() function is for converting the image from one color code to other. **cvtColor()함수는 이미지를 한 칼라코드에서 다른 코드로 전환하는 함수**
- Convert the image to grayscale **이미지를 회색으로 바꿈**

```
image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
cv2.imshow('gray_flower',image);cv2.waitKey(0)
```

Edge Detection

- OpenCV provides very simple and useful function called Canny()for detecting the edges. **가장자리를 인식하기 위해 Canny함수를 사용**
- Uses the Canny () function for detecting the edges of the already read image. **Canny함수를 써서 이미 읽어놓은 이미지의 가장자리를 찾아냄**

```
cv2.imwrite('edges_flower.jpg',cv2.Canny(image,200,300))

cv2.imshow('edges', cv2.imread('edges_flower.jpg'))
```

Haar Cascade Classifier

- A machine learning object detection algorithm used to identify objects in an image or video 이미지나 비디오에서 사물을 식별하기 위한 기계학습 사물감지 알고리즘
- You can download it form https://github.com/opencv/opencv/tree/master/data/haarcascades

Face Detection

▪OpenCV has a built-in facility to perform face detection. 얼굴을 감지하기 위한 빌트인 분류기를 포함

```
face_detection=cv2.CascadeClassifier('haarcascade_frontalface_default.xml') # Haar cascade classifier사용
img = cv2.imread('human.jpg') #사진을 읽고
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #회색으로 바꾸고
faces = face_detection.detectMultiScale(gray, 1.3, 5) #실제얼굴 감지
For (x,y,w,h) in faces: #얼굴주위에 네모를 그림
   img = cv2.rectangle(img,(x,y),(x+w, y+h),(255,0,0),3)
cv2.imwrite('face_human.jpg',img) #이미지저장
cv2.imshow('face_human', cv2.imread('face_human.jpg'));cv2.waitKey(0)
```

Eye Detection

▪OpenCV has a built-in facility to perform eye detection. 눈을 감지하기 위한 빌트인 분류기를 포함

```
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml') #Haar cascade classifier사용
img = cv2.imread('human1.jpg') #이미지를 읽음
Gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #회색으로 바꿈
Eyes = eye_cascade.detectMultiScale(gray, 1.03, 5) #실제 눈감지
For (ex,ey,ew,eh) in eyes: #눈주위에 직사각형 그림
   img = cv2.rectangle(img,(ex,ey),(ex+ew, ey+eh),(0,255,0),2)
cv2.imwrite('eye_human.jpg',img) #파일에 저장
cv2.imshow('eye_human', cv2.imread('eye_human.jpg'));cv2.waitKey(0)  #다시 불러서 봄
```

Smoothing Images

- OpenCV provides a cv2.filter2D() function to convolve a kernel with an image. 이미지를 커널을 가지고 바꾸는 필터제공
- 5x5 averaging filter kernel

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

```
img = cv2.imread('flower.jpg')

kernel =
np.ones((5,5),np.float32)/25

dst = cv2.filter2D(img,-1,kernel)

plt.subplot(121),plt.imshow(img),p
lt.title('Original')

plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(dst),p
lt.title('Averaging')

plt.xticks([]), plt.yticks([])

plt.show()
```

Smoothing Images

- Averaging

```
blur = cv2.blur(img,(5,5))
```

- Gaussian Filtering

```
cv2.GaussianBlur(img,(5,5),0)
```

- Median Filtering

```
median = cv2.medianBlur(img,5)
```

- Bilateral Filtering

```
blur = cv2.bilateralFilter(img,9,75,75)
```

Image Transforms

- Fourier Transform is used to analyze the frequency characteristics of various filters. **푸리에변환은 빈도를 분석하기 위해 사용됨**
- Np.fft.fft2() provides the frequency transform and the zero frequency component (DC component) at top left corner bring to the center using np.fft.fftshift(). **빈도변환과 이동을 위해 fft2와 fftshift가 사용됨**

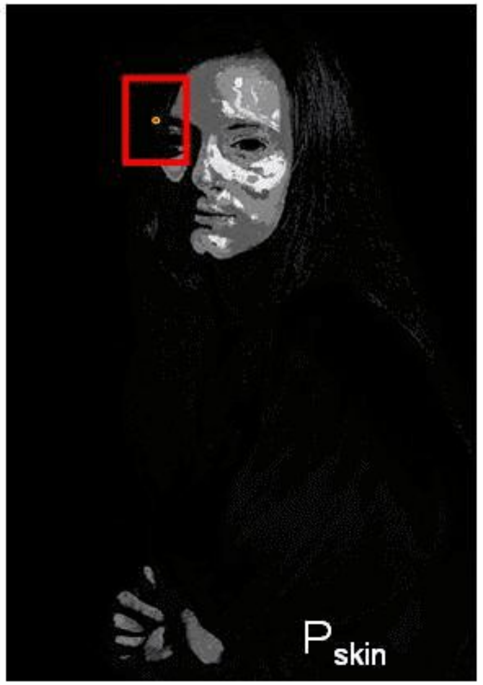
```
img = cv2.imread('flower.jpg',0)

f = np.fft.fft2(img)

fshift = np.fft.fftshift(f)

magnitude_spectrum =
20*np.log(np.abs(fshift))

plt.subplot(121),plt.imshow(img,
cmap = 'gray')

plt.title('Input Image'),
plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(magnitud
e_spectrum, cmap = 'gray')

plt.title('Magnitude Spectrum'),
plt.xticks([]), plt.yticks([])

plt.show()
```

Video Analysis

- Meanshift

데이터 집합의 밀도분포
(특징점,코너,색상)를
기반으로
Region Of Interest를
고속으로 추적하는
알고리즘

- Camshift



Mean shift window
initialization

Mean-shift 를
사용하되 탐색윈도우
크기를 스스로
조정할수 있음

k-Nearest Neighbour

```
trainData =
np.random.randint(0,100,(25,2)).astype(n
p.float32) #훈련데이터 만들기
responses =
np.random.randint(0,2,(25,1)).astype(np.
float32) #레드가 0, 블루가 1
red = trainData[responses.ravel()==0]
#레드
blue = trainData[responses.ravel()==1]
#블루
newcomer =
np.random.randint(0,100,(1,2)).astype(np
.float32) #새 입력값


plt.scatter(red[:,0],red[:,1],80,'r','^'
)
plt.scatter(blue[:,0],blue[:,1],80,'b','
s')
plt.scatter(newcomer[:,0],newcomer[:,1],
80,'g','o')
plt.show()
```

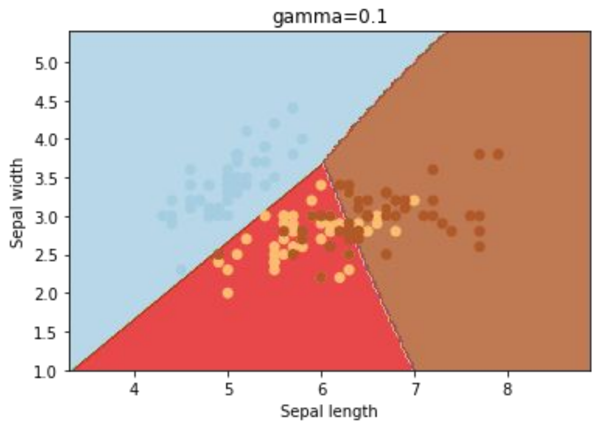
```
knn =
cv2.ml.KNearest_create()

knn.train(trainData,cv2.ml.R
OW_SAMPLE, responses)

ret, results, neighbours
,dist =
knn.findNearest(newcomer, 3)

print("result: ",
results,"\n")

print("neighbours: ",
neighbours,"\n")

print("distance: ", dist)
```

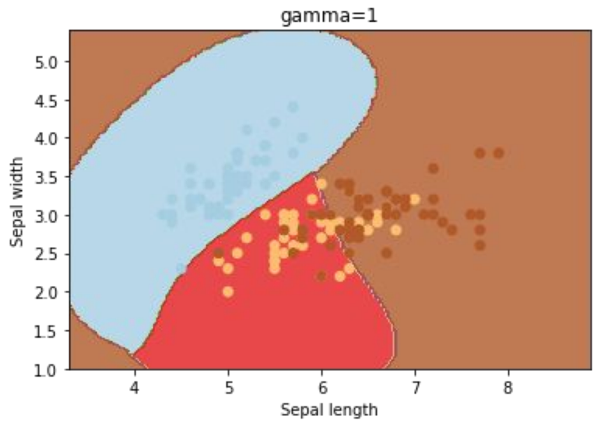
Support Vector Machine

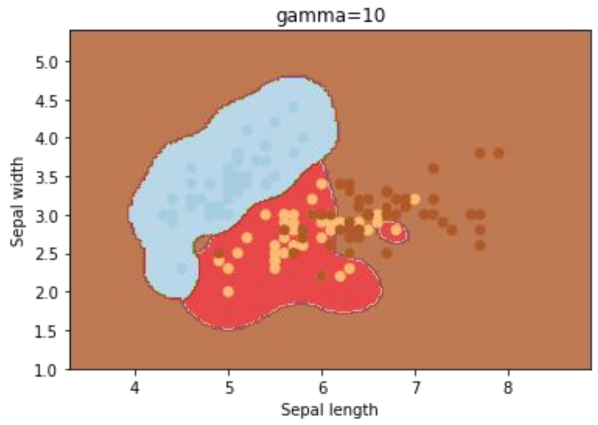
```
svm = cv2.ml.SVM_create()
svm.setGamma(1)
svm.setC(1)
svm.setKernel(cv2.ml.SVM_LINEAR)
svm.setType(cv2.ml.SVM_C_SVC)
svm.train(trainData, cv2.ml.ROW_SAMPLE, responses)
results, predicted = svm.predict(newcomers)
print("predicted: ", predicted)
```

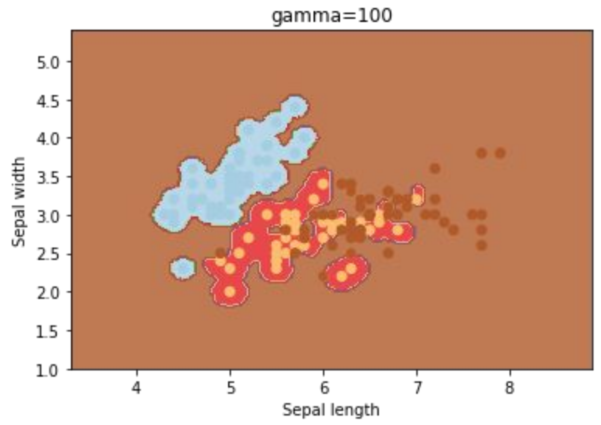
SVM gamma

- Gamma is a parameter for non-linear hyperplanes. The higher the gamma value it tries to exactly fit the training data set 비선형 하이퍼플레인에 대한 계수. 감마가 높을수록 훈련데이터에 더 잘 맞음



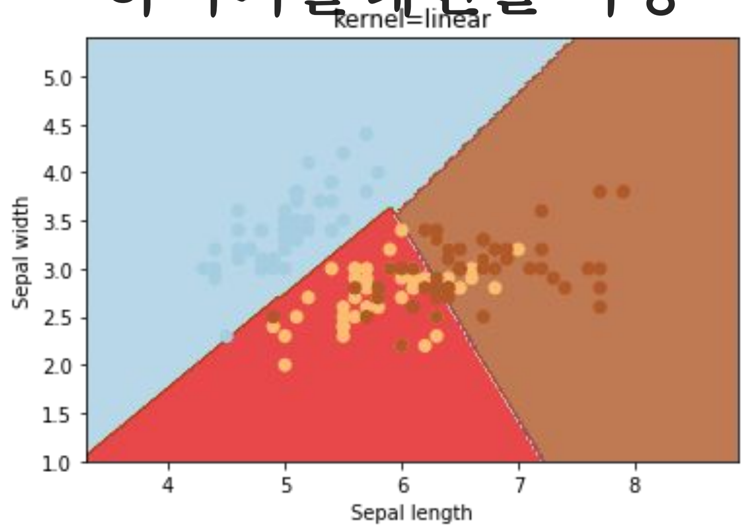


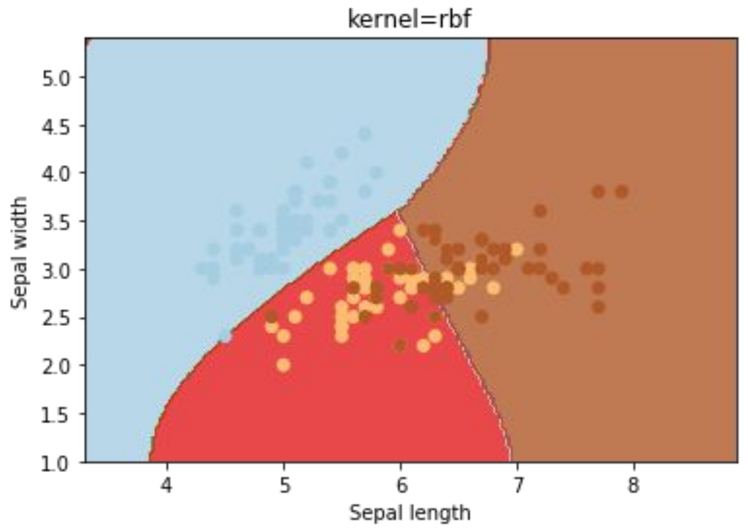


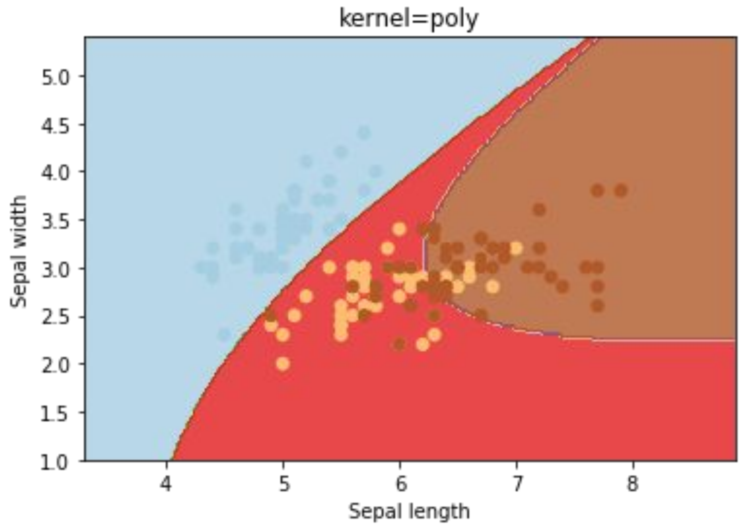


SVM Kernel

- Kernel parameters selects the type of hyperplane used to separate the data. Using ‘linear’ will use a linear hyperplane (a line in the case of 2D data). ‘rbf’ and ‘poly’ uses a non linear hyper-plane
  커널은 데이터를 가를때 사용하는 하이퍼플레인의 종류, rbf와 poly는 비선형 하이퍼플레인을 사용



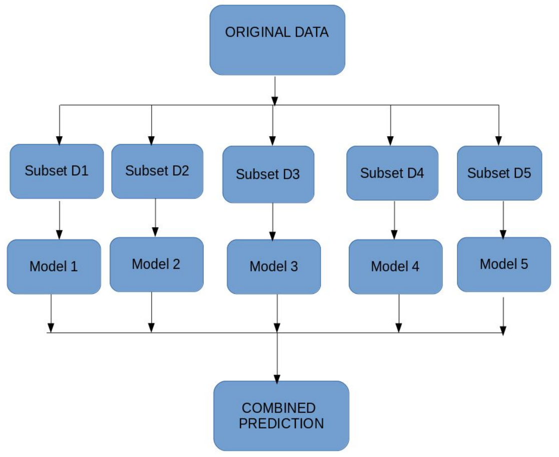




Bagging and Boosting

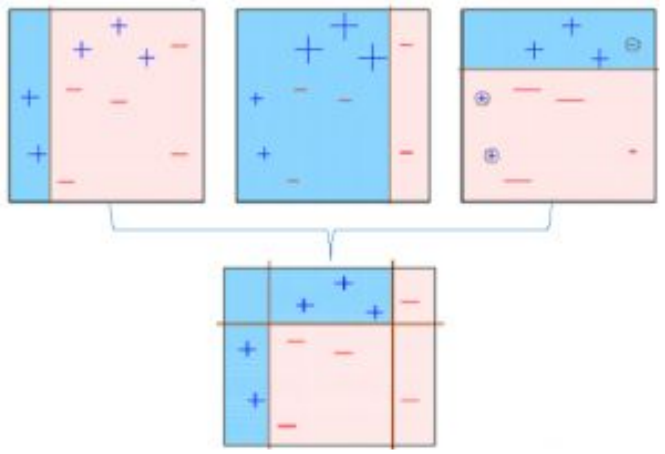
BAGGING

- A way to decrease the variance in the prediction by generating additional data for training from dataset using combinations with repetitions to produce multi-sets of the original data. 나눠서 모델만들고 합침



BOOSTING

- An iterative technique which adjusts the weight of an observation based on the last classification. 반복해서 모델만들고 결정



Boosting

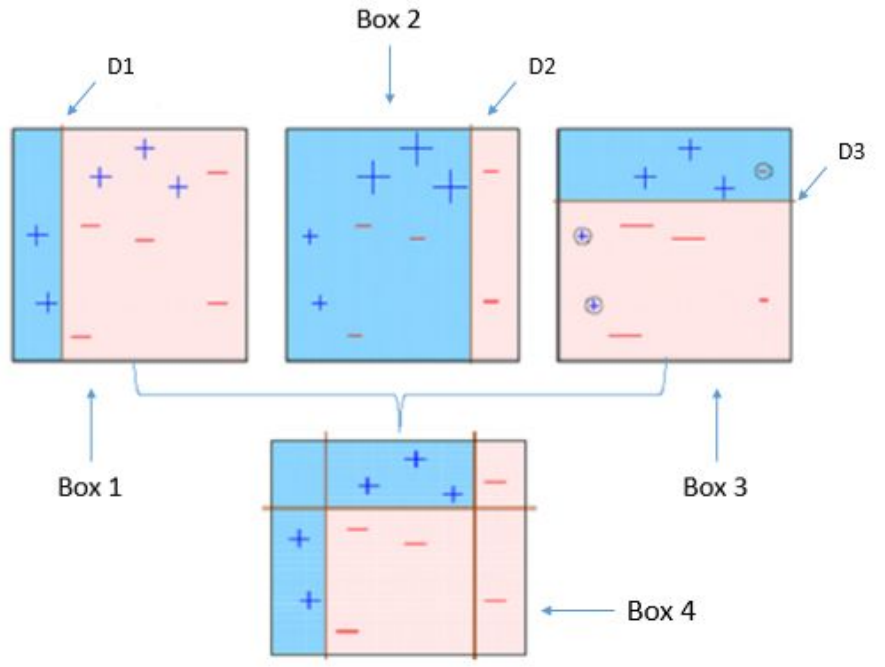
- A general ensemble method that creates a strong classifier from a number of weak classifiers. 약한 분류기들로부터 강한 분류기를 만드는 앙상블 기법
  - AdaBoost (Adaptive Boosting)
  - Gradient Tree Boosting
  - XGBoost

Boosting Algorithm

- Builds a model from the training data, then creating a second model that attempts to correct the errors from the first model. 훈련데이터로부터 모델을 하나 개발하고, 첫번째 모델의 오차를 수정하는 방향으로 두번째 모델을 개발
- Models are added until the training set is predicted perfectly or a maximum number of models are added. 훈련셋을 완벽히 예측할때까지 또는 최대모델수에 도달할때까지 모델을 추가함

AdaBoost (Adaptive Boosting)

- The first successful boosting algorithm developed for binary classification. 이진분류문제를 위해 개발된 가장 성공적인 첫번째 부스팅모델
- Used to boost the performance of decision trees 의사결정 트리의 성과를 높이기 위해 사용됨
- Can be used to boost the performance of any machine learning algorithm. 성과를 높이기 위해 어떤 기계학습 알고리즘과도 사용할수 있음



It fits a sequence of weak learners on different weighted training data.

AdaBoost Code

```
from sklearn.ensemble
import
AdaBoostClassifier

model =
AdaBoostClassifier(rando
m_state=1)

model.fit(x_train,
y_train)
model.score(x_test,y_tes
t)
```

```
from sklearn.ensemble
import AdaBoostRegressor

model =
AdaBoostRegressor()

model.fit(x_train,
y_train)

model.score(x_test,y_tes
t)
```

AdaBoost Parameters

- base_estimators: It helps to specify different ML algorithm. 알고리즘 종류
- n_estimators: It controls the number of weak learners. Default 10 몇번 반복하는지
- learning_rate: Controls the contribution of weak learners in the final combination. There is a trade-off between learning_rate and n_estimators. 마지막 결정때 웍러너의 기여도

Gradient Tree Boosting

- An approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. 새모델을 추가할때 경사하강알고리즘을 이용해서 로스를 최소화함

Gradient Tree Boosting

```
#For Classification

from sklearn.ensemble import GradientBoostingClassifier

clf = GradientBoostingClassifier(n_estimators=100,
learning_rate=1.0, max_depth=1) clf.fit(X_train, y_train)

#For Regression

from sklearn.ensemble import GradientBoostingRegressor

model= GradientBoostingRegressor()

model.fit(x_train, y_train)

model.score(x_test,y_test)
```

Gradient Boosting Parameters

- n_estimators: It controls the number of weak learners.
- learning_rate: Controls the contribution of weak learners in the final combination. There is a trade-off between learning_rate and n_estimators.
- max_depth: maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. Tune this parameter for best performance; the best value depends on the interaction of the input variables. **개별 회귀분석의 최대깊이, 모델을 향상시키기 위해 조정함**

XGBoost (eXtreme Gradient Boosting)

- An algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data 구조화, 테이블 데이터에 적합한 머신러닝 기법. 최근에 성과가 가장 좋았음
- An implementation of gradient boosted decision trees designed for speed and performance 경사하강법의 속도와 성과를 높임

```
pip install xgboost
```

Advantages of XGBoost

- Regularization
- Parallel Processing
- High Flexibility
- Handling Missing Values
- Tree Pruning
- Built-in Cross-Validation

XGBoost Code

```
import xgboost as xgb

model=xgb.XGBClassifier(
random_state=1,learning_
rate=0.01)

model.fit(x_train,
y_train)

model.score(x_test,y_tes
t)
```

```
import xgboost as xgb

model=xgb.XGBRegressor()

 model.fit(x_train,
y_train)

model.score(x_test,y_tes
t)
```

Summary

- CNN
- RNN
- Forecasting
- Text Mining
- Machine Learning