Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Course Project

Tutorial of EE4146

Presenter : Wuyang LI
E-mail: wuyangli2-c@my.cityu.edu.hk

Department of Electrical Engineering
City University of Hong Kong
03/11/2021

Outline

- Course Project
  1. Objective
  2. Kaggle Competition
  3. General Pipeline
- Basic Techniques
  1. Better Classifiers
  2. Better Features
  3. Offline Validation

Outline

- Course Project
  1. Objective
  2. Kaggle Competition
  3. General Pipeline
- Basic Techniques
  1. Better Classifiers
  2. Better Features
  3. Offline Validation

Objective

- Task: 6-class image classification


buildings


forest


glacier


mountain


sea

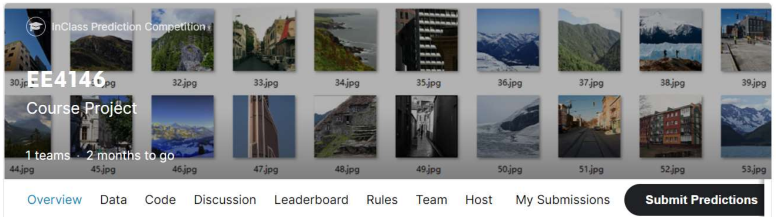

street

- Project objective:
  - Find and solve machine learning problems
  - Learn to read official documents and implement machine learning algorithms
  - Optimize your model with offline validation

Kaggle Competition

- How to participate:

  URL: *https://www.kaggle.com/t/5032b0f015b412f29411ddceb93b55c7*



EE 4146
COURSE PROJECT

- Data Information
  - Train/Test raw images
  - Off-the-shelled image features
  - Tarin set labels (csv file)

```
raw_images
    test
    train
res50_features
    test_feat
    train_feat
train_labels.csv
```

- Quick start
  - A baseline folder helping you quickly get start.

```
naive_baseline
    feature_extractor.ipynb
    naive_baseline.ipynb
```

- Deadline:
  - 8/12/2021

Kaggle Competition

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Result submission
  - Generate two-column results (.csv)
    Image ID (referred to image name) and Corresponding Predictions. (Arbitrary order is OK)

1	ID	label
2	0	street
3	1	forest
4	2	sea
5	3	forest
6	4	street
7	5	forest

- Evaluation matrix: classification accuracy

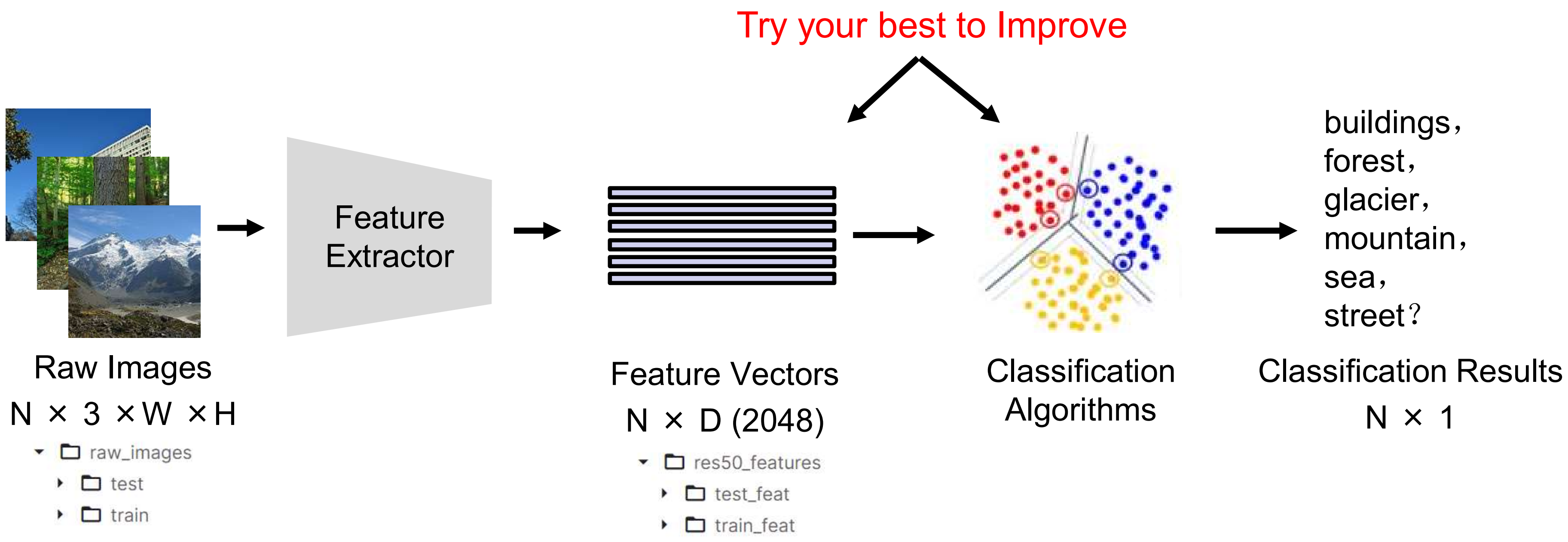
$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- Leaderboard rules:
  - 50% test data for the public leaderboard
  - The other 50% test data for the final rank

#	Team Name	Notebook	Team Members	Score	Entries	Last
1	naive-baseline			0.77636	1	2d

- Scoring
  - 5% performance, 5% report

General Pipeline



General Pipeline (Coding)

1. Load training labels and training/testing image features (pandas, numpy)

```python
# load training data labels
train_labels_info = pd.read_csv('train_labels.csv', header=0)
categories = list(np.unique(train_labels_info['label']))

# load train features
for feat in os.listdir(train_feat_root):
    img_id = int(feat.split('_')[0])
    train_feats.append(np.load(os.path.join(train_feat_root, feat)))
    label = train_labels_info['label'][img_id]
    train_labels.append(cat2num[label])

# load test features and corresponding ID
for feat in os.listdir(test_feat_root):
    img_id = int(feat.split('_')[0])
    test_img_id.append(img_id)
    test_feats.append(np.load(os.path.join(test_feat_root, feat)))
```

2. Improve your training features (Sklearn)

```python
# Improve your features using dimensionality reduction
pca = decomposition.KernelPCA(n_components=1024,kernel='rbf')
trainW = pca.fit_transform(trainX)  # fit the training set
valW  = pca.transform(valX)  # use the pca model to transform the test set
```

3. Adopt classification algorithms (Sklearn)

```python
# Training your classifiers
svmclf  =svm.SVC(kernel='linear')
svmclf.fit(trainW, trainY)
predY_svm = svmclf.predict(valW)
acc_svm = metrics.accuracy_score(valY, predY_svm)
print("Kernel PCA svm validation accuracy =", acc_svm)
```

4. Predict on test data and generate submission files (Sklearn)

```python
# Generate submission files
predY = svmclf.predict(test_feats)
write_csv_kaggle_sub('naive_baseline.csv',  test_img_id, predY)
```

Refer to “naive_baseline.ipynb”

- naive_baseline
  - feature_extractor.ipynb
  - naive_baseline.ipynb

You can also follow “featre_extractor.ipynb” to extract your own features

Outline

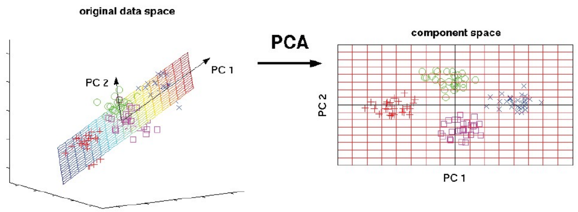
- Course Project
  1. Objective
  2. Kaggle Competition
  3. General Pipeline
- Basic Techniques
  1. Better Features
  2. Better Classifiers
  3. Offline Validation

Better Features: Dimensionality Reduction

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Linear dimensionality reduction (PCA, NMF, etc.)

  Project high-dim data on a low-dim linear surface (line, plane, etc.)




```
svmclf = svm.SVC(kernel='linear')
svmclf.fit(trainX, trainY)
predY_svm = svmclf.predict(valX)
acc_svm = metrics.accuracy_score(valY, predY_svm)
print("linear svm accuracy =", acc_svm)
✓ 15.5s
linear svm accuracy = 0.8928571428571429
```

Without PCA
89.2%
15.5s

- Advantages:
  - Reduce noise (outliers)
  - More abstracted (representative) features
- For classification
  - Accelerate classification
  - Improve accuracy (not always)

```
svmclf.fit(trainW, trainY)
predY_svm = svmclf.predict(testW)
acc_svm = metrics.accuracy_score(valY, predY_svm)
print("red svm test accuracy =", acc_svm)
✓ 6.8s
red svm test accuracy = 0.8938095238095238
```

2048D→ 512D
89.3% （+0.1%）
6.8s

sklearn.decomposition.PCA

class sklearn.decomposition.PCA(*n_components=None*, ***, *copy=True*, *whiten=False*, *svd_solver='auto'*, *tol=0.0*, *iterated_power='auto'*, *random_state=None*) [source]

Better Features: Dimensionality Reduction

Department of Electrical Engineering
香港城市大學 City University of Hong Kong

- Non-Linear dimensionality reduction (Kernel PCA)

  Project high-dim data on a low-dim non-flat surface
- How to do it?
  - Apply non-linear transformation on data

    $\mathbf{x}_i \Rightarrow \phi(\mathbf{x}_i)$
  - Project data on low-dim linear surface

    (i.e. run PCA on $\phi(\mathbf{x}_i)$)
- Advantages:
  - Reduce noise (outliers)
  - More abstracted (representative) features
  - Introduce more nonlinear representations
- For classification
  - Accelerate classification (not always)
  - Improve accuracy (not always)

```
svmclf = svm.SVC(kernel='linear')
svmclf.fit(trainX, trainY)
predY_svm = svmclf.predict(valX)
acc_svm = metrics.accuracy_score(valY, predY_svm)
print("linear svm accuracy =", acc_svm)
✓ 15.5s
linear svm accuracy = 0.8928571428571429
```

Without Kernel PCA
89.2%
15.5s

```
svmclf  =svm.SVC(kernel='linear')
svmclf.fit(trainW, trainY)
predY_svm = svmclf.predict(testW)
acc_svm = metrics.accuracy_score(valY, predY_svm)
print("lineear svm test accuracy =", acc_svm)
✓ 18.8s
lineear svm test accuracy = 0.9152380952380952
```

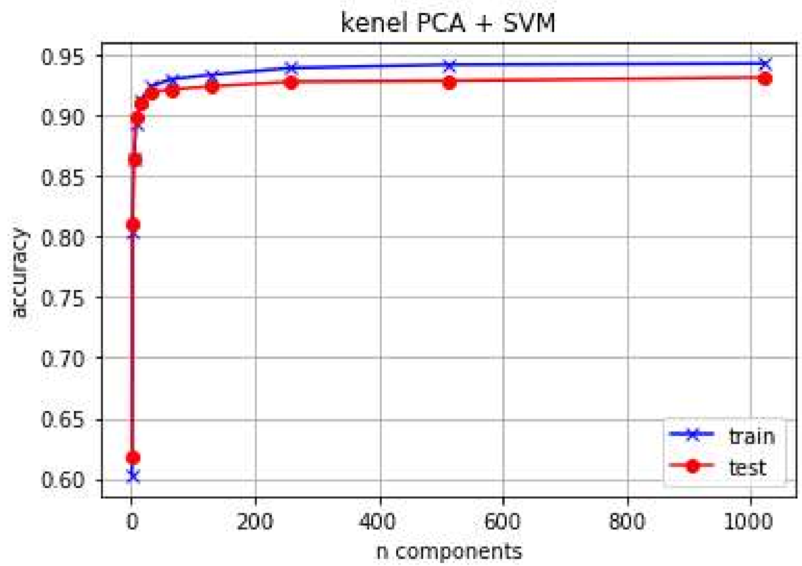
2048D→ 512D
91.5% （+2.3%）
18.8s

sklearn.decomposition.KernelPCA

class sklearn.decomposition.KernelPCA(*n_components=None*, ***, *kernel='linear'*, *gamma=None*, *degree=3*, *coef0=1*, *kernel_params=None*, *alpha=1.0*, *fit_inverse_transform=False*, *eigen_solver='auto'*, *tol=0*, *max_iter=None*, *iterated_power='auto'*, *remove_zero_eig=False*, *random_state=None*, *copy_X=True*, *n_jobs=None*) [source]

Better Features: Dimensionality Reduction

- Try to find the optimal number of reduced dimension
  e.g. Plotting the curve of accuracy vs. dimension



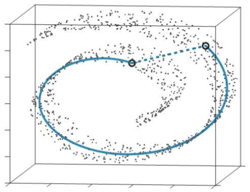
Better Features: Dimensionality Reduction (Extension)

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

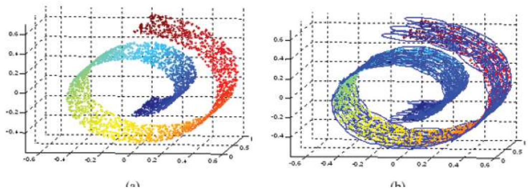
- Linear/ Non-Linear dimensionality reduction

  Project high-dim data on a low-dim Euclidean space:

  Pair-wise distance is defined as Euclidean distance, e.g. L1/L2/Cosine (dashed lines)



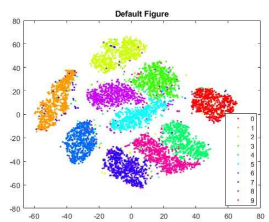
- Extension: manifold embedding (T-SNE reduction)



- Advantages:
  - Evaluation the quality of deep features: deep features are always embedded in manifold.

sklearn.manifold.TSNE

```
class sklearn.manifold.TSNE(n_components=2, *, perplexity=30.0, early_exaggeration=12.0, learning_rate='warn', n_iter=1000,
n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean', init='warn', verbose=0, random_state=None,
method='barnes_hut', angle=0.5, n_jobs=None, square_distances='legacy')
```
[source]



Better Features: Better Feature Extractors

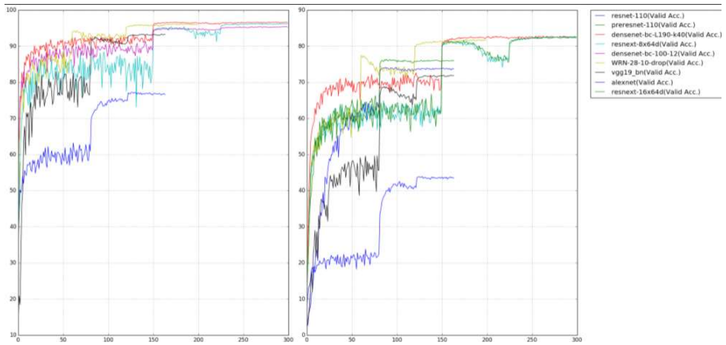
Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Deep feature extractors
  - ImageNet pre-trained deep feature extractors
    (VGG / ResNet / EfficientNet / DenseNet)
  - These models have been well-integrated in the torchvision library, and you can easily use them
  - The released features are extracted from ResNet 50

    - res50_features
      - test_feat
      - train_feat

  - More toys you can play with:
    https://pytorch.org/vision/stable/models.html

Model	Params (M)	CIFAR-10 (%)	CIFAR-100 (%)
alexnet	2.47	22.78	56.13
vgg19_bn	20.04	6.66	28.05
ResNet-110	1.70	6.11	28.86
PreResNet-110	1.70	4.94	23.65
WRN-28-10 (drop 0.3)	36.48	3.79	18.14
ResNeXt-29, 8x64	34.43	3.69	17.38
ResNeXt-29, 16x64	68.16	3.53	17.30
DenseNet-BC (L=100, k=12)	0.77	4.54	22.88
DenseNet-BC (L=190, k=40)	25.62	3.32	17.17

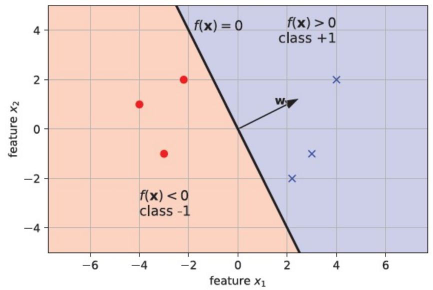


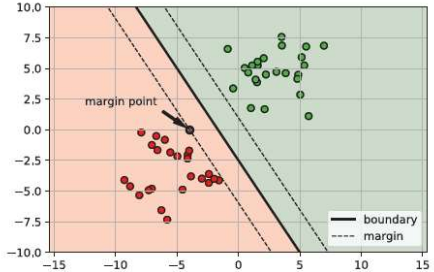
Better Classifiers: Linear Classifier

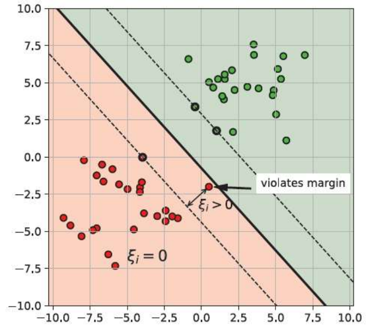
- Linear classifier (SVM/ Logistic Regression):
  Train a linear surface to distinguish two categories
- Support Vector Machines (SVM)
  Maximum margin of “margin points” (support vector)
- How to do it?
  - Define the most neared margin (margin points)

$$d_i = \frac{|f(\mathbf{x}_i)|}{||\mathbf{w}||} \qquad \gamma = \min_i \frac{|f(\mathbf{x}_i)|}{||\mathbf{w}||}$$

  - Maximum the margin
- Soft-SVM:
  - Allow some samples to violate the margin
  - Controlled by C







sklearn.svm.SVC

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', 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', break_ties=False,
random_state=None)
```
[source]

Better Classifiers: Linear Classifier

- Support Vector Machines (SVM)

  Maximum Margin of “margin points” (support vector)
- Advantages
  - Works well on high-dimensional data
  - Fast
- Disadvantages
  - Decision boundary can only be linear

```
svmclf = svm.SVC(kernel='linear')
svmclf.fit(trainX, trainY)
predY_svm = svmclf.predict(valX)
acc_svm = metrics.accuracy_score(valY, predY_svm)
print("linear svm validation accuracy =", acc_svm)
✓ 12.5s
linear svm validation accuracy = 0.9135238095238095
```

Linear SVM: 91.3%

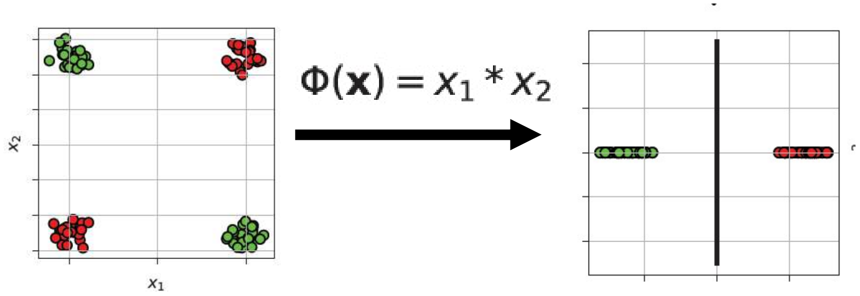
Better Classifiers: Non-Linear Classifier

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Kernel-based classifier

  Similar to Kernel PCA: transform inputs with non-linear kernel function

- A toy example:



- How to do it?
  - Transform input features with kernel function

    Different kernel functions: RBF, Poly-nomal, Sigmoid ,etc.
  - Train a linear classifier in the transformed space.

sklearn.svm.SVC

class sklearn.svm.SVC(**, C=1.0, kernel='rbf', degree=3, gamma='scale', 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', break_ties=False, random_state=None*) [source]

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'*

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

Better Classifiers: Non-Linear Classifier

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Kernel-based classifier

  Similar to Kernel PCA: transform inputs with non-linear kernel function
- Advantages
  - Non-linear decision boundary
  - Kernel function can be used in various data format instead of vector-data
- Disadvantages
  - Sensitive to the used kernel function
  - Computationally expensive on large-scale data (large kernel matrix)

```
svmclf = svm.SVC(kernel='linear')
svmclf.fit(trainX, trainY)
predY_svm = svmclf.predict(valX)
acc_svm = metrics.accuracy_score(valY, predY_svm)
print("linear svm validation accuracy =", acc_svm)
✓ 12.5s
linear svm validation accuracy = 0.9135238095238095
```

Linear SVM: 91.3%

```
svmclf = svm.SVC(kernel='rbf')
svmclf.fit(trainX, trainY)
predY_svm = svmclf.predict(valX)
acc_svm = metrics.accuracy_score(valY, predY_svm)
print("rbf svm validation accuracy =", acc_svm)
✓ 24.6s
rbf svm validation accuracy = 0.932952380952381
```

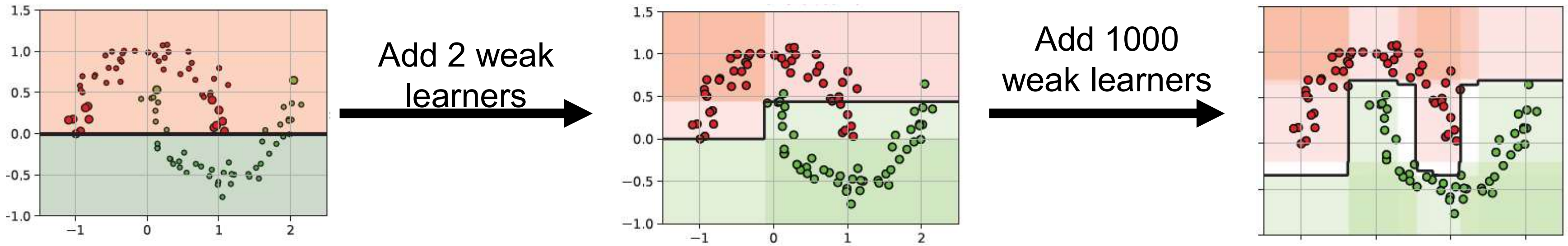
RBF kernel SVM: 93.3% （+2%）

Better Classifiers: Non-Linear Classifier

- Ensemble method: Boosting

  Training multiple classifiers, each focusing on the errors made by the previous classifiers (larger weight)

- A toy example:



- How to do it?

  Weighted data sample & weighted classifier

  - Choose and train a weak learner with minimized classification error using weighted data
  - Set the classifier weight of weak learners according to classification error
  - Add to ensemble according to classifier weight : *Classifier* = *A1 classifier* + *A2 classifier* +…
  - Update data weight for each sample (To focus on those misclassified data)

Better Classifiers: Non-Linear Classifier

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Ensemble method: Boosting

  Training multiple classifiers, each focusing on the errors made by the previous classifiers
- Advantages
  - Good generalization
  - Build-in feature selection: which feature is more important
- Disadvantages
  - Can be sensitive to outliers

sklearn.ensemble.AdaBoostClassifier

class sklearn.ensemble.AdaBoostClassifier(*base_estimator=None, *, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None*) [source]

sklearn.ensemble.GradientBoostingClassifier¶

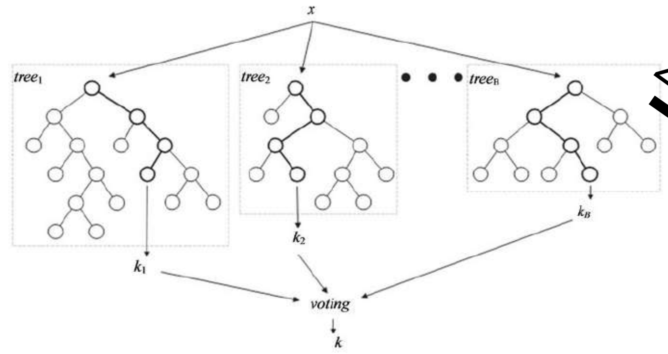
class sklearn.ensemble.GradientBoostingClassifier(*, *loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, init=None, random_state=None, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0*) [source]

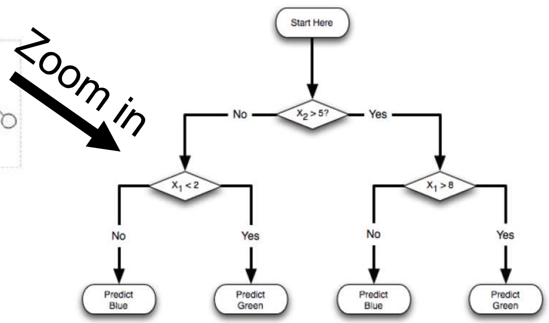
Better Classifiers: Non-Linear Classifier

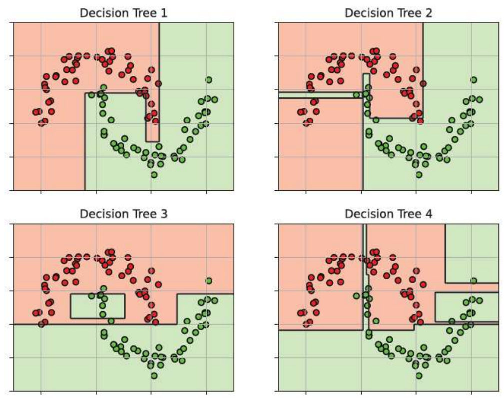
- Bagging method: Random Forest

  Training multiple classifiers from randomly selected training data

- A toy example:







- How to do it?
  - Train: For each tree (classifier):
    - Create a new subset of data by randomly sampling from all training data
    - Train this tree using sampled subset features
  - Test: For each test sample :
    - Aggregating all predictions given by trees (voting)

Better Classifiers: Non-Linear Classifier

- Ensemble method: Random Forest

  Training multiple classifiers from randomly selected training data
- Advantages
  - Non-linear decision boundary
  - Good generalization
  - Fast in training (in parallel)
- Disadvantages
  - Can be sensitive to outliers
  - Tree-based classifier : hard to represent “diagonal” decision boundary

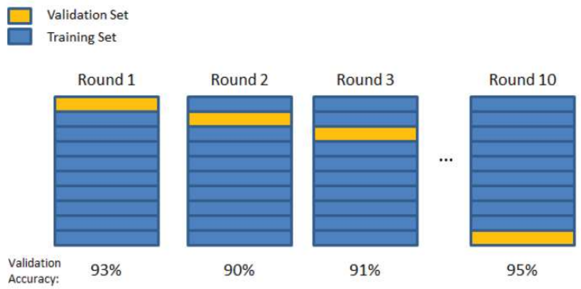
sklearn.ensemble.RandomForestClassifier¶

class sklearn.ensemble.RandomForestClassifier(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

Offline Validation

- Cross-validation

  Run experiments on the training set several times to obtain the optimal setting
  - Split training data into batches of training and validation set
  - Try different settings on each splitting
  - Pick the best one working well on all splitting



- Integrated Validation

  Sklearn library has integrated grid-search to perform validation automatically and obtain the best parameters

sklearn.model_selection.GridSearchCV

class sklearn.model_selection.GridSearchCV(*estimator, param_grid, *, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False*) [source]

Summary

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Better features
  - Use some tricks, e.g. dimensionality reduction
  - Use stronger feature extractors (VGG/ ResNet/ DenseNet…)
- Better Classifiers
  - Linear Classifiers (SVM, Logistic Regression)
  - Non-linear Classifier:
    - Kernel-based methods
    - Boosting: Ada boosting/ Gradient boosting…
    - Bagging: Random forest…

Necessary Libraries

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Python: https://www.python.org/
  Basic coding language
- Numpy https://numpy.org/
  Comprehensive matrix calculation library
- Sklearn: https://scikit-learn.org/stable/
  Off-the-shelled machine algorithms
- Matplotlib https://matplotlib.org/
  Do visualization, e.g. plotting figures
- Pytorch*(selective) https://pytorch.org/
  Deep feature extractors

Practice more!
Have fun!