EE 569 Discussion 14

Yao Zhu

04/23/2021

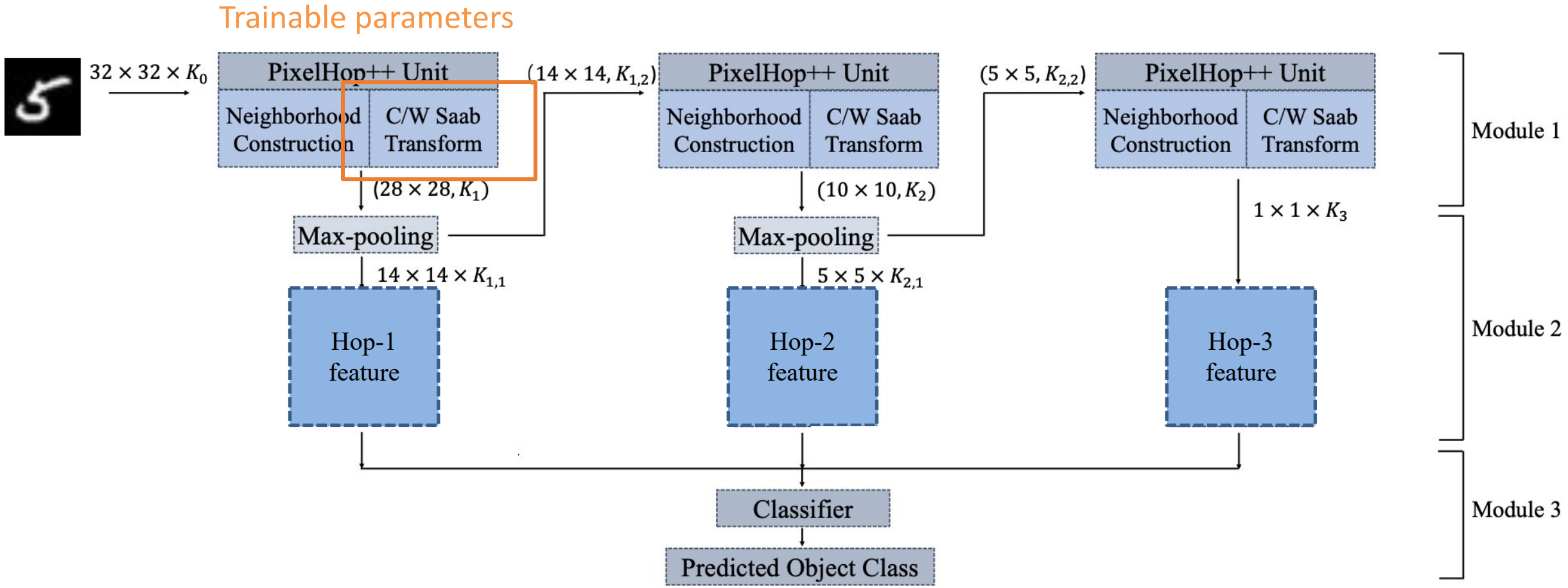
Contents

- Homework 6: Successive Subspace Learning
  - How to calculate the model size?
  - Error Analysis
  - Memory
  - About XGBoost

Contents

- Homework 6: Successive Subspace Learning
  - How to calculate the model size for PixelHop++?
  - Error Analysis
  - Memory
  - About XGBoost

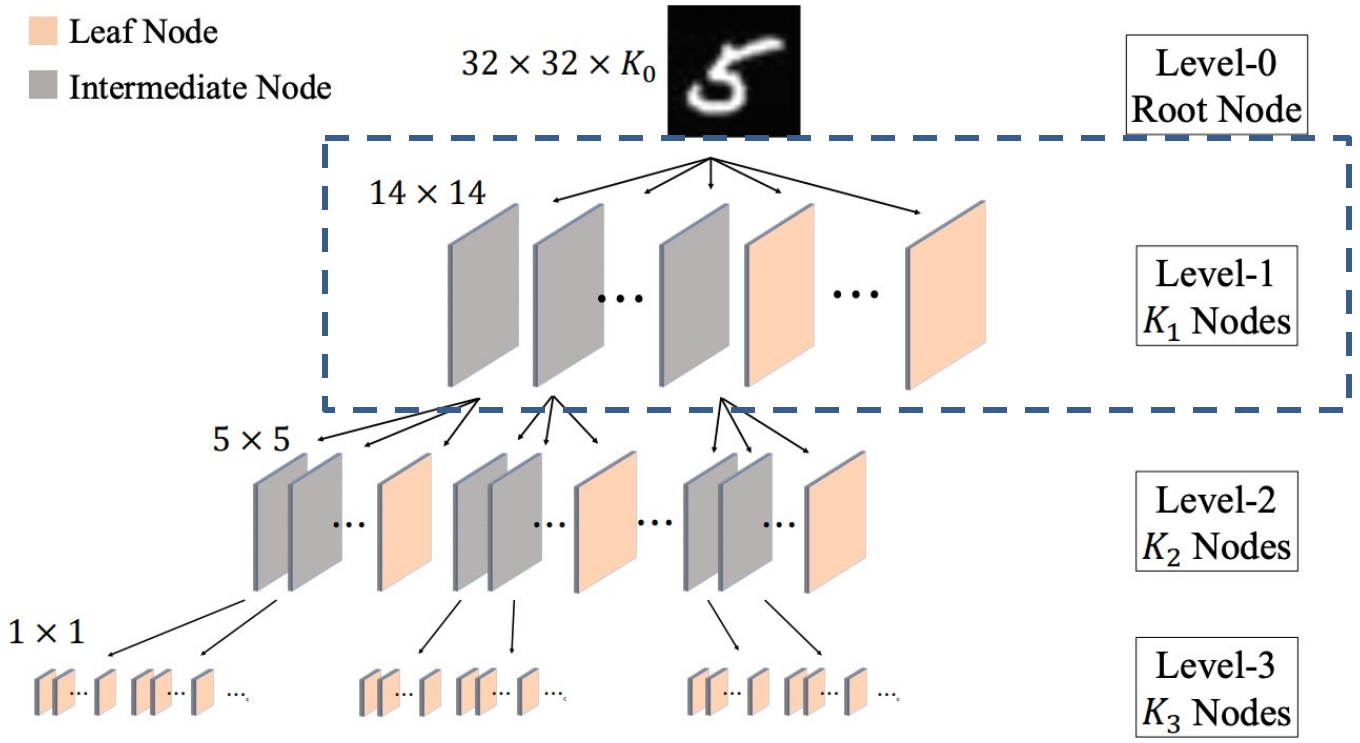
Model Size


Trainable parameters
$32 \times 32 \times K_0$
PixelHop++ Unit
Neighborhood Construction
C/W Saab Transform
$(14 \times 14, K_{1,2})$
PixelHop++ Unit
Neighborhood Construction
C/W Saab Transform
$(5 \times 5, K_{2,2})$
PixelHop++ Unit
Neighborhood Construction
C/W Saab Transform
$(28 \times 28, K_1)$
$(10 \times 10, K_2)$
$1 \times 1 \times K_3$
Max-pooling
Max-pooling
$14 \times 14 \times K_{1,1}$
$5 \times 5 \times K_{2,1}$
Hop-1 feature
Hop-2 feature
Hop-3 feature
Classifier
Predicted Object Class
Module 1
Module 2
Module 3

Model Size

PixelHop++ Units (parameters of CWSaab filters)

[depends on spatial neighborhood size, and energy thresholds TH1 & TH2]



e.g.: For **Hop1 (Level 1 in Figure)**

If:

- Result in K1 nodes after discarding some nodes based on TH2

•Used 5x5 in spatial

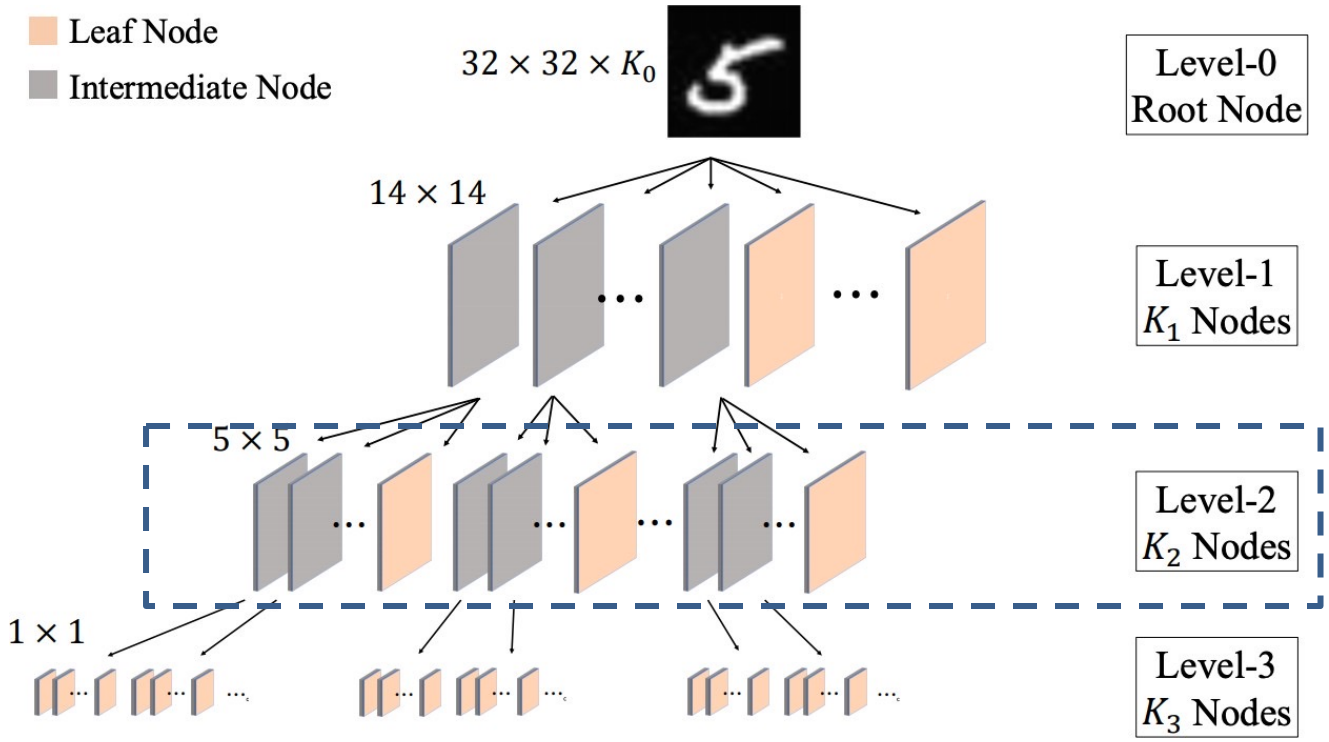
Then:

Hop 1 # of trainable parameters = 5*5*K1

Model Size

PixelHop++ Units (parameters of CWSaab filters)

[depends on spatial neighborhood size, and energy thresholds TH1 & TH2]



e.g.: For **Hop2 (Level 2 in Figure)**

If:

- $K_{1,2}$ nodes pass to Hop2, result in totally K2 nodes after discarding some nodes based on TH2

•Used 5x5 in spatial

Then:

Hop 2 # of trainable parameters = 5*5*K2

Model Size

How to get K1, K2, K3 from code?

- Method 1: from get_feat function

```
# get Hop1 feature
get_feat(x_train, num_layers=0) # or get_feat(x_train, num_layers=1)
```

result shape: (Num_imgs, 28, 28, K1)

```
# get Hop2 feature
get_feat(x_train, num_layers=2)
```

result shape: (Num_imgs, 10, 10, K2)

- Method 2: from trained PixelHop model

```
Hop1_energy = p2.Energy['Layer0']   # list of length K1
Hop2_energy = p2.Energy['Layer1']   # list of list
Hop3_energy = p2.Energy['Layer2']   # list of list
```

Outer list: length K1,2
Inner list: each channel's K2
K2 = sum of all inner list's length

Similarly…

Outer list: length K2,2
Inner list: each channel's K2
K3 = sum of all inner list's length

Model size

PixelHop Units (parameters of Saab filters)

[depends on spatial neighborhood size, and energy thresholds TH2 only]

- Method 1: same as PixelHop++

- Method 2:

```
Hop1_energy = p2.Energy['Layer0']   # list of length K1
Hop2_energy = p2.Energy['Layer1']   # list of length K2
Hop3_energy = p2.Energy['Layer2']   # list of length K3
```

Contents

- Homework 6: Successive Subspace Learning
  - How to calculate the model size?
  - Error Analysis
  - Memory
  - About XGBoost

P2(c) Error Analysis

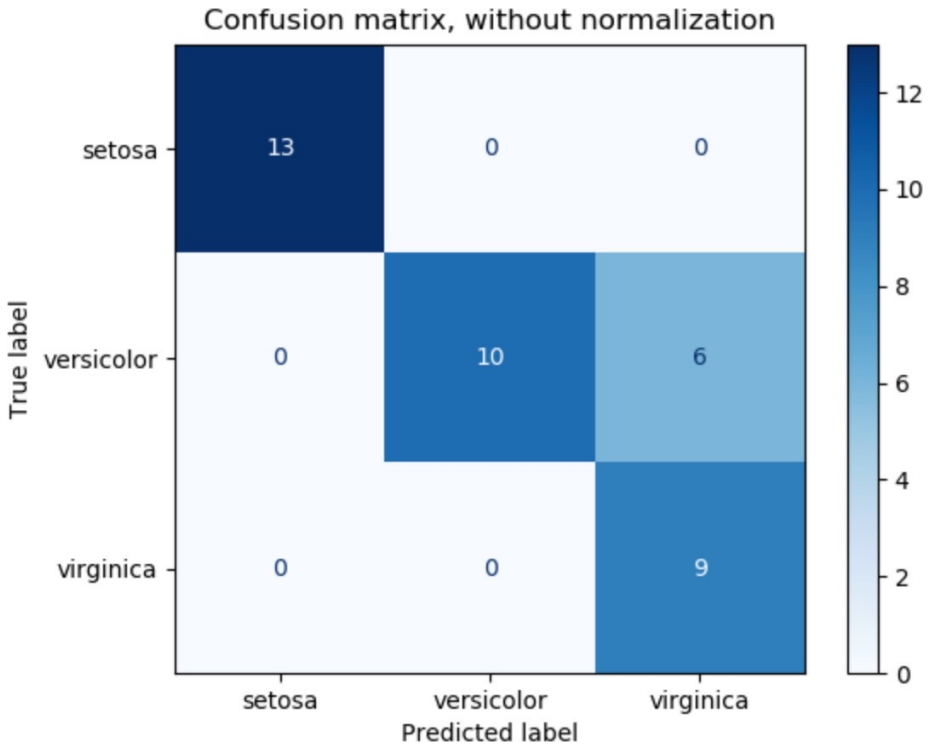
- It shows how your classification model is confused among different classes

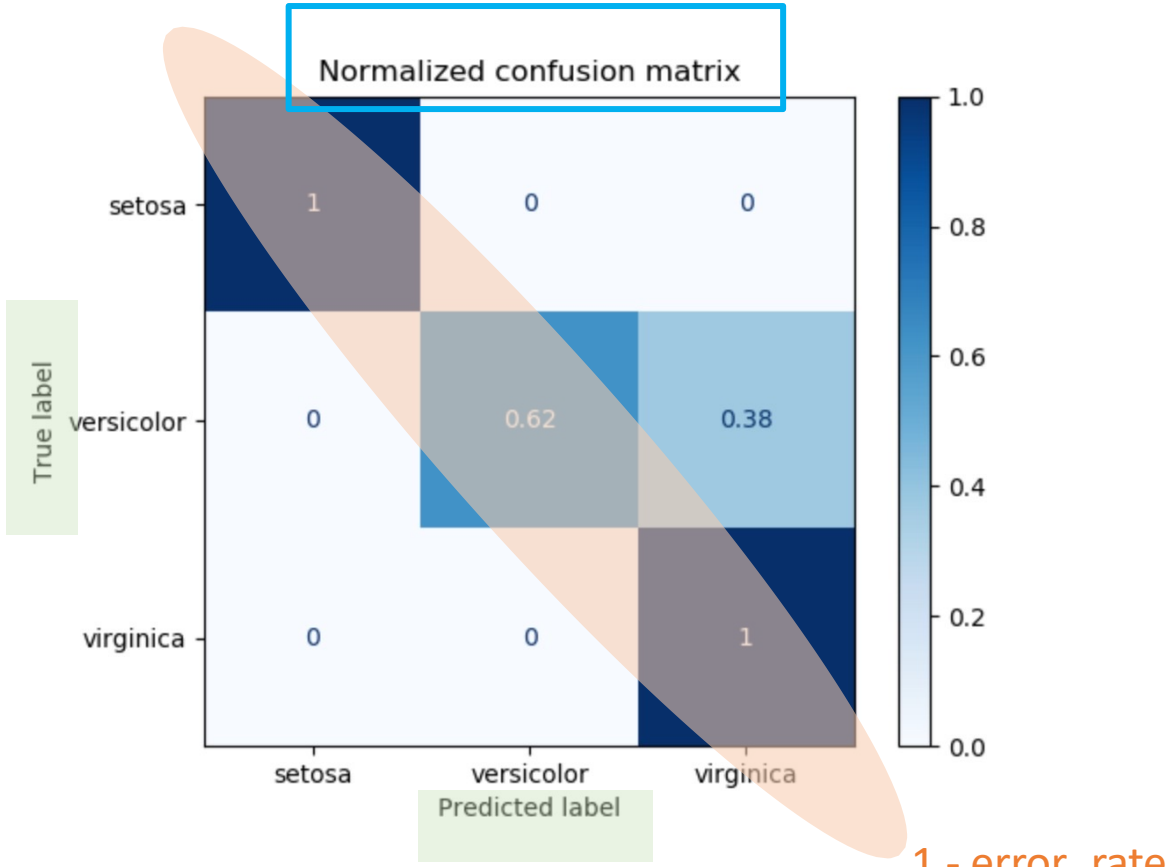
  e.g. Binary class:

		Predicted Class: Spam	Predicted Class: Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

P2(c) Error Analysis

- It shows how your classification model is confused among different classes





1 - error_rate

P2(c) Error Analysis

- Some useful tools in Python for calculating and plotting confusion matrix

sklearn.metrics.confusion_matrix:

```
from sklearn.metrics import confusion_matrix
```

plot_confusion_matrix:

requirement: scikit-learn 0.22 or later

P2(c) Error Analysis

- Find confusing groups based on the confusion matrix

An example of PixelHop on FashionMNIST:

	T-shirt/top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
T-shirt/top	0.883	0.000	0.015	0.016	0.005	0.000	0.072	0.000	0.009	0.000
Trouser	0.001	0.980	0.000	0.013	0.002	0.000	0.002	0.000	0.002	0.000
Pullover	0.015	0.001	0.877	0.009	0.053	0.000	0.044	0.000	0.001	0.000
Dress	0.017	0.006	0.010	0.919	0.023	0.000	0.022	0.000	0.003	0.000
Coat	0.000	0.001	0.056	0.027	0.866	0.000	0.050	0.000	0.000	0.000
Sandal	0.000	0.000	0.000	0.000	0.000	0.979	0.000	0.016	0.000	0.005
Shirt	0.110	0.000	0.048	0.021	0.072	0.000	0.742	0.000	0.007	0.000
Sneaker	0.000	0.000	0.000	0.000	0.000	0.010	0.000	0.971	0.000	0.019
Bag	0.003	0.001	0.003	0.002	0.002	0.001	0.001	0.004	0.983	0.000
Ankle boot	0.000	0.000	0.000	0.000	0.000	0.005	0.001	0.026	0.000	0.968

P2(c) Error Analysis

- Find confusing groups based on the confusion matrix

  An example of PixelHop on FashionMNIST:

Visually quite similar

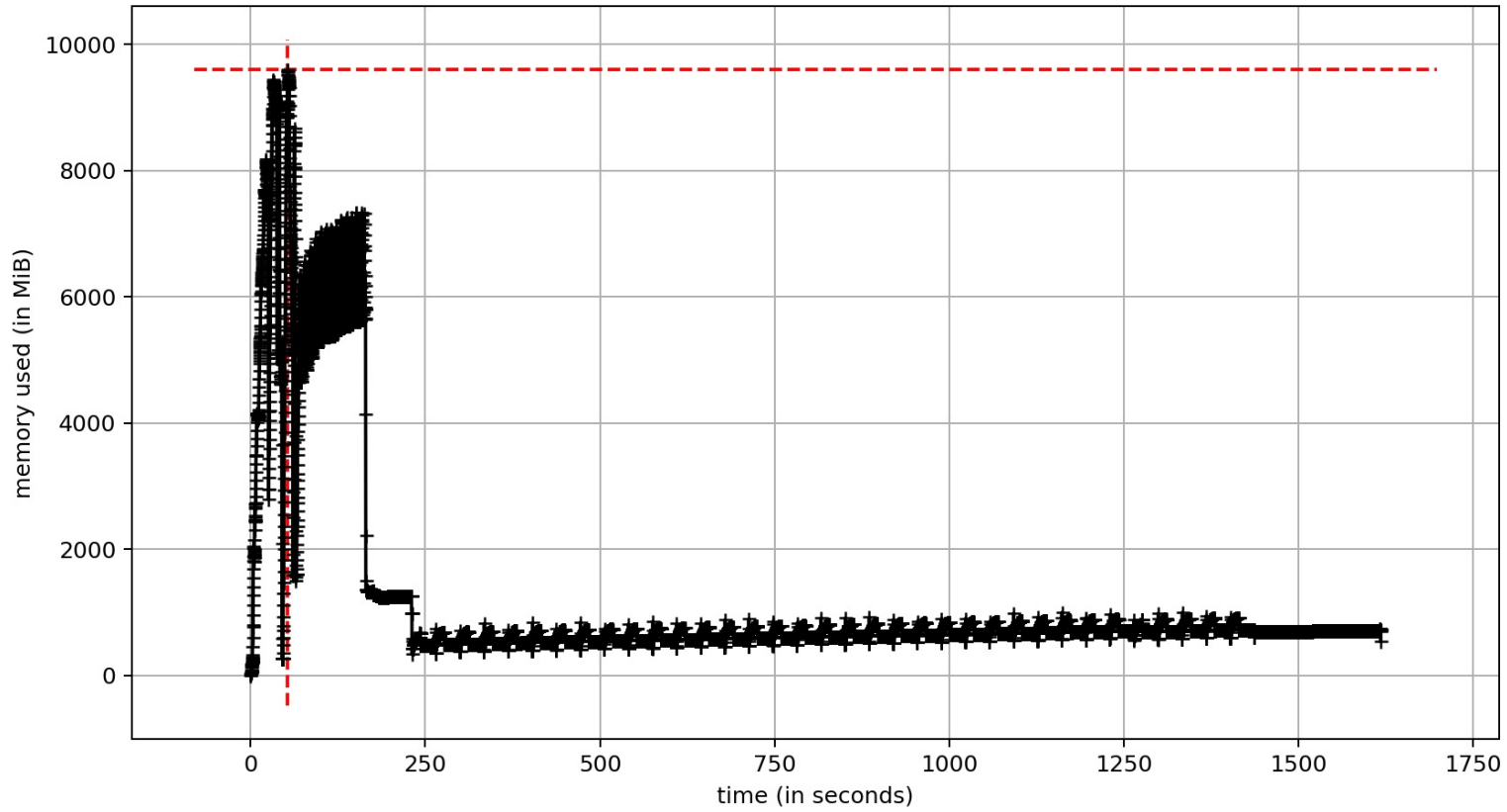
Contents

- Homework 6: Successive Subspace Learning
  - How to calculate the model size for PixelHop++?
  - Error Analysis
  - Memory
  - About XGBoost

Memory – PixelHop++

PixelHop++ architecture:

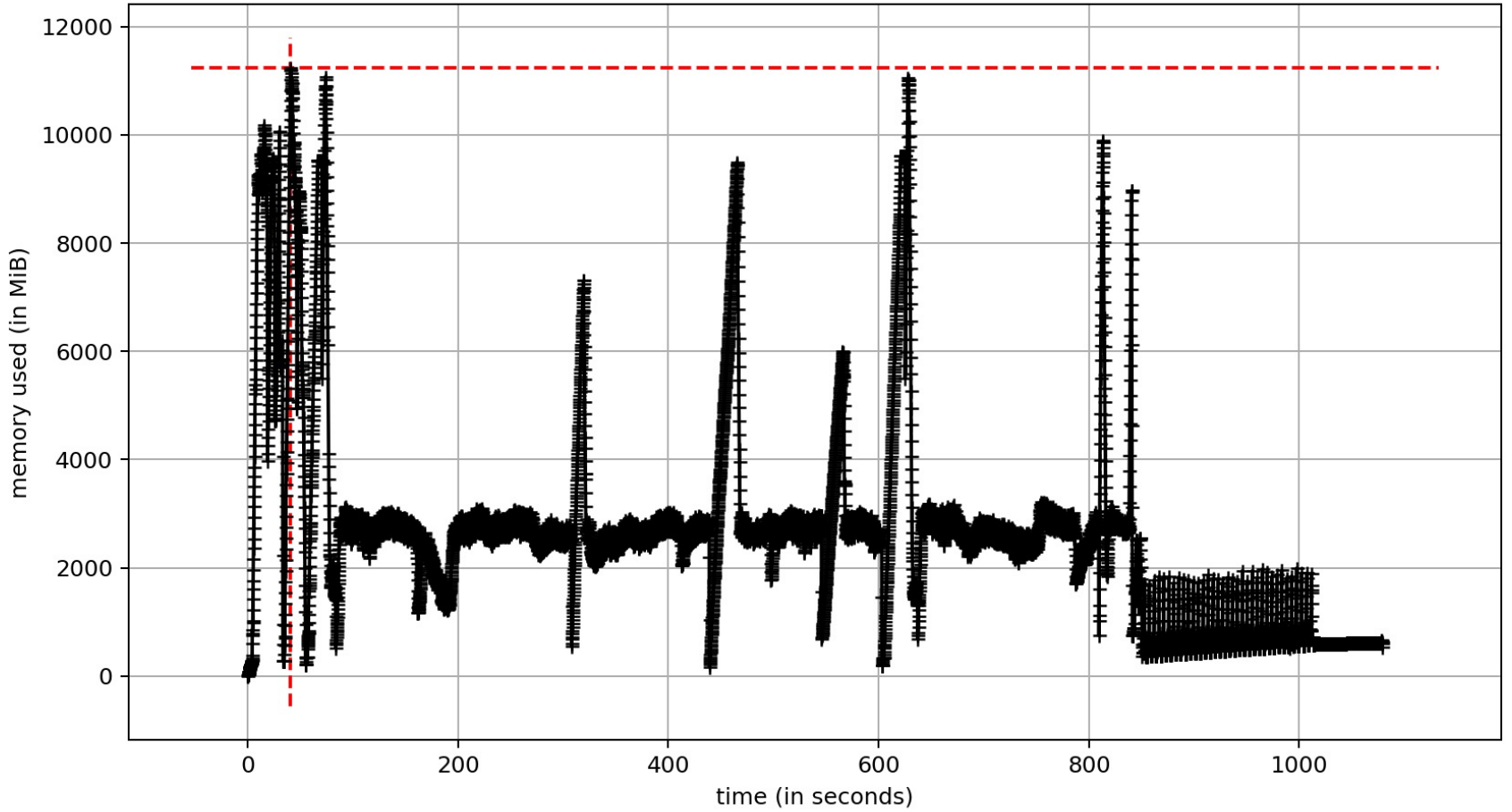
- 60000 images train Module 1: peak memory 9.5GB



Memory – PixelHop

PixelHop architecture:

- 60000 images train Module 1: peak memory 11GB



Contents

- Homework 6: Successive Subspace Learning
  - How to calculate the model size for PixelHop++?
  - Error Analysis
  - Memory
  - About XGBoost

About XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework.

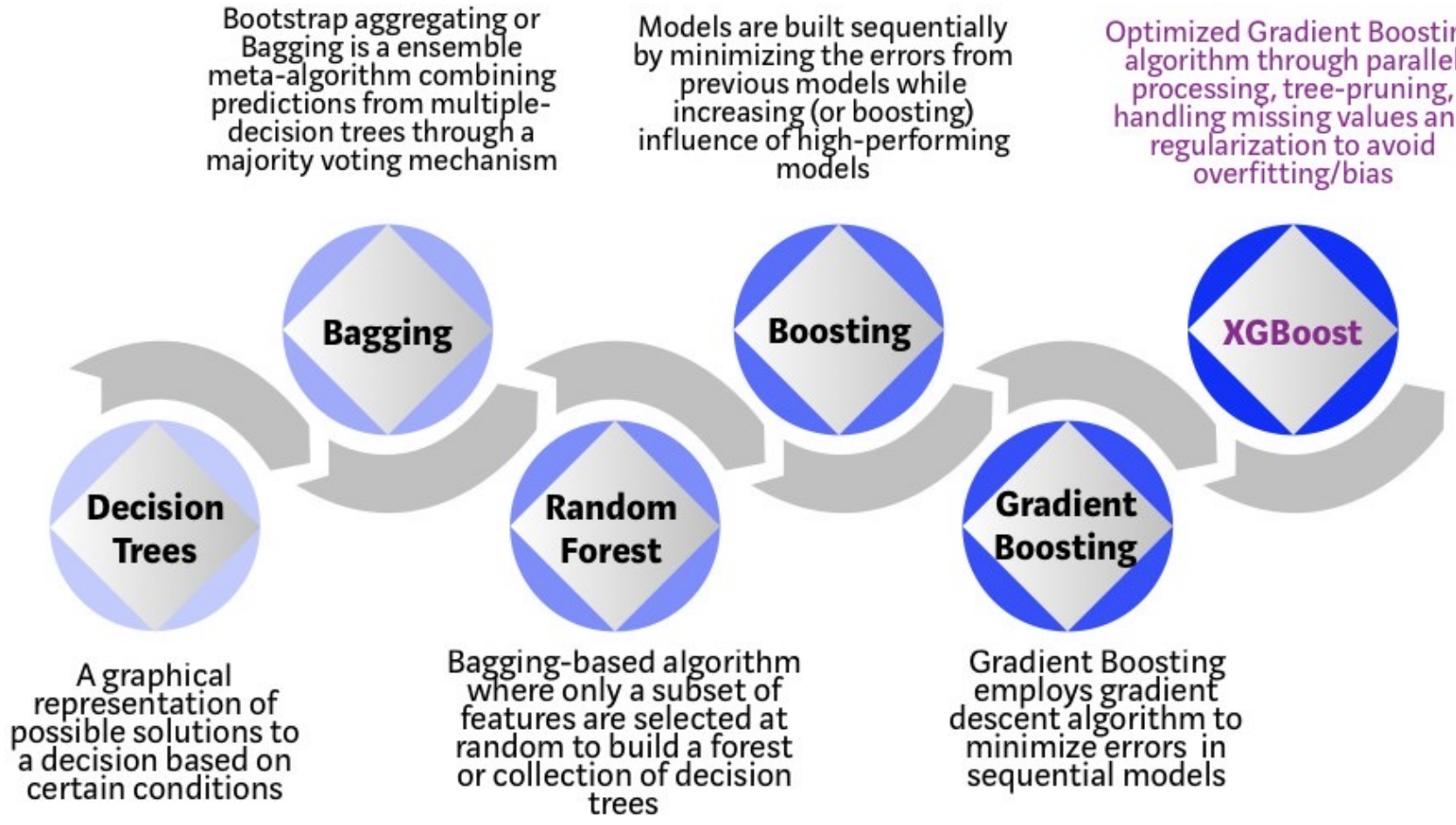


Figure credit to: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

About XGBoost

1. A wide range of applications: regression, classification, ranking, and user-defined prediction problems.
2. Portability: Runs smoothly on Windows, Linux, and OS X.
3. Languages: Supports all major programming languages including C++, Python, R, Java, Scala, and Julia.
4. Cloud Integration: Supports AWS, Azure, and Yarn clusters and works well with Flink, Spark, and other ecosystems.

In HW6, you are only required to use XGBoost classifier from Python, sklearn API

About XGBoost

Hyperparameters matters!
Randomized search from sklearn is a good tool to find good parameters for XGBoost

```
from xgboost import XGBClassifier
folds = 4
param_comb = 10
params = {
'min_child_weight': [1, 2, 3, 5, 7, 11, 13, 17, 19, 23],
'gamma': [0.5, 1, 1.5, 2, 5],
'subsample': [0.6, 0.8, 1],
'colsample_bytree': [0.6, 0.8, 1],
'learning_rate': [0.01, 0.1, 0.2],
'max_depth': [6, 8, 10],
}
xgb = XGBClassifier(n_estimators=100, eval_metric='auc',
scale_pos_weight=(len(train_labels[train_labels==0])/len(train_labels[train_labels==1])))
skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 0)
clf = RandomizedSearchCV(xgb, param_distributions=params, n_iter=param_comb,
scoring='roc_auc', n_jobs=8, cv=skf.split(train_regions_prob,train_labels),
random_state=1001)
clf.fit(train_regions_prob, train_labels)
clf = clf.best_estimator_
```


Q
A