Network Sparsification, Pruning, Quantization

MLAI Lab.

2020. 07. 23

많은 것을 배웠지만?!

어떻게 만들어야 되는지 전혀 감이 잡히지 않는다 !

실습 목표

가중치 가지치기 (Weight Pruning)의 원리와 코드를 ***쉽게*** 이해하자 !

강의 내용 중에서..

Making Deep Neural Networks More Efficient

Reducing the number of parameters of given networks (weights / neurons)

- **Static:** Pruning, sparsity-inducing regularizations and priors (Bayesian)
- **Dynamic:** Partial network evaluation

Reducing the memory and computation requirement per weight.

- Quantization (Bit compression)

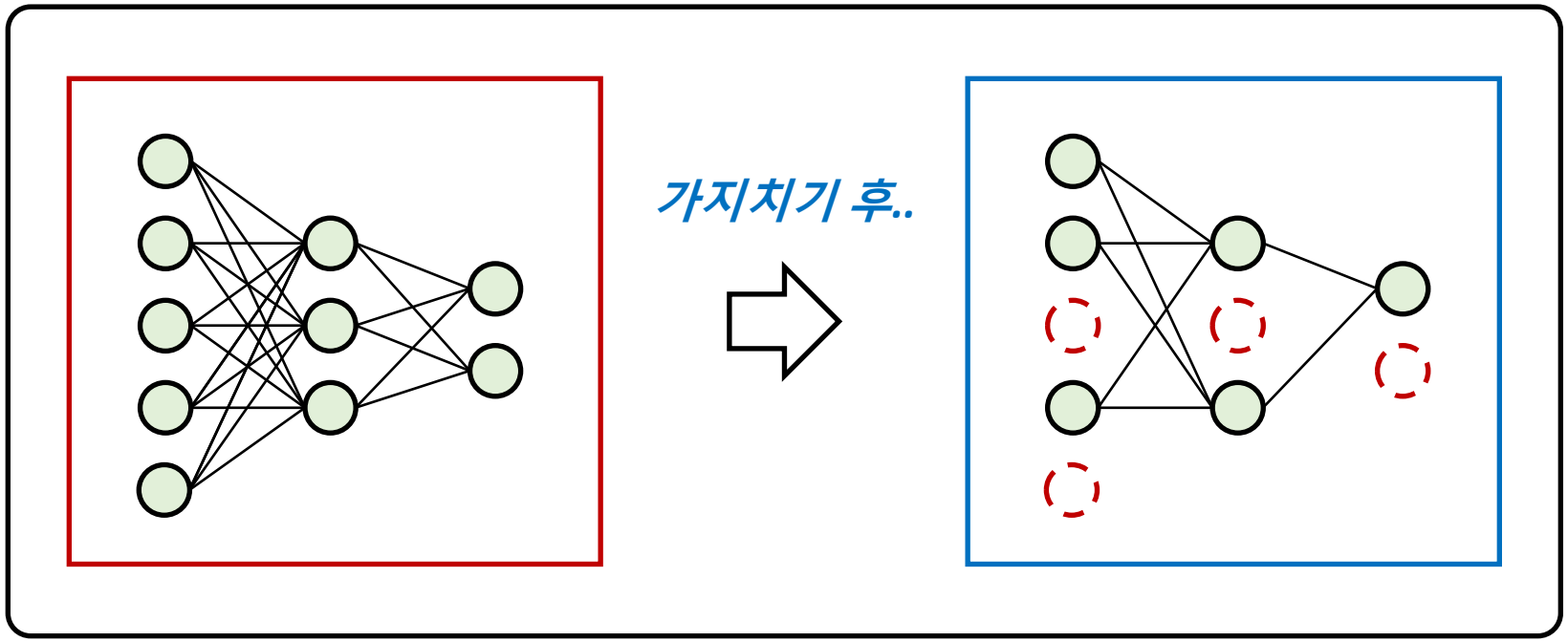
Transferring knowledge to smaller networks – Knowledge distillation

Designing inherently efficient architectures – MobileNetv2, ShuffleNet

가중치 가지치기의 정의

"***Weight pruning means*** eliminating unnecessary values in the ***weight*** tensors."

그림으로 이해해보자면?



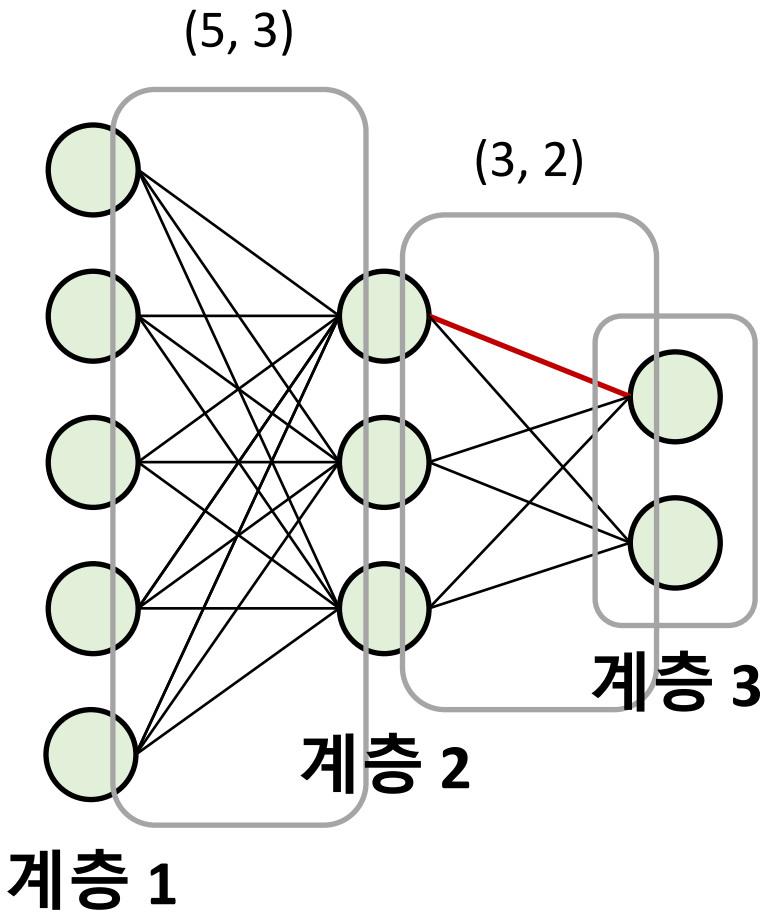
가지치기의 ***기준***은?

금일 실습할 가지치기 기준

1. Weights의 ***절댓값***을 기준으로 가지치기

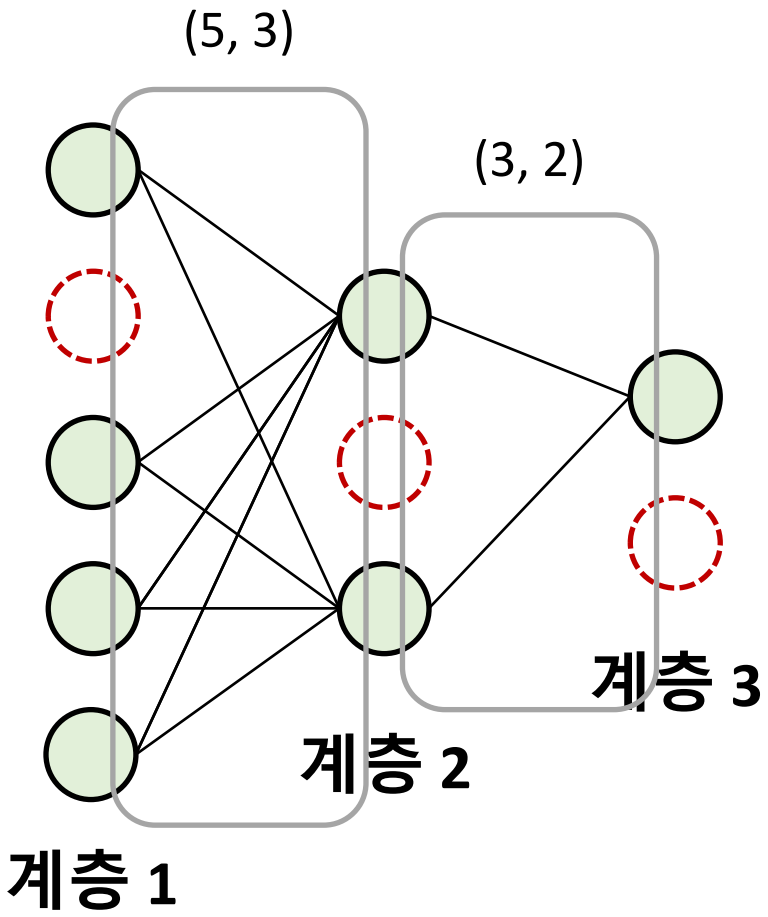
2. Weights의 ***L2Norm***을 기준으로 가지치기 → 금일의 구현 목표 !

절댓값을 기준으로 가지치기



"*Weights의 절대값을 기준*으로 *정렬*하여
작은 순서대로 가지치기"

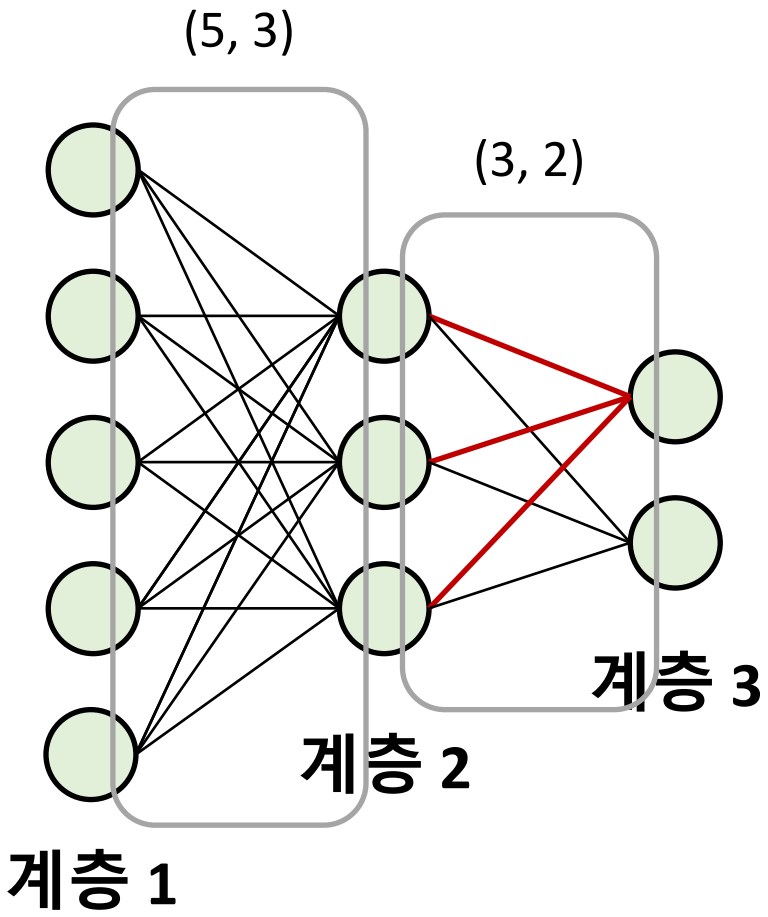
절댓값을 기준으로 가지치기



(3, 2) weights의 가지치기를 수행한다고 가정하면,

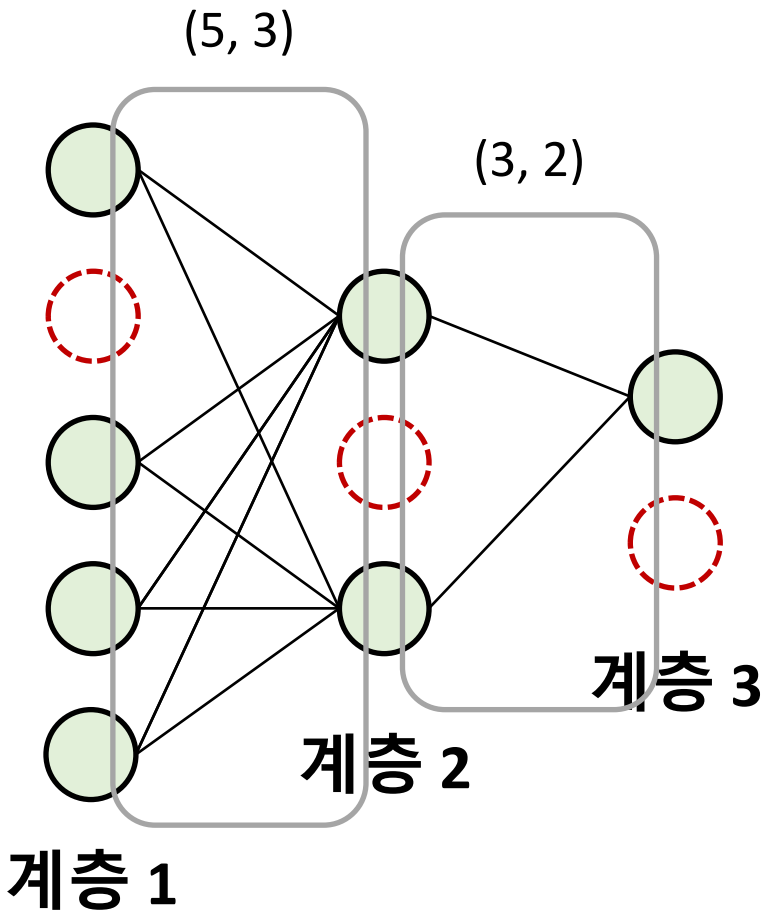
$$\begin{bmatrix} 0.5 & 0.21 \\ -0.1 & 0.12 \\ -0.4 & 0.35 \end{bmatrix} \xrightarrow{abs(\cdot)} \begin{bmatrix} 0.5 & 0.21 \\ 0.1 & 0.12 \\ 0.4 & 0.35 \end{bmatrix}$$

L2Norm을 기준으로 가지치기



"한 개의 뉴런에 대한 Weights에 대하여
N2Norm을 기준으로 ***정렬***하여 가지치기"

L2Norm을 기준으로 가지치기



(3, 2) weights의 가지치기를 수행한다고 가정하면,

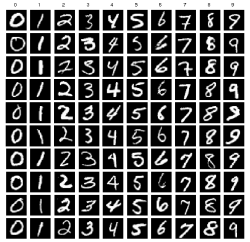
$$\begin{bmatrix} 0.5 & 0.21 \\ -0.1 & 0.12 \\ -0.4 & 0.35 \end{bmatrix} \xrightarrow{\mathrm{L2}Norm(\cdot)} \begin{bmatrix} 0.648 & 0.421 \end{bmatrix}$$

$$\mathrm{L2}Norm(\cdot) = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

$$= \sqrt{(0.5)^2 + (-0.1)^2 + (-0.4)^2}$$

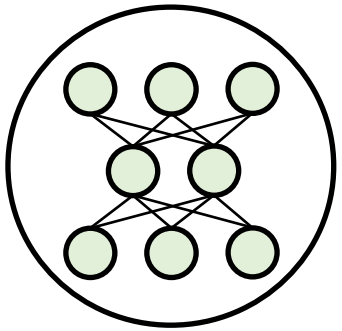
$$= 0.648$$

실습 과정



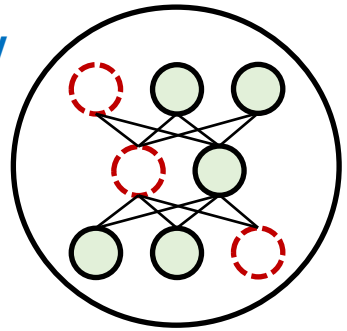
데이터 셋

학습



심층 학습 모델

가지치기



경량화 된 심층 학습 모델

“시각화 하기”

실습 과정

1. 실습 서버 접속하기
2. 환경설정하기
3. 실습 코드 설명
4. 실습 진행
5. 실습 환경 지우기 **(중요)**

실습 서버 접속하기

- **MobaXterm**을 이용해주세요.  MobaXterm
- 다음 명령어를 이용해서 **각 자리에 할당 된 IP**에 접속해주세요.

```
$ ssh -X -p 2222 com[LastDigit(s)]@[IP]
```

기본적으로 **5~26 번 PC**까지는 계정 번호와 IP주소 끝자리가 일치합니다.
ex) com**05**@143.248.159.**5** / com0502@143.248.159.**5**
ex) com**26**@143.248.159.**26** / com0502@143.248.159.**26**

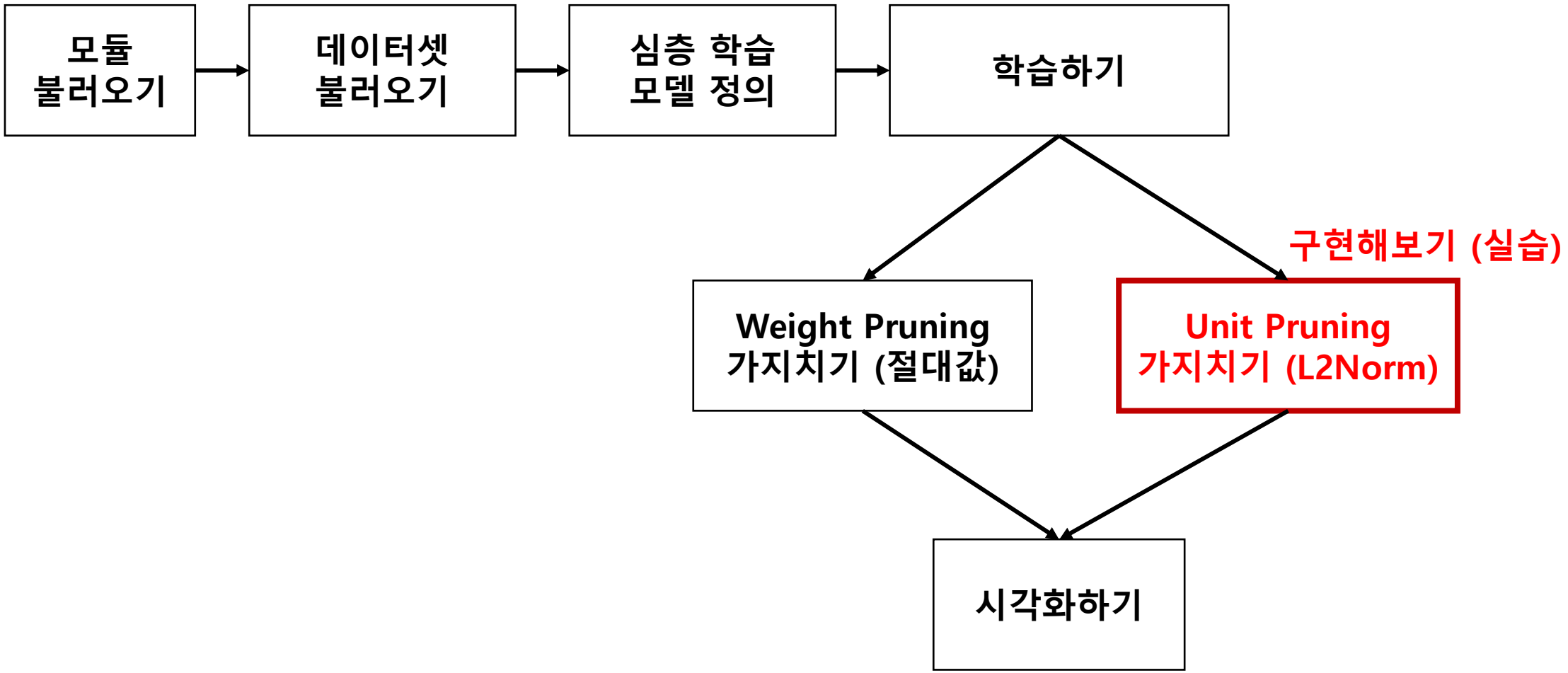
그러나, **예외**가 있습니다.
1번 PC: com**01**@143.248.159.**27** / com0102@143.248.159.**27**
2번 PC: com**02**@143.248.159.**28** / com0202@143.248.159.**28**
4번 PC: com**04**@143.248.159.**3** / com0402@143.248.159.**3**

환경설정하기

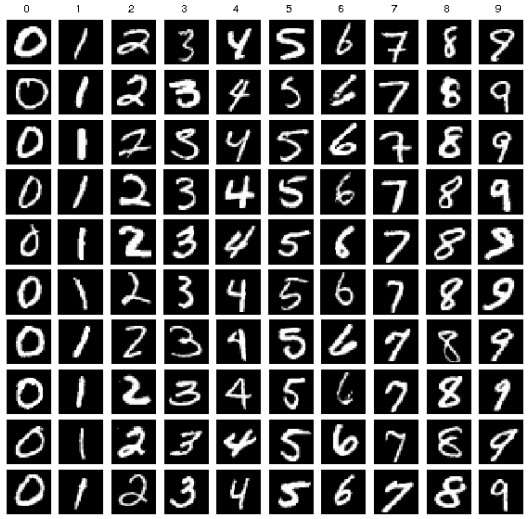
다음 명령어로 환경설정 해주세요.

```
$ source ~/.bashrc
$ conda activate pruning_simple
```

실습 코드 설명


모듈
불러오기
데이터셋
불러오기
심층 학습
모델 정의
학습하기
구현해보기 (실습)
Weight Pruning
가지치기 (절대값)
Unit Pruning
가지치기 (L2Norm)
시각화하기

데이터셋



MNIST

Fashion-MNIST

MNIST: 0부터 9까지 총 10개의 분류가능한 클래스

Fashion-MNIST: T-shirt, Dress 등 총 10개의 분류가능한 클래스

심층 학습 모델 & 학습

5개의 Fully-connected layer와 ReLU 함수로 구성.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1000)	785000
dense_1 (Dense)	(None, 1000)	1001000
dense_2 (Dense)	(None, 500)	500500
dense_3 (Dense)	(None, 200)	100200
dense_4 (Dense)	(None, 10)	2010

Total params: 2,388,710
Trainable params: 2,388,710
Non-trainable params: 0

마지막 단에 Softmax 함수를 통해 클래스 예측 확률을 0~1 사이로 정규화

Cross Entropy Loss 함수를 이용하여 Loss 학습 & 최소화

모듈 불러오기

```
# 필요한 라이브러리 불러오기
import math
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.axes as axes
import numpy as np
import os
# os.environ["CUDA_VISIBLE_DEVICES"]="0"
# os.environ["CUDA_VISIBLE_DEVICES"]="1"

import pandas as pd
import seaborn as sns
import tempfile
import tensorboard
import tensorflow as tf
import timeit
import zipfile

from IPython.core.pylabtools import figsize
from numpy import linalg as LA
from tensorflow.keras.models import load_model
from tensorflow_model_optimization.sparsity import keras as sparsity

# Tensorflow 버전 확인
print(tf.__version__)

plt.style.use('fivethirtyeight')
sns.set_context('notebook')
pd.set_option('display.max_rows', 30)
np.random.seed(1337)
%config InlineBackend.figure_format = 'retina'
%load_ext tensorboard
```

조교 가이드에 따라 0 또는 1로 설정해주세요.

데이터셋 불러오기

```python
# MNIST와 Fashion-MNIST 데이터셋 다운로드 / 불러오기 함수
def load_dataset(dataset='mnist'):
    """
    Loads and preprocesses the data for this task.
    Args:
      dataset: the name of the dataset to be used for this classification task.
        (mnist | fmnist)
    Returns:
      x_train: Features for training data
      x_test: Features for test data
      y_train: Labels for training data
      y_test: Labels for test data
      num_classes: Number of classes for the dataset
    """
```

MNIST와 Fashion-MNIST를 로드합니다.

```python
# MNIST 데이터셋 정의
mnist_x_train, mnist_x_test, mnist_y_train, mnist_y_test, num_classes, input_shape = load_dataset(dataset='mnist')

# Fashion-MNIST 데이터셋 정의
fmnist_x_train, fmnist_x_test, fmnist_y_train, fmnist_y_test, num_classes, input_shape = load_dataset(dataset='fmnist')
```

심층 학습 모델 정의

```python
# 심층 학습 모델 빌드 함수
def build_model_arch(input_shape, num_classes, sparsity=0.0):
    """
    Builds the model architecture
    Args:
      input_shape: The tuple describing the input shape
      num_classes: how many classes the data labels belong to
      sparsity: For compressing already sparse models, how much sparsity was used
    Returns:
      model: an un-compiled TF.Keras model with 4 hidden
        dense layers with shapes [1000, 1000, 500, 200]
    """

    model = tf.keras.Sequential()

    # 5 Fully-connected layer with ReLU function 정의
    # 한개의 클래스의 Probability를 최대화 하기위해서 마지막 단에 Softmax function 사용
    model.add(l.Dense(int(1000-(1000*sparsity)), activation='relu',
                      input_shape=input_shape)),
    model.add(l.Dense(int(1000-(1000*sparsity)), activation='relu'))
    model.add(l.Dense(int(500-(500*sparsity)), activation='relu'))
    model.add(l.Dense(int(200-(200*sparsity)), activation='relu'))
    model.add(l.Dense(num_classes, activation='softmax'))

    return model
```

네 개의 Fully Connected Layer와 마지막 Softmax layer로 구성된 모델을 빌드합니다.

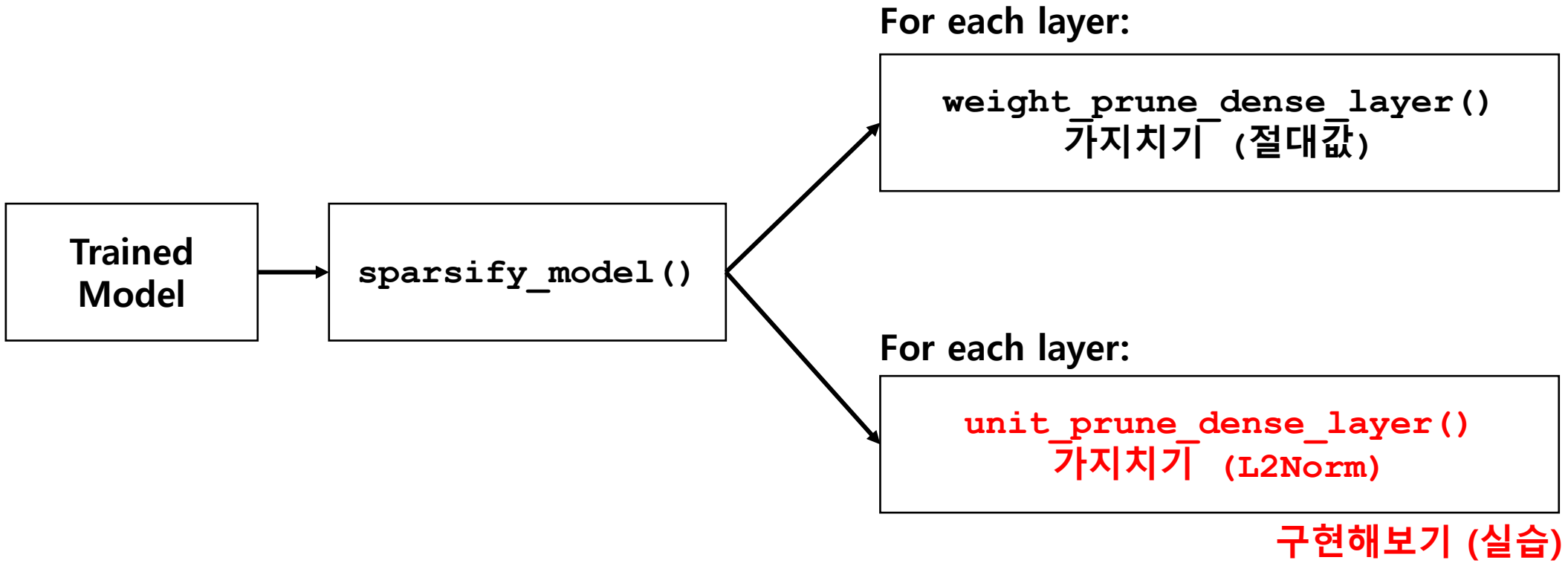
```python
# MNIST와 Fashion-MNIST 모델 빌드 (Sparsity == 0.0)
mnist_model_base = build_model_arch(input_shape, num_classes)
fmnist_model_base = build_model_arch(input_shape, num_classes)
```

학습하기 (Non-Sparse)

```python
# 심층 학습 모델 학습 함수
def make_nosparse_model(model, x_train, y_train, batch_size, epochs, x_test, y_test):
    """
    Training our original model, pre-pruning
    Args:
      model: Uncompiled Keras model
      x_train: Features for training data
      y_train: Labels for training data
      batch_size: Batch size for training
      epochs: Number of epochs for training
      x_test: Features for test data
      y_test: Labels for test data
    Returns:
      model: compiled model
      score: List of both final test loss and final test accuracy
    """
    model.compile(
        optimizer='adam',
        loss=tf.keras.losses.categorical_crossentropy,
        metrics=['accuracy'])

    # 심층 학습 모델 학습
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              verbose=1,
              callbacks=callbacks,
              validation_data=(x_test, y_test))

    # 심층 학습 모델 평가
    score = model.evaluate(x_test, y_test, verbose=0)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])

    return model, score
```

Optimizer: Adam Optimizer
Loss Function: Cross Entropy
Metric: Accuracy
Batch Size: 128
Epochs: 10

가지치기


Trained Model
sparsify_model()
For each layer:
weight_prune_dense_layer()
가지치기 (절대값)
For each layer:
unit_prune_dense_layer()
가지치기 (L2Norm)
구현해보기 (실습)

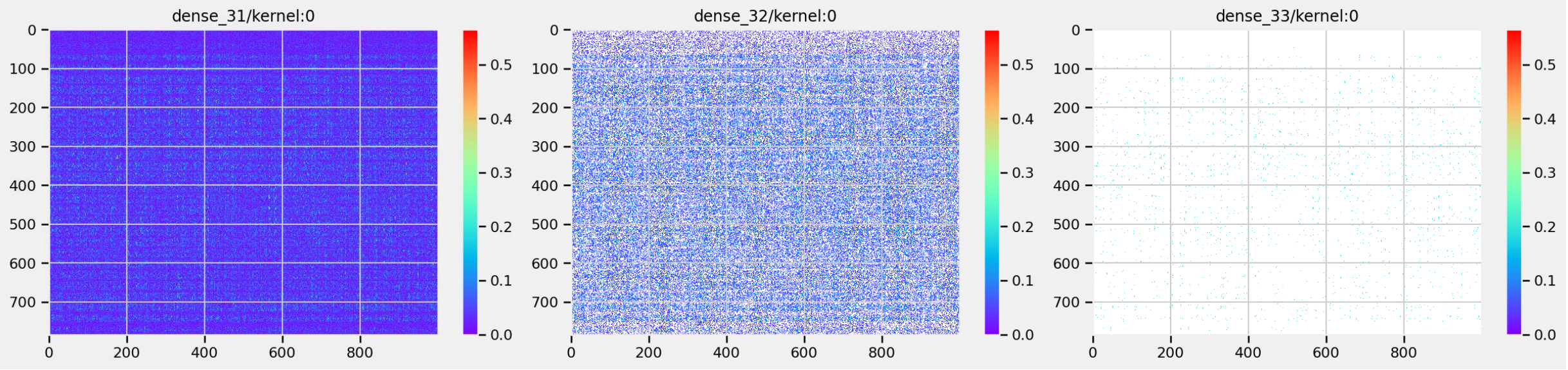
가지치기 (절대값)

```python
# Pruning 함수 (Weight)
def weight_prune_dense_layer(k_weights, b_weights, k_sparsity):
    """
    Takes in matrices of kernel and bias weights (for a dense
      layer) and returns the unit-pruned versions of each
    Args:
      k_weights: 2D matrix of the
      b_weights: 1D matrix of the biases of a dense layer
      k_sparsity: percentage of weights to set to 0
    Returns:
      kernel_weights: sparse matrix with same shape as the original
        kernel weight matrix
      bias_weights: sparse array with same shape as the original
        bias array
    """
```

```python
    # Kernel weights 복사
    kernel_weights = np.copy(k_weights)

    # Weight의 절대값을 기준으로 인덱스 정렬 (2D martix)
    ind = np.unravel_index(
        np.argsort(
            np.abs(kernel_weights),
            axis=None),
        kernel_weights.shape)

    # Pruning 하는 Weights 갯수 정의
    cutoff = int(len(ind[0])*k_sparsity)

    # 정렬된 인덱스를 기준으로 Weight pruning 수행 (0)
    sparse_cutoff_inds = (ind[0][0:cutoff], ind[1][0:cutoff])
    kernel_weights[sparse_cutoff_inds] = 0.

    # Kernel bias에 대하여 동일한 작업 수행
    bias_weights = np.copy(b_weights)

    ind = np.unravel_index(
        np.argsort(
            np.abs(bias_weights),
            axis=None),
        bias_weights.shape)

    cutoff = int(len(ind[0])*k_sparsity)

    sparse_cutoff_inds = (ind[0][0:cutoff])
    bias_weights[sparse_cutoff_inds] = 0.

    return kernel_weights, bias_weights
```

시각화하기

```python
# 모델 파라미터 Visulization
def visualize_model_weights(sparse_model):
    """
    Visualize the weights of the layers of the sparse model.
    For weights with values of 0, they will be represented by the color white
    Args:
      sparse_model: a TF.Keras model
    """
```



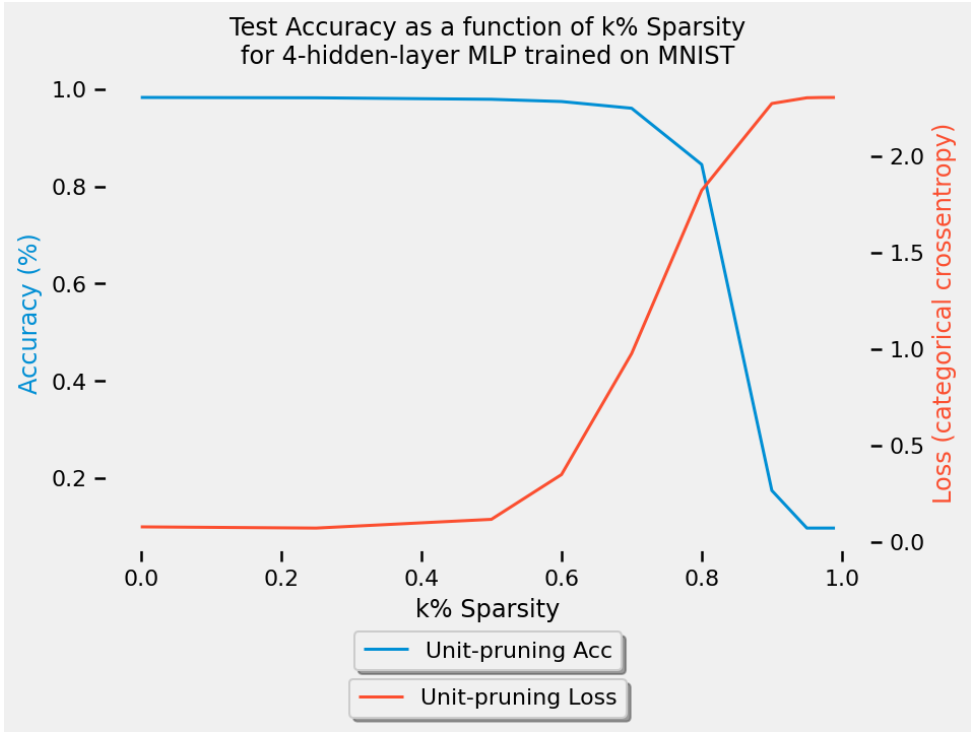
Sparsity = 0.0

Sparsity = 0.5

Sparsity = 0.99

시각화하기

```python
# MNIST 데이터셋 성능 Visualization
fig = plt.figure()
ax1 = fig.add_subplot(1, 1, 1)
plt.grid(b=None)
ax2 = ax1.twinx()
plt.grid(b=None)
plt.title('Test Accuracy as a function of k% Sparsity\nfor 4-hidden-layer MLP trained on MNIST')
# ax1.plot(sparsity_summary['k_sparsity'].values,
#          sparsity_summary['mnist_acc_weight'].values,
#          '#008fd5', linestyle=':', label='Weight-pruning Acc')
ax1.plot(sparsity_summary['k_sparsity'].values,
         sparsity_summary['mnist_acc_unit'].values,
         '#008fd5', linestyle='-', label='Unit-pruning Acc')
# ax2.plot(sparsity_summary['k_sparsity'].values,
#          sparsity_summary['mnist_loss_weight'].values,
#          '#fc4f30', linestyle=':', label='Weight-pruning Loss')
ax2.plot(sparsity_summary['k_sparsity'].values,
         sparsity_summary['mnist_loss_unit'].values,
         '#fc4f30', linestyle='-', label='Unit-pruning Loss')

ax1.set_ylabel('Accuracy (%)', color='#008fd5')
ax2.set_ylabel('Loss (categorical crossentropy)', color='#fc4f30')
ax1.set_xlabel('k% Sparsity')
ax1.legend(loc='upper center', bbox_to_anchor=(0.5, -0.15), shadow=True, ncol=2);
ax2.legend(loc='upper center', bbox_to_anchor=(0.5, -0.25), shadow=True, ncol=2);
plt.savefig('images/MNIST_sparsity_comparisons.png')
```



가지치기 (L2Norm)

무엇이든 물어보세요!

```python
# Pruning 함수 (Unit)
def unit_prune_dense_layer(k_weights, b_weights, k_sparsity):
    """
    Takes in matrices of kernel and bias weights (for a dense
      layer) and returns the unit-pruned versions of each
    Args:
      k_weights: 2D matrix of the
      b_weights: 1D matrix of the biases of a dense layer
      k_sparsity: percentage of weights to set to 0
    Returns:
      kernel_weights: sparse matrix with same shape as the original
        kernel weight matrix
      bias_weights: sparse array with same shape as the original
        bias array
    """

    # Kernel weights 복사

    # Column-wise L2 Norms 계산 후 인덱스 정렬

    # Pruning 하는 Weights 갯수 정의

    # 정렬된 인덱스를 기준으로 Weight pruning 수행 (0)

    # Kernel bias에 대하여 Weight를 기준으로 Pruning 된 인덱스에 대하여 0 적용

    return kernel_weights, bias_weights
```

실습 환경 지우기

다음 명령어로 실습 환경을 삭제해주세요.

```
$ conda deactivate
$ conda env remove -n pruning
```

실습 과제

1. Adam Optimizer 대신 SGD Optimizer 사용해보기

2. 기존 모델에서 두 개 이상의 Fully-connected Layer를 더 쌓아보기

3. L2Norm 기준 가지치기 함수 완성

4. CNN 기반 심층 학습 모델 정의 후 학습&가지치기 수행하기 (심화)