Machine learning 07

Byung Chang Chung

Gyeongsang National University

bcchung@gnu.ac.kr

Gyeongsang National University 경상국립대학교

Contents

- Training deep networks
  - gradient calculation
  - transfer learning

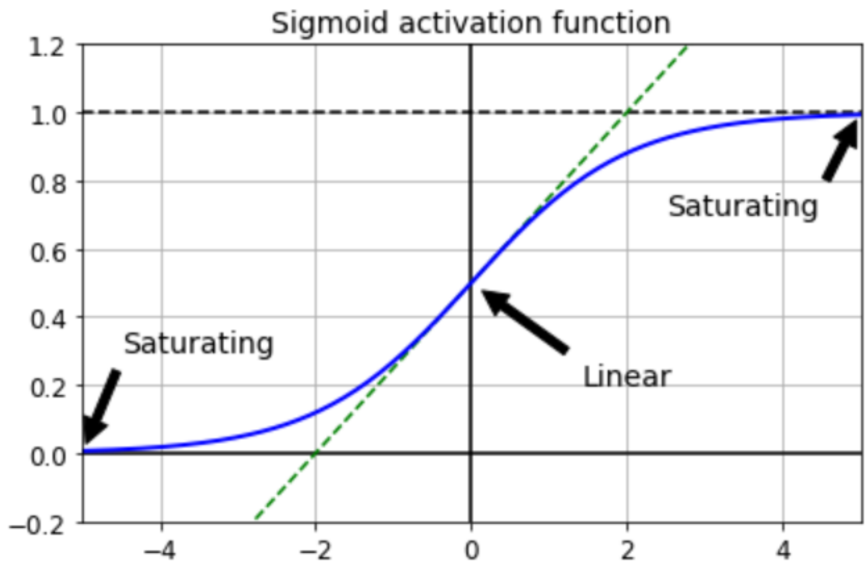
Gyeongsang National University 경상국립대학교

Vanishing gradient

- Definition
  - during each iteration of training each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight
  - the gradient will be vanishingly small, effectively preventing the weight from changing its value

Initialization

- Gradient tendency
  - as backpropagation proceeds, decrease the amount of variations



Gyeongsang National University 경상국립대학교

Initialization

- Glorot/Xavier initialization
  - variance in input weights
  - variance in output weights
  - same theory applies in gradient

Gyeongsang National University 경상국립대학교

Initialization

- Other initialization methods
  - according to the type of activation function

초기화 전략	활성화 함수	σ^2 (정규분포)
글로럿	활성화 함수 없음, 하이퍼볼릭 탄젠트, 로지스틱, 소프트맥스	$1 / fan_{avg}$
He	ReLU 함수와 그 변종들	$2 / fan_{in}$
르쿤	SELU	$1 / fan_{in}$

Gyeongsang National University 경상국립대학교

Initialization

- Initialization in keras

```
[name for name in dir(keras.initializers) if not name.startswith("_")]
```

```
['Constant',
 'GlorotNormal',
 'GlorotUniform',
 'HeNormal',
 'HeUniform',
 'Identity',
 'Initializer',
 'LecunNormal',
 'LecunUniform',
 'Ones',
 'Orthogonal',
 'RandomNormal',
 'RandomUniform',
 'TruncatedNormal',
 'VarianceScaling',
 'Zeros',
 'constant',
 'deserialize',
 'get',
 'glorot_normal',
 'glorot_uniform',
 'he_normal',
 'he_uniform',
 'identity',
 'lecun_normal',
 'lecun_uniform',
 'ones',
 'orthogonal',
 'random_normal',
 'random_uniform',
 'serialize',
 'truncated_normal',
 'variance_scaling',
 'zeros']
```

```
keras.layers.Dense(10, activation="relu", kernel_initializer="he_normal")
```

```
<keras.layers.core.Dense at 0x7f98d0126828>
```

```
init = keras.initializers.VarianceScaling(scale=2., mode='fan_avg',
                                          distribution='uniform')
keras.layers.Dense(10, activation="relu", kernel_initializer=init)
```

```
<keras.layers.core.Dense at 0x7f98207e8240>
```

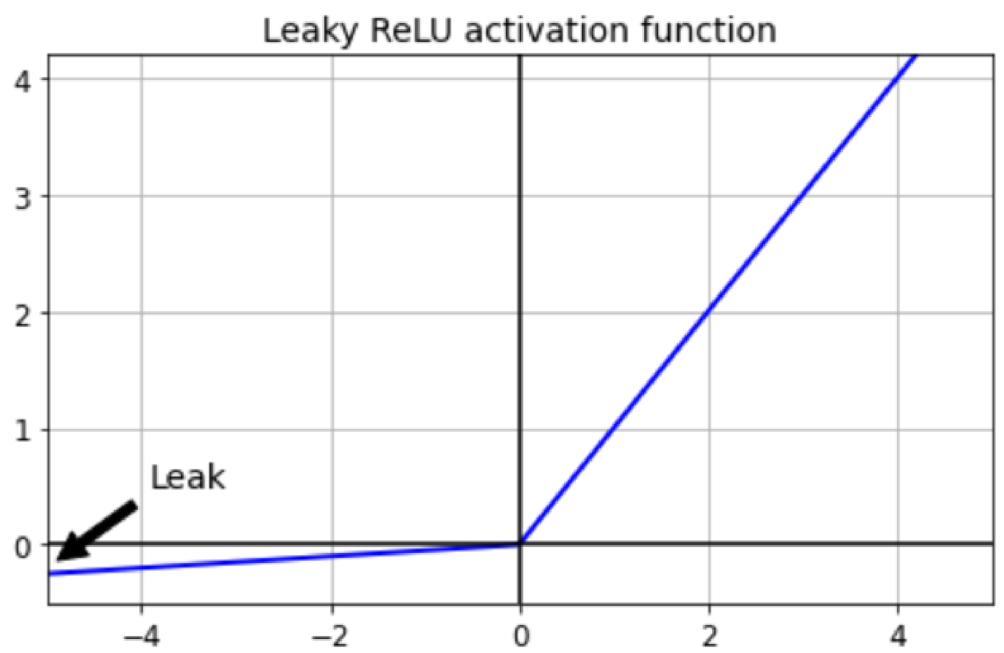
Gyeongsang National University 경상국립대학교

Activation function

- Activation function which does not converge
  - types of activation function
    - sigmoid
    - ReLU
    - LeakyReLU
    - ELU
    - Scaled ELU

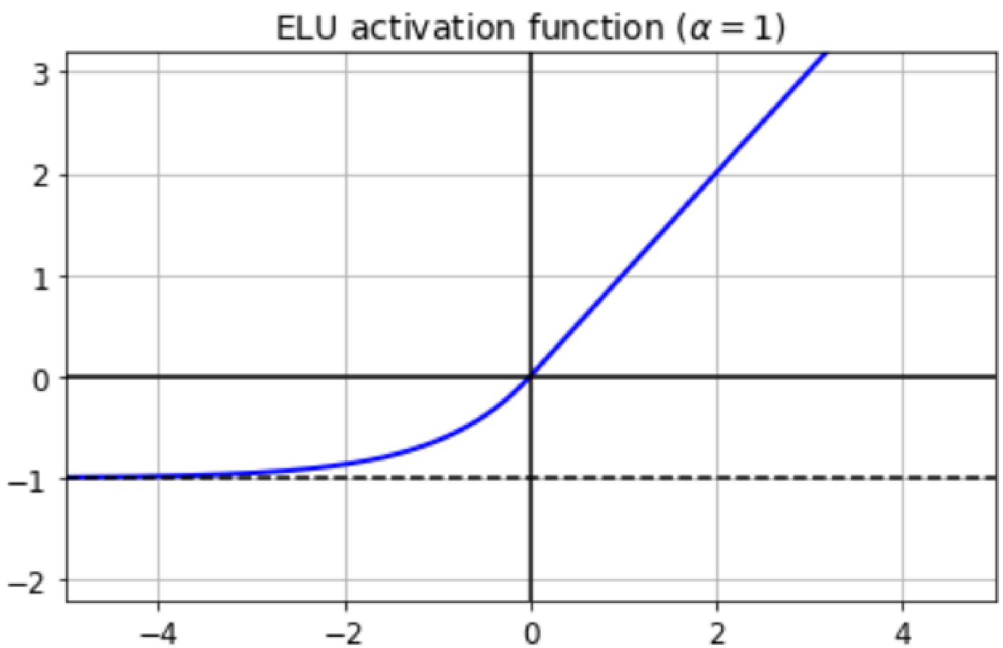
Activation function

- ReLU types (Leaky ReLU)



Activation function

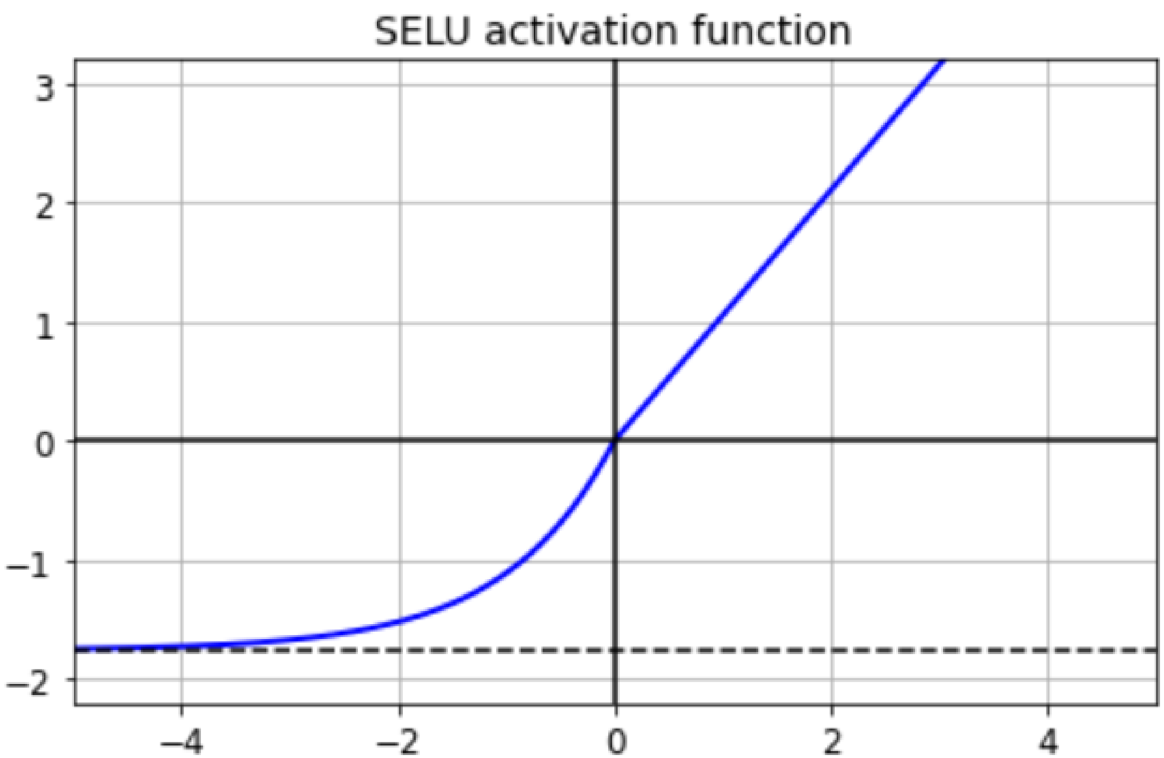
- ReLU types (ELU)



Gyeongsang National University 경상국립대학교

Activation function

- ReLU types (SELU)



Activation function

- ReLU types in keras

```
[m for m in dir(keras.activations) if not m.startswith("_")]
```

```
['deserialize',
 'elu',
 'exponential',
 'gelu',
 'get',
 'hard_sigmoid',
 'linear',
 'relu',
 'selu',
 'serialize',
 'sigmoid',
 'softmax',
 'softplus',
 'softsign',
 'swish',
 'tanh']
```

```
tf.random.set_seed(42)
np.random.seed(42)

model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(300, kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(),
    keras.layers.Dense(100, kernel_initializer="he_normal"),
    keras.layers.LeakyReLU(),
    keras.layers.Dense(10, activation="softmax")
])
```

```
[m for m in dir(keras.layers) if "relu" in m.lower()]
```

```
['LeakyReLU', 'PReLU', 'ReLU', 'ThresholdedReLU']
```

Batch normalization

- A solution to treat vanishing gradient
  - normalization according to the mean and variance from inputs

식 11-3 배치 정규화 알고리즘

1. $\boldsymbol{\mu}_B = \dfrac{1}{m_B}\displaystyle\sum_{i=1}^{m_B}\mathbf{x}^{(i)}$

2. ${\boldsymbol{\sigma}_B}^2 = \dfrac{1}{m_B}\displaystyle\sum_{i=1}^{m_B}\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_B\right)^2$

3. $\hat{\mathbf{x}}^{(i)} = \dfrac{\mathbf{x}^{(i)} - \boldsymbol{\mu}_B}{\sqrt{{\boldsymbol{\sigma}_B}^2 + \varepsilon}}$

4. $\mathbf{z}^{(i)} = \boldsymbol{\gamma} \otimes \hat{\mathbf{x}}^{(i)} + \boldsymbol{\beta}$

GNU Gyeongsang National University 경상국립대학교

Batch normalization

- Batch normalization in keras

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(10, activation="softmax")
])
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 784)               0
_________________________________________________________________
batch_normalization (BatchNo (None, 784)               3136
_________________________________________________________________
dense_212 (Dense)            (None, 300)               235500
_________________________________________________________________
batch_normalization_1 (Batch (None, 300)               1200
_________________________________________________________________
dense_213 (Dense)            (None, 100)               30100
_________________________________________________________________
batch_normalization_2 (Batch (None, 100)               400
_________________________________________________________________
dense_214 (Dense)            (None, 10)                1010
=================================================================
Total params: 271,346
Trainable params: 268,978
Non-trainable params: 2,368
_________________________________________________________________
```

Batch normalization

- Batch normalization in keras
  - normalization before activation

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(300, use_bias=False),
    keras.layers.BatchNormalization(),
    keras.layers.Activation("relu"),
    keras.layers.Dense(100, use_bias=False),
    keras.layers.BatchNormalization(),
    keras.layers.Activation("relu"),
    keras.layers.Dense(10, activation="softmax")
])
```

Gradient clipping

- Clipping
  - a way to reduce exploding gradient
  - discarding gradient which is greater than certain threshold
  - usually applied in recurrent neural network

Gradient clipping

- Clipping in keras

```
optimizer = keras.optimizers.SGD(clipvalue=1.0)
```

```
optimizer = keras.optimizers.SGD(clipnorm=1.0)
```

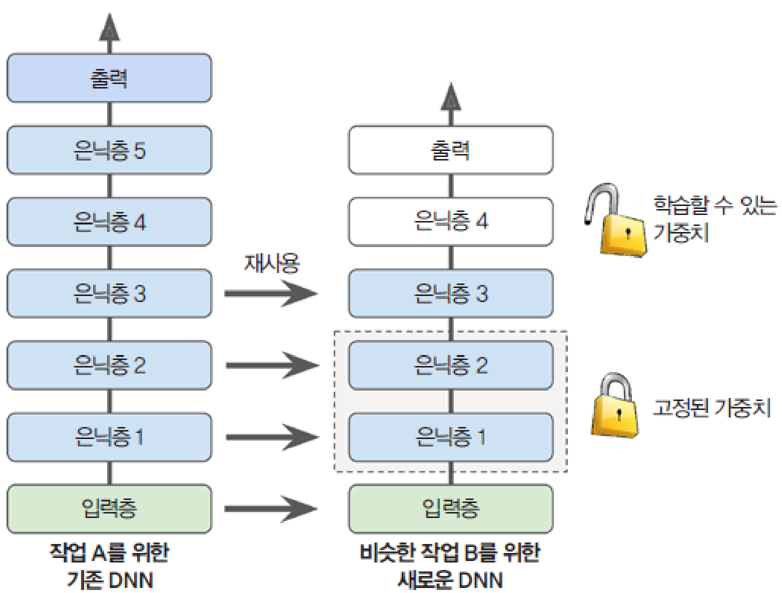
Gyeongsang
National
University
경상국립대학교

Transfer learning

- Definition
  - learning to reuse the sublayer of the neural network
  - increase the speed of learning
  - mitigate the burden of learning data size

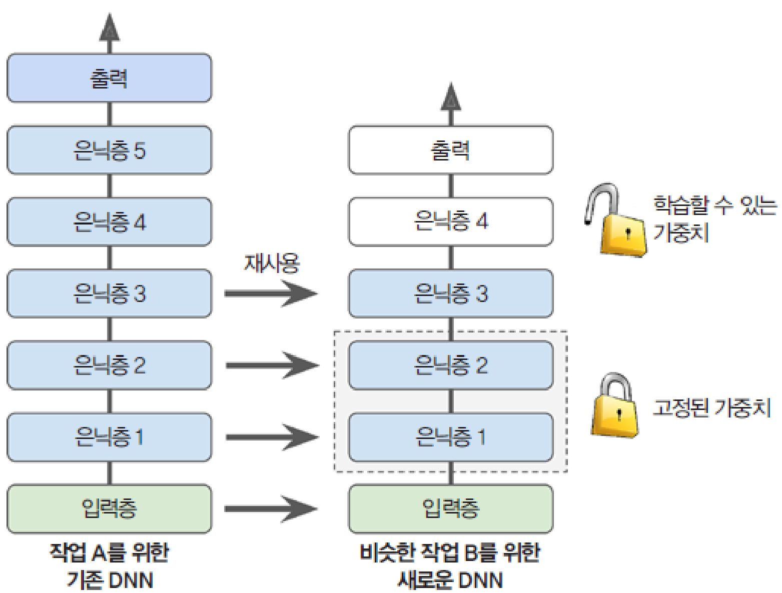
Transfer learning

- Example
  - Assume that you already have the image classifier which can classify 100 categories such as animal, plant and car



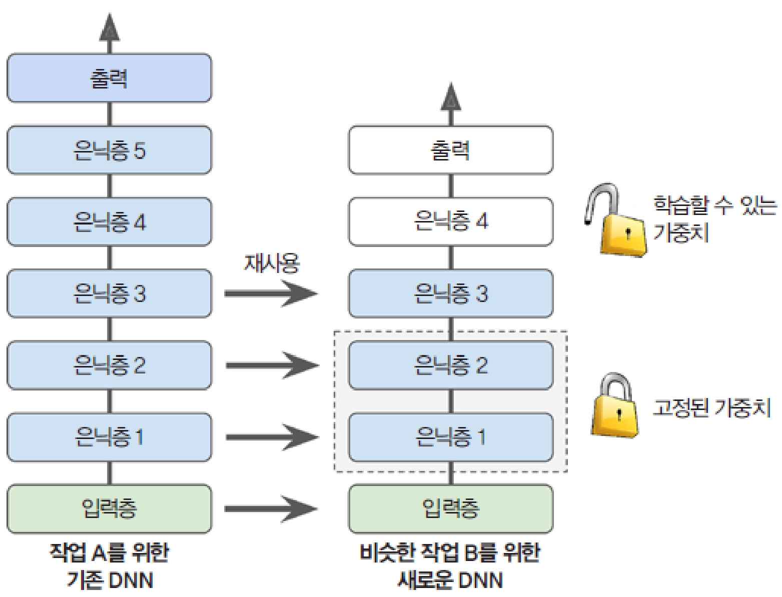
Transfer learning

- Example
  - Lower hidden layer perform low-level classification of images (fixed in transfer learning)



Transfer learning

- Example
  - Higher hidden layer perform precise classification of images (update weights in transfer learning)



Transfer learning

- Implementation in keras
  - model A: fashion MNIST without sandals and shirts
  - model B: fashion MNIST only with sandals and shirts

```
def split_dataset(X, y):
    y_5_or_6 = (y == 5) | (y == 6) # sandals or shirts
    y_A = y[~y_5_or_6]
    y_A[y_A > 6] -= 2 # class indices 7, 8, 9 should be moved to 5, 6, 7
    y_B = (y[y_5_or_6] == 6).astype(np.float32) # binary classification task: is it a shirt (class 6)?
    return ((X[~y_5_or_6], y_A),
            (X[y_5_or_6], y_B))

(X_train_A, y_train_A), (X_train_B, y_train_B) = split_dataset(X_train, y_train)
(X_valid_A, y_valid_A), (X_valid_B, y_valid_B) = split_dataset(X_valid, y_valid)
(X_test_A, y_test_A), (X_test_B, y_test_B) = split_dataset(X_test, y_test)
X_train_B = X_train_B[:200]
y_train_B = y_train_B[:200]
```

Transfer learning

- Implementation in keras
  - model A training

```
model_A = keras.models.Sequential()
model_A.add(keras.layers.Flatten(input_shape=[28, 28]))
for n_hidden in (300, 100, 50, 50, 50):
    model_A.add(keras.layers.Dense(n_hidden, activation="selu"))
model_A.add(keras.layers.Dense(8, activation="softmax"))
```

```
model_A.compile(loss="sparse_categorical_crossentropy",
                optimizer=keras.optimizers.SGD(learning_rate=1e-3),
                metrics=["accuracy"])
```

```
history = model_A.fit(X_train_A, y_train_A, epochs=20,
                    validation_data=(X_valid_A, y_valid_A))
```

GNU
Gyeongsang National University
경상국립대학교

Transfer learning

- Implementation in keras
  - model B training using transfer learning

```
model_A_clone = keras.models.clone_model(model_A)
model_A_clone.set_weights(model_A.get_weights())
model_B_on_A = keras.models.Sequential(model_A_clone.layers[:-1])
model_B_on_A.add(keras.layers.Dense(1, activation="sigmoid"))
```

```
for layer in model_B_on_A.layers[:-1]:
    layer.trainable = False

model_B_on_A.compile(loss="binary_crossentropy",
                     optimizer=keras.optimizers.SGD(learning_rate=1e-3),
                     metrics=["accuracy"])
```

Gyeongsang National University 경상국립대학교

Feel free to question

Through e-mail & LMS

본 자료의 연습문제는 수업의 본교재인
한빛미디어, Hands on Machine Learning(2판)에서 주로 발췌함