DL Chap 13.3

입력 특성 전처리

- ✓ 데이터 파일 준비 전
- ✓ 데이터 API로 데이터 적재 시(동적)
- ✓ 전처리 층을 모델에 직접 포함

입력 특성 전처리

```
# Lambda 층으로 표준화를 수행하는 층 구현
means = np.mean(X_train, axis=0, keepdims=True)
stds = np.std(X_train, axis=0, keepdims=True)
eps = keras.backend.epsilon()
model = keras.models.Sequential([
                                keras. layers.Lambda(lambda inputs: (inputs - means) / (stds + eps)), # 각 특성의 평균을 뺀 뒤, 표준편차로 나눔
                                [...]    # 다른층
])
```

- ✓ Lambda 층을 사용해 표준화를 수행하는 층 구현하기
- ✓ 각 특성의 평균을 빼고, 표준편차로 나눔

* 0으로 나눗셈 되는 걸 막기 위해 작은 수 더함

입력 특성 전처리

✓ 완전한 사용자 정의 층_ Standardization

```
class Standardization(keras.layers.Layer):
  def adapt(self, data_sample):
    self.means_ = np.mean(data_sample, axis=0, keepdims=True)
    self.stds_ = np.std(data_sample, axis=0, keepdims=True)
  def call(self, inputs):
    return (inputs - self.means_) / (self.stds_ + keras.backend.epsilon())
```

```
std_layer = Standardization()
std_layer.adapt(data_sample)
```


일반적인 층처럼 사용 가능

원-핫 벡터를 사용해 범주형 특성 인코딩하기

범주의 개수가 작음 → 원-핫 인코딩 사용

```
vocab = ["<1H OVEAN", "INLAND", "NEAR OCEAN", "NEAR BAY", "ISLAND"]
indices = tf.range(len(vocab), dtype=tf.int64)
table_init = tf.lookup.KeyValueTensorInitializer(vocab, indices)
num_oov_buckets = 2
table = tf.lookup.StaticVocabularyTable(table_init, num_oov_buckets)
```

✓ oov 버킷을 사용하는 이유?

범주 개수가 많고, 데이터셋이 크거나 범주가 자주 바뀌면 전체 범주 리스트를 구하는 것이 어려움

↓

샘플 데이터 기반으로 어휘 사전 정의, 샘플 데이터에 없는 다른 범주를 oov 버킷에 추가

원-핫 벡터를 사용해 범주형 특성 인코딩하기

```
categories = tf.constant(["NEAR BAY", "DESERT", "INLAND", "INLAND"])
cat_indices = table.lookup(categories)
cat_indices
```

```
<tf.Tensor: shape=(4,), dtype=int64, numpy=array([3, 5, 1, 1])>
```

```
cat_one_hot = tf.one_hot(cat_indices, depth=len(vocab) + num_oov_buckets)
cat_one_hot
```

* tf.one_hot(): 어휘 사전 크기와 oov 버킷 수를 더한 인덱스 총 개수를 지정해야 함

```
<tf.Tensor: shape=(4, 7), dtype=float32, numpy=
array([[0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0.]], dtype=float32)>
```

✓ 인덱스를 원-핫 벡터로 바꾸려면 -> 이 층을 모델 시작 부분에 추가 후, tf.one_hot()함수가 적용된 Lambda 층 추가

임베딩을 사용해 범주형 특성 인코딩하기

표현 학습 representation learning

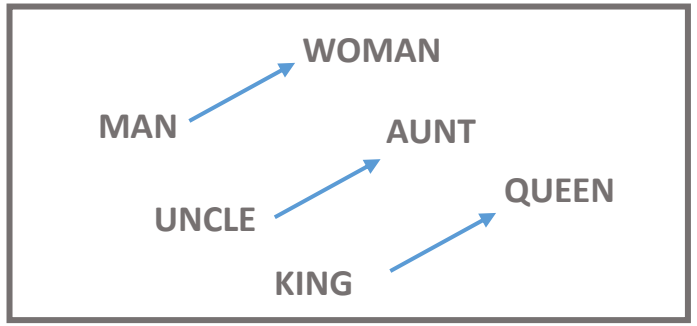
- ✓ 샘플을 묘사하는데 가장 뛰어난 특성과 이런 특성을 원본 데이터에서 추출하는 방법을 학습하는 것

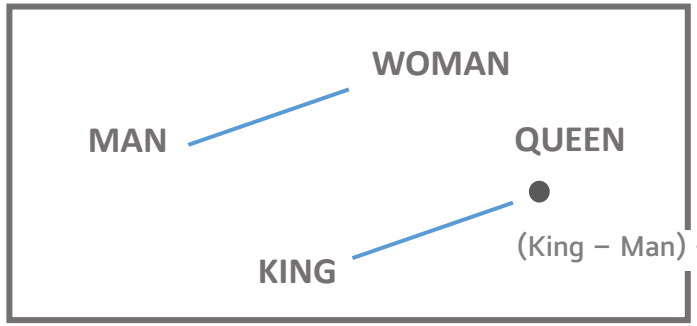
임베딩을 사용해 범주형 특성 인코딩하기

단어 임베딩

- ✓ 원-핫 인코딩과는 다른 방식으로 단어를 공간상의 벡터로 표현하는 기술
- ✓ 단어 임베딩을 사용할 경우 비슷한 의미가 있는 단어들이 서로 가까운 곳에 나타나게 됨

  => 벡터 상에 단어의 의미를 포함시킬 수 있음





임베딩을 사용해 범주형 특성 인코딩하기

✓ 임베딩 행렬 생성: 범주와 oov 버킷마다 하나의 행, 임베딩 타원마다 하나의 열을 가짐

```
embedding_dim = 2
embed_init = tf.random.uniform([len(vocab) + num_oov_buckets, embedding_dim])
embedding_matrix = tf.Variable(embed_init)
```

```
embedding_matrix
```

```
<tf.Variable 'Variable:0' shape=(7, 2) dtype=float32, numpy=
array([[0.9046538 , 0.29638696],
       [0.41187608, 0.9010956 ],
       [0.70784247, 0.11329353],
       [0.00864708, 0.10841095],
       [0.8625331 , 0.81973505],
       [0.18372226, 0.5557064 ],
       [0.4466331 , 0.8234887 ]], dtype=float32)>
```

임베딩을 사용해 범주형 특성 인코딩하기

```
categories = tf.constant(["NEAR BAY", "DESERT", "INLAND", "INLAND"])
cat_indices = table.lookup(categories)
cat_indices
```

```
<tf.Tensor: shape=(4,), dtype=int64, numpy=array([3, 5, 1, 1])>
```

```
tf.nn.embedding_lookup(embedding_matrix, cat_indices)
```

* tf.nn.embedding_lookup(): 임베딩 행렬에서 주어진 인덱스에 해당하는 행을 찾음

```
<tf.Tensor: shape=(4, 2), dtype=float32, numpy=
array([[0.00864708, 0.10841095],
       [0.18372226, 0.5557064 ],
       [0.41187608, 0.9010956 ],
       [0.41187608, 0.9010956 ]], dtype=float32)>
```

임베딩을 사용해 범주형 특성 인코딩하기

- ✓ keras.layers.Embedding 층 생성 시, 임베딩 행렬을 랜덤하게 초기화하고 어떤 범주 인덱스로 호출될 때 임베딩 행렬에 있는 그 인덱스의 행 반환

```
regular_inputs = keras.layers.Input(shape=[8])
categories = keras.layers.Input(shape=[], dtype=tf.string)
cat_indices = keras.layers.Lambda(lambda cats: table.lookup(cats))(categories)
cat_embeded = keras.layers.Embedding(input_dim=6, output_dim=2)(cat_indices)
encoded_inputs = keras.layers.concatenate([regular_inputs, cat_embeded])
outputs = keras.layers.Dense(1)(encoded_inputs)
model = keras.models.Model(inputs=[regular_inputs, categories],
                           outputs=[outputs])
```

- ✓ keras.layers.TextVecterization 층 사용 가능하다면 adapt() 메서드를 호출해 샘플 데이터에서 어휘 사전 추출

케라스 전처리 층

keras.layers.Normalization 층	특성 표준화 수행
TextVectorization 층	입력에 있는 각 단어를 어휘 사전에 있는 인덱서를 인코딩

층을 만들고 샘플 데이터로 adapt() 메서드 호출

이후, 일반적인 층처럼 모델에 사용 가능

\+

* keras.layers.Discretizaion 층 -> 연속적인 데이터를 몇 개의 구간으로 나누고, 각 구간을 원-핫 벡터로 인코딩

* PreprocessingStage 클래스 -> 여러 전처리 층 연결 가능

케라스 전처리 층

TextVectorization 층

✓ 단어 인덱스 대신 단어 카운트 벡터를 출력하는 옵션을 가짐

Ex)
[“and”, “basketball”, “more”] → [1, 0, 2]

↳ 단어 순서를 완전히 무시하므로 BOW라 부름

✓ 단어 카운트는 자주 등장하는 단어의 중요도를 줄이는 방향으로 정규화 되어야 함

TF-IDF: 전체 샘플 수를 단어가 등장하는 훈련 샘플 개수로 나눈 로그를 계산,
이후 단어 카운트와 곱하는 기법

Thank you