exam 개인학습

Lec 2

AI summary Try on this page

Type something...

01) Derivatives and Implementation of word2vec algorithm

01_Derivatives

- 플렉슬 앱에다가 solution 정리해둠.

02_Implementation of word2vec algorithm

1. Sigmoid

```python
def sigmoid(x):
    """
    Compute the sigmoid function for the input here.
    Arguments:
    x -- A scalar or numpy array.
    Return:
    s -- sigmoid(x)
    """

    ### YOUR CODE HERE (~1 Line)
    s = 1 / (1 + np.exp(-x))
    ### END YOUR CODE

    return s
```

- Sigmoid는 지금까지 여러번 구현해봤기 때문에 간단하게 구현할 수 있다.

2. naiveSoftmaxLossAndGradient

- Navie한 softmaxloss와 gradient를 구하는 문제이다.

$$P(w_o|w_c) = \hat{y}_o = \frac{\exp(\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{w \in Vocab} \exp(\mathbf{u}_w^T \mathbf{v}_c)}$$

We can then define a *naive softmax* based objective function to minimize during the optimization process which further simplifies to a *negative log-likelihood* function.

$$\text{i.e. } J = H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j) = -\log(\hat{y}_o); \text{ since } y_j = 1 \text{ when } i = o, \text{ and } y_j = 0 \text{ otherwise.}$$

$$\Rightarrow J_{naive-softmax} = -\log P(O = o|C = c) = -\log(\hat{y}_o)$$

- outer/center를 기준으로 softmax를 이용해서 확률을 구해준 후 정답 label과 곱하여 loss function을 구한다.
- label이 1또는 0이기 때문에 -log(P)의 평균을 구해주면 된다.
- softmax함수를 이용해서 전체를 다 구해준 다음 outer에 해당하는 index를 이용해서 그 값만 추출해준다.

```
loss = 0
gradCenterVec = np.zeros(centerWordVec.shape)
gradOutsideVecs = np.zeros(outsideVectors.shape)

y_hat = softmax(np.dot(outsideVectors, centerWordVec)) # y_hat 구하기
y = np.zeros(y_hat.shape) # 정답 레이블 만들기, (10, )
y[outsideWordIdx] = 1

loss = -np.log(y_hat[outsideWordIdx])
gradCenterVec = np.dot(outsideVectors.T, (y_hat - y)) # (3, )
gradOutsideVecs = np.outer(y_hat - y, centerWordVec) # (10, 3)
```

```
loss = -np.log(y_hat[outsideWordIdx])
gradCenterVec = np.dot(outsideVectors.T, (y_hat - y)) # (3, )
#gradOutsideVecs = np.outer(y_hat - y, centerWordVec) # (10, 3)
gradOutsideVecs = np.dot((y_hat - y).reshape(-1, 1), centerWordVec.reshape(-1, 1).T)
### END YOUR CODE
```

- 외적을 사용해도 되고 행렬곱을 사용해도 된다.

3. negSamplingLossAndGradient

- sigmoid함수를 이용해서 cost function과 gradient를 계산하자.
- size가 맞지 않는 부분은 전치행렬을 시켜줘서 계산해주었다.

```
indices = [outsideWordIdx] + negSampleWordIndices
### YOUR CODE HERE (~10 Lines)
u_o = outsideVectors[outsideWordIdx]
u_k = outsideVectors[negSampleWordIndices]
v_c = centerWordVec

loss = -np.log(sigmoid(np.dot(u_o.T, v_c))) - np.sum(np.log(sigmoid(-np.dot(u_k, v_c))))

gradCenterVec = -(1-sigmoid(np.dot(u_o.T, v_c))) * u_o - np.dot(-(1-sigmoid(-np.dot(u_k, v_c))).T, u_k)

gradOutsideVecs = np.zeros(outsideVectors.shape)
gradOutsideVecs[outsideWordIdx, :] = (sigmoid(np.dot(u_o.T, v_c)) - 1) * v_c
unique, indices, counts = np.unique(negSampleWordIndices, return_index=True, return_counts=True)
sigmoidUniqueSamplesCenter = sigmoid(-np.dot(u_k, v_c))[indices]
gradOutsideVecs[unique, :] = - ((sigmoidUniqueSamplesCenter - 1) * counts).reshape(-1, 1)* v_c.reshape(1, -1)
```

- 마지막 negative sampling의 경우 count를 해주어서 그 수만큼 곱해주었다.

4. skipgram

- 각각의 loss와 gradient를 구했으니 합치는 과정을 진행할 것이다.
- centerWordVectors에서 현재 center word의 벡터를 추출한다.
- 그 후 outsideWords를 하나씩 돌면서 진행한다.

```
loss = 0.0
gradCenterVecs = np.zeros(centerWordVectors.shape)
gradOutsideVectors = np.zeros(outsideVectors.shape)

### YOUR CODE HERE (~8 Lines)
centerWordIdx = word2Ind[currentCenterWord]
centerWordVec = centerWordVectors[centerWordIdx]

for word in outsideWords:
    outsideWordIdx = word2Ind[word]
    gradloss, gradCenterVec, gradOutsideVecs = word2vecLossAndGradient(centerWordVec, outsideWordIdx, outsideVectors, dataset)
    loss += gradloss
    gradCenterVecs[centerWordIdx] += gradCenterVec
    gradOutsideVectors += gradOutsideVecs
```

5. sgd

```
loss = None
### YOUR CODE HERE (~2 lines)
loss, grad = f(x)
x -= step * grad
```

- step_size * grad 를 빼줘서 parameter를 업데이트 시켜준다.
- 이전에 구현해봤던 코드들과 동일하다.