Loss was generated taking negative logs of sigmoid and softmax functions.
Gradients were generated by multiplying other word vectors with (values - 1) where values are output of softmax/sigmoid functions. For centre word, outside word vectors were multiplied while for outside words, centre word vectors was multiplied.

Sigmoid function-

```
def sigmoid(x):
    s = 1 / (1 + np.exp(-x))
    return s
```

Naive softmax loss and gradient-

```
def naiveSoftmaxLossAndGradient(centerWordVec, outsideWordIdx, outsideVectors, dataset):
    matrix = np.matmul(outsideVectors, centerWordVec)
    probability = softmax(matrix)
    loss = -np.log(probability[outsideWordIdx])
    values = probability.copy()
    values[outsideWordIdx] = values[outsideWordIdx] - 1
    gradCenterVec = np.matmul(outsideVectors.T, values)
    gradOutsideVecs = np.outer(values, centerWordVec)
    return loss, gradCenterVec, gradOutsideVecs
```

Negative sample loss and gradient-

```
def negSamplingLossAndGradient(centerWordVec, outsideWordIdx, outsideVectors, dataset,
K=10):
    negSampleWordIndices = getNegativeSamples(outsideWordIdx, dataset, K)
    indices = [outsideWordIdx] + negSampleWordIndices
    # zero gradients and losses
    gradCenterVec   = np.zeros(centerWordVec.shape)
    gradOutsideVecs = np.zeros(outsideVectors.shape)
    loss = 0.0
    u_o = outsideVectors[outsideWordIdx]
    z = sigmoid(np.dot(u_o,centerWordVec))
    loss -= np.log(z)
    gradCenterVec += u_o*(z-1)
    gradOutsideVecs[outsideWordIdx] = centerWordVec*(z-1)
    return loss, gradCenterVec, gradOutsideVecs
```

Skipgram-

```
def skipgram(currentCenterWord, windowSize, outsideWords, word2Ind, centerWordVectors,
outsideVectors, dataset, word2vecLossAndGradient=naiveSoftmaxLossAndGradient):
    loss = 0.0
    gradCenterVecs = np.zeros(centerWordVectors.shape)
    gradOutsideVectors = np.zeros(outsideVectors.shape)
    center_id = word2Ind[currentCenterWord]
    centerWordVec = centerWordVectors[center_id]
    for word in outsideWords:
        outside_id = word2Ind[word]
        loss_mini, gradCenter_mini, gradOutside_mini=
        word2vecLossAndGradient(centerWordVec=centerWordVec,
        outsideWordIdx=outside_id, outsideVectors=outsideVectors, dataset=dataset)
        loss += loss_mini
        gradCenterVecs[center_id] += gradCenter_mini
        gradOutsideVectors += gradOutside_mini
    return loss, gradCenterVecs, gradOutsideVectors
```