Dropout前向传播

def dropout\_forward(x, dropout\_param):

    """

    Performs the forward pass for (inverted) dropout.

    Inputs:

    - x: Input data, of any shape

    - dropout\_param: A dictionary with the following keys:

      - p: Dropout parameter. We keep each neuron output with probability p.

      - mode: 'test' or 'train'. If the mode is train, then perform dropout;

        if the mode is test, then just return the input.

      - seed: Seed for the random number generator. Passing seed makes this

        function deterministic, which is needed for gradient checking but not

        in real networks.

    Outputs:

    - out: Array of the same shape as x.

    - cache: tuple (dropout\_param, mask). In training mode, mask is the dropout

      mask that was used to multiply the input; in test mode, mask is None.

    NOTE: Please implement \*\*inverted\*\* dropout, not the vanilla version of dropout.

    See http://cs231n.github.io/neural-networks-2/#reg for more details.

    NOTE 2: Keep in mind that p is the probability of \*\*keep\*\* a neuron

    output; this might be contrary to some sources, where it is referred to

    as the probability of dropping a neuron output.

    """

    if mode == "train":

        #######################################################################

        # TODO: Implement training phase forward pass for inverted dropout.   #

        # Store the dropout mask in the mask variable.                        #

        #######################################################################

        # \*\*\*\*\*START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)\*\*\*\*\*

        mask=(np.random.rand(\*x.shape)<p)/p

        out=mask\*x

        # \*\*\*\*\*END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)\*\*\*\*\*

        #######################################################################

        #                           END OF YOUR CODE                          #

        #######################################################################

    elif mode == "test":

        #######################################################################

        # TODO: Implement the test phase forward pass for inverted dropout.   #

        #######################################################################

        # \*\*\*\*\*START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)\*\*\*\*\*

        out=x

        # \*\*\*\*\*END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)\*\*\*\*\*

        #######################################################################

        #                            END OF YOUR CODE                         #

        #######################################################################

Dropout反向传播

def dropout\_backward(dout, cache):

    """

    Perform the backward pass for (inverted) dropout.

    Inputs:

    - dout: Upstream derivatives, of any shape

    - cache: (dropout\_param, mask) from dropout\_forward.

    """

    if mode == "train":

        #######################################################################

        # TODO: Implement training phase backward pass for inverted dropout   #

        #######################################################################

        # \*\*\*\*\*START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)\*\*\*\*\*

        dx=dout\*mask

        # \*\*\*\*\*END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)\*\*\*\*\*

        #######################################################################

        #                          END OF YOUR CODE                           #

        #######################################################################