```
      self.W -= grad * learning_rate

      # ========================================================== #
      # END YOUR CODE HERE
      # ========================================================== #

      if verbose and it % 100 == 0:
        print('iteration {} / {}: loss {}'.format(it, num_iters, loss))

    return loss_history

  def predict(self, X):
    """
    Inputs:
    - X: N x D array of training data. Each row is a D-dimensional point.

    Returns:
    - y_pred: Predicted labels for the data in X. y_pred is a 1-dimensional
      array of length N, and each element is an integer giving the predicted
      class.
    """
    y_pred = np.zeros(X.shape[1])
    # ========================================================== #
    # YOUR CODE HERE:
    #   Predict the labels given the training data.
    # ========================================================== #

    y_pred = np.argmax(X.dot(self.W.transpose()), axis = 1) #we want the maximum value for each row

    # ========================================================== #
    # END YOUR CODE HERE
    # ========================================================== #

    return y_pred

```