```
def backpropa(self, X, answers):
        a = X
        acts = [a]
        zs = []
  # Forward
         for w, b in zip(self.weights, self.biases):
                z = w.dot(a) + b
                a = self.activation_function(z)
                acts.append(a)
                 zs.append(z)
# Backpropa
           net_answers = acts[-1]
           delta = J_deriv (net_answers, answers) * self.act _deriv(zs[-1])
           deltas = [delta]
           for L in range (2, self.nb_layers):
                 delta = self.weights[-L+1].T.dot(delta) * self.act _deriv(zs[-L])
                 deltas = [delta] + deltas

# Gradients
        dJdb = [delta.sum(axis=1) for delta in deltas]
        dJdw = [delta.dot(a) for delta, a in zip(deltas, acts[:-1])]
return dJdb, dJdw
```