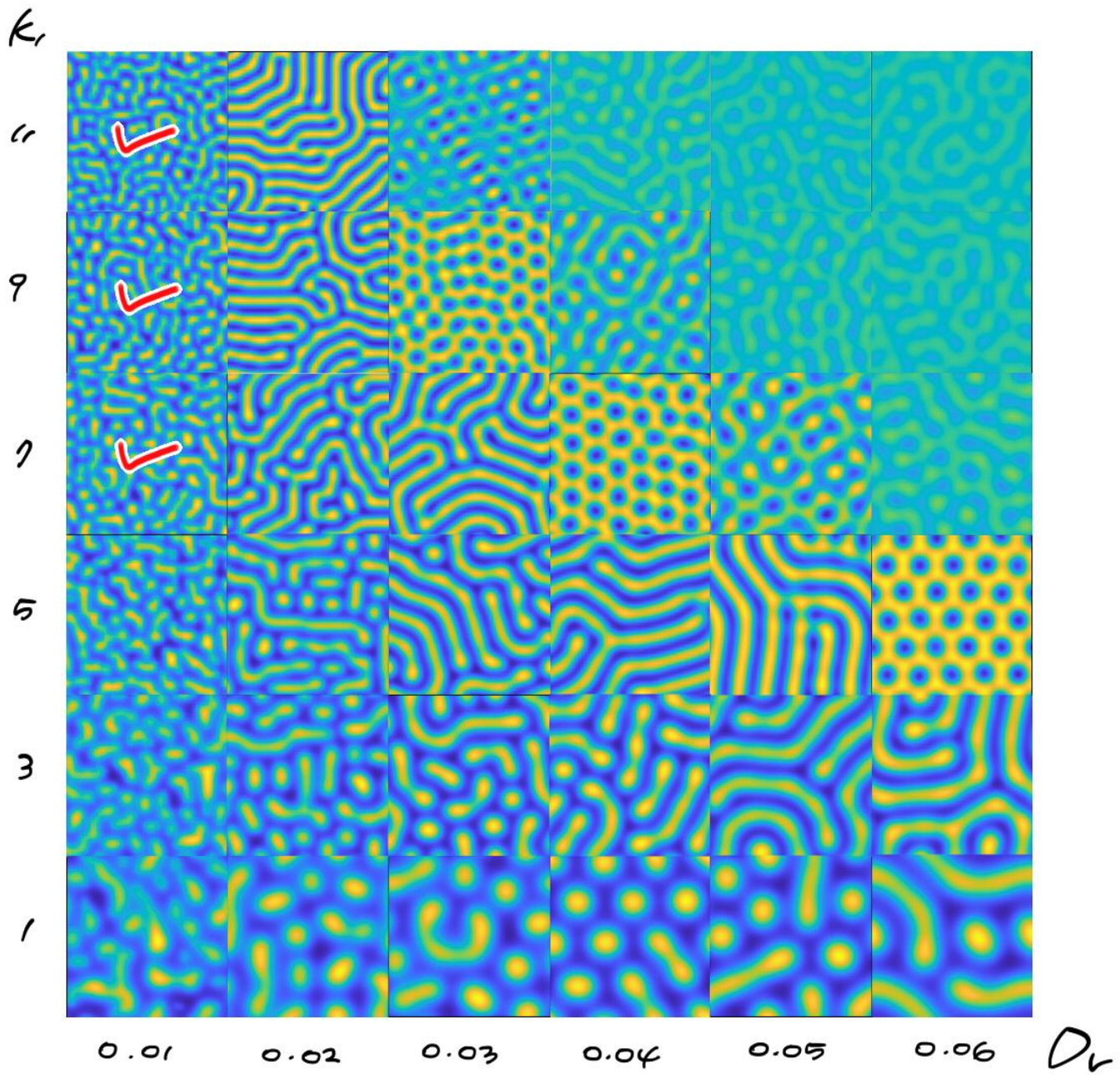
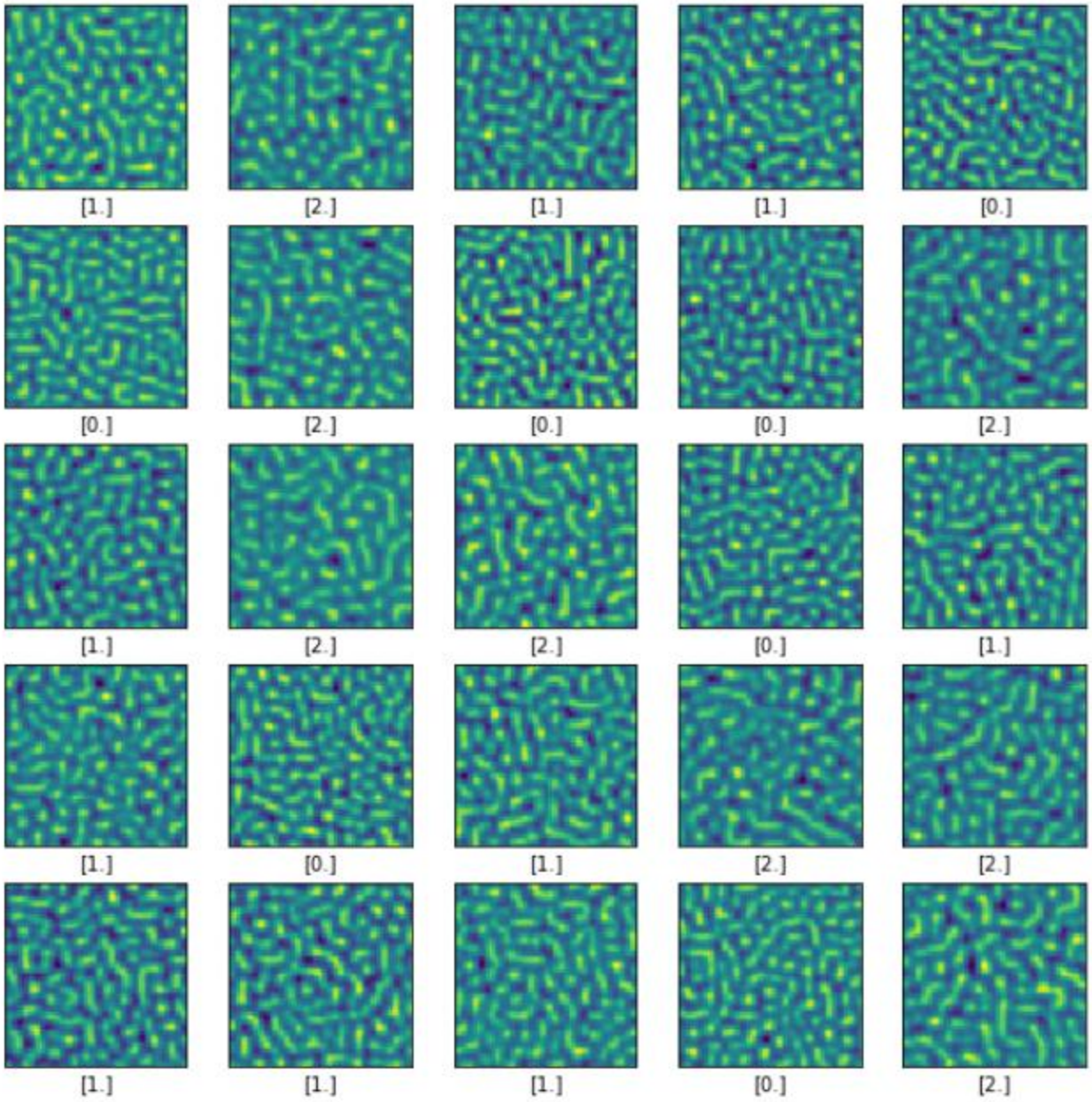

k_r
11
9
7
5
3
1
0.01
0.02
0.03
0.04
0.05
0.06
D_r


[1.] [2.] [1.] [1.] [0.]
[0.] [2.] [0.] [0.] [2.]
[1.] [2.] [2.] [0.] [1.]
[1.] [0.] [1.] [2.] [2.]
[1.] [1.] [1.] [0.] [2.]

$$X \rightarrow Z = w \cdot \|\nabla x\|^2 + b$$

$$\rightarrow A = softmax(Z) \rightarrow \hat{y}$$

$$(u(i+1,j) - u(i-1,j))^2$$

$$+ (u(i,j+1) - u(i,j-1))^2 \approx \|\nabla_d u(i,j)\|^2$$

```
def gradient_nonv(X):
    g_X = np.zeros((X.shape[0], X.shape[1]))
    for i in range(1,X.shape[0]-1):
        for j in range(1,X.shape[1]-1):
            g_X[i,j] = (X[i+1,j] - X[i-1,j])**2 + (X[i,j+1] - X[i,j-1])**2
    g_X[0,:] = g_X[X.shape[0]-2,:]
    g_X[X.shape[0]-1,:] = g_X[1,:]
    g_X[:,1] = g_X[:,X.shape[1]-2]
    g_X[:,X.shape[1]-1] = g_X[:,1]
    return g_X
```

```
def gradient_vec(X):
    g_X_r = np.gradient(X, axis = 1)
    g_X_c = np.gradient(X, axis = 0)
    g_X = g_X_r**2 + g_X_c**2
    return g_X
```

```
# non_numpy
tic = time.process_time()

Z1 = np.dot(W, gradient_nonv(x_test))+ b

toc = time.process_time()
print ("\n ----- Computation time = " + str(1000*(toc - tic)) + "ms")

# numpy
tic = time.process_time()

Z2 = np.dot(W, gradient_vec(x_test))+ b

toc = time.process_time()
print ("\n ----- Computation time = " + str(1000*(toc - tic)) + "ms")
```

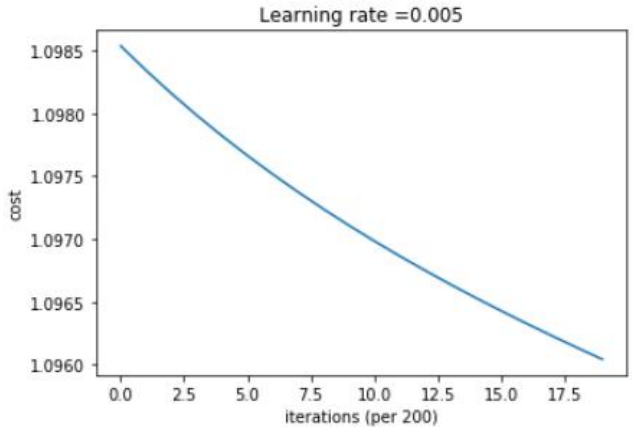
```
----- Computation time = 2578.125ms

----- Computation time = 187.5ms
```

- 2000 iteration

gradient descent

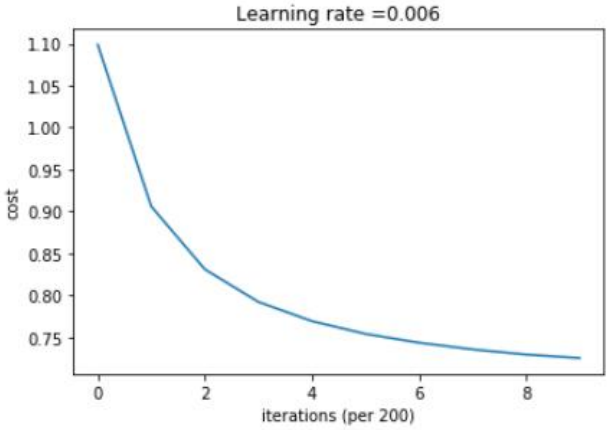
```
Cost after iteration 0: 1.098536
Cost after iteration 200: 1.098158
Cost after iteration 400: 1.097820
Cost after iteration 600: 1.097515
Cost after iteration 800: 1.097238
Cost after iteration 1000: 1.096985
Cost after iteration 1200: 1.096751
Cost after iteration 1400: 1.096533
Cost after iteration 1600: 1.096329
Cost after iteration 1800: 1.096136
```



train accuracy : 0.3341666666666667
test accuracy : 0.33

ADAM

```
Cost after iteration 0: 1.098627
Cost after iteration 200: 0.906004
Cost after iteration 400: 0.831171
Cost after iteration 600: 0.792285
Cost after iteration 800: 0.769404
Cost after iteration 1000: 0.754191
Cost after iteration 1200: 0.743681
Cost after iteration 1400: 0.735736
Cost after iteration 1600: 0.729602
Cost after iteration 1800: 0.725414
```

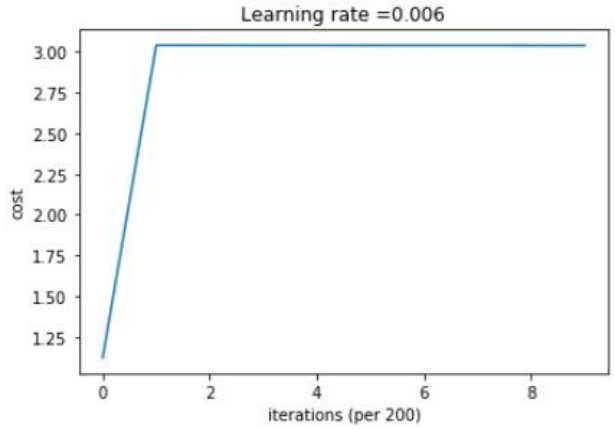


train accuracy : 0.86
test accuracy : 0.34

- 2000 iteration

non-gradient

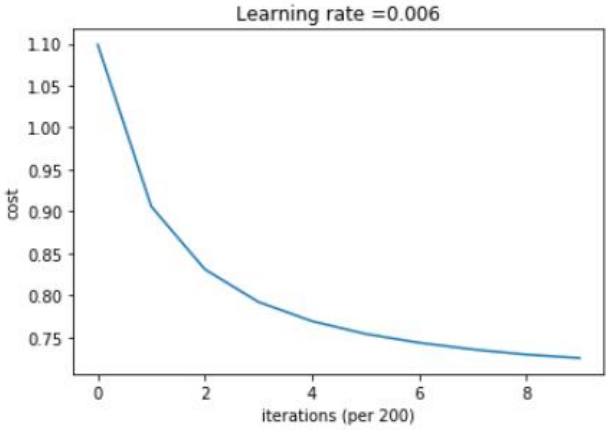
```
Cost after iteration 0: 1.123351
Cost after iteration 200: 3.039338
Cost after iteration 400: 3.039131
Cost after iteration 600: 3.038925
Cost after iteration 800: 3.038720
Cost after iteration 1000: 3.038515
Cost after iteration 1200: 3.038310
Cost after iteration 1400: 3.038107
Cost after iteration 1600: 3.037904
Cost after iteration 1800: 3.037701
```



train accuracy : 0.335
test accuracy : 0.32666666666666666

gradient

```
Cost after iteration 0: 1.098627
Cost after iteration 200: 0.906004
Cost after iteration 400: 0.831171
Cost after iteration 600: 0.792285
Cost after iteration 800: 0.769404
Cost after iteration 1000: 0.754191
Cost after iteration 1200: 0.743681
Cost after iteration 1400: 0.735736
Cost after iteration 1600: 0.729602
Cost after iteration 1800: 0.725414
```



train accuracy : 0.86
test accuracy : 0.34