exc_01_solution

April 26, 2018

0.1 Exercise 01

1 Random Forests

In [2]:
```
%matplotlib inline
import numpy as np
import h5py as h5
import matplotlib.pyplot as plt
import scipy.ndimage
import skimage.color
import skimage.feature
import sklearn.ensemble
from matplotlib.colors import LinearSegmentedColormap
import ipywidgets as widgets

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets
```

1.0.1 1.1 Load the data

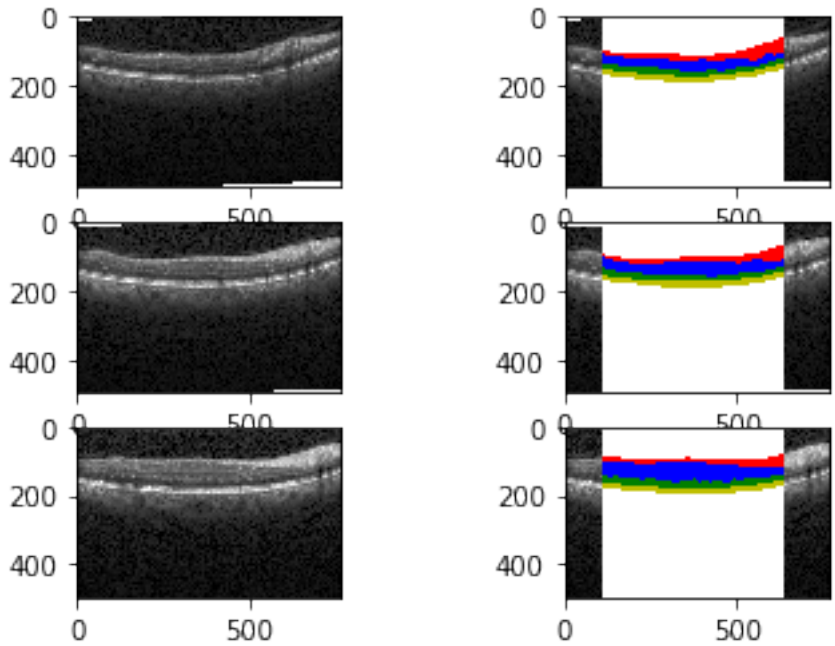
In [3]:
```
images_02 = []
labels_02 = []

images_10 = []
labels_10 = []

f_im_02 = h5.File('images_subject02.h5','r')
f_lb_02 = h5.File('labels_subject02.h5','r')

f_im_10 = h5.File('images_subject10.h5','r')
f_lb_10 = h5.File('labels_subject10.h5','r')

for key in f_im_02:
    images_02.append(f_im_02[key][...])
for key in f_lb_02:
    labels_02.append(f_lb_02[key][...])
for key in f_im_10:
    images_10.append(f_im_10[key][...])
for key in f_lb_10:
    labels_10.append(f_lb_10[key][...])

f_im_02.close()
f_lb_02.close()
```

```
        f_im_10.close()
        f_lb_10.close()
```

```
In [4]: train_set = [images_02[0],images_02[1],images_02[2]]
        train_label = [labels_02[0], labels_02[1],labels_02[2]]
        test_set = images_10[0]
        test_label = labels_10[0]
```

Visualize the training set

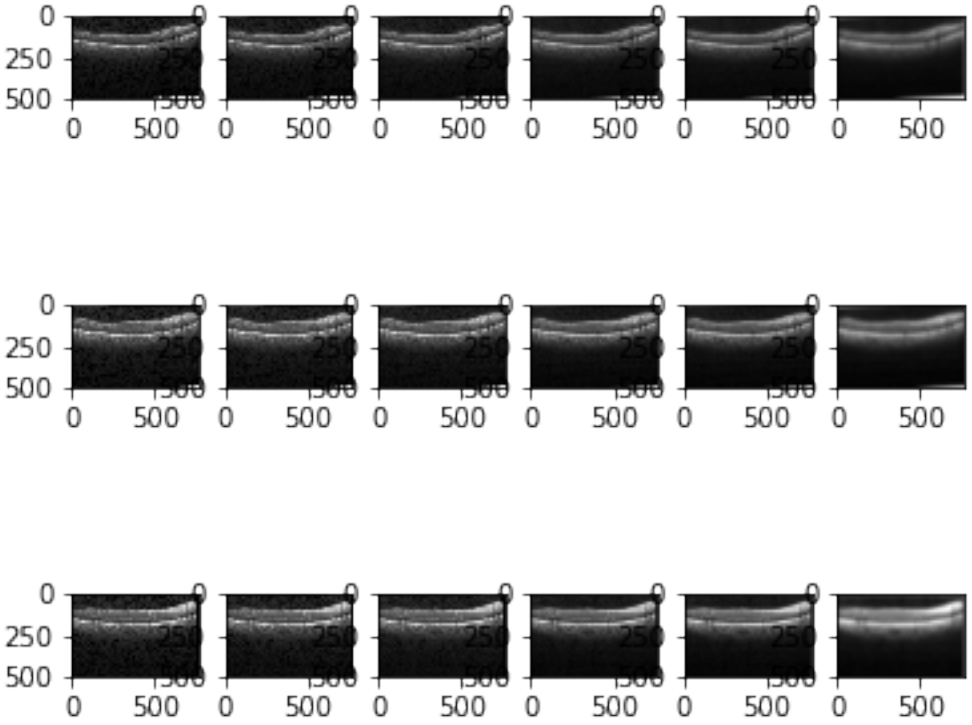
```
In [30]: colors = ['w','r','b','g','y']
         cmap_name = 'my_list'
         cm = LinearSegmentedColormap.from_list(
                 cmap_name, colors, N=5)

         fig,ax = plt.subplots(3,2)
         for i in range(3):
             masked_label = np.ma.array(train_label[i], mask=train_label[i]<0)

             ax[i,0].imshow(train_set[i],cmap=plt.cm.gray)
             ax[i,1].imshow(train_set[i],cmap=plt.cm.gray)
             ax[i,1].imshow(masked_label,cmap=cm)
```



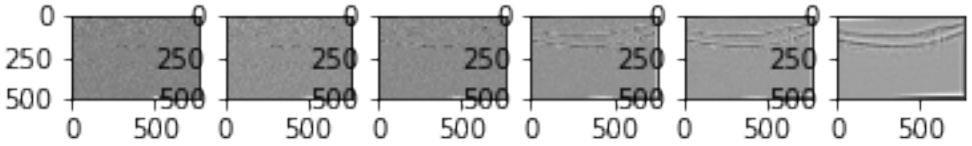
1.0.2 1.2 Compute Featrues

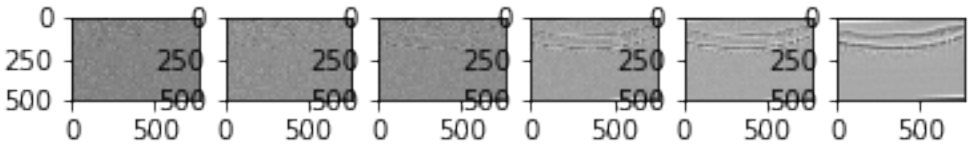
```
In [31]: all_filters = {}
         for t, train in enumerate(train_set):
             all_filters['train_'+str(t)] = []
```

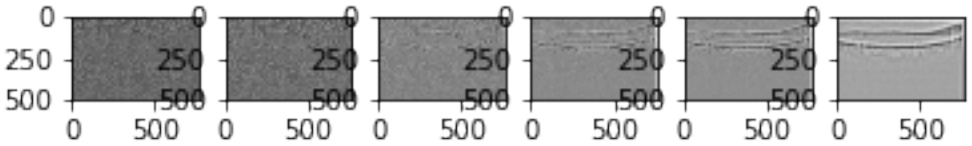
```
for t,train in enumerate(train_set):

    f = plt.figure()
    sigmas = [0.7, 1, 1.6, 3.5, 5, 10]
    for i, sigma in enumerate(sigmas):
        res = scipy.ndimage.gaussian_filter(train, sigma=sigma)
        ax = f.add_subplot(1, len(sigmas), i+1)
        ax.imshow(res, cmap='gray')
        all_filters['train_'+str(t)].append(res)
```



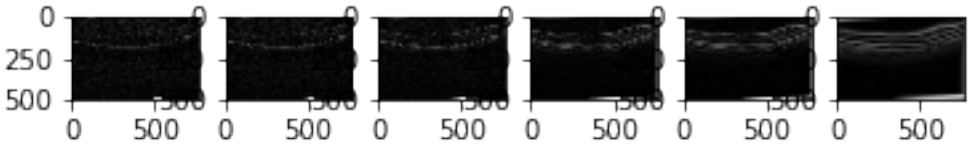
In [32]:
```
for t,train in enumerate(train_set):

    f = plt.figure()
    for i, sigma in enumerate(sigmas):
        res = scipy.ndimage.gaussian_laplace(train, sigma=sigma)
        ax = f.add_subplot(1, len(sigmas), i+1)
        ax.imshow(res, cmap='gray')
        all_filters['train_'+str(t)].append(res)
```

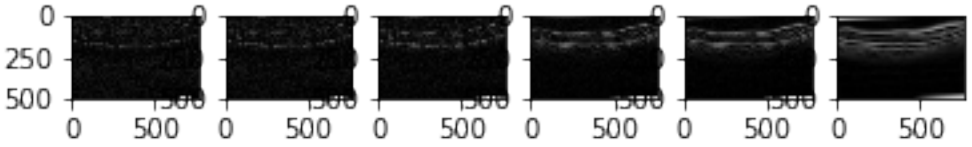


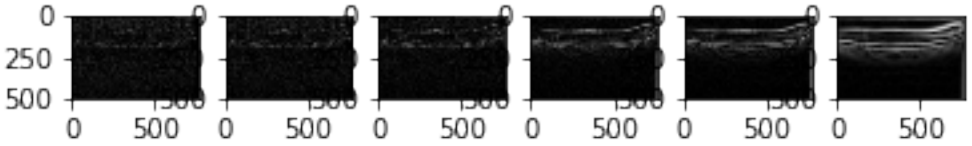




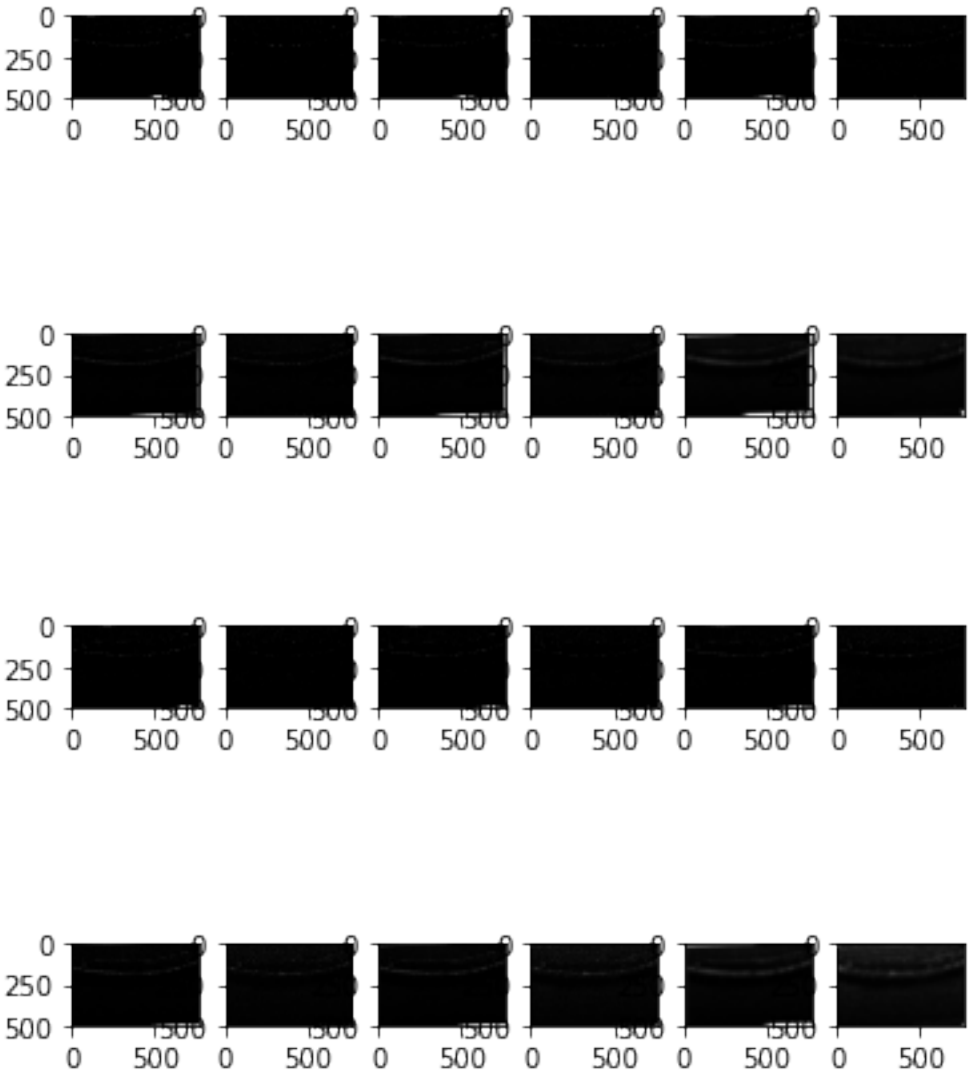
In [33]:
```
for t,train in enumerate(train_set):

    f = plt.figure()
    for i, sigma in enumerate(sigmas):
        res = scipy.ndimage.gaussian_gradient_magnitude(train, sigma=sigma)
        ax = f.add_subplot(1, len(sigmas), i+1)
        ax.imshow(res, cmap='gray')
        all_filters['train_'+str(t)].append(res)
```

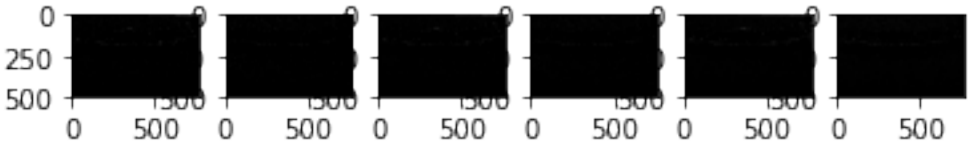


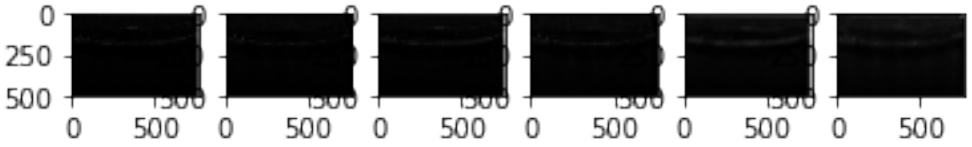




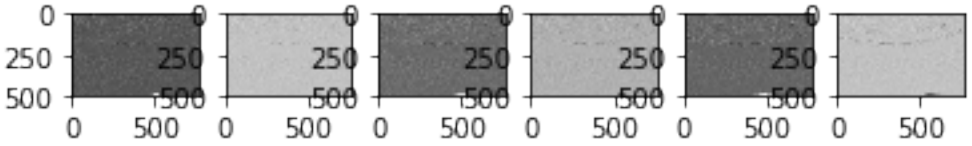
```
In [34]: for t,train in enumerate(train_set):

             f = plt.figure()
             for i, sigma in enumerate(sigmas):
                 a,b = skimage.feature.structure_tensor_eigvals(*skimage.feature.structure_tensor(train
                 ax0 = f.add_subplot(2, len(sigmas), 2*i+1)
                 ax1 = f.add_subplot(2, len(sigmas), 2*i+2)
                 ax0.imshow(a, cmap='gray')
                 ax1.imshow(b, cmap='gray')
                 all_filters['train_'+str(t)].append(a)
                 all_filters['train_'+str(t)].append(b)
```

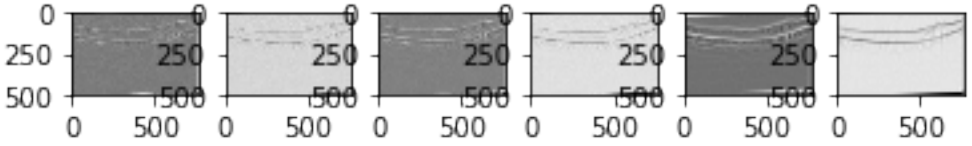


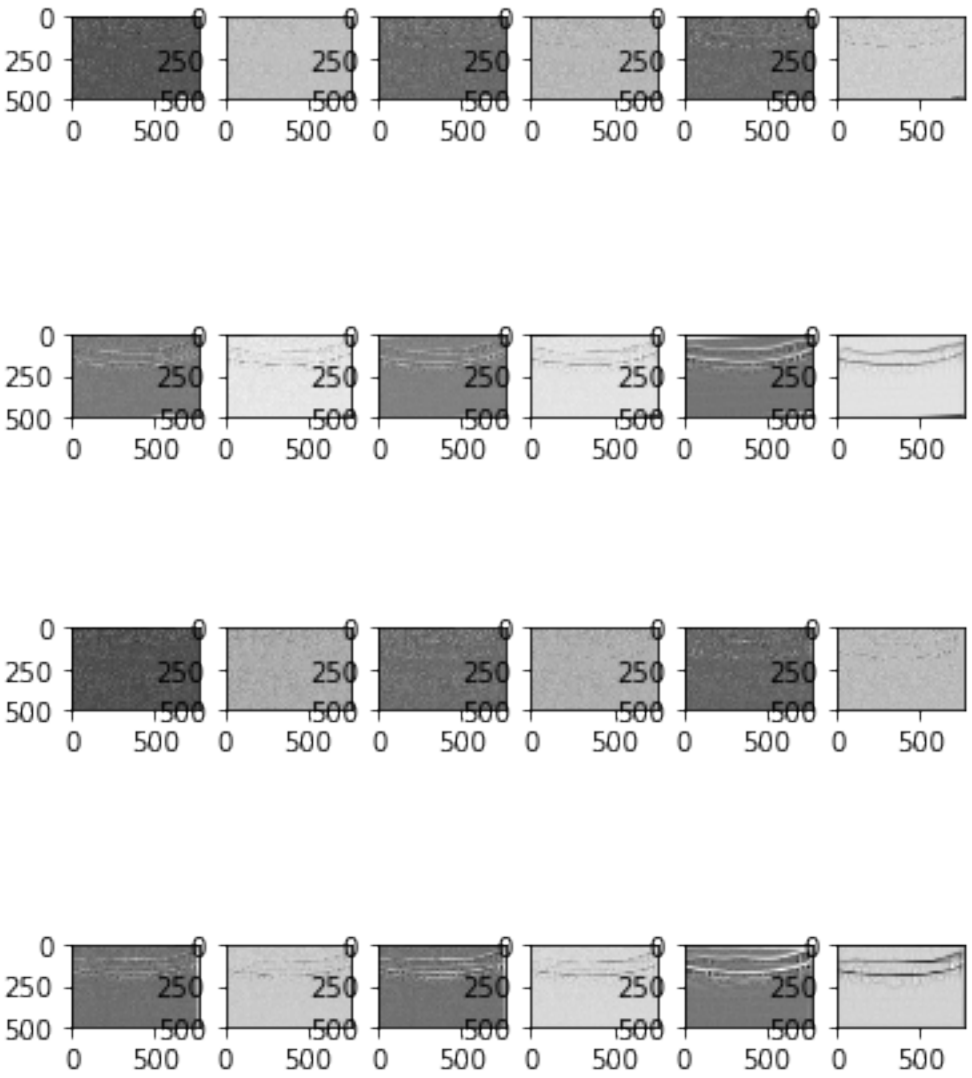




In [35]:
```
for t,train in enumerate(train_set):

    f = plt.figure()
    for i, sigma in enumerate(sigmas):
        a,b = skimage.feature.hessian_matrix_eigvals(*skimage.feature.hessian_matrix(train, si
        ax0 = f.add_subplot(2, len(sigmas), 2*i+1)
        ax1 = f.add_subplot(2, len(sigmas), 2*i+2)
        ax0.imshow(a, cmap='gray')
        ax1.imshow(b, cmap='gray')
        all_filters['train_'+str(t)].append(a)
        all_filters['train_'+str(t)].append(b)
```








```
In [11]: filter_arrays = []
         for t in range(len(all_filters)):
             filters_t = all_filters['train_'+str(t)]
             filters_t = np.array(filters_t)
             filter_arrays.append(filters_t)
         filter_arrays = np.stack(filter_arrays,axis=0)
         print(filter_arrays.shape)
```

```
(3, 42, 496, 768)
```

```
In [12]: train_X_all = []
         train_Y_all = []

         for t in range(filter_arrays.shape[0]):
             filters = filter_arrays[t]
             label = train_label[t]
```

```
            print(filters.shape)
            mask = np.logical_not(label < 0)
            train_X = filters[:,mask].T
            train_Y = label[np.logical_not(label<0)]

            print(train_X.shape)
            print(train_Y.shape)

            train_X_all.append(train_X)
            train_Y_all.append(train_Y)

        train_X = np.concatenate(train_X_all,axis=0)
        train_Y = np.concatenate(train_Y_all)

        print(train_X.shape)
        print(train_Y.shape)
```

```
(42, 496, 768)
(259904, 42)
(259904,)
(42, 496, 768)
(259904, 42)
(259904,)
(42, 496, 768)
(259904, 42)
(259904,)
(779712, 42)
(779712,)
```

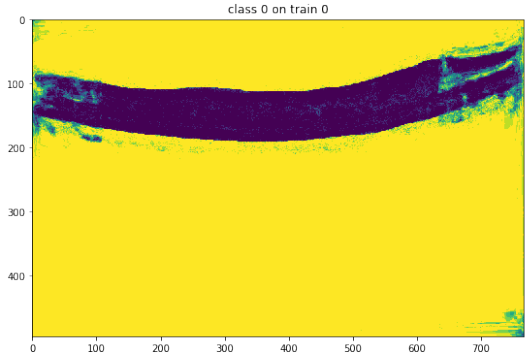
1.0.3 1.3 Random Forest

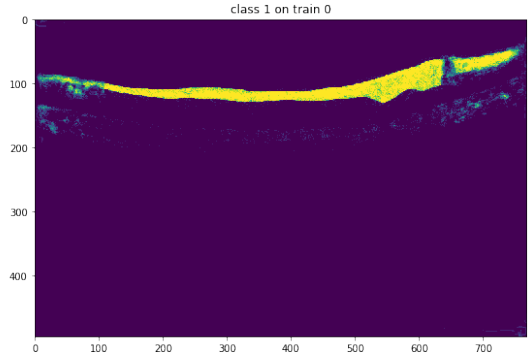
In [13]:
```
rf = sklearn.ensemble.RandomForestClassifier(n_estimators=10)
rf.fit(train_X, train_Y)
```

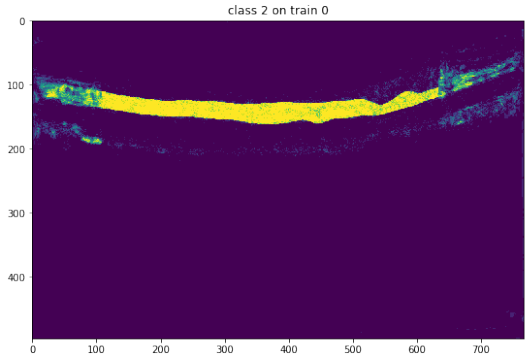
Out[13]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

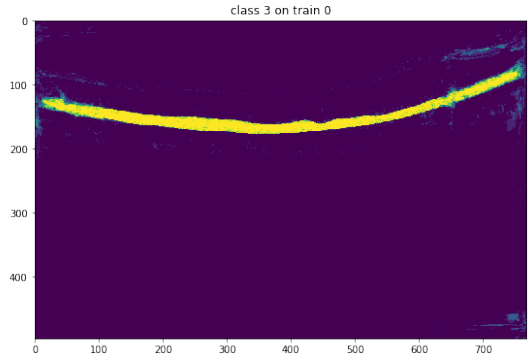
In [36]:
```
def f(t):
    filters = filter_arrays[t]
    train = train_set[t]

    pred0 = rf.predict_proba(filters.reshape((filters.shape[0], -1)).T)
    pred0 = pred0.reshape(train.shape+(-1,))

    #with h5.File('predictions.h5','a') as save:
    #    save.create_dataset('train_'+str(t),data=pred0)

    f = plt.figure()
    for i_class in range(pred0.shape[-1]):
        ax = f.add_subplot(4, 2, i_class+1)
        ax.imshow(pred0[:,:,i_class])
```

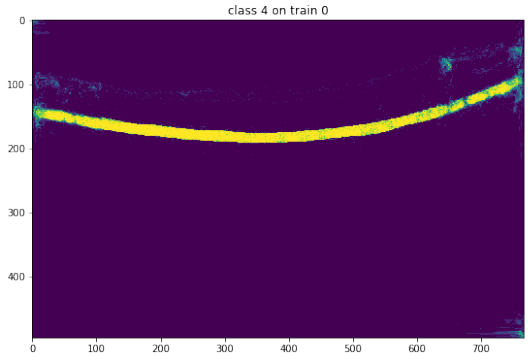
```
            ax.set_title("class "+str(i_class)+" on train "+str(t))
        f.set_size_inches(20,40)

        pred_full = np.argmax(pred0, axis=2)
        f = plt.figure()
        ax = f.add_subplot(111)
        ax.imshow(pred_full)
        ax.set_title("label image, train "+str(t))
        f.set_size_inches(20,40)

interact(f,
         t=widgets.IntSlider(min=0,max=filter_arrays.shape[0],step=1,value=0));
```

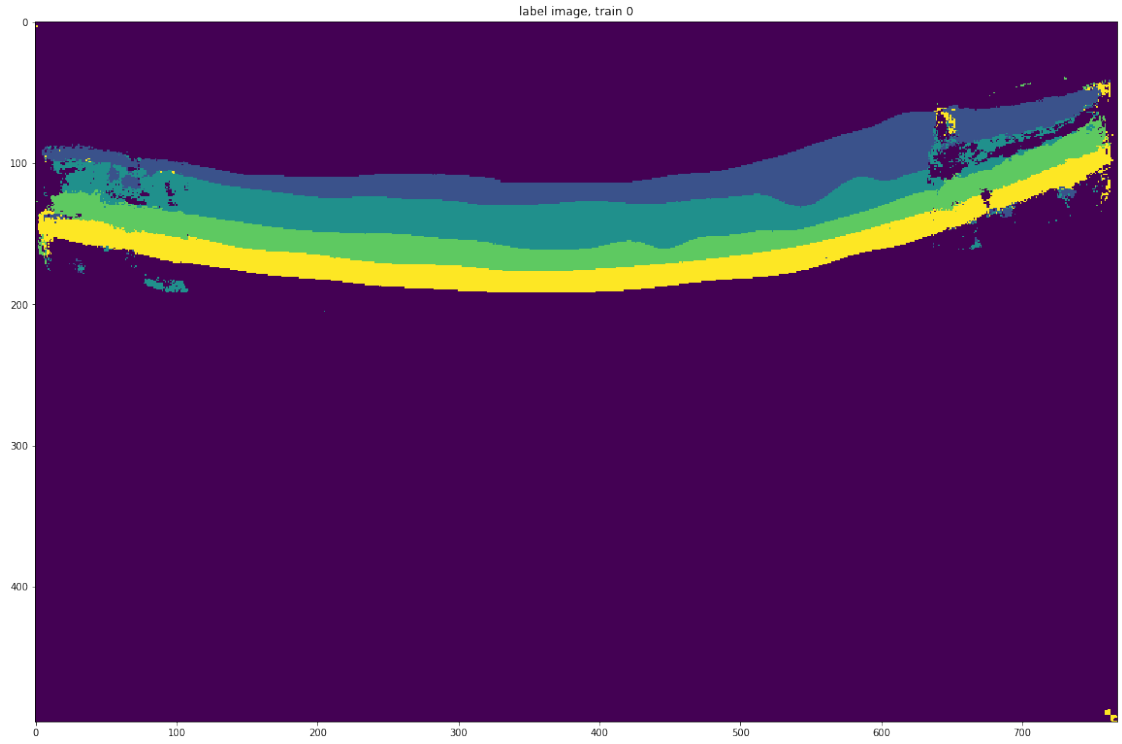

class 0 on train 0
0
100
200
300
400
0
100
200
300
400
500
600
700


class 1 on train 0
0
100
200
300
400
0
100
200
300
400
500
600
700


class 2 on train 0
0
100
200
300
400
0
100
200
300
400
500
600
700


class 3 on train 0
0
100
200
300
400
0
100
200
300
400
500
600
700

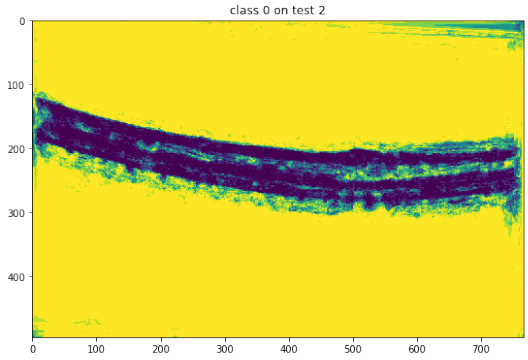

class 4 on train 0
0
100
200
300
400
0
100
200
300
400
500
600
700

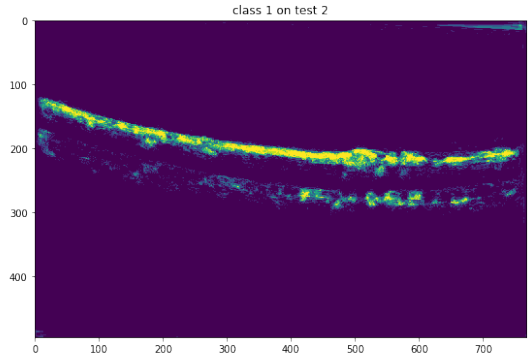


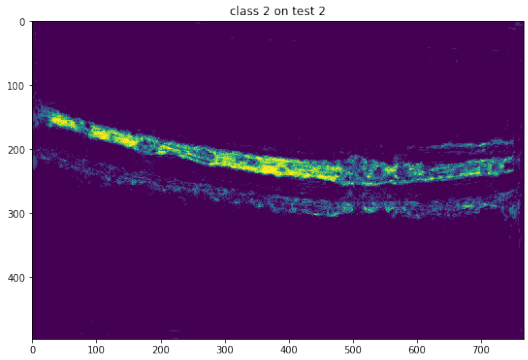
Predict labels on test set

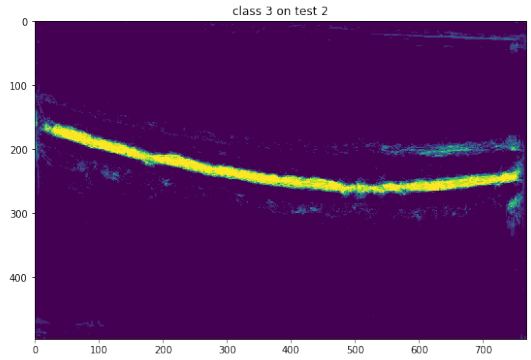
```
In [15]: test_filters = []
         for i, sigma in enumerate(sigmas):
             res = scipy.ndimage.gaussian_filter(test_set, sigma=sigma)
             test_filters.append(res)
         for i, sigma in enumerate(sigmas):
             res = scipy.ndimage.gaussian_laplace(test_set, sigma=sigma)
             test_filters.append(res)
         for i, sigma in enumerate(sigmas):
             res = scipy.ndimage.gaussian_gradient_magnitude(test_set, sigma=sigma)
             test_filters.append(res)
         for i, sigma in enumerate(sigmas):
             a,b = skimage.feature.structure_tensor_eigvals(*skimage.feature.structure_tensor(test_set,
             test_filters.append(a)
             test_filters.append(b)
         for i, sigma in enumerate(sigmas):
             a,b = skimage.feature.hessian_matrix_eigvals(*skimage.feature.hessian_matrix(test_set, sig
             test_filters.append(a)
             test_filters.append(b)

In [16]: test_filters = np.array(test_filters)
         pred = rf.predict_proba(test_filters.reshape((test_filters.shape[0], -1)).T)
         pred = pred.reshape(test_set.shape+(-1,))

         with h5.File('predictions.h5','a') as save:
             if not 'test' in save:
                 save.create_dataset('test',data=pred)
```

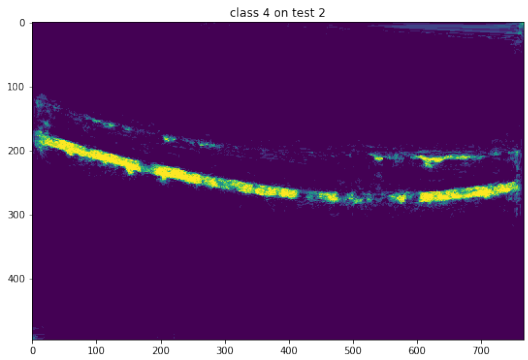
In [37]:
```
f = plt.figure()
for i_class in range(pred.shape[-1]):
    ax = f.add_subplot(4, 2, i_class+1)
    ax.imshow(pred[:,:,i_class])
    ax.set_title("class %i on test %i" %(i_class,t))
f.set_size_inches(20,40)
```



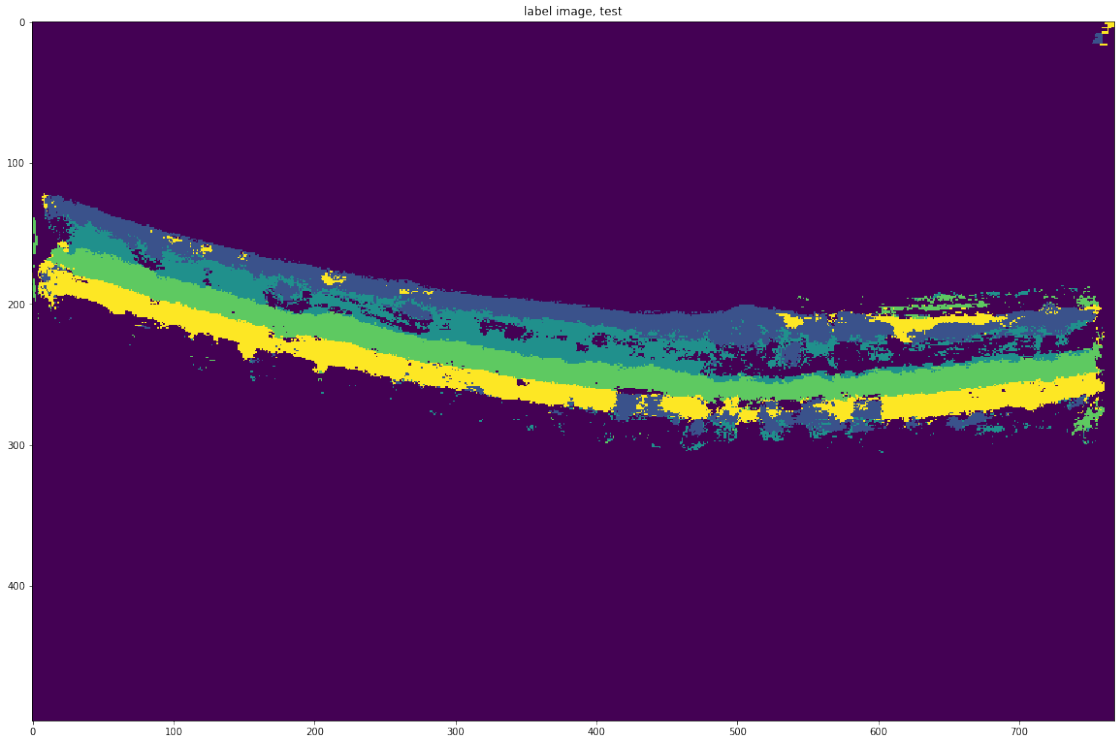




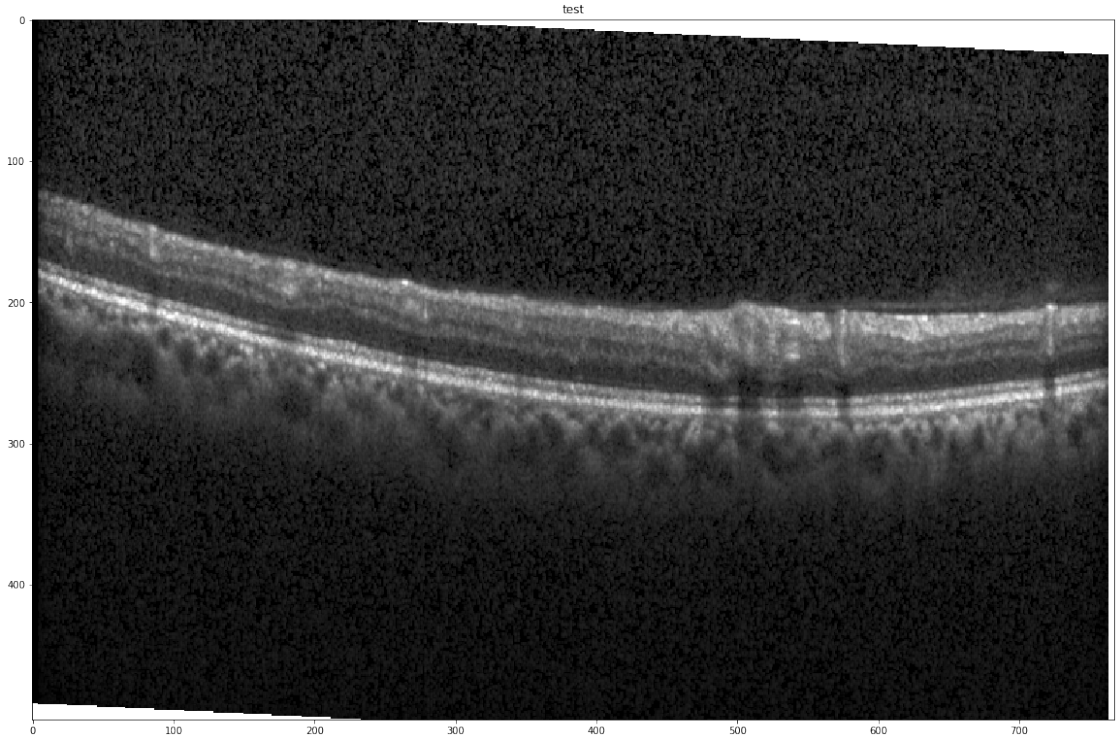




In [38]:
```
pred_full = np.argmax(pred, axis=2)
f = plt.figure()
ax = f.add_subplot(111)
ax.imshow(pred_full)
ax.set_title("label image, test")
f.set_size_inches(20,40)
```



In [39]:
```
f = plt.figure()
ax = f.add_subplot(111)
ax.imshow(test_set,cmap=plt.cm.gray)
ax.set_title("test")
f.set_size_inches(20,40)
```



1.0.4 1.4 Acuracy Score

```
In [20]: pred_comp = pred_full[np.logical_not(test_label < 0)]
         labels_comp = test_label[np.logical_not(test_label < 0)]
         print(pred_comp.shape)
         print(labels_comp.shape)
```

```
(259904,)
(259904,)
```

```
In [28]: for i_class in range(pred.shape[-1]):
             TP = np.sum(np.equal(pred_comp[np.where(pred_comp==i_class)],labels_comp[np.where(pred_comp
             FP = np.sum(np.not_equal(pred_comp[np.where(pred_comp==i_class)],labels_comp[np.where(pred_
             FN = np.sum(np.not_equal(pred_comp[np.where(labels_comp==i_class)],labels_comp[np.where(lab
             P = TP / (TP + FP)
             R = TP / (TP + FN)
             F_1 = 2*P*R/(P+R)
             print("Class %i: \n\tP = %f\n\tR = %f \n\tF1 = %f"%(i_class,np.round_(P,3),np.round_(R,3),n
```

```
Class 0:
        P = 0.980000
        R = 0.986000
        F1 = 0.983000
Class 1:
        P = 0.557000
        R = 0.796000
        F1 = 0.655000
Class 2:
```

```
        P = 0.839000
        R = 0.563000
        F1 = 0.674000
Class 3:
        P = 0.893000
        R = 0.920000
        F1 = 0.906000
Class 4:
        P = 0.778000
        R = 0.728000
        F1 = 0.752000
```

```
In [29]: for i_class in range(pred.shape[-1]):
             TP = np.sum(np.equal(pred_comp[np.where(pred_comp==i_class)],labels_comp[np.where(pred_comp
             FP = np.sum(np.not_equal(pred_comp[np.where(pred_comp==i_class)],labels_comp[np.where(pred_
             FN = np.sum(np.not_equal(pred_comp[np.where(labels_comp==i_class)],labels_comp[np.where(lab
             P = TP / (TP + FP)
             R = TP / (TP + FN)
             F_1 = P*R/(P+R)
             print("Class %i: \n\tP = %f\n\tR = %f \n\tF1 = %f"%(i_class,np.round_(P,3),np.round_(R,3),n
```

```
Class 0:
        P = 0.980000
        R = 0.986000
        F1 = 0.492000
Class 1:
        P = 0.557000
        R = 0.796000
        F1 = 0.328000
Class 2:
        P = 0.839000
        R = 0.563000
        F1 = 0.337000
Class 3:
        P = 0.893000
        R = 0.920000
        F1 = 0.453000
Class 4:
        P = 0.778000
        R = 0.728000
        F1 = 0.376000
```

```
In [ ]:
```