run_classifier

January 14, 2020

In [1]:
```
from test_face_classifier import test_face_classifier_l
from test_face_classifier import test_face_classifier_rf
from get_testing_data import get_testing_data
from get_hog_images import get_hog_images
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
```

In [2]:
```
# save hog descriptors for testing data as csv
# descriptors, classes, races = get_testing_data(300, 9)
# np.savetxt("hog_descriptors.csv", descriptors, delimiter=",")
# np.savetxt("hog_races.csv", races, delimiter=",")
```

In [7]:
```
# Run classifier over num_trials and return average testing accuracies and false posit
# training with normal nonfaces, random forest
def run_classifier(num_trials, num_training, classifier):
    all_overall = np.zeros([5, num_trials])
    all_white = np.zeros([5, num_trials])
    all_black = np.zeros([5, num_trials])
    all_falsepos = np.zeros([5, num_trials])
    for i in range(num_trials):
        print("trial " + str(i))
        overall_acc = np.zeros(5)
        white_acc = np.zeros(5)
        black_acc = np.zeros(5)
        false_pos = np.zeros(5)
        overall, white, black, falsepos = classifier(num_training, 300, 9, 'train_0',
        overall_acc[0] = overall
        white_acc[0] = white
        black_acc[0] = black
        false_pos[0] = falsepos
        overall, white, black, falsepos = classifier(num_training, 300, 9, 'train_25',
        overall_acc[1] = overall
        white_acc[1] = white
        black_acc[1] = black
        false_pos[1] = falsepos
        overall, white, black, falsepos = classifier(num_training, 300, 9, 'train_50',
        overall_acc[2] = overall
```

```
            white_acc[2] = white
            black_acc[2] = black
            false_pos[2] = falsepos
            overall, white, black, falsepos = classifier(num_training, 300, 9, 'train_75',
            overall_acc[3] = overall
            white_acc[3] = white
            black_acc[3] = black
            false_pos[3] = falsepos
            overall, white, black, falsepos = classifier(num_training, 300, 9, 'train_100'
            overall_acc[4] = overall
            white_acc[4] = white
            black_acc[4] = black
            false_pos[4] = falsepos
            # set datasets
            all_overall[:, i] = overall_acc
            all_white[:, i] = white_acc
            all_black[:, i] = black_acc
            all_falsepos[:, i] = falsepos

        return (all_overall, all_white, all_black, all_falsepos)
```

In [8]:
```
# get plot for average accuracies
def plot_accuracies(all_overall, all_white, all_black):
    x = [0, 25, 50, 75, 100]
    # stddev
    std_o = np.std(all_overall, axis=1)
    std_w = np.std(all_white, axis=1)
    std_b = np.std(all_black, axis=1)
    print(std_w)
    # average
    avg_o = np.average(all_overall, axis=1)
    avg_w = np.average(all_white, axis=1)
    avg_b = np.average(all_black, axis=1)
    # plot
    f, ax = plt.subplots()
    ax.errorbar(x, avg_o, std_o, label='o', capsize=5)
    ax.errorbar(x, avg_b, std_b, label='b', capsize=5)
    ax.errorbar(x, avg_w, std_w, label='w', capsize=5)

    return ax
```

0.1 Logistic

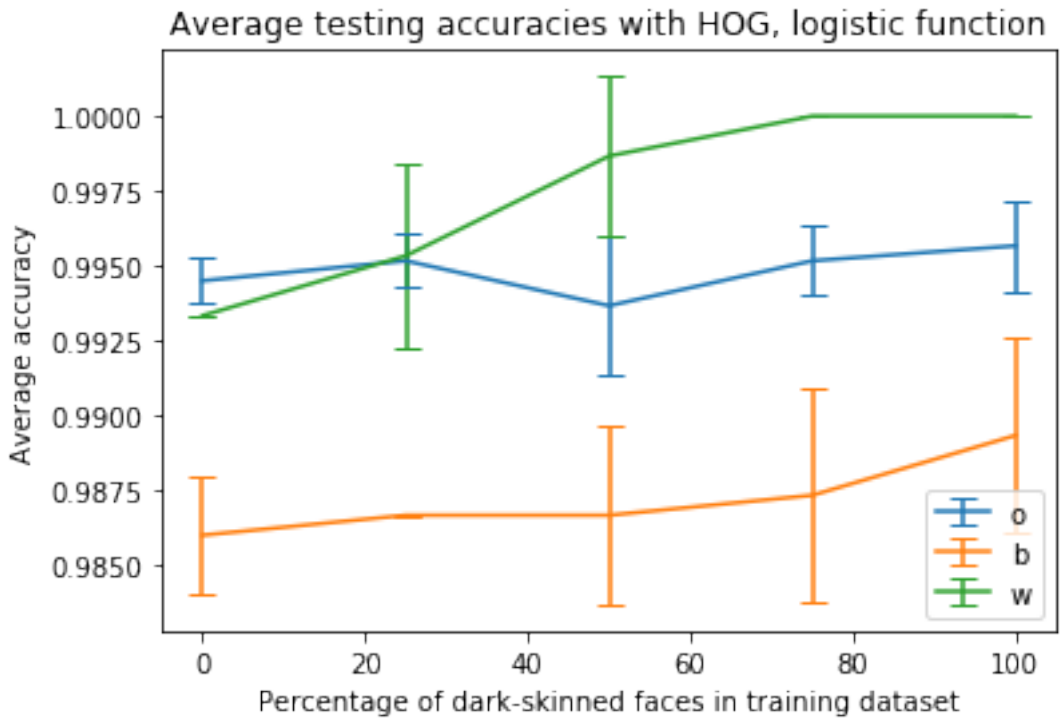
In [12]:
```
# run logistic classifier with 10 trials
all_overall_l, all_white_l, all_black_l, all_falsepos_l = run_classifier(10, 4020, tes
```

```
trial 0
trial 1
```

```
trial 2
trial 3
trial 4
trial 5
trial 6
trial 7
trial 8
trial 9
```

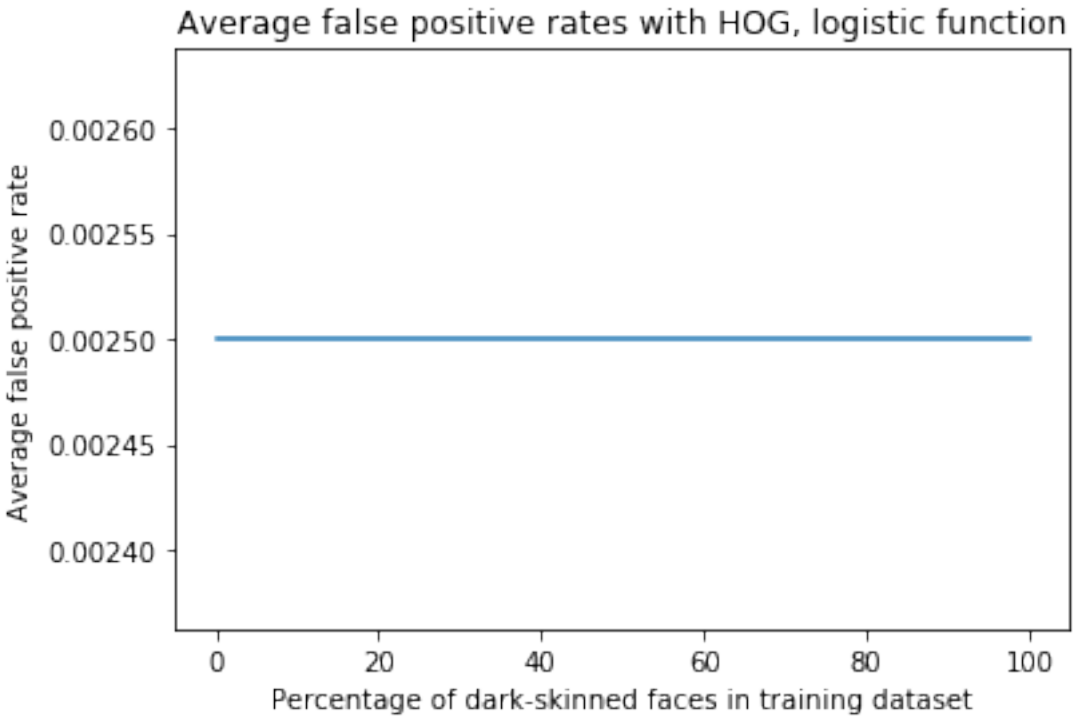
In [13]:
```
plot = plot_accuracies(all_overall_l, all_white_l, all_black_l)
plt.legend(loc="lower right")
plt.title("Average testing accuracies with HOG, logistic function")
plt.xlabel("Percentage of dark-skinned faces in training dataset")
plt.ylabel("Average accuracy")
plt.savefig('hog_avg_logistic.png', dpi=300)
plt.show()
```

```
[1.11022302e-16 3.05505046e-03 2.66666667e-03 0.00000000e+00
 0.00000000e+00]
```



In [41]:
```
# plot average false positive rates
avg_fp_l = np.average(all_falsepos_l, axis=1)
```

```
x = [0, 25, 50, 75, 100]
plt.plot(x, avg_fp_l)
plt.title("Average false positive rates with HOG, logistic function")
plt.xlabel("Percentage of dark-skinned faces in training dataset")
plt.ylabel("Average false positive rate")
plt.savefig('hog_falsepos_logistic.png', dpi=300)
plt.show()
```



In [12]:
```
# # training with harder nonfaces
# overall_acc = np.zeros(5)
# white_acc = np.zeros(5)
# black_acc = np.zeros(5)
# false_pos = np.zeros(5)
# overall, white, black, falsepos = test_face_classifier_l(300, 300, 9, 'train_0', 'h
# overall_acc[0] = overall
# white_acc[0] = white
# black_acc[0] = black
# false_pos[0] = falsepos
# overall, white, black, falsepos = test_face_classifier_l(300, 300, 9, 'train_25', '
# overall_acc[1] = overall
# white_acc[1] = white
# black_acc[1] = black
# false_pos[1] = falsepos
# overall, white, black, falsepos = test_face_classifier_l(300, 300, 9, 'train_50', '
```

```
        # overall_acc[2] = overall
        # white_acc[2] = white
        # black_acc[2] = black
        # false_pos[2] = falsepos
        # overall, white, black, falsepos = test_face_classifier_l(300, 300, 9, 'train_75', '
        # overall_acc[3] = overall
        # white_acc[3] = white
        # black_acc[3] = black
        # false_pos[3] = falsepos
        # overall, white, black, falsepos = test_face_classifier_l(300, 300, 9, 'train_100',
        # overall_acc[4] = overall
        # white_acc[4] = white
        # black_acc[4] = black
        # false_pos[4] = falsepos
```

In [13]:
```
        # # graph
        # x = [0, 25, 50, 75, 100]
        # plt.plot(x, overall_acc, label='overall')
        # plt.plot(x, white_acc, label='light-skinned')
        # plt.plot(x, black_acc, label='dark-skinned')
        # plt.legend(loc="lower right")
        # plt.title("Average accuracies with HOG, logistic function, hard nonfaces")
        # plt.xlabel("Percentage of dark-skinned faces in training dataset")
        # plt.ylabel("Average accuracy")
        # plt.show()
```

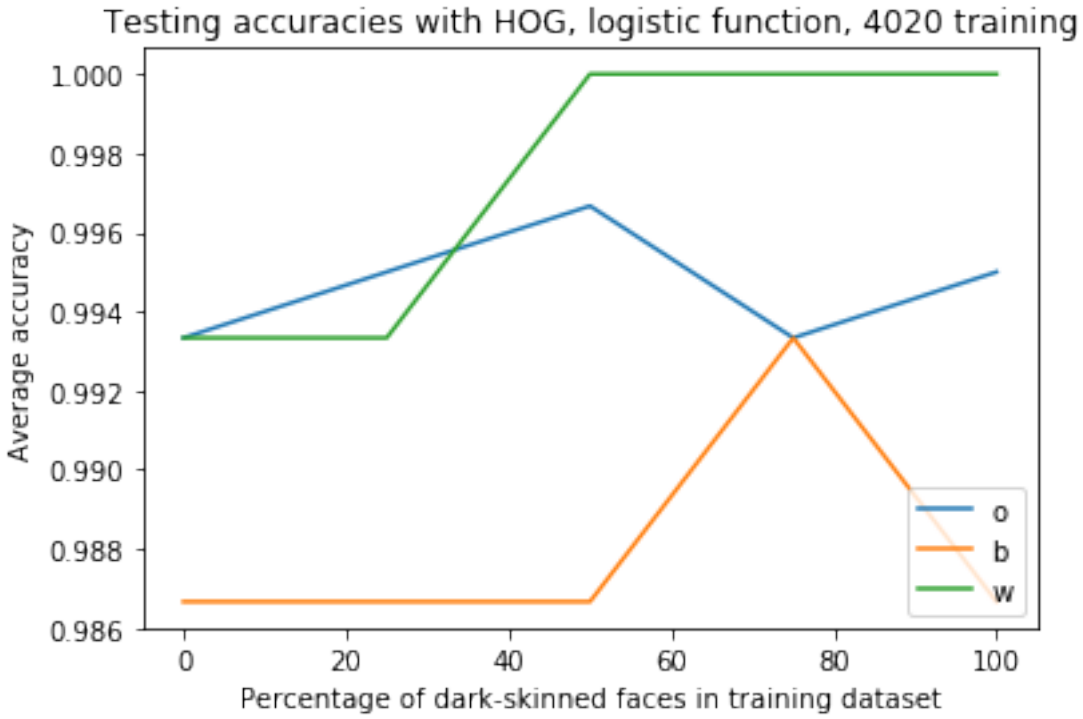
In [14]:
```
        # # plot average false positive rates
        # x = [0, 25, 50, 75, 100]
        # plt.plot(x, avg_falsepos)
        # plt.title("Average false positive rates with HOG, logistic function, hard nonfaces"
        # plt.xlabel("Percentage of dark-skinned faces in training dataset")
        # plt.ylabel("Average false positive rate")
        # plt.show()
```

In [10]:
```
        # train with 4020 examples
        avg_overall_l, avg_white_l, avg_black_l, avg_falsepos_l = run_classifier(1, 4020, tes
```

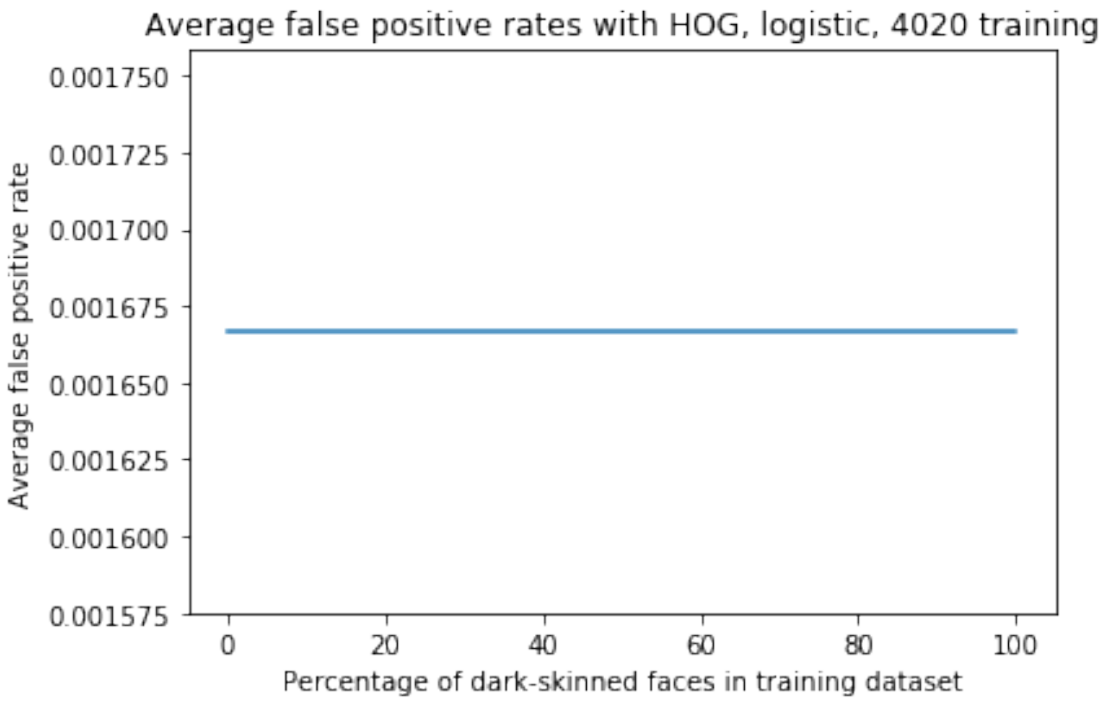
```
trial 0
```

In [21]:
```
        # plot accuracies
        x = [0, 25, 50, 75, 100]
        plt.plot(x, avg_overall_l, label='o')
        plt.plot(x, avg_black_l, label='b')
        plt.plot(x, avg_white_l, label='w')
        plt.legend(loc="lower right")
        plt.title("Testing accuracies with HOG, logistic function, 4020 training")
        plt.xlabel("Percentage of dark-skinned faces in training dataset")
        plt.ylabel("Average accuracy")
```

```
plt.savefig('hog_4020_logistic.png', dpi=300)
plt.show()
```



In [19]:
```
# plot false positive rates
x = [0, 25, 50, 75, 100]
plt.plot(x, avg_falsepos_l)
plt.title("Average false positive rates with HOG, logistic, 4020 training")
plt.xlabel("Percentage of dark-skinned faces in training dataset")
plt.ylabel("Average false positive rate")
plt.savefig('hog_4020_falsepos_logistic.png', dpi=300)
plt.show()
```



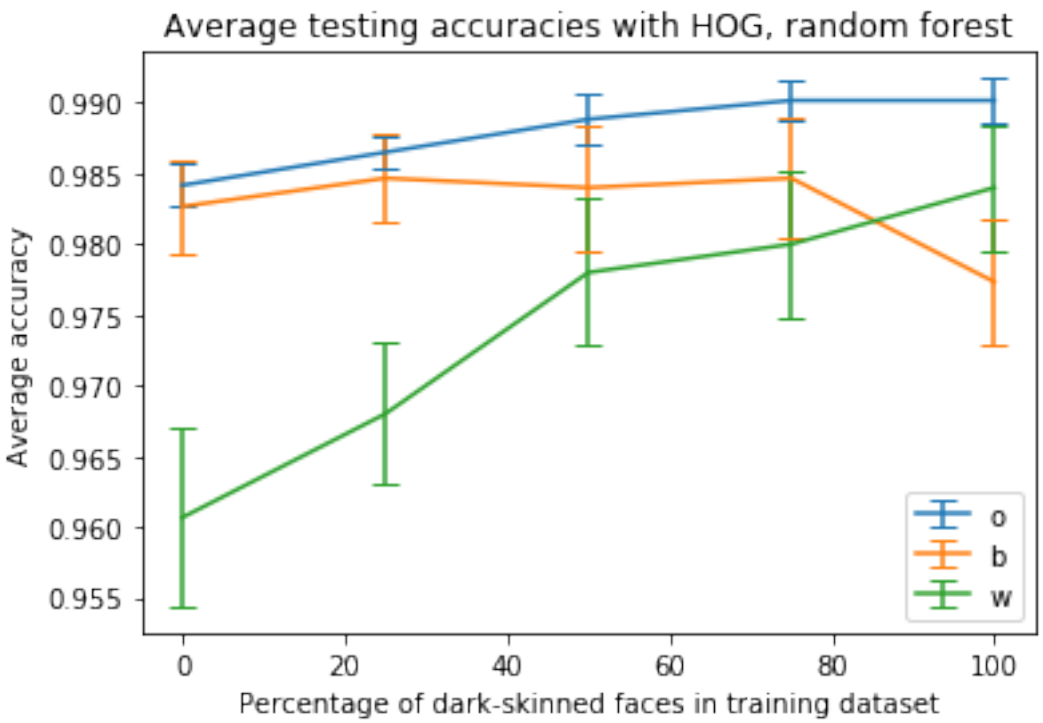
0.2 Random forest

```
In [9]: # run random forest classifier with 10 trials
        all_overall_rf, all_white_rf, all_black_rf, all_falsepos_rf = run_classifier(10, 4020,
```

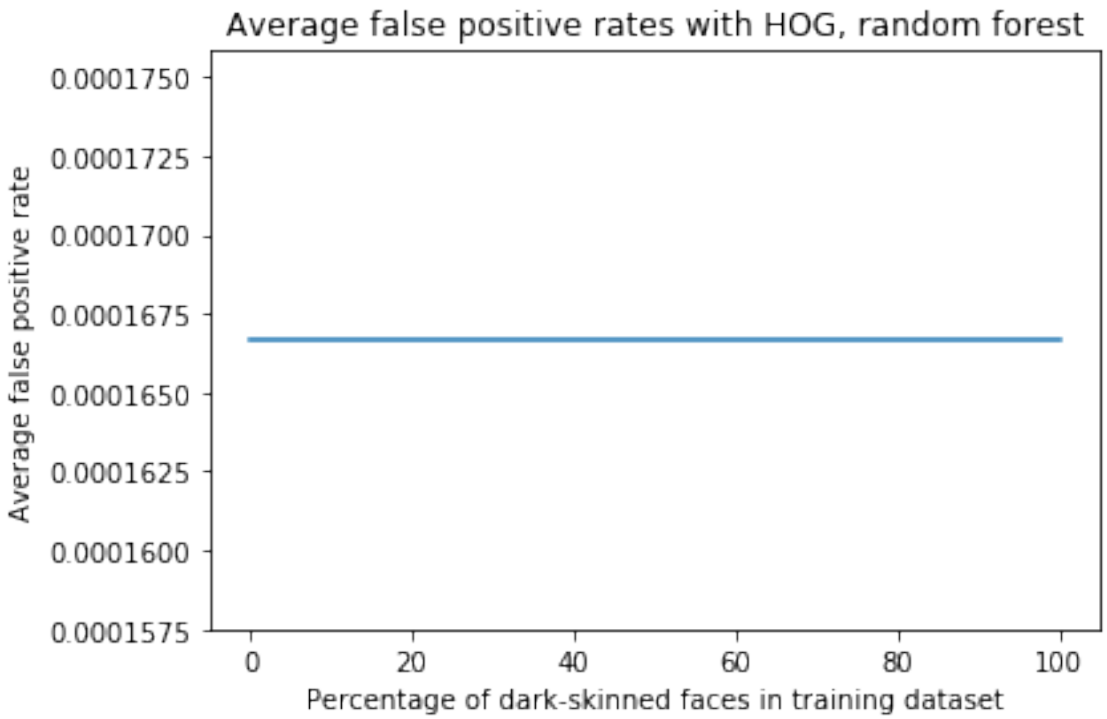
```
trial 0
trial 1
trial 2
trial 3
trial 4
trial 5
trial 6
trial 7
trial 8
trial 9
```

```
In [10]: plot = plot_accuracies(all_overall_rf, all_white_rf, all_black_rf)
         plt.legend(loc="lower right")
         plt.title("Average testing accuracies with HOG, random forest")
         plt.xlabel("Percentage of dark-skinned faces in training dataset")
         plt.ylabel("Average accuracy")
         plt.savefig('hog_avg_rf.png', dpi=300)
         plt.show()
```

```
[0.00628932 0.00498888 0.00520683 0.00516398 0.00442217]
```

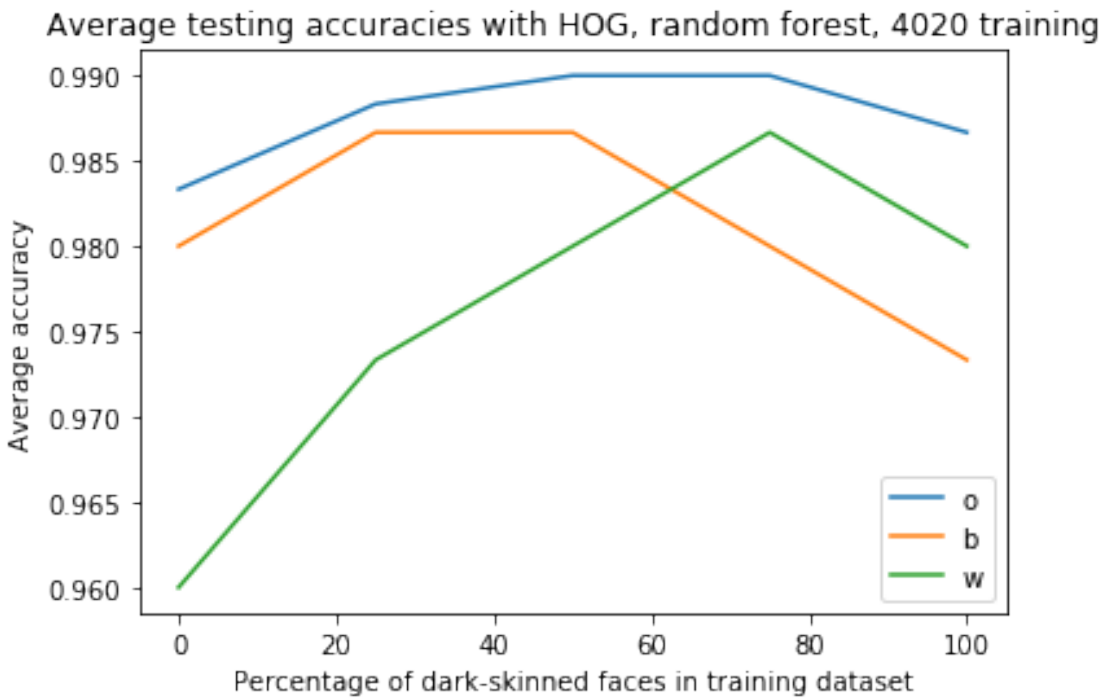


In [11]:
```
# plot false positive rates
x = [0, 25, 50, 75, 100]
avg_fp_rf = np.average(all_falsepos_rf, axis=1)
std_fp_rf = np.std(all_falsepos_rf, axis=1)
plt.plot(x, avg_fp_rf)
#plt.errorbar(x, avg_fp_rf, std_fp_rf)
plt.title("Average false positive rates with HOG, random forest")
plt.xlabel("Percentage of dark-skinned faces in training dataset")
plt.ylabel("Average false positive rate")
plt.savefig('hog_falsepos_rf.png', dpi=300)
plt.show()
```

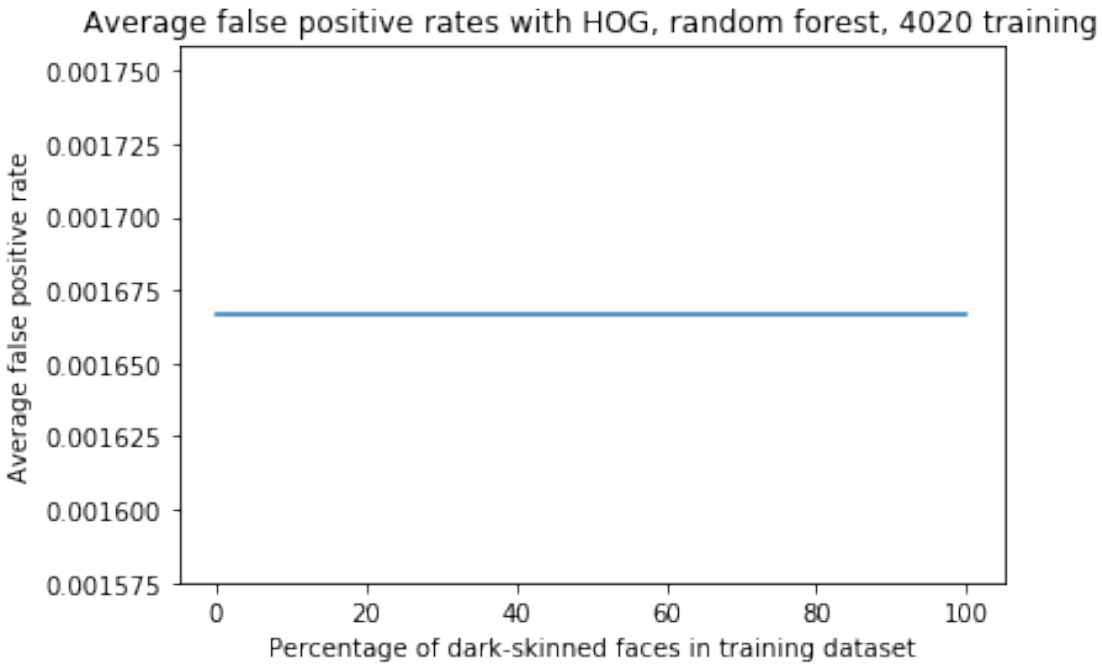



```
In [29]: # train with 4020 examples
         avg_overall_rf, avg_white_rf, avg_black_rf, avg_falsepos_rf = run_classifier(1, 4020,

trial 0
```

```
In [31]: # plot accuracies
         x = [0, 25, 50, 75, 100]
         plt.plot(x, avg_overall_rf, label='o')
         plt.plot(x, avg_black_rf, label='b')
         plt.plot(x, avg_white_rf, label='w')
         plt.legend(loc="lower right")
         plt.title("Average testing accuracies with HOG, random forest, 4020 training")
         plt.xlabel("Percentage of dark-skinned faces in training dataset")
         plt.ylabel("Average accuracy")
         plt.show()
```



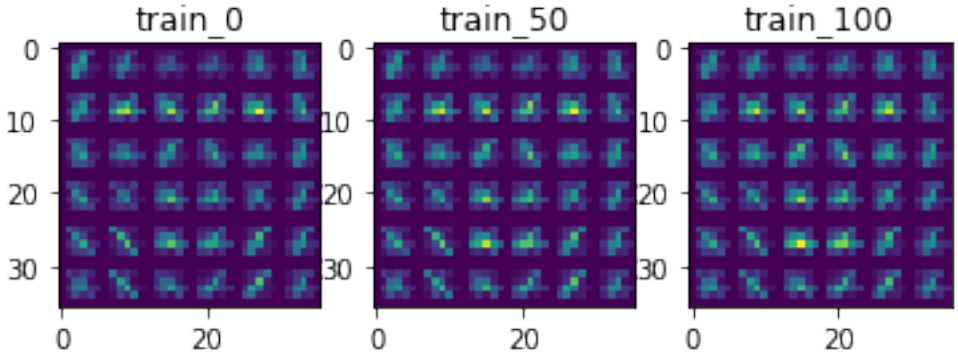
In [32]:
```
# plot false positive rates
x = [0, 25, 50, 75, 100]
plt.plot(x, avg_falsepos_rf)
plt.title("Average false positive rates with HOG, random forest, 4020 training")
plt.xlabel("Percentage of dark-skinned faces in training dataset")
plt.ylabel("Average false positive rate")
plt.show()
```



0.2.1 Compare top hog features

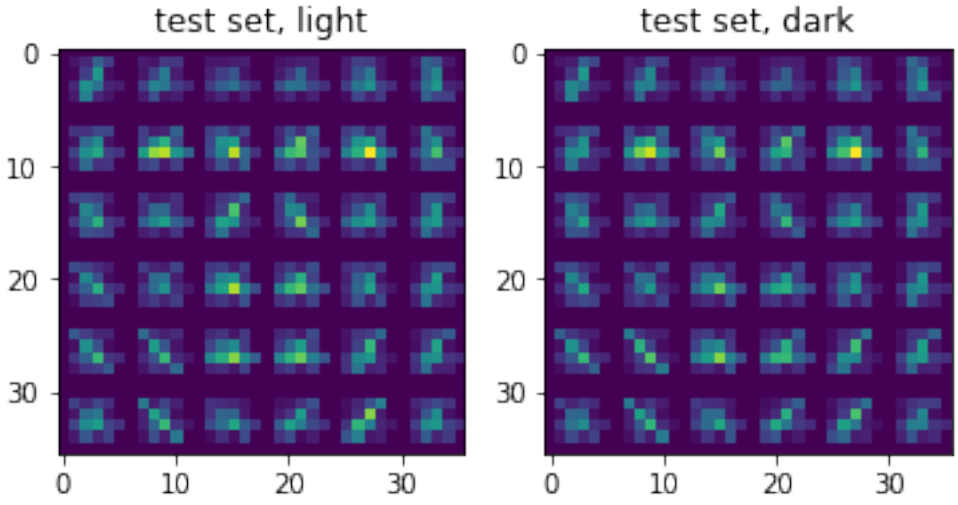
```
In [5]: def get_avg_image(images):
            # average hog images together and display
            avg_image = np.zeros([36, 36])
            num_images = images.shape[0]
            for image in images:
                for i, row in enumerate(image):
                    avg_image[i, :] = np.add(avg_image[i, :], row)
            avg_image = avg_image / num_images
            return avg_image

In [2]: # 0%
        hog_images_0, _ = get_hog_images(4020, 9, 'train_0')

In [3]: # 50%
        # 0%
        hog_images_50, _ = get_hog_images(4020, 9, 'train_50')

In [4]: # 100%
        hog_images_100, _ = get_hog_images(4020, 9, 'train_100')

In [6]: f, axarr = plt.subplots(1,3)
        axarr[0].imshow(get_avg_image(hog_images_0))
        axarr[0].set_title("train_0")
        axarr[1].imshow(get_avg_image(hog_images_50))
```

```
axarr[1].set_title("train_50")
axarr[2].imshow(get_avg_image(hog_images_100))
axarr[2].set_title("train_100")
f.savefig('avg_hog_features_train.png', dpi=300)
```



In [37]:
```
# hog descriptors for testing data
hog_images_test, races = get_hog_images(300, 9, 'testing_faces')# only need first hal
w_hog_images = hog_images_test[races == 0]
b_hog_images = hog_images_test[races == 1]
```

In [38]:
```
f, axarr = plt.subplots(1,2)
axarr[0].imshow(get_avg_image(w_hog_images))
axarr[0].set_title("test set, light")
axarr[1].imshow(get_avg_image(b_hog_images))
axarr[1].set_title("test set, dark")
f.savefig('avg_hog_features_test.png', dpi=300)
```



In [ ]: