recapitulation_support_vector_machine

November 23, 2021

Lau Møller Andersen
November 23 2021

[1]:
```
# import data
from sklearn import datasets
import numpy as np
import matplotlib.pyplot as plt
iris = datasets.load_iris()

X = iris.data[:, 0:2] ## look at first two features
y = iris.target
```

[2]:
```
## find random training indices
from random import sample
np.random.seed(7)
indices = np.arange(0, len(y))
train_indices = sample(range(len(y)), 120)
test_indices = np.setdiff1d(indices, train_indices)

X_train = X[train_indices, :]
y_train = y[train_indices]
X_test  = X[test_indices, :]
y_test  = y[test_indices]
```

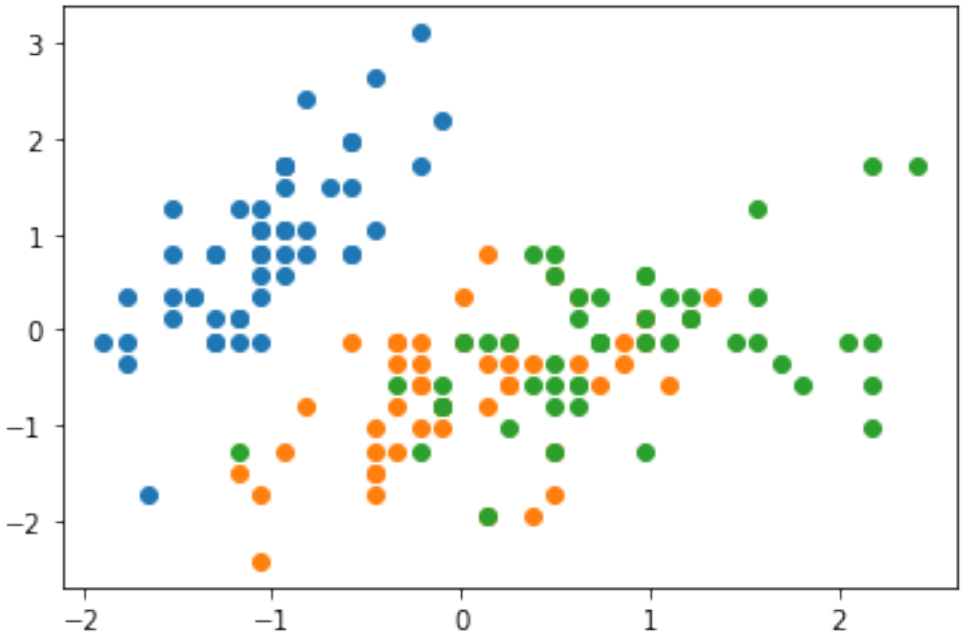
[3]:
```
## scale data    remove mean and divide by sd :)) (standardizing)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
X_combined_std = sc.transform(X)
```

[4]:
```
## kernel function
def kernel(x1, x2, sd):
    gamma = 1 / sd**2 ## precision
    return np.exp(-gamma * np.abs(x1 - x2)**2)
```

[5]:
```
x1 = X_train_std[:, 0:1]
x2 = X_train_std[:, 1:2]

plt.figure()
for target in range(3):
    plt.plot(X_combined_std[y == target, 0],
             X_combined_std[y == target, 1], 'o')
plt.show()
```

a linear classifier works fine for blue, but the other data not



[6]:
```
## decision regions plotting function from Raschka 2015
from matplotlib.colors import ListedColormap

def plot_decision_regions(X, y, classifier,
test_idx=None, resolution=0.02, title=''):
# setup marker generator and color map
    plt.figure()
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
    np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
```

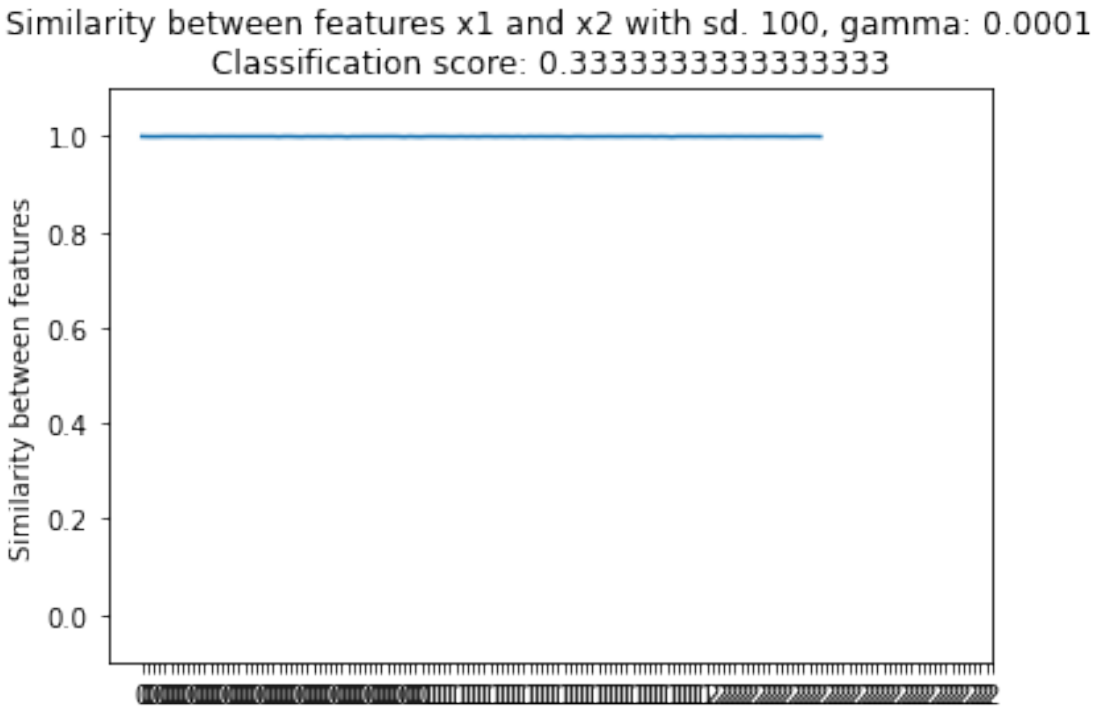
```
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    # plot all samples
    X_test, y_test = X[test_idx, :], y[test_idx]
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
        alpha=0.8, color=cmap(idx),
        marker=markers[idx], label=cl)
        # highlight test samples
    if test_idx is not None:
        X_test, y_test = X[test_idx, :], y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1], color='k',
        alpha=0.2, linewidth=1, marker='o',
        s=55, label='test set')
    plt.title(title)
    plt.show()
```

[9]:
```
## plot radial functions with different precision
import warnings
from sklearn.svm import SVC
sds = [100, 10, 1, 0.1, 0.01, 0.001]   standard deviations
for sd in sds:
    gamma = 1 / sd **2
    k = kernel(x1, x2, sd)
    svm = SVC(kernel='rbf', random_state=0, gamma=gamma)   support vector machine
    svm.fit(X_train_std, y_train)
    score = svm.score(X_test_std, y_test)
    plt.figure()
    plt.plot(k)
    plt.xticks(ticks=range(len(y)), labels=y)
    plt.title('Similarity between features x1 and x2 with sd. ' + str(sd) + \
              ', gamma: ' + str(gamma) + '\nClassification score: ' +␣
↪str(score))
    plt.ylim(-0.1, 1.1)
    plt.ylabel('Similarity between features')
    plt.show()
    plot_decision_regions(X_combined_std, y, svm, test_indices, title='sd: ' +␣
↪str(sd))
```

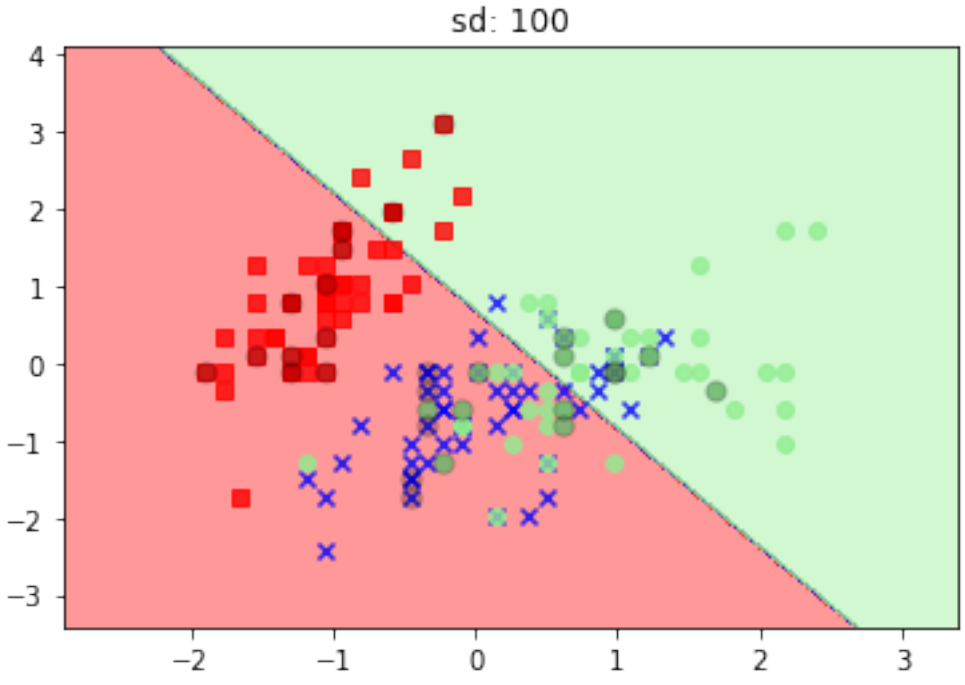
lave plot efter function ??????

gamma controls how much we rate the similarity

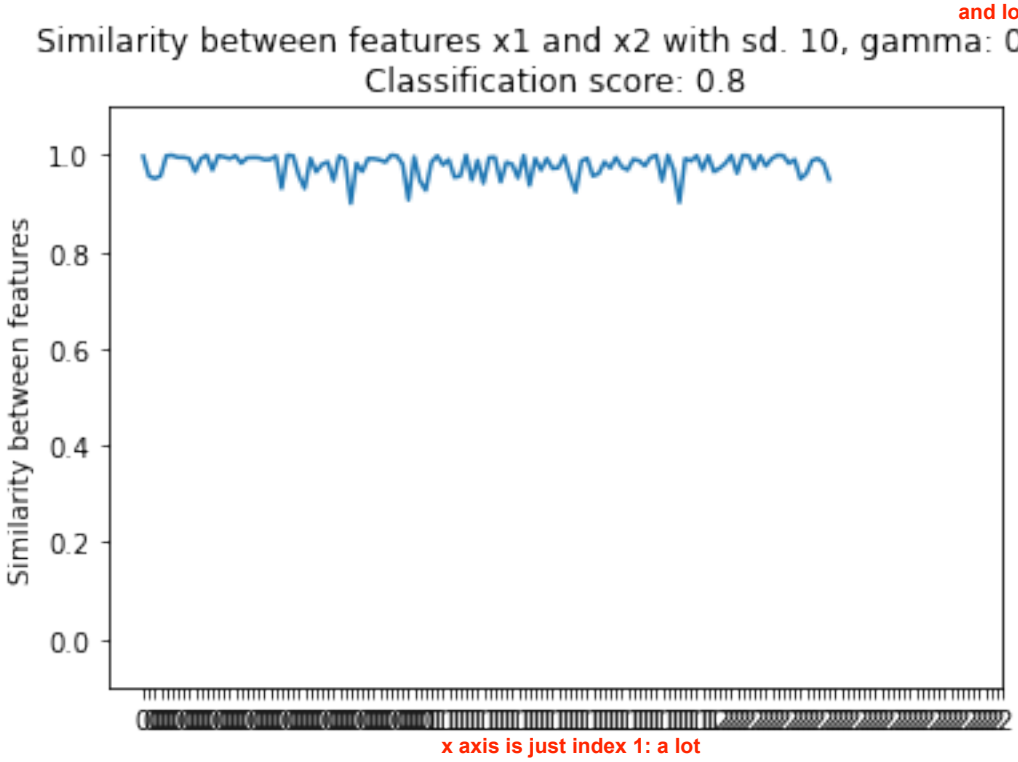
low gamma (low precision)
—> very imprecise



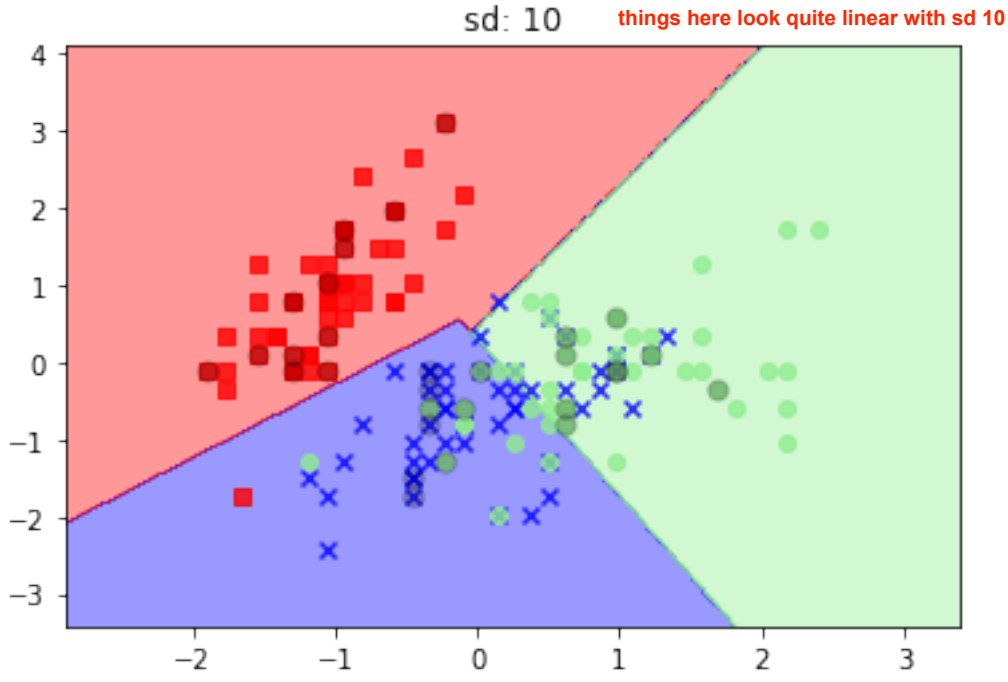
x axis is just index 1: a lot



higher gamma, more precise..
and lower SD

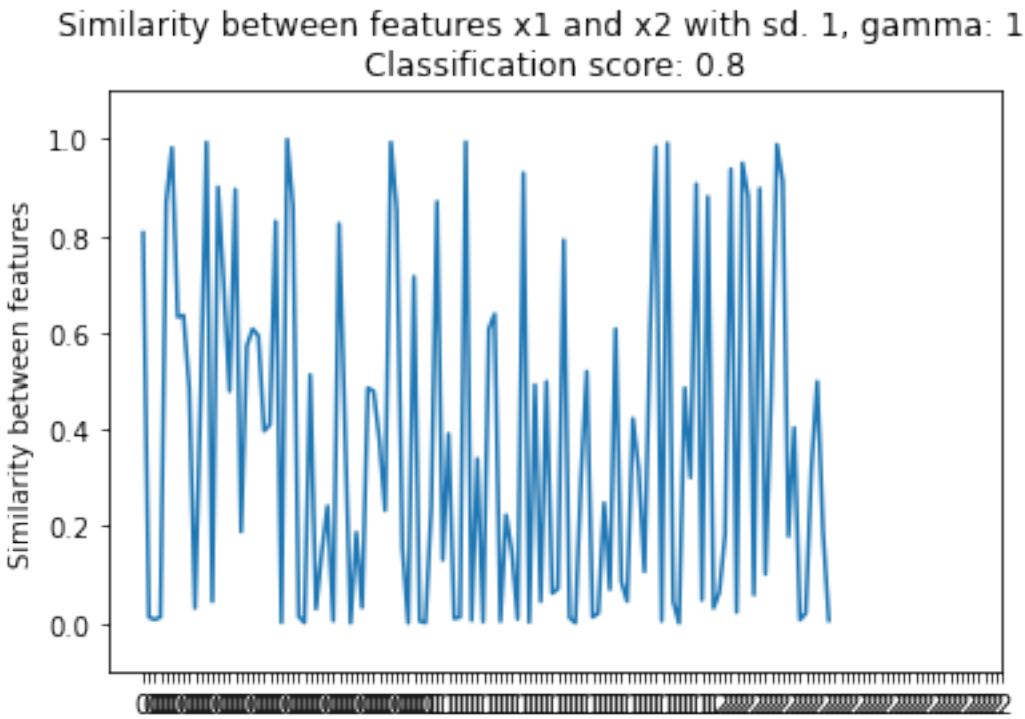


x axis is just index 1: a lot

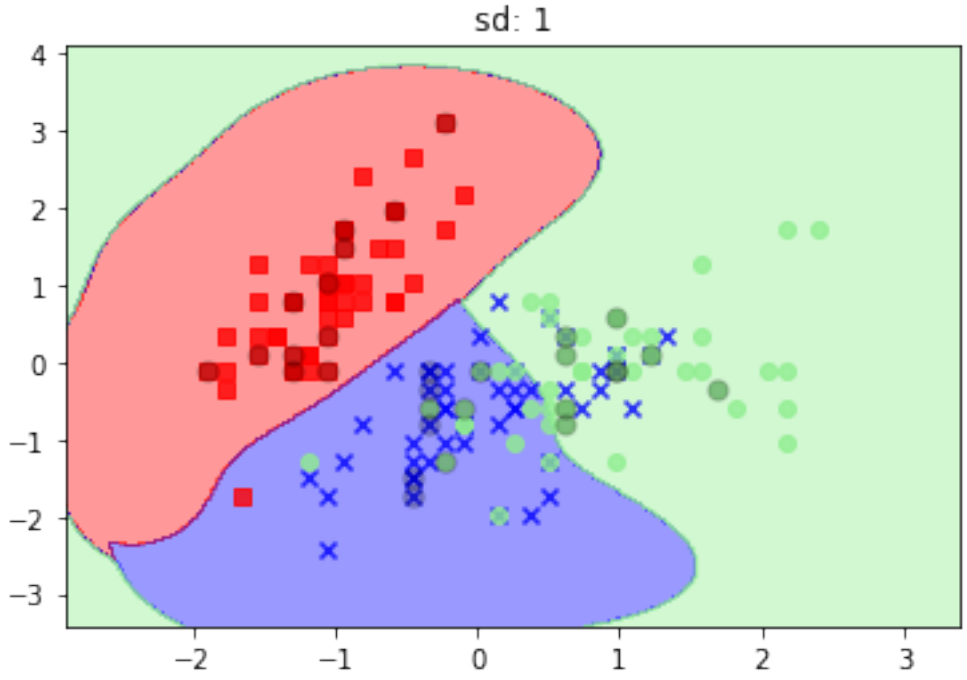


things here look quite linear with sd 10

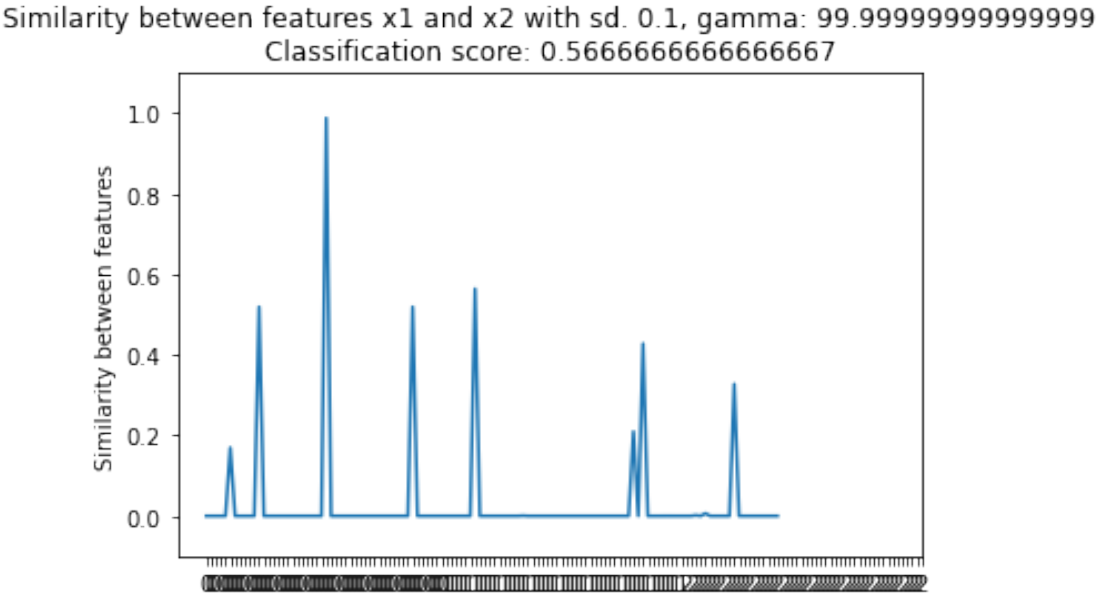
large gamma= (too large??)
smaaaaaaler SD



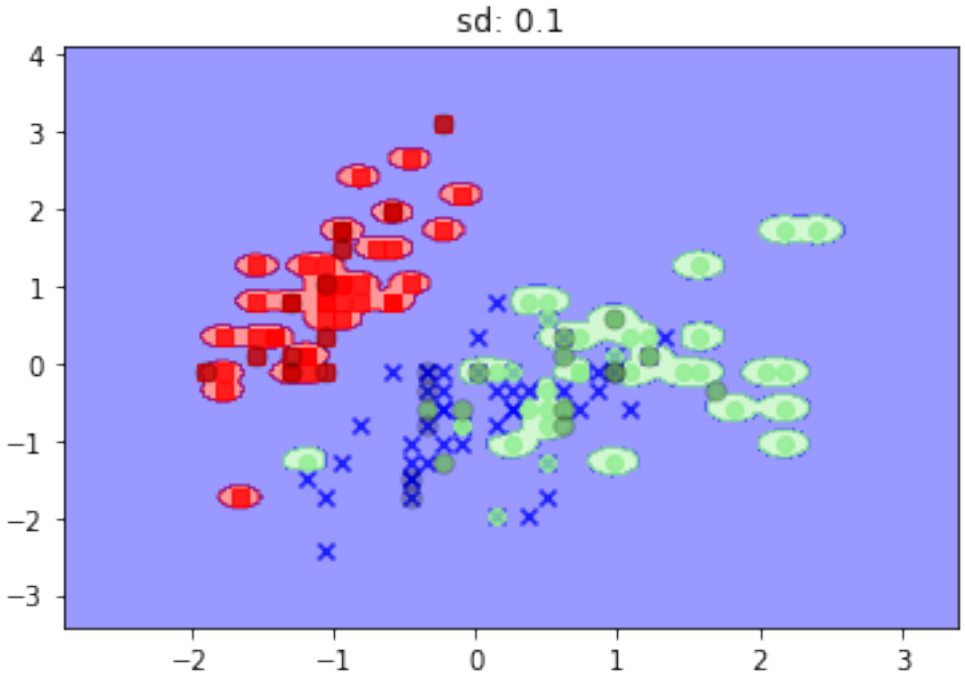
x axis is just index 1: a lot

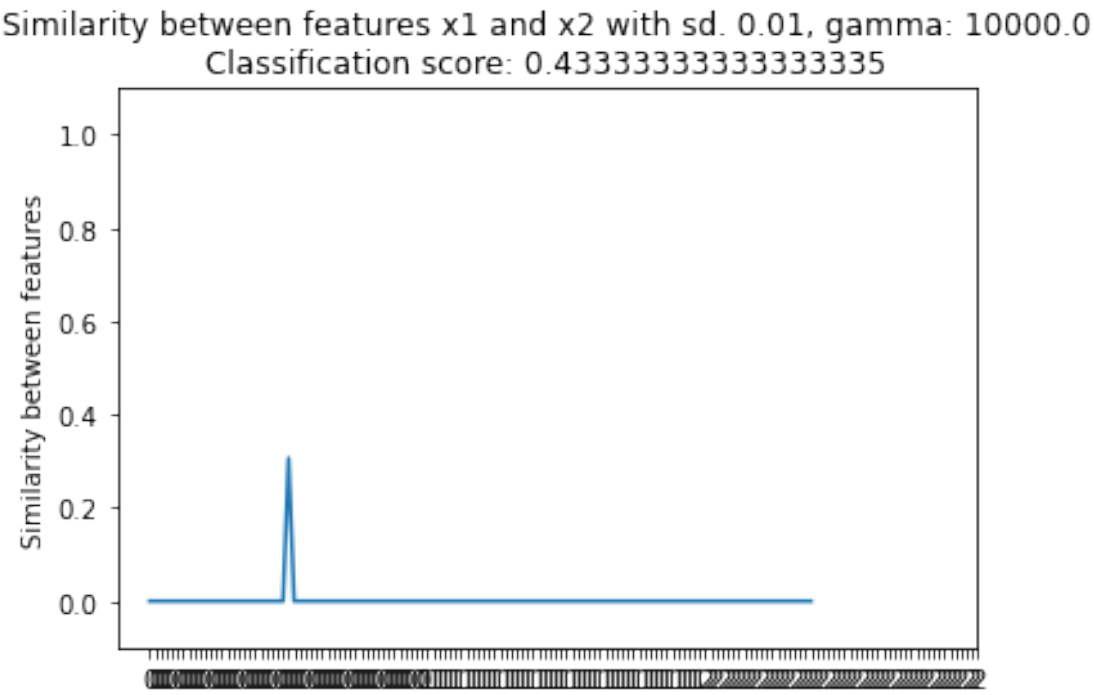


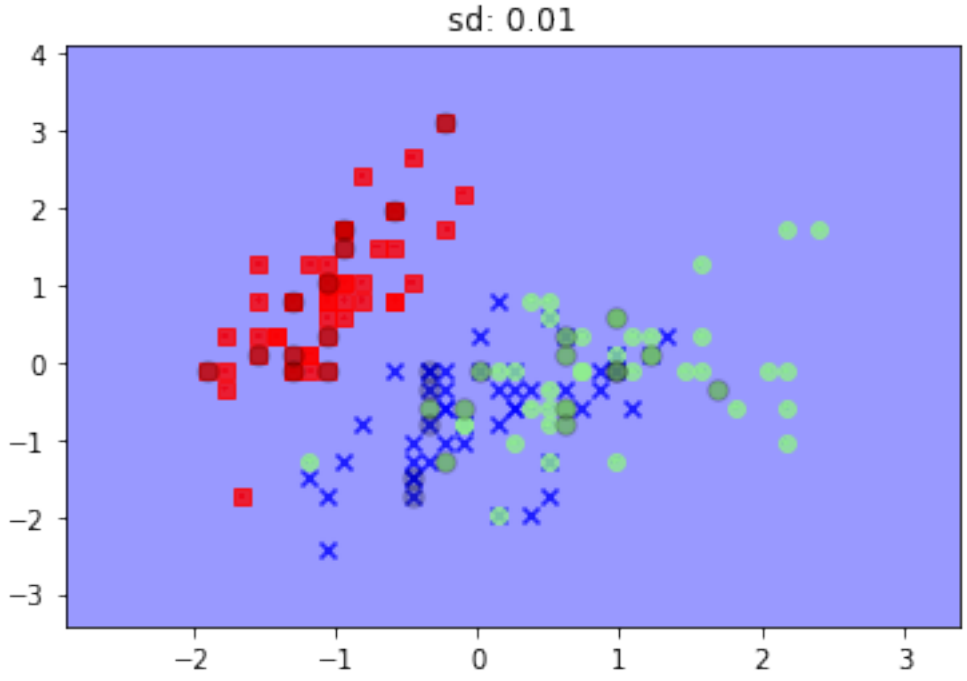
too large gamma, too small sd

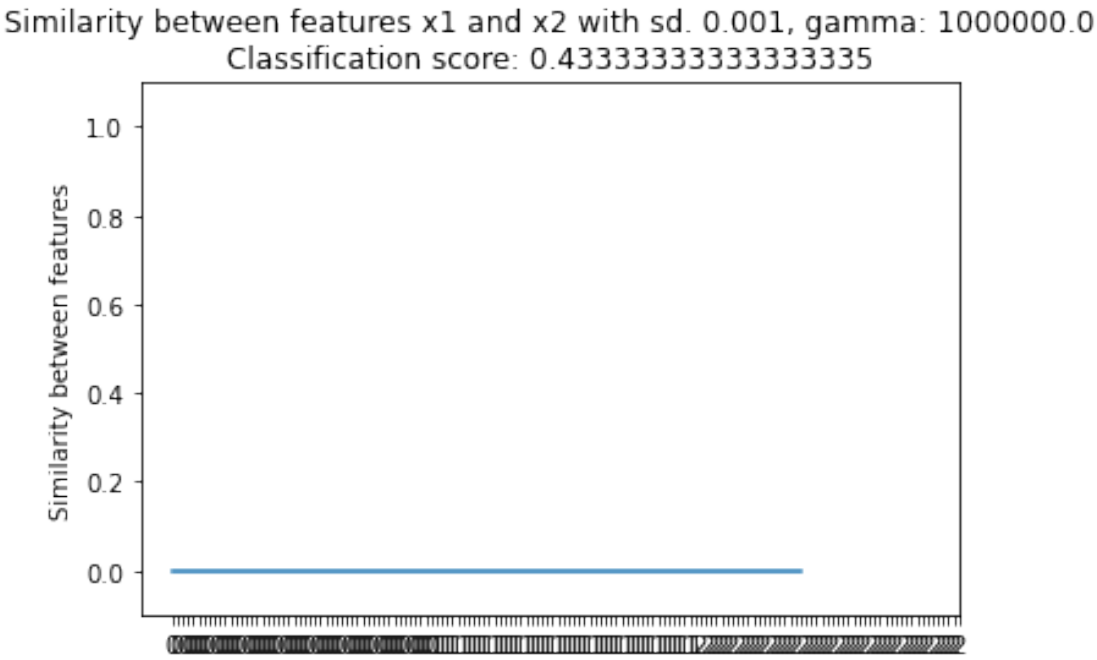


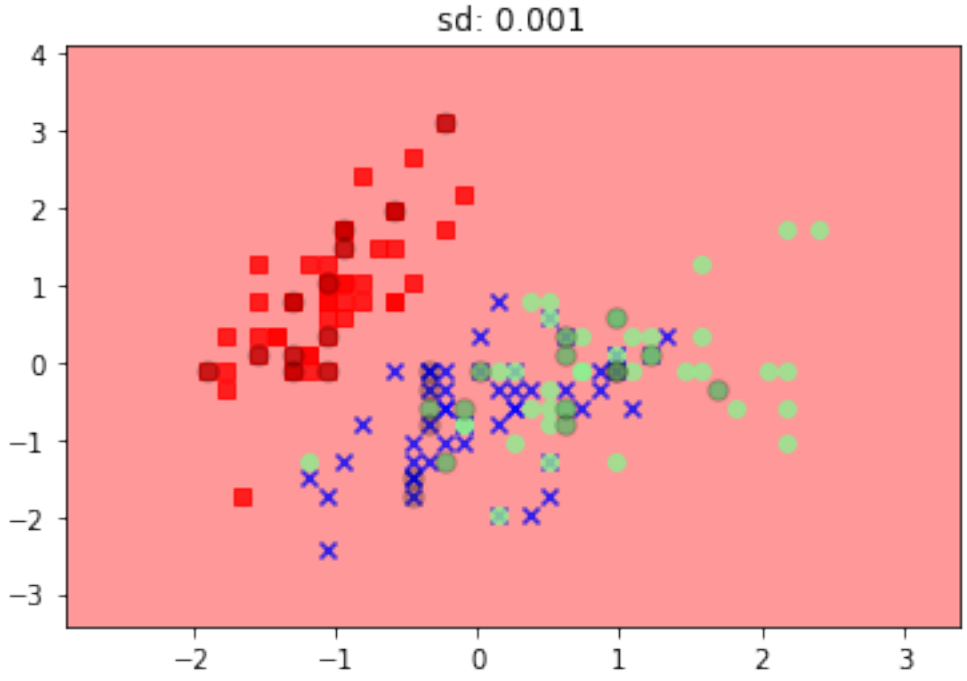
very high precision, not good for generalizing




Similarity between features x1 and x2 with sd. 0.01, gamma: 10000.0
Classification score: 0.43333333333333335
Similarity between features
1.0
0.8
0.6
0.4
0.2
0.0


sd: 0.01
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3


Similarity between features x1 and x2 with sd. 0.001, gamma: 1000000.0
Classification score: 0.43333333333333335
Similarity between features
1.0
0.8
0.6
0.4
0.2
0.0


sd: 0.001
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3

[ ]: