Exercise_09_Support_Vector_Machines

January 17, 2018

0.1 Exercise 09: Support Vector Machines

Group Name: Alwaysonline

0.2 Exercise H9.1: Deriving the C-SVM optimization problem

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib
        %matplotlib inline
        np.random.seed(6)
```

0.3 Exercise H9.2: C-SVM with standard parameters

In this exercise, we use C-SVMs to solve the “XOR”-classification problem from exercise sheet 7. To this end (1) first create a training set of 80 data as described in exercise H7.1 and (2) create a test set of 80 data from the same distribution. 1

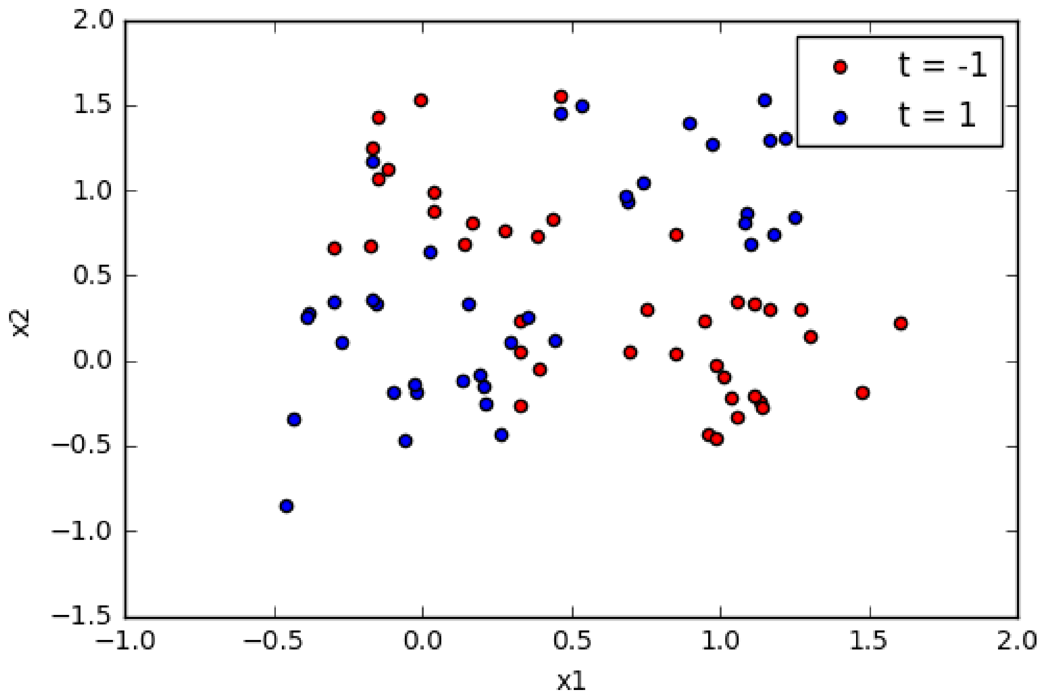
```
In [2]: def gendate(n):
            x = np.zeros((n, 2))
            for i, coin in enumerate(np.random.rand(int(n/2))):
                if coin < 0.5:
                    x[i] = np.random.multivariate_normal([0, 1], 0.1 * np.identity
                else:
                    x[i] = np.random.multivariate_normal([1, 0], 0.1 * np.identity

            for i, coin in enumerate(np.random.rand(int(n/2))):
                if coin < 0.5:
                    x[n/2+i] = np.random.multivariate_normal([0, 0], 0.1 * np.ident
                else:
                    x[n/2+i] = np.random.multivariate_normal([1, 1], 0.1 * np.ident

            t = np.zeros(n)
            t[:n/2] = -1
            t[n/2:] = 1

            return x, t
```

```
#sample_data(2)
```

In [3]:
```
train_x, train_t = gendate(80)
test_x, test_t = gendate(80)
```

In [4]:
```
plt.scatter(train_x[:40, 0], train_x[:40, 1], c='r', label='t = -1')
plt.scatter(train_x[40:, 0], train_x[40:, 1], c='b', label='t = 1')
plt.legend(numpoints=1, scatterpoints=1)
plt.xlabel('x1')
plt.ylabel('x2')
```

Out[4]: <matplotlib.text.Text at 0x7f414edb1550>

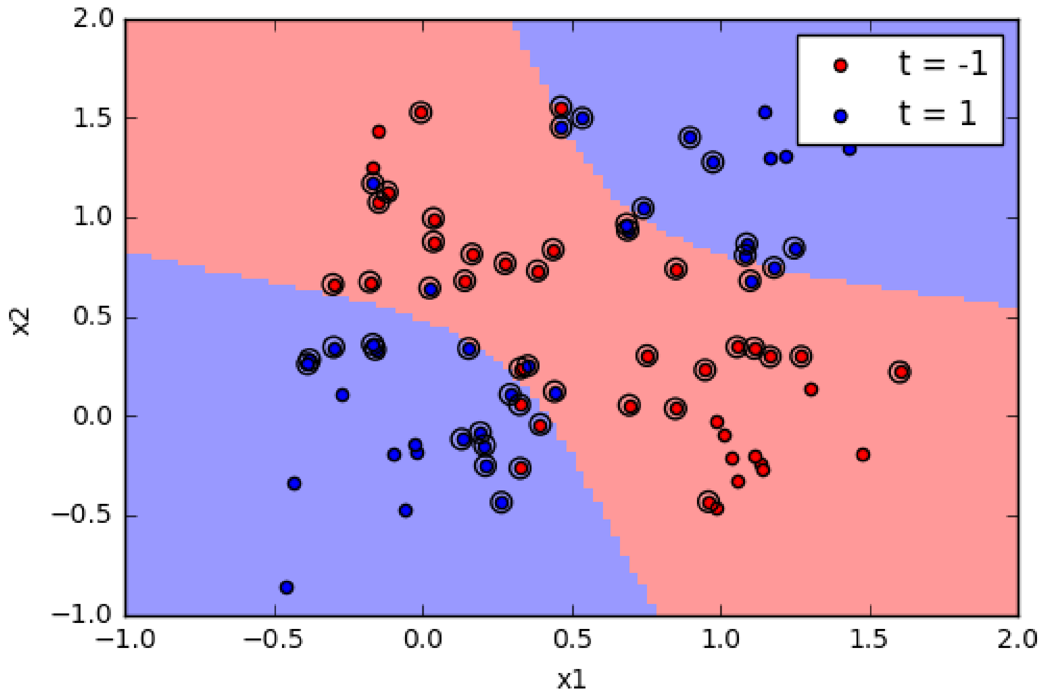


In [ ]:

In [5]:
```
from sklearn.svm import SVC
svc = SVC(kernel='rbf')
svc.fit(train_x, train_t)
print(svc.score(train_x, train_t), svc.score(test_x, test_t))
print(svc.n_support_)
print(sum(svc.n_support_))
```

```
(0.86250000000000004, 0.84999999999999998)
[28 28]
56
```

```
In [6]: def plotDecisionBound(svc):
            svc=svc
            num_points = 100

            xx, yy = np.meshgrid(np.linspace(-1, 2, num_points), np.linspace(-1, 2,
            pred_t = np.zeros((num_points, num_points))
            for i in range(num_points):
                for j in range(num_points):
                    pred_t[i, j] = svc.predict([[xx[i, j], yy[i, j]]])

            cmap = matplotlib.colors.LinearSegmentedColormap.from_list('rb', [[1.,
            plt.pcolor(xx, yy, pred_t, cmap=cmap)
            plt.scatter(svc.support_vectors_[:, 0], svc.support_vectors_[:, 1], s=6
                                facecolors='none', zorder=10, edgecolors='k')
            plt.scatter(train_x[:40, 0], train_x[:40, 1], c='r', label='t = -1')
            plt.scatter(train_x[40:, 0], train_x[40:, 1], c='b', label='t = 1')
            plt.legend(numpoints=1, scatterpoints=1)
            plt.xlabel('x1')
            plt.ylabel('x2')

            plt.xlim(-1, 2)
            plt.ylim(-1, 2)
            plt.show()

        plotDecisionBound(svc)
```



0.4 Exercise H9.3: C-SVM parameter optimization

(a) (2 points) Use cross-validation and grid-search to determine good values5

```
for the C and
the kernel parameter γ. Follow the procedure described in the guide: Define the gri
exponentially growing sequences of C and γ, e.g. C ∈ {2
−6
, 2
−4
, . . . , 2
10}, γ ∈
{2
−5
, 2
−3
, . . . , 2
9}. Make sure youonly use the training data in this step. Plot the mean
training-set classification rate and cross-validation performance as a function of
(e.g. using contour plots as in figure 2 of the guide).
```

```
In [7]: print(np.arange(-6, 12, 2))
        print(np.arange(-5, 10, 1))
```

```
[-6 -4 -2  0  2  4  6  8 10]
[-5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9]
```

```
In [8]: param_grid = {'kernel':["rbf"], 'C': 2.**np.arange(-6, 12, 2), 'gamma': 2.*
```

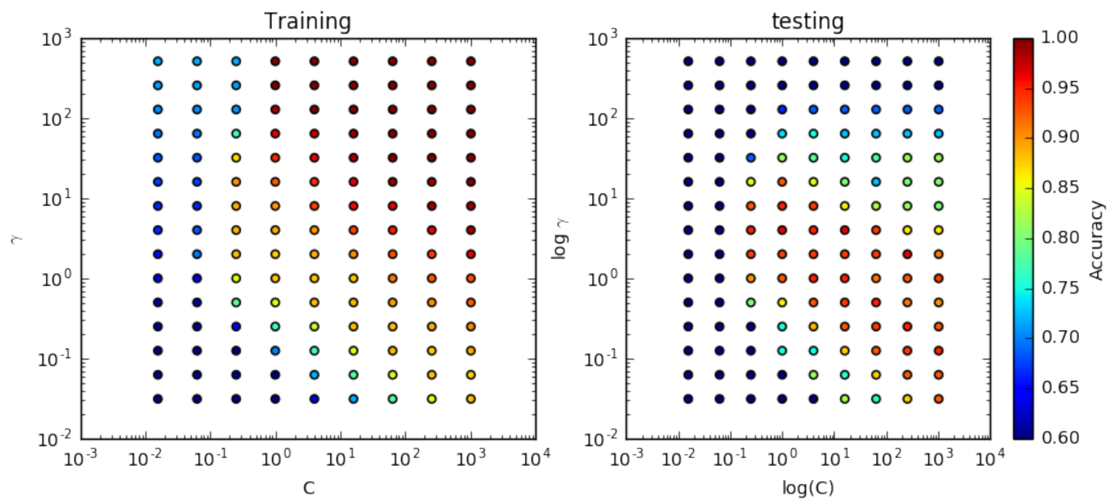
```
In [9]: from sklearn.model_selection import GridSearchCV
        from sklearn.model_selection import ShuffleSplit

        cv = ShuffleSplit(n_splits=10, random_state=0)
        grid_search = GridSearchCV(SVC(), param_grid,return_train_score=True,cv=cv)
        grid_search.fit(train_x, train_t)
```

```
Out[9]: GridSearchCV(cv=ShuffleSplit(n_splits=10, random_state=0, test_size='defaul
               train_size=None),
               error_score='raise',
               estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
          decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
          max_iter=-1, probability=False, random_state=None, shrinking=True,
          tol=0.001, verbose=False),
               fit_params=None, iid=True, n_jobs=1,
               param_grid={'kernel': ['rbf'], 'C': array([  1.56250e-02,   6.25000e
                 4.00000e+00,   1.60000e+01,   6.40000e+01,   2.56000e+02,
```

```
                1.02400e+03]), 'gamma': array([  3.12500e-02,   6.25000e-02,   1.2
                5.00000e-01,   1.00000e+00,   2.00000e+00,   4.00000e+00,
                8.00000e+00,   1.60000e+01,   3.20000e+01,   6.40000e+01,
                1.28000e+02,   2.56000e+02,   5.12000e+02])},
          pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
          scoring=None, verbose=0)
```

In [10]:
```python
import pandas as pd
cv_results = pd.DataFrame(grid_search.cv_results_)
fig = plt.figure(figsize=(10, 4))

plt.subplot(121)
plt.title('Training')
plt.scatter(cv_results['param_C'], cv_results['param_gamma'], c=cv_results
plt.xscale('log')
plt.yscale('log')
plt.xlabel('C')
plt.ylabel(r'$\gamma$')

plt.subplot(122)
plt.title('testing')
plt.scatter(cv_results['param_C'], cv_results['param_gamma'], c=cv_results
plt.xscale('log')
plt.yscale('log')
plt.xlabel('log(C)')
plt.ylabel(r'log $\gamma$')
plt.colorbar(label='Accuracy')
```

Out[10]: <matplotlib.colorbar.Colorbar at 0x7f413e349950>



(b) (1 point) Find the best combination of C and γ and train the RBF C-SVM on the entire training data, this time using these "optimal" parameters. Plot the results in the same way as in exercise H9.2.

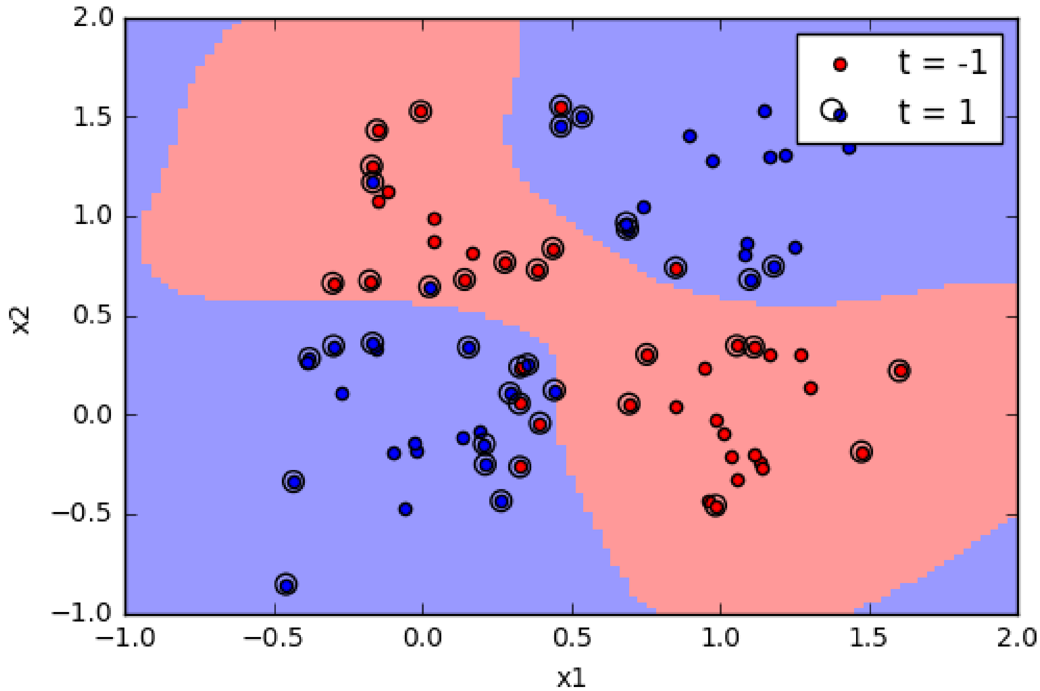
```
In [11]: best_svc = grid_search.best_estimator_
         print(best_svc.score(train_x, train_t), best_svc.score(test_x, test_t))
         print np.shape(best_svc.n_support_)
         print best_svc.n_support_
         print sum(best_svc.n_support_)
```

```
(0.90000000000000002, 0.92500000000000004)
(2,)
[22 21]
43
```

(c) (1 point) Compare the results with those obtained in H9.2, both in terms of statistics (e.g. classification performance, number of support vectors) and visually (e.g. signs of over- and under-fitting). What happens when you divide C or γ by 4?

```
In [13]: print(best_svc.score(train_x, train_t), best_svc.score(test_x, test_t))
         print np.shape(best_svc.n_support_)
         print best_svc.n_support_
         print sum(best_svc.n_support_)
         plotDecisionBound(best_svc)
```

```
(0.90000000000000002, 0.92500000000000004)
(2,)
[22 21]
43
```

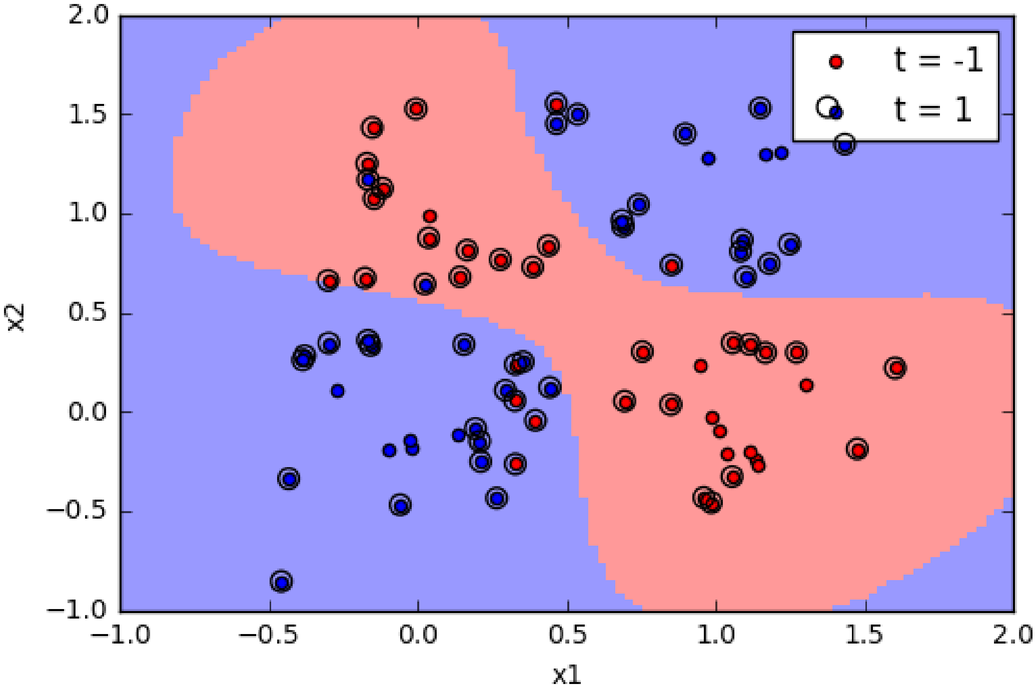


Compare the results with those obtained in H9.2,

85%, 93%, best svm with gridsearch, inclease 8% accuracy. the former plot seems to be underfit, after the optimization, fit beter. differen numbers of support vectors.

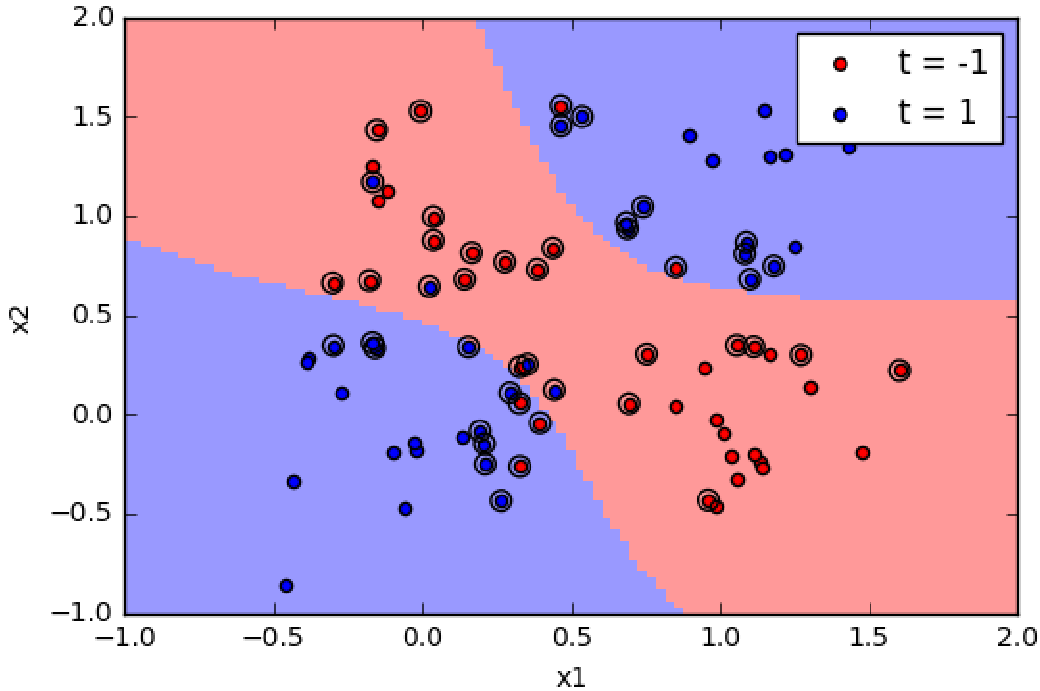
```
In [14]: svc = SVC(C=grid_search.best_params_['C'] / 4., gamma=grid_search.best_par
         svc.fit(train_x, train_t)
         print(svc.score(train_x, train_t), svc.score(test_x, test_t))
         print svc.n_support_
         print sum(svc.n_support_)
         plotDecisionBound(svc)
```

```
(0.90000000000000002, 0.9375)
[31 32]
63
```




```
In [15]: svc = SVC(C=grid_search.best_params_['C'], gamma=grid_search.best_params_[
         svc.fit(train_x, train_t)
         print(svc.score(train_x, train_t), svc.score(test_x, test_t))
         print svc.n_support_
         print sum(svc.n_support_)
         plotDecisionBound(svc)
```

```
(0.8874999999999996, 0.9375)
[24 22]
46
```



If C or γ are divided by 4 (from the best parameters).

It depend on the randome states. But in this case,

both inceased the numbers of support vectors, and perform better in test set. the parameter C control the degree that allow for wrong classifcation.

```
In [ ]:
```