COMPSCI 371D Homework 7

Problem 0 (3 points)

Jihyeon Je (jj271), Tim Ho (th265), Rahul Prakash (rp221)

Part 1: The Representer Theorem

Problem 1.1 (Exam Style)

For logistic regression, we can define the risk with regularization L_T as follows:
$L_T(b, w) + \mu\|\mathbf{v}\|^2 = \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, s(x_n)) + \mu\|\mathbf{v}\|^2$

and the cross entropy loss defined as: $\ell(y, p) = -y \log p - (1 - y)\log(1 - p)$

And therefore $L_T(b, w) + \mu\|\mathbf{v}\|^2 = \frac{1}{N}\sum_{n=1}^{N} -y_n \log s(x_n) - (1 - y_n)\log(1 - s(x_n)) + \mu\| \begin{matrix}\mathbf{w}^* \\ b^*\end{matrix} \|^2$

We conclude that R(||w||) = $\mu\| \begin{matrix}\mathbf{w}^* \\ b^*\end{matrix} \|^2$ is strictly increasing for all positive μ.

In addition, with $s(x_n) = \frac{1}{1+e^{-(b+wx_n)}}$,

we conclude $S(w^T x_1 + b, \ldots, w^T x_N + b) = \frac{1}{N}\sum_{n=1}^{N} -y_n \log s(x_n) - (1 - y_n)\log(1 - s(x_n))$ is a function from $\mathbb{R}^N$ to $\mathbb{R}$

Therefore, the regularized version of L_T, L_{Treg} has the form $L_{Treg} = R(||w||) + S(w^T x_1 + b, \ldots, w^T x_N + b)$. Thus, the Representer Theorem holds. This implies that $\mathbf{w}^*$ in $\mathbf{b}^*$, $\mathbf{w}^* = argmin_{b,w} L_{Treg}(w, b)$ satisfies $\mathbf{w}^*$ = $\sum_{n=1}^{N} \beta_n x_n$. The vector $\mathbf{w}^*$ is a linear combination of the data points in the training set.

Problem 1.2 (Exam Style)

The risk function for the logistic regression classifier is weakly convex, $L(\mathbf{w}, b) \geq L(\mathbf{w}^*, b)$.

However, if $R(||\mathbf{w}||) = R(||\mathbf{w}^*||) = 0$, then $L(\mathbf{w}, b) = L(\mathbf{w}^*, b)$. This is because $L_T(w, b) = R(||w||) + S(w^T x_1 + b, \ldots, w^T x_N + b)$.

Therefore, this does not contradict the assumption that $\mathbf{w}^*$ is optimal, and we cannot conclude that u must be zero and most generally, we can only state that $\mathbf{w}^* = \sum_{n=1}^{N} \beta_n x_n + u$.

Part 2: Function Convexity and Composition

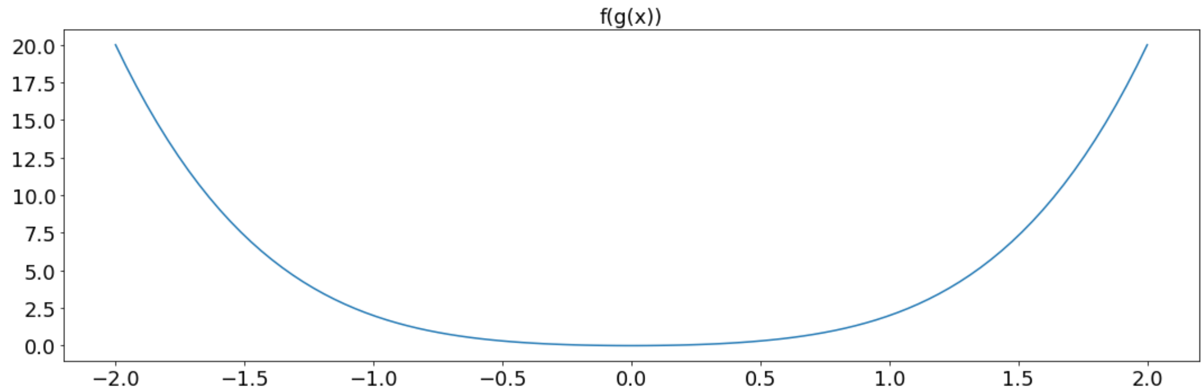
In [295]:
```
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline


def plot(fct, x, title=None, pair=None, font_size=18):
    y = fct(x)
    plt.figure(figsize=(15, 5), tight_layout=True)
    plt.plot(x, y)
    if pair is not None:
        a, b = pair
        f_pair = fct(a), fct(b)
        plt.plot(pair, f_pair)
    if title is not None:
        plt.title(title, fontsize=font_size)
    plt.xticks(fontsize=font_size)
    plt.yticks(fontsize=font_size)
    plt.show()
```

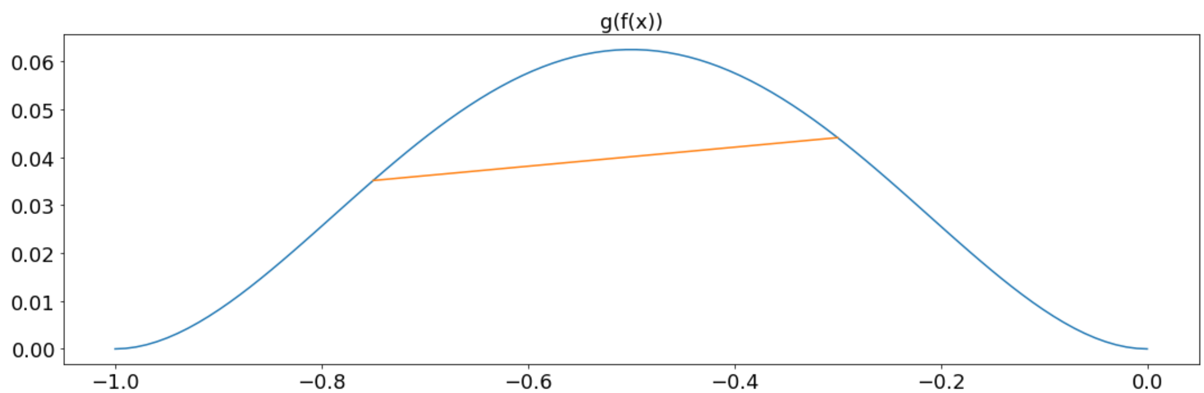
Problem 2.1 (Exam Style)

In [25]:
```
def f(x):
    y = x**2+x
    return y
def g(x):
    y = x**2
    return y

x1 = np.linspace(-1,0,100)
plot(lambda x: f(g(x)),x,'f(g(x))')
```



In [26]:
```
x1 = np.linspace(-1,0,100)
pair = -.75,-0.3
plot(lambda x: g(f(x)),x1,'g(f(x))',pair)
```



Let $f(x) = x^2 + x$ and $g(x) = x^2$

Taking the double deriviative, $f''(x) = 2$ and $g''(x) = 2$ and since $f''(x) > 0$ and $g''(x) > 0$ for all x, we can conclude that functions f and g are strictly convex everywhere.

Then, $f(g(x)) = x^4 + x^2$ and taking the double derivative of the function, we get $f(g(x))'' = 12x^2 + 2 > 0$ for all x and thus $f(g(x))$ is strictly convex everywhere.

$g(f(x)) = x^4 + 2x^3 + x^2$ and taking the double derivative of the function, we get $g(f(x))'' = 12x^2 + 12x + 2$. Therefore, the second derivative is negative beween $x = -0.789$ and $x = -0.2114.

For two graph points $(-0.75, 0.035)$ and $(-0.3, 0.044)$, $f(-0.75u + (-0.3)(1-u)) = f(-0.45u - 0.3)$ and this value is not smaller than $0.035u + 0.044(1-u) = 0.044 - 0.009u$ for all $u \in (0, 1)$ as at $u = 1, f(-0.75) = 0.044$ is larger than $0.044 - 0.009(1) = 0.035$.

Therefore, this function is neither strictly nor weakly convex everywhere in D.

Problem 2.2 (Exam Style)

Given f is strictly convex, $f(u\mathbf{z} + (1-u)\mathbf{z}') \ < \ uf(\mathbf{z}) + (1-u)f(\mathbf{z}')$ for all $u \in (0, 1)$ everywhere on $\mathbb{R}$.

Given g is also at least weakly convex, so $g(u\mathbf{z} + (1-u)\mathbf{z}') \leq ug(\mathbf{z}) + (1-u)g(\mathbf{z}')$ for all $u \in (0, 1)$ everywhere on $\mathbb{R}$.

Applying g to both sides of the convexity equation of f(x), $g(f(u\mathbf{z} + (1-u)\mathbf{z}')) < g(uf(\mathbf{z}) + (1-u)f(\mathbf{z}'))$, this is because g is strictly monotonically increasing everywhere on $\mathbb{R}$, $g(z) < g(z')$ for $z < z'$, so $g(f(u\mathbf{z} + (1-u)\mathbf{z}')) < g(uf(\mathbf{z}) + (1-u)f(\mathbf{z}'))$.

Since g is weakly convex, $g(uf(\mathbf{z}) + (1-u)f(\mathbf{z}')) \leq ug(f(\mathbf{z})) + (1-u)g(f(\mathbf{z}'))$ Therefore $g(f(u\mathbf{z} + (1-u)\mathbf{z}')) < ug(f(\mathbf{z})) + (1-u)g(f(\mathbf{z}'))$

From this, we can conclude $h(x) = g(f(x))$ is strictly convex everywhere on $\mathbb{R}$.

Problem 2.3 (Exam Style)

Given f is both weakly convex and weakly concave, $f(u\mathbf{z} + (1-u)\mathbf{z}') = uf(\mathbf{z}) + (1-u)f(\mathbf{z}')$ for all $u \in (0, 1)$ everywhere on $\mathbb{R}$.

Assuming g is at least weakly convex, so $g(u\mathbf{z} + (1-u)\mathbf{z}') \leq ug(\mathbf{z}) + (1-u)g(\mathbf{z}')$ for all $u \in (0, 1)$ everywhere on $\mathbb{R}$.

Applying g to both sides of the convexity equation of f(x), $g(f(u\mathbf{z} + (1-u)\mathbf{z}')) = g(uf(\mathbf{z}) + (1-u)f(\mathbf{z}'))$.

If g is weakly convex, $g(uf(\mathbf{z}) + (1-u)f(\mathbf{z}')) \leq ug(f(\mathbf{z})) + (1-u)g(f(\mathbf{z}'))$ Therefore $g(f(u\mathbf{z} + (1-u)\mathbf{z}')) \leq ug(f(\mathbf{z})) + (1-u)g(f(\mathbf{z}'))$

If g is strongly convex, $g(uf(\mathbf{z}) + (1-u)f(\mathbf{z}')) < ug(f(\mathbf{z})) + (1-u)g(f(\mathbf{z}'))$ Therefore $g(f(u\mathbf{z} + (1-u)\mathbf{z}')) < ug(f(\mathbf{z})) + (1-u)g(f(\mathbf{z}'))$

Therefore, function h, where $h(x) = g(f(x))$, has the same convexity as g.

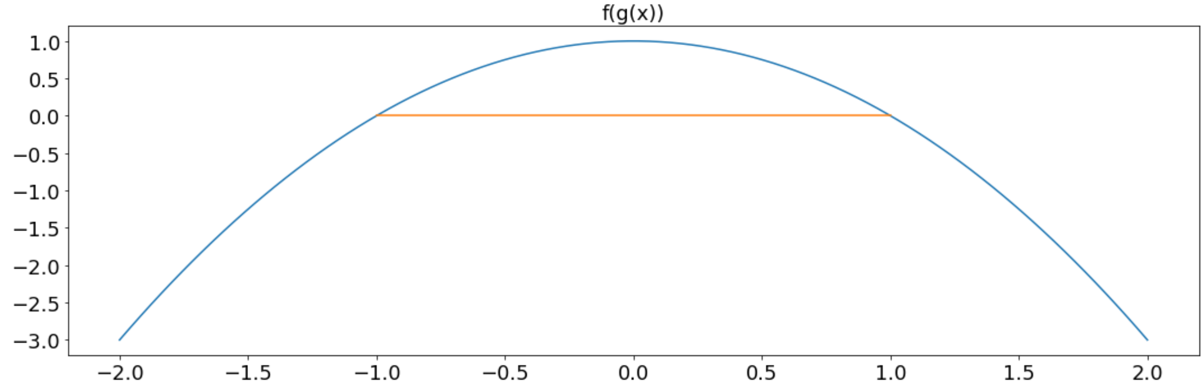
Problem 2.4 (Exam Style)

Let $f(x) = -x + 1$ and $g(x) = x^2$ Taking the second derivative of $g(x)$, $g''(x) = 2 > 0$ and thus $g(x)$ is strictly convex everywhere.

We can let $f(g(x)) = -x^2 + 1$. We can use the plot function to prove this:

In [33]:
```
def f(x):
    y = -x+1
    return y
def g(x):
    y = x**2
    return y

x1 = np.linspace(-0.75,-0.5,100)
pair = -1,1
plot(lambda x: f(g(x)),x,'f(g(x))', pair)
```



Part 3: Support Vector Machines

In [2]:
```
from urllib.request import urlretrieve
from os import path as osp


def retrieve(file_name, semester='fall21', course='371d', homework=7):
    if osp.exists(file_name):
        print('Using previously downloaded file {}'.format(file_name))
    else:
        fmt = 'https://www2.cs.duke.edu/courses/{}/compsci{}/homework/{}
/{}'
        url = fmt.format(semester, course, homework, file_name)
        urlretrieve(url, file_name)
        print('Downloaded file {}'.format(file_name))
```

In [3]:
```
import pickle

retrieve('data.pickle')
with open('data.pickle', 'rb') as file:
    data = pickle.load(file)
```

```
Downloaded file data.pickle
```

In [5]:
```
c_values = [0.0001, 0.001, 0.01, 0.1, 1., 10., 100.]
```

Problem 3.1

In [78]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import numpy as np

classifier = SVC(kernel='rbf', gamma='auto')
hyper_parameters = {'C': c_values}
folds = 15
c = GridSearchCV(classifier, hyper_parameters, scoring='accuracy', cv=fo
lds)
```

In [132]:
```
h = c.fit(data['train']['x'], data['train']['y'])
```

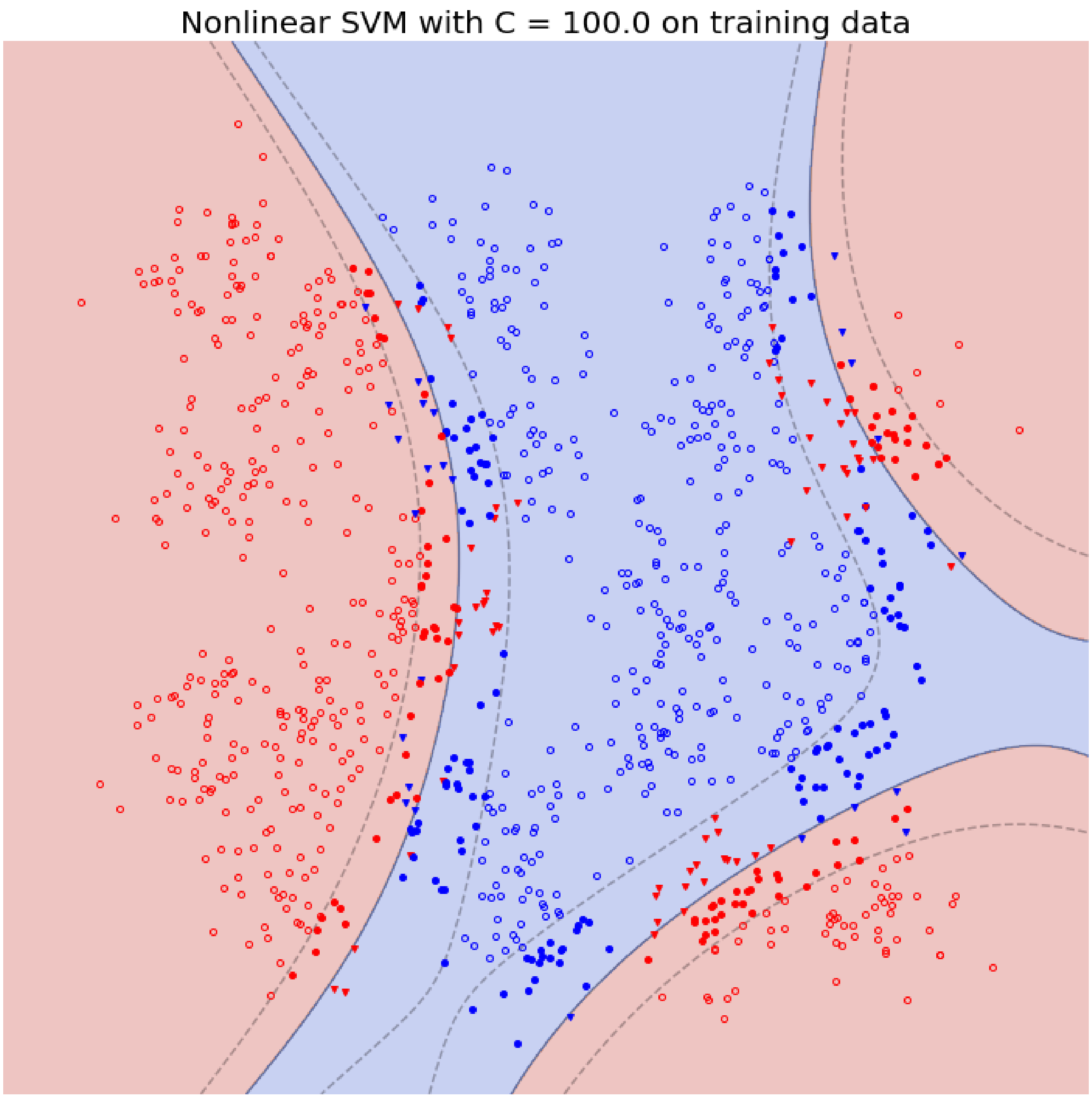
In [293]:
```
def plotting(dtype):
    xcoor =data[dtype]['x'][:,0]
    ycoor = data[dtype]['x'][:,1]
    lbls = data[dtype]['y']
    xx = np.linspace(-0.3, 1.3, 1000)
    yy = np.linspace(-0.2, 1.3, 1000)
    X,Y = np.meshgrid(xx,yy)
    z = h.predict(np.c_[X.ravel(), Y.ravel()]).reshape(X.shape)
    plt.contourf(X, Y, z, cmap=plt.cm.coolwarm_r, alpha=0.3)
    decfct = h.decision_function(np.c_[X.ravel(), Y.ravel()]).reshape(X.
shape)
    plt.contour(X, Y, decfct, colors='k',levels = [-1,0,1], linestyles =
['--','-','--'], alpha=0.3)

    #cdict = {-1: 'red', 1: 'blue'}
    support = h.best_estimator_.support_ if dtype == 'train' else [-1]
    prop_count = 0
    preds = h.decision_function(data[dtype]['x'])
    for i in range(0,1000):
        marker = 'o' if (lbls[i] * preds[i])>0 else 'v'
        if (lbls[i] * preds[i])>0:
            prop_count += 1
        color = 'red' if (lbls[i] == -1) else 'blue'
        fill = 'full' if i in support else 'none'
        plt.plot(xcoor[i], ycoor[i], c = color, marker = marker, markers
ize = 4, fillstyle = fill )
        plt.title(f'Nonlinear SVM with C = {h.best_params_["C"]} on {dty
pe}ing data',
                  fontsize=20)

    plt.axis('off')
    if (dtype=='train'):
        print(f'number of support vector found: {len(support)}')
    print(f'percent accuracy for {dtype}ing: {prop_count/10}%')
```

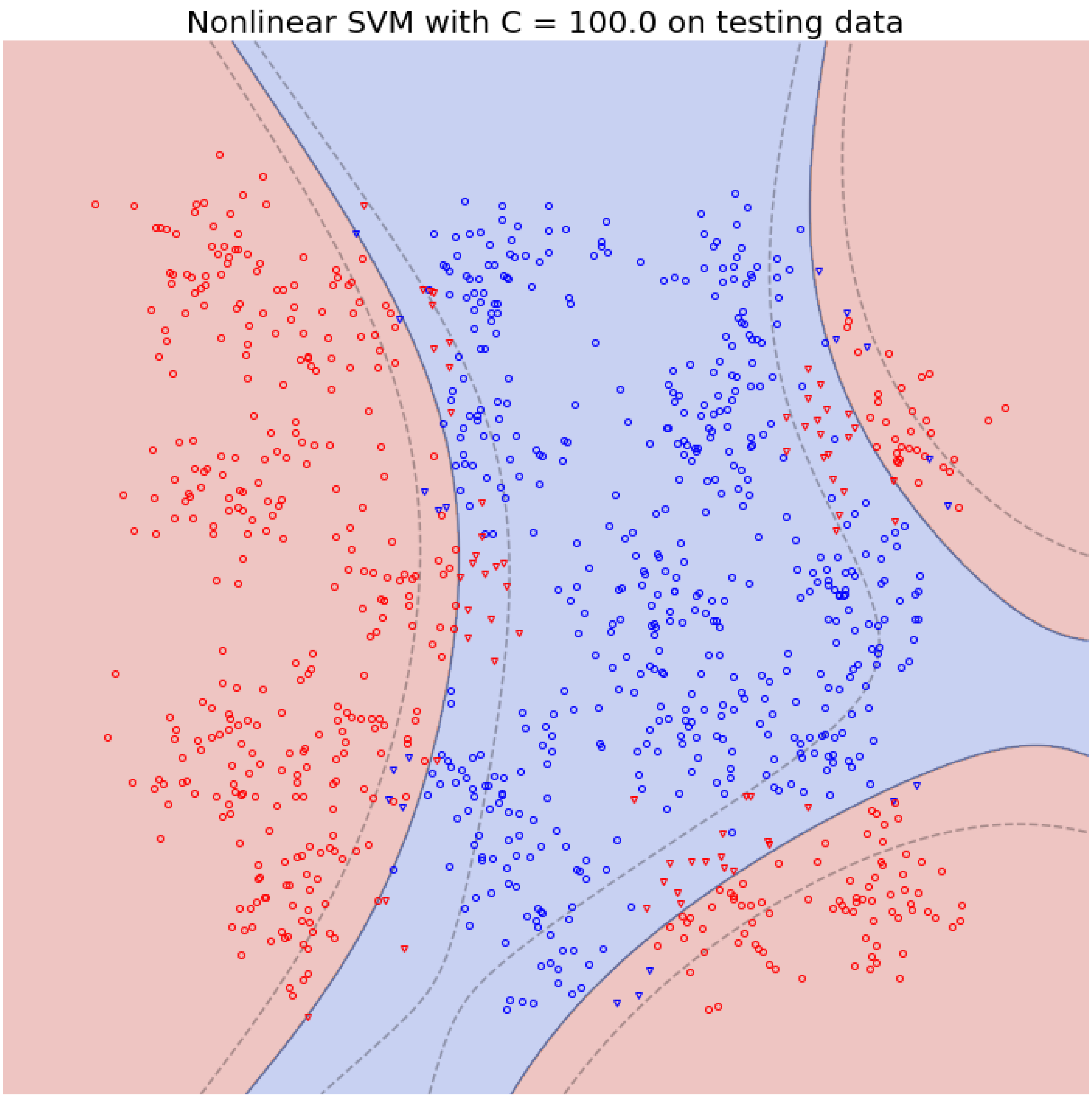
In [299]:
```
fig = plt.figure(figsize=(10, 10), tight_layout=True)
plotting('train')
```

```
number of support vector found: 309
percent accuracy for training: 91.3%
```



```
In [300]: fig = plt.figure(figsize=(10, 10), tight_layout=True)
          plotting('test')
```

```
percent accuracy for testing: 91.6%
```



Problem 3.2 (Exam Style)

The RBF SVM classifier neither overfit nor underfit (thus neither) for this problem. Accuracy in the training set was quite high (91.3%), but the test accuracy was high as well (91.6%). Thus, the algorithm probably did not overfit (since test accuracy is high) nor did it underfit as it was able to generalize quite well to unknown data.

Compared to the linear SVM, the RBF SVM also used almost half as many support vectors, and gained a higher training and test accuracy. This is because the majority of the training/test data is not very linearly separable, which means that a non-linear decision boundary can generalize better. Thus, the RBF SVM performs better than the linear SVM.