CSCI-GA.2566-001
Foundations of Machine Learning

Problem Set 2 Aditya Ramesh

Exercise 2.1.

(a) What is the VC-dimension of circles in the plane?

(b) Use the previous question and the results of the previous homework to give an upper bound for the VC-dimension of the family of concepts defined as the intersection of p circles.

(c) Give an upper bound on the VC-dimension of convex polytopes with p sides.

Lemma 1. *Let V be an inner product space with induced norm $\|\cdot\|_V : V \to \mathbb{R}$. If we define the metric $d : V^2 \to \mathbb{R}$ by $d(x, y) = \|x - y\|_V$, then (V, d) forms a metric space.*

Lemma 2. *Let V be an inner product space with induced metric d. Then any closed ball $\bar{B}(p, r)$ is convex.*

Proof. If $x, y \in \bar{B}(p, r)$, then $\|x - p\|, \|y - p\| < r$. Let $\lambda \in [0, 1]$. We have

$$\begin{aligned}\|\lambda x + (1 - \lambda)y - p\| &= \|\lambda(x - p) + (1 - \lambda)(y - p)\| \\ &\leq \lambda\|x - p\| + (1 - \lambda)\|y - p\| \\ &< \lambda r + (1 - \lambda)r = r.\end{aligned}$$

It follows that $\lambda x + (1 - \lambda)y \in \bar{B}(p, r)$, as desired. □

Lemma 3. *Let V be an inner product space iwth induced metric d. Let $\bar{B}(p_1, r_1)$ be a closed ball, and let $x, y \in \bar{B}(p_1, r_1)$. Then there exists $\bar{B}(p_2, r_2) \subseteq \bar{B}(p_1, r_1)$ with $x, y \in \partial B(p_2, r_2)$ (the circle of radius r_2 centered at p_2).*

Lemma 4 (Intersecting Chords Theorem)**.** *Let p be the intersection of two chords $\overline{ac}, \overline{bd}$ of a circle. Then $\overline{ap} \cdot \overline{pc} = \overline{bp} \cdot \overline{pd}$.*

Solution.

(a) Let C be the concept class of circles in the plane. We first show that $\text{VC}(C) \geq 3$. To this end, consider the set $S = \{(-1, 0), (1, 0), (0, \sqrt{3})\}$, whose points form an equilateral triangle of side length two. The base of the triangle is centered at the origin. Clearly, the $(+, +, +)$, $(-, -, -)$, and $(+, -, -)$ labelings (and all permutations thereof) are all realizable. If we show that the $(-, +, +)$ labeling is also realizable, then by symmetry, it would follow that the $(+, -, +)$ and $(+, +, -)$ labelings are also realizable. Let $\bar{B}((0, 0), 1)$ be the closed unit ball

centered at the origin. We have $(-1, 0), (1, 0) \in \bar{B}((0, 0), 1)$, but $(0, \sqrt{3}) \notin \bar{B}((0, 0), 1)$, since

$$d((0, 0), (0, \sqrt{3})) = \sqrt{3} > 1.$$

It follows that $\text{VC}(C) \geq 3$.

To prove that $\text{VC}(C)$ is precisely three, we show that no four points can be shattered. Let $S = \{a, b, c, d\} \subset \mathbb{R}^2$. By Radon's Theorem, we can partition S into disjoint sets X_1, X_2 such that $\text{conv}(X_1) \cap \text{conv}(X_2) \neq \varnothing$. This yields the following three cases for the cardinalities of X_1 and X_2, provided that we relabel X_1 and X_2 as necessary:

1. $|X_1| = 0$ and $|X_2| = 4$;
2. $|X_1| = 1$ and $|X_2| = 3$; and
3. $|X_1| = 2$ and $|X_2| = 2$.

We first consider case 1. Suppose without loss of generality that a, b, c, d lie on the x-axis, from left to right, in that order. By Lemma 2, the $(+, -, +, +)$ labeling is not realizable. Now consider case 2. Then $\text{conv}(X_2)$ is a triangle, and X_1 is a single point contained within this triangle. Assume without loss of generality that $d \in X_1$, and $a, b, c \in X_2$. As a simple consequence of the triangle inequality, any triangle is convex. Thus by Lemma 2, any circle containing a, b and c must contain the triangle $\text{conv}(X_2)$. It follows that the $(+, +, +, -)$ labeling is not realizable.

Now consider case 3. Then $\text{conv}(X_1 \cup X_2)$ is a quadrilateral $abcd$. Suppose that $\overline{ac}, \overline{bc}$ are the diagonals of the quadrilateral, and let p be their intersection. Assume without loss of generality that $\overline{ap} \cdot \overline{pc} \geq \overline{bp} \cdot \overline{pd}$. Further assume that a, c lie on the y-axis, with a above b, c left of the y-axis, and d right of the y-axis. Suppose that the labeling $(+, -, +, -)$ is realizable, for otherwise we are done. By Lemma 3, there exists a closed ball $\bar{B}(p, r)$ such that $a, c \in \partial B(p, r)$, but $b, d \notin \bar{B}(p, r)$.

The circle $\partial B(p, r)$ must intersect the line through $\overline{bd}$ twice; call these intersections e and f, where e lies left of f. Now suppose that e lies left of b. Since $e, p \in \bar{B}(p, r)$ and $b \in \overline{ep}$, we can apply Lemma 2 to reach a contradiction. Repeating this argument with f and d shows that $e, f \in \overline{bd}$. Applying Lemma 4 to the chords $\overline{ac}$ and $\overline{ef}$ of $\partial B(p, r)$ shows that

$$\overline{ap} \cdot \overline{pc} = \overline{ep} \cdot \overline{pf} \leq \overline{bp} \cdot \overline{pd}.$$

But we assumed that the $(+, -, +, -)$ is realizable, so $b, p \notin \bar{B}(p, r)$. So in fact we have strict inequality, implying that

$$\overline{ap} \cdot \overline{pc} = \overline{ep} \cdot \overline{pf} < \overline{bp} \cdot \overline{pd}.$$

This contradicts our assumption that $\overline{ap} \cdot \overline{pc} \geq \overline{bp} \cdot \overline{pd}$. It follows that the $(-, +, -, +)$ labeling is not realizable. Hence no four points can be shattered by C, so we conclude that $\text{VC}(C) = 3$.

(b) Let C be a concept class of VC-dimension d, and let

$$H = \left\{ \bigcap_{i=1}^{p} c_i : c_i \in C \right\}.$$

A result from the previous assignment shows that

$$\Pi_H(m) \leq (\Pi_C(m))^p.$$

By Sauer's lemma, we have

$$\Pi_H(m) \leq \left(\frac{em}{d}\right)^{pd},$$

for all $m \geq d$. To bound VC(H), we seek a value for $m \geq d$ such that

$$\left(\frac{em}{d}\right) < 2^m.$$

Choosing $m = 2pd\log_2(3p)$, we get

$$\begin{aligned}
\left(\frac{em}{d}\right) &< 2^m && \text{iff} && (1)\\
(2ep\log_2(3p))^{pd} &< (9p^2)^{pd} && \text{iff} \\
2ep\log_2(3p) &< 9p^2 && \text{iff} \\
2e\log_2(3p) &< 9p && \text{iff} \\
2e &< \frac{9p}{\log_2(3p)}. && && (2)
\end{aligned}$$

The last inequality holds for $p = 1$, since

$$\frac{9p}{\log_2 3p} = \frac{9}{\log_2 4} = \frac{9}{2} > 6 > 2e.$$

To show that the last inequality holds for all $p \geq 1$, we show that the RHS of Inequation 2 is increasing. Observe that

$$\frac{d}{dp}\frac{9p}{\log_2 3p} = \frac{9\log_2 3p - 9/\ln 2}{(\log_2 3p)^2}.$$

Clearly, the denominator is positive. The numerator is also positive, for

$$\begin{aligned}
9\log_2 3p - 9/\ln 2 &\geq 9\left(\log_2(3p) - \frac{1}{\ln 2}\right) \\
&= 9\left(\log_2(3p) - \log_2 e\right) \\
&= 9\log_2\left(\frac{3}{e}\right) \\
&> 9\log_2 1 = 0.
\end{aligned}$$

Therefore the RHS of Inequation 2 is increasing, and it follows that Equation 1 holds for all $m \geq d$ and $p \geq 1$. We conclude that

$$\text{VC}(H) \leq 2pd\log_2(3p).$$

In the case of circles in the plane, $d = 3$.

(c) Let P be the concept class of p-sided polytopes in $\mathbb{R}^n$, and C the concept class of hyperplanes. It is shown in the text that $\text{VC}(C) = n + 1$. We have

$$P \subset H := \left\{ \bigcap_{i=1}^{p} c_i : c_i \in C \right\}.$$

By part (2), $\text{VC}(H) \leq 2(n+1)p \log_2 3p$. Since $P \subset H$, we have

$$\text{VC}(P) \leq \text{VC}(H) \leq 2(n+1)p \log_2 3p.$$

Exercise 2.2.

(a) Show that the kernel K defined by $K(x, y) = \min(x, y) - xy$ is a PDS kernel over $[0, 1]^2$.

(b) Show that the kernel K defined by $K(x, y) = \exp(-t \sin^2(x - y))$ is a PDS kernel over $\mathbb{R}^2$ for any $t > 0$.

(c) Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. V could represent a set of documents or biosequences and E the set of connections between them. Let $w[e] \in \mathbb{R}$ denote the weight assigned to edge $e \in E$. The weight of a path is the product of the weights of its constituent edges. Show that the kernel K over V^2 where $K(p, q)$ is the sum of the weights of all paths of length two between p and q is PDS.

Solution.

(a) First note that K is clearly symmetric. We proceed to show that K is PDS by showing that it is the product of the two PDS kernels $\min(x, y)$ and $1 - \max(x, y)$. By the closure of PDS kernels under product, it would follow that K is PDS. To prove that $\min(x, y)$ is a PDS kernel, it suffices to show that there exists a Hilbert space H and a feature mapping $\phi : [0, 1] \to H$ such that $\min(x, y) = \langle \phi(x), \phi(y) \rangle$.

To this end, let H be the Hilbert space of measurable functions $f : [0, 1] \to \mathbb{R}$, with the associated inner product

$$\langle f, g \rangle = \int_0^1 f(t)g(t)dt.$$

Now consider the integral

$$\int_0^1 1\{t \in [0, x]\}1\{t \in [0, y]\}dt.$$

For fixed x and y, $1\{t \in [0, x]\}1\{t \in [0, y]\} = 1$ if and only if $t < \min(x, y)$. Therefore the integral resolves to $\min(x, y)$. Setting $\phi : [0, 1] \to H$ to $\phi(x) = 1\{t \in [0, x]\}$, we see that $\min(x, y)$ can be expressed as the inner product $\min(x, y) = \langle \phi(x), \phi(y) \rangle$. It follows that $\min(x, y)$ is PDS.

Similarly,

$$\int_0^1 1\{t \in [x, 1]\}1\{t \in [y, 1]\}dt = 1 - \max(x, y).$$

So setting $\phi(x) = 1\{t \in [x, 1]\}$ and noting that $1 - \max(x, y) = \langle \phi(x), \phi(y) \rangle$ shows that $1 - \max(x, y)$ is PDS. Since PDS kernels are closed under product, the kernel

$$\begin{aligned} \min(x, y)(1 - \max(x, y)) &= \min(x, y) - \min(x, y) \max(x, y) \\ &= \min(x, y) - xy \end{aligned}$$

is PDS.

(b) By Theorem 5.8 in the text, K is PDS if and only if $\sin^2(x-y)$ is NDS. Observe that

$$\begin{aligned}\sin^2(x-y) &= 1-\cos^2(x-y)\\ &= 1-(\cos(x)\cos(y)-\sin(x)\sin(y))\\ &= 1-\langle\phi(x),\phi(y)\rangle,\end{aligned}$$

where $\phi(x)=(\cos(x),\sin(x))$, and the inner product in the last inequality is the dot product. Since $\|\phi(x)\|=1$,

$$\begin{aligned}\frac{1}{2}\|\phi(x)-\phi(y)\|^2 &= \frac{1}{2}\|\phi(x)\|^2+\frac{1}{2}\|\phi(y)\|^2-\langle\phi(x),\phi(y)\\ &= 1-\langle\phi(x),\phi(y)\rangle.\end{aligned}$$

It follows that

$$\sin^2(x-y)=\frac{1}{2}\|\phi(x)-\phi(y)\|^2.$$

As any squared-norm kernel is NDS, and NDS kernels are closed under products with positive scalars, it follows that $\sin^2(x-y)$ is NDS. Therefore K is PDS.

(c) Let $n = |V|$, and let $W \in \mathbb{R}^{n \times n}$ symmetric matrix defined for all $i, j \in [1, n]$ by

$$W_{ij} = \begin{cases} w[i\text{—}j] & \text{if } i\text{—}j \in E \\ 0 & \text{otherwise.} \end{cases}$$

Consider two nodes $p, q \in V$. If there exists a node $r \in V$ such that p—r, r—$q \in E$, then the weight of the path p—r—q is $W_{pr}W_{qr}$. On the other hand, if either p—$r \notin E$ or r—$q \notin E$, then $W_{pr}W_{qr} = 0$. Therefore, we can write

$$K(p, q) = \sum_{i=1}^{n} W_{pi}W_{qi} = \langle w_p, w_q \rangle,$$

where w_i is the ith column of W and the inner product is the dot product. Since any inner product is PDS, it follows that K is PDS.

Exercise 2.3.

(a) Download and install the `libsvm` software library, and briefly consult the documentation to become more familiar with the tools.

(b) Use the `libsvm` scaling tool to scale the features of all the data. The scaling parameters should be computed only on the training data and then applied to the test data.

(c) Consider the corresponding binary classification which consists of distinguishing two types of splice junctions in DNA sequences using about 60 features. Use SVMs combined with polynomial kernels to tackle this problem.

To do that, randomly split the training data into ten equal-sized disjoint sets. For each value of the polynomial degree, $d = 1, 2, 3, 4$, plot the average cross-validation error plus or minus one standard deviation as a function of C (let other parameters of polynomial kernels in `libsvm` be equal to their default values), varying C in powers of 10, starting from a small value $C = 10^{-k}$ to $C = 10^k$, for some value of k. k should be chosen so that you see a significant variation in training error, starting from a very high training error to a low training error. Expect longer training times with `libsvm` as the value of C increases.

(d) Let (C^*, d^*) be the best pair found previously. Fix C to be C^*. Plot the ten-fold cross-validation error and the test errors for the hypotheses obtained as a function of d. Plot the average number of support vectors obtained as a function of d. How many of the support vectors lie on the margin hyperplanes?

(e) Determine the set of outliers, that is, as discussed in class, the support vectors with corresponding weight $\alpha_i = C^*$. Order the list of outliers $x_1, \ldots, x_l$ in the order of their appearance in the training sample. Remove the first μ fraction of the outliers from the training sample, retrain, and report the test error, for $\mu = 0.2, 0.4, 0.6, 0.8, 1.0$ and $d = d^*$.

Solution.

(a) OK.

(b) The following commands were used to accomplish this.

```
svm-scale -s out/scale_parameters.txt raw/splice_noise_train.txt \
        > out/splice_noise_train_scaled.txt
svm-scale -r out/scale_parameters.txt raw/splice_noise_test.txt \
        > out/splice_noise_test_scaled.txt
```

(c) The following commands were used to prepare the training data as described.

```
% Shuffle the scaled data files.
for i in $(ls out/splice*); do shuf $i -o ${i%.*}_shuffled.txt; done
% Split the scaled and shuffled training data file into 10 equal-sized parts.
% Since the original file consists of 2175 lines, each part will consist of 217
% lines.
sed -n '1,217 p' splice_noise_train_scaled_shuffled.txt > train_1.txt
sed -n '218,434 p' splice_noise_train_scaled_shuffled.txt > train_2.txt
sed -n '435,651 p' splice_noise_train_scaled_shuffled.txt > train_3.txt
sed -n '652,868 p' splice_noise_train_scaled_shuffled.txt > train_4.txt
sed -n '869,1085 p' splice_noise_train_scaled_shuffled.txt > train_5.txt
sed -n '1086,1302 p' splice_noise_train_scaled_shuffled.txt > train_6.txt
sed -n '1303,1519 p' splice_noise_train_scaled_shuffled.txt > train_7.txt
sed -n '1520,1736 p' splice_noise_train_scaled_shuffled.txt > train_8.txt
sed -n '1737,1953 p' splice_noise_train_scaled_shuffled.txt > train_9.txt
sed -n '1954,2170 p' splice_noise_train_scaled_shuffled.txt > train_10.txt
% Generate the data sets for cross-validation.
cat train_2.txt train_3.txt train_4.txt train_5.txt train_6.txt train_7.txt \
        train_8.txt train_9.txt train_10.txt > train_cv_1.txt
cat train_1.txt train_3.txt train_4.txt train_5.txt train_6.txt train_7.txt \
        train_8.txt train_9.txt train_10.txt > train_cv_2.txt
cat train_1.txt train_2.txt train_4.txt train_5.txt train_6.txt train_7.txt \
        train_8.txt train_9.txt train_10.txt > train_cv_3.txt
cat train_1.txt train_2.txt train_3.txt train_5.txt train_6.txt train_7.txt \
        train_8.txt train_9.txt train_10.txt > train_cv_4.txt
cat train_1.txt train_2.txt train_3.txt train_4.txt train_6.txt train_7.txt \
        train_8.txt train_9.txt train_10.txt > train_cv_5.txt
cat train_1.txt train_2.txt train_3.txt train_4.txt train_5.txt train_7.txt \
        train_8.txt train_9.txt train_10.txt > train_cv_6.txt
cat train_1.txt train_2.txt train_3.txt train_4.txt train_5.txt train_6.txt \
        train_8.txt train_9.txt train_10.txt > train_cv_7.txt
cat train_1.txt train_2.txt train_3.txt train_4.txt train_5.txt train_6.txt \
        train_7.txt train_9.txt train_10.txt > train_cv_8.txt
cat train_1.txt train_2.txt train_3.txt train_4.txt train_5.txt train_6.txt \
        train_7.txt train_8.txt train_10.txt > train_cv_9.txt
cat train_1.txt train_2.txt train_3.txt train_4.txt train_5.txt train_6.txt \
        train_7.txt train_8.txt train_9.txt > train_cv_10.txt
```

The following Python code was used to perform the cross-validation procedure.

```
#! /usr/bin/env python

import re
import os
import subprocess

# Various global constants.
d_max = 4
logc_range = 4
cv_range = 10

# Used to store the best parameters found through cross-validation.
e_best = 1
d_best = 0
logc_best = 0
stddev_best = 0

for d in range(1, d_max + 1):
        # Data for average cross-validation plot.
        f = open("results/cv_data_{d}.txt".format(**vars()), "w")
        # Plus one standard deviation.
        fp = open("results/cv_data_{d}_plus.txt".format(**vars()), "w")
```

```
# Minus one standard deviation.
fm = open("results/cv_data_{d}_minus.txt".format(**vars()), "w")

# We exceed the maximum number of iterations if we go past C = 10^3.
for logc in range(-logc_range, min(4, logc_range + 1)):
        print("Running cross-validation for (d, log(C)) = ({d}, {logc}).".
                format(**vars()))
        errors = []
        mean = 0
        stddev = 0

        for k in range(1, cv_range + 1):
                print("Working on {k} out of {cv_range}.".format(**vars()))

                out, err = subprocess.Popen(
                        ["../libsvm-3.17/svm-train",
                        "-s", "0", "-t", "1", "-d", str(d), "-c", str(10**logc),
                        "out/train_cv_{k}.txt".format(**vars()), "tmp/model.txt"],
                        stdout=subprocess.PIPE, stderr=subprocess.PIPE
                ).communicate()

                if len(err) != 0:
                        print("Error occurred while training: {0}".
                                format(err.decode("utf-8")))
                        exit()

                out, err = subprocess.Popen(
                        ["../libsvm-3.17/svm-predict",
                        "out/train_{k}.txt".format(**vars()),
                        "tmp/model.txt", "tmp/output.txt"],
                        stdout=subprocess.PIPE, stderr=subprocess.PIPE
                ).communicate()

                if len(err) != 0:
                        print("Error occurred while testing: {0}".
                                format(err.decode("utf-8")))
                        exit()

                # Parse the test error from the output.
                m = re.findall("[0-9.]+%", out.decode("utf-8"))[0]
                # Strip out the percent sign and parse the value.
                errors.append(1 - float(m[:-1]) / 100)
                mean = mean + errors[k - 1]

        mean = mean / cv_range
        for e in errors:
                stddev = stddev + (e - mean)**2
        stddev = (1.0 / (cv_range - 1) * stddev)**(1.0/2)

        print("Average cross-validation error: {0}% (standard deviation {1}%).".
                format(100 * mean, 100 * stddev))
        f.write("{logc}\t{mean}\n".format(**vars()))
        fp.write("{0}\t{1}\n".format(logc, mean + stddev))
        fm.write("{0}\t{1}\n".format(logc, mean - stddev))

        if mean < e_best:
                (e_best, d_best, logc_best, stddev_best) = (mean, d, logc, stddev)

f.close()
fp.close()
fm.close()
```

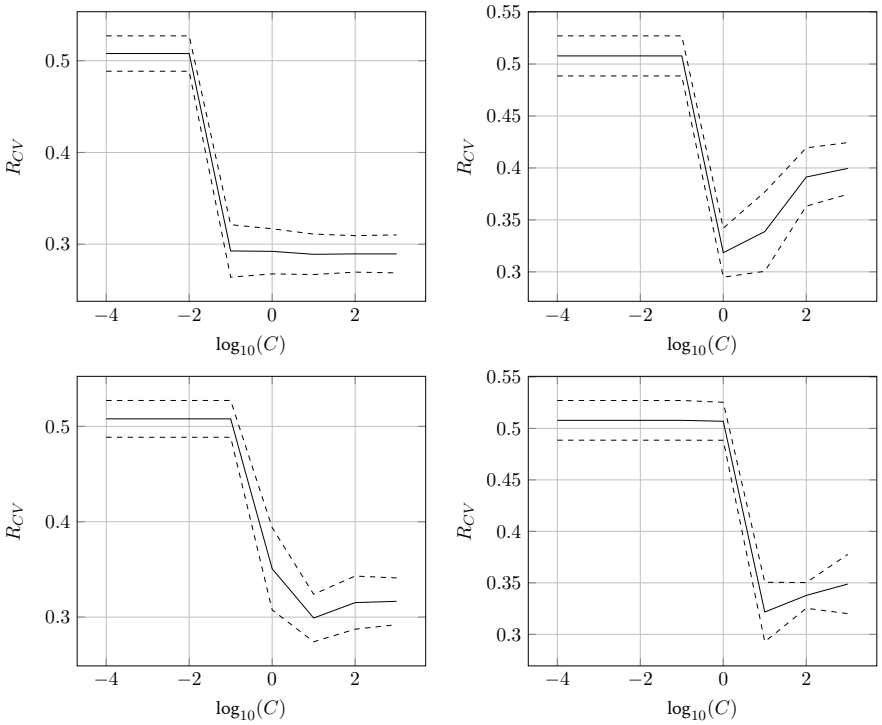

Figure 1: Plots of the average cross-validation error (plus or minus one standard deviation) as a function of C, for $d \in [1, 4]$ (from top to bottom, left to right).

```
print("Best error: {0}% (standard deviation {1}%).".
        format(100 * e_best, 100 * stddev_best))
print("Best parameters: (d, log(C)) = ({d_best}, {logc_best}).".
        format(**vars()))
```

The relevant plots are shown in Figure 1. After running the cross-validation procedure, I determined that $(d^*, C^*) = (1, 10)$.

(d) The following Python code was used to accomplish this.

```
#! /usr/bin/env python

import re
import os
import subprocess

d_max = 4
cv_range = 10
logc = 1

# Data for average cross-validation plot.
f = open("results/test_cv_data.txt", "w")
# Plus one standard deviation.
fp = open("results/test_cv_data_plus.txt", "w")
# Minus one standard deviation.
fm = open("results/test_cv_data_minus.txt", "w")
# Average number of total support vectors.
fs = open("results/test_mean_total_sv.txt", "w")
# Average number of marginal support vectors.
ft = open("results/test_mean_marg_sv.txt", "w")

for d in range(1, d_max + 1):
        print("Running cross-validation for (d, log(C)) = ({d}, {logc}).".
                format(**vars()))
        errors = []
        mean = 0
        stddev = 0
        mean_total_sv = 0
        mean_marg_sv = 0

        for k in range(1, cv_range + 1):
                print("Working on {k} out of {cv_range}.".format(**vars()))

                out, err = subprocess.Popen(
                        ["../libsvm-3.17/svm-train",
                        "-s", "0", "-t", "1", "-d", str(d), "-c", str(10**logc),
                        "out/train_cv_{k}.txt".format(**vars()), "tmp/model.txt"],
                        stdout=subprocess.PIPE, stderr=subprocess.PIPE
                ).communicate()

                # Compute the number of support vectors.
                out = out.decode("utf-8")
                idx = out.index("nSV")
                out = out[idx:]
                m = re.findall("[0-9]+", out)
                mean_total_sv = mean_total_sv + int(m[0])
                mean_marg_sv = mean_marg_sv + int(m[0]) - int(m[1])
                assert(int(m[0]) > int(m[1]))

                if len(err) != 0:
                        print("Error occurred while training: {0}".
                                format(err.decode("utf-8")))
                        exit()

                out, err = subprocess.Popen(
                        ["../libsvm-3.17/svm-predict",
                        "out/splice_noise_test_scaled.txt",
                        "tmp/model.txt", "tmp/output.txt"],
                        stdout=subprocess.PIPE, stderr=subprocess.PIPE
                ).communicate()
```

```
                if len(err) != 0:
                        print("Error occurred while testing: {0}".
                                format(err.decode("utf-8")))
                        exit()

                # Parse the test error from the output.
                m = re.findall("[0-9.]+%", out.decode("utf-8"))[0]
                # Strip out the percent sign and parse the value.
                errors.append(1 - float(m[:-1]) / 100)
                mean = mean + errors[k - 1]

        mean = mean / cv_range
        mean_total_sv = mean_total_sv / cv_range
        mean_marg_sv = mean_marg_sv / cv_range
        for e in errors:
                stddev = stddev + (e - mean)**2
        stddev = (1.0 / (cv_range - 1) * stddev)**(1.0/2)

        print("Average cross-validation error: {0}% (standard deviation {1}%).".
                format(100 * mean, 100 * stddev))
        f.write("{d}\t{mean}\n".format(**vars()))
        fp.write("{0}\t{1}\n".format(d, mean + stddev))
        fm.write("{0}\t{1}\n".format(d, mean - stddev))
        fs.write("{d}\t{mean_total_sv}\n".format(**vars()))
        ft.write("{d}\t{mean_marg_sv}\n".format(**vars()))

f.close()
fp.close()
fm.close()
fs.close()
ft.close()
```

The relevant plots are shown in Figure 2. The average number of marginal support vectors for each value of d is tabulated in Figure 3.

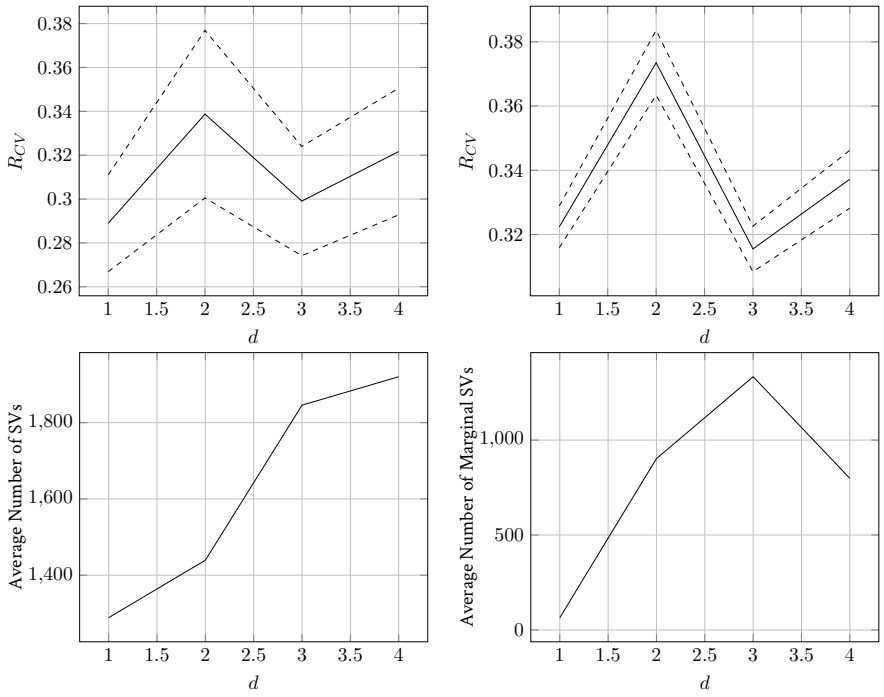

Figure 2: From top to bottom, left to right: plots of the average cross-validation error, average test error, average number of total support vectors, and average number of marginal support vectors.

d	Average Number of Marginal SVs
1	64.6
2	902.7
3	1333.7
4	798.6

Figure 3: The average number of marginal support vectors corresponding to each value of d for which ten-fold cross-validation was performed.

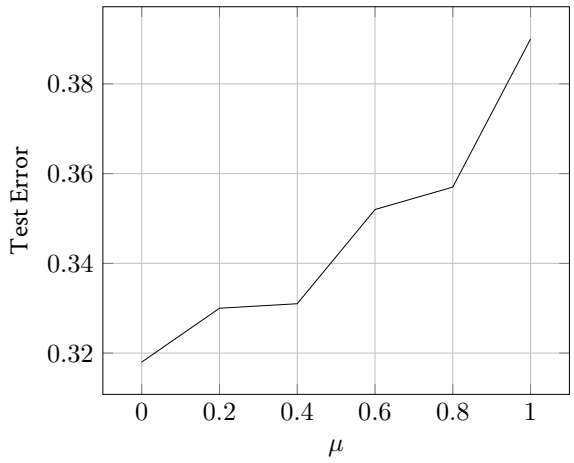

Figure 4: Plot of the test error as a function of μ.

(e) The following Python code was used to accomplish this.

```
#! /usr/bin/env python

import os
import sys

base_dir = os.path.abspath("../libsvm-3.17/python")
sys.path.append(base_dir)

from svmutil import *

tr_y, tr_x = svm_read_problem("out/splice_noise_train_scaled_shuffled.txt")
ts_y, ts_x = svm_read_problem("out/splice_noise_test_scaled.txt")
m = svm_train(tr_y, tr_x, "-s 0 -t 1 -d 1 -c 10")
#label, acc, val = svm_predict(ts_y, ts_x, m)

outliers = []

for i in range(0, len(m.get_sv_coef())):
        if m.get_sv_coef()[i][0] == 10:
                outliers.append(m.get_sv_indices()[i])

f = open("results/outlier_test.txt", "w")

for mu in [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]:
        print("Working on mu = {mu}.".format(**vars()))
        y = [tr_y[i] for i in range(len(tr_y)) if (i + 1) not in
                outliers[: round(mu * len(outliers))]]
        x = [tr_x[i] for i in range(len(tr_x)) if (i + 1) not in
                outliers[: round(mu * len(outliers))]]
        m = svm_train(y, x, "-s 0 -t 1 -d 1 -c 10")
        label, acc, val = svm_predict(ts_y, ts_x, m)
        f.write("{0}\t{1}\n".format(mu, 1 - acc[0] / 100))

f.close()
```

The relevant plot is given in Figure 4.