Mehryar Mohri
Foundations of Machine Learning 2020
Courant Institute of Mathematical Sciences
Homework assignment 2
Oct 08, 2020
Due: Oct 28, 2020 [before class starts].

A. Rademacher complexity

1. Non-negativity of empirical Rademacher complexity: Show that for any hypothesis set $\mathcal{H}$ and sample S, we have $\widehat{\mathfrak{R}}_S(\mathcal{H}) \geq 0$.

   Solution: By the sub-additivity of supremum, we can write:

$$\begin{aligned}\widehat{\mathfrak{R}}_S(\mathcal{H}) &= \frac{1}{m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\sigma_i h(x_i)\right]\\ &\geq \frac{1}{m}\sup_{h\in\mathcal{H}}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sum_{i=1}^{m}\sigma_i h(x_i)\right]\\ &= \frac{1}{m}\sup_{h\in\mathcal{H}}\sum_{i=1}^{m}\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\sigma_i\right]h(x_i) = 0.\end{aligned}$$

   □

2. Empirical Rademacher complexity of products: let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hypothesis sets of functions mappiing from the input space $\mathcal{X}$ to $\{0,1\}$. Let $\mathcal{H} = \{h_1 h_2 \colon h_1 \in \mathcal{H}_2, h_2 \in \mathcal{H}_2\}$. Show that the following inequality holds:

$$\widehat{\mathfrak{R}}_S(\mathcal{H}) \leq \widehat{\mathfrak{R}}_S(\mathcal{H}_1) + \widehat{\mathfrak{R}}_S(\mathcal{H}_2).$$

   Hint: you could use Talagrand's contraction lemma.

   Solution: Observe that for any $h_1 \in \mathcal{H}_1$ and $h_2 \in \mathcal{H}_2$, we can write $h_1 h_2 = (h_1 + h_2 - 1)1_{h_1+h_2-1\geq 0} = (h_1 + h_2 - 1)_+$. Since $x \mapsto (x-1)_+$ is 1-Lipschitz over $[0,2]$, by Talagrand's lemma, the following holds: $\widehat{\mathfrak{R}}_S(\mathcal{H}) \leq \widehat{\mathfrak{R}}_S(\mathcal{H}_1 + \mathcal{H}_2) \leq \widehat{\mathfrak{R}}_S(\mathcal{H}_1) + \widehat{\mathfrak{R}}_S(\mathcal{H}_2)$. □

B. VC-dimension of neural networks

Let C be a concept class over $\mathbb{R}^r$ with VC-dimension d. A C-neural network with one intermediate layer is a concept defined over $\mathbb{R}^n$ that can be

represented by a directed acyclic graph such as that of Figure 1, in which the input nodes are those at the bottom and in which each other node is labeled with a concept $c \in C$.

The output of the neural network for a given input vector $(x_1, \ldots, x_n)$ is obtained as follows. First, each of the n input nodes is labeled with the corresponding value $x_i \in \mathbb{R}$. Next, the value at a node u in the higher layer and labeled with c is obtained by applying c to the values of the input nodes admitting an edge ending in u. Note that since c takes values in $\{0, 1\}$, the value at u is in $\{0, 1\}$. The value at the top or output node is obtained similarly by applying the corresponding concept to the values of the nodes admitting an edge to the output node.

1. Let H denote the set of all neural networks defined as above with $k \geq 2$ internal nodes. Show that the growth function $\Pi_H(m)$ can be upper bounded in terms of the product of the growth functions of the hypothesis sets defined at each intermediate layer.

   Solution: Let $\Pi_u(m)$ denote the growth function at a node u in the intermediate layer. For a fixed set of values at the intermediate layer, using the concept class C the output node can generate at most $\Pi_C(m)$ distinct labelings. There are $\prod_u \Pi_u(m)$ possible sets of values at the intermediate layer since, by definition, for a sample of size m, at most $\Pi_u(m)$ distinct values are possible at each u. Thus, at most $\Pi_C(m) \times \prod_u \Pi_u(m)$ labelings can be generated by the neural network and $\Pi_H(m) \leq \Pi_C(m) \prod_u \Pi_u(m)$. □

2. Use that to upper bound the VC-dimension of the C-neural networks (*Hint*: you can use the implication $m = 2x \log_2(xy) \Rightarrow m > x \log_2(ym)$ valid for $m \geq 1$, and $x, y > 0$ with $xy > 4$).

   Solution: For any intermediate node u, $\Pi_u(m) = \Pi_C(m)$. Thus, for $\tilde{k} = k + 1$, $\Pi_H(m) \leq \Pi_C(m)^{\tilde{k}}$. By Sauer's lemma, $\Pi_C(m) \leq \left(\frac{em}{d}\right)^d$, thus $\Pi_H(m) \leq \left(\frac{em}{d}\right)^{d\tilde{k}}$. Let $m = 2\tilde{k}d \log_2(e\tilde{k})$. In view of the inequality given by the hint and $e\tilde{k} > 4$, this implies $m > d\tilde{k} \log_2\left(\frac{em}{d}\right)$, that is $2^m > \left(\frac{em}{d}\right)^{d\tilde{k}}$. Thus, the VC-dimension of H is less than

   $$2\tilde{k}d \log_2(e\tilde{k}) = 2(k+1)d \log_2(e(k+1)).$$

   □

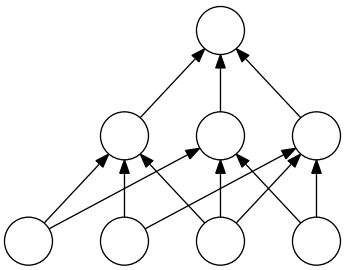

Figure 1: A neural network with one intermediate layer.

3. Let C be the family of concept classes defined by threshold functions $C = \{\text{sgn}(\sum_{j=1}^{r} w_j x_j) \colon \mathbf{w} \in \mathbb{R}^r\}$. Give an upper bound on the VC-dimension of H in terms of k and r.

   Solution: For threshold functions, the VC-dimension of C is r, thus, the VC-dimension of H is upper bounded by

   $$2(k+1)r \log_2(e(k+1)).$$

□

C. Support Vector Machines (SVMs)

1. Download and install the `libsvm` software library from:

   `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

2. Download the `abalone` data set:

   `http://archive.ics.uci.edu/ml/datasets/Abalone`

   Use the `libsvm` scaling tool to scale the features of all the data. Use the first 3133 examples for training, the last 1044 for testing. The scaling parameters should be computed only on the training data and then applied to the test data.

   Solution:

```
gunzip libsvm-3.0.tar.gz
tar xvf libsvm-3.0.tar
cd libsvm-3.0
make
```

□

3. Download the `abalone` data set:

   `http://archive.ics.uci.edu/ml/datasets/Abalone`

   Use the `libsvm` scaling tool to scale the features of all the data. Use the first 3133 examples for training, the last 1044 for testing. The scaling parameters should be computed only on the training data and then applied to the test data.

   Solution:

```
# Rewrite the dataset in a libsvm compatible format,
# skipping the first feature which is the Abalone's age,
# and turning the Abalone's sex in 3 binary features.
awk -F, '
BEGIN {
  sex["M"] = "1:1 2:0 3:0";
  sex["F"] = "1:0 2:1 3:0";
  sex["I"] = "1:0 2:0 3:1";
}
{
  class = ($NF <= 9) ? -1 : 1;
  printf("%d %s", class, sex[$1]);
  for (i = 2; i <= NF - 1; ++i)
    printf(" %d:%s", i + 2, $i);
  printf("\n");
}' data/abalone.data.txt > output/dataset.txt

# Split in training and test set.
awk 'NR <= 3133 { print; }' data/dataset.txt \
  > output/train.txt
awk 'NR > 3133 { print; }' data/dataset.txt \
  > output/test.txt

# Scale training and test set.
libsvm-3.0/svm-scale -s output/scale.txt \
  output/train.txt > output/train.scaled.txt
libsvm-3.0/svm-scale -r output/scale.txt \
  output/test.txt > output/test.scaled.txt
```

□

4. Consider the binary classification that consists of distinguishing classes 1 through 9 from the rest. Use SVMs combined with polynomial kernels to tackle this binary classification 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 2, starting from a small value $C = 2^{-k}$ to $C = 2^k$, 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.

Solution:

```
# Split the training data into ten equal-sized disjoint sets:
sort -R output/train.scaled.txt \
  > output/train.scaled.shuffled.txt

NUM_SAMPLES=$(cat output/train.scaled.shuffled.txt \
  | wc --lines)
for SPLIT in $(seq 1 10); do
  awk "NR < $NUM_SAMPLES * ($SPLIT - 1) / 10.0 \
    || NR >= $NUM_SAMPLES * $SPLIT / 10.0" \
      output/train.scaled.shuffled.txt \
      > output/train.$SPLIT.txt

  awk "NR >= $NUM_SAMPLES * ($SPLIT - 1) / 10.0 \
    && NR < $NUM_SAMPLES * $SPLIT / 10.0" \
      output/train.scaled.shuffled.txt \
      > output/dev.$SPLIT.txt
done

# Compute accuracy for different values of $C$ and
# for degrees 1 through 4 of the polynomial kernel.
for LOG2C in $(seq -10 10); do
  for DEGREE in 1 2 3 4; do
    for SPLIT in $(seq 1 10); do
      # Train models.
      C=$(python -c "print 2 ** $LOG2C")
      echo "c="$C "d="$DEGREE "split="$SPLIT

      libsvm-3.0/svm-train -t 1 -d $DEGREE -c $C \
        output/train.$SPLIT.txt \
        output/model.$LOG2C.$DEGREE.$SPLIT.txt \
        > output/train.$LOG2C.$DEGREE.$SPLIT.log.txt

      libsvm-3.0/svm-predict output/dev.1.txt \
        output/model.$LOG2C.$DEGREE.$SPLIT.txt \
        output/dev.$LOG2C.$DEGREE.$SPLIT.prediction.txt \
```

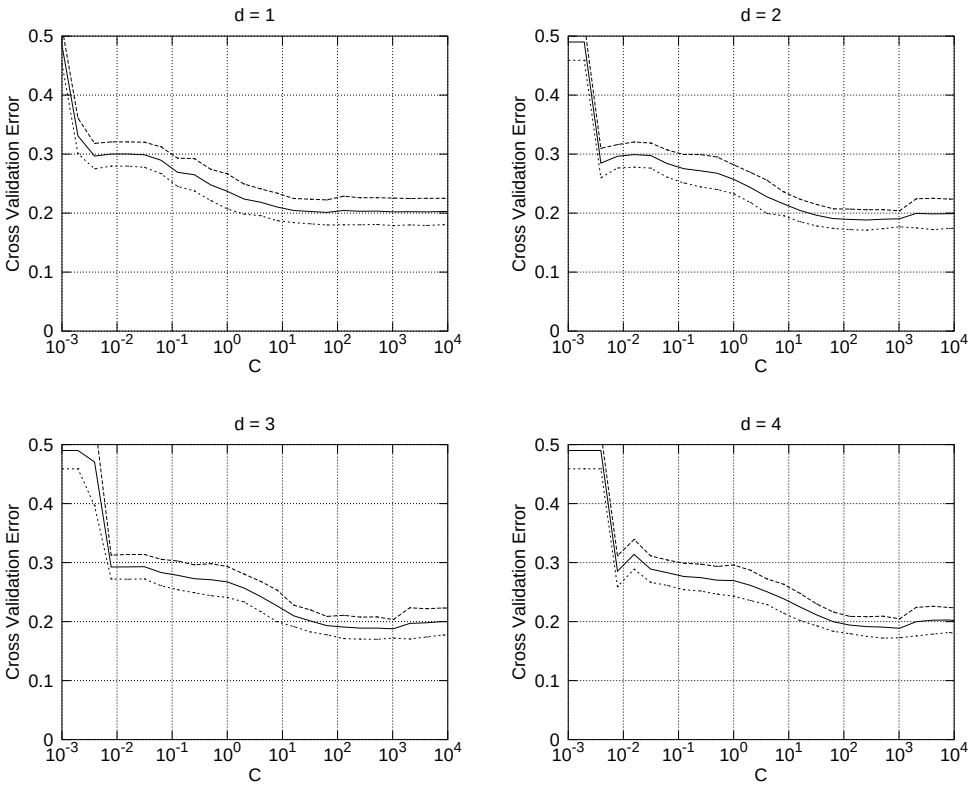

Figure 2: Mean cross validation error $\pm$ 1 standard deviation for different degrees d of the polynomial kernel, as a function of C.

```
        > output/dev.$LOG2C.$DEGREE.$SPLIT.log.txt
    done
  done
done

# Compute mean and standard deviation
# of classification accuracy.
echo -n > output/dev.results.txt
for F in output/dev.*.log.txt; do
  echo $F $(cat $F) | \
    sed 's;.*\.\(.*\)\.\(.*\)\.\(.*\)\.log.* = \(.*\)%.*;\1 \2 \3 \4;' \
    | grep -v 'classification' \
    >> output/dev.results.txt;
done

awk '{
       acc = $4 / 100;
       accuracy_mean[$1" "$2] += acc / 10;
       accuracy_var[$1" "$2] += acc ^ 2 / (10 - 1);
     }
     END {
       for (cond in accuracy_mean) {
         mean = accuracy_mean[cond];
         std = sqrt(accuracy_var[cond] - mean ^ 2 * 10 / (10 - 1));
         print cond, mean, std;
       }
     }' output/dev.results.txt \
       | sort -n -k 3 > output/dev.results.summary.txt
```

Plots of the results are shown in Figure 2. The best C and d from cross-validation are $(C^*, d^*) = (1024, 3)$. □

5. 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?

   Solution:

   Plots for this question are shown in Figure 3. The number of support vector on margin hyperplanes are obtained as $nSV - nBSV$ (number of support vectors - number of bounded support vectors). They are respectively 9, 26, 43, 48 for polynomial kernel degrees 1 through 4. □

6. In class, we gave two types of argument in favor of the SVMs algorithm: one based on the sparsity of the support vectors, another based on the

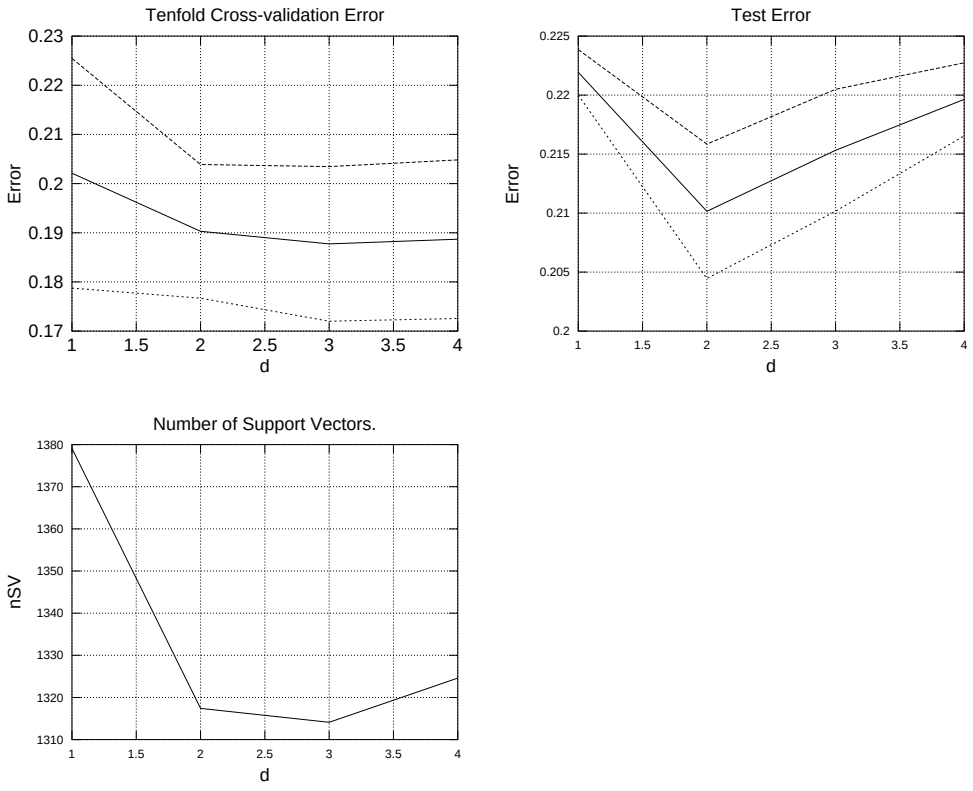

Figure 3: Tenfold cross-validation and test error, and average number of support vectors as a function of the degree d of the polynomial kernel.

notion of margin. Suppose that instead of maximizing the margin, we choose instead to maximize sparsity by minimizing the norm p of the vector $\boldsymbol{\alpha}$ that defines the weight vector $\mathbf{w}$, for some $p \geq 1$. For simplicity, fix $p = 2$. This gives the following optimization problem for a kernel function K:

$$\begin{aligned} &\min_{\alpha,b} \frac{1}{2}\sum_{i=1}^{m}\alpha_i^2 + C\sum_{i=1}^{m}\xi_i \qquad (1)\\ &\text{subject to } y_i\Big(\sum_{j=1}^{m}\alpha_j y_j K(x_i,x_j) + b\Big) \geq 1-\xi_i, i \in [1,m]\\ &\qquad \xi_i, \alpha_i \geq 0, i \in [1,m]. \end{aligned}$$

(a) Show that modulo the non-negativity constraint on $\boldsymbol{\alpha}$, the problem coincides with an instance of the primal optimization problem of SVMs (indicate exactly how to view it as such).

Solution: Let

$$x_i' = \left(y_1 K(x_i,x_1), \ldots, y_m K(x_i,x_m)\right).$$

Then the optimization problem becomes

$$\begin{aligned} \min_{\alpha,b,\xi}\ & \frac{1}{2}||\alpha||^2 + C\sum_{i=1}^{m}\xi_i\\ \text{subject to }\ & y_i\left(\alpha \cdot x_i' + b\right) \geq 1-\xi\\ & \xi_i, \alpha_i \geq 0, i \in [1,m] \end{aligned}$$

which is the standard formulation of the primal SVM optimization problem on samples x_i', modulo the non-negativity constraints on α_i. □

(b) Is the positive-definiteness of the kernel function K needed to ensure that this is a convex optimization problem? Justify your response.

Solution: Modulo the non-negativity constraints on α_i, the optimization problem in (a) is known to be convex for any set of samples x_i'. Restricting the domain of the problem with additional convex constraints $\alpha_i \geq 0$ preserves convexity, so the problem is convex regardless of the positive-definiteness of K. □

(c) Derive the dual optimization of problem of (1).

Solution: The Lagrangian of (1) for all $\alpha_i \geq 0, \xi_i \geq 0, b, \alpha'_i \geq 0, \beta_i \geq 0, \gamma_i \geq 0, i \in [1, m]$ is

$$L = \frac{1}{2}||\alpha||^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha'_i(y_i(\alpha^T x'_i + b) - 1 + \xi_i) - \sum_{i=1}^{m}\beta_i\xi_i - \sum_{i=1}^{m}\gamma_i\alpha_i$$

and the KKT conditions are

$$\begin{aligned}
\nabla_\alpha L = 0 &\quad\Leftrightarrow\quad \alpha = \sum_{i=1}^{m}\alpha'_i y_i x'_i + \gamma \\
\nabla_b L = 0 &\quad\Leftrightarrow\quad \sum_{i=1}^{m}\alpha'_i y_i = 0 \\
\nabla_{\xi_i} L = 0 &\quad\Leftrightarrow\quad \alpha'_i + \beta_i = C
\end{aligned}$$

and

$$\begin{aligned}
\alpha'_i(y_i(\alpha^T x'_i + b) - 1 - \xi_i) = 0 \\
\beta_i\xi_i = 0 \\
\gamma_i\alpha_i = 0.
\end{aligned}$$

Using the KKT conditions on L we get

$$\begin{aligned}
L &= \frac{1}{2}\left(\sum_{i=1}^{m}\alpha'_i y_i x'_i + \gamma\right)^T\left(\sum_{j=1}^{m}\alpha'_j y_j x'_j + \gamma\right) + C\sum_{i=1}^{m}\xi_i \\
&\quad - \sum_{i=1}^{m}\alpha'_i\left(y_i\left(\left(\sum_{j=1}^{m}\alpha'_j y_j x'_j + \gamma\right)^T x'_i + b\right) - 1 + \xi_i\right) \\
&\quad - \cancel{\sum_{i=1}^{m}\beta_i\xi_i} - \cancel{\sum_{i=1}^{m}\gamma_i\alpha_i} \\
&= -\frac{1}{2}\sum_{i=1}^{m}\alpha'_i y_i x_i'^T\left(\sum_{j=1}^{m}\alpha'_j y_j x'_j + \gamma\right) + \cancel{\frac{1}{2}\gamma^T\alpha} \\
&\quad + \sum_{i=1}^{m}C\xi_i - \alpha'_i\left(y_i b - 1 + \xi_i\right) \\
&= \sum_{i=1}^{m}\alpha'_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha'_i\alpha'_j y_i y_j x_i'^T\left(x'_j + \gamma\right) \\
&\quad + \cancel{\sum_{i=1}^{m}(C - \alpha'_i)\xi_i} - \cancel{\sum_{i=1}^{m}\alpha'_i y_i b}.
\end{aligned}$$

Thus the dual optimization problem is

$$\begin{aligned} \max_{\alpha',\gamma} \quad & \sum_{i=1}^{m} \alpha'_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha'_i \alpha'_j y_i y_j x_i'^T \left(x'_j + \gamma\right) \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha'_i y_i = 0 \\ & 0 \leq \alpha'_i \leq C, \gamma_i \geq 0, i \in [1, m] \end{aligned}$$

□

(d) Suppose we omit the non-negativity constraint on $\boldsymbol{\alpha}$. Use `libsvm` to solve the problem. Plot the ten-fold cross-validation training and test errors for the hypotheses obtained based on the solution $\boldsymbol{\alpha}$ as a function of d, for the best value of C measured on the validation set.

Solution: New datasets of samples x'_i for different degrees of the polynomial kernel K can be computed e.g. with the following Matlab code.

```
load dataset.txt

train_set = dataset(1:3133, :);
test_set = dataset(3134:end, :);
train_labels = train_set(:, 1);
train_x = train_set(:, 2:end);
test_labels = test_set(:, 1);
test_x = test_set(:, 2:end);

dim = size(train_x, 2);
for degree = 1:4
    degree
    train_x_prime = ...
        (train_x * (repmat(train_labels, ...
                    [1 dim]) .* train_x)') ...
                    .^ degree + 1;
    train_set_prime = [ train_labels, ...
                        train_x_prime ];
    save(['train.' num2str(degree) '.txt' ], ...
          'train_set_prime');
    test_x_prime = ...
        (test_x * (repmat(train_labels, ...
                   [1 dim]) .* train_x)') ...
                   .^ degree + 1;
    test_set_prime = [ test_labels, ...
                       test_x_prime ];
    save(['test.' num2str(degree) '.txt'], ...
```

```
                        'test_set_prime');
end
\end{lstlisting}
```

Training, cross-validation and testing can then be conducted exactly as in question 3. of this problem. The best value of C from cross-validation is $C^* = 1024$. Resulting error rates are shown in Figure 4.

□

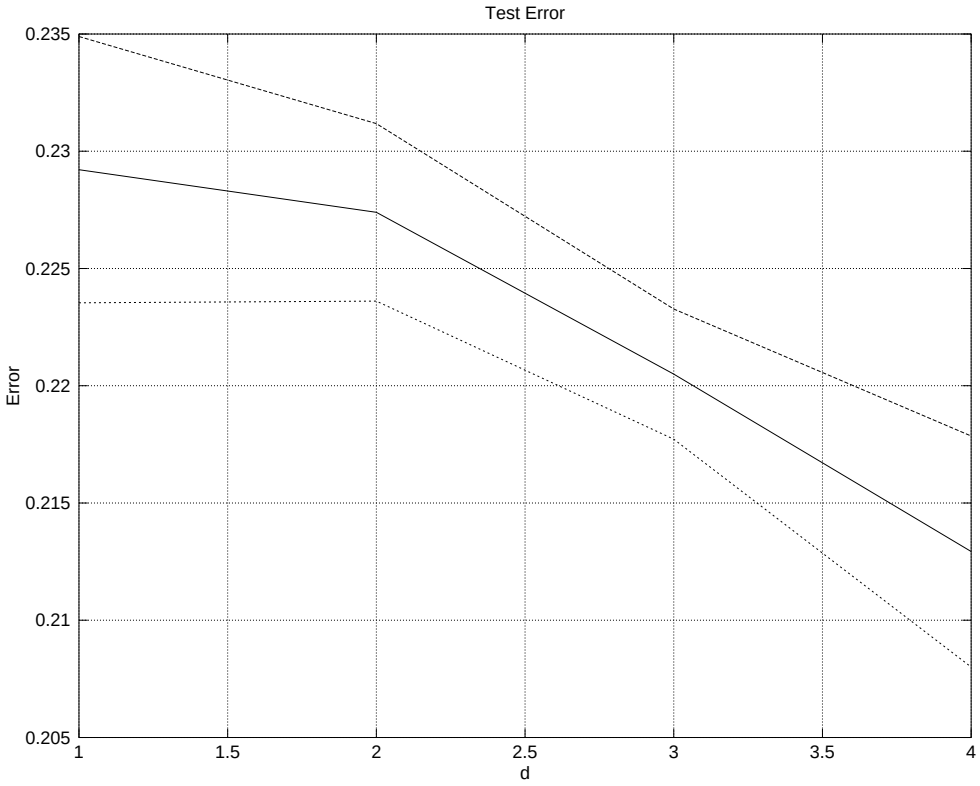

Figure 4: Test error $\pm$ one standard deviation for $C^* = 1024$ as a function of the degree d of the polynomial kernel, where training is done by solving the optimization problem described in question 5.