Introduction to Learning and Analysis of Big Data HW 2

Tomer Elgavish, Shay Kricheli

December 2019

1 Question 3

1.1 Section a

The following plot shows the training error and the test error as a function of the training sample size, for both the 3/5 and the 5/6 learning tasks.

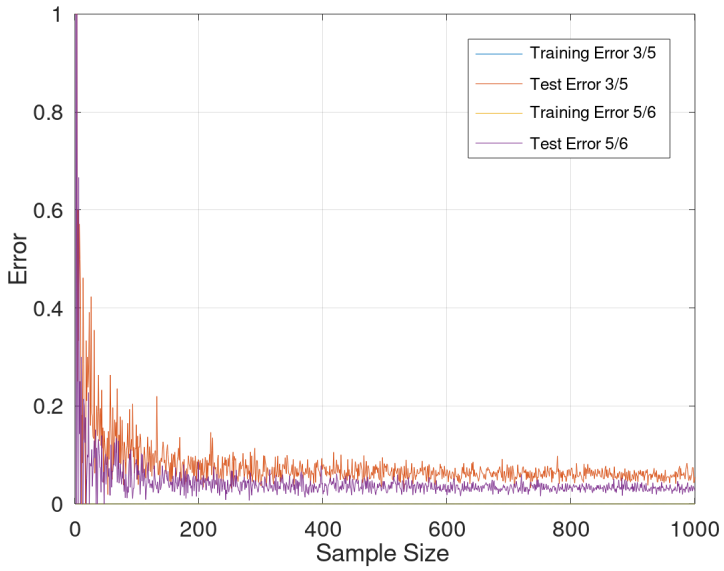

Figure 1: Q3.a - Errors vs. Training Sample Size

We took the *maxupdates* parameter to be $5 \cdot m$ where m is the current iteration's training sample size.

1.2 Section b

One can see that the training errors for both 3/5 and 5/6 learning tasks are always at 0, which makes sense at cases where the training set is separable - since we are training the perceptron on that exact training set, and if we assume that the training set is separable then we have a guarantee that the perceptron algorithm will come to halt - which means by its definition that it correctly predicts the label of each training example. In cases where the training set isn't separable (that didn't occur in our simulation) - the perceptron algorithm would not come to halt by itself, but in our case would due to the max-iterations parameter. In those cases, as we saw in class, even though the perceptron

algorithm will do its best to predict the examples correctly, the error on the training set can never be zero. In general, an increase in the training set size will increase the probability of it not being separable so we can, in the general case expect the training set error to be non-zero in larger training sample set sizes.

1.3 Section c

One can see in general that trends for both test errors decrease with the training sample size. That makes sense since an increase in the training sample set size inputs more examples to train on and therefore makes the prediction of the perceptron more accurate on new examples in the test set.

1.4 Section d

Since we got 0 training set error on all sample sizes, the overfitting is (approximately):

$$overfitting := err(\hat{h}_S, \mathcal{D}) - err(\hat{h}_S, S) \approx err(\hat{h}_S, \mathcal{D})$$

That means the overfitting of our predictor behaves approximately as the error of our predictor on the distribution. Since the test sample is chosen from the distribution - the error on the test sample is induced from the error on the distribution and thus the error on test sample also behaves as the error of our predictor on the distribution. Since in section c we saw that the trends of the test errors are decreasing, then the general trend of the overfitting is also decreasing. As we saw in class, there is (usually) a positive correlation between the overfitting of the model and its estimation error. Thus, we can conclude that the estimation error also has a decreasing trend.

1.5 Section e

One can notice that the error on the 3/5 task is practically always greater than that of the 5/6 task. From that we can infer that the algorithm distinguishes between 5 and 6 better than 3 and 5. Thus we can conclude that, statistically speaking, the error of best prediction rule for the 5/6 task will be lower than that of the 3/5 task. Since the approximation error is defined as:

$$approximation - error := \inf_{h \in \mathcal{H}} err(h, \mathcal{D})$$

we can conclude that the the approximation error for the 5/6 task will be lower than that of the 3/5 task.

1.6 Section f

For a single run of the perceptron on a sample of size 1000 for distinguishing 5 and 6, the following plot displays the test error after each iteration, as a function of the number of updates:

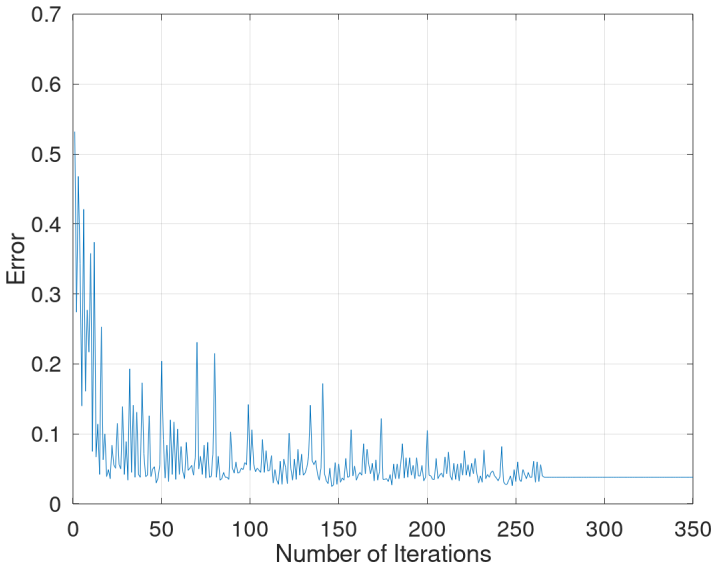

Figure 2: Q3.f - Errors vs. Number of Updates

As the plot shows, the algorithm converges after about 260 updates. Based on that observation, we would update our rule for the *maxupdates* parameter to be $m/4$ where m is the sample size.

2 Question 4

In this question we are asked to run our implementation of the soft-SVM algorithm that was done in question 2 on the MNIST data set.

2.1 Section a

In this section we are asked to run the implementation of the soft-SVM on various samples of size 100 with examples only of digits 3 and 5 for 17 values of λ when: $\lambda \in \{3^n \mid n \in \{-4, -3, \cdots 12\}\}$. The following plot displays the average test and train errors as a function of λ:

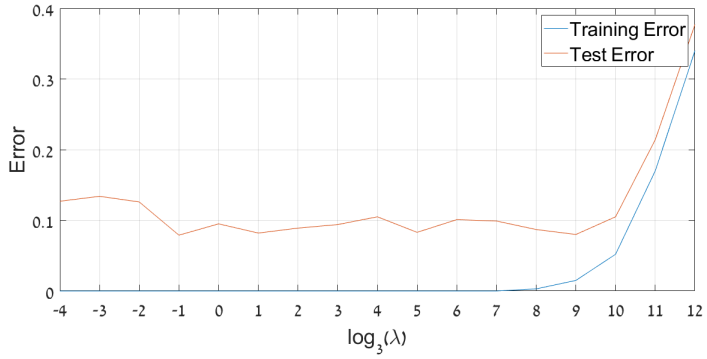

Figure 3: Q4.a - Errors vs. λ

2.2 Section b

In this question we are asked what trend is expected in the training error as a function of λ and whether or not we see it in our plot. Let us consider the definition of the Soft-SVM minimization problem:

$$\min_{w \in \mathbb{R}^n} \left[\lambda ||w||^2 + \sum_{i=1}^{m} \ell_h(w, (x_i, y_i)) \right]$$

where $\ell_h(w, (x_i, y_i)) = \max\{0, 1 - y_i \langle w, x_i \rangle\}$. From that - for smaller values of λ, the algorithm enables larger values for the norm of w and therefore it encourages smaller values for the hinge-loss - which is a a measure of the error on the training set. Therefore for smaller values of λ we expect less training error. This correlates well with the result we got in the figure. When increasing λ the algorithm encourages smaller values for the norm of w and thus increasing the margin of the linear separator. This causes the algorithm to give a lower priority to minimizing the hinge loss and thus for larger values of λ - the algorithm allows for more misclassifications on the training sample and thus we expect a larger training error - which also correlates to the figure.

2.3 Section c

As mentioned in the previous section - for smaller of λ the algorithm puts more emphasis on minimizing the hinge-loss and therefore may cause overfitting for very small values of λ. This can be seen in our figure as for very small values of λ we get relatively larger test error values as opposed to the intermediate values of λ where we get smaller test error values. For very large values of λ, the algorithm puts almost all of its effort on minimizing the norm of w - which effectively causes w to approach the vector that has the minimal norm such the constraints still hold. This causes the vector

w to not generalize well and therefore renders larger errors on random test sets, as can be seen in the figure.

2.4 Section d

Based on the results we got, we would choose an intermediate value of λ which is not very small or very large - such as $\lambda^* = 3^3 = 27$. We can see in the figure that this value is well in the trade off between low overfitting and large margin.

3 Question 5

Let $d \in \mathbb{N}$ the number of features for our examples (the dimension) and let $\mathcal{X} = \mathbb{R}^d$ - the set of all possible examples and $\mathcal{Y} = \{-1, 1\}$ - the set of all possible labels defined as shown in the exercise description. Moreover - let $S = \{(x_1, y_1), \cdots (x_m, y_m)\}$ where $m = d$ be our sample set. Let us observe that in this special case detailed, it holds that:

$$y_i x_i = (-1)^{i+1} \cdot \left((-1)^i, (-1)^i, \cdots (-1)^i, (-1)^{i+1}, 0, 0, \cdots 0\right) =$$
$$= (\underbrace{-1, -1, \cdots, -1}_{\text{i-1 elements}}, 1, \underbrace{0, 0, \cdots 0}_{\text{d-i elements}})$$

3.1 Section a

In this section we are to show that for any iteration t of the perceptron on the given sample S as described, and for any $1 \leq j \leq d$: $|w^{(t+1)}(j)| \leq t$. We'll use induction on t - the number of iterations.

- Base case: According to the algorithm's initialization - at the beginning $t = 1$. Moreover, the algorithm initializes w with $w^{(1)} = (0, \cdots, 0)$. Let us calculate $w^{(2)}$: Since $w^{(1)} = (0, \cdots, 0)$, $y_i \langle w^{(1)}, x_i \rangle = 0 \; ; \; \forall i \in \{1, \cdots, m\} = \{1, \cdots, d\}$. That means that the condition in the perceptron algorithm holds. Let $i \in \{1, \cdots, d\}$ be the index of the example that the algorithm chose first out of all the examples. Then:

  $$w^{(2)} = w^{(1)} + y_i x_i = (0, \cdots, 0) + y_i x_i = (-1, -1, \cdots, -1, 1, 0, 0, \cdots .0)$$

  As one can see for any $j \leq d$: $|w^{(2)}(j)| \leq t = 1$.

- Induction assumption: Let $n \in \mathbb{N}$. Let us assume that for $t = n$, it holds that: for any $j \leq d$: $|w^{(t+1)}(j)| \leq t \longrightarrow |w^{(n+1)}(j)| \leq n$.

- Induction step: Let $t = n + 1$. We'll show that for any $j \leq d$: $|w^{(t+1)}(j)| \leq t$ which means: $|w^{(n+2)}(j)| \leq n + 1$:

  $$|w^{(n+2)}(j)| = |w^{(n+1)}(j) + y_i x_i(j)| \overset{1}{\leq} |w^{(n+1)}(j)| + |y_i x_i(j)| \overset{2}{\leq}$$
  $$\overset{2}{\leq} n + |y_i x_i(j)| = n + |(-1, -1, \cdots, -1, 1, 0, 0, \cdots 0)(j)| \overset{3}{\leq} n + 1$$

  Let us explain each transition:

  1. Appliance of the triangle inequality
  2. Appliance of the induction assumption
  3. The absolute value of any coordinate in $y_i x_i$ is at most 1

■

3.2 Section b

Let $w^{(T)}$ be the separator that the perceptron outputs. We are to prove that for every coordinate j: $|w^{(T)}(j)| \geq 2^{j-1}$. We will use complete induction on j to prove an even stronger claim: for every coordinate j: $w^{(T)}(j) \geq 2^{j-1}$. Since the term 2^{j-1} is always positive, that would lead to $|w^{(T)}(j)| \geq 2^{j-1}$. Using the fact that the separator defined by $w^{(T)}$ labels correctly all the examples in the given sample S, we know that for all $i \in \{1, \cdots, d\}$:

$$\langle w^{(T)}, y_i x_i \rangle > 0 \longrightarrow (*) \sum_{k=1}^{d} w^{(T)}(k) \cdot y_i x_i(k) > 0$$

- Base case: For $j = 1$ we are to show that $w^{(T)}(1) \geq 2^{1-1} = 2^0 = 1$. Splitting $(*)$ from above to the first term and so on we get:

  $$w^{(T)}(1) \cdot y_i x_i(1) + \Sigma_{k=2}^{d} w^{(T)}(k) \cdot y_i x_i(k) > 0$$

  Since this equation is true for each $i \in \{1, \cdots, d\}$:, it is true in particular, for $i = 1$. By setting $i = 1$ in the vector $y_i x_i$ that we defined above, we get: $y_1 x_1 = (1, 0, 0, \cdots, 0)$. So:

  $$w^{(T)}(1) \cdot y_1 x_1(1) + \sum_{k=2}^{d} w^{(T)}(k) \cdot y_1 x_1(k) = w^{(T)}(1) \cdot 1 + \sum_{k=2}^{d} w^{(T)}(k) \cdot 0 = w^{(T)}(1) > 0$$

  Since all the coordinates in $w^{(T)}$ are whole numbers, we get: $w^{(T)}(1) \geq 1$ as required.

- Induction assumption: Let $n \in \mathbb{N}$ such that $n \leq d$. Let us assume that for any $j < n$: $w^{(T)}(j) \geq 2^{j-1}$.

- Induction step: Let $j = n$. We'll show that $w^{(T)}(j) \geq 2^{j-1}$ which means: $w^{(T)}(n) \geq 2^{n-1}$. Using the definition of $y_i x_i$ let us rewrite $(*)$ in the following manner:

  $$\sum_{k=1}^{d} w^{(T)}(k) \cdot y_i x_i(k) > 0 \longrightarrow \sum_{k=1}^{d} w^{(T)}(k) \cdot (-1, -1, \cdots, -1, 1, 0, 0, \cdots 0)(k) > 0$$
  $$\longrightarrow -\sum_{k=1}^{i-1} w^{(T)}(k) + w^{(T)}(i) + \sum_{k=i+1}^{d} w^{(T)}(k) \cdot 0 > 0 \longrightarrow \; w^{(T)}(i) > \sum_{k=1}^{i-1} w^{(T)}(k)$$

  This, again, holds for any $1 \leq i \leq d$ and since $n \leq d$, for n as well. Therefore for $i = n$:

  $$w^{(T)}(n) > \sum_{k=1}^{n-1} w^{(T)}(k)$$

  Let us observe that the induction assumption holds for each coordinate of w with value $k < n$ in the sum in the right hand side and therefore:

  $$w^{(T)}(n) > \sum_{k=1}^{n-1} w^{(T)}(k) \geq \sum_{k=1}^{n-1} 2^{k-1} = \frac{1}{2} \sum_{k=1}^{n-1} 2^k = \frac{1}{2} \cdot 2 \cdot (2^{n-1} - 1) = 2^{n-1} - 1$$

  Again, since all the coordinates in $w^{(T)}$ are whole numbers, we get: $w^{(T)}(n) \geq 2^{n-1}$ as required. ■

3.3 Section c

Let T be the final number of iterations of the perceptron algorithm. In this section we are to conclude from the previous two sections that: $T = \Omega(2^d)$. From setting $t = T - 1$ in the inequality in section a we get that:

$$\forall 1 \leq j \leq d \ : \ T - 1 \geq |w^{(T)}(j)|$$

From that and from section b we get:

$$\forall 1 \leq j \leq d \ : \ T - 1 \geq |w^{(T)}(j)| \geq 2^{j-1}$$

Setting $j = d$ we get:

$$T - 1 \geq |w^{(T)}(d)| \geq 2^{d-1} \longrightarrow T \geq 2^{d-1} + 1 \longrightarrow T = \Omega(2^d) \ \blacksquare$$

4 Question 6

Let us consider the following problem: Given a training sample $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$ let us define the modified version of the soft-SVM optimization problem:

$$\min_{w \in \mathbb{R}^n} \left[\lambda ||w||_1^2 + \sum_{i=1}^{m} \ell_h(w, (x_i, y_i)) \right]$$

where $\ell_h(w, (x_i, y_i)) = \max\{0, 1 - y_i \langle w, x_i \rangle\}$.

4.1 Section a

In this section, we are asked to write a quadratic minimization problem with constraints that is equivalent to the problem above. To do so, let us denote w_i to be the i'th coordinate of the vector w. We'll define the following additional variables:

$$\theta_i = |w_i| \ ; \forall i \in [1, d]$$

$$\Psi = \sum_{i=1}^{d} \theta_i$$

$$\xi_i = \ell_h(w, (x_i, y_i)) \ ; \forall i \in [1, m]$$

Let us consider the following quadratic minimization problem:

$$\begin{aligned}
\min_{w \in \mathbb{R}^n} \quad & \left[\lambda \Psi^2 + \sum_{i=1}^{m} \xi_i \right] & \\
\text{s.t.} \quad & \theta_i \geq w_i \quad ; & \forall i \in [1, d] \\
& \theta_i \geq -w_i \quad ; & \forall i \in [1, d] \\
& \Psi \geq \sum_{i=1}^{d} \theta_i & \\
& -\Psi \geq -\sum_{i=1}^{d} \theta_i & \\
& \xi_i \geq 0 \quad ; & \forall i \in [1, m] \\
& y_i \langle w, x_i \rangle \geq 1 - \xi_i \quad ; & \forall i \in [1, m]
\end{aligned}$$

4.2 Section b

In this section, we are asked to write what are H, u, A, v in the definition of a Quadratic Program should be set to so as to solve the minimization problem we wrote in the previous section. Let us define the following modified vector:

$$z = (\Psi, w_1, w_2, \cdots, w_d, \theta_1, \theta_2, \cdots, \theta_d, \xi_1, \xi_2, \cdots, \xi_m)^T \in \mathbb{R}^{(1+2d+m)\times 1}$$

Let us denote $\alpha_{r\times s}$ as a block matrix of dimensions $r \times s$ that has all elements set to α. Moreover, let us denote:

$$Y_i X_i = \begin{bmatrix} y_1 x_1(1) & \cdots & y_1 x_1(d) \\ \vdots & & \vdots \\ y_m x_m(1) & \cdots & y_m x_m(d) \end{bmatrix} \in \mathbb{R}^{m\times d}$$

Now, let us define:

$$H = \begin{bmatrix} 2\lambda & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{(1+2d+m)\times(1+2d+m)}$$

$$u = (\underbrace{0, \cdots 0}_{\text{2d+1}}, \underbrace{1, \cdots 1}_{\text{m}})^T \in \mathbb{R}^{(1+2d+m)\times 1}$$

$$v = (\underbrace{0, \cdots 0}_{\text{2d+2+m}}, \underbrace{1, \cdots 1}_{\text{m}})^T \in \mathbb{R}^{(2d+2+2m)\times 1}$$

$$A = \begin{bmatrix} 0_{d\times 1} & -I_d & I_d & 0_{d\times m} \\ 0_{d\times 1} & I_d & I_d & 0_{d\times m} \\ 1 & 0_{1\times d} & (-1)_{1\times d} & 0_{1\times m} \\ -1 & 0_{1\times d} & 1_{1\times d} & 0_{1\times m} \\ 0_{m\times 1} & 0_{m\times d} & 0_{m\times d} & I_m \\ 0_{m\times 1} & Y_i X_i & 0_{m\times d} & I_m \end{bmatrix} \in \mathbb{R}^{(2d+2+2m)\times(2d+m+1)}$$