Data Mining: Learning from Large Data Sets

Final exam

Feb 2, 2016

Time limit:	120 minutes
Number of pages:	18
Total points:	100

You can use the back of the pages if you run out of space. Collaboration on the exam is strictly forbidden. Please show *all* of your work and always *justify* your answers.

Please write your answers with a *pen*.

(1 point) Please fill in your name and student ID.

Please leave the table below empty.

Problem	Maximum points	Obtained
1.	13	
2.	13	
3.	13	
4.	19	
5.	16	
6.	25	
Total	100	

1. MapReduce (13 points)

We consider a system that spawns a new process for each request it has to serve. These processes are expected to run for 5 time units before finishing. For monitoring purposes, the system logs in every time unit the state of each process still in the system. The log is stored in a file, and each line is of the form "t p c", where $t \in \mathbb{N}$ is a timestamp, $p \in \mathbb{N}$ is a unique process identifier, and $c \in \{\texttt{START}, \texttt{RUNNING}, \texttt{FINISHED}\}$ describes the process state. To keep the code readable, in your solutions you can refer to these states as `S`, `R` and `F` respectively.

Below you can see an example of a (very short) log file.

```
0 123 START
1 123 RUNNING
1 55 START
2 123 RUNNING
2 55 RUNNING
3 55 RUNNING
3 123 RUNNING
4 123 RUNNING
4 55 RUNNING
5 123 FINISHED
5 55 RUNNING
6 55 FINISHED
```

Note that in any time unit several processes can run in parallel.

(3 points) (a) Write a MapReduce program that, given a log file, outputs all numbers t for which exactly one line of the form "t p `RUNNING`" occurs in the log. In particular, when given the log file above, the program should output `1` and `5`.

Note: The space for your answer is on the next page..

```
def map(key, value):
  # Arguments: * key   : line identifier, can be ignored
  #            * value : a string corresponding to a line from the log
  # Use emit(k, v) to emit a key-value pair (k, v).
```

```
def reduce(key, values):
  # Use emit(t, 1) to indicate that t satisfies the above condition.
```

(10 points) (b) We now want to check if these processes are actually behaving as expected. A process p is said to be *compliant* if whenever a line of the form "t p `START`" occurs with $t, p \in \mathbb{N}$, then

- a line of the form "$t + 5$ p `FINISHED`" occurs, and
- for $i = 0, \ldots, 4$, a line of the form "$t + i$ p `RUNNING`" occurs.

Observe that the log file presented above is compliant. Write a MapReduce program that finds all non-compliant processes.

Note: It is fine if you report the same process as non-compliant multiple times, but you should not report compliant processes as non-compliant.

```
def map(key, value):
  # Arguments: * key   : line identifier, can be ignored
  #            * value : a string corresponding to a line from the log
  # Use emit(k, v) to emit a key-value pair (k, v).
```

```
def reduce(key, values):
  # Use emit(p, 1) to indicate that process p is non-compliant.
```

2. Locality-sensitive functions (13 points)

A *trit* is a value in $\{0,1,2\}$ and a *tryte* is a sequence of 8 trits. For a tryte A, let $A[i]$, for $0 \le i < 8$, denote the i-th trit. The L_1-distance between two trytes A and B is defined as follows:

$$d(A,B) := \sum_{0 \le i < 8} |A[i] - B[i]| \, .$$

For $0 \le i < 8$ and $0 \le j < 2$, we define a function h_j^i, such that for any tryte A,

$$h_j^i(A) := \begin{cases} 1 & \text{if } A[i] > j, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{H} := \{h_j^i : 0 \le i < 8, 0 \le j < 2\}$.

(3 points) (a) Let A and B be two trytes and let h be chosen uniformly at random from $\mathcal{H}$. Compute $Pr_{\mathcal{H}}(h(A) = h(B))$. You may use the following fact: $|\{h \in \mathcal{H} : h(A) \neq h(B)\}| = d(A,B)$.

(3 points) (b) Prove that, for the distance d and for any $\delta_1, \delta_2 \in [0,16]$, $\mathcal{H}$ is $\left(\delta_1, \delta_2, 1 - \frac{\delta_1}{16}, 1 - \frac{\delta_2}{16}\right)$-sensitive.

(7 points) (c) The L_∞-distance between two trytes A and B is defined as follows.

$$d_\infty(A, B) := \max_{0 \le i < 8} |A[i] - B[i]| \,.$$

Prove that, for the distance d_∞ and for any $\delta_1, \delta_2 \in [0, 2]$, $\mathcal{H}$ is $(\delta_1, \delta_2, \mathbf{1} - \frac{\boldsymbol{\delta_1}}{\mathbf{2}}, 1 - \frac{\delta_2}{16})$-sensitive.

3. Boosting locality-senstive functions (13 points)

Let $r, b \in \mathbb{N}$ and $\mathcal{F}$ be a family of functions. Let $\mathcal{F}_r^{\texttt{AND}}$ be the family of functions obtained by applying r-way `AND` to $\mathcal{F}$ and $\mathcal{F}_b^{\texttt{OR}}$ be the family of functions obtained by applying b-way `OR` to $\mathcal{F}$.

Let S be a non-empty set. Let $\mathcal{F}$ and d_1 be a family of functions and a distance metric on S, respectively. For any $s_1, s_2 \in S$, the following hold:

- $0 \leq d_1(s_1, s_2) \leq 1$.
- For f chosen uniformly at random from $\mathcal{F}$, $Pr_{\mathcal{F}}\left(f(s_1) = f(s_2)\right) = 1 - d_1(s_1, s_2)$.

For each of the following statements, say whether it is true, false, or cannot be decided with the given information. Justify your answers.

(1 point) (a) $\mathcal{F}$ is $(0.1, 0.9, 0.0, 1.0)$-sensitive.

(2 points) (b) $\mathcal{F}$ is $(0.1, 0.9, 0.5, 0.5)$-sensitive.

(3 points) (c) $\mathcal{F}_{100}^{\texttt{OR}}$ is $(0.1, 0.9, 0.9, 0.1)$-sensitive.

Let $\mathcal{G}$ and d_2 be a new family of functions and a new distance on S. For any $s_1, s_2 \in S$, the following hold:

- $0 \leq d_2(s_1, s_2) \leq 1$.
- For g chosen uniformly at random from $\mathcal{G}$, $Pr_{\mathcal{G}}(g(s_1) = g(s_2)) \geq 1 - d_2(s_1, s_2)$.

For each of the following statements, say whether it is true, false, or cannot be decided with the given information. Justify your answers.

(3 points) (d) $\mathcal{G}_2^{\texttt{AND}}$ is $(0.1, 0.9, 0.5, 1.0)$-sensitive.

(4 points) (e) For any $\delta_1, \delta_2 \in [0, 1]$ with $\delta_1 < \delta_2$, $\mathcal{G}$ is $(\delta_1, \delta_2, 1 - \delta_1, 1 - \delta_2)$-sensitive.

4. Online Convex Programming (19 points)

Assume that you are in the online convex programming setting over the domain $\mathcal{X} = [0, 2]$. Furthermore, assume that there are a total of four rounds, so that each algorithm has to propose four points x_1, x_2, x_3 and x_4. The four functions used to compute the respective losses are fixed and equal to

$$f_1(x) = \frac{1}{2}(x-1)^2,\ f_2(x) = \frac{1}{2}(x-2)^2,\ f_3(x) = \frac{1}{2}(x-3)^2,\ f_4(x) = \frac{1}{2}(x-2)^2,$$

so that you want to *minimize* their sum.

(6 points) (a) An algorithm suggested $x_1 = 1, x_2 = 2, x_3 = 1, x_4 = 2$. How much regret was incurred?

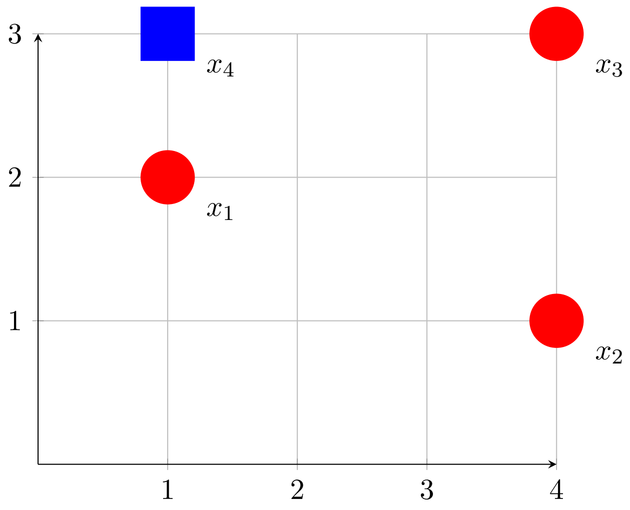
(6 points) (b) Assume we run online projected gradient descent on the above problem with a fixed step size of $\eta_i = 1$ starting from $x_1 = 0$. How much regret will be accrued?

(4 points) (c) Assume that we are given the sequence of functions beforehand. What is the minimal regret achievable by any algorithm in this setting?

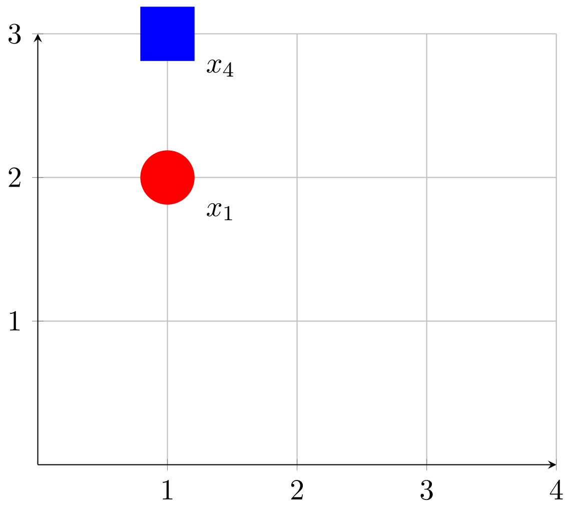
(3 points) (d) As in the previous question, assume that we are given the sequence of functions beforehand. What is the maximum possible regret any algorithm can have if change the domain to $\mathcal{X} = \mathbb{R}$?

5. Linear Decision Rules and Active Learning (16 points)

Consider a domain with four points x_1, x_2, x_3, x_4, each one labelled as either a square or circle (for the labels we will use $+1$ and -1 respectively). The exact positioning of these data points and their true labels are shown on the figure below.



Lisa wants to train a classifier to correctly classify the data. While she does not have access to the complete data set, she can draw samples from a distribution P that chooses from the four points *uniformly* at random. After each sample that she obtains, she trains a new linear classifier by solving the SVM optimization problem on the sampled data. She first drew two samples and obtained $\{(x_1, -1), (x_4, +1)\}$, so that the data that she sees looks as on the figure below.

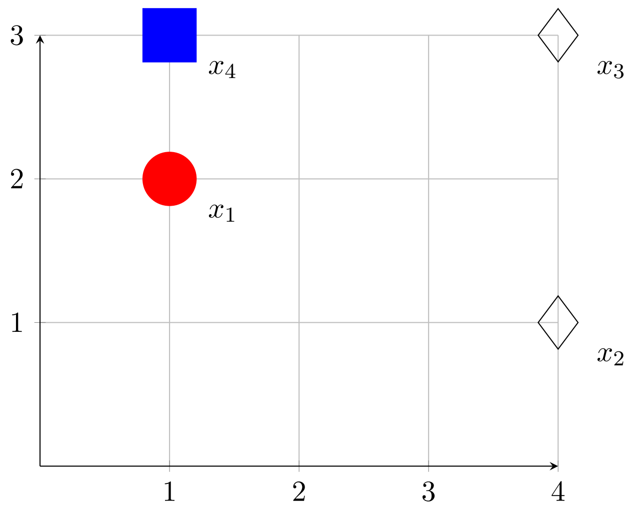


(2 points) (a) On the figure above draw the SVM solution that has been computed on the sampled data.

(9 points) (b) After having sampled the two points shown above, Lisa continues to draw samples from P. How many samples will she need *in expectation* until she computes a solution that correctly classifies *all four points*?

Hint: $\sum_{k=1}^{\infty} kp(1-p)^{k-1} = 1/p$, *for any* $0 < p < 1$.

(5 points) (c) Now assume that we are in the pool-based active learning setting. In addition to having the labels for x_1 and x_4, Lisa can actively query for the labels of x_2 and x_3. In other words, she has the following picture of the problem (we use diamonds for the points with unknown labels).



If she is using the uncertainty sampling strategy, please answer (i) how many queries will she make in total, and (ii) after how many queries will she compute a classifier that correctly classifies *all four points*.

6. k-armed bandit with unknown horizon (25 points)

In this task we consider the k-armed bandit problem. Each arm $i = 1, \dots, k$ is associated with a unknown probability distribution P_i with mean μ_i and support in $[0, 1]$. In each round t the player picks an arm i_t and obtains a random reward r_t sampled from P_{i_t}. The game is played for a total of T rounds, however we assume that T is *not* known to the player.

As usual the expected regret at time T is defined as $R_T = T\mu^* - \sum_{t=1}^{T} \mu_{i_t}$, where $\mu^* = \max_{i=1,\dots,k} \mu_i$, and the goal is to minimize regret.

(2 points) (a) Formally define what it means for a strategy to have 'no regret'.

(8 points) (b) Someone suggests the following, simple strategy, also known as 'explore-then-commit'. First each arm is sampled m times (exploration) and then the arm which at that point has the best empirical mean is played for the remaining rounds (exploitation). Show that for general distributions this strategy cannot have the no-regret property in expectation if m is a constant independent of T.

(5 points) (c) We define a confidence set $\mathcal{C}_i^t = \left\{\mu : |\mu - \hat{\mu}_i^t| \leq \sqrt{\frac{2\ln(t)}{n_i^t}}\right\}$ for each arm i. Explain how the sets $\mathcal{C}_i^t$ are related to the UCB1 strategy and informally argue why UCB1 can be seen as following the "optimism in the case of uncertainty" principle.

(10 points) (d) It can be shown that the $\mathcal{C}_i^t$ contain the true mean μ_i with probability at least $1 - 2/t^4$. Show that under the assumption that all confidence sets contain the true mean (ie for all t and all i, $\mu_i \in \mathcal{C}_i^t$), a suboptimal arm i is played at most $\frac{8\ln(T)}{(\mu^* - \mu_i)^2} + 1$ times.