

Data Mining – Learning from Large Data Sets
Final Exam

Date: 15 August 2013
Time limit: 120 minutes
Number of pages: 11
Maximum score: 100 points

You can use the back of the pages if you run out of space. Most subproblems can be solved independently from each other. Collaboration on the exam is strictly forbidden.

Please fill in your name:

1 [15 points] Frequent Itemsets Generation With Map-Reduce

In this question you are asked to provide a Map-Reduce program for finding frequently bought itemsets in a supermarket. You are given a set of **transactions** $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$, $t_i \subseteq \mathcal{S}$, where \mathcal{S} is the set of items available at the supermarket. Each transaction t_i represents a set of items that were bought together.

ID	Items
1	a, b, c
2	a, c,
3	a, c, e
4	c, e, d

Each subset of a transaction that contains more than one item represents an itemset. For example, transaction with ID 1 contains the following itemsets: $\{a, b\}$, $\{a, c\}$, $\{b, c\}$ and $\{a, b, c\}$. The itemset $\{a, c\}$ occurs in 3 transactions, while itemset $\{c, e\}$ occurs in two transactions. If the itemset occurs **at least K times** in the set of transactions we call it **frequent**.

You are given the functions:

- **hash(s)**: returns the **unique** hash value for the itemset **s**,
- **cost(s)**: returns the price of the itemset **s**, e.g. **cost**($\{a, c\}$) could be 5,
- **subsets(s)**: returns a list of all subsets of the set **s** with cardinality greater than one, e.g. **subsets**($\{a, b, c\}$) returns $\{a, b\}$, $\{a, c\}$, $\{b, c\}$ and $\{a, b, c\}$.

2 [25 points] Active learning from pairwise comparisons

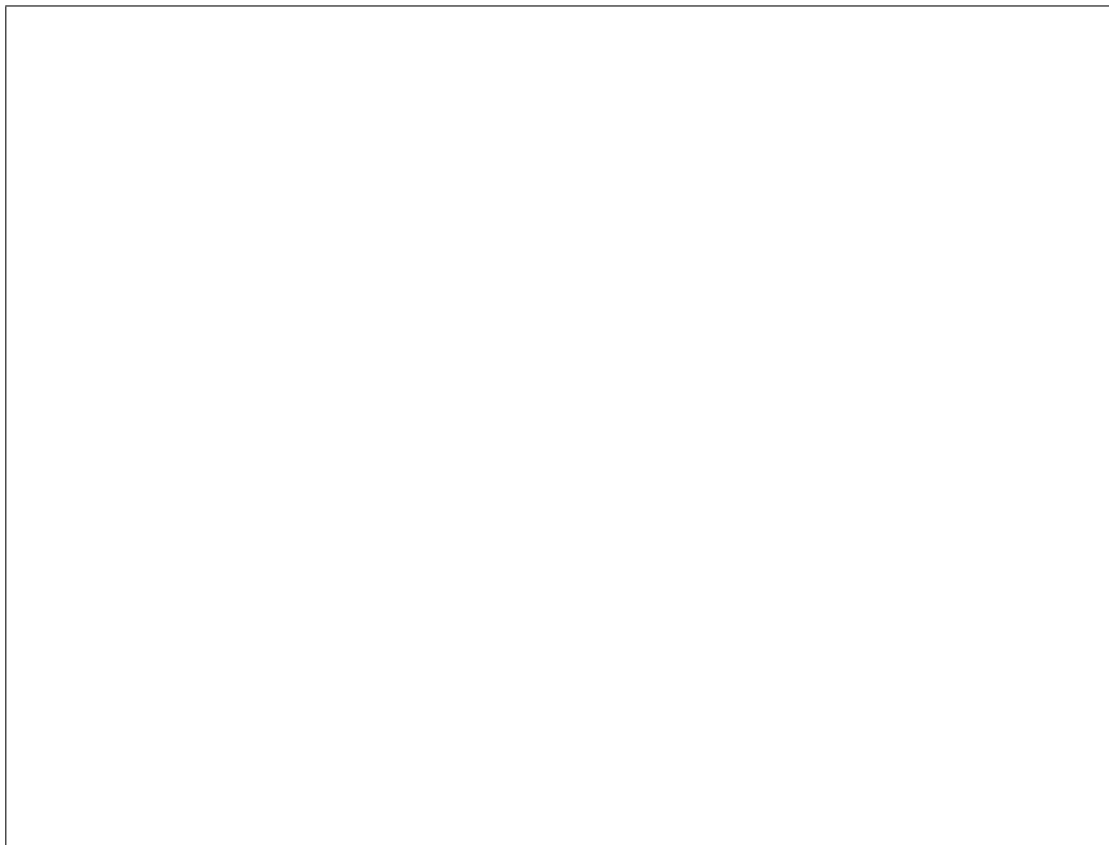
In this problem, you seek to learn a ranking of objects through pairwise comparisons. Let $\{1, \dots, n\}$ be the collection of objects we wish to rank, and $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$ be a set of normalized feature vectors associated with these objects. To determine the ranking, you want to devise an active learning strategy, where you can ask a human expert about a pair of items. The expert will respond which item of them has higher rank. You can assume that the expert always gives you an answer that is consistent with the true ranking.

Assume there is an underlying *linear* ranking function:

$$f^*(\mathbf{x}) = \langle \mathbf{w}^*, \mathbf{x} \rangle,$$

such that for any pair of objects $(\mathbf{x}_i, \mathbf{x}_j)$, it satisfies $f^*(\mathbf{x}_i) > f^*(\mathbf{x}_j)$ whenever the rank of \mathbf{x}_i is higher than the rank of \mathbf{x}_j (denoted $\mathbf{x}_i \succ \mathbf{x}_j$). Assume that for all $i \neq j$ it holds that $f(\mathbf{x}_i) \neq f(\mathbf{x}_j)$. We call \mathbf{w}^* an optimal *preference vector*. As a concrete example, in case $d = 2$ and $\mathbf{w}^* = [2, 0]^T$, the vector $\mathbf{x}_1 = [0, 1]^T$ would have lower rank than $\mathbf{x}_2 = [1, 0]^T$, because $f^*(\mathbf{x}_1) - f^*(\mathbf{x}_2) = \langle \mathbf{w}^*, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = [2, 0] \cdot [-1, 1]^T = -2 < 0$.

- (a) [10 points] Show that the problem of determining the optimal preference vector \mathbf{w}^* can be reduced to a problem of learning a classifier $h : \mathbb{R}^d \rightarrow \{+1, -1\}$, which, for any given pair of objects i, j , will predict whether \mathbf{x}_i has higher rank than \mathbf{x}_j . In particular, show how, given a training set of pairwise comparisons, a feasible preference vector can be obtained by solving a linear classification problem.



- (b) [8 points] Using the result from (a), develop a strategy for active learning through pairwise comparisons, which uses *uncertainty sampling* for SVMs. Write down your pseudo code below (the first two lines are provided).

```
begin
  Set  $U \leftarrow$  set of all pair of objects;      // unlabeled pairs
  Set  $L \leftarrow \emptyset$ ;                      // labeled pairs

end
```

- (c) [7 points] Recall that in class we have used locality sensitive hashing to quickly find the most uncertain examples for active learning. Similarly, in the case of active learning through pairwise comparisons, how can you use LSH to speed up your algorithm proposed in (b)? Write down the family of hash functions you use, and briefly explain your answer.

3 [20 points] Exemplar based k -means and submodularity

In this question you will consider a variant of the k -means clustering problem where the cluster centers are restricted to be picked from among the points in the dataset - we call this *exemplar based k -means*. The algorithm is given a set of n data points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, denoted by \mathcal{V} . The goal is to choose a set of k cluster centers, denoted by \mathcal{A} , *s.t.* $\mathcal{A} \subseteq \mathcal{V}$. We define the loss function for selecting \mathcal{A} as the set of cluster centers as follows:

$$L(\mathcal{A}) = \sum_{i \in \mathcal{V}} \min_{j \in \mathcal{A}} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$$

Hence, the exemplar based k -means can be posed as a subset selection problem where the goal is pick set \mathcal{A} of size k so as to minimize $L(\mathcal{A})$.

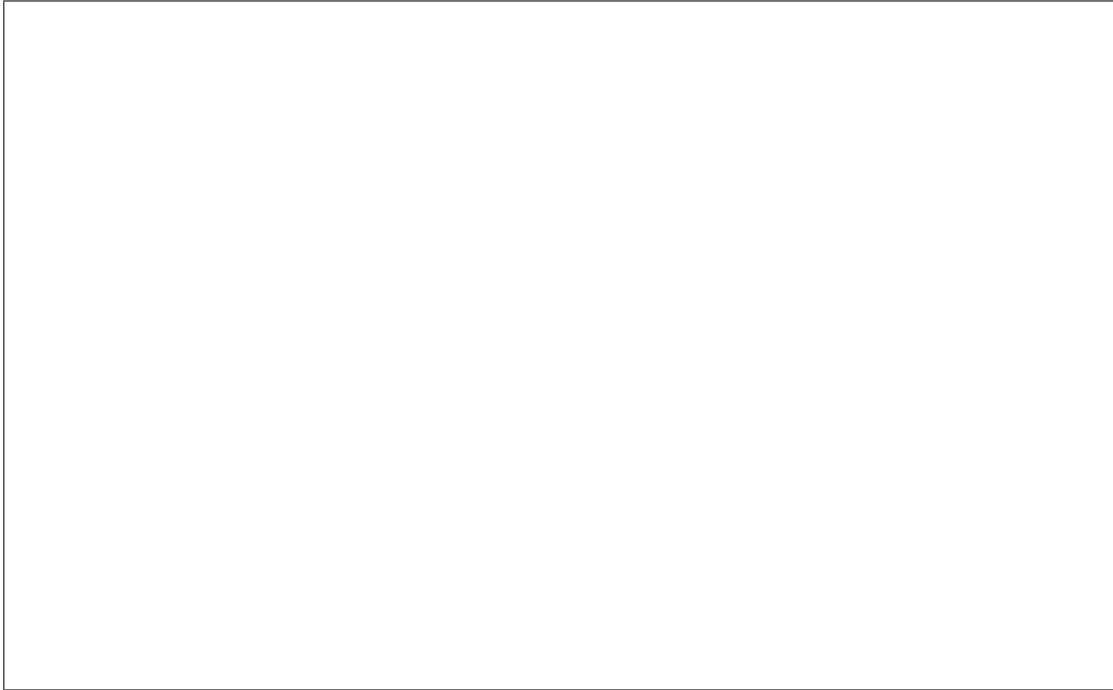
Let us suppose that one of the k cluster centers is fixed to be \mathbf{x}' . Now, the goal is to pick the remaining $k-1$ points that we denote by set \mathcal{S} (*i.e.* $\mathcal{A} = \mathcal{S} \cup \{\mathbf{x}'\}$). We pose this as the following maximization problem through the function F :

$$S^* = \underset{\mathcal{S} \subseteq \mathcal{V} \setminus \{\mathbf{x}'\}, \text{ s.t. } |\mathcal{S}|=k-1}{\arg \max} F(\mathcal{S})$$

$$\text{where } F(\mathcal{S}) = L(\{\mathbf{x}'\}) - L(\mathcal{S} \cup \{\mathbf{x}'\})$$

- (a) [14 points] Prove that function F is submodular. *Hint: Show that F can be written as a sum of simpler functions F_i , *i.e.*, $F(\mathcal{S}) = \sum_i F_i(\mathcal{S})$, each of which is submodular.*

- (b) [6 points] Using the submodularity property of F , write an algorithm to approximately solve the exemplar based k -means problem when one of the k centers is fixed to be \mathbf{x}' .



4 [25 points] Online asymmetric SVM for imbalanced data

For this question, you will derive an alternate version of the online SVM for the case of imbalanced data where the proportion of positive examples in the data is much higher than that of negative examples. The goal is to design an SVM with an asymmetric hinge loss function so that more weight is placed on the false-positives (i.e. wrongly classifying a negative example) compared to that of false-negatives (i.e. wrongly classifying a positive example). We call this alternate version of the SVM as ASYM-SVM. For a given $\alpha \geq 1$, we can formulate the ASYM-SVM as the following optimization problem:

$$\begin{aligned} \min_w \quad & w^T w \\ \text{s.t.} \quad & y_i \cdot w^T x \geq 1 \quad \forall y_i = +1 \\ & y_i \cdot w^T x \geq \alpha \quad \forall y_i = -1 \end{aligned}$$

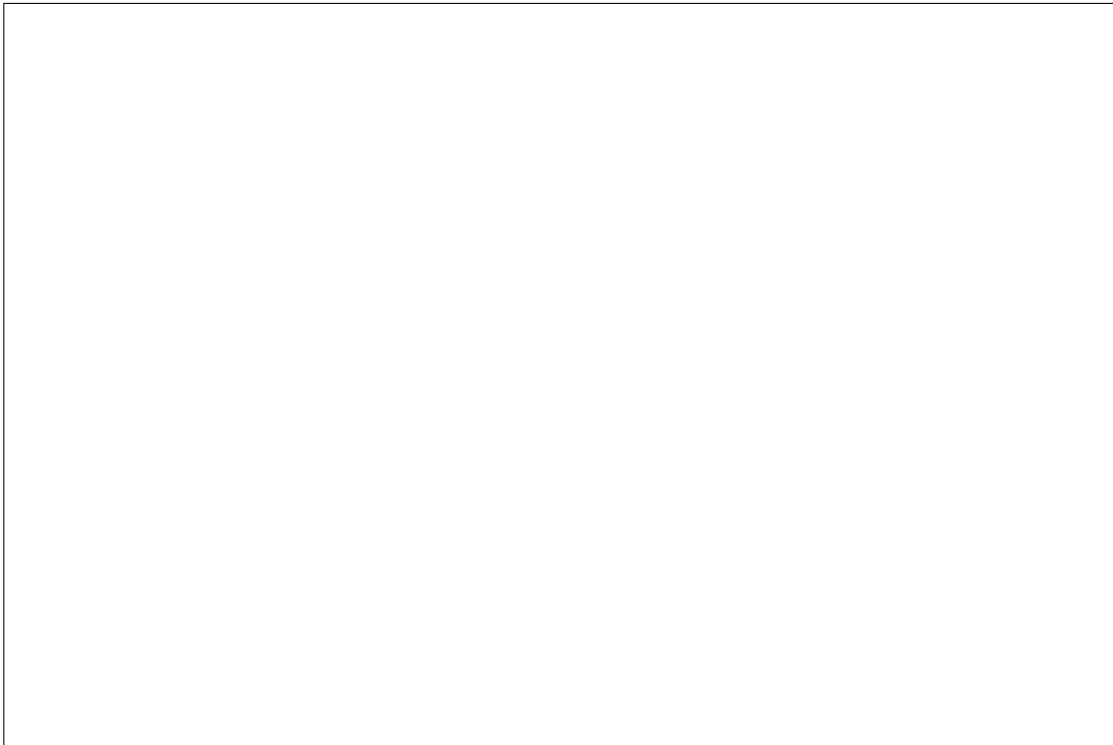
Note that for $\alpha = 1$, the problem reduces to the classic SVM as studied in the course. By introducing slack variables to handle noise, we can reformulate the above optimization problem into an online convex program using an (asymmetric) hinge loss function as follows:

$$\min_{\mathbf{w}} \sum_{i=1}^n \ell_{\text{asym}}(y_i, \mathbf{x}_i; \mathbf{w}) \text{ such that } \|\mathbf{w}\|_2 \leq \frac{1}{\lambda}$$

(a) [8 points] Derive the asymmetric hinge loss function ℓ_{asym} for the ASYM-SVM.



(b) [5 points] Plot the hinge loss function ℓ_{asym} separately for a positive example and for a negative example as a function of the confidence $\eta = y\mathbf{w}^T \mathbf{x}$.



- (c) [5 points] Derive the subgradient of the loss function ℓ_{asymp} with respect to the weight vector \mathbf{w} .

- (d) [7 points] Augment the pseudo code below so that it performs online convex programming for solving the ASYM-SVM optimization problem.

```
begin
   $\mathbf{w}_0 \leftarrow 0$ ;
  for  $t = 1$  to  $T$  do
    // Observe example  $(\mathbf{x}_t, y_t)$  at time  $t$ 
    // Fill in code to implement ASYM-SVM
  end
end
```


5 [15 points] Offline Bandit Evaluation

Suppose you have a set of n products, $N = \{1, 2, \dots, n\}$, in an inventory of an online store. For each user visit to the website, you can recommend exactly one of these products. The user then either buys the product or leaves the website. The user at time t is denoted by a feature vector $z_t \in \mathbb{R}^d$. The z_t 's arrive in an i.i.d sequence. For the sake of simplicity, assume that you have no features for the products. Also assume, the list of available products is constant. The problem of determining the optimal recommendation then translates to the *contextual k -armed bandit* problem.

For each user visit, t , the website stores a triple (z_t, i_t, r_t) where z_t is context, $i_t \in N$ is the product that was displayed and $r_t \in \{0, 1\}$ indicates whether the user bought the product or not. The collection of these triples $D_T = \{(z_1, i_1, r_1), \dots, (z_T, i_T, r_T)\}$ is referred to as the access log up to time T .

Under this setting, answer the following two questions:

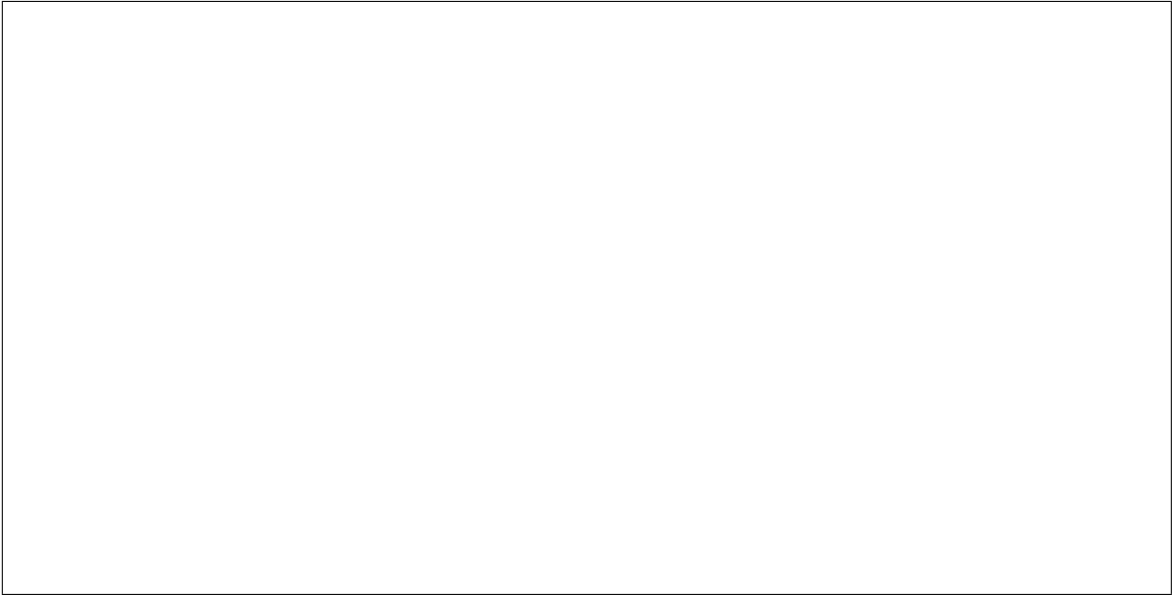
- (a) [9 points] Assume a scheme A (described here) was used to recommend the products. Scheme A uses a fixed distribution P over the items and picks an item i_t for each user visit according to the distribution P independently at random. Also, assume that every item has probability at least p_{min} of being picked where $p_{min} > 0$.

Realizing that A is not an ideal algorithm, an alternative algorithm B has been suggested as a replacement. B is adaptive in the sense that the item recommended by B at t depends on the rewards of the items picked up to time $t - 1$. You are tasked with evaluating B offline before actually deploying it on the website. To do so, you are also given the access log file D_T from the website that was generated when using scheme A.

From the lectures, we know that if P were the *uniform* distribution, then we can use a rejection sampling technique to obtain an unbiased estimate of B's performance using D_T . Now, suppose P is *not necessarily uniform* and as described above. Develop a (sub)sampling scheme, which produces a new log \hat{D}_T , such that the rejection sampling approach discussed in class can be used on \hat{D}_T to obtain an unbiased estimate of the performance of B.



- (b) **[3 points]** If T_1 is the total number of lines in the access log D_T , what is the expected number of lines in \hat{D}_T produced by the scheme you developed in (a)?



- (c) **[3 points]** Now, suppose the minimum sampling probability p_{\min} used by scheme A is 0 (i.e., some items are never recommended by scheme A). Can we still subsample D_T to obtain \hat{D}_T such that we can use rejection sampling to obtain an unbiased performance estimate of B? Please justify your answer.

