CYBER DATA ANALYTICS (CS4035)

Trial Exam, June 2016
3 hours to complete

Important: This exam consists out of 6 questions. Always give full explanations of your answers and number all steps of the asked algorithms. Do not forget to put your name and student number on every sheet of paper. Answers are required to be in English.

Question 1 - fraud detection

You are asked to build a machine learning framework for credit card fraud detection.

(a) Give three arguments why fraud detection is different from traditional machine learning. (10 pt)

---

solution

1. The class labels are unbalanced, very few fraud instances when compared to non-fraud.
2. The goal is to catch people that actively try to hide, i.e., fraudsters. Simple decision rules based on thresholds are ineffective as fraud detectors because fraudsters quickly discover how to stay below the radar.
3. There is a clear need for white-box solutions (interpretable models). Blocking a client due to fraud suspection requires a well-founded reason, simply telling them "because the computer says so" (for instance when using neural networks) is no working solution.

(b) In order to cope with the above differences, a friend of yours suggests to use the SMOTE technique. Briefly explain this technique using pseudo code, what is it for, and give one advantage and one disadvantage of using SMOTE over resampling the data. (10 pt)

---

solution SMOTE is the Synthetic Minority Oversampling TEchnique. It generates new "realistic" instances for the minorty class by randomly combining their features. It can only be applied to numeric data sets. From slides:

1. For every minority instance i
2. Randomly select one of the the k nearest neighbors x
3. Compute the vector $v(i,x)$ between i and x: $i + v(i,x) = x$
4. Randomly select a point p along this vector: $p = i + \text{RAND}(0,1) * v(i,x)$
5. Add p to the minority instances

One advantage is that it reduces a key problem of oversampling, namely that the models will tend to overfit on the resampled points.

One disadvantage is that it creates artificial instances, which can be incorrect/very noisy and even overlap with positive instances. (Removing Tomek links aims to reduce this problem.)

Question 2 - anomaly detection

Suppose you want to detect anomalies on a time-series dataset $Y \in \mathbb{R}^{T\times n}$, where rank(Y) = k,

(a) Let $n = 1$. Discuss whether a PCA or ARMA model would be better suited for anomaly detection on such a dataset. (10 pt)

---

solution A PCA-based approach can only be used with multivariate time-series, whereas ARMA model can be learned from both univariate and multivariate time series. Since $n = 1$, the time-series is univariate and thus PCA cannot be directly applied on the time-series. Hence, an ARMA-based approach is better suited.

(b) Let $n = 30$ and rank(Y) = k = 20. Consider a PCA-based anomaly detection algorithm, where the Singular Value Decomposition of the data is $Y = USV^\top$, where $U \in \mathbb{R}^{m\times k}$, $S \in \mathbb{R}^{k\times k}$ is a diagonal with the ordered singular values as its entries, and $V \in \mathbb{R}^{n\times k}$.

Describe how a PCA-based anomaly detection method could be implemented, given the data set Y and the matrices P, S, and V. (10 pt)

---

solution A PCA-based anomaly detection method has three main steps: 1) to learn a spatial model of the normal behavior across the different dimensions, based on the multivariate time-series data; 2) to generate a residual, $\tilde{y}_k$, for the data points of the time-series, $y_k \in \mathbb{R}^n$, at each time-step k (note that $y_k^\top$ corresponds to the k-th row of the matrix Y); 3) to evaluate the residual $\tilde{y}_k$ at each time-step. Next we describe the methodology behind each of these steps.

The first step is to learn a model of the normal behavior. Using a PCA-based approach, the normal behavior is captured in a few number of principal components (the ones with largest singular values), while the anomalies are captured in the remaining principal components (with smaller singular values). Note that there are at most $k = 20$ principal components, even though the multivariate time-series data has $n = 30$ dimensions. Hence, the number of principal components modeling the normal behavior must be a number, r, between 1 and 20. A large value of r will lead to an over-fitting phenomena with a low detection (anomalies are also captured in the model), whereas a very small value might lead to a high false-alarm rate (some normal behavior is not captured by the chosen principal components). For a given value of r, the learned model can be derived from the SVD decomposition: we take the first r principal components, namely the first r columns of U and V, and the first r diagonal entries of S. Since we are interested in a spatial model across the different dimensions, we take the first r columns of $V \in \mathbb{R}^{n\times k}$ and construct the projection matrix $P = [v_1\ v_2\ \ldots\ v_r] \in \mathbb{R}^{n\times r}$ as our model.

The second step is to generate the residual at each time-step, which can be obtained as $\tilde{y}_k = y_k - \hat{y}_k$, where $\hat{y}_k$ is the modeled component of the data obtained as $\hat{y}_k = PP^\top y_k$. The residual is then generated as $\tilde{y}_k = (I - PP^\top)y_k$.

Finally, the third step is to evaluate the residual at each time-step using, for instance, a threshold-based approach. In particular, one may compare the norm of the residual, $\|\tilde{y}_k\|^2$, against a given threshold δ_α^2. In such a case, an anomaly is detected at time k if the residual's norm exceeds the threshold, i.e., $\|\tilde{y}_k\|^2 > \delta_\alpha^2$.

Note: other versions of a PCA-based anomaly detection are also be possible.

Question 3 - distributed processing

Consider a set of $n = 100$ data points (x_i, y_i), $i = 1, \ldots, n$, where x_i and y_i denote the input and output attributes, respectively. A multi-perceptron neural network is to be learned from the n data points, by training the weights w_j, $j = 1, \ldots, m$, according to the batch algorithm:

$$w_j^{(k+1)} = w_j^{(k)} - \eta_k\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\partial L(w_j, x_i, y_i)}{\partial w_j}\right), \forall j = 1, \ldots, m.$$

Sketch a MapReduce implementation for the given algorithm, assuming that you have 5 mappers available. Clearly indicate what data is available to each node, the data exchanges between nodes, and the main steps of the implementation. (15pt)

solution They key point in computing the weight update in a distributed fashion is to distribute the computation of the average gradient, $\frac{1}{n}\sum_{i=1}^{n}\frac{\partial L(w_j, x_i, y_i)}{\partial w_j}$. Since we have 5 mappers, we will split the data set D into 5 mutually exclusive data sets, $\{D_m\}$, whose union corresponds to D. A sketch of a MapReduce implementation is as follows.

```
While weights have not converged
  Split data points among mappers: each mapper $M_m$ has access to the dataset $D_m \subset D$;
  Send current weights $\{w_j\}$ to all mappers;
  For each mapper $M_m$, run the map function $\mathrm{Map}((\mathrm{x_i}, \mathrm{y_i})_{\mathrm{D_m}},\ \{\mathrm{w_j}\})$, which produces a set of key-value pairs $< j;\ \gamma_k,\ |D_m| >$. In the latter, the weight index $j$ is the key and the tupple $(\gamma_m,\ |D_m|)$ is the value, with $\gamma_m = \sum_{i \in D_m}\frac{\partial L(w_j, x_i, y_i)}{\partial w_j}$ and $|D_m|$ being the number of data points in $D_m$;
  Sort and Shuffle the key-value pairs computed by the mappers and assign all pairs with the same key to the corresponding reducer;
  For each key $j$, run the reduce function $\mathrm{Reduce}(\gamma_\mathrm{m},\ |\mathrm{D_m}|)$ that returns the result $\delta_j = \frac{1}{\sum_{m=1}^{5}|D_m|}\sum_{m=1}^{5}\gamma_m$;
  Given the results from the reducers, the Master updates the weights as $w_j^{(k+1)} = w_j^{(k)} - \eta_k \delta_j$.
end while
```

Question 4 - stream mining

Consider a stream of length n. Each item in the stream is an integer. We know that there exists a majority element in the stream, i.e., one of the integers repeats itself at least $\frac{n}{2}$ times (we don't know anything about the other items, they may repeat or may be unique).

Give an algorithm to find the majority element. You can use only a constant amount of extra space. Also provide a brief correctness argument (need not be a mathematical proof, 3-5 lines will suffice) for your algorithm. (15 pt)

solution Algorithm:

```
1. num = 0, count = 0
2. while stream.hasNext() do
3.     curInt = stream.next()
4.     if count == 0 then
5.         num = curInt, count = 1
6.     else if stream.next() == num then
7.         count++
8.     else
9.         count--
10.   endif
11. endwhile
10. output num
```

The main idea of the proof is that if you remove two distinct elements from the stream, the answer remains unchanged, i.e.: Suppose we read a majority element form the stream (which we know exists), then depending on count, 3 things can happen (lines 4, 6, and 8). In the first 2 cases, we increment count. This extra count can later be decremented by a non-majority element (through line 8). In the last case, it undoes a count increase for a non-majority element (through line 8). Removing all such pairs of majority and non-majority elements thus does not influence the count or the result of the algorithm. Since there is a majority element, after removal there will be only majority elements left, which will thus be output at line 10. Clearly, the required space of constant: one pointer and one counter.

Question 5 - sequential data mining

You are given the following discrete sequence data, collected by observing a network of different hosts:

host 1	A,B,A,B,A,B,A,B,A,B
host 2	A,B,A,B,B,C,B,B,C,A
host 3	B,B,B,C,B,B,A,C,B,B
host 4	A,C,B,B,A,C,B,B,B,C

(a) Provide the 2-gram probabilities for each of these hosts. (5 pt)

solution

	host 1	host 2	host 3	host 4
$P(A\|A)$	0	0	0	0
$P(B\|A)$	1	1	0	0
$P(C\|A)$	0	0	1	1
$P(A\|B)$	1	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{5}$
$P(B\|B)$	0	$\frac{2}{5}$	$\frac{4}{6}$	$\frac{3}{5}$
$P(C\|B)$	0	$\frac{2}{5}$	$\frac{1}{6}$	$\frac{1}{5}$
$P(A\|C)$	?	$\frac{1}{2}$	0	0
$P(B\|C)$	?	$\frac{1}{2}$	1	1
$P(C\|C)$	?	0	0	0

(b) Explain the difference between a fingerprint and a behavioral/probabilistic profile using the above models. Describe how these can be detected in new data sequences. (5pt)

solution A fingerprint is a short sequence that uniquely identifies a host, i.e., a subsequence that occurs only in this host and no other. An example is CA for host 2. $P(A|C)$ has zero probability in all other hosts. A behavioral profile is a distribution over sequences, for instance a 2-gram. Fingerprints are straightforward to detect, simply match the new data sequences with the known fingerprint. A profile can be detected by counting the observed frequencies of the individual subsequences (i.e. 2-grams) and comparing those with the expected frequences given the known profile (2-gram model). When the difference between the expected and observed frequencies is sufficiently small, we detect the profile. Another option is to perform this procedure for each known profile and choosing the one that fits the data best using maximum likelihood.

(c) Can you find a sequence that matches as a fingerprint with one host, but as a behavioral/probabilistic profile with another? Answer yes or no and why. (5pt)

solution Both answers are good with proper reasoning:

- **No.** The only fingerprint is CA for host 2. Any sequence that contains CA has zero probability in all other hosts, therefore the likelihood that any of those hosts generated data containing CA is 0 and therefore no profile but the one from host 2 can possibly match such a sequence.
- **Yes.** The only fingerprint is CA for host 2. A possible sequence containing CA is ACBBACBBBCA. The observed counts from this sequence closely match the expected ones for a type 4 host. The profile from host 4 will therefore match using profiles while the fingerprint from host 2 matches.

Question 6 - anonymity

Suppose you are asked to publish the following table, in which Gender and Birth year are assumed to be quasi-identifiers, and BSN is an identifier:

BSN	Name	Gender	Birth year	Disease	Treatment
78979879	Alice	Female	1980	A	1
89779003	Bob	Male	1980	B	2
12123987	Cheryl	Female	1979	C	3
57893980	Diana	Female	1981	A	4
56787922	Erik	Male	1981	B	4
90890892	Frederic	Male	1979	B	2
67899909	Gennaro	Male	1980	A	1

(a) Descibe two methods that can be applied to the BSN, Gender and Birth year columns in order to avoid record linkage. (5pt)

solution

- **Suppression** The BSN column needs to be removed (suppressed completely) since it uniquely identifies the people in the data set.
- **Generalization** The Birth year column needs to be generalized (using year ranges or by removing year digits) in order to make it harder for an attacker who knows the exact birth date to identify the person (s)he seeks.

(b) Give the definition of an anonimity set and compute the anonimity sets for every row in the table (with respect to the quasi-identifiers). (5pt)

solution Definition: an anonimity set is a set of people with identical values for the quantities of interest (quasi-identifiers Gender and Birth-year).

BSN	Anonimity set
78979879	{ 78979879 }
89779003	{ 89779003, 67899909 }
12123987	{ 12123987 }
57893980	{ 57893980 }
56787922	{ 56787922 }
90890892	{ 90890892 }
67899909	{ 89779003, 67899909 }

(c) A classifier for determining the treatment assigns Treatment 1 to Disease A, Treatment 2 to Disease B, and Treatment 3 to disease C, unless the Birth year is strictly greater than 1980, in that case it assigns treatment 4. Is it possible to make the data 2-anonymous without influencing the accuracy of this classifier? If yes, how? If no, why? (5pt)

solution Yes, this is possible. Simple generalize and suppress the data as follows:

BSN	Name	Gender	Birth year	Disease	Treatment
*	Alice	*	$<= 1980$	A	1
*	Bob	*	$<= 1980$	B	2
*	Cheryl	*	$<= 1980$	C	3
*	Diana	*	> 1981	A	4
*	Erik	*	> 1981	B	4
*	Frederic	*	$<= 1980$	B	2
*	Gennaro	*	$<= 1980$	A	1

Now the anonimity sets are all of size at least 2 and all the required information on Birth year is still available.