CS 337 (Spring 2019): Class Test 2

Instructor: Shivaram Kalyanakrishnan

4.00 p.m. – 5.15 p.m., February 25, 2019, 101/103/105 New CSE Building

Total marks: 20

Note. Provide brief justifications and/or calculations along with each answer to illustrate how you arrived at the answer.

Question 1. Consider the *bootstrap* operation: given a data set $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$ (the form of (x^i, y^i) does not matter for this question), we create new data set D_1, of the same size, as follows.

> $D_1 \leftarrow \emptyset$.
> Repeat n times:
> Select $i \in \{1, 2, \ldots, n\}$ uniformly at random and add $(x^i, y^i) \in D$ to D_1.

Assume D is a *set* with distinct elements. In general D_1 can be a *multiset* with repeated entries.

1a. What is the probability that a given data point $(x^i, y^i) \in D$ will occur exactly k times in D_1, where $0 \leq k \leq n$? [1 mark]

1b. What is the expected number of times that a given data point $(x^i, y^i) \in D$ will occur in D_1? [1 mark]

1c. Suppose a data set D_2 is created from D using the same process used to generate D_1, but independently. What is the expected number of elements that D_1 and D_2 will have in common? We use the convention that the number of elements common to D_1 and D_2 is $|\text{Set}(D_1) \cap \text{Set}(D_2)|$, where Set() only retains the unique elements. For example, multisets $\{A, A, B, C, D\}$ and $\{A, A, A, B, B, E\}$ have *two* elements, A and B, in common. [2 marks]

Question 2. Consider the problem of linear regression with regularisation. We know that regularisation is meant to yield smaller-magnitude weights—but is it guaranteed to do so? Concretely, consider L_2 regularisation. As usual, let $\{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^n, y^n)\}$ be the input data set, with $\mathbf{x}^r \in \mathbb{R}^d$ and $y^r \in \mathbb{R}$ for $r \in \{1, 2, \ldots, n\}$. For $0 < \lambda_1 < \lambda_2$, let

$$\mathbf{w}^1 = \underset{\mathbf{w} \in \mathbb{R}^d}{\text{argmin}} \left(\sum_{r=1}^{n} (y^r - \mathbf{w} \cdot \mathbf{x}^r)^2 + \lambda_1 \|\mathbf{w}\|^2 \right), \text{ and } \mathbf{w}^2 = \underset{\mathbf{w} \in \mathbb{R}^d}{\text{argmin}} \left(\sum_{r=1}^{n} (y^r - \mathbf{w} \cdot \mathbf{x}^r)^2 + \lambda_2 \|\mathbf{w}\|^2 \right),$$

where $\|\mathbf{w}\| = \sqrt{(\mathbf{w}_1)^2 + (\mathbf{w}_2)^2 + \cdots + (\mathbf{w}_d)^2}$. Is it guaranteed that $\|\mathbf{w}^2\| \leq \|\mathbf{w}^1\|$? Prove that your answer is correct. [4 marks]

Question 3. Consider the special case of the k-means clustering problem with $k = 2$ and $d = 1$: in other words, we wish to cluster $n \geq 3$ points $x^1, x^2, \ldots, x^n \in \mathbb{R}$ into two clusters and assign the clusters centres so as to minimise the sum squared error. Is the k-means clustering algorithm guaranteed to find an optimal clustering for this special case? Assume that the initial clustering has one or more points assigned to each cluster, but is otherwise arbitrary. Prove that your answer is correct. [4 marks]

Question 4. You are given access to a function median() that takes as input a real-valued array Z and returns the median of the elements in Z. Note that Z need not be sorted, and also note that the median is a value, not an index. Indeed there are versions of median() that run in time linear in the size of Z; assume you are given such a version.

Provide pseudocode for a function NearestNeighbours(k, n, d, x, D), which should return the labels of k nearest points in the n-sized data set D to a query point $x \in \mathbb{R}^d$. Assume $1 \leq k \leq n$. D is of the form $(x^r, y^r)_{r=1}^n$, where $x^r \in \mathbb{R}^d$ and $y^r \in \{0, 1\}$.

- Use $D.x[i]$ to access the i^{th} data element of D, which is a vector of size d. The corresponding label is $D.y[i]$. You can also use $D.x$ and $D.y$ to access the entire list of data points and labels, respectively.
- Use Euclidean distance for identifying nearest neighbours. Assume no two points in D are at the same distance from x, and hence there are no ties to resolve.
- Your code should return a k-sized array of labels, each 0 or 1.
- Use standard operators for arithmetic, comparison, logic, assignment, etc. The only functions you can use are median() and those functions you fully define yourself.

Your aim is to make NearestNeighbours(k, n, d, x, D) as efficient as you can by using median() appropriately. What is the complexity of your implementation as a function of n and k (ignore the dependence on d)? [6 marks]

Question 5. In the real world, labelled data sets are hard to come by, but there is a large amount of unlabeled data. For example, a hospital might have a large collection of chest X-rays from previous years (whose labels have not been entered into their database), but only a small number of recent ones in which the X-rays are tagged with the corresponding diagnosis.

Abstractly, assume that we have labelled data set $D_L = \{(x^1, y^1), (x^2, y^2), \ldots (x^n, y^n)\}$ and also an unlabelled data set $D_U = \{z^1, z^2, \ldots, z^m\}$, where x and z come from the same domain and distribution. We have seen several ways to build classification models by training on D_L. Do you think D_U can also help in the process? In what ways? [2 marks]

Solutions

1a. $\binom{n}{k}p^k(1-p)^{n-k}$ where $p = \frac{1}{n}$.

1b. Since the bootstrap operation is symmetric with respect to all the elements of D, and since D_1 has the same number of elements as D, the expected number of times a given point $(x^i, y^i) \in D$ will occur in D_1 is 1. Alternatively, we can also use the answer to the previous question to calculate the expectation as follows:

$$\begin{aligned}
\sum_{k=0}^{n} k\binom{n}{k}p^k(1-p)^{n-k} &= \sum_{k=1}^{n} k\binom{n}{k}p^k(1-p)^{n-k} \\
&= \sum_{k=1}^{n} n\binom{n-1}{k-1}p^k(1-p)^{n-k} \\
&= \sum_{l=0}^{n-1} np\binom{n-1}{l}p^l(1-p)^{(n-1)-l} \\
&= np \\
&= 1.
\end{aligned}$$

1c. The expected number of times D_1 and D_2 will have some fixed element $(x^i, y^i) \in D$ in common is the probability that it occurs in both D_1 and D_2, which is $(1-(1-p)^n)^2$. The expected number of elements D_1 and D_2 will have in common is the sum of the expectations for the individual data points, and therefore $n(1-(1-p)^n)^2$.

2. For $\mathbf{w} \in \mathbb{R}^d$, let $E(\mathbf{w}) = \sum_{r=1}^{n}(y^r - \mathbf{w}\cdot\mathbf{x}^r)$. Since $\mathbf{w}^1 = \text{argmin}_{w\in\mathbf{R}^d}(E(\mathbf{w}) + \lambda_1\|\mathbf{w}\|^2)$, we have

$$E(\mathbf{w}^1) + \lambda_1\|\mathbf{w}^1\|^2 \leq E(\mathbf{w}^2) + \lambda_1\|\mathbf{w}^2\|^2.$$

Similarly, from the definition of $\mathbf{w}^2$, we get

$$E(\mathbf{w}^2) + \lambda_2\|\mathbf{w}^2\|^2 \leq E(\mathbf{w}^1) + \lambda_2\|\mathbf{w}^1\|^2.$$

These inequalities combine to give

$$\lambda_2(\|\mathbf{w}^2\|^2 - \|\mathbf{w}^1\|^2) \leq E(\mathbf{w}^1) - E(\mathbf{w}^2) \leq \lambda_1(\|\mathbf{w}^2\|^2 - \|\mathbf{w}^1\|^2),$$

which means

$$(\lambda_2 - \lambda_1)(\|\mathbf{w}^2\|^2 - \|\mathbf{w}^1\|^2) \leq 0.$$

Since $\lambda_2 > \lambda_1$, we infer $\|\mathbf{w}^2\|^2 \leq \|\mathbf{w}^1\|^2 \implies \|\mathbf{w}^2\| \leq \|\mathbf{w}^1\|$.

3. We show by example on a data set with $n = 3$ points that the k-means clustering algorithm need not find an optimal clustering. Take $x_1 = 1; x_2 = 5, x_3 = 8$. Let

- Clustering 1 assign x_1 and x_2 to cluster 1, with centre $\mu^1 = 3$; and x_3 to cluster 2, with centre $\mu^2 = 8$, and
- Clustering 2 assign x_1 cluster 1, with centre $\mu^1 = 1$; and x_2 and x_3 to cluster 2, with centre $\mu^2 = 6.5$.

For both clusterings, the centres are optimal, and given the centres, the cluster assignment is optimal. In other words, both clusterings are locally optimal. Hence, if initialised with either of these clusterings, the k-means clustering algorithm will have already converged. However, we note that the SSE of Clustering 1 is 8 and the SSE of Clustering 2 is 4.5. Clustering 2 is the sole optimal 2-clustering of this data set—which means k-means sometimes converges to a suboptimal clustering.

4. It is possible to find the k nearest neighbours of x in $O(n)$ time for all k, without any preprocessing. First we calculate the distance of each point to x; this step needs $O(n)$ time. We can find the median distance again in $O(n)$ time, and through comparison, identify the $\lceil n/2 \rceil$ closest points and the $\lfloor n/2 \rfloor$ farthest points, putting these in separate arrays. The total time taken thus far is $O(n)$. Now, if $k \leq \lceil n/2 \rceil$, it follows that we only need search the "closer points" array for the k nearest neighbours; otherwise we look for the $(k - \lceil n/2 \rceil)$ nearest points in the "farther points" array and return them along with the entire "closer points" array. In either case, our new search is restricted to $\lceil n/2 \rceil$ points, and so the total time taken remains linear in n.

The pseudocode on the following page implements this logic.

Distance(x, x')
$s \leftarrow 0$.
For $i \in \{1, 2, \ldots, |x|\}$:
    $t \leftarrow x[i] - x'[i]$.
    $s \leftarrow s + t \times t$.
$s \leftarrow \sqrt{s}$.
Return s.

Append(x, x')
For $i \in \{1, 2, \ldots, |x|\}$:
    $x''[i] \leftarrow x[i]$.
For $i \in \{1, 2, \ldots, |x'|\}$:
    $x''[i + |x|] \leftarrow x'[i]$.
Return x''.

NearestNeighbours(k, n, d, x, D)
If $k = n$
    Return $D.y$.
For $i \in \{1, 2, \ldots, n\}$:
    $Z[i] \leftarrow$ Distance$(x, D.x[i])$.
$z \leftarrow$ Median(Z).
$l \leftarrow 1; r \leftarrow 1$.
For $i \in \{1, 2, \ldots, n\}$:
    If $Z[i] \leq z$
        $D_{left}.x[l] \leftarrow D.x[i]$.
        $D_{left}.y[l] \leftarrow D.y[i]$.
        $l \leftarrow l + 1$.
    Else
        $D_{right}.x[r] \leftarrow D.x[i]$.
        $D_{right}.y[r] \leftarrow D.y[i]$.
        $r \leftarrow r + 1$.
If $k \leq \lceil \frac{n}{2} \rceil$
    Return NearestNeighbours$(k, \lceil \frac{n}{2} \rceil, d, x, D_{left})$.
Else
    Return Append$(D_{left}.y,$ NearestNeighbours$(k - \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor, d, x, D_{right}))$.

5. Not surprisingly, it is of much interest to use D_U for training a better model: the entire area of "semi-supervised learning" is devoted to this cause. The student is referred to the relevant Wikipedia page for a descriptive account of this topic:

`https://en.wikipedia.org/wiki/Semi-supervised_learning`.

In short, D_U (which is usually much larger than D_L) can provide statistical strength to any operations performed solely on the "x" part of D_L, such as dimensionality reduction. Another popular approach is transduction, in which labels are estimated for points in D_U based on their proximity to points in D_L (say by clustering), and thereby a larger labelled data set is used (using the same supervised learning pipeline as before) for training.