University of Glasgow | School of Computing Science

DayOfWeek DayOfMonth Month 2018

XX.XX am/pm – XX.XX am/pm

(Duration: 1.5 hours)

EXAMINATION FOR

DEGREES OF MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

Data Fundamentals (H)

Answer 2 of 3 Questions

This examination paper is worth a total of 50 marks

For examinations of at least 2 hours duration, no candidate shall be allowed to leave the examination room within the first hour or the last half-hour of the examination

INSTRUCTIONS TO INVIGILATORS

Please collect all exam question papers and return to School together with exam answer scripts

ANSWERS HIGHLIGHTED -- REMOVE FROM FINAL PAPER.

1. You are asked to develop numerical processing software for an embedded device. This device has a floating point unit (FPU) capable of SIMD operations on IEEE754 floating point values.

   (a) Define SIMD, and explain in what situation it can improve performance in numerical operations.

   [3]

   SIMD is single instruction multiple data [#2] and it can be used in operations when the same operation has to be performed to many data elements simultaneously [#1].

   (b) The FPU is configured for 64 bit floating point numbers. Draw a diagram showing the partition of a 64-bit IEEE 754 number ("float64") into three key components. Briefly describe the purpose of each of component and state the number of bits used for each component.

   [4]

   ```
   1  |  11  |  52
   S  |  E   |  M
   ```

   - S: determines the sign of the number [#1]
   - E: exponent determines the power of 2 the mantissa is multiplied by [#1]
   - M: mantissa represents the number in normalised form [#1], with an implied leading 1 [#1]

   [#1] marks for correct number of bits in each category

   (c) Multiplying the IEEE754 `float64` number below with the `float64` value -2.0 will have what effect? Do **not** rewrite the whole number; just describe the effect briefly in words.

   [2]

   ```
   1111111111110000000000000000000000000000000000000000000000000011
   ```

   The number will be unchanged, as it represents NaN (not a number)[#1]. Any operation involving NaN resuts in NaN.[#1]

   (d) A colleague suggests implementing a multidimensional array of float64 numbers as a nested set of *linked lists*. Each element of this data structure has three fields:

   – **value**: float64 or 64 bit pointer to sublist
   – **type**: a 64 bit integer stating whether this element is a floating point value or a sublist
   – **ptr**: a 64 bit pointer to the next element.

   Each dimension of the array is represented using sublists. A (3,4) 2D array would be represented in the form of lists like this:

   ```
   [[a,b,c,d],
    [d,e,f,g],
    [h,i,j,k]]
   ```

   As each 64 bit element needs 8 bytes of storage, this representation would take a minimum of 8+8+8=24 bytes per array element, along with an overhead of at least 24 bytes for each sub-list.

   Criticise this choice of data structure in terms of memory usage, and outline a more efficient choice. Your answer should describe the benefits of the representation, and **briefly** outline the important

fields in a structure definition necessary to hold your datastructure. You should describe the role of each important field in similar terms to that above.

You may assume the array will be rectangular and only contain elements of float64 type.

[8]

The linked list will require storing pointers to elements [#1] resulting in an overall size much larger than required to store just the elements themselves [#1] (at least 3x as large). A more efficient choice would be a strided array [#1]; this has a small fixed overhead compared to storing the values of the elements themselves [#1], giving memory use of close to 8 bytes per `float64` for large arrays [#1]

The data structure would require, at minimum:

* A pointer to a block of memory containing the array elements [#1]

* A stride array, giving the offset to jump, in bytes, to increment in any given dimension [#1]

* A shape array, giving the size of each dimension of the array as a sequence of integers [#1]

Also would require the total number of dimensions

(e) A tensor X has dtype float64 and has shape [2,2,2] with C (row-major) layout, and is stored using a strided array structure. At what byte offset from the start of the numerical data of X will the element referred to by index `X[1,1,0]` be found? In your answer, write down the stride array, show how the computation is performed and identify the memory offset, in bytes.

[5]

- strides = [32, 16, 8] [#2]
- indexing requires computing `(1*strides[0] + 1*strides[1] + 0*strides[2])` [#1]
- This is 32*16+0 = 48 bytes after the start of data [#2]

(f) Contrast the time-complexity of transposition of 2D arrays in the linked list case of (d) to that in the strided representation, in terms of the total number of array elements N. Explain how this is achieved.

[3]

The time complexity of linked list transposition will be at least O(N), as an iteration over every element is required. [#1] Transposition of a strided array takes constant time, O(1), [#1] as the operation only requires swapping the order of the strides and does not alter the array elements at all [#1].

2.

Cosmic ray events strike an aircraft at random intervals. The intensity of each event is random. The damage the cosmic ray does to an aircraft is modeled as the square of the intensity of the cosmic ray event (in other words a doubling of ray intensity means a quadrupling of damage to the aircraft).

(a) Assume that the **intensity** of each ray event is represented by a random variable X. X is a univariate continuous random variable with a density function $f_X(x)$. Write down the expected damage to the aircraft from a single cosmic ray event.

[4]

The expectation $\mathbb{E}[X^2] = \int_x f_X(x)x^2$

- [#1] for integral over x
- [#2] for x^2
- [#1] for $f_X(x)$

(b) To get a sense of the intensity distribution, you are asked to prepare a visualization that will let analysts eyeball the data and get a sense of the unknown density function. The input is one long vector **`events`**, consisting of thousands of measurements of the intensity of ray events.

(i) Give a statistical transform ("stat" in the language of the Layered Grammar of Graphics) which would be appropriate to transform **`events`** for plotting to show the approximate density. Describe what effect it would have, and any parameters of the stat that might need to be adjusted.

[3]

A binning operation [#1] (histogram would be acceptable even if inaccurate) would group the events into bins and count the number of events falling into each bin [#1]. The spacing of bins must be configured to cover a reasonable range without being too sparse [#1].

(ii) A model of intensity is proposed, which gives an approximation of $f_X(x)$ as a normal distribution. Sketch a visualisation that will use the stat described in (i) to represent the raw data and provide a visual comparison with the proposed approximation, which is a continuous function of *x*. Include all relevant details and caption the visualisation. **The specific shape of the graph is not relevant, as the details of the data are *not* provided.**

[5]

The graph should include:
Axis labels for the counts (y-axis) and intensity (x-axis). [#1]
Some visual indication of the histogram as bars, and indication of the intensity model, as a smooth curve [#1]

A legend which specifies two different layers in the graph. [#1]
Layering of two different geoms -- one for histogram, one for the approximation model. (alternatively, these could be plotted on two subfigures in a faceted style) [#1]
A caption [#1]

(c) It is assumed that the number of the cosmic ray events that occur in one hour can be modeled as a particular kind of probability distribution, which has a single parameter λ. This is a discrete distribution with a PMF that gives P(N=n| λ), where n is the number of events observed in one hour.

Without knowing anything further about this distribution, outline in a few sentences how **iterative optimisation** could be used to estimate a value for λ given a series of measurements of the counts of cosmic ray measurements in sequential hours $[n_1, n_2, \dots]$.

[3]

Maximum likelihood estimation [#1] uses optimisation to searching for a setting of parameters [#1] (here, just λ) that is most likely given the observations by evaluating the likelihood function for various configurations of the parameters [#1]

(d)

(i) State the philosophical differences between Bayesian and frequentist models of probability.

[2]

Bayesians treats probability as a calculus of belief [#1] and frequentists treat probability as the long-term average of a series of experiments [#1].

(ii) Optimising to find a single value of λ is a frequentist approach to estimation. Why might a Bayesian criticise this approach and what benefit might a Bayesian approach bring?

[2]

A Bayesian would be concerned with learning a distribution over possible values of λ, rather than a single likely value. [#1] This would capture how certain the estimate of λ was. [#1]

(iii) A Bayesian updating procedure is used to estimate new values for λ_i as each hour of event counts are observed. Each hour is indexed *i*. Write down the Bayesian update for the posterior distribution $P(\lambda_i|n_i)$ after observing counts for a count n_i, in hour *i*, *in terms of* a prior $P(\lambda_i)$ and the likelihood function $P(N = n_i|\lambda_i)$.

[6]

$$P(\lambda_i|n_i) = \frac{P(N = n_i|\lambda_i)P(\lambda_i)}{P(n_i)}$$

[#4] where:

$$P(n_i) = \int_{\lambda_i} P\,(N = n_i|\lambda_i)P(\lambda_i),$$

[#2] for getting evidence term from the integration over lambda_i

3.

(a) List the three elementary operations that can be performed on vectors in a vector space equipped with a norm.

[3]

- Vector addition **x+y** [#1]
- Scalar multiplication a**x** [#1]
- Norm (length) of a vector ||x|| [#1]

(b) You are employed to analyse the data from a sonar mine detecting system. Each time it is used to capture a sample, this system emits a sound pulse into the ground. Following this pulse, a microphone measures the amplitude of the sound reflected back every millisecond for 64 milliseconds. Each sample is stored as a 64 element floating point vector, representing the amplitudes at each time interval.

You have a collection of 10,000 test samples collected in the field, each of which is a 64D vector. These are collected into 10000×64 2D array X. A separate set of data has been collected from a specific type of anti-personnel mine, and these vectors have been collected into a 500×64 2D array Y.

Assume each row of X is represented within a topological vector space of 64 dimensions. Describe, in terms of **elementary operations**, how a *specific element* of X that is a good "average representation" of the set of vectors collected in Y could be found. State any appropriate equations in your answer.

[5]

The mean vector [#1] of Y can be computed using vector addition of each yi [#1] and scalar multiplication of the total by $1/N_y$ [#1]

The nearest element of X can be computed by finding the minimum norm of the difference between each vector in X and this mean vector μ [#1]

$$\text{argmin}_i ||\mu - X_i||$$

[#1] mark is awarded if a norm used, even without stating argmin.

(this can be stated as the sum of $\mu + cX_i$ with $c = -1$, but this is not required)

(c) Write down NumPy code that implements the solution above and evaluates to the respective element of X. You *do not* have to restrict yourself to elementary vector operations. Assume the Euclidean norm.

[3]

```
X[np.argmin(np.linalg.norm(X-np.mean(Y)))]
```

- [#1] mark for argmin
- [#1] mark for computing norm
- [#1] mark for subtraction of mean vector from X

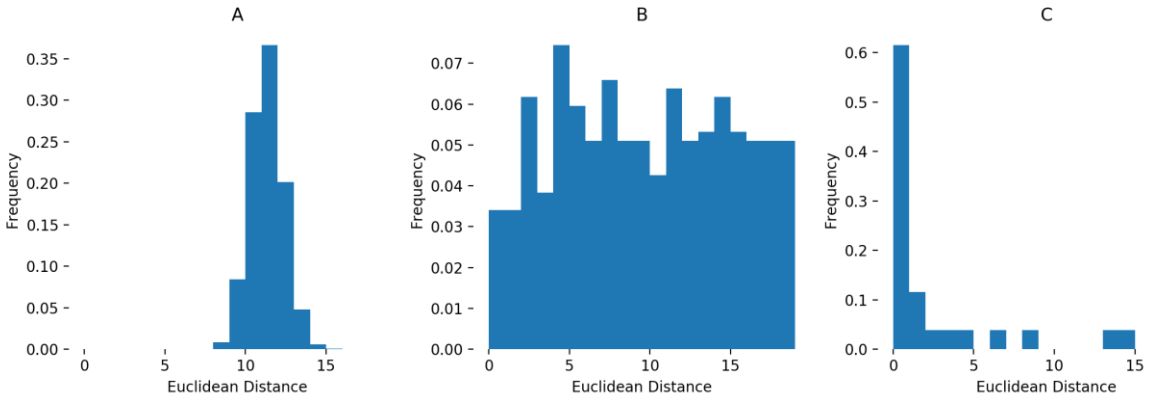
syntax does not have to be exact to get full credit

(d) A colleague suggests using a 64D histogram to estimate the density of values of X across the whole 64D space. Explain the relevance of the **curse of dimensionality** to this approach and discuss whether this is likely to be viable. Explain why or why not.

[4]

The curse of dimensionality is a consequence to the fact the volume of space is exponential in the number of dimensions [#2]. This makes generalising in high dimensional spaces difficult as data will be sparse [#1]. In this case, there are 64 dimensions; even splitting each dimension into just two bins would require 2^64 bins -- and there are only 10,000 data points. Almost every bin will be empty [#1].

(e) The same colleague runs three different scripts to generate a histogram of the Euclidean distances of each vector in X from every other vector in X. Two of the scripts are believed to be buggy. Given the three graphs below, explain which script looks *most likely* to be working correctly, and explain why.



[3]

A is the most likely to be correct [#2]. We would expect almost all distances to be clustered in a small set of distances in a high dimensional space [#1].

(f) It is suggested that the data, although measured with 64 dimensions, really only varies in two or three important ways, and the other dimensions are redundant measurements of these. It is further suggested that the **eigenspectrum of the covariance matrix** of X would help assess whether this is a correct hypothesis, by revealing how many of the dimensions of X are really independent.

Assuming the covariance matrix of X is computed as $\Sigma = cov(X)$

(i) Outline an iterative algorithm for finding the leading eigenvector $\mathbf{x_1}$ of the matrix Σ.

[5]

Power iteration can be used to find the leading eigenvector [#1]

This involves:

* choosing a random vector x_0 [#1]

* repeating the operation:

$$\mathbf{x_n} = \frac{\sum \mathbf{x_{n-1}}}{\|\mathbf{x_{n-1}}\|}$$

[#2]

* Repeat this operation until change in x_n from x_{n-1} becomes very small. [#1]

(ii) Explain how the leading eigenvalue λ_1 can be computed from this leading eigenvector $\mathbf{x_1}$ (you may assume this eigenvalue will be real).

[2]

By definition $\Sigma \mathbf{x_1} = \lambda_1 x_1$ therefore we can simply compute the ratio $\lambda_1 = \frac{\Sigma \mathbf{x_1}}{\mathbf{x_1}}$ [#2].