Week 6 exercises

This is the fourth page of *assessed* questions, as described in the background notes. These questions form 70% of your mark for Week 6. The introductory questions in the notes and the Week 6 discussion group task form the remaining 30% of your mark for Week 6.

Unlike the questions in the notes, you'll not immediately see any example answers on this page. However, you can edit and resubmit your answers as many times as you like until the deadline (Friday 30 October 4pm UK time, **now UTC**). This is a *hard deadline*: This course does not permit extensions and any work submitted after the deadline will receive a mark of zero. See the late work policy.

Queries: Please don't discuss/query the assessed questions on hypothesis until after the deadline. If you think there is a mistake in a question this week, please email Iain. You may ask about the computational complexity of other algorithms in the notes, for example relating to Bayesian regression and GPs.

Please only answer what's asked. Markers will reward succinct to-the-point answers. You can put any other observations in the "Add any extra notes" button (but this is for your record, or to point out things that seemed strange, not to get extra credit). Some questions ask for discussion, and so are open-ended, and probably have no perfect answer. For these, stay within the stated word limits, and limit the amount of time you spend on them (they are a small part of your final mark).

Feedback: We'll return feedback on your submission via email by Friday 6 November.

Good Scholarly Practice: Please remember the University requirements for all assessed work for credit. Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (permitting access only to yourself). You may not publish your solutions after the deadline either.

1 Computational cost of ML algorithms we've seen

It's useful to have at least a rough idea of how the algorithms that you learn scale to large problems. Both because it's useful for ruling out some algorithms, and because technical interviews for some jobs ask about the scaling of algorithms!

Computer scientists often use "big-O notation" to describe how algorithms scale. However, applied papers usually use big-O notation more sloppily than in a formal complexity class. You don't need to study the technical details to answer this question. The examples given here should be sufficient.

Big-O complexities usually don't contain multiplicative constants. If each element of a length-D vector is created from five additions, we say creating the vector costs $O(D)$, not $O(5D)$.

We also usually drop terms that aren't significant asymptotically. If an algorithm has 3 stages that take $5ND^2$ operations, $200ND$ operations, and (D^3+10) operations, we say the cost is $O(ND^2 + D^3)$. We've dropped constants. For large enough D, the $5ND^2$ becomes more important than the $200ND$ term (despite the smaller constant), so we ignore the less important term. We keep both ND^2 and D^3, because the first term is important for large N, and the second term becomes more important for large D.

Assume the following "big-O" computational complexities: matrix-matrix multiplication AB costs $O(LMN)$ for $L\times M$ and $M\times N$ matrices A and B. Inverting an $N\times N$ matrix G and/or finding its determinant costs $O(N^3)$.[1]

1. Good implementations usually take a factorization of G rather than inverting it, but the complexity story doesn't change: The factorizations also cost $O(N^3)$. Given that factorization, we can compute the determinant cheaply and $G^{-1}H$ in $O(N^2K)$, where H is $N\times K$ – the same cost as multiplying by the inverse.

Keep your analyses simple but useful, as an applied computer scientist would. While you could (for example) say that matrix-matrix multiplication of two $N\times N$ matrices is $O(N^{42})$ (because formally big-O provides an upper bound), in practice people quote the smallest bound they reasonably can. On the other hand, while large matrix multiplication can be done faster in theory than quoted above, please base your results on the straightforward complexities given (as is common practice).

As an example, consider Gaussian process regression, in particular mean prediction, equation (23). In that equation we can precompute $M^{-1}\mathbf{y}$ at training time. So at test time, the most expensive operation in mean prediction is computing N kernel values $\mathbf{k}^{(*)}$, which is $O(ND)$ for the Gaussian kernel. At training time, we need to compute N^2 kernel values costing $O(DN^2)$, and invert/factor an $N\times N$ matrix $O(N^3)$. So training complexity is $O(DN^2 + N^3)$.

2 The questions

For each of parts a-c), state a complexity and then give as brief a justification as you can, while still making it clear how you got your result. Brief but clear answers will be rewarded over unnecessarily-long or unclear answers. You do not need to list every operation, you can state what the most important operations are, and their costs.

a) What is the big-O complexity of performing N stochastic gradient descent training updates for a logistic regression classifier with D-dimensional feature vectors? [15 marks]

   [The website version of this note has a question here.]

b) What is the big-O complexity of fitting a Gaussian Bayes classifier to N training points in D-dimensions, with K classes? [15 marks]

   [The website version of this note has a question here.]

c) What is the big-O complexity of predicting the class of single test point, using the Gaussian Bayes classifier above? Assume that you have pre-computed everything you can at training time, before seeing the test point. [15 marks]

   [The website version of this note has a question here.]

d) Briefly discuss how you would choose between logistic regression and a Gaussian Bayes classifier with 2 classes, using the above results, and anything else you think is relevant. We're looking for how to choose which one to try first, if just trying one – not how to experimentally compare them based on validation performance. If there are things that are difficult to know in advance, then state that as part of your answer. [25 marks]

   Maximum 250 words.

   [The website version of this note has a question here.]