CSE Ph.D. Qualifying Exam, Spring 2009

You should choose two areas to work on. Each area consists of four problems, and you can choose three of them. Show all your work and write in a readable way.

1 Data Analysis

1. Suppose we use support-vector machines with the kernel:

$$K(x,z) = \begin{cases} 1 & \text{if } x = z, \\ 0 & \text{otherwise.} \end{cases}$$

This can be considered as mapping each x to a vector $\psi(x)$ in some high-dimensional feature space (that need not be specified) so that $K(x,z) = \psi(x)^T\psi(z)$. We are given m training examples $(x_1, y_1), \ldots, (x_m, y_m)$ with $y_i = \pm 1$. For simplicity, assume the x_i's are distinct, i.e., $x_i \neq x_j, i \neq j$. We only consider SVM where the hyperplane in the feature space goes through the origin, i.e., the intercept $b = 0$.

   (a) Recall the weight vector w used in SVM has the form

$$w = \sum_i \alpha_i y_i \psi(x_i)$$

   Compute the α_i's explicitly that would be found using SVMs with this kernel.

   (b) Recall that the SVM algorithm outputs a classifier that, on input x, computes the sign of $w^T\psi(x)$. What is the value of this inner product on training example x_i? What is the value of this inner product on any example x not seen during training? Based on these answers, what kind of generalization error do you expect will be achieved by SVMs using this kernel?

   (c) Recall that the generalization error of SVMs can be bounded using the margin δ (which is equal to $1/\|w\|$), or using the number of support vectors. What is δ in this case? How many support vectors are there in this case? How are these answers consistent with your answer in part (b)?

2. Markov chain monte carlo (MCMC) are a collection of sampling algorithms used to approximate intractable summations and integrals. They start from a sample from an initial distribution f and repeatedly generate new samples using the markov chain transition kernel T until reaching a sample from the stationary distribution p.

   (a) The detailed balance property of T with respect to p specifies that $p(r)T(r,w) = p(w)T(w,r)$, $\forall r, w$. Prove that if this property holds then p is a stationary distribution of T i.e., $\int p(r)T(r,w)\,dr = p(w)$. What additional conditions are required for ergodicity or T? Show that if $T_1, \ldots, T_m$ satisfy the detailed balance property than so does a linear combination $\sum \alpha_i T_i$.

(b) Metropolis-Hastings is an MCMC method that generates the next step $z^{(t+1)}$ given $z^{(t)}$ as follows:

$$z^{(t+1)} = \begin{cases} z' & \text{with probability } r(z^{(t)}, z') = \min\left(1, \frac{p(z')}{p(z^{(t)})} \frac{q(z^{(t)}|z')}{q(z'|z^{(t)})}\right) \\ z^{(t)} & \text{with probability } 1 - r(z^{(t)}, z') \end{cases}$$

where $z' \sim q(z|z^{(t)})$ and q is the proposal distribution. Derive the transition kernel T corresponding to this method and show that it satisfies the detailed balance property.

(c) Consider metropolis-Hastings sampling using a Gaussian proposal $q(z|z') = N(z', \sigma^2 I)$ and a specific p. What is the tradeoff involved with increasing or decreasing σ. Describe how you would choose σ in practice.

3. Assume we have l labeled documents and u unlabeled ones (the label is missing) $(x^{(1)}, y^{(1)}), \ldots, (x^{(l)}, y^{(l)}), x^{(l+1)}, \ldots, x^{(l+u)}$, where all documents from class r were generated from a multinomial or naive Bayes distribution with parameter $\theta^{(r)}$.

(a) What is the maximum likelihood estimator for $\theta^{(r)}$ using only the labeled data.

(b) What is the mean squared error of this estimator?

(c) We can construct an estimator for $\theta^{(1)}, \ldots, \theta^{(k)}$ that uses the unlabeled as well as the labeled data by maximizing the likelihood of the observed data. Write down the loglikelihood function in this case.

(d) Under what conditions would the MLE in case (a) converge to the true parameter? Can these conditions be relaxed if we also use unlabeled data (c)?

In your answers please use V to denote the vocabulary, k to denote the number of classes, and $c(x^{(i)}, w)$ to denote the number of times word w appeard in document $x^{(i)}$.

4. Would you expect k-fold cross-validation to work well for time series data? Why or why not? Describe at least one possibility for a scheme that might work better, with intuition for why it might.

2 High Performance Computing

1. **Parallel all-pairs shortest paths**: Given a graph represented by the adjacency matrix, A, where entry $A(i,j)$ is the weight of edge connecting vertex i to vertex j, we wish to compute a new matrix D such that $D(i,j)$ is the length of the shortest path between i and j. Recall Floyd's algorithm:

   $D \leftarrow A$
   for $k \leftarrow 1, 2, \ldots, n$ **do**
     for $i \leftarrow 1, 2, \ldots, n$ **do**
       for $j \leftarrow 1, 2, \ldots, n$ **do**
         $D(i,j) \leftarrow \min\{D(i,j), D(i,k) + D(k,j)\}$
       end for
     end for
   end for

   Given an efficient parallel version of this algorithm for distributed memory machines, assuming that the graph is nearly dense. Be sure to explain how you distribute the graph (A) in your scheme. Analyze the per-processor storage costs as well as the time of your algorithm in the $\alpha - \beta$ model, where α is the message latency of sending a message (units of time) and β is the network bandwidth (units of words per time), assuming no overlapping of computation and communication. (Scoring: 50% for a correct algorithm, including correctness justification, and 50% for your analysis.)

2. **Parallel overhead**:

   (a) Show the following facts about the overhead, $T_o = p \cdot T_p - T_1$, of some problem whose serial execution time is T_1 and whose parallel execution time on p processors is T_p.

      i. (15%) Suppose T_o scales at worse linearly in the number of processors, p. That is, suppose $T_o \leq \Theta(p)$. Show that the parallel execution time will decrease as p increases, and furthermore, that it asymptotically approaches a constant value.

      ii. (25%) Suppose $T_o > \Theta(p)$. Show that T_p decreases with increasing p at first but then eventually begins increasing, *i.e.*, that there is a minimum.

   (b) Suppose we wish to add n numbers using p processors, where each processor initially has $\frac{n}{p}$ elements (assume for simplicity that p divides n). Suppose an addition costs δ cycles, and that communication can be described by the α-β model (with α in cycles and β in words/cycle).

      i. (10%) Compute the serial execution time, T_1.

      ii. (10%) Describe an efficient parallel algorithm for this problem. What is its running time, T_p?

      iii. (10%) Compute the speedup, $\frac{T_1}{T_p}$.

iv. (10%) Compute the overhead, $T_o = p \cdot T_p - T_1$.

v. (20%) For what value of p is T_p minimized for your parallel algorithm? What does this value of p say about the quality of your algorithm?

3. **Amdahl's Law for heterogeneous multicore**: Recall Amdahl's law: For a program where a fraction f of the total work must be executed sequentially, we can bound the speedup on a p processor system as follows:

$$\begin{aligned} \text{Speedup}(f,p) &\leq \frac{\frac{1}{\rho_0}}{\frac{f}{\rho_0} + \frac{1-f}{p \cdot \rho_0}} \\ &= \frac{1}{f + \frac{1-f}{p}} \end{aligned} \tag{1}$$

where ρ_0 is the peak processing rate on one processor. (Let's ignore effects that could lead to superlinear speedup.)

Intelis designing a new *heterogeneous* multicore chip, code-named the Atlanta, and has asked for some advice in the design. In particular, rather than giving you p homogeneous cores, Intel says it can fit $\frac{p}{r}$ larger cores on a chip, where each large core has performance $\rho_0\sqrt{r}$. (Assume $r \geq 1$ and that r divides p.)

(a) (40%) In design A, Intel will fill the chip with r large cores. Estimate $\text{Speedup}(f,p,r)$ of your application on the Atlanta over a single *large* core.

(b) (40%) In design B, suppose instead that we use a heterogeneous design consisting of 1 large core, with the remaining space for smaller cores. Estimate the speedup for this design.

(c) (20%) Suppose $p = 64$, and $r = 8$. At what value of f is design B better than design A?

4. **Parallel wavelet transform**: You have been asked to implement the 1-D biorthogonal wavelet transform (used in the JPEG2000 standard) on a shared memory machine with a non-uniform memory access (NUMA) architecture. Given an array X of length n, the pseudocode is:

```
// Even phase 1; a & b are constants
for all even numbered points i do:
  x[i] = a*x[i-1] + b*x[i+1]
// Odd phase 1; c & d are constants
for all odd numbered points i do:
  x[i] = c*x[i-1] + d*x[i+1]
// Even phase 2; e & f are constants
for all even numbered points i do:
  x[i] = e*x[i-1] + f*x[i+1]
```

```
// Odd phase 2; g & h are constants
for all odd numbered points i do:
  x[i] = g*x[i-1] + h*x[i+1]
// Scale; s is a constant
for all points i do:
  x[i] = s*x[i];
```

(a) (40%) Though this algorithm appears straightforward to parallelize, there are a number of potential performance issues, particularly for shared memory machine. Describe these issues and how you might in general address them.

(b) (60%) Based on the issues you raised above, describe a parallel algorithm/implementation for this kernel. Justify your approach.

3 Discrete Algorithms

When asked to describe an algorithm, give concise pseudo-code that includes the essential steps in your algorithm. You can give a sequential or parallel algorithm (your choice), unless specified otherwise.

The clarity of your write-up will influence your final grade significantly.

1. Let $G = (V, E)$ be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct.

   Let $T = (V; E')$ be a spanning tree of G. Let the bottleneck edge of T be the edge of T with the greatest cost. A spanning tree T of G is a minimum-bottleneck spanning tree if there is no spanning tree T' of G with a cheaper bottleneck edge.

   (a) Is every minimum-bottleneck tree of G a minimum spanning tree of G? Prove or give a counter example.

   (b) Is every minimum spanning tree of G a minimum-bottleneck tree of G? Prove or give a counter example.

   (c) Give an $O(n + m \log n)$-work construction algorithm for a minimum-bottleneck spanning tree.

2. Let the array representation of a tree be specified by n links PARENT$[j]$ for $1 \leq j \leq n$. (Assume that the parent of the root is the root itself). Give an algorithm that constructs a double linked circular list containing the nodes of this tree in preorder traversal and reports the root of the tree. Give the work complexity of your algorithm. E.g.

$$\begin{aligned} j &= 1, 2, 3, 4, 5, 6, 7, 8 \\ \text{PARENT}[j] &= 3, 8, 4, 4, 4, 8, 3, 4 \end{aligned}$$

   Then, the left L and right R links should be

$$\begin{aligned} L[j] &= 3, 8, 4, 6, 7, 2, 1, 5 \\ R[j] &= 7, 6, 1, 3, 8, 4, 5, 2. \end{aligned}$$

   (Circular list is a linked list in which the last node links back to the first node.)

3. Give an algorithm for finding a maximum bipartite matching and give a brief worst-case complexity analysis as a function of V and E, where V is the number of vertices and E is the number of edges. Prove the correctness of your algorithm.

4. Give a parallel merge algorithm for two sorted arrays of length $n = 2^l$ using the PRAM model. Give the time and work complexity of your algorithm as a function of n and the number of processors p. Discuss the work-optimality of your algorithm.