A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering

Pedro Domingos PEDROD@CS.WASHINGTON.EDU
Geoff Hulten GHULTEN@CS.WASHINGTON.EDU
Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195, U.S.A.

Abstract

We propose to scale learning algorithms to arbitrarily large databases by the following method. First derive an upper bound for the learner's loss as a function of the number of examples used in each step of the algorithm. Then use this to minimize each step's number of examples, while guaranteeing that the model produced does not differ significantly from the one that would be obtained with infinite data. We apply the method to K-means clustering, and empirically observe its speedup relative to the standard version on large databases.

1. Introduction

Traditionally, the bottleneck preventing the development of more-intelligent systems via machine learning was the limited data available. However, in many domains the size of the datasets available is now so large that the limiting factor is learners' inability to use all the data in the available time. We propose to overcome this problem by developing learners that are effectively able to learn from infinite data in finite time. By this we mean that, in finite time, they learn models that are indistinguishable from those they would learn given infinite data. Our method is based on bounding the finite-data model's loss relative to the infinite-data one as a function of the number of examples used at each learning step, and then minimizing these numbers of examples while guaranteeing that the bound is preserved. We apply the method to the design of VFKM, a faster version of K-means clustering. Empirical studies show the advantage of VFKM relative to standard K-means, and the quality of the bounds obtained. We then apply VFKM to clustering a large database of Web page requests for a distributed caching application. The paper concludes with a discussion of related work and a summary of contributions and future research directions.

2. The Method

In this section we present our method for speeding up a learner while guaranteeing that its loss relative to the infinite-data case remains bounded.

Let learning algorithm A consist of a sequence of steps $s_1, s_2, \ldots, s_i, \ldots$. Assume A is given a training set S composed of N i.i.d. (independent and identically distributed) examples. Let $n_i \leq N$ be the number of examples used in step s_i, and let $\vec{n} = (n_1, n_2, \ldots, n_i, \ldots)$. If step s_i consists of making a decision between a finite number of choices, let δ_i be the probability of making the wrong decision at that step. If s_i consists of estimating a real value, let δ_i be the probability that the true value differs from the estimated one by more than ϵ_i. Express δ_i and ϵ_i as functions of n_i using Hoeffding or similar bounds: $\delta_i = f(n_i)$, $\epsilon_i = g(n_i)$ (Hoeffding, 1963). Let $L(M_1, M_2)$ be a loss function penalizing the difference between models M_1 and M_2. Let M_∞ be the model learned by A using infinite data at each step (i.e., with $\forall i \; n_i = \infty$). Let $M_{\vec{n}}$ be the model learned by A using n_i examples in step s_i. Derive a bound on $L(M_{\vec{n}}, M_\infty)$ as a function of δ_i and ϵ_i (if applicable), and therefore as a function of n_i, for all steps of A: $L(M_{\vec{n}}, M_\infty) \leq G_{A,S}(\vec{n})$ with probability at least $1 - F_{A,S}(\vec{n})$, where the subscripts A and S indicate that the functions F and G depend on the learner A, and may depend on the training set S. When A is run on S, $F_{A,S}(\vec{n})$ and $G_{A,S}(\vec{n})$ can be used to inform the user of the quality of the results. Let the user requirements be that $L(M_{\vec{n}}, M_\infty)$ be at most ϵ with probability at least $1 - \delta$. If $G_{A,S}(\vec{n}) \leq \epsilon$ and $F_{A,S}(\vec{n}) \leq \delta$, the training set is sufficiently large; otherwise it may be advisable to gather more data. Conversely, we can use knowledge of $F_{A,S}(\vec{n})$ and $G_{A,S}(\vec{n})$ to determine $\vec{n}$ that minimizes A's running time $T(\vec{n})$ while satisfying user requirements (δ, ϵ). Where feasible, this method should result in speedup factors that increase without limit as the training set size grows, without significant loss in model performance according to the given criteria.

For example, in decision tree induction the basic step is deciding which attribute to test at a node, and δ_i is the probability that the wrong attribute is chosen for the ith node ("wrong" in the sense that it is different from the attribute that would have been chosen at the node given infinite data). In Domingos and Hulten (2000) we applied a more primitive version of the method above to decision tree induction under a variety of loss functions (difference in error rate between the trees, probability that they make different predictions, probability that a random example follows different paths in the two trees). This allowed the resulting algorithm to learn on databases several orders of magnitude larger than those feasible for C4.5, with corresponding gains in accuracy. In this paper we generalize the previous treatment and apply it to K-means clustering.

3. A Loss Bound for K-Means

K-means (MacQueen, 1967) is one of the most widely used clustering algorithms. It is shown in pseudo-code in Table 1. The output of K-means is a set of K cluster centroids $C = \{c_1, \ldots, c_K\}$. A natural loss function to use when comparing two clusterings C_1 and C_2 having the same number of clusters K is[1]

$$L(C_1, C_2) = \sum_{k=1}^{K} ||c_{1,k} - c_{2,k}||^2 \qquad (1)$$

In this section we develop a bound for this loss as a function of the number of examples n_i used in each iteration of K-means. In the standard version of K-means described in Table 1, $\forall i \;\; n_i = N$. In the next section we develop a faster version of K-means that uses only as many examples in each iteration as required to meet pre-specified user requirements on $L(C_{\vec{n}}, C_\infty)$.

Consider a real-valued random variable x whose range is R. Suppose we have made n independent observations of this variable, and computed their mean $\bar{x}$. The Hoeffding bound (Hoeffding, 1963) states that, with probability at least $1 - \delta$, the true mean of the variable is within ϵ of $\bar{x}$, where

$$\epsilon = \sqrt{\frac{R^2 \ln(2/\delta)}{2n}} \qquad (2)$$

The Hoeffding bound has the very attractive property that it is independent of the probability distribution generating the observations. The price of this generality is that the bound is more conservative than distribution-dependent ones (i.e., it will take more observations to reach the same δ and ϵ). Although our method is in principle applicable with any relevant bound, in this paper we will use the Hoeffding bound, for the sake of generality.

[1]This assumes the correspondence between centroids is given. If not, the loss should be the minimum of this measure over all possible correspondences; in practice, the best correspondence is typically found by greedy search.

Table 1. The K-means clustering algorithm.

```
Inputs:  $S$ is a set of examples,
         $K$ is a desired number of clusters,
         $\gamma$ is a convergence threshold.
Output:  $C$ is a set of cluster centroids $\{c_1, \ldots, c_K\}$.

Procedure K-Means ($S, K, M$)
i = 0.
For $k = 1$ to $K$
   Let $c_{k0}$ be a random example from $S$.
Repeat
   i = i + 1.
   Let $S_{ki} =$
      $\{x \in S \,|\, \forall (k' \neq k) \; ||x - c_{k,i-1}|| \leq ||x, c_{k',i-1})||\}$.
   For $k = 1$ to $K$
      $c_{ki} = \frac{1}{|S_{ki}|} \sum_{x \in S_{ki}} x$.
Until $\sum_{k=1}^{K} ||c_{ki} - c_{k,i-1}||^2 \leq \gamma$.
Return $C = \{c_{1i}, \ldots, c_{Ki}\}$.
```

At each iteration, K-means estimates the position of each centroid as the average of the examples it wins. There are two sources of error in this estimate. One is the finite number of examples used; for each coordinate of the cluster centroid c_{kd}, this type of error is bounded by Equation 2, with n as the number of examples won by the cluster and R as the coordinate's range (assumed known or reasonably approximated). We call this type of error *sampling error*. The other source of error is that the centroid positions at the beginning of the iteration may not be the correct ones, potentially resulting in examples being assigned to the wrong cluster. (By "correct" centroid positions we mean those that would be obtained by K-means with infinite data.) We call this type of error *assignment error*. The initialization step is error-free, assuming the finite- and infinite-data K-means are initialized with the same centroids. Thereafter the assignment errors in one iteration are a function of the total errors in the previous iteration.

Let D be the dimensionality of the domain. Let c_{kdi} be the infinite-data value of the dth coordinate of the kth centroid at iteration i, $\hat{c}_{kdi}$ be the corresponding finite-data estimate, and $\bar{c}_{kdi}$ be the estimate that would be obtained if there were no assignment errors in iteration

i. Let ϵ_{kdi} be a bound on the total error in estimating c_{kdi}: $\hat{c}_{kdi} - \epsilon_{kdi} \leq c_{kdi} \leq \hat{c}_{kdi} + \epsilon_{kdi}$. Then the distance between the finite- and infinite-data centroids $||\hat{c}_{ki} - c_{ki}||$ is bounded by $\epsilon_{ki} = \sqrt{\sum_{d=1}^{D} \epsilon_{kdi}^2}$. Suppose that at iteration i example x is won by centroid $\hat{c}_{ki}$, which is at distance d_{ki} from it. Then, if there is another centroid $\hat{c}_{k'i}$ such that $d_{k'i} - \epsilon_{k'i} < d_{ki} + \epsilon_{ki}$, the example may have been incorrectly assigned to $\hat{c}_{ki}$. The number of examples $\tilde{n}_{ki}^+$ satisfying this condition is an upper bound on the number of examples n_{ki}^+ incorrectly assigned to $\hat{c}_{ki}$. Similarly, the number of examples $\tilde{n}_{ki}^-$ for which the condition $d_{ki} + \epsilon_{ki} < d_{k'i} - \epsilon_{k'i}$ holds, with $\hat{c}_{k'i}$ being the centroid that won the example ($k' \neq k$), is an upper bound on the number of examples n_{ki}^- that $\hat{c}_{ki}$ did not win but should have. Let x_{kdi}^+ be the average of the dth coordinate of the examples that $\hat{c}_{ki}$ won incorrectly, and similarly for x_{kdi}^-. Let $\hat{n}_{ki}$ be the number of examples that $\hat{c}_{ki}$ won. Then the number of examples that it should have won is $\hat{n}_{ki} - n_{ki}^+ + n_{ki}^-$, and

$$\begin{aligned}
&|\hat{c}_{kdi} - \bar{c}_{kdi}| \\
&= \left| \hat{c}_{kdi} - \frac{\hat{n}_{ki}\hat{c}_{kdi} - n_{ki}^+ x_{kdi}^+ + n_{ki}^- x_{kdi}^-}{\hat{n}_{ki} - n_{ki}^+ + n_{ki}^-} \right| \\
&= \left| \frac{n_{ki}^+(x_{kdi}^+ - \hat{c}_{kdi}) - n_{ki}^-(x_{kdi}^- - \hat{c}_{kdi})}{\hat{n}_{ki} - n_{ki}^+ + n_{ki}^-} \right| \quad (3)
\end{aligned}$$

An upper bound on this expression can be obtained by upper-bounding the numerator and lower-bounding the denominator. Since $0 \leq n_{ki}^+ \leq \tilde{n}_{ki}^+$ and $0 \leq n_{ki}^- \leq \tilde{n}_{ki}^-$, $\hat{n}_{ki} - n_{ki}^+ + n_{ki}^- \geq \hat{n}_{ki} - \tilde{n}_{ki}^+$. Let S_{ki}^+ be the set of examples that may have been incorrectly won by $\hat{c}_k$, and similarly for S_{ki}^-. Given an example x with coordinates x_d, let $\Delta x_{kdi} = x_d - \hat{c}_{kdi}$ if $x \in S_{ki}^+$ and $\Delta x_{kdi} = -(x_d - \hat{c}_{kdi})$ if $x \in S_{ki}^-$. Then an upper bound on the numerator of Equation 3 is $\max\{\sum_{x|\Delta x_{kdi}>0} \Delta x_{kdi}, \sum_{x|\Delta x_{kdi}<0} |\Delta x_{kdi}|\}$, since the absolute value of a sum is upper-bounded by either the sum of the positive elements or the sum of the absolute values of the negative elements. Thus

$$\begin{aligned}
&|\hat{c}_{kdi} - \bar{c}_{kdi}| \\
&\leq \left| \frac{\max\{\sum_{x|\Delta x_{kdi}>0} \Delta x_{kdi}, \sum_{x|\Delta x_{kdi}<0} |\Delta x_{kdi}|\}}{\hat{n}_{ki} - \tilde{n}_{ki}^+} \right| \quad (4)
\end{aligned}$$

This bound is tight in the sense that equality can occur, but given the information available in a specific run of K-means (positions of examples relative to centroids, etc.) it might be possible to obtain a tighter bound for most cases; we leave this as future work.

Let R_d be the dth coordinate's range, and n_{ki} be the number of examples that $\hat{c}_{ki}$ should have won (i.e., if there were no assignment errors). Then

$$|\bar{c}_{kdi} - c_{kdi}| \leq \sqrt{\frac{R_d^2 \ln(2/\delta)}{2n_{ki}}} \leq \sqrt{\frac{R_d^2 \ln(2/\delta)}{2(\hat{n}_{ki} - \tilde{n}_{ki}^+)}} \quad (5)$$

and

$$|\hat{c}_{kdi} - c_{kdi}| \leq |\hat{c}_{kdi} - \bar{c}_{kdi}| + \sqrt{\frac{R_d^2 \ln(2/\delta)}{2(\hat{n}_{ki} - \tilde{n}_{ki}^+)}} = \epsilon_{kdi} \quad (6)$$

where $|\hat{c}_{kdi} - \bar{c}_{kdi}|$ is bounded by Equation 4. The total error in c_{ki} is bounded by $\sqrt{\sum_{d=1}^{D} \epsilon_{kdi}^2}$. This can in turn be used to bound the assignment errors at the next iteration $|\hat{c}_{kd,i+1} - \bar{c}_{kd,i+1}|$ according to Equation 4, and so on recursively, starting from $\forall k, d\ \epsilon_{kd0} = 0$. If the finite- and infinite-data K-means converge in the same number of iterations m, the loss due to finite data is (see Equation 1)

$$L(C_{\vec{n}}, C_\infty) \leq \sum_{k=1}^{K} \epsilon_{km}^2 = \sum_{k=1}^{K} \sum_{d=1}^{D} \epsilon_{kdm}^2 \quad (7)$$

with the ϵ_{kdm}'s given by Equation 6. In general (with probability specified below), infinite-data K-means converges at one of the iterations for which the minimum possible change in centroid positions is below the convergence threshold γ (see Table 1), and is guaranteed to converge at the first iteration for which the maximum possible change is below γ. More precisely, it converges at one of the iterations for which $\sum_{k=1}^{K}\sum_{d=1}^{D}(\max\{|\hat{c}_{kd,i-1} - \hat{c}_{kdi}| - \epsilon_{kd,i-1} - \epsilon_{kdi}, 0\})^2 \leq \gamma$, and is guaranteed to converge at the first iteration for which $\sum_{k=1}^{K}\sum_{d=1}^{D}(|\hat{c}_{kd,i-1} - \hat{c}_{kdi}| + \epsilon_{kd,i-1} + \epsilon_{kdi})^2 \leq \gamma$. In order to obtain a bound for $L(C_{\vec{n}}, C_\infty)$, finite-data K-means must be run until the latter condition holds. Let M be the set of iterations at which infinite-data K-means could have converged. Then we finally obtain

$$L(C_{\vec{n}}, C_\infty) \leq \max_{i \in M} \left\{ \sum_{k=1}^{K} \sum_{d=1}^{D} (|\hat{c}_{kdi} - \hat{c}_{kdm}| + \epsilon_{kdi})^2 \right\} \quad (8)$$

where m is the total number of iterations carried out. This bound holds if all of the Hoeffding bounds for the ϵ_{kdi}'s hold. Since each of these bounds holds with probability at least $1 - \delta$, the bound above holds with probability at least $1 - \delta^* = (1 - \delta)^{kdm}$, (pessimistically) assuming independence. Notice that the exponential dependence of $1 - \delta^*$ on kdm is offset by the Hoeffding bound's logarithmic dependence on δ. The bound we have just derived utilizes run-time information, namely the distance of each example to each cluster in each iteration. This allows it to be tighter than *a priori* bounds. Notice also that it would be trivial to modify the treatment for any other loss criterion that depends only on the ϵ_{kd}'s (e.g., absolute loss).

4. A Fast K-Means Algorithm

We now apply the previous section's result to reduce the number of examples used by K-means at each iteration while keeping the loss bounded. We call the resulting algorithm VFKM. The goal is to learn in minimum time a clustering whose loss relative to K-means applied to infinite data is at most ϵ^* with probability at least $1 - \delta^*$. Using the notation of the previous section, if n_i examples are used at each iteration then the running time of K-means is $O(KD\sum_{i=1}^m n_i)$, and can be minimized by minimizing $\sum_{i=1}^m n_i$. Assume for the moment that the number of iterations m is known. Then, using Equation 1, we can state the goal more precisely as follows.

Goal: *Minimize $\sum_{i=1}^m n_i$, subject to the constraint that $\sum_{k=1}^K ||\hat{c}_{km} - c_{km}||^2 \leq \epsilon^*$ with probability at least $1-\delta^*$.*

A sufficient condition for $\sum_{k=1}^K ||\hat{c}_{km} - c_{km}||^2 \leq \epsilon^*$ is that $\forall k \;\; ||\hat{c}_{km} - c_{km}|| \leq \sqrt{\epsilon^*/k}$. We thus proceed by first minimizing $\sum_{i=1}^m \hat{n}_{ki}$ subject to $||\hat{c}_{km} - c_{km}|| \leq \sqrt{\epsilon^*/k}$ separately for each cluster.[2] In order to do this, we need to express $||\hat{c}_{km} - c_{km}||$ as a function of the $\hat{n}_{ki}$'s. By the triangle inequality,

$$||\hat{c}_{ki} - c_{ki}|| \leq ||\hat{c}_{ki} - \bar{c}_{ki}|| + ||\bar{c}_{ki} - c_{ki}|| \tag{9}$$

By Equation 5,

$$||\bar{c}_{ki} - c_{ki}|| \leq \sqrt{\sum_{d=1}^{D} \frac{R_d^2 \ln(2/\delta)}{2(\hat{n}_{ki} - \tilde{n}_{ki}^+)}} = \sqrt{\frac{R^2 \ln(2/\delta)}{2(\hat{n}_{ki} - \tilde{n}_{ki}^+)}} \tag{10}$$

where $R^2 = \sum_{d=1}^D R_d^2$ and $\delta = 1 - \sqrt[kdm]{1-\delta^*}$ per the discussion following Equation 8. By Equation 4,

$$||\hat{c}_{ki} - \bar{c}_{ki}|| \leq \frac{\sqrt{\sum_{d=1}^D X_{kdi}^2}}{\hat{n}_{ki} - \tilde{n}_{ki}^+} \tag{11}$$

where $X_{kdi} = \max\{\sum_{x|\Delta x_{kdi}>0} \Delta x_{kdi}, \sum_{x|\Delta x_{kdi}<0} |\Delta x_{kdi}|\}$. To keep the analysis tractable, we upper-bound $\tilde{n}_{ki}^+$ and the numerator of Equation 11 by functions proportional to $\hat{n}_{ki}\epsilon_{k,i-1}$. This captures the notion than the number of potentially incorrectly won examples in one iteration should increase with the total error in the previous one, and similarly for the effect of the incorrectly won or lost examples on $\sum_{x \in S_{ki}} x$. Thus, letting the proportionality factors be respectively b_{ki} and a_{ki}, and using Equations 10 and 11, Equation 9 becomes

$$\begin{aligned} ||\hat{c}_{ki} - c_{ki}|| &\leq \frac{a_{ki}\epsilon_{k,i-1}}{1 - b_{ki}\epsilon_{k,i-1}} + \sqrt{\frac{R^2 \ln(2/\delta)}{2\hat{n}_{ki}(1 - b_{ki}\epsilon_{k,i-1})}} \\ &= \epsilon_{ki} \end{aligned} \tag{12}$$

Approximating ϵ_{ki} by the first two terms of this expression's Taylor expansion around a point $\epsilon_{k,i-1}^0$, and discarding terms in the first derivative that become negligible as $\hat{n}_{ki}$ increases, we obtain

$$\epsilon_{ki} = \alpha_{ki}\epsilon_{k,i-1} + \beta_{ki} \tag{13}$$

with

$$\alpha_{ki} = \frac{a_{ki}}{(1 - b_{ki}\epsilon_{k,i-1}^0)^2} \tag{14}$$

$$\begin{aligned} \beta_{ki} &= \frac{a_{ki}\epsilon_{k,i-1}^0}{1 - b_{ki}\epsilon_{k,i-1}^0} + \sqrt{\frac{R^2 \ln(2/\delta)}{2\hat{n}_{ki}(1 - b_{ki}\epsilon_{k,i-1}^0)}} \\ &\quad -\alpha_{ki}\epsilon_{k,i-1}^0 \end{aligned} \tag{15}$$

Given Equation 13 and $\epsilon_{k0} = 0$, it can be shown by induction that

$$\epsilon_{km} = \sum_{i=1}^m \beta_{ki} \prod_{j=i+1}^m \alpha_{kj} = \sum_{i=1}^m \frac{r_{ki}}{\sqrt{\hat{n}_{ki}}} - r_k \tag{16}$$

where

$$\begin{aligned} r_{ki} &= \sqrt{\frac{R^2 \ln(2/\delta)}{2(1 - b_{ki}\epsilon_{k,i-1}^0)}} \prod_{j=i+1}^m \alpha_{kj} \\ r_k &= \sum_{i=1}^l \frac{a_{ki}b_{ki}(\epsilon_{k,i-1}^0)^2}{(1 - b_{ki}\epsilon_{k,i-1})^2} \prod_{j=i+1}^m \alpha_{kj} \end{aligned} \tag{17}$$

Since $||\hat{c}_{km} - c_{km}|| \leq \epsilon_{km}$ (Equation 12), the target bound will be satisfied by minimizing $\sum_{i=1}^m \hat{n}_{ki}$ subject to $\epsilon_{km} = \sqrt{\epsilon^*/k}$.[3] We can thus apply the method of Lagrange multipliers with the Lagrangian function (see Equation 16)

$$L(\vec{n}, \lambda) = \sum_{i=1}^m \hat{n}_{ki} + \lambda \left(\sum_{i=1}^m \frac{r_{ki}}{\sqrt{\hat{n}_{ki}}} - r_k - \sqrt{\frac{\epsilon^*}{k}} \right) \tag{18}$$

Equating to zero the gradient of $L(\vec{n}, \lambda)$ with respect to the $\hat{n}_{ki}$'s and λ and solving for the $\hat{n}_{ki}$'s yields

$$\hat{n}_{ki} = \left(\frac{\sum_{j=1}^m \sqrt[3]{r_{ki} r_{kj}^2}}{\sqrt{\frac{\epsilon^*}{k}} + r_k} \right)^2 \tag{19}$$

[2]This will generally lead to a suboptimal solution; improving it is a matter for future work.

[3]This may lead to a suboptimal solution for the $\hat{n}_{ki}$'s, in the unlikely case that $||\hat{c}_{km} - c_{km}||$ increases with them.

Let f_{ki} be the fraction of examples that cluster k wins in iteration i: $f_{ki} = \hat{n}_{ki}/n_i$. Then, for each cluster to win in iteration i the number of examples required by Equation 19, we need to make

$$n_i = \max_k \left\{ \frac{\hat{n}_{ki}}{f_{ki}} \right\} \quad (20)$$

with the $\hat{n}_{ki}$'s given by Equation 19.

The VFKM algorithm consists of a sequence of runs of K-means with each run using more examples than the last, until the bound $L(C_{\vec{n}}, C_\infty) \leq \epsilon^*$ is satisfied, with $L(C_{\vec{n}}, C_\infty)$ bounded according to Equation 8. In the first run, VFKM postulates a maximum number of iterations m, and uses it to set $\delta = 1 - \sqrt[kdm]{1-\delta^*}$. If m is exceeded, for the next run it is set to 50% more than the number needed in the current run. (A new run will be carried out if either the δ^* or ϵ^* target is not met.) The number of examples used in the first run of K-means is the same for all iterations, and is set to

$$n_i = 1.1\,\frac{K}{2}\left(\frac{R}{\epsilon^*}\right)^2 \ln\left(\frac{2}{\delta}\right) \quad (21)$$

This is 10% more than the number of examples that would theoretically be required in the best possible case (no assignment errors in the last iteration, leading to a pure Hoeffding bound, and a uniform distribution of examples among clusters). The numbers of examples for subsequent runs are set according to Equation 20. For iterations beyond the last one in the previous run, the number of examples is again set according to Equation 21. A run of K-means is terminated when the convergence criterion $\sum_{k=1}^K \sum_{d=1}^D (|\hat{c}_{kd,i-1} - \hat{c}_{kdi}| + \epsilon_{kd,i-1} + \epsilon_{kdi})^2 \leq \gamma$ is met (see discussion after Equation 7), or two iterations after the regular K-means criterion $\sum_{k=1}^K ||c_{ki} - c_{k,i-1}||^2 \leq \gamma$ is met, whichever comes first. The latter condition avoids overly long unproductive runs. If the user target bound is ϵ, ϵ^* in Equations 21 and 19 is set to $\min\{\epsilon, \gamma/3\}$, to facilitate meeting the first criterion above. If at any iteration $\tilde{n}_{ki}^+ = \hat{n}_{ki}$ for some cluster (making α_{ki} (Equation 14) and β_{ki} (Equation 15) undefined), we restart the run with twice the number of examples. When $\tilde{n}_{ki}^+ = \hat{n}_{ki}$ occurred in a run, or when the convergence threshold for infinite-data K-means was not reached even when the whole training set was used, VFKM reports that it was unable to find a bound; otherwise the bound obtained is reported.

VFKM ensures that the total number of examples used in one run is always at least twice the number N used in the previous run. This is done by, if $\sum n_i < 2N$, setting the n_i's instead to $n_i' = 2N(n_i/\sum n_i)$. If at any point $\sum n_i > m|S|$, where m is the number of iterations carried out and S is the full training set, $\forall i\ n_i = |S|$ is used. Thus, assuming that the number of iterations does not decrease with the number of examples, VFKM's total running time is always less than three times the time taken by the last run of K-means. (The worst case occurs when the one-but-last run is carried out on almost the full training set.)

The run-time information gathered in one run is used to set the n_i's for the next run. In each iteration, we record the $\hat{n}_{ki}$'s (and therefore the f_{ki}'s) and the actual bounds obtained ϵ_{ki}. These are used as the ϵ_{ki}^0's around which the Taylor expansion of the bound is carried out (see Equations 12-15). We compute each b_{ki} as $\tilde{n}_{ki}^+/(\hat{n}_{ki}\epsilon_{k,i-1}^0)$, and each a_{ki} as $\sqrt{\sum_{d=1}^D X_{kdi}^2}/(\hat{n}_{ki}\epsilon_{k,i-1}^0)$ (see Equation 11). The approximations made in the derivation will be good, and the resulting n_i's accurate, if the centroids' paths in the current run are similar to those in the previous run. This may not be true in the earlier runs, but their running time will be negligible compared to that of later runs, where the assumption of path similarity from one run to the next should hold.

5. Empirical Study

We conducted a series of experiments on synthetic datasets to compare VFKM with K-means. All datasets were generated by mixtures of spherical Gaussians with means μ_k in the unit hypercube. Each dataset was generated according to three parameters: the dimensionality D, the number of mixture components K, and the standard deviation of each coordinate in each component σ. The μ_k's were generated one at a time by sampling each dimension uniformly from the range $(2\sigma, 1-2\sigma)$. This ensured that most of the datapoints generated were within the unit hypercube. The range of each dimension in VFKM was set to one. Rather than discard points outside the unit hypercube, we left them in to test VFKM's robustness to outliers. Any μ_k that was less than $(\sqrt{D}/K)\sigma$ away from a previously generated mean was rejected and regenerated, since problems with very close means are unlikely to be solvable by either K-means or VFKM. Examples x were generated by choosing one of the μ_k's with uniform probability, and setting the value of each dimension of the example x_d by randomly sampling from a Gaussian distribution with mean μ_{kd} and standard deviation σ.

For each choice of settings (D, K, σ) we generated a database of 10 million examples and ran VFKM and K-means on it. The algorithms were started from the same set of initial centroid locations, which were se-

Table 2. Experimental results. D is the dimensionality of the dataset. #B is the number of runs in which a bound was found (0, 1 or 2). Times are in seconds. All values are averages of two runs (except that, when a bound was only found once, that value is reported).

D	Algorithm	#B	Time	Bound	Loss
2	VFKM	0	2441	-	.209811
	K-means	0	940	-	.213600
4	VFKM	0	3639	-	.042148
	K-means	0	2486	-	.022181
6	VFKM	2	868	.000294	.000002
	K-means	2	1711	.000110	.000005
8	VFKM	2	1435	.000441	.000003
	K-means	1	2194	.000114	.000017
10	VFKM	2	1196	.000475	.000001
	K-means	2	2686	.000134	.000000
12	VFKM	2	1691	.000900	.000002
	K-means	1	3709	.000149	.000000
14	VFKM	1	5812	.001197	.652042
	K-means	1	4913	.000192	.720955
16	VFKM	2	1888	.000571	.000001
	K-means	1	4189	.000194	.000000
18	VFKM	1	6829	.000479	.732382
	K-means	1	3912	.000223	.732382
20	VFKM	2	2377	.000711	.000002
	K-means	2	5167	.000247	.000001

lected by scanning through the the database and using each example that was more than $\sqrt{D}/(2K)$ away from all previous selected examples, until K examples were found. The convergence threshold γ was set to $0.0001DK$; this appropriately scales it linearly with K and D. VFKM's other parameters were $\delta^* = 0.05$ and $\epsilon^* = \gamma/3$. We ran the experiments on two 800Mhz Pentium III computers under Linux. Table 2 contains results on 20 different databases, generated by holding K constant at 5, σ constant at 0.1, varying D from 2 to 20 in increments of 2, and generating two datasets for each resulting parameter combination. The bound is computed according to Equation 8. To compute the loss (Equation 1), we match the centroids obtained with the closest true ones using a greedy procedure. Notice that the losses are relative to the true centroids, while the bounds are relative to infinite-data K-means, and thus the two are not directly comparable.

VFKM took between 2 and 5 runs for all settings. When VFKM finds a bound it is about twice as fast as K-means. (It is also faster, and has lower loss, than K-means run on half the data.) However, on several runs it was not possible to find bounds for either algorithm. On these runs, VFKM took about twice as long as K-means. It is interesting to notice that when VFKM did not have a bound both algorithms finished with much higher loss than when a bound was obtained. Thus the fact that VFKM is unable to find a bound may itself be valuable information to the user, suggesting that the resulting clustering is suspect and that more data, a different initial configuration, or a different K may lead to better results. In general, we observe that (not surprisingly) VFKM has difficulty in finding a bound when two or more of the cluster centroids pass near each other during a run, because this leads to high values of $\tilde{n}^+_{ki}$ and X^2_{kdi} (see Equation 11). Interestingly, the hardest problems (high losses, no bounds) seem to be the lowest-dimensional ones. (This observation is necessarily tentative, given the high variance of runs and the fact that we only have two data points for each value of D.) We attribute this to the fact that in higher dimensions the data is sparser and the average distance between centroids is greater, making close brushes less likely. We observed that VFKM appropriately varied the number of examples in each iteration, increasing it in the early iterations at the expense of later ones when the early errors dominated (after propagation), and vice-versa.

We also varied K and σ; space limitations preclude a full reporting of results. We found that lower values of K and σ generally made it easier for VFKM to find a bound, except that for very small σ's (below 0.05) the results become extremely sensitive to the selection of good initial centroid locations.

VFKM's speedup relative to K-means will generally approach infinity as the database size approaches infinity. The key question is thus: what are the database sizes at which VFKM becomes worthwhile? The tentative evidence from these experiments is that they will be in the millions. Databases of this size are now common, and their growth continues unabated. In the next section we report on VFKM's application to one such database.

6. Application to Web Data

We are currently applying VFKM to mining the stream of web page request emanating from the whole University of Washington main campus. The data is described in detail in Wolman et al. (1999). In our experiments we used an anonymized trace of all the external web accesses made from campus during a one-week period in May 1999. The trace file contains 82.8 million requests from 23,000 clients. Each request is tagged with an anonymized organization ID which associates it with one of the 170 organizations (colleges, departments, etc.) within the university. One potential use of this data is to find subsets of the organizations that would benefit from sharing a single web cache rather than using individual ones. This can be accomplished by identifying sets of web pages that are accessed more by the clients in some small group of organizations than by clients outside of this group. We approach the problem as follows. We first select the D organizations with the largest number of requests in the trace file, $O_0, O_1, \ldots, O_D$; $D = 10$ in the experiment we report. We then split the trace into a series of equal-duration time slices $T_0, T_1, \ldots, T_t$; an hour each for this experiment. Within each time slice we create a matrix X with one row for each of the first n cacheable web pages requested (400,000 in this experiment) and one column for each of the D organizations, with X_{ij} containing a count of the number of times page i was accessed by organization j. We then normalize X to remove the effect of the relative activity levels of the organizations and the relative popularity of the web pages. This is done by scaling each column and then each row to sum to 1. Each row of X is an example. Using this procedure we generated a database containing $N = 20$ million examples. An example where one dimension is 1 and the rest are 0 represents a web page that is only accessed by one organization; an example where each dimension is $\frac{1}{D}$ represents a web page that is equally important to all the organizations; and an example where most of the dimensions are near 0 and a few are large represents a web page that is shared among a subset of the organizations. By finding clusters of these subset-shared examples we identify opportunities for shared web caches.

We ran VFKM on this data using $\delta^* = 0.05$, $\gamma = 0.01$, $\epsilon^* = \gamma/3$, and $K \in \{5, 7, 10\}$. The runs for $K = 5$ and $K = 7$ achieved loss bounds relative to infinite-data K-means of 0.00197 and 0.00206, respectively; the $K = 10$ run did not achieve a bound. The running times were: 8715 s for $K = 5$, 7057 s for $K = 7$, and 14055 s for $K = 10$. From the centroids found in the $K = 5$ and $K = 7$ runs we were able to identify three shared cache locations ($\{O_3, O_4, O_8\}$, $\{O_1, O_6\}$ and $\{O_3, O_5, O_8\}$), and what pages to cache at them. Sharing a cache increases its benefit because there will be fewer, but relatively more active, objects competing for the limited cache space. These results are satisfying because we mined a tiny fraction of the essentially limitless stream of web requests, and yet we are highly confident that more data will not improve the results.

7. Related Work

In Domingos and Hulten (2000), we successfully applied a more primitive version of the method described here to decision tree induction. This problem and K-means clustering differ in the type of learning (supervised vs. unsupervised), the type of learning step (discrete decision vs. continuous parameter estimation), and the type of loss function used. The fact that we were able to successfully apply the method in both cases is encouraging evidence of generality.

This direction of research was originally inspired by the work of Maron and Moore (1994) on Hoeffding races for model selection. Our work can be viewed as bringing the use of Hoeffding bounds down to the level of individual learning steps. A similar idea for the case of decision tree induction was proposed by Gratch (1996). Other references on subsampling methods for supervised learning are given in Domingos and Hulten (2000). To our knowledge, our work is the first to provide non-trivial guarantees of closeness to the infinite-data case. It would be interesting to combine VFKM with Hoeffding races to efficiently choose the number of clusters with the same type of guarantees.

John and Langley's (1996) notion of "probably close enough" learning can be viewed as a predecessor of our proposal. They consider a finite-size database (as opposed to an infinite one) as the reference for loss, and their approach is based on fitting a learning curve, offering no guarantees that the criterion will be satisfied. More generally, progressive sampling approaches attempt to iteratively determine the best number of examples to use, often via extrapolation of learning curves. However, they are hindered by the fact that real learning curves do not fit power laws and other simple forms well enough for reliable extrapolation (Provost et al., 1999). Our method provides guarantees that the necessary number of examples to satisfy a loss criterion has been reached. The price of this guarantee is that more examples than strictly necessary may be used. However, compared to progressive sampling our method has the additional advantage of allowing optimization of the number of examples used in each learning step, as opposed to determining only the number used by the whole algorithm.

Scalable clustering algorithms have been the subject of much recent research. Representative examples are CLARANS (Ng & Han, 1994), BIRCH (Zhang et al., 1997) and DENCLUE (Hinneburg & Keim, 1998). An advantage of VFKM relative to these systems is that its input-output behavior is equivalent to that of a simple, well-understood and widely used clustering method (K-means), making its results easier to interpret and communicate. Bradley et al. (1998) recently proposed an *ad hoc* method for speeding up K-means. We plan to compare it with VFKM, and possibly combine the two.

Our method bears an interesting relationship to recent work in computational learning theory that uses algorithm-specific and run-time information to obtain better loss bounds (e.g., Freund (1998), Shawe-Taylor et al. (1996)). The key difference is that we attempt to bound a learner's loss relative to the same learner running on infinite data, instead of relative to the best possible model (or the best possible from a given class). We also make more extensive use of run-time information. These two changes potentially make realistic bounds for widely-used learners (e.g., ID3, K-means) possible for the first time. However, being worst-case bounds, they are still necessarily pessimistic for most cases. Greater speedups might be obtainable by using expected loss instead of a loss bound, as in the process-oriented evaluation framework of Domingos (1999).

8. Conclusion

In this paper we proposed a method for minimizing a learner's running time while guaranteeing that the model it produces is not significantly different from the one it would given infinite data. The method is based on bounding the learner's loss as a function of the number of examples used at each step. We applied it to K-means clustering, and observed the speedups obtained on large databases.

Directions for future work include: detecting early on cases where a bound will probably not be found; further refining VFKM and applying it to other databases; improving the bound on K-means' loss (e.g., by making more extensive use of run-time information); extending our method to incorporate active sampling; and applying it to other learners, starting with the EM algorithm for mixtures of Gaussians.

Acknowledgments

This research was partly funded by NSF CAREER and IBM Faculty awards to the first author.

References

Bradley, P. S., Fayyad, U., & Reina, C. (1998). Scaling clustering algorithms to large databases. *Proc. Fourth Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 9–15). New York: AAAI Press.

Domingos, P. (1999). Process-oriented estimation of generalization error. *Proc. Sixteenth Intl. Joint Conf. on Artificial Intelligence* (pp. 714–719). Stockholm, Sweden: Morgan Kaufmann.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. *Proc. Sixth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 71–80). Boston, MA: ACM Press.

Freund, Y. (1998). Self-bounding learning algorithms. *Proc. Eleventh Annual Conf. on Computational Learning Theory.* Madison, WI: Morgan Kaufmann.

Gratch, J. (1996). Sequential inductive learning. *Proc. Thirteenth National Conf. on Artificial Intelligence* (pp. 779–786). Portland, OR: AAAI Press.

Hinneburg, A., & Keim, D. A. (1998). An efficient approach to clustering in large multimedia databases with noise. *Proc. Fourth Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 58–65). New York: AAAI Press.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*, 13–30.

John, G., & Langley, P. (1996). Static versus dynamic sampling for data mining. *Proc. Second Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 367–370). Portland, OR: AAAI Press.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297). Berkeley, CA: University of California Press.

Maron, O., & Moore, A. (1994). Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in neural information processing systems 6.* San Mateo, CA: Morgan Kaufmann.

Ng, R., & Han, J. (1994). Efficient and effective clustering method for spatial data mining. *Proc. Twentieth Intl. Conf. on Very Large Databases* (pp. 144–155). Santiago de Chile, Chile: Morgan Kaufmann.

Provost, F., Jensen, D., & Oates, T. (1999). Efficient progressive sampling. *Proc. Fifth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 23–32). San Diego, CA: ACM Press.

Shawe-Taylor, J., Bartlett, P. L., Williamson, R. C., & Anthony, M. (1996). *Structural risk minimization over data-dependent hierarchies* (Tech. Rept.). Dept. Comp. Sci., Univ. London, Egham, UK.

Wolman, A., Voelker, G., Sharma, N., Cardwell, N., Brown, M., Landray, T., Pinnel, D., Karlin, A., & Levy, H. (1999). Organization-based analysis of Web-object sharing and caching. *Proc. Second USENIX Conf. on Internet Technologies and Systems* (pp. 25–36). Boulder, CO.

Zhang, T., Ramakrishnan, R., & Livny, M. (1997). BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, *1*, 141–182.