Journal of VLSI Signal Processing 37, 199–209, 2004

A Maximum Entropy Approach for Collaborative Filtering

JOHN BROWNING AND DAVID J. MILLER
Department of Electrical Engineering, The Pennsylvania State University, Rm 227-C EEW, University Park, PA 16802-2701, USA

Abstract. Collaborative filtering (CF) involves predicting the preferences of a user for a set of items given partial knowledge of the user's preferences for other items, while leveraging a database of profiles for other users. CF has applications e.g. in predicting Web sites a person will visit and in recommending products. Fundamentally, CF is a pattern recognition task, but a formidable one, often involving a huge feature space, a large data set, and many missing features. Even more daunting is the fact that a CF inference engine must be capable of predicting *any* (user-selected) items, given *any* available set of partial knowledge on the user's other preferences. In other words, the model must be designed to solve any of a huge (combinatoric) set of possible inference tasks. CF techniques include memory-based, classification-based, and statistical modelling approaches. Among these, modelling approaches scale best with large data sets and are the most adept at handling missing features. The disadvantage of these methods lies in the statistical assumptions (e.g. feature independence), which may be unjustified. To address this shortcoming we propose a new model-based CF method, based on the maximum entropy principle. For the *MS Web* application, the new method is demonstrated to outperform a number of CF approaches, including naive Bayes and latent variable (cluster) models, support vector machines (SVMs), and the (Pearson) correlation method.

Keywords: maximum entropy, collaborative filtering, iterative scaling, latent clustering, naive Bayes

1. Introduction

In the last decade there has been rapid growth both in the number and size of electronic data repositories and in the accessibility of these databases via the Internet or private networks. Such databases include records of consumers and the products they buy, Web sites visited by users, medical records, text article databases, and bioinformatics data. The field of data mining has arisen specifically in response to this upsurge in accessible data resources. One fundamental data mining task is *information retrieval*, wherein one searches a (large) database for some small subset of records relevant to a particular subject. A related task is *collaborative filtering*, where the database records are leveraged as examples to support automated decisionmaking/predictions for new examples. Collaborative filtering (CF) differs from standard classification in two important respects. First, both the database records and the new examples will typically have many missing attribute values. Second, the attributes to be inferred or predicted for the new examples are *task-dependent* and are in general *unknown a priori*. Both of these aspects make the CF objective a particularly challenging one. CF has applications to Web browsing, marketing, product recommendation, candidate selection (for employment or admissions), information retrieval (e.g. text article retrieval) from databases, and to rule aggregation in expert and knowledge-based systems.

In this work we propose a new statistical technique for CF that has advantages over existing methods in (1) its scalability for databases of increasing size; (2) its ability to handle (an arbitrarily large number of) missing features in both the database of records and in the new data examples; and (3) the accuracy of the inferences that it produces. The new method is based on *the principle of maximum entropy* [8]. ME techniques are usually associated with off-line learning e.g. [2, 11,

12] mainly due to the learning complexity, whereas CF may require *on-line* learning and inference. While this suggests the ME approach may be impractical for CF, we explain in the sequel how practical on-line ME learning and inference can often be achieved. Our approach represents one of the first practical applications of ME modelling in an on-line learning and inference setting. In the next section, we review the CF problem in more detail. In Section 3 we develop our ME method for CF. Our approach is based on the *iterative scaling* methodology for learning ME models [2, 11, 12]. In Section 4 experimental results are given in comparison with four other well-known CF techniques on the UC Irvine *Microsoft Web* and the EachMovie databases. Finally we summarize the contribution of this work.

2. Collaborative Filtering

Consider a random feature vector $\underline{F} = (F_1, F_2, \ldots, F_N)$, with $F_i \in \mathcal{A}_i$, and $\mathcal{A}_i$ the finite set $\{1, 2, 3, \ldots, |\mathcal{A}_i|\}$. In the CF context, a realization $\underline{f} \equiv (f_1, f_2, \ldots, f_N)$ represents one record (example) in the database. For concreteness, if we consider an application to predicting which Web sites a user will visit, then N represents the number of Web sites, F_i a particular site, $\mathcal{A}_i \in \{0, 1, 2\}\, \forall i$, with the discrete values representing 'missing information', 'no visit', and 'visit', respectively. Likewise, in a product recommendation context, F_i represents a particular product, again taking on a discrete set of values, e.g. {0, 1, 2} representing 'missing information', 'dislike', and 'like', respectively, with N the number of products. In this case, $\underline{f}$ represents one example consumer. In CF applications, one is assumed to be given a database of examples $\mathcal{F}_d \equiv \{\underline{f_1^{(d)}}, \underline{f_2^{(d)}}, \ldots, \underline{f_T^{(d)}}\}$, where $\underline{f_t^{(d)}} \equiv (f_{t1}^{(d)}, f_{t2}^{(d)}, \ldots, f_{tN}^{(d)})$, $f_{ti}^{(d)} \in \mathcal{A}_i$. The (demanding) CF inference objective is then stated as follows:

Given:

(1) A target attribute i^* to be predicted, with $i^* \in \{1, 2, \ldots, N\}$.
(2) A set of $L(<N)$ *known* attributes $i_1, i_2, \ldots, i_L$.
(3) A set of K new (reduced dimension) examples $\mathcal{F}_n \equiv \{\underline{f_1^{(n)}}, \underline{f_2^{(n)}}, \ldots, \underline{f_K^{(n)}}\}$, where $\underline{f_k^{(n)}} \equiv (f_{ki_1}^{(n)}, f_{ki_2}^{(n)}, \ldots, f_{ki_L}^{(n)})$, $f_{ki_m}^{(n)} \in \mathcal{A}_{i_m}$.

Goal: Predict the values $f_{ki^*}^{(n)}$, $k = 1, 2, \ldots, K$.

Example. Given 4 users $(\underline{f_1^{(d)}}, \underline{f_2^{(d)}}, \underline{f_3^{(d)}}, \underline{f_4^{(d)}})$ and their ratings on 8 movies. The possible ratings are 0–5 and an empty box indicates that the user has not seen that movie. Predict how new user $\underline{f_1^{(n)}}$ will rate Movie 6.

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6	Movie 7	Movie 8
$\underline{f_4^{(d)}}$	2	3	2		2	3		5
$\underline{f_3^{(d)}}$	1	4		3	0	2	1	
$\underline{f_2^{(d)}}$	3		1	2	4			4
$\underline{f_1^{(d)}}$		3	2		1	5	4	
$\underline{f_1^{(n)}}$			1		2	?	4	3

Several important observations can be made about this problem in general:

(1) The target feature is essentially a *class* feature. Thus, specifying i^* and the known attributes amounts to specifying a classification problem that needs to be solved. Also, although we have indicated the goal of inferring a *single* target attribute value for each new example, this is easily extended to the prediction of a *collection* of attribute values.
(2) The number of such classification problems is *huge* (combinatoric), as one can choose both the target feature(s) and the subset of known features. Thus, off-line learning and storage of one classifier for each possible inference task is utterly infeasible.[1] Practical methods for CF must thus be versatile approaches, capable of producing inferences for *any* posed CF task *without* undue delay, i.e. by either a direct (in some sense) 'on-line' inference procedure or by 'on-line' model learning based on

$\mathcal{F}_d$ *followed* by inference. Our ME approach will fall in the latter category.

(3) Whenever $L < N - 1$, a classifier designed to infer F_{i^*} given the remaining $N - 1$ feature values will require a (heuristic) missing feature imputation strategy. SVMs [9] is one such approach. Note further that in practice we will have $L \ll N - 1$. In particular, for the *MS Web* domain, $N = 285$ and we will (reasonably) assume $L = 5$ or 10. Thus, SVMs will need to impute *many* missing features in practice.

(4) Often the database examples $\underline{f_t^{(d)}}$ will *also* have many missing attribute values. This will again create difficulties for classification-based CF methods (e.g. SVMs), which must (heuristically) impute values for the missing features during the learning phase.

There are three basic approaches taken to attacking the CF problem: (1) memory-based methods; (2) classification-based approaches; and (3) statistical modelling approaches. We next briefly review these.

2.1. *Memory-Based Methods*

These methods essentially perform weighted voting using the examples in the database, with the weight for each database example based on the degree of similarity between the example and the new instance $\underline{f^{(n)}}$. Often, before voting is done, the similarity values are compared to a threshold to reduce the size of the voting ensemble. This is somewhat akin to the K-nearest neighbor method in classification. A variety of similarity measures have been used, including Pearson's correlation coefficient and its variants and cosine-based similarity [3].

Some practical concerns for memory-based methods include: (1) the reliability of the similarity measure, which is based on only the known feature subset (of size $L \ll N$); (2) the choice of the threshold, which controls the number of voters (and hence which can afiect performance [1]). However, the most serious issue is how complexity scales with increasing database size—since the similarity is evaluated for each database example, inference complexity grows *linearly* with T. Thus, for millions of database records, complexity will become a serious obstacle. Clustering the records is one way to mitigate this problem.

2.2. *Classification-Based Methods*

One example of this approach is the application of SVMs to CF [5] (although other classifiers can also be used). Here, one builds N SVMs, with each one dedicated to the prediction of a *single* feature F_i. Since the SVM learning complexity grows with the data set size T, the learning must be done off-line. Moreover, since it is infeasible to learn and store one classifier for each possible inference task (i.e. each combination of target feature i^* and known feature subset), each SVM must be built for the *full* space of dimension N, i.e. assuming an $(N - 1)$-dimensional input vector with all values known. One difficulty this creates is that the database will have many missing feature values. Thus, a heuristic imputation strategy must be used to 'fill in' these values prior to SVM training. More serious is the fact that the new (test) examples will *also* have many missing features. Thus, heuristic imputations must also be made for many attributes in the new (test) examples before SVM-based inference can proceed. These imputations can seriously affect inference accuracy. Another issue is (even) the *off-line* learning complexity, since this grows at least with the square of N.[2] Dimensionality reduction methods such as the singular value decomposition (SVD) can be used to mitigate this problem [6].

2.3. *Statistical Modelling Approaches*

These methods are based on a joint probability model over the feature space consisting of the known feature subset plus the feature to be predicted. First, given a specified CF inference task, the joint probability model $P[F_{i^*}, F_{i_1}, F_{i_2}, \ldots, F_{i_L}]$ is formed (in some cases, learned). The predictions are then based on the *a posteriori* probabilities $P[F_{i^*} = f_{i^*} \mid f_{i_1}, f_{i_2}, \ldots, f_{i_L}]$, which are computed (via Bayes rule) consistent with the joint probability model. This approach is grounded in the well-known theoretical result that the optimal (Bayes) classifier in the presence of missing features uses the *a posteriori* probabilities that condition *only on the set of known features*—i.e., it is suboptimal to perform missing feature inference if one can instead evaluate the proper *a posteriori* probabilities, which condition only on the known features [15].

We believe that this class of CF methods is the most promising one. Statistical modelling approaches do not sufier from the missing feature difficulties of the classification-based methods,[3] and their inference

complexity may have very limited dependence on the database size T. Moreover, these methods produce probabilities as inferences, which are attractive in some applications. We next briefly describe two such methods.

The Naive Bayes Model. This model assumes all known features are conditionally independent given F_{i^*}. The joint probability is thus

$$P\left[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = f_{i^*}\right] = P[F_{i^*} = f_{i^*}] \prod_{j=1}^{L} P\left[F_{i_j} = f_{i_j} \mid F_{i^*} = f_{i^*}\right]$$

and the associated *a posteriori* probabilities are (via Bayes rule)

$$P\left[F_{i^*} = f_{i^*} \mid f_{i_1}, f_{i_2}, \ldots, f_{i_L}\right] = \frac{P\left[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = f_{i^*}\right]}{\sum_{k=1}^{|\mathcal{A}_{i^*}|} P\left[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = k\right]}. \quad (1)$$

The conditional probabilities involved in (1) are estimated via frequency counts over the database, i.e.

$$P\left[F_{i_j} = f_{i_j} \mid F_{i^*} = f_{i^*}\right] = \frac{N\left(F_{i^*} = f_{i^*}, F_{i_j} = f_{i_j}\right)}{N(F_{i^*} = f_{i^*})},$$

where $N(\cdot)$ is the number of times an event occurred in the database.

The Latent Variable (Cluster) Model. This is a more powerful (mixture) model than naive Bayes, one that assumes features are conditionally independent given an unobserved (latent) 'cluster' variable, C. The joint probability is given by

$$P\left[F_{i_1} = f_{i_1}, F_{i_2} = f_{i_2}, \ldots, F_{i_L} = f_{i_L}, F_{i^*} = f_{i^*}\right] = \sum_{m=1}^{N_c} P[C = m] P[F_{i^*} = f_{i^*} \mid C = m] \times \prod_{j=1}^{L} P\left[F_{i_j} = f_{i_j} \mid C = m\right].$$

The associated *a posteriori* probabilities are again easily formed via Bayes rule, as given in (1). This model can be learned for each new CF task via maximum likelihood estimation (MLE).[4] Since the model is not very large ($L \ll N$), MLE can often be practically accomplished 'on-line'. We have implemented the learning via the Expectation/Maximization (EM) algorithm [4]. Note also that one must choose the number of (latent) clusters, N_c. We have found that an improper choice can lead either to overfitting or to underfitting the data. Cluster models for CF were proposed in several previous works [3, 7].

3. A Maximum Entropy CF Approach

One disadvantage of the previous models, and of naive Bayes in particular, is the assumption of feature independence. It appears quite unlikely, for any *arbitrarily chosen* CF task, that the set of known features will be independent given the feature to be predicted. An alternative to explicitly assuming a parametric form for the joint pmf is to view model learning as a problem of *constraint encoding*, with the goal to encode as many *accurately measured* constraints as possible, while remaining maximally noncommittal with respect to any other (unjustified) statistical assumptions. The principle of maximum entropy (ME) [8] embodies this idea, suggesting to find the solution of maximum entropy consistent with the specified constraints. The ME solution is unique and has been given theoretical justification [8]. ME has found frequent use, especially in natural language modelling and speech-related applications [2, 11, 12]. The use of ME for more general classification and inference tasks has also recently been investigated [10, 13]. Since learning an ME model in these settings generally requires optimization over a large set of parameters via gradient-based or other iterative techniques, model learning is almost always an *off-line* task, requiring hours or even days of computation. However, in the CF setting, a key observation is that, although the feature space size (N) and database size (T) may both be huge, the number of *known* features L is typically quite small, and it is this parameter which may determine the complexity of ME learning. Thus, for CF, we suggest that practical *on-line* ME model learning and inference is often actually viable. We next develop such a learning and inference approach.

3.1. ME Model

There are several approaches to ME learning, including "iterative scaling" and its extensions [2, 12], as well as the method developed in [10]. While the approach

in [10] does have some potential advantages,[5] iterative scaling has lower computational complexity and is thus most appropriate in the CF setting. In the ME framework, one learns parameters of an ME *a posteriori* model $P[F_{i^*} = f_{i^*} \mid f_{i_1}, f_{i_2}, \ldots, f_{i_L}]$ to maximize entropy while ensuring that expected value (e.g. lower order probability) constraints measured with respect to the model distribution agree with those measured based on a uniform (empirical) distribution over the data set, i.e. based on *frequency counts*. Since the accuracy of frequency count estimates of probabilities decreases with the *order* of the probabilities, one typically must limit constraints to low order. Thus, in this work we suggest to encode all the *pairwise* pmf constraints involving the feature to be predicted, i.e. $\{P[F_{i^*}, F_{i_1}], P[F_{i^*}, F_{i_2}], \ldots, P[F_{i^*}, F_{i_L}]\}$. The parametric *form* for the ME *a posteriori* probabilities consistent with these constraints is *exponential*:

$$P_{\mathrm{ME}}\big[F_{i^*} = f_{i^*} \,\big|\, f_{i_1}, f_{i_2}, \ldots, f_{i_L}\big] = \frac{e^{\sum_{j=1}^{L} \gamma(F_{i^*}=f_{i^*}, F_{i_j}=f_{i_j})}}{\sum_{m=1}^{|\mathcal{A}_{i^*}|} e^{\sum_{j=1}^{L} \gamma(F_{i^*}=m, F_{i_j}=f_{i_j})}}, \tag{2}$$

where $\gamma(F_{i^*} = f_{i^*}, F_{i_j} = f_{i_j})$ is the Lagrange multiplier associated with the constraint on the probability $P[F_{i^*} = f_{i^*}, F_{i_j} = f_{i_j}]$. In generalized iterative scaling [2], one assumes a uniform pmf over (training) database examples. Thus, for each reduced-dimension example $\underline{\tilde{f}_t^{(d)}} \equiv (f_{ti_1}^{(d)}, f_{ti_2}^{(d)}, \ldots, f_{ti_L}^{(d)})$, the *model* joint pmf takes the form $P[\underline{\tilde{f}_t^{(d)}}, F_{i^*} = f_{i^*}] = \frac{1}{T} P_{\mathrm{ME}}[F_{i^*} = f_{i^*} \mid \underline{\tilde{f}_t^{(d)}}]$. Accordingly, the model's conditional entropy is

$$H = \sum_{\underline{f_t} \in \mathcal{F}_d} \frac{1}{T} \sum_{k=1}^{|\mathcal{A}_{i^*}|} P_{\mathrm{ME}}\big[F_{i^*} = k \,\big|\, \underline{\tilde{f}_t^{(d)}}\big] \times \log P_{\mathrm{ME}}\big[F_{i^*} = k \,\big|\, \underline{\tilde{f}_t^{(d)}}\big].$$

Likewise, a pairwise constraint on any probability $P[F_{i^*} = f_{i^*}, F_{i_j} = f_{i_j}]$ can be expressed as:

$$\sum_{\underline{f_t}^{(d)} \in \mathcal{F}_d : f_{ti_j}^{(d)} = f_{i_j}} \frac{1}{T} P_{\mathrm{ME}}\big[F_{i^*} = f_{i^*} \,\big|\, \underline{\tilde{f}_t^{(d)}}\big] = \frac{N\big(F_{i^*} = f_{i^*}, F_{i_j} = f_{i_j}\big)}{T}. \tag{3}$$

In [2], it was shown that the Lagrangian objective function associated with the problem of maximizing H subject to all the specified constraints is equivalent to a special *likelihood* function. Given our constraints (3), the ME Lagrangian function takes the form

$$\mathcal{L} = H + \sum_{j=1}^{N_c} \sum_{m=1}^{|\mathcal{A}_{i^*}|} \sum_{f_{i_j}=1}^{|\mathcal{A}_{i_j}|} \gamma_{(F_{i^*}=m, F_{i_j}=f_{i_j})} \times \left(\frac{N(F_{i^*} = f_{i^*}, F_{i_j} = f_{i_j})}{T} - \sum_{\underline{f_t}^{(d)} \in \mathcal{F}_d : f_{ti_j}^{(d)} = f_{i_j}} \frac{1}{T} P_{\mathrm{ME}}\big[F_{i^*} = f_{i^*} \,\big|\, \underline{\tilde{f}_t^{(d)}}\big] \right). \tag{4}$$

After several simplifying steps, it is discovered that this can be rewritten in the form of the *conditional log-likelihood*

$$\mathcal{L} = \log \prod_{\underline{f_t} \in \mathcal{F}_d} P_{\mathrm{ME}}\big[F_{i^*} = f_{i^*} \mid \underline{\tilde{f}_t^{(d)}}\big]^{\frac{1}{T}}. \tag{5}$$

Taking the partial derivative with respect to any of the Lagrange multipliers can be shown to give the following (satisfying) result:

$$\frac{\partial \mathcal{L}}{\partial \gamma_{(F_{i^*}=f_{i^*}, F_{i_j}=f_{i_j})}} = \frac{N\big(F_{i_j} = f_{i_j}, F_{i^*} = f_{i^*}\big)}{T} - \sum_{\underline{f_t}^{(d)} \in \mathcal{F}_d : f_{ti_j}^{(d)} = f_{i_j}} \frac{1}{T} P_{\mathrm{ME}}\big[F_{i^*} = f_{i^*} \,\big|\, \underline{\tilde{f}_t^{(d)}}\big], \tag{6}$$

i.e., setting the partial derivatives to zero, a necessary optimality condition is simply the constraint condition (3), which says that constraint probabilities measured with respect to the model's joint pmf must equal those measured with respect to the empirical (uniform) distribution over the database. Thus, ME learning amounts to simply choosing the parameters of the exponential model to satisfy the constraints. Moreover, since the constraints are linear, the solution is unique [2]. Thus, ME learning seeks the (unique) exponential model satisfying the given constraints (3). There are several techniques that can be used to perform this learning of the $\gamma(\cdot, \cdot)$ parameters, including gradient descent and generalized iterative scaling [2]. Both of these methods will require computing the sums

$$\sum_{\underline{f_t}^{(d)} \in \mathcal{F}_d : f_{ti_j}^{(d)} = f_{i_j}} \frac{1}{T} P_{\mathrm{ME}}\big[F_{i^*} = f_{i^*} \,\big|\, \underline{\tilde{f}_t^{(d)}}\big]. \tag{7}$$

The efficiency of this computation can in some cases be greatly increased with the following observation: since

$L \ll N$, it may also be true that $|\mathcal{A}_{i_1} \times \mathcal{A}_{i_2} \times \cdots \times \mathcal{A}_{i_L}| \ll T$. When this is true, the number of terms in the sum can be drastically reduced. In particular the sum can be rewritten as

$$\sum_{\underline{f} \in \mathcal{A}_{i_1} \times \mathcal{A}_{i_2} \times \cdots \times \mathcal{A}_{i_L} \times \mathcal{A}_{i^*} : F_{i_j} = f_{i_j}} \frac{1}{T} N(\underline{f}) P_{\mathrm{ME}}[F_{i^*} = f_{i^*} \mid \underline{\tilde{f}}]. \tag{8}$$

For the MSWEB database, this simplification leads to substantial savings in learning time. For Each-Movie, this approach does not help since $|\mathcal{A}_{i_1} \times \mathcal{A}_{i_2} \times \cdots \times \mathcal{A}_{i_L}| > T$ for the choices of L that we will consider. Below we describe the iterative scaling approach we have taken for ME learning.

3.2. Iterative Scaling

The generalized Iterative Scaling procedure was first developed in [12] as a method of finding the Lagrange Multipliers to maximize the log likelihood (5) when $P_{\mathrm{ME}}[F_{i^*} = f_{i^*} \mid \underline{\tilde{f}}_t^{(d)}]$ has an exponential form, as in (2). It has been shown e.g. in [12, 16] that this log-likelihood function is convex in the Lagrange multipliers and thus has a unique maximum.

3.2.1. Algorithm. In the following, when we wish to refer to the vector composed from the full set of Lagrange multipliers or to a perturbation of this vector, we will use the notation $\underline{\gamma}$ or $\underline{\Delta\gamma}$, respectively. The generalized iterative scaling algorithm consists of the following basic steps:

1. Initialize the Lagrange Multipliers, $\underline{\gamma}$. This can be done randomly; $k \leftarrow 0$.
2. Calculate an increment for the Lagrange Multipliers $\underline{\Delta\gamma}^{(k)}$ that is guaranteed to increase the log likelihood function.
3. Add the calculated increments to the Lagrange Multipliers, i.e.

$$\underline{\gamma}^{(k+1)} \leftarrow \underline{\gamma}^{(k)} + \underline{\Delta\gamma}^{(k)}; \quad k \leftarrow k + 1 \tag{9}$$

Steps 2 and 3 are repeated until a convergence criterion is met.

Step 2 is implemented through the use of an auxiliary function $A(\underline{\Delta\gamma}, \underline{\gamma})$. This function is chosen so that:

(1) For any $\underline{\Delta\gamma}$

$$\mathcal{L}(\underline{\gamma} + \underline{\Delta\gamma}) - \mathcal{L}(\underline{\gamma}) \geq A(\underline{\Delta\gamma}, \underline{\gamma}); \tag{10}$$

(2) $A(\underline{\Delta\gamma}, \underline{\gamma})$ is a convex function in $\underline{\Delta\gamma}$, i.e. it has a unique maximum, also $\max_{\underline{\Delta\gamma}} A(\underline{\Delta\gamma}, \underline{\gamma}) \geq 0$.

Let $\underline{\Delta\gamma}^{(k)}$ be the value that maximizes $A(\underline{\Delta\gamma}, \underline{\gamma})$. This increment to the Lagrange multipliers will guarantee an increase in the log likelihood function. As the algorithm iterates, $\mathcal{L}(\underline{\gamma}^{(k)})$ will thus increase monotonically. It is proved in [12] that, in fact, the likelihood converges to $\max \mathcal{L}(\underline{\gamma}^{(k)})$ and $\underline{\gamma} \rightarrow \arg\max_{\underline{\gamma}} \mathcal{L}(\underline{\gamma})$.

For our ME problem the auxiliary function is given by

$$\begin{aligned} A(\underline{\Delta\gamma}, \underline{\gamma}) = & \sum_{j=1}^{N_c} \sum_{m=1}^{|A_{i^*}|} \sum_{f_{i_j}=1}^{|A_{i_j}|} \Delta\gamma_{(F_{i^*}=m, F_{i_j}=f_{i_j})} \\ & \times \frac{N\left(F_{i^*} = f_{i^*}, F_{i_j} = f_{i_j}\right)}{T} + 1 \\ & - \sum_{j=1}^{N_c} \sum_{m=1}^{|A_{i^*}|} \sum_{f_{i_j}=1}^{|A_{i_j}|} \frac{1}{LT} \sum_{\underline{f_t}^{(d)} \in \mathcal{F}_d : f_{ti_j}^{(d)} = f_{i_j}} \\ & \times P_{\mathrm{ME}}[F_{i^*} = f_{i^*} \mid \underline{\tilde{f}}_t^{(d)}] e^{L\Delta\gamma_{(F_{i^*}=m, F_{i_j}=f_{i_j})}}, \end{aligned} \tag{11}$$

where L is the number of known features. It can be shown that $A(\underline{\Delta\gamma}, \underline{\gamma})$ satisfies (10), see [12]. The convexity of this function can be shown by computing the Hessian matrix and confirming that it is indeed negative definite.

To maximize $A(\underline{\Delta\gamma}, \underline{\gamma})$, we take the partial derivative, set it to zero, and solve for $\Delta\gamma_{(F_{i^*}=m, F_{i_j}=f_{i_j})}$ $\forall_{i_j}, J = 1, \ldots, L \forall m \forall_{f_j}$. We then find that, for each Lagrange multiplier the update is:

$$\begin{aligned} & \Delta\gamma_{(F_{i^*}=m, F_{i_j}=f_{i_j})}^{(k+1)} \\ & = \frac{1}{L} \log \frac{\frac{N(F_{i^*}=f_{i^*}, F_{i_j}=f_{i_j})}{T}}{\frac{1}{T} \sum_{\underline{f_t}^{(d)} \in \mathcal{F}_d : f_{ti_j}^{(d)} = f_{i_j}} P_{\mathrm{ME}}\left[F_{i^*} = f_{i^*} \mid \underline{\tilde{f}_t}^{(d)}\right]}. \end{aligned} \tag{12}$$

This is the update used in step 2 of the algorithm. Note that when a pairwise constraint is satisfied, the expression inside the log( ) is one, and the associated component of $\underline{\Delta\gamma}$ is thus zero. The auxiliary function $A(\underline{\Delta\gamma}, \underline{\gamma})$ has several important properties [12]:

1. $A(\underline{0}, \underline{\gamma}) = 0$, which implies that the maximum of $A(\underline{\Delta\gamma}, \underline{\gamma})$ is at least 0. This condition is necessary because if $\underline{\Delta\gamma} = \underline{0}$ there is no change in the log likelihood.
2. The directional derivatives of $\mathcal{L}(\underline{\gamma} + \underline{\Delta\gamma})$ and $A(\underline{\Delta\gamma}, \underline{\gamma})$ are equal at $\underline{\Delta\gamma} = \underline{0}$, i.e.

$$\frac{d}{dt}\mathcal{L}(\underline{\gamma} + t\underline{\Delta\gamma})|_{t=0} = \frac{d}{dt}A(t\underline{\Delta\gamma}, \underline{\gamma})|_{t=0}. \quad (13)$$

Since $\frac{d}{dt}A(t\underline{\Delta\gamma}, \underline{\gamma})|_{t=0} = \nabla A(\underline{0}, \underline{\gamma}) \cdot \underline{\Delta\gamma}$ this implies that at $\underline{\Delta\gamma} = \underline{0}$ the gradients of $A(\underline{\Delta\gamma}, \underline{\gamma})$ and $\mathcal{L}(\underline{\gamma})$ are equal. Specifically if $\nabla A(\underline{0}, \underline{\gamma}) = 0$, i.e. if the maximum of $A(\underline{\Delta\gamma}, \underline{\gamma})$ is found when $\underline{\Delta\gamma} = \underline{0}$, then the maximum of $\mathcal{L}(\underline{\gamma})$ has also been found.

3.2.2. Convergence. A convergence proof is given in [12]. We provide a brief summary of the argument, as follows. We can see from the algorithm described above that every iteration step increases the log likelihood. Since the sequence steps are probability distributions which are members of a compact set, there is a convergent subsequence and at least one cluster point. It is shown in [12] that all cluster points of the algorithm will be maxima of the log likelihood. Further, since the log likelihood is convex it has only one maximum and hence only one cluster point. Therefore the algorithm converges to the unique point that satisfies $\nabla A(\underline{0}, \underline{\gamma}) = \nabla\mathcal{L}(\underline{\gamma}) = \underline{0}$. This also satisfies the constraints in (3).

3.3. Implicit Versus Explicit Databases and Missing Votes

We have tested our ME inference method on two different CF databases, EachMovie and MSWEB. The EachMovie database consists of movie ratings from 61,264 users. There are a total of 1622 rated movies. Each user rates movies he or she has seen. The ratings are '1' to '6' with a value of '0' for all movies not seen. The MSWEB database lists Microsoft web sites visited by 32,711 users. There are 285 web sites that have been visited by at least one user. The database values are '0' and '1' for 'not visited' and 'visited', respectively. The CF objective is to predict whether a user will visit a given web site or not. The first database, EachMovie, is an example of an *explicit* database while the second one is an example of an *implicit* database. The difference is that the records represent users' preferences based on either their (1) explicit voting or rating of objects or (2) their past behavior. Explicit voting involves explicit rating of items. For example, rating movies or newspaper articles. In this case there may be many missing features, i.e. a person will not usually see every single movie that Hollywood has released, and so many movies are unrated in the database. On the other hand, implicit votes are based on the interpretation of a user's behavior. If a person accesses an item, for instance buying a product or visiting a web site, that is counted as a 'yes' vote. All items not accessed are counted as 'no' votes. Implicit voting is interpreted as having no missing information, i.e. by definition any items not accessed are assigned 'no' votes.[6] For the case of explicit voting, one needs to address the problem of missing feature values in the database records when building the model. We next consider this problem, proposing several novel learning approaches amenable to the ME statistical modelling approach to account for these missing feature values.

We have the following strategies for handling missing votes:

1. One can learn a model where $P_{\text{ME}}[F_{i^*} = \text{"unrated"} \mid \underline{f}]$, is a valid choice. In this case, for EachMovie, the possible ratings would be the six values '1' to '6' and another value representing 'unrated', making a total of seven possible ratings. If the maximum *a posteriori* rule $f_{i^*} = \arg\max_l P_{\text{ME}}[F_{i^*} = l \mid \underline{f}]$ is applied, this will result in many 'unrated' predictions because the database has many unrated instances. Since our objective is to predict *ratings* for the movies, these unrated predictions would be unacceptable. An alternative, after model learning, is to apply a rule that disallows unrated predictions, i.e. to use the rule $f_{i^*} = \arg\max_l P_{\text{ME}}[F_{i^*} = l \mid \underline{f}, l \in \{\text{ratings choices}\}]$, where $P_{\text{ME}}[F_{i^*} = l \mid \underline{f}, l \in \{\text{ratings choices}\}] = \frac{P_{\text{ME}}[F_{i^*} = l \mid \underline{f}]}{(1 - P_{\text{ME}}[F_{i^*} = \text{"unrated"} \mid \underline{f}])}, l \in \{\text{ratings choices}\}$. This is a MAP rule that *excludes* 'unrated' as a valid choice.
2. A second method is to *learn* a model that restricts the values taken on by F_{i^*} to be ratings values. To achieve this, one must discard all database examples for which $F_{i^*} =$ 'unrated' when forming frequency count estimates of the pairwise statistics $\{P[F_{i^*}, F_j]\}$. This will result in the ratings-restricted learned pmf $P_{\text{ME}}[F_{i^*} = l \mid \underline{f}], l \in \{\text{ratings choices}\}$. This method, called the 'throw away data' method, achieves better results than the first method.

3. A third method we propose for on-line ME is a two stage learning method: First, learn an ME model using method 2. Next, evaluate $P_{\text{ME}}[F_{i^*} = l \mid \underline{f}^{(n)}], l \in \{\text{ratings choices}\}$ for all database examples for which $F_{i^*} =$ "unrated". Effiectively, this produces probabilistic ratings for all unrated database examples. Thus, all database examples will now have ratings, either deterministic or probabilistic ones. We can thus form new constraint probabilities via

$$P\left[F_{i^*} = k, F_{i_j} = f_{i_j}\right] = \frac{N(F_{i^*} = k, F_{i_j} = f_{i_j}) + \sum_{t: f_{ti^*}^{(d)} = \text{"unrated"}} P_{\text{ME}}[F_{i^*} = k \mid \underline{\tilde{f}^{(d)}}]}{T}, \tag{14}$$

$k \in \{\text{ratings choices}\}$. In this way, unlike method 2, all the data is tapped for estimating the constraint probabilities. Finally, we relearn the ME model $P_{\text{ME}}[F_{i^*} = l \mid \underline{f}^{(n)}], l \in \{\text{ratings choices}\}$ and use it in a MAP rule. This method, which we call 'relearn model', is the best missing feature method we have found for on-line ME classification.

4. Results

Two databases, EachMovie and MSWEB were used for evaluation. For the MSWEB database, the following constitutes one trial for a CF algorithm. Either $L = 2$, 5 or 10 of the 285 visited web sites were randomly chosen as known, with values for the remaining sites to be inferred. Of the 5000 users in the test set, only those with at least one of the selected sites visited were included in the test set for this particular trial. That means that the number of test vectors differ for each trial. A model was learned for each inference task and prediction was then done on all of the test vectors in the trial. Because the results varied from one trial to another, a total of fifteen trials were done. The reported results were obtained by micro-averaging over all of the trials. The methods we compare include Correlation, SVMs, naive Bayes, the Cluster model (BC), a brute-force frequency-count estimator (FC), and ME. For Correlation, we used the Pearson coefficient as the basis for the weights, as in [3]. The weights were formed for all the training examples and then weighted voting was performed. Weight amplification with a factor of 2 was also used. For the Cluster model, we chose $N_c = 7$ since this gave the best (test set) results. The FC method estimates *a posteriori* probabilities directly via $\frac{N(F_{i^*}=f_{i^*}, \underline{F}=\underline{f})}{N(\underline{F}=\underline{f})}$. For SVMs, the *SVM light* program of Thorsten Joachims [9] was used. We created 285 different linear SVMs. To do inference, the known values were set to 1 or 0 and all unknown values were set to 0.5. The dot products for all sites to be inferred were formed in the usual way and the values were compared to a threshold for prediction. While the threshold is typically chosen to be zero, we found that a non-zero value gave better performance. Results are reported using the recall/precision metric. Recall (r) is the percentage of visited web sites that we predicted as visited. Precision (p) is the percentage of predicted visits that really were visited. For MSWEB we report the F_1 measure which combines precision and recall. The definition of F_1 is:

$$F_1 = \frac{2pr}{p + r} \tag{15}$$

For each method the threshold was found that gave the point where precision equals recall. As aforementioned, since results can vary considerably for different groups of known features, we averaged the results over 15 (randomly selected) groups.

As seen in Table 1, the statistical modelling methods clearly outperform SVMs and Correlation. The poor performance of SVMs is apparently due to missing features, since SVM performance was observed to improve as we continued to increase L (beyond 10). The FC method does surprisingly well. We attribute its success to the fact that L is relatively small and the amount of data, T, is large. We also report on execution times on a Sunsparc10 workstation. For naive Bayes and FC, it took less than one second to compile the necessary statistics and less than 1 second to do one CF inference

Table 1. MSWEB prediction results for various methods given 2, 5 and 10 known features. The performance is based on the $F1$ measure.

Data sets	Given 2	Given 5	Given 10
ME	0.331	0.329	0.332
BC	0.316	0.319	0.319
NBC	0.317	0.303	0.326
FC	0.329	0.325	0.331
CORR	0.198	0.195	0.278
SVM	0.197	0.204	0.273

task with $K = 237$ (K is the number of test vectors in 1 trial). For Correlation, it took about 6 seconds to do 280 predictions on a *single* example (i.e., $K = 1$). For ME, with $L = 5$, it takes <1 second to create the model, after which it takes <1 second to do one CF inference task (with $K = 237$). SVMs requires anywhere from 15 minutes to 2 hours to build each model. Each prediction takes on the order of milliseconds. Given the modest learning time for ME for small L, it does appear feasible to use this approach in an on-line, i.e. real-time setting.

For the EachMovie Database, only records with 40 or more rated movies were used for testing. There are 20,322 such records. Of these, 600 were randomly selected for the test set and 15,322 records were used as the training set. For each of the users in the test set, five sites were chosen at random for the known sites and 35 of the other sites were randomly chosen as sites to be predicted. A model was learned for each inference problem and then prediction was performed. Because EachMovie is an explicit database with missing values, the MSWEB method of building a model and using it to do predictions on the whole test set cannot be used. Therefore a different model had to be built for each of the 35 prediction tasks performed for each test vector. For this database, the possible ratings are '1',...,'6'. We choose the rating with the highest probability as our prediction (an alternative is to use the expected value—these two methods give comparable results). All the results in Table 2 are for $L = 5$. We report the average absolute value of the prediction error, over 21,000 predictions. The average absolute value of the prediction error is defined as

$$\text{error}_{\text{av}} = \frac{1}{K} \sum_{k=1}^{K} \left| f_{ki^*}^{(n)} - \widehat{f_{ki^*}^{(n)}} \right| \tag{16}$$

Table 2. EachMovie prediction results for various methods given 5 known features. The performance is based on the average absolute value of the prediction error.

Method	Result
Correlation	1.18
NBC	0.975
ME second best	0.923
ME throw away data	0.861
ME relearn model	0.851

where $\widehat{f_{ki^*}^{(n)}} k_{i^*}$ is the estimated value of $f_{ki^*}^{(n)}$. A higher average absolute value of prediction error indicates greater deviation from the correct answers and poorer performance.

For the EachMovie data set, as for MSWEB, the Maximum Entropy method is seen to give appreciably better performance than the correlation method. By way of comparison, [3] gets an average absolute prediction of 1.139 for correlation as compared to our value of 1.18. A possible reason for this difference is that [3] uses a slightly different training/test split for EachMovie than we did. Also, methods (2) and (3) for handling missing features in ME models further improve the accuracy of the predictions. The correlation method requires about 4 seconds to do inference for one test vector consisting of 35 movies to be inferred. ME requires about 10 seconds to do learning and inference for one inference task.

5. Discussion

Our new (ME) method falls into the class of 'statistical modelling' approaches to CF. As such, it has an inherent advantage over classification-based approaches (e.g. SVMs) in handling missing features in the training set. Even more importantly, unlike SVMs the new method does not require (heuristic) missing feature imputations when forming its predictions. Another advantage of the new method and other modelling approaches to CF lies in their scalability, both with increasing size of the feature space, but especially with increasing size of the database (T). Classification-based approaches will typically require a linear increase in *learning* complexity with increasing database size. As a more serious obstacle, memory-based methods will experience a linear increase in the complexity of the *prediction* phase. By contrast, the complexity of model-based approaches may have very little dependence on T. The complexity of estimating constraint probabilities via frequency counts does grow with T, but this takes very little time even for T on the order of millions. More importantly, the complexity of ME learning of the *a posteriori* model is *independent of* T if the sums required in iterative scaling are computed by summing over all elements of the known feature space, as in (8), rather than by summing directly over the training set, as in (7). ME model-building complexity thus may only grow with the number of *known*

feature values and/or the number of ME constraints that are encoded. Thus, we expect that the new method will be more practically viable than memory-based methods for domains with huge databases (e.g. millions of data records) and limited known attribute information (e.g. 5, 10, or 20 known feature values). Moreover, whereas the complexity of classification-based methods typically grows (at least) quadratically with the size of the feature space (N), complexity for the new method (and for other model-based approaches) grows (for each inference task) only with the size of the *known* feature subset (L). Even for spaces with thousands of features (e.g. the *EachMovie* domain), the size of the known feature subset may only be as large as ten or twenty features. While dimensionality reduction techniques can be used with classification-based methods, e.g. [6], one may need to sacrifice important information to achieve a practical dimensionality for learning and inference. Finally, compared with alternative statistical approaches, the advantage of the ME method lies in its improved inference accuracy. The new method was observed to achieve gains in the F1-measure and in absolute deviations over naive Bayes and cluster models, as well as over SVMs and the Pearson correlation method. The gains over the other statistical models are attributable to the ME method's encoding of more constraints, as well as to its inherent avoidance of unncessary statistical assumptions (e.g. independence).

6. Future Work

Our ME solution could potentially be improved in some cases by encoding *more* constraints (those involving the feature to be predicted and *groups* of known features) and in some cases by encoding *fewer* constraints (i.e., only encoding some constraints involving pairs of *known* features[7]). One approach to encoding fewer constraints involving only known features is [10]. However, this method requires greater complexity than the iterative scaling ME method used here. Another future direction is to develop an extension for CF problems with *mixed* continuous/discrete feature spaces. A starting point for such development can be found in [13]. Finally, other applications could be investigated, for example: (1) retrieval of relevant articles from a database based on abstracts; (2) aggregation of rule bases in data mining; and (3) applications in marketing research.

Acknowledgments

This work was supported by National Science Foundation grants IIS-9624870 and IIS-0082214.

Notes

1. Classification-based methods, discussed in the sequel, do use off-line learning to build N classifiers, with each dedicated to the prediction of a single feature, given knowledge of *all* the remaining feature values. However, this approach does not build a classifier dedicated to each *inference* task, i.e. it does not account for the (many) missing features that will occur in the new (test) examples.
2. There are N SVMs to build, with learning complexity for each one growing at least linearly with N.
3. No missing feature imputations are required for test examples, since the model is based solely on the (known) features $F_{i_1}, F_{i_2}, \ldots, F_{i_L}$ and the feature to be inferred, F_{i^*}. For model-building, low-order statistics (probabilities) must be estimated via frequency counts over the database. Here, missing features are encountered. However, for calculating a particular statistic involving e.g. a pair of attributes, data examples with missing values for one attribute or for the pair can be discarded with small effect on the accuracy of the statistics.
4. Alternatively, a single cluster model can be learned, *off-line*, for the full N-dimensional space.
5. Iterative scaling assumes a uniform pmf over the training set, whereas this pmf is optimized in the ME sense in [10]. Some examples are provided in [14] where optimizing leads to improved results. Moreover, in iterative scaling, based on the uniform pmf assumption, one is implicitly encoding constraints on *all* (high order) collections of features that do not include the target feature F_{i^*}. When the data set is not sufficiently large, this can lead to overfitting. By contrast, the method in [10, 14] allows *selective* encoding of constraints not including the target feature. This leads to improved accuracy in some cases as well [14].
6. This representation will sometimes falsely interpret the user's behavior, e.g. a user may have had very limited time to explore a Web database. Thus, the sites indicated as visited may not accurately reflect the sites the user would be *inclined* to visit in general. However, there is no additional data available for gauging users' intentions.
7. Note that the iterative scaling assumption that the joint pmf takes the form $\frac{N(f_{i_1}, f_{i_2}, \ldots, f_{i_L})}{T} P[F_{i^*} = f_{i^*} \mid f_{i_1}, f_{i_2}, \ldots, f_{i_L}]$ ensures that the ME joint pmf will agree with *all* statistics (measured over the database) involving groups of known features. This may be imprudent, especially for *small* T. An alternative approach that optimizes the joint pmf on the known feature subset so as to agree with selective constraints involving known features is given in [10, 14].

References

1. B. Arwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of Recommendation Algorithms for E-commerce," in *Proc. of the 2nd ACM Conf. on Electronic Commerce*, 2000.

2. A. Berger, S. Della Pietra, and V. Della Pietra, "A Maximum Entropy Approach to Natural Language Processing," *Computational Linguistics*, vol. 22, 1996, pp. 39–68.
3. J. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Proc. of the Fourteenth Conf. on Uncertainty in Art. Intell.*, 1998.
4. A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum-Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Roy. Stat. Soc., Ser. B*, vol. 39, 1977, pp. 1–38.
5. D. Fisher, K. Hildrum, J. Hong, M. Newman, M. Thomas, and R. Vuduc, "Swami: A Framework for Collaborative Filtering Algorithm Development and Evaluation," in *Proc. of the 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Info. Retrieval*, ACM Press, 2000, pp. 366–368.
6. T. Hoffman, "Probabilistic Latent Semantic Indexing," in *Proc. of the 22nd Annual Int. ACM Conf. on Research and Development in Info. Retrieval*, 1999.
7. T. Hoffman and J. Puzicha, "Latent Class Models for Collaborative Filtering," in *Proc. of the IJCAI '99*, 1999.
8. E.T. Jaynes. "Information Theory and Statistical Mechanics," in *Papers on Probability, Statistics and Statistical Physics*, R.D. Rosenkrantz (Ed.), Dordrecht, The Netherlands: Kluwer Academic Publishers, 1989 (Reprint of the original 1957 papers in *Physical Review*).
9. T. Joachims, *Advances in Kernel Methods-Support Vector Learning*, MIT Press, 1999.
10. D.J. Miller and L. Yan, "Approximate Maximum Entropy Joint Feature Inference Consistent with Arbitrary Lower-Order Probability Constraints: Application to Statistical Classification," *Neural Computation*, vol. 12, 2000, pp. 2175–2207.
11. K. Nigam, J. Lafferty, and A. McCallum, "Using Maximum Entropy for Text Classification," in *Proc. of the IJCAI '99 Workshop on Machine Learning for Info. Filtering*, 1999.
12. S. Della Pietra, V. Della Pietra, and J. Lafferty, "Inducing Features of Random Fields," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 19, 1996, pp. 380–393.
13. L. Yan and D.J. Miller, "General Statistical Inference for Discrete and Mixed Spaces by an Approximate Application of the Maximum Entropy Principle," *IEEE Trans. on Neural Networks*, vol. 11, 2000, pp. 558—573.
14. L. Yan and D.J. Miller, "Approximate Maximum Entropy Learning for Classification: Comparison with Other Methods," in *Proc. of the 2001 IEEE Int. Workshop on Neural Networks for Sig. Proc.*, 2001, pp. 243–252.
15. Q. Zhu, "On the Minimum Probability of Error of Classification with Incomplete Patterns," *Pattern Recognition*, vol. 23, 1990, pp. 1281–1290,
16. S.C. Zhu, Y.N. Wu, and D. Mumford, "Minimax Entropy Principle and its Application to Texture Modeling," *Neural Computation*, vol. 9, 1997, pp. 1627–1660.

John Browning received the B.A. degree in math from Rutgers University, New Brunswick NJ, in 1982, and the M.S.E.E. degree in electrical engineering from Drexel University, Philadelphia Pa., in 1996. From February 1982 to August 1997, he was with General Atronics Corp., Wyndmoor, PA. He is currently pursuing the Ph.D. degree and is a research assistant at The Pennsylvania State University. His research interests include statistical pattern recognition and signal processing.
jdb263@psu.edu

David J. Miller received the B.S.E. degree from Princeton University, Princeton, NJ, in 1987, the M.S.E. degree from the University of Pennsylvania, Philadelphia, PA, in 1990, and the Ph.D. degree from the University of California, Santa Barbara, in 1995, all in electrical engineering. From January 1988 through January 1990, he was with General Atronics Corp., Wyndmoor, PA. From Sept. 1995–July 2001 he was Assistant Professor of electrical engineering at The Pennsylvania State University. He is now tenured Associate Professor of electrical engineering at The Pennsylvania State University. His research interests include statistical pattern recognition, machine learning, source coding and coding over noisy channels, and image and video coding. Dr. Miller received the National Science Foundation CAREER Award in 1996. Since 1997, he has been a Member of the Neural Networks for Signal Processing Technical Committee within the IEEE Signal Processing Society. He was Co-general chair for the 2001 IEEE Workshop on Neural Networks for Signal Processing, held in Falmouth, Massachusetts.
millerdj@ee.psu.edu