A FAST IMPORTANCE SAMPLING ALGORITHM FOR UNSUPERVISED LEARNING OF OVER-COMPLETE DICTIONARIES

T. Blumensath, M. Davies

Queen Mary, University of London
Department of Electronic Engineering
Mile End Road, London E1 4NS, UK

ABSTRACT

We use Bayesian statistics to study the dictionary learning problem in which an over-complete generative signal model has to be adapted for optimally sparse signal representations. With such a formulation we develop a stochastic gradient learning algorithm based on Importance Sampling techniques to minimise the negative marginal log-likelihood. As this likelihood is not available analytically, approximations have to be utilised. The Importance Sampling Monte Carlo marginalisation proposed here improves on previous methods and addresses three main issues: 1) bias of the gradient estimate; 2) multi-modality of the distribution to be approximated; and 3) computational efficiency. Experimental results show the advantages of the new method when compared to previous techniques. The gained efficiency allows the treatment of large scale problems in a statistically sound framework as demonstrated here by the extraction of individual piano notes from a polyphonic piano recording.

1. INTRODUCTION

Finding sparse over-complete signal representations has recently attracted increased attention [1]. Not only does such a representation offer an efficient description of the signal [2], it might also make present structure and patterns more apparent [3]. However, previous algorithms solving this problem had three main shortcomings. From a Bayesian point of view a statistical formulation of the problem requires marginalisation over nuisance parameters for which no analytical solution is available. This problem was previously addressed by the use of approximations of this integral around a MAP estimate [4, 2, 5]. However, the resulting approximation of the gradient is not guaranteed to be unbiased. Furthermore, for multi-modal distributions, the MAP estimate can generally not be found as the optimisation algorithms used only converge to local maxima, which might lead to poor approximations. Most importantly, the iterative methods used to find such local maxima are computationally intensive.

This paper addresses these problems by Importance Sampling Monte Carlo approximations of the required integrals. Importance Sampling estimates are unbiased and are not influenced by the number of modes of the distribution to be sampled. Computational efficiency is achieved by utilising 1) a data dependent proposal distribution which can be sampled efficiently; and 2) by using fast methods to calculate the sample weights.

Part of this work was conducted at LabROSA, Columbia University, New York.

2. THEORY

As in [6] we assume a generative model of the form:

$$\mathbf{x}_j = \mathbf{A}\mathbf{s}_j + \epsilon \tag{1}$$

where $\mathbf{x}_j \in \mathbb{R}^M$ is the j^{th} observation, $\mathbf{s}_j \in \mathbb{R}^N$ is an unknown, efficient, alternative representation of the data, $\mathbf{A} \in \mathbb{R}^{M\times N}$ is the dictionary to be learned, $\epsilon \in \mathbb{R}^M$ is a vector of observation noise assumed here to be i.i.d. Gaussian and $N > M$. The data likelihood is then:

$$p(\mathbf{x}_j|\mathbf{A}, \mathbf{s}_j) \sim \mathcal{N}(\mathbf{A}\mathbf{s}_j, \lambda_\epsilon^{-1}\mathbf{I}). \tag{2}$$

In order to enforce sparsity of the coefficients $\mathbf{s}_j$ (and therefore efficiency of the representation) we impose the following mixture prior as in [6].

$$p(\mathbf{s}|\mathbf{u}) = \prod_n p(s_n|u_n) = \prod_n u_n\sqrt{\frac{\lambda_p}{2\pi}}e^{-\frac{\lambda_p}{2}s_n^2} + (1-u_n)\delta_0(s_n) \tag{3}$$

where u_n is a binary indicator variable with discrete distribution:

$$p(u_n) = \frac{1}{1+e^{-\frac{\lambda_u}{2}}}e^{-\frac{\lambda_u}{2}u_n} \tag{4}$$

and $\delta_0(s_n)$ is the Dirac mass at zero. This prior is a mixture of a Gaussian distribution and the Dirac mass, therefore forcing many of the coefficients to be exactly zero with the hyper-prior regulating the sparsity of the distribution.

The parameters defining this model are $\theta = \{\mathbf{A}, \lambda_p, \lambda_u, \lambda_\epsilon\}$[1]. Instead of adopting a fully Bayesian approach to the estimation of these parameters, i.e. instead of specifying prior distributions and calculating their joint posterior distribution or the maximum thereof, we can either assume parameters to be known, or learn them using a stochastic gradient descent algorithm to find the maximum likelihood estimate. The coefficients $\mathbf{s}$ and $\mathbf{u}$ are not of direct interest during model learning and should therefore be integrated out of the data likelihood. The maximum likelihood estimate is then

$$\hat{\theta} = \arg\max_\theta \prod_j \int p(\mathbf{x}_j, \mathbf{s}_j, \mathbf{u}_j|\theta)\, d\{\mathbf{s}_j, \mathbf{u}_j\}. \tag{5}$$

This maximisation can be solved using a stochastic gradient optimisation by approximating the gradient with respect to all data

[1]The model described here has a scale ambiguity. To overcome this we either fix λ_p or re-normalise the dictionary elements after each update.

with the gradient with respect to a single data point $\mathbf{x}_j$. A textbook result for the gradient of the logarithm of a marginalised likelihood with respect to a single data point is

$$\frac{\partial}{\partial\theta}\log p(\mathbf{x}|\theta) = \int p(\mathbf{s},\mathbf{u}|\mathbf{x},\theta)\frac{\partial}{\partial\theta}\log p(\mathbf{x},\mathbf{s},\mathbf{u}|\theta)\, d\mathbf{s}\, d\mathbf{u}, \quad (6)$$

where from now on we drop the index j. This expectation cannot be evaluated analytically in general and different approximations have been proposed in the literature [1, 2, 4], all of which require the calculation of the MAP estimate of $p(\mathbf{s},\mathbf{u}|\mathbf{x},\theta)$. However, for many prior distributions the posterior over the coefficients is multimodal and such estimates then only reflect a section of the distribution and might fail to account for most of the probability mass. Furthermore, such estimates are generally biased, so that convergence to the true maximum of the likelihood is not guaranteed.

Stochastic gradient learning of the parameters randomly iterates through the available data, updating the parameters by a small amount in each iteration. This method therefore averages the gradient over several steps. This suggests the use of a less accurate approximation of the gradient in equation 6 which itself is already a rather poor approximation of the true gradient with respect to all available data. The stochastic gradient algorithm is then still able to converge to a maximum, given the unbiasedness of the approximation is ensured and the learning rate is decreased to zero [7].

We propose a Monte Carlo approximation of the above integral using Importance Sampling [8]. This technique does not rely on MAP estimation and can therefore be implemented efficiently as shown below.

Importance Sampling approximates an integral by a sum of weighted samples.

$$\int p(\mathbf{s},\mathbf{u}|\mathbf{x},\theta)\frac{\partial}{\partial\theta}\log p(\mathbf{x},\mathbf{s},\mathbf{u}|\theta) \approx \sum_i^I w_i \frac{\partial}{\partial\theta} p(\mathbf{x},\hat{\mathbf{s}}_i,\hat{\mathbf{u}}_i|\theta) \quad (7)$$

where $\hat{\mathbf{s}}_i$ and $\hat{\mathbf{u}}_i$ are samples drawn from a proposal distribution $q(\mathbf{s},\mathbf{u})$ with the same support as $p(\mathbf{s},\mathbf{u}|\mathbf{x},\theta)$. The weights are calculated as:

$$w_i = \frac{1}{I}\frac{p(\hat{\mathbf{s}}_i,\hat{\mathbf{u}}_i|\mathbf{x},\theta)}{q(\hat{\mathbf{s}}_i,\hat{\mathbf{u}}_i)} = \frac{1}{I}\frac{p(\hat{\mathbf{s}}_i|\hat{\mathbf{u}}_i,\mathbf{x},\theta)p(\hat{\mathbf{u}}_i|\mathbf{x},\theta)}{q(\hat{\mathbf{s}}_i,\hat{\mathbf{u}}_i)}, \quad (8)$$

which then leads to an unbiased gradient estimate. It can be shown that the above Monte Carlo approximation converges for $I \to \infty$.

3. ALGORITHM

The dictionary learning algorithm is an iterative procedure repeating the three steps below until convergence.

1. Draw a data point $\mathbf{x}_j$ at random from the available training data and draw a set of samples $\hat{\mathbf{s}}_i$ and $\hat{\mathbf{u}}_i$ from the data-dependent proposal distribution $p(\mathbf{s}_n|\mathbf{u}_n,\mathbf{x},\mathbf{A})\alpha(\mathbf{u}_n|\mathbf{x})$.
2. Calculate the weights w_i for each of the samples $\hat{\mathbf{s}}_i$ and $\hat{\mathbf{u}}_i$.
3. Update the parameters θ using a gradient step with a gradient approximation found by Monte Carlo integration using the weighted samples.

Each of the above steps and the required calculations are further discussed below.

3.1. Proposal distribution and sampling

The used mixture prior enables us to draw samples $\hat{\mathbf{s}}$ conditionally on $\hat{\mathbf{u}}$ by setting $\hat{s}_n = 0$ if $\hat{u}_n = 0$. The non-zero coefficients $\hat{\mathbf{s}}_{\o}$ are then Gaussian distributed with variance Λ^{-1} and mean $\Lambda\Lambda_{n,\o}^{-1}\tilde{\mathbf{s}}_{\o}$ where $\tilde{\mathbf{s}}_{\o}$ is the least squares solution to the system $\mathbf{x} = \mathbf{A}_{\o}\mathbf{s}_{\o}$, $\Lambda = (\Lambda_{n,\o} + \lambda_p\mathbf{I})$ and $\Lambda_{n,\o} = \lambda_\epsilon \mathbf{A}_{\o}^T\mathbf{A}_{\o}$. Here the subscript ø refers to a vector or matrix only including the elements associated with the non-zero coefficients u_n, e.g. $\mathbf{A}_{\o}$ is a matrix with those columns of $\mathbf{A}$ which are multiplied by non-zero coefficients s_n. These calculations can be executed efficiently if only a few of the coefficients are non-zero. The distribution $p(\mathbf{u}|\mathbf{x},\theta)$ cannot be sampled efficiently so that we resort to Importance Sampling. The variance of the approximation of the integral in equation 6 then depends not only on the number of samples used but also on the similarity between the proposal density and the density of interest. We therefore specify a proposal density which is proportional to the correlation between each function and the data, hoping that this is a good first approximation of the true density.

$$\alpha(u_n = 1|\mathbf{x}) = p(u_n = 1) * f_n(\mathbf{x}), \quad (9)$$

with

$$f_n(\mathbf{x}) = 2 * \frac{|\mathbf{a}_n^T\mathbf{x}|^{0.4}}{\max_n |\mathbf{a}_n^T\mathbf{x}|}, \quad (10)$$

where $\mathbf{a}_n$ is the n^{th} column of $\mathbf{A}$. The optimal non-linearity in $f_n(\mathbf{x})$ depends on the unknown distribution $p(\mathbf{u}|\mathbf{x},\theta)$ and is therefore problem specific. The above formulation was used in the experiments below.

3.2. Calculating the weights

Equation 8 can be replaced by:

$$\hat{w}_i = \frac{p(\mathbf{x}|\hat{\mathbf{s}}_i,\hat{\mathbf{u}}_i,\theta)p(\hat{\mathbf{s}}_i|\hat{\mathbf{u}}_i)p(\hat{\mathbf{u}}_i)}{p(\hat{\mathbf{s}}_i|\hat{\mathbf{u}}_i,\mathbf{x},\theta)q(\hat{\mathbf{u}}_i)}, \quad (11)$$

$$w_i = \frac{\hat{w}_i}{\sum_i \hat{w}_i}. \quad (12)$$

This is much faster to compute but will introduce a bias. $\sum_i \hat{w}_i$ will however converge to $p(\mathbf{x}|\theta)$ as $I \to \infty$ so that the gradient estimate is unbiased in this limit.

3.3. Updating the parameters

The parameters can be updated in each iteration using a gradient step. The approximations of the gradients are calculated as follows:

$$\Delta\mathbf{A} = \lambda_\epsilon \sum_{i=1}^I w_i(\mathbf{x} - \mathbf{A}\hat{\mathbf{s}}_i)\mathbf{s}_i^T \quad (13)$$

$$\Delta\lambda_\epsilon = \sum_{i=1}^I w_i \left(\frac{M}{2\lambda_\epsilon} - \frac{1}{2}(\mathbf{x} - \mathbf{A}\mathbf{s}_i)^T(\mathbf{x} - \mathbf{A}\mathbf{s}_i)\right) \quad (14)$$

$$\Delta\lambda_p = \sum_{i=1}^I w_i \left(\frac{\mathbf{u}_i^T\mathbf{u}_i}{2\lambda_p} - \frac{1}{2}(\mathbf{s}_i)^T(\mathbf{s}_i)\right) \quad (15)$$

$$\Delta\lambda_u = \sum_{i=1}^I w_i \left. \quad \frac{N}{2(1+e^{\frac{\lambda_u}{2}})} - \frac{1}{2}\mathbf{u}_i^T\mathbf{u}_i\right) \quad (16)$$

4. SHIFT-INVARIANT DICTIONARIES

For time-series analysis it is often desired to learn a dictionary which leads to a representation that is invariant to shifts of the input signal, i.e. a representation that is shifted by the same amount in accordance with an input shift. This can be achieved by enforcing structure on the dictionary $\mathbf{A}$ so that its rows include all possible shifted versions of each of a set of functions $\mathbf{a}_k$ [9, 3]. If we denote the l^{th} shifted version of the k^{th} function as $\mathbf{a}_{kl}$ and use s_{kl} to denote the associated coefficient, the generative model can be written as:

$$\mathbf{x} = \sum_{k \in K, l \in L} \mathbf{a}_{kl} s_{kl} + \epsilon \qquad (17)$$

where $\mathbf{x}$ is now the sum of the output of K linear filters with observation noise.

If $\mathbf{x}$ and $\mathbf{a}_k$ are both in $\mathbb{R}^M$, we can write the gradient update rule for the set of functions $\{\mathbf{a}_k\}$ as [9, 3]:

$$\Delta\{\mathbf{a}_k\} = \lambda_\epsilon \sum_{i=1}^{I} w_i (\mathbf{x} - \mathbf{A}\mathbf{s}_i) \star \mathbf{s}_{ki}, \qquad (18)$$

where $\star$ is the convolution operator.

Our method is applicable to much more general models than this shift-invariant formulation. However, the shift-invariant model normally leads to a high number of coefficients $\mathbf{s}$ as well as to a dictionary $\mathbf{A}$ with highly correlated elements, making the problem particularly difficult to solve. This model will therefore be used in the following as a testbed for our algorithm.

5. RESULTS

5.1. A toy problem

We first generate test data using a dictionary of five functions (and all their shifts) as shown with dashed lines in figure 1. The data was generated with $\lambda_N = 100, \lambda_p = 1$ and $\lambda_u = 9.1902$. The learning rate was slowly decreased using the relationship: $\mu = \mu_{max} \frac{100}{100+iteration}$ with μ_{max} being an appropriate starting value. 100 samples were drawn in each step from the proposal distribution in equation 9. As the time required to compute the weights is proportional to the number of functions selected, it seems advisable to set the starting value of λ_u close to the expected value but erring on the safe side, i.e. preferring higher values to lower ones, keeping the number of functions selected in each sample low until the value has finally converged.

After 1000 iterations the functions shown with solid lines in figure 1 were found. The correlations between learned functions and the true functions a to e in the data generating dictionary were 0.837, 0.9756, 0.5697, 0.9907, 0.9836 respectively. The learned parameters were $\hat{\lambda}_\epsilon = 30.48, \hat{\lambda}_p = 0.8515$ and $\hat{\lambda}_u = 8.96$. From figure 1 it is clear that functions a and c were learned at shifted positions, which explains the relatively poor correlation between the learned and original functions given above. Taking the shift into account, the correlations between the original and learned functions were 0.9365 and 0.9771 for these two functions.

We repeated the same experiment using functions generated as i.i.d Gaussian signals with unit variance. After 2000 iterations the learned parameters were $\hat{\lambda}_\epsilon = 21.88, \hat{\lambda}_p = 0.89$ and $\hat{\lambda}_u = 9.49$ and the correlations between original and learned functions were 0.9888, 0.9416, 0.9021, 0.9454 and 0.6033. However, the

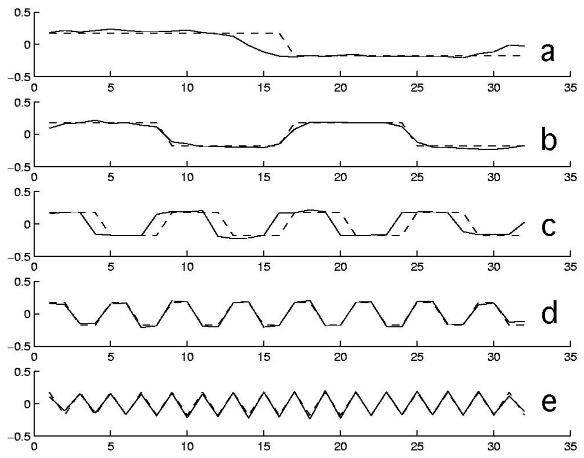

Fig. 1. Toy problem results. The learned functions (solid lines) are very close to the original functions (dotted lines).

last learned function had a higher correlation (0.6742) with the 4^{th} original function.

In both experiments it is evident that the noise scale parameter has always been underestimated.

5.2. Comparison to previous methods

We measured the time required to calculate the MAP estimate using 1) the FOCUSS algorithm [1]; and 2) a standard gradient descent algorithm and compare the results to the generation of 100 samples and the calculation of the associated weights using our approach. In this section we assume that all model parameters (including the not necessarily shift-invariant dictionary) are known.

The FOCUSS algorithm was developed using a different prior formulation to the one used in this paper [1]. For certain parameters it is equivalent to the EM algorithm in [10]. In our experiments we used the Jeffrey's hyper-prior as suggested in [10]. For the gradient descent algorithm we use the improper prior $p(s_n) = s^{-2}$ which is the marginalised prior used in the FOCUSS algorithm when using the Jeffrey's hyper-prior. This algorithm requires many more iterations to converge to the MAP estimate, however the computationally expensive matrix inversion required for the FOCUSS algorithm is not necessary.

Both the computation time for the FOCUSS algorithm and for the gradient descent algorithm depend on the stopping rule employed. We stopped both algorithms when the change in the optimised function was below 0.0001. The FOCUSS algorithm has a computational complexity of $\mathrm{O}(M^3)$, the gradient algorithm of $\mathrm{O}(MN)$ while the proposed sampling algorithm only relies on the average number of non-zero components in the samples.

Figure 2 gives the computation time (using Matlab on a Macintosh G4 1.42 GHz dual processor machine) in seconds (averaged over 10 runs) for the three algorithms and for different dictionary sizes. The sizes of the used dictionarys $\mathbf{A}$ were (from left to right) 64×480, 128×960, 192×1440, 256×1920 and 320×2400, i.e. both M and N were increased linearly. The average number of non-zero coefficients was fixed to 1% so that the average number of non-zero coefficients was also increased linearly. The diamonds in figure 2 show the performance of our method, the crosses are the measurments from the FOCUSS algorithm and the circles are the

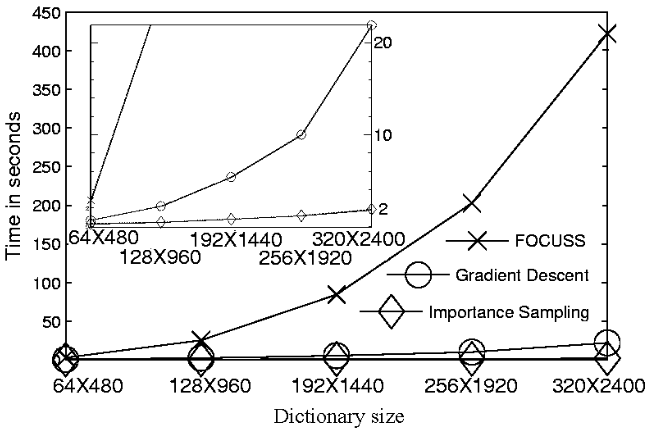

Fig. 2. Computation time for the different algorithms for different dictionary sizes. Top left pannel shows plot zoomed in on y-axis.

measurments from the gradient descent algorithm. The inner pannel in figure 2 shows the graph zoomed in on the y-axis to reveal the difference between the gradient descent and the Importance Sampling algorithms.

5.3. Learning piano notes: A real world problem

The speed advantage of the proposed method allowed us to learn a shift-invarint dictionary ($\mathbf{A} \in \mathbb{R}^{2048\times175104}$) from a polyphonic piano recording. If we assume that a piano note will always have a similar time domain representation, then we should be able to learn a set of functions representing individual piano notes. This problem has been studied previously in [3], however in this work heuristics had to be used to reduce the number of functions in each iteration in order to use the slower FOCUSS algorithm.

We use a recording of L. van Beethoven's Bagatelle No. 1 Opus 33 which we summed to mono and resampled at 8000 Hz. We learned 57 functions (the number of different notes in the recording), all of which converged to periodic functions. In figure 3 we show one of the learned functions. The top panel displays the time domain representation whilst the lower panel shows the power spectrum of this function. Here the harmonic structure of the learned functions is clearly visible. The learned parameters were $\hat{\lambda}_\epsilon = 321$, $\hat{\lambda}_p = 1.056$ and $\hat{\lambda}_u = 23.02$.

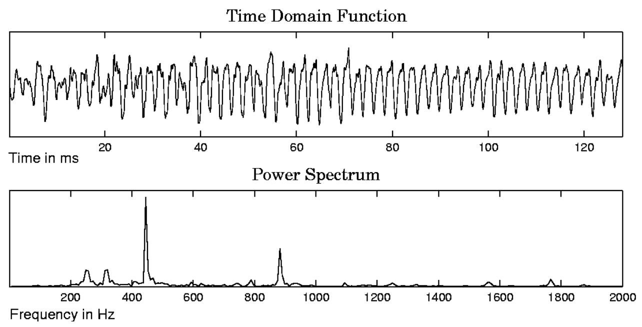

Fig. 3. One of the learned piano notes. The harmonic structure is clearly visible.

6. CONCLUSION

Bayesian learning can lead to computationally intractable problems for even simple models. This paper investigated a linear generative model with parameterised distributions for which both the dictionary as well as the parameters of the distributions had to be determined. We also assumed the coefficients $\mathbf{s}$ and $\mathbf{u}$ to be unknown, which led to the required integration over the posterior distribution of these coefficients in the calculation of the gradient in the parameter space. We have shown experimentally that this integration can be approximated efficiently using an Importance Sampling Monte Carlo method. The used stochastic gradient maximisation of the parameters effectively averaged over the gradients. We have shown that the performance of this algorithm is superior in speed to previously suggested methods which rely on MAP estimation. It should further be noted that the suggested approach can easily be extended to other prior distributions for the coefficients, so that different constraints can be implemented without significantly changing the algorithm. Finally we have demonstrated that the method could be used for time-series analysis of real world data. The speed of the computations allowed the application of formal Bayesian theory, superior to the previously suggested ad hoc methods.

7. REFERENCES

[1] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Computation*, vol. 15, pp. 349–396, 2003.

[2] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural Computation*, no. 12, pp. 337–365, 2000.

[3] T. Blumensath and M. Davies, "Unsupervised learning of sparse and shift-invariant decompositions of polyphonic music," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004.

[4] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, no. 381, pp. 607–609, 1995.

[5] S. Abdallah, *Towards Music Perception by Redundancy Reduction and Unsupervised Learning in Probabilistic Models*. PhD thesis, King's College London, February 2003.

[6] P. Sallee and B. A. Olshausen, "Learning sparse multiscale image representations," *Advances in Neural Information Processing Systems*, 2003.

[7] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[8] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer texts in statistics, Springer-Verlag, 1999.

[9] H. Wersing, J. Eggert, and E. Körner, "Sparse coding with invariance constraints," in *Proc. Int. Conf. Artificial Neural Networks ICANN*, pp. 385–392, 2003.

[10] M. A. T. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1150–1159, Sept. 2003.