A Fast Incremental Gaussian Mixture Model - Submission to PLOS Journals

Rafael Pinto[1], Paulo Engel[1],

1 Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brazil

Av. Bento Gonçalves, 9500 - Agronomia - Porto Alegre, RS - Zip 91501-970 - Brazil
{rcpinto,engel}@inf.ufrgs.br

Abstract

This work builds upon previous efforts in online incremental learning, namely the Incremental Gaussian Mixture Network (IGMN). The IGMN is capable of learning from data streams in a single-pass by improving its model after analyzing each data point and discarding it thereafter. Nevertheless, it suffers from the scalability point-of-view, due to its asymptotic time complexity of $\mathrm{O}\big(NKD^3\big)$ for N data points, K Gaussian components and D dimensions, rendering it inadequate for high-dimensional data. In this paper, we manage to reduce this complexity to $\mathrm{O}\big(NKD^2\big)$ by deriving formulas for working directly with precision matrices instead of covariance matrices. The final result is a much faster and scalable algorithm which can be applied to high dimensional tasks. This is confirmed by applying the modified algorithm to high-dimensional classification datasets.

1 Introduction

The Incremental Gaussian Mixture Network (IGMN) [1, 2] is a supervised algorithm which approximates the EM algorithm [3]. It creates and continually adjusts a probabilistic model consistent to all sequentially presented data, after each data point presentation, and without the need to store any past data points. Its learning process is aggressive, meaning that only a single scan through the data is necessary to obtain a consistent model.

IGMN adopts a Gaussian mixture model of distribution components that can be expanded to accommodate new information from an input data point, or reduced if spurious components are identified along the learning process. Each data point assimilated by the model contributes to the sequential update of the model parameters based on the maximization of the likelihood of the data. The parameters are updated through the accumulation of relevant information extracted from each data point.

The IGMN is capable of supervised learning, simply by assigning any of its input vector elements as outputs (any element can be used to predict any other element, like autoassociative neural networks [4]). This feature is useful for simultaneous learning of forward and inverse kinematics, as well as for simultaneous learning of a value function and a policy in reinforcement learning [5].

However, the IGMN suffers from cubic time complexity due to matrix inversion operations and determinant computations. Its time complexity is of $\mathrm{O}\big(NKD^3\big)$, where

N is the number of data points, K is the number of Gaussian components and D is the problem dimension. It makes the algorithm prohibitive for high-dimensional tasks (like visual tasks) and thus of limited use. One solution would be to use diagonal covariance matrices, but this decreases the quality of the results, as already reported in previous work [6, 7]. In this work, we propose the use of rank-one updates for both inverse matrices and determinants applied to full covariance matrices, thus reducing the time complexity to $\mathrm{O}\big(NKD^2\big)$ for learning while keeping the quality of a full covariance matrix solution.

For the specific case of the IGMN algorithm, to the best of our knowledge, this has not been tried before, although we can find similar efforts for related algorithms. In [8], rank-one updates were applied to an iterated linear discriminant analysis algorithm in order to decrease the complexity of the algorithm. Rank-one updates were also used in [9], where Gaussian models are employed for feature selection. Finally, in [10], the same kind of optimization was applied to Maximum Likelihood Linear Transforms (MLLT).

The next Section describes the algorithm in more detail with the latest improvements to date. Section 3 describes our improvements to the algorithm. Section 4 shows the experiments and results obtained from both versions of the IGMN for comparison, and Section 5 finishes this work with concluding remarks.

2 Incremental Gaussian Mixture Network

In the next subsections we describe the current version of the IGMN algorithm.

2.1 Learning

The algorithm starts with no components, which are created as necessary (see subsection 2.2). Given input $\mathbf{x}$ (a single instantaneous data point), the IGMN algorithm processing step is as follows. First, the squared Mahalanobis distance $d^2(\mathbf{x}, j)$ for each component j is computed:

$$d^2_M(\mathbf{x}, j) = (\mathbf{x} - \boldsymbol{\mu}_j)^T \mathbf{C}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \tag{1}$$

where $\boldsymbol{\mu}_j$ is the j^{th} component mean, $\mathbf{C}_j$ its full covariance matrix . If any $d^2(\mathbf{x}, j)$ is smaller than than $\chi^2_{D,1-\beta}$ (the $1 - \beta$ percentile of a chi-squared distribution with D degrees-of-freedom, where D is the input dimensionality and β is a user defined meta-parameter, e.g., 0.1), an update will occur, and posterior probabilities are calculated for each component as follows:

$$p(\mathbf{x}|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_j|}} \exp\left(-\frac{1}{2} d^2_M(\mathbf{x}, j)\right) \tag{2}$$

$$p(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{\sum\limits_{k=1}^{K} p(\mathbf{x}|k)p(k)} \quad \forall j \tag{3}$$

where K is the number of components. Now, parameters of the algorithm must be updated according to the following equations:

$$v_j(t) = v_j(t-1) + 1 \tag{4}$$

$$sp_j(t) = sp_j(t-1) + p(j|\mathbf{x}) \tag{5}$$

$$\mathbf{e}_j = \mathbf{x} - \boldsymbol{\mu}_j(t-1) \tag{6}$$

$$\omega_j = \frac{p(j|\mathbf{x})}{sp_j} \tag{7}$$

$$\Delta\boldsymbol{\mu}_j = \omega_j \mathbf{e}_j \tag{8}$$

$$\boldsymbol{\mu}_j(t) = \boldsymbol{\mu}_j(t-1) + \Delta\boldsymbol{\mu}_j \tag{9}$$

$$\mathbf{e}^*_j = \mathbf{x} - \boldsymbol{\mu}_j(t) \tag{10}$$

$$\mathbf{C}_j(t) = (1-\omega_j)\mathbf{C}_j(t-1) + \omega_j \mathbf{e}^*_j \mathbf{e}^{*T}_j - \Delta\boldsymbol{\mu}_j \Delta\boldsymbol{\mu}^T_j \tag{11}$$

$$p(j) = \frac{sp_j}{\sum\limits_{q=1}^{M} sp_q} \tag{12}$$

where sp_j and v_j are the accumulator and the age of component j, respectively, and $p(j)$ is its prior probability.

2.2 Creating New Components

If the update condition in the previous subsection is not met, then a new component j is created and initialized as follows:

$$\boldsymbol{\mu}_j = \mathbf{x}; \quad sp_j = 1; \quad v_j = 1; \quad p(j) = \frac{1}{\sum\limits_{i=1}^{K} sp_i}; \quad \mathbf{C}_j = \boldsymbol{\sigma}^2_{ini}\mathbf{I}$$

where K already includes the new component and $\boldsymbol{\sigma}_{ini}$ can be obtained by:

$$\boldsymbol{\sigma}_{ini} = \delta std(\mathbf{x}) \tag{13}$$

where δ is a manually chosen scaling factor (e.g., 0.01) and std is the standard deviation of the dataset. Note that the IGMN is an online and incremental algorithm and therefore it may be the case that we do not have the entire dataset to extract descriptive statistics. In this case the standard deviation can be just an estimation (e.g., based on sensor limits from a robotic platform), without impacting the algorithm.

2.3 Removing Spurious Components

A component j is removed whenever $v_j > v_{min}$ and $sp_j < sp_{min}$, where v_{min} and sp_{min} are manually chosen (e.g., 5.0 and 3.0, respectively). In that case, also, $p(k)$ must be adjusted for all $k \in K$, $k \neq j$, using (12). In other words, each component is given some time v_{min} to show its importance to the model in the form of an accumulation of its posterior probabilities sp_j.

2.4 Inference

In the IGMN, any element can be predicted by any other element. This is done by reconstructing data from the target elements ($\mathbf{x}_t$, a slice of the entire input vector $\mathbf{x}$) by estimating the posterior probabilities using only the given elements ($\mathbf{x}_i$, also a slice of the entire input vector $\mathbf{x}$), as follows:

$$p(j|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|j)p(j)}{\sum_{q=1}^{M} p(\mathbf{x}_i|q)p(q)} \quad \forall j \tag{14}$$

It is similar to (3), except that it uses a modified input vector $\mathbf{x}_i$ with the target elements $\mathbf{x}_t$ removed from calculations. After that, $\mathbf{x}_t$ can be reconstructed using the conditional mean equation:

$$\hat{\mathbf{x}_t} = \sum_{j=1}^{M} p(j|\mathbf{x}_i)(\boldsymbol{\mu}_{j,t} + \mathbf{C}_{j,ti}\mathbf{C}_{j,i}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_{j,i})) \tag{15}$$

where $\mathbf{C}_{j,ti}$ is the submatrix of the jth component covariance matrix associating the unknown and known parts of the data, $\mathbf{C}_{j,i}$ is the submatrix corresponding to the known part only and $\boldsymbol{\mu}_{j,i}$ is the jth's component mean without the element corresponding to the target element.

3 Fast IGMN

One of the contributions of this work lies in the fact that Equation 1 (the squared Mahalanobis distance) requires a matrix inversion, which has a asymptotic time complexity of $\mathrm{O}(D^3)$, for D dimensions ($\mathrm{O}(D^{log_2 7 + \mathrm{O}(1)})$ for the Strassen algorithm or at best $\mathrm{O}(D^{2.3728639})$ with the most recent algorithms to date [11]). This renders the entire IGMN algorithm as impractical for high-dimension tasks. Here we show how to work directly with the inverse of covariance matrix (also called the precision or concentration matrix) for the entire procedure, therefore avoiding costly inversions.

Firstly, let us denote $\mathbf{C}^{-1} = \boldsymbol{\Lambda}$, the precision matrix. Our task is to adapt all equations involving $\mathbf{C}$ to instead use $\boldsymbol{\Lambda}$.

We now proceed to adapt Equation 11 (covariance matrix update). This equation can be seen as a sequence of two rank-one updates to the $\mathbf{C}$ matrix, as follows:

$$\bar{\mathbf{C}}_j(t) = (1 - \omega_j)\mathbf{C}_j(t-1) + \omega_j \mathbf{e}_j^* \mathbf{e}_j^{*T} \tag{16}$$

$$\mathbf{C}_j(t) = \bar{\mathbf{C}}_j(t) - \Delta\boldsymbol{\mu}_j \Delta\boldsymbol{\mu}_j^T \tag{17}$$

This allows us to apply the Sherman-Morrison formula [12]:

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}} \tag{18}$$

This formula shows how to update the inverse of a matrix plus a rank-one update. For the second update, which subtracts, the formula becomes:

$$(\mathbf{A} - \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} + \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1 - \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}} \tag{19}$$

In the context of IGMN, we have $\mathbf{A} = (1-\omega)\mathbf{C}_j(t-1) = (1-\omega)\boldsymbol{\Lambda}_j^{-1}(t-1)$ and $\mathbf{u} = \mathbf{v} = \sqrt{\omega}\mathbf{e}^*$ for the first update, while for the second one we have $\mathbf{A} = \bar{\mathbf{C}}_j(t)$ and

$\mathbf{u} = \mathbf{v} = \Delta\boldsymbol{\mu}_j$. Rewriting 18 and 19 we get (for the sake of compactness, assume all subscripts for $\boldsymbol{\Lambda}$ and $\Delta\boldsymbol{\mu}$ to be j):

$$\bar{\boldsymbol{\Lambda}}(t) = \frac{\boldsymbol{\Lambda}(t-1)}{1-\omega} - \frac{\frac{\omega}{(1-\omega)^2}\boldsymbol{\Lambda}(t-1)\mathbf{e}^*\mathbf{e}^{*T}\boldsymbol{\Lambda}(t-1)}{1+\frac{\omega}{1-\omega}\mathbf{e}^{*T}\boldsymbol{\Lambda}(t-1)\mathbf{e}^*} \tag{20}$$

$$\boldsymbol{\Lambda}(t) = \bar{\boldsymbol{\Lambda}}(t) + \frac{\bar{\boldsymbol{\Lambda}}(t)\Delta\boldsymbol{\mu}\Delta\boldsymbol{\mu}^T\bar{\boldsymbol{\Lambda}}(t)}{1-\Delta\boldsymbol{\mu}^T\bar{\boldsymbol{\Lambda}}(t)\Delta\boldsymbol{\mu}} \tag{21}$$

These two equations allow us to update the precision matrix directly, eliminating the need for the covariance matrix $\mathbf{C}$. They have $\mathrm{O}(N^2)$ complexity due to matrix-vector products.

Following on the adaptation of the IGMN equations, Equation 1 (the squared Mahalanobis distance) allows for a direct substituion, yielding the following new equation:

$$d_M^2(\mathbf{x}, j) = (\mathbf{x} - \boldsymbol{\mu}_j)^T\boldsymbol{\Lambda}_j(\mathbf{x} - \boldsymbol{\mu}_j) \tag{22}$$

which now has a $\mathrm{O}(N^2)$ complexity, since there is no matrix inversion as the original equation. After removing the cubic complexity from this step, the determinant computation will be dealt with next.

Since the determinant of the inverse of a matrix is simply the inverse of the determinant, it is sufficient to invert the result. But computing the determinant itself is also a $\mathrm{O}(D^3)$ operation, so we will instead perform rank-one updates using the Matrix Determinant Lemma [13], which states the following:

$$|\mathbf{A} + \mathbf{u}\mathbf{v}^T| = |\mathbf{A}|(1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}) \tag{23}$$

$$|\mathbf{A} - \mathbf{u}\mathbf{v}^T| = |\mathbf{A}|(1 - \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}) \tag{24}$$

Since the IGMN covariance matrix update involves a rank-two update, adding a term and then subtracting one, both rules must be applied in sequence, similar to what has been done with the $\boldsymbol{\Lambda}$ equations. Equations 16 and 17 may be reused here, together with the same substitutions previously showed, leaving us with the following new equations for updating the determinant (again, j subscripts were dropped):

$$|\bar{\mathbf{C}}(t)| = (1-\omega)^D|\mathbf{C}(t-1)|\left(1 + \frac{\omega}{1-\omega}\mathbf{e}^{*T}\boldsymbol{\Lambda}(t-1)\mathbf{e}^*\right) \tag{25}$$

$$|\mathbf{C}(t)| = |\bar{\mathbf{C}}(t)|(1 - \Delta\boldsymbol{\mu}^T\bar{\boldsymbol{\Lambda}}(t)\Delta\boldsymbol{\mu}) \tag{26}$$

This was the last source of cubic complexity, which is now quadratic.

Finishing the adaptation in the learning part of the algorithm, we just need to define the initialization for $\boldsymbol{\Lambda}$ for each component. What previously was $\mathbf{C}_j = \boldsymbol{\sigma}_{ini}^2\mathbf{I}$ now becomes $\boldsymbol{\Lambda}_j = \boldsymbol{\sigma}_{ini}^{-2}\mathbf{I}$, the inverse of the variances of the dataset. Since this matrix is diagonal, there are no costly inversions involved. And for initializing the determinant $|\mathbf{C}|$, just set it to $\prod \boldsymbol{\sigma}_{ini}^2$, which again takes advantage of the initial diagonal matrix to avoid costly operations. Note that we keep the precision matrix $\boldsymbol{\Lambda}$, but the determinant of the covariance matrix $\mathbf{C}$ instead. See algorithms 1 to 3 for a summary of the new learning algorithm (excluding pruning, for brevity).

Finally, the inference Equation 15 must also be updated in order to allow the IGMN to work in supervised mode. This can be accomplished by the use of a block matrix decomposition (note that here $\mathbf{C}$ is just another sub-matrix, not the covariance matrix as used before):

Algorithm 1 Fast IGMN Learning

```
Input: $\delta$,$\beta$,$\mathbf{X}$
  $K > 0$, $\boldsymbol{\sigma}_{ini}^{-1} = (\delta std(\mathbf{X}))^{-1}, M = \emptyset$
  for all input data vector $\mathbf{x} \in \mathbf{X}$ do
    if $K = 0$ or $\exists j,\ d_M^2(\mathbf{x}, j) < \chi^2_{D,1-\beta}$ then
      $update(\mathbf{x})$
    else
      $M \leftarrow M \cup create(\mathbf{x})$
    end if
  end for
```

Algorithm 2 update

```
Input: $\mathbf{x}$
  for all Gaussian component $j \in M$ do
    Compute equations 1 to 12 substituting 1 for 22 and 11 for 20 and 21
    Compute equations 25 and 26
  end for
```

$$\boldsymbol{\Lambda} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{Z} & \mathbf{W} \end{bmatrix} =$$

$$\begin{bmatrix} (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix}$$

Here, according to Equation 15, we need $\mathbf{C}$ and $\mathbf{A}^{-1}$. But since the terms that constitute these sub-matrices are relative to the original covariance matrix (which we do not have), they must be extracted from the precision matrix directly. Looking at the decomposition, it is clear that $\mathbf{Y}\mathbf{W}^{-1} = -\mathbf{A}^{-1}\mathbf{B} = -\mathbf{C}\mathbf{A}^{-1}$ (the terms between parenthesis in $\mathbf{Y}$ and $\mathbf{W}$ cancel each other, while $\mathbf{B} = \mathbf{C}^T$ due to symmetry). So Equation 15 can be rewritten as:

$$\hat{\mathbf{x}_t} = \sum_{j=1}^{M} p(j|\mathbf{x}_i)(\boldsymbol{\mu}_{j,t} - \mathbf{Y}\mathbf{W}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_{j,i})) \qquad (27)$$

where $\mathbf{Y}$ and $\mathbf{W}$ can be extracted directly from $\boldsymbol{\Lambda}$. However, we still need to compute the inverse of $\mathbf{W}$. So we can say that this particular implementation has $\mathrm{O}(NKD^2)$ complexity for learning and $\mathrm{O}(NKD^3)$ for inference. The reason for us to not worry about that is that $d = i + o$, where i is the number of inputs and o is the number of outputs. The inverse computation acts only upon the output portion of the matrix. Since, in general, $o \ll i$ (in many cases even $o = 1$), the impact is minimal, and the same applies to the $\mathbf{Y}\mathbf{W}^{-1}$ product. In fact, Weka (the data mining platform used in this work [14]) allows for only 1 output, leaving us with just scalar operations.

4 Experiments

In order to evaluate the performance of the proposed algorithm, 11 classification tasks (Table 1) were given to the original and improved IGMN algorithms ($\delta = 1$ and $\beta = 0$, so a single component was created for each run and we could focus on speed ups only due to dimensionality). Results were obtained from 2-fold cross-validation and statistical significances came from paired t-tests with $p = 0.05$.

This experiment was meant to verify that both IGMN implementations produce exactly the same results, which was confirmed, as well as to show that the proposed

Algorithm 3 create

```
Input: x
    $K \leftarrow K + 1$
    return new Gaussian component $K$ with $\boldsymbol{\mu}_K = \mathbf{x}$, $\boldsymbol{\Lambda}_K = \boldsymbol{\sigma}_{ini}^{-1}\mathbf{I}$, $|\mathbf{C}_K| = |\boldsymbol{\Lambda}_K|^{-1}$, $sp_j = 1$, $v_j = 1$, $p(j) = \frac{1}{\sum_{k=1}^{K} sp_i}$
```

Table 1. Datasets

Dataset	Instances (N)	Attributes (D)	Classes
breast-cancer	286	9	2
german-credit	1000	20	2
pima-diabetes	768	8	2
Glass	214	9	7
ionosphere	351	34	2
iris	150	4	3
labor-neg-data	57	16	2
soybean	683	35	19
twospirals	193	2	2
MNIST [15] (subset)	1000	784	10
CIFAR-10 [16] (subset)	1000	3072	10
CIFAR-10b (subset)	100	3072	10

improvements really deliver the expected speed up (the Weka packages for both variations of the IGMN algorithm can be found at `http://www.inf.ufrgs.br/~rcpinto`). This was also confirmed, as can be seen in tables 2 and 3 (note that the experiments were divided into training and test phases just for comparison purposes, but IGMN is in fact an online algorithm; also note that standard deviations were rounded to 3 decimal places and, in fact, there are not any null standard deviations in the results). Our improved algorithm could deliver a speed up of 2 orders of magnitude in training time (learning) for the CIFAR-10 subset, which follows our expectations according to the new time complexity. Datasets with less dimensions could benefit from the improvements too, albeit in a smaller scale, which was also expected. The other confirmation came from the testing times (inference): they were also improved, since the inverse matrix computation was eliminated from the probability density equation, which is also necessary for inference, and the matrix multiplications and inversions are actually scalar operations, having no impact. In fact, the speed up for inference was around 3 orders of magnitude for the CIFAR-10 subset.

Table 2. Training Time (in seconds)

Dataset	IGMN	Fast IGMN
breast-cancer	0.010± 0.004	0.006 ± 0.000
german-credit	0.031± 0.012	0.016 ± 0.000
pima-diabetes	0.013± 0.000	0.010 ± 0.001
Glass	0.008± 0.000	0.005 ± 0.000 ●
ionosphere	0.022± 0.002	0.014 ± 0.002 ●
iris	0.005± 0.000	0.007 ± 0.001
labor-neg-data	0.006± 0.001	0.007 ± 0.001
soybean	0.042± 0.004	0.030 ± 0.003
twospirals	0.005± 0.001	0.006 ± 0.001 ○
MNIST	281.257± 3.157	10.675 ± 0.272 ●
CIFAR-10	20768.494±1244.221	175.243 ± 1.190 ●
Average	1754.417	15.526

○, ● statistically significant increase or decrease in time

Although not the focus of this work, we could also observe the accuracy of the IGMN algorithm on the tested datasets in comparison to other algorithms available in the Weka software, like Support Vector Machines (SVM) and the state-of-the-art

Table 3. Testing Time (in seconds)

Dataset	IGMN		Fast IGMN		
breast-cancer	0.001±	0.000	0.001	±	0.001
german-credit	0.013±	0.004	0.002	±	0.000
pima-diabetes	0.004±	0.000	0.001	±	0.000 ●
Glass	0.001±	0.000	0.001	±	0.000
ionosphere	0.010±	0.001	0.008	±	0.008
iris	0.000±	0.000	0.001	±	0.000 ○
labor-neg-data	0.000±	0.000	0.001	±	0.001
soybean	0.026±	0.007	0.004	±	0.000
twospirals	0.000±	0.000	0.000	±	0.000
MNIST	225.262±	5.638	0.607	±	0.149 ●
CIFAR-10	17821.407±	1699.599	7.793	±	0.098
Average	1504.097		0.702		

○, ● statistically significant increase or decrease in time

Table 4. Area Under ROC Curve

Dataset	Neural Network		1-NN	Naive Bayes	SVM	IGMN	FIGMN
breast-cancer	0.65±	0.01	0.59±0.02	0.70±0.02	0.61±0.04	0.60±0.00	0.60±0.00
CIFAR-10b	0.83±	0.03	0.61±0.19	0.51±0.03	0.64±0.14	0.62±0.20	0.62±0.20
german-credit	0.79±	0.01	0.65±0.01 ●	0.79±0.00	0.60±0.01	0.62±0.03	0.62±0.03
pima-diabetes	0.84±	0.00	0.64±0.02	0.81±0.01	0.73±0.02	0.69±0.03	0.69±0.03
Glass	0.81±	0.02	0.78±0.05	0.70±0.10	0.72±0.12	0.79±0.04	0.79±0.04
ionosphere	0.95±	0.03	0.81±0.01	0.93±0.00	0.82±0.03 ●	0.90±0.03	0.90±0.03
iris	1.00±	0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00
labor-neg-data	0.93±	0.01	0.79±0.07	0.94±0.02	0.94±0.02	0.91±0.02	0.91±0.02
MNIST	1.00±	0.00	0.97±0.03	0.96±0.00	0.97±0.04	0.95±0.05	0.95±0.05
soybean	1.00±	0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00
twospirals	0.50±	0.08	0.76±0.02	0.48±0.00	0.47±0.03	0.61±0.12	0.61±0.12
Average	0.84		0.78	0.80	0.77	0.79	0.79

● statistically significant degradation

Dropout Neural Networks [17] (with 50 hidden neurons, 50% dropout for the hidden layer and 20% dropout for the input layer; the specific implementation can be found at https://github.com/amten/NeuralNetwork). It was statistically similar to them with $\beta = 0.001$ (now more than one Gaussian component was allowed) and tuning the δ parameter by 2-fold cross-validation using 3 different values (0.01, 0.1 and 1), but with the additional benefit of a single-scan through data, allowing it to operate on data streams. Results are shown in table 4 (note that, for this experiment, a smaller subset of CIFAR-10 was used, in order to compensate for the higher computational requirements of more Gaussian components).

5 Conclusion

We have shown how to work directly with precision matrices in the IGMN algorithm, avoiding costly matrix inversions by performing rank-one updates. The determinant computations were also avoided using a similar method, effectively eliminating any source of cubic complexity for the learning algorithm. This resulted in substantial speed ups for high-dimensional datasets, turning the IGMN into a good option for this kind of tasks. The inference operation still has cubic complexity, but we argue that it has a much smaller impact on the total runtime of the algorithm, since the number of outputs is usually much smaller than the number of inputs. This was confirmed for one-dimensional outputs, which require only scalar operations.

In general, we could see that the fast IGMN is a good option for supervised learning, with low runtimes and good accuracy after adjusting the two main meta-parameters β and δ. It should be noted that this is achieved with a single-pass through the data, making it also a valid option for data streams.

References

1. Heinen MR, Engel PM, Pinto RC. Using a Gaussian mixture neural network for incremental learning and robotics. In: Neural Networks (IJCNN), The 2012 International Joint Conference on. IEEE; 2012. p. 1–8.
2. Heinen MR, Engel PM, Pinto RC. IGMN: An incremental gaussian mixture network that learns instantaneously from data flows. Proc VIII Encontro Nacional de Inteligência Artificial (ENIA2011). 2011;.
3. Engel P, Heinen M. Incremental learning of multivariate gaussian mixture models. Advances in Artificial Intelligence–SBIA 2010. 2011;p. 82–91.
4. Rumelhart DE, McClelland JL. Parallel distributed processing. MIT Pr.; 1986.
5. Heinen MR, Engel PM. IGMN: An incremental connectionist approach for concept formation, reinforcement learning and robotics. Journal of Applied Computing Research. 2011;1(1):2–19.
6. Heinen MR. A connectionist approach for incremental function approximation and on-line tasks. Universidade Federal do Rio Grande do Sul - Instituto de Informática; 2011.
7. Pinto RC, Engel PM, Heinen MR. Echo State Incremental Gaussian Mixture Network for Spatio-Temporal Pattern Processing. In: Proceedings of the IX ENIA-Brazilian Meeting on Artificial Intelligence, Natal (RN); 2011. .
8. Salmen J, Schlipsing M, Igel C. Efficient update of the covariance matrix inverse in iterated linear discriminant analysis. Pattern Recognition Letters. 2010;31(13):1903 – 1907. Meta-heuristic Intelligence Based Image Processing. Available from: `http://www.sciencedirect.com/science/article/pii/S0167865510000826`.
9. Lefakis L, Fleuret F. Jointly Informative Feature Selection. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics; 2014. p. 567–575.
10. Olsen PA, Gopinath RA. Extended mllt for gaussian mixture models. Transactions in Speech and Audio Processing. 2001;.
11. Gall FL. Powers of tensors and fast matrix multiplication. arXiv preprint arXiv:14017714. 2014;.
12. Sherman J, Morrison WJ. Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. The Annals of Mathematical Statistics. 1950 03;21(1):124–127. Available from: `http://dx.doi.org/10.1214/aoms/1177729893`.
13. Harville DA. Matrix algebra from a statistician's perspective. Springer; 2008.
14. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: an update. ACM SIGKDD explorations newsletter. 2009;11(1):10–18.
15. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278–2324.

16. Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. Computer Science Department, University of Toronto, Tech Rep. 2009;.

17. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:12070580. 2012;.