Furthering the Application of Machine Learning to the Prediction of Oceanic Plankton Biomass

Rory Grandin
Department of Computer Science
University of Cape Town
Rondebosch, 7701
South Africa
rgrandin@cs.uct.ac.za

Hayley McIntosh
Department of Computer Science
University of Cape Town
Rondebosch, 7701
South Africa
hmcintos@cs.uct.ac.za

Andrew Symington
Department of Computer Science
University of Cape Town
Rondebosch, 7701
South Africa
andrew@fpef.co.za

ABSTRACT

The Plankton Prediction System (PPS) is a joint project between the Computer Science and Zoology departments of the University of Cape Town. Its purpose is to research and develop machine-learning software capable of predicting the level and distribution of subsurface oceanic chlorophyll, given related data. In so doing the PPS provides marine biologists with valuable information that would otherwise be both time-consuming and expensive to retrieve.

The work outlined in this paper furthers earlier research [9] by Fenn, Curtis and Oberholzer and, as well as addressing a few shortcomings, expands upon a number of topics that demanded closer investigation. The following five items were chosen by the 2006 project team as core research areas:

1. The production of a more structured and coherent set of data from which to perform predictions.
2. The effect of various clustering algorithms on depth profile data.
3. The use of a dynamic Bayesian network to incorporate the effect of time on chlorophyll predictions.
4. The use of topic maps as a means to dynamically display the relationship between data.
5. A greater degree of accompanying documentation and modular design.

It is best to think of the work outlined in this paper as three stages in a pipeline. The first stage, *preprocessing*, is responsible for the integration of all the raw data from a number of different sources. After integration, the data is further discretized through a clustering process, which reduces its complexity. The second stage, *prediction*, is responsible for training a Dynamic Bayesian Network (DBN) with the clustered data produced in the *preprocessing* stage. Once training is complete, absent sub-surface chlorophyll data is inferred from the resultant network. The final stage in the PPS pipeline concerns itself with the *visualization* of the results obtained from both the *preprocessing* and *prediction* stages. Technologies, such as Topic Maps and hypergraphs are implemented to create a dynamic view of the relationship between data. Moreover, inference results are rendered as colour rasters for viewing within the web-based PPS interface.

CATEGORIES & SUBJECT DESCRIPTORS

I 2.6 [**Artificial Intelligence**] *Learning – Concept learning*
I 5.1 [**Pattern Recognition**] *Models*
I 5.3 [**Pattern Recognition**] *Clustering*

GENERAL TERMS

Algorithms, Design.

KEYWORDS

Clustering, Machine Learning, Dynamic Bayesian Networks, Visualisation, Topic Maps, Chlorophyll, Plankton

1. INTRODUCTION

Plankton levels are currently estimated using the amount of chlorophyll in the ocean. Plankton is composed of organic material, of which chlorophyll is roughly twenty percent. Chlorophyll can be measured by satellites and the most recent satellite to do so is SeaWifs. Plankton levels can indicate the likeliness of finding fish in the area and even particular fish species.

Since plankton levels can be inferred from chlorophyll, it was proposed by the marine biology department and in particular Professor John Field to develop a system to predict the amount of chlorophyll present around the Southern Africa coast on a weekly basis. To do so, new technology in the form of a Dynamic Bayesian Network (DBN) was introduced as well as the necessity to represent this structure intuitively. The introduction also necessitated the need for new chlorophyll profiles to use within the DBN. The Plankton Prediction System (PPS) was divided between the three team members; pre-processing, prediction and visualization.

2. BACKGROUND

2.1 Depth Profiles

A chlorophyll depth profile is constructed from a set of evently-spaced readings taken from below the surface of the ocean. The readings are usually spaced at meter intervals and begin one meter below the surface of the ocean. The readings continue to a depth of 100 meters. Each point represents the quantity (in mg.m^{-3}) of chlorophyll at that depth. Figure 1 shows an interpolated depth profile, with the lighter region indicating the curve's integrated chlorophyll value. These readings are recorded by ships at number of stations around the Southern African coastline.

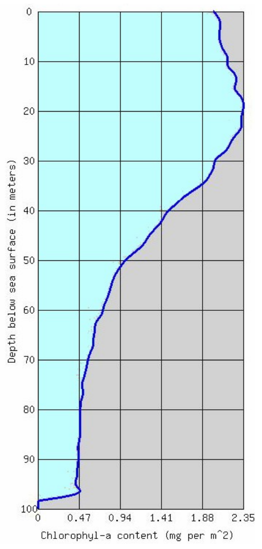


Figure 1: A sample chlorophyll depth profile

2.2 Interpolation

In order to resample the depth profile data, which is currently in point readings, a curve will first need to be fitted to each depth profile. There are a number of different interpolation techniques available. Common examples are *linear interpolation, polynomial interpolation, cubic spline interpolation* and *Akima interpolation* [3].

Unlike its counterparts, *Akima interpolation* provides a curve with an unusually natural fit which, through experimentation [3], has been the most akin to that preferred by the human eye. In Akima's paper more information is provided on the derivation of the algorithm and its performance relative to alternate interpolation techniques.

2.3 Clustering Techniques

Although there exist many different methods in which to cluster data, only some are relevant to the requirements of the Plankton Prediction System. All of the applicable techniques can be broadly categorized within one of the following categories: *hierarchical, partition relocation* and *machine learning* [4,5]. All three categories have the same two core requirements. Firstly, data must be presented as a set of real vectors of arbitrary, but uniform, dimension. Secondly, there must exist a metric that quantifies the distance between any two vectors within the dataset. Although metrics such as the correlation and city block distance do exist, usually the standard Euclidean distance metric is used.

Hierarchical Clustering

In this method of clustering, *dendrograms* (cluster trees) [10] are constructed bottom-up or top-down through the successive agglomeration or division of data. Nodes (collections of data points) are split or merged according to their linkage type and proximity. The resulting cluster tree may be traversed and split from the top down until the desired number of clusters is achieved.

Hierarchical clustering algorithms have the primary advantage of speed over their counterparts. Unfortunately, however, once a node split or merge has been performed the nodes in mention are never revisited. Although the best option at the time, the split or merge may, from a macroscopic viewpoint, have been the wrong choice to have made.

Partition Relocation Clustering

In this form of clustering an initial cluster layout is either provided at random or derived from the results of a hierarchical clustering algorithm. Data points are iteratively swapped between clusters a fixed number of times or until the system converges to an acceptable error.

The *Expectation Maximization (EM) algorithm* is a concept used by a large number of *probabilistic* clustering techniques that fall under the category of partition relocation. The *k-means, k-medians* and *k-medoids* algorithms [2] are trivial examples of the *EM algorithm*. More complex examples involve the iterative refinement of a set of Gaussian probability distributions that define each cluster centroid.

Since partition relocation algorithms that are initialized randomly suffer the problem of converging at local minima, it is a common practice to rerun the algorithm more than once - taking the best clustering result as the final solution. Owing to the favourable compromise between result quality and execution speed partition relocation clustering is the most widely adopted clustering technique.

Machine-Learning Clustering

There are a large number of clustering algorithms that fall within this category. Although *genetic algorithms* and many other techniques are available, the most commonly applied machine-learning clustering technique to chlorophyll depth profiles is the *Self Organizing Maps (SOMS)* algorithm [8]. The SOMS algorithm reduces the dimensionality of all input vectors by replacing the actual vector with a set of parameters that describe a representative curve (usually a Gaussian). The data is structured on a SOMS grid based on its low-dimensional representative form. *Learning rate* and *neighbourhood* variables influence how drastically each element, and its nearest neighbours, change in value respectively after each iteration of the algorithm. These two variables are slowly decreased over time, so that the algorithm converges to a stable solution. After the algorithm halts, each element of the final SOMS grid and its affiliated data points, are treated as the cluster centroid and points respectively.

2.3.1 Previous Research

Two previous attempts have been made on clustering depth profile data. The first method approximates each depth profile with a four-parameter Gaussian function. The four parameters are used as the representative vectors in an implementation of the *Self Organizing Maps (SOMS)* clustering algorithm. The second approach, once again, fitted a Gaussian curve to each profile. However, in this approach the *Expectation Maximization (EM)* algorithm was used to cluster the data.

2.4 Dynamic Bayesian Networks (DBN)

Dynamic Bayesian Networks are an extension of Bayesian networks to handle temporal changes. They are directed acyclic graphs, with each node representing a discrete/continuous variable, and the edges connecting the nodes indicating causal relationships. [17], [18]

Each node has an associated conditional probability table, which gives the probability distribution of the possible values of the node, given the values of its parent nodes. A Bayesian network therefore consists of two parts: the values in each nodes CPT, which are known as parameters, and the structure, which are the actual nodes and arcs. In the case of a dynamic Bayesian network, the structure does not change over time. The system being modelled is dynamic, hence the name given to it. [17], [18]

A dynamic Bayesian network is defined to be a pair of Bayes nets, (B_1, $B_\rightarrow$), where the first member of the pair defines the prior $P(Z_1)$, and the second member is a two slice temporal Bayes net which defines $P(Z_t|Z_{t-1})$ by means of a DAG as follows:

$$P(Z_t \mid Z_{t-1}) = \prod_{i=1}^{N} P\left(Z_t^i \mid \mathrm{Pa}(Z_t^i)\right)$$

where Z_t^i is the *i*'th node at time *t*, and $\mathrm{Pa}(Z_t^i)$ are the parents of Z_t^i in the graph. The nodes in the first slice do not have any parameters associated with them, but each node in the second slice has an associated conditional probability distribution (CPD), which defines $P\left(Z_t^i \mid \mathrm{Pa}(Z_t^i)\right)$ for all t>1. The parameters of the CPD's are assumed to be unchanging over time. [17], [18]

To determine changing probabilities over time, the resulting DBN can be "unrolled" until we have the desired number of time slices, T. The resulting joint probability distribution is then given by:

$$P(Z_{1:T}) = \prod_{t=1}^{T}\prod_{i=1}^{N} P\left(Z_t^i \mid \mathrm{Pa}(Z_t^i)\right)$$

2.5 Learning in DBN's

2.5.1 Maximum Likelihood Estimation

Maximum likelihood estimation is used if the data being used for learning is complete, i.e. all the nodes in the network are observed, and none are hidden. This is not the case for our project, as we are attempting to infer chlorophyll profiles from satellite data. The data we have on profiles is fairly sparse, especially when linked with the satellite data. However, a form of MLE is used in the EM algorithm, so a brief explanation will be given here.

The goal of learning in this case is to maximize the likelihood estimates of the parameters of each nodes conditional probability distribution, i.e. the parameters of the CPT's which maximize the likelihood of the training data. For a detailed explanation, see Murphy [17], [18].

2.5.2 Expectation Maximization

The EM algorithm is used when the data is not complete, i.e. some of the nodes in the network are hidden. There are three steps involved in this algorithm: an E step, an M step, and calculation of the log-likelihood of the current step, to compare it with the previous step. [10], [11], [17], [18], [21]

The E (expectation) step is where the EM algorithm differs from the MLE algorithm. In this step, the expectation of the likelihood is calculated by including the hidden variables as if they were observed. An inference algorithm is used here to compute the unknown values from the observed values. [10], [11], [17], [18], [21]

For the M (maximization) step, the maximum likelihood estimates of the parameters are computed by attempting to maximize the expected likelihood found on the E step. [10], [11], [17], [18], [21]

The log-likelihood of the current parameter set is then compared with the log-likelihood of the previous parameter set, and if the change is beneath the specified threshold, the EM algorithm steps. Otherwise, the result of the M step is used in the next iteration of the algorithm to start the E step. [10], [11], [17], [18], [21]

The EM algorithm is thus merely an extension of the MLE algorithm, as the EM algorithm makes use of maximum likelihood estimates.

2.6 Inference in DBN's

2.6.1 Boyen-Koller

The idea behind the Boyen-Koller algorithm works as follows: if the interface cliques are too large, approximate the joint probability distributions on the interface as the product of marginals of smaller clusters *c* which partition the interface. The junction tree for a 1½ slice DBN is then created. This is a DBN which only maintains the nodes for the current time slice, and the nodes from the previous slice which had links to nodes in the current time slice. One step of exact Bayesian updating is then performed. The belief state is then approximated by a product of marginals, $P(I_t \mid y_{1:t}) \approx \prod_{c=1}^{C} P(I_t^c \mid y_{1:t})$, where $P(I_t^c \mid y_{1:t})$ is the distribution on nodes in cluster *c*. For a more detailed description of the algorithm, see Murphy [17].

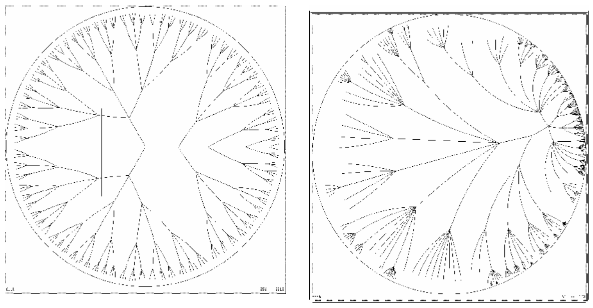


Figure 3: Hypergraph with uniform distribution and a distortion to the right

2.7 Knowledge Representation

Knowledge representation is a key issue within visualization. Topic Maps are a useful standard to describe knowledge structures and relate them to the underlying information resources. Topic Maps [20] conforms to the XML 1.0 standard and provides an inter-operable technology with existing tools. Topic Maps consist of three components, namely topics, associations and occurrences. Topics are subjects and are the nodes of a map. Associations show the relationships between topics, while occurrences are links to relevant information about each topic.

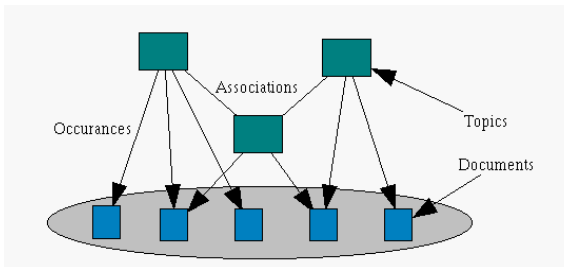


Figure 2: Topic Map highlighting the three components and its relation to the underlying structure.

Hypergraphs [13] are graphs whose edges connect two or more nodes. Topic Maps are natural extensions of Hypergraphs and hence the mathematical theory underlying

Hypergraphs, hyperbolic theory, can be applied to visualizing Topic Maps. When viewing graphs, the common problem of focus and context [15] must be taken into account. Known solutions are elision, in which unnecessary nodes are hidden, rapid zooming to view a small portion of the information, and distortion, in which the current point of focus is spatially expanded at the expense of the remaining nodes.

Temporal information models supply the viewer with a historical view of the developing information space. Currently a handful of research systems attempt to visualize this temporal view, and make use of anchors, colours and height proportional to the importance over time. These create a more liquid view than the common static view of information spaces.

DBNs can be represented by Topic Maps but equating network nodes to topics and edges to associations. DBNs are by nature temporal given the temporal relationships in the network, and hence these need to be considered when visualized. Currently no visualizing tool exists to dynamically view DBNs, however static views such as BayesiaLab [1] exist. The Hypergraph geometry allows for both elision and distortion and provides a dynamic viewing environment in which to study the DBN. The temporal relationships involved could be visualized using one or a combination of the techniques

Colour scales for displaying chlorophyll and sea surface temperature are almost universally drawn using 255 colours flowing from violet through blue into green and then to red.

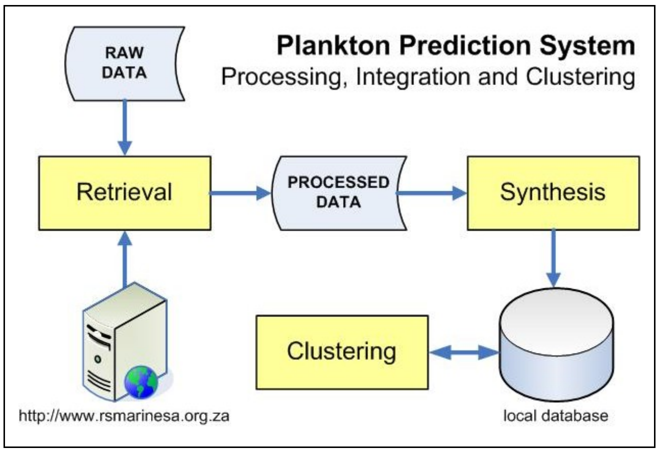

RAW DATA
Plankton Prediction System
Processing, Integration and Clustering
Retrieval
PROCESSED DATA
Synthesis
Clustering
http://www.rsmarinesa.org.za
local database

Figure 4: The preprocessing pipeline

3. DATA PREPROCESSING

For the purposes of modularity, the preprocessing stage is broken down into three phases of operation. The first phase, *retrieval*, downloads and prepares unprocessed data for use by the *synthesis* phase. The *synthesis* phase integrates data into a large database, discarding unnecessary information and quantizing values where necessary. Finally, the *clustering* stage discretizes all data, so that it may be efficiently processed by the Bayesian prediction engine. All three phases were implemented separately in C++, a language chosen for its flexibility and speed.

3.1 Retrieval

3.1.1 Target Data

Raw data originates from one of four collections. The first collection contains historic *sea surface temperature*, *sea surface chlorophyll* and *wind* data dating from the late 1986 to 2002. The second data collection comprises of all subsurface *chlorophyll* and *temperature* depth profiles, dating from 1988 to 2002. Both collections were sourced from the Zoology department. The third collection of data, sourced from [16], contains global bathymetric and topographic readings. The fourth and final data collection is a web repository that contains more recent *sea surface temperature* and *chlorophyll readings*. The goal of the retrieval process is traverse, decompress, add any missing metadata, and restructure all data by date.

3.1.2 Implementation

The retrieval program was written in C++ using the cURL libraries to provide support for downloading files from HTTP and FTP sources. A simple implementation of string pattern matching was implemented to extract date information from file names. Although an important stage in the preprocessing pipeline, not much time was allocated towards the *retrieval* phase, as its complexity was somewhat overshadowed by the *synthesis* and *clustering* phases.

3.1.3 Outcome and Validation

The result of the *retrieval* phase is a structured directory of outstanding data, ready for traversal by the *synthesis* phase. Where needed, the data is ordered by date and has all of its metadata intact. The successful execution of the next phase in the preprocessing pipeline, *synthesis*, validated the outcome of the *retrieval* phase.

3.2 Synthesis

3.2.1 Target Data

The *synthesis* phase operates on the data files created by the *retrieval* phase of the preprocessing pipeline. All time-dependant unprocessed data is stored in the ~/bin/DATA directory, while the relatively temporally static data, such as bathymetric and depth profile data is stored in the ~/bin/INC directory. Sea Surface Temperature and Chlorophyll satellite images are stored in the ESRI Band Interleaved by Line (BIL) file format, while wind data is presented in the Unidata Net Common Data Format. The bathymetric data is written in floating point (FLT) file format and both the subsurface chlorophyll and temperature profiles are presented in Microsoft Excel file format. Each format has a unique metadata structure associated with it.

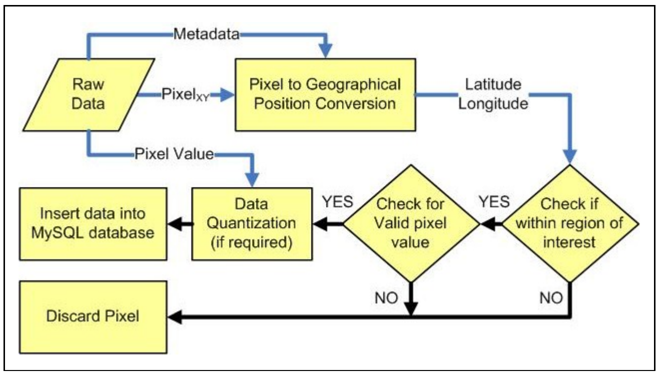

Metadata
Raw Data
Pixel$_{XY}$
Pixel to Geographical Position Conversion
Latitude Longitude
Pixel Value
Insert data into MySQL database
Data Quantization (if required)
YES
Check for Valid pixel value
YES
Check if within region of interest
NO
NO
Discard Pixel

Figure 5: The synthesis pipeline

3.2.2 Implementation

The majority of the *synthesis* phase's implementation is devoted to accessing data from the plethora of different various file formats. The *C++ Image class* used in the management of satellite images was a rewrite of the original class used in [9]. The *Network Common Data Format* libraries [22] and *CSV Parser* [6] classes were used extensively in interpreting the contents of the wind data files

All content in the file was reduced to the system's spatial and temporal granularity. For example, all dates were converted to a *weekly* representation, whilst location was converted to an offset from the top left pixel of a 650x650 raster of the geographical region 5S 5E to 40S 40E. Any data that was not considered valid (land, coast, indeterminate or outside the geographical region of interest) was discarded.

A large challenge of implementation was the efficient insertion of hundreds of millions of data points into the project database. Using C++ file I/O, temporary tables and the MySQL *LOAD DATA INFILE* command, insertion time was reduced by a factor of ten.

3.2.3 Outcome and Validation

The result of the synthesis phase is the population of a panoply of tables in the project database. Each table is carefully named, indexed and optimized for speed. The actual data contained within the tables is validated for accuracy using the visualization software described later in this paper.

3.3 Clustering

3.3.1 Target Data

In order to reduce the time and spatial requirement of the Dynamic Bayesian Network, the sea surface temperature (SST), sea surface chlorophyll (CHL), wind direction and speed (WND), bathymetric (DEP) and subsurface chlorophyll data (SND) is clustered. For the first four data types mentioned the clustering process identifies a set of contiguous intervals on the range 0 to 255, such that each individual data reading falls into a unique interval.

The extra dimension of information makes SND data slightly more complex to cluster, as the depth profiles are first resampled before being clustered. However, in order to resample the SND data, one first needs to interpolate a best-fit curve for the depth profile readings. *Akima spline interpolation* is used to approximate the depth profile curves to 100 meters below the ocean surface for all 2412 depth profiles. The curves are now resampled at a lower rate and the resultant vectors of lower dimensionality are passed to the clustering engine.

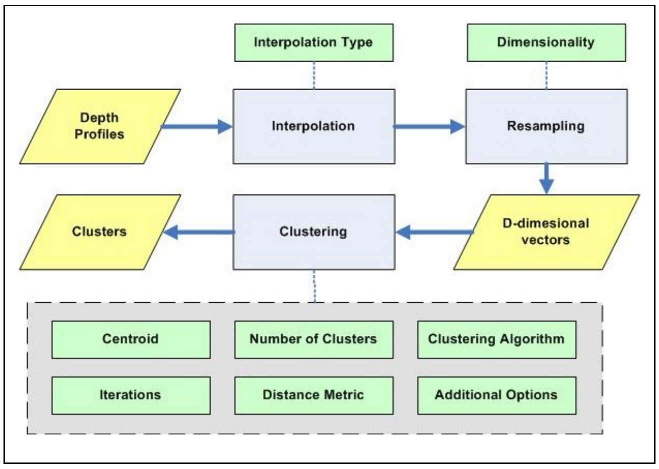


Figure 6: The clustering pipeline

3.3.2 Implementation

The limited success of previous depth profile clustering approaches prompted the need to explore alternate clustering techniques in this project. Therefore, the clustering application implemented in this project provides a means to partition data using one of the *k-means*, *k-medians* and *k-medoids* techniques. In addition, *Hierarchical*, *SOMS* and *hybrid*[1] clustering were implemented. A substantial portion of the code used to implement these algorithms was taken from the open-source *Cluster 3.0* program [7]. Figure 6 shows the flow of the clustering application.

3.3.3 Outcome and Validation

Generally, clustering introduces a degree of error to a given data set. The r^2 *goodness-of-fit* statistic is a suitable metric that describes how much error in a model is explained by natural variance between data. Thus, for each cluster, the closer the r^2 value is to one, the better the fit of the cluster to the data. An average r^2 value, weighted on the number of points contained in each cluster, may be used to quantify the *goodness-of-fit* of an entire clustering model

Through experimentation in conjunction with the *Weka* data mining package it was found that the *k-medians* clustering algorithm yielded the highest weighted r^2 value of 0.81. More importantly, it was interesting to observe how poorly the *hierarchical* and *SOMS* clustering performed in comparison to all the other algorithms tested. Unfortunately, it proved difficult to compare the results gleaned from the experiments with previous research, as the formulation of the r^2 statistic differed. However, a favourable reaction was received from the project supervisor and marine biologist, Prof. Field, on inspection of the final clustering results.

A final observation was made about the hierarchical clustering algorithm, which had an effect also on the hybrid clustering. It was noted that, due to statistical outliers, there were many leaf nodes at the top of the cluster tree. It turns out that this resulted in many singleton clusters being formed in the final cluster set. It was suggested that future work be conducted within the field of managing these statistical anomalies.

[1] *Hybrid clustering* is, essentially, a *k-means, k-medians* or *k-medoids* algorithm initialized with the results of a hierachical clustering in place of the usual random initialization.

4. DYNAMIC BAYESIAN PREDICTION

The language of choice for the implementation of this section was C++, partially due to the speed offered by it, and also because the library being used to create the network, perform learning, and perform inference, the Intel Probabilistic Network Library [14], was only available in C++. The OS being developed for was FreeBSD, which meant that no windows specific libraries could be used. This does have the benefit that the programs will work on both Windows and Unix platforms.

4.1 Model Structure

The dynamic Bayesian network had to incorporate the following variables: surface chlorophyll (Chl); sea surface temperature (SST); wind direction; wind speed; depth; region; and season. The links between nodes in time step *t* and time step *t*+1 also had to be specified. A basic example of this can be seen in figure 1, which illustrates which nodes in *t* link up with nodes in *t*+1, but with only Chl, SST, and Depth as variables.

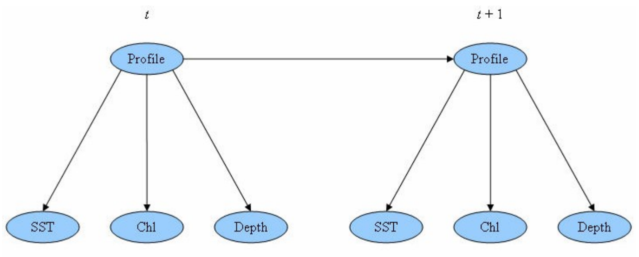


Figure 7: A basic example of a DBN

4.2 Learning

For learning to occur, data needs to be extracted from the database per location. It also has to be in discrete intervals. This is done by executing a complex query which selects the satellite data, joins it to interval tables for the relevant variables so that only an interval value is returned, and then returns all of these to the learning program. This allows the learning program to learn on all data for a particular location, for an entire year, or however long has been specified when executing. The learning program will do this for each point, or at least for each point which is not a land mass or covered by cloud. The learning algorithm used by the PNL [14] in this case is the Expectation Maximization algorithm, explained earlier.

Once all the data for the specified time period has been learned from, the learning program will write the learned conditional probability tables to the database for later use by the inference program.

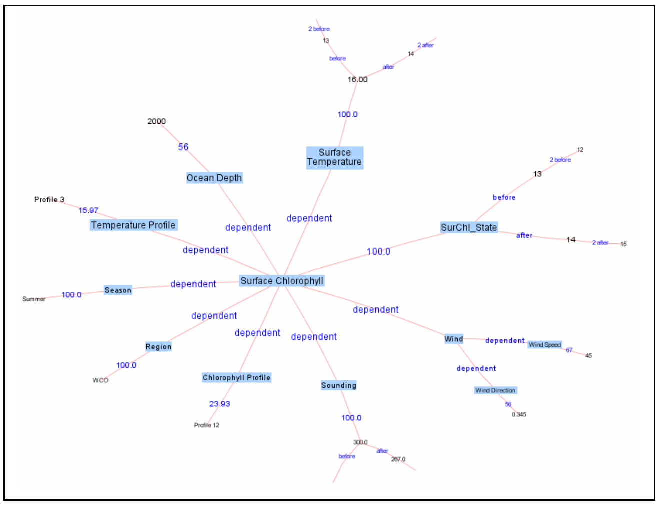


Figure 8: An example Hypergraph tree

4.3 Inference

Inference is done based on satellite data, which is stored in the database. Prior to performing inference on this data, the stored conditional probability tables must be retrieved. If they are not retrieved from the database, inference can still occur, however it will be less accurate, as it will essentially be learning the conditional probability tables for each node as it goes.

Once the tables have been retrieved, the inference program will extract an entire years worth of data for one location to perform inference on. The inference algorithm used by the PNL [14] is the Boyen-Koller inference algorithm, explained earlier. The data needs to be extracted for only one location at a time, as inference in a dynamic Bayesian network can only occur on a temporal sequence of data. Once the data for a location is retrieved, it is entered into the network as evidence.

The final step is to perform inference on the data entered as evidence, and to store the result of the inference back in the database. This is done for each week of data that was entered for a location.

5. VISUALISATION

5.1 Topic Maps

Topic Maps were written in XML and parsed using Java with the combination of Hypergraph libraries available from Sourceforge. The Topic Map structure was determined based on the structure of the DBN. Each temporal variable, such as season, wind and sub surface chlorophyll were identified and extra associations added to past and future nodes of the same type. These resembled prongs attached to each variable that users could zoom on to view the current variable within context.

User testing was conducted post implementation to test whether the Hypergraph's dynamic view is better than the BayesiaLab's static view. Participants were asked conceptual questions about DBNs and then asked to complete a series of tasks using each interface. To counteract the effects of learning, participants were split into groups and started the tests on opposing interfaces. Initial results of testing show no performance benefits to use a dynamic view; however participants chose the dynamic view in preference over the static view.

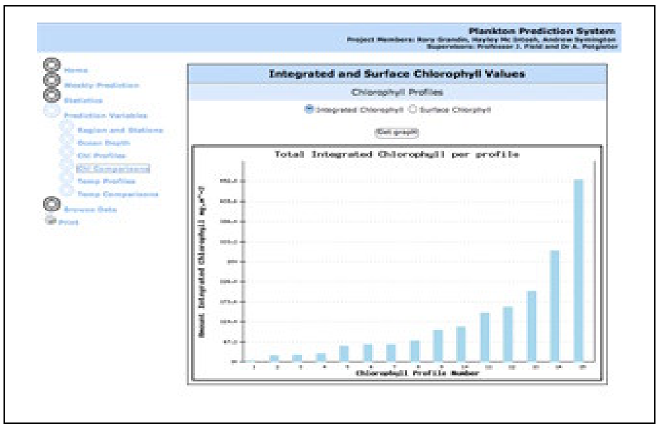


Figure 9: Screenshot of the PPS system

5.2 User Interface

Developing the user interface to the PPS was done by applying a user-centred design approach. User-centred design approaches and in particular the Star Model task-based and places evaluation as a key component. The Star Model involves the users and is applicable to rapid prototyping to create a flexible interface. Throughout the design phase numerous prototypes where developed to refine the functionality and usability of an interface.

The interface was developed in PHP, with GD and PHPlot, to create a platform independent, flexible interface. The web-based environment meant users did not have to install extra software on their computers before use and the users did not learn a new method to navigate through the functionality. User testing revealed that while marine biology students understood the results, general users need extra information to correctly interpret the results.

6. RESULTS

6.1 Dynamic Bayesian Prediction

Both learning and inference were found to be slowed down by having to retrieve data from the database for learning and prediction. This made testing for accuracy of predictions difficult, and suggests that to make the system more useable, data retrieval should be made more efficient.

7. CONCLUSION

Clustering of depth profiles using various algorithms provided a thorough exploration of the data and the resulting r^2 *goodness-of-fit* testing showed that k-medians provides the least error between the actual data and the representative clusters. At the time of writing, inference was completed for a region of the map and showed promising results. The visualisation of both the DBN and the interface provided a means to view inference results and the structure of the DBN. The user interface developed through sound software engineering principles was both functional and intuitive to use.

8. FUTURE WORK

8.1 Arbitrary Sizes of Satellite Images

At present the Dynamic Bayesian Network is only set up to take as input satellite images which are 650x650. These are stored in the database, with their pixel indices converted to a location representation, to reduce the amount of data stored in the database. A possible solution to this is to simply have a table which stores the dimensions of the satellite images. This would work because the calculation of the location of a pixel is entirely dependent on the dimensions of the satellite image, thus storing these values in the database would allow one to simply retrieve these dimensions from the database before calculating locations. This way, code would never have to be changed if the image size changed. The values in the table are the only things which would have to be changed.

8.2 Arbitrary Regions for Prediction

At present the regions which a pixel location can fall into are hard-coded. This prevents one from easily changing the regions, as the source code would need to be changed, and the binary recompiled. The regions are at the moment defined as square areas on the satellite images, except for one region which covers everything which is not the West Coast, West Agulhas Bank, or East Agulhas Bank. Because of this, one could simply store the region information in a table in the database, as two pairs of x, y values for each region. The source code would also have to be changed to take this into account. This is, however, a trivial matter, as it simply involves a query to the databases for the region information.

8.3 Arbitrary Seasons for Prediction

As the season is a variable used in prediction, it is necessary to know what season each week falls into. This is because the database holds satellite images stored by week. These images are then used as input in both the learning and the inference stages. So for each image, it is necessary to know what season it is associated with.

At present, the seasons are hard-coded. While this is fine for use in South Africa, and for places in the world which have the same seasonal patterns, this would be a problem if the system is used in other parts of the world with a different seasonal pattern. For instance, North America experiences winter from December to February, whereas Southern Africa’s winter period is in the middle of the year. Putting the seasonal information into the database would allow the system to be easily migrated to regions with different seasonal patterns. This is because we can simply define a season by the weeks into which it falls in the year. These ranges of weeks for each season can be stored in the database as a start week and an end week, which can be retrieved before attempting to determine what season a week falls into.

8.4 Primary Production

Primary production can be estimated from surface chlorophyll and the solar irradiance. Solar irradiance is calculated through the use of photosynthesis models of various complexities. Some models include the vertical distribution of phytoplankton, in our case, the clustered profiles. These profiles are important for time series estimation. A number of profiles from the preprocessing phase have subsurface peaks. These subsurface peaks are dependent on time and are an important indicator of the ecosystem. The primary production is also dependent on a light source, namely the sun. Primary production is measured in the same way as chlorophyll and can easily be added to the predictions with the implementation of a light approximating algorithm. Once implemented, summaries of these predictions could be generated, similar to the existing surface chlorophyll and integrated chlorophyll summaries and displayed in the statistics section.

ACKNOWLEDGEMENTS

Thank you to both the supervisors, Dr. Anet Potgieter and Prof. John Field for their guidance and support throughout the duration of the project. In addition, the PPS project team would also like to thank Prof. Frank Shillington and Christo Whittle from the UCT Oceanography department, as well as Norman Kuring from NASA, for their assistance in processing and visualizing satellite data.

REFERENCES

[1] "*BayesiaLab*". Available at: http://www.Bayesia.com. Accessed: 31 July 2006.

[2] Aggarwal, C. *et al.* *"Fast Algorithms for Projected Clustering"*. Philidelphia, PA. SIGMOD paper presentation. June, 1999.

[3] Akima, H. (1970) "A new method of Interpolation and Smooth Curve Fitting Based on Local procedures". Journal of the Association for Computing Machinery. Vol 17. No. 4, USA.

[4] Alsabti, K. et al. (2002) "An Efficient K-Means Clustering Algorithm". Information Technology Laboratory, Hitachi America Ltd, USA.Berkhin, P. (2002) "Survey of Clustering Data Mining Techniques". Accrue Software, USA.

[5] Berkhin, P. (2002) "Survey of Clustering Data Mining Techniques". Accrue Software, USA.

[6] Bose, M. (2006) *"CSV Parsing Class"*. http://www.mayukhbose.com/freebies/csvclass.tar.gz Accessed: 10 November 2006.

[7] de Hoon, M. (2006). "Software: Cluster 3.0". Available at: http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/software.htm Accessed: 10 November 2006.

[8] Demarcq, H., Richardson AJ. and Field, J. (2006). "*Estimating primary production of the Southern Benguela upwelling system from SeaWiFS and an archive of vertical chlorophyll profiles*". Marine Ecology Progress Series, submitted.

[9] Fenn, R. (2005) "*Predicting Plankton Profiles from Satellite Data"*. University of Cape Town Computer Science Department, Honours Project Report, South Africa (unpublished).

[10] Friedman, N., Murphy, K., and Russell, S. (1998). *"Learning the structure of dynamic probabilistic networks"*. Proc. Fourteenth Conference on Uncertainty in Arti. cial Intelligence *(UAI '98)*, 139–147.

[11] Ghahramani, Z. (1998). *"Learning Dynamic Bayesian Networks"*, Adaptive Processing of Sequences & Data Structures. Lecture Notes in AI, Springer, 168–197.

[12] Giada, L. and Marsili, M. (2002) "Algorithms of maximum likelihood clustering with applications". Max-Planck Institut für Kolloid-und Grenzflächenforschung, Germany.

[13] Hypergraphs. (2003). "*HyperGraphs*". Available at: http://hypergraph.sourceforge.net/index.html. Accessed: 10 June 2006.

[14] Intel PNL. (2005). *Intel PNL*. Available at: https://sourceforge.net/projects/openpnl/. Accessed: 12 October 2006.

[15] Lamping J. , Rao R. and Pirolli P. (1995). "*A focus+context technique based on hyperbolic geometry for visualizing large hierarchies*". Proceedings of the SIGCHI conference on Human factors in computing systems. p401-408.

[16] Moore, CJ. (2006). "National Geophysical Data Center (NGDC)". Available at: http://www.ndgc.noaa.gov/mgg/fliers/01mmg04.html. Accessed: 15 October 2006.

[17] Murphy, K., (2002), *"Dynamic Bayesian Networks: Representation, Inference and Learning"*, PhD Thesis, UC Berkeley. Available at: http://www.cs.ubc.ca/~murphyk/Thesis/thesis.pdf . Accessed: 14 May 2006.

[18] Murphy, K., (2002), *"Dynamic Bayesian Networks"*. [Online]. Available at: http://www.cs.ubc.ca/~murphyk/Papers/dbnchapter.pdf . Accessed 20 May 2006.

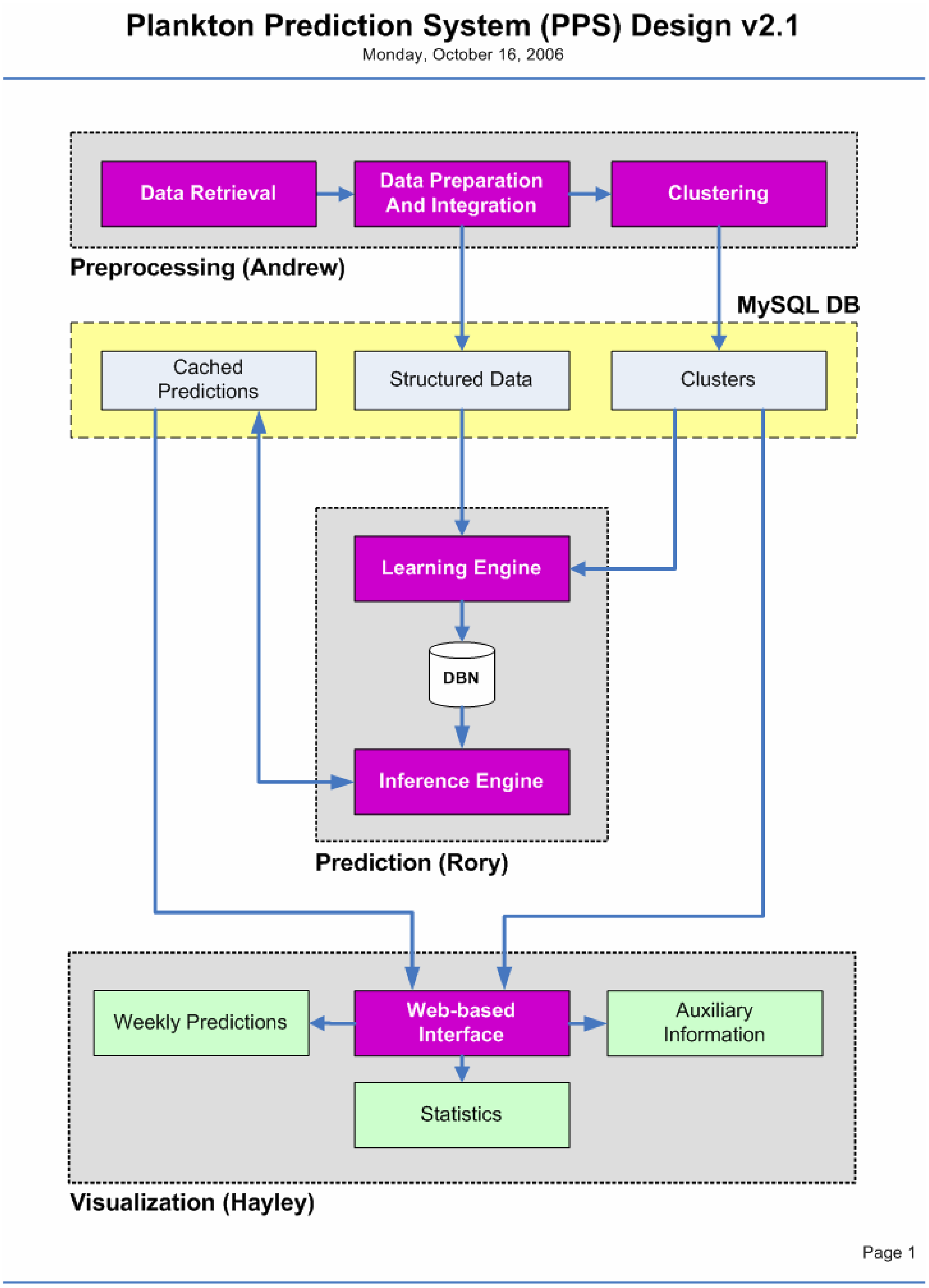
[19] No author given. (2006) "*Weka 3 – Data Mining Software in java"*. The University of Waikato, New Zealand. Available online: www.cs.waikato.ac.nz/ml/weka/. Accessed 10 November 2006

[20] TopicMaps.Org XTM Authoring Group. (2001). "*XTM: XML Topic Maps (XTM) 1.0: TopicMaps.Org Specification*". Available at: http://www.topicmaps.org/xtm/1.0/. Accessed: 7 May 2006.

[21] Tucker, A. & Liu, X., (2003*). "Learning Dynamic Bayesian Networks from Multivariate Time Series with Changing Dependencies"*. In Proc. 5th Intelligent Data Analysis Conference (IDA 2003), 100–110. Springer-Verlag, LNCS 2810.

[22] Unidata. *"Network Common Data Format"*. Available at: : http://www.unidata.ucar.edu/software/netcdf/. Accessed: 10 November 2006

APPENDIX A


Plankton Prediction System (PPS) Design v2.1
Monday, October 16, 2006
Data Retrieval
Data Preparation And Integration
Clustering
Preprocessing (Andrew)
MySQL DB
Cached Predictions
Structured Data
Clusters
Learning Engine
DBN
Inference Engine
Prediction (Rory)
Weekly Predictions
Web-based Interface
Auxiliary Information
Statistics
Visualization (Hayley)
Page 1