A Randomized Framework for Discovery of Heterogeneous Mixtures

Mark A. Livingston, Aditya M. Palepu, Jonathan Decker, and Mikel Dermer

Naval Research Laboratory, 4555 Overlook Ave, Washington, DC, USA 20375

ABSTRACT

Mixture models are the term given to models that consist of a combination of independent functions creating the distribution of points within a set. We present a framework for automatically discovering and evaluating candidate models within unstructured data. Our abstraction of models enables us to seamlessly consider different types of functions as equally possible candidates. Our framework does not require an estimate of the number of underlying models, allows points to be probabilistically classified into multiple models or identified as outliers, and includes a few parameters that an analyst (not typically an expert in statistical methods) may use to adjust the output of the algorithm. We give results from our framework with synthetic data and classic data.

Keywords: Data mining, unsupervised machine learning, statistical classification, exploratory data analysis

1. INTRODUCTION

The goal of taking unstructured data as input and grouping the points according to candidate underlying models has a long history in many fields. Data mining[1] uses machine learning techniques to discover models that describe observed data. Of particular interest in visual analytics is the discovery of potential models that may describe the underlying processes that generated the data. Typically, these mechanisms are not directly observable, but may be inferred from measurements. Extracted models lead to insights about the events encapsulated in data.

Many issues must be resolved to extract models from unstructured data. Algorithms may need to know the types and number of models present or permitted to be found. Most algorithms classify points on a binary basis, rather than allow for multiple possible models to be associated with a single point. Many do not enable fundamentally different types of functions to be compared for their applicability to the data. Algorithms must deal with noisy points that may not fit any model very well. The data in Figure 1 lends itself to an interpretation of a set of clusters, a potential 1D grouping, points which could belong to multiple models, and probable outliers. Our primary goal was to design an algorithm that would automatically find candidate models so that an analyst could determine whether they were appropriate hypotheses for a given application and data set.

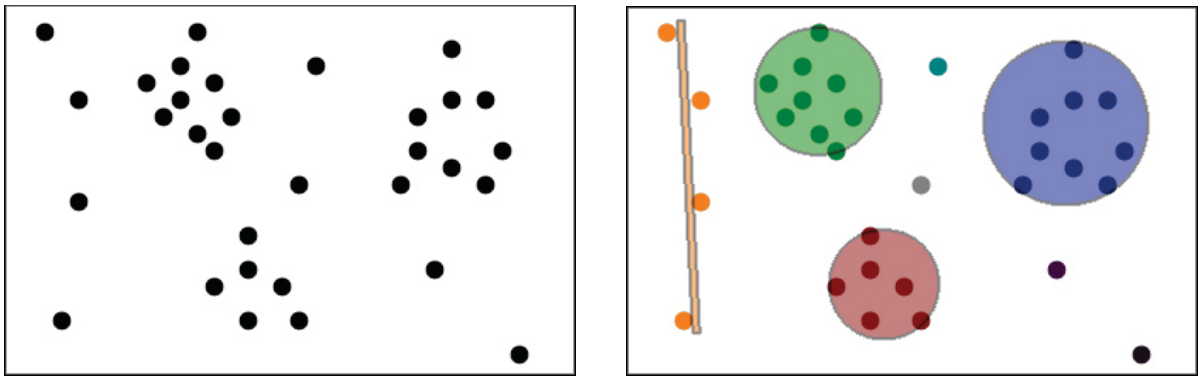

Figure 1. An example of unstructured data with a manually-specified mixture model illustrates the goals of our algorithm. The right-hand plot shows how one could discern three potential clusters, a linear relationship at left, some points between models, and some outlier points that do not seem to fit any of the component models.

Corresponding author: mark.livingston@nrl.navy.mil; phone 202 767-0380; fax 202 404-1192

Visualization and Data Analysis 2011, edited by Pak Chung Wong, Jinah Park, Ming C. Hao, Chaomei Chen, Katy Börner, David L. Kao, Jonathan C. Roberts, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7868, 78680A · © 2011 SPIE-IS&T · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.872660

Our goals in designing an algorithm included avoiding or solving all of these issues. We wanted the algorithm to assess the fit of a variety of model types to the data, which required that we find a fair method of comparison between different functions. This in turn required that we design an abstraction of each function type flexible enough to operate on the data without requiring special functions, yet rigidly conform to the interface to which all other functions would adhere. Additionally, we desired a simple interface that would make it easy to add new functions to the library. Because we want to develop a tool that an analyst (generally not an expert in statistical techniques) may use to generate hypotheses about the data, we want to provide an intuitive set of controls for controlling the generation of the hypothesized models. Finally, we did not want our algorithm to require any other initial data beyond the unstructured input data; no initial estimates or cues as to the number of underlying functions are required. Although it has not been a concern yet, we further believe that our algorithm will lend itself to computational efficiencies in its implementation.

The following section provides an overview of related work from many fields which relate to the problem. Section 3 introduces our algorithmic framework and gives some details of our current implementation. Section 4 presents some preliminary results with a variety of data sets. We conclude with discussion and plans for improvement in Section 5.

2. RELATED WORK

Mixture models have been applied in a number of fields. A full survey of related methods is beyond the scope of this paper; we highlight a few methods from machine learning.[2] Most often, mixture models consist of Gaussian functions with independent mean and variance parameters. There are many classes of algorithms that have been applied.[3] The conventional approaches assign each observation to one source function; our algorithm is an example of what is often called a fuzzy algorithm, in that it assigns each data point to a group with a probability that may be between 0 and 1. To determine the parameters of each component function, most approaches employ expectation-maximization (EM). This is a very general strategy, but is subject to getting stuck in a local optimum and requires a good initial estimate (including prior probabilities) in order to find the correct global optimum. A particular formulation from which we draw inspiration is that of unsupervised learning, which is done without knowledge of the classes into which the data points are to be grouped. However, many algorithms assume that the number of classes is known, an assumption we prefer to avoid. It is also common to assume that a good initial estimate is known, another assumption we prefer to avoid, at least as an absolute requirement.

Projection pursuit[4,5] uses linear subspaces of dimension $n-1$ in an n-dimensional domain to find a basis for grouping data into clusters. A projection index measures the spread of the data and the tightness of clusters to evaluate the projection axis. Such an approach often requires a reconstruction onto the original domain for comprehension. Greedy basis pursuit[6] extends this approach by selecting new basis functions with a criterion for discarding functions on which more recent selections have improved. This approach requires having a dictionary of candidates bases to approximate the underlying function. FlexMix[7] allows finite mixtures of linear regression models (Gaussian and exponential distributions) and an extendable infrastructure. However, to reduce the parameter space, the package restricts some parameters from varying or restricts the variance. It assumes the number of component models is known, though it allows for component removal for vanishing probabilities, which reduces the problems caused by overfitting. Jain et al.[8] provide two methods for unsupervised learning of Gaussian clusters: one a "decorrelated k-means" algorithm that minimizes an objective function of error and decorrelation for a fixed number of clusters, and the second a "sum of parts" algorithm that uses EM not only to learn the parameters of a mixture of Gaussians but also to factor them.

One could also conceive of the problem as a statistical classification problem.[2] This often requires an a priori labeling of a training set, or at least a few labeled data points with which other points may be grouped. A number of algorithms or conceptualizations have been used to separate or group points, such as linear and quadratic classifiers, support vector machines, and decision trees.[1] Simple approaches such as a naïve Bayes classifier often work quite well, but assume that the attributes are independent, an assumption which is often violated in real-world data sets (including those of interest in our application). Choosing an independent subset of variables is also not a trivial matter. Linear regression attempts to fit the data to a hyperplane and works surprisingly well in many applications. Extensions to higher-order functions also can work well (though overfitting becomes an issue). However, there are few algorithms that enable linear and non-linear models to compete on equal footing

against each other for being the "best" fit for a set of data points. Clustering using level-set methods[9] solves some of these issues.

Many approaches to discovering general mixture models in theoretical computer science have also restricted components to be Gaussians or other concave (or log-concave) functions; thus outliers do not pose a significant problem in the analysis of the algorithm. Much research has focused on finding minimal bounds for the separation required between such distributions.[10] Some approaches do not require separation, but do require independence between the distributions.[11] Most fix the (maximum) number of underlying models (though removal of models may occur) and require that all points be classified into exactly one distribution. Spectral projection[12] has been applied to reduce the search space and find subspaces in which the points from one model are separated from those of another model. Determining the right projection may require a training set.[13] Computational inefficiencies or difficulties, such as discretization of the search space for initial estimates or assumptions regarding correlations or variances are a concern, even in the theoretical literature. Ease of implementation and applicability to real data sets (where the assumptions on distributions are violated) are noted.[12]

Another issue of great theoretical importance is the notion of identifiability of a mixture – i.e. whether a mixture has a unique composition.[14] For practical purposes, we assume that in real data sets, there are indeed multiple possible solutions. Since we are interested in generating hypotheses that a human analyst will evaluate, we do not concern ourselves greatly with the uniqueness of generated models (only enough to lower the computational effort expended) and instead focus on developing an algorithm that will propose multiple, perhaps overlapping or competing, patterns within the data. Another practical concern is that of overfitting many parameters, which can lead to models of too high a degree to be preferred in the presence of noise or errors in a training set. For unsupervised learning such as our algorithm, an analogous concern is the closeness of fit of functions with different degrees, and whether they can be compared with each other on an equal basis.

3. PROPOSED ALGORITHM

We begin our description of the proposed algorithm with the abstraction of a model, including metrics by which models may be compared to each other. This gives rise to a simple pseudocode for the algorithm, as well as a refined design that requires improved metrics on a model for implementation. We then describe two example models we employ in our current implementation.

3.1 Model Abstraction

We took an object-oriented approach to the design of a model. The fundamental operations we identified for a model were the construction of a model from a list of points and the computation of a residual error associated for a point, given the current model parameters.

Construction of a model has two modes; however, a single routine should easily handle both modes. In the first mode, we build an instance of a model from the minimum number of points required to specify it. In the second mode, we build a least-squares approximation from an arbitrary number of points (greater than this minimum). As is well-known, a least-squares approximation should equal an exact solution when the number of points is precisely that which is required. Thus this function requires one routine and that the module knows how many points are required at a minimum. It implicitly assumes that an instance of a model has a method of identifying points from which it is to build an approximation (or exact model). Here these same two cases differ slightly. In the first mode, a random selection will suffice; why this is so will become clear in describing the algorithm in the next subsection. In the second mode, points are selected on the basis of their residual error from a previously-constructed model.

Thus we come to the computation of residual error for a point. This will naturally have a slightly different meaning for different functions, but for many functions is merely an abstraction of a distance function with geometric meaning in the space. We use this for our linear and elliptical cluster models. (In a previous implementation, we used Gaussians–a frequent component of mixture models–with the natural residual associated with such a function, the Z-score.) Implicitly, there is a metric for whether a particular value of the residual is deemed sufficiently close to an instance to warrant inclusion of the point as a supporting data point. Thus every function must have a way, given access to the input data, to build a random instance of itself, to compute

residuals to its current instance, to evaluate which points are worthy of further consideration, and to build a least-squares approximation from those points.

3.2 Algorithm

Our algorithm relies on the RANSAC[15] framework for estimation of good candidate models. However, we alter the algorithm in a key (but simple) way. We do not seek only one "best" answer (by any metric); we seek to generate reasonable models that are candidate hypotheses for the underlying cause(s) behind the input data. Thus we maintain an internal list of candidates (via a priority queue) rather than a single candidate. We can then iteratively generate models at random, assess their quality, and compare them against other models previously generated. The lowest quality candidate among the current set is discarded when a better candidate is discovered.

We make an analogous change in the stopping conditions for RANSAC. The original specification included two criteria: a maximum number of attempts to find the best model and a minimum amount of support required. We can maintain the first criterion directly: we limit the number of models tested. The second criterion requires some adjustment, since we allow that multiple underlying models may be responsible for the data. Thus we count the number of points that are supporting at least one candidate model to determine whether a sufficient portion of the data has been modeled. It is important to note that we do not prevent a point from supporting two completely independent models. This requires some global accounting to ensure that points are not counted multiple times when determining how many points in the data are modeled by the candidate functions.

Algorithm: Hypothesis (Model) Identification

1. support $\leftarrow 0$
2. candidateList $\leftarrow \emptyset$
3. *repeat*
   (a) newCandidate $\leftarrow$ *generate random*
   (b) compute all points' residuals for newCandidate
   (c) form best-fit (least-squares) model from support set
   (d) re-compute all points' residuals
   (e) compare newCandidate's quality to candidateList
   (f) if a minimum threshold of the number of models has been met
   discard lowest quality among { candidateList $\cup$ newCandidate }
   else
   add newCandidate to candidateList
   (g) update count of supported points (amongst all models)
4. *until* minimum support attained or maximum models tried

The remaining question to complete the implementation is the comparison of the quality of one model to another. In order to compare models of different types, we devised a metric based on the density (in whatever number of dimensions the model is embedded). This concept applies equally well to area-based measures in 2D as to volume-based measures in 3D, hyper-volume-based measures in 4D, and so on. The only issue is the correct computations for the number of dimensions. Since our goal is to describe as much of the input data as possible, we incorporate into our comparison metric the number of supporting points each model has. Small differences in the number are supporting points (currently less than $\pm 25\%$) are outweighed by large (greater than $\pm 25\%$) differences in the density metric, whereas density becomes the consideration when models have nearly the same level of support.

3.3 Implemented Models

As noted above, we have implemented two types of models: lines and clusters. Our motivation for these two models came from the data on which we ultimately hope to deploy our algorithm: GIS data. One specific problem of interest is the clustering of criminal activity. If we wish to compare whether particular streets or particular neighborhoods are more susceptible, then we need models which can accurately characterize both linear and area features in 2D. Extensions to 3D will enable the algorithm to apply to more general gridded data, and the reader will note that there is little adaptation required in order to apply all the modeling discussed below in any number of dimensions.

3.3.1 Linear Models

Lines represent the simplest function, requiring just two points to build. Thus the initial construction picks two points at random and computes the slope and intercept. Residuals may then be computed as the distance from this initial line. Currently, we consider points that are within a distance that represents 10% of the range of the data in any dimension to be support for that line. A standard least-squares approximation[16] may be computed from the identified points using orthogonal regression.

3.3.2 Elliptical Models

We fit an elliptical model to detect clusters in the data. Thus in 2D, this model is an ellipse; in 3D, an ellipsoid; in 4D, a hyper-ellipsoid, and so on, for any number of dimensions. From the initial (random) selection of points (one more than the number of dimensions in order to be fully specified, with a check that the points have variance in all the dimensions), we compute the sample mean and variance. Supporting points within the elliptical shape are detected, and a new mean and variance computed to arrive at the final model.

3.3.3 Final Mixture Model

The final mixture model should also have a residual for each point against each of the component models. This serves two purposes. First, it enables fuzzy assignment of every point in the original data to each of the models in the final candidate set. This allows a point to support multiple models and demonstrate that either hypothesis might explain a particular data point. Second, it enables us to identify potential outlier points that are not explained by any of the candidate models.

3.4 Control Parameters

With the algorithm thus specified, we can quickly see the parameters through which a user may control the performance of the algorithm. We typically enforce a minimum amount of support for a model to be considered valid at all. This serves as a proxy for an analyst to request that more or fewer candidate models be proposed. A large number prevents the algorithm from generating an extremely high number of models and increases the likelihood that the models generated will be meaningful. However, raising this number too high can cause the algorithm to miss a valid model that explains a small number of points. (An adaptive metric could be considered, based on the current number of unlabeled points.)

We also inspect models for similarity to each other. This may be performed by inspecting the overlap between the supporting point sets, which is portable across models embedded in different dimensions and of different types. We have considered whether this inspection should be done as soon as a new candidate model is proposed, which may help avoid unnecessary computations that will merely explain an already-captured portion of the data. Currently, this merely helps rank candidate models during the search process.

The user may set a threshold to determine how well a point should fit a model in order to be considered to support it. Again, this is an intuitive control parameter in that it directly controls the assignment of points to models. But it not entirely intuitive, in that it also indirectly controls the shape of models. Since this parameter will change the number of points found to fit a proposed model, it will determine from which points a least-squares model is computed.

In our adapted RANSAC implementation, we may specify the total amount of the input data that must support our final (mixed) model in order to stop the algorithm. The user may also set a computation time or number of models to attempt.

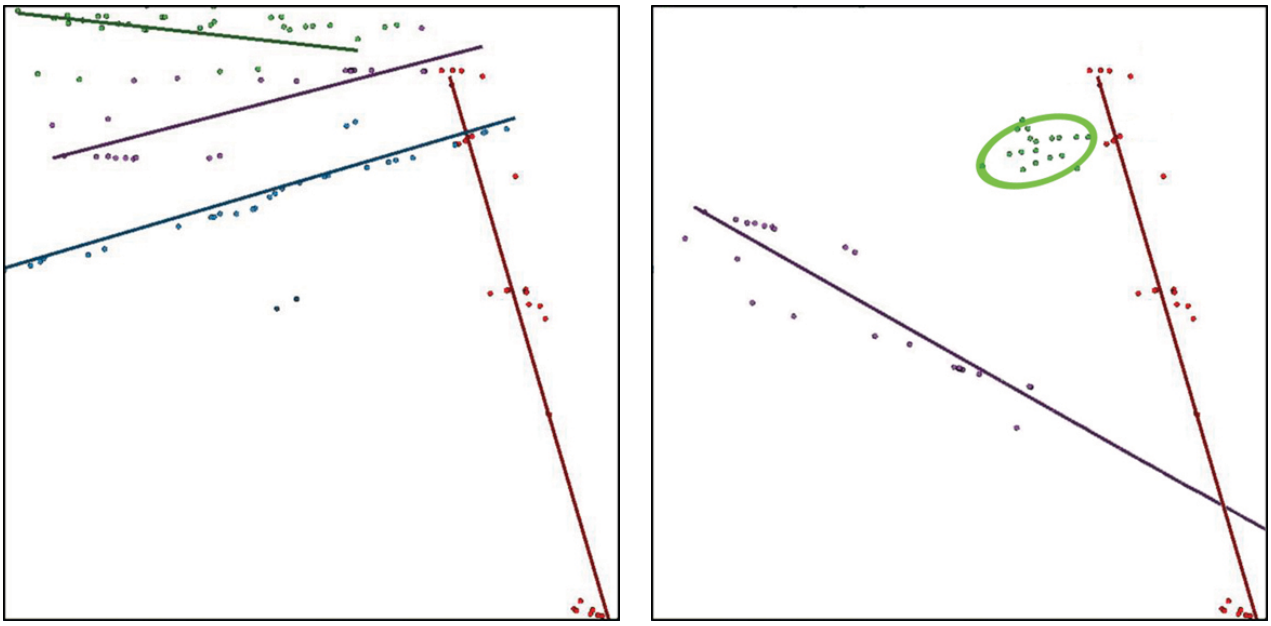

Figure 2. Two synthetic data sets present several challenging cases. (Left) intersections, near intersections, uneven distribution, and nearly equidistant models. (Right) a mixture of lines with an elliptical cluster.

4. RESULTS

We conducted preliminary tests on our algorithm in two ways. We began with synthetically generated data. Although we knew the ground truth for this data, it is not clear that even the generation of points from a specific model automatically implies that it should be classified with that model. It could easily be that a point will be generated by one function but closer to another on output. Since we seek only to generate hypotheses for a human analyst, we present qualitative rather than quantitative results. These are much more meaningful in our application. Similarly, with classic data sets for similar algorithms, we can perform quantitative analysis, but prefer qualitative results which could steer an analyst for where to look into the details.

4.1 Synthetic Data

To test our algorithm's robustness to complex patterns of lines and clusters, we generated two synthetic data sets. The first set (Figure 2, left) consisted of four line models, with points scattered along the lines. Note that one of the models (extending from the lower-right corner) has a very wide and uneven distribution along the extent of the line. Also, this first line crosses another at approximately a 90° angle. In two other places, the endpoint of one models comes close to intersecting another model. Our algorithm finds the four lines and groups the points accordingly. Note that near the intersection, some of the points generated along the more horizontal line were labeled as more likely part of the more vertical line. Given the models found, there is no guarantee that this is incorrect. The probabilities for each model are nearly 50%. This also occurs in the region between the two top-most line models, where the probabilities for some points are again nearly equal between the two models, although lower than other points for each model due to the greater distance from the model.

The second synthetic data set featured a cluster and two lines (Figure 2, right); one of the lines is nearly identical to the near-vertical line from the previous example, since the points that were used to compute it were shared in the input data for the two examples. Again, our algorithm determines a reasonable classification of the points into three distinct groups. In this data, the intersection of the two lines poses no confusion, since no points are near the point of intersection. Note that the points in the bottom right are still much closer to the more vertical line than to the more horizontal line. Thus, despite the separation from the other points assigned to the more vertical line, they are assigned to the same group. Similarly to the case of intersecting lines, the separation between the elliptical cluster and the near-vertical line presents some ambiguity. Few points are extremely close to the border determined, however, and thus the separation appears quite natural.

4.2 Fisher Iris Data

One of the classic data sets in pattern classification is the iris flower data set.[17] The data consists of four variables (sepal length, sepal width, petal length, and petal width) for three classes of iris: setosa, virginica, and versicolor, with 50 instances of each class. This data set has become a standard test case for linear discriminant analysis and other classification algorithms; thus, we wanted to run our algorithm on it. The comparison is not direct, but it gives us some insight into the performance with real-world data.

The top of Figure 3 shows some early candidate models that were generated from the iris data overlaid on the lower triangle of the scatterplot matrix for the data. Note that the ellipse exists in two dimensions (sepal length and petal length); it is thus shown in only one cell of the scatterplot matrix. On the other hand, the line model exists in three dimensions (sepal length, sepal width, and petal width). While it can be hard to tell from these plots exactly which points support a model in all dimensions in which that model exists, this figure begins to demonstrate the ability of our algorithm to select dimensions simultaneously while searching for candidate models within the data.

As further candidates are generated, the list of candidates is analyzed by the density measures described above. An advantage of our algorithm is the ability to compare models that are embedded in different dimensions through this density measure. As the process continues (Figure 3, bottom), the line model disappears in favor of other models. A new ellipsoid model appears in three dimensions (sepal length, petal length, and petal width), which garners support in the virginica and versicolor classes. A line has emerged in the dimensions of sepal width and petal width that attempts to model the setosa class.

Figure 4 shows the final candidate models. Not surprisingly, the separation of the two classes clustered together (virginica and versicolor) is not captured as well as the isolation of the setosa class. It is interesting to note that the three best models were all ellipses; all lines models and all models of greater than two dimensions were judged not to fit the data as well. While there is no guarantee that this will be the result even on another test run of the program, this observation should at least help us understand the figure. The three highest-ranking models (solid lines) also don't happen to exist in the same pairs of dimensions. The setosa class was modeled by an ellipse in the sepal length and petal width dimensions. The versicolor class (green dots – if reading a color version of the manuscript, light-grey dots if a grayscale verion, near the middle of the set) is modeled by a narrow ellipse in the petal length and petal width dimensions. Finally, the virginica class is modeled (with less accuracy than the other two groups) by an ellipse in the sepal length and petal length dimensions. A 3D line model generated late in the process but ultimately discarded is shown with a dashed line in the cells for sepal width, petal length, and petal width.

5. CONCLUSION

Our method uses an abstraction of the model that includes a generating function (which underlies all the computations), a notion of the residual error for a point, and a measure of the quality of the support for a model among the input data. This enables direct comparisons of qualitatively different models in different dimensions and of different parameterizations of a single type of model. Thus there are no assumptions made about the number or shape of the models that should be found in the data set. Previous methods have some number of the following limitations.

1. Many methods must be told the number of underlying models. If this number is incorrect, the computations are guaranteed to come up with inaccurate assessment of the data.
2. Points must be assigned to a single model. This limits the ability of an algorithm to handle noise or properly classify points that are ambiguous in their location, especially when models have overlap.
3. Many algorithms operate with only a single type of model. Typically, the assumption is that the individual models fit a normal (Gaussian) distribution. This limits the ability of the algorithm to accurately classify points that come from other generating functions, such as linear relationships.
4. Methods work in defined dimensions, unable to compare models embedded in different numbers of dimensions.

5. The algorithm requires training data in order to learn what models may exist and their patterns.

6. Interaction and control of the algorithms is either extremely rigid or through algorithmic details known to the programmer, not the expert analyst looking at the data.

7. The algorithm requires an initial estimate of the correct solution.

While many algorithms solve one or some of these problems, we believe that our algorithm is among the first that solves all of these simultaneously. Of these, one may correctly point out that it can be a great advantage to an approach to begin with an initial estimate. This is true; it may save computational expense and provide a better solution when an expert (through human or automated analysis) can give hints. However, we believe that there is benefit in being able to propose models that surprise the expert analyst. Thus we believe there is benefit in not *requiring* an initial estimate. We plan to add an ability to receive an initial estimate as input either before or during the mixture model discovery process in future work.

We require a user-settable threshold for the amount of support there should be for a model in the input data. This parameter establishes an intuitive control for the algorithm. Similarly, we allow the user to specify the amount of overlap between a new candidate grouping and the previously-accepted set of candidates in order to be kept as a hypothesis. These two controls enable an analyst to indirectly control the number of hypotheses generated and the distribution of them within the domain. Controls that derive directly from our use of the RANSAC framework include the total amount of points that must support at least one hypothesis and the number of hypotheses generated (equivalently, computation time allowed). Again, these are intuitive parameters. The final control parameter is the maximum permitted residual for a point to be considered in support of a model.

There are many improvements and further tests we hope to perform on our algorithm as implemented thus far. Our model abstraction enables our algorithm to run with a simple control structure that incorporates any number of dimensions and (in theory) any number of model types. Of course, we would like to add more types of functions and improvement the memory and computational efficiency of our implementation. We are investigating geometric and numerical alternatives to the support and residual metrics we currently use; cluster intensity functions seem to be a promising alternative.[9] We enable the user to specify dimensions to ignore; we currently use this to restrict the algorithm to operating on *numerical* dimensions only. While any number of dimensions is handled by the currently implementation, a good interface for an analyst to decide the number of dimensions that would be appropriate is one possible avenue for extension. An automated method would be also of interest. Adding the ability to handle non-numeric data types would be of interest, but the distance function computations would need to be considered carefully in such a case.

We hope to perform additional tests with our algorithm as well. While quantitative results from our algorithm are of some interest for comparison against similar algorithms, we feel that the benefit of our algorithm is in the combination of features it provides and the qualitative value it can provide to an expert analyst. Thus we have eschewed numerical tables of results in favor of visual presentations similar to what we imagine an analyst would want to see in an overview of data and hypotheses. When coupled with a visual analytics interface that enables the analyst to steer the algorithm in the right direction – without requiring an initial estimate or determining the shape of the solution – we believe our algorithm will assist in the generation of hypotheses about the data and provide measures that help an analyst compare them to each other. This would make our algorithm a valuable tool in an analyst's toolbox.

REFERENCES

[1] Witten, I. H. and Frank, E., [*Data Mining: Practical Machine Learning Tools and Techniques*], Elsevier/Morgan Kaufmann Publishers, 2nd ed. (2005).

[2] Duda, R. O., Hart, P. E., and Stork, D. G., [*Pattern Classification*], John Wiley and Sons (2001).

[3] Theodoridis, S. and Koutroumbas, K., [*Pattern Recognition*], Academic Press, 2nd ed. (2003).

[4] Friedman, J. H. and Tukey, J. W., "A projection pursuit algorithm for exploratory data analysis," *IEEE Transactions on Computers* **C-23**, 881–890 (Sept. 1974).

[5] Friedman, J. H. and Stuetzle, W., "Projection pursuit regression," *Journal of the American Statistical Association* **76**, 817–823 (Dec. 1981).
[6] Huggins, P. S. and Zucker, S. W., "Greedy basis pursuit," Tech. Rep. Technical Report TR-1359, Yale Univ. Dept. of Computer Science (June 2006).
[7] Grün, B. and Leisch, F., "Fitting finite mixtures of linear regression models with varying & fixed effects in R*," in [*Compstat 2006 (Proceedings in Computational Statistics)*], 853–860 (2006).
[8] Jain, P., Meka, R., and Dhillon, I. S., "Simultaneous unsupervised learning of disparate clusterings," *Statistical Analysis and Data Mining* **1**, 195–210 (Nov. 2008).
[9] Yip, A. M., Ding, C., and Chan, T. F., "Dynamic cluster formation using level set methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**, 877–889 (June 2006).
[10] Dasgupta, A., Hopcroft, J., Kleinberg, J., and Sandler, M., "On learning mixtures of heavy-tailed distributions," in [46^{th} *Annual Symposium on Foundations of Computer Science*], (2005).
[11] Feldman, J., O'Donnell, R., and Servedio, R. A., "Learning mixtures of product distributions over discrete domains," in [46^{th} *Annual Symposium on Foundations of Computer Science*], (2005).
[12] Kannan, R., Salmasian, H., and Vempala, S., "The spectral method for general mixture models," in [*Conference on Learning Theory (Lecture Notes in Artifical Intelligence v.3559)*], 444–457 (June 2005).
[13] Achlioptas, D. and McSherry, F., "On spectral learning of mixtures of distributions," in [*Conference on Learning Theory (Lecture Notes in Artifical Intelligence v.3559)*], 458–469 (June 2005).
[14] Tan, P.-N., Steinbach, M., and Kumar, V., [*Introduction to Data Mining*], Pearson/Addison-Wesley (2006).
[15] Fischler, M. A. and Bolles, R. C., "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM* **24**, 381–395 (June 1981).
[16] Schneider, P. J. and Eberly, D. H., [*Geometric Tools for Computer Graphics*], Morgan Kaufmann Publishers (2003).
[17] Fisher, R. A., "The use of multiple measurements in taxonomic problems," *Annals of Eugenics* **7**, 179–188 (1936).

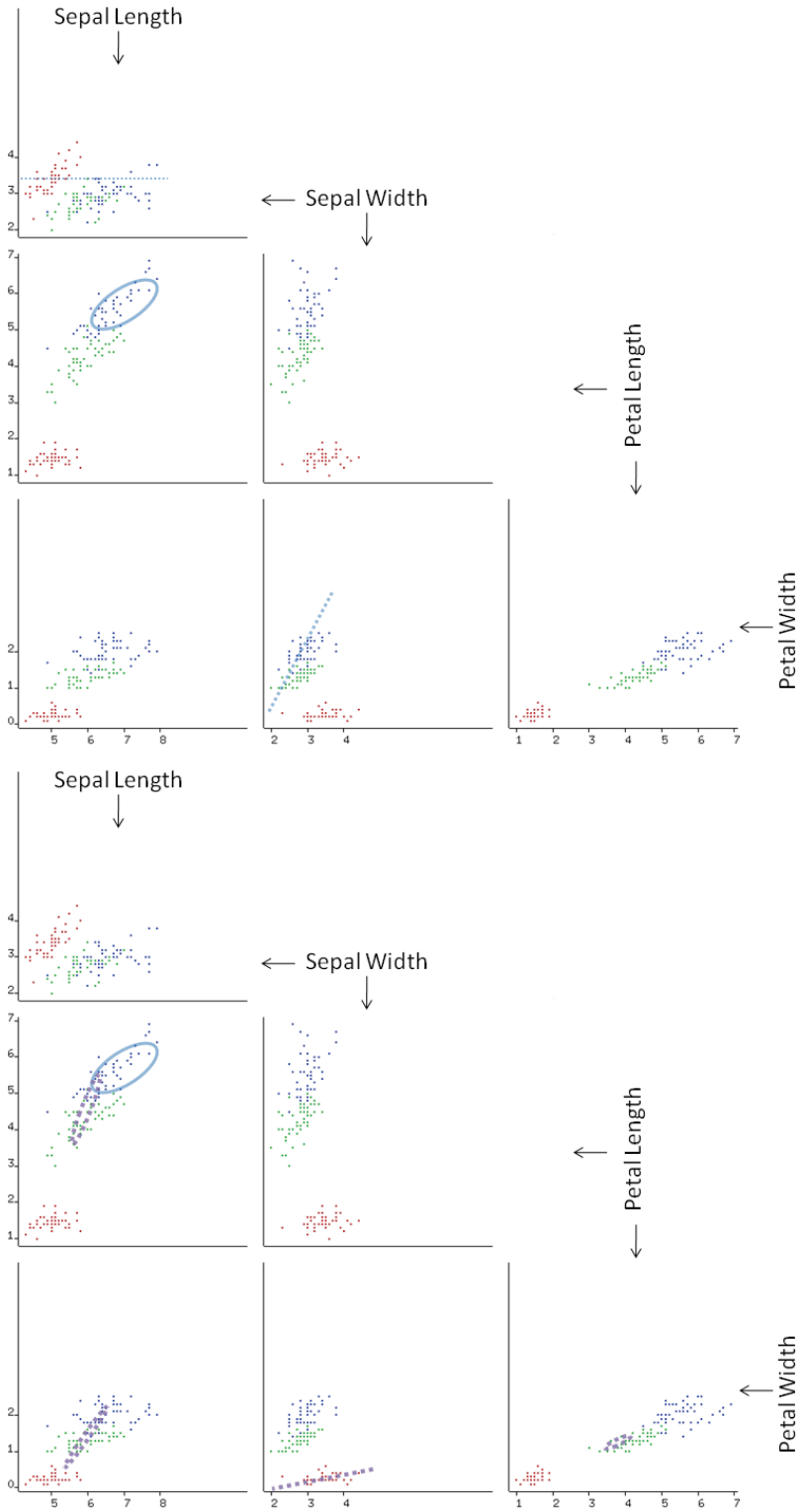

Figure 3. Early (top) and intermediate (bottom) stages of candidate models generated for the Fisher iris data. Our algorithm smoothly operates in whichever dimensions and with whatever functions are selected at random, converging to solutions in any dimensions. All models are generated sequentially; multiple models are introduced in each image to reduce space. For illustration, models discarded after the stages depicted are drawn with dashed lines. New models are drawn in a new color. The elliptical model in the intermediate is a 3D model, appearing in the appropriate scatterplots.

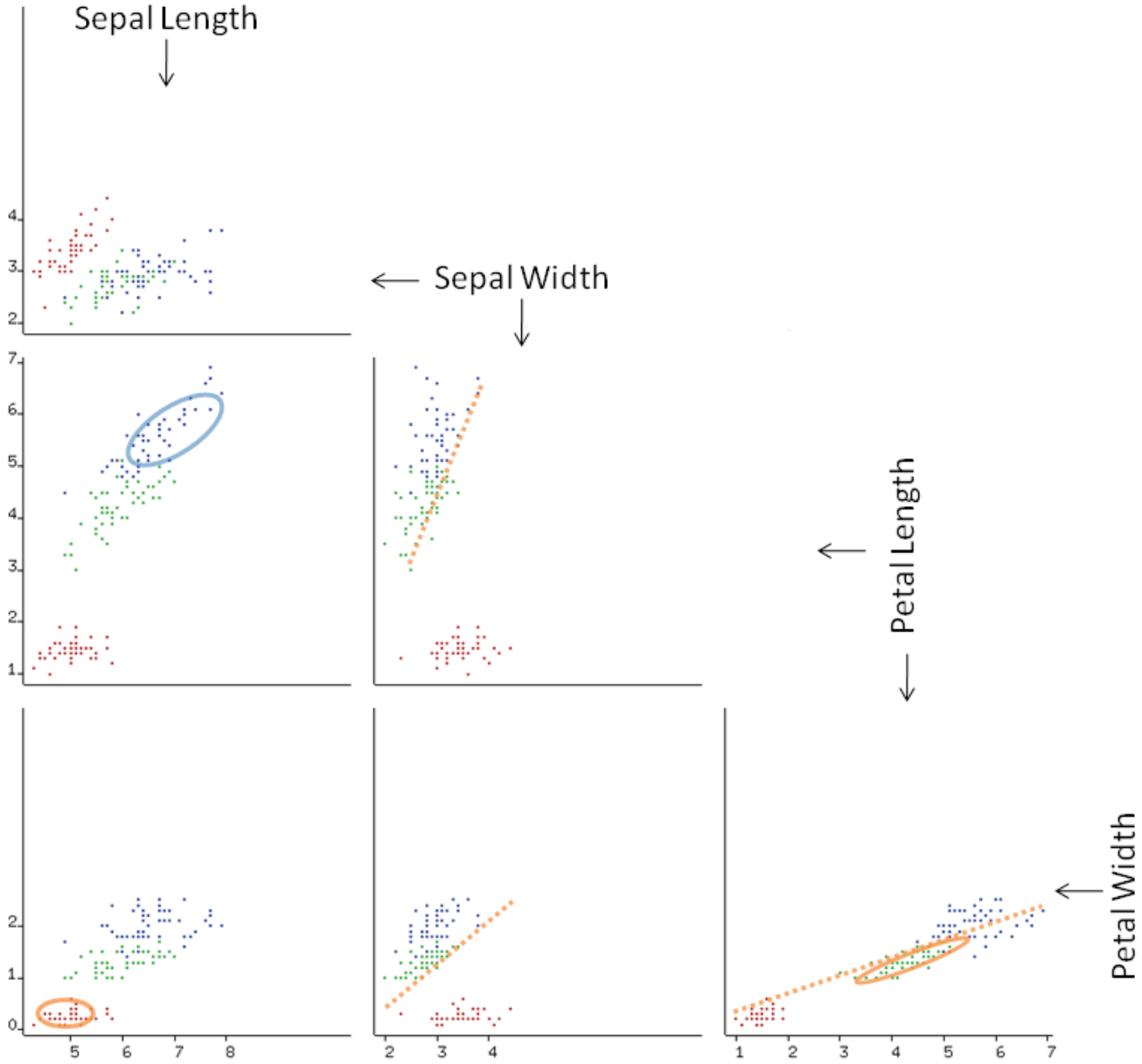

Figure 4. Final models proposed, along with a 3D line candidate generated late in the process, but discarded. It is interesting to observe that all three final models are ellipses; all line models and all models of more than two dimensions were discarded. The virginica class (blue or dark dots higher in value along the petal length dimension) is not entirely captured by the ellipse that supported mostly by the virginica points.