UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ENHANCED SPATIOTEMPORAL RELATIONAL PROBABILITY TREES AND FORESTS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

NATHANIEL TROUTMAN
Norman, Oklahoma
2010

ENHANCED SPATIOTEMPORAL
RELATIONAL PROBABILITY TREES AND FORESTS

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Dr. Amy McGovern (Chair)

Dr. Dean F. Hougen

Dr. Andrew H. Fagg

Dr. Kelvin Droegemeier

Dr. Keith Brewster

Acknowledgments

First and foremost I'd like to thank my wife, Megan, and my family for encouraging me to follow my interests in computer science and artificial intelligence and to actually finish the process of obtaining a Masters degree.

I'd like to thank my committee members, Andrew H. Fagg, Dean F. Hougen, Keith Brewester, Kelvin K. Droegemeier, for their time, insights, and comments that strengthened my thesis. And especially to Amy McGovern as my committee chair and thesis adviser, for the time and effort she has put into ensuring I finish and the friendship and encouragement along the way.

I would also like to thank the REU students and graduate students in the IDEA lab for their help in transforming much of the data used, especially the meteorological data, into a useable form for the SRRFs. Thanks to Nathan C. Heirs, David John Gagne II, Timothy A. Supinie, Ross Kimes, and Bradley L. Pritle for their work on the tornado dataset, to Derek Rosendahl for the initial simulations using the Advanced Regional Prediction System developed by Center for Analysis and Prediction of Storms (CAPS); to Timothy A. Supinie for his work on the Turbulence dataset and to John K. Williams and Jennifer Abernethy at the National Center for Atmospheric Research (NCAR) for providing the data; to David John Gagne II for his work on the Fronts dataset, James E. Hocker and Jeffery B. Basara for the Oklahoma Supercell data, and to the Oklahoma Mesonet for providing the needed meteorological data.

I am also grateful for my friends who've made the long processes of research and even longer process of actually writing the thesis bearable by providing encouragement, academic discussions, and the occasional entertainment: Scott Hellman[1], Zachery Tidwell, Patrick Yost, and Chris Utz.

Much of the computing for this project was performed at the OU Supercomputing Center for Education & Research (OSCER) at the University of Oklahoma (OU).

[1] And for the ever so helpful description of my results in the early stages of writing: "We did some stuff and things worked pretty well."

The Oklahoma Mesonet is funded by the taxpayers of Oklahoma through the Oklahoma State Regents for Higher Education and the Oklahoma Department of Public Safety.

This research was supported by the National Science Foundation under Grant No. NSF/IIS/0746816.

Table of Contents

[2]Description of Turbulence domain adapted from (McGovern et al. 2010)
[3]Description of Fronts domain adapted from (McGovern et al. 2010)

List Of Tables

List Of Figures

Abstract

Many real world domains, such as severe weather events, are inherently spatiotemporal in nature. Each year severe weather induced by thunderstorms causes property damage, injury, and loss of life. Convectively induced turbulence is a hazard to airlines, which at best requires rerouting flight paths, but can lead to significant delays and even structural damage to the aircraft and loss of life. Tornados are possibly the most impressive and destructive potential product of thunderstorms. Domains such as severe weather require a system that is capable of representing and reasoning about complex spatiotemporal data. The dynamics of attributes and relationships varying both spatially and temporally provides a unique set of challenges.

Spatiotemporal Relational Probability Trees (SRPTs) are a type of decision-tree that reason with complex spatiotemporal relational data. High level objects and their relationships are extracted from the raw low level dataset. This allows the SRPTs to reason in more abstract terms and to use relationships that are critical to understanding how things interact. Combining SRPTs into random forests creates a Spatiotemporal Relational Random Forest (SRRF), which is capable of capturing more varied and complex concepts than single trees.

This thesis introduces significant enhancements to SRPT that increases its ability to reason about spatiotemporal data. Often, high-level objects come from scalar- or vector-valued two- or three-dimensional temporal regions called fields. Previously these fields were discarded after the generation of high-level objects. We add the ability to reason about both the objects and the fields within the objects. These fields allow us to add the ability to ask question about the gradient, divergence, and curl of those fields. We also add the ability to recognize the shape of fields, allowing for questions regarding change of shape and orientation. Lastly, we add the ability to reference a single object within the data, and simple boolean operations for combining two questions. These additions are validated using SRRFs on a several real-world.

The SRRF algorithm learns robust classifiers on each of the domains, either outperforming the SRRF without fields or performing equally well. Analysis of the forests

produced showed that features the SRRF algorithm used were consistent with meteorological theories. We show that the addition of fields can be a valuable resource to the SRRF algorithm for spatiotemporal analysis.

Chapter 1

Introduction to Relational Learning using Spatiotemporal Relational Probability Trees and Forests

Our world is filled with objects such as people, buildings, vehicles, all of which are in relationships with each other. For example,'Bob is *in* his Car' or 'Jill *works at* ABC Construction.' Many real-world domains are readily, and intuitively, represented using the relational nature of the world. Frequently these relations are not stationary in time, Bob is not always in his Car and Jill might quit her job. Furthermore, much of the real-word is located in space, such as the location of Jill's office building, and yet others move through space, such as the location of Bob's car. But until recently research has been focused on flattened representations of the data that ignore the inherent relations. Instead the algorithms use a propositional model of the data such as is done by association rules in sales data (Agrawl and Srikant 1994).

Of particular interest to us is using decision trees to study relational data. One benefit of decision trees is that they are human readable. Human readability allows for verification of the knowledge by domain experts and for sanity checks during initial experiment setup. Sanity checking is very important as the algorithms simply do what they are told regardless of what one intends for them to do. Several times during early phases of this research the classifiers were able to essentially cheat by finding patterns in the data that were a result of the processing and conversion of raw data to a useable form. One such example comes from our work classifying storms as potentially tornadic, where the algorithm picked up on the duration of the storm as important. While that was supported by the data provided, it wasn't answering the question in a useful manner for prediction.

Many techniques exist for learning relational models. Recent work has expanded the ability of these relational models to include changes in time and space. While there are interesting static relational datasets, many other fascinating datasets exist that

are temporal, such as severe weather. Temporally varying data adds a new dimension of difficulty to creating accurate models. However, it also provides useful information that can be leveraged (Sharan and Neville 2008). With the addition of spatial data one is able to represent an even larger number of potential datasets. The need to reason on relational data that varies temporally, spatially, and spatiotemporally has led to the active field of research into spatiotemporal relational models (McGovern et al. 2008).

Decision trees are not a new field of research. Early use of decision trees worked solely with propositional data. Quinlin's C4.5 algorithm (Quinlan 1993), a statistical classifier, has become a cornerstone of decision trees in machine learning. Neville et al. (2003) introduced relational probability trees (RPT) built upon probability estimation trees (PET) (Provost and Domingos 2000) by adding the ability to use relations in the data. RPTs were extended yet further by McGovern et al. (2008) to leverage temporal and spatial aspects of the data creating spatiotemporal relational probability trees (SRPT). However, the early SRPT algorithm was limited on its ability to ask spatial questions, containing more temporal distinctions than spatial.

Many models take a biased position on modeling the spatiotemporal aspects of data, resulting in models that are focused primarily on the spatial or temporal features alone. This thesis focuses on improving the modeling of the *combination* of spatial and temporal features. By creating a model that allows for equal ability to make temporal, spatial, and spatiotemporal distinctions, we improve the ability of the spatiotemporal relational models to accurately model real world problems. We choose to extend SRPT's because of the benefits of decision trees and the effectiveness of the SRPT algorithm for classifying spatiotemporal relational data (McGovern et al. 2008; Supinie et al. 2009; Gagne II et al. 2010; McGovern et al. 2010).

The major contribution to the SRPT algorithm is keeping the low level fields with the high level objects to create fielded objects. The retention of fields allows for two new types of distinctions, shape based and field functions. The shape of a field is detected and can be used as an attribute to reason upon using one of three questions: Does the field have a given primitive shape? Does the field's shape change from one primitive shape to another primitive shape? Does the major axis of the shape tilt by a given amount? Three field functions are applied to the fields, gradient to scalar fields and curl and divergence to vector fields. In addition to the new field based distinctions, we added a distinction that selects a single object and applies another

distinction to only that object, and a conjugate boolean distinction that combines two distinctions with boolean-and or boolean-or.

Chapter 2

Literature Review of Propositional, Relational, and Spatiotemporal Relational Data Mining Algorithms

As technology advances and digital storage becomes increasingly inexpensive, a massive amount of data is being collected and stored. The amount of data has quickly surpassed what traditional long-hand techniques can process, thus requiring new automated methods of data processing. This has given rise to the field of Knowledge Discovery and Data Mining (KDD) that since its inception has been a fast and continuously growing area of study.

Blockeel (1998) presented the idea that KDD is composed of three subtasks: preprocessing the data, application of an algorithm, and post-processing the algorithms result. The first step of *pre-processing* the data takes the raw data, in whatever form it is in, and transforms it into a form useable by the data mining algorithm that will be used. Next a *data mining algorithm* is applied to the pre-processed data producing some form of results. The results from the algorithm are *post-processed* into the most useable form. In this review, we focus on data ming techniques. Data mining techniques can be divided into to two categories, descriptive and predictive. Descriptive techniques try to characterize as much of the data as possible by finding patterns and associations within it. Predictive techniques try to form a hypothesis that correctly classifies all the data. Since the work in this thesis is predictive, we focus the review on the this aspect of data mining.

KDD has a long and rich history with a wide variety of techniques that approach the predictive problem from different angles. Figure 2.1 is a hierarchy of some of the major algorithms for predictive data-mining. The algorithms fall into two broad categories, propositional and relational. Propositional algorithms work on datasets that are easily visualized as a single table where each row an example and the columns are the attributes. Within the propositional framework several splits formed as some

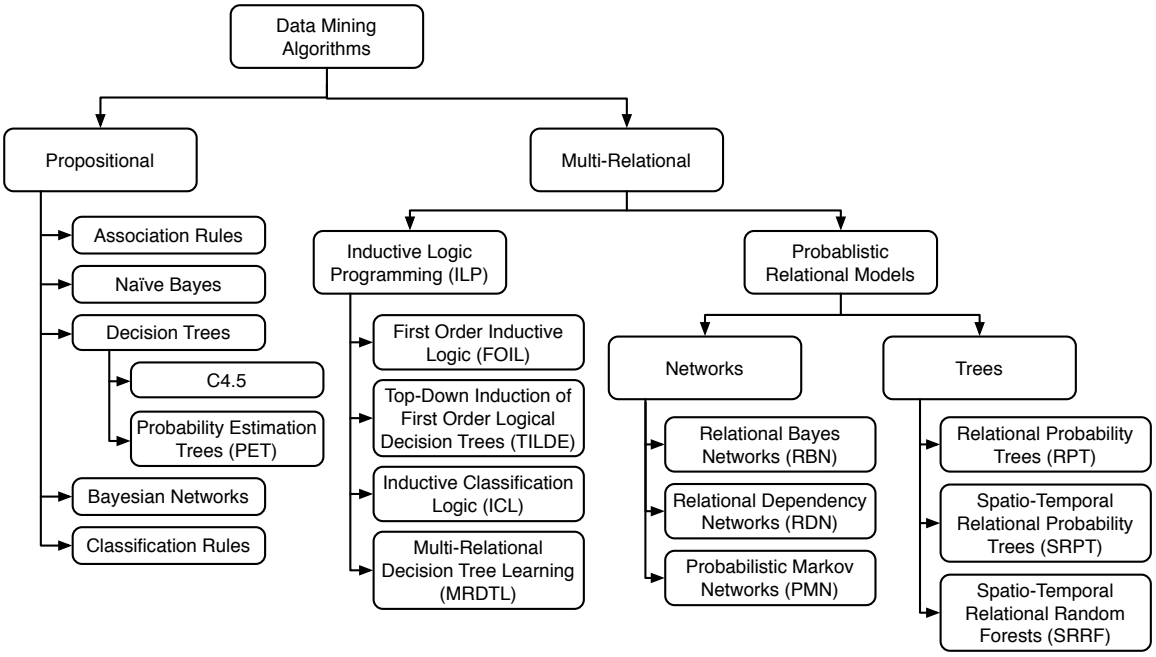


Figure 2.1: Data mining algorithms

researchers focused on rule based models, *association rules* and *classification rules*, and others graphical-models, *Naïve Bayes* and *Bayesian networks*.

A third approach was also taken with the introduction of decision trees. Quinlan (1993) introduced a series of propositional decisions trees that including the well known algorithm C4.5. The C4.5 algorithm takes a set of vectors as training data. Each component of the vector is an attribute of the sample, either continuous or discrete. A discrete class label is also associated with each training vector. Trees are grown by selecting a test for the current node, starting at the root and growing recursively. There are three possible tests. Let A be the value of a discrete attribute with possible values Z. The first test is simply "$A = z$ s.t. $z \in Z$." The second test is "$A \in G$ s.t. $G \subset Z$" where G is found by a greedy partition of Z that maximizes the splitting criteria. If the value A comes from a continuous attribute, let Z be the unique values it takes on in the dataset. The only test for continuous attributes is "$A \leq \theta$" with theta selected considering a value between each pair of consecutive values in Z, such that the that the splitting criteria is maximized. C4.5 uses information gain ratio as the splitting criteria to select from the set of possible splits. The *best* split is the one with the highest information gain ratio and no statistical testing is done for the significance of the split. The tree is allowed to grow until all leaves contain pure subsets of the training data or the tree runs out of attributes. Instead of stopping

criteria to halt the growth of a tree, C4.5 prunes the fully grown tree. C4.5 trees are very capable classifiers for propositional data, however, the algorithm is limited to propositional data and cannot be directly applied to relational datasets.

Probability estimation trees (PETs) were created to address the fact that prior decision tree work was on categorical classifiers. However, for many applications categorical classification isn't sufficient and class probabilities are required. For instance, consider a search ranking algorithm that ranks results on how interesting they are to the user. Class probabilities enable the results to be ranked by the probability that the results are in the *interesting* class. PETs are decision trees where the leaf nodes contain an estimation of class probability instead of a single categorical classification. Research had shown that naïve use of decision trees produced poor probability estimators. However, Provost and Domingos (2000) showed that decision trees are not inherently poor estimators and steps can be taken to produce good class probability estimators using decision trees. A large body of work has shown successful use of PETs in a wide variety of domains from speech recognition (Jelinek 1997) to collaborative filtering (Heckerman et al. 2000) to network diagnosis (Danyluk and Provost 2000) and cost-sensitive learning research (Domingos 1999; Provost et al. 1998).

Provost and Domingos (2000) address several methods that can increase the quality of the probability estimation of PETs to make decision trees reasonable probability estimators. One potential problem with PETs is the probability estimation from small samples. Provost and Domingos (2000) suggest the use of a Laplace correction. The normal class probability estimate is $\frac{f_c}{N}$ where f_c is the number of instances of the class c at the current leaf node and N is the total number of instances at the node. The Laplace correction estimates the probability as $\frac{f_c+1}{N+C}$ where C is the total number of classes.

Provost and Domingos (2000) also show that the algorithms used to grow the decision trees are at fault for producing poorly performing probability estimators. The modern technique of growing large decision trees and then pruning them back, while yielding highly accurate and small trees, yields generally poor probability estimators. Care must be taken when pruning to retain good probability estimation. Reduced error estimation is generally a poor choice for this goal, instead a form of chi-squared pruning produces better estimators. Chi-squared pruning collapses leaves into their parent node if the chi-squared test fails to show a statistically significant difference in the class distribution at the leaves compared to the parent. This thesis takes

the opposite approach to pruning, instead opting to stop growth early by requiring splits to be statistically significant. Our approach is covered in further detail in the description of our algorithm.

While propositional models existed from the start of KDD, they were quickly joined by relational models. In order to better model real world problems, data are no longer flattened into a single propositional table but instead left in multiple "tables" connected by relationships. This provides a wealth of new information and prompts the creation of new data mining algorithms specifically designed for relational data. These new algorithms are called "multi-relational" algorithms, in contrast to "propositional."

Within multi-relational data mining two schools formed: one based upon logic programming and the other on probabilistic models. Logic programming is a programming paradigm where programs consist of first order logic clauses to represent relations between objects. Using these clauses, deductive reasoning can be applied. Inductive logic programming (ILP) was initially created as a method to synthesize logic programs from examples and background knowledge. Given an initial set of first order logic clauses, called the background, and a set of positive and negative examples, ILP generates a set of hypotheses (first order logic programs) that when combined with the background knowledge explain the examples. (Muggleton and De Raedt 1994). ILP can also perform the standard classification task.

First Order Inductive Logic (FOIL) extends work in ILP by using relations in addition to simple first order logical clauses (Quinlan 1990). Previous ILP work focused on propositional data with attributes describing each object. However, this limits the expressive power of the standard language of ILP. For instance, consider a network of related nodes. Using a propositional language requires an attribute for each possible connection. Even if the maximum degree is limited this still produces an unwieldy representation. FOIL adds a representation to the language that allows for describing relationships. Relationships are represented by a named set of tuples where each tuple in the set represents a related group of objects. This allows for a compact representation of the above example of a graph. The goal of FOIL is to generate a set of first order logic clauses that describes a relation in terms of itself or other relations. First order logic, while capable of describing a wide variety of domains, is still limited. It cannot express temporal relations or temporal attributes with out significant modifications.

Blockeel and De Raedt (1998) introduced a new data mining algorithm that combined ILP and decision trees. Top-Down Induction of Logical Decision Trees (TILDE) takes the simple descriptive nature of short first order logic clauses and combines it with the readability of decision trees. TILDE performs the classification task in the standard method of decision trees. At each internal node of the tree is a conjunction of literals that splits examples as they fall through the tree. They eventually end up at leaf nodes that place them into a class. A major advantage of TILDE over other ILP methods, including FOIL, is that it has the ability to make global statements ($\exists X : p(X)$) in addition to existential ones ($\forall X : p(X)$). However, TILDE requires that everything be represented in first order logic. This does not allow for a structure comprised of individual related objects to be represented such as an attributed relational graph.

Multi-Relational Decision Trees (MRDT) extends TILDE to work on a relational database (Knobbe et al. 1999). In MRDT, the trees use selection graphs for the internal node. Each node in a selection graph represents an object type (internally a table in the database) and the edges between nodes represents a relationship between the two objects (a foreign key relation between the two tables). Edges can be flagged as present or absent requiring that they either exist or don't exist respectively. Attached to each node can be a set of constraints on the attributes of the object (constraints on the values of the columns of a table). Selection graphs are built in a top-down manner using a series of refinements. The simplest selection graph is a single node, so connections between nodes are really connections between selection graphs. One refinement adds a new edge between selection sub-graphs. Adding a positive edge requires a relationship to exist and negative edge requires that the relationship does not exist. Another refinement places a constraint upon the value of an attribute of a node. Figure 2.2 is an example selection graph that selects the parents who have at least one child who have a toy and none of these children, who have a toy, are male. The node labeled "parent" selects all parent objects. The upper subgraph limits the selected parents to those who are connected to a "child" object and those children must be connected to a "toy" object. The lower branch further limits the selected parents by requiring that the subgraph *does not* match any of the parents. The subgraph means that no parent that is connected to a "child" object, that have the attribute "gender" with the value "M" (male), and that are connected to a "toy", should be selected.

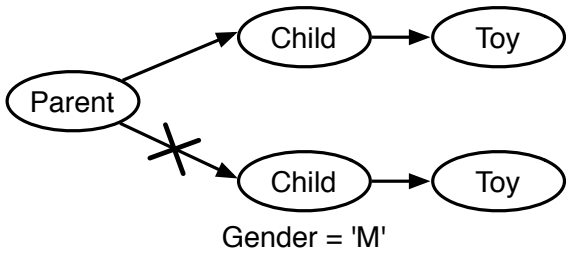


Figure 2.2: Example selection graph used by MRDT that selects a parent who has children who have toys, but none of those children (who have toys) are male. *Modified from (Knobbe et al. 1999)*

The induction of a decision tree follows a recursive algorithm that grows successively refined selection graphs. It starts with a single node in the root selection graph that selects all the objects of the correct type without any refinements. Before each split is attempted the induction algorithm checks the stopping criteria to determine if a split is allowed. A split is attempted by searching all refinements of the parent node's selection graph that are consistent with the data model. Should a satisfactory split be found, the refinement and its complement form the left and right branch of the split respectively. The refinement's complement selects the objects that don't satisfy the refinement. For example, if the refinement is to require a relationship between parent and child, meaning select the parents that has at least one child, then the complement is the parents that have no children. If the stopping criteria is reached or no useable split is found, a leaf node is inserted instead.

An example of a partially grown MRDT is given in Figure 2.3 that classifies a "parent" as having a car or not. With each branch the successive refinements of the previous selection graph is clear, as well as the complementation of a selection graph to form the left and right branches. The root node selects all the *parent* objects. The refinement on the root node selects "parents who have at least one child" and forms the left branch of the tree. The complement of the refinement selects "parents who have no children" and forms the right branch.

Building upon PETs, Neville et al. (2003) developed a new multi-relational probabilistic model on decision trees. Relational Probability Trees (RPT) search over the space of relational features by dynamically propositionalizing the relational features using aggregation. Traditional decision tree algorithms assume independently distributed and homogenous training instances (Jensen and Neville 2002; Jensen et al. 1999), however this assumption is violated in relational data due to the existence

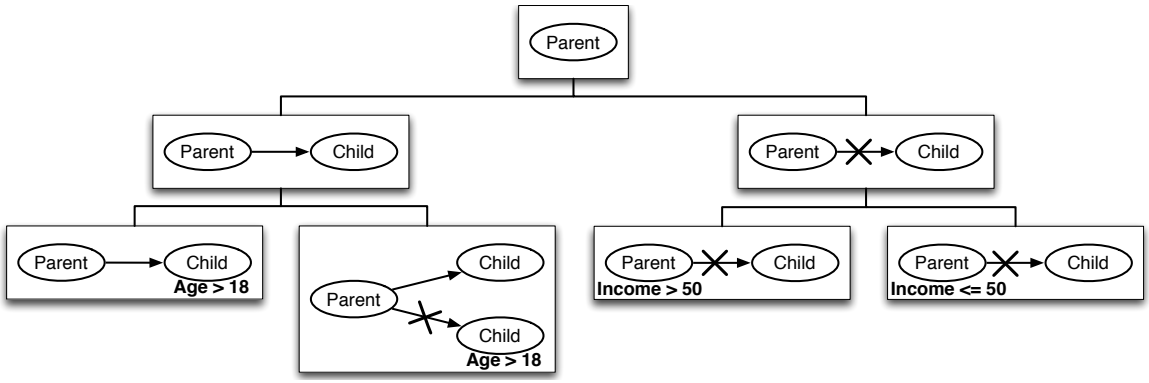


Figure 2.3: Example MRDT that classifies a "parent" as having a car or not. *Modified from (Knobbe et al. 1999)*

of relations. The mere existence of relations makes the objects interdependent, not independent. In addition, not all objects of the same type have the same types of relations or same number of relations. This makes the data heterogenous. RPTs extend the work of PETs to function on such heterogenous and interdependent data. The previously mentioned TILDE and MRDT also work on relational data but under the constraint of first-order logic. RPTs sacrifice some the representational power of ILP systems, such as being able to refer to a single object, in exchange for the ability to search a larger number of feature classes.

Training instances are represented by graphs of objects connected to each other by relationships forming a relational neighborhood. Each graph has a single target object that is to be classified by the RPT. The objects and relationships within the graph can also have standard propositional attributes attached. A common example in relational data mining is predicting if a movie will do well or not. Figure 2.4 shows an example of a training instance graph for the example movie domain. The RPT algorithm searches over the space of binary relational features to split the data. *Mode, average, minimum, maximum, exists, count, proportion, and degree* are the aggregation functions available to RPTs to apply to the relational features. Mode is used for discrete attributes, MODE(actor.gender) == female, and average for continuous ones, AVERAGE(actor.age) > 15. Minimum, maximum, and exists being special cases of these aggregation functions. Count, proportion, and degree use thresholds to split the instances. PROPORTION(actor.gender == female) > 10% and COUNT(actor.age > 55) > 5 are example distinctions. Figure 2.5 is an example RPT on the example movie domain that classifies a movie as being a "success" or a "flop".

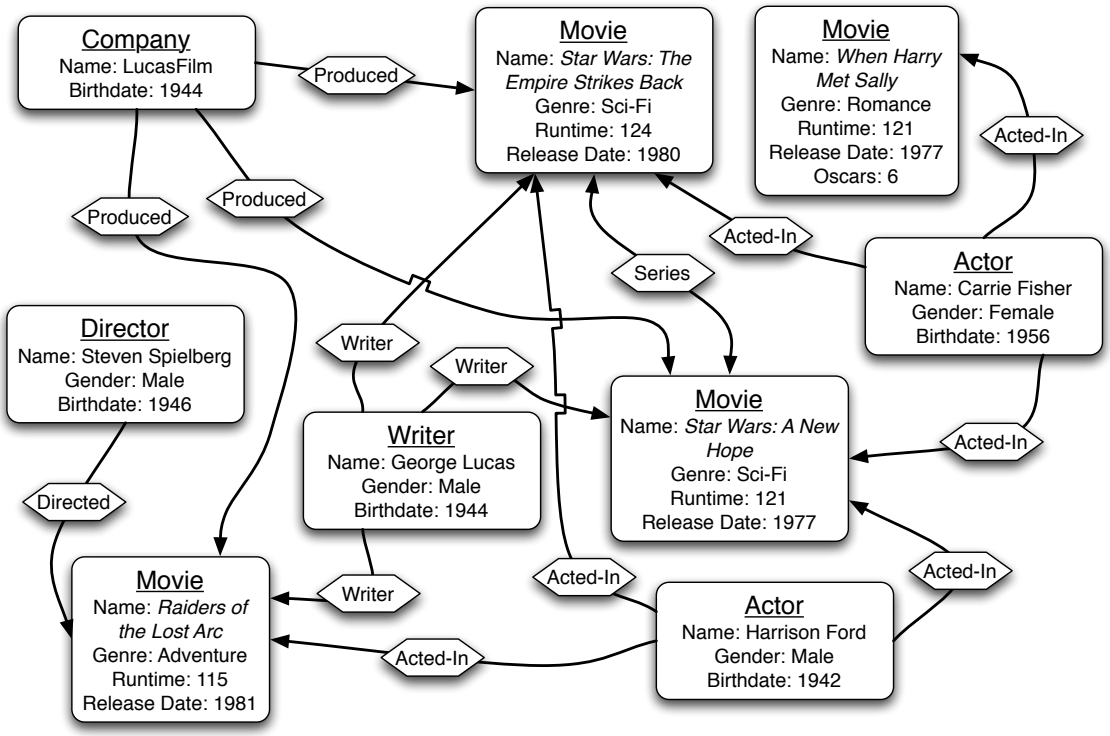


Figure 2.4: Example relational graph used as training instances for RPT.

The questions listed above are not much different than traditional propositional decision trees. Degree features were introduced by RPTs and give RPTs much of their reasoning power. The degree of relationships (e.g. number of phone calls between two people), the degree of objects (e.g. number of movies a producer has produced), and even the degree of attributes (e.g. number of screen-names for an actor) are all degree features searchable by RPTs. These questions provide RPTs a mechanism for handling the heterogeneity of relational data. Neville et al. (2003) showed that RPTs achieve comparable or superior performance than other conditional models, and in addition, build significantly smaller trees.

Neville and Jensen (2007) presented a new probabilistic graphical model for modeling relational data called Relational Dependency Networks (RDN). RDNs have several benefits over relational Bayes networks (Neville et al. 2004) and relational Markov models (Taskar et al. 2002; Richardson and Domingos 2006), including simple methods for parameter estimation, the ability to represent cyclic dependencies, and efficient techniques for learning structure. Much of the strength of RDNs come from the use of a pseudolikelihood learning technique, capable of efficiently approximating the full

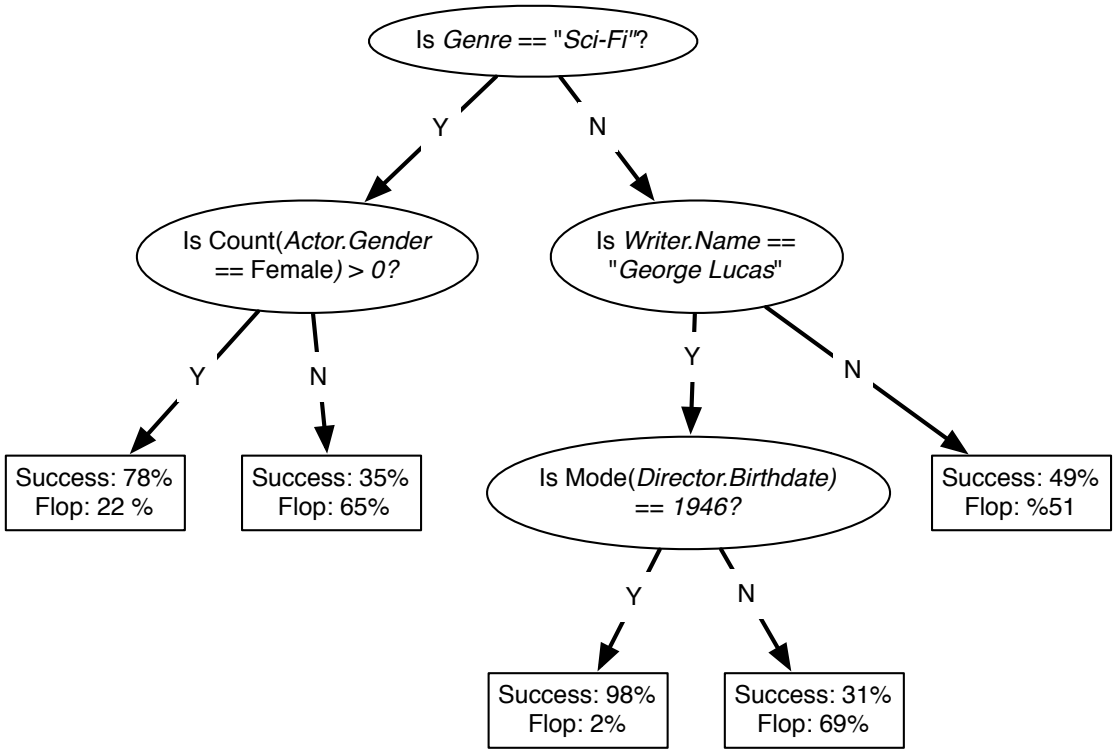


Figure 2.5: Example RPT for the example Movie domain.

joint distribution. RDNs leverage the existence of relational autocorrelation present in most relational datasets (Jensen and Neville 2002). One way RDNs leverage autocorrelation is by learning cyclic autocorrelation dependencies that are required to exploit the autocorrelation during inference.

RDN's primary difference from the similar Bayesian and Markov networks is the approximation of the joint distribution using a set of conditional probability distributions (CPDs) that are learned independently. This technique yields significant efficiency gains over the exact models, however, due to the independent learning of the CPDs, RDNs do not guarantee consistent joint distributions. This restricts the application of exact inference techniques and inference of causal relationships. Nonetheless, RDNs do quite well at inferring predictive relationships.

Chapter 3

Enhancing Spatiotemporal Relational Probability Trees and Forests

3.1 Spatiotemporal Relational Probability Trees

Decision trees can be used for classification by asking questions about an item to be classified. In a binary decision tree, each question can only be answered yes or no. At each internal node of the tree, a question is asked and the item is sent down either the yes or no branch based upon the answer to the question. As the item falls down the decision tree it will eventually land at a leaf node where it is given a class label. The questions asked by the decision tree are also called distinctions as they make a distinction between items with different classes by send them down different branches. Probability Estimation Trees (PETs) give a probability distribution across the possible class labels, instead of a single definitive class label (Provost and Domingos 2000). Neville et al. (2003) extended PETs to reason on relational data instead of solely propositional data creating Relational Probability Trees (RPTs). RPTs can ask questions about the relationships between objects. This allows them to capture more complex concepts then propositional trees. In addition, RPTs added a set of distinctions dealing with degree features of the data.

Spatiotemporal Relational Probability Trees (SRPTs) are an extension of relational probability trees (McGovern et al. 2008). SRPTs add new questions that ask about spatial, temporal, and spatiotemporal characteristics of the data. This requires datasets that support spatiotemporal questions by including spatial and temporal data in addition to the relational data used by algorithms such as RPTs. An example of a possible spatial question is "Does Object A wrap around Object B?" while "Did Object A exist before Object B?" is a temporal question. Combining space and time means asking spatiotemporal questions such as "Is Object A moving closer to Object B?" Figure 3.1 gives an example SRPT that is used for classifying a thunderstorm as tornado-producing or non-tornado-producing.

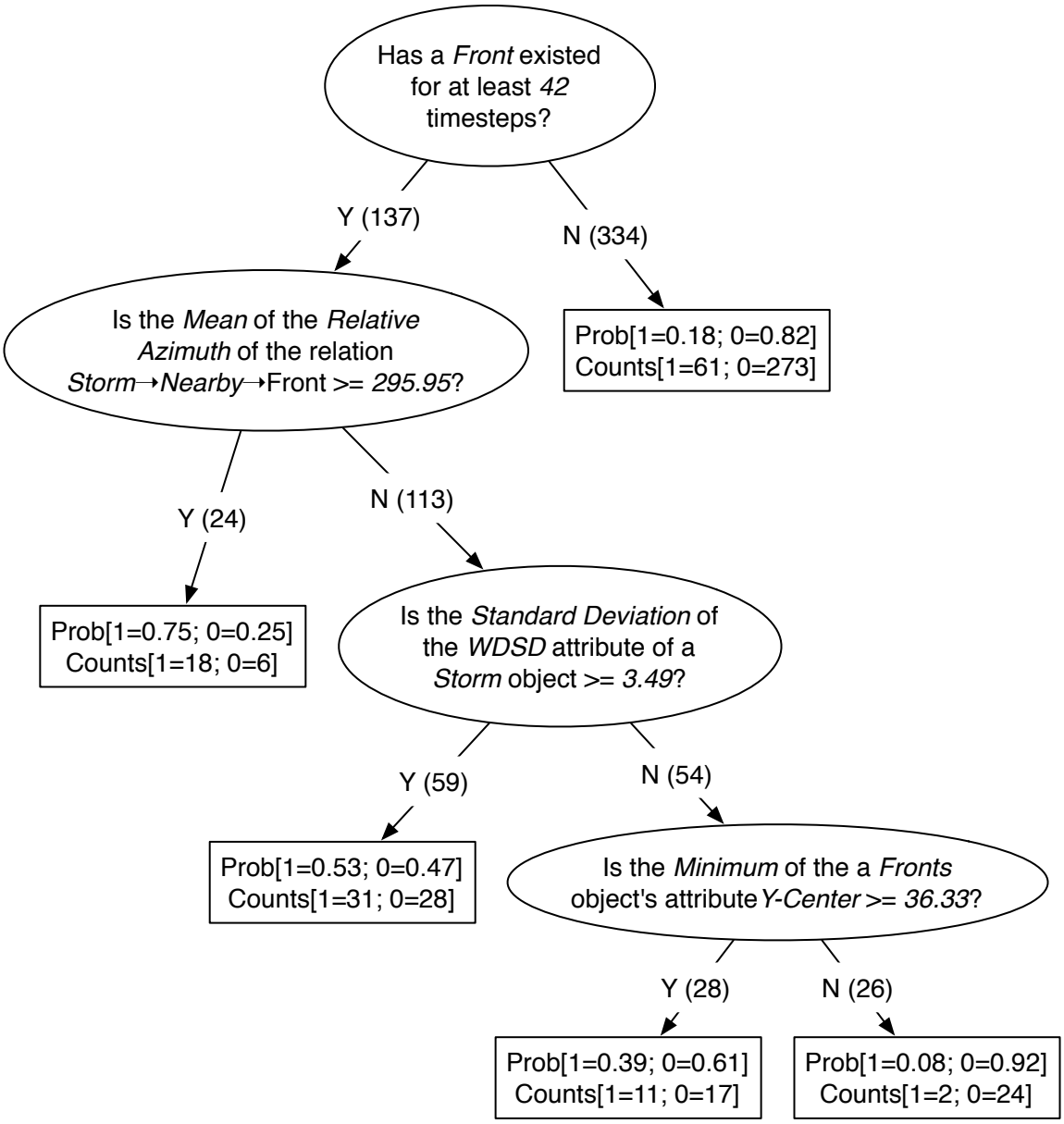


Figure 3.1: Example Spatiotemporal Probability Tree from the Fronts dataset that classifies a supercell as being tornadic or non-tornadic. Oval nodes are distinction nodes and square nodes give the probability distribution function for classification. *Storm→Nearby→Front* is indicates a storm object is related to a front object via a nearby relationship. The labels on the edges indicate if it is the yes or no branch and how many instances went down the branch during training. The leaf nodes contain two pieces of information, the probability distribution (Prob) and the number of instances of each class that were in this leaf during training (Counts).

Before the enhancements we added to SRPTs for this thesis, the SRPT had eight distinctions. They were divided into two categories of distinctions, non-temporal and temporal. We describe each of these distinctions below. For distinctions that have multiple fixed options the options are listed [a, b, c, d], such as the *Attribute* distinction can use of the statistic functions *min*, *max*, *mean*, or *standard deviation* (STD). Some distinctions are conjugate distinctions that ask questions about the results of another distinction. Non-conjugate distinctions are called base distinctions. An example of a conjugate distinction is the *count conjugate* distinction. It takes a base distinction and counts the number of matching items in the graph. If the number of matching items is $\geq$ some threshold then the graph matches. For instance, a count conjugate distinction with the base distinction "Exists object of type *ball*?" and a threshold of *10*, then matching graphs must have 10 ball objects.
The non-temporal distinctions are:

- Exists: Does an object or relation of a particular type exist?
- Attribute (Temporal-Scalar): Does an temporal-scalar attribute a have a [MIN, MAX, MEAN, STD] value $\geq v$?
- Attribute (Discrete): Does an discrete attribute a have a value $= v$?
- Count Conjugate: Are there at least n yes answers to non-conjugate distinction d?
- Structural Conjugate: Is an answer to non-conjugate distinction d related to an object of type t through a relation of type r?

The temporal distinctions are:

- Temporal Exists: Does an object or a relation of a particular type exist for at least t timesteps?
- Temporal Ordering: Do the matching items from non-conjugate distinction a occur in a temporal relationship with the matching items from a non-conjugate distinction b? The seven types of temporal ordering are: *before*, *meets*, *overlaps*, *equals*, *starts*, *finishes*, and *during* (Allen 1991).
- Temporal Partial Derivative (Non-Field): Is the partial derivative with respect to time on a non-field attribute a on a object or relation of type $t \geq v$?

The enhancements we introduce to the SRPT algorithm allow for the addition of a number of new spatial and spatiotemporal distinctions based on shape detection and fielded objects. The distinctions added for using the shape detection are:

- Shape (3D): Is the primary 3D shape of a fielded object a rectangular-prism, sphere, cylinder, or cone?
- Shape (2D): Is the primary 2D shape of a fielded object a rectangle, circle, isosceles triangle, or ellipse?
- Shape Change: Has the shape of an object changed from one of the primary shapes over to a new shape at any point?
- Shape Tilt: Has the primary axis of the shape, at any timestep, tilted by at least θ compared to the vector $\vec{v}$?

The field based distinctions vary in their behavior based upon the type of field, vector or scalar, and the field function applied, gradient, curl, or divergence. For clarity these are listed as separate distinctions and the attributes are all assumed to be temporal.

- Field Gradient: Does the angle between a vector $\vec{v}$ and the [MIN, MAX, MEAN, STD] vector of the gradient of a scalar-field attribute a ever have an angle $\geq \theta$?
- Field Curl: Does the angle between a vector $\vec{v}$ and the [MIN, MAX, MEAN, STD] vector of the gradient of a vector-field attribute a ever have an angle $\geq \theta$?
- Field Divergence: Does the [MIN, MAX, MEAN, STD] of the divergence of a vector-field attribute a ever $\geq \theta$?
- Field Attribute Gradient: Does a field attribute a have a [MIN, MAX, MEAN, STD] of the magnitude of [MIN, MAX, MEAN, STD] vector of the gradient?
- Field Attribute Curl: Does a field attribute a have a [MIN, MAX, MEAN, STD] of the magnitude of [MIN, MAX, MEAN, STD] vector of the curl?
- Field Attribute Divergence: Does a vector-field attribute a have a [MIN, MAX, MEAN, STD] of its [DIVERGENCE] $\geq v$?
- Temporal Partial Derivative (Field): Is the partial derivative with respect to time of the magnitude of the [MIN, MAX, MEAN, STD] of the [CURL, DIVERGENCE, GRADIENT] of the field-attribute $a \geq v$?

The final two new distinctions allow the SRPT to combine two distinctions and to select, and reason on, a single object.

- Boolean AND Conjugate: Does a graph end up in both the yes branches of both base distinction a and base distinction b?
- Boolean OR Conjugate: Does a graph end up in either of the yes branches of both base distinction a and base distinction b?

- Single Item Select By Attribute: Does the single object/relation with the [MIN, MAX] value of the attribute a pass the base distinction b.
- Single Item Select Oldest: Does the oldest object/relation of a particular type pass the base distinction b.
- Single Item Select List First: Does the first object/relation of a given type pass the base distinction b. (Ordering is arbitrary, but consistent,)

3.2 Constructing Probability Trees

The construction of an SRPT follows the approach outlined in Algorithm 3.1 and Algorithm 3.2. The tree grows recursively until adding a new node, or distinction, fails to statistically significantly change the classification distribution or is stopped by the parameters to the learning algorithm. Growth of the trees is controlled by the following user supplied parameters:

- α – p-value threshold
- K – number of samples
- D – max tree depth
- S – distinction set

Growing a SRPT requires using a set of pre-labeled graphs for the training data. As the tree is built, the set of graphs is recursively split as it falls down the tree through each distinction. Choosing these distinctions is the main work of the learning algorithm.

Distinctions are sampled stochastically from the set of all possible distinctions and the "best" distinction is taken. To determine which distinction is best a contingency table is built where the rows are the yes and no branches and the columns are the class distributions. From the contingency table a χ^2 value is calculated from which we can obtain a p-value. As long as the p-value is below the threshold, growth continues along that branch of the tree. Currently the p-value threshold is not adjusted using the Bonferroni correction.

Since the space of possible distinctions is extremely large, stochastic sampling is used to find the best distinction. In early versions of the SRPT algorithm, the sample count was chosen according to a heuristic that relates the number of samples to a desired confidence of α[1] in sampling the top κ percent to consider. α was fixed at

[1]The α in Srinivasan's heuristic should not be confused with the α for stopping SRPT growth.

Algorithm 3.1: **GrowSRPT**:Algorithm used for growing a Spatiotemporal Relational Probability Tree

Input: K = Number of samples, D = Maximum depth of tree, α = p-value threshold, G = training data, S = distinction set, d = current tree depth

Output: A SRPT

```
if d ≤ D then
    node = FindBestSplit(G, K, α)
    if node ≠ ∅ then
        yesSubset, noSubset = SplitData(node, G)
        node.addYesBranch(GrowSRPT(K, D, α, yesSubset, d + 1))
        node.addNoBranch(GrowSRPT(K, D, α, noSubset, d + 1))
        Return node
    end
end
Return new LeafNode(G)
```

95% and κ was a parameter to the algorithm. Srinivasan (1999) showed the number of samples chosen using this method is independent of the size of the search space with the number of samples calculated as $K = \frac{\ln(1-\alpha)}{\ln(1-\kappa)}$. The method of stochastic sampling of the distinction space as performed by our implementation of the SRPT algorithm does not meet the assumption of uniform sampling required for Srinivasan's heuristic to hold. For this reason, the current implementation of the SRPT algorithm uses a user supplied parameter, k, that directly specifies the number of distinctions to sample.

A tree can potentially grow unbounded resulting in over-fitting and unnecessary complexity. The maximum size of the tree is controlled by the *max-depth* parameter, which limits tree's depth by forcing it to terminate early. The depth can range from zero, having only a root distinction, to a depth limited only by machine memory. At times, the p-value threshold (α) will cut off growth along a branch before the maximum depth is reached; on the other hand, max-depth can cut off the growth even if there still remain distinctions that would meet the p-value requirement. At each node of the tree, the best distinction is chosen by sampling. In order to study the SRPT algorithm under various constraints, the algorithm can be limited to using only a subset of the possible distinctions.

Tree growth ends with the insertion of a leaf node when the tree has reached its maximum depth, no statistically significant split can be found, the number of

Algorithm 3.2: **FindBestSplit**: Algorithm for finding the best split in a set of graphs

Input: G = training data, K = number of samples, α = p-value threshold, S = distinction set

Output: An internal Node or $\emptyset$

best = $\emptyset$

for *i = 1* **to** K **do**

distinction = *new* **InternalNode**(**RandomDistinction**(S))

fit, p = **Calc**χ^2(distinction, G)

if $p \leq \alpha$ ***and*** *fit* > *best.fit* **then** best = distinction

end

Return best

graphs in the current branch has dropped below 20, or the graphs in the current branch comprise only one class. If the number of graphs is below 20 then the χ^2 test is not stable with such small numbers of samples. The naïve estimate of the class probability at the leaf node is $\frac{f_c}{N}$ where f_c is the frequency of class c in the samples, and N is the total number of samples at the leaf node. Provost and Domingos (2000) suggested correcting the probability distribution using a Laplace correction factor. This makes the probability estimation become $\frac{f_c+1}{N+C}$ where C is the total number of possible classes. The Laplace correction is most useful when the number of instances of a class is small and has little effect when the class sizes are large. Many domains, such as the three severe weather domains used in this thesis, contain unbalanced classes, hence, we feel that using the Laplace correction is valid.

3.3 Spatiotemporal Relational Random Forests

A Random Forest (RF) is a simple, yet powerful, technique that has shown promise in many domains (Segal 2004; Meinshausen 2006; Fislason et al. 2006; Bosch et al. 2007; Williams et al. 2008). Introduced by Breiman (2001), random forests combine multiple classification trees, originally C4.5 decision trees (Quinlan 1993), into an ensemble. Random forests outperform single trees and combat challenges such as over-fitting. To grow a random forest, the trees in the forest are grown individually with independent bootstrap resamplings of the training data. The instances chosen by bootstrap resampling for training a tree are called in-bag-instances, those not chosen

Algorithm 3.3: **GrowSRRF**: Algorithm used for growing a Spatiotemporal Relational Random Forest

Input: G = training data, N =number of trees in the forest, K = number of samples, P = p-value threshold, S = distinction set

Output: A SRRF

for i = *1* **to** N **do**

[in-bag-data, out-of-bag-data] = **BootstrapResample**(G)

tree_i = **GrowSRPT**(in-bag-data, K, P, S)

end

Return **Forest**($\text{tree}_{1...n}$)

are the out-of-bag instances. Since each tree is trained on a different subset of the data it allows individual trees to focus on different aspect s of the data. This allows the forest to capture more varied and expressive concepts than any single tree would be capable of. Instead of C4.5 trees, we use spatiotemporal relational probability trees yielding Spatiotemporal Relational Random Forests (SRRFs). This makes random forests capable of reasoning about spatiotemporal data. Algorithm 3.3 details the algorithm for growing a SRRF.

One drawback to forests is that they lose some of the human readability of a single tree. Variable importance is a measure of the importance of an attribute to the classification power of a random forest (Breiman 2001). Variable importance is calculated using Algorithm 3.4. The idea behind variable importance is that there is a link between the values an attribute takes on and the class of an instance. Attributes used in distinctions are picked because their values are connected to the object's class. It is assumed that if the values for attributes used in distinctions are changed then the distinctions will be ineffective. Using the assumption that an attribute's value is important, and that changing the value will have negative effect on performance of the random forest, then variable importance is calculated as follows. First, the out-of-bag instances for each tree are classified and the number of correct classifications the tree makes are counted. Then the attribute's values are shuffled within the out-of-bag instances and the tree is reevaluated, counting the number of correct classifications. If the attribute was important then shuffling would decrease the performance of the tree and fewer instances would have been classified correctly. The difference between the non-shuffled count and the shuffled count yields a raw score. Breiman (2001) uses the assumption that the scores between trees are independent and can therefore be

Algorithm 3.4: Algorithm for calculating variable importance

Input: *forest* = forest to calculate variable importance on, *out-of-bag* = the out-of-bag samples for the given forest

Output: Variable importance for each variable

```
raw_scores = new List
for attribute in all_attributes do
    for tree in forest do
        votes = CountCorrectClassifications(tree, out-of-bag)
        shuffled_outofbag = ShuffleAttribute(out-of-bag, attribute)
        shuffled_votes = CountCorrectClassifications(tree, shuffled_outofbag)
        AppendTo(raw_scores, votes − shuffled_votes)
    end
    importance[attribute] = Mean(raw_scores) / StandardError(raw_scores)
end
Return importance
```

converted into a Z-score that is the variable importance of the attribute. We follow this assumption.

3.4 Fielded Objects

The high-level objects used by the SRPT are derived from raw data that has been aggregated. For instance, in the tornado dataset, a storm might contain a vorticity object representing an area of strong rotational winds. The object could contain an attribute representing the current wind speed and direction in the form of a single vector. The benefit of this high level representation is its compactness and simplicity. However, some information is lost since the vortex being represented occupies a volume of space with wind speed and direction (amongst other variables) available at many points within the region.

Previously, this information was discarded during the creation of graphs used to represent the high level objects. All attributes were reduced into a single discrete, scalar, or vector value. These values are allowed to vary over time producing temporal-discrete, temporal-scalar, and temporal-vector valued attributes. One of the main contributions of this thesis is the use of the underlying raw data of such high level objects. In geographic information systems (GIS) these regions are called fields, and the addition of fields to the high level objects will produce *fielded objects* (FIOBs)

(Goodchild et al. 2007; Cova and Goodchild 2002). By keeping a reference to the underlying field, the SRPT is able to choose from several new distinctions that are all based upon the data in the field, such as gradients over space and/or time. The objects the fields come from are not likely to be rectangular, but the fields are stored in rectangular matrices. This necessitates a mask be kept with the field that indicates which values from matrix are actually part the fielded object.

To use the fields within distinctions, several field functions are used to analyze the data. The standard gradient function can be applied to scalar fields and the curl and divergence operators can be applied to vector fields. Curl in 3D and gradient in 2D and 3D return vector fields, curl in 2D and divergence in 2D and 3D return scalar fields. Since the fields are temporal, the field function is applied to each timestep. After the application of the field function, several statistics are taken to reduce the vector field at each timestep into a single vector, yielding a temporal-vector. These are the minimum, maximum, mean and standard deviation.

The mean and standard deviation are taken by simply calculating the component-wise statistic over the entire field. The minimum and maximum vectors are determined by the standard ℓ^2-norm. The reduction of the field to a single vector per timestep allows for the use of understandable (and more general) distinctions. Also, this allows for the comparison of fields of different sizes. Some distinctions further reduce each temporal-vector into a single temporal scalar by taking the magnitude of the vector.

We add two field distinctions to the SRPT called *Field Attribute* and *Field Function*. The *Field Attribute* distinction is essentially the same as the standard Attribute distinction but it works on the temporal-scalar reduction of the fields. It splits if the min, max, mean or standard deviation of the reduction is greater than or equal to a split value. The *Temporal Partial Derivative* distinction is not a new distinction, but it can use the temporal-scalar reduction of field attributes in addition to normal temporal-scalar attributes. *Field Function* uses the reduction of the field to a temporal-vector. It splits if the angle between a temporal-vector and a split angle is always less than a maximum theta. For example, *Is the angle between the maximum gradient vector of the updraft and the vector <0,1,0> always less than* 5 *degrees?*

3.5 Shape Detection

A new addition that we expect to be particularly useful in tornado prediction is the ability to detect the primitive shape of a field. For 3D fields, the possible primitive shapes are spheres, boxes, cylinders, and cones. In 2D the possible primitive shapes are circles, ellipses, rectangles, and isosceles triangles. We chose the detectable shapes because each shape has a simple method for determining how far a point lies from the surface and determining the normal surface vector at an point on the surface of the shape. Each field has a mask that selects individual points from the raw rectangular/cubiod region. This mask is viewed as point cloud, which is a set of points with regular spacing, that is used for shape detection. A couple of different algorithms with variations were tried before settling on the current implementation. We describe the unsuccessful approaches before the successful one in order for the reader to understand the final choice of approach. For all algorithms a method is needed to determine the quality of the fit of a proposed shape to the field. We used the sum squared-distance between a sub-set of points from the point-cloud and the nearest point on the proposed shape. Half of the points were randomly selected as the subset used for the error calculation of the proposed shape. For testing during writing the algorithms, random points were generated that came from known shapes.

All of the shape detection algorithms followed the same basic steps. First, they propose a set of shapes by generating parameters for the primitive shapes that might fit the point cloud. Different algorithms generate a different number of proposed parameters and some propose multiple parameter sets per primitive shape. The next step is to calculate the error of each proposed shape. Using the error estimations, the best fitting shape is then selected to represent the shape of the point cloud.

3.5.1 3D Shapes

The first algorithm was a RANSAC (RANdom SAmple Consensus) algorithm inspired by the work of Schnabel et al. (2007) on shape detection of point clouds using RANSAC. All the points in the cloud were used and parameterization of each of the four shapes was generated using between two and three randomly sampled points. The error function was the squared distance from a sample point to the surface of the proposed shape. If the sample point was interior to the shape, then the error was

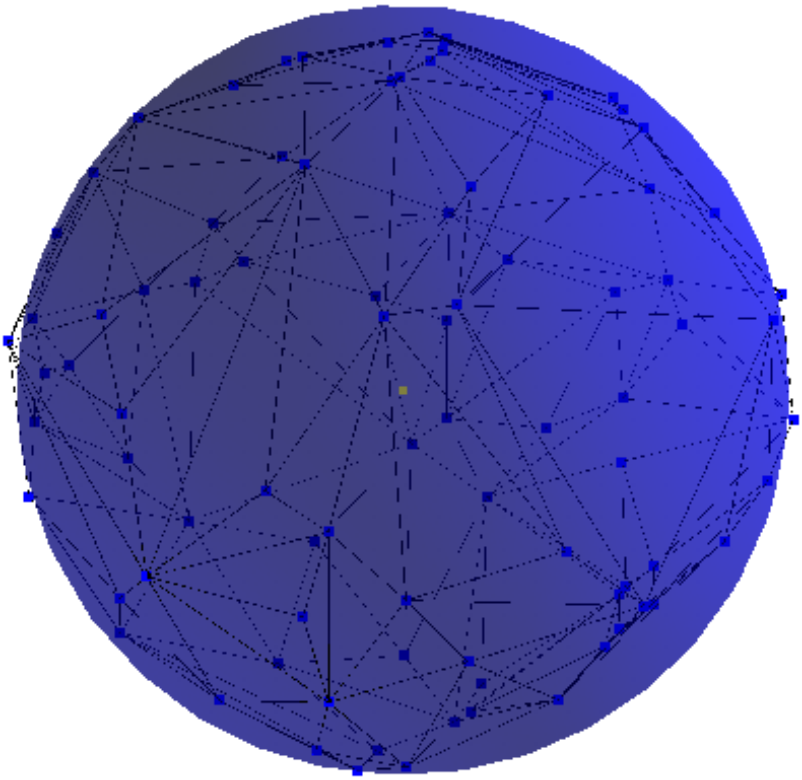
zero. This method preformed poorly, generally classifying things as spheres regardless of the underlying shape.

Hence, we designed our final approach. While Schnabel et al.'s work was concerned with detecting multiple shapes within the point cloud, we are only interested in detecting a single shape. Hence a new algorithm was devised. First the entire point cloud was reduced to a convex hull using Qhull (Barber et al. 1996). Qhull also calculates the face normals allowing us to calculate the vertex normal for each point on the convex hull. This new algorithm is based upon first finding the minimum volume bounding box of the convex hull. Finding minimum volume bounding boxes (MVBB) is in itself a difficult problem. Several methods exist for finding the MVBB. Perhaps the best known is by O'Rourke (1985). It uses the fact that the two edges of the convex hull lie on two adjacent faces of the MVBB. However, the implementation of the exact algorithm is quite complicated due to the Gaussian sphere data structure used to speed up computations. We implemented a similar algorithm without the Gaussian sphere.

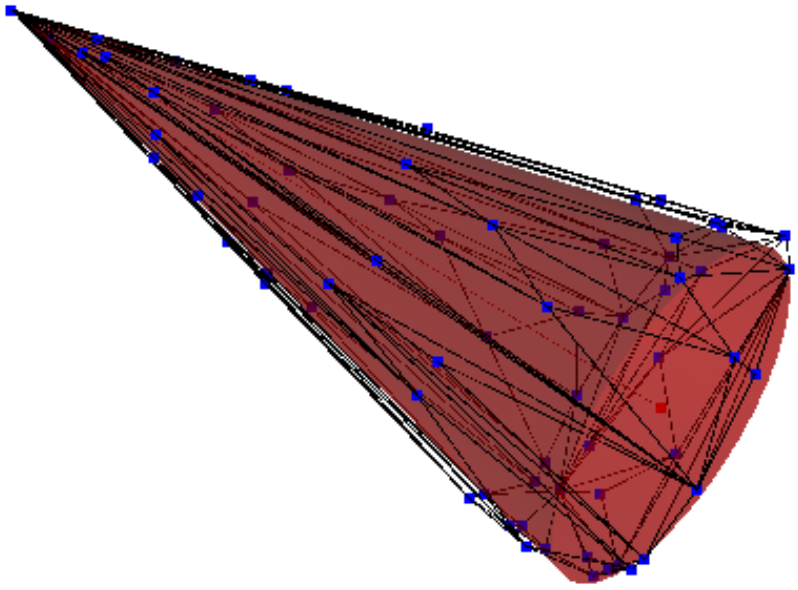
The vertex normals were used to improve error estimations over the simple method used in the previous algorithm. When each sample point is compared to the proposed shape to calculate the error, the sample point's normal is also compared to what the normal would be on the proposed shape. The error for a proposed shape is then the sum error divided by the number of sample point's with the angle between the sample normal and the shape normal is less than a given theta (arbitrarily set to 25 degrees). Additionally, if the number of normals whose angles don't pass the threshold is less than half the total number of sample points, the proposed shape is given infinite error. This method of error estimation greatly improved the quality of fit for detected shapes.

RANSAC is a stochastic algorithm. This means the quality of proposed shapes was left to chance and at times a very large number of proposed shapes were needed to find even a decent fit. Our algorithm based on the MVBB is deterministic and produces a single proposal for each shape from which we pick the best fitting primitive shape. Figure 3.2 shows some examples of detecting each of the different 3D shapes. The MVBB itself is the proposed shape for fitting a box to the points.

We next describe the algorithm more formally. Let P be all the points in the convex hull, k be the number of points in P, and $\hat{N}$ be the surface normals for each of those points (i.e. $\hat{n}_i$ is the normal for p_i). $|\vec{v}|$ will denote the magnitude of a vector $\vec{v}$

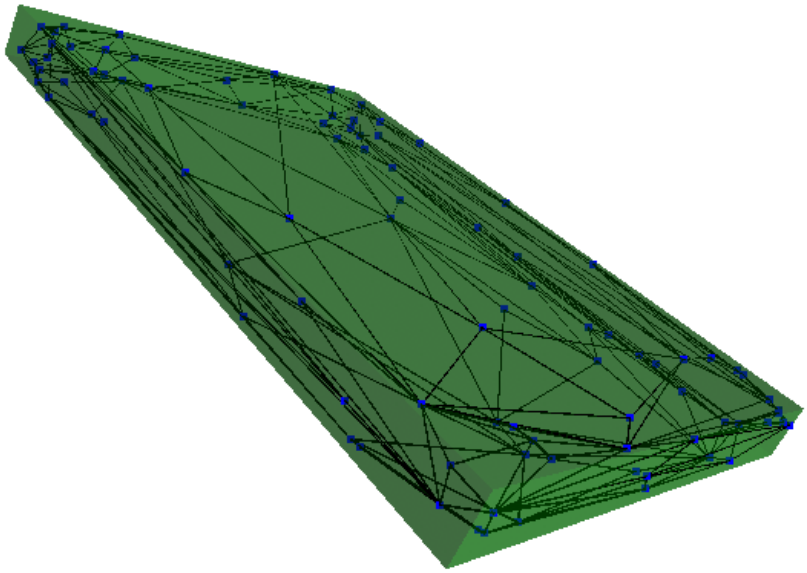


(a) Sphere

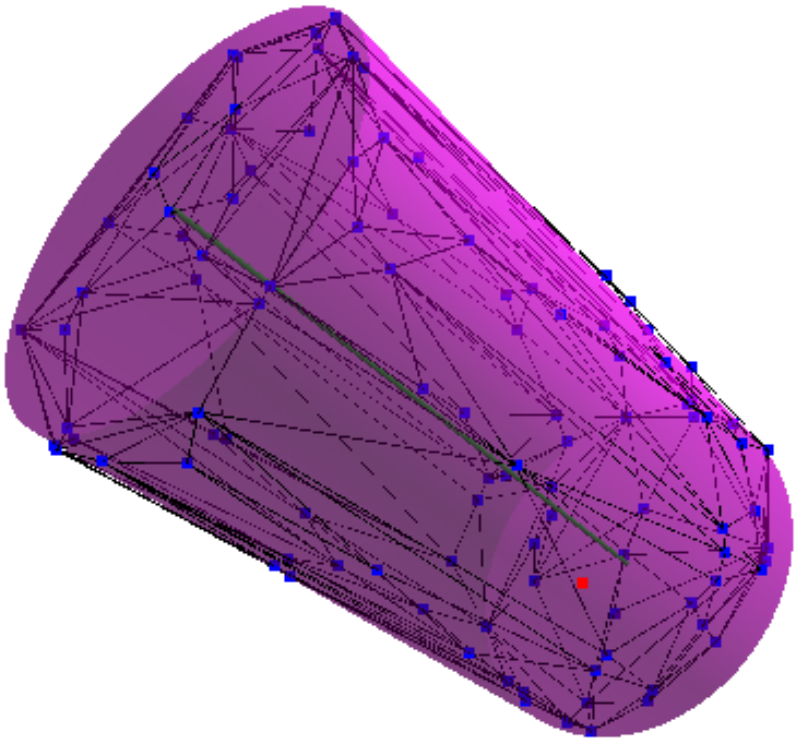


(b) Cone

Figure 3.2: Examples of detecting a sphere and a cone. The convex hull is drawn as a wire frame with the points on the hull shown in blue.



(a) Box



(b) Cylinder

Figure 3.3: Examples of detecting a box and a cylinder. The convex hull is drawn as a wire frame with the points on the hull shown in blue.

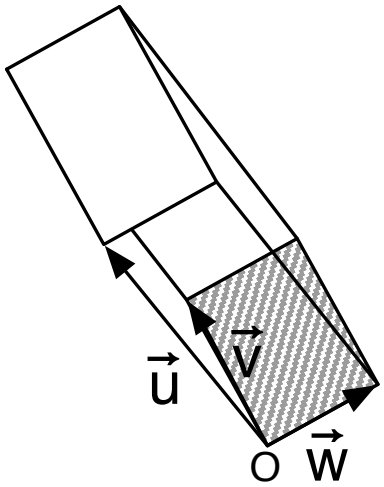


Figure 3.4: Illustration of the assumed orientation of a minimum volume bounding box (MVBB) and the reference edges used to represent it. The "base" of the MVBB is shaded and "O" denotes the origin of the co-ordinate system.

using the ℓ^2 norm. Let $\vec{u}, \vec{v}, \vec{w}$ be vectors representing the edges of the bounding box from one of its corners. Without loss of generality let $|\vec{u}| \geq |\vec{v}| \geq |\vec{w}|$ and translate all points in P such that the corner of the MVBB is located at the origin. Figure 3.4 illustrates how the MVBB is assumed to be oriented and which edges are referred to by $\vec{u}, \vec{v}, \vec{w}$.

The proposed parameters for a sphere have the center located at

$$c = \frac{1}{k} \sum_{i=1}^{k} p_i$$

and the radius

$$r = \frac{1}{k} \sum_{i=1}^{k} |c - p_i| \quad .$$

The mean distance from the center is used as the radius instead of the min or max distance as min/max would select outliers producing a poor fit.

Cylinders are proposed by taking the longest edge of the MVBB, u, as the main axis, $\vec{a} = \vec{u}$ and the radius equals

$$r = \frac{|\vec{v}| + |\vec{w}|}{2} \quad .$$

The base of the cylinder is

$$b = \frac{1}{2}(\vec{v} + \vec{w}) \quad ,$$

which is the center of the "base" of the MVBB.

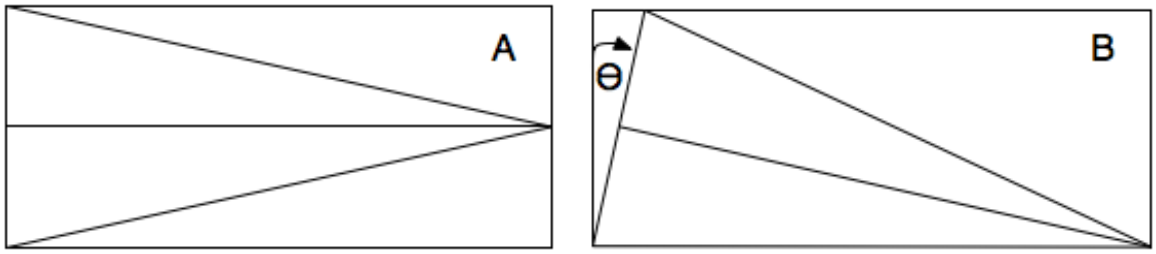


Figure 3.5: Isosceles triangle inscribed in a rectangle illustrates orientation of cones/triangles within *minimum area/volume* rectangular bounding boxes/volumes.

Cones can be viewed as 3D isosceles triangles that have been rotated about their axis to lift them from 2D into 3D. Hence, we can also look at the cone as a projection from 3-space onto two orthogonal planes parallel to the cones axis. The challenge with detecting cones in relation to an MVBB comes from the fact that, for certain isosceles triangles, the base doesn't share an edge with the rectangle, but instead one of the sides shares an edge. Figure 3.5 gives an example of this property with the area of rectangle A being 6.003 units and the area of B is 5.981.

Let the all the points P be viewed as a matrix P_M such that P_M is a $k \times 3$ matrix. Define

$$\text{proj}_{\vec{s}}(p_i) = \frac{\vec{s} \cdot p_i}{\vec{s} \cdot \vec{s}}$$

as the projection of the point p_i onto the vector $\vec{s}$.

To determine the orientation of the cone's axis we find which end is the "wide" of the cone. This is done by projecting all the points on the long edge of the MVBB and then weighting the edge by the distance from a point to its projection. This creates a "heavy" end of the edge which is the base of the cone. First, let A be the set of points on one half of the edge

$$A = \bigcup_{i=1}^{k} p_i \text{ where } \text{proj}_{\vec{u}}(p_i) \geq .5 \ \forall \ p_i \in P$$

and ΣA be the weight of the set

$$\Sigma A = \sum_{i=1}^{|A|} \text{proj}_{\vec{u}}(a_i) \quad .$$

Similarly let B be the set of points on the other half of the edge

$$B = \bigcup_{i=1}^{k} p_i \text{ where } \text{proj}_{\vec{u}}(p_i) < .5 \ \forall \ p_i \in P$$

and ΣB be the weight of the set

$$\Sigma B = \sum_{i=1}^{|B|} \text{proj}_{\vec{u}}(b_i) \quad .$$

If $\Sigma A > \Sigma B$ then the tip of the cone is

$$t = b_j \text{ s.t. } \text{proj}_{\vec{u}}(b_j) = \min(\text{proj}_{\vec{u}}(b_i)) \; \forall \; b_i \in B$$

otherwise the tip is

$$t = a_j \text{ s.t. } \text{proj}_{\vec{u}}(a_j) = \max(\text{proj}_{\vec{u}}(a_i)) \; \forall \; a_i \in A \quad .$$

The initial location of the base is determined using geometry by "viewing" the cone as a projection onto the two vertical planes of the MVBB on which $\vec{v}$ and $\vec{w}$ lie. Using these two planes we can calculate the base of the two triangles projected onto them by the cone. Define vecAngle$(\vec{s}, \vec{t})$ to be the angle in radians between the vectors $\vec{s}$ and $\vec{t}$, rotate$(\vec{s}, \phi, \vec{t})$ be the rotation of the vector $\vec{s}$ ϕ radians about the vector $\vec{t}$, and norm$(\vec{s})$ be the unit length vector of $\vec{s}$. First the following quantities are calculated:

$$\begin{array}{llll} z_1' & = \sqrt{|\vec{u}|^2 - |\vec{v}|^2} & z_2' & = \sqrt{|\vec{u}|^2 - |\vec{w}|^2} \\ z_1 & = |\vec{u}| - z_1' & z_2 & = |\vec{u}| - z_2' \\ \theta_1 & = \tan^{-1}(|\vec{v}|/z_1) & \theta_2 & = \tan^{-1}(|\vec{v}|/z_2) \\ \phi_1 & = \frac{\pi}{2} - \theta_1 & \phi_2 & = \frac{\pi}{2} - \theta_2 \\ r_1 & = |\vec{u}| * \sin(\phi_1) & r_2 & = |\vec{u}| * \sin(\phi_2) \end{array}$$

Using these values we can calculate the two bases. If vecAngle$(\vec{v}, \vec{u}) >$ vecAngle$(b_1, \vec{u})$

$$b_1 = r_1 * \text{norm}(\text{rotate}(\vec{v}, \phi_1, \vec{w})) \quad ,$$

else

$$b_1 = r_1 * \text{norm}(\text{rotate}(\vec{v}, -\phi_1, \vec{w})) \quad .$$

And if vecAngle$(\vec{w}, \vec{u}) >$ vecAngle$(b_2, \vec{u})$

$$b_2 = r_2 * \text{norm}(\text{rotate}(\vec{w}, \phi_1, \vec{v})) \quad ,$$

else

$$b_2 = r_2 * \text{norm}(\text{rotate}(\vec{w}, -\phi_1, \vec{v})) \quad .$$

From b_1 and b_2 we calculate the initial estimation of the base of the cone to be

$$b_0 = b_2 + \text{proj}_{\vec{v}}(b_1)\vec{v} \quad .$$

To refine the estimate of the base's position let $\vec{b_0}$ be the initial base b_0 seen as a vector. Then perform a binary search on the parameter f over the range $0 < f < 2$ to minimize the error, σ, of the cone's fit. The new estimate of the base, b', at each step is

$$b' = f * \vec{b_0}$$

and the search is stopped when $|\sigma_t/\sigma_{t-1}| < 10^{-16}$. From this estimate of the bases position we update the major axis of the cone to be from the base to the tip, $\vec{a} = b' - t$.

The initial estimate of the radius is

$$r = |b_0| \quad .$$

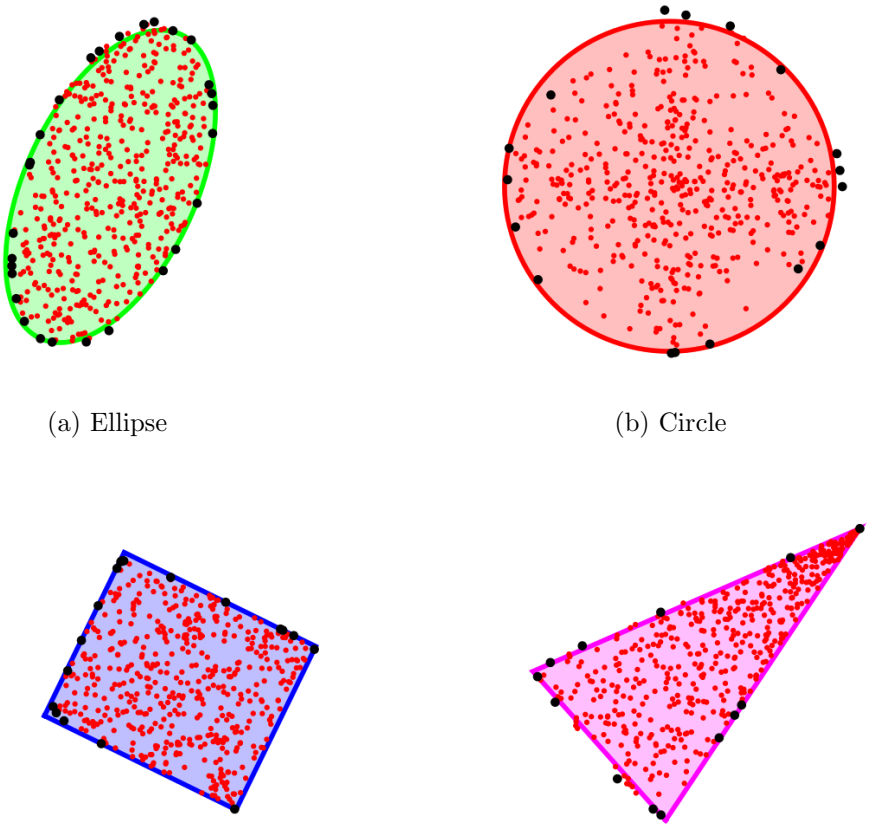
Another binary search is performed on the parameter g over the range $0 < g < 2$ to minimize the error, σ, of the cone's fit. The new estimate of the radius, r', at each step is

$$r' = g * r$$

and the search is stopped when $|\sigma_t/\sigma_{t-1}| < 10^{-16}$. The final parameter estimation for the cone is base b', radius r' and major axis $\vec{a}$.

3.5.2 2D Shapes

In addition to doing 3D shape detection, we added the ability to detect 2D shapes. The first algorithm implemented was also a RANSAC algorithm. While it performed better than the similar 3D algorithm, there was still room for improvement. The final 2D shape detection algorithm is based on the deterministic 3D one. An example of detecting each of the 2D shapes is in Figure 3.6. As with 3D detection, the first step is to find the 2D minimum area bounding box (MABB). The MABB will share an edge with convex hull of the raw points so the algorithm simply tries each edge of the hull (Freeman and Shapira 1975). All the points from the hull are projected onto the initial edge to determine the bounds along that basis, then the basis is rotated 90 degrees to produce a second perpendicular basis. Again the points are projected, now onto the second basis, to determine the bounds. Then using the bounds along



(a) Ellipse

(b) Circle

(c) Rectangle

(d) Triangle

Figure 3.6: Examples of detecting each of the primitive shapes from a point cloud. Red points are the full field with black points being on the convex hull.

each basis the area is calculated. After testing all edges of the convex hull the MABB is found.

We can recognize four 2D shapes: circles, rectangles, isosceles triangles, and ellipses. Isosceles triangles were chosen over the general triangles because a projection of the 3D cone onto a 2D plane from a perspective orthogonal to the main axis projects an isosceles triangles. The equations for estimating the 2D shapes will use many of the definitions form 3D shapes. Similarly to 3D, let $\vec{u}$ and $\vec{v}$, with $\vec{u}$ the longer one, be the edges of the MABB whose corner is the origin. The proposed shape for a rectangle is just the MABB that has already been found.

The proposed parameters for a circle have the center located at

$$c = \frac{1}{k}\sum_{i=1}^{k} p_i$$

and the radius

$$r = \frac{1}{k}\sum_{i=1}^{k} |c - p_i| \quad .$$

The mean distance from the center is used as the radius for the same reason it was in 3D.

The parameters for an ellipse are estimated with the major axis $\vec{a} = .5\vec{u}$, the minor axis $\vec{b} = .5\vec{v}$, and the center at $c = \vec{a} + \vec{b}$. Let

$$F = \frac{\sqrt{|\vec{a}|^2 - |\vec{b}|^2}}{|\vec{a}|} \quad ,$$

from this the two foci are at $F_1 = c - F\vec{a}$ and $F_2 = c + F\vec{a}$.

Triangles, like cones, are more involved than the other shapes. Let the all the points P be viewed as a matrix P_M such that P_M is a $k \times 2$ matrix. Define

$$\text{proj}_{\vec{s}}(p_i) = \frac{\vec{s} \cdot p_i}{\vec{s} \cdot \vec{s}}$$

as the projection of the point p_i onto the vector $\vec{s}$.

To determine the orientation of the triangle's axis let A be the set of points

$$A = \bigcup_{i=1}^{k} p_i \text{ where } \text{proj}_{\vec{u}}(p_i) \geq .5 \ \forall \ p_i \in P$$

ΣA be

$$\Sigma A = \sum_{i=1}^{|A|} \text{proj}_{\vec{u}}(a_i) \quad .$$

Similarly let

$$B = \bigcup_{i=1}^{k} p_i \text{ where } \text{proj}_{\vec{u}}(p_i) < .5 \ \forall \ p_i \in P$$

ΣB be

$$\Sigma B = \sum_{i=1}^{|B|} \text{proj}_{\vec{u}}(b_i) \quad .$$

If $\Sigma A > \Sigma B$ then the tip of the cone is

$$t = b_j \text{ s.t. } \text{proj}_{\vec{u}}(b_j) = \min(\text{proj}_{\vec{u}}(b_i)) \ \forall \ b_i \in B$$

otherwise the tip is

$$t = a_j \text{ s.t. } \text{proj}_{\vec{u}}(a_j) = \max(\text{proj}_{\vec{u}}(a_i)) \; \forall \; a_i \in A \quad .$$

Let $\vec{v'}$ be the short edge of the MABB farthest form the tip t. Let

$$b_1 = p_j \text{ s.t. } \text{proj}_{\vec{v'}}(p_j) = \min(\text{proj}_{\vec{v'}}(p_i)) \; \forall \; p_i \in P$$

$$b_2 = p_j \text{ s.t. } \text{proj}_{\vec{v'}}(p_j) = \max(\text{proj}_{\vec{v'}}(p_i)) \; \forall \; p_i \in P \quad .$$

The base of the triangle is then the midpoint on the line between b_1 and b_2.

3.5.3 Shape Distinctions

We use the automatically identified shapes to add several new distinctions to the SRPT. The simplest is *Shape Match*. It simply checks to see if the field at any time has the given shape type S_1. *Shape Change* asks if at any point the the field has the shape type S_1 and then at some later time changed to type S_2. *Shape Tilt* checks to see if at anytime the angle between the main axis of the field and a given vector v is greater than θ. The main axis for the shapes triangle, cone, cylinder, ellipse is clear. For boxes and rectangles it is the first vector of bounding box and for spheres and circles it is always the y-axis.

3.6 Brittleness of Trees

A decision tree is considered to be "brittle" if a small change, or even no change at all, to the parameters used by the algorithm produces radically different decision trees. The differences between solutions can be both in their *structure* and *classification power*. The structure of an SRPT solution is the shape and choice of nodes in the tree produced. Since structure determines classification power, structural stability will also result in stable classification power.

The stochastic nature of the SRPT algorithm introduces brittleness into the solutions generated. Domain experts in such fields as meteorology are wary of the fact that independent runs over identical data and parameter sets can produce different solutions. However, brittleness and the wariness of domain experts are a concern not only in SRPTs but also in other machine learning techniques such as feature selection (Kalousis et al. 2007)

Random forests draw their power from the differences between the trees from which they are composed. The different trees capture different aspects of the data allowing the forest as a whole to capture a much broader and more detailed picture of the data and hence reason more precisely. This indicates that brittleness is an important feature exploited by random forests. The stochasticity of the algorithm used to generate SRPTs introduces the brittleness, but also allows the algorithm to search a much broader area of the solution space yielding the distinct trees that are located at different spots in the solution space.

Brittleness can thus be used as a measure of over-fitting. By comparing all the trees in a forest, the brittleness of the forest can be measured. A brittle forest indicates that the forest is comprised of very different trees and hence each tree is playing a part in classification. If a forest is not brittle then it is beginning to look homogenous with respect to the individual trees. Homogenous trees means the forest is not capturing a wide variety of aspects, but instead has narrowed down to a small subset and likely over-fitting the data.

Studying brittleness under parameter and experimental changes requires the choice of a metric of similarity between solutions. Since the solutions are trees, the metric must measure similarity between trees. One possibility is the *alignment-distance* metric, which measures the "distance" between two trees based upon the number of changes, deletions, and insertions required to change one tree into the other (Bille 2003). Each of these operations is assigned an associated cost, and the total distance between trees is the total weighted sum of the operations needed to change one tree into the other. For the SRPT the change operation is divided into three weighted operations: adding/deleting a missing/extra distinction; changing the distinction type; and changing the parameters of the distinction.

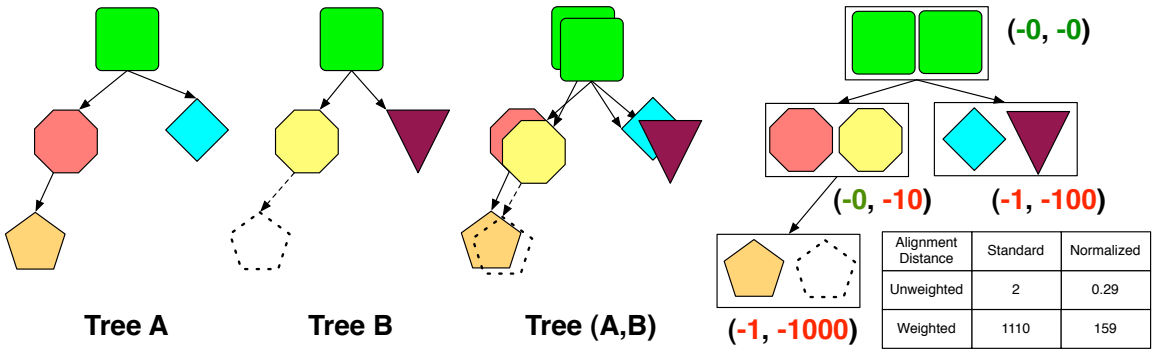


Figure 3.7: Example calculation of alignment distance between two SRPTs.

An example of calculating the alignment distance between trees is illustrated in Figure 3.7. In this example the shapes represent the different types of distinctions and the color represents the parameters for the distinctions. Tree's A and B are aligned on top of each other to form the new tree A, B whose nodes are pairs of distinctions. The alignment is constrained by the inherent ordering of the nodes, with yes and no always being the left and right branches respectively. The distance between the trees is calculated by taking the sum of a distance function applied to each pair of nodes.

First, calculating the alignment-distance requires modifying the trees by adding to them a number of null-nodes such that the two trees become isomorphic when ignoring the actual distinctions made by the real nodes. Next, the two trees are overlapped producing a pair of distinctions at each node. The distance function $\gamma(\delta_1, \delta_2)$ gives the distance between the distinctions δ_1 and δ_2. The alignment-distance of the trees is the sum of the distances between all pairs of nodes.

The choice of the distance function is an important one. Two different functions are used: γ is an unweighted function and γ_w is a weighted one. The distinctions have a type, τ, and a set of parameters, ρ. The full list of distinction types and associated parameter sets is given in Table 3.1. A sample distinction would be "Does there *exist* an object of type *acyclic-downdraft*." In this example τ would be *exist* and ρ would be *acyclic-downdraft*. For the null-nodes, $\tau = \rho = \emptyset$.
The distance functions are defined as follows:

$$\gamma(\delta_{\rho_1}^{\tau_1}, \delta_{\rho_2}^{\tau_2}) = \begin{cases} 1 \text{ if } \tau_1 \neq \tau_2 \\ 0 \text{ otherwise} \end{cases}$$

and

$$\gamma_w(\delta_{\rho_1}^{\tau_1}, \delta_{\rho_2}^{\tau_2}) = \begin{cases} 1000 \text{ if } \tau_1 = \emptyset \text{ or } \tau_2 = \emptyset \\ 100 \text{ if } \tau_1 \neq \tau_2 \\ 10 * |NotInBoth(\rho_1, \rho_2)| \text{ if } \rho_1 \neq \rho_2 \\ 0 \text{ otherwise} \end{cases}$$

where the function *NotInBoth*(A, B) returns the set of elements not in both the sets A and B.

The alignment-distance is biased towards the depth of the tree. Since deeper trees have more nodes, there are more possibilities for discrepancies between trees. Thus, shallower trees have a smaller maximum-alignment distance than deeper trees. To correct for this bias, a normalization term of $1/\sum_{i=0}^{D} 2^i = 1/(2^{D+1} - 1)$ is applied to the alignment-distance, where D is the maximum depth parameter. This results

Distinction Name	Parameters
Exists	*object/relation type*
Attribute	*attribute*, *stat* $\in$ [min, max, mean, std, median, mode, less than, exact], *split value*
Count Conjugate	*count*, *base distinction*
Structural Conjugate	*relation type*, *base distinction*
Temporal Exists	*object/relation type*, *extent*
Temporal Partial Derivative	*attribute*, *extent*, if attribute is a field (*field function* $\in$ [gradient, curl, divergence], *stat* $\in$ [min, max, mean, std])
Temporal Ordering	*temporal ordering* $\in$ [before, meet, overlap, equal, start, finish, during], *base distinction A*, *base distinction B*
Single Item Select Conjugate	*attribute or object/relation type*, *base distinction*, $\textit{item selector} \in \begin{cases} \text{[max, min]} & \text{if attribute} \\ \text{[list first, earliest, oldest]} & \text{if object or relation} \end{cases}$
Boolean	*boolean operator* $\in$ [and, or], *base distinction A*, *base distinction B*
Shapes	*field attribute, split type* $\in$ [match, change, tilt], $\textit{split value} = \begin{cases} \text{shape class} & \text{if split type is match} \\ \text{shape class A, shape class B} & \text{if split type is change} \\ \text{vector, theta} & \text{if split type is tilt} \end{cases}$
Field Function	*field attribute*, *field function* $\in$ [gradient, curl, divergence], *stat* $\in$ [min, max, mean, std], vector, theta
Field Attribute	*field attribute*, *field function* $\in$ [gradient, curl, divergence], *stat* $\in$ [min, max, mean, std, median, mode, less than, exact], *split value*

Table 3.1: List of distinctions and their associated set of parameters.Base distinctions are any non-conjugate distinctions.

in a *normalized brittleness metric* that is a per-node distance, making possible the comparison of trees of different maximum depths.

3.7 Missing Values

A common difficulty when dealing with real-world data is handling missing values. A great amount of research has gone into various techniques to deal with imperfect data. Liu et al. (1997) gives a brief overview of several techniques for handling missing values. In the spatiotemporal framework there are three cases of missing values. The first is where a non-temporal attribute is missing its value, which is analogous to missing values in propositional data. The second and third cases deal with temporal and spatial attributes. Temporal attributes could be missing time steps within the temporal stream, but still have value for other timesteps. Likewise, spatial attributes could be missing values from within the field, but still have many values present. Each of these cases needs to potentially be handled in a different manner.

The simplest method is to discard objects that contain missing values (White 1987). This can have detrimental effects in propositional data, but is likely to have even larger effects in relational data as the removal of an object would also remove all the relationships connected to it. The power of relational algorithms is the ability to leverage the structure imposed by relationships.

In the case of SRPTs it would be possible to simply remove the attribute from the single object leaving the rest of the attributes and the relations unchanged. This comes from the ability of the SRPT to handle heterogenous data. All attributes, objects, and relations are optional. While the SRPT can handle a missing attribute it is still likely that the removal of the good values would be detrimental.

The alternative to removal is to replace the missing values through some method of interpolation. Many different techniques have been suggested each with strengths and weaknesses, such as inferring the value from objects in the same class (Kononenko et al. 1984) or using decision trees to predict the value (Quinlan 1986). Many other techniques for handling missing data have also be proposed (Breiman et al. 1984; White 1987; Friedman et al. 1996). Previous SRPT work made no attempts to handle missing data and required all data to be preprocessed such that the data given to the algorithm was pristine. We added a simple method of dealing with missing values

in 1-dimensional temporal attributes and spatial-attributes (both temporal and non-temporal). In the case of a single 1-dimensional non-temporal attribute we assume that the attribute will simply not be in the data, i.e. there is no indication that the attribute is missing. The SRPT algorithm can easily handle the heterogeneity by the missing attribute.

To handle cases two and three of missing values we use a simple mean of non-missing nearest neighbors interpolation. For case two, a 1-dimensional temporal attribute, the nearest neighbors are temporally local. Hence if a_t is missing it is replaced with the linear interpolation from the two temporally closest existing values, a_{t-i} and a_{t+j} on either side of a_t. For case three the nearest neighbors are spatially local not temporally, regardless if the attribute is temporal or not. In 2-dimensions the nearest neighbors are the eight touching values (counting diagonal as touching), and in 3-dimensions they are the 26 touching values. The missing value is replaced by the mean of the nearest neighbors that are not-missing. Missing values are replaced in order of increasing indices of the z, y,and x dimension ($< 0, 0, 0 >, < 0, 0, 1 >, ..., < 0, 1, 0 >, ..., < 1, 0, 1 >, ..., < n_x, n_y, n_z >$).

The use of a simple mean of nearest neighbors introduces some degree of temporal, for case two, and spatial, for case three, autocorrelation. Research has shown that real-world data is frequently auto-correlated (O'Sullivan and Unwin 2002; Longley et al. 2005; Neville and Jensen 2005), hence the introduction of small amounts of auto-correlation should have negligible effects. At the very least we assume that the possible negative effects of the introduction of auto-correlation is far less than the negative effects of simply removing the attribute. This assumption should be tested in future research on the topic of auto-correlation.

Chapter 4

Empirical Results

In order to fully explore each of the datasets we performed a factorial experiment across three of the parameters used in the construction of SRRFs. The number of trees were [1, 10, 50, 100]. Note that a SRRF containing a single tree is equivalent to an SRPT. The number of distinctions sampled were [10, 100, 500, 1000], and the maximum depth was [1,3,5]. In total, this makes 48 parameter sets. Each set was repeated 30 times so as to be able to generate confident statistics.

Variable importance was also calculated for a subset of the parameter sets in order to yield further insight into the SRRFs produced. We calculated variable importance only on the subset of 100 trees, 1000 samples and max depth 5 in order to see which variables were important in the best performing set of forests. These are the most interesting forests with regards to variable importance. In addition, variable importance is computationally expensive. By limiting the number of forests we use for calculating variable importance to the most interesting subset, the runtime was reduced from 2-3 days to 5-6 hours on some domains . We also calculated the frequency of which distinctions were picked, as well as the frequency of which objects, relations, and attributes were used in those distinctions. For distinctions that referenced an attribute, it counted towards the object or relation counts. Also, relations were counted toward the source and destination objects.

The classification power of a SRPT is measured using the Gerrity Skill Score (GSS). We chose GSS because it was designed for measuring performance on multi-class classification problems, and some of the datasets used in this study contain more than two classes. GSS is Gandin and Murphy's equitable skill score (Gandin and Murphy 1992) with a scoring matrix derived to satisfy the constraints of symmetry and equitability as stated by Gandin and Murphy. To calculate GSS let S be an equitable scoring matrix whose calculation is given below and E be a matrix (contingency table) such that $e(i, j)$ is the relative frequency of instances with the true class i being classified as class j. From S and E, GSS = trace($S^{\mathsf{T}}E$). To calculate

S, following Gerrity (1992), let $P(r)$ be the relative frequency of class r. Using $P(r)$ define the following:

$$D(n) = \frac{1 - \sum_{r=1}^{n} P(r)}{\sum_{r=1}^{n} P(r)} \quad \text{, and}$$

$$R(n) = \frac{1}{D(r)} \quad .$$

Let K be the number of classes and $\kappa = \frac{1}{K-1}$.The elements of S are calculated as:

$$\begin{aligned}
s_{n,n} =& \kappa \left[\sum_{r=1}^{n-1} R(r) + \sum_{r=n}^{K-1} D(r) \right] && n = (1, 2, \cdots, K) \\
s_{m,n} =& \kappa \left[\sum_{r=1}^{m-1} R(r) + \sum_{r=m}^{n-1} (-1) + \sum_{r=n}^{K-1} D(r) \right] && 1 \leq m < K, m < n \leq K \\
s_{n,m} =& s_{m,n} && 2 \leq n \leq K, 1 \leq m < n
\end{aligned}$$

GSS varies from -1 to 1 (Gerrity 1992). A value of -1 indicates "intentionally" classifying incorrectly; a value of 0 is equivalent to the performance of a random classifier; and a value of 1 indicates perfect classification.

We considered a variety of skill metrics including Heidke's Skill Score (HSS) and Matthew's Correlation Coefficient (MCC). Previous work on SRPTs and SRRFs used True Skill Score (TSS) (aka Hanssen-Kuipers Discriminant, True Skill Statistic, and Peirces's Skill Score), however, we chose GSS to replace TSS as it is believed to be a better skill indicator without the bias introduced by rare-events that TSS exhibits (Jolliffe and Stephenson 2003; Marzban 1998). It is worth noting that GSS and TSS yield identical scores in the case of two classes. In addition to GSS, we use Area under the Receiver Operator Characteristic Curve (AUC) on domains that only contain two classes as AUC has a long, respected history (Bradley 1997; Egan 1975). AUC measures the robustness of the classifier across the various probability thresholds (Provost and Fawcett 2001).

To show the improvement from the addition of fields and new distinctions we also generate a non-fielded version of each dataset. This was done by reducing each field into four new attributes based on the minimum, maximum, mean, and standard deviation of the field. This is how the data would have been processed before the

addition of fields. We repeated the same parameter search over the non-field datasets as we did the fielded datasets. This allows for the comparison of the performance between fields and non-fields.

4.1 Points

Points is a synthetic dataset designed to test the new capabilities of SRPTs, particularly shape detection. Each graph is comprised of 3 to 10 objects each having two attributes: a discrete color and a 3D-field of integers from $\{0, 1\}$. The 3D-field is used to represent a point cloud with a 1 indicating a point at that location and 0 indicating empty space. A single relationship type is used with a random number of relations within each graph. To generate the data we randomly created a uniform distribution of class labels. Using the class labels we filled in the required number of objects with the correct attributes to meet the definition of each class label. Additional random objects were added to the graph along with relations that randomly related objects within the graph. These extra objects and relations served to add noise, forcing the SRRF to extract the relevant information from the domain. The full schema for Points is in Figure 4.1.

There are three classes in the Points dataset: *change*, *grow*, and *flip*. Each of the classes is based only upon color and/or the field, the relationships provide noise in the data and are not relevant for classification. *Change* has a box to turn into a sphere and cone to turn into a cylinder. *Grow* has three blue spheres that grow to have a volume greater than ≈ 167 cubic units (radius of 40 units). The last class *flip* has a cone that flips 180 degrees along its axis.

Figure 4.2 shows the performance of the SRRFs on Points for max depth of 5. Increasing the number of samples increases performance, however, there is an asymptotic behavior as adding more and more samples fails to increase performance. Increasing the number of trees also increases performance, though it quickly asymptotes as the addition of trees does not capture any additional features of the data. In Figure 4.3 we see that there is the expected drop between training data and testing

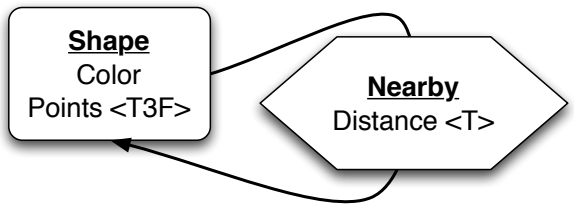


Figure 4.1: Schema for Points dataset: Attributes tagged <T> are temporal, <T3F> are temporal 3d fields, and untagged attributes are discrete.

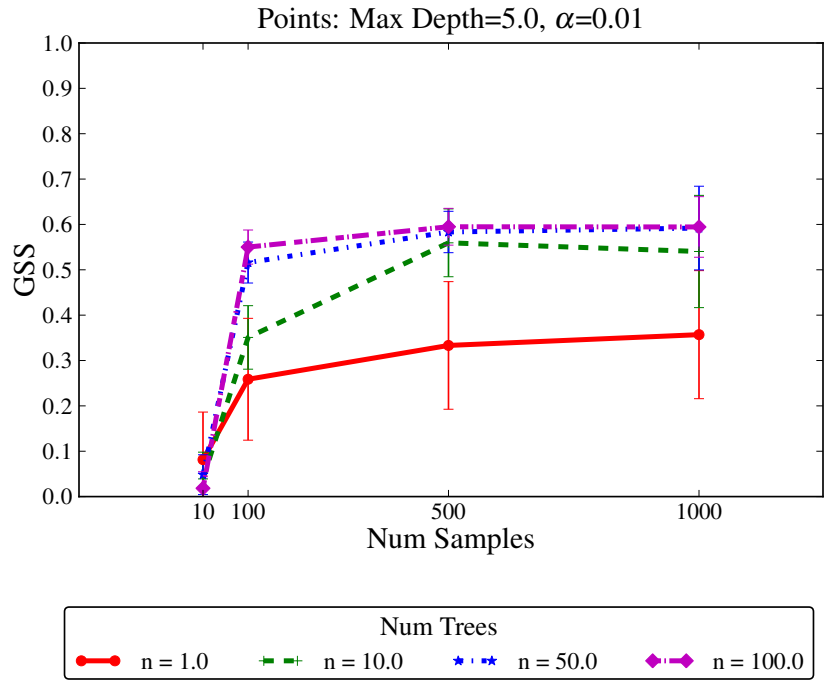


Figure 4.2: Testing set GSS on Points varying across K and N with D=5. Error bars are standard deviation.

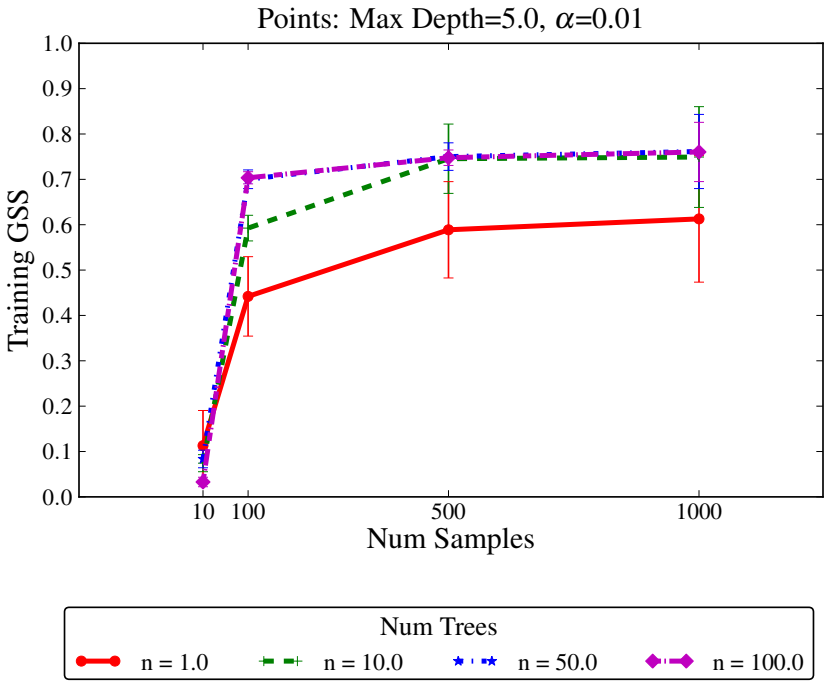


Figure 4.3: Training set GSS for Points varying across K and N with D=5. Error bars are standard deviation.

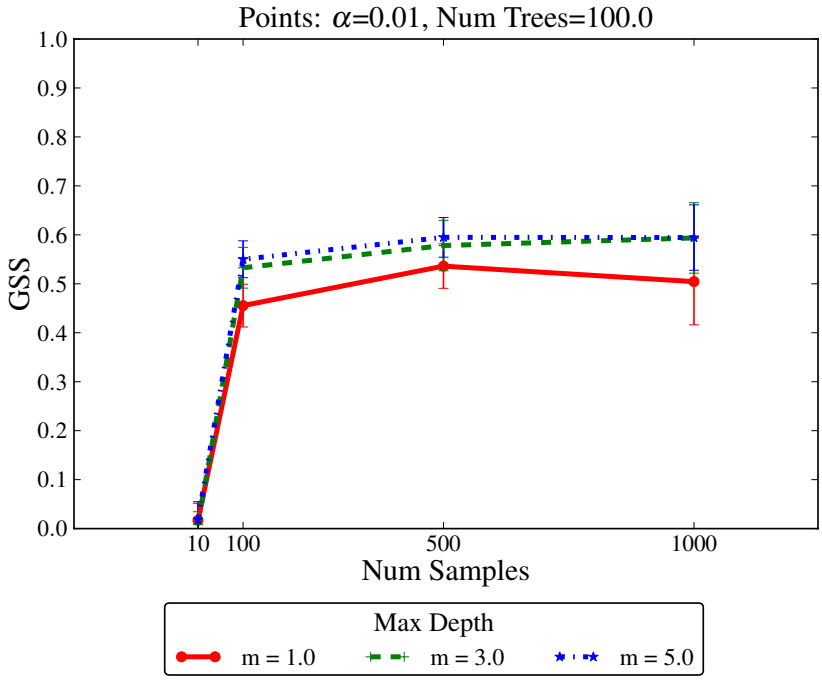


Figure 4.4: GSS for Points varying across K and D with N=100. Error bars are standard deviation.

data. However the change in GSS is relatively small from a training mean GSS of $\approx .7$ to a testing GSS of $\approx .6$. Figure 4.4 shows GSS of the SRRFs on Points illustrating how changing depth affects GSS. There is not much difference between depth 5 and 3 but between 3 and 1 there appears to be some difference. This trend exists in the other domains as well, hence we will only be looking at depth 5 results.

Table 4.1 contains N-way analysis of variance (ANOVA) for the factors number of samples (K), number of trees (N), and max depth (D) versus GSS. The first three columns shows the significance of single factor effects and the remaining four show interaction effects between parameters. Each of the three factors, K, N, D individually show a statistically significant effect on GSS, meaning that a change in the factor produces a statistically significant change in GSS. These follow common assumptions on decision trees and random forests. Increasing sampling should allow the SRRFs to find better distinctions and this is supported by statistics. Also increasing the number of trees and max depth should increases the performance by allowing the SRRFs to capture more complex concepts. Again, the effects of each of these parameters is supported by the statistics.

GSS	N	K	D	N×K	N×D	K×D	N×K×D
	≈ 0	**≈ 0**	**≈ 0**	**≈ 0**	0.541	**4.98e-4**	**0.004**

Table 4.1: N-Way ANOVA for Points Across all parameters. Values in bold are significant with $\alpha = 5\%$. **≈ 0** implies a value <= 0.

The interaction between number of samples and number of trees means that varying the two together produces more of a change in GSS than varying either one alone. The interaction effect between samples and depth is also significant. While ANOVA reports that increasing the number of trees (and number of samples) is statistically significant, if we look at a single slice, such as the right-most set of points in Figure 4.2. It is questionable if varying the number of trees for 1000 samples and max depth 5 produces a significant change in GSS. To test if there is a difference we performed an unpaired bootstrap randomization test with 10,000 resamplings to compare the mean GSS. The results are shown in Table 4.2. The tests indicate that there is a statistically significant difference between 1 and 10 trees, between 10 and 50 trees, but not between 50 and 100 trees. We also applied the bootstrap randomization to varying the number of samples at 100 trees. It shows a significant difference between 10 to 100 samples and 100 to 500 samples, but no difference between 500 and 1000 samples.

Table 4.3 contains the distinction counts and the counts of the attributes used by the distinctions. From the table we can see that the SRRFs are mostly picking distinctions that make sense give the definition of each class. *Count conjugate* is the most frequently chosen distinction. Counting is an important element of the definition of the grow class. The second most frequently chosen distinction is *Attribute.Exact*. The only attribute it selects the shape's color. Color is critical for successful classification,

Number of Trees	1 and 10	10 and 50	50 and 100
P-Value	**≈ 0**	**0.003**	0.819
Number of Samples	10 and 100	100 and 500	500 and 1000
P-Value	**≈ 0**	**0.003**	0.956

Table 4.2: Unpaired Bootstrap Randomization on GSS for K=1000 with D=5 between various N; and for N=100 with D=5 between various K . Values in bold are significant with $\alpha = 5\%$.

	% Total	Total	Shape.Points	Shape.Color
% Total	100		32	15
Total		398531	125699	60422
CountConjugate	20	78484		
Attribute.Exact	10	39879		39879
Shapes.Tilt	9	34195	34195	
Field.Gradient	7	28660	28660	
Attribute.LessThan	5	20543		20543
Shapes.Match	5	18766	18766	
FieldAttribute.Gradient	4	16927	16927	
Shapes.Change	4	16614	16614	
Exist	4	15783		
TemporalExists	3	13274		

Table 4.3: Distinction counts for the top 10 distinctions and top 2 attributes.

so the frequent use of matching color makes sense. *Shapes.Tilt*, *Shapes.Match* and *Shapes.Change* also show up as we would expect given the definition of each class. Overall the SRRFs do a decent job of predicting this highly spatiotemporal domain.

The Points domain is focused on the ability of SRRFs to reason with fielded objects and should be quite difficult to perform well on with the fielded objects reduced to values. Figure 4.5 shows the results of training SRRFs on the non-fielded Points. The resulting performance shows some ability to learn the classification problem despite the loss of fields. However, having the fields in the data does produce statistically significantly higher GSS, according to an unpaired bootstrap randomization test with 10,000 resamplings, with a p-value of $\approx \mathbf{0}$[1]. The variable importance for non-fields shows that *shapes.color* is by far the most important attribute. The next closest attribute's variable importance isn't statistically significantly different than 0. After examining some of the trees and returning to the definition of the classes, the requirement of one class having 3 blue spheres makes that class predictable even

[1]Despite the algorithm for bootstrap randomization returning $\approx \mathbf{0}$, with 10,000 resamplings the strongest statement that can actually be made about p is $p < 10^{-4}$

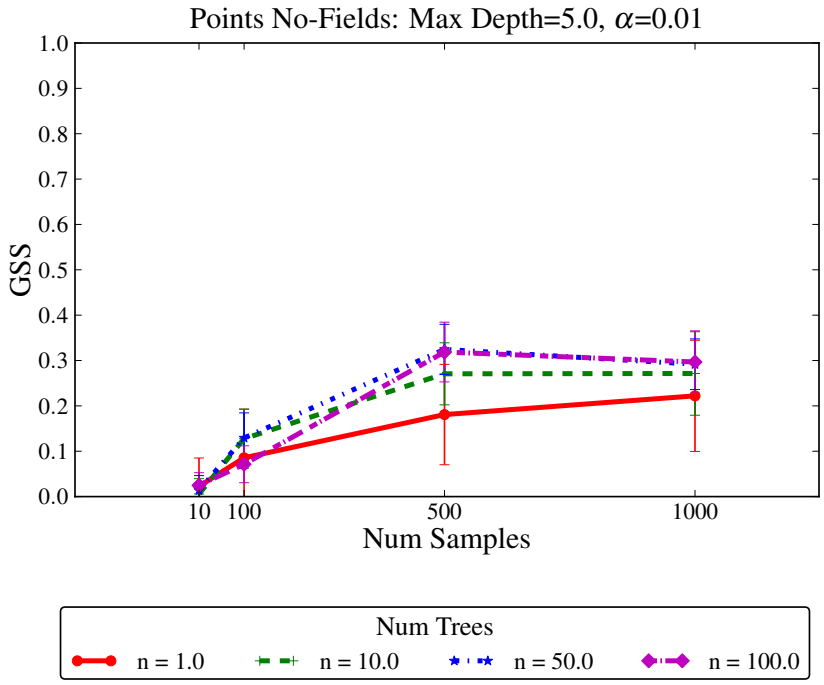


Figure 4.5: Testing set GSS for No-Fields Points varying across K and N with D=5. Error bars are standard deviation.

without the ability to check the other requirement of the class definition. Despite attempts to hinder the SRRF it was able to do moderately well. From this we can see that humans easily overlook aspects of a domain they believe are not important to the classification task. However, the SRRF algorithm considers all the information provided without the preconceived notations of importance. Moreover, SRRFs can sometimes find utility in the "unimportant" data.

4.2 Turbulence[2]

Convectively-induced turbulence (CIT) presents a major hazard for aviation. CIT is atmospheric turbulence in and around thunderstorms that frequently causes flight delays, routing changes and bumpy rides for passengers. In addition, the turbulence encountered by aircraft can cause structural damage to the aircraft, serious injuries and even fatalities. Providing better information about probable locations of turbulence to airline dispatch, air traffic control, and the pilots will enable them to avoid unnecessary ground delays and plan efficient routes that mitigate or avoid turbulence encounters. The National Center for Atmospheric Research includes better convectively-induced turbulence prediction as one of its goals in its current 2009-2013 RAL Strategic Plan. [3]

The Graphical Turbulence Guidance (GTG) system is the current system used for the forecasting of turbulence over the US (Sharman et al. 2006). It was developed by the FAA's Aviation Weather Research Program and currently runs at NOAA's Aviation Weather Center. The algorithm employed by the GTG system uses a combination of turbulence "diagnostic" quantities derived from a numerical weather prediction model that forecasts on a three dimensional grid. The Richardson number is an example of an diagnostic quantity used by the models. It measures the ratio of atmospheric stability to wind shear with low values suggesting the transition from

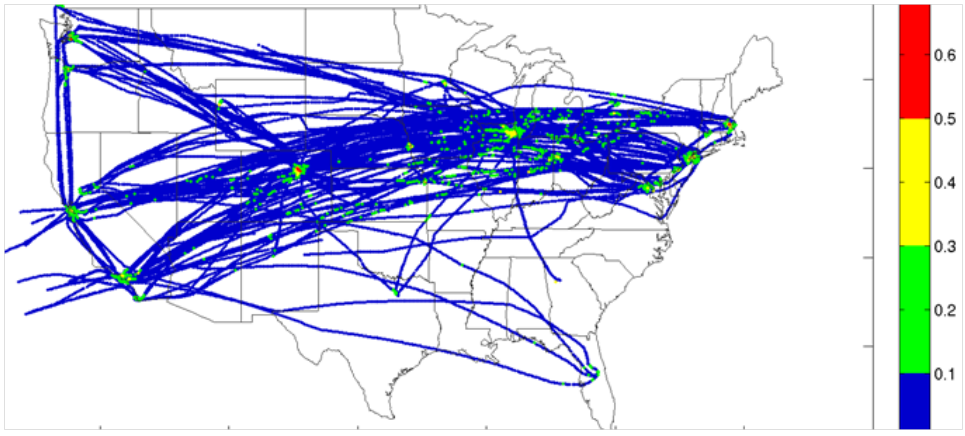


Figure 4.6: Map of eddy dissipation rate (EDR) collected by the automated turbulence reporting system on airplanes over a 24 hours period. [4]

[2]Description of Turbulence domain adapted from (McGovern et al. 2010)
[3]http://www.ral.ucar.edu/strategic_plan/2009/goal_aap.php
[4]Image courtesy of the National Center for Atmospheric Research

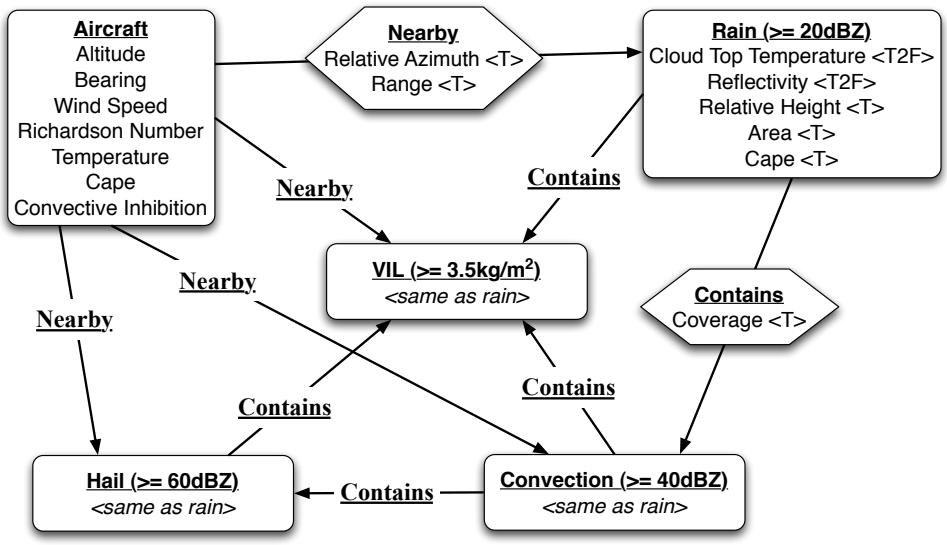


Figure 4.7: Schema for turbulence dataset: Attributes tagged <T> are temporal and <T2F> are 2D temporal fields, untagged attributes are scalars.

laminar to turbulent flow (Wallace and Hobbs 2006). Unfortunately, the 3-D grid of the numerical model uses for computatoin is too course to resolve thunderstorms making it incapable of fully capturing the mechanics of CIT generation.

Some modern commercial aircraft are fitted with automated turbulence reporting systems. These are used to augment the forecasts of the numerical models by associating objective atmospheric turbulence measurements with the models and other observations. By using rapid measurements of the aircrafts vertical acceleration, the automated turbulence reporting system can deduce atmospheric winds and use statistical analysis of the wind fluctuations to determine the intensity of the current turbulence. The turbulence of in-flight aircraft is measured in terms of eddy dissipation rates (EDR) over 1-minutes segments. Figure 4.6 shows a map of the EDR measurements over a single 24-hour time period. Note areas of turbulence are visible, such as over the Rocky Mountains.

Data were combined from aircraft reports, archived weather observations, and archived real-time weather models to create the dataset used for this experiment. The aircraft reports we used were collected from United Airlines Boeing 757 aircraft in the summer of 2007. Summer was chosen due to the increase of convection during the season. This helps in isolating convection as the source of turbulence. To generate useable data, the raw collected data were transformed into a spatiotemporal relational representation detailed by the schema in Figure 4.7. Using the in-situ aircraft data

and the NWP gridded model and observation data, we formed objects that represent meteorological concepts or distinct regions. Thresholds were applied to the radar reflectivity data generating connected areas with rain being represented by areas greater than 20dBZ, convection greater than 40dBZ, and hail greater than 60dBZ.

The natural rarity of CIT and skilled pilots whose job includes avoiding turbulence make aircraft experienced turbulence an especially rare event. This meant that much of the data collected contained instances of null or light turbulence with only 1% of the collected data reporting light-to-moderate or greater (LMOG) turbulence. Rare events can complicate learning and is a common hurdle in machine learning (Provost and Fawcett 2001). We under-sampled the data to combat the effects of rare events to bring the percentage of LMOG turbulence to 26% of the 2055 instances. The final class distribution is shown in Table 4.4.

Total	No	Yes
2055	1514 (74%)	541 (26%)

Table 4.4: Turbulence class distribution.

We performed the full parameter search using SRRFs producing Figure 4.8 that shows an AUC of 0.8236 at 100 trees 1000 samples. Figure 4.9 shows a mean GSS of 0.279 with 100 trees, 100 samples, and depth 5 . Comparing the GSS on the training set to the performance on the testing set, Figure 4.10, we see a drop in GSS down to 0.441 that could indicate some over fitting.

ANOVA interestingly indicates that all parameters are statistically significant for affecting GSS. It also shows strong interaction effects between all variables. Focusing on a particular set of parameters, it appears that there is no difference between 10, 50, and 100 trees at 1000 samples. Performing an unpaired bootstrap randomization test with 10,000 resamplings to compare the mean GSS, shown in Table 4.6, confirms there is no statistically significant difference. The test also confirms an asymptote

GSS	K	N×D	N	D	K×D	N×K	N×K×D
	0.0118	**≈ 0**	**≈ 0**	**8.71e-6**	**9.13e-5**	**≈ 0**	**2.52e-8**

Table 4.5: N-Way ANOVA on Turbulence GSS. Values in bold are significant with $\alpha = 5\%$.

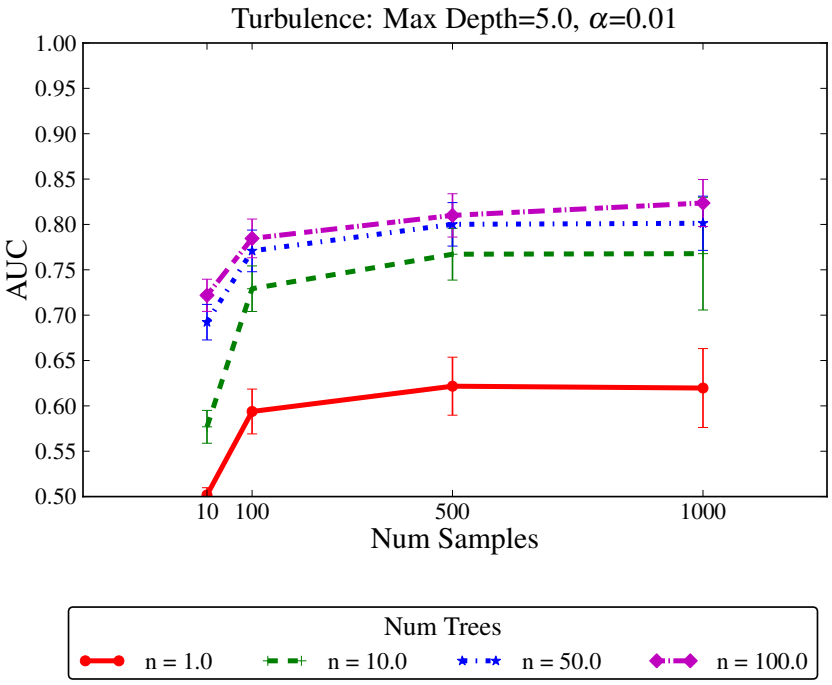


Figure 4.8: AUC for the testing set for turbulence across K and N with D=5. Error bars are standard deviation.

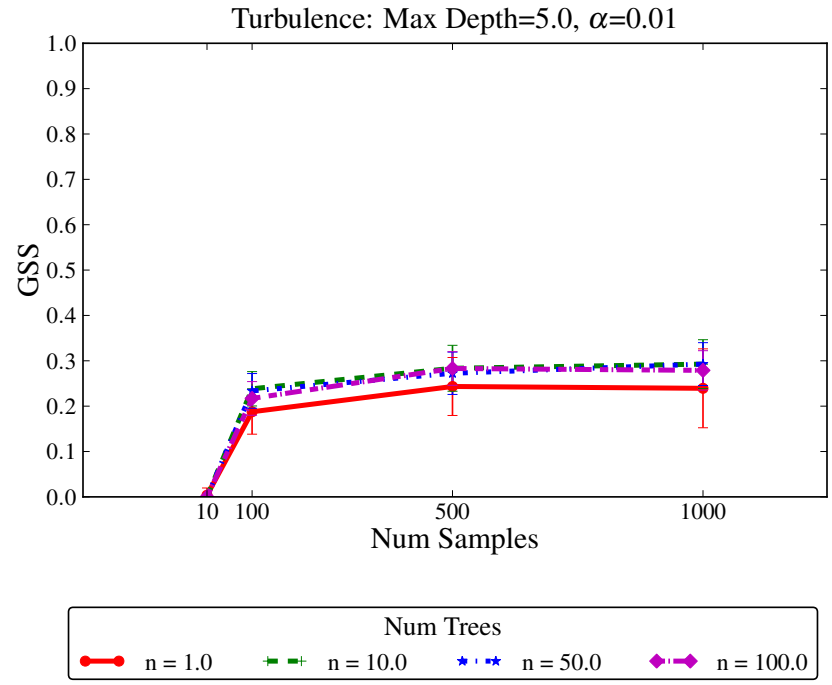


Figure 4.9: GSS for the testing set for turbulence across K and N with D=5. Error bars are standard deviation.

Number of Trees	1 and 10	10 and 50	50 and 100
AUC	$\approx$ **0**	$\approx$ **0**	$\approx$ **0**
GSS	**5e-04**	0.996	0.164

Number of Samples	10 and 100	100 and 500	500 and 1000
AUC	$\approx$ **0**	$\approx$ **0**	**0.011**
GSS	$\approx$ **0**	$\approx$ **0**	0.687

Table 4.6: Unpaired Bootstrap Randomization on GSS for K=1000 with D=5 between various N; and for N=100 with D=5 between various K . Values in bold are significant with $\alpha = 5\%$.

when increasing the number of samples. Going from 10 to 100 samples on 100 trees and depth 5 is significant, but 100 to 500 and 500 to 1000 is not significant.

By reducing the fields in the dataset into temporal-scalars as described at the beginning of Chapter 4, we can determine if adding fields increased the performance of SRRFs on Turbulence. Figure 4.11 shows GSS for turbulence with no fields. The

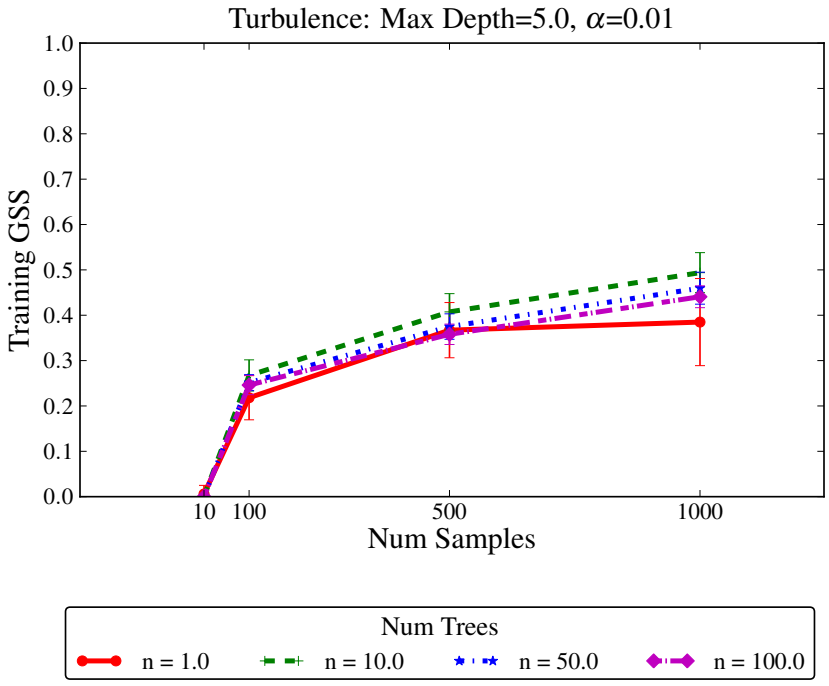


Figure 4.10: GSS on training set for turbulence across K and N with D=5. Error bars are standard deviation.

resulting GSS appears identical between them. An unpaired bootstrap randomization test with 10,000 resamplings to compare the mean GSS determines there is no statistically significant difference between the two with a p-value of 0.943. Comparing AUC using the same test gives the same result with no significant difference with a p-value of 0.069.

Variable importance (Table 4.7) gives no indication that the SRRF focused on the same non-field attributes between the two datasets. Only one non-field attribute, the plane's Richardson number, shows up as being significantly important for both fields and non-fields. This makes sense as the Richardson number is a measure of flow stability. As the atmosphere becomes unstable it tends to produce turbulent air flow instead of smooth laminar flow.

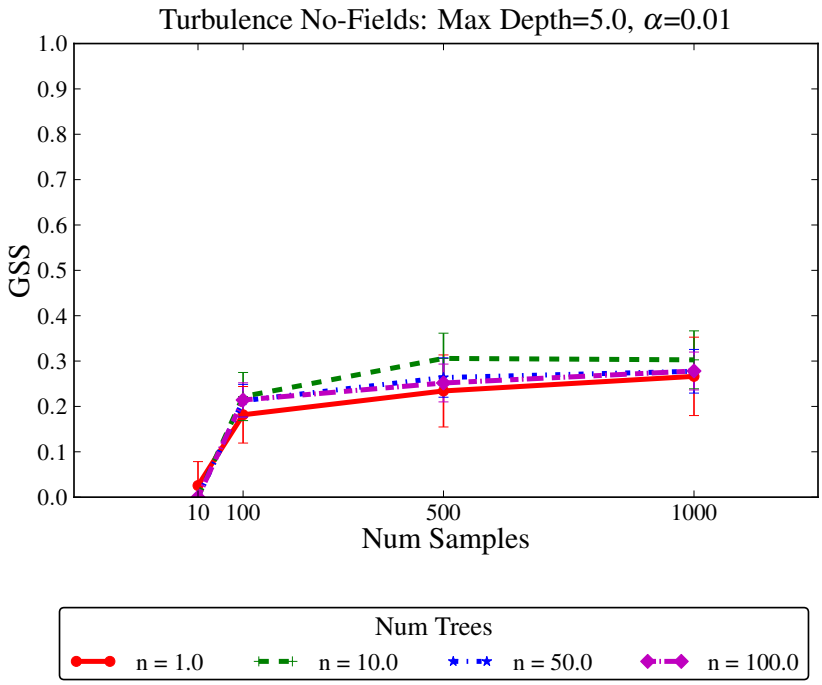


Figure 4.11: GSS for Turbulence without Fields varying across K and N with D=5. Error bars are standard deviation.

Fields	Mean Variable Importance	Non-Fields	Mean Variable Importance
Vil.Area	0.198	Plain.ConvectionInhibition	0.153
Plane.RichardsonNumber	0.106	**Plane.RichardsonNumber**	0.128
Rain→Contains.Coverage→Vil	0.085	Conv.StdDev_CloudTopTemperature	0.124
Rain→Contains.Coverage→Conv	0.084	Conv.Area	0.110
Conv→Contains.Coverage→Vil	0.061	Conv.Max_CloudTopTemperature	0.105
Rain.Area	0.056	Plane→Nearby.Azimuth→Conv	0.101
Hail.CloudTopTemperature	0.054	Conv.Mean_Reflectivity	0.090
Plane→Nearby.Range→Rain	0.053	Vil.StdDev_Reflectivity	0.081
Vil.CloudTopTemperature	0.052	Rain.StdDev_Reflectivity	0.080
Plane→Nearby.Range→Vil	0.048	Conv.StdDev_Reflectivity	0.071

Table 4.7: Top 10 statistically significant ($\alpha = .05$) attributes according to variable importance on the Turbulence dataset. Non-field attributes present in both are shown in bold. Non-field attributes the name of the field is prefixed with the statistic used in the reduction from a field to a scalar, e.g. Vil.StdDev_Reflectivity is the standard deviation of the Vil's Reflectivity field.

		Total	Conv.CloudTopTemp	Rain.Reflectivity	Vil.Reflectivity	Rain.CloudTopTemp	Vil.CloudTopTemp	Conv.Reflectivity	Rain.AcRelHeight	Plane.Altitude	Vil.AcRelHeight	Plane.Temp
	% Total	100	7	6	6	6	6	5	5	4	4	4
All	Total	110746	7851	7043	6629	6333	6316	5998	5642	4799	4625	3974
	Attribute.LessThan	12659.0	–	–	–	–	–	–	–	3356	–	2530
	Field.Gradient	10367.0	1645	2588	1566	1460	1175	1610	–	–	–	–
	Attribute.Min	8071.0	613	175	138	402	337	142	1270	–	1081	–
	Attribute.Median	7889.0	418	168	212	411	333	176	1129	–	879	–
	Attribute.Mean	7837.0	534	139	173	395	313	172	1186	–	826	–
	Attribute.Mode	7685.0	342	341	423	314	344	397	187	–	216	–
	CountConjugate	6825.0	410	326	465	283	381	375	370	242	292	206
	Attribute.Max	6510.0	464	104	118	210	280	135	696	–	542	–
	Attribute.Exact	6133.0	–	–	–	–	–	–	–	676	–	888
	FieldAttribute.Gradient	5726.0	1143	889	864	740	878	843	–	–	–	–

Table 4.8: Top 10 Distinctions and Attributes/Objects/Relations for Turbulence with 100 trees, 1000 samples, depth 5 set of trees.

4.3 Fronts[5]

Frontal movements are theorized to play a major role in the formation of tornadoes. As air masses move across the land and meet, they form boundary regions such as along a warm front or cold front. With the collision comes the mixing of the air masses. This mixing is not instantaneous but forms transition zones that can produce regions of strong temperature and moisture gradients. Formation of boundaries can also occur along dry-lines or along outflow regions from thunderstorms. Boundaries from warm, moist air lifting up and over cool, dry air is frequently associated with generation of thunderstorms. However, the role of such boundaries play in tornadogenesis is not well understood. These boundaries have been shown to yield zones of enhanced horizontal rotation that can be titled by a strong updraft present in a supercell thunderstorm moving through the region. This tilting assists the production of the tornadoes. A study performed by Markowski et al. (1998) on supercell thunderstorms over a one-year period found that 70% occurred near frontal boundaries. Due to the limited sample size and short time period additional study over a longer period is needed in order to quantify the boundary-tornado relationship.

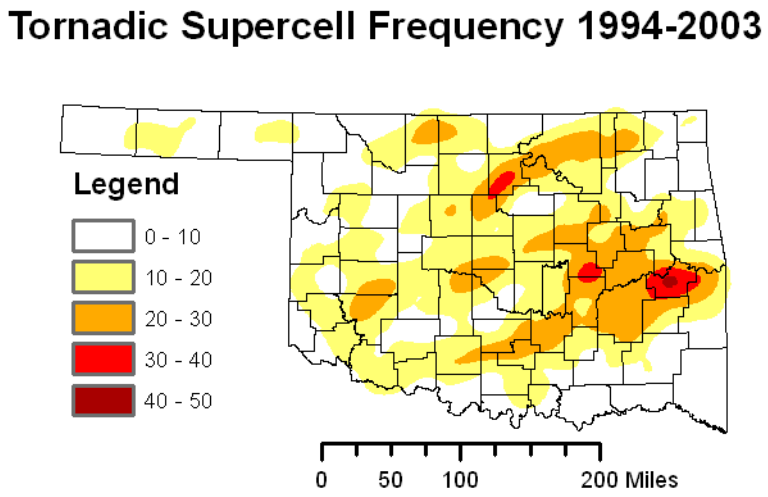


Figure 4.12: Frequency of tornadic supercells within 30km from 1994-2003.

[5]Description of Fronts domain adapted from (McGovern et al. 2010)

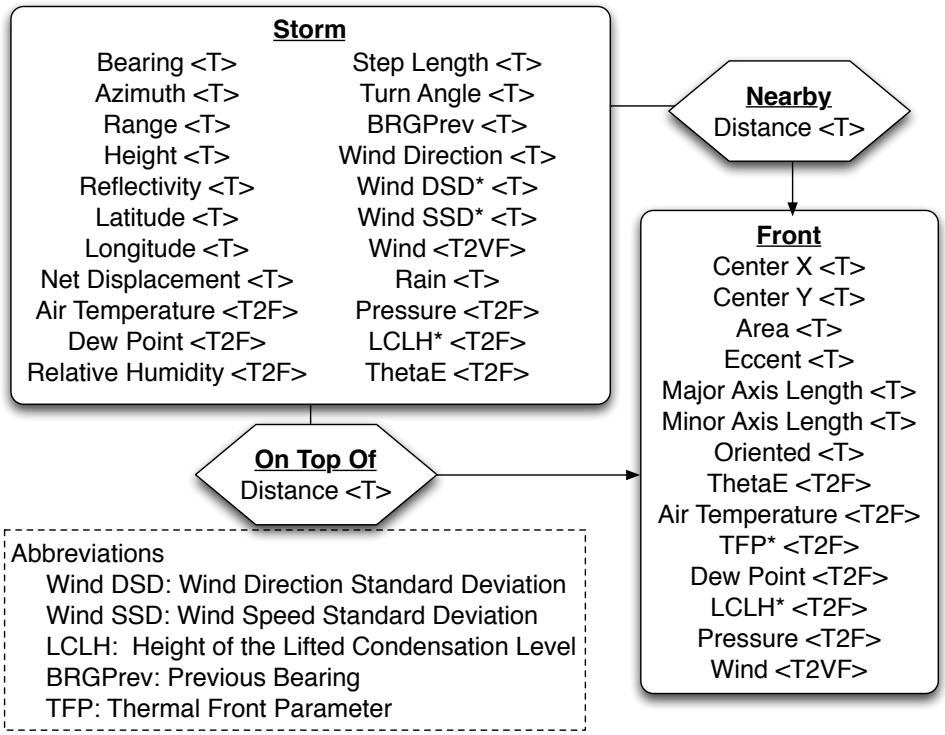


Figure 4.13: Schema for Fronts dataset: Attributes tagged <T> are temporal, <T2F> are temporal 2D fields, and <T2VF> are temporal 2D vector fields.

Our data were created by the combination of two sources, one providing the supercells and the other providing the data need for extracting frontal boundaries. The supercells come from a nine year study performed by Hocker and Basara from 1994–2003 of 950 Oklahoma supercells (Hocker and Basara 2008). Frontal boundaries associated with each supercell were extracted from Oklahoma Mesnot observations (McPherson et al. 2007) using objective front analysis (Jenker et al. 2005; Renard and Clarke 1965). Each supercell and frontal boundaries group was labeled based on if it produced a tornado. Figure 4.13 shows the schema used for our experiments. The schema details the objects, relations, and attributes provided to the SRRF algorithm. A Nearby relationship indicates a storm and a front are less than 40km apart. On-Top-Of indicates the distance between them is less than 10km. Figure 4.12 shows the spatial distribution of the supercells across Oklahoma. Table 4.9 gives the class distribution in the Fronts domain. [6]

Total	No	Yes
926	711 (77%)	215 (23%)

Table 4.9: Fronts class distribution.

Fronts achieves a maximum mean AUC of 0.7265 at 100 trees, 1000 samples, and max depth of 5. Figure 4.14 shows GSS across number of trees and number of samples showing what appears to be well graduated differences between parameter sets. An unpaired bootstrap randomization test with 10,000 resamplings to compare the mean GSS is shown in Table 4.11. The test shows there is a statistically significant difference between 1 and 10 trees and between 10 and 50 trees both at 100 samples. Comparing the number of samples from 10 to 100 with 100 trees also yields a statistically significant difference.

The low mean GSS shown in Figure 4.15 indicates that the Fronts domain is difficult for the SRRFs, with a maximum mean GSS of 0.149. Examining the results from the testing set shown in Figure 4.16, we see that the mean GSS is 0.493. This is a fairly large drop between training and testing sets, and is most likely evidence of over fitting. The ANOVA shows that each factor is important in affecting the GSS achieved by SRRFs on Fronts. SRRFs with 10 trees have a statistically significant higher GSS than other number of trees. Also increasing the number of samples does not statistically asymptote until 500 samples.

[6]Description of domain adapted from (McGovern et al. 2010)

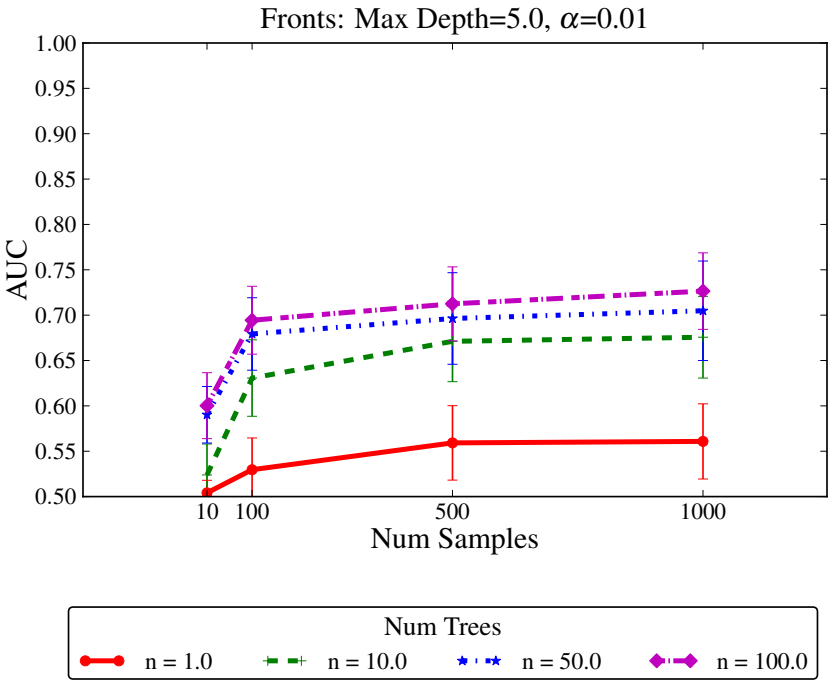


Figure 4.14: AUC for Fronts varying across K and N with D=5. Error bars are standard deviation.

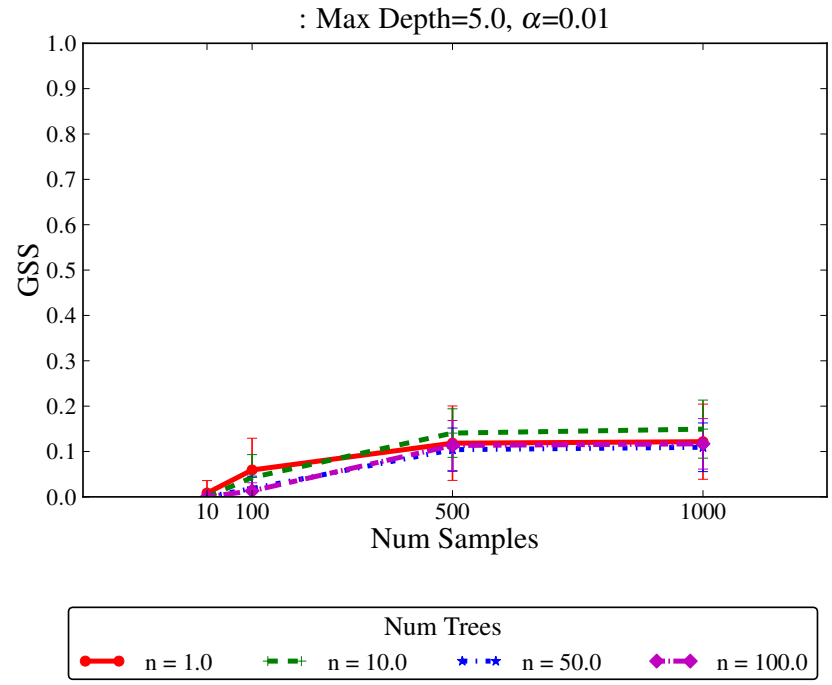


Figure 4.15: GSS for Fronts varying across K and N with D=5. Error bars are standard deviation.

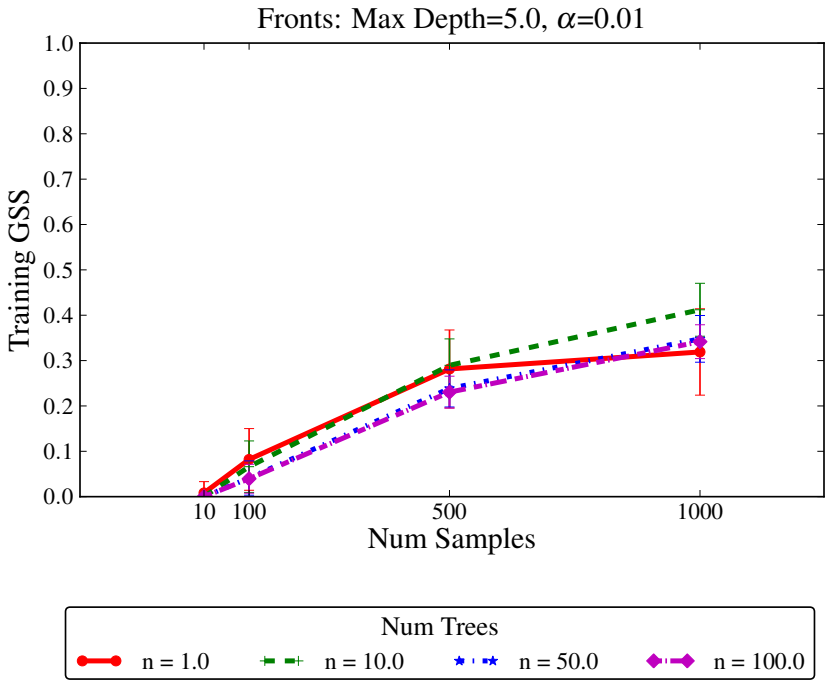


Figure 4.16: Training GSS for Fronts varying across K and N with D=5.

GSS	K	N×D	N	D	K×D	N×K	N×K×D
	1.19e-14	**≈ 0**	**2.65e-15**	**2.54e-5**	0.152	**2.84e-6**	0.898

Table 4.10: N-Way ANOVA on Fronts GSS. Values in bold are significant with $\alpha = 5\%$.

Number of Trees	1 and 10	10 and 50	50 and 100
AUC	**≈ 0**	**0.025**	0.085
GSS	0.129	**0.004**	0.577

Number of Samples	10 and 100	100 and 500	500 and 1000
AUC	**1.38e-14**	0.089	194
GSS	**4e-4**	**≈ 0**	0.798

Table 4.11: Unpaired Bootstrap Randomization on GSS for K=1000 with D=5 between various N; and for N=100 with D=5 between various K . Values in bold are significant with $\alpha = 5\%$.

All													
	% Total	100		6	5	5	5	5	5	4	4	4	4
	Total		67116	3755	3615	3314	3273	3096	3038	2862	2858	2845	2799
	Attribute.Mode	12	7830.0	245	210	228	264	–	203	–	204	263	239
	FieldAttribute.Gradient	10	6967.0	1007	998	545	564	–	557	–	545	616	411
	Field.Gradient	9	6155.0	308	478	424	562	–	460	–	524	622	381
	Attribute.Median	8	5129.0	289	195	308	139	–	199	–	161	56	317
	TemporalExists	7	4870.0	–	–	–	–	2502	–	2359	–	–	–
	Attribute.Mean	7	4827.0	297	128	258	148	–	147	–	127	59	234
	Attribute.Max	7	4816.0	156	165	200	170	–	181	–	79	70	177
	Attribute.StdDev	7	4662.0	163	170	83	100	–	102	–	100	84	85
	Attribute.Min	7	4654.0	109	126	159	102	–	97	–	162	75	101
	CountConjugate	4	2781.0	154	131	147	103	133	108	117	100	131	96

Table 4.12: Top 10 Distinctions and Attributes/Objects/Relations for Fronts with 100 trees, 1000 samples, depth 5 set of trees.

Fields	Mean Variable Importance	Non-Fields	Mean Variable Importance
Storm.AirTempature	0.162	**Storm.NetDisplacement**	0.323
Storm.ThetaE	0.130	**Storm.Bearing**	0.233
Storm.NetDisplacement	0.120	Front.CenterY	0.202
Storm→Nearby.RelativeAzimuth →Front	0.117	**Storm →Nearby.RelativeAzimuth →Front**	0.196
Storm.DewPoint	0.112	*Front.Max_ThetaE*	0.195
Storm.Bearing	0.109	Front.Max_DewPoint	0.164
Front.ThetaE	0.088	Front.Min_Wind	0.155
Front.ThermalFrontParameter	0.085	Front.StdDev_Pressure	0.137
Storm.Pressure	0.085	Storm.Latitude	0.129
Storm.LiftedCondensationLevelHeight	0.079	Storm.Longitude	0.125

Table 4.13: Top 10 statistically significant ($\alpha = .05$) attributes according to variable importance on the Fronts dataset. Non-field attributes present in both are shown in bold and field attributes in both are shown in italics. For the non-field attributes the name of the field is prefixed with the statistic used in the reduction from a field to a scalar, e.g. Front.Max_DewPoint is the maximum of the Front's Dew Point field.

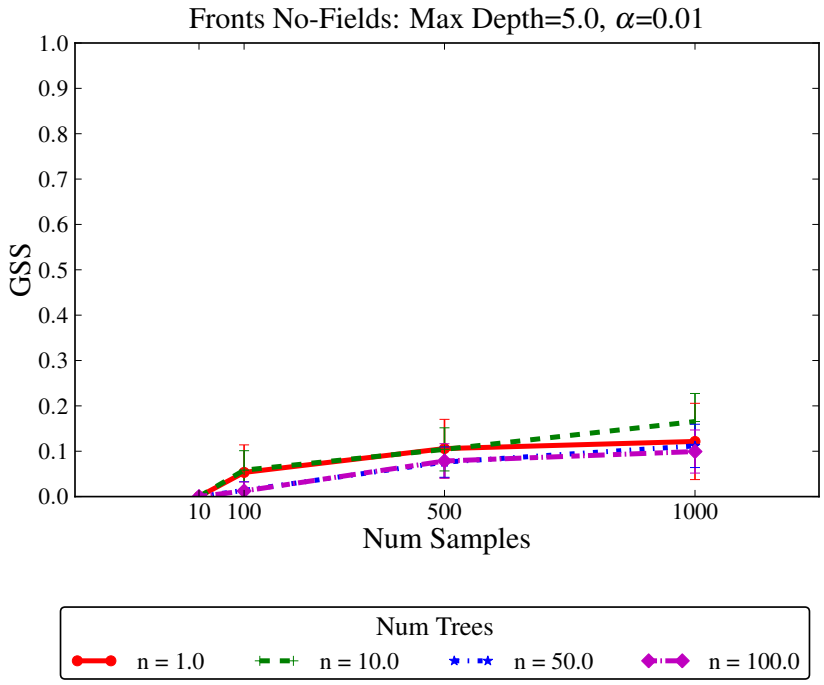


Figure 4.17: GSS for Fronts without Fields varying across K and N with D=5. Error bars are standard deviation.

As before, we repeated the full parameter search on Fronts without fields. Figure 4.17 shows GSS without fields. Comparing the plots with and without fields, it appears that there is little to no difference. Calculating ANOVA confirms this as shown in Table 4.10. The p-value for the comparison of all parameter sets is 0.167. In addition we performed a unpaired bootstrap randomization test with 10,000 resamplings. The bootstrap randomization was limited to only forests of 100 trees at 1000 samples with a maximum depth of 5 and yields a p-value of 0.196. Comparing AUC with the name test between fields and non-fields also shows no statistically significant difference with a p-value of 0.473.

Our hypothesis that adding fields would increase the GSS obtained by SRRFs fails for the Fronts domain. One possible explanation is that the most useful variables are not fields, hence they exist with the same data in both the fielded and non-fielded datasets. Calculating variable importance will give an insight into the most important attributes and perhaps contain some indication of the poor performance. Table 4.13 shows that 6 of 10 of the most important attributes are non-fields. *Storm→Nearby→Front* is our notation for indicating a storm object is related to a front object via a nearby relation. And *Storm→Nearby.RelativeAzimuth→Front* is

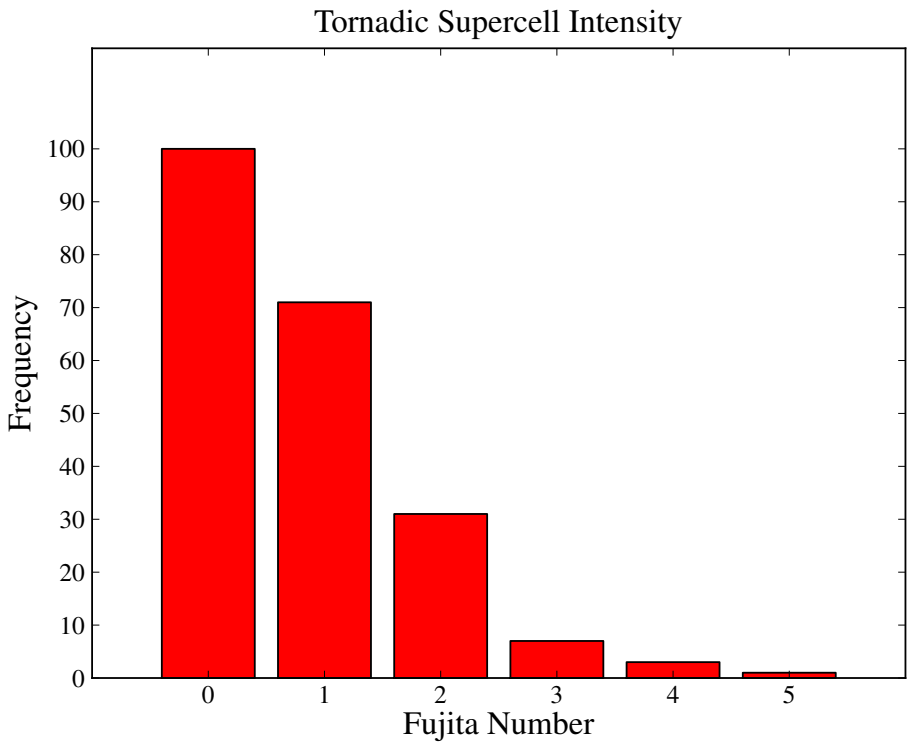


Figure 4.18: Frequency of Tornadoes in the Fronts Data by Fujita Scale. [7]

the relative azimuth attribute of the relation. Several of the attributes also show up in both the field and non-field versions.

Variable importance (Table 4.13) fails to show any strong evidence of why there is no statistically significant difference between the field and non-field versions. Given the very poor performance of the SRRFs on both versions a possible explanation is that the needed data to do a good job is simply not present in the data. Another possibility is that the distribution of tornadoes by intensity being biased towards F0 tornadoes makes the task more difficult. Figure 4.18 shows the distribution of tornados by Fujita number in the Fronts data. Since the wind speeds in an F0 tornado, a tornado with maximum wind speeds less than 73 mph (33 m/s), are not much different than severe thunderstorm winds, winds greater than 58 mph (25 m/s), it is difficult to distinguish features of the storm and its environment that produce each of these phenomena. Better results might be achieved by counting F0's and/or F1's as negatives. F2 and greater are considered "significant" tornadoes in meteorology and the focus is frequently placed on predicting them.

[7]Plot courtesy of David John Gagne II

4.4 Tornado

Supercell thunderstorms can create several serious hazards including hail 4 inches in diameter and winds in excess of 100 miles per hour. However, the potentially most dangerous and damaging are tornadoes. Tornadoes cause millions of dollars of damage every year and can even result in loss of life. In 2008, tornadoes caused 126 deaths (Storm Prediction Center 2009). The past decade has shown a great improvement in the ability to predict supercell formation and movement. Despite these improvements, the ability to predict if a given supercell will produce a tornado is still far less than satisfactory and poses a challenge to researchers (Markowski and Richardson 2009).

In order to predict tornadogenesis, we needed data of tornadic and non-tornadic storms. This was achieved by using a numerical simulation framework that could produce simulations of thunderstorms, and specifically supercells. The supercell data used was generated by Rosendahl (2008) using the Advanced Regional Prediction System (ARPS) model (Xue et al. 2003). Rosendahl's model used a 500 m horizontal grid spacing over a 100 km $\times$ 100 km domain with 50 vertical levels resulting in a domain of $200 \times 200 \times 50$ points. To produce the various simulations the initial conditions were varied across a set of parameter values for the surface mixing ratio and the shape of the hodograph in the sounding. The simulation was run for three hours with snapshots of the domain's state saved every 30 seconds. The saved data consists of 21 time series of $200 \times 200 \times 50$ grids of meteorological variables. Table 4.14 lists all 21 variables.

The application of SRRFs for prediction of tornadogenesis requires that the data be represented in a spatiotemporal relational manner and labeled. The supercell simulations were used to generate spatiotemporal relational attributed graphs representing storms from the simulations. Each simulation could produce multiple storms each of which was separated into its own graph and labeled independently. The gridded domain was broken into regions using thresholds on the meteorological variables that created objects of different types. Once the regions were determined the meteorological variables were saved with each object. The objects were then related based on spatial proximity. Figure 4.19 details each object type and the relations in the graphs we generated.

Storms were given one of three labels, *Yes*, *No*, *Maybe*, based on the occurrence of strong low-altitude rotations. Due to the 500m resolution of the simulations, it is

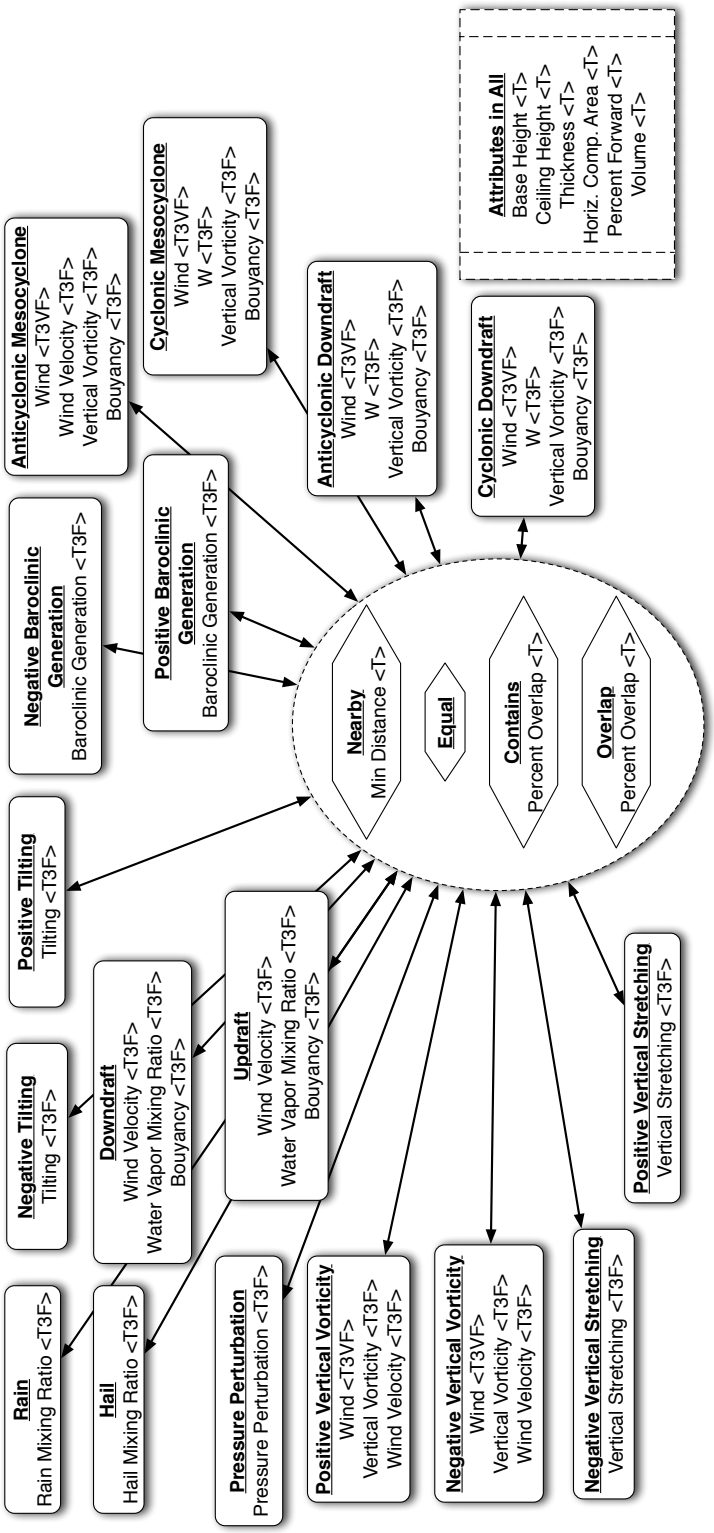


Figure 4.19: Schema for Tornado dataset: Attributes tagged <T> are temporal, <T3F> are temporal 3d fields, <T3VF> are temporal 3d vector fields. All objects can be related to each other via all relations and on the lower right are attributes present in all objects.

Water Vapor Mixing Ratio	Water Vapor Number Concentration
Cloud Water Mixing Ratio	Cloud Water Number Concentration
Rain Mixing Ratio	Rain Number Concentration
Ice Mixing Ration	Ice Number Concentration
Graupel Mixing Ratio	Graupel Number Concentration
Snow Mixing Ratio	Snow Number Concentration
Hail Mixing Ratio	Hail Number Concentration
Base Pressure	U Component of Wind
Pressure Perturbation	V Component of Wind
Base Potential Temperature	W Component of Wind
Potential Temperature Perturbation	

Table 4.14: Full list of extracted meteorological variables from the ARPS simulation that produced the data for the Tornado domain.

not possible to determine the existence of a tornado using the strict meteorological definitions of a tornado. A strong low-altitude rotation is said to occur in a storm if four things happen. First, there must be a pressure drop of 300 Pa within a five minute interval. Second, during that time period the minimum pressure perturbation value must be less than -900 Pa. Third, the pressure perturbation object under going the pressure drop must overlap, be contained by, or be equal to a vertical vorticity object. Lastly, those two objects must be below 2km. A storm passing all of these requirements is labeled as *Yes*, meaning it contains a strong low-altitude rotation, if at least two of the four conditions are met it is labeled as a *Maybe*, otherwise it is labeled *No*.

The end result produces storms with the class distribution in Table 4.15. However, this produced a very uneven distribution that is heavily weighted towards the negative. To produce a less biased distribution the negative storms were under-sampled. We tossed out all the storms in all ARPS simulations that only produced negative storms. From a meteorological view point an ARPS simulation that only produced negative storms is uninteresting, hence this was used instead of random under-sampling. This reduced the number of negative storms from 846 to 364 yielding the class distribution also in Table 4.15.

In Figure 4.20 we see that the SRRFs achieve a GSS of $\approx .4$ for 100 trees and 100 samples at depth 5. The ANOVA calculations in Table 4.16 show that each

	Total	No	Maybe	Yes
Original	1057	846 (80%)	124 (12%)	87 (8%)
Under-Sampled	575	364 (63%)	124 (22%)	87 (15%)

Table 4.15: Tornado class distribution with and without under-sampling.

parameter individually has a statistically significant effect on GSS. Table 4.17 shows that the appearance of 50 trees out performing 100 trees is not statistically significant. In fact 10, 50, and 100 trees are statistically the same by an unpaired bootstrap randomization test with 10,000 resamplings to compare the mean GSS, however, they all our statistically higher than 1 tree. Figure 4.21 shows only a slight drop of around 0.1 when moving from the training to the testing set that indicates over-fitting was most likely not an issue.

To determine if the new additions to SRRFs aided the SRRFs in their task we also performed the full parameter search on the tornado dataset after all fields were reduced. Figure 4.22 shows GSS for the non-fielded version of the tornado data. There is a small, but noticeable, drop in GSS between the fielded and non-fielded. Taking all

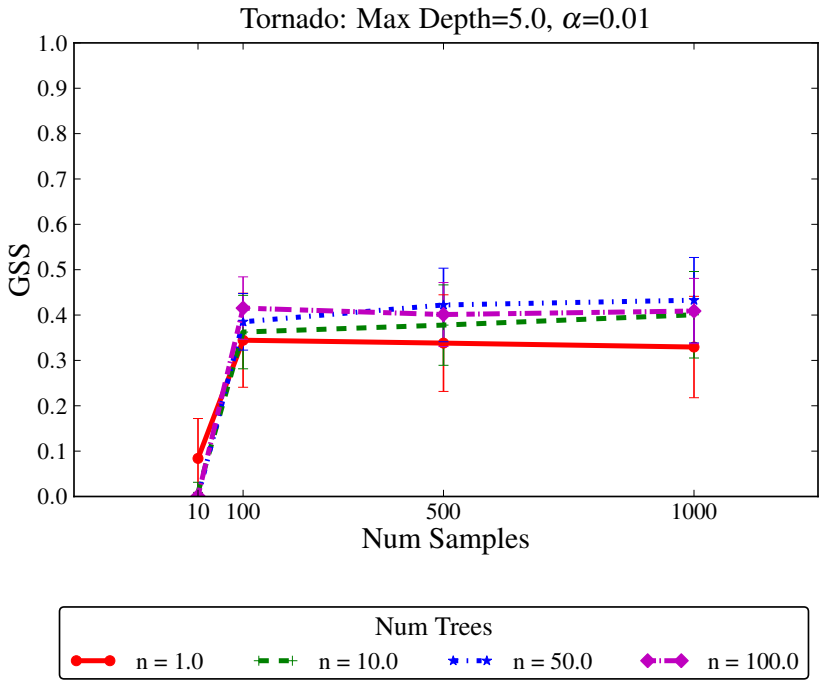


Figure 4.20: GSS for Tornado varying across K and N with D=5. Error bars are standard deviation.

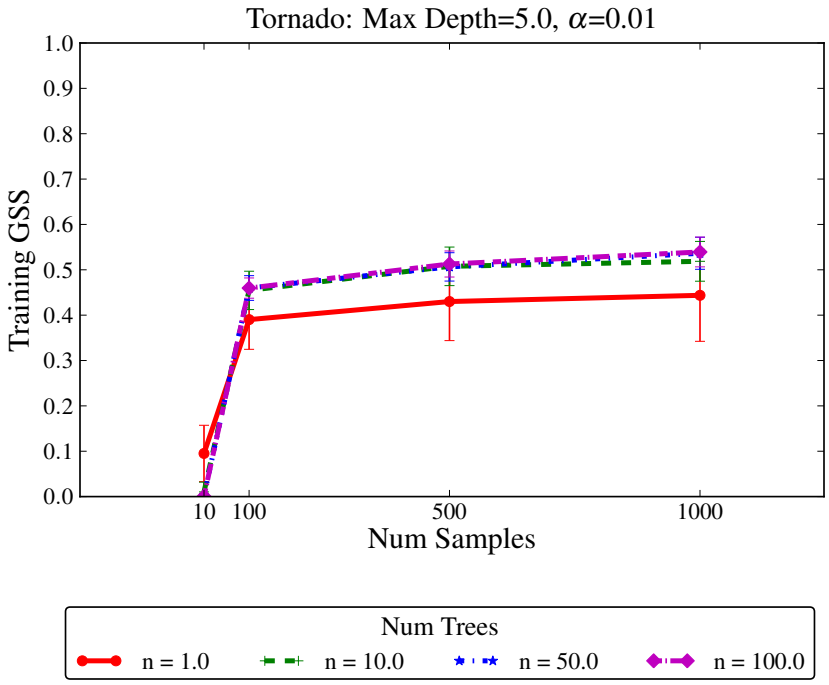


Figure 4.21: Training GSS for Tornado varying across K and N with D=5. Error bars are standard deviation.

the data together and running ANOVA indicates that there is a very strong statistical significance with a p-value of $\approx \mathbf{0}$[8] when switching between the two datasets. We also compared non-fields to fields on a single set of parameters of 100 trees, 1000 samples and max depth 5. An unpaired bootstrap randomization test with 10,000 resamplings still shows a strong significance with a p-value of 4.05e-5.

Table 4.18 lists the top 10 statistically significant attributes. Given the description of how strong low-altitude rotations are determined the inclusion of multiple attributes of the pressure perturbation object seems quite reasonable. Also, updrafts are responsible for the tilting of horizontally rotating air, formed by wind shear, into vertically rotating air, with respect to its axis of rotation. Table 4.19 (continued in Table 4.20) lists the distinction counts of the top 10 distinctions and top 10

GSS	K	N×D	N	D	K×D	N×K	N×K×D
	1.70e-7	$\approx \mathbf{0}$	$\approx \mathbf{0}$	$\approx \mathbf{0}$	0.772	$\approx \mathbf{0}$	0.786

Table 4.16: N-Way Anova on GSS. Values in bold are significant with $\alpha = 5\%$.

[8]Floating-point precession cannot represent values $\leq 2.2e-16$

Number of Trees	1 and 10	10 and 50	50 and 100
P-Value	**0.006**	0.095	0.172
Number of Samples	10 and 100	100 and 500	500 and 1000
P-Value	$\approx$ **0**	0.443	0.675

Table 4.17: Unpaired Bootstrap Randomization on GSS for K=1000 with D=5 between various N; and for N=100 with D=5 between various K. Values in bold are significant with $\alpha = 5\%$.

attributes. The counts of the picked attributes reflect the results of the variable importance. Figure 4.23 shows the best SRRF with one tree, i.e. an SRPT. In this example we can see several uses of the top attributes (by variable importance and distinction count).

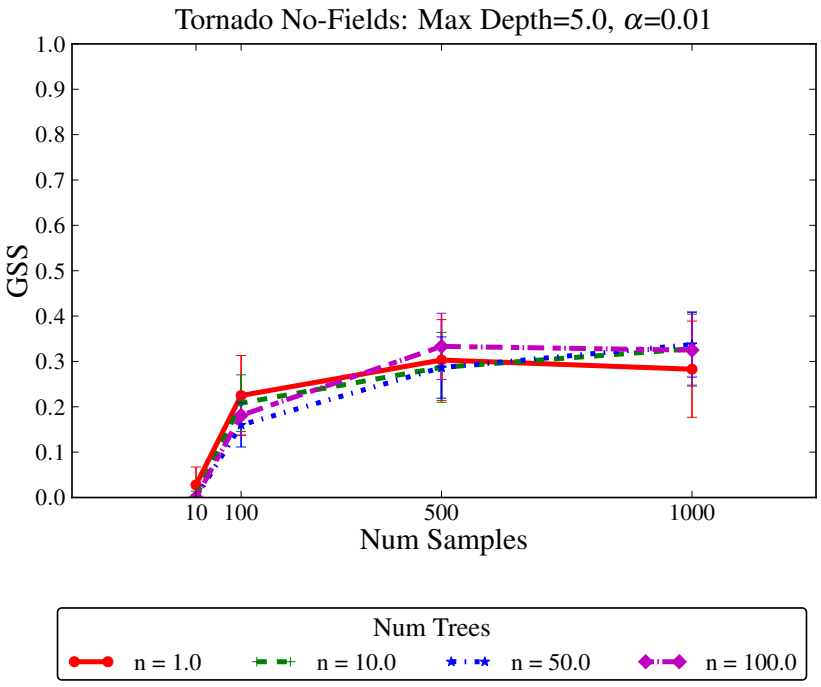


Figure 4.22: GSS for Tornado without Fields varying across K and N with D=5. Error bars are standard deviation.

Fields	Mean Variable Importance	Non-Fields	Mean Variable Importance
PressurePerturbation.PressurePerturbation	0.251	PressurePerturbation.Thickness	0.264
PressurePerturbation→ Overlap.PercentOverlap→Rain	0.188	PressurePerturbation.Volume	0.179
Hail.HorizontalCompositeArea	0.114	Hail.HorizontalCompositeArea	0.177
PressurePerturbation.Volume	0.107	PressurePerturbation→ Overlap.PercentOverlap→Rain	0.156
PressurePerturbation.Thickness	0.101	Hail.Volume	0.134
PressurePerturbation.BaseHeight	0.099	PressurePerturbation→Nearby.MinDistance →PressurePerturbation	0.105
Hail.Volume	0.089	Updraft.Volume	0.067
Updraft.Volume	0.086	PressurePerturbation.PercentForward	0.065
Updraft.Thickness	0.061	Hail.BaseHeight	0.060
Updraft.Buoyancy	0.060	Hail→Nearby.MinDistance →PressurePerturbation	0.060

Table 4.18: Top 10 Statistically Significant Attributes According to Variable Importance on the Tornado Dataset

	% Total	100		25	12	11	9	8
	Total		25204	6339	3127	2875	2383	2027
	PressurePerturbation→Overlap.PercentOverlap →Rain	4	938	–	189	215	209	192
	PressurePerturbation.PressurePerturbation	3	814	–	340	31	3	5
	Hail	2	630	516	–	–	–	–
	PressurePerturbation→Contains.PercentOverlap→Downdraft	2	573	–	108	100	99	89
	PosTilting	2	512	510	–	–	–	–
	Hail→Overlap→PressurePerturbation	2	482	476	–	–	–	–
	Hail.Qh	2	474	–	16	14	10	10
	PressurePerturbation	2	465	33	–	–	–	–
	PressurePerturbation→Overlap.PercentOverlap→Updraft	2	403	–	85	87	90	60
	PosTilting.Tilting	2	382	–	10	20	11	16
All								

Table 4.19: Top 1-5 Distinctions and top 10 Attributes/Objects/Relations for Tornado with 100 trees, 1000 samples, depth 5 set of trees. *(Continued in Table 4.20)*

Field.Gradient		3	652	–	33	–	–	–	71	–	–	47
TemporalExists		4	922	–	–	112	–	2	–	429	–	–
FieldAttribute.Gradient		5	1264	–	312	–	–	–	140	–	–	109
Attribute.Mode		7	1862	15	17	–	71	–	11	–	24	8
Attribute.Min		8	1938	111	11	–	101	–	22	–	56	21
Total			25204	938	814	630	573	512	474	465	403	382
% Total	100			4	3	2	2	2	2	2	2	2
All	% Total											
	Total											
	PressurePerturbation→Overlap.PercentOverlap→Rain											
	PressurePerturbation.PressurePerturbation											
	Hail											
	PressurePerturbation→Contains.PercentOverlap→Downdraft											
	PosTilting											
	Hail→Overlap→PressurePerturbation											
	Hail.Qh											
	PressurePerturbation											
	PressurePerturbation→Overlap.PercentOverlap→Updraft											
	PosTilting.Tilting											

Table 4.20: Top 6-10 Distinctions and top 10 Attributes/Objects/Relations for Tornado with 100 trees, 1000 samples, depth 5 set of trees.*(Continued from Table 4.19)*

Figure 4.23: Highest Scoring (by GSS) Single Tornado Tree

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we made several major enhancements to the SRPT algorithm. The addition of fields to objects, both 2D and 3D fields, either scalar or vector, greatly increases the information available to the SRPTs. To make use of this new data we added two types of distinctions increasing the number and type of questions that SRPTs can ask of the data. First, we added shape detection in both 2D and 3D allowing for reasoning on the shape of a field, if a field is changing shapes, and if the axis of the shape is tilting. Second, we provided the means to reason about the actual value of data in the fields by providing distinctions that deal with the gradient, curl, and divergence of fields. Our hypothesis was that the addition of fielded objects would noticeably increase the performance of the SRPT algorithm.

To test our hypothesis, we applied spatiotemporal relational random forests to four domains, one synthetic and three real world. The Shapes domain, the only synthetic domain, showed that the SRRFs were able to use the new shapes based distinctions to reason about a problem consisting of mainly 3D fields where the shapes and the changing of those shapes were crucial to the classification task.

Turbulence, the first of the real-world domains, involved the prediction of convectively induced turbulence. The results showed that Turbulence is a very difficult domain for SRRFs to gain traction. However, the AUC of the SRRFs show a robust classifier. While the additions to the SRRF algorithm, specifically the addition of fielded data, did not show statistically significant improvement over the non-fielded version, neither did it cause an loss of performance. It is unclear why the fielded Turbulence domain was not able to outperform the non-fielded version with the non-fielded, but it is likely that the sheer difficulty of the problem, as shown by the low GSS, contributed to the lack of a difference. A new turbulence dataset is in progress.

It will contain 3D instead of 2D fields and more fielded attributes. We expect improved results on this new dataset.

Fronts is the second real-world domain. The goal of Fronts is to predict tornado formation based on frontal movement and boundaries and the meteorological variables in those fronts and nearby thunder storms. Similar to Turbulence the SRRFs were unable to gain traction on the domain yielding an even lower GSS, but still not performing worse. Again, like Turbulence, the AUC results for Fronts show a robust classifier. Comparing the fielded Fronts domain to the non-fielded version showed the same result as with Turbulence, there was no statistically significant difference between them with GSS or AUC. A likely reason for the lack of performance difference is that both Turbulence and Fronts only contain 2D fields and that the data added by 2D fields is not significantly more useful than the reduced non-field version.

Tornado is the last domain to which we applied SRRF's. Similar to Fronts, Tornado deals with classifying a thunderstorm as being tornadic or not. For this domain the SRRFs were able produce markedly better performance than on Fronts or Turbulence. When compared to its non-fielded version Tornado showed a statistically significant difference between the performance of fields and non-fields. Unlike Fronts and Turbulence, Tornado contains 3D fields. This may contribute the increase in performance by the addition of fields. 3D fields contain much more information for comparably sized 2D fields meaning much more information is lost when reduced to a single scalar or vector.

While the addition of fields to SRPTs is not guaranteed to significantly increase the performance as measured by GSS or AUC on all domains, in none of the domains has it caused a degradation in performance. The classifiers generated by the SRRF algorithm in all three domains are quite robust as indicated by their AUC. The low GSS on Fronts and Turbulence suggests that the needed information to perform well at the classification task is simply missing, hence neither fielded or non-fielded absent information is useful. Of the three real-world domains only Tornado show an significant benefit from the addition of fields. This indicates that the domain is the key factor how well the SRRF can learn the classification task. We have shown that the addition of fields can be a valuable resource to the SRRF algorithm on some spatiotemporal relational domains.

Further details of the implementation, source code, and experimental results can be found at: http://idea.cs.ou.edu/theses/ntroutman

5.2 Future Work

The multi-relational decision tree learning (MRDTL) algorithm performs one-step look-ahead when it fails to find a significant split (Leiva 2002). The addition of this to the SRPT algorithm would be an interesting feature to explore. This would give the SRRFs the ability to look one step down the tree allowing it to search questions that might not have immediate significance but could later on. One approach would be to generate random subtrees of three distinction nodes and check if either child of the subtree produced a significant split. Look-ahead searching could be enabled at all times though it would greatly increase the computational expense of the algorithm. A more moderate approach would be to limit the look-ahead to when no single significant distinction is found. The current Boolean distinction allows for some limited amount of the functionality look-ahead would provide, but not nearly as much as having full look-ahead would provide.

Autocorrelation was mentioned briefly in Section 3.7 in regards to handling missing values, however it can have much wider effects and frequently appears naturally instead of as a side-effect of pre-processing. While autocorrelation is usually considered a nuisance because of its effects on traditional statistics (Jensen and Neville 2002), it is an accepted fact in geographical (spatial) studies and can even be exploited to advantage (O'Sullivan and Unwin 2002). Thus, under certain circumstances, the autocorrelation becomes part of the structure that you are studying and should be left alone. For example, O'Sullivan and Unwin (2002) states "Geographical data are often not samples in the sense meant in standard statistics. Frequently, geographical data represent the whole population. Often, we are only interested in understanding the study region, not in making wider inferences about the whole world, so that the data *are* the entire population of interest." This is best determined on a case by case basis.

Although Jensen and Neville (2002) demonstrated that relational autocorrelation can be corrected for using randomization by attribute value (instead of by class label), it is not immediately clear how to apply this to either spatially or temporally varying attributes. In some cases, there may be ways to exploit this autocorrelation(Neville and Jensen 2005; Angin and Neville 2008). The effects of auto-correlation on spatiotemporal relational learning, particularly spatial autocorrelation, still need much study. Randomization methods allow for determining the effects of autocorrelation by breaking the autocorrelation and analyzing the changes in performance. While

relational randomization is easy, and even to some degree temporal randomization, exactly how spatial randomization should be handled is an open question.

In addition, we would like to add the ability to reason with degree features, as the RPT can do. For instance, SRPTs have no method for counting the number of possible values of a discrete attribute. Nor can they count the number of relations for a single object, such as asking the question, "Have any of the actors in this movie acted in more than 10 other movies?" That question requires counting the the number of *acted-in* relations for each actor. Addition of these questions to the SRPTs arsenal might be beneficial to some domains.

Reference List

Agrawl, R. and R. Srikant, 1994: Fast algorithms for mining association rules. *Proceedings of the 20th VLDB Conference*, 487–499.

Allen, J. F., 1991: Time and time again: The many ways to represent time. *International Journal of Intelligent Systems*, **6**, 341–355.

Angin, P. and J. Neville, 2008: A shrinkage approach for modeling non-stationary relational autocorrelation. *International Conference on Data Mining*, 707–712.

Barber, C., D. Dobkin, and H. Huhdanpaa, 1996: The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, **22**, 469–483.
URL `http://www.qhull.org`

Bille, P., 2003: Tree edit distance, alignment distance and inclusion. Technical report, IT University of Copenhagen Technical Report Series TR-2003-23.

Blockeel, H., 1998: *Top-down Induction of First Order Logical Decision Trees*. Ph.D. thesis, Department of Computer Science, Katholieke Universiteit Leuven.

Blockeel, H. and L. De Raedt, 1998: Top-down induction of logical decision trees. *Artificial Intelligence*, **101**, 285–297.

Bosch, A., A. Zisserman, and X. Munoz, 2007: Image classification using random forests and ferns. *Proceedings of the 11th International Conference on Computer Vision*, 1–8.

Bradley, A. P., 1997: The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, **30**, 1145–1159.

Breiman, L., 2001: Random forests. *Machine Learning*, **45**, 5–32.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone, 1984: *Classification and regression trees*. Belmont: Wadsworth.

Cova, T. and M. Goodchild, 2002: Extending geographical representation to include fields of spatial objects. *International Journal of Geographical Information Science*, **16**, 509–532.

Danyluk, A. and F. Provost, 2000: *Telecommunications Network Diagnosis*. Handbook of Knowledge Discovery and Data Mining, 897–902.

Domingos, P., 1999: Metacost: A general method for making classifiers cost-sensitive. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 155–164.

Egan, J. P., 1975: *Signal Detection Theory and ROC Analysis*, Academic Press, New York. Series in Cognition and Perception.

Fislason, P. O., J. A. Benedikstsson, and J. Sveinsson, 2006: Random forests for land cover classification. *Pattern Recognition Letters*, **27**, 294–300.

Freeman, H. and R. Shapira, 1975: Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Communications of the ACM*, **18**, 409–413.

Friedman, H. F., R. Kohavi, and Y. Yun, 1996: Lazy decision trees. *Proceedings of the 13th National Conference on Artificial Intelligence*, 717–724.

Gagne II, D. J., T. Supinie, A. McGovern, J. Basara, and R. A. Brown, 2010: Analyizing the effects of low level boundaries on tornadogensis through spatiotemporal relational data mining. *Presented at the 8th Conference on Artificial Intelligence Applications to Environmental Science*, electronically published.

Gandin, L. S. and A. H. Murphy, 1992: Equitable skill scores for categorical forecasts. *Monthly Weather Review*, **120**, 361–370.

Gerrity, J. P., 1992: A note on Gandin and Murphy's equitable skill score. *Monthy Weather Review*, **120**, 2709–2712.

Goodchild, M., M. Yuan, and T. Cova, 2007: Towards a general theory of geographic representation in GIS. *International Journal of Geographical Information Science*, **21**, 239–260.

Heckerman, D., M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, 2000: Dependency networks for density estimation, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 49–75.

Hocker, J. and J. Basara, 2008: A geographic information systems-based analysis of supercells across Oklahoma from 1994-2003. *Journal of Applied Meteorology Climatology*, **47**, 1518–1538.

Jelinek, F., 1997: *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA.

Jenker, J., M. Sprenger, I. Schwenk, and C. S. D. D. Leuenberger, 2005: Detection and climatology of fronts in a high-resolution model reanalysis over the alps. *Meteorological Applications*, **17**, 1–18.

Jensen, D. and J. Neville, 2002: Linkage and autocorrelation cause feature selection bias in relational learning. *Proceedings of the International Conference on Machine Learning*, 259–266.

Jensen, D., J. Neville, and M. Hay, 1999: Avoiding bias when aggregating relational data with degree disparity. *Proceedings of the 3rd European Conference on Principles & Practice of KDD*, 378–383.

Jolliffe, I. T. and D. B. Stephenson, 2003: *Forecast Verification: A Practitioner's Guide in Atmospheric Science*. Wiley.

Kalousis, A., J. Prados, and M. Hilario, 2007: Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge Information Systems*, **12**, 95–116.

Knobbe, A. J., A. Seibes, and D. van der Wallen, 1999: Multi-relational decisision tree induction. *In Proceedings of PKDD' 99*, 378–383.

Kononenko, I., I. Bratko, and E. Roskar, 1984: Experiments in automatic learning of medical diagnostic rules. Technical report, Jozef Stefan Institute, Ljubjana, Yugoslavia.

Leiva, H. A., 2002: MRDTL: A multi-relational decision tree learning algorithm.

Liu, W. Z., A. P. White, S. G. Thompson, and M. A. Bramer, 1997: *Techniques for Dealing with Missing Values in Classification*, Springer-Verlag Berlin Heidelberg. Advances in Intelligent Data Analysis, 527–536.

Longley, P. A., M. Goodchild, D. J. Maguire, and D. W. Rhind, 2005: *Geographic Information Systems and Science*. Wiley.

Markowski, P., E. Rasmussen, and J. Straka, 1998: The occurrences of tornadoes in supercells interacting with boundaries during vortex. *Weather Forecasting*, **13**, 852–859.

Markowski, P. and Y. Richardson, 2009: Tornadogenesis: Our current understanding, forecasting considerations, and questions to guide future research. *Atmospheric Research*, **93**, 3–10.

Marzban, C., 1998: Scalar measures of performance in rare-event situations. *Weather and Forecasting*, **13**, 753–763.

McGovern, A., N. Hiers, M. Collier, D. J. Gagne II, and R. A. Brown, 2008: Spatiotemporal relational probability trees. *Proceedings of the 2008 IEEE International Conference on Data Mining*, 935–940.

McGovern, A., T. Supinie, D. J. Gagne II, N. P. Troutman, M. Collier, R. A. Brown, J. Basara, and J. K. Williams, 2010: Augmenting spatiotemporal relational random forests for use in real-world severe weather applications. *To appear in NASA Conference on Intelligent Data Understanding: CIDU 2010*, pages 9 unumbered.

McPherson, R. A., C. A. Fiebrich, K. C. Crawford, R. L. Elliott, J. R. Kilboy, D. L. Grimsley, J. E. Martinez, J. Basara, B. G. Illston, D. A. Morris, K. A. Kloesel, S. J. Stadler, A. D. Melvin, A. J. Sutherland, H. Shrivastava, J. D. Carlson, J. M. Wolfinbarger, J. P. Bostic, and D. B. Demko, 2007: Statewide monitoring of mesoscale environment: A technical update on the Oklahoma mesonet. *Journal of Atmospheric and Oceanic Technology*, **24**, 301–321.

Meinshausen, M., 2006: Quantile regression forests. *Journal of Machine Learning Research*, **7**, 983–999.

Muggleton, S. and L. De Raedt, 1994: Inductive logic programming: Theory and methods. *Journal of Logic Programming*, **19**, 629–679.

Neville, J. and D. Jensen, 2005: Leveraging relational autocorrelation with latent group models. *Proceedings of the International Conference on Data Mining*, 322–329.

—, 2007: Relational dependency networks. *The Journal of Machine Learning Research*, **8**.

Neville, J., D. Jensen, L. Friedland, and M. Hay, 2003: Learning relational probability trees. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 625–630.

Neville, J., D. Jensen, and B. Gallagher, 2004: Simple estimators for relational bayesian classifiers. Technical report, Knowledge Discovery Laboratory , Department of Computer Science, University of Massachusetts Amherst.

O'Rourke, J., 1985: Finding minimal enclosing boxes. *International Journal of Computer and Information Sciences*, **14**, 17.

O'Sullivan, D. and D. J. Unwin, 2002: Geographic information analysis. *John Wiley and Sons, Inc., Hoboken, New Jersey*.

Provost, F. and P. Domingos, 2000: Well-trained PETs: Improving probability estimation trees. *University of Washington; CDER working paper 00-04-is , Stern School of Business , NYU*, electronically published.

Provost, F. and T. Fawcett, 2001: Robust classification for imprecise environments. *Machine Learning*, **42**, 203–231.

Provost, F., T. Fawcett, and R. Kohavi, 1998: The case against accuracy estimaiton for comparing induction algorithms. *Proceedings of the Fifteenth International Conference on Machine Learning*, 445–453.

Quinlan, J. R., 1986: Induction of decision trees. *Journal of Machine Learning*, **1**, 81–106.

—, 1990: Learning logical definitions from relations. *Machine Learning*, **5**, 239–266.

—, 1993: *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., Santa Mateo, CA.

Renard, R. J. and L. C. Clarke, 1965: Experiments in numerical objective frontal analysis. *Monthly Weather Review*, **93**, 547–556.

Richardson, M. and P. Domingos, 2006: Markov logic networks. *Machine Learning*, **62**, 107–136.

Rosendahl, D. H., 2008: *Identifying precursors to strong low-level rotation with numerically simulated supercell thunderstorms: A data mining approach*. Master's thesis, University of Oklahoma, School of Meteorology.

Schnabel, R., R. Wahl, and R. Klein, 2007: Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, **26**, 214–226.

Segal, M. R., 2004: Machine learning benchmarks and random forest regression.

Sharan, U. and J. Neville, 2008: Temporal-relational classifiers for prediction in evolving domains. *Proceedings of the International Conference on Data Mining*.

Sharman, R., C. Tebaldi, G. Weiner, and J. Wolff, 2006: An integrated approach to mid- and upper-level turbulence forecasting. *Weather and Forecasting*, **21**, 268–287.

Srinivasan, A., 1999: A study of two probabilistic methods for searching large spaces with ILP. Technical report, PRG-TR-16-00 Oxford University Computing Laboratory, University of Oxford.

Storm Prediction Center, 2009: Annual fatal tornado summaries.
URL `http://www.spc.noaa.gov/climo/torn/fataltorn.html`

Supinie, T., A. McGovern, J. K. Williams, and J. Abernethy, 2009: Spatiotemporal relational random forests. *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, 630–635.

Taskar, B., P. Abbeel, and D. Koller, 2002: Discriminative probabilistic models for relational data. *In Proceedings of UAI'02*.

Wallace, J. M. and P. V. Hobbs, 2006: *Atmospheric Science: An Introductory Survey*. Academic Press.

White, A. P., 1987: *Probabilistic induction by dynamic path generation in virtual trees*, Cambridge: Cambridge University Press. Research and Development in Expert Systems III, 34–46.

Williams, J. K., D. Ahijevych, S. Dettling, and M. Steiner, 2008: Combining observations and model data for short-term storm forecasting. *Remote Sensing Applications for Aviation Weather Hazard Dection and Decision Support*.

Xue, M., D. Wang, J. Gao, K. Brewster, and K. K. Droegemeier, 2003: The advanced regional prediction system (ARPS), storm-scale numerical weather prediction and data assimilation. *Meteorology and Atmospheric Physics*, 139–170.

Appendix

1 Schema and Graph Files

The proposed extensions to the SRPT mandated a complete redesign of the existing code base. With the goal of easy maintenance and eventual release of the implementation to the scientific community, the redesign also included reworking the XML format of the schema and graphs. The schema was simplified by enforcing naming conventions of xml elements in both the schema and the graphs. A simple schema file is show in Table A.1. A new addition includes the ability to define relationships between many object types simultaneously by cross-production construction using a list of source and destination types. This was partially dictated by the change that all relations are now directional whereas the old SRPT assumed all relations were bi-directional.

Along with schema format changes came an overhaul to the format of the XML files for storing graphs. The schema not only describes what objects, relations, and attributes can be in the dataset, but also the actual format of the XML files. This is used to verify that the XML of the graph is correct by forcing all objects, relations, and attributes to agree with the schema. Table A.2 shows an example XML graphs file. It can been seen that the format of the XML file follows the schema, using object and relationship names as the actual tag names. This makes for very human friendly XML files.

Other notable improvements include experiment files which define the specifics of an experiment, such as which schema file to use and where the XML and pickled graphs are stored. This replaces a massive if statement that required constant updating whenever details were changed or new experiments added. The SRPT tree class was updated to use extendable node types. This allows for easily adding new classification node types to be used at the leaves.

2 Storing Field Data

In order to facilitate a wide number of use cases there are several ways to store field data. The field data can be stored local to each attribute, separate but still

```
<schema>
  <attrib name='max_time' type='int'>10</attrib>
  <attrib name='class_labels' type='str-array'>0,1</attrib>

  <graph>
    <object type='ball'>
      <attribute name='volume' type='float-array' />
      <attribute name='color' type='discrete' />
    </object>

    <object type='square'>
      <attribute name='volume' type='float-array' />
      <attribute name='color' type='discrete' />
    </object>

    <object type='pyramid'>
      <attribute name='volume' type='float-array' />
      <attribute name='color' type='discrete' />
    </object>

    <relation type='onTopOf' source-type='square,pyramid,ball'
     target-type='square,pyramid,ball'>
      <attribute name='distance' type='float-array' />
    </relation>

    <relation type='nearby' source-type='square,pyramid,ball'
     target-type='square,pyramid,ball' reflection='nearby'>
      <attribute name='distance' type='float-array' />
    </relation>
  </graph>
</schema>
```

Table A.1: Sample XML Schema file used by new SRPT.

```
<graphs>
  <graph start_time='1' end_time='27' class_label='1' id='2'>
      <ball start_time='4' end_time='10' id='1'>
         <color>green</color>
         <volume>6,7,8,9,10,11</volume>
      </ball>
      <square start_time='23' end_time='27' id='2'>
         <color>blue</color>
         <volume>20,20,20,20</volume>
      </square>
      <pyramid start_time='10' end_time='16' id='3'>
         <color>green</color>
         <volume>5,6,7,8,9,10</volume>
      </pyramid>

      <onTopOf start_time='8' end_time='9' source='1' target='3'>
         <distance>1,2</distance>
      </onTopOf>
      <nearby start_time='13' end_time='14' source='2' target='3'>
         <distance>5,3</distance>
      </nearby>
      <onTopOf start_time='8' end_time='9' source='1' target='2'>
         <distance>2,2</distance>
      </onTopOf>
   </graph>
</graphs>
```

Table A.2: Sample XML of a graphs file used by new SRPT.

in the XML file. We plan to add the ability to store the fields in Hierarchical Data Format/Network Common Data Form (HDF/NetCDF) files and reference them from the XML. Since part of the focus of SRPT's is temporal data, each timestep of an attribute defines its own field to allow for the changing nature of spatial objects over time. For the ease of storage and parsing, two dimensional fields are stored as rectangular regions and three dimensional fields use rectangular cuboids, essentially two and three dimensional matrices. The regions are defined by a corner attribute and a size attribute.

The fields are stored as matrices which are square/rectangular but not all the points may be of actual points of interest. Because of this the raw values are also accompanied with a mask. The mask selects a subset of the values that will actually be used by the SRPT algorithm. It is represented as a boolean matrix of the same size as the data region, with a 1 (used) or 0 (not used) for each value in the field.

With local storage only the appropriate region of the field is stored directly with the attribute. This is most useful if there is only a local region or if only a small portion of a much larger parent field is ever used, Table A.3 is an example of the XML format. The second option is to store the entire parent field as a separate object within the XML file. Then each attribute just references that object with the region's location and size.

Four types of general fields are possible, float and integer scalar fields and float and integer vector fields. Two and three dimensional fields can be used and all fields are assumed to be uniformly spaced. Uniform spacing of the fields means that it is assumed that the distance between any two adjacent values is the same and similarly for any two diagonal values. For simplicity they are represented the same in the XML file. All field data is stored as a single one dimensional array, as is the mask. The array is reshaped into the correct size and dimensionality using the given size attribute from which the number of dimensions along with the data type is also inferred. The individual elements of the arrays are space or comma separated and the data and mask are separated by a semicolon.

Further details of the implementation, source code, and experimental results can be found at: http://idea.cs.ou.edu/theses/ntroutman

```
<happiness datalocation='self'>
      <timestep corner='2,1' size='2,3'>
         0.81 0.88 -0.82 -0.78 0.43 0.16; 1 1 0 0 1 0
      </timestep>
      <timestep>...</timestep>
</happiness >
```

Table A.3: Example XML for a two-dimensional scalar field stored locally in each timestep.