Geographical Hidden Markov Tree for Flood Extent Mapping

Miao Xie
Department of Computer Science
University of Alabama
mxie9@crimson.ua.edu

Zhe Jiang*
Department of Computer Science
University of Alabama
zjiang@cs.ua.edu

Arpan Man Sainju
Department of Computer Science
University of Alabama
asainju@crimson.ua.edu

ABSTRACT

Flood extent mapping plays a crucial role in disaster management and national water forecasting. Unfortunately, traditional classification methods are often hampered by the existence of noise, obstacles and heterogeneity in spectral features as well as implicit anisotropic spatial dependency across class labels. In this paper, we propose geographical hidden Markov tree, a probabilistic graphical model that generalizes the common hidden Markov model from a one dimensional sequence to a two dimensional map. Partial order class dependency is incorporated in the hidden class layer with a reverse tree structure. We also investigate computational algorithms for reverse tree construction, model parameter learning and class inference. Extensive evaluations on both synthetic and real world datasets show that proposed model outperforms multiple baselines in flood mapping, and our algorithms are scalable on large data sizes.

CCS CONCEPTS

• **Information systems** → *Geographic information systems*; *Data mining*; • **Computing methodologies** → *Machine learning*; • **Applied computing** → *Earth and atmospheric sciences*;

KEYWORDS

Geographical Hidden Markov Tree; Spatial classification

ACM Reference Format:
Miao Xie, Zhe Jiang, and Arpan Man Sainju. 2018. Geographical Hidden Markov Tree for Flood Extent Mapping. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220053

1 INTRODUCTION

Flood extent mapping plays a crucial role in addressing grand societal challenges such as disaster management, national water forecasting, as well as energy and food security. For example, during Hurricane Harvey floods in 2017, first responders needed to know where flood water was in order to plan rescue efforts. In national water forecasting, detailed flood extent maps can be used to calibrate and validate the NOAA National Water Model [15], which can forecast the flow of over 2.7 million rivers and streams through the entire continental U.S. [4].

In current practice, flood extent maps are mostly generated by flood forecasting models, whose accuracy is often unsatisfactory in high spatial details [4]. Other ways to generate flood maps involve sending a field crew on the ground to record high-water marks, or visually interpreting earth observation imagery [2]. However, the process is both expensive and time consuming. With the large amount of high-resolution earth imagery being collected from satellites (e.g., DigitalGlobe, Planet Labs), aerial planes (e.g., NOAA National Geodetic Survey), and unmanned aerial vehicles, the cost of manually labeling flood extent becomes prohibitive.

The focus of this paper is to develop a classification model that can automatically classify earth observation imagery pixels into flood extent maps. The results can be used by first responders to plan rescue efforts, by hydrologists to calibrate and validate water forecasting models, as well as by insurance companies to process claims. Specifically, we can utilize a small set of manually collected ground truth (flood and dry locations) in one earth imagery to learn a classification model. Then the model can be used to classify flood pixels in other imagery where ground truth is not available.

However, flood mapping poses several unique challenges that are not well addressed in traditional classification problems. First, data contains rich noise and obstacles. For example, high-resolution earth imagery often has noise, clouds and shadows. The spectral features of image pixels are insufficient to distinguish classes. Second, class confusion exists due to heterogeneous features. For instance, pixels of tree canopies overlaying flood water have the same spectral features with those trees in dry areas, yet their classes are different. Third, implicit directed spatial dependency exists between flood pixel locations. Specifically, due to gravity, flood water tends to flow to nearby lower locations following topography. Such dependency is not uniform in all directions (anisotropic). Finally, the data volume is huge in high-resolution imagery (e.g., hundreds of millions of pixels in one city), requiring scalable algorithms.

To address these challenges, we propose a novel spatial classification model called *geographical hidden Markov tree (HMT)*. It is a probablistic graphical model that generalizes the common hidden Markov model (HMM) from a one-dimensional sequence to a two dimensional geographical map. Specifically, the hidden class layer contains nodes (pixels) in a reverse tree structure to represent anisotropic spatial dependency with a partial order constraint. Each hidden class node has an associated observed feature node for the same pixel. Such a unique model structure can potentially reduce classification errors due to noise, obstacles, and heterogeneity among spectral features of individual pixels.

We further investigate computational algorithms for reverse tree construction, model parameter learning, and class inference.

[*] Corresponding author.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220053

Specifically, reverse tree is constructed following topological orders based on elevations. In order to learn model parameters given a hidden class layer, we utilize the EM algorithm with message propagation along the reverse tree. Finally, for class inference, we design a greedy algorithm that assign class labels for tree nodes to maximize overall probability following the partial order constraint.

In summary, we make the following contributions:

- We propose a novel geographical hidden Markov tree (HMT) model that incorporates partial order class dependency in a reverse tree structure in a hidden class layer. Unlike existing hidden Markov trees [5] which model dependency in two-dimensional time-frequency domain for signal processing, our geographical HMT captures anisotropic (directed) spatial dependency with a partial order constraint.
- We design efficient algorithms for reverse tree construction, model parameter learning and class inference.
- We conduct theoretical analysis on the correctness and time complexity of HMT algorithms.
- We evaluate proposed model in both synthetic and real world datasets for flood mapping. Results show that proposed model outperforms multiple baseline methods in flood mapping, and our algorithms are scalable for large data sizes.

2 PROBLEM STATEMENT

2.1 Preliminaries

Definition 2.1. A *spatial raster framework* is a tessellation of a two dimensional plane into a regular grid of N cells. Spatial neighborhood relationship exists between cells based on cell adjacency. The framework can consist of m non-spatial explanatory feature layers (e.g., spectral bands in earth imagery), one spatial contextual layer (e.g., elevation), and one class layer (e.g., *flood*, *dry*).

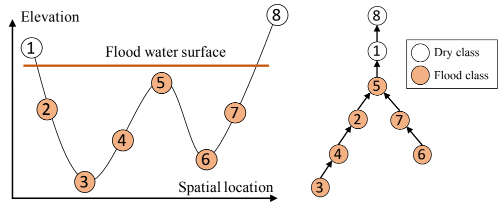
Definition 2.2. Each cell in a raster framework is a *spatial data sample*, noted as $\mathbf{s_n} = (\mathbf{x}_n, \phi_n, y_n)$, where $n \in \mathbb{N}, 1 \le n \le N$, $\mathbf{x}_n \in \mathbb{R}^{m\times 1}$ is a vector of m non-spatial explanatory feature values with each element corresponding to one feature layer, $\phi_n \in \mathbb{R}$ is a cell's value in the spatial contextual layer, and $y_n \in \{0, 1\}$ is a binary class label.

A raster framework with all samples is noted as $\mathcal{F} = \{\mathbf{s_n} | n \in \mathbb{N}, 1 \le n \le N\}$, non-spatial explanatory features of all samples are noted as $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$, the spatial contextual layer is noted as $\mathbf{\Phi} = [\phi_1, ..., \phi_N]^T$, and the class layer is noted as $\mathbf{Y} = [y_1, ..., y_N]^T$.

Definition 2.3. Due to physics, spatial dependency exists between cells based on their values in the spatial contextual layer. Such dependency is often non-uniform in different directions (*anisotropic*). For example, due to gravity, flood water can only flow to neighboring cells with lower elevation values.

Definition 2.4. Anisotropic dependency often follows a *partial order constraint*. Formally, assuming the spatial contextual layer is a potential field (e.g., elevation), a partial order dependency $\mathbf{s}_i \rightsquigarrow \mathbf{s}_j$ exists if and only if there exist a sequence of neighboring (adjacent) cells $< \mathbf{s}_i, \mathbf{s}_{p_1}, \mathbf{s}_{p_2}, ..., \mathbf{s}_{p_l}, \mathbf{s}_j >$ such that $\phi_j \ge \phi_i$ and $\phi_j \ge \phi_{p_k}$ for any $1 \le k \le l$.

Figure 1(a) shows an illustrative example with eight spatially adjacent cell samples in one dimensional space. Due to gravity, if cell $\mathbf{s}_5$ is *flood*, its nearby cells with lower elevations including $\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_6, \mathbf{s}_7$ should also be *flood*, even if their feature values indicate otherwise. Thus, we can establish partial order spatial dependency between cell locations such as $\mathbf{s}_4 \rightsquigarrow \mathbf{s}_5$, $\mathbf{s}_2 \rightsquigarrow \mathbf{s}_5$.



(a) Eight consecutive sample locations in one dimensional space **(b) Partial order constraint in a reverse tree**

Figure 1: Illustration of partial order class dependency

Definition 2.5. Partial order dependency across all pairs of samples in a raster framework can be represented by a reverse tree structure, which is called *spatial dependency (reverse) tree* or *dependency tree*. We sometimes omit the word "reverse" for simplicity. The tree structure removed some redundant dependency between cell locations. Due to the reverse nature, a tree node n can have at most one child $C_n \in \mathbb{N}$, but multiple parents $\mathcal{P}_n = \{k \in \mathbb{N} | \mathbf{s}_k \to \mathbf{s}_n\}$ and multiple siblings $\mathcal{S}_n = \{k \in \mathbb{N} | \exists\, c \in \mathbb{N}\ s.t.\ \mathbf{s}_k \to \mathbf{s}_c, \mathbf{s}_n \to \mathbf{s}_c\}$, where $\to$ represents a tree edge from a parent to a child.

Figure 1(b) shows an example of dependency tree corresponding to samples in Figure 1(a). Class dependency $\mathbf{s}_3 \rightsquigarrow \mathbf{s}_2$ is redundant given dependency $\mathbf{s}_3 \to \mathbf{s}_4$ and $\mathbf{s}_4 \to \mathbf{s}_2$. It is worth noting that we assume an arbitrary order when comparing nodes with the same elevation values. For instance, if node $\mathbf{s}_1$ and node $\mathbf{s}_8$ had the same elevation, the top of the tree could be either $\mathbf{s}_5 \to \mathbf{s}_1 \to \mathbf{s}_8$ or $\mathbf{s}_5 \to \mathbf{s}_8 \to \mathbf{s}_1$.

2.2 Formal problem definition

We now formally define the problem.

Input:
• Spatial raster framework $\mathcal{F} = \{\mathbf{s}_n | n \in \mathbb{N}, 1 \le n \le N\}$
• Explanatory features of samples $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$
• Spatial contextual layer (elevation) of samples: $\mathbf{\Phi} = [\phi_1, ..., \phi_N]^T$
• Training samples $\{\mathbf{s}_k | k \in \textit{training set}\}$
Output: A spatial classification model $f : \mathbf{Y} = f(\mathbf{X})$
Objective: minimize classification errors
Constraint:
• Explanatory feature layers contain noise and obstacles
• Partial order dependency exists between sample classes based on spatial contextual layer
• Sample class is binary, $y_n \in \{0, 1\}$

3 PROPOSED APPROACH

In this section, we start with overview of our hidden Markov tree model and its probabilistic formulation. We then introduce specific algorithms for dependency tree construction, model parameter learning and class inference.

3.1 Overview of Hidden Markov Tree

We propose a hidden Markov tree (HMT) model, which generalizes the common hidden Markov model from a total order chain structure to a partial order (reverse) tree structure. As illustrated in Figure 2, a HMT model consists of two layers: a hidden layer of sample classes (e.g., flood, dry), and an observation layer of sample feature vectors (e.g., spectral vectors). Each node corresponds to a spatial data sample (raster cell). Edge directions show probabilistic conditional dependence structure. Specifically, the model assumes that feature vectors of different samples are conditionally independent with each other given their classes, and sample classes follow a partial order dependency in a reverse tree structure.

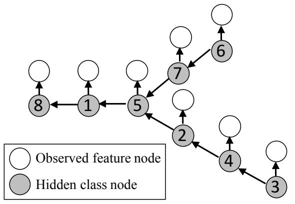


Figure 2: Illustration of hidden Markov tree framework

Hidden Markov tree is a probabilistic graphic model. The joint distribution of all samples' features and classes can be expressed as Equation 1, where $\mathcal{P}_n$ is the set of parent samples of the nth sample in the dependency tree, and $y_{k\in\mathcal{P}_n} \equiv \{y_k | k \in \mathcal{P}_n\}$ is the set of class nodes corresponding to parents of the nth sample. For a leaf node n, $\mathcal{P}_n = \emptyset$, and $P(y_n | y_{k\in\mathcal{P}_n}) = P(y_n)$.

$$P(\mathbf{X}, \mathbf{Y}) = P(\mathbf{X}|\mathbf{Y})P(\mathbf{Y}) = \prod_{n=1}^{N} P(\mathbf{x}_n|y_n) \prod_{n=1}^{N} P(y_n|y_{k\in\mathcal{P}_n}) \quad (1)$$

The conditional probability of sample feature vector given its class can be assumed i.i.d. Gaussian for simplicity, as shown in Equation 2, where $\boldsymbol{\mu}_{y_n}$ and Σ_{y_n} are the mean and covariance matrix of feature vector $\mathbf{x}_n$ for class y_n ($y_n = 0, 1$). It is worth noting that $P(\mathbf{x}_n|y_n)$ could be more general than i.i.d. Gaussian.

$$P(\mathbf{x}_n|y_n) \sim \mathcal{N}(\boldsymbol{\mu}_{y_n}, \Sigma_{y_n}) \quad (2)$$

Class transitional probability follows the partial order constraint. For example, due to gravity, if any parent's class is *dry*, the child's class must be *dry*; if all parents' classes are *flood*, then the child has a high probability of being *flood*. Consider *flood* as the positive class (class value 1) and *dry* as the negative class (class value 0), the transitional probability is actually conditioned on the product of parent classes $y_{\mathcal{P}_n} \equiv \prod_{k\in\mathcal{P}_n} y_k$. The formula is in Equation 3, where ρ is the probability of a child in class 1 given all parents in class 1 (note that we assume $0^0 \equiv 1$). In other words, if any parent is in class 0 ($y_{\mathcal{P}_n} = 0$), the current node must also be in class 0 ($y_n = 0$); if all parents are in class 1 ($y_{\mathcal{P}_n} = 1$), then the current node has a probability of ρ being in class 1.

$$P(y_n|y_{\mathcal{P}_n}) = 1^{(1-y_n)(1-y_{\mathcal{P}_n})} \times 0^{y_n(1-y_{\mathcal{P}_n})} \times \rho^{y_n y_{\mathcal{P}_n}} \times (1-\rho)^{(1-y_n)y_{\mathcal{P}_n}} \quad (3)$$

For a leaf node n, $\mathcal{P}_n = \emptyset$. The transitional probability is degraded into simple class probability $P(y_n|y_{k\in\mathcal{P}_n}) \equiv P(y_n) = \pi^{y_n} \times (1-\pi)^{1-y_n}$, where π is the probability of y_n being in class 1.

Though we introduce our HMT in the context of flood mapping, the model can potentially be used for a broad class of classification problems in which class labels follow a partial order dependency. Examples include predicting pollutants in river stream networks and traffic congestion in road networks.

3.2 Dependency Tree Construction

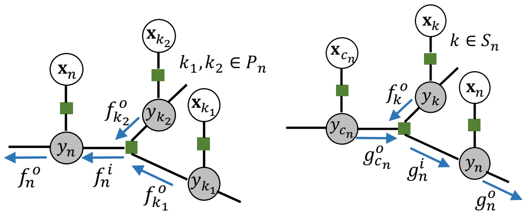
Given geopotential field values (e.g., elevation) of all cells in a raster framework, the goal is to produce a partial order class dependency tree, in which each tree node corresponds to the class label of a cell. The process is computationally challenging due to the large number of cells (tree nodes) in real world high-resolution earth imagery (e.g., hundreds of millions of pixels).

Algorithm 1 Spatial Dependency Tree Construction

```
Input:
  • A raster framework of samples: $\mathcal{F} = \{\mathbf{s_n} | n \in \mathbb{N}, 1 \le n \le N\}$
  • A spatial contextual layer of samples: $\Phi = [\phi_1, ..., \phi_N]^T$
Output:
  • A spatial dependency tree
1:  Initialize all samples as unvisited
2:  Sort all samples by increasing $\phi$ values
3:  for each sample $\mathbf{s_n}$ in an ascending order of $\phi$ do
4:      Mark $\mathbf{s_n}$ as visited
5:      Create a new tree node of $\mathbf{s_n}$
6:      if there exists visited neighbor of $\mathbf{s_n}$ then
7:          for each visited neighbor $\mathbf{s_k}$ of $\mathbf{s_n}$ do
8:              Traverse from node $\mathbf{s_k}$ to the rear of its tree branch
9:              Attach node $\mathbf{s_n}$ to the rear if have not done so
10:     else
11:         Create a tree branch starting from node $\mathbf{s_n}$ as a leaf
12: return the root node of dependency tree
```

To address the challenge, we propose an algorithm that constructs the tree by adding nodes in topological order. Details are in Algorithm 1. The algorithm starts with an empty tree and an empty set of *visited* cells (all cells are *unvisited*, step 1). It sorts all cells by their geopotential field (elevation) values (step 2). After this, *unvisited* cells are added into the tree (i.e., become visited) one by one. Specifically, for each cell following an ascending order of geopotential, the algorithm first marks it as *visited* (step 4), creates a tree node for the cell (step 5), and attaches the tree node to the rear of every tree branch that passes through a *visited* neighbor of the cell (steps 6 to 9). If no neighbor of the cell is *visited*, the cell is a local minimum in geopotential field, and the algorithm creates a new tree branch starting from the node of the cell (steps 10 to 11).

We now use the example of Figure 1 to illustrate the algorithm execution trace. The example contains cells in one dimensional space, but generalization to the case of two dimensional space is trivial. The input contains eight cells from $\mathbf{s}_1$ to $\mathbf{s}_8$. The algorithm first sorts these cells by an ascending order of elevation, and gets a sequence of $\mathbf{s}_3, \mathbf{s}_6, \mathbf{s}_4, \mathbf{s}_7, \mathbf{s}_2, \mathbf{s}_5, \mathbf{s}_1, \mathbf{s}_8$. Then, leaf nodes are created for $\mathbf{s}_3$ and $\mathbf{s}_6$ respectively, since none of their neighbors have been visited by then. Next, when adding $\mathbf{s}_4$, its neighbor $\mathbf{s}_3$ is *visited*, so the algorithm attaches node $\mathbf{s}_4$ to the rear of the branch that passes through $\mathbf{s}_3$. Similarly, node $\mathbf{s}_7$ and $\mathbf{s}_2$ are attached to the two



(a) From leaves to root **(b) From root to leaves**

Figure 3: Illustration of message propagation in a HMT

existing branches respectively. When adding the node for $\mathbf{s}_5$, both of its neighbors are *visited*, so $\mathbf{s}_5$ is attached to the rear of both branches. After this, nodes $\mathbf{s}_1$ and $\mathbf{s}_8$ are added consecutively.

Time complexity analysis: Algorithm 1 involves a one-time sorting of N cells, which is $O(N \log N)$. Then, for each of the N cells, the main operation is to attach the cell to the rear of the branches of its visited neighbors. A naive implementation will cost $O(N)$, making the total cost $O(N^2)$. A smarter way to do this is to maintain a rear node pointer for each branch when it is created (i.e., when a leaf node is added). Assuming that geopotential field values on neighboring cells are contiguous (this is often true since real world elevation of nearby locations do not change suddenly), finding the rear of a neighboring cell's branch is within a constant cost, making the total time cost $O(N \log N + N) = O(N \log N)$ (cost after sorting is linear).

3.3 Model Parameter Learning

The parameters of hidden Markov tree include the mean and covariance matrix of sample features in each class, prior probability of leaf node classes, and class transition probability for non-leaf nodes. We denote the entire set of parameters as $\mathbf{\Theta} = \{\rho, \pi, \boldsymbol{\mu}_c, \Sigma_c | c = 0, 1\}$. Learning the set of parameters poses two major challenges: first, there exist unknown hidden class variables $\mathbf{Y} = [y_1, ..., y_N]^T$, which are non-i.i.d.; second, the number of samples (nodes) is huge (up to hundreds of millions of pixels).

To address these challenges, we propose to use the expectation-maximization (EM) algorithm and message (belief) propagation. Our EM-based approach has the following major steps:

(a) Initialize parameter set $\mathbf{\Theta_0}$
(b) Compute posterior distribution of hidden classes: $P(\mathbf{Y}|\mathbf{X}, \mathbf{\Theta_0})$
(c) Compute posterior expectation of log likelihood: $LL(\mathbf{\Theta}) = \mathbb{E}_{\mathbf{Y}|\mathbf{X},\mathbf{\Theta_0}} \log P(\mathbf{X}, \mathbf{Y}|\mathbf{\Theta})$
(d) Update parameters: $\mathbf{\Theta_0} \leftarrow \arg\max_{\mathbf{\Theta}} LL(\mathbf{\Theta})$
Return $\mathbf{\Theta_0}$ if it's converged, otherwise goto (b)

Among the four steps above, step (b) that computes the joint posterior distribution of all sample classes is practicallly infeasible due to the large number of hidden class nodes that are non-i.i.d. Fortunately, it is not necessary to compute the entire joint posterior distribution of all sample classes $P(\mathbf{Y}|\mathbf{X}, \mathbf{\Theta_0})$. In fact, we only need the marginal posterior distribution of a node's and its parents' classes for non-leaf nodes, as well as the marginal posterior distribution of a node's class for leaf nodes. The reason can be explained through the expression of the posterior expectation of log likelihood in Equation 4.

$$\begin{aligned} LL(\mathbf{\Theta}) &= \mathbb{E}_{\mathbf{Y}|\mathbf{X},\mathbf{\Theta_0}} \log P(\mathbf{X}, \mathbf{Y}|\mathbf{\Theta}) \\ &= \mathbb{E}_{\mathbf{Y}|\mathbf{X},\mathbf{\Theta_0}} \log \left\{ \prod_{n=1}^{N} P(\mathbf{x}_n | y_n, \mathbf{\Theta}) \prod_{n=1}^{N} P(y_n | y_{k \in \mathcal{P}_n}, \mathbf{\Theta}) \right\} \\ &= \sum_{\mathbf{Y}} P(\mathbf{Y}|\mathbf{X}, \mathbf{\Theta_0}) \sum_{n=1}^{N} \{\log P(\mathbf{x}_n|y_n, \mathbf{\Theta}) + \log P(y_n | y_{k \in \mathcal{P}_n}, \mathbf{\Theta})\} \\ &= \sum_{n=1}^{N} \log P(\mathbf{x}_n|y_n, \mathbf{\Theta}) P(y_n|\mathbf{X}, \mathbf{\Theta_0}) \\ &\quad + \sum_{n=1}^{N} \sum_{y_n, y_{k\in\mathcal{P}_n}} \log P(y_n|y_{k\in\mathcal{P}_n}, \mathbf{\Theta}) P(y_n, y_{k\in\mathcal{P}_n}|\mathbf{X}, \mathbf{\Theta_0}) \end{aligned} \tag{4}$$

Note that for leaf node, $\mathcal{P}_n = \emptyset$, and the last term in the last line of above equation is degraded, $\log P(y_n|y_{k\in\mathcal{P}_n}, \mathbf{\Theta}) P(y_n, y_{k\in\mathcal{P}_n}|\mathbf{X}, \mathbf{\Theta_0}) = \log P(y_n|\mathbf{\Theta}) P(y_n|\mathbf{X}, \mathbf{\Theta_0})$.

To compute the marginal posterior distribution $P(y_n, y_{k\in\mathcal{P}_n})$ and $P(y_n)$ (we omit the condition on $\mathbf{X}$ and $\mathbf{\Theta_0}$ for brevity), we propose to use the message propagation method based on the sum and product algorithm [12, 19]. Message propagation along graph (or tree) nodes is a process of marginalizing out those corresponding node variables in a joint distribution.

Figure 3 illustrates the recursive message propagation process on our HMT model. Specifically, forward message propagation from leaves to root is based on Equation 5 and Equation 6, where $f^i_n(y_n)$ and $f^o_n(y_n)$ are the incoming message into and outgoing message from a hidden class node y_n respectively.

$$f^i_n(y_n) = \begin{cases} P(y_n) & \text{if } y_n \text{ is leaf} \\ \sum\limits_{y_{k\in\mathcal{P}_n}} P(y_n|y_{k\in\mathcal{P}_n}) \prod\limits_{k\in\mathcal{P}_n} f^o_k(y_k) & \text{otherwise} \end{cases} \tag{5}$$

$$f^o_n(y_n) = f^i_n(y_n) P(\mathbf{x}_n|y_n) \tag{6}$$

Backward message propagation from root to leaves also follows a recursive process, as shown in Equation 7 and Equation 8, where $g^i_n(y_n)$ and $g^o_n(y_n)$ are the incoming and outgoing messages for class node y_n respectively. The main difference from forward propagation is that when computing incoming message $g^i_n(y_n)$, we need to multiply not only outgoing message from a child node and class transitional probability, but also outgoing messages from sibling nodes in the forward propagation (also illustrated in Figure 3(b)).

$$g^i_n(y_n) = \begin{cases} 1 & \text{if } y_n \text{ is root} \\ \sum\limits_{y_{c_n}, y_{k\in\mathcal{S}_n}} g^o_{c_n} P(y_{c_n}|y_n, y_{k\in\mathcal{S}_n}) \prod\limits_{k\in\mathcal{S}_n} f^o_k(y_k) & \text{otherwise} \end{cases} \tag{7}$$

$$g^o_n(y_n) = g^i_n(y_n) P(\mathbf{x}_n|y_n) \tag{8}$$

After both forward and backward message propagation, we can compute marginal posterior distribution of hidden class variables based on the following theorem.

THEOREM 3.1. *The unnormalized marginal posterior distribution of the class of a leaf node, as well as the classes of a non-leaf node with parents can be computed by (9) and (10) respectively. Their normalized* *marginal posterior distributions can be computed by (11) and (12) respectively.*

$$P'(y_n|\mathbf{X},\mathbf{\Theta_0}) = f_n^i(y_n)g_n^i(y_n)P(\mathbf{x}_n|y_n) \tag{9}$$

$$P'(y_n, y_{k\in\mathcal{P}_n}|\mathbf{X},\mathbf{\Theta_0}) = \prod_{k\in\mathcal{P}_n} f_k^o(y_k)g_n^o(y_n)P(y_n|y_{k\in\mathcal{P}_n}) \tag{10}$$

$$P(y_n|\mathbf{X},\mathbf{\Theta_0}) \leftarrow \frac{P'(y_n|\mathbf{X},\mathbf{\Theta_0})}{\sum\limits_{y_n} P'(y_n|\mathbf{X},\mathbf{\Theta_0})} \tag{11}$$

$$P(y_n, y_{k\in\mathcal{P}_n}|\mathbf{X},\mathbf{\Theta_0}) \leftarrow \frac{P'(y_n, y_{k\in\mathcal{P}_n}|\mathbf{X},\mathbf{\Theta_0})}{\sum\limits_{y_n, y_{k\in\mathcal{P}_n}} P'(y_n, y_{k\in\mathcal{P}_n}|\mathbf{X},\mathbf{\Theta_0})} \tag{12}$$

Proof. Detailed proof can be found in [24]. The main intuition is that the messages on node variables can be proved to have statistical meanings (corresponding to certain probability functions) by induction. Based on this, the marginal posterior distributions can be easily proved. □

After computation of marginal posterior distribution, we can update model parameters by maximizing the posterior expectation of log likelihood (the maximization or M step in EM). Taking the marginal posterior distributions in (11) and (12) above as well as parameters for probabilities in (2) and (3) into the posterior expectation of log likelihood in (4), we can easily get the following parameter update formulas.

$$\rho = \frac{\sum\limits_{n|\mathcal{P}_n\neq\emptyset}\sum\limits_{y_n}\sum\limits_{y_{\mathcal{P}_n}} y_{\mathcal{P}_n} y_n P(y_n, y_{\mathcal{P}_n}|\mathbf{X},\mathbf{\Theta_0})}{\sum\limits_{n|\mathcal{P}_n\neq\emptyset}\sum\limits_{y_n}\sum\limits_{y_{\mathcal{P}_n}} y_{\mathcal{P}_n} P(y_n, y_{\mathcal{P}_n}|\mathbf{X},\mathbf{\Theta_0})} \tag{13}$$

$$\pi = \frac{\sum\limits_{n|\mathcal{P}_n=\emptyset}\sum\limits_{y_n} y_n P(y_n|\mathbf{X},\mathbf{\Theta_0})}{\sum\limits_{n|\mathcal{P}_n=\emptyset}\sum\limits_{y_n} P(y_n|\mathbf{X},\mathbf{\Theta_0})} \tag{14}$$

$$\boldsymbol{\mu}_c = \frac{\sum\limits_{n} \mathbf{x}_n P(y_n = c|\mathbf{X},\mathbf{\Theta_0})}{\sum\limits_{n} P(y_n = c|\mathbf{X},\mathbf{\Theta_0})}, c = 0, 1 \tag{15}$$

$$\Sigma_c = \frac{\sum\limits_{n} (\mathbf{x}_n - \boldsymbol{\mu}_c)(\mathbf{x}_n - \boldsymbol{\mu}_c)^T P(y_n = c|\mathbf{X},\mathbf{\Theta_0})}{\sum\limits_{n} P(y_n = c|\mathbf{X},\mathbf{\Theta_0})}, c = 0, 1 \tag{16}$$

Algorithm 2 shows the parameter learning process. First, we initialize parameters either with random values within reasonable range or with initial estimates based on training samples (e.g., the mean and covariance of features in each class). After parameters are initialized, the algorithm starts the iteration till parameters converge. In each iteration, it propagates messages first from leaves to root (steps 4-5) and then from root to leaves (steps 6-7). Marginal posterior distribution of node classes are then computed (steps 8-9). Based on this, the algorithm updates parameters (step 10).

Time complexity: The cost of Algorithm 2 mainly comes from the iterations. In each iteration, message propagation is done through tree traversal, which costs $O(N)$ (N is the total number of samples or tree nodes). It can also be seen easily that marginal probability computation and parameter update both have costs of $O(N)$. Thus, the total cost is $O(N \cdot I)$, where I is the number of iterations.

Algorithm 2 EM Algorithm for Hidden Markov Tree

```
Input:
  • $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$: cell sample feature matrix
  • $\mathcal{T}$: a reverse tree for spatial dependency
  • $\epsilon$: parameter convergence threshold
Output:
  • $\mathbf{\Theta} = \{\rho, \pi, \boldsymbol{\mu}_c, \Sigma_c | c = 0, 1\}$: set of model parameters
1: Initialize $\mathbf{\Theta_0}, \mathbf{\Theta}$
2: while $||\mathbf{\Theta_0} - \mathbf{\Theta}||_\infty > \epsilon$ do
3:     $\mathbf{\Theta_0} \leftarrow \mathbf{\Theta}$
4:     for each $y_n$ from leaf to root do
5:         Compute messages $f_n^i(y_n), f_n^o(y_n)$ by (5)-(6)
6:     for each $y_n$ from root to leaf do
7:         Compute messages $g_n^i(y_n), g_n^o(y_n)$ by (7)-(8)
8:     for each $y_n, 1 \leq n \leq N$ do
9:         // Compute marginal distributions:
              $P(y_n|\mathbf{X},\mathbf{\Theta_0}), P(y_n, y_{k\in\mathcal{P}_n}|\mathbf{X},\mathbf{\Theta_0})$ by (9)-(12)
10:    Update $\mathbf{\Theta}$ based on marginal distributions:
          $\mathbf{\Theta} \leftarrow \arg\max_{\mathbf{\Theta}} \mathbb{E}_{\mathbf{Y}|\mathbf{X},\mathbf{\Theta_0}} \log P(\mathbf{X}, \mathbf{Y}|\mathbf{\Theta})$ by (13)-(16)
11: return $\Theta$
```

Is the model unsupervised or semi-supervised? From discussions above, it is possible to learn HMT parameters in an unsupervised manner without training class labels. However, this relies on strong assumptions on data distributions. Particularly, it requires samples in different classes to be somehow distinguishable merely based on their feature distribution ($P(\mathbf{x}_n|y_n)$), since class transitional probability in dependency tree only enforces a partial order constraint between class nodes. This assumption can be violated in many real world applications where different classes cannot be easily distinguished via unsupervised feature clustering. In such cases, we can utilize training samples with class labels to initialize parameters of $P(\mathbf{x}_n|y_n)$, i.e., $\{\boldsymbol{\mu}_c, \Sigma_c | c = 0, 1\}$, by maximum likelihood estimation. In this way, initialized probability $P(\mathbf{x}_n|y_n)$ is a descent initial guess. In this case, the model learning is semi-supervised [26].

3.4 Class Inference

After learning model parameters, we can infer hidden class variables by maximizing the overall probability. In a traditional hidden Markov model, inference on hidden variables are done through Viterbi algorithm [18] based on dynamic programming. However, its computational cost is still very high for a large number of nodes (e.g., hundreds of millions). To address this challenge, we propose a greedy algorithm that guarantees correctness based on the partial order class constraint. Taking the logarithm of joint probability in Equation 1, we get the objective function in Equation 17 below.

$$\log P(\mathbf{X}, \mathbf{Y}) = \sum_{n=1}^{N} \log P(\mathbf{x}_n|y_n) + \sum_{n=1}^{N} \log P(y_n|y_{k\in\mathcal{P}_n}) \tag{17}$$

The goal of class inference is to assign a class label to each tree node such that the overall sum of log probability terms in Equation 17 is maximized. Each term in the summation can be considered as a reward. For instance, $\log P(\mathbf{x}_n|y_n)$ is the reward for assigning class y_n to node n (i.e., node reward), $\log P(y_n|y_{k\in\mathcal{P}_n})$ is the reward for assigning class y_n and $y_{k\in\mathcal{P}_n}$ to node n and its parents respectively (i.e., edge reward). Thus, class inference in HMT becomes a *node coloring problem*. Our goal is to find a node coloring to maximize the overall sum of rewards. In addition, the color must follow a partial order constraint, e.g., *dry* (class 0) nodes cannot follow *flood* (class 1) nodes, because otherwise, $P(y_n|y_{k\in\mathcal{P}_n}) = 0$. Therefore, we can enumerate all feasible node coloring through one bottom-up tree traversal, as described in Algorithm 3. We can initialize all node color as class 0 (negative class, e.g., *dry*), and gradually changed node colors from class 0 to class 1 from leaves to the root. When we change the color of a node, only the reward of the node itself, as well as the rewards of edges between the nodes to its parents and child will be updated, as illustrated in Figure 4. Thus, we can easily compute the gain of rewards when updating node colors (Δ_{LL}), and maintain the current cumulative gain ($g_{cur}(n)$) as well as the maximum cumulative gain (g_{max}) that we've come across so far. When we reach the root node, the maximum overall gain of rewards has been recorded. We can traverse the tree again to find its corresponding node coloring.

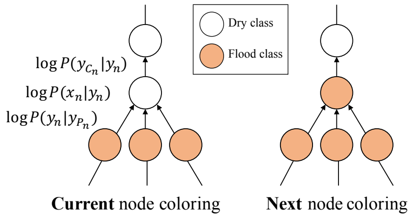


Figure 4: Illustration of class inference process

Algorithm 3 Class Inference for Hidden Markov Tree

```
Input:
  • $\mathcal{T}$: reverse tree for spatial dependency
  • $\Theta = \{\rho, \pi, \theta, \boldsymbol{\mu}_c, \Sigma_c | c = 0, 1\}$: set of model parameters
Output:
  • $\mathbf{Y} = [y_1, ..., y_n]$: inferred classes for all hidden nodes
Initialize $y_n \leftarrow 0$ for $1 \leq n \leq N$
Initialize $g_{cur}(n) \leftarrow 0$ for $1 \leq n \leq N$
Initialize $g_{max}(n) \leftarrow 0$ for $1 \leq n \leq N$
for each node $n$ in topological order from leaf to root do
    $y_n \leftarrow 1$
    $\Delta_{LL} \leftarrow \log\left(P(\mathbf{x_n}|y_n)P(y_{c_n}|y_n, y_{k\in\mathcal{S}_n})P(y_n|y_{k\in\mathcal{P}_n})\right)\Big|_{y_n=0}^{y_n=1}$
    // $y_{k\in\mathcal{P}_n} = \emptyset$ for leaf node $n$
    $g_{cur}(n) \leftarrow \sum_{k\in\mathcal{P}_n} g_{cur}(k) + \Delta_{LL}$
    $g_{max}(n) \leftarrow \sum_{k\in\mathcal{P}_n} g_{max}(k)$
    if $g_{max}(n) < g_{cur}(n)$ then
        $g_{max}(n) \leftarrow g_{cur}(n)$
Do breadth first tree traversal to find the frontier of maximum $g_{max}$
Set $y_n \leftarrow 0$ for nodes above the frontier
return $\mathbf{Y} = [y_1, ..., y_n]$, the class labels of all nodes
```

Time complexity analysis: The initialization steps cost $O(N)$, where N is the number of samples (tree nodes). Each iteration of the for loop has a constant cost, making the total cost $O(N)$. Similarly, the breadth first traversal and re-coloring in last step cost $O(N)$. Thus, the entire algorithm has a cost of $O(N)$.

4 EXPERIMENTAL EVALUATION

In this section, we compared our proposed method with baseline methods on both synthetic dataset and two real world datasets in classification performance. We also evaluated the computational scalability of our method on synthetic data with different sizes. Experiments were conducted on a Dell workstation with Intel(R) Xeon(R) CPU E5-2687w v4 @ 3.00GHz, 64GB main memory, and Windows 10. Candidate classification methods include:

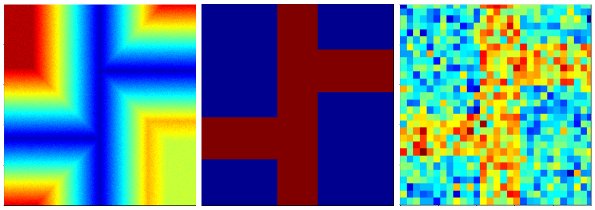
- **Non-spatial classifiers with raw features**: We tested decision tree (**DT**), random forest (**RF**), maximum likelihood classifier (**MLC**), and gradient boosted tree (**GBM**) in R packages on **raw** features (red, green, blue spectral bands).
- **Non-spatial classifiers with elevation features**: We tested **DT**, **RF** and **MLC** with additional elevation feature **(elev.)** We do not include **GBM** due to space limit.
- **Non-spatial classifier with post-processing label propagation (LP):** We also tested **DT**, **RF** and **MLC** on raw features but with post-processing on predicted classes via label propagation [27]. We used 4-neighborhood. We do not include **GBM** due to space limit.
- **Transductive SVM:** Since our method utilizes features of test samples, we included Transductive SVM (SVM-Light [10]), a semi-supervised tranductive method for fair comparison.
- **Markov random field (MRF):** We used open source implementation [22] based on the graph cut method [20].
- **Hidden Markov Tree (HMT):** We implemented HMT in C++.

Unless specified otherwise, we used default parameters in open source tools for baseline methods.

4.1 Synthetic Data

We first evaluated our proposed approach on synthetic data. Specifically, we generated a regular grid with 1000 by 1000 pixels. Elevations and classes (flood, dry) of pixels are shown in Figure 5(a-b). Feature values of pixels in two classes follow two one-dimensional Gaussian distributions with means $\mu_1 = 110, \mu_2 = 150$ and standard deviations $\sigma_1 = \sigma_2 = 20$ (these numbers are arbitrary). To reflect the spatial autocorrelation effect, we generated one common feature value for a group of contiguous pixels in a coarse resolution (50 × 50) (see Figure 5(c)). Training samples from two classes were generated based on the two Gaussian distributions of feature values.

Computational scalability: We measured the computational time costs of different components in our HMT algorithms on varying sizes of study area (from around 2 million pixels to around 20 million pixels). We also fixed the number of iterations as 3 when running algorithms on different data sizes. The time costs were measured in the average of 10 runs. Figure 6 shows the time costs of tree construction (Algorithm 1), parameter learning (Algorithm 2), and class inference (Algorithm 3) respectively. We can see that as the number of pixels increases, time costs of all algorithms are increasing. The parameter learning part takes the vast majority of



(a) Elevation map **(b) Class map (red for flood, blue for dry)** **(c) Feature map**

Figure 5: Illustration of synthetic data (best viewed in color)

time costs. Its time costs increase linearly with data sizes, because the message propagation in each iteration is done through tree traversal operations, which has a linear time complexity. Overall, our algorithms cost less than 5 minutes on a synthetic data with 20 million samples.

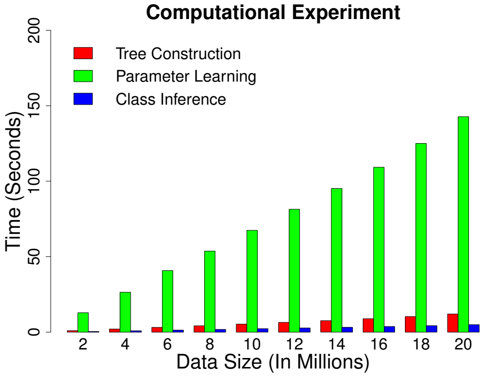
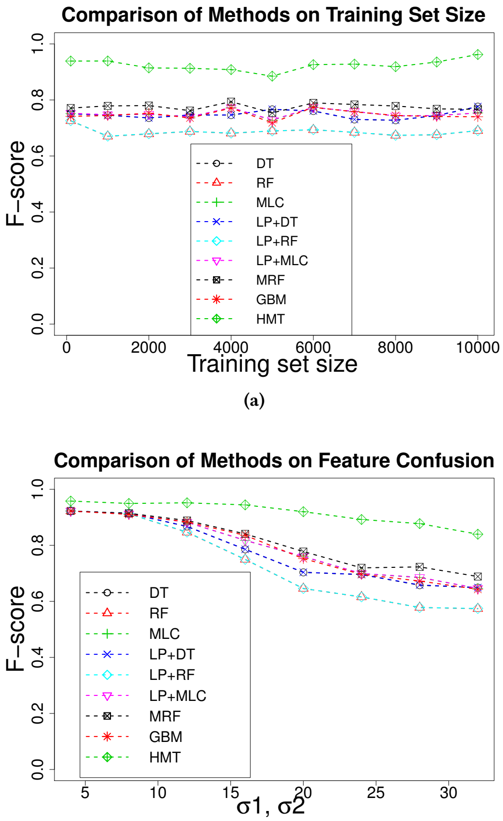


Figure 6: Computational time costs of HMT on different data sizes

Classification performance: We compared the F-score of different methods on test pixels with different parameter settings of synthetic data generation. We exclude pre-processing and post-processing methods because our synthetic data generation cannot simulate the real feature textures. In the first setting, we conducted comparison on varying numbers of training pixels from 10, 1000, to 10000. Results in Figure 7(a) showed that the classification performance of different methods were relatively stable (easily reaching plateau) for different training set sizes. The reason was probably that one dimensional Gaussian distributions on feature values in two classes were very easy to learn. In the second setting, we fixed other parameters and varied the standard deviations σ_1, σ_2 of feature values in two classes. The higher the values were, the more confusion (Bayes error) there were between two classes. Results of different methods in Figure 7(b) showed that as σ_1, σ_2 increase, the classification performance of all methods degraded, but our HMT model persistently outperformed other baseline methods, due to incorporating anisotropic spatial dependency across locations.



(a)

(b)

Figure 7: Classification performance comparison across methods on synthetic data

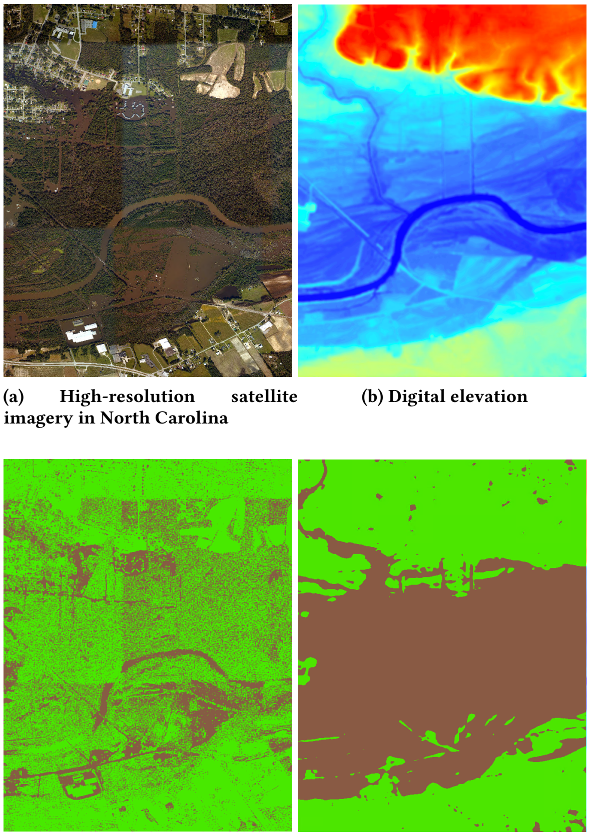
4.2 Hurricane Matthew Floods 2016

Here we validated our approach in flood inundation extent mapping during Hurricane Mathew, NC, 2016. We used high-resolution aerial imagery from NOAA National Geodetic Survey [14] as explanatory features (three spectral band features including red, green, blue), and digital elevation map from the University of North Carolina Libraries [16]. All imagery data were re-sampled into a resolution of 2 meters. A test region with 1743 by 1349 pixels was used. A training set with 10000 pixels (5000 *dry* and 5000 *flood*) were manually labeled outside the test region, and 94608 test pixels (47092 *dry*, 47516 *flood*) were labeled within the test region.

Classification performance comparison: We compared methods on precision, recall, and F-score. Results were summarized in Table 1. We can see that decision tree, random forest, gradient boosted tree, and maximum likelihood classifier all performed poorly on raw features, with overall F-score around 0.7. Adding post-processing through label propagation slightly impaired performance. For instance, adding label propagation (LP) to decision tree results improved the recall of the *dry* class but degraded the recall of the *flood* class. Markov random field and Transductive SVM had comparable results with decision tree. Adding elevation features

Table 1: Comparison on Hurricane Mathew Flood data

Classifiers	Class	Precision	Recall	F	Avg. F
DT+Raw	Dry	0.62	0.84	0.71	0.65
	Flood	0.76	0.48	0.59	
RF+Raw	Dry	0.59	0.96	0.73	0.61
	Flood	0.90	0.33	0.49	
GBM+Raw	Dry	0.69	0.76	0.72	0.71
	Flood	0.74	0.67	0.70	
MLC+Raw	Dry	0.64	0.93	0.76	0.69
	Flood	0.88	0.48	0.62	
DT+elev.	Dry	0.99	0.55	0.71	0.76
	Flood	0.69	0.99	0.82	
RF+elev.	Dry	0.99	0.66	0.79	0.82
	Flood	0.74	0.99	0.85	
MLC+elev.	Dry	0.84	0.90	0.87	0.87
	Flood	0.89	0.84	0.86	
DT+LP	Dry	0.61	0.92	0.74	0.65
	Flood	0.85	0.43	0.57	
RF+LP	Dry	0.57	0.99	0.72	0.57
	Flood	0.99	0.26	0.42	
MLC+LP	Dry	0.64	0.97	0.77	0.69
	Flood	0.95	0.46	0.62	
MRF	Dry	0.62	0.99	0.76	0.67
	Flood	0.98	0.41	0.58	
TSVM	Dry	0.62	0.86	0.72	0.66
	Flood	0.78	0.49	0.60	
HMT	Dry	0.93	0.99	0.96	0.96
	Flood	0.99	0.93	0.96	



(a) High-resolution satellite imagery in North Carolina

(b) Digital elevation

(c) Decision tree result

(d) HMT result

Figure 8: Results on Mathew flood mapping (flood in brown, dry in green, best viewed in color)

improved the overall classification performance dramatically for decision tree, random forest, and maximum likelihood classifier. The reason is that most *flood* pixels have lower elevations than *dry* pixels. However, its performance was still quite inferior (less than 0.87 in F-score) compared with our hidden Markov tree (0.96 in F-score), probably because models based on absolute elevation values cannot generalize well to the test region.

Some visualization of classification results were shown in Figure 8. The spectral features and elevation values were shown in Figure 8(a-b). Results of decision tree were in Figure 8(c), which only identified *flood* pixels with open surface, and mistakenly classified the vast majority of *flood* pixels below tree canopies (the spectral features of trees indicated the *dry* class if not considering spatial dependency with nearby pixels). In contrast, our HMT model correctly identified most of the *flood* pixels, even if the flood water was below tree canopies. The reason is that our HMT incorporates the anisotropic spatial dependency across pixel locations (if a location is *flood*, its nearby lower locations should also be *flood*).

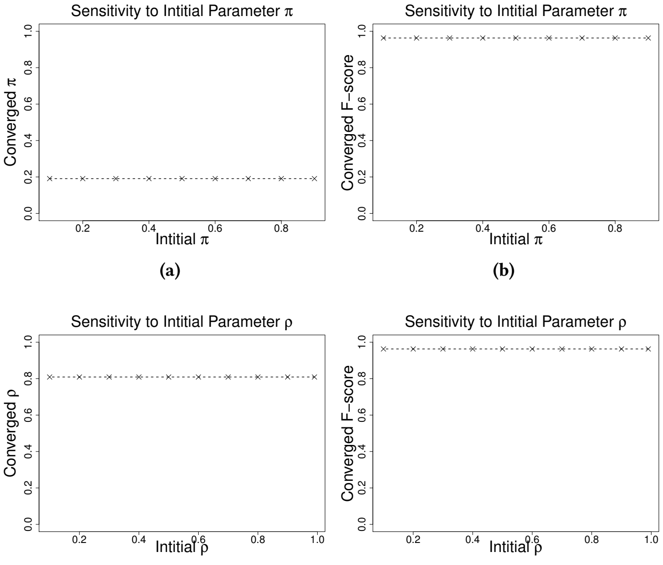
Sensitivity of HMT to initial parameters: We conducted sensitivity of our HMT model to different initial parameter values on prior class probability π and class transitional probability ρ (the parameters of $\{\boldsymbol{\mu}_c, \Sigma_c | c = 1, 2\}$ were initialized based on maximum likelihood estimation on the training set). First, we fixed initial $\rho = 0.99$ and varied initial π from 0.1 to 0.9. Results of converged value of ρ together with final F-score were shown in Figure 9(a-b). It can be seen that our HMT model was quite stable with different initial π values. Similarly, we fixed initial $\pi = 0.5$, and varied initial ρ from 0.2, 0.3, to 0.99. Results in Figure 9(c-d) showed the same trend. In practice, we can select an initial π value around 0.5 and a relatively high initial ρ value such as 0.9 (because *flood* pixels' neighbor is more likely to be *flood* due to spatial autocorrelation).

Parameter iterations and convergence in HMT: Here we fixed the initial $\pi = 0.5$ and initial $\rho = 0.99$, and measured the parameter iterations and convergence. Our convergence threshold was set 0.001%. The values of π and ρ at each iteration were summarized in Figure 10 (we omitted $\boldsymbol{\mu}_c, \Sigma_c$ because there were too many variables). The parameters converged after 10 iterations.

4.3 Hurricane Harvey Floods 2017

The high-resolution earth imagery we used were from Plant Labs. Inc. with red, green, and blue bands in 3 meter resolution, and the digital elevation data was from Texas natural resource management department. We manually collected a training set with 5000 flood samples and 5000 dry samples. We selected a test scene with 4174 rows and 4592 columns, within which we manually labeled 74305 flood samples, and 52658 dry samples as the test set.

We compared different methods on precision, recall, and F-score. Results were summarized in Table 2. We can see that decision tree,



(a) (b)

(c) (d)

Figure 9: Sensitivity of HMT to different initial parameters π and ρ

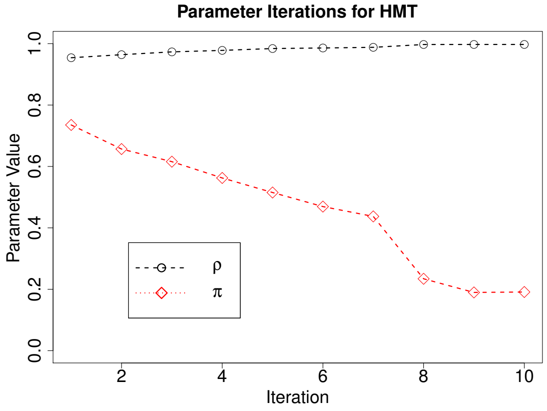


Figure 10: Parameter iterations and convergence in HMT

random forest, gradient boosted tree, and maximum likelihood classifier all performed poorly on raw features, with overall F-score less than 0.75. Adding additional elevation feature improved the classification accuracy slightly, but the overall F-score was still below 0.8. The reason is probably that models based on absolute elevation values do not generalize well to the test area. Similarly, adding label propagation in post-processing and the MRF model barely improved classification performance compared with decision tree and random forest on raw features, because the errors were mostly systematic instead of salt-and-pepper noise. TSVM performed even worse than supervised methods without unlabeled samples, which was somehow surprising. In contrast to baseline methods, our hidden Markov tree achieved superior performance with around 0.95 F-score on both classes. Visualization of some results were shown in Figure 11. We can see that our HMT model significantly outperformed decision tree on flood locations under tree canopies, similar to the results in Figure 8.

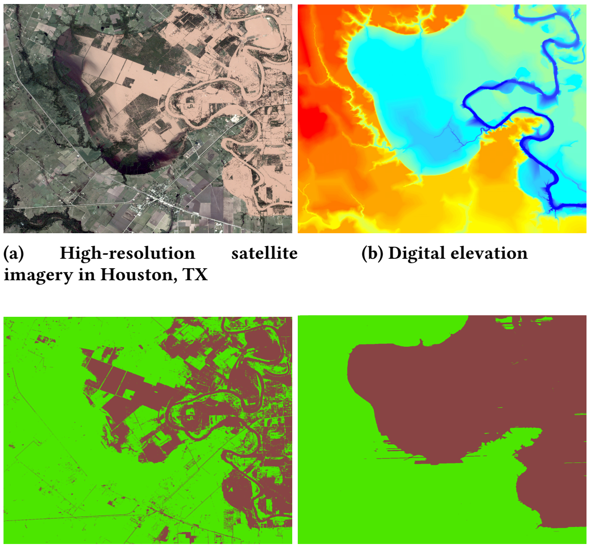
Table 2: Comparison on Harvey Flood Data

Classifiers	Class	Precision	Recall	F	Avg. F
DT+Raw	Dry	0.58	0.88	0.70	0.69
	Flood	0.87	0.56	0.68	
RF+Raw	Dry	0.62	0.96	0.76	0.74
	Flood	0.95	0.59	0.73	
GBM+Raw	Dry	0.56	0.80	0.66	0.66
	Flood	0.80	0.56	0.66	
MLC+Raw	Dry	0.63	0.93	0.75	0.74
	Flood	0.93	0.61	0.73	
DT+elev.	Dry	0.61	0.99	0.76	0.74
	Flood	0.99	0.56	0.72	
RF+elev.	Dry	0.66	0.99	0.79	0.79
	Flood	0.99	0.65	0.78	
MLC+elev.	Dry	0.65	0.97	0.78	0.77
	Flood	0.97	0.62	0.76	
DT+LP	Dry	0.59	0.90	0.71	0.70
	Flood	0.89	0.56	0.69	
RF+LP	Dry	0.63	0.97	0.76	0.75
	Flood	0.97	0.59	0.74	
MLC+LP	Dry	0.63	0.94	0.76	0.75
	Flood	0.93	0.61	0.74	
MRF	Dry	0.63	0.94	0.75	0.74
	Flood	0.94	0.61	0.74	
TSVM	Dry	0.55	0.67	0.60	0.63
	Flood	0.72	0.61	0.66	
HMT	Dry	0.91	0.98	0.94	0.95
	Flood	0.98	0.93	0.95	

5 RELATED WORK

Over the years, various techniques have been developed to incorporate spatial properties into classification algorithms for earth imagery data. Many methods are based on preprocessing and post-processing, including neighborhood window filters [3, 6], spatial contextual variables and textures [17], spatial autocorrelation statistics [9], morphological profiling [1], spatial-spectral classifiers [21, 23] and object-based image analysis [7]. Markov random field model explicitly captures spatial dependency, but the dependency is undirected [13]. [11] proposes a spatial classification model that captures directed spatial dependency on classes but assumes dependency to follow a total order. Deep learning methods have recently been applied to earth imagery classification [25] such as land cover mapping [8], target recognition and scene identification. To the best of our knowledge, none of these existing works focuses on incorporating anisotropic spatial dependency in partial order constraints, which is important in hydrological applications such as flood mapping.

Hidden Markov models have been extensively studied in the signal processing literature [18]. Learning and inference of hidden


(a) High-resolution satellite imagery in Houston, TX

(b) Digital elevation

(c) Decision tree result

(d) HMT result

Figure 11: Results on Harvey flood mapping (flood in brown, dry in green, best viewed in color)

Markov models are often based on EM algorithms and message propagation (sum-and-product algorithm) [12]. [19] proposes a message propagation algorithm called “upward-downward" on a dependence tree structure. [5] proposes a wavelet-domain hidden Markov tree to model dependency in the two-dimensional time-frequency plane. The model is used to characterize properties of wavelet coefficients in signal processing such as clustering, persistence, and compression, which is dramatically different from our HMT model which captures spatial dependency in the geographic space based on topography.

6 CONCLUSIONS AND FUTURE WORKS

In this paper, we propose hidden Markov tree (HMT), an anisotropic spatial classification model for flood mapping on earth imagery. Our HMT model explicitly captures directed class dependency with partial order constraint by a reverse tree structure in the hidden class layer. We also propose efficient algorithms for reverse tree construction, parameter learning and class inference. Evaluations on both synthetic data and real world data show that our HMT algorithms are scalable to large data sizes, and can utilize the partial order spatial dependency to reduce classification errors.

In future work, we plan to extend our HMT model for spatially non-stationary data. To this end, we need to generalize the tree structure in hidden class layer into poly-tree with each sub-tree for a spatial zone. We also plan to explore integration of deep learning framework with our geographical HMT.

REFERENCES

[1] Jón Atli Benediktsson, Jón Aevar Palmason, and Johannes R Sveinsson. 2005. Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE Transactions on Geoscience and Remote Sensing* 43, 3 (2005), 480–491.

[2] PA Brivio, R Colombo, M Maggi, and R Tomasoni. 2002. Integration of remote sensing data and GIS for accurate mapping of flooded areas. *International Journal of Remote Sensing* 23, 3 (2002), 429–441.

[3] Raymond H Chan, Chung-Wa Ho, and Mila Nikolova. 2005. Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization. *Image Processing, IEEE Transactions on* 14, 10 (2005), 1479–1485.

[4] Don Cline. 2009. *Integrated Water Resources Science and Services: an Integrated and Adaptive Roadmap for Operational Implementation.* Technical Report. National Oceanic and Atmospheric Administration.

[5] Matthew S Crouse, Robert D Nowak, and Richard G Baraniuk. 1998. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on signal processing* 46, 4 (1998), 886–902.

[6] S Esakkirajan, T Veerakumar, Adabala N Subramanyam, and CH PremChand. 2011. Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter. *Signal Processing Letters, IEEE* 18, 5 (2011), 287–290.

[7] GJ Hay and G Castilla. 2008. Geographic Object-Based Image Analysis (GEOBIA): A new name for a new discipline. In *Object-based image analysis*. Springer, 75–89.

[8] Xiaowei Jia, Ankush Khandelwal, Guruprasad Nayak, James Gerber, Kimberly Carlson, Paul West, and Vipin Kumar. 2017. Incremental dual-memory lstm in land cover prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 867–876.

[9] Zhe Jiang, Shashi Shekhar, Xun Zhou, Joseph Knight, and Jennifer Corcoran. 2015. Focal-test-based spatial decision tree learning. *IEEE Transactions on Knowledge and Data Engineering* 27, 6 (2015), 1547–1559.

[10] T. Joachims. 1999. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola (Eds.). MIT Press, Cambridge, MA, Chapter 11, 169–184.

[11] Ankush Khandelwal, Varun Mithal, and Vipin Kumar. 2015. Post Classification Label Refinement Using Implicit Ordering Constraint Among Data Instances. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*. 799–804.

[12] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory* 47, 2 (2001), 498–519.

[13] Stan Z Li. 2009. *Markov random field modeling in image analysis*. Springer Science & Business Media.

[14] National Oceanic and Atmospheric Administration. [n. d.]. Data and Imagery from NOAA's National Geodetic Survey. https://www.ngs.noaa.gov. ([n. d.]).

[15] National Oceanic and Atmospheric Administration. 2018. National Water Model: Improving NOAA's Water Prediction Services. http://water.noaa.gov/documents/wrn-national-water-model.pdf. (2018).

[16] NCSU Libraries. 2018. LIDAR Based Elevation Data for North Carolina. https://www.lib.ncsu.edu/gis/elevation. (2018).

[17] Anne Puissant, Jacky Hirsch, and Christiane Weber. 2005. The utility of texture analysis to improve per-pixel classification for high to very high spatial resolution imagery. *International Journal of Remote Sensing* 26, 4 (2005), 733–745.

[18] Lawrence R Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286.

[19] O Ronen, JR Rohlicek, and M Ostendorf. 1995. Parameter estimation of dependence tree models using the EM algorithm. *IEEE Signal Processing Letters* 2, 8 (1995), 157–159.

[20] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. 2006. A comparative study of energy minimization methods for markov random fields. In *European conference on computer vision*. Springer, 16–29.

[21] Yuliya Tarabalka, Jón Atli Benediktsson, and Jocelyn Chanussot. 2009. Spectral–spatial classification of hyperspectral imagery based on partitional clustering techniques. *IEEE Transactions on Geoscience and Remote Sensing* 47, 8 (2009), 2973–2987.

[22] The Middlebury Computer Vision Pages. 2018. C++ Source Code of MRF. http://vision.middlebury.edu/MRF/code/. (2018).

[23] Liguo Wang, Siyuan Hao, Qunming Wang, and Ying Wang. 2014. Semi-supervised classification for hyperspectral imagery based on spatial-spectral label propagation. *ISPRS Journal of Photogrammetry and Remote Sensing* 97 (2014), 123–137.

[24] Xie, Miao and Jiang, Zhe and Sainju, Arpan Man. 2018. Geographical Hidden Markov Tree for Flood Extent Mapping (With Proof Appendix). https://arxiv.org/abs/1805.09757. (2018).

[25] Liangpei Zhang, Lefei Zhang, and Bo Du. 2016. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine* 4, 2 (2016), 22–40.

[26] Xiaojin Zhu. 2005. Semi-supervised learning literature survey. (2005).

[27] Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. (2002).