VisGBT: Visually Analyzing Evolving Datasets for Adaptive Learning

Keke Chen Fengguang Tian

Department of Computer Science and Engineering, Wright State University
Dayton, OH 45435, USA
{keke.chen,tian.9}@wright.edu

***Abstract*— Many machine learning problems involve changes in both feature distribution and label distribution, such as domain adaptation and learning drifting concepts from data streams. Correctly detecting, identifying, and understanding the changes of data distributions can help us properly select data samples or algorithms for learning models. However, since the training datasets are often in high dimensionality and large size, it has been difficult to effectively analyze them. Furthermore, the joint distribution between features and labels makes the problem more difficult to handle. In this paper, we propose a visual analysis method (VisGBT) that combines the gradient-boosting-trees (GBT) modeling method, regression analysis, and multi-dimensional visualization to capture the mismatches between datasets and models. The GBT model consists of a series of trees with a predefined number of terminal (leaf) nodes per tree. These terminal nodes partition the high dimensional space with a few most informative features to minimize the label prediction error. VisGBT maps various kinds of detailed model information to the terminal node matrix (TNM) and visualizes it with an appropriate design. With this visual analysis method, we can easily find out the detailed differences between datasets with the help of a learned model. We will illustrate the use of various visual patterns and in particular show how this method can help us analyze domain similarity for domain adaptation.**

I. Introduction

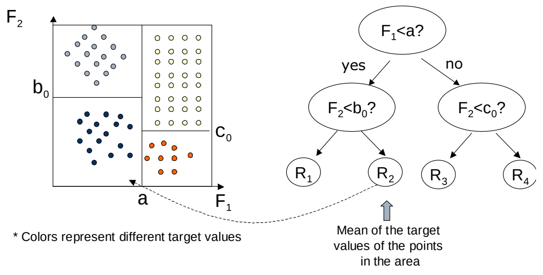
In traditional machine learning problems, we often assume that the underlying data distribution does not change when we apply the learned model to new data. However, the assumption does not hold when we want to learn models for similar domains with significant differences or for data streams where concepts (models) keep changing. In such a setting, an old model may need to be renovated to adapt to the changes of data distribution, or datasets from one domain is adopted by another domain for learning models, i.e., so called *adaptive learning*. However, model renovation or adaptation may be difficult and costly to perform for several reasons.

First of all, renovating models often involves significant cost in training data collecting, model training, model validation, and deployment. If the change is not significant, or caused by noises, we may want to apply existing models to different domains or keep using existing models for a data stream. It is thus important to help domain experts understand how the data change happens and determine what model renovation is required. Visualization will be an ideal tool for serving such a purpose under a collaborative environment.

Second, since in supervised learning labeled training examples are often expensive to obtain $-$ we need domain experts to manually judge and label the training examples, it is often desired to reuse existing labeled training data from *similar* domains to improve the model in the target domain [1], [18], [4], [20], [25], [12], [10]. However, blindly migrating datasets from one domain to another domain may result in useless models. An important step to successfully perform domain adaptation is to understand how similar the candidate domains are [20], [10], where visualization tools can help.

In a collaborative mining scenario, we want domain experts (maybe from different application domains) to understand and monitor the mismatch between the data and the model or between datasets, and thus be able to properly renovate the model or adapt to changes in minimum cost. In this paper, we propose the VisGBT method to visualize the data distribution based the Gradient Boosting Trees (GBT) [14] learning model. There is extensive research in visualizing multidimensional data [30] and visualizing learning models, such as decision tree models [2]. However, to our knowledge, there is no well-developed visualization method for visualizing the change of data distribution for machine learning. The challenge of this problem lies in two aspects. First, visualizing multidimensional data in general is well recognized as a difficult problem. It is still one of the major research challenges in visual analytics [30], [23], [28]. Second, since each training example consists of a feature vector $\mathbf{x}_i$ and a label, or target value, y_i, it is difficult to design appropriate visualization methods to capture the differences in the joint distribution of features and labels in a useful and scalable manner.

The basic idea of our proposed approach is to use the GBT learning method to partition the multidimensional feature space of one training dataset, while minimizing the prediction error on its label distribution. With the trained model, we can map another dataset onto this partitioned subspaces to understand the difference of both sample distributions and label distributions between the two datasets. The unique benefit is that GBT will automatically pick up a few most important features among possibly hundreds or thousands of features, in modeling the corresponding label distribution. Thus, we can explore the high-dimensional space with much less number of features that are most relevant to the model. Concretely, a GBT model consists of a series of regression trees [15] and each tree has a fixed number of terminal nodes, i.e., leaf nodes. Figure 1 shows a typical regression tree and how it cuts the space. Since each training example will be directed to one and only one of the terminal node for each tree, the terminal node becomes the most informative unit in the model.



F2
b0
c0
a
F1
F1<a?
yes
no
F2<b0?
F2<c0?
R1
R2
R3
R4
Mean of the target values of the points in the area
* Colors represent different target values

Fig. 1. A perfectly fitted tree perfectly partitions the data space according to the joint feature-label distribution.

Our visual analysis method is based on the terminal node matrix (TNM) (the number of trees $\times$ the number of terminal nodes per tree). We can map various types of information derived from the terminal node matrix (TNM) to a "bar matrix" and visually compare different sets of information on the bar matrix. In particular, we will describe how to apply TNM-based visualizations to characterize the fitness of a model to datasets and analyze domain similarity for domain adaptation.

The rest of the paper is organized as follows. Section 2 gives some background knowledge about the gradient-boosting-trees modeling method. In Section 3, we start with the problems with the mismatch of data and model, and then develop a visual analysis method based on the GBT model to address these problems. Section 4 is dedicated to some use cases and show how this visual analysis method can be possibly used to improve the understanding and capture the changes of the underlying datasets. Section 5 will show one application of the visual analysis method to a real dataset for domain adaption in the ranking problem. Section 6 will give some related work in domain adaptation, drifting concept learning, and multidimensional data visualization.

II. Preliminary

In this section we will give the definition of the training data and briefly cover the knowledge of the regression tree modeling and the gradient-boosting-trees method, to help understand the visual analysis methods. The reason of using the GBT method is for its unique benefits (1) The boosting methods, including the GBT method, give high-quality models, with low overfitting [27], [14], among the best learning algorithms, such as Support Vector Machine(SVM). (2) Although it is designed for regression modeling, the GBT method can also be applied to solve classification problems as well. (3) Each regression tree in the GBT model partitions the feature vector space geometrically, which makes visualization possible [15], while it is hard to visualize learning models like SVM.

A. Training Data and Regression Modeling

In supervised learning, i.e., regression or classification, each training example is represented as $\{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}_i$ represents the feature vector describing the features associated with the real application, and y_i is the target value that we want to predict. The difference between regression modeling and classification modeling is the type of labels. For regression, y_i is continuous or ordinal (ordinal regression [16]) in a confined domain. For classification y_i is categorical and the number of classes is often small. In this paper, we assume the features are numerical, i.e., continuous or ordinal values, and categorical features have been transformed to boolean features, so that the described method can be applied. As a convention, we will use capitals to represent variables and lower cases to represent constants.

B. Regression Tree Modeling

Figure 1 shows a sample regression tree, which is a binary tree with one predicate at each internal node. The predicate consists of a variable (feature) and a splitting value, typically in form of $F < \tau$?. In such a tree, an internal tree node partitions the training data that reach the node into two parts with the predicate. The tree is grown with a top-down manner, i.e., starting from the root and terminating when the fixed number of terminal nodes, i.e., leaf nodes, is reached. In the following, we describe how the training algorithm decides which feature and splitting value are used for growing child nodes.

Splitting a leaf node to grow a tree should give some kind of "gain", namely, optimizing the goal of regression, i.e., minimizing the square error between the predicted value and the target value. We assume that there are n_i training records reaching the node i, each of which, $\mathbf{x}_j$, has a target value r_{ij} to fit at the node i. $r_{ij} = y_j$ if the current node is the root, otherwise, r_{ij} is the residual by fitting the parent node. It represents how well this example is fit so far from the existing part of tree. The best-effort predictor for all records falling onto the current node is the mean of all r_{ij}, i.e., $\hat{r}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} r_{ij}$ [15]. With $\hat{r}_i$, the square error E for the current node is

$$E = \sum_{j=1}^{n_i} (r_{ij} - \hat{r}_i)^2$$

Finding the Best Split for a Node.
Let F_p denote the feature and $v_{p,q}$ is a feasible value for F_p. $(F_p, v_{p,q})$ partitions the data into two parts: those records with $F_p < v_{p,q}$ go to the left subtree and the rest records go to the right subtree. After performing this partition, similarly, we can get the square error E_L for the left subtree, and E_R for the right subtree. We define the gain by splitting this node as $gain = E - E_L - E_R$. By scanning through all possible features and feasible splitting values for each feature, we can find the best splitting condition, which satisfies

$$argmin_{(F_p, v_{p,q})}\{E_L + E_R\}, \text{for all possible } p,\ q.$$

With the above criterion, a greedy but efficient search procedure is often used to determine the leaf node $-$ find the one that brings the highest gain among all existing leaf nodes for splitting. It is a hill-climbing procedure, which does not necessarily result in the globally optimal tree. However, it is efficient since the cost is linear to the number of features and often the result is very satisfactory. Figure 1 shows a perfectly fitted tree to the underlying target value distribution. To extend it to general multidimensional cases, we can understand that each node represents a "multidimensional bounding box" defined by the disjointed partitioning conditions along the path from the root to that node. For example, the leaf node labeled with R_2 in Figure 1 is defined by the bounding box $F_1 < a \wedge F_2 < b_0$.

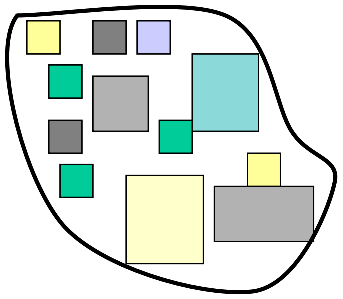


Fig. 2. Imagine the multidimensional space is partitioned by a regression tree and each block has different target values which are represented by different colors.

In Figure 2, we use the small blocks to illustrate the local areas in the high-dimensional space (i.e., "the high-dimensional bounding boxes" in regression tree modeling) that are covered by the training data. Different colors represent different target values for the blocks. In regression tree modeling, each leaf node tries to approximately model one of these blocks, and each internal node groups a few nearby blocks with close target values.

Calculating Leaf Node Response.
In the above algorithm, during the growing phase the predicted value $\hat{r}_i$ for the node i is recorded, and the residuals $r_{ij} - \hat{r}_i$ are used as the new target values for its child nodes. Let $\{\hat{r}_i^{(\omega)}$, node i in the path from the root to any node $\omega\}$ denote the predicted values along the path from the root to the leaf node t. The final predicted response R_ω for the leaf node ω is

$$R_\omega = \sum_i r_i^{(\omega)}$$

When a training example (a feature vector) is directed to the terminal node ω according to the conditions, the value R_ω is given as the predicted target value of the regression tree.

Note that regression tree can be used to model both regression and classification problems. For a classification problem, there are methods to transform the class labels or to change the loss function [15]. For simplicity, we will only discuss regression modeling in this paper.

C. Gradient Boosting Trees

Gradient boosting trees are a series of regression trees, denoted by $h_i(\mathbf{x})$. The final function is based on these regression trees.

$$H(\mathbf{x}) = \sum_{i=1}^{k} \gamma_i h_i(\mathbf{x})$$

where γ_i is the learning rate, which is often small, e.g., 0.05. A formal description of the training algorithm can be found in the literature [14]. The GBT learning method trains the k-th tree, based on the previous trees h_j, $1 \le j < k$, with a set of random samples from the training dataset. The steps can be briefly described as follows.

1) randomly sample the training data with certain sample rate to get a subset of training examples $\mathcal{S}_k$;

2) set the target r_i of the example in $\mathcal{S}_k$ to the original target y_i for k=1, or to the residual of the previous trees h_j, for $k > 1$, $1 \le j < k$, $r_i = y_i - \sum_{j=1}^{k-1} \gamma_j h_j(\mathbf{x}_i)$.

3) train the regression tree h_k with the examples $\{(\mathbf{x}_i, r_i)\}$, $\mathbf{x}_i \in \mathcal{S}_k$.

Note these trees are highly related $-$ the tree i is trained to fit the residuals from the previous $i-1$ trees. Therefore, it is meaningful to observe the data distribution over a series of trees.

III. VisGBT: A Visualization Approach for Analyzing Learning Models

To trace the change of data distribution for models, two key distributions should be monitored: the feature vector distribution and the label distribution. We will discuss how to observe these distributions and derive other information with the visualized Terminal Node Matrix (TNM). First, we will present the basic method for using the TNM to observe the overall and local fitness of the model. Three simple synthetic datasets will be used to demonstrate how a TNM looks like according to the underlying data distribution. Second, we show the basic method for detecting the change of feature vector distribution and label distribution.

A. Properties of Trees in the Model

The rationale of the TNM-based visualization method is based on the properties of regression tree and the GBT model. Let's revisit the meaning of a node in a regression tree. Let d represent the total number of features, and $range(F_i)$ represent the value range of F_i selected by the conditions along the path from the root to the node. Note that the same feature may appear several times in the path, which results in a continuous range. A typical range is like $v_{i,1} < F_i \le v_{i,2}$, where $v_{i,1}$ can be $-\infty$ and $v_{i,2}$ can be $+\infty$, according to the splitting process. If the feature does not appear in the path from the root to the node, we consider $range(F_i)$ containing the full range of F_i values. Therefore, we have the following Lemma

Lemma 1: A node represents a partition $\bigcup_{i=1}^{d} range(F_i)$ and this partition is continuous.
Based on the definition of the tree, we can also derive

Lemma 2: The entire feature space is completely partitioned by the terminal nodes in the same tree.

From the construction process described in the Preliminary section, it is straightforward to derive these two lemmas. We will ignore the proofs.

In addition, the GBT modeling process also implies the following properties

- A new tree tries to fit the residuals of the target values from the previous trees for the sampled training data. Ideally, if there exists a meaningful model, the residuals are decreasing with the increasing number of trees.
- Since we use the same sample rate crossing trees, each tree uses a same number of samples, which allows us to normalize the data distribution crossing trees.

In summary, with appropriate normalization and alignment, the above properties allow us to use the terminal node matrix to model the overall distribution of the feature vectors and the labels.

B. *The Basic Visual Design*

Concretely, we use the "bar matrix" to represent the metrics defined on the terminal node matrix. Figure 3 shows the basic design. In the bar matrix, columns represent the nodes and rows represents the trees. For each tree, we order the nodes by their positions from the left to right and encode them by the sequence $[1..m]$, where m is the number of terminal nodes per tree (Figure 4). The node ID is mapped to the column ID. Each cell in the the matrix can be used to show either a pair of metrics to contrast them or to show just one metric to see the overall patterns crossing nodes and trees. The lengths of the bars can be normalized according to the overall value distribution of the represented metric. The advantage of bar matrix is that we can accommodate a large number of trees easily and we can easily see the overall distribution crossing nodes and trees, while the detailed information of each node can also be observed under its context, i.e., the neighboring nodes in the same tree and crossing neighboring trees. Note that we haven't discussed what the bars represent $-$ the concrete design will depend on the semantics we want to represent. We leave the details in later discussion.

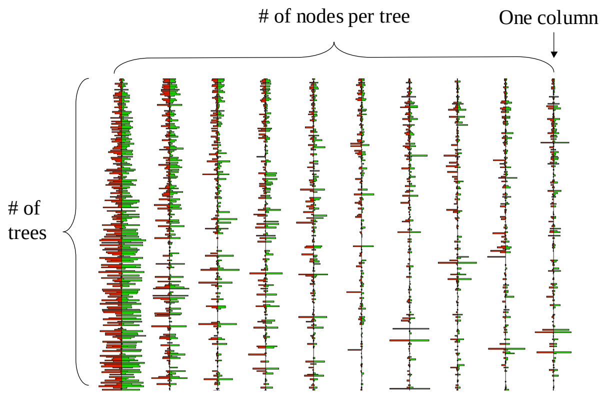


Fig. 3. Mapping metrics to the terminal node matrix

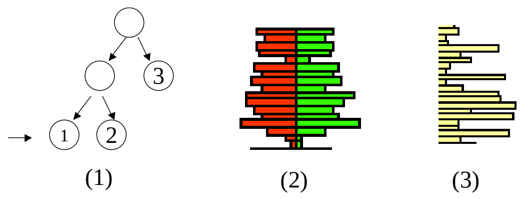


Fig. 4. (1) Terminal nodes (leaf nodes) are labeled in the order from left to right. (2) showing two contrasting metrics for each cell. (3) showing one metric for each cell.

The bar-matrix based visualization can also be combined with our previously developed VISTA cluster rendering system [8] to explore the subset of points that are covered by the target terminal node (Figure 5. In the VISTA visualization, only the selected features in the path from the root to the terminal node are used in exploration, while other features can certainly be added later if the user is interested in them. Each point in the VISTA visualization is colored according to its target value (for tree 1, it is the original target value; for tree i, $i > 1$, it is the residual from previous trees). Since the focus of this paper is on the VisGBT method, we will ignore the detail of the VISTA visualization. Interested user can refer to our previous papers [8], [9]. In addition, the specific information about the node will be shown when the user points to a particular node.

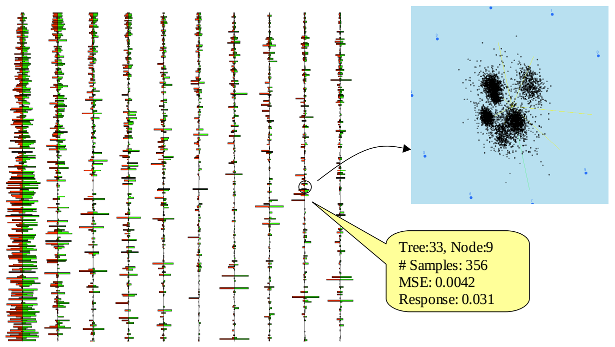


Fig. 5. Using the VISTA system to explore the subsect of points in the selected terminal node; The conditions for generating the terminal node can be observed when the user pointing to the node.

C. *Basic Visual Analysis Methods*

In this section, we demonstrate a few use cases for the TNM-based model visualization method. We will show how to use the visualization to analyze the fitness of the model to a dataset. Note the datasets can be the training dataset that is used to generate the model, or a pair of datasets, the training dataset and the candidate dataset for comparison. We will particularly discuss how to compare two datasets in next section.

1) Basic Metrics Visualized with TNM.: Three basic metrics can be visualized with the TNM structure. The first one is the number of samples falling on the terminal node, i.e., the feature vector distribution, which is visualized with the single-bar method ((3) in Figure 4). We name it Feature-Mass visualization. This visualization gives an overall feeling of data distribution. The longer the bar is, the more samples the terminal node has.

The second one is the metric representing model fitness. Let R_i represent the response value of the terminal node i, and there are n_i samples reaching this terminal node. Let r_{ij} represent the target value of the point j. We define the mean square error (MSE) of the terminal node i as follows.

$$MSE_i = \frac{1}{n_i}\sum_{j=1}^{n_i}(r_{ij} - R_i)^2$$

Since MSE is the loss function the regression modeling wants to minimize, MSE_i is used to represent the fitness of this node to the training examples. Note that a perfect node splitting in growing the tree will result in zero MSE in the two new terminal nodes. We name such a visualization as a MSE visualization. It is also visualized with the single-bar method.

The third one is the combination of the feature vector distribution and model fitness, which is visualized with the contrasting-bar method ((2) in Figure 4). We use MSE_i to categorize the samples into two categories: those having their errors greater than MSE_i are counted by the red bar and the remaining are counted by the green bar, i.e., the red bar represents the examples not well fitted by the model. We name this visualization as MSE-contrast visualization.

Overall, the combination of the three types of metrics can help us visually understand the underlying data and the fitness of the model to the data.

2) Observing Distributions: We will use three simple synthesized datasets to show how the visualizations look like. All of the three datasets have 10 feature dimensions and 10,000 samples. The first dataset is generated by randomly fetching feature values from uniform distribution U(0,1) and the target values are generated in the same manner. Therefore, there is no meaningful model in the data. We use a 100 trees x 10 nodes setting to generate the GBT model. Figure 6 shows that in its MSE-contrast visualization the bars are almost distributed uniformly randomly. Most importantly, the MSE visualization shows that MSEs are almost same crossing different nodes and different trees, which indicates no meaningful model. If we see any dataset has a similar visualization like this, we can assert there is no meaningful model with the dataset.

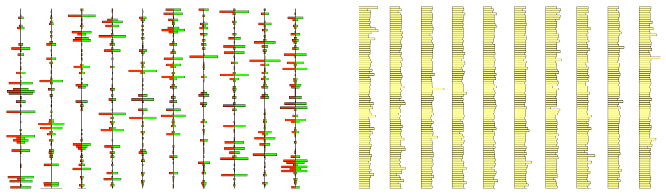


Fig. 6. For the uniform dataset, left: Feature-MSE-contrast visualization; right: MSE-only visualization

The feature values in the second dataset follow a standard Gaussian distribution with mean 0 and variance 1, N(0,1). The target value of a feature vector is defined as the distance of the feature vector to the center $\tilde{\mathbf{0}}$. Similarly, we use the 100 trees x 10 nodes setting to generate the GBT model. Figure 7 shows the Feature-MSE-contrast visualization and the MSE-only visualization. According to the node generation procedure and the node labeling sequence, the data should distributed symmetrically around the middle node in the tree. The result confirms that most samples are in the middle terminal nodes. The MSE visualization also shows some unique pattern that is very different from the pattern from uniform data.

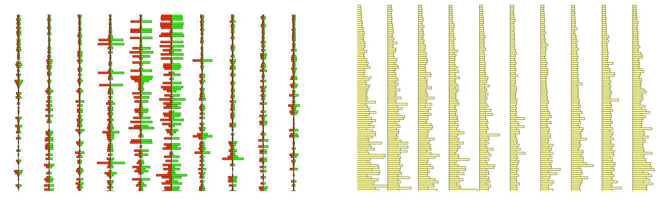


Fig. 7. For the Gaussian dataset, left: Feature-MSE-contrast visualization; right: MSE-only visualization

The third dataset is similar to the Gaussian dataset. However, we tune the feature value distribution to a skewed long-tail distribution (Figure 8). The target value of a feature vector is defined as the distance of the feature vector to the mean $\tilde{\mu}$. We also add in 10% noises, of which the target values are perturbed randomly. We use the same setting to generate the GBT model. Figure 9 shows the Feature-MSE-contrast visualization and the MSE visualization. Now the mass of data has been moved to the first few terminal nodes of the tree, which matches our expectation.

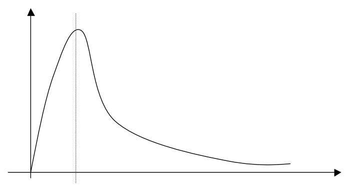


Fig. 8. Skewed long tail distribution

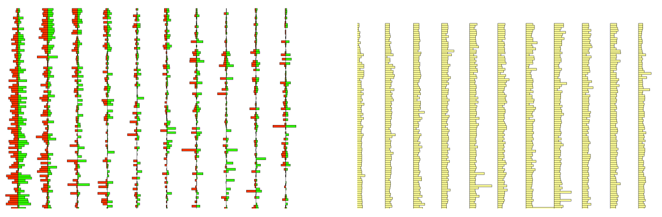


Fig. 9. For the long-tail dataset, left: Feature-MSE-contrast visualization; right: MSE-only visualization

We use these examples to demonstrate that the proposed visualizations can well capture the fitted models – if there is one, and each meaningful model will have a particular "visual signature".

3) Observing Noises.: Note that the Feature-MSE-contrast visualization can also be used to detect noises. Within the subspace (the bounding box) defined by the terminal node, noises are those samples that have their target values significantly deviated from the MSE of that terminal node. If noises exist, the length difference between the red and green bars might be significant. We use a new design called Feature-MSE-difference visualization to observe the possible noise distributions. Let the length of the original red bar and green bar be l_r and l_g, respectively. We define the length of red/green bar, notated as l'_r and l'_g in the visualization as follows.

$$\begin{cases} l'_r = l_r - l_g, l'_g = 0 & \text{if } l_r > l_g \\ l'_r = 0, l'_g = l_g - l_r, & \text{otherwise} \end{cases}$$

Note that the long-tail dataset contains 10% noises, of which the target values are perturbed randomly. Using the Feature-MSE-difference visualization, we can clearly observe that the long-tail dataset contains a significant amount of noises, while the Gaussian dataset almost has no noise (Figure 10).

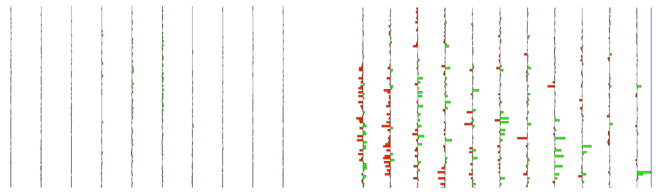


Fig. 10. Feature-MSE-difference, left: the Gaussian dataset has no noise; right: the long-tail dataset has many noises

IV. Analyzing the Similarity of Training Datasets for Domain Adaptation

It is well known if the training data are not sufficiently large, the generated models might be of low quality. In domain adaptation, it is preferred to pool small labeled training datasets from similar domains to create a larger training dataset in the hope of achieving better models [20], [10]. However, it is also risky because one domain's data may not necessarily help the other, and sometimes it may deteriorate the quality of the data. The key of effective model adaptation is to find out domains that (1) similar feature vectors from different domains should give similar labels, (2) for less-overlapped or non-overlapped feature subspaces, one domain's data can complement the other's. In other words, if feature distributions overlap the two domains should agree on labels, otherwise, one set of data should dominate the other. The TNM visualization model can be used to identify both types of training examples.

It is difficult to compare the similarity of two datasets and finding out the dissimilar subspaces in terms of the joint distribution of feature vectors and labels. For a single dimension, it is quite easy to compare a pair of distributions and see the similarity/dissimiarity of label distributions. A normal way is to plot two two-dimensional figures (e.g., x-axis is the investigated feature dimension and y-axis is the label) for the two datasets, respectively, and compare their distributions. However, it would be tedious to do so for hundreds of dimensions and it is also difficult to synthesize the differences from so many dimensions. There are tools for visualizing clustering structures in multidimensional large datasets [8], but data analysis for supervised learning (classification and regression) imposes very different challenges, which requires different visual design.

We design a visualization method for evaluating the dataset similarity based on the TNM visualization and regression modeling. The basic idea can be described as follows. Let's take one of the two domains as the reference domain. Without loss of generality, we use D_s domain for the reference domain, and the other domain is D_t. We train a GBT model with the domain-specific data from D_s. Then, we use this GBT model as the reference model and map the data from the both domains onto the subspaces formed by the terminal nodes. This mapping of records is easily done by following the condition on each internal node. After all records are mapped, the mean target value for each terminal node can be calculated for both domains, respectively. We compare the feature vector distributions and the label distributions based on TNM visualizations. Below we discuss the metric design first and then describe the visual design.

A. Metric Design

We generate a couple of statistics to describe the samples from one domain using the terminal node information. Let p be the number of terminal nodes per tree.

1) The *normalized* number of samples falling onto each terminal node, $n_i, i = 1 \ldots p$. Since the total number of samples may not be the same for a pair of domains, we need to normalize this number. Assume the sample is uniformly drawn from the domain, i.e., with the increase of total population, the samples in each subspace will be increased proportionally. Let the rate r between two sample sets is $r = N_s/N_t$, where N_t and N_s are the number of samples in the domain D_t and D_s, respectively. The number of domain D_t samples, $n_i^{(t)}$, is normalized to $r \times n_i^{(t)}$.
2) The mean square errors (MSE), $e_i, i = 1 \ldots p$. Since a terminal node models a part of the regression function, it is also meaningful to compare the terminal-node-level response and modeling error for both the source and target domains.

For each domain we will get two sets of the above metrics, n_i and e_i, notated by $^{(s)}$ and $^{(t)}$ for the domain D_s and D_t, respectively. According to the definition, we have the following interpretation to the above vectors: the difference of $n_i^{(s)}$ and $n_i^{(t)}$ series can roughly describe the difference between the sample feature distributions, while the difference between $e_i^{(s)}$ and $e_i^{(t)}$ series represents the difference between joint feature-label distributions. Therefore, with the regression tree structure, we can reduce the complicated high dimensional distribution analysis to a comparison between the series of values. By visualizing these series of values and contrasting them between two domains, we can find whether they are similar overall and dissimilar in particular subspaces.

Similar data distributions will surely give similar n_i and e_i series, while the difference observed in the comparison can help us understand the domain disimilarity/similarity. To make the result easier to visually understand and analyze, we use the following visual design.

B. Visual Design and Analysis

First, we design two basic TNM visualizations. The first TNM visualization visualizes the series $n_i^{(s)}$ and $n_i^{(t)}$. On the same terminal node, $n_i^{(s)}$ is represented as the red bar on the left and $n_i^{(t)}$ the green bar on the right. In a similar design, the second TNM visualization is used to visualize $e_i^{(s)}$ and $e_i^{(t)}$.

Linking the two visualizations together, we can get several important indications. The final composite visualization will be designed to capture the following information. First, we look at the sample distributions. (1)If we see well matched sample distributions in the first TNM visualization, we have good chances that the label distributions will be matched as well. (2) If we see some nodes do not have samples from the target domain, this may be caused by several reasons and indicate some opportunities. The missing samples are possibly caused by small sample size or incomplete sampling due to the scale and the complexity of the domain. This pattern indicates a good opportunity that the source domain data can possibly complement the target domain dataset. Domain adaptation [20], [10] may help the target domain – the missing part can be possibly patched by the source domain data with appropriate domain adaptation algorithms.

If the sample distributions are very similar except for some missing parts, we can turn to check the MSE distribution. (1) If we see MSE distributions have no big difference, it is highly possible that the two domains are very similar and the source domain function can often be directly applied to the target domain. (2) If the MSEs of the target domain are much higher than that of source domain data, it is confirmed that the label distribution of the target domain is very different from that of the source domain. The source domain data may not help in modeling. (3) In some rare cases, we can also see the target domain data have lower MSEs on average, which means the reference model fits the target domain even better than the original source domain. This can happen if the source domain labels are noisier than the target domain. In this situation, the two datasets may also have high similarity.

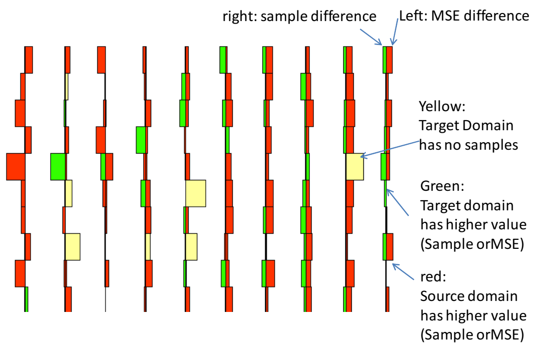


Fig. 11. Composite visualization for comparing both sample distributions and MSE distributions.

We can synthesize the comparisons on sample distributions and MSE distributions into one visualization. Figure 11 shows how such a composite visualization is designed. We use red to represent the source domain and green to represent the target domain. For the same cell, the left bar is used to represent the sample difference – if the source domain has more samples, it is painted red, otherwise green. The right bar is used to represent the MSE difference and the same coloring scheme is applied. If the target domain has no samples on some nodes, then the color of both bars is yellow.

We can also use TNM visualizations to identify special training examples. Let τ and υ be some small values much less than 1. (1) Those terminal nodes satisfying the conditions: $\frac{|m_i^{(s)}-m_i^{(t)}|}{|m_i^{(s)}|} > \tau$ (significant label difference) and $\frac{|n_i^{(s)}-n_i^{(t)}|}{|n_i^{(s)}|} <= \upsilon$ (no significant sample difference) are marked as potential conflicts of interest. In domain adaptation, we may exclude the source domain training examples in these terminal nodes, because they contradict the target domain labels. (2) Those terminal nodes satisfying the conditions: $\frac{|m_i^{(s)}-m_i^{(t)}|}{|m_i^{(s)}|} <= \tau$ (no significant label difference) and $\frac{n_i^{(s)}-n_i^{(t)}}{|n_i^{(s)}|} > 1 - \upsilon$ (dominating source domain samples) indicate that the source domain may contain complementary examples at these nodes. It is possible that adding only these complementary examples to the target domain will be sufficient to improve the model quality.

V. Experiment: Visualizing Domain Difference for Learning to Rank

In last section, we present the basic methods for visually analyzing domain similarity with TNM based visualizations. In this section, we will show how to apply these methods to some real datasets for learning to rank [26]. First of all, we will give a brief description about these datasets. The major task is to understand the similarity between domains for the ranking problem and explore the relationship between domain similarity and the effectiveness of domain adaptation.

A. Datasets

We will use the preprocessed TREC web track data in the publicly available LETOR datasets [26] for experiments. TREC web track datasets are designed to study the retrieval behaviors when the collection to be searched is in a large hyperlinked structure such as the World Wide Web. There are three search tasks in TREC Web track: topic distillation (TD), homepage finding (HP) and named page finding (NP). Topic distillation aims to find a list of entry points for good websites principally devoted to the topic. Homepage finding task is required to return the homepage of the query, and named page finding task aims to return the page whose name is exactly the query. Generally speaking, there is only one answer for homepage finding query or named page finding query, while topic distillation may have multiple answers. TREC evaluators provide two-grade judgments for these tasks, i.e., {relevant, irrelevant}. The recent LETOR datasets (version

3.0) include six TREC web track datasets, i.e., 2003/2004 TD/HP/NP data. Note that the same type of datasets may have different distributions from year to year due to the evolution of the Web. Because they share the same set of features, we are able to use them to simulate six different domains {TD03,HP03,NP03,TD04,HP04,NP04}. These datasets are used in two sets of experiments: (1) for studying the similarity between these domains with the proposed methods: relevance correlation and sample distribution similarity analysis; (2) for verifying the results from the simulated domain similarity.

In the LETOR package, each dataset has been randomly sampled and partitioned to generate five folds for cross validation. All of our results are the average performance over the five folds. We use the normalized version that has all values normalized to the range [0,1], to maximize the possibility of the overlapping on feature vector distribution between the source and the target domain. Readers can find detailed information in the paper [26] or from the web site[1].

Features for Learning to Rank The features for learning to rank are generated in three categories: query-only, document-only, and query-document. We briefly describe these three categories of features. (1) Features that model the user query only do not change over different documents in the document set, such as the number of terms in the query and the frequency of a term in the corpus. (2) Features that model the web document only are constant across all the queries, such as the number of inbound links to the document and PageRank. (3) Features that model the query-document relationship describe the matching between the query and the document, such as the frequency of each query term in the title of the document. The list of features defined in LETOR datasets can be found in LETOR description[26].

B. Similarity Analysis and Effectiveness of Adaptation

We study three pairs of domains, which have been shown with different level of domain similarity in terms of training ranking functions [7]: HP04 and TD04 with high similarity, TD04 and TD03 with very low similarity, and NP04 and HP04 with medium high similarity.

We use the composite visualization (as shown in Figure 11) to compare both the difference of sample distributions and the difference of label distributions. For the pair TD04 and HP04, we use TD04 data to train the reference model (presented in red), with parameters: 10 trees and 10 terminal nodes. Figure 12 shows that HP04 and TD04 have very similar sample distribution (left columns that represent the difference of sample distributions are almost empty) and HP04 data also have lower MSEs on TD04 model (most right columns are red).

Again, we use TD04 as the reference model to compare the pair TD04 and TD03. Figure 13 shows that TD04 and TD03 have very different sample distributions and MSE distributions. Even on some nodes, TD03 has no samples. The visualization confirms that TD04 and TD03 may have low similarity.

[1]http://research.microsoft.com/users/LETOR/

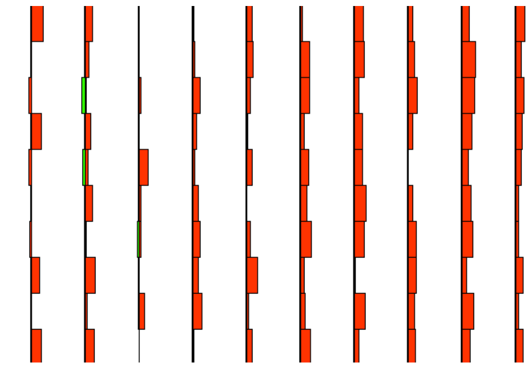


Fig. 12. Comparison on TD04 and HP04.

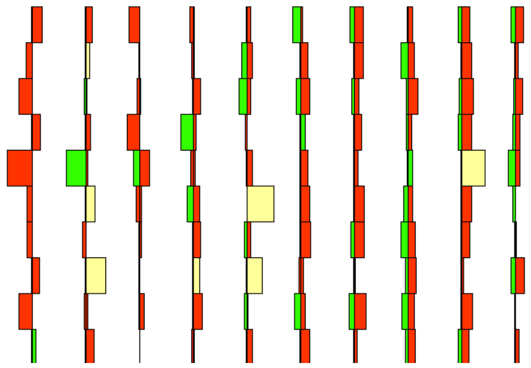


Fig. 13. Comparison on TD04 and TD03.

The composite visualization on NP04 (the reference model) and HP04 shows a very special pattern (Figure 14), where both the sample distributions and the MSE distributions are very similar, i.e., the differences are small as shown in the Figure. We then look at the sample distributions (left subgraph at Figure 15) and MSE distributions (right subgraph Figure 15) separately. It shows that most samples are absorbed by the first node of each tree, while only a few nodes show large MSEs. It seems both datasets have the particular distributions and the reference model fits both datasets very well.

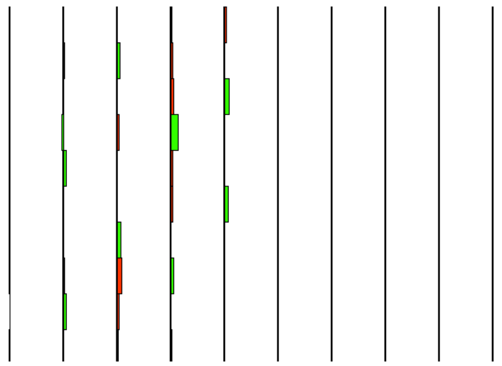


Fig. 14. Comparison on NP04 and HP04.

The experimental study on model adaption has shown that the effectiveness of domain adaptation could be characterized by the domain similarity [7]. We include some of the experi-

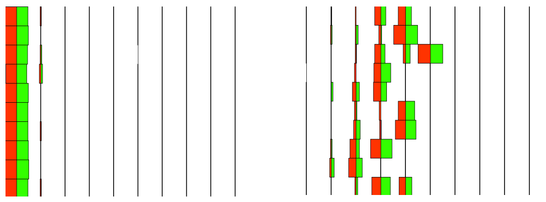


Fig. 15. Comparison on NP04 and HP04 (left: sample distribution, right, MSE distribution).

mental results that are related to the three pairs of datasets to make the comparison more interesting.

We test two adaptation algorithms in the experiment: data combination [20] and Trada [10], and use the source domain function (applying the source domain function directly to the target domain) as the baseline for comparison. In applying the Trada algorithm, we also use the baseline function as the base function in training [10]. All models use the setting of 100 trees and 10 terminal nodes per tree. Figures 16, 17, and 18 show the performance of different adaptation algorithms. For each figure, we also give a brief description in the caption. Due to the space limitation, we will not go to details.

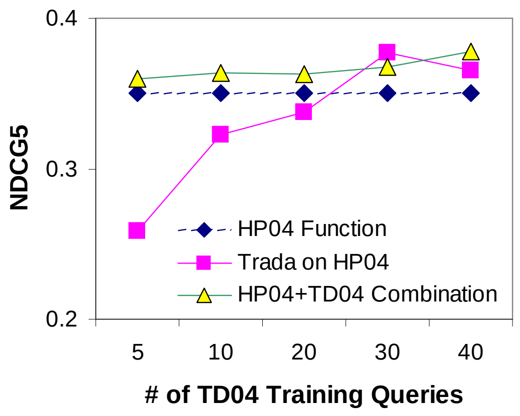


Fig. 16. TD04 and HP04 have high ranking correlation [7]. Using TD04 data can generate good models for both domains. Data combination can slightly help and it is slightly better than Trada (not statistically significant).

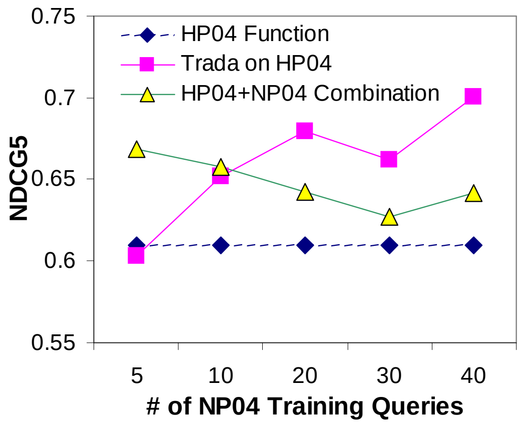


Fig. 17. NP04 and HP04 have medium high correlation. Applying NP04 function directly to HP04 yields moderate performance. With the increase of data size, data combination becomes worse than Trada. Trada can generate statistically significant improvement on larger target training data.

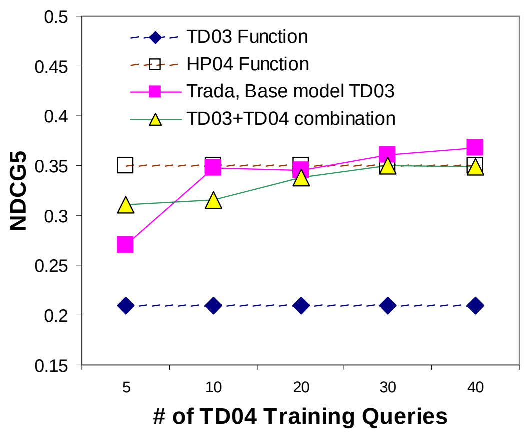


Fig. 18. TD04 and TD02 have low correlation. Domain adaptation helps in this case. However, it will not be better than the function from a highly correlated domain (i.e., HP04 function used for TD04).

VI. Related Work

Visual analytics [30] studies techniques that combine information visualization with statistical analysis, data mining and machine learning. Visually analyzing multidimensional data is one important issue in visual analytics. The major techniques include parallel coordinates [19], star coordinates [22], [8], [29], and Grand Tour [11]. We have studied the interactive visualization techniques for cluster analysis [8], [9]. In this paper, we study the classification problem with the combination of visualization, statistical analysis, and the Gradient Boosting Trees method [14].

Model adaptation has been of great interest in some areas, such as natural language processing [1], [18], [4], [20], speech recognition [25], [12], and learning to rank [10], [7]. In model adaptation, we try to reuse the training samples or models from one domain to enhance the models in another domain. From domain to domain, both the feature distribution and the label distribution change. Thus, understanding how similar between the two domains is critical to the performance of model adaptation. It has been observed that the effectiveness of model adaptation may vary from dataset to dataset [20], [10]. The proposed visual analysis method can be applied to analyze domain similarity as we have shown in the experiment.

Concept drifting is a well known problem with learning from data streams [33], [17], which is also characterized by changing feature/label distributions. There are two types of concept drift: sudden drift or gradual drift. The challenge is distinguishing between true concept drift and noise, since it is often difficult to tell sudden drift from noise. Concept drift caused by the change of underlying data(feature) distribution is also called virtual concept drift, since the real label distribution may not change [33]. Therefore, it is important to monitor such cases and not overact to any change of data. The common approaches to learning drifting concepts are sample selection, sample weighting [24], and ensemble learning [32]. Ensemble learning methods include boosting and bagging [3], [27]. Gradient boosting trees [14] is one type of boosting methods. Understanding the change of concepts and distinguishing real concept drift and noisy drift are two important topics in this area. We will investigate these topics with the proposed visual analysis method as well.

VII. Conclusion

Understanding the similarity between datasets is important for many learning problems, such as domain adaptation and concept drift learning, where the underlying feature and label distributions between datasets may change. However, due to the complexity of multidimensional data distribution, it has been difficult to effectively analyze such changes. We propose a visual analysis method VisGBT based on the gradient-boosting-trees (GBT) learning model. The GBT learning model consists of an array of trees, each of which has fixed number of terminal nodes. We map the important statistics onto the matrix of terminal nodes – Terminal Node Matrix (TNM), so that the multidimensional data distribution and the model fitness information can be visualized in a convenient way. We also showed how to use this technique to perform model fitness analysis and domain similarity analysis. The TNM based visualization can be combined with other point-wise exploration tools, such as VISTA cluster rendering system [8], to conveniently provide more information. Further study will be performed on enhancing the interactive operations, exploring more analytic tasks with the VisGBT method, and providing a toolkit for conveniently integrating this method into real applications.

References

[1] Bacchiani, M., and Roark, B. Unsupervised language model adaptation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2003).

[2] Barlow, T., and Neville, P. Case study: Visualization for decision tree analysis in data mining. *Information Visualization, IEEE Symposium on 0* (2001), 149.

[3] Bauer, E., and Kohavi, R. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning 36*, 1-2 (1999), 105–139.

[4] Blitzer, J., McDonald, R., and Pereira, F. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing* (Sydney, Australia, 2006).

[5] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. Learning to rank using gradient descent. In *Proceedings of International Conference on Machine Learning (ICML)* (2005).

[6] Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning* (New York, NY, USA, 2007), ACM, pp. 129–136.

[7] Chen, K., Bai, J., Reddy, S., and Tseng, B. On domain similarity and effectiveness of adapting to rank. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)* (2009).

[8] Chen, K., and Liu, L. VISTA: Validating and refining clusters via visualization. *Information Visualization 3*, 4 (2004), 257–270.

[9] Chen, K., and Liu, L. iVIBRATE: Interactive visualization based framework for clustering large datasets. *ACM Transactions on Information Systems 24*, 2 (2006).

[10] Chen, K., Lu, R., Wong, C., Sun, G., Heck, L., and Tseng, B. Trada: Tree based ranking function adaptation. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)* (2008).

[11] Cook, D., Buja, A., Cabrera, J., and Hurley, C. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics 23* (1995), 155–172.

[12] Daumé III, H., and Marcu, D. Domain adaptation for statistical classifiers. *Journal of Machine Learning Research* (2006).

[13] Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research 4* (2003), 933–969.

[14] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29*, 5 (2001), 1189–1232.

[15] Hastie, T., Tibshirani, R., and Friedmann, J. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[16] Herbrich, R., Graepel, T., and Obermayer, K. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers* (2000), 115–132.

[17] Hulten, G., Spencer, L., and Domingos, P. Mining time-changing data streams. In *Proceedings of ACM SIGKDD Conference* (2001).

[18] Hwa, R. Supervised grammar induction using training data with limited constituent information. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)* (1999).

[19] Inselberg, A. Multidimensional detective. In *IEEE Symposium on Information Visualization* (1997), pp. 100–107.

[20] Jiang, J., and Zhai, C. Instance weighting for domain adaptation in NLP. In *Conference of the Association for Computational Linguistics (ACL)* (2007).

[21] Joachims, T. Optimizing search engines using clickthrough data. In *Proceedings of ACM SIGKDD Conference* (2002).

[22] Kandogan, E. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In *Proceedings of ACM SIGKDD Conference* (2001), pp. 107–116.

[23] Keim, D. Visual exploration of large data sets. *ACM Communication 44*, 8 (2001), 38–44.

[24] Klinkenberg, R. Learning drifting concepts: Example selection vs. example weighting. *Intellegent Data Analysis 8*, 3 (2004), 281–300.

[25] Leggetter, C., and Woodland, P. Flexible speaker adaptation using maximum likelihood linear regression. In *Proceedings of Eurospeech* (1995).

[26] Liu, T.-Y., Qin, T., Xu, J., Xiong, W., and Li, H. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *Workshop of Learning to Rank for Information Retrieval, in conjunction with SIGIR* (2007).

[27] Schapire, R. E. The boosting approach to machine learning: An overview. *Nonlinear Estimation and Classification"* (2003).

[28] Shneiderman, B. Inventing discovery tools: Combining information visualization with data mining. *Information Visualization 1* (2002), 5–12.

[29] Teoh, S. T., and Ma, K.-L. Starclass: Interactive visual classification using star coordinates. In *Proceedings of SIAM International Conference on Data Mining (SDM)* (2003).

[30] Thomas, J. J., and Cook, K. A. *Illuminating the Path:The Research and Development Agenda for Visual Analytics*. the National Visualization and Analytics Center, 2005.

[31] Tsai, M.-F., Liu, T.-Y., Qin, T., Chen, H.-H., and Ma, W.-Y. Frank: a ranking method with fidelity loss. In *Proceedings of ACM SIGIR Conference* (New York, NY, USA, 2007), ACM, pp. 383–390.

[32] Wang, H., Fan, W., Yu, P. S., and Han, J. Mining concept drifting data streams using ensemble classifiers. In *Proceedings of ACM SIGKDD Conference* (2003).

[33] Widmer, G., and Kubat, M. Effective learning in dynamic environments by explicit context tracking. In *European Conference on Machine Learning* (1993), Springer-Verlag, pp. 227–243.

[34] Xu, J., and Li, H. AdaRank: a boosting algorithm for information retrieval. In *Proceedings of ACM SIGIR Conference* (2007).

[35] ZHENG, Z., CHEN, K., SUN, G., AND ZHA, H. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR* (2007), pp. 287–294.