TECHNICAL REPORT

No. CCLS-08-04

Title: Susceptibility Ranking of Electrical Feeders: A Case Study

Authors: P. Gross, A. Salleb-Aouissi, H. Dutta, A. Boulanger

Susceptibility Ranking of Electrical Feeders: A Case Study

Philip Gross Ansaf Salleb-Aouissi Haimonti Dutta Albert Boulanger

Center for Computational Learning Systems,
Columbia University,
850 Interchurch Center / MC 7717
475 Riverside Drive,
New York, NY 10115
{phil, ansaf, haimonti, aboulanger}@ccls.columbia.edu

Abstract

Ranking problems arise in a wide range of real world applications where an ordering on a set of examples is preferred to a classification model. These applications include collaborative filtering, information retrieval and ranking components of a system by susceptibility to failure. In this paper, we present an ongoing project to rank the feeder cables of a major metropolitan area's electrical grid according to their susceptibility to outages. We describe our framework and the application of machine learning ranking methods, using scores from Support Vector Machines (SVM), RankBoost and Martingale Boosting. Finally, we present our experimental results and the lessons learned from this challenging real-world application.

1. Introduction

Electrical infrastructure has four main components: generation, transmission, primary distribution, and secondary distribution. Between the high-voltage transmission system and the household-voltage secondary system, electricity is sent through *primary distribution feeders*, cables which move energy around a metropolitan area.

In the urban electrical grid under consideration, most of the system is organized into *networks*, where each network is served from a particular substation. Within the network part of the system, a fault anywhere along a feeder will cause the entire feeder to disconnect itself from the grid. Every customer is redundantly connected to the substation so that a single feeder failure will not cause loss of service. However, such a failure increases the stress on the surviving feeders in the network, and subsequent failures will raise the stresses higher still. Eventually, the network may suffer a *cascade failure*, and the entire network must be shut down until the problems can be repaired.

In the metropolitan area considered in this study, there are three regions of interest – A, B and C – which together have over 1000 feeders. Individual feeders fail with some regularity. On average there were over five failures per day over all regions during summer 2007. One of the most important tasks for keeping the electrical grid running is the maintenance and swift repair of primary feeders. For both long-term upgrade and maintenance planning, as well as short-term problem management, engineers and operators would like to know which feeders are at high risk of failure.

Our goal was to create an ordered list of feeders in the system ranked from most susceptible to failure to least susceptible. There are a number of significant challenges. The number of attributes is very large, while failures are relatively rare. The system is believed to exhibit concept drift, where the causes of failures change significantly over the course of the year. Many of the attributes are in the form of time series which must be aggregated. Other attributes map to feeder sub-components (e.g. cable sections, averaging 150 per feeder) or super-components (e.g. substations, which serve many feeders), and must be aggregated or disaggregated to the feeder level. Even constructing a metric to evaluate our ranking performance is not trivial.

This paper has a number of contributions to the area of component susceptibility rankings. First, it is a comparative study of a pairwise ranking algorithm (RankBoost), a classification score-based ranking algorithm (SVM score Ranker) and a sorting-based ranking algorithm (Martingale Boosting, also known as MartiRank). Second, the dataset generation process solves a number of problems inherent in learning to predict events that are rare in space and time.

The rest of the paper is organized as follows: Section 2 describes related work; Section 3 provides a detailed de-

scription of our application, the data generation process and pre-processing steps before machine learning; Section 4 describes the ranking algorithms we use and presents our results. Finally Section 5 presents possible directions for future work.

2. Related Work

The problem of *learning to rank* [1] has received much attention in the machine learning community in recent years.

Pairwise ranking [7] assumes that the learner is provided with information about the relative ranking of individual pairs of instances. Freund et al. [17] proposed a boosting based approach called *RankBoost* that combines many "weak" rankings into a single accurate ranking. A brief review of this algorithm is presented in Section 4.2. Long and Servedio developed another boosting-based method for ranking named *Martingale Boosting* [14] and applied it successfully to some power engineering problems [8]. Joachims [12] used pairwise ranking to design retrieval functions for search engines using Support Vector Machines (SVMs). He utilized the query log of search engines in connection with the log of clicked links to present a ranking and hence determine which documents are more relevant than others.

Cao et al. [4] studied the problem of "listwise" ranking as opposed to the "pairwise" approach described above. They used lists of objects as "instances" in learning and propose a probabilistic model to describe the listwise loss function. In addition to *pure* ranking methods that learn *ranking functions*, some *classification functions* that output scores have also shown good performance in ranking [18]. For instance, Herbrich et al. [9] showed that Support Vector Machine (SVM) classifiers are good rankers as well. Typically, the SVM produces a classifier that labels examples, then thresholds the outputs. Instead, one can rank the examples by how strongly the classifier predicts the class of each example.

Other contributions include a gradient-descent method [3], a naive Bayes approach [19] and new approaches for ordinal regression proposed by Wei and Keerthi [5]. For the feeder susceptibility ranking application described in this paper, we consider three different ranking algorithms: (1) RankBoost, a pairwise ranking algorithm (2) The SVM Score Ranker, a classification score-based ranking algorithm, and (3) Martingale Boosting, a sorting-based ranking algorithm.

In the following section we describe the feeder susceptibility ranking application in more detail.

3. The Feeder Susceptibility Application

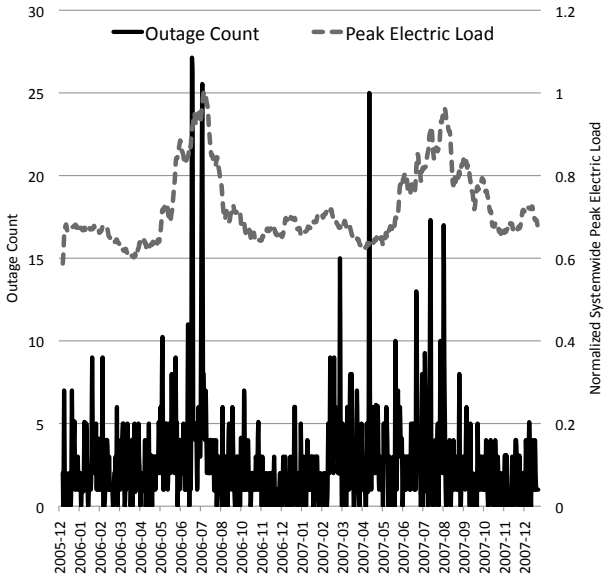


Figure 1. Number of Feeder Outages per Day, 2006-2007, below with axis at left, and Normalized Systemwide Peak System Load above with axis at right

As can be seen in Figure 1, a small number of feeder failures occur daily throughout the year. However, this rate noticeably increases during warm weather. Air conditioning units put substantial load on the system, causing electricity usage to increase by roughly 50% during the summer. It is during these times when the system is most at risk. As operations resources are limited, any information which could help with the optimal allocation of manpower would be of great assistance in keeping the grid running reliably.

A substantial amount of data was made available to us for training and testing, but converting the data into a form suitable for machine learning presents a number of challenges.

3.1. Overall Architecture and Data Sources

For the feeders themselves, the attributes can be characterized as *physical characteristics*, known from the underlying components that compose the feeder cable; *electrical characteristics*, from electric load flow simulations; and dynamic data, from real-time telemetry attached to the feeder. Additionally, we use some *derived attributes*, computed from formulas developed by domain experts.

Our data on the physical composition of feeders is noisy: errors in database entry and rewiring of components from one feeder to another make it impossible to get a perfect snapshot of the current state of the system. However, the main complexity with the physical characteristics is component aggregation: a typical feeder is composed of over a hundred *cable sections*, connected by a similar number of *joints*, and terminating in a few tens of transformers. For a single feeder, these subcomponents are a hodgepodge of types and ages. It is possible for a brand-new cable section to be connected to one that is many decades old.

We have considered a number of approaches to solve this "roll-up" problem, including looking at maxima, averages, 90th percentile (similar to max, but avoids outliers), domain-expert-suggested thresholds, and histogram counts. Feature selection is discussed below in section 3.4.

Electrical characteristics (e.g., how much current a feeder is expected to carry under various network conditions) are necessarily imprecise, as the electric load-flow simulations that generate them must assume the correctness of the given system model, and can only guess at the status of various complex interconnections in the secondary network for which there is little telemetry.

Dynamic data presents a similar problem to physical data, but here the problem is aggregation in time instead of space. Telemetry data is collected at rates varying from hundreds of times per second (for power quality data) to every two minutes (most standard current and voltage measurements) or slower (weather data). We aggregate these over time, again using functions such as max or average, using different time windows. Some of the time windows are relatively simple (e.g. aggregating over 4 or 24 hours, or 7, 15, and 45 days), while others take advantage of the system's periodicity (see Figure 1), and aggregate over the most recent data, plus data from the same time of year in previous year(s).

An additional problem with dynamic data is its sheer volume. We would like to create a dynamic tool, able to provide rerankings every hour, if not faster. Our interest in more elaborate aggregations must be balanced against the time to run them against gigabytes of dynamic data.

Finally, even the labels of the examples present difficulty. Feeders can go offline for a variety of reasons, including being taken offline due to a field worker noticing a problem, failing when being brought back online after repairs, failing a scheduled test, and so forth. We have restricted ourselves to the most serious failure type, where the entire feeder is automatically taken offline by emergency substation relays, due to some sort of fault being detected by sensors.

3.2. Outage Derived Data Sets (ODDS)

Our current system for generating data sets attempts to address the problem of the rarity of positive examples (feeder failures), in space as well as time. An actual feeder failure incident is instantaneous. A snapshot of the system at that moment will have only one failure example. To better balance the data, we tried labeling any example which had experienced a failure over some time window as positive. However, the dynamic data for these examples did not correspond with the conditions at the time they failed. This was a problem, as the domain experts believed that some of the dynamic data might only have predictive value in the period right before the failure.

To solve this problem, we decided to switch to time-shifted positive examples, including examples of all past outages within some time window, along with dynamic data from the moment before they failed. Additionally, we include the current snapshot of all feeders in the system. Not only does this approach (which we call Outage Derived Data Sets, or ODDS) capture the dynamic data from right before the failure, it helps to reduce the massive imbalances between positive and negative examples we see in our data.

A simplified diagram of the overall system is shown below in Figure 2. A number of applications use the data in the Output Data Repository to highlight areas of risk through graphical displays and map overlays.

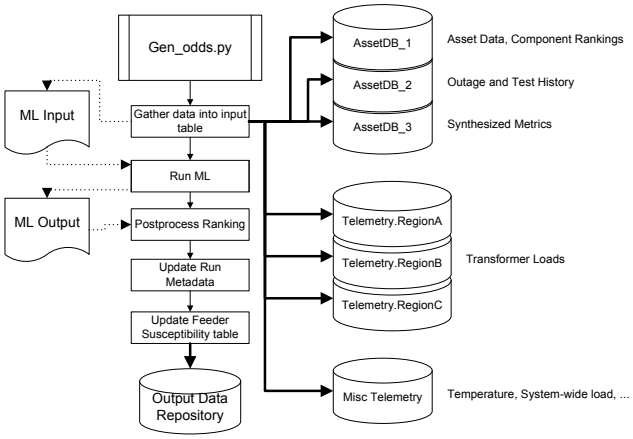


Figure 2. Simplified diagram of ODDS and ML application architecture

3.3. Data Pre-processing

As in many real-life applications, our application suffers from the problem of missing data. The system is being continually upgraded and reconfigured, and new data for various attributes may not be available. Figure 3 shows the extent of missing data on each attribute. Since most well

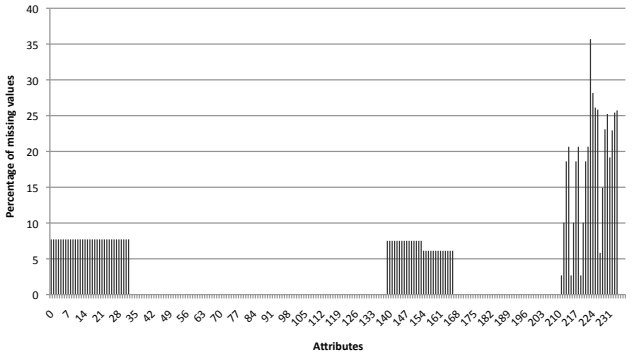


Figure 3. Percentage of missing values per attribute for August 13th, 2007 dataset.

established machine learning algorithms assume fully populated feature vectors, after linearly scaling each attribute to the range [0, +1] to give the same importance to each attribute, we perform a mean imputation to fill in the missing values, by replacing missing values with the mean of the attribute.

3.4. Feature Selection

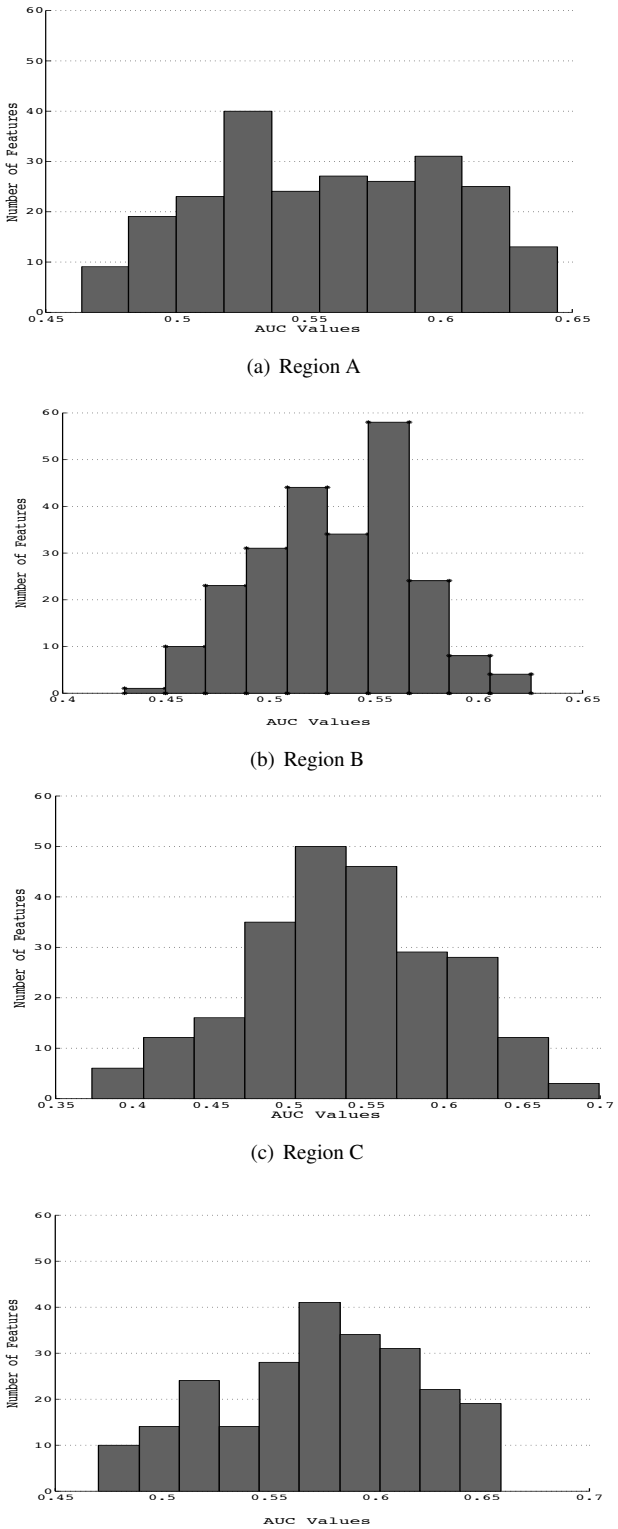
Our datasets contain a relatively large number of static and dynamic attributes (about 237 in all). However, not all the attributes collected are useful for the purpose of feeder susceptibility ranking.

We subjected the attributes collected in our application to a univariate feature selection mechanism. Each feature was used *individually* to rank feeders (using the RankBoost algorithm[1]) according to their susceptibility to failure on training data collected over a 7 day period in summer and tested on data collected over the next 7 days. We recorded the Area Under the Curves (AUCs described in Section 4) for each feature. The AUC values collected over each week in summer were then averaged and sorted to obtain the best features for ranking.

In Figure 4, we present statistics of how well the individual features performed in all regions combined, and separately for each region. The figures suggest that the features that perform well in each region are substantially different. About 48 features have AUC greater than 0.6 in Region A, 5 in Region B, 44 in Region C and 77 if all the regions are taken into consideration at one time. This motivates the need for using different learning models for each region.

Tables 1 and 2 list the top 5 attributes we found in each region, and all regions combined.

[1]For brevity, we do not provide results of feature selection with SVM and Martingale Boosting.



(a) Region A

(b) Region B

(c) Region C

(d) All Regions

Figure 4. Histogram of AUC values of features when used individually for feeder ranking

Region A		Region C	
Feature	AUC	Feature	AUC
$PILC\_Count\_1$	0.644565	$Bad\_Joint\_Count\_1$	0.699328
$Cable\_Length\_1$	0.643745	$PILC\_Count\_4$	0.681529
$Cable\_Length\_2$	0.642091	$Joint\_Rank$	0.667768
$PILC\_Count\_2$	0.641425	$Bad\_Joint\_Count\_2$	0.662215
$PILC\_Count\_3$	0.640964	$Bad\_Joint\_Count\_3$	0.659523

Table 1. Top 5 features found by our feature selection mechanism in Regions A and C, renamed to indicate the type of the attribute. A number of the features relate to counts of known bad subcomponents: Paper-Insulated Lead Cable (PILC), and problematic types of connecting joints

Region B		All	
Feature	AUC	Feature	AUC
$Bad\_Joint\_Count\_4$	0.625402	$Bad\_Joint\_Count\_5$	0.658598
$Bad\_Joint\_Count\_2$	0.617278	$Bad\_Joint\_Count\_2$	0.657514
$PILC\_Count\_5$	0.606627	$Bad\_Joint\_Count\_1$	0.657362
$Electrical\_Importance$	0.604762	$Cable\_Length\_1$	0.655841
$Bad\_Joint\_Count\_5$	0.596526	$PILC\_Count\_1$	0.654492

Table 2. Top 5 features found by our feature selection mechanism in Region B, and all regions.

In the following section we describe in detail the ranking algorithms we use for the feeder ranking application.

4. Ranking Algorithms

We address the problem of *supervised ranking* of data. The term *ranking* refers to the process of taking a collection of data and ordering it in a meaningful and useful order. Supervised ranking outputs such order using the features and guided by the label assigned to each object.

More formally, we would like to order a set of examples

$$(x_1, y_1), \ldots, (x_n, y_n)$$

where $x_1, \ldots, x_n$ are vectors of features describing a set of examples, and each example is given a label $y_i \in \{+1, -1\}$. The sets of positive and negative examples are denoted by $\mathcal{X}^+$ and $\mathcal{X}^-$, respectively.

Ideally, we want to learn a scoring hypothesis h that will rank all positive examples above all negative ones. That is, $\forall x_i \in \mathcal{X}^+$, $\forall x_j \in \mathcal{X}^-$: $h(x_i) > h(x_j)$. Since a perfect ranking is unrealistic in practical applications, we allow mistakes where some negative examples are ranked higher than some positive examples.

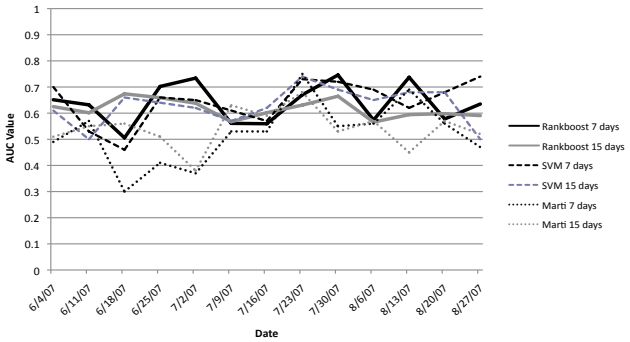
We use Receiver Operating Characteristic (ROC) curves [2] to analyze the ranking results, since they provide a good way of measuring the quality of a ranking when the only ground truth we have is whether or not each data point belongs on the top of the ranking (labeled $+1$) or on the bottom (labeled -1). ROC is essentially normalized by the class cardinality. The quality of an ROC curve is measured by the area under the curve (AUC), which is in the range $[0, 1]$, where an AUC of 0.5 can be achieved by a random ordering of the data, and an AUC of near 1.0 is achieved by perfectly ranking the positive examples at the top and the negative ones at the bottom.

4.1. SVM Ranking

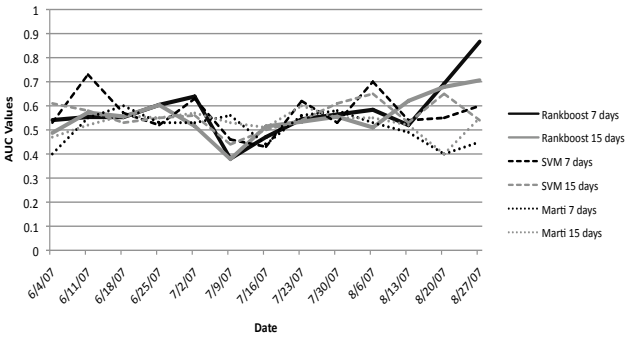
Even though we want to output a ranking, our problem inputs consist of positive and negative examples, not pairwise rankings. Therefore, we use a classification method and convert its output into a ranking. Specifically, we rank objects by sorting the decision values of a linear support vector machine (SVM) ([16]). The SVM is:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}||w||^2 + C \sum_{y_i=-1} \xi_i + RC \sum_{y_j=+1} \xi_j \\ \textit{s. t.} \quad & \forall k,\ y_k[w^T x_k + b] \geq 1 - \xi_k \end{aligned} \tag{1}$$

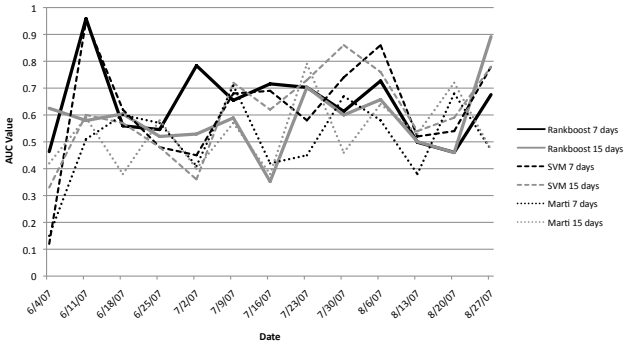
where the ξ variables are slack variables and the C parameter determines the tradeoff between regularization and penalizing misclassification. The R parameter scales the penalty for the positive class. Since we want to penalize mislabeling of an example by the proportion of the total



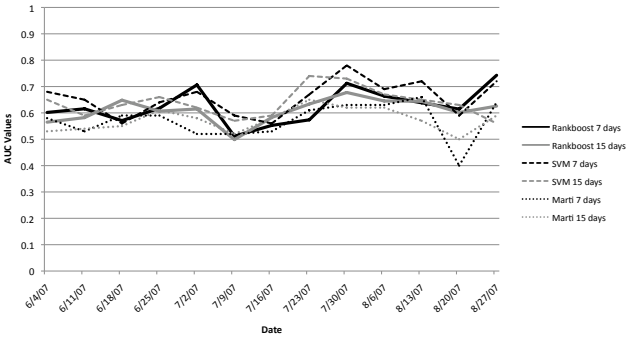
(a) Region A



(b) Region B



(c) Region C



(d) All Regions

Figure 5. AUC values for RankBoost, SVM score Ranker and MartiRank for summer 2007 in each region, as well as combined

population of the class, we can set parameter R by the following:

$$R = \frac{\text{number of true negatives}}{\text{number of true positives}} \tag{2}$$

Typically, the SVM produces a classifier that labels examples x with $y = sign(w^T x + b)$, but we do not threshold our outputs so we can sort and rank our examples by how strongly the linear classifier predicts the class of each example.

To find a good setting of the regularization parameter C, we used a four-fold cross validation on our training set. Here we randomly split the training data into four subsets and use each of these subsets as a validation set for the SVM trained on the remaining three subsets. Trying various values of C, we choose the setting for which the ranking produced by the resulting weight vector scores the highest AUC on the validation sets. Finally, we train on the full training set using the optimal setting of C and use the weight vector w obtained from the SVM to rank the testing set.

4.2. RankBoost

Another algorithm for ranking used in our application is RankBoost [6]. Here, a set of weak rankings are first found that can be "boosted" to obtain a final ranking. For this purpose, we partition all the feeders into only two equivalence sets, those that had an outage and those that did not. We start with a ranked list of feeders that has all the positive examples at the top of the list and the negative examples at the bottom.

Formally, let D_t be the distribution over $\mathcal{X} \times \mathcal{X}$ where $\mathcal{X}$ represents the set of all the positive and negative examples. Suppose that $x_0 \in \mathcal{X}$ and $x_1 \in \mathcal{X}$ is a pair so that we want x_1 to be ranked higher than x_0. At each iteration of RankBoost, the distribution D_t is updated as follows:

$$D_{t+1}(x_0, x_1) = \frac{D_t(x_0, x_1) \exp(\alpha_t(h_t(x_0) - h_t(x_1)))}{Z_t} \tag{3}$$

where Z_t is a normalization factor chosen so that D_{t+1} is a distribution. If the parameter $\alpha_t > 0$, then this equation decreases the weight $D_t(x_0, x_1)$ if h_t gives a correct ranking and increases the weight otherwise. This enables D_t to concentrate on the pairs whose ranking is hardest to determine.

4.3. Martingale Boosting

Martingale Boosting, also known as MartiRank, is a boosting-style ranking algorithm invented by Long and Servedio [14]. MartiRank combines several "weak learners", or simple classification rules, each with low individ-

ual accuracy, into a powerful single model with high predictive accuracy. In our case, the weak learners are functions defined by the ranking produced by sorting on a single attribute. More precisely, MartiRank operates in several rounds and greedily selects in each of its rounds the attribute that is most correlated with the positive examples (in our case outages) in the training data set. The algorithm records the selected attribute along with the direction of its sort (ascending or descending). Thus, in round t, it splits the total data set into t sub-lists, on which it applies its greedy attribute selection procedure; the list is partitioned so that each sub-list contains the same number of positive examples. The trained model is used to obtain a susceptibility ranking of the whole list of examples. This is achieved by repeatedly sorting the list according to the attributes chosen by the trained model.

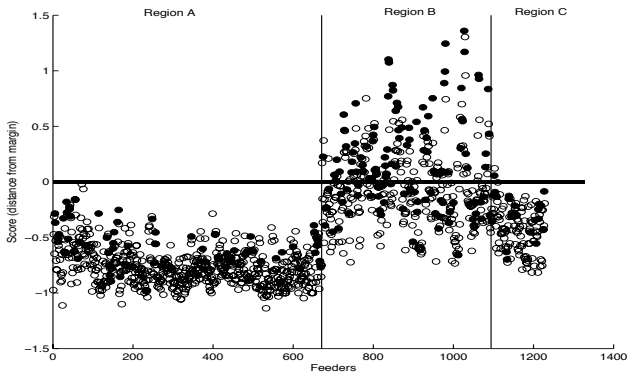
4.4. Ranking Results

The results from feature selection (described in Section 3.4) suggest that the characteristics of data from different regions vary significantly. For example, no one attribute can be voted as the most important attribute in Tables 1 and 2. Similarly, Cable_Length_2 seems to play a significant role in ranking for Region A data, but this does not occur in the top five feature lists for other regions. We hypothesize that the underlying processes leading to feeder failure in each region are significantly different. This is also verified by domain experts, who proposed studying each region independently.

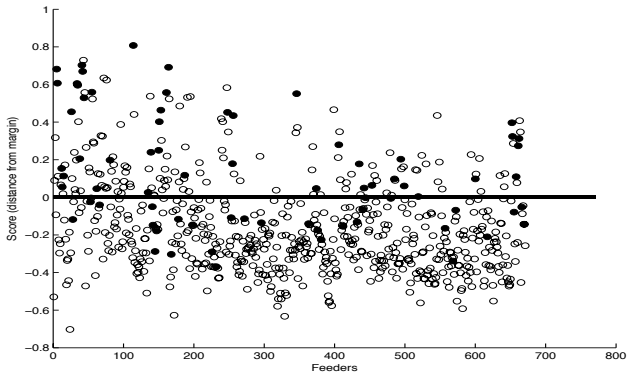
Learning on regions separately was also suggested by our SVM experiments(using SVM-Light [11]). Indeed, as shown in the scatter plot of all the examples in Figure 6(a), one can notice an agglomeration of Region A feeders within a small range of negative score values (representing good feeders). In fact, an SVM model trained for all regions was unable to discriminate good from bad feeders in Region A, leading us to do experiments on each region separately. Figure 6(b) shows Region A feeders along with the SVM margin that discriminates good from bad feeders.

We suspect the length of feeders to be the cause of Region A's particular behavior: This region has much shorter feeders in general, due to higher density of customers. Within this distribution of feeders, length makes a large difference in the likelihood of failure, as extremely short feeders are less likely to fail. This is reflected in the most important attributes chosen by feature selection given in Table 1 where cable lengths appear among the top attributes for Region A. In contrast, even the shortest feeders in regions B and C are quite long in comparison, and thus feeder length has a less pronounced effect.

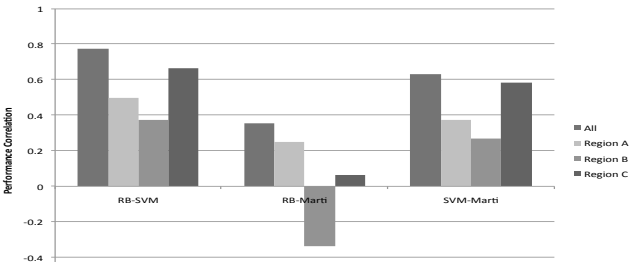
Training Data: We trained our models over a fixed 45-day time window. For example, for the week of August



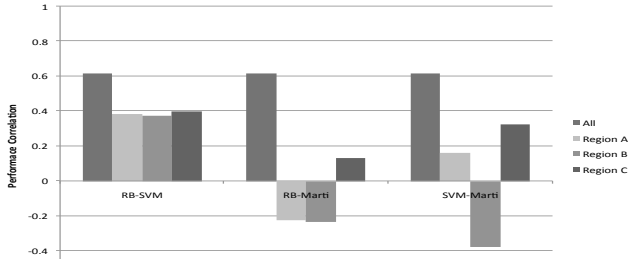
(a) All Regions



(b) Region A, SVM confidence margin

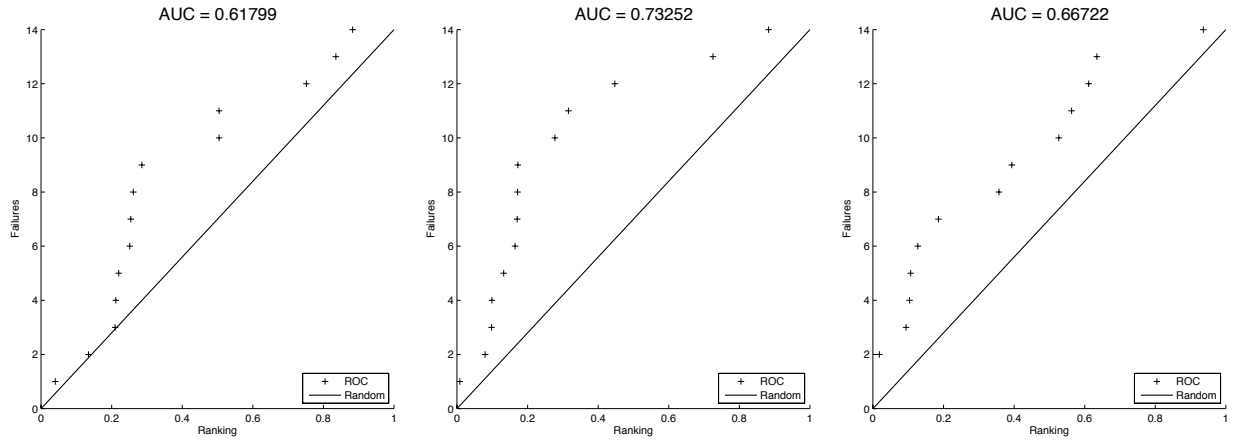


(c) 7-day correlation coefficient



(d) 15-day correlation coefficient

Figure 6. Top graphs: Scatter plot of August 13th data with SVM confidence margin for all regions and just Region A. Feeders are ordered by region and serial id. Filled (resp. empty) circles represent positive (resp. negative) examples. Lower graphs: Correlation coefficients between pairs of learning methods for 7 and 15 days.



(a) RankBoost (b) SVM Score Ranker (c) Martingale Ranking

Figure 7. ROC curves for all regions on 13th August, 2007 using RankBoost, SVM Score Ranker, and Martingale Ranking

Algorithm	Reg. A	Reg. B	Reg. C	All
RankBoost 7d	0.6373	0.5777	0.6431	0.6247
RankBoost 15d	0.6163	0.5563	0.5855	0.6096
SVM 7d	0.6431	0.5700	0.6162	0.6562
SVM 15d	0.6277	0.5608	0.6108	0.6377
MartiRank 7d	0.5215	0.5092	0.5077	0.5715
MartiRank 15d	0.5423	0.5292	0.5331	0.5738

Table 3. Average AUC Values for RankBoost, SVM and MartiRank in all regions over 7 day and 15 day test periods

13th, 2007 our model is trained on data aggregated over fifteen days prior i.e. from 30th July - 13th August, 2007. In addition, we incorporate data from the previous year i.e. from 30th July, 2006 - 27th August, 2006. This is because the performance of the electrical grid is heavily dependent on temperature fluctuations. Assuming weather fluctuations follow similar trends over years, we expect to capture this trend in our ranking models.

Each training data snapshot contains 237 attributes and is an outage-derived data set (ODDS) described in section 3.2; the label is binary: 1 if the feeder had an outage otherwise -1. We built the linear-kernel SVM[2], RankBoost, and Martingale Boosting models on this training data.

Test Data: In order to evaluate the performance of the models, we chose two test periods: 7 days and 15 days from the date under consideration. Thus for the week of August 13th, 2007 our 7-day test data contains all the outages that occur during 13-20 August, 2007, and the 15-day contains outages occurring during 13-27 August, 2007. We are interested in testing whether feeders ranked high on the susceptibility ranking list actually failed.

Figure 7 presents the ROC curves obtained for the Rank boost, SVM and MartiRank models in all regions for the week of 13th August, 2007. The solid line represents an AUC of 0.5 which can be achieved by a random ranking. We tested the performance of the ranking models for June, July, and August, 2007. For each regions, we present 7-day and 15-day test results for RankBoost, SVM Score Ranker, and Martingale Ranking. Figure 5(a), Figure 5(c), Figure 5(b) and Figure 5(d) present results for individual regions and all regions combined. In order to quantify the differences between the three ranking methods, we introduced two metrics: (1) Performance Correlation (2) Average AUC. Performance Correlation is defined as the correlation coefficient between the AUCs obtained from a pair of ranking techniques. Figures 6(c) and 6(d) show the Performance Correlation between RankBoost-SVM, SVM-Marti and RankBoost-Marti for each region when combined together over summer. While Performance Correlation provides a metric of how well the model performed with respect to another, we rely on the average AUC (Table 3) to indicate which one has the best performance.

Our experiments revealed three interesting results:

1. For our application, the pairwise ranking model RankBoost performs as well as an SVM score Ranker. MartiRank appears to perform worse than both RankBoost and SVM score Ranker.

[2] An RBF kernel did not show improved performance.

2. Prediction over 7 days appears to be better than 15 days since AUCs obtained are higher for the former. This is probably due to the time sensitivity of dynamic attributes. As weather and system load changes, the models trained on outdated data become less accurate at prediction.

3. The system seems to exhibit concept drift, and all the algorithms agree on the time periods when such drifts occur. For example, in the week of July 9th, there appears to be a sudden drop in AUC values in almost all regions. We believe this is due to a change in underlying behavior of the electrical grid.

In addition to these results, we observe that our ranking results for Region C are particularly inconsistent among the different ranking techniques we used. We suspect this to be because of the relatively few examples we can use for training.

4.5. On the Detection of Concept Drift

The machine learning results presented so far use a simple batch mechanism. The ODDS dataset uses a current snapshot of all the feeders, plus a time-shifted example for every outage that happens during a fixed time window. We were interested in detecting concept drift using SVMs. Concept drift can be defined as a phenomenon that occurs when the underlying distribution of the data changes over time. So the models need to be able to track such changes, in order to predict future events well.

Joachims presented a window-adjustment algorithm in [13] to detect concept drift in a classification framework. We adapted this to our ranking problem as follows: for each time t, the algorithm trains an SVM using various window sizes, first trying to predict with just the training data of $t-1$, then with $t-1 \cup t-2$, and so forth, extending further and further back in time. The window size minimizing the $\xi\alpha$ classification error estimate (as described in Joachims [13]) is selected. In our case, we select the window size maximizing the AUC on the the predictions of t.

The goal of this experiment was to find out whether an adaptively-sized window would potentially improve results in our framework and possibly detect when concept drift occurred. We considered a set of 35 independent, non-overlapping datasets (called *batches* [13]) of 7 days each, from January through August 2007. The number of outages per week ranged from 10 to 50. We tried to predict each dataset t using up to 6 earlier datasets. The results are reported in Figure 8 for C=0.05. Experiments with other C settings are not reported here as they had very similar results.

Figure 8 compares the best AUC learned using a fixed-size window of 7 days to the adaptive window scenario. Clearly, adaptive window learning exhibits significantly higher AUC than the fixed-size window. Figure 8 also shows the window size chosen by the algorithm. Here we can see some evidence for concept drift. When the optimal window size drops to 1, we deduce that the system has changed, as only new data is useful for prediction. The old data no longer reflects the current status of the system and is not helpful for learning. Concept drift seems to happen several times in the range time of study.

In particular, drift seems to have occurred at the end of March, the end of April and the last week of July. Note that the system recovered quickly after the end of July, once again successfully using data from before the apparent drift. Thus the causes of failures seem to have shifted at that time, and then changed back to their earlier state.

Detecting concept drift in our complex system is challenging and we plan in future work to address this issue using SVMs, RankBoost and MartiRank.

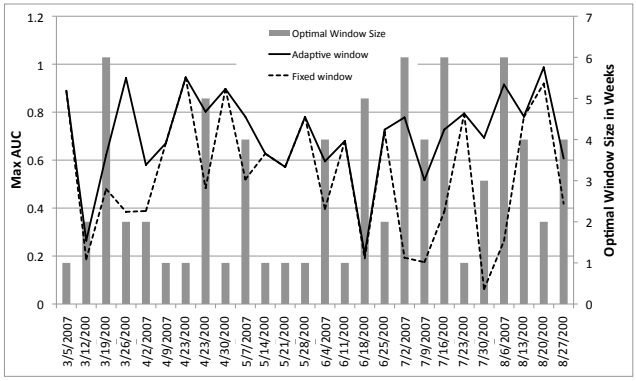


Figure 8. Lines above, with axis at left: Comparison of the maximum AUC obtained with a fixed size window versus an adaptive size window. Bars below with axis at right: Window size selected for each batch.

In the following section we present directions for future work for the feeder susceptibility application.

5. Contributions and Future Work

The main previous work in the field of susceptibility ranking of electrical components is the Ranker for Open-Auto Maintenance Scheduling (ROAMS) [8]. Our approach, using simpler, non-ensemble machine learning methods, differs from the ROAMS approach in several ways. First, we present a comparative study of a pairwise ranking algorithm (RankBoost), a classification score based ranker (SVM score Ranker) and Martingale Boosting. Second, our datasets are generated to address a number of prob-

lems. They address the rarity of positive examples (i.e. feeder failures) described in Section 3.2; the difficulty of capturing the state of dynamic features immediately before outages happen; the integration of rankings of *subcomponents* that make up the feeder such as cables and joints; and the integration of weather-related attributes, which our domain experts believe to be a primary source of concept drift in the electricity grid.

The application of feeder susceptibility ranking is a relatively new one and there are several directions for future work. For instance, in the data collection step, we are constantly experimenting with new attributes which are designated useful by domain experts. These include attributes pertaining to electrical characteristics and dynamic data related to feeders. Furthermore, since feeders are removed or new ones added on a yearly basis, our attributes pertaining to the physical characteristics of feeders also undergo changes.

We also believe that good features are a key component for good performance of machine learning algorithms; hence we are working on using different algorithms for combining subsets of features that produce the best AUC values. One such approach that we are investigating is a Wrapper-based multivariate feature selection scheme [10]. In addition, we hope to study different techniques of imputing missing variables and aggregating time series data.

We have used SVM score Ranker, RankBoost and Martingale Boosting algorithms for the purpose of ranking. Another direction of future work involves developing theoretical results for analyzing the effect of noise on the performance of our models. This is particularly challenging since modeling noise in real world data is a non-trivial problem in itself. While most of our ranking techniques are based on Boosting and SVM based techniques, we are also investigating decision tree based ranking models such as Probability Estimate Trees (PETs) [15] and their variants. Since our preliminary results and domain knowledge indicate that the system may experience concept drift, we hope to integrate into our application algorithms that track and predict such drift.

6. Acknowledgments

Thanks to Maggie Chow, Arthur Kressner, Serena Lee, and many others at Consolidated Edison Company of New York, Charles Lawson at Digimedia, Marta Arias at Universitat Politècnica de Catalunya and David Waltz, Roger Anderson, Cynthia Rudin, Bert Huang, Hatim Diab, Sam Lee and Leon Wu at the Center for Computational Learning Systems, Columbia University, New York. Funding for this work is provided by a contract between Columbia University and the Consolidated Edison Company of New York.

References

[1] S. Agarwal, C. Cortes, and R. Herbrich, editors. *Proceeding of the NIPS 2005 Workshop on Learning to Rank*, December 2005.

[2] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, July 1997.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2005. ACM Press.

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

[5] W. Chu and S. S. Keerthi. Support vector ordinal regression. *Neural Computation*, 2006. in press.

[6] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 170–178, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[7] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *ECML*, pages 145–156, 2003.

[8] P. Gross, A. Boulanger, M. Arias, D. L. Waltz, P. M. Long, C. Lawson, R. Anderson, M. Koenig, M. Mastrocinque, W. Fairechio, J. A. Johnson, S. Lee, F. Doherty, and A. Kressner. Predicting electricity distribution feeder failures using machine learning susceptibility analysis. In *The Eighteenth Conference on Innovative Applications of Artificial Intelligence IAAI-06*, Boston, Massachusetts, 2006.

[9] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

[10] A. E. Isabelle Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

[11] T. Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

[12] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[13] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[14] P. M. Long and R. A. Servedio. Martingale boosting. In *Learning Theory: 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings*, volume 3559 of *Lecture Notes in Artificial Intelligence*, pages 79–94. Springer, 2005.

[15] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Mach. Learn.*, 52(3):199–215, 2003.

[16] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[17] Yoav Freund, Raj Iyer, Robert E. Schapire and Yoram Singer. An efficient boosting algorithm for combining preferences. In J. W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 170–178, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[18] H. Yu. SVM selective sampling for ranking with application to data retrieval. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 354–363, New York, NY, USA, 2005. ACM.

[19] H. Zhang, L. Jiang, and J. Su. Augmenting naive bayes for ranking. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 1020–1027, New York, NY, USA, 2005. ACM Press.