Real-time Ranking of Electrical Feeders using Expert Advice

Hila Becker\textsuperscript{1,2}, Marta Arias\textsuperscript{1}
\textsuperscript{1}Center for Computational Learning Systems
\textsuperscript{2}Computer Science Department
Columbia University

Overview

- The problem
  - The electrical system
  - Available data
- Approach
- Challenges
- Our solution using Online learning
- Experimental results
The Electrical System

1. Generation
2. Transmission
3. Primary Distribution
4. Secondary Distribution

The Problem

- Distribution feeder failures result in automatic feeder shutdown, called “Open Autos” or O/As.
- O/As stress networks, control centers, and field crews.
- O/As are expensive ($ millions annually).
- Proactive replacement is much cheaper and safer than reactive repair.
- How do we know which feeders to fix?
Some facts about feeders and failures
- mostly 0-5 failures per day
- more in the summer
- strong seasonality effects

Feeder data
- **Static** data
  - Compositional/structural
  - Electrical
- **Dynamic** data
  - Outage history (updated daily)
  - Load measurements (updated every 15 minutes)
- Roughly **200** attributes for each feeder
  - New ones are still being added.
Overview

- The problem
  - The electrical system
  - Available data

- **Approach**

- Challenges

- Our solution using Online learning

- Experimental results

Machine Learning Approach

- Leverage Con Edison's domain knowledge and resources

- Learn to rank feeders based on failure susceptibility

- **How?**
  - Assemble data
  - Train ranking model based on past data
  - Re-rank frequently using model on current data
Feeder Ranking Application

- **Goal:** rank feeders according to failure susceptibility
  - High risk placed near the top
- **Integrate different types of data**
- **Interface that reflects the latest state of the system**
  - Update feeder ranking every 15 min.

Application Structure

- Decision Support App
  - Decision Support GUI
  - Action Driver
  - Action Tracker
- SQL Server DB
- ML Models
- Static data
- ML Engine
- Rankings
- Outage data
- Xfrm Stress data
- Feeder Load data
Overview

- The problem
  - The electrical system
  - Available data

- Approach

- Challenges

- Our solution using Online learning

- Experimental results
Simple Solution

- **Supervised batch-learning algorithms**
  - Use past data to train a model
  - Re-rank frequently using this model on current data
- **Use the best performing learning algorithm**
  - How do we measure performance?
  - MartiRank - boosting algorithm by [Long & Servedio, 2005]
    - Use MartiRank for dynamic feeder ranking

Performance Metric

- Normalized **average rank of failed feeders**

\[
1 - \frac{\sum \text{rank}(\text{failure}_i)}{\#\text{failures} \times \#\text{feeders}}
\]
Performance Metric Example

\[
1 - \frac{\sum rank(failure_i)}{\#failures \times \#feeders}
\]

<table>
<thead>
<tr>
<th>ranking outages</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

\[
1 - \frac{2 + 3 + 5}{3 \times 8} = 0.5833
\]

Real-time ranking with MartiRank

- MartiRank is a “batch” learning algorithm
- Deal with changing system by:
  - generating new datasets with latest data
  - Re-training new model, replacing old model
  - Using newest model to generate ranking
- Must implement “training strategies”
Real-time ranking with MartiRank

How to measure performance over time

- Every ~15 minutes, generate new ranking based on current model and latest data
- Whenever there is a failure, look up its rank in the latest ranking before the failure
- After a whole day, compute normalized average rank
Using MartiRank for real-time ranking of feeders

- MartiRank seems to work well, but...
  - User decides when to re-train
  - User decides how much data to use for re-training
  - Performance degrades over time
- Want to make system automatic
  - Do not discard all old models
  - Let the system decide which models to use

Overview

- The problem
  - The electrical system
  - Available data
- Approach
- Challenges
- Our solution using Online learning
- Experimental results
Learning from expert advice

- Consider each model as an expert
- Each expert has associated weight
  - Reward/penalize experts with good/bad predictions
  - Weight is a measure of confidence in expert’s prediction
- Predict using weighted average of top-scoring experts

Advantages
- Fully automatic
- Adaptive
- Can use many types of underlying learning algorithms
- Good performance guarantees from learning theory: performance never too far off from best expert in hindsight

Disadvantages
- Computational cost: need to track many models “in parallel”
Weighted Majority Algorithm
[Littlestone & Warmuth ‘88]

- Introduced for binary classification
  - Experts make predictions in [0,1]
  - Obtain losses in [0,1]

- Pseudocode:
  - Learning rate as main parameter, \( \beta \) in (0,1]
  - There are \( N \) “experts”, initially weight is 1 for all
  - For \( t=1,2,3, \ldots \)
    - Predict using weighted average of experts’ prediction
    - Obtain “true” label; each expert incurs loss \( l_i \)
    - Update experts’ weights using \( w_{i,t+1} = w_{i,t} \cdot \text{pow}(\beta, l_i) \)

Weighted Majority Algorithm

- Calculate \( l_i = \text{loss}(i) \) for \( i=1,\ldots,N \)
- Update: \( w_{i,t+1} = w_{i,t} \cdot \text{pow}(\beta, l_i) \) for learning rate \( \beta \), time \( t \) and \( i=1,\ldots,N \)
In our case, can’t use WM directly
- Use ranking as opposed to binary classification
- More importantly, do not have a fixed set of experts

Dealing with ranking vs. binary classification
- Ranking loss as normalized average rank of failures as seen before, loss in [0,1]
- To combine rankings, use a weighted average of feeders’ ranks
Dealing with a moving set of experts

- Introduce new parameters
  - B: “budget” (max number of models) set to 100
  - p: new models weight percentile in [0,100]
  - α: age penalty in (0,1]

- If too many models (more than B), drop models with poor q-score, where
  - $q_i = w_i \cdot \text{pow}(\alpha, \text{age}_i)$
  - i.e., $\alpha$ is rate of exponential decay

Online Ranking Algorithm
Other parameters

- How often do we train and add new models?
  - Hand-tuned over the course of the summer
  - Alternatively, one could train when observed performance drops.. not used yet
- How much data do we use to train models?
  - Based on observed performance and early experiments
    - 1 week worth of data, and
    - 2 weeks worth of data

Overview

- The problem
  - The electrical system
  - Available data
- Approach
- Challenges
- Our solution using Online learning
- Experimental results
Experimental Comparison

- Compare our approach to
  - Using “batch”-trained models
  - Other online learning methods
- Ranking Perceptron
  - Online version
- Hand Picked Model
  - Tuned by humans with domain knowledge

Performance – Summer 2005
Performance – Winter 2006

Parameter Variation - Budget
Future Work

- Concept drift detection
  - Add new models only when change is detected
- Ensemble diversity control
- Exploit re-occurring contexts