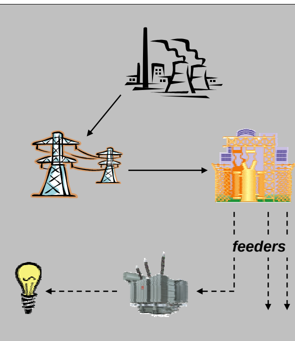
An Online Learning System for the Prediction of Electricity Distribution Feeder Failures

Hila Becker
Columbia University
hila@cs.columbia.edu

Marta Arias
Center for Computational Learning Systems
marta@ccls.columbia.edu

Motivation

- Electrical feeder cables are an essential part of the network that distributes electricity to the boroughs of New York City
- The feeders have a significant failure rate, and many resources are devoted to their maintenance and repair
- We would like to produce a ranking of these feeders according to their failure susceptibility, in order to monitor them and take preventive action
- Since we can gather a lot of data about feeder characteristics and performance, it is natural to use machine learning for this ranking task

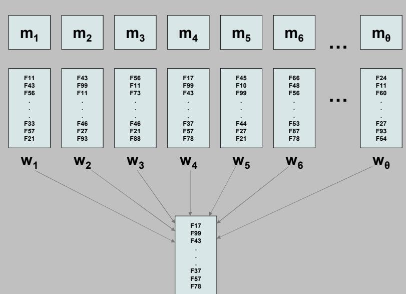


The Problem

- The feature set for each feeder include
  - Static data – age, composition of feeder sections
  - Dynamic data – electrical load on a feeder and its transformers
- Dynamic data values lead to different models, depending on the date and time of training
- Models have to be trained frequently to reflect the current state of the system
- Need to come up with a strategy for training new models that would best adapt to the changing system

Approach

- An Online-Learning system that treats batch-trained models as "experts"



- Build on the notion of learning from expert advice as formulated in the continuous version of the Weighted Majority algorithm
- Each model has a weight, which serves as a measurement of its performance throughout the algorithm
- To predict, we combine the ranking of the top performing models by computing the weighted average rank per feeder re-sorting according to these ranks
- The weights are updated at every round to reflect the performance of the model in the current round with respect to the true labels
- We measure performance as a normalized average rank of failures. For example, in a ranking of 50 items with actual failures ranked #4 and #20, the performance is: 1 - (4 + 20) / (2*50) = 0.76
- We can add new models at every round in order to adapt to the changes in the state of the system
- We also remove poorly performing and old models to avoid having to monitor an ever-increasing set of experts

The Algorithm

Several parameters can be tuned to improve the performance of our algorithm:

- **β** : **Learning Rate** - a constant (0,1] used in the weight update function
- **N** : **Max Number of Models** - number of models which may be considered for use in the expert ensemble
- **M** : **Max Ensemble Size** - the number of experts used to make a prediction
- **α** : **Age Penalty** - rate for exponential decay by age, used for dropping models
- **p** : **New Models Weight Percentile** – determines what weight to assign new models as a percentile in the range [min,max] for the minimum and maximum weights of the existing models
- **n** : **New Models** - the number of models to add in each round

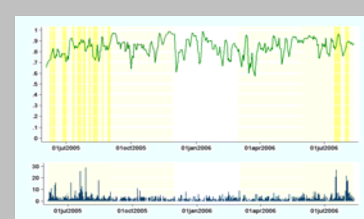
Let **T** be the number of rounds and **θ**=0 the initial number of models

For t=1 to T:

- *Train n new models $m_{\theta+1}, \dots, m_{\theta+n}$; $\theta=\theta+n$*
- *Assign a weight to each new model: $w_{\theta+i}$ = p'th percentile of current weights*
- *Receive new data and for each model m_i , $i=1\dots\theta$ generate ranking r_i*
- *Predict by combining the ranking of the M highest-weight models*
- *Compute the weighted average rank per feeder and sort to produce the algorithm's predicted ranked list*
- *Receive the actual ranking, compute performance score s_i and suffer loss $L_i = (s_{best} - s_i)/(s_{best} - s_{worst})$ for each model m_i*
- *Update the weights: $w_{i,t+1} = w_{i,t} * \beta^{L_i}$*
- *If total number of models θ > N*
  - *Calculate $q_i = w_{i,t+1}$ * α^age for each model*
  - *Drop the (θ-N) models with lowest q value*

Experiments

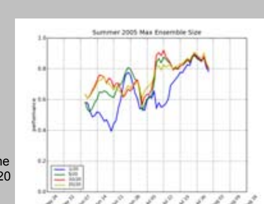
June-August 2005 performance with a weak training strategy
top: performance of SVMs, MartiRank and Linear Regression algorithms measured as the normalized average rank of failures per day, new models trained every two weeks
bottom: number of outages per day

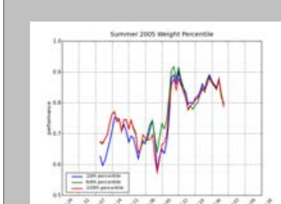


Performance of the online system June 2005-August 2006
top: average rank of failures per day
bottom: number of outages per day

Summer 2005 Variation in performance of the online system by tuning the max ensemble size parameter

Shows the tradeoff between combining the advice of 1,5,10 and 20 experts for the final prediction





Summer 2005 Variation in performance of the online system by tuning the weight percentile parameter

Shows the tradeoff between weight assignment of new experts in the 10th, 60th and 100th percentile

This work has been partly supported by a research contract from Consolidated Edison Company of New York