Leveraging High Performance Computation for Statistical Wind Power Prediction

Cy Chan[*], James Stalker[**], Alan Edelman[*], and Stephen Connors[*]

Massachusetts Institute of Technology[*] and RESPR, Inc.[**]

Overview

This poster presents a new application of a particular machine learning technique for improving wind forecasting. The technique, known as kernel regression, is somewhat similar to fuzzy logic in that both make predictions based on the similarity of the current state to historical training states. Unlike fuzzy logic systems, kernel regression relaxes the requirement for explicit event classifications and instead leverages the training set to implicitly form a multi-dimensional joint density and compute a conditional expectation given the available data.

The need for faster, highly accurate, and cost-effective predictive techniques for wind power forecasting is becoming imperative as wind energy becomes a larger contributor to the energy mix in places throughout the world. In wind forecasting, like in may other scientific domains, it is often important to be able to tune the trade-off between accuracy and computational efficiency. The work presented here represents the first steps toward building a portable, parallel, auto-tunable forecasting program where the user can select a desired level of accuracy, and the program will respond with the fastest machine-specific parallel algorithm that achieves that accuracy target.

Even though tremendous progress has been made in wind forecasting in the recent years, there remains significant work to refine and automate the synthesis of meteorological data for use by wind farm and grid operators, for both planning and operational purposes. This presentation will demonstrate the effectiveness of computationally tunable machine learning techniques for improving wind power prediction, with the goal of finding better ways to deliver accurate forecasts and estimates in a timely fashion.

Kernel Density Estimation

KDE is a non-parametric model that does not assume any particular structure for the target distribution (linear, quadratic, etc). It uses a historical data set to construct a conditional probability density function to make estimates. The density estimate is similar to a histogram. On each data point, we put a probability mass and then sum all the point masses to get the joint density estimate:

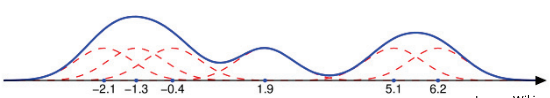


Image: Wikimedia Commons

Why Kernel Density Estimation?

Kernel Density Estimation is our algorithm of choice because it has lots of "knobs" to adjust the power of the algorithm. In one situation, we could turn the knobs to utilize a server farm's worth of computation for multiple hours to yield highly accurate results. In another scenario, we may need to make an estimate in a more timely fashion, so we can adjust the algorithm to sacrifice some accuracy in favor of expediency. As computational hardware becomes more complex, there will no longer be a one-size-fits-all solution. We will need tunable algorithms such as this to make the best use of the hardware at hand.

The Nadaraya-Watson KDE Model

Let *x* represent the vector of predictor variables and *y* the quantity to be estimated (in our case, wind speed). Given a historical training set (x_j, y_j) and kernel function *K* the kernel density estimate at (*x*, *y*) is:

$$f(x,y) = \frac{1}{N}\sum_{j=1}^{N} K(x-x_j)K(y-y_j)$$

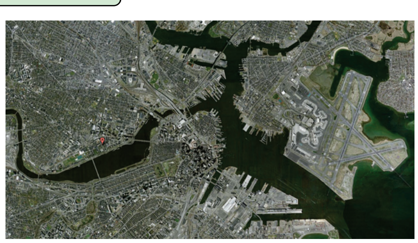
We can then calculate the conditional expectation of *y* given *x*:

$$E[Y \mid X] = \int y \cdot \frac{f(x,y)}{f(x)}\,dy = \frac{\sum_{j=1}^{N} y_j \cdot K(x-x_j)}{\sum_{j=1}^{N} K(x-x_j)}$$

There are several nice things about the form of this expression. First, we can use any sort of variable for the predictors. Wind speed, wind direction, temperature, time of day, day of year can all be predictor variables. These variables can come from multiple sites, including the site where the estimate is being made, neighboring measurement sites, and grid points from a numerical weather forecast such as the NAM 12km model. It is because of this flexibility that we can use this approach on different application types, such as forecasting and site assessment. Also note that during parallelization the algorithm can be broken down into relatively independent pieces to minimize the communications burden on a distributed memory machine.

Forecast Analysis on Wind Speeds at MIT

To evaluate the effectiveness of our methodology, we analyzed a test site on MIT campus. Data was taken from sensors on the top of the Green Building (Building 44) on the east side of campus. There are plans to install a small-scale turbine on campus by the end of 2010. The turbine installation is being planned by MIT Facilities and the MIT Full Breeze student group.

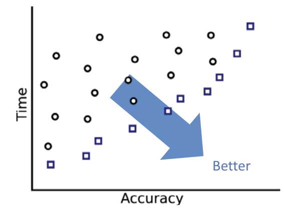


Computational Approach

The recent trend in computing has been towards increasingly parallel machines. This is not just in the space of high performance computing (i.e. supercomputing), but also for everyday machines such as desktops, notebooks, and even embedded devices! Because power consumption increases with the cube of the clock frequency, chip designers are now favoring massive parallelism over faster single core performance. As the number of cores increases, everything around them becomes more complex, especially the memory subsystem.

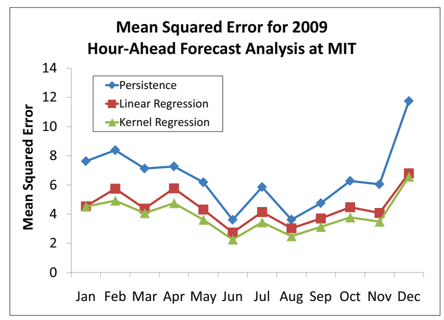
The hardware problem has thus become a software problem. Designing portable, maintainable software that can harness the power of parallel computers is of utmost importance . In order to manage the search for an efficient algorithm, we plan to leverage a new programming language and compiler, called PetaBricks, to search the space of forecast estimation algorithms for the one that will work best given our accuracy requirements and hardware and time constraints.

The figure to the right illustrates a potential set of algorithms where the user can trade off computation time for accuracy of the result. For example, if we need an answer quickly and are willing to sacrifice some accuracy, we would pick an algorithm on the left. If we wanted the highest accuracy and are flexible in the amount of time required, then we would pick an algorithm on the right.



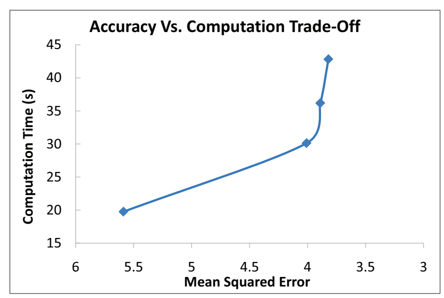
Experiment and Results

For this experiment, we used outputs from the NAM 12km Model to make forecasts at a location on MIT campus (see lower left). We used hourly data for the year 2009 to make one hour ahead predictions and compared the performance of our kernel density estimates against persistence and linear regression.



The kernel regression estimate performed better on average than both of the other techniques. Kernel regression had a MSE 40% lower than persistence and 12.5% percent lower than linear regression.

The second graph shows how tuning the "knobs" of the algorithm allows the user to trade accuracy for faster computation. The more predictor variables used, the higher the accuracy achieved, but at a higher computational cost.



These results were obtained using a relatively small set of predictor variables. We hope to achieve better results with more room for improvement using a better estimation algorithm (discussed below) and more diverse predictor variables.

We also ran a measure-correlate-predict (MCP) analysis on NOAA ocean buoy data. The kernel density estimation achieved an improvement of greater than 25% (in terms of mean squared error versus observed) compared to using the variance-ratio MCP method in estimating missing historical data. These MCP computations were performed on a SGI Altix 350 machine with 12 Intel Itanium 2 processors running Interactive Supercomputing's STAR-P software. Overall, using all 12 processors on the machine, a 8.9x speedup compared to serial performance was achieved.

In the future, we plan to use a modification of the algorithm presented here to minimize the mean squared error plus a regularization term, which helps with generalization. This estimation algorithm is more complex as it involves solving a large symmetric linear system to attain the objective minimization. Moving to this more complex algorithm will provide greater accuracy as well as fertile ground for exploring performance vs. accuracy trade-offs.

Conclusions

We have shown that the use of tunable kernel density estimation and regression techniques can be applied effectively when leveraging high performance parallel computing resources. Not only are the results achieved better than those produced when using methods such as persistence and linear regression, but also the algorithms are tunable to allow the user to trade accuracy for computational performance and vice versa to suit the user's needs.

These types of techniques will become ever more important as parallel computing becomes ubiquitous across all types of computing platforms. As software developers struggle to update their programming practices to utilize these types of resources, techniques such as the automatic tuning of performance parameters to achieve the user's desired results will become extremely valuable. In the future, we plan to implement these techniques with the PetaBricks programming language, which will do automatic algorithm selection and parameter tuning to achieve high performance, portable, parallel, variable accuracy software for wind prediction applications.

This project is funded by an MIT Martin Family Fellowship and a MITEI Seed Fund Grant