Title
Leveraging High Performance Computation for Statistical Wind Prediction

Authors
Chan, CP
Stalker, J
Edelman, A
et al.

Publication Date
2010-05-23

Peer reviewed

Leveraging High Performance Computation for Statistical Wind Prediction

Cy P Chan James R Stalker
Alan Edelman Stephen R Connors

WINDPOWER 2010
American Wind Energy Association
Dallas, Texas, USA May 23–26, 2010

Leveraging High Performance Computation for Statistical Wind Prediction

Cy P Chan[†] James R Stalker[◇] Alan Edelman [†⋆] Stephen R Connors [‡]

[†] *Computer Science and AI Lab (CSAIL), MIT, Cambridge, MA*
[◇] *RESPR, Inc., Las Cruces, NM*
[⋆] *Department of Mathematics, MIT, Cambridge, MA*
[‡] *Laboratory for Energy and the Environment, MIT, Cambridge, MA*

cychan@csail.mit.edu jstalker@respr.com edleman@math.mit.edu connorsr@mit.edu

Abstract

This paper 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 form an implicit multi-dimensional joint density and compute a conditional expectation given any 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. Several approaches that depend on large computing resources are already in use today; however, high performance computing can help us not only solve existing computational problems faster or with larger inputs, but also create and implement new real-time forecasting mechanisms.

In wind power forecasting, like in many 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 the 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.

1. Introduction

High performance computing is about more than just solving existing computational problems faster or with larger inputs. It is about changing the way we approach problems so that we may create new real-time mechanisms for us to use. The need for predictive techniques for wind power forecasting is becoming apparent as wind energy becomes a larger contributor to the energy mix in places throughout the world. 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.

In this project, we explore the application of computationally intensive machine learning techniques to the area of near-term wind speed and power estimation and prediction. We leverage our research group's high performance computing resources to deliver faster estimates than we are able to achieve using a serial prediction program. Our goal is to improve our methods to find better ways to deliver accurate forecasts and estimates in a timely fashion.

The need for predictive techniques for wind power forecasting is becoming apparent as wind energy becomes a larger contributor to the energy mix in places like Denmark, Spain, and Texas (see [1]). Recently, a rapid drop in wind output from wind farms in West Texas nearly caused a statewide blackout. While numerous companies are springing up to fill this forecasting gap, 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.

High speed computing techniques will be particularly beneficial as we look to short-term and real-time prediction of wind output. Forecasting wind several days ahead allows grid operators to better schedule non-wind generation assets, and reduce overall costs. Near real time, hours-to-minute, predictions of changes in wind across areas of the regional power grid informs grid operators about real-time power reserves, reducing uncertainty, increasing reliability, and avoiding the potential need for backup generation.

In this paper, we present an hour-ahead forecast analysis for a site on the MIT campus. Using very basic kernel regression techniques, we attained a 40 percent improvement over persistence and a 12.5 percent improvement over linear regression (using mean squared error versus actual wind speed as a metric). Further, we demonstrate the tunable nature of the kernel regression algorithm, where the user may trade off accuracy of the result for a faster computation time. For example, by allowing the mean squared error of our estimate to increase by 46 percent, one can increase the speed of computation a factor of 2.2x. Depending on the application, the ability to make trade-offs such as these could be critical to the user.

1.1 Outline

Section 2 describes the Nadaraya-Watson kernel regression technique used. Section 3 describes the utilization of high performance computing resources in our estimation framework. Section 4 describes other relevant implementation details. Section 5 describes our experimental setup and results. Section 6 covers related work. Finally, Sections 7 and 8 describe future work and conclusions.

2. Nadaraya-Watson kernel regression

2.1 Motivation

Kernel density estimation constructs 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

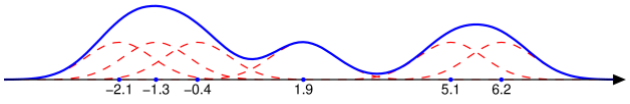

Figure 1. Illustration of a 1-D kernel density estimation. A probability mass in the shape of the selected kernel function (in this case, a Gaussian function) is placed at the coordinate of each data point in the training set. The normalized sum of the probability masses gives an estimate of the underlying probability distribution function. See Section 2 for further details.

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. Figure 1 illustrates this process graphically.

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.

2.2 Technical details

Let the *target variable* Y be the random variable we wish to estimate (typically the wind speed at target location), and let the *observed variables* $X_1, X_2, \ldots, X_k$ (predictor variables) be the known data that are inputs into the estimation computation (typically observed quantities such as wind speed at a relatively earlier time and prediction model outputs).

As an example, we may set Y to be the wind speed at the target location at some point in time t and set (X_1, X_2) equal to the wind direction and speed at the target location at time $t-1$, (X_3, X_4) equal to the wind direction and speed at a neighboring site at time $t-1$, and X_5 equal to the predicted wind speed at some neighboring grid point at time t. Then the random variables $Y, X_1, X_2, X_3, X_4, X_5$ have a joint density based on what values those variables might take at some random time t.

Given N historical training data points $\{x_{i,j}, y_j\}_{i=1\ldots k, j=1\ldots N}$, the kernel density estimate is

$$f(x_1, x_2, \ldots, x_k, y) = \frac{1}{N} \sum_{j=1}^{N} \prod_{i=1}^{k} K(x_i - x_{i,j}) K(y - y_j),$$

where K is our kernel density function [2, pg.182]. In practice, each kernel function can have a different width as appropriate to the data (these are parameters that must be tuned). Our kernel regression estimate of Y given $X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k$ is then:

$$E[Y \mid X = x] = \int y \cdot f(y|x)\, dy = \int y \cdot \frac{f(x,y)}{f(x)}\, dy,$$

where $X = (X_1, X_2, \ldots, X_k)$ and $x = (x_1, x_2, \ldots, x_k)$. Thus

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

$$= \frac{\sum_{j=1}^{N} y_j \cdot \prod_{i=1}^{k} K(x_i - x_{i,j})}{\sum_{j=1}^{N} \prod_{i=1}^{k} K(x_i - x_{i,j})}.$$

We used a Gaussian kernel because it is easy to compute and exhibits good spatial locality (nearby points are weighed much more heavily than far away points) while still allowing far away points to influence our estimate when no nearby points are available.

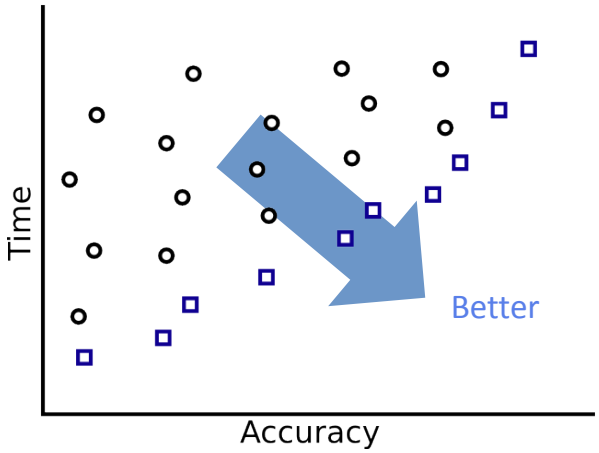

Figure 2. Various algorithms to solve a problem can be plotted according to their accuracy and their performance. Those that lie on the lower right boundary of the figure form a set of Pareto-optimal algorithms. We may then choose an algorithm from this optimal set depending on the accuracy and speed requirements of the user.

In the limiting case where our kernel density estimate converges to the true joint density, this estimator for Y is optimal in the sense that we minimize the expected squared error (conditioned on the state of the predictor variables). Of course, our estimated joint density is not the true density, so there will be some estimation error whose magnitude decreases as the size of our training set increases.

There are several nice things about the form of this expression. Note that 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 for different application types; not only for wind and power forecasting, but also for measure-correlate-predict analyses for site selection.

Further, during parallelization the algorithm can be broken down into relatively independent pieces to minimize the communications burden on a distributed memory machine.

3. High performance 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.

Figure 2 illustrates a potential set of algorithms plotted by their accuracy and computation time. The algorithms that lie along the lower-right border (denoted by blue squares), represent a Pareto-efficient frontier of algorithms. Each algorithm on the frontier represents the best accuracy one can achieve for a given amount of compute time (among the algorithms plotted).

By trading one algorithm on the frontier for another, the user can trade computation time for accuracy. For example, if we need

an answer quickly and are willing to sacrifice some accuracy, we would pick an algorithm on the left side of the frontier. If we want a very high accuracy and are flexible in the amount of time to get a result, then we can pick an algorithm on the right side of the frontier.

For the experiments presented in this paper, we utilized two high performance computing resources. The first was Interactive Supercomputing's Star-P®(now part of Microsoft) running on an SGI Altix 350, a 12 processor shared-memory small-scale supercomputer. The shared memory NUMA (non-uniform memory access) architecture utilized in the Altix 350 is in many ways a predecessor to today's multi-core, multi-socket architectures such as that used in the Intel Nehalem and AMD Opteron microprocessors.

The second resource we used was the MATLAB®Parallel Computing Toolbox running on a cluster of dual-processor nodes each equipped with two single-core 2.2 GHz Intel Xeons. Both parallel computing resources were invaluable to the parameter identification process, providing faster regression and search for the optimal kernel widths and predictor variable weights.

In order to better manage and automate the search for the family of Pareto-efficient algorithms, in the future we plan to leverage a new programming language and compiler, called PetaBricks [3][4], to search the space of forecast estimation algorithms for the one that will work best given our accuracy requirements and hardware and time constraints.

4. Implementation Details

This section describes some of the details of the implementation, including some of the difficulties encountered and how we addressed them.

4.1 Geometry of density estimate

In the case of wind speed, the range of values is simply the positive real line. For wind direction, the range is circular between [0, 360). To deal with the circular nature of wind direction, we wrap the kernel function around the circle so that relative differences between directions are simply taken to be between -180 and 180.

A renormalization is normally needed to make the kernel function a proper probability density function; however, since we are only interested in relative weighting of samples, this normalization can be ignored. Conceptually, the joint density of wind speed and direction is embedded in a multidimensional space where speed dimensions are half lines and direction dimensions are circular. Whether this or another geometry is the best approach is open for exploration.

4.2 Utilization of historical data

Since the quality of a kernel regression relies on a large quantity of historical data to estimate the joint density, we want to use as many data points as possible. Unfortunately, many historical data points may be incomplete. For example, suppose we are using wind and direction data from a neighboring sites in a six hour window; if any one of those data is missing then we cannot apply the standard kernel regression formula.

The way we overcame this difficulty to leverage incomplete data points was to make another estimate as to what effect the missing value *would* have had in the kernel regression formula. We replaced $K(x_i - x_{i,j})$ with $E[K(x_i - X_i)]$ whenever $x_{i,j}$ was unavailable. This simple heuristic does not take advantage of any of the other data available around the missing point; however, computing an estimate based on nearby data is somewhat circular since this is what we are doing in the first place.

5. Experiments and results

The following subsections describe the experimental setup, our statistical results, the computational performance aspects, and a scalability analysis showing algorithm performance as we increase the number of processors working on the problem.

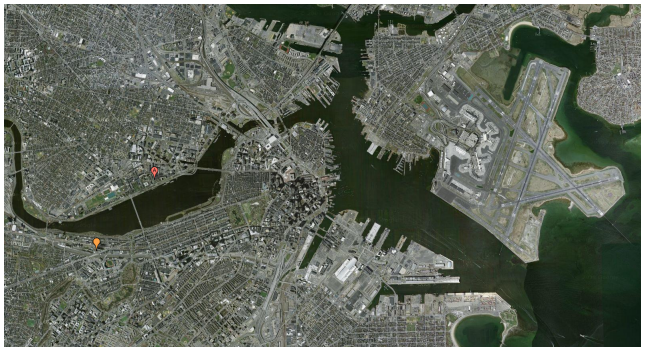

Figure 3. Satellite image of eastern Cambridge, Boston, and surrounding areas. The location of MIT's Green Building is given by the red marker **A**. The NAM grid point used for reference during regression is given by the orange marker to the south-west, across the Charles River. Logan Airport can be seen on the right side of the map.

5.1 Experiment

To evaluate the effectiveness of our methodology, we analyzed a test site on the MIT campus, an urban environment. For the *target site* for estimation, data was taken from sensors on the top of the MIT Green Building (Building 44), a 70 meter tall building on the east side of campus (42.360349 N, 71.089289 W). For the *reference site*, we chose to use the North American Mesoscale (NAM) 12km model output at the grid point closest to the target site (42.34874 N, 71.10034 W). See Figure 3 for a map of both locations relative to downtown Boston and Logan airport.

In this study, we chose to evaluate our methodology using a small number of predictor variables to keep things fast and simple. We synthesized an hourly wind speed and direction data set by interpolating available NAM model output at the reference site and observed data collected at the target site. We then forecasted wind speed one hour ahead at the target site using observed wind speed in the preceding hours at the target site and modeled wind speed and direction at the reference site (the NAM grid point) in both the past and future. We applied this methodology to data over the entire year of 2009 to make predictions and compared the performance of our kernel density regression estimates against persistence and linear regression.

Although we are presenting a simplified version of the algorithm, the algorithm itself is scalable to use more predictor variables (such as humidity, temperature, season, etc.) as well as data from additional sites (such as nearby observation towers or airports), increasing both the accuracy and the computational complexity of the estimation process. We plan to investigate the improvements in accuracy that result from this.

5.2 Statistical results

Figure 4 shows the average mean squared error of each of the three evaluated techniques over the year 2009. Our kernel density estimator performed better than both other techniques, outperforming persistence by roughly 40 percent on average and linear regression by 12.5 percent on average over the course of the year.

Figure 5 shows the performance of the three prediction techniques by month over the course of the year. Even though the magnitude of the errors fluctuated significantly throughly the year, the order of the three algorithms remained the same. We speculate that the higher mean wind speeds during the winter contributed to relatively larger prediction errors. Throughout the

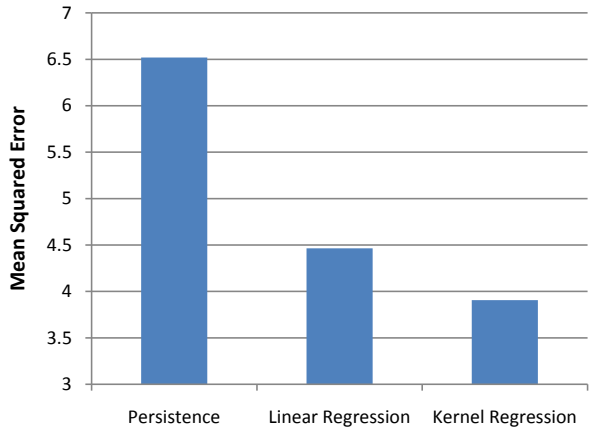

Figure 4. Comparison of the three evaluated estimation algorithms. The vertical axis shows the mean squared error of the estimate averaged over the entire 2009 hourly data set. The kernel regression estimator performed 40 percent better than persistence and 12.5 percent better than linear regression.

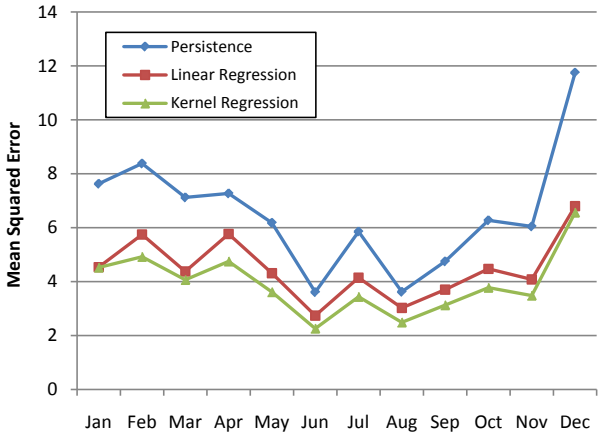

Figure 5. Monthly performance of each of the three evaluated estimation algorithms over the year 2009. Even though the magnitude of the errors fluctuated significantly throughly the year, the order of the three algorithms remained the same.

year, the kernel regression algorithm managed to outperform both techniques, especially persistence.

5.3 Computational performance

Figure 6 shows how tuning the "knobs" of the algorithm can allow us to trade off accuracy for faster computation time. The vertical axis shows the combined computation time to evaluate the kernel regression estimate for every hour of the year 2009, and the horizontal axis shows the mean squared error of the resulting estimates. The more predictor variables we used, the higher the accuracy we were able to achieve, but at a higher computational cost.

In Figure 6, the only knob we are turning is the number of hours of data before the hour of prediction we use to make our estimate. Since this is a fairly simplistic version of the kernel regression algorithm, we seemed to hit a wall for reducing mean squared error. After including the past couple of hours, there starts to be less and less value for including more past data, so the accuracy does not improve much as we increase the number of predictor variables.

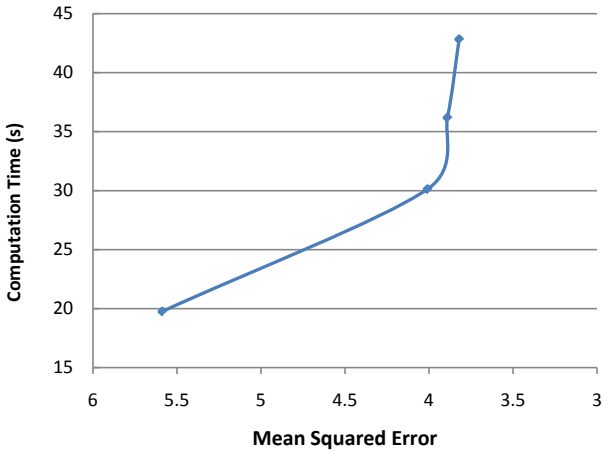

Figure 6. Accuracy vs. performance trade-off in the kernel regression algorithm. As we increase the number of predictor variables used during estimation, both the accuracy level and the computation time increase. Depending on the circumstances, the user may wish to trade accuracy for speed (or vice versa), and choose a different algorithm along the line of Pareto-optimal choices.

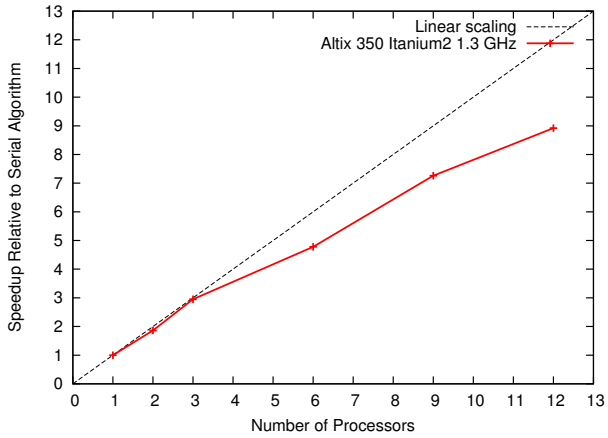

Figure 7. Speedup curve showing parallel performance compared to the performance of a reference serial algorithm

We expect that using different classes of predictor variables, such as temperature, humidity, and also data from other geographical sites or other model outputs would be more effective in giving us better estimates. In that case, we would see a more continual improvement in the accuracy of our algorithm as we increase the amount of computation time available.

5.4 Scalability analysis

The kernel regression algorithm we explored parallelizes fairly easily, essentially breaking down into a map operation followed by a reduce operation. Once the algorithmic components were written and tested serially in MATLAB, we were able to leverage STAR-P's task-based parallelism support in conjunction with optimized MEX files, implemented in C for increased performance.

The resulting parallelized code is several times faster than optimized serial MATLAB / C code. A speedup curve of our algorithm is shown in Figure 7, showing the performance of our parallelized code compared to the use of a single processor. Please note that the data shown here was collected from our algorithm doing a measure-correlate-predict analysis rather than a

forecast analysis; however, since the same algorithm is used in both situations, the parallel scalability should be very similar. Overall, using all 12 processors on the machine, we were able to achieve a 8.9x speedup compared to serial performance.

6. Related work

Existing methods for localizing weather forecast outputs from large weather simulation models (such as the Global Forecast System and the Weather Research and Forecasting Model) to turbine locations include Model Output Statistics (MOS) techniques. MOS techniques typically rely on multiple linear regression to transform predictions at grid output points from the numerical simulation models to predictions at the desired target site [5]. Nonlinear behavior is modeled by using nonlinear functions of source data as predictors.

Other research has also attempted to address this shortcoming. A fuzzy logic system was built and evaluated in [6] to predict wind speeds in Greece. In their system, rules were explicitly created, classifying wind speeds as low, medium, and high at different locations, and devising separate prediction functions for each classification. The need for the explicit determination of nonlinear predictor functions in MOS and the fuzzy rule sets in [6] effectively shifts the burden of predictive modeling from the computer to the person writing the functions or rules. Since there is no guarantee that the person will be correct, we tried an approach that automatically infers nonlinear behavior from the training data set.

As mentioned in Section 2, kernel regression is not limited to forecasting applications, but may also find use in the arena of site evaluation and wind resource assessment. In particular, it may be useful for doing estimations in the style of measure-correlate-predict (MCP) algorithms. In [7] the variance-ratio method was identified as a high quality regression algorithm for doing MCP analysis. In [8] this technique was used to fill in gaps in historical wind speed data from NOAA buoys to analyze potential emissions offsets and revenue for offshore sites in the Northeast United States. In our preliminary studies, we found that the our kernel regression algorithm was able to fill in missing historical data with a mean squared error 25 percent better than the variance ratio method.

7. Future work

We have started to explore a modification of this algorithm using a form of kernel regression which solves for regression coefficients to minimize the least squares error over the training set plus a regularization term, which helps with generalization (the behavior of the system when applied to data not used in the training set). This estimation algorithm is more complex, involving solving a large symmetric linear system to solve for the exact objective minimization.

We plan to explore ways of solving this linear system by trading accuracy of the objective minimization for performance using auto-tuning techniques, such as those provided by the PetaBricks programming language [3] [4]. Because there are parallel components of the algorithm that are relatively data-independent and components that require a tighter data-sharing connection between nodes, the tuning of the parallelization strategy could be a fertile area for improvement.

Although our current methodology has shown measurable improvements over certain existing methods, our goal is to achieve results competitive to much more complex state-of-the-art prediction methodologies currently in use in industry. Such methodologies typically use higher-resolution physics and dynamics-based numerical weather prediction (NWP) models *in combination* with statistical methods like the ones we are currently exploring.

The use of such high-resolution model output is critical to the accuracy of the produced forecasts since they provide a targeted, high resolution version of the weather forecast. For our next set of experiments, we plan on utilizing a high-resolution model and feeding the results into our statistical framework in a manner similar to the NAM model output. We could then produce localized wind speed and power forecasts for specific turbine locations and analyze the benefits of upgrading the model's resolution.

8. 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.

Acknowledgements

We would like to thank the MIT Energy Initiative Seed Fund and the Martin Family Society of Fellows for Sustainability for their generous support of this research.

References

[1] Peter Fairley. Scheduling wind power: Better wind forecasts could prevent blackouts and reduce pollution. *MIT Technology Review*, April 2008.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[3] Jason Ansel, Cy Chan, Yee Lok Wong, Marek Olszewski, Qin Zhao, Alan Edelman, and Saman Amarasinghe. Petabricks: A language and compiler for algorithmic choice. In *PLDI '09: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2009.

[4] Cy Chan, Jason Ansel, Yee Lok Wong, Saman Amarasinghe, and Alan Edelman. Autotuning multigrid with petabricks. In *SC '09: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2009.

[5] J. Paul Dallavalle, Mary C. Erickson, and Joseph C. Maloney III. Model output statistics (MOS) guidance for short-range projections. *20th Conference on Weather Analysis and Forecasting/16th Conference on Numerical Weather Prediction*, 2004.

[6] Ioannis G. Damousis, Minas C. Alexiadis, John B. Theocharis, and Petros S. Dokopoulos. A fuzzy model for wind speed prediction and power generation in wind parks using spatial correlation. *IEEE Transactions on Energy Conversion*, 19(2):352–361, June 2004.

[7] Anthony L. Rogers, J. W. Rogers, and J. F. Manwell. Review of measure-correlate-predict algorithms and comparison of the performance of four approaches. *Journal of Wind Engineering and Industrial Aerodynamics*, 93(3):243–264, March 2005.

[8] Michael Berlinski and Stephen Connors. Economic and environmental performance of potential Northeast offshore wind energy resources. *MIT Laboratory for Energy and the Environment Publication No. LFEE 2006-02 Report*, January 2006.