CS 229 Autumn 2018: AI is the New Electricity

Kevin Hu, Shagun Goel

{huke, gshagun}@stanford.edu

Stanford
CS Department

Introduction

As the world evolves in the direction of renewable energy sources, one key limiting factor is their dependency on climate conditions. Our goal is to use past weather and energy production data to determine the efficiency of implementing solar panels and wind farms at given locations. In particular, given a target location, our model estimates and compares the energy production of photovoltaic cells and wind turbines per unit area and determines the form of renewable energy that maximizes the output. .

Data Collection

- Weather data: Synoptic APIs. Input: timespan and coordinates. Output: Weather data as JSON.
- Solar data: National Renewable Energy Laboratory. Labelled solar output for particular locations over an year.
- Wind data: US Wind Turbine Database. Labelled wind output for particular locations over an year.

Feature Selection

- Raw input features: wind speed, air temperature, relative humidity, and pressure from the weather data.
- SVM models: Gaussian kernel since weather features potentially interact with each other in a multitude of ways. Thus infinite dimensional projection could reveal valuable insights.

Models

For our project, we implemented, for each of solar and wind energy, a linear regression model (baseline), an SVM model, a neural network, and a generalized additive model (GAM). See the comparisons in the rightmost column.

Future Work

- Maximum power output does not always imply the optimal location. Our next step is to incorporate topographical, demographic and economic data into the learning process.
- To improve performance, we could continue to adjust our architecture for FCNN and GAM (e.g. more hidden layers for FCNN or smoother functions for GAM).

References

- Synoptic APIs, *"Mesonet API"*
- NREL, *"Solar Power Data for Integration Studies"*
- USWTDB, *"Geospatial Web Services"*
- Pablo Oberhauser, *"pyGAM : Getting Started with Generalized Additive Models in Python"*

Results

The errors were measured using the root mean squared error (RMSE) metric. Both for solar and wind energy, the GAM model demonstrated the least error.

Solar Energy Results

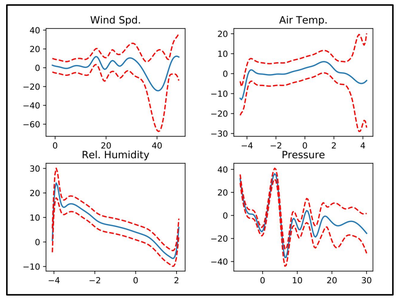
Model	Training error (RMSE)	Test error (RMSE)	Training set size	Test set size
Linear regression	0.129	0.395	940000	11081
SVM	0.218	0.385	940000	11081
Neural Network	15.281	15.248	940000	11081
GAM	0.121	0.369	940000	11081

Wind Energy Results

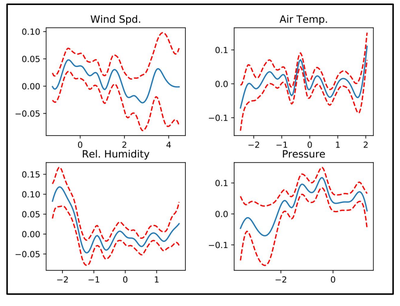
Model	Training error (RMSE)	Test error (RMSE)	Training set size	Test set size
Linear regression	0.11	0.163	25000	2000
SVM	0.719	1.121	25000	2000
Neural Network	20.473	20.361	25000	2000
GAM	0.047	0.072	25000	2000

Discussion

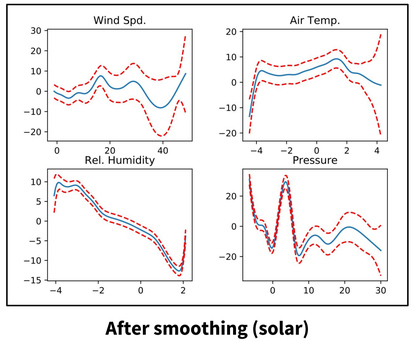
- Looking at the RMSE errors in the previous tables, it is clear that the GAM models are best suited for energy output predictions.
- The performance of neural networks was unexpectedly bad, while linear regression managed to far surpass its original position as a simple baseline.
- In particular, the neural network updates slowed down to a snail's pace after 500 epochs.
- On the other hand, GAMs are extremely interesting in that their additive nature allows us to explore and interpret individual features by holding others at their mean. The graphs to the right show the original relationship between each individual feature and the energy output.



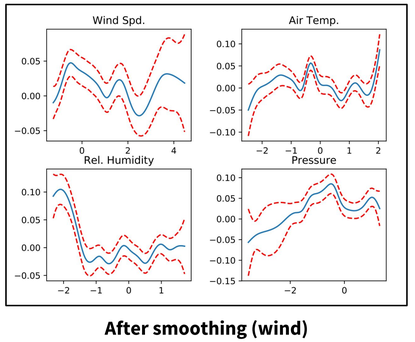
Original relationship (solar)



Original relationship (wind)



After smoothing (solar)



After smoothing (wind)

- Partial dependency plots are really useful since we can infer the dependence of the output on isolated features easily.
- For example, from the diagrams for example, it is clear that an increase in relative humidity leads to a decrease in solar output.
- The jagged nature of some of these plots leads to the conclusion that we should smooth over some of these functions.
- The graphs to the left show the relationship between input features and the output after we smoothed some of the functions.

Linear Regression (Baseline)

For our baselines, we used linear regression with batch gradient descent with batch sizes of 1000 and 100 for solar and wind energy, respectively. Optimal learning rate in either case was 5e-10. The mini-batch gradient descent update rule was:

$$\theta_j := \theta_j + \frac{\alpha}{m}\sum_{i=1}^{m}(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

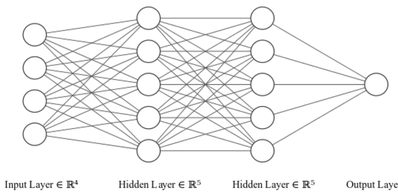
Support Vector Machines

We implemented an SVM using the Gaussian Kernel and L-1 regularization. For solar energy, optimal results were found for C = 1 and epsilon = 0.1 for all features. For wind energy, optimal results were found for C = 7 and epsilon = 0.05 for all features.

$$\min_{\gamma,w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{s.t.} \quad |y^{(i)} - w^T x^{(i)} - b| \geq 1 - \xi_i$$
$$\text{and } \xi_i \geq 0 \quad \text{for } i = 1, 2, \ldots, m$$

Fully Connected Neural Network

The architecture of our FCNN is shown below. The activation function for both hidden layers is the sigmoid function. The output layer uses ReLU as the activation function since energy outputs are non-negative. The optimal learning rate in either case was 5e-7 and the algorithm was run for 1000 epochs with a batch size of 10000 for solar and 100 for wind.



Generalized Additive Model (GAM)

GAMs are smooth models trying to predict:

$$g(\mathbb{E}[y|X]) = \beta_0 + f_1(X_1) + f_2(X_2, X_3) + \cdots + f_M(X_n)$$

by assuming that:

$$y \sim ExponentialFamily(\mu|X)$$

In our case, we assumed the exponential distribution to be Gaussian, the link function to be linear, and each of the function forms f_i to be spline terms.