ECE 598: Computational Power Systems **Winter 2025**

Lecture 9: Introduction to Renewable Energy Forecasting

Lecturer: Vladimir Dvorkin *Scribe(s): Megan Jones & Jo Brooks & Qingyuan Xu & Renjian Ruan*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

9.1 Why Forecasting?

Forecasting is the first step in decision making, particularly in the energy sector where many models rely on forecasts as inputs. By reducing the uncertainty surrounding decision-making inputs, forecasting enables more informed and confident decisions. The ability to anticipate future conditions helps stakeholders plan and allocate resources more effectively.

9.1.1 Who needs a forecast?

A wide range of stakeholders require forecasts. Power producers are among the primary users. Conventional generators rely on demand forecasts to determine unit commitment, while electricity price forecasts are necessary for the optimal allocation of resources to maximize revenue. Renewable energy resource operators, such as wind farm operators, depend on wind and weather forecasts to predict power output. Additionally, they use market forecasts often in combination with weather forecasts to decide when and how much energy to bid into electricity markets.

Utility companies, large industrial consumers, and aggregators also rely heavily on forecasting. For example, utilities use load forecasts to determine how much energy they need to contract from the market. They also engage in market forecasting to decide whether to participate in electricity markets or to enter into power purchasing agreements.

9.1.2 Types of forecasts

Many types of forecasts are needed. These include renewable power forecasts, electricity demand forecasts, day-ahead and real-time electricity price forecasts, and real-time system imbalance or congestion forecasts. Additionally, any other forecast that supports improved decision-making is of value to stakeholders in the energy industry.

9.1.3 Use of Renewable Energy Forecasting in Decision Making

Renewable energy generation forecasts provide inputs to many decision-making problems, including reserve quantification (i.e., backup capacity for the system operator), unit commitment, economic dispatch, contingency screening, and trading strategy for renewables, aggregators, utilities, etc. Unit commitment uses a deterministic forecast; the reserve is then determined based on the forecast's uncertainty margin. These

forecasts can take several forms, but the three most relevant are: (1) deterministic forecasts, (2) probabilistic forecasts, such as quantiles, intervals, and predictive distributions, and (3) probabilistic forecasts in the form of trajectories/scenarios. All of these forecast products are discussed in more detail under SEC 9.2.

9.1.4 Forecasting for Power System Dispatch

From the system operator perspective, there are two stages of decision making to manage the uncertainty of renewables: day-ahead and real-time dispatch. Day ahead dispatch minimizes the cost of supplying power using a renewable energy generation forecast. This decision-making is done, as you could probably guess, the day before. Real-time dispatch happens day-of and does costly re-dispatch to accommodate for forecasting errors by dispatching flexible resources that can ramp up and down quickly. The costs of real-time re-dispatch increase as renewable power capacity increases. Because of this, electricity prices, and subsequently, revenues, are a function of the forecast; thus, grid operators need a "good" power forecast to reduce the cost of uncertainty.

Example: annual re-dispatch costs in Germany [insert bar chart from slide 6 here]

9.1.4.1 Market Outcomes Example

The simulation was conducted on the Dutch electricity pool over the course of a year. It involved a 15 MW wind farm without storage capacity and no control over its output. Both point and probabilistic forecasts were generated using state-of-the-art forecasting tools.

Three different forecast models were evaluated and compared against the benchmark of perfect predictions. The results revealed that market outcomes are highly sensitive to the choice of forecast model. Given these dynamics, which forecasting approach would you choose?

Observations from the table:

Persistence prediction, assumes that future electricity production will remain the same as the most recent production level. Although this assumption ignores changes in weather, demand, or system dynamics it surprisingly performs well in practice due to its simplicity and consistency.

In an ideal scenario, one would naturally opt for a perfect prediction model. If forecasts were 100% accurate, there would be no need for re-dispatching, effectively eliminating associated costs. However, this level of accuracy is not practically feasible. Based on simulation results, probabilistic prediction tends to perform best overall. Probabilistic prediction outperforms other methods because it accounts for the asymmetry between up and down regulation costs. Forecasting in this way typically results in less surplus and more shortage, which is advantageous since down regulation tends to be more expensive than up regulation. This leads to lower overall re-dispatching costs.

[insert table from slide 7 here]

9.2 Forecasting Products

9.2.1 Point Forecast

A point forecast gives information on the conditional expectation of power generation. It gives a single value – the expected value – predicting power generation for a given time.

[insert graph from slide 9 here]

This point taskforce performs well for the first 18 hours before deviating fairly significantly from the observed generation. When we try to predict wind power, we are always wrong.

[insert second graph from slide 9 and wind power zones from slide 8 here]

Here you can see how point forecasts generally follow the trends found in the observations but do experience a reasonable amount of deviation. Additionally, since point forecasts provide no information about a reasonable predictive range, they cannot be used to determine necessary reserves.

9.2.2 Quantile Forecast

Quantile forecasts are a probabilistic threshold for power generation. Quantiles are "cut points" dividing the range of a probability distribution into continuous intervals with equal probabilities. Given a cumulative distribution function of the random process – in this case, wind power generation – the quantile forecast is determined based on the parameter α, which determines the equal probabilities that the continuous intervals have. In FIG [insert ref to figure below here], we see a quantile forecast with $\alpha = 0.5$.

[insert graph from slide 10 here]

9.2.3 Interval Forecast

A prediction interval is an interval within which there is a certain probability the power generation will be. In FIG [insert ref to figure below here] below, the 90% prediction interval is shown.

To understand the boundaries of the shaded region, we need to remember the concept of a cumulative distribution function, or CDF, from statistics. A cumulative distribution function evaluated at x tells you the probability that the random variable will take a value less than or equal to x. This is usually written as CDF(x). The lower boundary of the gray region in the figure below is the 5% CDF, while the upper is the 95% CDF. Thus, the value of the variable has a $95 - 5 = 90\%$ probability of falling within the interval.

As a note, we can recognize that a 100% confidence interval would be useless, as it would just be the entire range from $(-\infty, \infty)$; thus, decision makers must decide the level of confidence to consider. Higher confidence intervals generally correlate with more conservative decisions and higher costs. However, since renewable generation setups tend to be non-profits, they tend to consider high confidence prediction intervals, on the order of 99%.

[insert graph from slide 11 here]

9.2.4 Predictive Density

A predictive density fully describes the probabilistic distribution of power generation for every lead time. They're created by essentially taking several quantile predictions and stacking them together. The objective is to quantify uncertainty around the forecast.

[insert graphs from slide 12 here]

You can use a predictive density product to decide reserve by planning for the expected value, then preparing backup based on the density function. You would pick a quantile to plan for; your choice is generally based on how much reserve you can afford.

9.2.5 Trajectory (Scenarios)

Trajectories are equally-likely samples of multivariate predictive densities for power generation in time and/or space. For these forecasts, you solve for many scenarios, then determine a solution feasible for all scenarios. The idea is that the scenarios will mimic the underlying stochastic behavior of the variable. However, running many scenarios in a complex model can be a computational burden.

[insert graphs from slide 13 here]

9.3 Basics of Forecasting

9.3.1 Wind Power Curve

A theoretical wind power curve, as shown in FIG [insert ref to figure below here] below, maps meteorological features, such as wind speed, to wind power output. Two characteristics of this graph that can be observed are (1) dead band: the section of slow wind speeds where the turbines generate no power and (2) cut-off wind speed: the wind speed at which the power output levels off and further increases in wind speed do not result in further increases in generation. These curves are simple, easy to understand, and can be scaled from one turbine to the entire farm.

[insert graph from slide 14 here]

A practical wind power curve, on the other hand, is very different. There is a lot of uncertainty in the conversion process, which is amplified by weather prediction errors. The practical curve in FIG [insert ref to figure below here] below has a collection of data points; their spread captures the influence of other variables, such as air density and temperature, than can influence turbine output. In practice, operators combine the outputs of many models and assign confidence weights to generate forecasting products. As a note, we can tell that the data in the practical wind curve seen here is from an onshore wind farm, as the majority of data points are at lower wind speeds; offshore wind farms experience higher speeds more often.

[insert graph from slide 15 here]

9.3.2 Linear Regression for Wind Power Curve Fitting

Since dispatch models require a time series single-value, deterministic input for the forecast, we need to translate the practical wind power curve into single values. This will give us data-informed generation predictions. The way to do this is by performing regression on the practical wind power curve. The most straightforward type of regression (that is usually the first applied) is linear regression.

Dataset of $\{(x_1, t_1), ..., (x_i, t_i), ..., (x_n, t_n)\}$ of n observations

Vector x_i of weather features and target t_i for power output.

Objective: Fit a linear model to relate power output to model

$$y = w^T x + b$$

w: weight, b: bias (intercept)

Loss function measures per-instance loss (i.e. how wrong the prediction is).

$$\mathbb{L}(y,t) = \frac{1}{2}(y-t)^2$$

Cost function measures the loss across the entire dataset.

$$\mathbb{C}(y,t) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - t_i)^2$$

To find the w and b that minimize loss, we optimize the model by solving this convex optimization:

$$\min_{w,b} \quad \frac{1}{2n}\sum_{i=1}^{n}(y_i - t_i)^2 \tag{9.1}$$

$$\text{where} \quad y_i = w^T x_i + b \forall i = 1...n \tag{9.2}$$

9.3.2.1 Closed Form Solution

The natural algorithm to solve this problem is to plug the equation for y into the function, take the derivative, set it equal to zero, and solve for b and w. The method for doing so is shown below.

$$X = \begin{bmatrix} 1 & x \end{bmatrix}$$

$$\min_{w,b} \frac{1}{2n}\sum_{i=1}^{n}(w^T x_i + b - t_i)^2 = \min_{w,b} \frac{1}{2n}(X \begin{bmatrix} b \\ w \end{bmatrix} - t)^2$$

$$\frac{1}{2n} 2X^T(X \begin{bmatrix} b \\ w \end{bmatrix} - t) = 0 \tag{9.3}$$

$$X^T(X \begin{bmatrix} b \\ w \end{bmatrix} - t) = 0 \tag{9.4}$$

$$X^T X \begin{bmatrix} b \\ w \end{bmatrix} - X^T t = 0 \tag{9.5}$$

$$X^T X \begin{bmatrix} b \\ w \end{bmatrix} = X^T t \tag{9.6}$$

$$\begin{bmatrix} b \\ w \end{bmatrix} = (X^T X)^{-1} X^T t \tag{9.7}$$

$\begin{bmatrix} b \\ w \end{bmatrix} = (X^T X)^{-1} X^T t$ is the closed-form solution. However, at high dimensions, matrix inversion is a very expensive operation, so this approach can be computationally inefficient. This solution approach is also limited because it only works for the linear model. The limitations of this closed form solution motivate the use of a different approach to solving the optimization problem, which we discussed earlier in the course: gradient descent.

9.3.2.2 Gradient Descent

Add some smoothing sentences

Derivatives of the cost function:

$$\frac{\partial C}{\partial w} = \frac{\partial}{\partial w}\frac{1}{2n}\sum_{i=1}^{n}(y_i - t_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)x_i$$

$$\frac{\partial C}{\partial b} = \frac{\partial}{\partial b}\frac{1}{2n}\sum_{i=1}^{n}(y_i - t_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)$$

For some chosen step size $\alpha > 0$ and iter_max:

For k = 1...iter_max:

$$w \leftarrow w - \alpha\frac{\partial C}{\partial w} \quad \text{weight update} \tag{9.8}$$

$$b \leftarrow b - \alpha\frac{\partial C}{\partial b} \quad \text{bias update} \tag{9.9}$$

Gradient descent is better than the closed form solution because it can be applied to a much broader set of models (not just linear) and is computationally efficient in higher dimensions.

[insert graph from slide 19 here]

9.3.3 Feature Transformation

In real-world applications such as wind power modeling, the relationship between input features (e.g., wind speed u) and output (e.g., power p) is often highly nonlinear. According to the physical law of wind energy extraction:

$$p(u) = \frac{1}{2}C_p\rho A u^3$$

where:

- A is the rotor swept area,
- ρ is air density,
- u is the wind speed,
- C_p is the power coefficient.

Despite this nonlinear relationship, linear regression can still be applied by transforming the input features:

- Add cubic wind speed u^3 as a new feature add citation
- Fit a logistic function (s-shape) to approximate the curve add citation

In these cases, the model remains **linear in the transformed features**.

Limitations. While these transformations can help linear models capture some nonlinearity, they require manual engineering and still assume a predefined functional form. The question arises:

Is there a more flexible (less restrictive) approach to curve fitting?

This motivates the use of **non-parametric models** such as those based on Radial Basis Functions (RBFs), which adaptively approximate the shape of the data without requiring explicit feature transformations.

9.4 RBF-Based Support Vector Regression

9.4.1 Fitting using radial basis functions

In many regression tasks, especially in complex settings, it's often difficult to pre-specify the exact functional form of the relationship between the input variables and the output. Traditional approaches like linear or polynomial regression require a pre-defined model structure, which may be too rigid to capture nonlinear patterns in real-world data.

To address this, we seek a more flexible and expressive approach to curve fitting—one that automatically adapts to the complexity and geometry of the data.

Radial Basis Functions (RBFs) provide such a framework. By constructing the model as a weighted combination of distance-based kernel functions centered at different points, we can approximate a wide class of nonlinear functions without making strong assumptions upfront. This makes RBF-based models highly suitable for data-driven function approximation, such as in Support Vector Regression (SVR).

Radial Basis Functions (RBFs) are a class of real-valued functions φ whose values depend only on the distance between the input vector $\mathbf{x}$ and a fixed point (called the *center*) $\mathbf{c}$. Typically, this distance is Euclidean. Mathematically, an RBF is written as:

$$\varphi_{\mathbf{c}}(\mathbf{x}) = \hat{\varphi}(\|\mathbf{x} - \mathbf{c}\|)$$

where $\hat{\varphi}$ is a radial kernel function, and $\|\cdot\|$ denotes the Euclidean norm.

A set of such functions centered at different locations $\mathbf{c}_1, \ldots, \mathbf{c}_k$ can form a basis for a function space of interest. The function $f(\mathbf{x})$ can then be approximated as a weighted sum of these basis functions:

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i \varphi_{\mathbf{c}_i}(\mathbf{x}) = \sum_{i=1}^{k} w_i \hat{\varphi}(\|\mathbf{x} - \mathbf{c}_i\|)$$

Here, w_i are the weights to be learned, often through optimization (e.g., solving a regression problem or using regularization techniques which we will introduce later).

Example Several popular choices for $\hat{\varphi}(r)$ (also referred to as kernels) include:

- **Gaussian:**
$$\hat{\varphi}(r) = \exp\left(-(\gamma r)^2\right)$$
- **Inverse Quadratic:**
$$\hat{\varphi}(r) = \frac{1}{1 + (\gamma r)^2}$$

- **Inverse Multiquadric:**

$$\hat{\varphi}(r) = \frac{1}{\sqrt{1 + (\gamma r)^2}}$$

- **Thin Plate Spline:**

$$\hat{\varphi}(r) = r^2 \ln(r)$$

Each of these kernels controls the shape and smoothness of the resulting function approximation. For example,

Add the figure in slide page 22

Figure shows an example where a set of Gaussian RBFs (dashed lines) are distributed along the input domain of wind speed. The goal is to learn the optimal weights w_i for each RBF such that their weighted combination accurately follows the shape of the data. In this way, RBFs provide a smooth and adaptive approximation to the underlying wind power curve without assuming a specific function class beforehand.

9.4.1.1 RBF-Based Support Vector Regression

Given a dataset $\{(\mathbf{x}_i, t_i)\}_{i=1}^n$ with input features $\mathbf{x}$ and targets t, we approximate the output y using a weighted sum of RBF basis functions:

$$y = w_0 + \sum_{j=1}^{k} w_j \varphi_j(\mathbf{x})$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_k]^\top$ is the parameter vector to be optimized.

To learn the weights, we minimize the **mean squared error**:

$$\min_{\mathbf{w}} \quad \frac{1}{2n} \sum_{i=1}^{n} \left(w_0 + \sum_{j=1}^{k} w_j \varphi_j(\mathbf{x}_i) - t_i \right)^2$$

This objective is a quadratic function of the weights $\mathbf{w}$. Since the RBF values $\varphi_j(\mathbf{x}_i)$ are fixed once the centers and shape parameters are chosen, the optimization problem is:

- **Convex in $\mathbf{w}$** (i.e., the weights),

Therefore, once the RBF features are set, learning $\mathbf{w}$ is a convex problem and can be efficiently solved using standard methods like least squares or convex solvers.

9.4.1.2 Regularization

To avoid overfitting—especially when the number of RBFs is large—we augment the objective function with a regularization term $\mathcal{R}(\mathbf{w})$:

$$\min_{\mathbf{w}} \quad \frac{1}{2n} \sum_{i=1}^{n} \left(w_0 + \sum_{j=1}^{k} w_j \varphi_j(\mathbf{x}_i) - t_i \right)^2 + \lambda \mathcal{R}(\mathbf{w})$$

Here, $\lambda > 0$ is a small regularization parameter (e.g., 10^{-5}).

Ridge Regression (Tikhonov Regularization). Using the ℓ_2-norm penalty:

$$\mathcal{R}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$$

leads to **ridge regression**. Intuitively it shrinks all coefficients toward zero, but tries to keep them nonzero. This is useful when many RBFs are highly correlated.

LASSO Regression. Using the ℓ_1-norm:

$$\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1$$

leads to **LASSO** (Least Absolute Shrinkage and Selection Operator). LASSO has the effect of driving many weights exactly to zero, which:

- Encourages **sparse solutions**,
- Performs **automatic feature selection** by identifying and using only the most informative RBFs.

Note: Why is it called "LASSO"? The name emphasizes two key properties:

- **Shrinkage**: it pulls weights toward zero.
- **Selection**: it entirely excludes some RBFs by assigning them zero weight—like throwing a lasso and catching only the most important features.

9.4.2 Solution Method : Stochastic Gradient Descent

While the optimization problem for learning the RBF weights $\mathbf{w}$ is convex, solving it directly is often computationally expensive in practice. This is due to the potentially high dimensionality of the feature matrix:

$$\mathbf{X} = \begin{bmatrix} 1 & \varphi_1(\mathbf{x}) & \cdots & \varphi_k(\mathbf{x}) \end{bmatrix}$$

Computing the full gradient over all n samples at each iteration involves costly summations:

$$w_0 \leftarrow w_0 - \frac{\alpha}{n}\sum_{i=1}^{n}\left(w_0 + \sum_{j=1}^{k} w_j\varphi_j(\mathbf{x}_i) - t_i\right)$$

$$w_j \leftarrow w_j - \frac{\alpha}{n}\sum_{i=1}^{n}\left(w_0 + \sum_{j=1}^{k} w_j\varphi_j(\mathbf{x}_i) - t_i\right)\varphi_j(\mathbf{x}_i), \quad \forall j = 1, \ldots, k$$

These updates are computationally expensive to evaluate at every iteration, especially when the dataset is large and the number of basis functions k is high.

To reduce the per-iteration computational cost, we seek an efficient alternative. This motivates the use of **stochastic gradient descent (SGD)**, which approximates the gradient using a small random subset of data points (or even just one sample).

The main idea of Stochastic Gradient Descent(SGD) lies in taking only one sample index i at each iteration, and use the gradient of the problem with the model parameters on index i to conduct a simple gradient descent.

for $k = 1, \ldots,$ `iter_max` **do**

step 1. Sample index $i \sim \mathcal{U}[1, n]$ from a uniform distribution

step 2. Update model parameters on (x_i, t_i) only

$$w_0 \leftarrow w_0 - \alpha\Big(w_0 + \sum_{j=1}^{k} w_j \varphi_j(x_i) - t_i\Big)$$

$$w_j \leftarrow w_j - \alpha\Big(w_0 + \sum_{j=1}^{k} w_j \varphi_j(x_i) - t_i\Big)\varphi_j(x_i) \quad \forall j = 1, \ldots, k$$

end for

The pseud code here shows how SGD are conducted for the problem of RBF-SVM.

One thing to notice here is that it's a special case of SGD of only selecting one randm index among the unifrom distribution 1 to n. The more common way of doing SGD is to randomly shuffle the training set, and then get the average of the sum of all gradients.

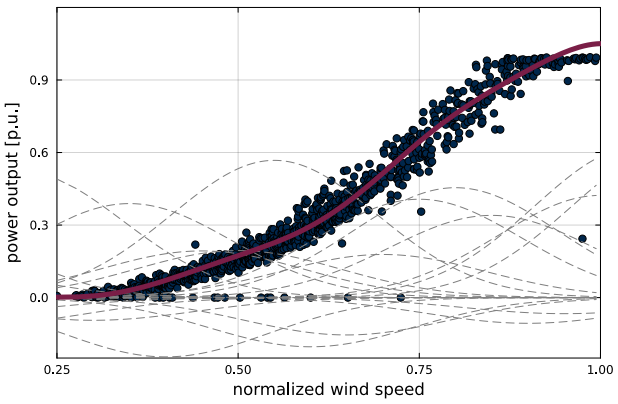
When the training set is too large, we may need to select a mini-batch at each iteration to conduct SGD.

SGD can be combined with regularization, following are two regularizer the lecture included:

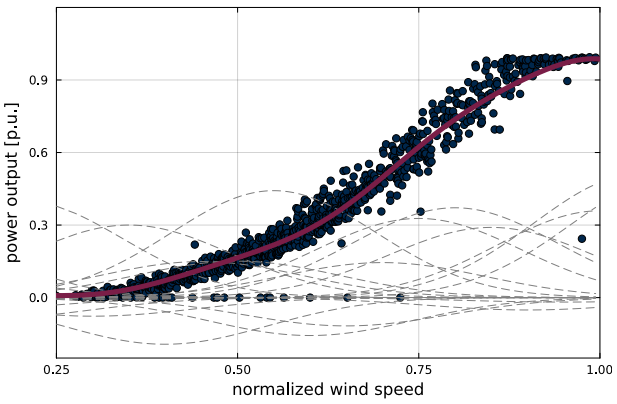
9.4.2.1 Tikhonov Regularization

The Tikhonov Regularization, also more known as l2-regularization, which to some extent reduces overfitting. After adding the regularization, the problem is:

and the SGD is:



no regularization



Tikhonov regularization

9.4.2.2 LASSO Regularization

So LASSO Regularization is also known as l1-regularization. However, the problem is that l1-norm in not smooth, considering the point at 0. So this means that we cannot directly get the gradient of it directly. Instead, we first consider the subgradient of l1 norm for $\forall w_i$, given as:

$$\frac{\partial}{\partial w_0}\lambda|w_0| = \begin{cases} \lambda, & w_0 > 0 \\ -\lambda, & w_0 < 0 \\ \text{any } g \in [-\lambda, \lambda], & w_0 = 0 \end{cases}$$

To make this make sense in terms of gradient descent, we introduce a soft thresholding function, given by:

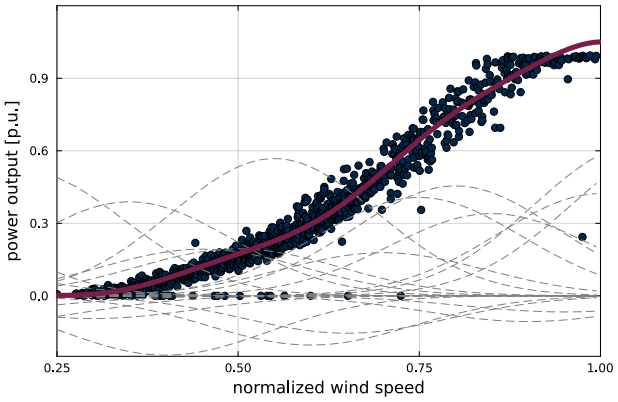
$$S_\tau(x) = \begin{cases} x - \tau, & x > \tau \\ 0, & |x| \leq \tau \\ x + \tau, & x < -\tau \end{cases}$$

So that by using this soft-thresholding function, we can have a gradient descent method:

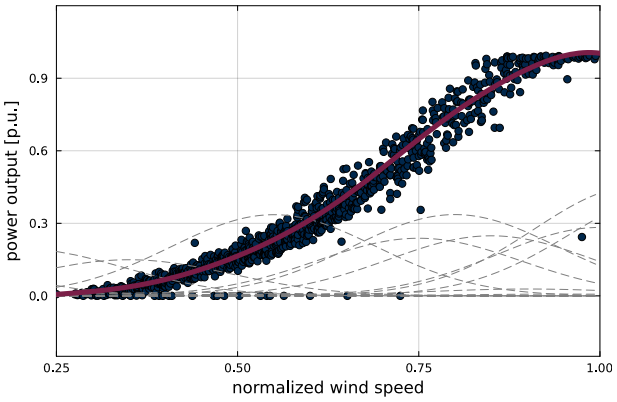
$$r_0 = w_0 + \sum_{j=1}^{k} w_j \varphi_j(x_i) - t_i,\ w_0 = \begin{cases} w_0 - \alpha r_0 - \alpha\lambda, & w_0 - \alpha r_0 > \alpha\lambda \\ 0, & |w_0 - \alpha r_i| \leq \alpha\lambda \\ w_0 - \alpha r_0 + \alpha\lambda, & w_0 - \alpha r_0 < -\alpha\lambda \end{cases}$$

$$r_i = \Big(w_0 + \sum_{j=1}^{k} w_j \varphi_j(x_i) - t_i\Big)\varphi_j(x_i),\ w_i = \begin{cases} w_i - \alpha r_i - \alpha\lambda, & w_i - \alpha r_i > \alpha\lambda \\ 0, & |w_i - \alpha r_i| \leq \alpha\lambda \\ w_i - \alpha r_i + \alpha\lambda, & w_i - \alpha r_i < -\alpha\lambda \end{cases}$$

The result is like:



no regularization



LASSO regularization

It is apparent that after adding this L2 regularization, there are no negative wights, and that the number of weights equals to 0 increase, meaning an increased sparsity of the optimization.