IFPRI Modeling Systems

Informing future pathways and priorities

Simplifying Crop Models into Statistical Emulators of Varying Complexity

IFPRI Modeling Systems Technical Paper No. 3

Richard Robertson

International Food Policy Research Institute

The International Food Policy Research Institute (IFPRI), a CGIAR Research Center established in 1975, provides research-based policy solutions to sustainably reduce poverty and end hunger and malnutrition. IFPRI's strategic research aims to foster a climate-resilient and sustainable food supply; promote healthy diets and nutrition for all; build inclusive and efficient markets, trade systems, and food industries; transform agricultural and rural economies; and strengthen institutions and governance. Gender is integrated in all the Institute's work. Partnerships, communications, capacity strengthening, and data and knowledge management are essential components to translate IFPRI's research from action to impact. The Institute's regional and country programs play a critical role in responding to demand for food policy research and in delivering holistic support for country-led development. IFPRI collaborates with partners around the world.

www.ifpri.org

CGIAR

CGIAR is a global research partnership for a food-secure future, dedicated to transforming food, land, and water systems in a climate crisis. CGIAR science aims to reduce poverty, enhance food and nutrition security, and improve natural resources and ecosystem services. As the world's largest agricultural innovation network, its research is carried out by 15 CGIAR Centers working around the world in close collaboration with hundreds of partners, including national and regional research institutes, civil society organizations, academia, development organizations, and the private sector.

www.cgiar.org

Authors

Richard Robertson (r.robertson@cgiar.org) is a Research Fellow in the Foresight and Policy Modeling unit of the International Food Policy Research Institute in Washington, DC, United States of America.

Recommended Citation: Robertson, R. (2026). Simplifying crop models into statistical emulators of varying complexity. Modeling Systems Technical Paper No. 3. International Food Policy Research Institute.

International Food Policy Research Institute
1201 Eye Street, NW
Washington, DC 20005-3915 USA
www.ifpri.org

Contents

Tables

Figures

ABSTRACT

Process based crop simulation models can be used to anticipate how crop yields behave under different weather conditions which makes them useful for assessing the implications for yields of weather variability, uncertainty, and climate change. However, they can be cumbersome to use or embed in other models. Hence, developing statistical models that emulate the full models can provide a lighter-weight means of harnessing their power for rapid or high-volume assessments.

We develop emulators for several major crops. Climate and fertilizer rates are matched to the yields simulated by a full crop model to provide a dataset covering the entire land surface of the globe. We use neural network specifications to be able to easily try out different levels of complexity. The final selection of the emulator models is a balance between complexity and performance.

The large amount of data meant that the emulators could support complicated specifications. We ultimately settled on models with approximately 200 parameters. Neither rainfed nor irrigated conditions were consistently better performing. In general, the r-squared values of the estimated emulators ranged from 0.85 to 0.95. No crops appeared significantly easier or more difficult to model than the others.

The emulators we developed provide a simple substitute for more complicated crop models when a high level of detail is not needed. The approach can be easily updated with new data or be applied to more specific circumstances if necessary by creating custom datasets that emphasize the particular conditions anticipated to be most important.

Keywords: crop modeling, model emulator

ACKNOWLEDGMENTS

Support for this research was provided by the Community Jameel, the CGIAR Policy Innovations Program, and the CGIAR Foresight Initiative. Review comments from Diego Pequeno guided improvements to this manuscript and are gratefully appreciated.

ACRONYMS

CO2: carbon dioxide, a chemical and gas generated by burning or decomposing other chemicals containing carbon such as wood or fossil fuels

DSSAT: the Decision Support System for Agrotechnology Transfer, a framework and family of process-based crop simulation models

GCM: Global Circulation Model, a major type of computational model used to analyze and simulate the climate

IPSL-CM6A-LR: a particular GCM developed at the Institut Pierre-Simon Laplace

IR: a shorthand for irrigated conditions

MPI-ESM1-2-HR: a particular GCM, the Max Planck Institute Earth System Model

ppm: parts per million, a unit of concentration often used for measuring how much carbon dioxide is in the atmosphere

RCP: Representative Concentration Pathway, a standardized scheme laying out atmospheric carbon dioxide concentrations for the future as part of scenarios when simulating future climates using global circulation models

RF: a shorthand for rainfed conditions

SPAM: the Spatial Production Allocation Model, a collection of raster maps showing how much area is dedicated to the cultivation of different crops across the world

SSP3/RCP7.0: a particular scenario for simulating future climates consisting of assumptions regarding general economic and policy conditions (Shared Socioeconomic Pathway #3) and greenhouse gas concentrations (Representative Concentration Pathway known as 7.0)

SIMPLIFYING CROP MODELS INTO STATISTICAL EMULATORS OF VARYING COMPLEXITY

Process-based crop simulation models provide a detailed picture of how crops might grow under a variety of conditions. Such detail in outputs requires detailed inputs, detailed settings, and computational resources. As such, using a full-blown crop model can be impractical if a wide variety of cases are to be analyzed or if it would be helpful to embed the crop model inside a larger, say, economic model. Hence, it can be useful to develop simplified summary or statistical models to "emulate" the more complicated models.

By design, an emulator of this sort will not perfectly reproduce the process-based model. So, when developing them, the question arises: what sort of trade-off is there between model complexity and performance? Compared to complicated models, simple models are more readily estimated, less prone to coding/user errors, and less likely to be led astray by the peculiarities of a particular dataset. However, they are also less likely to adequately represent the underlying patterns, which is the entire purpose of a statistical model in the first place. Here, we report our experience in applying statistical models of varying complexity to develop emulators of process-based crop models.

1. Neural networks as a convenient specification

We employ a basic neural network approach to allow for easy changes in the complexity of the model. The advantages of a neural network specification are that it is smooth, flexible, and easy to adjust the level of that flexibility. The major downside is that estimation is never perfectly optimal nor completed. In contrast, specifications based on polynomials or piecewise domains, (or both) can be quickly and optimally estimated. However, polynomials can have difficulty maintaining flexibility over wide domains and piecewise models exhibit discontinuities either in the actual values or in rates of change. Both polynomial and piecewise models suffer from the curse of dimensionality where the number of desired parameters can outrun the data available to estimate them. Since the objective of all of these approaches is to more-or-less faithfully reproduce the patterns in some dataset, with sufficient effort, they all tend to end up with roughly the same level of performance. Of course, a simple neural network will be worse than a complicated piecewise polynomial and vice versa; the point is that there is no secret sauce that makes some particular approach vastly superior to any other.

As usual, we start with a set of explanatory variables matched with target values to be reproduced. In this particular case, we are predicting crop yields based on growing conditions such as rainfall.

The neural network computes the predictions (yields, for us) in a series of steps reminiscent of linear regression. The values of the explanatory values (growth conditions in our case) are first transformed by a collection of functions known as "neurons". These operate by multiplying the values of the explanatory variables by parameters (historically known as weights) and summed along with a constant parameter (historically known as the bias). The (linear) total is then transformed by a (non-linear) "activation function". There are many reasonable possibilities for the activation function. For computational convenience, we use the cumulative logistic function to monotonically convert the value to a number between zero and one. We can pick how many of these neurons to use and give them each their own parameters. To obtain the final predicted value we multiply each individual neuron activation by another weight and add them up along with a constant parameter. The parameters to be estimated are the weights connecting the explanatory/input values to the "hidden neurons" and those connecting the hidden neurons to the final output along with the bias/constant terms for each of those.

A way to think of how this introduces flexibility is that a single neuron does something like a linear regression on the inputs. When the output of that linear model is run through the activation function, the very large negative values are squashed nearly to zero and the large positive values constrained below one. This limits the actual response of the neuron to a sub-domain of the explanatory variables. By having multiple neurons, each can work on a different piece of the domain and smoothly hand off to each other. The result is that we can have the flavor of a piecewise model without discontinuities, arbitrary break points, or forcing the pieces to match at the edges.

The generic neural network model specification lends itself to an easy, if coarse, manipulation of how flexible the model is: simply pick different numbers of neurons. As the number of neurons increases, the model can accommodate more complicated shapes. Of course, estimating the appropriate parameters becomes more difficult along with more opportunities for duplication and symmetry that must be avoided to take advantage of the flexibility. Some simple examples of how a neural network can approximate an arbitrary shape are provided in the first Appendix to demonstrate the effect of changing the number of neurons.

The various parameters (relating the inputs to the "neurons" and from the "neurons" to the output) are typically estimated by optimization. In our case, we employ the least-squares principle which is exactly the same as is used in classical regression. In traditional linear regression, the straightforward structure of the main part of the model means that there is a single, unique best choice of parameter values that minimizes the squared error (and with the appropriate assumptions, maximizes the likelihood). As a result of the non-linear "activation" part of the neural network specification and the various symmetries, there is no single optimum to be obtained (and sometimes no true optimum at all). Still, though estimation may be never ending (often asymptotically approaching an optimal value), it can proceed using variations on gradient descent. Deciding when to stop is based on balancing the benefits of the prospects of a better fit and the costs in time and effort to obtain them. Of course, minimizing the error depends on being able to make a prediction and then seeing how close to the target it is.

Mathematically, here is how the neural network combines the variable and parameter values to compute a predicted value. Suppose we have a single output/dependent variable to understand based on K input/explanatory/independent variables. Furthermore, let us have H neurons in our network besides the inputs and the output ("H" is for "hidden"). For each neuron called h, there will be (K + 1) parameters: one coefficient for each explanatory variable (indexed by k) and one extra for a constant. The neuron first computes the linear combination, that is, the sum for hidden neuron h as

$$S_h = \beta_{h0} + \sum_k x_k \beta_{hk}$$

This sum is then transformed by the activation function, which in our case is the cumulative logistic function:

$$\Lambda(a) = \frac{1}{1 + e^{-a}}$$

Thus, the neuron's activation value is:

$$\Lambda(S_h) = \Lambda\left(\beta_{h0} + \sum_k x_k \beta_{hk}\right) = \frac{1}{1 + e^{-(\beta_{h0} + \sum_k x_k \beta_{hk})}}$$

The final prediction ($\hat{Y}$) from the entire network is a linear combination of the individual neuron activations:

$$\hat{Y} = \alpha_0 + \sum_h \Lambda(S_h)\alpha_h = \alpha_0 + \sum_h \Lambda\left(\beta_{h0} + \sum_k x_k \beta_{hk}\right)\alpha_h$$

As a result, the total number of parameters is the number of parameters connecting the inputs to the "hidden" neurons (along with their constants) plus the number of parameters connecting the "hidden" neurons to the output (along with a different constant); that is, total number of parameters = H×(K + 1) + (H + 1) = HK + H + H + 1 = H×(K + 2) + 1.

Also, note that since the neurons are interchangeable, there are myriad symmetries. Some examples are simply flipping all the signs on the parameters from the inputs to that neuron (which flips all the activation values to one minus the original) along with the sign of the parameter connecting that neuron to the output, and an appropriate adjustment to the constant term on the output. We can also simply swap the numbering of any two neurons (for example, make what was originally neuron #1 into the new #7, and turn #7 into the new #1).

2. Modeling choices for application to crop yields

There are a few modeling choices made to facilitate our focus on crop yields. The guiding principle for making these decisions is that the emulator should take a set of conditions and provide a yield typical of what would be seen if reasonable choices were made regarding variety, planting date, and so on.

The data consist of simulated yields for rice (japonica and indica), wheat (spring and winter), maize, groundnuts, soybeans, sorghum, and potatoes, generated by the Mink system (Robertson, 2017) which employs the DSSAT family of models (Hoogenboom, et al., 2024; Hoogenboom, et al., 2019; Jones, et al., 2003). The climate data for both the crop models and the emulators are based on the Princeton Global Forcing (Sheffield, et al., 2006) for the baseline/historical weather along with possible future weather conditions developed using on adjustments based on the ISIMIP3b datasets (Lange, et al., 2024).

The first modeling choice we had to make is due to the form that the data take. All of the data were originally generated as maps covering almost every bit of land in the world at a resolution of 0.5 arc-degrees. This provides climatic differences by looking across different places. In addition to using historical climate conditions, we also have yields that are based on future climates typical of 2050 for 5 different GCMs and 3 RCPs. This provides an even wider variety of climate conditions prospectively through time. For each future climate, we have two simulations: one that keeps atmospheric carbon dioxide concentrations at the same level used for the historical yield simulations and one with the carbon dioxide concentration associated with the RCP in 2050. All of these data are pooled together into a single large dataset for estimating parameters for the emulator models.

There is a wrinkle introduced because of the pixel map format: each row of pixels occupies a different amount of area on the ground. This is because we are using latitude-longitude maps and the distance between two lines of longitude is wide near the equator and narrow near the poles (while the "height" of the pixels remains effectively constant). To maintain the relative importance of each observation, when estimating the parameters, we weight the contribution of each pixel to the overall error by the size of the pixel.

Additionally, in the training or estimation process, we make use of simulations based on the variety of climatic conditions in both recent historical as well as plausible future climates. We are only using one

"past". There are many possible "futures" (15, in our case; 30 when counting the carbon dioxide variations). While we are very interested in the future conditions, we do not want them to overwhelm the historical conditions. After all, the crop models were built on historical conditions and are probably most reliable in that domain. To maintain some balance, we decided to count the historical climate as 10 times more important than any single one of the future datasets. While making this decision, we explored a small number of cases, varying the weighting between the historical and future climates. Compared to the traditional uniform weighting of ones for included datapoints, we do end up with different estimates and predictions, but not wildly so. This is not surprising since, across the globe, while there are projected to be brand new climatic conditions, many of the future climatic conditions are still similar to historical ones, just in different places. For example, a medium region which warms a little in the future would be similar to a mildly warm region in the recent past. Employing a plausible weighting scheme does not magically result in a perfect model nor does it catastrophically degrade the exercise.

The second decision regards the choice of crop varieties. For any crop of interest, there are many possible varieties which each respond to climate and management differently. Some of these varieties may be very good only in a narrow set of conditions while others might behave similarly across many different situations. In general, we employ several varieties and then choose the highest yielding variety for each location and climate combination to represent the crop yield for that situation. Rather than trying to replicate the yield for "maize #17" in a particular kind of climate, we are looking for the potential yield that could be expected if an appropriate variety of maize were grown in that climate. This is roughly like mega-environment approach to breeding strategy where the world is broken up into several different zones with each zone sharing climate and other attributes. Then breeding efforts can be focused on adapting and improving varieties appropriate to each mega-environment. In our case, we are identifying which of the varieties available to us perform best in each location. The emulator then encapsulates the overall behavior of the crop when the appropriate variety is being used. Having a small number or single emulator per crop helps fulfill the purpose of developing the emulators which is to have easier, more compact ways to simulate crop yields.

The explanatory variables are kept simple. Most of the variables represent the climate: rainfall and high temperature. The emphasis in on the first few months of the presumed growing seasons. Of course, every growing season requires the initial months, but only long growing seasons require later months. We use the monthly averages of the high temperatures in the first four months of the growing season and the monthly total rainfall for each of the first four months.

The values of the temperatures in degrees Celsius are used directly.

Rainfall poses challenges. The distribution of rainfall values is quite wide with many low values and a few very high values (see Appendix B). Having such disparate values makes it more difficult for the numerical machinery of the optimization process.

The flexibility of neural networks means that they are able to adapt to monotonic transformations of the inputs. We can take advantage of that property to make it easier for the optimizer to find good parameters. When the distribution of an explanatory variable has a few uncommon but very different values stretching out away from the main body, the disparity makes it difficult for optimizers to work toward better parameter values. Optimizers working under these conditions are typically quite sensitive and can easily lose their way or end up proceeding very slowly. The extreme explanatory variable values can exert an outsized influence on the error, making it hard to determine the correct direction that will reduce the error. Conversely, when the explanatory variable values are packed too densely together, it is hard for an optimizer to distinguish the right parameter values to match what might be a rapidly changing curve in that small space. Rainfall values usually exhibit both of these problems: a huge spike of exactly or nearly zero values followed by a fairly dense interval of low to moderate values paired with a very long, thin tail.

We would expect there to be important effects on yields in the narrow range when rainfall goes from nothing to something at the scale of tens or hundreds of millimeters. Other effects should appear at higher amounts, but changing more slowly over hundreds and thousands of millimeters.

So, instead of using the rainfall values directly, we use their square root. Employing a transformation like square root helps to even out the distribution of values by spreading out the very small values while bringing the extremely large values closer in. The intention is to have the curves in the rainfall/yield relationship be of similar sizes rather than some needing to be tight and sharp with others being loose and wide.

In some rare cases, due to the choice of planting month, there is no rainfall at all during the growing season and yet the crop model predicts positive yields. This can occur because rainfall during the 3-month "spinup period" before planting can partially remain in the soil with the plant growing on the residual moisture. In an attempt to capture this, we also include the square root of the average monthly rainfall during the spinup period.

The last two variables are very coarse. The atmospheric carbon dioxide concentration is the same for all locations. Its variability is across time. The baseline climate is associated with 379 parts per million. We include two versions of the future climate cases. The "optimistic" cases use the concentration associated with the Representative Concentration Pathway for 2050 (469ppm for RCP 2.6, 541ppm for RCP 7.0, and 563ppm for RCP 8.5), calling upon the full expression of carbon dioxide fertilization as embodied in the crop model. The "pessimistic" cases reflect the other extreme of carbon dioxide fertilization failing to be realized and hence the baseline level of 379ppm is maintained.

Fertilizer application rates are kept constant across all time periods with the variability being across space. Different countries and sometimes sub-regions within countries can have different fertilizer applications rates. For soybeans and groundnuts, no fertilizer is used anywhere and so that variable is not included when building the emulators.

3. Results

We develop emulators for seven different crops: groundnuts, maize, potatoes, rice, sorghum, soybeans, and wheat. Rice and wheat are treated slightly differently from the rest in that we choose two representative cultivars and estimate separate emulators for each of them. Rice has two major types: japonica which is more suited for colder environments and indica which is more suited for warmer environments. Wheat is produced in two major ways, spring wheat (a straightforward growing season) and winter wheat (wherein the growing season is interrupted by a dormant phase with colder temperatures). All crops are further split into idealized irrigated conditions and purely rainfed.

Emulator performance by specification

In each case, we estimate a whole series of neural network specifications with the number of hidden neurons ranging from a single one up to sixteen. Since our datasets are reasonably large (1,742,789 datapoints for most crops) compared to the numbers of parameters we are estimating (roughly 200), we are able to split them into training and validation sets. The training set was used to estimate the parameters based on only 1% of the overall dataset which was still plenty. We did explore different fractions. Larger amounts did not provide meaningfully different performance but took much longer to compute.

Using the candidate parameters, we make predictions for both sets and used them to compute performance measures (root mean squared error and r-squared). By comparing the measurements between

the training and validation sets, we can see the degree of overtraining that may or may not be occurring. In the extreme, an "ID number" model could be perfectly correct for all of the training data but would be disastrously mistaken on the validation set. A good specification will usually have better performance in the training set than validation, but not greatly so. As the model specifications become more complicated (that is, moving more closely to an ID number model), we would expect the performance gap between the two sets to increase. We can know that a specification is too complicated when its validation performance is worse than the validation performance of a simpler and hence more appropriate specification. In practice, with large datasets, this does not happen readily; rather, large increases in complexity lead to negligible improvements in performance and hence are not worth the effort to estimate them.

As an example, consider how the model performance for groundnuts changes as the number of neurons increases. In Figure 3.1, each point represents a particular number of hidden neurons and how they performed. On the far left with the fewest parameters are the single hidden neuron specifications which are roughly analogous to a linear model. Each subsequent point represents an additional hidden neuron up to seven followed by a gap all the way up to sixteen (on the far right). For similar graphs of the remaining crops, see Appendix C.

The blue lines show the r-squared value for the comparatively small training set and red for validation. The neural networks for the rainfed case (the solid lines) perform poorly until the fourth neuron is specified, but improvements from additional neurons (and their attendant parameters) become more modest.

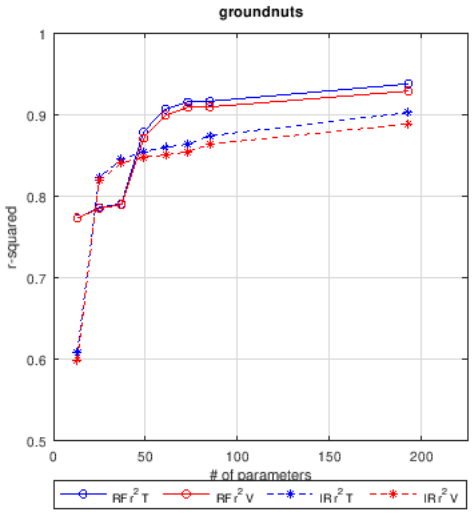

Figure 3.1. Model performance by number of parameters for candidate emulators for groundnuts. Circle connected by solid lines are for rainfed conditions; asterisks connected by broken lines are for idealized irrigated conditions. Blue shows the performance of the emulators on the set of datapoints used to estimate the parameters (the training set) while the red shows the performance on the remaining data (the validation set). The large amount of data can support specifications with many parameters.

The gap in performance between the training and validation sets (Figure 3.2) generally widens as more neurons are added (and more effort is put into optimization), but is still quite narrow at 16. This indicates that the size of the dataset is enough to support the model's $H\times(K+2) + 1 = 16\times(9+2) + 1 = 177$ parameters (recall that H = number of hidden neurons; K = number of explanatory variables). We explored even more complicated models up to 32 neurons for one or two cases but found that they did not perform better. The other crops show a similar pattern across the specifications. The diminishing

improvements lead to our decision that a 16 neuron specification is a good compromise between performance, complexity, and effort.

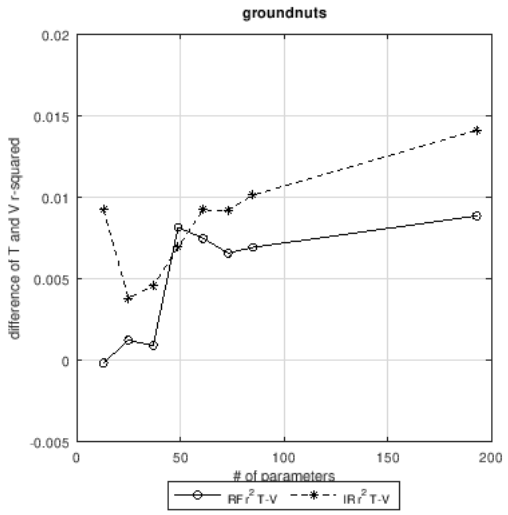

Figure 3.2. Difference between training and validation performance by number of parameters for candidate emulators for groundnuts. The relative performance in and out of sample is very close, but is slightly larger for the most complicated specification employed.

Reproduction of climate change effects

The motivating application for developing these emulators is to compute climate change effects on yields in a simpler way than using a full process-based crop simulation model. For both the full process-based model and the estimated emulators of them, we compare two future climate cases with the historical climate. Specifically, we look at climates typical of 2050 under the SSP3/RCP7.0 scenario with the first coming from MPI-ESM1-2-HR with 379ppm atmospheric carbon dioxide and the IPSL-CM6A-LR with 541ppm atmospheric carbon dioxide. These reflect a pessimistic view of the potential for carbon dioxide fertilization by keeping the same level as the 2005 level and an optimistic view using the level specified by the scenario.

We consider two ways of looking at the effect of climate change on simulated yields. The first way is to create maps showing the type of yield change at each point on land across the globe. Of course, there will be many pixels where the crop is not currently grown nor is likely to ever be grown; we want to be able to look at all of the possibilities. Still, to avoid distractions in presentation of the maps, we only show changes in locations where we are confident that simulated yields are meaningfully large and reliable; specifically, higher than 500 kg/ha dry weight. The map of changes shows how yields change between the baseline and future conditions, grouped into broad categories for large and small gains and losses along with minimal and exactly zero changes. We did this for both the raw crop modeling "target" values and for the emulators' approximation.

As an example, again consider rainfed groundnuts, low carbon dioxide, MPI-ESM1-2-HR, 2050 conditions under SSP3/RCP7.0. The map in Figure 3.3 shows the changes based on the crop modeling results. It shows the usual pattern that historically low yielding areas in the far north and south benefit from warmer temperatures while areas that start out warmer are hurt by the even warmer future conditions.

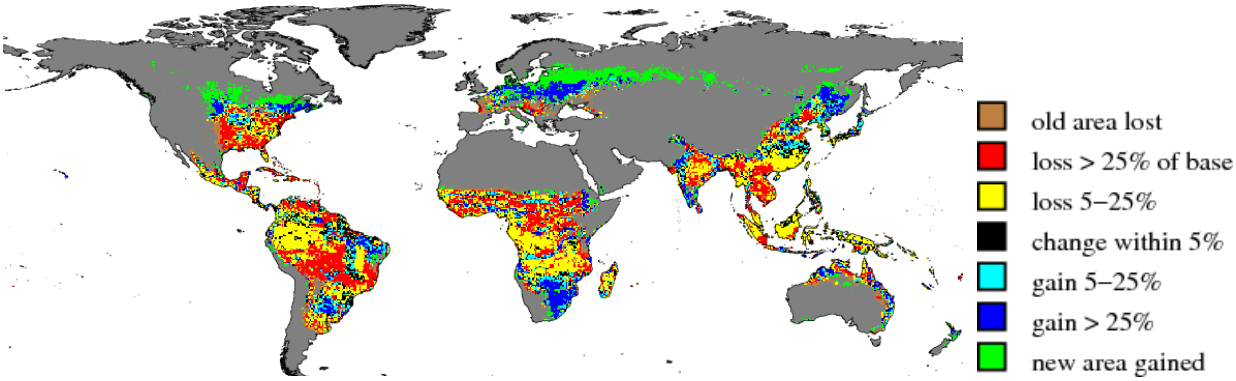

Figure 3.3. Map of the original, full crop model simulation of the effect of climate change on rainfed groundnuts yields (2005 to 2050). Climate based on MPI-ESM1-2-HR under SSP3/RCP7.0.

Using the emulator, we created the same type of map for Figure 3.4.. The broad pattern is reproduced, although the individual pixels are often different from the results using the direct crop modeling values.

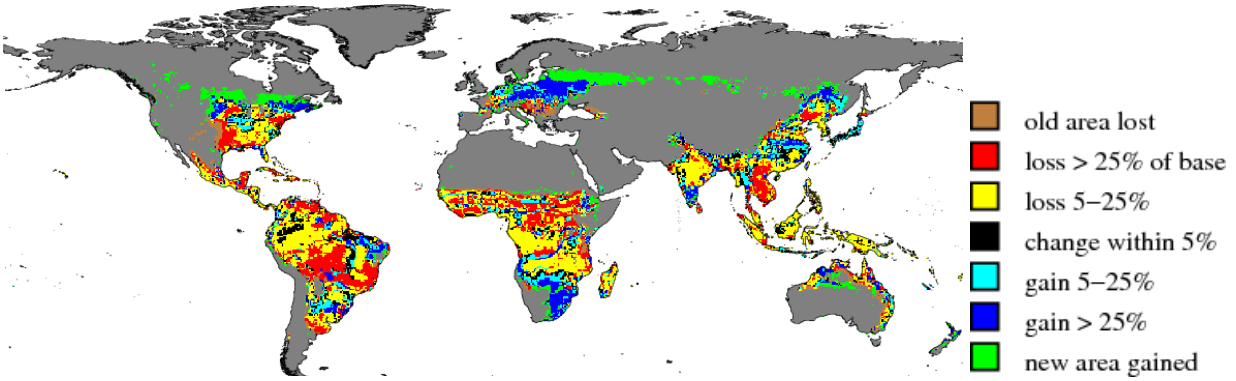

Figure 3.4. Map of the emulator simulation of the effect of climate change on rainfed groundnuts yields (2005 to 2050). Climate based on MPI-ESM1-2-HR under SSP3/RCP7.0.

The second way to verify the utility of the emulators under climate change is to look at regional area-weighted average yields. This involves multiplying the pixel-level yields by some appropriate pixel-level areas to obtain pixel-level production analogues. Then, the pixel-level production analogues are added up over a region of interest and divided by the sum of the area to obtain the weighted average. The fractional climate change effect is the change in yield (between the future case and the historical case) divided by the historical case. For simplicity, we look at the entire world as our region. We also consider two schemes for the area weighting: the historical crop areas for each crop in 2005 (as enumerated by SPAM maps; IFPRI 2016) and separately with the entire land surface of the world using the physical area of each pixel.

Compared to the original crop modeling results, the emulators are reproducing similar yield changes at the worldwide level. This can be seen by looking at the numbers in several ways. First, we list out some fractional yield changes using the SPAM weighting (Table 3.1) and then provide graph versions that also show all land area (Figure 3.5 and 3.6). Notice that in general, the scenario employing the optimistic CO2 assumptions is generally more optimistic than the pessimistic one (except for maize and sorghum which are so-called C4 plants that are not as limited by CO2 availability).

Table 3.1. Global average yield changes on appropriate cropland as simulated by emulators and full crop models.

climate scenario	MPI-ESM1-2-HR 2050 RCP7.0 pessimistic CO2		IPSL-CM6A-LR 2050 RCP7.0 optimistic CO2	
	emulator	DSSAT	emulator	DSSAT
groundnuts/RF	-10%	-15%	0%	-1%
maize/RF	-15%	-11%	-17%	-17%
potatoes/RF	-8%	-8%	1%	1%
rice (indica)/RF	-9%	-8%	11%	6%
rice (japonica)/RF	-3%	-3%	4%	3%
sorghum/RF	-12%	-13%	-14%	-14%
soybeans/RF	-7%	-7%	6%	12%
wheat (spring)/RF	-9%	-8%	-2%	-5%
wheat (winter)/RF	9%	12%	28%	26%
groundnuts/IR	-9%	-12%	4%	-1%
maize/IR	-11%	-15%	-14%	-19%
potatoes/IR	-5%	-6%	5%	4%
rice (indica)/IR	-2%	-6%	8%	6%
rice (japonica)/IR	-15%	-15%	0%	-4%
sorghum/IR	-3%	3%	-8%	-4%
soybeans/IR	-7%	-8%	9%	7%
wheat (spring)/IR	2%	-2%	12%	15%
wheat (winter)/IR	1%	-2%	2%	1%

When considering the entire land surface of the world, the emulator yields align more closely with the original crop modeling results than when looking at only cropland areas. This manifests itself in that the points are more closely clustered around the diagonal line in Figure 3.6 (using all possible land areas) than they are in Figure 3.5 (which looks only at appropriate SPAM crop areas). The main reason for this is that there are many places that are very low yielding both in the past and likely in the future; hence, not contributing much of a change. The SPAM weighting, by definition, concentrates on those place that have a good chance of a good yield and hence have more opportunity to feel an effect as the climate changes. In general, the emulator is a bit more optimistic than the raw crop model (with the SPAM weighting, 67% of the crops are more optimistic and with all areas, 72% are more optimistic).

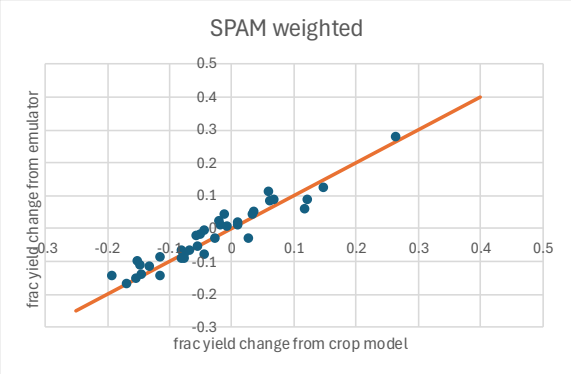

Figure 3.5. Yield changes from emulator and full crop model as a scatterplot, weighted by appropriate SPAM cropland areas.

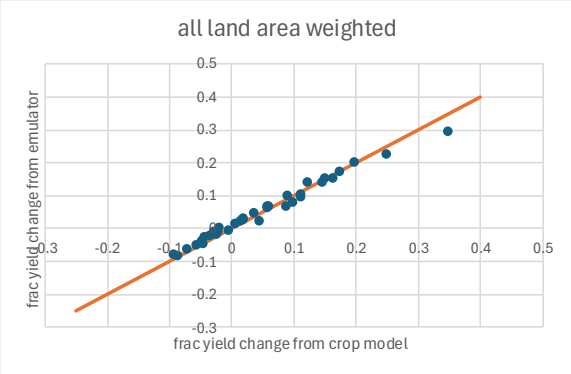

Figure 3.6. Yield changes from emulator and full crop model as a scatterplot, weighted by all land area.

Perhaps some crops are easier to model than others and would consistently have a better match between the emulators and the original crop models for the climate change effects. It turns out that among the crops we dealt with, that is not the case. To lay this out, we compute the discrepancy between the emulator's prediction of the change and the original's. Then we put them in order with the left being the crop where the emulator predicts too negative of an effect and the rightmost where it is too positive. For the two climate situations we are using as examples, the orders are completely different. This is true whether SPAM areas are used for weighting the yields (Figures 3.7 and 3.8) or all land areas (Figures 3.9 and 3.10). So, we can see that there is no simple pattern of which crops have good agreement between DSSAT and the emulator.

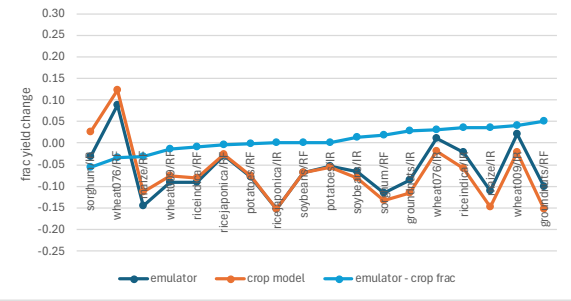

Figure 3.7. Difference between emulator and full crop model climate change effects. Future climate based on MPI-ESM1-2-HR, pessimistic CO2, using appropriate SPAM cropland area as weighting.

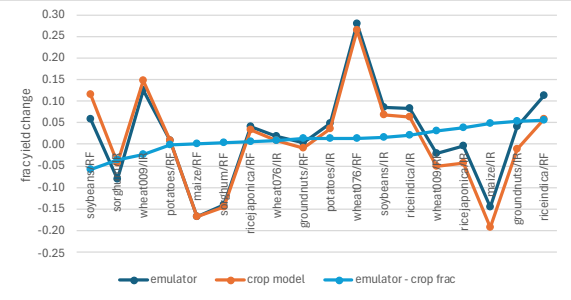

Figure 3.8. Difference between emulator and full crop model climate change effects. Future climate based on IPSL-CM6A-LR, optimistic CO2, using appropriate SPAM cropland area as weighting.

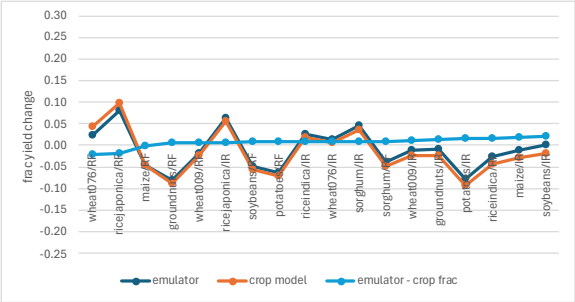

Figure 3.9. Difference between emulator and full crop model climate change effects. Future climate based on MPI-ESM1-2-HR, pessimistic CO2, using all possible physical area as weighting.

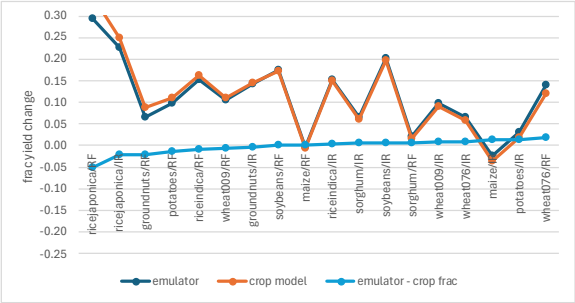

Figure 3.10. Difference between emulator and full crop model climate change effects. Future climate based on IPSL-CM6A-LR, optimistic CO2, using all possible physical area as weighting.

4. Conclusions

Statistical emulators of crop models create opportunities for rapid assessment of crop yields across a variety of hypothetical situations. Their comparative simplicity would allow them to be used in investigating climate variability across very large numbers of possibilities or to be embedded inside other models. Taking advantage of our existing capabilities of using process-based crop simulation models at the global scale, we developed statistical emulators for seven major crops.

Although the purpose of an emulator is to be “simple”, we found that they must not be too simple. The way that plants respond to their environment is complicated and crop simulation models reflect many of those complications. By using neural network specifications, we were able to easily control the complexity of the emulators. We found that simplest models were inadequate; or, at least, that allowing a moderate amount of complexity boosted performance dramatically and that the datasets could support quite a bit of complexity before the improved performance was no longer worth it.

The original crop modeling results indicated that under pessimistic assumptions about the efficacy of CO2 fertilization, yields on historical cropland areas are likely to decline between zero and 15% by 2050 compared to 2005; optimistic assumptions about CO2 could roughly restore those losses. The emulators were able to capture most of the yield responses simulated by the crop models and are ready to be tested in broader applications.

Finally, having established a process for developing the emulators, we will be able to create or update emulators to reflect future research needs as additional information becomes available.

Appendix A: Stylized example comparing optimal polynomials and sub-optimal neural networks

Let us consider the simple case which is difficult to model with simple regression techniques: some sharp transitions in a single variable that, when graphed, look like islands of high values sticking up above a sea of low values. We need to build a dataset of explanatory variables and the corresponding targets to be modeled. Starting with 1000 uniformly, randomly distributed points on the interval from -5 to 5 for the explanatory variable, we built two of these graphical "islands" having value one between +1 and +3.5 and the other between -2 and -1 while the "sea" has a value of zero. We then used 334 of the points in this constructed dataset to estimate the models and the other 666 as validation.

First, we can consider the polynomial approach, for which the optimal parameters can be directly estimated. The "islands" and the set of polynomial approximations can be seen in Figure A.1. If we use a truly linear model, it will not fit very well, but there will be some optimal slope and intercept that will do less badly than any other combination. In this case, the fitted line splits the difference while tilting toward the side with more datapoints. We can progressively add more polynomial terms and with enough, we can obtain a decent fit. However, this example involves a single explanatory variable, so the number of parameters is kept under control. When multiple explanatory variables are involved, the number of parameters can quickly become unwieldy or even overwhelm the number of datapoints available (the classic curse of dimensionality). Using polynomials, it takes a 6th order specification to capture the general pattern and many more to begin to get close. In the following graph, we can see how additional terms improve the general shape and performance. It shows polynomials of order 1 to 6 as well as 17.

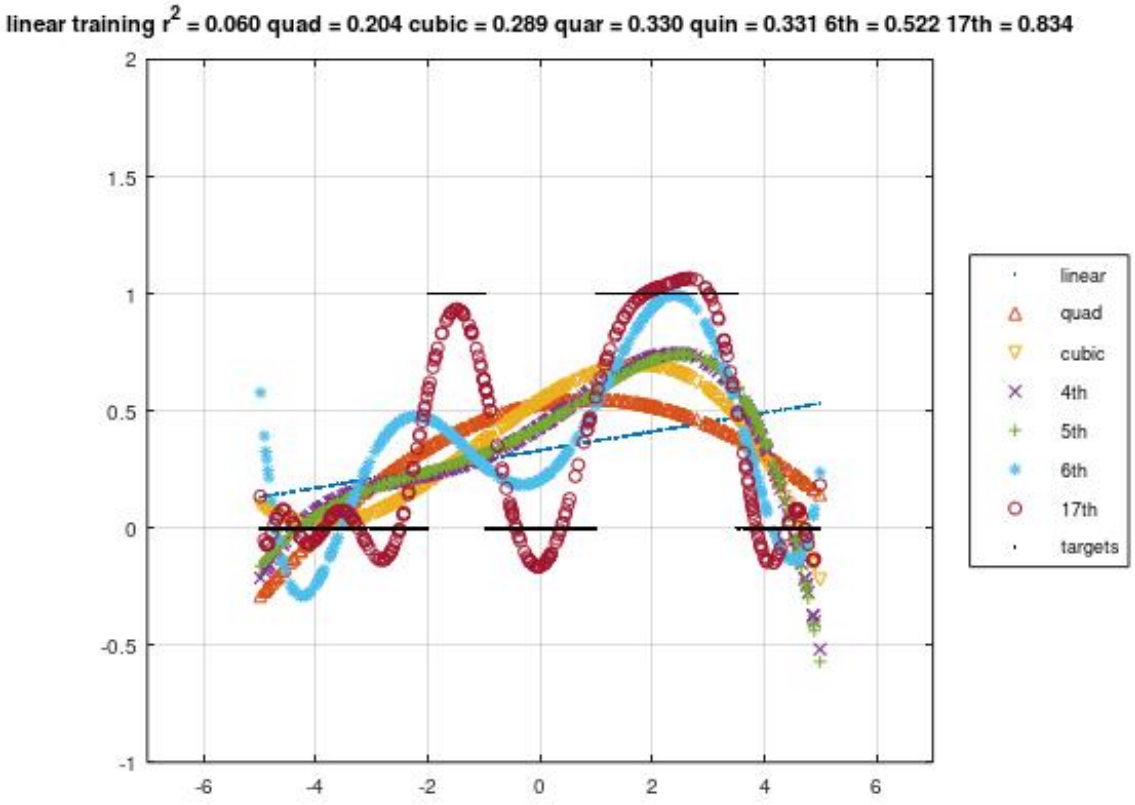

Figure A.1. Polynomials approximating sharply defined islands.

Let us now try with some neural networks. The smooth curve of the neuron activation functions (as specified here; other specifications are possible) means that even though they are in general shaped like the sudden transition, the curve and asymptotic tails cannot perfectly reproduce the constant portions. Furthermore, their asymptotic nature means that we can easily be in a situation where at least some of the optimal parameters would be infinite, but with a finite ratio. Still, we can find parameters that get us arbitrarily close. Hence, even in the best of circumstances, an iterative solver will never really finish.

The islands example with a single neuron (which can only accommodate one change in value and thus is underspecified, much like the linear model) is shown in Figure A.2.

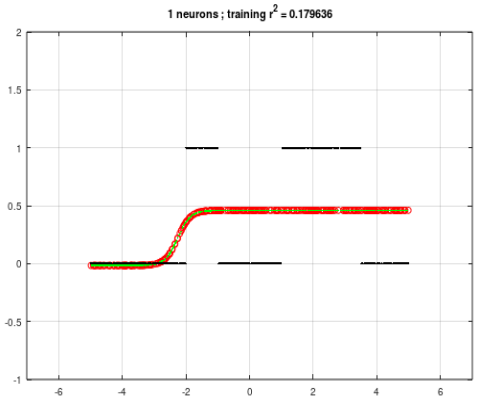

Figure A.2. A neural network with 1 hidden neuron approximating sharply defined islands.

Notice that the near-optimal choice is to closely model the zeros on the left side (since overall, there are more zeros than ones and the block of zeros on the left side is bigger than the block on the right side) and then split the difference on the rest. However, the transition zone just before -2 can be made more and more steep, but will never get to truly vertical. The iterative process that is refining the parameter estimates shows this as well. This single input, single neuron, single output model has four parameters: a constant and coefficient for the input value, and a constant and coefficient to transform the single neuron activation value into the output value. The evolution of these parameter values during part of the estimation process can be seen here:

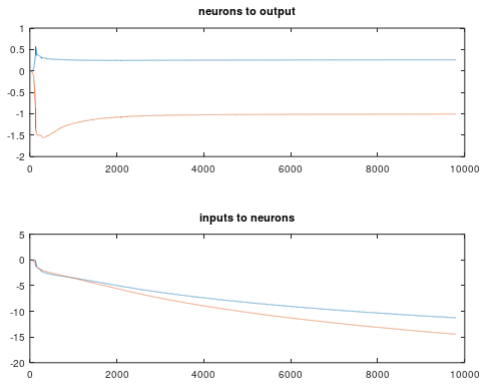

Figure A.3. Evolution of the estimated parameters for a neural network with 1 hidden neuron showing about 10,000 iterations of the optimization process.

At this point, the parameters determining the output are essentially fixed: simply taking what the neuron gives it and repeating it with some slight variation. Indeed, since there is a single neuron, those

parameters could be fixed with constant = 0 and coefficient = 1 (with more neurons, that will not be the case). But, the parameters going from the input to the neuron are moving together, getting larger and larger (though not simply proportionally).

A little thought reveals that each bump will require two neurons: one to get up the hill and one to get back down. Since there are two humps, four neurons should suffice. With the benefit of being able to see the entire problem easily and understanding the structure of neural networks we can rig the parameters and try to move directly to a good approximation like this one:

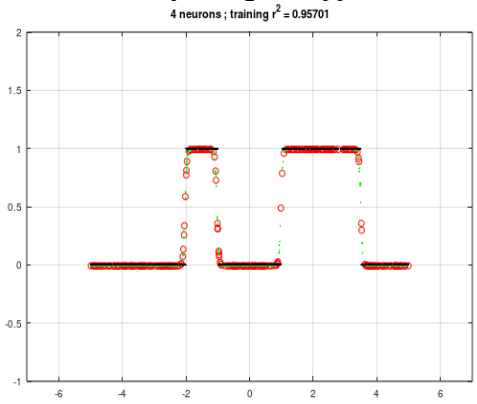

Figure A.4. A neural network with 4 hidden neurons approximating sharply defined islands. The parameters have been chosen directly based on the known structure of the data and the properties of the hidden neurons.

Taking an inspired seed like that and doing some optimization, we can find even better fits like that shown in Figure A.5.; that is, we are using the solver, but starting from the set of parameters leading to Figure A.4. But, be aware that at this point that the details of the computation and rounding errors in the actual computer (which only has so many decimal places to work with) can start to matter along with how the data are distributed in the domain. Remember, the ideal model would have infinite slope at the transitions which cannot be accommodated with real numbers, let alone the discrete mathematics underlying computers.

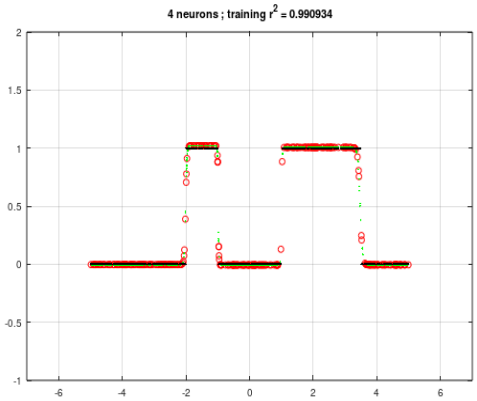

Figure A.5. A neural network with 4 hidden neurons approximating sharply defined islands. A refined version of the model in Figure A.4.

Based on the structure of the model, we can recover the edges of the islands from the estimated parameters. They come out to be -2.00852, -0.98403, 1.00447, 3.47545 which are pretty close to the "true" values.

What has just been described is the nearly ideal situation where we know exactly what we are looking for. But, when we start from scratch and search for good parameters, the (imperfect) optimizer will not necessarily find the most clean set of parameters; or at least, not very quickly. We will likely end up choosing a model that is more complicated than absolutely necessary. But, one that can actually have decent parameters found fairly easily. As we increase the number of neurons, the original pattern is more faithfully reproduced, but eventually, there can be so many neurons that it is difficult to estimate the parameters. In practice, this can mean that overly complicated models can have worse performance because of our inability to find a good set of parameter values. What follows in Figure A.6 to A.14 are examples of the kind of shape and performance obtained when exerting a similar level of reasonable effort to estimate the different specifications.

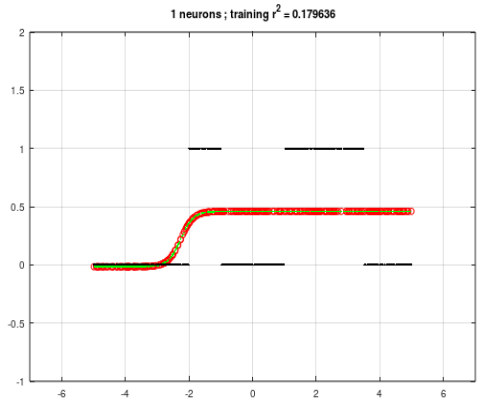

Figure A.6. A neural network with 1 hidden neuron approximating sharply defined islands.

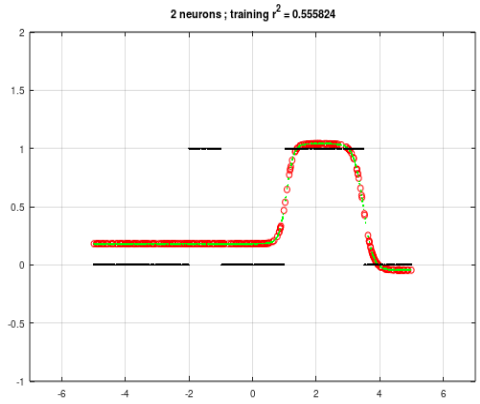

Figure A.7. A neural network with 2 hidden neurons approximating sharply defined islands.

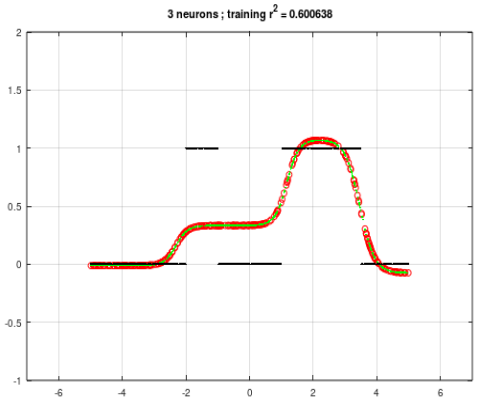

Figure A.8. A neural network with 3 hidden neurons approximating sharply defined islands.

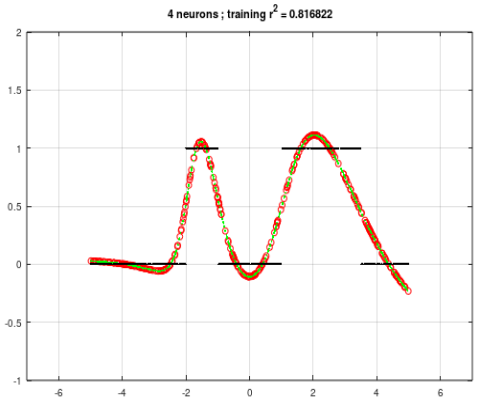

Figure A.9. A neural network with 4 hidden neurons approximating sharply defined islands. Based on parameters starting from scratch and with a moderate amount of "training". That is, the amount of time and attention spent running the optimizer is fairly short. If the process is continued, we obtain a better fit like in Figure A.10.

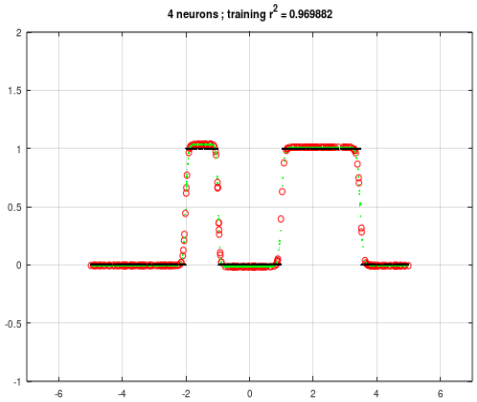

Figure A.10. A neural network with 4 hidden neuron approximating sharply defined islands. Based on parameters starting from scratch and with a great deal of patience in the training process. This particular set of parameters required much more time to be spent optimizing/estimating the parameters than what was expended on the other figures in this appendix.

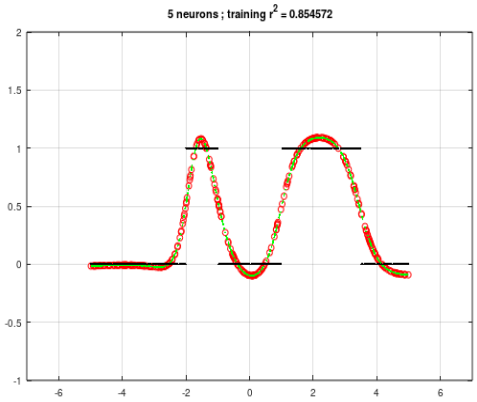

Figure A.11. A neural network with 5 hidden neurons approximating sharply defined islands.

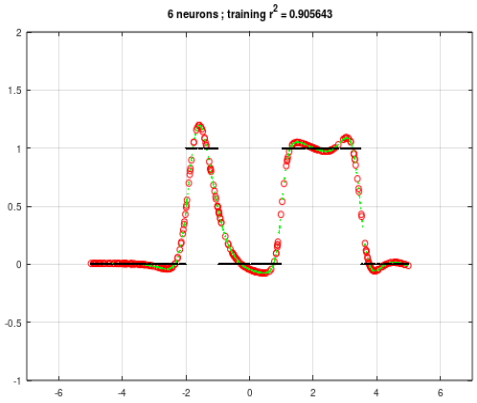

Figure A.12. A neural network with 6 hidden neurons approximating sharply defined islands.

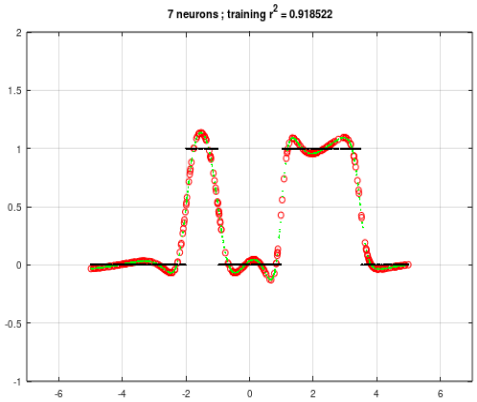

Figure A.6. A neural network with 7 hidden neurons approximating sharply defined islands.

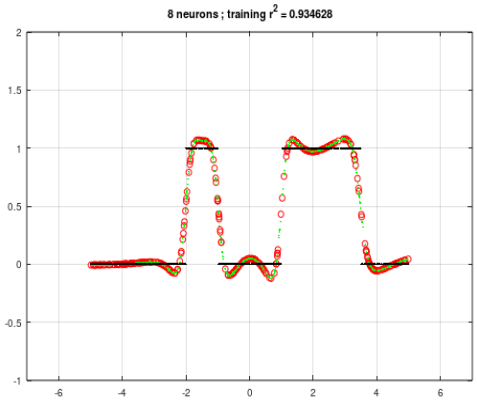

Figure A.13. A neural network with 8 hidden neurons approximating sharply defined islands.

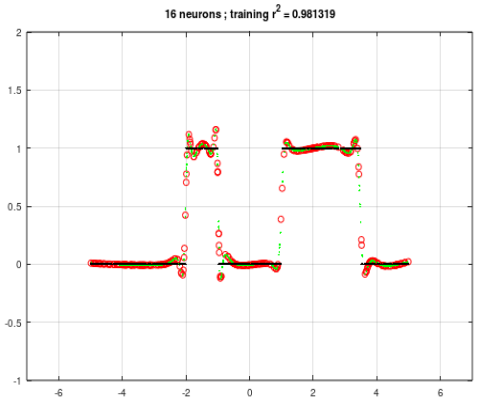

Figure A.14. A neural network with 16 hidden neurons approximating sharply defined islands.

The relationship between the number of parameters and realistic model performance is summarized in the Table A.1 and Figure A.15. The polynomial specifications struggle to reproduce the sharp shifts, but the neural networks catch the basic pattern once four neurons or more are used. Using more neurons (and hence parameters leading to greater flexibility) provides a shortcut to a reasonably well-fitting model even though it may be more complicated than is actually necessary. Notice how the carefully estimated four neuron specification achieves a 0.991 R-squared but the moderate effort version of the same specification is only 0.817 while the moderate effort 16-neuron specification comes in at 0.981.

Table A.1. Comparison of performance between different model specifications for sharply defined islands.

Polynomial models	**number of parameters**	**R-squared**
linear (1st order; fully optimized)	2	0.06
quadratic (2nd order)	3	0.204
cubic (3rd order)	4	0.289
4th	5	0.33
5th	6	0.331
6th	7	0.522
7th	8	0.611
8th	9	0.964
9th	10	0.719
10th	11	0.752
17th (fully optimized)	18	0.834

Neural network models	number of parameters	R-squared
NN 4 neurons, parameters manually chosen, refined with careful optimization	13	0.991
NN 1 neuron, parameters realistically estimated	4	0.180
NN 2 neurons	7	0.556
NN3	10	0.601
NN4	13	0.817
NN5	16	0.855
NN6	19	0.906
NN7	22	0.919
NN8	25	0.935
NN16	49	0.981

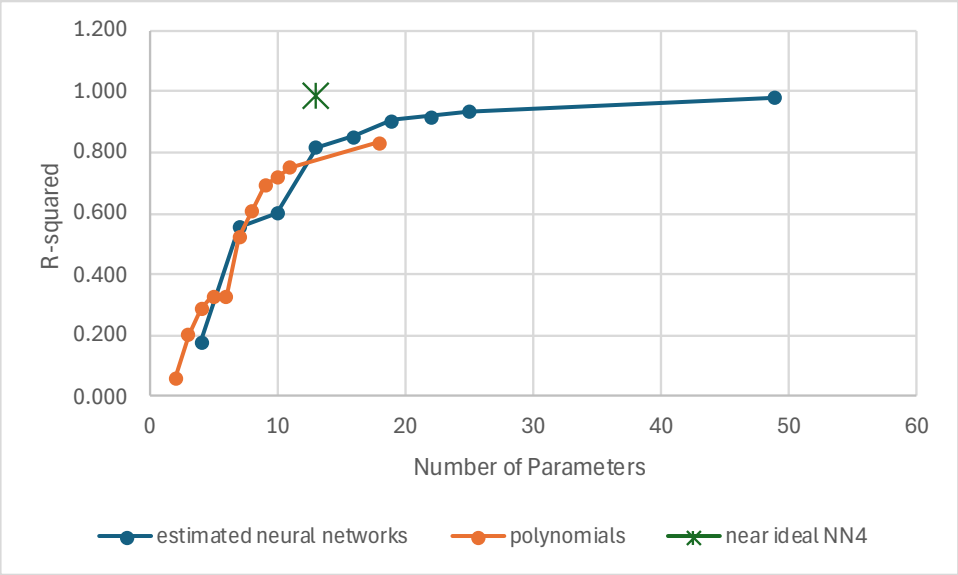

Figure A.15. Graph of comparison of performance between different model specifications for sharply defined islands.

Neural networks have a well-known ability to make very good use of a small number of parameters. In this particular case (which is nearly ideal for the neural networks), we can achieve essentially a perfect fit with a 4 neuron network having 13 parameters while a 17th order polynomial with 18 parameters does not fare nearly as well. This parsimony becomes even more important when there are multiple explanatory variables.

To be fair, we should also look at a case that is nearly ideal for polynomials: a couple periods of a sine wave. In this case, the polynomials perform better for the number of parameters involved. The progression of performance is as shown in Table A.2 and Figure A.23. While the neural networks are able to do a decent job reproducing the pattern (Figures A.17 to A.22), this is where polynomials shine: they reproduce the shape more closely (Figure A.16) with fewer parameters than the neural network.

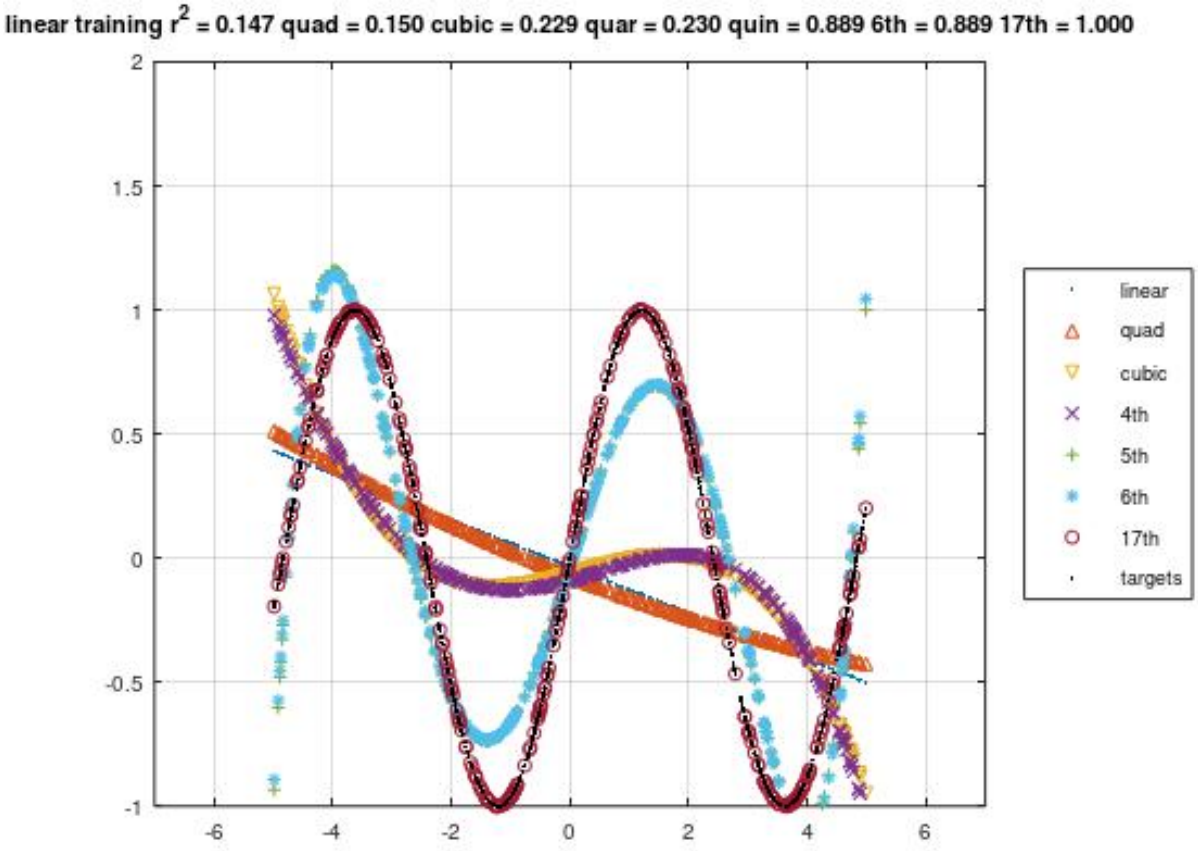

Figure A.16. Polynomial approximations of a portion of a sine wave.

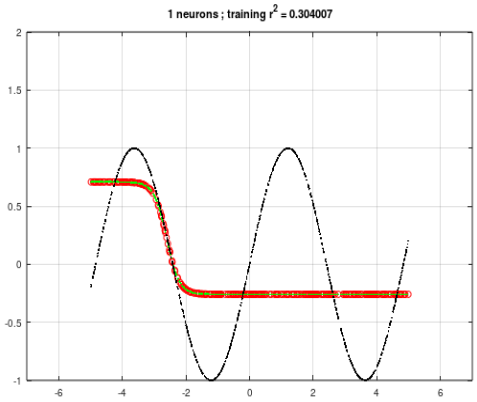

Figure A.17. A neural network with 1 hidden neuron approximating a portion of a sine wave.

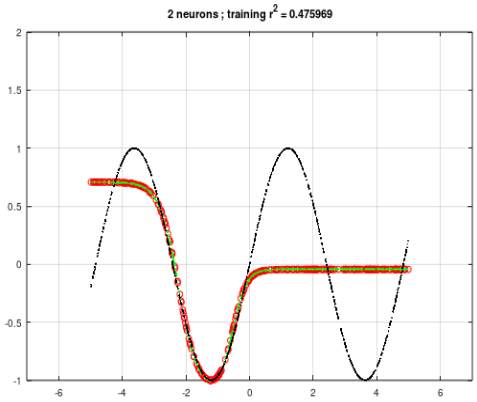

Figure A.18. A neural network with 2 hidden neurons approximating a portion of a sine wave.

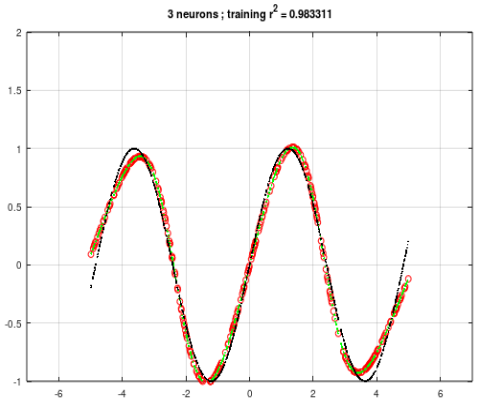

Figure A.19. A neural network with 3 hidden neurons approximating a portion of a sine wave.

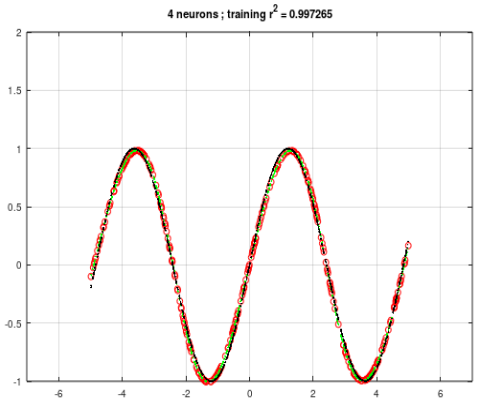

Figure A.20. A neural network with 4 hidden neurons approximating a portion of a sine wave.

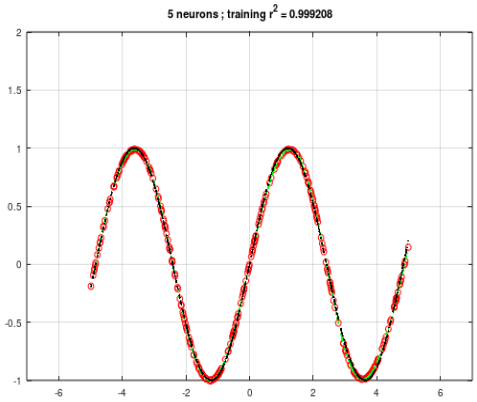

Figure A.21. A neural network with 5 hidden neurons approximating a portion of a sine wave.

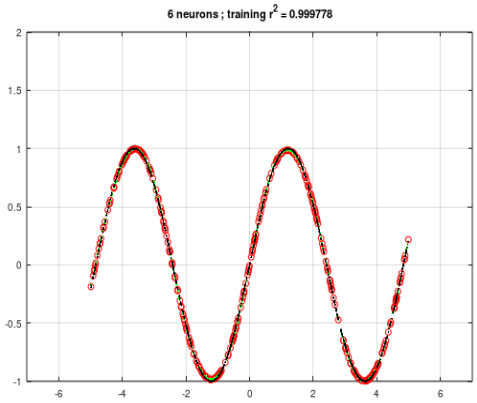

Figure A.22. A neural network with 6 hidden neurons approximating a portion of a sine wave.

Table A.2. Comparison of performance between different model specifications for a portion of a sine wave.

Model description	**number of parameters**	**R-squared**
linear (1st order; fully optimized)	2	0.147
quadratic (2nd order)	3	0.15
cubic (3rd order)	4	0.229
4th	5	0.23
5th	6	0.889
6th	7	0.889
7th	8	0.995
8th	9	0.995
9th	10	1.000
10th	11	1.000
17th (fully optimized)	18	1.000
NN 1 neuron, parameters realistically estimated	4	0.304
NN 2 neurons	7	0.476
NN3	10	0.983
NN4	13	0.997
NN5	16	0.999
NN6	19	1.000

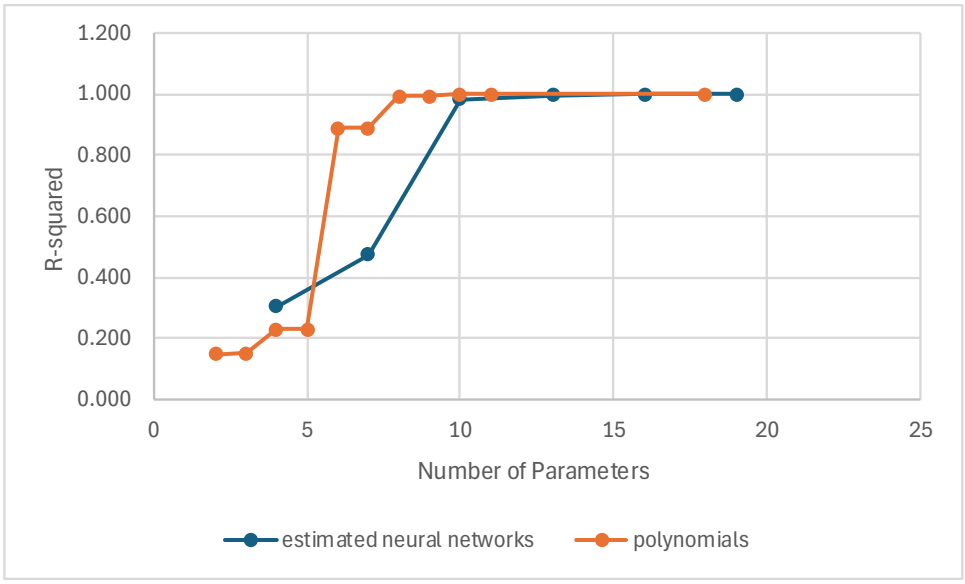

Figure A.23. Graph of comparison of performance between different model specifications for a portion of a sine wave.

Appendix B: Example descriptive statistics for a crop yield dataset

Here, we present descriptive statistics for the case of rainfed sorghum to illustrate the reason for transformations of the values. To make it easier for the numerical optimizer, the explanatory variables are recentered and rescaled to make each one used by the model have mean zero and unitary variance. This works well for temperature, but the awkward distribution of rainfall means that a monotonic transformation is helpful.

The maximum temperature values are simply recentered and rescaled because the extreme values are not very extreme.

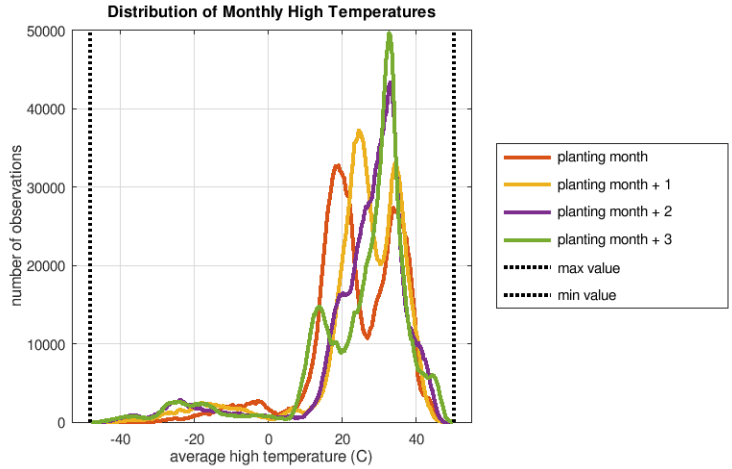

Figure B.1. Histogram of original monthly high temperatures.

Rainfall has a very stretched out distribution with many zero and near zero values along with a very long tail in the wet direction. These extreme values make it difficult for the optimizer to do its work.

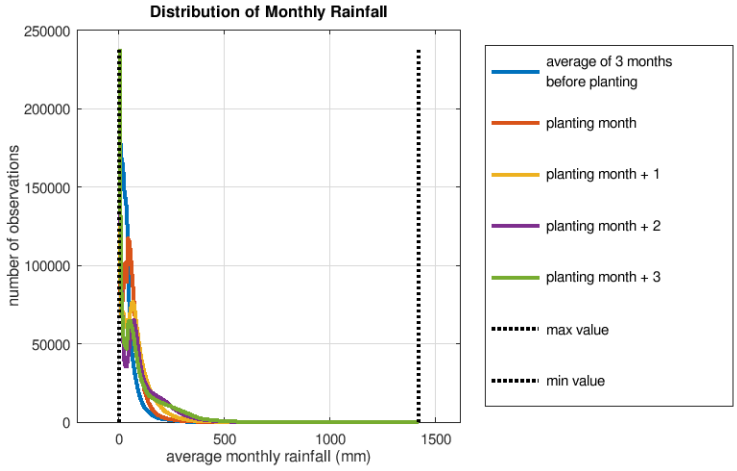

Figure B.2. Histogram of original monthly rainfall amounts.

To make the values more manageable by the optimizer, we use the square root transformation to bring the very high values closer in.

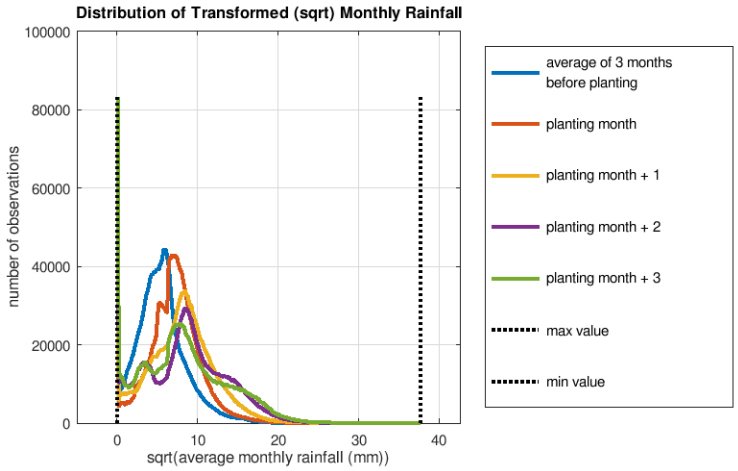

Figure B.3. Histogram of square-root transformed monthly rainfall amounts.

The descriptive statistics for these data are listed in Table B.1. For simplicity, these are the unweighted values counting each pixel as equally important.

Table B.1. Unweighted descriptive statistics for rainfed sorghum dataset

Variable	Minimum	Maximum	Median	Mean	Standard Deviation
Average daily high temperature (C)					
planting month	-37.8	49.2	22.9	23.4	12.3
planting month + 1	-43.0	48.2	26.7	24.9	13.7
planting month + 2	-46.3	48.4	29.0	25.1	15.3
planting month + 3	-48.0	50.1	29.4	24.2	16.0
Monthly rainfall (mm)					
Average over the 3 months prior to planting	0.0	643.1	28.8	40.3	44.0
planting month	0.0	1161.9	51.2	61.4	52.1
planting month + 1	0.0	1389.3	66.3	81.0	72.3
planting month + 2	0.0	1418.0	73.2	98.0	96.3
planting month + 3	0.0	1387.9	63.6	98.6	107.2
Square root of monthly rainfall					
Average over the 3 months prior to planting	0.0	25.4	5.4	5.6	3.0
planting month	0.0	34.1	7.2	7.2	3.2
planting month + 1	0.0	37.3	8.1	8.1	3.9
planting month + 2	0.0	37.7	8.6	8.6	4.9
planting month + 3	0.0	37.3	8.0	8.4	5.3

Appendix C: Model performance by crop and specification

The kind of neural network specification used in this paper is based on explanatory variables, "hidden neurons", and the "output neuron". We use a fully connected network where all of the hidden neurons make use of all of the explanatory variables. And, of course, the output neuron makes use of the values determined by each of the hidden neurons. While it is possible to eliminate or zero out some of these parameters, we did not do so.

The number of parameters in our neural networks depends on the number of explanatory variables and the number of hidden neurons. In our case, the only change in the number of explanatory variables is whether or not fertilizer is relevant since soybeans and groundnuts do not make use of nitrogen fertilizer in our models.

As previously noted, the total number of parameters is the number of parameters connecting the inputs to the "hidden" neurons (along with their constants) plus the number of parameters connecting the "hidden" neurons to the output (along with a different constant).

That is the total number of parameters = $H \times (k + 1) + (H + 1) = Hk + H + H + 1 = H \times (k + 2) + 1$

The following graphs show how the performance of the estimated models changes as more and more hidden neurons, and thus parameters are added.

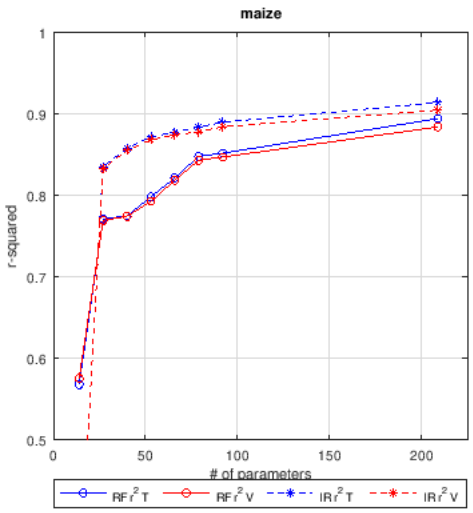

Figure C.1. Model performance by number of parameters for maize.

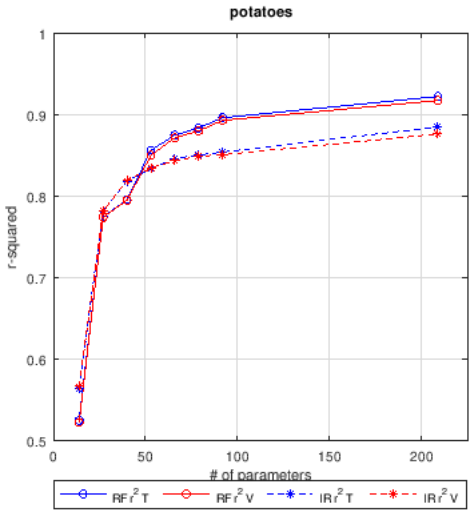

Figure C.2. Model performance by number of parameters for potatoes.

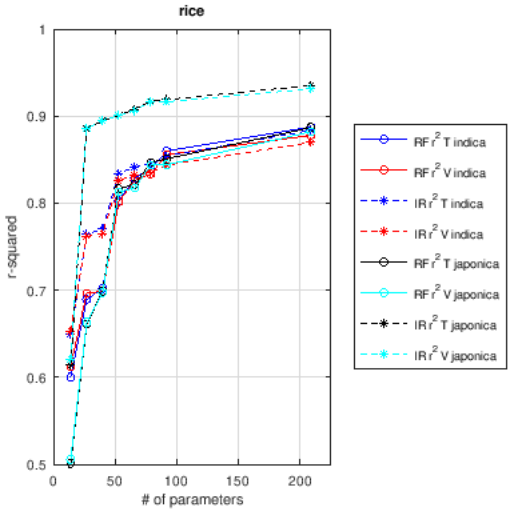

Figure C.3. Model performance by number of parameters for rice.

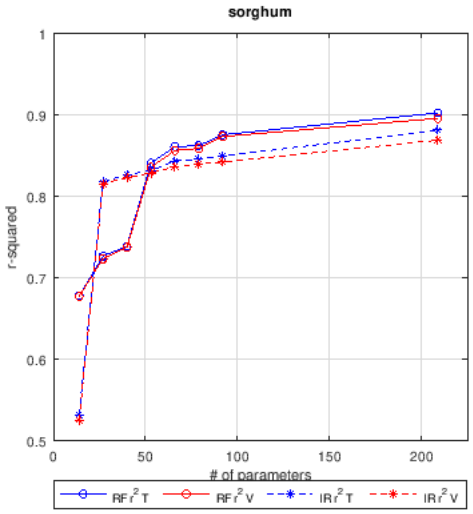

Figure C.4. Model performance by number of parameters for sorghum.

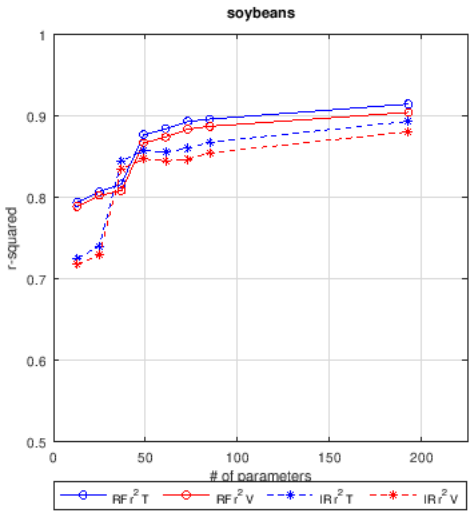

Figure C.5. Model performance by number of parameters for soybeans.

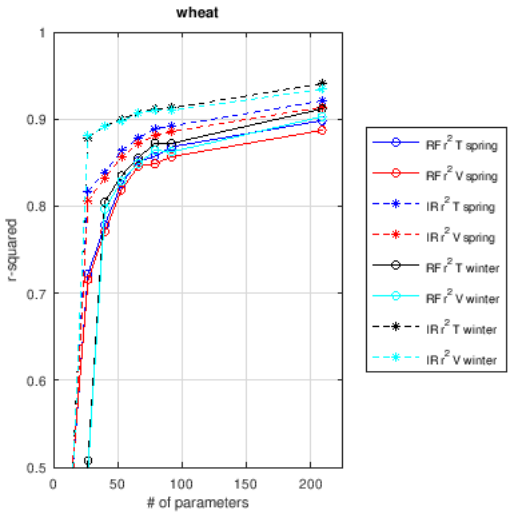

Figure C.6. Model performance by number of parameters for wheat.

REFERENCES

Hoogenboom, G., C.H. Porter, K.J. Boote, V. Shelia, P.W. Wilkens, U. Singh, J.W. White, S. Asseng, J.I. Lizaso, L.P. Moreno, W. Pavan, R. Ogoshi, L.A. Hunt, G.Y. Tsuji, and J.W. Jones. 2019. The DSSAT crop modeling ecosystem. In: p.173-216 [K.J. Boote, editor] Advances in Crop Modeling for a Sustainable Agriculture. Burleigh Dodds Science Publishing, Cambridge, United Kingdom https://dx.doi.org/10.19103/AS.2019.0061.10.

Hoogenboom, G., C.H. Porter, V. Shelia, K.J. Boote, U. Singh, W. Pavan, F.A.A. Oliveira, L.P. Moreno-Cadena, T.B. Ferreira, J.W. White, J.I. Lizaso, D.N.L. Pequeno, B.A. Kimball, P.D. Alderman, K.R. Thorp, S.V. Cuadra, M.S. Vianna, F.J. Villalobos, W.D. Batchelor, S. Asseng, M.R. Jones, A. Hopf, H.B. Dias, A. Jintrawet, R. Jaikla, E. Memic, L.A. Hunt, and J.W. Jones. 2024. Decision Support System for Agrotechnology Transfer (DSSAT) Version 4.8.5 (www.DSSAT.net). DSSAT Foundation, Gainesville, Florida, USA.

International Food Policy Research Institute (IFPRI); International Institute for Applied Systems Analysis (IIASA). 2016. Global Spatially-Disaggregated Crop Production Statistics Data for 2005 Version 3.2. Harvard Dataverse, V9. https://doi.org/10.7910/DVN/DHXBJX

Jones, J.W., G. Hoogenboom, C.H. Porter, K.J. Boote, W.D. Batchelor, L.A. Hunt, P.W. Wilkens, U. Singh, A.J. Gijsman, and J.T. Ritchie. 2003. The DSSAT cropping system model. European Journal of Agronomy 18:235-265. https://doi.org/10.1016/S1161-0301(02)00107-7

Stefan Lange, Dánnell Quesada-Chacón, Matthias Büchner. 2024. Secondary ISIMIP3b bias-adjusted atmospheric climate input data (v1.5). ISIMIP Repository. https://doi.org/10.48364/ISIMIP.581124.5

Robertson, Richard D. 2017. Mink: Details of a global gridded crop modeling system. Washington, DC: International Food Policy Research Institute (IFPRI). https://hdl.handle.net/10568/148024

Sheffield J, Goteti G, Wood EF. 2006. Development of a 50-yr high-resolution global dataset of meteorological forcings for land surface modeling. J Climate 19:3088–3111 https://hydrology.soton.ac.uk/data/pgf/

INTERNATIONAL FOOD POLICY RESEARCH INSTITUTE
www.ifpri.org

IFPRI HEADQUARTERS
1201 Eye Street, NW
Washington, DC 20005 USA
Tel.: +1-202-862-5600
Fax: +1-202-862-5606
Email: ifpri@cgiar.org