Predictive Design Technology

Norman Packard
ProtoLife Inc.

Predictive Design Technology (PDT) is an implementation of advanced data-science algorithms for experimental discovery and optimization, now made available by ProtoLife via a web interface. This white paper aims to explain the basics of how PDT works, including some details of PDT's machine-learning modeling tools not directly apparent through the web interface. The target audience is comprised of potential users of PDT, including those who do not have deep expertise in data science or machine learning.

INTRODUCTION

The world is *complex*: science has made fantastic progress understanding the world well enough to produce a constant stream of technological advance, but many parts of the world resist such powerful understanding because non-linear interactions between system components prevent formulation of laws that enable traditional engineering. Living systems are a canonical example of such complex systems, but many examples of non-living complex systems exist as well, e.g., complex materials and complex chemicals.

Complex systems cannot be engineered using conventional top-down design approaches. Instead, exploratory experimentation is often the only way to make engineering breakthroughs. This can be problematic when the needed experiments are *expensive*. However, in the past decades, experimental technology has developed to the level that many complex experiments may be performed simultaneously. Large sets of experiments executed in parallel are often referred to as *high-throughput experiments* (HTE). The technological advances enabling HTE open opportunities for unprecedented experimental exploration, but also reveal new difficult challenges.

The essential challenge in HTE may be cast as *design of experiments* (DOE) problem: after one set (say, a *generation*) of experiments is performed, what experiments should be performed in the following generation, in order to maximize the probability of obtaining better, and ideally *optimal*, experimental results? This problem is particularly difficult when an experiment has several parameters that interact with each other in unknown, non-linear ways, and when data is limited because of high experiment cost. *Predictive Design Technology* (PDT) offers a practical and effective approach to this problem, now easily accessible through an easy-to-use web interface.

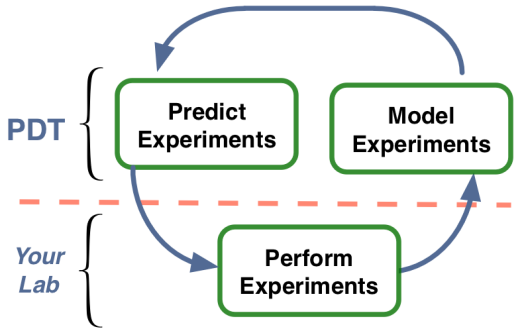

FIG. 1. PDT traverses the experimental loop by building predictive models from experimental data, and delivering new experiments that are most likely to give improved results.

PDT In a Nutshell

When data is limited because of experiment cost, PDT will effectively leverage the data you gather, by iteratively building a model of your experimental process and using this model to predict the experimental response of millions of new experiments. At every generation (iteration), a new model is built using data from all previous generations. With each generation, the model is further and further refined with new accumulated data, and its predictive performance improves. This, in turn, leads to the discovery of increasingly better experiments. Figure 1 schematically illustrates this iterative procedure, which we call the *experimental loop*.

EXPERIMENTAL SPACE AND RESPONSE SURFACE

Let's start our discussion by introducing a few key concepts, which we will need to be able to represent experimental results as data points, ready to be used as input for the PDT model.

The *experimental space* is the set of all experiments that are potentially accessible. An experiment is typi-

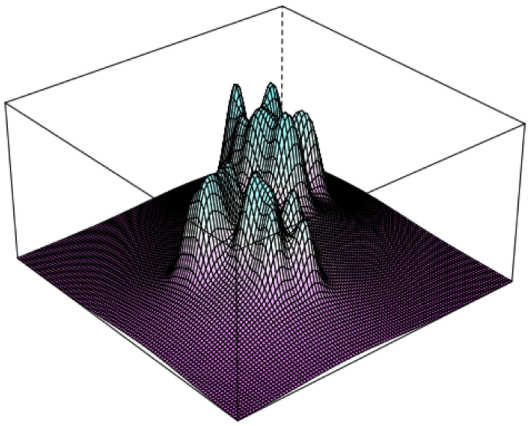

FIG. 2. A schematic representation of a complex response surface on a two-dimensional experimental space.

cally specified by a set of experimental parameters, e.g., concentrations of different constituent chemicals, temperatures, incubation times, etc. For experimental exploration, various of these parameters will be fixed and others allowed to vary. The parameters that are allowed to vary become the coordinates of the experimental space E. Every accessible experiment corresponds to a point in E.

Experimental measurements typically yield a numerical value for each experiment, so we may think of the set of experimental results for all experiments in the space as a real-valued function on E. This function is also known as *response surface*. We will consider the case that *higher* experimental measurements are desirable, so the highest peak of the response surface is optimal. The response surface is complex if it has many local maxima; a complex response surface is indicated schematically in Figure 2. We may consider each coordinate of E to be discretized, i.e. taking on a discrete, finite set of (usually numerical) values. In general, the distance between any two consecutive values in that set should be large enough to be experimentally realizable, but small enough so that the experimental response is expected to vary somewhat smoothly in between those values.

A set of experimental parameters and of corresponding possible values is what we refer to as an *experimental space definition* (ESD). A simple example of ESD is shown in Table I, where each experiment is defined by 7 experimental parameters (so the dimension of E is $d_{\rm exp} = 7$), with each parameter able to take 2 to 5 different discrete values.

The *size* of the experimental space is the number of possible experiments determined by its ESD. A particular experiment is specified by a choice of values for each experimental parameter. Assuming no constraints on the choice of such values, the total number of possible experiments is given by

TABLE I. A simple example of experimental space definition. There are 7 different experimental parameters, so the dimension of the experimental space is $d_{\rm exp} = 7$. The size of the experimental space is 7200 (cf. explanation in the text).

Param. Name	**Value 1**	**Value 2**	**Value 3**	**Value 4**	**Value 5**
QuenchYesNo	0	1			
StepNum	1	2	3		
Duration	60	120	180		
Buffer	7000	7900	8850	10550	
Catalyzer	0	0.85	2	3.2	
AmphiLogRatio	-1.1	-0.22	1.23	1.89	3.2
DNAQuant	100	180	260	311	400

$$n_{\rm exp} = \prod_{i=1}^{d_{\rm exp}} n_i, \tag{1}$$

where n_i is the number of values that experimental parameter i can take. The ESD in Table I, for example, has $n_1 = 2$, $n_2 = 3$, $n_3 = 3$, $n_4 = 4$, $n_5 = 4$, $n_6 = 5$, and $n_7 = 5$, yielding a size of $n_{\rm exp} = 7200$. The size of the experimental size typically grows *exponentially* with the number of experimental parameters: for example, adding 1, 2, 3, ..., extra parameters with 5 values each to the ESD in Table I would result in a space with 36000, 180000, 900000, ..., experiments, respectively.

For an experimental campaign, each complete generation will consist of a certain number N_g of experiments, with each experiment given by a pair (x, r), where x is a point in E (a vector of $d_{\rm exp}$ values), and r is the measured experimental response value for that experiment. E.g., if the experimental platform yields a different experiment for each well in a 96-well plate, the number of experiments per generation might be $N_g = 96$. Each generation of experimental results $\{(x_i, r_i)\}$, with $i = 1, 2, \ldots, N_g$, would then consist of a table of numbers, with N_g rows and $d_{\rm exp} + 1$ columns.

THE PDT MODEL

Now that the concepts of experimental space and experimental surface are well defined, we can focus the discussion on the core component of the experimental loop: the PDT model.

PDT models map points in the experimental space E to values on the real line of experimental responses. A model M_β will typically have various parameters β that are *fit* to the experimental data. A very simple example is a linear model for the case $d_{\rm exp} = 1$, which may be fit by linear regression. In this case, β consists of two numbers, the slope and the intercept, both of which are obtained by the fit. A more advanced case might be a neural network model to fit data from the ESD specified

in Table I. The neural network would have $d_{\text{exp}} = 7$ inputs, and one output to predict the response. Parameters β would be all the weights and biases of the neural network, and their number would depend on the network's architecture.

After m generations, model M_β will be *trained* (fit) on data set $D = \{(x_i, r_i)\}$, which contains a total of $N_d = mN_g$ experiment-response pairs (x_i, r_i). The β vector of this model will typically be the minimizer $\beta^*(D)$ of a fitting error function, such as the mean squared error:

$$\beta^*(D) = \arg\min_\beta MSE(M_\beta, D) = \frac{1}{N_d}\sum_{i=1}^{N_d}(\hat{r}_i - r_i)^2\,, \tag{2}$$

where $\hat{r}_i$ is the model's fit for response r_i.

Machine-learning models often include another class of parameters, called *hyper-parameters*, which control such factors as the complexity of the model and how the model *learns* from the data. Hyper-parameters are usually fixed prior to the fitting of parameters β, and, unlike β, are not fit directly on D, but selected using different strategies. We will denote the hyper-parameters of a PDT model by α, and the fitted model with hyper-parameters α by M_{α,β^*}. For a neural network, α includes:

- Network-architecture parameters, such as the number of hidden layers, and the number of hidden nodes, which in turn translate into the length of the parameter vector β.
- Weight decay, a *regularization* term in the training algorithm (usually taken to be the *back-propagation* algorithm), which penalizes large weight values, with the goal of making the model smoother.
- Number of training iterations, each of which consisting of an incremental change made to β by the back-propagation algorithm. A larger the number of iterations usually results in a more complex model.

Model Selection

Careful hyper-parameter tuning is crucial to the model's *generalization performance*, that is, its ability to accurately predict unobserved data (also known as *testing data* or *out-of-sample data*). An overly simple model, for example, will tend to *underfit* the training data, that is, miss important patterns (for example, interactions between experimental parameters) that determine the response of an experiment in a repeatable way. An overly complex model, on the other hand, will tend to *overfit* the training data, that is, learn details that are specific only to that data, such as random measurement noise, and that do *not* determine the response an experiment in a repeatable way. In either case, the model will not be able to provide reliable predictions for the response of experiments that have not yet been performed.

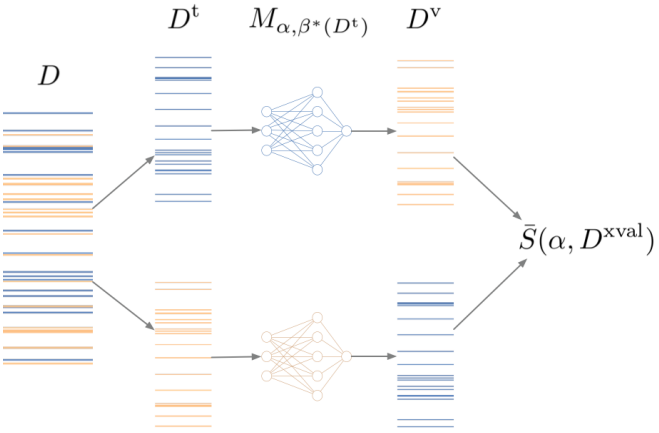

FIG. 3. Illustration of cross-validation. The overall data set is partitioned into two subsets. The blue subset is used for training one model with hyper-parameters α, which is then validated on the orange subset. Conversely, the orange subset is used for training another model with hyper-parameters α, which is then validated on the blue subset. The average validation performance of the two models gives the score assigned to α.

Cross-Validation

PDT estimates the generalization performance of a model through a process of *cross-validation*. The basic idea is to obtain an estimate of how $M_{\alpha,\beta}$ behaves out-of-sample by splitting the available data set D into two disjoint subsets D^{t} and D^{v}:

$$D^{\text{t}} \cup D^{\text{v}} = D,\ D^{\text{t}} \cap D^{\text{v}} = \{\emptyset\} \tag{3}$$

where 't' stands for "training" and 'v' stands for "validation". A model with hyper-parameters α is then trained on D^{t} and its performance is measured on D^{v}, for example by calculating the following score:

$$S(\alpha, D^{\text{t}}, D^{\text{v}}) = MSE\left(M_{\alpha,\beta^*(D^{\text{t}})}, D^{\text{v}}\right). \tag{4}$$

Note how the equation denotes the model's dependence on hyper-parameters α and on parameters β^*, which in turn depend on training data D^{t}, and the error function dependence on validation data D^{v}.

The score computed above for a model with hyper-parameters α is contingent on a particular $\{D^{\text{t}} \cup D^{\text{v}}\}$ split. If we try another split, we would likely obtain a different score. Repeating the procedure for many, say, M cross-validation splits provides us with an empirical distribution of scores. We can finally extract from this distribution a representative score to assign to the model

with hyper-parameters α, for example:

$$\bar{S}(\alpha, D^{\text{xval}}) = \frac{1}{M}\sum_{j=1}^{M} S\left(\alpha, D_j^{\text{t}}, D_j^{\text{v}}\right), \tag{5}$$

where $D^{\text{xval}} = \left\{\{D_j^{\text{t}}, D_j^{\text{v}}\},\ j = 1, 2, \ldots, M\right\}$ is a collection of cross-validation splits. A simple illustration of cross-validation is shown in Figure 3.

Hyper-parameter optimization

To avoid training models that underfit or overfit the experimental data, we must step back to consider a model's hyper-parameters α as well as its parameters β. Given a family of models M_{α,β^*} with different hyper-parameters α and corresponding fitted parameters β^*, we want to identify the models within this family that are more likely *not* to incur underfitting or overfitting, and that are consequently more likely to yield accurate out-of-sample predictions of experimental response. This will entail finding the minimizer α^* of scoring function (5)

$$\alpha^* = \arg\min_{\alpha} \bar{S}(\alpha, D^{\text{xval}}), \tag{6}$$

where we are looking for a minimum since (5) increases with the estimated generalization error.

Optimal hyper-parameters α^*, and the corresponding optimal model complexity, will typically depend on the roughness of the response surface (a more irregular response surface will often result in a more complex model), but also on the amount of available experimental data (less data will often result in a simpler model).

Experiment Prediction

Once α^* is found, the corresponding model $M_{\alpha^*,\beta*(D)}$ can be finally trained on *all* of the available data D, and this model can be used to select the best generation of experiments to perform next.

The value $\hat{r}_i = M_{\alpha^*,\beta*(D)}(x_i)$, i.e., the prediction of model $M_{\alpha^*,\beta*(D)}$ for experiment x_i, may be regarded as a computational simulation or "virtual approximation" of the actual, physical process that defines experiment x_i. Since it is virtual, this approximation can be queried cheaply, quickly, and repeatedly for different x_i's, to find the best predicted next generation of experiments. If the experimental space is small enough (say, no more than a few million individual experiments), the response of each and every experiment in that space can be predicted, and the best experiments can be selected directly (for example, identifying the N_g experiments x_i with the highest $\hat{r}_i$ values). For larger experimental spaces, PDT replaces exhaustive prediction with advanced *global optimization* methods, aimed at finding the overall maximum or maxima of a potentially complex, multi-peaked predicted response surface.

Experiment Exploration

Broadly speaking, we can expect the predictive power of the PDT model to increase as the experimental information stored in D becomes richer. What makes D informative is not only its size, i.e., the number of experimental observations it contains, but also its *coverage* of the experimental space E. Coverage can be interpreted as the degree with which D contains experimental observations that are representative of all regions of E. To complement experiment prediction, which is aimed at the selection of new experiments in specific, promising regions of E, PDT also incorporates algorithms for experiment *exploration*, aimed at the selection of new experiments located in unexplored regions of E, where it is more likely that experimental observations will produce *surprise*. A simple implementation of experiment exploration may consist of randomly sampling experiments x_i with non-uniform probabilities p_i, proportional to the mean distance d between x_i and each of N_d experiments in D:

$$p_i \sim \frac{1}{N_d}\sum_{x_k \in D} d(x_i, x_k), \tag{7}$$

which will on average select experiments that are "far" (i.e., different) from those already performed in E.

Manually selected experiments (e.g., calibrations, or experiments based on expert-knowledge intuition) can also be incorporated in the experiment exploration approach to further enhance the model's predictive power.

At the very first generation of an experimental campaign, i.e., when no data is yet available, experiment exploration is typically the only criterion that PDT uses for selecting experiments.

Model Evolution

Each iteration of the loop shown in Figure 1 represents a generation, i.e., a collection of experiments selected by prediction and exploration, based on the experimental data collected in all previous generations. As increasing amounts of experimental results are obtained by successive generations, the PDT model becomes increasingly accurate in its prediction of good experiments. The resulting process is illustrated in Figure 4, where we denote by D^g the data set available at generation g, and by $M^g_{\alpha^*,\beta*(D^g))}$ the optimal model trained on D^g.

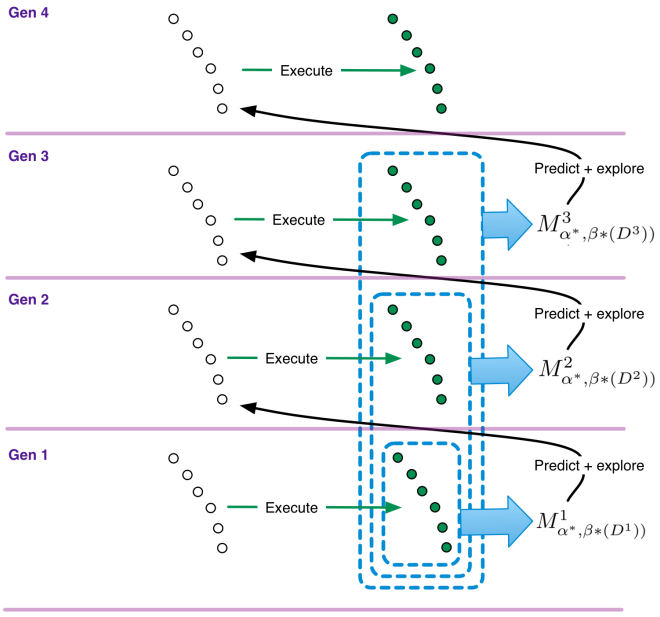

FIG. 4. Illustration of PDT model evolution. White circles represent experiments (points in the experimental space) and green circles represent experimental results (experiments with measured response). Green arrows represent experiment execution (e.g., in the lab). Blue arrows represent the process of model training via cross-validated hyper-parameter optimization. Black arrows represent experiment selection for generation $g+1$, combining predictions from the model trained on all of the data collected up to generation g and corresponding exploration.

Model Classes

In this overview of the PDT model, we have used neural networks as a model class for our illustrations, but most of what we covered holds for a variety of other model classes as well, e.g., Gaussian process regression, support vector regression, Bayesian networks, etc. Each different model class will have its own parameters β and hyper-parameters α, fitting into the discussion above. PDT uses smart algorithms to choose the model class, or combination of model classes, best suited to a specific experimental space and response surface.

WEB INTERFACE TO PDT

A very intuitive and easy-to-use web interface to PDT is available at https://protolife.com. This interface gives you access to a powerful *artificial-intelligence* platform, which automatically and seamlessly deploys models and algorithms for efficient, cost-effective experimental discovery and optimization similar to those described here, *without the need of any expertise in data science, machine learning, programming, or design of experiments on your part.* Signing up for a PDT account is free, and will allow you to explore all of PDT's functionalities on a demo experimental campaign, run on the same ESD given in Table I.