Personalized Nutrition by Prediction of Glycemic Responses

David Zeevi, Tal Korem, Niv Zmora, David Israeli, Daphna Rothschild, Adina Weinberger, Orly Ben-Yacov, Dar Lador, Tali Avnit-Sagi, Maya Lotan-Pompan, Jotham Suez, Jemal Ali Mahdi, Elad Matot, Gal Malka, Noa Kosower, Michal Rein, Gili Zilberman-Schapira, Lenka Dohnalova, Meirav Pevsner-Fischer, Rony Bikovsky, Zamir Halpern, Eran Elinav and Eran Segal

Nomi Hadar, 27.12.16
• The **glycemic response** to a food is the effect that food has on blood sugar (glucose) levels after consumption.
• A **low glycemic food** will release glucose more slowly and steadily, which leads to lower postprandial blood glucose readings.
• A **high glycemic food** causes a more rapid rise in blood glucose levels after meals.
• **PPGRs** = postprandial glycemic responses
• **Blood glucose levels are rapidly increasing in the population,** as evident by the sharp incline in the prevalence of prediabetes.

• **Prediabetes,** characterized by chronically impaired blood glucose responses, is a significant risk factor for type II diabetes.

• **Maintaining normal blood glucose levels is critical** for preventing and controlling diabetes and many other diseases.
• Dietary intake is a central determinant of blood glucose levels.

• In order to achieve normal glucose levels, it is imperative to make **food choices that induce normal postprandial glycemic responses.**

• Despite their importance, **no method exists for predicting PPGRs to food.**
• The current practice is to use the meal carbohydrate content, even though it is a poor predictor of the PPGR.

• Other methods: glycemic index, glycemic load.

• Ascribing a single PPGR to each food assumes that the response is solely an intrinsic property of the consumed food.
• However, few small-scale studies found **high variability in PPGRs of different people to the same food**.

• Factors that may affect interpersonal differences in PPGRs:
  o Genetics.
  o Lifestyle.
  o Insulin sensitivity.
  o Propensity for obesity
  o Gut microbiota (little known).
  o And more.
Goals of study

- To quantitatively measure individualized PPGRs, characterize their variability across people and identify factors associated with this variability.

- Devised a machine learning algorithm that predicts personalized PPGRs.
The researchers continuously monitored glucose levels during an entire week in a cohort of 800 healthy and prediabetic individuals.
Main cohort: 800 healthy and prediabetic individuals

Per person profiling

- Gut microbiome
  - 16S rRNA
  - Metagenomics
- Blood tests
- Questionnaires
  - Food frequency
  - Lifestyle
  - Medical
- Anthropometrics
Main cohort: 800 healthy and prediabetic individuals
Main cohort: 800 healthy and prediabetic individuals

Per person profiling

- Gut microbiome: 16S rRNA, Metagenomics
- Blood tests
- Questionnaires: Food frequency, Lifestyle, Medical
- Anthropometrics

Diary (food, sleep, physical activity)
Using smartphone-adjusted website
5,435 days, 46,896 meals, 9.8M Calories, 2,532 exercises

Continuous glucose monitoring
Using a subcutaneous sensor (iPro2)
130K hours, 1.56M glucose measurements

Standardized meals (50g available carbohydrates)

Glycemic responses

Meal response predictor

Results

Motivation

Background

Computational analysis

Main cohort

800 Participants

PPGR prediction

Validation cohort

100 Participants

Dietary intervention

26 Participants
PPGRs associate with risk factors.
Shown are PPGRs, BMI, HbA1c%, age, and wakeup glucose of all participants, sorted by median standardized meal PPGR (top, red dots).
Correlation of factors with the median PPGRs to standardized meals is shown along with a moving average line.

Moving average line = series of averages of different subsets of the full data set.
Kernel density estimation (KDE) smoothed **histogram of the PPGR to four types of standardized meals** provided to participants. Dashed lines represent histogram modes.

**Kernel density estimation (KDE) =** A technique to estimate the unknown probability distribution of a random variable, based on a sample of points taken from that distribution.
Example of **high interpersonal variability and low intra-personal variability** in the PPGR to bread across four participants (two replicates per participant consumed on two different mornings).
Example of two replicates of the PPGR to two standardized meals (left) / real-life meals (right) for two participants exhibiting reproducible yet **opposite** PPGRs.
So how should we know which food is the best for us in terms of glycemic response?
Prediction of Personalized Postprandial Glycemic Responses
Prediction of Personalized Postprandial Glycemic Responses

Background
Meal response predictor
Results

General scheme
Regression trees
Gradient boosting regression
Partial dependence plots
Prediction of Personalized Postprandial Glycemic Responses

Background

Meal response predictor

Results

General scheme

Regression trees

Gradient boosting regression

Partial dependence plots

![Diagram of meal response predictor](image)
Regression trees - intro

• Decision tree is a predictive model which maps observations about an item (the branches) to conclusions about the item's target value (the leaves).

• **Classification trees:** target variable is *categorical*.

• **Regression trees:** target variable is *continuous.*
CART (Classification And Regression Tree) algorithm

CART is a term to refer to decision tree algorithms that can be used for classification or regression predictive modeling problems.

The main elements of CART are:

1. Rules for splitting data at a node based on the value of one variable.
2. Stopping rules for deciding when a branch is terminal and can be split no more.
3. Finally, a prediction for the target variable in each terminal node.

Breiman et al.
Regression trees - CART algorithm

**outline**

- The tree is built through *binary recursive partitioning*.
- Initially, all records in the training set are allocated into the first two partitions or branches, using every possible binary split on every field.
- The algorithm selects the split that *minimizes the sum of the squared deviations from the mean* in the two separate partitions.
- This splitting rule is then applied to each of the new branches.
- This process continues until each node reaches a user-specified minimum node size and becomes a terminal node.
Regression trees - CART algorithm

• N observations consists of p inputs and a target $y$, that is:
  $$(x_i, y_i) \text{ for } i = 1, 2, \ldots, N \text{ with } x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}).$$

• Suppose first that we have a partition into M regions $R_1, R_2, \ldots, R_M$ and we model the target as a constant $c_m$ in each region:
  $$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

• If our minimization criterion is the sum of square $\Sigma(y_i - f(x_i))^2$ the best $\hat{c}_m$ is just the average of $y_i$ in region $R_m$:
  $$\hat{c}_m = \text{avg}(y_i | x_i \in R_m)$$
Regression trees - CART algorithm

• Finding the best binary partition in terms of minimum sum of squares is generally computationally infeasible.

• Hence we proceed with a greedy algorithm.
Regression trees - CART algorithm

- Starting with all of the data, consider a splitting variable $j$ and split point $s$, and define the pair of half-planes

$$ R_1(j, s) = \{X | X_j \leq s\} \text{ and } R_2(j, s) = \{X | X_j > s\} $$
Regression trees - CART algorithm cont.

• Starting with all of the data, consider a splitting variable $j$ and split point $s$, and define the pair of half-planes
  \[ R_1(j, s) = \{X \mid X_j \leq s\} \text{ and } R_2(j, s) = \{X \mid X_j > s\} \]

• Then we seek the splitting variable $j$ and split point $s$ that solve
  \[
  \min_{j, s} \left[ \min_{c_1} \sum_{x_i \in R_1(j, s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j, s)} (y_i - c_2)^2 \right]
  \]
Regression trees - CART algorithm cont.

- Starting with all of the data, consider a splitting variable $j$ and split point $s$, and define the pair of half-planes
  \[ R_1(j, s) = \{X | X_j \leq s\} \text{ and } R_2(j, s) = \{X | X_j > s\} \]

- Then we seek the splitting variable $j$ and split point $s$ that solve
  \[
  \min_{j, s} \left[ \min_{c_1} \sum_{x_i \in R_1(j, s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j, s)} (y_i - c_2)^2 \right]
  \]

- For any choice $j$ and $s$, the inner minimization is solved by
  \[
  \hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j, s)) \text{ and } \hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j, s))
  \]
Regression trees - CART algorithm cont.

• For each splitting variable, the determination of the split point $s$ can be done very quickly and hence by scanning through all of the inputs, determination of the best pair $(j, s)$ is feasible.

• Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of the two regions.
Regression trees - CART algorithm cont.

• How large should we grow the tree? A very large tree might overfit the data, while a small tree might not capture the important structure.

• Find sub-tree of that has the optimal trade-off of accuracy and complexity (the cross-validation is used to finding this sub-tree).

“The Elements of Statistical Learning, Friedman

“Inside every big tree is a small, perfect tree waiting to come out.”
- Dan Steinberg
Is the regression tree a strong learner?
The origin of boosting

• The idea of boosting came out of the idea of whether a weak learner can be modified to become better.

• A weak learner is defined as one whose performance is at least slightly better than random chance.

• The idea is to use the weak learning method several times to get a succession of hypotheses, each one refocused on the examples that the previous ones found difficult and misclassified.
Gradient Boosting

- AdaBoost (Adaptive Boosting) was the first boosting algorithm. Gradient boosting generalizes it.

**Gradient boosting involves three elements:**

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.
Gradient Descent - a short primer

• We want to minimize the function $f(x)$. Assume $x$ to be a scalar.
• One way to *iteratively* minimize, and find the corresponding $x$ at the minima, is to follow this update rule at the $i^{th}$ iteration:
  $$x^{(i)} = x^{(i-1)} - \eta \frac{\partial f(x^{(i-1)})}{\partial x}$$
  where $\eta$ is a positive constant.

• We stop when $x^{(i)} = x^{(i-1)}$.
• We may start with an arbitrary value for $x^{(0)}$. 
Gradient Descent - a short primer

• In the case where \( x \) is a vector, we adjust every individual *dimension* of \( x \) based on the slope along that direction. For the \( i^{th} \) iteration and the \( j^{th} \) dimension, the update rule is:

\[
x_j^{(i)} = x_j^{(i-1)} - \eta \frac{\partial f(x^{(i-1)})}{\partial x_j^{(i-1)}}
\]

• At each iteration *all* dimensions are adjusted. The idea is to move the vector \( x \), as a whole, in a direction where *each individual component* minimizes \( f(x) \).
Gradient Boosting

- We start with a function to minimize. Function whose value increases with how bad the regressor is.

- We refer to this function as the *loss function* represented by $L$. For gradient boosting loss functions must be *differentiable*.

- An example is the *squared error* between the actual and predicted value i.e.

$$L = (y_i - h(x_i))^2$$

*Friedman (2001)*
• We want to minimize $f(x) = \sum_{i=1}^{N} L(y_i, h(x_i))$ i.e. the loss over all points $(x_i, y_i)$.

Here $h(x_i)$ is the regressor, which for we'll refer to as the predictor. $N$ is the total number of points.

• In the example of Gradient Descent we minimized with respect to $x$. What are we minimizing w.r.t here? We are minimizing w.r.t the predictor function $h(x)$ since we want a predictor that minimizes the total loss $f(x)$. 
Moving to the iterative world of gradient descent these are the steps we now take:

1. Initialize $h^0(x) = c$, a constant, such that $c$ minimizes $f(x)$ i.e. pick $c$ that minimizes $\sum_{i=1}^{N} L(y_i, c)$.

2. At the $i^{th}$ iteration, for $j = 1, 2, ..., N$ compute

$$r_{ji} = -\frac{\partial L(y_j, h^{i-1}(x_j))}{\partial (h^{i-1}(x_j))}$$

for the squared error $\frac{\partial L}{\partial h} = -2(y - h)$

we are only plugging in the values of $y_j$ and $h^{i-1}(x_j)$ in the differentiated expression. This is analogous to how we dealt with the components of the vector $x$ in the previous section.
3. The previous step gives us a value \( r_{ji} \) for each point \( j \). Thus we have a set of tuples \( (x_j, r_{ji}) \). We use this set of points as training data to construct a regression tree that can predict \( r_{ji} \) given \( x_j \). This tree approximates the gradient.

This takes place of the \( \frac{\partial f(x^{(i-1)})}{\partial x_j^{(i-1)}} \) expression we saw in GD, with this one tree sort of embodying the gradient for all points \( x_j \). We'll refer to this tree as \( T_g^{(i)} \) (g for gradient, i is for the iteration). As before we want this gradient-tree to play a role in the update equation, but we are still left with the task of finding \( \eta \).
4. Assume that the tree $T^{(i)}_g$ has $K$ leaves. The leaves of a tree fragment the feature space into disjoint regions. Let's refer to these regions as $R_k$, for $K = 1, 2, \ldots, K$.

If we send each point $x_j$ down the tree $T^{(i)}_g$, it will end up at some region $R_k$. We now want to associate a constant $\eta_k$ for each such region $R_k$ such that the loss in a region, defined as:

$$\sum_{x_j \in R_k} L(y_i, h^{(i-1)}(x_j) + \eta_k)$$

is minimized. These are solved as $k$ (simple) independent minimization problems for the $k$ regions.
4. cont. Note that now we have a tree providing well defined regions $R_k$, and we have constant values $\eta_k$, associated with each of these regions. In effect, this combination may be seen as another tree: structure given by $T_g^{(i)}$, but $\eta_k$ as predictions at the leaves.
5. Finally, we come to the update step:

\[ h^{(i)}(x) = h^{(i-1)}(x) + \sum_k \eta_k I(x \in R_k) \]

- This second term is effectively a tree derived from \( T_g^{(i)} \).
- We can see why \( \eta_k \) was determined in the way it was: the minimization in the last step and the updation have the same form; thus the updated function has the minimum possible loss.
- The update equation is similar to that of GD.
- It is very interesting that there is actually no addition taking place in this updation step - if you want to compute \( h^{(i)}(x) \), compute \( h^{(i-1)}(x) \), and add to it what ever \( \eta_k \) you obtain by passing \( x \) down the tree represented by the second term.
6. Keep going back to step 2 till you have iterated the number of times – say \( M \) – you want to (in article \( M = 4000 \)).

7. Finally return \( h^{(M)}(x) \) as your predictor. Since at every iteration, our only update to the function at that point is adding a tree in step 5, what you finally return is a sum of trees. Or rather, we return a bunch of trees whose sum (plus \( c \) from Step 1) is supposed to give us the function \( h^{(M)}(x) \).
Gradient Boosting - Summary

In essence we have learnt a function $h^{(M)}(x)$ based on the values $(x_i, y_i)$, that minimizes prediction errors $f(x)$. The minimization is done in multiple steps:

at every step we add a tree (Steps 4 and 5) that emulates adding a gradient based correction - very much like in GD.
Partial dependence plots

• An insight into the contribution of the different features in the algorithm’s predictions.

• PDPs graphically visualize the marginal effect of a given feature on prediction outcome after accounting for the average effect of all other features.

• While this effect may be indicative of feature importance, it may also be misleading due to higher-order interactions. Nonetheless, PDPs are commonly used for knowledge discovery in large datasets such as this.
Partial dependence plots

- Consider the subvector $X_S$ of $l < p$ of the input predictor variables $X^T = (X_1, X_2, ..., X_p)$, indexed by $S \subset \{1,2, ..., p\}$.
- Let $C$ be the complement set, with $S \cup C = \{1,2, ..., p\}$.

- A general function $f(X)$ will in principle depend on all of the input variables: $f(X) = f(X_S, X_C)$.

- One way to define the average or partial dependence of $f(X)$ on $X_S$ is
  \[ f_S(X_S) = E_{X_C} f(X_S, X_C) \]
Partial dependence plots

- **partial dependence**: \( f_S(X_S) = E_{X_C} f(X_S, X_C) \)

- This is a marginal average of \( f \), and can serve as a useful description of the effect of the chosen subset on \( f(X) \).

- Partial dependence functions can be used to interpret the results of any “black box” learning method. They can be estimated by:

  \[
  \hat{f}_S(X_S) = \frac{1}{N} \sum_{i=1}^{N} f(X_S, X_{iC})
  \]

- Where \( \{X_{1C}, X_{2C}, \ldots, X_{NC}\} \) are the values of \( X_C \) occurring in the training data.
Partial dependence plots

A  Meal carbohydrates (2)

<table>
<thead>
<tr>
<th>Weight (g)</th>
<th>Partial dependence (a.u.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8400</td>
</tr>
<tr>
<td>40</td>
<td>7518</td>
</tr>
<tr>
<td>80</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>

C  Meal fat / carbohydrates (4)

<table>
<thead>
<tr>
<th>log2(fat/carbs)</th>
<th>Partial dependence (a.u.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>8303</td>
</tr>
<tr>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>7611</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Background

Meal response predictor

Results

B  Carbohydrate-only prediction

\[ R = 0.38 \]

C  Calories-only prediction

\[ R = 0.33 \]

D  Main cohort prediction (cross-validation)

\[ R = 0.68 \]

E  Validation cohort prediction

\[ R = 0.70 \]
Happy Hanukkah!

Don’t forget to check your predicted glycemic response to sufganiot...