Value of information in Analytica

Rob Johnson `rob.johnson@mrc-bsu.cam.ac.uk`
Chris Jackson `chris.jackson@mrc-bsu.cam.ac.uk`
MRC Biostatistics Unit, University of Cambridge, U.K.

February 8, 2018

This document accompanies the Analytica `VOI` module.

Suppose we have a model which relates an outcome or decision to a set of input variables. Uncertainty on the inputs is expressed by probability distributions. The purpose of the model can be either to make a decision between a finite number of actions, or to estimate some quantity precisely.
VOI (value of information) methods have a dual purpose, to determine:

- which input variables explain the most uncertainty in the outcome
- where it would be most valuable to obtain further data to improve the model

Specifically, the `VOI` module calculates the *expected value of partial perfect information (EVPPI)* from a random sample consisting of draws from the distributions of the input variables and the corresponding model outcomes.

The theory of VOI is introduced in Section 1. Use of the functions in Analytica is described in Section 2. An explanation of how EVPPI is calculated is given in Section 4.

1 Value of Information: definitions

The EVPPI is based on Bayesian decision theory (Raiffa and Schlaifer, 1961; Parmigiani and Inoue, 2009) and is defined as the expected benefit (or reduction in loss) if we were to learn the exact value of one of the model inputs.

We denote the set of all model input parameters as X. Denote the ith input parameter by X_i, and denote the set of all inputs excluding X_i by X_{-i}.

1.1 Model is used for decision making

Suppose the purpose of the model is to choose between a set of decisions $d = 1, ..., D$. The optimal decision is the one which maximises the expected return $R(d, X)$ (or utility, or benefit) under current information about X. The "current information" is quantified by the uncertainty distributions defined for X. The expected return for the optimal decision is therefore:

$$\max_d \mathbb{E}_X\{R(d, X)\} \tag{1}$$

Suppose we were to learn the exact value of a particular input parameter X_i, but the remaining parameters X_{-i} are still uncertain. The expected return would then be

$$\max_d \mathbb{E}_{X_{-i}|X_i}\{R(d, X_i, X_{-i})\}$$

However, under current information, we do not know what this value will be. Therefore to calculate the expected return after learning X_i, we take a further expectation over the current uncertainty distribution of X_i:

$$\mathbb{E}_{X_i}\left[\max_d \mathbb{E}_{X_{-i}|X_i}\{R(d, X_i, X_{-i})\}\right] \tag{2}$$

The *expected value of partial perfect information* (EVPPI) is then defined as the expected increase in return if we were to learn X_i:

$$\text{EVPPI}(X_i) = \mathbb{E}_{X_i}\left[\max_d \mathbb{E}_{X_{-i}|X_i}\{R(d, X_i, X_{-i})\}\right] - \max_d \mathbb{E}_X\{R(d, X)\}. \tag{3}$$

Equivalently, the decision problem can be defined as minimising the expected *loss* $L(d, X) = -R(d, X)$. The EVPPI is then the expected reduction in loss if we were to learn X_i:

$$\text{EVPPI}(X_i) = \min_d \mathbb{E}_X\{L(d, X)\} - \mathbb{E}_{X_i}\left[\min_d \mathbb{E}_{X_{-i}|X_i}\{L(d, X_i, X_{-i})\}\right]. \tag{4}$$

1.2 Model is used for estimating a quantity

Suppose the model is not used to choose between a series of decisions, but simply to estimate the value of some quantity Y of interest, which is a function of the uncertain model inputs, $Y = h(X)$. This can also be expressed as a decision problem, which enables VOI ideas to be used.

The space of possible decisions d is the (typically infinite) set of all values that Y could take. To complete the definition of the decision problem, we need to define a loss for each decision d (point estimate of Y) given the true value of X. A reasonable loss function here is the *squared error* of an estimate $d = \hat{Y}$ compared to the true value $Y = h(X)$, so that $L(d, X) = (\hat{Y} - Y)^2$.

The optimal estimate is then the mean $E_X(Y)$ over all simulations from the distribution of X, and the expected loss at this estimate is the corresponding variance $var(Y) = E((Y - E(Y))^2)$.

The EVPPI for an input quantity of interest X_i is the expected reduction in loss if the exact value of X_i were to be learnt. In this case it is simply the expected reduction in the variance of the output if we were to learn X_i:

$$\text{EVPPI}(X_i) = \text{var}(Y) - E_{X_i}\{\text{var}(Y|X_i)\} \tag{5}$$

2 How to use the VOI module in Analytica

The module can be copied into an existing model or imported as a library.

Figure 1 shows the Analytica VOI module. The two user functions are within `VOI module`, and are called `Point_EVPPI` and `Decision_EVPPI`.

2.1 Decision making

`Decision_EVPPI` is the function used when assessing the value of information about a variable when making a *decision* between a finite set of options. It takes two parameters. The first is a matrix of random samples for the set of uncertain input variables X (model parameters), where in one dimension are the different variables (with index `all_parameters`) and in the other the samples. The other parameter is a matrix of the corresponding *losses* $L(d, X)$, where in one dimension are the decision options (with index `modes`) and in the other the samples. The decision is made by minimising the expected loss over the decision options.

The command chain leading to `D_EVPPI` in Figure 1 is

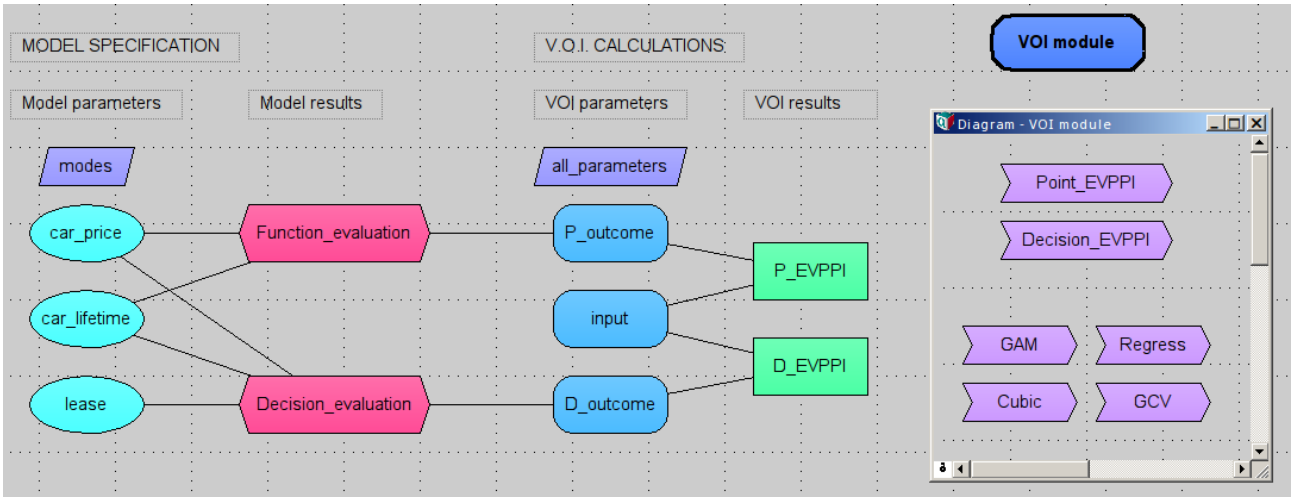

Figure 1: VOI in Analytica. The visualisation of the calculation is separated into three parts. The left-hand part contains the user-specified model. The central part contains the VOI calculations. The model provides input for these. The right-hand module contains the functions for the VOI calculations.

```
all_parameters := [car_price, car_lifetime, lease];

input := Sample(all_parameters);

D_outcome := Sample(Decision_evaluation);

D_EVPPI := Decision_EVPPI(input,D_outcome);
```

2.2 Point estimation

`Point_EVPPI` is the function used to assess the expected value of learning an input variable X_i for improving the estimate of an outcome *value*. It takes two parameters. The first is a matrix of random samples for a set of uncertain input variables (model parameters), where in one dimension are the different variables and in the other the samples. The other parameter is a vector of the corresponding sampled outcomes.

The command chain leading to `P_EVPPI` in Figure 1 is

```
all_parameters := [car_price, car_lifetime, lease];

input := Sample(all_parameters);

P_outcome := Sample(Function_evaluation);

P_EVPPI := Point_EVPPI(input,P_outcome);
```

2.3 User notes

Which function should I use? The correct function to use depends on the question to be answered. If the objective is to improve decision making, i.e. you have a decision function and some variables that influence the decision, the function to use is `Decision_EVPPI`. If the objective is to improve estimation of a value, e.g. a cost or a parameter, which is continuous in nature, then the function to use is `Point_EVPPI`.

Do I need to set any parameters? You should not need to set or alter any parameters to use these functions. The methods used to compute EVPPI may need tuning in rare cases, by altering the functions named `Cubic`, `GAM`, `Regress` and `GCV`. For details of these methods and how to tune them, see Section 4 and Appendix A. Their default settings have worked well in our applications.

What assumptions does the method make? The method used to calculate EVPPI assumes that the output is a smooth function of the inputs, and is based on approximating this function. If you think the relationship between input and outcome is not well represented by a smooth curve because of a discontinuity that is very abrupt, or if subtle details of the curve are of particular interest, then the method might not be suitable. A piecewise curve might be more appropriate (see Section B).

The method for estimating EVPPI for point estimation also assumes that the outcome has an approximately Normal distribution, but in theory the method can be extended to more general cases by advanced users (see Section B). However, the input distributions can take any form. Indeed, they need not even be samples from Analytica-defined distributions, but can simply be a set of random values that represent your input of interest.

3 Worked examples

We use the example of car hire vs. car purchase. There are three uncertain input variables: the cost of buying a car (X_1), the lifetime in years of a bought car (X_2), and the monthly cost of leasing a car (X_3).

The outcomes we are interested in are, for the decision scenario, which is more cost-effective: buying or leasing a car; and, for the point-estimation scenario, what is the monthly cost of buying a car.

3.1 Decision making

We are aiming to minimise the cost (or the "loss", in the language of decision theory) of accessing a car. We are comparing the uncertain lease cost, X_3, to the uncertain monthly cost of buying a car, $X_1/X_2/12$. EVPPI will tell us which uncertain input variables are most useful to learn in terms of reducing our cost.

Figure 2 (left) shows that the most valuable variable to learn, in terms of cost reduction, is the price of the car, followed by the lease cost.

3.2 Point estimation

We are aiming to estimate the unknown monthly cost of buying a car, $X_1/X_2/12$. EVPPI will tell us which unknown variables are most useful to learn in terms of reducing uncertainty in our estimate of the monthly cost.

Figure 2 (right) shows that the most valuable variable to learn, in terms of improving the estimate of the cost, is the purchase price of the car, since most of the variance in the outcome is attributable to that variable.

Note that we know the true EVPPI for the lease price in this case is exactly zero, because it does not influence the monthly cost of buying. However the *estimate* of the EVPPI is not exactly zero, because there is a small error due to the approximation method used (described in section 4). Running more Monte Carlo simulations is expected to reduce this error. In practice it is recommended to run enough simulations to ensure that the estimate of EVPPI (and the other results of the model) converge to a stable value.

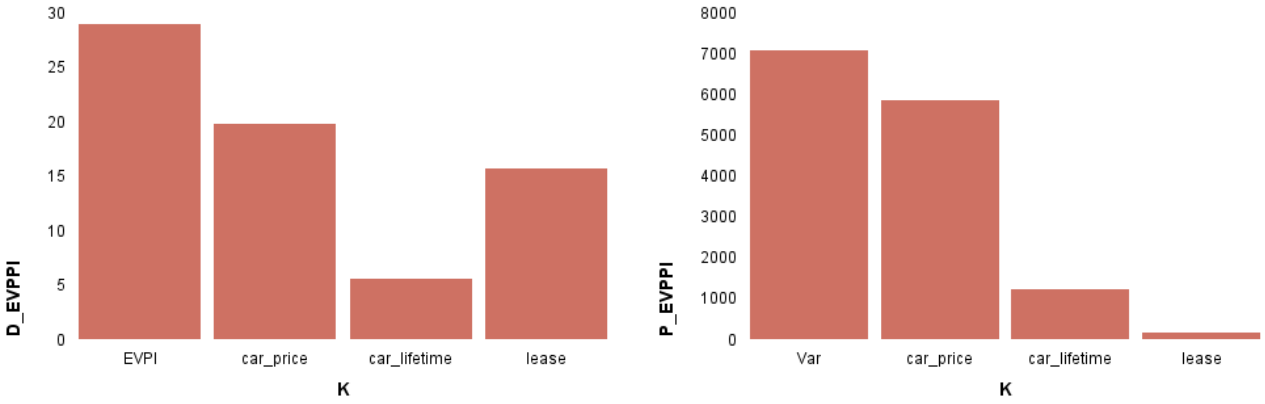

Figure 2: Results for the worked examples. Left: EVPPI for decision making. The units of the y-axis represent loss, i.e. cost. The left-most bar shows the expected value of perfect information (EVPI). The bar heights represent the reduction in loss, or the financial benefit, expected if we were to learn the exact value of the parameter. Right: EVPPI for point estimation. The left-most bar shows the total variance of the outcome. The other three "variable" bars show the reduction in variance to be expected should the variable be learnt.

4 Method of calculating EVPPI

This section outlines the method used by the Analytica module to calculate the EVPPI. The Analytica functions `GAM`, `Cubic`, `Regress`, `GCV` used to implement this method are documented in further detail in Appendix A.

$\mathbb{E}_X\{R(d,X)\}$ in Equation 3 could simply be estimated as the mean over the values of $R(d,X)$ given by K randomly sampled values for X. A "brute force" method for estimating the nested expectation in Equation 3 is shown by Brennan et al. (2007). This involves two loops of Monte Carlo simulations, an outer loop with K iterations and an inner loop with J iterations. Thus Equation 3 can be estimated as

$$\text{EVPPI}(X_i) \approx \frac{1}{K}\sum_{k=1}^{K} \max_d \frac{1}{J}\sum_{j=1}^{J} R\left(d, x_i^{(k)}, x_{-i}^{(j,k)}\right) - \max_d \frac{1}{K}\sum_{k=1}^{K} R(d, x^{(k)}), \tag{6}$$

where $x^{(k)}$ is a random sample from the distribution of X. However, this method is too computationally intensive to provide accurate estimates of EVPPI in most realistic situations.

Instead, in the Analytica module, the EVPPI is estimated by the method of Strong et al. (2014). This method uses generalised additive model (GAM) regression (Wood, 2006) fitted to the sampled outputs in terms of the sampled inputs, in order to estimate how the outputs depend on the inputs. This allows EVPPI to be calculated using a single level of Monte Carlo simulation.

This method of calculating EVPPI for decision problems is presented in Section 4.1. A generalisation of this method to point estimation, from Jackson et al. (2017), is presented in Section 4.2.

In our implementation, the regression model assumes the outputs are continuous and errors are normally distributed, though the theory of GAMs allows for these assumptions to be relaxed.

4.1 Calculating EVPPI for decision making

Strong et al. (2014) estimate the EVPPI for decision-making uncertainty (Equation 3) as follows. Firstly the function defininining the return (or utility, or benefit) for decision d and input variables X is expressed as

$$R\left(d,X\right) = g\left(d,X_i\right) + \epsilon, \tag{7}$$

where g is some function of the input variable of interest X_i, and ϵ is a noise term with mean 0, so that $E_{X_{-i}|X_i}\left\{R\left(d,X\right)\right\} = g\left(d,X_i\right)$. The purpose of the generalised additive model is to estimate the function $g()$ that approximates how the expected return depends on the input variable of interest X_i. Then the EVPPI can be estimated from the fitted values from the regression, $\hat{g}$, for the K samples of X_i as

$$\widehat{\text{EVPPI}}(X_i) = \frac{1}{K}\sum_{k=1}^{K} \max_d \hat{g}\left(d, x_i^{(k)}\right) - \max_d \frac{1}{K}\sum_{k=1}^{K} \hat{g}\left(d, x_i^{(k)}\right). \tag{8}$$

The function $g()$ is intended to be smooth but arbitrarily flexible, and is defined as a cubic regression spline. Details of the spline fitting are given in Appendix A.

4.2 Calculating EVPPI for point estimation

Recall that if the purpose of the model is to estimate a quantity rather than to make a decision, the EVPPI for an input parameter X_i is the expected reduction in variance if we were to learn the exact value of X_i (Equation 5).

This can again be calculated using a sample of size K from the model inputs X and corresponding model outputs $Y = h(X)$, where $k = 1, \ldots, K$. Then (Jackson et al., 2017) the estimate of the EVPPI is

$$\widehat{\text{EVPPI}}(X_i) = \widehat{\text{var}}(Y) - \frac{1}{K}\sum_{k=1}^{K}\left(y^{(k)} - \hat{g}\left(x_i^{(k)}\right)\right)^2. \tag{9}$$

The first term in Equation 9 is the sample variance of the $y^{(k)}$. The second term is the expected variance of the outcome given the input variable of interest X_i. This is calculated as the sample variance of the "residuals" (observed values minus fitted values) from the regression of the sampled outputs $y^{(k)}$ on the inputs $x_i^{(k)}$. Again, spline regression is used, detailed in Appendix A.

Appendix A Analytica functions for fitting generalised additive models

A.1 `Decision_EVPPI/Point_EVPPI`

These functions take as inputs variable samples and resulting loss samples. They return EVPPI estimates for the variables, appended to the EVPI (for `Decision_EVPPI`) or the output variance (for `Point_EVPPI`). They call the function `GAM` in calculating Equations 8 and 9.

A.2 `GAM`

This function returns the estimated output values given the input of interest – i.e. an estimate $\hat{g}$ of g in Equation 7. It takes as input two vectors, x and y, both of length K (the number of samples).

The function first defines x^*, of length q, which specifies the location of *knots* in the spline. q determines the level of flexibility in the curve relating the input and the outcome.

Using the input and the knots, the function forms two matrices. First, a $K \times (q+2)$ matrix M, whose i-th row is defined as follows:

$$M_i = [1, x_i, \mathcal{C}(x_i, x_1^*), \mathcal{C}(x_i, x_2^*), ..., \mathcal{C}(x_i, x_q^*)], \tag{10}$$

where the function $\mathcal{C}$ is defined in `Cubic`.

Second, a *penalty* matrix, S, is formed. It is a $(q+2) \times (q+2)$ matrix, whose i-th row/column (for $i = 3, ..., q+2$) is defined as follows:

$$S_i = [0, 0, \mathcal{C}(x_{i-2}^*, x_1^*), \mathcal{C}(x_{i-2}^*, x_2^*), ..., \mathcal{C}(x_{i-2}^*, x_q^*)]. \tag{11}$$

M and S define the model matrix on which the model outcome is regressed using a penalised least squares method using the `Regress` function. The model is optimised according to the GCV score calculated using the function `GCV`.

A.3 `Cubic`

This function implements a simple cubic spline for a sample x_i transformed to a new variable v on the interval $[0, 1]$. It takes as input two variables (v and the knot locations x^*) and returns a single value. The spline is defined (Wood, 2006)

$$\mathcal{C}(v, x^*) = \frac{1}{4}\left[(x^* - 1/2)^2 - 1/12\right]\left[(v - 1/2)^2 - 1/12\right] - \frac{1}{24}\left[(|v - x^*| - 1/2)^4 - 1/2(|v - x^*| - 1/2)^2 + 7/240\right]. \tag{12}$$

It is used by `GAM` to form the M and S matrices.

A.4 `Regress`

This function, called from within `GAM`, performs regression. It takes as input the model outcome, and the penalised, smoothed model. It returns the fitted values from the regression model ($\hat{g}$).

We obtain $\hat{g}$ by estimating the regression parameters β, using penalised iteratively re-weighted least squares. First, model-specific values are calculated using the current estimates for $\hat{\beta}$. Given a link function f and the outcome y with estimated mean $\mu = M\hat{\beta}$:

$$W_{i,i} \propto \frac{1}{V(\mu_i) f'(\mu_i)^2}, \qquad z_i = f'(\mu_i)(y_i - \mu_i) + M_i\hat{\beta}. \tag{13}$$

Second, β is updated by minimising

$$\left|\left|\begin{bmatrix}\sqrt{W} & 0\\ 0 & I\end{bmatrix}\left(\begin{bmatrix}z\\ 0\end{bmatrix}-\begin{bmatrix}M\\ \sqrt{\lambda S}\end{bmatrix}\beta\right)\right|\right|^2. \tag{14}$$

The above process iterates to convergence below some distance threshold between μ and z.

Currently implemented in Analytica is a linear link function: $f'(\mu_i) = 1$. We have a Gaussian distribution model for the outcome, hence $V(\mu_i) = 1$ (Wood, 2006). Then $W_{i,i} = 1$ and $z_i = (y_i - \mu_i) + M_i\beta = y_i$. Thus, Equation 14 amounts to regressing $\begin{bmatrix}M\\ \sqrt{\lambda S}\end{bmatrix}$ onto $\begin{bmatrix}z\\ 0\end{bmatrix}$.

A.5 `GCV`

The optimal value for the penalty parameter λ is chosen as that which minimises the value of a generalised cross-validation statistic calculated by this function. The function `regress` is performed for 50 different values of λ (ranging from 10^{-5} to 10^9) and each model is assessed by `GCV`. The number of trials is set in `GAM`.

Appendix B Advanced use: changing the defaults

In occasional situations the default settings and assumptions of the functions above may need to be changed.

Increasing the number of knots The number of knots defaults to 10. This allows good flexibility for fitting different shapes. The number does not need to be decreased, as there is an inbuilt penalty to prevent overfitting, but decreasing this number will increase the speed of the program. Therefore, if you are confident of having enough points and need the gain in speed, decrease the number of knots. On the contrary, a user might want to increase this number if they deem 10 knots insufficient to capture the shape of the relationship between input and outcome. Increasing the number of knots will allow greater flexibility.

Piecewise curves A relationship between an input and an outcome that includes a discontinuity might be well described with a piecewise curve. To use a piecewise GAM, you would need to edit `Point_EVPPI` (or `Decision_EVPPI`) so that each part of the curve is fit separately. Note that you would need to specify the parameter under consideration. For example:

```
var threshold := 10;
var single_input := input[all_parameters=car_lifetime];

Index low_index := Subset(single_input<threshold);
Index high_index := Subset(single_input>=threshold);
var length1 := Size(low_index);
Index ind1 := 1..length1;
Index ind2 := 1..(SampleSize-length1);

var single_input1 := Array(ind1,single_input[Run=low_index]);
var single_input2 := Array(ind2,single_input[Run=high_index]);
var output1 := Array(ind1,output[Run=low_index]);
var output2 := Array(ind2,output[Run=high_index]);
```

```
var regression_fit1 := GAM(single_input1,output1,R:ind1);
var regression_fit2 := GAM(single_input2,output2,R:ind2);
var regression_fit := Array(Run,0);
regression_fit[Run=low_index] := Array(low_index,regression_fit1)[low_index=Run];
regression_fit[Run=high_index] := Array(high_index,regression_fit2)[high_index=Run];
```

Different outcome distributions and link functions The VOI module is currently defined for Normally distributed outcomes with linear link functions to the inputs. To add alternative distributions and link functions, edit the "family", "link", "weight", and "derivative" variables/functions in the `Regress` function (see Appendix A and Wood (2006), pages 74 and 149, for further guidance).

References

Brennan, A., Kharroubi, S., Hagan, A. O. and Chilcott, J. (2007), 'Calculating Partial Expected Value of Perfect Information via Monte Carlo Sampling Algorithms Alan', *Medical Decision Making* **27**(4), 448–470.

Jackson, C., Presanis, A., Conti, S. and De Angelis, D. (2017), 'Value of Information: Sensitivity Analysis and Research Design in Bayesian Evidence Synthesis', *arXiv* .

Parmigiani, G. and Inoue, L. (2009), *Decision theory: principles and approaches*, Wiley.

Raiffa, H. and Schlaifer, R. (1961), *Applied statistical decision theory*, Harvard University.

Strong, M., Oakley, J. E. and Brennan, A. (2014), 'Estimating multiparameter partial expected value of perfect information from a probabilistic sensitivity analysis sample: a nonparametric regression approach.', *Medical decision making : an international journal of the Society for Medical Decision Making* **34**(3), 311–26.

Wood, S. N. (2006), *Generalized Additive Models: An Introduction with R*, Chapman and Hall/CRC, Boca Raton, FL.