Variable selection methods: an introduction

Matteo Cassotti and Francesca Grisoni

Milano Chemometrics and QSAR Research Group - Dept. of Environmental Sciences, University of Milano-Bicocca, P.za della Scienza 1 – 20126 Milano (Italy)

In order to develop regression/classification models, QSAR analysis typically uses molecular descriptors as independent variables. The number of molecular descriptors has hugely increased over time and nowadays thousands of descriptors, able to describe different aspects of a molecule, can be calculated by means of dedicated software. However, when modelling a particular property or biological activity, it is reasonable to assume that only a small number of descriptors is actually correlated to the experimental response and is, therefore, relevant for building the mathematical model of interest.

As a consequence, a key step is the selection of the optimal subset of variables (*i.e.* molecular descriptors) for the development of the model. This is precisely the aim of the so-called variable selection methods, which allow to:

- improve interpretability (simple models);
- neglect not significant effects, thus reducing noise;
- increase the model predictive ability;
- speed up modelling time.

During the years different variable selection methods have been proposed, from relatively simple to more recent ones that took inspiration from different scientific fields, like genetics and ethology. Furthermore, some methods able to perform both regression and variable selection simultaneously have recently been proposed.

In this tutorial we give a short overview of the main variable selection methods.

All Subset Models (ASM)

The All Subset Models (ASM) method is the most simple and computationally consuming. It consists in the generation of all the possible combinations of the p variables, from size 1 to p, p being the total number of variables. This method guarantees that the best subset of variables is found, but it's very computationally consuming, being the total number of combinations of p variables given by:

$$2^p - 1$$

As a consequence, the method becomes unsuitable for large numbers of variables. If one is interested in developing simple models, *i.e.* models comprising a limited number k of variables, one can calculate all the possible combinations of the *p* variables up to a maximum model size *k* (*e.g.* $1 \le k \le 20$), the beneficial effects being both an easier interpretation of the model and a reduced computational time. In this case, given *p* the number of available variables, the total number of models *t*, from size 1 to *k*, is given by:

$$t = \sum_k \left(\frac{p!}{k!(p-k)!} \right) \le 2^p - 1$$

The total number of models to be generated is smaller than the case $k = p$, but it is still huge when the number of variables, *p*, is high (even with a small *k* value).

For example, if we consider a problem where $p = 130$ and $k = 20$, the total number of generated models is 2.04×10^{23}. Assuming that our computer can compute 10.000 models per second, which is a reasonable estimate for current laptops, the time required to compute all the models is 6.46×10^{11} years, which means we should have started long before the Big Bang to have the calculation completed by now. Figure 1 depicts the sharp increase in the number of generated models (in logarithmic scale) for modest increase in the number of variables. An estimate of the computational time is reported along the secondary vertical axis.

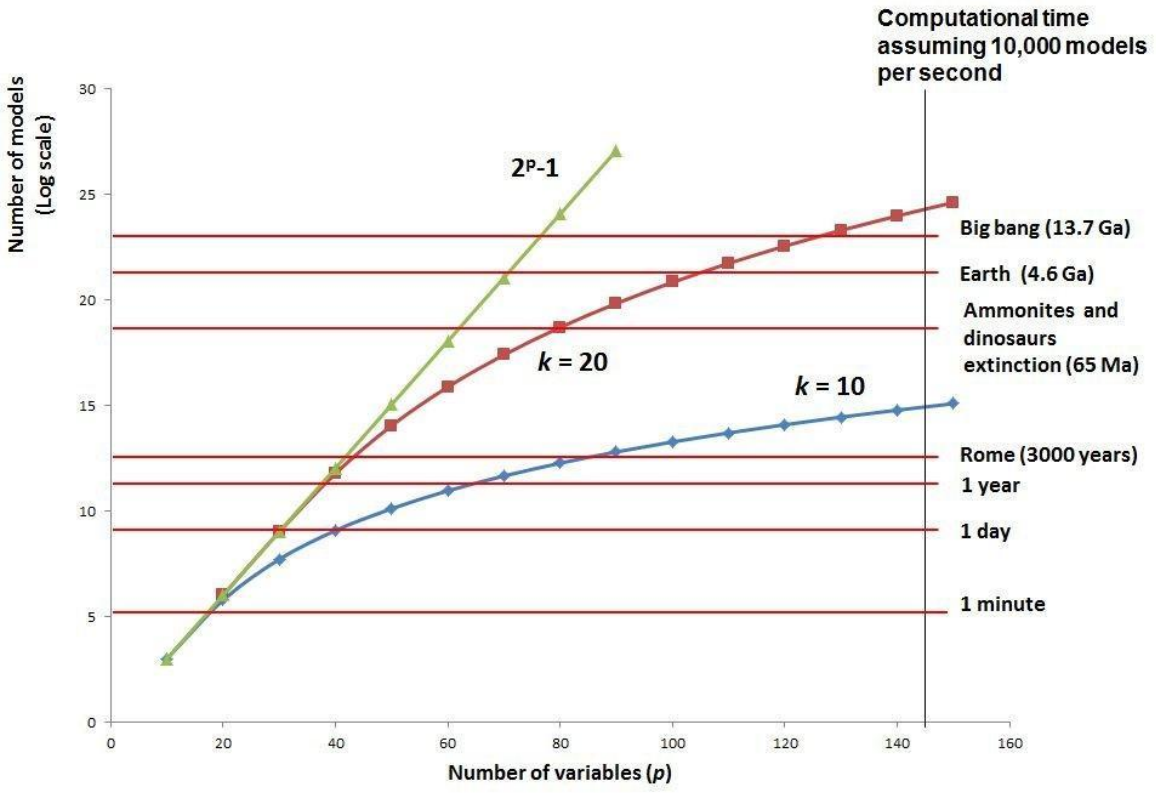

Figure 1. Number of models vs. number of variables for an All Subset Models method with k = 20, k = 10 and 2^p – 1, assuming a computational speed of 10.000 models per second.

Sequential Search (SS)[1]

Sequential Search (SS) is a simple method aimed at finding optimal subsets of variables for a specified model size. The basic idea is to replace each variable at a time with all the remaining ones and see whether a better model is obtained. This procedure differs from the All Subset

Models method in that in this case not all the possible combinations of the *p* variables are tested, the method thus being less time consuming and meta-heuristic.

The initial population is usually randomly generated, giving constraints on the number of variables (size) for each seed (model). All variables are replaced with all the others and the new seed is chosen only after all the variables in the model have been replaced and the obtained models have been compared. Figure 2 depicts the algorithm.

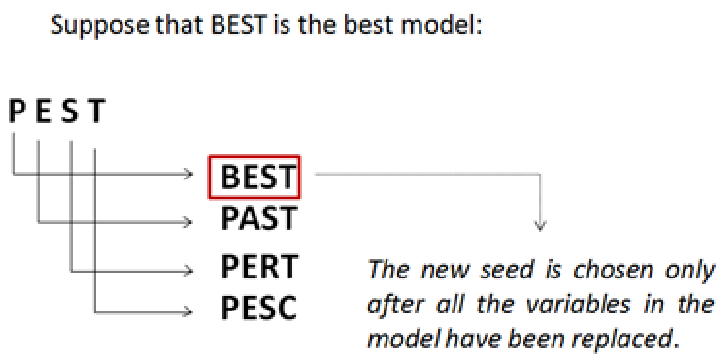
Suppose that BEST is the best model:
PEST
BEST
PAST
PERT
PESC
The new seed is chosen only after all the variables in the model have been replaced.

Figure 2. Depiction of the Sequential Search algorithm.

StepWise methods (SW)[1,2]

StepWise regression methods are among the most known subset selection methods, although currently quite out of fashion. StepWise regression is based on two different strategies, namely Forward Selection (FS) and Backward Elimination (BE).

- Forward Selection method starts with a model of size 0 and proceeds by adding variables that fulfill a defined criterion. Typically the variable to be added at each step is the one that minimizes Residual Sum of Squares (*RSS*) at most. This can be evaluated also by a F-test, defined by:

$$F_j^+ = \max_j \left[\frac{RSS_p - RSS_{p+j}}{s^2_{p+j}} \right] > F_{in}$$

  where RSS_p and RSS_{p+j} are the residuals sum of squares of the models with *p* and *p+j* variables, s^2_{p+j} is the variance of the model built with variables *p+j* and F_{in} is used as stop criterion (usually F_{in} is set to 4), corresponding to the probability α, with 1 degree of freedom for the numerator and (*n-p-1*) for the denominator.

- Backward Elimination method proceeds in the opposite way, in that it starts from a model of size *p*, *p* being the total number of variables, and eliminates not relevant variables in a step by step procedure. In this case, the variable to be deleted is usually the one that give the minimum increase in *RSS*. Like in the FS method this can be evaluated by a F-test, defined by:

$$F_j^- = \min_j \left[\frac{RSS_{p-j} - RSS_p}{s^2_p} \right] < F_{out}$$

  where F_{out} is used as stop criterion.

The original algorithm was later improved by Efroymson[3] in 1960 by combining Forward Selection and Backward Elimination. It starts with Forward Selection and after each variable (other than the first one) is added to the model, a test is made to see if any of the selected variables can be eliminated without largely increasing the *RSS*.

The strategy based on F-tests is quite out of fashion and more recent versions select variables that minimize other functions, such as the Akaike Information Criterion[2,4] (AIC) score, defined as:

$$\min(AIC_p) = -2(L_p - p')$$

where *p′* is the number of parameters in the model and L_p is the log-likelihood of the model with *p′* parameters. For regression models, the optimal complexity according to the Akaike criterion is obtained by minimizing the following[5]:

$$\min(AIC_p) = \frac{n + p' + 1}{n - p' - 1} . RSS_p$$

where *RSS* is the residual sum of squares of the model with *p* variables.
Because often variables come in groups, smart stepwise procedures are able to add or drop whole groups at a time, taking proper account of their degrees of freedom[2].

Genetic Algorithms (GAs)[6,7,8]

Genetic Algorithms (GAs) are an agent-based method presented first in 1961 by Bledsoe[9] and mathematically formalized by Holland[10] in 1975, which takes inspiration from Darwin's theory of the evolution. Each chromosome (model) competes with the others according to the concept of the "survival of the fittest". GAs are based on a bottom-up approach, *i.e.* a complex and adaptive global behavior of the system emerges from simple interactions of the agents[11].

According to GAs terminology, each gene corresponds to a variable and a sequence of genes, *i.e.* a chromosome, to a model. The population of chromosomes is randomly initialized and the presence/absence of each variable is coded by a binary digit. Chromosomes are evaluated for their quality, according to a predefined fitness function (such as, for example, Q^2 leave-one-out) and sorted accordingly. Pairs of chromosomes can generate offspring by a *crossover procedure*, the selection of parent chromosomes being both random or biased towards the best ones. The gene pool shared by parent chromosomes is retained by the offspring, whereas other genes can be changed according to a crossover probability. The second evolution phase, by which new chromosomes can be generated, is a *mutation procedure*, in which each gene can be changed according to a mutation probability. Mutation probability is usually set to a low value in order to avoid considerable drifts, which may lead far from the optimal region. Every time a new chromosome with a better response than already existing ones is generated, it enters the population and the worst model is discarded. In this way chromosomes compete against each other and only the fittest survive. The evolution phase is iterated until a stop criterion is satisfied. Figure 3 provides a scheme for the algorithm of GAs.

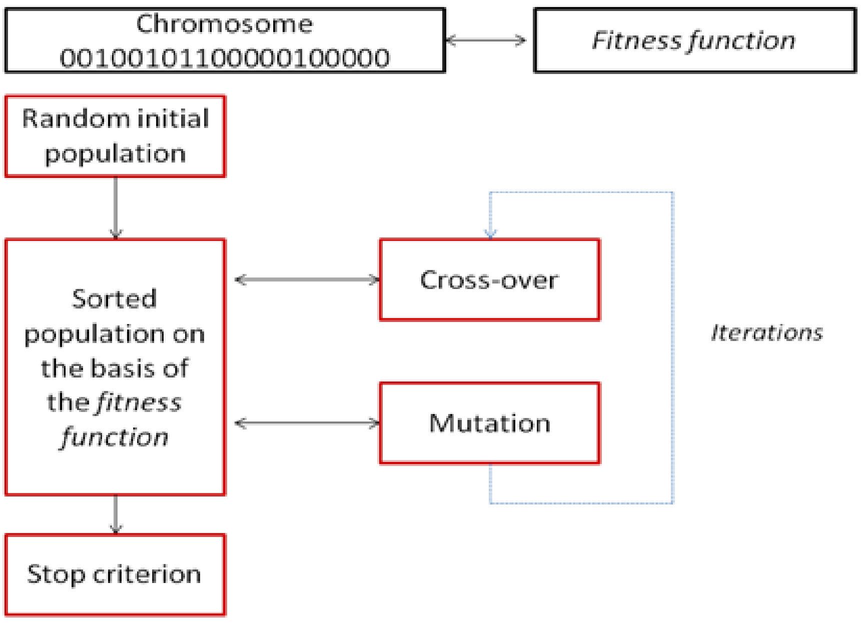

Figure 3. Scheme for GAs.

Particle Swarm Optimization (PSO)[12,7]

Particle Swarm Optimization (PSO) is an agent-based method that was inspired by the behavior of flock of birds. Differently from GA, PSO agents do not compete but cooperate in order to find the optimal solution. PSO was initially thought as an optimization method and was only later modified in order to be specifically applied to variable selection[13].

The number of particles is constant and the population is randomly initialized. PSO particles move in a binary search space and each position corresponds to a model. In the modified version of PSO, at each iteration a velocity vector is randomly extracted in the range [0,1]. The range [0,1] is divided into 3 bins and the new position is calculated depending on the bin the velocity falls into, according to the following formulas:

$$\begin{array}{lll} \text{if}\,(0 < v_{id} \le a) & \rightarrow & x_{id}(new) = x_{id}(old) \\ \text{if}\,(a < v_{id} \le 0.5\cdot(1+a)) & \rightarrow & x_{id}(new) = p_{id} \\ \text{if}\,(0.5\cdot(1+a) < v_{id} \le 1) & \rightarrow & x_{id}(new) = p_{gd} \end{array}$$

where v_{id} is the velocity of the *i-th* particle along the *d-th* dimension (variable), *a* is a tunable parameter, x_{id} is the position of the *i-th* particle along the *d-th* dimension (variable), p_{id} is the value of the *d-th* variable for the *i-th* particle in its previous best personal position (best model found by that particle so far) and p_{gd} is the value of the *d-th* variable for the *g-th* particle corresponding to the best global model found so far. In such a way, the motion of each particle is influenced by its previous personal best position, p_{id}, and by the best global position, p_{gd}. The parameter *a* is named static probability and its initial value is usually set to 0.5. The static probability plays the role of balancing exploration and exploitation: the larger the value of the parameter *a*, the greater is the probability to overleap local optima. On the other hand, a small value of parameter *a* is favorable for the particle to follow the two best past global positions and for the algorithm to converge more quickly. The modified PSO is intended to possess more

exploration ability at the beginning and then an enhanced exploitation ability to search the local area around the particle. Accordingly, the parameter is set to decrease along with the generations. Therefore, the static probability *a* usually starts with a value equal to 0.5 and decreases to a final value equal to 0.33 at convergence[13].
Figure 4 provides a scheme for the PSO algorithm.

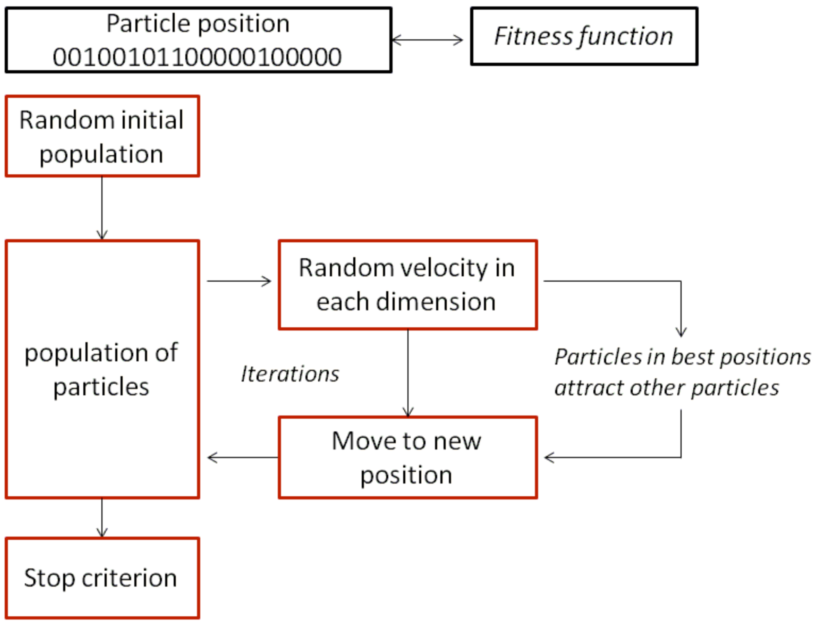

Figure 4. Scheme for PSO algorithm.

Ant Colony Optimization (ACO)[7,14,15]

Ant Colony Optimization (ACO) is another agent-based method that was inspired by nature, the source of inspiration being colonies of ants, who manage to find the shortest path connecting their nest and the source of food by pheromone deposition. As ants travel a route from a starting point to a food source, they deposit pheromone. Subsequent ants will generally choose paths with more pheromone and after many trials they will converge on an optimal path. ACO lies somewhere in between GA and PSO in that particles communicate each other sharing information, but they are also eliminated at each iteration.

Ants are randomly generated with a predefined number of variables, then they start building a path in the search space according to either a probabilistic or deterministic transition rule. Their pheromone deposition will be proportional to the quality of the solution they are able to find.
$\tau_{i,J}$ (t) is the probability (strength of the pheromone trail) of choosing task *J* after task *i*, or the probability of choosing a particular value *J* for a particular parameter *i* in generation *t*. Initially all probabilities are set to the same small, non-zero value. If a transition between particular tasks is used, the pheromone level is increased by an amount proportional to the fitness, *f*(*k*), according to:

$$\Delta\tau_{i,j}(t) = \Delta\tau_{i,j}(t) + f(k)^{\beta}$$

where β is a constant in the range (0,1]. When all the ants have been evaluated, the pheromone level is updated according to:

$$\tau_{i,j}(t) = (1-\rho)\tau_{i,j}(t-1) + \rho\Delta\tau_{i,j}(t)$$

where ρ represents the evaporation rate.
Given *J* the set of all tasks, the next task is chosen from either of the following expressions:

$$\begin{aligned} j &= \max\left\{\tau_{i,j}(t)\right\} \text{ if } r \le r_0 \\ &= J \text{ otherwise} \end{aligned}$$

where the probability of choosing a particular task *j* from the set of available tasks *J* is given by:

$$P_{i,j}(t) = \frac{\left[\tau_{i,j}(t)\right]^{\alpha}}{\sum_{l \in J_i^k}\left[\tau_{i,l}(t)\right]^{\alpha}}$$

where J_i^k is the set of ant *k*'s unvisited variables, *r* is a random number in the range [0,1] and r_0 is a threshold value. If *r* is less than or equal to r_0, a deterministic selection is made (exploitation), while if *r* is greater than r_0 a probabilistic selection is made (exploration). α controls how much the pheromone level affects the probability that a particular task is chosen, whereas β controls the extent to which the fitness increases the amount of pheromone deposited. Another characteristic of the ACO algorithm is the pheromone evaporation rate ρ, a process that leads to a decrease in the pheromone trail intensity over time: it helps in avoiding rapid convergence of the algorithm towards a sub-optimal region. At each iteration, the subsets are evaluated, the pheromone is updated, a new set of ants is created and the procedure is reiterated until a stop criterion is met. The criterion for stopping the search can be based on a pre-defined evaluation function (*e.g.* Q^2 leave-one-out): usually, whether addition/deletion of any feature does not produce a better subset the process is stopped. Figure 5 provides a scheme for the ACO algorithm.

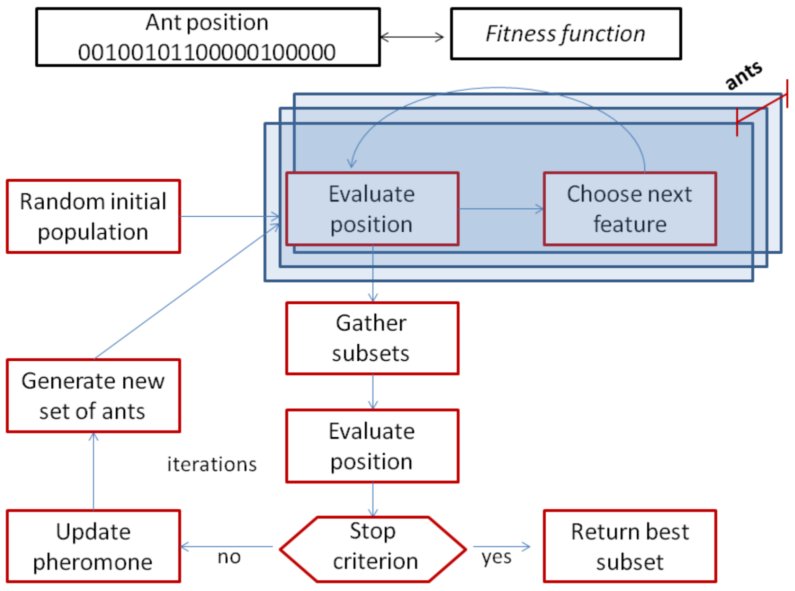

Figure 5. Scheme for ACO algorithm.

Least Absolute Shrinkage and Selection Operator (LASSO)[2,16]

The LASSO is a regression method proposed by R. Tibshirani in 1996. Similar to Ordinary Least Squares (OLS) regression, LASSO minimizes the Residual Sum of Squares (*RSS*) but poses a constraint to the sum of the absolute values of the coefficients being less than a constant. This additional constraint is moreover similar to that introduced in Ridge regression, where the constraint is to the sum of the squared values of the coefficients. This simple modification allows LASSO to perform also variable selection because the shrinkage of the coefficients is such that some coefficients can be shrunk exactly to zero.
The linkage between OLS, LASSO and Ridge is:

$$\text{OLS} = \min\Big(\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j \chi_{ij})^2$$

$$\text{LASSO} = \min\Big(\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j \chi_{ij})^2 + \lambda\sum_{j=1}^{p}\left|\beta_j\right|\Big)$$

$$\text{Ridge} = \min\Big(\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j \chi_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2\Big)$$

where *n* is the number of objects, *p* the number of variables and λ a parameter.
It can be said that LASSO is an improvement over Ridge, in that LASSO has the beneficial aspects of Ridge, *i.e.* higher bias and lower variance (compared to OLS), but also allows to select variables, leading to an enhanced interpretability of the developed models. The λ parameter can be tuned in order to set the shrinkage level, the higher the λ is, the more coefficients are shrunk to 0.

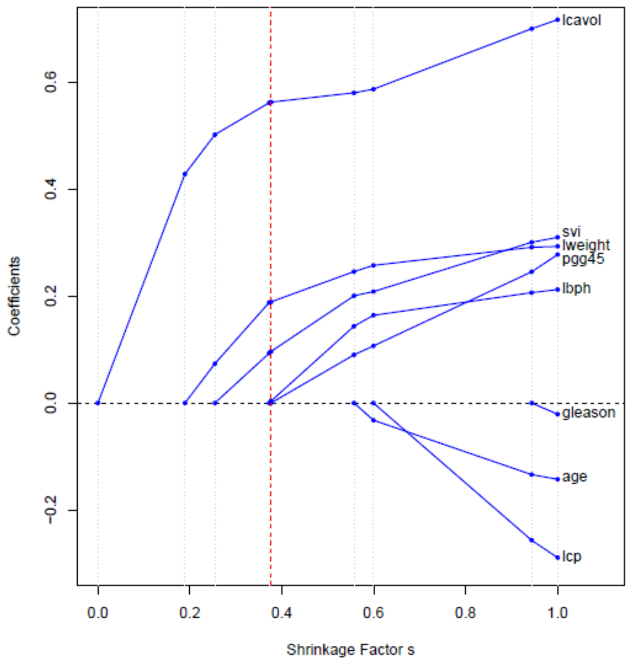

Figure 6. Shrinkage of coefficients by LASSO. When s = 1, LASSO and OLS solutions coincide (right side of the graph). When s < 1 regression coefficients are shrunk. The lower the s value is, the more the coefficients are shrunk to 0. For example, for s close to 0.4 (dotted red line), only 3 variables are selected.

The shrinkage level can also be set by the shrinkage factor *s*, defined as:

$$s = \frac{\sum|\boldsymbol{\beta}|}{\sum\left|\hat{\boldsymbol{\beta}}_{LS}\right|}$$

where $\hat{\boldsymbol{\beta}}_{LS}$ are the OLS coefficients. When *s* = 1, the shrinkage level is zero and the LASSO solution corresponds to the OLS solution; when *s* < 1, LASSO shrinks the coefficients (the lower the *s* value, the higher the λ value). For certain values of *s*, some coefficients are shrunk exactly to zero. This path is depicted in Figure 6[2].

Elastic Net[2]

The Elastic Net is a regression method proposed by Zou and Hastie[17] in 2005 that combines the penalty terms of LASSO and Ridge regression.

$$\lambda \cdot \sum_{j=1}^{p}((1-\alpha)\left|\beta_j\right| + \alpha\beta_j^2)$$

The Ridge term (β_j^2) allows to shrink the coefficients, whereas the LASSO term is able to shrink some coefficients to 0, thus performing variable selection. The two terms can be properly tuned by the parameter α, depending on the problem under analysis. The Elastic Net method seems to be particularly useful when dealing with highly correlated variables. In such a situation the Ridge term shrinks coefficients of correlated variables toward each other, whereas the LASSO term picks one among the correlated variables and puts all weight on it.
Figure 7[2] shows a comparison of the effects of the shrinkage between LASSO and Elastic Net. It is clear how the Elastic Net results in more non-zero coefficients, but with smaller magnitude.

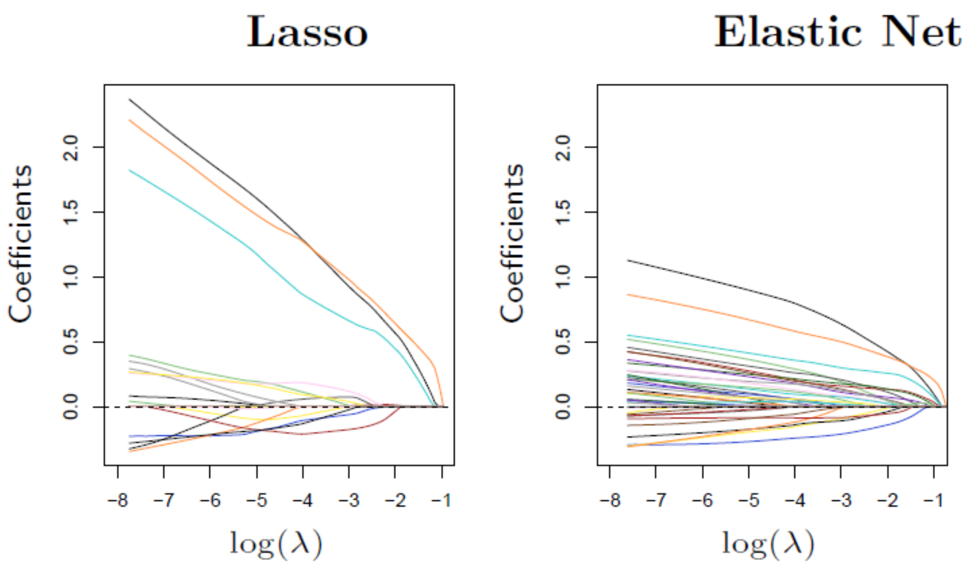

Figure 7. Comparison of shrinkage between LASSO and Elastic Net.

Variables Importance on PLS projections (VIP)[18,19]

The Variables Importance on Partial Least Squares (PLS) projections (*VIP*) is a variable selection method based on the Canonical Powered PLS (CPPLS)[20] regression. The CPPLS algorithm assumes that the column space of ***X*** has a subspace of dimension M containing all information relevant for predicting *y* (known as the relevant subspace). The different strategies for PLS-based variable selection are usually based on a rotation of the standard solution by a manipulation of the PLS weight vector (***w***) or the regression coefficient vector, **b**.

The *VIP* method selects variables by calculating the *VIP* score for each variable and excluding all the variables with *VIP* score below a predefined threshold *u* (usually *u* = 1). All the parameters that provide an increase in the predictive ability of the model are retained[20].

The *VIP* score for the variable *j* is defined as:

$$VIP_j = \sqrt{\frac{p}{\sum_{m=1}^{M} SS(b_m.t_m)} . \sum_{m=1}^{M} w_{mj}^2 . SS(b_m.t_m)}$$

where *p* is the number of variables, *M* the number of retained latent variables, w_{mj} the PLS weight of the *j*-th variable for the *m*-th latent variable and $SS(b_m \cdot \boldsymbol{t}_m)$ is the percentage of *y* explained by the *m-th* latent variable.
The *VIP* value is namely a weighted sum of squares of the PLS weights (*w*), which takes into account the explained variance of each PLS dimension.
The "greater than one" rule is generally used as a criterion for variable selection because the average of squared *VIP* scores is equal to 1.

References

1. Miller, A. Subset selection in regression, 2nd edition. Chapman & Hall/CRC, 2002.
2. Hastie, T., Tibshirani, R., Friedman, J. The elements of statistical learning, 2nd edition. Springer, 2009.
3. Efroymson, M. A. Multiple regression analysis, in Mathematical methods for digital computers. Wiley, New York, 1960.
4. Akaike, H. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21 (1969), 243.
5. Todeschini, R., Consonni, V. Molecular descriptors for chemoinformatics, 2nd edition. Wiley-VCH, 2009.
6. Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading. Kluwer Academic Publishers, Boston, MA, 1989.
7. Leardi, R. Nature-inspired Methods in Chemometrics: Genetic Algorithms and Artificial Neural Networks. Data Handling in Science and Tecnology, 2003.
8. Leardi, R., Boggia R., Terrile M. Genetic algorithms as a strategy for feature selection. J. Chemometrics, 1992.
9. Bledsoe, W.W. The use of biological concepts in the analytical study of systems. Paper presented at ORSA-TIMS National Meeting, San Francisco, 1961.
10. Holland, J. H. Adaptation in natural and artificial systems. MIT Press, 1992.
11. Kim, K. J., Cho S. B. A comprehensive overview of the applications of artificial life. *Artificial Life*, 12 (2006), 153-182.

12. Bliss L. Particle Swarm Optimization. Pace University DPS program, 2002.
13. Shen, Q., Jiang, J. H., Jiao, C. X., Shen, G. L., Yu, R. Q. Modified particle swarm optimization algorithm for variable selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists. *European Journal of Pharmaceutical Sciences*, 22 (2004), 145-152.
14. Goodharzi M., Freitas M. P. and Jensen R. Ant Colony Optimization as a Feature Selection Method in the QSAR Modeling of anti-HIV-1 activities of 3-(3,5-dimethylbenzyl)uracil derivatives using MLR, PLS and SVM regressions. *Chemometrics and Intelligent Laboratory Systems*, 98 (2009), 123-129.
15. Dorigo, M., Blum, C. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344 (2005), 243 – 278.
16. Tibshirani, R. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society B*, 58 (1996), 267-288.
17. Zou and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67 (2005), 301-320.
18. Wold, S. PLS for multivariate linear modeling. Chemometric methods in molecular design, Vol 2. Wiley-VCH, 1995.
19. Mehmood, T., Martens, H., Sæbø, S., Warringer, J., Snipen, L. A Partial Least Squares algorithm for parsimonious variable selection. *Algorithms for Molecular Biology*, 6 (2011), 27.
20. Indahl, U., Liland, K., Næs, T. Canonical partial least squares: A unified PLS approach to classification and regression problems. *Journal of Chemometrics*, 23 (2009), 495–504.
21. Chong, I.G., Jun, C.H. Performance of some variable selection methods when multicollinearity is present. *Chemometrics and Intelligent Laboratory Systems* 78 (2005), 103–112.