Annals of Actuarial Science, Vol. 6, part 2, pp. 307–343. © Institute and Faculty of Actuaries 2012
doi:10.1017/S1748499512000036
First published online 18 May 2012

Computational intelligence with applications to general insurance: a review

I – The role of statistical learning

Pietro Parodi*
Willis Global Solutions (Consulting Group), London

Abstract

This paper argues that most of the problems that actuaries have to deal with in the context of non-life insurance can be usefully cast in the framework of computational intelligence (a.k.a. *artificial* intelligence), the discipline that studies the design of agents which exhibit intelligent behaviour. Finding an adequate framework for actuarial problems has more than a simply theoretical interest: it also allows a knowledge transfer from the computational intelligence discipline to general insurance, wherever techniques have been developed for problems which are common to both contexts. This has already happened in the past (neural networks, clustering, data mining have all found applications to general insurance) but not systematically, with the result that many useful computational intelligence techniques such as sparsity-based regularisation schemes (a technique for feature selection) are virtually unknown to actuaries.

In this first of two papers, we will explore the role of statistical learning in actuarial modelling. We will show that risk costing, which is at the core of pricing, reserving and capital modelling, can be described as a supervised learning problem. Many activities involved in exploratory analysis, such as data mining or feature construction, can be described as unsupervised learning. A comparison of different computational intelligence methods will be carried out, and practical insurance applications (rating factor selection, IBNER analysis) will also be presented.

Keywords

Statistical learning; Machine learning; Intelligent agents; Risk agents; Regularisation; GLM; Neural networks

1. Introduction

This paper aims to show that computational intelligence provides a natural framework for understanding and managing risk in the context of non-life insurance, and possibly – although all examples are taken from non-life insurance – in the context of life insurance, pension and risk management as well. It does this by reviewing relevant computational intelligence methodologies, and illustrating their possible use in non-life insurance.

*Correspondence to: Pietro Parodi, Willis Global Solutions (Consulting Group), Willis Ltd, 51 Lime Street, London EC3 M 7DQ. E-mail: pietro.parodi@willis.com

The use of computational intelligence techniques in non-life insurance is not new (see, e.g., Cummins & Derrig, 1997; Guo, 2003, Lemaire, 1990; Yao, 2008; Shapiro, 2004; Taylor, 2008). This paper claims that this is not incidental: there is a profound analogy between the main problem of computational intelligence – the discipline that studies the design of agents which exhibit intelligent behaviour – and the problem of understanding and managing risk. The reason behind this is that in order to manage risks successfully one also needs to understand the environment where one operates: as an example (Terzopoulos *et al*., 1994), an artificial fish which has been made to live in an artificial aquarium with other artificial fish has to learn from data, make decisions based on limited information and compete for resources based on little certain knowledge of what its competitors will do – much in the same way as an insurer does.

One reason why establishing this connection between non-life insurance and computational intelligence is useful is of a theoretical nature: it leads us to a deeper understanding of the problems we (as actuaries) face. Another reason is more practical and has to do with knowledge transfer: as plenty of work has been done in computational intelligence on problems that are analogous to non-life actuarial problems, many computational intelligence methods will be available off-the-shelf for us to use. A secondary objective of this paper is therefore to provide a review of relevant computational intelligence methods. Some of these methods (e.g. neural networks, Bayesian networks) will be already familiar to many, but others will be new to most actuaries.

The paper is structured in two parts. In the first part ("I. The role of statistical learning"), we will look at the role of statistical learning (a.k.a. machine learning) in risk costing, which may be defined as the problem of estimating the future cost of risk and is an essential part of the three fundamental activities of non-life insurance: pricing, reserving, and capital modelling.

Section 2 ("Learning from data") shows that the problem of risk costing can be cast in the framework of statistical learning theory, the branch of computational modelling which is concerned with selecting and validating models of the environment. More specifically, we will see how risk costing is an example of **supervised learning**. This is most obvious when many variables are involved, such as in rating factors selection; however, it is a general fact that also applies, for example, to the simple problem of fitting a severity distribution to a sample of data points.

Unsupervised learning will also be shown to play an important part in risk costing, as it provides useful methods for discovering hidden patterns in data and has been successfully used in exploratory analysis and data mining (Section 3, "Exploratory analysis").

Section 4 ("Conclusions") draws the partial conclusions of this first part of the investigation.

In the second part of this paper ("II. Dealing with uncertain knowledge"), we will focus on how to deal with uncertain data and knowledge, and how to make decisions in an uncertain environment.

2. Learning from Data

The fundamental actuarial problems of non-life insurance – pricing, reserving and capital modelling – require a risk costing exercise which is (to a varying extent) driven by historical data. *Predictive modelling*, a technique widespread among actuaries, is also an example of risk costing: it attempts to infer from the data the factors that better explain the risk in order to price different policyholders a different amount of money, or to reserve different types of claims differently.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

2.1. Generalities on supervised learning

Data-driven risk costing is a straightforward example of a statistical learning (a.k.a. machine learning) problem. Specifically, it is an example of supervised learning. We will see that statistical learning has a very rigorous way of building models based on data, parametrising them and validating them. Also, it has developed many different techniques for building models, some of which are little known to the actuarial community.

The term "learning" comes from the fact that we are trying to *learn*[1] the features of a model (structure and parameters) based on a sample of inputs and outputs, e.g. in rating factor selection the outputs are the claims and the inputs are the candidate rating factors. The term "supervised" reflects the fact that we know a number of actual outputs (e.g. the actual claims), and these drive our choice of the model. For this reason, supervised learning is also called "learning with a teacher": during the **training stage**, the teacher feeds the model with pairs of inputs and outputs, and then selects a model. During the **testing stage**, the model is checked against new pairs of inputs and outputs.

A more mathematical definition, based on the discussion of statistical decision theory in Hastie *et al.* (2001) (Section 2.4), is given below. According to this definition, supervised learning is a problem of functional optimisation.

Supervised learning *Given: (i) a real valued random input vector* $X \in \mathbb{R}^p$, *(ii) a real valued output vector* $Y \in \mathbb{R}$ *with joint distribution* $Pr(X,Y)$, *(iii) a loss function* $L(Y, f(X))$, *that maps pairs* $(Y, f(X))$ *into positive real numbers, find* $Y = f(X)$ *that minimises the expected prediction error* $\mathbf{EPE}(f) = \boldsymbol{E}(L(Y, f(X)))$.

This definition is helpful but loose: for example, it does not say to which class of function f must belong. In many cases, a countable dictionary of basis functions $\{f_\alpha\}_{\alpha \in N}$ (the "basis functions") will be specified, and f will be required to be written as a linear combination of these basis functions: $f(x_1, x_2, \ldots x_n) = \sum_\alpha a_\alpha f_\alpha(x_1, x_2, \ldots x_n)$, with $\sum_\alpha |a_\alpha|^2 < \infty$. However, other possibilities exist, as we will see later.

As mentioned above, supervised learning is a two-stage process: (i) a training stage during which a model is fitted to the training data, i.e. a sample of pairs $\{(X_i, Y_i),\ i = 1, \ldots K\}$, and (ii) a testing phase, during which the model's goodness-of-fit is assessed against an independent subset $\{(X_i, Y_i),\ i = K + 1, \ldots N\}$ of the data available.

One key issue is, as expected, the choice of the loss function. The most popular loss function for regression is the squared loss $L(Y, f(X)) = \|Y - f(X)\|_{L_2}^2$, where the functional $\|\cdot\|_{L_2}^2$ is the L_2 -norm, defined as $\|f - g\|_{L_2}^2 = \int_\Omega (f(x) - g(x))^2 dx$ for continuous functions and reducing to $\|f - g\|_{l_2}^2 = \sum_{i=1}^\infty (f(x_i) - g(x_i))^2$ for discrete functions. Examples of alternative choices are the l_1 norm, the l_∞ norm, the log P norm, etc.

2.2. Model selection

When selecting a model to interpret historical data, our main objective is to find one which has good (possibly optimal) predictive accuracy when presented with new independent data. It turns out that prediction accuracy is somehow related to the model complexity[2].

[1] Notice the linguistic change: in classical statistics we would have used "estimating" or "calculating" rather than "learning".

[2] Note that there is no universal measure of complexity in the context of learning algorithms. Rather, complexity is problem-specific: for example, the number of parameters in a statistical model, or the *effective* number of parameters in a neural network.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

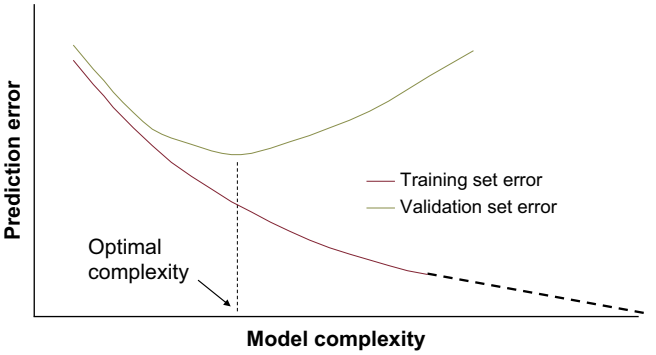

Figure 1. This graph illustrates a general behaviour of learning algorithms: the more complex the model becomes, the lower the training set error (red line). As for the validation set error (green line), it first decreases with complexity, and then picks up again. Model selection aims at striking the right balance – finding the optimal complexity (the minimum in the graph above).

This relationship is illustrated in Figure 1, which shows the expected prediction error on both the training set (training error) and a test set (test error). The training error:

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{f}(x_i)\right) \tag{1}$$

always decreases as the model becomes more complex, as the model adapts to the nuances of the historical data. However, the test error:

$$\text{Err} = \mathbb{E}\left[L\left(Y, \hat{f}(X)\right)\right] \tag{2}$$

is large when the complexity is low (when the model is too simple to capture the regularities of the data), then decreases, and then picks up again (when the model starts overfitting the data). There is a point where the expected prediction error is minimal – the optimal complexity. The idea is that our model will have predictive power if it identifies the genuine regularities in the data, without adding any spurious features.

2.3. Model validation

The expected prediction error also guides the process of *validating* the model. With Solvency II approaching, validation has become a topical issue in insurance. The reader is referred for example to the report of the working party on model validation and monitoring presented at GIRO 2009 by Berry *et al.* (2009) for a review.

The reason why the problem of model validation is so topical is that actuaries are subject, in their daily practice, to the pitfalls of *data snooping*. Data snooping (a.k.a. data dredging, or data fishing) is what happens when a data set is used to discover regularities, and the same data set is used to test the statistical significance of these regularities. This is a particularly important concern in disciplines where one makes intensive use of data mining (data mining will be mentioned more extensively in Section 3), e.g in medical research (see for example the papers White, 2000 and by Ioannidis, 2005).

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

That this is a common problem in actuarial practice may be realised by the simple example of determining the severity distribution that fits a number of losses. If this is done without a clear hypothesis to test (e.g., that the severity distribution be a lognormal) one ends up choosing that severity distribution among those on offer in a distribution-fitting tool which happens to conform to the data in the closest fashion, perhaps because it has the lowest Kolmogorov-Smirnov distance (or any other statistical criterion). However, this closeness has typically no predictive value on future losses, since by choosing among, say, 20 distributions, some of which have 3 or 4 parameters, we are almost always bound to find a good fit by sheer chance. In other words, we have discovered regularities that are not really there.

Statistical learning provides the ideal framework to deal with data snooping as rigorously as possible, in that it provides clear protocols for model validation, by separating the process by which the model is chosen from the process by which the model is validated.

The first step in understanding how this can be done is recognising that the only genuine and general way of validating the model is against data points that have not been used during the learning process: in the severity distribution example above, the only way of gaining confidence in the fact that we have selected the correct distribution is to make sure we test our severity model against losses that have not been used to choose the model in the first place.

In general (and ideally) one should divide the database randomly into three data sets: the **training set**, used to fit and parameterise all the candidate models; the **validation set**, used to estimate the expected prediction error on all candidate models, with the purpose of selecting the model with the lowest expected prediction error; the **test set**, used to estimate the expected prediction error on the "best" model as selected in the previous step. This is a rigorous and general method, and has the advantage of fully decoupling model selection and model validation. For the process to be rigorous, the test set should be used only once, during the final estimation of the prediction error.

The three-sets method described above is fine in a data-rich situation, but may be unworkable where data are scant. For this reason, approximate alternative methods to calculate the expected prediction error EPE(f) are often used. Two of them, K-fold cross-validation and information criteria, are explained below. More information is available in Hastie *et al.* (2001, Chapter 7).

Cross-validation. Cross-validation, or more specifically K-fold cross-validation, estimates the extra-sample test error by dividing *at random* the data set into K different subsets. Each subset $k = 1, \ldots K$ is in turn removed from the data set and the model is fitted to the remaining K-1 subsets. The k-th subset is used as a test set. The process is repeated for all subsets and the cross-validation estimate of the prediction error is given by

$$CV = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{f}^{\kappa(i)}(x_i)\right)$$

where $\kappa : \{1, \ldots N\} \mapsto (1, \ldots K)$ is an indexing function mapping the partition to which observation i is allocated by the randomisation, and $\hat{f}^{\kappa(i)}(x_i)$ is the fitted function calculated with the $k(i)$ -th subset removed.

For model selection purposes, we do not have a single model to cross-validate but a set of different models $f(x;\alpha)$ indexed by a parameter α, which may represent for example the complexity of

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

the model. For this set of models, we have a different value of CV(α) for every α, and the function CV(α) is effectively an estimate of the test error curve depicted in Figure 1. The optimal model is then $f(x;\alpha^*)$, where α^* is the value that minimises CV(α).

One key issue with cross-validation is of course the choice of K. Anything from $K = 2$ to $K = N$ is possible. Typical values are $K = 5$ and $K = 10$.

Analytical criteria. The use of the three-set protocol and cross-validation is not widespread among actuaries, and usually limited to practitioners in personal lines insurance. One reason is that many methods that actuaries use for feature selection come with built-in mechanisms to avoid using too many features.

One such mechanism is the Akaike Information Criterion (AIC), which applies to maximum likelihood estimators: adding one parameter to a statistical model can only be justified by a gain in the log-likelihood at least equal to two. Several other analytical approaches are also possible. All of these methods punish complexity, playing the same role as regularisation, and none of them requires validating the model on a different set.

For a better understanding of how these methods work, notice that beside the training error (Equation (1)) and the test error, a.k.a. extra-sample error (Equation (2)), we can define a third type of error, the **in-sample error**, which gives the distance between the model and an independent sample which shares with the original sample the points at which the model is evaluated. The idea is that part of the discrepancy between the model and the test sample is due to the model being poor, and part is due to the fact that in general the points at which the model is calculated differ from those of the training sample. The in-sample error is defined as

$$\mathrm{Err}_{in} = \sum_{i=1}^{N} \mathbb{E}_y \mathbb{E}_{Y^{\mathrm{new}}} L\left(Y_i^{\mathrm{new}}, \hat{f}(x_i)\right)$$

where the x_i's are the training points and Y^{new} is the new response at the same points.

The in-sample error is not a completely satisfactory estimate of model error, and therefore is not completely satisfactory for model validation purposes. However, it usually does a good job for model selection purposes, where the main issue is not the absolute size of the error but how the error of one model compares to the other.

The *Akaike information* is an estimate $\widehat{\mathrm{Err}}_{\mathrm{in}}$ of Err_{in} that can be used in the case of a log-likelihood loss function (as is the case for GLM). It can be shown that the following holds:

$$\widehat{\mathrm{Err}}_{\mathrm{in}} \propto \mathrm{AIC} = -\frac{2}{N}\mathrm{loglik} + 2\frac{d}{N}$$

asymptotically as $\mathrm{N} \to \infty$. As usual, loglik in the equation above is the maximised log-likelihood. The Akaike Information Criterion for model selection prescribes to choose the model with the smallest AIC over the set of models considered[3].

[3] A misconception about AIC is that it only serves to discriminate between nested models. This misconception arose because Akaike himself stated that that was the case in his original paper. He later showed that his criterion applied to any pair of models, but the misconception lingered on.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

A modified version of the AIC, denoted AICc, has been proposed to deal with the case when the number of data points N is small or the number of parameters d is large:

$$\text{AICc} = \frac{1}{N}\left(-2\,\text{loglik} + 2\,d + \frac{2\,d(d+1)}{N-d-1}\right)$$

Using this criterion – which obviously converges to AIC when $N \to \infty$ – has been shown to decrease the probability of overfitting. This is also achieved by the Bayesian Information Criterion below.

The *Bayesian Information Criterion (BIC)* has a similar form to AIC:

$$\text{BIC} = -2\,\text{loglik} + d\log N$$

Since the factor 2 is replaced with $\log N$, when the number of data points is 8 or more the penalty for more complex models is larger for BIC for $N > 8$. It arises in the context of Bayesian model selection, Bishop (2007), as the posterior probability of a model $\mathcal{M}_r$ given a choice among R possible models and training data Z is

$$Pr(\mathcal{M}_r|Z) = \frac{e^{-\frac{1}{2}\text{BIC}_r}}{\sum_{r=1}^{R} e^{-\frac{1}{2}\text{BIC}_r}}$$

Other estimates of the in-sample error can be obtained using the *Minimum Description Length (MDL)* criterion (Grunwald *et al.*, 2005), which is formally identical to BIC, and the bootstrap (Hastie *et al.*, 2001).

Comparison of selection/validation methods. We now discuss how the different selection/validation methods compare to one another.

- AIC and BIC are less general than cross-validation. They can be used in their basic form only for the log-likelihood loss function and when the number of degrees of freedom is defined. For certain nonlinear models, such as neural networks, the number of degrees of freedom can sometimes be replaced with the effective number of parameters (see Hastie *et al.*, 2001). However, for complex models this soon becomes very difficult and AIC, BIC become impractical:
- the AIC and BIC tend to overestimate the test error by a much larger amount than cross validation. This may be a problem in model selection (but only if this changes the relative order of models) and is certainly a problem in model validation, because the test error is exaggerated;
- BIC v AIC: BIC is asymptotically correct, in the sense that as the number N of data points tends to infinity, the true model is selected with probability 1. AIC does not have this property, and it tends to favour more complex models when N becomes very large. At the same time, BIC tends to over-punish complexity for finite samples and pick models that are too simple;
- BIC, AIC are simpler to calculate than cross-validation, as they do not require splitting the data into subsets and calculating the parameters many times.

Cross-validation therefore appears to be a far more general and more rigorous methodology, which however often requires more discipline and extra work.

2.4. Efficiency of model selection/validation methods

A good model selection algorithm should be able to determine what the best model is and what its parameters are in an efficient way – i.e. with low computational complexity.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

The brute-force approach to best subset selection, which simply looks all possible combinations of features to choose those to include in the model, is computationally intractable (see Papadimitriou, 1994), as the time it takes to perform the selection increases exponentially with the number of features ("combinatorial explosion").

There are two typical approaches to break intractability:

- Use *greedy algorithms*: that is, start with a very basic model (for example one that includes no features) and then start adding the features one by one, each time selecting the features that give the most impressive results, e.g. the one that explains most of the variance. This is the approach which is probably the most familiar to actuaries. The problem with this approach is that there is no guarantee that a global minimum (a truly optimal subset of features) will be reached: the order by which we add new features matters (especially when the features are correlated).
- An alternative to greedy algorithms are the *sparsity-based regularisation schemes* such as the lasso or elastic net, which will be introduced in Section 4.2. These work by adding a penalty term to the loss function, thus creating a regularised loss function. The size of the penalty term is controlled by a multiplicative parameter. By tuning this parameter, a different subset of features is automatically selected. This is called continuous subset selection and in many cases can be performed quite efficiently, as we will see in Section 2.6.

We now look into the most widespread techniques for feature selection, starting from that which is most popular among actuaries, generalised linear modelling. Regularisation theory and neural networks will also be described, and a comparison between the different methods will then be attempted.

2.5. Model selection by GLM with forward/backward selection

GLM is an extension of the linear model, in which the outputs Y are related to the inputs $X = (X_1, \dots X_n)$ through a matrix of coefficients β, with the addition of *Gaussian* noise ε which is usually assumed to be independent of X : $Y = X \cdot \beta + \varepsilon$. The extension works by: (i) replacing the simple linear combination of inputs with a *linear* combination of a wider dictionary of functions $\psi_j(X)$, e.g. X_1, $X_1X_2, \log(X_1 + X_2), \dots$; (ii) replacing the Gaussian noise with a more general type of noise, belonging to the so-called exponential family, which includes among the others Gaussian, binomial, Poisson and Gamma noise and is dependent on X; (iii) introducing a further link function g which transforms the responses so as to enforce certain constraints on the output (e.g. positivity). This yields the general form of a GLM:

$$Y = g^{-1}\left(\sum_j \beta_j \cdot \psi_j(X_1, \dots X_n) + \varepsilon\right)$$

The parameters of a given model $f(X) = \sum_j \beta_j \cdot \psi_j(X_1, \dots X_n)$ can be calculated by maximum likelihood estimation, or in other words by minimising a log-likelihood loss function:

$$L(Y, f(X)) = -2 \log \Pr_{f(X)}(Y)$$

where the factor "2" has been introduced so as to reduce to the standard squared loss in the case of Gaussian noise.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

As an example, claims frequency is usually modelled by a GLM with a Poisson link function and Poisson noise, whereas claims amounts are often modelled using Gamma noise. See Modlin *et al.* (2004) for a more thorough discussion.

The specific model for a set of data is usually selected using a greedy approach. This comes in two varieties, forward selection and backward selection. Forward selection starts from the simplest model, $Y =$ constant, and adds at each step the function $\psi_j\,(X_1,\ldots X_n)$ that reduces the loss function by the largest degree[4]. Backward selection starts from the most complex model and removes functions one by one: however, this method can only be applied when the dictionary of functions is finite.

To decide whether the selected additional function in forward selection is actually an improvement, a typical test is to check whether the AIC decreases. A simplified, less rigorous test is whether the coefficient of the newly introduced function is significantly different from zero.

2.6. Model selection by regularised regression

The main idea of regularisation as a tool for model selection is that minimising the functional $EPE(f) = \mathbb{E}(L(Y, f(X)))$ on an independent sample is equivalent to (or rather, can be approximated by) minimising a different regularised functional on the training sample:

$$\hat{f} = \text{argmin}_\beta \left\{ \mathbb{E}(L(Y, f(X))) + \lambda g_\beta(X) \right\} \tag{3}$$

The term $g_\beta(X)$ is called a penalty term and embeds our prior knowledge on what the desirable features of the parameters are. The most famous example is ridge regression (or Tychonov regularisation), in which[5] $g_\beta(X) = \|\beta\|^2_{l^2}$: in this case the idea is that coefficients should be as small as possible.

Although theoretically any loss function could be used in Equation (3), in practice quadratic loss is usually adopted when modelling quantitative (rather than categorical) responses, and most computational methods to derive solutions have been devised for the quadratic loss case.

Table 1 shows some examples of regularisation schemes. **Ridge regression** has a long history, dating back at least to Tychonoff's work on ill-posed problems (Tychonoff, 1943), and is probably the best known to actuaries. The reasoning behind this is that in the case of correlated variables, solutions where one large coefficient is compensated by another large coefficient of opposite sign for a correlated variable are common. The l^2 penalty term tends to shrink the coefficients, keeping this behaviour at bay, and therefore preventing the inclusion of spurious factors.

One of the advantages of ridge regression was that an analytical solution is available (see Table 1). As computing power became more easily available, it was possible to experiment with other types of penalty, achieving more sophisticated effects. In 1996 Tibshirani introduced the **lasso**, a simple modification of Tychonoff's regularisation scheme where the l_2 penalty term was replaced

[4] Assuming that the dictionary of functions is finite. When the dictionary is (infinite) countable, we need to impose constraints on how we add functions. For example we may demand that at each step we only add one extra variable of the n variables allowed.

[5] The l_2 norm is defined as the sum of squared differences over a Hilbert space: $a-b^2_{l_2} = \sum_{i=1}^{\infty} (a_i-b_i)^2$, and reduces to the standard Euclidean distance when the number of components is finite. The l_1 norm is defined similarly as $a-b_{l_1} = \sum_{i=1}^{\infty} |a_i-b_i|$.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

Table 1. Examples of regularisation schemes.

Regularisation scheme	Regularised functional	Notes
Non-regularised	$\|Y-f_\beta(X)\|_{l_2}^2$	• The solution is the standard least square solution: $\hat{\beta} = (X^T X)^{-1} X^T y$ (a linear function of the observations)
Ridge regression *(Tychonoff, 1943)*	$\|Y-f_\beta(X)\|_{l_2}^2 + \lambda\|\beta\|_{l_2}^2$	• Performs coefficient shrinkage • It has a simple analytical solution: $\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$
Lasso *(Tibshirani, 1996)*	$\|Y-f_\beta(X)\|_{l_2}^2 + \lambda\|\beta\|_{l_1}$	• Note that the l^1 norm (sum of absolute values) is used in the penalty term • Performs **factor selection** *and* coefficient shrinkage • There is no analytical solution: the solution can be obtained by quadratic programming, or by least angle regression (Efron *et al.*, 2004), which takes the same time to compute as a single least squares calculation • Has problems when there are groups of variables with strong correlations – only one variable from each group is often selected in this case
Elastic net *(Zou & Hastie, 2005)*	$\|Y-f_\beta(X)\|_{l_2}^2 + \lambda\|\beta\|_{l_2}^2 + \mu\|\beta\|_{l_1}$	• Performs factor selection *and* coefficient shrinkage • No analytical solutions • Can address the case where the number of predictors is >> of the number of data points • Can also address the problems with correlation faced by the lasso

with an l_1 term. The loss of the nice analytical properties of the ridge regression is compensated by the fact that the lasso has the (perhaps surprising) effect of enforcing *sparsity*, i.e. forcing the coefficients of the least important factors to zero and by that keeping the model simple. The underlying reason for this counterintuitive effect is illustrated in Figure 2.

The most significant drawbacks of the lasso are that (Zou & Hastie, 2005) (i) when the number p of predictors is larger than the number N of observations, the lasso selects at most N variables then saturates; (ii) if there is a group of variables among which the pairwise correlations are very high, the lasso tends to select only one variable from the group; (iii) when large pairwise correlations are involved, the lasso has been shown to perform worse than ridge regression.

These are burning issues in some contexts, such as DNA microarray data analysis, which attempts to find the combinations of genes responsible for a disease. In this case, the number of predictors (genes) is typically larger than the number of observations (DNA samples). Furthermore, some of the genes are highly correlated as they are part of small interacting networks which have to be identified fully to understand the underlying biological mechanism. This has provided the motivation to Zou & Hastie (2005) to come up with a regularisation scheme which would overcome lasso's shortcomings while preserving its sparsity-enforcing properties. The results of this

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

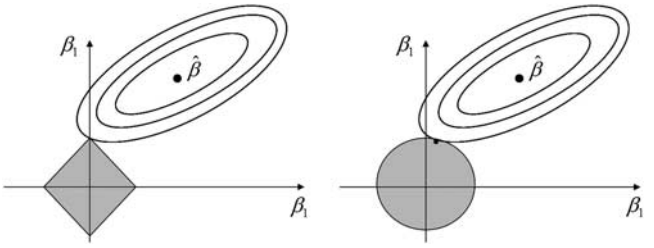

Figure 2. In the figure above, the ellipses represent points for which $\|Y-f_\beta(X)\|_{l_2}^2$ is constant, whereas the geometric figure around the (0,0) point represents the set of points for which $\|\beta\|_{l_1} < s$ (lasso, left) or $\|\beta\|_{l_2} < s$ (ridge regression, right). The minimisation of the regularised functional is equivalent to the problem of finding β such that $\|Y-f_\beta(X)\|_{l_2}^2$ is minimal, subject to the condition that $\|\beta\|_{l_1} < s$ (or $\|\beta\|_{l_2} < s$). This will occur at a point for which the ellipsis just touches the circle/rhombus. For ridge regression, this intersection point can be anywhere. In the case of the lasso, the intersection is likely to be at one of the vertices for the lasso. This figure is taken from Hastie *et al.* (2001, p. 71).

effort were the *elastic net*, which is effectively a hybrid of ridge regression and the lasso and uses a composite l_2/l_1 penalty term (see Table 1). De Mol *et al.* (2009) provided mathematical foundation for this regularisation scheme, showing that the elastic net solution can be interpreted as the fixed point of a contraction in a Banach space, and that under suitable conditions the elastic net estimator is consistent for both prediction and variable selection purposes.

Bayesian interpretation. It should be noted that all the regularisation schemes mentioned above are liable to a simple Bayesian interpretation. In this framework, we are looking for the values of the parameters that maximise the posterior probability given the data, and the penalty term corresponds to the prior probability on the coefficients. The ridge regression penalty term corresponds to a multivariate normal prior probability centred around zero: $\Pr(\beta) \propto \exp(-\lambda\|\beta\|_{l_2}^2)$; the lasso regression, to a Laplace distribution: $Pr(\beta) \propto \exp(-\mu\|\beta\|_{l_1})$; the elastic net to the distribution $\Pr(\beta) \propto \exp(-\lambda\|\beta\|_{l_2}^2 + \mu\|\beta\|_{l_1})$. This sheds further light on the nature of regularisation, which is ultimately the incorporation of prior knowledge of the model into the fitting process. The ridge regression reflects the belief that the parameters are small; the lasso reflects the belief that the model is sparse; the elastic net reflects both beliefs.

Applications to general insurance. The lasso provides a simple alternative to the "GLM with forward/backward feature selection" paradigm commonly used in insurance. This way is both rigorous and efficient. As to the drawbacks listed above, the "large p, small N scenario" is usually not a concern, at least for personal line insurance applications, but it may become so in applications with sparse data (see Section 2.7 below).

The deterioration of prediction performance in the $N > p$ scenario when highly correlated variables are present is instead something that is quite applicable in our context, as many variables that are considered during the underwriting process, such as the age of the driver and whether the policy purchased is comprehensive or third-party, fire and theft (TPFT) liability only are strongly correlated. An elastic net regularisation scheme might therefore be more adequate in many contexts. The presence of small cliques of predictors that need to be preserved, however, seems to be more specific to biology: there seems no harm in selecting any of a group of correlated variables as long as the overall performance is good.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

2.7. Model selection by neural networks

Neural networks were initially developed as simple models for the human brain. The basic units of the network correspond to the neurons, and the connections between these units correspond to the synapses. This provided the inspiration for much research into artificial intelligence, in the hope to emulate the desirable characteristics of the brain, such as Hertz *et al.* (1991): its robustness and fault tolerance; its flexibility and ability to learn; its ability to deal with fuzzy, noisy or inconsistent information.

Research on artificial neural networks has gone in waves. The first wave can be traced back to a work by McCulloch & Pitts (1943), and has culminated in a number of works by Rosenblatt (1958) in the 1960s on perceptrons, which were networks of neurons organised in layers, such as that in Figure 3, with weights associated with each connection. The first generation of perceptrons – which were single-layered – were found not to be able to perform some elementary computations (Minsky & Papert (1969), and research stagnated, until in the mid-1980s the back-propagation algorithm (an example of the gradient descent method) for updating the weights of a multi-layered neural network was discovered by Rumelhart *et al.* (1987). Neural networks have ever since been applied to a wide range of problems, from computer vision to particle physics, and as John Denker, an aviation scientist, once famously remarked, they seem to provide "*the second best way of doing just about anything*". Many researchers, however, have found it difficult to accept this paradigm, as the solutions it provides are not transparent - neural networks work, but it is difficult to understand why!

As Hastie *et al.* (2001) observe – rejecting both the hype and an outright rejection, neural networks are simply examples of non-linear statistical models, much like project pursuit regression.

Neural networks come in many flavours, the most classical of which is the single hidden layer, feed-forward neural network shown in Figure 3. The single hidden layer, feed-forward neural network shown there can be used for both regression and classification, although for regression there is only one output (whereas for classification more outputs are needed, the number K of units at the top typically being equal to the number of classes.

The units at the bottom layer are called either features or simply inputs; those in the hidden layer are called hidden units or derived features; and those of the top layer are called either outputs or responses.

Let us outline the main ingredients of the neural network model of Figure 3 and the equations regulating its behaviour:

- Inputs: $X_1, \ldots X_p$
- Derived features: $Z_1, \ldots Z_M$, where $Z_m = \sigma(\alpha_{0m} + \alpha_m^T X)$, $m = 1, \ldots M$. In this equation σ is the activation function. The most popular choice for the activation function is the sigmoid function, $\sigma(v) = (1 + \exp(-v))^{-1}$

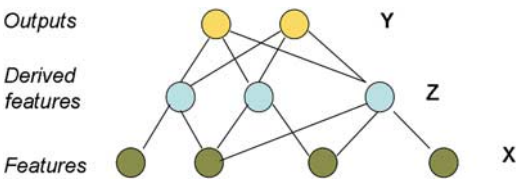

Figure 3. A single hidden layer, feed-forward neural network.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

- Outputs: $Y_1, \ldots Y_K$, where $Y_k = f_k(X) = g_k(\beta_{0k} + \beta_k^T Z)$, $k = 1, \ldots K$. Typically, $g(T) = T$ (identity function) for regression problems.

Solving a problem with a neural network means finding the correct weights $\theta = \{\alpha_{0m}, \alpha_m | m = 1, 2 \ldots M; \beta_{0k}, \beta_k | k = 1, 2 \ldots K\}$ to use for a specific problem. The parameters are learned from the data during the training stage, by minimising a suitable loss functions. For regression, this is normally the sum of squares:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} \left(y_{ik} - f_k(x_i)\right)^2$$

The minimisation can be performed for example by back-propagation, which is an application of dynamic programming.

A key issue with neural networks is that since there are usually many weights there is a danger of overfitting, by adapting too closely to the nuances of the data. To avoid this one can introduce a penalty term and find a regularised solution, for example by minimising $R(\theta) + \lambda J(\theta)$ instead of $R(\theta)$, where $J(\theta) = \sum_{km} \beta_{km}^2 + \sum_{ml} \alpha_{ml}^2$. This form of regularisation is called *weights decay* and is analogous to ridge regression. Alternatively, one can use (as it used to be in the early days of neural networks) an early stopping rule, by which the training of the model is stopped way before a global minimum is reached.

It should be noted that neural networks are very general and that when they include hidden layers they can express any function[6] (see the list of examples in Lowe & Pryor, 1996). Furthermore, they can be validated like any other method and should not therefore be rejected on the grounds that their results are not reliable.

There is, however, another important reason why neural networks are not suitable for many applications, and it is their lack of transparency: it is difficult to interpret the results of these models. Neural networks "are effective where prediction without interpretation is the goal" (Hastie *et al.*, 2001) – less so when it is important to understand the roles of individual inputs. E.g., they will determine how much a specific customer should be charged, but it will not be obvious what characteristics of the customer have led to that decision. However, the extraction of explanations from neural networks is the topic of active research (see, e.g., Jacobsson (2006), and the very brief discussion of how some authors address the interpretation problem in Hastie *et al.*, 2001).

Applications to insurance. In certain circumstances, the opacity of neural networks may be an advantage as a detailed model specification is not required.

However, the fact that it is difficult to interpret the results of neural networks is certainly a problem in general because they are not going to help us with *understanding* risk. It is therefore difficult to see how underwriters, managers at an insurance company or the regulator could accept lightly pricing, reserving, capital modelling decisions based on neural networks. Furthermore, they are computationally very demanding.

Despite this problem, some applications of neural networks to general insurance have been attempted, for example by Dugas *et al.* (2003), which investigated their use for motor rating in North America. Furthermore, neural networks are routinely used for exploratory analysis as part of data mining and in other financial contexts such as for credit card fraud detection.

[6] This is a weaker statement than stating that neural networks can *learn* any function.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

Let us now present a simple specific example from general insurance to make it easier to follow the subsequent sections, and go through some general considerations regarding how models should be selected and validated.

2.8. Other methods for supervised learning

This section is a quick foray into a number of more advanced statistical learning techniques, which could be loosely grouped under the header of "kernel methods". The reader who is keen to explore the topic in more depth is referred to the books by Bishop (2007), Hastie *et al.* (2001) and MacKay (2003). This section follows mainly Bishop (2007) (chapters 6 and 7) in the way the material is presented. As usual, we have focused on the problem of regression, although all of these methods can be equally well (and sometimes more naturally, as is the case of support vector machines) used for classification purposes.

Kernel methods

Kernel methods are based on the observation that for many learning algorithms the predicted output is a function of the features (basis functions) $\psi_j(\boldsymbol{x})$ built upon the input vector $\boldsymbol{x}$ only through the scalar product $\boldsymbol{\psi}^T(\boldsymbol{x})\boldsymbol{\psi}(\boldsymbol{x}')$. The idea then arises to replace such scalar product with another function $k(\boldsymbol{x},\boldsymbol{x}')$ of the inputs which exhibits some desired properties, e.g. to depend only on the distance $\|\boldsymbol{x}-\boldsymbol{x}'\|$ between $\boldsymbol{x}$ and $\boldsymbol{x}'$. Commonly used kernels are:

- $k(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{x}^T\boldsymbol{x}'$ (linear kernel)
- $k(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{\psi}^T(\boldsymbol{x})\boldsymbol{\psi}(\boldsymbol{x}')$ (linear kernel – linear in the features space)
- $k(\boldsymbol{x},\boldsymbol{x}') = k(\boldsymbol{x}-\boldsymbol{x}')$ (stationary kernel)
- $k(\boldsymbol{x},\boldsymbol{x}') = k(\|\boldsymbol{x}-\boldsymbol{x}'\|)$ (homogeneous kernel, or radial basis function)
- $k(\boldsymbol{x},\boldsymbol{x}') = \exp(-\|\boldsymbol{x}-\boldsymbol{x}'\|^2/2\sigma^2)$ (Gaussian kernel – an example of a homogenous kernel)

There are many ways to construct valid kernels (see Bishop (2007) for a review) and one crucial aspect of this is that kernels can be defined over such disparate objects as sets and graphs.

To see with a simple example how kernels arise in practice, consider a least-squares problem solved with ridge regression, which involves minimising the regularised error function

$$J(\boldsymbol{\beta}) = \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{\beta}^T\boldsymbol{\psi}(\boldsymbol{x}_n)-t_n)^2 + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta}$$

Given a new input $\boldsymbol{x}$, the predicted output for such model is

$$y(\boldsymbol{x}) = \boldsymbol{\beta}^T\boldsymbol{\psi}(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x})^T(\boldsymbol{K} + \lambda\boldsymbol{I}_N)^{-1}\boldsymbol{t}$$

where $\boldsymbol{K}$ is the Gram matrix $\boldsymbol{K} = \boldsymbol{\Psi}^T\boldsymbol{\Psi}$, an $N \times N$ matrix with elements

$$K_{n,m} = \boldsymbol{\psi}(\boldsymbol{x}_n)^T\boldsymbol{\psi}(\boldsymbol{x}_m) = k(\boldsymbol{x}_n,\boldsymbol{x}_m) \tag{4}$$

and $\boldsymbol{k}(\boldsymbol{x})$ is the vector whose elements are $\boldsymbol{k}(\boldsymbol{x}_n,\boldsymbol{x})$. Therefore, the least-squares solution is expressed solely in terms of the kernel function k and the vectors and matrices built upon it. As a result, it is possible to replace the linear kernel with another *valid* kernel (i.e. a kernel for which the Gram matrix is positive semidefinite) and the solution will still be in the format above. Note that this also makes it unnecessary to introduce basis vectors $\{\psi_\alpha\}$ explicitly.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

Gaussian processes

Gaussian processes can be viewed as an application of kernel models: for a Gaussian process, the kernel is simply the covariance function of the predictor. However, Gaussian processes have a more general interest in that they represent a different approach to supervised learning based on a Bayesian framework, and therefore deserve to be given some space on their own.

The ideas around Gaussian processes have been around for a long time and in such diverse fields as geophysics (where Gaussian process regression is called kriging) and control theory (the Kalman filter can be seen as an example of Gaussian process). The most comprehensive text on Gaussian processes is the book by Rasmussen & Williams (2006). More succinct treatments can be found in MacKay (2003) and in Bishop (2007), this paper combines the material in these books.

We saw in Section 2.6 that many regularisation schemes are liable to a simple Bayesian interpretation: a particular choice of the penalty function can be viewed as the imposition of a prior distribution on the parameters of the model in the form $y(\boldsymbol{x}, \boldsymbol{\beta}) = \boldsymbol{\beta}^T \boldsymbol{\psi}(\boldsymbol{x})$ (a linear combination of basis functions). The prior distribution incorporates our prior expectations on the model, e.g. that the coefficients $\boldsymbol{\beta}$ be as small as possible, or that the model is as sparse as possible (fewer basis functions involved).

More generally, in a Bayesian framework the problem of supervised regression can be reformulated as the problem of inferring the true function $y(x)$ based on a set of noisy observations $\{t_1, \ldots t_N\}$ corresponding to input data points $\{x_1, \ldots x_N\}: t_k = y(x_k) + \varepsilon_k$ where ε_k is random noise. A prior distribution $\Pr(y(x))$ is defined *directly over the space of*functions. Upon presentation of a new input x, we can calculate the predictive distribution for $y(x)$ conditional on the data:

$$Pr(y(x)|t_1, \ldots t_N, x_1, \ldots x_N) = \frac{Pr(t_1, \ldots t_N|y(x), x_1, \ldots x_N)\Pr(y(x))}{\Pr(t_1, \ldots t_N|x_1, \ldots x_N)} \tag{5}$$

Naturally if our model is parametric, e.g. $y(\boldsymbol{x}, \boldsymbol{\beta}) = \boldsymbol{\beta}^T \boldsymbol{\psi}(\boldsymbol{x})$, the prior distribution on $y(x)$ will simply correspond to a prior distribution $\Pr(\beta)$ on the coefficients β. However, the inference process described above is more general and also holds when $\Pr(y(x))$ is defined on the space of functions directly.

Gaussian processes can now simply be defined as a way of associating a probability distribution $\Pr(y(x))$ over functions $y(x)$ so that the vector $\{y(x_1), \ldots y(x_N)\}$ of $y(x)$ evaluated at an arbitrary set of points $x_1, \ldots x_N$ follows a joint Gaussian distribution which is (by definition) completely specified by its mean and covariance matrix. Since we are working in the space of functions, both the mean and the covariance are themselves functions: specifically, the covariance $C(x, x')$ is a function which expresses the covariance between the values of the function y at points x and x'.

Let us now go back to our supervised regression problem $t_k = y(x_k) + \varepsilon_k$. In a Gaussian process, the error ε_k is itself Gaussian: $\varepsilon \sim N(0, \beta^{-1})$, where β represents the "precision" of the noise. The joint distribution on the data points $t_1, \ldots t_N$ conditioned on $x_1, \ldots x_N$ is again Gaussian and is given by $t \sim N(0, C)$ where

$$C(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm} \tag{6}$$

The reason why there are two terms in this covariance matrix is that there are two sources of randomness which are independent: first of all $y(x)$ is a random element in the space of functions

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

(distributed according to a Gaussian distribution by definition) and that of the random noise. Since the two sources of randomness are independent, their covariances simply add up.

The interesting term in equation (6) is the kernel $k(x_n, x_m)$, which is the Gram matrix for the noiseless version of the Gaussian process ($\varepsilon = 0$). For a parametric model using basis functions, e.g. $y(x,\beta) = \beta^{\mathrm{T}}\psi(x)$, the kernel $k(x_n, x_m)$ is given by equation (4). However, we can use the kernel trick and replace it with, e.g., a Gaussian kernel $k(x, x') = \exp(-\|x-x'\|^2/\sigma^2)$ or an exponential kernel $k(x, x') = \exp(-\theta\,\|x-x'\|)$, obtaining Gaussian processes that do not use basis functions.

We are now in a position to calculate the predictive distribution of t_{N+1} for a new point x given the observations $(x_1, t_1), \ldots (x_N, t_N)$, which is what we set out to do in the first place and for which we have already written down the generic solution in the Bayesian context, equation (5).

It is easy to prove that the conditional distribution $p(y(x)|t_1, \ldots t_N)$ is a Gaussian distribution with mean and covariance given by

$$m(x) = k^T C^{-1} t$$

$$\sigma^2(x) = c - k^T C^{-1} k$$

where

$$c = k(x, x) + \beta^{-1}$$

and: C is the matrix whose elements are given by equation (6), k a vector with elements $k(x_k, x)$ for $k = 1, \ldots N$.

Note that the formulae for the mean and covariance above define the predictive distribution for Gaussian process regression with an arbitrary kernel, the only constraint being that the covariance matrix be positive definite (not semidefinite). For this to be true it is sufficient that $k(x_n, x_m)$ be positive semidefinite since the eigenvalues of C are $\lambda_i + \beta^{-1}$ with $\lambda_i \geq 0, \beta > 0$.

The advantage of such an elegant model as Gaussian process regression is that the output of the regression is not just a number but a full predictive distribution, and the definition of a prior forces analysts to make explicit all the expectations that they have on the types of function that can be fitted to the data (or, expressing this more neutrally, to make explicit all the desired properties of the regression function).

One disadvantage (apart from the obvious limitation that Gaussian processes must be, by definition, Gaussian) is that solving Gaussian processes regression normally involves inverting an $N \times N$ matrix (which takes $O(N^3)$ steps) and this is to be compared with the usual basis function model with M basis functions, which takes $O(M^3)$ steps. Typically $M << N$ (the number of model parameters is far smaller than the number of training points) and therefore it is more efficient to deal with basis functions. However, it should be mentioned that (i) what seems like a disadvantage becomes an advantage in the case where the number of basis functions is very large, or actually infinite, and (ii) approximate solutions to the regression problem which are more efficient are available (see Bishop (2007) for a list of references).

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

Sparse kernel machines

One of the problems of non-linear kernel algorithms is that $k(x_n, x_m)$ needs to be evaluated for each pair of training points, and this may be computationally demanding for large numbers of data points.

A possible way of increasing efficiency is using algorithms for which the solution of the learning problem does not depend on the whole training set but only on a subset of the training points. An example is given by **support vector machines** (SVM), which owe their name to the fact that the solution depends only on a small number of training points, which are called "support vectors". A good introduction to support vector machines (and relevance vector machines, which are their Bayesian counterpart) is provided in Bishop (2007).

SVM for regression is best approached by an example. Let us start by reminding ourselves how linear regression works. In linear regression, the solution to a learning problem is obtained by maximising an error function of the form

$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}\|w\|^2$$

In this formulation, the value of the parameters that minimise the error function will depend on all data points.

Now, let us replace the quadratic term $\sum_{n=1}^{N}\{y_n - t_n\}^2$ with a function such as

$$E_\epsilon(y(x) - t)) = \begin{cases} 0 & \text{if} \left|y(x) - t\right| < \epsilon \\ \left|y(x) - t\right| - \epsilon & \text{otherwise} \end{cases}.$$

This has the property of being zero when the difference between the model and the target values is smaller than ϵ. It is easy to convince oneself that when minimising the error function

$$C\sum_{n=1}^{N} E_\epsilon(y(x_n) - t)) + \frac{1}{2}||w||^2$$

the training points that are within a distance of ϵ from a given model $y(x)$ do not contribute to the error and therefore to the predictions (however, there is a subtlety: it is not possible to say a priori which points can be discarded until the model is actually calculated!). Note how in the error function above the parameter C works as a regularisation parameter, which needs to be determined through cross-validation.

Formally, one finds (see Bishop, 2007) that the prediction for a new input $\boldsymbol{x}$ is

$$y(x) = \sum_{n=1}^{N} q_n k(x, x_n) + b \tag{7}$$

where q_n is zero for all points x_n for which $\left|y(x_n) - t\right| < \epsilon$, and $k(x,x_n)$ is given by (4). All the other points x_n are the support vectors of the prediction, and are the only ones for which $k(x,x_n)$ needs to be calculated. As usual, one can perform the kernel trick by replacing the standard form (4) with another positive definite kernel with the desired properties.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

The idea of support vector machines has been developed further by the Bayesian community, in order to find a learning algorithm based on the same principle which could produce (as in the case of Gaussian processes) a full predictive distribution as an output, rather than a single number. This has given rise to **relevance vector machines** (**RVM**). The general form of the prediction for a new input is Equation (7) as for SVMs: however, the method has certain advantages: (i) there is no regularisation parameter to be optimised; (ii) there is no requirement for k to be positive-definite; and most importantly, (iii) the kernel is usually far sparser than for SVMs (often using a number of relevant vectors which is an order of magnitude lower than the number of support vectors for a SVM).

2.9. Practical general insurance example: rating factor selection for reinsurance

Consider the problem of predicting the reinsurance premium charged for a given layer to different insurance companies for a specific line of business (motor, in this case) based on the profile of the insurer. This is an exercise in second-guessing how reinsurers rate excess-of-loss reinsurance for layers where loss experience is limited.

The factors that define the profile of the insurance company are chosen from a list including:

- percentage of private (v commercial) vehicles
- percentage of comprehensive (v TPFT) policies
- percentage of male drivers
- percentage of young drivers ($\leq$25 y.o.)
- percentage of mid-age (26-…) drivers
- percentage of drivers with $\geq$60% no-claims discount
- Average direct (original) premium
- Market share
- …

The input is in the form of a table such as that shown in Table 2. This table is based on real data but have been scrambled and scaled beyond recognition.

This is not a very sophisticated example and a multivariate Gaussian model of the linear variety will suffice: i.e., we model the reinsurance premium Y as a linear function of the rating factors: $Y = a_0 + a_1X_1 + \ldots + a_nX_n$

GLM approach. The standard approach to this problem with GLM is through forward or backward selection of the best model. Forward selection is a form of greedy approach and starts with the simplest model (in our case, $Y = a_0$, or $Y \sim 1$ in GLM shorthand), then adding at each step the variable which reduces the variance between the model and the empirical data by the largest amount, until the addition of a new variable does not add a statistically significant advantage. Backward selection (which will not be used here) works the other way round, starting from the complete model (that where all the variables are included) and removing one variable at a time.

Figure 4 shows the application of the forward selection model to our case, which leads to the selection of the model $Y \sim$ ADP + %Young. Table 3 provides the background numbers for this choice.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

Table 2. Reinsurance premium xs £5 m and possible rating factors for different insurance companies. Figures are based on real data but are scrambled and randomised beyond recognition.

Client	Private	Fleet	Male	Young	MidAge	Comp	NCB	AvgPremium	Mktshare	R/i premium
1	81%	0.0%	52%	3%	63%	0%	74%	255.2	8%	2.4
2	45%	45.6%	47%	1%	73%	80%	49%	720.9	1%	7.7
3	58%	6.2%	74%	3%	60%	58%	48%	318.9	12%	3.6
4	41%	11.8%	60%	2%	58%	83%	0%	412.6	6%	4.5
5	100%	0.0%	54%	4%	72%	98%	91%	325.2	6%	2.9
6	68%	6.9%	74%	2%	46%	79%	58%	287.5	2%	4.3
…	…	…	…	…	…	…	…	…	…	…
14	75%	0.0%	67%	37%	59%	55%	0%	446.2	1%	6.8
15	98%	0.0%	51%	20%	61%	90%	59%	430.3	6%	5.8

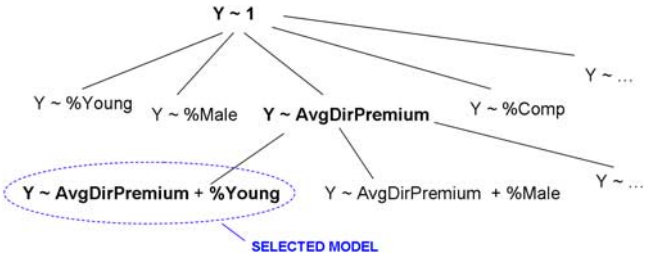

Figure 4. Illustration of the forward selection method.

Table 3. Increasingly complex models for the reinsurance premium, and their statistical significance. Every time the variable that explains most of the residual variance is added to the model, the variance is reduced. The column Pr(>|*t*|) gives the probability that the ratio between the coefficient and the standard error *of the latest variable added* is larger than |*t*| by chance although the true value of the coefficient is zero. A value above a significance value of 5–10% indicates that the coefficient is not significantly different from zero. More rigorously, the drop in variance obtained by the addition of a new variable is statistically significant only if the value of AIC (fourth column) decreases. Both these criteria indicate in this case that the best model is $Y \sim \text{ADP} + \%\text{Young}$.

Model	Variance	Pr(>|*t*|)	AIC
Y ~ 1	54.8	2.10^{-8}	69.1
Y ~ ADP	12.1	6.10^{-6}	46.9
Y ~ ADP + %Young	7.1	0.009	40.3
Y ~ ADP + %Young + MktShare	6.6	0.39	41.3

The main problem with this approach lies in the model selection process, which does not guarantee that it will not fall into local minima, and is relatively inefficient.

Regularisation approach. We now use a simple lasso regularisation approach to solve the same problem. We use an R package called "LARS", authored by Hastie & Efron (2012). This package performs an automatic selection process which is guaranteed to find the global minimum, and it does it within the time it takes to do a single least square regression.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

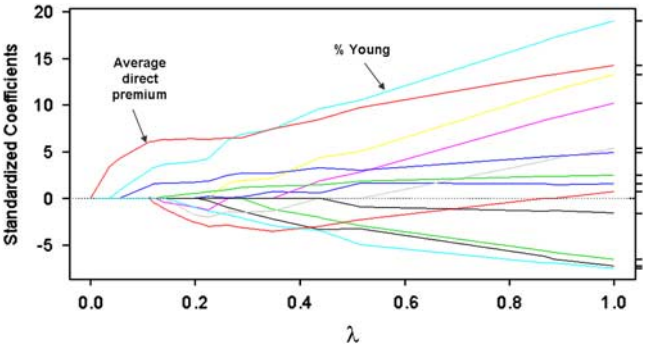

Figure 5. Illustration of the least-angle regression method (LARS) to generate all lasso solutions for different values of the regularisation coefficient.

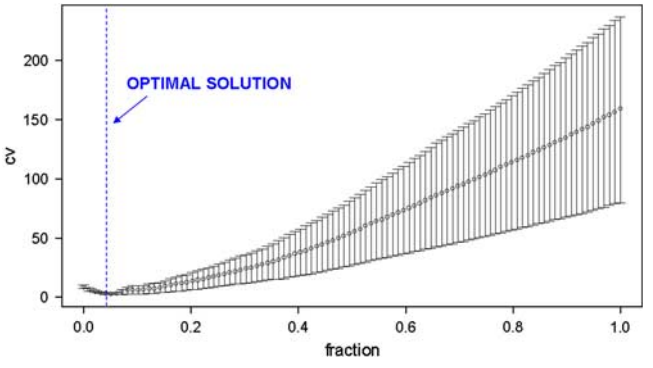

Figure 6. Expected prediction error for different values of the regularisation parameter, obtained through cross validation.

The coefficient values are plotted against the regularisation parameter λ, which is expressed as a fraction of $t^* = \sum_1^p \left|\hat{\beta}_j\right|$, where $\hat{\beta}$ is the least squares estimate of coefficient β_j. Note that when the regularisation parameter becomes larger than t^*, the lasso estimates coincide with those of least squares regression. Owing to the nature of the lasso, the coefficients are all zeros when $\lambda = 0$, then as we increase λ the coefficients become positive one by one, as in Figure 5.

As Figure 5 shows, the first coefficient to become non-zero is that for the average direct premium, and the second is that for the percentage of young drivers, then all the others, one by one.

The same package offers a method to decide which value of λ should be selected based on cross-validation, as shown in Figure 6, which gives the expected prediction error as a function of the value of λ, again expressed as a fraction to the largest coefficient.

The minimum value of the prediction error is obtained around 0.05. This corresponds to including the variables ADP and %Young, exactly as for the GLM model. The model has however been obtained effortlessly and is guaranteed to yield the global minimum.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

2.10. Comparison of different techniques for supervised learning

We now summarise our considerations on the different techniques for supervised learning that we have investigated so far, in order to give an at-a-glance comparison among these techniques.

Neural networks

- Neural networks are very flexible: they are not limited by linearity and they can express any function of the data.
- Neural networks do not require any model specification, and are therefore easy to implement.
- A key disadvantage of neural networks is that they provide "prediction without interpretation": the output of a risk costing exercise are the values assigned to the weights of the network, which have no intuitive meaning. This does not mean that they are any less rigorous than GLM or regularisation: their models can be validated exactly in the same way. However, they do not help us *understand* risk and this makes their acceptance by underwriters, management and regulators problematic.
- Neural networks are computationally demanding: training them takes a fair amount of time.

Generalised linear modelling

- The loss function can be chosen to reflect (to some degree) the underlying process noise.
- Generalised linear models are easy to interpret: the factors included in the model are those that have a significant impact on the risk, and the chosen functional dependency (e.g. exponential, polynomial) is usually simple.
- Generalised linear models are limited by linearity. However, in practice, a large dictionary of functions is possible and linearity in the coefficient will seldom be impairing.
- Model selection with forward or backward selection is computationally awkward and may get trapped in local minima.
- Standard methods for model validation (as those based on AIC) may overestimate the test error.

Regularised regression

- Feature selection is completely automated, and a global minimum is guaranteed to be reached. Model selection and validation can be performed very efficiently, avoiding the combinatorial explosion of best subset selection.
- The interpretation of regularisation results is easy as for GLM.
- Specific types of regularisation such as the elastic net can address the problems arising when there are highly correlated variables and when the number of predictors exceeds the number of available data points.
- The use of quadratic loss function for classical regularised regression is limited and causes degradation of performance with some types of noise. This has been demonstrated in Parodi (2009)with the use of artificial data for the case of Poisson noise: the expected prediction error increases as the Poisson rate of the underlying processes decreases (this makes sense as for large Poisson rates the distribution of the number of losses will be near-Gaussian).
- A different loss function can be used, but some of the theoretical results obtained for squared loss may be lost, and the solution may not be as simple (see Rosasco *et al*., 2004).

The comments above reflect the "textbook approach" of the relevant methodologies. It is fair to mention, however, that these methodologies come in many flavours and that hybrid approaches are possible and have actually been attempted. As an important example, GLM can be coupled with

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

regularisation: see for example Park & Hastie (2007), where l_1–regularization (lasso) was applied to a log P loss function. In the author's opinion this is one of the most promising avenues that actuaries should explore.

Also, cross-validation is used more in the context of neural networks and regularised regression than in the context of GLMs, but there is no reason why it should not be used for GLM as well. In a paper presented last year by a general insurance research organising (GIRO) working party (see Berry *et al*., 2009), the use of a hold-out sample was advocated as an important validation tool.

2.11. Dealing with sparse data

It is fair to mention at this point that most techniques discussed in this section work best when there is a large amount of data to play with. Indeed, techniques such as GLM are mostly used in personal lines insurance where the quantity of data is not an issue. However, statistical learning can in principle be applied as well where data are sparse, and some techniques have either been developed exactly to deal with the data sparsity issue or they help address data scarcity.

More specifically:

- some regularisation techniques such as the elastic net were devised to address the problem where the number of data points is much smaller than the number of explanatory factors (a clear case of data sparsity, and a counterintuitive one);
- in at least one context where data sparsity is almost physiological, i.e. the modelling of large losses, extreme value theory provides theoretical constraints on the type of models that can be fitted (e.g. a generalised Pareto distribution must be used to model the severity of losses above a large threshold). This has an interesting interpretation in terms of statistical learning, as one of the methods to avoid overfitting is to limit the number of models that one can choose from. In the case of EVT, the model selection phase can actually be skipped altogether and we can jump straight to model parameterisation and model validation;
- if we adopt a Bayesian approach to statistical learning, it is possible to incorporate the prior probability of models (see II. Dealing with uncertain knowledge, Section 2.4), and also to incorporate expert judgment. In this context, statistical learning provides a way of making up for the lack of data by tilting the balance away from the actual experience and towards prior or external knowledge.

The issue of the sparsity of data and how statistical learning can help with it was recently addressed in a talk at GIRO (see Chhabra & Parodi, 2010).

3. Exploratory Analysis

Section 2 (Learning from data) endeavoured to show that data-driven loss modelling can be described as problems of supervised learning. This section holds that another branch of machine learning, *unsupervised learning*, also has an important (albeit more tentative) role to play in insurance.

Informally (and loosely), unsupervised learning is finding patterns in data without a "teacher", i.e. someone who can confirm whether the patterns we have found correspond to something more meaningful than random fluctuations.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

To make a simple example, suppose you want to group people in Italy according to the dialect they speak, based on a number of unlabelled recordings of their speech that have been delivered to you by mail. You may select a number of features to analyse, perhaps properties of the frequency spectrum.

If you can hire an interpreter in the initial stage of your research, you will be able to classify a sample of the recordings (the training sample) by giving it the proper label: e.g. Piedmontese, Ligurian, Neapolitan, etc. When you receive a new recording, you will then be able to attempt classifying it as one of the previously labelled dialects based on the algorithm developed during the training stage. This is supervised learning.

In unsupervised learning, you may want to solve the same problem, but this time there is no one who can help you with labelling each speech sequence. However, you will still be able to perceive differences between the speeches – Calabrese will sound rather different from Venetian, even if you know nothing else – and your algorithm is likely to detect similarities between people using the same dialect. This is unsupervised learning. Actually there is an element of cheating here because the underlying categories (the dialects) are ultimately verifiable, whereas in most practical cases we are not that lucky.

Using the language of probability, unsupervised learning is characterised in Hastie *et al*. (2001 as follows:

Unsupervised learning. *Given a set of observations* $(X_1, \ldots X_n)$ *of a random p-vector X having a joint density Pr*(X), *infer the properties of Pr*(X) ***without the help of a teacher.***

(This definition is still quite loose, as it does not make it clear what these "properties" should be. Since unsupervised learning is in itself a very open-ended activity, it is difficult to produce a more formal definition. Ultimately, the activity of unsupervised learning is justified by the use we make of its outputs for further analysis.)

In unsupervised learning the target is unknown, and there are no dependent variables. This is to be compared to supervised learning, where one is concerned with determining $\Pr(Y|X)$ from a training sample $(x_1,y_1),\ldots(x_n,y_n)$, where Y is the dependent variable.

According to Hastie *et al*. (2001), the main problems of unsupervised learning are:

- *Clustering*. Find convex regions of the X− space containing modes of $\Pr(X)$ (that is, can $\Pr(X)$ be represented as a mixture of simpler densities?). More informally, this means segmenting data points into sets so that points in the same set are as similar as possible to each other and points in different sets are as dissimilar as possible.
- *Association rules*. Construct simple descriptions that describe regions of high density for high-dimensional (usually binary-valued) data (for example, market basket analysis)

Applications of unsupervised learning to general insurance problems are for example:

- Exploratory analysis, to check if there are patterns in data that may lead to further investigation.
- Descriptive data mining, which is actually a more organised version of exploratory analysis which is usually applied to large bases of data that would be otherwise difficult to make sense of.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

A specific example from the insurance literature is that of territories clustering as performed, for example, by Yao (2008). The input was the output of a GLM exercise, where a number of features – excluding location – was used to produce a predictive model of claims experience for various categories of drivers. By performing clustering, the author creates an extra variable describing the residual effect of location, and this variable can then be used as a rating factor.

The most important example of the use of clustering and association rules is arguably in data mining, which is the efficient discovery of previously unknown patterns in large databases. Data mining is used by many large companies (not necessarily, and not especially, in the financial sector) to make sense of their large collection of data and spot business opportunities. It is not a single technique – more an ongoing activity based on a collection of techniques which are embedded in an ad-hoc software system. For this reason, we will focus here on the most important of these techniques – clustering – and refer the interested reader to the paper by Guo (2003) for an example of actuarial application.

Here we just want to mention a couple of examples of data mining applications to general insurance:

- descriptive data mining has allowed the discovery of surprising relationships, e.g. the fact that the credit rating of an individual is strongly linked to the individual's propensity to claim;
- for one insurer it has been found out by this method that a subset of a very risky category (males less than 21 y.o.) in its portfolio, those that have vintage cars, were far less risky than average, and could therefore be targeted with special prices.

As important as data mining is, it is important to notice that data mining lends itself by construction to the problem of data snooping: having a large data set at our disposal with the stated goal of finding regularities, we are bound to find some regularities simply by chance. It is therefore important to apply model validation techniques to all findings by data mining techniques.

The issue of data snooping in data mining is especially insidious in the analysis of financial time series, as – since there is only one series to analysis – the scope for validating the results against an independent data set is very limited, and the only option is really to hold out a sample of the more recent data points. As a consequence, some degree of data snooping is inevitable in this case.

In the rest of the section we will focus on clustering. We will first look at a number of different clustering techniques (Section 3.1). We will then illustrate in some detail a practical general insurance application (Section 3.2).

3.1. Clustering techniques

One possible categorisation of clustering techniques is this:

- Partitioning techniques ($K-$means, K-medoids, EM), which partition data based on a given dissimilarity measure.
- Hierarchical methods, which subdivide data by a successive number of splits.
- Spectral clustering, which works by graph partitioning techniques after a transformation into a suitable space.
- Dimensionality reduction techniques (principal component analysis, self-organising maps (SOMs), generative topographic mapping, principal component analysis, etc), which are based on

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

the observation that often the data points lie in (possibly non-linear) low-dimensional manifolds embedded in a data space with many more dimensions.

- Other: density-based methods, grid methods, kernel clustering. These will not be dealt with here.

We are going to look at each of them in turn.

Partitioning techniques. Partitioning techniques subdivide data based on a given (dis)similarity measure. The most popular algorithm for clustering is perhaps *K*-means, as it is simple and intuitive.

The dissimilarity measure for *K*-means is typically (but not necessarily) the *squared* Euclidean distance $d(x_i, x_j) = \|x_i - x_j\|^2$ (notice that x_i and x_j are in general vectors in $\mathbb{R}^p$).

The aim is to minimise the within-cluster point scatter:

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(j)=k} d(x_i, x_j) =$$
$$= \frac{1}{2}\sum_{k=1}^{K} N_k \sum_{C(j)=k} d(x_j, m_k)$$

The method works iteratively by

1. Choosing the number K of clusters
2. Choosing randomly (or cleverly) K data point as the initial cluster means m_k
3. Assigning each object x to the closest cluster mean
4. Updating cluster means by minimising W(C) with respect to the m_k's
5. Iterating steps 3 and 4 until assignment does not change

This method's advantages are that it is easy to understand and apply. Also, it always converges quickly. However, it has several disadvantages:

- the number of clusters must be decided in advance. Different values of K have therefore to be tried to check which value gives the "best" results;
- in case the Euclidean distance is used, the method is sensitive to noise, as the sum of the squared distance between points may be overly influenced by a few data points with larger distance;
- the algorithm may get stuck in a suboptimal local minimum. As regards to this, the choice of the initial clusters is crucial;
- clusters lie in disjoint convex sets of the underlying space. This may be a disadvantage when the actual shape of the "real"' clusters is unusual, as in Figure 7;
- there are issues when the density of different clusters is not uniform.

Examples of other partitioning methods (besides *K*-means) are *K*-medoids and expectation maximisation. *K*-medoids is very similar to *K*-means, but rather than choosing freely among points in the underlying space to be the cluster means, it always chooses one of the actual points in the dataset. Expectation maximisation is a ubiquitous technique in statistics, that can be applied to a number of contexts, including clustering. See for example Hastie *et al*. (2001) for an explanation.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

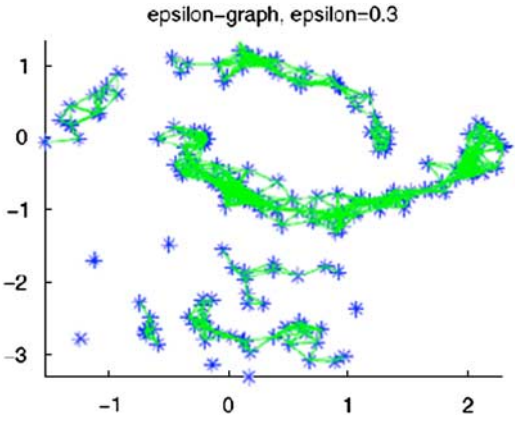

Figure 7. The clusters in this figure are not convex. This will cause problems with *k*-means. The clusters here are obtained using spectral clustering (explained later on in this section).

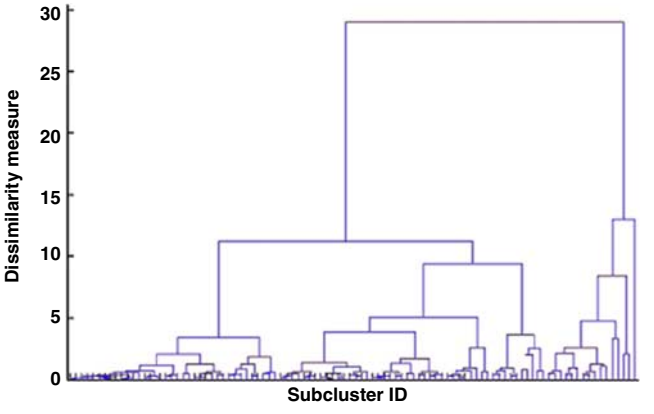

Figure 8. An example of dendrogram [modified from Yao paper, Yao, 2008]. Notice how the number of clusters depends on the dissimilarity measure at which two dendrograms are split.

Hierarchical methods. Hierarchical methods use cluster-to-cluster similarity in order to decide whether to merge/split clusters. They create a hierarchical decomposition forming a dendrogram (a tree that splits recursively). The idea behind a dendrogram is that depending on the dissimilarity threshold at which you decide that two clusters should be merged (y–axis), the number of clusters (i.e., the number of vertical lines crossed by a line with constant y varies. An example of a dendrogram is shown in Figure 8.

Hierarchical methods have some similarity with rule-induction systems, which segment data according to their attributes, e.g. based on a number of "If-Then" conditions applied to each attribute in turn (see, e.g., Liang, 1992).

One advantage of hierarchical methods is that – unlike *K*-means – they do not commit to a number of clusters K in advance, giving them more flexibility. Also, they are easy to understand

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

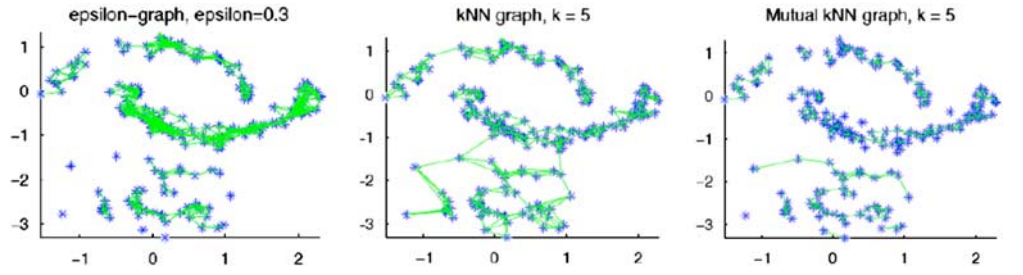

Figure 9. Different types of similarity graph: ε-neighbourhood graph (left), k-NN graph (centre), mutual k-NN graph (right). The difference between k-NN and mutual k-NN is that in the latter we include an edge between vertices v_i and v_j only if the vertices are both among each other's k nearest neighbours. Notice the non-convex shape of the clusters and the fact that the clusters have different densities. The figure is borrowed from von Luxburg (2007).

(the dendrogram provides a highly intuitive description of the product of hierarchical clustering at different levels of dissimilarity) and apply.

One disadvantage is that the dendrogram does not provide any clear decision criteria – judgment must be used to select the threshold of the dissimilarity measure which will then produce a specific clustering.

Spectral clustering. Spectral clustering is becoming one of the most popular techniques in clustering, owing to its flexibility, its efficiency and its theoretical elegance. Only a cursory overview will be given here, but the interested reader will find an entertaining reference in the tutorial by von Luxburg (2007).

Spectral clustering has its roots in graph theory, and its central idea is to solve the problem of clustering not in the original space (as K-means does) but on a transformed space embedded in a graph, the so-called similarity graph. The similarity graph is a graph whose vertices are the data points and two data points are connected by an edge if they are similar, as in Figure 9. The similarity function is somehow related to the "distance" between the points in a suitable space. For example, in a Euclidean space $\mathbb{R}^k$ the similarity function might be defined as $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/2\sigma^2)$, where $\|x_i - x_j\|$ is the Euclidean distance in $\mathbb{R}^k$. Finding the clusters in the data set is then equivalent to finding partitions (groups of vertices) in this graph which are weakly connected to each other but whose vertices are very similar inside.

The essential aspect of spectral clustering is how you build the similarity graph, so it is perhaps worth spending a few words on this. Three types of similarity graphs are commonly used:

- the **ε-neighbourhood graph**, in which every pair of vertices for which the similarity function is larger than ε is connected (see Figure 9(left)). This graph is usually unweighted;
- the **k-nearest neighbour graph**, in which the k nearest neighbours of a given vertex are connected to that vertex (see Figure 9(centre and right). This type of similarity graph is useful especially when we have clusters at different scales - that is, one cluster may be denser, the other more rarified, as in Figure 7;
- the **fully connected graph**, in which *all* vertices are connected. This graph will obviously be weighted and the weighting should decay exponentially, thus effectively defining a neighbourhood radius.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

One version of the spectral clustering algorithm ("unnormalised spectral clustering") works as follows:

1. Build the similarity graph $G = (V, E)$ where $V = \{v_1, \ldots v_n\}$ are the vertices (the data points), E is a set of edges, i.e. a subset of $V \times V$ (pairs of vertices). The graph is assumed to be undirected. If the graph is weighted, that means that every edge (v_i, v_j) will have an associated weight $w_{i,j}$. The matrix $W = \{w_{i,j}\}_{i,j=1,\ldots n}$ is called the weighted adjacency matrix.
2. Calculate the graph Laplacian $\mathcal{L} = DW$, where W is the adjacency matrix defined above and $D = \mathrm{diag}(d_1, \ldots d_n)$, where $d_i = \sum_j w_{i,j}$. The laplacian $\mathcal{L}$ has many interesting properties: for example, it is symmetric and positive semi-definite, and has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. The multiplicity k of the eigenvalue 0 of $\mathcal{L}$ equals the number of connected components $A_1, \ldots A_k$ in the graph.
3. Calculate the first k eigenvectors $u_1, \ldots u_k$ of $\mathcal{L}$ and build the matrix $U = (u_1, \ldots u_k)$ (columns) $= (y_1, \ldots y_n)^T$ (rows).
4. Group points $y_1, \ldots y_n$ into k clusters $C_1, \ldots C_k$ using, e.g., K-means

This algorithm produces sensible clusters, but it is not obvious why! Several arguments have been put forward to explain why spectral clustering works: see for example von Luxburg (2007). The perturbation theory argument is probably the most intuitive, and can be outlined as follows.

Perturbation theory studies how the behaviour of eigenvalues and eigenvectors changes if one applies a small perturbating matrix H to the original matrix A. This may help us understand the workings of spectral clustering by considering the ideal case – that is, the case where the clusters are already perfectly separated at the stage when one builds the similarity graph. In this case, the points $y_1, \ldots y_n$ built by the algorithm above will have the simple form $(0, \ldots 0,1,0, \ldots 0)$, and the position of the 1 indicates the connected component this point belongs to. All the vectors belonging to the same connected component are coincidental, and identifying the clusters is trivial.

If we now perturb the original matrix by a small amount, the points y_i will not be exactly $(0, \ldots 0, 1, 0, \ldots 0)$ as above, but a K-means algorithm will still find the same clusters.

The **pros and cons of spectral clustering** as listed below are mostly borrowed from von Luxburg (2007).

- No strong assumptions on the shape of the clusters are needed. In particular, it is not necessary for clusters to be convex in the original space (see for example the moon-slice-shaped clusters in Figure 7).
- Spectral clustering is efficient on sparse graphs – of course, it is up to us to ensure that we choose the similarity graph so that it *is* sparse!
- Finding the solution given the similarity graph is a linear problem and therefore one cannot get stuck in local minima. Also, it is not necessary to restart the algorithm several times with different initial values, as for many other clustering methods.
- On the downside, the choice of a good similarity graph is crucial and not always easy. It is unstable under different choices of the parameters for the neighbourhood graph. All this amounts to saying that spectral clustering cannot be used as a black box!
- Spectral clustering may be difficult to understand/communicate: the very reason why it works is not obvious. However, its obscurity is not of the same type as that of neural networks, because here it is clear what the algorithm is trying to achieve – finding clusters on the similarity graph rather than on the original space.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

Dimensionality reduction techniques. These techniques are based on the observation that data points, although they may live in a data space with many dimensions, in practice may all lie close to a low-dimensional manifold[7] embedded in the data space.

As an example, buyers of house insurance may be described as points in a space with as many dimensions as there are rating factors that have been collected – say 20 or more of them: however, in practice you may find that certain rating factors which you have zealously collected (e.g., the current colour of the house) end up being not material, or that only certain combinations of the values of the factors occur in practice.

Sometimes the underlying manifold is a linear manifold, i.e. it is a linear subspace of the data space. In this case linear modelling techniques suffice to discover this subspace. An example of these techniques is given by *principal component analysis* (PCA), which projects all the data points onto a principal subspace with some optimisation criterion, e.g. the subspace that captures most of the variance. PCA comes in many fashions: straight PCA, probabilistic PCA, EM PCA, Bayesian PCA, factor analysis, kernel PCA (see Bishop (2007) for a short review).

In general, the manifold around which the data lie will not be linear, and nonlinear techniques are therefore needed. This can be dealt with by, e.g.,

- approximating the non-linear manifold with a piecewise linear manifold (e.g. approximating a hypersurface locally with hyperplanes);
- using self-organising maps (aka Kohonen maps). A self-organising map represents a 2D nonlinear manifold as an array of discrete points, called prototype vectors. Initially, these prototype vectors are distributed at random, but during training they "self organise" (hence the name of this technique) into a discrete rendition of a low-dimensional smooth manifold. One disadvantage of this technique is that there is not a clearly defined cost function that can be minimised and according to which convergence can be assessed. The next technique, generative topographical mapping, is a variant of SOMs that addresses this problem;
- using generative topographical mapping (GTM). This is a probabilistic version of the self-organising maps mentioned above, and produces a probability density in the data space made up of a mixture of Gaussian distributions with the same variance and with means lying on a smooth two-dimensional manifold. The "optimal" mixture corresponds to the maximum of a log-likelihood function.

The reader who is interested in more detail on these techniques is referred to Bishop (2007) and to the literature cited therein.

3.2. Practical general insurance example: Clustering for IBNER data

A possible application of unsupervised learning is to IBNER analysis. The underlying problem is that of predicting the ultimate value of a claim (or a probability distribution) given its previous history of paid and incurred amounts. An example is shown in Figure 10.

The numerical example illustrated here was analysed with the help of Nicola Rebagliati, from the Department of Computer Science of the University of Genoa.

[7] A manifold is an m-dimensional geometric structure embedded in an n-dimensional space with n>m. Simple examples of manifolds are lines or surfaces in three-dimensional spaces, and lines in two-dimensional spaces. Examples of *linear* manifolds are planes and *straight* lines in three-dimensional spaces.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

	2000	2001	2002	2003	2004	2005	2006	2007	2008
Paid	19,792	363,306	487,648	1,735,328	1,922,504	1,922,504	1,922,504	1,922,504	1,922,504
O/S	967,500	877,200	753,360	147,060	0	0	0	0	0
Incurred	987,292	1,240,506	1,241,008	1,882,388	1,922,504	1,922,504	1,922,504	1,922,504	1,922,504
O/S ratio	98.0%	70.7%	60.7%	7.8%	0.0%	0.0%	0.0%	0.0%	0.0%
IBNER factor		1.256	1.000	1.517	1.021	1.000	1.000	1.000	1.000

Figure 10. The history of a claim occurred in 2000 and reported in the same year. For each year of development, the paid, outstanding and incurred (estimated) amounts are given. The outstanding ratio (outstanding amount divided by incurred amount) is also given for each year. The last row shows the IBNER factors for this claim, calculated as the ratio between the incurred amounts of two successive years. The claim shown here is a real claim after scaling by a random amount and the removal of any identifiable information.

For pricing or reserving purposes, especially for large losses, one often looks at the distribution of past IBNER factors to predict the ultimate value of a particular claim or to put aside some IBNER reserves. The idea is that by spotting a systematic bias (on average) on the claims estimates, the estimates can be put right (again, on average!) by correcting all claims with the same characteristics.

The calculation of IBNER factors is usually performed by chain ladder calculations (or something similar) and often a simple average over all open claims (settled claims will of course have no IBNER, unless they are re-opened) is all that is used, especially if the number of individual claims histories is not large: the output of such analysis would then be an IBNER factor which depends only on the development year.

However, it is obvious that the IBNER factor will depend in general on many factors, such as:

- development year (the younger the claim, the more scope there is for it to deviate from the incurred estimate), as already mentioned;
- size of claim (larger claims may have more uncertainty about them);
- outstanding ratio (the larger the outstanding amount, the more conjectural the estimate of the incurred value will be);
- type of claim (for example, bodily injury claims will be more difficult than property claims to estimate reliably as the victim's health may worsen unexpectedly);
- reserving style (claims managers may be more or less effective at reserving, and some may introduce a systematic upward/downward bias);
- …

As in all situations in which one needs to understand how different factors impact a given variable, one can use the data set to perform supervised learning as in Section 2 ("Learning from data"). However, not all the variables in the list above lend themselves to be used easily as input variables: specifically, the "reserving style"' above is quite difficult to use as the claims history is a vector of, say, 15 years of development.

One possible solution is to collect all individual claims development histories and group them into clusters of similar behaviour. Once clustering is performed, the cluster ID can be used as one of the rating factors in a regression exercise with GLM or regularisation.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

1	2	3	4	5
0.332	0.332	0.332	0.442	1.050
0.679	0.679	0.942	1.056	1.000
0.785	0.819	0.967	0.979	0.999
0.546	0.546	0.819	0.956	1.025
0.509	0.509	0.764	1.019	1.019
1.044	1.044	0.992	0.992	1.000
0.611	0.614	0.718	0.780	0.994
2.313	3.700	4.010	1.850	1.017
1.347	1.797	1.018	1.000	1.000
0.996	1.000	1.000	1.000	1.000
0.535	0.936	1.000	1.000	1.000
1.167	1.009	1.000	1.000	1.000
0.326	0.432	0.534	0.722	0.997
0.333	0.449	0.557	0.779	1.001
0.813	0.813	1.216	1.013	1.000
0.551	0.551	1.033	1.033	1.000
0.569	0.988	0.987	1.013	1.005
1.008	1.008	1.008	1.008	1.008

Figure 11. Examples of loss progressions. The most recent estimate of the claim (possibly beyond the 5-year horizon) is normalised to 1.

		Cluster ID									
		1	2	3	4	5	6	7	8	9	10
Development year	1	1.589	2.996	1.010	1.677	4.745	0.914	1.334	0.510	2.294	3.222
	2	1.545	1.867	1.311	1.852	4.773	0.973	1.073	0.626	2.545	3.453
	3	1.161	1.071	1.271	1.793	3.266	1.006	1.000	0.778	2.150	3.522
	4	1.080	1.078	1.108	1.313	1.251	1.010	0.999	0.914	1.472	1.147
	5	1.001	1.005	1.004	1.000	1.005	1.000	1.000	1.004	0.998	1.003

Figure 12. Cluster centres for the clusters calculated with K-means, $k = 10$.

One complication of this exercise is that different claims histories will have different lengths (some may be 2 years long, some 15) and therefore one must put a little care into how one defines a similarity measure. Since here we are only interested in giving an illustration of various applications of clustering, we will limit ourselves to extract sections of claims histories of exactly 5 years (the history of the claim may go on after 5 years, but we look at a section of the data spanning 5 years only).

The data used are large motor losses from several UK insurers. The losses have been normalised so that the latest known incurred value (not necessarily that at the fifth year) is set to 1 – which makes them unrecognisable and removes the dependency on loss size.

Figure 11 shows a few examples of the claims histories we are trying to cluster, after normalisation.

The clusters were obtained with a simple K-means algorithm. A value of $K = 10$ was chosen, which gives good results in terms of separation of clusters. The cluster means are shown in Figure 12.

Notice that these clusters can be interpreted as follows:

- cluster 6 is the most typical behaviour and occurs when the estimate of the incurred amount is quite stable after an initial 8% increase. Cluster 7 is also relatively stable but it starts with a 30% over-reserving;

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

	Cluster ID									
	1	2	3	4	5	6	7	8	9	10
A	0.00%	0.00%	0.00%	0.00%	0.00%	**100.00%**	0.00%	**0.00%**	0.00%	0.00%
B	0.00%	0.00%	0.00%	0.00%	0.00%	**66.67%**	0.00%	**33.33%**	0.00%	0.00%
C	0.00%	0.00%	0.00%	0.00%	0.00%	**44.44%**	0.00%	**55.56%**	0.00%	0.00%
D	0.00%	0.00%	25.00%	0.00%	0.00%	**50.00%**	0.00%	**25.00%**	0.00%	0.00%
E	0.00%	7.14%	0.00%	0.00%	0.00%	**42.86%**	35.71%	**7.14%**	7.14%	0.00%
F	10.23%	4.55%	5.68%	3.41%	1.14%	**48.86%**	4.55%	**20.45%**	1.14%	0.00%
G	0.00%	0.00%	0.00%	0.00%	0.00%	**100.00%**	0.00%	**0.00%**	0.00%	0.00%
H	4.26%	2.13%	10.64%	4.26%	2.13%	**42.55%**	4.26%	**27.66%**	2.13%	0.00%
I	50.00%	0.00%	0.00%	0.00%	0.00%	**0.00%**	0.00%	**50.00%**	0.00%	0.00%
J	2.94%	0.00%	8.82%	8.82%	0.00%	**35.29%**	2.94%	**38.24%**	2.94%	0.00%
K	0.00%	5.56%	11.11%	0.00%	0.00%	**22.22%**	27.78%	**16.67%**	11.11%	5.56%
L	8.33%	1.39%	15.28%	1.39%	0.00%	**38.89%**	9.72%	**20.83%**	1.39%	2.78%
...	...	...	...	...	...	...	...	...	...	...

Figure 13. The percentage of each type of cluster for a selection of UK companies.

- cluster 8 shows a typical pattern of under-reserving;
- clusters 1, 2, 4, 5, 9, 10 are typical patterns of over-reserving;
- cluster 3 shows claims for which the incurred amount has returned to the initial value after a period of over-reserving.

It is interesting to compare these clusters with a piece of information that we have not used in the clustering process – the name of the company that reserved those claims. The results of this comparison are shown in Figure 13. It is clear that certain companies (such as Company J) have a higher-than-average tendency to under-reserve.

It is interesting to compare these results with those one obtains with spectral clustering. To use spectral clustering, we first have to choose a similarity graph. One choice is the ε –neighbourhood graph with similarity function $s(x_i, x_j) = \exp(-||x_i - x_j||^2/2\sigma^2)$, where $||x_i - x_j||$ is the distance in $\mathbb{R}^5$ between our vectors representing the development of claims. However, this is going to give results very similar to the K-means algorithm tested above.

We have therefore used the k-nearest-neighbour similarity graph.The choice of k is of course crucial. Unfortunately, there are few theoretical results guiding us in the choice of k. A few heuristics are often recommended, though. One is that k be chosen so that the resulting similarity graph is connected, or that the number of connected components is less than the number of expected clusters – otherwise there is a danger that the algorithm will simply return the connected components as clusters. For very large graphs, a rule of thumb is to look at values of k in the region of $\log n$(n being the number of data points). In our case, this does not give sensible results.

Another option is to look at the so-called Laplacian spectrum, which gives (for a given number of k of the kNN graph) the value of all the n eigenvalues (some of them possibly repeated if their eigenspace has dimension >1) in decreasing order. In the ideal situation we would have only K eigenvalues[8] that are near zero and can therefore be considered—from a perturbation theory point of view – as "noisy" versions of the first eigenvalue $\lambda_1 = 0$ of multiplicity K. Then there should be, always in the ideal case, a gap between the K-th and the $K+1$th eigenvalue,related to the gap between λ_1 and λ_2.

[8] K indicates here the number of clusters, k is the number of nearest neighbours in the kNN graph.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

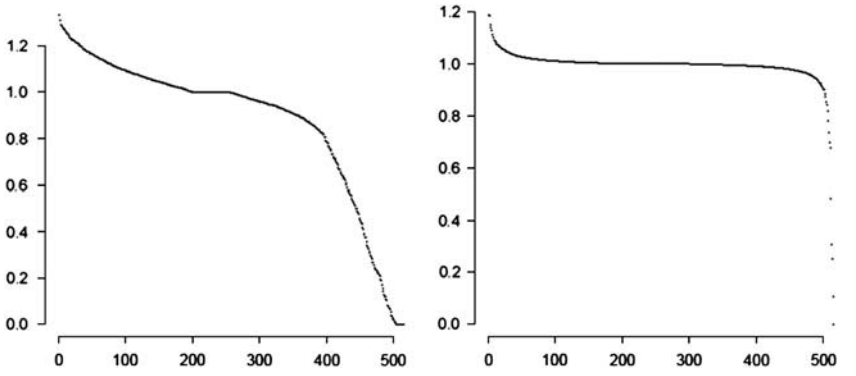

Figure 14. *Left*. Laplacian spectrum for the k-nearest-neighbour similarity graph, $k = 20$. No structure is apparent in the graph and there is a large number of eigenvalues near zero, without a clear difference between the near-zero behaviour and the rest. It is therefore not possible to identify a small number of eigenvalues that are near-zero. *Right*. Laplacian spectrum for the k-nearest-neighbour similarity graph, $k = 300$. The structure is better and there are only a few eigenvalues below 0.5 (five of them). There is then a significant gap between the fifth and the sixth eigenvalue. This structure therefore suggests the existence of five clusters (but anything between 3 and 5 would be acceptable).

		Cluster ID				
		1	2	3	4	5
Development year	1	2.335	3.643	1.427	0.948	0.492
	2	1.978	4.025	1.478	0.992	0.607
	3	1.711	3.022	1.246	1.020	0.755
	4	1.309	1.157	1.089	1.016	0.909
	5	1.001	1.003	1.001	1.001	1.004

Figure 15. Cluster centres for the clusters calculated with spectral clustering. A k -nearest-neighbour similarity graph has been used, with $k = 300$.

After trying out several values of k, we have found that we need to go up to about $k = 300$ to have the appropriate structure of the graph, and the Laplacian spectrum indicates that a reasonable choice for the number of clusters is 5. See Figure 14 for examples of Laplacian spectrums.

Notice that the selected number ($k = 300$) is quite an extreme choice for the number of nearest neighbours in the similarity graph. The underlying reason might just be that there are no natural clusters in this IBNER problem, because the patterns in IBNER development are not sharply defined. However, the existence of a few (3-5) patterns is indeed confirmed by a comparison of the results of spectral clustering and K-means, and by looking at which clusters are more common for each company.

These results are shown in Figure 15, which shows the centres of the five clusters found with spectral clustering, and in Figure 16, which shows the percentage of each cluster for each company. There is indeed a striking similarity with the results obtained with K-means: in both cases, we have an important cluster collecting all the claim developments that are stable (cluster 6 for K-means and cluster 5 for spectral clustering). Also, in both cases we have a single cluster which attracts all claims

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

Company	Cluster ID 1	2	3	4	5
A	0.00%	0.00%	0.00%	100.00%	0.00%
B	0.00%	0.00%	0.00%	66.67%	33.33%
C	0.00%	0.00%	0.00%	44.44%	55.56%
D	0.00%	0.00%	25.00%	50.00%	25.00%
E	14.29%	0.00%	14.29%	64.29%	7.14%
F	9.09%	1.14%	14.77%	59.09%	15.91%
G	0.00%	0.00%	0.00%	100.00%	0.00%
H	6.38%	4.26%	10.64%	57.45%	21.28%
I	0.00%	0.00%	50.00%	0.00%	50.00%
J	8.82%	0.00%	8.82%	47.06%	35.29%
K	16.67%	5.56%	5.56%	61.11%	11.11%
L	2.78%	4.17%	16.67%	58.33%	18.06%
...	...	...	...	...	...

Figure 16. A comparison of a few companies (after scrambling, and selection of the first half of them, to make them fully unrecognisable).

that are under-reserved and sometimes keep growing year-on-year. The other clusters are in both cases mostly examples of over-reserving and could easily be further aggregated. The main difference is the presence in *K*-means of a cluster (cluster 3) that starts from 1, goes up to 1.3 and goes back to 1 again – however, this is likely to be an artifact, and spectral clustering is probably right in ignoring it.

Overall, there seems to be some improvement with respect to *K*-means. However, it should be mentioned that in this example the clusters are probably already convex in the original space, and clusters are not sharply defined because claims developments vary over a continuous range.

Although in this case the difference between *K*-means and spectral clustering is not striking, spectral clustering is a far more flexible method (which incorporates *K*-means anyway in the standard implementation of the method) and despite the fact that it takes a while to explain, it is quite simple to implement, it is very efficient and has some important properties (local minima, convexity in the appropriate space). It is therefore something that actuaries should consider alongside *K*-means when doing clustering, especially when the structure of their data is complex, before concluding that clustering does not help them.

4. Conclusions

The conclusion of the first part of this investigation is that statistical learning provides a natural framework for data-driven loss modelling. More specifically:

- The familiar problems of pricing, reserving and capital modelling based on data and soft knowledge are essentially examples of *supervised machine learning*. Common techniques for dealing with this problem include regularised regression, GLM, GAM, neural networks, projection pursuit.
- Exploratory analysis – in itself a rather loose and unstructured activity – can benefit from unsupervised learning techniques such as clustering and association rules. This is crucial especially for personal lines insurance, where data mining offers the possibility of finding hidden patterns in large data sets. Also, clustering can be used to identify a small number of values for risk factors that are multi-dimensional in nature.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

4.1. Practical findings

Although the main question addressed by this paper was epistemological, "What is the appropriate framework to describe and understand risk?"', this paper contains a number of practical results:

- A comparison of different statistical learning techniques was carried out. The techniques looked at were GLM (the current industry standard), neural networks and regularisation. Neural networks are of limited use for general insurance applications as they provide prediction without interpretation: in other terms, they calculate risk without enhancing our understanding of it. Regularised regression provides a valid alternative to GLM, and sparsity-based techniques such as the lasso and the elastic net are especially interesting as they provide an efficient way of selecting the relevant variables. Elastic net should also be considered for use with situations in which the number of observations is limited, even far smaller than the number of parameters. The main drawback of regularised regression is that usually this uses a squared loss function, which leads to a deterioration of results in the case, for example, of Poisson noise. GLM uses a log P loss function which is more general. A promising approach is that of using regularised GLM, which has a general loss function (log P) and still benefits from the application of sparsity-based techniques.
- Clustering techniques (also from machine learning) are useful for a number of tasks ranging from exploratory analysis to data mining. When addressing these problems, the actuarial community should be aware not only of the basic clustering techniques such as K-means but also of more modern approaches such as spectral clustering, which provide far more flexibility.
- Machine learning also offers a number of different methods for model selection and diagnostics for model validation: splitting the observations into three sets (training sample, selection sample, test sample), K-fold cross-validation, AIC, BIC, MDL… Some of these diagnostics (hold-out sample, AIC) are familiar to the actuarial community, especially in personal lines, but we would benefit from enlarging our toolkit, especially with the systematic use of K-fold cross-validation, which provides a clever way of overcoming the problem of scarce data.

4.2. Limitations and future research

This review of computational intelligence techniques is by necessity limited, and the examples from general insurance are only an illustration and there is plenty of scope of more in-depth research. Specifically, the comparison between GLM and regularised regression should be made far more systematic and extended beyond the simple case of the lasso, through a battery of controlled experiments under different experimental conditions, e.g. also when the number of rating factors exceeds the number of data points. Regularised GLM should also be tested to see how it performs with respect to regularised (squared loss) regression and traditional GLM.

A more systematic comparison of different clustering techniques on IBNER data should also be attempted. The example analysed here only involved five years of development for data, but the most general case should be analysed, possibly on a larger set of data. It should also be determined whether this helps improve the results of predictive modelling for IBNER purposes.

Acknowledgements

I would like to thank the Profession for giving me the opportunity to pursue this piece of research. In particular I would like to thank Julian Lowe, my internal supervisor, for his time and guidance and especially for the kind encouragement throughout the project. I would also like to thank Mark Flower

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press

for his painstaking revision of an earlier draft. I am indebted to Alessandro Verri for allowing me to spend several weeks at the University of Genoa, and for many mind-opening conversations. My thanks extend to his team, for talking me through their projects and for their precious feedback on this work. For this first part of the paper, I am especially thankful to Nicola Rebagliati, who ran the experiments on IBNER clustering and turned me into an instant fan of spectral clustering; to Annalisa Barla, for the time she spent helping me with the regularisation software, and to Curzio Basso for organising my talks at the University of Genoa. I would also like to thank these people for very helpful discussion: Jill Boulton, James Coyle, Ernesto De Vito, Phil Ellis, Clinton Freeman, Chris Gingell, Di Kuang, Viviana Mascardi, Eamonn McMurrough, Stephen Roberts, Andy Smyth, Jen Tan Ning, Chris Russell, Claire Wilkinson and Ji Yao.

References

Berry, J., Hemming, G., Matov, G. & Morris, O. (2009). Report of the model validation and monitoring in personal lines pricing working party. *GIRO Proceedings*. Available at http://www.actuaries.org.uk/research-and-resources/documents/report-model-validation-and-monitoring-personal-lines-pricing-worki

Bishop, C.M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*Springer.

Chhabra, A. & Parodi, P. (2010). Dealing with sparse data: Practical Challenges and techniques. *GIRO Proceedings*. Available at http://www.actuaries.org.uk/research-and-resources/documents/b04-dealing-sparse-data

Cummins, J.D. & Derrig, R.A. (1997). Fuzzy financial pricing of property-liability insurance. *North American Actuarial Journal*, **1**(4), 21–44.

De Mol, C., De Vito, E. & Rosasco, L. (2009). Elastic-net regularization in learning theory. *Journal of Complexity*, **25**(2), 201–230.

Dugas, C., Bengio, Y., Chapados, N., Vincent, P., Denoncourt, G. & Fournier, C. (2003). *Statistical learning algorithms applied to automobile insurance ratemaking*. 2003 CAS Winter Forum, Casualty Actuarial Society.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, **32**(2), 407–499.

Grunwald, P.D., Myung, J. & Pitt, M.A. (2005)? 2009 in text on page 8. *Advances in minimum description length*. MIT Press.

Guo, L. (2003). Applying data mining techniques in property/casualty insurance. *CAS Forum*.

Hastie, T. & Efron, B. (2012). *lars: Least Angle Regression, Lass and Forward Stagewise*. R package (http://cran.r-project.org/web/packages/lars/)

Hastie, T., Tibshirani, R. & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

Hertz, J., Krogh, A. & Palmer, R.G. (1991). *Introduction to the theory of neural computation*. Addison-Wesley Pub. Co., Redwood City, Calif.

Ioannidis, J.P.A. (2005). Why Most Published Research Findings Are False. PLoS Medicine (San Francisco: Public Library of Science) 2 (8).

Jacobsson, H. (2006). *Rule Extraction from Recurrent Neural Networks*. PhD dissertation. Department of Computer Science, University of Sheffield. Available at http://www.dcs.shef.ac.uk/intranet/research/phdtheses/Jacobsson2006.pdf

Lemaire, J. (1990). Fuzzy insurance. *Astin Bulletin*, **20**(1), 33–55.

Liang, T-P. (1992). A composite approach to inducing knowledge for expert systems design. *Management Science*, **38**, No. 1, 1992.

Lowe, J. & Pryor, L. (1996). Neural networks v. GLMs in pricing general insurance. *GIRO Proceedings*. Available at http://www.actuaries.org.uk/research-and-resources/documents/neural-networks-vs-glms-pricing-general-insurance

MacKay, D.J.C. (2003). *Information Theory, Inference, and Learning Algorithms (4th Printing)*. Cambridge University Press.

McCulloch, W.S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133.

Minsky, M.L. & Papert, S.A. (1969). *Perceptrons*. Cambridge: MIT Press.

Modlin, C., Feldblum, S., Schirmacher, D., Schirmacher, E., Thandi, N. & Anderson, D. (2004). A practitioner's guide to generalized linear models. *CAS Discussion Paper.*

Papadimitriou, C.M. (1994). *Computational complexity*. Addison-Wesley, Reading, Massachusetts.

Parodi, P. (2009). Computational intelligence techniques for general insurance. Research dissertation for the Actuarial Profession. Available at http://www.actuaries.org.uk/research-and-resources/documents/computational-intelligence-techniques-general-insurance

Park, M.Y. & Hastie, T. (2007). L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society. Series B (Methodological)*, **69**(4), 659–677.

Rasmussen, C.E. & Williams, C.K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Rosasco, L., De Vito, E., Caponnetto, A., Piana, M. & Verri, A. (2004). Are loss functions all the same? *Neural Computation*, **16**(5), 1063–1076.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386–408.

Rumelhart, D.E., Hinton, G.E. & McClelland, J.L. (1987). A general framework for parallel distributed processing. In D.E. Rumelhart, J.L. McClelland, et al., editors, *Parallel Distributed Processing: Volume 1: Foundations*, pages 45–76. MIT Press, Cambridge.

Shapiro, A.F. (2004). Fuzzy logic in insurance. *Insurance: Mathematics and Economics*, **35**(2), 399–424.

Taylor, G. (2008). A simple model of insurance market dynamics. *GIRO proceedings*. Available at http://www.actuaries.org.uk/research-and-resources/documents/simple-model-insurance-market-dynamics-slides

Terzopoulos, D., Tu, X. & Grzeszczuk, R. (1994). Artificial fishes: autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artif. Life,*, **1**(4), 327–351.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, **58**(1), 267–288.

Tychonoff, A.N. (1943). "Об устойчивости обратныхч задач [On the stability of inverse problems]". *Doklady Akademii Nauk SSSR*, **39**(5), 195–198.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, **17**(4), 395–416.

White, H. (2000). A Reality Check for Data Snooping. *Econometrica*.

Yao, J. (2008). Clustering in ratemaking: with application in territories clustering. *Casualty Actuarial Society Discussion Paper Program Casualty Actuarial Society - Arlington, Virginia*, pages 170–192.

Zou, H. & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, **67**, 301–320.

https://doi.org/10.1017/S1748499512000036 Published online by Cambridge University Press