ROYAL INSTITUTE
OF TECHNOLOGY

Predicting customer level risk patterns in non-life insurance

ERIK VILLAUME

Master of Science Thesis Stockholm, Sweden

Abstract

Several models for predicting future customer profitability early into customer life-cycles in the property and casualty business are constructed and studied. The objective is to model risk at a customer level with input data available early into a private consumer's lifespan. Two retained models, one using Generalized Linear Model another using a multilayer perceptron, a special form of Artificial Neural Network are evaluated using actual data. Numerical results show that differentiation on estimated future risk is most effective for customers with highest claim frequencies.

Keywords: Predictive Modeling, Generalized Linear Models, Artificial Neural Networks.

Acknowledgements

I wish to thank my supervisor Henrik Hult for guiding me through the process of writing and conducting the thesis. I would also like to thank Camilla Bjurhult and Karl-Sebastian Lindblad from Customer Analytics at If for for taking a genuine interest in the project.

Contents

List of Figures

1 Introduction

During the past three decades the introduction of direct distribution channels, Customer Relationship Management (CRM) systems and increased customer focus have led to a volatile consumer market in the property and casualty insurance business. Customer turnover rates are historically high. In a harsher financial and economic climate, underwriting profit and customer retention are becoming increasingly important. The days when sufficient profits could be made on investments of premiums alone are gone.

The combination of a volatile consumer market and increased focus on underwriting necessitates a higher level of intelligence in portfolio management and growth. The aim should be to actively seek groups of new customers that can and should be added to the portfolio, as well as conducting up-selling activity on the existing customers most likely to respond *and* with highest probability of being profitable in the future.

The aim of this thesis is to produce a quantitative measure of expected future profitability.

Furthermore, this measure is attempted to be integrated with existing models on probability of purchase. In practice, such a combined score is envisioned to be used as a prioritization within a list of prospects for a given CRM activity. The potential benefits of such a scheme are among others increased hit-rate in campaign activities, reduced costs of acquisition and proactive portfolio pruning of risk.

In this thesis, modeling risk patterns at a customer level and not the usual product-based risk approach is attempted using Generalized Linear Models and Artificial Neural Networks.

The area of focus is first-time customers. We attempt to predict future profitability as early as possible in a customer life-cycle. Attempts on predictions of future customer loyalty as measured by the longevity of active policy coverage has also been made.

This thesis has been conducted at If, the largest non-life insurance company in the Nordic region. It is hereafter referred to as the insurance company.

Outline of the report

The thesis is divided into the following sections:

Section 2 introduces some important concepts from the field of predictive modeling followed by some details on how the models have been implemented and built.

In section 3, the Generalized Linear Model is described and the techniques used to find the estimated parameters are introduced.

Artificial Neural Networks are uncommon in traditional actuarial science, wherefore they are described in some detail in Section 4.

Section 5 presents the results of testing the models against actual data for the different models.

The thesis concludes with Section 6 with a discussion on possible future development.

2 Introductory Notes on Predictive Modeling

Estimation of risk premiums and hence pricing of non-life insurance policies is made by studying historical correlations and trends in large amounts of data. In a broader perspective, this endeavor can be thought of as an application of predictive modeling to a well studied field, with standardized assumptions of distributions and other modeling preliminaries. This section focuses on some of the preliminary steps when building a predictive model. It outlines how the development of the model is made in terms of data preparation and constructing response variables.

2.1 Data Sources and Data Preparation

The principal source of data used in this theses is the company's own records, organized in the traditional form of an exposure table and a claim table. The exposure table contains details on what kind of risk is covered by a given policy, while the claim table describes claims with details such as timing, cost and cause of damage. Additional information pertaining to the customer such as demographic data and financial information from external sources can be added to create a large source table, referred to as the customer summary table.

A number of decisions have to be made during the data preparation step. Decisions such as to what extent should outliers be removed? Claims with no payment, are they to be counted as claims? And at what level should the claim severity be capped? Another decision for the claim severity is whether or not to define it as paid claims or incurred claims which includes case reserves. Further considerations can be made, such as inflation adjustment of historical claim costs versus maintaining the original value. One could include the accident year as a factor so that inflationary effects are reflected in this variable.

Final steps in data preparation are also the ones of a more technical nature, making sure that policies that cancel mid-term, or rather very early after the start date, test rows and internal customers are treated appropriately. Treatment of missing values, grouping of variables and so forth are also questions that arise at this stage.

Once constructed, the customer summary table is then usually separated (randomly) into three subsets: training, validation and testing. The training data is used to construct the model, validation data is used to test the efficiency of the model and important to detect over-fitting in the modeling environment while the final test data set is used to evaluate and compare the predictive power of the models.

2.2 Response Variables

Modeling future customer characteristics in this thesis effectively boils down to one desire:

Differentiate customers based on expectations of profitability

Exact and all-applicable definitions of customer profitability metrics are as elusive as profitability itself. A number of different metrics, seen as response variables, or target variables in predictive analytics merit consideration:

- Claim Frequency
- Customer Duration
- Claim \ No Claim
- Claim Count
- Paid Premiums - Paid Claims
- $\frac{Claim\,Count}{Paid\,Premiums}$

In this thesis implemented models use Claim Frequency. Results for the binary variable Claim \ No Claim are presented as well.

2.3 Sampling Period

Choice of sampling period is important for several reasons. The sample needs to be large enough to build and test the model with enough confidence. It also needs to be representative of the 'normal' situation. Extreme weather, legal changes and other external factors that can affect the response variable can have adverse affects if a model built on unrepresentative data is applied in production.

A second decision to be made at the drawing board is also: to whom should the model be applied in practice. When trying to enhance customer loyalty, it is generally accepted that earlier, better and more targeted communication is better. This time urgency poses some difficulties when modeling: the shorter the customer life-time is, the less the company knows about the customer. This balancing of data availability and urgency to communicate was made in this thesis by choosing first-time customers one year into the customer file.

2.4 Propensity to Buy

The final objective of the thesis was to append a predicted level of risk with existing propensity to buy model, this section briefly discusses how such a model works.

Predictive models on propensity to buy are increasingly used in the business world. The goal is to quantify a given customers probability to complete a purchase in the near future. As the insurance business is similar to subscription based selling, the model is trained on existing customers and the target variable is the binary renewal or no renewal. The model used at the insurance company is a logistic regression on a number of variables to predict a propensity to renew a policy.

2.5 Constructing the Models

When constructing a regression type model, like the generalized linear model, one often uses step-wise inclusion of additional independent variables. An additional variable is included in the model if it increases the overall goodness fit of the model and enhances its predictive powers. A way to visualize the isolated effect of a given independent variable is shown in Figure 1 below. The way in which variables are studied and potentially added in the regression is the following:

1. Look at the model prediction along a potential variable dimension
2. Compare a the results of a regression with and without this variable
3. If the regression including the potential variable is significantly better, include it.
4. Repeat the previous steps while also looking at the effects on previous variables and other measures of fit, like the AIC discussed in Section 3.

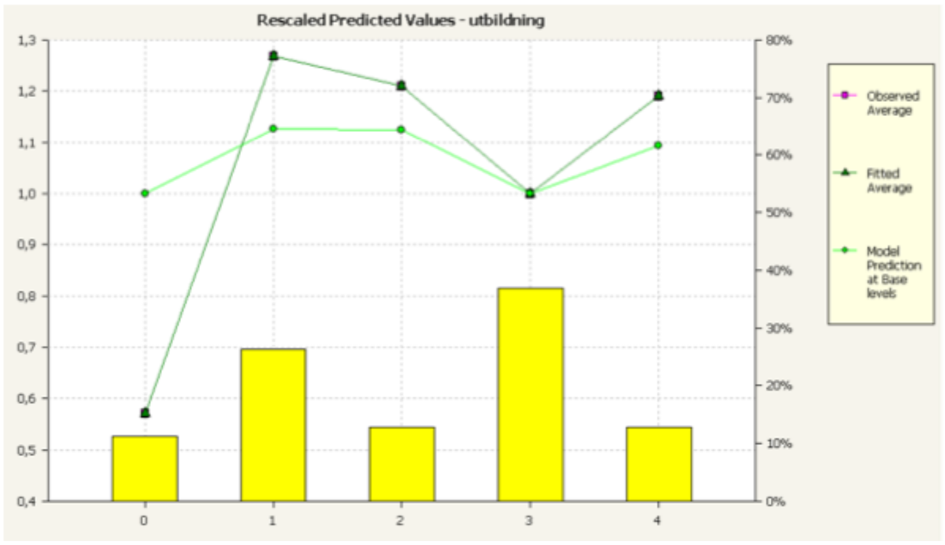


Figure 1: Visualization of the effects of an independent variable

For the ANN, all input variables are used by the model albeit being weighted differently. Building a neural network consists of firstly examining the independence of the input variables and secondly looking at the fit for a given network setup. In the model building phase, one can compare the final prediction errors on the training sets by iteration shown in Figure 2.

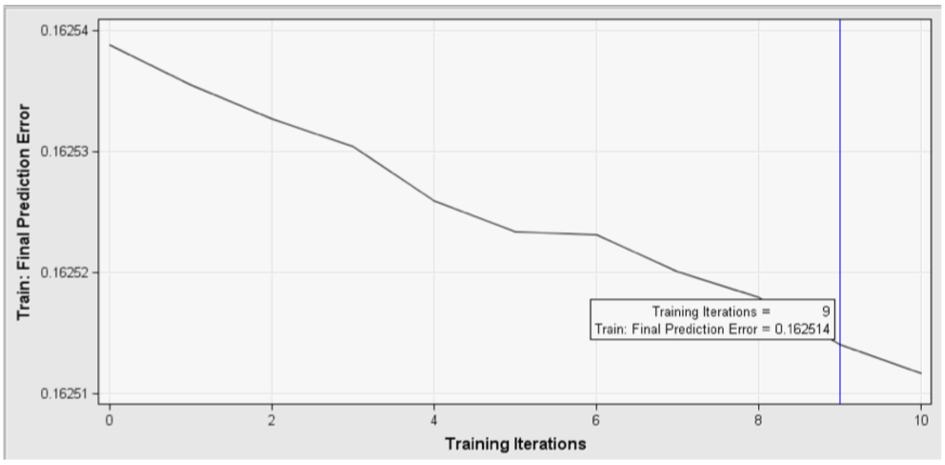


Figure 2: Final prediction error by training iteration

2.6 Comparing GLM and ANN

The models used to predict future profitability of customers in this thesis are introduced separately in Sections 3 and 4 respectively. Comparing the two, the fundamental difference is one of assumptions. The GLM used in this thesis assumes that claim frequencies are poisson distributed. It also assumes a linear relationship between dependent and independent variables. The ANN is built with fewer such assumptions. The claim frequency is not assumed to follow a specific distribution, and no assumptions of linear relationship between independent variables and dependent variable is made. In terms of transparency, the GLM offers more insight and larger room for interpretations on the importance of an independent variable.

3 Generalized Linear Models

Correctly pricing policies in order to generate enough revenue to cover costs of claims, expenses and create profit is the bread-and-butter of the insurance business. In a market full of competitors there is a constant threat of adverse selection. This guarantees that pricing of policies at an increasingly finer level is needed to stay profitable in the long run. In non-life insurance, line of business often entail large portfolios of small and similar risks. Generalized Linear Models (GLMs) are often used in these situations. In this thesis, we apply the GLMs to model the overall profitability of customers across lines of business.

One valid objection to this endeavor is: if actuaries can model the cost and frequency of claims to an acceptable degree, it means that customers are already paying the appropriate premium to cover their expenses, and should therefore by and large be profitable. Nevertheless, we may defend using GLMs to this purpose with the following logic: Insurance companies are allowed to use more variables in prospecting than pricing, and more importantly, pricing is done separately for different products. If we can find groups of people who are more profitable than other on the whole, the company should prioritize them with loyalty building activity. In this section the Generalized Linear Model is introduced, as well as commonly used parameter estimation techniques. Finally some tools for diagnosing and evaluating the goodness of fit of the models are discussed.

3.1 Structure of Generalized Linear Models

The standard multivariate Gaussian regression model expresses observation i of the dependent variable y as a linear function of $(p-1)$ independent variables $x_1, x_2, \ldots, x_{p-1}$ in the following way:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots \beta_{p-1} x_{i(p-1)} + \varepsilon_i.$$

In matrix form, this may be summarized by:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ and $\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \ldots & x_{1(p-1)} \\ 1 & x_{21} & \ldots & x_{2(p-1)} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \ldots & x_{n(p-1)} \end{pmatrix}$.

Let also $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_{p-1})^T$ be the vector containing the p parameters to be estimated and $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^T$ denote the residuals. These residual components are assumed to be independent and normally $(\mathcal{N}(0, \sigma^2))$ distributed.

In GLMs, this assumption is relaxed, allowing for a larger ensemble of distributions for the error term. A GLM is defined by:

$$\mathbb{E}[\mathbf{y}] = \boldsymbol{\mu} = g^{-1}(\boldsymbol{\eta}), \tag{1}$$

where $g(\cdot)$ is referred to as a link function and y follows a distribution from the exponential family. The linear predictor, $\boldsymbol{\eta}$ is defined by $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$.

3.2 Exponential Family

A distribution is a part of the exponential family if its density or probability mass function can be written in the following form:

$$f(y; \theta, \phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right),$$

where functions $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$ determine the parametric subfamily for a given distribution. The canonical and dispersion parameters are denoted θ and ϕ respectively. Constraints on the function b are limited to it being twice differentiable and convex.

If Y is a member of the exponential family, the following is true:

$$\mathbb{E}[Y] = b'(\theta)$$
$$\text{Var}[Y] = a(\phi) b''(\theta).$$

To see that this holds, one can evaluate the following differentials of the likelihood function of a distribution from the exponential family:

$$\frac{\partial l(\theta,\phi;y)}{\partial\theta} = \frac{\partial}{\partial\theta}\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right) = \frac{y - b^{'}(\theta)}{a(\phi)}$$
$$\frac{\partial^2 l}{\partial\theta^2} = -\frac{b''(\theta)}{a(\phi)}.$$

Since we know that $\mathbb{E}(\frac{\partial l}{\partial\theta}) = 0$ we obtain that $\mathbb{E}[y] = \mu = b'(\theta)$ from the first differential above. Another property of the Maximum Likelihood Estimator(MLE) is that $\mathbb{E}[\frac{\partial^2 l}{\partial\theta^2}] + \mathbb{E}[(\frac{\partial l}{\partial\theta})^2] = 0$. From this we obtain that

$$\mathrm{Var}[y] = a(\phi)b''(\theta).$$

Since b' is invertible, this means that θ is a function of $\mathbb{E}[Y]$. Since c is not a function of θ, by extension it cannot depend on $\mathbb{E}[Y]$. Equation (1) makes estimation of this function irrelevant when model parameters are estimated.

The expression of the variance of Y as a function of θ, fittingly named the variance function links the mean and variance of the distributions in the exponential family. One often lets $V = b''(\theta)$.

Link function

A number of link functions have the desirable property that $g(\mu) = \theta$, and are then referred to as canonical links. A few such functions are listed below for commonly seen distributions. It is interesting to note that although these links often are used by default in statistical software, there is no *a priori* reason that on the whole a canonical link is better than an alternative link function without this statistical property.

Distribution	Variance Function Var$[x]$	Canonical link $g(x)$
Poisson	μ	$\log(x)$
Gaussian	1	x
Bernoulli	$\mu(1-\mu)$	$\log(\frac{x}{1-x})$
Gamma	μ^2	$\frac{1}{x}$

Table 1: Table of distributions in the Exponential Family and commonly used link functions

3.3 GLM Parameter Estimation

Parameter estimation is often done by means of the Maximum Likelihood method. The estimates are thus taken as the values that maximize the log likelihood, which can be written as (for a single observation):

$$l = log[L(\theta, \phi; y)] = \frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi).$$

Differentiation of l with respect to the parameters, the elements of $\boldsymbol{\beta}$ yields

$$\frac{\partial l}{\partial \beta_j} = \frac{\partial l}{\partial \theta} \frac{\partial \theta}{\partial \mu} \frac{\partial \mu}{\partial \eta} \frac{\partial \eta}{\partial \beta_j}.$$

Using the knowledge that $b^{''} = V$ and that $\eta = \sum_j \beta_j x_j$, we get the expression,

$$\begin{aligned} \frac{\partial l}{\partial \beta_j} &= \frac{y - \mu}{a(\theta)} \frac{1}{V} \frac{\partial \mu}{\partial \eta} x_j \\ &= \frac{W}{a(\theta)} (y - \mu) \frac{\partial \mu}{\partial \eta} x_j, \end{aligned}$$

where W is defined by

$$W^{-1} = \left(\frac{\partial \eta}{\partial \mu}\right)^2 V.$$

Recall that the likelihood above has been written for a single observation. The likelihood equation for a given parameter, β_j, is given by

$$\sum_i \frac{W_i(y_i - \mu_i)}{a(\theta)} \frac{\partial \mu_i}{\partial \eta_i} x_{ij}.$$

The MLE of the parameter vector is asymptotically multivariate normally distributed $\mathcal{N}(\theta, I_\theta^{-1})$. The asymptotic covariance matrix is given by the inverse of the Fisher information matrix, I_θ, which has the elements,

$$I_{j,k} = \mathbb{E}\left[\left(\frac{\partial l}{\partial \theta_j}\right)\left(\frac{\partial l}{\partial \theta_k}\right)\right].$$

In general there are no closed form solutions for the Maximum Likelihood Estimation problem for GLMs. In practice, numerical algorithms are used. The likelihood function can be written as:

$$L(y, \theta, \phi) = \prod_{j=1}^{n} f(y_j; \theta_j, \phi) = \prod_{j=1}^{n} c(y_j, \phi) \exp\left(\frac{y_j \theta(\beta, x_j) - a(\theta(\beta, x_j))}{\phi}\right).$$

A commonly used approach is the iteratively re-weighted least squares approach, described in [10]. One can summarize the process with the following steps:

1. Linearization of the link function. E.g. first order Taylor series. $g(y) \approx z$. Where $z = g(\mu) + (y - \mu) g^{'}(\mu)$.

2. Let η_0 be the current estimate of the linear predictor, and let $\hat{\mu}_0$ be the corresponding fitted value derived from the link function $\eta = g(\mu)$. Form the adjusted dependent variate $z_0 = \eta_0 + (y - \mu_0)(\frac{d\eta}{d\mu})|_{\mu=\hat{\mu}_0}$.

3. Calculate the weight matrix W from $W_0^{-1} = (\frac{d\eta}{d\mu})^2 V_0$, where V denotes the variance functions.

4. Apply a weighted regression of z on predictors $x_1, x_2, \ldots, x_n$ using weights W_0. This gives an updated estimate $\hat{\beta}_1$ from which an updated estimate of the linear predictor, $\hat{\eta}_1$ is produced.

5. Repeat steps 1-4 until stop conditions apply.

3.4 Assessing the Fit of the Model

One way to benchmark the goodness of fit of any model is to compare it to the deviance. It can be defined as:

$$D = 2(l(y, \phi; y) - l(\hat{\mu}, \phi; y)).$$

In the special case of the Normal distribution, the deviance is equal to the residual sum of squares. If the model is to be considered reasonable, the deviance should tend asymptotically towards a χ^2 -distribution as n increases.

Another goodness of fit measure is the generalized Pearson χ^2 statistic, which can be defined by

$$\chi^2 = \sum_i \frac{(y_i - \mu)^2}{\hat{V}(\hat{\mu})},$$

where $\hat{V}(\hat{\mu})$ refers to the estimated variance function. Plots of the residuals discussed above against the fitted value should show a 'random' pattern with constant range and mean equal to zero. Erroneous link functions and omitted non-linear terms in the linear predictor may explain any deviance from such a plot.

A measure of fit often used in model selection is

$$D_C = D - \alpha q \phi,$$

where D is the deviance, q the number of parameters and ϕ is the dispersion parameter. For values of $\alpha = 2$, this measure is referred to as Akaike's Information Criterion (AIC). The models with small values of D_C are preferred over the ones with larger values.

4 Artificial Neural Networks

Artificial Neural Networks can be thought of as machines designed to model the behavior of biological neurons. In this thesis they are used to predict future customer behavior through a process of supervised learning. The aim of this section is to describe the way in which the multi-layer perceptron network functions and how the model is constructed. The outline of this section is as follows:

First we start by introducing the basic component of a neural network, the neuron. The second part relates to linear adaptive filters. This related field is evoked in the hopes that the multi-layer perceptron and the back-propagation algorithm will be easier to understand having first studied adaptive filters, the Least-Mean-Squares algorithm and reviewed some optimization techniques. Third and fourth part treat the single and multi-layer perceptron networks. As the field of ANNs is vast, our goal is only to emphasize the most important features used during this project.

4.1 Introductory Notes on Artificial Neural Networks

Artificial Neural Networks can be regarded as an adaptive machine. We may define it by quoting [6] as: *A neural network is a massively parallel distributed processor made up of simple processing units, which have a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two aspects:*

1. *Knowledge is acquired by the network from its environment through a learning process.*

2. *Interneuron connection strength, known as synaptic weights, are used to store the acquired knowledge.*

The essential building block of a neural network, the neuron, can be schematically be modeled in Figure 3.

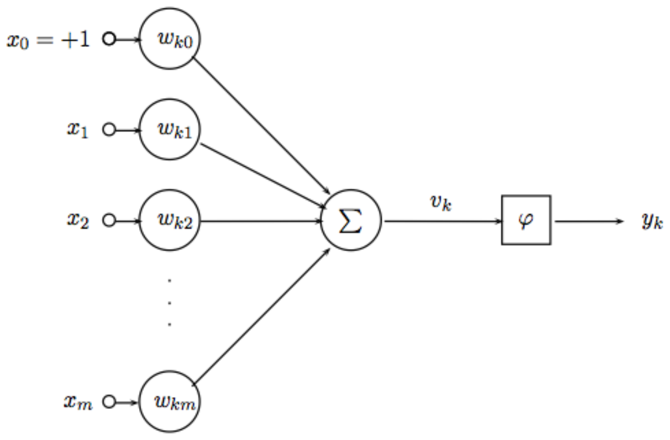


Figure 3: Signal-flow chart of a perceptron

Key elements of the model are:

1. Synapses, which determine the importance of a given input signal for a given neuron by means of a synaptic weight. Synaptic weight w_{kj} is the multiplying factor of signal x_j in neuron k. In general, the weights can take negative and positive values.

2. Linear combiner, or adder for summation of the input signals weighted by the respective synapses.

3. Activation function for limiting the output amplitude. Typically the range of permissible output is [0,1] or [-1,1].

A neuron can also be summarized by the equations:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

and

$$y_k = \varphi(u_k + b_k),$$

where $x_1, x_2, \ldots, x_m$ denotes the input signals, $w_{k1}, w_{k2}, \ldots, w_{km}$ are the synaptic weights, u_k is the linear combiner output, $\varphi(\cdot)$ is the activation function and $b_k = w_{k0}$ is an externally applied bias. The bias has the effect of applying an affine transformation to the linear combiner output. $v_k = u_k + b_k$. v_k is called *induced local field.* Note that the bias is considered to an external parameter of artificial neuron k.

The activation function can be of various forms. Three basic forms are Heaviside function, piecewise-linear functions and the most commonly used sigmoid function. The most commonly used sigmoid function is the logistic function $\varphi(v) = \frac{1}{1+e^{-av}}$ where the parameter a determines the slope.

There are many different types of architectures and forms of neural networks. In this thesis, the feedforward multilayer perceptron was used. A feedforward network does not allow cycles from later layers to previous, it feeds input in a fixed forward direction.

4.2 Adaptive Filters

Adaptive filters are applied to a given dynamic system of unknown mathematical form. Known features of the system are limited to input and output data generated by the system at regular time intervals. In the case when inputs are $\mathbf{x}(i)$, the system response is the scalar $d(i)$, where $i = 1, 2, \ldots, n, \ldots$ denotes the time. The adaptive filters seeks to replicate this system to the fullest extent possible.

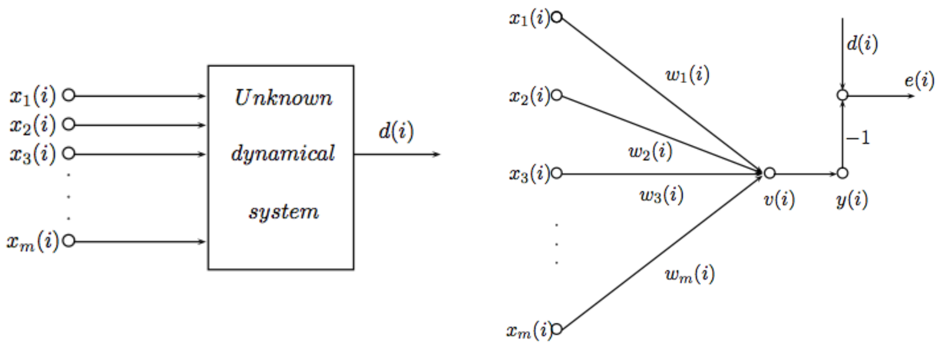


Figure 4: The adaptive filter to the right seeks to replicate the dynamical system on the left

The external behavior of the system is described by the data set:

$$\mathfrak{T} = \{\mathbf{x}(i), d(i)\,; i = 1, 2, \ldots, n, \ldots\},$$

where

$$\mathbf{x}(i) = (x_1(i), x_2(i), \ldots, x_m(i))^T.$$

The components of $\mathbf{x}$ are iid samples from some unknown distribution. The neuronal model can be described as an adaptive filter, whose operation consists of two continuos processes.

1. Filtering process:
   (a) Computing the output signal $y(i)$ that is produced as a response of the stimulus vector $\mathbf{x}(i) = (x_1(i), x_2(i), \ldots, x_m(i))$.
   (b) Computing an error signal, $e(i) = y(i) - d(i)$, where $d(i)$ denotes the target signal.
2. Adaptive process, where the synaptic weights are adapted in accordance with the error signal $e(i)$.

The combination of these processes results in a feedback loop around the neuron.

In Figure 4, the output at time i, $y(i)$ can be written as:

$$y(i) = v(i) = \sum_{k=1}^{m} w_k(i)x(i),$$

where $w_1(i), w_2(i), \dots, w_m(i)$ are the synaptic weights measured at time i. The manner in which the error signal is used to control the adjustments to the synaptic weights is determined in the cost function used in the adaptive filtering. This area takes natural influences from optimization theory, which is why a few unconstrained optimization techniques are discussed below. We refer to [9] for further details from the field of optimization.

4.3 Unconstrained Optimization Techniques

Consider a cost function $\mathscr{C}(\mathbf{w})$ which is a continuously differentiable function of some unknown weight vector $\mathbf{w} \in \mathbb{R}^m$. We want to solve the unconstrained optimization problem:

Find $\mathbf{w}^\star$ such that

$$\mathscr{C}(\mathbf{w}^\star) \leq \mathscr{C}(\mathbf{w}).$$

The necessary condition for optimality is the usual:

$$\nabla(\mathbf{w}^\star) = 0.$$

Iterative descent methods are generally useful in the context of adaptive filtering. The main idea is to start with an initial guess, $w(0)$ and generate a sequence of $w(1), w(2), \dots,$ such that the cost function is reduced iteratively, i.e. that

$$\mathscr{C}(\mathbf{w}(n+1)) \leq \mathscr{C}(\mathbf{w}(n)). \tag{2}$$

Three unconstrained optimization methods related to the iterative descent methods are presented below.

Method of steepest descent

The method of steepest descent updates the weights in a direction opposed to the gradient of the cost function $\mathscr{C}$, updating the weights each step by

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta\nabla\mathscr{C}(\mathbf{w}), \tag{3}$$

where η is the step size, or learning-rate parameter. In the steepest descent algorithm, the update from iteration n to $n+1$ is given by:

$$\Delta\mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = -\eta\nabla\mathscr{C}(\mathbf{w}).$$

To see that this method iteratively reduces the error in accordance with Equation (2), one can linearize around $\mathbf{w}(n)$ for small η. Denoting $\mathbf{g}(n) = \nabla\mathscr{E}(\mathbf{w})$, one gets the expansion:

$$\begin{aligned}\mathscr{E}(\mathbf{w}(n+1)) &\approx \mathscr{E}(\mathbf{w}(n) + \mathbf{g}^T(n)\Delta\mathbf{w}(n)) \\ &\mathscr{E}(\mathbf{w}(n) - \eta\mathbf{g}^T(n)\mathbf{g}(n)) \\ &\mathscr{E}(\mathbf{w}(n) - \eta||\mathbf{g}(n)||^2).\end{aligned} \tag{4}$$

This shows that for positive values of η the cost function is reduced in each iteration.

The learning-rate parameter η has a substantial influence on the convergence of the steepest descent method. If it is too large the algorithm becomes unstable and the algortithm diverges.

Newton's Method

Newton's method aims at minimizing the quadratic approximation of the cost function around the current point. Using specifically a second-order Taylor expansion of the cost function, one can write:

$$\begin{aligned}\Delta\mathscr{E}(\mathbf{w}(n)) &= \mathscr{E}(\mathbf{w}(n+1)) - \mathscr{E}(\mathbf{w}(n)) \\ &\approx \mathbf{g}^T(n)\Delta\mathbf{w}(n) + \frac{1}{2}\Delta\mathbf{w}^T(n)\mathbf{H}(n)\mathbf{w}(n),\end{aligned} \tag{5}$$

$$\text{where } \mathbf{g} = \begin{pmatrix} \frac{\partial\mathscr{E}}{\partial w_1} \\ \frac{\partial\mathscr{E}}{\partial w_2} \\ \vdots \\ \frac{\partial\mathscr{E}}{\partial w_m} \end{pmatrix} \text{ and } \mathbf{H} = \begin{pmatrix} \frac{\partial^2\mathscr{E}}{\partial w_1^2} & \frac{\partial^2\mathscr{E}}{\partial w_1 w_2} & \cdots & \frac{\partial^2\mathscr{E}}{\partial w_1 w_m} \\ \frac{\partial^2\mathscr{E}}{\partial w_2 w_1} & \frac{\partial^2\mathscr{E}}{\partial w_2^2} & \cdots & \frac{\partial^2\mathscr{E}}{\partial w_2 w_m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2\mathscr{E}}{\partial w_m w_1} & \frac{\partial^2\mathscr{E}}{\partial w_m w_2} & \cdots & \frac{\partial^2\mathscr{E}}{\partial w_m^2} \end{pmatrix}$$

The update in weights that minimize the error is found by differentiating (5) with respect to $\Delta\mathbf{w}$ and solving

$$\frac{\partial\Delta\mathscr{E}(\mathbf{w}(n))}{\partial\Delta\mathbf{w}(n)} = \mathbf{g}^T(n) + \mathbf{H}(n)\mathbf{w}(n) = 0.$$

The update is that satisfies this is $\Delta\mathbf{w}(n) = -\mathbf{H}^{-1}(n)\mathbf{g}(n)$

Newton's method for updating the weights can thus be summarized by

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \Delta\mathbf{w}(n) \\ &= \mathbf{w}(n) - \mathbf{H}^{-1}(n)\mathbf{g}(n).\end{aligned}$$

For Newton's method to work, the Hessian, $\mathbf{H}(n)$ has to be a positive definite matrix for all n. Unfortunately, there is no guarantee that this is true for all steps of the algorithm. We refer to [6] for further explanations on how to handle this.

Gauss-Newton Method

The Gauss-Newton is applicable to cost functions expressed as the sum of squared errors,

$$\mathscr{C}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n} e^2(i).$$

The error term in this method are evaluated around a fix weight vector $\mathbf{w}$ during for the observations $1 \leq i \leq n$. The error signal is evaluated using

$$\mathbf{e}^{'}(n,\mathbf{w}) = \mathbf{e}(n) + \mathbf{J}(n)(\mathbf{w} - \mathbf{w}(n)), \tag{6}$$

where $\mathbf{J}(n)$ is the $n \times m$ Jacobian matrix, $\mathbf{J}(n) = \begin{pmatrix} \frac{\partial e(1)}{\partial w_1} & \frac{\partial e(1)}{\partial w_2} & \cdots & \frac{\partial e(1)}{\partial w_m} \\ \frac{\partial e(2)}{\partial w_1} & \frac{\partial e(2)}{\partial w_2} & \cdots & \frac{\partial e(2)}{\partial w_m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e(n)}{\partial w_1} & \frac{\partial e(n)}{\partial w_2} & \cdots & \frac{\partial e(n)}{\partial w_m} \end{pmatrix}_{\mathbf{w}=\mathbf{w}(n)}$

The update of the weights is done by:

$$\mathbf{w}(n+1) = \arg\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{e}^{'}(n,\mathbf{w})\|^2 \right\}.$$

Using (6) we can evaluate this as

$$\begin{aligned}\frac{1}{2}\|\mathbf{e}^{'}(n,\mathbf{w})\|^2 =& \frac{1}{2}\|\mathbf{e}(n)\|^2 + \mathbf{e}^T(n)\mathbf{J}(n)(\mathbf{w} - \mathbf{w}(n)) \\ &+ \frac{1}{2}(\mathbf{w} - \mathbf{w}(n))^T \mathbf{J}^T(n)\mathbf{J}(n)(\mathbf{w} - \mathbf{w}(n)).\end{aligned}$$

Differentiating and solving for $\mathbf{w}$, one gets the expression

$$\mathbf{w}(n+1) = \mathbf{w}(n) - (\mathbf{J}^T(n)\mathbf{J}(n))^{-1}\mathbf{J}^T(n)\mathbf{e}(n). \tag{7}$$

For the Gauss-Newton iteration to be computable, $\mathbf{J}^T(n)J(n)$ needs to be nonsingular. This means that it has to have rank n. If this matrix is found to be rank deficient, one can add a diagonal matrix to ensure linearly independent rows. The method uses, in a modified form the following updated weights:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - (\mathbf{J}^T(n)\mathbf{J}(n) + \delta\mathbb{I})^{-1}\mathbf{J}^T(n)\mathbf{e}(n).$$

The effect of this modification is reduced as the iteration increases.

4.4 Linear Least-Squares Filter

We may define the error signal as:

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{X}(n)\mathbf{w}(n),$$

where $\mathbf{d}(n)$ is the desired response vector, known when training the model. Differentiating the equation above with respect to $\mathbf{w}(n)$ yields

$$\begin{aligned} \nabla\mathbf{e}(n) &= -\mathbf{X}^T(n) \\ \mathbf{J}(n) &= -\mathbf{X}(n). \end{aligned}$$

We now show that the Gauss-Newton method converges in one iteration. Substituting the expressions for the Jacobian and error terms into equation (7) we get,

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + (\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\mathbf{X}^T(n)(\mathbf{d}(n) - \mathbf{X}(n)\mathbf{w}(n)) \\ &= \mathbf{X}^{+}\mathbf{d}(n). \end{aligned}$$

The matrix $\mathbf{X}^{+}(n) = (\mathbf{X}^T(n)\mathbf{X}(n))^{-1}\mathbf{X}^T(n)$ denotes the *pseudoinverse.*

4.5 Least-Mean-Square Algorithm

The Least-Mean-Square Algorithm (LMS) uses the error signal in its cost function,

$$\mathscr{C}(\mathbf{w}) = \frac{1}{2}e^2(n).$$

As with the Least Squares filter, the LMS uses a Linear Neuron model, ensuring that we can write

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n).$$

Differentiation of $\mathscr{C}(\mathbf{w})$ gives

$$\frac{\partial\mathscr{C}(\mathbf{w})}{\partial\mathbf{w}} = e(n)\frac{\partial e(n)}{\partial\mathbf{w}}.$$

Hence,

$$\frac{\partial\mathscr{C}(\mathbf{w})}{\partial\mathbf{w}} = -\mathbf{x}(n)e(n).$$

The last relation can be used as an estimate for the gradient, $\mathbf{g}$. The LMS algorithm uses the method of steepest descent to find the updated weights. The LMS algorithm, using (3) may then be written as

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \eta\mathbf{x}(n)e(n).$$

It can be shown that the feedback loop around the weight vector $\hat{\mathbf{w}}$ above behaves like a low-pass filter, letting low frequency components of the error signal through while attenuating the higher ones. The average time constant of the filter is inversely proportional to the learning-rate parameter η. This means that small values of η means slower progress of the algorithm, while the accuracy of the filtering improves.

4.6 Single-Layer Perceptron

The perceptron, originally introduced in by Rosenblatt in 1958, is a simple form of a neural network used for binary classifications of patterns that are said to be linearly separable. It is built around the non-linear neuron, the McCulloch–Pitts model in which the linear combiner is followed by a hard delimiter (signum) as activation function. In essence, it consists of a single neuron with adaptable synaptic weights and bias. The usage of a single perceptron, limited to classification into two classes can be expanded to allow for classification in the presence of several classes by adding parallel perceptrons. To simplify notation the single-layer network consisting only of a single perceptron is presented rather than a network of several neurons within the same layer is presented in this section. Expansion to the latter case is readily made by simply writing more. It is important to note (see [6]) that even with other nonlinear choices of delimiter functions, successful usage of the perceptron is limited to cases when we seek to organize in the presence of linearly separable patterns. A signal-flow representation of a perceptron is shown in Figure 5.

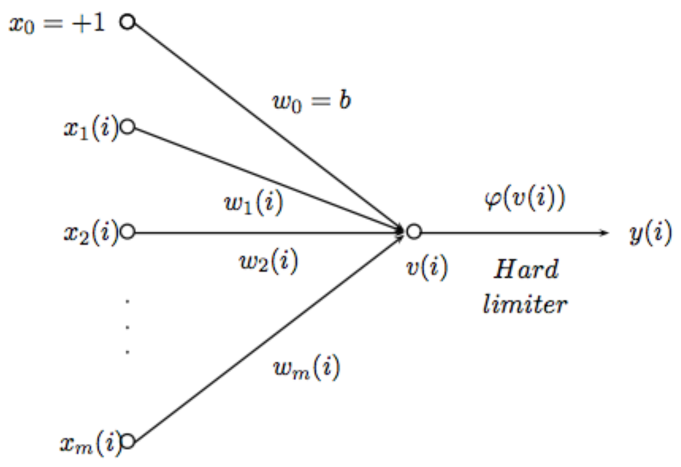


Figure 5: Signal-flow representation of a perceptron

The goal of the perceptron is to correctly classify the set of external simuli $x_1, x_2, \ldots, x_m$ into one of two classes $\mathscr{C}_1, \mathscr{C}_2$. Since the perceptron uses the hard delimiter to classify the inputs, the decision making is made with the following rule:

$$\begin{cases} \mathbf{x} \in \mathscr{C}_1 & \text{if} \quad v = \sum_{i=1}^{m} w_i x_i + b > 0 \\ \mathbf{x} \in \mathscr{C}_2 & \text{if} \quad v = \sum_{i=1}^{m} w_i x_1 + b < 0 \end{cases}$$

In a simplistic form of a perceptron application the classification can be thought to be spanned by two classification regions, separated by the hyperplane defined by

$$\sum_{i=1}^{m} w_i x_i + b = 0. \tag{8}$$

An illustration of the case with two inputs, in which the hyperplane is a line is shown in figure 6.

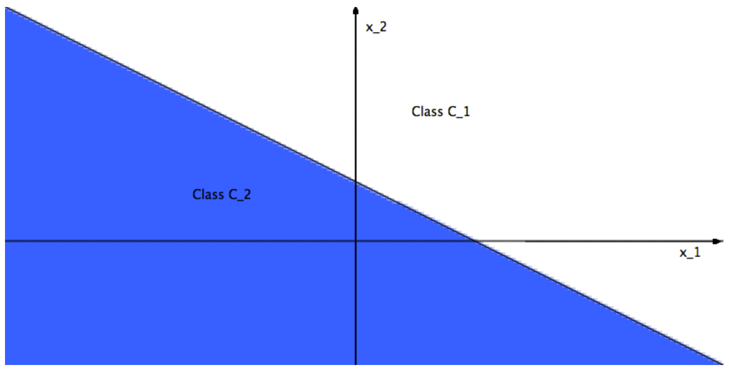


Figure 6: Decision regions divided by the hyperplane, a line in the case of two inputs

4.7 Relation Between Naive Bayes Classification and the Perceptron

When the environment is Gaussian the perceptron is equivalent to a linear classifier, the same form taken by a Bayesian classifier in the same environment. We present this special case below. We refer to [6] for more details on learning perceptrons.

4.8 Bayes Classifier

The Bayesian classification scheme aims at reducing the average risk. For a two-class problem, we can define this as:

$$\begin{aligned} \mathscr{R} = {} & c_{11} p_1 \int_{\mathscr{X}_1} f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_1) d\mathbf{x} + c_{22} p_2 \int_{\mathscr{X}_2} f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_2) d\mathbf{x} \\ & + c_{21} p_1 \int_{\mathscr{X}_2} f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_1) d\mathbf{x} + c_{12} p_2 \int_{\mathscr{X}_1} f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_2) d\mathbf{x}, \end{aligned}$$

where

- p_i denotes the *a priori probability* that the observation vector $\mathbf{x}$ is drawn from subspace $\mathscr{X}_i$
- c_{ij} is the cost we assign of deciding that $\mathbf{x}$ is drawn from $\mathscr{X}_i$, when it is fact drawn from $\mathscr{X}_j$. It is natural to assign values of c such that correct classification has a lower cost than erroneous, i.e. that $c_{11} < c_{12}$ and $c_{22} < c_{21}$
- $f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_i)$ refers to the conditional probability density function of $\mathbf{X}$ given that the observed vector is drawn from subspace i.

Since the subspaces form a partition of the total space, we can reformulate the average risk as

$$\begin{aligned}\mathscr{R} = c_{11}p_1 \int_{\mathscr{X}_1} f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_1)d\mathbf{x} + c_{22}p_2 \int_{\mathscr{X}\setminus\mathscr{X}_1} f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_2)d\mathbf{x} \\ + c_{21}p_1 \int_{\mathscr{X}\setminus\mathscr{X}_1} f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_1)d\mathbf{x} + c_{12}p_2 \int_{\mathscr{X}_1} f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_2)d\mathbf{x},\end{aligned}$$

or furthermore as,

$$\mathscr{R} = c_{22}p_2 + c_{21}p_1 + \int_{\mathscr{X}_1} \Big[p_2(c_{12} - c_{22})f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_2) - p_1(c_{21} - c_{11})f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_1)\Big] d\mathbf{x}.$$

A study of the average risk expressed in the latter forms allows for the following deduction of a path towards an optimum (minimum) value:

1. Assigning all values of $\mathbf{x}$ for which the integrand is negative to class $\mathscr{C}_1$ lowers the average risk.
2. Assigning all the values of $\mathbf{x}$ for which the integrand is positive to class $\mathscr{C}_2$ lowers the average risk, as these values would then add zero to the overall risk.
3. Values of $\mathbf{x}$ for which the integrand is zero has no effect, and can be mandated to class $\mathscr{C}_2$.

Following this recipe, the Bayes Classification can then be compressed as the following rule: *If*

$$\boxed{p_2(c_{12} - c_{22})f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_2) < p_1(c_{21} - c_{11})f_{\mathbf{X}}(\mathbf{x} \mid \mathscr{C}_1)}$$

Assign this observation to class $\mathscr{C}_1$, otherwise assign it to class $\mathscr{C}_2$

4.9 Multilayer Perceptrons

Multilayer perceptrons (MLP) is a natural extension of the single layer perceptron network reviewed earlier. It is characterized by a forward flow of inputs passing through subsequent hidden or computational layers composed by perceptron neurons. The usage of MLPs is defended by the fact that they are able to predict and detect more complicated patterns in data. In this section we will describe the back-propagation algorithm used in this thesis to train the network. In essence the back-propagation algorithm consists of two steps;

1. Step 1, forward pass: the inputs are passed through the network, layer by layer and an output is produced. During this step the synaptic weights are fixed.
2. Step 2, backward pass: the output from step 1 is compared to the target, producing an error signal that is propagated backwards. During this step the aim is to reduce the error in a statistical sense by adjusting the synaptic weights according to a defined scheme.

The multilayer perceptron has the following characteristics:

1. All neurons within the network features a nonlinear activation function that is differentiable everywhere.
2. The network has one or more hidden layers, made up of neurons that are removed from direct contact with input and output. These neurons calculate a signal expressed as a nonlinear function of its input with synaptic weights and an estimate of the gradient vector.
3. There is a high degree of interconnectivity within the network.

4.10 Back-Propagation Algorithm

At iteration n (the n:th row in the training set) we may calculate the error, for neurons in the output layer as

$$e_j(n) = d_j(n) - y_j(n). \tag{9}$$

The error energy for the entire network is defined by

$$\mathscr{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n), \tag{10}$$

where C denotes the set of neurons in the output layer. The average error energy for an entire training set is given by

$$\mathscr{E}_{AV} = \frac{1}{N} \sum_{n=1}^{N} \mathscr{E}(n). \tag{11}$$

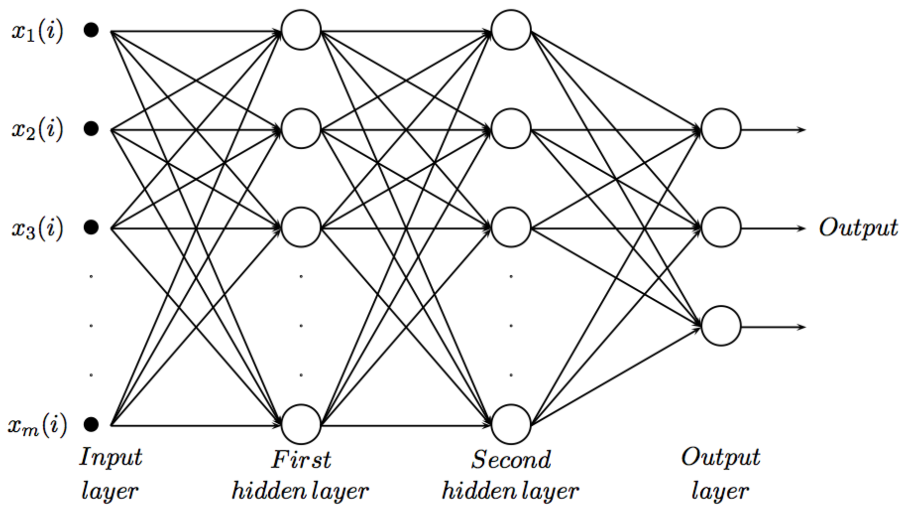


Figure 7: Example of a Multilayer Perceptron with two hidden layers

For a given training set, $\mathscr{C}_{AV}$ represents a cost function, a measure of learning performance. The goal is to adjust the free parameters such as the bias and the synaptic weights to minimize this cost.

Consider again a neuron j in the output layer. We may express its output as

$$\begin{aligned} v_j &= \sum_{i=0}^{m} w_{ji}(n) y_i(n) \\ y_j &= \varphi_j(v_j(n)). \end{aligned}$$

As in the LMS algorithm, the back-propagation algorithm applies a weight adjustment $\Delta w_{ji}(n) \propto \frac{\partial \mathscr{C}(n)}{\partial w_{ji}(n)}$, where

$$\frac{\partial \mathscr{C}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{C}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{12}$$

Plugging these straight forward differentiations made on the equations of this section yields:

$$\frac{\partial \mathscr{C}(n)}{\partial w_{ji}(n)} = -e_j(n) y_i(n) \varphi_j'(v_j(n)). \tag{13}$$

As in the steepest descent method, the update applied to the weights is made by

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathscr{C}(n)}{\partial w_{ji}(n)} = \eta \delta_j(n) y_i(n),$$

where

$$d_j = -\frac{\partial \mathscr{C}(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n)). \tag{14}$$

In (14) d_j is called the local gradient and η is again a learning rate parameter.

The error signal $e_j(n)$ is used explicitly in these expressions for updating the synaptic weights, which creates the following two situations for obtaining its value depending on where the neuron j is located within the network.

Output layer

In this case the error signal is calculated by (9) as the target response d is directly available. The local gradient is obtained using equation 14

Hidden layer

For a neuron located in the hidden layer the desired response is not directly accessible, creating the need for recursive iteration over all the neurons it is connected to. We now focus on describing how this can be done: Using knowledge above we can write

$$\begin{aligned} d_j(n) &= -\frac{\partial \mathscr{C}(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= -\frac{\partial \mathscr{C}(n)}{\partial y_j(n)}\varphi_j'(v_j(n)). \end{aligned} \tag{15}$$

The second term $\varphi_j'(v_j(n))$ is directly known from the activation and local induced field of hidden neuron j. The first, term can be evaluated using (10). Using another dummy index, k to indicate that the summation of the error energy is made through summation over output neurons, we get

$$\begin{aligned} \frac{\partial \mathscr{C}(n)}{\partial y_j(n)} &= \frac{\partial}{\partial y_j}\sum_k e_k^2(n) \\ &= \sum_k e_k(n)\frac{\partial e_k(n)}{\partial y_j(n)} \\ &= \sum_k e_k(n)\frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)}. \end{aligned} \tag{16}$$

From equation (9) we see that since k is an output neuron

$$\begin{aligned} \frac{\partial}{\partial v_k(n)}e_k(n) &= \frac{\partial}{\partial v_k(n)}\Big(d_k(n) - y_k(n)\Big) \\ &= \frac{\partial}{\partial v_k(n)}(d_k(n) - \varphi_k(v_k(n))) \\ &= -\varphi_k'(v_k(n)). \end{aligned} \tag{17}$$

The induced field for neuron k can be expressed as the weighted sum (including the bias for this neuron as $w_k 0 = b_k$) all its input from the previous layers:

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n). \tag{18}$$

Differentiating yields

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n). \tag{19}$$

The local gradient for a hidden neuron j can thus be expressed using (15 ,16 ,17 19) as,

$$d_j(n) = \varphi_j'(v_j(n)) \sum_k e_k(n) \varphi_k'(v_k(n)) w_{kj}(n). \tag{20}$$

Recognizing the expression in the sum as the local gradient as defined in (14) we can rewrite this last expression as

$$d_j(n) = \varphi_j'(v_j(n)) \sum_k d_k(n) w_{kj}(n). \tag{21}$$

Equation (21) is finally the back-propagation formula for the local gradient for a hidden neuron. The update in weights is made using

$$\Delta w_{ji} = \eta \delta_j(n) y_i(n). \tag{22}$$

Summary of the back-propagation algorithm

Two computational steps are done in the back-propagation algorithm. The forward pass consists, left to right propagation through the network using fixed synaptic weights throughout. The output signals are calculated for all neurons individually using

$$y_i = \varphi(v_j(n)),$$

where v_j is the induced local field of neuron j, is given by

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n),$$

where m is the number of inputs to neuron j and y_i is the input signal. In the special case when neuron j is either in the input or output layer, $y_i(n) = x_i(n)$ If j is in the output layer, the output signal $y_j(n)$ is compared to the target value $d(i)$, rendering the error $e_j(n)$. The backward pass goes right to left, starting at the output layer, recursively calculating the local gradient δ and updating the synaptic weights using (22).

When training a feedforward multilayer perceptron, there are two ways in which the the back-propagation can be implemented. Using it incrementally, the algorithm is used for every training input. In the second, batch mode, all training examples are supplied to the network before the weights are reevaluated. The fixed weights in the forward pass are obtained using a given initialization method, usually a random sample from a distribution with zero mean.

Highly simplified, the training is made following the steps:

- Initialize weights (e.g. small random numbers)
- Pass a record through the network and calculate output.
- Update weights proportional to the error in output, propagate errors backwards from output layer to first hidden layer.
- Repeat for each record until stop conditions apply.

5 Results

This section presents results for two implemented models: a Generalized Linear Model and feedforward multilayer perceptron. The GLM model uses Claim Frequency as response variable, whereas the Neural Network has been developed using claim frequency as well as the binomial claim/no claim as response variables. The chapter is organized in the following way: first results for the predicted claim frequency from Generalized Linear regression model and the Artificial Neural Network are compared. Secondly, some results for the Neural model using a binary response variable are presented. This chapter concludes with presenting the results of the cross-application of these results with the already developed propensity to buy model described in chapter 2.

5.1 Comparison of the GLM and Neural Model

The Generalized Linear Model and the Artificial Neural Network use the following parameters

Variable	GLM	ANN
ClaimCnt	×	×
FirstProd	×	×
SalesChannel	×	×
YearOfBirthGroup	×	×
MaritalStatus	×	×
EducationLevel	×	×
HHIncomeDecile	×	×
Gender		×
MonthOfBirth		×
RegionOfResidency		×
HousingType		×
LifePhase		×
LastRelocationDate		×

The list of candidate independent variables studied did not motivate a larger set of variables included in the GLM. Given the fact that the ANN works on a larger set of input variables and has a larger degree of freedom it should be somewhat better. However, some caution is necessary when following this logic. A known risk when using neural networks, and regression models for that matter, is the risk of over-fitting. This is when noise and random behavior in the training data is fitted rather than the overall pattern. When more variables and hidden layers are added to a model, the overall fit of the data tends to increase and the R^2 increases. It is important to note that the model is built on one set of data, and that testing and systemic usage of the models is made on another. If the model is too specific, it may be overemphasizing certain traits in the training set and can have lower predictiveness than a more generalizing model built on fewer variables.

In this section, the goal is to compare and cross-validate the GLM and Neural Network model against each other. To do this, the customers are assigned into deciles according to the predicted claim frequency of each model respectively. The first decile is defined as 10 % of the customers with highest claim frequency, second decile is another tenth of the customers with claim frequencies lower than the first decile but higher than the subsequent deciles and so on.

Comparing the predicted deciles using the ANN with the actual, realized claim frequency deciles of customers is made in Figure 8.

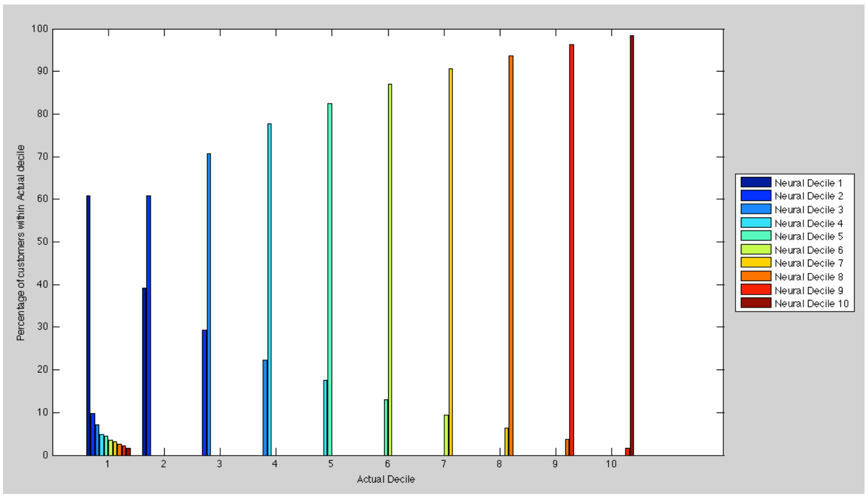


Figure 8: Comparison of actual decile assignment with predicted assigments using ANN

The results indicate that actual first decile claim frequency customers where largely also predicted to be in that category. Among the customers in that segment 60 % were also predicted to be in that category. The remaining 40 % were predicted to be in other segments. For subsequent deciles the model is able to predict largely the same category. Most of the discrepancy between the realized and the predicted lies in the first segment.

The same comparison with actual claim frequency and the predicted assignment made using the GLM is shown in Figure 9. The GLM is less accurate in predicting the decile compared with the neural network. It shows a larger tendency of over and under predicting claim frequency. It does however correctly shift the 'center of mass' in the assigned deciles as one moves from left to right in the chart.

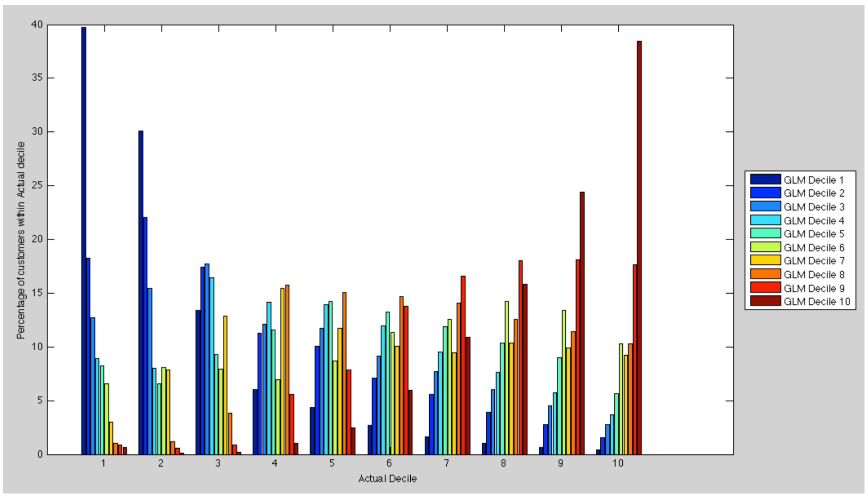


Figure 9: Comparison of actual decile assignments with predicted assigments using a GLM

For validation, one can evaluate the *risk ratio* by predicted values. Risk ratio, is a common metric in insurance reporting and is defined as cost of claims over premiums paid of a given time interval. Comparing the Risk Ratio by the predicted deciles in Figure 10 shows that the models finds the most claim intense customers with some success, but is less precise at differentiating between customers in the lower claim frequency ranges.

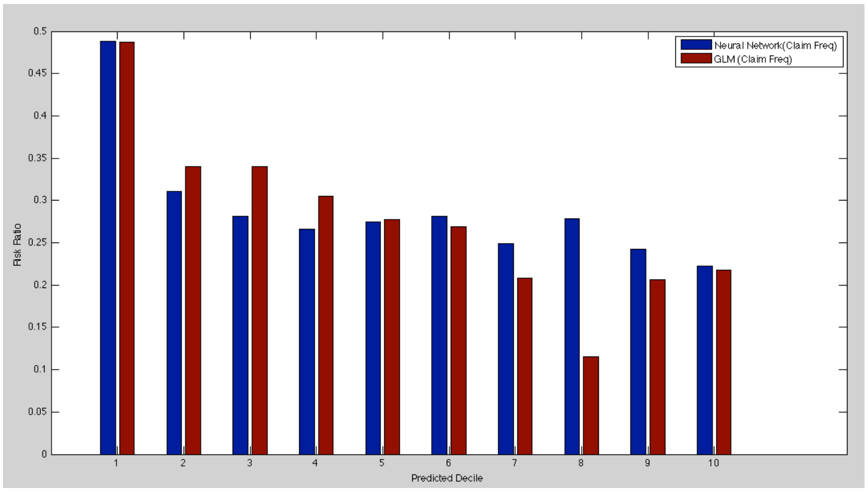


Figure 10: Risk ratio by decile

5.2 Results for the Artificial Neural Network: Response variable Binary Claim

To validate the ANN architecture and setup, a study on the binary target variable claim or no claim was made. The Receiving Operator Characteristics (ROC) chart below illustrates how well the customers are assigned into one of the two classes. The plot shows true positive rate against the false positive rate, or in other words the model's overall ability to correctly flag future claimers against a false prediction of an actual non-claiming customer as a customer with claims. The network chosen was the one closes to the upper left corner.

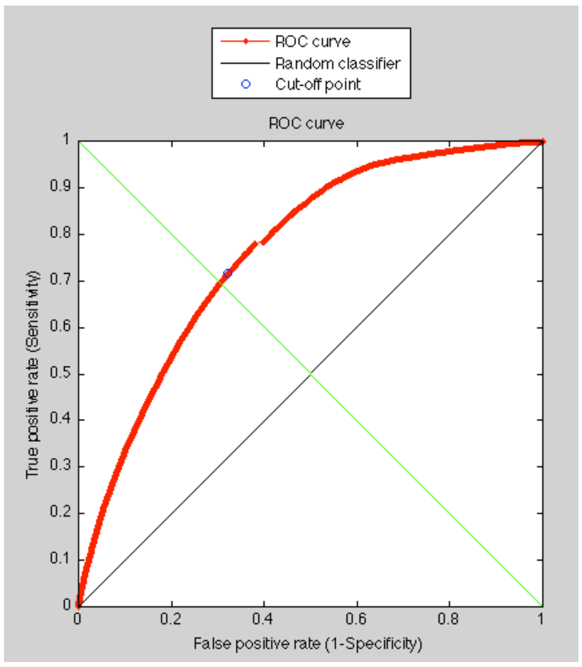


Figure 11: ROC chart

A similar plot, the gain curve shows how well the customer based in sorted into descending risk. The neural network assigns 50 % of the all claimers in the top two deciles.

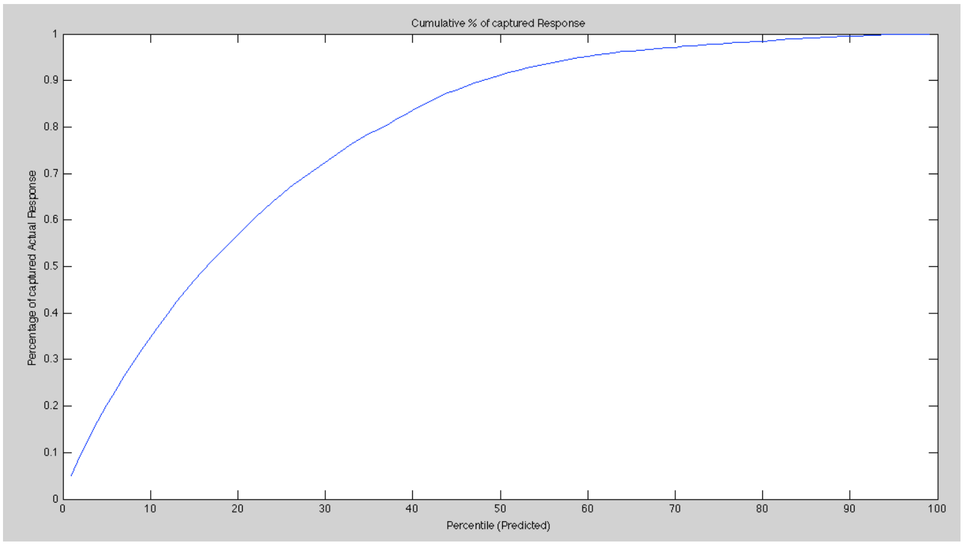
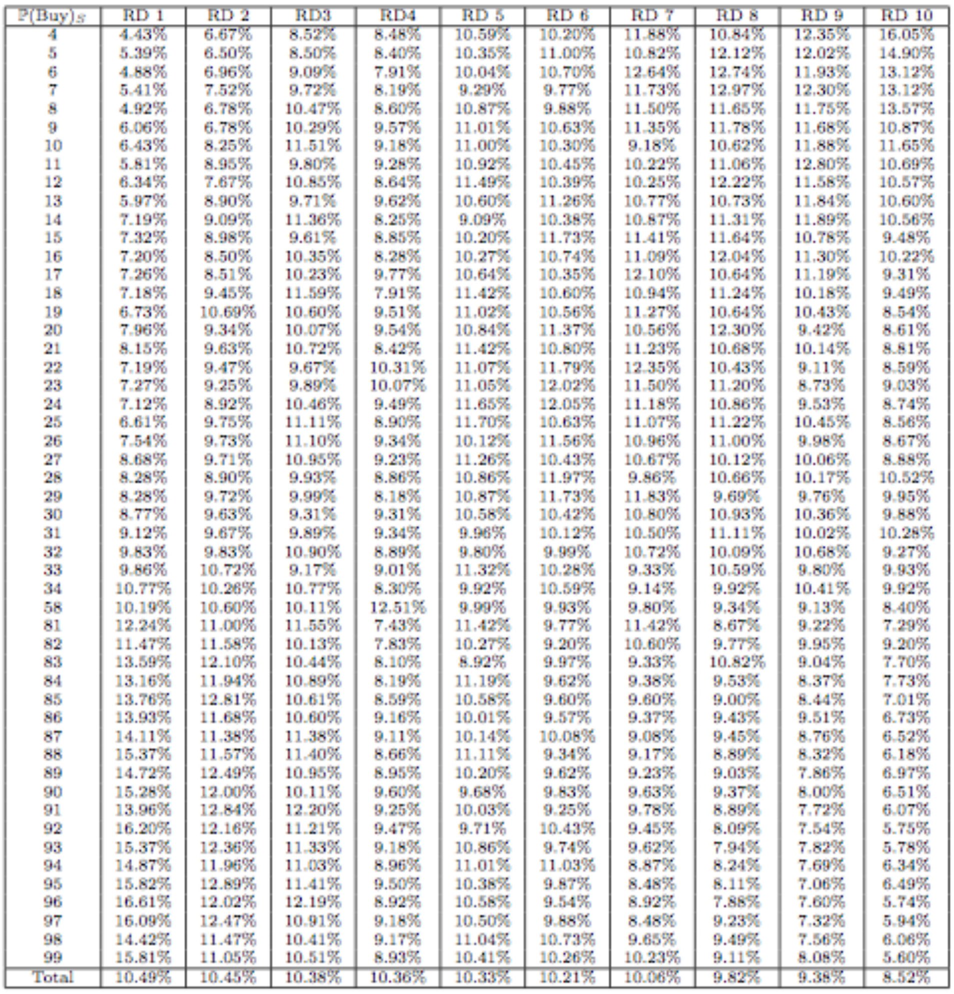


Figure 12: Gains Curve

5.3 Propensity to Buy and Risk Metric

A final objective of this thesis was to combine at the customer level, a predicted risk propensity metric with an estimated probability of purchase. The results are presented in Table 13, where the first column represents a propensity to buy score, the subsequent columns represent a measure of risk in descending order. The values represent percentage of row. The interesting results come from the fact that higher probability of purchase shows higher risk-prone characteristics and the converse, low probability of purchase corresponds to lower predicted risk.



$P(Buy)_S$	RD 1	RD 2	RD3	RD4	RD 5	RD 6	RD 7	RD 8	RD 9	RD 10
4	4.43%	6.67%	8.52%	8.48%	10.59%	10.20%	11.88%	10.84%	12.35%	16.05%
5	5.39%	6.50%	8.50%	8.40%	10.35%	11.00%	10.82%	12.12%	12.02%	14.90%
6	4.88%	6.96%	9.09%	7.91%	10.04%	10.70%	12.64%	12.74%	11.93%	13.12%
7	5.41%	7.52%	9.72%	8.19%	9.29%	9.77%	11.73%	12.97%	12.30%	13.12%
8	4.92%	6.78%	10.47%	8.60%	10.87%	9.88%	11.50%	11.65%	11.75%	13.57%
9	6.06%	6.78%	10.29%	9.57%	11.01%	10.63%	11.35%	11.78%	11.68%	10.87%
10	6.43%	8.25%	11.51%	9.18%	11.00%	10.30%	9.18%	10.62%	11.88%	11.65%
11	5.81%	8.95%	9.80%	9.28%	10.92%	10.45%	10.22%	11.06%	12.80%	10.69%
12	6.34%	7.67%	10.85%	8.64%	11.49%	10.39%	10.25%	12.22%	11.58%	10.57%
13	5.97%	8.90%	9.71%	9.62%	10.60%	11.26%	10.77%	10.73%	11.84%	10.60%
14	7.19%	9.09%	11.36%	8.25%	9.09%	10.38%	10.87%	11.31%	11.89%	10.56%
15	7.32%	8.98%	9.61%	8.85%	10.20%	11.73%	11.41%	11.64%	10.78%	9.48%
16	7.20%	8.50%	10.35%	8.28%	10.27%	10.74%	11.09%	12.04%	11.30%	10.22%
17	7.26%	8.51%	10.23%	9.77%	10.64%	10.35%	12.10%	10.64%	11.19%	9.31%
18	7.18%	9.45%	11.59%	7.91%	11.42%	10.60%	10.94%	11.24%	10.18%	9.49%
19	6.73%	10.69%	10.60%	9.51%	11.02%	10.56%	11.27%	10.64%	10.43%	8.54%
20	7.96%	9.34%	10.07%	9.54%	10.84%	11.37%	10.56%	12.30%	9.42%	8.61%
21	8.15%	9.63%	10.72%	8.42%	11.42%	10.80%	11.23%	10.68%	10.14%	8.81%
22	7.19%	9.47%	9.67%	10.31%	11.07%	11.79%	12.35%	10.43%	9.11%	8.59%
23	7.27%	9.25%	9.89%	10.07%	11.05%	12.02%	11.50%	11.20%	8.73%	9.03%
24	7.12%	8.92%	10.46%	9.49%	11.65%	12.05%	11.18%	10.86%	9.53%	8.74%
25	6.61%	9.75%	11.11%	8.90%	11.70%	10.63%	11.07%	11.22%	10.45%	8.56%
26	7.54%	9.73%	11.10%	9.34%	10.12%	11.56%	10.96%	11.00%	9.98%	8.67%
27	8.68%	9.71%	10.95%	9.23%	11.26%	10.43%	10.67%	10.12%	10.06%	8.88%
28	8.28%	8.90%	9.93%	8.86%	10.86%	11.97%	9.86%	10.66%	10.17%	10.52%
29	8.28%	9.72%	9.99%	8.18%	10.87%	11.73%	11.83%	9.69%	9.76%	9.95%
30	8.77%	9.63%	9.31%	9.31%	10.58%	10.42%	10.80%	10.93%	10.36%	9.88%
31	9.12%	9.67%	9.89%	9.34%	9.96%	10.12%	10.50%	11.11%	10.02%	10.28%
32	9.83%	9.83%	10.90%	8.89%	9.80%	9.99%	10.72%	10.09%	10.68%	9.27%
33	9.86%	10.72%	9.17%	9.01%	11.32%	10.28%	9.33%	10.59%	9.80%	9.93%
34	10.77%	10.26%	10.77%	8.30%	9.92%	10.59%	9.14%	9.92%	10.41%	9.92%
58	10.19%	10.60%	10.11%	12.51%	9.99%	9.93%	9.80%	9.34%	9.13%	8.40%
81	12.24%	11.00%	11.55%	7.43%	11.42%	9.77%	11.42%	8.67%	9.22%	7.29%
82	11.47%	11.58%	10.13%	7.83%	10.27%	9.20%	10.60%	9.77%	9.95%	9.20%
83	13.59%	12.10%	10.44%	8.10%	8.92%	9.97%	9.33%	10.82%	9.04%	7.70%
84	13.16%	11.94%	10.89%	8.19%	11.19%	9.62%	9.38%	9.53%	8.37%	7.73%
85	13.76%	12.81%	10.61%	8.59%	10.58%	9.60%	9.60%	9.00%	8.44%	7.01%
86	13.93%	11.68%	10.60%	9.16%	10.01%	9.57%	9.37%	9.43%	9.51%	6.73%
87	14.11%	11.38%	11.38%	9.11%	10.14%	10.08%	9.08%	9.45%	8.76%	6.52%
88	15.37%	11.57%	11.40%	8.66%	11.11%	9.34%	9.17%	8.89%	8.32%	6.18%
89	14.72%	12.49%	10.95%	8.95%	10.20%	9.62%	9.23%	9.03%	7.86%	6.97%
90	15.28%	12.00%	10.11%	9.60%	9.68%	9.83%	9.63%	9.37%	8.00%	6.51%
91	13.96%	12.84%	12.20%	9.25%	10.03%	9.25%	9.78%	8.89%	7.72%	6.07%
92	16.20%	12.16%	11.21%	9.47%	9.71%	10.43%	9.45%	8.09%	7.54%	5.75%
93	15.37%	12.36%	11.33%	9.18%	10.86%	9.74%	9.62%	7.94%	7.82%	5.78%
94	14.87%	11.96%	11.03%	8.96%	11.01%	11.03%	8.87%	8.24%	7.69%	6.34%
95	15.82%	12.89%	11.41%	9.50%	10.38%	9.87%	8.48%	8.11%	7.06%	6.49%
96	16.61%	12.02%	12.19%	8.92%	10.58%	9.54%	8.92%	7.88%	7.60%	5.74%
97	16.09%	12.47%	10.91%	9.18%	10.50%	9.88%	8.48%	9.23%	7.32%	5.94%
98	14.42%	11.47%	10.41%	9.17%	11.04%	10.73%	9.65%	9.49%	7.56%	6.06%
99	15.81%	11.05%	10.51%	8.93%	10.41%	10.26%	10.23%	9.11%	8.08%	5.60%
Total	10.49%	10.45%	10.38%	10.36%	10.33%	10.21%	10.06%	9.82%	9.38%	8.52%

Figure 13: Propensity to Buy and Estimated future Risk

6 Conclusion

Different models used to predict future profitability early into a customer life cycles were developed. Several modeling techniques and response variables were considered and tested. Claim frequency was retained as the most viable option as a response variable. The basic theory underlying the two resulting models using Generalized Linear Models and a feedforward multilayer perceptron was presented. The model performances were evaluated and compared by estimating the future claim frequency *ex post* and comparing the predictions with actual data. The results show that the implemented models are able to differentiate higher propensity of risk, but less effective across the entire frequency range.

The estimated future risk was appended to existing models on propensity to buy future insurance policies. Interestingly enough, the combined distribution suggests that the customers most likely to purchase also tend to be less profitable. Retaining profitable customers and prioritization within the existing set of customers should therefor be seen as an alternative levy against adverse selection.

The combined predictions on propensity to purchase and expected profitability can be used in several ways. One solution is to restrict the number of prospects with low expectations on profitability actively treated by sales agent or used in any campaign activity. In practice this could work as a filter, where the expected profitability needs to be larger than a certain threshold, or that a given percentage e.g. 5% customers with the lowest predicted profitability are continuously filtered out. Another approach is to assign a combined score, akin to a cost of being in a given point in a matrix shown in Table 13. An example of this could be a matrix of scores as shown below.

	RD 1	RD 2	RD3	RD4	RD 5	RD 6	RD 7	RD 8	RD 9	RD 10
PD 1	100	90	80	70	60	50	40	30	20	10
PD 2	100	81	72	63	54	45	36	27	18	9
PD 3	100	72	64	56	48	40	32	24	16	8
PD 4	100	63	56	49	42	35	28	21	14	7
PD 5	100	54	48	42	36	30	24	18	12	6
PD 6	100	45	40	35	30	25	20	15	10	5
PD 7	100	36	32	28	24	20	16	12	8	4
PD 8	100	27	24	21	18	15	12	9	6	3
PD 9	100	18	16	14	12	10	8	6	4	2
PD 10	100	9	8	7	6	5	4	3	2	1

Another future direction to be taken from adding a measure of risk in CRM prospecting activity is the compensations to underwriters and sellers: Selling a policy is good, selling a more profitable one should be considered better. Expanding the goals and incentives of volume increase and retention seeking activity with profitability measurements could be a way of cultivating long-term profitability.

7 Bibliography

References

[1] Thomas Mikosch. ”Non-Life Insurance Mathematics”. 2nd Edition Springer-Verlag Berlin Heidelberg 2009.

[2] Pavel Cizek, Wolfgang Karl Härdle, Rafał Weron Statistical. ”Statistical Tools for Finance and Insurance”. 2nd Edition Springer-Verlag Berlin Heidelberg 2005.

[3] Cira Perna, Marilena Sibillo. ”Mathematical and Statistical Methods in Insurance and Finance”. Springer-Verlag Italia Milano 2008.

[4] Robert L Grossman. ”Data mining for scientific and engineering applications”. Kluwer Academic Publishers, 2001.

[5] Ulf Olsson. ”Generalized Linear Models”. Studentliteratur, Lund 2002.

[6] Simon Haykin. ”Neural Networks, A Comprehensive Foundation”. 2nd Edition Pearson Eduction 1999.

[7] Pierre Peretto. ”An Introduction to the Modeling of Neural Networks”. Cambridge Univeristy Press, 1992.

[8] Patricia B Cerrito. ”Introduction to Data Mining: Using SAS Enterprise Miner”. SAS Publishing 2007.

[9] Stephen G. Nash and Ariela Sofer. ”Linear and Nonlinear Programming”. McGraw-Hill 1996.

[10] P. McCullagh and J. A. Nelder. ”Generalized Linear Models”.2nd Edition Chapman & Hall/CRC 1990.