Credal Classification of Automobile Risk

Ben Willis

February 13, 2017

Abstract

This project investigates how a credal classifier can be applied to the problem of determining auto mobile insurance risk. Currently auto mobiles are assigned a risk rating by an expert who takes in to account various attributes of a vehicle. We build a classifier capable of emulating the experts assessments.

Contents

Chapter 1

Introduction

1.1 Classification

Classification is the problem of identifying which class an object belongs to. Each object can be distinguished by a set of properties know as features and each object belongs to a single class. A classifier is an algorithm which, given previous observations and their classes, can determine which class a new observation belongs to [5]. There are many applications of classifiers including image recognition, sentiment analysis and medical diagnosis.

There are two types of classifiers, supervised and unsupervised. Unsupervised classifiers infer classes from the data. Supervised classifiers are constructed from a set of data for which the true classes are known and this is the type of classifier we will be exploring [1].

1.2 Auto mobile Insurance

Classifiers have many applications in the finance industry ranging from financial trading [2] to credit card fraud detection [4]. We will study the problem of classifying the risk to an insurer of a car and comparing this solution to the classification of an expert. We will then examine how both classifications compare to the normalised loss to the insurer.

The data set we will be analysing contains vehicular information from 205 auto mobiles. Its features include dimensions, engine specifications and vehicle characteristics. It also contains an expert's assessed risk to the insurer of the vehicle on an integer scale of -2 to 3 with 3 being most risky and -2 being least risky. In addition to the technical information and the experts assessment, the data set also contains the normalized loss to the insurer. This ranges from 65 to 256 and is normalized for all vehicles within a particular size classification (two-door small, station wagons, etc.) and represents the

average loss per car per year [6].

Chapter 2

Naive Bayes Classifier

2.1 Theory

To help explain how the naive Bayes classifier works we will introduce a new data set from a remote sensing study. The study measured spectral information in the green, red and infrared wavelengths on three separate dates of 523 different areas of forest in Japan. In total each observation has nine continuous attributes and falls in to one of four possible classes: Sugi forest, Hinoki forest, Mixed deciduous forest and other non-forest land. This data set was chosen as it contained a large number of observations and demonstrates a situation where this type of classifier works well.

Formally, let us denote the class variable by C, taking values from the set $\{0, 1, 2, 3, 4\}$ for each of the four types of forest. We measure 9 features $A_1, \ldots, A_9$ which we discretize so that they all take values from the set $\{0, \ldots, 9\}$. We denote observations of the observed values as c and $a_1, \ldots, a_9$ respectively.

We are interested in the probability of a forest being of type c given sensor readings $\mathbf{a}$ i.e. $P(c \mid \mathbf{a})$. Using Bayes theorem we can rewrite this as:

$$P(c \mid \mathbf{a}) = \frac{P(c)P(\mathbf{a} \mid c)}{P(\mathbf{a})} \tag{2.1}$$

Moreover we can make use of the naivety assumption. The naivety assumptions states that each attribute is conditionally independent given the class. In the context of this data set we are assuming the sensor readings are conditionally independent given the type of forest. We can therefore write the probability of observing an object with attributes $a_1, \ldots\ a_9$ given it is in class c as:

$$P(\mathbf{a} \mid c) = \prod_{i=1}^{9} P(a_i \mid c) \tag{2.2}$$

Note that $P(\mathbf{a})$ is independent of c and is just a scaling constant. So by bringing together

eqs. (2.1) and (2.2) we can write:

$$P(c \mid \mathbf{a}) \propto P(c) \prod_{i=1}^{k} P(a_i \mid c) \tag{2.3}$$

To turn this into a classifier we need a way to make a decision for which class an object falls into based on the estimated probabilities. We can introduce a *loss function* to allow us to choose the class. As standard choice is the 0-1 loss function we defined as:

$$L(c, \hat{c}) = \begin{cases} 0 & \text{if } c = \hat{c} \\ 1 & \text{otherwise} \end{cases} \tag{2.4}$$

We can write our expected loss as:

$$E(L) = \sum L(c, \hat{c}) P(c \mid \mathbf{a}) = 1 - P(\hat{c} \mid \mathbf{a}) \tag{2.5}$$

So to minimize our expected loss we choose:

$$\hat{c} = \arg\max_{c \in \mathcal{C}} P(c) \prod_{i=1}^{k} P(a_i \mid c) \tag{2.6}$$

This is known as the maximum a posteriori (MAP) estimate.

Now that we have our method for making our decision we need to estimate the required probabilities.

Firstly we parametrise these probabilities. We denote the unknown chances of observing an object in class c by θ_c and the chance of observing an object in class c with attributes $\mathbf{a}$ by $\theta_{\mathbf{a},c}$. Similarly we denote the conditional chances of observing an individual attribute a_i and a set of attributes $\mathbf{a}$ given $C = c$ by $\theta_{a_i|c}$ and $\theta_{\mathbf{a}|c}$ respectively.

Now we have parametrised the probabilities, we wish to estimate we can consider the likelihood function for θ, the vector whose elements are the chances θ_c and $\theta_{a_i|c}$. Using our data we denote the frequencies of objects in each class c by $n(c)$ and the number of objects in class c with attribute a_i by $n(a_i, c)$. For example the number of observations of class 0 is 158 so $n(0) = 158$ We then consider the vector $\mathbf{n}$ which contains these frequencies.

The likelihood function can be expressed as:

$$l(\theta \mid \mathbf{n}) \propto \prod_{c \in \mathcal{C}} \left[\theta_c^{n(c)} \prod_{i=1}^{k} \prod_{a_i \in \mathcal{A}_i} \theta_{a_i|c}^{n(a_i,c)} \right] \tag{2.7}$$

A simple estimate for these parameters is the maximum likelihood estimate (MLE). To find the MLE first we take the log likelihood:

$$L(\theta \mid \mathbf{n}) \propto \sum_{c \in \mathcal{C}} n(c) log(\theta_c) + \sum_{c \in \mathcal{C}} \sum_{i=1}^{k} \sum_{a_i \in \mathcal{A}_i} n(a_i, c) log(\theta_{a_i|c}) \tag{2.8}$$

So to maximise the likelihood function we need to maximise the each part of the log likelihood function.

To do so we use the method of Lagrange multipliers. This is a strategy for finding local maxima and minima of a function subject to constraints.

For the θ_c parameters we want to maximise:

$$f(\theta, \mathbf{n}) = \sum_{c \in \mathcal{C}} n(c) log(\theta_c) \tag{2.9}$$

under the constraint:

$$g(\theta, \mathbf{n}) = \sum_{c \in \mathcal{C}} \theta_c - 1 = 0 \tag{2.10}$$

This gives us our Lagrangian:

$$\mathcal{L}(\theta, \mathbf{n}, \lambda) = \sum_{c \in \mathcal{C}} n(c) log(\theta_c) - \lambda(\sum_{c \in \mathcal{C}} \theta_c - 1) \tag{2.11}$$

Differentiating with respect to θ_c we have:

$$\nabla_{\theta_c} \mathcal{L}(\theta, \mathbf{n}, \lambda) = \frac{n(c)}{\theta_c} - \lambda \tag{2.12}$$

Setting this to zero and using 2.10 we have maximum likelihood estimate:

$$\hat{\theta}_c = \frac{n(c)}{N} \tag{2.13}$$

This is just the relative frequency of observations that fall into that class. Returning to our example data set we have $N = 523$ and $n(0) = 158$ so $\hat{\theta}_0 = \frac{158}{523} \approx 0.302$

Similarly for the $\theta_{a_i|c}$ parameters we want to maximize:

$$f(\theta, \mathbf{n}) = \sum_{c_i \in \mathcal{A}_i} n(a_i, c) log(\theta_{a_i|c}) \tag{2.14}$$

for each $c \in \mathcal{C}$ and $i \in \{1, \ldots, k\}$ under the constraint:

$$g(\theta, \mathbf{n}) = \sum_{a_i \in \mathcal{A}_i} \theta_{a_i|c} - 1 = 0 \tag{2.15}$$

We again solve this using the Lagrangian to find the maximum likelihood estimate:

$$\hat{\theta}_{a_i|c} = \frac{n(a_i, c)}{n(c)} \tag{2.16}$$

We now have our naive Bayes classifier. We estimate:

$$P(c) \text{ by } \frac{n(c)}{N} \tag{2.17}$$

$$P(a_i \mid c) \text{ by } \frac{n(a_i, c)}{n(c)} \tag{2.18}$$

Then we choose the class c which maximises eq. (2.6).

2.2 Diagnostics

To measure how successful our classifier is we will initially use a technique known as k-fold cross validation. In k-fold cross validation we split our dataset into k equally sized groups. Then for each of these group we train the classifier on all the other groups then test it on the excluded group. We then average all these measurements to return an estimate for the measurement of our classifier.

The choice of k leads to different types of cross validation. A special case of cross validation is when $k = n$ (the number of observations). This is knowns as *leave-one-out cross validation* [3].

Initially we will consider two metrics. The first is accuracy, this is the percentage of correctly classified objects. The seconds is failed classifications, this is percentage of objects with no MAP estimate for their class and usually occurs when $P(c \mid \mathbf{a}) = 0$ for all $c \in \mathcal{C}$.

To properly estimate these metrics we carry out the cross validation repeatedly until the standard error of the mean is less than 0.2%. The accuracy of the classifier is 81.77% and the percentage of unclassified objects is 2.04% on the forest data set, using 5-fold cross validation.

2.3 Application to Auto Mobile Data set

To make our auto mobile data appropriate for this method we discretize the continuous variables into 10 bins with an equal frequency.

Unlike in the trees data set, in this data set we have objects with missing values for attributes. We have no mechanism for considering these so we must discard these observations for now. This reduces our data set from 205 observations to 193 observations.

For a single iteration of cross-validation these are the class probabilities estimates:

c	-2	-1	0	1	2	3
$P(c)$	0.017	0.121	0.324	0.249	0.173	0.116

We can see that almost a third of the observations in our data set fall in to 0 class.

As before we repeatedly carry out 5-fold cross validation until we have a standard error of less that 0.2%. The accuracy of our classifier is 59.95% on the auto mobile data set. This is considerably worse than the example forest data set, moreover 22.45% of the objects were not classified.

2.4 Conclusions

Clearly the classifier performs better on the forest type dataset than on the auto mobile data set. There are also general failings in our classifier we can fix to improve its accuracy for both data sets.

Firstly our classifier falls down if there are no observations with attribute a_j and class c in our training set. In these case the maximum likelihood estimate for $\theta_{a_j|c}$ is 0. This estimate leads to $P(c \mid \mathbf{a}) = 0$ and would rule out assigning any objects with the attribute a_j to class c. In fact there are 50 objects in our data set for which our classifier returns $P(c \mid \mathbf{a}) = 0$ for all $c \in C$. This is especially problematic for small sets of data and contributes to the large percentage of failed classifications we was. We can tackle this by introducing prior probabilities for the theta chances.

One thing that the auto mobile data set has that the forest type data set doesn't is missing values. By discarding the objects with missing values we throw away 20 observations from an already small data set. If we could make use of these incomplete observation we should be able to improve our classifier.

We can also make use of the structure in the auto mobile data set's ordered classes. The accuracy metric does not take into account how close the classification is. For example if the true class is 2 an assigned class of 1 should be considered better than an assigned class of -2. The 0-1 loss function also does not take this into account and a better choice of loss function may prove beneficial.

Chapter 3

Corrected NBC with Dirichlet Prior

3.1 Theory

We return to our likelihood function eq. (2.7) for our theta variables. We can introduce a prior distribution for these parameters and then consider the posterior distribution.

The Dirichlet distribution is the multinomial extension of the beta distribution for $\theta_1, \ldots, \theta_k$ where $\theta_i \in (0,1)$ and $\sum_{i=1}^{k} \theta_i = 1$ with probability density function:

$$f(\theta_1, \ldots, \theta_k \mid s, t(1), \ldots, t(k)) \propto \prod_{i=1}^{k} \theta_i^{st(i)-1} \tag{3.1}$$

where $s > 0$ and each $t(i) > 0$ such that $\sum_{i=1}^{k} t(i) = 1$.

We introduce a distribution that is similar to our likelihood as our prior density:

$$f(\theta \mid \mathbf{t}, s) \propto \prod_{c \in \mathcal{C}} \left[\theta_c^{st(c)-1} \prod_{i=1}^{k} \prod_{a_i \in \mathcal{A}_i} \theta_{a_i|c}^{st(c,a_i)-1} \right] \tag{3.2}$$

This is in the same form as the likelihood however each $n(\cdot)$ is replaces by $st(\cdot) - 1$. $s > 0$ is a fixed constant and we have the following constraints on $t(\cdot)$:

$$\sum_{c \in \mathcal{C}} t(c) = 1 \tag{3.3}$$

$$\sum_{a_i \in \mathcal{A}_i} t(a_i, c) = t(c) \qquad \forall i, c \tag{3.4}$$

$$t(a_i, c) > 0 \qquad \forall i, a_i, c \tag{3.5}$$

This distribution is the conjugate prior for the likelihood function eq. (2.7). When we multiply our likelihood by this prior density we get a posterior in the same family as this prior. If the prior has hyper parameters $st(\cdot)$ the posterior will have hyper parameters $\dot{st()} + n(\cdot)$.

We can now estimate the parameters by taking the MAP estimates i.e.

$$\hat{\theta_c} = \frac{n(c) + st(c)}{N + s} \tag{3.6}$$

We estimate:

$$P(c) \text{ by } \frac{n(c) + st(c)}{N + s} \tag{3.7}$$

$$P(a_i \mid c) \text{ by } \frac{n(a_i, c) + st(a_i, c)}{n(c) + st(c)} \tag{3.8}$$

3.2 Application

To apply this classifier to our data set we need to choose the hyper parameters of the prior distribution. We note that s affects the speed at which our classifier learns and $t(\cdot)$ represents our beliefs for θ.. To comply with the constraints 3.3 let us set:

$$s = 1 \tag{3.9}$$

$$t(c) = \frac{1}{|C|} \tag{3.10}$$

$$t(a_i, c) = \frac{1}{|A_i||C|} \tag{3.11}$$

Previously we've considered the accuracy of our classifier. An alternative metric is the mean squared error of our estimate.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{c_i} = c_i)^2 \tag{3.12}$$

	Accuracy	MSE	Failed Classifications
NBC	59.95%	0.689	22.45%
Corrected NBC	68.17%	2.823	0%

This again shows an improvement over the previous classifier.

3.3 Conclusions

Bibliography

[1] D. J. Spiegelhalter D. Michie and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* 1994.

[2] Eduardo A. Gerlein, Martin McGinnity, Ammar Belatreche, and Sonya Coleman. Evaluating machine learning classification for financial trading. *Expert Syst. Appl.*, 54(C):193–207, July 2016.

[3] Paul E. Keller Kevin L. Priddy. *Artificial Neural Networks: An Introduction.* SPIE Press, 2005.

[4] Andrea Dal Pozzolo. *Adaptive Machine Learning for Credit Card Fraud Detection.* PhD thesis, Universit Libre de Bruxelles, 2015.

[5] Konstantinos Koutroumbas S. Theodoridis. *Pattern Recognition.* Elsevier Science, 2003.

[6] J. Schlimmer. *Automobile Data Set*, 1987 (accessed November 8, 2016). `https://archive.ics.uci.edu/ml/datasets/Automobile`.