Picturing Bayesians Classifiers: A Visual Data Mining Approach to Parameters Optimization

Giorgio Maria Di Nunzio

Department of Information Engineering — University of Padua

Alessandro Sordoni

Déptartement d'Informatique et Recherche Opérationnelle — Université de Montréal

Abstract

The goal of supervised classification is to assign a new object to a class from a given set of classes based on the attribute values of this object and on a training set. Although "supervised", classification algorithms provide only very limited forms of guidance by the user. Typically, the user selects the dataset and sets the values for some parameters of the algorithm - which are often difficult to determine a priori. We believe the user should be involved more interactively in the process of classification because, by providing adequate data and knowledge visualizations, the pattern recognition capabilities of the human can be used to increase the effectivity of classifier construction. Moreover, users often want to validate and explore the classifier model and its output. To address these issues, the classification system should have an intuitive and interactive explanation capability. We present a two-dimensional visualization tool for Bayesian classifiers that can help the user understand why a classifier makes the predictions it does given the vector of parameters in input. The user can interact with the classifier by: selecting different models, changing the distribution of the prior, tuning the mis-classification costs. To help people discover (sub)optimal parameters, we develop several visual interaction methods that allow people to interactively analyze objects. Finally, we present a case study to demonstrate the effectiveness of our solution in text classification.

Key words: Visual Analytics, Bayesian Inference, Smoothing, Text Classification

Email addresses: dinunzio@dei.unipd.it (Giorgio Maria Di Nunzio),

1 Introduction

In machine learning, "computers are programmed to optimize a performance criterion using example data or past experience" [Alpaydin, 2009]. It is reasonable to assume that there is a hidden process that explains the data we observe. Though we do not know the details of this process, we know that it is not completely random. This enacts the possibility of finding a good and useful approximation despite we may not be able to identify the process completely. Mathematical models defined upon parameters can be used for this task. The "learning" part of the model consists in choosing the parameters which optimize a performance criterion with respect to observed data.

"Application of machine learning methods to large databases is called data mining" [Alpaydin, 2009]. Data mining applications can retrieve and explore existing information as well as extrapolate, predict, and derive new information from the given database. Classification is a special kind of the prediction task which deals with the need of classifying items based on previously classified training data. The research in this field has been very active in the last years [1] . For example, mining billions of user ratings for musical pieces to discover user profiles and predict which songs users will listen to, or mining tweets contents to capture users interests to serve them with potentially interesting items thus reducing information overload. In literature, many successful algorithms for pattern classification, inference and prediction has been presented [Hastie et al., 2009]. Some of these techniques require that the user selects the dataset and performs some tuning on the algorithm's parameters - which are often difficult to determine a priori. Moreover, some of these act as black boxes screening the user out of the analysis process. In this context, interpreting learned parameters and discovering the causal process underlying observed data become difficult tasks. Data mining applications may benefit significantly by providing visual feedback and summarization. This is the goal of Visual Data Mining.

Visual data mining is a general approach which aims to include the human in the data exploration process, thus gaining benefit from his perceptual abilities. In particular, users often want to validate and explore the classifier model and its output or understand the classification rationale. To address these issues, the classification system should have an intuitive and interactive explanation capability [Poulin et al., 2006,Wong, 1999,de Oliveira and Levkowitz, 2003]. Visual data mining techniques have proven to be of high value in exploratory data analysis and they also have a high potential for exploring large databases [Hansen

sordonia@iro.umontreal.ca (Alessandro Sordoni).

[1] http://www.sigkdd.org/

and Johnson, 2004, Part XI, pp. 819-830]. Two or three-dimensional representation is probably the most 'natural' metaphor a visualization system can offer to model object relationships. This is how we perceive world as humans: two objects that are 'close' each other are probably more similar than two objects far away. The interactive visualization and navigation of such space becomes a means to browse and explore the corpus which match predetermined characteristics [Chalmers and Chitson, 1992, Wise et al., 1995, Wise et al., 1995, Becker, 1997, Rohrer et al., 1999, Becks et al., 2000, Kohonen, 1995, Ankerst et al., 2000, Harrell, 2006, Mozina et al., 2004, Leban et al., 2006, Poulin et al., 2006, Poulet, 2008, Seifert and Lex, 2009].

In this chapter we focus our attention on the Bayesian classifier. Although based upon strong conditional dependence assumptions, this classifier is still widely used in practice mostly likely due to its tradeoff between very efficient model training and good empirical results. These characteristics make this type of classifier very suitable for analyzing large-scale datasets and synthesizing big amounts heterogeneous data quickly. The chapter is organized in two main parts: we present the Bayesian framework which characterizes the nature of the classification problem by introducing bayesian data analysis; then we describe a visualization tool to support the classification process.

The chapters is divided into the following sections: we discuss related works in Section 2; in Section 3, we give the motivations of our work and list the requirements of the R code. Section 4 presents the Bayesian probabilistic framework we use to describe the Naïve Bayes classifiers. The two-dimensional visualization system is described in Section 5.

2 Related Works

Some works in literature have specifically tackled the problem of the visualization of Naïve Bayes (NB) classifiers and in this respect are the ones that can be directly compared to our visualization tools. The Evidence Visualizer [Becker et al., 2002] can display Bayes model decisions as pies and bar charts. In particular, the rows of pie charts represent each attribute, and each pie chart represents an interval or value of the attribute. The attributes are listed in order of usefulness for predicting the label. The height of the pies shows the number record having a particular value. In [Mozina et al., 2004], authors show how to adapt Naïve Bayesian classifiers and present them with a visualization technique called nomograms [Harrell, 2006]. On of the main benefits of this approach is the simple and clear visualization of the complete model and the quantitative information it contains. The

visualization can be used both for exploratory analysis and for decision making. It can also be used to compare different models, including those coming from logistic regression. ExplainD [Poulin et al., 2006] is a framework for explaining decisions made by classifiers that use additive evidence. It has been applied to different linear model such as support vector machines, logistic regression and Naïve Bayes. The main goal of this framework is to visually explaining the decisions of machine-learned classifiers and the evidence for those decisions. There are five explanation capabilities: Decision, Decision Evidence, Decision Speculation, Ranks of Evidence, Source of Evidence. Each successive capability increases the user's ability to understand and audit an aspect of the classification process, based on the evidence. The Class Radial Visualization [Seifert and Lex, 2009] is an integrated visualization system that provides interactive mechanisms for a deep analysis of classification results and procedures. In this system, class items are displayed as squares and equally distributed around the perimeter of a circle. Objects to be classified are displayed as colored points in the circle and the distance between the point and the squares represent the uncertainty of assigning that object to the class.

It is worth mentioning another approach that does not specially address the NB classifier but resembles our idea of visualizing the line of the decision of the classifier. In [Poulet, 2008], authors presents two interactive methods to improve the results of a classification task: the first one is an interactive decision tree construction algorithm with a help mechanism based on SVM; the second one is a visualization method used to try to explain SVM results. In particular, it uses a histogram of the data distribution according to the distance to the boundary and linked, a set of scatter-plot matrices or the parallel coordinates. This method can also be used to help the user in the parameter tuning step of SVM algorithm and reduce significantly the time needed for the classification.

3 Motivations and Requirements

Two or three-dimensional representation is probably the most 'natural' metaphor a visualization system can offer to model object relationships. This is how we perceive world as humans: two objects that are 'close' each other are probably more similar than two objects far away. The interactive visualization and navigation of such space becomes a means to browse and explore the corpus which match predetermined characteristics [Chalmers and Chitson, 1992, Wise et al., 1995, Wise et al., 1995, Becker, 1997, Rohrer et al., 1999, Becks et al., 2000, Kohonen, 1995, Ankerst et al., 2000, Harrell, 2006, Mozina et al., 2004, Leban et al., 2006, Poulin et al., 2006, Poulet, 2008, Seifert and Lex, 2009].

In [Becker, 1997], a list of desired requirements for the visualization of the structure of probabilistic classifiers are discussed. We summarize the main points here since we use them as a basis for the design of our visualization tool. In particular, users should be able:

(1) to quickly grasp the primary factors influencing the classification,
(2) to see the whole model and understand how it applies to classification objects, rather than the visualization being specific to a single object,
(3) to compare the relative evidence contributed by every value of every attribute,
(4) to see a characterization of a given class, that is a list of attributes that differentiate that class from others,
(5) to infer record counts and confidence in the shown probabilities so that the reliability of the classifier's prediction for specific values can be assessed quickly from the graphics,
(6) to interact with the visualization to perform classification.

Moreover, there is one last requirement which concerns the system:

(7) the system should handle many attributes without creating an incomprehensible visualization or a scene that is impractical to manipulate.

In this chapter we present a state-of-the-art visualization tool for Bayesian classifiers that can help the user understand why a classifier makes the predictions it does given a vector of parameters in input [Di Nunzio and Sordoni, 2012]. The user can interact with the classifier by: (i) selecting different parametric distributions, (ii) choosing different smoothing methods (iii) changing models' parameters.

During the requirements analysis, we analyzed the possible alternatives currently available for interactive R plots. The first choice was RGGobi which is one of the most powerful R tools for interactive data analysis [Lang et al., 2011]. However, in our case it was not possible to adapt it to the particular two-dimensional visualization model. A possible option could have been the RStudio [2] environment with its very useful "manipulate" package [RStudio, 2011]. Unfortunately, this package is available only within the RStudio environment and we did not want to force a user to install the whole IDE. For these reasons, we decided to create our own data visualization system based on R packages which are available on the Comprehensive R Archive Network (CRAN). [3] In particular, we make use of the *ggplot2* plotting system for R that has a strong underlying model which supports

[2] `http://www.rstudio.org/`
[3] `http://cran.r-project.org/`

the production of any kind of statistical graphic. [4]

3.1 R Packages Requirements

The code developed for this chapter requires the following R packages:

- *gWidgets*: it provides a toolkit-independent API for building interactive GUIs [Verzani, 2012];
- *gWidgetsRGtk2*: it allows the gWidgets API to use the RGtk2 package allowing the use of the GTK libraries within R [Lawrence and Verzani, 2012];
- *cairoDevice*: it allows the user to embed an R plot in a GTK user interface constructed with RGtk2 [Lawrence, 2011];
- *ggplot2*: it is an implementation of the grammar of graphics in R [Wickham, 2009];
- *tm*: a text mining framework in R [Feinerer et al., 2008];
- *Matrix*: implementation of dense and sparse matrices and fast operations on them using Lapack and SuiteSparse [Bates and Maechler, 2007].

4 Probabilistic Framework of NB Classifiers

In this section we present the Bayesian framework which characterizes the nature of the classification problem. This framework is based upon two commonly used assumptions: the data is produced by a mixture model, and there is a one-to-one correspondence between mixture components and classes [Domingos and Pazzani, 1997, Nigam et al., 1998].

In Section 1, we described machine learning as the problem of guessing the process that explains the data we observe. In the classification problem we want to characterize, every object is generated according to a probability distribution given by a mixture model. In later subsections, we address the problem of choosing the parametric form of the model (Section 4.1). For the moment, we can generically assume that all the parameters of interest are stored into a vector θ. Mixture components are encoded in a random variable $C = \{c_1, ..., c_n\}$ and objects in a random variable $O = \{o_1, ..., o_d\}$. The generative process for a classification object o_j consists in selecting a component according to the class probabilities $P(C = c_i; \theta) \equiv P(c_i; \theta)$, then picking o_j with probability

[4] `http://had.co.nz/ggplot2/`

$P(O = o_j|c_i;\theta) \equiv P(o_j|c_i;\theta)$. Thus, the probability of generating o_j can be found by marginalizing out the class variable:

$$P(o_j;\theta) = \sum_{i=1}^{n} P(o_j|c_i;\theta)P(c_i;\theta) \ . \tag{1}$$

Let us suppose that an 'oracle' tells us the optimal estimate $\hat{\theta}$ of the parameters of this problem. How do we find the class c^* of an unlabeled object o_j? One straightforward solution consists in selecting the mode of the posterior probability distribution of the class variable given the object:

$$c^* = \arg\max_{c_i} P(c_i|o_j;\hat{\theta}) = \arg\max_{c_i} \frac{P(o_j|c_i;\hat{\theta})P(c_i;\hat{\theta})}{P(o_j;\hat{\theta})} = \arg\max_{c_i} P(o_j,c_i;\hat{\theta}) \ , \tag{2}$$

which is obtained by a simple application of the Bayes' rule and by considering that $P(o_j;\theta)$ does not depend on c_i. In pattern classification problems, objects usually belong to more than one class (this problem is usually known as multi-label classification). For practical and efficiency reasons, instead of building one classifier able to "attach" n labels it is easier to build n binary classifiers able to decide whether the object belong to class c_i or not, $\bar{c}_i = C - c_i$. This is called binary classification.

Before digging into more details, we shall present the very core "object" of our system: the `nb` function reported in Listing 1.

```
1  nb <- function(model = "bernoulli", smoothing = "laplace") {
2    # Initialize a simple Naive-Bayes binary classifier
3    #
4    # Args:
5    #   model: The parametric form of the features distribution.
6    #   smoothing: The smoothing method to be used for the estimation.
7    #
8    # Returns:
9    #   A Naive-Bayes classifier ready to be estimated.
10
11   # Initialize the object and the basic structures
12   inst <- list()
13   class(inst) <- c("nb", model)
14   inst$type <- model
15   inst$smoothing <- smoothing
16   inst$classes.names <- list()
17   inst$features.num <- 0
18   inst$examples.num <- 0
19   # Create distributional parameters
20   inst$smoothing.params <- list()
21   inst$features.params <- list()
22   inst$classes.params <- list()
```

```
23    # Sufficient statistics for the objects' parametric distribution
24    inst$features.freq <- list()
25    # Data structure to speed-up computation
26    inst$features.freq.tot <- list()
27
28    return(inst)
29  }
```

Listing 1. `nb` function (file *nb.R*)

This function creates a generic empty "object" (the list `inst`, line 12) that contains all the informations about the model: its parametric form and smoothing type, feature counts, sufficient statistics and estimated parameters. It is valuable to consider the step done in line 13: the parametric form specified in the variable `model` is added to the object class. This enables us to use R powerful "generic" function paradigm which turns out to be useful in dealing with the estimation phase (see Section 4.1).

In the following subsections, we describe in more detail some parts of the classification model that we believe important for the visualization tool:

- How do we choose the parametric form of our model?
- Can we tune the estimate $\hat{\theta}$ of the optimal parameters θ?
- Once we have the estimate, can we measure how far we are from the optimal decision?

4.1 Choosing the model

Choosing the model of our problem means finding the right mathematical description according to some hypotheses we can make. Actually, there are two things to consider: how to model the distribution of the classes $P(c_i;\theta)$ and how to model the distribution of the objects in a class $P(o_j|c_i;\theta)$.

The choice of the model for the first point is easier when we consider the binary classification problem: the class we observe can be either c_i or $\bar{c}_i$. This binary setting can well be modeled by a Bernoulli random variable:

$$c_i \sim \text{Bern}(\theta_c)\ , \tag{3}$$

where the notation θ_c is used to specify the parameters relevant to the class distribution.

In the *nb.R* file (line 22), the parameters of the mixture are stored under the list key

`classes.params`. The estimation of the Bernoulli parameter θ_c calculated by Maximum Likelihood Estimation (MLE) is simply the number of examples of that class divided by the total number of examples (see Section 4.2 for a discussion about smoothing). The *R* code for this estimation process is given in Listing 2.

```
55  # Bernoulli parameter for each class
56  x$classes.params <- as.vector(table(labels)[x$classes.names]) / x$examples.num
```

Listing 2. Parameters of the mixture (file *nb.R*)

Here, `x` is the `nb` model under estimation, `x$examples.num` is the total number of examples, `labels` a vector which specifies which objects belong to that category and `x$classes.names` the name of the classes for this classification problem. Constructing a `table` from the labels vector is a useful trick to automatically count the number of examples in each class.

The choice of the mathematical model for the distribution of the objects requires more attention. Hereafter, we present three possible models to parametrize the conditional probability $P(o_j|c_i;\theta)$.

4.1.1 Multivariate Bernoulli model

In the multivariate Bernoulli model, an object is a binary vector over the space of features. Given a set of features F, $|F| = m$, each object o_j is represented as a vector of m Bernoulli random variables $o_j \equiv (f_1, ..., f_m)$ such that:

$$f_k \sim \text{Bern}(\theta_{f_k|c}) \ . \tag{4}$$

We can write the probability of an object by using the NB conditional independence assumption which states that features variables are independent given the class variable. Formally:

$$P(o_j|c_i;\theta) = \prod_{k=1}^{m} P(f_k|c_i;\theta) = \prod_{k=1}^{m} \theta_{f_k|c_i}^{x_k}(1-\theta_{f_k|c_i})^{1-x_k} \ , \tag{5}$$

where x_k is either 0 or 1 indicating whether feature f_k is present or absent in object o_j. When the number of features is very large, this product goes quickly to zero. For this

reason, a monotonic transformation such as a log transformation is usually performed:

$$\log(P(o_j|c_i;\theta)) = \log\left(\prod_{k=1}^{m} P(f_k|c_i;\theta)\right) = \sum_{k=1}^{m} x_k \log\left(\frac{\theta_{f_k|c_i}}{1-\theta_{f_k|c_i}}\right) + \sum_{k=1}^{m} \log(1-\theta_{f_k|c_i}) \ , \tag{6}$$

This last equation is not only a way to avoid arithmetical anomalies but also a very efficient implementation of the same calculation: the last sum on the RHS can be precomputed and the first term ranges only over the features appearing in the current object ($x_k = 1$).

4.1.2 Multinomial model

In contrast to the multivariate Bernoulli model, in the multinomial model we have one single random variable F which can take values over the set of features. By assuming that each feature event is independent of each other, an object o_j is represented as a vector of frequencies whose entries correspond to features. This vector is drawn from a multinomial distribution:

$$o_j \equiv (N_{1,j},...,N_{m,j}) \sim Multinomial(\theta_{f|c}) \ . \tag{7}$$

where $N_{k,j}$ indicates the number of times feature f_k appears in the object o_j. The probability is proportional to:

$$P(o_j|c_i;\theta) \propto \prod_{k=1}^{m} \theta_{f_k|c_i}^{N_{kj}} \ . \tag{8}$$

Again, we can use a monotonic transformation to avoid zero-probabilities:

$$\log\left(P(o_j|c_i;\theta)\right) \propto \sum_{k=1}^{m} N_{kj} \log(\theta_{f_k|c_i}) \ . \tag{9}$$

4.1.3 Poisson model

In the Poisson model, an object is generated by a multivariate Poisson random variable. Each object o_j is represented as a m-dimensional vector of frequencies, $o_j \equiv (N_{1,j},...,N_{m,j})$, and each feature count is governed by a Poisson random variable:

$$N_{i,\cdot} \sim \text{Pois}(\theta_{f_i|c}) \ . \tag{10}$$

Using the NB conditional independence assumption, we can write the probability of the object as:

$$P(o_j|c_i;\theta) \propto \prod_{k=1}^{m} \theta_{f_k|c_i}^{N_{kj}} e^{-\theta_{f_k|c_i}} \ , \tag{11}$$

and, by taking the logs we obtain:

$$\log\left(P(o_j|c_i;\theta)\right) \propto \sum_{k=1}^{m} \left(N_{kj} \log\left(\theta_{f_k|c_i}\right) - \theta_{f_k|c_i}\right) \ . \tag{12}$$

With these concepts in hand, we can easily apply Equation 2 in order to calculate the posterior distribution $P(c_i|o_j;\theta)$ and classify the unlabeled object o_j. The generic function `nbClassify` allows you to call the correct classification function according to the class of the object `x` that is passed as argument. An example of the implementation of the classification function using Bernoulli model (Equation 6) is reported in Listing 3.

```
204 nbClassify.bernoulli <- function(x, dataset) {
205   dataset <- prepareDataset(x, dataset)
206
207   # precompute the sum over features of log(1 - param)
208   sumnegative  <- colSums(log(1 - x$features.params))
209
210   # compute p(d, c)
211   logfeatures  <- log(x$features.params)
212   lognfeatures <- log(1 - x$features.params)
213   scores <- dataset %*% (logfeatures - lognfeatures)
214   scores <- t(t(scores) + sumnegative + log(x$classes.params))
215
216   return(scores)
217 }
```

Listing 3. Classification function for Bernoulli model (file *nb.R*)

The methods returns the joint probabilities $P(o_j, c_i; \hat{\theta})$ for the two classes c_i, $\bar{c}_i$, for all the objects' vectors contained in the `dataset` matrix.

4.2 Estimating the parameters

When the model and the features which characterize the objects are defined, the next step is to estimate the parameters of our model θ. In the examples of the previous sections, we let an 'oracle' to tell the vectors of estimates of the parameters. In a realistic situation, how can we estimate these parameters?

In general, there are two main paths you may follow to accomplish this task: the MLE or the Bayesian approach to estimation.[5] With MLE, under the hypothesis that the data D contains observations which are independent and identically distributed, the estimate $\hat{\theta}^{MLE}$ is calculated by maximizing the likelihood of the data with respect to the parameter:

$$\hat{\theta}^{MLE} = \arg\max_{\theta} P(D|\theta) = \arg\max_{\theta} \prod_{j \in D} P(o_j|\theta) \tag{13}$$

For many models, like the ones we presented in this paper, a maximum likelihood estimator can be found as an explicit function of the observed data. However, due to data sparseness, the MLE of the probability of unseen features tend to be zero [Gelman et al., 2003]. To prevent this undesirable "zero-probability" behavior, we need to smooth the estimates of our parameters: the estimation of a parameter is strongly tied to smoothing techniques.

In a Bayesian framework, smoothing is implicitly considered by treating model parameters θ as random variables governed by a probability distribution, or *prior*, indicated as $\mathrm{P}(\theta)$. The prior can be seen as encoding our "beliefs" about how θ should behave. Intuitively, this solves the problem of zero probability features because we relax the strict dependency of the estimates upon the statistics gathered from data. The parameters controlling the prior are usually called *hyper-parameters*.[6] Being bayesian means adjusting the prior distribution upon observed data, as known as computing the *posterior* given the data $\mathrm{P}(\theta|D)$. This is achieved through the application of the Bayes' rule:

$$\overbrace{\mathrm{P}(\theta|D)}^{posterior} = \frac{\overbrace{\mathrm{P}(D|\theta)}^{likelihood}\overbrace{\mathrm{P}(\theta)}^{prior}}{\mathrm{P}(D)}, \tag{14}$$

where D represent our data, and $P(D)$ is the probability of the particular instance D according some generative model of the data.

The bayesian estimator $\hat{\theta}^{B}$ of the parameter is the posterior mean:

$$\hat{\theta}^{B} = E_{\theta|D}[\theta] = \int \theta P(\theta|D) d\theta. \tag{15}$$

[5] Actually, there is a third way of estimating the probabilities which is the Maximum a Posteriori (MAP) estimator and a complete account of it falls outside the scope of this paper.

[6] The prefix "hyper" is used to distinguish the parameters of the prior from the parameters of the original model, like θ.

Table 1
Differences between the MLE and the Bayesian approach where $n_{k,i}$ is the number of objects belonging to c_i in which feature f_k appears and n_i the total number of objects in c_i, $N_{k,i}$ the frequency of feature f_k in c_i, α_k the k-th component of vector $\vec{\alpha}$.

model	$\hat{\theta}^{MLE}$	$\hat{\theta}^{B}$
Bernoulli	$\frac{n_{k,i}}{n_i}$	$\frac{n_{k,i}+\alpha}{n_i+\alpha+\beta}$
Multinomial	$\frac{N_{k,i}}{\sum_k(N_{k,i})}$	$\frac{N_{k,i}+\alpha_k}{\sum_k(N_{k,i}+\alpha_k)}$
Poisson	$\frac{N_{k,i}}{n_i}$	$\frac{N_{k,i}+\alpha}{n_i+\alpha+\beta}$

Equation 15 can be mathematically hard to solve because of the integration over a product of two functions, $P(\theta|D)$ and $P(\theta)$. One way to approach this problem is to find the "conjugate" prior of the likelihood function $P(D|\theta)$ which makes the posterior function $P(\theta|D)$ come out with the same functional form as the prior. If the likelihood belongs to the exponential family there always exists a conjugate prior. NB models have a likelihood of this type:

- The multivariate Bernoulli model conjugate prior is the Beta distribution $Beta(\theta;\alpha,\beta)$,
- The Multinomial model conjugate prior is the distribution $Dir(\theta;\vec{\alpha})$,
- the multi-variate Poisson conjugate prior is the Gamma distribution $Gamma(\theta;\alpha,\beta)$.

where α, β are hyper-parameters, and $\vec{\alpha} = \alpha_1,...,\alpha_m$ is a vector of hyper parameters, one for each features.

A comparison of the two estimation methods is given in Table 1. We can easily see that the zero-probability behavior arises in MLE when the counts $n_{k,i}$ or $N_{k,i}$ are zero. On the contrary, Bayesian estimation adds "pseudo-counts" that avoid the problem. Our visualization tool implements three kind of smoothing methods: Laplace, Bayesian and Fixed Interpolation. The Laplace method is a special case of the Bayesian estimation and is obtained by setting the hyper parameters $\alpha = 1$, $\beta = 1$ and $\alpha_k = 1$, for each k. The Fixed interpolation method solves the problem by interpolating the MLE estimator with a "collection" model, in which the statistics are gathered regardless the class variable. A Fixed interpolation estimator $\theta^I_{f_k|c_i}$ for a Bernoulli model would be:

$$\theta^I_{f_k|c_i} = \lambda \frac{n_{k,i}}{n_i} + (1-\lambda)\frac{\sum_i n_{k,i}}{\sum_i n_i}, \tag{16}$$

where $\lambda \in [0,1]$ is a free parameter.

The generic function *nbEstimate* (Listing 4) gathers the sufficient statistics from the vectors contained in the variable `dataset` and selects the correct estimation function according to the class of model `x` we have chosen. Specifically this two steps are: (i) count the occurrences of the dataset and store them in the model (Listing 4); (ii) call the generic function `nbUpdate` which computes posterior estimate by plugging the hyper-parameters (Listing 5).

```
# Estimate Bernoulli parameters
nbEstimate.bernoulli <- function(x, dataset, labels, params) {
  # Preprocess dataset for Bernoulli
  dataset <- prepareDataset(x, dataset)

  # Initialize dataset specific variables
  # Sorted class names
  x$classes.names <- sort(unique(labels), decreasing=TRUE)
  x$smoothing.params <- params

  # Number of unique features
  x$features.num <- dim(dataset)[2]
  x$examples.num <- dim(dataset)[1]

  # Compute sufficient statistics for each class
  # features.freq is a features x class matrix
  # features.freq.tot is a tot_features x class matrix for the current model
  x$features.freq <- sapply(x$classes.names,
                            function(r) colSums(dataset[c(labels == r),]))
  x$features.freq.tot <- as.vector(table(labels)[x$classes.names])

  # Bernoulli parameter for each class
  x$classes.params <- as.vector(table(labels)[x$classes.names]) / x$examples.num

  # Compute probability estimates
  x <- nbUpdate(x, params)
  return(x)
}
```

Listing 4. Estimation of Bernoulli model parameters (file *nb.R*)

```
# Update estimates
nbUpdate <- function(x, ...) {
  UseMethod("nbUpdate")
}

# Update Bernoulli estimates
nbUpdate.bernoulli <- function(x, params) {
  # Laplace smoothing for Bernoulli
  if (x$smoothing == "laplace") {
    x$features.params <- t(t(x$features.freq + 1) / (x$features.freq.tot + 2))
  }
  # Beta prior
  else if (x$smoothing == "prior") {
    x$smoothing.params$alpha <- (alpha <- params$alpha)
    x$smoothing.params$beta  <- (beta <- params$beta)
    x$features.params <- t(t(x$features.freq + alpha) /
```

```
133                           (x$features.freq.tot + (alpha + beta)))
134   }
135   # Jelinek-Mercer interpolation
136   else if (x$smoothing == "interpolation") {
137     x$smoothing.params$lambda <- (lambda <- params$lambda)
138     features.freqs    <- rowSums(x$features.freq)
139     collection.freqs  <- sum(x$features.freq.tot)
140     x$features.params <- (1 - lambda) * t(t(x$features.freq) / x$features.freq.tot) +
141                          lambda * (features.freqs / collection.freqs)
142   }
143   return(x)
144 }
```

Listing 5. Updating Bernoulli parameters (file *nb.R*)

5 Two-Dimensional Visualization System

The model which upholds the visualization tool defines a direct relationship between the probability of an object given a category of interest and a point on a two-dimensional space [Di Nunzio, 2009]. The idea is to associate each object of the dataset to a point in the two-dimensional space: the abscissa reflects how much the object is relevant to the category, the ordinate reflects how much the object is not relevant to the category. In this light, it is possible to graph entire collections of objects on a Cartesian plane. Remembering that in a binary classification setting, we classify the object o_j under category c_i if

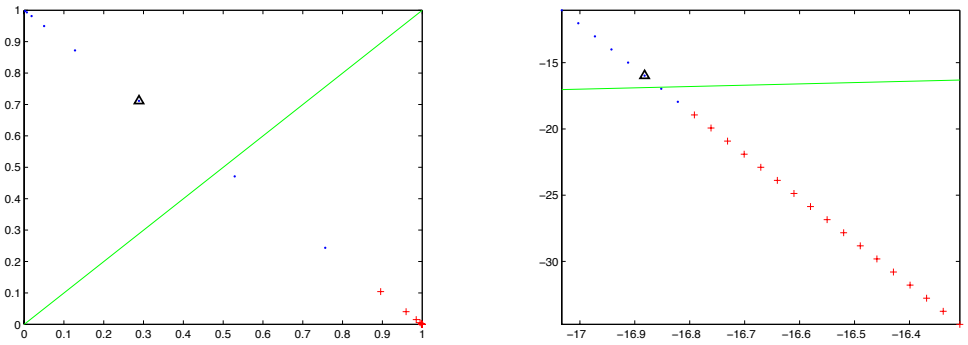
$$P(c_i|o_j;\hat{\theta}) > P(\bar{c}_i|o_j;\hat{\theta}) \,, \tag{17}$$

we can get the two coordinates of a NB classifier in the following way:

$$\underbrace{P_1}_{x} > \underbrace{P_0}_{y} \,, \tag{18}$$

where $P_1 = P(c_i|o_j;\hat{\theta})$, $P_0 = P(\bar{c}_i|o_j;\hat{\theta})$, $c_i = c_1$, $\bar{c}_i = c_0$. In Figure 1(a) three points in this two-dimensional space are shown together with the decision line. The visual metaphor is immediate: if the point is 'below' the decision line, the object is classified under category c_1 (because $x > y$), if the point is 'above' the line, the object is classified under category c_0 (because $x < y$). Points that lie exactly on the line are those for which we need to take an explicit decision (because they have the same chance of belonging to either category). [7]

[7] In this paper we classify these documents under category c_0.



(a) Normalized probabilities (b) Non-normalized log probabilities

The further distant the point from the line, the higher confidence in the classification. In this figure, the point highlighted with a triangle would be classified under c_0.

5.1 Design Choices

The two-dimensional visualization described earlier there may present the following problems:

- In a binary classification problem $P_0 = 1 - P_1$ and all the points are on the segment (0,1)-(1,0), see Figure 1(a). With datasets of the size of thousands of objects, points may result too cluttered.
- There may be arithmetical anomalies given the fact that the product of the probability of the features goes rapidly to zero [8] . Either P_0 or P_1 can be approximated to zero given an insufficient number of bits.

To avoid points cluttering, we take the non-normalized probabilities:

$$P(o_j|c_1;\hat{\theta})P(c_1;\hat{\theta}) > P(o_j|c_0;\hat{\theta})P(c_0;\hat{\theta}) \tag{19}$$

which means not dividing by the probability of the data $P(o_j|\theta)$. This does not change the result of the decision as it was justified in Equation 2. To eliminate the arithmetical

[8] If we have 50 features and the probability of each feature given a class is 0.5, the product is $0.5^{50} \sim 10^{-15}$.

anomaly, we take the logarithm of both sides of the equation:

$$\log(P(o_j|c_1;\theta)) + \log(P(c_1;\theta)) > \log(P(o_j|c_0;\theta)) + \log(P(c_0;\theta)) \tag{20}$$

Figure 1(a) shows the result of this transformation. Note that we are now in the third quadrant of the cartesian plane, since we are adding logarithm of probabilities, therefore logarithms of number between 0 and 1.

5.2 Visualization Design

The system we propose consists of several visual components that correspond to the points addressed in Section 4. The main components of the system are:

- *View Panel*: displays the two-dimensional plot of the dataset according to the choices of the user.
- *Interaction Panel*: allows for the interaction between the user and the parameters of the probabilistic models.
- *Performance Panel*: displays the performance measures of the model.

Figure 1 shows the Main window and the panels. The interactivity of the plot is realized by means of the *gWidgets* package and the layout is realized by grouping widgets together.

The link between the interactive window and the probabilistic model starts when the *hndleCreateModel* function is called. This function is mainly the automated version of the creation of a naïve bayes model we showed in the previous section in Listing 10.

```
13  hndleCreateModel <- function (h, ...) {
14
15    # Get selected model
16    model <- svalue(combo.model)
17    # Get selected smoothing method
18    smoothing <- svalue(combo.smooth)
19
20    # Delete parameters layouts
21    for (w in list.layouts) {
22      delete(frm.params, w)
23    }
24
25    # Build widgets according to prior and model
26    if (smoothing == "prior") {
27      if (model == "bernoulli") {
28        names(frm.params) <- "Adjust Beta Prior"
29        add(frm.params, list.layouts$bernprior)
30      } else if (model == "poisson") {
31        names(frm.params) <- "Adjust Gamma Prior"
```

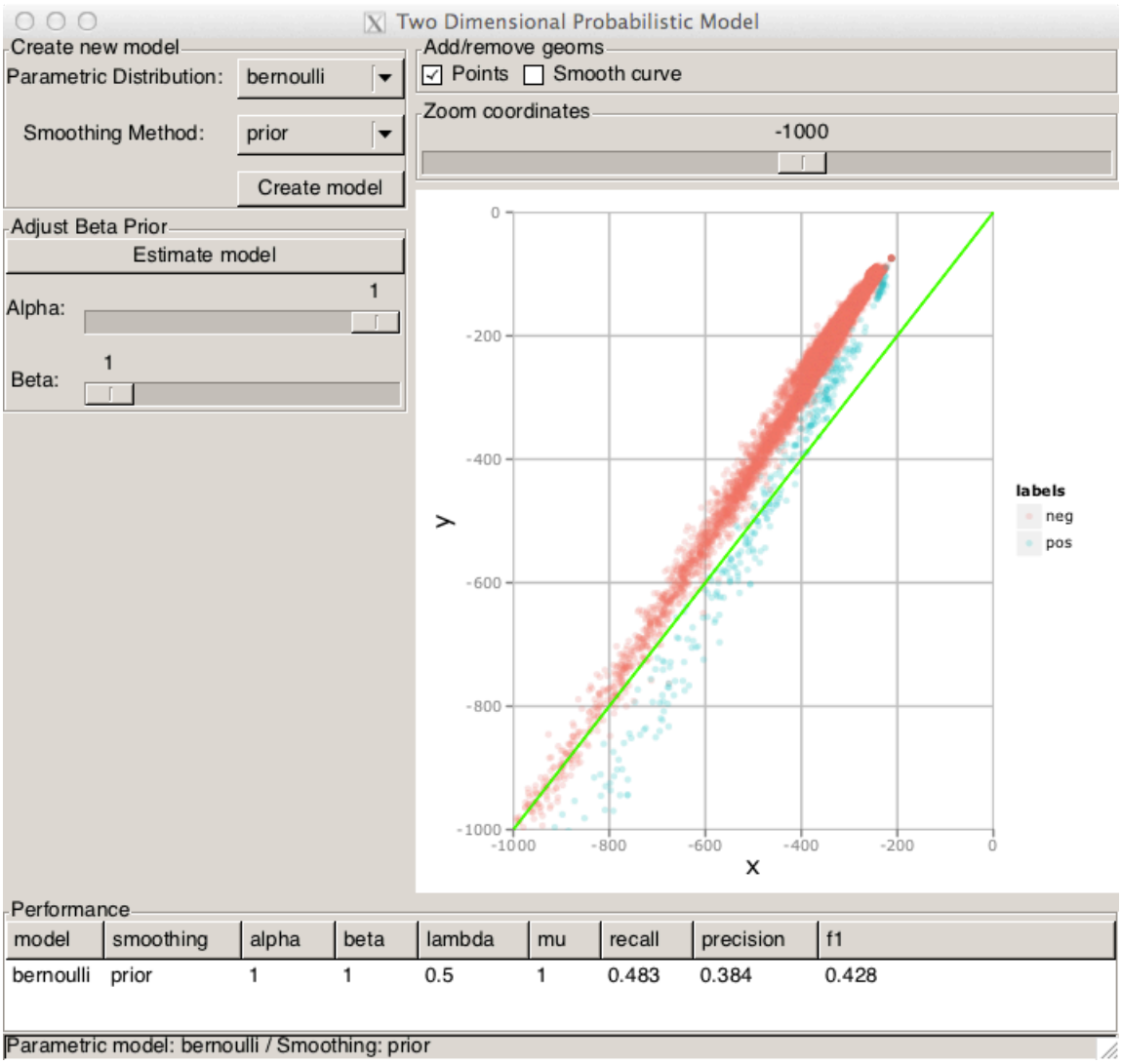


Fig. 1. Visualization tool: main window

```
32          add(frm.params, list.layouts$bernprior)
33        } else {
34          names(frm.params) <- "Adjust Dirichlet Prior"
35          add(frm.params, list.layouts$multiprior)
36        }
37      } else if (smoothing == "interpolation") {
38        names(frm.params) <- "Adjust Interpolation Factor"
39        add(frm.params, list.layouts$interp)
40      }
41
42      # Initialize parameters
43      params <<- lapply(list.sliders, svalue)
```

```
44
45    # Create the model
46    nbmodel <<- nb(model, smoothing)
47    nbmodel <<- nbEstimate(nbmodel, dataset, labels, params)
48
49    # Update everything
50    fireUpdate()
51
52    # Update status bar value
53    svalue(status.bar) <- paste("Parametric model:", model, "/ Smoothing:", smoothing)
54  }
```

Listing 6. Initialize model by calling the *hndleCreateModel* function (file *twodm.R*).

Note that from line 21 to line 41 we first hide the interactive widgets that are used to tune the prior hyper-parameters and then we build them according to the desired model.

When the "Estimate Mode" button is clicked, the window requests an update of the coordinates to the model. The model computes the estimates of the probabilities according to the new parameters and sends the scores to the view, as shown in Listing 7.

```
57  fireUpdate <- function() {
58    # Update the value of the parameters
59    params  <<- lapply(list.sliders, svalue)
60
61    # Update the model
62    nbmodel <<- nbUpdate(nbmodel, params)
63    # Update scores
64    scores  <<- nbClassify(nbmodel, dataset)
65    # Update plot and performances
66    updatePlot()
67    updatePerformances()
68  }
69
70  # Update the plot
71  updatePlot <- function(...) {
72    # Update dataframe
73    df$x <<- scores[, 1]
74    df$y <<- scores[, 2]
75
76    pp <- p
77    pp <- (pp %+% df)
78    # Add layers
79    if (svalue(chk.points))
80    {
81      pp <- pp + geom_point(alpha = 0.2)
82    }
83    if (svalue(chk.smooth))
84    {
85      pp <- pp + geom_smooth()
86    }
87    # Add decision line
88    pp <- pp + geom_abline(colour = "green")
89
```

```
90      # Adjust coordinate
91      pp <- pp + coord_cartesian(xlim = c(svalue(slider.coord), 0),
92                                 ylim = c(svalue(slider.coord), 0))
93      # Plot
94      print(pp)
95    }
```

Listing 7. Update probabilities and plot results (file *twodm.R*).

5.3 A Case Study: Text Classification

The task of text classification is to assign one or multiple pre-defined class labels to a textual document. It has been a very popular research topic at the end of the '90s with the rapid increase of text in digital form such as web pages, newswire and scientific literature and the need of organize them. Today, classification has witnessed a new wave of interest due to new technologies that user can perform or new specific tasks such as: query classification, blog classification, patent classification, medical document classification.

For this case study, we used the Reuters-21578 collection which consists of 21,578 news stories appeared on the Reuters newswire in 1987. The documents manually assigned to categories are actually 12,902.

5.3.1 Creating Document Term Matrices

When dealing with text collections, there are a number of steps that are necessary to clean noisy documents and obtain a nicely formatted machine readable dataset. We used the *tm* package, a text mining package in R that gives many options to process of raw text files into document-term matrices. Listing 8 shows the sequence of steps to load the Reuters dataset.

```
1  require(tm)
2
3  # Use TextMining package to load and preprocess documents
4  loadReutersDataset <- function() {
5    cat(sprintf("load dataset...\n"))
6    reuters <- Corpus(DirSource("./data/reuters21578/"),
7                      readerControl = list(reader = readReut21578XML))
8    return(reuters)
9  }
10
11 # This function preprocess an XML Corpus (tm package) into a plain text dataset.
12 preprocessDataset <- function(dataset) {
13     # Transform XML into plain text
```

```
    cat(sprintf("transform into plain text...\n"))
    dataset_plain <- tm_map(dataset, as.PlainTextDocument)

    # Remove extra white spaces
    cat(sprintf("remove extra white spaces...\n"))
    dataset_plain <- tm_map(dataset_plain, stripWhitespace)

    # To lower case
    cat(sprintf("letters to lower case...\n"))
    dataset_plain <- tm_map(dataset_plain, tolower)

    # Remove stopwords
    cat(sprintf("remove stopwords...\n"))
    dataset_plain <- tm_map(dataset_plain, removeWords, stopwords("english"))

    # Stem words (currently not working on MAC OS X Lion)
    # dataset_plain <- tm_map(dataset_plain, stemDocument)

    return(dataset_plain)
}
```

Listing 8. Case study: load Reuters collection (file *preprocess_dataset.R*).

In Listing 9, we show the sequence of commands to initialize a collection before starting the two-dimensional view.

```
> source("twodm.R")
> # load dataset
> dataset <- loadReutersDataset()
load dataset...
> # transform XML documents into plain documents
> dataset_plain <- preprocessDataset(dataset)
transform into plain text...
remove extra white spaces...
letters to lower case...
remove stopwords...
> # build document-term matrix
> dtm <- createDocumentTermMatrix(dataset_plain)
> # extract matrix from corpus
> dataset_matrix <- extractDocumentTermMatrix(dtm)
> # extract labels
> labels <- extractLabels(dataset, "acq")
> # start two dimensional visualization
> twodm(dataset_matrix, labels)
```

Listing 9. Case study: loading Reuters dataset (file *preprocess_dataset.R*)

5.3.2 *Loading Existent Term Document Matrices*

Depending on the type of hardware available and on the size of the document collection, it might be more convenient to process the dataset outside R and produce a dataset in a sparse format: our system can handle datasets encoded in row-column-value triplets. For example, the first lines of the *reuters.matrix* are shown hereby:

```
1 70 1
1 1020 1
1 1045 1
1 1104 1
1 1121 1
[...]
```

Listing 10. First line of the file *reuters.matrix*

This very compact representation can be loaded as a table and transformed into a sparse matrix, as shown in Listing 11.

```
# Take in input a triplet form row,column,value
# and returns a sparse matrix
loadDataset <- function(dataset) {
  t <- read.table(dataset)
  return(sparseMatrix(t[,1], t[,2], x=t[,3]))
}
```

Listing 11. Case study: load *reuters.matrix* (file *preprocess_dataset.R*).

5.3.3 *Optimizing Parameters*

The output of Listing 9 is shown in Figure 3. We intentionally left the beta prior parameters equal to 1 to simulate a laplacian smoothing (in fact, $\alpha = \beta = 1$ corresponds to a uniform prior). The visualization tool immediately shows that in theory the two sets of documents are well separated, but in practice they are misplaced with respect to the (green) decision line (the line where $P_1 = P_0$)

By tweaking the two hyper-parameters, we can see the effects on both the plot and in the performance panel. The results are immediately evident: the cloud of point is better aligned with the decision line and all the performance measure have significantly increased. Note that in this Figure we intentionally changed the transparency level of the points to highlight the areas of higher density.

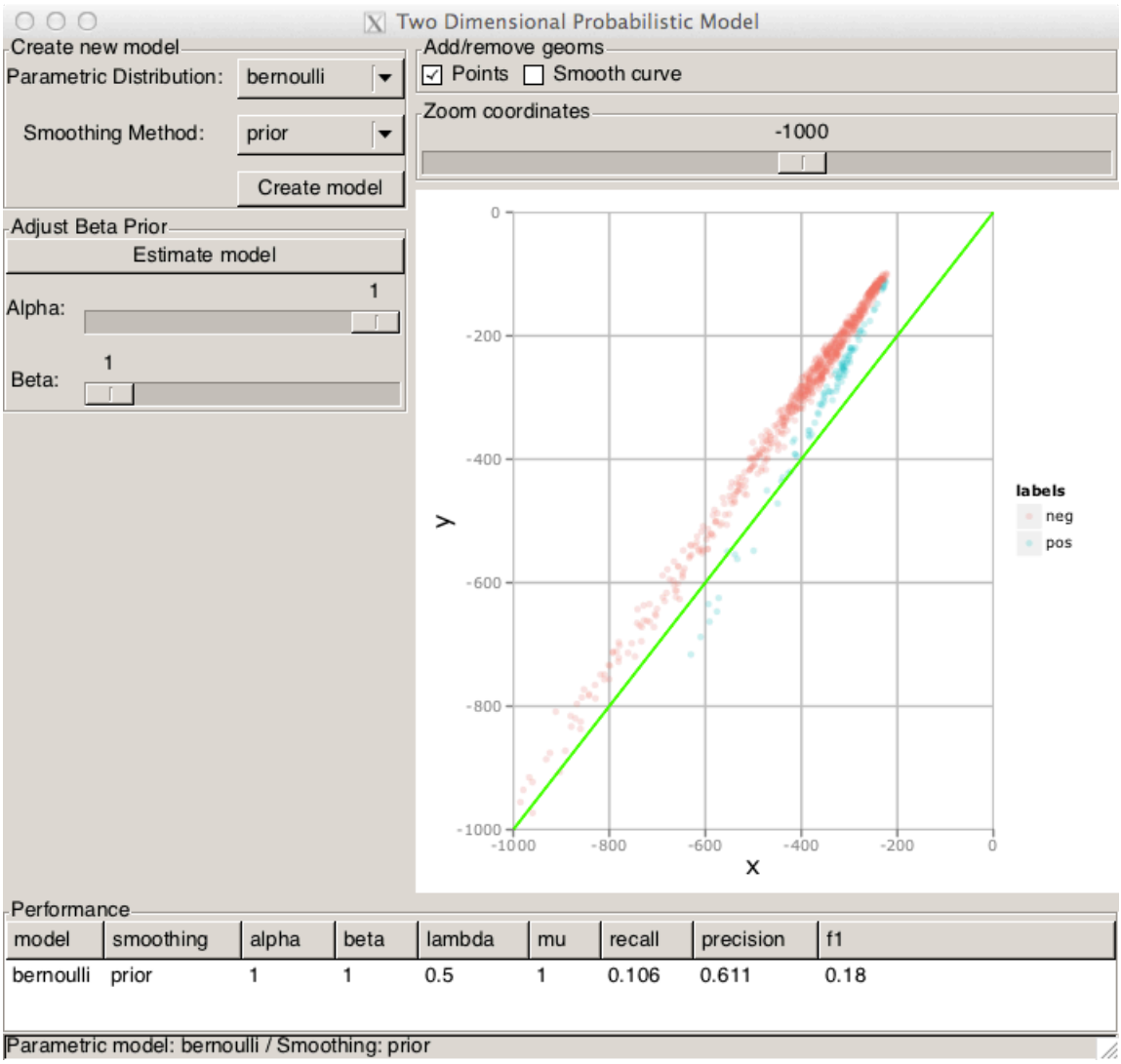


Fig. 2. Case study: Bernoulli with Beta prior

5.3.4 Comparing Models

With the two-dimensional visualization it is possible to assess the quality of a model quickly and compare it to other models. In Figure 4, Figure 5, the multinomial models and the Poisson model have been initialized with Laplacian smoothing (or uniform prior). In these plots two fitted curves, one for each class, have been computed by the `geom_smooth()` function of the *ggplot2* package. Listing 12 shows the function that draws the actual plot.

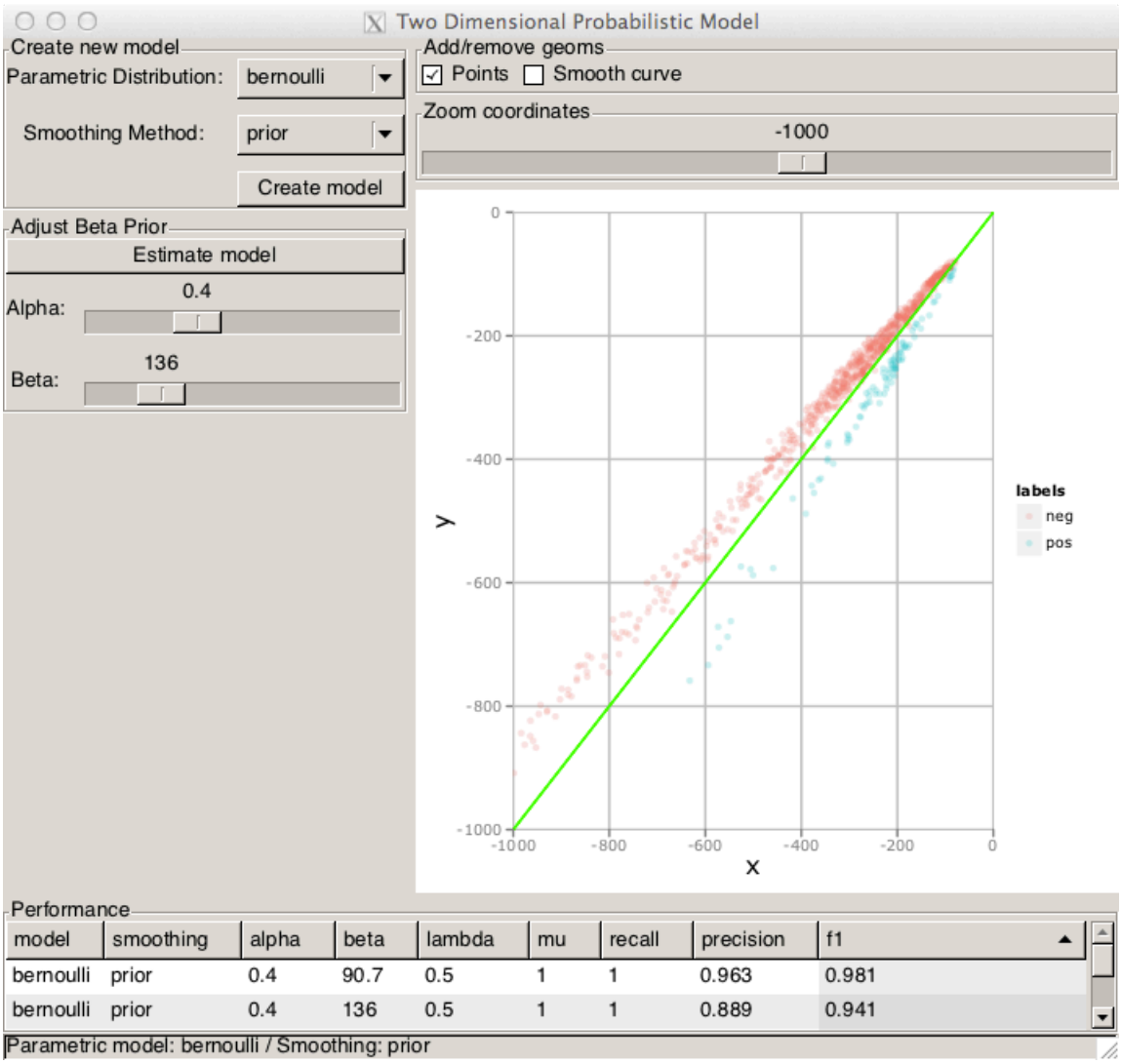


Fig. 3. Case study: Bernoulli with Beta prior, optimized

```
  # Update the plot
  updatePlot <- function(...) {
    # Update dataframe
    df$x <<- scores[, 1]
    df$y <<- scores[, 2]

    pp <- p
    pp <- (pp %+% df)
    # Add layers
    if (svalue(chk.points))
    {
```

```
81        pp <- pp + geom_point(alpha = 0.2)
82      }
83      if (svalue(chk.smooth))
84      {
85        pp <- pp + geom_smooth()
86      }
87      # Add decision line
88      pp <- pp + geom_abline(colour = "green")
89
90      # Adjust coordinate
91      pp <- pp + coord_cartesian(xlim = c(svalue(slider.coord), 0),
92                                 ylim = c(svalue(slider.coord), 0))
93      # Plot
94      print(pp)
95    }
```

Listing 12. Case study: plot Reuters collection (file *twodm.R*).

In theory, when the hyper-parameters are close to the optimal values, the two lines should be perfectly aligned and specular with respect to the decision line, for example Figure 4.

When one of the two lines (or both) intersects the decision line, it means that the model has not been optimized. In these cases, a tweaking of the hyper-parameters, by setting the smoothing method to prior, is required. Figure 6 shows the performance of an optimized Poisson model. Note that at the bottom of the window how the two models are now very close in terms of F1.

References

[Alpaydin, 2009] Alpaydin, E. (2009). *Introduction to Machine Learning*. Adaptive Computation and Machine Learning Series. The MIT Press, second edition.

[Ankerst et al., 2000] Ankerst, M., Ester, M., and Kriegel, H.-P. (2000). Towards an effective cooperation of the user and the computer for classification. In *KDD'00*, pages 179–188, Boston, MA, USA. ACM.

[Bates and Maechler, 2007] Bates, D. and Maechler, M. (2007). matrix: A matrix package for r. *R package version 0.99875-2, URL http://CRAN. R-project. org*, 15.

[Becker et al., 2002] Becker, B., Kohavi, R., and Sommerfield, D. (2002). Information visualization in data mining and knowledge discovery. In Fayyad, U., Grinstein, G. G., and Wierse, A., editors, *Information visualization in data mining and knowledge discovery*, chapter Visualizing the simple Baysian classifier, pages 237–249. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

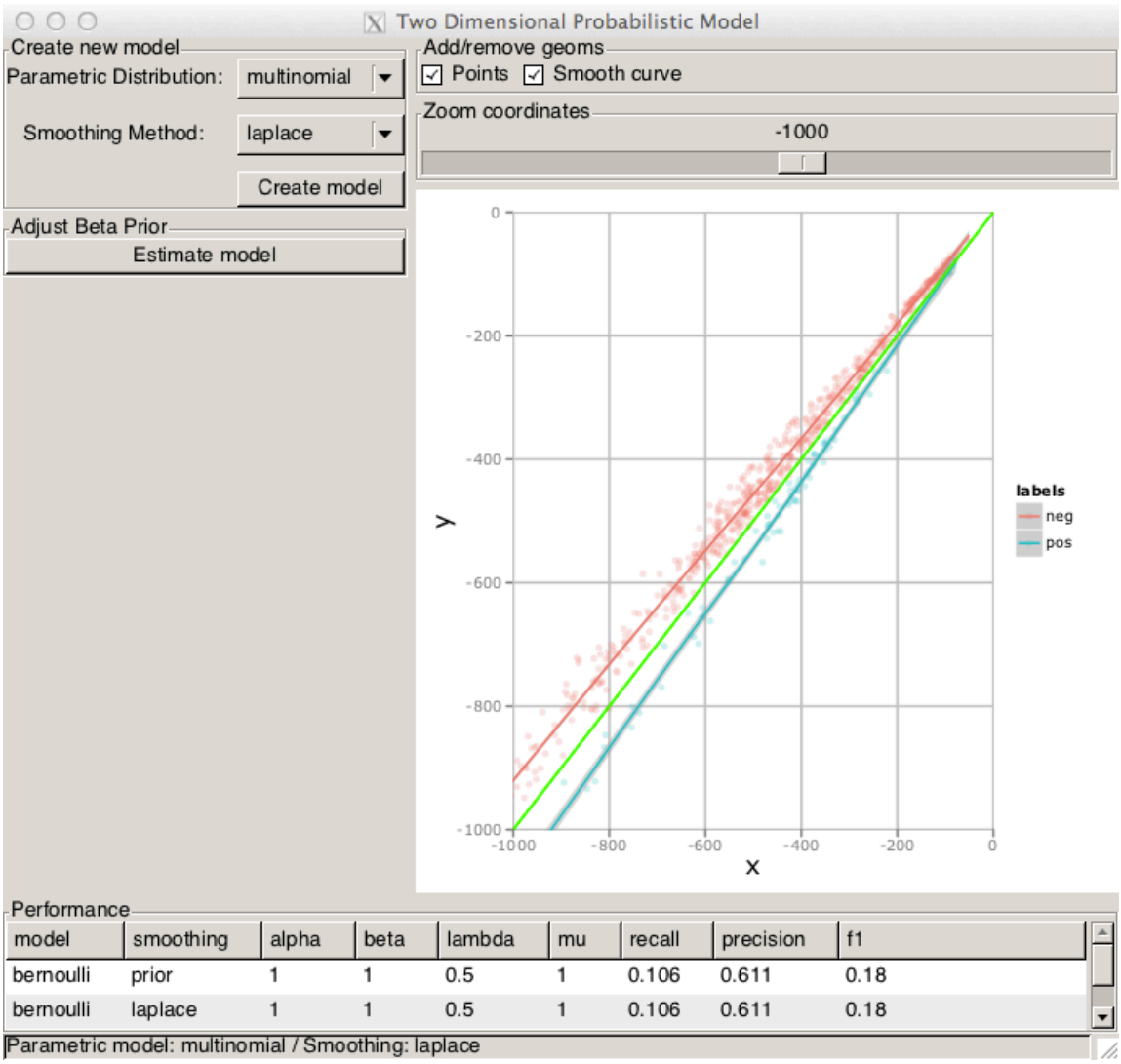


Fig. 4. Case study: Multinomial with Laplacian smoothing and fitted lines

[Becker, 1997] Becker, B. G. (1997). Using mineset for knowledge discovery. *IEEE Computer Graphics and Applications*, 17(4):75–78.

[Becks et al., 2000] Becks, A., Sklorz, S., and Jarke, M. (2000). Exploring the semantic structure of technical document collections: A cooperative systems approach. In Etzion, O. and Scheuermann, P., editors, *CoopIS*, volume 1901 of *Lecture Notes in Computer Science*, pages 120–125. Springer.

[Chalmers and Chitson, 1992] Chalmers, M. and Chitson, P. (1992). Bead: Explorations in

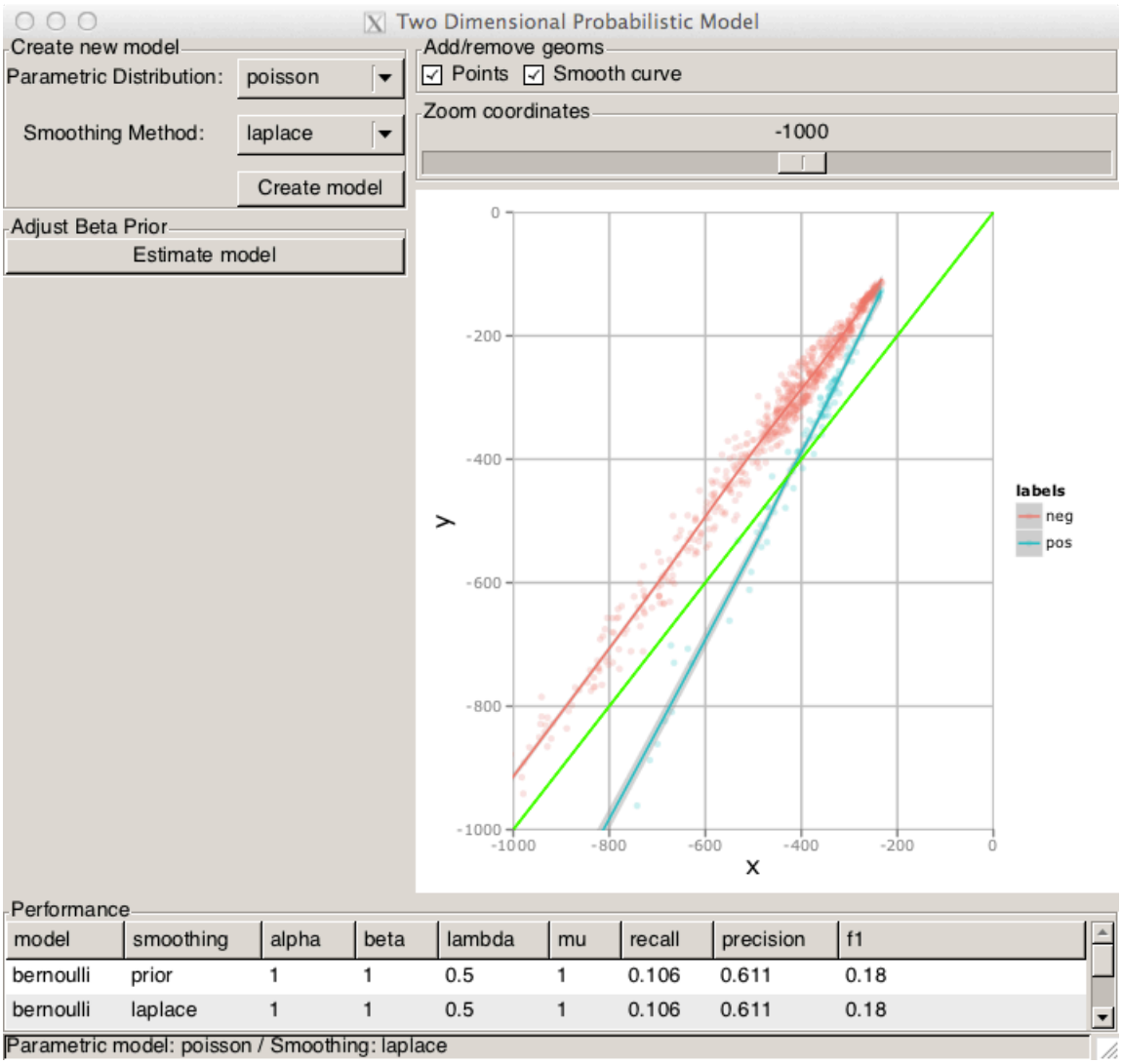


Fig. 5. Case study: Poisson with Laplacian smoothing and fitted lines

information visualization. In Belkin, N. J., Ingwersen, P., and Pejtersen, A. M., editors, *SIGIR*, pages 330–337. ACM.

[de Oliveira and Levkowitz, 2003] de Oliveira, M. C. F. and Levkowitz, H. (2003). From visual data exploration to visual data mining: A survey. *IEEE Trans. Vis. Comput. Graph.*, 9(3):378–394.

[Di Nunzio, 2009] Di Nunzio, G. (2009). Using scatterplots to understand and improve probabilistic models for text categorization and retrieval. *Int. J. Approx. Reasoning*, 50(7):945–

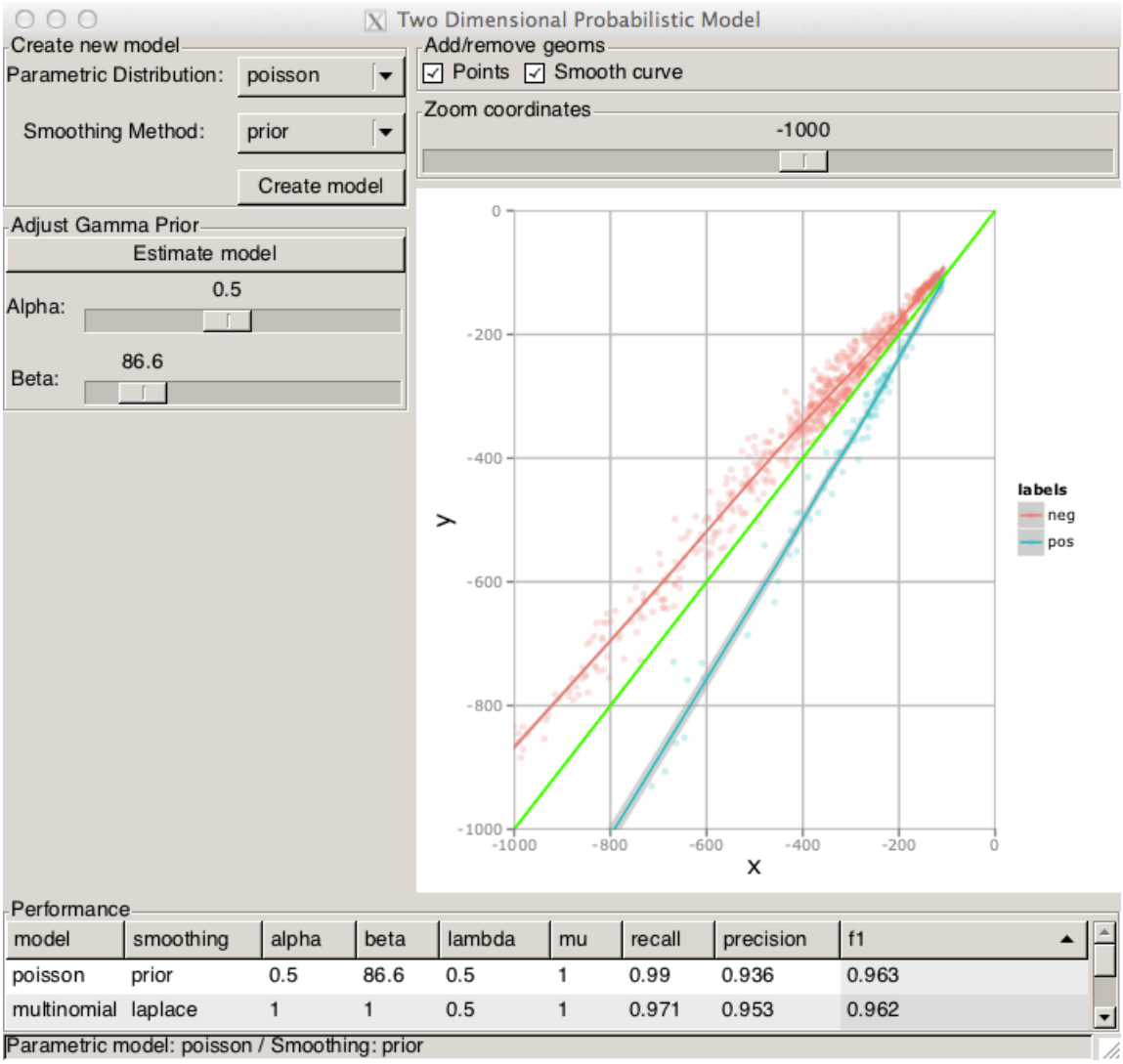


Fig. 6. Case study: Poisson with Gamma prior smoothing and fitted lines

956.

[Di Nunzio and Sordoni, 2012] Di Nunzio, G. and Sordoni, A. (2012). A Visual Tool for Bayesian Data Analysis: The Impact of Smoothing on Naïve Bayes Text Classifiers. In *SIGIR*, page In press, Portland, Oregon.

[Domingos and Pazzani, 1997] Domingos, P. and Pazzani, M. J. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130.

[Feinerer et al., 2008] Feinerer, I., Hornik, K., and Meyer, D. (2008). Text mining infrastructure in

r. *Journal of Statistical Software*, 25(5).

[Gelman et al., 2003] Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2003). *Bayesian Data Analysis, Second Edition (Chapman & Hall/CRC Texts in Statistical Science)*. Chapman and Hall/CRC, 2 edition.

[Hansen and Johnson, 2004] Hansen, C. D. and Johnson, C. R. (2004). *Visualization Handbook*. Academic Press, 1 edition.

[Harrell, 2006] Harrell, F. (2006). *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer Series in Statistics. Springer, second edition.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, second edition.

[Kohonen, 1995] Kohonen, T. (1995). *Self-Organizing Maps*. Springer Series in Information Retrieval. Springer, second edition.

[Lang et al., 2011] Lang, D. T., Swayne, D., Wickham, H., and Lawrence, M. (2011). *RGGobi: Interface between R and GGobi*. R package version 2.1.17.

[Lawrence, 2011] Lawrence, M. (2011). *cairoDevice: Cairo-based cross-platform antialiased graphics device driver.* R package version 2.19.

[Lawrence and Verzani, 2012] Lawrence, M. and Verzani, J. (2012). *gWidgetsRGtk2: Toolkit implementation of gWidgets for RGtk2*. R package version 0.0-81.

[Leban et al., 2006] Leban, G., Zupan, B., Vidmar, G., and Bratko, I. (2006). Vizrank: Data visualization guided by machine learning. *Data Min. Knowl. Discov.*, 13(2):119–136.

[Mozina et al., 2004] Mozina, M., Demsar, J., Kattan, M. W., and Zupan, B. (2004). Nomograms for visualization of naive bayesian classifier. In Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *PKDD*, volume 3202 of *Lecture Notes in Computer Science*, pages 337–348. Springer.

[Nigam et al., 1998] Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. M. (1998). Learning to classify text from labeled and unlabeled documents. In Mostow, J. and Rich, C., editors, *AAAI/IAAI*, pages 792–799. AAAI Press / The MIT Press.

[Poulet, 2008] Poulet, F. (2008). Towards effective visual data mining with cooperative approaches. In Simoff, S. J., Böhlen, M. H., and Mazeika, A., editors, *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, pages 389–406. Springer.

[Poulin et al., 2006] Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D. S., Fyshe, A., Pearcy, B., Macdonell, C., and Anvik, J. (2006). Visual explanation of evidence with additive classifiers. In *AAAI*, pages 1822–1829. AAAI Press.

[Rohrer et al., 1999] Rohrer, R. M., Sibert, J. L., and Ebert, D. S. (1999). A shape-based visual interface for text retrieval. *IEEE Computer Graphics and Applications*, 19(5):40–46.

[RStudio, 2011] RStudio (2011). *manipulate: Interactive Plots for RStudio*. R package version 0.96.316.

[Seifert and Lex, 2009] Seifert, C. and Lex, E. (2009). A novel visualization approach for data-mining-related classification. In Banissi, E., Stuart, L. J., Wyeld, T. G., Jern, M., Andrienko, G. L., Memon, N., Alhajj, R., Burkhard, R. A., Grinstein, G. G., Groth, D. P., Ursyn, A., Johansson, J., Forsell, C., Cvek, U., Trutschl, M., Marchese, F. T., Maple, C., Cowell, A. J., and Moere, A. V., editors, *IV*, pages 490–495. IEEE Computer Society.

[Verzani, 2012] Verzani, J. (2012). *gWidgets: gWidgets API for building toolkit-independent, interactive GUIs*. R package version 0.0-50.

[Wickham, 2009] Wickham, H. (2009). *ggplot2: elegant graphics for data analysis*. Springer New York.

[Wise et al., 1995] Wise, J. A., Thomas, J. J., Pennock, K., Lantrip, D., Pottier, M., Schur, A., and Crow, V. (1995). Visualizing the non-visual: spatial analysis and interaction with information from text documents. In Gershon, N. D. and Eick, S. G., editors, *INFOVIS*, pages 51–58. IEEE Computer Society.

[Wong, 1999] Wong, P. C. (1999). Guest editor's introduction: Visual data mining. *IEEE Computer Graphics and Applications*, 19(5):20–21.