1228 Probabilistic Graphical Models
H. Zhu and J. Boedecker
May 5, 2026

Bayesian Classifiers

Contents

1 Bayesian classifiers

1.1 Probabilistic classification

Given a set of samples X and a set of class labels Y whose entries are normally considered as nonnegative integers, *i.e.*, $\mathbf{dom}\, Y \subseteq \mathbf{Z}_+$. An 'ordinary' classifier is some function $f\colon X \to Y$ that assigns each sample x a class label $\hat{y}$. Probabilistic classifiers, instead of providing a deterministic prediction on class label, predict the posterior probability $\mathbf{P}(y \mid x)$ for all $y \in Y$ given $x \in X$, and these posterior probabilities satisfy

$$\sum_y \mathbf{P}(y \mid x) = 1,$$

for all $x \in X$. The similar 'hard' classification as in ordinary classifiers can then be calculated by

$$\hat{y} = \operatorname*{argmax}_y \mathbf{P}(y \mid x).$$

The formulation of the *Bayesian classifier* is based on the application of the Bayes rule to estimate the posterior probability of each class given the sample $x \in X$:

$$\mathbf{P}(y \mid x) = \frac{\mathbf{P}(x \mid y)\,\mathbf{P}(y)}{\mathbf{P}(x)}. \tag{1.1}$$

Note that the denominator $\mathbf{P}(x)$ is just a normalizing constant and is barely taken into account in practice. The probability $\mathbf{P}(y)$ is known as the *prior probability* of the class labels. Therefore, to estimate the posterior probability we only need to calculate the likelihood $\mathbf{P}(x \mid y)$ according to the given data. Assuming that x is represented as a n-dimensional vector $(x_1, \ldots, x_n)$, with each entry being a random variable X_i denoting the ith feature of sample x. Then according to the chain rule, the likelihood term can be calculated as

$$\begin{aligned}\mathbf{P}(x \mid y) &= \mathbf{P}(x_1, \ldots, x_n \mid y) \\ &= \mathbf{P}(x_1 \mid y)\,\mathbf{P}(x_2 \mid x_1, y) \cdots \mathbf{P}(x_n \mid x_{n-1}, \ldots, x_1, y).\end{aligned} \tag{1.2}$$

Calculating (1.2) can be computationally expensive when n is large since the number of parameters in the likelihood term increase exponentially with the dimension of x. This will not only imply a huge amount of memory to store all the parameters, but it will also be very difficult to estimate all the probabilities from the data. Thus, the Bayesian classifier can only be of practical use for relatively small problems in terms of the dimension of the feature vector of x. An alternative is to consider some independence properties as in graphical models, in particular that all features are independent given the class, resulting in the *naive Bayesian classifier*.

1.2 Naive Bayesian classifiers

The *naive Bayesian classifier* is based on the assumption that all the features of sample x are independent given the class variable. Under this assumption, (1.2) can be written as:

$$\mathbf{P}(x \mid y) = \mathbf{P}(x_1, \ldots, x_n \mid y) = \prod_{i=1}^{n} \mathbf{P}(x_i \mid y). \tag{1.3}$$

Substituting the likelihood term (1.3) under the naive Bayesian classifier assumption into (1.1), the posterior probability of class label y given sample x is

$$\mathbf{P}(y \mid x) = \frac{1}{Z}\,\mathbf{P}(y) \prod_{i=1}^{n} \mathbf{P}(x_i \mid y),$$

where Z is the normalization factor with

$$Z = \mathbf{P}(x) = \sum_{y} \mathbf{P}(y)\,\mathbf{P}(x \mid y),$$

so that the estimated posterior probabilities satisfy $\sum_y \mathbf{P}(y \mid x) = 1$.

The naive Bayes formulation drastically reduces the complexity of the Bayesian classifier, since in this case as the parameters for the model, we only require the prior probability of the class, given by

$$(\mathbf{P}(y_1), \mathbf{P}(y_2), \ldots), \tag{1.4}$$

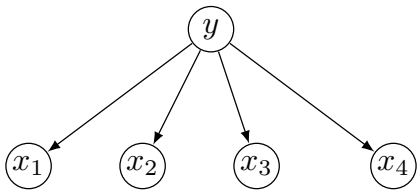

Figure 1 Example of a naive Bayesian classifier.

which is a one dimensional vector with the dimension equals to the cardinality of Y, and the n conditional probabilities of each feature of sample x given the class y, *i.e.*,

$$\begin{bmatrix} \begin{bmatrix} \mathbf{P}(x_1 \mid y_1) \\ \vdots \\ \mathbf{P}(x_n \mid y_1) \end{bmatrix} & \begin{bmatrix} \mathbf{P}(x_1 \mid y_2) \\ \vdots \\ \mathbf{P}(x_n \mid y_2) \end{bmatrix} & \cdots \end{bmatrix}, \tag{1.5}$$

which is a two dimensional matrix with the dimension of $n \times |Y|$. That is, the space requirement is reduced from exponential to linear in the number of features. A graphical representation of the naive Bayesian classifier is shown in figure 1. This tree-like structure depicts the property of conditional independence between all the features given the class, as there are no arcs between the feature nodes.

To learn the parameters of a naive Bayesian classifier, the prior probabilities (1.4) of the class variable Y can either be considered as uniformly distributed, *i.e.*,

$$\mathbf{P}(y_i) = \frac{1}{|Y|}, \quad \text{for all } y_i \in Y,$$

or be determined by estimating the class probability from the training data:

$$\mathbf{P}(y_i) = \frac{\#\text{ samples in class } y_i}{\#\text{ samples in total}}, \quad \text{for all } y_i \in Y.$$

If all features of each sample $x \in X$ are represented with discrete random variables, each entry of the likelihood matrix (1.5) can be directly learnt from the given data by calculating:

$$\mathbf{P}(x_k \mid y_i) = \frac{\#\text{ samples in class } y_i \text{ with feature } x_k}{\#\text{ samples in class } y_i},$$

for all $k = 1, \ldots, n$, $y_i \in Y$. As for handling continuous features, one trivial way would be using binning to discretize the feature values, so that the likelihood term can still be learnt using the above equations. However, the discretization may throw away discriminative information from the data. Another common technique for handling continuous values is to assume a parameterized distribution for the features from the training dataset, such as Gaussian distribution, multinomial distribution, or Bernoulli distribution, so that the learning of the likelihood matrix (1.5) can be transformed to learning the parameters of the assumed distribution.

Example. *Gaussian naive Bayes.* When dealing with continuous featured samples, a typical assumption is that within each class, the values of each continuous feature are Gaussian distributed, *i.e.*,

$$p(x_k \mid y_i) = \frac{1}{\sqrt{2\pi}\sigma_{k|y_i}} \exp\left(-\frac{(x_k - \mu_{k|y_i})^2}{2\sigma_{k|y_i}^2}\right), \quad k = 1, \ldots, n,$$

for all $y_i \in Y$, where $\mu_{k|y_i}$ and $\sigma_{k|y_i}^2$ are the mean and variance of the distribution of feature x_k within class y_i, respectively. Then to estimate the likelihood term required for naive Bayesian classifiers, we only need to learn $\mu_{k|y_i}$ and $\sigma_{k|y_i}$ from the training data according to

$$\mu_{k|y_i} = \mathbf{E}(X_k \mid y_i), \quad k = 1, \ldots, n,$$

$$\sigma_{k|y_i} = \sqrt{\mathbf{var}(X_k \mid y_i)} = \sqrt{\mathbf{E}\left((X_k - \mathbf{E}(X_k \mid y_i))^2 \,\middle|\, y_i\right)}, \quad k = 1, \ldots, n,$$

for all $y_i \in Y$.

In the inference stage, the density function of the assumed distribution evaluated at feature x_k for class y_i, *i.e.*, $p(x_k \mid y_i)$, is used to provide a relative estimation of each likelihood in matrix (1.5).

1.3 Augmented Bayesian classifiers

The general Bayesian classifier and the naive Bayesian classifier are the two extremes of possible dependency structures for Bayesian classifiers. The former represents the most complex structure with no independence assumptions, while the latter is the simplest structure that assumes that all the features are independent given the class. Between these two extremes there is a wide variety of possible models of varying complexities.

Example. *Tree augmented Bayesian classifiers.* The tree augmented Bayesian classifier incorporates some dependencies between the features of $x \in X$ by building a directed tree among the feature variables. That is, the n features form a graph which is restricted to a directed tree that represents the dependency relations between the features. Additionally there is an arc between the class labels and each feature. The structure of a tree augmented Bayesian classifier is depicted on the left of figure 2.

Example. *Bayesian network augmented Bayesian classifiers.* The Bayesian network augmented Bayesian classifier can be considered as an generalization of tree augmented Bayesian classifier by relaxing the constrain that the graph structure between features has to be a tree. In this case, it considers that the dependency structure among the features constitutes a directed acyclic graph. As with all aforementioned classifiers, there is a directed arc between the class node and each feature. The structure of a Bayesian network augmented Bayesian classifier is depicted on the right of figure 2.

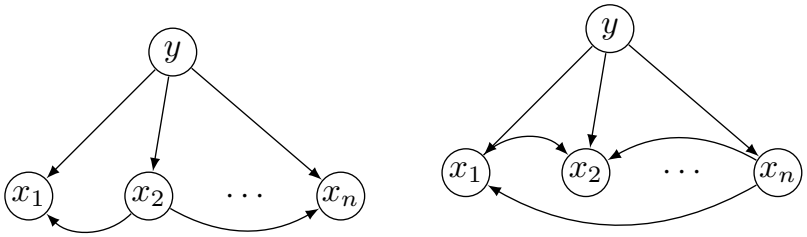

Figure 2 *Left.* Example of a tree augmented Bayesian classifier. *Right.* Example of a Bayesian network augmented Bayesian classifier.

The likelihood term (1.2) for the sample features x given the class label y can be obtained in a similar way as with the naive Bayesian classifiers. However, now each feature not only depends on the class but also on some other features according to the structure of the graph. Thus, we need to consider the conditional probability of each feature given the class and the feature of its parent nodes:

$$\mathbf{P}(x \mid y) = \mathbf{P}(x_1, \ldots, x_n \mid y) = \prod_{i=1}^{n} \mathbf{P}(x_i \mid \mathbf{pa}(x_i), y).$$

where $\mathbf{pa}(x_i)$ is the set of parent nodes of feature x_i according to the feature dependency structure of the classifier.

The tree and Bayesian network augmented Bayesian classifiers can be considered as particular cases of a more general model, that is, *Bayesian networks*, which will be covered later in the course with much more detail.

1.4 Semi-naive Bayesian classifiers

Another alternative to deal with dependent features is to transform the basic structure of a naive Bayesian classifier, while maintaining a tree-structured network. This has as an advantage that the efficiency and simplicity of the naive Bayesian classifier is maintained, and at the same time the performance is improved for cases where the features are not independent. These types of Bayesian classifiers are known as *semi-naive Bayesian classifiers*.

The basic idea of semi-naive Bayesian classifiers is to eliminate or join features which are not independent given the class label, such that the performance of the classifier improves. This is analogous to feature selection in machine learning, and there are two types of approaches:

- *Filter.* The features are selected according to a local measure, for instance the mutual information between the feature and the class.
- *Wrapper.* The features are selected based on a global measure, usually by comparing the performance of the classifier with and without the feature.

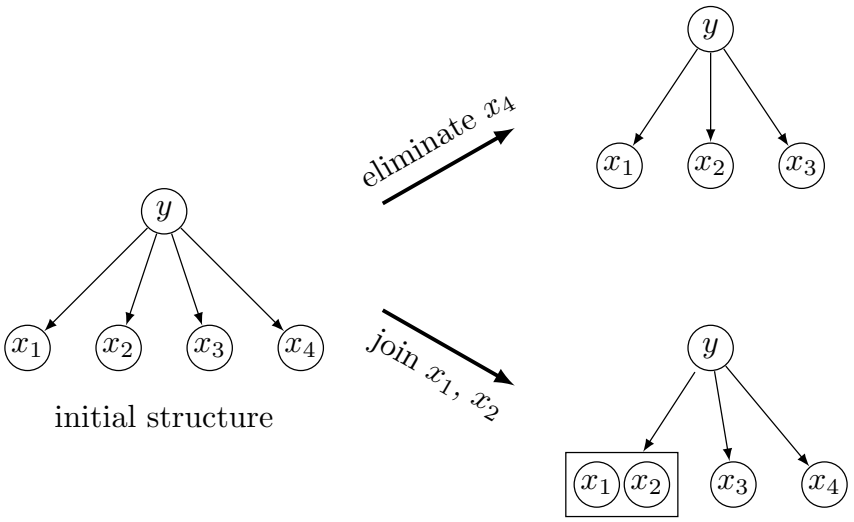

Figure 3 Example of structure improving process of a semi-naive Bayesian classifier.

Additionally, the learning algorithm can start from an empty structure and add features, or from a full structure with all the features, and eliminate (or combine) features. Figure 3 illustrates the two alternative operations to modify the structure of a naive Bayesian classifier starting from a full graph. Node elimination consists in simply eliminating a feature x_i from the graph, this could be because it is not relevant for the class (x_i and y are independent), or because the feature x_i and another feature x_j are not independent given the class. The rationale for eliminating one of the dependent features is that if the features are not independent given the class, one of them is redundant and could be eliminated. Node combination consists in merging two features x_i and x_j into a new feature, *e.g.*, $x_i x_j$, which is an alternative when two features are not independent given the class. By merging them into a single feature, the independence condition is not longer relevant. In principle we should select from the two alternatives that implies a higher improvement in the performance of the classifier and the feature selection process can be performed multiple iterations until there are no more superfluous or dependent features. Then the same parameter learning and inference process as in naive Bayesian classifiers can be directly applied.

2 Multi-dimensional classification

All previous classifiers consider that there is a single class variable; that is, each sample belongs to one and only one class. Several important problems need to predict several classes simultaneously, in which more than one class can be as-

signed to a sample $x \in X$. Such problems are called multi-dimensional classification problems. Similarly, an 'ordinary' multi-dimensional classifier consists in finding a vector-valued function $f\colon X \to Y$, which directly assigns to each sample $x \in X$ an m-dimensional vector of class values $\hat{y} \in Y$, with $\mathbf{dom}\, Y \subseteq \mathbf{Z}_+^m$. From a probabilistic point of view, the classifier first predicts the posterior probability $\mathbf{P}(y \mid x)$ for all $y \in Y$ given sample $x \in X$, then the 'hard' class labels $\hat{y}$ will be assigned according to $\hat{y} = \operatorname{argmax}_y \mathbf{P}(y \mid x)$.

In this chapter we only consider a particular case of multi-dimensional classification problems, named *multi-label classification*, where all class labels are binary, *i.e.*, $\mathbf{dom}\, Y_i = \{0, 1\}$, $i = 1, \ldots, m$. We will introduce the Bayesian network methods in subsequent chapters, which can be applied to solve the general multi-dimensional classification problems.

2.1 Basic approaches

There are two basic approaches for multi-label classification problems: *binary relevance* and *label power-set*. Binary relevance approaches transform the multi-label classification problem into m independent binary classification problems, one for each class $Y_1, \ldots, Y_m$. A classifier is independently learned for each class and the results are combined to determine the predicted class set, while the dependencies between classes are not considered. The label power-set approach transforms the multi-label classification problem into a single-class scenario by defining a new compound class variable whose possible values are all the possible combinations of values of the original classes. In this case the interactions between classes are implicitly considered. Essentially, binary relevance can be effective when the classes are relatively independent, and label power-set when there are few class variables.

2.2 Chain classifiers

Under the framework of Bayesian classifiers, we can consider an alternative to the binary relevance approach, named the *chain classifiers*, which implicitly incorporate the dependencies between classes by adding additional features to each independent classifier. A chain classifier consists of m base binary classifiers $(f_1, \ldots, f_m)$ which are linked in a chain, so that each classifier incorporates the predicted classes $\hat{y}_i \in \{0, 1\}$, $i = 1, \ldots, m$, by the previous classifiers as additional features. Thus, each classifier f_i in the chain is trained to learn the association of label $\hat{y}_i$ given the features augmented with all previous predicted class labels $(\hat{y}_1, \ldots, \hat{y}_{i-1})$ in the chain. Formally, this process can be denoted as

$$\begin{aligned} \hat{y}_1 &= \operatorname{argmax}_{y_1} \mathbf{P}(y_1 \mid x), \\ \hat{y}_i &= \operatorname{argmax}_{y_i} \mathbf{P}(y_i \mid x, \hat{y}_1, \ldots, \hat{y}_{i-1}), \quad i = 2, \ldots, m. \end{aligned} \tag{2.1}$$

As in the binary relevance approach, the final prediction of the class vector is determined by concatenating the outputs of all the binary classifiers in the chain,

$$\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m).$$

A challenge for chain classifiers is to select the order of the classes in the chain, since the order can affect the performance of the classifier. We describe two approaches to address this challenge as follows.

2.2.1 Circular chain classifiers

In a *circular chain classifier*, the propagation of the predicted classes from the previous binary classifiers is done iteratively in a circular way. In the first cycle, as in the standard chain classifiers (2.1), the predictions of the previous classifiers are additional features for each classifier in the chain. After the first cycle, each binary classifier in the chain receives the predictions of all other classifiers as additional feature, and update its prediction according to

$$\hat{y}_i = \underset{y_i}{\operatorname{argmax}} \mathbf{P}(y_i \mid x, \hat{y}_{-i}), \quad i = 1, \ldots, m,$$

where $\hat{y}_{-i}$ denotes the predicted class vector $\hat{y}$ with the ith entry removed. This process is repeated for a prefixed number of cycles or until convergence.

2.2.2 Bayesian chain classifiers

Bayesian chain classifiers consider the following two assumptions. Firstly, the dependency of different classes given the features can be represented as a directed acyclic graph. Then the posterior probability of class vector y given sample $x \in X$ reduces to

$$\mathbf{P}(y \mid x) = \mathbf{P}(y_1, \ldots, y_m \mid x) = \prod_{i=1}^{m} \mathbf{P}(y_i \mid \mathbf{pa}(y_i), x).$$

Secondly, since the calculation of the final prediction $\hat{y}$ involves solving a hard combinatorial optimization problem on variable y,

$$\text{maximize} \quad \textstyle\prod_{i=1}^{m} \mathbf{P}(y_i \mid \mathbf{pa}(y_i), x),$$

Bayesian chain classifiers assume an approximation on this problem that it can be solved approximately via a sequence of independent optimization problems, each focusing on maximizing the conditional probability of a single variable y_i:

$$\text{maximize} \quad \mathbf{P}(y_i \mid \mathbf{pa}(y_i), x), \tag{2.2}$$

for all $i = 1, \ldots, m$. That is, the most probable joint combination of class assignments is approximated by the concatenation of the most probable individual classes. The final output class vector $\hat{y}$ of the Bayesian chain classifier will be a simple concatenation of all outputs $\hat{y}_i$ from the ith run of (2.2).

The first assumption is reasonable if we have enough data to obtain a good approximation of the class dependency structure, and assuming that this is obtained conditioned on the features. Regarding the second assumption, the total abduction or most probable explanation is not always equivalent to the maximization of the individual classes. However, the assumption is less strong than that assumed

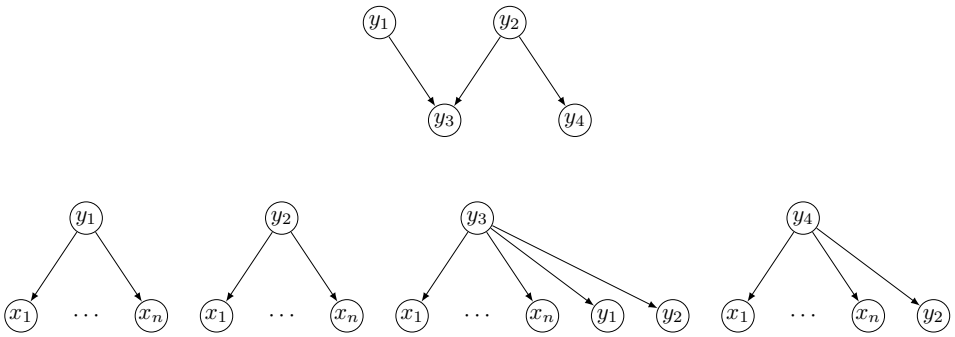

Figure 4 Example of a Bayesian chain classifier. *Top.* The graph representing the class dependency structure. *Bottom.* Naive Bayesian classifiers, one for each class.

by the binary relevance approach. Bayesian chain classifiers provide an attractive alternative to multi-dimensional classification, as they incorporate in certain ways the dependencies between class variables, and they keep the efficiency of the binary relevance approach. For the base classifier that belongs to each class we can use any of the Bayesian classifiers presented in the previous sections, for instance a naive Bayesian classifier. The general idea for building a Bayesian chain classifier is illustrated in figure 4.

Bibliography

This chapter is adapted from [Suc21, §4]. This book also provides some introduction on solving hierarchical classification problems with Bayesian classifiers in §4.7, which is not included in our discussion.

The book by Michie *et al.* [MSTC95] can be referred to for a general introduction and comparison between different classification approaches.

Some discussion about the lost of discriminative information when using binning to discretize the continuous feature values for naive Bayesian classifiers are provided in [HY01].

In Gaussian naive Bayes, sometimes the distribution of class-conditional marginal densities for continuous features is far from normal. In these cases, kernel density estimation can be used for a more realistic estimate of the marginal densities of each class. This method, which was introduced by John and Langley [JL13] can boost the accuracy of the classifier considerably.

Except the Gaussian naive Bayes introduced in this chapter, there are many other useful extensions of naive Bayesian classifiers to continuous feature spaces, such as Multinomial naive Bayes [RSTK03] and Bernoulli naive Bayes [MN98].

A more detailed description about variants of augmented Bayesian classifiers can be found in [FGG97].

The semi-naive Bayesian classifier was initially introduced in [MS06], and later extended by Pazzani [Paz95]. The paper [MS06] can be referred to for more detail about the feature selection process of a semi-naive Bayesian classifier.

The paper [TK07] provides an overview on multi-label classification problems, where some additional discussions about the label power-set approach can be found.

Chain classifiers for multi-label classification problems were initially proposed in [RPHF11], and were extended to a Bayesian network architecture by Sucar *et al.* [SBM+14]. Rivas *et al.* [ROS18] did some convergence analysis on circular chain classifiers and it has been shown empirically that circular chain classifiers tend to converge in a few iterations and that the order of the classes in the chain does not affect the performance according to several metrics.

References

[FGG97] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[HY01] D. J. Hand and K. Yu. Idiot's Bayes — Not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

[JL13] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. *arXiv*, 1302.4964, 2013.

[MN98] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, volume 752, pages 41–48. Association for Computational Linguistics, 1998.

[MS06] M. Martinez-Arroyo and L. E. Sucar. Learning an optimal naive Bayes classifier. In *18th International Conference on Pattern Recognition*, volume 3, pages 1236–1239. IEEE, 2006.

[MSTC95] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1995.

[Paz95] M. Pazzani. Searching for attribute dependencies in Bayesian classifiers. In *5th International Workshop on Artificial Intelligence and Statistics*, pages 424–429. PMLR, 1995.

[ROS18] J. J. Rivas, F. Orihuela-Espina, and L. E. Sucar. Circular chain classifiers. In *International Conference on Probabilistic Graphical Models*, pages 380–391. PMLR, 2018.

[RPHF11] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85:333–359, 2011.

[RSTK03] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger. Tackling the poor assumptions of naive Bayes text classifiers. In *Proceedings of the 20th International Conference on Machine Learning*, pages 616–623. AAAI Press, 2003.

[SBM+14] L. E. Sucar, C. Bielza, E. F. Morales, P. Hernandez-Leal, J. H. Zaragoza, and P. Larrañaga. Multi-label classification with Bayesian network-based chain classifiers. *Pattern Recognition Letters*, 41:14–22, 2014.

[Suc21] L. E. Sucar. *Probabilistic Graphical Models: Principles and Applications*. Springer, 2nd edition, 2021.

[TK07] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.