Charu C. Aggarwal

Machine Learning for Text

5.2.6 Embedded Feature Selection Models

Many classification and regression models provide the ability to perform embedded feature selection by leveraging the output of intermediate steps. Feature selection is accomplished with the use of *regularization* in order to reduce overfitting, which is similar in principle to the goals of feature selection. As a result, the intermediate outputs of these regularized algorithms provide useful insights for feature selection. For example, consider the following linear regression model (see Sect. 6.2.2 of Chap. 6), in which the numerical dependent variable y_i is predicted using the following linear relationship to the feature variables $\overline{X_i}$:

$$y_i \approx \overline{W} \cdot \overline{X_i} \quad \forall i \in \{1 \ldots n\} \tag{5.10}$$

The notation $\overline{W}$ represents a d-dimensional vector of coefficients that is learned by the training model. This vector is computed by solving the following optimization model:

$$\text{Minimize} \underbrace{\sum_{i=1}^{n} (\overline{W} \cdot \overline{X_i} - y_i)^2}_{\text{Prediction Error}} + \underbrace{\lambda \sum_{i=1}^{d} |w_i|}_{\text{Penalty for using features}}$$

Here, $\lambda > 0$ is a regularization parameter, which controls the severity of the penalty. Such a penalty ensures that the optimization will not assign a large non-zero coefficient for that feature, unless the feature conveys important and irreplaceable information about the dependent variable. Feature penalization is referred to as regularization. The type of penalty discussed above is referred to as the L_1-penalty, and it has the remarkable property of favoring a coefficient vector $\overline{W}$ in which many values of w_i are zero. Such features are effectively dropped because they will have no influence on prediction of test instances according to Eq. 5.10. The natural idea in embedded feature selection is that it leverages on built-in (regularization) mechanisms by many algorithms to avoid overfitting. After all, the main goal of feature selection is also the prevention of overfitting. A detailed discussion of L_1-regularization is provided in Sect. 6.2.2 of Chap. 6.

5.2.7 Feature Engineering Tricks

Two types of feature engineering tricks are commonly used in the text domain. The first trick is done to get rid of sparsity, which can be a problem for some classifiers such as decision trees. The second technique uses representation mining techniques to embed sequential representations of text to multidimensional representations. The latter approach is able to leverage the sequential ordering information among words to incorporate greater semantic knowledge in learning. Since the second approach will be discussed in Chap. 10, the following will discuss only the feature engineering methods used to address sparsity.

Sparsity can cause challenges with certain types of classifiers like decision trees, which use attributes *one at a time* in the modeling process. Since each term contains information relevant to only a small subset of documents in which it is present, and the absence of terms is noisy information, it often causes overfitting when classifiers make important decisions with individual attributes. Therefore, in such cases, methods like latent semantic analysis (LSA) are not just useful for dimensionality reduction, but they can be viewed as feature engineering methods that enable the use of certain types of classifiers. A particular variant of LSA, known as a *Rotation Ensemble* is particularly useful for ensemble-centric implementations. The basic idea is to use the following approach:

Randomly split the d terms into K disjoint subsets of size d/K to create K projected data sets;
Perform LSA on *each* projected data set to extract $r \ll d/K$ features;
Pool all extracted features to create a $(K \cdot r)$-dimensional data set;
Apply a classifier on the new representation;

This approach can be applied multiple times, and the prediction of a test instance can be averaged over multiple such transformations. A particularly common classifier that is used with this approach is the *decision tree*, and the resulting classifier is referred to as the *Rotation Forest* [413]. Another feature engineering method is the Fisher's linear discriminant (cf. Sect. 6.2.3 of Chap. 6), which provides *discriminative* directions in the space. Such methods have also been used in conjunction with decision trees [82].

5.3 The Naïve Bayes Model

The naïve Bayes classifier uses a probabilistic generative model that is identical to the mixture model used for clustering (cf. Sect. 4.4 of Chap. 4). The model assumes that the corpus is generated from a mixture of different classes. The generative process, which is applied once for each observed document, is as follows:

1. Select the rth class (mixture component) $\mathcal{C}_r$ with prior probability $\alpha_r = P(\mathcal{C}_r)$.

2. Generate the next document from the probability distribution for $\mathcal{C}_r$. The most common choices are the Bernoulli and multinomial distributions.

The observed (training and test) data are assumed to be outcomes of this generative process, and the parameters of this generating process are estimated so that the log-likelihood of this data set being created by the generative process is maximized. Generally, only the training data is used to estimate the parameters, because the training data contains additional information about the identity of the mixture component that generated each document. Subsequently, these parameters are used to estimate the probability of the generation of each unlabeled test document from each mixture component (class). This results in a probabilistic classification of unlabeled documents.

Each cluster $\mathcal{G}_r$ in the expectation-maximization algorithm of Sect. 4.4 is analogous to a class $\mathcal{C}_r$ in this setting. One can view naïve Bayes as a simplification of the iterative expectation-maximization algorithm in which the presence of labels allows the execution of the approach in a single iteration. Unlike clustering, the training process in classification uses *a single* application of the M-step (on labeled data), and the probabilistic prediction of test instances is a single application of the E-step on the unlabeled test instances (to estimate posterior probabilities). Furthermore, the naïve Bayes classifier has analogous Bernoulli and multinomial models to those used in clustering.

5.3.1 The Bernoulli Model

In the Bernoulli model, it is assumed that only the presence or absence of each term in the document is observed. Therefore, the frequencies of the terms are ignored, and the vector-space representation of a document is a sparse binary vector. The Bernoulli model assumes that the jth term, t_j, in the lexicon is present in a document generated from the rth class (mixture component) with probability $p_j^{(r)}$. Then, the probability $P(\overline{Z}|\mathcal{C}_r)$ of the generation

of the document $\overline{Z}$ from mixture component $\mathcal{C}_r$ is given[2] by the product of the d different Bernoulli probabilities corresponding to presence of absence of various terms:

$$P(\overline{Z}|\mathcal{C}_r) = \prod_{t_j \in \overline{Z}} p_j^{(r)} \prod_{t_j \notin \overline{Z}} (1 - p_j^{(r)}) \tag{5.11}$$

An important assumption here is that the presence or absence of the various terms are conditionally independent with respect to the choice of class. Therefore, one can express the joint probability of the attributes in $\overline{Z}$ as the product of the corresponding values on individual attributes. This assumption is also referred to as the *naïve Bayes assumption*, which is also the reason that the method is referred to as a *naïve Bayes classifier*. The term "naïve" is used because this type of approximation is generally not true in real settings.

The main task in the *training phase* of the Bayes classifier is to estimate the (maximum likelihood) values of the prior probabilities α_r and class-specific generative probabilities $p_j^{(r)}$. These parameters are estimated so that the observed data has the maximum likelihood of being generated by the model, and are then used to perform the prediction of the labels of unseen test instances. One can summarize this process as follows:

- **Training phase:** Estimate the maximum-likelihood values of the parameters $p_j^{(r)}$ and α_r using only the training data.
- **Prediction phase:** Use the estimated values of the parameters to predict the class of each unlabeled test instance.

The training phase is executed first, which is followed by the prediction phase. However, since the prediction phase of a naïve Bayes classifier is the key to understanding it, we will present the prediction phase before the training phase. Therefore, the following section will assume that the model parameters have already been learned in the training phase.

5.3.1.1 Prediction Phase

The prediction phase uses the Bayes rule of posterior probabilities to predict an instance. The basic idea is that the learner uses the aggregate frequency of each class in the training data to learn a *prior* probability $\alpha_r = P(\mathcal{C}_r)$, of each class. Subsequently, it needs to estimate the *posterior* probability $P(\mathcal{C}_r|\overline{Z})$ after observing a *specific* document (with binary representation $\overline{Z} = (z_1 \ldots z_d)$) for which the label is not known. This estimation provides a probabilistic prediction for the test instance $\overline{Z}$ of belonging to a particular class.

According to the Bayes rule of posterior probabilities, the posterior probability of $\overline{Z}$ being generated by the mixture component $\mathcal{C}_r$ of the rth class can be estimated as follows:

$$P(\mathcal{C}_r|\overline{Z}) = \frac{P(\mathcal{C}_r) \cdot P(\overline{Z}|\mathcal{C}_r)}{P(\overline{Z})} \propto P(\mathcal{C}_r) \cdot P(\overline{Z}|\mathcal{C}_r) \tag{5.12}$$

A constant of proportionality[3] is used instead of the $P(\overline{Z})$ in the denominator, because the estimated probability is only compared between multiple classes to determine the predicted class, and $P(\overline{Z})$ is independent of the class.

[2] Although $\overline{X_i}$ is a binary vector, we are treating it like a set when we use a set-membership notation like $t_j \in \overline{X_i}$. Any binary vector can also be viewed as a set of the 1s in it.

[3] The constant of proportionality can be easily inferred by ensuring that the sum of the posterior probabilities across all classes is 1. As we will see later, there are scenarios associated with ranking instances to belong to specific classes, where the constant of proportionality does matter.

An important observation here is that all the parameters on the right-hand side of the conditional can be estimated using the Bernoulli model. We further expand the relationship in Eq. 5.12 using the Bernoulli distribution of Eq. 5.11 as follows:

$$P(\mathcal{C}_r|\overline{Z}) \propto P(\mathcal{C}_r) \cdot P(\overline{Z}|\mathcal{C}_r) = \alpha_r \prod_{t_j \in \overline{Z}} p_j^{(r)} \prod_{t_j \notin \overline{Z}} (1 - p_j^{(r)}) \tag{5.13}$$

Note that all the parameters on the right-hand side are estimated during the training phase discussed below. Therefore, one now has an estimated probability of each class being predicted up to a constant factor of proportionality. The class with the highest posterior probability is predicted as the relevant one, although the output is sometimes provided in the form of probabilities. It is noteworthy that this step is identical to the E-step used for mixture modeling in clustering (cf. Sect. 4.4.1), except that it is applied only to the unlabeled test instances.

5.3.1.2 Training Phase

The training phase of the Bayes classifier uses the labeled training data to estimate the maximum likelihood values of the parameters in Eq. 5.13. It is evident that we need to estimate two sets of parameters, which are the prior probabilities α_r and the Bernoulli generative parameters, $p_j^{(r)}$, for each mixture component. The statistics available for parameter estimation include the number of labeled documents n_r belonging to the rth class $\mathcal{C}_r$, and the number, $m_j^{(r)}$, of the documents belonging to class $\mathcal{C}_r$ that contain term t_j. The maximum likelihood estimates of these parameters can be shown to be the following:

1. *Estimation of prior probabilities:* Since the training data contains n_r documents for the rth class in a corpus size of n, the natural estimate for the prior probability of the class is as follows:

$$\alpha_r = \frac{n_r}{n} \tag{5.14}$$

   If the corpus size is small, it is helpful to perform Laplacian smoothing by adding a small value $\beta > 0$ to the numerator and $\beta \cdot k$ to the denominator:

$$\alpha_r = \frac{n_r + \beta}{n + k \cdot \beta} \tag{5.15}$$

   The precise value of β contains the amount of smoothing, and it is often set to 1 in practice. When the amount of data is very small, this results in the prior probabilities being estimated closer to $1/k$, which is a sensible assumption in the absence of sufficient data.

2. *Estimation of class-conditioned mixture parameters:* The class-conditioned mixture parameters, $p_j^{(r)}$, are estimated as follows:

$$p_j^{(r)} = \frac{m_j^{(r)}}{n_r} \tag{5.16}$$

   It is particularly important to use Laplacian smoothing on the class-conditioned probabilities because a particular term t_j might not even be present in the training documents of the rth class, particularly when the corpus is small. In such a case, one would estimate the corresponding value of $p_j^{(r)}$ to 0. As a result of the multiplicative

nature of Eq. 5.13, the presence of term t_j in an unseen document will always lead to an estimated probability of 0 for the rth class. Such predictions are often erroneous, and are caused by overfitting to the small training data size.

Laplacian smoothing of class-conditioned probability estimation is performed as follows. Let d_a be the average number of 1s in the binary representation of each training document and d be the size of the lexicon. The basic idea is to add a Laplacian smoothing parameter $\gamma > 0$ to the numerator of Eq. 5.16 and $d\,\gamma/d_a$ to the denominator:

$$p_j^{(r)} = \frac{m_j^{(r)} + \gamma}{n_r + d\,\gamma/d_a} \tag{5.17}$$

The value of γ is often set to 1 in practice. When the amount of training data is very small, this choice leads to a default value of d_a/d for $p_j^{(r)}$, which reflects the level of sparsity in the document collection.

It is noteworthy that the training phase in the Bayes classifier is a simplified variant of the M-step used in the mixture model for clustering (cf. Sect. 4.4.1). This simplification is because *labeled* training data is available to infer the membership of documents in mixture components.

5.3.2 Multinomial Model

While the Bernoulli model uses only the presence of absence of terms in documents, the multinomial model explicitly uses their term frequencies. Just as the parameter $p_j^{(r)}$ in the Bernoulli model denotes the probability whether a term is observed in a particular component, the parameter q_{jr} in the multinomial model denotes the fractional presence of term t_j in the rth mixture component, including the effect of repetitions. The values of q_{jr} sum to 1 for a particular mixture component r over all terms (i.e., $\sum_{j=1}^{d} q_{jr} = 1$).

The generative process for the multinomial mixture model first selects the rth class (mixture component) with probability $\alpha_r = P(\mathcal{C}_r)$. Then, it throws a loaded die (owned by the rth class) L times to generate a document with L *tokens* (counting repetitions). The loaded die has as many faces as the number of terms d, and the probability of the jth face showing up is given by q_{jr} for the die owned by the rth class. Therefore, if the die is thrown L times, then the number of times each face shows up provides the number of times each term shows up in the observed document. If we assume that the frequency vector of the document $\overline{Z}$ is given by $(z_1 \ldots z_d)$, then the generative probability of the ith document is given by the following multinomial distribution:

$$P(\overline{Z}|\mathcal{C}_r) = \frac{(\sum_{j=1}^{d} z_j)!}{z_1!z_2!\ldots z_d!} \prod_{j=1}^{d} (q_{jr})^{z_j} \propto \prod_{j=1}^{d} (q_{jr})^{z_j} \tag{5.18}$$

The constant of proportionality holds for fixed $\overline{Z}$ and varying class, because it depends only on $\overline{Z}$ and is independent of the class $\mathcal{C}_r$.

The overall process of both prediction and training in the multinomial model is very similar to that of the Bernoulli model. As in the case of the Bernoulli model, one can use the Bayes rule and Eq. 5.18 to derive the following values for the estimated posterior probability that the test instance $\overline{Z}$ belongs to class $\mathcal{C}_r$:

$$P(\mathcal{C}_r|\overline{Z}) \propto P(\mathcal{C}_r) \cdot P(\overline{Z}|\mathcal{C}_r) \propto \alpha_r \prod_{j=1}^{d} (q_{jr})^{z_j} \tag{5.19}$$

If needed, the constant of proportionality can be inferred by ensuring the posterior probabilities over all classes sum to 1. The class with the largest posterior probability can be predicted as the relevant one for the test instance $\overline{Z}$.

In order to compute the values on the right-hand side of Eq. 5.19, one only needs to estimate the parameters α_r and q_{jr} during the training phase. The fractional presence of each class in the training data is used as the estimate of α_r. Laplacian smoothing can be used if needed. Furthermore, if $\nu(j, r)$ is the number of times that the term t_j shows up in the documents belonging to class r (with proportionate credit given to repetitions in a single document), then the estimate q_{jr} can be computed as follows:

$$q_{jr} = \frac{\nu(j, r)}{\sum_{j=1}^{d} \nu(j, r)} \tag{5.20}$$

One can also view this estimate as the fraction of the number of *tokens* (i.e., positions) in a class that correspond to a particular term. This is different from the Bernoulli model that estimates the class-conditioned probabilities as the fraction of class-specific documents containing a particular term. It is also possible to use Laplacian smoothing in order to smooth the estimation. In this case, we add a small value $\gamma > 0$ to the numerator, and $\gamma \cdot d$ to the denominator. This results in the following estimation:

$$q_{jr} = \frac{\nu(j, r) + \gamma}{\sum_{j=1}^{d} \nu(j, r) + \gamma \cdot d} \tag{5.21}$$

It is common to set γ to 1. This type of smoothing biases the estimation of the probability of each of the d faces in the multinomial die roll towards $1/d$, which implies that all terms are equally favored. This is a reasonable assumption in the absence of sufficient data.

5.3.3 Practical Observations

The naïve assumption of conditional independence is never really true in practical settings. In spite of this fact, the actual predictions are surprisingly robust. Using more complicated assumptions often end up overfitting the data. Several insights are provided in [140] about why the naïve assumption works so well in practice.

A natural question arises as to when it is preferable to use either the Bernoulli or the multinomial models. Note that the Bernoulli model uses both the presence and the absence of terms in a document, but it does not use the term frequencies. The two main factors are (1) the typical length of each document and, (2) the size of the lexicon from which the terms are drawn. For short documents that have a non-sparse representation with respect to a small lexicon, it makes sense to use the Bernoulli model. In short documents, there are a limited number of repetitions of terms, which reduces the gain obtained from including frequency information. Furthermore, if the lexicon size is very small and the vector-space representation is non-sparse, then even the absence of a term in a document is informative. When the document representation is sparse, information about absence of terms is noisy, which hurts the Bernoulli model. Furthermore, the ignoring of frequency information will also increase the inaccuracy of the Bernoulli model. Therefore, it makes sense to use the multinomial model in such cases.

5.3.4 Ranking Outputs with Naïve Bayes

The prediction problem of classification is not always posed in terms of selecting the class of a single test instance. In many cases, a set of test instances $\overline{Z_1} \ldots \overline{Z_{n_t}}$ is provided, and