CS838-1 Advanced NLP: Text Categorization with Naive Bayes Classifiers

Xiaojin Zhu

2007
Send comments to jerryzhu@cs.wisc.edu

Say you want to file your emails into two folders: "study", "fun" (not that they should be different!). And you want to do it *automatically.* How?

1 Machine Learning Basics

- example, item, point, instance: each input object (e.g., a document, an image, a person).
- feature $x \in \mathcal{X}$: usually a fixed-dimensional numerical vector that characterizes the instance. For example, for a document x can be the word count vector. $\mathcal{X}$ is the feature space.
- label $y \in \mathcal{Y}$: numerical encoding of output. It can be a discrete number (e.g. -1, 1 or 0, 1) for binary classification, $1, \ldots, K$ for multiclass classification, a real number for regression.
- classifier: $f : \mathcal{X} \to \mathcal{Y}$ when $\mathcal{Y}$ encodes discrete classes. This also implies that the particular class encoding is not important.
- training set: $\{(x_1, y_1), \ldots, (x_n, y_n)\} \sim p(x, y)$. We assume the training examples are drawn i.i.d. from an unknown but fixed joint probability $p(x, y)$.
- training error (rate): $\frac{1}{n}\sum_{i=1}^{n}[f(x_i) \neq y_i]$.
- generalization error: $\mathbb{E}_p[f(x) \neq y]$. *Goal of machine learning:* Given training set, find f to minimize generalization error. This is difficult because we assume $p(x, y)$ is unknown.
- test set error: $\frac{1}{m}\sum_{i=n+1}^{n+m}[f(x_i) \neq y_i]$. The examples are again drawn i.i.d. from $p(x, y)$. Test set error is an estimate of generalization error.
- Overfitting: A classifier trained to minimize training error will often perform poorly on test set. This is known as overfitting. However, it is not

a good idea to *tune parameters to minimize test set error* either, because it is essentially overfitting the test set, and the true generalization error will be higher. Tuning on test set in most cases is regarded as *cheating* in machine learning.

- Tuning set: One may randomly split the training data into two parts: a smaller 'training set' (say 70%) and a 'tuning set' (say 30%). One trains a classifier from the training set, but tunes any parameters to minimize the *tuning set error* instead of training set error. The best model's performance is then measure on the test set. This is a valid procedure because the test set is not used to select a classifier.

- k-fold cross validation: Sometimes one only has the training set $\{(x, y)_{1:n}\}$, but not a separate test set. One can simulate test set error as follows: Randomly split the training set into k equal *folds*. First, use folds 1, ..., $k-1$ to train a classifier, and treat fold k as the test set to compute error e_k. Second, use folds 1, ..., $k-2$, k to train a different classifier, and treat fold $k-1$ as the test set to compute error e_{k-1}. Repeat the procedure for all k folds. Finally the *k-fold cross validation error* is the average of $e_1, \ldots, e_k$. In order not to waste data, the final classifier is trained on the complete training set. When $k = n$, this is known as leave-one-out cross validation.

2 Naive Bayes Classifier

Let each document be represented by $x = (c_1, \ldots, c_v)^\top$ the word count vector, otherwise known as *bag of word* representation. We assume within each class y, the probability of a document follows the multinomial distribution with parameter θ_y:

$$p(x|y) \propto \prod_{w=1}^{v} \theta_{yw}^{c_w}. \tag{1}$$

The log likelihood is

$$\log p(x|y) = x^\top \log \theta_y + const. \tag{2}$$

Note different classes have different θ_y's. Also note that the multinomial distribution assume *conditional independence* of feature dimensions $1, \ldots, v$ given the class y. We know this is not true in reality, and more sophisticated models would assume otherwise. For this reason, such assumption on independence of features is known as the *naïve Bayes* assumption[1].

[1]Whether you put two dots above i is a matter of personal taste.

Classification is done via Bayes rule:

$$\begin{aligned}
y^* &= \arg\max_y p(y|x) && (3)\\
&= \arg\max_y \frac{p(x|y)p(y)}{p(x)} && (4)\\
&= \arg\max_y p(x|y)p(y) && (5)\\
&= \arg\max_y x^\top \log\theta_y + \log p(y), && (6)
\end{aligned}$$

where $p(y)$ is often estimated from the frequency of class y in training data. In this process we computed $p(y|x)$[2], and we assumed that the parameters θ_y are known for all classes. The process of computing the marginal distribution of unknown variable (y) given observed variables (x) is called *inference*.

Given a training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, *training* or *parameter learning* involves finding the best parameters $\Theta = \{\pi, \theta_1, \ldots, \theta_C\}$. The model is $p(y = j) = \pi_j$, and $p(x|y = j) = \text{Mult}(x; \theta_j) \propto \prod_{w=1}^{V} \theta_{jw}^{x_w}$. For simplicity we use the MLE here, but MAP is common too. We maximize the joint (log) likelihood of the training set:

$$\begin{aligned}
\ell &= \log p((x,y)_{1:n}|\Theta) \text{ ; hide } \Theta \text{ below} && (7)\\
&= \log \prod_{i=1}^{n} p(x_i, y_i) && (8)\\
&= \sum_{i=1}^{n} \log p(x_i, y_i) && (9)\\
&= \sum_{i=1}^{n} \log p(y_i) + \log p(x_i|y_i). && (10)
\end{aligned}$$

We can formulate this as a constrained optimization problem,

$$\begin{aligned}
&\max_\Theta && \ell && (11)\\
&\text{s.t.} && \textstyle\sum_{j=1}^{C} \pi_j = 1, \; C \text{ is the number of classes} && (12)\\
& && \textstyle\sum_{w=1}^{v} \theta_{jw} = 1, \forall j = 1 \ldots C. && (13)
\end{aligned}$$

It is easy to solve it using Lagrange multipliers and arrive at

$$\begin{aligned}
\pi_j &= \frac{\sum_{i=1}^{n} [y_i = j]}{n} && (14)\\
\theta_{jw} &= \frac{\sum_{i:y_i=j} x_{iw}}{\sum_{i:y_i=j} \sum_{u=1}^{V} x_{iu}}. && (15)
\end{aligned}$$

[2]We did not normalize it, but normalization could be done if desired.

2.1 Naive Bayes as a Generative Model

A generative model is a probabilistic model which describe the full generation process of the data, or the joint probability $p(x, y)$. Our Naive Bayes model consists of $p(y)$ and $p(x|y)$, which do just that: One can generate data (x, y) by first sample $y \sim p(y)$, and then sample word counts from the multinomial $p(x|y)$.

There is another family of models known as discriminative models, which do not model $p(x, y)$. Instead, they focuses on the conditional $p(y|x)$, or a similar but non-probabilistic quantity, which is directly related to classification. We will see our first discriminative model when we discuss logistic regression.

2.2 Naive Bayes as a Linear Classifier

Consider binary classification where $y = 0$ or 1. Our classification rule with arg max can equivalently be expressed with log odds ratio

$$\begin{aligned} f(x) &= \log \frac{p(y=1|x)}{p(y=0|x)} && (16) \\ &= \log p(y=1|x) - \log p(y=0|x) && (17) \\ &= (\log \theta_1 - \log \theta_0)^\top x + (\log p(y=1) - \log p(y=0)). && (18) \end{aligned}$$

The decision rule is to classify x with $y = 1$ if $f(x) > 0$, and $y = 0$ otherwise. Note for given parameters, this is a *linear function* in x. That is to say, the Naive Bayes classifier induces a linear decision boundary in feature space $\mathcal{X}$. The boundary takes the form of a hyperplane.

2.3 Naive Bayes as a Special Case of Bayes Networks

A *Bayes Network* is a directed graph that represent a family of probability distributions. This is covered in detail in [cB] Chapter 8.1, 8.2. Outline:

- nodes: each node is a random variable. We have one y node, and v x_w nodes.
- directed edges: No directed cycles allowed, i.e. must be a DAG. For naive Bayes, from y to x_w.
- meaning: the joint probability on all nodes $s_{1:K}$ is factorized in a particularly form
$$p(s) = \prod_{i=1}^{K} p(s_i|\mathrm{pa}(s_i)), \quad (19)$$
where $\mathrm{pa}(s_i)$ are the parents of s_i. For naive Bayes, $p(x_{1:v}, y) = p(y) \prod_{i=1}^{v} p(x_i|y)$.
- observed nodes: nodes with known values, e.g. $x_{1:v}$. Shaded.
- plate: a lazy way to duplicate the node (and associated edges) multiple times. Our $x_{1:v}$ can be condensed into a plate.