CS838-1 Advanced NLP: Text Categorization with Logistic Regression

Xiaojin Zhu

2007
Send comments to jerryzhu@cs.wisc.edu

Naive Bayes is a *generative model*. It models the joint $p(x, y)$. However, if our ultimate goal is classification, the relevant part is $p(y|x)$. In Naive Bayes this is computed via Bayes rule. One might wonder whether it is possible to estimate $p(y|x)$ directly. A model that estimates $p(y|x)$ directly is known as a *discriminative model*. Logistic regression is one such model.

Consider binary classification with $y = -1, 1$, and each example is represented by a feature vector x. The intuition is to first map x to a real number, such that very positive number means x is likely to be positive ($y = 1$), and very negative number means x is negative ($y = -1$). This can be done via the inner product to a parameter vector θ:

$$\theta^\top x \tag{1}$$

Note this is a linear mapping. The inner product is in $(-\infty, \infty)$. It is worth emphasizing that $\theta \in \mathbb{R}^d$, where d is the dimensionality of input features x, instead of the d-simplex. In other words, here θ is a vector of d arbitrary real numbers. In contrast, in naive Bayes $\theta_y = p(x|y)$ must be a probability vector and thus is in the d-simplex.

A practical note: It is often convenient to append a constant feature with value one to x, and the dimensionality of θ is increased by one accordingly. This new dimension serves as an offset, which is equivalent to $\theta^\top x + \theta_0$.

The next step is to squash the range down to $[0, 1]$ so one can interpret it as a probability. This is usually done via a *logistic sigmoid* function:

$$p(y = 1|x) = \sigma(\theta^\top x) = \frac{1}{1 + \exp(-\theta^\top x)}. \tag{2}$$

For binary classification with -1, 1 class encoding, one can unify the definition for $p(y = 1|x)$ and $p(y = -1|x)$ with

$$p(y|x) = \frac{1}{1 + \exp(-y\theta^\top x)}. \tag{3}$$

Logistic regression can be easily generalized to multiple classes. Let there be K classes. Each class has its own[1] parameter θ_k, which maps x to $\theta_k^\top x$. The probability is defined via the *softmax* function

$$p(y = k|x) = \frac{\exp(\theta_k^\top x)}{\sum_{i=1}^K \exp(\theta_i^\top x)}. \tag{4}$$

We however focus on binary classification in the rest of this note.

0.1 Training

Training (i.e., finding the parameter θ) can be done by maximizing the *conditional* log likelihood of training data $\{(x, y)_{1:n}\}$:

$$\max_\theta \sum_{i=1}^n \log p(y_i|x_i, \theta). \tag{5}$$

However, when the training data is linearly separable, two bad things happen: 1. $\|\theta\|$ goes to infinity; 2. There are infinite number of MLE's. To see this, note any step function (sigmoid with $\|\theta\| = \infty$) that is in the gap between the two classes is an MLE.

One way to avoid this is to incorporate a prior on θ in the form of a zero-mean Gaussian with covariance $\frac{1}{2\lambda}I$,

$$\theta \sim \mathcal{N}\left(0, \frac{1}{2\lambda}I\right), \tag{6}$$

and seek the MAP estimate. This is essentially smoothing, since large θ values will be penalized more. That is,

$$\begin{aligned} & \max_\theta \log p(\theta, y_{1:n}|x_{1:n}) && (7)\\ = \quad & \log p(\theta) + \sum_{i=1}^n \log p(y_i|x_i, \theta) && (8)\\ = \quad & -\lambda\|\theta\|^2 + \sum_{i=1}^n \log p(y_i|x_i, \theta) && (9)\\ = \quad & -\lambda\|\theta\|^2 - \sum_{i=1}^n \log\left(1 + \exp(-y_i\theta^\top x_i)\right). && (10)\end{aligned}$$

Equivalently, one minimizes the ℓ_2-regularized negative log likelihood loss

$$\min_\theta \lambda\|\theta\|^2 + \sum_{i=1}^n \log\left(1 + \exp(-y_i\theta^\top x_i)\right). \tag{11}$$

[1]Strictly speaking, one needs only $K - 1$ parameter vectors. One of the classes can have an all-one parameter vector to remove this degree of freedom.

This is a convex function so there is a unique global minimum of θ. Unfortunately there is no closed form solution. One typically solves the optimization problem with Newton-Raphson iterations, also known as *iterative reweighted least squares* for logistic regression.

0.2 Graphical Model

Logistic regression can be represented as a directed graphical model (Bayes Network) with a y node, and a set of x nodes (one for each feature dimension). The arrows go from the x nodes to the y node. Note this is exactly the opposite of Naive Bayes models.

Further notice that we do not model $p(x)$ in logistic regression. Therefore it is *not* possible to have a sampling program to generate (x, y) data. This is a difference between generative (e.g., Naive Bayes) vs. discriminative (e.g., logistic regression) models.

0.3 Logistic Regression as a Linear Classifier

Logistic regression is a linear classifier, with the decision boundary

$$\theta^\top x = 0. \tag{12}$$

Recall that Naive Bayes is a linear classifier too. They both divide the feature space $\mathcal{X}$ with a hyperplane. Their essential difference is how they find their hyperplane: Naive Bayes optimizes a generative objective function, while logistic regression optimizes a discriminative objective function. It has been shown that logistic regression tends to have higher accuracy when training data is plenty. However, Naive Bayes can have an advantage when the training data size is small.