ML

Runi Malladi

July 2, 2023

1 unsorted

1.1 transforming feature vectors

Situation

We represent words as feature n-vectors. There exist two subclasses of words, say $\mathcal{C}$ and $\mathcal{D}$. We believe there is a relationship between a word and another word, e.g.

- a country and its capital
- an English word and its Telugu translation

We specifically assume this relationship is a linear transformation, consistent between individual feature vectors, i.e. the same linear transform maps

- every country to its capital
- every English word to its Telugu translation.

Approximations

In reality, the relationship is not exactly the same for all words; partly because of noise in measurements, partly because the mapping between $\mathcal{C}$ and $\mathcal{D}$ is not one-to-one (e.g. there isn't a direct translation for every English word).

Consider m feature (n-)vectors belonging to $\mathcal{C}$. Arrange then as the rows of an $m \times n$ matrix A. Arrange their corresponding vectors in D into an $(m \times n)$ matrix B.

Let R be an $(n \times n)$-matrix, and let's consider what the product AR represents. Well the ith row of AR is $A_{i,\bullet}R$, hence R is a lienar map sending rows of A to rows of AR. Hence we seek an R which is the best approximation

$$AR = B.$$

measuring closeness

Since we can't hope for R to exactly map rows of A to the rows of B, we seek a best approximation. But how do we measure how good an approximation is? One way is to use the Frobenius norm.

Definition 1.1. Let A be an $m \times n$ matrix. The *Frobenius norm* of A is the square root of the sum of the squares of its entries:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{i,j})^2}.$$

Equivalently,

$$\|A\|_F = \sqrt{\mathrm{tr}(A^T A)}.$$

Proposition 1.2. The two definitions are equivalent.

Proof.

$$\begin{aligned}\mathrm{tr}(A^T A) =& \sum_{j=1}^{n} A^T_{j,\bullet} A_{\bullet,j} = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} A^T_{j,i} A_{i,j}\right)\\ =& \sum_{j=1}^{n}\sum_{i=1}^{m} A^2_{i,j}.\end{aligned}$$

□

Using the Frobenius norm, our goal now is to minimize

$$\mathrm{Loss}(R) = \|AR - B\|_F.$$

We use gradient descent.

Proposition 1.3.

$$\nabla_R \mathrm{Loss}(R) = 2A^T(AR - B).$$

Proof. Be sure to check our conventions. Well

$$\begin{aligned}\mathrm{Loss}(R) =& \|AR - B\|_F^2 = \mathrm{tr}((AR - B)^T(AR - B))\\ =& \mathrm{tr}((R^T A^T - B^T)(AR - B))\\ =& \mathrm{tr}(R^T A^T AR - R^T A^T B - B^T AR - B^T B).\end{aligned}$$

Then

$$\begin{aligned}\frac{\partial}{\partial R}\mathrm{Loss}(R) =& \frac{\partial}{\partial R}\mathrm{tr}(R^T(A^T A)R) - \frac{\partial}{\partial R}(R^T(A^T B)) - \frac{\partial}{\partial R}((B^T A)R) - \frac{\partial}{\partial R}(B^T B)\\ =& R^T(A^T A + A^T A) - B^T A - B^T A = 2(R^T A^T A - B^T A)\\ =& 2(R^T A^T - B^T)A.\end{aligned}$$

Then

$$\nabla_R \mathrm{Loss}(R) = (\frac{\partial}{\partial R}\mathrm{Loss}(R))^T = 2A^T(AR - B).$$

□

2 appendix

2.1 logistic regression

The purpose of logistic regression is to take data associating various values of the independent variables to binary outcomes and produce a model which takes values of the independent variables and returns a probability of a binary outcome occuring.

Background

Consider p to be the probability of an event occurring. We can further assume only two outcomes: either the event occurs, or it doesn't. We define the *odds ratio* to be

$$\begin{aligned} \text{odds ratio} : [0,1) &\to [0,\infty) \\ p &\mapsto \frac{p}{1-p}. \end{aligned}$$

We define the *log-odds ratio*, or *logit*, to be

$$\begin{aligned} \text{logit} : (0,1) &\to (-\infty,\infty) \\ p &\mapsto \log\left(\frac{p}{1-p}\right). \end{aligned}$$

The graphs of these functions are depicted below (Figure 1):

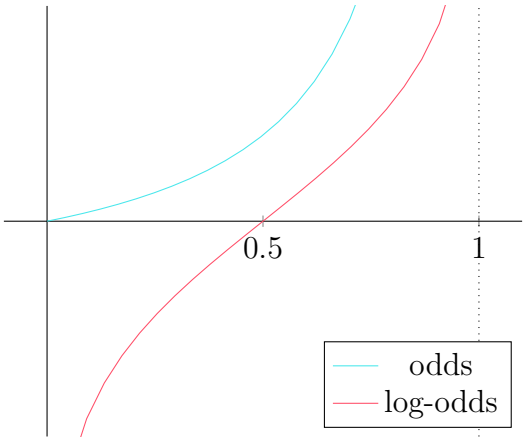

Figure 1: Graphs of odds and log-odds functions.

Assumptions

The fundamental assumption of logistic regression is a linear relationship between the independent variables and the log-odds.

For instance, consider a situation with two independent variables X_1, X_2 which determine a binary outcome (either 0 or 1). We assume

- it is reasonable to model the probability of an input (x_1, x_2) resulting in the binary outcome 1. That is, each outcome y_i is Bernoulli distributed.
- this relationship is linear: letting p denote the probability of (x_1, x_2) producing 1, we have
$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2.$$

for some $\beta_0, \beta_1, \beta_2 \in \mathbb{R}$. Note that the β_i do not depend on the x_i.

Objective

Assuming a linear relationship between log-odds and the independent variables

$$\text{logit}(p(x)) = \beta \cdot \begin{pmatrix} 1 \\ x \end{pmatrix} = \beta_0 + \beta_1 x_1 + \cdots \beta_n x_n,$$

the objective of logistic regression is to determine (or approximate) the coefficients β in the above linear combination. As a matter of convention, by $\beta \cdot x$ or $\beta^T x$ we will mean the above dot product, where we have added an $x_0 = 1$ term to the original x.

As we will demonstrate, once the coefficients β have been determined, we can determine the probability $p(x)$ of input x succeeding using the following formula:

$$p(x) = \frac{1}{1 + e^{-\beta^T x}}.$$

2.2 naive Bayes classifier

The situation is when we have some number of features (random variables) $x = (x_1, \ldots, x_n)$ and finitely many possible outcomes C_k. The *naive Bayes probabilistic model* is a computation of the conditional probabilities $p(C_k|x)$, i.e. the probability of the outcome C_k given the features x. The *naive Bayes classifier* determines which outcome C_k is most likely given the feature x. It essentially picks the largest conditional probability $p(C_k|x)$. What makes both of these models "naive" is their assumption on the independence of the features in x, which greatly simplifies the above computations.

background

The key insight is the simplification provided by assuming the features in x are independent. Specifically, we are assuming that the probability of the ith feature equaling x_i is indenpendent of the jth feature equaling x_j (provided $i \neq j$). Then $p(x_i|x_j) = p(x_i)$.

Let's see this simplification in action (this computation will be relevant later). By Bayes' rule, we can write a joint probability in terms of the conditional probability:

$$p(x, y) = p(x|y)p(y).$$

Using this, we can write

$$\begin{aligned} p(C_k, x_1, \ldots, x_n) =& p(x_1, \ldots, x_n, C_k) \\ =& p(x_1|x_2, \ldots, x_n, C_k)p(x_2, \ldots, x_n, C_k) \\ =& \cdots \\ =& p(x_1|x_2, \ldots, x_n, C_k)p(x_2|x_3, \ldots, x_n, C_k) \cdots p(x_{n-1}|x_n, C_k)p(x_n|C_k)p(C_k). \end{aligned}$$

By our independence assumption,

$$p(x_i|x_{i+1}, \ldots, x_n, C_k) = p(x_i|C_k).$$

So

$$p(C_k, x_1, \ldots, x_n) = p(C_k) \prod_{i=1}^{n} p(x_i|C_k).$$

assumptions

We assume that the values of each feature are independent of each other. For example, given a collection of features $X = (X_1, \ldots, X_n)$, the features X_i and X_j are independent (provided $i \neq j$).

This is rarely true, and the dependence among features is often significant. Suppose we are trying to classify fruits. Suppose we are tracking two features: color and taste. Is it reasonable to assume these two are independent? You have to decide.

the probabilistic model

As mentioned earlier, the naive Bayes probabilistic model is concerned with computing the conditional probability $p(C_k|x)$.

Proposition 2.1. Under the assumptions of the naive Bayes model,

$$p(C_k|x) = \frac{p(C_k)}{p(x)} \prod_{i=1}^{n} p(x_i|C_k).$$

Proof. We compute using the independence assumption:

$$\begin{aligned} p(C_k|x) =& \frac{p(C_k, x)}{p(x)} = \frac{p(C_k, x_1, \ldots, x_n)}{p(x)} \\ =& \frac{p(C_k)}{p(x)} \prod_{i=1}^{n} p(x_i|C_k). \end{aligned}$$

□

the classifier

The naive Bayes classifier is concerned with, given a fixed x, which conditional probability $p(C_k|x)$ is the largest. Intuitively, it determines which outcome C_k is most likely given the features x.

Proposition 2.2. For a fixed x,

$$\underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k|x) = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k).$$

Proof. The point is that we can remove the $p(x)$ from the probabilistic model. We can do this because we fix x, so $p(x)$ is a constant factor shared by all terms we are taking the argmax over. □

Definition 2.3. The naive Bayes classifier is the function

$$\begin{aligned} &X_1 \times \cdots X_n \to \{C_k\} \\ &(x_1, \ldots, x_n) \mapsto \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k). \end{aligned}$$

Let's consider a special case. There are two possible outcomes C_1, C_2 and n features $X_1, \ldots, X_n$. Then we just need to compare the two conditional probabilities $p(C_1|x)$ and $p(C_2|x)$. Assuming $p(C_2|x) \neq 0$, we can consider the ratio

$$R = \frac{p(C_1|x)}{p(C_2|x)} = \frac{p(C_1)\prod_{i=1}^{n} p(x_i|C_k)}{p(C_2)\prod_{i=1}^{n} p(x_i|C_2)}.$$

If $R > 1$ the C_1 is more likely, if $R < 1$ then C_2 is more likely, and if $R = 1$ the both are equally likely.

2.3 additive smoothing

Additive smoothing helps assign a nonzero probability to conditional properties which are calculated to be zero from the data.

background

Consider a d-dimensional multinomial distribution, of which we take N trials. This means that there are d possible outcome classes, and we have classified N data points into these classes (one at a time, so the same data point may be placed in different classes from one trial to another). Suppose we take another data point x_i. We want to determine the probability.

$$p(x_i|C_k).$$

By Bayes' theorem, we know

$$p(x_i|C_k) = \frac{p(x_i, C_k)}{p(C_k)}.$$

Let $\text{freq}(x_i, C_k)$ denote the number of times x_i was classified as C_k (in the N trials we took of the multinomial distribution). Note this may be 0. Also let N_k be the number of times a data point was sorted into class C_k. Then, assuming $N_k \neq 0$,

$$p(x_i, C_k) = \frac{\text{freq}(x_i, C_k)}{N},$$
$$p(C_k) = \frac{N_k}{N},$$
$$p(x_i|C_k) = \frac{\text{freq}(x_i, C_k)}{N_k}.$$

Now there are two potential problems here. The first is that $p(x_i|C_k) = 0$ if $\text{freq}(x_i, C_k)$. But it might not be accurate to say this; just because x_i was never classified as C_k in any over our N trials doesn't necessarily mean that it should never by classified as C_k. We took a finite number of trials which may not accurate represent the true probabilities. The second issue is that we assumed $N_k \neq 0$, i.e. that during our N trials at least one data point was classified as C_k. This may also not be reasonable to assume.

assumptions

We assume that each data point x_i has a nonzero probability of being C_k. The extent of this assumption is codified in the "pseudocount" parameter in the additive smoothing formula.

formula

Additive smoothing redefines all conditional probabilities $p(x_i|C_k)$:

Definition 2.4. Additive smoothing with pseudocount $\alpha > 0$ is

$$\hat{p}(x_i|C_k) = \frac{\text{freq}(x_i, C_k) + \alpha}{N_k + d\alpha}.$$

The reason this is reasonable is the following:

Proposition 2.5. Suppose α is an integer. Consider n data points $x_1, \ldots, x_n$. Suppose we have a multinomial distribution with $N + dn\alpha$ trials, which is identical to the old distribution for the first N trails and afterwards each data point x_i was classified α times into each C_k. Then

$$p_{\text{new}}(x_i|C_k) = \frac{\text{freq}_{\text{old}}(x_i, C_k) + \alpha}{(N_k)_{\text{old}} + n\alpha}.$$

Proof. Well

$$\text{freq}_{\text{new}}(x_i, C_k) = \text{freq}_{\text{old}}(x_i, C_k) + \alpha,$$
$$(N_k)_{\text{new}} = (N_k)_{\text{old}} + n\alpha.$$

Just plug that into

$$p_{\text{new}}(x_i|C_k) = \frac{\text{freq}_{\text{new}}(x_i, C_k)}{(N_k)_{\text{neq}}}.$$

□