

Logistic Regression and Stochastic Gradient Training

Charles Elkan
elkan@cs.ucsd.edu

November 13, 2007

An important extension of the idea of likelihood is *conditional* likelihood. The conditional likelihood of θ given data x and y is $L(\theta; y|x) = f(y|x; \theta)$. Intuitively, y follows a probability distribution that is different for different x , but x itself is never unknown, so there is no need to have a probabilistic model of it. Technically, for each x there is a different distribution $f(y|x; \theta)$ of y , but all these distributions share the same parameters θ .

Given training data consisting of $\langle x_i, y_i \rangle$ pairs, the principle of maximum conditional likelihood says to choose a parameter estimate $\hat{\theta}$ that maximizes the product $\prod_i f(y_i|x_i; \theta)$. Note that we do not need to assume that the x_i are independent in order to justify the conditional likelihood being a product; we just need to assume that the y_i are independent conditional on the x_i . For any specific value of x , $\hat{\theta}$ can be used to predict values for y ; we assume that we never want to predict values of x .

If y is a binary outcome and x is a real-valued vector, then the conditional model

$$p = p(y|x; \alpha, \beta) = \frac{1}{1 + \exp -[\alpha + \sum_{j=1}^d \beta_j x_j]}$$

is called logistic regression. We use j to index over the feature values x_1 to x_d of a single example of dimensionality d , since we use i below to index over training examples 1 to n .

The logistic regression model is easier to understand in the form

$$\log \frac{p}{1-p} = \alpha + \sum_j \beta_j x_j.$$

The ratio $p/(1-p)$ is called the odds of the event y given x , and $\log[p/(1-p)]$ is called the log odds. Since probabilities range between 0 and 1, odds range between 0 and $+\infty$ and log odds range unboundedly between $-\infty$ and $+\infty$. A linear expression of the form $\alpha + \sum_j \beta_j x_j$ can also take unbounded values, so it is reasonable to use a linear expression as a model for log odds, but not as a model for odds or for probabilities. Essentially, logistic regression is the simplest possible model for a random yes/no outcome that depends linearly on predictors x_1 to x_d .

For each feature j , $\exp(\beta_j x_j)$ is a multiplicative scaling factor on the odds $p/(1-p)$. If the predictor x_j is binary, then $\exp(\beta_j)$ is the extra odds of having the outcome $y = 1$ when $x_j = 1$, compared to when $x_j = 0$.

Given x and y , the conditional log likelihood is $\log L(\beta; x, y) = \log p$ if $y = 1$ and $\log L(\beta; x, y) = \log(1-p)$ if $y = 0$. In order to maximize this, we need to evaluate its partial derivative with respect to each parameter β_j . To simplify the following discussion, assume that $\alpha = \beta_0$ and $x_0 = 1$ for every example x from now on. If $y = 1$ the partial derivative is

$$\frac{\partial}{\partial \beta_j} \log p = \frac{1}{p} \frac{\partial}{\partial \beta_j} p$$

while if $y = 0$ it is

$$\frac{\partial}{\partial \beta_j} \log(1-p) = \frac{1}{1-p} \left(- \frac{\partial}{\partial \beta_j} p \right).$$

Let $e = \exp[-\sum_j \beta_j x_j]$ where the sum ranges from $j = 0$ to $j = d$, so $p = 1/(1+e)$ and $1-p = (1+e-1)/(1+e) = e/(1+e)$. With this notation we have

$$\begin{aligned} \frac{\partial}{\partial \beta_j} p &= (-1)(1+e)^{-2} \frac{\partial}{\partial \beta_j} e \\ &= (-1)(1+e)^{-2} (e) \frac{\partial}{\partial \beta_j} [-\sum_j \beta_j x_j] \\ &= (-1)(1+e)^{-2} (e)(-x_j) \\ &= \frac{1}{1+e} \frac{e}{1+e} x_j \\ &= p(1-p)x_j. \end{aligned}$$

So $(\partial/\partial \beta_j) \log p = (1-p)x_j$ and $(\partial/\partial \beta_j) \log(1-p) = -px_j$. Given training examples $\langle x_1, y_1 \rangle$ to $\langle x_n, y_n \rangle$, the total partial derivative of the log likelihood with

respect to β_j is

$$\sum_{i:y_i=1} (1 - p_i)x_{ij} + \sum_{i:y_i=0} -p_ix_{ij} = \sum_i (y_i - p_i)x_{ij}$$

where x_{ij} is the value of the j th feature of the i th training example. Setting the total partial derivative to zero yields

$$\sum_i y_ix_{ij} = \sum_i p_ix_{ij}.$$

We have one equation of this type for each parameter β_j . The equations can be used to check the correctness of a trained model.

There are several sophisticated ways of actually doing the maximization of the total conditional log likelihood, i.e. the conditional log likelihood summed over all training examples $\langle x_i, y_i \rangle$; for details see [Min07, KM05]. However, here we consider a method called stochastic gradient ascent. This method changes the parameter values to increase the log likelihood based on one example at a time. It is called stochastic because the derivative based on a randomly chosen single example is a random approximation to the true derivative based on all the training data.

Consider a single training example $\langle x, y \rangle$, where again we drop the subscript i for convenience. Consider the j th parameter for $0 \leq j \leq d$. The partial derivative of the log likelihood given this single example is

$$\frac{\partial}{\partial \beta_j} \log L(\beta; x, y) = (y - p)x_j$$

where $y = 1$ or $y = 0$. For each j , we increase the log likelihood incrementally by doing the update $\beta_j := \beta_j + \lambda(y - p)x_j$. Here λ is a multiplier called the learning rate that controls the magnitude of the changes to the parameters.

Stochastic gradient ascent (or descent, for a minimization problem) is a method that is often useful in machine learning. Experience suggests some heuristics for making it work well in practice.

- The training examples are sorted in random order, and the parameters are updated for each example sequentially. One complete update for every example is called an epoch. Typically, a small constant number of epochs is used, perhaps 3 to 100 epochs.

- The learning rate is chosen by trial and error. It can be kept constant across all epochs, e.g. $\lambda = 0.1$ or $\lambda = 1$, or it can be decreased gradually as a function of the epoch number.
- Because the learning rate is the same for every parameter, it is useful to scale the features x_j so that their magnitudes are similar for all j . Given that the feature x_0 has constant value 1, it is reasonable to normalize every other feature to have mean zero and variance 1, for example.

Stochastic gradient ascent (or descent) has some properties that are very useful in practice. First, suppose that $x_j = 0$ for most features j of a training example x . Then updating β_j based on x can be skipped. This means that the time to do one epoch is $O(nfp)$ where n is the number of training examples, p is the number of features, and f is the average number of nonzero feature values per example. If an example x is the bag-of-words representation of document, then p is the size of the vocabulary but fp is the average length of a document.

Second, suppose that the number n of training examples is very large, as is the case in many modern applications. Then, a stochastic gradient method may converge to good parameter estimates in less than one epoch of training. In contrast, a training method that computes the log likelihood of all data and uses this in the same way regardless of n will be inefficient in how it uses the data.

For each example, a stochastic gradient method updates all parameters once. The dual idea is to update one parameter at a time, based on all examples. This method is called coordinate ascent (or descent). For feature j the update rule is

$$\beta_j := \beta_j + \lambda \sum_i (y_i - p_i) x_{ij}.$$

The update for the whole parameter vector $\bar{\beta}$ is

$$\bar{\beta} := \bar{\beta} + \lambda(\bar{y} - \bar{p})^T X$$

where the matrix X is the entire training set and the column vector \bar{y} consists of the 0/1 labels for every training example. Often, coordinate ascent converges too slowly to be useful. However, it can be useful to do one update of $\bar{\beta}$ after all epochs of stochastic gradient ascent.

Regardless of the method used to train a model, it is important to remember that optimizing the model perfectly on the training data usually does not lead to the best possible performance on test examples. There are several reasons for this:

- The model with best possible performance may not belong to the family of models under consideration. This is an instance of the principle “you cannot learn it if you cannot represent it.”
- The training data may not be representative of the test data, i.e. the training and test data may be samples from different populations.
- Fitting the training data as closely as possible may simply be overfitting.
- The objective function for training, namely log likelihood or conditional log likelihood, may not be the desired objective from an application perspective; for example, the desired objective may be classification accuracy.

References

- [KM05] Paul Komarek and Andrew W. Moore. Making logistic regression a core data mining tool with TR-IRLS. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 685–688, 2005.
- [Min07] Thomas P. Minka. A comparison of numerical optimizers for logistic regression. First version 2001. Unpublished paper available at <http://research.microsoft.com/~minka>, 2007.