Logistic Regression and Stochastic Gradient Training

Charles Elkan
elkan@cs.ucsd.edu

January 29, 2010

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

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

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

$$\log \frac{p}{1-p} = \alpha + \sum_{j=1}^{d} \beta_j x_j$$

where p is an abbreviation for $p(y|x;\alpha,\beta)$. 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 $x_j=0$. If the predictor x_j is real-valued, then $\exp(\beta_j)$ is the extra odds of having the outcome $y=1$ when the value of x_j increases by one unit. A major limitation of the basic logistic regression model is that the probability p must either increase monotonically, or decrease monotonically, as a function of each predictor x_j. The basic model does not allow the probability to depend in a U-shaped way on any x_j.

Given the training set $\{\langle x_1, y_1\rangle, \ldots, \langle x_n, y_n\rangle\}$, we learn a logistic regression classifier by maximizing the log joint conditional likelihood. This is the sum of the log conditional likelihood for each training example:

$$LCL = \sum_{i=1}^{n} \log L(\theta; y_i|x_i) = \sum_{i=1}^{n} \log f(y_i|x_i;\theta).$$

Given a single training example $\langle x_i, y_i\rangle$, the log conditional likelihood is $\log p_i$ if the true label $y_i=1$ and $\log(1-p_i)$ if $y_i=0$, where $p_i = p(y=1|x_i;\theta)$.

To simplify the following discussion, assume from now on that $\alpha=\beta_0$ and $x_0=1$ for every example x, so the parameter vector θ is $\beta\in\mathbb{R}^{d+1}$. By grouping together the positive and negative training examples, we can write the total conditional log likelihood as

$$LCL = \sum_{i:y_i=1} \log p_i + \sum_{i:y_i=0} \log(1-p_i).$$

The partial derivative of LCL with respect to parameter β_j is

$$\sum_{i:y_i=1} \frac{\partial}{\partial\beta_j} \log p_i + \sum_{i:y_i=0} \frac{\partial}{\partial\beta_j} \log(1-p_i).$$

For an individual training example $\langle x, y\rangle$, if its label $y=1$ 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

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

Let $e = \exp[-\sum_{j=0}^{d} \beta_j x_j]$ 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

$$\frac{\partial}{\partial \beta_j} \log p = (1-p)x_j \quad \text{and} \quad \frac{\partial}{\partial \beta_j} \log(1-p) = -p x_j.$$

For the entire training set the partial derivative of the log conditional likelihood with respect to β_j is

$$\frac{\partial}{\partial \beta_j} LCL = \sum_{i:y_i=1} (1-p_i)x_{ij} + \sum_{i:y_i=0} -p_i x_{ij} = \sum_i (y_i - p_i)x_{ij}$$

where x_{ij} is the value of the jth feature of the ith training example. Setting the partial derivative to zero yields

$$\sum_i y_i x_{ij} = \sum_i p_i x_{ij}.$$

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.

Informally, but not precisely, the expression $\sum_i y_i x_{ij}$ is the average value over the training set of the ith feature, where each training example is weighted 1 if its true label is positive, and 0 otherwise. The expression $\sum_i p_i x_{ij}$ is the same average, except that each example i is weighted according to its predicted probability p_i of being positive. When the logistic regression classifier is trained correctly,

then these two averages must be the same for every feature. The special case for $j = 0$ gives

$$\frac{1}{n}\sum_i y_i = \frac{1}{n}\sum_i p_i.$$

In words, the empirical base rate probability of being positive must equal the average predicted probability of being positive.

1 Stochastic gradient training

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 [Minka, 2007, Komarek and Moore, 2005]. 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 jth 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 $O(nfd)$ where n is the number of training examples, d 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 d is the size of the vocabulary (often over 30,000) but fd is the average number of words actually used in a document (often under 300).

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.

2 Regularization

Consider learning a logistic regression classifier for categorizing documents. Suppose that word number j appears only in documents whose labels are positive. The partial derivative of the log conditional likelihood with respect to the parameter for this word is

$$\frac{\partial}{\partial \beta_j} LCL = \sum_i (y_i - p_i) x_{ij}.$$

This derivative will always be positive, as long as the predicted probability p_i is not perfectly one for all these documents. Therefore, following the gradient will send β_j to infinity. Then, every test document containing this word will be predicted to be positive with certainty, regardless of all other words in the test document. This over-confident generalization is an example of overfitting.

There is a standard method for solving this overfitting problem that is quite simple, but quite successful. The solution is called regularization. The idea is to impose a penalty on the magnitude of the parameter values. This penalty should be minimized, in a trade-off with maximizing likelihood. Mathematically, the optimization problem to be solved is

$$\hat{\beta} = \text{argmax}_\beta LCL - \mu ||\beta||_2$$

where $||\beta||_2$ is the L_2 norm of the parameter vector, and the constant μ quantifies the trade-off between maximizing likelihood and minimizing parameter values.

This type of regularization is called quadratic or Tikhonov regularization. A major reason why it is popular is that it can be derived from several different points

of view. In particular, it arises as a consequence of assuming a Gaussian prior on parameters. It also arises from theorems on minimizing generalization error, i.e. error on independent test sets drawn from the same distribution as the training set. And, it arises from robust classification: assuming that each training point lies in an unknown location inside a sphere centered on its measured location.

Stochastic gradient following is easily extended to include regularization. We simply include the penalty term when calculating the gradient for each example. Consider

$$\frac{\partial}{\partial \beta_j}[\log p(y|x;\beta) - \mu \sum_{j=0}^{d} \beta_j^2] = [\frac{\partial}{\partial \beta_j} \log p(y|x;\beta)] - \mu 2\beta_j.$$

Remember that for logistic regression the partial derivative of the log conditional likelihood for one example is

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

so the update rule with regularization is

$$\beta_j := \beta_j + \lambda[(y-p)x_j - 2\mu\beta_j]$$

where λ is the learning rate. Update rules like the one above are often called "weight decay" rules, since the weight β_j is made smaller at each update unless $y - p$ has the same sign as x_j.

Straightforward stochastic gradient ascent for training a regularized logistic regression model loses the desirable sparsity property described above, because the value of every parameter β_j must be decayed for every training example. How to overcome this computational inefficiency is described in [Carpenter, 2008].

Writing the regularized optimization problem as a minimization gives

$$\hat{\beta} = \text{argmin}_\beta \sum_{i=1}^{n} -\log p(y_i|x_i;\beta) + \mu \sum_{j=0}^{d} \beta_j^2.$$

The expression $-\log p(y_i|x_i;\beta)$ is called the "loss" for training example i. If the predicted probability, using β, of the true label y_i is close to 1, then the loss is small. But if the predicted probability of y_i is close to 0, then the loss is large. Losses are always non-negative; we want to minimize them. We also want to minimize the numerical magnitude of the trained parameters.

CSE 250B Quiz 4, January 28, 2010

The objective function to be minimized when training an L_2-regularized linear regression model is

$$E = \sum_{i=1}^{n} (f(x_i; w) - y_i)^2 + \mu \sum_{j=0}^{d} w_j^2$$

where the model is

$$f(x_i; w) = \sum_{j=0}^{d} w_j x_{ij}.$$

All notation above is the same as in the class lecture notes.

[3 points] Work out the partial derivative of the objective function with respect to weight w_j.

Answer. Let f_i be an abbreviation for $f(x_i; w)$ and let d_i be an abbreviation for $\frac{\partial}{\partial w_j} f_i$. Note that $d_i = x_{ij}$. The expanded objective function is

$$E = [\sum_{i=1}^{n} f_i^2 - 2 f_i y_i + y_i^2] + \mu \sum_{j=0}^{d} w_j^2.$$

The partial derivative is

$$\frac{\partial}{\partial w_j} E = [\sum_{i=1}^{n} [2 f_i d_i - 2 y_i d_i + 0] + 2\mu w_j$$

which is

$$\frac{\partial}{\partial w_j} E = 2[\mu w_j + \sum_{i=1}^{n} (f_i - y_i) x_{ij}].$$

Additional note. Because the goal is to minimize E, we do gradient descent, not ascent, with the update rule

$$w_j := w_j - \lambda \frac{\partial}{\partial w_j} E.$$

The update rule says that if the average over all training examples i of $(f_i - y_i) x_{ij}$ is positive, then w_j must be decreased. Assume that x_{ij} is non-negative; this update rule is reasonable because then f_i is too big on average, and decreasing w_j will make f_i decrease. The update rule also says that, because of regularization, w_j must always be decreased even more, by $2\lambda\mu$ times its current value.

Project assignment 2

For this assignment you may work in a team of two with one partner, or in a team of three. For multiple reasons, working alone or in a larger team is not appropriate. The joint report for your team must be submitted in hard copy at the start of class on Thursday, February 4, 2010.

This project uses data published by Kevin Hillstrom, a well-known data mining consultant. You can find the data at `http://cseweb.ucsd.edu/users/elkan/250B/HillstromData.csv`. For a detailed description see `http://minethatdata.com/blog/2008/03/minethatdata-e-mail-analytics-and-data.html`.

You should build three separate models: one for customers who are not sent any email promotion, one for customers who are sent the "men's clothing" email, and one for customers who are sent the "women's clothing" email. Each model consists of two regularized logistic regression classifiers and one linear regression function. The three submodels are part of the overall model

$$E[\text{spend}|x, \text{treatment}] = E[\text{spend}|\text{purchase}, x, \text{treatment}]\cdot$$

$$p(\text{purchase}|\text{visit}, x, \text{treatment})\cdot p(\text{visit}|x, \text{treatment}).$$

In your report, explain the equation above carefully. Note that

- x is the vector of attribute values describing a customer,
- "women's clothing" email, "men's clothing" email, and no email are the three possible values of the random variable "treatment,"
- "visit" and "purchase" are binary random variables that have a certain probability of being true or false for each customer, and
- "spend" is a real-valued random variable for each customer.

You should implement your own Matlab code for training logistic regression with regularization. The optimization method should be stochastic gradient descent (SGD). Your implementation should handle discrete features as well as real-valued features, and up to 30,000 training examples with up to 30 features. One difficulty with SGD is choosing the learning rate λ. You can find advice at `http://leon.bottou.org/projects/sgd`. Use a grid search with cross-validation to select a good value for the strength of regularization. In your

report, analyze experimentally and theoretically the big-O time and space complexity of your implementation. Explain any non-trivial obstacles that you did or did not overcome in trying to achieve good time complexity.

You must think carefully about how to do experiments that are conceptually sound. Your report should contain a separate section that explains the design of your experiments. Use a separate final test set of customers to estimate the total revenue achievable by sending the "men's clothing" and "women's clothing" email to optimal subsets of customers. Also report the results of separate experiments to confirm that your logistic regression classifiers yield well-calibrated conditional probabilities, and that your linear regression functions yield unbiased estimates of spending amounts.

You may want to use a scripting language to turn the raw data into a numerical dataset for input into Matlab. Describe any preprocessing and feature selection that you do in your report.

References

[Carpenter, 2008] Carpenter, B. (2008). Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. Technical report, Alias-i. Available at `http://lingpipe-blog.com/lingpipe-white-papers/`.

[Komarek and Moore, 2005] Komarek, P. and Moore, A. W. (2005). 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.

[Minka, 2007] Minka, T. P. (2007). A comparison of numerical optimizers for logistic regression. First version 2001. Unpublished paper available at `http://research.microsoft.com/~minka`.