Lazy Sparse Stochastic Gradient Descent for Regularized Mutlinomial Logistic Regression

Bob Carpenter
Alias-i, Inc.
carp@alias-i.com

Abstract

Stochastic gradient descent efficiently estimates maximum likelihood logistic regression coefficients from sparse input data. Regularization with respect to a prior coefficient distribution destroys the sparsity of the gradient evaluated at a single example. Sparsity is restored by lazily shrinking a coefficient along the cumulative gradient of the prior just before the coefficient is needed.

1 Multinomial Logistic Model

A multinomial logistic model classifies d-dimensional real-valued input vectors $x \in \mathbb{R}^d$ into one of k outcomes $c \in \{0, \ldots, k-1\}$ using $k-1$ parameter vectors $\beta_0, \ldots, \beta_{k-2} \in \mathbb{R}^d$:

$$p(c \mid x, \beta) = \begin{cases} \frac{\exp(\beta_c \cdot x)}{Z_x} & \text{if } c < k-1 \\ \frac{1}{Z_x} & \text{if } c = k-1 \end{cases} \tag{1}$$

where the linear predictor is inner product:

$$\beta_c \cdot x = \sum_{i<d} \beta_{c,i} \cdot x_i \tag{2}$$

The normalizing factor in the denominator is the partition function:

$$Z_x = 1 + \sum_{c<k-1} \exp(\beta_c \cdot x) \tag{3}$$

2 Corpus Log Likelihood

Given a sequence of n data points $D = \langle x_j, c_j \rangle_{j<n}$, with $x_j \in \mathbb{R}^d$ and $c_j \in \{0, \ldots, k-1\}$, the log likelihood of the data in a model with parameter matrix β is:

$$\log p(D \mid \beta) = \log \prod_{j<n} p(c_j \mid x_j, \beta) = \sum_{j<n} \log p(c_j \mid x_j, \beta) \tag{4}$$

3 Maximum Likelihood Estimate

The maximum likelihood (ML) estimate $\hat{\beta}$ is the value of β maximizing the likelihood of the data D:

$$\hat{\beta} = \arg\max_{\beta} \; p(D \mid \beta) = \arg\max_{\beta} \; \log p(D \mid \beta) \tag{5}$$

4 Maximum a Posteriori Estimate

Given a prior probability density $p(\beta)$ over a parameter matrix β, the maximum a posteriori (MAP) estimate $\hat{\beta}$ is:

$$\begin{aligned} \hat{\beta} &= \arg\max_{\beta} \; p(\beta \mid D) \\ &= \arg\max_{\beta} \; \frac{p(D \mid \beta) \; p(\beta)}{p(D)} \\ &= \arg\max_{\beta} \; p(D \mid \beta) \; p(\beta) \end{aligned} \tag{6}$$

4.1 Gaussian Prior

A Gaussian prior for parameter vectors with zero means and diagonal covariance is specified by a variance parameter σ_i^2 for each dimension $i < d$. The prior density for the full parameter matrix is:

$$p(\beta) = \prod_{c<k-1} \prod_{i<d} \mathsf{Norm}(0, \sigma_i^2)(\beta_{c,i}) \tag{7}$$

where the normal density function with mean 0 and variance σ_i^2 is:

$$\mathsf{Norm}(0, \sigma_i^2)(\beta_{c,i}) = \frac{1}{\sigma_i \sqrt{2\pi}} \; \exp\left(-\frac{\beta_{c,i}^2}{2\sigma_i^2}\right) \tag{8}$$

4.2 Laplace Prior

A Laplace, or double-exponential, prior with zero means and diagonal covariance is specified by a variance parameter σ_i^2 for each dimension $i < d$. The prior density for the full parameter matrix is:

$$p(\beta) = \prod_{c<k-1} \prod_{i<d} \mathsf{Laplace}(0, \sigma_i^2)(\beta_{c,i}) \tag{9}$$

where the Laplace density function with mean 0 and variance σ_i^2 is:

$$\mathsf{Laplace}(0, \sigma_i^2)(\beta_{c,i}) = \frac{\sqrt{2}}{2\sigma_i} \; \exp\left(-\sqrt{2} \; \frac{\mid \beta_{c,i} \mid}{\sigma_i}\right) \tag{10}$$

The Laplace prior has thinner tails than the Gaussian, and thus concentrates posteriors closer to its zero mean than a Gaussian of the same variance.

4.3 Cauchy Prior

A Cauchy, or Lorentz, distribution is a Student-t distribution with one degree of freedom. A Cauchy prior centered at zero with scale $\lambda > 0$ has density:

$$\mathsf{Cauchy}(0, \lambda)(\beta_{c,i}) = \frac{1}{\pi\lambda \left(1 + \left(\frac{\beta_{c,i}}{\lambda}\right)^2\right)} = \frac{1}{\pi} \left(\frac{\lambda}{\beta_{c,i}^2 + \lambda^2}\right) \tag{11}$$

The Cauchy prior does not have a mean or variance; the integrals diverge. The distribution is symmetric, and centered around its median, zero. Larger scale values, like larger variances, stretch the distribution out to have fatter tails. The Cauchy prior has fatter tails than the Gaussian prior and thus has less of a tendency to concentrate posterior probability near the prior median.

4.4 Uniform Prior

The uniform, or uninformative, prior has a constant density:

$$p(\beta) \propto 1 \tag{12}$$

The uniform prior is improper in the sense that the integral of the density diverges:

$$\int_{\mathbb{R}^d} 1 \, dx = \infty \tag{13}$$

The maximum likelihood estimate is the MAP estimate under an (improper) uninformative uniform prior:

$$\begin{aligned} \hat{\beta} &= \arg\max_{\beta} p(D \mid \beta) \, p(\beta) \\ &= \arg\max_{\beta} p(D \mid \beta) \end{aligned} \tag{14}$$

An equivalent formulation is as a Gaussian or Laplace prior with infinite variance. As $\sigma_i^2 \to \infty$, the distributions $\mathsf{Norm}(0, \sigma_i^2)$ and $\mathsf{Laplace}(0, \sigma_i^2)$ flatten toward uniform and the gradient with respect to parameters $\beta_{c,i}$ (see section 6) approaches zero.

5 Error Function

The error function Err_R is indexed by the type of prior R, which must be one of maximum likelihood (ML), Laplace (L1), Gaussian (L2) or Cauchy (t1). The error is the sum of the likelihood error Err_ℓ and the prior error Err_R for prior type R:

$$\begin{aligned} \text{Err}_R(\beta, D, \sigma^2) &= \text{Err}_\ell(D, \beta) + \text{Err}_R(\beta, \sigma^2) \\ &= -\log p(D \mid \beta) - \log p(\beta \mid \sigma^2) \\ &= \sum_{j<n} -\log p(c_j \mid x_j, \beta) + \sum_{i<d} -\log p(\beta_i \mid \sigma_i^2) \\ &= \sum_{j<n} \text{Err}_\ell(c_j, x_j, \beta) + \sum_{i<d} \text{Err}_R(\beta_i, \sigma_i^2) \end{aligned} \tag{15}$$

The last line introduces notation for the contribution to the error due to the likelihood of a single case $\langle x_j, c_j \rangle$ and a single parameter β_i.

6 Error Gradient

The gradient of the error function is the collection of partial derivatives of the error function with respect to the parameters $\beta_{c,i}$:

$$\nabla_{c,i} \, \text{Err}_R(D, \beta, \sigma^2) = \frac{\partial}{\partial \beta_{c,i}} \text{Err}_R(D, \beta, \sigma^2) \tag{16}$$

$$= \frac{\partial}{\partial \beta_{c,i}} \text{Err}_\ell(D, \beta) + \text{Err}_R(\beta, \sigma^2) \tag{17}$$

$$\begin{aligned} &= \frac{\partial}{\partial \beta_{c,i}} \left(-\log p(D \mid \beta) - \log p(\beta \mid \sigma^2) \right) \\ &= -\frac{\partial}{\partial \beta_{c,i}} \log p(D \mid \beta) - \frac{\partial}{\partial \beta_{c,i}} \log p(\beta \mid \sigma^2) \end{aligned}$$

The derivatives distribute through the cases for the data log likelihood:

$$\frac{\partial}{\partial \beta_{c,i}} \log p(D \mid \beta) = \sum_{j<n} \frac{\partial}{\partial \beta_{c,i}} \log p(c_j \mid x_j, \beta) \tag{18}$$

with a single case $\langle x_j, c_j \rangle$ contributing the following to the gradient of the error for output category c at input dimension i:

$$\frac{\partial}{\partial \beta_{c,i}} \log p(c_j \mid x_j, \beta)$$

$$
\begin{aligned}
&= \frac{\partial}{\partial \beta_{c,i}} \log \frac{\exp(\beta_{c_j} \cdot x_j)}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \\
&= \frac{\partial}{\partial \beta_{c,i}} \log \exp(\beta_{c_j} \cdot x_j) - \frac{\partial}{\partial \beta_{c,i}} \log(1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)) \\
&= \frac{\partial}{\partial \beta_{c,i}} (\beta_{c_j} \cdot x_j) - \frac{1}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \frac{\partial}{\partial \beta_{c,i}} (1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)) \\
&= x_{j,i} \mathrm{I}(c = c_j) - \frac{1}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \sum_{c' < k-1} \frac{\partial}{\partial \beta_{c,i}} \exp(\beta_{c'} \cdot x_j) \\
&= x_{j,i} \mathrm{I}(c = c_j) - \frac{1}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j) \frac{\partial}{\partial \beta_{c,i}} (\beta_{c'} \cdot x_j) \\
&= x_{j,i} \mathrm{I}(c = c_j) - \frac{\exp(\beta_{c,i} \cdot x_j)}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} x_{j,i} \\
&= x_{j,i} \mathrm{I}(c = c_j) - p(c \mid x_j, \beta) x_{j,i} \\
&= x_{j,i} (\mathrm{I}(c = c_j) - p(c \mid x_j, \beta))
\end{aligned}
$$

where the indicator function is:

$$
\mathrm{I}(c_j = c) = \begin{cases} 1 & \text{if } c_j = c \\ 0 & \text{if } c_j \neq c \end{cases} \tag{19}
$$

The residual for training example j for outcome category c is the difference between the outcome and the model prediction:

$$
R_{c,j} = \mathrm{I}(c_j = c) - p(c \mid x_j, \beta) \tag{20}
$$

The derivatives also distribute through parameters for the prior:

$$
\begin{aligned}
\frac{\partial}{\partial \beta_{c,i}} \log p(\beta \mid \sigma^2) &= \frac{\partial}{\partial \beta_{c,i}} \sum_{c' < k-1} \sum_{i' < d} \log p(\beta_{c',i'} \mid \sigma_{i'}^2) \qquad (21) \\
&= \frac{\partial}{\partial \beta_{c,i}} \log p(\beta_{c,i} \mid \sigma_i^2)
\end{aligned}
$$

For the Gaussian prior:

$$
\begin{aligned}
\frac{\partial}{\partial \beta_{c,i}} \log p(\beta_{c,i} \mid \sigma_i^2) &= \frac{\partial}{\partial \beta_{c,i}} \log \frac{1}{\sigma_i \sqrt{2\pi}} \exp(-\frac{\beta_{c,i}^2}{2\sigma_i^2}) \qquad (22) \\
&= \frac{\partial}{\partial \beta_{c,i}} \log \frac{1}{\sigma_i \sqrt{2\pi}} + \log \exp(-\frac{\beta_{c,i}^2}{2\sigma_i^2}) \\
&= \frac{\partial}{\partial \beta_{c,i}} - \frac{\beta_{c,i}^2}{2\sigma_i^2} \\
&= -\frac{\beta_{c,i}}{\sigma_i^2}
\end{aligned}
$$

For the Laplace prior:

$$
\begin{aligned}
\frac{\partial}{\partial \beta_{c,i}} \log p(\beta_{c,i} \mid \sigma_i^2) &= \frac{\partial}{\partial \beta_{c,i}} \log \frac{\sqrt{2}}{2\sigma_i} \exp(-\sqrt{2} \frac{\mid \beta_{c,i} \mid}{\sigma_i}) \qquad (23) \\
&= \frac{\partial}{\partial \beta_{c,i}} \log \exp(-\sqrt{2} \frac{\mid \beta_{c,i} \mid}{\sigma_i}) \\
&= \frac{\partial}{\partial \beta_{c,i}} - \sqrt{2} \frac{\mid \beta_{c,i} \mid}{\sigma_i}
\end{aligned}
$$

$$= \begin{cases} -\frac{\sqrt{2}}{\sigma_i} & \text{if } \beta_{c,i} > 0 \\ \frac{\sqrt{2}}{\sigma_i} & \text{if } \beta_{c,i} < 0 \end{cases}$$

For the Cauchy prior:

$$\begin{aligned} \frac{\partial}{\partial \beta_{c,i}} \log p(\beta_{c,i} \mid \lambda_i) &= \frac{\partial}{\partial \beta_{c,i}} \log \left(\frac{1}{\pi} \left(\frac{\lambda_i}{\beta_{c,i}^2 + \lambda_i^2} \right) \right) \\ &= \frac{\partial}{\partial \beta_{c,i}} - \log \pi + \log \lambda_i - \log(\beta_{c,i}^2 + \lambda_i^2) \\ &= -\frac{\partial}{\partial \beta_{c,i}} \log(\beta_{c,i}^2 + \lambda_i^2) \\ &= -\frac{1}{\beta_{c,i}^2 + \lambda_i^2} \ \frac{\partial}{\partial \beta_{c,i}} (\beta_{c,i}^2 + \lambda_i^2) \\ &= -\frac{2\beta_{c,i}}{\beta_{c,i}^2 + \lambda_i^2} \end{aligned} \tag{24}$$

7 Error Hessian

The Hessian of the error function is the symmetric matrix of second derivatives with respect to pairs of parameters. In the multinomial setting with $k-1$ parameter vectors of dimensionality d, the Hessian matrix itself is of dimensionality $((k-1)d) \times ((k-1)d)$ with rows and columns indexed by pairs of categories and dimensions (c, i), with $c \in \{0, \ldots, k-2\}$ and $i < d$.

The value of the Hessian is the sum of the value of the Hessian of the likelihood and the Hessian of the prior. The Hessian for the likliehood function at row (c, i) and column (b, h) is derived by differentiating with respect to parameters $\beta_{c,i}$ and $\beta_{b,h}$.

$$\begin{aligned} \nabla^2_{(c,i),(b,h)} \mathrm{Err}_\ell(D, \beta) &= \frac{\partial^2}{\partial \beta_{c,i} \partial \beta_{b,h}} \mathrm{Err}_\ell(D, \beta) \\ &= \frac{\partial}{\partial \beta_{b,h}} \nabla_{c,i} \, \mathrm{Err}_\ell(D, \beta) \\ &= \sum_{j<n} \frac{\partial}{\partial \beta_{b,h}} \nabla_{c,i} \, \mathrm{Err}_\ell(x_j, c_j, \beta) \\ &= \sum_{j<n} \frac{\partial}{\partial \beta_{b,h}} x_{j,i} (\mathrm{I}(c = c_j) - p(c \mid x_j, \beta)) \end{aligned} \tag{25}$$

where:

$$\begin{aligned} & \frac{\partial}{\partial \beta_{b,h}} x_{j,i} (\mathrm{I}(c = c_j) - p(c \mid x_j, \beta)) \\ = \ & \frac{\partial}{\partial \beta_{b,h}} - x_{j,i} \, p(c \mid x_j, \beta) \\ = \ & -x_{j,i} \frac{\partial}{\partial \beta_{b,h}} \ \frac{\exp(\beta_c \cdot x_j)}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \\ = \ & -x_{j,i} \left(\frac{1}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \frac{\partial}{\partial \beta_{b,h}} \exp(\beta_c \cdot x_j) \right. \\ & \qquad \left. + \exp(\beta_c \cdot x_j) \frac{\partial}{\partial \beta_{b,h}} \frac{1}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \right) \\ = \ & -x_{j,i} \left(\frac{\exp(\beta_c \cdot x_j)}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \frac{\partial}{\partial \beta_{b,h}} \beta_c \cdot x_j \right. \end{aligned}$$

$$+ \exp(\beta_c \cdot x_j) \left(\frac{1}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \right)^2 \frac{\partial}{\partial \beta_{b,h}} \left(1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j) \right) \Bigg)$$

$$\begin{aligned}
= \; & -x_{j,i} \Bigg(p(c \mid x_j, \beta) \mathrm{I}(c = b) x_{j,h} \\
& \quad + p(c \mid x_j, \beta) \frac{1}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \frac{\partial}{\partial \beta_{b,h}} \exp(\beta_b \cdot x_j) \Bigg) \\
= \; & -x_{j,i} \, p(c \mid x_j, \beta) \left(\mathrm{I}(c = b) x_{j,h} + \frac{\exp(\beta_b \cdot x_j)}{1 + \sum_{c' < k-1} \exp(\beta_{c'} \cdot x_j)} \frac{\partial}{\partial \beta_{b,h}} \beta_b \cdot x_j \right) \\
= \; & -x_{j,i} \, p(c \mid x_j, \beta) \left(\mathrm{I}(c = b) x_{j,h} + p(b \mid x_j, \beta) x_{j,h} \right) \\
= \; & -x_{j,i} \, x_{j,h} \, p(c \mid x_j, \beta) \left(\mathrm{I}(c = b) + p(b \mid x_j, \beta) \right)
\end{aligned}$$

The priors contribute their own terms to the Hessian, but the only non-zero contributions are on the diagonals for Cauchy and Gaussian priors. The maximum likelihood prior is constant, and thus has a zero gradient and zero Hessian.

For the Laplace prior:

$$\begin{aligned}
\nabla^2_{(c,i),(b,h)} \mathrm{Err}_{\mathrm{L1}}(\beta \mid \sigma^2) &= \frac{\partial}{\partial \beta_{b,h} \, \partial \beta_{c,i}} \mathrm{Err}_{\mathrm{L1}}(\beta \mid \sigma^2) \\
&= \frac{\partial}{\partial \beta_{b,h}} - \frac{\mathrm{signum}(\beta_{c,i})}{\sigma_i^2} \\
&= 0
\end{aligned} \tag{26}$$

For the Gaussian prior:

$$\begin{aligned}
\nabla^2_{(c,i),(b,h)} \mathrm{Err}_{\mathrm{L2}}(\beta \mid \sigma^2) &= \frac{\partial}{\partial \beta_{b,h} \, \partial \beta_{c,i}} \mathrm{Err}_{\mathrm{L2}}(\beta \mid \sigma^2) \\
&= \frac{\partial}{\partial \beta_{b,h}} - \frac{\beta_{c,i}}{\sigma_i^2} \\
&= -\mathrm{I}((c,i) = (b,h)) \frac{1}{\sigma_i^2}
\end{aligned} \tag{27}$$

For the Cauchy prior:

$$\begin{aligned}
& \nabla^2_{(c,i),(b,h)} \mathrm{Err}_{\mathrm{t1}}(\beta \mid \lambda^2) \\
= \; & \frac{\partial}{\partial \beta_{b,h} \, \partial \beta_{c,i}} \mathrm{Err}_{\mathrm{t1}}(\beta \mid \lambda^2) \\
= \; & \frac{\partial}{\partial \beta_{b,h}} - \frac{2\beta_{c,i}}{\beta_{c,i}^2 + \lambda_i^2} \\
= \; & -\mathrm{I}((c,i) = (b,h)) \frac{\partial}{\partial \beta_{c,i}} \frac{2\beta_{c,i}}{\beta_{c,i}^2 + \lambda_i^2} \\
= \; & -\mathrm{I}((c,i) = (b,h)) \left(2\beta_{c,i} \frac{\partial}{\partial_{c,i}} \frac{1}{\beta_{c,i}^2 + \lambda_i^2} + \frac{1}{\beta_{c,i}^2 + \lambda_i^2} \frac{\partial}{\partial_{c,i}} 2\beta_{c,i} \right) \\
= \; & -\mathrm{I}((c,i) = (b,h)) \left(2\beta_{c,i} \left(\frac{1}{\beta_{c,i}^2 + \lambda_i^2} \right)^2 \frac{\partial}{\partial_{c,i}} (\beta_{c,i}^2 + \lambda_i^2) + \frac{2\beta_{c,i}}{\beta_{c,i}^2 + \lambda_i^2} \right) \\
= \; & -\mathrm{I}((c,i) = (b,h)) \left(\left(\frac{2\beta_{c,i}}{\beta_{c,i}^2 + \lambda_i^2} \right)^2 + \frac{2\beta_{c,i}}{\beta_{c,i}^2 + \lambda_i^2} \right)
\end{aligned} \tag{28}$$

8 Concavity and Existence of Solution

For any of the priors, uninformative, Cauchy, Gaussian or Laplace, the error function is a concave function of the feature vectors, so that for all vectors $\beta, \beta' \in \mathbb{R}^d$ and all λ such that

$0 \leq \lambda \leq 1$:

$$\text{Err}_R(D, (\lambda\beta + (1-\lambda)\beta'), \sigma^2) \geq \lambda\text{Err}_R(D, \beta, \sigma^2) + (1-\lambda)\text{Err}_R(D, \beta', \sigma^2) \tag{29}$$

To show that a function is concave, it suffices to show that its negative Hessian is positive semi-definite; this generalizes the negative second derivative condition required for a one-dimensional function to be concave. A square matrix M is positive semi-definite if and only if for every (column) vector x:

$$x^\top M x \geq 0 \tag{30}$$

which evaluating the product, amounts to:

$$\sum_i \sum_j x_i \, x_j \, M_{i,j} \geq 0 \tag{31}$$

For the maximum likelihood error, this condition amounts to requiring for all vectors z of dimensionality $c(k-1)$:

$$-z^\top \nabla^2 \text{Err}_\ell(D, \beta) \, z \geq 0 \tag{32}$$

which is equivalent to:

$$-\sum_{(c,i)} \sum_{(b,h)} z_{c,i} \, z_{b,h} \, \nabla^2_{(c,i),(b,h)} \text{Err}_\ell(D, \beta) \geq 0 \tag{33}$$

which plugging in the Hessian's value, yields:

$$\sum_{(c,i)} \sum_{(b,h)} \sum_{j<n} z_{c,i} \, z_{b,h} \, x_{j,i} \, x_{j,h} \, p(c \mid x_j, \beta) \; (I(c=b) + p(b \mid x_j, \beta)) \geq 0 \tag{34}$$

For the Cauchy and Gaussian priors with finite positive scale or variance, the gradient of the error function is strictly concave, which strengthens this requirement for any pair of unequal vectors β, β' such that $\beta \neq \beta'$ and any λ such that $0 < \lambda < 1$ to:

$$\text{Err}_R(D, (\lambda\beta + (1-\lambda)\beta'), \sigma^2) > \lambda\text{Err}_R(D, \beta, \sigma^2) + (1-\lambda)\text{Err}_R(D, \beta', \sigma^2) \tag{35}$$

To show that a function is strictly concave, it suffices to show that its negative Hessian is positive definite. A square matrix M is positive definite if and only if for every non-zero vector x:

$$x^\top \, M \, x > 0 \tag{36}$$

With a Cauchy or Gaussian prior, the Hessian is positive definite, and hence the combined error function is positive definite.

With a concave error function, there exists a MAP parameter estimate $\hat{\beta}$ exists where the gradient evaluated at $\hat{\beta}$ is zero:

$$\nabla \, \text{Err}_R(D, \beta, \sigma^2)(\hat{\beta}) = \frac{\partial}{\partial \beta} \text{Err}_R(D, \beta, \sigma^2)(\hat{\beta}) = 0 \tag{37}$$

In the maximum likelihood setting or with uninformative priors for some dimensions in other cases, there may not be a unique solution. With strictly concave error functions, as arises with finite priors, there will be a unique MAP estimate.

9 MAP Estimate Expectation Constraints

In maximum likelihood estimation, the estimated parameters $\hat{\beta}$ are the zero of the gradient at every category $c < k-1$ and dimension $i < d$:

$$\begin{aligned} (\nabla_{c,i} \text{Err}_\ell(D, \beta)) \, (\hat{\beta}_{c,i}) &= \left(\frac{\partial}{\partial \beta_{c,i}} \text{Err}_\ell(D, \beta) \right) (\hat{\beta}_{c,i}) \\ &= \sum_{j<n} x_{j,i} \, \left(\text{I}(c_j = c) - p(c \mid x_j, \hat{\beta}) \right) \\ &= 0 \end{aligned} \tag{38}$$

and therefore $\hat{\beta}$ satisfies:

$$\sum_{j<n} x_{j,i}\ p(c \mid x_j, \hat{\beta}) = \sum_{j<n} x_{j,i}\, \mathrm{I}(c_j = c) \tag{39}$$

In words, the expectation of the count of a feature i with a model parameterized by $\hat{\beta}$ over all of the training cases is equal to the empirical count of the feature in the training data.

With informative priors, the gradient of the full error function must be zero at every dimension of every category parameter vector:

$$\begin{aligned} \left(\nabla_{c,i}\mathrm{Err}_R(D, \beta, \sigma^2)\right)(\hat{\beta}_{c,i}) &= \left(\nabla_{c,i}\mathrm{Err}_\ell(D, \beta) + \nabla_{c,i}\mathrm{Err}_R(\beta, \sigma^2)\right)(\hat{\beta}_{c,i}) \\ &= 0 \end{aligned} \tag{40}$$

which substituting in the gradient of the likelihood-based error reduces to:

$$\begin{aligned} \sum_{j<n} x_{j,i}\ p(c \mid x_j, \hat{\beta}) &= \sum_{j<n} x_{j,i}\, \mathrm{I}(c_j = c) - \left(\nabla_{c,i}\mathrm{Err}_R(\beta, \sigma^2)\right)(\hat{\beta}_{c,i}) \\ &= \sum_{j<n} x_{j,i}\, \mathrm{I}(c_j = c) - \nabla_{c,i}\mathrm{Err}_R(\beta_{c,i}, \sigma_i^2)(\hat{\beta}_{c,i}) \end{aligned} \tag{41}$$

In this case, the feature expectations are equal to the empirical feature counts discounted by the prior regularization. For instance, for the Gaussian prior, this amounts to the expectations matching the empirical counts discounted by the prior:

$$\sum_{j<n} x_{j,i}\ p(c \mid x_j, \hat{\beta}) = \left(\sum_{j<n} x_{j,i}\, \mathrm{I}(c_j = c)\right) - \frac{\hat{\beta}_{c,i}}{\sigma_i^2} \tag{42}$$

10 Stochastic Gradient Descent

Gradient descent algorithms initialize the parameter vectors and then descend along the gradient of the error function until reaching minimum error.

10.1 Batch versus Online Algorithms

Traditional gradient descent methods like conjugate gradient compute the gradient of the entire corpus and then do a line search along that gradient to find the optimal learning rate (distance) for an update along the gradient. The line searches are expensive because they require log likelihood computations over the entire corpus which update the parameters along the gradient, which is typically non-zero in every dimension over the entire corpus. And the directions searched zig-zag because after the update along the gradient, the next update must be conjugate in the sense of being orthogonal as a vector to the previous update. Each corpus gradient is unlikely to point at the MAP solution, so a zig-zag path is charted toward an eventual solution.

Stochastic search methods such as stochastic gradient descent takes steps in the direction of the contribution of a single training case $\langle x_j, c_j \rangle$ and fraction of the prior to the gradient. Incremental updates allow each iteration through the data to steer more directly in the direction of the global minimum by adjusting direction after each training instance, resulting in a finer-grained zig-zag toward the global minimum error.

There are two advantages of stochastic approaches. First, they only require the parameters and a single training example to be stored in memory. Algorithms with this property are said to be online algorithms. In order to converge, most online algorithms require multiple passes through the data. The algorithm itself doesn't change if it's run over the same data more than once or run over fresh data.

With a huge amount of data, it is typically better to use an online algorithm that visits each data element once or a few times rather than sampling a smaller batch that can be processed with batch algorithms.

Intermediate algorithms that take batches of larger than one training example but smaller than the whole corpus are also popular.

10.2 Code Listing

A full listing of an SGD optimizer is provided as Algorithm 1 on page 19. The error function Err_R is indexed by the prior, with ml being the uninformative prior, L_2 being the Gaussian prior, L_1 being the Laplace prior, and t_1 being the Cauchy prior.

10.3 Caching

For efficiency, values in the loops such as the exponentiated linear predictors and probability estimates should be stored rather than recomputed.

10.4 Learning Rate

With an appropriate choice of learning rate, stochastic gradient descent is guaranteed to converge. The conditions on the learning rate η_e, the learning rate in epoch e, convergence requires the sum of all learning rates to diverge:

$$\sum_{e<\infty} \eta_e = \infty \tag{43}$$

and the sum of all squared learning rates to converge:

$$\sum_{e<\infty} \eta_e^2 < \infty \tag{44}$$

Together, the first equation says the learning rate moves fast enough and far enough, and the second equation says it eventually settles down to a fine enough grain to get within ϵ of the optimal solution. In practice, learning for large-scale problems is typically cut-off at only a few decimal places of precision.

It is possible to use an adaptive learning rate that depends on features of the data and the algorithm's progress. For instance, a sample of the data points may be taken and a (typically coarse) line search used to evaluate an optimal stochastic learning rate for that sample. This empirically estimated rate may then be applied to subsequent examples. In the limit of samples equal to the entire data set, this reduces to conjugate gradient search.

Rather than a line search, rates fractionally above or below the current rate may be explored and the learning rate adjusted accordingly. Momentum terms are often added to the learning rate on a dimension by dimension basis, making the learning rate a sum of the last learning rate and the current learning rate. Momentum has the possibility to wildly overshoot, so it is typically combined with evaluations that will reduce the learning rate if the current learning rate increases the likelihood error.

For any of these algorithms to converge, they must first get close to a solution then gradually lower the learning rate until convergence.

10.5 On-the-Fly Log Likelihood Calculation

Algorithm 1 for SGD calculates the log likelihood of the corpus at the end of an epoch after the parameters have been updated.

The log likelihood of the corpus plus log prior may be computed on the fly rather than in a standalone loop. In particular, the value of $p(c_j \mid x_j, \beta)$ computed in the inner loop may be used. The contribution of the prior can be added at the end of the loop.

With parameter vectors changing at each training case, on-the-fly likelihood calculations don't reflect the contribution of training on the current example. In practice, training on the example being evaluated and the overall stochasticity of the algorithm tends to improve the model throughout an epoch, so that the on-the-fly calculation is an underestimate of the log likelihood in the final parameterization for an epoch.

10.6 Clipping Regularization at Zero

The regularization may be clipped so that subtracting the regularization gradient will never change the sign of a parameter. The original inner loop:

$$\beta_c \leftarrow \beta_c + \eta_e \nabla_c \, \text{Err}_\ell(x_j, c_j, \beta) + \eta_e \nabla_c \, \text{Err}_R(\beta_c, \sigma^2) \tag{45}$$

may be modified to increment the likelihood gradient and then clip the regularization increment:

$$\begin{array}{l}\beta_c \leftarrow \beta_c + \eta_e \nabla_c \operatorname{Err}_\ell(x_j, c_j, \beta) \\ \textbf{if } (\beta_c < 0) \textbf{ then} \\ \quad \beta_c \leftarrow \min(0, \beta_c + \eta_e \nabla_c \operatorname{Err}_R(\beta_c, \sigma^2)) \\ \textbf{else} \\ \quad \beta_c \leftarrow \max(0, \beta_c + \eta_e \nabla_c \operatorname{Err}_R(\beta_c, \sigma^2))\end{array} \tag{46}$$

10.7 Sparseness of Updates

The inner loop is the stochastic update of the parameters along the contribution to the gradient of a single training case $\langle x_j, c_j \rangle$:

$$\beta_c \leftarrow \beta_c + \eta_e \ \nabla_c \operatorname{Err}_R(x_j, c_j, \beta, \sigma^2) \tag{47}$$

where the gradient is expanded as:

$$\nabla_c \operatorname{Err}_R(x_j, c_j, \beta, \sigma^2) = x_j \cdot (\mathrm{I}(c = c_j) - p(c \mid x_j, \beta)) + \frac{\nabla_c \operatorname{Err}_R(\beta, \sigma^2)}{n} \tag{48}$$

The update from the gradient of the log likelihood only touches dimensions for which the input vector x_j is non-zero. Thus if the inputs are sparse, the gradient update for log likelihood is sparse.

In the maximum likelihood setting, there is no gradient for the prior and thus the updates remain sparse. But in the case of regularization, the gradient of the log prior density is non-zero at every non-zero parameter dimension.

11 Lazy Regularized Stochastic Gradient Descent

A simple solution to deal with the density of regularization updates is to batch them up and only do them so often. With a batch size larger than the feature density, this will actually be the most efficient approach. The downside is the loss in stochasticity of the search.

Even though the parameter vectors are regularized at each step, only the non-zero dimensions of an input x_j need be accessed. The regularizations may thus be computed in a lazy fashion, just before they are needed.

The vector u of dimensionality d stores in u_i the last epoch in which dimension i's parameter value was shrunk by regularization. Then, when an input x_j for which $x_{j,i} \neq 0$ arises, the parameters $\beta_{c,i}$ for all categories c are caught up with regularization scheme R using:

$$\frac{u_{c,i} - q}{n} \ \nabla_{c,i} \operatorname{Err}_R(\beta_{c,i}, \sigma^2) \tag{49}$$

The standard regularized stochastic gradient descent algorithm is not equivalent to regularizing all at once. Suppose a coefficient starts at value $\beta_{c,i}$. After one step of stochastic regularization using a Gaussian prior with variance σ^2, assuming for simplicity a learning rate of $\eta_i = 1$, the value will be:

$$\beta^{(1)} = \beta^{(0)} - \frac{1}{n} \ \frac{\beta^{(0)}}{\sigma^2} \tag{50}$$

After two updates:

$$\beta^{(2)} = \beta^{(1)} - \frac{1}{n} \ \frac{\beta^{(1)}}{\sigma^2} \tag{51}$$

and in general, at the k-th step:

$$\beta^{(k)} = \beta^{(0)} \left(1 - \left(\frac{1}{n\sigma^2}\right)^k\right) \tag{52}$$

Thus a single update regularizing at a rate of k/n does not produce the same result as k updates at the rate of $1/n$. In the algorithm, there is no nonlinear adjustment for batch size. The lower amount of stochasticity simply approaches the non-stochastic regularization, which should actually be more accurate than the stochastic approximation.

For the Laplace prior, the gradient does not depend on the current value of β, so a stochastic update of size k/n is the same as k stochastic updates of batch size $1/n$:

$$\beta^{(k)} = \beta^{(0)} - \frac{k\sqrt{2}}{n\sigma} \tag{53}$$

Even clipping to zero has the same effect with the Laplace prior.

Appendices

A Separability and Divergence

The training data $\langle x_j, c_j \rangle_{j<n}$ for a two-category problem ($c_j \in \{0, 1\}$) is linearly separable if there exists a parameter vector β such that $\beta \cdot x_j > 0$ if and only if $c_j = 0$. In this case, the parameter vector β leads to a perfect first-best classifier for the training data. But the problem is that there is no maximum likelihood solution, because scaling β to be larger strictly increases the log likelihood:

$$p(0 \mid x_j, \beta) = \frac{\exp(\beta \cdot x_j)}{1 + \exp(\beta \cdot x_j)} \tag{54}$$

If $\lambda > 1$, then $\lambda\beta$ produces a higher likelihood estimate than β:

$$\frac{\exp((\lambda\beta) \cdot x_j)}{1 + \exp((\lambda\beta) \cdot x_j)} > \frac{\exp(\beta \cdot x_j)}{1 + \exp(\beta \cdot x_j)} \tag{55}$$

with

$$\lim_{\lambda \to \infty} \frac{\exp(\lambda\beta \cdot x_j)}{1 + \exp(\lambda\beta) \cdot x_j)} = 1 \tag{56}$$

So there is no maximum likelihood solution; the larger the scaling λ, the closer the likelihood gets to 1.

With an informative prior of finite positive variance or scale, the prior will offset the tendency of the parameter vector to grow without bound and a unique solution will exist that balances the prior and the likelihood; eventually the gain from scaling β will be offset by the cost of scaling β.

If the convergence conditions for the SGD estimation algorithm were defined over the parameters, the algorithm will not converge with linearly separable training data. But with convergence defined in terms of corpus log likelihood improvements dropping below a threshold, the algorithm will terminate even in the face of separable training data.

B Non-Linearity and Interactions

Some classification problems which are not (approximately) linear in the input features may be made (approximately) linear by introducing non-linear functions of the input features. Typically these will be quadratic or logarithmic. For instance, a dimension i whose response is quadratic may be expanded so that input vectors contain x_i and another dimension x_j whose value is $x_j = x_i^2$.

Although logistic regression remains linear in the expanded feature space, it is non-linear in the original space.

Logistic regression treats the dimensions in the input independently, but it is sometimes helpful to model interactions of features x_i and x_j. This is typically done by introducing a new feature $x_k = x_i x_j$ whose value is the product of the original features. In the binary feature case, the resulting interaction features will be binary if coded as 0 and 1.

C Binomial Logistic Regression

The logit function is:

$$\text{logit}(b) = \log \frac{b}{1 - b} \tag{57}$$

The inverse logit function is the logistic sigmoid function:

$$\text{logit}^{-1}(a) = \frac{1}{1+\exp(-a)} = \frac{\exp(a)}{1+\exp(a)} \tag{58}$$

With two categories, there is only a single parameter vector β_0. Logistic regression is so-called because in this case, the logit (log odds) of the probability is a linear function of the input x:

$$\begin{aligned} \beta_0 \cdot x &= \text{logit}(p(0 \mid x)) \\ &= \log \frac{p(0 \mid x)}{1 - p(0 \mid x)} \\ &= \log \frac{p(0 \mid x)}{p(1 \mid x)} \end{aligned} \tag{59}$$

The probability of outcome 0 is the logistic transform of the linear predictor:

$$\begin{aligned} p(0 \mid x) &= \text{logit}^{-1}(\beta_0 \cdot x) \\ &= \frac{\exp(\beta_0 \cdot x)}{\exp(\beta_0 \cdot x) + 1} \end{aligned} \tag{60}$$

D k Versus $(k-1)$ Parameter Vectors

Some presentations of multinomial logistic regression assume k parameter vectors rather than $k-1$. This simplifies the definition of probability to:

$$p(c \mid x, \beta) = \frac{\exp(\beta_c \cdot x)}{Z_x} \tag{61}$$

for all $c < k$ with partition function:

$$Z_x = \sum_{c<k} \exp(\beta_c \cdot x) \tag{62}$$

If $\beta_{k-1} = 0$, this definition reduces to equations (1) and (3), because:

$$\exp(0 \cdot x) = \exp(0) = 1 \tag{63}$$

The addition of an extra parameter vector β_{k-1} does not increase the class of distributions that can be represented. For simplicity, consider the binary case ($k = 2$) with two parameter vectors β_0 and β_1:

$$\begin{aligned} & \frac{\exp(\beta_0 \cdot x)}{\exp(\beta_0 \cdot x) + \exp(\beta_1 \cdot x)} \\ = \; & \frac{\exp(-\beta_1 \cdot x)\exp(\beta_0 \cdot x)}{\exp(-\beta_1 \cdot x)(\exp(\beta_0 \cdot x) + \exp(\beta_1 \cdot x))} \\ = \; & \frac{\exp((\beta_0 - \beta_1) \cdot x)}{\exp((\beta_0 - \beta_1) \cdot x) + \exp((\beta_1 - \beta_1) \cdot x)} \\ = \; & \frac{\exp((\beta_0 - \beta_1) \cdot x)}{\exp((\beta_0 - \beta_1) \cdot x) + 1} \end{aligned} \tag{64}$$

The resulting distribution is equivalent to the model with a single parameter vector $(\beta_0 - \beta_1)$. Any linear interpolation between $\langle \beta_0 - \beta_1, 0 \rangle$ and $\langle 0, \beta_1 - \beta_0 \rangle$ results in the same classification distribution.

The above shows that the parameters are not uniquely identified in the k-parameter vector maximum likelihood setting. With proper Gaussian or Laplace priors, a unique solution is identified. For instance, with a Gaussian prior σ^2 for all dimensions, the MAP solution will have the length of β_0 and β_1 being equal.

E Bayes Optimal Decision Boundary

A first-best classifier predicts a single category b for an input x. If the real category is c, the 0/1 loss for this case is:

$$\text{Err}_{0/1}(c, b) = 1 - \text{I}(c = b) = \begin{cases} 0 & \text{if } c = b \\ 1 & \text{if } c \neq b \end{cases} \tag{65}$$

The lowest expected 0/1 loss is called the Bayes rate and is achieved by choosing the most likely category:

$$\text{classify}(x) = \arg\max_c \; p(c \mid x, \beta) \tag{66}$$

Taking $\beta_{k-1} = 0$ for convenience (see appendix D), this reduces to taking the category with the largest linear predictor:

$$\begin{aligned} \arg\max_c \; p(c \mid x, \beta) &= \arg\max_c \; \frac{\exp(\beta_c \cdot x)}{Z_x} \\ &= \arg\max_c \; \exp(\beta_c \cdot x) \\ &= \arg\max_c \beta_c \cdot x \end{aligned} \tag{67}$$

In the two-category case, where there is a single parameter vector β_0, the Bayes optimal decision rule reduces to:

$$\text{classify}(x) = \begin{cases} 0 & \text{if } \beta_0 \cdot x > 0 \\ 1 & \text{if } \beta_0 \cdot x \leq 0 \end{cases} \tag{68}$$

This is the same binary decision rule used for linear discriminant analysis (LDA), perceptrons, and support vector machines (SVM), the difference being in how the parameters β are estimated.

F Binary Features

In some data, all features are binary in the sense of taking on only two values, on and off. Conventionally, on is coded as 1 and off as 0. If most predictors are off, the resulting input vectors are sparse.

Given the linearity of the classifier, any two distinct numbers will work, and they may be in any order and on any scale. If the order is reversed, the signs of the parameters will reverse to compensate. If the scale is changed, the scale of the parameters will adjust to compensate. See appendix K for a discussion of the effects of scaling and offset on priors.

G Generalized Feature Functions

Some presentations of logistic regression assume arbitrary inputs $x \in X$ (where no assumptions are made about X) along with a feature function $\phi : X \to \mathbb{R}^d$ mapping elements $x \in X$ to d-dimensional real vectors. This paper takes the inputs to be vectors directly without modeling the feature function.

An even more general feature function $\phi : (X \times \{0, \ldots, k-1\}) \to \mathbb{R}^d$ is sometimes used, which allows the features to vary by category and input. For example, $\phi(x, c)$ might not use the same features as $\phi(x, c')$. In this case, every category is required to have its own parameter vector, and the probability of a category is defined by:

$$p(c \mid x) \propto \exp(\beta_c \cdot \phi(x, c)) \tag{69}$$

Modifying features on a per-category basis amounts to setting hard priors on some features in some categories. If the set of features is taken to be the set of all possible features in all categories, then features that are turned off in some categories may be modeled by setting their prior mean and variance to zero, which forces the posterior to also be zero. In implementation terms, this requires priors to vary by category, which is a simple generalization of the algorithms presented here.

H Exponential Basis Presentation

Some presentations of logistic regression use an alternative parameterization of models γ where:

$$\gamma_{c,i} = \exp(\beta_{c,i}) \tag{70}$$

Finding optimal $\gamma_{c,i} \geq 0$ is equivalent to finding optimal $\beta_{c,i}$, because:

$$\begin{aligned} \prod_{i<d} \gamma_{c,i}^{x_i} &= \prod_{i<d} \exp(\beta_{c,i})^{x_i} \\ &= \prod_{i<d} \exp(\beta_{c,i} x_i) \\ &= \exp(\sum_{i<d} \beta_{c,i} x_i) \\ &= \exp(\beta_c \cdot x) \end{aligned} \tag{71}$$

Changing the basis does not affect maximum likelihood estimation, but it does change the scale of priors. The same priors may be used if they are rescaled for the transform.

An exponential basis is convenient if features are binary and coded as 0 and 1:

$$\prod_{i<d} \gamma_{c,i}^{x_i} = \prod_{x_i \neq 0} \gamma_{c,i} \tag{72}$$

I Maximum Entropy

The maximum entropy principle is a model selection technique that favors the distribution with the maximum entropy among all distributions that satisfy some set of constraints. Typically, these are feature expectation constraints such as those listed in equation (39). In general, $p(c \mid x_j, \hat{\beta})$ is the distribution with the maximum entropy that satisfies the constraints in equation (39).

The entropy of a discrete distribution $p(c)$ is defined to be the expectation under $p(c)$ of the log probability $\log p(c)$:

$$\mathrm{H}(p(c)) = \sum_b p(b) \log p(b) \tag{73}$$

If the log is base e, the units are called nats; if it is base 2, the units are called bits. The base won't matter here, as it only introduces a multiplicative constant.

For logistic regression, the entropy we care about is the entropy of the category distributions over all training examples:

$$\mathrm{H}(\beta) = \sum_{j<n} \mathrm{H}(p(c \mid x_j, \beta)) \tag{74}$$

With logistic regression, the maximum likelihood solution $\hat{\beta}$ is such that it maximizes $\mathrm{H}(\beta)$:

$$\begin{aligned} \hat{\beta} &= \arg\max_\beta \mathrm{H}(\beta) \\ &= \arg\max_\beta \mathrm{H}(p(c \mid x_j, \beta)) \\ &= \arg\max_\beta \sum_b p(b \mid x_j, \beta) \log p(b \mid x_j, \beta) \end{aligned} \tag{75}$$

The maximum entropy principle may be restated in terms of Kullback-Leibler divergence of the implied category distributions relative to the prior category distributions. The Kullback-Leibler divergence of $p'(c)$ from $p(c)$ is:

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}(p(c) \parallel p'(c)) &= \sum_b p(b) \log \frac{p(b)}{p'(b)} \\ &= \sum_b p(b) \left(\log p(b) - \log p'(b)\right) \end{aligned} \tag{76}$$

$$
\begin{aligned}
&= \sum_b p(b) \log p(b) - \sum_b p(b) \log p'(b) \\
&= H(p(c)) + H(p(c), p'(c))
\end{aligned}
$$

where the cross-entropy is defined to be the negative expectation under $p(b)$ of $\log p'(b)$:

$$
H(p(c), p'(c)) = -\sum_b p(b) \log p'(b) \tag{77}
$$

The expansion of the definition shows that the KL-divergence of $p'(c)$ from $p(c)$ is equal to the entropy of $p(c)$ minus the cross-entropy of $p'(c)$ under $p(c)$. Gibbs' inequality states:

$$
\mathrm{D}_{\mathrm{KL}}(p(c) \parallel p'(c)) \geq 0 \tag{78}
$$

with equality if and only if $p(c) = p'(c)$. Thus the cross-entropy is always greater than or equal to the (self-) entropy.

The maximum likelihood estimate $\hat{\beta}$ is not only maximum entropy, but also minimizes KL-divergence from a uniform category distribution:

$$
\mathrm{Uniform}(k)(c) = \frac{1}{k} \tag{79}
$$

The uniform distribution contributes only a constant to the denominator of the log equal to its entropy:

$$
\begin{aligned}
\mathrm{D}_{\mathrm{KL}}(p(c) \parallel \mathrm{Uniform}(k)(c)) &= \sum_b p(b) \log \frac{p(b)}{\mathrm{Uniform}(k)(c)} \qquad (80) \\
&= \sum_b p(b) \log \frac{p(b)}{1/k} \\
&= \sum_b p(b) \left(\log p(b) + \log k\right) \\
&= \sum_b p(b) \log p(b) + \sum_b p(b) \log k \\
&= \log k + \mathrm{H}(p(c))
\end{aligned}
$$

Thus the KL-divergence is minimized at the same $\hat{\beta}$ as the entropy.

Logistic regression with priors may be modeled by an appropriate choice of baseline distribution against which to measure divergence.

J Kernelization

In the stochastic gradient algorithm, the parameter vectors β_c are initialized to zero and updated by:

$$
\beta_c \leftarrow \beta_c + \eta_e x_j (\mathrm{I}(c = c_j) - p(c \mid x_j, \beta)) + \eta_e \nabla_c \, \mathrm{Err}_R(\beta, \sigma^2) \tag{81}
$$

The gradient of the log likelihood contributes a linear multiple of an input vector x_j. Ignoring the gradient of the prior, the maximum likelihood estimate $\hat{\beta}_c$ is a linear combination of the the training inputs x_j:

$$
\hat{\beta}_c = \sum_{j<n} \alpha_{c,j} x_j \tag{82}
$$

where:

$$
\alpha_{c,j} = \sum_e \eta_e (\mathrm{I}(c = c_j) - p(c \mid x_j, \beta^{e,j})) \tag{83}
$$

with $\beta^{e,j}$ the values of β in epoch e before evaluating training case j.

The training data vectors x_j which make non-zero contributions to the trained parameter vectors are called support vectors. Without some kind of thresholding, logistic regression will use every training data vector for support. The more step-like loss functions used for perceptrons

(0/1 loss) and support vector machines (hinge loss), typically lead to a sparse set of support vectors.

With the Laplace prior, the reuslt is still linear in the inputs if there is an intercept feature (see appendix K).

The kernelized version of the optimization problem is to directly optimize the $\alpha_{c,j}$ parameters. The kernel optimization problem can be stated purely in terms of inner products of the training data points $x_j \cdot x_m$, because:

$$\begin{aligned} \hat{\beta}_c \cdot x_m &= \left(\sum_{j<n} \alpha_{c,j} \, x_j \right) \cdot x_m \\ &= \sum_{j<n} \alpha_{c,j} (x_j \cdot x_m) \end{aligned} \tag{84}$$

With this kernelized form of optimization, we can introduce an expansion or transformation of our basis using an arbitrary function on vectors $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^f$. The kernel function then replaces the dot-product in the algorithm:

$$K(x_j, x_m) = \phi(x_j) \cdot \phi(x_m) \tag{85}$$

The "kernel trick" allows us to compute the kernel function implicitly without transforming the bases and computing the potentially very large dot product $\phi(x_j) \cdot \phi(x_m)$. It is even possible to map input vectors to infinite-dimensional representations in some cases. In practice, users tend to stick to fairly simple kernels, such as the (inhomogeneous) polynomial kernels of degree $r \geq 1$:

$$K(x_j, x_m) = (x_j \cdot x_m + 1)^r \tag{86}$$

or the spherical Gaussian radial basis kernels with variance σ^2:

$$K(x_j, x_m) = \exp(-\frac{\sum_{i<d} (x_{j,i} - x_{m,i})^2}{2\sigma^2}) \tag{87}$$

Mercer's theorem tells us that for both of these cases (and many more), there exists a function ϕ such that $K(x_j, x_m)$ may be defined according to equation (85).

K Intercept and Variance Normalization

It is common to set aside one dimension in the d-dimensional inputs as an intercept feature. By convention, the intercept feature is the first element of an input vector, and its value is set to 1:

$$x_{j,0} = 1 \tag{88}$$

With this feature set to a constant value, the parameters $\beta_{c,0}$ are intercepts in the sense that:

$$\beta_c \cdot x = \beta_{c,0} + \sum_{0<i<d} \beta_{c,i} \, x_i \tag{89}$$

It is also common to center inputs to have zero means and scale them to have identical (typically unit) variance. This is accomplished by replacing elements with their z scores. For dimension $i < d$, let $\hat{\mu}_i$ be the mean of the $x_{j,i}$ and $\hat{\sigma}_i^2$ be the variance of the $x_{j,i}$:

$$\hat{\mu}_i = \frac{1}{n} \sum_{j<n} x_{i,j} \tag{90}$$

$$\hat{\sigma}_i^2 = \frac{1}{n} \sum_{j<n} (x_{i,j} - \hat{\mu}_i)^2 \tag{91}$$

The z-transform then maps:

$$z(x_{j,i}) = \frac{x_{j,i} - \hat{\mu}_i}{\hat{\sigma}_i} = \frac{x_{j,i}}{\hat{\sigma}_i} - \frac{\hat{\mu}_i}{\hat{\sigma}_i} \tag{92}$$

The resulting sequence values $\langle z(x_{0,i}), \ldots, z(x_{n,i}) \rangle$ has a zero mean, unit variance, and unit standard deviation.

Applied explicitly, centering by subtracting the mean value will destroy sparsity in the algorithm because of the offset term $-\hat{\mu}_i/\hat{\sigma}_i$.

The main motivation for centering is to be able to choose a prior with mean zero. With a fixed intercept feature with an uninformative prior, the intercept coefficient can absorb all of the constant offsets $-\hat{\mu}_i/\hat{\sigma}_i$ without changing the error function. Thus centering may be done implicitly.

The z-transform itself is linear, so it does not have any effect on maximum likelihood estimates. Assuming an intercept feature in the input, the maximum likelihood estimate for normalized inputs simply rescales the parameters by the deviation σ_i for dimension i and shifts the intercept parameter by $\hat{\mu}_i/\hat{\sigma}_i$.

The MAP solutions with Gaussian, Laplace or Cauchy priors are not equivalent under rescaling of inputs. Setting aside the intercept input, the other parameters will be equivalent if the priors for input dimension $i < d$ are rescaled by $\hat{\sigma}_i$.

The main motivation for variance normalization is to be able to choose a single prior variance τ^2 for every dimension. With the ability to set prior variances per dimension, the same effect is achieved by scaling τ^2 by $\hat{\sigma}_i^2$, defining the prior for dimension i to be $\tau^2/\hat{\sigma}_i^2$.

L To-Do

L.1 Graphs

- prior distributions overlayed (a la Gelman et al.)
- coeffs vs. prior (a la Hastie et al.)
- total log likelihood training + held out
- posterior parameter distro (a la Goodman)

L.2 AKA

lasso, ridge, shrinkage, regularization

sequential gradient descent, pattern-based gradient descent

softmax, 1-level perceptron with sigmoid activation and log loss, exponential model, log-linear model, generalized linear model

L.3 Add

matrix notation for gradient and Hessian

alternative $c_i \in \{-1, 1\}$ notation plus δ_ϕ delta functions.

References

Textbooks

1. Bishop, Christopher M. 1995. *Neural Networks for Pattern Recognition.* Oxford University Press.
2. Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning.* Springer.
3. Gelman, Andrew, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2003. *Bayesian Data Analysis*, 2nd Edition. Chapman-Hall.
4. Gelman, Andrew and Jennifer Hill. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models.* Cambridge University Press.
5. Hastie, Trevor, Robert Tibshirani, and Jerome Feldman. 2001. *The Elements of Statistical Learning.* Springer.
6. MacKay, David J. C. 2003. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press.

Tutorials

1. Bottou, Léon. 2007. Learning with Large Datsets.
   http://leon.bottou.org/slides/largescale/lstut.pdf
2. Klein, Dan and Chris Manning. 2003. *Maxent Models, Conditional Estimation, and Optimization.*
   http://www.cs.berkeley.edu/ klein/papers/maxent-tutorial-slides-6.pdf
3. Pereira, Ferando. *Multinomial Logistic Regression/Maximum Entropy.*
   http://www.cis.upenn.edu/ pereira/classes/CIS620/lectures/maxent.pdf

Software Packages

1. Alias-i. *LingPipe.* [Java]
   http://alias-i.com/lingpipe
2. Baldridge, Jason, Tom Morton, and Gann Bierner. *OpenNLP Maxent.* [Java]
   http://maxent.sourceforge.net/
3. Bird, Steven, Edward Loper, and Ewan Klein. *NLTK: Natural Language Toolkit.* [Python]
   http://nltk.org
4. Bottou, Léon. *SGD: Stochastic Gradient Descent.* [C++]
   http://leon.bottou.org/projects/sgd
5. Cohen, William et al. *Minor Third.* [Java]
   http://minorthird.sourceforge.net
6. Daumé, Hal, III. *Megam: MEGA Model Optimization Package.* [O'Caml]
   http://www.cs.utah.edu/ hal/megam/index.html
7. Genkin, Alexander, David D. Lewis, and David Madigan. *BMR: Bayesian Multinomial Regression Software.* [C++]
   http://www.stat.rutgers.edu/ madigan/BMR/
8. Langford, John, Lihong Li, and Alex Strehl. *Vowpal Wabbit.* [C++]
   http://hunch.net/ vw/
9. Manning, Christopher and Dan Klein. *Stanford Classifier.* [Java]
   http://nlp.stanford.edu/software/classifier.shtml
10. McCallum, Andrew K. et al. *Mallet.* [Java]
    http://mallet.cs.umass.edu/
11. Och, Franz Josef. *YASMET: Yet Another Small MaxEnt Toolkit.* [C++]
    http://www.fjoch.com/YASMET.html

Input

k	number of output categories $k \in \mathbb{N}, k > 1$
d	number of input dimensions $d \in \mathbb{N}, d > 0$
$\langle \sigma_i^2 \rangle_{i<d}$	prior variances $\sigma_i^2 \in \mathbb{R}, \sigma_i^2 > 0$
$\langle x_j, c_j \rangle_{j<n}$	training data $x_j \in \mathbb{R}^d$, $c_j \in \{0, \ldots k-1\}$
m	maximum number of training epochs $m \in \mathbb{N}$, $m > 0$
ϵ	minimum relative error improvement per epoch $\epsilon \in \mathbb{R}$, $\epsilon > 0$
η_0	initial learning rate $\eta_0 \in \mathbb{R}$, $\eta_0 > 0$
δ	annealing rate $\delta \in \mathbb{R}$, $\delta > 0$

Output

$\langle \beta_c \rangle_{c<k-1}$	model parameters $\beta_c \in \mathbb{R}^d$

$\beta \leftarrow 0$
for $e \leftarrow 0$ **to** $m - 1$ **do**
  $\eta_e \leftarrow \frac{\eta_0}{1 + e/\delta}$
  for $j \leftarrow 0$ **to** $n - 1$ **do**
    $Z \leftarrow 1 + \sum_{c<k-1} \exp(\beta_c \cdot x_j)$
    for $c \leftarrow 0$ **to** $k - 2$ **do**
      $p(c \mid x_j, \beta) \leftarrow \exp(\beta_c \cdot x_j)/Z$
      $\beta_c \leftarrow \beta_c + \eta_e(\nabla_c \operatorname{Err}_\ell(x_j, c_j, \beta) + \frac{\nabla_c \operatorname{Err}_R(\beta, \sigma^2)}{n})$
  $\ell_e = -\sum_{j<n-1} \log p(c_j \mid x_j, \beta) + \operatorname{Err}_R(\beta, \sigma^2)$
  if $\operatorname{relDiff}(\ell_e, \ell_{e-1}) < \epsilon$ **then**
    return β

$$\nabla_c \operatorname{Err}_\ell(x_j, c_j, \beta) = x_j \left(\mathrm{I}(c = c_j) - p(c \mid x_j, \beta)\right)$$

$$\nabla_{c,i} \operatorname{Err}_{\mathrm{ml}}(\beta, \sigma^2) = 0$$

$$\nabla_{c,i} \operatorname{Err}_{\mathrm{L}_2}(\beta, \sigma^2) = -\frac{\beta_{c,i}}{\sigma_i^2}$$

$$\nabla_{c,i} \operatorname{Err}_{\mathrm{L}_1}(\beta, \sigma^2) = -\frac{\sqrt{2}\ \operatorname{signum}(\beta_{c,i})}{\sigma_i}$$

$$\nabla_{c,i} \operatorname{Err}_{\mathrm{t}_1}(\beta, \lambda) = -\frac{2\beta_{c,i}}{\beta_{c,i}^2 + \lambda_i^2}$$

$$\operatorname{relDiff}(x, y) = \frac{|\ x - y\ |}{|\ x\ | + |\ y\ |}$$

Algorithm 1: Stochastic Gradient Descent

Input

k	number of output categories $k \in \mathbb{N}, k > 1$
d	number of input dimensions $d \in \mathbb{N}, d > 0$
$\langle \sigma_i^2 \rangle_{i<d}$	prior variances $\sigma_i^2 \in \mathbb{R}, \sigma_i^2 > 0$
$\langle x_j, c_j \rangle_{j<n}$	training data $x_j \in \mathbb{R}^d$, $c_j \in \{0, \ldots k-1\}$
m	maximum number of training epochs $m \in \mathbb{N}$, $m > 0$
ϵ	minimum relative error improvement per epoch $\epsilon \in \mathbb{R}$, $\epsilon > 0$
η_0	initial learning rate $\eta_0 \in \mathbb{R}$, $\eta_0 > 0$
δ	annealing rate $\delta \in \mathbb{R}$, $\delta > 0$

Output

$\langle \beta_c \rangle_{c<k-1}$	model parameters $\beta_c \in \mathbb{R}^d$

$\beta \leftarrow 0$
$u \leftarrow 0$
$q \leftarrow 0$
for $e \leftarrow 0$ **to** $m - 1$ **do**
  $\eta_e \leftarrow \frac{\eta_0}{1 + e/\delta}$
  for $j \leftarrow 0$ **to** $n - 1$ **do**
    $Z \leftarrow 1 + \sum_{c<k-1} \exp(\beta_c \cdot x_j)$
    for i *such that* $x_{j,i} \neq 0$ **do**
      for $c \leftarrow 0$ **to** $k - 2$ **do**
        $\beta_{c,i} \leftarrow \beta_{c,i} + \eta_e \frac{u_i - q}{n} \nabla_{c,i} \mathrm{Err}_R(\beta_{c,i}, \sigma^2)$
      $u_i \leftarrow q$
    for $c \leftarrow 0$ **to** $k - 2$ **do**
      $p(c \mid x_j, \beta) \leftarrow \exp(\beta_c \cdot x_j)/Z$
      $\beta_c \leftarrow \beta_c + \eta_e \nabla_c \mathrm{Err}_\ell(x_j, c_j, \beta)$
    $q \leftarrow q + 1$
  $\ell_e = -\sum_{j<n-1} \log p(c_j \mid x_j, \beta) + \mathrm{Err}_R(\beta, \sigma^2)$
  if $\mathrm{relDiff}(\ell_e, \ell_{e-1}) < \epsilon$ **then**
    return β

Algorithm 2: Lazy Stochastic Gradient Descent