The linear model `lin-1`

Carl Edward Rasmussen

May 1, 1996

The linear model is one of the most popular models used in data analysis. The reasons for its popularity is that it is both conceptually and computationally simple to fit a linear model and the resulting model is easily given an intuitive representation. The most prominent deficiency of the linear model, is its strong assumptions about the true relationship in the data; for data which do not conform well to a linear model, predictions may be inaccurate.

I have implemented the linear model which I will call `lin-1`. The strategy of fitting a linear model is well known and the only heuristics necessary is a principled way of handling the situation where the linear system used to determine the parameters of the model is close to singular.

For simplicity, I will initially assume that the targets are scalar; later on a simple extension to handle multiple outputs will be given. The linear model is defined as

$$f_w(x) = \sum_{i=1}^{m+1} x_i w_i, \tag{1}$$

where $w_i,\ i = 1, \ldots, m+1$ are the $m+1$ model parameters, w_{m+1} being the bias and $x_i,\ i = 1, \ldots, m$ are the inputs, augmented by $x_{m+1} \equiv 1.0$ to take care of the bias. The model is fit to the training data $\mathcal{D} = \{x^{(i)}, y^{(i)} | i = 1 \ldots n\}$ by maximum likelihood, assuming zero mean Gaussian noise with variance σ^2. The likelihood is

$$p\big(y^{(1)}, \ldots, y^{(n)} \big| x^{(1)}, \ldots, x^{(n)}, w, \sigma^2\big) \propto \prod_{c=1}^{n} \exp\Big(-\frac{\big(f_w(x^{(c)}) - y^{(c)}\big)^2}{2\sigma^2}\Big). \tag{2}$$

The maximum likelihood estimate for the weights w_{ML} is independent of σ^2 and is found by minimizing the cost function

$$E(w) = \sum_{c=1}^{n} \big(f_w(x^{(c)}) - y^{(c)}\big)^2, \tag{3}$$

with respect to the model parameters. Notice, that the Gaussian noise assumption gives rise to a squared error cost function; this cost function will always be used regardless of whatever *loss function* we chose to evaluate the linear model. The solution to this optimization problem is well known from linear algebra; if the solution is unique, then

$$\begin{gathered}\frac{\partial E(w)}{\partial w_j} = \mathbf{0} \;\Rightarrow\; \sum_{c=1}^{n}\Big(\sum_{i=1}^{m+1} w_i x_i^{(c)} - y^{(c)}\Big)x_j^{(c)} = \mathbf{0} \;\Rightarrow\; w_{\mathrm{ML}} = \mathbf{A}^{-1}\mathbf{b} \\ \text{where } \mathbf{A}_{ii'} = \sum_{c=1}^{n} x_i^{(c)} x_{i'}^{(c)}, \quad \mathbf{b}_i = \sum_{c=1}^{n} y^{(c)} x_i^{(c)}, \quad i, i' = 1, \ldots, m+1.\end{gathered} \tag{4}$$

If $\mathbf{A}$ is ill-conditioned numerical evaluation of eq. (4) may be troublesome, and even if fairly accurate solutions could be obtained these would not necessarily lead to good model predictions. In an attempt to define a method which has a high degree of reproducibility I propose to chose w such that directions in input space with insufficient variation in the training set are ignored by the model. This can conveniently be computed using singular value decomposition (SVD), see for example (Press et al. 1992). The decomposition is $\mathbf{A} = \mathbf{U}\,\mathrm{diag}(\lambda_i)\mathbf{V}^{\mathrm{T}}$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and λ is a vector of length $m+1$ containing the singular values of $\mathbf{A}$. A regularised $\tilde{\mathbf{A}}^{-1}$ can be computed by setting $1/\lambda_i = 0$ for those i whose λ_i are too small in

$$\tilde{\mathbf{A}}^{-1} = \mathbf{V}\,\mathrm{diag}(1/\lambda_i)\mathbf{U}^{\mathrm{T}}. \tag{5}$$

The exact criterion for regularisation is: set $1/\lambda_i = 0$ whenever $\lambda_i < 10^{-6}\max_j(\lambda_j)$, ie, whenever $\mathbf{A}$ is close to singular. The constant of 10^{-6} is chosen as a rather conservative estimate of machine precision which will not interfere when $\mathbf{A}$ is well-conditioned. The condition number of $\mathbf{A}$ depends on the scale of the inputs, so the procedure should always be used in conjunction with the standard normalisations provided by DELVE. The (modified) maximum likelihood weights can then be computed as

$$\tilde{w}_{\mathrm{ML}} = \tilde{\mathbf{A}}^{-1}\mathbf{b}. \tag{6}$$

We now derive the predictive distribution for the model. For simplicity we will derive the predictive distribution for a fixed value of the noise, and only account for uncertainty in the predictions arising from the estimated noise inherent in the data and from uncertainty in the estimate for the weights. The noise is estimated by

$$\hat{\sigma}^2 = \frac{1}{n-k}\sum_{c=1}^{n}\big(f_{\tilde{w}_{\mathrm{ML}}}(x^{(c)}) - y^{(c)}\big)^2 \tag{7}$$

where k is the number of parameters in w_{ML} which were fit by the data; this is $m+1$ minus 1 for every singular value whose reciprocal was set to zero in eq. (5). This estimate may break down if there are too few training cases (if $n \leq m+1$), in which case we can't compute a predictive distribution. Since the prior on weights is uniform (non-existent), the posterior for the weights is proportional to the likelihood. Thus, the predictive distribution for a test case with input $x^{(n+1)}$ is

$$\begin{aligned}&p\big(y^{(n+1)}\big|(x^{(1)}, \ldots, x^{(n+1)}, y^{(1)}, \ldots, y^{(n)}, \hat{\sigma}^2\big) \\ &\quad\propto \int p\big(y^{(n+1)}\big|x^{(1)}, \ldots, x^{(n+1)}, y^{(1)}, \ldots, y^{(n)}, \hat{\sigma}^2\big)p\big(w\big|\mathcal{D}, \hat{\sigma}^2\big)\,d^{m+1}w \\ &\quad\propto \int p\big(y^{(n+1)}\big|x^{(n+1)}, w, \hat{\sigma}^2\big)p\big(y^{(1)}, \ldots, y^{(n)}\big|x^{(1)}, \ldots, x^{(n)}, w, \hat{\sigma}^2\big)\,d^{m+1}w \\ &\quad\propto \int \exp\Big(-\frac{\big(f_w(x^{(n+1)}) - y^{(n+1)}\big)^2}{2\hat{\sigma}^2} - \frac{1}{2}(w - \tilde{w}_{\mathrm{ML}})^{\mathrm{T}}\tilde{\mathbf{A}}(w - \tilde{w}_{\mathrm{ML}})\Big)\,d^{m+1}w.\end{aligned} \tag{8}$$

This Gaussian integral can be solved exactly; we get a Gaussian predictive distribution with mean and variance:

$$\hat{\mu} = f_{\tilde{w}_{\mathrm{ML}}}(x^{(n+1)}) \qquad \text{and} \qquad \hat{e}^2 = \hat{\sigma}^2 + (x^{(n+1)})^{\mathrm{T}}\tilde{\mathbf{A}}^{-1}x^{(n+1)}. \tag{9}$$

The optimal point predictions for any symmetric loss function are given by $f_{\hat{w}_{\mathrm{ML}}}(x)$. Consequently, these predictions can be used for both absolute error and squared error loss functions. For negative log density loss function, we compute the log of the density of the targets under the predictive Gaussian distribution

$$\log p\big(y^{(n+1)} = t^{(n+1)} \big| x^{(1)}, \ldots, x^{(n+1)}, y^{(1)}, \ldots, y^{(n)}, \hat{\sigma}^2\big) = \\ -0.5 \log(2\pi\hat{e}^2) - \frac{(\hat{\mu} - t)^2}{2\hat{e}^2}. \tag{10}$$

For tasks that have multiple outputs we can re-use the decomposition of **A** from eq. (5) for every set of weights. The maximum likelihood weights and inherent noise estimates can be found by using eq. (6) and (7) for each target attribute, and point predictions can be obtained from the maximum likelihood weights as before. For the log density predictions we assume that the joint density for the targets is Gaussian with a diagonal covariance matrix, such that the density of the targets can be obtained by summing (in the log domain) contributions of the form in eq. (10).

This completes the definition of the linear model; following this recipe the model will always be uniquely defined from the data. Many elaborations to this basic linear model exist, but they will not be pursued further here.

The computational complexity involved in fitting `lin-1` is $O(nm^2 + m^3)$, for computing and decomposing the **A** matrix respectively. Even for fairly large tasks this can usually be considered trivial.

References

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. 1992. *Numerical Recipes in C*. Cambridge University Press, second edition.