CS 536 : Logistic Regression

16:198:536

Defining the Model

Consider again a classification problem: we have a data set of the form $\{(\underline{x}^i, y^i)\}_{i=1}^m$, where each y^i represents the class label of the point $\underline{x}^i$, either $y^i = 0$ or $y^i = 1$. In this way we again have a binary class problem, as we did with perceptrons and SVMs. However, we want to take a softer approach - rather than a model predicting the actual class label, we would like to model the probability or likelihood of a data point having a given class label. This might capture the idea that a given label isn't necessarily determined completely or had some noise associated with it. But the higher the probability of having a certain label, the more confident we are in that labeling.

Hence we want to define a model space where $F(\underline{x})$ is interpretable as a probability, between 0 and 1, and we will interpret $F(\underline{x})$ as the likelihood of $\underline{x}$ being given class label 1, so that $1 - F(\underline{x})$ is the likelihood of being given class label 0. As a first model for this, we consider the *logistic regression model*, where we take

$$F_{\underline{w},b}(\underline{x}) = \frac{1}{1 + e^{-(\underline{w}.\underline{x}+b)}}. \tag{1}$$

Roughly, for any $\underline{x}$ such that $\underline{w}.\underline{x} + b = 0$, the probability of belonging to either class is taken to be $1/2$, and the more positive $\underline{w}.\underline{x} + b$ is the more likely $\underline{x}$ is to be of class 1 ($F \to 1$), the more negative it is the more likely $\underline{x}$ is to be of class 0 ($F \to 0$).

Note that it is convenient to imagine that each $\underline{x}^i$ has a '0th coordinate' of 1, so that the bias value b may be absorbed as a 0th coordinate of $\underline{w}$, and we might instead write more simply

$$F_{\underline{w}}(\underline{x}) = \frac{1}{1 + e^{-\underline{w}.\underline{x}}}. \tag{2}$$

Defining $\sigma(z) = 1/(1 + e^{-z})$ as the **sigmoid function**, it's also convenient to write $F_{\underline{w}}(\underline{x}) = \sigma(\underline{w}.\underline{x})$.

This can be viewed as a specification of a model or hypothesis space - there is a dividing line or plane specified by $\underline{w}.\underline{x} = 0$, and the classes are generally associated with either side of the line. We might consider more interesting geometries or non-linear divisions between the class regions by embedding our data in a higher dimensional feature space (quadratic feature space giving us circles, etc), but the key notion of this model is the probabilistic interpretation of the output and the 'soft' division between the classes. This is particularly well suited for situations when there may be a lot of noise in the labeling.

Defining the Error

Having specified a model space, we then want to consider the problem of fitting such a model to a data set - finding the parameters $\underline{w}$ or line $\underline{w}.\underline{x} = 0$ that fits the data as well as possible. To do this, we need to specify an error.

Noting that $F(\underline{x})$ close to 1 is interpreted as a high likelihood that y is 1 for that $\underline{x}$, and that $F(\underline{x})$ close to 0 is interpreted as a high likelihood that y is 0 for that $\underline{x}$, this suggests potentially using an error function like

$$\mathrm{err}_{\mathrm{train}}(F) = \frac{1}{m} \sum_{i=1}^{m} \left(F(\underline{x}^i) - y^i\right)^2, \tag{3}$$

which will try to drive the output values towards the class labels.

However, this doesn't really capture the desired notion of the F as class probabilities. To do so, I wan to revisit a notion from previous lectures, that of *maximum likeihood estimation*. Let $F(\underline{x}^i)$ be 'the probability that $\underline{x}^i$ received class label 1'. In this way, the *likelihood of the class labels* $y^1, \ldots, y^m$ *given* $\underline{x}^1, \ldots, \underline{x}^m$ is given by

$$\text{lik}(y^1, \ldots, y^m | \underline{x}^1, \ldots, \underline{x}^m) = \prod_{i=1}^{m} F(\underline{x}^i)^{y^i} (1 - F(\underline{x}^i))^{1-y^i}. \tag{4}$$

Why? Hint, what does one of these factors look like if $y^i = 0$*? What if* $y^i = 1$*?*

Given the above notion of likelihood for this data set, we can ask: what is the vector $\underline{w}$ that makes the specified data as likely as possible? i.e., what is the maximum likelihood estimator for $\underline{w}$?

Note that the product given in (4) is a relatively straightforward function (smooth, differentiable) of the data and the unknown vector $\underline{w}$ - we could treat it as such, and apply any given optimization algorithm to find the maximizer $\underline{w}^*$. But it is worth massaging it slightly and putting it in a more 'standard' form. In particular, instead of maximizing the likelihood, it is convenient to consider maximizing the *log* of the likelihood. This doesn't change the $\underline{w}^*$ that produces the maximizer, but it does render the objective function a sum rather than a product:

$$\ln \text{lik}(\{y^i\} | \{\underline{x}^i\}) = \sum_{i=1}^{m} \left[y^i \ln F(\underline{x}^i) + (1 - y^i) \ln(1 - F(\underline{x}^i)) \right]. \tag{5}$$

Similarly, to frame it in a more similar way to what we've seen so far, instead of trying to maximize the above, we can consider trying to minimize the negative of the above:

$$-\ln \text{lik}(\{y^i\} | \{\underline{x}^i\}) = \sum_{i=1}^{m} \left[-y^i \ln F(\underline{x}^i) - (1 - y^i) \ln(1 - F(\underline{x}^i)) \right]. \tag{6}$$

The last modification to consider is mostly superficial, but the minimizer is again not changed by scaling the objective function by a constant, such as $1/m$. Hence, we can view the problem of finding the maximum likelihood $\underline{w}^*$ as finding the minimizer of

$$-\frac{1}{m} \ln \text{lik}(\{y^i\} | \{\underline{x}^i\}) = \frac{1}{m} \sum_{i=1}^{m} \left[-y^i \ln F(\underline{x}^i) - (1 - y^i) \ln(1 - F(\underline{x}^i)) \right], \tag{7}$$

which suggests an error function that really captures this notion of the probabilistic interpretation of F is more reasonably given by

$$\text{err}_{\text{train}}(F) = \frac{1}{m} \sum_{i=1}^{m} \left[-y^i \ln F(\underline{x}^i) - (1 - y^i) \ln(1 - F(\underline{x}^i)) \right]. \tag{8}$$

This is the so called log loss error function, strongly related to the cross-entropy loss, etc, commonly utilized in classification applications .

Fitting the Model

We now have, roughly, a well specified machine learning problem: given a data set, and defining $F(\underline{x}) = \sigma(\underline{w}.\underline{x})$, we want to find $\underline{w}^*$ to solve the following:

$$\min_{\underline{w}} \ \frac{1}{m} \sum_{i=1}^{m} \left[-y^i \ln F(\underline{x}^i) - (1 - y^i) \ln(1 - F(\underline{x}^i)) \right]. \tag{9}$$

The above is a nice, differentiable function in terms of $\underline{w}$, so we could technically use something like gradient descent to try to solve it - I'll observe here that if we simply tried to take $\nabla_{\underline{w}} \text{err}_{\text{train}} = 0$, this yields a highly non-linear system of equations to try to solve for the minimizing $\underline{w}^*$.

In addition to gradient descent, the form of the above function suggests that we might consider *stochastic* gradient descent. Instead of taking the gradient of the full error, we could consider only the error on data point i:

$$\text{err}^i_{\text{train}}(\underline{w}) = -y^i \ln F(\underline{x}^i) - (1-y^i)\ln(1-F(\underline{x}^i)). \tag{10}$$

With the above, we might implement the following algorithm:

- At time $t=0$, take $\underline{w}^0$ to be some initial guess.
- At time t, select some data point $i = 1, \ldots m$ at random, and update

$$\underline{w}^{t+1} = \underline{w}^t - \alpha_t \nabla_{\underline{w}} \text{err}^i_{\text{train}}(\underline{w}^t). \tag{11}$$

- Repeat until convergence.

It remains to compute this gradient to get a complete algorithm.

Consider computing the derivative with respect to any w_j component (remembering that w_0 is the bias term corresponding to $x_0 = 1$).

$$\begin{aligned}
\frac{\partial}{\partial w_j}\text{err}^i_{\text{train}}(\underline{w}) &= \frac{\partial}{\partial w_j}\left[-y^i \ln F(\underline{x}^i) - (1-y^i)\ln(1-F(\underline{x}^i))\right] \\
&= -y^i \frac{\frac{\partial}{\partial w_j}F(\underline{x}^i)}{F(\underline{x}^i)} - (1-y^i)\frac{-\frac{\partial}{\partial w_j}F(\underline{x}^i)}{1-F(\underline{x}^i)} \\
&= \left[-y^i \frac{1}{F(\underline{x}^i)} - (1-y^i)\frac{-1}{1-F(\underline{x}^i)}\right]\frac{\partial}{\partial w_j}F(\underline{x}^i) \\
&= \left[-y^i \frac{1}{F(\underline{x}^i)} - (1-y^i)\frac{-1}{1-F(\underline{x}^i)}\right]\frac{\partial}{\partial w_j}\sigma(\underline{w}.\underline{x}^i) \\
&= \left[-y^i \frac{1}{F(\underline{x}^i)} - (1-y^i)\frac{-1}{1-F(\underline{x}^i)}\right]\sigma'(\underline{w}.\underline{x})\frac{\partial}{\partial w_j}\left[\underline{w}.\underline{x}^i\right] \\
&= \left[-y^i \frac{1}{F(\underline{x}^i)} - (1-y^i)\frac{-1}{1-F(\underline{x}^i)}\right]\sigma'(\underline{w}.\underline{x}^i)x^i_j
\end{aligned} \tag{12}$$

It's worth doing the algebra on this, but you can show that $\sigma'(z) = \sigma(z)(1-\sigma(z))$, hence $\sigma'(\underline{w}.\underline{x}^i) = F(\underline{x}^i)(1-F(\underline{x}^i))$. Simplifying,

$$\begin{aligned}
\frac{\partial}{\partial w_j}\text{err}^i_{\text{train}}(\underline{w}) &= \left[-y^i \frac{1}{F(\underline{x}^i)} - (1-y^i)\frac{-1}{1-F(\underline{x}^i)}\right]\sigma'(\underline{w}.\underline{x}^i)x^i_j \\
&= \left[-y^i \frac{1}{F(\underline{x}^i)} - (1-y^i)\frac{-1}{1-F(\underline{x}^i)}\right]F(\underline{x}^i)(1-F(\underline{x}^i))x^i_j \\
&= \left[-y^i(1-F(\underline{x}^i)) - (1-y^i)(-F(\underline{x}^i))\right]x^i_j \\
&= \left[F(\underline{x}^i) - y^i\right]x^i_j
\end{aligned} \tag{13}$$

Hence, the jth component of the gradient of the error term is simply a scalar time the jth component of the $\underline{x}$. This gives us

$$\nabla_{\underline{w}}\text{err}^i_{\text{train}}(\underline{w}) = \left[F(\underline{x}^i) - y^i\right]\underline{x}^i. \tag{14}$$

Hence we can give a full stochastic gradient descent update for logistic regression as:

SGD for Logistic Regression

- At time $t = 0$, take $\underline{w}^0$ to be some initial guess.
- At time t, select some data point $i = 1, \dots m$ at random, and update

$$\underline{w}^{t+1} = \underline{w}^t - \alpha_t \left[F(\underline{x}^i) - y^i\right] \underline{x}^i. \tag{15}$$

- Repeat until convergence.

Multi-Class Problems

In the above analysis we considered only a binary class problem, so that the output of the model F could be interpreted strictly as a probability of belonging to one of the classes. This kind of probabilistic regression admits a very nice and relatively immediate generalization to the multiclass problem as well.

Suppose there are C many classes. Instead of labeling each point $\underline{x}$ with a class label, consider labeling it with an indicator or one hot encoded vector, $\underline{y} = (0, 0, \dots, 1, \dots, 0)$, where a 1 in the c-component indicates membership in class c.

In this case, we now want to output not a single real value, but multiple real values - we would like to output for each class a probability of belonging to that class. As such, we would like $\underline{F}(\underline{x})$ to be a C-dimensional vector, where each component is non-negative, and the sum of the components is 1. We can then interpret the output vector as the vector of probabilities of class membership for each class. Note that ensuring the sum of the components of $\underline{F}(\underline{x})$ is 1 means that the classes are taken to be *exclusive* - being in one class immediately restricts you from being in other classes.

The previous discussion of error and maximum likelihood estimation generalizes immediately, yielding an error function of the form

$$\frac{1}{m} \sum_{i=1}^{m} \left[- \sum_{c=1}^{C} y_c^i \ln \left[F_c(\underline{x}^i)\right] \right], \tag{16}$$

taking $F_c(\underline{x})$ as the c-component of $\underline{F}(\underline{x})$. If $\underline{F}$ is a parameterized model, it remains to find the parameters the minimize the above error.

But, in a somewhat circuitous way, this leads us back to the question of how to define our model space. To ensure that the F_c are non-negative and sum to 1, a classic solution is to generalize the simple logistic regression model with the use of a **softmax** function. Each class c has some vector $\underline{w}^c$ associated with it, and the c-component of F is taken to be

$$F_c(\underline{x}) = \frac{e^{\underline{w}^c \cdot \underline{x}}}{\sum_{c'=1}^{C} e^{\underline{w}^{c'} \cdot \underline{x}}}. \tag{17}$$

The exponential function is always non-negative, and the normalization in the above expression ensures that $\sum_c F_c = 1$, so the components of $\underline{F}$ may be interpreted as valid probabilities. Defining this softmax logistic model in this way, it remains to find the vectors $\{\underline{w}^c\}$ that minimize the above error. The only additional complication this presents over all is a slightly more complex gradient computation.

It's worth observing here that taking a softmax model with two classes, $C = 2$, the original logistic regression model can be recovered with weight vector $\underline{w} = \underline{w}^2 - \underline{w}^1$.