CS 189/289A Introduction to Machine Learning
Fall 2022 Jennifer Listgarten, Jitendra Malik DIS3

1 Logistic Regression

Assume that we have n i.i.d. data points $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, where each y_i is a binary label in $\{0, 1\}$. We model the posterior probability as a Bernoulli distribution and the probability for each class is the sigmoid function, i.e., $p(y|\mathbf{x}; \mathbf{w}) = q^y(1-q)^{1-y}$, where $q = s(\mathbf{w}^\top \mathbf{x})$ and $s(\zeta) = \frac{1}{1+e^{-\zeta}}$ is the sigmoid function. Write out the likelihood and log likelihood functions. Comment on whether it is possible to find a closed form maximum likelihood estimate of $\mathbf{w}$, and describe an alternate approach.

Solution:

The likelihood is:

$$L(\mathbf{w}) = \prod_{i=1}^{n} p(y = y_i|\mathbf{x}_i) = \prod_{i=1}^{n} q_i^{y_i}(1-q_i)^{1-y_i}.$$

Now, we step through minimizing the negative log likelihood of the training data as a function of the parameters $\mathbf{w}$:

$$\begin{aligned}
\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} -L(\mathbf{w}) &= \arg\min_{\mathbf{w}} \left[-\prod_{i=1}^{n} q_i^{y_i}(1-q_i)^{1-y_i} \right] \\
&= \left[\arg\min_{\mathbf{w}} -\sum_{i=1}^{n} y_i \log(q_i) + (1-y_i)\log(1-q_i) \right] \\
&= \left[\arg\min_{\mathbf{w}} -\sum_{i=1}^{n} y_i \log\left(\frac{q_i}{1-q_i}\right) + \log(1-q_i) \right]
\end{aligned}$$

Since q_i is the sigmoid function, we plug it in and simplify:

$$\begin{aligned}
\hat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \left[-\sum_{i=1}^{n} y_i \log\left(\frac{q_i}{1-q_i}\right) + \log(1-q_i) \right] \\
&= \arg\min_{\mathbf{w}} \left[-\sum_{i=1}^{n} y_i \log\left(\frac{\frac{1}{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}}{1 - \frac{1}{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}}\right) + \log\left(1 - \frac{1}{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}\right) \right] \\
&= \arg\min_{\mathbf{w}} \left[-\sum_{i=1}^{n} y_i \log\left(\frac{\frac{1}{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}}{\frac{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}-1}{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}}\right) + \log\left(\frac{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}-1}{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}\right) \right]
\end{aligned}$$

$$= \arg\min_{\mathbf{w}} \left[-\sum_{i=1}^{n} y_i \log\left(\frac{1}{\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}\right) + \log\left(\frac{\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}\right)\right]$$

$$= \arg\min_{\mathbf{w}} \left[-\sum_{i=1}^{n} y_i \mathbf{w}^\top \mathbf{x}_i - \mathbf{w}^\top \mathbf{x}_i + \log\left(1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}\right)\right]$$

We bring in the negative sign to get:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left[\sum_{i=1}^{n} (1-y_i)\mathbf{w}^\top \mathbf{x}_i + \log\left(1+\exp\left\{-\mathbf{w}^\top \mathbf{x}_i\right\}\right)\right]$$

Let us denote $J(\mathbf{w}) = \sum_{i=1}^{n}(1-y_i)\mathbf{w}^\top \mathbf{x}_i - \log\left(1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}\right)$. Notice that $J(\mathbf{w})$ is convex in $\mathbf{w}$, so global minimum can be found. Note that $s'(\zeta) = s(\zeta)(1-s(\zeta))$. Now let us take the gradient of $J(\mathbf{w})$ w.r.t $\mathbf{w}$:

$$\nabla_w J = \sum_{i=1}^{n}(1-y_i)\mathbf{x}_i - \frac{\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}{1+\exp\{-\mathbf{w}^\top \mathbf{x}_i\}}\mathbf{x}_i = \sum_{i=1}^{n}(-1+s(\mathbf{w}^\top \mathbf{x}_i)-y_i+1)\mathbf{x}_i = \sum_{i=1}^{n}(s_i-y_i)\mathbf{x}_i = \mathbf{X}^\top(\mathbf{s}-\mathbf{y})$$

where, $s_i = s(\mathbf{w}^\top \mathbf{x}_i), \mathbf{s} = (s_1, \ldots, s_n)^\top, \mathbf{y} = (y_1, \ldots, y_n)^\top$ and $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$.

Unfortunately, we can't get a closed form estimate for $\mathbf{w}$ by setting the derivative to zero, given that the term $\mathbf{s}$ still contains $\mathbf{w}$, and further-order derivatives will continue to carry expressions over $\mathbf{w}$. However, the convexity of this problem allows for first-order optimization algorithms, such as gradient descent, to converge to a global minimum.

2 Initialization of Weights for Backpropagation

Assume a fully-connected 1-hidden-layer network. Denote the dimensionalities of the input, hidden, and output layers as $d^{(0)}$, $d^{(1)}$, and $d^{(2)}$. That is, the input (which we will denote with a superscript (0)) is a vector of the form $x_1^{(0)}, \dots, x_{d^{(0)}}^{(0)}$. Let g denote the activation function applied at each layer. We will let $S_j^{(l)} = \sum_{i=1}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}$ be the weighted input to node j in layer l, and let $\delta_j^{(l)} = \frac{\partial \ell}{\partial S_j^{(l)}}$ be the partial derivative of the final loss ℓ with respect to $S_j^{(l)}$.

Recall that backpropagation is an efficient method to compute the gradient of the loss function so we can use it for gradient descent. Gradient descent requires the parameters to be initialized to some value(s).

(a) To better orient yourself with the operations described in this 1-hidden-layer network, draw out a diagram of the layers, including weights, activation functions, and the outputs of each operation during the forward pass. In addition, identify where the partial derivatives $\delta_j^{(l)}$ are calculated during backpropagation.

Solution:

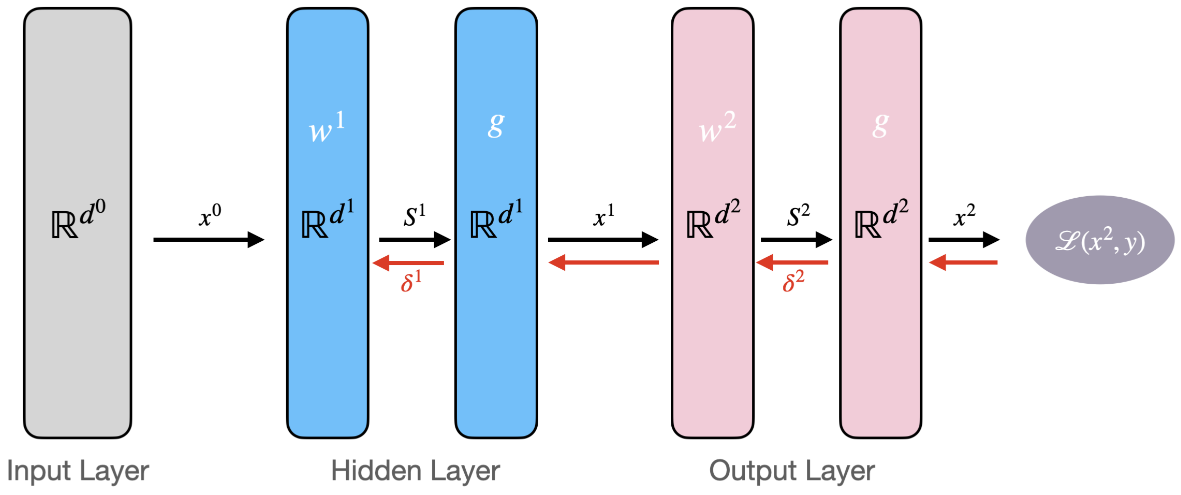

Figure 1: Simple diagram of the 1-hidden layer network

Note that the output layer does not necessarily have to include an activation function (and often in practice does not); we add one here for consistency.

(b) Imagine that we initialize every element of each weight $w^{(l)}$ to be the same constant scalar value a. After performing the forward pass, what is the value of $x_j^{(1)}$ in terms of the elements of $\{x_i^{(0)} : i = 1, \dots, d^{(0)}\}$? What is the relationship between each $x_j^{(1)}$?

Solution: Since all of the weights are equal, we have:

$$x_j^{(1)} = g\left(\sum_{i=1}^{d^{(0)}} w_{ij}^{(1)} x_i^{(0)}\right) = g\left(a \sum_{i=1}^{d^{(0)}} x_i^{(0)}\right)$$

Note that this equation does not depend on j so all $x_j^{(1)}$ are equal.

(c) Following from the previous part, after the backward pass of backpropagation, compute the values for each member of the set $\{\delta_i^{(1)} : i = 1, \ldots, d^{(1)}\}$, assuming we have calculated $\{\delta_j^{(2)} : j = 1, \ldots, d^{(2)}\}$. What is the relationship between each $\delta_i^{(1)}$?

Solution: Since all of the weights are equal:

$$\begin{aligned}\delta_i^{(1)} &= \sum_{j=1}^{d^{(2)}} \delta_j^{(2)} w_{ij}^{(2)} g'(S_i^{(1)}) \\ &= g'(S_i^{(1)}) a \sum_{j=1}^{d^{(2)}} \delta_j^{(2)} \\ &= g'\left(a \sum_{k=1}^{d^{(0)}} x_k^{(0)}\right) a \sum_{j=1}^{d^{(2)}} \delta_j^{(2)}\end{aligned}$$

The sum term within the activation function is the same for all dimensions i in layer 1 (note that this is not referring to being the same across all nodes, but rather the entries within each node.) The members of the set are equal.

(d) For a reasonable loss function, are all of the $\delta_i^{(2)}$ equal to each other? Why or why not?

Solution: No. $\delta_i^{(2)}$ depends on y_i, which is different for each i. Note that *any* reasonable output loss/activation should result in $\delta_i^{(2)}$ depending on a target y_i (otherwise the output is not attempting to approach a particular target value).

(e) After the weights are updated and one iteration of gradient descent has been completed, what can we say about the weights?

Solution: Our gradient descent update looks like this, for some learning rate η:

$$\begin{aligned} w_{ij}^{(l)} &= w_{ij}^{(l)} - \eta \delta_j^{(l)} x_i^{(l-1)} = a - \eta \delta_j^{(l)} x_i^{(l-1)} \\ \implies w_{ij}^{(1)} &= a - \eta \delta_j^{(1)} x_i^{(0)} \\ \implies w_{ij}^{(2)} &= a - \eta \delta_j^{(2)} x_i^{(1)}\end{aligned}$$

For a given i, $w_{ij}^{(1)}$ will be the same for all j because $\delta_j^{(1)}$ is equal for all j.

For a given j, $w_{ij}^{(2)}$ will be the same for all i because $x_i^{(1)}$ is equal for all i.

(f) In the previous part, you showed that $w_{ij}^{(2)}$ is different for each j, but for a fixed j, it is the same for each i. In fact, no matter how many subsequent iterations of gradient descent you take, this property will continue to be true. Show why this is the case.

Solution: Let $w_{ij}^{(1)} = w_i^{(1)}$ for all i. Let $w_{ij}^{(2)} = w_j^{(2)}$ for all j. Then:

$$x_j^{(1)} = g\left(\sum_{i=1}^{d^{(0)}} w_{ij}^{(1)} x_i^{(0)}\right) = g\left(\sum_{i=1}^{d^{(0)}} w_i^{(1)} x_i^{(0)}\right)$$

Which does not depend on j. Also,

$$\delta_i^{(1)} = \sum_{j=1}^{d^{(2)}} \delta_j^{(2)} w_{ij}^{(2)} g'\left(\sum_{k=1}^{d^{(0)}} w_{ki}^{(1)} x_k^{(0)}\right) = \sum_{j=1}^{d^{(2)}} \delta_j^{(2)} w_j^{(2)} g'\left(\sum_{k=1}^{d^{(0)}} w_k^{(1)} x_k^{(0)}\right)$$

Which does not depend on i.

All $x_j^{(1)}$ being the same and all $\delta_i^{(1)}$ being the same will continue no matter how many iterations you perform. Intuitively, $x_j^{(1)}$ will always be the same for all j, due to the "outgoing" symmetry in the weights in layer 1, and the $\delta_i^{(1)}$ will always be the same due to the "incoming" symmetry in the weights in layer 2.

(g) To solve this problem, we randomly initialize our weights. This is called symmetry breaking. Note that for logistic regression, we don't run into this issue; that is, gradient descent will find the optimal values of the weights even if we initialize them at 0. Explain why this discrepancy exists between our 1-hidden-layer neural network and logistic regression.

Solution: Logistic regression is a fully-connected neural network with one output node and zero hidden layers (remember that the $\delta_j^{(2)}$'s are different).

Another reason, following the above logic: in logistic regression, the loss function is convex. Any starting point should lead us to the global optimum.