

Regularization

CS 360 Machine Learning
Week 5, Day 2

February 22, 2024

Contents

1	Multi-class Logistic Regression (softmax)	1
2	Regularization	2
2.1	Fairness Regularization	4

1 Multi-class Logistic Regression (softmax)

Recall that our prediction is now going to be based on a set of weights for every class. We have the prediction vector:

$$\hat{\mathbf{y}} = h_{\vec{w}}(\vec{x}) = \begin{bmatrix} p(y = 1 | \vec{x}) \\ p(y = 2 | \vec{x}) \\ \cdot \\ \cdot \\ p(y = K | \vec{x}) \end{bmatrix} = \frac{1}{\sum_{k=1}^K e^{\vec{w}^{(k)} \cdot \vec{x}}} \begin{bmatrix} e^{\vec{w}^{(1)} \cdot \vec{x}} \\ e^{\vec{w}^{(2)} \cdot \vec{x}} \\ \cdot \\ e^{\vec{w}^{(k)} \cdot \vec{x}} \\ \cdot \\ e^{\vec{w}^{(K)} \cdot \vec{x}} \end{bmatrix} = \begin{bmatrix} e^{s_1} \\ e^{s_2} \\ \cdot \\ e^{s_k} \\ \cdot \\ e^{s_K} \end{bmatrix}$$

Normalization term ↑

We think of each $s_k = \vec{w}^{(k)} \cdot \vec{x}$ as a score, which is normalized by the softmax function to create probabilities for each class

$$p(y = k | \vec{x}) = \frac{e^{\vec{w}^{(k)} \cdot \vec{x}}}{\sum_{i=1}^K e^{\vec{w}^{(i)} \cdot \vec{x}}}$$

Note that basing these terms off of e and including the normalization terms makes sure that the resulting terms are a probability distribution, even though the values of $\vec{w} \cdot \vec{x}$ can be positive or negative. This is a common generalization of logistic regression to a multi-class setting.

How do we incorporate this into our cost function? We're still looking at the negative log likelihood J , where W is now a $(p + 1)$ by K weight matrix, and our overall goal is still to determine these weights. We incorporate the logistic function, which is now also defined using our s_k values, such that the negative log likelihood J is defined as:

$$J(W) = \sum_{i=1}^n \sum_{k=1}^K y_{ik} \log(p(y_i = k | \vec{x}_i))$$

↑ True label
↓ Logistic function

We can think of this cost function as showing the cross entropy between the true label and the logistic function predicted label. Cross entropy is defined as:

$$H(p, q) = - \sum_x p(x) \log q(x)$$

If these two distributions are really similar, then it will reduce the cross entropy value. Using this loss incentivizes the predicted distribution to match the true distribution. We talked last time about how the true label is represented via a one-hot encoding where there's a probability distribution shown as a vector where only one of the classes is a 1. Putting this into a cross entropy loss function incentivizes that single class to be a 1 in the resulting predicted probability distribution.

In order to determine these weights based on the negative log likelihood, we use a generalized version of our previous methods: taking the derivative, setting it to zero, and putting that into our gradient descent step.

2 Regularization

Recall from CS 260 the notion of thresholds and decision boundaries. Suppose that we only have one feature ($p = 1$) and have the model:

$$h_{\vec{w}}(x) = p(\hat{y} = 1|x) = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$$

If $h_{\vec{w}}(x) \geq 0.5$ we predict $\hat{y} = 1$. If we plug in this threshold, we see that:

$$\begin{aligned} \frac{1}{1 + e^{-(w_0 + w_1 x)}} &\geq 0.5 \\ 1 &\geq 0.5 \left(1 + e^{-(w_0 + w_1 x)}\right) \\ \log(1) &\geq \log\left(e^{-(w_0 + w_1 x)}\right) \\ 0 &\geq -(w_0 + w_1 x) \\ w_0 &\geq -w_1 x \\ x &\geq -\frac{w_0}{w_1} \end{aligned}$$

We have a linear decision boundary telling us what to predict based on the value of x . But we could change w_0 and w_1 based on gradient descent. We might end up with weights that have this same ratio, but where the weights themselves have increased. Further runs of gradient descent might keep increasing these, making a sharper and sharper decision boundary. This doesn't allow us to capture uncertainty, and is a form of overfitting to the data. We don't want this — we'd prefer to have smaller weights and avoid this type of overfitting.

How do we avoid this problem? Regularization. Our goal with regularization is to encourage the weights of the learned weight vector to be small. Consider the regularized version of the cost function to be denoted $J^R(\vec{w})$:

$$J^R(\vec{w}) = \underbrace{\left[- \sum_{i=1}^n y_i \log h(\vec{x}_i) + (1 - y_i) \log(1 + h(\vec{x}_i)) \right]}_{\text{Original binary classification cost function}} + \underbrace{\frac{\lambda}{2} \sum_{j=1}^p w_j^2}_{\text{Regularization term}}$$

The regularization term goal is to incentivize small weights. $\lambda \in (0, 1)$ is a hyperparameter known as the *regularization parameter* and allows us to determine how much to weight the regularization term for a specific data set, for example based on the amount of the data. We would expect that small datasets might need smaller values of λ . Whatever the value of the regularization term, we want to make sure it doesn't overwhelm the cost function and still allows us to effectively learn!

The weights are squared because we don't care if they're positive or negative, we just want the magnitude to be small... but we can't just use the absolute value because we'll need to take the derivative of this term. The above regularization term is the usual term used for logistic regression, but other regularization terms can be added instead.

Now we'll need to take the gradient of the cost function. Consider what that looks like with respect to one example:

$$\nabla J_{\vec{x}_i}^R(\vec{w}) = (h_{\vec{w}}(\vec{x}_i) - y_i)\vec{x}_i + \lambda\vec{w}^*$$

Note that $\vec{w} = [w_0, w_1, \dots, w_p]$. Why does \vec{w} start at 0? w_0 is known as the *bias*, which we can think of as the y-intercept of our data. We won't regularize the w_0 term. Why don't we want to regularize w_0 ? If we make the bias small, we'll lose this information that's important to the model; we can think of the bias as the offset of the data from the origin, and that's still relevant to the model. So in the above, we define \vec{w}^* as:

$$\vec{w}^* = \begin{bmatrix} 0 \\ w_1 \\ w_2 \\ \cdot \\ \cdot \\ \cdot \\ w_p \end{bmatrix}$$

Since \vec{x}_i has a length of $p + 1$ we need to make sure that the weight vector matches, which is why we include the 0 value in that term.

What would the weight updates look like as part of the stochastic gradient descent algorithm?

$$\vec{w} \leftarrow \vec{w} - \alpha \left[(h_{\vec{w}}(\vec{x}_i) - y_i)\vec{x}_i + \lambda\vec{w}^* \right]$$

Note that we now have two hyperparameters, both the α representing the step size (how fast we update the weights) and λ representing how much we want to regularize the weights at each step. Considering just the terms in front of \vec{w} , we have the 1 and $\alpha\lambda$ coefficients. Multiplying those by the weights will decrease the magnitude of the weights, since the $\alpha\lambda$ term will always be small and positive:

$$\vec{w} \leftarrow (1 - \alpha\lambda)\vec{w} + \dots$$

Consider handout 10 question 1: determining the derivative of $g(x)$ the logistic function.

$$\begin{aligned} g'(z) &= -1(1 + e^{-z})^{-2}e^{-z} \\ &= \frac{e^{-z}}{(1 + e^{-z})(1 + e^{-z})} \end{aligned}$$

Using the trick of adding 0 (on the top right below), we get:

$$= \frac{1}{1 + e^{-z}} \left(\frac{e^{-z} + 1 - 1}{1 + e^{-z}} \right)$$

$$= g(z)(1 - g(z))$$

It's very useful here that we don't need to know the value of the derivative, but can just directly use the value of the function. What happens as $z \rightarrow \infty$? The derivative goes to 0. The same is true as it approaches $-\infty$. This means that we don't need to update weights anymore, which will allow us to stop gradient descent.

We'll come back to this in just a moment.

2.1 Fairness Regularization

Recall that we looked at per-demographic group confusion matrices in a previous class. How could we incentivize fairer outcomes using regularization?

We'll represent our features as X , our labels as $y \in \{0, 1\}$ where 1 represents the desired outcome (e.g., hired, admitted to college, given a loan, predicted low-risk of criminal reoffense, etc.) and 0 is the undesired outcome. Now we'll also assume that we have some sensitive attributes $A \in \{0, 1\}$ such as race or sex, where our goal is to make sure the outcomes are fair with respect to these groups. We'll denote the protected class by 1 and the unprotected class by 0. By "protected" we mean the group that has been historically marginalized, such as women. We'll still denote our prediction $\hat{y} \in \{0, 1\}$ and we want the prediction to be accurate with respect to y and fair with respect to A . We can already see how this may lead us to a dual objective function where we have two pieces we're concerned with.

We'll talk about two fairness metrics we're concerned with: demographic parity and error rate balance. Demographic parity focuses on the outcomes—just the predictions—but don't consider those with respect to the true labels. It's not concerned with "accuracy" but rather with the predicted outcomes;

$$DP = \frac{p(\hat{Y} = 1 | A = 1)}{p(\hat{Y} = 1 | A = 0)}$$

Our demographic parity fairness goal is that this ratio is equal to 1.

We'll also consider *equalized odds*, which now takes into account the true label from the test data:

$$\frac{p(\hat{Y} = 1 | A = 1, Y = y)}{p(\hat{Y} = 1 | A = 0, Y = y)} \text{ for } y \in \{0, 1\}$$

In the above, $y = 0$ gives the false positive rate per demographic group while $y = 1$ gives the true positive rate per demographic group. In some cases, we might focus only on the $y = 1$ case in the cost function. In this case, again we would want the ratio to be roughly equal to 1 as our fairness goal.

How can we add this into our cost function? We need a regularization term. Letting $h_{\vec{w}}$ be the logistic regression function and D being the training data, our regularization term for demographic parity is:

$$R(h_{\vec{w}}, D) = 1 - p(\hat{Y} = 1 | A = 1)$$

We want this regularization term to be small. Note that this means that if the probability is high, which would be good for fairness, the term will be small, as desired. Our new cost function will simply have this term in addition to the original cost function $J(\vec{w})$:

$$J^R(\vec{w}) = J(\vec{w}) + R(h_{\vec{w}}, D)$$

We can think of the terms in this cost function $J^R(\vec{w})$ as representing both accuracy ($J(\vec{w})$) and fairness ($R(h_{\vec{w}}, D)$).

How do we estimate the fairness regularization term? I'll use a counting approach like we have before. We'll think about all the examples that have the protected attribute and what our predictions were for those examples. That will give us:

$$1 - \frac{1}{|D_1|} \sum_{x \in D_1} p(\hat{y} = 1 | \vec{x})$$


Let D_1 be all the examples in the protected group ($A = 1$). (NB: On the board in class we denoted this D_0 , but for these notes we'll use D_1 .) Note that as usual our normalization term makes sure these terms are between 0 and 1. Plugging in our model, we have:

$$= 1 - \frac{1}{|D_1|} \sum_{\vec{x} \in D_1} h_{\vec{w}}(\vec{x})$$

Taking the derivative of this with respect to \vec{w} , recalling that $h_{\vec{w}}(\vec{x}) = \frac{1}{1+e^{-\vec{w} \cdot \vec{x}}}$, and recalling that $\nabla h_{\vec{w}}(\vec{x}) = h_{\vec{w}}(\vec{x})(1 - h_{\vec{w}}(\vec{x}))\vec{x}$, we need to add this regularization term gradient to our stochastic gradient descent update. Recall that the weight update is done with respect to a specific example.

Overall, this gives us a gradient update step that now has the gradient for the regularization term included only for the cases when $A = 1$ for the specific x_i considered. Note that it'll be updated as usual for $A = 0$. This gives us a weight update step as below:

$$\vec{w} \leftarrow \begin{cases} \vec{w} - \alpha \left[(h - y)\vec{x} - \frac{1}{|D_1|} (h_{\vec{w}}(\vec{x})(1 - h_{\vec{w}}(\vec{x}))\vec{x}) \right] & \text{if } A = 1 \text{ for } x_i \\ \vec{w} - \alpha \left[(h - y)\vec{x} \right] & \text{if } A = 0 \text{ for } x_i \end{cases}$$

Gradient of the fairness regularization term


This term effectively penalizes negative predictions for the protected class, with the goal of increasing the number of members of that marginalized group that get the positive outcome, while not modifying the weight for the unprotected class.

A similar technique can be used to create a regularization term for the per-demographic true positive rate, but where the counting approach only considers members of the protected class who had a positive outcome label in the training data.