CSE 475: Statistical Methods in AI **Monsoon 2019**

SMAI-M-2019 12: Logistic Regression

Lecturer: C. V. Jawahar *Date: DATE*

In the last lecture, we saw that the loss functions that measures simple classification error rates are difficult to optimize. We saw perceptron algorithm as a solution. We will also look at another solution today — Logistic Regression. A very popular solution to the classification problem..

Logistic Regression (LR) is in fact a classification scheme. (Q: Then why is it called regression?) LR introduces an extra nonlinearity $g()$ (called a logistic function or sigmoid) over our familiar $f(\mathbf{w},\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. i.e.,

$$p(y=1|\mathbf{x}) = g(\mathbf{w^Tx}) = \frac{1}{1+e^{-\mathbf{w^Tx}}}$$

$$p(y=0|\mathbf{x}) = g(-\mathbf{w^Tx}) = \frac{1}{1+e^{\mathbf{w^Tx}}}$$

One can consider this as the posterior probability also.

- Q: Do they sum up to 1? Note that the denominators are not identical. There is a minus sign.
- Q: Plot the $g(\mathbf{w}^T\mathbf{x})$ function against $\mathbf{w}^{\mathbf{x}}$.
- Q: Assume we had a scalar $\alpha \in [0,1]$ how does the graph of $g(\mathbf{w^Tx}) = \frac{1}{1+e^{-\alpha\cdot\mathbf{w^Tx}}}$ change ? When does it come and and come close to a step function?

12.71.1 Classification

Let us predict class ω_1 if $f(\mathbf{w},\mathbf{x}) \geq 0.0$ and ω_2 if $f(\mathbf{w},\mathbf{x}) < 0.0$. Classification rule corresponding to this is:

$$= \begin{cases} +1 & \text{when } g(\mathbf{w}^T\mathbf{x}) > 0.5 \text{ or } \mathbf{w}^T\mathbf{x} > 0 \\ -1 & \text{when } g(\mathbf{w}^T\mathbf{x}) \leq 0.5 \text{ or } \mathbf{w}^T\mathbf{x} \leq 0 \end{cases}$$

Or else from a probabilistic view, it is to decide to class ω_1 or ω_2 depending on $p(y=1|\mathbf{x})$ is smaller or larger compared to $p(y=0|\mathbf{x})$.

$$p(y=1|\mathbf{x}) = g(\mathbf{w^Tx}) = \frac{1}{1+e^{-\mathbf{w^Tx}}}$$

$$p(y=0|\mathbf{x}) = g(\mathbf{w^Tx}) = \frac{1}{1+e^{\mathbf{w^Tx}}}$$

Note that the decision boundary is still $\mathbf{w}^T\mathbf{x} = 0$. Then what changes in logistic regression?

Here, the loss is based on a nonlinear (sigmoid) function.

12.71.2 Closer Look

Let us look at this problem closely. By looking at the sign of $\mathbf{w}^T\mathbf{x}$, we predict the class labels. Let us make a minor change. Let us look for a classifier that predicts the probability of the instance in the class i.e., $p(y|\mathbf{x_i})$. Let us look for a classifier $p(y=1|\mathbf{x}_i) = \phi(\mathbf{w},\mathbf{x}_i) = g(\mathbf{w}^T\mathbf{x}_i)$.

$$\phi(\mathbf{w},\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}} \tag{12.22}$$

Note that:

- $0 \leq \phi(\mathbf{w},\mathbf{x}) \leq 1$
- $p(y=0|\mathbf{x};\mathbf{w}) + p(y=1|\mathbf{x};\mathbf{w}) = 1$

Note that:

$$1.0 - \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}} = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}} = \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}}$$

12.72 Logistic Function

Recollect that our difficulty was directly optimizing an error function which is defined based on the *sign*() or step function. We already saw how perceptrons addressed this. Another way we can handle this is by changing the step function used to a "smooth step function" or a logistic function as

$$g(x) = \frac{1}{1+e^{-x}} \tag{12.23}$$

This change from zero to one. The transtion from 0 to 1 takes place closer to the origin.

Logistic function is also known sometime as "S" function. (due to the shape) Logistic function, also called sigmoid function is defined as:

$$g(z) = \frac{1}{1+e^{-z}} = \frac{1}{2} + \frac{1}{2}\tanh(\frac{z}{2})$$

A nice useful property of this function is that its first derivative can be expressed in its own terms:

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

- Q: Do verify this.

12.72.1 Sigmoid and Tanh

Both sigmoid and tanh are two popular nonlinear functions in machine learning. We will see them also as popular activation functions when we talk about neural networks and neuron models.

- In the simplest form, both are smoothed versions of the step functions/discontinuities.
- Sigmoid is in the range $[0, 1]$ while tanh is in the range $[0, 1]$.

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

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- Q: Verify $tanh(z) = 2 \cdot g(2z) - 1$

12.73 Loss Function

12.73.1 A Simple Loss Function

One can get tempted to use the same loss function here as:

$$J(\mathbf{w}) = \frac{1}{2N} \sum_{i=1}^{N} (g(\mathbf{w}^T \mathbf{x}_i) - y_i)^2$$

Unfortunately, this is a poor choice of loss function. This is non-convex.

- Question: Prove or Verify that.

Note: we assumed here $y_i \in \{0, 1\}$

12.73.2 MLE based Loss Function

Let us use a cost function from Maximum Likelihood Estimate (MLE) in this case.

Likelihood of the data is given by:

$$l(\mathbf{w}) = \prod_{i=1}^{N} p(y_i | \mathbf{x_i}; \mathbf{w})$$

Looking for $\mathbf{w}$ that maximizes the likelihood

$$\mathbf{w}_{MLE} = \arg\max_{\mathbf{w}} \; l(\mathbf{w}) = \arg\max_{\mathbf{w}} \prod_{i=1}^{N} p(y_i | \mathbf{x_i}; \mathbf{w})$$

(If you are not familiar with the notation, $\prod$ is product, just like $\sum$ is sum.)

The popular trick in formulations like this is to do two transformations of the objective function:

- optimize log of the function instead of the function directly. This converts the multiplication to addition. (note: $\log(ab) = \log(a) + \log(b)$)
- Take negative. It converts the maximization problem to a minimization problem.

12.74 Loss Fn with $y_i \in \{0, 1\}$

Let us look at the loss function with the convention/notation of $y_i \in \{0, 1\}$.

$$p(y = 1 | \mathbf{x}) = g(\mathbf{w}^T \mathbf{x})$$

$$p(y = 0 | \mathbf{x}) = 1 - g(\mathbf{w}^T \mathbf{x})$$

or in a compact manner by combining these two

$$p(y | \mathbf{x}) = g(\mathbf{w}^T \mathbf{x})^y (1 - g(\mathbf{w}^T \mathbf{x}))^{1-y} \tag{12.24}$$

Assuming independence in the data/samples, likelihood is

$$\prod_{i=1}^{N} g(\mathbf{w}^T \mathbf{x_i})^{y_i} (1 - g(\mathbf{w}^T \mathbf{x_i}))^{1-y_i}$$

Taking log, taking negative, we get the objective for the minimization problem as

$$\sum_{i=1}^{N} [y_i \log(g(\mathbf{w}^T \mathbf{x_i})) + (1 - y_i) \log(1 - g(\mathbf{w}^T \mathbf{x_i}))]$$

12.75 Loss Fn with $y_i \in \{-1, +1\}$

We can also rewrite the objective with the $y \in \{-1, +1\}$ convention.

$$p(y = +1 | \mathbf{x}) = g(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

$$p(y = -1 | \mathbf{x}) = 1 - g(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

We can combine both into single expression as

$$p(y_i | \mathbf{x_i}) = \frac{1}{1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}}$$

Assuming independence, the likelihood is

$$\prod_{i=1}^{N} \frac{1}{1 + e^{-y_i \mathbf{w}^T \mathbf{x_i}}}$$

Then the negative log likelihood is

$$\sum_{i=1}^{N} \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x_i}})$$

Explanation of the Objective

- when the sample is classified correctly, $-y_i\mathbf{w}^T\mathbf{x_i}$ is negative and $\log(1+e^{-y_i\mathbf{w}^T\mathbf{x_i}})$ is nearly zero.

- when the sample is wrongly classified, $-y_i\mathbf{w}^T\mathbf{x_i}$ is positive and $\log(1+e^{-y_i\mathbf{w}^T\mathbf{x_i}})$ is large.

Question: provide an intuitive explanation why the loss function with $\{0,1\}$ is also equally good. One can plot the individual loss functions and see that, if $y=1$, cost is zero if the prediction is correct. if prediction is close to zero, the cost increases to ∞. Similarly one can see for $y=1$. Larger mistakes get larger penalties.

12.76 Regularization

It is common to regularize the loss function with an additional term.

$$J_R(\mathbf{w}) = J(\mathbf{w}) + \lambda\sum_{i=1}^{d} w_j^2$$

The regularized objective function J_R has an extra term, compared to the original one.

Adding an extra term has many advantages:

- This can avoid overfitting, which is a major concern for us.

- With regularization, we prefer certain type of solutions over other. For example, we encourage, simpler solutions over complex solutions for better "generalization".

- It is common to add an extra term which is the norm of $\mathbf{w}$. If we choose a norm that measures the number of non-zero elements, then while finding the "best" $\mathbf{w}$, we also find one which is sparse. (Q: which norm measures the number of non-zero? Q: How do we minimize such a loss? Read about LASSO and Ridge regression.

- In the GD framework that we use, it is common to add the L2 norm which leads to an additional quadratic term in the objective and then a linear term in the update rule. Since this is an "addition", it does not complicate the derivation/update.

12.77 Gradient Descent Solution

Q: Derive GD update rule and write pseudo code for LR and Regularized LR for (i) single sample , (ii) batch and (iii) mini batch SGD variations. ($3\times 2=6$ algorithms.)

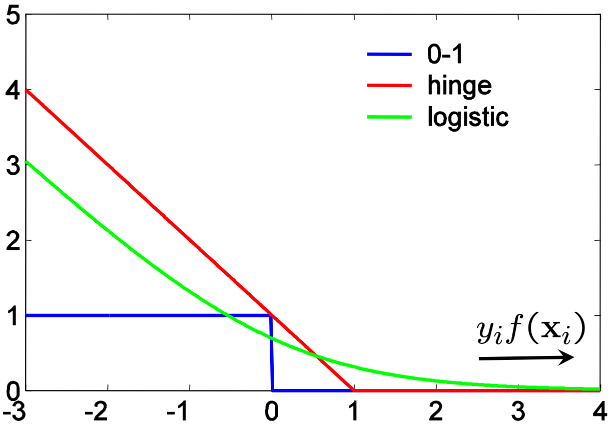

Figure 12.7: Comparison of different losses

12.78 Discussions

Logistic regression is a popular classification scheme. Let us understand the loss in comparison with other methods.

You will see another popular scheme later Support Vector Machine (SVMs). This also optimizes a very similar objective function. Let us write the objective function corresponding to both these schemes in a very similar manner.

SVM:

$$\min_{\mathbf{w}} C\sum_{i=1}^{N}\max(0,1-y_i f(\mathbf{x_i})) + ||\mathbf{w}||^2$$

Logistic Regression

$$\min_{\mathbf{w}}\sum_{i=1}^{N}\log(1+e^{-y_i f(\mathbf{x_i})}) + \lambda||\mathbf{w}||^2$$

Both objective functions balance the relative importances with an additional term C VS λ. Parameters like this balance the relative importance of terms in an objective function. You may want to guess them right. But not very difficult in most cases.

Let us plot and see how different loss functions look like in the figure 12.7. You can see that both SVM and LR have very similar loss functions. One more smooth than the other.

12.78.1 Margin

In the perceptron algorithm, we had observed that the iterative algorithm terminates when $\mathcal{E}$ is empty. It does not look for a "good" solution; rather it looks for a "valid" solution.

We can see that the MLE based objective also encourage to have a rather large γ (see the notation in perceptron) than a small that perceptron could land up with.

x

12.78.2 Fast and Interpretable

LR is a simple algorithm at the test time. It classifies with $sign(\mathbf{w}^{\mathbf{x}})$. This makes it very efficient at test time, and also scalable.

Far more, w^j tells us the importance of the feature x^j. This also makes it more "interpretable". We can say how much each feature is contributing to the decision or when a decision is made (example a loan is rejected or a person is rejected from entering into a country), we know what feature contributed how much to this decision.

12.79 Multiclass Extension

Most of our discussions till now have been on binary classification. i.e., when the number of classes is 2. However, many practical problems demand more than two classes, say K classes. Can logistic regression be extended for this purpose?

We know that:

$$p(y=1|\mathbf{x};\mathbf{w}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}}$$

$$p(y=0|\mathbf{x};\mathbf{w}) = 1 - p(y=1|\mathbf{x};\mathbf{w})$$

$$= \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}} = \frac{e^{-\mathbf{w}^T\mathbf{x}}}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

Numerator is the weight/confidence of the sample being that class and denominator is the sum of weights for all classes. This allows us to extend to the multiclass setting as

$$p(y=c|\mathbf{x};\mathbf{w_1},\mathbf{w_2},\mathbf{w_K}) = \frac{e^{\mathbf{w_c}^T\mathbf{x}}}{\sum_{i=1}^{K} e^{\mathbf{w_i}^T\mathbf{x}}}$$

Note that the sum of probabilities across all classes sum upto one. Finally a sample is classified into

$$\arg\max_i p(y=i|\mathbf{x})$$

This is also called popularly as **softmax**. Very popular in the modern deep learning architectures.

Q: How does this lead to an objective/loss and a computational procedure for multi-class classification?