CSE 475: Statistical Methods in AI **Monsoon 2018**

Lec 10: Linear Methods-VI

Lecturer: C. V. Jawahar *Date: Sep. 3, 2018*

10.25 Logistic Regression

In the last lecture, we started with logistic regression (LR). 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 $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. i.e.,

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

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

One can consider this as the posterior probability also. (Q: Do they sum up to 1?)

Classification rule corresponding to this is:

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

Note that the decision boundary is still $\mathbf{w}^T\mathbf{x} = 0$. However, the loss is based on a nonlinear (sigmoid) function.

10.26 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.

10.27 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

$$p(y|\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})^y(1-g(\mathbf{w}^T\mathbf{x}))^{1-y} \tag{10.19}$$

Assuming independence in the data/samples, likelihood is

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

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

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

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

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

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

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

We can combine both into single expression as

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

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

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

- 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.

Q: 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.

10.29 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.

10.29.1 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.)

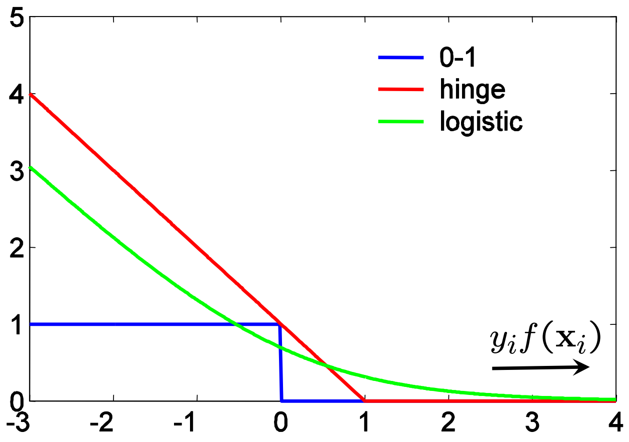

Figure 10.2: Comparison of different losses

10.30 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 10.2. You can see that both SVM and LR have very similar loss functions. One more smooth than the other.

10.31 Multiclass Extension

We know that:

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

$$
\begin{aligned}
p(y=0|\mathbf{x};\mathbf{w}) &= 1 - p(y=1|\mathbf{x};\mathbf{w}) \\
&= \frac{1}{1+e^{\mathbf{w}^T\mathbf{x}}} = \frac{e^{-\mathbf{w}^T\mathbf{x}}}{1+e^{-\mathbf{w}^T\mathbf{x}}}
\end{aligned}
$$

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.

10.32 Other Multiclass Extensions

We had seen binary classification schemes. You will see more of them. How do we extend them to a multiclass setting in general?

Let us consider we have K classes.

- We can build K one Vs rest classifiers.
- We can build ${}_KC_2$ pairwise classifiers.

How do we arrive at a final classification decision?

If the K one vs rest classifiers can provide probabilities, life is simple as we had seen in the case of softmax. Pick the class with highest probability.

If probabilities are not available (and only class label is predicted), then we may have two cases to worry. All classifiers says "rest". or multiple classifiers assign a class-label. In either case, the final decision is difficult without some additional information.

When there are ${}_KC_2$ classes, one can use simple majority voting to find the best classifier. One can also use DAG (directed acyclic graph) like structure.