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

Lec 8: Linear Methods-IV

Lecturer: C. V. Jawahar *Date: Aug. 27, 2018*

8.15 GD for Classification

We know the gradient descent equation as

$$\mathbf{w}^{n+1} \leftarrow \mathbf{w}^n - \eta \nabla J \tag{8.11}$$

We start with an initialization $\mathbf{w}^0$ and update the weights/vector until we get the separating hyperplane (or until it converges).

Let us specially focus on the classification problem today. We wish to use a loss that measures classification accuracy. That is, the percentage of samples misclassified.

Let us now consider the objective as

$$J = \frac{1}{N}\sum_{i=1}^{N}(1 - y_i \cdot f(\mathbf{x_i})) \tag{8.12}$$

Where $y_i \in \{-1, +1\}$, and $f(\mathbf{x_i}) = \mathbf{w}^T\mathbf{x_i}$ This provides a "unit" loss for all the misclassifications and "zero" loss for all the correct classifications. (see notes elsewhere for details.) Our ideal goal could have been to minimze this loss.

However we have a serious problem. This is not differentiable. So what do we do? We can address this in two different ways. This leads to two popular algorithms:

- Perceptron Algorithm
- Logistic Regression

8.16 Perceptron Algorithm

Perceptron algorithm makes the assumption that all the samples are *linearly separable*. In other words $\exists \mathbf{w}$ such that $y_i f(\mathbf{x_i}) = +1 \ \forall i$.

Let us now look for an alternate loss function. We modify the loss as quantity as

$$J = \sum_{x_i \in \mathcal{E}} -y_i \mathbf{w}^T \mathbf{x_i} \tag{8.13}$$

where $\mathcal{E}$ is the set of misclassified samples.

How is this different? The summation is only over the misclassified samples. This does not change anything really. (We had zero loss for all the correctly classified ones anyway). We then have a "non-unity" loss for all the misclassified ones. (This is different from the previous loss). It is proportional to how far it is from the line/plane/hyperplane. The farther the sample is the more the loss is. (indeed, not very ideal!!). Q: Why is not ideal?

Pleasantly, this is now differentiable. That is an advantage. Also when all samples are correctly classified (when the problem is linearly separable), the loss becomes zero ($\mathcal{E}$ becomes empty set.) and our algorithm will converge (loss is zero, derivative is also zero.)

There is also an advantage with this loss. A sample that is far from the line/plane will pull/push/rotate the line/plane more than one that is very near the line. This will help in faster convergence.

This leads to the gradient descent equation as:

$$\mathbf{w}^{n+1} \leftarrow \mathbf{w}^n + \eta \sum_{\mathbf{x_i} \in \mathcal{E}} y_i \mathbf{x_i} \tag{8.14}$$

Another way we could define the weight update is with the help of desired/target (t) and output (o). Note that when desired and predicted outputs are same, $t - o$ is zero. Else it is either Positive or negative (when y_i as well as $t - o$ will have the same sign when the sample is misclassified.)

$$\mathbf{w}^{n+1} \leftarrow \mathbf{w}^n + \eta \sum_{i=1}^{N} (t_i - o_i)\mathbf{x_i} \tag{8.15}$$

Note that η is a learning rate and it could absorb any scaling. (η in all these equations need not be identical, see yourself, if they differ by a scale factor.). Here the summation is over all the samples. Note that, this does not change the update rule. The additional terms are zero.

Algorithm now has the following steps:

1. Initialize $\mathbf{w}$, k=0
2. We update the $\mathbf{w}$ as

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \eta \sum_{\mathbf{i=1}}^{N} (t_i - o_i)\mathbf{x_i}$$

or

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \eta \sum_{\mathbf{x} \in \mathcal{E}} y_i \mathbf{x_i}$$

3. $k \leftarrow k + 1$
4. Repeat steps 2-4
   - until $\mathcal{E}$ is empty. (the ideal termination criteria for perceptron algorithm) Or
   - until the change in weight is small (say less than θ).

8.17 Discussions and Examples

Let us consider some simple situations first and see how the perceptron algorithm behaves.

Let us simplify the update rule first. Assume $\eta = 1$. Let us assume that there is only one sample. Also let us assume that $\mathbf{x}$ has only one dimension i.e., scalar. This simplifies to:

$$w^{k+1} \leftarrow w^k + (t_i - o_i)x_i$$

Now let us consider some situations.

- When there is no misclassification. i.e., $t_i = o_i$. In this case, w does not change.
- Let us consider $t_i = +1$, $o_i = -1$ and x_i is positive. Sample is misclassified. Which means $w^k \cdot x_i$ is negative. (x_i is fixed and positive.) This means w^k is negative. We need to increase w^k to (positive to) minimize/avoid misclassification. If we substitute the values/signs in the above equation, we see that perceptron algorithm does this exactly.
- Now Let us consider $t_i = +1$, $o_i = -1$ and x_i is negative. Sample is misclassified. Which means $w^k \cdot x_i$ is negative. (x_i is fixed and negative.) Therefore, w^k is positive. We need to decrease w^k to minimize/avoid misclassification. If we substitute the values/signs in the above equations, we see that perceptron algorithm does this exactly.
- Q: Convince yourself for all other cases also.

8.17.1 Example/Problem in 2D

Q: Consider a set of vectors in 2D.

$$\{[1,1]^T, [1,3]^T, [2,1]^T, [2,2]^T,$$

$$[-1,-1]^T, [-1,-3]^T, [-2,-1]^T, [-2,-2]^T$$

The first four are from class 1 and the rest four are from class 2.

- Plot the samples in a 2D plane with "×" for positive classes and "○" for negative class. Is this set linearly separable?
- Start with a random vector for $\mathbf{w}$ and show that the perceptron algorithm converges to a separating line/plane for different values of η. (make sure that you start with a vector that has error!!).
- Why is that the final answers are different for different initializations/η? Then which is the best solution?

8.18 Some Theoretical Results

Perceotron algorithm has a number of interesting theoretical properties/results. A summary is below.

- If there exist a set of weights that are consistent with the data, the perceptron algorithm will converge.
- If the training data is not Linearly Separable, the perceptron algorithm will eventually repeat the same set of weights and thereby enter an infinite loop.
- If the training data is linearly separable, algorithm will converge in a maximum of M steps. (Q:Find M).
- Every boolean function can be represented by some network of perceptrons only two levels deep.

8.19 Neural Networks View Point

A popular view point of the perceptron is to appreciate it as a "neuron". Though the story has origins in our attempts to understand and reverse engineer human brain and perception skills, it is much easy to appreciate it as a powerful mathematical model at this stage.

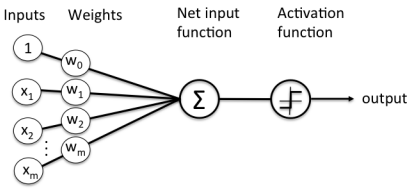
A neuron accepts multiple inputs. Weigh each one. Add the inputs. Pass through a nonlinearity. In the case of perceptron, the nonlinearity is a step like nonlinearity. See also figure.

We will revisit the neural network view point at a later stage when we discuss "multi layer perceptrons".

8.20 Variations in Gradient Descent

We now know: (i) How to define a loss function (ii) how to optimize it with gradient descent update equations.

There are in fact many variations in the implementation.



Schematic of Rosenblatt's perceptron.

Figure 8.1: A pictorial representation of the neuron. Often a single large circle is shown for a neuron wherein the weighted addition and nonlinearity is combined.

- Single sample or online version: which assumes that samples come one by one. This computes gradient per sample and update for each sample. A sample may be seen multiple times (in a cyclic manner).
- Batch version: Compute the gradient for the batch (full set) and update once.
- Mini-Batch: Instead of taking the full batch, take a smaller batch. This is useful when the data is large.
- Stochastic: Samples in online or mini-batch are selected randomly.

Q: Do write the pseudo code for each of these. Make sure your pseudo code is very close to the programming language that you use.