Fundamentals of Machine Learning: Theory

Author: Darien Nouri

Topics Covered

Theory

1. **Question T1: Empirical vs. Expected Cost**

   We approximate the true cost function with the empirical cost function defined by:

   $$\mathbb{E}[E(g(x), f(x))] = \frac{1}{N}\sum_{i=1}^{N} E(g(x_i), y_i) \tag{1}$$

   where N is the number of training samples, f is the unknown function, g is the learnable function, y_i is the label associated with the input x_i. In the above equation is it okay to give an equal weight to the cost associated with each training example? Given that we established that not every data x is equally likely, is taking the sum of all per-example costs and dividing by N reasonable? Should we weigh each per-example cost differently, depending on how likely each x is? Justify your answer.

   Answer:

   The traditional empirical cost function as given in Equation (1) assumes uniform importance for all training examples. However, if the data has different likelihoods or if certain data points are more crucial to learn from, then a differential weighting scheme may be more appropriate.

   One way to incorporate differential weighting is to modify the empirical cost function to include weights w_i for each data instance i. For a weighted empirical cost function where w_i reflects the importance or likelihood of the data point x_i. If a data point x_i is more likely or deemed more important, w_i would be larger, hence that data point would contribute more to the cost. This way, the learnable function g is driven to minimize the cost more aggressively for more important or likely data points. This approach can be benifitial in the case where informative but sparse data exists in the feature set. This allows for the empirical cost function to be more representative of the true cost function.

2. **Question T2: Perceptron Learning Algorithm**

   The weight update rule of the Perceptron Learning Algorithm (PLA) is given by:

   $$w(t+1) \leftarrow w(t) + y(t)x(t) \tag{2}$$

   Prove the following statements:

   (a) Show that $y(t)w^T(t)x(t) < 0$ (2 points)

   (b) Show that $y(t)w^T(t+1)x(t) > y(t)w^T(t)x(t)$ (4 points)

(c) Argue that the move from $w(t)$ to $w(t+1)$ is the right move as far as classifying $x(t)$ is concerned. (4 points)

part a:

To prove that $y(t)w^T(t)x(t) < 0$:

The PLA only updates the weights when there's a misclassification. Consider the following two cases:

1. *Positive Prediction, but Negative True Label:* If the prediction is positive, $w^T x(t) > 0$, but the true label is negative, $y(t) < 0$, then their product $y(t)w^T(t)x(t)$ will be negative.

2. *Negative Prediction, but Positive True Label:* If the prediction is negative, $w^T x(t) < 0$, but the true label is positive, $y(t) > 0$, then their product $y(t)w^T(t)x(t)$ will also be negative.

From both cases, the following holds true:

$$y(t)w^T(t)x(t) < 0$$

part b:

Starting from the weight update rule, substituting $w(t+1)$ with $w(t) + y(t)x(t)$ gives:

$$y(t)w^T(t+1)x(t) = y(t)[w^T(t) + y(t)x(t)^T]x(t)$$

Expanding:

$$= y(t)w^T(t)x(t) + y(t)^2 x(t)^T x(t)$$

Given that $y(t)^2$ is always positive and $x(t)^T x(t)$ is a measure of the squared length of $x(t)$ and is also positive, the entire term is positive. This can be interpreted such that the new weight vector $w(t+1)$ has a larger dot product in the direction of the correct label $y(t)$ than the previous weight $w(t)$. Therefore:

$$y(t)w^T(t+1)x(t) > y(t)w^T(t)x(t)$$

part c:

The Perceptron Learning Algorithm works to correct misclassifications. We need to show that the update rule

$$w(t+1) = w(t) + y(t)x(t)$$

correctly adjusts the weights in the event of a misclassification of $x(t)$.

From (a):

$$y(t)w^T(t)x(t) < 0$$

This implies that $w(t)$ misclassifies the input $x(t)$ given its true label $y(t)$.

After the weight update, we have:

$$y(t)w^T(t+1)x(t) > y(t)w^T(t)x(t)$$

This shows us that the dot product of the updated weight with $x(t)$ is more aligned in the direction of the true label $y(t)$ than before.

Considering the following, if $x(t)$ is misclassified, the angle θ between $w(t)$ and $x(t)$ will be greater than 90 degrees for $y(t) = 1$ or less than 90 degrees for $y(t) = -1$. The update shifts $w(t)$ closer to $x(t)$ in the direction specified by $y(t)$, effectively adjusting the angle θ.

Therfore, the move from $w(t)$ to $w(t+1)$ enhances the alignment of the weight vector with the correct classification direction, making it the right move for classifying $x(t)$.

3. **Question T3: Gradient of Logistic Regression**

The logistic regression loss for a single sample (x, y) can be written as

$$L_w(x,y) = -\left[y \cdot \log\sigma(wx) + (1-y)\cdot\log(1-\sigma(wx))\right] \tag{3}$$

where $\sigma(s)$ is the logistic function and w are the parameters of the model. Compute the gradient of the above loss function with respect to the parameter vector w. Show all the steps of the derivation.

Answer:

Given the logistic regression loss for a single sample (x, y):

$$L_w(x,y) = -\left[y \cdot \log\sigma(wx) + (1-y)\cdot\log(1-\sigma(wx))\right]$$

where

$$\sigma(s) = \frac{1}{1+e^{-s}}$$

To compute the gradient, we first need the derivative of $\sigma(s)$ with respect to s:

$$\frac{d\sigma(s)}{ds} = \sigma(s)(1-\sigma(s))$$

This is derived as:

$$\frac{d\sigma(s)}{ds} = \frac{d}{ds}\left(\frac{1}{1+e^{-s}}\right) = \frac{e^{-s}}{(1+e^{-s})^2} = \sigma(s)(1-\sigma(s))$$

Now, differentiating the loss function with respect to w:

$$\frac{dL_w(x,y)}{dw} = -\frac{d}{dw}\left[y \cdot \log\sigma(wx) + (1-y)\cdot\log(1-\sigma(wx))\right]$$

Using the chain rule:

$$\frac{dL_w(x,y)}{dw} = -\left[y \cdot \frac{d}{dw}\log\sigma(wx) + (1-y)\cdot\frac{d}{dw}\log(1-\sigma(wx))\right]$$

For the term $\frac{d}{dw}\log\sigma(wx)$:

$$\frac{d}{dw}\log\sigma(wx) = \frac{1}{\sigma(wx)}\cdot\sigma(wx)(1-\sigma(wx))\cdot x = x(1-\sigma(wx))$$

For the term $\frac{d}{dw}\log(1-\sigma(wx))$:

$$\frac{d}{dw}\log(1-\sigma(wx)) = -\frac{1}{1-\sigma(wx)} \cdot \sigma(wx)(1-\sigma(wx)) \cdot x = -x\sigma(wx)$$

Inserting these results into our main differentiation:

$$\frac{dL_w(x,y)}{dw} = -\left[xy - xy\sigma(wx) - x\sigma(wx) + xy\sigma(wx)\right] = -[xy - x\sigma(wx)]$$

Thus, the gradient is:

$$\nabla_w L_w(x,y) = x[\sigma(wx) - y]$$