EE603: Lecture 14 Notes

Aashish Patel & Prateek Jain

October 2021

1 Non-Linear Classification

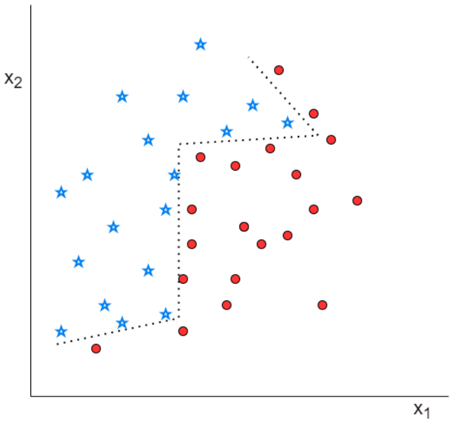

Figure 1: A non-linear classifier

1.1 Multiclass Classification

- **Definition:** Out of multiple given classes, only one class is true. We want our classifier to detect which class is true
- **Input**: Let us call it vector **x**
- **Output:** If there are C classes, the output of the model is **h**, or h_c, where c= 0,1,2,..., C-1. Apply softmax on h to get the final output **y**

$$\mathbf{y} = softmax(\mathbf{h})$$
$$y_c = \frac{exp(h_c)}{\sum_{c'} exp(h_{c'})}$$

- **Target:** Target is a one-hot vector of size C

- **Example:** The classification problem where we are given three kinds of flowers and we have to identify the type flower i.e. whether it is Rose or Lotus or Jasmine given height and width as the input Here C=3 and classes are Rose, Lotus, and Jasmine
- **Some Observations**
  - **Why exp in softmax?**
    - Derivative of **sigmoid** will be very small so it won't be able to learn properly. Basically if our error function is something that is not very steep i.e. it is very shallow then it won't learn very fast and there are also of noise etc.
    - But using an exponential function makes it steeper. Any differences are exaggerated by an exponential function(but they are suppressed by the sigmoid function).
    - Hence, if the derivative is greater than one the differences are exaggerated else if a derivative is less than one the differences are suppressed and sigmoid function derivative is very small therefore the differences are very much suppressed but on the other hand in the exponential function derivative is very large, hence differences are further magnified. That's is why the exponential function very well resembles the *Argmax* function.

    Takeaway: If we want to exaggerate the differences use an exponential function and if we want to suppress the differences, use a *sigmoid* or *logarithm* as *logarithm* also suppresses the difference.
- **Loss function**

  $$\sum_c y_c = 1 \textbf{ and } \sum_c t_c = 1$$

  We need a stronger pull to 0 or 1
  Categorical cross-entropy loss function can be used for training.

  $$E_{X_{ent}} = -\sum_c t_c \cdot log(y_c)$$
- **Some Observations**
  - **Why not use Mean Squared Error(MSE) as the loss function?**
    The derivatives are very small. The mean squared error leads to poor learning since it's a quadratic function and the derivatives are not sufficient to pull our weights to the right value, so we need a stronger pull to zero or one. Also, Mean squared error is good for the continuous approximation such as in regression problems but in the case of multi-class classification answers are discrete i.e. either 0 or 1 ideally. Therefore, we want a very strong pull to these discrete values that's why we don't use the mean squared error function as the loss function.

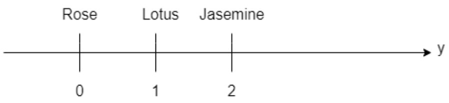

Figure 2: A representation of classes

- **Why can't we use single output {0,1,2} to represent the classes instead of using 3 neurons in the one-hot vector?**

  There is an inherent flaw in this representation: In the first representation (Figure 2), let us say we have roses, lotuses, and jasmine. Now, if the output y = {0,1, 2} , in other words if the detected class is rose, model will output y=1, if lotus output will be y=2 and for jasmine output y=3. This implies that rose is much more similar to lotus than to jasmine(since 1 is closer to to 2 than 3) and similarly, jasmine is much more similar to lotus than rose in this representation. But one-hot representation does not give any distance preference to any of the classes

1.2 Multilabel Classification

- **Definition:** To detect which attributes(labels) are present in an entity out of some given attributes is known as multilabel classification. These labels are independent of each other i.e. any of them can be true or false independent of the other labels.
- **Input**: Let us call it vector **x**
- **Output:** If there are total L labels, the output of the model is **h** or h_l where l= 0,1,2,...,L-1. Apply sigmoid on h to get the final output y

  $$\mathbf{y} = sigmoid(\mathbf{h})$$
  $$y_c = \frac{1}{1+e^{-h_l}}$$

- **Target:** Target is a multi-hot vector of size L
- **Example**: Face classification: A face may have different attributes. we want the classifier given a face as the input to detect whether those attributes are present or not

  i.e. we can define different binary attributes e.g. does a person have a beard, does a person wear glasses, does a person have hair, does a person have moustache, does a person have lipstick, It can be observed these attributes are independent of each other as a person with a beard can wear glasses or not wear glasses with equal probability

- **Loss function** Probabilistic interpretation:

  $$P(l = 1) = y_l$$
  $$P(l = 0) = 1 - y_l$$

  Binary cross-entropy loss function can be used for training.

  $$E_{binX_{ent}} = -\sum_l \{t_l \cdot log(y_l) + (1 - t_l) \cdot log(1 - y_l)\}$$

1.3 Difference between Multiclass Classification and Multilabel Classification

In multiclass classification, out of the given classes, only one class can be true at a time(i.e. an item can belong to anyone class only) whereas in multilabel classification, more than one label can be true simultaneously since they are independent of each other

In MCC, target = one hot vector
In MLC, target = multihot vector

Note: In both Multiclass Classification and Multilabel Classification, the hidden layers can have **tanh, ReLU, LeakyReLU** nonlinearity.

2 Formulae

1. **Softmax**

   - **Expression**

   $\sigma_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}=1}$

2. **Sigmoid**

   - **Expression**

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

   - **Derivative**

   $\sigma^{'}(z) = \sigma(z)$ **. (1-** $\sigma(z)$**)**

3. **Categorical Cross Entropy**

   - **Expression**

   $E_{X_{ent}} = -\sum_c t_c \cdot log(y_c)$

4. **Binary Cross Entropy**

   - **Expression**

   $E_{binX_{ent}} = -\sum_l \{t_l \cdot log(y_l) + (1-t_l) \cdot log(1-y_l)\}$

5. **Mean Squared**

   - **Expression**

   $MSE = \frac{1}{n} \cdot \sum_{i=1}^{n}(Y_i - Y_i^{'})^2$