

## Classification—Binary v.s Multiple

---- Statistics and Application

In general, Classification questions can be divided into 2 major parts: Binary and Multiple

>> **Binary Classification** can generally be 1, 0 (Yes, No)

>> **Multiclassification** can be: *multiclass classification, multilabel classification*

>> Besides, for multiclassification, there can also be another topic about *multioutput like multioutput regression*

**Multiclass Classification** >> emphasis in ‘classes’, mean to classify items into different groups and each item can only belong to one group. (*i.e. a cat can only be a cat instead of both cat and dog, it is likely to be 1 -> 1*)

**Multilabel Classification** >> emphasis in ‘labels’, which means it can manually be recognized to bear one or more certain fields of features based on different aspects of views. (*i.e. given a cat, we can attach it with the labels of mammal, pet, etc. it is likely to be certain-> general and 1-> more*)

**Multioutput Regression** >> emphasis in ‘output’, which means one status of output can bear multiple properties. (*i.e. given a cat, we can describe it with properties of breed, color, mass, etc. it is likely to be general -> detail and 1-> more*)

### Statistical Thinking

>> *Binary Classification: usually use logistic regression(bold means vectors):*

$$h_{\theta}(\mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

*Where, the statistical inference as follows:*

*Since it's a binary problem, we can transform it into the calculation of the probability of Bernoulli try Y with ‘happen’ as 1 and ‘not happen’ as 0, while  $\mathbf{x}$  as all regressor features, that is:*

$$Y \sim \text{Bernoulli}(p = \pi_i),$$

*where given a sample feature vector  $\mathbf{x}_i$ , then  $\pi_i = \pi_i(\mathbf{x}) = \Pr(Y_i = y | X_i = \mathbf{x}_i)$ , y can be 0 or 1*

*Then we make a transformation of  $\pi_i(\mathbf{x})$  so that it can fall within [0,1] just as the probability should be, and that transformation is actually ‘logit’ transformation, that is:*

$$\text{logit}(\pi_i) = \log\left(\frac{\pi_i}{1 - \pi_i}\right) = \theta_0 + \theta_1 x_{i1} + \cdots + \theta_j x_{ij}, \quad 1 \leq i \leq n, 1 \leq j \leq k$$

*express it in matrix:*

$$\text{logit}(\pi(\mathbf{x})) = \log\left(\frac{\pi(\mathbf{x})}{1 - \pi(\mathbf{x})}\right) = \boldsymbol{\theta}^T \mathbf{x}$$

Now, let's calculate to find  $\pi(\mathbf{x})$ :

$$\frac{\pi(\mathbf{x})}{1 - \pi(\mathbf{x})} = e^{\boldsymbol{\theta}^T \mathbf{x}}$$

$$\pi(\mathbf{x}) = \frac{e^{\boldsymbol{\theta}^T \mathbf{x}}}{e^{\boldsymbol{\theta}^T \mathbf{x}} + 1} = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

Since  $\pi(\mathbf{x})$  is a function with parameter  $\theta$ , we then write it as  $h_\theta(\mathbf{x})$ , and we get:

$$h_\theta(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

The loss function is based on log-likelihood, and that is :

$$l(\theta|\mathbf{x}) = \log L(\theta|\mathbf{x}),$$

Thus, our *log likelihood* loss function can be:

$$\text{cost}(h_\theta(\mathbf{x}), y) = \begin{cases} -\log(h_\theta(\mathbf{x})) & \text{if } y=1 \\ -\log(1-h_\theta(\mathbf{x})) & \text{if } y=0 \end{cases}$$

$$\gg \text{cost}(h_\theta(\mathbf{x}), y) = -y_i \log(h_\theta(\mathbf{x})) - (1 - y_i) \log(1 - h_\theta(\mathbf{x}))$$

.....

Multiple classification:

Multiclass Classification >> 3 ways:

1: To class from  $n$  class, we can generate  $n(n - 1)$  combinations of binary classification, going with the binary classification routine and then voting

2: Set up a baseline as reference class (0, 0, 0, ..., 0), then other  $(n - 1)$  classes can be (1, 0, 0, ...), (0, 1, 0, ...) ....

3: Use *Softmax* function

Softmax function is the generalized logistic

The distinguish function is:

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}$$

Log likelihood loss function:

$$C = - \sum_i y_i \log a_i$$

Thinking about integration thought in calculus, by assuming the outcomes of the multiclass classification are from so many neural network routines that the output probability from each certain neural net can be seen as to nearly follow a continuous distribution and that actually corresponds to probability density function (continuous), thus we get:

$$-\int p(x) \log(g(x)) dx$$

And it is the basic format for *Crossentropy loss function*

Multilabel Classification >>

As we know, the number of labels for an item is unsure, can be one, two, or more. However, given a set of pending labels, the number of total labels is for sure. Next, we will try to do +1 and -1 calculations: given an item, for each label, if it has such a label, +1, otherwise, -1. Then, we combine it with the label vector corresponded +1 with 1 and -1 with 0, so that we get matrixes for the item-label correspond information.

Here's an example:

i.e.

*assuming that we have 5 labels,*

*for a randomly sample x with vector defined as (-1, -1, +1, +1, -1)*

*then, we correspond it with label variable, then the correspond matrix would be:*

```
0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 0
```

Here, for this question, because it is not the general binary distribution any more, so that we cannot use the former softmax function to solve for it. However, here for each neural net, the last neural  $x_{ik}$  can still follow the binary property.

Thus for ith neural, the crossentropy can be:

$$y_i \log(h_\theta) + (1 - y_i) \log(1 - h_\theta)$$

$$C = -\sum_i \log a_i$$

$$cost(h_\theta(x), y) = \begin{cases} \log(a_i) & a_i = h_\theta, \text{ if } y_i = 1 \\ -\log(a_i) & a_i = 1 - h_\theta, \text{ if } y_i = 0 \end{cases}$$

| classifications | Activation Functions | Loss Functions           |
|-----------------|----------------------|--------------------------|
| Binary          | sigmoid              | binary_crossentropy      |
| Multiclass      | Softmax              | categorical_crossentropy |
| Multilabel      | sigmoid              | binary_crossentropy      |

Sklearn classification cheatsheet

**1->1**

*sklearn.svm.NuSVC*

*sklearn.svm.SVC.*

*sklearn.gaussian\_process.GaussianProcessClassifier (setting multi\_class = "one\_vs\_one")*

**1->more**

*sklearn.ensemble.GradientBoostingClassifier*

*sklearn.gaussian\_process.GaussianProcessClassifier (setting multi\_class = "one\_vs\_rest")*

*sklearn.svm.LinearSVC (setting multi\_class="ovr")*

*sklearn.linear\_model.LogisticRegression (setting multi\_class="ovr")*

*sklearn.linear\_model.LogisticRegressionCV (setting multi\_class="ovr")*

*sklearn.linear\_model.SGDClassifier*

*sklearn.linear\_model.Perceptron*

*sklearn.linear\_model.PassiveAggressiveClassifier*

**multilabel:**

*sklearn.tree.DecisionTreeClassifier*

*sklearn.tree.ExtraTreeClassifier*

*sklearn.ensemble.ExtraTreesClassifier*

*sklearn.neighbors.KNeighborsClassifier*

*sklearn.neural\_network.MLPClassifier*

*sklearn.neighbors.RadiusNeighborsClassifier*

*sklearn.ensemble.RandomForestClassifier*

*sklearn.linear\_model.RidgeClassifierCV*

**multiclass & multioutput:**

*sklearn.tree.DecisionTreeClassifier*

*sklearn.tree.ExtraTreeClassifier*

*sklearn.ensemble.ExtraTreesClassifier*

*sklearn.neighbors.KNeighborsClassifier*

*sklearn.neighbors.RadiusNeighborsClassifier*

*sklearn.ensemble.RandomForestClassifier*