==

ALGORITHM: Logistic Regression

In this lecture, we discuss a category of classification models that directly assign class label without computing class-conditional probabilities called *probabilistic discriminative models*. In particular, our topic of discussion is the *logistic regression*, which directly estimates the ***odds*** of a data instance x using its attribute values.

Considering the two class scenario, the basic idea of the logistic regression is to use a linear predictor, $z = w^T x + b$, for representing the ***odds*** of x as follows:

$$\frac{p(y_2|x)}{p(y_1|x)} = e^z = e^{w^T x + b}$$

where w and b are the parameters of the model and a^T denotes the transpose of a vector a. If $w^T x + b > 0$, then x belongs to class 1 since its odds is greater than 1. Otherwise, x belongs to class 0.

Since $P(y_1|x) + P(y_2|x) = 1$, it then means that

$$\frac{p(y_2|x)}{1 - P(y_2|x)} = e^z = e^{w^T x + b}$$

We can then simplifying this to consider $P(y_2|x)$ as follows:

$$\frac{p(y_2|x)}{1 - P(y_2|x)} = e^z$$

$$p(y_2|x) = e^z - e^z p(y_2|x)$$

$$p(y_2|x) + e^z p(y_2|x) = e^z$$

$$p(y_2|x)(1 + e^z) = e^z$$

$$p(y_2|x) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}} = \sigma(z)$$

where the function $\sigma(\cdot)$ is known as the logistic or sigmoid function.

Since we have $p(y_2|x)$, we can also derive $p(y_1|x)$ as follows:

$$p(y_1|x) = 1 - \sigma(z)$$

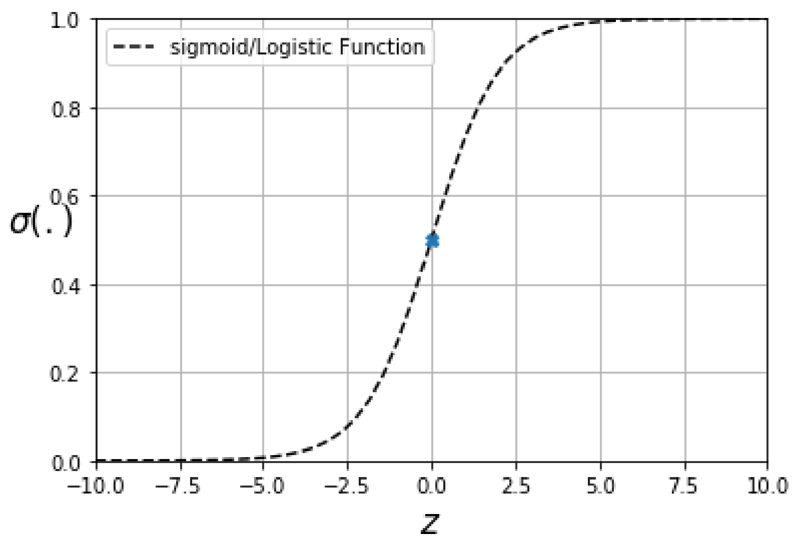
$$= 1 - \frac{1}{1 + e^{-z}}$$

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

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

This basically means if we have learned suitable value of parameters w and b, we are able to estimate the posterior probabilities of any data instance x and determine its class label.

Now, let us see if we can get some intuition from the behavior of the sigmoid function as we vary z. The figure below illustrates the plot of z against the sigmoid function $\sigma(z)$.



We can observe that $\sigma(z) \geq 0.5$ only when $z \geq 0$.

Learning Model Parameters

The parameters of logistic regression,(w, b), are estimated during training using the statistical approach known as the *maximum likelihood estimation* (MLE) method.

This method involves computing the likelihood of observing the training data given (w, b), and then determining the model parameters (w^*, b^*) that yield maximum likelihood.

Let there be dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ with m training instances and y_i is a binary variable (0 or 1).

For a given training instance x_i, we can compute its posterior probabilities as follows:

$$p(y_2|x) = \sigma(z)$$

$$p(y_1|x) = 1 - \sigma(z)$$

We can then express the likelihood of observing y_i given x_i $w, and\ b$ as follows:

$$P(y_i|x_i, w, b) = P(y_2|x_i)^{y_i} \bullet P(y_1|x_i)^{1-y_i}$$

$$= (\sigma(z_i))^{y_i} \bullet (1 - \sigma(z_i))^{1-y_i}$$

$$= (\sigma(w^T x_i + b))^{y_i} \bullet (1 - \sigma(w^T x_i + b))^{1-y_i}$$

where $\sigma(\cdot)$ is the sigmoid function as described above.

The expression above means that the likelihood $P(y_i|x_i, w, b)$ is equal to $P(y_2|x_i)$ when the class is y_2 and equal to $P(y_1|x_i)$ when class label is y_1.

The likelihood of all training instances, $L(w, b)$ can then be computed by taking the product of individual likelihoods (assuming independence among the training instances) as follows:

$$L(w, b) = \prod_{i=1}^{m} P(y_i|x_i, w, b) = \prod_{i=1}^{m} P(y_2|x_i)^{y_i} \bullet P(y_1|x_i)^{1-y_i}$$

This equation involves multiplying a large number of probability values. This naïve computation can become numerically unstable for large datasets.

Therefore, a more practical approach would be to consider the negative logarithm (to the base e) of the likelihood function, also known as the ***cross-entropy function.***

$$-logL(w,b) = -\left[\sum_{i=1}^{m} y_i \log\big(P(y_2|x_i)\big) + (1-y_i)\log(P(y_1|x_i))\right]$$

$$= -\left[\sum_{i=1}^{m} y_i \log\big(P(y_2|x_i)\big) + (1-y_i)\log(1-P(y_2|x_i))\right]$$

$$= -\left[\sum_{i=1}^{m} y_i \log(\sigma(w^T x_i + b)) + (1-y_i)\log(1-\sigma(w^T x_i + b))\right]$$

Concretely, the cross-entropy function is defined as a loss function that measures how unlikely it is for the training data to be generated from the logistic regression model with parameters(w, b).

Intuitively, we would like to find model parameters (w^*, b^*) that result in the lowest cross-entropy, $-log\ L(w^*, b^*)$.

$$(w^*, b^*) = argmin_{\ (w,b)} \mathrm{E}(w, b)$$

$$= argmin_{\ (w,b)} - log\ L(w^*, b^*)$$

where $\mathrm{E}(w, b) = -log\ L(w^*, b^*)$ is the loss function.

Implementation: *Logistic Regression*

So how are do we go about implementing the *Logistic regression algorithm*? We need the following:

i. The *cost function*:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(\hat{y}_i, y_i)$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{n} y_i \log(\,\sigma(w^T x_i + b)) + (1 - y_i)\log(1 - \sigma(w^T x_i + b))\right]$$

where $\hat{y}_i = \sigma(z)$ and $z = w^T x_i + b$

We can vectorized the cost function as follows:

$y_{hat} = \sigma(X\theta)$ as *vector of predictions* $\hat{y}_i$ for $i \in \{0,1,2,3,\ldots,m\}$

$J(\theta) = -\frac{1}{m}\cdot(y^T \log(y_{hat}) + (1 - y_{hat})^T \log(1 - y_{hat})$

ii. *Gradient function* (as derived from the cost function):

$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}[(\hat{y}_i - y_i)]x_i^j$, for $j \in \{0,1,2,3,\ldots,n\}$ and $i \in \{0,1,2,3,\ldots,m\}$

We can also vectorize the gradient function as follows:

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{m}\cdot X^T\cdot(y_{hat} - \vec{y})$$

iii. Using **gradient descent(***with fixed learning rate* η**)**

Initialize the weights at time step $t = 0$ to $\theta(0)$

for $t = 1,2,3,\ldots$ **do**:

Compute gradient: $\frac{\partial}{\partial\theta_j}J(\theta)$

Update the weights: $\theta_{t+1} = \theta_t - \eta\frac{\partial}{\partial\theta_j}J(\theta)$, where parameter η (learning rate)

return θ

----

iv. Alternatively we can use the advanced optimization techniques (See https://docs.scipy.org/doc/scipy/reference/optimize.html)

Characteristics of Logistic Regression

i. Logistic regression is a discriminative model for classification that directly computes the poster probability without making any assumption about the class-conditional probabilities.
ii. Logistic regression can be extend to multiclass classification, where it is known as multinomial logistic regression despite its expressive power is limited to learning only linear decision boundaries.
iii. Because there are different weights/parameters for every attribute, the learned parameters of logistic regression can be analyzed to understand the relationship between attributes and class labels
iv. Because logistic regression does not involve computing densities and distances in the attribute space, it can work more robustly even in high-dimensional settings than distance-based methods such as KNN.
v. Logistic regression can handle irrelevant attributes by learning weight parameters close to zero (0) for attributes that do not provide any gain in the performance during training.
vi. Logistic regression cannot handle data instances with missing values, since the posterior probabilities are only computes by taking a weighted sum of all the attributes.

References

Avati, A., 2019. *CS229: Machine Learning.* [Online]
Available at: http://cs229.stanford.edu/
[Accessed 9 September 2019].

James, G., Witten, D., Hastie, T. & Tibshirani, R., 2017. *An Introduction to Statistical Learning.* New York: Springer.

Marsland, S., 2015. *Machine Learning: An Algorithmic Perspective.* s.l.:CRC press.

Tan, P., Steinbach, M., Karpatne, A. & Kumar, V., 2019. *Introduction to Data Mining.* 2nd ed. New York: Pearson Education.

Zaki, M. J. & Wagner, M., 2014. *Data Mining and Analysis: Fundamental Concepts and Algorithms.* New York: Cambridge University Press.