Linear in The Parameters Classification

Intelligent Systems and Control

2019

Sepehr Maleki

University of Lincoln
School of Engineering

Binary Classification

We are given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, c_i)\}_{i=1}^{N}$ where $c_i \in \{0, 1\}$ are the target classes.

We can assign the probability that a novel input $\mathbf{x}_{new}$ belongs to class 1 using:

$$p(c = 1|\mathbf{x}_{new}) = f(\mathbf{w}^\top \mathbf{x}_{new}) \ ,$$

where:

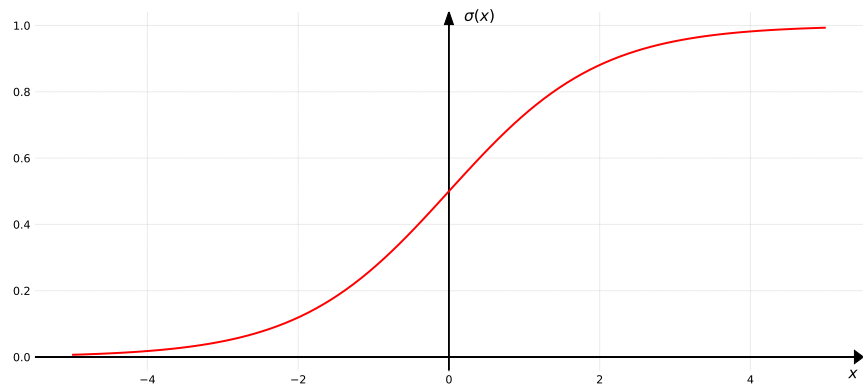
$$0 \leq f(x) \leq 1 \ .$$

(S. Maleki 2019)

Logistic Sigmoid Function

A popular choice for the function $f(x)$ is the logit (sigmoid) function which is given by:

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



Derivative of a sigmoid function with respect to its inputs is given by:

$$\frac{d}{dz}\sigma(z) = \sigma(z)(1-\sigma(z)) \ .$$

(S. Maleki 2019)

Logistic Regression

Logistic regression corresponds to the model:

$$p(c = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x}) \ ,$$

where $\mathbf{w}$ is the weight vector. The decision boundary is defined as the set of $\mathbf{x}$ for which:

$$p(c = 1|\mathbf{x}) = p(c = 0|\mathbf{x}) = \frac{1}{2} \ .$$

Alternatively, the decision boundary is given by the hyperplane:

$$\mathbf{w}^\top \mathbf{x} = 0 \ .$$

On the side of the hyperplane for which $\mathbf{w}^\top \mathbf{x} > 0$, inputs $\mathbf{x}$ are classified as 1's and on the other side as 0's.

(S. Maleki 2019)

Logistic Regression – Representation

Logistic regression can be written in the following equivalent form:

$$p(c|\mathbf{x};\mathbf{w}) = \Big(f(\mathbf{x})\Big)^{c}\Big(1-f(\mathbf{x})\Big)^{1-c},$$

where $f(\mathbf{x}) = \sigma(\mathbf{w}^\top\mathbf{x})$. This means that we can (and should) interpret each label as a Bernoulli random variable: $C \sim Ber(p)$ where $p = \sigma(\mathbf{w}^\top\mathbf{x})$.

In logistic regression, $\mathbf{w}$ is a vector of parameters of length d and we are going to learn the values of those parameters based off of n training examples.

(S. Maleki 2019)

Logistic Regression – How To Fit $\mathbf{w}$?

In order to chose values for the parameters of logistic regression, we use maximum likelihood estimation. We will:

1. Write the log-likelihood function ;
2. Find the values of $\mathbf{w}$ that maximise the log-likelihood function.

The labels that we are predicting are binary, $c \in \{0, 1\}$, and the output of our logistic regression function is supposed to be the probability that $c = 1$.

(S. Maleki 2019)

Logistic Regression – The Likelihood Function

We assume n iid training examples. Then we can write the likelihood of training labels as:

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^{n} p(C = c_i | X = \mathbf{x}_i)$$

Substituting the likelihood of a Bernoulli:

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^{n} \sigma\Big(\mathbf{w}^\top \mathbf{x}_i\Big)^{c_i} \Big(1 - \sigma(\mathbf{w}^\top \mathbf{x}_i)\Big)^{1-c_i} .$$

(S. Maleki 2019)

Logistic Regression – Log-Likelihood

Taking the $\log$ of the likelihood function, gives the log-likelihood for logistic regression:

$$L\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n} c_i \log \sigma\left(\mathbf{w}^\top \mathbf{x}_i\right) + (1 - c_i) \log\left(1 - \sigma(\mathbf{w}^\top \mathbf{x}_i)\right) .$$

Now, the only remaining step is to choose parameters ($\mathbf{w}$) that maximise log likelihood.

(S. Maleki 2019)

Gradient of The Log-Likelihood Function

Setting the derivative equal to zero, won't work: there's no closed form for the maximum.

However, we can find the best values of $\mathbf{w}$ by using an optimisation algorithm. The optimisation algorithm we will use requires the partial derivative of log likelihood with respect to each parameter.

$$\frac{\partial L\mathcal{L}(\mathbf{w})}{\partial w_j} = \sum_{i=1}^{n} \Big(c_i - \sigma(\mathbf{w}^\top \mathbf{x}_i) \Big) \mathbf{x}_i^{(j)} \ ,$$

where $\mathbf{x}_i^{(j)}$ is the j-th component of i-th training input.

(S. Maleki 2019)

Gradient Ascent Optimisation

To find the optimal values of $\mathbf{w}$, we employ an algorithm called gradient ascent.

The idea behind gradient ascent is that gradients point "uphill".

If you continuously take small steps in the direction of your gradient, you will eventually make it to a local maximum.

In the case of logistic regression you can prove that the result will always be a global maximum.

(S. Maleki 2019)

The Gradient Ascent Algorithm

The update to our parameters that results in each small step can be calculated as:

$$
\begin{aligned}
w_j &:= w_j + \alpha \frac{\partial L\mathcal{L}(\mathbf{w})}{\partial w_j} \\
&:= w_j + \alpha \sum_{i=1}^{n} \Big(c_i - \sigma(\mathbf{w}^\top \mathbf{x}_i) \Big) \mathbf{x}_i^{(j)} \ ,
\end{aligned}
$$

where α is the learning rate (magnitude of the step size that we take).

If we continue updating the parameters as above, we will converge on the best values of $\mathbf{w}$.

(S. Maleki 2019)

Gradient of The Log-Likelihood – Hard Way!

The derivative of gradient for one data point $(\mathbf{x}, c)$ is:

$$
\begin{aligned}
\frac{\partial L\mathcal{L}(\mathbf{w})}{\partial w_j} &= \frac{\partial}{\partial w_j} c \log \sigma(\mathbf{w}^\top \mathbf{x}) + \frac{\partial}{\sigma w_j}(1-c)\log(1-\sigma(\mathbf{w}^\top \mathbf{x})) \\
&= \left(\frac{c}{\sigma(\mathbf{w}^\top \mathbf{x})} - \frac{1-c}{1-\sigma(\mathbf{w}^\top \mathbf{x})} \right) \frac{\partial}{\partial w_j} \sigma(\mathbf{w}^\top \mathbf{x}) \\
&= \left(\frac{c}{\sigma(\mathbf{w}^\top \mathbf{x})} - \frac{1-c}{1-\sigma(\mathbf{w}^\top \mathbf{x})} \right) \sigma(\mathbf{w}^\top \mathbf{x})(1-\sigma(\mathbf{w}^\top \mathbf{x})) \mathbf{x}^{(j)} \\
&= \left(\frac{c-\sigma(\mathbf{w}^\top \mathbf{x})}{\sigma(\mathbf{w}^\top \mathbf{x})(1-\sigma(\mathbf{w}^\top \mathbf{x}))} \right) \sigma(\mathbf{w}^\top \mathbf{x})(1-\sigma(\mathbf{w}^\top \mathbf{x})) \mathbf{x}^{(j)} \\
&= \Big(c - \sigma(\mathbf{w}^\top \mathbf{x}) \Big) \mathbf{x}^{(j)} \ .
\end{aligned}
$$

(S. Maleki 2019)

Gradient of The Log-Likelihood – Easy Way!

Let $p = \sigma(\mathbf{w}^\top \mathbf{x})$ and $z = \mathbf{w}^\top \mathbf{x}$. Then:

$$
\begin{aligned}
\frac{\partial L\mathcal{L}(\mathbf{w})}{\partial w_j} &= \frac{\partial L\mathcal{L}(\mathbf{w})}{\partial p} \cdot \frac{\partial p}{\partial z} \cdot \frac{\partial z}{\partial w_j} \\
&= \left(\frac{c}{p} - \frac{1-c}{1-p} \right) . \sigma(z)(1 - \sigma(z)) . \mathbf{x}^{(j)} \\
&= \left(\frac{c}{p} - \frac{1-c}{1-p} \right) . p(1-p) . \mathbf{x}^{(j)} \\
&= \Big(c(1-p) - p(1-c) \Big) . \mathbf{x}^{(j)} \\
&= (c-p)\mathbf{x}^{(j)} \\
&= \Big(c - \sigma(\mathbf{w}^\top \mathbf{x}) \Big) \mathbf{x}^{(j)} .
\end{aligned}
$$

(S. Maleki 2019)

Logistic Regression – Alternative Approach to $\mathbf{w}$ Fitting

Define the following cost function:

$$J(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \text{Cost}\Big(f(\mathbf{x}_i),\ c_i \Big) \ ,$$

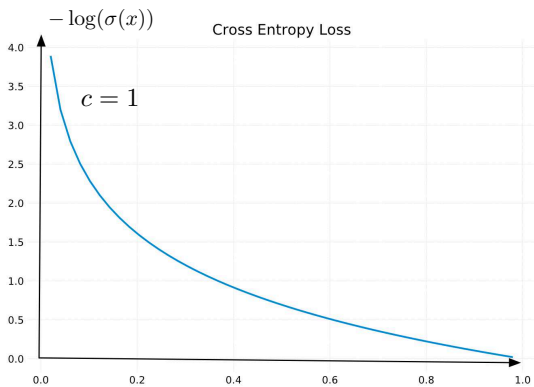
where

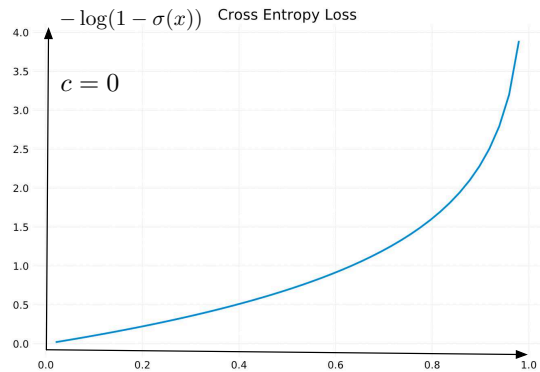
$$\text{Cost}\Big(f(\mathbf{x}),\ c \Big) = \begin{cases} -\log\Big(f(\mathbf{x}) \Big) & \text{if } c = 1 \\ -\log\Big(1 - f(\mathbf{x}) \Big) & \text{if } c = 0 \end{cases} .$$

(S. Maleki 2019)

Logistic Regression – Alternative $\mathbf{w}$ Fitting

$$\text{Cost}\Big(f(\mathbf{x}),\ c\Big) = \begin{cases} -\log\Big(f(\mathbf{x})\Big) & \text{if } c = 1 \\ -\log\Big(1 - f(\mathbf{x})\Big) & \text{if } c = 0 \end{cases}.$$





$$\text{Cost}\Big(f(\mathbf{x}),\ c\Big) = -c\log\Big(f(\mathbf{x})\Big) - (1-c)\log\Big(1 - f(\mathbf{x})\Big)$$

(S. Maleki 2019)

Logistic Regression – Alternative Approach to $\mathbf{w}$ Fitting

Logistic regression cost function:

$$
\begin{aligned}
J(\mathbf{w}) &= \frac{1}{n}\sum_{i=1}^{n} \text{Cost}\Big(f(\mathbf{x}_i),\ c_i \Big) \\
&= -\frac{1}{n}\Bigg(\sum_{i=1}^{n} c_i \log f(\mathbf{x}_i) + (1-c_i)\log(1-f(\mathbf{x}_i)) \Bigg)
\end{aligned}
$$

Parameters $\mathbf{w}$ that minimise the cost function $J(\mathbf{w})$ are the best values of $\mathbf{w}$.

(S. Maleki 2019)

Logistic Regression – Alternative Approach to $\mathbf{w}$ Fitting

Given the cost function for n training examples:

$$J(\mathbf{w}) = -\frac{1}{n}\left(\sum_{i=1}^{n} c_i \log f(\mathbf{x}_i) + (1-c_i)\log(1-f(\mathbf{x}_i))\right),$$

to $\min_{\mathbf{w}} J(\mathbf{w})$, we use the gradient descent algorithm:

Repeat until convergence $\Big\{$

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w}) := w_j - \alpha \sum_{i=1}^{n} \Big(f(\mathbf{x}_i) - y_i\Big)\mathbf{x}_i^{(j)}$$

$\Big\}$

(S. Maleki 2019)

The Perceptron Learning Algorithm

The perceptron deterministically assigns $\mathbf{x}$ to class 1 if $\mathbf{w}^\top \mathbf{x} \geq 0$, and to class 0 otherwise. That is:

$$p(c = 1|\mathbf{x}) = g(\mathbf{w}^\top \mathbf{x}) \ ,$$

where the threshold (step) function, $g(z)$, is defined as:

$$g(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases} \ .$$

The following update rule gives the perceptron learning algorithm:

$$w_j := w_j + \alpha \Big(y_i - f(\mathbf{x}_i) \Big) \mathbf{x}_i^{(j)} \ .$$

(S. Maleki 2019)

Multi-Class Classification

For more than two classes, one may use the softmax function:

$$p(c = i|\mathbf{x}) = \frac{e^{\mathbf{w}_i^\top \mathbf{x}}}{\sum_{j=1}^{C} e^{\mathbf{w}_j^\top \mathbf{x}}} \ ,$$

where C is the number of classes.

When $C = 2$ this can be reduced to the logistic sigmoid model.

(S. Maleki 2019)