Classification

Dr. Víctor Uc Cetina

Universität Hamburg

Content

Binary Classification

Suppose we want to build an email spam classification software:

> **You have been invited...**
>
> Opportunity is an online business network that takes the work out of networking by matching you with people who are actively in need of your services, products, skills and more. Each day the Opportunity matching algorithm scans the network to find people who can bring you business and/or help advance your career.
>
> Sign In / Sign Up

> Hello
> I want to know about different topics that relate to qualitative reinforcement learning and make abstraction&aggregation... to solve problem compactly . I have read some survey to know exactly but sometimes I doubt about some topics are related to or not. for example Qualitative Spatial Representation and Reasoning. can anyone tell me different categorized topics? I need to know the general classification of them.
>
> ...

Binary Classification

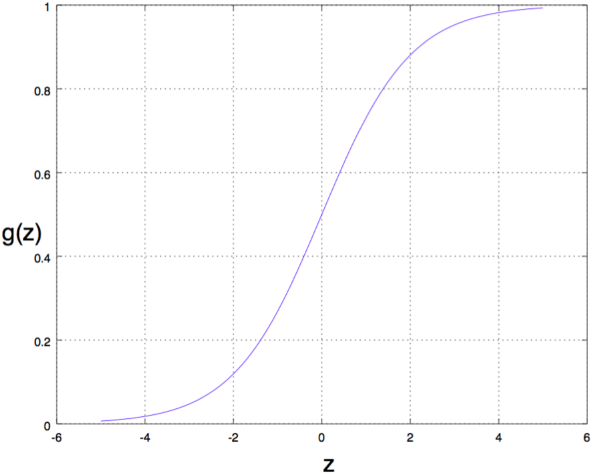
- Classification is the same as regression, the only difference is that the output variable y can take only a small number of discrete values.
- The most simple classification problem is the binary classification, in which $y = \{0, 1\}$.
- In a binary classification problem we have positive examples $y = 1$ (spam) and negative examples $y = 0$ (no spam).
- The $\mathbf{x}$ may be some features of some piece of email.

Binary Classification

- We could approach the classification problem ignoring the fact that y is a discrete value, and use linear regression to try to predict y given $\mathbf{x}$.
- However, it is easy to construct examples showing this method to perform very badly.
- Intuitively, it does not make sense for $h_\theta = (\mathbf{x})$ to take values larger than 1 or smaller than 0, when we know that $y \in \{0, 1\}$
- To fix this we will make $h_\theta(\mathbf{x}) = g(\theta^\top \mathbf{x}) = \frac{1}{1+e^{-\theta^\top \mathbf{x}}}$
- where $g(z) = \frac{1}{1+e^{-z}}$

Binary Classification

$$g(z) = \frac{1}{1+e^{-z}} \qquad g : \mathbb{R} \to (0,1)$$



Derivative of the Sigmoid Function

$$
\begin{aligned}
g'(z) &= \tfrac{d}{dz}\tfrac{1}{1+e^{-z}} \\
&= -\tfrac{1}{\left(1+e^{-z}\right)^2} \cdot \left(- e^{-z} \right) \\
&= \tfrac{1}{\left(1+e^{-z}\right)^2} \cdot e^{-z} \\
&= \tfrac{1}{\left(1+e^{-z}\right)} \cdot \tfrac{e^{-z}}{\left(1+e^{-z}\right)} \\
&= \tfrac{1}{\left(1+e^{-z}\right)} \cdot \tfrac{e^{-z}+1-1}{\left(1+e^{-z}\right)} \\
&= \tfrac{1}{\left(1+e^{-z}\right)} \cdot \left(\tfrac{\left(1+e^{-z}\right)}{\left(1+e^{-z}\right)} - \tfrac{1}{\left(1+e^{-z}\right)} \right) \\
&= g(z)\big(1 - g(z)\big).
\end{aligned}
$$

A Probabilistic Approach

Lets assume that:

$$P(y = 1|x; \theta) = h_\theta(x)$$

$$P(y = 0|x; \theta) = 1 - h_\theta(x)$$

Note that this can be written more compactly as:

$$p(y|x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Likelihood of the Training Data's Labels

Assuming that the m training examples were generated independently, we can then write:

$$
\begin{aligned}
L(\theta) &= p(\mathbf{y}|\mathbf{X};\theta) \\
&= \prod_{i=1}^{m} p(y_i|\mathbf{x}_i;\theta) \\
&= \prod_{i=1}^{m} (h_\theta(x_i))^{y_i}(1 - h_\theta(x_i))^{1-y_i}
\end{aligned}
$$

Log Likelihood

$$L(\theta) = \prod_{i=1}^{m} (h_\theta(x_i))^{y_i} (1 - h_\theta(x_i))^{1-y_i}$$

Working instead with the log likelihood:

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y_i \log h(x_i) + (1 - y_i) \log(1 - h(x_i)) \end{aligned}$$

Using gradient ascent we get an update rule like this:

$$\theta := \theta + \alpha \nabla_\theta \ell(\theta)$$

Maximizing the Likelihood

Working with one example, the derivatives are as follows:

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\theta) &= \frac{\partial}{\partial \theta_j} \Big[y \log h(x) + (1-y) \log(1 - h(x)) \Big] \\
&= \frac{\partial}{\partial \theta_j} \Big[y \log g(\theta^\top \mathbf{x}) + (1-y) \log(1 - g(\theta^\top \mathbf{x})) \Big] \\
&= \left(y \frac{1}{g(\theta^\top \mathbf{x})} - (1-y) \frac{1}{1 - g(\theta^\top \mathbf{x})} \right) \frac{\partial}{\partial \theta_j} g(\theta^\top \mathbf{x}) \\
&= \left(y \frac{1}{g(\theta^\top \mathbf{x})} - (1-y) \frac{1}{1 - g(\theta^\top \mathbf{x})} \right) g(\theta^\top \mathbf{x}) \Big(1 - g(\theta^\top \mathbf{x}) \Big) \frac{\partial}{\partial \theta_j} \theta^\top \mathbf{x} \\
&= \Big(y(1 - g(\theta^\top \mathbf{x})) - (1-y) g(\theta^\top \mathbf{x}) \Big) x_j \\
&= (y - h_\theta(\mathbf{x})) x_j
\end{aligned}
$$

The LMS Update Rule for Classification

Given that:

$$\frac{\partial}{\partial\theta_j}\ell(\theta) = (y - h_\theta(\mathbf{x}))x_j,$$

our LMS update rule for classification can be written as:

$$\theta_j := \theta_j + \alpha(y_i - h_\theta(\mathbf{x}_i))(x_i)_j,$$

where $h_\theta(\mathbf{x}_i) = g(\theta^\top \mathbf{x}_i) = \frac{1}{1+e^{(-\theta^\top \mathbf{x}_i)}}$ is now defined as a non-linear function of $\theta^\top \mathbf{x}_i$.

So, we end up with the same update rule for a different algorithm and learning problem.

LMS Algorithms for Classification

Batch Gradient Descent

Repeat until convergence $\{$

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left[y_i - g(\theta^\top x_i) \right] (x_i)_j \qquad \text{(for every } j\text{).}$$

$\}$

Stochastic Gradient Descent

Loop $\{$

for $i = 1$ to m $\{$

$$\theta_j := \theta_j + \alpha \left[y_i - g(\theta^\top x_i) \right] (x_i)_j \qquad \text{(for every } j\text{).}$$

$\}$

$\}$

LMS Algorithms for Classification

Mini-Batch Gradient Descent

Repeat until convergence $\{$

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{k} \left[y_i - h_\theta(x_i) \right] (x_i)_j \qquad \text{(for every } j\text{)}.$$

$\}$

Here we use mini-batches containing 10 to 1000 examples. This is $k \in [10, 1000]$.

Error of a Binary Classifier (1/2)

True Positive $=$ A positive example correctly identified as positive.

False Positive $=$ A negative example incorrectly identified as positive.

True Negative $=$ A negative example correctly identified as negative.

False Negative $=$ A positive example incorrectly identified as negative.

Error of a Binary Classifier (2/2)

Precision

It answers the question: How many of the examples identified as positives are indeed positives?

$$\text{Precision} = \frac{TP}{TP+FP}.$$

Recall (Sensitivity)

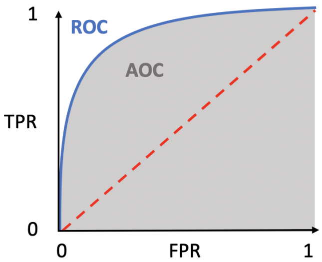
It answers the question: How many of the positive examples were correctly identified as positive?

$$\text{Recall} = \frac{TP}{P} = \frac{TP}{TP+FN}.$$

Receiver Operating Characteristic (ROC) curve

The Receiver Operating Characteristic curve is a graph showing the performance of a binary classifier with all the classification thresholds.

It shows two parameters: the true positive rate (TPR) and the false positive rate (FPR).



$$TPR = Recall = Sensitivity = \frac{TP}{TP+FN}$$

$$Specificity = \frac{TN}{FP+TN}$$

$$FPR = \frac{FP}{FP+TN} = 1 \text{ - specificity}$$

AOC = Area under the ROC curve.

The Perceptron Learning Algorithm

Consider modifying the logistic regression to output either 1 or 0:

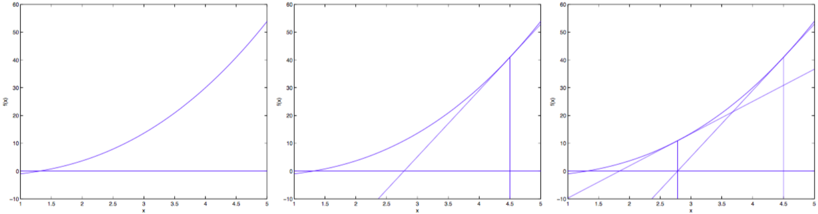
$$g(\mathbf{z}) = \begin{cases} 1 & \text{if } \mathbf{z} \geq 0 \\ 0 & \text{if } \mathbf{z} < 0 \end{cases}$$

By making $h_\theta(x) = g(\theta^\top \mathbf{x})$, then we have the update rule:

$$\theta_j := \theta_j + \alpha(y_i - h_\theta(x_i))(x_i)_j.$$

Newton's Method for Finding a Zero of a function

Suppose we have a function $f : \mathbb{R} \rightarrow \mathbb{R}$ and we want to find a value of θ such that $f(\theta) = 0$, with $\theta \in \mathbb{R}$.



Newton's method performs the following update rule:

$$\theta := \theta - \frac{f(\theta)}{f'(\theta)}.$$

Newton's Method for Finding a Zero of a function

Now, suppose we want to maximize a function ℓ. The maxima of ℓ correspond to points where its first derivative $\ell'(\theta)$ is zero.

So, by letting $f(\theta) = \ell'(\theta)$, we can use the same algorithm to maximize ℓ:

$$\theta := \theta - \frac{\ell'(\theta)}{\ell''(\theta)}.$$

Newton-Raphson Method

In our regression setting θ is vector-valued. The generalization of Newton's method to this multidimensional setting is given by

$$\theta := \theta - H^{-1} \nabla_\theta \ell(\theta),$$

where H is the Hessian matrix

$$H_{ij} = \frac{\partial^2 \ell(\theta)}{\partial \theta_i \partial \theta_j}.$$

Thank you!

Dr. Víctor Uc Cetina
cetina@informatik.uni-hamburg.de