Applied Machine Learning: Tutorial Number 2 Answers

School of Informatics, University of Edinburgh
Instructors: Oisin Mac Aodha and Siddharth N.

September 2022

1. Consider using logistic regression for a two-class classification problem in two dimensions:

$$p(y = 1|\boldsymbol{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2)$$

Here σ denotes the logistic (or sigmoid) function $\sigma(z) = 1/(1 + \exp(-z))$, y is the target which takes on values of 0 or 1, $\boldsymbol{x} = (x_1, x_2)$ is a vector in the two-dimensional input space, and $\boldsymbol{w} = (w_0,\ w_1, w_2)$ are the parameters of the logistic regressor.

(a) Consider a weight vector $\boldsymbol{w}_A = (-1, 1, 0)$. Sketch the decision boundary in $\boldsymbol{x}$ space corresponding to this weight vector, and mark which regions are classified with labels 0 and 1.
Solution: The decision boundary is given by $-1 + x_1 = 0$ or $x_1 = 1$. So in $\boldsymbol{x}$ space this is a vertical line through $x_1 = 1$, with the halfspace to the right being class 1.

(b) Consider a second weight vector $\boldsymbol{w}_B = (5, -5, 0)$. Again sketch the decision boundary in $\boldsymbol{x}$ space corresponding to this weight vector, and mark which regions are classified with labels 0 and 1.
Solution: The decision boundary is given by $5 - 5x_1 = 0$ or $x_1 = 1$. So in $\boldsymbol{x}$ space this is a vertical line through $x_1 = 1$, with the halfspace to the left being class 1.

(c) Plot $p(y = 1|\boldsymbol{x})$ as a function of x_1 for both $\boldsymbol{w}_A$ and $\boldsymbol{w}_B$, and comment on any differences between the two.
Solution: For $\boldsymbol{w}_A$ we have a logistic function which goes to 0 as $x_1 \to -\infty$, and to 1 as $x_1 \to \infty$. It has value 0.5 at $x_1 = 1$. For $\boldsymbol{w}_B$ we have a logistic function which goes to 1 as $x_1 \to -\infty$, and to 0 as $x_1 \to \infty$. It has value 0.5 at $x_1 = 1$. Most importantly the logistic function for $\boldsymbol{w}_B$ is steeper than that for $\boldsymbol{w}_A$ around $x_1 = 1$.

2. Consider the logistic regression setup in the previous questions, but with a new weight vector $\boldsymbol{w} = (0, -1, 1)$. Consider the following data set:

Instance	x_1	x_2	Class
1	0.5	-0.35	−
2	-0.1	0.1	−
3	-1.2	1.0	+

(a) Compute the gradient of the *negative log likelihood* of the logistic regression model for this data set.

(b) Suppose that we take a single step of gradient descent with step size $\eta = 3.0$. What are the updated values for the model weights?

(c) Do the new weights do a better job of classifying the three training instances above?

It will help you to remember the following facts:

- The negative log-likelihood in logistic regression is

$$\begin{aligned}\text{NLL}(\boldsymbol{w}) &= -\frac{1}{N}\sum_{i=1}^{N}\log p(y = y_i|\boldsymbol{x}_i; \boldsymbol{w}) \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log p(y = 1|\boldsymbol{x}_i; \boldsymbol{w}) + (1 - y_i)\log p(y = 0|\boldsymbol{x}_i; \boldsymbol{w})\right]\end{aligned}$$

- The partial derivative of the negative log-likelihood with respect to a parameter w_d is

$$\frac{\partial \text{NLL}}{\partial w_d} = \frac{1}{N}\sum_{i=1}^{N}(\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i) - y_i)x_{id}$$

- To minimize a function $\text{NLL}(\boldsymbol{w})$, we use the gradient *descent* rule, which is

$$\boldsymbol{w}' \leftarrow \boldsymbol{w} - \eta \nabla \text{NLL}$$

Solution: For each training instance, first compute $p(y = 1|\boldsymbol{x}_i) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)$ and $\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i) - y_i$ (where $y_i \in \{0, 1\}$):

Instance	x_1	x_2	$\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)$	$\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i) - y_i$
1	0.5	-0.35	0.30	0.30
2	-0.1	0.1	0.55	0.55
3	-1.2	1.0	0.90	-0.1

(a) Then the partial derivatives are

$$\begin{aligned}\frac{\partial \text{NLL}}{\partial w_0} &= \frac{1}{3}[(0.3 \cdot 1) + (0.55 \cdot 1) + (-0.1 \cdot 1)] = \frac{1}{3} \cdot 0.75 \\ \frac{\partial \text{NLL}}{\partial w_1} &= \frac{1}{3}[(0.3 \cdot 0.5) + (0.55 \cdot -0.1) + (-0.1 \cdot -1.2)] = \frac{1}{3} \cdot 0.215 \\ \frac{\partial \text{NLL}}{\partial w_2} &= \frac{1}{3}[(0.3 \cdot -0.35) + (0.55 \cdot 0.1) + (-0.1 \cdot 1.0)] = \frac{1}{3} \cdot -0.15\end{aligned}$$

Note that w_0 is special, because it corresponds to a feature that is always 1. Without it, the decision boundary would always pass through the origin.

(b) The initial weight vector was $\boldsymbol{w} = (0, -1, 1)$. After we take on step of gradient descent ($\boldsymbol{w}' \leftarrow \boldsymbol{w} - \eta \nabla \text{NLL}$) with our step size $\eta = 3.0$, the new weight vector is $\boldsymbol{w}' = (-0.75, -1.215, 1.15)$. In practice, we would not necessarily choose a step size like $\eta = 3.0$, as it is very large, but it makes for a better example. If we reproduce the above table at the new weight setting, we get

Instance	x_1	x_2	$p(y = 1 \mid \boldsymbol{x}_i)$
1	0.5	-0.35	0.15
2	-0.1	0.1	0.37
3	-1.2	1.0	0.87

(c) The two negative instances have seen their classification improve (0.3 to 0.15 and 0.55 to 0.37), while the positive instance has gotten slightly worse (0.9 to 0.87). This will be set right with further gradient steps.

3. You have a collection of 1000 nature photographs which were taken under many different conditions. All of the images are of size 300×300 pixels. You wish to develop a binary classifier that labels a photograph as to whether or not it depicts a sunny day on a beach. The images have been pre-processed in the following manner:

- Each image $i \in \{1 \ldots 1000\}$ is partitioned nine regions $R_{i,1} \ldots R_{i,9}$. Each region is 100×100 pixels. The regions are arranged in a 3×3 grid, so that the region R_{i1} is the top-left corner of image i, the region R_{i2} is the top middle portion of the image, and so on.
- For each region $R_{i,j}$, we compute the average *hue*[1] of pixels within the region $R_{i,j}$. The hue value is quantised into 7 discrete bins: "red", "orange", "yellow", "green", "blue", "indigo" and "violet".

(a) What features would you use to describe the data given the description above?

(b) How many features are there? Are they categorical, ordinal or numeric?

(c) What values can they take on?

Solution: The naive (and incorrect) solution is to use 9 categorical features $X_1 \ldots X_9$, where the possible values are the colour labels. This would work if there was a natural "structure" to the regions, e.g. if region R_1 represented the same thing in all images (e.g. the "sun" region). In practice, there is no structure or ordering to the regions: in one image the top-left region R_1 might contain the "sun" while in another R_1 could contain clear blue sky.

The correct answer is:

(a) Features will reflect presence or absence of particular colours in the image.

(b) There are 7 features (one per colour value), their values are numeric.

(c) The values are either binary (presence / absence) or integer, if we want to allow repetitions of colours: e.g. an image containing two "yellow" regions may be deemed different from an image containing one "yellow" region.

[1]The *hue* is a scalar representation of color. It ranges from 0° to 360°. For example, colors with hues around 0° look red, hues around 120° look blue, and hues around 240° look green.