Theoretical Background on Classification

Contents

Binary classification

Imagine that you have a bunch of observations consisting of inputs/features $\mathbf{x}_{1:n} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and the corresponding targets $y_{1:n} = (y_1, \ldots, y_n)$. Remember that we say that we have a classification problem when the targets are discrete labels. In particular, if the labels are two, say 0 or 1, then we say that we have a *binary classification problem.

Logistic regression

Imagine that you have a bunch of observations consisting of inputs/features $\mathbf{x}_{1:N} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and the corresponding targets $y_{1:N} = (y_1, \ldots, y_N)$. Remember that we say that we have a classification problem when the targets are discrete labels. In particular, if the labels are two, say 0 or 1, then we say that we have a *binary classification problem*.

The logistic regression model is one of the simplest ways to solve the binary classification problem. It goes as follows. You model the probability that $y = 1$ conditioned on having $\mathbf{x}$ by:

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \text{sigm}\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right) = \text{sigm}\left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\right),$$

where sigm is the sigmoid function, the $\phi_j(\mathbf{x})$ are m basis functions/features,

$$\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_M(\mathbf{x}))$$

and the w_j's are m weights that we need to learn from the data. The sigmoid function is defined by:

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

and all it does is it takes a real number and maps it to $[0, 1]$ so that it can represent a probability. In other words, logistic regression is just a generalized linear model passed through the sigmoid function so that it is mapped to $[0, 1]$.

If you need the probability of $y = 0$, it is given by the obvious rule:

$$p(y = 0|\mathbf{x}, \mathbf{w}) = 1 - p(y = 1|\mathbf{x}, \mathbf{w}) = 1 - \text{sigm}\left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\right)$$

You can represent the probability of an arbitrary label y conditioned on $\mathbf{x}$ using this simple trick:

$$p(y|\mathbf{x}, \mathbf{w}) = \left[\text{sigm}\left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\right)\right]^y \left[1 - \text{sigm}\left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})\right)\right]^{1-y}.$$

Notice that when $y = 1$, the exponent of the second term becomes zero and thus the term becomes one. Similarly, when $y = 0$, the exponent of the first tierm becomes zero and thus the term becomes one. This gives the right probability for each case.

The likelihood of all the observed data is:

$$p(y_{1:N}|\mathbf{x}_{1:n}, \mathbf{w}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{N} \left[\text{sigm}\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right)\right]^{y_i}\left[1 - \text{sigm}\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right)\right]^{1-y_i}.$$

We can now find the best weight vector $\mathbf{w}$ using the maximum likelihood principle. We need to solve the optimization problem:

$$\max_{\mathbf{w}} \log p(y_{1:N}|\mathbf{x}_{1:n}, \mathbf{w}) = \max_{\mathbf{w}} \sum_{i=1}^{N} \left\{y_i \,\text{sigm}\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right) + (1-y_i)\left[1 - \text{sigm}\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right)\right]\right\}.$$

Notice that the following maximization problem is equivalent to minimizing this loss function:

$$L(\mathbf{w}) = -\sum_{i=1}^{N} \left\{y_i \,\text{sigm}\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right) + (1-y_i)\left[1 - \text{sigm}\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)\right)\right]\right\}.$$

This is known as the cross-entropy loss function and you are very likely to encounter it if you dive deeper into modern data science. For example, we use the same loss function to train state-of-the-art deep neural networks that classify images. You now know that it does not come out of the blue. It comes from the maximum likelihood principle.

https://en.wikipedia.org/wiki/Maximum_likelihood_estimation

Examples

See this and this.

Making-decisions

Say that you have found a point estimate for the weights $\mathbf{w}$ based on the data. Let's call that point estimate $\mathbf{w}^*$. You can predict the probability of y taking any value by evaluating $p(y|\mathbf{x}, \mathbf{w} = \mathbf{w}^*)$. That's what your model predicts: a probability mass function over the two possible values of y. Now, let's say that you have to decide whether y is equal to 0 or 1. How do you do this? You need to pose and solve a decision making problem. Like we did before, you start by quantifying the loss you incur when you make the wrong decision. Mathematically, let $\hat{y}$ be what you choose and y be the true value of the label. You need to define the function:

$$\ell(\hat{y}, y) = \text{the cot of picking } \hat{y} \text{ when the true value is } y.$$

Of course, the choice of the cost function is subjective. For the binary classification case we are looking at, $\ell(\hat{y}, y)$ is just a 2×2 matrix (two possibilities for $\hat{y}$ and two possibilities for y). Once you have your loss function, the rational thing to do is to pick the $\hat{y}$ that minimizex your *expected loss*. The expectation here is over what you think the true value of y is. This state of knowledge is summarized in the trained logistic regression model. Therefore, the problem you need to solve to pick $\hat{y}$ is:

$$\min_{\hat{y}} \sum_{y=0,1} \ell(\hat{y}, y) p(y|\mathbf{x}, \mathbf{w} = \mathbf{w}^*).$$

where the summation here is attributed to the definition of discrete expectation

Notice that this is a different optimization problem for each possible value of $\mathbf{x}$.

Examples

See this.

Diagnostics for classification

As always you need to split your dataset in training and validation subsets. There are many different diagnostics you can use on your validation dataset. The two that we are going to cover in the video lecture are:

- Accuracy score
- Confusion matrix

See this.

Multi-class classification

Imagine that you have a bunch of observations consisting of inputs/features $\mathbf{x}_{1:n} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and the corresponding discrete labels $y_{1:n} = (y_1, \ldots, y_n)$. But now there are not just two labels. There are K possible values for the labels. That's *multi-class classification*.

Multi-class logistic regression

Just like before assume that you have a set of m basis function $\phi_j(\mathbf{x})$ which we will use to make generalized linear models. The multi-class logistic regession model is defined by:

$$p(y = k|\mathbf{x}, \mathbf{W}) = \text{softmax}_k \left(\mathbf{w}_1^T\boldsymbol{\phi}(\mathbf{x}), \ldots, \mathbf{w}_K^T\boldsymbol{\phi}(\mathbf{x})\right),$$

where

$$\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_K)$$

is a collection of K weight vectors with m elements (one m-dimensional weight vector for each of the K classes), and $\text{softmax}_k(z_1, \ldots, z_K)$ is the k-th component of a vector function of K real inputs defined by:

$$\text{softmax}_k(z_1, \ldots, z_K) = \frac{z_k}{\sum_{k'=1}^K z_{k'}}.$$

The role of the softmax is to take the real outputs of the generalized linear models corresponding to each label and map them to a probability.

Just like in the logistic regression case, you can train the model by putting a prior on the weights and then maximizing the logarithm of the posterior.

Examples

See this.

By Ilias Bilionis (ibilion[at]purdue.edu)