1 Theory

In project 1 we used linear regression to predict a continuous output from a set of inputs (reference project1). We used ordinary least squares regression (OLS), Ridge regression and Lasso regression, where the two latter impose a penalty to the OLS. In this project we will reuse the ideas and code of project 1, but we will also use neural networks to predict continuous variables. In addition we study situations where the outcome is discrete rather than continuous. This is a classification problem, and we will use logistic regression to model the probabilities of the classes.

1.1 Logistic regression

Just like a linear regression model, a logistic regression model computes a weighted sum of the predictor variables, written in matrix notation as $\boldsymbol{X}^T\beta$. However, the logistic regression returns the logistic of the this weighted sum as the probabilities. For a classification problem with K classes, the model has the following form (Hastie et al. p.119):

$$\begin{aligned} log\frac{Pr(G=1|X=x)}{Pr(G=K|X=x)} &= \beta_{10} \ + \ \beta_1^T x \\ log\frac{Pr(G=2|X=x)}{Pr(G=K|X=x)} &= \beta_{20} \ + \ \beta_2^T x \\ &\vdots \\ log\frac{Pr(G=K-1|X=x)}{Pr(G=K|X=x)} &= \beta_{(K-1)0} \ + \ \beta_{K-1}^T x \end{aligned} \tag{1}$$

It is arbitrary which class is used in the denominator for the log-odds above. Taking the exponential on both sides and solving for $Pr(G=k|X=x)$ gives the following probabilities:

$$\begin{aligned} Pr(G=k|X=x) &= \frac{exp(\beta_{k0} \ + \ \beta_k^T x)}{1+\sum_{l=1}^{K-1} exp(\beta_{l0} \ + \ \beta_l^T x)}, \ k=1,...,K-1, \\ Pr(G=K|X=x) &= \frac{1}{1+\sum_{l=1}^{K-1} exp(\beta_{l0} \ + \ \beta_l^T x)}, \end{aligned} \tag{2}$$

and the probabilities sum to one. The output is then classified as the class with the highest probability.

1.1.1 Fitting logistic regression model

The usual way of fitting logistic regression models is by maximum likelihood. The log-likelihood for N observations is defined as:

$$l(\theta) \ = \ \sum_{i=1}^{N} logp_{g_i}(x_i;\theta), \tag{3}$$

where $p_k(x_i;\theta) \ = \ Pr(G = k|X = x_i;\theta)$ and $\theta \ = \ \{\beta_{10}, \beta_1^T, ..., \beta_{(K-1)0}, \beta_{K-1}^T\}$.

One very common classification problem is a situation with binary outcomes, either it happens or it does not. As we see from Equation 1 above, setting K=2 simplifies the model considerable, since there will now be only a single linear function. θ in Equation 3 will also be simplified: $\theta = \beta = \{\beta_{10}, \beta_1^T\}$. The two-class case is what is used in this project, and the following discussion will assume the outcome has two classes.
We start by coding the two-class g_i with a 0/1 response y_i, where $y_i = 1$ when $g_i = 1$, and $y_i = 0$ when $g_i = 2$. Next, we let $p_1(x;\theta) \ = \ p(x;\beta)$, and $p_2(x;\theta) \ = \ 1 \ - \ p(x;\beta)$. The log-likelihood can then be written

$$\begin{aligned}
l(\beta) &= \sum_{i=1}^{N}\{y_i logp(x_i;\beta) + (1-y_i)log(1-p(x_i;\beta))\} \\
&= \sum_{i=1}^{N}\{y_i log\frac{p(x_i;\beta)}{1-p(x_i;\beta)} + log(1-p(x_i;\beta))\} \\
&= \sum_{i=1}^{N}\{y_i\beta^T x_i + log(1 - \frac{1}{1+exp(-\beta^T x_i)}\} \\
&= \sum_{i=1}^{N}\{y_i\beta^T x_i + log(\frac{exp(1}{1+exp(\beta^T x_i)}\} \\
&= \sum_{i=1}^{N}\{y_i\beta^T x_i - log(1+exp(\beta^T x_i))\}.
\end{aligned} \tag{4}$$

This is the equation we want to maximize to find the best fit. Following the approach in Géron's book (reference), we chose the equivalent approach of minimizing the following

$$J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\{y_i\beta^T x_i - log(1 + exp(\beta^T x_i)).\} \tag{5}$$

This is just the negative of Equation 4, divided by the number of samples. This is our cost function, and dividing by the number of training samples finds the mean cost.

To minimize this cost function we used gradient descent. Gradient descent measures the local gradient of the cost function, with regards to β in our case. Since the gradient goes in the direction of fastest increase, we will go in the opposite direction, i.e. negative gradient. We start by choosing random values for β (since our cost function is convex any choice should give correct results), calculate the gradient, update the β values, and do this iteratively until the algorithm converges to a minimum. The size of the steps is important, and is determined by the learning rate. If the learning rate is too small, we will need many iterations which is time consuming. However, if the learning rate is too high, we might overshoot and miss the minimum. One way to choose the learning rate is too let it depend on the size of the gradient. If the gradient is large, i.e a steep slope, the learning rate can be relatively high. When the gradient is small, the learning rate is also small.

Returning to the logistic regression problem, the derivative of the cost function is

$$\begin{aligned}\frac{\partial J(\beta)}{\partial \beta} &= -\frac{1}{N}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{p}) \\ &= \frac{1}{N}\boldsymbol{X}^T(\boldsymbol{p} - \boldsymbol{y}),\end{aligned} \tag{6}$$

where $\boldsymbol{X}$ is the $N \times (p+1)$ matrix of x_i values, $\boldsymbol{p}$ is the vector of fitted probabilities with ith element $p(x_i;\beta)$ and $\boldsymbol{y}$ is the vector of y_i values. The new β using gradient descent is then
$\beta_{new} = \beta_{old} - \frac{\partial J(\beta)}{\partial \beta} lr$, where lr is the learning rate (step size). This is done iteratively until we reach the set max iterations or $\frac{\partial J(\beta)}{\partial \beta}$ is within a given tolerance of zero.

Like we introduced Lasso and Ridge regression to avoid overfitting in Project 1, we can add a penalty term to the cost function in Equation 5. In our project we used two different penalties: $L1 = \lambda|\beta|$ and $L2 = \lambda||\beta||^2$. When fitting the model we need to include the derivatives of the penalty term in Equation 6. The gradient with the penalty term is

$$\begin{aligned} \frac{\partial J(\beta)}{\partial \beta} &= \frac{1}{N}\boldsymbol{X}^T(\boldsymbol{p}-\boldsymbol{y}) \ + \lambda \cdot sign(\beta), \ for\ L1\ regularization \\ &\quad or \\ \frac{\partial J(\beta)}{\partial \beta} &= \frac{1}{N}\boldsymbol{X}^T(\boldsymbol{p}-\boldsymbol{y}) \ + \lambda \cdot 2\beta, \ for\ L2\ regularization. \end{aligned} \tag{7}$$