M467/567 Notes

Predictive Analytics: Objective Functions

March 12, 2018

Springer

0.1 Introduction

The goal of predictive analytics is to predict a target value y using a vector of predictor values $\mathbf{x}$ and a function $f(\mathbf{x}|D)$ that has been trained on a data set D. We suppose that the data set consists of pairs of observations on both a target value and predictor vector, say, $D = \{(y_1, \mathbf{x}_1), \ldots, (y_n, \mathbf{x}_n)\}$. The prediction function f is to be applied when $\mathbf{x}$ is observed but not y. For instance, a digital image consists of a vector of pixel-based reflectance intensities $\mathbf{x}$ and the image may contain within it a facial image of a person of interest. The target, unobserved if only $\mathbf{x}$ is known, is the name of the person in the image.

In this discussion, we consider two types of predictive analytics (not completely distinct): regression techniques and machine learning techniques. Regression techniques are well-established and well-understood and based on statistical modeling principles. These methods tend to be simple, mathematically and computationally, as was fitting for their heyday and statistics practitioners. Machine learning, on the other hand, dispenses with statistical modeling principles in favor of computational brute force. The best approach depends on the specifics of the task and the available data.

Objective functions lie at the core of estimation, prediction, and forecasting methods. The objective function connects the general method, e.g., linear regression a specific predictive function. For example, $\widehat{y} = \widehat{\beta}_0 + \widehat{\beta} x_1 + \cdots + \widehat{\beta}_p x_p$ is the specific form of a regression model involving variables $x_1, \ldots, x_p$. The 'hat' above the parameters indicates that real values have been determined for the parameters (the values are determined by fitting the model to the data). One may think of the general method as a blueprint and the specific form as the built object that may be applied to compute the target (estimates, predictions, or forecasts). The process of creating the specific form is usually called *training* when the task is prediction or forecasting, and *model fitting* in the case of estimation. Training a prediction function is accomplished after selecting an objective function that quantitatively measures prediction error and deciding on the general form of the prediction function. Training is accomplished by choosing a specific form of the function that minimizes the prediction error when the rule is applied to the data. For example, if the prediction function is a k-nearest neighbor function, we'll select the neighborhood size k that yields the smallest prediction error. If the target is a quantitative variable, then one choice of objective function is the sum of the absolute error $\sum_{i=1}^{n} |y_i - \widehat{y}_i|$ ($\widehat{y}_i$ is a prediction of y_i). On the other hand, if the target is qualitative (a variable with a levels, say group identifiers), then the error function may be the sum of the *misclassified* targets: $\sum_{i=1}^{n} I_{y_i}(\widehat{y}_i)$, where $I_{y_i}(\widehat{y}_i) = 1$ if $y_i = \widehat{y}_i$ and $I_{y_i}(\widehat{y}_i) = 0$ if $y_i \neq \widehat{y}_i$.

Two methods must be developed to implement this scheme: methods of minimizing the objective function, and 2. methods of assessing error.

0.1.1 Notation and terminology

A *prediction (or forecasting) function* f will be constructed from a training set. The training set is a set of pairs consisting of a predictor vector $\mathbf{x}_i$ and an observed target (either a scalar, or real number, y_i), or a vector $\mathbf{y}_i$. For now, suppose that the target is a scalar y_i. Both quantitative and qualitative targets are included. The input to the prediction function is a *predictor vector* denoted by $\mathbf{x}_t$. The data set is denoted as $D = \{(y_1, \mathbf{x}_1), \ldots, (y_i, \mathbf{x}_i), \ldots, (y_n, \mathbf{x}_n)\}$.

For example, the forecasting problem assumes that the predictor vector $\mathbf{x}_t$ consists p past values observed on a target variable, say, the closing price of a stock, and collected as a row vector

$$\underset{1\times p}{\mathbf{x}_t} = [x_{t-p+1} \quad x_{t-p+2} \quad \cdots \quad x_t].$$

We've changed from using i as the index to t as a reminder that the data are chronologically ordered and that this ordering is important for prediction. The prediction function might be a linear predictor in which case it produces the prediction by combining the target values as a sum, say,

$$f(\mathbf{x}_t|D) = \widehat{\beta}_0 + \sum_{i=1}^{p} x_{t-p+i}\widehat{\beta}_i. \tag{0.1}$$

The coefficient vector $\widehat{\beta} = [\widehat{\beta}_0 \quad \widehat{\beta}_1 \quad \widehat{\beta}_2 \quad \cdots \quad \widehat{\beta}_p]^T$ is determined from the training data and an objective function that measures the forecasting error.

The forecast of $y_{t+\tau}$ can be written more simply as the inner or dot product of $\mathbf{x}_t$ and $\widehat{\boldsymbol{\beta}}$:

$$\widehat{y}_t = \widehat{\beta}_0 + \underset{1\times pp\times 1}{\mathbf{x}_t \quad \widehat{\boldsymbol{\beta}}}\,.$$

0.1.2 Models

A model is a simple abstraction of a process or population. Herein, we consider mathematical models created for the purpose of capturing the primary features of the process and thereby gaining insight to process, or for building a mathematical object that approximates the process.[1] Traditional statistics are strongly dependent on models, and in some sense, the model motivates the methods used by an analyst. Recent advances in machine learning notably do not attempt to realistically model the process of interest but instead use massively large datasets and computational power to build what might

[1] More specifically, the mathematical object is a function or algorithm that generates estimates, predictions, or forecasts.

best be thought of as a large mathematical approximation function (e.g., automatic language translation). With this in mind, the term *model* should not be used for all the cases of interest since there are methodologies that purposely avoid positing a model. There isn't a clear split between model-based and model-free methods, however. In particular, k-nearest neighbor prediction functions and artificial neural networks use what is usually called a model without any effort at realistically creating a simple abstraction of the underlying process that has generated the training data. The prediction function is called a model out of convenience and not because it provides insight to the underlying process.

So, if the objectives are principally prediction and forecasting, then the model is only a stepping stone used to achieve the objective of accurate predictions. There's often little interest in the model itself (a justifiable circumstance if the process is so complicated that forming a realistic model is unrealistic).[2] Our objectives are principally prediction and forecasting, and so estimation will be of concern only when we turn to accuracy assessment.

The application of estimation, prediction, and forecasting methods is guided by the desire to produce accurate estimates, predictions, and forecasts. Doing so requires examples—data—with which accuracy may be estimated and the method may be trained to yield accurate estimates, predictions, and forecasts. Naturally, the data ought to be representative of the process or population to which the method will be applied.[3]

We will discuss several important prediction methods that are model-based in the following sections. Keep in mind that the models are not constructed to be realistic representations of the populations or processes from which the targets of prediction and forecasting originate, but more simply, mathematical approximations of the targets.

0.2 Regression Techniques

We focus on two type of regression methods: linear model/least squares and nonlinear model/maximum likelihood methods.

[2] For example, it's inconceivable that a realistic model can be developed for the application of forecasting the national deficit given the large number of interacting factors that affect the debt. The situation is a little different when the objective is estimation because estimation usually attempts to characterize a population or process and a model in essence, represents or characterizes the population or process. Estimation is is a central theme of statistics. Accurate estimation requires both approximately realistic model and representative data.

[3] Creating a representative data set is a central theme of statistics. We'll proceed under the assumption that the data is representative and focus on the problem of achieving maximal accuracy.

0.2.1 Linear models and least squares

The linear model prediction function is

$$\widehat{y} = \beta_0 + \sum_{i=1}^{p} \beta_i \mathbf{x}_i. \tag{0.2}$$

Using vector notation, $\widehat{y} = \mathbf{x}\boldsymbol{\beta}$ after having set

$$\underset{1\times q}{\mathbf{x}} = [1 \;\; x_1 \;\; x_2 \;\; \cdots \;\; x_p],$$

$\boldsymbol{\beta} = [\beta_0 \; \beta_1 \;\; \cdots \;\; \beta_q]^T$ and $q = p + 1$. We've departed slightly from the notation used in the earlier example (equation 0.1.1). The least squares objective function is most often used for linear models , and it is

$$\varepsilon(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \mathbf{x}_i\boldsymbol{\beta})^2, \tag{0.3}$$

where $(y_i, \mathbf{x}_i) \in D$. In matrix form, the objective function is

$$\varepsilon(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}), \tag{0.4}$$

where $\mathbf{y}$ is an n-vector of observed target values and $\mathbf{X}$ is $n\times p$ and constructed by stacking the n predictor vectors as rows. The least squares estimator of $\boldsymbol{\beta}$ is determined by differentiating $f(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$, setting the vector of derivatives equal to $\mathbf{0}$, and solving for $\boldsymbol{\beta}$. This process yields the least squares estimator

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{0.5}$$

Equation (0.5) defines the best possible coefficient vector in terms of minimizing the errors when the prediction function is applied to the data set. It's important to note that the data has been used twice, in a somewhat abstract sense: once to determine the coefficients and a second time to assess the error. A completely distinct *test* data set might be used to assess the error and would surely yield a different sum of squared errors (even if the number pairs in test set were the same). If both sets were representatively drawn from the same population, the training set estimate of error will be smaller than the test set. This phenomenon, known as *over-fitting* will be explored in detail later, but for now, we often can improve the prediction error when the prediction function is applied to data sets besides the training set by introducing a regularization term into the original objective function.[4] The revised objective function is

[4] The consequence of introducing the regularization term is to reduce the norm of the coefficient vector.

$$\begin{aligned}\varepsilon_R(\boldsymbol{\beta}) &= (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_2^2 \\ &= (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}) + \lambda(\boldsymbol{\beta}^T\boldsymbol{\beta}),\end{aligned} \tag{0.6}$$

where $||\boldsymbol{\beta}||_2$ is the L_2, or Euclidean norm of the vector $\boldsymbol{\beta}$. It's relatively easy to determine the regularized coefficient vector by solving the matrix equation $\partial\varepsilon_R(\boldsymbol{\beta})/\partial\boldsymbol{\beta} = \mathbf{0}$ for $\boldsymbol{\beta}$. The solution is

$$\widehat{\boldsymbol{\beta}}_R = (\mathbf{X}^T\mathbf{X} + \mathbf{I}_p)^{-1}\mathbf{X}^T\mathbf{y}. \tag{0.7}$$

0.2.2 Least absolute deviation regression and the lasso

Let us consider modifying the least squares objective function. The change is to replace the sum of the squared errors by the sum of the absolute errors. The difference between the two objective functions lies with the effect of a particular difference $y_1 - \widehat{y}_i$ on the objective function. With regard to the absolute error objective function, the influence of an error equal to η in magnitude on the objective function is half of the effect of an absolute error of 2η. If the objective function is the sum of the squared errors, then the effect of an absolute error equal to 2η is instead $4\eta^2$. The effect is by large-magnitude errors have greater influence on the least squares estimator than the absolute error function. Using the sum of the absolute errors is referred to as least absolute deviation regression and, alternatively, as L_1 regression because the objective function is the L_1 norm of the vector of deviations $\mathbf{y} - \widehat{\mathbf{y}}$.

The absolute errors objective function is

$$\begin{aligned}\varepsilon(\boldsymbol{\beta}) &= \sum_{i=1}^{n} |y_i - \mathbf{x}_i\boldsymbol{\beta}| \\ &= \mathbf{j}^T|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}|,\end{aligned} \tag{0.8}$$

where $\mathbf{j}$ is a n-vector of 1's and $|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}| = [|y_1 - \mathbf{x}_1\boldsymbol{\beta}| \ \cdots \ |y_n - \mathbf{x}_n\boldsymbol{\beta}|]^T$ is and a regularization term as a objective function. Prediction based on this objective function is sometimes referred to as L_1 regression since the sum of the absolute errors is known in mathematics as the L_1 norm of error vector $\mathbf{y} - \mathbf{X}\beta$.

Introducing a regularization term not only helps reduce prediction error but also may used for variable selection, in which case, the method is referred to as the lasso.[5][1] The regularization term is not quite the same as in equation (0.6) because the L_2 norm of $\boldsymbol{\beta}$ is replaced with the L_1 norm. The objective function is therefore

[5] lasso is an acronym for *least absolute shrinkage and selection operator*.

$$\begin{aligned}\varepsilon(\boldsymbol{\beta}) &= \sum_{i=1}^{n} |y_i - \mathbf{x}_i\boldsymbol{\beta}| + \lambda \sum_{i=0}^{p} |\beta_i| \\ &= \mathbf{j}^T|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}| + ||\boldsymbol{\beta}||_1,\end{aligned} \tag{0.9}$$

where$||\boldsymbol{\beta}||_1$ is the L_1 norm of $\boldsymbol{\beta}$. At the present time, our interest is in the computational challenge of minimizing the objective function.

We may follow the approach used in least squares regression—differentiate the objective function with respect to $\boldsymbol{\beta}$, set the vector of partial derivatives equal to the zero vector and attempt to solve the matrix equation for $\boldsymbol{\beta}$. It that can vbe accomplised, then a closed-form, or analytical form will have be determined for the estimator of $\boldsymbol{\beta}$.

First, the absolute value function is differentiable everywhere but 0, and for $x \neq 0$,

$$\begin{aligned}\frac{\mathrm{d}|x|}{\mathrm{d}x} &= \begin{cases} -1, & \texttt{if } x < 0, \\ 1, & \texttt{if } x > 0 \end{cases} \\ &= I_+(x) - I_+(-x),\end{aligned} \tag{0.10}$$

where

$$I_+(x) = \begin{cases} 1, & \texttt{if } x > 0 \\ 0, & \texttt{if } x \leq 0 \end{cases}$$

If we ignore the possibility that one of the β_i's may be zero, $\partial\varepsilon(\boldsymbol{\beta})/\partial\boldsymbol{\beta}$ can be computed and set to $\mathbf{0}$. However, an analytical (closed-form) solution cannot be found.[6] The minimizing value $\widehat{\boldsymbol{\beta}}$ will have to be found using an alternative approach.

0.2.3 Pattern search

Pattern search (or direct search) is an iterative algorithm that does not utilize the derivatives of the objective function. In brief, it begins with an arbitrary initial value for $\boldsymbol{\beta}$. Then, an iteration phase commences and on each iteration, each coefficient is perturbed by adding or subtracting a constant. It the modified vector yields a smaller value for the objective function, then the coefficient vectors is updated (the perturbation, say β_i is replaced by $\beta_i + \epsilon$, otherwise, the coefficient vector is not changed). If, after cycling through each coefficient, there's be no update, then the perturbation is made smaller (halved, say) and the next iteration repeats the process of cycling through each coefficient with the smaller perturbations. There's no guarantee that the algorithm will succeed at finding the minimum value, but if it is used with

[6] If there were an analytical solution, then the median of a set of observations could be computed using a formula. This, of course not the case, and we must resort to a sorting or sorting-like algorithm to find the median.

care, it probably will produce a coefficient vector very near the minimizing value $\widehat{\boldsymbol{\beta}}$.

The steps to building and implementing a pattern search algorithm are described in more detail:

1. Create a function $\varepsilon(\mathbf{y}, \widehat{\mathbf{y}})$ that computes the objective function from the actual and predicted target vectors.
2. Set the number of iterations, say $n_I = 100$, and the initial perturbation, say $\eta = 1$.
3. Initialize $\boldsymbol{\beta}$, say $\boldsymbol{\beta}_0 = \mathbf{0}$.
4. Compute $\widehat{\mathbf{y}}$ using $\boldsymbol{\beta}_0$ and then $\varepsilon(\mathbf{y}, \widehat{\mathbf{y}})$.
5. Begin iterating. On each iteration,
   a. Create a copy[7] of $\boldsymbol{\beta}$, say $\boldsymbol{\beta} \to \boldsymbol{\beta}_i$.
   b. Cycle through each coefficient $\beta_{i,0}, \beta_{i,1}, \ldots, \beta_{i,p}$ and do the following:
      i. Perturb $\beta_{i,j}$ by assigning $\beta_{i,j} \leftarrow \beta_{i,j} + \eta$ where $\beta_{i,j}$ is the current value of the jth coefficient.
      ii. Compute $\widehat{\mathbf{y}}$ using $\boldsymbol{\beta}_i$.
      iii. Compute $\varepsilon(\mathbf{y}, \widehat{\mathbf{y}})$.
      iv. If $\varepsilon(\mathbf{y}, \widehat{\mathbf{y}})$ has *not* been reduced relative the the previous value of $\varepsilon(\mathbf{y}, \widehat{\mathbf{y}})$, then reset the coefficient by computing $\boldsymbol{\beta}_{i,j} - \eta \to \boldsymbol{\beta}_{i,j}$.
      v. Cycle through each coefficient $\beta_0, \beta_1, \ldots, \beta_p$ but *subtract* η instead of adding η. Carry out steps 5(b)ii, 5(b)iii and 5(b)iv except that reseting the coefficient (when necessary) is accomplished by computing $\boldsymbol{\beta}_{i,j} + \eta \to \boldsymbol{\beta}_{i,j}$.
   c. If the objective function was not reduced in step 5b, then reduce η by a factor between 0 and 1, say, $\eta \leftarrow .7\eta$. If there was a reduction in the objective function, then do not change η.
   d. Print the iteration counter, $\varepsilon(\mathbf{y}, \widehat{\mathbf{y}})$, η, and a boolean variable indicating whether or the objective function was reduced during the current iteration.
   e. Begin the next iteration (return to step 5a).

Pattern search will always produce a coefficient vector that is near the true optimal vector if the objective function has no local minimums that can trap the algorithm. There are numerous other more sophisticated algorithms for finding optimal vectors. For the time being, we turn now to logistic regression as it is an important prediction method and draws on several central subjects of predictive analytics.

[7] We use the arrow to indicate that the value of $\boldsymbol{\beta}$ is assigned to $\boldsymbol{\beta}_i$.

0.3 Newton's method

Newtons' method is a far more efficient means of optimizing an objective function than pattern search. There are limitations on it's utility since it can be used only if the function is differentiable. If the function of interest is an objective function, then it is also necessary that the matrix of second derivatives, say $\partial\varepsilon^2(\boldsymbol{\beta})/\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T$ can be computed.[8] We begin the discussion assuming that the function of interest is univariate, differentiable, and defined on $\mathbb{R}$.

A function f that is differentiable on the interval $[a, b]$ has a derivative at every $x \in [a, b]$.[9] This condition is satisfied often enough by commonly used objective functions that Newton's method is useful algorithm for predictive analytics. In fact, it's widely used for maximum likelihood estimation, including logistic regression and other *generalized linear models*. In most applications, the objective function f is defined for every real number in an interval of interest. For this discussion, let's assume that f has a unique global minimum at x^*,[10] and that the objective is to determine x^*. Since f is differentiable, the derivative of f at $x_0 \in \mathbb{R}$ is the limit[11]

[8] The matrix of second derivative is referred to as the Jacobian of f and sometimes the *Hessian* matrix.

[9] In other words, $\mathrm{d}f(y)/\mathrm{d}y|_x$ exists for every $x \in [a, b]$.

[10] Local minima are not uncommon in predictive analytics.

[11] Mathematician rarely compute derivatives by directly computing the limit shown in equation (0.12) because there are a handful of rules for computing derivatives that are easier to apply. For the sake of understanding derivatives, though, let's compute the derivative of $f(x) = x^2$. First, let's change the expression slightly by setting $\epsilon = x - x_0$. Then,

$$f'(x_0) = \lim_{\epsilon \to 0} \frac{f(x_0 + \epsilon) - f(x_0)}{\epsilon}.$$

Now replace $f(x_0)$ and $f(x_0 + \epsilon)$ with x_0^2 and $(x_0 + \epsilon)^2 = x_0^2 + 2x_0\epsilon + \epsilon^2$:

$$\begin{aligned} f'(x_0) &= \lim_{\epsilon \to 0} \frac{x_0^2 + 2x\epsilon + \epsilon^2 - x_0^2}{\epsilon} \\ &= \lim_{\epsilon \to 0} \frac{2x_0\epsilon + \epsilon^2}{\epsilon} \\ &= 2x_0 + \lim_{\epsilon \to 0} \epsilon = 2x_0. \end{aligned} \tag{0.11}$$

Now, we must imagine a number ϵ that's positive to begin with and shrinks to 0. For example, imagine a series of infinitely many values and the ith value is the previous value divided by 2. The limit $\lim_{\epsilon \to 0} \epsilon$ is the outcome of this mental exercise, and for all practical purposes, the outcome is 0. The argument justifying this statement use the fact that the limit is positive because any positive number divided by 2 is positive. Now, if you propose any nonzero number to be the limit, there is a step at which the value of ϵ is smaller (by going sufficiently far into the series) and so that value is better approximation of the limit. As there is no smallest number greater than zero, zero is the limit, and we write $\lim_{\epsilon \to 0} \epsilon = 0$. Returning to the last equation, we deduce that $f'(x_0) = 2x_0$.

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} \tag{0.12}$$

For the reader that is not familiar with calculus, the fraction in equation (0.12) is equivalent to the slope of a line $\{(x, f(x))|x \in \mathbb{R}\}$ assuming that f is a linear function. Therefore, the derivative of a function f at x_0 may be thought of as the slope of a linear approximation to f made at x_0.

If x is close to x_0, then we can approximate $f(x_0)$ rearranging the approximation

$$f'(x_0) \approx \frac{f(x) - f(x_0)}{x - x_0} \tag{0.13}$$

to obtain the Taylor series first-order approximation of f at x_0:

$$f(x_0) \approx f(x) + f'(x)(x_0 - x). \tag{0.14}$$

Equation (0.14) is a *linear approximation* since $f(x) - f'(x)x$ is constant and not depending on the variable x_0, and the variable x_0 is multiplied by a coefficient $f'(x)$. The sum of the two terms describes as linear function. The approximation also can be rearranged again to yield an approximation of x that satisfies an equation, say $y = f(x)$ providing that x_0 is close to x and the values $f(x_0)$ and $f'(x_0)$ are known. To do so, replace $f(x)$ with the value y and choose x_0 near x. The approximation x_1 of x is

$$x_1 = x_0 + \frac{f(x) - f(x_0)}{f'(x_0)} = x_0 + \frac{y - f(x_0)}{f'(x_0)}. \tag{0.15}$$

The accuracy of the approximation is depends on how far x_0 is from x (closer is better of course) and the degree of linearity of f at x_0. Newton's method successively computes improved approximations of x.

Specifically, an algorithm implementing Newton's method begins with a calculation of x_1 according to equation (0.15) and an initial guess x_0. Then, iterations commence using the iteration counter k and on each iteration, the previous value x_{k-1} is replaced with the most recently computed x_k. So, on iteration k, compute

$$x_k = x_{k-1} + \frac{y - f(x_{k-1})}{f'(x_{k-1})}. \tag{0.16}$$

As $f(x_{k-1})$ approaches y, $y - f(x_{k-1})$ tends to zero and the adjustment to the previous approximation tends to zero.

For many functions, Newton's method will not only converge to the solution, but the rate of convergence very rapid. For many problems, the convergence rate is quadratic and so as iterations take place, the difference between the solution and the approximation is a power of 2.[12]

[12] The difference would follow a sequence such as $10^{-2}, 10^{-4}, 10^{-8}, \ldots$.

The first-order Taylor series approximation of a multivariate function $f : \mathbb{R}^p \to \mathbb{R}^q$ is

$$\underset{q\times 1}{f(\boldsymbol{\beta}_0)} \approx \underset{q\times 1}{f(\boldsymbol{\beta})} + \underset{q\times p}{\frac{\partial f(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T}}(\underset{p\times 1}{\boldsymbol{\beta}_0} - \boldsymbol{\beta}). \tag{0.17}$$

0.3.1 Applying Newton's method to objective functions

We consider an objective function ε that is differentiable and maps $\mathbb{R}^p$ to $\mathbb{R}^+$, the non-negative real numbers, for some non-negative integer p. We'll use $\boldsymbol{\beta}$ as the argument; hence, $\varepsilon : \boldsymbol{\beta} \mapsto c \in \mathbb{R}^+$. Newton's method will be used to determine $\widehat{\boldsymbol{\beta}}$ that minimizes ε. To use Newton's method as it has been described requires the value $y = \varepsilon(\widehat{\boldsymbol{\beta}})$, which is rarely known in advance. The solution to this difficulty utilizes that fact that $\partial\varepsilon/\partial\boldsymbol{\beta}|_{\widehat{\boldsymbol{\beta}}} = \mathbf{0}$, since $\widehat{\boldsymbol{\beta}}$ minimizes $\varepsilon(\cdot)$. Therefore, we use Newton's method to solve the equation $\partial\varepsilon(\widehat{\boldsymbol{\beta}})/\partial\boldsymbol{\beta} = \mathbf{0}$ for $\boldsymbol{\beta}$.

Before proceeding to the multivariate case, let us develop the univariate case ($p = 1$). Using equation (0.16) and setting $y = 0$ leads to the updating formula for the kth iteration:

$$\beta_k = \beta_{k-1} - \frac{\varepsilon'(\beta_{k-1})}{\varepsilon''(\beta_{k-1})}, \tag{0.18}$$

for $\varepsilon''(\beta_{k-1}) \neq 0$.

Now consider a p-vector $\boldsymbol{\beta}$, for $p > 1$ and the problem of finding $\widehat{\boldsymbol{\beta}}$ that minimizes $\varepsilon(\boldsymbol{\beta})$. In general, $\varepsilon(\boldsymbol{\beta})$ is unknown, but if ε is differentiable, then it is known that $\partial\varepsilon(\widehat{\boldsymbol{\beta}})/\partial\boldsymbol{\beta} = \mathbf{0}$ and Newton's method can be used to solve the matrix equation

$$\mathbf{0} = \frac{\partial\varepsilon(\widehat{\boldsymbol{\beta}})}{\partial\boldsymbol{\beta}} = f(\widehat{\boldsymbol{\beta}}) \tag{0.19}$$

for $\widehat{\boldsymbol{\beta}}$. Since $\widehat{\boldsymbol{\beta}}$ minimizes $\varepsilon(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$, $f(\widehat{\boldsymbol{\beta}}) = \mathbf{0}$, and the first-order Taylor series expansion of $f : \mathbb{R}^p \to \mathbb{R}^p$ about $\widehat{\boldsymbol{\beta}}$ is

$$\begin{aligned} \mathbf{0} &\approx f(\boldsymbol{\beta}) + \underset{p\times p}{\frac{\partial f(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T}}(\underset{p\times 1}{\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}}) \\ &= \underset{p\times 1}{\frac{\partial\varepsilon(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}}} + \underset{p\times p}{\frac{\partial\varepsilon^2(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}}(\underset{p\times 1}{\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}}). \end{aligned} \tag{0.20}$$

Re-arranging equation (0.20) provides the updating formula for determining $\widehat{\boldsymbol{\beta}}$ by Newton's method:

$$\boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} - \left(\frac{\partial \varepsilon^2(\boldsymbol{\beta}_{k-1})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right)^{-1} \frac{\partial \varepsilon(\boldsymbol{\beta}_{k-1})}{\partial \boldsymbol{\beta}}. \tag{0.21}$$

Obviously, the inverse of the matrix of second derivatives (known as the *Hessian* matrix) must exist to use Newton's method.

0.4 Exercises

1. Compute $\mathrm{d}x^{-1}/\mathrm{d}x$ using a limit definition of the derivative (formula 0.12.)
2. Provide the `Python` code for finding a solution to the equation $f(x) = x^2 - 2$ using Newton's method or pattern search. (This is a clever way to compute $\sqrt{2}$ if your smart phone needs to be recharged).
3. Provide the `Python` code (and the solution) for solving the equation $1 = e^x - x$ for x using Newton's method or pattern search.
4. Provide the `Python` code for finding all of the roots of the polynomial $g : [-5, 20] \to \mathbb{R}$ given by

   $$g(x) = 30 + .1x - 3x^2 + .2x^3 - .001x^4. \tag{0.22}$$

   Hint: graph the function so that you can visually identify initial values for solving for each of the roots. What are the roots?
5. Suppose that the objective function of interest is by equation (0.4). Derive the updating formula for applying Newton's method to the computation of $\widehat{\boldsymbol{\beta}}$, thereby showing that only a single iteration is necessary to arrive at the minimizing value of $\boldsymbol{\beta}$.
6. Suppose that the objective function is the *cross-entropy* function

   $$\varepsilon(\boldsymbol{\beta}) = -\sum_{i=1}^{n} y_i \mathbf{x}_i \boldsymbol{\beta} + \sum_{i=1}^{n} \log\left[1 + \exp(\mathbf{x}_i \boldsymbol{\beta})\right], \tag{0.23}$$

   where $\mathbf{x}_i$ is $1 \times q$ and $\boldsymbol{\beta}$ is $q \times 1$. Equation (0.23) is the objective function used for logistic regression.[13] Verify the updating formula for Newton's method applied to equation (0.23) is

   $$\underset{q\times 1}{\boldsymbol{\beta}_k} = \underset{q\times 1}{\boldsymbol{\beta}_{k-1}} + (\underset{q\times q}{\mathbf{X}^T \mathbf{W}_{k-1}} \underset{q\times n}{\mathbf{X}})^{-1} \underset{q\times n}{\mathbf{X}^T} [\underset{n\times 1}{\mathbf{y}} - \underset{n\times 1}{\boldsymbol{\pi}(\boldsymbol{\beta}_{k-1})}], \tag{0.24}$$

   where

   $$\pi_i(\boldsymbol{\beta}_{k-1}) = [1 + \exp(-\mathbf{x}_i \boldsymbol{\beta}_{k-1})]^{-1}$$
   $$\boldsymbol{\pi}(\boldsymbol{\beta}_{k-1}) = [\pi_1(\boldsymbol{\beta}_{k-1}) \quad \pi_2(\boldsymbol{\beta}_{k-1}) \quad \cdots \quad \pi_n(\boldsymbol{\beta}_{k-1})],$$

[13] Cross-entropy is the negative of the log-likelihood, and so $\widehat{\boldsymbol{\beta}}$ can be determined by *minimizing* equation 0.23 with respect to $\boldsymbol{\beta}$

and

$$\underset{n\times n}{\mathbf{W}_{k-1}} = \text{diag}\{\pi_i(\boldsymbol{\beta}_{k-1})[1-\pi_i(\boldsymbol{\beta}_{k-1})]\}. \tag{0.25}$$

7. Write a function that carries out pattern search optimization. (See section 0.2.3 above). Compute the least squares estimator using the Boston housing data[14]. The data and a `Python` function to read the data are posted on the Moodle website in the *Introduction to Predictive Analytics* topic section. The target variable is `medv`—the median value of owner-occupied homes in $1000's. Use all other 12 variables as predictors, and as usual, the constant 1.

   a. Compute the least squares estimator of $\boldsymbol{\beta}$ directly[15] and using your pattern search algorithm. Verify that the two methods yield coefficient vectors that are essentially the same.
   b. Compute the coefficient vector using regularized least squares. (You will have to modify the objective function to include the penalty term). It may be helpful to verify that the usual least squares estimator is computed when $\lambda = 0$. Then set $\lambda = .001$.

8. Write a Newton's method algorithm for logistic regression. It's assumed that a design matrix $\mathbf{X}$ and target vector $\mathbf{y}$ have been constructed as `Numpy` matrices of dimension $n \times q$ and $n \times 1$, respectively. Presumably, one of the columns of $\mathbf{X}$ consists of ones.

   a. Write a function that computes the log-likelihood function (equation **??**). The arguments passed to the function are $\mathbf{X}$ and $\mathbf{y}$, and a coefficient vector $\boldsymbol{\beta}$, specifically, the kth iteration estimate of $\boldsymbol{\beta}$. The function returns the log-likelihood evaluated at the input vector $\boldsymbol{\beta}$.
   b. Create the initial estimate of $\boldsymbol{\beta}$ as a `Numpy` matrix, say

```
b = np.matrix(np.zeros(shape = (q, 1)))
```

   c. Create a `while` loop that iterates until the change between successive evaluations of the objective function is smaller than some ϵ. The shell of the loop is:

```
k = 0
while (abs(previousValue - objFnValue)) > 1e-4:
    previousValue = objFnValue
    ...
    objFnValue = objFn(X, Y, b)
    print(k, objFnValue)
    k += 1
```

[14] Details: `https://www.kaggle.com/c/boston-housing`

[15] Specifically, compute $\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$.

On the kth iteration, $k = 1, 2, \ldots$, Carry out the following operations:

d. Compute $\pi(\boldsymbol{\beta}_k)$ according to

$$\begin{aligned} \pi_i(\boldsymbol{\beta}_{k-1}) &= [1 + \exp(-\mathbf{x}_i\boldsymbol{\beta}_{k-1})]^{-1}, \\ \boldsymbol{\pi}(\boldsymbol{\beta}_{k-1}) &= [\pi_1(\boldsymbol{\beta}_{k-1}) \;\; \pi_2(\boldsymbol{\beta}_{k-1}) \;\; \cdots \;\; \pi_n(\boldsymbol{\beta}_{k-1})]. \end{aligned} \tag{0.26}$$

e. Compute the product $\mathbf{W}_k\mathbf{X}$. A computationally efficient method initializes $\mathbf{W}_k$ as a copy of $\mathbf{X}$, say, `WX = X.copy()`. Then, iterate over the rows of `WX` and replace ith row by the product of $\mathbf{x}_i$ and $w_{ii} = \pi_i(\boldsymbol{\beta}_k)[1 - \pi_i(\boldsymbol{\beta}_k)]$, say,

```
for i, x in enumerate(WX):
    WX[i,:] *= float((1 - piVector[i])*piVector[i])
```

The operation `a *= b` multiplies `a` by `b` and assigns the product to `a`.

f. Compute $\mathbf{X}^T\mathbf{W}_k\mathbf{X}$ and at the same time update $\boldsymbol{\beta}_k$ according to

$$\underset{q\times 1}{\boldsymbol{\beta}_k} = \underset{q\times 1}{\boldsymbol{\beta}_{k-1}} + (\underset{q\times q}{\mathbf{X}^T\mathbf{W}_{k-1}\mathbf{X}})^{-1}\underset{q\times n}{\mathbf{X}^T}[\underset{n\times 1}{\mathbf{y}} - \underset{n\times 1}{\boldsymbol{\pi}(\boldsymbol{\beta}_{k-1})}] : \tag{0.27}$$

```
b += np.linalg.solve(X.T*WX, X.T*(Y - piVector))
```

g. Compute the objective function and increment k. Program flow must then return to the beginning of the `while` loop so that the test (`(abs(previousValue - objFnValue)) > 1e-4`) is evaluated.

h. Complete the function by returning $\boldsymbol{\beta}_k$.

i. It's helpful to print the iteration counter and the objective function on each iteration (at least while coding the function).

j. Apply the estimation function to the Wisconsin breast cancer data set to compute $\widehat{\boldsymbol{\beta}}$.[16] The data, and a `Python` function to read the data file, build the design matrix $\mathbf{X}$, and the target vector $\mathbf{y}$ are posted on Moodle.

It's useful to note that ith row of $\underset{n\times q}{\mathbf{W}_{k-1}\mathbf{X}}$ is $[x_{i,1}w_i \;\; x_{i,2}w_i \;\; \cdots \;\; x_{i,q}w_i]$.

It's probably easier and more efficient to compute $\mathbf{W}_{k-1}\mathbf{X}$ by row rather than forming the diagonal matrix $\mathbf{W}$ and then computing the product of $\mathbf{W}$ and $\mathbf{X}$.

k. Report the *apparent accuracy* given by the proportion of predictions $\widehat{y}_i, i = 1, 2, \ldots, n$ that are equal to the actual values $y_i, i = 1, 2, \ldots, n$.

[16] `https://www.kaggle.com/uciml/breast-cancer-wisconsin-data`

References

1. G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning with Applications in R.* Springer, 2013.