Regression Tutorial

Dev Rajnarayan

6 May, 2010.

1 Introduction

The topic of regression deals with continuous input-output systems, and the goal is to 'learn' a given system. Learning is measured by the ability to predict the output of the system to any given input. A predictor algorithm is created and tuned by a process called training, during which we are given a typical set of known input-output pairs, called the training data[1], or often just 'the data'. This training process is a meta-algorithm, running one level up from the actual predictor algorithm. Whereas the predictor generates values that mimic the output of the system, the output of training algorithm is the predictor itself.

Machine-learning researchers pride themselves on developing smart learning algorithms, and this usually means developing these training algorithms, which often end up as instances of optimization problems. In contrast, people that *apply* regression to their applications use existing training algorithms to make predictors from their data, and then use these predictors for forecasting, data analysis, optimization, and so on. Using a predictor is usually a simple task, often as easy as matrix multiplication. This document briefly describes both these tasks: making predictors, and then using them to make predictions.

1.1 Notes on Using This Document

All links in this document are active, which means that footnotes, citations, and even urls can be clicked on. In terms of notation, vectors are column vectors unless otherwise indicated. We have not used boldface for vectors and matrices; this makes the text look cleaner, but it also requires more care while reading. The superscript (i) denotes the i^{th} sample. In the probabilistic sections, we have been slightly lax with mathematical notation in the interests of readability.

The topics as presented here are extremely small slivers extracted from whole reams of theory. The interpretations here are by no means complete, but they do aim to provide a reasonable summary of the main results. The interested (or dissatisfied) reader is invited to delve deeper into any of these topics, and upon doing so will find that, while the story here may not be the whole truth, it is by no means a blatant lie. It is perhaps better to have a dissatisfied reader delving than an overwhelmed one quitting.

[1] The problem of actually generating this data is an interesting field in itself, called Design Of Experiments (DOE).

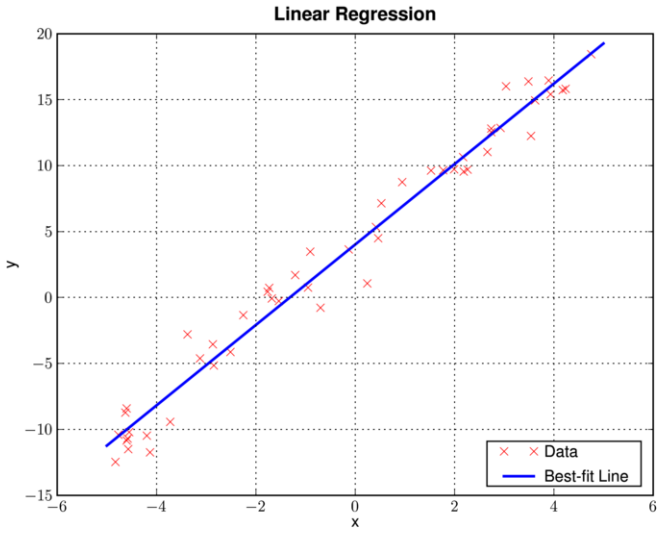

Figure 1: Simple example of linear regression.

1.2 Example: 1-D Linear Regression

We begin using the simplest of examples, one that should be familiar to readers with any science background. Suppose we have an input-output system that takes inputs $x \in \mathbb{R}$ to outputs $y \in \mathbb{R}$. Visual examination of a set of samples $\mathcal{D} = \{(x^{(i)}, y^{(i)}),\ i = 1, \ldots, m\}$ (the data) suggests that the input-output relationship is close to affine (linear + constant). A familiar concept is that of a best-fit line to the data $\mathcal{D}$, as shown in Fig. 1. Any line in two dimensions can be parametrized by a slope m and an intercept c. Consequently, any affine predictor is given by $y = mx + c$. The best-fit predictor $\hat{y}$ uses 'optimal' values of m and c. Computing these is the task of the training algorithm, and in this case, is fairly straightforward[2]. Using the regression is also easy: just multiply a given input x by m and add c.

2 Parametric Regression

The example above is possibly the simplest example of parametric regression, a class of regression problems where the explicit functional form of the predictor is known. In the general case, we have a system g that takes inputs $x \in X$ to outputs $y \in Y$. The predictor is a function f of its argument x, and is completely defined by a vector θ of parameters, and is written as follows:

$$\hat{y} = f(x; \theta). \tag{1}$$

Here, the semicolon serves to separate the arguments to the function from the parameters that define it. As before, the training algorithm is a means to compute a θ that is optimal in some sense. Since $\hat{y} = f(x; \theta)$ is our approximation to the true system $y = g(x)$, a widely-used measure of optimality is the

[2]By all means, try this at home. You won't break anything.

sum of squared prediction errors on the training data. In other words, the optimal value $\theta^\star$ is given by

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{m} \left[f(x^{(i)}; \theta) - y^{(i)} \right]^2 \tag{2}$$

In the case of parametric regression, the predictor is so completely defined by θ that once the training algorithm has generated an optimal set of parameters $\theta^\star$ from some training data, we may as well throw away that data, we no longer need it for prediction!

2.1 Example: Linear Regression Revisited

We now consider the n-dimensional analog of the regression problem in Sec. 1.2. The true system g now takes inputs $x \in \mathbb{R}^n$ to outputs $y \in \mathbb{R}$. For a change, we use a regression model that is not affine, but strictly linear $\hat{y} = \sum_{j=1}^{n} a_j x_j = a^T x$. We want to find the vector a that minimizes the sum of squared prediction errors on the training data

$$F(a) = \sum_{i=1}^{m} [y^{(i)} - \hat{y}^{(i)}]^2 = \sum_{i=1}^{m} [y^{(i)} - a^T x^{(i)}]^2. \tag{3}$$

We construct a matrix X of all the inputs $x^{(i)},\ i = 1, \ldots, m$ in the training data $\mathcal{D}$, by stacking them row-wise. Similarly, we concatenate the outputs to form the vector y. In matrix notation, we want to minimize $F(a) = \|Xa - y\|_2^2$. The optimal a for this problem is obtained by setting the derivative of $F(a)$ to zero, which yields the familiar least-squares solution $a^\star = (X^T X)^{-1} X^T y$. In MATLAB, this is achieved by typing '`aStar = X\y`'. Our training algorithm is therefore nothing but '`aStar = X\y`', and using the regression is as simple as performing the vector-vector product $\hat{y} = a^T x$.

2.2 Example: Quadratic Fit

In this example, we complicate things just a little, and use the regression model

$$\hat{y} = \frac{1}{2} x^T P x + q^T x + r. \tag{4}$$

It turns out that solving for the optimal parameters in this case just as easy as in the previous case. We just append each input vector $x^{(i)}$ with the higher-order terms $x_j^{(i)} x_k^{(i)}$, for $j = 1, \ldots, n,\ k = j, \ldots, n$, thus forming what's called a 'feature vector' ϕ. Call the corresponding column-extended matrix Φ. The entries in the matrix P, the vector q, and the scalar r, all get concatenated into one giant vector a, and now $\hat{y} = a^T \phi$. The resulting optimization problem, minimize $\|\Phi a - y\|_2^2$, is very similar to that in Sec. 2.1. The training algorithm is as simple as '`aStar = Phi\y`'. Using the regression is a little more involved now, and we have two equivalent options:

1. Take x, construct the feature vector ϕ by forming higher-order products, and then use $a^T \phi$,
2. Back out P, q and r from the optimal parameter vector $a^\star$, and then use Eq. 4.

2.3 Merits and Demerits of Parametric Regression

Parametric regression has the wonderful ability to compress most or all of the underlying information in a huge data set into a relatively small set of parameters θ. Once we have θ, we no longer need to deal with this large data set for prediction. As we demonstrated in the preceding sections, the training algorithms are usually simple and very robust. Often, using the regression for prediction is also very straightforward.

Nevertheless, in order to apply this technique, we need to first propose a parametric form $f(x;\theta)$. We need some rationale to justify any such parametrization, and indeed such justification is often possible: Taylor's theorem assures that in a sufficiently small neighbourhood, *any* smooth function looks affine, and on a slightly larger scale, quadratic. Physics or other domain-specific theory might yield insights into reasonable closed-form parametric approximations for a given system. Unfortunately, in the absence of such reasoning, we have little justification for using parametric models.

3 Nonparametric Regression and Gaussian Processes

It so happens that, even if we cannot make assumptions as strong as a functional form for our model, we can almost always propose weaker assumptions on system behaviour. For instance, it would be very useful to encode system knowledge such as "the exact functional form is not known, but it is more likely to be a smooth function that not", or "the system may not be representable by a smooth function, but outputs corresponding to nearby inputs are more similar than outputs corresponding to inputs far apart". As it turns out, we can in fact do *exactly* this using nonparametric regression techniques, the topic of this section.

Nonparametric regression differs from parametric regression in that we always need to keep all of the training data $\mathcal{D}$ to make predictions. The regressor still is a parametrized function, but these parameters do not completely define the function. Mathematically, this means that $\hat{y} = f(x;\mathcal{D},\theta)$.

Of the several nonparametric regression techniques in use, we focus on Gaussian Process Regression (GPR), a regression technique based on probabilistic models. For an excellent exposition on GPR, see Rasmussen and Williams (2006), they even have a free online version[3] of the book. We choose GPR for three reasons: one, it allows us to make precisely the kind of assumptions described above; two, many other nonparametric regression techniques happen to be specific instantiations of this general technique; and three, since it is a probabilistic technique, the predictor gives us much more than just a point prediction. One variant of GPR was first applied in geostatistics to model ore concentrations (Krige, 1951), and is therefore often called 'Kriging'.

Our description of GPR is divided into three parts. First, we introduce some background material on multivariate Gaussian random variables. Next, we will discuss the basic assumptions of GPR, one of which is that the outputs of the system, both in the given data and the desired prediction locations, can be thought of jointly as multivariate Gaussian random variables. Finally, putting these two facts together makes the task of prediction extremely straightforward.

[3]The hardcover version is inexpensive, and is a worthwhile investment for anyone even mildly interested in the field.

3.1 Multivariate Gaussian Random Variables

The notion of a scalar Gaussian distribution (also called a normal distribution) is quite familiar in many fields. The Gaussian probability density function is the familiar bell curve, given by

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right). \tag{5}$$

The density is completely specified by two scalar parameters: a mean μ, and a variance σ^2 (the standard deviation is σ.) Many random events in nature are Gaussian or close to Gaussian. There is a truly deep reason for this, encapsulated in the Central Limit Theorem.

While this applies to a scalar random variable, there is an analog for vectors of random numbers too. A vector $y \in \mathbb{R}^n$ is a multivariate Gaussian random variable if its individual components are all scalar Gaussian random variables. A key feature of the vector case is that these multiple components may be correlated, implying that their values cannot vary independently. If, on the one hand, y_i and y_j are independent (uncorrelated) components, knowing the value of y_i in a trial gives us no information about the value of y_j. On the other hand, if y_i and y_j are highly correlated, then knowing y_i gives us a great deal of information[4] about y_j. The correlations between all these components are usually encoded in a symmetric[5] matrix R called the correlation matrix, such that $r_{ij} = \mathbf{corr}(y_i, y_j)$. A scaled version of the correlation matrix is the covariance matrix Σ. The mean μ is now a vector, and along with the matrix Σ, completely specifies the probability distribution, which is given by

$$p(y) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right). \tag{6}$$

We can clearly see the similarity to the scalar case.

As we already mentioned, knowing some components of y gives us information about the others. This is called a conditional distribution: given the value of a set of random variables u, what is the 'updated' probability of another set of correlated random variables v? This is denoted by $p(v \mid u)$, pronounced 'the distribution of v given u'. It so happens that if we are given the values of some components of a Gaussian random variable y, the conditional distribution of the remaining components is also a Gaussian, but the mean vector and covariance matrix are now updated because of this knowledge. Let y_1 be a subset of components of y whose value b we are given. Let y_2 denote the remaining components. Partition the mean vector and the covariance matrix as

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}. \tag{7}$$

The conditional distribution is completely specified by the conditional mean vector and conditional covariance matrix.

$$\mathbf{E}(y_2 \mid y_1 = b) = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(b - \mu_1), \tag{8}$$

$$\mathbf{cov}(y_2 \mid y_1 = b) = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}. \tag{9}$$

Note, the conditional mean in Eq. 8 is the same size as μ_2, but its value is different because of the conditioning. Similarly, the conditional covariance in Eq. 9 is the same size as Σ_{22} but differs in value.

[4] If the correlation coefficient between two quantities is unity, knowing one will completely determine the other.

[5] In fact, it's guaranteed to be positive semidefinite.

3.2 Gaussian Processes

A stochastic process is similar to a deterministic function taking inputs x to outputs y, with the difference being that instead of a deterministic output, we have a conditional distribution $p(y \mid x)$ that determines the stochastic output y as a 'function' of the input x. This means that if we repeatedly give this process the same input x, it will return different output values y on each trial, but these outputs will be drawn from $p(y \mid x)$. The quantity $p(y \mid x)$ is often called 'the likelihood'. A Gaussian Process (GP) is a stochastic process such that any finite set of outputs is a multivariate Gaussian random vector. Thus, for any such finite subset, Eqs. 8 and 9 apply directly, if we could only define the mean vector μ and the covariance matrix Σ. One way to do this is as follows. Without regard to x, we propose that the mean of $p(y \mid x)$ is the *scalar* μ. That means that, regardless of x, the outputs returned are around the value μ. Thus, the mean vector for a set of outputs is $\mu\mathbf{1}$, where $\mathbf{1}$ is the vector of ones. Next, we propose that, regardless of x, the variance of a *single* output y is σ_0^2. Finally, since we're dealing with a vector, we need to specify covariances in addition to just variance. We do this by specifying how outputs are correlated depending on the distance of the corresponding inputs. For instance,

$$\mathbf{corr}(y_i, y_j) = r(d(x_i, x_j)), \tag{10}$$

where r is some function of the distance d between the associated inputs. This is where we plug in our weak assumption that nearby inputs have highly correlated outputs. If r is a monotonically decreasing function, the correlation between outputs monotonically decreases as distance between their inputs increases.

Almost immediately, things start clicking into place: if we are given a set of input locations, Eq. 10 determines the correlation matrix for the outputs at those locations. Multiplied by the variance σ_0^2, this specifies their covariance matrix Σ. The mean vector is just $\mu\mathbf{1}$. These two quantities are specified *before* we see any explicit values of x and y, and are therefore called priors, or, for those who want to show off their Latin, they define the *a priori* distribution.

Now, if we do know some subset y_1 of these outputs, Eq. 8 and 9 immediately yield a conditional distribution over the remaining (unknown) outputs, as described in the previous section. In that case, an unknown output close to a known output will have small conditional variance and a mean similar to that known output, but an output far from all known outputs will have mean close to μ and variance of almost σ_0^2. These updated quantities define $p(y \mid x, y_1)$ at the locations of the unknown outputs. This is called a posterior, or in Latin, an *a posteriori* distribution.

3.3 Gaussian Process Regression

In this section, we apply Gaussian Process concepts to the nonparametric regression problem. In GPR, we have a true system g, which, for the purposes of this document, we assume to be deterministic. Our model $\mathcal{G}$, on the other hand, is a stochastic process, a GP. We specify the priors, which comprise the mean μ, variance σ_0^2 and the correlation function r. The exact form of the correlation function in Eq. 10 encodes more information about g. For example, the squared exponential correlation function, given by

$$r(x_i, x_j) = \exp\left(-\sum_{k=1}^{n}\left[\frac{x_i^{(k)} - x_j^{(k)}}{\tau_k}\right]^2\right) \tag{11}$$

assumes a system g that is smooth (infinitely differentiable). Other functional forms result in various similar assumptions (see Rasmussen and Williams, 2006, Chap. 4).

In order to make a prediction at a given location x_0, we assume that the m outputs in the training data $\mathcal{D}$ and this one unknown output y_0 form a joint Gaussian random vector of size $m+1$. The prior mean is just $\mu\mathbf{1}$, and the prior covariance Σ is obtained by just plugging the various input locations into r. More specifically,

$$\Sigma_{ij} = \sigma_0^2 r(x^{(i)}, x^{(j)}),\ i = 1, \ldots, m+1,\ j = 1, \ldots, m+1. \tag{12}$$

Now, as described previously, we can partition Σ into blocks corresponding to the known and unknown outputs. For instance, Σ_{21} is a $1 \times m$ block corresponding to the covariance of the unknown output y_0 with each of the known outputs. Σ_{11} is the covariance matrix of all known outputs. We can now compute the posterior by simply plugging these quantities into Eqs. 8 and 9. This gives us our predictor; the predicted value is the posterior mean, and the uncertainty in the prediction is the posterior variance. We have arrived naturally at the Kriging formula, instead of following the oft-seen *deus ex machina* approach. In fact, this interpretation using GPs yields many insights, and thus enables many more applications, many described by Rasmussen and Williams (2006).

Making simultaneous predictions at multiple locations is trivial. The formulæ remain unchanged; only the sizes of the blocks change. Also note that such simultaneous predictions will be *a posteriori* correlated not only with the known outputs, but with each other as well! Shown below is the posterior distribution that defines the predictor, in terms of unknown and known quantities denoted by the subscripts u and k respectively. When convenient, we refer to all the known quantities as the data $\mathcal{D}$

$$\begin{aligned} \hat{y} = \mathbf{E}(y_u \mid \mathcal{D}) &= [\mathbf{cov}(y_u, y_k)][\mathbf{cov}(y_k, y_k)]^{-1} y_k + \mu\mathbf{1}, \\ &= \Sigma_{uk}\Sigma_{kk}^{-1} y_k + \mu\mathbf{1}, \qquad (13) \\ \mathbf{cov}(y_u \mid \mathcal{D}) &= [\mathbf{cov}(y_u, y_k)][\mathbf{cov}(y_k, y_k)]^{-1}[\mathbf{cov}(y_k, y_u)], \\ &= \Sigma_{uk}\Sigma_{kk}^{-1}\Sigma_{ku}. \qquad (14) \end{aligned}$$

Note that, for a prediction at a single location, the Σ_{uk} is a $1 \times m$ matrix (call this the vector ϕ_0) which depends both on the prediction point and all of the data $\mathcal{D}$. Also note that $\Sigma_{kk}^{-1} y_k$ is a vector α, and the predictor can now be written as $\alpha^T \phi_0$, which quite resembles the parametric predictor in Sec. 2.2. The vector α can be precomputed once we have the data, and is called the vector of regressor weights; it is the closest analog of the parameter vector θ in parametric regression. Note, however, that the predictor still needs $\phi_0 = \Sigma_{uk}$, which depends on all the data.

3.4 GPR Training

The keen reader will have noticed that, while we have described GPR, we have not said anything about how to set the prior mean and covariance, or the length-scale parameters τ_k in Eq. 11. These 'hyperparameters'[6] encapsulate all our knowledge of the system; τ_k determine how quickly correlations die out in different dimensions, σ_0^2 determines how much uncertainty is present very far from the data, and μ determines the predictor value very far from all existing data.

[6]The vector α is deemed analogous to the parameters.

If we are given these hyperparameters, we have no need for a training algorithm other than constructing the relevant covariance matrices. Unfortunately, we are seldom given these hyperprameters, and optimizing them is not as simple as in the case of parametric regression. At the same time, it's not very difficult to understand, and an excellent exposition is in Chap. 5 of Rasmussen and Williams (2006).

4 Discussion

We now summarize our overview of regression. The goal of regression is the successful prediction of outputs of a continuous input-output system. There are two components to regression: the training algorithm, which makes predictors from a set of data, and the predictor itself, which makes predictions. First, we motivated parametric regression using the simple and familiar example of a best-fit line. In parametric regression, the predictor is fully defined by a small set of parameters. Once these parameters have been computed by the training algorithm, the data can be discarded. The main of advantage of parametric regression is simplicity, but proposing a parametric form is not always easy.

We can then take recourse to nonparametric regression, which enables us to incorporate much weaker assumptions, but at the cost of simplicity. Nonparametric regressors also are also parametrized, but these parameters don't fully define the predictor; we also need the training data to make predictions. We introduced a particularly powerful type of nonparametric regression, called Gaussian Process Regression, which embodies several other kinds of nonparametric regression. We divided the discussion into three sections: first, we introduced multivariate Gaussian random variables, the cornerstone upon which the technique is built. Then, we introduced the basic assumptions of Gaussian Processes, which is that any finite set of outputs of the system form a multivariate Gaussian random variable. Next, we defined the parameters of this random variable, by proposing a prior mean, variance, and a correlation function that defines a covariance matrix. Finally, we used a previously-introduced result to compute *a posteriori* conditional distributions for the outputs at one or more prediction locations. We did not describe the optimization of the (hyper)parameters for GPR, but left this as a task for interested readers.

As mentioned before, this is hardly the end of the story. Implications, interpretations, and applications for this technique abound. The topic is often presented in an arcane manner, almost as if there were some underlying black magic driving the process. Hopefully, this document has served as an introduction to the general topic of regression, as well as an 'ice-breaker' to a logical, albeit complex and powerful technique of Gaussian Process Regression.

References

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL `http://www.gaussianprocess.org/gpml/chapters/`.

D. G. Krige. A statistical approach to some mine valuations and al lied problems at the witwater- a statistical approach to some mine valuations and allied problems at the witwatersrand. Master's thesis, University of Witwatersrand, 1951.