ECE-GY 6143, Spring 2020

Regression

Suppose we have observed data-label pairs $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ and there is some (unknown) functional relationship between x_i and y_i. We will assume the label y_i can be any real number. The problem of *regression* is to discover a function f such that $y_i \approx f(x_i)$. (Later, we will study a special case known as *binary classification*, where the labels are assumed to be ± 1). The hope is that once such a function f is discovered, then for a *new*, so-far-unseen data point x, we can simply apply f to x to predict its label.

Examples include:

- predicting stock prices (y) from econometric data such as quarterly revenue (x)
- predicting auto mileage (y) from vehicle features such as weight (x). We have introduced this as a class lab exercise.
- forecasting Uber passenger demand (y) from population density (x) for a given city block
- $\dots$ and many others.

Thinking about it a little bit, we quickly realize that this, of course, is an ill-posed problem — there is an infinity of such functions f that can be constructed! (Can you reason about why this is the case?) To make the problem well-defined, we need to restrict the space of *possible* functions from which f can arise. This is called the *hypothesis class*.

Simple linear regression

As our simplest hypothesis class, we assume a *linear* model on the function (i.e., the label y_i is a linear function of the data x_i).

Why linearity? For starters, linear models are simple to understand and interpret and intuitively explain to someone else. ("If I double some quantity, some other quantity doubles too..")

Linear models are also (relatively) easy from a computation standpoint. We will analytically derive below examples of linear models for a given training dataset in closed form.

Finally, if we recall the concept of *Taylor series expansions*, functions that arise in natural applications can often be locally expressed as linear functions. We will make this idea precise in later lectures.

Univariate regression

Let us start super simple and assume that both data and labels are scalars (such as the horsepower-mpg example shown in class.) We have observed data-label pairs $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. We need to figure out a linear model f that maps x to y.

As explained in Lecture 1, many ML problems involve a three-step solution. First, we need a *representation* for the data model. Then, we need a *measure of goodness* that tells us how well our model is serving us. Lastly, we need an *algorithm* that produces the best-possible model.

The first step is the representation. Since we have assumed a linear model, mathematically this translates to:

$$y = w_0 + w_1 x.$$

The second step in solving regression problems is to define a suitable *loss function* with respect to the model parameters, and then discover the model that minimizes the loss. A model with zero loss can perfectly predict the training data; a model with high loss is considered to be poor.

The most common loss function is the *mean-squared-error* loss, or the MSE:

$$\begin{aligned} MSE &= \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}[y_i - (w_0 + w_1 x_i)]^2. \end{aligned}$$

Geometrically speaking, one can interpret w_1 to be the "slope" and w_0 to be the "intercept".

Note that there is nothing sacred about the MSE and there can be other loss-functions too! Later, we will encounter something called the *logistic* loss, which will lead to a technique known as logistic regression. But for now let us remain simple.

The third step is an algorithm to produce the *best* model. Since we are dealing with scalars, this answer can be obtained using ordinary calculus. Viewing MSE as a function of w_0 and w_1, the minimum MSE is attained when the (partial) derivative of the MSE with respect to w_0, as well as with respect to w_1, equal zero, giving us the equations:

$$\frac{\partial MSE}{\partial w_0} = 0, \frac{\partial MSE}{\partial w_1} = 0.$$

From the first equations, we get the optimal value of w_0 by calculating:

$$\frac{1}{n}\sum_{i=1}^{n}[y_i - (w_0 + w_1 x_i)] = 0 \implies w_0^* = \bar{y} - w_1^* \bar{x}$$

where $\bar{x}, \bar{y}$ represent the means of x and y respectively. Similarly, from the second equation, we get:

$$\frac{1}{n}\sum_{i=1}^{n}[x_i y_i - x_i w_0 - w_1 x_i^2] = 0.$$

Plugging in the value of w_0 and solving for w_1 (with some algebraic simplification), we get:

$$w_1^* = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i - \bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - \bar{x}^2}.$$

One might be able to recognize the terms on the right hand side. The denominator is simply the *variance* of x (call it σ_x^2) while the numerator is the *cross covariance* between x and y (call it σ_{xy}). It is somewhat natural to expect this kind of behavior: the slope coefficient w_1 being the ratio of σ_{xy} to σ_x^2.

There we have it, then. For any new data point x, we can compute its predicted value y by simply writing out $\hat{y} = w_0^* + w_1^* x$, where w_0^*, w_1^* have the closed form expressions as above.

The minimum MSE can also be computed this way in closed form by plugging in these values of w_0^*, w_1^*. We get:

$$MSE^* = \frac{1}{n}\sum_{i=1}^{n}[y_i - (w_0^* + w_1^* x_i)]^2,$$

and with some tedious but simple algebra, we can derive the following expression:

$$MSE^* = \sigma_y^2 - \frac{\sigma_{xy}^2}{\sigma_x^2}.$$

where σ_y^2 is the variance of y. Rewriting slightly, we get:

$$\frac{MSE}{\sigma_y^2} = 1 - \frac{\sigma_{xy}^2}{\sigma_x^2 \sigma_y^2}.$$

The ratio on the left hand side is called the *fraction of unexplained variance*. For a perfectly interpolating model, this would be equal to zero. The ratio on the right hand side is called the *coefficient of determination*. Statisticians like to call it R^2 ('r-squared'). An R^2 close to 1 implies a well-fitted model, while an R^2 of close to zero implies the opposite.

Multivariate regression

The above approach can be generalized to the case of high-dimensional (vector-valued) data. In this case, the functional form is given by:

$$y_i \approx \langle w, x_i \rangle, \ i = 1, \ldots, n.$$

where $w \in \mathbb{R}^d$ is a vector containing the *regression coefficients*. We need to figure out w from the data. Linear models are simple, powerful, and widely used.

Solving linear regression

For linear models, this reduces to $L(w) = \frac{1}{2}(y - \langle w, x \rangle^2$. For conciseness, we write this as:

$$L(w) = \frac{1}{2}\|y - Xw\|^2$$

the norm above denotes the Euclidean norm, $y = (y_1, \ldots, y_n)^T$ is an $n \times 1$ vector containing the y's and $X = (x_1^T; \ldots; x_n^T)$ is an $n \times d$ matrix containing the x's, sometimes called the "design matrix".

The gradient of $L(W)$ is given by:

$$\nabla L(w) = -X^T(y - Xw).$$

The above function $L(w)$ is a convex (in fact, quadratic) function of w. The value of w that minimizes this (say, w^*) can be obtained by setting the gradient of $L(w)$ to zero and solving for w:

$$\begin{aligned} \nabla L(w) &= 0, \\ -X^T(y - Xw) &= 0, \\ X^T X w &= X^T y, \quad \text{or} \\ w &= (X^T X)^{-1} X^T y. \end{aligned}$$

The above represents a set of d linear equations in d variables, and are called the *normal equations*. If $X^T X$ is invertible (i.e., it is full-rank) then the solution to this set of equations is given by:

$$w^* = \left(X^T X\right)^{-1} X^T y.$$

If $n \geq d$ then one can generally (but not always) expect it to be full rank; if $n < d$, this is not the case and the problem is under-determined.

Computing $X^T X$ takes $O(dn^2)$ time, and inverting it takes $O(d^3)$ time. So, in the worst case (assuming $n > d$), we have a running time of $O(nd^2)$, which can be problematic for large n and d.