Regression

based on a document by Andrew Ng

May 5, 2014

1 Linear regression

Lets consider a supervised learning example where data points are houses with 2 features (X^1= living area; X^2 = number of bedrooms) and labels are the prices.

Living area (ft^2)	#bedrooms	price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
...	...	...

Each datapoint is thus $(\mathbf{x}_t, y_t)$, where $\mathbf{x}_t = (x_t^1, x_t^2)$ are the living area and the number of bedrooms respectively, and y_t is the price. The main problem addressed by *regression* is: Can we predict the price (or label) y_t from input features $\mathbf{x}_t$? Today we study *linear regression*, that is if we can predict the price from the features using a linear regressor

$$h_w(\mathbf{x}) = w^0 + w^1 x^1 + w^2 x^2$$

where $w = (w^0, w^1, w^2)$ are the parameters of the regression function. Within the class of linear functions (regressors) our task shall be to find the best parameters w. When there is no risk of confusion, we will drop w from the h notation, and we will assume a dummy feature $x^0 = 1$ for all datapoints such that we can write

$$h(\mathbf{x}) = \sum_{d=0}^{D} w^d x^d$$

where d iterates through input features 1,2,... ,D (in our example $D = 2$).

What do we mean by the best regression fit? For today, we will measure the error (or cost) of the regression function by the *mean square error*

$$J(w) = \sum_t (h_w(\mathbf{x}_t) - y_t)^2$$

and we will naturally look for the w that minimizes the error function. The regresion obtained using the square error function J is called *least square regression*, or *least square fit*. We present two methods for minimizing J: a direct linear algebra solution next, and gradient descent optimization later.

2 Least mean square via normal equations

2.1 Matrix derivatives

Let $f : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$ a function that takes a matrix as input and outputs a real number. We define the derivative of f with respect to the matrix A as

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial a_{11}} & \dots & \frac{\partial f}{\partial a_{1n}} \\ \dots & \dots & \dots \\ \frac{\partial f}{\partial a_{m1}} & \dots & \frac{\partial f}{\partial a_{mn}} \end{bmatrix}$$

where a_{ij} is the element of A on row i and column j. For example, consider $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and the function $f(A) = \frac{3}{2}a_{11} + 5a_{12}^2 + a_{21}a_{22}$. Then

$$\nabla_A f(A) = \begin{bmatrix} \frac{3}{2} & 10a_{12} \\ a_{22} & a_{21} \end{bmatrix}$$

Today we will look at the *trace* function as f. The trace of a matrix A is the sum of the elements of the main diagonal

$$\mathrm{tr}(A) = \sum_i a_{ii}$$

The following are simple and well known properties of the trace:

$$\begin{aligned} \mathrm{tr}(AB) &= \mathrm{tr}(BA) \\ \mathrm{tr}(A) &= \mathrm{tr}(A^T) \\ \mathrm{tr}(A+B) &= \mathrm{tr}(A) + \mathrm{tr}(B) \\ \mathrm{tr}(xA) &= x\mathrm{tr}(A) \end{aligned}$$

The following properties of the trace matrix derivative are going to be useful for finding an exact regression solution:

$$\nabla_A \mathrm{tr}(AB) = B^T \tag{1}$$

$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T \tag{2}$$

$$\nabla_A \mathrm{tr}(ABA^TC) = CAB + C^TAB^T \tag{3}$$

Combining the second and third statements above we get

$$\nabla_{A^T} \mathrm{tr}(ABA^TC) = B^TA^TC^T + BA^TC \tag{4}$$

2.2 An exact solution for regression using linear algebra

Given the datapoints $(\mathbf{x}_t, y_t)$ for $t = 1, 2, ..., m$, with D input dimensions (features), we shall look at them in a matrix form

$$X = \begin{bmatrix} x_1^1 & \dots & x_1^D \\ \dots & \dots & \dots \\ x_m^1 & \dots & x_m^D \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix}$$

Then the error array asociated with our regressor $h_w(\mathbf{x}) = \sum_d w^d x^d$ is

$$E = \begin{bmatrix} h_w(\mathbf{x}_1) - y_1 \\ \dots \\ h_w(\mathbf{x}_m) - y_m \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 w \\ \dots \\ \mathbf{x}_m w \end{bmatrix} - \begin{bmatrix} y_1 \\ \dots \\ y_m \end{bmatrix} = Xw - Y$$

(we used $w = (w^0, w^1, ...w^d)^T$ as a vector column). We can now write the mean square error as

$$J(w) = \frac{1}{2}\sum_t (h_w(\mathbf{x}_t) - y_t)^2 = \frac{1}{2}E^T E = \frac{1}{2}(Xw - Y)^T(Xw - Y)$$

Then

$$\begin{aligned}
\nabla_w J(w) &= \nabla_w \frac{1}{2}E^T E = \nabla_w \frac{1}{2}(Xw - Y)^T(Xw - Y) \\
&= \frac{1}{2}\nabla_w (w^T X^T X w - w^T X^T Y - Y^T X w + Y^T Y) \\
&= \frac{1}{2}\nabla_w \mathrm{tr}(w^T X^T X w - w^T X^T Y - Y^T X w + Y^T Y) \\
&= \frac{1}{2}\nabla_w (\mathrm{tr}(w^T X^T X w) - 2\mathrm{tr}(Y^T X w)) \\
&= \frac{1}{2}(X^T X w + X^T X w - 2X^T Y) \\
&= X^T X w - X^T Y
\end{aligned}$$

because: in the third step we have $\mathrm{tr}(x) = x$, in the four step we have $\mathrm{tr}(A) = \mathrm{tr}(A^T)$, and in the fifth step we are using equation 4 with $A^T = w, B = B^T = X^T X, C = I$.

Since we are trying to minimize J, a convex function, a sure way to find w that minimizes J is to set its derivative to zero. In doing so we obtain

$$X^T X w = X^T Y \text{ or } w = (X^T X)^{-1} X^T Y$$

This is the exact w that minimizes the mean square error.

3 Least mean square probabilistic interpretation

Why mean square error? We show now that the objective J used is a direct consequence of a very common assumption over the data. Lets look at the errors

$$\epsilon_t = h(\mathbf{x}_t) - y_t$$

and lets make the assumption that they are IID according to a gaussian (normal) distribution of mean $\mu = 0$ and variance σ^2. That we write $\epsilon\ \mathcal{N}(0, \sigma^2)$ or

$$p(\epsilon) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$$

which implies

$$p(y|x; w) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(w\mathbf{x} - y)^2}{2\sigma^2}\right)$$

Note that above w is a parameter (array) and not a random variable. Given the input X, what is the probability of Y given the parameters w ? Equivalently, this is the *likelihood* that w is the correct parameter for the model

$$L(w) = L(w; X, Y) = p(Y|X; w)$$

Using the IID assumption the likelihood becomes

$$\begin{aligned}
L(w) &= \prod_t p(y_t|\mathbf{x}_t); w \\
&= \prod_t \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(w\mathbf{x}_t - y_t)^2}{2\sigma^2}\right)
\end{aligned}$$

since we have now a probabilistic model over the data, a common way to determine the best parameters is to use *maximum likelihood*; in other words find w that realizes the maximum L. Instead of maximizing L we shall maximize the *log likelihood* $\log L(w)$ because it simplifies the math (and produces the same "best" w)

$$\begin{aligned} l(w) &= \log L(w) \\ &= \log \prod_t \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(w\mathbf{x}_t - y_t)^2}{2\sigma^2}\right) \\ &= \sum_t \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(w\mathbf{x}_t - y_t)^2}{2\sigma^2}\right) \\ &= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_t (h_w(\mathbf{x}_t) - y_t)^2 \end{aligned}$$

Hence, the maximizing the likelihood L produces the same w as minimizing the mean square error (since the front term does not depend on w). That is to say that, if we believe the errors to be IID normally, then the maximum likelihood is obtained for the parameters w that minimizes the mean square error.

4 Classification and logistic regression

In classification, the labels y are not numeric values (like prices), but instead *class labels*. For today, lets assume that we have two classes denoted by 0 and 1; we call this *binary classification* and we write $y \in \{0, 1\}$.

4.1 Logistic transformation

We could, in principle try to run the linear regression we just studied, without making use of the fact that $y \in \{0, 1\}$. (Essentially assume y are simply real numbers). There are several problem with this approach: first, the regression assumes the data supports a linear fit, which might not be true anymore for classification problems; second, our regressor $h(\mathbf{x})$ will take lots of undesirable values (like the ones far outside the interval [0,1]).

To make an explicit mapping between the real valued regressor h and the set {0,1}, we would like a function that preserves differentiability and has a easy interpretable meaning. We choose the *logistic function*, also called *sigmoid*

$$g(z) = \frac{1}{1 + e^{-z}}$$

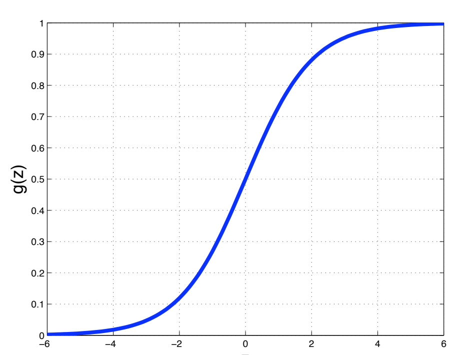

Figure 1: Logistic function

Note that g acts like a indicator for {0,1}, but it is much more sensitive than a linear function. We can make it even more sensitive by, for example, doubling the input z before applying the function. Lets state the derivative of g, since we are going to use it later on

$$\begin{aligned} g'(z) &= \frac{\partial g(z)}{\partial z} \\ &= \frac{1}{(1+e^{-z})^2} e^{-z} \\ &= \frac{1}{1+e^{-z}} \left(1 - \frac{1}{1+e^{-z}}\right) \\ &= g(z)(1-g(z)) \end{aligned}$$

4.2 Logistic regression

We apply g to the linear regression function to obtain a *logistic regression*. Our new hyphothesis (or predictor, or regressor) becomes

$$h_w(\mathbf{x}) = g(w\mathbf{x}) = \frac{1}{1+e^{-w\mathbf{x}}} = \frac{1}{1+e^{-\sum_d w^d x^d}}$$