# Linear regression

Karim Pichara B.
Computer Science Department
Pontificia Universidad Católica de Chile

March 13, 2017

Let $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be a set of $N$ training examples (observations), where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$. The main idea is to learn a model (using the set of training examples) that predicts the value of $y$ for any instance $x$.

Probabilistic interpretation of regression model:

$y = f(x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$

Where:

$f(x_i) = \hat{\beta}_0 + \sum_{j=1}^{d} \hat{\beta}_j x_{ij}$

is easy to note that $y_i \sim \mathcal{N}(f(x_i), \sigma^2) = \mathcal{N}(\hat{\beta}_0 + \sum_{j=1}^{d} x_{ij}\hat{\beta}_j, \sigma^2)$.

Note that $Y =$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

$X =$

$$\begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nd} \end{bmatrix}$$

$W =$

$$\begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_d \end{bmatrix}$$

**MLE Approach**

We can solve the regression problem searching for the parameters that maximize the probability of data given the model (likelihood). This is known as the Maximum Likelihood Estimator (MLE). We can express the likelihood as:

$$\begin{aligned} P(x_1, \ldots, x_N|W) &= \prod_{i=1}^{N} P(x_i|W) && (1) \\ &= \prod_{i=1}^{N} \mathcal{N}(\hat{\beta}_0 + \sum_{j=1}^{d} \hat{\beta}_j x_{ij}, \sigma^2) && (2) \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{y_i - \hat{\beta_0} - \sum_{j=1}^{d} x_{ij}\hat{\beta}_j}{\sigma})^2} && (3) \end{aligned}$$

For simplicity we maximize the log likelihood instead of the likelihood:

To find the parameters that optimize the log likelihood we calculate the derivatives of RSS with respect to the parameters and set it to zero. For simplicity we will minimize the negative log likelihood (NLL):

$$
\begin{aligned}
RSS &= (Y - XW)^T (Y - XW) \qquad (4)\\
&= Y^T Y - Y^T XW - W^T X^T Y + W^T X^T XW \\
&= Y^T Y - 2Y^T XW + W^T X^T XW \qquad (5)
\end{aligned}
$$

$$
\begin{aligned}
\frac{dRSS}{dW} &= (-2Y^T X)^T + 2X^T XW \qquad (6)\\
&= -2X^T (Y - XW)
\end{aligned}
$$

Assuming that $X$ is not singular (variables describing $x_i$ are not strongly correlated)

$$
\begin{aligned}
-2X^T (Y - XW) &= 0 \qquad (7)\\
X^T Y &= X^T XW \\
W &= (X^T X)^{-1} X^T Y
\end{aligned}
$$

In other words, we just need to build the design matrix $X$ with the locations or positions $(x_i)$ of the training observations $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ and the vector $Y$ with the respective observed values $y_i$ in $S$. Figure 1 shows an example of a simple linear regression model:

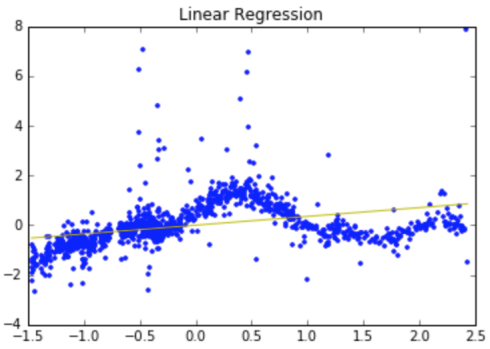


Figure 1: Simple linear regression applied to real data

An alternative way to train a non linear model is to use a set of $B$ functions, each one constructed from a set of $B$ data points (basis) $\{\mu_1, \mu_2, \ldots, \mu_B\}$. In this case the basis functions are simply a Gaussian kernel:

$$k(x_i, \mu_j) = e^{-\frac{1}{2}(\frac{x_i - \mu_j}{\sigma})^2} \quad \forall i \in [1 \ldots N],\ j \in [1 \ldots B].$$

Where $\sigma$ is an input parameter that indicates the width of the Gaussian distribution centered on each base. The wider the Gaussian in the base, the bigger the area that is influenced from the base, and the smother the regression model.

Now the function to transform data is defined as $\Phi(x_i) = (1, k(x_i, \mu_1), k(x_i, \mu_2), \ldots, k(x_i, \mu_B))$. The design matrix is transformed into:

$$X = \begin{bmatrix} 1 & k(x_{11}, \mu_1) & k(x_{11}, \mu_2) & \cdots & k(x_{11}, \mu_B) \\ 1 & k(x_{21}, \mu_1) & k(x_{21}, \mu_2) & \cdots & k(x_{21}, \mu_B) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & k(x_{N1}, \mu_1) & k(x_{N1}, \mu_2) & \cdots & k(x_{N1}, \mu_B) \end{bmatrix}$$

We usually pick randomly the set of B points from the observations. Note that here is also assumed that each $x_i \in \mathbb{R}$. In general expanding the inputs $x_i$ to higher spaces makes sense only in cases where the inputs $x_i$ belong to a very low dimensional space. Figure 2 shows an example of a set of Gaussian basis functions. In figure 3 we can visualize the result of applying linear regression with Gaussian basis functions to the same data used above.

Another way to learn a non linear model is to expand the values of the $x_i$ using a polynomial basis function $\Phi(x_i, p) = (1, x_i^2, x_i^3, \ldots, x_i^p)$. In this case we assume that $d = 1$ ($x_i \in \mathbb{R}\ \forall i \in [1 \ldots N]$ ). Applying the function $\Phi$ to the inputs $x_i$, we obtain the following design matrix:

$$X = \begin{bmatrix} 1 & x_{11} & x_{11}^2 & \cdots & x_{11}^p \\ 1 & x_{21} & x_{21}^2 & \cdots & x_{21}^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N1}^2 & \cdots & x_{N1}^p \end{bmatrix}$$

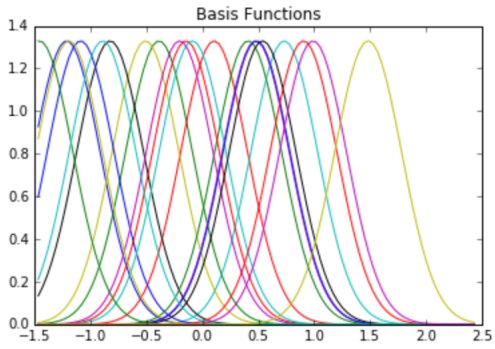


Figure 2: Set of basis defined as Gaussian functions

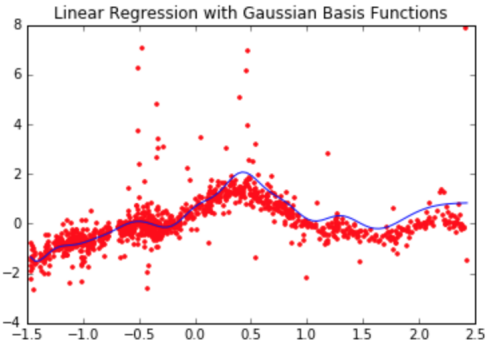


Figure 3: Linear regression with Gaussian basis functions applied to real data.

I strongly suggest you to write Python code to implement the regression models described above. In the next class we will study the MAP approach for linear regression and Bayesian Linear Regression, we will implement both with Python as well.