*April 30, 2018*

# CS 361: Probability & Statistics

Regression

# Regression

# Regression

Like with classification, regression proceeds from a set of labeled training data

We have $N$ pairs of $d$-dimensional points plus their “label", except in this case the label is not a class label but a numerical value that the data item takes on at the corresponding point

For example, the data could be the square footage of a house and the “label” is the price that the house sold for recently

Our goal, then, would be to learn a mapping from $d$-dimensional points to these numerical values

# Regression

Assuming that we have *N* pairs (**x_i**, y_i), we think of the y_i as being the value of some function evaluated at **x_i**, with some random component added

We refer to the **x_i** as **explanatory variables** and the y_i as the **dependent variable**.

We want to use the *N* items we have—the training data—to build a model for the dependence between y and **x**. In order to predict values of y on data we have never seen before—test data.

# Linear regression

A good simple model is to assume that the dependent variable is obtained by evaluating a linear function of the explanatory variables, then adding a zero-mean normal random variable

$$y = \mathbf{x}^T \beta + \xi$$

Where $\xi$ is a zero-mean normal random variable of unknown variance

$\beta$ is a random vector of weights which we must estimate or learn from the training data

Thus, in this model $y$ should be thought of as a random variable and the prediction of a $y$ from an $\mathbf{x}$ should be viewed probabilistically

# Training the model

So the question is how do we learn the weight vector beta? The answer is we will look at the problem probabilistically and come up with a good estimate

Let's look at our model

$$y = \mathbf{x}^T \beta + \xi$$

In training, the **x** and *y* are from the training data, Beta is what we are hoping to learn, and Xi is a zero-mean normal random variable of unknown variance

It makes sense to think of the value of *y* as being the outcome of a random process since there is a random variable on the right hand side, i.e. *y* is a random variable and we may be able to get a good value of Beta by considering the quantity $p(y \mid \mathbf{x})$

# Training the model

$$y = \mathbf{x}^T\beta + \xi$$

For a given **x_i**, y_i, how can we express p(y_i | **x_i**)?

Well we have a zero-mean normal random variable to which we are adding a constant, so for training example *i* we can write p(y_i | **x_i**) as a normal random variable with mean $\mathbf{x_i}^T\beta$

Recall the form of the normal density $p(x) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$

Which means we have the following density

$$p(y_i|\mathbf{x_i},\beta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\mathbf{x_i}^T\beta)^2}{2\sigma^2}\right)$$

# Training the model

$$p(y_i|\mathbf{x_i},\beta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\mathbf{x_i}^T\beta)^2}{2\sigma^2}\right)$$

We want to find a value for Beta that maximizes the probability of all of the data, i.e. a max-likelihood value of Beta

$$\mathcal{L}(\beta) = p(\text{data}|\beta) = \prod_i\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\mathbf{x_i}^T\beta)^2}{2\sigma^2}\right)\right)$$

It will be easier to use log-likelihood as we have done in the past, which will give

$$\frac{1}{2\sigma^2}\sum_i -(y_i-\mathbf{x}_i^T\beta)^2 + \text{some term not depending on } \beta$$

# Training the model

$$\frac{1}{2\sigma^2}\sum_i -(y_i - \mathbf{x}_i^T\beta)^2 + \text{some term not depending on } \beta$$

Dropping the constant, we want to maximize

maximize $\sum_i -(y_i - \mathbf{x_i}^T\beta)^2$

Notice that maximizing the above quantity is the same as minimizing the total squared error of our predictions, which had we proceeded to solve the problem from the point of view of minimizing some loss may have been a natural choice

minimize $\sum_i (y_i - \mathbf{x_i}^T\beta)^2$

# Training the model

It is convenient to write things out with matrices and vectors for our ultimate solution, so write

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

and

$$X = \begin{pmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_N}^T \end{pmatrix}$$

Then the function we wish to optimize becomes

$$(y - X\beta)^T (y - X\beta)$$

# Training the model

$$(y - X\beta)^T(y - X\beta)$$

We would then expand this, take the derivative with respect to Beta and set that equal to 0. Pages of unenlightening matrix calculus would tell us that we are interested in finding a Beta that solves

$$X^T X\beta - X^T y = 0$$

Or

$$\beta = (X^T X)^{-1} X^T y$$

# Training the model

So our algorithm for finding a good weight vector Beta for a given training data set is to covert the **x**'s of the training data to a matrix and the y's to a vector

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \quad \text{and} \quad X = \begin{pmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_N}^T \end{pmatrix}$$

And then solve $\beta = (X^T X)^{-1} X^T y$

# Making predictions

When using this model to predict a value y for some new **x**, we cannot predict what value $\xi$ will take, and so we will always predict that it takes the value of its mean, namely 0, or that for some new **x** our prediction is just

$$y = \mathbf{x}^T \beta$$

Looking at this prediction function, you might worry that our model can only produce lines going through the origin, but we can fix that since we have some control over how to model the explanatory variables, as the next example shows

# Example

Suppose we are trying to model the dependent variable with just a single explanatory variable, i.e. our training data is a set of 1-dimensional x's and their corresponding y's

If, prior to training, we create for each data item a 2-dimensional vector given by $\mathbf{x} = \begin{bmatrix} u & 1 \end{bmatrix}^T$ where $u$ was the original explanatory variable, then we learn the model, we will have learned a $\beta_1, \beta_2$ from the data and our prediction function will be

$$y = \beta_1 x + \beta_2$$

The line associated with this model doesn't necessarily go through the origin since the prediction for x = 0 is $\beta_2$

# Residuals

We don't expect that the line that we get by finding Beta will perfectly match our training data. Some of the predicted $y$'s in our training data will be different than the actual $y$'s if our data doesn't lie on a perfectly linear surface

The **residual** is the vector

$$\mathbf{e} = \mathbf{y} - X\beta$$

The mean squared error is the number we get by computing

$$m = \frac{\mathbf{e}^T\mathbf{e}}{N}$$

and it gives the average squared error of prediction on the training examples

# R squared and explained variance

Another way to measure the quality of our predictions is to look at the amount of variance in the original dependent variable and compare it to the amount of variance in our predictions

The *y_i* over our whole training data can itself be regarded as a dataset, where computing the mean and variance as well-defined. Likewise our $\mathbf{x_i}^T \beta$ can be treated as a dataset

A useful quantity compares the variance of each

$$R^2 = \frac{\text{var}[\mathbf{x_i}^T \beta]}{\text{var}[y_i]}$$

# R squared

$$R^2 = \frac{\text{var}[\mathbf{x_i}^T \beta]}{\text{var}[y_i]}$$

This quantity will achieve a maximum of 1 when our regression perfectly predicts the y_i and its minimum value would be 0 if our regression predicted a constant for every x_i

Good predictions result in a high value of R squared

# Applications

Prediction is one application. We might have a whole bunch of historical labeled data about some quantity of interest and wish to make predictions about in the future on data items where the label won't be available at the time fo the prediction

Another application of regression is to visualize trends in data and to compare how well data items follow this trend

# Example: trends

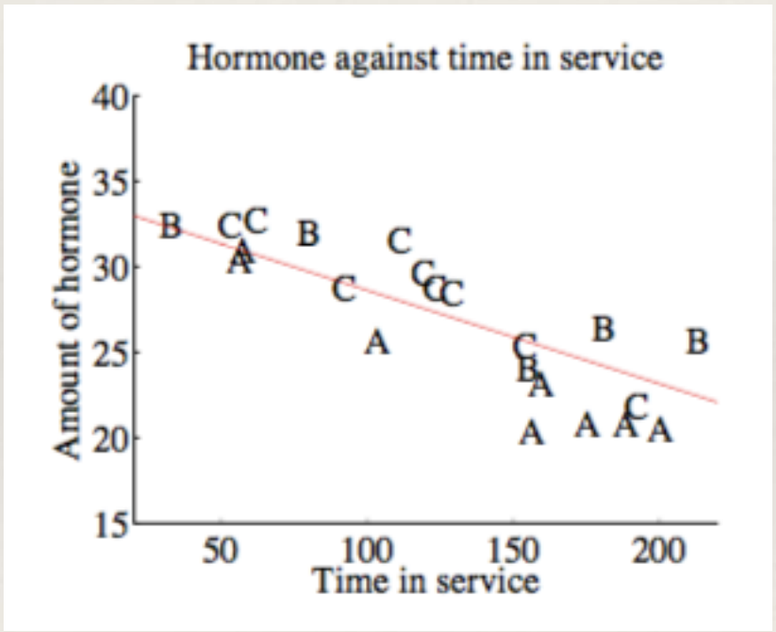
Suppose we have a dataset on a set of medical devices that remain in someone's body and release a hormone over time. The data are of the form (time_in_body, amount_of_hormone_in_device). Furthermore we have data from 3 different production lots of the device—A, B, and C—we are interested in doing quality control to see if there is any difference between the lots

We choose to model the amount of hormone remaining in the device as

$$a \text{ x (time in service)} + b$$

# Example: trends

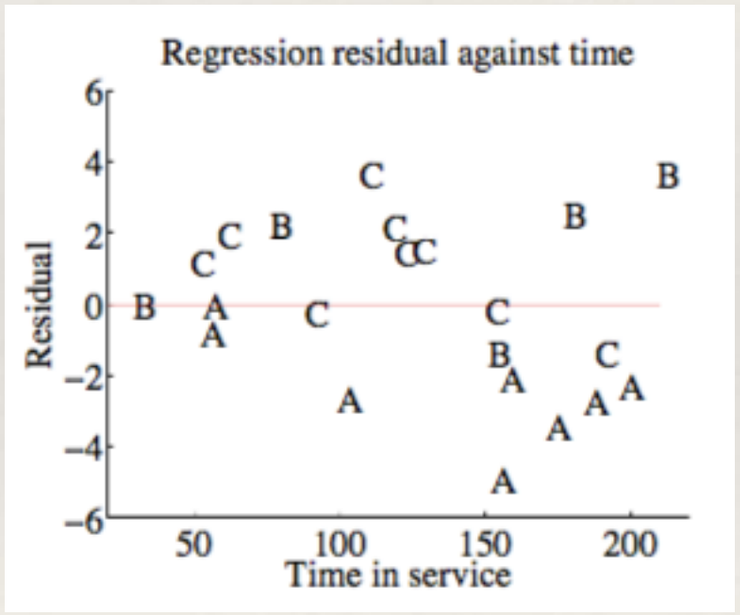
If we have some procedure for finding a good $a$ and $b$ we will have learned a line from the dataset which we can use to answer the question of differences between lots




It looks like the A's are consistently below the line in a way the other lots aren't

# Example: trends

Since it can be hard to evaluate distances by eye, we can also make a plot by subtracting the amount of hormone that's predicted by the model from the amount that was measured. This difference is called **the residual**




And here the trend with the As being different is even more apparent