Intuition

Linear regression is an excellent method for predicting results, using a hypothesis that is computed from training data. It can filter noisy measurements as well as interpolate or extrapolate unknown datapoints. And best: It's easy to implement. This flyer will show you how.

- Use a suitable hypothesis (aka fitting function) with an initial guess for the parameters.
- Compare results to actual training data using a cost function. (This is called supervised learning.)
- Try to minimize the cost function iteratively by computing gradients and updating parameters accordingly.
- Alternatively compute the normal equation (useful for smaller systems.)

Given the limited space of the flyer, there are several (more or less fundamental) topics that could not be covered in detail. Aspects that were beyond the scope of the flyer, are

- Correlation, variance, covariance in normal equation formalism.
- Feature scaling, mean normalization.
- Implementing gradient descend; learning rate.
- Strategies for good initial guesses of parameters.
- Debugging strategies when encountering problems.

Some of these aspects will be subject of the Numerical Optimization flyer. In addition a lot of literature can be found on the net that provides a deeper insight, e.g.

- Wikipedia [1]
- Machine Learning @ The Standford OpenClassroom [2]
- Calculating Regression Examples @ Wolfram Alpha [3]

The flyer can be distributed free of charge, as long as it is not altered in any way. If you have questions or comments contact the flyered science project.

fsadmin@flyered-science.org

[1] http://en.wikipedia.org/wiki/Linear_regression
[2] http://openclassroom.stanford.edu
[3] http://www.wolframalpha.com/examples/RegressionAnalysis.html

Numerical Solution (in Octave)

Using a numerical solver to compute Θ, one has to define cost function (3) and gradients (5), and pass it along with an inital guess of parameters to a suitable linear solver.

In Octave the cost function for linear regression (without renormalization) can be written as follows.

```
function [J,grad] = linearCF(theta,X,y)
  J = 1/2/m * sum((theta'*X - y)*...
      (theta'*X - y)');
  grad = 1/m*(theta'*X - y)*X';
endfunction
```

Here *theta* is a $(n+1)$-dimensional column vector, X is a $(n+1) \times m$ matrix and y is a m-dimensional row vector.

The logistic regression cost function (without renormalization) can be written accordingly, using the logistic function.

```
function res = lFun(z)
  res = 1 ./ (1 + exp(-z));
endfunction

function [J,grad] = logisticCF(theta,X,y)
  J = -1/m * (log(lFun(theta'*X)*y')+...
      log(1-lFun(theta'*X)) * (1-y)');
  grad = 1/m * (lFun(theta'*X)-y) * X';
endfunction
```

The optimal Θ can then be computed using e.g. fminunc.

```
opt = optimset('GradObj', 'on', ...
                'MaxIter', 100);
[theta,cost] = ...
fminunc(@(t)(linearCF(t,X,y)),init,opt);
```

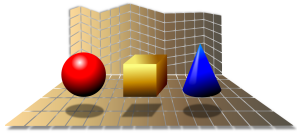
Here *init* is an initial guess for Θ, and opt the solver options. (For more information on commands consult Octave help.)

Flyered Science
Knowledge factsheets for the everyday scientist

Linear Regression

An introduction to representation and algorithm, with implementation in Octave

The Flyered Science Series provides condensed factsheets on scientific topics. Persons new to the subject will find the intuitive introduction with links to further reading quite useful, while the experienced user benefits from the structured representation, formulary and implementation examples. Fits into every pocket, serves perfectly as bookmark in your textbooks.



Theory

Problem

Given a set of m datapoints $(X^{(i)}, y^{(i)})$, where $y^{(i)}$ is the result of the ith measurement ($0 \leq i \leq m$), depending on n variables $X^{(i)} = (x_1^{(i)}, x_2^{(i)}, .., x_n^{(i)})$, find a linear function that best fits the data. The function can then be used to filter measurement noise or to determine unknown datapoints for predicting stock prices, weather, etc.

Some Conventions

The following definitions are used throughout the flyer, that are common when talking about linear regression:

m	number of datapoints $(X^{(i)}, y^{(i)})$
n	number of variables $(X^{(i)} \in \Re^n)$ (If $n = 1$ the regression is called univariate, otherwise multivariate.)

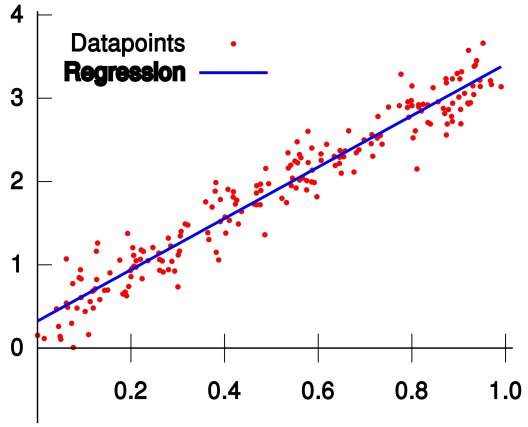


Figure 1: Example of a univariate linear regression function fitting the (red) datapoints $(X^{(i)} = (x_1^{(i)}))$

.

Results $y^{(i)}$ can be written as an m-dimensional vector y, while variables $x_j^{(i)}$ are well represented as an $n \times m$ dimensional matrix X. As we will see, this is often useful to simplify computation, e.g. when working with Octave.

Representation

We will now introduce the following definitions:

$h_\Theta(X)$	hypothesis, i.e. fitting function
Θ	fitting parameters $(\Theta \in \Re^{n+1})$

Ideally the hypothesis should match the results of the datapoints, i.e.

$$\begin{aligned} h_\Theta(X) &= \Theta_0 + \Theta_1 X_1 + ... + \Theta_n X_n \\ &= \Theta^T X \equiv y \end{aligned} \tag{1}$$

Here Θ^T is the transposed of Θ. We also introduced an extra dimension for X to take into account the offset Θ_0. X becomes an $(n+1) \times m$ matrix.

$$X = \begin{pmatrix} 1 & 1 & ... & 1 \\ x_1^{(1)} & x_1^{(2)} & ... & x_1^{(m)} \\ ... & ... & ... & ... \\ x_n^{(1)} & x_n^{(2)} & ... & x_n^{(m)} \end{pmatrix} \tag{2}$$

Polynomial Regression

The term linear regression refers to the linear system of equations in (1). It does not mean that one cannot fit a polynom to datapoints with linear regression.

Considering a univariate linear regression problem with variables $X = (1, x_1^{(i)})$ one can then construct a new variables matrix X_{new} taking into account higher orders of x_1, e.g. $X_{new} = (1, x_1^{(i)}, x_2^{(i)} = (x_1^{(i)})^2, x_3^{(i)} = (x_1^{(i)})^3)$, again applying the hypothesis of equation (1).

You can also use functions other than polynoms, like *sin*, *cos*, *exp* and so on, making the linear regression fitting very flexible.

Binary and continous results

So far we thought of the result vector y to hold continous numbers (real, complex). The same representation can also be used for binary results (i.e. 0 or 1, Yes or No ...). We then speak of logistic regression rather than linear regression. There are some differences regarding how to solve logistic regression problems, that will be covered in the following.

Cost function

The cost function $J(\Theta)$ calculates the difference between hypothesis and actual result. Often this is implemented as a least squares approach, summing the squares of differences.

$$\begin{aligned} J(\Theta) &= \frac{1}{2m} \sum_i^m (h_\Theta(X^{(i)}) - y^{(i)})^2 \\ &= \frac{1}{2m} (\Theta^T X - y)(\Theta^T X - y)^T \end{aligned} \tag{3}$$

The last line of equation (3) refers to linear regression problems. For logistic regression we choose a slightly different representation, using the logistic (or sigmoid) function.

$$h_\Theta(X_{(i)}) = \frac{1}{1 + \exp(-\Theta^T X^{(i)})} \tag{4}$$

The logistic function is 1 for large positiv values of the exponent, while 0 for large negative values. These are the two possible results we expect for logistic regression problems. The parameters Θ are chosen accordingly.

Normal Equation

When solving for the optimal set of parameters Θ one is looking for the minimum of the cost function. This implies the partial derivates to vanish:

$$\frac{\partial J(\Theta)}{\partial \Theta_j} = 0 \tag{5}$$

This condition is met by the normal equation, where

$$\Theta = (X^T X)^{-1} X^T y \tag{6}$$

Calculating the inverse of $X^T X$ is computationally costly, so this approach is useful mainly for small systems.

Regularization

Sometimes h_Θ fits the training data well, but is not smooth enough between datapoints. To avoid this a penalty term is introduced.

$$J(\Theta) \longrightarrow J(\Theta) + \lambda \sum_j^n \Theta^2 \tag{7}$$

With this, small values of the parameters Θ are favoured, causing a smoother approximation and less overfitting.