Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 and MoSIG M1 — Winter Semester 2016
Lecture 2 — 5 February 2016

Supervised Learning and Regression

Outline:

Notation

x_d	A feature. An observed or measured value.
$\vec{X}$	A vector of D features.
D	The number of dimensions for the vector $\vec{X}$
$\vec{y}$	A dependent variable to be estimated.
$\hat{y} = f(\vec{X}, \vec{w})$	A model that predicts $\vec{y}$ from $\vec{X}$
$\vec{w}$	The parameters of the model.
$\{\vec{X}_m\}$ $\{y_m\}$	Training samples for learning.
M	The number of training samples.

Regression Analysis

Regression is the estimation of the parameters for a function that maps a set of independent variables into a dependent variable.

$$\hat{y} = f(\vec{X}, \vec{w})$$

Where

$\vec{X}$ is a vector of D independent (unknown) variables.
$\hat{y}$ is an estimate for a variable y that depends on $\vec{X}$.
and
$f()$ is a function, also known as a model, that maps $\vec{X}$ onto $\hat{y}$
$\vec{w}$ is a vector of parameters for the model.

Note:

For $\hat{y}$, the "hat" indicates an estimated value for the target value y
$\vec{X}$ is upper case because it is a random (unknown) vector.

Regression analysis refers to a family of techniques for modeling and analyzing the mapping one or more independent variables from a dependent variable.
For example, consider the following table of age, height and weight for 10 females:

M	AGE	H (M)	W (kg)
1	17	163	52
2	32	169	68
3	25	158	49
4	55	158	73
5	12	161	71
6	41	172	99
7	32	156	50
8	56	161	82
9	22	154	56
10	16	145	46

We can use any two variables to estimate the third.
We can use regression to estimate the parameters for a function to predict any feature $\hat{y}$ from the two other features $\vec{X}$.

For example we can predict Weight from height and age as a function.

$\hat{y} = f(\vec{X}, \vec{w})$ where $\hat{y} = Weight$, $\vec{X} = \begin{pmatrix} Age \\ Height \end{pmatrix}$ and $\vec{w}$ are the model parameters

Linear Models

A linear model has the form

$$\hat{y} = f(\vec{X},\vec{w}) = \vec{w}^T\vec{X} + b = w_1x_1 + w_2x_2 + ... + w_Dx_D + b$$

The vector $\vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{pmatrix}$ are the "parameters" of the model that relates $\vec{X}$ to $\hat{y}$.

The equation $\vec{w}^T\vec{X} + b = 0$ is a hyper-plane in a D-dimensional space,

$\vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{pmatrix}$ is the normal to the hyperplane and b is a constant term.

It is generally convenient to include the constant as part of the parameter vector and to add an extra constant term to the observed feature vector.
This gives a linear model with *D+1* parameters where the vectors are:

$$\vec{X} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix} \text{ and } \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix} \text{ where } w_0 \text{ represents b.}$$

This gives the "homogeneous equation" for the model:

$$\hat{y} = f(\vec{X},\vec{w}) = \vec{w}^T\vec{X}$$

Homogeneous coordinates provide a unified notation for geometric operations.

Lines, Planes and Hyper-planes in homogeneous coordinates
(a quick review of basic geometry)

In homogeneous coordinates, vectors and matrices are expressed with an extra dimension. For example, a point in a 2D space is expressed as:

$$\vec{P} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

In a 2-D space, a line is a set of points that obeys the relation:

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

This is called a "homogeneous" equation because the all terms are first order.
Technically this is a "first order" homogeneous equation.

The equation $w_0 + w_1 x_1^2 + w_2 x_2^2 = 0$ would be a second order homogeneous equation.

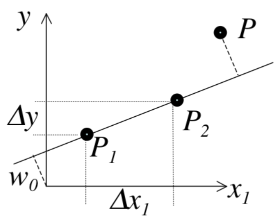
The line equation can be expressed as a simple product of vectors:

$$\vec{W}^T \vec{P} = \begin{pmatrix} w_0 & w_1 & w_2 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} = 0 \quad \text{where } \vec{W} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \text{ and } \vec{P} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

For example we can predict Weight from Height as a linear function.

$\hat{y} = f(\vec{X}, \vec{w})$ where $\hat{y} = Weight$, $\vec{X} = \begin{pmatrix} 1 \\ Height \end{pmatrix}$ and $\vec{w}$ are the model parameters

and the linear model would be a 3D plane in the space $\hat{y} = f(\vec{X}, \vec{w}) = w_0 + w_1 x_1$



We can initialize the model with $w_1 = \frac{y^{(2)} - y^{(1)}}{x_1^{(2)} - x_1^{(1)}}$ and $w_0 = -w_1 x_1^{(1)}$

We can predict Weight from Height and Age as a function.

$\hat{y} = f(\vec{X}, \vec{w})$ where $\hat{y} = Weight$, $\vec{X} = \begin{pmatrix} 1 \\ Age \\ Height \end{pmatrix}$ and $\vec{W} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$ are the model parameters, and the surface is a plane in the space (weight, age, height).
In a D dimensional space, linear homogeneous equation is called a hyper-plane.

Supervised learning

In supervised learning, we learn the parameters of a model from a labeled set of training data. The training data is composed of M sets of independent variables, $\{\vec{X}_m\}$ for which we know the value of the dependent variable $\{y_m\}$.
The training data is the set $\{\vec{X}_m\}$, $\{y_m\}$

Least squares estimation of a hyperplane from set of sample.

For a linear model, learning the parameters of the model from a training set is equivalent to estimating the parameters of a hyperplane using least squares.

In the case of a linear model, there are many ways to estimate the parameters:
For example, matrix algebra provides a direct, closed form solution.

Assume a training set of M observations $\{\vec{X}_m\}$ $\{y_m\}$ where the constant d is included as a "0th" term in $\vec{X}$ and $\vec{w}$.

$$\vec{X} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix} \text{ and } \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$$

We seek the parameters for a linear model: $\hat{y} = f(\vec{X}, \vec{w}) = \vec{w}^T \vec{X}$
This can be determined by minimizing a "Loss" function that can be defined as the Square of the error.

$$L(\vec{w}) = \sum_{m=1}^{M} (\vec{w}^T \vec{X}_m - y_m)^2$$

To build or function, we will use the M training samples to compose a matrix **X** and a vector **Y**.

$$X = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & \cdots & x_{2M} \\ \cdots & \cdots & \ddots & \vdots \\ x_{D1} & x_{D2} & \cdots & x_{DM} \end{pmatrix} \text{ (D+1 rows by M columns)} \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{pmatrix} \text{ (M rows).}$$

We can factor the loss function to obtain: $L(\vec{w}) = (\vec{w}^T X - Y)^T (\vec{w}^T X - Y)$

To minimize the loss function, we calculate the derivative and solve for $\vec{w}$ when the derivative is 0.

$$\frac{\partial L(\vec{w})}{\partial \vec{w}} = 2X^T Y - 2X^T X\vec{w} = 0$$

which gives $X^T Y = 2X^T X\vec{w}$

and thus $\vec{w} = (X^T X)^{-1} X^T Y$

While this is an elegant solution for linear regression, it does not generalize to other models. A more general approach is to use Gradient Descent.

Gradient Descent

Gradient descent is a popular algorithm for estimating parameters for a large variety of models. Here we will illustrate the approach with estimation of parameters for a linear model.

As before we seek to estimate that parameters $\vec{w}$ for a model

$$\hat{y} = f(\vec{X},\vec{w}) = \vec{w}^T \vec{X}$$

from a training set of M samples $\{\vec{X}_m\}$ $\{y_m\}$

We will define our loss function as $\frac{1}{2}$ average error $L(\vec{w}) = \frac{1}{2M}\sum_{m=1}^{M}(f(\vec{X}_m,\vec{w}) - y_m)^2$

where we have included the term $\frac{1}{2}$ to simplify the algebra later.

The gradient is the derivative of the loss function with respect to each term w_d of $\vec{w}$ is

$$\vec{\nabla} f(\vec{X},\vec{w}) = \frac{\partial f(\vec{X},\vec{w})}{\partial \vec{w}} = \begin{pmatrix} \frac{\partial f(\vec{X},w_0)}{\partial w_0} \\ \frac{\partial f(\vec{X},w_1)}{\partial w_1} \\ \vdots \\ \frac{\partial f(\vec{X},w_D)}{\partial w_D} \end{pmatrix}$$

where:

$$\frac{\partial f(\vec{X},w_d)}{\partial w_d} = \frac{1}{M}\sum_{m=1}^{M}(f(\vec{X}_m,\vec{w}) - y_m)x_{dm}$$

x_{dm} is the dth coefficient of the mth training vector. Of course $x_{0m} = 1$ is the constant term.

We use the gradient to "correct" an estimate of the parameter vector for each training sample. The correction is weighted by a learning rate "α"

We can see $\frac{1}{M}\sum_{m=1}^{M}(f(\vec{X}_m,\vec{w}^{(i-1)}) - y_m)x_{dm}$ as the "average error" for parameter $w_d^{(i-1)}$

Gradient descent corrects by subtracting the average error weighted by the learning rate.

Gradient Descent Algorithm

Initialization: *(i=0)* Let $w_d^{(o)} = 0$ for all D coefficients of $\vec{W}$

Repeat until $\left\| L(\vec{w}^{(i)}) - L(\vec{w}^{(i-1)}) \right\| < \varepsilon$: $\vec{w}^{(i)} = \vec{w}^{(i-1)} - \alpha \vec{\nabla} f(\vec{X}, \vec{w}^{(i-1)})$

where $L(\vec{w}) = \frac{1}{2M} \sum_{m=1}^{M} (f(\vec{X}_m, \vec{w}) - y_m)^2$

That is: $w_d^{(i)} = w_d^{(i-1)} - \alpha \frac{1}{M} \sum_{m=1}^{M} (f(\vec{X}_m, \vec{w}^{(i-1)}) - y_m) x_{dm}$

Note that all coefficients are updated in parallel.
The algorithm halts when the change in $\Delta L(\vec{w}^{(i)})$ becomes small:

$$\left\| L(\vec{w}^{(i)}) - L(\vec{w}^{(i-1)}) \right\| < \varepsilon$$

For some small constant ε.

Gradient Descent can be used to learn the parameters for a non-linear model.
For example, when D=2, a second order model would be:

$$\vec{X} = \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \\ x_2 \\ x_1^2 \\ x_1 x_2 \end{pmatrix} \text{ and } f(\vec{X}, \vec{w}) = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_2 + w_4 x_2^2 + w_5 x_1 x_2$$

Practical Considerations for Gradient Descent

The following are some practical issues concerning gradient descent.

Feature Scaling

Make sure that features have similar scales (range of values). One way to assure this is to normalize the training date so that each feature has a range of 1.

Simple technique: Divide by the Range of sample values.
For a training set $\{\vec{X}_m\}$ of M training samples with D values.

Range: $r_D = Max(x_d) - Min(x_d)$

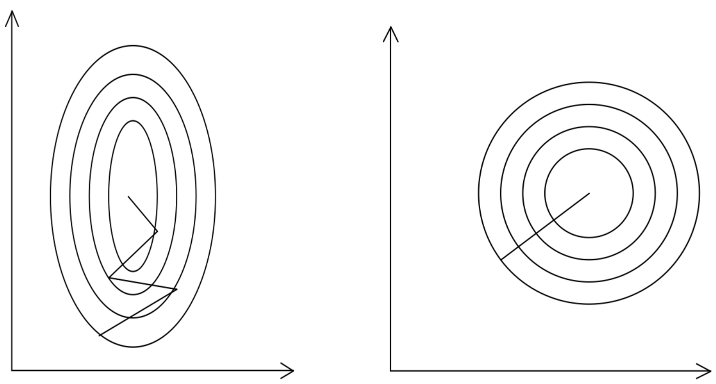
Then

$$\forall_{m=1}^{M} : x_{dm} := \frac{x_{dm}}{r_d}$$

Even better would be to scale with the mean and standard deviation of the each feature in the training data

$$\mu_d = E\{x_{dm}\} \qquad \sigma^2 = E\{(x_{dm} - \mu_d)^2\}$$

$$\forall_{m=1}^{M} : x_{dm} := \frac{(x_{dm} - \mu_d)}{\sigma_d}$$



Verifying Gradient Descent

The value of the loss function should always decrease:
Verify that $L(\vec{w}^{(i)}) - L(\vec{w}^{(i-1)}) < 0$.

if $L(\vec{w}^{(i)}) - L(\vec{w}^{(i-1)}) > 0$ then decrease the learning rate "α"

Gradient Descent vs Direct Solution

Form M training samples composed of D features:

<u>Direct Solution</u>:

Advantages:

1) No need to choose a learning rate (α)
2) No need to iterate - Predictable computational cost.

Inconvenient: Need to compute $(\vec{X}^T\vec{X})^{-1}$ which has a computational cost *$O(M^3)$*

<u>Gradient Descent</u>:

Advantages: Works well for Large

Inconvenient:

1) Need to choose a learning rate (α)
2) Can require many iterations to converge, each iteration costs O(M).
   (number of iterations not known in advance.)