Pragmatic Programming Techniques

Sunday, November 8, 2009

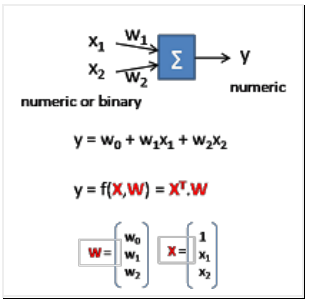
Machine Learning with Linear Model

Linear Model is a family of model-based learning approaches that assume the output y can be expressed as a linear algebraic relation with the input attributes x1, x2 ...

The input attributes x1, x2 ... is expected to be numeric and the output is expected to be numeric as well.

Here our goal is to learn the parameters of the underlying model, which is the coefficients.

Linear Regression



Here the input and output are both numeric, related through a simple linear relationship. The learning goal is to figure out the hidden weight value (ie: the W vector).

Notice that non-linear relationship is equivalent of a linear relationship at a higher dimension. e.g. if x2 = x1 * x1, then it becomes a quadratic relationship. Because of this, the polynomial regression can be done using linear regression technique.

Given a batch of training data, we want to figure out the weight vector W such that the total sum of error (which is the difference between the predicted output and the actual output) to be minimized.

$$err = y - (w_0 + w_1x_1 + w_2x_2)$$

Find w_0, w_1, w_2 to minimize $\sum err_i^2 = \sum(y_i - (w_0 + w_1x_{i1} + w_2x_{i2}))^2$

For $k = 0, 1, 2 \quad \partial(\sum err_i^2) / \partial(w_k) = 0$

$$\partial(\sum err_i^2) / \partial(w_0) = \sum 2(y_i - (w_0 + w_1x_{i1} + w_2x_{i2})).(-1) = 0$$
$$\partial(\sum err_i^2) / \partial(w_1) = \sum 2(y_i - (w_0 + w_1x_{i1} + w_2x_{i2})).(-x_{i1}) = 0$$
$$\partial(\sum err_i^2) / \partial(w_2) = \sum 2(y_i - (w_0 + w_1x_{i1} + w_2x_{i2})).(-x_{i2}) = 0$$

which is ...

$$\sum y_i = nw_0 + w_1\sum x_{i1} + w_2\sum x_{i2}$$
$$\sum x_{i1}y_i = w_0\sum x_{i1} + w_1\sum x_{i1}x_{i1} + w_2\sum x_{i2}x_{i1}$$
$$\sum x_{i2}y_i = w_0\sum x_{i2} + w_1\sum x_{i2}x_{i1} + w_2\sum x_{i2}x_{i2}$$

Express in Matrix form ...

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ x_{11} & x_{21} & x_{31} & x_{41} & x_{51} \\ x_{12} & x_{22} & x_{32} & x_{42} & x_{52} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ x_{11} & x_{21} & x_{31} & x_{41} & x_{51} \\ x_{12} & x_{22} & x_{32} & x_{42} & x_{52} \end{pmatrix} \begin{pmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \\ 1 & x_{41} & x_{42} \\ 1 & x_{51} & x_{52} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$X^T Y = X^T X W$$
$$(X^T X)^{-1} X^T Y = (X^T X)^{-1} (X^T X) W$$
$$W = (X^T X)^{-1} X^T Y$$

Instead of using the batch processing approach, a more effective approach is to learn incrementally (update the weight vector for each input data) using a gradient descent approach.

Gradient Descent

Gradient descent is a very general technique that we can use to incrementally adjust the parameters of the linear model. The basic idea of "gradient descent" is to adjust each dimension

About Me

Ricky Ho

I am a software architect and consultant passionate in Distributed and parallel computing, Machine learning and Data mining, SaaS and Cloud computing.

View my complete profile

Popular Posts

MongoDb Architecture
NOSQL has become a very heated topic for large web-scale deployment where scalability and semi-structured data driven the DB requirement tow...

Designing algorithms for Map Reduce
Since the emerging of Hadoop implementation, I have been trying to morph existing algorithms from various areas into the map/reduce model. ...

NOSQL Patterns
Over the last couple years, we see an emerging data storage mechanism for storing large scale of data. These storage solution differs quite...

Couchbase Architecture
After receiving a lot of good feedback and comment on my last blog on MongoDb , I was encouraged to do another deep dive on another popular ...

Predictive Analytics: Overview and Data visualization
I plan to start a series of blog post on predictive analytics as there is an increasing demand on applying machine learning technique to ana...

Predictive Analytics: Generalized Linear Regression
In the previous 2 posts, we have covered how to visualize input data to explore strong signals as well as how to prepare input data to a fo...

BigTable Model with Cassandra and HBase
Recently in a number of "scalability discussion meeting", I've seen the following pattern coming up repeatedly ... To make you...

Blog Archive

$$err = y - f(X,W)$$

$$W^{t+1} = W^t - \alpha.\partial(err^2) / \partial(W)$$

$$W^{t+1} = W^t - \alpha \begin{pmatrix} \partial(err^2) / \partial(w_0) \\ \partial(err^2) / \partial(w_1) \\ \partial(err^2) / \partial(w_2) \end{pmatrix}$$

(w0, w1, w2) of the W vector according to their contribution of the square error. Their contribution is measured by the gradient along the dimension which is the differentiation of the square error with respect to w0, w1, w2.

In the case of Linear Regression ...

$$\partial(err^2) / \partial(w_0) = 2(y - (w_0 + w_1x_1 + w_2x_2)).(-1)$$
$$\partial(err^2) / \partial(w_1) = 2(y - (w_0 + w_1x_1 + w_2x_2)).(-x_1)$$
$$\partial(err^2) / \partial(w_2) = 2(y - (w_0 + w_1x_1 + w_2x_2)).(-x_2)$$

$$W^{t+1} = W^t + \alpha \begin{pmatrix} y - (w_0 + w_1x_1 + w_2x_2) \\ (y - (w_0 + w_1x_1 + w_2x_2)).x_1 \\ (y - (w_0 + w_1x_1 + w_2x_2)).x_2 \end{pmatrix}$$

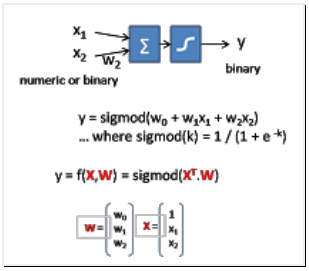
$$W^{t+1} = W^t + \alpha(y - (w_0 + w_1x_1 + w_2x_2)) \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

$$W^{t+1} = W^t + \alpha.err.X$$

Logistic Regression

Logistic Regression is used when the output y is binary and not a real number. The first part is the same as linear regression while a second step sigmod function is applied to clamp the output value between 0 and 1.

We use the exact same gradient descent approach to determine the weight vector W.



$$err = y - sigmod(w_0 + w_1x_1 + w_2x_2)$$

$$\partial(err^2) / \partial(w_0) = 2.err.\partial(err) / \partial(w_0)$$
Since $\partial(sigmod(k)) / \partial(k) = sigmod(k) . (1 - sigmod(k))$
$$\partial(err^2) / \partial(w_0) = -2.err.sigmod(w_0 + w_1x_1 + w_2x_2) . (1 - sigmod(w_0 + w_1x_1 + w_2x_2)).1$$
$$\partial(err^2) / \partial(w_1) = -2.err.sigmod(w_0 + w_1x_1 + w_2x_2) . (1 - sigmod(w_0 + w_1x_1 + w_2x_2)).x_1$$
$$\partial(err^2) / \partial(w_2) = -2.err.sigmod(w_0 + w_1x_1 + w_2x_2) . (1 - sigmod(w_0 + w_1x_1 + w_2x_2)).x_2$$

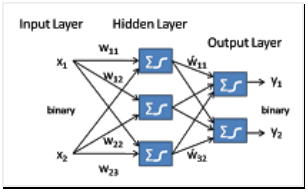
Let the estimated output $\hat{y} = sigmod(w_0 + w_1x_1 + w_2x_2)$

$$W^{t+1} = W^t + \alpha \begin{pmatrix} err . \hat{y} . (1 - \hat{y}) . 1 \\ err . \hat{y} . (1 - \hat{y}) . x_1 \\ err . \hat{y} . (1 - \hat{y}) . x_2 \end{pmatrix}$$

$$W^{t+1} = W^t + \alpha . err . \hat{y} . (1 - \hat{y}) \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

$$W^{t+1} = W^t + \alpha.err.\hat{y}.(1 - \hat{y})X$$

Neural Network



Inspired by how our brain works, Neural network organize many logistic regression units into layers of perceptrons (each unit has both input and outputs in binary form).

Learning in Neural network is to discover all the hidden values of w. In general, we use the same technique above to adjust the weight using gradient descent layer by layer. We start from the output layer and move towards the input layer (this technique is called ***backpropagation***). Except the output layer, we don't exactly know the error at the hidden layer, we need to have a way to estimate the error at the hidden layers.

But notice there is a symmetry between the weight and the input, we can use the same technique how we adjust the weight to estimate the error of the hidden layer.

▼ 2009 (31)
- ► December (1)
- ▼ November (9)
  - Query Processing for NOSQL DB
  - Cloud Computing Patterns
  - Impression on Scala
  - NOSQL Patterns
  - Amazon Cloud Computing
  - Support Vector Machine
  - Machine Learning with Linear Model
  - What Hadoop is good at
  - Principal Component Analysis
- ► October (4)
- ► September (2)
- ► August (3)
- ► July (2)
- ► May (6)
- ► April (1)
- ► January (3)

► 2008 (22)

► 2007 (11)

Search This Blog

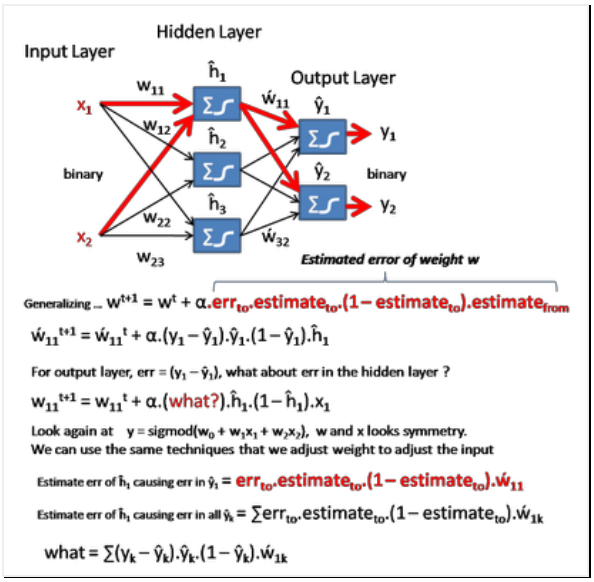
Search

Labels

machine learning data mining map reduce Architecture Design Cloud computing algorithm NOSQL Hadoop scalability Distributed system parallel processing big data predictive analytics Design patterns SOA performance REST ensemble method recommendation engine

Pages

- Home



Posted by Ricky Ho at 5:33 PM

+1 Recommend this on Google

1 comment:

Michele Filannino said...

Hi Mr. Ho, first of all congratulation for your blog. It's so interesting! :)

In respect of Neural Networks, there are different tecniques we can use to adjust weights and they substantially depend from neural network topology.

In particular there are:
- Wiener-Hopf Method
- Steepest Descent Method (your Gradient Algorithm)
- Least-Mean-Square Algorithm (Stocastic Gradient Algorithm)

With Kolmogorov theorem we know that each continue, limited and mototone function (with n variables) could be represented as sum of many mono-variable functions. The problem is that this is a Theorem of existence: it say that exists a set of functions but do not say how we can calculate them. For this reason we use a Neural Network.

Bye and congratulation again.

December 2, 2009 at 12:39 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)