Pragmatic Programming Techniques

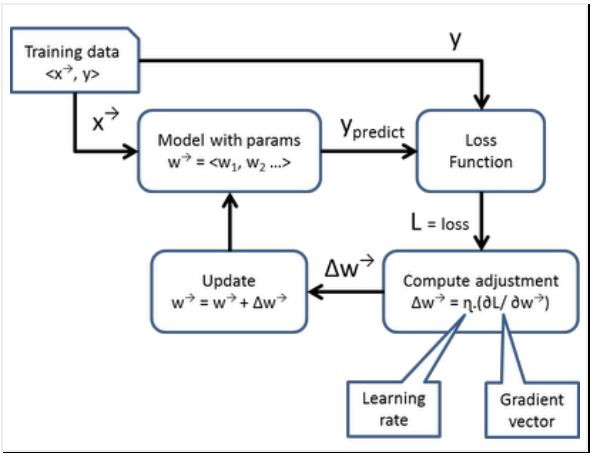
Thursday, October 4, 2012

Machine Learning in Gradient Descent

In Machine Learning, gradient descent is a very popular learning mechanism that is based on a greedy, hill-climbing approach.

Gradient Descent

The basic idea of Gradient Descent is to use a feedback loop to adjust the model based on the error it observes (between its predicted output and the actual output). The adjustment (notice that there are multiple model parameters and therefore should be considered as a vector) is pointing to a direction where the error is decreasing in the steepest sense (hence the term "gradient").



Notice that we intentionally leave the following items vaguely defined so this approach can be applicable in a wide range of machine learning scenarios.

- The Model
- The loss function
- The learning rate

Gradient Descent is very popular method because of the following reasons ...

- Intuitive and easy to understand
- Easy to run in parallel processing architecture
- Easy to run incrementally with additional data

On the other hand, the greedy approach in Gradient Descent can be trapped in local optimum. This can be mitigated by choosing a convex LOSS function (which has a single optimum), or multiple starting points can be picked randomly (in the case, we hope the best local optimum is close to the global optimum).

Batch vs Online Learning

While some other Machine Learning model (e.g. decision tree) requires a batch of data points before the learning can start, Gradient Descent is able to learn each data point independently and hence can support both batch learning and online learning easily. The difference lies in how the training data is fed into the model and how the loss function computes its error.

In batch learning, all training will be fed to the model, who estimates the output for all data points. Error will then be summed to compute the loss and then update the model. Model in this case will be updated after predicting the whole batch of data points.

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

► 2013 (8)
▼ 2012 (18)
► November (1)
▼ October (1)

In online learning mode (also called stochastic gradient descent), data is fed to the model one at a time while the adjustment of the model is immediately made after evaluating the error of this single data point. Notice that the final result of incremental learning can be different from batch learning, but it can be proved that the difference is bound and inversely proportional to the square root of the number of data points.

The learning rate can be adjusted as well to achieve a better stability in convergence. In general, the learning rate is higher initially and decrease over the iteration of training (in batch learning it decreases in next round, in online learning it decreases at every data point). This is quite intuitive as you paid less attention to the error as you have learn more and more. Because of that online learning is sensitive to the arrival order of data.

One way to adjust the learning rate is to have a constant divide by the square root of N (where N is the number of data point seen so far).

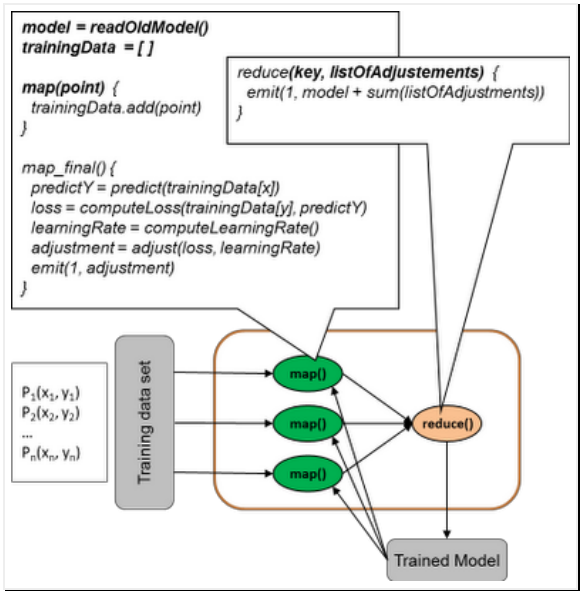
$\eta = \eta\_initial / (t \hat{} 0.5)$.

By using different decay factor, we can control how much attention we should pay for the late coming data. In online learning, as data comes in time of occurrence, we can play around with this decay factor to guide how much attention the learning mechanism should be paying to latest arrival data. Online learning automatically adapt to change of trends over time.

Most real world machine learning scenario relies on stationality of the model. By the way, learning is about "learning from past experience". If the environment changes too rapidly that the past experience is invalid, there is little value to learn. Because of this reason, most machine learning project are satisfied by using batch learning (daily or weekly) and the demand of online learning is not very high. A very common batch learning model is described in my previous blog here.

Parallel Learning

Because of no dependency in data processing, Gradient Descent is very easy to put into a parallel processing environment such as Map/Reduce. Here we illustrate how to parallelize the execution of batch learning.



Notice that there are multiple rounds of Map/Reduce until the model converges. On the other hand, online learning is not possible for Hadoop Map/Reduce which doesn't support real-time at this moment.

In summary, gradient descent is a very powerful approach of machine learning and works well in a wide spectrum of scenarios.

Posted by Ricky Ho at 12:23 PM

+1 +24 Recommend this on Google

Labels: data mining, gradient descent, machine learning

6 comments:

Simon Skov Boisen said...

For a framework around online gradient descent (and online processing in general) I would

Machine Learning in Gradient Descent

► September (1)
► August (2)
► July (1)
► June (3)
► May (3)
► April (3)
► March (1)
► February (1)
► January (1)

► 2011 (6)
► 2010 (18)
► 2009 (31)
► 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

point to Storm which is a distributed and fault-tolerant realtime computation framework developed by Nathan Marz from Twitter.

October 4, 2012 at 11:37 PM

Agnonchik said...

Hi,
I have learnt that one should randomly pick up training examples when applying stochastic gradient descent, which might not be true for your MapRedice pseudocode. If I understood you correctly, each mapper will processes a subset of training examples and they will do it in parallel. The problem is that these calculations can be interdependent and then different ordering produces different results. Is this still OK?
Perhaps, the processing order is not so important for SGD, is it?
Thank you!

November 7, 2012 at 10:14 AM

Agnonchik said...

Hi,
I have learnt that one should randomly pick up training examples when applying stochastic gradient descent, which might not be true for your MapRedice pseudocode. If I understood you correctly, each mapper will processes a subset of training examples and they will do it in parallel. The problem is that these calculations can be interdependent and then different ordering produces different results. Is this still OK?
Perhaps, the processing order is not so important for SGD, is it?
Thank you!

November 7, 2012 at 10:15 AM

benslin kard said...

Cassandra boots quickly, and its performance scales smoothly as new nodes are added.

November 8, 2012 at 1:58 AM

Ricky Ho said...

Yes, SGD is non-deterministic anyway.
So different order is OK.

November 9, 2012 at 2:24 PM

katherine james said...

Very interesting entry, I look forward to the next! Thanks for sharing. I am searching for informative articles about Australian web design services. If you have idea about good posts of this please share with me

June 11, 2013 at 11:08 PM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)