Stochastic Gradient Descent

Harold Doran
Human Resources Research Organization
hdoran@humrro.org

Computational Topics for Psychometricians:
Algorithms Used in Machine Learning

Background

- Psychometrics must confront the changing dynamics associated with massive data and computationally demanding psychometric models.
- We no longer simply deal with "big" data but instead with "massive" data which may have many millions of observations.
- This will require that we consider new ways to analyze data that "scales"—or grows in relationship to the size of the problem.
- This presentation will focus on one scalable algorithm popular in machine learning termed **stochastic gradient descent**.

Why Focus on Scalable Algorithms?

- We can train machines (e.g., self-driving cars) to recognize patterns (e.g., stop signs) and make a decision (e.g., stop the car).
- But all of that training sits on the shoulders of numerical methods (i.e., algorithms) that analyze massive data files.
- We need stable and scalable algorithms to make these things possible and extend the scientific reach of modern psychometrics.
- We say an algorithm is "scalable" when it is flexible and can grow with an increasingly large number of inputs.

Machine/Deep Learning is Changing Psychometrics

- Machine/deep learning (ML) is taking significant foothold in psychometrics (e.g., automated essay scoring, IRT parameter forecasting) and these tend to use very large data and very computationally demanding methods.
- The idea of ML can be explained as teaching a machine to "learn" from data and and then later make a prediction or a decision when presented with new data.
- ML models tend to focus on prediction or classification using models that in many respects are familiar to quantitative scientists (e.g., logistic regression).
- However, the means by which analyses are performed is changing rapidly and we need to open the black box and consider algorithms used in popular environments like PyTorch/TensorFlow.

Stochastic Gradient Descent

Definition: Stochastic Gradient Descent in Words

A general purpose optimization technique useful for minimization of any differentiable objective (loss) function. It is a powerful approach for building highly scalable applications that reduces the computational burden associated with large data by simplifying it to an iterative solution passing over the gradient one sample at a time and iteratively updating the parameter estimates.

- We can introduce the concept of SGD using ordinary least squares.
- SGD has much broader applications, but this will connect the new concept to our prior knowledge using a standard and familiar way to detect patterns and form predictions.

Linear Least Squares

- OLS, or linear regression, is a model that relates a dependent variable, y (or a target in ML terms), to a set of predictor variables, x, or covariates (or features in ML terms).

Simple Linear Regression

$$y_i = \mu + \beta_1 x_{i1} + \beta_2 x_{i2} + e_i$$

- We might "run" this regression on our data and obtain estimates for the fixed effects, $\widehat{\mu}$, $\widehat{\beta}_1$, and $\widehat{\beta}_2$ and then use those parameter estimates to form predictions.

Forming Predictions

$$\widehat{y}_i = \widehat{\mu} + \widehat{\beta}_1 x_{i1} + \widehat{\beta}_2 x_{i2}$$

What Have We Done?

- If the data we used to estimate the fixed effects was "good" we have now completed our training.
- We would now test out our model on a validation set to evaluate how well it works—a topic we will not explore here.
- If our data for training is small we can use standard software for linear regression and do not need SGD.
- But, what if our data are really big, or massive—so massive that standard software for linear regression chokes and cannot estimate the parameters we need for training?
- This is where we can use SGD (or a variant called mini-batch) instead of using standard OLS procedures.

A Small Math Detour

- It is more convenient to consider OLS in the following matrix form

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

  where $\boldsymbol{Y} \in \mathbb{R}^n$, $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, $\boldsymbol{\beta} \in \mathbb{R}^p$, $\boldsymbol{\epsilon} \in \mathbb{R}^n$, and $p \ll n$.
- We sometimes see the "solution" for the estimates of the fixed effects written as

Algebraic OLS Representation

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}.$$

This is an algebraically correct representation, but not a computationally accurate representation. **Do not compute it this way**.

A Small Math Detour

- We sometimes see code taking the same form as an algebraic expression and this is rarely the right way to approach a problem.
- Instead, use decompositions and solve via triangular systems and avoid matrix inversion (e.g., Cholesky, QR, SVD).
- For further references see the white paper at this URL `https://hdoran.github.io/Blog/compPsych.pdf`.
- The point is with small data, standard procedures work just fine.
- However, when the dimensions of the data become extremely large, standard procedures do not scale well and may not even work with the data.

Stochastic Gradient Descent

- SGD is the alternative approach we might use in this scenario to obtain parameters with massive data.
- SGD never works with the full model matrix $\boldsymbol{X}$, but instead only works with individual rows in this matrix to compute gradients and updates parameters.
- The term "stochastic" here is used meaning it randomly chooses rows in the matrix $\boldsymbol{X}$ to work with.
- It is not used in the sense that we take random draws from some distribution, $h(\cdot)$, as we would commonly use in MCMC settings.
- We are using "real" data, but we randomly choose which data are used at each iteration of the SGD process.

Stochastic Gradient Descent: The Formal Definition

- Let $\nabla \mathcal{J}_i(\boldsymbol{\theta})$ denote the gradient vector of the objective (loss) function with respect to the parameters to be minimized for an ith sample in the data.

Stochastic Gradient Descent

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla \mathcal{J}_i(\boldsymbol{\theta})$$

- We refer to α as a learning rate *hyperparameter*
- The gradient is a vector of length p containing the partial derivatives of the parameters in the objective function.

Stochastic Gradient Descent: The Algorithm

Algorithm 1 Stochastic Gradient Descent

Input: Choose initial values for $\boldsymbol{\theta}$, $\nabla \mathcal{J}_i(\boldsymbol{\theta})$, and select a value for α.

1: Set $\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t - \alpha \nabla \mathcal{J}_i(\boldsymbol{\theta})$ at iteration t.
2: Sample $\boldsymbol{X}_i$, in $\boldsymbol{X}$ and its corresponding value, $\boldsymbol{y}_i$.
3: Compute $\nabla \mathcal{J}_i(\boldsymbol{\theta})$ using the tuple $\{\boldsymbol{X}_i, \boldsymbol{y}_i\}$.
4: Iterate between steps (1) and (3) until criterion for a stopping rule has been satisfied.

Stochastic Gradient Descent: The Terminology

Stochastic Gradient Descent

When $i = 1$, then $\boldsymbol{X}_i$ is the ith row of the model matrix.

Mini-Batch Gradient Descent

When $1 < i < n$, the same iterative process can be used in small batches and $\boldsymbol{X}_i$ is a collection of rows in the model matrix.

Batch Gradient Descent

When $i = n$, then $\boldsymbol{X}_i$ is the entire model matrix.

Other Terms

Iterations are termed **epochs** and the **learning rate** is a value (sometimes a scalar but may also be a matrix) used for tuning the model and may change on a **schedule** as the algorithm proceeds to become more refined.

Stochastic Gradient Descent: Basic R Code (OLS)

```
sgd <- function(size,alpha,y,X,iter=1000){
  beta <- matrix(0, nrow=iter, ncol=ncol(X))
  for(iter in seq_len(iter - 1)) {
    samp <- sample(seq_len(nrow(X)), size = size)
    y.samp <- y[samp]; X.samp <- X[samp,]
    if(size == 1) X.samp <- t(as.matrix(X.samp))
    gradient <- (2 / size) * (crossprod(X.samp)
        %*% beta[iter, ] - crossprod(X.samp,y.samp
        ))
    beta[iter + 1, ] <- beta[iter, ] - alpha *
        gradient
  }
  list(coef = beta[iter,], history = beta)
}
```

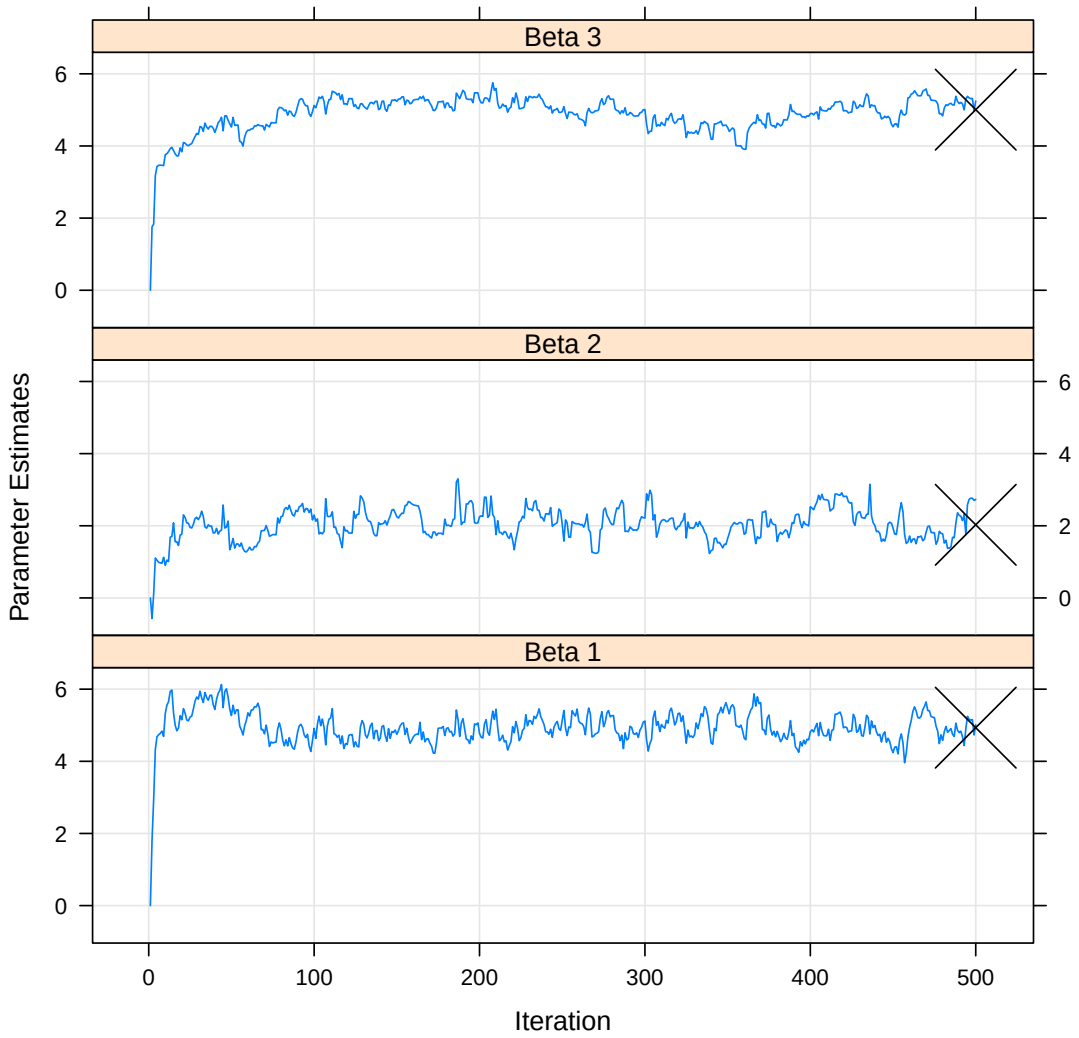
Simple Self-Contained Example

```
Generate simulated data
N <- 1000
df <- data.frame(x1=rnorm(N), x2=rnorm(N),
    epsilon=rnorm(N))
df$y <- 5 + 2 * df$x1 + 5 * df$x2 + df$epsilon
Run linear regression using R built in function
out1 <- lm(y ~ x1 + x2, df)
X <- model.matrix(out1)
Run SGD
out2 = sgd(size = 1, alpha = .1, df$y,X,iter=500)
}
```

- Experiment with the `size` and the `alpha` arguments.
- If `size` is larger than 1, this is mini-batch.
- If `size = 1000`, then this is standard batch gradient descent.

Stochastic Gradient Descent: A Simple Result

- The plot illustrates the result of a simple linear regression using SGD.
- The "X" marks the parameter estimates obtained using the closed form expression.
- The blue line shows that SGD wiggles through the parameter space and rather nicely ends up in about the right spot.



Summary

- Here we have introduced the concept of stochastic gradient descent using simple linear regression and it is a motivating use case to explore for purposes of learning.
- However, SGD is a general purpose tool that can be used much more broadly than in this simple scenario.
- Implementation of SGD in other ways (e.g., logistic regression, ordered probit models) simply requires that we find the appropriate gradient for our problem and then implement the same process outlined in Algorithm 1.
- What is also exciting about SGD is that it has two other notions that help it scale:
  - SGD can stream in only small subsets of the data instead of reading in the entire data file.
  - SGD can be executed in parallel.
- The combination of these ideas with SGD improve its potential to serve psychometric modeling in very powerful ways.