This paper talks about the importance of stochastic gradient descent with a large training set as well as a few methods or tricks for the effective usage of SGD. SGD is an iterative technique for optimizing an objective function with acceptable smoothness qualities in stochastic gradient descent, or SGD (e.g., differentiable, or subdifferentiable). Since it uses an estimate of the gradient instead of the actual gradient (derived from the whole data set), it can be thought of as a stochastic approximation of gradient descent optimization (calculated from a randomly selected subset of the data). This lessens the extremely high computational cost, especially in high-dimensional optimization problems, allowing for faster iterations at the expense of a reduced convergence rate.

The noisy approximation of the true gradient restricts the stochastic gradient descent's rate of convergence. The variance of the parameter estimate w_t(weights) falls slowly when the learning rates are too slow to decrease. The expectation of the parameter estimate w_t(weights) takes a very long time to approach the optimum when the learning rates decline too quickly.

The gradients are multiplied by a positive definite matrix t that approaches the inverse of the Hessian in second-order stochastic gradient descent. This adjustment does not significantly improve the variance of w_t(weights) since it does not decrease stochastic noise. The paper recommends using SGD when training time is the bottleneck. Despite being the weakest optimization algorithms, SGD, and 2SGD take less time than other algorithms to reach a predefined expected risk. Consequently, stochastic learning algorithms perform asymptotically better in the large-scale setup, that is, where the processing time rather than the number of samples is the limiting factor.

A few recommendations provided in the paper while using SGD are:

1. Preprocessing the data like feature scaling
2. Examine validation error and keep track of training cost (should decrease)
3. Check the gradient using finite differences
4. Experiment with learning rates using a small training set sample

A few recommendations provided in the paper while using SGD with L2 regularization are:

1. Leverage the sparsity of the training examples
2. Use learning rates of the form $\gamma_t = \gamma_0 (1 + \gamma_0 \lambda t)^{-1}$
3. Try averaged stochastic gradient with Learning rates of the form $\gamma_t = \gamma_0 (1 + \gamma_0 \lambda t)^{-3/4}$ and Averaging rates $\mu_t = 1 / \max \{1, t-d, t-n\}$

The paper concludes by reporting some experimental results illustrating the actual performance of SGD and ASGD on a variety of linear systems:

1. SGD runs considerably faster than either the standard SVM solvers SVMLight and SVMPerf or the super-linear optimization algorithm TRON.
2. ASGD achieves near-optimal results after one epoch only.
3. SGDQN appears more attractive because ASGD does not reach its asymptotic performance. All three algorithms (SGD, SGDQN, ASGD) reach the best test set performance in a couple of minutes.

References:

Bottou, Léon. Stochastic gradient descent tricks. Neural networks: Tricks of the trade, 2012.