CSE-712 SEMINAR: PRESENTATION

Paper - Stochastic Gradient Descent Tricks - by Leon Bottou

Akash Deshmukh(50408237)

University at Buffalo The State University of New York

Introduction

The paper provides materials to explain why Stochastic Gradient Descent is a good learning algorithm when the training set is large. The following topics are discussed in this paper:

- What is Stochastic Gradient Descent ? Difference between Gradient Descent and SGD.
- Convergence of SGD
- When to use SGD ? Comparison of asymptotic analysis considering large dataset
- General Recommendations for using SGD algorithms
- Recommendations for training large linear models with L_2 regularization.

What is Stochastic Gradient Descent

Consider a supervised learning problem :

- z : pair of x,y (x : vector input, y: scalar output)
- l : loss function l (y^, y) measures the cost of predicting y^ when the actual answer is y

$$\boxed{E_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)}$$

- The empirical risk $E_n(f)$ measures the training set performance.
- We seek the function f that minimizes the loss Q(z,w) = l(fw(x); y) averaged on the examples.

Gradient Descent

- Minimize the empirical risk En(fw) using gradient descent (GD). Each iteration updates the weights w on the basis of the gradient of En(fw)

$$\boxed{w_{t+1} \;=\; w_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(z_i, w_t)\,,}$$

- where ϒ is an adequately chosen learning rate.
- Under sufficient regularity assumptions, when the initial estimate w is close enough to the optimum, and when the learning rate is sufficiently small, this algorithm achieves linear convergence.

University at Buffalo The State University of New York

Gradient Descent contd....

- Much better optimization algorithms can be designed by replacing the scalar learning rate by a positive definite matrix Γt that approaches the inverse of the Hessian of the cost at the optimum:

$$w_{t+1} = w_t - \Gamma_t \frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(z_i, w_t) .$$

- This second order gradient descent (2GD) is a variant of the well-known Newton algorithm.
- Under sufficiently optimistic regularity assumptions and provided that w_0 is sufficiently close to the optimum, second order gradient descent achieves quadratic convergence. When the cost is quadratic and the scaling matrix Γ is exact, the algorithm reaches the optimum after a single iteration.

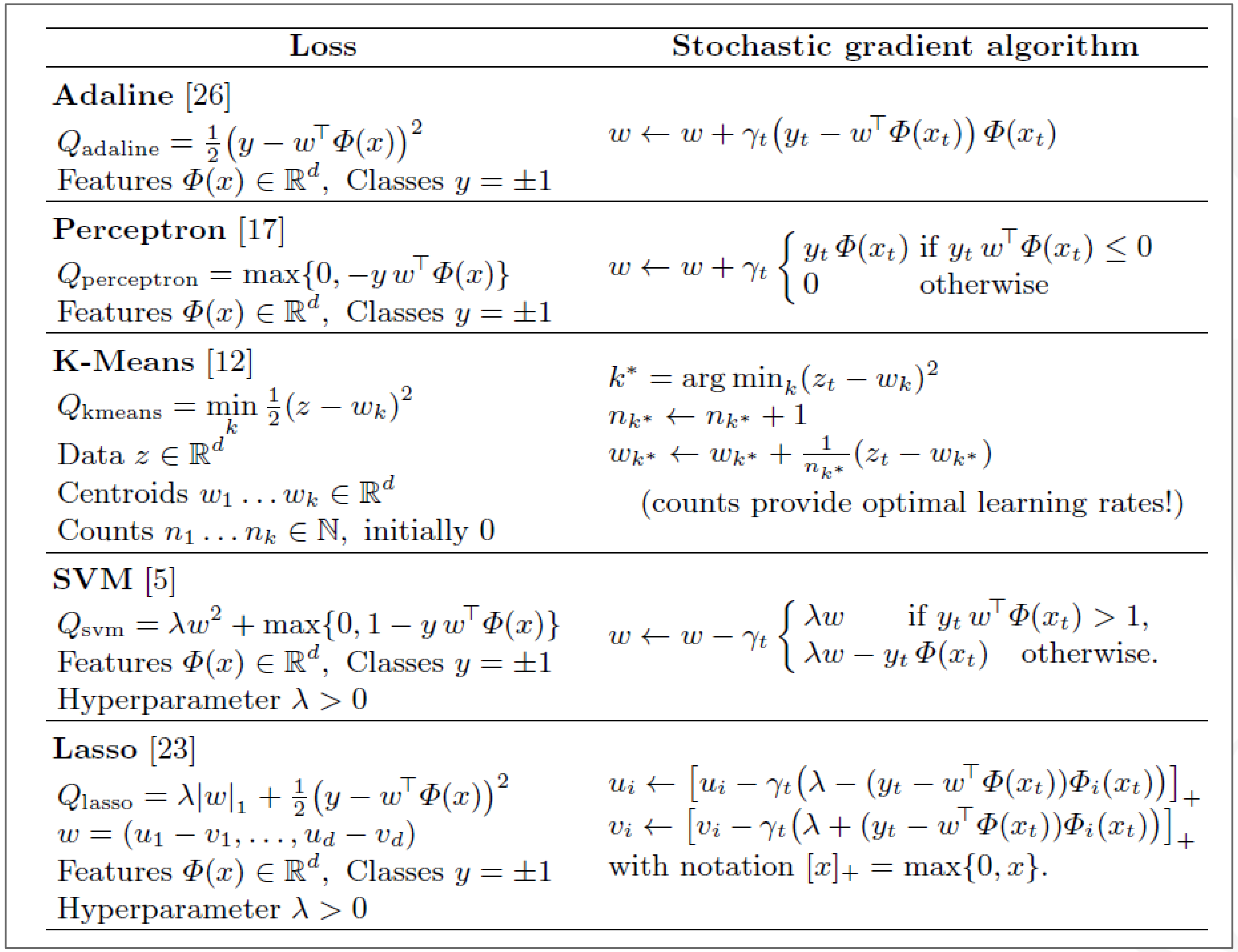
Stochastic Gradient Descent

- The stochastic gradient descent (SGD) algorithm is a drastic simplification. Instead of computing the gradient of En(fw) exactly, each iteration estimates this gradient based on a single randomly picked example z_t

$$\boxed{w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t) .}$$

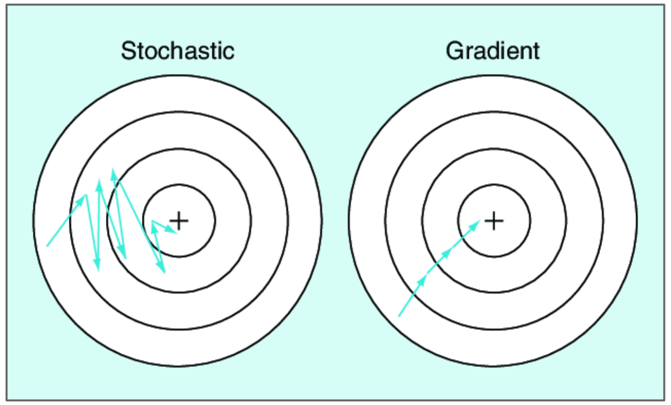
- The stochastic process depends on the examples randomly picked at each iteration.
- It is hoped that SGD behaves like its expectation GD despite the noise introduced by this simplified procedure.
- Since the stochastic algorithm does not need to remember which examples were visited during the previous iterations, it can process examples on the fly in a deployed system. In such a situation, the stochastic gradient descent directly optimizes the expected risk, since the examples are randomly drawn from the ground truth distribution.

Stochastic Gradient Descent for various systems



Loss	Stochastic gradient algorithm
Adaline [26] $Q_{\text{adaline}} = \frac{1}{2}\left(y - w^\top \Phi(x)\right)^2$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$	$w \leftarrow w + \gamma_t \left(y_t - w^\top \Phi(x_t)\right) \Phi(x_t)$
Perceptron [17] $Q_{\text{perceptron}} = \max\{0, -y\, w^\top \Phi(x)\}$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$	$w \leftarrow w + \gamma_t \begin{cases} y_t\, \Phi(x_t) & \text{if } y_t\, w^\top \Phi(x_t) \le 0 \\ 0 & \text{otherwise} \end{cases}$
K-Means [12] $Q_{\text{kmeans}} = \min_k \frac{1}{2}(z - w_k)^2$ Data $z \in \mathbb{R}^d$ Centroids $w_1 \dots w_k \in \mathbb{R}^d$ Counts $n_1 \dots n_k \in \mathbb{N}$, initially 0	$k^* = \arg\min_k (z_t - w_k)^2$ $n_{k^*} \leftarrow n_{k^*} + 1$ $w_{k^*} \leftarrow w_{k^*} + \frac{1}{n_{k^*}}(z_t - w_{k^*})$ (counts provide optimal learning rates!)
SVM [5] $Q_{\text{svm}} = \lambda w^2 + \max\{0, 1 - y\, w^\top \Phi(x)\}$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$ Hyperparameter $\lambda > 0$	$w \leftarrow w - \gamma_t \begin{cases} \lambda w & \text{if } y_t\, w^\top \Phi(x_t) > 1, \\ \lambda w - y_t\, \Phi(x_t) & \text{otherwise.} \end{cases}$
Lasso [23] $Q_{\text{lasso}} = \lambda \lvert w \rvert_1 + \frac{1}{2}\left(y - w^\top \Phi(x)\right)^2$ $w = (u_1 - v_1, \dots, u_d - v_d)$ Features $\Phi(x) \in \mathbb{R}^d$, Classes $y = \pm 1$ Hyperparameter $\lambda > 0$	$u_i \leftarrow \left[u_i - \gamma_t\left(\lambda - (y_t - w^\top \Phi(x_t))\Phi_i(x_t)\right)\right]_+$ $v_i \leftarrow \left[v_i - \gamma_t\left(\lambda + (y_t - w^\top \Phi(x_t))\Phi_i(x_t)\right)\right]_+$ with notation $[x]_+ = \max\{0, x\}$.

Convergence of Stochastic Gradient Descent



Source - https://www.researchgate.net/figure/Stochastic-gradient-descent-compared-with-gradient-descent_fig3_328106221

- The convergence speed of stochastic gradient descent is limited by the noisy approximation of the true gradient.
- When the learning rates decrease too slowly, the variance of the parameter estimate w_t decreases equally slowly.
- When the learning rates decrease too quickly, the expectation of the parameter estimate w_t takes a very long time to approach the optimum.

Second order Stochastic Gradient Descent

- Second order stochastic gradient descent (2SGD) multiplies the gradients by a positive definite matrix Γ_t approaching the inverse of the Hessian :

$$\boxed{w_{t+1} = w_t - \gamma_t \Gamma_t \nabla_w Q(z_t, w_t)\,.}$$

- This modification does not reduce the stochastic noise and therefore does not significantly improve the variance of w_t
- Therefore, as an optimization algorithm, stochastic gradient descent is asymptotically much slower than a typical batch algorithm.

When to use Stochastic Gradient Descent ?

- In both gradient descent (GD) and stochastic gradient descent (SGD), we update a set of parameters in an iterative manner to minimize an error function.
- In GD, it must run through **ALL** the samples in your training set to do a single update for a parameter in a particular iteration.
- In SGD, on the other hand, you use ONLY ONE or SUBSET of training sample from your training set to do the update for a parameter in a particular iteration.
- Thus, if the **number of training samples is extremely huge**, then utilizing gradient descent may take too long since changing the parameter values in each iteration involves looping through the whole training set.
- SGD, on the other hand, is quicker since it uses only one training sample and begins improving itself immediately after the first sample.
- SGD usually converges quicker than GD, however the error function is not as well reduced as in the case of GD. In most circumstances, the close approximation that SGD provides for parameter values is sufficient since they attain optimal values and continue to oscillate there.

Asymptotic analysis of various optimization algorithms

	GD	2GD	SGD	2SGD
Time per iteration :	n	n	1	1
Iterations to accuracy ρ :	$\log \frac{1}{\rho}$	$\log\log \frac{1}{\rho}$	$1/\rho$	$1/\rho$
Time to accuracy ρ :	$n \log \frac{1}{\rho}$	$n \log\log \frac{1}{\rho}$	$1/\rho$	$1/\rho$
Time to excess error ε :	$\frac{1}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}$	$\frac{1}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}$	$1/\varepsilon$	$1/\varepsilon$

- From the above table we can find that:
- Although the stochastic gradient algorithms, SGD and 2SGD, are clearly the worst optimization algorithms (third row), they need less time than the other algorithms to reach a predefined expected risk (fourth row).
- Therefore, in the large-scale setup, that is, when the limiting factor is the computing time rather than the number of examples, the stochastic learning algorithms performs asymptotically better

Recommendations for using SGD

1. Preparing the data

Randomly shuffle the training examples.

- The way we split the data into batches impact the can affect the learning
- Consider your dataset D as an ordered list and just split it into k sub list
- You will always obtain the same batches if you do not shuffle this ordered sequence before splitting, which means that if there is some information linked with the exact ordering of this sequence, it may bias the learning process. That's one of the reasons why you may want to shuffle the data.
- When examining the convergence properties of SGD, you typically expect that your samples be independent and identically distributed, and that the learning rate meet certain constraints (the Robbins–Monro conditions).If this is not the case, SGD may fail to arrive at the proper solution.
- As a result, sampling or shuffling can be useful in SGD.

Recommendations for using SGD

1. Preparing the data

Use preconditioning techniques.

- Stochastic gradient descent is a first-order algorithm and therefore suffers dramatically when it reaches an area where the Hessian is ill-conditioned.
- Fortunately, many simple preprocessing techniques like feature scaling can vastly improve the performance of SGD.

Recommendations for using SGD

2. Monitoring and debugging

Monitoring both the training cost and validation error

- We are having large training example. We can save some training instances to generate a good validation set since stochastic gradient descent is useful when training time is a main issue.
- It is critical to examine the validation error during training on a regular basis since we can stop training if the validation error has not improved in a long time.
- It is also necessary to determine the training cost on a regular basis. Since the algorithm aims to optimize the training cost, the training cost should be decreasing in general.

Recommendations for using SGD

2. Monitoring and debugging

A good training approach

- Zip once through the shuffled training set and perform the stochastic gradient descent updates.
- With an additional loop over the training set, compute the training cost. Training cost here means the criterion that the algorithm seeks to optimize.
- With an additional loop over the validation set, to compute the validation set error. Error here means the performance measure of interest, such as the classification error.
- You can also take advantage of this loop to cheaply compute other metrics.
- Computing the training cost and the validation error represent a significant computational effort because it requires additional passes over the training and validation data. But it's better than running blind.

Recommendations for using SGD

2. Monitoring and debugging

Check the gradients using finite differences

- When the gradients are slightly incorrectly computed, stochastic gradient descent often operates slowly and unpredictably. The improper technique to examine the gradients is to carefully inspect each line of the gradient computation code. Make use of finite differences:

1. Pick an example z.
2. Compute the loss Q(z;w) for the current w.
3. Compute the gradient g = ΔwQ(z;w).
4. Apply a slight perturbation w0 = w+δ. For instance, change a single weight by a small increment, or use δ =-ϒ g with small ϒ enough.
5. Compute the new loss Q(z;w’) and verify that Q(z;w’) ~ Q(z;w) + δg .

This process can be automated and should be repeated for many examples z, many perturbations δ, and many initial weights w

Recommendations for using SGD

2. Monitoring and debugging

Experiments with the learning rates Υ_t using a small sample of the training set

- If the gradients are right, the easiest technique to ascertain the proper learning rates is to conduct trials with a small but representative sample of the training set.
- Because the sample is small, conventional optimization techniques may be conducted on the same dataset to acquire a reference point and define the training cost target.
- When the algorithm does well on the small dataset's training cost, retain the same learning rates and let it operate on the whole training set.
- Expect validation performance to plateau after about the same number of epochs roughly comparable to the number of epochs needed to reach this point on the small training set.

Recommendations for training large linear models with L_2 Regularization

- The objective function of linear model with L_2 regularization is shown below

$$E_n(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(y_t w x_t)$$

- where yt = ±1, and where the function l(m) is convex. The corresponding stochastic gradient update is then obtained by approximating the derivative of the sum by the derivative of the loss with respect to a single example

$$w_{t+1} = (1 - \gamma_t \lambda) w_t - \gamma_t y_t x_t \ell'(y_t w_t x_t)$$

Recommendations for training large linear models with L_2 Regularization

1. **Sparsity**

Leverage the sparsity of the training examples $\{x_t\}$

- The training examples often are very high dimensional vectors with only a few nonzero coefficients. The stochastic gradient update is the inconvenient because it first rescales all coefficients of vector w by factor (1-ϓƛ).

$$\boxed{w_{t+1} = (1 - \gamma_t \lambda) w_t - \gamma_t y_t x_t \ell'(y_t w_t x_t)}$$

- The rest of the update, on the other hand, only concerns the weight coefficients that correspond to a nonzero coefficient in the pattern xt.
- Expressing the vector w_t as the product s_t*W_t, where s is a scalar, provides a workaround. The stochastic gradient update can then be divided into operations whose complexity scales with the number of nonzero terms in x_t:

$$\begin{aligned} g_t &= \ell'(y_t s_t W_t x_t)\,, \\ s_{t+1} &= (1 - \gamma_t \lambda) s_t\,, \\ W_{t+1} &= W_t - \gamma_t y_t g_t x_t / s_{t+1} \end{aligned}$$

Recommendations for training large linear models with L2 Regularization

2. **Learning rates**

Use learning rates of the form $\gamma_t = \gamma_0 \left(1 + \gamma_0 \lambda t\right)^{-1}$
– Determine the best γ_0 using a small training data sample.

- The most robust approach is to determine the best 0 as explained earlier, using a small sample of the training set. This is justified because the asymptotic SGD convergence rates are independent from the sample size.

- In order to make the method more robust, we can use a Υ_0 slightly smaller than the best value observed on the small training sample.

Conclusion

Stochastic gradient descent and its variations are flexible approaches that have proven helpful as a learning algorithms for large dataset.

The best advise for successfully applying these strategies is to

1. conduct small-scale trials with subsets of the training data
2. pay close attention to the validity of the gradient computation.

References

- https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/tricks-2012.pdf
- https://ai.stackexchange.com/questions/24391/should-we-also-shuffle-the-test-dataset-when-training-with-sgd
- https://leon.bottou.org/publications/pdf/mloptbook-2011.pdf

Thank you