ECE 598: Interplay between Control and Machine Learning **Fall 2020**

Lecture 4

Connection between Stochastic Optimization Methods and Feedback Systems

Lecturer: Bin Hu, Date:09/03/2020

In this lecture, we start to talk about stochastic optimization methods for the following finite-sum optimization

$$\underset{x\in\mathbb{R}^p}{\text{minimize}} \;\; f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{4.1}$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a strongly-convex objective function. We will introduce several popular finite-sum algorithms and discuss how these methods can be represented as feedback dynamical systems as shown in Figure 4.1. In the next lecture, we will discuss how the dissipation inequality approach covered in the previous lecture can be used to unify the analysis of stochastic optimization methods.

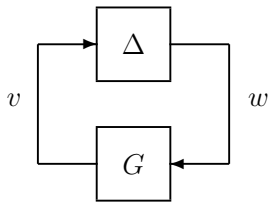

Figure 4.1. The Block-Diagram Representation for Feedback Interconnection $F_u(G, \Delta)$

4.1 Motivations: ERM and Supervised Learning

Empirical risk minimization (ERM) is a key paradigm in machine learning and naturally leads to the finite-sum optimization problem (4.1). Specifically, many supervised learning tasks, including ridge regression, logistic regression, and support vector machines, can be formulated as the following empirical risk minimization problem

$$\underset{x\in\mathbb{R}^p}{\text{minimize}}\, f(x) := \frac{1}{n}\sum_{i=1}^{n} l_i(x) + \lambda\Omega(x)$$

Here one wants to fit some prediction/classification model parameterized by x using the training data. The training set consists of n data points. The task is to fit a model x that works well for all the "unseen" data.

Interpretations for $l_i(x)$. The loss function $l_i(x)$ measures how well x performs on the i-th data point in the training set. If $l_i(x)$ is large, it means that the "loss" on the i-th data point is big and the model x works poorly on this data point. If $l_i(x)$ is small, it means the "loss" on the i-th data point is small and the model x works well on this data point. By minimizing the empirical risk $\frac{1}{n}\sum_{i=1}^n l_i(x)$, one expects the model x to work reasonably well for training data. This prevents underfitting.

Importance of $\Omega(x)$. If we allow the model to be arbitrarily complicated, we can obtain zero loss on training data but the model will work poorly on the data that have not been seen. This is called over-fitting. Roughly speaking, the difference between the model performance on the training data and the unseen data is called generalization error. One way to prevent overfitting and induce generalization is to add a regularizer $\Omega(x)$ that measures the model complexity. By adding such a term in the cost function, one expects the complexity of the resultant x is somehow controlled and hence the model x should "generalize" to the unseen data.[1] For example, one can choose $\Omega(x) = \|x\|^2$, and there exists some learning theory (e.g. stability theory) that can be used to explain how such ℓ_2-regularization induces generalization. Confining the search of x on small norm models can help generalization in many situations. Sometime $\Omega(x)$ is used to induce other desired structures. For example, the ℓ_1-regularization is typically used to induce sparsity.

What is λ? In ERM, λ is a hyperparameter which is tuned to trade off training performance and generalization. For the purpose of this course, let's say λ is a fixed positive number. In practice, λ is typically set as a small number between 10^{-8} and 0.1.

Example 1: Ridge regression. The ridge regression is formulated as an ERM problem with the following objective function

$$f(x) = \frac{1}{n}\sum_{i=1}^n (a_i^\mathsf{T} x - b_i)^2 + \frac{\lambda}{2}\|x\|^2 \tag{4.2}$$

where $a_i \in \mathbb{R}^p$ and $b_i \in R$ are data points used to fit the linear model x.

- What is this problem about? The purpose of this problem is to fit a linear relationship between a and b. One wants to predict b from a as $b = a^\mathsf{T} x$. The ridge regression gives a way to find such x based on the observed pairs of (a_i, b_i).
- Why is there a term $\frac{\lambda}{2}\|x\|^2$? Again, the term $\frac{\lambda}{2}\|x\|^2$ is just the ℓ_2-regularizer. It confines the complexity of the linear predictors you want to use. The high-level idea is that you want x to work for all (a, b), not just the observed pairs (a_i, b_i). Again, this is called generalization in machine learning. So adding such a term can induce the so-called stability and helps the predictor x to "generalize" for the data you have not seen. You need to take a machine learning course if you want to learn about generalization.

[1] What we mean is that the model x should work similarly on the training data and the unseen data.

- What is λ? λ is a hyperparameter which is tuned to trade off training performance and generalization. Again, λ is typically set as a small number between 10^{-8} and 0.1.

It is worth mentioning that f is L-smooth and m-strongly convex in this case. It is straightforward to verify that

$$f(x) = \frac{1}{2}x^\mathsf{T}\left(\frac{2}{n}\sum_{i=1}^n a_i a_i^\mathsf{T} + \lambda I\right)x - \left(\frac{2}{n}\sum_{i=1}^n b_i a_i\right)^\mathsf{T} x + \frac{1}{n}\sum_{i=1}^n b_i^2$$

which is a special case of the positive definite quadratic minimization problem. Notice $\frac{2}{n}\sum_{i=1}^n a_i a_i^\mathsf{T} + \lambda I > 0$ and hence f is m-strongly convex and L-smooth (why?). Therefore, we can apply gradient method to ridge regression, and obtain a convergence rate $\rho = 1 - \frac{1}{\kappa}$ where κ is the condition number of the positive definite matrix $\frac{2}{n}\sum_{i=1}^n a_i a_i^\mathsf{T} + \lambda I$.

Example 2: ℓ_2-Regularized Logistic regression. The ℓ_2-regularized logistic regression is formulated as an ERM problem with the following objective function

$$f(x) = \frac{1}{n}\sum_{i=1}^n \log(1 + e^{-b_i a_i^\mathsf{T} x}) + \frac{\lambda}{2}\|x\|^2 \tag{4.3}$$

where $a_i \in \mathbb{R}^p$ and $b_i \in \{-1, 1\}$ are data points used to fit the linear model x.

- What is this problem about? The purpose of this problem is to fit a linear "**classifier**" between a and b. Let's say you have collected a lot of images of cats and dogs. You augment the pixels of any such image into a vector a and wants to predict whether the image is a cat or a dog. Let's say $b = 1$ if the image is a cat, and $b = -1$ if the image is a dog. So you want to predict b based on a. You want to find x such that $b = 1$ when $a^\mathsf{T} x \geq 0$, and $b = -1$ when $a^\mathsf{T} x < 0$. The logistic regression gives a way to find such x based on the observed feature/label pairs of (a_i, b_i). You may want to take a statistics course or a machine learning course if you want to learn more about logistic regression.

- Why is there a term $\frac{\lambda}{2}\|x\|^2$? Again, the term $\frac{\lambda}{2}\|x\|^2$ is the ℓ_2-regularizer. It is used to induce generalization and help x work on all the (a, b) not just the observed data points (a_i, b_i).

The function (4.3) is also L-smooth and m-strongly convex.

Other examples. There are many other convex examples including multi-class logistic regression, support vector machines, elastic nets, and PCA. The ERM problems in deep learning involve non-convex loss functions. The optimization of deep learning has not been fully understood and it is an important research topic.

Finite-sum Structure of ERM. The ERM problem is in the form of the finite-sum optimization (4.1) where $f = \frac{1}{n}\sum_{i=1}^n f_i$. If we apply the gradient method or Nesterov's method, we need to evaluate $\nabla f = \frac{1}{n}\sum_{i=1}^n \nabla f_i$ for each iteration. In other words, we need to evaluate the gradient on all the data points. The computation cost for each iteration scales with $O(n)$. For big data applications, n is typically very large. The per iteration cost of the gradient method and Nesterov's method is high. This motivates the use of stochastic optimization methods that sample one or a small batch of data points for gradient estimate at every iteration. In stochastic optimization, the computation cost for each iteration does not depend on n and scales with $O(1)$. The hope is that there will be a lot of redundancy between data points and these stochastic methods will work well in some average sense in the long run. We will talk about various stochastic optimization methods and represent them as feedback interconnections in the next lecture.

4.2 Stochastic Optimization Methods for ERM

A classical way to solve (4.1) is the gradient method, which uses the following iteration:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \tag{4.4}$$

Since f is strongly convex, the gradient method with a well-chosen constant stepsize converges at a linear rate. To achieve accelerated convergence, we can apply Nesterov's method. Both the gradient method and Nesterov's method require computing a full gradient $\nabla f(x_k)$ at each step. Hence the iteration cost of these methods scale linearly with n. This leads to a high iteration cost when the size of the training set is large.

Consequently, stochastic optimization methods become more popular for large-scale ERM problems. Now we give a few examples.

- Stochastic gradient method: The baseline algorithm for large-scale learning tasks is the stochastic gradient (SG) method that iterates as

  $$x_{k+1} = x_k - \alpha \nabla f_{i_k}(x_k) \tag{4.5}$$

  where for each step k, the index i_k is sampled uniformly from the set $\{1, 2, \ldots, n\}$. The per iteration cost of the SG method is independent of n. At every step k, only one (or a small batch of) data point is sampled for gradient evaluation. The stochastic gradient $\nabla f_{i_k}(x_k)$ is an estimate for the true gradient $\nabla f(x_k)$. The hope is that in the long run the stochastic gradient method leads to a solution that works reasonably well in some average sense. The SG method is the most popular optimization method for large-scale learning tasks. However, one issue is that the SG method only converges linearly to some tolerance of the optimum for a well-chosen constant stepsize. Just think about that initializing the SG method from the optimal point x^* satisfying $\nabla f(x^*) = 0$. Notice $\nabla f(x^*) = 0$ does not mean $\nabla f_i(x^*) = 0$. Hence the SG method will not stay at this optimal point even if it is initialized there. The issue is that x^* is not a fixed

point for the SG method. If diminishing stepsize is used, the SG method will converge to the optimum at a sublinear rate.

- Stochastic average gradient (SAG): Compared with the stochastic gradient $\nabla f_{i_k}(x_k)$, an average gradient may be used to provide a better estimate for the true gradient. The basic idea is that one can use a vector y_k to memorize the gradient on each data point as follows

$$y_{k+1}^{(i)} := \begin{cases} \nabla f_i(x_k) & \text{if } i = i_k \\ y_k^{(i)} & \text{otherwise} \end{cases} . \tag{4.6}$$

  where at each step k, a random training example i_k is drawn uniformly from the set $\{1, 2, \ldots, n\}$. Hence, at each k, one still only samples one data point and updates $y_k^{(i)}$ for that data point. Since the vector y has memorized the gradient on all the data points, averaging y should lead to a better estimate for the full gradient ∇f. SAG uses such an average gradient and iterates as

$$x_{k+1} = x_k - \alpha \left(\frac{1}{n} \sum_{i=1}^{n} y_{k+1}^{(i)} \right) \tag{4.7}$$

  Therefore, at every step k, SAG first updates y for the average gradient evaluation and then updated x using the average gradient. With well-chosen constant stepsize α, SAG converges to the optimal solution of ERM. Why does SAG work? Intuitively, as x_k converges to the optimal point x^*, the change in x_k becomes smaller and smaller. Hence the average value $\left(\frac{1}{n} \sum_{i=1}^{n} y_{k+1}^{(i)} \right)$ approximates the true gradient better and better, and eventually converges to the true gradient. The detailed analysis for SAG is quite lengthy. This motivates the development of SAGA.

- SAGA: The idea is similar to SAG, but the update for x_{k+1} is modified as

$$x_{k+1} = x_k - \alpha \left(\nabla f_{i_k}(x_k) - y_k^{(i_k)} + \frac{1}{n} \sum_{i=1}^{n} y_k^{(i)} \right) \tag{4.8}$$

  To see the difference between SAG and SAGA, just notice SAG's update rule (4.7) can be rewritten as

$$x_{k+1} = x_k - \alpha \left(\frac{\nabla f_{i_k}(x_k) - y_k^{(i_k)}}{n} + \frac{1}{n} \sum_{i=1}^{n} y_k^{(i)} \right) \tag{4.9}$$

  Although the update rules for SAG and SAGA are similar, the convergence rate proof for SAGA is much simpler. This partially explains why SAGA gets more popular than SAG. The LMI tools in the controls field can be used to tell which method is easier to analyze at the early stage of algorithm developments. We will come back to this point later.

In this lecture, we focus on the above methods. In the next section, we will represent the above methods as feedback interconnections and comment on other stochastic methods, e.g. SVRG, Finito, and SDCA. Hopefully you will be convinced that stochastic optimization methods for ERM are just feedback dynamical systems.

4.3 Stochastic Methods as Feedback Systems

To model stochastic optimization methods as feedback systems, we need to allow either Δ or G to depend on the sampling index i_k. This leads to the following two formulations.

1. We can use an LTI system G and a stochastic perturbation Δ to form a feedback model for the SG method (and SVRG-like methods).
2. We can use a dynamical jump system G and a deterministic static nonlinearity Δ to form a feedback model for SAGA-like methods.

4.3.1 Using stochastic Δ to model SG

The SG method can be modeled as a feedback interconnection $F_u(G, \Delta)$ shown in Figure 4.1 if we choose $w = \Delta(v)$ as a stochastic nonlinear mapping $w_k = \nabla f_{i_k}(v_k)$ and set G to be the following LTI system

$$\begin{aligned} \xi_{k+1} &= \xi_k - \alpha w_k \\ v_k &= \xi_k \end{aligned}$$

To see this, we just set $\xi_k = x_k$. Then the first equation in the above LTI model becomes $x_{k+1} = x_k - \alpha w_k = x_k - \alpha \nabla f_{i_k}(v_k) = x_k - \alpha \nabla f_{i_k}(x_k)$. Notice in the modeling for the gradient method $x_{k+1} = x_k - \alpha \nabla f(x_k)$, we choose Δ as a static nonlinearity ∇f. For the SG method, the perturbation Δ depends on i_k. Therefore, it is not surprising that the convergence rate proofs for the gradient method and the SG method are quite similar. Notice that the dissipation inequality approach presented in Lecture 2 can be used to handle various types of Δ. Actually the stochastic mapping ∇f_{i_k} can also be directly handled via dissipation inequality as long as we are able to construct some informative supply rates for such a mapping.

In later lectures, we will show that standard assumptions (smoothness, convexity, etc) on f_i can be manipulated as quadratic supply rate conditions $\mathbb{E} \begin{bmatrix} \xi_k \\ w_k \end{bmatrix}^{\mathsf{T}} X \begin{bmatrix} \xi_k \\ w_k \end{bmatrix} \leq M$ with well-chosen X and M. Such supply rate conditions can be used to recover standard rate bounds for the SG method via our analysis routine.

Extensions. Many other stochastic methods can also be modeled as feedback interconnections of an LTI system G and a stochastic perturbation Δ. To handle stochastic gradient with momentum, we only need to modify the matrices (A, B, C) in the LTI model of G. To handle SVRG-like methods, we only need to modify Δ.

4.3.2 Jump system models for SAGA-like methods

SAGA and SAG can be rewritten as special cases of the following general jump system

$$\begin{aligned} \xi_{k+1} &= A_{i_k}\xi_k + B_{i_k}w_k \\ v_k &= C\xi_k \\ w_k &= \begin{bmatrix} \nabla f_1(v_k) \\ \nabla f_2(v_k) \\ \vdots \\ \nabla f_n(v_k) \end{bmatrix} \end{aligned} \tag{4.10}$$

The above general jump system model is just an interconnection of a linear jump system G and a static nonlinearity Δ that maps v to w as $w_k = \begin{bmatrix} \nabla f_1(v_k) \\ \nabla f_2(v_k) \\ \vdots \\ \nabla f_n(v_k) \end{bmatrix}$. Here, Δ depends on the gradient information of all the data points in the training set. It seems that the computation of w_k at each k requires gradient information on all the data points. However, B_{i_k} is typically sparse for SAGA-like methods. Therefore, $B_{i_k}w_k$ only involves gradient evaluation on one data point, ensuring the low per-iteration cost of SAGA-like methods.

The feedback interconnection $F_u(G, \Delta)$ provides a unified model for SAGA-like methods in the sense that we can rewrite SAG, SAGA, and many other variants in this form by properly choosing (A_{i_k}, B_{i_k}, C) for the linear jump system G. Now we show how to choose (A_{i_k}, B_{i_k}, C) for SAG and SAGA.

- **Jump system model for SAG:** First note that the gradient update rule for SAG is (4.6): $y_{k+1}^{(i)} = \nabla f_i(x_k)$ if $i = i_k$ and $y_{k+1}^{(i)} = y_k^{(i)}$ otherwise. Define the following stacked vector:

$$y_k = \begin{bmatrix} y_k^{(1)} \\ y_k^{(2)} \\ \vdots \\ y_n^{(k)} \end{bmatrix} \tag{4.11}$$

  At every step, the y information is almost unchanged except on the i_k-th data point. This can be summarized by the jump system iteration:

$$y_{k+1} = \left((I_n - e_{i_k}e_{i_k}^T) \otimes I_p\right) y_k + \left((e_{i_k}e_{i_k}^T) \otimes I_p\right) w_k \tag{4.12}$$

  where $w_k = \left[\nabla f_1(x_k)^T \cdots \nabla f_n(x_k)^T\right]$, and e_i is an n-dimensional vector whose i-th entry is 1 and other entries are 0. Here the notation "$\otimes$" denotes the Kronecker product.[2] Clearly, $e_i e_i^\mathsf{T}$ is a matrix whose (i,i)-th entry is 1 and all other entries are 0.

[2]To illustrate how Kronecker product works, just notice $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes I_p = \begin{bmatrix} aI_p & bI_p \\ cI_p & dI_p \end{bmatrix}$.

Now we can rewrite (4.9) as

$$\begin{aligned} x_{k+1} &= x_k - \frac{\alpha}{n}(e_{i_k}^T \otimes I_p)(w_k - y_k) - \frac{\alpha}{n}(e^T \otimes I_p)y_k \\ &= x_k - \frac{\alpha}{n}\left((e - e_{i_k})^T \otimes I_p\right) y_k - \frac{\alpha}{n}(e_{i_k}^T \otimes I_p)w_k \end{aligned} \tag{4.13}$$

where e is a vector whose entries are all 1. Since $v_k = x_k$, we can combine (4.12) and (4.13) to obtain the following jump system G mapping from w to v:

$$\begin{aligned} \begin{bmatrix} y_{k+1} \\ x_{k+1} \end{bmatrix} &= \begin{bmatrix} (I_n - e_{i_k}e_{i_k}^T) \otimes I_p & \tilde{0} \otimes I_p \\ -\frac{\alpha}{n}(e - e_{i_k})^T \otimes I_p & I_p \end{bmatrix} \begin{bmatrix} y_k \\ x_k \end{bmatrix} + \begin{bmatrix} (e_{i_k}e_{i_k}^T) \otimes I_p \\ (-\frac{\alpha}{n}e_{i_k}^T) \otimes I_p \end{bmatrix} w_k \\ v_k &= \begin{bmatrix} \tilde{0}^T \otimes I_p & I_p \end{bmatrix} \begin{bmatrix} y_k \\ x_k \end{bmatrix} \end{aligned} \tag{4.14}$$

Since we already have $w = \Delta(v)$, we can represent SAG as $F_u(G, \Delta)$ where G is described by the above linear jump system model. Notice in this case the state of G is $\xi_k := \begin{bmatrix} y_k \\ x_k \end{bmatrix}$.

- **Jump system model for SAGA:** Notice the update of y_k is still captured by (4.12) with $w_k = \left[\nabla f_1(x_k)^T \cdots \nabla f_n(x_k)^T\right]^T$. Now we can rewrite (4.8) as

$$\begin{aligned} x_{k+1} &= x_k - \alpha(e_{i_k}^T \otimes I_p)(w_k - y_k) - \frac{\alpha}{n}(e^T \otimes I_p)y_k \\ &= x_k - \frac{\alpha}{n}\left((e - ne_{i_k})^T \otimes I_p\right) y_k - \alpha(e_{i_k}^T \otimes I_p)w_k \end{aligned} \tag{4.15}$$

Since $v_k = x_k$, we can combine (4.12) and (4.15) to obtain the following jump system G mapping from w to v:

$$\begin{aligned} \begin{bmatrix} y_{k+1} \\ x_{k+1} \end{bmatrix} &= \begin{bmatrix} (I_n - e_{i_k}e_{i_k}^T) \otimes I_p & \tilde{0} \otimes I_p \\ -\frac{\alpha}{n}(e - ne_{i_k})^T \otimes I_p & I_p \end{bmatrix} \begin{bmatrix} y_k \\ x_k \end{bmatrix} + \begin{bmatrix} (e_{i_k}e_{i_k}^T) \otimes I_p \\ (-\alpha e_{i_k}^T) \otimes I_p \end{bmatrix} w_k \\ v_k &= \begin{bmatrix} \tilde{0}^T \otimes I_p & I_p \end{bmatrix} \begin{bmatrix} y_k \\ x_k \end{bmatrix} \end{aligned} \tag{4.16}$$

Again, we have $\xi_k = \begin{bmatrix} y_k \\ x_k \end{bmatrix}$. Putting the above model for G in a feedback loop with Δ directly realizes SAGA as a special case of (4.10).

Fixed points of the jump system models for SAG and SAGA. Suppose x^* satisfies $\nabla f(x^*) = 0$. Then we define $w^* = \left[\nabla f_1(x^*)^T \cdots \nabla f_n(x^*)^T\right]$. Next we can set $\xi^* = \left[(w^*)^T \quad (x^*)^T\right]^T$, and $v^* = x^*$. Using the fact that $\sum_{i=1}^n \nabla f_i(x^*) = 0$, we can verify that (ξ^*, w^*, v^*) provides a fixed point for the jump system model of SAG and SAGA. If ξ_k converges to ξ^*, then x_k converges to x^* and $y_k^{(i)}$ converges to $\nabla f_i(x^*)$. It SAGA and SAG are initialized at such fixed points, they are going to stay there. This partially fixes the issue of the SG method.

Extensions. Many other SAGA-like methods including Finito, SDCA, and point-SAGA can be directly modeled using the above jump system model. We only need to modify the matrices (A_{i_k}, B_{i_k}, C).

4.4 Unified analysis via Dissipation Inequality

In Lecture 3, we have presented the dissipation inequality approach as a general analysis tool for feedback systems. In today's lecture, we will discuss how to tailor the dissipation inequality approach for stochastic finite-sum methods.

Suppose G is an LTI system satisfying $\xi_{k+1} - \xi^* = A(\xi_k - \xi^*) + B(w_k - w^*)$. Suppose we know $S = \begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix}^\mathsf{T} X \begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix} \leq 0$ for any $w = \Delta(C\xi)$.[3] If there exists a positive definite matrix P s.t.

$$\begin{bmatrix} A^\mathsf{T}PA - \rho^2 P & A^\mathsf{T}PB \\ B^\mathsf{T}PA & B^\mathsf{T}PB \end{bmatrix} - X \leq 0, \tag{4.17}$$

then we have $V(\xi_{k+1}) \leq \rho^2 V(\xi_k) + S \leq \rho^2 V(\xi_k)$ where $V(\xi_k) := (\xi_k - \xi^*)^\mathsf{T} P (\xi_k - \xi^*)$. This establishes the linear convergence rate bound $\|\xi_k - \xi^*\| \leq \sqrt{\mathrm{cond}\,(P)}\rho^k \|\xi_0 - \xi^*\|$. We have already discussed how to perform such an analysis for the gradient method. To handle stochastic finite-sum methods, we only need to make some minor modification to the dissipation inequality approach. We first present the high-level ideas.

- **Interconnection of an LTI system G and stochastic Δ:** In this case, one typically will be able to construct some expected supply rate condition $\mathbb{E}S \leq M$. Then the LMI condition (4.17) can still be used to construct a (almost sure) dissipation inequality $V(\xi_{k+1}) \leq \rho^2 V(\xi_k) + S$. How to obtain a convergence bound from such a dissipation inequality? Since the supply rate condition holds in the average sense, we have to take expectation of the dissipation inequality and obtain $\mathbb{E}V(\xi_{k+1}) \leq \rho^2 \mathbb{E}V(\xi_k) + \mathbb{E}S$. Depending on what M is , this expected dissipation inequality can be used to prove various things. For example, when analyzing the stochastic gradient method for smooth strongly-convex f_i, we will figure out that M is just a constant, and the dissipation inequality can be iterated to show $\mathbb{E}V(\xi_k) \leq \rho^2 V(\xi_0) + \frac{M}{1-\rho^2}$. This just states that the stochastic gradient method converges linearly to a small ball whose size is controlled by $\frac{M}{1-\rho^2}$. Notice in this case, the supply rate is not decreasing to 0 and the total internal energy is not going to converge to 0.
- **Interconnection of a jump system G and a deterministic nonlinearity Δ:** As discussed in Lecture 3, we can use the property of Δ to construct some quadratic supply rate conditions and then analyze the feedback interconnection using the following LMI

$$\sum_{i=1}^{n} \left(p_i \begin{bmatrix} A_i^\mathsf{T}PA_i - \rho^2 P & A_i^\mathsf{T}PB_i \\ B_i^\mathsf{T}PA_i & B_i^\mathsf{T}PB_i \end{bmatrix} - X \right) \leq 0.$$

[3] Here we assume $v_k = C\xi_k$ and hence $w = \Delta(v) = \Delta(C\xi)$.

Here we assume X is independent of i_k. This type of supply rate conditions arise naturally when the matrix C in G does not depend on i_k and Δ is deterministic. The above LMI can be directly applied to SAGA-like methods.

4.4.1 Dissipation Inequality for Stochastic Gradient

Now we present a detailed analysis for the SG method under the following two assumptions:

1. f is m-strongly convex.
2. f_i is L-smooth and convex for all i.

Under these two assumptions, we can show that SGD satisfies a bound in the following form:

$$\mathbb{E}\|x_k - x^*\|^2 \leq \rho^{2k}\|x_0 - x^*\|^2 + H \tag{4.18}$$

where $\rho^2 = 1 - 2m\alpha + O(\alpha^2)$ and $H = O(\alpha)$. Here ρ^2 quantifies the convergence speed and H quantifies the accuracy. Therefore, for SGD, there is a fundamental trade-off between the convergence speed and the accuracy. If one wants a very accurate solution, one has to decrease α so that H is decreased. However, ρ^2 increases as α decreases and the convergence speed becomes slower.

As mentioned before, the supply rate condition used to prove (4.18) has a form $\mathbb{E}S \leq M$. Recall that the SG method is equivalent to a feedback system $F_u(G, \Delta)$. Here Δ is a stochastic operator mapping v to w as $w_k = \nabla f_{i_k}(v_k)$. In addition, G is governed by an LTI model with $A = I$, $B = -\alpha I$, and $C = I$. We emphasize that for the SG method we have $\xi_{k+1} - \xi^* = A(\xi_k - \xi^*) + Bw_k$ and we do not shift w_k to $(w_k - w^*)$. Again, we perform our analysis in two steps. In Step 1, we construct the supply rates. In Step 2, we solve an LMI to construct the dissipation inequality.

1. Based on $w_k = \nabla f_{i_k}(v_k)$, we can show the following inequalities:

$$\mathbb{E}\begin{bmatrix} v_k - x^* \\ w_k \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 0 & -LI \\ -LI & I \end{bmatrix} \begin{bmatrix} v_k - x^* \\ w_k \end{bmatrix} \leq \frac{2}{n}\sum_{i=1}^{n}\|\nabla f_i(x^*)\|^2 = M \tag{4.19}$$

$$\mathbb{E}\begin{bmatrix} v_k - x^* \\ w_k \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mI & -I \\ -I & 0 \end{bmatrix} \begin{bmatrix} v_k - x^* \\ w_k \end{bmatrix} \leq 0 \tag{4.20}$$

We skip the proofs here. Now we just set $X_1 = \begin{bmatrix} 0 & -LI \\ -LI & I \end{bmatrix}$ and $X_2 = \begin{bmatrix} 2mI & -I \\ -I & 0 \end{bmatrix}$. Notice that it is the first supply rate that causes the convergence issue for the SG method. Since this supply rate keeps on delivering energy to the system, the internal energy does not decrease to 0.

2. Now we test if there exists $P > 0$ and non-negative scalers (λ_1, λ_2) such that

$$\begin{bmatrix} A^\mathsf{T}PA - \rho^2 P & A^\mathsf{T}PB \\ B^\mathsf{T}PA & B^\mathsf{T}PB \end{bmatrix} - \lambda_1 X_1 - \lambda_2 X_2 \leq 0. \tag{4.21}$$

If so, we have

$$\begin{aligned} &\mathbb{E}(\xi_{k+1} - \xi^*)^\mathsf{T} P(\xi_{k+1} - \xi^*) - \rho^2 \mathbb{E}(\xi_k - \xi^*)^\mathsf{T} P(\xi_k - \xi^*) \\ \leq &\lambda_1 \mathbb{E} \begin{bmatrix} \xi_k - \xi^* \\ w_k \end{bmatrix}^\mathsf{T} X_1 \begin{bmatrix} \xi_k - \xi^* \\ w_k \end{bmatrix} + \lambda_2 \mathbb{E} \begin{bmatrix} \xi_k - \xi^* \\ w_k \end{bmatrix}^\mathsf{T} X_2 \begin{bmatrix} \xi_k - \xi^* \\ w_k \end{bmatrix} \\ \leq &\lambda_1 M \end{aligned}$$

For simplicity, we can choose $P = I$. Recall for SGD we have $A = I$ and $B = -\alpha I$. Hence (4.21) is equivalent to

$$\begin{bmatrix} 1-\rho^2 & -\alpha \\ -\alpha & \alpha^2 \end{bmatrix} - \lambda_1 \begin{bmatrix} 0 & -L \\ -L & 1 \end{bmatrix} - \lambda_2 \begin{bmatrix} 2m & -1 \\ -1 & 0 \end{bmatrix} \leq 0 \tag{4.22}$$

Now we set the left side to be a zero matrix. We have $\lambda_1 = \alpha^2$, $\lambda_2 = \alpha - \lambda_1 L$, and $\rho^2 = 1 - 2m\lambda_2 = 1 - 2m\alpha + 2mL\alpha^2$. Now the dissipation inequality leads to

$$\mathbb{E}\|x_{k+1} - x^*\|^2 \leq \rho^2 \mathbb{E}\|x_k - x^*\|^2 + \lambda_1 M$$

Iterating the above bound leads to

$$\begin{aligned} \mathbb{E}\|x_k - x^*\|^2 \leq &\rho^2 \mathbb{E}\|x_{k-1} - x^*\|^2 + \lambda_1 M \\ \leq &\rho^4 \mathbb{E}\|x_{k-1} - x^*\|^2 + (\rho^2 + 1)\lambda_1 M \\ \leq &\rho^{2k} \mathbb{E}\|x_0 - x^*\|^2 + \left(\sum_{t=0}^{\infty} \rho^{2t}\right) \lambda_1 M \\ = &\rho^{2k} \mathbb{E}\|x_0 - x^*\|^2 + \frac{\lambda_1 M}{1 - \rho^2} \end{aligned}$$

From Step 2, we have $\rho^2 = 1 - 2m\alpha + 2mL\alpha^2 = 1 - 2m\alpha + O(\alpha^2)$, and $H = \frac{\lambda_1 M}{1-\rho^2} = O(\alpha)$. This leads to the desired conclusion (4.18).

4.4.2 Dissipation Inequality for SAGA-like Methods

To show (4.10) converges to its fixed point, we are actually looking at the following iteration:

$$\begin{aligned} \xi_{k+1} - \xi^* &= A_{i_k}(\xi_k - \xi^*) + B_{i_k}(w_k - w^*) \\ v_k - v^* &= C(\xi_k - \xi^*) \\ w_k - w^* &= \begin{bmatrix} \nabla f_1(v_k) - \nabla f_1(v^*) \\ \nabla f_2(v_k) - \nabla f_2(v^*) \\ \vdots \\ \nabla f_n(v_k) - \nabla f_n(v^*) \end{bmatrix} \end{aligned} \tag{4.23}$$

Again, we can follow the two steps in the dissipation inequality framework.

1. First, we try to construct the following supply rate conditions for $j = 1, \ldots, J$.

$$\begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix}^{\mathsf{T}} X_j \begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix} \leq 0. \tag{4.24}$$

The supply rate constructions typically require using the matrix C and some properties of Δ. We will cover this in more details in next section. For now, let's look at one example. Suppose we know f_1 is L-smooth and m-strongly convex. Hence we know

$$\begin{bmatrix} v_k - v^* \\ \nabla f_1(v_k) - \nabla f_1(v^*) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ \nabla f_1(v_k) - \nabla f_1(v^*) \end{bmatrix} \leq 0. \tag{4.25}$$

Now notice we have $v_k - v^* = C(\xi_k - \xi^*)$ and $\nabla f_1(v_k) - \nabla f_1(w^*) = (e_1^{\mathsf{T}} \otimes I)(w_k - w^*)$ where e_1 is a vector whose first entry is 1 and all other entries are 0. Therefore, we have

$$\begin{bmatrix} v_k - v^* \\ \nabla f_1(v_k) - \nabla f_1(v^*) \end{bmatrix} = \begin{bmatrix} C & 0_{p\times(np)} \\ 0 & e_1^{\mathsf{T}} \otimes I \end{bmatrix} \begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix}$$

Substituting the above equation into (4.25) leads to

$$\begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} C & 0_{p\times(np)} \\ 0 & e_1^{\mathsf{T}} \otimes I \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} C & 0_{p\times(np)} \\ 0 & e_1^{\mathsf{T}} \otimes I \end{bmatrix} \begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix} \leq 0$$

Therefore, we can just choose $X_1 = \begin{bmatrix} C & 0_{p\times(np)} \\ 0 & e_1^{\mathsf{T}} \otimes I \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} C & 0_{p\times(np)} \\ 0 & e_1^{\mathsf{T}} \otimes I \end{bmatrix}$. Clearly X_1 depends on m, L, and C. You can imagine that properties of f_i and f can all be transformed into quadratic inequalities in the form of (4.24) via similar algebraic manipulations.

2. Now we can perform our LMI-based analysis. If there exists a positive definite matrix P and non-negative scalers λ_j s.t.

$$\sum_{i=1}^{n} \left(p_i \begin{bmatrix} A_i^{\mathsf{T}} P A_i - \rho^2 P & A_i^{\mathsf{T}} P B_i \\ B_i^{\mathsf{T}} P A_i & B_i^{\mathsf{T}} P B_i \end{bmatrix} \right) \leq \sum_{j=1}^{J} \lambda_j X_j,$$

then we have the expected dissipation inequality $\mathbb{E}V(\xi_{k+1}) \leq \rho^2 \mathbb{E}V(\xi_k) + \mathbb{E}S(\xi_k, w_k)$ where the storage function is defined as $V(\xi_k) = (\xi_k - \xi^*)^{\mathsf{T}} P (\xi_k - \xi^*)$ and the supply rate S is defined as

$$S(\xi_k, w_k) = \sum_{j=1}^{J} \lambda_j \begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix}^{\mathsf{T}} X_j \begin{bmatrix} \xi_k - \xi^* \\ w_k - w^* \end{bmatrix}.$$

The proof for this part is based on standard Lyapunov arguments you have seen many times. We just left and right multiply both sides of the LMI condition with

$\begin{bmatrix}(\xi_k-\xi^*)^\mathsf{T} & (w_k-w^*)^\mathsf{T}\end{bmatrix}$ and $\begin{bmatrix}\xi_k-\xi^*\\ w_k-w^*\end{bmatrix}$. This directly leads to the desired dissipation inequality. The non-negativity of λ_j guarantees $S \leq 0$ and hence we have the linear convergence bound $\mathbb{E}V(\xi_k) \leq \rho^{2k}\mathbb{E}V(\xi_0)$. For given (A_i, B_i, ρ) and X_j, out testing condition is linear in the decision variables P and λ_j, and can be solved as LMIs.

Numerically solving LMIs can be done by existing semidefinite program solvers. However, analytically solving the LMIs may require case-by-case constructions of P. The good news is that we can use the numerical solutions of LMIs to guide our constructions of analytical proofs. We will see a few examples in Homework 1.

Once we have the supply rate conditions, the constructions of dissipation inequality can be somehow routinized by solving LMIs. Now we are ready to construct supply rates for stochastic finite-sum methods. In many situations, the analysis of stochastic finite-sum methods only require simple supply rates that can be obtained by manipulating the quadratic constraints covered in the last lecture. First we will focus on SAGA-like methods. Then we will briefly discuss SVRG which is another important finite-sum method.

4.5 Supply Rates for SAGA-Like Methods

Recall that SAGA-like methods can be represented as $F_u(G, \Delta)$ where G is a jump system and the operator Δ maps v to w as

$$w_k = \begin{bmatrix}\nabla f_1(v_k)\\ \nabla f_2(v_k)\\ \vdots \\ \nabla f_n(v_k)\end{bmatrix} \tag{4.26}$$

For this operator Δ, we want to construct pointwise quadratic constraints on the input/output pair (v, w):

$$\begin{bmatrix}v_k-v^*\\ w_k-w^*\end{bmatrix}^\mathsf{T} M \begin{bmatrix}v_k-v^*\\ w_k-w^*\end{bmatrix} \leq 0, \tag{4.27}$$

where M is a symmetric matrix, and (w^*, v^*) are determined by the fixed points of the feedback interconnection $F_u(G, \Delta)$. For SAGA, we know $v^* = x^*$ and $w^* = \begin{bmatrix}\nabla f_1(x^*)^T \cdots \nabla f_n(x^*)^T\end{bmatrix}$ where $\nabla f(x^*) = \frac{1}{n}\sum_{k=1}^n \nabla f_i(x^*) = 0$.

Again, if we know $v_k - v^* = C(\xi_k - \xi^*)$, the above quadratic constraint (4.27) just gives the following supply rate condition

$$\begin{bmatrix}\xi_k-\xi^*\\ w_k-w^*\end{bmatrix}^\mathsf{T} \left(\begin{bmatrix}C & 0\\ 0 & I\end{bmatrix}^\mathsf{T} M \begin{bmatrix}C & 0\\ 0 & I\end{bmatrix}\right) \begin{bmatrix}\xi_k-\xi^*\\ w_k-w^*\end{bmatrix} \leq 0.$$

Hence we just focus on how to obtain the quadratic constraint (4.27). Various assumptions on f_i and f can be converted into inequalities in the form of (4.27). Now let's look at a few concrete examples.

- Assumption 1: f_i is L-smooth and m-strongly convex. In this case, we know

$$\begin{bmatrix} v_k - v^* \\ \nabla f_i(v_k) - \nabla f_i(v^*) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ \nabla f_i(v_k) - \nabla f_i(v^*) \end{bmatrix} \leq 0. \quad (4.28)$$

We need to make use of the following key relation:

$$w_k - w^* = \begin{bmatrix} \nabla f_1(v_k) - \nabla f_1(v^*) \\ \nabla f_2(v_k) - \nabla f_2(v^*) \\ \vdots \\ \nabla f_n(v_k) - \nabla f_n(v^*) \end{bmatrix} \quad (4.29)$$

which leads to $\nabla f_i(v_k) - \nabla f_i(v^*) = (e_i^{\mathsf{T}} \otimes I)(w_k - w^*)$ where e_i is a vector whose i-th entry is 1 and all other entries are 0. Therefore, we have

$$\begin{bmatrix} v_k - v^* \\ \nabla f_i(v_k) - \nabla f_i(v^*) \end{bmatrix} = \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^{\mathsf{T}} \otimes I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ w_k - w^* \end{bmatrix} \quad (4.30)$$

Substituting the above equation into (4.28) leads to

$$\begin{bmatrix} v_k - v^* \\ w_k - w^* \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^{\mathsf{T}} \otimes I \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^{\mathsf{T}} \otimes I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ w_k - w^* \end{bmatrix} \leq 0$$

Therefore, we can just choose $M = \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^{\mathsf{T}} \otimes I \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^{\mathsf{T}} \otimes I \end{bmatrix}$.

- Assumption 2: f is L-smooth and m-strongly convex. In this case, we know

$$\begin{bmatrix} v_k - v^* \\ \nabla f(v_k) - \nabla f(v^*) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ \nabla f(v_k) - \nabla f(v^*) \end{bmatrix} \leq 0. \quad (4.31)$$

Based on (4.29), we have $\nabla f(v_k) - \nabla f(v^*) = \frac{1}{n}(e^{\mathsf{T}} \otimes I)(w_k - w^*)$ where $e := \sum_{i=1}^n e_i$ is a vector whose entries are all 1. Therefore, we have

$$\begin{bmatrix} v_k - v^* \\ \nabla f(v_k) - \nabla f(v^*) \end{bmatrix} = \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & \frac{1}{n}e^{\mathsf{T}} \otimes I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ w_k - w^* \end{bmatrix}$$

Substituting the above equation into (4.31) leads to

$$\begin{bmatrix} v_k - v^* \\ w_k - w^* \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & \frac{1}{n}e^{\mathsf{T}} \otimes I \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & \frac{1}{n}e^{\mathsf{T}} \otimes I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ w_k - w^* \end{bmatrix} \leq 0.$$

Therefore, we can just choose $M = \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & \frac{1}{n}e^{\mathsf{T}} \otimes I \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & \frac{1}{n}e^{\mathsf{T}} \otimes I \end{bmatrix}$.

- Assumption 3: f_i is L-smooth but may not be convex. In this case, we know

$$\begin{bmatrix} v_k - v^* \\ \nabla f_i(v_k) - \nabla f_i(v^*) \end{bmatrix}^\mathsf{T} \begin{bmatrix} -L^2 I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ \nabla f_i(v_k) - \nabla f_i(v^*) \end{bmatrix} \leq 0. \tag{4.32}$$

  Similarly, we can substitute (4.30) into (4.32) and get

$$\begin{bmatrix} v_k - v^* \\ w_k - w^* \end{bmatrix}^\mathsf{T} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^\mathsf{T} \otimes I \end{bmatrix}^\mathsf{T} \begin{bmatrix} -L^2 I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^\mathsf{T} \otimes I \end{bmatrix} \begin{bmatrix} v_k - v^* \\ w_k - w^* \end{bmatrix} \leq 0$$

  Therefore, we can just choose $M = \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^\mathsf{T} \otimes I \end{bmatrix}^\mathsf{T} \begin{bmatrix} -L^2 I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & e_i^\mathsf{T} \otimes I \end{bmatrix}$.

- Assumption 4: f satisfies the "one-point convexity" condition:

$$\begin{bmatrix} v_k - x^* \\ \nabla f(v_k) \end{bmatrix}^\mathsf{T} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} v_k - x^* \\ \nabla f(v_k) \end{bmatrix} \leq 0. \tag{4.33}$$

  Notice the difference between (4.33) and (4.31) is that v^* is allowed to be any point in (4.31). Due to the facts $v^* = x^*$ and $\frac{1}{n}(e^\mathsf{T} \otimes I)w^* = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x^*) = 0$, we still have $\nabla f(v_k) - \nabla f(v^*) = \frac{1}{n}(e^\mathsf{T} \otimes I)(w_k - w^*)$. Similar to before, we can just choose $M = \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & \frac{1}{n}e^\mathsf{T} \otimes I \end{bmatrix}^\mathsf{T} \begin{bmatrix} 2mLI & -(L+m)I \\ -(L+m)I & 2I \end{bmatrix} \begin{bmatrix} I & 0_{p\times(np)} \\ 0 & \frac{1}{n}e^\mathsf{T} \otimes I \end{bmatrix}$.

How to use the above quadratic constraints? Depending on the assumptions on f_i and f, we can choose multiple M_j $(j = 1, \ldots, J)$ accordingly and formulate the following LMI

$$\sum_{i=1}^n \left(p_i \begin{bmatrix} A_i^\mathsf{T} P A_i - \rho^2 P & A_i^\mathsf{T} P B_i \\ B_i^\mathsf{T} P A_i & B_i^\mathsf{T} P B_i \end{bmatrix} \right) \leq \sum_{j=1}^J \lambda_j \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix}^\mathsf{T} M_j \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix},$$

where the positive definite matrix P and non-negative scalers λ_j are decision variables. When the assumptions on f_i and f change, typically one only needs to modify M_j accordingly. The convergence rates of SAGA and several standard finite-sum methods (SDCA, Finito, etc) can be obtained using the above quadratic constraints and LMI formulations. However, the convergence rate proof of SAG is more subtle and requires the so-called Lure-Postnikov-type Lyapunov function. We will talk about that in the next lecture.

4.6 Supply Rates for SVRG

Finally we briefly discuss SVRG that is built upon the idea of variance reduction. Originally we model the SG method as $F_u(G, \Delta)$ where Δ maps v to w as $w_k = \nabla f_{i_k}(v_k)$. We directly developed the supply rate condition for Δ and obtain some condition in the form of $\mathbb{E}S \leq C$

where C is a positive constant. A physical interpretation is that the stochastic gradient $\nabla f_{i_k}(v_k)$ keeps on supplying energy into the system and hence the system is not going to converge to its fixed point. Now we take a closer look. We can actually rewrite the SG method as

$$x_{k+1} = x_k - \alpha(\nabla f_{i_k}(x_k) - \nabla f_{i_k}(x^*)) - \alpha \nabla f_{i_k}(x^*)$$

If we choose $\xi_k = x_k$, $v_k = \xi_k$, $w_k = \begin{bmatrix} \nabla f_{i_k}(v_k) - \nabla f_{i_k}(x^*) \\ \nabla f_{i_k}(x^*) \end{bmatrix}$, $A = I$, $B = \begin{bmatrix} -\alpha I & -\alpha I \end{bmatrix}$, and $C = I$, we obtain a new feedback representation for the SG method. Now the input w_k has two entries. Actually it is trivial to construct a supply rate condition to couple the first entry of w_k with $x_k - x^*$. For example, if f_i is L-smooth and m-strongly convex, the following inequality holds in an almost sure sense

$$\begin{bmatrix} v_k - x^* \\ \nabla f_{i_k}(v_k) - \nabla f_{i_k}(x^*) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 2mLI & -(m+L)I \\ -(m+L)I & 2I \end{bmatrix} \begin{bmatrix} v_k - x^* \\ \nabla f_{i_k}(v_k) - \nabla f_{i_k}(x^*) \end{bmatrix} \leq 0. \quad (4.34)$$

Hence the first entry of w_k is not delivering energy into the system. The troublesome term is the second entry of w_k. The term $\nabla f_{i_k}(x^*)$ keeps on delivering energy into the system.

SVRG modifies the second entry of w_k as $\nabla f_{i_k}(x^*) - \nabla f_{i_k}(x_0) + \nabla f(x_0)$. Now this input depends on the initial state x_0. One will be able to obtain a supply rate condition in the form of $\mathbb{E}S \leq L\|x_0 - x^*\|^2$. SVRG is an epoch-based algorithm and at the beginning of each epoch it will update x_0 as the last (or average) iterate of the last epoch. Notice for each epoch, one needs to evaluate one full gradient $\nabla f(x_0)$. Hence the selection of the epoch length is going to affect the performance of SVRG. Within one epoch, x_0 is a fixed vector. As more epochs are run, x_0 gets closer to x^*. The supplied energy eventually decreases to 0 as x_0 converges to x^*. This is a rough physical explanation for the convergence mechanism of SVRG. The dissipation inequality approach can be applied to analyze SVRG and its accelerated variant Katyusha. We omit the details here.