Capítulo 7

Métodos de Redução de Ruído

Os estudos teóricos existentes (consultar [1] e as referencias aí incluídas) juntamente com extensa experiência computacional, levaram muitos investigadores da comunidade da aprendizagem automática a ver o método do gradiente estocástico como a abordagem de otimização ideal para aplicações de grande dimensão. Argumenta-se na literatura, no entanto, que isso está longe de estar estabelecido. O método sofre, entre outras coisas, o efeito adverso de estimativas de gradientes ruidosos (do inglês *noisy gradient estimates*). Isto impede que o método convirja para a solução quando são usados comprimentos do passo fixos, e conduz a uma lenta taxa de convergência sublinear quando é usada uma sucessão $\{\alpha_k\}$ decrescente.

Para ultrapassar esta limitação, surgiram na literatura métodos dotados com capacidade de redução de ruído. Estes métodos, que reduzem o erro nas estimativas do gradiente e/ou nos iterandos, mostraram-se eficazes na prática e gozam de propriedades teóricas atrativas.

Existem duas classes de métodos que alcançam redução de ruído de uma maneira que permite que eles possuam uma taxa de convergência linear usando um comprimento de passo fixo. A classe dos métodos de amostragem dinâmica (do inglês *dymanic sampling methods*) e a classe dos métodos de agregação do gradiente (do inglês *gradient aggregation methods*). Os métodos destas classes, não calculam simplesmente mini-batches de tamanho fixo, nem calculam gradientes completos em cada iteração. Em vez disso, substituem ou incorporam dinamicamente novas informações do gradiente, a fim de construir

um passo mais fiável com menor variância do que o do passo do método do gradiente estocástico. Apresentam-se nas secções seguintes apenas dois métodos, um de cada uma destas classes.

7.1 Métodos de amostras de tamanho dinâmico

Os *dymanic sample size methods*, conseguem redução de ruído, aumentando gradualmente o tamanho do mini-batch usado no cálculo do gradiente; usando assim, estimativas de gradiente cada vez mais precisas à medida que o processo de otimização prossegue. Apresenta-se a aqui um destes métodos.

Este método usa a equação iterativa do método do gradiente mini-batch com α fixo, e em que o tamanho do mini-batch, usado para calcular os gradientes estocásticos, cresce geometricamente em função do número de iterações.

Dado $w^{(1)} \in R^d$ e $\tau > 1$, este método é definido por:

$$w^{(k+1)} = w^{(k)} - \frac{\overline{\alpha}}{n_k} \sum_{i \in S_k} \nabla f_i(w^{(k)}) \text{ com } n_k := |S_k| = \lceil \tau^{k-1} \rceil$$

onde $\lceil . \rceil$ é a função teto (do inglês *ceil function*) que faz o arredondamento para o inteiro mais próximo superiormente.

Algoritmo: Método de amostras de tamanho dinâmico

---

1. Dar: conjunto de treino $S = \{(x_i, y_i), i = 1, \ldots, n\}$, $w^{(1)} \in R^d$, $\alpha > 0$ e $\tau > 1$
2. **para** $k = 1, 2, \ldots$ **fazer**
3. Calcula o tamanho do min-bacth $n_k = \lceil \tau^{k-1} \rceil$
4. Escolhe aleatoriamente um subconjunto de índices $S_k \subseteq \{1, \ldots, n\}$ com $|S_k| = n_k$.
5. Calcula o gradiente $\nabla F_{S_k}(w^{(k)}) = \frac{1}{n_k} \sum_{i \in S_k} \nabla f_i(w^{(k)})$
6. Calcula o novo iterando $w^{(k+1)} = w^{(k)} - \dfrac{\alpha}{n_k} \sum_{i \in S_k} \nabla f_i(w^{(k)})$
7. **fim para**

Algumas considerações de implementação:

- Na prática, é necessário encontrar um valor para o parâmetro $\tau > 1$ que gere bom desempenho do algoritmo para o problema em causa.

- Pode-se adiar a aplicação do tamanho dinâmico da amostra para evitar que o conjunto completo S seja usado cedo demais (ou preferencialmente nunca). Tais adaptações heurísticas podem ser difíceis na prática.

É de salientar, no entanto, que *dymanic sampling methods* exigem um balanço cuidadoso para se atingir a taxa de convergência linear desejada sem comprometer os custos por iteração.

7.2 Métodos de agregação do gradiente

Os *gradient aggregation methods*, melhoram a qualidade das direções de procura, guardando estimativas do gradiente calculados em iterações anteriores. Nesta abordagem, em vez de enriquecer o cálculo do gradiente com *nova* informação do gradiente em cada iteração, é possível obter uma redução da variância reutilizando gradientes calculados em passos anteriores.

7.2.1 Método SVRG

O método do gradiente estocástico de variância reduzida (do inglês *stochastic variance reduced gradient* (SVRG) *method*) em cada iteração k, com $w^{(1)} \in R^d$ dado, é definido por um ciclo externo e um ciclo interno. No início de cada iteração k do ciclo externo, está disponível $w^{(k)}$ e o algoritmo cálcula o gradiente completo $\nabla F(w^{(k)}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w^{(k)})$. A seguir, inicializa $\widetilde{w}^{(1)} \leftarrow w^{(k)}$, entra num ciclo interno indexado por j e atualiza

$$\widetilde{g}_j \leftarrow \nabla f_{i_j}(\widetilde{w}^{(j)}) - (\nabla f_{i_j}(w^{(k)}) - \nabla F(w^{(k)}))$$

onde o índice $i_j \in \{1, \ldots, n\}$ (que corresponde ao elemento amostral $(x_{i_j}, y_{i_j}) \in S$) é selecionado aleatoriamente. Uma vez que $E[\nabla f_{i_j}(w^{(k)})] = \nabla F(w^{(k)})$ para todo $i_j \in \{1, \ldots, n\}$, pode-se ver $\nabla f_{i_j}(w^{(k)}) - \nabla F(w^{(k)})$ como o viés/enviesamento no gradiente estimado $\nabla f_{i_j}(w^{(k)})$. Assim, em cada iteração, o algoritmo aleatoriamente calcula um

gradiente estocástico $\nabla f_{i_j}(\widetilde{w}^{(j)})$ no iterando interno corrente $\widetilde{w}^{(j)}$ e corrige-o com base no viés detetado. Em geral, $\widetilde{g}_j$ representa um estimador não-enviesado de $\nabla F(\widetilde{w}^{(j)})$ mas com uma variância que se espera que seja menor do que se fosse simplesmente escolhido $\widetilde{g}_j = \nabla f_{i_j}(\widetilde{w}^{(j)})$.

Apresenta-se a seguir o algoritmo geral que descreve algumas variantes do método SVRG.

Algoritmo: Métodos SVRG

---

1. Dar: conjunto de treino $S = \{(x_i, y_i), i = 1, \ldots, n\}$, $w^{(1)} \in R^d$, o comprimento do passo $\alpha > 0$, e um inteiro positivo m.
2. **para** $k = 1, 2, \ldots$ **fazer**
3. Calcula o gradiente $\nabla F(w^{(k)}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w^{(k)})$
4. Inicializa $\widetilde{w}^{(1)} = w^{(k)}$
5. **para** $j = 1, \ldots, m$ **fazer**
6. Escolhe aleatoriamente um índice i_k de $\{1, \ldots, n\}$
7. Calcula o gradiente estocástico $\widetilde{g}_j \leftarrow \nabla f_{i_j}(\widetilde{w}^{(j)}) - (\nabla f_{i_j}(w^{(k)}) - \nabla F(w^{(k)}))$
8. Atualiza o iterando $\widetilde{w}^{(j+1)} \leftarrow \widetilde{w}^{(j)} - \alpha \widetilde{g}_j$
9. **fim para**
10. Calcula o novo iterando:
11. Opção (a): $w^{(k+1)} = \widetilde{w}^{(m+1)}$
12. Opção (b): $w^{(k+1)} = \frac{1}{m} \sum_{j=1}^{m} \widetilde{w}^{(j+1)}$
13. Opção (c): escolhe aleatoriamente j de $\{1, \ldots, m\}$ e $w^{(k+1)} = \widetilde{w}^{(j+1)}$
14. **fim para**

Prova-se que este algoritmo pode atingir uma taxa de convergência linear quando aplicado a (4.2) em que F é fortemente convexa. Mais precisamente, esta taxa é atingida quando o comprimento do passo α e o comprimento do ciclo interno m, são escolhidos de modo que:

$$\frac{1}{1 - 2\alpha L}\left(\frac{1}{mc\alpha} + 2L\alpha\right) < 1$$

onde $L > 0$ é a constante de Lipschitz da função ∇F e $c > 0$ é a constante de fortemente convexa de F.

Na prática, quando estes valores não são conhecidos, estes podem ser escolhidos por experimentação.

Notar que, cada iteração k do algoritmo envolve $2m + n$ avaliações de componentes do gradiente: Passo 7 requer duas avaliações do gradiente estocástico por iteração, e o Passo 3 requer n (um gradiente completo). Portanto, uma iteração do SVRG é de maior custo computacional do que no método do gradiente estocástico, e é comparável a um com gradiente completo.

Ainda assim, na prática, o SVRG mostra ser bastante eficaz em certas aplicações comparativamente ao método do gradiente estocástico, especialmente se ambos os algoritmos são executados para um bom número de *épocas*, por exemplo da ordem das dezenas.

7.2.2 Método SAGA

O método SAGA [Ref43] tem uma estrutura semelhante ao do método do gradiente estocástico, na medida em que não opera em ciclos nem calcula o gradiente completo (exceto possivelmente no ponto inicial). Este método, em cada iteração, calcula um gradiente estocástico g_k baseado na média dos gradientes estocásticos avaliados em iterandos anteriores. Mais concretamente, na iteração k, o método tem guardado $\nabla f_i(w^{[i]})$ para todo $i \in \{1, \ldots, n\}$ onde $w^{[i]}$ representa o último iterando onde foi avaliado ∇f_i. Escolhe aleatoriamente um índice $j \in \{1, \ldots, n\}$ e o gradiente estocástico é dado por

$$g_k \leftarrow \nabla f_j(w^{(k)}) - \nabla f_j(w^{[j]}) + \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w^{[i]}). \tag{7.1}$$

Apresenta-se a seguir o algoritmo geral do método SAGA.

Algoritmo: Método SAGA

---

1. Dar: conjunto de treino $S = \{(x_i, y_i), i = 1, \ldots, n\}$, $w^{(1)} \in R^d$ e $\alpha > 0$.
2. **para** $i = 1, \ldots, n$ **fazer**
3. Calcula $\nabla f_i(w^{(1)})$
4. Guarda $\nabla f_i(w^{[i]}) \leftarrow \nabla f_i(w^{(1)})$
5. **fim para**
6. **para** $k = 1, 2, \ldots$ **fazer**
7. Escolhe aleatoriamente $j \in \{1, \ldots, n\}$
8. Calcula $\nabla f_j(w^{(k)})$
9. Calcula o gradiente estocástico $g_k \leftarrow \nabla f_j(w^{(k)}) - \nabla f_j(w^{[j]}) + \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w^{[i]})$
10. Guarda $\nabla f_j(w^{[j]}) \leftarrow \nabla f_j(w^{(k)})$

11. Calcula o novo iterando $w^{(k+1)} = w^{(k)} - \alpha g_k$
12. **fim para**

Calculando o valor esperado de g_k no que diz respeito a todas as escolhas de $j \in \{1, \ldots, n\}$, novamente tem-se que $E[g_k] = \nabla F(w^{(k)})$. Portanto, o método utiliza estimativas do gradiente não enviesadas, mas com uma variância que se espera que seja menor do que os gradientes estocásticos utilizados na rotina básica do método do gradiente estocástico. Tirando a fase inicial, este método tem os mesmos custos computacionais que um método básico de gradiente estocástico.

Em [ref43] prova-se que que este algoritmo pode atingir uma taxa de convergência linear quando aplicado a (1.2) em que F é fortemente convexa. Esta taxa é atingida com

$$\alpha = \frac{1}{2(cn + L)}$$

onde onde $L > 0$ é a constante de Lipschitz da função ∇F e $c > 0$ é a constante de fortemente convexa de F. Se c não é conhecido, é estabelecido um resultado de convergência semelhante com $\alpha = \frac{1}{3L}$.

Na prática, podem-se usar outras técnicas de inicialização mais eficazes do que calcular todos os gradientes $\{\nabla f_i\}_{i=1}^n$ no ponto inicial. Por exemplo, executar uma *época* do método do gradiente estocástico. Outra possibilidade é, à medida que os iterandos vão sendo calculados o cálculo do g_k vai usando apenas os gradientes que existem disponíveis até ao iterando corrente.

Uma desvantagem do algoritmo SAGA é a necessidade de guardar os n vetores gradientes estocásticos, o que será proibitivo em aplicações de grande dimensão. Note-se, no entanto, que se as funções f_i forem da forma $f_i(w^{(k)}) = \widehat{f}(x_i^T w^{(k)})$, então

$$\nabla f_i(w^{(k)}) = \widehat{f}\,'(x_i^T w^{(k)}) x_i.$$

Assim, quando os vetores $\{x_i\}_{i=1}^n$ já estão disponíveis em armazenamento, basta armazenar o escalar $\widehat{f}\,'(x_i^T w^{(k)})$ para se construir o $\nabla f_i(w^{(k)})$. Esta forma funcional de f_i ocorre na regressão logística e de mínimos quadrados.

7.3 Exercícios