Algorytmy statystyki praktycznej - projekt 1

Stanisław Wilczyński

March 25, 2018

1 Wstęp

Celem projektu jest sprawdzenie skuteczności i czasu działania czterech różnych metod optymalizacji:

1. Spadek po gradiencie
2. Stochastyczny spadek po gradiencie
3. SAGA [1]
4. SVRG [2]

Do porównania metod użyjemy modelu regresji logistycznej:

- Mamy n punktów $x_1, \ldots, x_n$ d-wymiarowych i odpowiadające im klasy $y_1, \ldots, y_n \in \{0, 1\}$
- Każdy y_i jest realizacją zmiennej losowej Y_i i $p_i = P(Y_i = 1|x_i)$
- Zakładamy, że $\log(\frac{p_i}{1-p_i}) = x_i\beta^T$, gdzie $\beta = (\beta_1, \ldots \beta_d)^T$ jest wektorem ukrytym

Celem dla każdego z naszych algorytmów jest wyznaczenie optymalnych (w sensie funkcji wiarygodności) p_i. Oczywiście jest to równoważne z wyznaczeniem optymalnego wektora β. Będziemy więc szukać wektora β minimalizującego -logarytm z funkcji wiarygodności:

$$f(\beta) = -\log(L(\beta)) = \sum_{i=1}^{n} \log\left(1 + \exp(x_i\beta^T)\right) - y_i x_i \beta^T = \sum_{i=1}^{n} f_i(\beta)$$

2 Opis metod

W tej sekcji krótko opiszemy użyte porównane metody.

1. Spadek po gradiencie (GD) - ten algorytm opiera się na bardzo prostej zasadzie. W każdej iteracji obliczamy gradient funkcji kary (f) względem β i zmieniamy ten wektor o nieduże wartości w kierunku przeciwnym do gradientu. Dzięki temu w kolejnych krokach algorytmu powinniśmy otrzymywać coraz mniejsze wartości kary (jeśli stała uczenia γ nie jest zbyt duża). Problemem spadku po gradiencie jest kosztowność każdej iteracji algorytmu - dla dużych n w każdym kroku musimy policzyć wartości wszystkich f_i. W celu pozbycia się problemu dużej złożoności obliczeniowej została zaproponowana poniższa metoda.

2. Stochastyczny spadek po gradiencie (SGD) - jest to prosta modyfikacja poprzedniego algorytmu. Zamiast poruszać się w kierunku przeciwnym do gradientu f w każdej iteracji losowo wybieramy indeks i i zmieniamy wartości wektora β przeciwnie do kierunku gradientu f_i. Ze względu na dużą liczbę iteracji oczekujemy, że będziemy się poruszać jak w przypadku GD (przy dużej liczbie iteracji każde i zostanie wybrane mniej więcej tyle samo razy). Problemem tego algorytmu jest niestety losowość związana z wyborem i-tej funkcji w każdym kroku algorytmu. Zgodnie z [2] taki losowy wybór powoduje, że w każdym kroku algorytmu nie poruszamy się w odpowiednim kierunku (zgodnie z gradientem całej funkcji kary) i problemy ze zbieżnością algorytmu. Następne dwie metody próbują radzić sobie z tym problemem, redukując wariancję (odstąpienia od właściwego kierunku) przy wykonywaniu kroku gradientowego.

3. SAGA, SVRG (stochastic variance reduced gradient) - idea obu tych metod jest bardzo podobna. W celu redukcji wspomnianej wariancji dla SGD przy wykonywaniu kroku gradientowego będziemy też brać pod uwagę średnią z gradientów dla wszystkich f_i (niekoniecznie dla aktualnej wartości wektora β). Co ważne dzięki odpowiedniej implementacji taka zmiana powoduje bardzo niewielki narzut obliczeniowy w stosunku do SGD. Główną różnicą między SAGA, a SVRG jest co ile kroków algorytmu uaktualniamy nasz średni gradient - w przypadku SAGA dzieje się to w każdym kroku, w przypadku SVRG dzieje się to co T(parametr algorytmu) kroków.

3 Wyniki

W tej sekcji prezentujemy otrzymane wyniki dla trzech zadań: skuteczności metod, czasu działania metod, rezultaty dla rzadkiego wektora β przy stosowaniu kary uwzględniającej pierwszą normę naszego wektora.

3.1 Skuteczność

W zadaniu kazano porównać metody również ze względu na sposób skorelowania danych $x_1, \ldots, x_n$:

- niezależne $\Sigma = I_n$
- parami tak samo skolerowane $\sigma_{ii} = 1, \sigma ij = \rho$ dla $i < j$ i $\rho < 1$
- autoskorelowane - $\sigma_{ij} = \rho^{|i-j|}$

Co więcej, porównywanie efektywności naszych metod ma sens tylko wtedy, gdy dla każdej z nich znajdziemy optymalną stałą uczenia (γ). W przeciwnym wypadku nie bylibyśmy w stanie stwierdzić, która metoda jest lepsza. W związku z czasochłonnymi obliczeniami z tym związanymi i trzema różnymi przypadkami generowania danych, w tej sekcji użyjemy tylko dwóch różnych rozmiarów zbioru danych $n = 10^4, d = 10$ oraz $n = 1000, d = 5$. Aby znaleźć optymalną stałą uczenia sprawdzamy 10 równo rozłożonych logarytmicznie wartości na przedziale $[10^{-5}, 10^{-2}]$. Taki przedział został wybrany, aby obniżyć szansę na błędy numeryczne spowodowane zbyt dużą stałą uczenia (powodowałaby ona bardzo dużą niestabilność kary po kolejnych iteracjach naszego algorytmu). Poniżej przedstawiamy wykresy kar względem liczby iteracji w każdym z trzech przypadków dla naszych algorytmów. Kryterium stopu była zarówno liczba iteracji (maksymalnie 5000), jak i średnia wartości bezwzględna różnicy między współrzędnymi wektora β w dwóch kolejnych iteracjach - jeśli mniejsza niż 10^{-7} przerywamy działanie algorytmu.

Figure 1: Wyniki dla $n = 10000, d = 10$

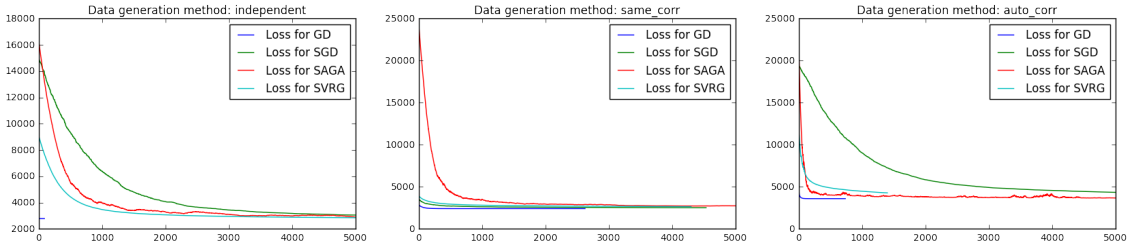
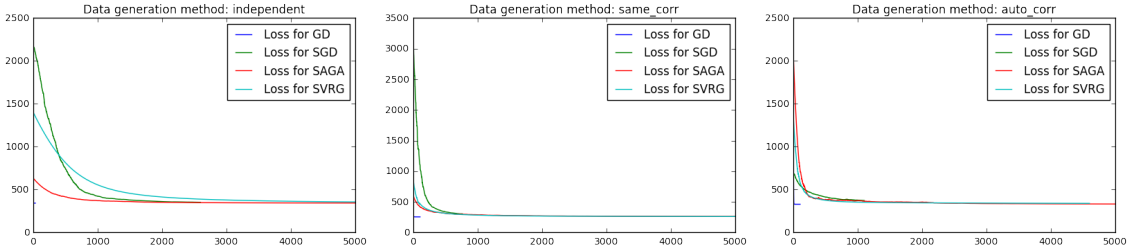


Figure 2: Wyniki dla $n = 1000, d = 5$

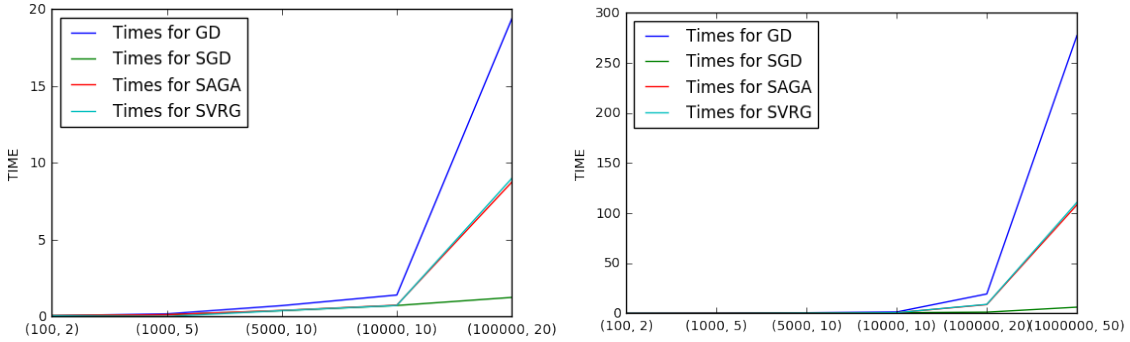


Widzimy, że rozmiar danych nie powoduje zbyt dużej różnicy względnych wyników między metodami - wspólną cechą jest szybsza zbieżność dla mniejszych danych. Co więcej GD zbiega zdecydowania szybciej niż wszystkie pozostałe metody i praktycznie w każdym wypadku osiąga najlepszy wynik. Jest to spodziewany rezultat, gdyż ten algorytm najlepiej optymalizuje funkcję kary - w końcu zawsze podążamy w idealnym kierunku spadku funkcji. Co do pozostałych 3 metod otrzymane wyniki są bardzo podobne. Na wykresach jest to słabo widoczne, ale w załączonym do raportu pliku html możemy zobaczyć, że stochastyczny gradient wypada najgorzej, natomiast SAGA i SVRG działają lepiej na zmianę. Jeśli chodzi o błąd kwadratowy przybliżenia wektora β (suma kwadratów różnic między prawdziwym, a wyestymowanym po współrzędnych), możemy zauważyć, że dla GD jest on rzędu dziesiątych lub setnych natomiast dla pozostałych metod rzędu jedności, a więc różnica jest znaczna na korzyść GD. Biorąc pod uwagę to kryterium SAGA wydaje się być troszeczkę lepsza od SVRG i SGD (również widoczne w pliku html). Jeśli chodzi o sposób generowania danych nie widać, żeby miał on wpływ na wyniki. Możemy jednak zauważyć, że dla drugiego sposobu generowania danych zbieżność wszystkich metod jest wyraźnie najszybsza.

3.2 Czas działania

Aby porównać czas działania algorytmów rozpatrujemy następujące wymiary zbioru danych: $(n, d) = (100, 2), (1000, 5), (5000, 10), (10^4, 10), (10^5, 20), (10^6, 50)$. W tym wypadku pomijamy już szukanie najlepszych parametrów dla każdego z algorytmów i ustawiliśmy stałą uczenia na 0.00001. Wyniki są zaprezentowane na poniższym wykresie (czas jest podany w sekundach).

Figure 3: Czas działania metod dla rosnących wartości n i d



Na wykresie widzimy, że zgodnie z oczekiwaniami GD działał najwolniej i zwiększając rozmiar danych, różnica między nim a pozostałymi metodami rośnie bardzo szybko. SAGA i SVRG działają podobnie długo, a SGD jest oczywiście wyraźnie najszybszy, jednak jest to okupione najmniejszą dokładnością wyników, o czym wspominaliśmy w poprzedniej sekcji.

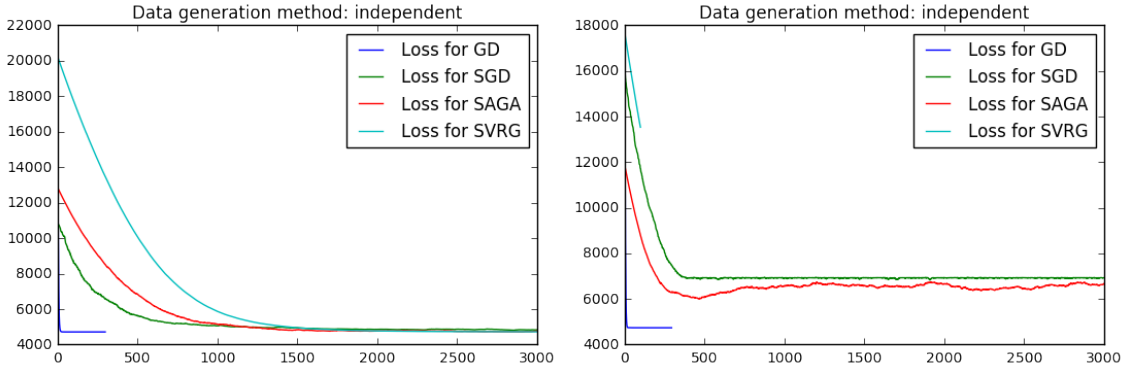
3.3 Rzadka β

W tej części optymalizujemy starą funkcję kary z dodanym składnikiem $\lambda \sum_{i=1}^{d} |\beta_i|$(kara LASSO). Aby stosować jakąkolwiek wersję spadku po gradiencie dla takiej nie różniczkowalnej funkcji po każdej iteracji na otrzymanym nowym wektorze β wykonujemy operację proximal:

$$\beta = sign(\beta)(|\beta| - \gamma\lambda)_+$$

Tym razem nasz wektor β będzie rzadki, tzn. 87.5% współrzędnych będzie zerami. Jeśli teraz podobnie jak w przypadku badania skuteczności zaczniemy wyszukiwać najlepsze parametry (względem nowej kary) dla naszych metod (γ i λ) to otrzymamy wyniki jak na wykresie po niżej po lewej stronie. Niestety, mimo niezerowych wartości lambd dla każdego z algorytmów otrzymane wektory bet miały wszystkie współrzędne niezerowe. Wobec tego, żeby otrzymać lepszą estymację naszego wektora β musimy zdecydowanie zwiększyć parametr λ dla każdego z algorytmów. Aby otrzymać pożądany efekt musieliśmy pomnożyć znalezione wartości nawet 1000-krotnie (dokładne wartości lambd można zobaczyć w pliku html). Rezultaty przy nowych wartościach lambd są widoczne na wykresie poniżej po prawej stronie.

Figure 4: Wyniki działania metody z uwzględnieniem kary LASSO



O ile dla GD otrzymaliśmy bardzo dobrą estymację (zerowe współrzędne wyjściowego wektora są wyzerowane również w estymatorze), a co za tym idzie nie dużo gorszy wynik (w sensie sumy kwadratów błędów od oryginalnego wektora), o tyle dla pozostałych algorytmów wyniki nie są satysfakcjonujące. Przy wielokrotnym uruchomieniu dla tych samych parametrów otrzymujemy drastycznie różne liczby niezerowych współrzędnych (zdarza się zarówno 0, jak i 10). Również wyniki są dużo gorsze niż dla optymalnych wartości parametrów. Wynika to z losowości, która jest nieodłączną częścią tych algorytmów. W związku z tym możemy stwierdzić, że te metody nie są wystarczająco stabilne, żeby ich używać do znajdowania rzadkich estymatorów za pomocą kary LASSO.

4 Podsumowanie

Na podstawie przeprowadzonych symulacji możemy stwierdzić, że spadek po gradiencie jest najskuteczniejszą z testowanych metod, jednak jak pokazaliśmy dla dużych rozmiarów danych jego czas działania bardzo ogranicza jego potencjalne zastosowania. Z pozostałych metod SAGA i SVRG osiągały podobne rezultaty, lepsze od SGD. Jeśli chodzi o problem estymowania rzadkich wektorów β to widzimy, że metody stochastyczne oparte na losowości nie sprawdzają się i tylko GD potrafi dobrze odwzorować wyjściowy wektor.

References

[1] Aaron Defazio, Francis R. Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *CoRR*, abs/1407.0202, 2014.

[2] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 315–323. Curran Associates, Inc., 2013.