Metody numeryczne

12. Minimalizacja: funkcje wielu zmiennych

Daleko od minimum

P. F. Góra

https://zfs.fais.uj.edu.pl/pawel_gora

9 stycznia 2024

Minimalizacja funkcji wielu — niekiedy **bardzo** wielu — zmiennych jest jednym z podstawowych zastosowań metod numerycznych. Zarazem może to być problem o wielkim koszcie numerycznym. Dlatego, w zależności od okoliczności, stosuje się dwie różne strategie:

- Jeśli jesteśmy daleko od poszukiwanego minimum, szybko podążamy w jego kierunku, nie starając się jednak znaleźć dokładnego położenia minimum. Jest to strategia szczególnie często stosowana w problemach o dużej wymiarowości.
- Jeśli jesteśmy blisko poszukiwanego minimum i zależy nam na jego dokładnym zlokalizowaniu, staramy się o taką dokładną lokalizację, wiedząc, że może ona być kosztowna nawet w problemach kilkuwymiarowych.

W całym wykładzie rozważamy funkcje $f : \mathbb{R}^N \to \mathbb{R}$, odpowiednio wiele razy różniczkowalne.

Podstawową zasadą algorytmów minimalizacyjnych jest, że idziemy zawsze “w dół”, w stronę malejących wartości funkcji. To sprawia, że minimalizacja jest numerycznie prostsza od ogólnego problemu rozwiązywania układów równań algebraicznych (formalnie, matematycznie, znalezienie minimum oznacza rozwiązanie równania $\nabla f = 0$), ale oznacza też, że przedstawione tu algorytmy znajdują jedynie minimum lokalne.

Daleko od minimum: metoda najszybszego spadku

Daleko od minimum staramy się wyłącznie o to, aby w kolejnych krokach naszej procedury wartości funkcji malały. Ponieważ gradient funkcji wielu zmiennych pokazuje kierunek jej najszybszego wzrostu, kierunek przeciwny pokazuje kierunek jej najszybszego spadku. A ponieważ zależy nam, aby funkcja malała tak szybko, jak to możliwe, podążamy w tym właśnie kierunku. Otrzymujemy w ten sposób algorytm najszybszego spadku (ang. *steepest descent* lub *gradient descent*):

Niech aktualne położenie wynosi $\mathbf{x}_k$, $f_k = f(\mathbf{x}_k)$.

1. $$\mathbf{p}_k = -\nabla f(\mathbf{x}_k) \tag{1a}$$
2. $\gamma = \gamma_{\text{start}}$
3. $$\tilde{\mathbf{x}} = \mathbf{x}_k + \gamma \mathbf{p}_k \tag{1b}$$
   $$\tilde{f} = f(\tilde{\mathbf{x}}) \tag{1c}$$
4. Jeżeli $\tilde{f} < f_k$
   - $\mathbf{x}_{k+1} = \tilde{\mathbf{x}}$, $f_{k+1} = \tilde{f}$
   - $k = k + 1$
   - `goto 1`
5. Jeżeli $\tilde{f} \geqslant f_k$
   - opcjonalnie: zmniejsz γ, potem `goto 3`
   - `STOP`

Ta metoda jest nieprecyzyjna, nie dostarcza kryterium czy jesteśmy blisko minimum i w zamyśle służy do szybkiego przemieszczenia się w okolice minimum. Często — zwłaszcza gdy ciąg kolejnych wartości f_k maleje *coraz wolniej*, co interpretujemy w ten sposób, że krajobraz się "wypłaszcza" — jest to *jedyna* metoda, jakiej się w praktyce używa. Należy jednak pamiętać, że jej wyniki na ogół są nieprecyzyjne i jeżeli zależy nam na lepszym, bardziej dokładnym zlokalizowaniu minimum, powinniśmy użyć jakichś innych metod, przynajmniej w końcowej fazie minimalizacji, pod warunkiem, że nie będzie to zbyt kosztowne ze względu na wymiarowość problemu.

Współczynnik γ na ogół musi być dość mały, gdyż przy dużych wartościach gradientu ryzykujemy, że "przeskoczymy" poszukiwane minimum.

Z punktu widzenia Big Data...

W ciągu ostatnich kilkunastu lat rozmiary dostępnych zbiorów danych rosną *szybciej* niż rośnie prędkość procesorów (nawet po uwzględnieniu paralelizacji i obliczeń na GPU). Z tego punktu widzenia możliwości uczenia maszynowego są ograniczone raczej przez możliwości obliczeniowe niż przez dostępne zbiory danych. Tę klasę problemów zwyczajowo określa się jako *Big Data*.

Uczenie maszynowe bardzo często oznacza konieczność minimalizowania funkcji

$$Q(\mathbf{X}) = \frac{1}{N} \sum_{i=1}^{N} Q_i(\mathbf{X}; x_i, y_i) \tag{2}$$

gdzie

- $\mathbf{X}$ jest wielowymiarową zmienną, po której minimalizujemy — wektorem estymatorów, których wartości chcemy znaleźć. Tego typu problemy pojawiają się w zagadnieniu najmniejszych kwadratów.
- $\forall i: Q_i \geqslant 0$,
- Q_i mają taką samą postać funkcyjną, różnią się tylko elementami x_i, y_i (elementami zbioru uczącego),
- $N \gg 1$ (N jest zazwyczaj **BARDZO** duże, rzędu kilkudziesięciu, niekiedy kilkuset tysięcy).

Problem (2) można rozwiązać za pomocą metody najszybszego spadku, którejś z bardziej precyzyjnych metod minimalizacji lub, jeżeli mowa o liniowym zagadnieniu najmniejszych kwadratów, za pomocą przybliżonego rozwiązywania nadokreślonych układów równań przy pomocy SVD. Jednak dla bardzo dużych N, czyli dla bardzo dużych zbiorów uczących, inne podejście może być bardziej efektywne.

Punktem wyjścia jest metoda najszybszego spadku: W każdym kroku podążamy w kierunku minus gradientu funkcji $Q(\mathbf{X})$, teoretycznie aż do osiągnięcia minimum kierunkowego*, czyli minimum w kierunku aktualnego gradientu.

Jak jednak pamiętamy, daleko od minimum nie minimalizujemy, tylko podążamy z arbitralnie dobranym krokiem w kierunku ujemnego gradientu.

*Patrz następny wykład

Dlatego zamiast minimalizować, w n-tym kroku iteracji przyjmujemy

$$\begin{aligned} \mathbf{X}_{n+1} &= \mathbf{X}_n - \gamma \cdot \nabla_{\mathbf{X}} Q(\mathbf{X})|_{\mathbf{X}_n} \\ &= \mathbf{X}_n - \gamma \cdot \frac{1}{N} \sum_{i=1}^{N} \nabla_{\mathbf{X}} Q_i(\mathbf{X}; x_i, y_i)|_{\mathbf{X}_n} \ . \end{aligned} \tag{3}$$

$\gamma =$ const i jest nazywane *prędkością uczenia* (ang. *learning rate*).

Uwaga: Przyjęcie stałego kroku nie jest aż tak naiwne, jak by się to mogło wydawać. Jeśli rozpatrzymy układ równań różniczkowych typu

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}) \tag{4}$$

to najprostszą (zdecydowanie nie najlepszą, ale najprostszą i najszybszą) metodą jego numerycznego rozwiązywania jest *metoda Eulera*

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h \cdot \mathbf{f}(\mathbf{x}_n)\,, \tag{5}$$

gdzie h jest stałym krokiem narastania zmiennej niezależnej x. Metoda najszybszego spadku (3) ze stałym krokiem γ ma taką samą postać, co metoda Eulera, po utożsamieniu $\mathbf{f} \equiv -\nabla Q$.

Kryterium stopu

Jakie kryterium stopu przyjąć dla algorytmu najszybszego spadku (3)? Sensowne wydają się dwie opcje:

(i). Przesunięcie jest dostatecznie małe:

$$\begin{aligned} \left\| \mathbf{X}_{n+1} - \mathbf{X}_n \right\| &< \varepsilon_1 \\ \gamma \cdot \left\| \nabla_{\mathbf{X}} Q(\mathbf{X})|_{\mathbf{X}_n} \right\| &< \varepsilon_1 \\ \left\| \nabla_{\mathbf{X}} Q(\mathbf{X})|_{\mathbf{X}_n} \right\| &< \frac{\varepsilon_1}{\gamma} \end{aligned} \tag{6}$$

(ii). Zmiana wartości funkcji jest dostatecznie mała:

$$\begin{aligned} \left|Q(\mathbf{X}_{n+1}) - Q(\mathbf{X}_n)\right| &< \varepsilon_2 \\ \left|Q(\mathbf{X}_n) + \left(\mathbf{X}_{n+1} - \mathbf{X}_n\right)^T \nabla_{\mathbf{X}} Q(\mathbf{X})|_{\mathbf{X}_n} - Q(\mathbf{X}_n)\right| &< \varepsilon_2 \end{aligned} \quad (7)$$

gdzie skorzystaliśmy z rozwinięcia Taylora do pierwszego rzędu. Ponieważ

$$\left|\left(\mathbf{X}_{n+1} - \mathbf{X}_n\right)^T \nabla_{\mathbf{X}} Q(\mathbf{X})|_{\mathbf{X}_n}\right| \leqslant \left\|\mathbf{X}_{n+1} - \mathbf{X}_n\right\| \cdot \left\|\nabla_{\mathbf{X}} Q(\mathbf{X})|_{\mathbf{X}_n}\right\| \quad (8)$$

to jeżeli wielkość po prawej stronie (8) jest mniejsza od ε_2, to także zmiana wartości funkcji jest mniejsza. Z kolei

$$\left\|\mathbf{X}_{n+1} - \mathbf{X}_n\right\| \cdot \left\|\nabla_{\mathbf{X}} Q(\mathbf{X})|_{\mathbf{X}_n}\right\| < \varepsilon_2 \quad (9)$$

jeśli przesunięcie jest dostatecznie małe lub gdy gradient jest dostatecznie mały. Stwierdziliśmy jednak, że gdy dostatecznie mały jest gradient, małe jest też przesunięcie.

Zatem jako kryterium stopu przyjmujemy

$$\left\| \nabla_{\mathbf{X}} Q(\mathbf{X})|_{\mathbf{X}_n} \right\| < \varepsilon \,. \tag{10}$$

Metodę najszybszego spadku zatrzymujemy, gdy gradient funkcji jest dostatecznie mały — funkcja "wypłaszcza się". Przyjęcie takiego kryterium dodatkowo uwalnia nas od konieczności obliczania wartości funkcji w każdym kroku.

Stochastic Gradient Descent

Jeśli w (3) $N \gg 1$, czas obliczania sumy gradientów cząstkowych $\sum_i^N \nabla Q_i$ może być bardzo znaczny. Zarazem wyrażenie $\frac{1}{N}\sum_i^N \nabla Q_i$ ma postać średniego gradientu cząstkowego. Jeśli założymy, że poszczególne elementy tej sumy nie odbiegają zbytnio od średniej, możemy średni, kosztowny w wyliczaniu gradient, zastąpić *losowo wybranym* gradientem cząstkowym. Otrzymujemy zatem algorytm *Stochastic Gradient Descent*:

- wybierz losowo $k \in \{1, 2, \ldots, N\}$
- przyjmij

$$\mathbf{X}_{n+1} = \mathbf{X}_n - \gamma \, \nabla_{\mathbf{X}} Q_k(\mathbf{X}; x_k, y_k)|_{\mathbf{X}_n} \tag{11}$$

- Dodatkowo akceptujemy powyższy krok tylko jeżeli $Q(\mathbf{X}_{n+1}) < Q(\mathbf{X}_n)$. Jeśli ta nierówność nie zachodzi, nie wykonujemy kroku, ale losujemy nowe k. Ten wariant nazwiemy *Stochastic Gradient Descent z ograniczeniem*.

Iterację kończymy albo po wykonaniu z góry określonej liczby kroków, albo — częściej — gdy spełnione zostanie kryterium (10), gdzie zamiast pełnego gradientu bierzemy wylosowany gradient cząstkowy. Rodzi to jednak niebezpieczeństwo, że wylosowany gradient cząstkowy przypadkowo będzie mały, podczas gdy cały gradient — niekoniecznie. Bezpieczne byłoby kryterium, że zmiana funkcji jest dostatecznie mała:

$$\left|Q(\mathbf{X}_n) - Q(\mathbf{X}_{n+1})\right| < \varepsilon\,, \tag{12}$$

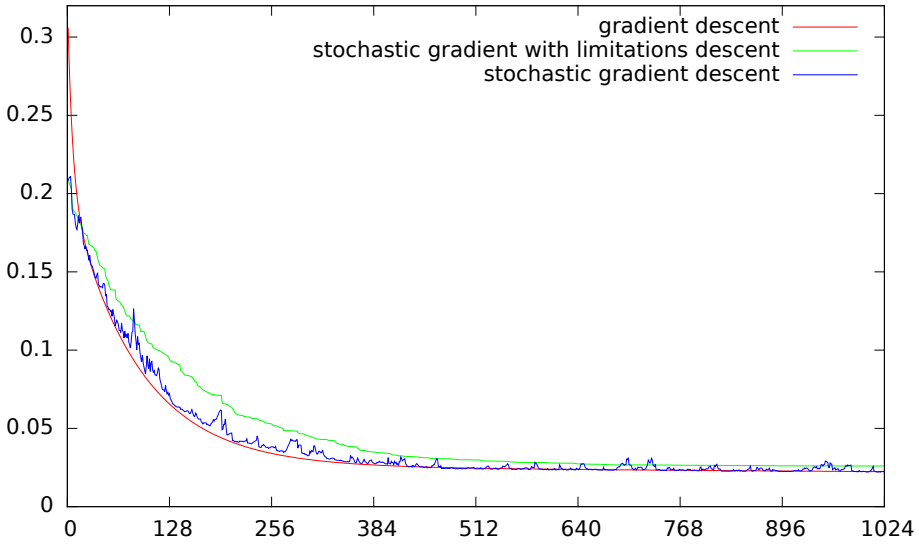
gdzie ε jest ustaloną tolerancją. Powoduje to jednak konieczność obliczania wartości Q w każdym kroku, co z kolei może być dużym obciążeniem, gdy liczba składników w (3) jest bardzo duża.

Przy zastosowaniu wariantu “z ograniczeniem” i tak musimy korzystać z kryterium (12).

Wartość *prędkości uczenia* γ oraz tolerancję ε dobieramy "eksperymentalnie", to znaczy kierując się znajomością natury problemu i doświadczeniem uzyskanym przy rozwiązywaniu podobnych problemów.

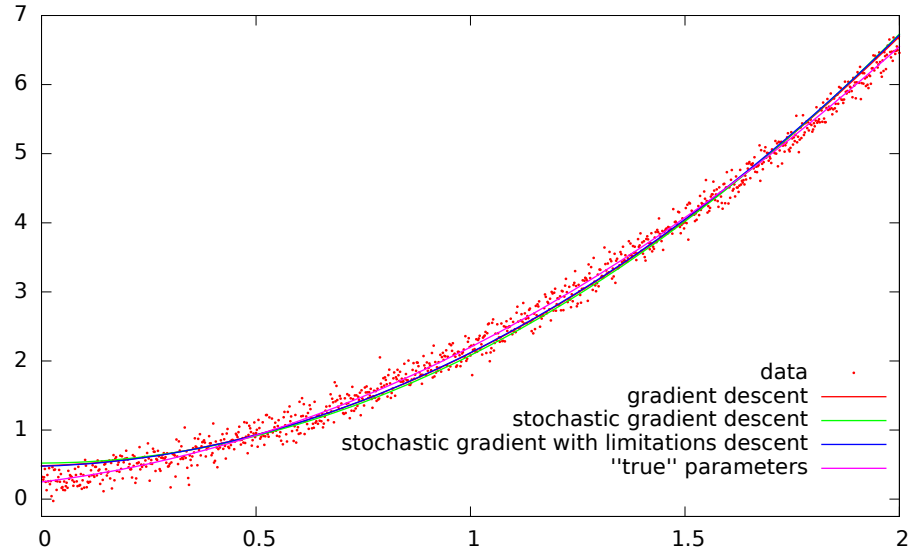
Okazuje się, że metoda *Stochastic Gradient Descent* działa zdumiewająco dobrze dla bardzo dużych zbiorów uczących. Co ciekawe, przyjęcie warunku, iż wartość funkcji Q musi maleć po wykonaniu kroku na ogół *pogarsza* wyniki: Od czasu do czasu można/trzeba wykonać krok "w złą stronę".

Przykład



Wygenerowano $N = 1024$ punktów $y_i = ax_i^2 + bx_i + c + \xi_i$, gdzie $\{\xi\}_{i=1}^N$ są liczbami o rozkładzie normalnym, natomiast $x_i \in [0, 2]$. Rysunek przedstawia kolejne wartości funkcji $Q(a, b, c)$ uzyskane w metodzie najszybszego spadku (czerwony), w metodzie *Stochastic Gradient Descent* (niebieski) oraz *Stochastic Gradient Descent* z ograniczeniami (zielony). $\gamma = 1/128$. W przypadku *Stochastic Gradient Descent* wartość minimalizowanej funkcji niekiedy rośnie, ale po dostatecznie dużej liczbie iteracji wyniki tej metody

są nieco *lepsze*, niż dla metody z ograniczeniem. Co ważniejsze, wyniki są porównywalne z wynikami uzyskanymi w obliczeniowo droższej metodzie najszybszego spadku.



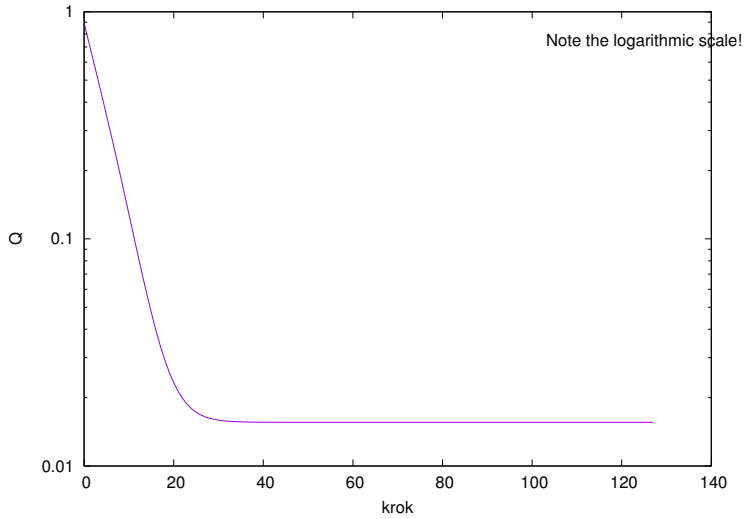
Dopasowane krzywe są niemalże nieodróżnialne w obszarze odpowiadającym danym wejściowym (zawartości zbioru uczącego).

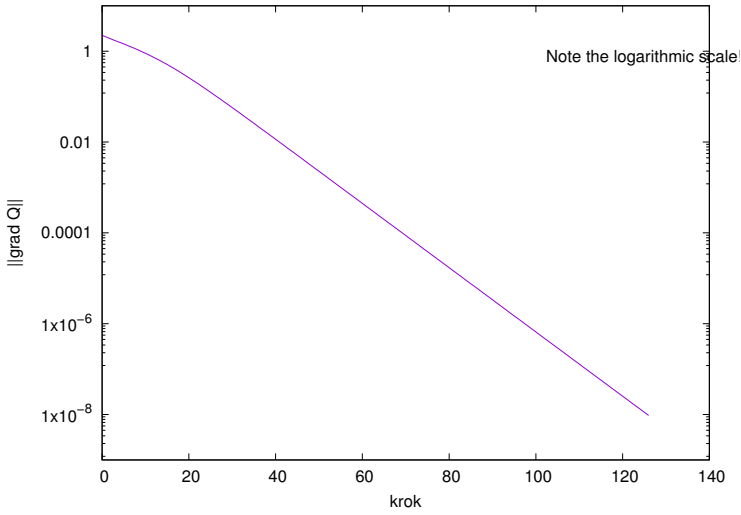
Przykład

Wygenerowano $N = 1024$ punktów $y_i = \left(ae^{-x_i} + be^{x_i}\right)/(a+b) + \sigma\xi_i$, gdzie, jak wyżej, $\{\xi\}_{i=1}^{N}$ są liczbami o rozkładzie normalnym, natomiast $x_i \in [-2, 2]$. $\sigma = 0.125$. Zagadnienie dopasowania krzywej o zadanej postaci funkcyjnej do punktów danych jest przykładem *nieliowego*† zagadnienia najmniejszych kwadratów.

Przyjmując $\gamma = 1/32$, przeprowadzono minimalizację (a tak naprawdę procedrę podążania w stronę minimum) metodą najszybszego spadku.

†Jest to słaba nieliniowość, model $y = \alpha e^{-x} + \beta e^{x}$, liniowy w parametrach, dałby praktycznie te same wyniki. 4

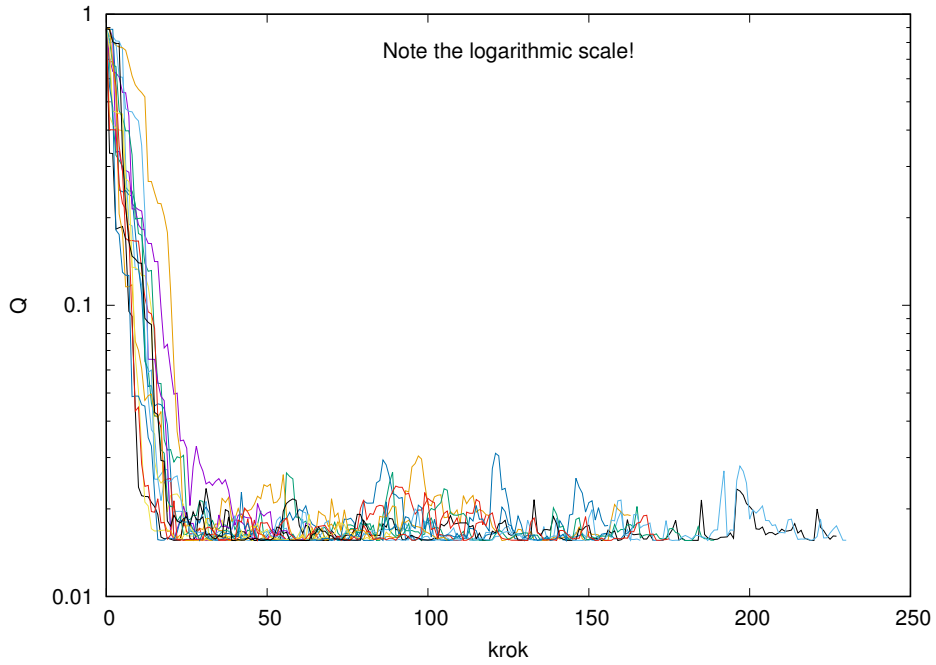




Stochastic Gradient Descent with Resetting

Aby uniknąć problemów wynikających, z jednej strony, z przypadkowego przedwczesnego zatrzymaniem procedury Stochastic Gradient Descent (SDG), z drugiej, z faktu, że trajektoria stochastyczna może zacząć *oddalać* się od minimum, powtarzamy SGD wielokronie, zatrzymując przebieg *albo* gdy (cząstkowy) gradient spadnie poniżej progu, *albo* gdy wykonamy ustaloną z góry, lecz losową liczbę kroków; okazuje się, że liczbę kroków należy losować z rozkładu Poissona. Po zatrzymaniu, proces *restarujemy* z tymi samymi warunkami początkowymi. Zapamiętujemy parametry końcowe osiągnięte w każdej trajektorii. Na koniec wybieramy parametry odpowiadające najmniejszej wartości Q.

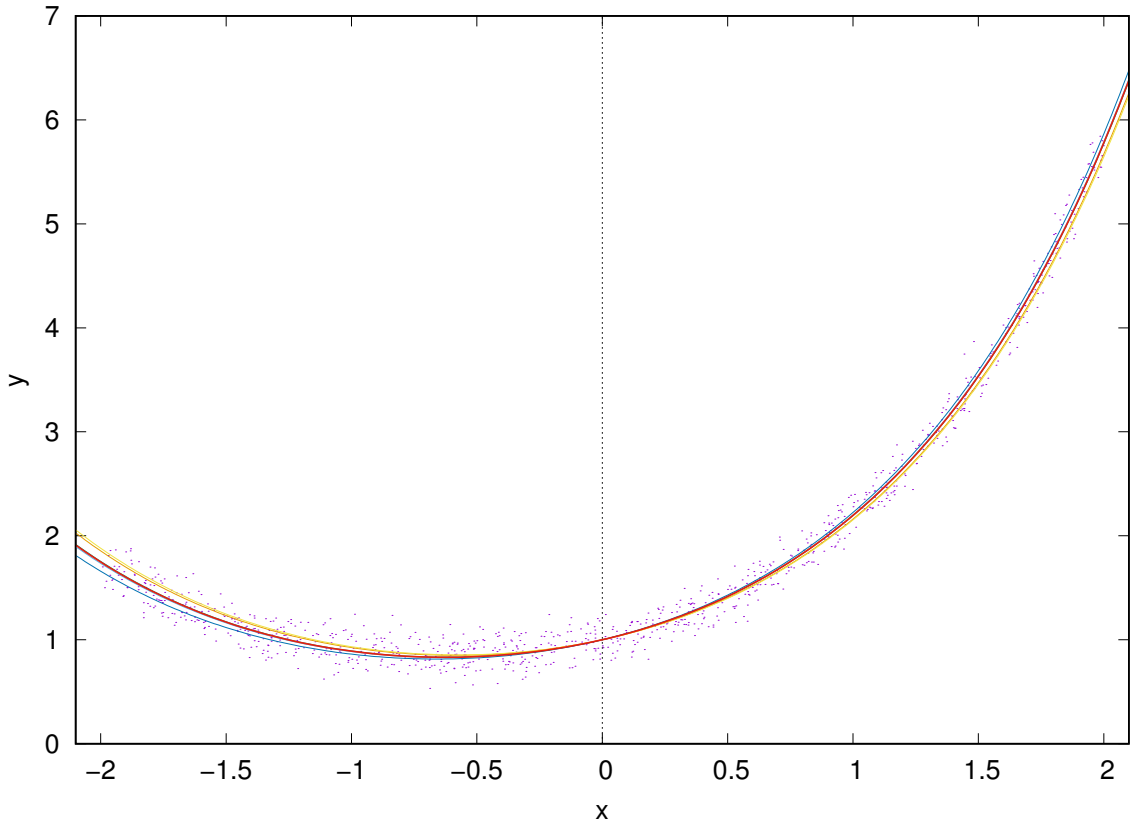
Dla opisanego wyżej problemu przykładowe trajektorie wyglądają tak:



Widać, że początkowo na każdej trajektorii wartości funkcji Q opadają, po czym mniej-więcej osiągają plateau, od którego fluktuacyjnie się odchylają.

Podkreślam, że wartości Q są oblicznae wzdłuż trajektorii tylko dla celów ilustracyjnych. Mając N punktów danych, zastosowanie **pełnej** metody najszybszego spadku wymaga w każdym kroku oblicznie N przyczynków do gradientu, N przyczynków do Q oraz wysumowania ich: jeden krok ma zatem złożoność $O(N)$, co może być kosztowne dla $N \gg 1$, a taka stuacja zachodzi w przypadku *Big Data*. Przy zastosowaniu Stochastic Gradient Descent w każdym kroku obliczamy tylko jeden przyczynek, a wartość funkcji Q oblicza się tylko raz, po zakończeniu każdej trajektorii.

Co ciekawe, w omawianym przykładzie dopasowane krzywe *niezbyt* różnią się od siebie w obszarze, w którym dostępne są dane (ekstrapolacja poza ten obszar może spowodować znaczne różnice).



Przykład ten pokazuje, że metoda najszybszego spadku dobrze działa *daleko* od minimum, natomiast jeśli zależy nam na precyzyjnej lokalizacji minimum, metoda ta staje się nieefektywna.

Metody najszybszego spadku warto używać do pewnego zgrubnego zlokalizowania minimum. **BARDZO** często to wystarcza! Jeśli jednak zależy nam na precyzyjnej lokalizacji minimum, należy użyć innych, bardziej zaawansowanych metod.

Zastosowanie metody Newtona

Formalnie znalezienie minimum funkcji wielu zmiennych jest równoważne z rozwiązaniem równania

$$\nabla f = 0 \tag{13}$$

Można spróbować rozwiązywać to równanie za pomocą wielowymiarowej metody Newtona. W kroku Newtonowskim pojawiłaby się macierz pochodnych cząstkowych wektora ∇f; jest to oczywiście hessjan, czyli macierz *drugich* pochodnych cząstkowych funkcji f:

$$\mathbf{H}_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}, \quad i,j = 1,\ldots,N \tag{14}$$

Ostatecznie metoda Newtona miałaby postać

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}^{-1}(\mathbf{x}_k)\, \nabla f|_{\mathbf{x}_k} \tag{15}$$

Tak hessjan, jak i gradient należy wyliczać w bieżącym punkcie $\mathbf{x}_k$.

Problemy z metodą Newtona

Metoda Newtona w postaci (15) nastręcza kilka problemów:

- Dla $N \gg 1$ zastosowanie metody Newtona jest **monstrualnie** kosztowne
- Nawet dla N rzędu kilku-kilkunastu, wielokrotne wyliczanie pochodnych cząstkowych, a następnie rozwiązywanie układu równań $\mathbf{Hz} = \nabla f$, może być drogie i uciążliwe
- Jeżeli minimalizowana funkcja **nie** jest formą kwadratową (a na ogół nie jest!), krok Newtonowski może być zbyt długi; metoda może wtedy "przestrzelić" nad poszukiwanym minimum
- Warunkiem koniecznym na to, aby iteracja (15) prowadziła do zmniejszania się wartości funkcji, jest symetria i dodatnia określoność wszystkich napotykanych po drodze hessjanów. Istotnie, aby po wykonaniu

kroku Newtonowskiego minimalizowana funkcja malała, musi zachodzić

$$(\mathbf{x}_{k+1} - \mathbf{x}_k)^T \nabla f|_{\mathbf{x}_k} < 0 \tag{16a}$$

— iloczyn skalarny gradientu funkcji i przesunięcia, czyli rzut przesunięcia na gradient, jest ujemny, co oznacza, że przesunięcie zostało wykonane w kierunku *malejących* wartości funkcji. Po formalnym rozwiązaniu (15) ze względu na $\nabla f|_{\mathbf{x}_k}$ i wstawieniu wyniku do powyższego równania, daje to

$$-(\mathbf{x}_{k+1} - \mathbf{x}_k)^T \mathbf{H} (\mathbf{x}_{k+1} - \mathbf{x}_k) < 0 \tag{16b}$$

co jest spełnione, jeżeli $\mathbf{H}$ jest dodatnio określone.

Dodatnia określoność hessjanu

Możemy bezpiecznie założyć, że jeśli minimalizowana funkcja jest dostatecznie gładka, hessjan jest w każdym punkcie symetryczny. Niestety, nie można tego założyć odnośnie do dodatniej określoności.

W minimum hessjan jest dodatnio określony. Na podstawie ciągłości drugich pochodnych wnioskujemy, że musi istnieć pewne otoczenie minimum, w którym hessjan jest dodatnio określony. Jednak daleko od minimum hessjan dodatnio określony być nie musi. Możemy traktować dodatnią określoność hessjanu jako matematyczny warunek tego, że jesteśmy blisko minimum.

Metoda Levenberga-Marquardta

Daleko od minimum stosowanie metody najszybszego spadku ma sens. Jak jednak zobaczymy, metoda ta nie nalepiej nadaje się do precyzyjnego określenia położenia minimum. Chcielibyśmy skonstruować kryterium pozwalające stwierdzić, czy jesteśmy już na tyle blisko minimum, że warto/można przełączyć się na metody bardziej precyzyjne, jeśli zachodzi taka potrzeba.

Metodą taką jest metoda Levenberga-Marquardta. **Zaleca** się jest sosowanie, gdy

- wymiarowość problemu nie jest bardzo duża, $N \sim 100$ co najwyżej
- obliczanie drugich pochodnych jest możliwe i nie jest zbyt uciążliwe
- gdy koszt obliczania drugich pochodnych **nie** jest wielki ze względu na strukturę problemu (jak w Big Data, gdzie wymiarowość problemu

— liczba dopasowywanych parametrów — może być niewielka, ale minimalizowana funkcja jest sumą **BARDZO** wielu składników).

Metoda Levenberga-Marquardta wywodzi się z metody Newtona. Daleko od minimum hessjan nie musi być nawet dodatnio określony, co powoduje, iż krok newtonowski wcale nie musi prowadzić do spadku wartości funkcji (por. (16b)). My jednak chcemy aby wartość funkcji w kolejnych krokach spadała. Zmodyfikujmy więc hessjan:

$$\widetilde{\mathbf{H}}_{ii} = (1+\lambda)\frac{\partial^2 f}{\partial x_i^2}\,, \tag{17a}$$

$$\widetilde{\mathbf{H}}_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}\,, \quad i \neq j\,, \tag{17b}$$

przy czym $\lambda \geqslant 0$.

Zauważmy, że zachodzi jedna z dwu możliwych sytuacji: (i) jeśli znajdujemy się w basenie atrakcji minimum, wówczas dla odpowiednio dużego λ macierz (17) stanie się dodatnio określona lub też (ii) jeśli dla żadnego dodatniego λ macierz (17) nie staje się dodatnio określona, oznacza to, że znajdujemy się na monotonicznej gałęzi funkcji, poza basenem atrakcji minimum.

Rozpoczynamy z jakimś niewielkim λ, na przykład $\lambda = \lambda_0 = 2^{-10} = 1/1024$. Przypuśćmy, iż aktualnym przybliżeniem minimum jest punkt $\mathbf{x}_k$. Dostajemy zatem...

verte

Algorytm Levenberga-Marquardta

1. Oblicz $\nabla f|_{\mathbf{x}_k}$.
2. Oblicz $\widetilde{\mathbf{H}}(\mathbf{x}_k)$.
3. Oblicz

$$\mathbf{x}_{\text{test}} = \mathbf{x}_k - \widetilde{\mathbf{H}}^{-1}(\mathbf{x}_k)\, \nabla f|_{\mathbf{x}_k} \ . \tag{18}$$

4. Jeżeli $f(\mathbf{x}_{\text{test}}) > f(\mathbf{x}_k)$, to
   (a) $\lambda \to 8\lambda$ (można też powiększać o inny znaczny czynnik).
   (b) Idź do punktu 2.
5. Jeżeli $f(\mathbf{x}_{\text{test}}) < f(\mathbf{x}_k)$, to
   (a) $\lambda \to \lambda/8$ (można też zmniejszać o inny znaczny czynnik).
   (b) $\mathbf{x}_{k+1} = \mathbf{x}_{\text{test}}$.
   (c) Idź do punktu 1.

Komentarz

Daleko od minimum *nie minimalizujemy* (nie poszukujemy minimów kierunkowych), a jedynie podążamy w kierunku malejących wartości funkcji.

Dodatkowo, jeśli $\lambda > \lambda_{\text{max}} \gg 1$, uznajemy, iż znajdujemy się poza basenem atrakcji minimum i algorytm zawodzi. Jeśli natomiast $\lambda < \lambda_{\text{min}} \ll 1$, macierz $\widetilde{\mathbf{H}}$ jest w praktyce równa hessjanowi, który jest dodatnio określony, o czym z kolei świadczy fakt, że uzyskujemy malejące wartości funkcji f. Modyfikacja (17) przestaje być potrzebna, możemy za to przerzucić się na metody właściwe dla bliskich okolic minimum.

Ponadto w celu przyspieszenia obliczeń, jeżeli $f(\mathbf{x}_{\text{test}}) < f(\mathbf{x}_k)$, możemy *chwilowo* zrezygnować ze zmniejszania λ i modyfikowania $\widetilde{\mathbf{H}}$ i przeprowadzić kilka kroków z tą samą macierzą, a więc korzystając z tej samej faktoryzacji.

Zauważmy, iż przypadek $\lambda \gg 1$ oznacza, iż jesteśmy daleko od minimum. Z drugiej strony jeśli $\lambda \gg 1$, macierz $\widetilde{\mathbf{H}}$ staje się w praktyce diagonalna, a zatem

$$\begin{aligned}\mathbf{x}_{\text{test}} &\simeq \mathbf{x}_k - (1+\lambda)^{-1}\text{diag}\left\{\left(\frac{\partial^2 f}{\partial x_1^2}\right)^{-1}, \left(\frac{\partial^2 f}{\partial x_2^2}\right)^{-1}, \ldots, \left(\frac{\partial^2 f}{\partial x_N^2}\right)^{-1}\right\}\nabla f|_{\mathbf{x}_k} \\ &\simeq \mathbf{x}_i - \text{const}\cdot\nabla f|_{\mathbf{x}_k}\ , \end{aligned} \tag{19}$$

o ile drugie pochodne cząstkowe w poszczególnych kierunkach nie różnią się znacznie od siebie. Widać, iż daleko od minimum, gdzie warunek zachowujący raz osiągnięte minima kierunkowe nie zachodzi, algorytm Levenberga-Marquardta zachowuje się prawie jak metoda najszybszego spadku.