Optymalizacja ciągła

5. Metoda stochastycznego spadku wzdłuż gradientu

Wojciech Kotłowski

Instytut Informatyki PP
http://www.cs.put.poznan.pl/wkotlowski/

04.04.2019

Wprowadzenie

Minimalizacja **różniczkowalnej** funkcji $f(\boldsymbol{w})$ **bez ograniczeń**:

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w})$$

Uwaga: na potrzeby tego wykładu zmieniamy oznaczenie wektora zmiennych z $\boldsymbol{x}$ na $\boldsymbol{w}$

W wielu zastosowaniach optymalizacji funkcja celu ma postać sumy po N elementach:

$$f(\boldsymbol{w}) = \sum_{i=1}^{N} f_i(\boldsymbol{w}),$$

To założenie będzie nam towarzyszyło przez cały wykład.

Przykład: regresja liniowa

Przewidywania/wyjaśnienie zmian ciągłej zmiennej wyjściowej (Y) pod wpływem zmian innych zmiennych ($X_1, \ldots, X_n$).

Przykłady

- X – ceny akcji w ostatnim tygodniu, Y – cena akcji jutro.
- X – wyniki testów medycznych, Y – poziom zaawansowania choroby.
- X – wielkość programu, Y – czas pisania programu.
- X – warunki na drodze, czas, lokalizacja, Y – średnia prędkość samochodów.
- X – cechy domu Y – cena domu.

Przykład: regresja liniowa

Modelujemy zmienną Y jako funkcję liniową $X_1, \ldots, X_n$:

$$Y \;=\; w_0 + w_1 X_1 + \ldots + w_n X_n \;=\; \boldsymbol{X}^\top \boldsymbol{w}$$

gdzie $\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$ jest wektorem **współczynników** (zmiennych decyzyjnych), natomiast $\boldsymbol{X} = (1, X_1, \ldots, X_n)$.

Przykład: regresja liniowa

Modelujemy zmienną Y jako funkcję liniową $X_1, \ldots, X_n$:

$$Y \;=\; w_0 + w_1 X_1 + \ldots + w_n X_n \;=\; \boldsymbol{X}^\top \boldsymbol{w}$$

gdzie $\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$ jest wektorem **współczynników** (zmiennych decyzyjnych), natomiast $\boldsymbol{X} = (1, X_1, \ldots, X_n)$.

Wartości współczynników dopasowujemy **na podstawie zbioru danych**

Przykład: regresja liniowa

Otrzymujemy zbiór danych historycznych, na którym znane są wartości y:

$$
\begin{array}{ll}
(x_{11}, x_{12}, \ldots, x_{1n}, y_1) & \\
(x_{21}, x_{22}, \ldots, x_{2n}, y_2) & \\
\ldots & \text{lub w skrócie} \\
(x_{N1}, x_{N2}, \ldots, x_{Nn}, y_N) &
\end{array}
\qquad
\begin{array}{l}
(\boldsymbol{x}_1, y_1) \\
(\boldsymbol{x}_2, y_2) \\
\ldots \\
(\boldsymbol{x}_N, y_N)
\end{array}
$$

Przykład: regresja liniowa

Otrzymujemy zbiór danych historycznych, na którym znane są wartości y:

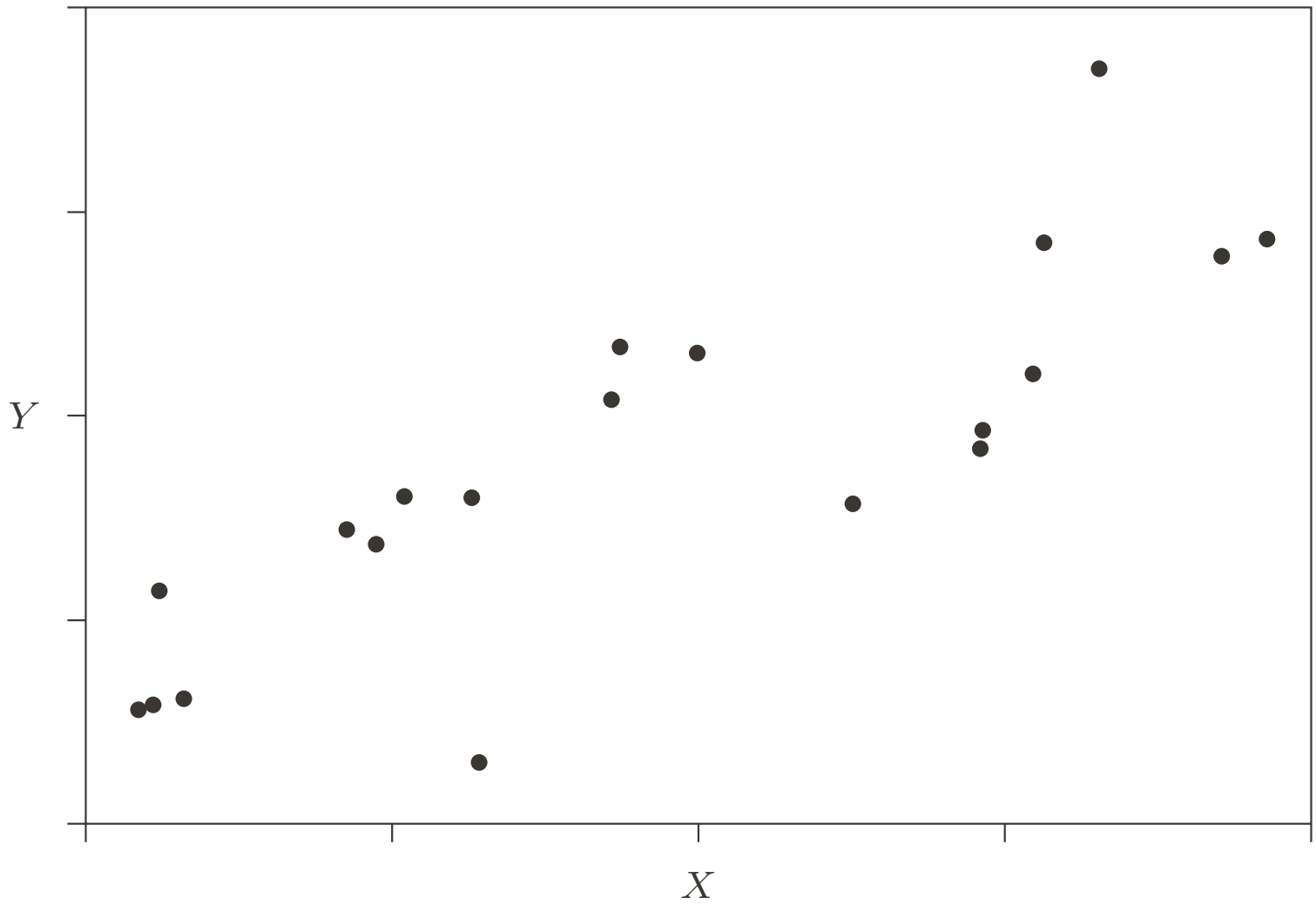
$$\begin{array}{ll} (x_{11}, x_{12}, \ldots, x_{1n}, y_1) & \\ (x_{21}, x_{22}, \ldots, x_{2n}, y_2) & \\ \ldots & \text{lub w skrócie} \\ (x_{N1}, x_{N2}, \ldots, x_{Nn}, y_N) & \end{array} \qquad \begin{array}{l} (\boldsymbol{x}_1, y_1) \\ (\boldsymbol{x}_2, y_2) \\ \ldots \\ (\boldsymbol{x}_N, y_N) \end{array}$$

Wyznaczamy współczynniki $\boldsymbol{w}$ tak, aby funkcja liniowa jak najlepiej przybliżała wartości zmiennej Y na danych, tzn. aby:

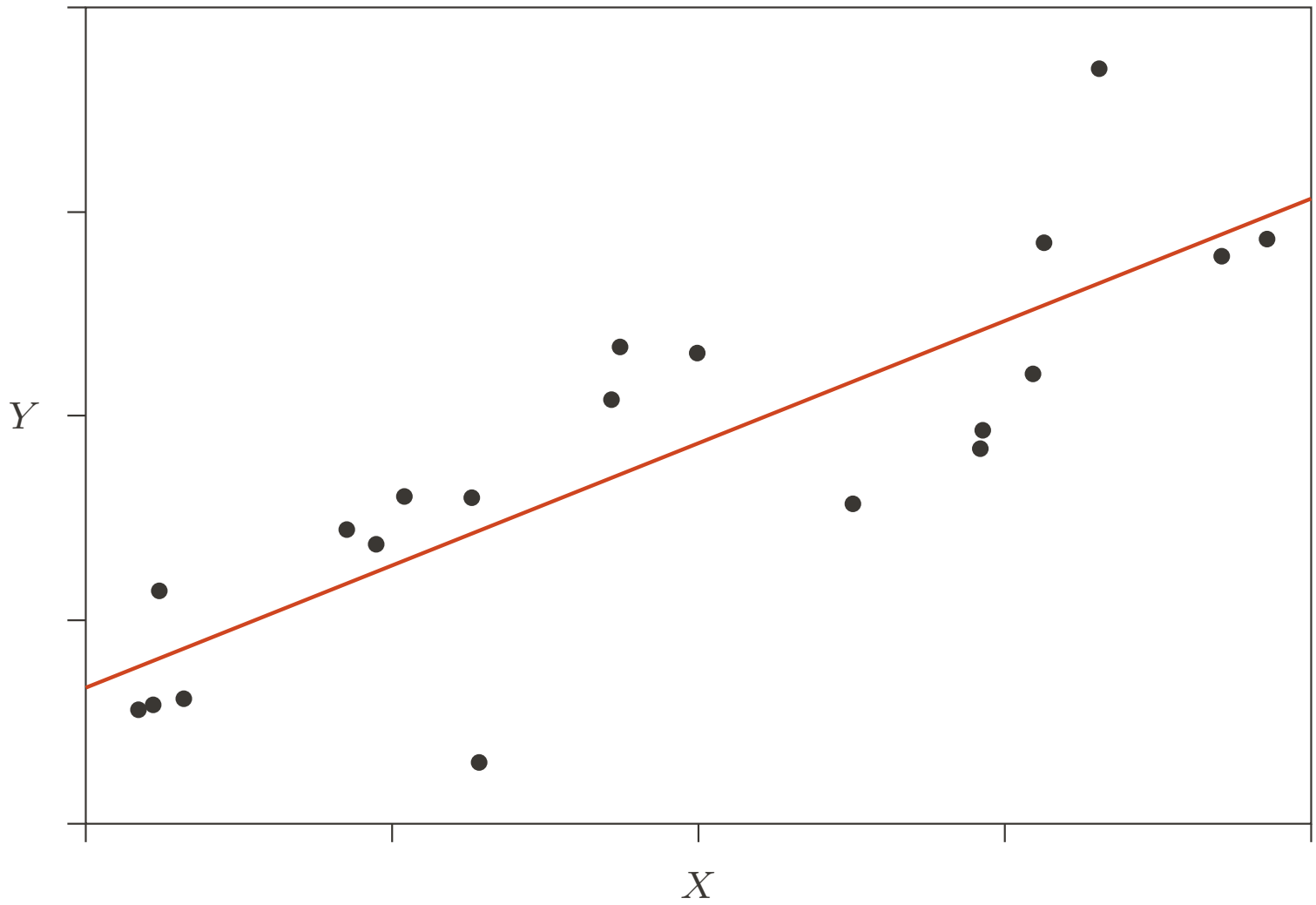
$$\hat{y}_i = \boldsymbol{x}_i^\top \boldsymbol{w}$$

było jak najbliższe y_i dla wszystkich $i = 1, \ldots, N$

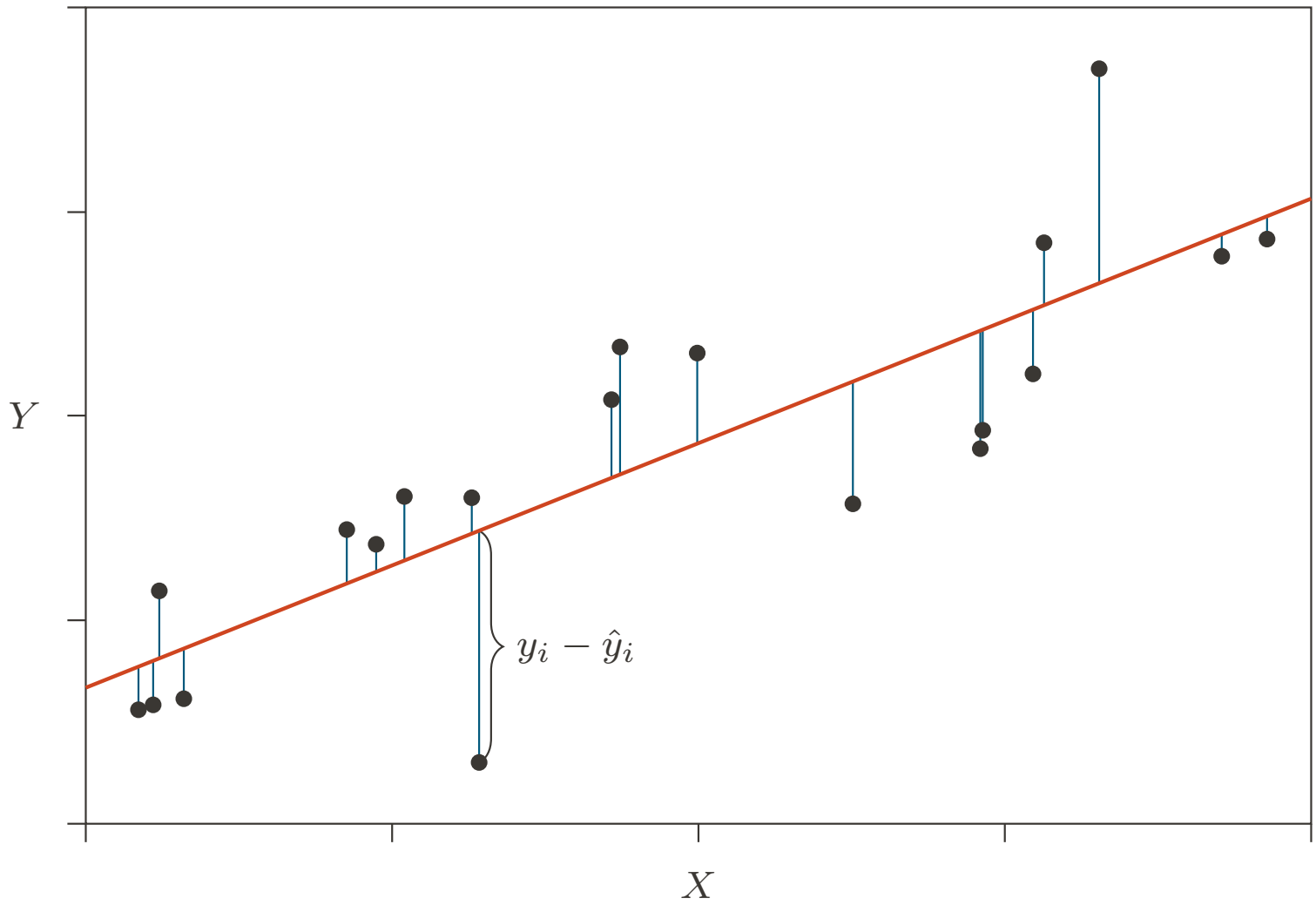
Przykład: regresja liniowa



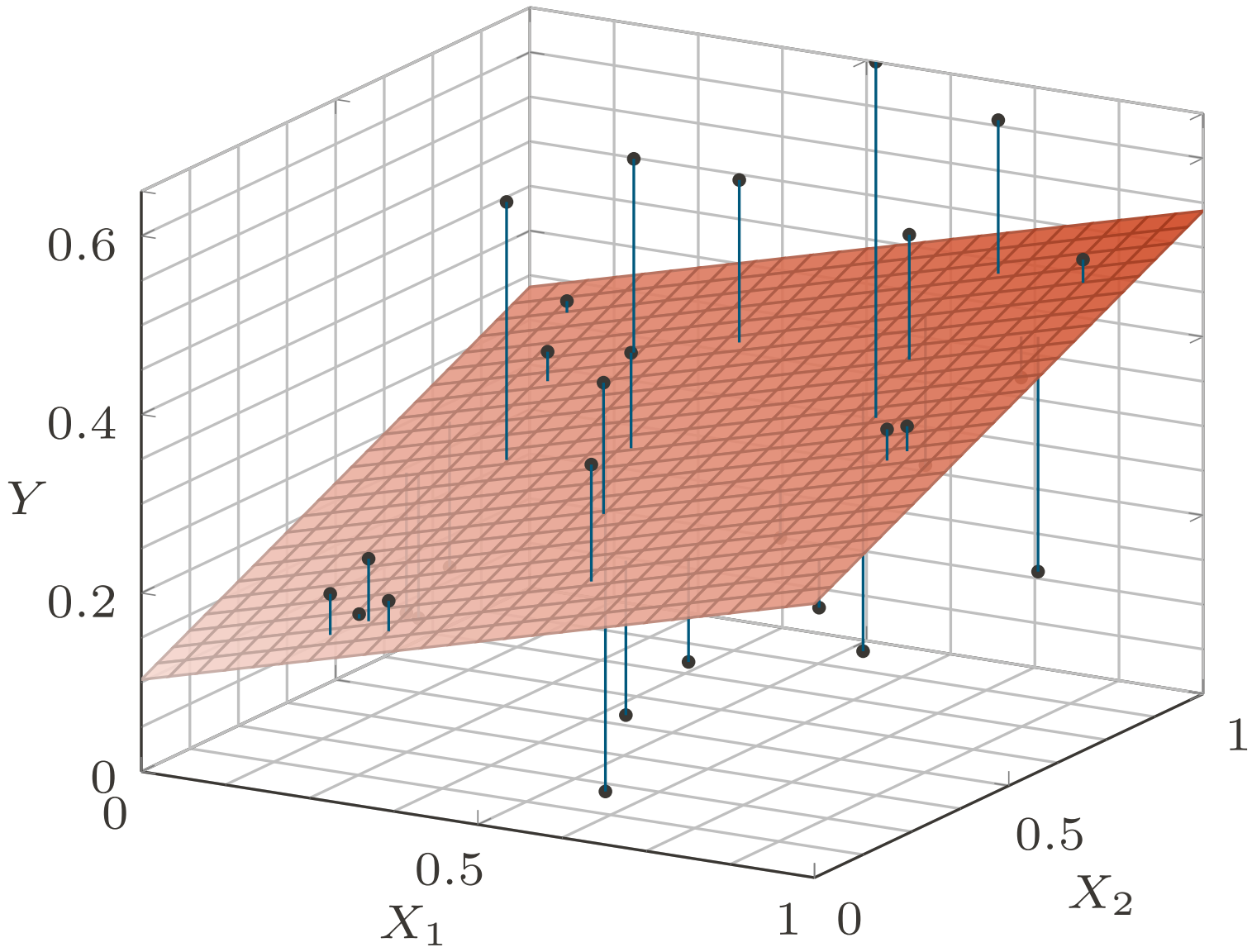
Przykład: regresja liniowa



Przykład: regresja liniowa



Przykład: regresja liniowa



Przykład: regresja liniowa

Metoda najmniejszych kwadratów:
Wyznacz współczynniki regresji $\boldsymbol{w}$ by minimalizować sumę kwadratów odchyleń modelu od danych:

$$\min_{\boldsymbol{w} \in \mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Przykład: regresja liniowa

Metoda najmniejszych kwadratów:
Wyznacz współczynniki regresji $\boldsymbol{w}$ by minimalizować sumę kwadratów odchyleń modelu od danych:

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2$$

Przykład: regresja liniowa

Metoda najmniejszych kwadratów:
Wyznacz współczynniki regresji $\boldsymbol{w}$ by minimalizować sumę kwadratów odchyleń modelu od danych:

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} \underbrace{(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2}_{f_i(\boldsymbol{w})}$$

Przykład: regresja liniowa

Metoda najmniejszych kwadratów:
Wyznacz współczynniki regresji $\boldsymbol{w}$ by minimalizować sumę kwadratów odchyleń modelu od danych:

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} \underbrace{(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2}_{f_i(\boldsymbol{w})}$$

Innymi słowy: wyznaczamy współczynniki modelu liniowego minimalizując **sumaryczny błąd** (wyrażony w postaci funkcji kwadratowej) **na całym zbiorze danych**

Przykłady: uczenie maszynowe

$$f(\boldsymbol{w}) = \sum_{i=1}^{N} f_i(\boldsymbol{w})$$

Zdecydowana większość metod **uczenia maszynowego** opiera się zasadniczo na tym samym schemacie:

Mając **zbiór danych**, wyznacz **parametry** modelu predykcyjnego **minimalizując sumaryczny błąd modelu na danych**

- $\boldsymbol{w}$ – parametry modelu (współczynniki regresji, wagi w sieci neuronowej, itp.)
- $f_i(\boldsymbol{w})$ – błąd na pojedynczej obserwacji ze zbioru danych (np. błąd kwadratowy w regresji)
- $f(\boldsymbol{w})$ – sumaryczna wartość błędu na całym zbiorze danych

Podejście klasyczne

Znajdź rozwiązanie problemu:

$$\min_{\boldsymbol{w} \in \mathbb{R}^n} f(\boldsymbol{w})$$

stosując jedną z klasycznych metod optymalizacji, np. metodę spadku wzdłuż gradientu lub Newtona-Raphsona

Podejście klasyczne

Znajdź rozwiązanie problemu:

$$\min_{\boldsymbol{w} \in \mathbb{R}^n} f(\boldsymbol{w})$$

stosując jedną z klasycznych metod optymalizacji, np. metodę spadku wzdłuż gradientu lub Newtona-Raphsona

Typowe problemy w praktyce:

- Policzenie gradientu/hesjanu wymaga sumowania N gradientów/hesjanów składowych funkcji celu:

$$\nabla f(\boldsymbol{w}) \; = \; \sum_{i=1}^{N} \nabla f_i(\boldsymbol{w})$$

- Wymiar problemu n może być bardzo duży, przez co metoda Newtona-Raphsona może być zbyt kosztowna do uruchomienia

Podejście klasyczne: spadek wzdłuż gradientu

Rozpoczynając od $\boldsymbol{w}_1$, w kolejnych iteracjach $k = 1, 2, \ldots$:

$$\boldsymbol{w}_{k+1} \;=\; \boldsymbol{w}_k - \alpha_k \underbrace{\sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}_k)}_{\nabla f(\boldsymbol{w}_k)}$$

Wykonanie jednego kroku: przejście po wszystkich składowych funkcji celu.

Podejście klasyczne: spadek wzdłuż gradientu

Rozpoczynając od $\boldsymbol{w}_1$, w kolejnych iteracjach $k = 1, 2, \ldots$:

$$\boldsymbol{w}_{k+1} \;=\; \boldsymbol{w}_k - \alpha_k \underbrace{\sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}_k)}_{\nabla f(\boldsymbol{w}_k)}$$

Wykonanie jednego kroku: przejście po wszystkich składowych funkcji celu.

Pomysł: policz gradient na **losowej pojedynczej składowej** f_i zamiast na całej funkcji f!

Metoda stochastycznego spadku wzdłuż gradientu (*stochastic gradient descent, SGD*)

Rozpoczynając od $\boldsymbol{w}_1$, w kolejnych iteracjach $k = 1, 2, \ldots$:

Wylosuj jedną ze składowych $i_k \in \{1, \ldots, N\}$

Wykonaj krok wzdłuż ujemnego gradientu funkcji $f_{i_k}(\boldsymbol{w}_k)$:

$$\boldsymbol{w}_{k+1} \;=\; \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k)$$

Metoda stochastycznego spadku wzdłuż gradientu (*stochastic gradient descent, SGD*)

Rozpoczynając od $\boldsymbol{w}_1$, w kolejnych iteracjach $k = 1, 2, \ldots$:

Wylosuj jedną ze składowych $i_k \in \{1, \ldots, N\}$

Wykonaj krok wzdłuż ujemnego gradientu funkcji $f_{i_k}(\boldsymbol{w}_k)$:

$$\boldsymbol{w}_{k+1} \;=\; \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k)$$

Dlaczego miałoby to w ogóle działać?

Metoda stochastycznego spadku wzdłuż gradientu (*stochastic gradient descent, SGD*)

Rozpoczynając od $\boldsymbol{w}_1$, w kolejnych iteracjach $k = 1, 2, \ldots$:

Wylosuj jedną ze składowych $i_k \in \{1, \ldots, N\}$

Wykonaj krok wzdłuż ujemnego gradientu funkcji $f_{i_k}(\boldsymbol{w}_k)$:

$$\boldsymbol{w}_{k+1} \;=\; \boldsymbol{w}_k - \alpha_k \nabla f_{i_k}(\boldsymbol{w}_k)$$

Dlaczego miałoby to w ogóle działać?

Możemy myśleć o gradiencie pojedynczej składowej $\nabla f_i(\boldsymbol{w}_k)$ jako o **oszacowaniu** całościowego gradientu $\nabla f(\boldsymbol{w}_k)$

Rzeczywiście, **średnio** gradient będzie wynosił:

$$\frac{1}{n}\nabla f_1(\boldsymbol{w}_k) + \frac{1}{n}\nabla f_2(\boldsymbol{w}_k) + \ldots + \frac{1}{n}\nabla f_N(\boldsymbol{w}_k) = \frac{1}{n}\nabla f(\boldsymbol{w}_k)$$

Przykład: problem regresji liniowej

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} \underbrace{(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2}_{f_i(\boldsymbol{w})}$$

Przykład: problem regresji liniowej

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} \underbrace{(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2}_{f_i(\boldsymbol{w})}$$

Liczymy pochodne cząstkowe pojedynczej funkcji $f_i(\boldsymbol{w})$:

$$\frac{\partial f_i(\boldsymbol{w})}{\partial w_j} = -2(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})x_{ij}$$

Przykład: problem regresji liniowej

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} \underbrace{(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2}_{f_i(\boldsymbol{w})}$$

Liczymy pochodne cząstkowe pojedynczej funkcji $f_i(\boldsymbol{w})$:

$$\frac{\partial f_i(\boldsymbol{w})}{\partial w_j} = -2(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})x_{ij}$$

Gradient wynosi więc: $\nabla f_i(\boldsymbol{w}) = -2(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})\boldsymbol{x}_i$

Przykład: problem regresji liniowej

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} \underbrace{(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2}_{f_i(\boldsymbol{w})}$$

Liczymy pochodne cząstkowe pojedynczej funkcji $f_i(\boldsymbol{w})$:

$$\frac{\partial f_i(\boldsymbol{w})}{\partial w_j} \;=\; -2(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})x_{ij}$$

Gradient wynosi więc: $\nabla f_i(\boldsymbol{w}) = -2(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})\boldsymbol{x}_i$

Rozpoczynając od $\boldsymbol{w}_1$, w kolejnych iteracjach $k = 1, 2, \ldots$:

Wylosuj jedną ze obserwacji $i_k \in \{1, \ldots, N\}$

Wykonaj krok wzdłuż ujemnego gradientu funkcji $f_{i_k}(\boldsymbol{w}_k)$:

$$\boldsymbol{w}_{k+1} \;=\; \boldsymbol{w}_k + 2\alpha_k(y_{i_k} - \boldsymbol{x}_{i_k}^\top \boldsymbol{w}_k)\boldsymbol{x}_{i_k}$$

Przykład: problem regresji liniowej

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) = \sum_{i=1}^{N} \underbrace{(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})^2}_{f_i(\boldsymbol{w})}$$

Liczymy pochodne cząstkowe pojedynczej funkcji $f_i(\boldsymbol{w})$:

$$\frac{\partial f_i(\boldsymbol{w})}{\partial w_j} \;=\; -2(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})x_{ij}$$

Gradient wynosi więc: $\nabla f_i(\boldsymbol{w}) = -2(y_i - \boldsymbol{x}_i^\top \boldsymbol{w})\boldsymbol{x}_i$

Rozpoczynając od $\boldsymbol{w}_1$, w kolejnych iteracjach $k = 1, 2, \ldots$:

Wylosuj jedną ze obserwacji $i_k \in \{1, \ldots, N\}$

Wykonaj krok wzdłuż ujemnego gradientu funkcji $f_{i_k}(\boldsymbol{w}_k)$:

$$\boldsymbol{w}_{k+1} \;=\; \boldsymbol{w}_k + 2\alpha_k(y_{i_k} - \boldsymbol{x}_{i_k}^\top \boldsymbol{w}_k)\boldsymbol{x}_{i_k}$$

Krok w kierunku wektora obserwacji $\boldsymbol{x}_{i_k}$, z długością kroku proporcjonalną do „przeszacowania predykcji" $(y_{i_k} - \boldsymbol{x}_{i_k}^\top \boldsymbol{w}_k)$.

Zalety i wady algorytmu SGD

Zalety

- **Szybkość**: obliczenie gradientu wymaga wzięcia tylko jednej obserwacji.
- **Skalowalność**: cały zbiór danych nie musi nawet znajdować się w pamięci operacyjnej.
- **Prostota**: gradient funkcji f_i daje bardzo prosty wzór na modyfikację wag.

Wady

- **Wolna zbieżność**: algorytm w ogólności zbiega wolno i wymaga wielu iteracji po zbiorze uczącym.
- **Ustalenie długości kroku** α_k: wyznaczenie α_k przez jednowymiarową optymalizację nie przynosi dobrych rezultatów, ponieważ nie optymalizujemy oryginalnej funkcji f tylko jeden jej składnik f_i.

Zalety znacznie przewyższają wady: Algorytm SGD jest obecnie najczęściej używanym algorytmem w uczeniu maszynowym!

SGD w praktyce

SGD w praktyce

- Zwykle nie losuje się obserwacji, ale przechodzi się po zbiorze danych w losowej kolejności (działa bardzo podobnie, a daje znacznie prostszą implementację)

SGD w praktyce

- Zwykle nie losuje się obserwacji, ale przechodzi się po zbiorze danych w losowej kolejności (działa bardzo podobnie, a daje znacznie prostszą implementację)
- Zbieżność wymaga często przejścia parokrotnie po całym zbiorze danych (jednokrotne przejście nazywa się **epoką**).

SGD w praktyce

- Zwykle nie losuje się obserwacji, ale przechodzi się po zbiorze danych w losowej kolejności (działa bardzo podobnie, a daje znacznie prostszą implementację)
- Zbieżność wymaga często przejścia parokrotnie po całym zbiorze danych (jednokrotne przejście nazywa się **epoką**).
- Metody ustalania współczynników długości kroku α_k:

SGD w praktyce

- Zwykle nie losuje się obserwacji, ale przechodzi się po zbiorze danych w losowej kolejności (działa bardzo podobnie, a daje znacznie prostszą implementację)
- Zbieżność wymaga często przejścia parokrotnie po całym zbiorze danych (jednokrotne przejście nazywa się **epoką**).
- Metody ustalania współczynników długości kroku α_k:
  - Ustalamy **stałą wartość** $\alpha_k = \alpha$

SGD w praktyce

- Zwykle nie losuje się obserwacji, ale przechodzi się po zbiorze danych w losowej kolejności (działa bardzo podobnie, a daje znacznie prostszą implementację)
- Zbieżność wymaga często przejścia parokrotnie po całym zbiorze danych (jednokrotne przejście nazywa się **epoką**).
- Metody ustalania współczynników długości kroku α_k:
  - Ustalamy **stałą wartość** $\alpha_k = \alpha$
    $\Longrightarrow$ Zwykle tak się robi w praktyce, działa dobrze ale wymaga ustalenia α metodą prób i błędów

SGD w praktyce

- Zwykle nie losuje się obserwacji, ale przechodzi się po zbiorze danych w losowej kolejności (działa bardzo podobnie, a daje znacznie prostszą implementację)
- Zbieżność wymaga często przejścia parokrotnie po całym zbiorze danych (jednokrotne przejście nazywa się **epoką**).
- Metody ustalania współczynników długości kroku α_k:
  - Ustalamy **stałą wartość** $\alpha_k = \alpha$
    $\Longrightarrow$ Zwykle tak się robi w praktyce, działa dobrze ale wymaga ustalenia α metodą prób i błędów
  - Bierzemy wartość kroku **malejącą jak** $\sim \frac{1}{\sqrt{k}}$: $\alpha_k = \alpha/\sqrt{k}$

SGD w praktyce

- Zwykle nie losuje się obserwacji, ale przechodzi się po zbiorze danych w losowej kolejności (działa bardzo podobnie, a daje znacznie prostszą implementację)
- Zbieżność wymaga często przejścia parokrotnie po całym zbiorze danych (jednokrotne przejście nazywa się **epoką**).
- Metody ustalania współczynników długości kroku α_k:
  - Ustalamy **stałą wartość** $\alpha_k = \alpha$
    $\Longrightarrow$ Zwykle tak się robi w praktyce, działa dobrze ale wymaga ustalenia α metodą prób i błędów
  - Bierzemy wartość kroku **malejącą jak** $\sim \frac{1}{\sqrt{k}}$: $\alpha_k = \alpha/\sqrt{k}$
    $\Longrightarrow$ Zapewniona zbieżność, ale czasem może zbiegać zbyt wolno.

SGD w praktyce

- Zwykle nie losuje się obserwacji, ale przechodzi się po zbiorze danych w losowej kolejności (działa bardzo podobnie, a daje znacznie prostszą implementację)
- Zbieżność wymaga często przejścia parokrotnie po całym zbiorze danych (jednokrotne przejście nazywa się **epoką**).
- Metody ustalania współczynników długości kroku α_k:
  - Ustalamy **stałą wartość** $\alpha_k = \alpha$
    $\Longrightarrow$ Zwykle tak się robi w praktyce, działa dobrze ale wymaga ustalenia α metodą prób i błędów
  - Bierzemy wartość kroku **malejącą jak** $\sim \frac{1}{\sqrt{k}}$: $\alpha_k = \alpha/\sqrt{k}$
    $\Longrightarrow$ Zapewniona zbieżność, ale czasem może zbiegać zbyt wolno.
- W praktyce często liczy się gradient nie na pojedynczej obserwacji, ale na ich niewielkiej grupie (tzw. *mini-batch*)

Metoda stochastycznego spadku wzdłuż gradientu

Twierdzenie: Niech $f_i(\boldsymbol{w})$ $(i = 1, \ldots, N)$ będą funkcjami wypukłymi i niech $\boldsymbol{w}^*$ będzie minimum (globalnym) funkcji $f(\boldsymbol{w})$. Po K iteracjach algorytm stochastycznego spadku wzdłuż gradientu ze **stałą** długością kroku α gwarantuje:

$$\min_{k=1,\ldots,K} f(\boldsymbol{w}_k) \; - \; f(\boldsymbol{w}^*) \leqslant \frac{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^{K} \|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2$$

spa

wpływ „warunków początkowych”
(odległość od optimum na starcie)
(maleje z α)

wpływ wielkości gradientów
na trajektorii algorytmu
(rośnie z α)

$\ldots,$ ni i

ym) ch

algorytm stochastycznego spadku wzdłuż gradientu ze **stałą** długością kroku α gwarantuje:

$$\min_{k=1,\ldots,K} f(\boldsymbol{w}_k) \; - \; f(\boldsymbol{w}^*) \leqslant \frac{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^{K} \|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2$$

Metoda stochastycznego spadku wzdłuż gradientu

Twierdzenie: Niech $f_i(\boldsymbol{w})$ $(i = 1, \ldots, N)$ będą funkcjami wypukłymi i niech $\boldsymbol{w}^*$ będzie minimum (globalnym) funkcji $f(\boldsymbol{w})$. Po K iteracjach algorytm stochastycznego spadku wzdłuż gradientu ze **stałą** długością kroku α gwarantuje:

$$\min_{k=1,\ldots,K} f(\boldsymbol{w}_k) \; - \; f(\boldsymbol{w}^*) \leqslant \frac{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^{K} \|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2$$

Wniosek: sensowne jest wzięcie α tak, aby oba człony były podobnej wielkości:

$$(\ldots)\frac{1}{\alpha K} \; \simeq \; \frac{\alpha}{K} \sum_{k=1}^{K} (\ldots)$$

Metoda stochastycznego spadku wzdłuż gradientu

Twierdzenie: Niech $f_i(\boldsymbol{w})$ $(i = 1, \ldots, N)$ będą funkcjami wypukłymi i niech $\boldsymbol{w}^*$ będzie minimum (globalnym) funkcji $f(\boldsymbol{w})$. Po K iteracjach algorytm stochastycznego spadku wzdłuż gradientu ze **stałą** długością kroku α gwarantuje:

$$\min_{k=1,\ldots,K} f(\boldsymbol{w}_k) \; - \; f(\boldsymbol{w}^*) \leqslant \frac{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^{K} \|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2$$

Wniosek: sensowne jest wzięcie α tak, aby oba człony były podobnej wielkości:

$$(\ldots)\frac{1}{\alpha K} \; \simeq \; \alpha(\ldots)$$

Metoda stochastycznego spadku wzdłuż gradientu

Twierdzenie: Niech $f_i(\boldsymbol{w})$ $(i = 1, \ldots, N)$ będą funkcjami wypukłymi i niech $\boldsymbol{w}^*$ będzie minimum (globalnym) funkcji $f(\boldsymbol{w})$. Po K iteracjach algorytm stochastycznego spadku wzdłuż gradientu ze **stałą** długością kroku α gwarantuje:

$$\min_{k=1,\ldots,K} f(\boldsymbol{w}_k) \; - \; f(\boldsymbol{w}^*) \leqslant \frac{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^{K} \|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2$$

Wniosek: sensowne jest wzięcie α tak, aby oba człony były podobnej wielkości:

$$(\ldots)\frac{1}{K} \; \simeq \; \alpha^2$$

Metoda stochastycznego spadku wzdłuż gradientu

Twierdzenie: Niech $f_i(\boldsymbol{w})$ $(i = 1, \ldots, N)$ będą funkcjami wypukłymi i niech $\boldsymbol{w}^*$ będzie minimum (globalnym) funkcji $f(\boldsymbol{w})$. Po K iteracjach algorytm stochastycznego spadku wzdłuż gradientu ze **stałą** długością kroku α gwarantuje:

$$\min_{k=1,\ldots,K} f(\boldsymbol{w}_k) \; - \; f(\boldsymbol{w}^*) \leqslant \frac{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^{K} \|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2$$

Wniosek: sensowne jest wzięcie α tak, aby oba człony były podobnej wielkości:

$$\alpha \; \simeq \; \frac{(\ldots)}{\sqrt{K}}$$

Metoda stochastycznego spadku wzdłuż gradientu

Twierdzenie: Niech $f_i(\boldsymbol{w})$ $(i = 1, \ldots, N)$ będą funkcjami wypukłymi i niech $\boldsymbol{w}^*$ będzie minimum (globalnym) funkcji $f(\boldsymbol{w})$. Po K iteracjach algorytm stochastycznego spadku wzdłuż gradientu ze **stałą** długością kroku α gwarantuje:

$$\min_{k=1,\ldots,K} f(\boldsymbol{w}_k) \; - \; f(\boldsymbol{w}^*) \leqslant \frac{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^{K} \|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2$$

Wniosek: sensowne jest wzięcie α tak, aby oba człony były podobnej wielkości:

$$\alpha \; \simeq \; \frac{(\ldots)}{\sqrt{K}}$$

Stąd stosowane w praktyce malejące wartości α_k rzędu $\alpha_k = \frac{\text{const}}{\sqrt{k}}$

Metoda stochastycznego spadku wzdłuż gradientu

Twierdzenie: Niech $f_i(\boldsymbol{w})$ $(i = 1, \ldots, N)$ będą funkcjami wypukłymi i niech $\boldsymbol{w}^*$ będzie minimum (globalnym) funkcji $f(\boldsymbol{w})$. Po K iteracjach algorytm stochastycznego spadku wzdłuż gradientu ze **stałą** długością kroku α gwarantuje:

$$\min_{k=1,\ldots,K} f(\boldsymbol{w}_k) \; - \; f(\boldsymbol{w}^*) \leqslant \frac{\|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^{K} \|\nabla f_{i_k}(\boldsymbol{w}_k)\|^2$$

Wniosek: sensowne jest wzięcie α tak, aby oba człony były podobnej wielkości:

$$\alpha \; \simeq \; \frac{(\ldots)}{\sqrt{K}}$$

Stąd stosowane w praktyce malejące wartości α_k rzędu $\alpha_k = \frac{\text{const}}{\sqrt{k}}$

Powyższe wartości α_k dają gwarancje na suboptymalność rzędu $O\left(\frac{1}{\sqrt{K}}\right)$

Porównanie zbieżności

algorytm	zbieżność	błąd po k iteracjach
Metoda Cauchy'ego	liniowa	$\propto e^{-ck}$
Metoda Newtona-Raphsona	kwadratowa	$\propto e^{-c_1 e^{c_2 k}}$
SGD	subliniowa	$\propto \frac{c}{\sqrt{k}}$

Algorytm SGD jest **bardzo wolny** w stosunku do uprzednio poznanych

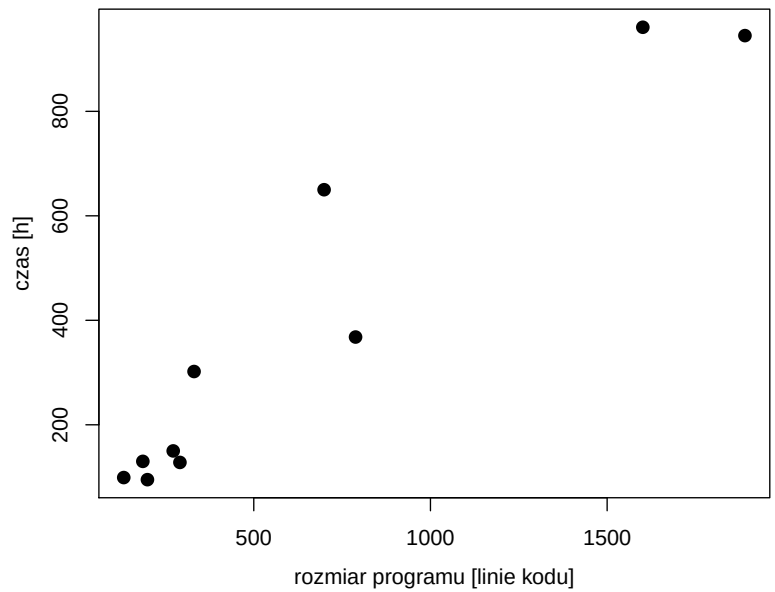
Stosuje się go wtedy, gdy nie jest nam potrzebna duża dokładność punktu optimum, a zależy nam na czasie optymalizacji

Przykład: szacowanie czasu pracy programistów

X	Y
Rozmiar programu	Oszacowany czas
186	130
699	650
132	99
272	150
291	128
331	302
199	95
1890	945
788	368
1601	961

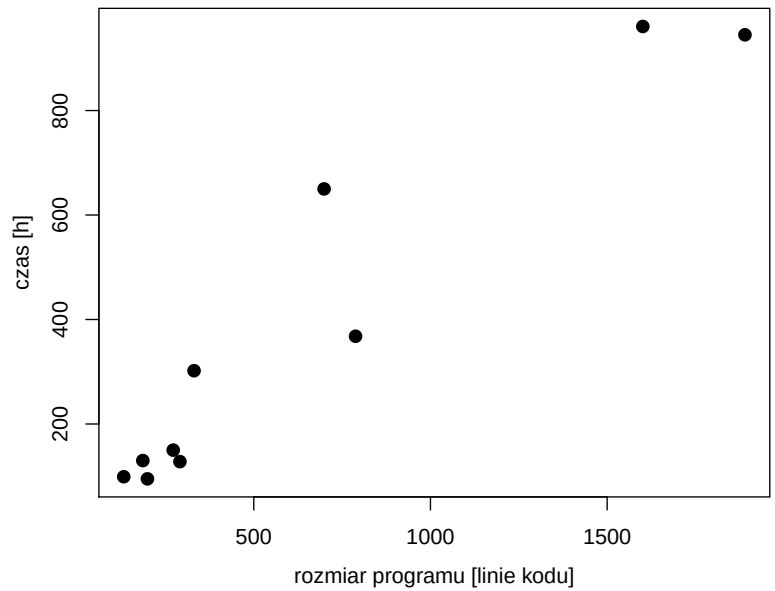
Przykład: szacowanie czasu pracy programistów

X	Y
Rozmiar programu	Oszacowany czas
186	130
699	650
132	99
272	150
291	128
331	302
199	95
1890	945
788	368
1601	961



Przykład: szacowanie czasu pracy programistów

X	Y
Rozmiar programu	Oszacowany czas
186	130
699	650
132	99
272	150
291	128
331	302
199	95
1890	945
788	368
1601	961



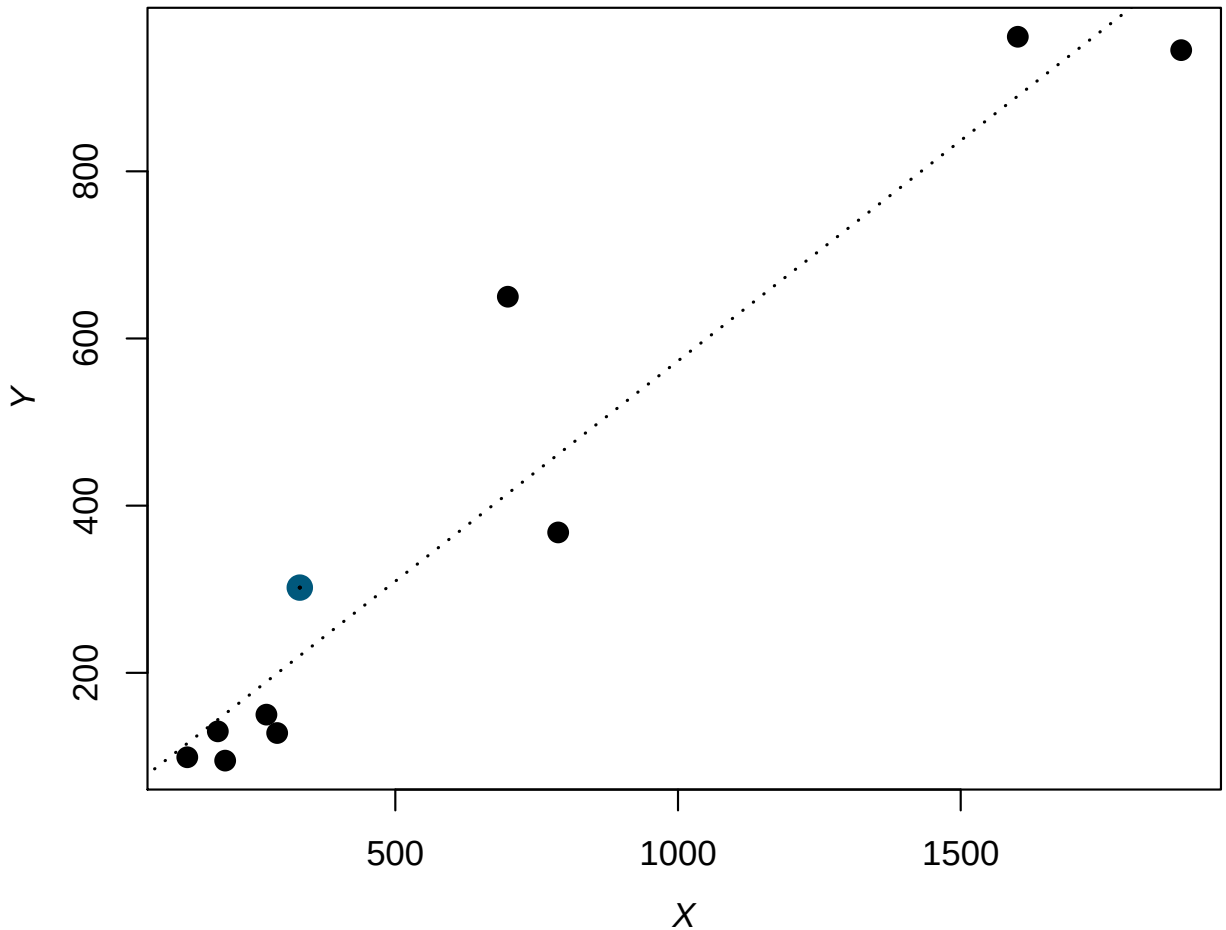
$$w_0 = 45.93, w_1 = 0.5273$$

Szacowanie czasu programistów

- Zaczynamy z rozwiązaniem $\boldsymbol{w} = (w_0, w_1) = \mathbf{0}$.
- Krok: $\alpha_k = 0.5/\sqrt{k}$.
- Jednostki zostały zamienione na 1000min i 1000 lini kodu, aby uniknąć dużych liczb.
  (ujednolicenie skali jest w praktyce bardzo istotne dla zbieżności metody)

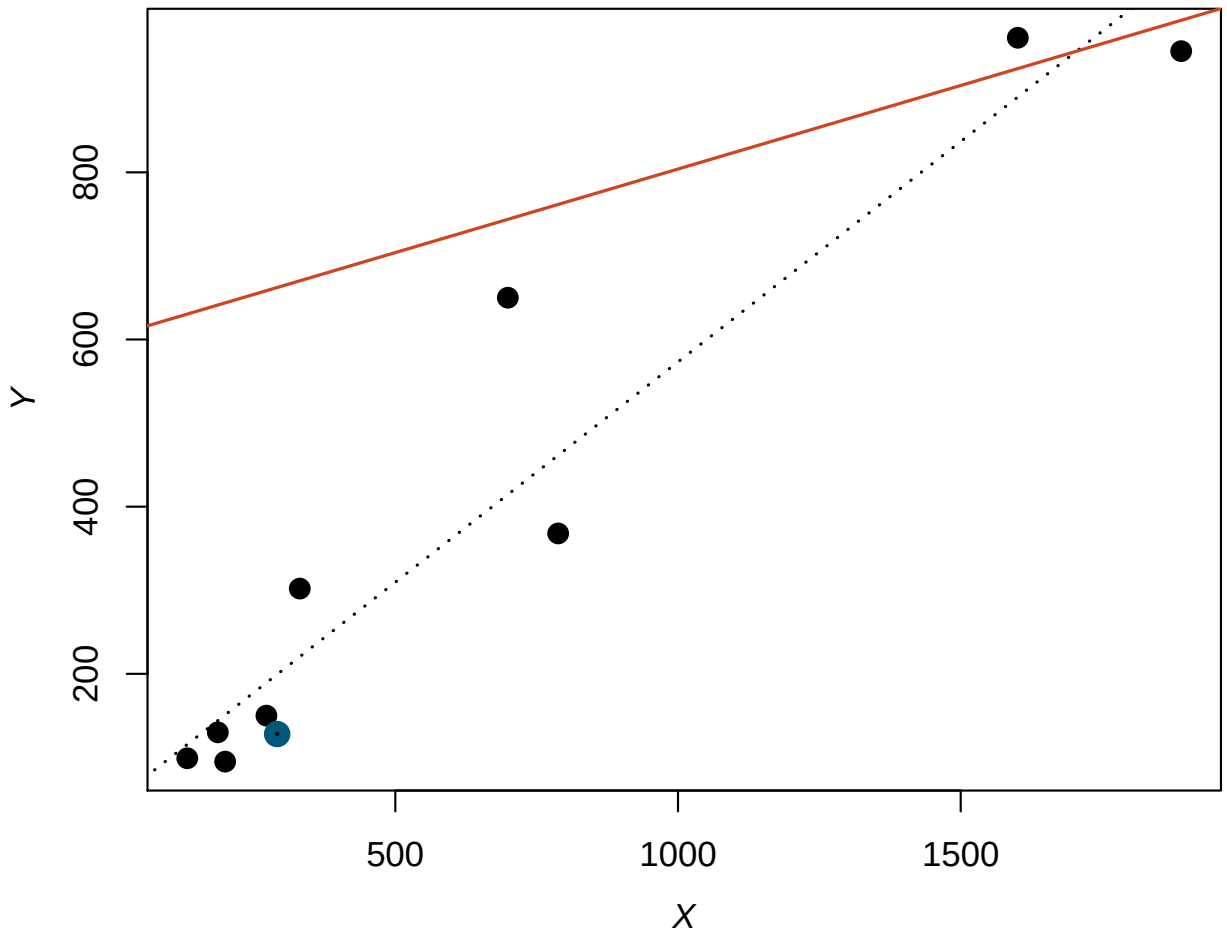
Szacowanie czasu programistów

iteracja 1



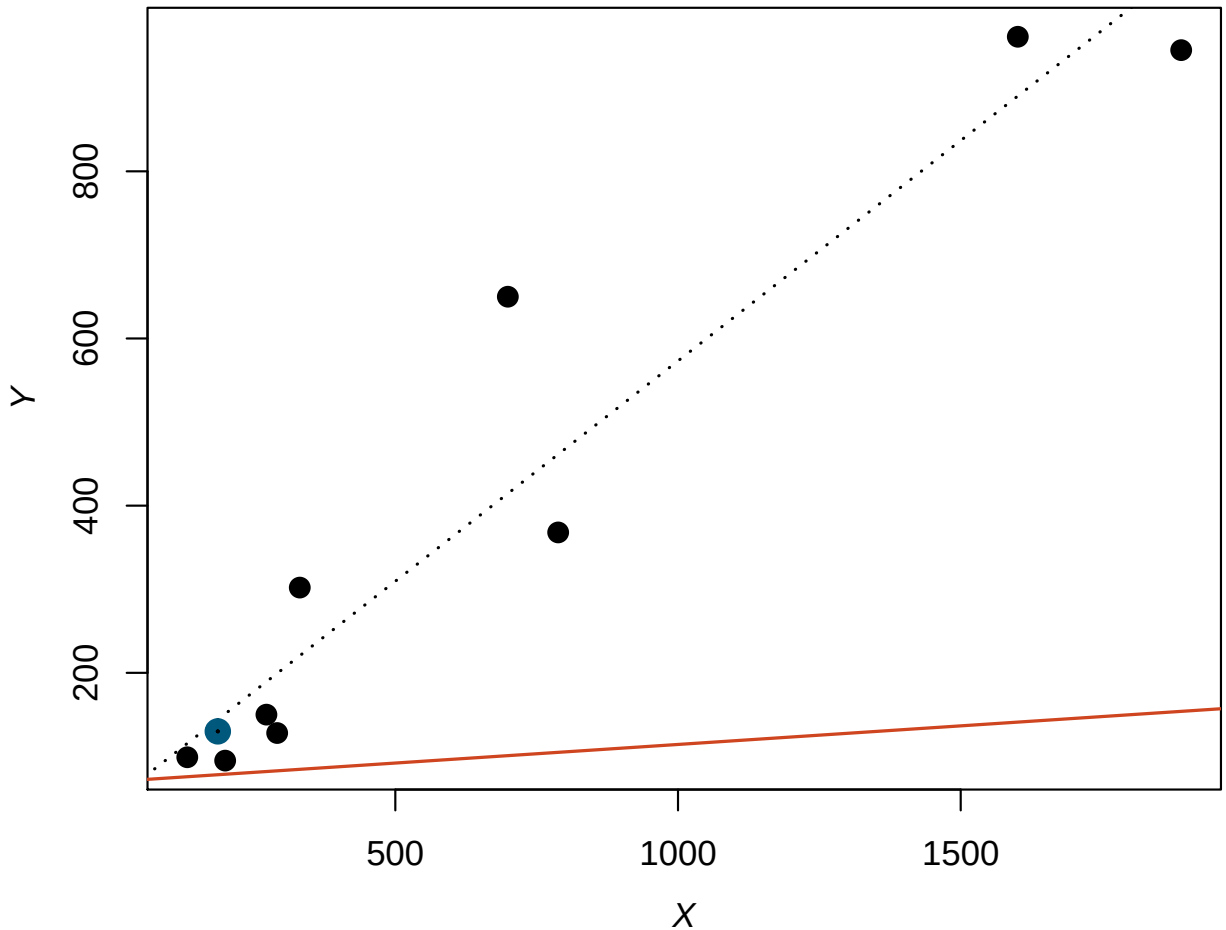
Szacowanie czasu programistów

iteracja 2



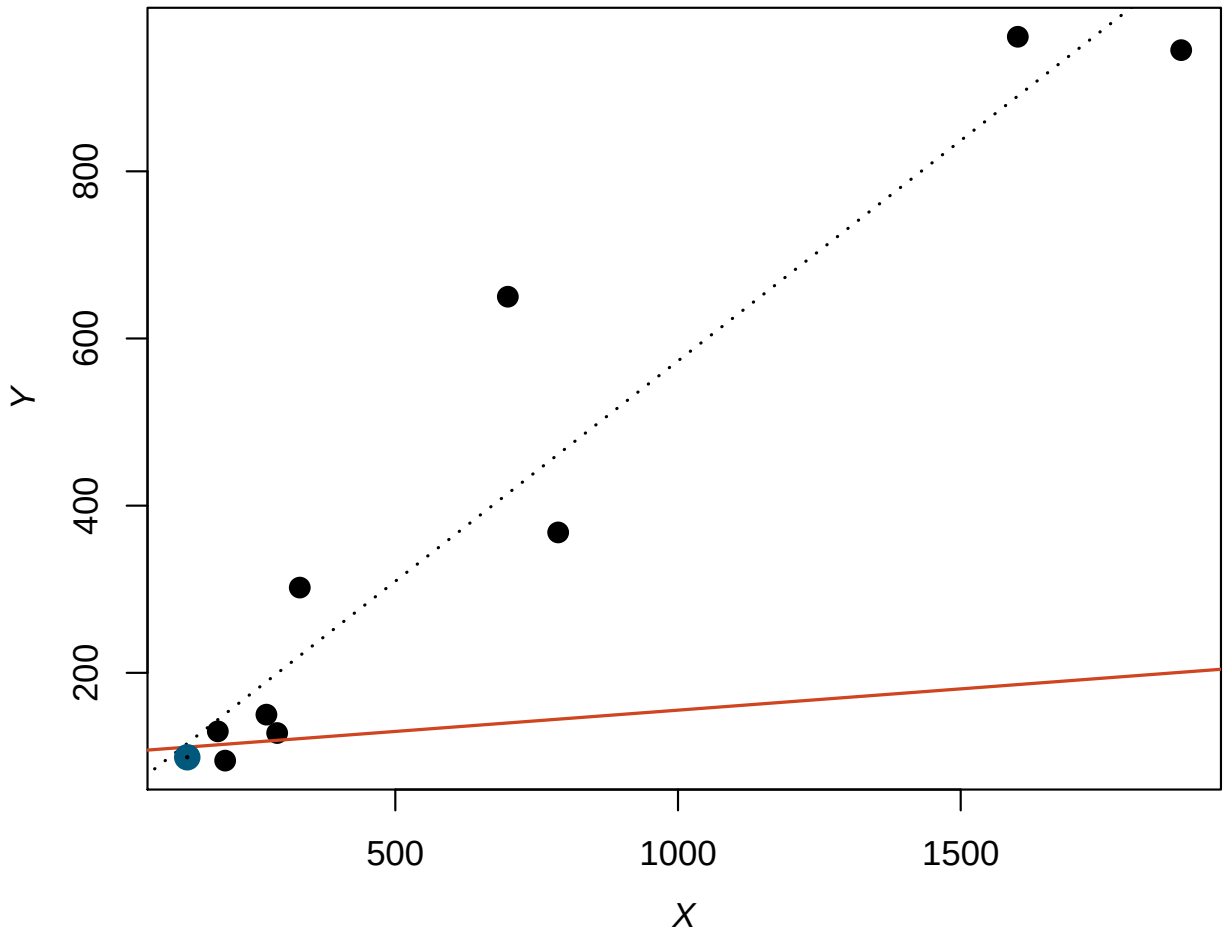
Szacowanie czasu programistów

iteracja 3



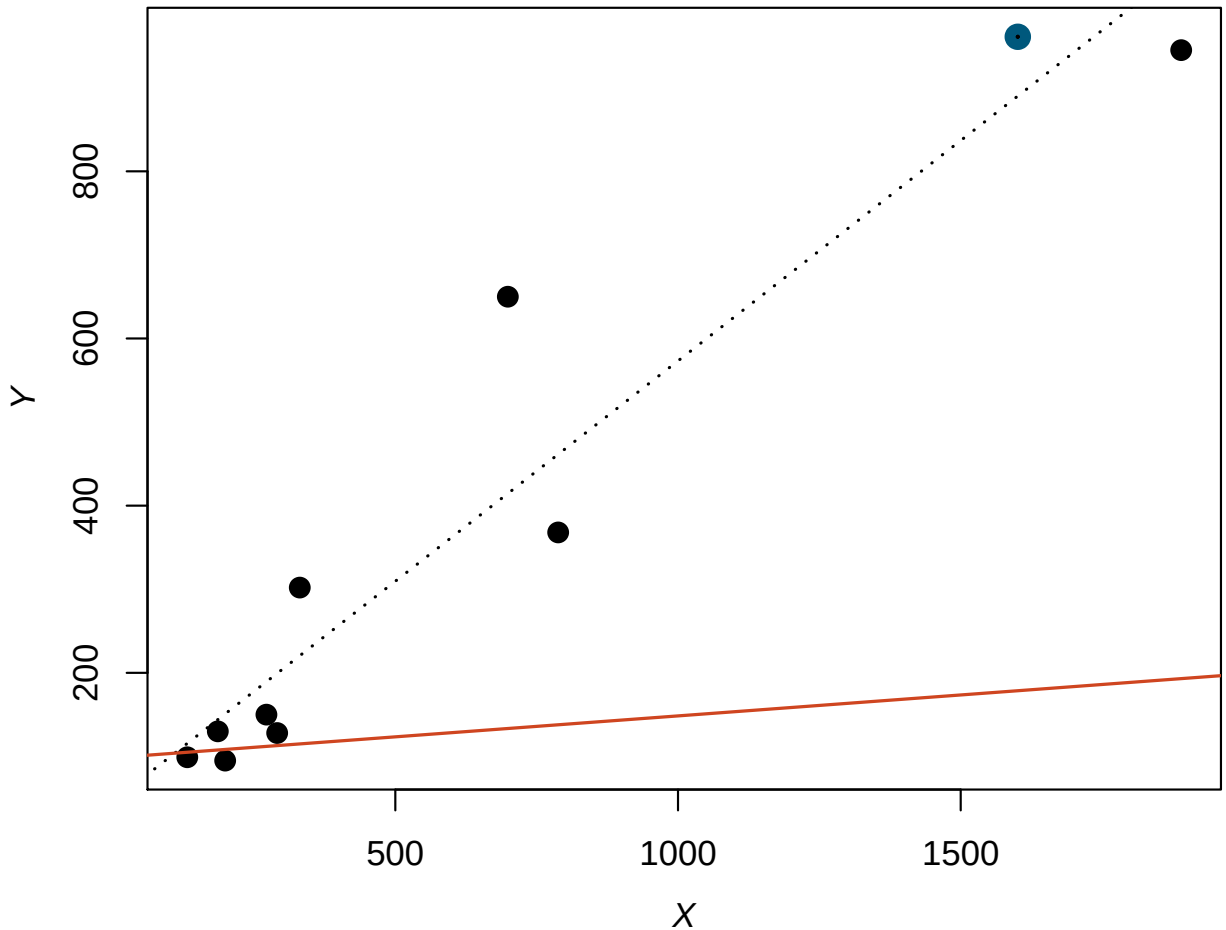
Szacowanie czasu programistów

iteracja 4



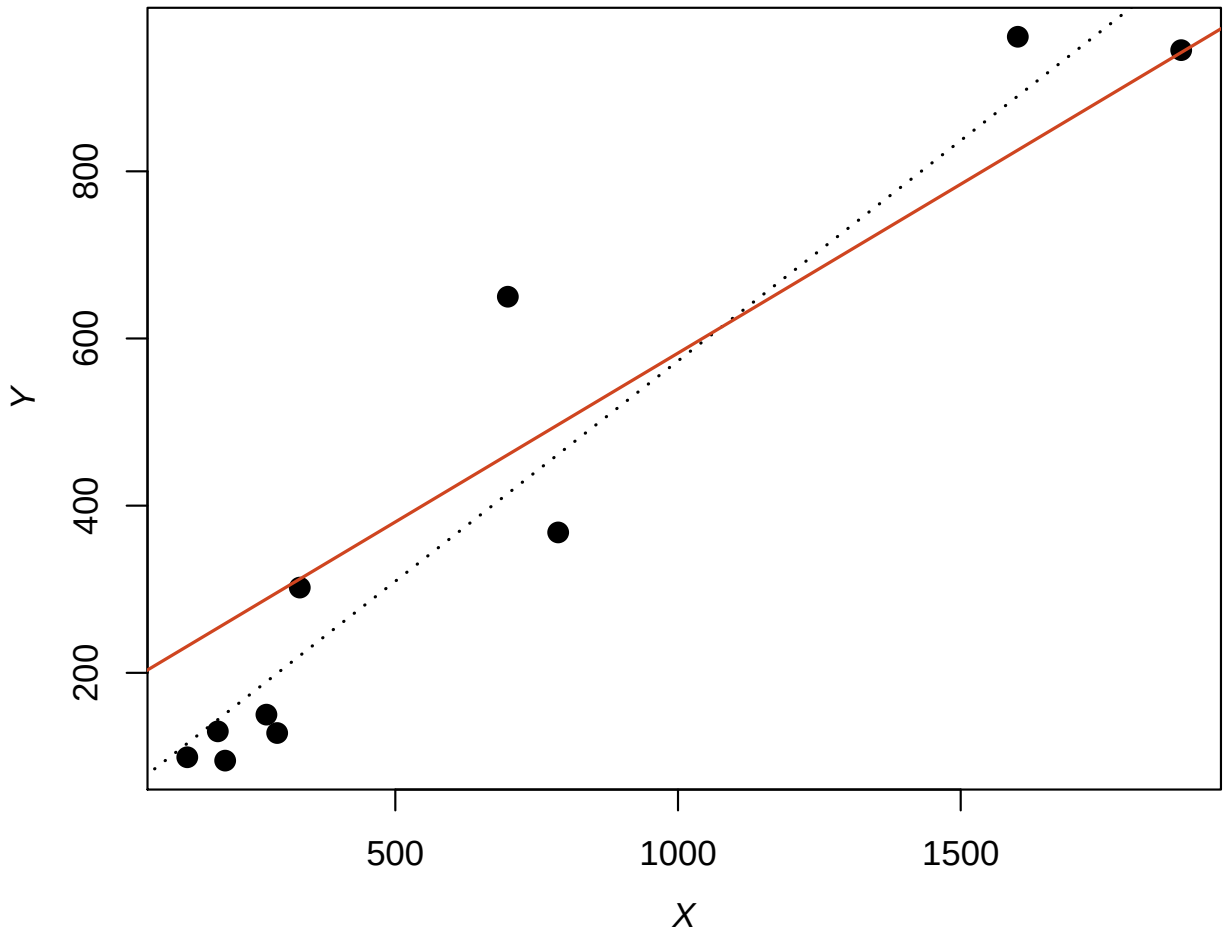
Szacowanie czasu programistów

iteracja 5



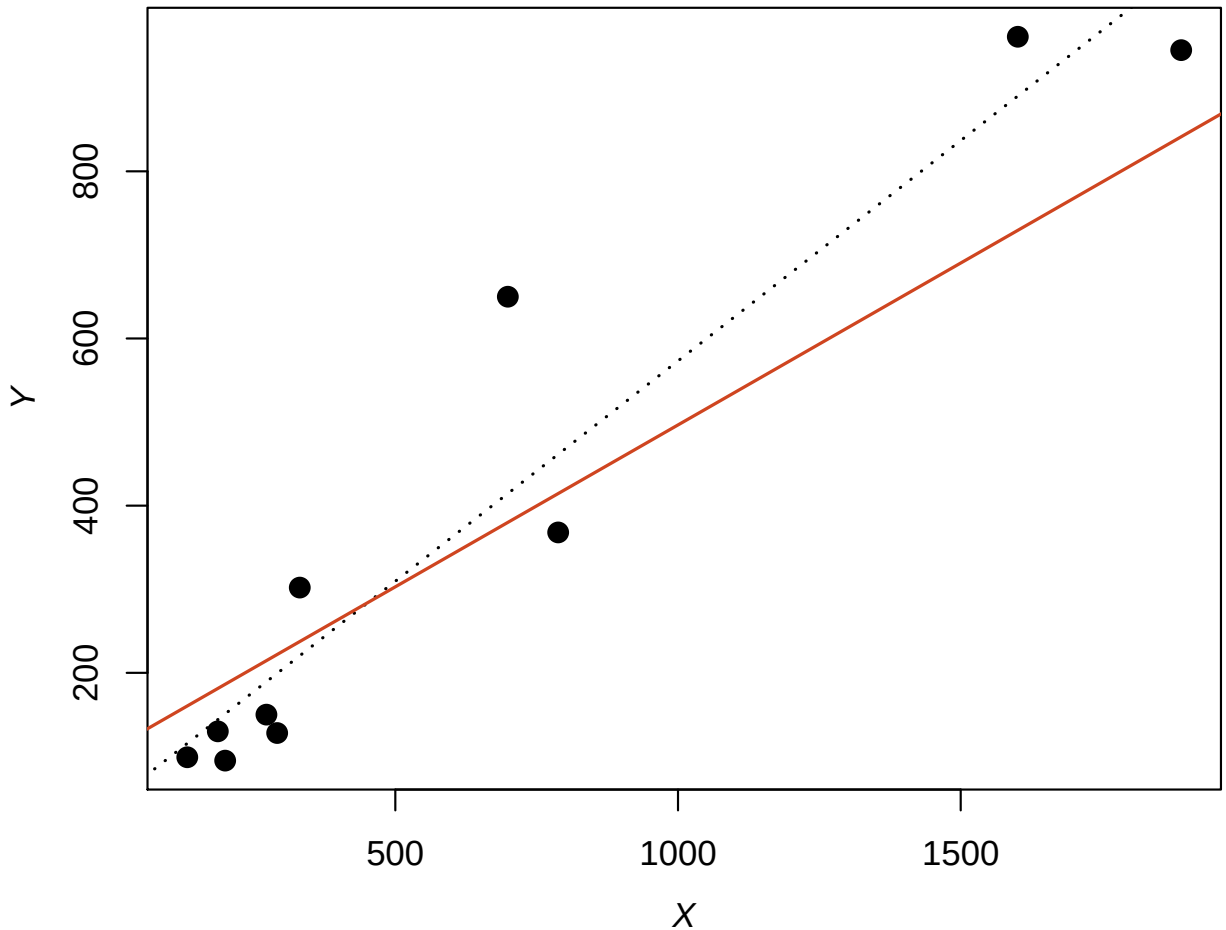
Szacowanie czasu programistów

iteracja 10



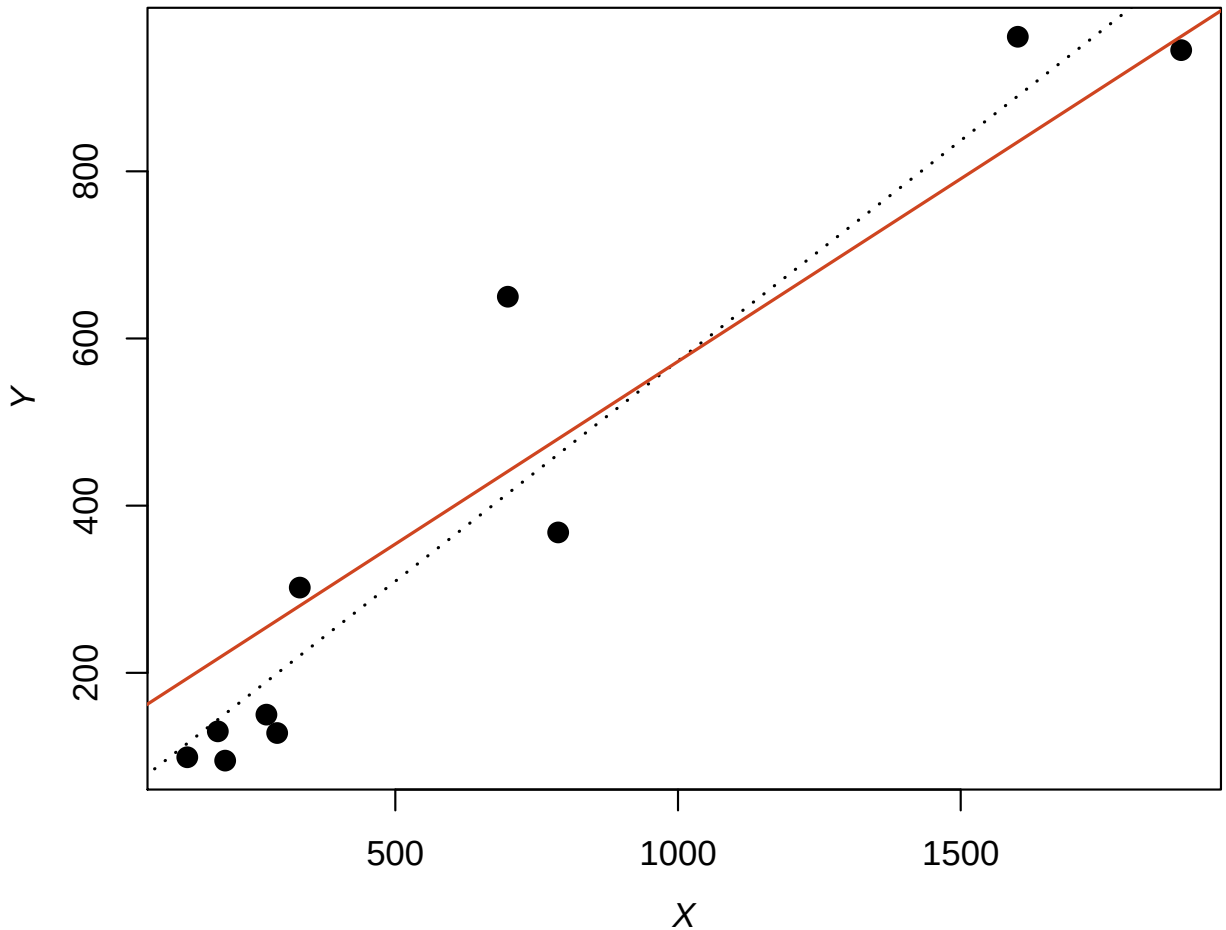
Szacowanie czasu programistów

iteracja 15



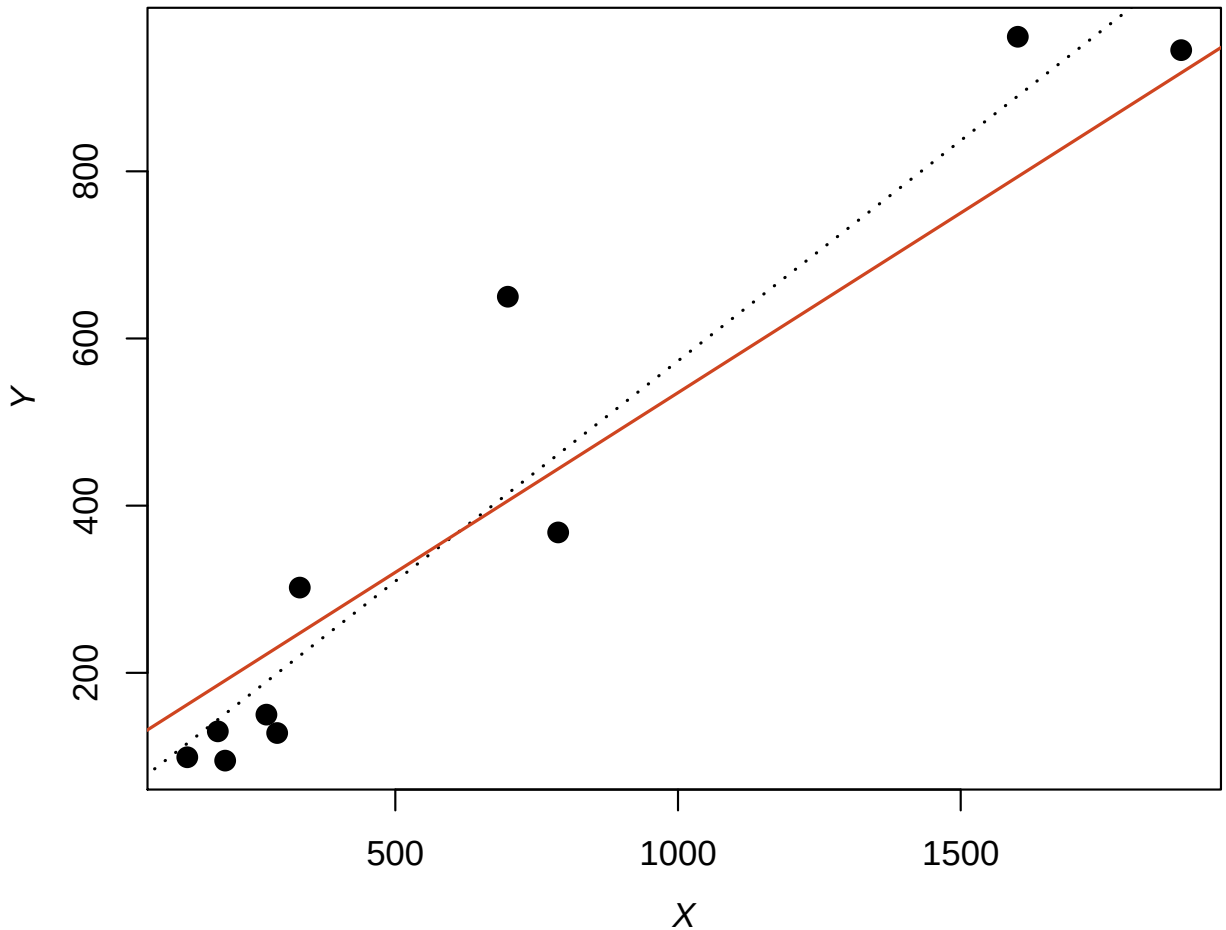
Szacowanie czasu programistów

iteracja 20



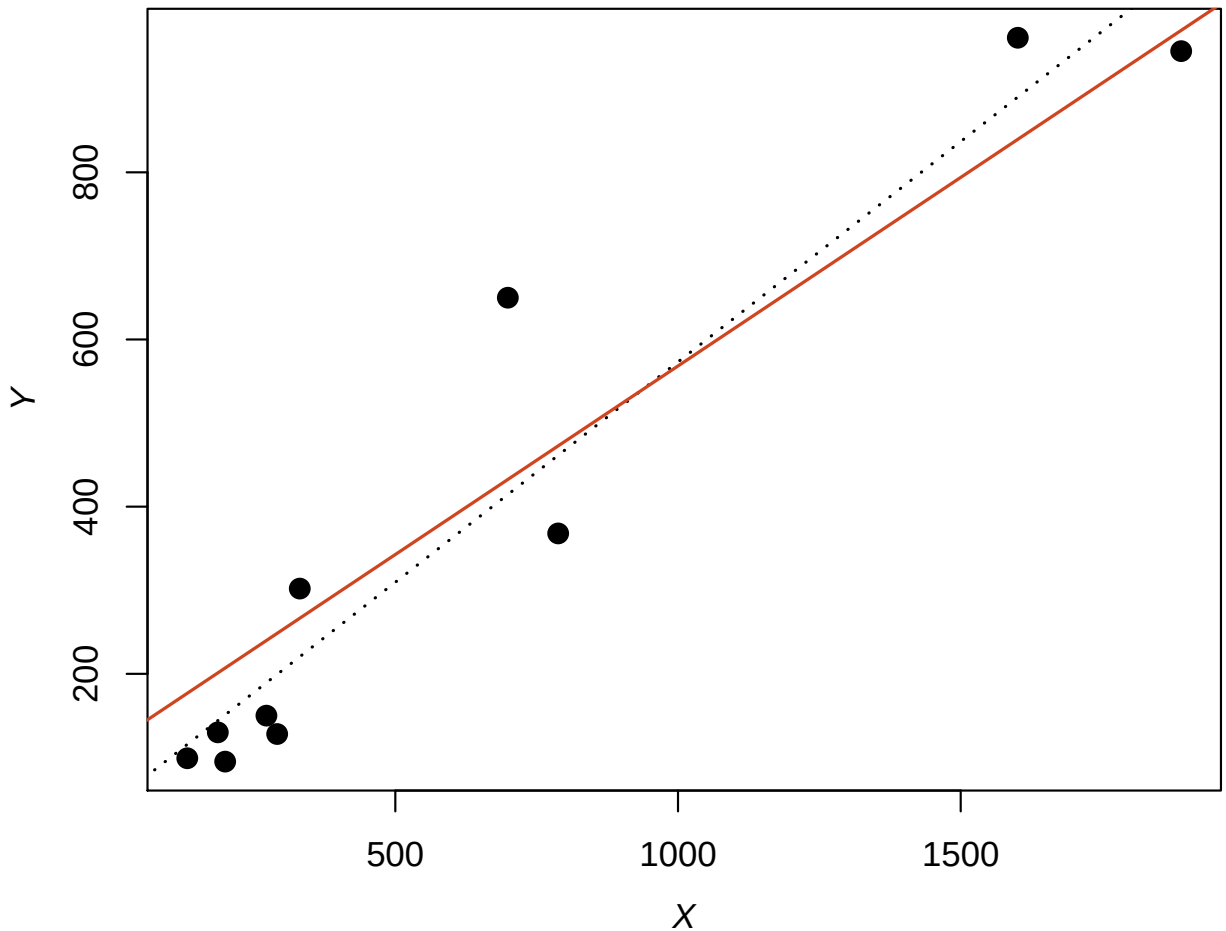
Szacowanie czasu programistów

iteracja 25



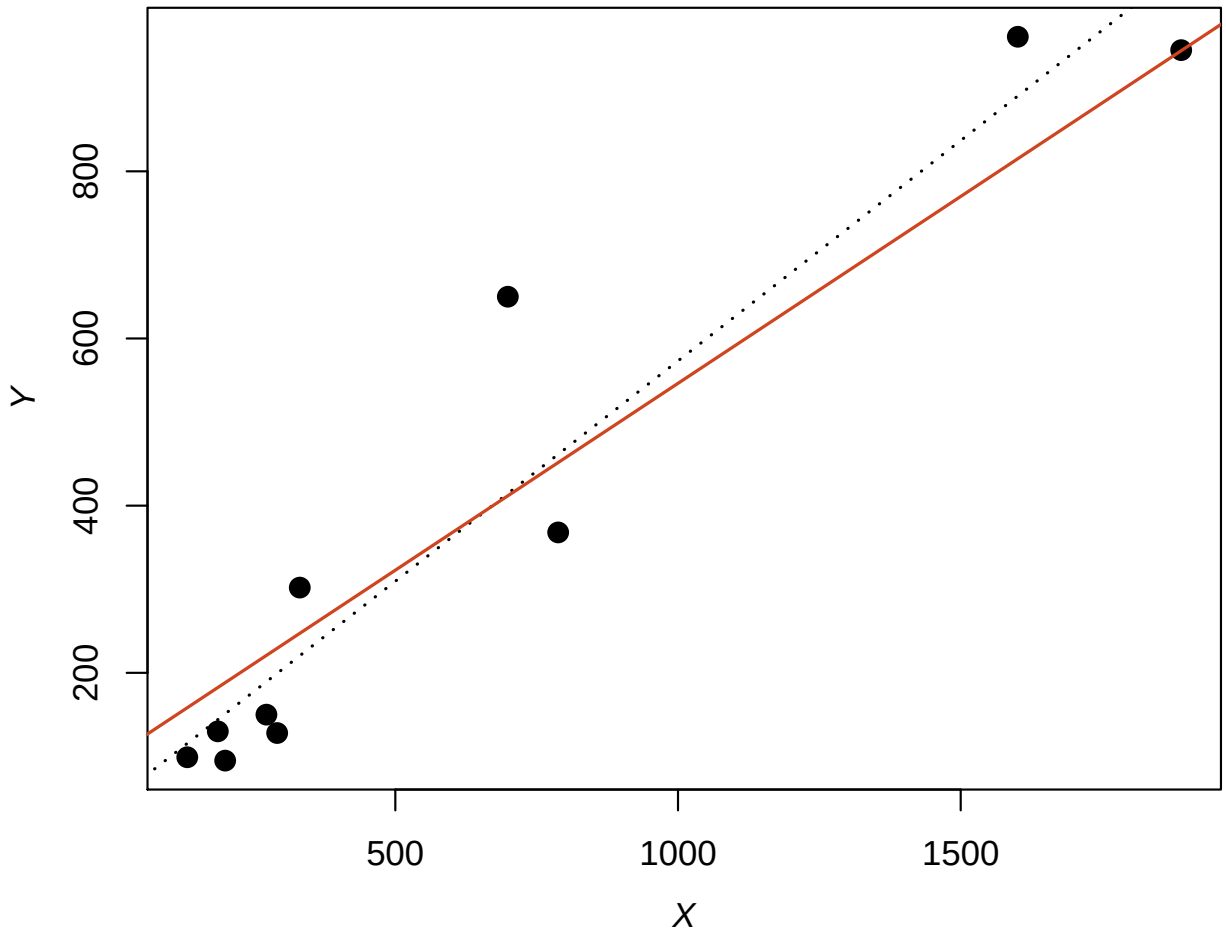
Szacowanie czasu programistów

iteracja 30



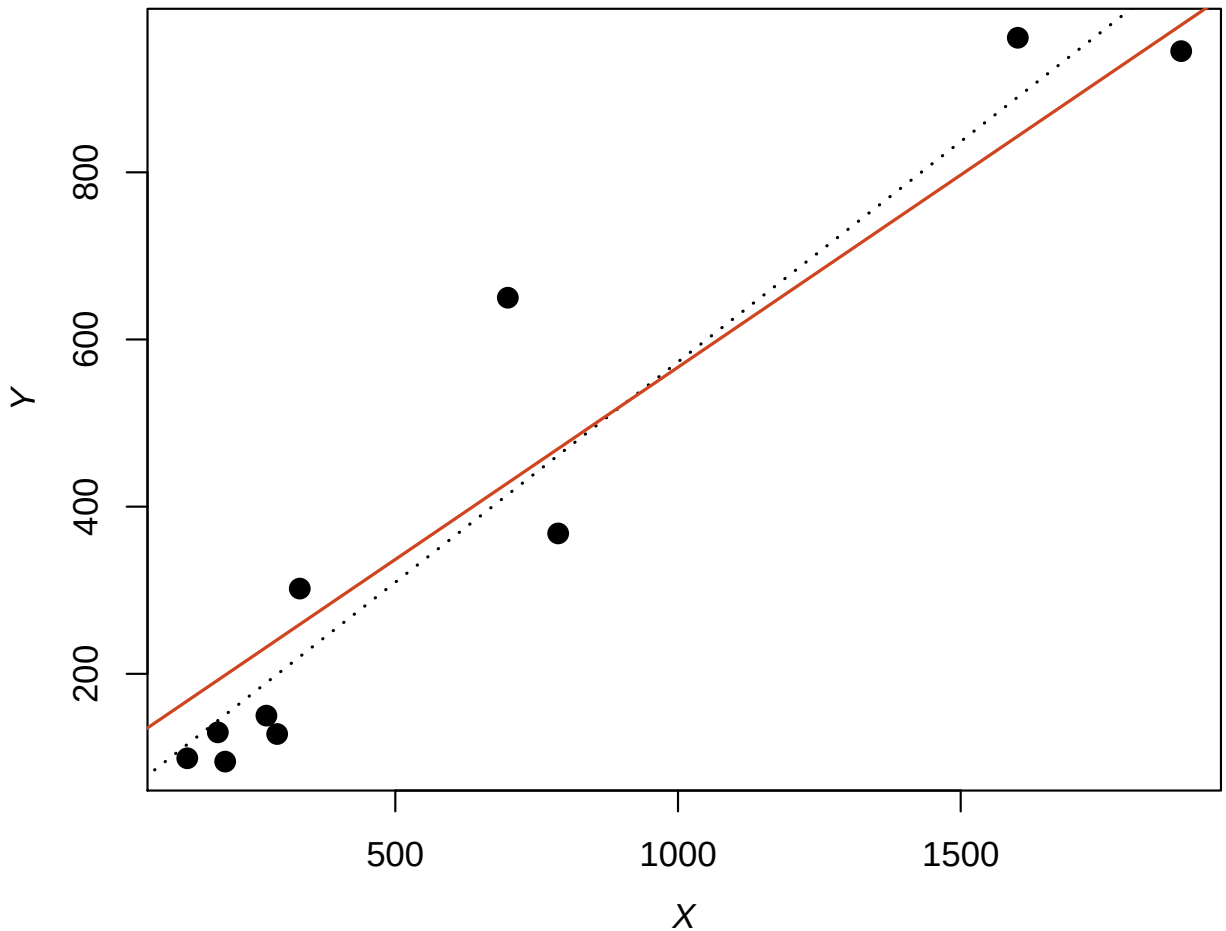
Szacowanie czasu programistów

iteracja 35



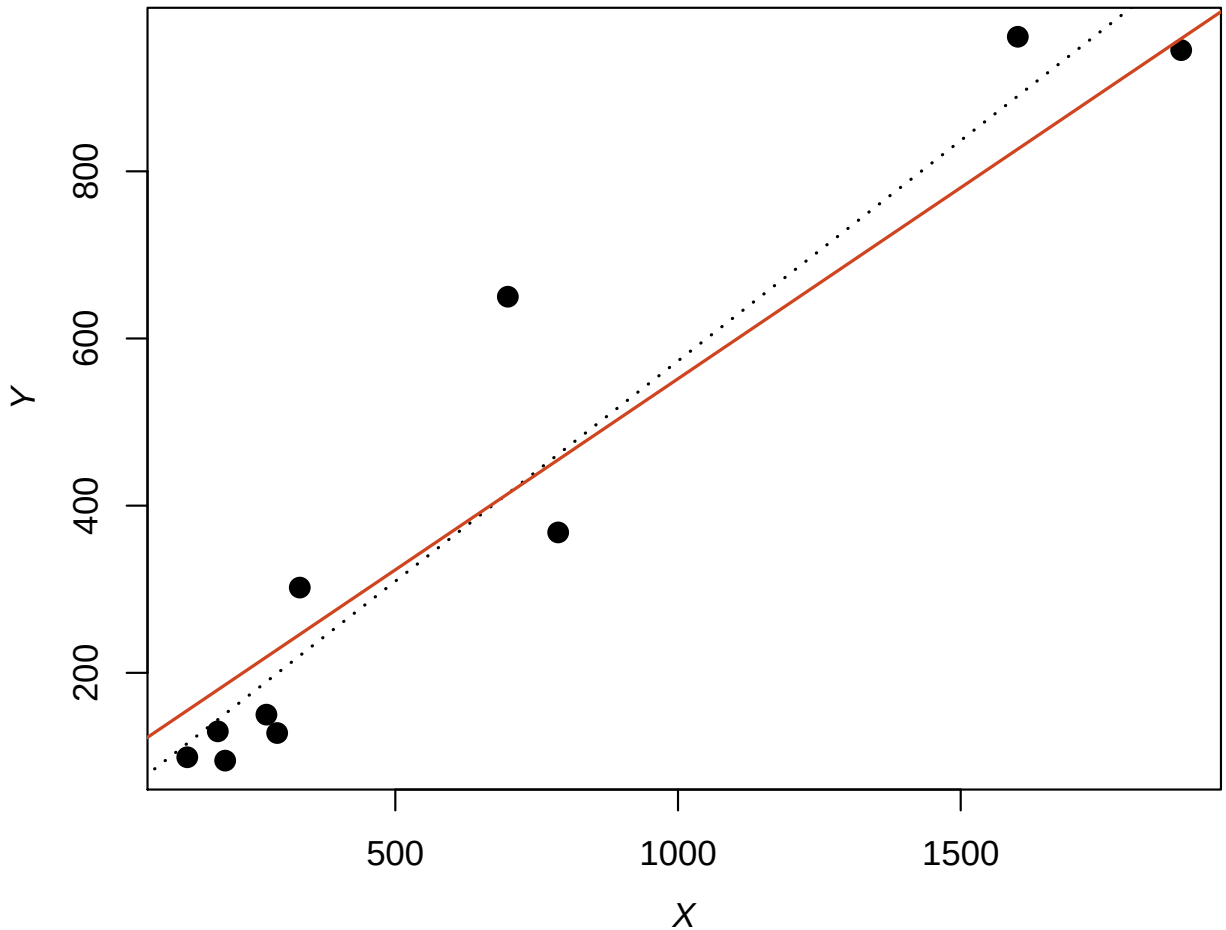
Szacowanie czasu programistów

iteracja 40



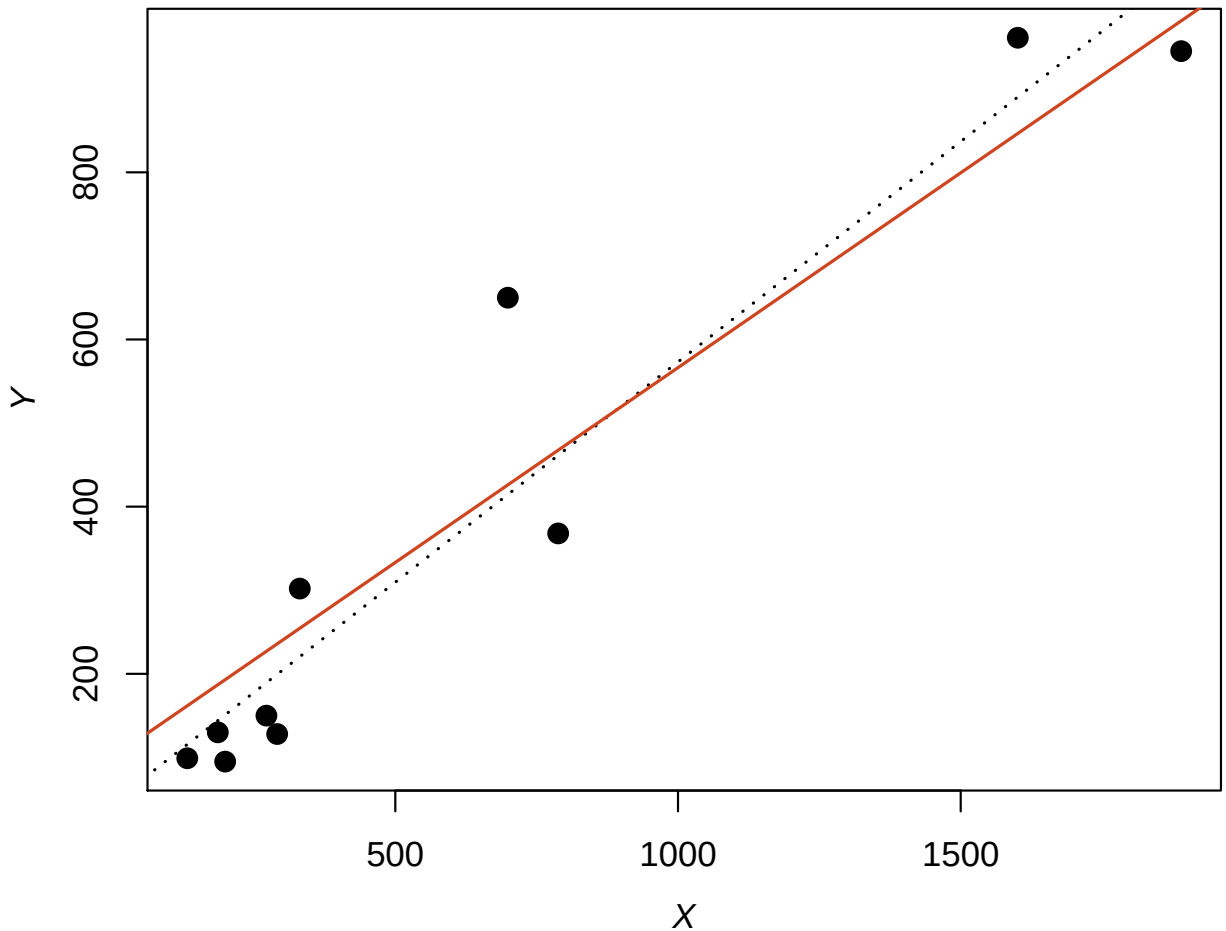
Szacowanie czasu programistów

iteracja 45



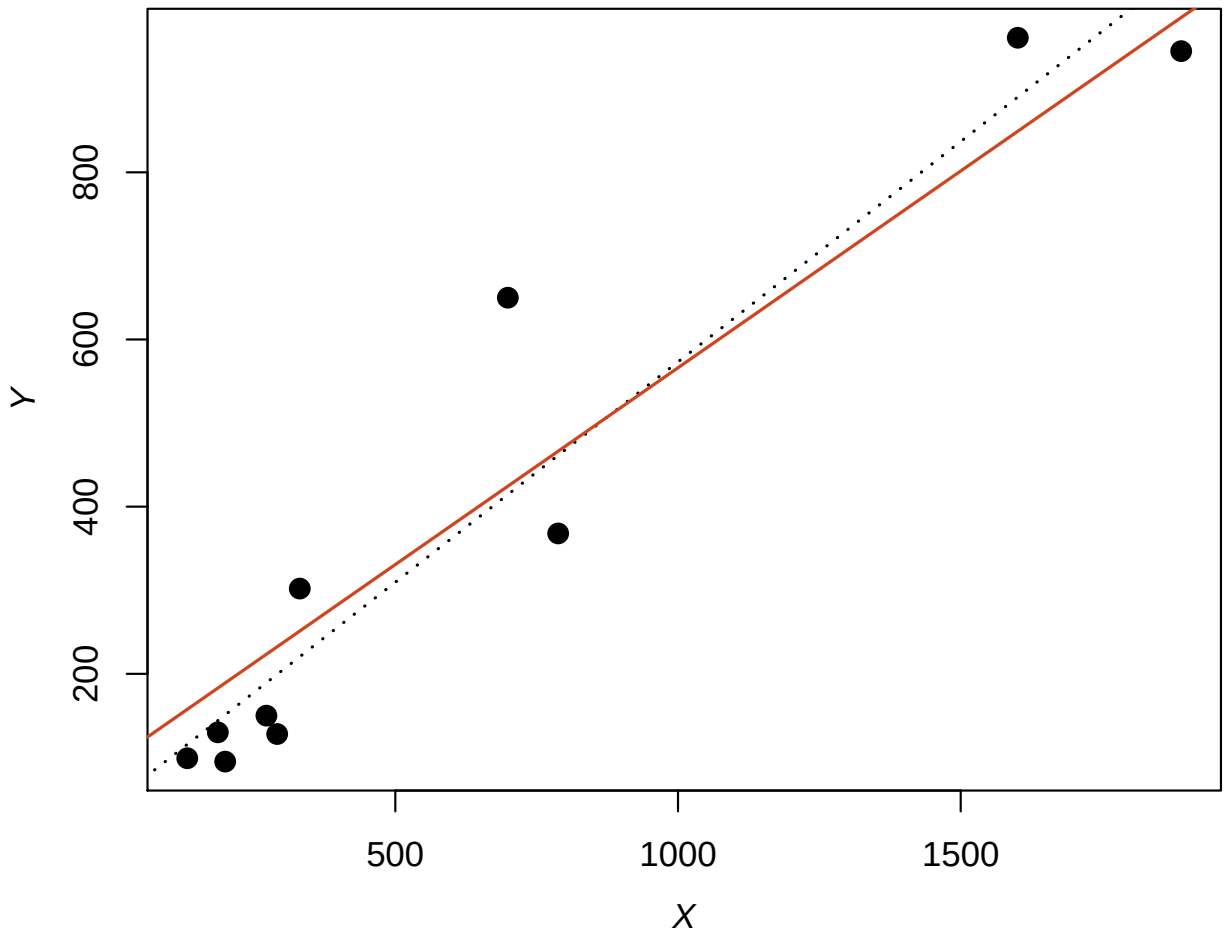
Szacowanie czasu programistów

iteracja 50



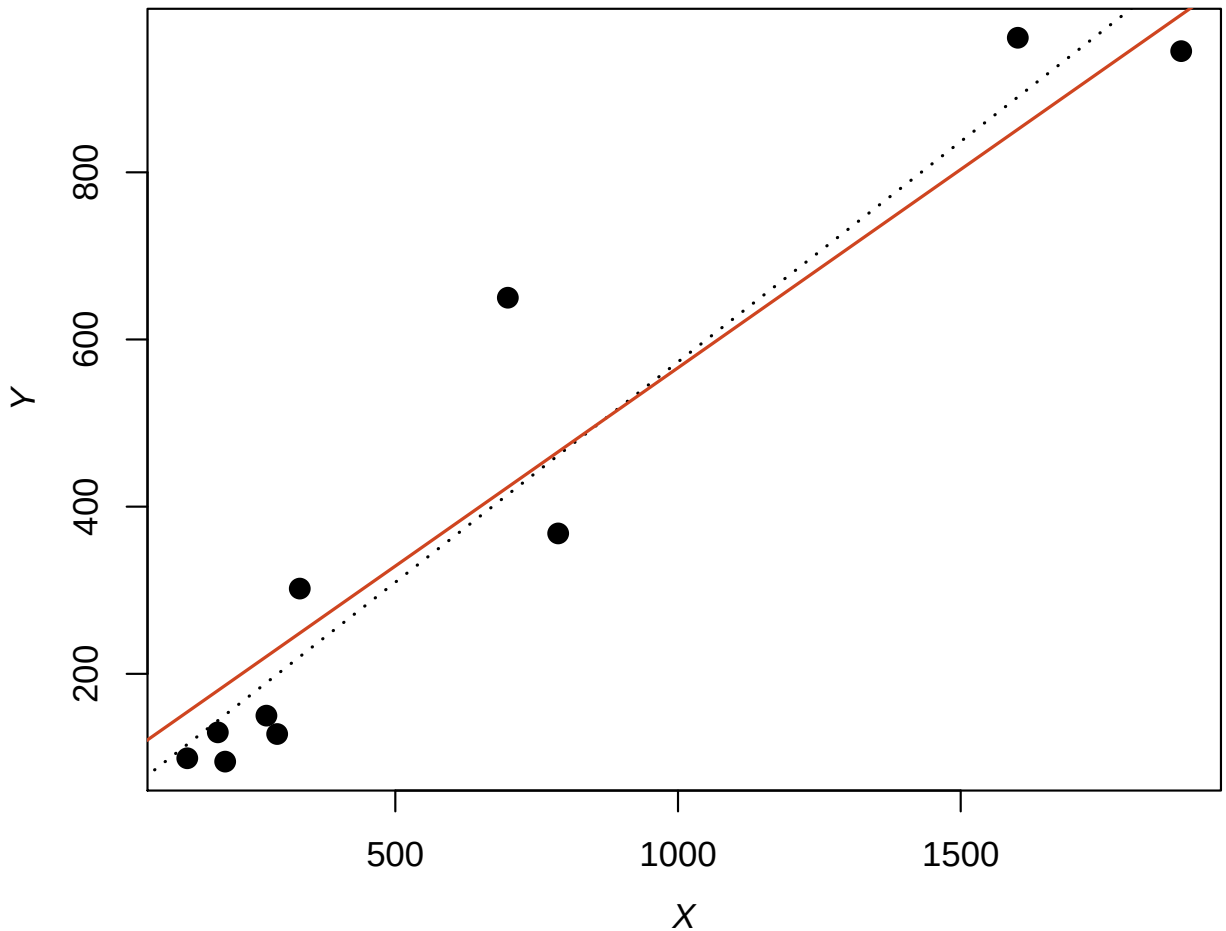
Szacowanie czasu programistów

iteracja 60



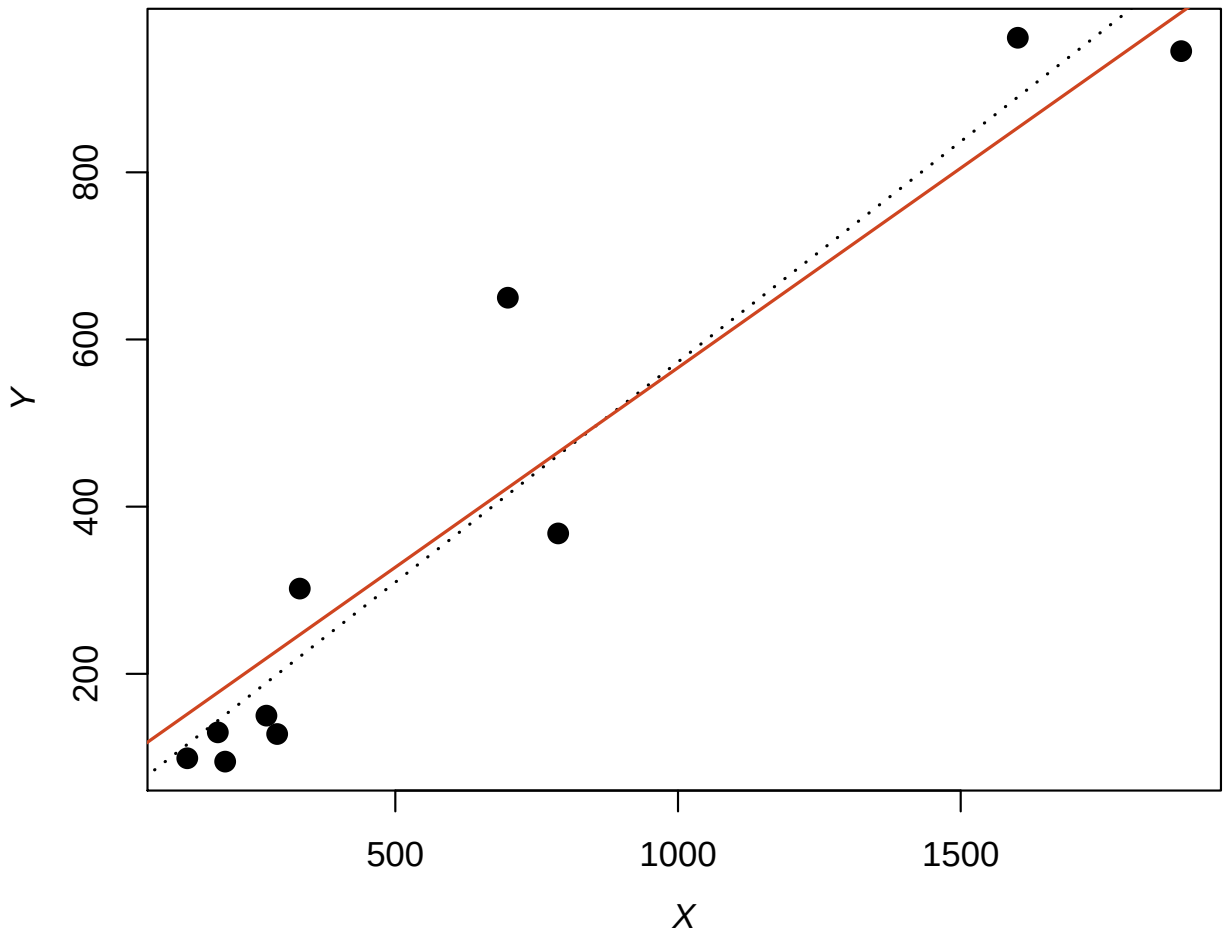
Szacowanie czasu programistów

iteracja 70



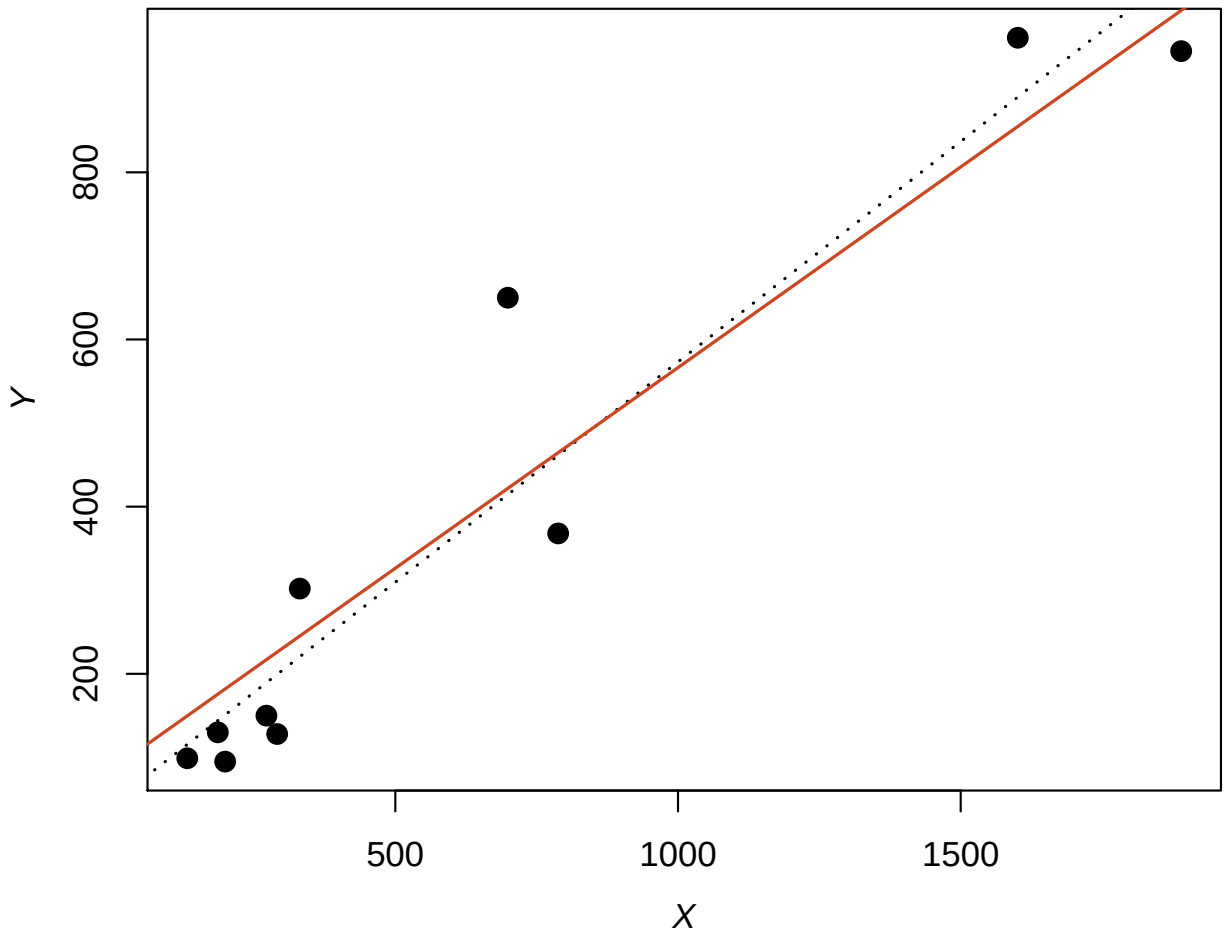
Szacowanie czasu programistów

iteracja 80



Szacowanie czasu programistów

iteracja 90



Szacowanie czasu programistów

iteracja 100