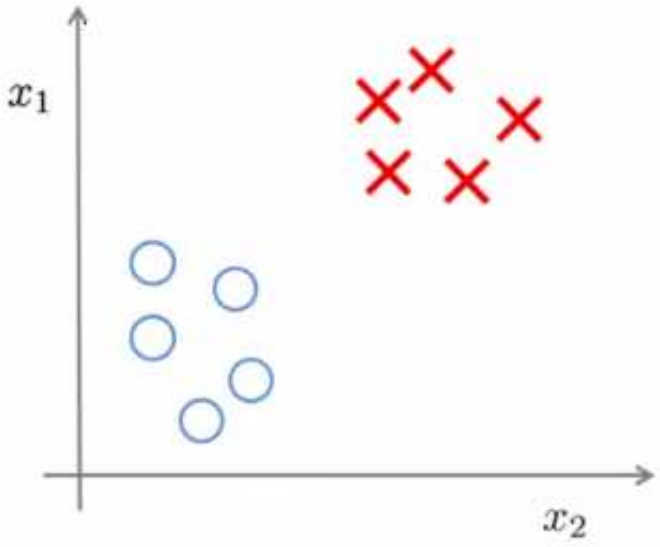
Uczenie nienadzorowane

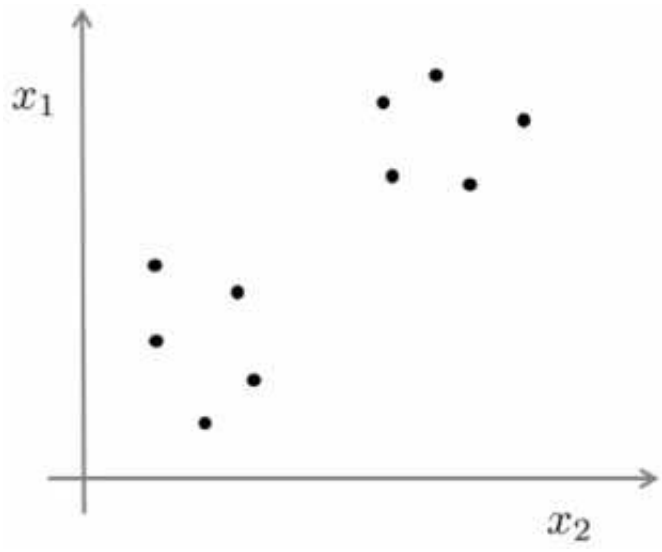
Nadzorowane, klasyfikacja:



Zbiór uczący:

$\{\langle(x_1^1, x_2^1), c^1\rangle, \langle(x_1^2, x_2^2), c^2\rangle, ...\langle(x_1^N, x_2^N), c^N\rangle\}$

Nienadzorowane, **analiza skupień** (*clustering*):



Zbiór uczący:

$\{\langle(x_1^1, x_2^1)\rangle, \langle(x_1^2, x_2^2)\rangle, ...\langle(x_1^N, x_2^N)\rangle\}$

Algorytm k-means

Bardzo prostą, popularną i skuteczną metodą analizy skupień jest algorytm **k-means**. Oparty jest na porównywaniu odległości i wyznaczaniu skupień reprezentowanych przez ich środki geometryczne — **centroidy**, minimalizujących pewną funkcję jakości.

Algorytm zakłada, że liczba skupień K, które należy wygenerować jest znana. Algorytm wykonuje powtarzalnie dwa kroki: krok etykietowania i krok przesunięcia centroidów.

Algorytm k-means:

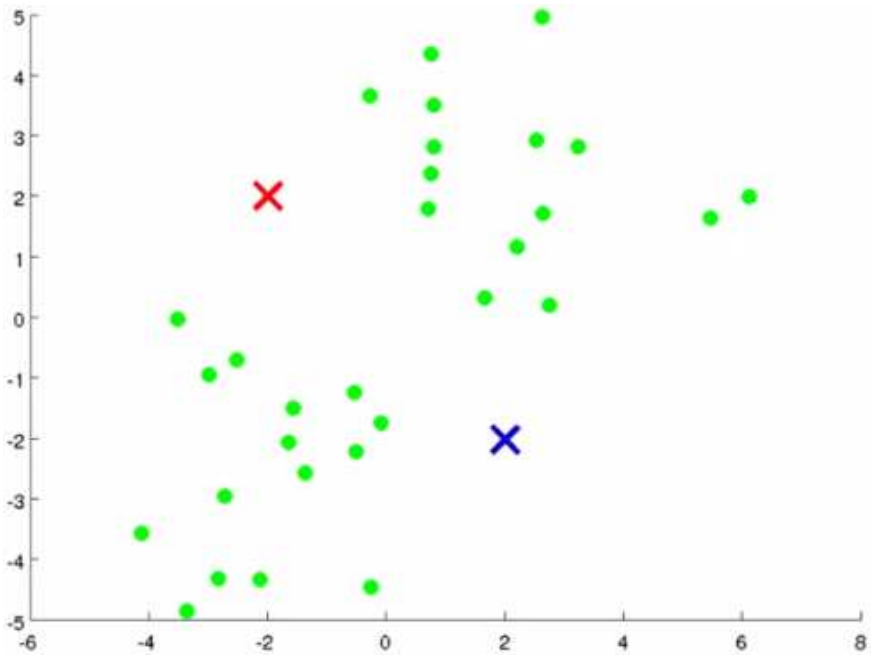
Krok 0 (inicjalizacja): ustaw wartości początkowe wszystkich K centroidów

REPEAT {

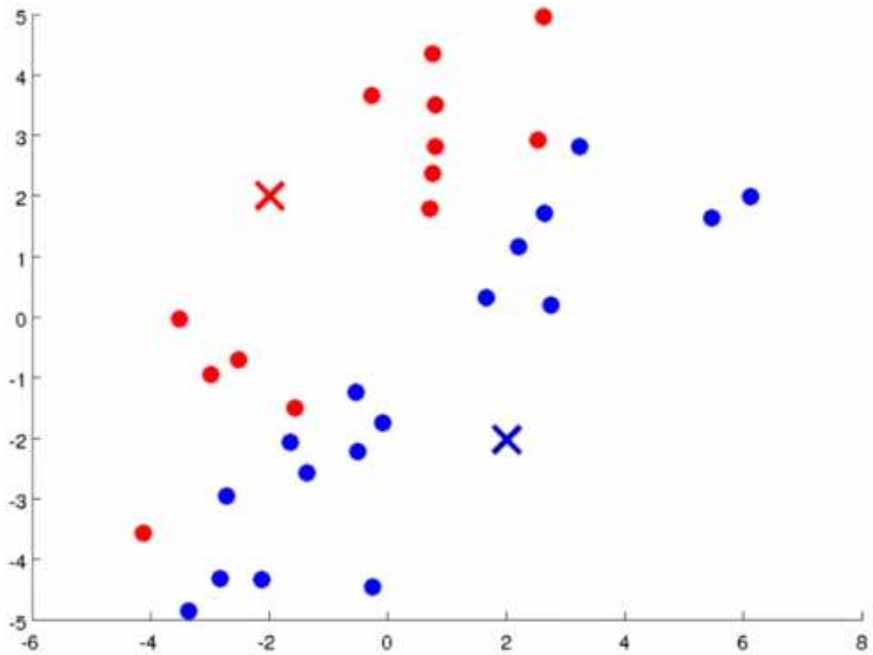
Krok 1 (etykietowanie): oznacz wszystkie próbki etykietą najbliższego centroidu

Krok 2 (przesunięcie centroidów): przesuń wszystkie centroidy do geometrycznego środka ich skupień

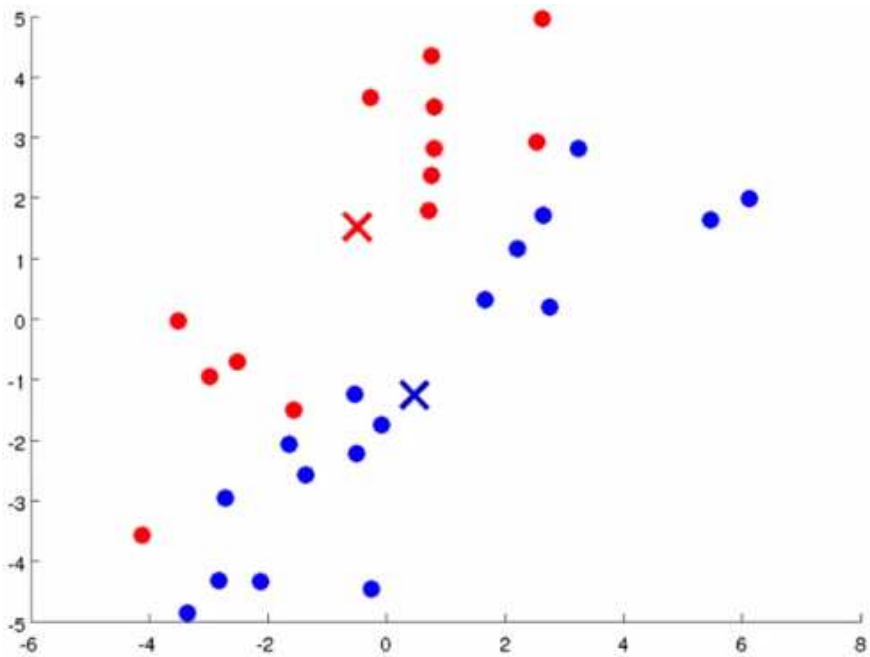
}



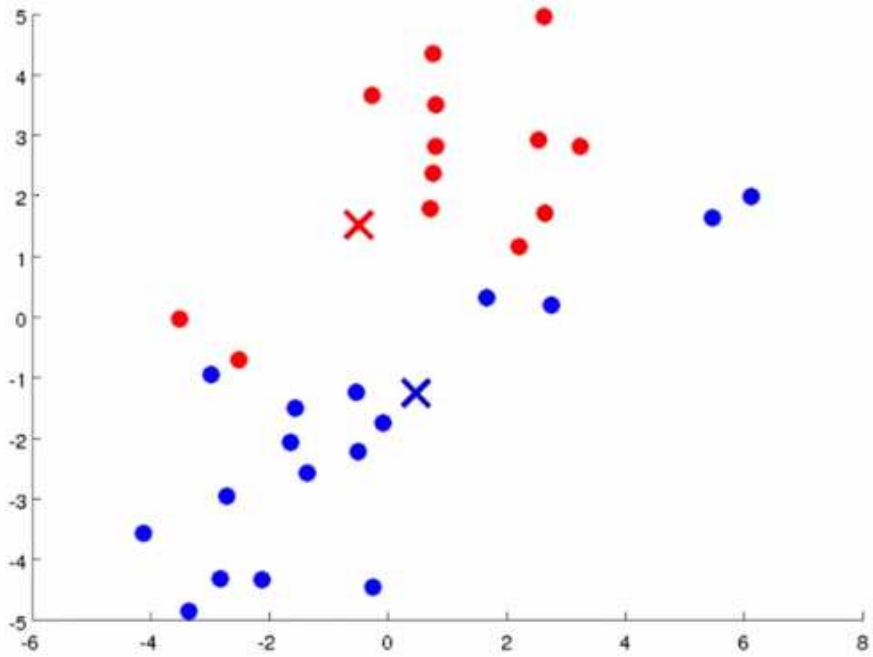
Krok 0 (inicjalizacja)



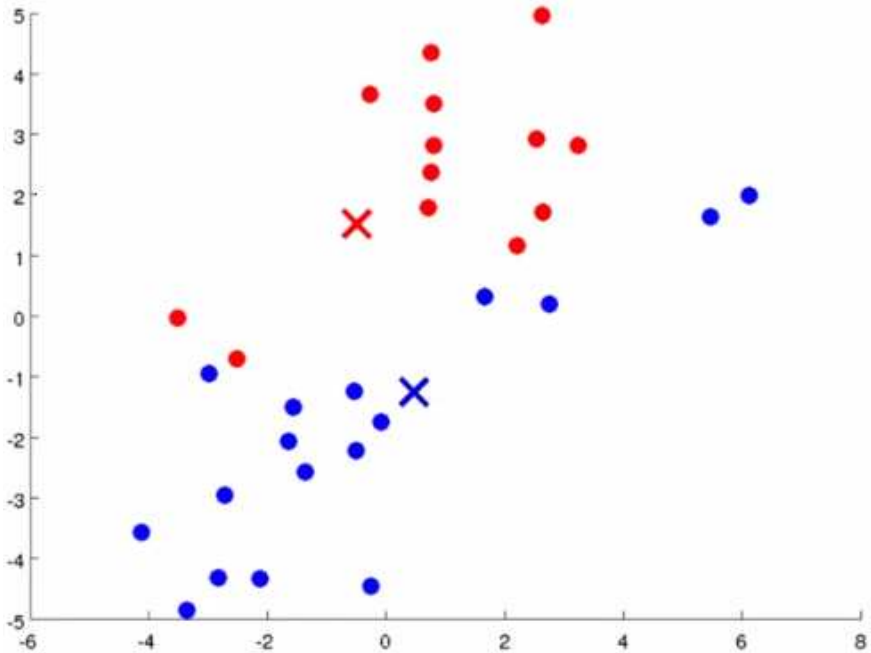
Krok 1 (etykietowanie)



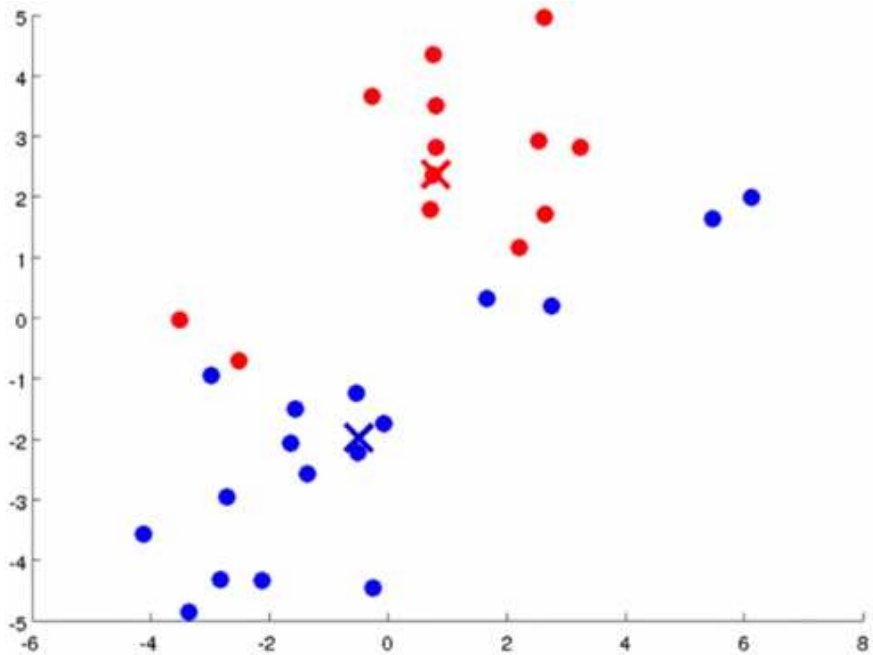
Krok 2 (przesunięcie centroidów)



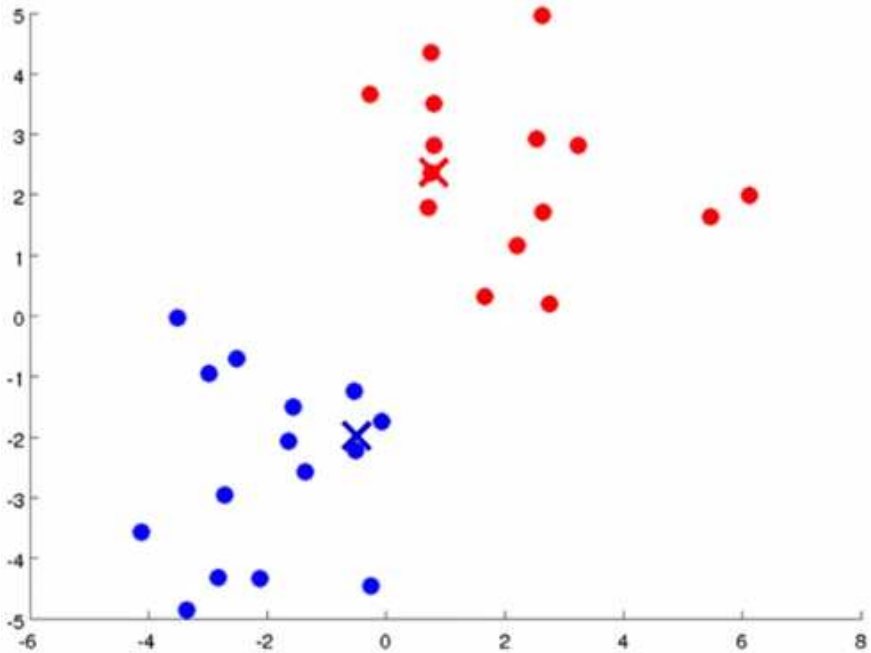
Krok 1 (etykietowanie)



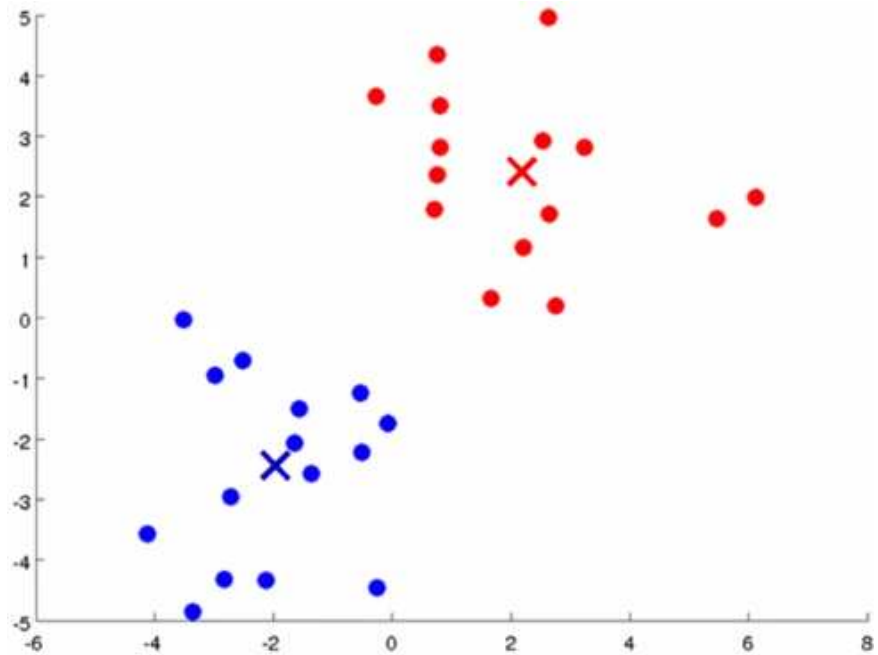
Krok 1 (etykietowanie)



Krok 2 (przesunięcie centroidów)



Krok 1 (etykietowanie)



Krok 2 (przesunięcie centroidów)

Algorytm k-means — kryterium jakości

Algorytm k-means usiłuje znaleźć minimum pewnej funkcji kosztu, która jest miarą jakości wygenerowanego zbioru skupień. Ta funkcja kosztu jest ważoną sumą odległości wszystkich punktów od centroidów ich skupień.

Można zauważyć, że pierwszy krok algorytmu (etykietowanie) dokonuje optymalizacji tej funkcji kosztu ze względu na składowe odległości, przy zachowaniu aktualnych centroidów.

Drugi krok algorytmu (przesunięcie centroidów) dokonuje optymalizacji tej samej funkcji ze względu na położenie centroidów, przy zachowaniu aktualnych zbiorów punktów wszystkich skupień.

Algorytm k-means — pomiar odległości

Można stosować różne miary do obliczania odległości w algorytmie k-means:

euklidesowa	$\sqrt{\Sigma_i (a_i - b_i)^2}$
Manhattan	$\Sigma_i \lvert a_i - b_i \rvert$
max	$\max_i \lvert a_i - b_i \rvert$

Ogólnie: ze względu na możliwą rozbieżność wielkości, podobnie jak w przypadku innych metod opartych na obliczaniu odległości, poszczególne współrzędne należy skalować do obliczania odległości w przestrzeni cech. Współczynnikiem skalowania może być wariancja wartości danej współrzędnej na zbiorze treningowym.

Specjalnym problemem są dane nienumeryczne. W niektórych przypadkach, jak np. napisy tekstowe, istnieje szereg metryk dla nich dedykowanych Przykładami prostych metryk odległości stringów są: odległość Hamminga i odległość Levenshteina.

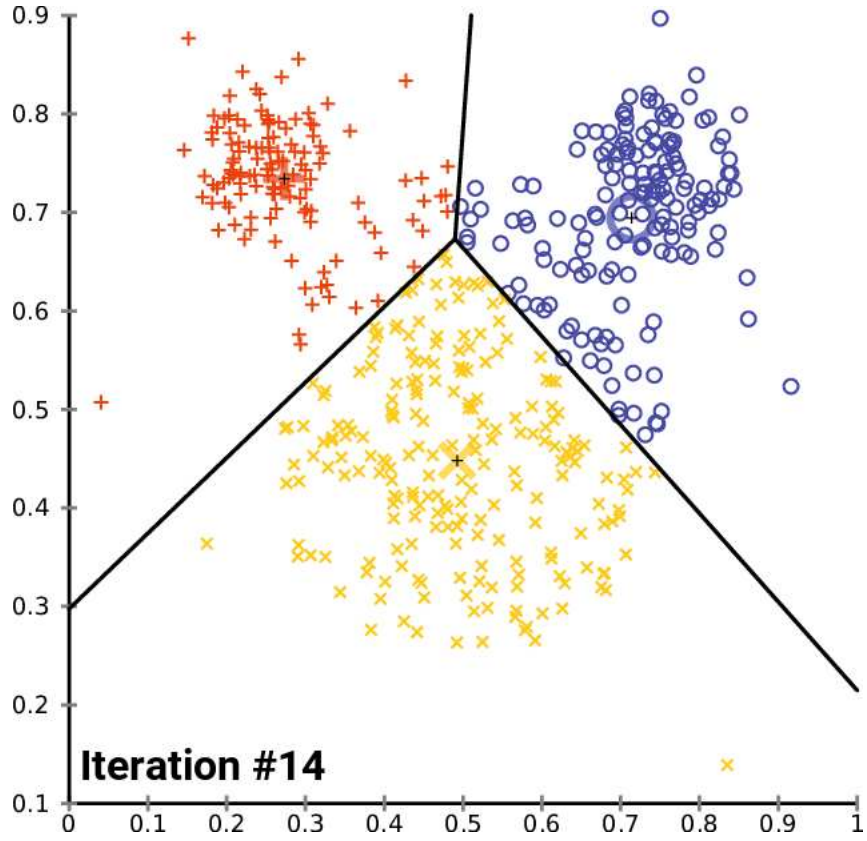
Odległość Hamminga (tylko dla stringów równej długości) = liczba pozycji znakowych, na których stringi się różnią. Jest ona równa minimalnej liczbie podmian pojedynczych znaków potrzebnej do przekształcenia jednego stringa w drugi.

Odległość Levenshteina (dla dowolnych stringów) = minimalna liczba podmian, wstawień, oraz usunięć pojedynczych znaków potrzebna do przekształcenia jednego stringa w drugi.

Algorytm k-means — inny przykład

Przykład z Wikipedii:

`https://upload.wikimedia.org/wikipedia/commons/e/ea/K-means_convergence.gif`



Jak widać na powyższym przykładzie, k-means nie zawsze generuje tak intuicyjnie poprawne wyniki, jak na wcześniejszych przykładach. Istnieje szereg okoliczności specjalnych, które należy/warto uwzględnić, aby otrzymać optymalne wyniki.

Przypadek specjalny k-means — centroid ze zbiorem pustym

Co robić, gdy w trakcie pracy algorytmu powstanie centroid z pustym zbiorem punktów?

Metoda 1: pominąć ten centroid w dalszym ciągu.

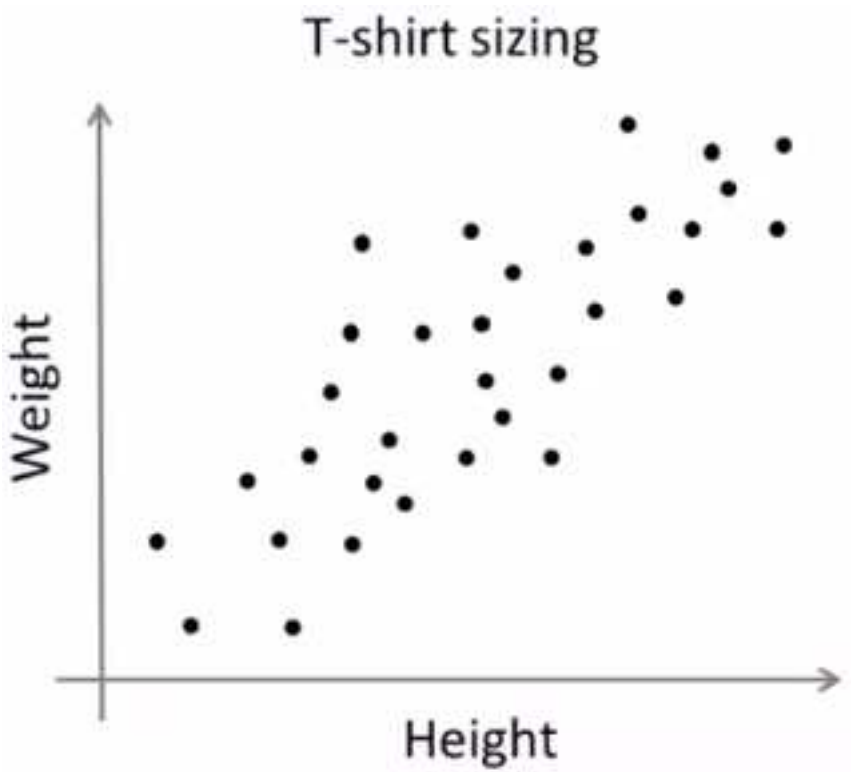
Jednak jest możliwe, że liczba skupień jest narzucona, i chcemy ją utrzymywać. Wtedy:

Metoda 2: ponownie zainicjalizować położenie tego centroidu, i kontynuować.

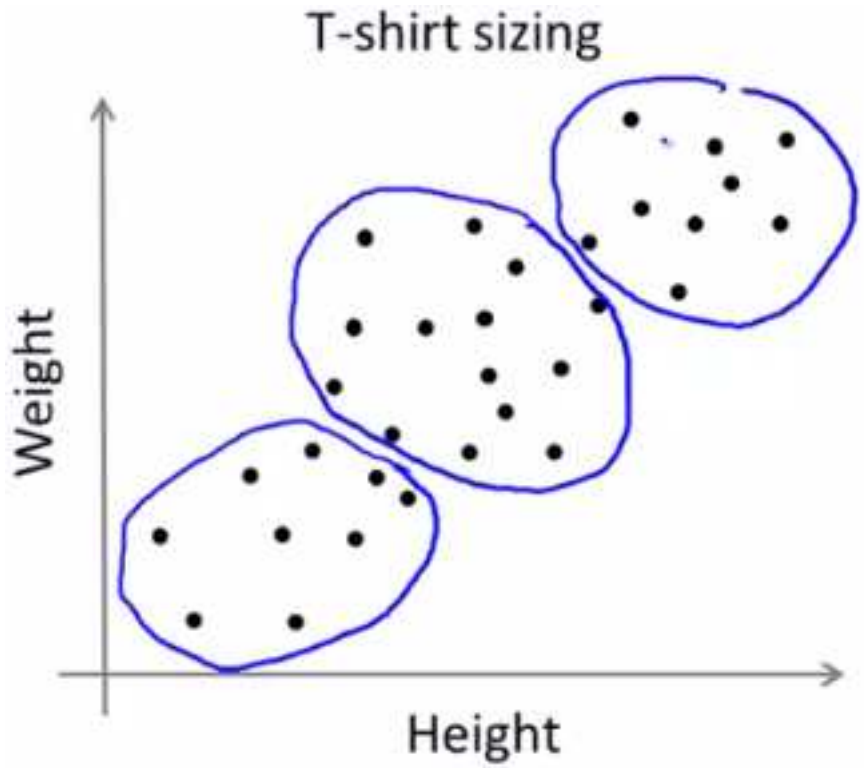
Przypadek specjalny k-means — brak separacji skupień

Nie zawsze zbiór próbek układa się w zdecydowanie odseparowane skupienia. Możemy chcieć mimo wszystko pogrupować dane.

Np. producent koszulek T-shirt zrobił badania antropometryczne aby zaprojektować dobrze dopasowane koszulki w kilku rozmiarach (np.: S,M,L):

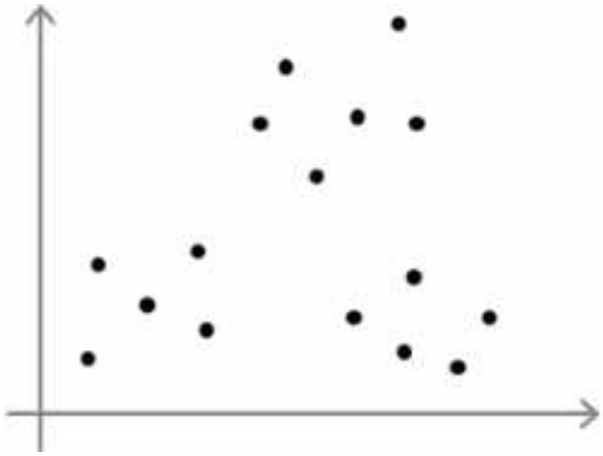


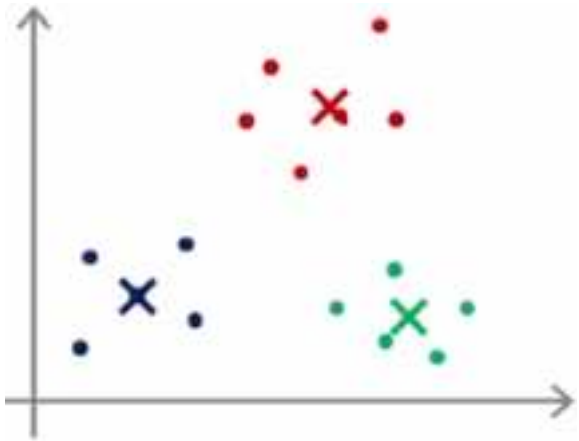
Algorytm nadal działa poprawnie, znajdując zadaną liczbę skupień w oparciu o odległości:



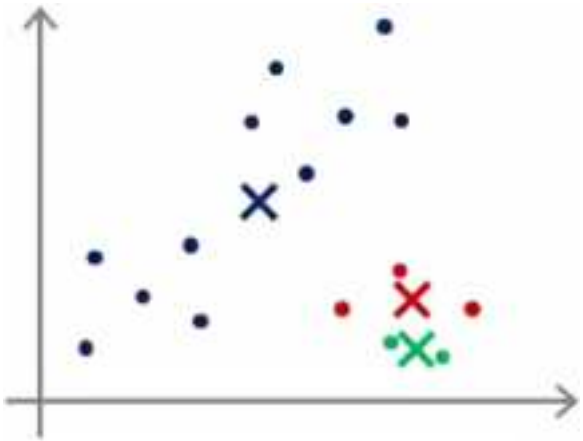
Algorytm k-means — inicjalizacja

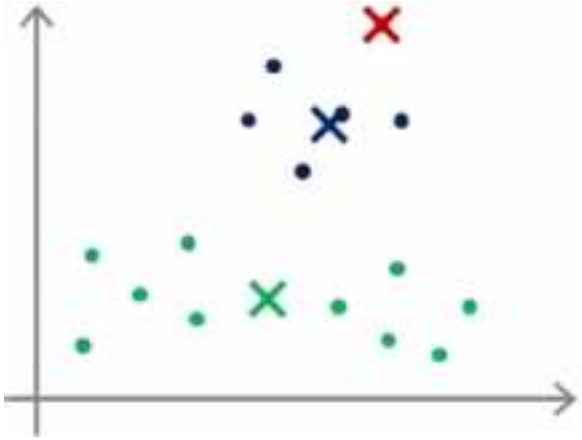
W najprostszym przypadku inicjalizacja może być przypadkowa, np. dowolne K próbek zbioru treningowego. Jednak nie zawsze daje to dobre wyniki.





W przypadku jak powyżej po lewej można uzyskać pożądane rozwiązanie (powyżej po prawej). Jednak niefortunna inicjalizacja może wygenerować rozwiązania jak poniżej.





Algorytm k-means — inicjalizacja (cd.)

Jak zapobiec niefortunnej inicjalizacji, która może prowadzić do wygenerowania nieoptymalnych skupień, osiągających lokalne maksimum funkcji kosztu?

Podobnie jak w metodzie wyżarzania, można porzucić wygenerowane centroidy, i wybrać je ponownie losowo. Jednak aby porównać miarę jakości (funkcję kosztu, czyli ważoną sumę odległości wszystkich punktów od centroidów ich skupień), należy doprowadzić algorytm w obu przypadkach do końca.

W praktyce oznacza to wielokrotne (100?, 1000 razy?) powtórzenie algorytmu k-means dla losowo wybranych punktów startowych, i wybraniu globalnie najlepszego rozwiązania.

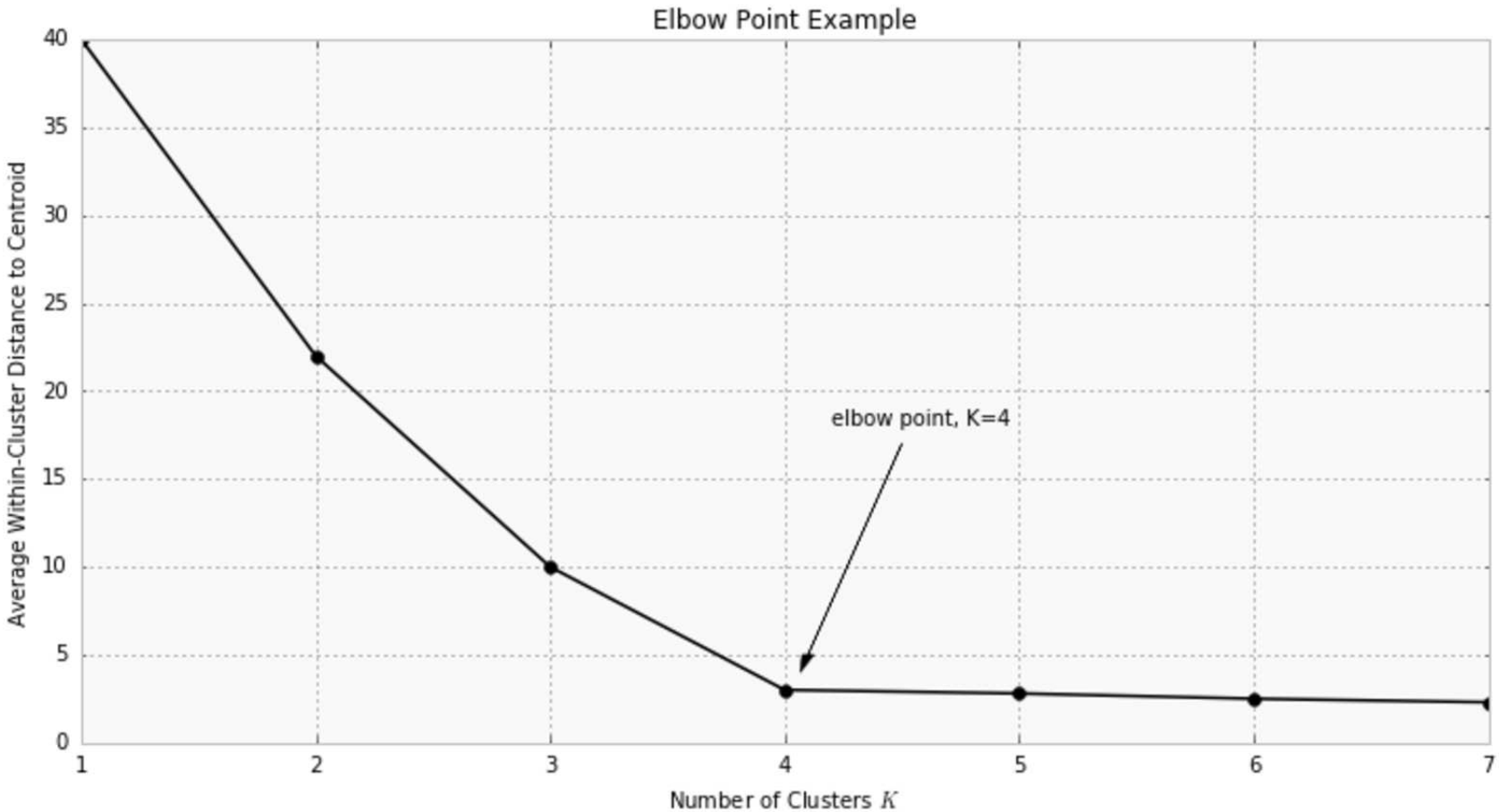
Istnieją podejścia do inicjalizacji algorytmu k-means bardziej „naukowe" niż próby losowe, np. algorytm inicjalizacji o nazwie `k-means`++, który znacząco poprawia wynik następującego po nim zastosowania k-means. K-means++ wykonuje k przebiegów na zbiorze danych, zatem nie skaluje się dobrze dla dużych zbiorów. Jego ulepszenie o nazwie k-means‖ daje równoważne wyniki i jest znacznie lepiej skalowalny.

1.D.Arthur, S.Vassilvitskii: "K-means++: the advantages of careful seeding", 2007
2.B.Bahmani, B.Moseley, A.Vattani, R.Kumar, S.Vassilvitskii: "Scalable K-means++"

Algorytm k-means — określenie liczby skupień

Wymagana przez algorytm liczba skupień K nie zawsze jest z góry znana, i czasami trzeba ją określić eksperymentalnie.

Metoda **punktu łokcia** (*elbow point*):



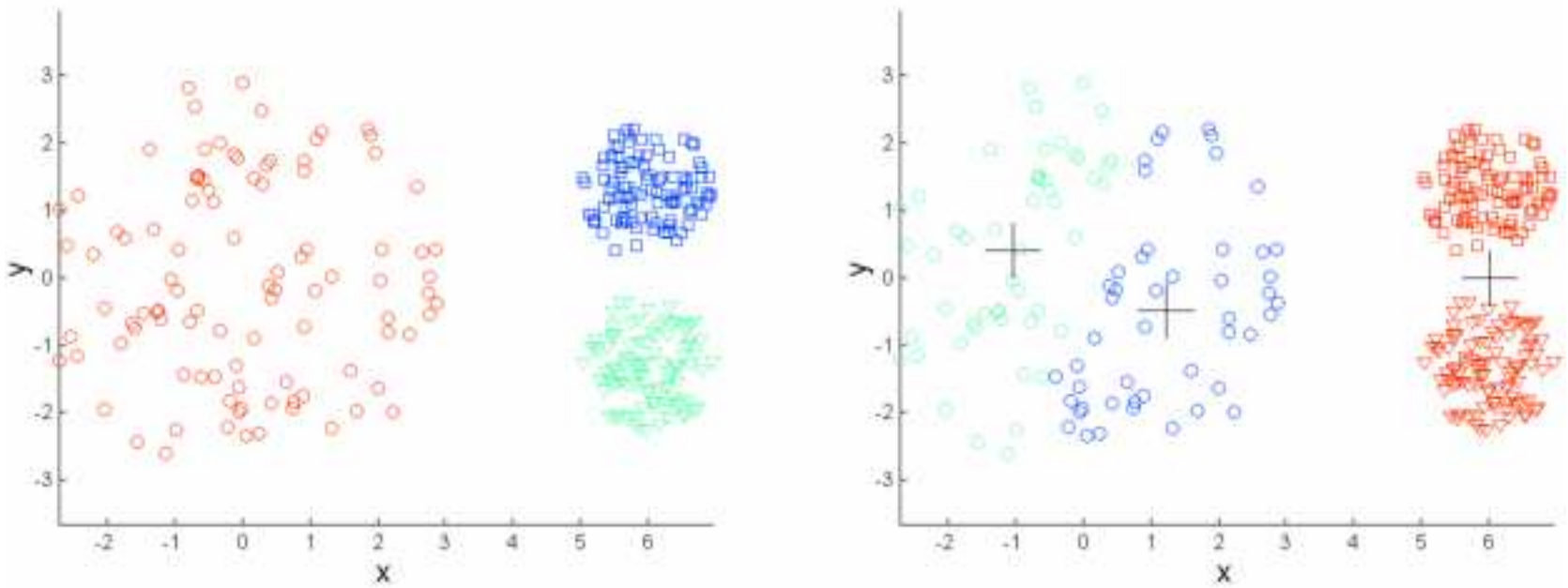
Metoda punktu łokcia nie zawsze się sprawdza. Często krzywa nie wykazuje charakterystycznego punktu załamania, i po prostu asymptotycznie maleje wraz ze wzrostem liczby skupień.

Niestety, w tym przypadku nie możemy próbować optymalizacji kryterium jakości, czyli ważonej sumy odległości wszystkich punktów od ich centroidów. Albowiem ta suma osiąga zero dla liczby skupień równej liczbie próbek $K = N$.

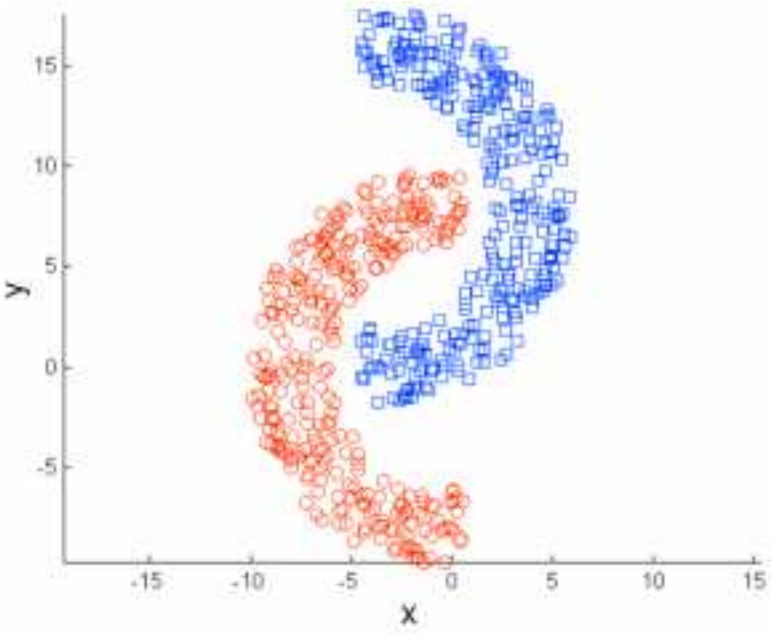
W takim przypadku można odwołać się do specyfiki problemu, z którego pochodzą próbki. Należy dokonać subiektywnej oceny, jaka liczba skupień będzie odpowiednia dla tej dziedziny problemowej.

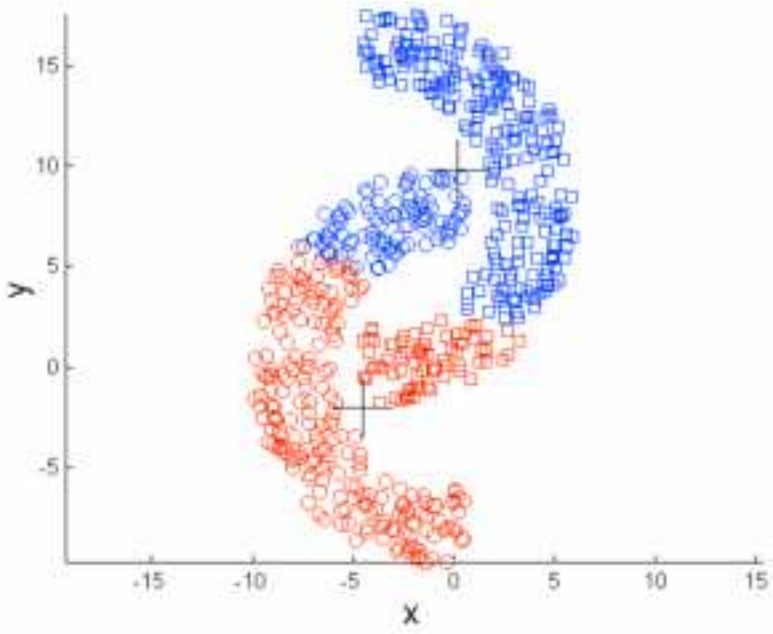
Algorytm k-means — problemy specjalne

Algorytm k-means dobrze działa w wielu przypadkach praktycznych, jednak są przypadki, gdzie definitywnie nie radzi sobie. Takimi przypadkami są skupienia różniące się wielkością, a także skupienia różniące się gęstością próbek w zbiorze treningowym.

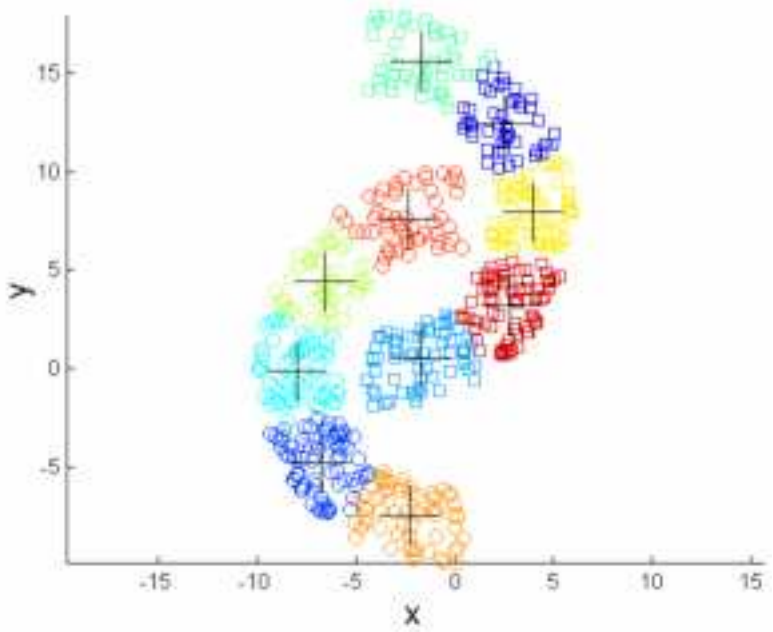


Algorytm k-means — problemy specjalne (2)





Problem z wklęsłymi skupieniami można rozwiązać pośrednio, zwiększając liczbę skupień.



Algorytm k-means — podsumowanie

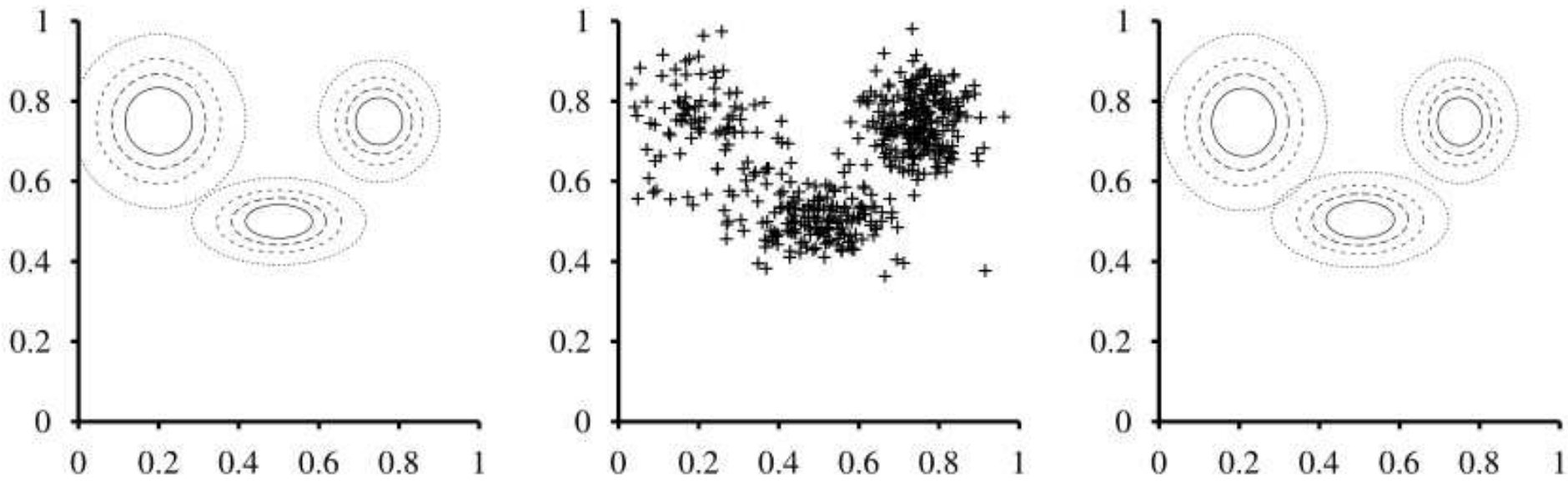
Algorytm k-means jest prostym i skutecznym algorytmem analizy skupień. Jego złożoność obliczeniowa wynosi $O(tKN)$ gdzie K, N są odpowiednio liczbą skupień i próbek, natomiast t jest liczbą iteracji algorytmu. Zwykle $K, t \ll N$.

Jednak posiada kilka istotnych problemów, które utrudniają, lub uniemożliwiają jego zastosowanie:

- wymaga wyznaczenia liczby skupień K,
- wrażliwy na inicjalizację centroidów, może zbiegać się do maksimów nielokalnych,
- ma zastosowanie do danych liczbowych (obliczanie średnich i odległości), problemy z danymi kategorycznymi,
- problemy ze skupieniami o niewypukłych kształtach,
- problemy ze skupieniami różniącymi się wielkościami,
- problemy ze skupieniami różniącymi się gęstościami.

Algorytm EM — Expectation Maximization

Można zastosować podejście podobne do algorytmu k-means na gruncie probabilistycznym. Zakładając, że punkty zbioru treningowego należą do K skupień z pewnym losowym rozkładem prawdopodobieństwa, naturalnie jest przyjąć, że te skupienia wynikają z normalnych rozkładów prawdopodobieństw, tzw. **mieszaniny rozkładów normalnych** albo **gausowskich** (*mixture of Gaussians*). Algorytm **EM** (*Expectation Maximization*) uczy się parametrów takiej mieszaniny rozkładów.



Rysunek po lewej przedstawia mieszaninę trzech symulowanych rozkładów normalnych. Środkowy rysunek przedstawia zbiór punktów wygenerowanych dla tego rozkładu. Rysunek po prawej przedstawia mieszaninę rozkładów wyuczoną przez algorytm EM.

Algorytm EM — Expectation Maximization (cd.)

Zakładając, że zmienna C oznacza składową mieszaniny z wartością $1, ..., K$, rozkład prawdopodobieństwa mieszaniny dany jest wzorem:

$$P(\mathbf{x}) = \sum_{i=1}^{K} P(C = i) P(\mathbf{x}|C = i)$$

gdzie **x** jest wektorem atrybutów próbki.

Parametrami rozkładu są: $w_i = P(C = i)$ (waga składowej i), μ_i (średnia składowej i), i Σ_i (kowariancja składowej i).

Idea algorytmu polega na tym, że początkowo zakładamy pewne wartości parametrów powyższego rozkładu. W każdym cyklu algorytmu, dla każdego punktu obliczane są prawdopodobieństwa, że należy on do poszczególnych składowych. Następnie, przeliczane są parametry wszystkich składowych na podstawie wszystkich punktów, z wagami będącymi prawdopodobieństwami przynależności danego punktu do danej składowej. Te dwa kroki powtarzane są aż do uzyskania zbieżności algorytmu, podobnie jak w metodzie k-means.

Algorytm EM — Expectation Maximization (cd.)

Algorytm EM:

Inicjalizacja: ustaw wartości początkowe parametrów wszystkich składowych

REPEAT {

Krok E: Oblicz prawdopodobieństwa $p_{ij} = P(C = i|\mathbf{x}_j)$, że próbka $\mathbf{x}_j$ należy do składowej i. Na mocy reguły Bayesa mamy: $p_{ij} = \alpha P(\mathbf{x}_j|C = i)P(C = i)$. Określamy $n_i = \Sigma_j p_{ij}$, czyli efektywną liczbę punktów aktualnie przypisanych do składowej i.

Krok M: Oblicz nowe średnie, kowariancje, i wagi składowych za pomocą następujących wzorów:

$$
\begin{aligned}
\mu_i &\leftarrow \sum_j p_{ij}\mathbf{x}_j/n_i \\
\Sigma_i &\leftarrow \sum_j p_{ij}(\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^\top/n_i \\
w_i &\leftarrow n_i/N
\end{aligned}
$$

}

Algorytm EM — Expectation Maximization (cd.)

Algorytm EM nie jest wolny od pewnych problemów. Możliwy jest przypadek, kiedy jedna ze składowych zredukuje się do pojedynczego punktu, z zerową wariancją i prawdopodobieństwem równym 1. Innym problemem jest nałożenie się dwóch składowych, które następnie współdzielą ten sam zbiór punktów.

Takie zjawiska prowadzą do zbiegnięcia się algorytmu w lokalnym maksimum. Jest to poważny problem, zwłaszcza w wielowymiarowych przestrzeniach. Rozwiązaniem może być reinicjalizacja składowej z nowymi parametrami, podobnie jak w przypadku algorytmu k-means.

Związek pomiędzy metodami k-means i EM

Algorytmy są w pewnym sensie podobne, realizują na przemian dwa kroki: (1) generowanie skupień, i (2) przenoszenie próbek pomiędzy skupieniami.

Jedną istotną różnicą jest, że w algorytmie k-means punkty przypisywane są skupieniom w sposób kategoryczny, podczas gdy EM przypisuje wszystkim punktom prawdopodobieństwa przynależności do poszczególnych dystrybucji.

Inną różnicą jest model gausowski, leżący u podstaw działania algorytmu EM. Algorytm k-means jest w stanie generować dystrybucje wynikowe, które nie są w żaden sposób podobne do rozkładów gausowskich. Z drugiej strony, wiele zjawisk naturalnych jest zgodnych z modelm gausowskim, zatem algorytm EM działa dla nich poprawnie.

Hierarchiczna analiza skupień

Analizę skupień można przeprowadzić, budując hierarchię skupień jednym z dwóch podstawowych sposobów:

- od dołu do góry, zaczynając od każdej próbki reprezentującej osobny klaster, następnie łącząc je w pary, i dalej budując większe klastry;
  takie podejście jest określane jako **aglomeracyjne** (*agglomerative clustering*),

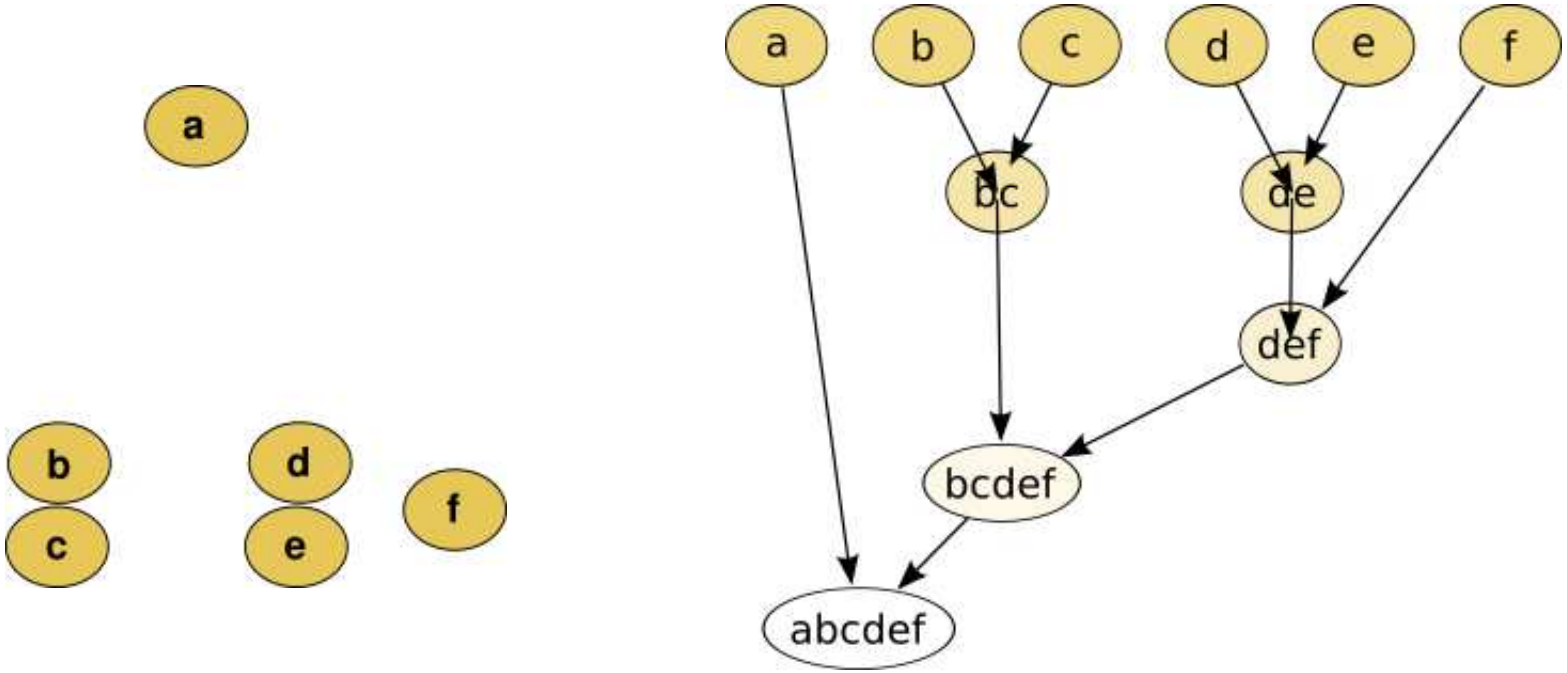
- od góry do dołu, zaczynając od jednego klastra reprezentującego wszystkie próbki, następnie dzieląc większe klastry na mniejsze;
  podejście nazywane metodą **deglomeracyjną** (*divisive clustering*).

Decyzje o tym, które klastry powinny zostać połączone, lub które podzielone, i dokładnie jak, zwykle są oparte na pomiarach odległości między próbkami <u>oraz klastrami</u>.

W pewnym sensie jest to podobne do algorytmu k-means, ale jest również odmienne przez konieczność mierzenia odległości między klastrami.

Aglomeracyjna analiza skupień — przykład

Bardziej powszechne są metody aglomeracyjne. Przykład:



Powstałe drzewo można odciąć na pewnej wysokości, aby uzyskać pożądaną liczbę klastrów.

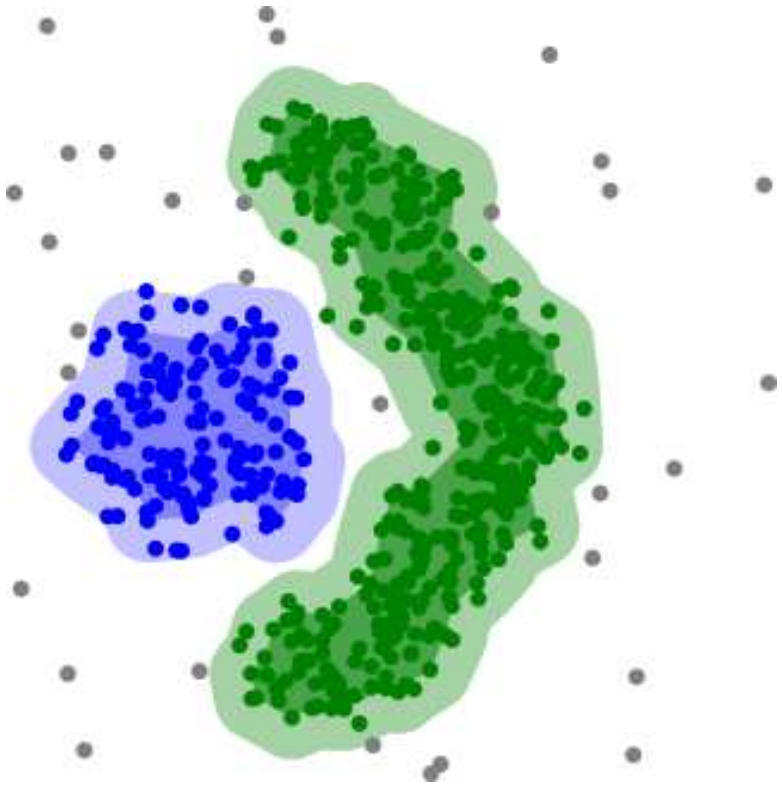
Metryki odległości między klastrami

Stosuje się szereg metod pomiaru odległości między klastrami:

- MIN — ma tendencję do tworzenia długich rozproszonych klastrów
- MAX — ma tendencję do tworzenia bardziej kompaktowych klastrów
- uśrednianie grupowe — uwzględnia średnią odległość między każdym punktem w jednym klastrze do każdego punktu w drugim
- odległość pomiędzy centroidami —
- metoda Warda — podobna do uśredniania grupowego, ale sumuje kwadraty odległości, w praktyce minimalizując całkowitą wariancję wewnątrz klastra

Inne podejścia do klasteryzacji

Istnieje szereg innych podejść do klasteryzacji, na przykład oparty na gęstości próbek algorytm DBSCAN. Jest on w stanie rozwiązać pewne problemy, z którymi kłopoty mają zarówno k-means jak i EM:



`https://en.wikipedia.org/wiki/DBSCAN`

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu: "A density-based algorithm for discovering clusters in large spatial databases with noise", in: Evangelos, Simoudis, Jiawei Han, Usama M. Fayyad (eds): Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226-231, 1996

Redukcja wymiaru

Istnieje szereg metod **redukcji wymiaru** pozwalających przekształcić reprezentację danych do innej przestrzeni, o mniejszym wymiarze. Jedną z przesłanek motywujących takie przekształcenie jest **klątwa wymiarowości**, będąca jednym z głównych problemów uczenia maszynowego.

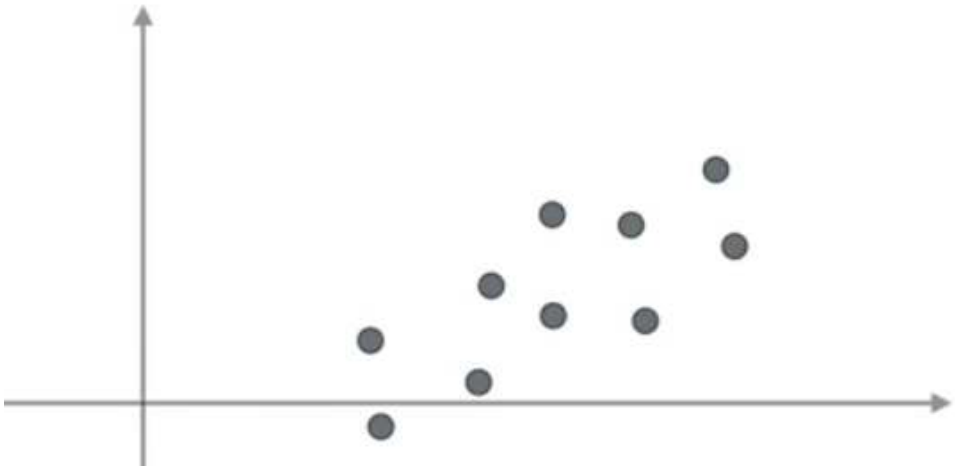
Mówiąc obrazowo, wiele algorytmów uczenia maszynowego, które dobrze sprawują się dla danych w niskowymiarowych przestrzeniach, przestają działać zadowalająco gdy wymiar przestrzeni jest duży.

Z innego punktu widzenia, dane są typowo reprezentowane przez szereg parametrów, z których niektóre mogą nie mieć istotnego wpływu na zdolność klasyfikacji lub grupowania tych danych. Takie nadmiarowe parametry nie tylko nie pomagają w automatycznym wykrywaniu wzorców istniejących w danych, ale istotnie w nim przeszkadzają, ponieważ wprowadzają pozorne zależności skutecznie utrudniające pracę algorytmów.

Zatem opłaca się, przed przystąpieniem do eksperymentu maszynowego uczenia, wykonać analizę i redukcję wymiarowości. Jedną z niezwykle skutecznych jej metod jest algorytm analizy składowych głównych PCA (*Principal Component Analysis*).

Analiza składowych głównych (PCA) — przykład

Rozważmy pewien zbiór punktów:



Przesuwamy jego środek geometryczny do początku układu współrzędnych:

Obliczamy macierz kowariancji dla przedstawionego zbioru punktów:

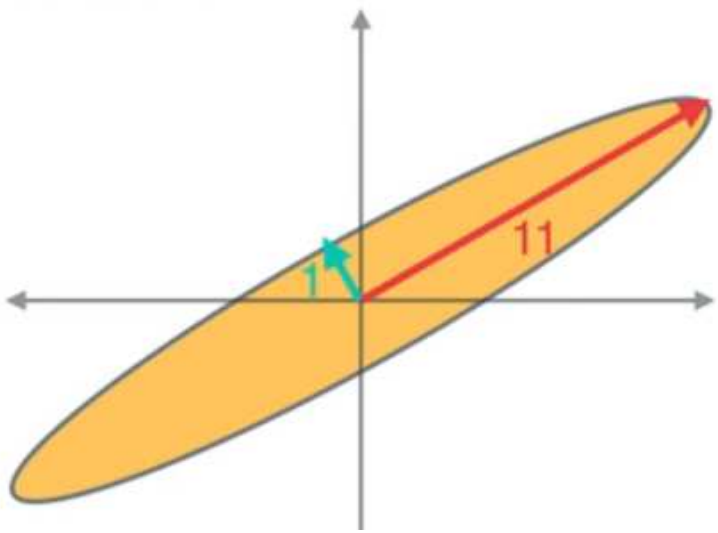
$$\Sigma = \begin{pmatrix} var(x) & cov(x,y) \\ cov(x,y) & var(y) \end{pmatrix} = \begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

Ta macierz kowariancji generuje pewne przekształcenie liniowe:

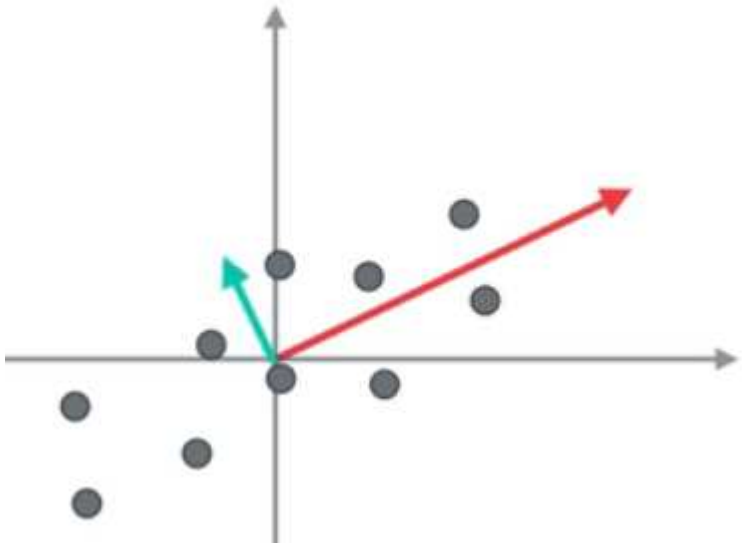
$$(x, y) \longrightarrow (9x + 4y, 4x + 3y)$$

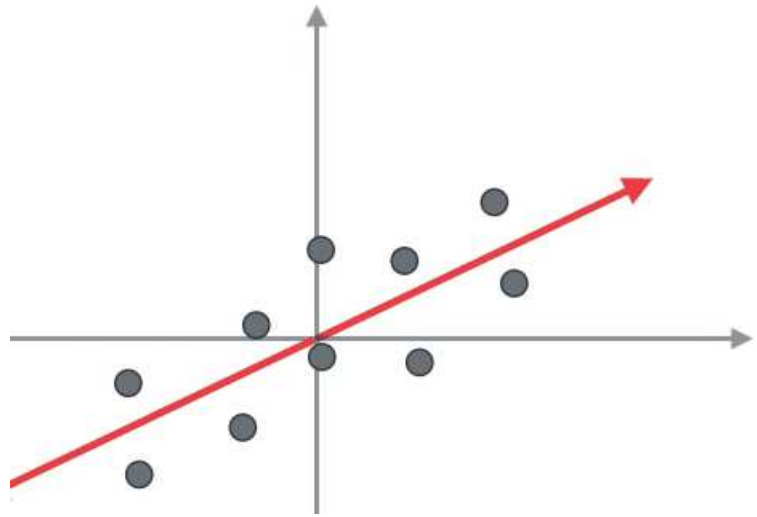
To przekształcenie przenosi oryginalne punkty do układu współrzędnych, którego osiami są wektory własne macierzy kowariancji, a rozciągnięcie liniowe generują wartości własne:

Wektory własne	$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 2 \end{bmatrix}$
Wartości własne	11	1

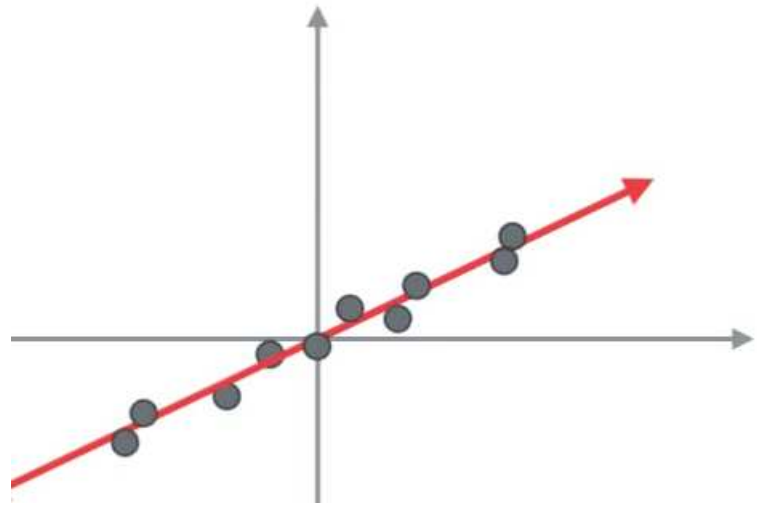


Oryginalne punkty są dokładnie reprezentowane w nowym układzie współrzędnych, którego osie nazywane są **składowymi głównymi**. Jednak dla celów redukcji wymiaru dążymy do wyboru tylko jednej współrzędnej. Musi nią być składowa główna o największej wartości własnej.





Redukcja wymiaru oznacza reprezentację punktów przez ich zrzutowanie do przestrzeni o niższej wymiarowości, czyli w tym wypadku na wybraną oś współrzędnych.
Jest to zatem reprezentacja przybliżona.



Algorytm PCA

Algorytm PCA znajduje M-wymiarowe przybliżenie zbioru danych $\{\mathbf{x}^n : n = 1, ..., N\}$ o wymiarze oryginalnym $\dim(\mathbf{x}^n) = D$ $(M < D)$:

1. Wyznacz wektor średnich $\mathbf{m}$ zbioru próbek o rozmiarze $D \times 1$ i macierz kowariancji $\mathbf{S}$ o rozmiarze $D \times D$:

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}^n, \quad \mathbf{S} = \frac{1}{N-1} \sum_{n=1}^{N} (\mathbf{x}^n - \mathbf{m})(\mathbf{x}^n - \mathbf{m})^T.$$

2. Wyznacz wektory własne $\mathbf{e}^1, ..., \mathbf{e}^D$ macierzy kowariancji $\mathbf{S}$ posortowane według malejących wartości własnych wektorów własnych. Utwórz macierz $\mathbf{E} = [\mathbf{e}^1, ...\mathbf{e}^M]$.

3. Niskowymiarowa reprezentacja $\mathbf{y}^n$, oraz przybliżona rekonstrukcja $\mathbf{x}'^n$ każdej z próbek n są dane przez:

$$\mathbf{y}^n = \mathbf{E}^T(\mathbf{x}^n - \mathbf{m}), \quad \mathbf{x}^n \approx \mathbf{x}'^n = \mathbf{m} + \mathbf{E}\mathbf{y}^n.$$

4. Całkowity błąd kwadratowy przybliżenia dla zbioru treningowego wynosi:

$$\sum_{n=1}^{N} (\mathbf{x}^n - \mathbf{x}'^n)^2 = (N-1) \sum_{j=M+1}^{D} \lambda_j$$

gdzie $\lambda_{M+1}, ..., \lambda_N$ są pominiętymi wartościami własnymi.

Uwaga: algorytm PCA jest wrażliwy na wielkość wartości parametrów. Jeśli jeden z parametrów będzie przyjmował wartości znacznie większe niż inny, to ten pierwszy parametr zostanie wybrany jako największa główna składowa, a drugi jako kolejna. Z tej przyczyny przed wykonaniem algorytmu PCA wymagane jest uniwersalne skalowanie wszystkich parametrów.

Analiza koszyka rynkowego

Istnieje szereg metod drążenia danych/uczenia maszynowego dotyczących zakupów. Zwróćmy uwagę na dwa rodzaje rekomendacji wyświetlanych przez portale sprzedażowe takie jak Amazon:

- Często kupowane razem: ...
- Klienci, którzy kupili ten produkt kupili również ...

Pierwsza grupa metod jest określana jako **analiza koszyka rynkowego** (*Market Basket Analysis*). Metody MBA koncentrują się na **odkrywaniu reguł asocjacyjnych** (*association rule mining*) charakteryzujących typowo wykonywane zakupy, czyli znajdowaniu związków pomiędzy różnymi artykułami kupowanymi w różnych transakcjach. Takie związki reprezentowane są na przykład w postaci reguł JEŻELI-TO. Jednym z najbardziej znanych algorytmów tej grupy jest Apriori.

Nieco inne podejście do drążenia danych historii transakcji zakupowych wyrażają **systemy rekomendacji** (*Recommender Systems*). Zmierzają one do określania preferencji indywidualnych konsumentów. Najbardziej znane algorytmy należą do grupy filtrowania kolaboracyjnego (*collaborative filtering*), i będą omówione później.

Analiza koszyka rynkowego — model

Rozważamy historię transakcji zakupowych jako zbiór transakcji $T = \{t_1, t_2, ..., t_n\}$, gdzie pojedyncza transakcja jest podzbiorem $t \subseteq I$ zbioru wszystkich dostępnych elementów $I = \{i_i, i_2, ..., i_m\}$.

Przykład:

Transaction 1	🍎 🍺 🍚 🍗
Transaction 2	🍎 🍺 🍚
Transaction 3	🍎 🍺
Transaction 4	🍎 🍐
Transaction 5	🍼 🍺 🍚 🍗
Transaction 6	🍼 🍺 🍚
Transaction 7	🍼 🍺
Transaction 8	🍼 🍐

Analiza koszyka rynkowego — reguły asocjacyjne

Chcemy odkrywać związki pomiędzy elementami transakcji zakupowych w postaci **reguł asocjacyjnych** postaci, przyjmując, że A i B są rozłącznymi ($A \cap B = \emptyset$) zbiorami elementów zakupowych ($A, B \subseteq I$):

$$A \Rightarrow B$$

Takich związków istnieje zapewne wiele w całej historii kupowania, lecz jedne mogą być silniejsze niż inne. Wprowadzamy następujące miary wiarygodności reguły $A \Rightarrow B$ na podstawie zbioru transakcji T.

Wsparciem (*Support*) reguły $A \Rightarrow B$ w zbiorze transakcji T nazywamy częstotliwość transakcji, które zawierają wszystkie elementy z A i B:

$$\text{Support}(A \Rightarrow B) = P(A \cup B)$$
$$\text{Support}(X) = P(X)$$

Zaufaniem (*Confidence*) reguły $A \Rightarrow B$ w zbiorze transakcji T nazywamy częstotliwość takich transakcji zawierających B, które również zawierają A:

$$\text{Confidence}(A \Rightarrow B) = P(B|A)$$

Inaczej mówiąc, zaufanie reguły mówi jak często była ona prawdziwa w zbiorze transakcji.

Dodatkowo definiujemy **Przyrost** (*Lift*) jako ułamek:

$$\text{Lift}(A \Rightarrow B) = \frac{P(A \cup B)}{P(A) \cdot P(B)}$$

Przyrost wskazuje ogólne znaczenie reguły w następującym sensie:

$$\begin{aligned}
\text{Lift}(A \Rightarrow B) &> 1 \quad \text{oznacza, że } A, B \text{ są dodatnio skorelowane} \\
\text{Lift}(A \Rightarrow B) &< 1 \quad \text{oznacza, że } A, B \text{ są ujemnie skorelowane} \\
\text{Lift}(A \Rightarrow B) &= 1 \quad \text{oznacza, że } A, B \text{ są niezależne}
\end{aligned}$$

Analiza koszyka rynkowego — zbiory częste

Poszukiwanie reguł asocjacyjnych wymaga zdefiniowania minimalnego progu wsparcia oraz minimalnego progu zaufania. Wyznaczenie wszystkich reguł asocjacyjnych może być zrealizowane następująco:

- Znajdź wszystkie zbiory **częste** kupowanych elementów.

  Zbiór elementów jest częsty jeśli przekracza zadany próg wsparcia *minsupport*.

- W obrębie każdego zbioru częstego wyznacz właściwe reguły asocjacyjne.

  Reguła asocjacyjna jest musi przekraczać minimalny próg zaufania *minconf* aby zostać wybrana.

Znajdowanie zbiorów częstych wymaga sprawdzania wszystkich podzbiorów zbioru elementów I, których jest $2^{|I|} - 1$ (z pominięciem podzbioru pustego). Zwykły pełny przegląd tych podzbiorów byłby zbyt nieefektywny.

Algorytm Apriori — generowanie zbiorów częstych

Własność *apriori*: wszystkie podzbiory zbioru częstego są również zbiorami częstymi.
I na odwrót: nadzbiór zbioru, który nie jest częsty, również nie może być częsty.

Tę własność można wykorzystać dla efektywnej generacji zbiorów częstych według idei: znajdź wszystkie częste zbiory jednoelementowe, następnie na ich bazie wygeneruj wszystkie zbiory dwuelementowe i odfiltruj z nich tylko zbiory częste, następnie tak samo utwórz trzyelementowe zbiory częste, itd.

Algorytm:

1. Utwórz kolekcję jednoelementowych zbiorów częstych F_1.
2. Dla k=2..$|I|$ dopóki $F_{k-1} \neq \emptyset$ wykonuj:
   (a) na podstawie F_{k-1} utwórz kolekcję C_k takich k-elementowych kandydatów c_k na zbiory częste, które są połączeniem dwóch $(k-1)$-elementowych zbiorów częstych f_{k-1}^i i f_{k-1}^j o $(k-2)$ elementach wspólnych
   (b) każdy $(k-1)$-elementowy podzbiór c_k musi być zbiorem częstym w F_{k-1}
   (c) utwórz kolekcję k-elementowych zbiorów częstych F_k odfiltrowując z C_k zbiory nieczęste.

Warunek (b) powoduje wczesne filtrowanie zbiorów nieczęstych na podstawie własności *apriori*.

Algorytm Apriori — generowanie reguł asocjacyjnych

Algorytm generowania reguł:

1. Dla każdego zbioru częstego X

   (a) Dla każdego właściwego i niepustego podzbioru A zbioru X:
   niech $B = X \setminus A$
   $A \Rightarrow B$ będzie regułą asocjacyjną jeśli: Confidence$(A \Rightarrow B) \geq$ *minconf*

Algorytm Apriori — przykład

Rozważmy bazę danych transakcji zakupowych:

ID	Elementy
T1	I1 I3 I4
T2	I2 I3 I5
T3	I1 I2 I3 I5
T4	I2 I5

Do wykonania algorytmu Apriori przyjmiemy parametry *minsupport*=0.5; *minconf*=0.75.

Pierwsza iteracja:

ID	Elementy
T1	I1 I3 I4
T2	I2 I3 I5
T3	I1 I2 I3 I5
T4	I2 I5

$\Rightarrow$ C1:

zbiór	Support
{I1}	2/4=0.5
{I2}	3/4=0.75
{I3}	3/4=0.75
{I4}	1/4=0.25
{I5}	3/4=0.75

$\Rightarrow$ F1:

zbiór	Support
{I1}	0.5
{I2}	0.75
{I3}	0.75
{I5}	0.75

Druga iteracja:

ID	Elementy
T1	I1 I3 I4
T2	I2 I3 I5
T3	I1 I2 I3 I5
T4	I2 I5

$\Rightarrow$ C2:

zbiór	Support
{I1,I2}	1/4=0.25
{I1,I3}	2/4=0.5
{I1,I5}	1/4=0.25
{I2,I3}	2/4=0.5
{I2,I5}	3/4=0.75
{I3,I5}	2/4=0.5

$\Rightarrow$ F2:

zbiór	Support
{I1,I3}	2/4=0.5
{I2,I3}	2/4=0.5
{I2,I5}	3/4=0.75
{I3,I5}	2/4=0.5

Trzecia iteracja:

ID	Elementy
T1	I1 I3 I4
T2	I2 I3 I5
T3	I1 I2 I3 I5
T4	I2 I5

$\Rightarrow$ C3:

zbiór	Support
{I1,I2,I3}	1/4=0.25
{I1,I3,I5}	1/4=0.25
{I2,I3,I5}	2/4=0.5

$\Rightarrow$ F3:

zbiór	Support
{I2,I3,I5}	2/4=0.5

Należy zauważyć, że zbiory {I1,I2,I3} i {I1,I3,I5} nawet przed obliczeniem ich wsparcia zostały odrzucone przez algorytm Apriori, ponieważ zawierają podzbiory nie należące do F_2 (krok 2b algorytmu).

Algorytm zatrzymał się na czwartej iteracji, ponieważ zbiór C_4 jest pusty. Do generacji reguł przechodzą zbiory częste z F_2 i F_3 ponieważ jednoelementowe zbiory z F_1 nie generują żadnych reguł.

Generacja reguł:

ID	Elementy
T1	I1 I3 I4
T2	I2 I3 I5
T3	I1 I2 I3 I5
T4	I2 I5

$\Rightarrow$

ID	zbiór	reguła			Confidence
R1	{I1,I3}	{I1}	$\Rightarrow$	{I3}	2/2=1.0
R2	{I1,I3}	{I3}	$\Rightarrow$	{I1}	2/3=0.66
R3	{I2,I3}	{I2}	$\Rightarrow$	{I3}	2/3=0.66
R4	{I2,I3}	{I3}	$\Rightarrow$	{I2}	2/3=0.66
R5	{I2,I5}	{I2}	$\Rightarrow$	{I5}	3/3=1.0
R6	{I2,I5}	{I5}	$\Rightarrow$	{I2}	3/3=1.0
R7	{I3,I5}	{I3}	$\Rightarrow$	{I5}	2/3=0.66
R8	{I3,I5}	{I5}	$\Rightarrow$	{I3}	2/3=0.66
R9	{I2,I3,I5}	{I2,I3}	$\Rightarrow$	{I5}	2/2=1.0
R10	{I2,I3,I5}	{I2,I5}	$\Rightarrow$	{I3}	2/3=0.66
R11	{I2,I3,I5}	{I3,I5}	$\Rightarrow$	{I2}	2/2=1.0
R12	{I2,I3,I5}	{I2}	$\Rightarrow$	{I3,I5}	2/3=0.66
R13	{I2,I3,I5}	{I3}	$\Rightarrow$	{I2,I5}	2/3=0.66
R14	{I2,I3,I5}	{I5}	$\Rightarrow$	{I2,I3}	2/3=0.66

Ostatecznie dla zadanych parametrów algorytm Apriori wygenerował pięć reguł asocjacyjnych.

Systemy rekomendacji

Przyjrzymy się teraz innemu podejściu związanemu z analizą danych zakupowych, dotyczącemu systemów rekomendacji. Chcąc zarekomendować jakiś towar pewnemu konsumentowi chcemy przewidzieć jego możliwą ocenę tego towaru. Dla tej procedury wyznaczania możliwej oceny konsumenta jest często używane określenie **filtrowania**, które w statystyce jest stosowane w odniesieniu do aproksymowania wielkości bieżących (a nie przyszłych albo przeszłych).

Aby wyznaczyć spodziewaną ocenę towaru A przez konsumenta X możemy zastosować:

filtrowanie kolaboracyjne, oparte na podobieństwie konsumentów (*collaborative filtering*)
jeśli konsument X jest podobny do Y, i Y kupił A; poleć A konsumentowi X

filtrowanie oparte na podobieństwie treści (*content-based filtering*)
jeśli konsument X kupił A, i A jest podobne do B; poleć B konsumentowi X

Są również możliwe podejścia hybrydowe w różny sposób łączące oba powyższe.

Systemy rekomendacji uczą się wyznaczania konkretnych wartości na podstawie zapamiętanych danych, a więc formalnie należą do metod uczenia nadzorowanego. Omawiamy je w grupie algorytmów poświęconej uczeniu nienadzorowanemu ze względu na podobieństwo zastosowań.

Filtrowanie kolaboracyjne i oparte na podobieństwie treści

Zasadniczo zadanie wyznaczenia nieznanego parametru A próbki X gdy parametr ten jest znany w serii uczącej, jest typowym zadaniem klasyfikacji (gdy wartości parametru A mają charakter dyskretny) lub regresji (gdy te wartości wybierane są z ciągłego zakresu liczbowego). Zatem można do nich stosować wcześniej poznane algorytmy. To podejście do zagadnienia rekomendacji określane jest jako *model based*.

Możliwe jest jednak jeszcze inne podejście, polegające na wyborze z posiadanego zbioru danych próbek dotyczących konsumentów „podobnych" w jakimś sensie do konsumenta X, i oparcie rekomendacji na parametrach tylko tych konsumentów. (Albo, w przypadku filtrowania opartego na podobieństwie treści, na wyborze tylko towarów „podobnych" do przedmiotowego A.) To podejście określane jest jako *memory-based* i zostanie tu pokrótce przedstawione.

Jedną z zalet tego ostatniego podejścia jest, że otrzymany wynik jest łatwo przedstawić w kontekście i uzasadnić.

Filtrowanie kolaboracyjne — obliczanie podobieństwa konsumentów

Zakładając, że dostępne dane zawierają oceny wielu towarów wyznaczone przez konsumentów, możemy obliczać „podobieństwo" konsumentów X i Y na podstawie ich ocen $r_{x,1}, r_{x,1}, ..., r_{y,1}, r_{y,2}...$ pewnej grupy towarów I_{xy} ocenionych przez obu konsumentów, korzystając z jednego z poniższych modeli podobieństwa.

Model podobieństwa Pearsona:

$$\text{simil}(x, y) = \frac{\sum\limits_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum\limits_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2} \sqrt{\sum\limits_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Podobieństwo obliczone jako cosinus kąta między wektorami atrybutów:

$$\text{simil}(x, y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum\limits_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum\limits_{i \in I_{xy}} r_{x,i}^2} \sqrt{\sum\limits_{i \in I_{xy}} r_{y,i}^2}}$$

Filtrowanie kolaboracyjne — wyznaczanie oceny konsumenta

Mając podobieństwa konsumentów obliczone ze względu na oceniane towary, możemy wybrać pewien zbiór U N konsumentów najbardziej zbliżonych do danego konsumenta u, i wyznaczyć jego ocenę towaru i jako:

$$
\begin{aligned}
r_{u,i} &= \frac{1}{N} \sum_{u' \in U} r_{u',i} \qquad \text{lub:} \\
r_{u,i} &= \frac{\sum_{u' \in U} \mathsf{simil}(u, u') r_{u',i}}{\sum_{u' \in U} |\mathsf{simil}(u, u')|} \qquad \text{lub:} \\
r_{u,i} &= \bar{r}_u + \frac{\sum_{u' \in U} \mathsf{simil}(u, u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in U} |\mathsf{simil}(u, u')|}
\end{aligned}
$$

Pierwsza wersja jest zwykłą średnią ocen „podobnych” konsumentów, druga jest średnią ocen ważoną wzajemnym podobieństwem konsumentów, a trzecia dodatkowo uniezależnia się od bezwzględnej wartości ocen poszczególnych konsumentów, uwzględniając tylko różnice od ich średnich ocen, i przenosząc uzyskaną względną zwyżkę lub zniżkę oceny towaru i na średnią ocenę konsumenta u.

Filtrowanie oparte na podobieństwie treści

W przypadku filtrowania opartego na podobieństwie treści posługujemy się tymi samymi danymi ocen wielu towarów przez wielu konsumentów, ale tym razem obliczamy macierz podobieństw towarów, stosując te same wzory podobieństwa.

Następnie wyznaczamy zbiór N towarów najbardziej podobnych do towaru, którego ocenę dla konsumenta u chcemy wyznaczyć, i określamy ocenę jedną z metod uśredniania podobnie jak w przypadku uśredniania ocen konsumentów.

Faktoryzacja macierzy ocen

Całkowicie odmiennym podejściem do filtrowania kolaboracyjnego jest idea faktoryzacji oryginalnej macierzy ocen $R \in \mathcal{R}^{\text{users}\times\text{items}}$ na dwie macierze, z których pierwsza H ma rzędy odpowiadające wszystkim konsumentom, a druga W ma kolumny odpowiadające wszystkim towarom. Kolumny pierwszej i rzędy drugiej macierzy są związane z nowo utworzonymi **zmiennymi ukrytymi** (*latent factors*) w taki sposób, że iloczyn wektorowy macierzy H i W możliwie wiernie przybliża macierz ocen, i zatem uzupełnia oceny, których brakuje i które należy wyznaczyć:

$$\tilde{R} = H \times W$$

gdzie:
$\tilde{R} \in \mathcal{R}^{\text{users}\times\text{items}}$ jest macierzą predykcji ocen,
$H \in \mathcal{R}^{\text{users}\times\text{latent factors}}$ jest macierzą zmiennych ukrytych dla danego użytkownika,
$W \in \mathcal{R}^{\text{latent factors}\times\text{items}}$ jest macierzą zmiennych ukrytych towarów (transponowaną).

Ta idea **faktoryzacji macierzy** (*matrix factorization*) ma na celu wyłonienie pewnej liczby zmiennych stanu (zmiennych ukrytych) pozwalających wyrażać oceny za pomocą macierzy o mniejszej wymiarowości. Stanowi to pewien odpowiednik metody analizy składowych głównych (*Pricipal Component Analysis*, PCA).

Metody faktoryzacji macierzy ocen

Zmienne ukryte wyznacza się zwykle metodami uczenia maszynowego, na przykład sieciami neuronowymi. Zaproponowanych został szereg metod faktoryzacji macierzy ocen. Pierwsza wersja faktoryzacji macierzy ocen, zwana **Funk MF** powstała w odpowiedzi na konkurs Netflixa. W tej wersji predykcja $\bar{r}_{ui}$ oceny użytkownika u towaru i jest obliczana według wzoru:

$$\bar{r}_{ui} = \sum_{f=0}^{n \text{ factors}} H[u, f]W[f, i]$$

Siła ekspresji przestrzeni zmiennych ukrytych jest związana z jej wymiarem. Dla jednej zmiennej ukrytej ta reprezentacja sprowadza się do wyboru najczęściej rekomendowanego towaru, niezależnie od konsumenta. Zwiększanie liczby zmiennych pozwala na wprowadzenie personalizacji, i jakość rekomendacji wynikających z wartości ocen się zwiększa, a powyżej pewnej liczby zmiennych zaczyna mieć tendencję do przeuczenia. Dla uniknięcia przeuczenia stosowana jest regularyzacja polegająca na dodaniu składnika regularyzacyjnego do funkcji oceny.

Metoda Funk MF minimalizuje funkcję oceny:

$$\text{argmin}_{H,W} ||R - \tilde{R}||_F + \alpha||H|| + \beta||W||$$

gdzie $||.||_F$ jest normą Frobeniusa będącą pewnym uogólnieniem normy Euklidesowej.

Uczenie się bayesowskich sieci przekonań

Przypomnijmy sobie sieci bayesowskie i ćwiczenie z budową sieci dla danego zbioru danych. Widzieliśmy, że niektóre programy do budowy i obliczeń na takich sieciach miały funkcję automatycznej budowy sieci. Jak mógłby działać taki algorytm?

Przypomnijmy sobie podstawowe zasady budowy sieci bayesowskich:

- sieć bayesowska jest acyklicznym grafem skierowanym (DAG — *Directed Acyclic Graph*), lub zbiorem acyklicznych grafów skierowanych, tzn. dopuszczamy rozdzielenie zbioru zmiennych losowych na podzbiory (partycje), które są od siebie całkowicie niezależne,

- zmienne losowe, które są niezależne, nie są bezpośrednio połączone,

- zmienne losowe, z których jedna zależy bezpośrednio od drugiej, połączone są łukiem skierowanym, wskazującym zależność,

- ogólnie korzystne są sieci o małej liczbie łuków — zwłaszcza małej liczbie rodziców każdego węzła — ponieważ wielu rodziców oznacza rozbudowane rozkłady prawdopodobieństw warunkowych, które są złożone obliczeniowo, i dla których estymacji potrzeba wielkich ilości danych.

Rozważymy to zagadnienie po kolei na dwóch poziomach.

Najpierw założymy, że struktura sieci jest znana, natomiast zbiór danych uczących jest niekompletny, i brak wartości pewnych zmiennych. Chcemy nauczyć się kompletnych parametrów sieci, tzn. wszystkich rozkładów prawdopodobieństw warunkowych.

Z kolei, założymy, że dane uczące są kompletne, ale struktura sieci nie jest znana, i chcemy wygenerować najbardziej poprawną strukturę sieci dla danego zbioru danych.

Wyznaczanie parametrów sieci przekonań

Zakładamy, że znana jest struktura sieci, i zbiór danych uczących, w których jednak brakuje pewnych wartości. Zauważmy, że sieć bayesowska służy do określania najbardziej prawdopodobnych wartości pewnych parametrów, gdy dane są (dowolne) inne parametry. Możemy to wykorzystać do uzupełnienia brakujących parametrów.

Zastosujemy pewną wersję algorytmu EM (Expectation Maximization), który działa przez powtarzanie następujących dwóch kroków, do osiągnięcia zbieżności:

Inicjalizacja: oblicz wstępne wartości parametrów sieci, ignorując brakujące dane.

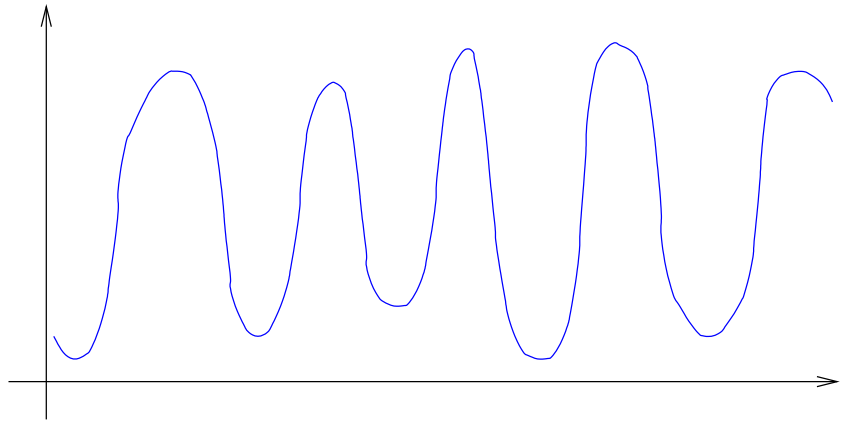
Krok E (Expectation): dla każdej próbki danych oblicz wartości oczekiwane parametrów, których wartości brakuje.

Krok M (Maximization): oblicz nowe parametry sieci (warunkowe rozkłady prawdopodobieństw wszystkich zmiennych).

Algorytm EM jest dość ogólnym schematem, znajdującym zastosowanie do szeregu różnych problemów. Działa obliczając na przemian: lepsze przybliżenie poszukiwanych wartości (krok E), i lepszy, zoptymalizowany model (krok M).
Pytanie: jaka jest wada tego podejścia?
Odpowiedź: jest to algorytm gradientowy, podatny na zbiegnięcie się do lokalnego, zamiast globalnego, maksimum funkcji oceny.

Wyznaczanie parametrów sieci przekonań — maksima lokalne

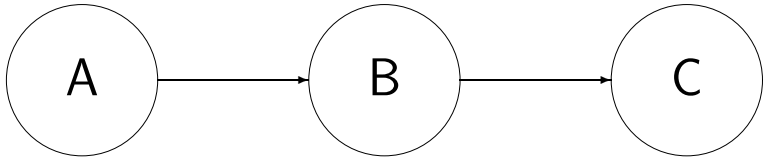
Algorytm EM wykonuje przeszukiwanie gradientowe, i zbiega się do lokalnego maksimum funkcji jakości. W ogólnym przypadku możemy nigdy nie mieć pewności, że znalezione maksimum jest globalne. To jednak może nie być do końca zła wiadomość.



Krzywa kryterium jakości parametrów sieci może mieć wiele lokalnym maksimów, ale w sumie niekoniecznie jest istotne, aby jako rozwiązanie znaleźć jego globalne maksimum.

W uczeniu maszynowym celem nie jest maksymalne dopasowanie modelu do zbioru uczącego, lecz uogólnianie. Nie ma wcale pewności, że globalne maksimum będzie oferować najlepsze uogólnianie. Dlatego jako wynik wyznaczania parametrów sieci bayesowskiej algorytmem EM można przyjąć dowolne optimum lokalne, jeśli mamy przekonanie, że nie jest ono dużo gorsze od globalnego.

Wyznaczanie parametrów sieci bayesowskiej — przykład



Próbki:	0	1	1
	1	0	0
	1	1	1
	1	?	0

Inicjalizacja: $P(B|A) = 0.5$ $P(C|B) = 1.0$
$P(A) = 0.75$ $P(B|\neg A) = 1.0$ $P(C|\neg B) = 0.0$

Krok E: $P(? = 1) = P(B|A, \neg C) = \frac{P(A,B,\neg C)}{P(A,\neg C)} = ... = 0$

$$P(A, B, \neg C) = P(A) * P(B|A) * P(\neg C|A, B) = P(A) * P(B(|A) * P(\neg C|B)$$
$$= 0.75 * 0.5 * 0.0$$

$$P(A, \neg C) = P(A, B, \neg C) + P(A, \neg B, \neg C) = ...$$

Krok M: $P(B|A) = 0.33$ $P(C|B) = 1.0$
$P(A) = 0.75$ $P(B|\neg A) = 1.0$ $P(C|\neg B) = 0.0$

Krok E: $P(? = 1) = 0$ **(algorytm zbiegł się)**

Uczenie się struktury

Generowanie struktury sieci jest zadaniem trudnym obliczeniowo. Dla D zmiennych istnieje eksponencjalnie wiele struktur grafowych łączących odpowiednie węzły. Skuteczny algorytm poszukiwania optymalnej struktury grafu musi stosować heurystyki i/lub dodatkowe założenia ograniczające.

Istnieje szereg takich algorytmów. Przeanalizujemy tu dwa z nich:

- algorytm PC tworzący szkielet grafu nieskierowanego, począwszy od grafu w pełni połączonego, przez usuwanie krawędzi łączących zmienne, które mogą być uznane za niezależne, albo są zależne, ale są warunkowo niezależne pod warunkiem trzeciej zmiennej, z którą obie są połączone,

- algorytm określający kierunki łuków grafu na podstawie informacji o ich warunkowej niezależności.

Algorytm PC generowania szkieletu grafu

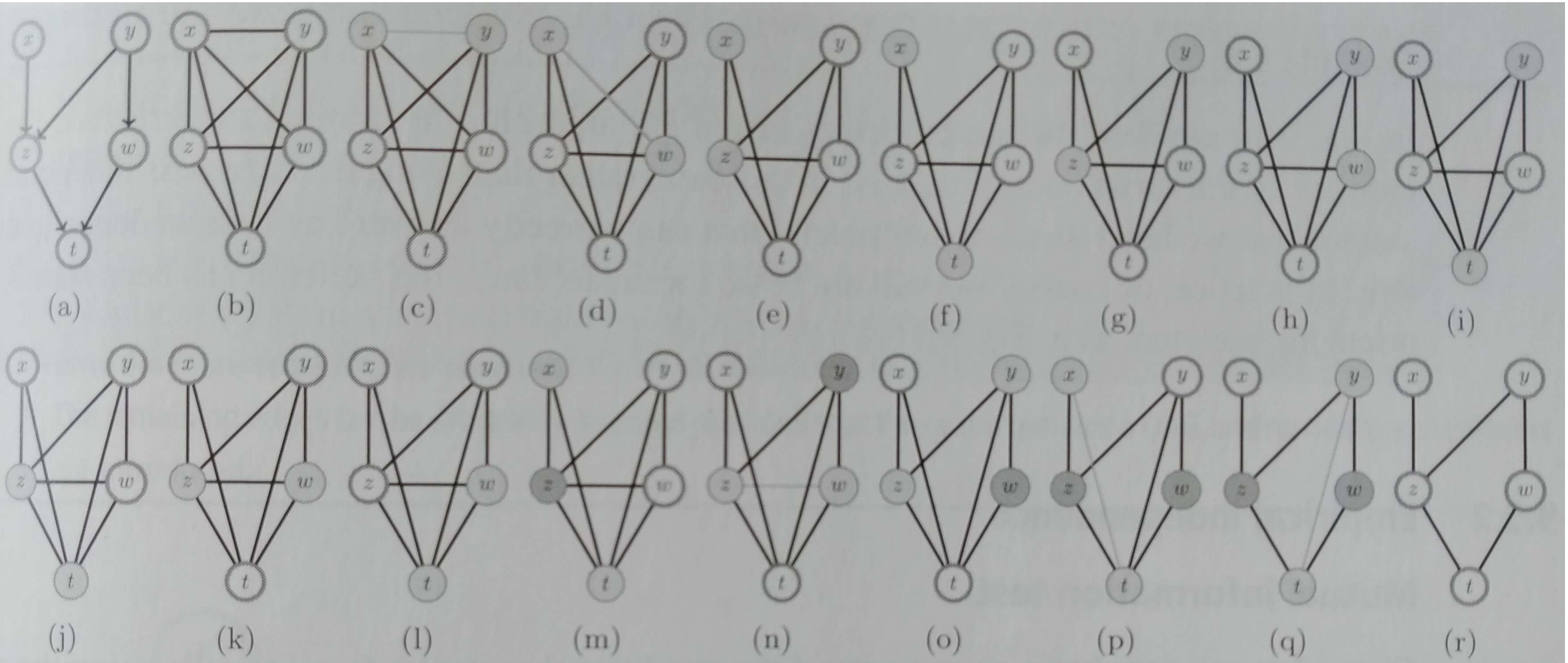


Figure 9.10 PC algorithm. (**a**) The BN from which data is assumed generated and against which conditional independence tests will be performed. (**b**) The initial skeleton is fully connected. (**c–l**) In the first round ($i = 0$) all the pairwise mutual informations $x \perp\!\!\!\perp y | \emptyset$ are checked, and the link between x and y removed if deemed independent (grey line). (**m–o**) $i = 1$. We now look at connected subsets on three variables x, y, z of the remaining graph, removing the link x–y if $x \perp\!\!\!\perp y | z$ is true. Not all steps are shown. (**p, q**) $i = 2$. We now examine all $x \perp\!\!\!\perp y | \{a, b\}$. The algorithm terminates after this round (when i gets incremented to 3) since there are no nodes with three or more neighbours. (**r**) Final skeleton. During this process the sets $S_{x,y} = \emptyset$, $S_{x,w} = \emptyset$, $S_{z,w} = y$, $S_{x,t} = \{z, w\}$, $S_{y,t} = \{z, w\}$ were found. See also `demoPCoracle.m`.

Algorytm PC:
utwórz w pełni połączony nieskierowany graf G zbioru wierzchołków V
i=0
repeat

for $x \in V$
do
for $y \in$ Neighbors(x)
do
sprawdź, czy istnieje podzbiór S o liczności i sąsiadów x (poza y),
dla których $x \perp\!\!\!\perp y \mid S$
jeśli istnieje taki zbiór S to usuń krawędź $x - y$ z grafu G i $S_{xy} = S$
done

done
$i = i + 1$

until wszystkie wezły mają co najwyżej i sąsiadów połączonych

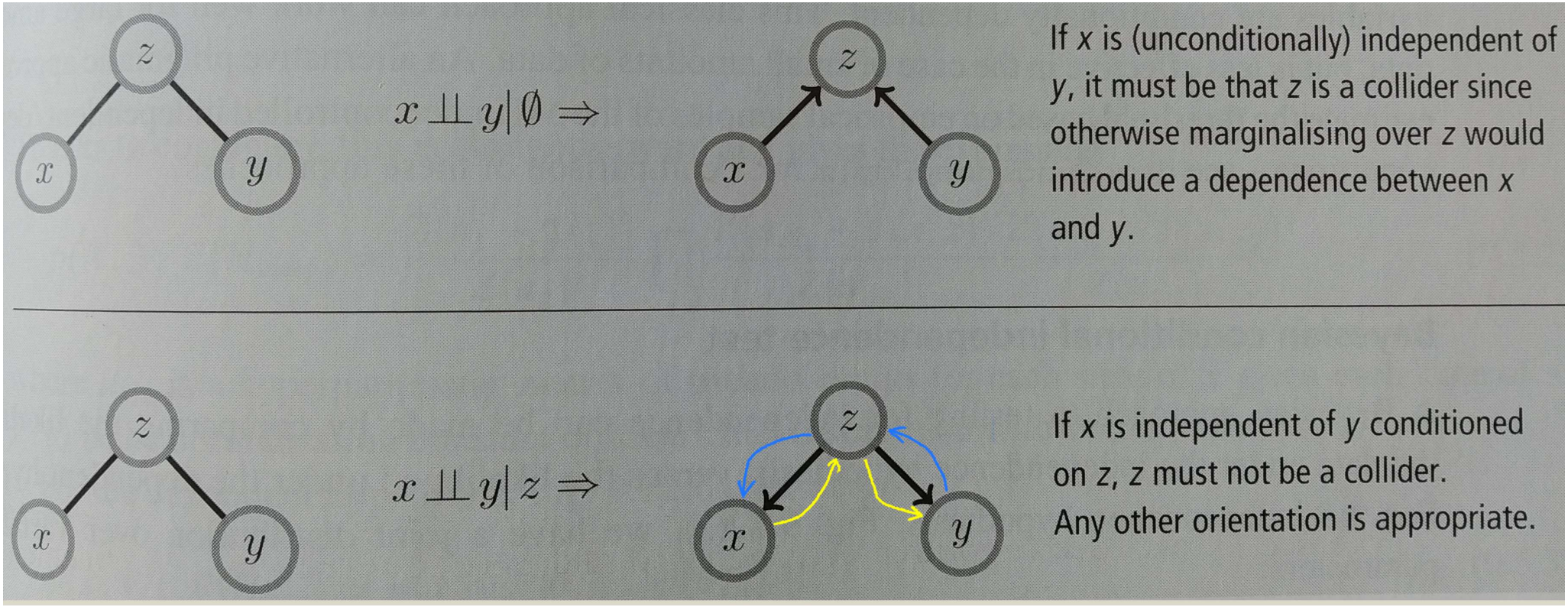
Zauważmy, że algorytm buduje zbiory S_{xy} dla wszystkich par węzłów x, y które nie mają połączenia w grafie. Zbiory te stanowią „uzasadnienie" usunięcia danej krawędzi z grafu, i będą służyć do prawidłowego skierowania pewnych łuków na ścieżce pomiędzy x i y.

Zasady eksperymentalnego określania warunkowej niezależności

Algorytm PC wymaga odpowiedzenia na pytanie, czy dwie zmienne x, y są od siebie niezależne pod warunkiem trzeciej zmiennej z. W praktyce wymaga to przyjęcia pewnego progu decyzyjnego, ponieważ dla małych zbiorów danych, dwie zmienne zawsze okazują się w pewnym stopniu — być może niewielkim — od siebie zależne.

Zasady określania kierunku łuków grafu

Określanie skierowania łuków grafu nie jest dobrze określonym algorytmem. Tylko w niektórych przypadkach można stwierdzić coś z pewnością. Takim przypadkiem jest sytuacja, gdy dwie zmienne x, y niezależne (bezwarunkowo), są połączone poprzez trzecią zmienną z. W tej sytuacji zmienna z jest nazywana **zderzaczem** (*collider*).



W przypadku, gdy w tej samej konfiguracji, zmienne x, y są niezależne warunkowo pod warunkiem z, dopuszczalne są wszystkie trzy pozostałe konfiguracje skierowań.

Określanie kierunku łuków grafu

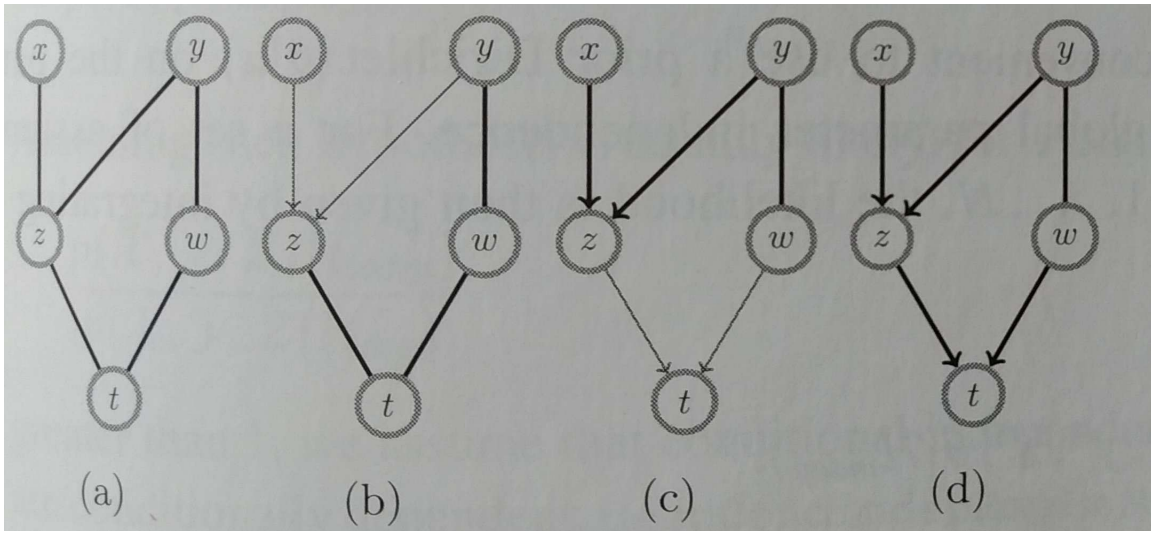


Figure 9.11 Skeleton orientation algorithm. (**a**) The skeleton along with $S_{x,y} = \emptyset$, $S_{x,w} = \emptyset$, $S_{z,w} = y$, $S_{x,t} = \{z, w\}$, $S_{y,t} = \{z, w\}$. (**b**) $z \notin S_{x,y}$, so form collider. (**c**) $t \notin S_{z,w}$, so form collider. (**d**) Final partially oriented DAG. The remaining edge may be oriented as desired, without violating the DAG condition. See also `demoPCoracle.m`.

Algorytm określania skierowania łuków

1. Sprawdź nieskierowane łuki $x - z - y$; jeśli $z \notin S_{xy}$ to ustaw $x \to z \leftarrow y$
2. **repeat**

   przekształć wszystkie $x \to z - y$ na $x \to z \to y$
   dla wszystkich $x - y$, jeśli nie istnieje skierowana ścieżka od x do y, ustaw $x \to y$
   jeśli dla $x - z - y$ istnieje w takie, że $x \to w, y \to w, z - w$, ustaw $z \to w$

until nie da się już ustawić więcej skierowań

3. pozostałe łuki mogą być skierowane dowolnie, pod warunkiem, że graf nadal nie ma cykli skierowanych, i nie zostaną wprowadzone dodatkowe *collider*y

Materiały

W tej prezentacji wykorzystane zostały materiały z następujących opracowań:

1. Andrew Ng: Unsupervised learning, Coursera video lecture

2. Stuart J. Russell, Peter Norvig: Artificial Intelligence A Modern Approach (Third Edition), Prentice-Hall, 2010

3. Kevin P. Murphy: Machine Learning A Probabilistic Perspective, MIT Press, 2012

4. Pedro Domingos: Data Mining, Machine learning, cykl wykładów wideo dostępnych przez Youtube, Paul G. Allen School of Computer Science & Engineering, University of Washington, 2016

5. Wikipedia: Collaborative filtering
`https://en.wikipedia.org/wiki/Collaborative_filtering`

6. Wikipedia: Matrix factorization (recommender systems)
`https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems)`

7. David Barber: Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012