Кейс стади 2

Обработка текста: bag of words. Теорема Байеса и наивный байесовский классификатор. Анализ временного ряда. Кластеризация: k-means, EM-алгоритм. Определение тональности текста. Определение спама в тексте. Тестирование гипотез (статистические тесты на нескольких выборках).

Даниил Корбут
Специалист по Анализу Данных

Даниил Корбут
DL Researcher
Insilico Medicine, Inc

Окончил бакалавриат ФИВТ МФТИ (Анализ данных) в 2018г
Учусь на 2-м курсе магистратуры ФИВТ МФТИ
Работал в Statsbot и Яндекс. Алиса.
Сейчас в Insilico Medicine, Inc, занимаюсь генерацией активных молекул и исследованиями старения с помощью DL.

Классическое Обучение

Данные заранее категоризированы или численные

Данные никак не размечены

С учителем

Предсказать категорию

Предсказать значение

Классификация

«Разложи носки по цвету»

Регрессия

«Разложи галстуки по длине»

Без учителя

Разделить по схожести

Найти зависимости

Выявить последовательности

Кластеризация

«Разложи похожие вещи по кучкам»

Ассоциация

«Найди какие вещи я часто ношу вместе»

Уменьшение Размерности (обобщение)

«Собери из вещей лучшие наряды»

https://vas3k.ru/blog/machine_learning/

НЕТОЛОГИЯ

Простейший спам-фильтр

(использовались года до 2010)

привет...	1829
валера ...	1710
нет ...	1191
куда ...	1012
небо ...	985
огурцы ...	873
говорить...	747
третий ...	739

нормальные письма

672 раза

«котик»

13 раз

виагра ...	1552
казино ...	1492
100% ...	1320
кредит...	1184
скидка ...	985
нажми ...	873
free ...	747
доход ...	739

спам-письма

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

формула Байеса

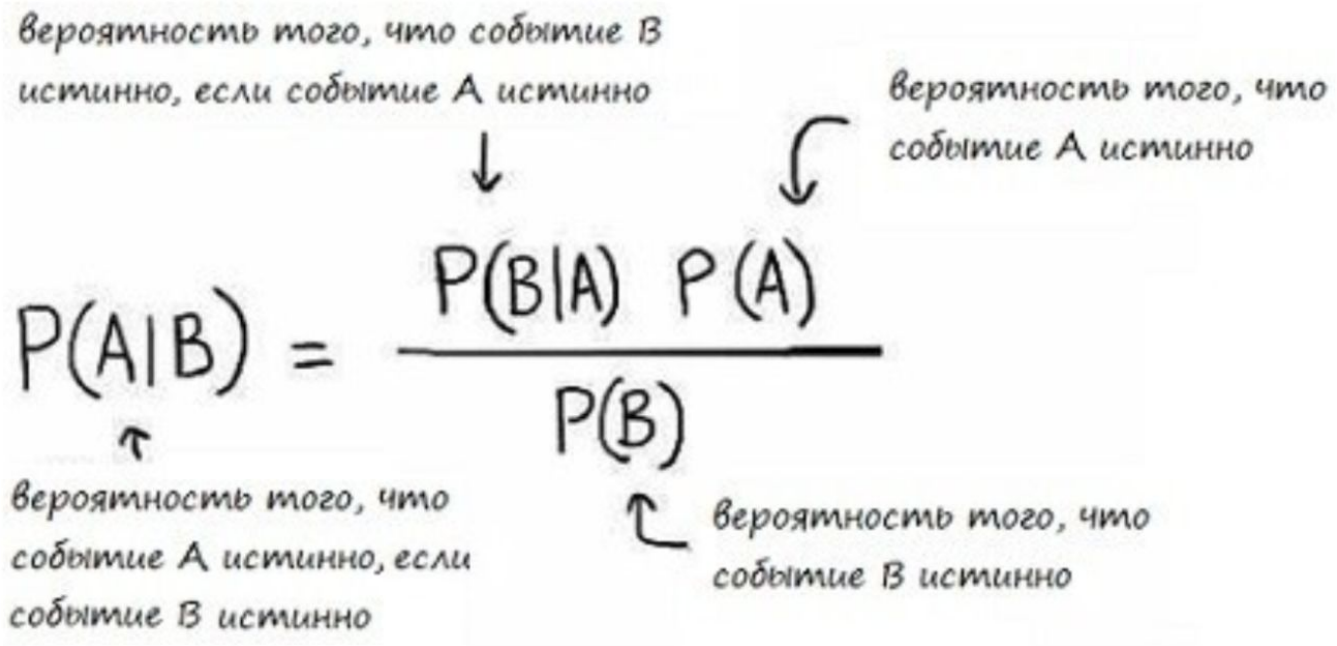
не спам

Наивный Байес

https://vas3k.ru/blog/machine_learning/

НЕТОЛОГИЯ

Формула Байеса



http://baguzin.ru/wp/den-morris-teorema-bajesa-vizualnoe-vvedenie-dlya-nachinayushhih/

Наивный Байес

Пусть у нас есть строка текста O. Кроме того, имеются классы C, к одному из которых мы должны отнести строку. Нам необходимо найти такой класс c, при котором его вероятность для данной строки была бы максимальна.

$$c = \arg\max_C P(C|O)$$

Вычислить P(C|O) сложно. Но можно воспользоваться теоремой Байеса и перейти к косвенным вероятностям:

$$P(C|o_1 o_2 ... o_n) = \frac{P(o_1 o_2 ... o_n|C)P(C)}{P(o_1 o_2 ... o_n)}$$

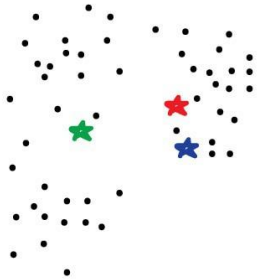
Но это опять сложно. Здесь включаем «наивное» предположение о том, что переменные O зависят только от класса C, и не зависят друг от друга. Это сильно упрощение, но зачастую это работает. Числитель примет вид:

$$P(C)P(o_1|C)P(o_2|Co_1)...P(o_n|Co_1 o_2 ... o_n) = P(C)P(o_1|C)P(o_2|C)..P(o_n|C) = P(C)\prod_i (o_i|C)$$

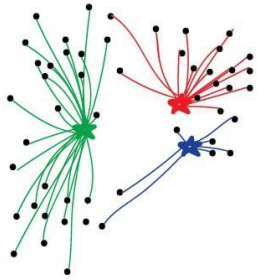
$$c = \arg\max_{c \in C} P(c|o_1 o_2 ... o_n) = \arg\max_{c \in C} P(c)\prod P(o_i|c)$$

Ставим три ларька с шаурмой оптимальным образом

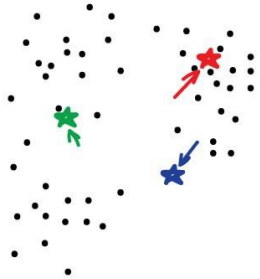
(иллюстрируя метод K-средних)



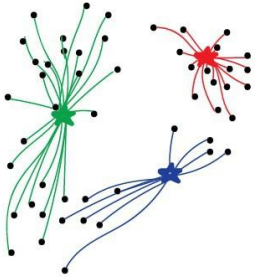
1. Ставим ларьки с шаурмой в случайных местах



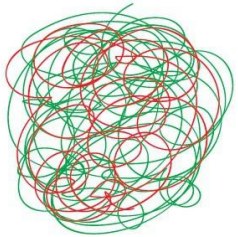
2. Смотрим в какой кому ближе идти



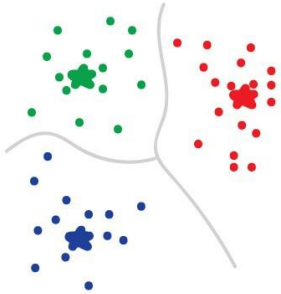
3. Двигаем ларьки ближе к центрам их популярности



4. Снова смотрим и двигаем



5. Повторяем много раз

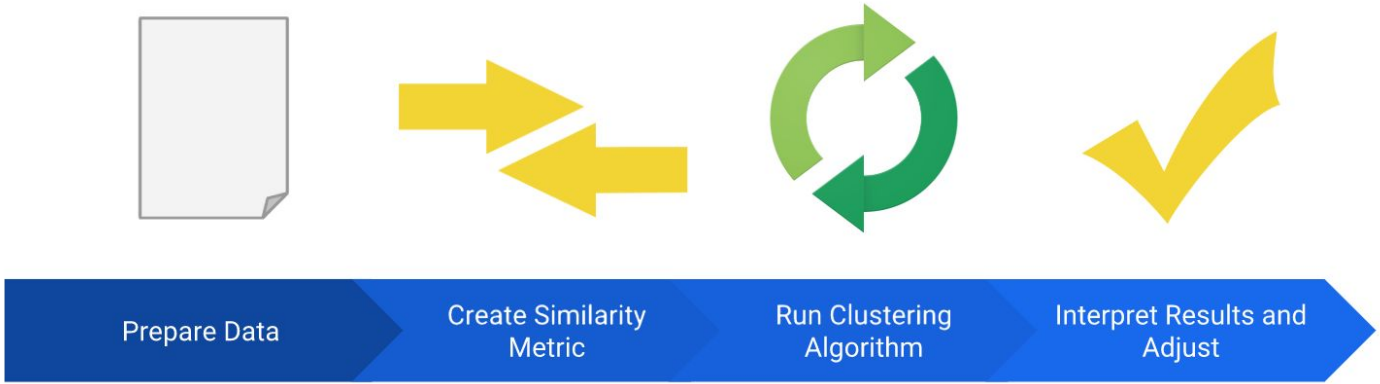


6. Готово, вы великолепны!

https://vas3k.ru/blog/machine_learning/

Кластеризация

1. Как и в случае остальных задач МЛ, необходимо нормализовать, отмасштабировать и трансформировать признаки. Дополнительно, необходимо убедиться, что данные позволяют удобно вычислить метрику похожести между объектами
2. Перед тем, как сгруппировать данные, алгоритм кластеризации должен знать, насколько похожи пары объектов. Записать эту похожесть численно позволяет метрика похожести. Её создание требует хорошего понимания данных и способа измерения похожести между признаками

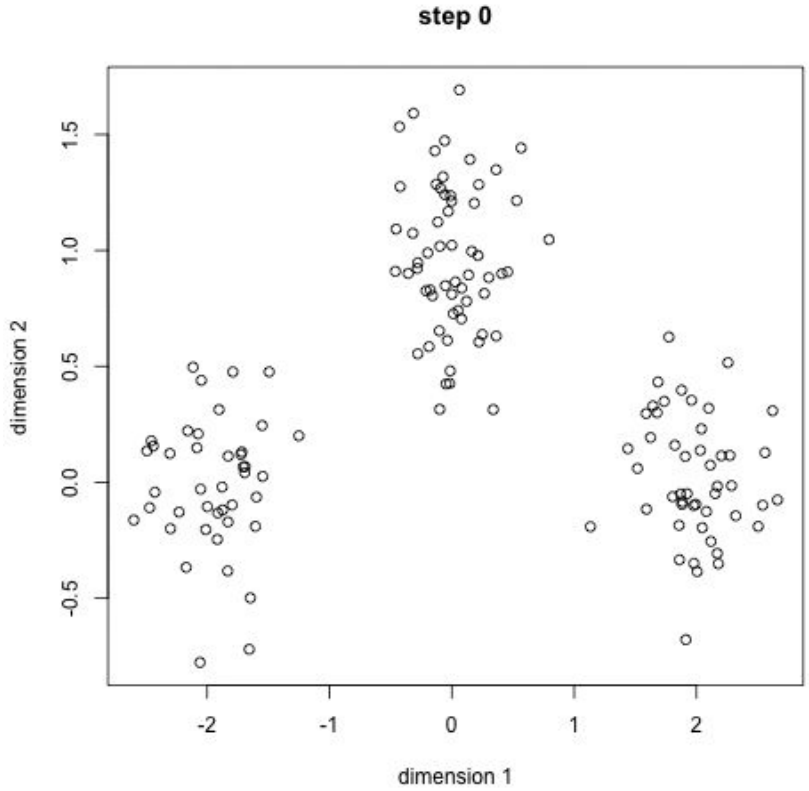


https://developers.google.com/machine-learning/clustering

Clustering: k-means

1) Инициализируем центры случайными объектами
2) Относим каждый объект к ближайшему из центров и присваиваем ему соответствующий класс
3) Двигаем центры в центры масс получившихся кластеров
4) Goto 2) пока сдвиг на 3) >= eps

Одна итерация за O(nk)



http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

Clustering: EM

- Observations $x_1 \dots x_n$
  - K=2 Gaussians with unknown μ, σ^2

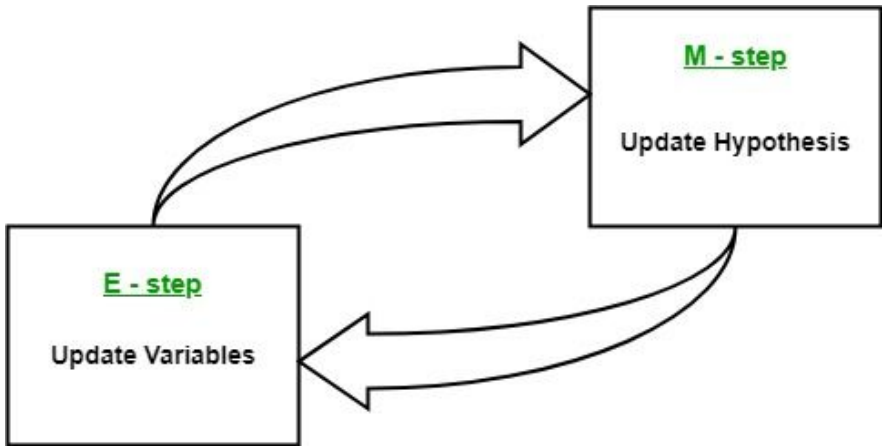
- Chicken and egg problem
  - need (μ_a, σ_a^2) and (μ_b, σ_b^2) to guess source of points
  - need to know source to estimate (μ_a, σ_a^2) and (μ_b, σ_b^2)
- EM algorithm
  - start with two randomly placed Gaussians (μ_a, σ_a^2), (μ_b, σ_b^2)

E-step:
  - for each point: $P(b|x_i)$ = does it look like it came from b?

M-step:
  - adjust (μ_a, σ_a^2) and (μ_b, σ_b^2) to fit points assigned to them
  - iterate until convergence

https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Кластеризация: EM-алгоритм



https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Bag of words

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

15

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

https://en.wikipedia.org/wiki/Bag-of-words_model

Bag of words

Несмотря на потерю информации о порядке слов в тексте, можно вычислять расстояние между векторами, например, с помощью косинусной метрики.
Мы можем пойти дальше и представить наш корпус (набор текстов) в виде матрицы “слово-документ” (term-document).

$$
\begin{array}{c} X \\ (\mathbf{d}_j) \\ \downarrow \end{array}
\qquad
U \qquad \Sigma \qquad
\begin{array}{c} V^T \\ (\hat{\mathbf{d}}_j) \\ \downarrow \end{array}
$$

$$
(\mathbf{t}_i^T) \rightarrow
\begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix}
=
(\hat{\mathbf{t}}_i^T) \rightarrow
\begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix}
\cdot
\begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix}
\cdot
\begin{bmatrix} [& \mathbf{v}_1 &] \\ & \vdots & \\ [& \mathbf{v}_l &] \end{bmatrix}
$$

https://habr.com/ru/company/ods/blog/329410/

TF-IDF: term frequency — inverse document frequency

$$TF - IDF(w, d, C) = \frac{count(w, d)}{count(d)} * log(\frac{\sum_{d' \in C} 1(w, d')}{|C|})$$

Итак, TF — это частота слова w в тексте d, здесь нет ничего сложного.
А вот IDF — существенно более интересная вещь: это логарифм обратной частоты распространенности слова w в корпусе C. Распространенностью называется отношение числа текстов, в которых встретилось искомое слово, к общему числу текстов в корпусе. С помощью TF-IDF тексты также можно сравнивать, и делать это можно с меньшей опаской, чем при использовании обычных частот.

https://habr.com/ru/company/ods/blog/329410/

Спасибо за внимание!