112.58
Рейтинг

Open Data Science

Крупнейшее русскоязычное Data Science сообщество

Подписан ×

Laggg 23 янв 2020 в 14:00

SVM. Подробный разбор метода опорных векторов, реализация на python

15 мин 123K

Блог компании Open Data Science, Python*, Data Mining*, Алгоритмы*, Машинное обучение*

Привет всем, кто выбрал путь ML-самурая!

Введение:

В данной статье рассмотрим метод опорных векторов (*англ. SVM, Support Vector Machine*) для задачи классификации. Будет представлена основная идея алгоритма, вывод настройки его весов и разобрана простая реализация своими руками. На примере датасета $Iris$ будет продемонстрирована работа написанного алгоритма с линейно разделимыми/неразделимыми данными в пространстве R^2 и визуализация обучения/прогноза. Дополнительно будут озвучены плюсы и минусы алгоритма, его модификации.

Рисунок 1. Фото цветка ириса из открытых источников

Решаемая задача:

Будем решать задачу бинарной (когда класса всего два) классификации. Сначала алгоритм тренируется на объектах из обучающей выборки, для которых заранее известны метки классов. Далее уже обученный алгоритм предсказывает метку класса для каждого объекта из отложенной/тестовой выборки. Метки классов могут принимать значения $Y = \{-1, +1\}$. Объект — вектор с N признаками $x = (x_1, x_2, \ldots, x_n)$ в пространстве R^n. При обучении алгоритм должен построить функцию $F(x) = y$, которая принимает в себя аргумент x — объект из пространства R^n и выдает метку класса y.

Общие слова об алгоритме:

Задача классификации относится к обучению с учителем. SVM — алгоритм обучения с учителем. Наглядно многие алгоритмы машинного обучения можно посмотреть в этой топовой статье (см. раздел «Карта мира машинного обучения»). Нужно добавить, что SVM может применяться и для задач регрессии, но в данной статье будет разобран SVM-классификатор.

Главная цель SVM как классификатора — найти уравнение разделяющей гиперплоскости $w_1x_1 + w_2x_2 + \ldots + w_nx_n + w_0 = 0$ в пространстве R^n, которая бы разделила два класса неким оптимальным образом. Общий вид преобразования F объекта x в метку класса Y: $F(x) = sign(w^T x - b)$. Будем помнить, что мы обозначили $w = (w_1, w_2, \ldots, w_n), b = -w_0$. После настройки весов алгоритма w и b (обучения), все объекты, попадающие по одну сторону от

построенной гиперплоскости, будут предсказываться как первый класс, а объекты, попадающие по другую сторону — второй класс.

Внутри функции $sign()$ стоит линейная комбинация признаков объекта с весами алгоритма, именно поэтому SVM относится к линейным алгоритмам. Разделяющую гиперплоскость можно построить разными способами, но в SVM веса w и b настраиваются таким образом, чтобы объекты классов лежали как можно дальше от разделяющей гиперплоскости. Другими словами, алгоритм максимизирует зазор (*англ. margin*) между гиперплоскостью и объектами классов, которые расположены ближе всего к ней. Такие объекты и называют опорными векторами (см. рис.2). Отсюда и название алгоритма.

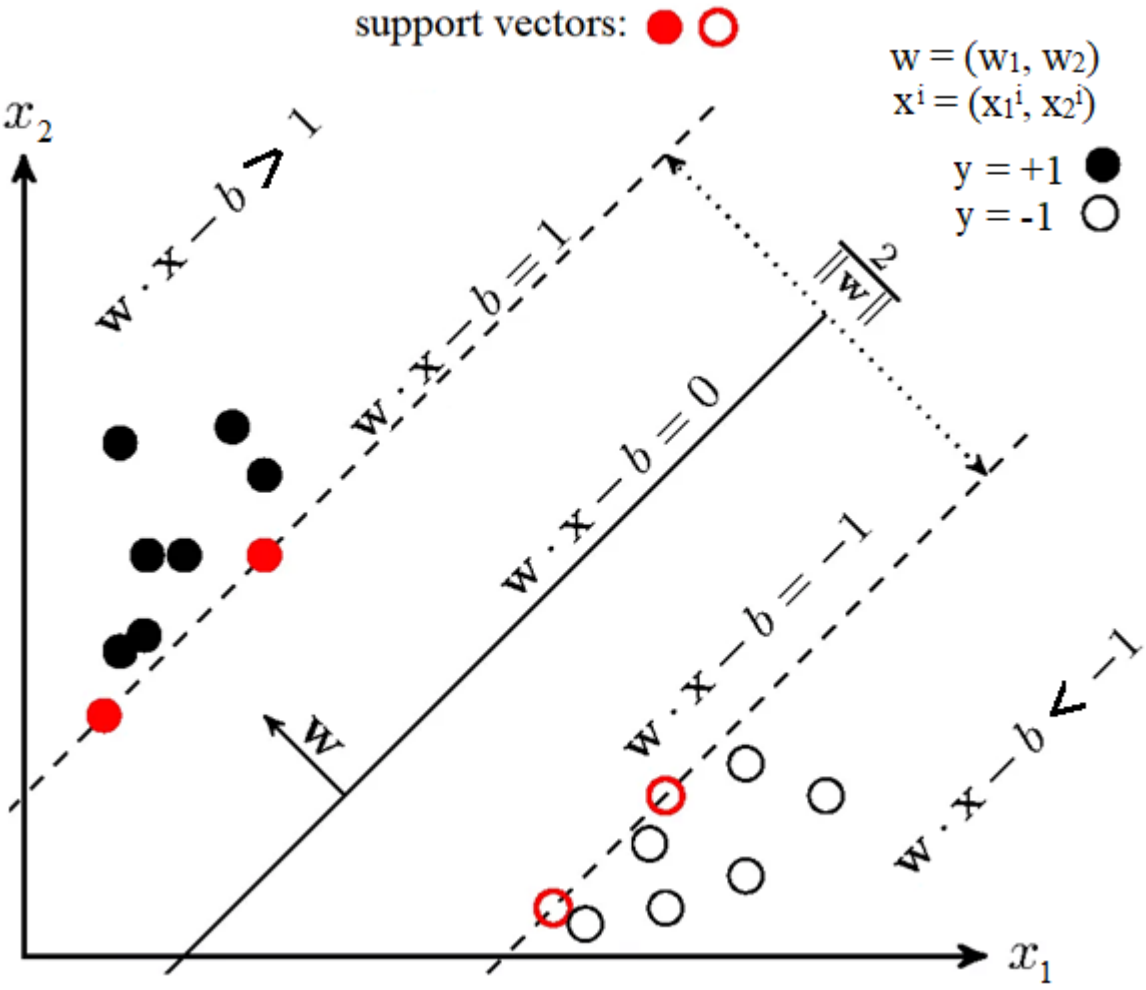


Рисунок 2. SVM (основа рисунка отсюда)

Подробный вывод правил настройки весов SVM:

Чтобы разделяющая гиперплоскость как можно дальше отстояла от точек выборки, ширина полосы должна быть максимальной. Вектор w — вектор нормали к разделяющей гиперплоскости. Здесь и далее будем обозначать скалярное произведение двух векторов как $\langle a, b \rangle$ или $a^T b$. Давайте найдем проекцию вектора, концами которого являются опорные вектора разных классов, на вектор w. Эта проекция и будет показывать ширину разделяющий полосы (см. рис.3):

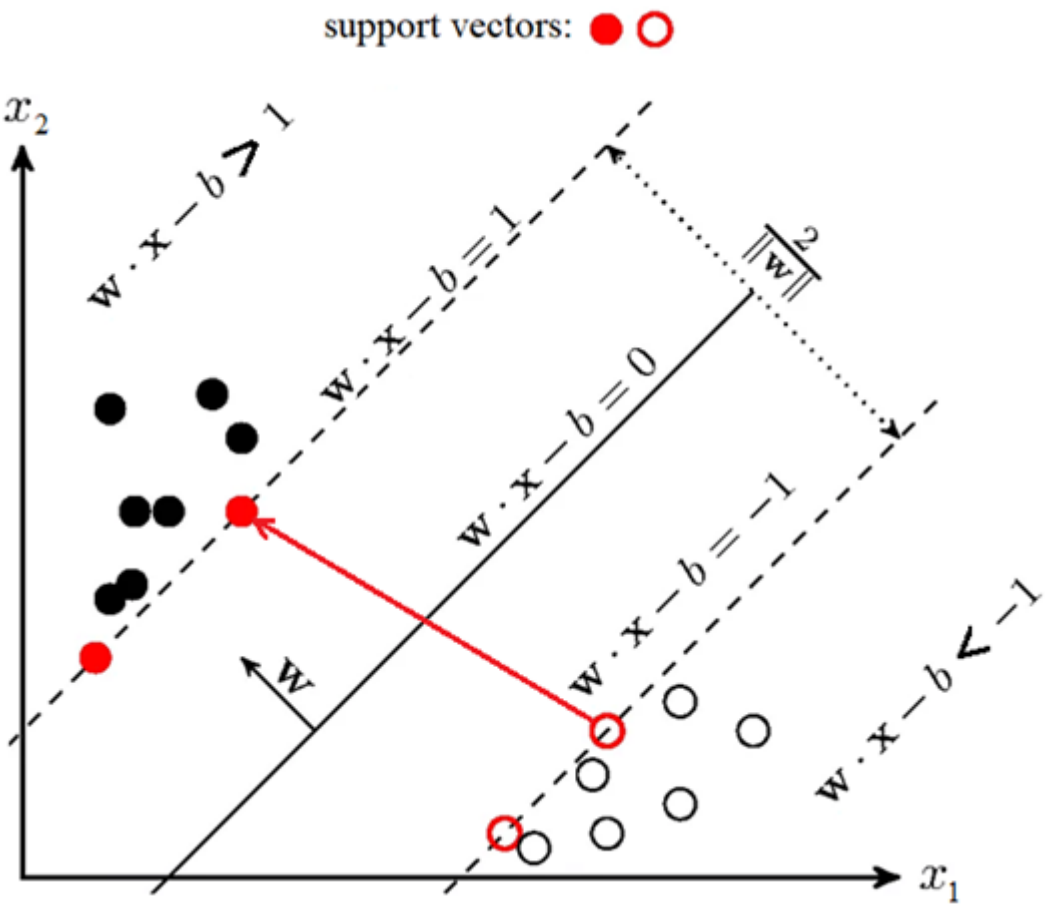


Рисунок 3. Вывод правил настройки весов (основа рисунка отсюда)

$$\langle (x_+ - x_-), w/\|w\| \rangle = (\langle x_+, w \rangle - \langle x_-, w \rangle)/\|w\| = ((b+1) - (b-1))/\|w\| = 2/\|w\|$$

$$2/\|w\| \to max$$

$$\|w\| \to min$$

$$(w^T w)/2 \to min$$

Отступом (*англ. margin*) объекта x от границы классов называется величина $M = y(w^T x - b)$. Алгоритм допускает ошибку на объекте тогда и только тогда, когда отступ M отрицателен (когда y и $(w^T x - b)$ разных знаков). Если $M \in (0, 1)$, то объект попадает внутрь разделяющей полосы. Если $M > 1$, то объект x классифицируется правильно, и находится на некотором удалении от разделяющей полосы. Т.е. алгоритм будет правильно классифицировать объекты, если выполняется условие:

$$y(w^T x - b) \geqslant 1$$

Если объединить два выведенных выражения, то получим дефолтную настройку *SVM с жестким зазором* (*hard-margin SVM*), когда никакому объекту не разрешается попадать на

полосу разделения. Решается аналитически через теорему Куна-Таккера. Получаемая задача эквивалентна двойственной задаче поиска седловой точки функции Лагранжа.

$$\begin{cases} (w^T w)/2 \to min \\ y(w^T x - b) \geqslant 1 \end{cases}$$

Всё это хорошо до тех пор, пока у нас классы линейно разделимы. Чтобы алгоритм смог работать и с линейно неразделимых данными, давайте немного преобразуем нашу систему. Позволим алгоритму допускать ошибки на обучающих объектах, но при этом постараемся, чтобы ошибок было поменьше. Введём набор дополнительных переменных $\xi_i > 0$, характеризующих величину ошибки на каждом объекте x_i. Введём в минимизируемый функционал штраф за суммарную ошибку:

$$\begin{cases} (w^T w)/2 + \alpha \sum \xi_i \to min \\ y(w^T x_i - b) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases}$$

Будем считать количество ошибок алгоритма (когда M<0). Назовем это штрафом (*Penalty*). Тогда штраф для всех объектов будет равен сумме штрафов для каждого объекта x_i, где $[M_i < 0]$ — пороговая функция (см. рис.4):

$$Penalty = \sum [M_i < 0]$$

$$[M_i < 0] = \begin{cases} 1 & \text{, если } M_i < 0 \\ 0 & \text{, если } M_i \geqslant 0 \end{cases}$$

Далее сделаем штраф чувствительным к величине ошибки (чем сильнее M "уходит в минус" — тем больше штраф) и заодно введем штраф за приближение объекта к границе классов. Для этого возьмем функцию, которая ограничивает пороговую функцию ошибки (см. рис.4):

$$Penalty = \sum [M_i < 0] \leqslant \sum (1 - M_i)_+ = \sum max(0, 1 - M_i)$$

При добавлении к выражению штрафа слагаемое $\alpha(w^T w)/2$ получаем классическую фукцию потерь *SVM с мягким зазором* (*soft-margin SVM*) для одного объекта:

$$Q = max(0, 1 - M_i) + \alpha(w^T w)/2$$

$$Q = max(0, 1 - yw^T x) + \alpha(w^T w)/2$$

Q — функция потерь, она же loss function. Именно ее мы и будем минимизировать с помощью градиентного спуска в реализации руками. Выведем правила изменения весов, где η – шаг спуска:

$$w = w - \eta \bigtriangledown Q$$

$$\bigtriangledown Q = \begin{cases} \alpha w - yx & \text{, если } yw^T x < 1 \\ \alpha w & \text{, если } yw^T x \geqslant 1 \end{cases}$$

Возможные вопросы на собеседованиях (основано на реальных событиях):

После общих вопросов про SVM: *Почему именно Hinge_loss максимизирует зазор?* – для начала вспомним, что гиперплоскость меняет свое положение тогда, когда изменяются веса w и b. Веса алгоритма начинают меняться, когда градиенты лосс-функции не равны нулю (обычно говорят: “градиенты текут”). Поэтому мы специально подобрали такую лосс-функцию, у которой начинают течь градиенты в нужное время. $Hingeloss$ выглядит следующим образом: $H = max(0, 1 - y(w^T x))$. Помним, что зазор $m = y(w^T x)$. Когда зазор m достаточно большой (1 или более), выражение $(1 - m)$ становится меньше нуля и $H = 0$ (поэтому градиенты не текут и веса алгоритма никак не изменяются). Если же зазор m достаточно мал (например, когда объект попадает на полосу разделения и/или отрицательный (при неверном прогнозе классификации), то Hinge_loss становится положительной ($H > 0$), начинают течь градиенты и веса алгоритма изменяются. Резюмируя: градиенты текут в двух случаях: когда объект выборки попал внутрь полосы разделения и при неправильной классификации объекта.

Для проверки уровня иностранного языка возможны подобные вопросы: *What are the similarities and differences between LogisticRegression and SVM?* – firstly, we will talk about similarities: both of algorithms are linear classification algorithms in supervised learning. Some similarities are in their arguments of loss functions: $log(1 + exp(-y(w^T x)))$ for LogReg and $max(0, 1 - y(w^T x))$ for SVM (look at picture 4). Both of algorithms we can configure using gradient descent. Next let’s talk about differences: SVM return class label of object unlike LogReg, which return probability of class membership. SVM can’t work with class labels $\{0, 1\}$ (without renaming classes) unlike LogReg (LogReg loss finction for $\{0, 1\}$: $-ylog(p) - (1 - y)log(1 - p)$, where y – real class label, p – algorithm’s return, probability of belonging object x to class$\{1\}$). More than that, we can solve hard-margin SVM problem without gradient descent. The task of searching support vectors is reduced to search saddle point in the Lagrange function – this task refers to quadratic programming only.

- Loss function's code:

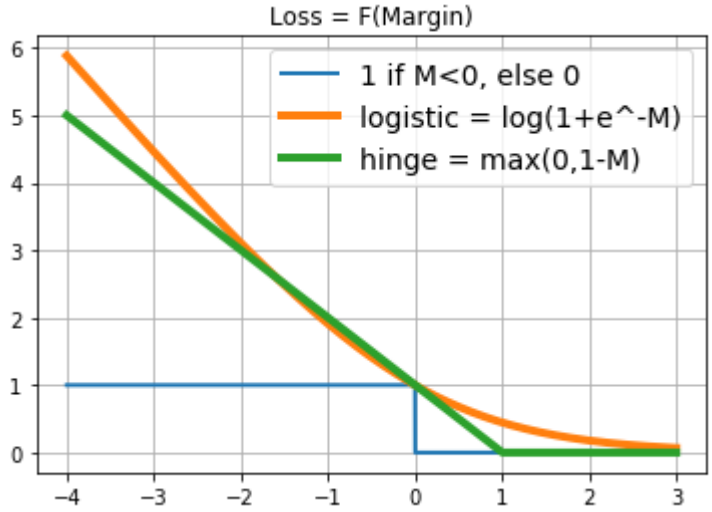


Рисунок 4. Функции потерь

Простая имплементация классического soft-margin SVM:

Внимание! Ссылку на полный код вы найдете в конце статьи. Ниже будут представлены блоки кода, вырванные из контекста. Некоторые блоки можно запускать только после отработки предыдущих блоков. Под многими блоками будут размещены картинки, которые показывают, как отработал код, размещенный над ней.

- Сначала подрубим нужные библиотеки и функцию отрисовки линии:

- Python код реализации soft-margin SVM:

Подробно рассмотрим работу каждого блока строчек:

1) создаем функцию *add_bias_feature(a)*, которая автоматически расширяет вектор объектов, добавляя в конец каждого вектора число 1. Это нужно для того, чтобы «забыть» про свободный член b. Выражение $w^T x - b$ эквивалентно выражению $w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_0 * 1$. Мы условно считаем, что единица — это последняя компонента вектора для всех векторов x, а $w_0 = -b$. Теперь настройку весов w и w_0 будем производить одновременно.

- Код функции расширения вектора признаков:

2) далее опишем сам классификатор. Он имеет внутри себя функции инициализации *init()*, обучения *fit()*, предсказания *predict()*, нахождения лосс функции *hinge_loss()* и нахождения общей лосс функции классического алгоритма с мягким зазором *soft_margin_loss()*.

3) при инициализации вводятся 3 гиперпараметра: _etha – шаг градиентного спуска (η), _alpha – коэффициент быстроты пропорционального уменьшения весов (перед квадратичным слагаемым

в функции потерь α), _epochs – количество эпох обучения.

- Код функции инициализации:

4) при обучении для каждой эпохи обучающей выборки (X_train, Y_train) мы будем брать по одному элементу из выборки, вычислять зазор между этим элементом и положением гиперплоскости в данный момент времени. Далее в зависимости от величины этого зазора мы будем изменять веса алгоритма с помощью градиента функции потерь Q. Заодно будем вычислять значение этой функции на каждой эпохе и сколько раз мы изменяем веса за эпоху. Перед началом обучения убедимся, что в функцию обучения пришло действительно не больше двух разных меток класса. Перед настройкой весов происходит их инициализация с помощью нормального распределения.

- Код функции обучения:

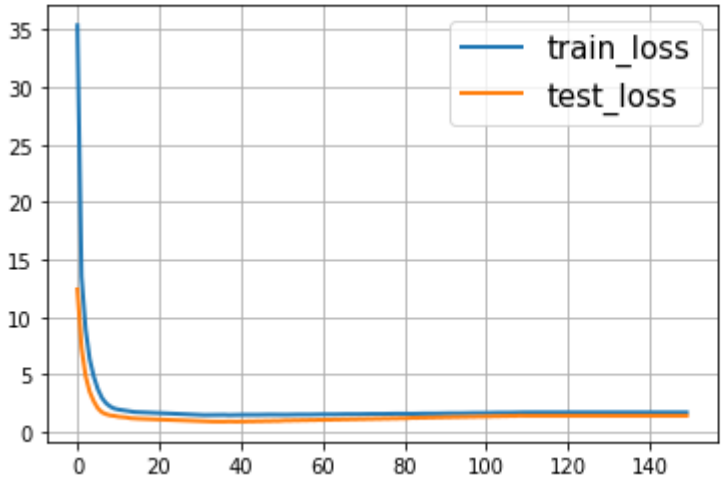
Проверка работы написанного алгоритма:

Проверим, что наш написанный алгоритм работает на каком-нибудь игрушечном наборе данных. Возьмем датасет Iris. Подготовим данные. Обозначим классы 1 и 2 как $+1$, а класс 0 как -1. С помощью алгоритма PCA (объяснение и применение тут) оптимальным образом сократим пространство 4-х признаков до 2-х с минимальными потерями данных (нам будет проще наблюдать за обучением и разультатом). Далее разделим на обучающую (трейн) выборку и отложенную (валидационную). Обучим на трейн выборке, прогнозируем и проверяем на отложенной. Подберем коэффициенты обучения таким образом, чтобы лосс функция падала. Во время обучения будем смотреть на лосс функцию обучающей и отложенной выборки.

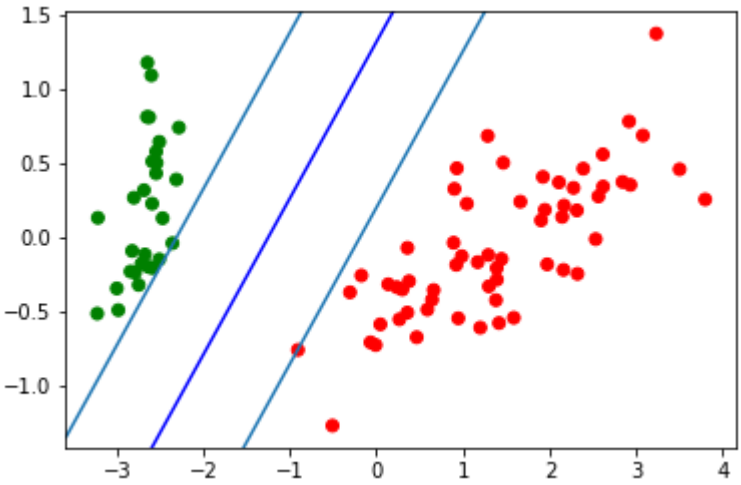
- Блок подготовки данных:

- Блок инициализации и обучения:

```
[65 32 24 19 14 15 13 10  9  7  6  5  5  6  5  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  3  3  3  3  3  2  2  2  1  2  1  2  1  2  1  2
  1  2  1  1  2  1  2  1  2  1  2  1  2  1  2  1  2  1  2  1  2  1  2  1
  2  1  2  1  2  1  2  1  2  1  1  2  1  2  1  2  1  2  1  2  1  2  1  2
  1  2  1  2  1  2  1  2  1  2  1  2  1  2  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0]
[ 0.96553668 -0.85848583  1.22764817]
```

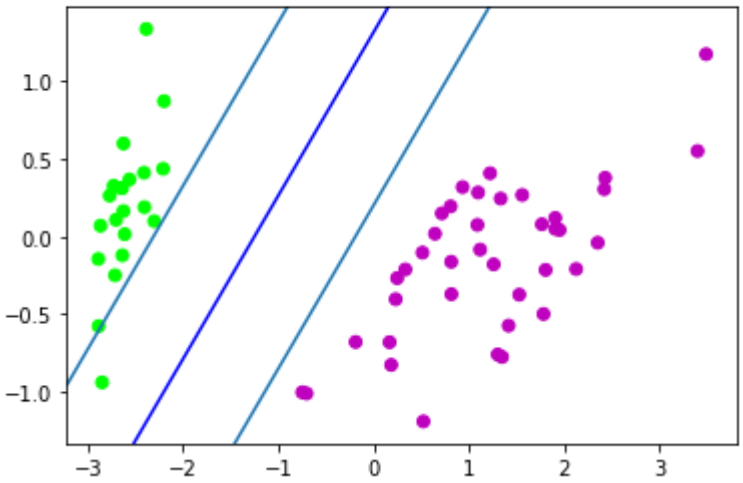


- Блок визуализации получившейся разделяющей полосы:



- Блок визуализации прогноза:

```
Количество ошибок для отложенной выборки:  0
```

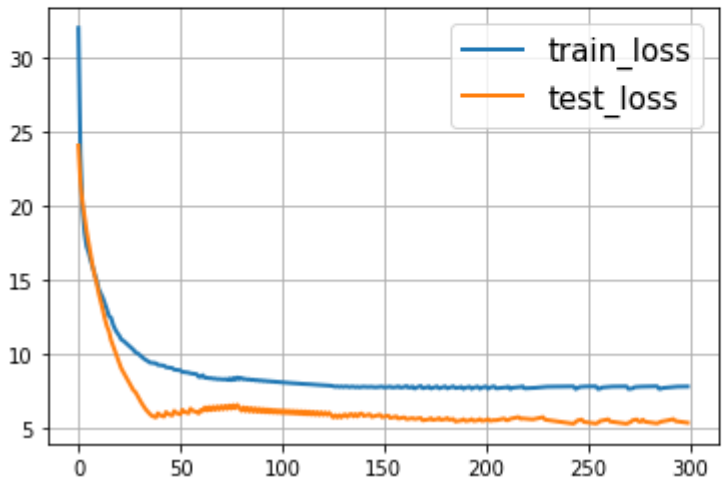


Отлично! Наш алгоритм справился с линейно разделимыми данными. Теперь заставим его отделить классы 0 и 1 от класса 2:

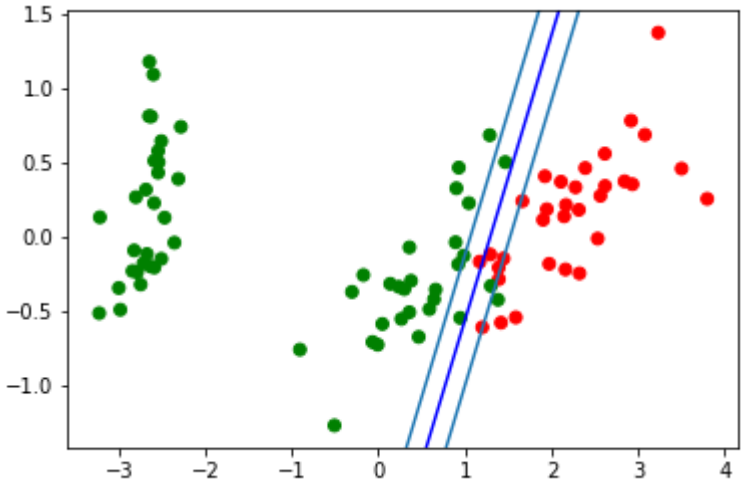
- Блок подготовки данных:

- Блок инициализации и обучения:

```
[49 39 34 29 29 25 25 25 25 24 25 24 23 22 23 24 20 22 22 22 19 18 17 17
 17 17 17 17 20 17 17 17 17 17 17 15 15 15 15 14 15 15 15 14 15 15 15 14
 15 15 15 14 15 15 15 14 15 15 15 13 12 15 12 13 12 13 12 13 12 13 12 13
 12 13 12 13 12 13 12 13 11 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12
 11 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12 11 12
 11 12 11 12 11 11 12 11 12 11 12 10 11 12 10 11 12 11 12 10 11 10 10 10
 11 10 10 10 10 11]
[ 4.3594492  -2.27373821 -5.62161378]
```

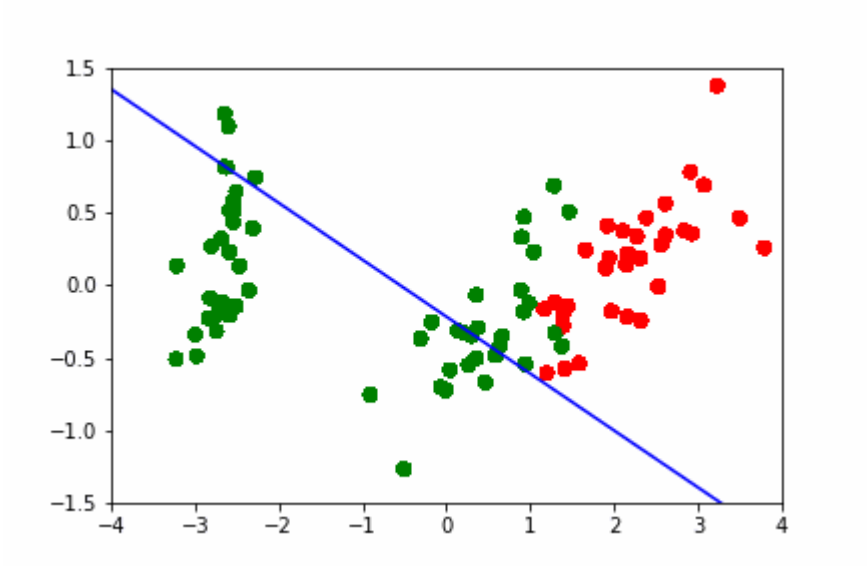


- Блок визуализации получившейся разделяющей полосы:



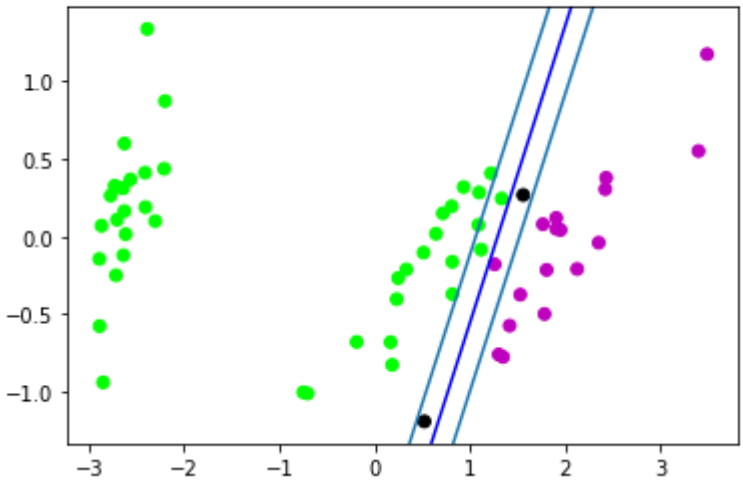
Посмотрим на гифку, которая покажет, как разделяющая прямая изменяла свое положение во время обучения (всего 500 кадров изменения весов. Первые 300 подряд. Далее 200 штук на каждый 130ый кадр):

- Код создания анимации:



- Блок визуализации прогноза:

```
Количество ошибок для отложенной выборки:  2
```



Спрямляющие пространства

Важно понимать, что в реальных задачах не будет простого случая с линейно разделимыми данными. Для работы с подобными данными была предложена идея перехода в другое пространство, где данные будут линейно разделимы. Такое пространство и называется спрямляющим. В данной статье не будут затронуты спрямляющие пространства и ядра. Самую полную математическую теорию Вы сможете найти в 14,15,16 конспектах Е. Соколова и в лекциях К.В.Воронцова.

Применение SVM из sklearn:

В действительности, почти все классические алгоритмы машинного обучения написаны за Вас. Приведем пример кода, алгоритм возьмем из библиотеки *sklearn*.

- Пример кода

Плюсы и минусы классического SVM:

Плюсы:

1. хорошо работает с пространством признаков большого размера;
2. хорошо работает с данными небольшого объема;
3. алгоритм максимизирует разделяющую полосу, которая, как подушка безопасности, позволяет уменьшить количество ошибок классификации;
4. так как алгоритм сводится к решению задачи квадратичного программирования в выпуклой области, то такая задача всегда имеет единственное решение (разделяющая гиперплоскость с определенными гиперпараметрами алгоритма всегда одна).

Минусы:

1. долгое время обучения (для больших наборов данных);
2. неустойчивость к шуму: выбросы в обучающих данных становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости;
3. не описаны общие методы построения ядер и спрямляющих пространств, наиболее подходящих для конкретной задачи в случае линейной неразделимости классов. Подбирать полезные преобразования данных – искусство.

Применение SVM:

Выбор того или иного алгоритма машинного обучения напрямую зависит от информации, полученной во время исследования данных. Но в общих словах, можно выделить следующие задачи:

- задачи с небольшим набором данных;
- задачи текстовой классификации. SVM дает неплохой baseline ([preprocessing] + [TF-iDF] + [SVM]), получаемая точность прогноза оказывается на уровне некоторых сверточных/ рекуррентных нейронных сетей (рекомендую самостоятельно попробовать такой метод для закрепления материала). Отличный пример приведен тут, "Часть 3. Пример одного из трюков, которым мы обучаем";
- для многих задач со структурированными данными связка [feature engineering] + [SVM] + [kernel] "все еще торт";
- так как Hinge loss считается довольно быстро, ее можно встретить в Vowpal Wabbit (по умолчанию).

Модификации алгоритма:

Существуют различные дополнения и модификации метода опорных векторов, направленные на устранение определенных недостатков:

- Метод релевантных векторов (Relevance Vector Machine, RVM)
- 1-norm SVM (LASSO SVM)
- Doubly Regularized SVM (ElasticNet SVM)
- Support Features Machine (SFM)
- Relevance Features Machine (RFM)

Дополнительные источники на тему SVM:

1. Текстовые лекции К.В.Воронцова
2. Конспекты Е.Соколова — 14,15,16
3. Крутой источник by Alexandre Kowalczyk
4. На хабре есть 2 статьи, посвященные svm:
    - 2010 г.
    - 2018 г.
5. На гитхабе могу выделить 2 крутые реализации SVM по следующим ссылкам:
    - MLAlgorithms от отечественного разработчика
    - ML-From-Scratch

Заключение:

Большое спасибо за внимание! Буду благодарен за любые комментарии, отзывы и советы. Полный код из данной статьи найдете на моем гитхабе.

p.s. Благодарю @yorko за советы по сглаживанию "углов". Спасибо Алексею Сизых @asizykh — физтеху, который частично вложился в код.

Теги: open data science, ods, ods.ai, machine learning, машинное обучение, алгоритмы, classification, svm, margin, опорные вектора, support vectors, loss function, python

Хабы: Блог компании Open Data Science, Python, Data Mining, Алгоритмы, Машинное обучение

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

useful_citizen 22 часа назад

Пришли домой из-за Element

Простой 2 мин 36K

Кейс Recovery Mode

+105 44 185 +185

BabayMazay 4 часа назад

Простой высоковольтный блок для питания разрядных трубок

Простой 6 мин 1.1K

Туториал

+37 15 13 +13

PatientZero 22 часа назад

Челлендж по обработке миллиарда строк на Go: от 1 минуты 45 секунд до 4 секунд

Средний 14 мин 8.7K

Кейс Перевод

+32 86 14 +14

Lunathecat 8 часов назад

Кот украл припой. Секретная китайская плата с логическими элементами

Простой 8 мин 3.7K

Кейс

+31 22 2 +2

bodyawm 9 часов назад

Мобильные экранчики в ваших проектах: большой и понятный о гайд о различных дисплеях

Средний 18 мин 3K

Туториал

+30 88 63 +63

PatientZero 8 часов назад

Это слишком опасно для C++

6 мин 10K

Обзор Перевод

+26 37 45 +45

Firemoon 8 часов назад

Нарушаем ограничения файловых систем *NIX

11 мин 2.9K

Кейс

+24 26 4 +4

AlekseiPodkletnov 11 часов назад

Путь Nvidia. Как компания, которая чуть не закрылась после первого чипа, стала «главной по мощностям»

14 мин 3.8K

+20 27 22 +22

CyberPaul 11 часов назад

Выдающаяся женщина в IT: история Шафриры Гольдвассер

Простой 7 мин 1.8K

Ретроспектива

+17 16 6 +6

lawellet 11 часов назад

Обзор Simulator — платформы для обучения инженеров безопасности Kubernetes с помощью CTF-сценариев

Простой 17 мин 1.1K

Обзор

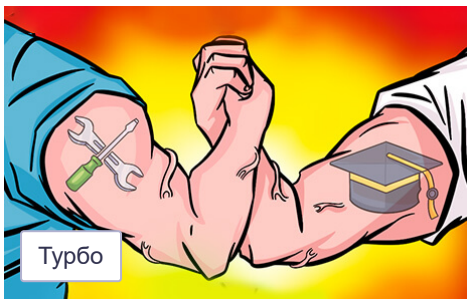
+17 14 0

Как мы разрабатывали свой Agile-велосипед и почему не используем популярные фреймворки

Турбо

Показать еще

МИНУТОЧКУ ВНИМАНИЯ



Когда сотрудник — барьер для киберпреступника

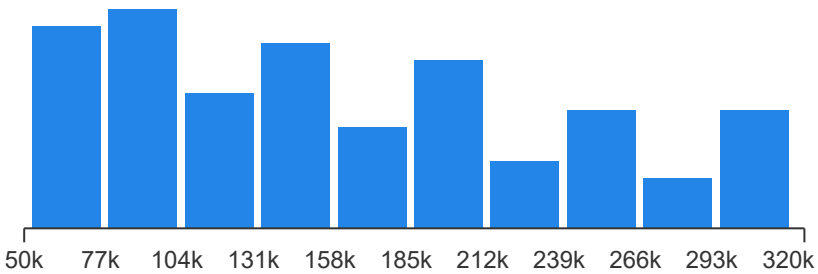
Топим снег скидками: промокодус в деле

Планируй своё время и его хватит на IT-ивенты из Календаря

СРЕДНЯЯ ЗАРПЛАТА В IT

168 812 ₽/мес.

— средняя зарплата во всех IT-специализациях по данным из 20 221 анкеты, за 1-ое пол. 2024 года. Проверьте «в рынке» ли ваша зарплата или нет!



Проверить свою зарплату

ИНФОРМАЦИЯ

Сайт	ods.ai
Дата регистрации	16 февраля 2017
Дата основания	12 марта 2015

Численность	5 001–10 000 человек
Местоположение	Россия

ССЫЛКИ

Присоединиться к сообществу ODS
ods.ai

БЛОГ НА ХАБРЕ

18 дек 2023 в 17:28

GPT-like модель «впервые сделала научное открытие»: что, как, и куда дальше?

92K 271 +271

4 дек 2023 в 09:51

Кто такие LLM-агенты и что они умеют?

22K 13 +13

11 ноя 2023 в 09:57

Главное событие в мире AI: создатель ChatGPT рассказал, в какое будущее он нас всех ведет

92K 104 +104

6 сен 2023 в 16:00

Пять книг про NLP, с которых можно начать

12K 7 +7

25 авг 2023 в 12:47

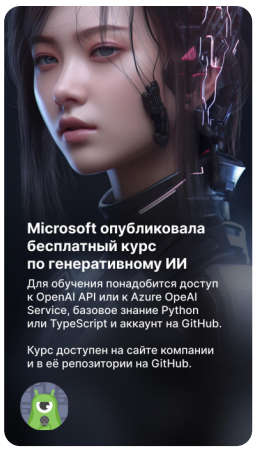
Дропаем ранжирующие метрики в рекомендательной системе, часть 3: платформа для экспериментов

2.5K 2 +2

ИСТОРИИ

OpenAI обнародовала переписку с Маском

Яндекс 360 призывает героев бэкенда



Бесплатный ИИ-курс

Полезные книги для библиотеки айтишника

Как новичкам освоиться в IT

Буке

Ваш аккаунт

- Профиль
- Трекер
- Диалоги
- Настройки
- ППА

Разделы

- Статьи
- Новости
- Хабы
- Компании
- Авторы
- Песочница

Информация

- Устройство сайта
- Для авторов
- Для компаний
- Документы
- Соглашение
- Конфиденциальность

Услуги

- Корпоративный блог
- Медийная реклама
- Нативные проекты
- Образовательные программы
- Стартапам

Настройка языка

Техническая поддержка