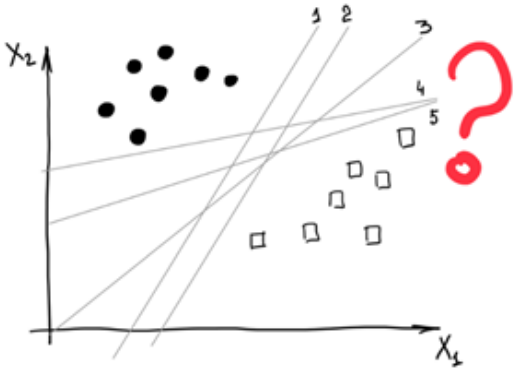
Лекция 6. SVM (Support Vector Machine) - метод опорных векторов

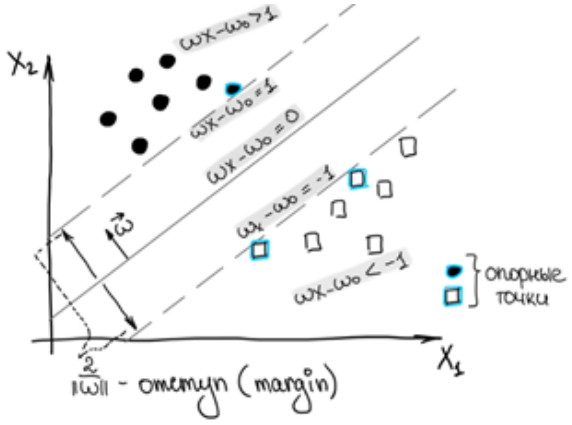
Ещё одним ярким примером линейного классификатора является метод опорных векторов (SVM). Как и в предыдущем примере с линейной регрессией, основной функционал этого алгоритма заключается в поиске уравнения разделяющей гиперплоскости, которая разделит обучающую выборку наиболее оптимальным образом.

$$f(x,\omega) = sign(\sum_{i=1}^{n} \omega_i f_i(x) - \omega_0 + \xi) = sign(X\vec{w} + \xi)$$



Но что такое “оптимальным” образом? В случае линейной регрессии у нас есть множество вариантов как именно расположить разделяющую гиперплоскость.

В SVM параметры модели настраиваются таким образом, чтобы объекты классов лежали как можно дальше от разделяющей гиперплоскости. Т.е., при обучении (настройке весовых коэффициентов) алгоритм максимизирует зазор (*англ. margin*) между гиперплоскостью и объектами классов, которые расположены ближе всего к ней. Объекты, которые находятся ближе всего к разделяющей гиперплоскости и относительно которых как раз и максимизируется зазор, и называются опорными векторами.



А как именно происходит настройка весов?

В процессе обучения алгоритма происходит максимизация полосы, разделяющей наши классы. Вектор \omega есть вектор нормали к нашей разделяющей гиперплоскости. Описав проекцию вектора, сформированного опорными векторами разных классов на вектор нормали, мы и получим ширины разделяющей полосы. Распишем проекцию этого вектора на вектор нормали через скалярное произведение векторов:

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

Т.к. нам требуется максимизировать зазор, то итоговый функционал выглядит следующим образом:

$$2/||\omega|| \to max => ||\omega|| \to min => (\omega^T \omega)/2 \to min$$

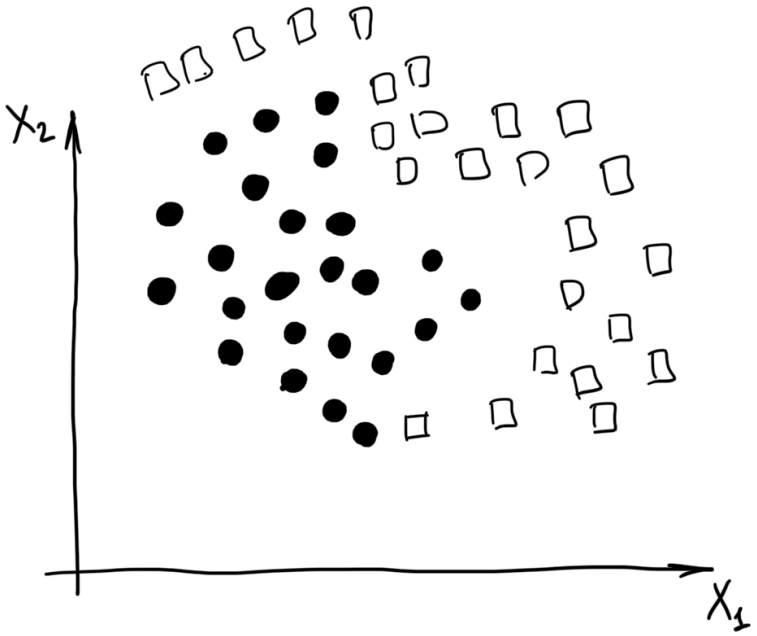
Чтобы финально записать систему уравнений, нам нужно освежить в памяти такое понятие как отступ. Отступом объекта x от границы классов есть величина $M = y(\omega^T x - b)$. Т.е., когда значение этого выражения отрицательно, это говорит о том, что алгоритм допустил ошибку. Когда $M \in (0, 1)$, то объект находится внутри зазора. Мы стремимся к тому, чтобы M всегда был больше или равен 1, т.е. объекты верно классифицированы.

Объединяя все уравнение, получим следующую систему уравнений:

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

Если внимательно посмотреть на получившуюся системы, то можно заметить, что данная система разрешима только в случае линейно разделимых классов. Такой вариант

называется Hard-margin SVM или SVM с жестким зазором. А как быть с линейно неразделимыми данными, как на примере?



В этом случае необходимо трансформировать систему, а именно, “разрешить” нашему алгоритму допускать ошибки. И конечно же, введем ограничение, чтобы этих ошибок не было очень много. Пусть дополнительные переменные ξ_i отвечают за величину ошибки на объекте x_i. В этом случае система будет выглядеть:

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

Будем считать количество ошибок алгоритма (когда M<0). Назовем это штрафом (*Penalty*). Тогда штраф для всех объектов будет равен сумме штрафов для каждого объекта:

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

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

Отлично, с самим штрафом разобрались, осталось сделать его внимательным к величине ошибки. Как мы и проговорили, с одной стороны, мы должны позволять алгоритму

совершать ошибки, но с другой - этих ошибок не должно быть много.

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

Добавим это выражение к функции штрафа и получаем классическую функцию потерь *SVM с мягким зазором* (*soft-margin SVM*) для одного объекта:

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

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

Q - функция потерь. Как раз этот функционал мы и будем минимизировать в процессе обучения с помощью градиентного спуска. Правила изменения весов будут выглядеть следующим образом:

$$\omega = \omega - \eta\triangledown Q$$

Преимущества и недостатки SVM

Преимущества

уменьшение количества ошибок за счет возможности управления шириной и положением разделяющей полосы;

хорошо работает с данными небольшого объема и пространством признаков большого размера;

в подобной постановке задача всегда имеет единственное решение, так как алгоритм сводится к решению задачи квадратичного программирования в выпуклой области.

Недостатки

неустойчивость к шуму: выбросы в обучающих данных становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости.

Подытожим

SVM - линейный метод классификации, который при обучении максимизирует зазор между классами, основываясь на опорных точках. Т.к. классы далеко не всегда линейны разделимы, то мы допускаем возможность ошибки, но при этом стараемся сделать так, чтобы эта ошибка была минимизирована. Несмотря на хитрую функцию потерь, этот алгоритм также обладает всеми недостатками всех линейных моделей: он плохо работает с выборками, где классы линейно неразделимы.