

Логистическая регрессия и SVM

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург

7 марта 2020 г.

Random facts:

- 7 марта 321 г. по указу императора Константина Великого воскресенье было объявлено выходным днём
- 7 марта 1530 г. папа римский отказал Генриху VIII в разводе
- 7 марта 1573 г. Иван Фёдоров основал во Львове типографию
- 7 марта 1824 г. 12-летний Ференц Лист впервые выступил с концертом в Париже
- 7 марта 1876 г. Александр Белл запатентовал телефон
- 7 марта 1933 г. Чарльз Дарроу в Атлантик-Сити изобрёл игру «Монополия»
- 7 марта 1981 г. на открытии Ленинградского рок-клуба прошёл первый концерт группы «Зоопарк»

Логистическая регрессия

- Итак, мы рассмотрели логистический сигмоид:

$$p(C_1 | \mathbf{x}) = \frac{p(\mathbf{x} | C_1)p(C_1)}{p(\mathbf{x} | C_1)p(C_1) + p(\mathbf{x} | C_2)p(C_2)} = \frac{1}{1 + e^{-a}} = \sigma(a),$$

$$\text{где } a = \ln \frac{p(\mathbf{x} | C_1)p(C_1)}{p(\mathbf{x} | C_2)p(C_2)}, \quad \sigma(a) = \frac{1}{1 + e^{-a}}.$$

- Вывели из него LDA и QDA, обучили их методом максимального правдоподобия, а потом отвлеклись на naïve Bayes.

- Возвращаемся к задаче классификации.
- Два класса, и апостериорное распределение – логистический сигмоид на линейной функции:

$$p(C_1 | \phi) = y(\phi) = \sigma(\mathbf{w}^\top \phi), \quad p(C_2 | \phi) = 1 - p(C_1 | \phi).$$

- *Логистическая регрессия* – это когда мы напрямую оптимизируем \mathbf{w} .

- Для датасета $\{\phi_n, t_n\}$, $t_n \in \{0, 1\}$, $\phi_n = \phi(\mathbf{x}_n)$:

$$p(\mathbf{t} | \mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}, \quad y_n = p(C_1 | \phi_n).$$

- Ищем параметры максимального правдоподобия, минимизируя $-\ln p(\mathbf{t} | \mathbf{w})$:

$$E(\mathbf{w}) = -\ln p(\mathbf{t} | \mathbf{w}) = -\sum_{n=1}^N [t_n \ln y_n + (1 - t_n) \ln(1 - y_n)].$$

- Пользуясь тем, что $\sigma' = \sigma(1 - \sigma)$, берём градиент (похоже на перцептрон):

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n.$$

- Если теперь сделать градиентный спуск, получим как раз разделяющую поверхность.
- Заметим, правда, что если данные действительно разделимы, то может получиться жуткий оверфиттинг: $\|\mathbf{w}\| \rightarrow \infty$, и сигмоид превращается в функцию Хевисайда. Надо регуляризовать.

- В логистической регрессии не получается замкнутого решения из-за сигмоида.
- Но функция $E(\mathbf{w})$ всё равно выпуклая, и можно воспользоваться методом Ньютона-Рапсона – на каждом шаге использовать локальную квадратичную аппроксимацию к функции ошибки:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mathbf{H}^{-1} \nabla E(\mathbf{w}),$$

где \mathbf{H} (Hessian) – матрица вторых производных $E(\mathbf{w})$.

- Замечание: давайте применим Ньютона-Рапсона к обычной линейной регрессии с квадратической ошибкой:

$$\begin{aligned}\nabla E(\mathbf{w}) &= \sum_{n=1}^N (\mathbf{w}^\top \phi_n - t_n) \phi_n = \mathbf{\Phi}^\top \mathbf{\Phi} \mathbf{w} - \mathbf{\Phi}^\top \mathbf{t}, \\ \nabla \nabla E(\mathbf{w}) &= \sum_{n=1}^N \phi_n \phi_n^\top = \mathbf{\Phi}^\top \mathbf{\Phi},\end{aligned}$$

и шаг оптимизации будет

$$\begin{aligned}\mathbf{w}^{\text{new}} &= \mathbf{w}^{\text{old}} - \left(\mathbf{\Phi}^\top \mathbf{\Phi}\right)^{-1} \left[\mathbf{\Phi}^\top \mathbf{\Phi} \mathbf{w}^{\text{old}} - \mathbf{\Phi}^\top \mathbf{t}\right] = \\ &= \left(\mathbf{\Phi}^\top \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^\top \mathbf{t},\end{aligned}$$

т.е. мы за один шаг придём к решению.

- Для логистической регрессии:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n = \mathbf{\Phi}^T (\mathbf{y} - \mathbf{t}),$$

$$\mathbf{H} = \nabla \nabla E(\mathbf{w}) = \sum_{n=1}^N y_n (1 - y_n) \phi_n \phi_n^T = \mathbf{\Phi}^T \mathbf{R} \mathbf{\Phi}$$

для диагональной матрицы \mathbf{R} с $R_{nn} = y_n(1 - y_n)$.

- Формула шага оптимизации:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \left(\Phi^T R \Phi \right)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) = \left(\Phi^T R \Phi \right)^{-1} \Phi^T R \mathbf{z},$$

где $\mathbf{z} = \Phi \mathbf{w}^{\text{old}} - R^{-1} (\mathbf{y} - \mathbf{t})$.

- Получилось как бы решение взвешенной задачи минимизации квадратического отклонения с матрицей весов R .
- Отсюда название: iterative reweighted least squares (IRLS).

- В случае нескольких классов

$$p(C_k | \phi) = y_k(\phi) = \frac{e^{a_k}}{\sum_j e^{a_j}} \text{ для } a_k = \mathbf{w}_k^\top \phi.$$

- Опять выпишем максимальное правдоподобие; во-первых,

$$\frac{\partial y_k}{\partial a_j} = y_k ([k = j] - y_j).$$

Несколько классов

- Теперь запишем правдоподобие – для схемы кодирования 1-of-K будет целевой вектор \mathbf{t}_n и правдоподобие

$$p(\mathbf{T} | \mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{n=1}^N \prod_{k=1}^K p(C_k | \phi_n)^{t_{nk}} = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}}$$

для $y_{nk} = y_k(\phi_n)$; берём логарифм:

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T} | \mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}, \text{ и}$$

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N (y_{nj} - t_{nj}) \phi_n.$$

- Оптимизировать опять можно по Ньютону-Рапсону; гессиан получится как

$$\nabla_{\mathbf{w}_k} \nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \dots, \mathbf{w}_K) = - \sum_{n=1}^N y_{nk} ([k=j] - y_{nj}) \phi_n \phi_n^T.$$

- А что если у нас другая форма сигмоида?
- Мы по-прежнему в той же постановке: два класса, $p(t = 1 | a) = f(a)$, $a = \mathbf{w}^\top \phi$, f – функция активации.
- Давайте установим функцию активации с порогом θ : для каждого ϕ_n , вычисляем $a_n = \mathbf{w}^\top \phi_n$, и

$$\begin{cases} t_n = 1, & \text{если } a_n \geq \theta, \\ t_n = 0, & \text{если } a_n < \theta. \end{cases}$$

- Если θ берётся по распределению $p(\theta)$, это соответствует

$$f(a) = \int_{-\infty}^a p(\theta) d\theta.$$

- Пусть, например, $p(\theta)$ – гауссиан с нулевым средним и единичной дисперсией. Тогда

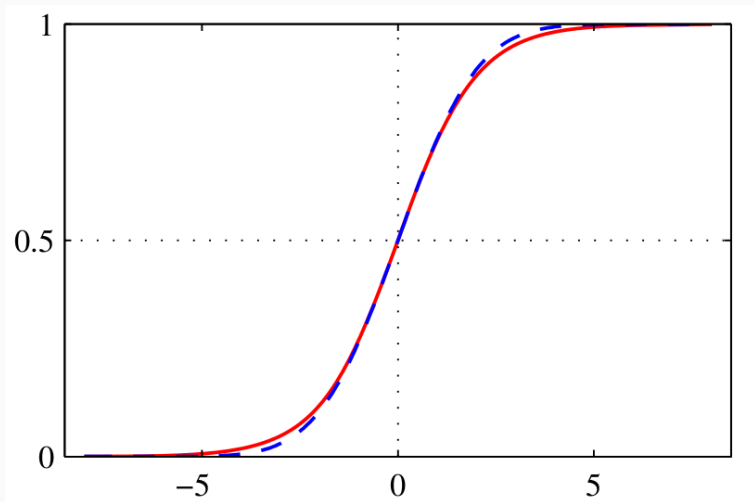
$$f(a) = \Phi(a) = \int_{-\infty}^a \mathcal{N}(\theta | 0, 1) d\theta.$$

- Это называется *пробит-функцией* (probit); неэлементарная, но тесно связана с

$$\operatorname{erf}(a) = \frac{2}{\sqrt{\pi}} \int_0^a e^{-\frac{\theta^2}{2}} d\theta :$$

$$\Phi(a) = \frac{1}{2} \left[1 + \frac{1}{\sqrt{2}} \operatorname{erf}(a) \right].$$

- Пробит-регрессия – это модель с пробит-функцией активации.



Лапласовская аппроксимация и
байесовская
логистическая регрессия

- Небольшое лирическое отступление: как приблизить сложное распределение простым?
- Например, как приблизить гауссианом возле максимума? (естественная задача)
- Рассмотрим пока распределение от одной непрерывной переменной $p(z) = \frac{1}{Z}f(z)$.

Лапласовская аппроксимация

- Первый шаг: найдём максимум z_0 .
- Второй шаг: разложим в ряд Тейлора

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2}A(z - z_0)^2, \text{ где } A = -\frac{d^2}{dz^2} \ln f(z) \Big|_{z=z_0} .$$

- Третий шаг: приблизим

$$f(z) \approx f(z_0)e^{-\frac{A}{2}(z-z_0)^2},$$

и после нормализации это будет как раз гауссиан.

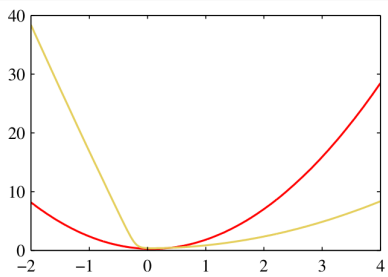
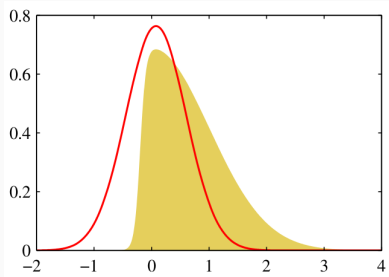
- Это можно обобщить на многомерное распределение $p(\mathbf{z}) = \frac{1}{Z}f(\mathbf{z})$:

$$f(\mathbf{z}) \approx f(\mathbf{z}_0)e^{-\frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^\top \mathbf{A}(\mathbf{z}-\mathbf{z}_0)},$$

$$\text{где } \mathbf{A} = -\nabla\nabla \ln f(\mathbf{z}) \big|_{\mathbf{z}=\mathbf{z}_0}.$$

Упражнение. Какая здесь будет нормировочная константа?

Лапласовская аппроксимация



Сравнение моделей по Лапласу

- Вооружившись лапласовской аппроксимацией, давайте применим её сначала к выбору моделей.
- Напомним: чтобы сравнить модели из множества $\{\mathcal{M}_i\}_{i=1}^L$, по тестовому набору D оценим апостериорное распределение

$$p(\mathcal{M}_i | D) \propto p(\mathcal{M}_i)p(D | \mathcal{M}_i).$$

- Если модель определена параметрически, то $p(D | \mathcal{M}_i) = \int p(D | \boldsymbol{\theta}, \mathcal{M}_i)p(\boldsymbol{\theta} | \mathcal{M}_i)d\boldsymbol{\theta}$.
- Это вероятность сгенерировать D , если выбирать параметры модели по её априорному распределению; знаменатель из теоремы Байеса:

$$p(\boldsymbol{\theta} | \mathcal{M}_i, D) = \frac{p(D | \boldsymbol{\theta}, \mathcal{M}_i)p(\boldsymbol{\theta} | \mathcal{M}_i)}{p(D | \mathcal{M}_i)}.$$

Сравнение моделей по Лапласу

- Мы раньше приближали фактически кусочно-постоянной функцией.
- Теперь давайте гауссианом приблизим; возьмём интеграл:

$$Z = \int f(\mathbf{z}) d\mathbf{z} \approx \int f(\mathbf{z}_0) e^{-\frac{1}{2}(\mathbf{z}-\mathbf{z}_0)^\top \mathbf{A}(\mathbf{z}-\mathbf{z}_0)} d\mathbf{z} = f(\mathbf{z}_0) \frac{(2\pi)^{M/2}}{|\mathbf{A}|^{1/2}}.$$

- А у нас $Z = p(D)$, $f(\boldsymbol{\theta}) = p(D | \boldsymbol{\theta})p(\boldsymbol{\theta})$.

Сравнение моделей по Лапласу

- Получаем

$$\ln p(D) \approx \ln p(D | \boldsymbol{\theta}_{\text{MAP}}) + \ln P(\boldsymbol{\theta}_{\text{MAP}}) + \frac{M}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{A}|.$$

- $\ln P(\boldsymbol{\theta}_{\text{MAP}}) + \frac{M}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{A}|$ – фактор Оккама.
- $\mathbf{A} = -\nabla\nabla \ln p(D | \boldsymbol{\theta}_{\text{MAP}})p(\boldsymbol{\theta}_{\text{MAP}}) = -\nabla\nabla \ln p(\boldsymbol{\theta}_{\text{MAP}} | D)$.

Сравнение моделей по Лапласу

- Получаем

$$\ln p(D) \approx \ln p(D | \boldsymbol{\theta}_{\text{MAP}}) + \ln P(\boldsymbol{\theta}_{\text{MAP}}) + \frac{M}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{A}|.$$

- Если гауссовское априорное распределение $p(\boldsymbol{\theta})$ достаточно широкое, и \mathbf{A} полного ранга, то можно грубо приблизить (докажите это!)

$$\ln p(D) \approx \ln p(D | \boldsymbol{\theta}_{\text{MAP}}) - \frac{1}{2} M \ln N,$$

где M – число параметров, N – число точек в D , а аддитивные константы мы опустили.

- Это *байесовский информационный критерий* (Bayesian information criterion, BIC), он же *критерий Шварца* (Schwarz criterion).

- Теперь давайте обработаем логистическую регрессию по-байесовски.
- Логистическую регрессию так просто не выпишешь, как линейную – точного ответа из произведения логистических сигмоидов не получается.
- Будем приближать по Лапласу.

Байесовская логистическая регрессия

- Априорное распределение выберем гауссовским:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0).$$

- Тогда апостериорное будет

$$\begin{aligned} p(\mathbf{w} \mid \mathbf{t}) &\propto p(\mathbf{w})p(\mathbf{t} \mid \mathbf{w}), \text{ и} \\ \ln p(\mathbf{w} \mid \mathbf{t}) &= -\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_0^{-1}(\mathbf{w} - \boldsymbol{\mu}_0) \\ &\quad + \sum_{n=1}^N [t_n \ln y_n + (1 - t_n) \ln(1 - y_n)] + \text{const}, \\ &\text{где } y_n = \sigma(\mathbf{w}^\top \boldsymbol{\phi}_n). \end{aligned}$$

- Чтобы приблизить, сначала находим максимум \mathbf{w}_{MAP} , а потом матрица ковариаций – это матрица вторых производных

$$\mathbf{\Sigma}_N = -\nabla\nabla \ln p(\mathbf{w} | \mathbf{t}) = \mathbf{\Sigma}_0^{-1} + \sum_{n=1}^N y_n(1 - y_n)\phi_n\phi_n^T.$$

- Наше приближение – это

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{w}_{\text{MAP}}, \mathbf{\Sigma}_N).$$

- Теперь можно описать байесовское предсказание:

$$p(C_1 | \phi, \mathbf{t}) = \int p(C_1 | \phi, \mathbf{w})p(\mathbf{w} | \mathbf{t})d\mathbf{w} \approx \int \sigma(\mathbf{w}^\top \phi)q(\mathbf{w})d\mathbf{w}.$$

- Заметим, что $\sigma(\mathbf{w}^\top \phi)$ зависит от \mathbf{w} только через его проекцию на ϕ .
- Обозначим $a = \mathbf{w}^\top \phi$:

$$\sigma(\mathbf{w}^\top \phi) = \int \delta(a - \mathbf{w}^\top \phi)\sigma(a)da.$$

- $\sigma(\mathbf{w}^\top \boldsymbol{\phi}) = \int \delta(a - \mathbf{w}^\top \boldsymbol{\phi}) \sigma(a) da$, а значит,

$$\int \sigma(\mathbf{w}^\top \boldsymbol{\phi}) q(\mathbf{w}) d\mathbf{w} = \int \sigma(a) p(a) da,$$

$$\text{где } p(a) = \int \delta(a - \mathbf{w}^\top \boldsymbol{\phi}) q(\mathbf{w}) d\mathbf{w}.$$

- $p(a)$ – это маргинализация гауссиана $q(\mathbf{w})$, где мы интегрируем по всему, что ортогонально $\boldsymbol{\phi}$.

Байесовская логистическая регрессия

- $p(a)$ – это маргинализация гауссиана $q(\mathbf{w})$, где мы интегрируем по всему, что ортогонально ϕ .
- Значит, $p(a)$ – тоже гауссиан; найдём его моменты:

$$\mu_a = \mathbb{E}[a] = \int a p(a) da = \int q(\mathbf{w}) \mathbf{w}^\top \phi d\mathbf{w} = \mathbf{w}_{\text{MAP}}^\top \phi,$$

$$\begin{aligned} \sigma_a^2 &= \int (a^2 - \mathbb{E}[a])^2 p(a) da = \\ &= \int q(\mathbf{w}) [(\mathbf{w}^\top \phi)^2 - (\mu_N^\top \phi)^2]^2 d\mathbf{w} = \phi^\top \Sigma_N \phi. \end{aligned}$$

- Итого получили, что

$$p(C_1 | \mathbf{t}) = \int \sigma(a) p(a) da = \int \sigma(a) \mathcal{N}(a | \mu_a, \sigma_a^2) da.$$

- $p(C_1 | \mathbf{t}) = \int \sigma(a) \mathcal{N}(a | \mu_a, \sigma_a^2) da.$
- Этот интеграл так просто не взять, потому что сигмоид сложный, но можно приблизить, если приблизить $\sigma(a)$ через пробит: $\sigma(a) \approx \Phi(\lambda a)$ для $\lambda = \sqrt{\pi/8}$.

Упражнение. Докажите, что для $\lambda = \sqrt{\pi/8}$ у σ и Φ одинаковый наклон в нуле.

- А если мы перейдём к пробит-функции, то её свёртка с гауссианом будет просто другим пробитом:

$$\int \Phi(\lambda a) \mathcal{N}(a \mid \mu, \sigma^2) da = \Phi\left(\frac{\mu}{\sqrt{\frac{1}{\lambda^2} + \sigma^2}}\right).$$

Упражнение. Докажите это.

- В итоге получается аппроксимация

$$\int \sigma(a) \mathcal{N}(a \mid \mu, \sigma^2) da \approx \sigma(\kappa(\sigma^2)\mu),$$

$$\text{где } \kappa(\sigma^2) = \frac{1}{\sqrt{1 + \frac{\pi}{8}\sigma^2}}.$$

Байесовская логистическая регрессия

- И теперь, собирая всё вместе, мы получили распределение предсказаний:

$$p(C_1 | \phi, \mathbf{t}) = \sigma(\kappa(\sigma_a^2)\mu_a), \text{ где}$$

$$\mu_a = \mathbf{w}_{\text{MAP}}^\top \phi,$$

$$\sigma_a^2 = \phi^\top \Sigma_N \phi,$$

$$\kappa(\sigma^2) = \frac{1}{\sqrt{1 + \frac{\pi}{8}\sigma^2}}.$$

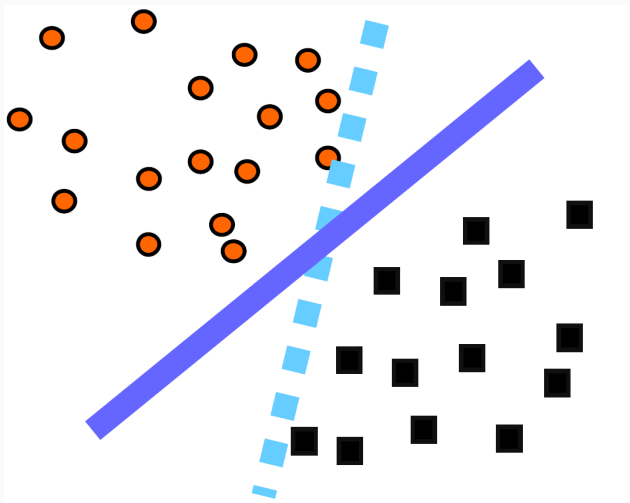
- Кстати, разделяющая поверхность $p(C_1 | \phi, \mathbf{t}) = \frac{1}{2}$ задаётся уравнением $\mu_a = 0$, и тут нет никакой разницы с просто использованием \mathbf{w}_{MAP} . Разница будет только для более сложных критериев.

SVM и задача линейной классификации

Постановка задачи

- Метод опорных векторов решает задачу классификации.
- Каждый элемент данных — точка в n -мерном пространстве \mathbb{R}^n .
- Формально: есть точки x_i , $i = 1..m$, у точек есть метки $y_i = \pm 1$.
- Мы интересуемся: можно ли разделить данные $(n - 1)$ -мерной гиперплоскостью, а также хотим найти эту гиперплоскость.
- Это всё?

- Нет, ещё хочется научиться разделять этой гиперплоскостью *как можно лучше*.
- То есть желательно, чтобы два разделённых класса лежали как можно дальше от гиперплоскости.
- Практическое соображение: тогда от небольших возмущений в гиперплоскости ничего не испортится.



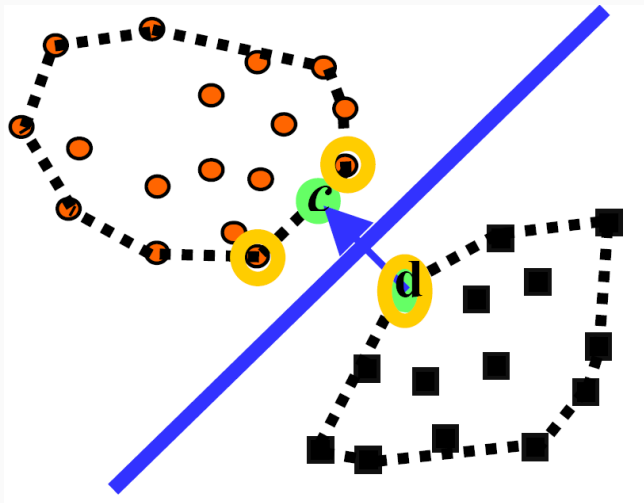
Выпуклые оболочки

- Один подход: найти две ближайшие точки в выпуклых оболочках данных, а затем провести разделяющую гиперплоскость через середину отрезка.
- Формально это превращается в задачу квадратичной оптимизации:

$$\min_{\alpha} \left\{ \|c - d\|^2, \text{ где } c = \sum_{y_i=1} \alpha_i x_i, d = \sum_{y_i=-1} \alpha_i x_i \right\}$$

при условии $\sum_{y_i=1} \alpha_i = \sum_{y_i=-1} \alpha_i = 1, \alpha_i \geq 0.$

- Эту задачу можно решать общими оптимизационными алгоритмами.



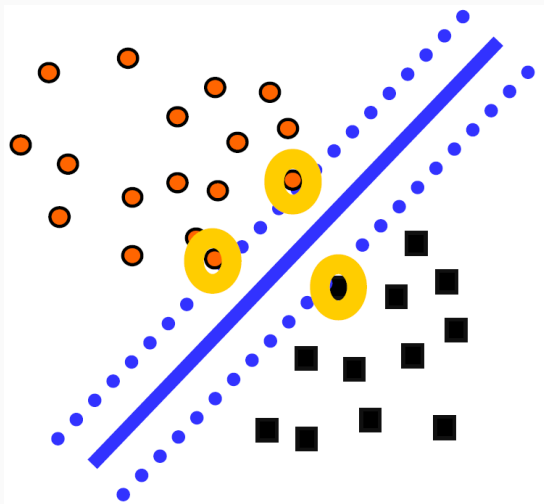
- Другой подход: максимизировать зазор (margin) между двумя параллельными опорными плоскостями, затем провести им параллельную на равных расстояниях от них.
- Гиперплоскость называется *опорной* для множества точек X , если все точки из X лежат под одну сторону от этой гиперплоскости.
- Формально: расстояние от точки до гиперплоскости $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = 0$ равно $\frac{|y(\mathbf{x})|}{\|\mathbf{w}\|}$.

Максимизация зазора

- Расстояние от точки до гиперплоскости $y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = 0$ равно $\frac{|y(\mathbf{x})|}{\|\mathbf{w}\|}$.
- Все точки классифицированы правильно: $t_n y(\mathbf{x}_n) > 0$ ($t_n \in \{-1, 1\}$).
- И мы хотим найти

$$\begin{aligned} \arg \max_{\mathbf{w}, w_0} \min_n \frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} &= \\ &= \arg \max_{\mathbf{w}, w_0} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n (\mathbf{w}^T \mathbf{x}_n + w_0)] \right\}. \end{aligned}$$

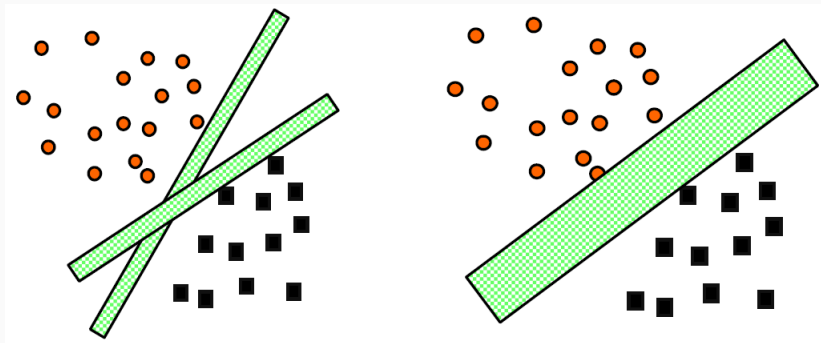
- $\arg \max_{\mathbf{w}, w_0} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n [t_n(\mathbf{w}^\top \mathbf{x}_n + w_0)] \right\}$. Сложно.
- Но если перенормировать \mathbf{w} , гиперплоскость не изменится.
- Давайте перенормируем так, чтобы $\min_n [t_n(\mathbf{w}^\top \mathbf{x}_n + w_0)] = 1$.



- Получается тоже задача квадратичного программирования:

$$\min_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\} \text{ при условии } t_n(\mathbf{w}^\top \mathbf{x}_n + w_0) \geq 1.$$

- Результаты получаются хорошие. Такой подход позволяет находить *устойчивые* решения, что во многом решает проблемы с оверфиттингом и позволяет лучше предсказывать дальнейшую классификацию.
- В каком-то смысле в решениях с «толстыми» гиперплоскостями между данными содержится больше информации, чем в «тонких», потому что «толстых» меньше.
- Это всё можно сформулировать и доказать (позже).



- Напомним, что такое дуальные задачи.
- Прямая задача оптимизации:

$$\min \{f(x)\} \text{ при условии } h(x) = 0, g(x) \leq 0, x \in X.$$

- Для дуальной задачи вводим параметры λ , соответствующие равенствам, и μ , соответствующие неравенствам.

- Прямая задача оптимизации:

$$\min \{f(x)\} \text{ при условии } h(x) = 0, g(x) \leq 0, x \in X.$$

- Дуальная задача оптимизации:

$$\min \{\phi(\lambda, \mu)\} \text{ при условии } \mu \geq 0,$$

$$\text{где } \phi(\lambda, \mu) = \inf_{x \in X} \{f(x) + \lambda^\top h(x) + \mu^\top g(x)\}.$$

- Тогда, если $(\bar{\lambda}, \bar{\mu})$ – допустимое решение дуальной задачи, а \bar{x} – допустимое решение прямой, то

$$\begin{aligned}\phi(\bar{\lambda}, \bar{\mu}) &= \inf_{x \in X} \{f(x) + \bar{\lambda}^\top h(x) + \bar{\mu}^\top g(x)\} \leq \\ &\leq f(\bar{x}) + \bar{\lambda}^\top h(\bar{x}) + \bar{\mu}^\top g(\bar{x}) \leq f(\bar{x}).\end{aligned}$$

- Это называется *слабой дуальностью* (только \leq), но во многих случаях достигается и равенство.

- Для линейного программирования прямая задача:

$$\min c^T x \text{ при условии } Ax = b, x \in X = \{x \leq 0\}.$$

- Тогда дуальная задача получается так:

$$\begin{aligned} \phi(\lambda) &= \inf_{x \geq 0} \{c^T x + \lambda^T (b - Ax)\} = \\ &= \lambda^T b + \inf_{x \geq 0} \{(c^T - \lambda^T A)x\} = \\ &= \begin{cases} \lambda^T b, & \text{если } c^T - \lambda^T A \geq 0, \\ -\infty & \text{в противном случае.} \end{cases} \end{aligned}$$

- Для линейного программирования прямая задача:

$$\min \{c^T x\} \text{ при условии } Ax = b, x \in X = \{x \leq 0\}.$$

- Дуальная задача:

$$\max \{b^T \lambda\} \text{ при условии } A^T \lambda \leq c, \lambda \text{ не ограничены.}$$

- Для квадратичного программирования прямая задача:

$$\min \left\{ \frac{1}{2}x^T Qx + c^T x \right\} \text{ при условии } Ax \leq b,$$

где Q – положительно полуопределённая матрица (т.е. $x^T Qx \geq 0$ всегда).

- Дуальная задача (проверьте):

$$\max \left\{ \frac{1}{2}\mu^T D\mu + \mu^T d - \frac{1}{2}c^T Q^{-1}c \right\} \text{ при условии } c \geq 0,$$

где $D = -AQ^{-1}A^T$ (отрицательно определённая матрица),
 $d = -b - AQ^{-1}c$.

Дуальная задача к SVM

- В случае SVM надо ввести множители Лагранжа:

$$L(\mathbf{w}, w_0, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_n \alpha_n [t_n(\mathbf{w}^T \mathbf{x}_n + w_0) - 1], \quad \alpha_n \geq 0.$$

- Берём производные по \mathbf{w} и w_0 , приравниваем нулю, получаем

$$\mathbf{w} = \sum_n \alpha_n t_n \mathbf{x}_n,$$

$$0 = \sum_n \alpha_n t_n.$$

Дуальная задача к SVM

- Подставляя в $L(\mathbf{w}, w_0, \boldsymbol{\alpha})$, получим

$$L(\boldsymbol{\alpha}) = \sum_n \alpha_n - \frac{1}{2} \sum_n \sum_m \alpha_n \alpha_m t_n t_m (\mathbf{x}_n^\top \mathbf{x}_m)$$

при условии $\alpha_n \geq 0, \sum_n \alpha_n t_n = 0$.

- Это дуальная задача, которая обычно в SVM и используется.

- А для предсказания потом надо посмотреть на знак $y(\mathbf{x})$:

$$y(\mathbf{x}) = \sum_{n=1}^N \alpha_n t_n \mathbf{x}^T \mathbf{x}_n + w_0.$$

- Получилось, что предсказания зависят от всех точек \mathbf{x}_n ...

- ...но нет. :) Условия ККТ (Karush–Kuhn–Tucker):

$$\alpha_n \geq 0,$$

$$t_n y(\mathbf{x}_n) - 1 \geq 0,$$

$$\alpha_n (t_n y(\mathbf{x}_n) - 1) = 0.$$

- Т.е. реально предсказание зависит от небольшого числа *опорных* векторов, для которых $t_n y(\mathbf{x}_n) = 1$ (они находятся собственно на границе разделяющей поверхности).

- Все эти методы работают, когда данные действительно линейно делимы.
- А что делать, когда их всё-таки немножко не получается разделить?
- Первый вопрос: что делать для первого метода, метода выпуклых оболочек?

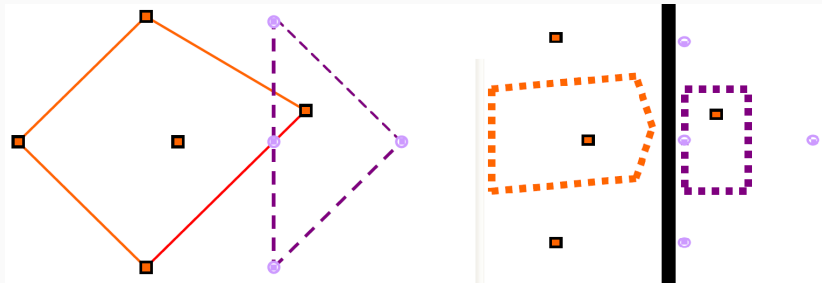
Редуцированные выпуклые оболочки

- Вместо обычных выпуклых оболочек можно рассматривать *редуцированные* (reduced), у которых коэффициенты ограничены не 1, а сильнее:

$$c = \sum_{y_i=1} \alpha_i X_i, \quad 0 \leq \alpha_i \leq D.$$

- Тогда для достаточно малых D редуцированные выпуклые оболочки не будут пересекаться.
- И мы будем искать оптимальную гиперплоскость между редуцированными выпуклыми оболочками.

Пример



Для метода опорных векторов

- Естественно, для метода опорных векторов тоже надо что-то изменить. Что?

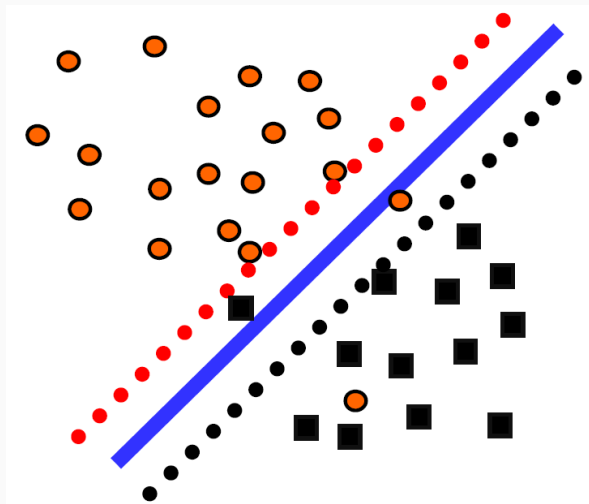
Для метода опорных векторов

- Естественно, для метода опорных векторов тоже надо что-то изменить. Что?
- Мы просто добавим в оптимизируемую функцию неотрицательную ошибку (slack):

$$\min_{\mathbf{w}, w_0} \left\{ \|\mathbf{w}\|^2 + C \sum_{i=1}^m z_i \right\}$$

при условии $t_i(\mathbf{w} \cdot \mathbf{x}_i - w_0) + z_i \geq 1$.

- Это прямая задача...



Дуальная переформулировка

- ...а вот дуальная:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^m t_i t_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^m \alpha_i, \right. \\ \left. \text{где } \sum_{i=1}^m t_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

- Эта формулировка чаще всего используется в теории SVM.
- Единственное отличие от линейно разделимого случая – верхняя граница C на α_j , т.е. на влияние каждой точки.

- Метод опорных векторов отлично подходит для линейной классификации.
- Решая задачу квадратичного программирования, мы получаем параметры оптимальной гиперплоскости.
- Точно так же, как и в дуальном случае, если бы мы просто искали середину между выпуклыми оболочками.

SVM и эмпирический риск

- Ещё один взгляд на SVM — какая вообще задача у любой классификации?
- Мы хотим минимизировать эмпирический риск, то есть число неправильных ответов:

$$\sum_n [y_i \neq t_i] \rightarrow \min_{\mathbf{w}}$$

- И если функция линейная с параметрами \mathbf{w} , w_0 , то это эквивалентно

$$\sum_n [t_i (\mathbf{x}_n^T \mathbf{w} - w_0) < 0] \rightarrow \min_{\mathbf{w}}$$

- Величину $M_i = \mathbf{x}_n^T \mathbf{w} - w_0$ назовём *отступом* (margin).
- Оптимизировать напрямую сложно...

- ...поэтому заменим на оценку сверху:

$$\sum_n [M_i < 0] \leq \sum_n (1 - M_i) \rightarrow \min_{\mathbf{w}} .$$

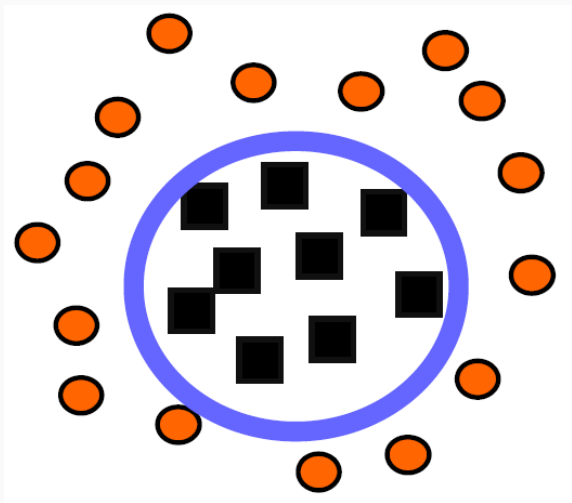
- А потом ещё добавим регуляризатор для стабильности:

$$\sum_n [M_i < 0] \leq \sum_n (1 - M_i) + \frac{1}{2C} \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w}} .$$

- И это снова получилась задача SVM!

SVM и разделение нелинейными функциями

- Часто бывает нужно разделять данные не только линейными функциями.
- Что делать в таком случае?



- Часто бывает нужно разделять данные не только линейными функциями.
- Классический метод: развернуть нелинейную классификацию в пространство большей размерности (feature space), а там запустить линейный классификатор.
- Для этого просто нужно для каждого монома нужной степени ввести новую переменную.

Пример

- Чтобы в двумерном пространстве $[r, s]$ решить задачу классификации квадратичной функцией, надо перейти в пятимерное пространство:

$$[r, s] \longrightarrow [r, s, rs, r^2, s^2].$$

- Или формальнее; определим $\theta : \mathbb{R}^2 \rightarrow \mathbb{R}^5$:
 $\theta(r, s) = (r, s, rs, r^2, s^2)$. Вектор в \mathbb{R}^5 теперь соответствует квадратичной кривой общего положения в \mathbb{R}^2 , а функция классификации выглядит как

$$f(\mathbf{x}) = \text{sign}(\theta(\mathbf{w}) \cdot \theta(\mathbf{x}) - b).$$

- Если решить задачу линейного разделения в этом новом пространстве, тем самым решится задача квадратичного разделения в исходном.

- Во-первых, количество переменных растёт экспоненциально.
- Во-вторых, по большому счёту теряются преимущества того, что гиперплоскость именно оптимальная; например, оверфиттинг опять становится проблемой.
- Важное замечание: *концептуально* мы задачу уже решили. Остались *технические* сложности: как обращаться с гигантской размерностью. Но в них-то всё и дело.

Основная идея и схема работы SVM

- Тривиальная схема алгоритма классификации такова:
 - входной вектор x трансформируется во входной вектор в feature space (большой размерности);
 - в этом большом пространстве мы вычисляем опорные векторы, решаем задачу разделения;
 - потом по этой задаче классифицируем входной вектор.
- Это нереально, потому что пространство слишком большой размерности.

Основная идея и схема работы SVM

- Оказывается, кое-какие шаги здесь можно переставить. Вот так:
 - опорные векторы вычисляются в исходном пространстве малой размерности;
 - там же они перемножаются (сейчас увидим, что это значит);
 - и только потом мы делаем нелинейную трансформацию того, что получится;
 - потом по этой задаче классифицируем входной вектор.
- Осталось теперь объяснить, что всё это значит. :)

- Напомним, что наша задача поставлена следующим образом:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^m y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^m \alpha_i, \right.$$

где $\left. \sum_{i=1}^m y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$

Постановка задачи

- Мы теперь хотим ввести некое отображение $\theta : \mathbb{R}^n \rightarrow \mathbb{R}^N$, $N > n$. Получится:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^m y_i y_j \alpha_i \alpha_j (\theta(\mathbf{x}_i) \cdot \theta(\mathbf{x}_j)) - \sum_{i=1}^m \alpha_i, \right. \\ \left. \text{где } \sum_{i=1}^m y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

- Придётся немножко вспомнить (или изучить) функциональный анализ.
- Мы хотим обобщить понятие *скалярного произведения*; давайте введём новую функцию, которая (минуя трансформацию) будет сразу вычислять скалярное произведение векторов в feature space:

$$k(\mathbf{u}, \mathbf{v}) := \theta(\mathbf{u}) \cdot \theta(\mathbf{v}).$$

- Первый результат: любая симметрическая функция $k(\mathbf{u}, \mathbf{v}) \in L_2$ представляется в виде

$$k(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{\infty} \lambda_i \theta_i(\mathbf{u}) \cdot \theta_i(\mathbf{v}),$$

где $\lambda_i \in \mathbb{R}$ — собственные числа, а θ_i — собственные векторы интегрального оператора с ядром k , т.е.

$$\int k(\mathbf{u}, \mathbf{v}) \theta_i(\mathbf{u}) d\mathbf{u} = \lambda_i \theta_i(\mathbf{v}).$$

Теория Гильберта–Шмидта–Мерсера

- Чтобы k задавало скалярное произведение, достаточно, чтобы все собственные числа были положительными. А собственные числа положительны тогда и только тогда, когда (*теорема Мерсера*)

$$\int \int k(\mathbf{u}, \mathbf{v}) g(\mathbf{u}) g(\mathbf{v}) d\mathbf{u} d\mathbf{v} > 0$$

для всех g таких, что $\int g^2(\mathbf{u}) d\mathbf{u} < \infty$.

- Вот, собственно и всё. Теперь мы можем вместо подсчёта $\theta(\mathbf{u}) \cdot \theta(\mathbf{v})$ в задаче квадратичного программирования просто использовать подходящее ядро $k(\mathbf{u}, \mathbf{v})$.

Теория Гильберта–Шмидта–Мерсера

- Итого задача наша выглядит так:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^m y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^m \alpha_i, \right. \\ \left. \text{где } \sum_{i=1}^m y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

- Просто меняя ядро k , мы можем вычислять самые разнообразные разделяющие поверхности.
- Условия на то, чтобы k была подходящим ядром, задаются теоремой Мерсера.

- Рассмотрим ядро

$$k(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^2.$$

- Какое пространство ему соответствует?

- После выкладок получается:

$$k(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^2 = (u_1^2, u_2^2, \sqrt{2}u_1u_2) \cdot (v_1^2, v_2^2, \sqrt{2}v_1v_2).$$

- Иначе говоря, линейная поверхность в новом пространстве соответствует квадратичной поверхности в исходном (эллипс, например).

- Естественное обобщение: ядро $k(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$ задаёт пространство, оси которого соответствуют всем *однородным* мономам степени d .
- А как сделать пространство, соответствующее произвольной полиномиальной поверхности, не обязательно однородной?

- Поверхность, описываемая полиномом степени d :

$$k(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d.$$

- Тогда линейная разделимость в feature space в точности соответствует полиномиальной разделимости в базовом пространстве.

- Нормальное распределение (radial basis function):

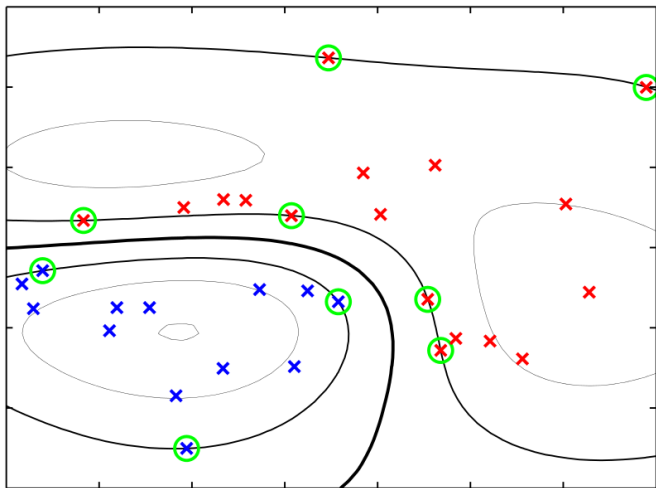
$$k(\mathbf{u}, \mathbf{v}) = e^{-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma}}.$$

- Двухуровневая нейронная сеть:

$$k(\mathbf{u}, \mathbf{v}) = o(\eta \mathbf{u} \cdot \mathbf{v} + c),$$

где o — сигмоид.

Пример



- Вот какой получается в итоге алгоритм.
 1. Выбрать параметр C , от которого зависит акцент на минимизации ошибки или на максимизации зазора.
 2. Выбрать ядро и параметры ядра, которые у него, возможно, есть.
 3. Решить задачу квадратичного программирования.
 4. По полученным значениям опорных векторов определить w_0 (как именно?).
 5. Новые точки классифицировать как

$$f(\mathbf{x}) = \text{sign}\left(\sum_i y_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) - w_0\right).$$

- Другой вариант для неразделимых данных – ν -SVM [Schölkopf et al., 2000].
- Максимизируем

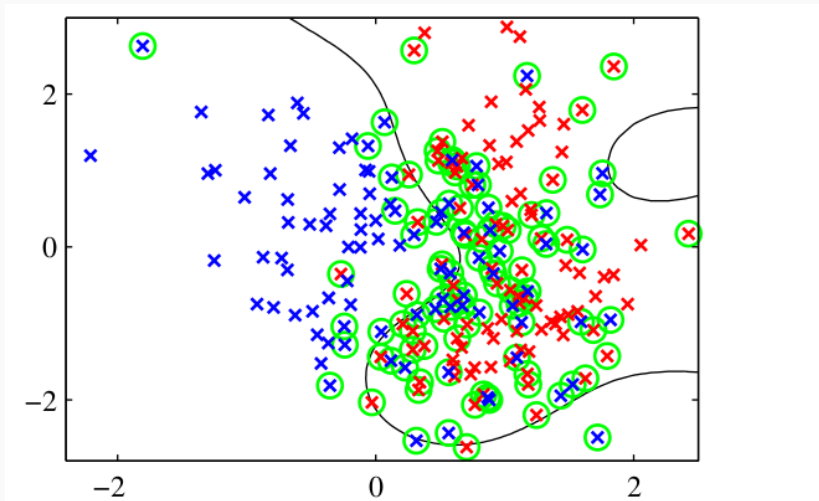
$$L(\mathbf{a}) = -\frac{1}{2} \sum_n \sum_m a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

с ограничениями

$$0 \leq a_n \leq \frac{1}{N}, \quad \sum_n a_n t_n = 0, \quad \sum_n a_n \geq \nu.$$

- Параметр ν можно интерпретировать как верхнюю границу на долю ошибок.

SVM для классификации



Связь с логистической регрессией

- В случае SVM с возможными ошибками мы минимизируем

$$C \sum_{n=1}^N \xi_n + \frac{1}{2} \|\mathbf{w}\|^2.$$

- Для точек с правильной стороны $\xi_n = 0$, с неправильной – $\xi_n = 1 - y_n t_n$.
- Так что можно записать *hinge error function* $E_{SV}(y_n t_n) = [1 - y_n t_n]_+$ и переписать как задачу с регуляризацией

$$\sum_{n=1}^N E_{SV}(y_n t_n) + \lambda \|\mathbf{w}\|^2.$$

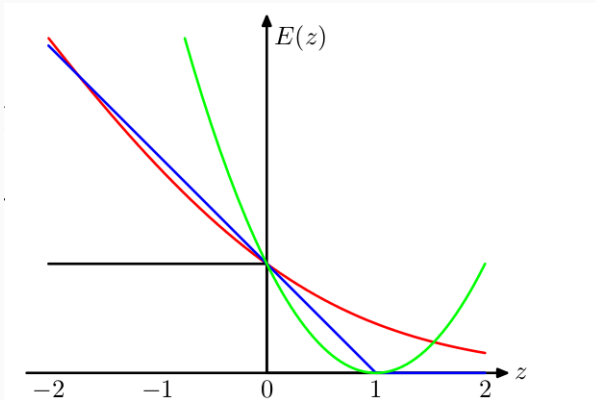
- Вспомним логистическую регрессию и переформулируем её для целевой переменной $t \in \{-1, 1\}$: $p(t = 1 | y) = \sigma(y)$, значит, $p(t = -1 | y) = 1 - \sigma(y) = \sigma(-y)$, и $p(t | y) = \sigma(yt)$.
- И логистическая регрессия – это минимизация

$$\sum_{n=1}^N E_{LR}(y_n t_n) + \lambda \|\mathbf{w}\|^2,$$

где $E_{LR}(y_n t_n) = \ln(1 + e^{-y_n t_n})$.

Связь с логистической регрессией

- График hinge error function, вместе с функцией ошибки для логистической регрессии:



- Как обобщить SVM на несколько классов? Варианты (без подробностей):
 1. обучить одну против всех и классифицировать $y(\mathbf{x}) = \max_k y_k(\mathbf{x})$ (нехорошо, потому что задача становится несбалансированной, и $y_k(\mathbf{x})$ на самом деле несравнимы);
 2. можно сформулировать единую функцию для всех K SVM одновременно, но обучение становится гораздо медленнее;
 3. можно обучить попарно $K(K - 1)/2$ классификаторов, а потом считать голоса – кто победит;
 4. DAGSVM: организуем попарные классификаторы в граф и будем идти по графу, для классификации выбирая очередной вопрос;
 5. есть даже методы, основанные на кодах, исправляющих ошибки.

- SVM также можно использовать с *одним* классом.
- Как и зачем?

- SVM также можно использовать с *одним* классом.
- Как и зачем?
- Можно при помощи SVM очертить границу области высокой плотности.
- Тем самым найдём выбросы данных (outliers).
- Задача будет такая: найти наименьшую поверхность (сферу, например), которая содержит все точки, кроме доли ν .

- SVM можно использовать для регрессии, сохраняя свойство разреженности (т.е. то, что SVM зависит только от опорных векторов).
- В обычной линейной регрессии мы минимизировали

$$\frac{1}{2} \sum_{n=1}^N (y_n - t_n)^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

- В SVM мы сделаем так: если мы попадаем в ϵ -окрестность предсказания, то ошибки, будем считать, совсем нет.

- ϵ -insensitive error function:

$$E_{\epsilon}(y(\mathbf{x}) - t) = \begin{cases} 0, & |y(\mathbf{x}) - t| < \epsilon, \\ |y(\mathbf{x}) - t| - \epsilon & \text{иначе.} \end{cases}$$

- И задача теперь выглядит как минимизация

$$C \sum_{n=1}^N E_{\epsilon}(y(\mathbf{x}_n) - t_n) + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

SVM для регрессии

- Чтобы переформулировать, нужны по две slack переменные, для обеих сторон «трубки»:

$$y(\mathbf{x}_n) - \epsilon \leq t_n \leq y(\mathbf{x}_n) + \epsilon$$

превращается в

$$t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n,$$

$$t_n \geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n,$$

и мы оптимизируем

$$C \sum_{n=1}^N E_{\epsilon}(\xi_n + \hat{\xi}_n) + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

- Если же теперь пересчитать дуальную задачу, то получится

$$L(\mathbf{a}, \hat{\mathbf{a}}) = -\frac{1}{2} \sum_n \sum_m (a_n - \hat{a}_n)(a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) - \\ - \epsilon \sum_{n=1}^n (a_n + \hat{a}_n) + \sum_{n=1}^N (a_n - \hat{a}_n) t_n,$$

и мы её минимизируем по a_n, \hat{a}_n с условиями

$$0 \leq a_n \leq C,$$

$$0 \leq \hat{a}_n \leq C,$$

$$\sum_{n=1}^N (a_n - \hat{a}_n) = 0.$$

- Когда решим эту задачу, сможем предсказывать новые значения как

$$y(\mathbf{x}) = \sum_{n=1}^N (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b,$$

где b можно найти как

$$\begin{aligned} b &= t_n - \epsilon - \mathbf{w}^\top \phi(\mathbf{x}_n) = \\ &= t_n - \epsilon - \sum_{m=1}^N (a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m). \end{aligned}$$

- А условия ККТ превращаются в

$$\begin{aligned}a_n (\epsilon + \xi_n + y(\mathbf{x}_n) - t_n) &= 0, \\ \hat{a}_n (\epsilon + \hat{\xi}_n - y(\mathbf{x}_n) + t_n) &= 0, \\ (C - a_n)\xi_n &= 0, \\ (C - \hat{a}_n)\hat{\xi}_n &= 0.\end{aligned}$$

- Отсюда очевидно, что либо a_n , либо \hat{a}_n всегда равны 0, и хотя бы один из них не равен, только если точка лежит на или за границей «трубки».
- Опять получили решение, зависящее только от «опорных векторов».

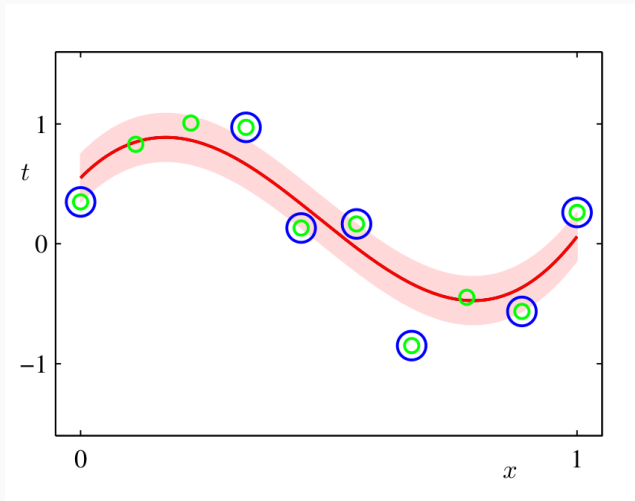
- Но снова можно переформулировать в виде ν -SVM, в котором параметр более интуитивно ясен: вместо ширины трубки ϵ рассмотрим ν – долю точек, лежащих вне трубки; тогда минимизировать надо

$$L(\mathbf{a}) = -\frac{1}{2} \sum_n \sum_m (a_n - \hat{a}_n) (a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) + \sum_{n=1}^N (a_n - \hat{a}_n) t_n$$

при условиях

$$\begin{aligned} 0 \leq a_n \leq \frac{C}{N}, & \quad \sum_{n=1}^N (a_n - \hat{a}_n) = 0, \\ 0 \leq \hat{a}_n \leq \frac{C}{N}, & \quad \sum_{n=1}^N (a_n + \hat{a}_n) \leq \nu C. \end{aligned}$$

SVM для регрессии



- На практике:
 - маленький C – гладкая разделяющая поверхность, мало опорных векторов;
 - большой C – сложная разделяющая поверхность, много опорных векторов.
- Для RBF ядра:
 - маленькое γ – опорные векторы влияют далеко, модель более простая;
 - большое γ – опорные векторы влияют только непосредственно рядом, модель более сложная.

Спасибо!

Спасибо за внимание!