



Лекция 11: Метод опорных векторов

Линейный бинарный классификатор на основе разделяющей гиперплоскости

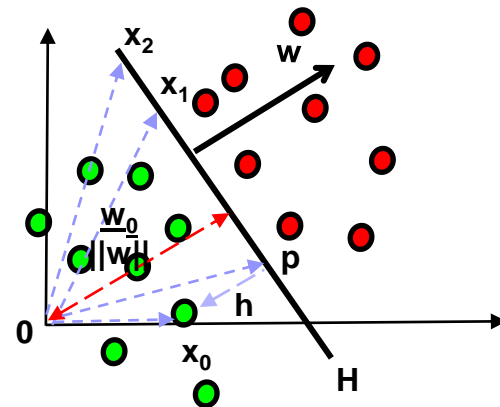
■ Основные определения и свойства:

- Отклик $Y = \{-1, +1\}$
- Дискр. ф-ция $g(x) = g_+(x) - g_-(x)$
- Отступ $M(x, y) = yg(x)$
- Линейность $g(x) = \langle w, x \rangle + w_0$
- Граница – гиперплоскость $H = \{x | \langle w, x \rangle + w_0 = 0\}$ определяется нормалью $w / \|w\|$ и смещением $w_0 / \|w\|$.
- То, что w – ортогонально H , доказывается из определения линейного $g(x)$ и равенства $g(x) = 0$ на границе: пусть $x_1, x_2 \in H \Rightarrow \langle w, x_1 \rangle + w_0 = 0$, $\langle w, x_2 \rangle + w_0 = 0 \Rightarrow \langle w, x_2 - x_1 \rangle = 0 \Rightarrow w$ ортогонально любым $x_1, x_2 \in H$
- Знаковое расстояние от точки x_0 до границы $d(x_0, H) = g(x_0) / \|w\|$ (подстановка в нормализованное уравнение гиперплоскости).

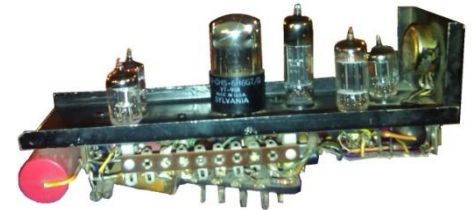
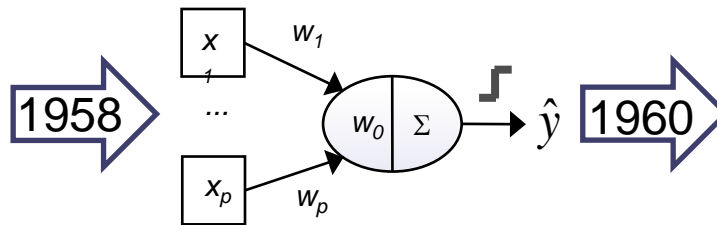
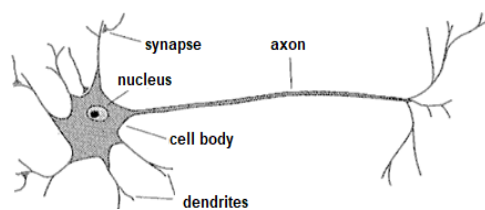
Пусть $x_0 = p + h$, $p \in H$ – проекция x_0 на H , h – ортогональное дополнение, тогда $h = d \frac{w}{\|w\|}$, $x_0 = p + d \frac{w}{\|w\|}$ домножаем скалярно на w прибавляем w_0 ,

$$\text{получаем: } \langle w, x_0 \rangle + w_0 = \langle w, p \rangle + w_0 + d \frac{\langle w, w \rangle}{\|w\|} \Rightarrow d = \frac{\langle w, x_0 \rangle + w_0}{\|w\|}$$

- Прогноз $a(x) = \text{sign}(g(x))$ – с какой стороны от H , расстояния от центра координат до H равно $w_0 / \|w\|$



Персептрон Розенблатта



- Модель - разделяющая гиперплоскость:

- Функция потерь $L_{perc}(M) = -[M]_+$,
- Обучение - SGD, доказана сходимость за конечное число шагов
- Для «ошибок» (примеров не с той стороны гиперплоскости):

$$\begin{pmatrix} w^{(t)} \\ w_0^{(t)} \end{pmatrix} + \eta \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix} \rightarrow \begin{pmatrix} w^{(t+1)} \\ w_0^{(t+1)} \end{pmatrix}$$

- Недостатки (их устранение - достоинства SVM):

- Несколько возможных решений при линейной разделимости классов (зависит от начального приближения)
- Не сходится при линейной неразделимости классов, а при линейной разделимости долго сходится (много шагов)

Обучение линейного классификатора

- «Пороговая» (персептрон) функция потерь $L_{perc}(M) = -[M]_+$
 - кусочно-постоянная \Rightarrow имеет нулевые градиенты
- Можно ограничить ее сверху другой гладкой функцией потерь и искать решение задачи оптимизации с регуляризацией:

- **Логистическая:**

$$L_{log}(M) = \log_2(1 + e^{-M})$$

- **Квадратичная:**

$$L_{sq}(M) = (1 - M)^2$$

- **Экспоненциальная:**

$$L_{exp}(M) = e^{-M}$$

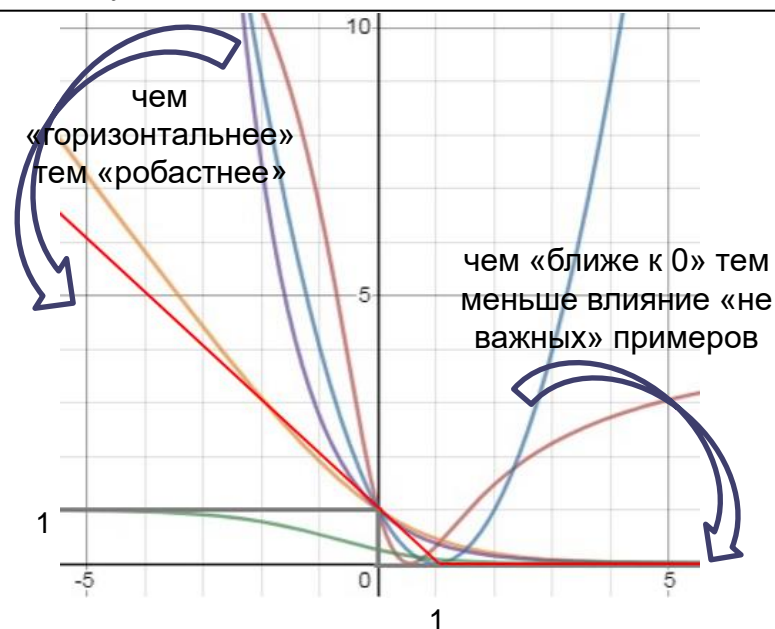
- **Тангесовая:**

$$L_{tng}(M) = (2 \arctan(M) - 1)^2$$

- **Hinge («шарнир»):**

$$L_{hinge}(M) = -[1 - M]_+$$

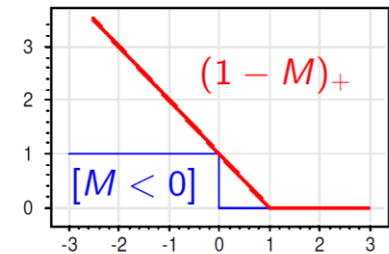
$$\min_w \frac{1}{l} \sum_i L_*(y_i(\langle x_i, w \rangle + w_0)) + \gamma L_p(w)$$



Аппроксимация Hinge функцией потерь с L_2 регуляризацией

- Ограничим сверху эмпирический риск персептрона L_2 - регуляризованным эмпирическим риском с с Hinge функцией потерь:

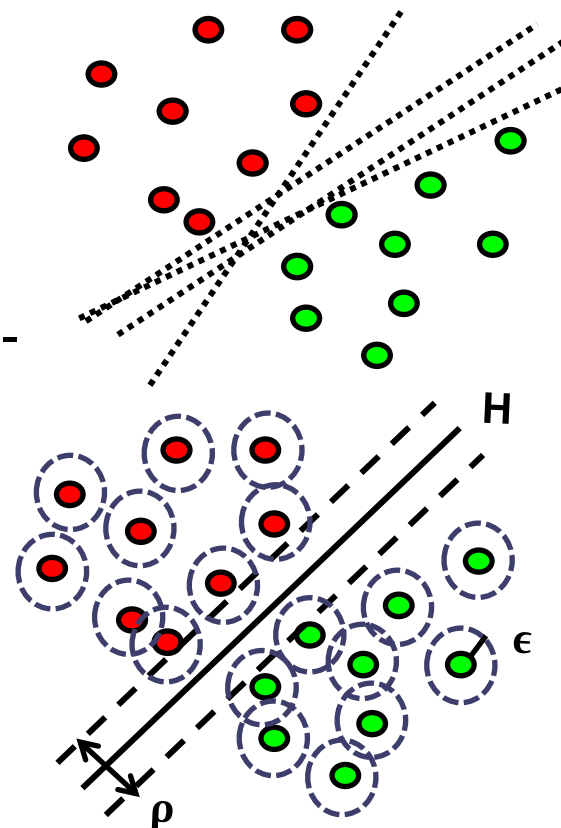
$$\begin{aligned} Q_{perc}(w, w_0) &= \sum_{i=1}^l [M_i(w, w_0) < 0] \leq \\ &\leq Q_{hinge}(w, w_0) = \sum_{i=1}^l (1 - M_i(w, w_0))_+ + \gamma \|w\|^2 \\ Q_{hinge} &\rightarrow \min_{w, w_0} \Rightarrow Q_{perc} \rightarrow \min_{w, w_0} \end{aligned}$$



- Первое слагаемое:
 - линейно штрафует за приближение к границе классов с «правильной стороны» ближе чем 1
 - линейно штрафует за удаление от границы с «неправильной стороны»
- Второе слагаемое:
 - штрафует за сложность, не давая переобучаться
 - контролирует стабильность при мультколлинеарности

Оптимальная разделяющая гиперплоскость в случае линейно разделимых классов

- В случае линейно разделимости классов:
 - можно провести бесконечно много разделяющих гиперплоскостей.
 - **Какая из них лучше?**
- Определим ширину разделяющей полосы - зазор (марджин) для множества точек как минимум по всем:
$$\rho = \min_{1 \leq i \leq l} M(x_i, y_i) = \min_{1 \leq i \leq l} y_i g(x_i)$$
- Т.к. есть случайная составляющая (шум):
 - наблюдения могут лежать в некоторой окрестности неизвестного радиуса ϵ
 - значит чем больше отступ ρ , тем меньше вероятность, что окрестность точек рядом с границей пересечет ее
- Вывод – нужно **максимизировать зазор**



Максимизация отступа в случае линейно разделимых классов

- Каноническое уравнение гиперплоскости:

- уравнение H определено с точностью до множителя, надо зафиксировать (с точностью до знака)
- нормируем параметры так, чтобы расстояние $d(x, H) = g(x)/||w||$ от границы до ближайшего наблюдения каждого класса было равно 1
- Это приводит к условиям: если $y_i = 1 \Rightarrow \langle w, x_i \rangle + w_0 \geq 1$, а для $y_i = -1 \Rightarrow \langle w, x_i \rangle + w_0 \leq -1$ и в общем виде $\forall i: y_i(\langle w, x_i \rangle + w_0) \geq 1$

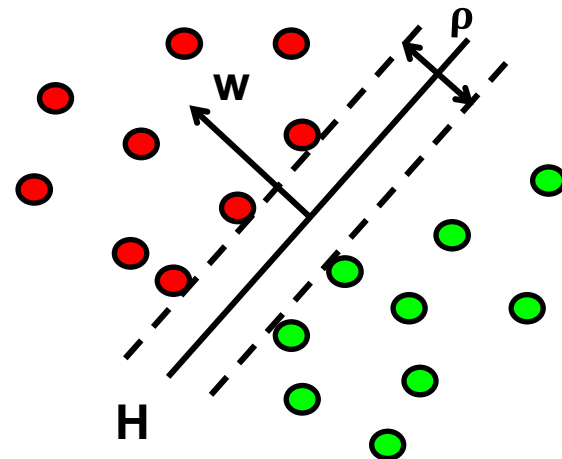
- Ширина разделяющей полосы (зазора между классами):

$$\rho = \frac{2}{||w||} \rightarrow \max_w$$

- Получаем задачу условной оптимизации:

$$\begin{cases} \min_w \frac{1}{2} ||w||^2 \\ \forall i: y_i(\langle w, x_i \rangle + w_0) \geq 1 \end{cases}$$

- Все выпуклое - **единственное решение!**



Решение в случае линейно разделимых классов

- Выпишем лагранжиан:

$$L(w, w_0; \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^l \alpha_i [y_i (\langle w, x_i \rangle + w_0) - 1]$$

- с множителями Лагранжа $\alpha_i \geq 0$ для каждого ограничения
- с условиями дополняющей нежёсткости (ККТ):

$$\forall i: \alpha_i [y_i (\langle w, x_i \rangle + w_0) - 1] = 0$$

- Из необходимых условий оптимальности следует:

$$\frac{\partial L(w, w_0; \alpha)}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^l \alpha_i y_i = 0, \quad \frac{\partial L(w, w_0; \alpha)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^l \alpha_i y_i x_i$$

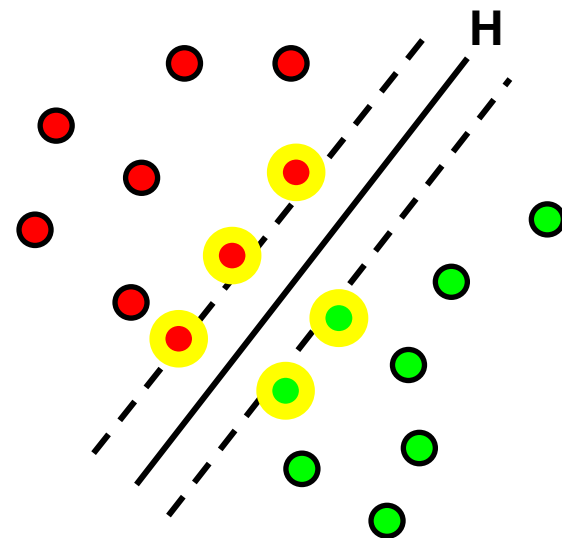
- Дискриминантная функция:

$$g(x) = \langle w, x \rangle + w_0 = \sum_{i=1}^l \alpha_i y_i \langle x_i, x \rangle + w_0$$

- Сдвиг w_0 может корректироваться «вручную», обычно инициализируется как: $w_0 = \frac{1}{l} \sum_{j=1}^l (y_j - \sum_{i=1}^l \alpha_i y_i \langle x_i, x \rangle)$

Опорные вектора в случае линейно разделимых классов

- По свойствам множителей Лагранжа: $y_i(\langle w, x_i \rangle + w_0) > 1 \Rightarrow \alpha_i = 0$:
 - $\alpha_i \neq 0$ для **опорных векторов** (наблюдения лежат строго на границе, их расстояние до H равно 1)
 - Дискриминантная функция (и модель) зависит **только от опорных векторов**:
$$a(x) = \text{sign}(\sum_{i \in SV} \alpha_i y_i \langle x_i, x \rangle + w_0)$$
 - Результат обучения не зависит от наличия в тренировочном наборе наблюдений, не лежащих на границе, их можно исключить из выборки и получить ту же модель SVM (вот только мы заранее не знаем, какие именно наблюдения лежат на границе)
 - Этим свойством пользуются алгоритмы оптимизации для SVM



Линейно неразделимые классы

- Классы не обязаны быть линейно разделимы:
 - можно попробовать перебрать оптимальные гиперплоскости, минимизируя число ошибок, но оказалось, что это NP-трудная задача (не найдено не экспоненциальных по сложности методов)
- Основной подход – дополнительно линейно штрафовать модель за «нарушение» неравенств канонической гиперплоскости:

$$\begin{cases} \min_{w, \xi, w_0} \frac{1}{2} \|w\|^2 + \frac{C}{l} \sum_{i=1}^l \xi_i \\ \forall i: y_i (\langle w, x_i \rangle + w_0) \geq 1 - \xi_i, \xi_i \geq 0 \end{cases}$$

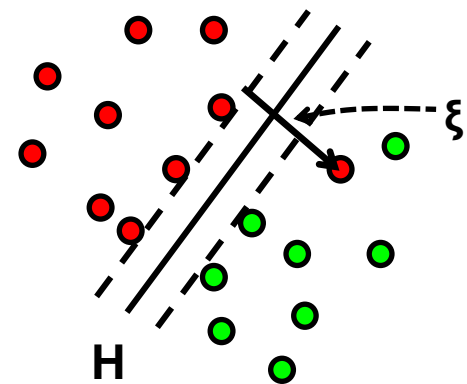
ошибка

обобщающая
способность

- параметр C – задает в явном виде компромисс между точностью и сложностью модели

- Аналогично безусловной минимизации Hinge функции потерь с L_2 регуляризацией:

$$Q_{hinge}(w, w_0) = \sum_{i=1}^l (1 - M_i(w, w_0))_+ + \gamma \|w\|^2 \rightarrow \min_{w, w_0}$$



Метод множителей Лагранжа для линейно неразделимых классов

- Снова выпишем лагранжиан:

$$L(w, w_0, \xi; \alpha, \eta) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^l \alpha_i [y_i (\langle w, x_i \rangle + w_0) - 1] - \sum_{i=1}^l \xi_i (\alpha_i + \eta_i - C)$$

- α_i - двойственные переменные к условиям $y_i (\langle w, x_i \rangle + w_0) \geq 1 - \xi_i$
- η_i - двойственные переменные к условиям $\xi_i \geq 0$
- условия дополняющей нежёсткости ККТ:

$$\forall i: \alpha_i [y_i (\langle w, x_i \rangle + w_0) - (1 - \xi_i)] = 0, \eta_i \xi_i = 0$$

(*)

- Из необходимых условий седловой точки функции Лагранжа:

$$\frac{\partial L(w, w_0, \alpha, \eta)}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^l \alpha_i y_i = 0, \frac{\partial L(w, w_0, \alpha, \eta)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^l \alpha_i y_i x_i,$$

$$\frac{\partial L(w, w_0, \alpha, \eta)}{\partial \xi} = 0 \Rightarrow \eta_i + \alpha_i = C$$

(**)

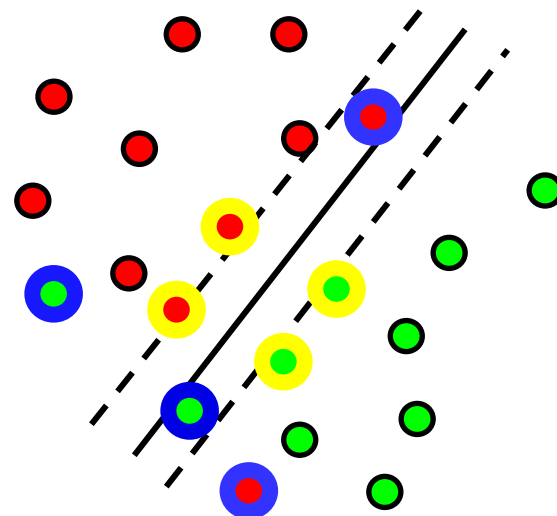
- Дискриминантная функция и сдвиг те же, но опорные вектора другие: $g(x) = \sum_{i=1}^l \alpha_i y_i \langle x_i, x \rangle + w_0$, $w_0 = \frac{1}{l} \sum_{j=1}^l (y_j - \sum_{i=1}^l \alpha_i y_i \langle x_i, x \rangle)$

Опорные вектора для линейно неразделимых классов

- Получаем два типа опорных векторов:
 - **Ошибки** – неравенство со штрафом строго НЕ выполняется:
 $\alpha_i = C, \eta_i = 0, \xi_i > 0, y_i(\langle w, x_i \rangle + w_0) > 1$
 - **Граничные** – неравенство выполняется как равенство:
 $0 < \alpha_i < C, 0 < \eta_i < C, \xi_i = 0, y_i(\langle w, x_i \rangle + w_0) = 1$
- Остальные (не важные) наблюдения:
 - **Периферийные** - неравенство со штрафом выполняется:
 $\alpha_i = 0, \eta_i = C, \xi_i = 0, y_i(\langle w, x_i \rangle + w_0) < 1$
 - снова от них ничего не зависит

Граничные
опорные вектора

опорные вектора -
ошибки



Двойственная задача

- Можно решать прямую задачу (есть для этого методы оптимизации), но оказалось, что удобнее решать двойственную
- Подставим равенства, полученные из условий (*) и (**) в $L(w, w_0, \xi; \alpha, \eta)$ и увидим, что Лагранжиан после всех сокращений зависит только от двойственных переменных α_i и имеет простую квадратичную форму:

$$L(w, w_0, \xi; \alpha, \eta) = W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j \langle x_j, x_i \rangle$$

- пользуясь свойством седловой точки Лагранжа:

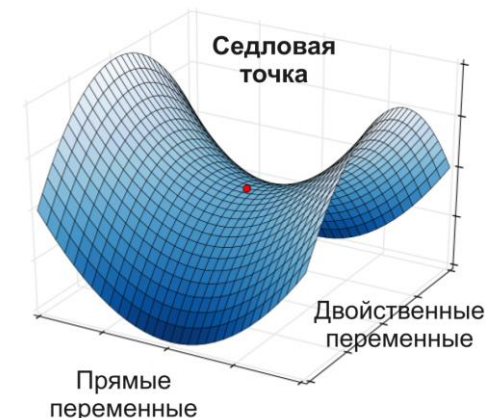
$$L(w^*, w_0^*, \xi^*; \alpha^*, \eta^*) = \min_{w, w_0, \xi} L(w, w_0, \xi; \alpha^*, \eta^*) = \max_{\alpha, \eta} L(w^*, w_0^*, \xi^*; \alpha, \eta)$$

- перейдем к решению двойственной задачи:

$$\begin{cases} \max_{\alpha} W(\alpha) \\ 0 \leq \alpha_i \leq C, \sum_{i=1}^l \alpha_i y_i = 0 \end{cases}$$

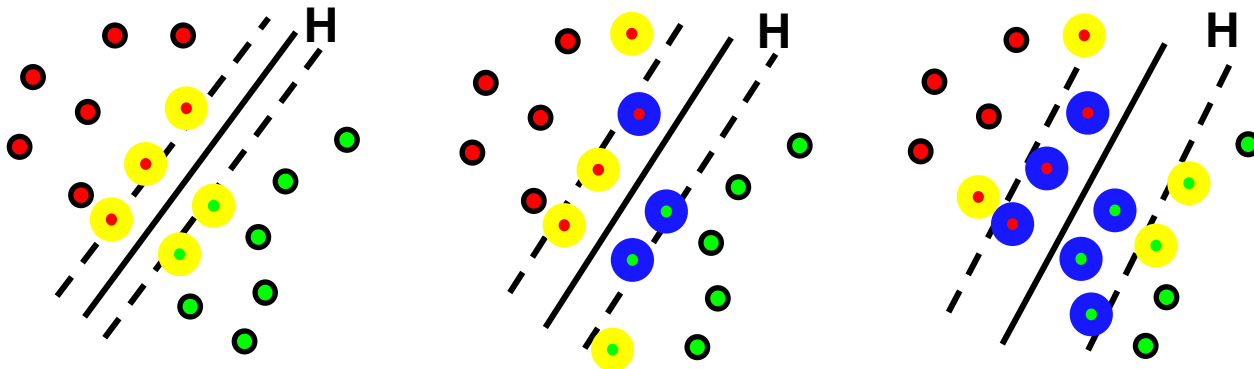
- решение прямой задачи выражается через него как:

$$a(x) = \text{sign}(\sum_{i \in SV} \alpha_i y_i \langle x_i, x \rangle + w_0)$$



Выбор параметра штрафа C

- Аналогично параметру регуляризации (но наоборот):
 - чем больше C тем меньше смещение и больше дисперсия модели
 - чем меньше C тем больше обобщающая способность и ошибка подгонки модели ($C_{left} > C_{middle} > C_{right}$)



- На практике:
 - используют стандартные эвристики: $C = \{0.1, 1, 10\}$
 - подбирают с помощью кросс-валидации (по сетке значений)
- **Не интуитивный** параметр
 - Тяжело: угадать точно, выбрать сетку для перебора, понять смысл

Nu-SVM

- Основная идея – напрямую максимизировать зазор (ширину разделяющей полосы) между классами ρ

$$\min_{\xi, \rho, w, w_0} \frac{1}{2} \|w\|^2 + \frac{1}{l} \sum_{i=1}^l \xi_i - \rho \nu$$

$$\forall i: (y_i(\langle x_i, w \rangle + w_0) \leq \rho - \xi_i, \rho \geq 0, \xi_i \geq 0$$

- Также через преобразование Лагранжиана сводится к задаче квадратичного программирования в двойственных переменных:

$$\begin{cases} \max_{\alpha} -\frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l \alpha_i \alpha_j y_i y_j \langle x_j, x_i \rangle \\ \mathbf{0} \leq \alpha_i \leq \frac{1}{l}, \sum_{i=1}^l \alpha_i y_i = 0, \sum_{i=1}^l \alpha_i \geq \nu \end{cases}$$

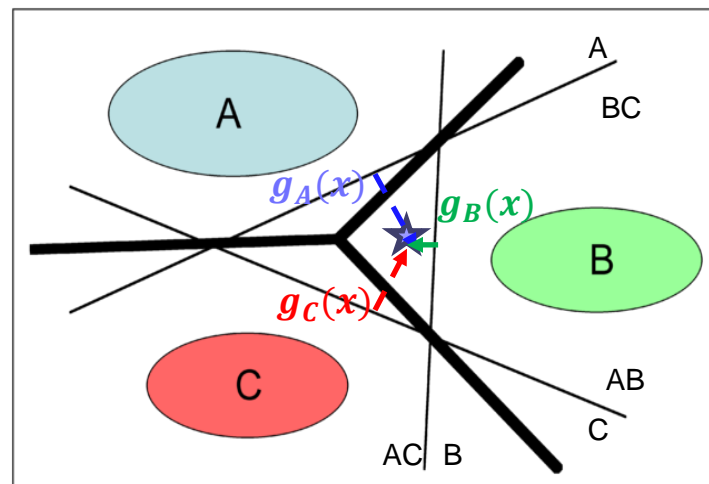
- Вместо метапараметра C используется ν с важными « ν -свойствами»:
 - ν -верхняя граница пропорции опорных векторов – ошибок
 - ν -нижняя граница пропорции опорных векторов - граничных
 - асимптотически с вероятностью 1 (при определенных условиях) эти границы достигаются

Многоклассовый SVM «каждый против всех»

- Каждый против всех:

- Строим k моделей (k -число классов), выбираем класс с наиболее уверенным прогнозом – наибольшей дискриминантной функцией:

$$\arg \max_{j=1,\dots,k} g_j(x)$$



- Особенности:

- Гарантировано есть хотя бы один несбалансированный набор (т.к. 1 класс против всех остальных)
- Вычислительно сложно при больших наборах данных – k бинарных задач с l наблюдениями в каждой
- независимое обучение – независимые $g_j(x)$, надо приводить на близкие шкалы, можно с помощью $\text{softmax}(g_1(x), \dots, g_k(x))$ или более корректно с помощью корректировки Платта

Корректировка Платта

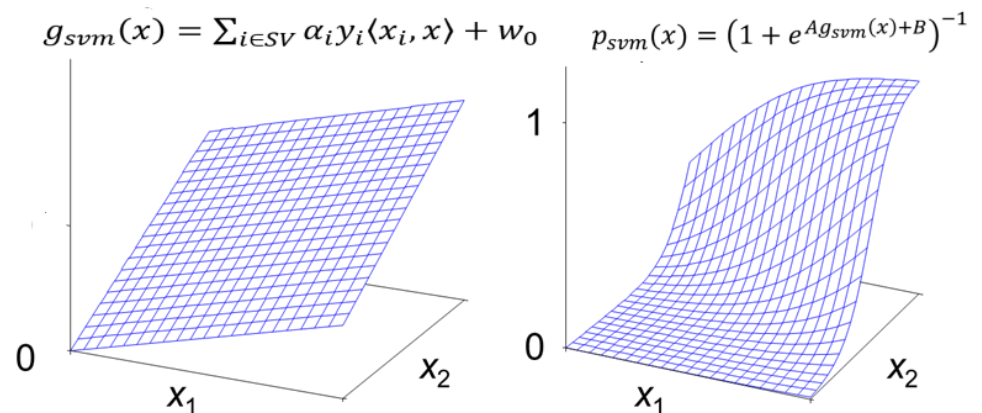
- Преобразуем отклик SVM из $(-\infty, +\infty)$ в **вероятностный** диапазон $[0,1]$ с помощью подгонки сигмоиды:

$$p_{svm}(x) = \frac{1}{1 + \exp(Ag_{svm}(x) + B)}$$

где $g_{svm}(x) = \sum_{i \in SV} \alpha_i y_i \langle x_i, x \rangle + w_0$, а A и B – параметры

- Чтобы не переобучиться:
 - параметры A и B подбираются как в логистической регрессии с одним предиктором (откликом SVM) на валидационной выборке (не использовалась для обучения SVM) или с помощью кросс-валидации
 - дополнительно часто используется «регуляризация» откликов:

$$y_i^{Platt} = \begin{cases} \frac{l_+ + 1}{l_+ + 2}, y_i = 1 \\ \frac{1}{l_+ + 2}, y_i = -1 \end{cases}$$

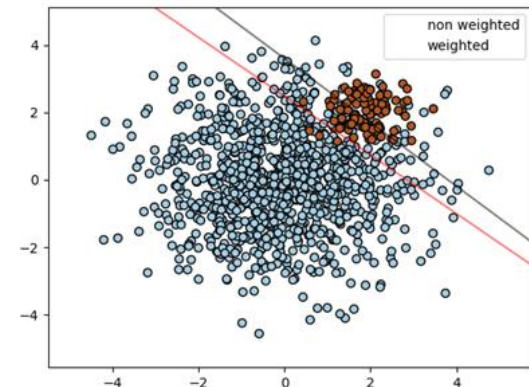


Дисбаланс классов

■ Возможные подходы к решению проблемы:

- В целом SVM менее чувствителен к дисбалансу, чем другие методы, т.к. модель зависит только от опорных векторов
- SMOTE (oversampling)
- Undersampling + корректировка сдвига w_0 – строим SVM на сбалансированной выборке, а w_0 выбираем с учетом дисбаланса, например $w_0^* = \underset{w_0}{\operatorname{argmin}} F_\beta(g(x, w_0), y)$
- Undersampling + корректировка Платта – строим SVM на сбалансированной выборке, а корректировку Платта на несбалансированной
- Используем веса наблюдений
- Используем веса классов:

$$\begin{cases} \min_{w, \xi, w_0} \frac{1}{2} \|w\|^2 + \frac{C_{-1}}{l_{-1}} \sum_{i: y_i = -1} \xi_i + \frac{C_1}{l_1} \sum_{i: y_i = +1} \xi_i \\ \forall i: y_i (\langle w, x_i \rangle + w_0) \geq 1 - \xi_i, \xi_i \geq 0 \end{cases}$$



Многоклассовый SVM «каждый против каждого»

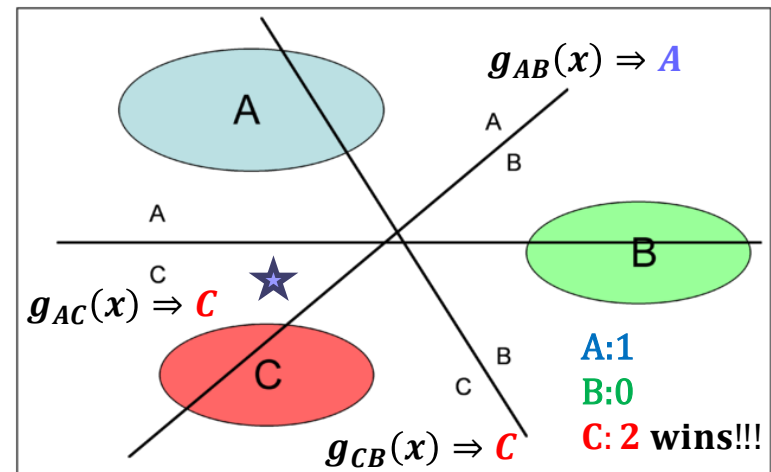
■ Каждый против каждого

- Строим $k(k-1)/2$ моделей (k -число классов), выбираем класс голосованием:

$$\arg \max_{j=1,\dots,k} \sum_{i \neq j} [g_{ij}(x)]_+$$

■ Особенности:

- меньше проблем с дисбалансом классов чем в каждом против всех
- вычислительно сложно при больших k , получаем $k(k-1)/2$ бинарных задач, правда наблюдений в каждой меньше I
- независимое обучение – независимые $g_{ij}(x)$ не так критично как в каждом против всех (не сравниваем отклики разных моделей друг с другом напрямую)
- могут быть «ничьи», простое голосование – не лучший подход, надо учитывать «уверенность» в прогнозе, а значит тоже корректировать отклики



Вероятности классов на основе попарных сравнений

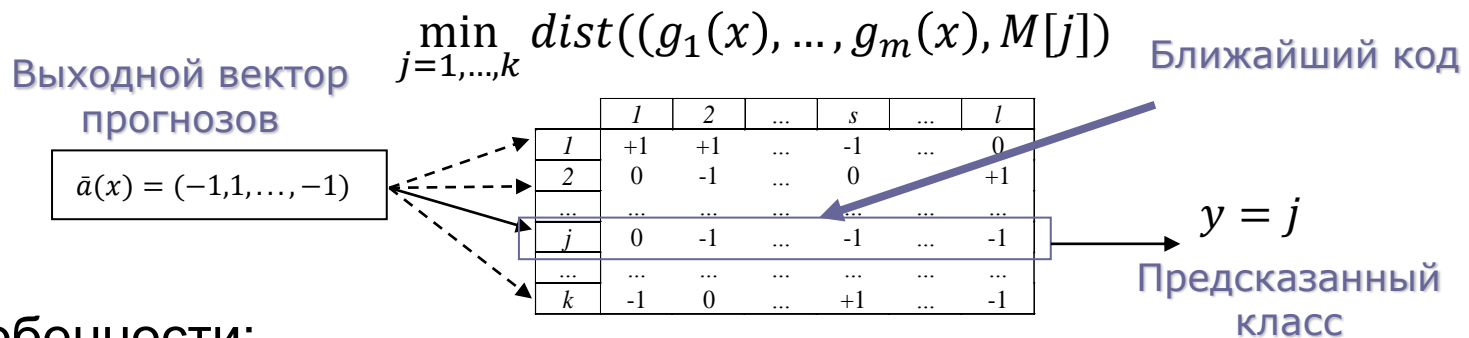
- Если по результатам применения подхода «каждый против каждого» необходимо вычислить вероятности принадлежности наблюдения x_0 каждому из k классов $p_1(x_0), \dots, p_k(x_0)$, то можно воспользоваться подходом попарных сравнений:
 - Применяем все $k(k-1)/2$ попарных моделей и получаем для каждой пары классов (i, j) значение дискриминантной функции $g_{ij}(x_0)$
 - Делаем корректировку Платта $p_{ij}(x_0)$ (x_0 можно не указывать, т.к. все считается только для него)
 - Принимаем предположение модели Брэдли-Терри для попарных сравнений: $p_{ij} = p_i / (p_i + p_j)$, где p_s неизвестны для $1 \leq s \leq k$
 - Находим их, минимизируя численным методом дивергенцию Кульбака-Лейблера :

$$\sum_{i,j} p_{ij} \log \left(\frac{p_{ij}(p_i + p_j)}{p_i} \right) + \sum_{i,j} \frac{p_i}{p_i + p_j} \log \left(\frac{p_i}{p_{ij}(p_i + p_j)} \right) \rightarrow \min_{p_1, \dots, p_k}$$

Многоклассовый ECOC SVM

■ ECOC:

- Строим кодовую матрицу M с t новыми «суперклассами», каждый объединяет комбинацию исходных классов и
- Обучаем t моделей $g_1(x), \dots, g_m(x)$
- При классификации получаем вектор прогнозов и выбираем класс с наиболее близким кодовым словом:



■ Особенности:

- вычислительную сложность можно контролировать числом столбцов
- можно рассчитывать «уверенность» в прогнозе на основе расстояний до кодовых слов или по модели Брэдли-Терри
- но качество зависит от M – если не угадали, то начинаем все заново

SVM с многоклассовой целевой функцией

■ Постановка задачи:

- пусть k – число классов
- вводим k гиперплоскостей и отдельно штрафует за нарушение каждой границы и отдельно штрафует каждую за ее сложность (максимизируем ширину каждой разделяющей полосы)
- все штрафы суммируем в целевой функции:

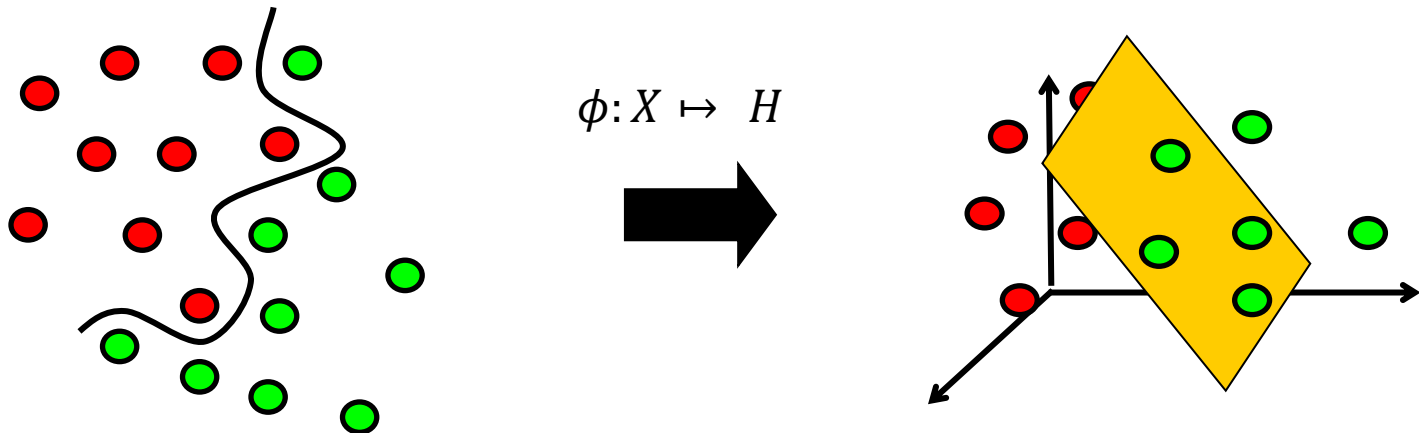
$$\left\{ \begin{array}{l} \min_{w, w_0, \xi} \frac{1}{2} \sum_{j=1}^k \|w^j\|^2 + \frac{C}{l} \sum_{i=1}^l \sum_{j \neq y_i} \xi_{ij} \\ \forall i, j: y_i (\langle w^{y_i}, x_i \rangle + w_0^{y_i}) \geq \langle w^j, x_i \rangle + w_0^j + 2 - \xi_{ij}, \xi_{ij} \geq 0 \end{array} \right.$$

■ Особенности:

- менее гибкие настройки по сравнению с остальными методами
- вычислительно сложно – много двойственных переменных при большом наборе, проблема дисбаланса тоже есть
- зато дискриминантные функции подгоняются вместе – прогнозы зависимы, не нужны корректировки

Случай существенно нелинейной границы между классами

- Следствие из Теоремы Ковера (о числе возможных линейных разбиений m точек в n -мерном пространстве):
 - В случае линейно неразделимых классов **нелинейное отображение** исходного пространства признаков в новое пространство признаков **большой** (или даже бесконечной) **размерности** увеличивает шансы линейного разделения в нем образов наблюдений из исходного пространства признаков
 - Новое пространство называется «спрямляющим»




Нелинейный метод опорных векторов

■ Основная идея:

- Нелинейное преобразование исходного пространства признаков в новое пространство большей или бесконечной размерности.
- Разделяющая плоскость строится в преобразованном пространстве
- В новом пространстве зависимость линейна, в исходном нелинейна

■ Постановка задачи оптимизации и модель C-SVM (и nu-SVM) не зависят от признаков, а только от их скалярного произведения:

- Целевая функция C-SVM:

$$W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$


Скалярное произведение

- Решающая функция: $a(x) = \text{sign} \left(\sum_{i=1}^n \alpha_i y_i \langle x_i, x \rangle + w_0 \right)$

■ Kernel trick (подмена ядра):

- Замена скалярного произведения на другое ядро неявно преобразует исходное пространство признаков в спрямляющее **без** необходимости явного **пересчета признаков**

Спрямяющее пространство для метода опорных векторов

- Спрямяющее пространство:

- Преобразование исходного пространства признаков X в гильбертово пространство H с помощью отображения $\phi: X \rightarrow H$ может быть реализовано **неявно**, за счет замены скалярного произведения в X на функцию **ядра** $K: X \times X \rightarrow \mathbb{R}$, которая является скалярным произведением в H : $\forall x_i, x_j: K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$

- Функция $K: X \times X \rightarrow \mathbb{R}$ является ядром тогда и только тогда K :

- симметрична: $\forall x_i, x_j: K(x_i, x_j) = K(x_j, x_i)$
- неотрицательно определена: $\forall f: X \rightarrow \mathbb{R}$:

$$\int_X \int_X K(x_i, x_j) f(x_i) f(x_j) dx_i dx_j \geq 0$$

- Нелинейный ядерный C-SVM (для nu-SVM аналогично):

$$W(\alpha) = \sum_{i=1}^l \alpha_i - \frac{1}{2} \sum_{i,j=1}^l \alpha_i \alpha_j y_i y_j K(x_i, x_j), a(x) = \text{sign} \left(\sum_{i=1}^l \alpha_i y_i K(x_i, x_j) + w_0 \right)$$

Ядра

Примеры популярных ядер

- Линейное ядро:

- $K(x_i, x_j) = \langle x_i, x_j \rangle$ - спрямляющее пространство совпадает с исходным

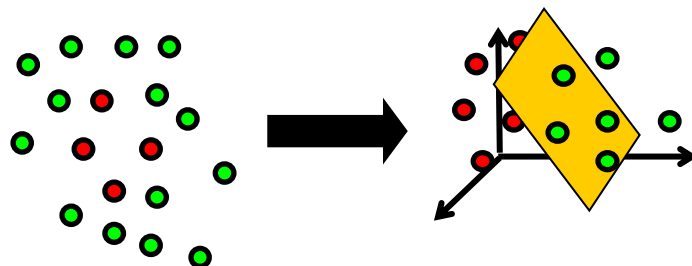
- Полиномиальное ядро степени d со сдвигом b :

- $K(x_i, x_j) = (b + \langle x_i, x_j \rangle)^d$ - разделяющая поверхность d -го порядка
 - параметр d контролирует «сложность» модели, а значит K – тоже **регуляризатор**
 - H эквивалентно пространству, полученному «ручной» генерацией полиномиальных признаков (например, PolynomialFeatures sklearn.preprocessing)
 - Частный случай $b=0$ (не получаем полный полином, только члены степени d), в результате H пространство мономов размерности C_d^{d+m-1} , простой пример при $b=0, d=2, m=2$:

$$\phi: \mathbb{R}^2 \rightarrow H = \mathbb{R}^3$$

$$\phi: ((x_i, x_j)) \rightarrow (x_i^2, x_j^2, x_i x_j)$$

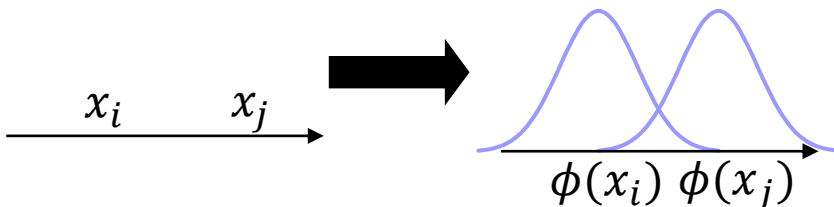
$$K(x_i, x_j) = \langle x_i, x_j \rangle^2$$



Примеры популярных «нейросетевых» ядер

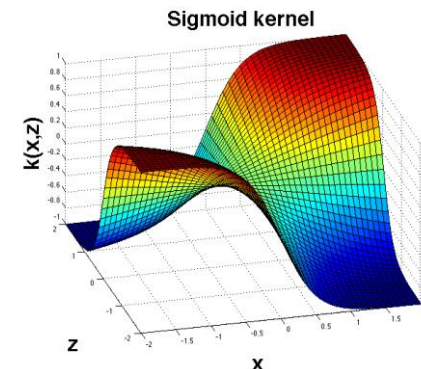
■ Гауссовская (RBF kernel):

- $K(x_i, x_j) = \exp(-\gamma(x_i - x_j)^2)$ с параметром ширина ядра γ , который «штрафует» расстояние между объектами
- Чем больше γ , тем сложнее граница – опять ядро-регуляризатор
- Спрямяющее пространство бесконечномерное пространство функций (нельзя явно выразить все координаты)



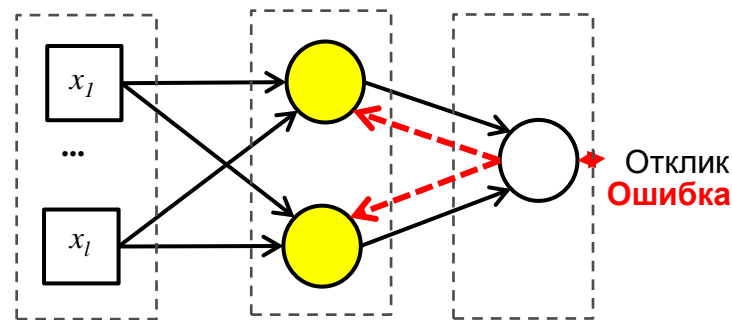
■ Сигмоидальное ядро (hyperbolic tangent kernel):

- $K(x_i, x_j) = \tanh(a\langle x_i, x_j \rangle + b)$ с параметром ширина ядра a и b
- формально является ядром не при всех значениях параметров, но тоже **регуляризует** модель
- Спрямяющее пространство – геодезическое (на эллипсоиде)



Сходство и отличия SVM и простых нейросетей

- Нейронный сети прямого распространения:
 - Сигнал передается от входного уровня к выходному по «слоям»
 - Внутри нейронов - расчет нелинейных выходных функций активации, от комбинации входных переменных, где каждый вход следующего слоя - композиции выходов предыдущего.
 - Нет задержек, времени, т.к. нет циклов (в отличии от рекуррентных сетей)
 - Модель – по сути параметрическая, уравнение зависимости отклика от предикторов определяется графом сети и функциями активации
 - Обучение – целевая функция (эмпирический риск) определяется типом и распределением отклика (как в GLM)
 - Применяются разные методы оптимизации, популярный – обратное распространение ошибки (SGD), но используют и методы 2 порядка



Входной слой Скрытый слой Выходной слой

Сходство и отличия сигмоидального SVM и однослойного персептрона

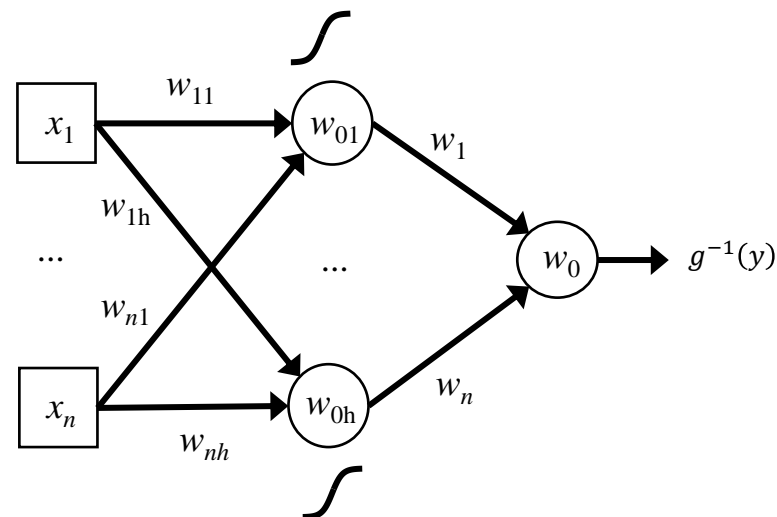
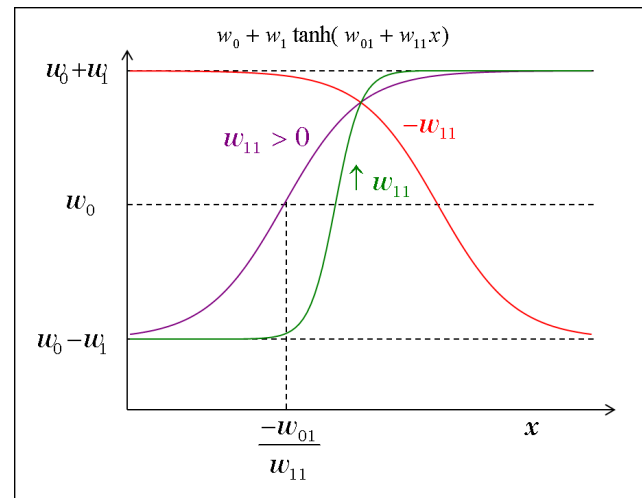
■ Однослойный персептрон (SLP):

- Один скрытый слой с сигмоидальными функциями активации
- Архитектура определяет параметрическую модель и решающую функцию как в SVM:

$$g^{-1}(y) = w_0 + \sum_{i=1}^h w_i \tanh\left(w_{0i} + \sum_j w_{ij} x_j\right)$$

■ Но с SVM принципиальные отличия:

- h - число SV, заранее не известно
- w_{0i} параметр ядра, не подгоняется
- w_{ij} - координаты опорных векторов
- w_i - произведение метки и множителя Лагранжа опорных векторов



Сходство и отличия SVM с гауссовским ядром и RBF нейросети

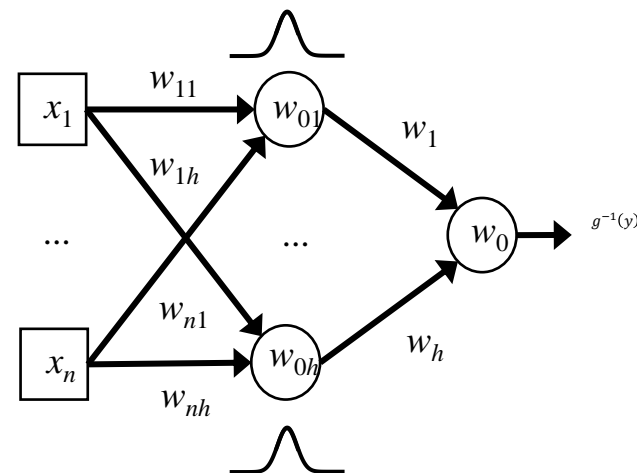
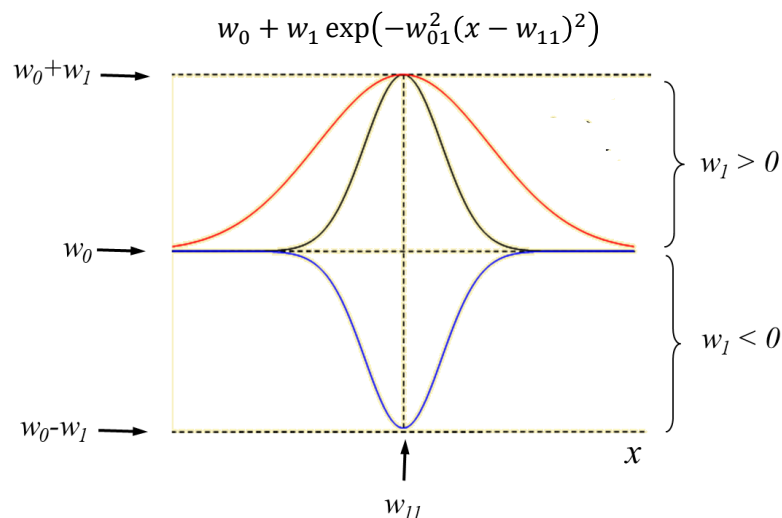
■ RBF нейросеть:

- Один скрытый слой с функциями активации Гаусса
- Архитектура определяет параметрическую модель и решающую функцию как в SVM:

$$g^{-1}(y) = w_0 + \sum_{i=1}^h w_i e^{-w_{0i}(\sum_j (w_{ij} - x_j)^2)}$$

■ Но с SVM принципиальные отличия:

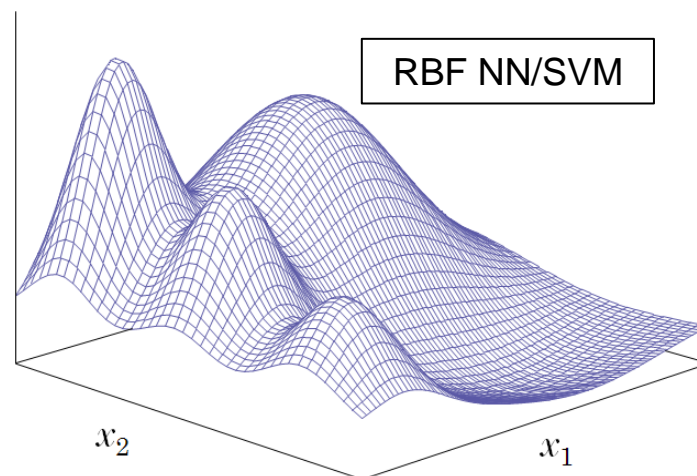
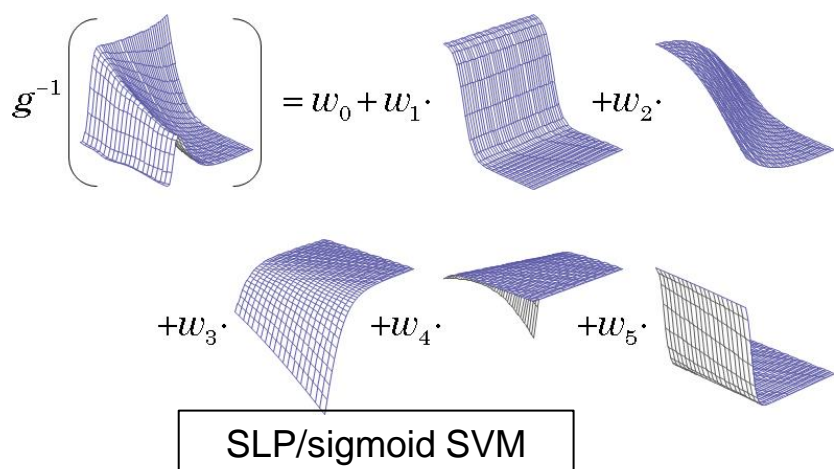
- h - число SV, заранее не известно
- w_{0i} параметр ядра, не подгоняется
- w_{ij} - координаты опорных векторов, а не прототипов нейронов
- w_i - произведение метки и множителя Лагранжа опорных векторов



Сходство и отличия SVM и простых нейросетей

■ Сходство:

- структурно одинаковые решающие функции и соответственно похожие формы искомых зависимостей (но параметры ищутся по-разному и априори зафиксированы разные параметры, у нейросетей более гибкий набор параметров):



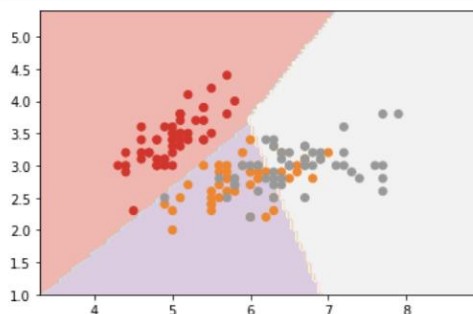
■ Ключевые преимущества SVM:

- **единственное решение** (при любом начальном приближении)
- более эффективные и контролируемые методы оптимизации

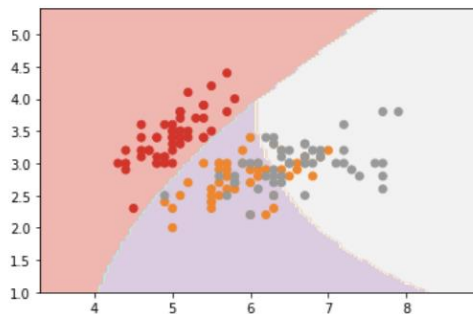
Влияние ядра

```
for degree in [1, 3, 5]:  
    DecisionBoundaryDisplay.from_estimator(  
        SVC(kernel="poly", degree=degree).fit(X, y), X, cmap="Pastell1")  
    plt.scatter(*X.T, c=y, cmap="Set1")
```

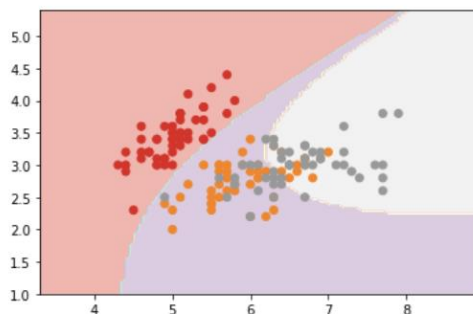
Degree=1



Degree=3

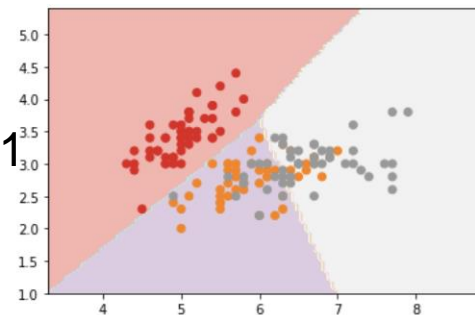


Degree=5

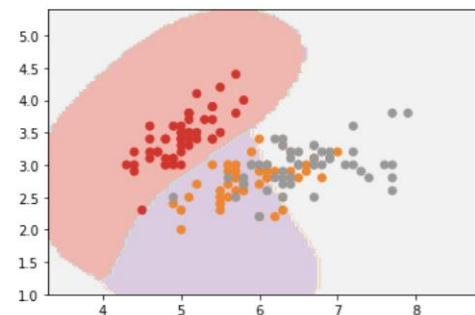


```
for gamma in [0.1, 1, 10]:  
    DecisionBoundaryDisplay.from_estimator(  
        SVC(kernel="rbf", gamma=gamma).fit(X, y), X, cmap="Pastell1")  
    plt.scatter(*X.T, c=y, cmap="Set1")
```

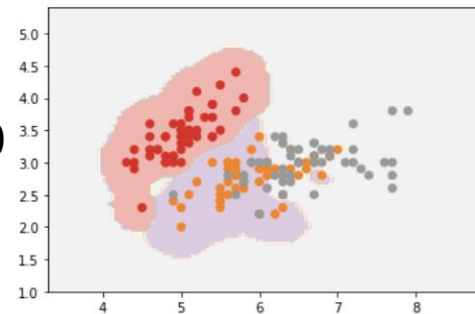
Gamma=0.1



Gamma=1



Gamma=10

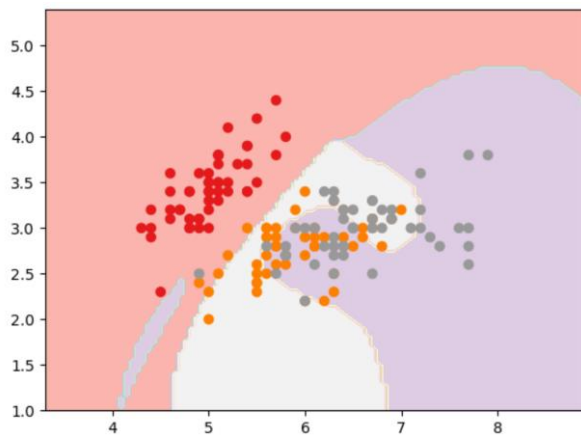


Влияние параметра штрафа за СЛОЖНОСТЬ

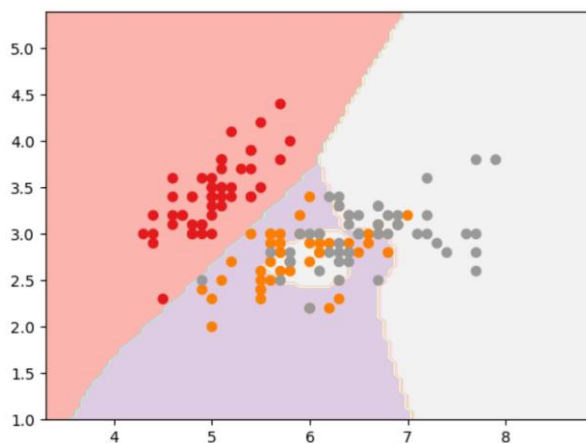
```
from sklearn.svm import NuSVC
```

```
X, y = load_iris(return_X_y=True)  
X = X[:, :2]
```

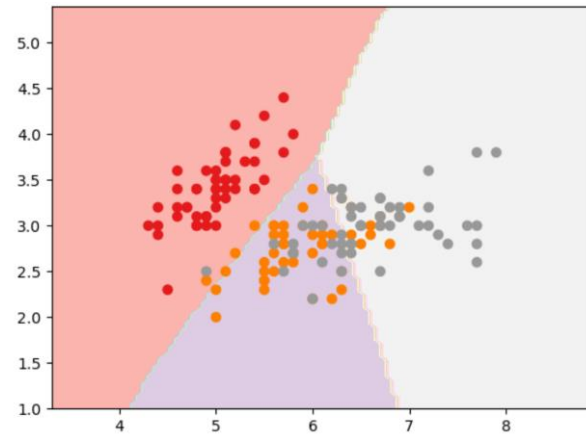
```
for i, nu in enumerate([0.1, 0.5, 0.9]):  
    DecisionBoundaryDisplay.from_estimator(  
        NuSVC(nu=nu, kernel="rbf").fit(X, y), X, cmap="Pastel1")  
    plt.scatter(*X.T, c=y, cmap="Set1")
```



Nu=0.1



Nu=0.5



Nu=0.9

Методы синтеза ядер

■ Популярные методы:

- Линейная комбинация (с положительными весами) ядер – ядро
- Произведение ядер - ядро
- RBF от любого расстояния – ядро (кстати, ядро задает расстояние в спрямляющем пространстве: $\sqrt{K(x_i, x_i) + K(x_j, x_j) - 2K(x_i, x_j)}$)
- $\forall \phi: X \rightarrow \mathbb{R}, K(x_i, x_j) = \phi(x_i)\phi(x_j)$ – ядро
- $\forall \phi: X \rightarrow X, K(x_i, x_j) = K_{base}(\phi(x_i), \phi(x_j))$ – ядро, если K_{base} - ядро
- $\forall s: X \times X \rightarrow \mathbb{R}$ симметричная и интегрируемая, то $K(x_i, x_j) = \int_X s(x_i, z) s(x_j, z) dz$ – ядро
- Если K_{base} - ядро и $f: \mathbb{R} \rightarrow \mathbb{R}$ представима в виде сходящегося степенного ряда с неотрицательными коэффициентами, то $K(x_i, x_j) = f(K_{base}(x_i, x_j))$ – ядро
- Если есть вероятностная модель, где $p(x|\theta)$ – правдоподобие, а M – положительно определенная квадратная симметричная матрица, то $K(x_i, x_j) = \nabla_{\theta} \ln p(x_i|\theta)^T M^{-1} \nabla_{\theta} \ln p(x_j|\theta)$ - ядро

Ядра для сложных структур (пример – спектральное ядро)

- Для работы с текстовыми данными:
 - можно использовать векторную модель мешка слов и любое стандартное ядро над ней
 - но такая модель не учитывает порядок слов и расстояние
- Можно построить ядро, учитывающее порядок и расстояния:
 - пусть Σ – фиксированный алфавит, а множество всевозможных строк длины n есть Σ^n , тогда множество всех строк $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$
 - спрямляющее пространство H , такое, что каждая координата связана с некоторой допустимой строкой u в Σ ,
 - тогда для любой строки s ее u -я координата может быть задана как
$$[\phi_n(s)]_u = \sum_{i:s(i)=u} \lambda^{l(i)}$$
 - где i – множество индексов, формирующее подстроку из s ,
 - $l(i)$ - расстояния между первым и последним индексом
 - $0 < \lambda < 1$ – весовой параметр, контролирует разреженность подстроки

Ядра для сложных структур (пример – спектральное ядро для строк)

- Пример: $[\phi_3(\text{"Nasdaq"})]_{\text{asd}} = \lambda^3$, а $[\phi_3(\text{"lass das"})]_{\text{asd}} = 2\lambda^5$

- Строковое ядро для строк s и t длины n :

$$K_n(s, t) = \sum_{u \in \Sigma^n} [\phi_n(s)]_u [\phi_n(t)]_u = \sum_{u \in \Sigma^n} \sum_{i, j: s(i)=t(j)=u} \lambda^{l(j)+l(i)}$$

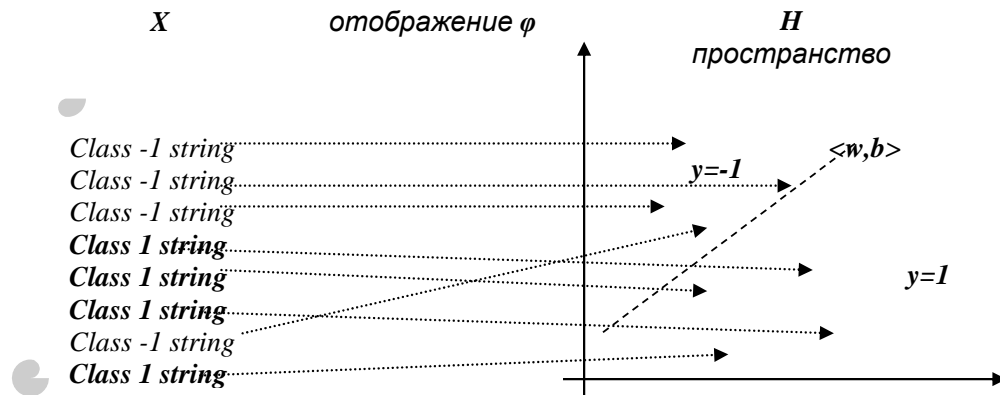
- Строковое ядро для строк s и t произвольной длины:

□ с набором параметров «веса» длин $c_n \geq 0$: $K(s, t) = \sum_n c_n K_n(s, t)$

- Есть эффективный алгоритм динамического программирования для расчета таких ядер

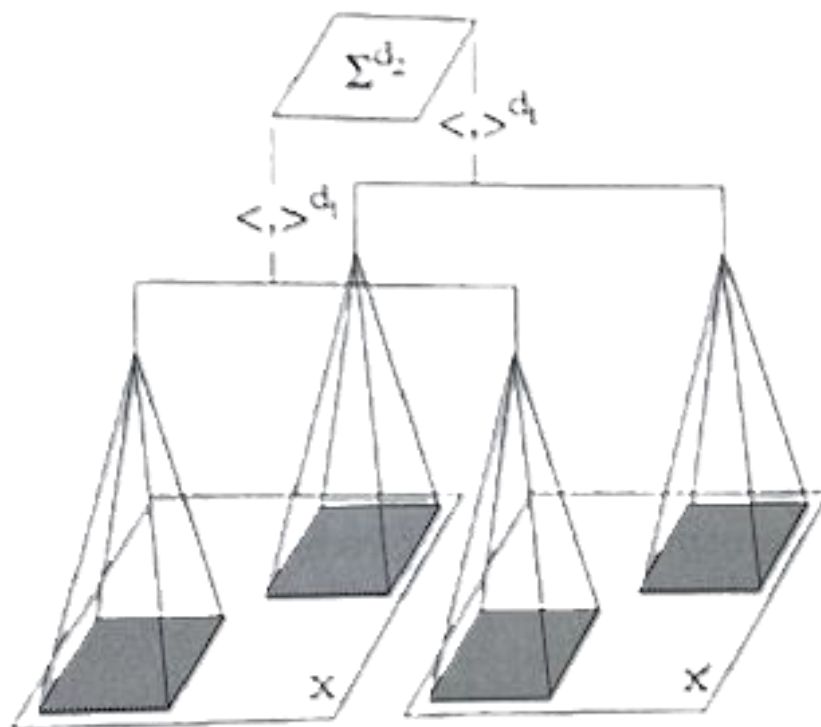
- Примеры применения:

- ДНК классификация
- SMS/chat anti-spam
- language identification



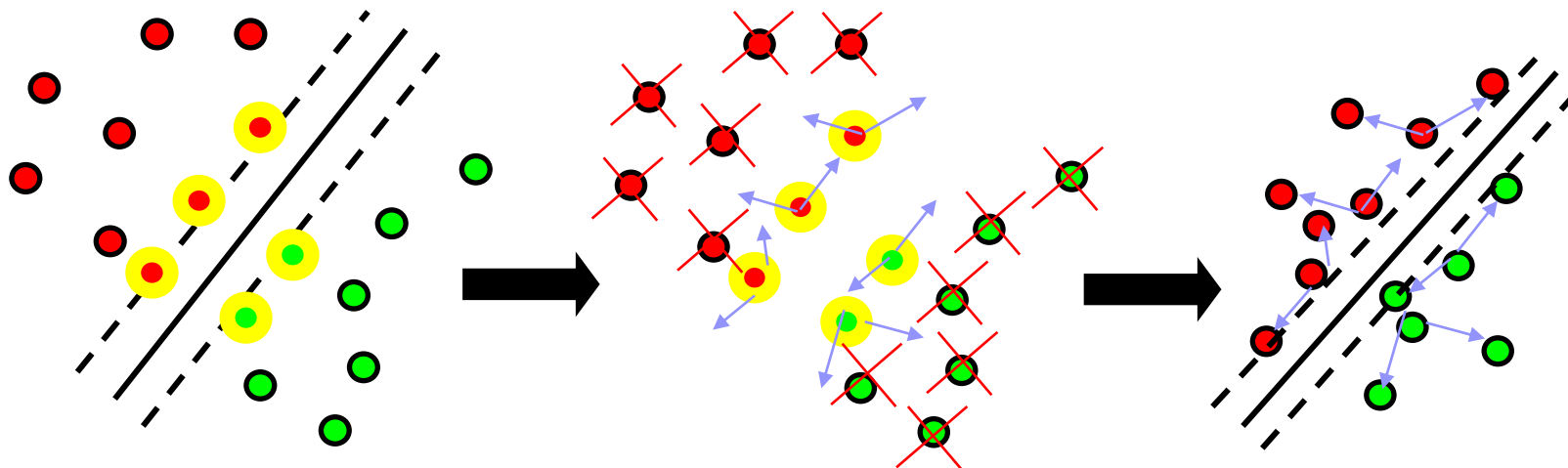
Локальные ядра для пространственных признаков (изображений, строк, ДНК ...)

- Аналог свертки в CNN:
 - Считаются ядра по локальным областям в пространстве признаков
 - Могут дополнительно учитываться веса признаков на основе удаленности от центра области локализации
 - Полученные значения агрегируются так, чтобы результат оставался ядром
 - Может быть несколько уровней «вложенности»



Augmentation в SVM – виртуальные опорные вектора

- Ключевая особенность SVM – нет необходимости «зашумлять» всю выборку:
 - решается задача без augmentation
 - «зашумляются» только опорные вектора
 - «искаженные» опорные вектора называются **виртуальными**
 - после этого строится классификатор только на них
 - можно повторить несколько раз



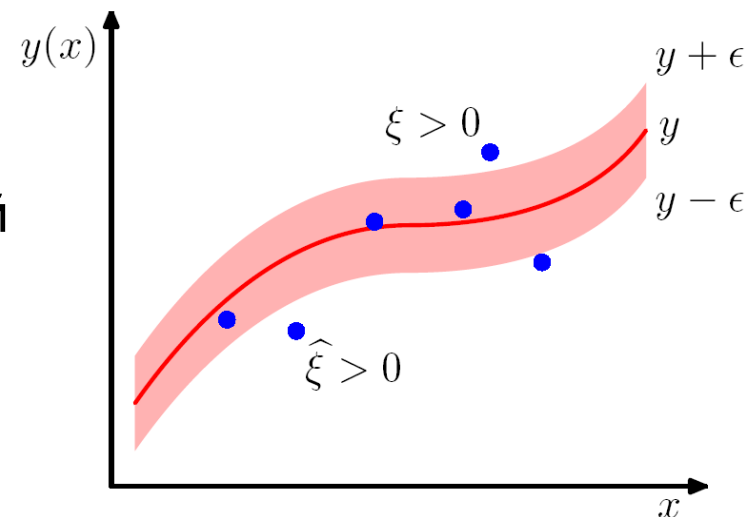
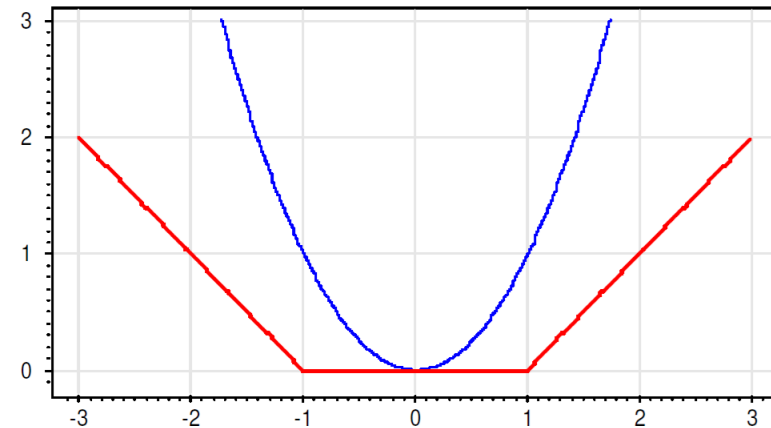
Нелинейная регрессия SVM

- ϵ -чувствительная функция потерь определяется как (линейное) расстояние до отклика за вычетом порога:

$$L_{\epsilon}(y, g(x)) = \begin{cases} 0, & |g(x) - y| < \epsilon \\ |g(x) - y| - \epsilon, & |g(x) - y| \geq \epsilon \end{cases}$$

- Точки «внутри» полосы отступа от отклика – не штрафуются
- Уравнение регрессии задается как:
$$g(x) = \langle w, x \rangle_H + w_0$$
- Формулируется регуляризованный эмпирический риск:

$$\min_{w, w_0} C \sum_{i=1}^l L_{\epsilon}(g(x_i), y_i) + \frac{1}{2} \|w\|^2$$



Прямая задача оптимизации

- ϵ -формулировка (прямая задача):

$$\left\{ \begin{array}{l} \min_{\xi^{+/-}, w, w_0} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^l (\xi_i^+ + \xi_i^-) \right) \\ (\langle w, x_i \rangle_H + w_0) - y_i \leq \epsilon + \xi_i^+, \xi_i^+ \geq 0 \\ y_i - (\langle w, x_i \rangle_H + w_0) \leq \epsilon + \xi_i^-, \xi_i^- \geq 0 \end{array} \right.$$

Diagram annotations: A green box labeled "штраф за сложность" (penalty for complexity) points to the term $\frac{1}{2} \|w\|^2$. A blue box labeled "штраф за ошибку" (penalty for error) points to the term $C \sum_{i=1}^l (\xi_i^+ + \xi_i^-)$.

- ν -формулировка (прямая задача) — ϵ не метепараметр:

$$\left\{ \begin{array}{l} \min_{\epsilon, \xi^{+/-}, w, w_0} \left(\frac{1}{2} \|w\|^2 + C \left(\frac{1}{l} \sum_{i=1}^l (\xi_i^+ + \xi_i^-) + \epsilon \nu \right) \right) \\ (\langle w, x_i \rangle_H + w_0) - y_i \leq \epsilon + \xi_i^+, \xi_i^+ \geq 0, \\ y_i - (\langle w, x_i \rangle_H + w_0) \leq \epsilon + \xi_i^-, \xi_i^- \geq 0 \\ \epsilon \geq 0 \end{array} \right.$$

Diagram annotations: A green box labeled "штраф за сложность" points to the term $\frac{1}{2} \|w\|^2$. A blue box labeled "штраф за ошибку" points to the term $C \left(\frac{1}{l} \sum_{i=1}^l (\xi_i^+ + \xi_i^-) + \epsilon \nu \right)$.

- ν -свойства аналогичны классификации: верхняя граница пропорции опорных векторов — ошибок и нижняя граница пропорции опорных векторов — граничных, асимптотически достигаются

Двойственная задача ϵ - SVR

- Аналогично задачи классификации:

- выписываем Лагранжиан, дифференцируем по прямым переменным, подставляем в Лагранжиан
- с учетом условий ККТ переходим к двойственной, она не зависит от переменных прямой задачи и является задачей квадратичного программирования

- Двойственная задача:

$$\begin{cases} \max_{\alpha^+, \alpha^-} -\frac{1}{2} \sum_{i,j=1}^l (\alpha_j^+ - \alpha_j^-) (\alpha_i^+ - \alpha_i^-) K(x_j, x_i) - \epsilon \sum_{i=1}^l (\alpha_i^+ + \alpha_i^-) + \sum_{i=1}^l y_i (\alpha_i^+ - \alpha_i^-) \\ \sum_{i=1}^l y_i (\alpha_i^+ - \alpha_i^-) = 0, 0 \leq \alpha_i^+ \leq \frac{C}{l}, 0 \leq \alpha_i^- \leq \frac{C}{l} \end{cases}$$

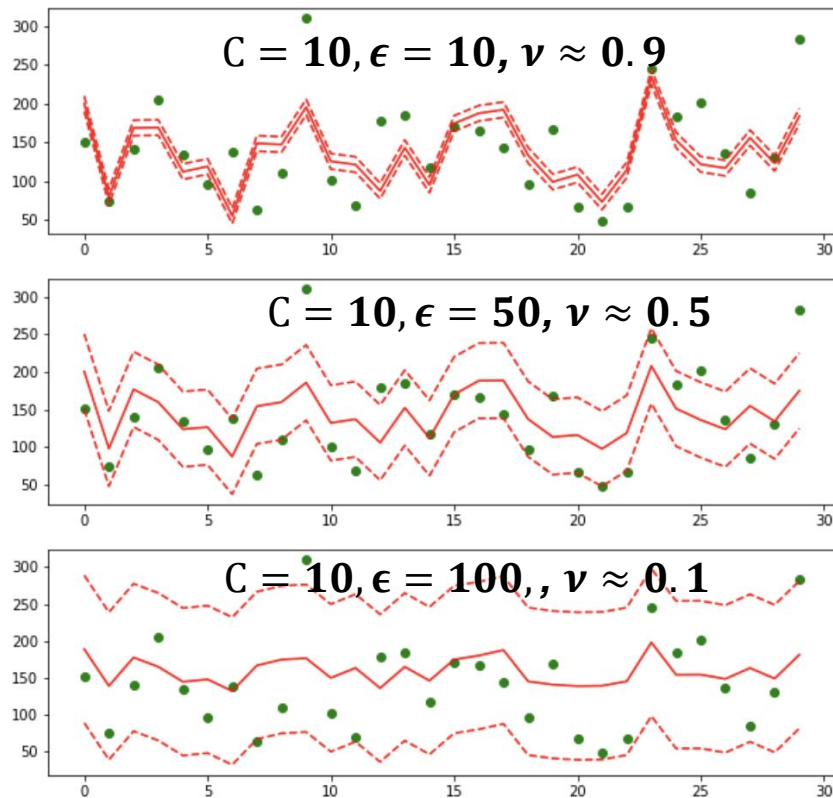
- Результат:

- Опорные вектора: $\alpha_i^+ = C/l$ или $\alpha_i^- = C/l$ для ошибок (за ϵ полосой)
- $\alpha_i^+ \alpha_i^- = 0$, $w_0 = \frac{1}{l} [\sum_{\alpha_i^+ > 0} (y_j - K(x, x_i) - \epsilon) + \sum_{\alpha_i^- > 0} (y_j - K(x, x_i) + \epsilon)]$
- Функция регрессии $g(x) = \sum_{i \in SV} (\alpha_i^+ - \alpha_i^-) K(x, x_i) + w_0$

Двойственная задача ν - SVR

■ Аналогичный вывод:

$$\left\{ \begin{array}{l} \max_{\alpha^+, \alpha^-} \sum_{i=1}^l y_i (\alpha_i^+ - \alpha_i^-) - \\ - \frac{1}{2} \sum_{i,j=1}^l (\alpha_j^+ - \alpha_j^-) (\alpha_i^+ - \alpha_i^-) K(x_j, x_i) \\ \sum_{i=1}^l y_i (\alpha_i^+ - \alpha_i^-) = 0, \\ 0 \leq \alpha_i^+ \leq \frac{C}{l}, 0 \leq \alpha_i^- \leq \frac{C}{l} \\ \sum_{i=1}^l y_i (\alpha_i^+ + \alpha_i^-) \leq \nu C \end{array} \right.$$



- Функция регрессии $g(x) = \sum_{i \in SV} (\alpha_i^+ - \alpha_i^-) K(x, x_i) + w_0$
- Связь ν - SVR и ϵ - SVR: при одинаковых C, w, w_0 однозначно связаны $\nu \Leftrightarrow \epsilon$, все равно какой из них задавать

Общие особенности методов оптимизации для SVM

- Основные вычислительные затраты:

- Расчет матрицы ядра (для экономии памяти кэшируют по элементно или по строкам или по блокам матрицы ядра)
- Численный метод оптимизации для задачи квадратичного программирования (когда остановится и какой использовать?)

- Если g – решающая функция C-SVM, то:

- можно оценить эмпирический риск с регуляризацией:

$$Q_{reg}(g) \geq Q_{reg}(g_{opt}) \geq Q_{reg}(g) - \frac{1}{Cl} Gap(g)$$

- сходимость обычно оценивают через $Gap(g)$
- C-классификация $Gap(g) = \sum_j C \max(0, 1 - y_j g(x_j)) + \alpha_j (y_j g(x_j) - 1)$
- ϵ -регрессия:

$$Gap(g) = \sum_j C (\xi_j^+ + \xi_j^-) + \alpha_j^+ (\epsilon + g(x_j) - y_j) + \alpha_j^- (\epsilon - g(x_j) + y_j)$$

- есть оценки $Gap(g)$ и для ν – SVM

Методы оптимизации для задачи квадратичного программирования



- Задача квадратичного программирования:

Q – симметричная матрица

$$\begin{cases} \min_{\alpha} \frac{1}{2} \alpha^T Q \alpha + c^T \alpha \\ A \alpha \leq b \end{cases}$$

- SVM особенности:

- Q – строится на основе матрицы ядер K и можно попытаться ее упростить
- Существенная часть переменных или не опорные вектора ($\alpha = 0$) или опорные вектора ошибки ($\alpha = C$), если найдем, то можно не пересчитывать

Градиентный спуск для задачи квадратичного программирования в SVM

- Постановка задачи (g – решающая функция SVM):

$$Q_{reg} = \frac{1}{l} \sum_i L(y_i, g(x_i)) + \frac{\gamma}{2} \|g\|^2, \partial Q_{reg} = \frac{1}{l} \sum_i L'(y_i, g(x_i)) + \gamma g$$

- Шаг градиента для дискриминантной функции (η -длина шага):

$$g \leftarrow g - \eta \partial Q_{reg} = (1 - \eta\gamma)g - \frac{\eta}{l} \sum_i L'(y_i, g(x_i))K(x_i, \cdot)$$

- Шаг градиента для коэффициентов дискриминантной функции:

$$\alpha \leftarrow \alpha - \eta(\gamma\alpha + c), \text{ где } c_i = L'(y_i, g(x_i))$$

- для классификации $L'(y_i, g(x_i)) = \begin{cases} -y_i, y_i g(x_i) < 1 \\ 0, \text{ иначе} \end{cases}$

- для классификации $L'(y_i, g(x_i)) = \begin{cases} 1, g(x_i) - y_i > \epsilon \\ -1, y_i - g(x_i) > \epsilon \\ 0, \text{ иначе} \end{cases}$

- для не опорных векторов α остается 0 ...

Жадная разреженная аппроксимация матрицы ядра

- Основная идея – уменьшить размерность матрицы ядра:
 - можно через матричные разложения, но вычислительно затратно
 - поэтому исходную матрицу ядра $K^{l \times l}$ приближают линейной комбинацией подмножеств ее строк и столбцов (без потери общности можем считать их первыми $m \ll l$), тогда:

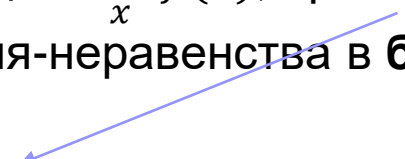
$$\|K - \tilde{K}\|^2 \rightarrow \min_{\beta} \Rightarrow \beta_{opt} = K^{l \times m} (K^{m \times m})^{-1}, \forall i, j: \tilde{K}(x_i, x_j) = \sum_{s=1}^m \beta_{is} K(x_i, x_s)$$

- Решаем задачу меньшей размерности с $K^{m \times m}$, и выражаем решение исходной задачи через нее
- Как выбрать индексы для m ?
 - есть простые процедуры $(K^{s+1 \times s+1})^{-1} \leftarrow (K^{s \times s})^{-1}$ и $\beta_{opt}^{l \times s+1} \leftarrow \beta_{opt}^{l \times s}$
 - жадный алгоритм: начинает с пустого множества индексов, берет небольшое случайное подмножество индексов, находит среди них лучший, добавляет его, берет следующее случайное подмножество и так далее пока не найдет m индексов

Методы внутренней точки для SVM

- Основные свойства (неформально):
 - Эффективны для небольших задач, полиномиальная сходимость
 - Для некоторых ядер можно эффективно сократить (аппроксимировать) матрицу ядра и распараллелить оптимизацию в том числе по данным
 - Выбор начального приближения «внутри» ограничений и последовательное приближение решения (например, с помощью ньютоновского метода) с штрафом за приближение к границе
 - Есть много вариантов для SVM, например, барьерные методы
- Упрощенный пример барьерного метода:

- сводим задачу условной оптимизации $\min_x f(x)$, при $cx \leq 0$ к безусловной, добавляя ограничения-неравенства в **барьерную функцию** с параметром μ :

$$\min_x f(x) + \mu \sum_j \log(-c_j(x))$$


- последовательно пересчитываем x , делая шаг методом Ньютона, и меняем $\mu \rightarrow 0$ по определенной стратегии (например, $\mu^{(t+1)} \leftarrow \mu^{(t+1)} \sigma$, $\sigma \in (0,1)$), после каждого пересчета x

Методы активных множеств

- Покоординатный спуск в пространстве переменных:

- Решаем стандартную задачу SVM квадратичного программирования:

$$\begin{cases} \min_{\alpha} \frac{1}{2} \alpha^T Q \alpha + c^T \alpha \\ A \alpha = b, \{0 \leq \alpha \leq u\} \text{ или } \{\alpha + t = u, \alpha \geq 0, t \geq 0\} \end{cases}$$

- «Замораживаем» часть переменных с индексами $S_f \subset [l]$, остальные индексы – рабочее (активное) множество $S_w = [l] \setminus S_f$

- Получаем:

$$Q = \begin{bmatrix} Q_{ww} & Q_{fw} \\ Q_{wf} & Q_{ff} \end{bmatrix}, c = (c_w, c_f), A = [A_w \ A_f], u = (u_w, u_f)$$

- Получаем и решаем такую же задачу, но меньшей размерности по α_w и с измененными граничными условиями: «заморожено» - не оптимизируется

$$\begin{cases} \min_{\alpha_w} \frac{1}{2} \alpha_w^T Q_{ww} \alpha_w + [c_w + Q_{wf} \alpha_f]^T \alpha_w + \frac{1}{2} \alpha_f^T Q_{ff} \alpha_f + c_f^T \alpha_f \\ A_w \alpha_w = b - A_f \alpha_f, \{0 \leq \alpha_w \leq u_w\} \text{ или } \{\alpha_w + t_w = u_w, \alpha_w \geq 0, t_w \geq 0\} \end{cases}$$

Методы активных множеств

- Основная проблема:
 - как выбрать индексы для активного множества?
 - приведет ли выбранная стратегия обновления активного множества к сходимости к глобальному оптимуму?
- Стратегии выбора (и перебора) индексов:
 - минимизация штрафа за ошибки (нарушения граничных условий)
 - максимизация градиента прямой или двойственной целевой функции
 - на основе улучшения Лагранжиана напрямую
- Популярный подход - последовательная минимальная оптимизация (Sequential minimal optimization, **SMO**)
 - В активном множестве только **два индекса** – позволяет получить аналитическое решение малой задачи для обоих (без итераций)
 - Перебираются пары индексов (возможно с дополнительными эвристиками для не рассмотрения части наблюдений)
 - Находятся новые значения множителей Лагранжа для каждой пары и значение Лагранжиана, по нему выбирается лучшая пара

Выводы по SVM

- SVM для классификации строит разделяющую гиперплоскость:
 - с максимально широкой границей в спрямляющем пространстве признаков, неявно индуцированном kernel функцией, используемой в качестве скалярного произведения
- SVR строит:
 - линейную в спрямляющем пространстве и (возможно) нелинейную в исходном пространстве признаков регуляризованную регрессию, используя ϵ –толерантную робастную функцию потерь
- Параметры регуляризации задают компромисс между точностью подгонки и обобщающей способностью модели:
 - контролируя ее сложность
 - уменьшая влияние выбросов и мультиколлинеарности
 - C или ν для классификации
 - C и ν или C и ϵ для регрессии
 - ν свойства позволяют явно контролировать ожидаемую пропорцию ошибок
 - параметры функции ядра также влияют на сложность модели

Выводы по SVM

- Модели опорных векторов представляют собой:
 - линейную комбинацию kernel функций от части наблюдений из тренировочного набора (опорных векторов) и зависят только от них
- Достоинства:
 - Единственное решение при любом начальном приближении
 - Kernel trick – смена пространства признаков «на лету» без необходимости их явно рассчитывать
 - Понятная геометрическая интерпретация
 - Относительная устойчивость к проклятию размерности
 - Явный контроль сложности модели
- Основные недостатки:
 - Качество существенно зависит от метапараметров регуляризации
 - Построение ядра для конкретной задачи –трудоемкий, плохо формализуемый процесс, особенно для структурированных данных
 - Вычислительная сложность как на этапе построения матрицы ядра, так и на этапе оптимизации
 - Нет встроенного отбора и оценка важности признаков