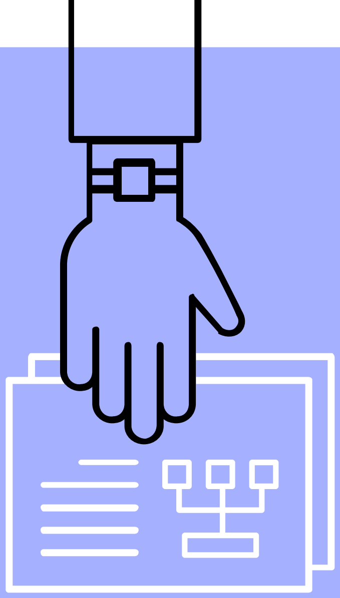
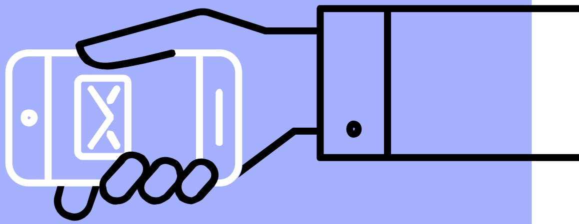
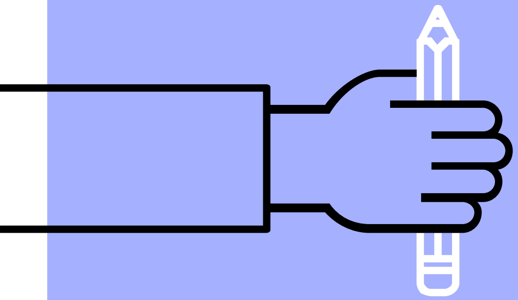
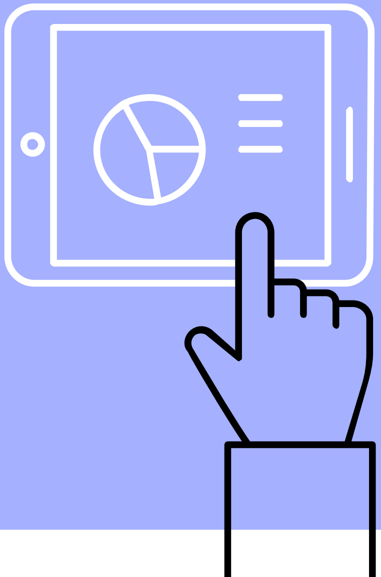




# Support Vector Machine





# Добрый день!

Весь используемый в проекте код:https://github.com/EvgeniaKomleva/ippi_svm

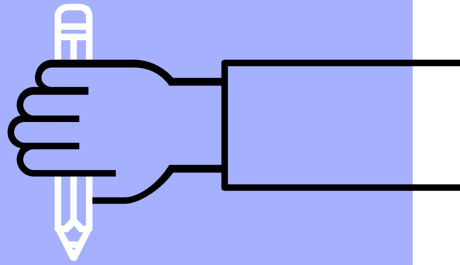
Работу выполнила Комлева Евгения для представления в Институте проблем передачи информации имени А. А. Харкевича РАН

По всем вопросам :

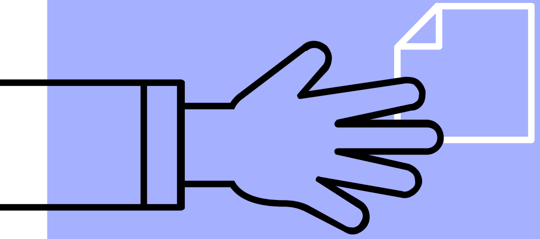
komleva.1999@inbox.ru

*Метод опорных векторов один из самых известных методов в машинном обучении. Точно входит в топ-3*

Воронцов К. В.

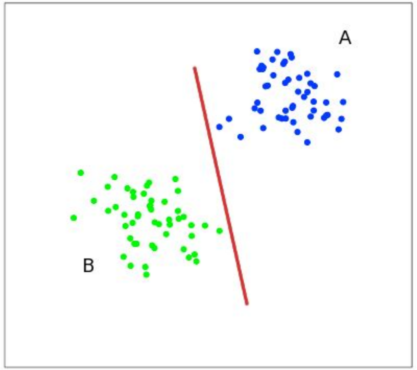


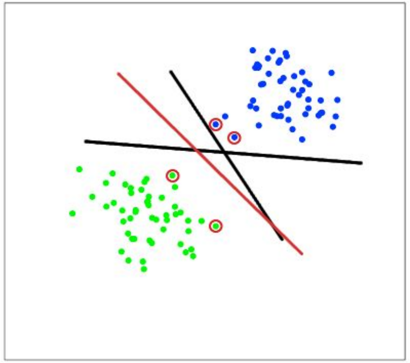
# 1.
# Линейно разделимая выборка



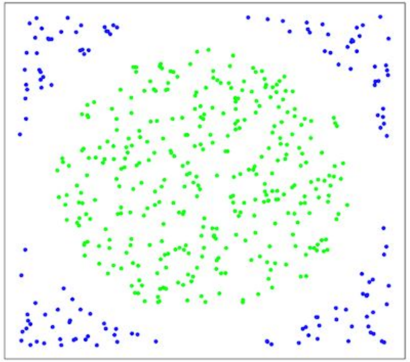
# Типы данных

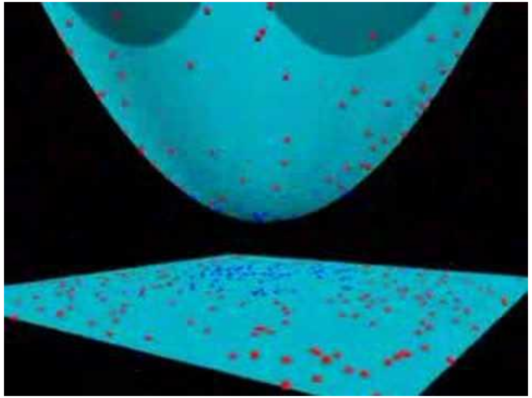
## Линейно разделимые

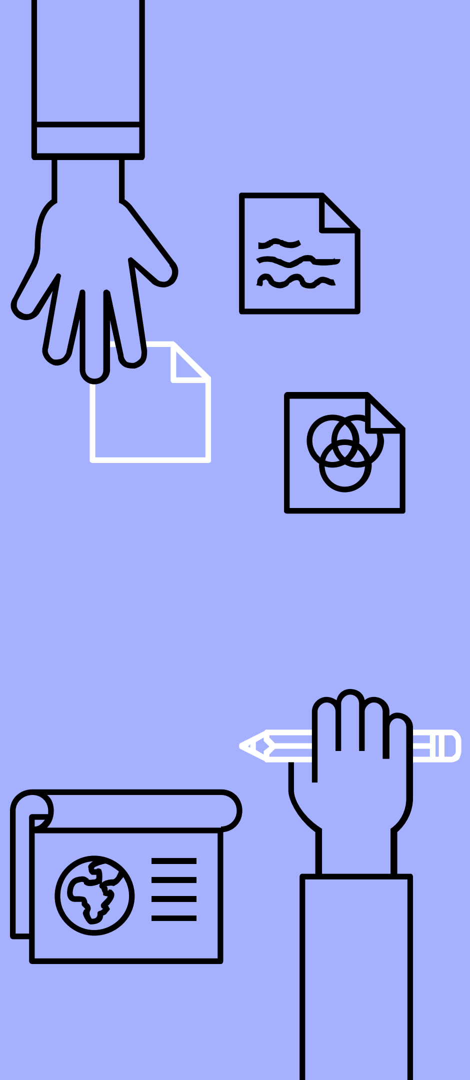


## Нелинейно разделимые







# Задача обучения линейного классификатора

**Дано:**

Обучающая выборка $X^\ell = (x_i, y_i)_{i=1}^{\ell}$,

$x_i$ — объекты, векторы из множества $X = \mathbb{R}^n$,

$y_i$ — метки классов, элементы множества $Y = \{-1, +1\}$.

**Найти:**

Параметры $w \in \mathbb{R}^n$, $w_0 \in \mathbb{R}$ линейной модели классификации

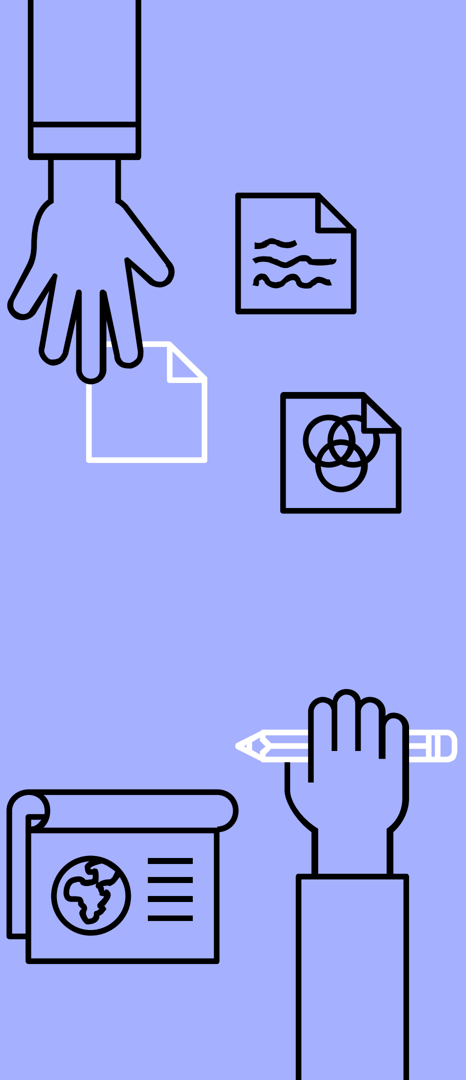
$$a(x; w, w_0) = \operatorname{sign}\big(\langle x, w\rangle - w_0\big).$$

**Критерий** — минимизация эмпирического риска:

$$\sum_{i=1}^{\ell} \big[a(x_i; w, w_0) \neq y_i\big] = \sum_{i=1}^{\ell} \big[M_i(w, w_0) < 0\big] \;\to\; \min_{w, w_0}.$$

где $M_i(w, w_0) = \big(\langle x_i, w\rangle - w_0\big) y_i$ — *отступ* (margin) объекта $x_i$,

$b(x) = \langle x, w\rangle - w_0$ — *дискриминантная функция*.



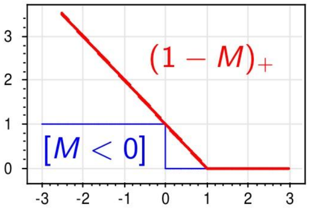
# Аппроксимация и регуляризация эмпирического риска

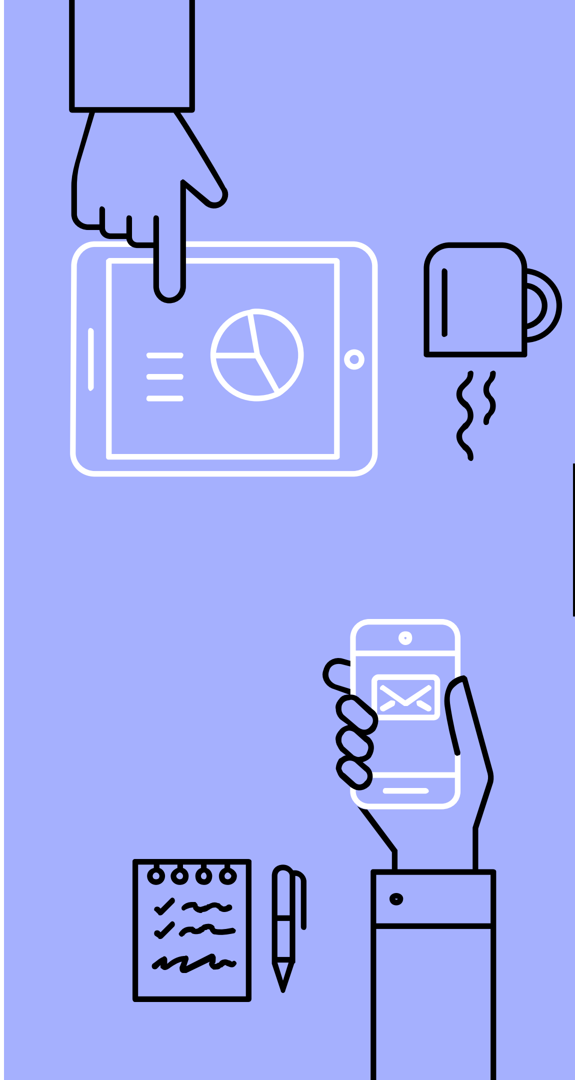
Эмпирический риск — это кусочно-постоянная функция.
Заменим его оценкой сверху, непрерывной по параметрам:

$$Q(w, w_0) = \sum_{i=1}^{\ell} \left[M_i(w, w_0) < 0\right] \leqslant$$

$$\leqslant \sum_{i=1}^{\ell} \left(1 - M_i(w, w_0)\right)_+ + \frac{1}{2C}\|w\|^2 \to \min_{w, w_0}.$$

- *Аппроксимация* штрафует объекты за приближение к границе классов, увеличивая зазор между классами
- *Регуляризация* штрафует неустойчивые решения в случае мультиколлинеарности



# Оптимальная разделяющая гиперплоскость

Линейный классификатор:

$$a(x, w) = \operatorname{sign}\bigl(\langle w, x\rangle - w_0\bigr), \quad w, x \in \mathbb{R}^n,\ w_0 \in \mathbb{R}.$$

Пусть выборка $X^\ell = (x_i, y_i)_{i=1}^\ell$ линейно разделима:

$$\exists w, w_0 : \quad M_i(w, w_0) = y_i\bigl(\langle w, x_i\rangle - w_0\bigr) > 0, \quad i = 1, \ldots, \ell$$

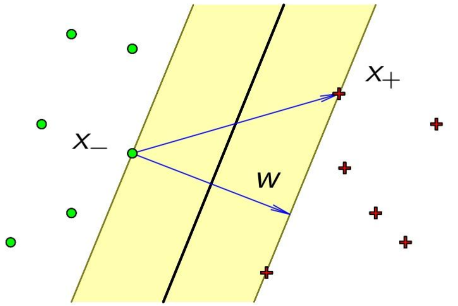
Нормировка: $\min\limits_{i=1,\ldots,\ell} M_i(w, w_0) = 1.$

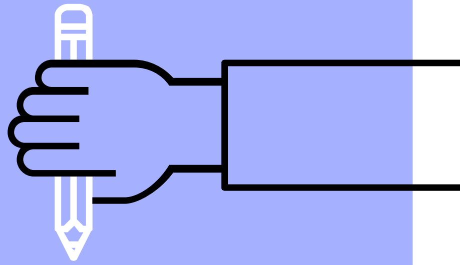
Разделяющая полоса:

$$\bigl\{x\colon\ -1 \leqslant \langle w, x\rangle - w_0 \leqslant 1\bigr\}.$$

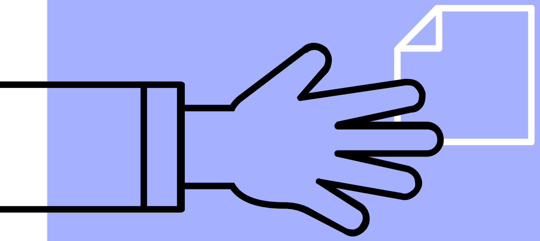
Ширина полосы:

$$\frac{\langle x_+ - x_-, w\rangle}{\|w\|} = \frac{2}{\|w\|} \to \max.$$





# 2.
# Линейно неразделимая выборка



# Переход к линейно неразделимой выборке

Постановка задачи в случае линейно разделимой выборки:

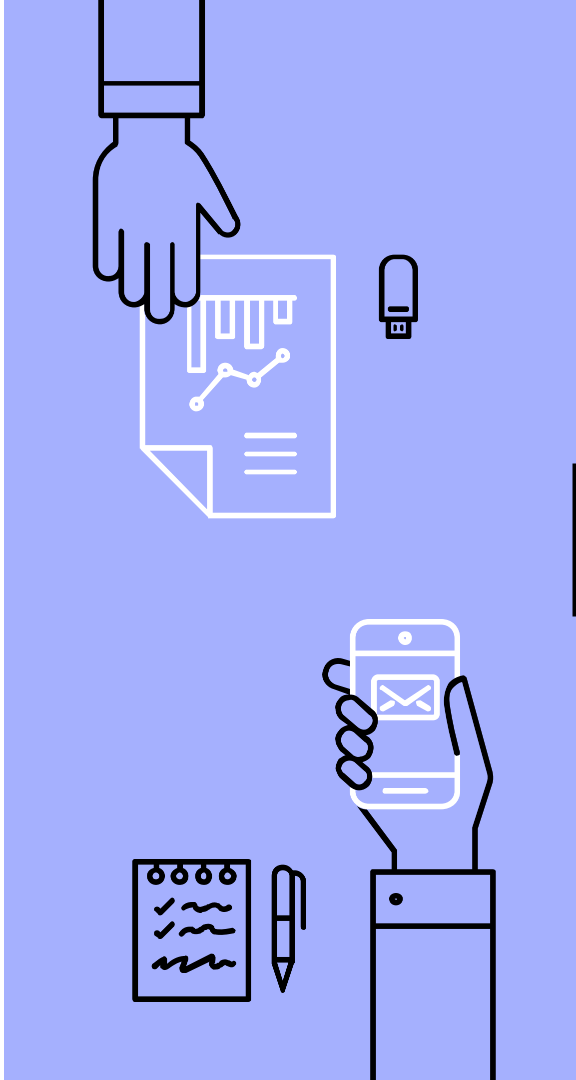
$$\begin{cases} \frac{1}{2}\|w\|^2 \to \min\limits_{w, w_0}; \\ M_i(w, w_0) \geqslant 1, \quad i = 1, \dots, \ell. \end{cases}$$

Общий случай — линейно неразделимая выборка:

$$\begin{cases} \frac{1}{2}\|w\|^2 + C \sum\limits_{i=1}^{\ell} \xi_i \to \min\limits_{w, w_0, \xi}; \\ M_i(w, w_0) \geqslant 1 - \xi_i, \quad i = 1, \dots, \ell; \\ \xi_i \geqslant 0, \quad i = 1, \dots, \ell. \end{cases}$$

Исключая $\xi_i$, получаем задачу безусловной минимизации:

$$C \sum_{i=1}^{\ell} \bigl(1 - M_i(w, w_0)\bigr)_+ + \frac{1}{2}\|w\|^2 \to \min_{w, w_0}.$$



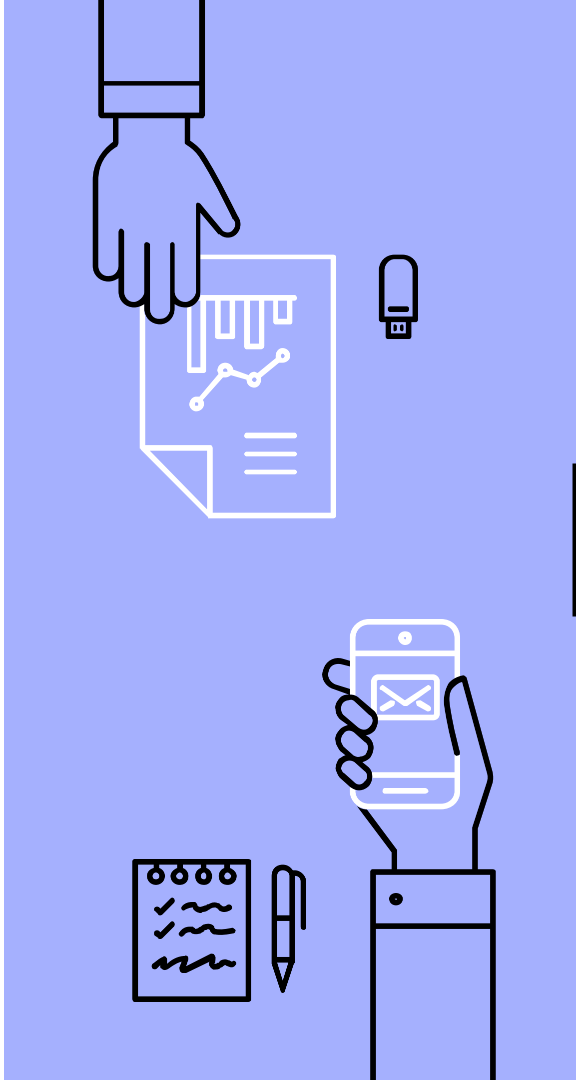
# Условие Каруша-Куна-Таккера

Задача математического программирования:

$$\begin{cases} f(x) \to \min\limits_{x}; \\ g_i(x) \leqslant 0, \quad i = 1, \dots, m; \\ h_j(x) = 0, \quad j = 1, \dots, k. \end{cases}$$

Необходимые условия. Если $x$ — точка локального минимума, то существуют множители $\mu_i$, $i = 1, \dots, m$, $\lambda_j$, $j = 1, \dots, k$:

$$\begin{cases} \dfrac{\partial \mathscr{L}}{\partial x} = 0, \quad \mathscr{L}(x; \mu, \lambda) = f(x) + \sum\limits_{i=1}^{m} \mu_i g_i(x) + \sum\limits_{j=1}^{k} \lambda_j h_j(x); \\ g_i(x) \leqslant 0;\ h_j(x) = 0;\ (\text{исходные ограничения}) \\ \mu_i \geqslant 0;\ (\text{двойственные ограничения}) \\ \mu_i g_i(x) = 0;\ (\text{условие дополняющей нежёсткости}) \end{cases}$$



# Применение условий ККТ в задаче SVM

Функция Лагранжа:

$$\mathscr{L}(w, w_0, \xi; \lambda, \eta) =$$

$$= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell} \lambda_i \bigl(M_i(w, w_0) - 1\bigr) - \sum_{i=1}^{\ell} \xi_i (\lambda_i + \eta_i - C),$$

$\lambda_i$ — переменные, двойственные к ограничениям $M_i \geqslant 1 - \xi_i$;
$\eta_i$ — переменные, двойственные к ограничениям $\xi_i \geqslant 0$.

$$\begin{cases} \frac{\partial \mathscr{L}}{\partial w} = 0, \quad \frac{\partial \mathscr{L}}{\partial w_0} = 0, \quad \frac{\partial \mathscr{L}}{\partial \xi} = 0; \\ \xi_i \geqslant 0, \quad \lambda_i \geqslant 0, \quad \eta_i \geqslant 0, \quad i = 1, \ldots, \ell; \\ \lambda_i = 0 \ \text{ либо } \ M_i(w, w_0) = 1 - \xi_i, \quad i = 1, \ldots, \ell; \\ \eta_i = 0 \ \text{ либо } \ \xi_i = 0, \quad i = 1, \ldots, \ell; \end{cases}$$

# Двойственная задача

$$\begin{cases} -\sum\limits_{i=1}^{\ell} \lambda_i + \frac{1}{2}\sum\limits_{i=1}^{\ell}\sum\limits_{j=1}^{\ell} \lambda_i\lambda_j y_i y_j \langle x_i, x_j\rangle \;\to\; \min\limits_{\lambda}; \\ 0 \leqslant \lambda_i \leqslant C, \quad i = 1, \ldots, \ell; \\ \sum\limits_{i=1}^{\ell} \lambda_i y_i = 0. \end{cases}$$

Решив эту задачу численно относительно $\lambda_i$,
получаем линейный классификатор:

$$a(x) = \operatorname{sign}\Bigl(\sum_{i=1}^{\ell} \lambda_i y_i \langle x_i, x\rangle - w_0\Bigr),$$

где $w_0 = \sum\limits_{i=1}^{\ell} \lambda_i y_i \langle x_i, x_j\rangle - y_j$ для такого $j$, что $\lambda_j > 0$, $M_j = 1$

**Определение**

Объект $x_i$ называется *опорным*, если $\lambda_i \neq 0$.

# Двойственная задача нелинейное обобщение SVM

$$\begin{cases} -\sum\limits_{i=1}^{\ell} \lambda_i + \frac{1}{2}\sum\limits_{i=1}^{\ell}\sum\limits_{j=1}^{\ell} \lambda_i\lambda_j y_i y_j K(x_i, x_j) \;\to\; \min\limits_{\lambda}; \\ 0 \leqslant \lambda_i \leqslant C, \quad i = 1, \dots, \ell; \\ \sum\limits_{i=1}^{\ell} \lambda_i y_i = 0. \end{cases}$$

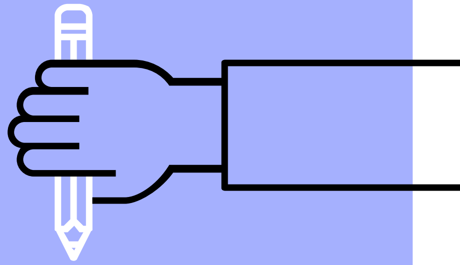
Решив эту задачу численно относительно $\lambda_i$, получаем линейный классификатор:

$$a(x) = \operatorname{sign}\Big( \sum_{i=1}^{\ell} \lambda_i y_i K(x_i, x) - w_0 \Big),$$

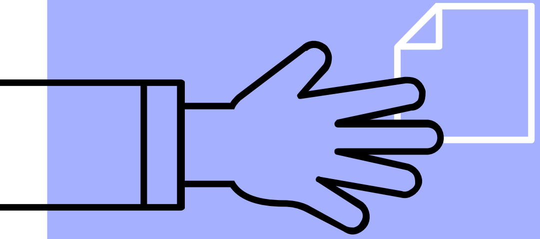
где $w_0 = \sum\limits_{i=1}^{\ell} \lambda_i y_i K(x_i, x_j) - y_j$ для такого $j$, что $\lambda_j > 0$, $M_j = 1$

**Определение**

Объект $x_i$ называется *опорным*, если $\lambda_i \neq 0$.



# 3.
# Ядра



# Ядра для нелинейного обобщения SVM

**Определение**

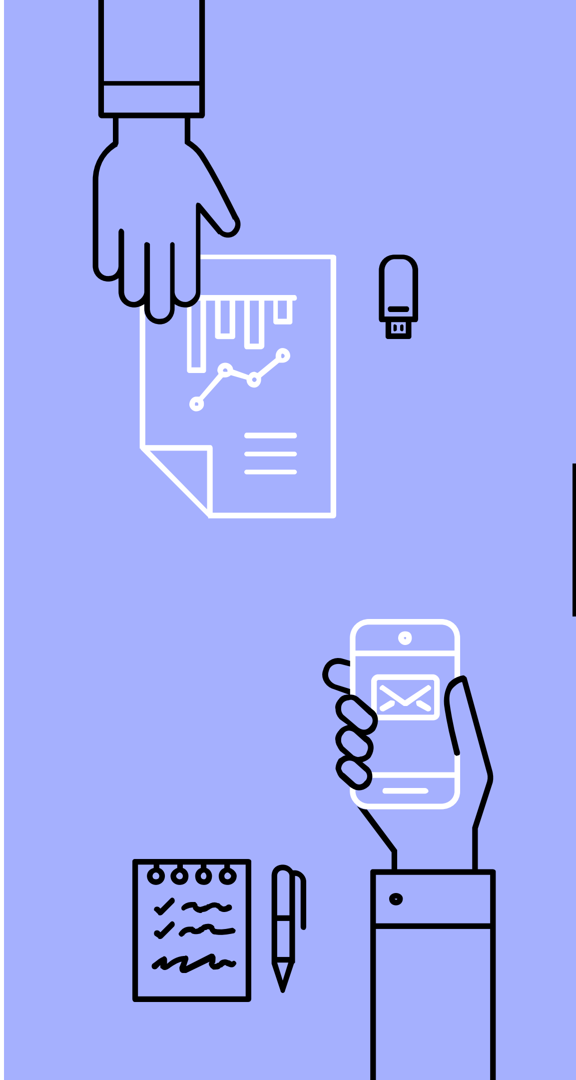
Функция от пары объектов $K(x, x')$ называется *ядром*, если она представима в виде скалярного произведения

$$K(x, x') = \langle \psi(x), \psi(x') \rangle$$

при некотором преобразовании $\psi \colon X \to H$ из пространства признаков $X$ в новое *спрямляющее* пространство $H$.

**Возможная интерпретация:**
признак $f_i(x) = K(x_i, x)$ — это оценка близости объекта $x$ к опорному объекту $x_i$. Выбирая опорные объектов, SVM осуществляет отбор признаков в линейном классификаторе

$$a(x) = \operatorname{sign}\Big( \sum_{i=1}^{\ell} \lambda_i y_i K(x_i, x) - w_0 \Big).$$



# Примеры ядер

Ядра в SVM расширяют линейную модель классификации:

1. $K(x, x') = \big(\langle x, x' \rangle + 1\big)^d$
— полиномиальная разделяющая поверхность степени $\leqslant d$;

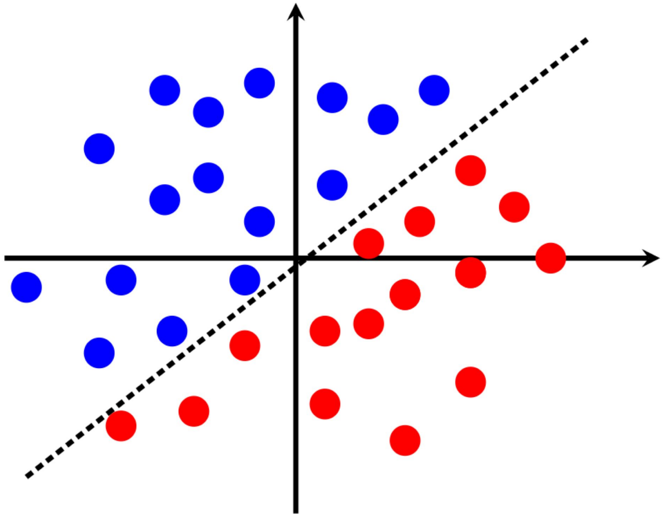
2. $K(x, x') = \sigma\big(\langle x, x' \rangle\big)$
— нейронная сеть с заданной функцией активации $\sigma(z)$
($K$ не при всех $\sigma$ является ядром);

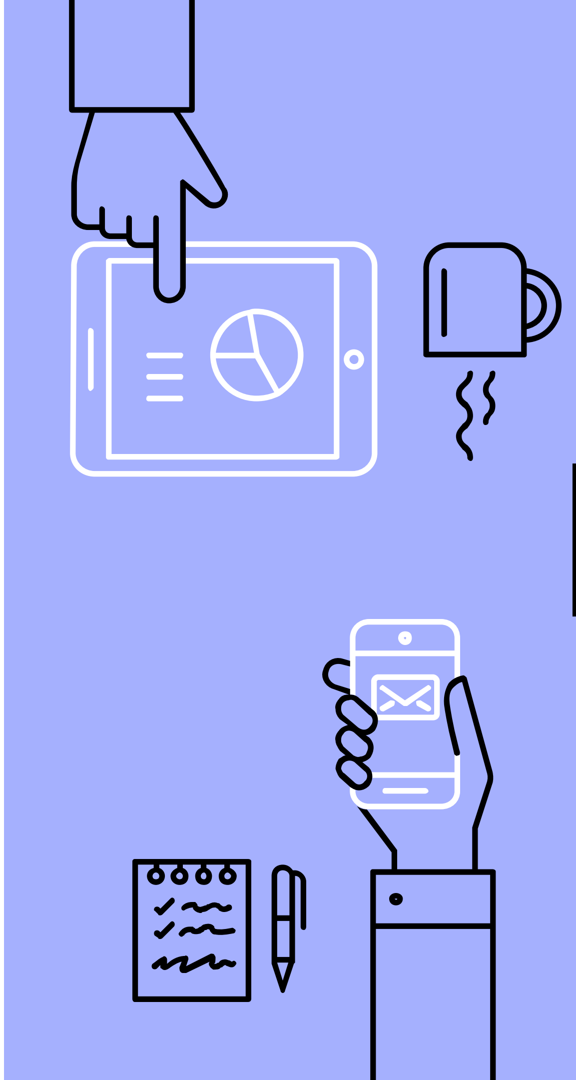
3. $K(x, x') = \mathrm{th}\big(k_1 \langle x, x' \rangle - k_0\big), \;\; k_0, k_1 \geqslant 0$
— нейросеть с сигмоидными функциями активации;

4. $K(x, x') = \exp\big(-\gamma \|x - x'\|^2\big)$
— сеть радиальных базисных функций (RBF ядро);

# Примеры ядер

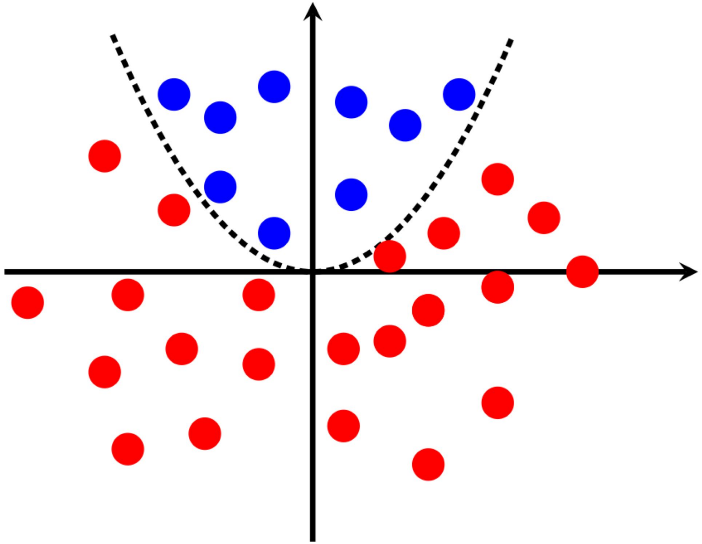
## Линейное

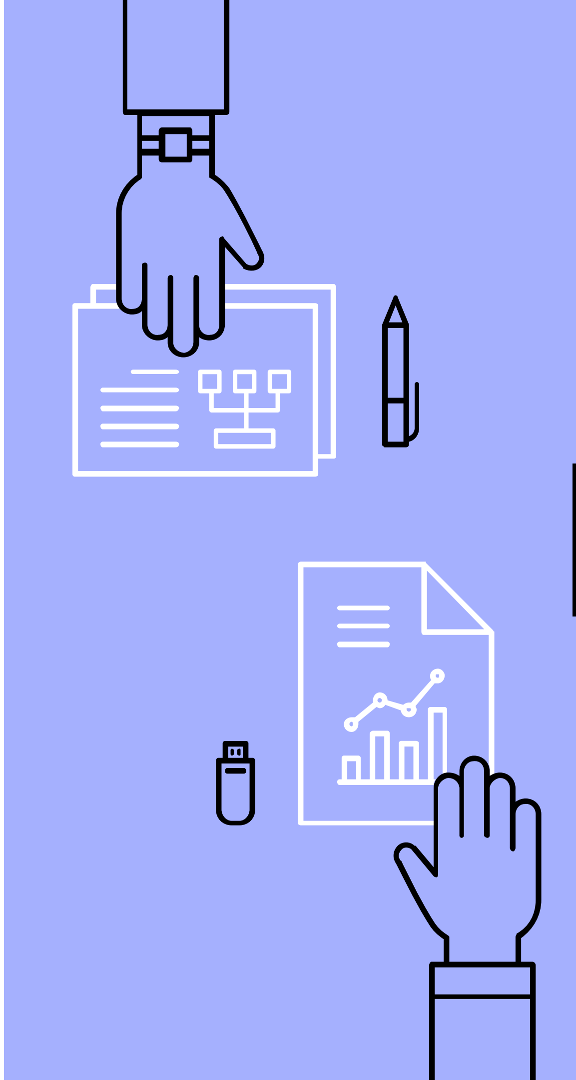




# Примеры ядер

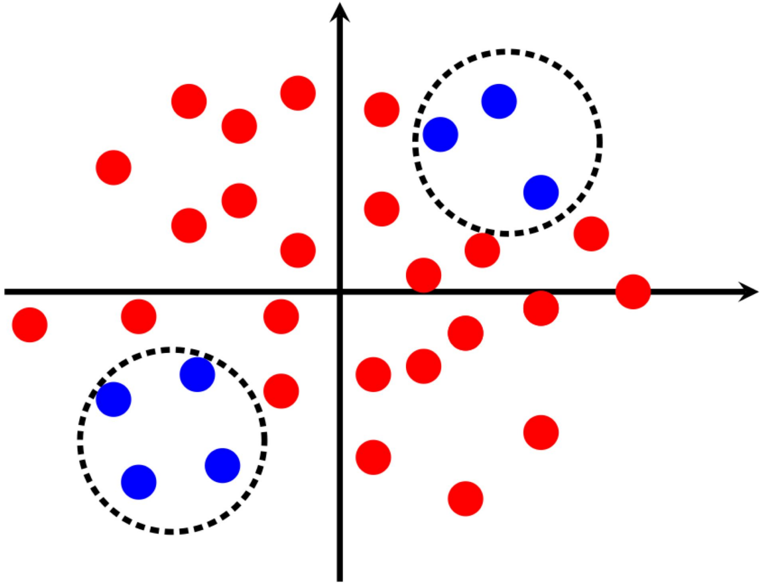
## Полиномиальное

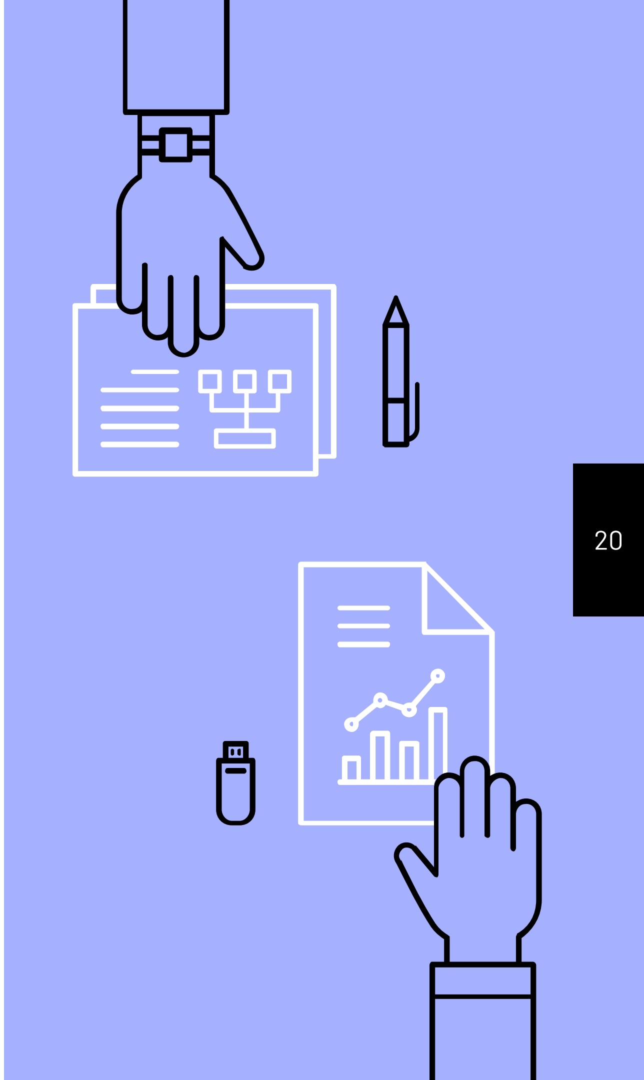




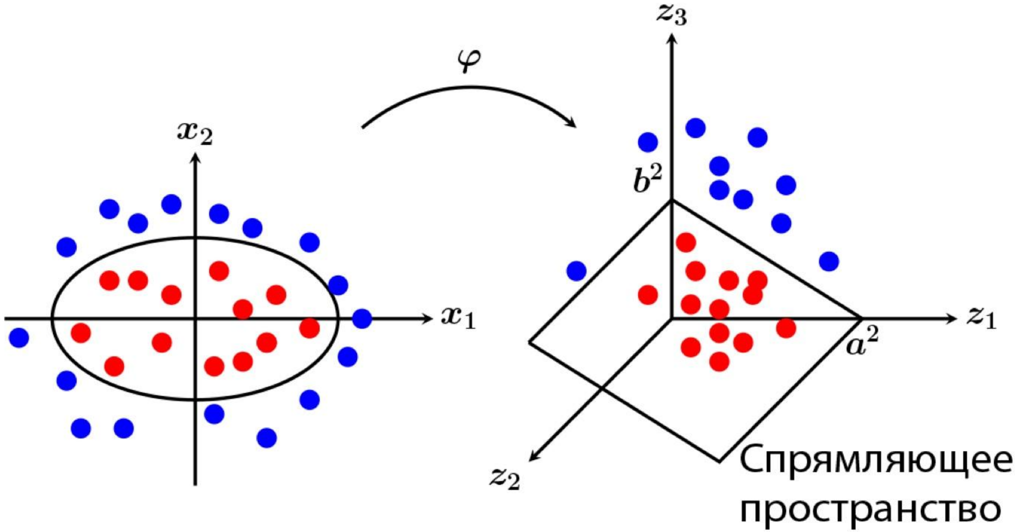
# Примеры ядер

## Радиальное





# Пример




$$\varphi : (x_1, x_2) \to (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

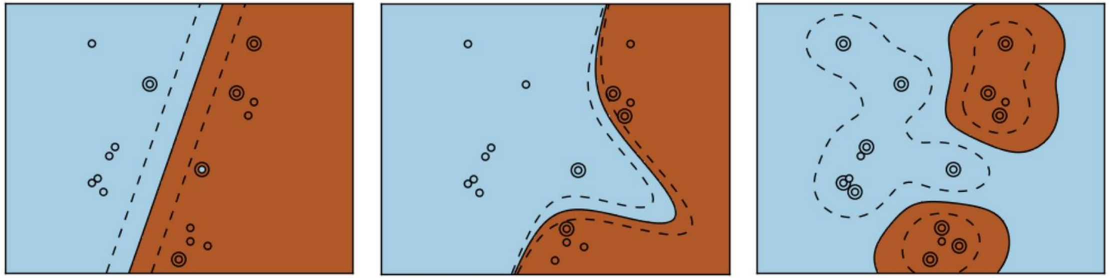
$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \;\to\; \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

# Классификация с различными ядрами

Гиперплоскость в спрямляющем пространстве соответствует нелинейной разделяющей поверхности в исходном.

Примеры с различными ядрами $K(x, x')$

| линейное | полиномиальное | гауссовское (RBF) |
| --- | --- | --- |
| $\langle x, x' \rangle$ | $(\langle x, x' \rangle + 1)^d, \quad d{=}3$ | $\exp(-\gamma \|x - x'\|^2)$ |

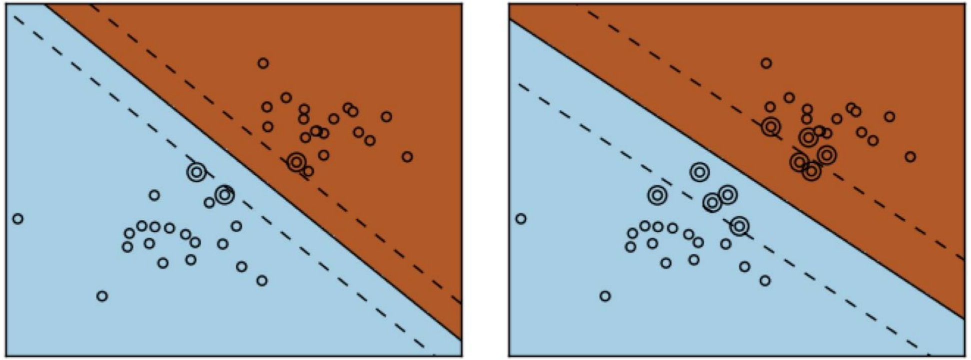


# Влияние C на решение SVM

SVM — аппроксимация и регуляризация эмпирического риска:

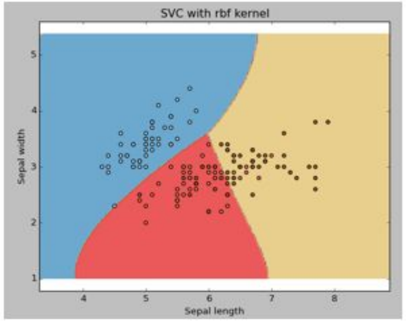
$$\sum_{i=1}^{\ell} \bigl(1 - M_i(w, w_0)\bigr)_+ + \frac{1}{2C}\|w\|^2 \to \min_{w, w_0}.$$

большой $C$ | малый $C$
слабая регуляризация | сильная регуляризация

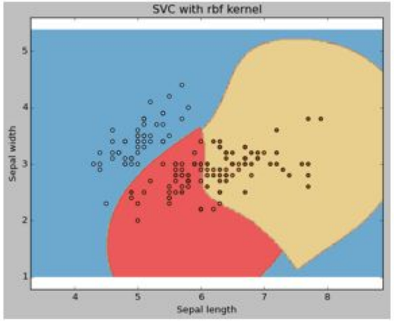


# Подбор параметров SVM

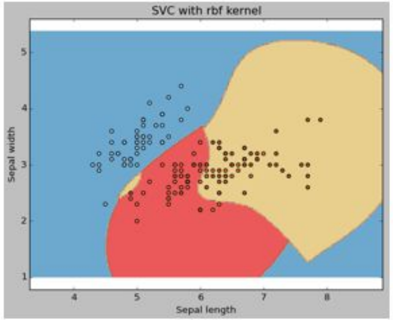
c =1



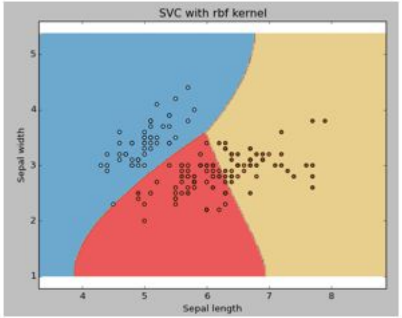

C =100



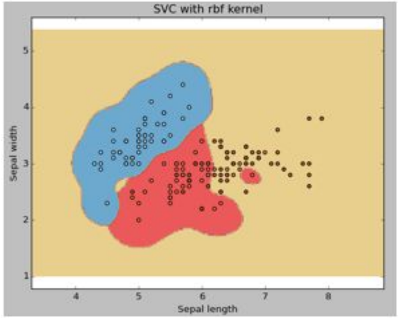

c =1000



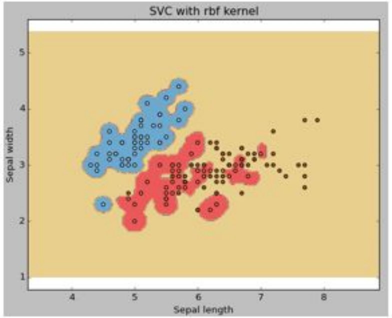

gamma =0




gamma =10




gamma =100




**C:** Предельный параметр C для ошибки. Он также контролирует расстояние между гладкой границей принятия решений и правильной классификацией точек тренировки.

**gamma** : Коэффициент ядра для 'rbf', 'poly' и 'sigmoid'. Чем выше значение гаммы, тем лучше будет задаваться как набор данных тренировки.

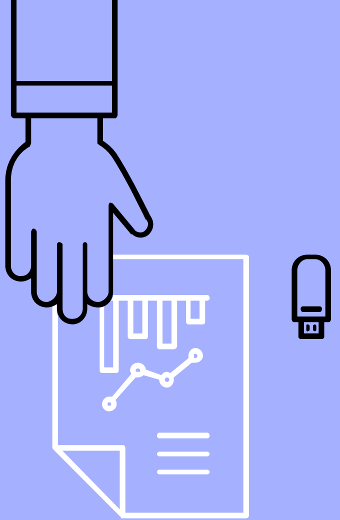
# Преимущества и недостатки SVM

**Преимущества:**

- Задача выпуклого квадратичного программирования имеет единственное решение.
- Выделяется множество опорных объектов.
- Имеются эффективные численные методы для SVM.
- Изящное обобщение на нелинейные классификаторы.

**Недостатки:**

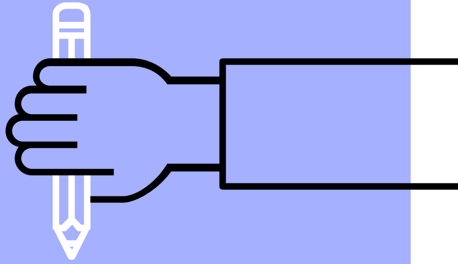
- Опорными объектами могут становиться выбросы.
- Нет отбора признаков в исходном пространстве $X$.
- Приходится подбирать константу $C$.



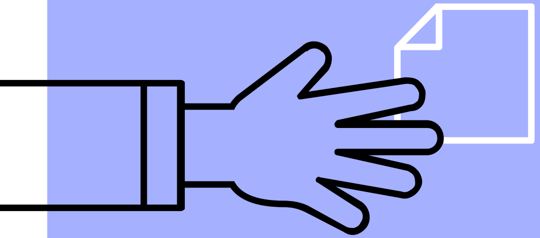
# Выводы

- Метод опорных векторов — линейный классификатор с кусочно-линейной функцией потерь (hinge loss) и $L2$ -регуляризатором
- Придуман метод был из соображений максимизации зазора между классами
- В случае линейно разделимой выборки это означает просто максимизацию ширины разделяющей полосы
- А в случае линейно неразделимой выборки просто добавляется возможность попадания объектов в полосу и штрафы за это

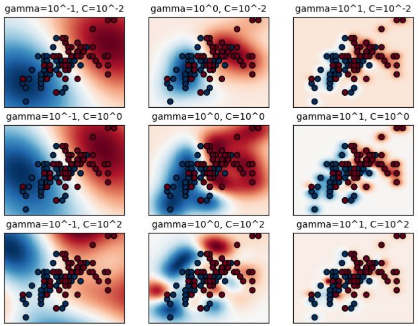
- SVM-Classification-Comparisons

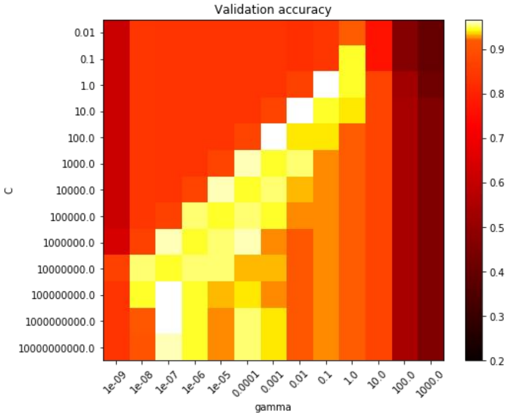


# 4.
# Примеры кода



# Подбор параметров SVM





```python
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_iris
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import GridSearchCV

class MidpointNormalize(Normalize):

    def __init__(self, vmin=None, vmax=None, midpoint=None, clip=False):
        self.midpoint = midpoint
        Normalize.__init__(self, vmin, vmax, clip)

    def __call__(self, value, clip=None):
        x, y = [self.vmin, self.midpoint, self.vmax], [0, 0.5, 1]
        return np.ma.masked_array(np.interp(value, x, y))
iris = load_iris()
X = iris.data
y = iris.target

X_2d = X[:, :2]
X_2d = X_2d[y > 0]
y_2d = y[y > 0]
y_2d -= 1

scaler = StandardScaler()
X = scaler.fit_transform(X)
X_2d = scaler.fit_transform(X_2d)

C_range = np.logspace(-2, 10, 13)
gamma_range = np.logspace(-9, 3, 13)
param_grid = dict(gamma=gamma_range, C=C_range)
cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=42)
grid = GridSearchCV(SVC(), param_grid=param_grid, cv=cv)
grid.fit(X, y)

print("The best parameters are %s with a score of %0.2f"
      % (grid.best_params_, grid.best_score_))

C_2d_range = [1e-2, 1, 1e2]
gamma_2d_range = [1e-1, 1, 1e1]
classifiers = []
for C in C_2d_range:
    for gamma in gamma_2d_range:
        clf = SVC(C=C, gamma=gamma)
        clf.fit(X_2d, y_2d)
        classifiers.append((C, gamma, clf))

plt.figure(figsize=(8, 6))
xx, yy = np.meshgrid(np.linspace(-3, 3, 200), np.linspace(-3, 3, 200))
for (k, (C, gamma, clf)) in enumerate(classifiers):
    # evaluate decision function in a grid
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # visualize decision function for these parameters
    plt.subplot(len(C_2d_range), len(gamma_2d_range), k + 1)
    plt.title("gamma=10^%d, C=10^%d" % (np.log10(gamma), np.log10(C)),
              size='medium')

    # visualize parameter's effect on decision function
    plt.pcolormesh(xx, yy, -Z, cmap=plt.cm.RdBu)
    plt.scatter(X_2d[:, 0], X_2d[:, 1], c=y_2d, cmap=plt.cm.RdBu_r,
                edgecolors='k')
    plt.xticks(())
    plt.yticks(())
    plt.axis('tight')

scores = grid.cv_results_['mean_test_score'].reshape(len(C_range),
                                                     len(gamma_range))
plt.figure(figsize=(8, 6))
plt.subplots_adjust(left=.2, right=0.95, bottom=0.15, top=0.95)
plt.imshow(scores, interpolation='nearest', cmap=plt.cm.hot,
           norm=MidpointNormalize(vmin=0.2, midpoint=0.92))
plt.xlabel('gamma')
plt.ylabel('C')
plt.colorbar()
plt.xticks(np.arange(len(gamma_range)), gamma_range, rotation=45)
plt.yticks(np.arange(len(C_range)), C_range)
plt.title('Validation accuracy')
plt.show()
```

# Анализ текстов

```python
import sklearn
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cross_validation import KFold

import sys
sys.path.append("..")
import os


class AnswerPrinter:
    def __init__(self):
        self.files = {}

    def print_answer(self, num, line, nl=False):
        if isinstance(line, float):
            line = "{:0.2f}".format(line)

        print(line)

        if num not in self.files:
            f = open(os.getcwd() + '/answers/a' + str(num) + '.txt', 'w+')
            self.files[num] = f
        else:
            f = self.files[num]

        if nl:
            line += '\n'

        f.write(str(line))


printer = AnswerPrinter()
print_answer = printer.print_answer
#from shad_util import print_answer

# 1. Загрузите объекты из новостного датасета 20 newsgroups, относящиеся к категориям "космос" и
# "атеизм" (инструкция приведена выше). Обратите внимание, что загрузка данных может занять несколько минут

newsgroups = datasets.fetch_20newsgroups(subset='all', categories=['alt.atheism', 'sci.space'])
X = newsgroups.data
y = newsgroups.target

# 2. Вычислите TF-IDF-признаки для всех текстов. Обратите внимание, что в этом задании мы предлагаем вам
# вычислить TF-IDF по всем данным. При таком подходе получается, что признаки на обучающем множестве используют
# информацию из тестовой выборки — но такая ситуация вполне законна, поскольку мы не используем значения целевой
# переменной из теста. На практике нередко встречаются ситуации, когда признаки объектов тестовой выборки известны
# момент обучения, и поэтому можно ими пользоваться при обучении алгоритма.

vectorizer = TfidfVectorizer()
vectorizer.fit_transform(X)

# 3. Подберите минимальный лучший параметр C из множества [10^-5, 10^-4, ... 10^4, 10^5] для SVM с
# линейным ядром (kernel='linear') при помощи кросс-валидации по 5 блокам. Укажите параметр random_state=241 и для
# и для KFold. В качестве меры качества используйте долю верных ответов (accuracy).

grid = {'C': np.power(10.0, np.arange(-5, 6))}
cv = KFold(y.size, n_folds=5, shuffle=True, random_state=241)
model = SVC(kernel='linear', random_state=241)
gs = grid_search.GridSearchCV(model, grid, scoring='accuracy', cv=cv)
gs.fit(vectorizer.transform(X), y)

C = gs.best_params_.get('C')

# 4. Обучите SVM по всей выборке с оптимальным параметром C, найденным на предыдущем шаге.

model = SVC(kernel='linear', random_state=241, C=C)
model.fit(vectorizer.transform(X), y)

# 5. Найдите 10 слов с наибольшим по модулю весом. Они являются ответом на это задание. Укажите их через запятую ил
# пробел, в нижнем регистре, в лексикографическом порядке.

words = vectorizer.get_feature_names()
coef = pandas.DataFrame(model.coef_.data, model.coef_.indices)
top_words = coef[0].map(lambda w: abs(w)).sort_values(ascending=False).head(10).index.map(lambda i: words[i])
top_words.sort()
print_answer(1, ','.join(top_words))
```

Одна из причин популярности линейных методов заключается в том, что они хорошо работают на разреженных данных. Так называются выборки с большим количеством признаков, где на каждом объекте большинство признаков равны нулю. Разреженные данные возникают, например, при работе с текстами. Дело в том, что текст удобно кодировать с помощью "мешка слов" — формируется столько признаков, сколько всего уникальных слов встречается в текстах, и значение каждого признака равно числу вхождений в документ соответствующего слова. Ясно, что общее число различных слов в наборе текстов может достигать десятков тысяч, и при этом лишь небольшая их часть будет встречаться в одном конкретном тексте.

# Нахождение опорных векторов

```
import os
import pandas
from sklearn.svm import SVC

import sys
sys.path.append("..")
#from shad_util import print_answer
class AnswerPrinter:
    def __init__(self):
        self.files = {}

    def print_answer(self, num, line, nl=False):
        if isinstance(line, float):
            line = "{:0.2f}".format(line)

        print( line)

        if num not in self.files:
            f = open(os.getcwd() + '/answers/a' + str(num) + '.txt', 'w+')
            self.files[num] = f
        else:
            f = self.files[num]

        if nl:
            line += '\n'

        f.write(str(line))


printer = AnswerPrinter()
print_answer = printer.print_answer
# 1. Загрузите выборку из файла svm-data.csv. В нем записана двумерная выборка (целевая переменная указана
# в первом столбце, признаки — во втором и третьем).

df = pandas.read_csv('svm-data.csv', header=None)
y = df[0]
X = df.loc[:, 1:]

# 2. Обучите классификатор с линейным ядром, параметром C = 100000 и random_state=241.
# Такое значение параметра нужно использовать, чтобы убедиться, что SVM работает с выборкой как с линейно разделимой
# При более низких значениях параметра алгоритм будет настраиваться с учетом слагаемого в функционале,
# штрафующего за маленькие отступы, из-за чего результат может не совпасть с решением классической задачи SVM для
# линейно разделимой выборки.

model = SVC(kernel='linear', C=100000, random_state=241)
model.fit(X, y)

# 3. Найдите номера объектов, которые являются опорными (нумерация с единицы). Они будут являться ответом на задание
# Обратите внимание, что в качестве ответа нужно привести номера объектов в возрастающем порядке через
# запятую или пробел. Нумерация начинается с 1.

n_sv = model.support_
n_sv.sort()
print_answer(1, ' '.join([str(n + 1) for n in n_sv]))
```

# Спасибо!

## Остались вопросы?

Контакты:

- https://vk.com/id104544842
- komleva.1999@inbox.ru