Машинное обучение и интеллектуальный анализ данных

Семинар 4

Г.А. Ососков*, О.И. Стрельцова*, Д.И. Пряхина*, Д.В. Подгайный*, А.В. Стадник*, Ю.А. Бутенко*

Государственный университет «Дубна»

***Лаборатория информационных технологий, ОИЯИ**

Дубна, Россия

Метод опорных векторов: support vector machines (SVM)

1.4. Support Vector Machines

- 1.4.1. Classification
  - 1.4.1.1. Multi-class classification
  - 1.4.1.2. Scores and probabilities
  - 1.4.1.3. Unbalanced problems
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
  - 1.4.6.1. Custom Kernels
    - 1.4.6.1.1. Using Python functions as kernels
    - 1.4.6.1.2. Using the Gram matrix
    - 1.4.6.1.3. Parameters of the RBF Kernel
- 1.4.7. Mathematical formulation
  - 1.4.7.1. SVC
  - 1.4.7.2. NuSVC
  - 1.4.7.3. SVR
- 1.4.8. Implementation details

Библиотека scikit-learn. *Machine Learning in Python*: https://scikit-learn.org/dev/

Метод опорных векторов:support vector machines (SVM)

Mathematical formulation:

Given training vectors $x_i \in \mathbb{R}^p$, i=1,..., n, in two classes, and a vector $y \in \{1, -1\}^n$, SVC solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i (w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

Its dual is

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$
$$\text{subject to } y^T \alpha = 0$$
$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ.

The decision function is:

$$\operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho\right)$$

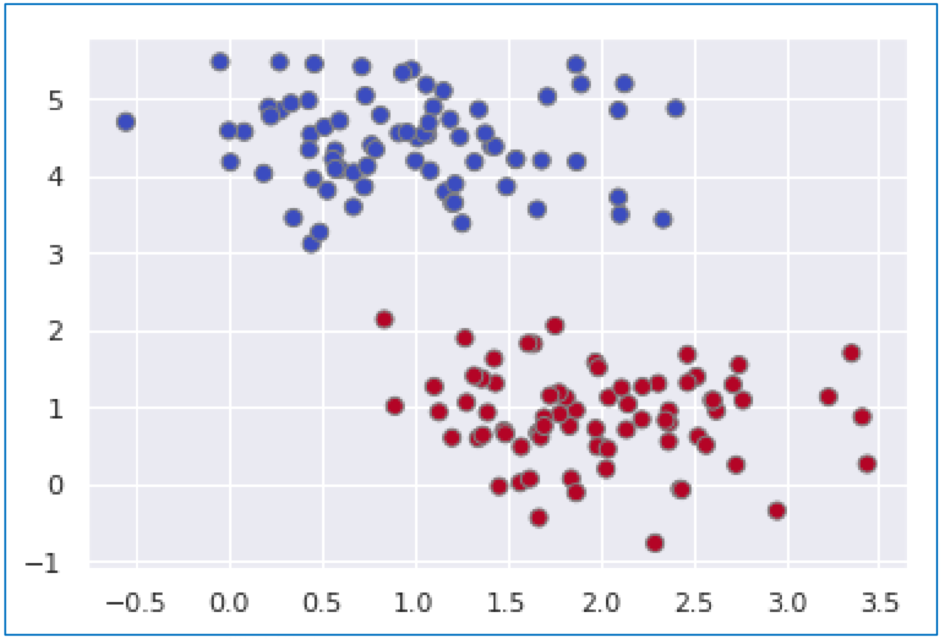
Класс CSV (C-Support Vector Classification)

Изменяем количество центров:
centers = 2 , cluster_std= 0.6

```
X, y = make_blobs(n_samples=150, centers=2, n_features=2,
                  random_state=0, cluster_std=0.6)
print(X.shape)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor="grey", s=50, cmap='coolwarm');
print(y)
```

Вызываем из библиотеки **scikit-learn** класс `CSV` ([C-Support Vector Classification](https://scikit-learn.org/stable/modules/svm.html#svm-classification))

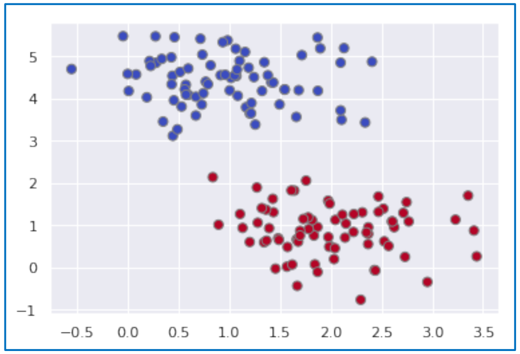
```
from sklearn.svm import SVC
```



Библиотека scikit-learn. *Machine Learning in Python*:
C-Support Vector Classification.
https://scikit-learn.org/stable/modules/svm.html#svm-classification

Класс CSV (C-Support Vector Classification): линейное ядро, большое значение параметра С

```
from sklearn.svm import SVC
```



Обучение SVM-модели,
классификатор на основе SVC:
с большим значением параметра **С:**

```
In [19]:  clf = SVC(kernel='linear', C=1E10)
          clf.fit(X,y)

Out[19]: SVC(C=10000000000.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
           kernel='linear', max_iter=-1, probability=False, random_state=None,
           shrinking=True, tol=0.001, verbose=False)
```

Результат обучения модели

Опорные векторы:

```
In [23]:  clf.support_vectors_

Out[23]: array([[0.44359863, 3.11530945],
                [2.33812285, 3.43116792],
                [0.83685684, 2.13635938]])
```

Библиотека scikit-learn. *Machine Learning in Python*: [C-Support Vector Classification](https://scikit-learn.org/stable/modules/svm.html#svm-classification).
https://scikit-learn.org/stable/modules/svm.html#svm-classification

Класс CSV (C-Support Vector Classification): линейное ядро, большое значение параметра C

```
from sklearn.svm import SVC
```

Результат обучения модели:

```
In [25]: clf.fit_status_
Out[25]: 0
```

fit_status_ : *int*
0 if correctly fitted,
1 otherwise (will raise warning)

```
In [27]: clf.coef_
Out[27]: array([[ 0.31892161, -1.91440186]])
```

coef_ :
*array, shape = [**n_class * (n_class-1) / 2, n_features**]*
Weights assigned to the features
(coefficients in the primal problem).
This is only available in the case of a linear kernel.

Опорные векторы:

```
In [23]: clf.support_vectors_
Out[23]: array([[0.44359863, 3.11530945],
                [2.33812285, 3.43116792],
                [0.83685684, 2.13635938]])
```

support_vectors_ :
*array-like, shape = [**n_SV, n_features**]*
Support vectors.

Библиотека scikit-learn. *Machine Learning in Python*: [C-Support Vector Classification](https://scikit-learn.org/stable/modules/svm.html#svm-classification).
https://scikit-learn.org/stable/modules/svm.html#svm-classification

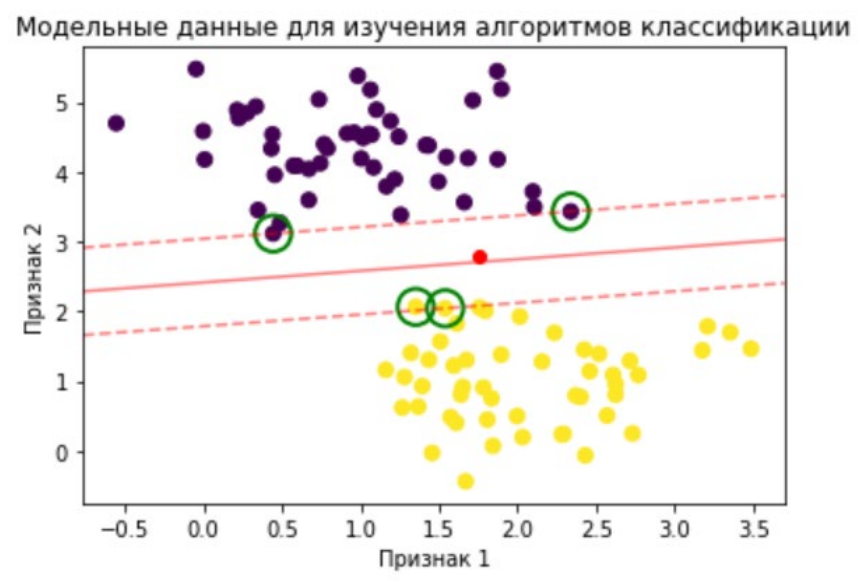
Класс CSV (C-Support Vector Classification): визуализация разделяющей гиперплоскости, отступов и опорных векторов

```
def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='r',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=300, linewidth=2,edgecolor="green", facecolors='none')
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)
```



Библиотека scikit-learn. *Machine Learning in Python*:
C-Support Vector Classification.
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

Наивный Байесовский классификатор: Gaussian Naive Bayes (GaussianNB)

`GaussianNB` implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma^2_y}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma^2_y}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.naive_bayes import GaussianNB
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(X_train, y_train).predict(X_test)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (X_test.shape[0], (y_test != y_pred).sum()))
Number of mislabeled points out of a total 75 points : 4
```

Библиотека scikit-learn. *Machine Learning in Python*: https://scikit-learn.org/stable/modules/naive_bayes.html