Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по предмету:

«Машинное обучение»

По лабораторной работе №11

«Реализация криптографических атак с помощью машинного обучения на физически неклонируемые функции»

Выполнил: Сенькович Дмитрий Сергеевич

магистрант кафедры информатики

группы №858642

Проверил: Стержанов Максим Валерьевич

доцент, кандидат технических наук

Минск 2019

**Оглавление**

[1 Введение 2](#_Toc25963548)

[2 Апроксимация ФНФ 3](#_Toc25963549)

[2.1 Постановка задачи в терминах машинного обучения 3](#_Toc25963550)

[2.2 Обучение первой модели 3](#_Toc25963551)

[2.3 Применение 3 алгоритмов 4](#_Toc25963552)

[2.4 Зависимость accuracy от N. Нужный объем данных, нужный для достижения 95% правильных классификаций 4](#_Toc25963553)

# 1 Введение

Физически неклонируемые функции - функции, результат которых зависит только от конкретного экземпляра физического объекта и не может быть повторен на другом физическом объекте даже с идентичными признаками. Такие функции могут использоваться, например, для идентификации подделок товаров.

# 2 Апроксимация ФНФ

### 2.1 Постановка задачи в терминах машинного обучения

В терминах машинного обучения задача будет следующая: обучить модель по выборке бинарных векторов и соответствующих им меток.

### 2.2 Обучение первой модели

Используем, например, SVM с Гауссовским ядром, так как, скорее всего, данные, если и зависимы, то эта зависимость сложная и нелинейная:

number\_of\_C\_samples = 10

number\_of\_gamma\_samples = 10

C\_to\_check = np.linspace(0.1, 1, number\_of\_C\_samples)

gamma\_to\_check = np.linspace(0.0001, 0.001, number\_of\_gamma\_samples)

best\_C = -1

best\_gamma = -1

best\_score = -1

curr\_model\_number = 0

for C in C\_to\_check:

for gamma in gamma\_to\_check:

classifier = SVC(C = C, gamma = gamma, kernel='rbf')

classifier.fit(x, y.flatten())

score = classifier.score(x\_val, y\_val.flatten())

if score > best\_score:

best\_score = score

best\_C = C

best\_gamma = gamma

print('------')

print('best score so far: %s' % best\_score)

print(best\_C)

print(best\_gamma)

print('------')

curr\_model\_number += 1

if curr\_model\_number % 1000 == 0:

print(curr\_model\_number)

print('best\_C %s, best\_gamma %s, best\_score %s' % (best\_C, best\_gamma, best\_score))

# best\_C 0.1, best\_gamma 0.0001, best\_score 0.665

classifier = SVC(C = 0.1, gamma = 0.0001, kernel='rbf')

classifier.fit(x, y.flatten())

predictions = classifier.predict(x\_val)

print(accuracy\_score(y\_val, predictions))

Для оценки качества алгоритма будем использовать accuracy, потому что мы, по сути, имеем задачу классификации (результаты 0 и 1). Лучший accuracy, который удалось получить - 0.605.

### 2.3 Применение 3 алгоритмов

Используем 3 алгоритма и посмотрим accuracy на них:

classifiers = [SVC(C = 0.1, gamma = 0.0001, kernel='rbf'), GradientBoostingClassifier(n\_estimators=500, max\_depth=5), QuadraticDiscriminantAnalysis()]

for classifier in classifiers:

classifier.fit(x, y)

predictions = classifier.predict(x\_val)

print(accuracy\_score(y\_val, predictions))

Лучшие accuracy, которые удалось получить: 0.67, 0.65 и 0.67.

### 2.4 Зависимость accuracy от N. Нужный объем данных, нужный для достижения 95% правильных классификаций

Построим график зависимости accuracy от количества экземпляров выборки:

samples\_counts = []

accuracies = []

for samples\_count in range(10, 800, 10):

xi = x[:samples\_count]

yi = y[:samples\_count]

xi\_val = x\_val[:samples\_count]

yi\_val = y\_val[:samples\_count]

classifier = GradientBoostingClassifier()

classifier.fit(x, y)

predictions = classifier.predict(xi\_val)

samples\_counts.append(samples\_count)

accuracies.append(accuracy\_score(yi\_val, predictions))

plt.plot(samples\_counts, accuracies)

plt.show()

Получим следующий график:

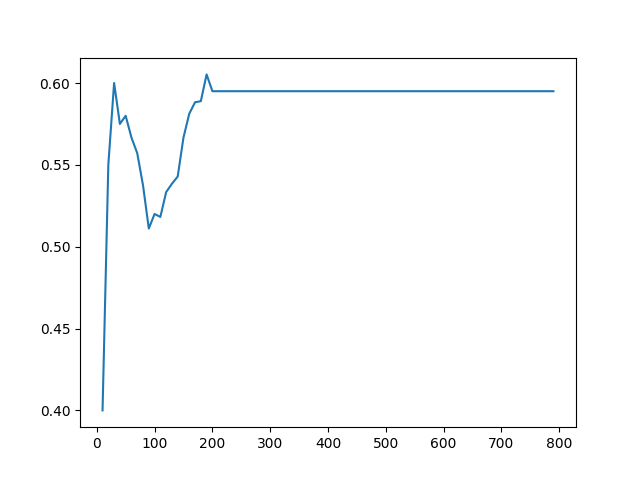


Рисунок 2.4.1. График зависимости accuracy от объема обучающей выборки

График дает ответ и на вопрос о количестве данных для достижений 95% правильных классификаций: к сожалению, наш алгоритм скорее всего, имеет high bias, поэтому с любым количеством данных из него не получится получить больше 60-65% правильных классификаций.