Лаборатория биометрических и нейросетевых технологий АО "ПНИЭИ "

Программный интерфейс библиотеки нейросетевого преобразователя биометрия-код (libnbcc.so)

Общие сведения

Назначение

Библиотека реализует преобразование биометрия-код, описанное в пакете стандартов ГОСТ Р 52633.xxxx, с учетом спецификации ТК 26 «Системы обработки информации. Криптографическая защита информации. Защита нейросетевых биометрических контейнеров с использованием криптографических алгоритмов».

Область применения

Применяется в процессе биометрической аутентификации пользователя, включая этапы:
- биометрической регистрации;
- биометрической аутентификации.

В процессе биометрической регистрации пользователь предъявляет примеры изображений одного и того же биометрического образа Свой (20-30 изображений лица). С помощью глубоких нейронных сетей или других методов, для каждого предъявленного примера вычисляется соответствующий вектор биометрических признаков. Размер вектора зависит от метода обработки. Типовые значения длины: 128, 416, 512 и т.д.
Пользователь определяет также значение выходного кода (пароль, ключ электронной подписи). Типовые значения длины: 64 бита, 256 бит, 2048 бит. Преобразователь биометрия-код автоматически настраивает свои параметры таким образом, чтобы при предъявлении примера биометрического образа Свой формировать определенный пользователем выходной код.

После завершения биометрической регистрации все временные значения, в том числе, биометрические образы, эталонный выходной код удаляются. Параметры преобразования биометрия-код размещаются в биометрическом контейнере, защищенном от попыток исследования с помощью самой биометрии и пароля.

В процессе биометрической аутентификации для каждого предъявленного пользователем примера преобразователь биометрия-код возвращает выходной код. Для примера образа Свой этот код, с высокой вероятностью, будет соответствовать эталонному значению. При предъявлении другого биометрического образа, Чужой, код на выходе будет с высокой вероятностью отличаться от эталонного в половине разрядов.

Ограничения демонстрационной версии

В настоящей демонстрационной версии установлены ограничения:
- ограничение длины соли 64 битами (влияет на стойкость к атакам подбора);
- качество предъявленных биометрических образов не оценивается;
- не реализованы механизмы обнаружения и исправления ошибок примеров Свой (важно при работе в сложных условиях или в случае нестабильной биометрии);
- не поддерживается дообучение на новых примерах уже обученного преобразователя.

Требования и совместимость

Исходный код библиотеки написан на языке C++.
Класс nbcc реализует всю функциональность нейросетевого преобразователя биометрия-код (далее – НПБК).
Он использует только стандартные типы данных и функции пространства STL
Библиотека может быть портирована под любой процессор и компилятор, в том числе низкой разрядности.

Представление входных и выходных признаков

Биометрические признаки представляются в виде плоского вектора размером число_биометрических_признаков x число_примеров. Биометрические примеры располагаются в векторах данных последовательно друг за другом.

Пример: Если каждому примеру биометрического образа соответствует 416 признаков, то для введенных пользователем 20 примеров потребуется вектор размером 8320 вещественных чисел.

Перед использованием биометрические признаки должны быть подготовлены, в частности, нормированы и центрированы с помощью базы ВсеЧужие, содержащей множество векторов биометрических признаков разных образов Чужой. Нормирование и центрирование выполняется таким образом, чтобы медиана (математическое ожидание, если закон распределения признака симметричный) любого биометрического признака, взятая по всей базе ВсеЧужие, была равна 0, а среднеквадратическое отклонение было равно 1.

Типы и константы класса nbcc

Тип TestType

Определяет тип проводимого теста по результатам обучения/защиты НПБК

P1 прогноз вероятности ошибок первого рода по установленным примерам Свой, значение [0..1.0]
P2 прогноз стойкости к атакам с использованием базы ВсеЧужие, бит
PC прогноз стойкости к атакам подбора без учета стойкости пароля, бит

Примечание:
Значения *P1*, *P2* могут использоваться в функции test только до вызова функции protect, а значение *PC* только после нее.

Тип Error

Определяет тип возвращаемой ошибки для функции.

ENo	нет ошибок
EUnknown	неизвестная ошибка
EInvalidSlot	неправильный слот (не используется в текущей реализации)
EInvalidData	неправильные входные данные
EUnsupportedState	недопустимый вызов функции для текущего состояния НПБК
EGenerate	ошибка распределения связей
EFit	ошибка обучения (настройки) НПБК
EProtect	ошибка защиты параметров НПБК
EInsufficientData	недостаточно примеров данных
ETargetP1	вероятность отказа Своему выше целевого порога
EFile	ошибка загрузка/выгрузки контейнера из потока, возможно неправильный формат данных

Примечание:
Функции в качестве признака результата возвращают значение true/false. Для получения кода ошибки и соответствующего текстового сообщения необходимо вызвать функцию error.

Тип State

Состояние НПБК

Empty	параметры не заданы (биометрический контейнер пустой)
Generated	определена структура нейронной сети и форматы данных
Trained	параметры настроены для выполнения преобразования
Protected	параметры защищены (может быть выгружен биометрический контейнер)

Методы класса nbcc

Перечисление методов преобразователя и их краткое назначение

state	текущее состояние
error	информация об ошибке
features	число входных признаков, N
length	длина выходного кода, C
power	число нейронов последнего слоя, M
clear	сброс состояния и параметров преобразователя
train	настройка параметров преобразования
protect	защита параметров преобразования
extract	извлечение выходного кода
test	тестирование настроенного преобразователя
save	сохранение биометрического контейнера
load	загрузка биометрического контейнера

Метод state

```
State state() const;
```

Получение текущего состояния НПБК.

Метод error

```
Error error(string *msg=0) const;
```

Получение кода ошибки последней вызванной функции.

Параметр:

msg	строка, в которую будет помещено текстовое сообщение с причиной ошибки

Возвращает:
код ошибки или *ENo*.

Метод features

```
int features() const;
```

Получение числа входных биометрических признаков (N), заданных во время обучения.

Возвращает:
число признаков или 0.

Метод length

```
int length() const;
```

Получение длины выходного кода.

Возвращает:
длину выходного кода в байтах или 0.

Метод power

```
int power() const;
```

Число нейронов на выходе обученной нейронной сети (M).

Возвращает:
число нейронов на выходе обученной нейронной сети или 0.

Примечание: Число нейронов определяется по результатам обучения и может быть гораздо меньше длины выходного кода.

Метод clear

```
void clear();
```

Сброс состояния НПБК к начальному (*Empty*) и удаление значений хранимых параметров.

Метод train

```
bool train(
        int nfeat,
        const vector<float> &own,
        const vector<float> &all,
        const vector<uchar> &code,
        const vector<uchar> &salt,
        float target_p1  = 0.05,
        float test_part = 0.25,
        float quality_downgrade = 0.8);
```

Настройка параметров работы НПБК.

Параметры:

nfeat	[вх.] число биометрических признаков
own	[вх.] биометрические признаки примеров Свой
all	[вх.] биометрические признаки примеров ВсеЧужие
code	[вх.] выходной код, формируемый на выходе преобразователя (1..1024 байт)
salt	[вх. опц.] случайное значение, используемое для создания таблиц связей (не должно запоминаться или восстанавливаться каким-либо способом вне НПБК)
target_p1	[вх. опц.] допустимый порог ошибки 1 рода на тестовых примерах
test_part	[вх. опц.] часть примеров, которая используется для проверки качества обучения [0,1]
quality_downgrade	[вх. опц.] коэффициент понижения качества обучающей выборки

Примечания:

1) Длина вектора own рассчитывается как число_биометрических_признаков x число_примеров. Для успешного обучения можно использовать от 10 до 50 примеров биометрического образа Свой, рекомендуется не меньше 20.

2) Длина вектора all рассчитывается как число_биометрических_признаков x число_примеров. Рекомендуется использовать от 50 до 1000 примеров, сбалансированных относительно базы ВсеЧужие.

3) В векторах own и all примеры должны быть расположены последовательно друг за другом.

Возвращает:
true, если обучение завершено успешно.
false, если обучение завершилось с ошибкой (см. error).

Метод protect

```
bool protect(const vector<uchar> &passw);
```

Защита параметров преобразователя от попыток исследования после обучения и оценки P_1, P_2.

Параметр:
passw [вх.] пароль, используемый в схеме защиты (может быть не задан)

Примечание:
Для защиты параметров преобразования без использования пароля, вводимого пользователем, необходимо передавать значения пароля по умолчанию, например, «123456».

Возвращает:
true, если защита выполнена успешно.
false, если возникла ошибка (см. error).

Метод extract

```
bool extract(const vector<float> &bio, vector<uchar> &code, const vector<uchar> &passw={});
```

Извлечение кода-ключа из контейнера.

Параметры:
bio [вх.] биометрические признаки
code [вх.] выходной код
passw [вх. опц.] пароль, если контейнер защищен на пароле

Примечание:
1) Если пароль не передается, подразумевается вызов метода для незащищенных параметров преобразования.
2) Можно передавать биометрические признаки сразу нескольких биометрических примеров. Длина результирующего вектора code в этом случае составит length() x число_примеров.

Возвращает:
true, если преобразование выполнено успешно.
false, если преобразование завершилось с ошибкой.

Метод test

```
bool test(TestType tt, const vector<float> &bio, float &p, vector<int> *h=0);
```

Оценка качества обучения и защиты НБПК по ГОСТ Р 52633.3.

Параметры:

tt	[вх.] тип теста
bio	[вх.] биометрические признаки примеров Свой (*tt = P1*); примеров ВсеЧужие (*tt = P2*) отсутствуют (*tt=PC*)
p	[вых.] значение оцениваемого параметра
h	[вх. опц.] вектор мер Хэмминга для переданных примеров, если требуется (*tt = P1*)

Возвращает:
true, если тестирование выполнено успешно, иначе, *false.*

Метод save

```
bool save(ostream &f) const;
```

Сохранение параметров защищенного преобразования биометрия-код в битовый поток.

Параметры:

f	[вх.] выходной поток для сохранения биометрического контейнера

Примечания:
1) Объем сохраняемой информации байтах можно приблизительно оценить как 4+32+C+8*N, где N – число биометрических признаков, C – длина выходного кода. Например, для 128 признаков и выходного кода длиной 32 байта, объем контейнера составит 1092 байт.

2) При необходимости, объем хранимой информации может быть уменьшен в 3-4 раза.

Возвращает:
true, если сохранение контейнера выполнено успешно, иначе, *false.*

Метод load

```
bool load(istream &f);
```

Загрузка параметров защищенного преобразования биометрия-код из битового потока.

Параметры:

f	[вх.] входной поток для загрузки биометрического контейнера

Возвращает:
true, если загрузка контейнера выполнена успешно, иначе, *false.*

Типовые последовательности вызовов методов

Последовательность вызовов во время биометрической регистрации

nbcc, train, test(tt=P1), test(tt=P2), protect, test(tt=PC), save, clear

Последовательность вызовов во время биометрической аутентификации

nbcc, load, extract, clear

Примеры использования

```
int main(int argc, char *argv[]){

    bool flag_use_err_test=false;

    string allname = "all.prm";
    string ownname = "user5-own.prm";
    string tstname = "user5-test.prm";

    int nfeat, nown, ntest, nall;
    vector<float> own, test, all;

    if (!load_prm(allname,all,nfeat,nall))
     return show_error_and_exit(nb,"error: load all.prm")
    if (!load_prm(ownname,own,nfeat,nown))
     return show_error_and_exit(nb,"error: load own.prm")
    if (!load_prm(tstname,test,nfeat,ntest))
     return show_error_and_exit(nb,"error: load test.prm")

    nbcc nb;
    vector<uchar> salt = {0,0,0,0,0,0,0,0,0};
    vector<uchar> code(65);
    std::iota(code.begin(),code.end(),1);

    //Обучить
    if (!nb.train(nfeat,own,all,code,salt,0.1,0.25,0.8))
        return show_error_and_exit(nb, "train");

    //Протестировать (при необходимости)
    float p2=0.0f, p1=0.0f;
    vector<int> hm;

    if (!nb.test(nbcc::P1,own,p1,&hm))
        return show_error_and_exit(nb, "test p1");

    cout<<"TEST"<<endl;
    cout<<"p1own="<<p1<<endl;
    cout<<"hamm="<<'\n'<<hm<<endl;

    if (!nb.test(nbcc::P1,test,p1,&hm))
        return show_error_and_exit(nb, "test p1");

    cout<<"p1tst="<<p1<<endl;;
    cout<<"hamm="<<'\n'<<hm<<endl;

    if (!nb.test(nbcc::P2,all,p2,0))
        return show_error_and_exit(nb, "test p2");

    cout<<"p2="<<p2<<endl;

    //Защитить
    vector<uchar> passw = {1,1,1,1,1,1,1};  //секретный пароль
    if (!nb.protect(passw))
        return show_error_and_exit(nb, "protect");
```

```
    //Экспортировать контейнер
    std::ofstream f1("test.nbc",std::ios::binary);
    if (!f1.is_open()) { cout<<"file not open"; return 2;}
    if (!nb.save(f1))
        return show_error_and_exit(nb,"save");
    f1.close();
    cout<<"save .nbc ok"<<endl;

    //Импортировать контейнер
    std::ifstream f2("test.nbc",std::ios::binary);
    if (!f2.is_open()) { cout<<"file not open"; return 2;}
    if (!nb.load(f2))
        return show_error_and_exit(nb,"load");
    f2.close();
    cout<<"load .nbc ok"<<endl;

    //Оценить стойкость защищенного контейнера к атакам подбора (ТК26)
    //без учета добавочной стойкости пароля
    float pc=0.0f;
    nb.test(nbcc::PC,own,pc,0);
    cout<<"pc="<<pc<<endl;

    //Извлечь выходной код для тестовой выборки
    vector<uchar> code2;

    if (!nb.extract(own,code2,passw))
        return show_error_and_exit(nb, "extract own");

    //Сравнить код с эталоном
    vector<int> cmp;
    cmp = compare_code(code,code2);
    cout<<"compare own="<<'\n'<<cmp<<endl;

    return 0;
}
```