CS900 Lab - Day 11
July 14th, 2003
Instructor: Thomas Hofmann

New Topics Covered
Pattern recognition and machine learning, Lecture: Day 11

Installing
Copy the file ocr.cpp from the course directory

```
cp /course/cs900/day11/src/ocr.cpp .
```

Compiling
You should use the gnu C++ compiler g++. Your program will be in ocr.cpp . To compile type

```
g++ ocr.cpp –o ocr
```

Problem : Implementing the Perceptron Algorithm

Support Code

In class, we have presented two simple classes to represent individual patterns and sets of patterns. Here is how the interface to the Pattern class looks like (provided in ocr.cpp)

```
class Pattern {
public:
   Pattern() { label = false; }
   bool attributes[NUM_ATTRIBUTES];
   bool label;
   void load(istream& is);
   void save(ostream& os) const ;
   void print(int ncol = 28) const ;
}
```

The important part for your work is the representation of binary attributes in the public Boolean array `attributes` and the binary label in member variable `label` (positive examples will have a “true” label, negative ones a “false” label.

The interface to the class that manages pattern sets is defined as follows

```
class PatternSet {
public:
   PatternSet(int n);
   ~PatternSet();
   void load(string s) ;
   void save(string s) const;
   Pattern& getPattern(int n);
   int getNum() const;
};
```

You will need to call the `getPattern()` method with argumenr `i` to get the i-th pattern .in the set. You can load existing files that exist in the course directory by adding the following line to your `main()` function

```
string fname = "/course/cs900/day11/data/labels0.bin";
```

You can also replace label0.bin by labels?.bin, where ? can be any number from 0-9. Each data set like this will contain 10,000 patterns and the problem is to discriminate between the digit in the filename (e.g. 0) from all other digits. You can then call the `load()` method with the `fname` string variable.

Perceptron Class

As discussed in class, you have to implement the `Perceptron` class, which is an extension of the `PatternSet` class and most crucially includes an implementation of the perceptron learning algorithm. Here is what the class looks like:

```
class Perceptron : public PatternSet {
public:
   Perceptron(int n);
   //-- perform a full learning cycle using patterns from start to end
   //-- returns number of updates performed
   int learn(int start, int end);
   //-- compute number of errors on patterns from start to end
   int error(int start, int end);
   //-- print incorrect examples one by one
   void printMistakes(int start, int end) const;
protected:
   //-- use i-th pattern for update
   void update(int i);
   //-- make prediction for i-th pattern
   bool predict(int i) const ;
   int w[NUM_ATTRIBUTES] ; //-- weight vector
   int b;                  //-- bias (threshold)
} ;
```

The public methods in boldface needs to be implemented by you. The protected methods are recommendations for internal methods that you may find helpful.

Experiments

After you have implemented the algorithm, run an experiment for every digit (data sets 0-9). Limit the number of perceptron cycles to some upper limit (such as 1000). For every experiment, use the first 9,000 patterns (0-8,999) for training and the remaining 1,000 patterns (9,000-9,999) for testing. Report the number of iterations, the training error, and the test error.