编程实验2

分类

Datasets

1. The US Postal (USPS) handwritten digit dataset: 10 classes
2. UCI spam: 2 classes

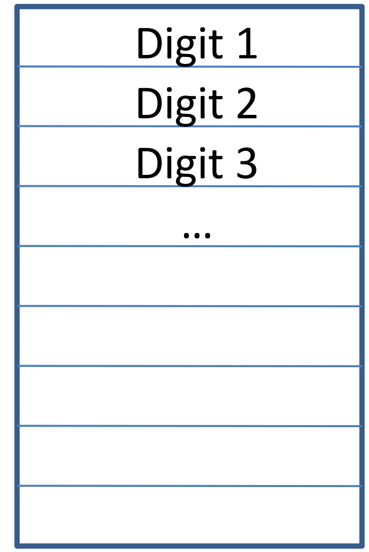
Note: You will get part of the data to train your classifiers, the rest is left for us to test your algorithms.

Dataset Representation

All the data is stored in .mat files

In Matlab, type “`load **.mat`” to load the data

Name	Value	Min	Max
Digit_Data_feature	<1000x256 double>	-1	0.9993
Digit_Data_label	<1000x1 double>	1	10
Spam_Data_feature	<1000x54 double>	0	1
Spam_Data_label	<1000x1 double>	0	1



Experiments

Algorithms	Naïve Bayes	Least Squares	SVM
Spam (2 classes)	√	√	√
USPS (10 classes)		√	√

1. Build a Naïve Bayes classifier

- Write a Matlab function "`nbayesclassifier`" that takes 5 arguments, `training`, `test`, `ytraining`, `ytest`, `k` as input, and returns a vector `ypred` as the predictions of the test data, as well as the percentage of prediction accuracy, "`accuracy`"

```
function [ypred,accuracy] = nbayesclassifier(traindata,
trainlabel, testdata, testlabel, threshold)
```

if P(spam|email)>threshold, then *spam*

2. Build a least squares classifier

- Write a Matlab function "`lsclassifier`" that takes 5 arguments, `training`, `test`, `ytraining`, `ytest`, `lambda` as input, and returns a vector `ypred` as the predictions of the test data, as well as the percentage of prediction accuracy, "`accuracy`"

$$\min_{\mathbf{w}} (X\mathbf{w} - \mathbf{y})^2 + \lambda \|\mathbf{w}\|^2$$

```
function [ypred,accuracy] = lsclassifier(traindata,
trainlabel, testdata, testlabel, lambda)
```

3. Build a support vector machine

- Write a Matlab function "`softsvm`" that takes 6 arguments, `training`, `test`, `ytraining`, `ytest`,`C`, `sigma` as input, and returns a vector `ypred` as the predictions of the test data, as well as the percentage of prediction accuracy, "`accuracy`"

```
function [ypred,accuracy] = softsvm(traindata, trainlabel, testdata,
testlabel, sigma, C)
```

when sigma=0, use linear kernel $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$,

otherwise use the RBF kernel $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2}{\sigma^2}}$

Multi-class SVM

One against All

- Inputs
  - $D_{train} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$
  - Labels $\mathbf{y}$ where $\mathbf{y} \in$ {1, …, *K*}
- Outputs:
  - A list of models $f_k(\mathbf{x}) = \mathbf{w}_k^{\top} \phi(\mathbf{x}) + b_k$ for each class
- Training:
  - For each *k* in {1 … *K*}:
  - Construct a new label vector $y_i' = 1$ where $y_i = k$; $y_i' = -1$ elsewhere
  - Given *X*, *y'* train an SVM f_k
- Testing:

$$\hat{y} = \arg\max_k f_k(\mathbf{x})$$

4. Cross Validation

- On each dataset:
  - Implement 5 fold cross validation to tune the parameters for each algorithm
  - For each algorithm:
    - Return a matrix: parameter (set) X accuracy on each fold
    - Select the parameter (set) with best average accuracy

Cross-validation

- The improved holdout method: k-fold *cross-validation*
  - Partition data into k roughly equal parts;
  - Train on all but j-th part, test on j-th part

For Naïve Bayes, select threshold from…? (e.g.: threshold=[0.5 0.6 0.7 0.75 0.8 0.85 0.9])

For least squares, select lambda from…?

$$\min_{\mathbf{w}} (X\mathbf{w} - \mathbf{y})^2 + \lambda \|\mathbf{w}\|^2$$

For SVM, select (C, sigma) value combination from:

C=[1, 10, 100, 1000], sigma?

5. Testing

Notes on building an SVM

- Make sure you understand the math
- `quadprog` in Matlab
  - Min and max objectives
- Use some simple synthetic data（模拟数据） to verify
- Use the same kernel during training and testing
- When calculating b, remember to use the same kernel!
- Check α_i to debug
  - Do they satisfy the constraints?

```
>> help quadprog
 QUADPROG Quadratic programming.
    X = QUADPROG(H,f,A,b) attempts to solve the quadratic programming
    problem:

             min 0.5*x'*H*x + f'*x   subject to:  A*x <= b
              x

    X = QUADPROG(H,f,A,b,Aeq,beq) solves the problem above while
    additionally satisfying the equality constraints Aeq*x = beq.

    X = QUADPROG(H,f,A,b,Aeq,beq,LB,UB) defines a set of lower and upper
    bounds on the design variables, X, so that the solution is in the
    range LB <= X <= UB. Use empty matrices for LB and UB if no bounds
    exist. Set LB(i) = -Inf if X(i) is unbounded below; set UB(i) = Inf if
    X(i) is unbounded above.

    X = QUADPROG(H,f,A,b,Aeq,beq,LB,UB,X0) sets the starting point to X0.
```

Calculate b in SVM

Dual optimization problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j) \quad \text{subject to} \qquad \alpha_i \geq 0, \forall i$$

$$\textstyle\sum_{i=1}^{n} \alpha_i y_i = 0$$

b can be recovered by

$$b = y_i - \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for any i that } \alpha_i \neq 0$$

$$b = y_i - \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for any i with maximal } \alpha_i$$

$$b = avg_{i:\alpha_i \neq 0} \left(y_i - \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right)$$