logistic regression model

$$P(y=1 \mid x, w) = \psi(w^t x) = \frac{1}{1+\exp(-w^t x)} = \frac{\exp(w^t x)}{1+\exp(w^t x)}$$

Fit the model by the maximum log-likelihood criterion

$$J_l(w) = \sum_{k=1}^{n} \log P(y_k \mid x_k, w)$$

$$= \sum_{k=1}^{n} \left[y_k \log P(y_k=1 \mid x_k, w) + (1-y_k) \log \left(1 - P(y_k=1 \mid x_k, w)\right) \right]$$

$$= \sum_{k=1}^{n} \left[y_k \left(w^t x_k - \log\left(1+\exp(w^t x_k)\right) \right) \right.$$

$$\left. + (1-y_k) \log \frac{1}{1+\exp(w^t x_k)} \right]$$

$$= \sum_{k=1}^{n} y_k w^t x_k - \log\left(1+\exp(w^t x_k)\right)$$

Thus, we set the derivatives of $J_l(w)$ w.r.t. the parameters to zero.

$$\frac{\partial J_l(w)}{\partial w_0} = \sum_{k=1}^{n} y_k - \frac{\exp(w^t x_k)}{1+\exp(w^t x_k)} = \sum_{k=1}^{n} y_k - P(y_k=1 \mid x_k, w) = 0$$

$$\frac{\partial J_l(w)}{\partial w_j} = \sum_{k=1}^{n} y_k x_{kj} - \frac{\exp(w^t x)\, x_{kj}}{1+\exp(w^t x_k)} = \sum_{k=1}^{n} \left(y_k - P(y_k=1 \mid x_k, w)\right) x_{kj} = 0$$

errors:

$$e_k = \left(y_k - P(y_k=1 \mid x_k, w)\right), \quad k=1, \cdots, n$$

stochastic gradient ascent ($J_l(w)$ is concave)

$$w \leftarrow w + \eta \cdot \frac{\partial}{\partial w} J_l^{(k)}(w)$$

$$= w + \eta \cdot \left(y_k - P(y_k=1 \mid x_k, w)\right) x_k$$

η is the learning rate.

Tutorial 8: Linear Classifiers

Rui Zhao

rzhao@ee.cuhk.edu.hk

Mar. 13, 2014

Outline

Support Vector Machine

Programming Toolkits

Lots of packages for SVM are available with interfaces including Matlab, Java, Python, R, Perl, Julia, etc. We list mostly used ones

- LibSVM
  `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
- Extensions of LibSVM, e.g., LibLinear, multi-class classification, support large training set, etc.
- SVM^{light}
  `http://svmlight.joachims.org/`
- Extensions of SVM^{light}, e.g., SVM^{struct}, SVM^{perf}, SVM^{rank}, etc.

LibSVM

Open source at: `https://github.com/cjlin1/libsvm`
Language: support Matlab/Octave/Python/Java Documentation:

1. Setup
2. Data format
3. Function options
4. Example in hand

LibSVM-Matlab/Octave

Setup:

1. Environment: Matlab, C/C++ compiler (MSVS in Windows or gcc in Linux/Unix)
2. Build mex binaries: type in command window mex -setup to choose a suitable compiler, and mex *.c

Function usage:

1. model = svmtrain(label_train, data_train, options);
   label_train$_{n\times 1}$: training label (dtype: double)
   data_train$_{n\times d}$: training data stacked by row (dtype: double)
   options: string of training options in LibSVM format.
2. [label_pred, acc, decision/prob] = svmpredict(label_test, data_test, model, options);
   label_test$_{n\times 1}$: testing label (dtype: double)
   data_test$_{n\times d}$: testing data stacked by row (dtype: double)
   options: string of training options in LibSVM format.

LibSVM-Matlab/Octave

Function options:

1. svmtrain()

   https://raw.github.com/cjlin1/libsvm/master/README

```
Usage: svm-train [options] training_set_file [model_file]
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC              (multi-class classification)
        1 -- nu-SVC             (multi-class classification)
        2 -- one-class SVM
        3 -- epsilon-SVR        (regression)
        4 -- nu-SVR             (regression)
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
        4 -- precomputed kernel (kernel values in training_set_file)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking : whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates : whether to train a SVC or SVR model for probability
estimates, 0 or 1 (default 0)
-wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
```

LibSVM-Matlab/Octave

Function options:

2. svmpredict()

   `https://raw.github.com/cjlin1/libsvm/master/README`

   Usage: svm-predict [options] test_file model_file output_file
   options: -b probability_estimates: whether to predict probability estimates, 0 or 1 (default 0); for one-class SVM only 0 is supported

```
% read provided data heart\_scale
[heart_label, heart_data] = libsvmread('../heart_scale');

% SVM training
model = svmtrain(heart_label, heart_data, '-c 1 -g 0.07');

% test the training data
[predict_label, accuracy, dec_values] ...
= svmpredict(heart_label, heart_data, model);
```

LibSVM-Python

Setup:

1. Environment: Python, C/C++ compiler (MSVS in Windows or gcc in Linux/Unix).
2. Build binaries
   Type *make* in terminal if in Linux/Unix, copy ../windows/libsvm.dll to C:/WINDOWS/system32/ if in Windows.
3. Use other python interface of LibSVM like scikit-learn
   `http://scikit-learn.org/stable/modules/svm.html`

Quick start:

```
# import libsvm packages
>>> from svmutil import *

# Read data in LIBSVM format
>>> y, x = svm_read_problem('../heart_scale')

# SVM training
>>> m = svm_train(y[:200], x[:200], '-c 4')

# SVM testing
>>> p_label, p_acc, p_val = svm_predict(y[200:], x[200:], m)
```