Machine Learning Term Project

103061527 李豪韋

1 Soft-margin Linear Classifier

1. Rewrite the problem:

$$\arg\max_{\alpha}(\alpha^T\mathbf{1} - \frac{1}{2}\alpha^T\tilde{K}\alpha)$$
$$\text{subject to } 0 \le \alpha \le C\mathbf{1},\ r^T\alpha = 0$$

into the standard QP form:

$$\arg\min_{x}(x^TQx + c^Tx) \text{ subject to } Gx \le h$$

$$-\alpha^T\mathbf{1} + \frac{1}{2}\alpha^T\tilde{K}\alpha = -\mathbf{1}^T\alpha + \alpha^T(\frac{1}{2}\tilde{K})\alpha = x^TQx + c^Tx$$
$$\text{where } x = \alpha,\ Q = \frac{1}{2}\tilde{K},\ c = -\mathbf{1} \text{ subject to } \alpha \le C\mathbf{1},\ -\alpha \le 0,\ r^T\alpha \le 0,\ -r^T\alpha \le 0$$
$$\Longrightarrow \begin{pmatrix} I_N \\ -I_N \\ r^T \\ -r^T \end{pmatrix}\alpha \le \begin{pmatrix} C\mathbf{1} \\ \mathbf{0} \\ 0 \\ 0 \end{pmatrix} \text{ where } G = \begin{pmatrix} I_N \\ -I_N \\ r^T \\ -r^T \end{pmatrix},\ h = \begin{pmatrix} C\mathbf{1} \\ \mathbf{0} \\ 0 \\ 0 \end{pmatrix}$$

2 SMO Implementation

The SVM classifier implemented with SMO algorithm is in *SMOClassifier.m*.

- Properties

  w, b Linear SVM parameter for normalized dataset such that $predict(X) := w^T\frac{X-\mu_X}{k} + b$

  alpha Parameter from $dual(\alpha) := \alpha(\alpha^T\mathbf{1} - \frac{1}{2}\alpha^T\tilde{K}\alpha)$ where $0 \le \alpha \le C\mathbf{1}$, $r^T\alpha = 0$

  SVs Support vectors from training set.

  r Labels corresponding to support vectors.

  C Box constrain parameter.

  kernel Kernel method option. (linear and Gaussian supported currently)

  trainMean Original mean value of each dimension of input set. (used for normalization)

  trainScale Original maximum of input set matrix. (used for normalization)

  trainingTime Training time after tuning parameters. (used for check performance)

  paramTuningTime Time taken for tuning parameters. (used for check performance)

- Member Functions

  SMOClassifier(·) Constructor.

  predict(·) To evaluate labels of testing set.

  ComputeAffine(·) Extract an affine transformation:
  Compute linear parameters for any (un)normalized dataset turned predicting from $predict(X) := w^T \frac{X-\mu_X}{k} + b$ into $predict(X) := w'^T X + b'$

- Static Functions

  train(·) To train SVM model with training set.
  Procedure: 1) Tuning C 2) Starting SMO 3) Remembering SVs and non-zero αa.

  RANSAC(·) Used for choosing an appropriate box constrain parameter.
  Objectives: 1) Reducing training time 2) Preserving accuracy
  Methods: Making the scaling of box bound roughly same with input dataset.
  Implementation: Finding the lower quartile (25百分位數) of all directional search parameters without computing all kernel values for avoiding memory issues (see line: 259-293). Therefore using RANSAC to find a 'pseudo' solution (accurate with high probability) that has high confidence with large iterations.
  i.e. making $\frac{r^{(i)}g_i - r^{(j)}g_j}{K_{ii}+K_{jj}-2K_{ij}}$ close to $(B^{(i)} - r^{(i)}\alpha_i)$ and $(r^{(j)}\alpha_j - A^{(j)})$.

  kernelEval(·) To compute a kernel matrix from multiple x's and y's, producing $K_{n(x)\times n(y)}$.

- Kernel Caching (line:107-109, 198-242 in *SMOClassifier.m*)

  Performance Significant increase for high dimensional dataset and complex kernel implementation.

  Cache size Default 100, but it is adjustable.

  Data structure Acting as a queue: if hit, extract the value; otherwise compute and save.

  Data in the cache Only save K_{ii}, K_{ij}, and K_{jj} in the cache each iteration.

- Parameter tuning

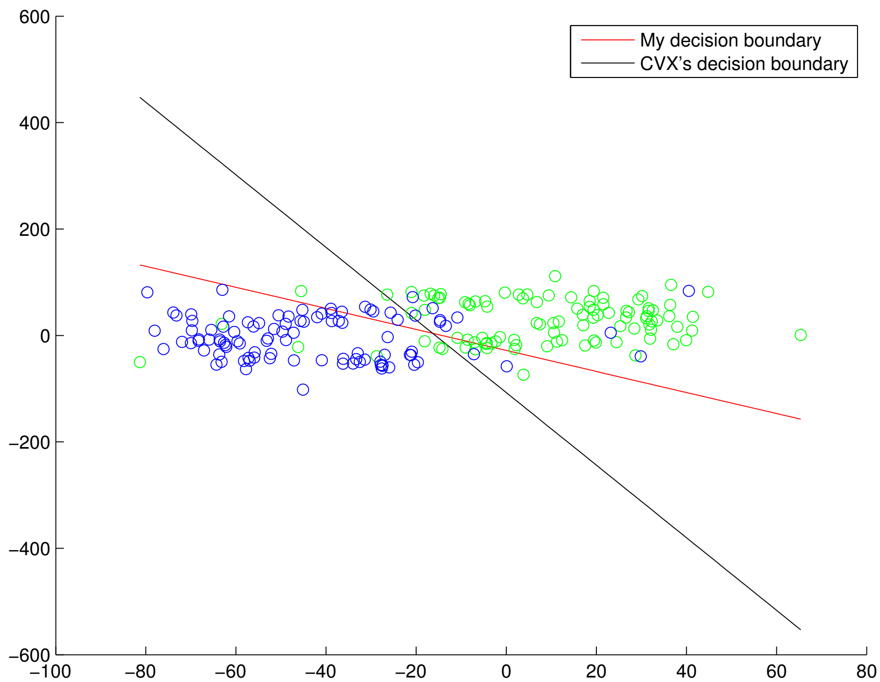
  C in box constrain making C close to $\frac{r^{(i)}g_i - r^{(j)}g_j}{K_{ii}+K_{jj}-2K_{ij}}$ because $|A^{(t)}|_{t=1}^{N}$, $|B^{(t)}|_{t=1}^{N} \leq C$.

  γ in Gaussian kernel simply setting γ close to $max(max(X))$.

  Some demo files are uploaded at my GitHub (click).

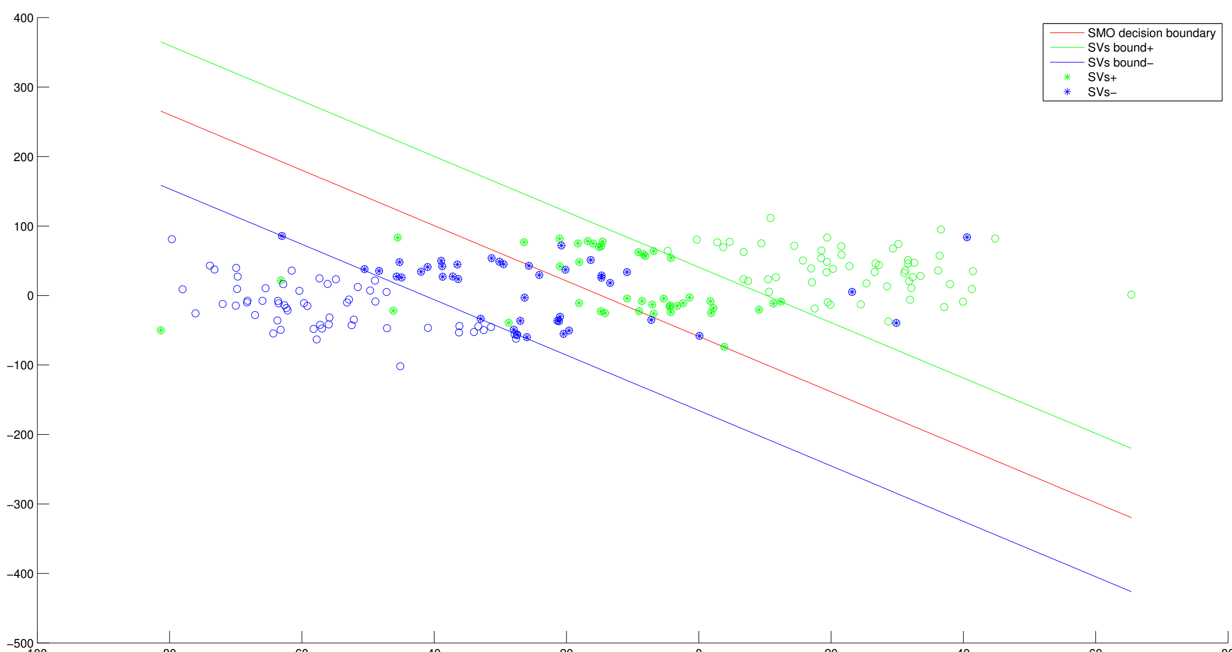
3 Results

- Decision boundary of soft-margin and SMO classifiers

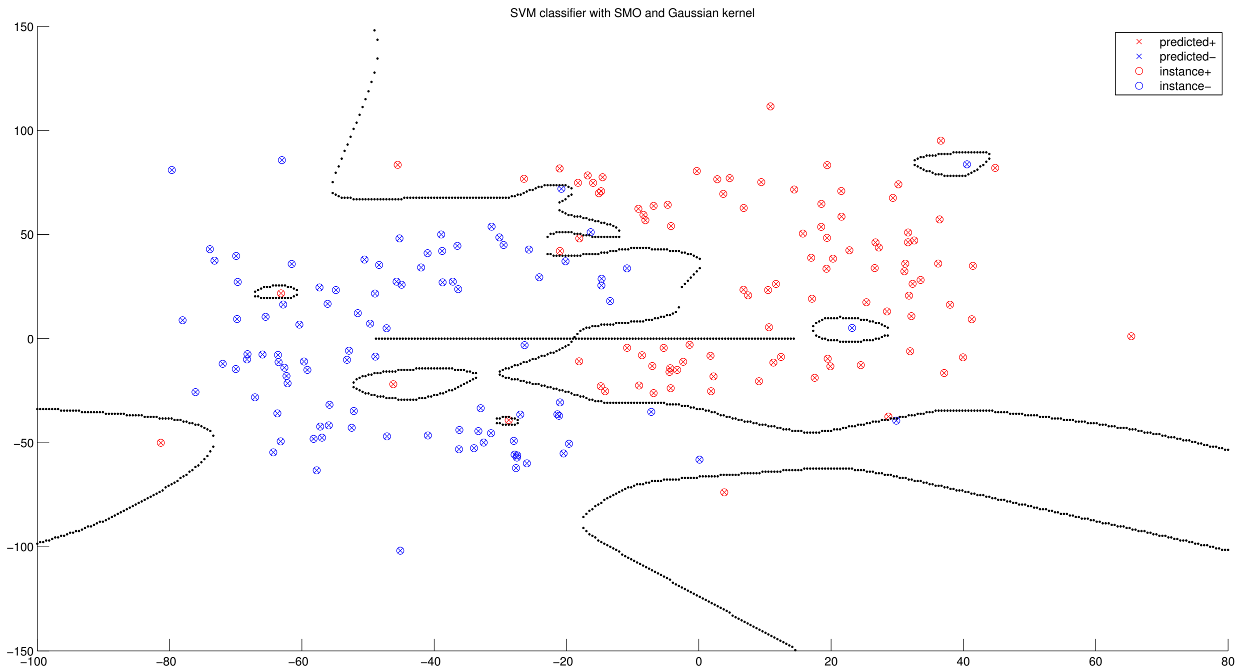


	CVX Optimization	Self-implemented
Empirical Accuracy	90%	86%
Speed	normal (1 sec.)	slow (>30 sec.)

- Decision boundary of linear SVM



- Decision boundary of Gaussian SVM



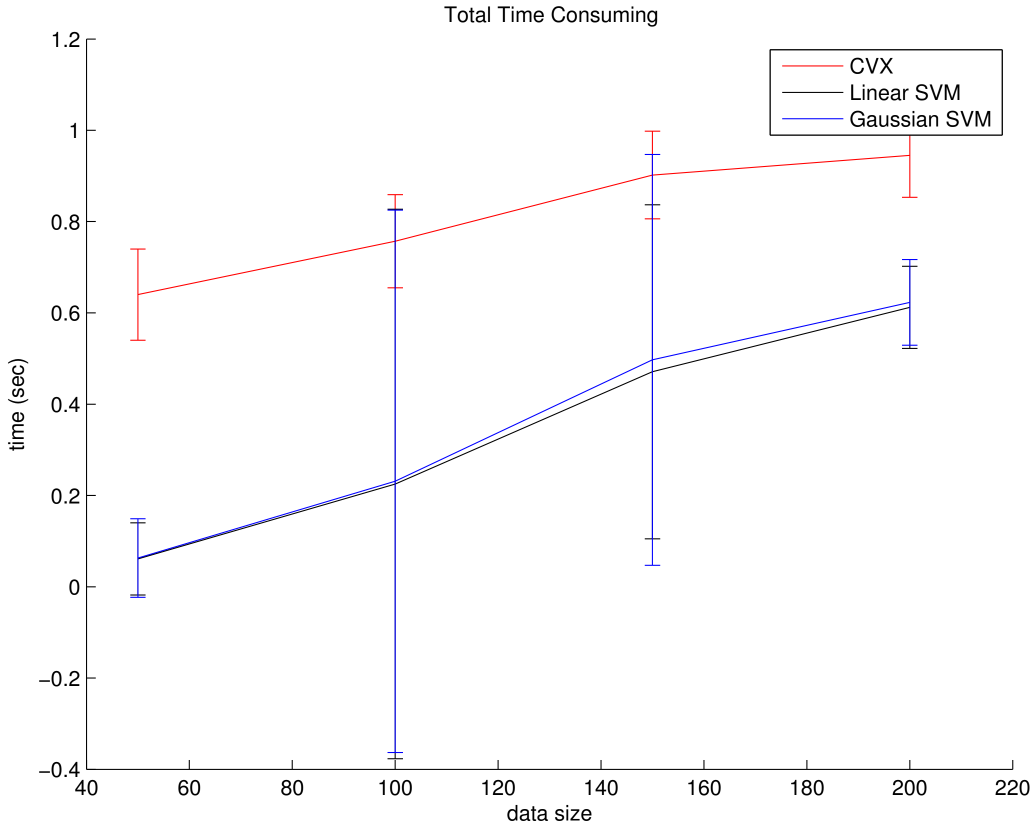
	Linear SVM	Gaussian SVM
Empirical Accuracy	90.5%	100%
Speed	fast (0.6 sec.)	normal ($\simeq$4 sec.)

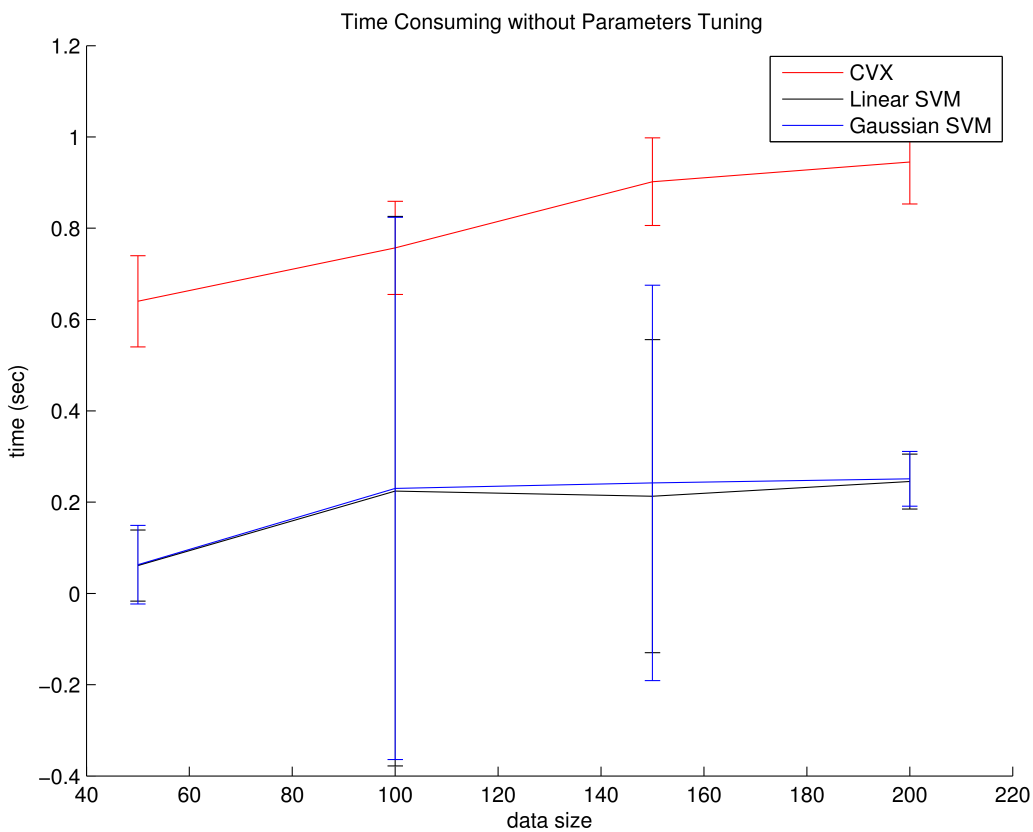
- Performance evaluated by multi-folds cross-validation

# of folds	avgAccuracy (CVX)	avgAccuracy (Linear SVM)	avgAccuracy (Gaussian SVM)
5	89.85% ± 0.944%	89.85% ± 0.474%	89.70% ± 1.059%
10	87.50% ± 1.312%	89.55% ± 0.550%	89.25% ± 0.677%
20	86.70% ± 1.798%	90.05% ± 0.438%	89.55% ± 0.158%

- Computation time with different data size

# of samples (unit: sec.)	50	100	150	200
Training (SoftMargin)	0.640 ± 0.100	0.757 ± 0.102	0.902 ± 0.096	0.945 ± 0.092
Tuning (Linear SVM)	0.000 ± 0.001	0.001 ± 0.000	0.258 ± 0.023	0.367 ± 0.030
Training (Linear SVM)	0.061 ± 0.078	0.224 ± 0.602	0.213 ± 0.343	0.245 ± 0.060
Tuning (Gaussian SVM)	0.000 ± 0.000	0.001 ± 0.000	0.255 ± 0.017	0.372 ± 0.034
Training (Gaussian SVM)	0.063 ± 0.086	0.230 ± 0.594	0.242 ± 0.433	0.251 ± 0.060


Total Time Consuming
CVX
Linear SVM
Gaussian SVM
time (sec)
data size
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
40
60
80
100
120
140
160
180
200
220


Time Consuming without Parameters Tuning
CVX
Linear SVM
Gaussian SVM
time (sec)
data size
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
40
60
80
100
120
140
160
180
200
220