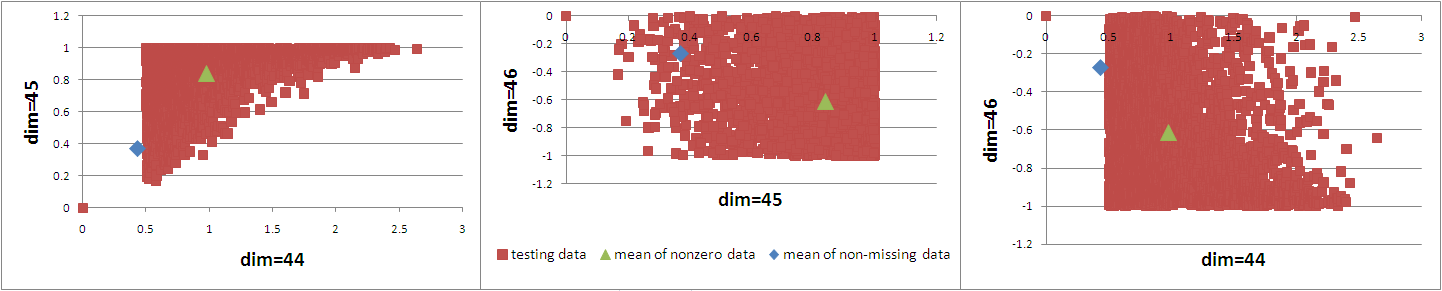
SVM

The tool we use is *LIBSVM*. We use the *easy.py* in *LIBSVM* to do our learning task, so every setup parameters (ex: scope of grid search) are the same with easy.py, and every learning parameters (ex: cost *C*) are fully determined by *easy.py*’s procedure.

We have some observations on the training data as follows:

Under a given dimension, we first define **nonzero data** = all data – missing data – zero-value data, and **non-missing data** = all data – missing data.

1. In dimension = {20, 21, 22}, there are about 68% data are missing
2. In dimension = {44, 45, 46}, there are about 28% data are missing, **43% data are zero-value data**, the others are non-zero data with a noticeable pattern. The zero-value data seems strange to us just like missing data. (as figure below)
3. In dimension = 29, there are about 58% missing data, the others are all zero-value data.
4. In dimension = 55, there are about 36% missing data, the others are all zero-value data.
5. In dimension = {47, 48, 49, 50, 51}, all the data are zero-value data.

So we design several different preprocessing strategies listed below:

1. Use original data directly as training data.
2. Ignore the values of dimension = {29, 55, 47, 48, 49, 50, 51}. i.e., using the remaining 71-dimensional data as training data.
3. Ignore the values of dimension = {29, 55, 47, 48, 49, 50, 51, 20, 21, 22, 44, 45, 46}. i.e., using the remaining 65-dimensional data as training data.
4. Ignore the values of dimension = {29, 55, 47, 48, 49, 50, 51}.

If dimension = {20, 21, 22}, for each dimension, fill the values of missing data with the mean of nonzero data.

If dimension = {44, 45, 46}, for each dimension, fill the values of **missing data and zero-value data** with the mean of **nonzero data**.

1. Ignore the values of dimension = {29, 55, 47, 48, 49, 50, 51}.

If dimension = {20, 21, 22}, for each dimension, fill the values of missing data with the mean of nonzero data.

If dimension = {44, 45, 46}, for each dimension, fill the values of **missing data** with the mean of **non-missing data**.

We use the strategies above to do learning task on Track400 and Track4000. Here is the result of **testing error** (using the half of TEST on the website as test data):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Training set | Preprocessing strategy | | | | |
| (a) | (b) | (c) | (d) | (e) |
| SMALL  (Track400) | 32.32% | 32.26% | 33.9% | 33.3% | 33.34% |
| MEDIUM  (Track4000) | 30.06% | 30.28% | 30.48% | 30.28% | 30.34% |

c

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | 8 | 8 | 8 | 8 |
| 32 | 32 | 8 | 8 | 8 |

g

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0.0078125 | 0.0078125 | 0.03125 | 0.03125 | 0.03125 |
| 0.001953125 | 0.001953125 | 0.0078125 | 0.0078125 | 0.0078125 |

Logistic regression

The tool we use is *LIBLINEAR*. All the parameters are set in default value except for cost parameter *C*, we simply do cross-validation in *C* = {1, 2, 4, 8, 16, 32} and choose the best *C* as our learning parameter.

We use the same preprocessing strategies described in the section of SVM and do learning task on Track400, Track4000 and Track40000. Here is the result of testing error (using the half of TEST on the website as test data):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Training set | Preprocessing strategy | | | | |
| (a) | (b) | (c) | (d) | (e) |
| SMALL  (Track400) | 31.84% | 31.94% | 34.42% | 32.24% | 32.18% |
| MEDIUM  (Track4000) | 29.64% | 29.64% | 30.38% | 29.64% | 29.70% |
| LARGE  (Track40000) | 29.80% | 29.76% | 29.84% | 29.76% | 29.76% |