目录

CONTENT

1.obscure

odel_training_xgboost.py

```python
'''
    main func
'''
if __name__ == "__main__":

    from sklearn.datasets import load_files
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import MultiLabelBinarizer
    from sklearn.preprocessing import LabelBinarizer
    from scipy.sparse import coo_matrix, hstack, vstack
    # from mlab.releases import latest_release as matlab
    import scipy.io as scio
    import numpy as np


    if len(sys.argv) >0:
        # DATA_NAME = sys.argv[1]
        PATH_CURR = os.path.abspath(".")
        PATH_STOP_WORDS = PATH_CURR + "\\stop_words.txt"
        PATH_DATA = PATH_CURR + "\\data\\" + 'news_content.txt'
        # PATH_TEST_RES = PATH_DATA + "_dm_test_res.mat"
        PATH_TMP = PATH_CURR +'\\source\\' + 'data_doc2vec_train
        PATH_TMP_DICT = os.path.join(PATH_TMP, "dict")
        PATH_TMP_DOC2VEC_MODEL = os.path.join(PATH_TMP, "doc2vec
        PATH_SEP_NEWS = PATH_CURR +  "\\data\\news"
        PATH_FEATURE_TMP = PATH_CURR + '\\feature_tmp\\'
        PATH_ORIGINAL_MODEL = PATH_CURR + '\\original_model\\'
        PATH_OUTPUT = PATH_CURR + '\\output\\'
```

model_training_ob.py

```python
if __name__ == "__main__" :
 if 11 - 11: IIiIiIIlli * IilI
 if 81 - 81: iiIlliiiiii + ilIlIiliiIii
 if 98 - 98: IIiIiIIlli
 if 95 - 95: ooo0Oo0 / ooo0Oo0
 from sklearn . datasets import load_files
 from sklearn . model_selection import train_test_split
 from sklearn . preprocessing import MultiLabelBinarizer
 from sklearn . preprocessing import LabelBinarizer
 from scipy . sparse import coo_matrix , hstack , vstack
 if 30 - 30: Illi + IiIIiIlIIiii / IiIIiIlIIiii % Illi . Illi
 import scipy . io as scio
 import numpy as np
 if 55 - 55: ooo0Oo0 - iIIliIIliiiiI + iIiiiIlIiIlI1 + iiIliiiiii1 % iiIII
 if 41 - 41: O00oooooo00 - iIIliIIliiiiI - iiIIIIIliiiI
 if len ( sys . argv ) > 0 :
  if 8 - 8: Ooo00oOo00o + OOoO - OOooOOo % IiIIiIlIIiii % OOooOOo * IilI
  IIIillIl1 = os . path . abspath ( "." )
  ilIIiilllll1 = IIIillIl1 + "\\stop_words.txt"
  iIIII = IIIillIl1 + "\\data\\" + 'news_content.txt'
  if 45 - 45: Illi % IIiIiIIlli - illiIiiIii
  iiiiiIiIIIii = IIIillIl1 + '\\source\\' + 'data_doc2vec_training'
  O0o0OOOOo0o0 = os . path . join ( iiiiiIiIIIii , "dict" )
  O0oOoOOOoOO = os . path . join ( iiiiiIiIIIii , "doc2vec_model" )
  oO0o0oooO0oO = IIIillIl1 + "\\data\\news"
  IiIiII1 = IIIillIl1 + '\\feature_tmp\\'
  IiiiiiIilII = IIIillIl1 + '\\original_model\\'
  OOOOOOoOo = IIIillIl1 + '\\output\\'
  if 14 - 14: IIiIiIIlli
  if 19 - 19: Ooo00oOo00o - IiIIiIlIIiii . IilI / IilI % ooo0Oo0
```

2.cython

model_training_ob.py

```
438 if __name__ == "__main__" :
439  if 11 - 11: IIiIiIIlli * IilI
440  if 81 - 81: iiIlliiiiii + ilIlIiliIlii
441  if 98 - 98: IIiIiIIlli
442  if 95 - 95: ooo0Oo0 / ooo0Oo0
443  from sklearn . datasets import load_files
444  from sklearn . model_selection import train_test_split
445  from sklearn . preprocessing import MultiLabelBinarizer
446  from sklearn . preprocessing import LabelBinarizer
447  from scipy . sparse import coo_matrix , hstack , vstack
448  if 30 - 30: Illi + IiIIilIlIiii / IiIIilIlIiii % Illi . Illi
449  import scipy . io as scio
450  import numpy as np
451  if 55 - 55: ooo0Oo0 - iIIliIIliiiiI + iIiiiIlIiIlIl + iiIliiiiiii % iiIIII
452  if 41 - 41: O00oooooo00 - iIIliIIliiiiI - iiIIIIIliiiI
453  if len ( sys . argv ) > 0 :
454   if 8 - 8: Ooo00oOo00o + OOoO - OOooOOo % IiIIilIlIiii % OOooOOo * IilI
455   IIIiiiIll = os . path . abspath ( "." )
456   ilIIiiiiiii = IIIiiiIll + "\\stop_words.txt"
457   iIIII = IIIiiiIll + "\\data\\" + 'news_content.txt'
458   if 45 - 45: Illi % IIiIiIIlli - illiIiiIii
459   iiliiIiIIIlii = IIIiiiIll + '\\source\\' + 'data_doc2vec_training'
460   O0o0OOOOo0o0 = os . path . join ( iiliiIiIIIlii , "dict" )
461   O0oOoOOOoOO = os . path . join ( iiliiIiIIIlii , "doc2vec_model" )
462   oO0o0ooo0OoO = IIIiiiIll + "\\data\\news"
463   IiIiIII = IIIiiiIll + '\\feature_tmp\\'
464   IiiiiiIilII = IIIiiiIll + '\\original_model\\'
465   OOOOOOoOo = IIIiiiIll + '\\output\\'
466   if 14 - 14: IIiIiIIlli
467   if 19 - 19: Ooo00oOo00o - IiIIilIlIiii . IilI / IilI % ooo0Oo0
```

```
static PyObject *__pyx_pf_17model_training_ob_IIlIiiiiI
  PyObject *__pyx_v_oo = NULL;
  PyObject *__pyx_v_IiIIlIlililii = NULL;
  PyObject *__pyx_v_Il = NULL;
  PyObject *__pyx_v_IIIII = NULL;
  PyObject *__pyx_r = NULL;
  __Pyx_RefNannyDeclarations
  PyObject *__pyx_t_1 = NULL;
  PyObject *__pyx_t_2 = NULL;
  PyObject *__pyx_t_3 = NULL;
  PyObject *__pyx_t_4 = NULL;
  PyObject *__pyx_t_5 = NULL;
  Py_ssize_t __pyx_t_6;
  PyObject *(*__pyx_t_7)(PyObject *);
  PyObject *__pyx_t_8 = NULL;
  PyObject *__pyx_t_9 = NULL;
  int __pyx_t_10;
  int __pyx_t_11;
  Py_ssize_t __pyx_t_12;
  __Pyx_RefNannySetupContext("IIlIiiiiiiii", 0);


  __pyx_t_1 = PySequence_List(__pyx_v_words); if (unlik
  __Pyx_GOTREF(__pyx_t_1);
  __pyx_v_oo = ((PyObject*)__pyx_t_1);
  __pyx_t_1 = 0;


  __pyx_t_2 = __Pyx_PyDict_NewPresized(0); if (unlikely
  __Pyx_GOTREF(__pyx_t_2);
  __pyx_t_3 = __Pyx_PyObject_GetAttrStr(__pyx_t_2, __py
  __Pyx_GOTREF(__pyx_t_3);
  __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;
  __pyx_t_2 = PyList_New(0); if (unlikely(!__pyx_t_2))
  __Pyx_GOTREF(__pyx_t_2);
  __pyx_t_4 = __Pyx_GetModuleGlobalName(__pyx_n_s_ilIIi
  __Pyx_GOTREF(__pyx_t_4);
  __pyx_t_5 = __Pyx_PyObject_CallOneArg(__pyx_builtin_c
  __Pyx_GOTREF(__pyx_t_5);
  __Pyx_DECREF(__pyx_t_4); __pyx_t_4 = 0;
  if (likely(PyList_CheckExact(__pyx_t_5)) || PyTuple_C
```

3.Del comment

```
__Pyx_GOTREF(__pyx_t_1);
if (PyDict_SetItem(__pyx_d, __pyx_n_s_Oo00o0OOOOOOo, __pyx_t_1) < 0) __PYX_ERR(0, 475, __py
__Pyx_DECREF(__pyx_t_1); __pyx_t_1 = 0;


__pyx_t_1 = __Pyx_GetModuleGlobalName(__pyx_n_s_ooO); if (unlikely(!__pyx_t_1)) __PYX_ERR(
__Pyx_GOTREF(__pyx_t_1);
__pyx_t_2 = __Pyx_PyObject_Dict_GetItem(__pyx_t_1, __pyx_n_s_INCREMENTAL_TRAINING); if (unl
__Pyx_GOTREF(__pyx_t_2);
__Pyx_DECREF(__pyx_t_1); __pyx_t_1 = 0;
__pyx_t_12 = (__Pyx_PyString_Equals(__pyx_t_2, __pyx_n_s_True, Py_EQ)); if (unlikely(__pyx_
__Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;
if (__pyx_t_12) {
    __pyx_t_2 = __Pyx_GetModuleGlobalName(__pyx_n_s_iIili); if (unlikely(!__pyx_t_2))
    __Pyx_GOTREF(__pyx_t_2);
    __pyx_t_14 = __Pyx_PyObject_GetAttrStr(__pyx_t_2, __pyx_n_s_read); if (unlikely(!
    __Pyx_GOTREF(__pyx_t_14);
    __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;
    __pyx_t_2 = __Pyx_PyObject_CallNoArg(__pyx_t_14); if (unlikely(!__pyx_t_2)) __PY
    __Pyx_GOTREF(__pyx_t_2);
    __Pyx_DECREF(__pyx_t_14); __pyx_t_14 = 0;
    __pyx_t_14 = __Pyx_PyObject_GetAttrStr(__pyx_t_2, __pyx_n_s_splitlines); if (unli
    __Pyx_GOTREF(__pyx_t_14);
    __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;
    __pyx_t_2 = __Pyx_PyObject_CallNoArg(__pyx_t_14); if (unlikely(!__pyx_t_2)) __PY
    __Pyx_GOTREF(__pyx_t_2);
    __Pyx_DECREF(__pyx_t_14); __pyx_t_14 = 0;
    if (PyDict_SetItem(__pyx_d, __pyx_n_s_Ilillllllilill, __pyx_t_2) < 0) __PYX_ERR(0,
    __Pyx_DECREF(__pyx_t_2); __pyx_t_2 = 0;

}
__Pyx_XDECREF(__pyx_t_9); __pyx_t_9 = 0;
__Pyx_XDECREF(__pyx_t_8); __pyx_t_8 = 0;
__Pyx_XDECREF(__pyx_t_7); __pyx_t_7 = 0;
goto __pyx_L30_try_end;
```

```
if (!__pyx_t_15) {
  __pyx_t_14 = __Pyx_PyObject_CallOneArg(__pyx_t_3, _
  __Pyx_DECREF(__pyx_t_16); __pyx_t_16 = 0;
  __Pyx_GOTREF(__pyx_t_14);
} else {
  #if CYTHON_FAST_PYCALL
  if (PyFunction_Check(__pyx_t_3)) {
    PyObject *__pyx_temp[2] = {__pyx_t_15, __pyx_t_16
    __pyx_t_14 = __Pyx_PyFunction_FastCall(__pyx_t_3,
    __Pyx_XDECREF(__pyx_t_15); __pyx_t_15 = 0;
    __Pyx_GOTREF(__pyx_t_14);
    __Pyx_DECREF(__pyx_t_16); __pyx_t_16 = 0;
  } else
  #endif
  #if CYTHON_FAST_PYCCALL
  if (__Pyx_PyFastCFunction_Check(__pyx_t_3)) {
    PyObject *__pyx_temp[2] = {__pyx_t_15, __pyx_t_16
    __pyx_t_14 = __Pyx_PyCFunction_FastCall(__pyx_t_3
    __Pyx_XDECREF(__pyx_t_15); __pyx_t_15 = 0;
    __Pyx_GOTREF(__pyx_t_14);
    __Pyx_DECREF(__pyx_t_16); __pyx_t_16 = 0;
  } else
  #endif
  {
    __pyx_t_10 = PyTuple_New(1+1); if (unlikely(!__py
    __Pyx_GOTREF(__pyx_t_10);
    __Pyx_GIVEREF(__pyx_t_15); PyTuple_SET_ITEM(__pyx
    __Pyx_GIVEREF(__pyx_t_16);
    PyTuple_SET_ITEM(__pyx_t_10, 0+1, __pyx_t_16);
    __pyx_t_16 = 0;
    __pyx_t_14 = __Pyx_PyObject_Call(__pyx_t_3, __pyx
```

Advantages

1. 代码行数686 -> 20221

2. 生成的.c源文件几乎没有可读性

3. .c文件通过gcc编译后能像python一样正常运行

3. .c文件可进一步编译成动态链接库dll文件，防止反编译

实　验

经过大量的调参，发现了一些规律：

- SPL的λ更新的步长参数，在大于0.1的情况下，与无穷大差别不大
- SPL的λ起始值有一定影响，在大部分数据集上，0.1或0.01表现较好
- 大部分细节方面的改变影响非常小，于是设置成了解释性或效率更好的值：
    1. 二次规划加上了约束：$w^T v = b$
    2. 优化精度设成与BMDR相同
    3. admm优化时过滤掉权小于0.01的样本

目前的实验设置：

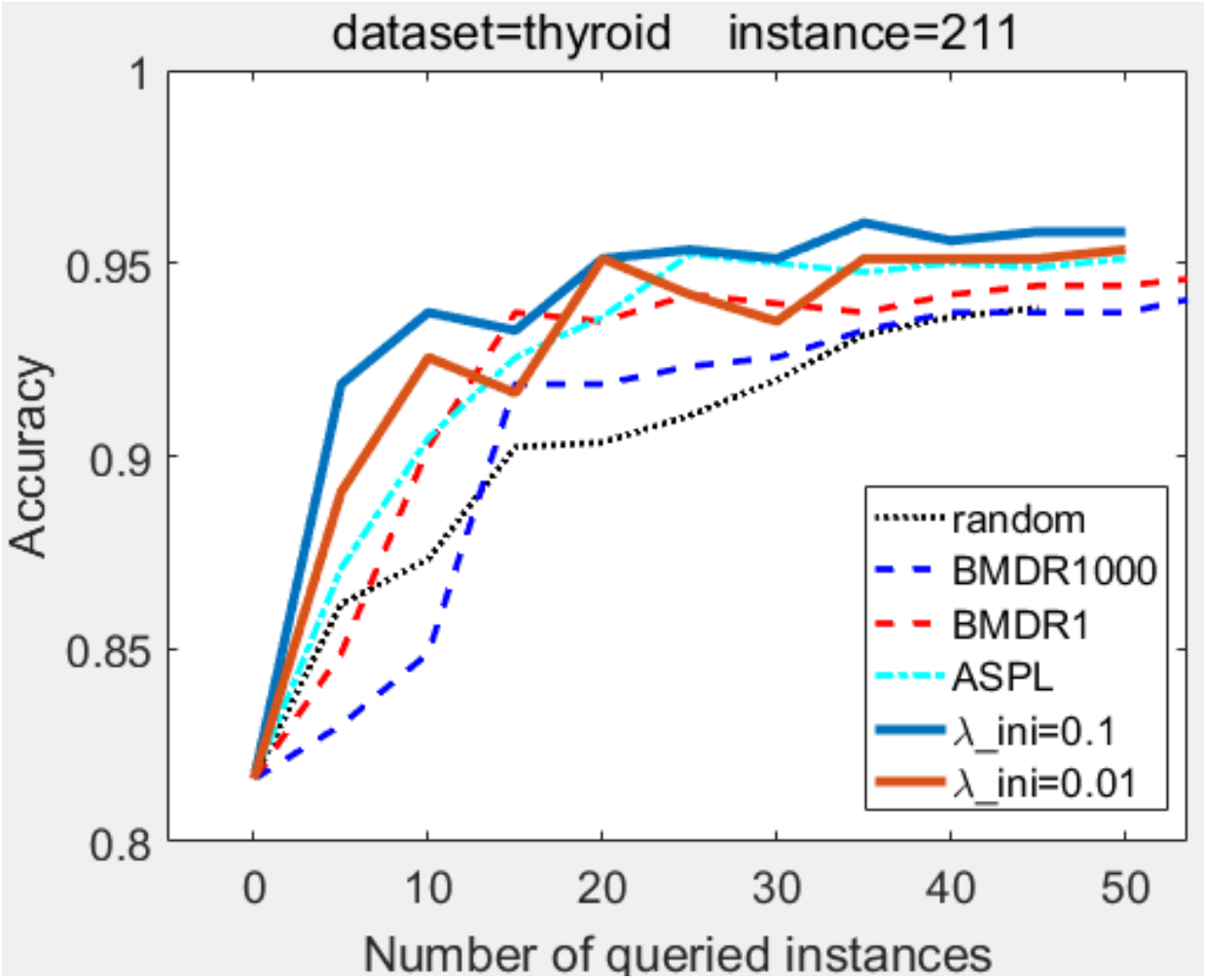
$$\min_{f,w,v} \sum_{\{x_i,y_i\}\in L} (y_i - f(x_i))^2 + \sum_{x_j\in U} v_j * w_j(\hat{y}_j - f(x_j))^2 + \lambda\left(\frac{1}{2}v^2 - v\right) + \beta MMD(S, L\cup Q) + \gamma||f||^2$$

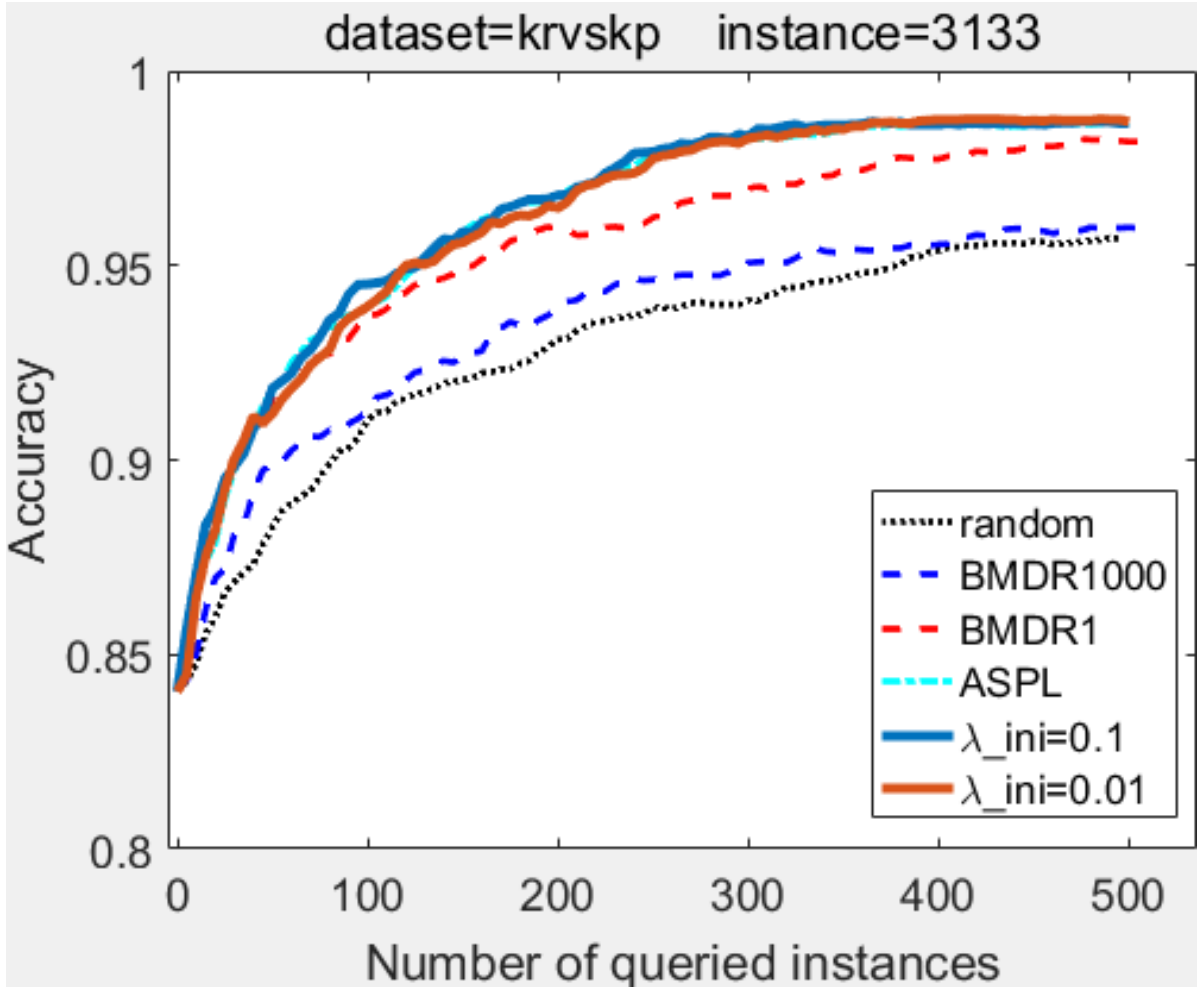
$$w, v \in [0,1]$$
$$w^T v = b$$

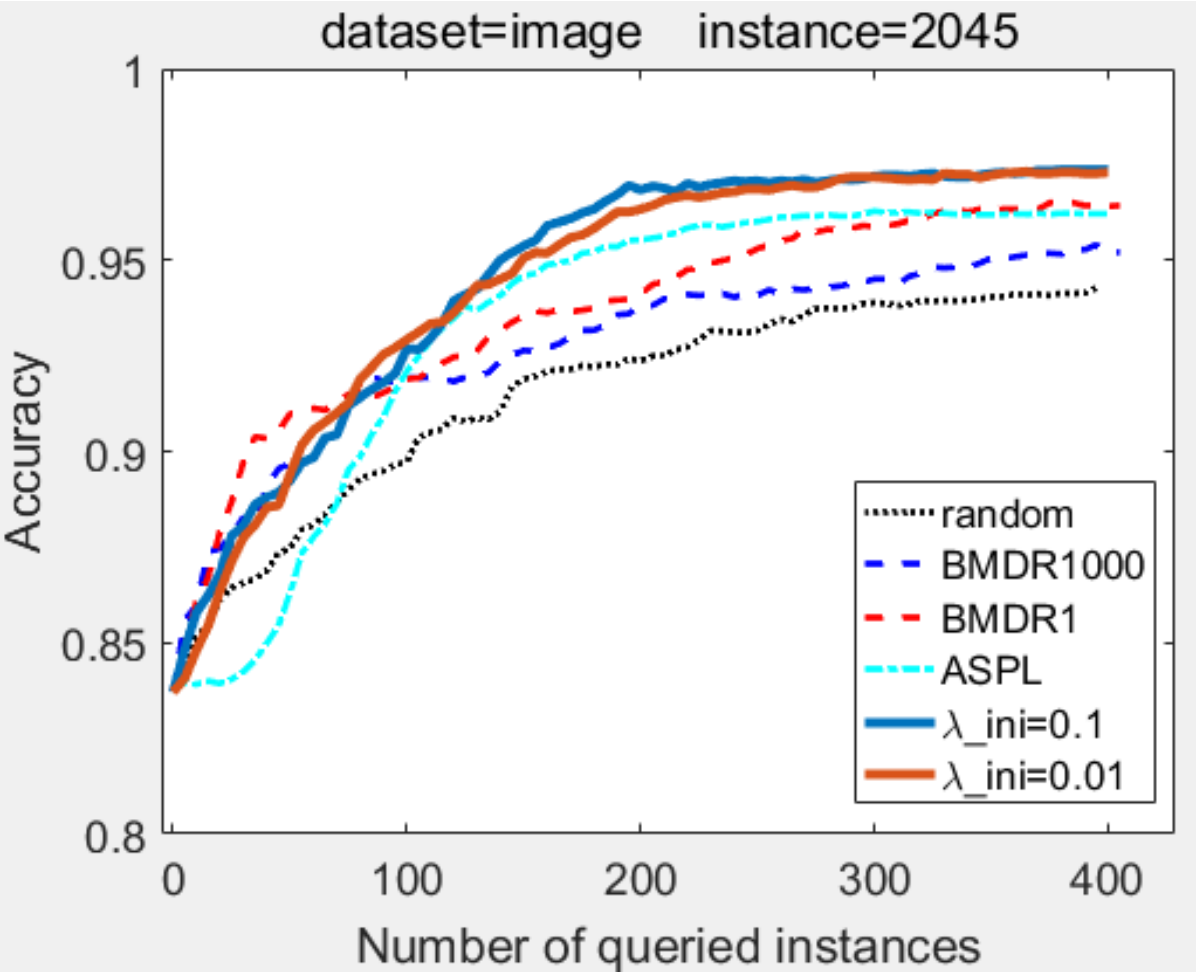
$$\lambda_{ini} = 0.01$$
$$\lambda_{pace} = [0.1, 0.01]$$
$$\beta = 0.1$$
$$\gamma = 0.1$$

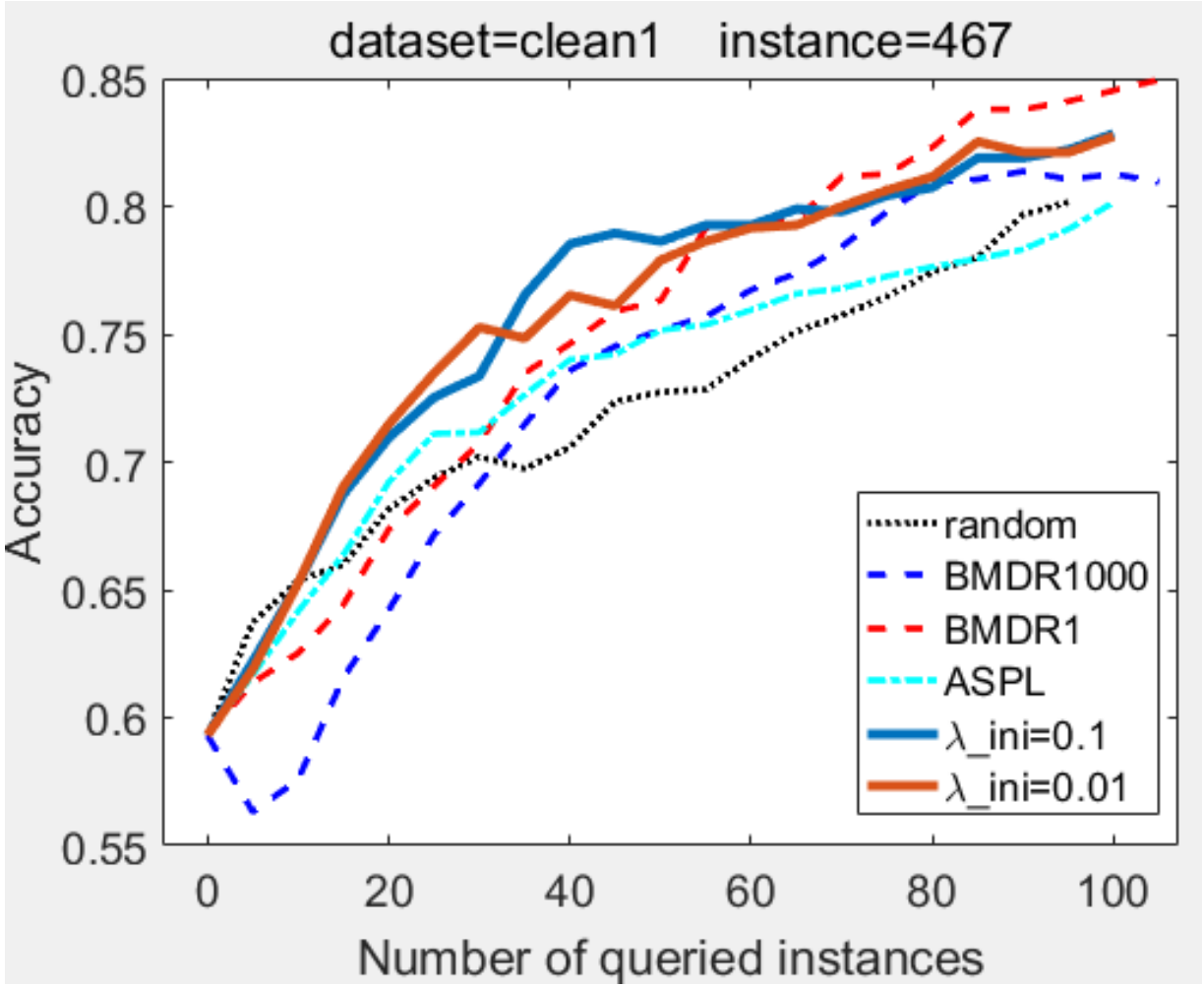
用于分类的模型也用0.1

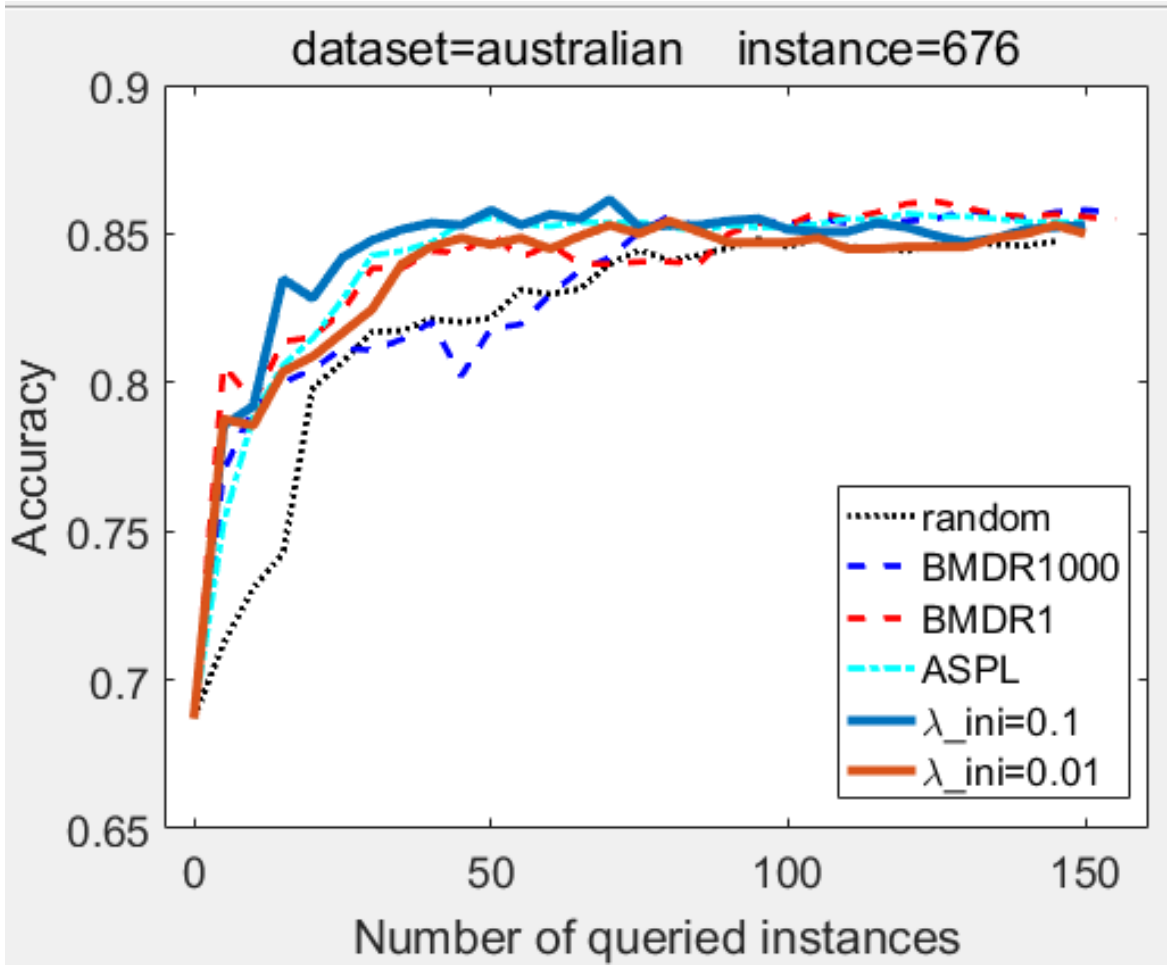
※之前的实验发现BMDR用0.1，分类模型也用0.1时效果会很好，换成1就变差了很多，而ASPL方法将此参数变化后，在大部分数据集上效果差不多
因此将分类模型的正则化参数设成0.1对BMDR与我们的方法是更有利的

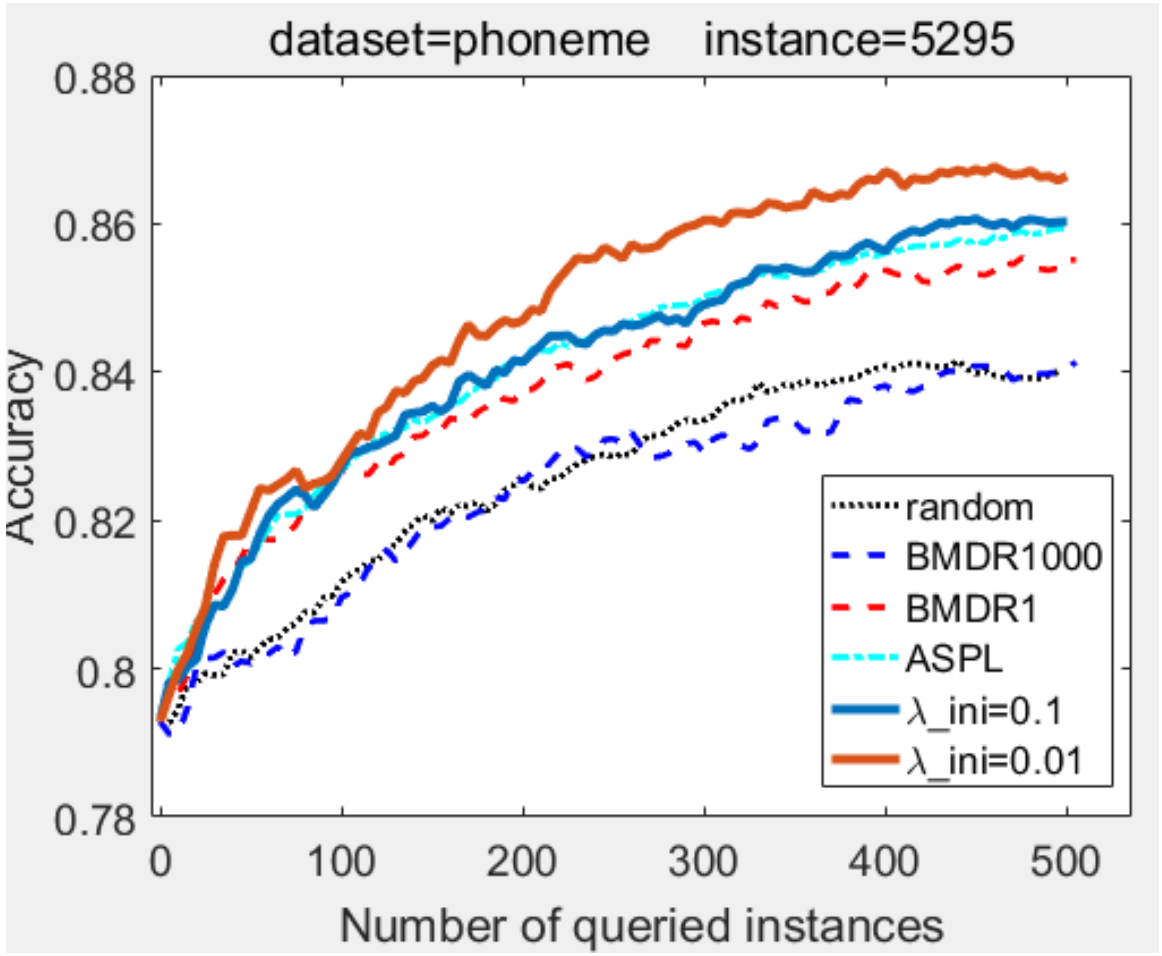

dataset=thyroid instance=211
Accuracy
Number of queried instances
random
BMDR1000
BMDR1
ASPL
λ_ini=0.1
λ_ini=0.01

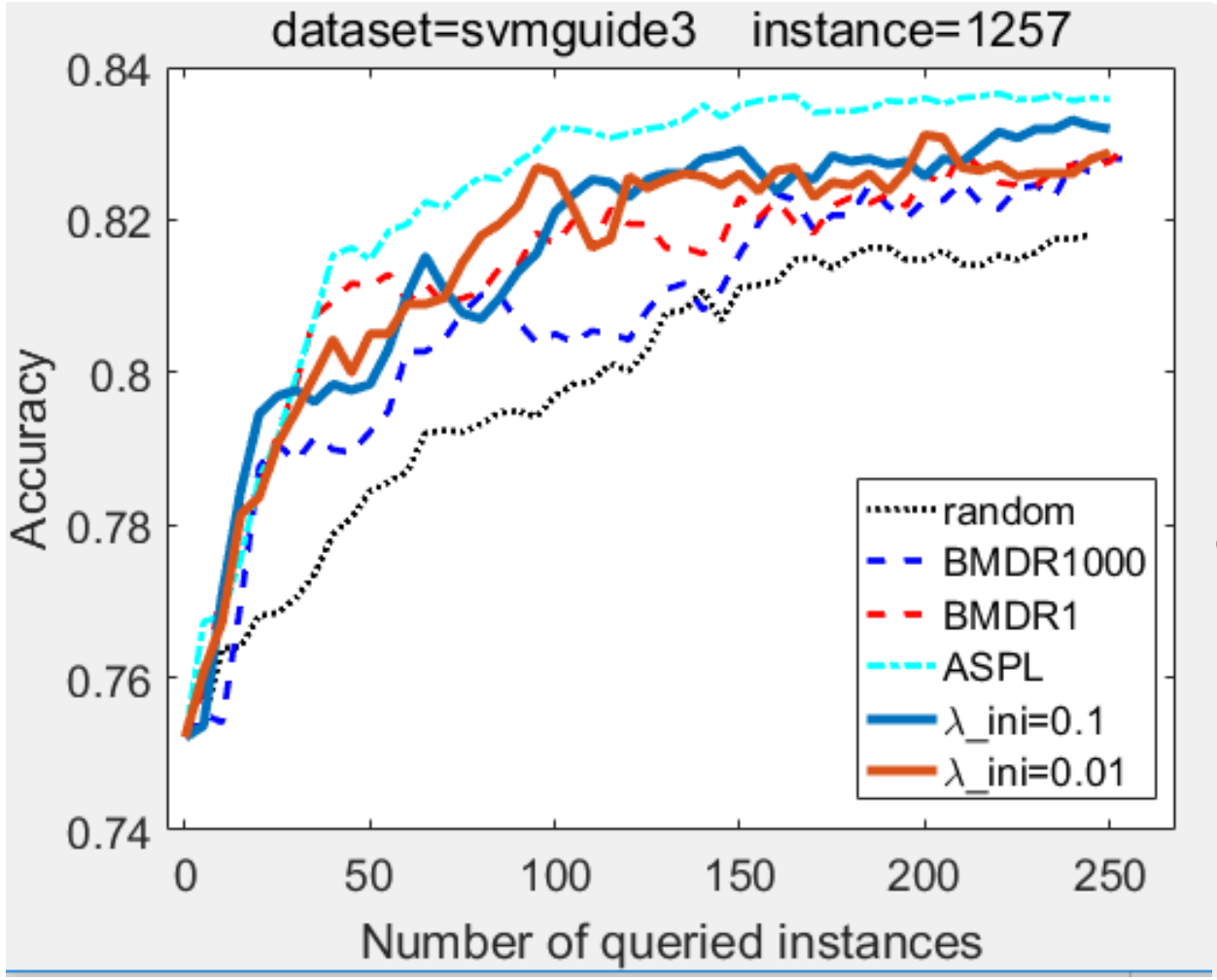

dataset=krvskp instance=3133
Accuracy
Number of queried instances
random
BMDR1000
BMDR1
ASPL
λ_ini=0.1
λ_ini=0.01

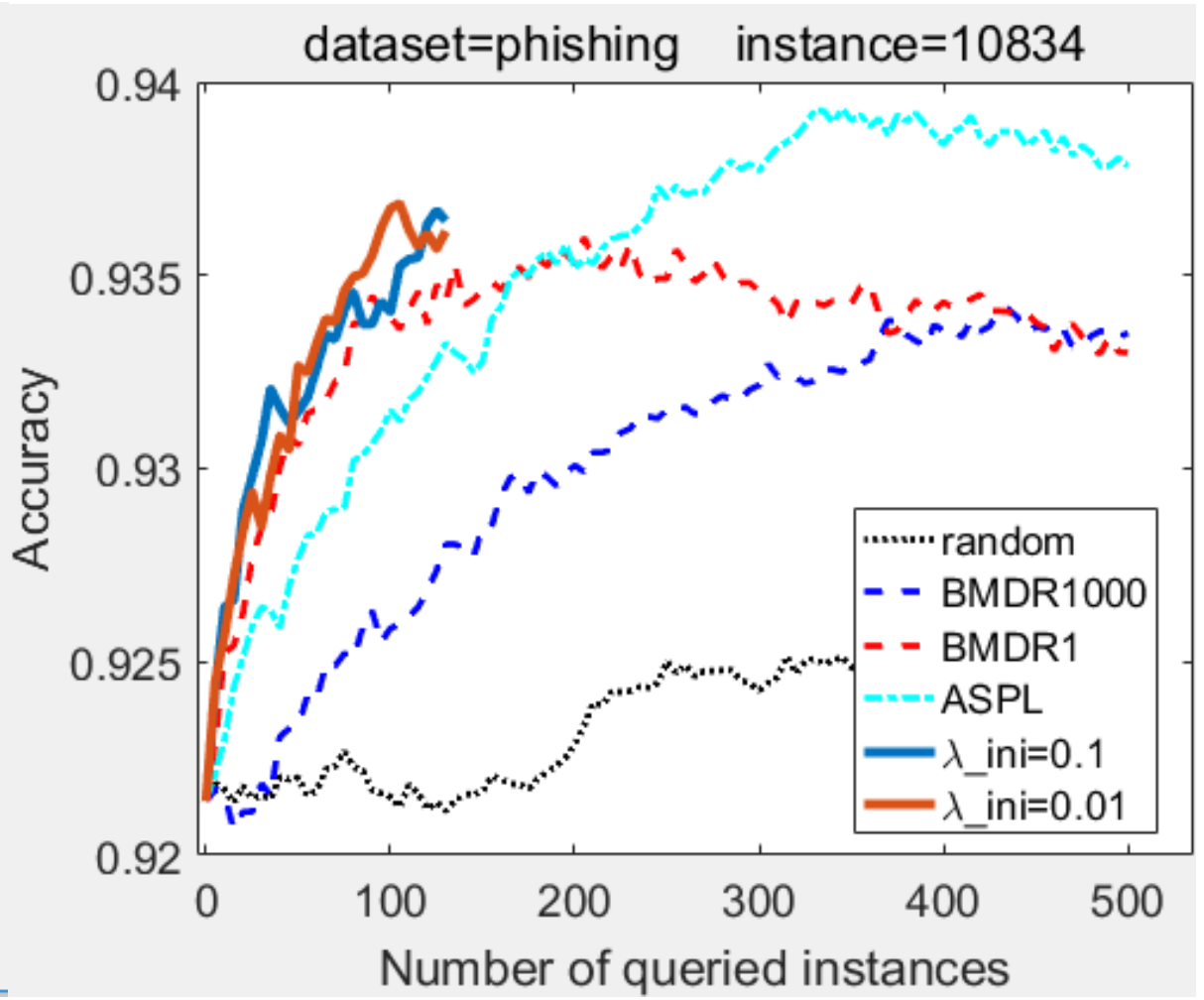

dataset=image instance=2045
Accuracy
Number of queried instances
1
0.95
0.9
0.85
0.8
0
100
200
300
400
random
BMDR1000
BMDR1
ASPL
λ_ini=0.1
λ_ini=0.01


dataset=clean1 instance=467
Accuracy
Number of queried instances
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0
20
40
60
80
100
random
BMDR1000
BMDR1
ASPL
λ_ini=0.1
λ_ini=0.01


dataset=australian instance=676
Accuracy
Number of queried instances
0.9
0.85
0.8
0.75
0.7
0.65
0
50
100
150
random
BMDR1000
BMDR1
ASPL
λ_ini=0.1
λ_ini=0.01


dataset=phoneme instance=5295
Accuracy
Number of queried instances
0.88
0.86
0.84
0.82
0.8
0.78
0
100
200
300
400
500
random
BMDR1000
BMDR1
ASPL
λ_ini=0.1
λ_ini=0.01


dataset=svmguide3 instance=1257
Accuracy
Number of queried instances
0.84
0.82
0.8
0.78
0.76
0.74
0
50
100
150
200
250
random
BMDR1000
BMDR1
ASPL
λ_ini=0.1
λ_ini=0.01


dataset=phishing instance=10834
Accuracy
Number of queried instances
0.94
0.935
0.93
0.925
0.92
0
100
200
300
400
500
random
BMDR1000
BMDR1
ASPL
λ_ini=0.1
λ_ini=0.01