CM146 Homework#3

Zhaoxing Deng, 005024802

February 28, 2018

1 VC Dimension

VC dimension of H is 3.

First we show that $H \geqq 3$. A possible shattering for $x_1 = -1$, $x_2 = 0$ $x_3 = 1$ is shown below.

$x_1$label	$x_2$label	$x_3$label	h(x)
0	0	0	$sgn(-0.001)$
0	0	1	$sgn(x - 0.001)$
0	1	0	$sgn(-x^2 + 0.001)$
0	1	1	$sgn(-x^2 + 0.5x + 1)$
1	0	0	$sgn(-x - 0.001)$
1	0	1	$sgn(x^2 - 0.001)$
1	1	0	$sgn(-x + 0.001)$
1	1	1	$sgn(0.001)$

Then we show that $H < 4$. Suppose H = 4 and we label $[1, 0, 1, 0]$ to $x = [0, 1, 2, 3]$,

then we have $h(0) > 0$,$h(1) < 0$,$h(2) > 0$,$h(3) < 0$. So $h(x)$ first comes down below x=0,

then comes up beyond x=0, then comes down again. Since$h(x)$ is continuous, we can infer

that $h(x)$ has three roots. But for $h(x) = ax^2 + bx + c$ we can have at most 2 roots.

So that's a contradiction. Therefore $H < 4$. So H = 3.

2 Kernels

$K_{\boldsymbol{\beta}} = (1 + \beta(\mathbf{x} \cdot \mathbf{z}))^3 = (1 + \beta(x_1 z_1 + x_2 z_2))^3 =<$
$1, \sqrt{3\beta}x_i, \sqrt{3}\beta x_i x_j, \beta^{\frac{3}{2}} x_i x_j x_k > \cdot < 1, \sqrt{3\beta}z_i, \sqrt{3}\beta z_i z_j, \beta^{\frac{3}{2}} z_i z_j z_k >$

$= 1+3\beta(x_1z_1+x_2z_2+x_3z_3)+3\beta^2(x_1z_1+x_2z_2+x_3z_3)^2+\beta^3(x_1z_1+x_2z_2+x_3z_3)^3$

$\phi_{\boldsymbol{\beta}}(x) =< 1, \sqrt{3\beta}x_1, \sqrt{3\beta}x_2, \sqrt{3\beta}x_3, \sqrt{3}\beta x_1^2, \sqrt{3}\beta x_1x_2, \sqrt{3}\beta x_1x_2,$

$\sqrt{3}\beta x_2^2, \beta^{\frac{3}{2}}x_1^3, \beta^{\frac{3}{2}}x_1^2x_2, \beta^{\frac{3}{2}}x_1x_2^2, \beta^{\frac{3}{2}}x_1^2x_2, \beta^{\frac{3}{2}}x_1x_2^2, \beta^{\frac{3}{2}}x_1^2x_2, \beta^{\frac{3}{2}}x_1^2x_2, \beta^{\frac{3}{2}}x_2^3 >$

The feature map is

$\phi_{\boldsymbol{\beta}}(x) =<$
$1, \sqrt{3\beta}x_1, \sqrt{3\beta}x_2, \sqrt{3\beta}x_3, \sqrt{3}\beta x_1^2, \sqrt{6}\beta x_1x_2, \sqrt{3}\beta x_2^2, \beta^{\frac{3}{2}}x_1^3, \sqrt{3}\beta^{\frac{3}{2}}x_1^2x_2, \sqrt{3}\beta^{\frac{3}{2}}x_1x_2^2, \beta^{\frac{3}{2}}x_2^3 >$

Let$\beta = 1$, then for $K = (1+\mathbf{x}\cdot\mathbf{z})^3$,

$\phi(x) =<$
$1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{3}x_3, \sqrt{3}x_1^2, \sqrt{6}x_1x_2, \sqrt{3}x_2^2, x_1^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_1x_2^2, x_2^3 >=$
$\phi_{\boldsymbol{\beta}}(x\beta^{-\frac{1}{2}})$

β can be used as a scaling factor that put more weight to the distance between data points.

3 SVM

(a) In this situation, we want to minimize $\frac{1}{2}\sqrt{w_1^2+w_2^2}$, subject to $w_1 + w_2 \geq$ 1and $-w_1 \geq 1$.

That is $w_2 \geq -w_1+1$and $w_1 \leq -1$. At $w^* =< -1, 2 >$we reach the minimum point.

The margin is $\gamma = \frac{1}{\sqrt{5}}$.

(b) We want to minimize $\frac{1}{2}\sqrt{w_1^2+w_2^2}$, subject to $w_1 + w_2 + b \geq 1$and $-w_1 - b \geq 1$.

That is $w_2 \geq 2$ and $w_1 \leq -1-b$. Let w_2 =2, $w_1 = 0$, then $w^* =< 0, 2 >$
$b^* = -1$. The margin is $\gamma = \frac{1}{2}$. The classifier changes the boundary to be a horizontal line through (0,0.5)

and the margin increases.

4 Twitter analysis using SVMs

4.1 Feature Extractions

(a) Implement extract_dictionary(...)

```
with open(infile, 'rU') as fid :
    ### ========== TODO : START ========== ###
    # part 1a: process each line to populate word_list
    for line in fid:
        wordsInLine = extract_words(line)
        for word in wordsInLine:
            if word not in word_list:
                word_list[word] = index
                index += 1

    ### ========== TODO : END ========== ###

return word_list
```

(b) Implement extract_feature_vectors(...)

```
with open(infile, 'rU') as fid :
    ### ========== TODO : START ========== ###
    # part 1b: process each line to populate feature_matrix
   lineNum = 0
   for line in fid:
       wordsInLine = extract_words(line)
       for word in wordsInLine:
           feature_matrix[lineNum][word_list[word]] = 1
       lineNum += 1
    ### ========== TODO : END ========== ###

return feature_matrix
```

(c) Split the data

```
X_train = X[0:560]
y_train = y[0:560]
X_test = X[560:630]
y_test = y[560:630]
```

(d) The dimension of feature matrix is (630, 1811)

4.2 Hyper-parameter Selection for a Linear-Kernel SVM

(a) The performance(...)

```
### ========== TODO : START ========== ###
# part 2a: compute classifier performance
score = 0
if metric == 'accuracy':
    score = metrics.accuracy_score(y_true, y_label)
elif metric == 'f1_score':
    score = metrics.f1_score(y_true, y_label)
elif metric == 'auroc':
    score = metrics.roc_auc_score(y_true, y_label)

return score
### ========== TODO : END ========== ###
```

(b) Implement cv_performance(...)

```
### ========== TODO : START ========== ###
# part 2b: compute average cross-validation performance
score = 0
for train_index, test_index in kf:
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    clf.fit(X_train, y_train)
    y_pred = clf.decision_function(X_test)
    score += performance(y_test, y_pred, metric)

score /= kf.n_folds
return score
### ========== TODO : END ========== ###
```

(c) Implement select_param_linear(...)

```
### ========== TODO : START ========== ###
# part 2: select optimal hyperparameter using cross-validation
bestScore = 0
C_Best = 0
for c in C_range:
    score = cv_performance(SVC(kernel = "linear",C = c), X, y, kf, metric)
    print "C = " + str(c) + " score = " + str(score)
    if score > bestScore:
        bestScore = score
        C_Best = c

return C_Best

### ========== TODO : END ========== ###
```

(d)

C	accuracy	F1-score	AUROC
10^{-3}	0.7089	0.8297	0.5
10^{-2}	0.7107	0.8306	0.5031
10^{-1}	0.8060	0.8755	0.7188
10^{0}	0.8146	0.8749	0.7531
10^{1}	0.8182	0.8766	0.7592
10^{2}	0.8182	0.8766	0.7592
best C	10	10	10

4.3 Test Set Performance

(a)

```
# part 3: train linear-kernel SVMs with selected hyperparameters
clf = SVC(kernel="linear", C = 10)
clf.fit(X_train, y_train)
```

(b)

```
    y_pred = clf.decision_function(X)
    print "Performance test with metric "+ str(metric) + ":"
    score = performance(y, y_pred, metric)
    return score
```

(c)

Metric	Score
Accuracy	0.7429
F1-score	0.4375
AUROC	0.6259