Buzz Prediction in Online Social Media Using Support Vector Machine with SMO, QP and MILP Optimization

Submitted To,
Dr. Gary F. Holness
Assistant professor
Department of Computer and Information Sciences
Delaware State university

Submitted By,
Abdullah-Al-Zubaer Imran
Graduate Student (MS)
Computer Science Program

Primal and Dual Formulation

The standard constrained optimization form is:

Minimize: $\frac{1}{2} w^T w$

Subject to, $y_n(w^T x_n + b) \geq 1$

For which the Lagrangian is defined as:

$$L\left(\underset{w}{\rightarrow}, \underset{\propto}{\rightarrow}, b\right) = \frac{1}{2} w^T w - \sum_{n=1}^{N} \propto_n \left(y_n(w^T x_n + b) - 1\right)$$

Then the optimization becomes:

$$min_{\underset{w}{\rightarrow}, b}\ max_{\underset{\propto}{\rightarrow}\ ;\ \propto_{i \geq 0}} L\left(\underset{w}{\rightarrow}, \underset{\propto}{\rightarrow}, b\right)$$

Therefore, the dual problem is defined as:

$$max_{\underset{\propto}{\rightarrow}\ ;\ \propto_{i \geq 0}}\ min_{\underset{w}{\rightarrow}, b}\ L\left(\underset{w}{\rightarrow}, \underset{\propto}{\rightarrow}, b\right)$$

Then,

$$L\left(\underset{w}{\rightarrow}, \underset{\propto}{\rightarrow}, b\right) = \mathrm{f}\left(\underset{w}{\rightarrow}, b\right) + \sum_{n=1}^{N} \propto_n\ g_n\left(\underset{w}{\rightarrow}, b\right)$$

Objective function Inequality constraints

Since, the Lagrangian is convex, the minimum occurs when the partial derivatives are zero.

Thus,

$$\nabla \underset{w}{\rightarrow} L = 0 \Rightarrow \underset{w}{\rightarrow} - \sum_{n=1}^{N} \propto_n y_n \underset{x_n}{\rightarrow} = 0$$

$$\text{i.e., } \underset{w}{\rightarrow} = \sum_{n=1}^{N} \propto_n y_n \underset{x_n}{\rightarrow}$$

And,
$$\frac{\delta L}{\delta b} = 0$$

$$\Rightarrow \sum_{n=1}^{N} \propto_n y_n = 0$$

Putting the values in Lagrangian equation,

$$min_{\underset{w}{\rightarrow},b} L\left(\underset{w}{\rightarrow}, \underset{\propto}{\rightarrow}, b\right) = \frac{1}{2} w^T w - \sum_{n=1}^{N} \propto_n \left(y_n (w^T x_n + b) - 1\right)$$

$= \frac{1}{2} \Sigma_{n=1}^{N} \propto_n y_n \overrightarrow{x_n} \Sigma_{m=1}^{N} \propto_n y_n \overrightarrow{x_n} - \Sigma_{n=1}^{N} \propto_n y_n \overrightarrow{x_n} \Sigma_{m=1}^{N} \propto_n y_n \overrightarrow{x_n} - b \Sigma_{n=1}^{N} \propto_n y_n + \Sigma_{n=1}^{N} \propto_n$

$= \Sigma_{n=1}^{N} \propto_n - \frac{1}{2} \Sigma_{n=1}^{N} \Sigma_{m=1}^{N} \propto_n y_n \vec{x}_n \propto_m y_m \vec{x}_m$, Which is maximization with respect to $\propto$.

$$L(\propto) = \sum_{n=1}^{N} \propto_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \propto_n y_n \underset{x}{\rightarrow}_n \propto_m y_m \underset{x}{\rightarrow}_m$$

Subject to, $\propto_n \geq 0 \; for \; n = 1, 2, \ldots\ldots, N$

and $\sum_{n=1}^{N} \propto_n y_n = 0$

Sequential Minimal Optimization(SMO)

Run information ===

Scheme:weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 2.0"

Relation: Tweet

Instances: 10000

Attributes: 78

NCD0

NCD1

NCD2

NCD3

NCD4

NCD5

NCD6

AI0

AI1

AI2

AI3

AI4

AI5

AI6

ASNA0

ASNA1

ASNA2

ASNA3

ASNA4

ASNA5

ASNA6

BL0

BL1

BL2

BL3

BL4

BL5

BL6

NAC0

NAC1

NAC2

NAC3

NAC4

NAC5

NAC6

AS0

AS1

AS2

AS3

AS4

AS5

AS6

CS0

CS1

CS2

CS3

CS4

CS5

CS6

AT0

AT1

AT2

AT3

AT4

AT5

AT6

NA0

NA1

NA2

NA3

NA4

NA5

NA6

ADL0

ADL1

ADL2

ADL3

ADL4

ADL5

ADL6

NAD0

NAD1

NAD2

NAD3

NAD4

NAD5

NAD6

ANNOTATION

Test mode:evaluate on training data

=== Classifier model (full training set) ===

SMO

Kernel used:

Poly Kernel: K(x,y) = <x,y>^2.0

Classifier for classes: Yes, No

BinarySMO

Number of support vectors: 925

Number of kernel evaluations: 110062003 (12.44% cached)

Time taken to build model: 49.17 seconds

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	9681	96.81 %
Incorrectly Classified Instances	319	3.19 %
Kappa statistic	0.8956	
Mean absolute error	0.0319	
Root mean squared error	0.1786	
Relative absolute error	10.2314 %	
Root relative squared error	45.2386 %	
Total Number of Instances	10000	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.892	0.014	0.94	0.892	0.915	0.939	Yes
0.986	0.108	0.974	0.986	0.98	0.939	No
Weighted Avg. 0.968	0.09	0.968	0.968	0.968	0.939	

=== Confusion Matrix ===

a	b	<-- classified as
1723	209	a = Yes
110	7958	b = No

Quadratic programming Optimization

Run Information: Run in Matlab using default quadratic programming function.

Matlab Code:

```
X = load('dataset.csv');

X = datasample(X, 1000);

train_targets = X(:, 78);

X = X(:, 1:77);



[rw, cl] = size(X);


X = zscore(X);


if (length(unique(train_targets)) == 2)
   Y   = 2*(train_targets>0) - 1;
else
   Y   = train_targets;
end
```

```
n =rw;

k = 3;

slack = 10;


K = ((X* X') + 1).^ k;


H = diag(Y)*K'*K*diag(Y);

A = [];

Aeq = Y';

L = zeros (n ,1);

f = -1* ones(n ,1);

b = [ ] ;

beq = 0;

u = slack* ones(n , 1);

options = optimset('Algorithm', 'interior-point-convex');


alpha = quadprog(H, f , A, b, Aeq, beq , L , u);
```

```
a_star              = (alpha.*Y)'*K';


 %Find the bias
sv = find(alpha > 0 & alpha < slack);
sv_one = zeros(n, 1);
sv_one(sv , 1 ) = 1 ;
bias = sv_one'* (Y - a_star')/sum(sv_one);



%Find support verctors
Nsv   = length(sv);
if isempty(sv),
  error('No support vectors found');
else
  disp(['Found ' num2str(Nsv) ' support vectors'])
end
```

```
Ki = (X(sv , : )* X' + 1).^k;

res = bsxfun (@plus , Ki'* (alpha(sv, : ).* Y(sv, : ) ) , bias );


res ( res >=0) = 1;

res ( res <0) = -1;


r = sum(res ~= Y) ;

r =(r/n)*100;


s =( Nsv/n)*100;
```

Mixed Integer Linear Programming (MILP)

Run Information: Run in Matlab using the default function for mixed integer linear programming.

Matlab Code:

```
X = load('dataset.csv');

X = datasample(X, 1000);

train_targets = X(:, 78);

X = X(:, 1:77);



[rw, cl] = size(X);


X = zscore(X);


if (length(unique(train_targets)) == 2)
   Y   = 2*(train_targets>0) - 1;
else
   Y   = train_targets;
end


n =rw;

k = 1;

slack = 1;
```

```
tic;
K = ((X* X') + 1).^ k;
H = diag(Y)*K'*K*diag(Y);
F = ones(n,1);
f = F'*H;
dvars = zeros(n,1);
pos = find(Y==1);
dvars(pos,:) = 1;



xvars = 1:2*n;
zvar = 2*n+1;

lb = zeros(2*n+1,1);
ub = ones(n,1);
ub(zvar) = Inf;

M = 150;
m = 100;
A = zeros(1,n);
A(dvars) = 1;
b = zeros(2,1);
b(1) = M;
b(2) = -m;
```

```
Aeq = zeros(1,2*n+1); % Allocate Aeq matrix
Aeq(xvars) = 1;
beq = 1;

alpha = intlinprog(f,dvars,A,b,Aeq,beq,lb,ub);

a_star              = (alpha.*Y)'*K';

 %Find the bias
sv = find(alpha > 0 & alpha < slack);
sv_one = zeros(n, 1);
sv_one(sv , 1 ) = 1 ;
bias = sv_one'* (Y - a_star')/sum(sv_one);


%Find support verctors
Nsv       = length(sv);
if isempty(sv),
   error('No support vectors found');
else
   disp(['Found ' num2str(Nsv) ' support vectors'])
end
```

```
Ki = (X(sv , : )* X' + 1).^k;

res = bsxfun (@plus , Ki'* (alpha(sv, : ).* Y(sv, : ) ) , bias );


res ( res >=0) = 1;

res ( res <0) = -1;


r = sum(res ~= Y) ;

r =(r/n)*100;


s =( Nsv/n)*100;
```

Comparison of SMO, QP and MILP

- Sequential Minimal Optimization is the fastest method than the other two to train the Support Vector machine.
- The rate of misclassification decreases almost for all three methods with increase in polynomial degree of kernel function.
- Quadratic programming has higher accuracy than mixed integer programming.
- The increase in exponent of polynomial kernel keeps the methods waiting longer to train the Support Vector Machine
- The increase in slack terms for quadratic programming problem and mixed integer linear programming problem reduces the number of misclassified data.
- From the following figures we can see the differences in misclassification rate for different optimization methods.

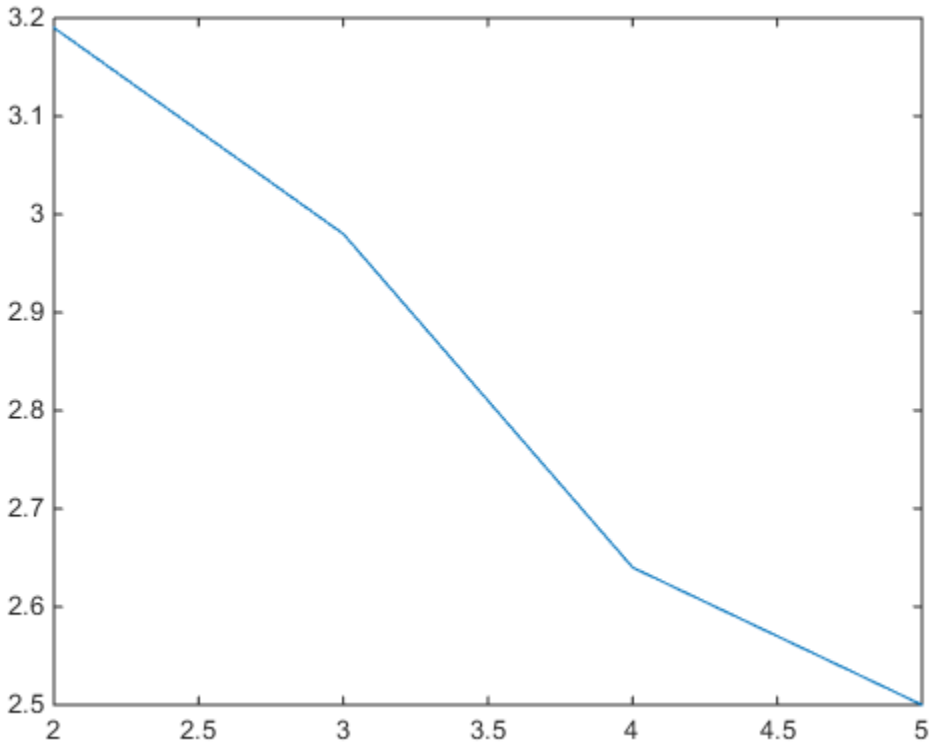

Fig.1: Change in misclassification rate(%) with the change in degree of polynomial kernel [Sequential Minimal Optimization]

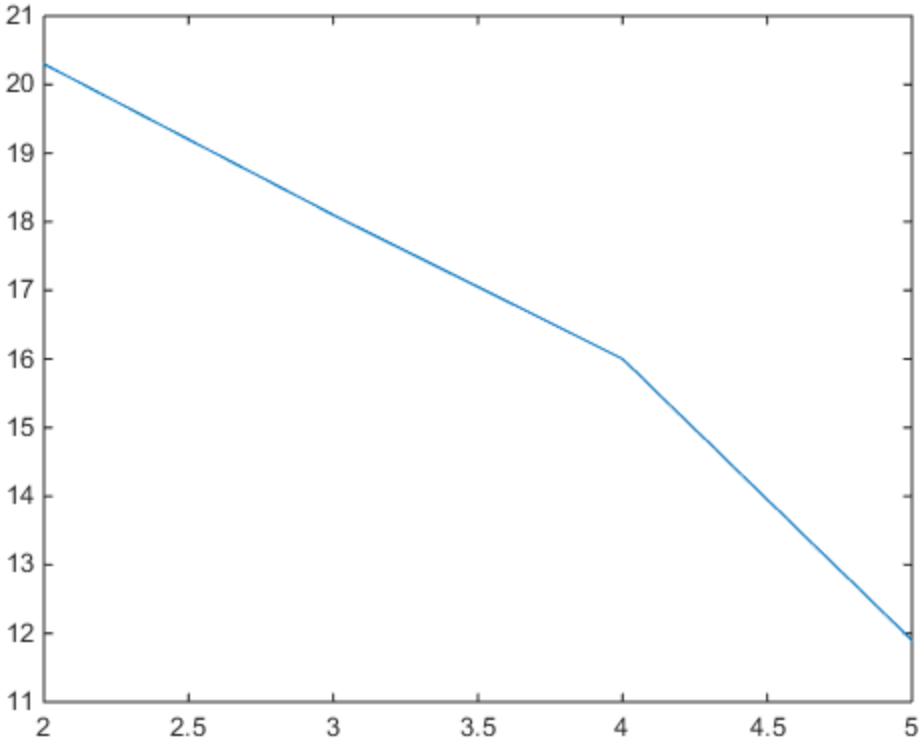

Fig.2: Change in misclassification rate(%) with the change in degree of polynomial kernel [Quadratic Optimization]

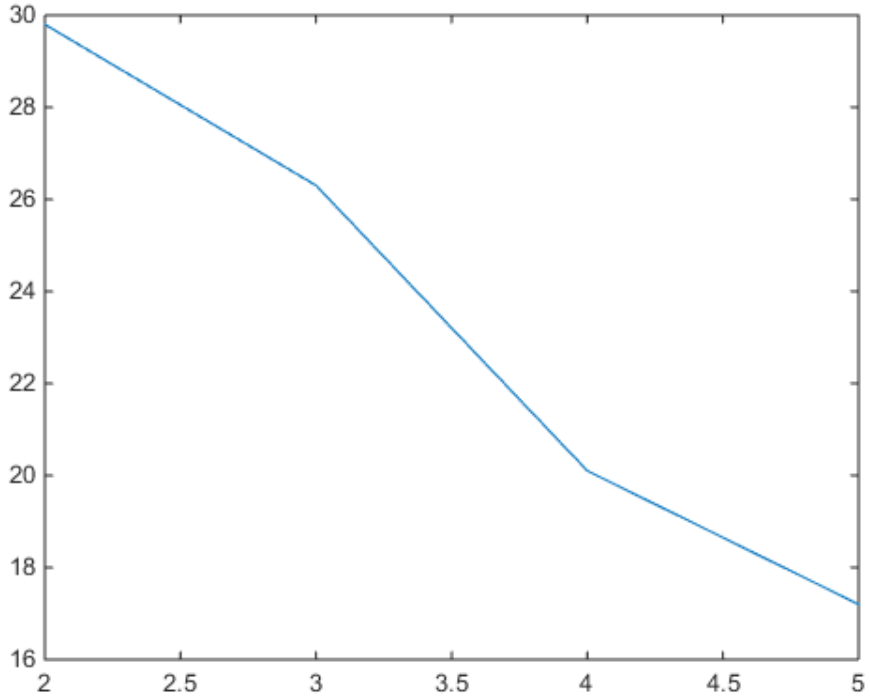

Fig.3: Change in misclassification rate(%) with the change in degree of polynomial kernel [Mixed Integer Linear Optimization]

Dataset Information:

The primary dataset used is Twitter_Absolute_sigma_500.data, which was achieved from http://ama.liglab.fr/data/buzz/classification/Twitter/Absolute_labeling/. The primary dataset was sampled to be compatible with the available system memory for running in Matlab and Weka. The sampled datasets are 'dataset.csv' for Matlab and 'Tweet.arff' for Weka.

Acknowledgement:

The special acknowledgement goes to Dr. Gary F. Holness for his cordial support and suggestions in working for this project.

……End…….