Foundations of Machine Learning — Homework Assignment 1

Anirudhan J Rajagopalan
N18824115
ajr619

November 08, 2015

A. VC-dimension of convex combinations

1

Given: H is a family of functions mapping from input space X to -1, +1. T is a positive integer.
Family of functions

$$\mathcal{F}_T = \{sgn(\sum_{t=1}^{T} \alpha_t h_t) : h_t \in H, \alpha_t \geq 0, \sum_{t=1}^{T} \alpha_t \leq 1\} \tag{1}$$

To find: The upper bound on the VC-dimension of the family of the functions $\mathcal{F}_T$.

Solution As given in the hint, if we consider the above as a neural network, we can think of the intermediate layers as representing the functions $h_1, h_2, ...h_t$ in the hypotheis set. By the same thought, we can see the output layer as representing the affine sgn() function.

Let $\prod_H(m)$ denote the grown function.

Given a concept class C, the output node can generate atmost $\prod_C(m)$ labelings for a fixed set of intermediate layers. Also, the intermediate layer can have $\prod_H \prod_H(m)$ possible set of values at the intermediate layers. Thus the total number of possible outputs from the network is

$$\prod_{\mathcal{F}}(m) \leq \prod_{c}(m) \times \prod_{H} \prod_{H}(m) \tag{2}$$

Since sgn is an affine function, its VC dimension is T+1.

$$\begin{aligned}
\prod_{c}(m) \leq & (\frac{em}{T+1})^{T+1} \\
\prod_{H}(m) \leq & (\frac{em}{d})^{d} \\
\prod_{F}(m) \leq & (\frac{em}{T+1})^{T+1}.((\frac{em}{d})^{d})^{T} \\
\leq & (em)^{T+1}(\frac{em}{d+1})^{dT}
\end{aligned}$$

Let $d^{'} = d+1$; $T^{'} = T+1$

$$\prod_{F}(m) \leq (\frac{em}{d^{'}})^{T^{'}d^{'}}$$

Let $m = 2T^{'}d^{'}\log_2 eT^{'}$

Now,

$$m = 2\log_2 xy => m \geq x\log_2(ym) \forall m \geq 1 \quad (3)$$

and x, y > 0 with xy > 4.

=> $m \geq d^{'}T^{'}\log_2(\frac{em}{d'})$ where x = $T^{'}d^{'}$ and y = $\frac{e}{d'}$ => $2^m \geq (\frac{em}{d'})^{T^{'}d^{'}}$

=> VCdim(F) $\leq$ 2(T+1)(d+1)$\log_2(e(T+1))$

B. Growth Functions

1

This is solved by referencing
`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.366.5645&rep=rep1&type=pdf`

The set W of separating vectors for $X+, X-$ is given by $W = w : w.x > 0, x \in X+; w.x \leq 0, x \in X-$

The dichotomy {X+ $\cup${x_{m+1}}, X- } is homogeneously linearly separable iff there exists a w in W such that wy>0; and the dichotomy {X+, X-$\cup$ {x_{m+1}}} is homogeneously linearly separable if and only if there exists a w in W such that w.y < 0

If {{ X+ $\cup$ { x_{m+1} }, X- } and { X+, X- $\cup$ { x_{m+1} }} are homogeneously linearly separable by w1 and w2, respectively, then w*=(-w2. x_{m+1})w1 +(w1.x_{m+1})w2 separates {X+, X-;} by the hyperplane {x: w*.x= 0} passing through y.

Conversely, if {X+, X- } is homogeneously linearly separable by a hyperplane containing x_{m+1}, then there exists a w* $\in$ W such that w* x_{m+1} =0. Since W is open, there exists an e> 0 such that w* +e .x_{m+1} and w*-e.x_{m+1} are in W.

Hence, {{ X+ $\cup$ { x_{m+1} }, X- } and { X+, X- $\cup$ { x_{m+1} }} are homogeneously linearly separable by w*+e.x_{m+1} and w* - e.x_{m+1}, respectively.

C. Support Vector Machines

1

Installed the software from[1]. The installed version of software is also checked into github at[2].

2

See the following command:

```
$ ./svm-scale -s splice_noise_train.txt.range \
> splice_noise_train.txt > splice_noise_train.txt.scale
$ ./svm-scale -r splice_noise_train.txt.range \
> splice_noise_test.txt > splice_noise_test.txt.scale
```

3

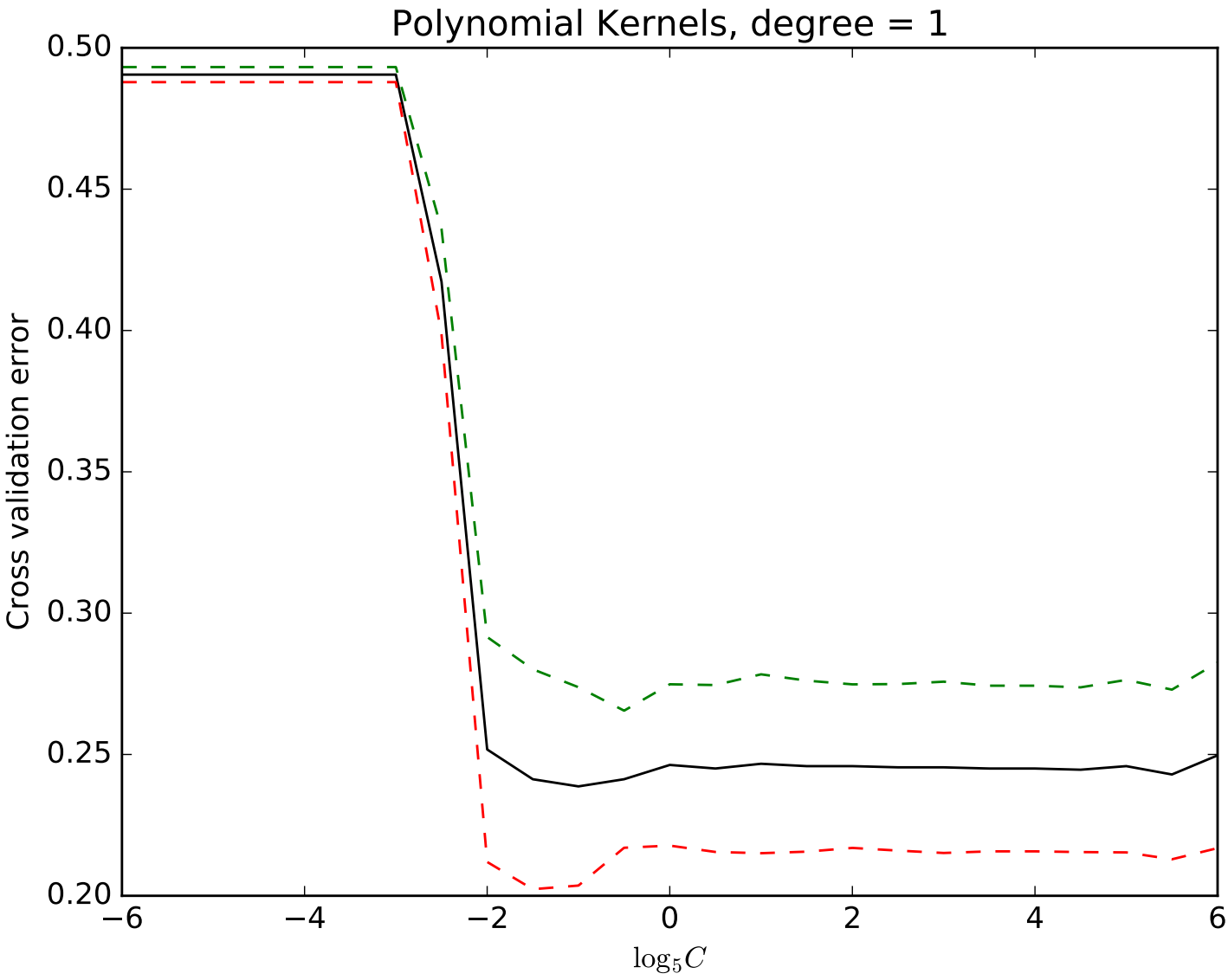
Run training and test script[3] by editing the KERNEL_DEGREE parameter for each value of d = 1, 3, 5.

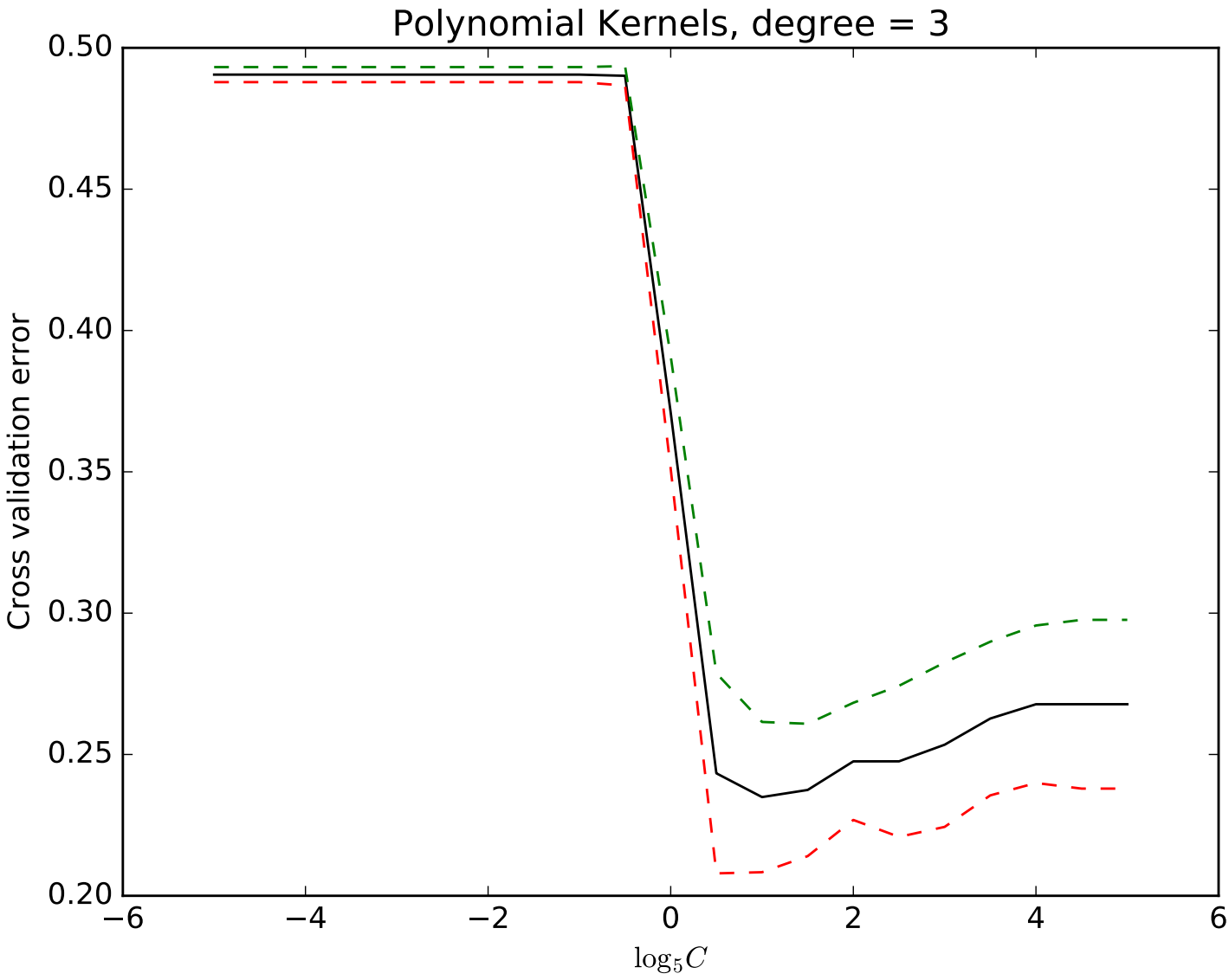
```
$ python cross_validation.py > deg1.out # KERNEL_DEGREE = 1
$ python cross_validation.py > deg3.out # KERNEL_DEGREE = 3
$ python cross_validation.py > deg5.out # KERNEL_DEGREE = 5
```

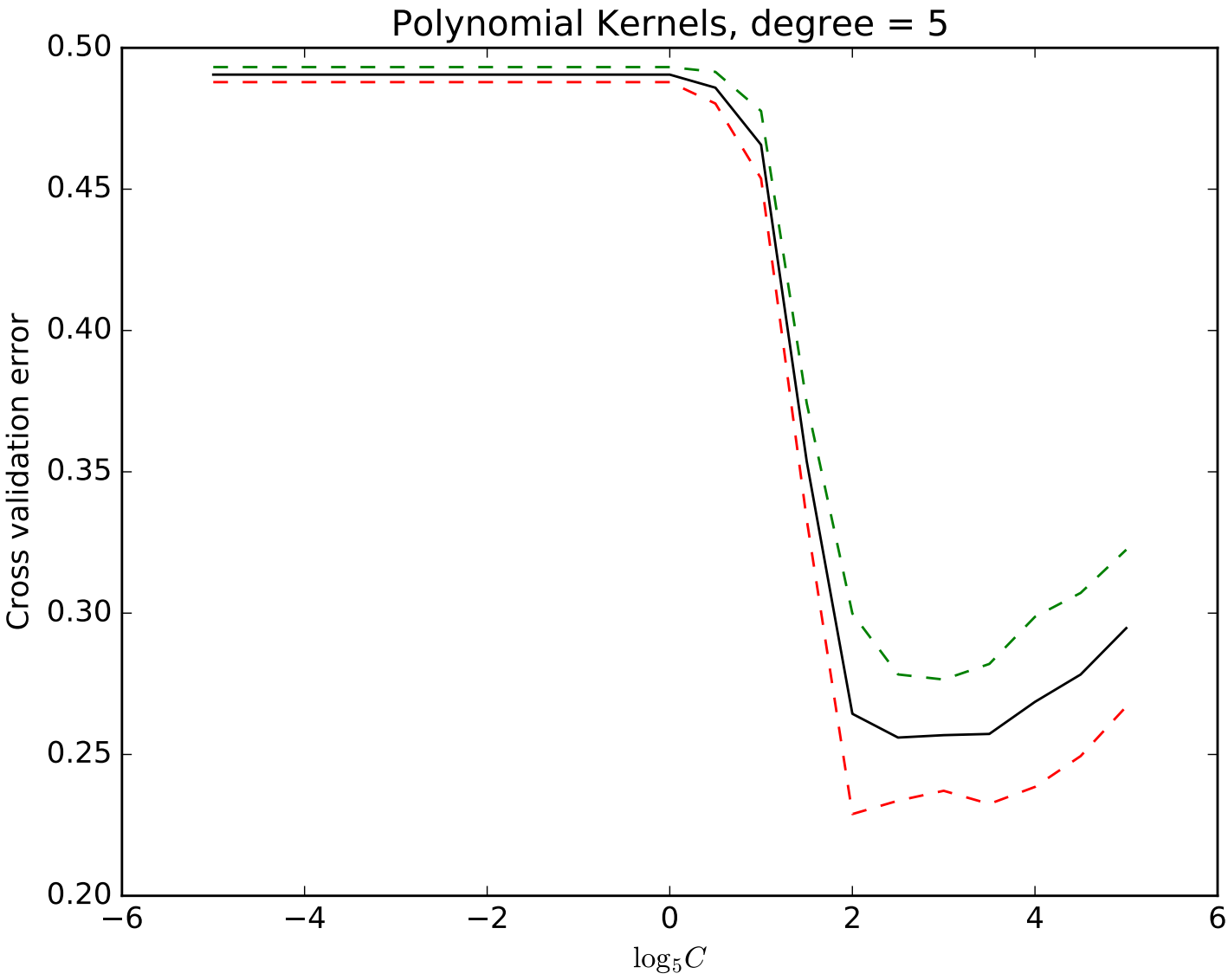
Filter the parameter and accuracy information from the run logs (deg1.out, deg3.out and deg5.out) by the command below.

```
$ cat deg1.out | grep OUR | cut -d' ' \
> -f2,3,4,5,6 > deg1.out.filtered
$ cat deg3.out | grep OUR | cut -d' ' \
> -f2,3,4,5,6 > deg3.out.filtered
$ cat deg5.out | grep OUR | cut -d' ' \
> -f2,3,4,5,6 > deg5.out.filtered
```

Use plotter.py[4] to create plots from the output values for KERNEL_DEGREE values 1, 3, 5. The output will be saved as deg1.pdf, deg3.pdf and deg5.pdf. All the three plots are embedded below one after the other. The dotted red line corresponds to -1 standard deviation. The dotted green line corresponds to +1 standard deviation and the black continuous line corresponds to the Mean.


Polynomial Kernels, degree = 1
Cross validation error
0.50
0.45
0.40
0.35
0.30
0.25
0.20
−6
−4
−2
0
2
4
6
$\log_5 C$


Polynomial Kernels, degree = 3
Cross validation error
0.50
0.45
0.40
0.35
0.30
0.25
0.20
−6
−4
−2
0
2
4
6
$\log_5 C$


Polynomial Kernels, degree = 5
Cross validation error
0.50
0.45
0.40
0.35
0.30
0.25
0.20
−6
−4
−2
0
2
4
6
$\log_5 C$

Best values of c for polynomial kernels 1, 3, 5 are:

$$d = 1 \qquad c* = 5^{-1.0} = 0.2 \qquad \text{cv} - \text{err} = 23.87\% \pm 3.5\%$$
$$d = 3 \qquad c = 5^{1.0} = 5 \qquad \text{cv} - \text{err} = 23.49\% \pm 2.65\%$$
$$d = 5 \qquad c = 5^{2.5} = 55.9017 \qquad \text{cv} - \text{err} = 25.59\% \pm 2.23\%$$

The best C measured in the cross-validation set is $C^* = 5^{1.0}$ with degree $d^* = 3$ which gives an average error of $23.49\% \pm 2.65\%$

4

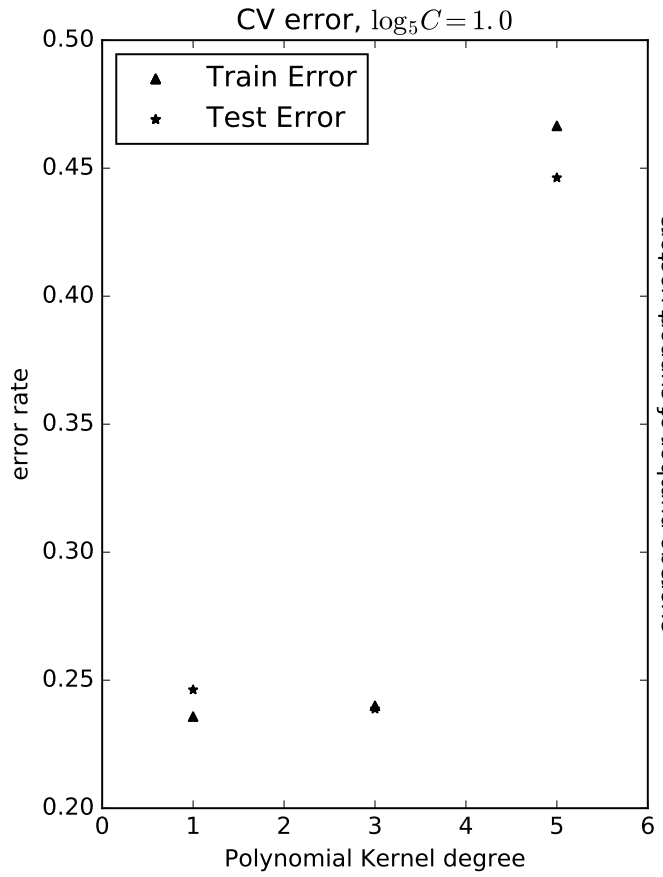
The average number of SV and BSV are taken across the cross validation sets. NumMarginal represents the average number of marginal support vectors.

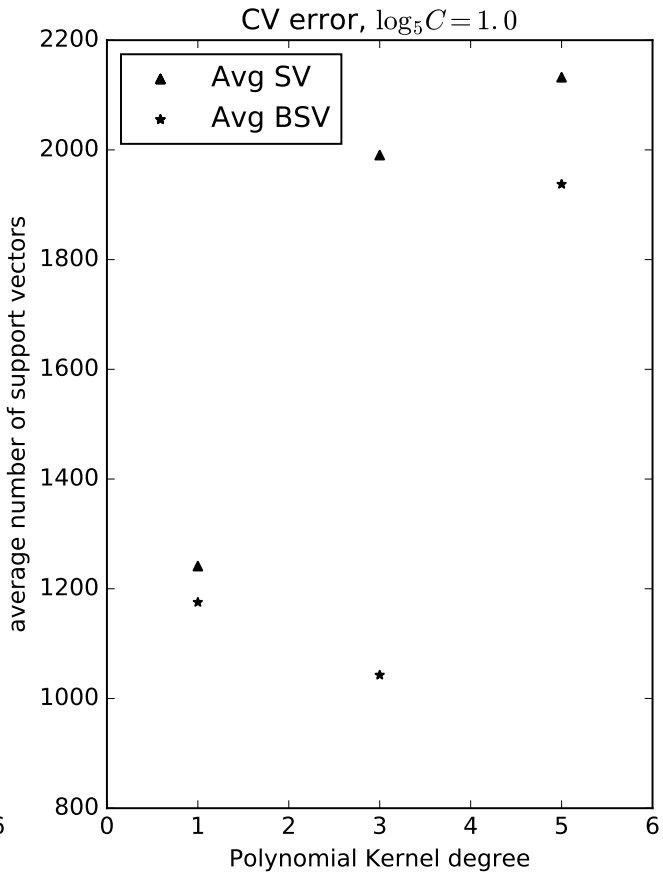
Similarly nSV and nBSV represent the number of support vectors and bounded support vectors in test set. Marginal represents the number of support vectors in test set.

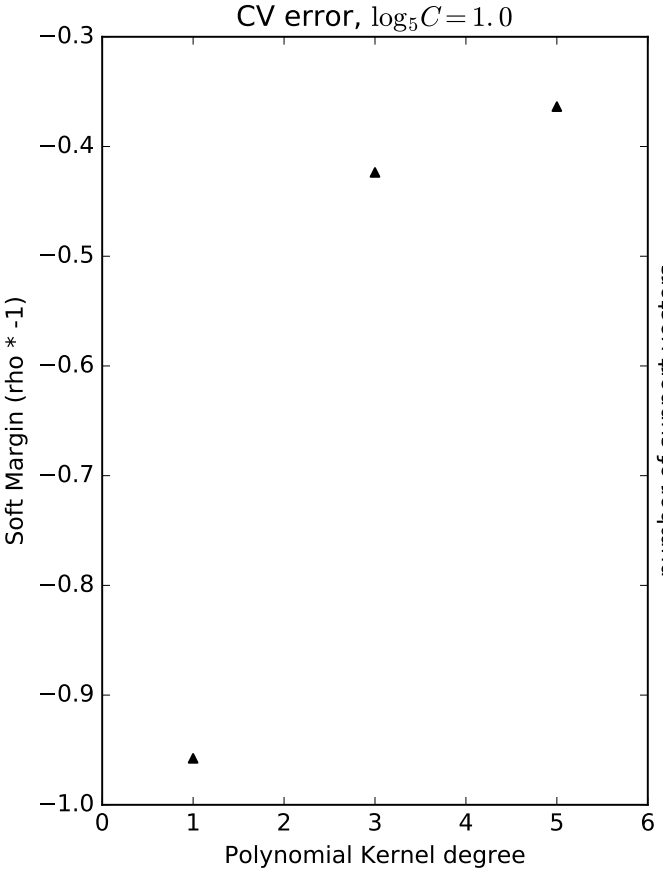
After manually populating d, cross-train accuracy, test accuracy, nSV, nBSV, rho, totalSV, totalBSV into a file[5], we run a script[6] to generate the plots as a pdf file.

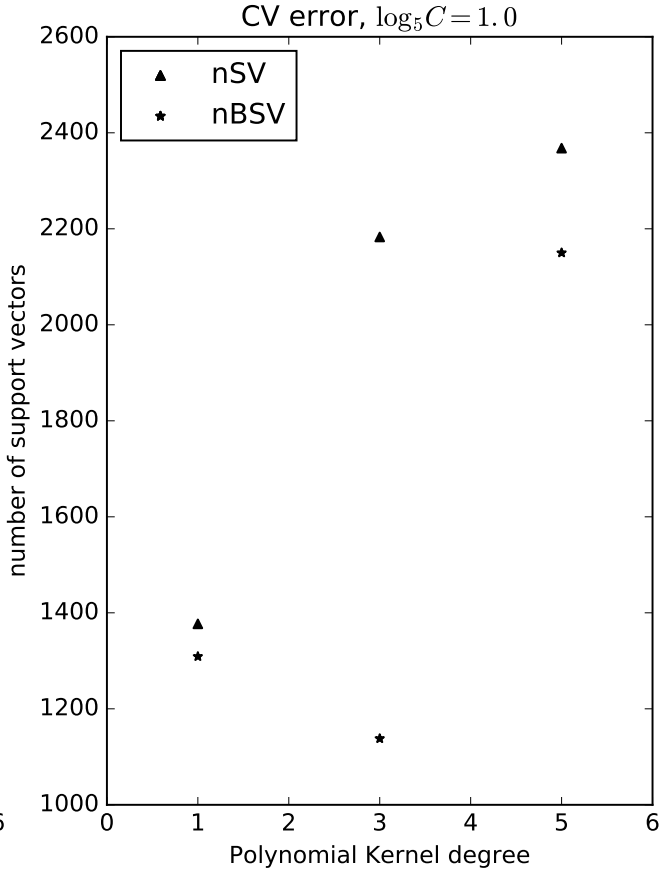
D	Avg Sv	Avg BSV	numMarginal	nSv	nBSV	marginal
1	1241.3	1175.4	65.9	1377	1309	68
3	1990.5	1042.7	947.8	2183	1138	1045
5	2132.4	1937.4	195.0	2368	2150	218

Soft margin is given by the rho value. Svm-train outputs negative margin values (rho). We use rho values to plot the soft margins in our plots. The plots generated are as shown below.


CV error, $\log_5 C = 1.0$
Train Error
Test Error
error rate
Polynomial Kernel degree


CV error, $\log_5 C = 1.0$
Avg SV
Avg BSV
average number of support vectors
Polynomial Kernel degree


CV error, $\log_5 C = 1.0$
Soft Margin (rho * -1)
Polynomial Kernel degree


CV error, $\log_5 C = 1.0$
nSV
nBSV
number of support vectors
Polynomial Kernel degree

5

We can modify grid.py to use 'C' in powers of 5 (change 2**c to 5**c in lines 142, 262, 348, 453 and 454). Running grid.py with -log5c -4,4,1 and -log2g -4,4,1 -v 10 -t 2, -s 0 gives us the best value of c and g.

We get the best cross validation accuracy of 79.7474 for c = 1.0 and g = 0.03125. Note that $\sigma = \sqrt{\frac{1}{2g}}$

We train the svm with the obtained c and v values

```
$ ./svm-train -t 2 -s 0 -c 1.0 -g 0.03125 \
> ./splice_hw/splice_noise_train.txt.scale ./rbf.model
$ ./svm-predict \./splice_hw/splice_noise_test.txt.scale \
> rbf.model rbf.output
```

gives us Accuracy = 80.125% (641/800) (classification)

The best accuracy we got with polynomial kernel SVM is 76.125 for $C^* = 5^{1.0}$ with degree $d^* = 3$.

The value of soft margin (rho) for the rbf kernel SVM is -0.586184.

6

D. Kernels

1

Given: Kernel, K is defined by $K(x, y) = \sum_{i=1}^{N} \cos^n(x_i^2 - y_i^2)$ for all $(X, Y) \in \mathbb{R}^N \times \mathbb{R}^N$

Solution: We know that

$$\cos(x_i^2 - y_i^2) = \sin(x_i^2).\sin(y_i^2) + \cos(x_i^2).\cos(y_i^2) \tag{4}$$

This can be written as a dot product of two vectors

$$\phi(x_i) = \begin{bmatrix} \cos(x_i^2) \\ \sin(x_i^2) \end{bmatrix} \qquad \text{and} \qquad \phi(y_i) = \begin{bmatrix} \cos(y_i^2) \\ \sin(y_i^2) \end{bmatrix} \tag{5}$$

K can be written as summation of N kernels that are a product of n $K^{'}$ kernels. $K' = \cos(x_i^2 - y_i^2)$. Hence to prove that K is PDS, it is enough to show that $K^{'}$ is PDS.

To show $K^{'}$ is PDS, we show that $c^T K' c > 0$ for any column vector $C = (C_1, C_2, ..., C_m)^T \in \mathbb{R}^m$)

$$C^T K^{'} C = \sum_{i,j=1}^{m} c_i c_j \begin{bmatrix}\cos(x_i^2)\\ \sin(x_i^2)\end{bmatrix} \begin{bmatrix}\cos(y_i^2)\\ \sin(y_i^2)\end{bmatrix} \tag{6}$$

$$= \sum_{i=1}^{n} c_i \begin{bmatrix}\cos(x_i^2)\\ \sin(x_i^2)\end{bmatrix} \times \sum_{j=1}^{n} c_j \begin{bmatrix}\cos(y_j^2)\\ \sin(y_j^2)\end{bmatrix} \tag{7}$$

$$= \left[\sum_{i=1}^{n} c_i \begin{bmatrix}\cos(x_i^2)\\ \sin(x_i^2)\end{bmatrix}\right]^2 \tag{8}$$

$$\geq 0 \tag{9}$$

Which proves $K^{'}$ is PDS.

Hence $K(x,y) = \sum_{i=1}^{N} \cos^n(x_i^2 - y_i^2)$ is PDS.

1

Given: $K(x,y) = e^{-\frac{\|x-y\|}{\sigma}}$

and $\|x-y\| = \frac{1}{2\Gamma\frac{1}{2}} \int_0^\infty \frac{1-e^{-t(\|x-y\|^2)}}{t^{3/2}} dt$ valid for all x,y

To prove: K is PDS

Proof: A normalized kernel $K^{'}$ for any kernel K is given by

$$K^{'} = \frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}}$$

For a PDS kernel given by K1 $= e^{2txy} \forall t \geq 0$ we have the normalized kernel K_1as

$$= \frac{e^{2txy}}{\sqrt{e^{2tx.x} e^{2ty.y}}} \tag{10}$$

$$= \frac{e^{2txy}}{e^{t(x.x+y.y)}} \tag{11}$$

$$= e^{-t(\|x-y\|^2)} \tag{12}$$

Since K1 is PDS, its normalized kernel K_1 is also a PDS kernel.

$=> -e^{-t(\|x-y\|^2)}$ is NDS and also $1 - e^{-t(\|x-y\|^2)}$ is also NDS. Since $t \geq 0$, we can say that $K_{nds} = \frac{1-e^{-t(\|x-y\|^2)}}{t^{3/2}}$ is also a NDS kernel.

Now, for the kernel $K_2 = \frac{1}{2\Gamma\frac{1}{2}} \int_0^\infty \frac{1-e^{-t(\|x-y\|^2)}}{t^{3/2}} dt$ for any column vector

$C = (C_1, C_2, ..., C_m)^T \in \mathbb{R}^m)$

$$\begin{aligned} C^T K_2 C &= \sum_{i,j=1}^{n} c_i c_j K_2(x_i x_j) \\ &= \sum_{i,j=1}^{n} c_i c_j \big[\frac{1}{2\Gamma\frac{1}{2}} \int_0^\infty \frac{1 - e^{-t(\|x-y\|^2)}}{t^{3/2}} dt\big] \\ &= \frac{1}{2\Gamma\frac{1}{2}} \big[\int_0^\infty \sum_{i,j=1}^{n} c_i c_j \frac{1 - e^{-t(\|x-y\|^2)}}{t^{3/2}} dt\big] \end{aligned}$$

Since we know that $K_{nds} \leq 0$;

$$C^T K_2 C \leq 0$$

$=>$ K_2 is also NDS.

This proves that $\|x - y\|$ is an NDS kernel. Since $\|x - y\|$ is NDS, from lecture5 which states that K is NDS iff exp(-tK) is PDS for all t > 0:

$=>$ $e^{-t\|x-y\|}$ is a PDS kernel where $t = \frac{1}{\sigma}$

References

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[2] `http://git.io/v80yn`

[3] `http://git.io/v80yY`

[4] `http://git.io/v80yk`

[5] `http://git.io/v8Edr`

[6] `http://git.io/v8EdH`

[7] `http://git.io/v8KT1`