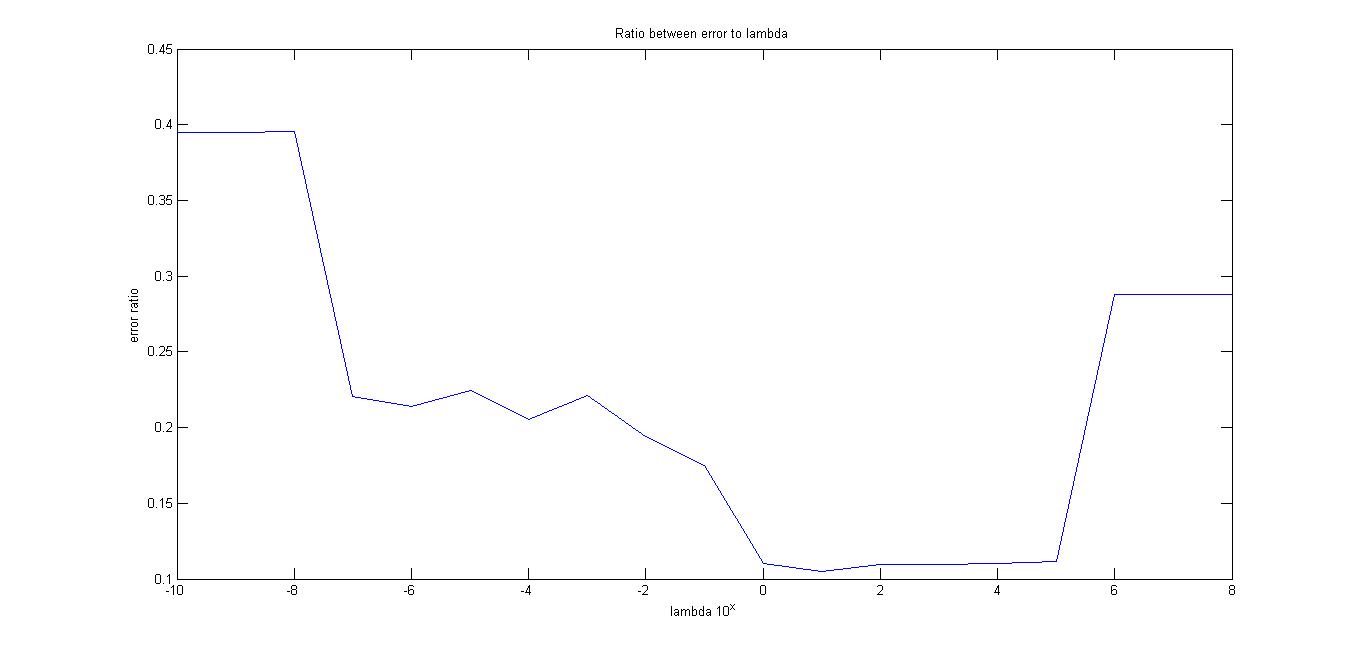
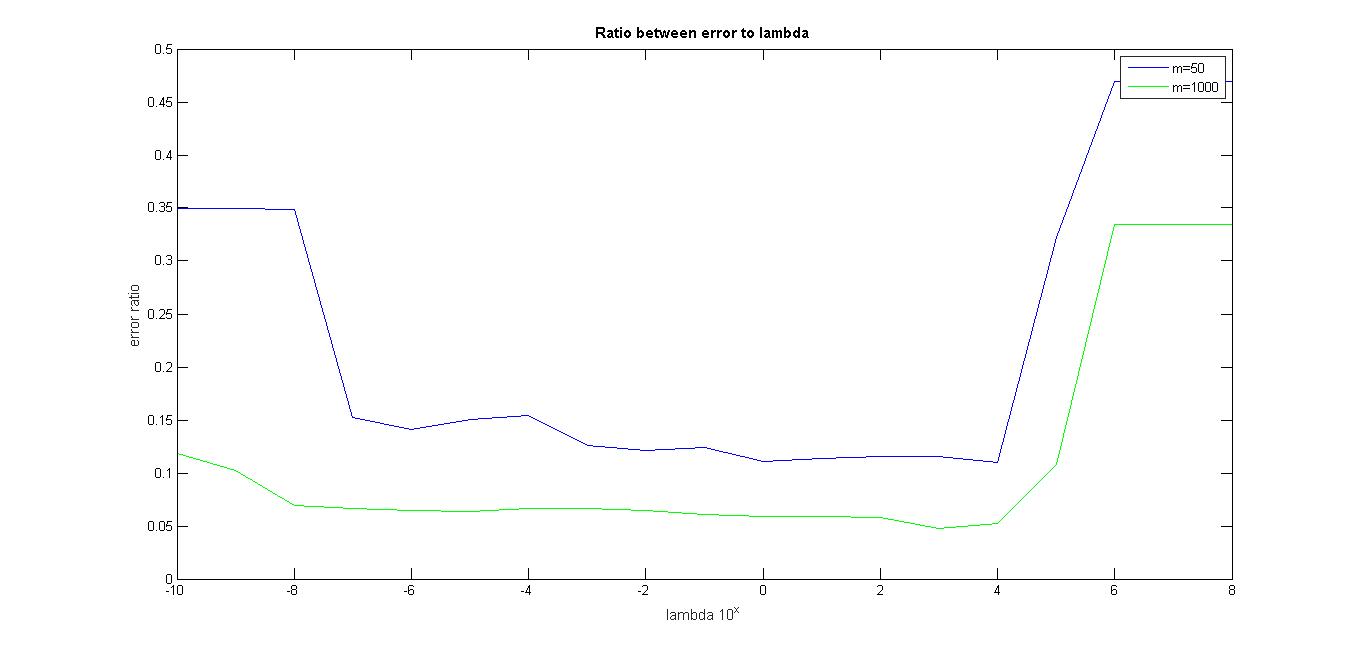
* If by the minimization process ->
* If , by the minimization process -> the minimization including the quadratic value.



Denote as the vector

3.

1. 
2. 

We saw in the class that the value is the tradeoff between the margin size and minimizing the hinge lost error component.

As we learned in the class, a small norm of will be expressed in a large margin, that will cause to more potential hinge lost error (points that are closed to the hyperplane) in a price of large margin.

The error equation of soft svm from the class:

*Small*

For we can see from the graph that for small values the error is quite big, and we associate it to the margin size (small margin for high ), because a small margin size can lead to false detect future samples (noise on the samples, noise on the current measure…).

But for a bigger sample size (m = 1000), even if we have small margin, the error is still low.

We can look at the equation and understand why it's true .

As we saw in the class, we can attribute this error to "*estimation error*", as long the margin is small and we need a lot more samples to overcome that error, and we see how big sample size is helping.

Big

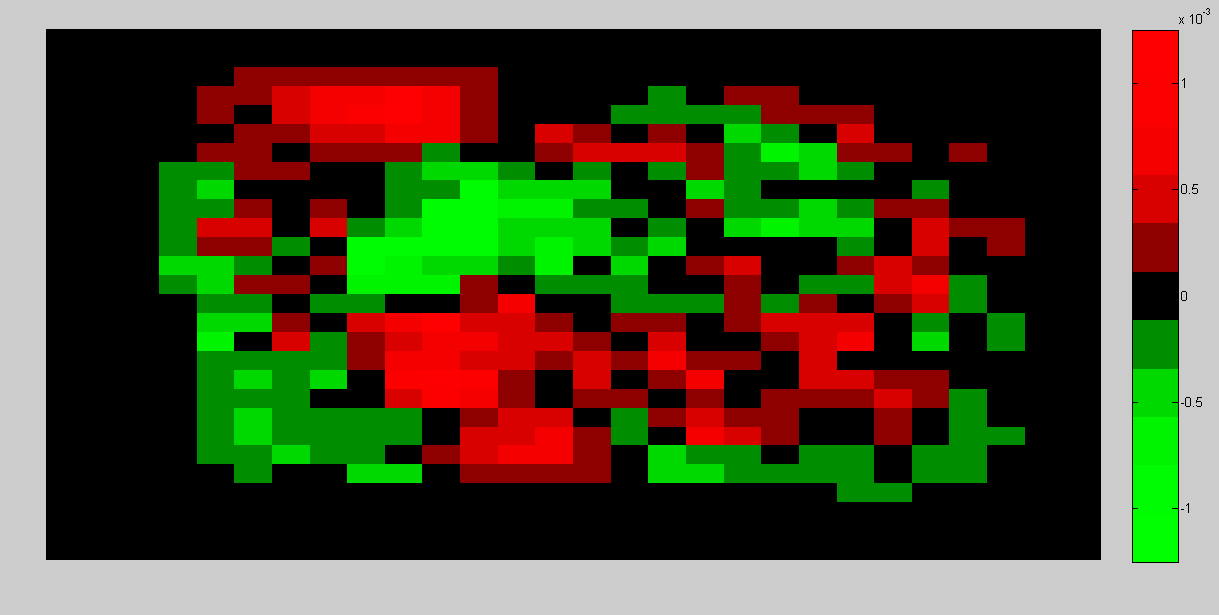
For both and m=1000, we can see that the error increased, and we can

We can also see that for a bigger sample size (green line) the error is lower, and again, by looking at the equation there is division by .

We can explain this phenomenon by "*approximation error*", as long we are decreasing the size of (Large margin).

The similarity between the lines are that both of them have bigger error on the sides (very large margin, and very small margin), and it makes possible according the the error formula.

This **question** demonstrates the bias-complexity tradeoff on each sample size. And we can see that in the middle we have a good result.

1. 

e)

the green belong to labels, and 5 is .

We can see that in the parts that only belong to "three" shape has an hard green (closer to ) and for the "five" shape the opposite thing happen (closer to 1).

The idea is that we w builet in a way that the dot product of them will be positive if it is 5 and negative if it is –1.

For example, the coordinates of shape 3 will multiplied by more negative coordinates of (the green coordinates), and the summation of them will properly be negative and classified as 3.

We can see that the shared part of the shape 3 and 5 are more darker be more neutrals.

4.

1. We will prove it by induction.

**Base case:**

We need to show that:

By perceptron algorithm:

and for each coordinate it is hold that:

**Assumption**:

Assume that for some iteration the inequality holds:

**Step:**

We need to show inequality hold for:

**Observation:**

1. For each coordinate on vector it is hold that , and for each index it is hold that .
2. From (a), for each it is hold that .

*Proof*:

Let's take a look on the vector, that created by the perceptron algorithm.

On each iteration update the value of changes as follow:

*Induction hypothesis:*

.

*Observation:*

And by the Triangle inequality:

1. We want to show that for each .

We will do it by induction over :

Base:

We need to show that it is valid for the coordinate

As we showed before, because separate **all** the samples in , by separating it exists that: .

Hypothesis:

Let assume that for each coordinate , it is hold that

(without the absolute value. It's following that is positive)

Proof:

Show on each value of :

We need to separate to cases:

*Even i:*

Let's take a look on vector , because is even, the vector look like:

By the samples S definition, and because labels correctly all the examples in S, it is hold that

Because is even, .

*1– by the induction hypothesis*

We prove that for any even it exists that

*Odd :*

Because is odd, the vector look like:

By the samples S definition, it is hold that

Because is odd, .

And we getting the same equation as the even case.

We show that and by the properties of absolute value:

1. By (a), the number of iteration is bigger then for any given in our sample.

And by (b), each coordinate .

And for max

Therefore, the number of iteration is exponential by .

But in the real life, the algorithm is way worse:

|  |  |
| --- | --- |
| d | Number of iterations |
| 1 | 1 |
| 2 | 6 |
| 3 | 22 |
| 4 | 86 |
| 5 | 342 |
| 6 | 1366 |
| 7 | 5462 |
| 8 | 21846 |
| 9 | 87382 |
| 10 | 349526 |