What is the influence of C in SVMs with linear kernel?

I am currently using an SVM with a linear kernel to classify my data. There is no error on the training set. I tried several values for the parameter C ($10^{-5}, \dots, 10^{2}$). This did not change the error on the test set.

Now I wonder: is this an error *caused by the ruby bindings* for `libsvm` I am using (rb-libsvm) or is this *theoretically explainable*?

Should the parameter C always change the performance of the classifier?

machine-learning svm libsvm

edited Apr 23 '16 at 19:10
Pål GD
45 11

asked Jun 23 '12 at 19:54

alfa
1,120 2 10 15

migrated from stackoverflow.com Jun 25 '12 at 12:27

This question came from our site for professional and enthusiast programmers.

Just a comment, not an answer: *Any* program that minimizes a sum of two terms, such as $|w|^2 + C\sum \xi_i$, should (imho) tell you what the two terms are at the end, so that you can see how they balance. (For help on computing the two SVM terms yourself, try asking a separate question. Have you looked at a few of the worst-classified points ? Could you post a problem similar to yours ?) – denis Jul 13 '12 at 20:02

5 Answers

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

edited Aug 23 '12 at 16:46

answered Jun 23 '12 at 20:43

Marc Shivers
1,456 1 10 6

1 OK, I understand that C determines the influence of the misclassification on the objective function. The objective function is the sum of a regularization term and the misclassification rate (see en.wikipedia.org/wiki/Support_vector_machine#Soft_margin). When I change C, this does not have any effect on the minimum of my objective function. Could that mean that the regularization term is always very small? – alfa Jun 24 '12 at 12:31

3 I would suggest trying a wider range of C values, maybe 10^[-5,...,5], or more if the optimization is fast on your dataset, to see if you get something that looks more reasonable. Both the training error and the value of the minimum cost should change as C is varied. Also, is the scale of your data extreme? In general, an optimal C parameter should be larger when you scale down your data, and vice versa, so if you have very small values for features, make sure to include very large values for the possible C values. If none of the above helps, I'd *guess* the problem is in the ruby bindings – Marc Shivers Jun 24 '12 at 19:59

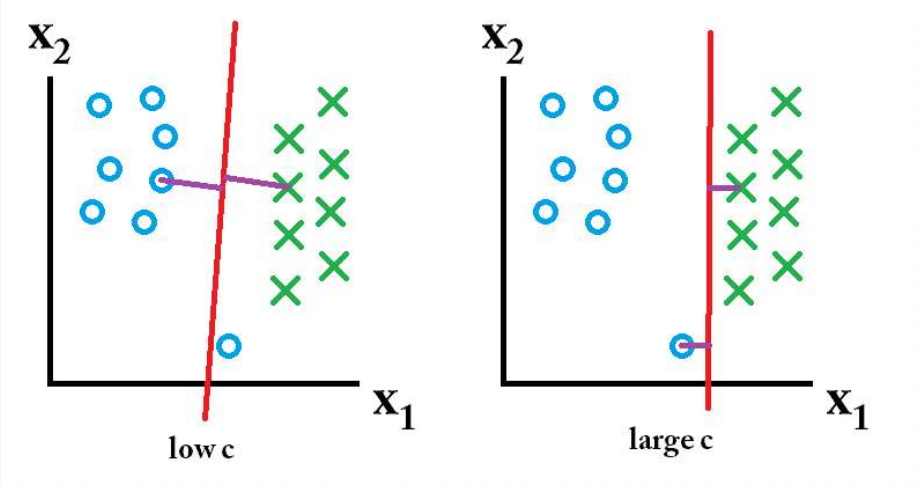
3 changing the balanced accuracy from 0.5 (just guessing) to 0.86 doesn't sound like a marginal influence to me. It would be a good idea to investigate a finer grid of values for C as Marc suggests, but the results you gave given seem to be fairly normal behaviour. One might expect the error to go back up again as C tends to infinity due to over-fitting, but that doesn't seem to much of a problem in this case. Note that if you are really interested in balanced error and your training set doesn't have a 50:50 split, then you may be able to get better results... – Dikran Marsupial Jun 25 '12 at 12:58

2 ... by using different values of C for patterns belonging to the positive and negative classes (which is asymptotically equivalent to resampling the data to change the proportion of patterns belonging to each class). – Dikran Marsupial Jun 25 '12 at 12:59

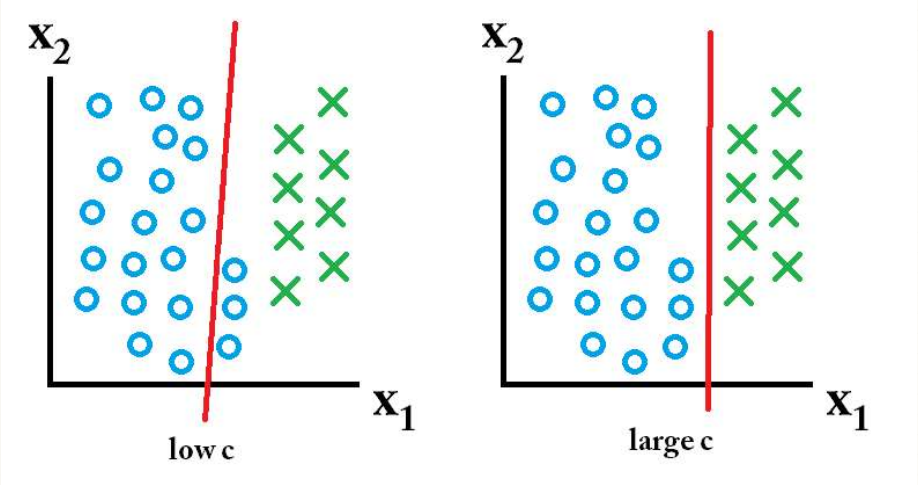
2 I think it is possible that once you get to C=10^0 the SVM is already classifying all of the training data correctly, and none of the support vectors are bound (the alpha is equal to C) in that case making C bigger has no effect on the solution. – Dikran Marsupial Jun 26 '12 at 12:26

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. ×

always be able to get both things. The c parameter determines how great your desire is for the latter. I have drawn a small example below to illustrate this. To the left you have a low c which gives you a pretty large minimum margin (purple). However, this requires that we neglect the blue circle outlier that we have failed to classify correct. On the right you have a high c. Now you will not neglect the outlier and thus end up with a much smaller margin.

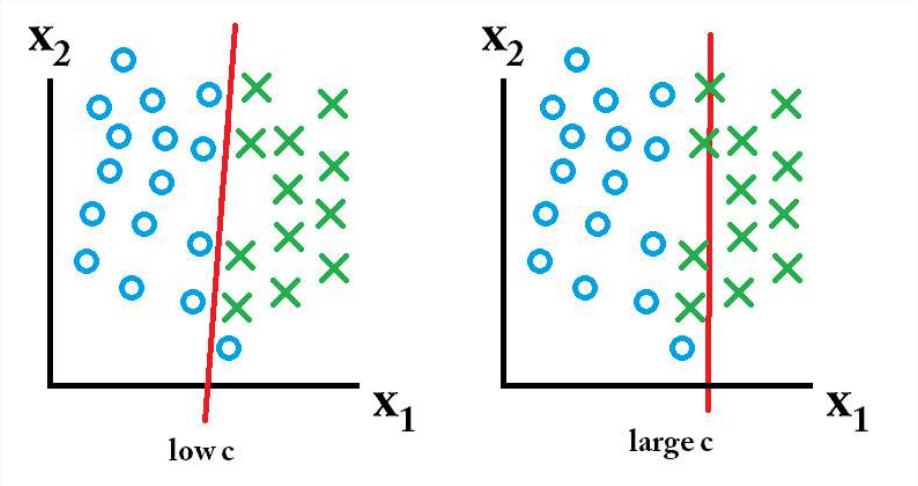


So which of these classifiers are the best? That depends on what the future data you will predict looks like, and most often you don't know that of course. If the future data looks like this:



then the classifier learned using a large c value is best.

On the other hand, if the future data looks like this:



then the classifier learned using a low c value is best.

Depending on your data set, changing c may or may not produce a different hyperplane. If it **does** produce a different hyperplane, that does not imply that your classifier will output different classes for the particular data you have used it to classify. Weka is a good tool for visualizing data and playing around with different settings for an SVM. It may help you get a better idea of how your data look and why changing the c value does not change the classification error. In general, having few training instances and many attributes make it easier to make a linear separation of the data. Also that fact that you are evaluating on your training data and not new unseen data makes separation easier.

By using our site, you acknowledge that you have read and understand our , , and our .

edited Jun 28 '15 at 16:14

answered Jun 28 '15 at 16:04

Kent Munthe Caspersen
1,823 1 7 8

4 I did not touch the data for more than 3 years now. It is very high-dimensional and noisy and I am not allowed to publish it. The question has been answered already but I think your visualization is very good and intuitive. – alfa Sep 2 '15 at 15:14

awesome. do you have such explanations for gama value too ? – MonsterMMORPG Oct 17 '15 at 18:33

2 The gamma parameter is used for the Gaussian kernel function. The kernel functions can be seen as an efficient way to transform your original features into another space, where a separating hyperplane in the new feature space does not have to be linear in the original feature space. For instance, the two dimensional position of a data point in the original feature space could be used to calculate a new feature representing the distance to some marker on a map. With this new feature, a non-linear classifier (in original space) can be made which decision boundary forms a circle around the marker – Kent Munthe Caspersen Oct 23 '15 at 12:57

@KentMuntheCaspersen isn't your explanation of C incorrect? It's the opposite of what it says in the book "Introduction to Statistical Learning". – diugalde Feb 6 '17 at 17:40

2 @diugalde can you quote from the book what exactly differs from my explanation? I always think of c as the cost of misclassification (easy to remember by c in classification). In that way higher c means high cost of misclassification, leading to the algorithm trying to perfectly separate all data points. With outliers this is not always possible or wont always lead to a good general result, which is a good reason for lowering / introducing c. – Kent Munthe Caspersen May 17 '17 at 9:16

C is essentially a regularisation parameter, which controls the trade-off between achieving a low error on the training data and minimising the norm of the weights. It is analageous to the ridge parameter in ridge regression (in fact in practice there is little difference in performance or theory between linear SVMs and ridge regression, so I generally use the latter - or kernel ridge regression if there are more attributes than observations).

Tuning C correctly is a vital step in best practice in the use of SVMs, as structural risk minimisation (the key principle behind the basic approach) is party implemented via the tuning of C. The parameter C enforces an upper bound on the norm of the weights, which means that there is a nested set of hypothesis classes indexed by C. As we increase C, we increase the complexity of the hypothesis class (if we increase C slightly, we can still form all of the linear models that we could before and also some that we couldn't before we increased the upper bound on the allowable norm of the weights). So as well as implementing SRM via maximum margin classification, it is also implemented by the limiting the complexity of the hypothesis class via controlling C.

Sadly the theory for determining how to set C is not very well developed at the moment, so most people tend to use cross-validation (if they do anything).

answered Jun 25 '12 at 12:52

Dikran Marsupial
35.2k 2 100 146

OK, I think I understand the meaning of C now. :) – alfa Jun 25 '12 at 19:20

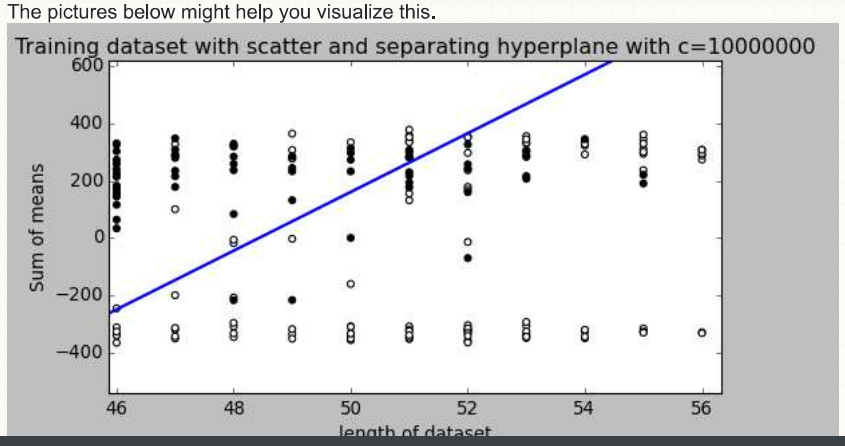
2 But if C is a regularization parameter, why does a high C increase overfitting, when generally speaking regularization is done to mitigate overfitting, i.e., by creating a more general model? – user1603472 Feb 26 '16 at 14:58

2 C is a regularisation parameter, but it is essentially attached to the data misfit term (the sum of the slack variables) rather than the regularisation term (the margin bit), so a larger value of C means less regularisation, rather than more. Alternatively you can view the usual representation of the rgularisation parameter as 1/C. – Dikran Marsupial Feb 26 '16 at 15:36

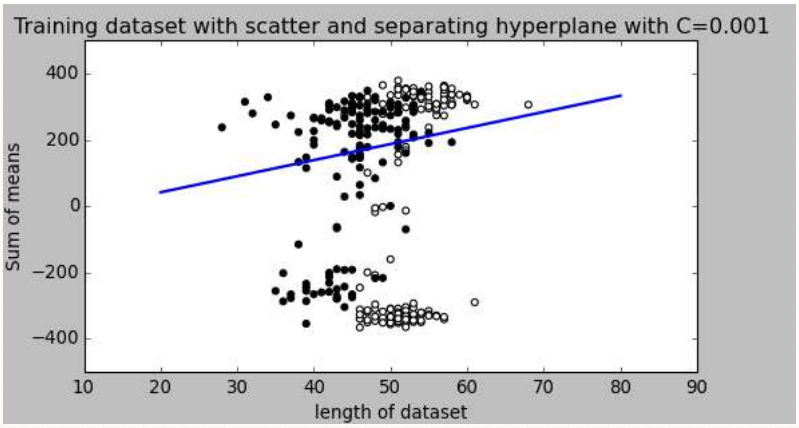
C is a regularization parameter that controls the trade off between the achieving a low training error and a low testing error that is the ability to generalize your classifier to unseen data.

Consider the objective function of a linear SVM : min |w|^2+C∑ξ. If your C is too large the optimization algorithm will try to reduce |w| as much as possible leading to a hyperplane which tries to classify each training example correctly. Doing this will lead to loss in generalization properties of the classifier. On the other hand if your C is too small then you give your objective function a certain freedom to increase |w| a lot, which will lead to large training error.

The pictures below might help you visualize this.



By using our site, you acknowledge that you have read and understand our , , and our .



answered Feb 13 '15 at 6:39

deerishi
71 1 3

I don't really understand your plots. Can you explain it? – alfa Mar 20 '15 at 10:14

@alfa : My intent for showing the plots was : 1) If C is too large (plot 1) , then your classifier will over fit ,i.e. it'll try to classify each training data point accurately. Plot 1 shows almost all training points being classified correctly. 2)On the other hand if C is too less (plot 2) , then your classifier will under fit. Plot 2 shows the under fit classifier. It does not segregate the points into their respective classes. Hope this helps. – deerishi Apr 10 '15 at 11:09

That means that your x- and y-axes show two different features. The labels "length of dataset" and "Sum of means" are a little bit confusing? – alfa Apr 10 '15 at 13:46

It would be interesting to see how the right choice for C helps in both cases. – alfa Apr 10 '15 at 13:49

2 I think it is not obvious to see that C=10000000 is a bad choice and I think the dataset is not the right one to demonstrate that. Maybe a dataset with only a few outliers on the wrong side of the separating hyperplane would be better? – alfa Apr 12 '15 at 12:28

The answers above are excellent. After carefully reading your questions, I found there are 2 important facts we might overlooked.

1. You are using linear kernel
2. Your training data is linearly separable, since "There is no error on the training set".

Given the 2 facts, if C values changes within a reasonable range, the optimal hyperplane will just randomly shifting by a small amount within the margin(the gap formed by the support vectors).

Intuitively, suppose the margin on training data is small, and/or there is no test data points within the margin too, the shifting of the optimal hyperplane within the margin will not affect classification error of the test set.

Nonetheless, if you set C=0, then SVM will ignore the errors, and just try to minimise the sum of squares of the weights(w), perhaps you may get different results on the test set.

edited Oct 4 '16 at 14:33

answered Oct 3 '16 at 15:56

luz
41 2

By using our site, you acknowledge that you have read and understand our , , and our .