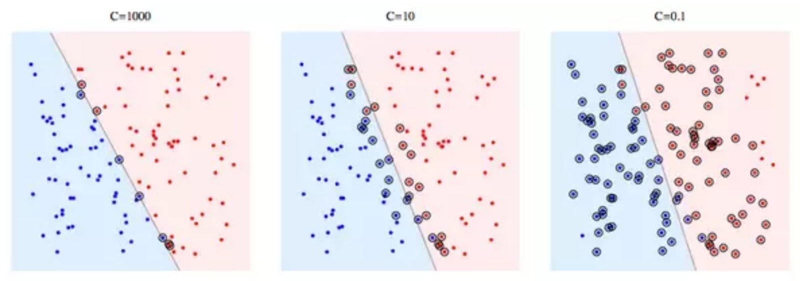
# What are C and gamma with regards to a support vector machine?

5 Answers

Jeffrey M Girard, Affective Computing Researcher
Updated Apr 23, 2016 · Author has **212** answers and **419.6k** answer views

C and Gamma are the parameters for a nonlinear support vector machine (SVM) with a Gaussian radial basis function kernel.

A standard SVM seeks to find a margin that separates *all* positive and negative examples. However, this can lead to poorly fit models if any examples are mislabeled or extremely unusual. To account for this, in 1995, Cortes and Vapnik proposed the idea of a "soft margin" SVM that allows some examples to be "ignored" or placed on the wrong side of the margin; this innovation often leads to a better overall fit. C is the parameter for the soft margin cost function, which controls the influence of each individual support vector; this process involves trading error penalty for stability.



From: SVM - hard or soft margins?

A standard SVM is a type of linear classification using dot product. However, in 1992, Boser, Guyan, and Vapnik proposed a way to model more complicated relationships by replacing each dot product with a nonlinear kernel function (such as a Gaussian radial basis function or Polynomial kernel). Gamma is the free parameter of the Gaussian radial basis function.

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0$$

A small gamma means a Gaussian with a large variance so the influence of x_j is more, i.e. if x_j is a support vector, a small gamma implies the class of this support vector will have influence on deciding the class of the vector **x_i** even if the distance between them is large**.** If gamma is large, then variance is small implying the support vector does not have wide-spread influence. Technically speaking, large gamma leads to high bias and low variance models, and vice-versa.

80k Views · View Upvoters

Support Vector Machines: Why is the default value of Gamma in Gaussian RBF kernel 1/number_of_features?

What is the use of a support vector machine?

What is a role kernel in a support vector machine?

How does a SVM choose its support vectors?

What is the bias value in a support vector machine?

How do I implement a support vector machine in R?

How is an HOG vector classified in a support vector machine?

How do support vector machines work?

What can be the option if Quad program fails in SVM?

In a support vector machine, the number of support vectors can be much smaller than the training set. How can this feature be useful?

+ Ask New Question

What is the use of a support vector machine?

What is a role kernel in a support vector machine?

How does a SVM choose its support vectors?

What is the bias value in a support vector machine?

+ Ask New Question

Dima Korolev, https://dimakorolev.quora.com/Against-Justificationism
Updated Apr 21, 2015 · Upvoted by Luis Argerich, Data Science Professor at the University of Buenos Aires (UBA) · Author has **3.9k** answers and **10.8m** answer views

**C** controls the cost of misclassification on the training data.

The goal of SVM is to find a hyperplane that would leave the widest possible "cushion" between input points from two classes. There is a tradeoff between "narrow cushion, little / no mistakes" and "wide cushion, quite a few mistakes".

Learning algorithms are about generalizing from input data, not explaining it. This is not to mention that, "thanks to" the curse of dimensionality, in large number of dimensions training data can often be explained quite well by over fitting the model.

Therefore, often times it is desirable to specifically allow some training points to be misclassified in order to have an "overall better" position of the separating hyperplane.

- Mathematically, "better" translates to "optimizing cost function valuing mistakes with certain coefficient".
- Intuitively, "better" implies "wider cushion, a few mistakes allowed".
- Practically "better" is to be understood as "performs well on real data".

**Small C makes the cost of misclassificaiton low** ("soft margin"), thus allowing more of them for the sake of wider "cushion".

**Large C makes the cost of misclassification high** ('hard margin"), thus forcing the algorithm to explain the input data stricter and potentially overfit.

The goal is to find the balance between "not too strict" and "not too loose". Cross-validation and resampling, along with grid search, are good ways to finding the best C.

Gamma is a kernel parameter, and I am not the best person to answer this part of the question.

34.3k Views · View Upvoters

Your response is private.

Is this answer still relevant and up to date? Yes No

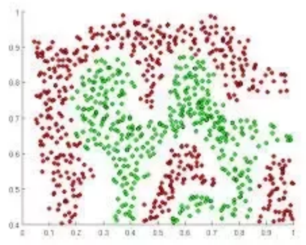
Luis Argerich, Data Science Professor at the University of Buenos Aires (UBA)
Answered Aug 14, 2014 · Author has **692** answers and **1.5m** answer views

C is the cost of classification as correctly stated by Dima.

Still have a question? Ask your own!

**What is your question?** Ask

Gamma is the parameter of a Gaussian Kernel (to handle non-linear classification). Check this points:



They are not linearly separable in 2D so you want to transform them to a higher dimension where they will be linearly sepparable. Imagine "raising" the green points, then you can sepparate them from the red points with a plane (hyperplane)

To "raise" the points you use the RBF kernel, gamma controls the shape of the "peaks" where you raise the points. A small gamma gives you a pointed bump in the higher dimensions, a large gamma gives you a softer, broader bump.

So a small gamma will give you low bias and high variance while a large gamma will give you higher bias and low variance.

You usually find the best C and Gamma hyper-parameters using Grid-Search.

49k Views · View Upvoters

Your response is private.

Is this answer still relevant and up to date? Yes No

Promoted by DatadogHQ.com

**Datadog: cloud monitoring as a service.**

Track your dynamic infrastructure with Datadog's cloud-scale monitoring. Start your free trial now.

Learn more at datadoghq.com

Rohan Varma, Machine Learning Enthusiast
Answered Mar 19, 2017 · Author has **137** answers and **232.7k** answer views

To answer this, we can first consider the optimization problem that gives us the “hard-margin” SVM, ie, an SVM that is capable of perfectly classifying data that are completely linearly separable. This is not really useful in practice, but provides us a good background for discussing why C and gamma are essential hyperparameters to tune in practice.

The optimization problem for the hard margin SVM can be written as follows:

$min_{w,b} \frac{1}{2}||w||_2^2$ s.t. $y_n(w^T x_n + b) \geq 1$

Without getting into too many of the gory details, this optimization problem basically tells us to maximize the margin, which is the distance from the hyperplane that represents the SVM’s decision boundary to the nearest training point, while making sure to confidently and correctly classify every example

Still have a question? Ask your own!

**What is your question?** Ask

To combat this, we can introduce the notion of "slack variables", which are variables that allow us to relax the constraint $y_n(w^T x_n + b) \geq 1$, meaning that we no longer will have to correctly and confidently classify every single training point.

We can define a slack variable as a value $\zeta$that, roughly, indicates how much we must move our point so that it is correctly and confidently classified. This makes sense - small slack variables means that we have a correct classification but aren't very confident, while large slack variables means that we have not classified the point correctly.

Our constraint then becomes a relaxed version of the previous. We should also penalize our objective function whenever we use slack variables, otherwise we can just set the slack variables to extremely high values and solve the optimization problem, but obtain a terrible solution to it:

$min_{w,b} \frac{1}{2}||w||_2^2 + C \sum_n \zeta_n \text{s.t. } y_n(w^T x_n + b) \geq 1 - \zeta_n$

This is where the variable $C$ comes from - it is a hyperparameter that controls how much we penalize our use of slack variables. If we let $C \rightarrow 0$ then we don't penalize slack variables at all, and as we increase $C$ , we penalize our slack variables more and more - it's essentially a tradeoff between penalizing slack variables and obtaining a large margin for our SVM.

SVMs can be kernelized using some interesting dual optimization techniques, which means that the SVM learning and prediction problems can be written as linear combinations of inner products between the training and testing data. This is pretty powerful, because it turns out that due to this, we can define an arbitrary kernel function $k(x_n, x_m)$that we can substitute for our inner products when we are learning the SVM (the kernel function can't be completely arbitrary, it must satisfy some constraints such that the associated kernel matrix must be positive semidefinite, but besides that the choice of kernel function is really only limited by your imagination).

One interesting kernel function is the Gaussian/rbf kernel: $k(x_n, x_m) = exp(-\gamma||x_n - x_m||_2^2)$. We can let $x_n$ be the training example and $x_m$ be the testing example.

There's a few interesting things about this kernel:

1. The value depends on the L2-squared distance between its inputs. One interpretation of this kernel can then be the idea that it measures how similar two feature vectors are.
2. The kernel can be interpreted as corresponding to a feature transformation onto an infinite dimensional space (this can be seen by writing out the Taylor polynomial expansion for the exponential function). This is another reason why kernels are useful - they can be computed in linear time as opposed to having to compute inner products between feature transformations, which usually have quadratic or worse time complexity in their size.
3. The hyperparameter $\gamma$ controls the tradeoff between error due to bias and variance in your model. If you have a very large value of gamma,

and be prone to low bias/high variance. On the other hand, a small value for gamma implies that the support vector has larger influence on the classification of $x_m$. This means that the model is less prone to overfitting, but you may risk not learning a decision boundary that captures the shape and complexity of your data. This leads to a high bias, low variance model.

13.8k Views · View Upvoters

Teru Watanabe, Data Analyst passionate about Statistics and Machine Learning
Answered Oct 12, 2016

RBF SVM parameters

Intuitively, the `gamma` parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The `gamma` parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

The `C` parameter trades off misclassification of training examples against simplicity of the decision surface. A low `C` makes the decision surface smooth, while a high `C` aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors.

11.6k Views · View Upvoters

## Related Questions

How do I implement a support vector machine in R?

How is an HOG vector classified in a support vector machine?

How do support vector machines work?

What can be the option if Quad program fails in SVM?

In a support vector machine, the number of support vectors can be much smaller than the training set. How can this feature be useful?

Are support vector machines friendly?

What is weight in support of a vector machine?

Can we use Support Vector Machine in prediction?

How can I get the error training in support vector machine?

Why is Support vector Machine hard to code from scratch where Logistic Regression is not?

**+ Ask New Question**