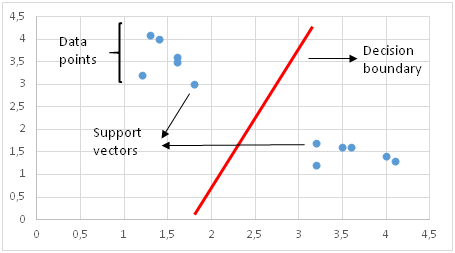
Support Vector Machine (SVM) is a widely-used supervised machine learning algorithm. It is mostly used in classification tasks but suitable for regression tasks as well. In this post, we dive deep into two important parameters of support vector machines which are **C**and **gamma**. So I will assume you have a basic understanding of the algorithm and focus on these parameters.

Most of the machine learning and deep learning algorithms have some parameters that can be adjusted which are called **hyperparameters**. We need to set hyperparameters before we train the models. Hyperparameters are very critical in building robust and accurate models. They help us find the balance between bias and variance and thus, prevent the model from overfitting or underfitting. To be able to adjust the hyperparameters, we need to understand what they mean and how they change a model. It would be a tedious and never-ending task to randomly trying a bunch of hyperparameter values.

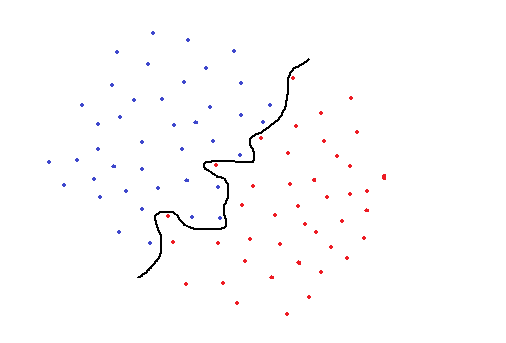
We emphasized the importance of hyperparameters. Let’s start our discussion on **C** and **gamma**. SVM creates a **decision boundary** which makes the distinction between two or more classes. How to draw or determine the decision boundary is the most critical part in SVM algorithms. When the data points in different classes are linearly separable, it is an easy task to draw a decision boundary.



Linearly separable data points

However, real data is noisy and not linearly separable in most cases. A standard SVM tries to separate all positive and negative examples (i.e. two different classes) and does not allow any points to be misclassified. This results in an overfit model or, in some cases, a decision boundary cannot be found with a standard SVM.

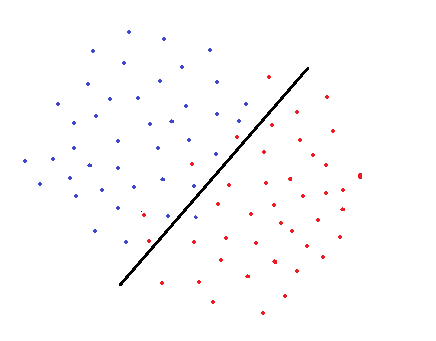
Consider the data points in a 2-dimensional space below:



Standard SVM

A standard SVM would try to separate blue and red classes by using the black curve line as a decision boundary. However, this is a too specific classification and highly likely to end up overfitting. An overfit SVM achieves a high accuracy with training set but will not perform well on new, previously unseen examples. This model would be very sensitive to noise and even very small changes in data point values may change the classification results. The SVM that uses this black line as a decision boundary is not generalized well to this dataset.

To overcome this issue, in 1995, Cortes and Vapnik, came up with the idea of “**soft margin**” SVM which allows some examples to be misclassified or be on the wrong side of decision boundary. Soft margin SVM often result in a better generalized model. In our example, the decision boundary for soft margin SVM might look like the black straight line as below:



Soft margin SVM

There are some misclassified points but we end up having a more generalized model. When determining the decision boundary, a soft margin SVM tries to solve an optimization problem with the following goals:

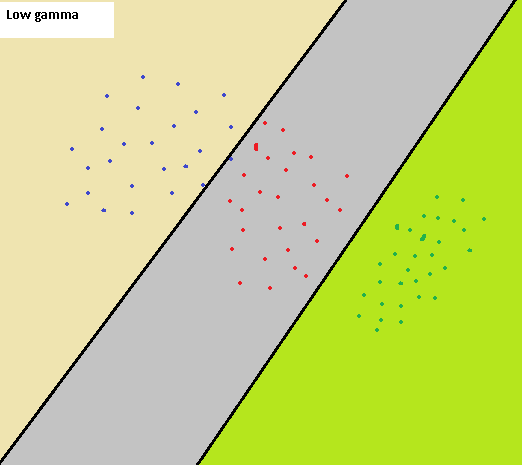
* Increase the distance of decision boundary to classes (or support vectors)
* Maximize the number of points that are correctly classified in the training set

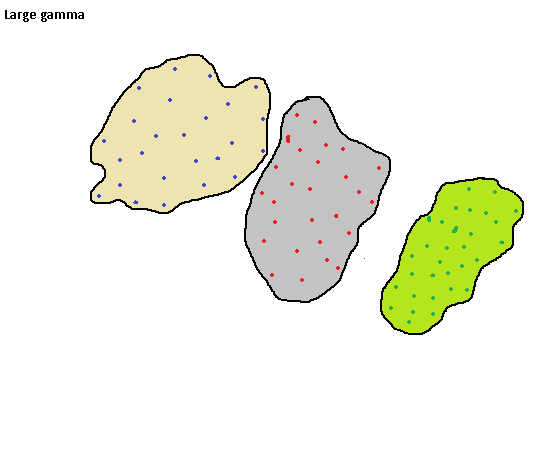
There is obviously a trade-off between these two goals. Decision boundary might have to be very close to one particular class to correctly label all data points in training set. However, in this case, accuracy on test dataset might be lower because decision boundary is too sensitive to noise and to small changes in the independent variables. On the other hand, a decision boundary might be placed as far as possible to each class with the expense of some misclassified exceptions. This trade-off is controlled by **c parameter.**

**C parameter**adds a penalty for each misclassified data point. If c is small, the penalty for misclassified points is low so a decision boundary with a large margin is chosen at the expense of a greater number of misclassifications. If c is large, SVM tries to minimize the number of misclassified examples due to high penalty which results in a decision boundary with a smaller margin. Penalty is not same for all misclassified examples. It is directly proportional to the distance to decision boundary.

Before introducing **gamma**parameter, we need to talk about kernel trick. In some cases, data points that are not linearly separable are transformed using kernel functions so that they become linearly separable. Kernel function is kind of a similarity measure. The inputs are original features and the output is a similarity measure in the new feature space. Similarity here means a degree of closeness. It is a costly operation to actually transform data points to a high-dimensional feature space. The algorithm does not actually transform the data points to a new, high dimensional feature space. Kernelized SVM compute decision boundaries in terms of similarity measures in a high-dimensional feature space without actually doing a transformation. I think this is why it is also called **kernel trick**.

One of the commonly used kernel functions is radial basis function (RBF). Gamma parameter of RBF controls the distance of influence of a single training point. Low values of gamma indicates a large similarity radius which results in more points being grouped together. For high values of gamma, the points need to be very close to each other in order to be considered in the same group (or class). Therefore, models with very large gamma values tend to overfit. Following visualizations explain the concept better:





The first image represents the case with a low gamma values. Similarity radius is large so all the points in the colored regions are considered to be in the same class. For instance, if we have a point the right bottom corner, it is classified as “green” class. On the other hand, the second image is the case with large gamma. For data points to be grouped in the same class, they must fall in the tight bounded area. Thus, a small noise may cause a data point to fall out of a class. Large gamma values are likely to end up in overfitting.

As the gamma decreases, the regions separating different classes get more generalized. Very large gamma values result in too specific class regions (overfitting).

Gamma vs C parameter

For a linear kernel, we just need to optimize the c parameter. However, if we want to use an RBF kernel, both c and gamma parameter need to optimized simultaneously. If gamma is large, the effect of c becomes negligible. If gamma is small, c affects the model just like how it affects a linear model. Typical values for c and gamma are as follows. However, specific optimal values may exist depending on the application:

0.0001 < gamma < 10

0.1 < c < 100

*It is very significant to remember for SVM that the input data need to be normalized so that features are on the same scale and compatible.*

**Why Feature Scaling is Required in SVM?**

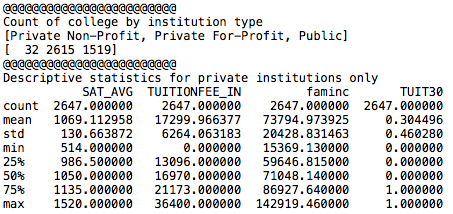
**Effect of Feature Standardization on Linear Support Vector Machines**

Because Support Vector Machine (SVM) optimization occurs by minimizing the decision vector *w,*the optimal hyperplane is influenced by the scale of the input features and it’s therefore recommended that data be standardized (mean 0, var 1) prior to SVM model training. In this post, I show the effect of standardization on a two-feature linear SVM, along with some (hopefully) practical interpretations of the results along the way.

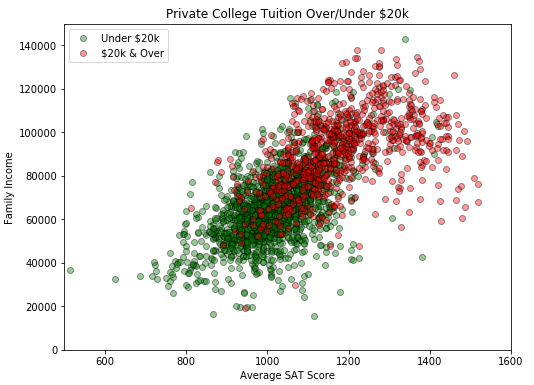
The data I’ll use here is from the [College Scorecard](https://collegescorecard.ed.gov/), a freely available US Department of Education data source with various statistics for US colleges. I used a version of the College Scorecard data that’s [hosted by Kaggle](https://www.kaggle.com/kaggle/college-scorecard).

Using this datasource, I’ll train an SVM to predict whether a college’s in-state tuition is greater than or less than $20k per year based on (1) average SAT score of enrolled students and (2) average family income of enrolled students. This model is a bit contrived and simplistic — my primary goal is not to successfully predict the target class per se, but rather to show the impact of feature standardization on the resulting model.

In this dataset, there are 4,166 college with non-NULL average SAT score & family income data. There are a significant number of colleges that have NULL values for these features, which I excluded from my analysis:



An in-state tuition of $20k per year is somewhere between the 50th and 75th percentile for that variable — so there’s a healthy amount of colleges on both sides of this target class.



Above, I’ve plotted college tuition rates by average SAT score and family income. What should be immediately clear is that this data isn’t linearly separable. In a linear SVM, there’s no hyperplane that will cleanly split the datapoints by tuition class — there will always be colleges on the ‘wrong side’ of any decision hyperplane chosen by the model. The sklearn SVM implementation will search for an optimal hyperplane (and +1/-1 margin hyperplanes) subject to a user-specified hyper-parameter ‘C’ that determines the degree of penalization for datapoints on the ‘wrong side’ of the margin hyperplanes.

For each group of features (non-standardized, then standardized) I ran 11 linear SVM models with C coefficients 5\*\*-3, 5\*\*-2,…5\*\*3.

Immediately apparent was a dramatic difference in runtime for fitting these two models:

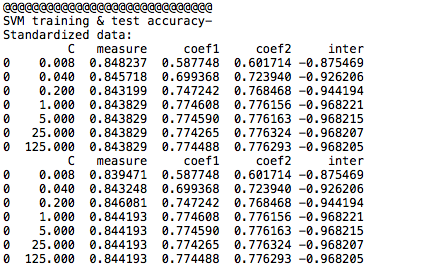
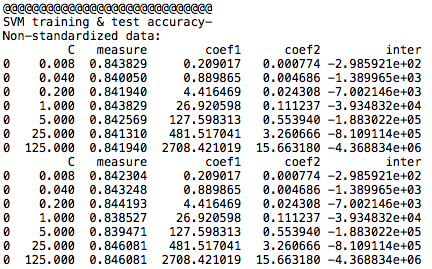
https://miro.medium.com/max/60/1*9tvhL-4pdMRsPEUlzPOIIQ.png?q=20

https://miro.medium.com/max/371/1*9tvhL-4pdMRsPEUlzPOIIQ.png

https://miro.medium.com/max/60/1*WNGFMUWy8Py6Fm3QZB1jcw.png?q=20

https://miro.medium.com/max/377/1*WNGFMUWy8Py6Fm3QZB1jcw.png

The following two tables capture the class prediction accuracy (field ‘measure’) of each fitted SVM. The top set of results represents a 60% training data subset while the bottom set of results represents a 40% test or holdout data subset. I’ve also included the fitted decision vector *w*and scalar*b* for each run. The accuracy rates across all models are similar, but the fitted values for *w*and *b* are dramatically different.



The below 6 plots show the fitted SVM hyperplanes and (+1,-1) margins for various values of penalty coefficient C. The top three plots show the fitted hyperplane for non-standardized features, while the bottom three plots show the fitted hyperplane for standardized features.

It’s not difficult to see that the non-standardized data produces decision hyperplanes that are highly sensitive to coefficient C. Each of the top three SVMs produce decision hyperplanes that are noticeably different to the naked eye. The standardized data produces much more consistent SVM hyperplanes across hyperparameters. A difference among the bottom plots is that the range for the (+1,-1) margin hyperplanes is wider with lower values of C. Intuitively, since datapoints on the ‘wrong side’ of the margin hyperplanes are penalized less in the low-C SVM models, the model logic has greater freedom to fit margin boundaries that are further from the decision hyperplane.

