support vector classification

andrew zastovnik and ryan shiroma

SAN JOSÉ STATE UNIVERSITY

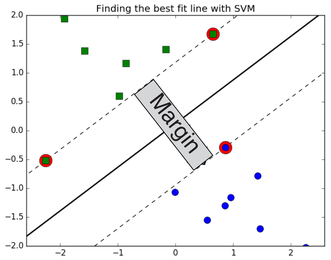
introduction

Classification with Support Vector Machines is a classy solution to the problem of classifying classes, or distinguishing between groups based on data. The fundamental idea of SVM is to find a function that can achieve the greatest separation between classes so that we can plug in the data from an observation and make a prediction of its class. SVM based classification is said to have been developed by a gentleman named Vladimir Vapnik in the 1970's has since become popular because of its beauty and predictive power. It has been applied with great success to biological problems such as identifying protein classes and recognizing hand written digits.

svm: method

In it's simplest form, SVM attempts to find an optimal hyperplane boundary that linearly separates two classes of data.

SVM aims to find a line that maximize a "margin" length between the separation line and the support vectors in each respective class. This line is represented as 0=wx+b.



To find this optimal margin(ie. solve for w and b), first let:

$$y_i = 1 \text{ when point } i \text{ is in class 1}$$

$$y_i = -1 \text{ when point } i \text{ is in class 2}$$

Then solve the maximization problem:
The optimal w, b, and prediction y can be found to be:

$$\max_{\lambda_1,\ldots,\lambda_n} \sum_i \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{where } \lambda_i \geq 0, \sum \lambda_i y_i = 0$$

$$\mathbf{w} = \sum_i \lambda_i y_i \mathbf{x}_i \qquad \hat{y}_i = \text{sign}(\mathbf{w}\mathbf{x}_i + b)$$

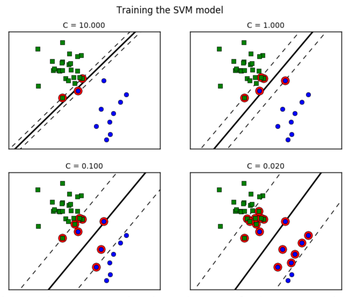
$$b = y_j - \mathbf{w}\mathbf{x}_j \text{ (where } j \text{ is a support vector)}$$

svm: regularization

The best fit line can be highly influenced by outliers being chosen as support vectors. In this case we should utilize some regularization to take into account more support vectors, thus hopefully getting a better fit on test data. To do this we introduce a new constraint to the to the maximation problem.

$$\max_{\lambda_1,\ldots,\lambda_n} \sum_i \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{where } \lambda_i \geq 0, \sum_i \lambda_i y_i = 0, \boxed{\lambda_i < C}$$



By increasing the regularization of the model, (smaller C means greater regularization), we increase the number of support vectors and can recieve a better fit line. However, overregularization can result in underfitting. A best value of C can be chosen with cross validation on the training set.

svm: kernelization

Kernel SVM may be more appropriate when the data isn't linearly seperable. Kernelization is a method in which we represent data in a higher dimensional space where it may then become linearly seperable. The two common types of kernels used are polynomial kernels and Gaussian kernels.

$$\Phi : \mathbb{R}^d \to \mathbb{R}^l \text{ (where } l > d)$$

A kernel function, is used on the data vector , x_i so we just need to replace x_i with $\Phi(\mathbf{x}_i)$ in the maximization problem.

$$\max_{\lambda_1,\ldots,\lambda_n} \sum_i \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

$$\mathbf{w} = \sum_i \lambda_i y_i \Phi(\mathbf{x}_i)$$

$$b = \frac{1}{y_j} - \Big(\sum_i \lambda_i y_i \Phi(\mathbf{x}_i)\Big)\Phi(\mathbf{x}_j)$$

Calculations with this higher dimensional data can prove to be impractical. Therefore using the "kernel trick" simplifies these calculations by quickly by only computing dot products of kernelized data.(For SVM, we only need dot products to calculate predictions.)

$$k(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$$

The equation to classify new data then becomes:

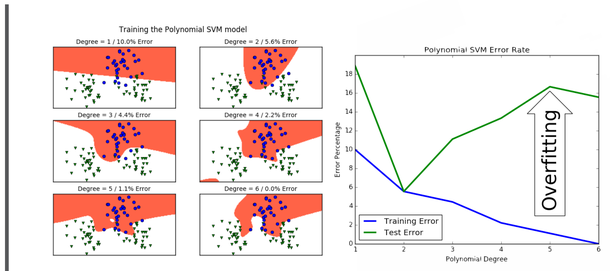
$$\hat{y}_j = \text{sign}(\sum_i \lambda_i y_i k(x_i, x_j) + b)$$

Polynomial Kernel:

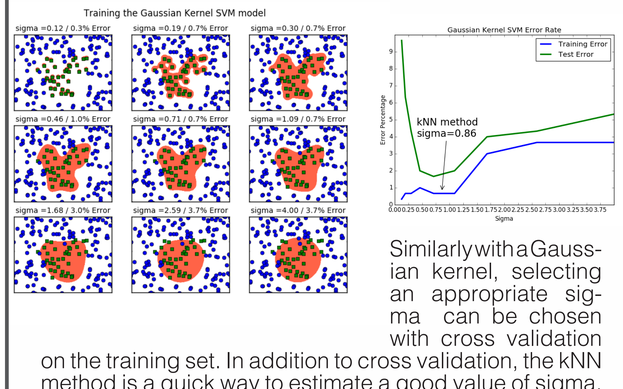
$$k(x_i, x_j) = (1 + x_i \cdot x_j)^p$$

Gaussian Kernel:

$$k(x_i, x_j) = exp(-||x_i - x_j||_2^2/(2\sigma^2))$$



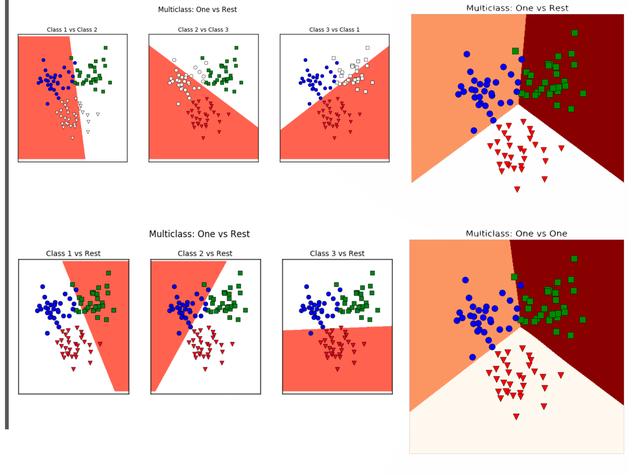
A polynomial curve for the data above is more suitable as a line of separation. However, finding an approprate degree polynomial (p) is essential to prevent overfitting. Here we see that a cubic polynomial or higher results in overfitting.



Similarly with a Gaussian kernel, selecting an appropriate sigma can be chosen with cross validation on the training set. In addition to cross validation, the kNN method is a quick way to estimate a good value of sigma.

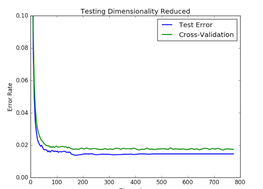
svm: multiple classes

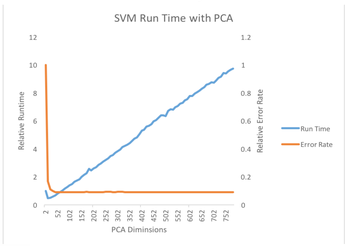
Extending the SVM method to multiple-class data sets can be accomplished by the "one vs one" or "one vs rest" methods.
One vs One:
Models are run comparing each class to every other class. The highest vote count is chosen is the predicted class.
One vs Rest:
Each class is compared to all other classes. The class with the largest probability is chosen as the predicted class.



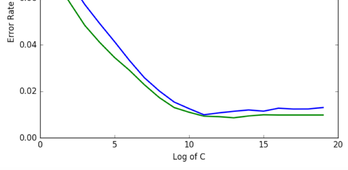
applications to MNIST

Many other classification techniques such as K Nearest Neighbors and logistic regression benefit from dimensionality reduction techniques such as principal component analysis (PCA). It is also possible that SVM classifiers can also benefit from PCA. So to test this out we ran one-vs-one SVM with m-fold cross validation and well as on the test data with many different levels of dimensionality reduction.





As you can see, SVM did not see any benefits from dimensionality reduction like we saw in the other cases. What about run time? There was a significant reduction in runtime by using PCA. Running SVM with all dimensions took around 10 times as long as running it with 50 dimensions.



we can select an appropriate C by cross validation.

Since the MNIST dataset is highly non-linear, we see that using a kernel is preferred. In our tests, the Gaussian kernel performed best under a one-vs-rest multiclass method achieving around a 1.5% error rate.

references

Chen, Guangliang. "Lecture 7: Support Vector Machines". Math 285 - Spring 2016.