SVM Kernel Optimization: An Example in Yeast Protein Subcellular Localization Prediction

Ṭaráz E. Buck
Computational Biology Program
tebuck@andrew.cmu.edu

Bin Zhang
School of Public Policy and Management
binzhang@cmu.edu

Abstract

Localization of proteins, a flourishing area in bioinformatics, can help us understand their respective functions. Currently there exist a number of localization approaches based on machine learning algorithms, and support vector machines (SVMs) have been used extensively. However, in terms of kernel optimization, a critical step in SVM design, there is no well-established systematic method so far. In this paper, we apply the Levenberg-Marquardt (LM) algorithm in kernel optimization, an algorithm that we believe will have better optimization performance than simple gradient descent and that hasn't been used in this field according to our knowledge, and test it using protein location data from yeast. We then experimentally compare the performance of our optimized system with others. Results show that automated parameter optimization can improve classification accuracy. Further research can complement such a method by controlling over-fitting.

1 Introduction

The prediction of protein subcellular localization is a thriving area in bioinformatics and a critical step of genomic data exploration. It is the foundation of any further research in protein functional characterization. With the help of machine learning, cellular protein localization accuracies are improving significantly. Several methods and systems have been developed using decision trees, SVMs, etc. Many researchers have issued different multiple-class SVM classifiers, using the one-versus-one [5], one-versus-all [3], and DAGSVM [12] methods. However, many researchers design their kernels by trying a finite number of different parameter values manually. Such a method is inefficient and often inaccurate. The objective of our research is to design a more automatic method for optimizing kernel function parameters. Such methods also provide a guideline for designing classifiers with better performance and comparing kernel functions.

In this project, we optimize SVM classifiers for predicting protein localization and test it against Horton's and Nakai's Yeast data set. Our features consist of 8 input scores for the protein sequence that are characteristic of a given localization pattern. Detailed explanation of features can be found in section 3.1. In section 2, we review relevant literature in protein localization using machine learning. We introduce our chosen kernel optimization method, the Levenberg-Marquardt algorithm, in section 3. In section 4, we implement our algorithm to optimize kernels, then design an SVM classifier using this kernel and conduct experiments to show our SVM method is numerically superior to other methods. Experiment results show that our method attains an overall prediction accuracy of 74.60%, which is much higher than 54.9%, the accuracy rate of the system by Nakai [6], the data provider. Finally we present discussion and the prospect of future work in section 5.

2 Background and literature review

2.1 Protein localization prediction

Machine learning has been used widely in computational biology and bioinformatics. Given the size and complexity of these data sets, most researchers are compelled to use machine learning techniques to classify genes and proteins. A number of systems have been developed that support automated prediction of subcellular localization. Some widely implemented methods are: training artificial neural networks (ANN) such as NNPSL [13]; exploiting the existence of peptide signals, which are short sub-sequences of approximately 3 to 70 amino acids, to predict specific cell locations, such as TargetP [4]; and otherwise extracting data, such as from homologs, and using a classifier on the sequence features. Other approaches include LOCkey [11] and then support vector machines (SVM) like SubLoc [7], which is a method attracting more attention.

2.2 Kernel optimization

Kernel selection plays an important role in SVM training and classification. A properly designed kernel function can minimize generalization error, accelerate convergence speed, and increase prediction accuracy. There are two common optimization methods, adding parameters and kernel alignment. Adding parameters is a method for putting additional parameters in the kernel and optimizing those parameters so as to improve the performance. ”These parameters could be simple as the β parameter in the radial basis kernel, weight each dimension of the input vectors, or more flexible as finding the best convex combination of basic kernels. [8]” The critical part of this method is choosing the measure we would optimize. We usually choose cross-validation or geometric margin. Geometric margin must be used accompany with normalization, since some linear operation of the feature vector will change the geometric margin. The most widely used optimization approach is kernel alignment. Alignment can be understood as similarity between Gram matrices. It adjusts the kernels parameters to align them to a target kernel.

2.3 Gradient descent

Gradient descent is widely used in non-linear optimization. In most cases, gradient descent is sufficient “to find satisfactory weights for a multi-layer perceptron (MLP) or a radial basis function (RBF) network, adapting possible hyper-parameters requires an estimation of the gradient after each training process, and then a downhill step toward a local minima could be done” [1]. However, gradient descent also has several shortcomings. First, gradient descent is time consuming because it takes infinitesimal steps in the direction of the gradient. Second, sometimes we cannot confirm which step size would be optimal. Third, the SVM needs a lot of training iterations until a solution is found. Fourth, in many cases, gradient descent does not produce reproducible results. It is not necessary to produce a downhill direction toward a minimum and “any discontinuity can lead to an arbitrary step along the error surface” [1].

Many current research deals with hyper-parameter adaption in neural networks. Most of this research consider fixing parameters involved in the training process that are not. Given the disadvantages of gradient descent, we use the Levenberg-Marquardt algorithm in kernel optimization. It has been used in neural networks but hasn't been put to use in kernel optimization. A detailed introduction is in the next section.

3 Methods and Systems

In this paper, we implement and evaluate the performance of the LM algorithm, a method for finding minima of sum-of-squares functions. Specifically, we are applying it to kernel optimization in support vector machines (SVMs). Its behavior varies between gradient descent and Gauss-Newton optimization, depending on the topology of the space it searches. It resembles the former method when the current solution is still far from the minimum, then changes to the latter when it is closer to the optimum. [9]

More [10] describes the LM algorithm as more robust than several other optimization methods, and he states that it shows "strong convergence properties." We hope that the LM algorithm will quickly converge on the distribution of our test data set, the Yeast Protein Database, which is introduced later.

The LM algorithm is listed below in pseudo-code.

```
k = 0
nu = 2
p = p_0
A = J' * J
epsilonp = x - f(p)
g = J' * epsilon_p
stop = (g' * g <= epsilon_1)
mu = tau * max(A_ii)
while (~stop and k < k_max)
    k = k+1
    do
        solve ((A + mu * I) * delta_p = g)
        if (delta_p' * delta_p <= epsilon_2 * p' * p)
            stop = true
        else
            p_new = p + delta_p xmfpn = x - f(p_new)
            rho = (epsilon_p' * epsilon_p - xmfpn' * xmfpn)
               / (delta_p' * (mu * delta_p + g))
            if (rho > 0)
                p = p_new
                A = J' * J
                epsilon_p = x - f(p)
                g = J' * epsilon_p
        stop = (g' * g <= epsilon_1) or (epsilon_p' * epsilon_p <= epsilon_3)
        mu = mu * max(1/3, 1 - (2 * rho - 1)$\hat{3}$)
        nu = 2
      else
        mu = mu * nu
        nu = 2 * nu
      endif
    endif
  until (rho > 0 or stop)
end
p_plus = p
return p_plus
```

4 Experiments

4.1 Data

We are using the Yeast database from the UCI ML Repository. There are 1484 records, each with eight feature values. These features are for signal sequence recognition such as transmembrane segments, mitochondrial proteins, endoplasmic reticulum (ER) recognition, peroxisomal protein recognition, vacuolar protein recognition, and nuclear protein recognition.

1. MCG: McGeoch's method for signal sequence recognition. This method calculates discriminant score using length of N-terminal positively-charged region (H-region), peak value of central hydrophobic region (H-region), and net charge of N-region. High discriminant score indicates high possibility of signal sequence.

2. GVH: von Heijne's method for signal sequence recognition. This method uses weight-matrix and the cleavage sites consensus pattern to detect signal-anchor sequences. High score indicates high possibility of having cleavable signal sequence.

3. ALM: Feature for transmembrane segments recognition. Score of the ALOM membrane spanning region prediction program. It indicates whether a segment is transmembrane or peripheral.

4. MIT: Feature for mitochondrial proteins recognition. Through discriminant analysis of the N-terminal region amino acid content, we can differentiate between mitochondrial and non-mitochondrial proteins.

5. ERL: Feature for endoplasmic reticulum (ER) recognition. It is a binary value indicating "HDEL" substring presence, which is a signal of endoplasmic reticulum lumenal protein.

6. POX: Feature for peroxisomal protein recognition. We use peroxisomal-matrix targeting sequences as targeting signal in the C-terminus.

7. VAC: Feature for vacuolar protein recognition. It is a score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins. In yeasts, vacuoles contain numerous hydrolytic enzymes.

8. NUC: Feature for nuclear protein recognition. Indicator of nuclear and non-nuclear proteins. Nuclear proteins occupy the majority in yeast genome, so the the total prediction

Table 1: Results of classifications and kernels

Kernels	Best Accuracy	Average Accuracy
Linear	56.04%	56.04%
Polynomial	74.60%	66.81%
RBF	70.34%	59.01%

accuracy depends a lot on nuclear proteins prediction accuracy.

Through the features introduced above, we consider 10 localization patterns within a yeast cell as below: cytosolic (CYT), endoplasmic reticulum lumen (ERL), extracellular (EXC), membrane protein with signal cleaved (ME1), membrane protein with signal uncleaved (ME2), membrane protein without N-terminal signal (ME3), mitochondrial (MIT), nuclear (NUC), peroxisomal (POX), vacuolar (VAC).

4.2 Results

We now compare the behavior of different kernels with the Yeast data set. We randomly chose a training set of 1126 records and a testing set of 358. We trained SVM classifiers using different kernels, namely the linear, polynomial, and radial basis function (RBF or Gaussian) kernels. We set the slack penalty to $C = 1$ and verified results with ten-fold cross-validation. The accuracies are listed in Table 1. A polynomial kernel yielded the highest average accuracy of 74.60%, an improvement of 19.7% over Horton and Nakai's classifier's 54.9%. Figure 1 displays a plot of a polynomial-kernel SVM classifier over our data. Our polynomial degrees are between two and ten, so we do not expect any issues with over-fitting. The best and average accuracies in Table 1 are taken with respect to a range of different kernel parameters.

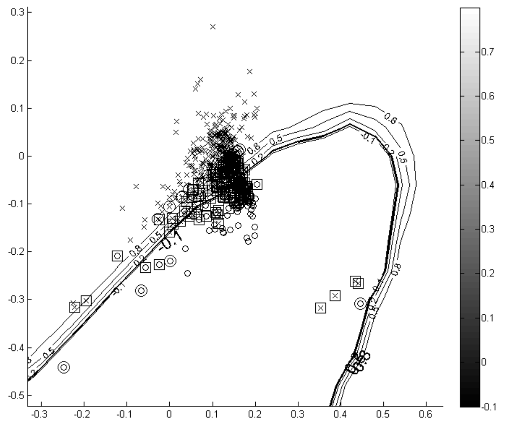

Figure 1: 1-vs-1 SVM decision boundary with 10th degree polynomial kernel

Table 2 shows accuracies for the RBF kernel. The RBF kernel can fit the data differently depending on its parameter, with classification improving as a decreases.

Table 2: Accuracies of different parameters, RBF kernel

Parameter a	**Accuracy**
0.125	70.34%
0.25	64.21%
0.5	61.19%
1	56.13%
2	43.16%

Slack penalty $C = 1$

Table 3: Accuracies before and after optimization

Kernel	**Initial Parameter**	**Accuracy without LM**	**Optimized Parameter**	**Accuracy with LM**	$\Delta_{Accuracy}$
Polynomial	2	60.21%	2	60.21%	0.00%
	10	74.60%	10	74.60%	0.00%
RBF	2	43.16%	2	43.16%	0.00%
	1	56.13%	0.97	56.22%	0.89%
	0.5	61.19%	0.5	61.01%	-0.18%
	0.25	64.21%	-0.0044	100%	35.79%
	0.125	70.34%	0.046	91.74%	21.40%

We optimized the polynomial and RBF kernels with the LM algorithm to contrast the performance with unoptimized kernels. Results are shown in Table 3. Polynomial kernels showed no difference between the initial and final parameters and thus the accuracy. The algorithm consistently failed to detect any gradient for polynomial kernels using both default and other settings for the finite differencing. We also found that RBFs showed a mean increase of 11.42% in classification accuracy, demonstrating LM's ability to properly locate a better parameter value. Given other kernels, the technique can be used to select a multidimensional parameter to aid in supervised optimization.

We also noticed the possibility that the overall increase in RBF accuracy is caused by over-fitting, a known issue with these kernels [2]. In the last two cases listed in Table 3, the LM algorithm significantly reduced the RBF parameter to the order of thousandths or ten thousandths, allowing many instances of the kernel to be placed by the SVM software. Given a large enough number of instantiated kernels and a small enough parameter, it is possible to properly classify all data points by creating a convoluted transformation to the higher dimensional space. However, upon examining a classifier trained on the first two principle components of the data set that showed high classification success, we recognize the need to do further investigation into this issue.

We have shown that the accuracies for some classifiers can be optimized, so LM can be useful as a method to optimize SVM kernels. Over-fitting, and determining when it occurs, is a major issue with many learning algorithms. To allow techniques such as LM-based parameter optimization to be properly utilized, we suggest that optimization requires another heuristic function to approximate the level of over-fitting associated with a particular decision boundary. Of course, this is a challenging problem given the vast variation in type of data, the number of samples present in the many data sets available, and individual preferences for elegant solutions.

5 Conclusion

In a multiple-category classification problem, it is very important for an SVM classifier to use optimal kernel parameters because the proper clustering of the data depends on the ability of kernel functions to properly partition the higher-dimensional space. We studied and compared different kernels against the Yeast data set, and we found that the polynomial kernel yielded the best classification accuracy. We also proposed an empirical error based method that uses a Levenberg-Marquardt algorithm to optimize the parameters, and test the accuracy before and after optimization. However our results did not show significant improvement for all kernels. We provided analytical explanation. On the other hand, Ayat et al. used the quasi-Newton method to optimize SVM kernels and had positive results [1]. We suspect that their results depend significantly on their data set. We have identified their NIST database, so a future task would be to obtain their data and test our algorithm against them.

References

[1] N. E. Ayat, M. Cheriet, and C. Y. Suen. Optimization of the svm kernels using an empirical error minimization scheme. pages 354–369, Niagara Falls, Canada, 2002.

[2] L. Bi, H. Huang, Z. Zheng, and H. Song. New heuristic for determination Gaussian kernels parameter. In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, pages 4299–4304, Guangzhou, China, 2005.

[3] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, L. Yam, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: A case study in handwriting digit recognition. In *Proceedings of International Conference Pattern Recognition*, pages 77–87, Israel, 1994.

[4] O. Emanuelsson, H. Nielsen, S. Brunak, and G. von Heijne. Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *Journal of Molcular Biology*, 300:1005–1006, 2000.

[5] J. Friedman. *Another Approach to Polychotomous Classification*. Department of Statistics, Stanford University, Stanford, CA, 1996.

[6] P. Horton and K. Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. In *Proceeding of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 109–115, St. Louis, MO, 1996.

[7] S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17:721–728, 2001.

[8] T. S. Jaakkola. *Machine Learning*. Massachusetts Institute of Technology, Cambridge, MA, 2006.

[9] M. I. A. Lourakis. *A Brief Description of the Levenberg-Marquardt Algorithm Implemented by levmar*. Foundation for Research and Technology, Greece, 2005.

[10] J. J. More. The levenberg-marquardt algorithm: implementation and theory. In G. A. Watson, editor, *Numerical Analysis, Lecture Notes in Mathematics*, pages 105–116. Springer-Verlag, Heidelberg, 1977.

[11] R. Nair and B. Rost. Mimicking Cellular Sorting Improves Prediction of Subcellular Prediction. *Journal of Molecular Biology*, 348(1):85–100, 2005.

[12] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, page 547C553. MIT Press, Cambridge, MA, 2000.

[13] A. Reinhardt and T. Hubbard. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26:2230–2236, 1998.