SUPPORT VECTOR MACHINES

A PREPRINT

David Wilson
University College London
London, United Kingdon
david.wilson.15@ucl.ac.uk

May 20, 2019

ABSTRACT

Support Vector Machines are supervised learning models, used for analyzing data for classification and regression. In this paper a summary of the mathematical formulation of Support Vector Machines is given. We then present a simulation study using models with Gaussian and Laplace kernels that classify a synthetic non-linear data. We investigated performance and training of the models by solving the dual problem using Sequential Minimal Optimization and a variant of Augmented Lagrangian method. And lastly, we showed the generalization of our models using MNIST data.

Keywords Support Vector Machines · Kernels · Supervised Learning · Optimization

1 Introduction

In machine learning, Support vector machines (SVM) are supervised learning models that analyze data for classification and regression analysis [1]. For binary classification the set of possible values that label can have is binary. In this article we denote them as -1, +1. In other words we consider predictors of the form [2]:

$$f : R^D \to \{-1, +1\} \tag{1}$$

We represent each example x_n as a feature vector of D real numbers. These labels are usually referred to as positive and negative classes, respectively.

In this summary, we consider the famous approach Support Vector Machines (SVM) which can solve this binary classification task. In this task, we have a set of examples x_n in R^D along with their corresponding labels y_n in $+1, -1$. Given the training data consisting of example labelled pairs $(x_1, y_1), \ldots, (x_N, y_N)$, we would like to estimate the parameters of the model that result in the best classification error [2].

2 Mathematical Formulation

2.1 Hyperplanes

Let x_i and x_j denote two examples. We compute the similarity between them using an inner product $\langle x_i, x_j \rangle$. In case of the binary classification, we want to partition the space of our data (R^D) into two parts corresponding to the positive and negative classes, respectively [1]. The simplest partition is to split it into two halves using a hyperplane: Let $x \in R^D$ be an element of the data space, and consider the function $f : R^D \to R^D$ parameterised by $b \in R$ and $w \in R^D$ as follows[1]:

$$f(x) = \langle w, x \rangle + b \tag{2}$$

Hyperplanes are affine subspaces. So, we define the hyperplane that separates the two classes in our binary classification problem as:

$$\{x \in R^D : f(x) = 0\} \tag{3}$$

Choosing any two examples x_1 and x_2 and showing that the vector between them is orthogonal to w, it can be shown that w is a normal vector to the hyperplane:

$$f(x_1) - f(x_2) = \langle w, x_1 \rangle + b - (\langle w, x_2 \rangle + b) = \langle w, x_a - x_b \rangle \tag{4}$$

Since x_1 and x_2 are on the hyperplane, this implies that $f(x_1) = 0$ and $f(x_2) = 0$, and hence $\langle w, x_1 - x_2 \rangle = 0$. Since the two vectors are orthogonal, their inner product must be zero, so we get that w is orthogonal to any vector on the hyperplane. We classify a test example as negative or positive depending on which side of the hyperplane it lies. The equation(4) defines the positive and negative side of the hyperplane. When training the classifier, we want to ensure that the examples with negative and positive labels are on the positive and negative side of the hyperplane respectively, [2], i.e,

$$y_n(\langle w, x_n \rangle + b) \geq 0. (y_n \in \{+1, -1\}) \tag{5}$$

2.2 Support Vector Classification

2.2.1 Linearly Separable case

Consider a problem involving the classification of two cluster of points, where there exists a hyperplane which linearly separates one cluster from the other with no errors. There are infinitely many possible hyper-planes that solve this, so we want to choose the one which has the largest margin, largest possible distance from both classes, and the smallest distance from each class to separating hyperplane called the margin [3]. This problem is expressed as:

$$\max_{w,b}(\text{margin}) = \max_{w,b}(\frac{2}{\|w\|}) \tag{6}$$

subject to:

$$\begin{array}{lll} \min(w^T x_i + b) = 1, & \text{if} & i : y_i = +1 \\ \max(w^T x_i + b) = -1, & \text{if} & i : y_i = -1. \end{array}$$

resulting in the classifier

$$y = \text{sign}(w^T x_i + b) \tag{7}$$

where $+1$ and -1 are assigned for positive and negative arguments respectively. Rewriting it gives:

$$\max_{w,b}(\frac{1}{\|w\|}) \qquad \text{or} \qquad \min_{w,b}(\|w\|^2) \tag{8}$$

subject to

$$y_i(w^T x_i + b) \geq 1.$$

2.2.2 Non-linear case

When the classes are not linearly separable, we might want to allow some number of errors in the classifier. Hence, we trade off the margin errors by maximizing the margin. This would mean optimizing:

$$\min_{w,b}(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} I[y_i(w^T x_i + b) < 1]) \tag{9}$$

where C controls the trade-off between maximum and the loss and $I(X)$ = 1 if X is true and 0 otherwise. This is a combinatorial optimization problem, which is very expensive to solve. To overcome this we replace the indicator function with a convex upper bound,

$$\min_{w,b}(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \theta[y_i(w^T x_i + b) < 1]) \tag{10}$$

where we used the hinge loss:

$$\theta(\alpha) = (1 - \alpha)_+ = 1 - \alpha \quad \text{if} \quad 1 - \alpha > 0 \quad \text{otherwise} \quad 0 \tag{11}$$

When substituting the hinge loss, we get

$$\min_{w,b,\xi}(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i) \tag{12}$$

subject to

$$\xi_i \geq 0 \qquad y_i(w^T x_i + b) \geq 1 - \xi_i$$

We can then write and solve the Lagrangian [4] for this problem:

$$L(w, b, \xi, \alpha, \lambda) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - y_i(w^T x_i + b) - \xi_i) + \sum_{i=1}^{n}\lambda_i(-\xi_i) \tag{13}$$

with dual constraints $\alpha_i \geq 0, \lambda_i 0$. Minimizing with respect to primal variables w,b and ξ. Derivative with respect to w:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0 \quad \rightarrow \quad w = \sum_{i=1}^{n}\alpha_i y_i x_i. \tag{14}$$

Derivative with respect to b:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n} y_i \alpha_i = 0. \tag{15}$$

Derivative with respect to ξ_i:

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i \quad \text{where} \quad \alpha_i = C - \lambda_i \tag{16}$$

We replace the final constraint by noting $\lambda_i 0$, thus $\alpha_i \leq C$. Now we can write the dual function, by substituting the derivatives into, to get:

$$g(\alpha, \lambda) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - y_i(w^T x_i + b) - \xi_i) + \sum_{i=1}^{n}\lambda_i(-\xi_i)$$

$=\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$

- b $\sum_{i=1}^{m}\alpha_i y_i + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i\xi_i - \sum_{i=1}^{m}(C - \alpha_i)\xi_i$

$= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\alpha_i\alpha_j y_i y_j x_i^T x_j.$

(17)

This means our goal is to maximize the dual,

$$g(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j. \tag{18}$$

subject to constraints $0 \leq \alpha_i \leq C, \sum_{i=1}^{n} y_i\alpha_i = 0$. By now, we have only defined the solution for w, but not for b (offset). This is quite simple to compute: for the margin SVs, we have $1 = y_i(w^T x_i + b)$. So we can get b from any of these, or take an average.
The resulting optimization problem for the dual is a convex constrained quadratic optimization problem, which can be solved using suitable optimization algorithms [4].

2.3 Kernels

We can define a maximum margin classifier in the feature space. For this we can simply write the hinge loss function [5]:

$$\min w(\frac{1}{2}\|w\|_{\mathcal{H}}^2 + C\sum_{i=1}^{n}(1 - y_i\langle w, k(x_i, .)\langle_{\mathcal{H}})_+) \tag{19}$$

for the Reproducible Kernel Hilbert space (RKHS) $\mathcal{H}$ with kernel $k(x, .)$. When we use kernels we take a use of the representer theorem, which says that the optimal weight vector in the primal is a linear combination of the examples[6]:

$$w = \sum_{i=1}^{n}\beta_i k(x_i,). \tag{20}$$

Here, maximizing the margin is equivalent to $\|w\|_{\mathcal{H}}^2$, i.e. for many RKHSs this is enforcing smoothness. Substituting the and introducing ξ_i variable we get:

$$\min_{w,b}(\frac{1}{2}\beta^T K\beta + C\sum_{i=1}^{n}\xi_i) \tag{21}$$

with the matrix K having i, j th entry $K_{ij} = k(x_i, x_j) \geq 1 - \xi_i$ Hence, the primal variable w is replaced with β. The problem remains convex since K is positive definite. After some trivial calculations, the dual becomes

$$g(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i, x_j), \tag{22}$$

subject to constraints $0 \leq \alpha_i \leq C$, and the decision function takes the form

$$w = \sum_{i=1}^{n} y_i\alpha_i k(x, .). \tag{23}$$

2.3.1 Gaussian Kernel

Let the instance space be R and consider the mapping ψ where for each non-negative integer n there exists an element $\psi(x)_n$ that equals $\frac{1}{\sqrt{n!}}e^{\frac{-x^2}{2}}x^n$. Then [5]

$$\langle\psi(x), (x')\rangle = \sum_{n=0}^{\infty}(\frac{1}{\sqrt{n!}}e^{\frac{-x^2}{2\sigma^2}}x^n)(\frac{1}{\sqrt{n!}}e^{\frac{-(x')^2}{2\sigma^2}}(x')^n) = e^{-\frac{x^2+(x')^2}{2\sigma^2}}\sum_{n=0}^{\infty}(\frac{(xx')^n}{n!}) = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \tag{24}$$

So the feature space is infinite-dimensional while the kernel is simple. More generally, for $\sigma > 0$ the Gaussian kernel is defined as

$$K(x, x') = e^{-\gamma\|x-x'\|^2} \tag{25}$$

Basically, the Gaussian kernel sets the inner product in the feature space between x, x' to be close to zero if the instances are far away from each other and close to 1 if they are close to each other. The hyper-parameter γ simply controls the scale determining what we mean by close. The kernel in the form of equation (24) is commonly known as the Radial Basis Function(RBF). In the study we use the kernel in the form (25), but we may use the terms Gaussian kernel and RBF interchangeably in this paper.

2.3.2 Laplace Kernel

As a second kernel, we consider the Laplace kernel [5]:

$$K(x, x') = e^{-\frac{\|x-x'\|}{\sigma}} \tag{26}$$

This is very similar to the Gaussian kernel, but its Fourier transform is a Cauchy distribution which drops much slower than the exponential function in the Fourier transform of a Gaussian kernel. This means that the function will have more high-frequency components. As a result, the function given by the Laplace kernel is "rougher" than the one given by the Gaussian kernel [6].

2.4 Optimization methods

2.4.1 Sequential Minimal Optimization (SMO)

First introduced by Platt (1998) [7], proposed to always use the smallest possible working set size, two elements. Each Successive quadratic programming sub problem has two variables. The equality constraint makes it a one dimensional optimization problem. The computation of the Newton step is very fast because the direction u contains only two non zero coefficient. The asymptotic convergence and finite termination properties of this particular decomposition method. SMO breaks the problem into a series of smallest possible sub-problems, that are solved then analytically. Due to the presence of the linear equality constraint involving the two Lagrange multipliers α_i, the smallest problem involving two such multipliers [4]. Hence, for any two multipliers α_1 and α_2, the constraints are reduced to:

$$0 \leq \alpha_1, \alpha_2 \leq C,$$
$$\mathrm{y}_1\alpha_1 + y_2\alpha_2 = k$$

A simplistic pseudo-code for this algorithm is[7]:
1. Find a Lagrange multiplier α_1 that violates the Karush–Kuhn–Tucker (KKT) conditions conditions for the optimization problem.
2. Pick a second multiplier α_2 and optimize the pair(α_1, α_2).
3. Repeat steps 1 and 2 until convergence.

The SMO algorithm is considered the state of the algorithm for training SVMs.

2.4.2 Unconstrained Augmented Lagrangian method(ALM)

Consider a generic optimization problem[8]:

$$\min_{\alpha} f(\alpha) = \frac{1}{2}\alpha^T A\alpha - \alpha^T b \tag{27}$$

subject to

$$\alpha^T y = 0$$
$$\mathrm{l} \leq \alpha \leq u$$

In the bound constrained AL technique, a quadratic penalty is added to the Lagrangian function. So that the AL is given by

$$L(\alpha, \lambda, \mu) = f(\alpha) + \lambda y^T\alpha + \frac{\mu}{2}(y^T\alpha)^2 \tag{28}$$

where λ is the Lagrange multiplier and $\mu > 0$ is a penalty parameter. A Bound constrained Augmented Lagrangian algorithm[8] is as follows:

given $\mu_0 \geq 0$ and initial points α^0 and λ_0 set k=0.
1. Use projected Newton to find the approximate minimizer α^{k+1} of $L(\alpha, \lambda_k, \mu_k)$ in subject to $l \leq \alpha \leq u$ with starting point α^k.
2. If $|y^T\alpha^{k+1}| < 10^{-6}$, stop with approximate solution α^{k+1}.
3. Update the Lagrange multiplier by $\lambda_{k+1} = \lambda_k + \mu_k y^T\alpha^k$ and choose a new penalty parameter μ_{k+1} and return to step 1.

This algorithm is taken further [9] by including the constraints into the AL function using nonzero slack variables s_i and t_i satisfying $l_i - \alpha_i + s_i^2 = 0$ and $\alpha_i - u_i + t_i^2 = 0$, so that this can be written as

$$\min_{\alpha} f(\alpha) = \frac{1}{2}\alpha^T A\alpha - \alpha^T b \tag{29}$$

subject to

$$y'\alpha = 0$$
$$\mathrm{l}_i - \alpha_i + s_i^2 = 0$$
$$\alpha_i - u_i + t_i^2 = 0$$

The complete derivations are too long for the purpose of this project. The reader is referred to the primary publication [9]. Next, all the constraints are incorporated into the augmented Lagrangian and after minimizing L with respect to the slack variables, the final form of the Augmented Lagrangian form can be written as minimizing:

$$\begin{aligned} \mathrm{L}(\alpha, \lambda_k, \gamma^k, \mu_k) &= f(\alpha) + \lambda_k y^T \alpha + \tfrac{\mu}{2}(y^T\alpha)^2 \\ &+ (1/(2\mu_k)) \textstyle\sum_{i=1}^{l} [\max 0, \gamma_{l_i}^k + \mu_k (l_i - \alpha_i)^2] - (\gamma_{l_i}^k)^2] \\ &+(1/(2\mu_k))(\textstyle\sum_{i=1}^{l} [\max 0, u_{l_i}^k + \mu_k(\alpha_i - u_i)^2] - (\gamma_{u_i}^k)^2]) \end{aligned} \tag{30}$$

and iterate with updates:

$$\begin{aligned} \lambda_{k+1} &= \lambda_k + \mu_k y^T \alpha^T k \\ \mu_{k+1} &= \beta\mu_k \quad \text{if} \quad |y^T\alpha^k| > v|y'\alpha^{k-1}| \\ \mu_{k+1} &= \mu_k \quad \text{if} \quad |y^T\alpha^k| \leq v|y^T\alpha^{k-1}| \\ \gamma_{l_i}^{k+1} &= \max 0, \gamma_{l_i}^k + \mu_k(l_i - \alpha_i^k), \\ \gamma_{u_i}^{k+1} &= \max 0, \gamma_{u_i}^k + \mu_k(\alpha_i^k - u_i), \end{aligned} \tag{31}$$

Indicator functions can be used to approximate the max functions. Overall this gives the following pseudo-code [9]:

given $\mu_0 0$ and initial points α^0 and λ_0 set k=0.
1. Fix indicator matrices using α_k , then use conjugate gradient to find approximate minimizer α_{k+1} of $L_k(\alpha, \lambda_k, \gamma_k, \mu_k)$ with starting point α_k.
2. If constraints are met, stop with stationary point α_{k+1}.
3.Update Lagrange multipliers and penalty parameter.
4. set $k = k + 1$.

Unconstrained augmented Lagrangian (AL) is a technique for solving the bound and equality constrained quadratic program that arise in SVM classifications problem. The term unconstrained was used by the authors since the algorithm requires only the use of the conjugate gradient method (CG) [9]. A possible benefit of this algorithm is using CG or even a direct solver such as Gaussian elimination for small-scale problems, can be used to compute its minimiser, resulting in the updated approximation α_{k+1}. Another main reason for choosing this algorithm is its significant difference from SMO for the purpose of the study.

2.5 Experiments

2.5.1 Objectives

Our simulation study involves the binary classification of a synthetic non-linear data using two different kernels, Gaussian and Laplacian kernel. Each of the studies are solved by using SMO and the ALM optimization method described previously. We will investigate the accuracy of each method and the convergence rates of each optimization algorithm. We will then assess whether our models generalize using the MNIST data.

2.5.2 Data

For the first data set, we chose synthetically generated 400 random data points, and half were allocated the label $+1$ clustered in the middle of the surface forming a circle like structure and the other half of label -1 clustered around the circle made by positive labels. Therefore, this is a simple non-linear data-set suitable for assessing and correcting

the performance of the algorithms.
For assessing the generalization of the models, we used the MNIST data set[10]. This is a very large data set of hand-written digits usually used for image processing purposes. It includes 60000 training data and 10000 testing data.

2.5.3 Binary classification results

We used the classifiers for the synthetic data as explained before. For tuning the hyper parameters, we used a trial and error approach, where few candidate variables were used and the best result was qualitatively chosen. The regularization parameter (C) was kept at value 1. The γ of the Gaussian kernel was assigned the value 3. The σ of the Laplace kernel was set at 0.1. The tolerance for the convergence part was 10^{-12}.
The results with the margins are visualized in Figure(1). The average CPU cost for each iteration was almost similar in every simulation and each algorithm, It was about $2s$ per each iteration.

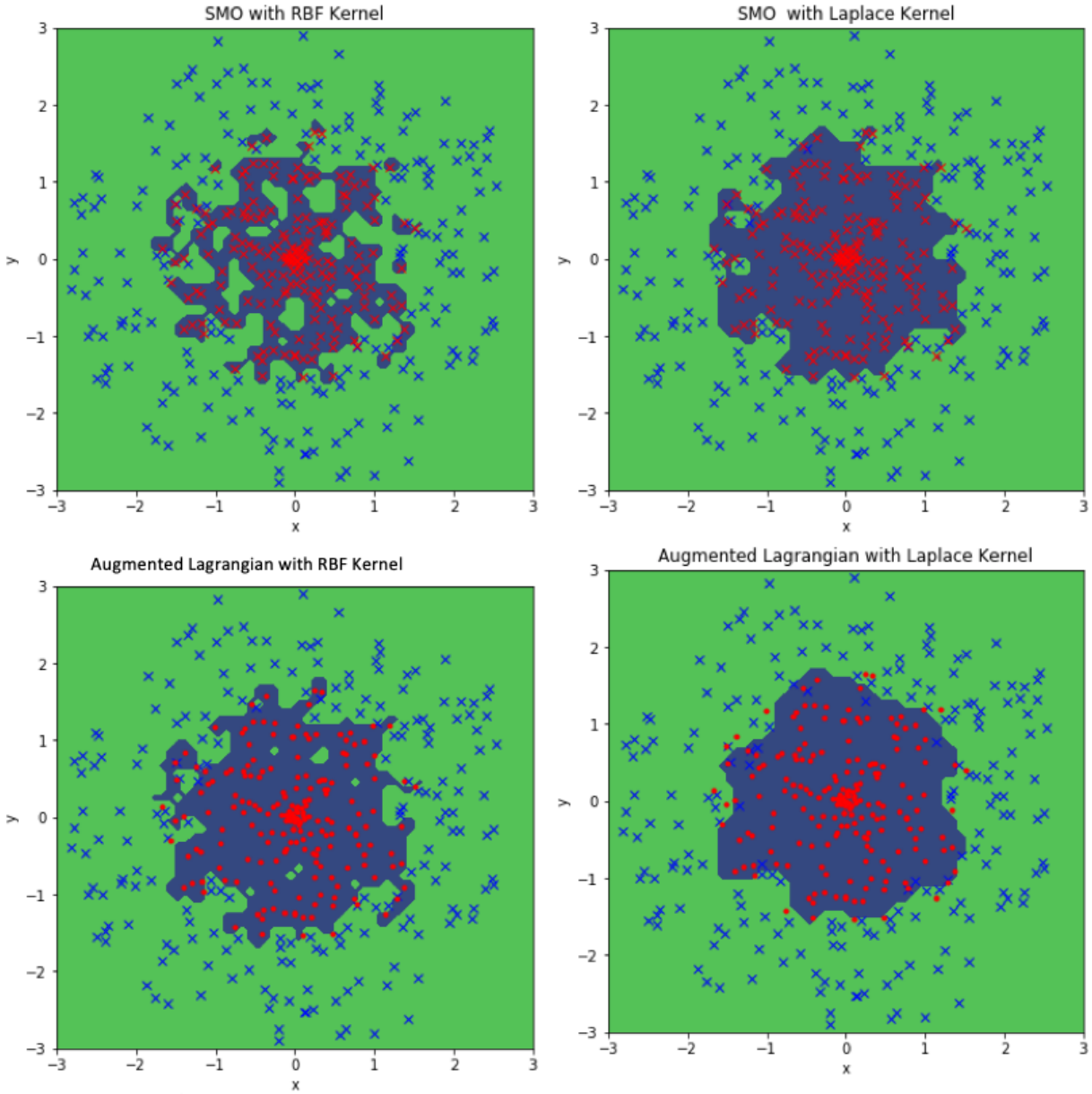

Figure 1. Visualization of the SVM results and margin boundaries
SMO with the RBF and Laplace kernel(top). ALM with the RBF and Laplace kernel(bottom).
Negative classes are red and Positive classes are blue.

We then investigated the convergence rates of the both optimization methods for the kernels. For the SMO method, the error was taken as the relative difference between the Lagrange multipliers after each iteration. For the ALM method, the error was the relative difference between the residuals after each iterations. The results are shown in Figure(2) and will be discussed in the Discussion section.

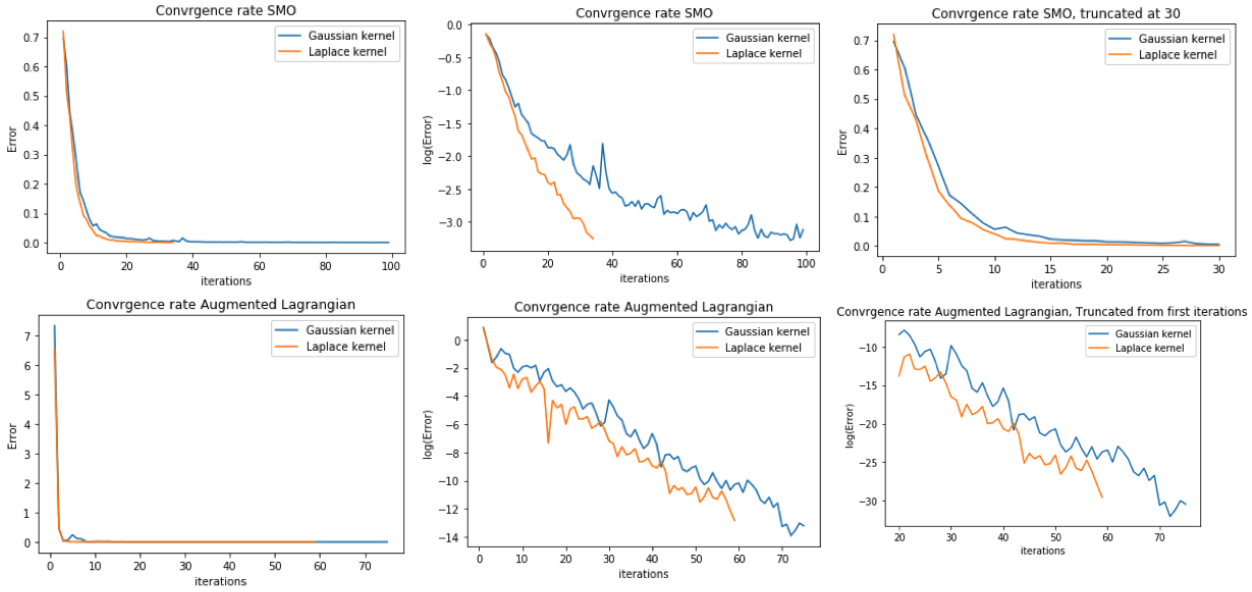

Figure 2. Convergence analysis of SMO and ALM
For SMO, the KKT gap was taken as the error and plotted against iteration counts (top).
For ALM, the relative residual difference was taken and plotted against the iteration counts (bottom).

2.5.4 Generalization on MNIST

Next we tested the generalization of our models using MNIST data. We used a binary classification task on MNIST, where one number (e.g. 8) is given a positive label and all others are given a negative label.We used the same hyper parameters as the ones we used in the previous section on synthetic data, but the tolerance was changed to 10^{-8} for speeding up the training due to the time constraints we were dealing with. We trained the model on 80% of the training data and then tested the accuracy of the module for predicting unseen examples on the remaining 20% data. It should be noted that MNIST has a testing data, however, since we're not bench marking our results against some community, we didn't deem this necessary. The result was that for the SMO: Gaussian kernel took 8 iterations and had a 98% accuracy and the Laplace kernel took 14 iterations and had a 97% accuracy. For the ALM, the Gaussian kernel took 10 iterations and had a 95% accuracy, the Laplace kernel took 15 iterations and had a 95% accuracy.

2.6 Discussion

In this study, we used two kernalized SVMs, Gaussian and Laplace, to classify a synthetic non-linear data with two different algorithms, SMO and ALM. As our results show, the models were able to correctly do the non-linear classifications, figure(1). The Gaussian kernel had better margins than Laplace kernel with each of the optimization algorithms. Moreover, the models trained using SMO optimization had better but not significantly different margins than the ones optimized by ALM. This can be explained by considering that the Fourier transform of the Laplace kernel is a Cauchy distribution which drops slower than the exponential function in the Fourier transform of a Gaussian kernel [5]. This resulted in the the function given by the Gaussian being "smoother" than that given by the Laplace kernel.

As our convergence results show, SMO for the Laplace kernel converges much faster than Gaussian kernel, 35 vs 100 iterations respectively. The convergence rate is super-linear for both methods, but after the first 20 iterations, the Gaussian kernel starts to show just linear convergence. The result for the SMO is what we expected. In order to get even better convergence, we could perhaps increase the value of C, but this would increase the training time significantly. The ALM method, is not as well studied as SMO, but this result could perhaps act as a weak proof of concept for it.

For ALM, the Laplace kernel converges faster than Gaussian kernel, 59 vs 75 iterations. the convergence rate is

super-linear for both kernels. ALM optimization is faster than SMO when using a Gaussian kernel, but slower when using a Laplace kernel. It should be noted that the average CPU cost of each iteration was almost identical in each study with each of the algorithms, so this didn't affect the results.
And lastly we investigated the generalization of our models. They all performed really well on unseen data. The results were comparable with more modern neural networks approaches for classifying MNIST data. The Gaussian kernel gave the best accuracy and the SMO was faster than ALM in this instance. However, for a more accurate analysis of this, there needs to be be a multi-class classification. This could be done using a one vs one or one vs all strategy [1]. Since this could take days to run, we didn't consider it in this experiment.

References

[1] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, January 2006.

[2] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, May 2019.

[3] Andrew Ng. Cs229 lecture notes (part v), 2016.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[6] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[7] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, April 1998.

[8] Dimitri P Bertsekas. Projected newton methods for optimization problems with simple constraints. *SIAM Journal on control and Optimization*, 20(2):221–246, 1982.

[9] Marylesa Howard. Computational methods for support vector machine classification and large-scale kalman filtering. 2013.

[10] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.