S21CSE471: Statistical Methods in AI **Spring 2021**

Support vector machines

Prepared by: Team 12 - Pratik Nashine, Prakhar Tripathi, Ashutosh Ranjan *4th Feb,2021*

In this Document we are going to discuss about the Lagrange Multiplier, KKT Conditions, Noise in SVM, Non-Linear SVM and Kernels in SVM

1 Intoduction

A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

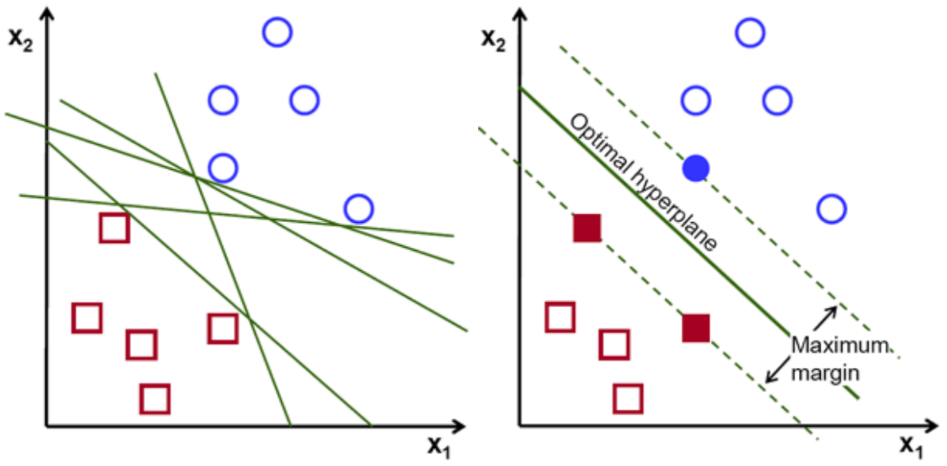

Figure 1: Possible Hyperplanes

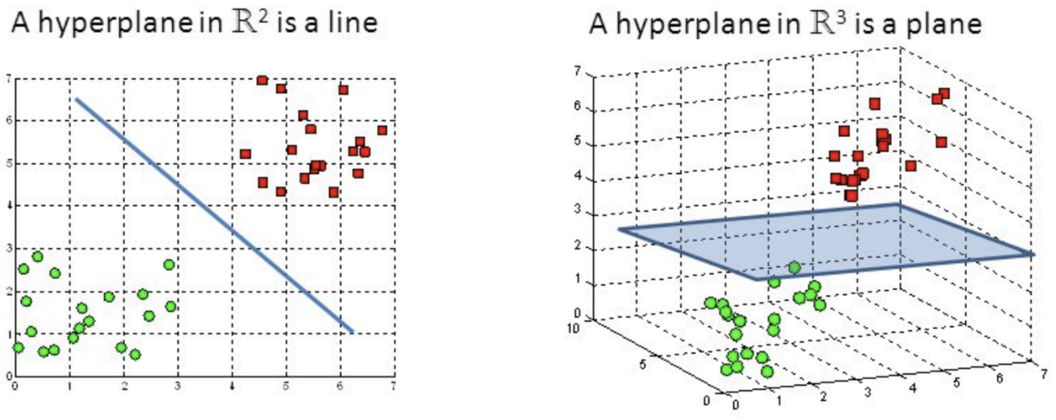

Figure 2: Hyperplanes in Dimensions

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

2 Langrange Multiplier

The Method of Lagrange Multipliers is a useful way to determine the minimum or maximum of a surface subject to a constraint. It is an alternative to the method of substitution and works particularly well for non-linear constraints. Here, all surfaces will be denoted as f (x, y) and all constraints as $g(x, y) = c$.

Let f and g be functions of two variables with continuous partial derivatives at every point of some open set containing the smooth curve $g\left(x, y\right) 0$. Suppose that f, when restricted to points on the curve $g\left(x, y\right) = 0$, has a local extremum at the point $\left(x_0, y_0\right)$ and that $\nabla g\left(x_0, y_0\right) \neq 0$. Then there is a number λ called a Lagrange multiplier, for which

$$\nabla f\left(x_0, y_0\right) = \lambda \nabla g\left(x_0, y_0\right).$$

2.1 Steps for Lagrange Multiplier

1. Given a multivariable function f (x, y) and a constraint g(x, y) = c , define the Lagrange function to be $L(x, y) = f(x, y) - \lambda(g(x, y)c)$, where λ(lambda) is multiplied (distributed) through the constraint portion.
2. Determine the partial derivatives L_x and L_y .
3. Set the partial derivatives L_x and L_y equal to zero.
4. Make note of any immediate solutions for x and y that satisfy $= 0$ L_x and $L_y = 0$. Often there are none, so proceed to step 5
5. Isolate the λ in each equation.
6. Equate the two λ equations, dropping out the λ altogether.
7. Substitute this equation into the constraint and algebraically solve for the variable that remains. This is where the critical points start to be determined.
8. Solve for the other variable by re-evaluating your results in step 8 back into the constraint.
9. Reduce the equation from step 6 as far as possible. This is the relationship between x and y that shall be substituted into the constraint.

2.2 Multiple constraints

The same technique allows us to solve problems with more than one constraint by introducing more than one Lagrange multiplier. For example, if we want to minimize $x^2 + y^2 + z^2$ subject to
x + y - 2 = 0
x + z - 2 = 0
we can write the Lagrangian
$L(x, y, z, p, q) = x^2 + y^2 + z^2 + p(x + y - 2) + q(x + z - 2)$
and the corresponding equations
$0 = \nabla_x L = 2x + p + q$
$0 = \nabla_y L = 2y + p$

$0 = \nabla_z L = 2z + q$
$0 = \nabla_p L = x + y - 2$
$0 = \nabla_q L = x + z - 2$
The solution to the above equations is

$\{p = \frac{-4}{3}, q = \frac{-4}{3}, x = \frac{4}{3}, y = \frac{2}{3}, z = \frac{2}{3}\}$

Here are the two constraints, together with a level surface of the objective function. Neither constraint is tangent to the level surface; instead, the normal to the level surface is a linear combination of the normals to the two constraint surfaces (with coefficients p and q).

3 KKT Conditions

The necessary conditions for a constrained local optimum are called the Karush Kuhn Tucker (KKT) Conditions, and these conditions play a very important role in constrained optimization theory and algorithm development. For an optimization problem:

$$\begin{gathered}\min_x f(x) \\ \text{subject to} \quad g_i(x) - b_i \geq 0 \quad i = 1, \ldots, k \\ g_i(x) - b_i = 0 \quad i = k+1, \ldots, m\end{gathered}$$

There are four KKT conditions for optimal primal (x) and dual (λ) variables. The asterisk (*) denotes optimal values.

3.1 Constrained Conditions

3.1.1 Feasible Constraints

$$g_i(x^*) - b_i \text{ is feasible}$$

3.1.2 No Feasible Descent

$$\nabla f(x^*) - \sum_{i=1}^m \lambda_i^* \nabla g_i(x^*) = 0$$

3.1.3 Complementary Slackness

$$\lambda_i^* \left(g_i(x^*) - b_i\right) = 0$$

3.1.4 Positive Lagrange Multipliers

$$\lambda_i^* \geq 0$$

The feasibility condition (2.1.1) applies to both equality and inequality constraints and is simply a statement that the constraints must not be violated at optimal conditions. The gradient condition (2.1.2) ensures that there is no feasible direction that could potentially improve the objective function. The last two conditions (2.1.3 and 2.1.4) are only required with inequality constraints and enforce a positive Lagrange multiplier when the constraint is active (=0) and a zero Lagrange multiplier when the constraint is inactive (¿0).

There are two possibilities for the optimal solution: it can occur either on the boundary of the feasible set (where $g = 0$) or on the interior (where $g < 0$). If it occurs on the boundary, then we are left with the equivalent of an equality constraint, in which case the simple method of Lagrange multipliers applies. The preceding stationarity condition is identical to the one for Lagrange multipliers, and it captures instances for which either $\nabla = 0$ (meaning that becomes flat on the boundary) or $\nabla f \propto \nabla g$ (meaning that the level sets of f and g lie tangent to each other). The first case forces $\mu = 0$, while the second forces $\mu \neq 0$.

If, on the other hand, the optimum occurs on the interior, then the constraint has no effect on the calculation of the optimum. This can only occur where $\nabla f = 0$, but the stationarity condition guarantees simply that at least one out of $\nabla f = 0$ and $\nabla f \propto \nabla g$ and is true, with no guarantees about which one holds. Additional conditions are needed to ensure that $\nabla f = 0$ for any candidate optimum on the interior.

4 NOISE IN SVM data

4.1 Label Noise Robust SVM

It is possible to directly model label noise by assuming that the labels in the training set

$$(x_i, y_i)_{i=1}^{n} \in X$$

multiplied by the set of [-1,+1]. In the dual SVM problem the class labels affects the matrix

$$Q = Kyy^t$$

When we consider the label noise the equation becomes

$$Q_(i,j) = y_i y_j K(x_i x_j)(1-2e_i)(1-2e_j)$$

If we assume that any label with the same probability is flipped independently, then

$$\in_i, i = 1, 2..........n$$

are in i.i.d.Boolean random variables whose mean is simply the probability of e at i = 1. With this assumption the expected value of Q becomes

$$\mathbb{E}[Q_(i,j)] = y_i y_j K(x_i x_j)(1-4\sigma^2), if i \neg j$$

$$\mathbb{E}[Q_(i,j)] = y_i y_j K(x_i x_j), otherwise$$

Now, to solve the SVM problem, we can use the expected value of Q(which is still a positive semi-definite kernel matrix). This should enhance the robustness of the learned SVM to mark flip noise fairly.
The solution is symmetric with respect to mean = 0.5 i.e alpha values obtained for

$$\mu = \mu^* and$$

$$\mu = 1 - \mu^*$$

is same, as the standard SVM solution when mean is either 0 or 1.Moreover, the equations of w and b obtained by solving the standard SVM problem have to be multiplied by 1(2*mean). Thus whenever the mean is greater than 0.5, then w and b are multiplied by a negative factor. This represents that more than half of the training examples are treated wrong and thus the decision boundary is inverted.

Due to this complication SVM is not able to decide the best hyperplane. Now we would see how to tune the hyperparameters so that we can remove the noise from the SVM data and make SVM work for the noisy data too.

4.1.1 Dual Problem and alpha equalization

Now we are in a position to better examine the shift caused by kernel correction in the Dual Problem SVM.The dual problem can,therefore, be re-written as

$$\min_{\alpha} \frac{1}{2}\alpha^\top(\mathbf{Q} \circ \mathbf{M})\alpha - \mathbf{1}^\top\alpha$$

$$s.t. \quad 0 \le \alpha_i \le C, i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Now the elements are written as

$$M_{ij} = \begin{cases} 1, & \text{if } i = j \\ 1 - S, & \text{otherwise} \end{cases}$$

After applying some Lagrangian modifications the new equation becomes:

$$\min_{\alpha} \frac{1}{2}\alpha^\top \mathbf{Q}\alpha - 1^\top\alpha + \frac{S}{1-S}\left[\frac{1}{2}\alpha^\top\left(\mathbf{Q} \circ \mathbb{I}_{n\times n}\right)\alpha - 1^\top\alpha\right]$$

$$s.t. \quad 0 \le \alpha_i \le C, i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

where the only difference with the standard SVM formulation is given by an additional term weighted by S/(1 - S). We will denote this extra term by x. So now when the mean increases from 0 to 0.5 x approaches infinity. Also now this does not depends on the class labels and only depends upon alpha and the diagonal of the Q. Due to this our solution will only be dependent on the diagonal of the Kernel matrix which won't be affected by the noise. This effect. Note also that this is not equivalent to increase the diagonal of the kernel matrix, as commonly done for improving numerical stability. The effect of the proposed correction is instead to reduce the impact of the out-of-diagonal elements of the kernel matrix, which are the only ones affected by label noise.

5 Non-Linear SVM

In the graph below, we notice that there are two classes of observations: the blue points and the purple points. There are tons of ways to separate these two classes as shown in the graph on the left. However, we want to find the "best" hyperplane that could maximize the margin between these two classes, which means that the distance between the hyperplane and the nearest data points on each side is the largest. Depending on which side of the hyperplane a new data point locates, we could assign a class to the new observation.

It sounds simple in the above example. However, not all data are linearly separable. In fact, in the real world, almost all the data are randomly distributed, which makes it hard to separate different classes linearly.

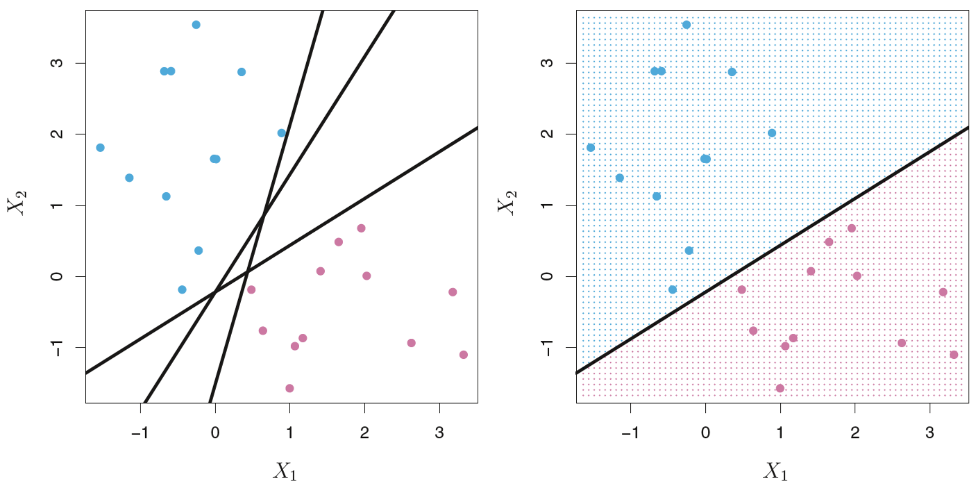

Figure 3: Linearly Separable Data

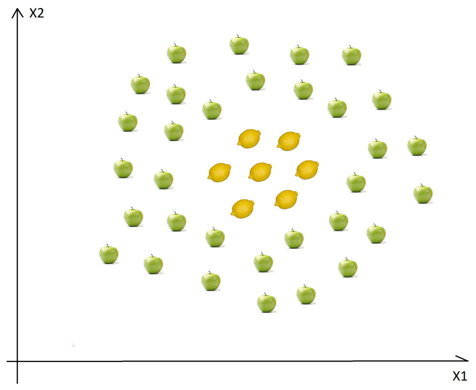

Figure 4: Linearly Inseparable data

The basic idea is that when a data set is inseparable in the current dimensions, add another dimension, maybe that way the data will be separable. Just think about it, the example above is in 2D and it is inseparable, but maybe in 3D there is a gap between the apples and the lemons, maybe there is a level difference, so lemons are on level one and lemons are on level two. In this case we can easily draw a separating hyperplane (in 3D a hyperplane is a plane) between level 1 and 2.

Let's assume that we add another dimension called Z. Another important transformation is that in the new dimension the points are organized using this formula $X1^2 + X2^2$.If we just plot the Z plane we get the following plot

Now if we consider that the origin is the lemon from the center, we will have plot like this:

Similarly we can transform 1-d data to 2-d using $y = x^2$ as a second dimension

Transforming the Dimension helps us with our decision of finding the separating hyperplane however we are faced with the difficulty of increased number of operations and complexity due to higher dimension

Let's look at an example:

$$x = (x_1, x_2, x_3)^T \tag{1}$$
$$y = (y_1, y_2, y_3)^T \tag{2}$$

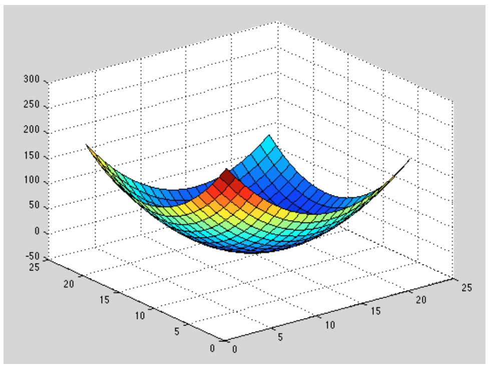

Figure 5: Z plane

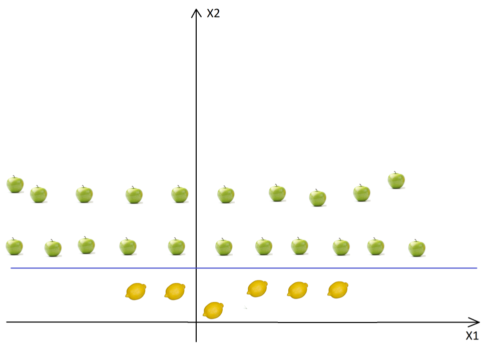

Figure 6: Transformed data

Now if we want to map above data points to 9-D space for some calculations to get a final scalar result the computational complexity in this case is as high as: $O(n^2)$

$$\phi(x) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T \tag{3}$$

$$\phi(y) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2) \tag{4}$$

The Scalar we get as

$$\phi(x)^T\phi(y) = \sum_{i,j=1}^{3} x_ix_jy_iy_j \tag{5}$$

This leads us to our SVM Post Mapping For Training as:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_jd_id_j\phi(x_i)^T\phi(x_j) \tag{6}$$

$$Subject\,to \quad \alpha_i \geq 0 \quad \forall i \quad and \quad \sum_{i=1}^{N} \alpha_id_i = 0$$

and For testing as:

$$Label = sign(\sum_{i=1}^{N}(\alpha_id_i\phi(x_i)^T\phi(x_{test}) \; + \; b_0)) \tag{7}$$

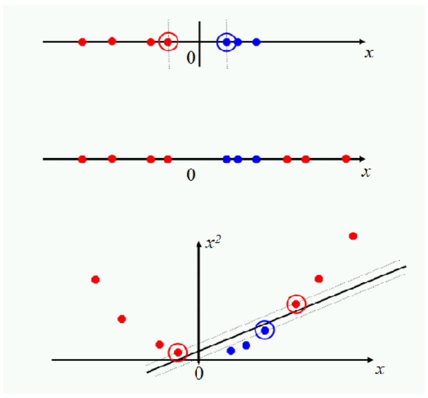

Figure 7: 1-D to 2-D transformation of data

6 Kernel SVM

In order to deal with such huge computations we are going to use a ***Kernel*** function, which is denoted as ***k(x,y)***, instead of doing the complicated computations in the 9-dimensional space, we reach the same result within the 3-dimensional space by calculating the dot product of x^T and y. The computational complexity, in this case, is $O(n)$. Any computations involving the dot products (x, y) can utilize the kernel trick.

$$\begin{aligned} k(x,y) &= (x^T y)^2 \\ &= (x_1y_1 + x_2y_2 + x_3y_3)^2 \\ &= \sum_{i,j=1}^{3} x_i x_j y_i y_j \end{aligned}$$

Which leads us to Modified SVM Equation with Kernel implementation as:

$$For \quad Training \quad : \quad Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(x_i, x_j) \tag{8}$$

$$Subject\,to \quad \alpha_i \geq 0 \quad \forall i \quad and \quad \sum_{i=1}^{N} \alpha_i d_i = 0$$

$$For \quad Testing \quad : \quad Label = sign(\sum_{i=1}^{N} (\alpha_i d_i K(x_i, x_{test}) \; + \; b_0)) \tag{9}$$

6.1 Kernel Function Types

Some Popular Kernel functions are:

1. **Linear Kernel** The Linear kernel is the simplest kernel function. It is given by the inner product ¡x,y¿ plus an optional constant c. Kernel algorithms using a linear kernel are often equivalent to

their non-kernel counterparts

$$k(x, y) = x^T y + c \tag{10}$$

2. **Ploynomial Kernel** The Polynomial kernel is a non-stationary kernel. Polynomial kernels are well suited for problems where all the training data is normalized

$$k(x, y) = (\alpha x^T y + c)^d \tag{11}$$

3. **Gaussian Kernel** The Gaussian kernel is an example of radial basis function kernel

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right) \tag{12}$$

   or also as

$$k(x, y) = \exp\left(-\gamma ||x - y||^2\right) \tag{13}$$

   The adjustable parameter σ plays a major role in the performance of the kernel, and should be carefully tuned to the problem at hand. If overestimated, the exponential will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. In the other hand, if underestimated, the function will lack regularization and the decision boundary will be highly sensitive to noise in training data.

4. **Exponential Kernel** The exponential kernel is closely related to the Gaussian kernel, with only the square of the norm left out. It is also a radial basis function kernel.

$$k(x, y) = \exp\left(-\frac{||x - y||}{2\sigma^2}\right) \tag{14}$$

5. **Laplacian Kernel** The Laplace Kernel is completely equivalent to the exponential kernel, except for being less sensitive for changes in the sigma parameter. Being equivalent, it is also a radial basis function kernel.

$$k(x, y) = \exp\left(-\frac{||x - y||}{\sigma}\right) \tag{15}$$

6. **Hyperbolic Tangent(Sigmoid) Kernel** The Hyperbolic Tangent Kernel is also known as the Sigmoid Kernel and as the Multilayer Perceptron (MLP) kernel. The Sigmoid Kernel comes from the Neural Networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons. A SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network. This kernel was quite popular for support vector machines due to its origin from neural network theory. Also, despite being only conditionally positive definite, it has been found to perform well in practice.

$$k(x, y) = \tanh(\alpha x^T y + c) \tag{16}$$

   There are two adjustable parameters in the sigmoid kernel, the slope α and the intercept constant c. A common value for α is $\frac{1}{N}$, where N is the data dimension.

7. **Anova Kernel** The ANOVA kernel is also a radial basis function kernel, just as the Gaussian and Laplacian kernels. It is said to perform well in multidimensional regression problems.

$$k(x,y) = \sum_{k=1}^{n} \exp(-\sigma(x^k - y^k)^2)^d \tag{17}$$

References

[1] https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[2] https://www.researchgate.net/publication/230663810_Const raint_Qualifications_for_Nonlinear_Programming

[3] https://www.researchgate.net/publication/262378087_Support_Vector_Machi nes_applied_to_noisy_data_classification_using_differential_evolution_with_local_search

[4] https://www.sciencedirect.com/science/article/abs/pii/S0167865503002812

[5] https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d

[6] http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/