Statistical Techniques in Robotics (16-831, S22) Lecture #10 (Monday, February 21)

Support Vector Machine

Lecturer: Kris Kitani *Scribe: Roshini Rajesh Kannan*

1 Review

In the course so far, we have seen Prediction with Expert Advice (PWEA) algorithms that had the characteristics of one-shot, instructive and exhaustive data. Later we moved to Online Linear Classification that included perceptron and winnowing algorithms. From there we moved to Online Convex Optimization problems that included Follow the Leader, Follow the regularized leader and we also connected it to Online Mirror Descent. Now in this lecture, we will be seeing (Online) Supervised Learning and how some supervised learning algorithms like Support Vector Machine can be solved using online learning algorithms.

1.1 Gradient descent

Last lecture we saw the concept of Gradient Descent and Stochastic Gradient Descent. It further explained how Online Gradient Descent is a special case of Online Mirror Descent (OMD). In the review will cover Gradient Descent and its links with OMD. Gradient Descent is an approach for minimizing differentiable (or non-differentiable in case of sub gradient descent) convex functions with regularization as an added constraint. From gradient descent we get updates in the form of

$$\mathbf{w} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)}) \tag{1}$$

This form of the parameter update is a result of:

1. A quadratic approximation of the loss function f
2. Linear loss function f with quadratic regularization
3. Quadratic loss function with linear constraints

The figure 1 shows the geometric interpretation of gradient descent. We can approximate this to speed up the computation through Stochastic Gradient Descent(SGD) or Online Gradient Descent(OGD)

1.2 Online Gradient Descent(OGD)

Online Mirror Descent is often used to solve optimization problems that contain a linear loss function $f(\mathbf{w})$ and a quadratic regularization $\psi(\mathbf{w})$. In case of OGD, we can write them as:

$$\psi(\mathbf{w}) = \frac{1}{2\eta}||\mathbf{w}||_2^2 \text{ and } f(w) = \langle \mathbf{w}, \theta \rangle \tag{2}$$

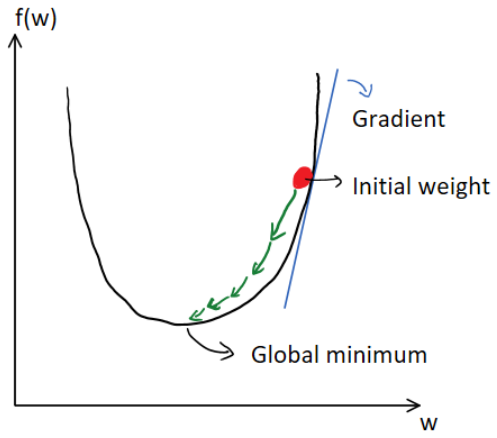

Figure 1: Geometric representation of gradient descent

Where θ is the parameter for dual space. From equation 2 we can see that $f(\mathbf{w})$ is linear and $\psi(\mathbf{w})$ is quadratic for OGD. Further when we solve for optimal parameter to minimize the convex function, we notice that $\mathbf{w}_n = \eta\theta$. Hence the mirror function for OGD is $g(\theta) = \eta\theta$. Therefore OGD is OMD with a linear loss and quadratic regularization (or quadratic loss with linear constraints).

2 Summary

2.1 Online learning and Supervised Learning

Online learning mainly has 2 steps: Update and Prediction. After the prediction, it calculates the loss and repeats the same process again. In supervised learning, the two steps happen distinctly and there is no continuous interaction between both. The parameters are fixed before testing and are not updated regularly. In figure 2, When we look closely, we can see that, the learning algorithm in the training part of supervised algorithm can be an online learning algorithm. In this way, we see that some supervised learning algorithms can be solved using online algorithms. Now, let us see how online learning algorithms are hidden in supervised learning algorithms like SVM.

2.2 Hyperplanes

Definition 1. Hyperplanes Is a subspace with a dimension one less than the dimension of the space in which it is present.
Before moving ahead to SVM, we need to understand the concept of hyperplanes to know about its geometry.

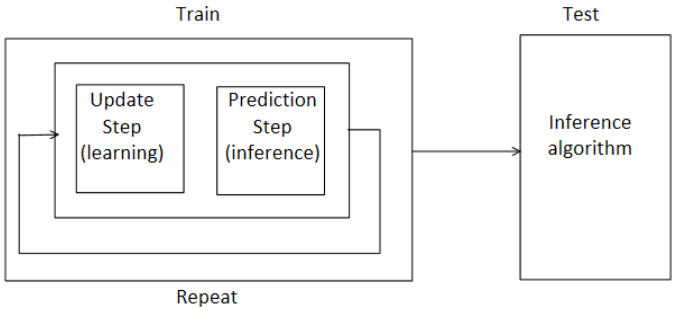

Figure 2: Online learning Vs. Supervised Learning

2.2.1 Hyperplanes in 2D

Hyperplanes are lines in 2-D and can be written as dot product plus a bias.

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{3}$$

The distance between a point (x_0, y_0) and line can be given by the equation

$$\frac{|\mathbf{w} \cdot \mathbf{x} + b|}{||\mathbf{w}||} \tag{4}$$

For the distance between the origin and line, we have to substitute the value of (x_0, y_0) as $(0, 0)$:

$$\frac{b}{||\mathbf{w}||} \tag{5}$$

From equation 5 we get the distance from origin to be $\frac{b}{||\mathbf{w}||}$ also shown in figure 3.
Similarly, we try to find the distance between two lines, where line 1 is $\mathbf{w}.\mathbf{x} + b = 0$ and line 2 is $\mathbf{w}.\mathbf{x} + b = -1$. Then the distance of line 1 to origin would be $\frac{b}{||\mathbf{w}||}$ and the distance of line 2 to origin would be $\frac{b+1}{||\mathbf{w}||}$. The difference between these distances would give the distance between the lines which is $\frac{1}{||\mathbf{w}||}$.

2.2.2 Hyperplanes in 3D

The hyperplanes in 3D also have an equation of $\mathbf{w} \cdot \mathbf{x} + b = 0$, where the dimensions of the $\mathbf{w}$ and $\mathbf{x}$ vectors without bias is the dimension of the plane. Consider we have 3 planes $\mathbf{w} \cdot \mathbf{x} + b = 0$, $\mathbf{w} \cdot \mathbf{x} + b = -1$ and $\mathbf{w} \cdot \mathbf{x} + b = 1$ as shown in figure 4. Then the distance between each of the outer planes to the centre plane is $\frac{1}{||\mathbf{w}||}$. And the distance between the two outer planes is $\frac{2}{||\mathbf{w}||}$

2.3 Support Vector Machine

When we try to find out the best hyperplane for a binary classification case, we might intuitively say it as the one that has the highest margin, i.e, the one which is farthest from all the interior points. In figure 5, the highlighted points are the ones closest to the planes and they are called the support vectors. So mathematically our goal is to find a $\mathbf{w}$ that maximizes the distance between

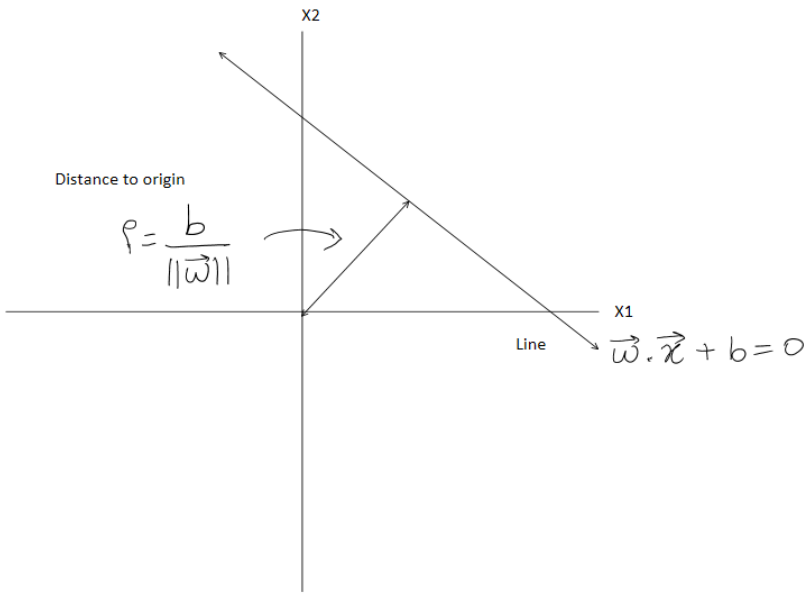

Figure 3: Distance from a line to origin

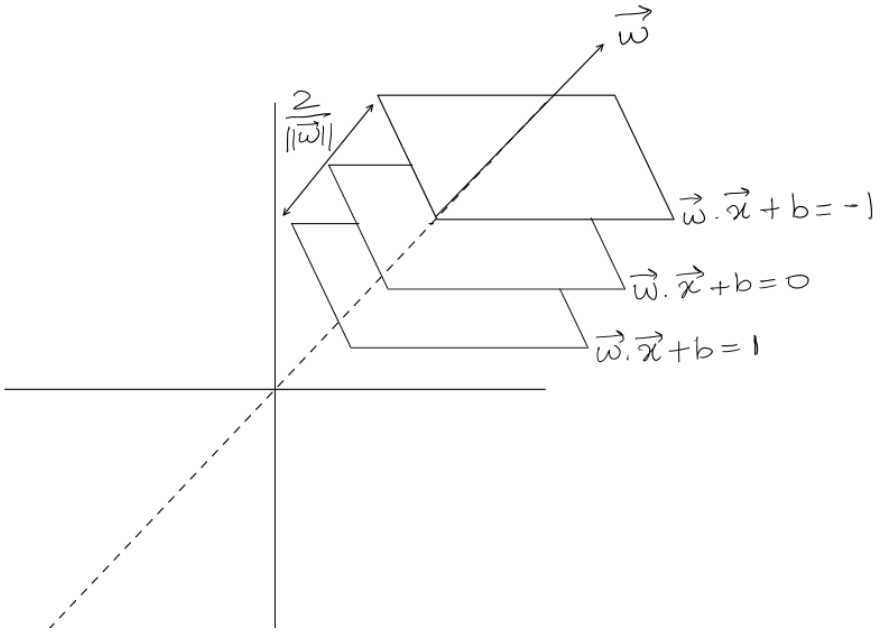

Figure 4: Distance between 2 hyperplanes

the parallel planes which is $\frac{2}{||\mathbf{w}||}$ as shown.

Apart from maximum margin, we also need to classify the points correctly, hence, there are 2

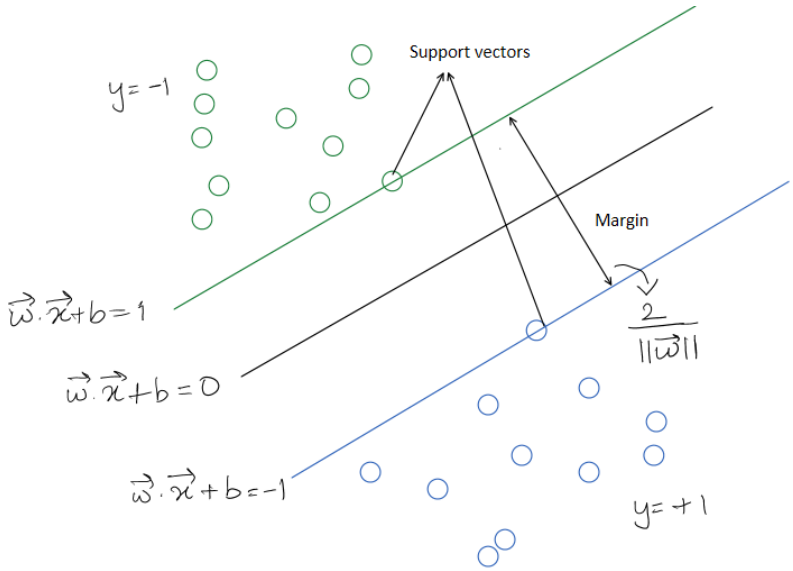

Figure 5: Support Vector

constraints added to make the goal more specific.

$$\max \frac{2}{||\mathbf{w}||} \text{ subject to } \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \text{ if } y_i = +1 \text{ and } \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \text{ if } y_i = -1 \tag{6}$$

The margin here need not necessarily be 1, it can be any constant value. The above maximization problem can also be written as an equivalent minimization problem as:

$$\min ||\mathbf{w}||^2 \text{ subject to the condition } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for i=1,...,N} \tag{7}$$

Finding the arg min $\mathbf{w}$ that minimizes $||\mathbf{w}||$ will be same as the $\mathbf{w}$ that minimizes $||\mathbf{w}||^2$. Hence, we have equation 7 to add more penalty on w. The above equation is the primal formulation of a linear SVM. Since the constraints are satisfied, we call this the Hard SVM.

2.4 Soft margin SVM

Equation 7 in the previous section is the constraint for the objective function of SVM and this is called the hard constraint. We might not be able to satisfy this constraint when the data is too noisy or if it is not linearly separable. This is the case of soft-SVM. If we try to find the best hyperplane in soft SVM that doesn't make any mistake, then the margin might be too narrow. Intuitively it might be better to allowing some misclassifications that might make the model more robust.

By allowing few mistakes we can get a bigger margin. To allow some misclassifications we modify the constraint by adding a slack variable ξ_i. Geometrically the slack variable is the distance of the point from the margin on the correct side the point has to be as shown in figure 6. The slack variable is always positive and we want it to be closer to zero so that the misclassified point is closer to its correct classification side.

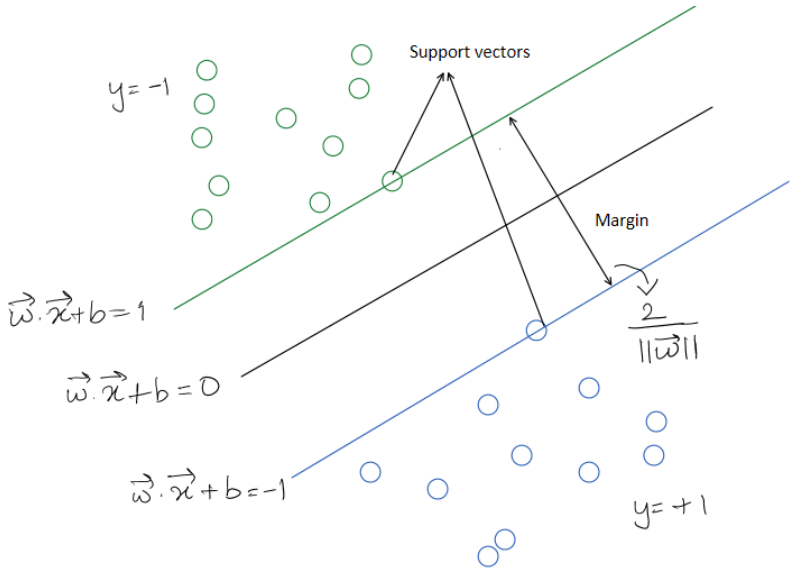

Figure 6: Slack Variable

The objective for soft margin is

$$\min_{\mathbf{w},\xi} ||\mathbf{w}||^2 + C\sum_i \xi_i \tag{8}$$

and it is subject to

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \text{ for i=1,2,...,N} \tag{9}$$

In equation 8, C is a regularization parameter. If C is close to zero, then constraints are ignored and we get a larger margin. When C is close to infinity, there is huge penalty and the margin is small. Let us now solve this problem.
From equation 9 we get,

$$1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) \leq \xi_i \tag{10}$$

By substituting this to the objective function we get,

$$\min_{\mathbf{w}}(\frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_{n=1}^{N} 1 - y_n\mathbf{w}^T\mathbf{x}_n) \tag{11}$$

The drawback with this equation is very large negative value for some correctly classified points overpowers positive values for mistaken points. In order to correct this, we want to penalize for mistakes, weakly correct and ignore 'very correct' data. To achieve this, we replace the term $1 - y_n\mathbf{w}^T\mathbf{x}_n$ with the hinge loss max{0,1 $- y_n\mathbf{w}^T\mathbf{x}_n$}. The function now becomes,

$$\min_{\mathbf{w}}(\frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_{n=1}^{N} max\{0, 1 - y_n\mathbf{w}^T\mathbf{x}_n\} \tag{12}$$

where the first term is the regularisation term and the second term is the loss function. This is a convex function and online mirror descent due to a piecewise linear hinge loss and quadratic regularisation function. This function is not differentiable due to max function in hinge loss. Since it is a convex non-differentiable function, we optimize it using the Online Sub-Gradient Descent. In

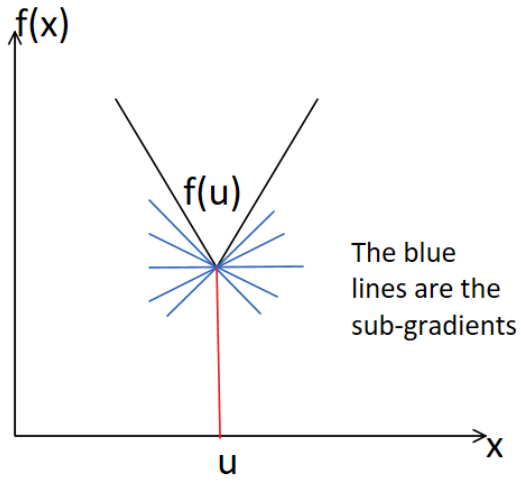

Figure 7: Sub-gradients

figure 7 we can see that when a convex function is non-differentiable, as it has many lines that lower bound the function $f(\mathbf{x})$ at $x = u$. These are the sub-gradients. To optimize our objective function, we will have to get the sub-gradients of the hinge loss. There are many possible sub-gradients, but we select the following two:

$$\mathbf{z}_n = \begin{cases} 0 \text{ if } y_n \mathbf{w}^T \mathbf{x}_n \geq 1 \\ -y_n \mathbf{x}_n \text{ otherwise} \end{cases} \tag{13}$$

The following is the algorithm for soft SVM:

Algorithm 1 Soft SVM

$\theta^{(0)} \leftarrow 0 \in \mathbb{R}^{(N)}$ ▷ Theta initialization
for $t = 1, \cdots, T$ **do**
  $y_d, \mathbf{x}_d \sim \mathrm{D}$ ▷ Receive training data
  $\theta^{(t)} = \theta_{(t-1)} + y_d \mathbf{x}_d . \mathbf{1}[y_d(\mathbf{w}^{(t)} . x^{(y)})] < 1$ ▷ Dual parameter update
  $\mathbf{w}^{(t+1)} \leftarrow \frac{1}{\lambda(t+1)} \theta^{(\mathbf{t})}$ ▷ Weight update
end for

2.5 Comparison with Perceptron algorithm

Algorithm 2 Perceptron Algorithm

$w^{(0)} \leftarrow 0$ ▷ Weight initialization
for $t = 1, \cdots, T$ **do**
  RECEIVE$(\mathbf{x}^{(t)} y^{(t)})$ ▷ Receive training data
  $\theta^{(t)} = \theta_{(t-1)} + y^{(t)} \mathbf{x}^{(t)} . \mathbf{1}[y^{(t)} \langle \mathbf{w}^{(t)}, x^{(t)} \rangle < 0]$ ▷ Dual parameter update
  $\mathbf{w}^{(t+1)} = \theta^{(t)}$ ▷ Mirror projection
end for

Both SVM and Perceptron have similar dual parameter update. SVM uses soft margin whereas, perceptron uses no margin. They both have a similar mirror function. Both have piecewise linear hinge loss and a quadratic regularisation.

3 Appendix

3.1 Multi-class SVM

Handling multiclass data with SVMs is still an active area of research. Methods involve creating multiple SVMs that compare feature vectors among themselves by using various techniques such as one-versus-Rest (OVR) or one-versus-one (OVO). For k classes, the OVR method trains k classifiers so that each class discriminates against the remaining k-1 classes. OVO creates one binary classification problem for all possible pairings of classes, so it requires $\frac{k(k-1)}{2}$ classifiers[1]. After constructing the number of required binary classifiers for either the OVR or OVO methods, the algorithm classifies a new object according the majority vote among the set of classifiers.

3.2 Advantages of SVM

1. Can easily accommodate data with large feature dimensions
2. Require fewer computational resources for both training and classification[3]
3. Less likely to overfit

3.3 Lagrangian Dual Form

The primal quadratic form can become hard to solve with large datasets because the optimization space considers the dot product of w and all the training. Hence, we use a dual form based on Lagrangian multipliers[2]. Some benefits of the dual form are a reduction of computational resources and using Kernel functions to find suitable hyperplanes that can separate non-linearly separable data. The dual problem requires learning only the number of support vectors.
Constructing the dual form involves constructing the Lagrange, by combining both the objective function $f(w)$ and the equality constraint $g(w)$ such that the Lagrangian function $L(w) = f(w)–\lambda g(w)$ with λ being the so-called Lagrangian multiplier. Hence, the objective function

$$f = \frac{1}{2}||w||^2$$

and constraint

$$y_i(w \cdot x_i + b) - 1 = 0$$

gives the dual form of

$$L(\alpha, w, b) = \frac{1}{2}||w||^2 - \sum_1 \alpha_i[(y_i(w \cdot x_i + b) - 1] \tag{14}$$

The maximum (or minimum) solutions of the Lagrangian that satisfies the constraints are where the gradient is zero

References

[1] S. Bhattacharyya. Understanding support vector machine: Part 2: Kernel trick; mercer's theorem, Oct 2020.

[2] R. Bridgelall. Tutorial on support vector machines. 2022.

[3] D. A. Pisner and D. M. Schnyer. Support vector machine. In *Machine learning*, pages 101–121. Elsevier, 2020.